

LEARN
ARDUINO PROJECTS

UNO R4



DETAILED USER GUIDE

WWW.QUADSTORE.IN



Table of Contents

Introduction

1. Introduction to UNO R4 Board
2. Warranty – Register Now!
3. **Need Help? – QuadBot our Smart AI Mentor available 24x7 Support**
4. Understanding Uno R4 board
5. Installing Arduino IDE
6. Connecting the UNO R4 Board
7. Uploading Your First Program (Blinking internal LED Test)

Projects

1. External LED Blink
2. Traffic Lights with 3 LEDs
3. LED control using Push Button
4. Passive Buzzer Sound
5. Active Buzzer Beep
6. RGB LED Color Mixing
7. Potentiometer-controlled LED Brightness (PWM)
8. LDR Night Lamp
9. Tilt Sensor Alert
10. Servo Motor Sweep
11. Ultrasonic Distance Finder
12. DHT11 Weather Monitor
13. Fire Sensor Alarm
14. I2C LCD Display
15. Stepper Motor
16. 1-Digit 7 Segment Display
17. 4-Digit 7 Segment Display
18. Infrared Remote
19. 8x8 Dot Matrix Display

Mini Projects

1. Automated Gate System & Smart Dustbin
2. Digital Locker
3. Smart Weather Station
4. Reaction Timer Game
5. LED Direction Bar

Closing Summary

Introduction to Uno R4 boards (Must Read)

About the UNO R4 Lite/Minima Board

The **UNO R4** board is an upgraded version of the previous UNO R3. There are **2 versions** of the Uno R4 board available which is **Minima** and **Wifi**.

What is difference between UNO R4 Lite & Uno R4 minima?

The board included in Quad Store kit is a compatible Uno R4 Lite which is exact same as the Minima board. Our boards has the exact same processor, larger memory, and uses a USB Type-C connector and expandable I/O ports.

Despite improvements, it is still easy to use for beginners.

So, in short, **Quad Store's Uno R4 Lite = Arduino Uno R4 Minima**.

What is difference between compatible and original board?

- **Arduino boards are open-source**, so anyone can legally manufacture their own version that works exactly like the original.
- A **compatible board** is a board built to work exactly like the original Arduino, following the same pin layout and functionality.
- Compatible boards (like Quad Store's) offer the same functionality but with upgraded PCB quality and durability.
- They often include **extra expansion** options such as additional I/O pads or improved connectors.
- Compatible boards usually provide better value, with longer warranties, rust-proof coating, and stronger local support

Why Quad Store Uno R4 Lite board is better than original Arduino Uno R4 Minima board?

- **Superior PCB** construction for improved signal integrity and durability.
- Additional plated-through holes and **expansion pads** for extra analogue/digital/IO connections — ideal for complex DIY builds.
- Protective **rust-proof coating** (conformal finish) to safeguard against humidity and corrosion (especially useful in Indian environments).
- Standard 3-month **warranty** included; register the product on our website and get an additional 3 months – giving you a full **6-month** peace-of-mind.
- **Enhanced customer support**: we provide quicker responses, dedicated documentation, and local spare-parts availability — beyond the generic support offered by many brands

Warranty: (Register Now!)

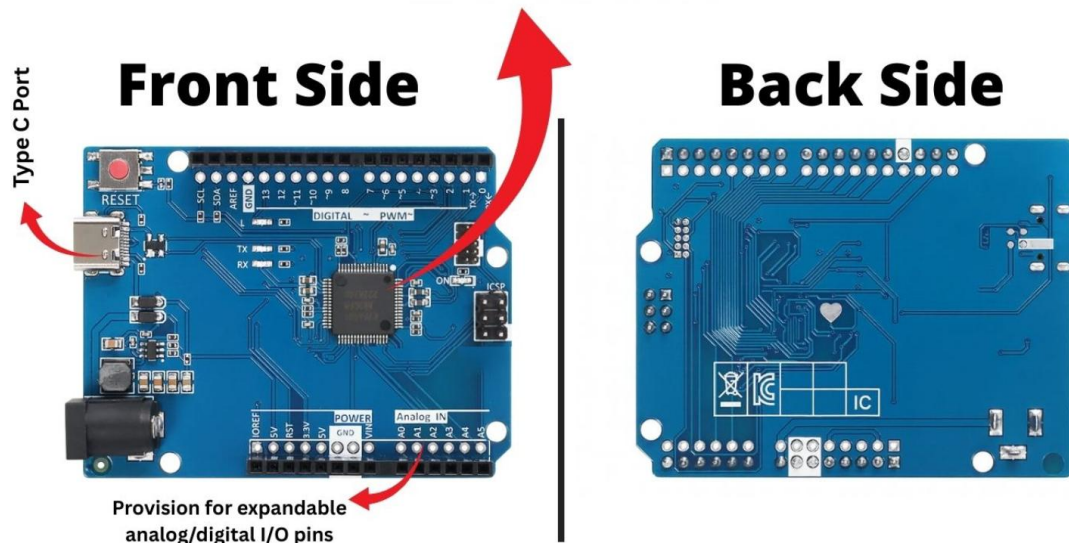
To **claim standard warranty** for Uno R4 Lite boards, **registration** on the Quad Store website is required. We provide a standard **3-month** warranty, and by registering your product online, you receive an additional 3 months—giving you a full **6 months** of peace of mind.

Visit below link and register within **7 days** of purchase of product.

<https://quadstore.in/warranty/>

Quad Store R4-Lite board

**Comes with Renesas RA4M1 series
microcontroller**



Our boards are fully compatible with Arduino IDE

- Model: Renesas RA4M1 (RA4M1F44LBA)
- Core: Arm® Cortex®-M4, 32-bit
- Clock speed: 48 MHz
- Flash memory: 256 KB
- SRAM: 32 KB
- Operating voltage: 5V

Need Help? Ask our Smart AI Mentor:



Meet QuadBot AI Assistant

Your Smart Robotics Learning Assistant

Stuck on a project? Facing an error?

Have a question about robotics or coding?

QuadBot AI Assistant is here to help you anytime!



Hi!
I'm QuadBot.
How can I
help you today?

24/7
AI SUPPORT

QuadBot can help you with:

- Fixing coding errors
- Circuit connection guidance
- Arduino IDE & ESP32 setup help
- Sensor troubleshooting
- Robotics project ideas
- Understanding electronics concepts
- Step-by-step explanations for beginners
- AI-powered learning support for students

GET HELP INSTANTLY!

Scan the QR code or click the link below
to chat with QuadBot AI Assistant.

Scan
me!



OR CLICK THE LINK BELOW



<https://tinyurl.com/QuadBot-ai>

EXAMPLE QUESTIONS YOU CAN ASK

“ Why is my
Arduino not
connecting?
...”

“ How do I
connect the
ultrasonic
sensor?
...”

“ Can you explain
this code for
beginners?
...”

“ My motor is not
rotating. What
should I check?
...”

“ Give me project
ideas using
ESP32.
...”



Learn Faster.
Build Smarter.
With QuadBot!

QuadBot AI Assistant is designed especially for
students, beginners, teachers, and hobbyists to
make robotics learning **easier, faster and**
more exciting!



Trusted Support



24/7 Availability



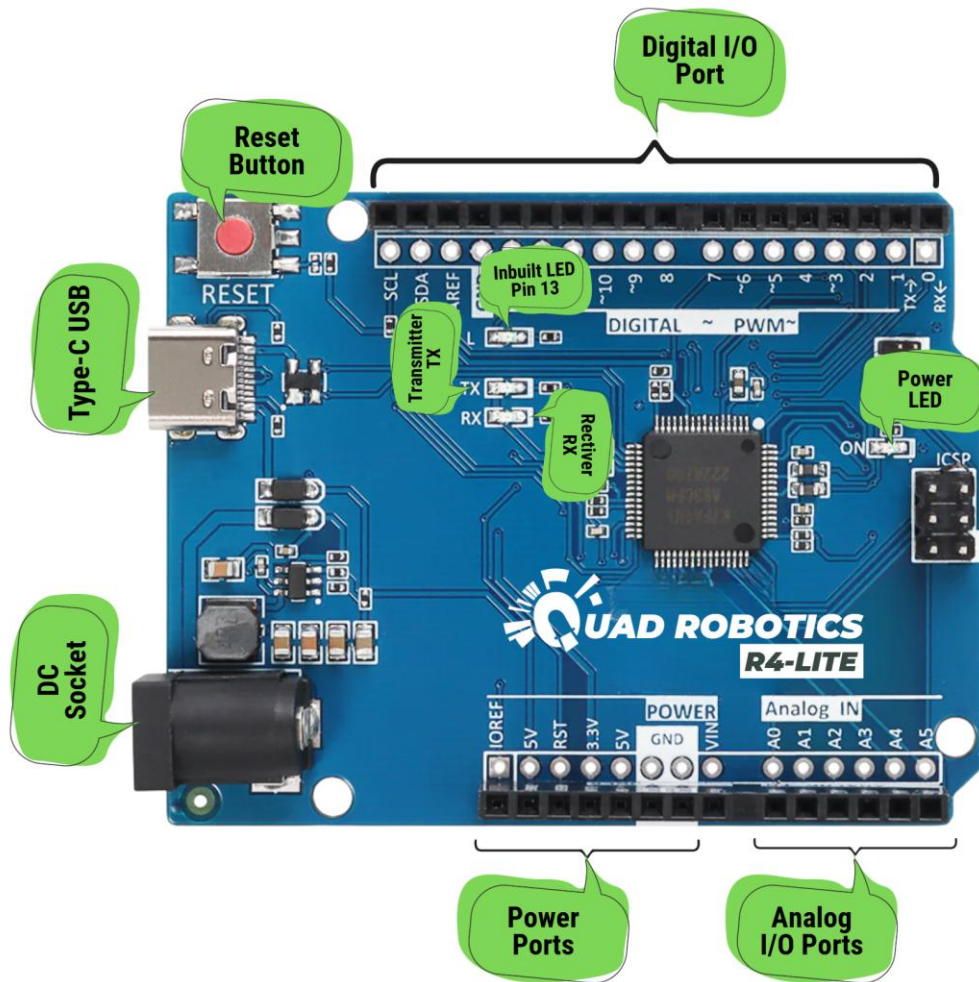
Made for Learners



Here to Help You Succeed!

Understanding Uno R4 Board:

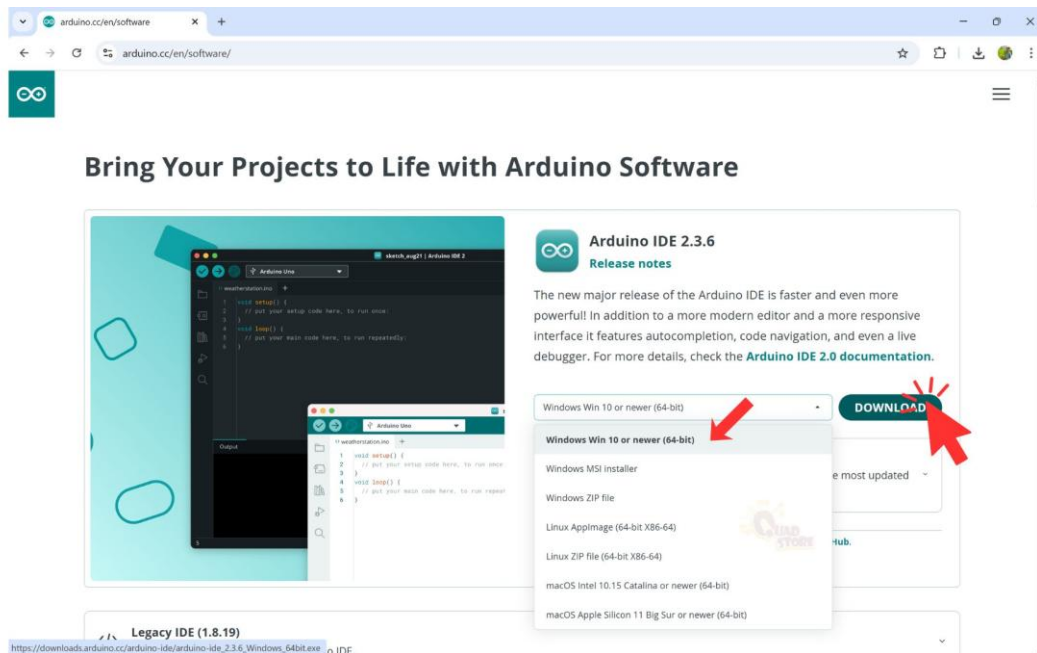
The reference board used in this book is **Quad Store's UNO R4 Lite board**, an Arduino-compatible Uno R4 minima model. A diagram of the Quad Store UNO board is shown below. The board included in your kit **may or may not** carry the **Quad Store / Quad Robotics logo**, as it depends on availability. However, all boards function identically.



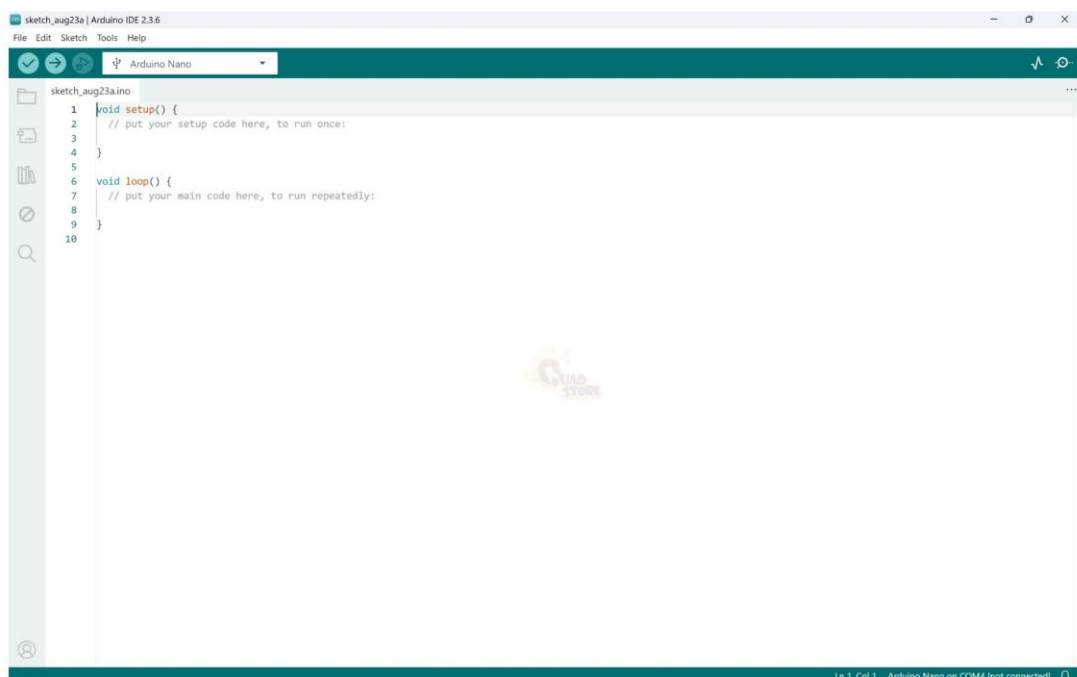
- **Type-C USB:** Used to power the board and upload programs from your computer.
- **Reset Button:** Restarts the board and reloads the running program.
- **DC Socket:** Allows powering the board using an external 7–12V adapter.
- **Digital I/O Port:** Pins used for digital input/output operations, including PWM.
- **Inbuilt LED Pin 13:** Built-in test LED connected to digital pin 13 for quick debugging.
- **Transmitter (TX):** Sends serial data from the board to other devices.
- **Receiver (RX):** Receives serial data into the board from other devices.
- **Power LED:** Indicates that the board is powered ON.
- **Power Ports:** Provide various voltage outputs (5V, 3.3V, GND, VIN) for sensors and modules.
- **Analog I/O Ports:** Pins A0–A5 used for reading analog sensor values.

Installing Arduino IDE

1. Visit the official Arduino website: <https://www.arduino.cc/en/software>
2. Download and install the Arduino IDE for Windows, Mac, or Linux.



3. Open the IDE by clicking on the desktop icon after installation. IDE should open as shown.

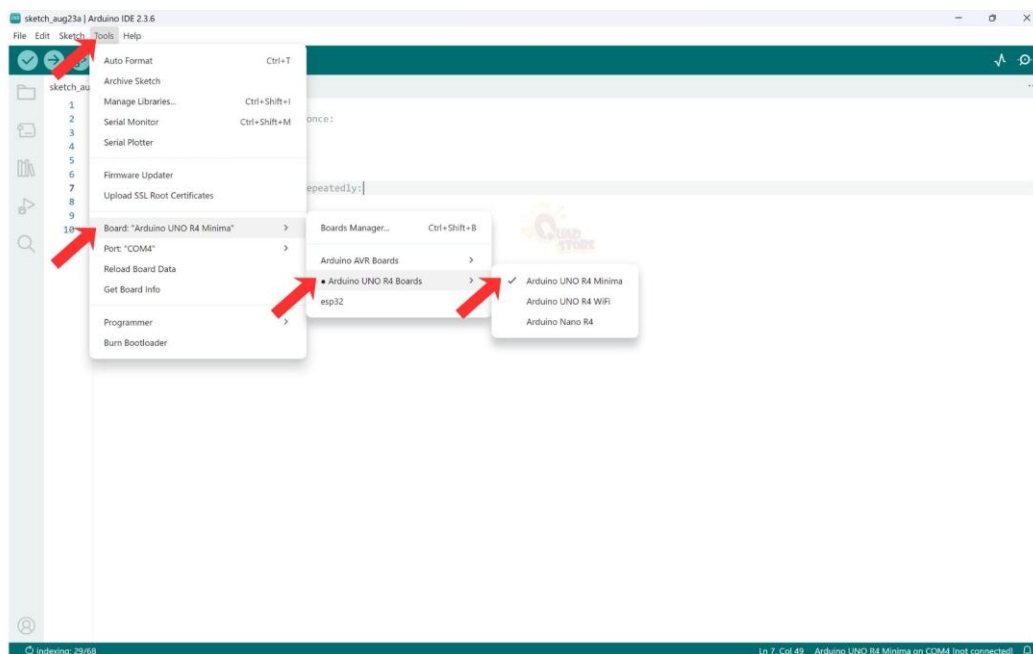


Connecting the UNO R4 Board

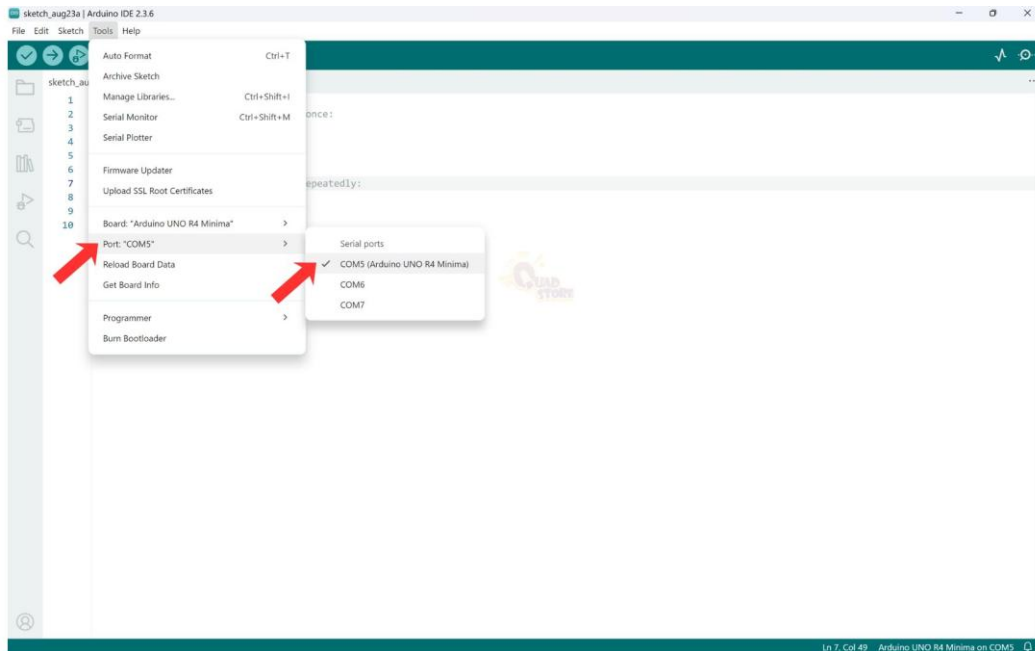
1. Use the Type-C to USB-A cable provided in the kit.
2. Plug the Type-C end into the UNO R4 and the USB-A end into your computer. If your computer or laptop has only a Type-C port, please purchase a Type-C to Type-A USB adapter and use it, or you may use a Type-C to Type-C USB cable instead.



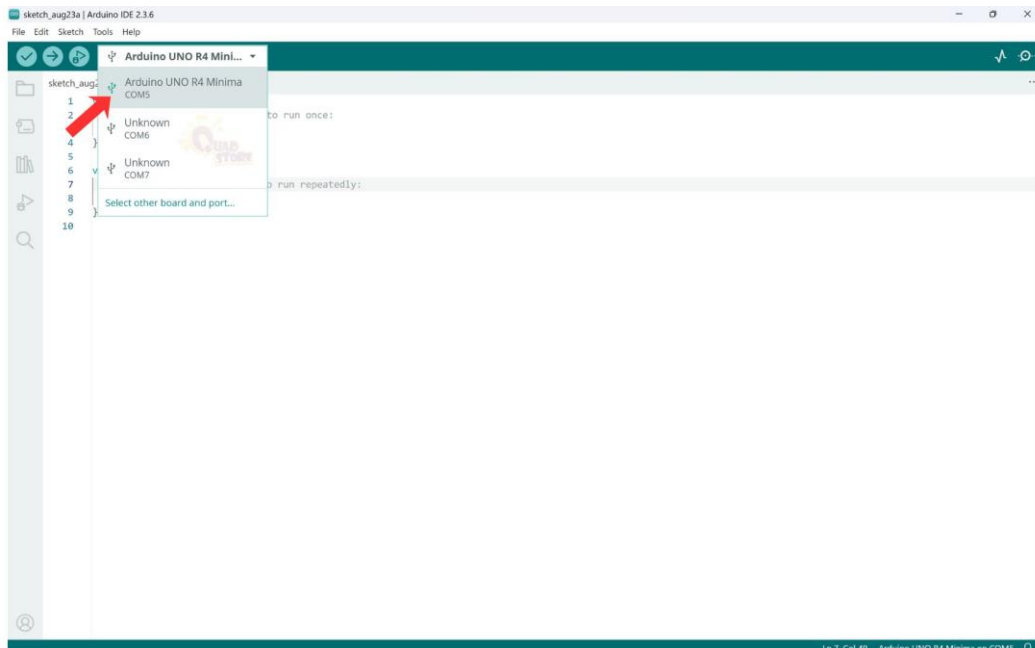
3. The power LED on the board will turn ON.
4. Go to **Tools** → **Board** → **Arduino Uno R4 Board** → Select “**Arduino Uno R4 Minima**” or “**Arduino Uno R4 Wifi**” based on your board. Here you need to select “**Arduino Uno R4 Minima**”.



5. Make sure the correct port is selected by going to **Tools → Port → COM** (Arduino Uno R4 boards) based on your COM port. **NOTE:** COM port number will change based on your computer/laptop.



6. Now you need to select the board. Click the drop-down menu under “Select Board” option. Select the corresponding board. Ensure the correct board is getting reflected in the IDE screen.





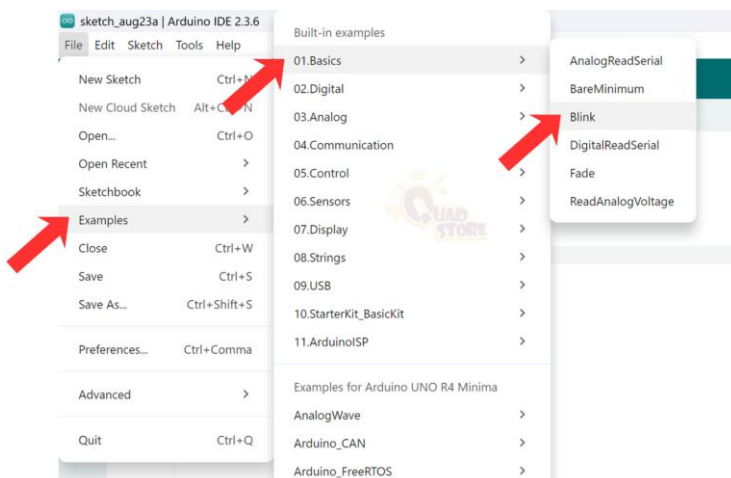
Uploading Your First Program (Blink Test)

This program makes the built-in LED blink and ensures the board and IDE are working correctly.

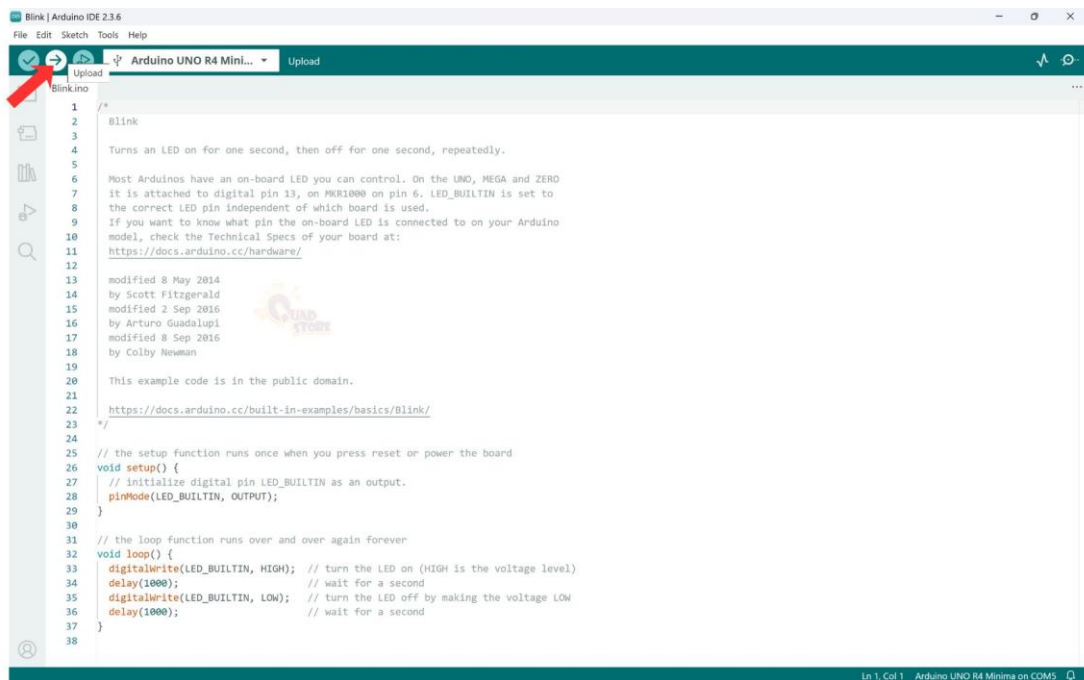
```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // Turn LED on
  delay(1000);                     // Wait 1 second
  digitalWrite(LED_BUILTIN, LOW);  // Turn LED off
  delay(1000);                     // Wait 1 second
}
```

Steps to Upload Code

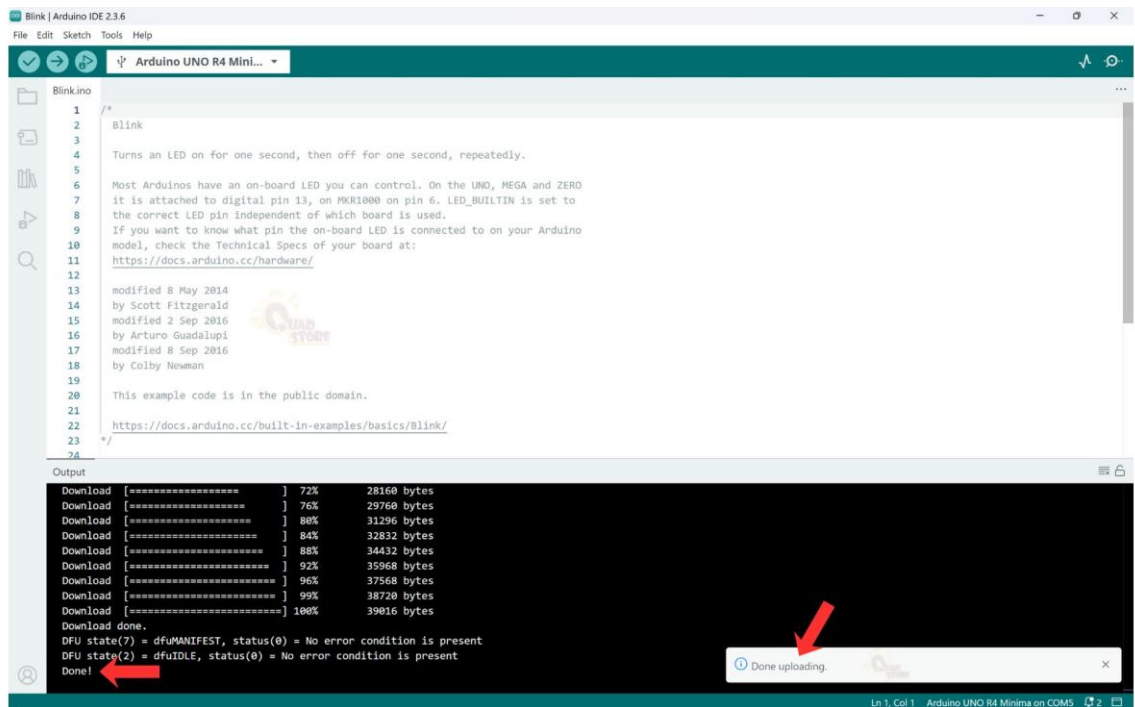
1. Either **copy & paste the above code** into the Arduino IDE **(OR)** open **File → Examples → 0.1 Basics → Blink** program.



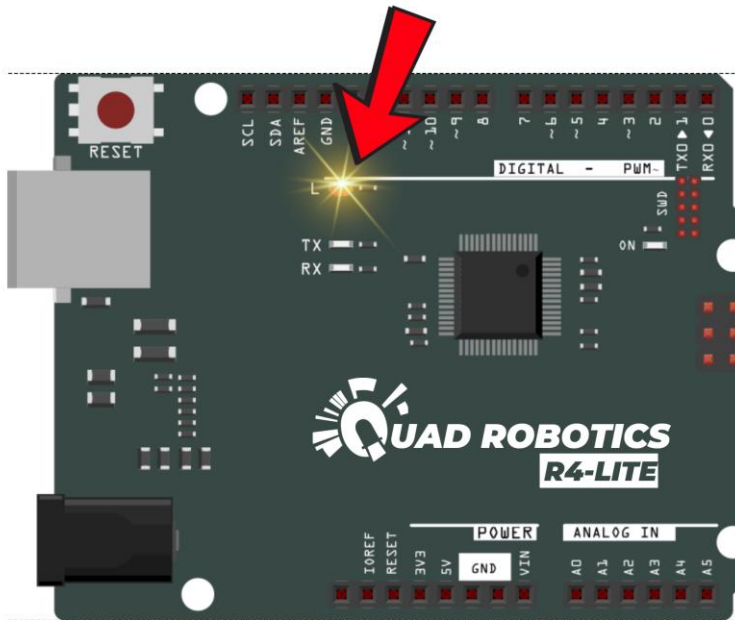
2. Click on the **Upload button** (right arrow icon).



3. Wait for the code to **compile and upload**. You should see a message **“Done!”** at the bottom of the screen. The board will start running the program automatically.



Output: You should see the built-in LED under PIN 13 starts to blink at a regular interval.



Ask Our Smart AI Mentor - QuadBot

Scan the QR code or click the picture link & ask our **Smart AI Mentor** for instant guidance!

Example Questions You Can Ask QuadBot:

- 💬 I accidentally clicked “No” while installing the UNO R4 driver. How do I fix it?
- 💬 My UNO R4 Lite board is not detected in Arduino IDE. What should I do?
- 💬 Why is the COM port not showing in Arduino IDE?
- 💬 Arduino IDE says “No board detected.” How can I solve it?
- 💬 How do I reinstall the UNO R4 drivers?
- 💬 Which board should I select for UNO R4 Lite in Arduino IDE?
- 💬 Why am I getting an upload error while uploading code?
- 💬 My board powers ON but code is not uploading. Why?
- 💬 How do I check if the USB driver is installed correctly?
- 💬 Which USB cable should I use for UNO R4 Lite?
- 💬 Why is my board not appearing in Device Manager?
- 💬 How do I manually install the UNO R4 driver?
- 💬 What should I do if Arduino IDE keeps showing “Port Busy” error?
- 💬 How do I fix “Failed uploading” error in Arduino IDE?
- 💬 Can you guide me step-by-step to connect and program my board?



Project 1: Blinking External LED

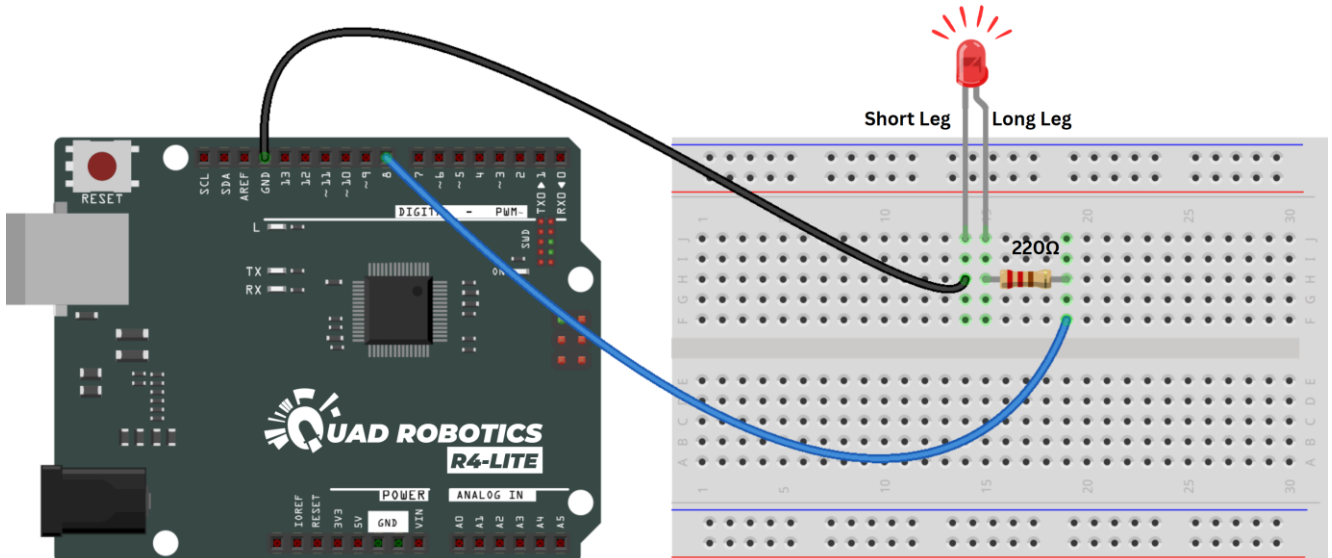
🌟 Objective: Blink an external LED connected on a breadboard.

📦 Components Required

- ✓ UNO R4 board
- ✓ Breadboard
- ✓ 1 × LED (any color)
- ✓ 1 × 220Ω Resistor
- ✓ Male to Male Jumper wires
- ✓ Type C to A USB cable

💡 Circuit Diagram

Component	Connection Method	Uno R4
LED (Anode, Long Leg +)	Through 220Ω resistor	Pin 8
LED (Cathode, Short Leg -)	Direct Connection	GND



💻 Code

```
int led = 8;

void setup() {
  pinMode(led, OUTPUT);
}

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

Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **copy and paste the above code** into the Arduino IDE OR open the file **1.LEDBlink.ino** from the provided **CODE folder**.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as Arduino Uno R4 minima from drop down list.
6. Click on the **Upload button** (right arrow icon).
7. Wait for the code to **compile and upload**. The board will start running the program automatically

Output

The program turns the LED ON (HIGH) and OFF (LOW) every second.

DIY Extension

Try different colors of LEDs.

Add more LEDs on pins 9 and 10 for patterns like traffic lights.

Ask Our Smart AI Mentor - QuadBot

Stuck while doing the project? Need help understanding the code or circuit?

Scan the QR code or click the picture link & ask our **Smart AI Mentor** for instant guidance!

Example Questions You Can Ask QuadBot:

- 💬 Why is my LED not blinking?
- 💬 My UNO board is not detected. What should I do?
- 💬 Which side of the LED is positive and negative?
- 💬 Why do we use a 220Ω resistor?
- 💬 How do I make the LED blink faster?
- 💬 Can you explain this code line by line?
- 💬 What happens if I remove the resistor?
- 💬 How do I upload code using Arduino IDE?
- 💬 Why am I getting a COM port error?
- 💬 Can I use a different pin instead of Pin 8?
- 💬 Can you give me a beginner-friendly explanation of this project?
- 💬 What are some fun DIY extensions for this project?
- 💬 Can I use this project with batteries instead of USB?



Project 2: Traffic Lights with 3 LEDs

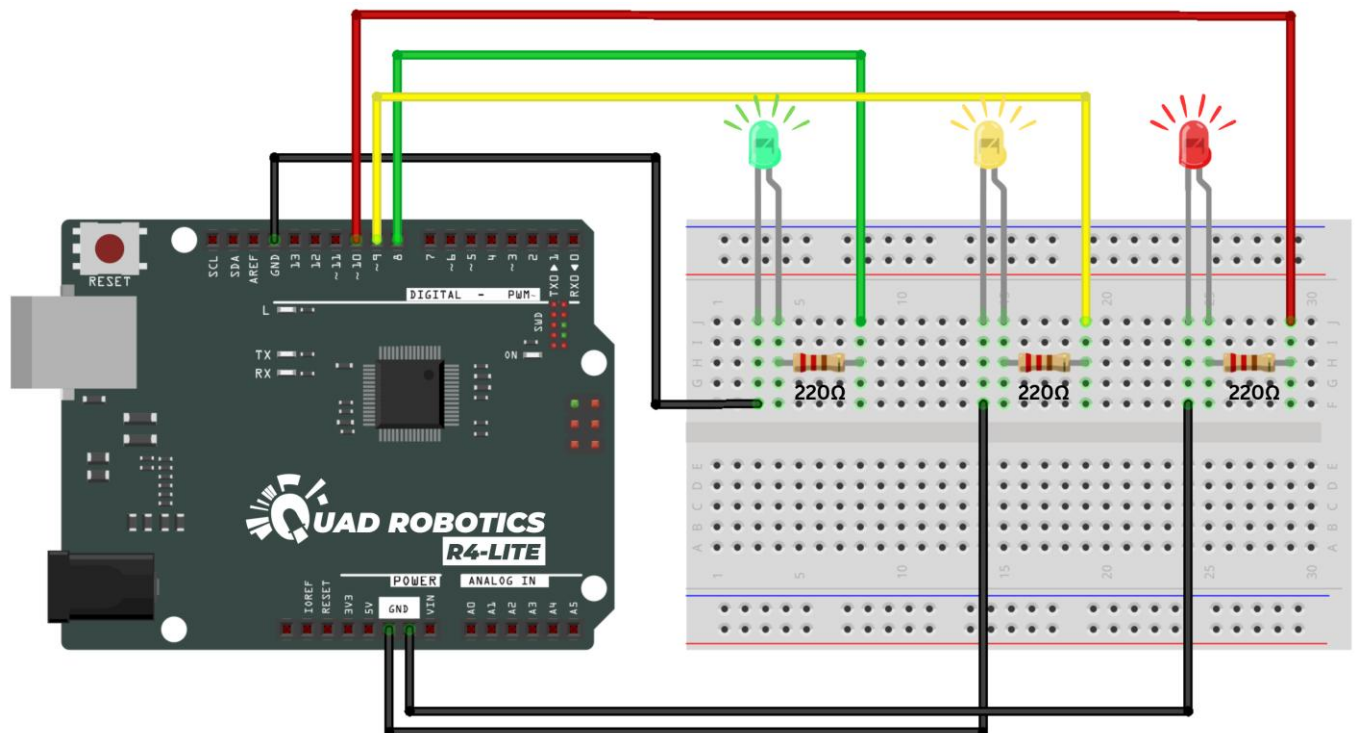
🌟 Objective: Simulate a simple traffic light system using 3 LEDs.

📦 Components Required

- ✓ UNO R4 board
- ✓ Breadboard
- ✓ 3 × LEDs (Red, Yellow, Green)
- ✓ 3 × 220Ω Resistors
- ✓ Male to Male Jumper wires
- ✓ Type C to A USB cable

💡 Circuit Diagram

Component	Connection Method	Uno R4
Green LED (Anode, Long Leg +)	Through 220Ω resistor	Pin 8
Green LED (Cathode, Short Leg -)	Direct Connection	GND
Yellow LED (Anode, Long Leg +)	Through 220Ω resistor	Pin 9
Yellow LED (Cathode, Short Leg -)	Direct Connection	GND
Red LED (Anode, Long Leg +)	Through 220Ω resistor	Pin 10
Red LED (Cathode, Short Leg -)	Direct Connection	GND



Code

```
int red = 10;
int yellow = 9;
int green = 8;

void setup() {
  pinMode(red, OUTPUT);
  pinMode(yellow, OUTPUT);
  pinMode(green, OUTPUT);
}
void loop() {
  digitalWrite(red, HIGH);
  delay(3000);
  digitalWrite(red, LOW);

  digitalWrite(yellow, HIGH);
  delay(1000);
  digitalWrite(yellow, LOW);

  digitalWrite(green, HIGH);
  delay(3000);
  digitalWrite(green, LOW);
}
```

Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **copy and paste above code** into the Arduino IDE OR open the file **2.TrafficLights.ino** from the **CODE folder**.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as Arduino Uno R4 minima from drop down list.
6. Click on the **Upload button** (right arrow icon).
7. Wait for the code to **compile and upload**. The board will start running the program automatically.

Output

Red stays ON for 3 seconds, Yellow for 1 second, and Green for 3 seconds.

DIY Extension

Add a pedestrian button to change the lights when pressed.

Use a buzzer for sound alerts.

Project 3: LED control using Push Button

🌟 Objective: To control an **LED** using a **push button** connected to an **UNO R4**.

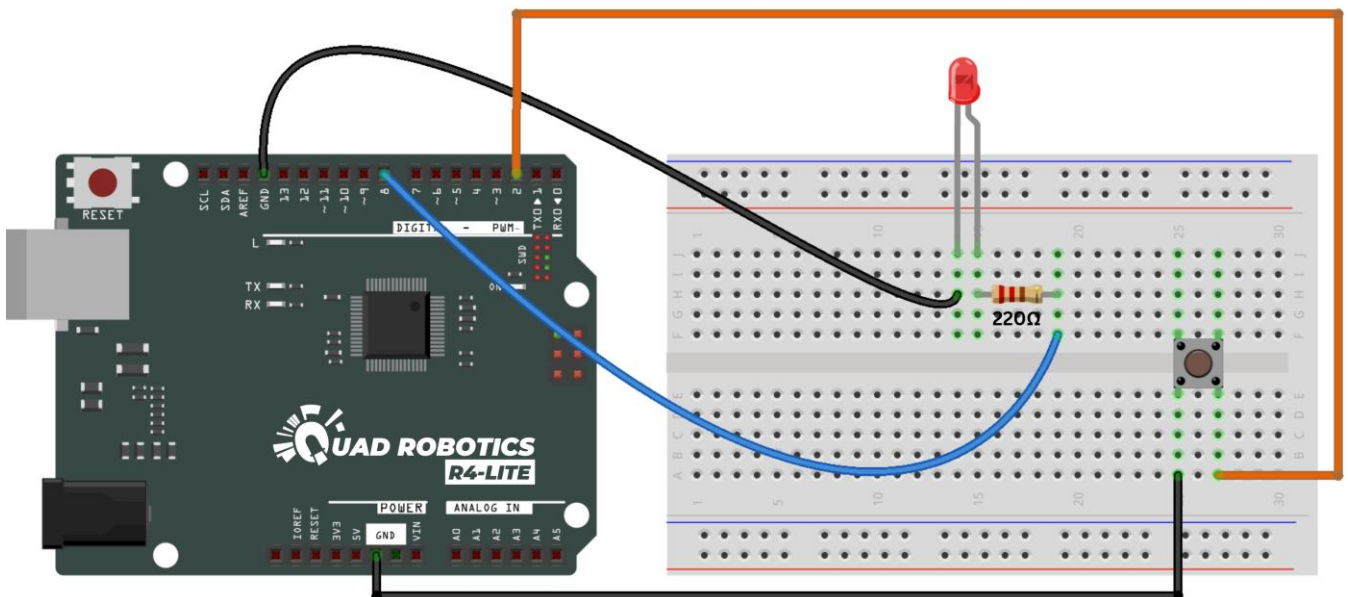
- When button is pressed → LED turns ON
- When button is released → LED turns OFF

🛒 Components Required

- ✓ UNO R4 board
- ✓ Breadboard
- ✓ 1 × LED (Any color of your choice)
- ✓ 1 × 220Ω Resistors
- ✓ 1 × Push Button Switch
- ✓ Male to Male Jumper wires

💡 Circuit Diagram

Component	Connection Method	Uno R4
LED (Anode, Long Leg +)	Through 220Ω resistor	Pin 6
LED (Cathode, Short Leg -)	Direct Connection	GND
Push Button	One side	Pin 2
Push Button	Other side	GND



Code

```
const int buttonPin = 2;
const int ledPin = 8;

void setup() {
  pinMode(buttonPin, INPUT_PULLUP); // internal pull-up
  pinMode(ledPin, OUTPUT);
}

void loop() {
  int buttonState = digitalRead(buttonPin);

  if (buttonState == LOW) {
    // button pressed
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **copy and paste above code** into the Arduino IDE OR open the file **3.PushButtonLED.ino** from the provided **CODE folder**.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as Arduino Uno R4 minima from drop down list.
6. Click on the **Upload button** (right arrow icon).
7. Wait for the code to **compile and upload**. The board will start running the program automatically.

Output

- LED stays OFF initially
- Press the button → LED turns ON
- Release the button → LED turns OFF.

DIY Extension

Toggle LED mode

Press button once → LED ON

Press again → LED OFF

Project 4: Passive Buzzer Sound

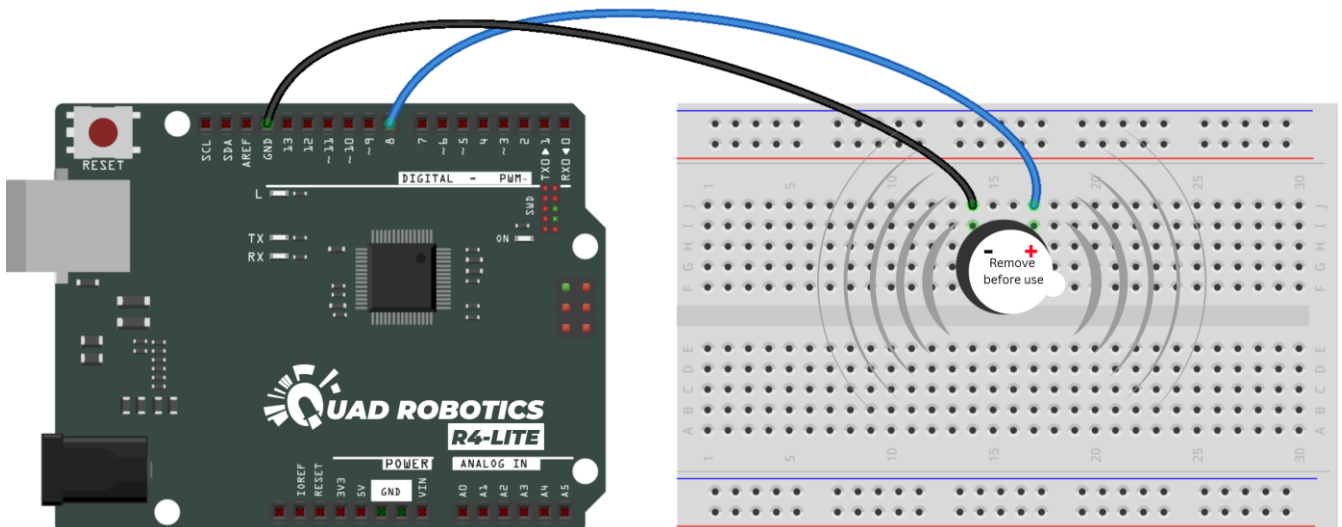
🌟 Objective: Generate sound using a passive buzzer with tone function.

🧰 Components Required

- ✓ UNO R4 board
- ✓ Passive Buzzer (remove the top sticker on the buzzer before use)
- ✓ Male to Male Jumper wires
- ✓ Breadboard

🔌 Circuit Diagram

Component	Connection Method	Uno R4
Buzzer (Positive +)	Direct Connection	Pin 8
Buzzer (Negative -)	Direct Connection	GND



💻 Code

```
int buzzer = 8;

void setup() {
}

void loop() {
  tone(buzzer, 1000); // 1kHz tone
  delay(1000);
  noTone(buzzer);
  delay(1000);
}
```


Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **copy and paste above code** into the Arduino IDE OR open the file **4.PassiveBuzzer.ino** from the provided **CODE folder**.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as Arduino Uno R4 minima from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Explanation & Output

The passive buzzer produces sound when driven with different frequencies.

Here we generate a 1kHz tone for 1 second ON and 1 second OFF.

DIY Extension

Change frequency for different musical notes.

Try making a melody.

Project 5: Active Buzzer Beep

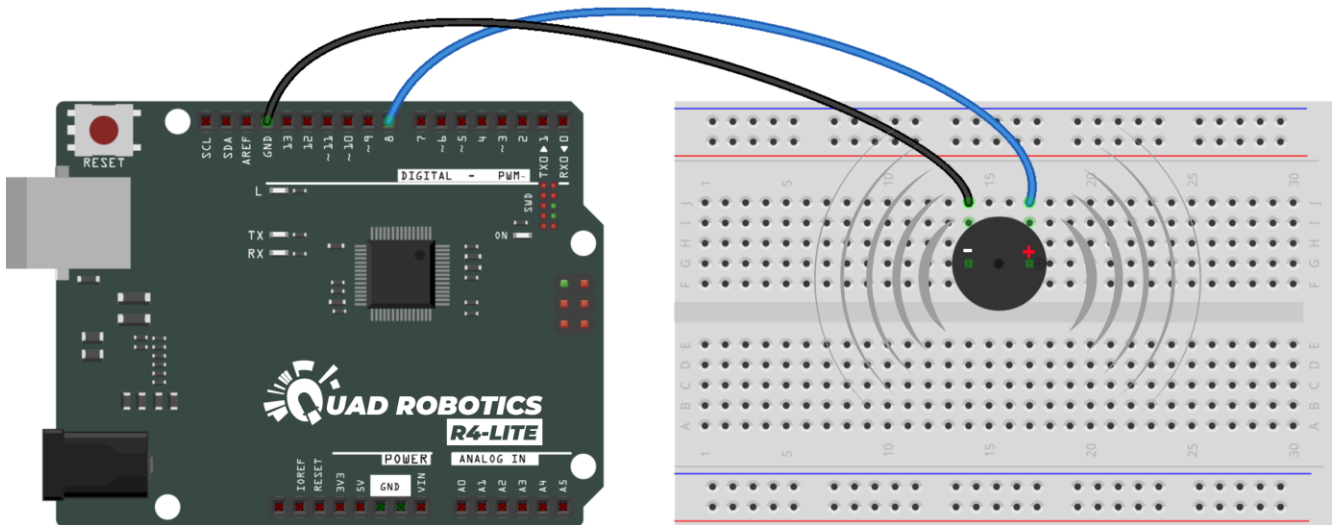
🌟 Objective: Turn an active buzzer ON and OFF.

📦 Components Required

- ✓ UNO R4 board
- ✓ Active Buzzer
- ✓ Male to Mae Jumper wires
- ✓ Breadboard

🔌 Circuit Diagram

Component	Connection Method	Uno R4
Buzzer (Positive +)	Direct Connection	Pin 8
Buzzer (Negative -)	Direct Connection	GND



💻 Code

```
int buzzer = 8;

void setup() {
  pinMode(buzzer, OUTPUT);
}

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

Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **5.ActiveBuzzer.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Explanation & Output

An active buzzer produces a fixed beep when powered.

We simply switch it ON and OFF every second.

DIY Extension

Use buzzer for alarms in sensor projects.

Project 6: RGB LED Color Mixing

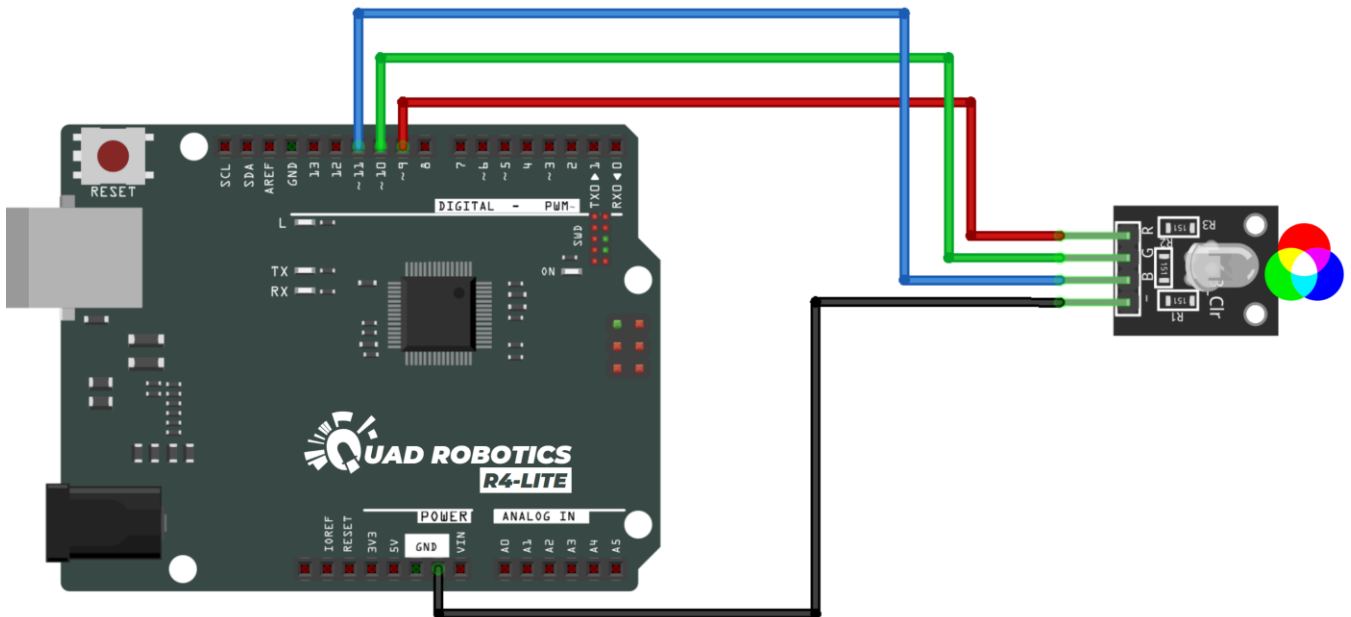
🔦 Objective: Control an RGB LED to show different colors.

📦 Components Required

- ✓ UNO R4 board
- ✓ RGB LED Module
- ✓ Male to Female Jumper wires. A breadboard is not required when using these wires, as you can directly connect the RGB module to the female end.

🔌 Circuit Diagram

Component	Connection Method	Uno R4
RGB Module – R (Red)	Direct Connection	Pin 9
RGB Module – G (Green)	Direct Connection	Pin 10
RGB Module – B (Blue)	Direct Connection	Pin 11
RGB Module – GND	Direct Connection	GND



Code

```
int red = 9;
int green = 10;
int blue = 11;

void setup() {
  pinMode(red, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(blue, OUTPUT);
}

void loop() {
  digitalWrite(red, HIGH);
  digitalWrite(green, LOW);
  digitalWrite(blue, LOW);
  delay(1000);

  digitalWrite(red, LOW);
  digitalWrite(green, HIGH);
  delay(1000);

  digitalWrite(green, LOW);
  digitalWrite(blue, HIGH);
  delay(1000);
}
```

Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **6.RGBLed.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Explanation & Output

An RGB LED combines red, green, and blue light to produce colors. We switch between red, green, and blue every second.

DIY Extension

Mix multiple colors by turning ON two pins together.
Use analogWrite for smooth fading.

Project 7: Potentiometer-controlled LED Brightness

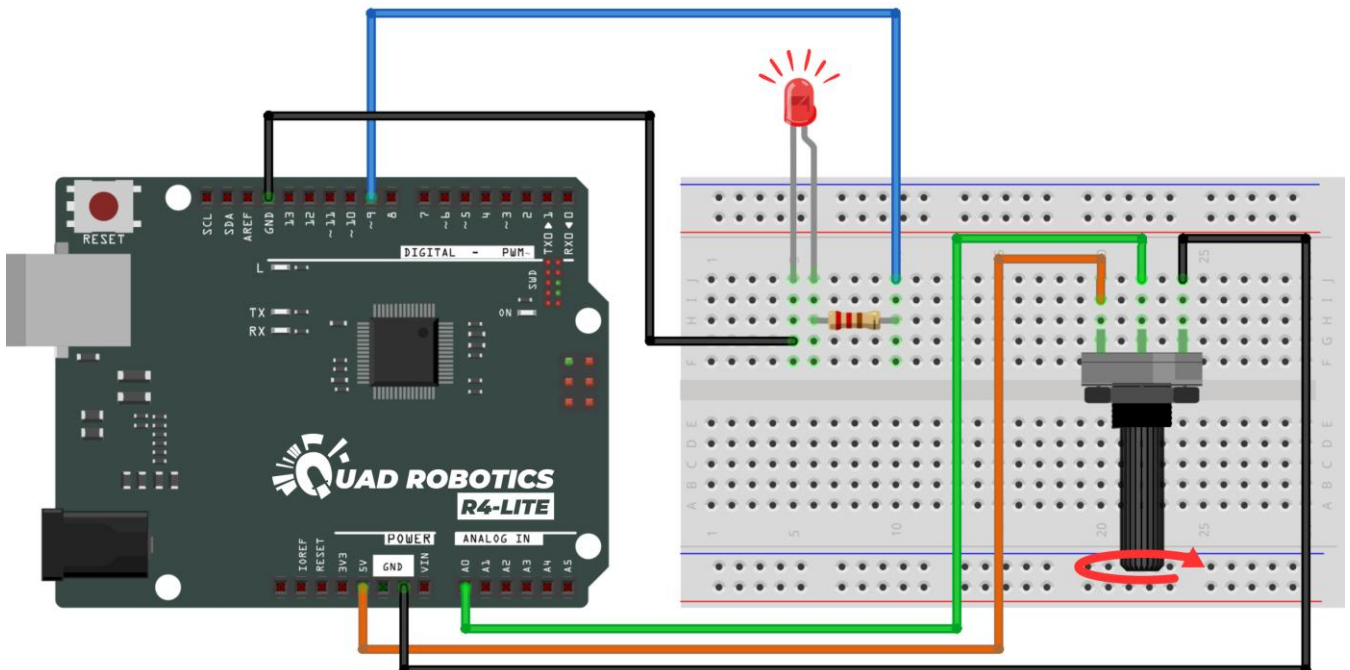
🌟 Objective: Use a potentiometer to adjust LED brightness (PWM).

🧰 Components Required

- ✓ UNO R4 board
- ✓ 10K Potentiometer
- ✓ LED
- ✓ 220Ω Resistor
- ✓ Breadboard
- ✓ Male to Male Jumper wires

💡 Circuit Diagram

Component	Connection Method	Uno R4
Potentiometer (Middle Pin / Wiper)	Direct Connection	A0
Potentiometer (Left Pin)	Direct Connection	5V
Potentiometer (Right Pin)	Direct Connection	GND
LED (Anode, Long Leg +)	Through 220Ω resistor	Pin 9
LED (Cathode, Short Leg -)	Direct Connection	GND



Code

```
int potPin = A0;
int led = 9;
int val = 0;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  val = analogRead(potPin);
  val = map(val, 0, 1023, 0, 255);
  analogWrite(led, val);
}
```

Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **7.PotLED.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Explanation & Output

Potentiometer provides analog input from 0–1023.

We map it to 0–255 to control LED brightness with PWM.

Output: Turn the potentiometer knob from left to right to see the LED brightness change from low to high.

DIY Extension

Try controlling buzzer pitch with potentiometer.

Project 8: Photoresistor/ Light Dependent Resistor (LDR) or Automatic Night Lamp

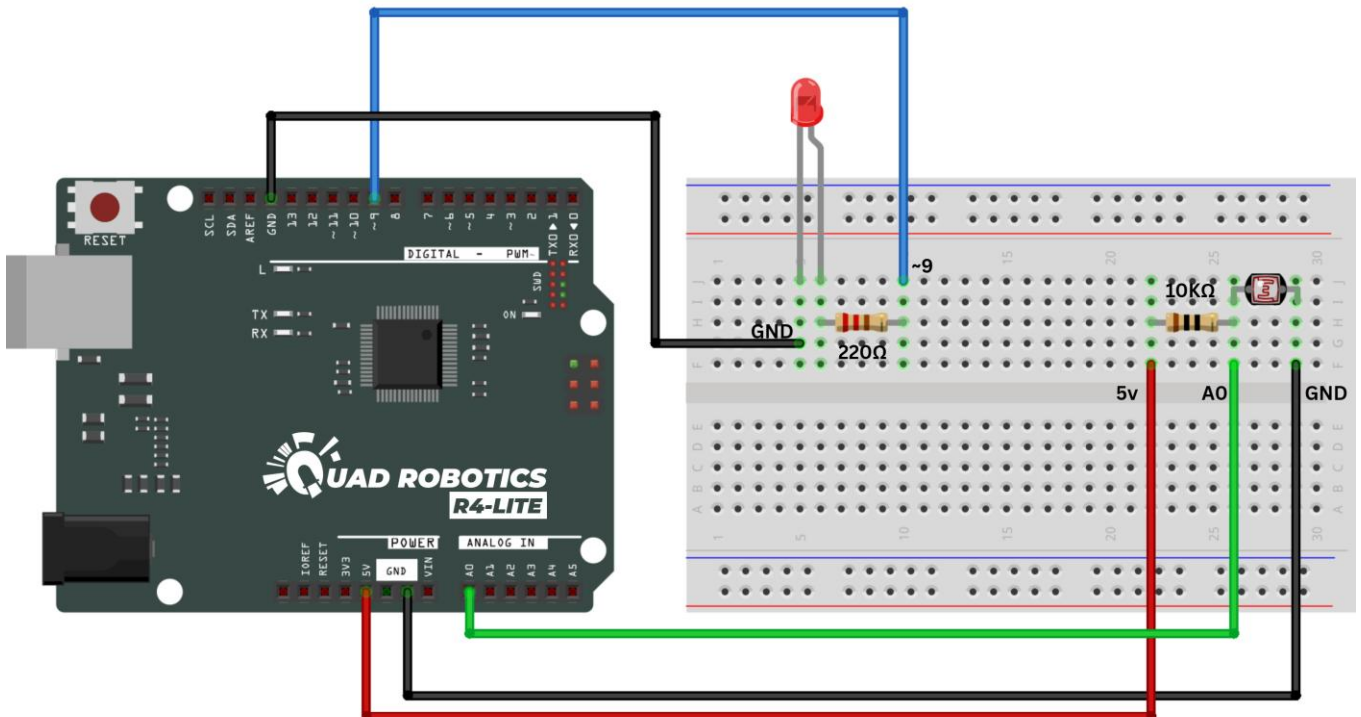
✦ Objective: Turn ON an LED in darkness using LDR sensor.

🧰 Components Required

- ✓ UNO R4 board
- ✓ Photoresistor / LDR
- ✓ LED
- ✓ 220Ω Resistor
- ✓ Breadboard
- ✓ Jumper wires

💡 Circuit Diagram

Component	Connection Method	Uno R4
LDR (One Leg)	Direct Connection	GND
LDR (Other Leg)	Through 10 kΩ resistor	5V
LDR + Resistor Junction (Middle Point)	Direct Connection	A0
LED (Anode, Long Leg +)	Through 220 Ω resistor	Pin 9
LED (Cathode, Short Leg -)	Direct Connection	GND



Code

```
int ldrPin = A0;
int ledPin = 9;

void setup() {
  pinMode(ledPin, OUTPUT);
}
void loop() {
  int lightValue = analogRead(ldrPin);
  if (lightValue < 500) {    // Bright
    digitalWrite(ledPin, LOW);
  } else {                  // Dark
    digitalWrite(ledPin, HIGH);
  }
}
```

Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **8.LDRlamp.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Explanation & Output

The LDR detects light level.

LED turns ON when it's dark (low value) or close the LDR with your finger to turn ON the LED.

DIY Extension

Adjust threshold value for sensitivity.

Use for automatic room lights.

Project 9: Detect Tilt/Vibration

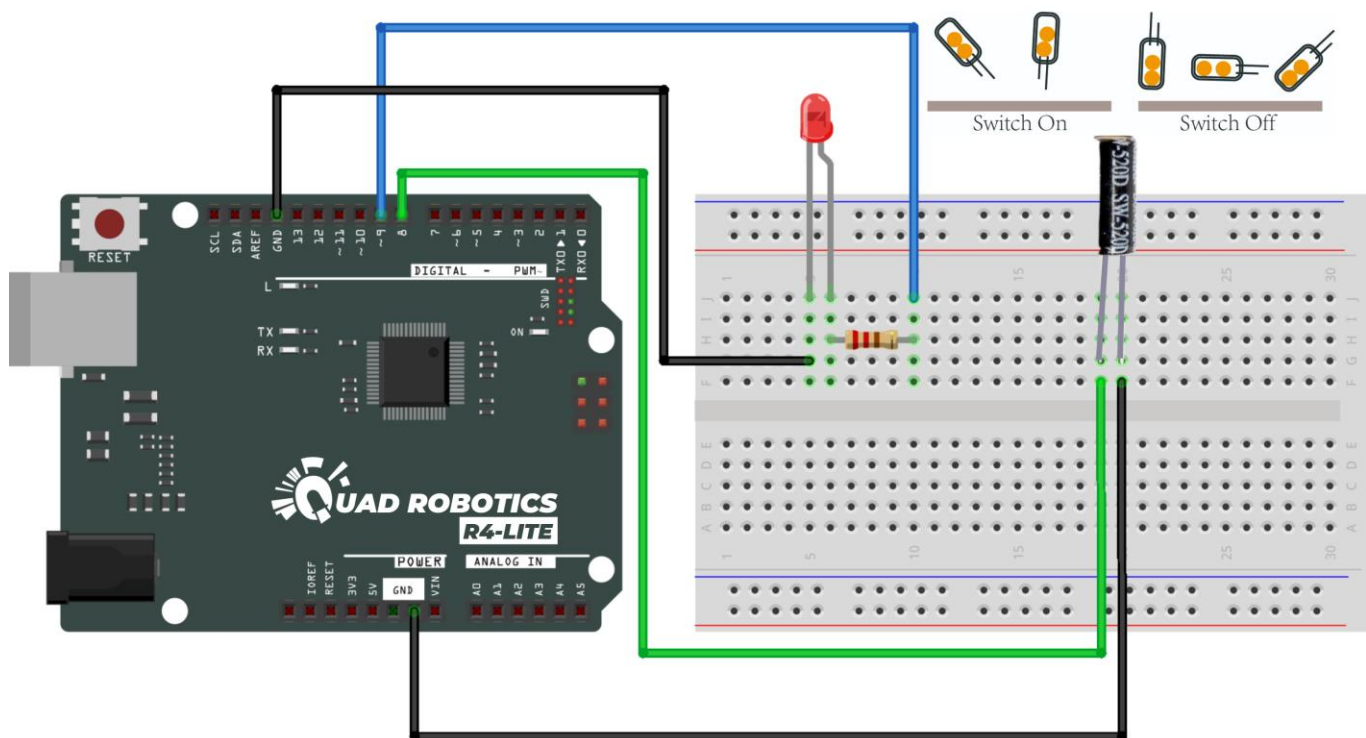
🌟 Objective: Detect tilt/vibration movement using SW-520D sensor.

📦 Components Required

- ✓ UNO R4 board
- ✓ Tilt Sensor
- ✓ Buzzer or LED
- ✓ Breadboard
- ✓ Jumper wires

🔌 Circuit Diagram

Component	Connection Method	Uno R4
Tilt Sensor (One Leg)	Direct Connection	Pin 8
Tilt Sensor (Other Leg)	Direct Connection	GND
LED (Positive / Long Leg +)	Through 220Ω resistor	Pin 9
LED (Negative / Short Leg -)	Direct Connection	GND



Code

```
int tilt = 8;
int led = 9;

void setup()
{
  pinMode(tilt, INPUT_PULLUP); // Use internal pull-up
  pinMode(led, OUTPUT);
}

void loop()
{
  if (digitalRead(tilt) == LOW)
  {
    digitalWrite(led, HIGH);
  }
  else {
    digitalWrite(led, LOW);
  }
}
```

Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **9.TiltSensor.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Explanation & Output

Tilt sensor works like a switch when moved.

Take the breadboard and tilt it upside down to see the LED/buzzer turn on and off when tilted.

DIY Extension

Use in anti-theft alarms.

Project 10: Servo Motor Sweep

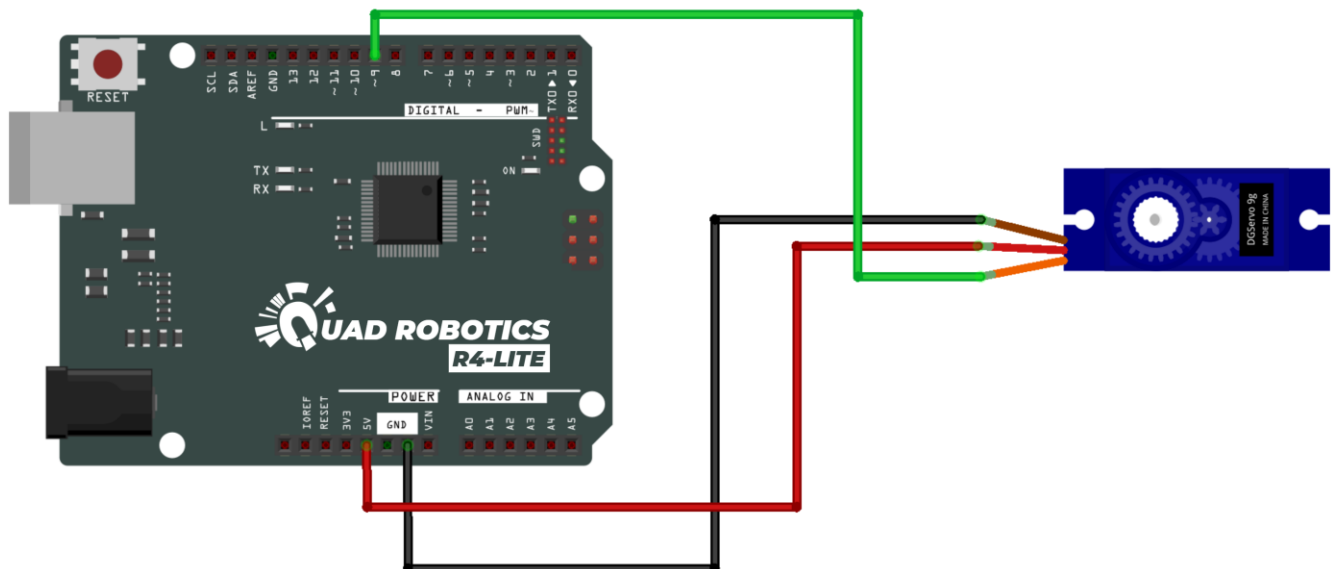
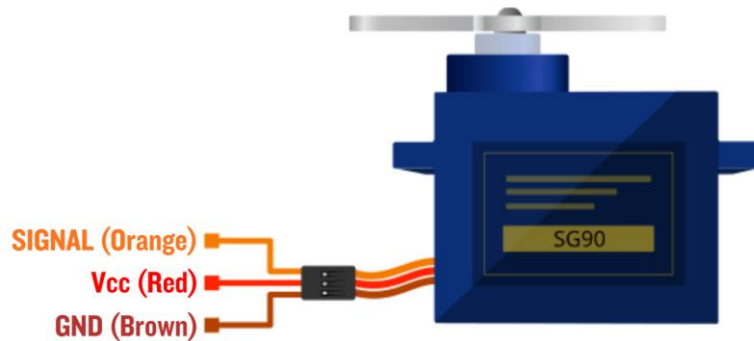
🌟 Objective: Move SG90 servo back and forth.

📦 Components Required

- ✓ UNO R4 board
- ✓ SG90 Servo
- ✓ Jumper wires. Use male to male jumper wires and insert into servo motor socket pins.

💡 Circuit Diagram

Component	Connection Method	Uno R4
Servo Signal (Orange wire)	Direct Connection	Pin 9
Servo VCC (Red wire)	Direct Connection	5V
Servo GND (Brown wire)	Direct Connection	GND



Code

```
#include <Servo.h>

Servo myservo;

void setup() {
  myservo.attach(9);
}

void loop() {
  for(int pos=0; pos<=180; pos++){
    myservo.write(pos);
    delay(15);
  }
  for(int pos=180; pos>=0; pos--){
    myservo.write(pos);
    delay(15);
  }
}
```

Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **10.ServoSweep.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Explanation & Output

Servo is controlled by PWM signals.

We sweep from 0° to 180° and back.

DIY Extension

Use for robot arms.

Project 11: Ultrasonic Distance Finder

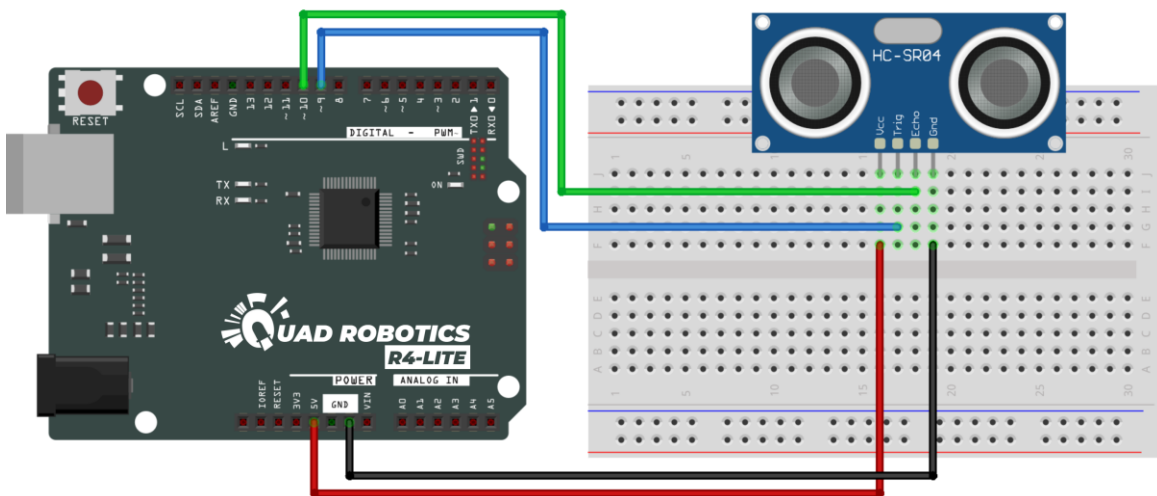
🌟 Objective: Measure distance using HC-SR04 ultrasonic sensor.

🧰 Components Required

- ✓ UNO R4 board
- ✓ Ultrasonic Sensor
- ✓ Breadboard
- ✓ Jumper wires

💡 Circuit Diagram

Component	Connection Method	Uno R4
Ultrasonic Sensor – Trig	Direct Connection	Pin 9
Ultrasonic Sensor – Echo	Direct Connection	Pin 10
Ultrasonic Sensor – VCC	Direct Connection	5V
Ultrasonic Sensor – GND	Direct Connection	GND



💻 Code

```
int trig = 9;
int echo = 10;
long duration;
int distance;

void setup() {
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(trig, LOW);
  delayMicroseconds(2);
```

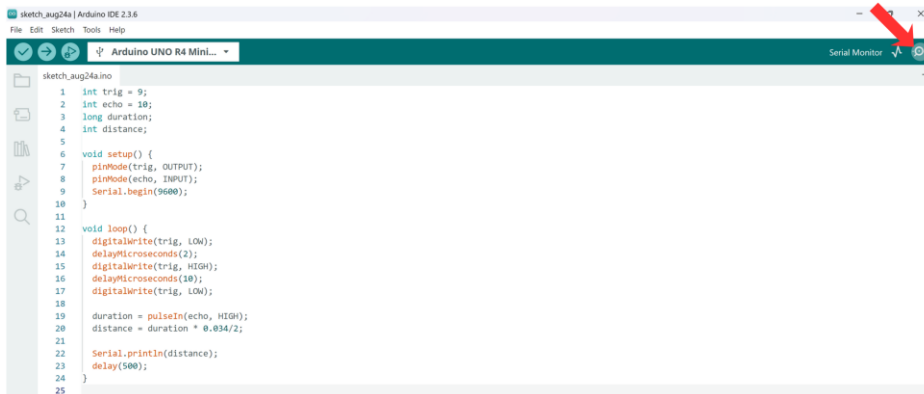
```
digitalWrite(trig, HIGH);
delayMicroseconds(10);
digitalWrite(trig, LOW);

duration = pulseIn(echo, HIGH);
distance = duration * 0.034/2;

Serial.println(distance);
delay(500);
}
```

Steps to Upload Code

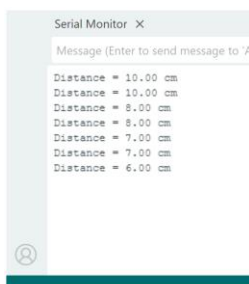
1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **11.Ultrasonics.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.



Explanation & Output

Sensor sends ultrasonic pulse and measures echo time.

Distance is calculated using speed of sound and it is displayed in the serial monitor.



✂️ DIY Extension: Make obstacle avoidance robots.

Project 12: DHT11 Weather Monitor

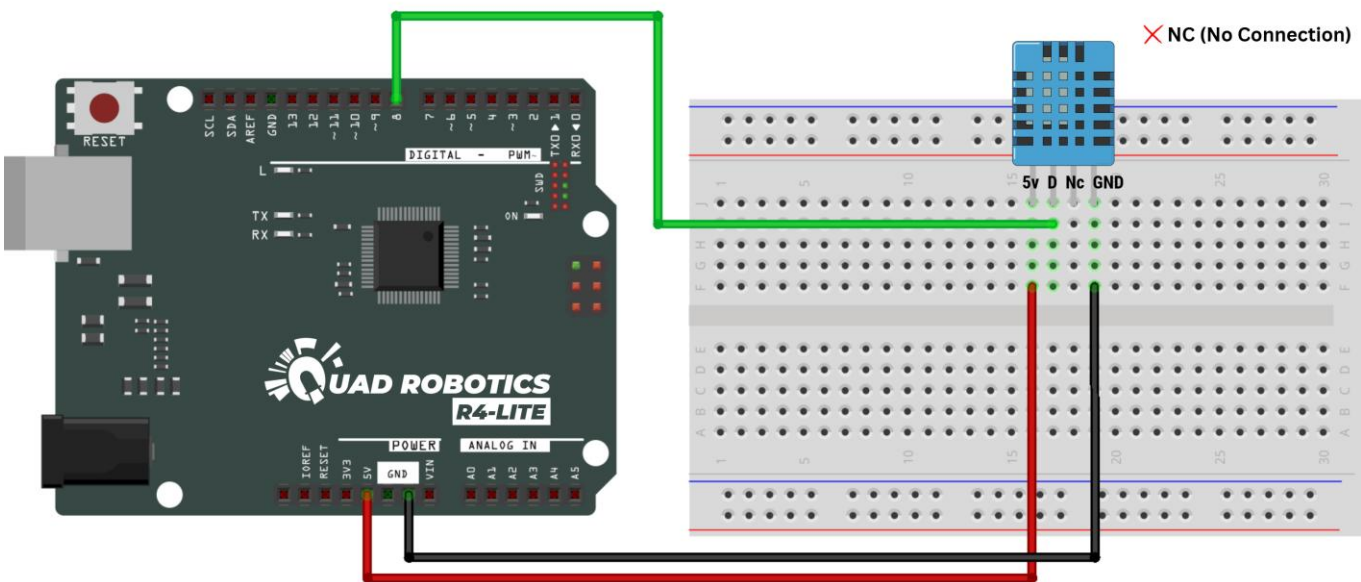
🌟 Objective: Read temperature and humidity using DHT11 sensor.

📦 Components Required

- ✓ UNO R4 board
- ✓ DHT11 Module
- ✓ Breadboard
- ✓ Jumper wires

🔌 Circuit Diagram

Component	Connection Method	Uno R4
DHT11 – VCC	Direct Connection	5V
DHT11 – Data	Direct Connection	Pin 8
DHT11 – NC	No Connection	—
DHT11 – GND	Direct Connection	GND



💻 Code

```
#include <DHT.h>
#define DHTPIN 8
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
```

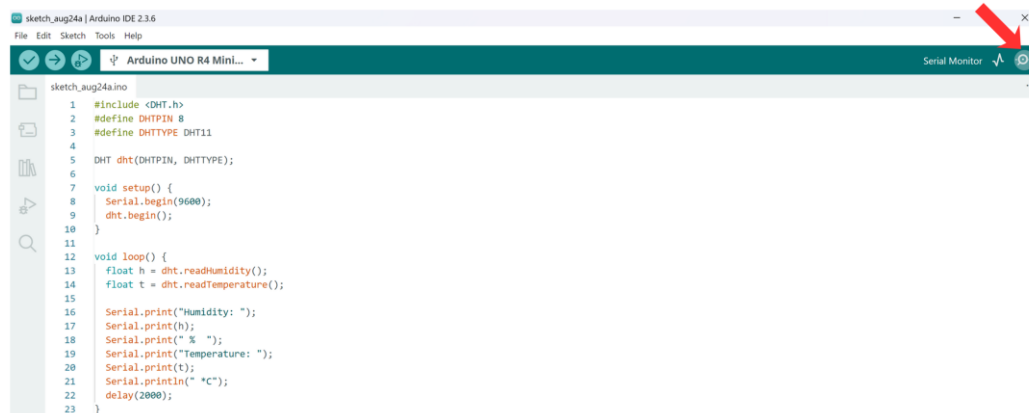
```

Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" % ");
Serial.print("Temperature: ");
Serial.print(t);
Serial.println(" *C");
delay(2000);
}

```

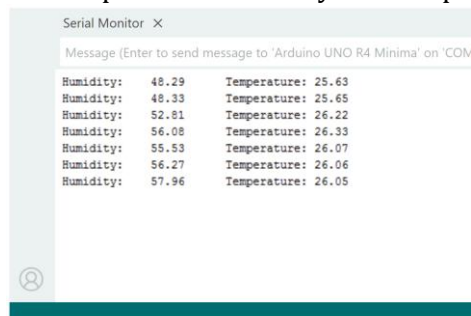
Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **12.DHT11Monitor.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.



Explanation & Output

DHT11 provides humidity and temperature readings. Values are printed on Serial Monitor.



 **DIY Extension:** Make your own weather station.

Project 13: Fire Sensor Alarm

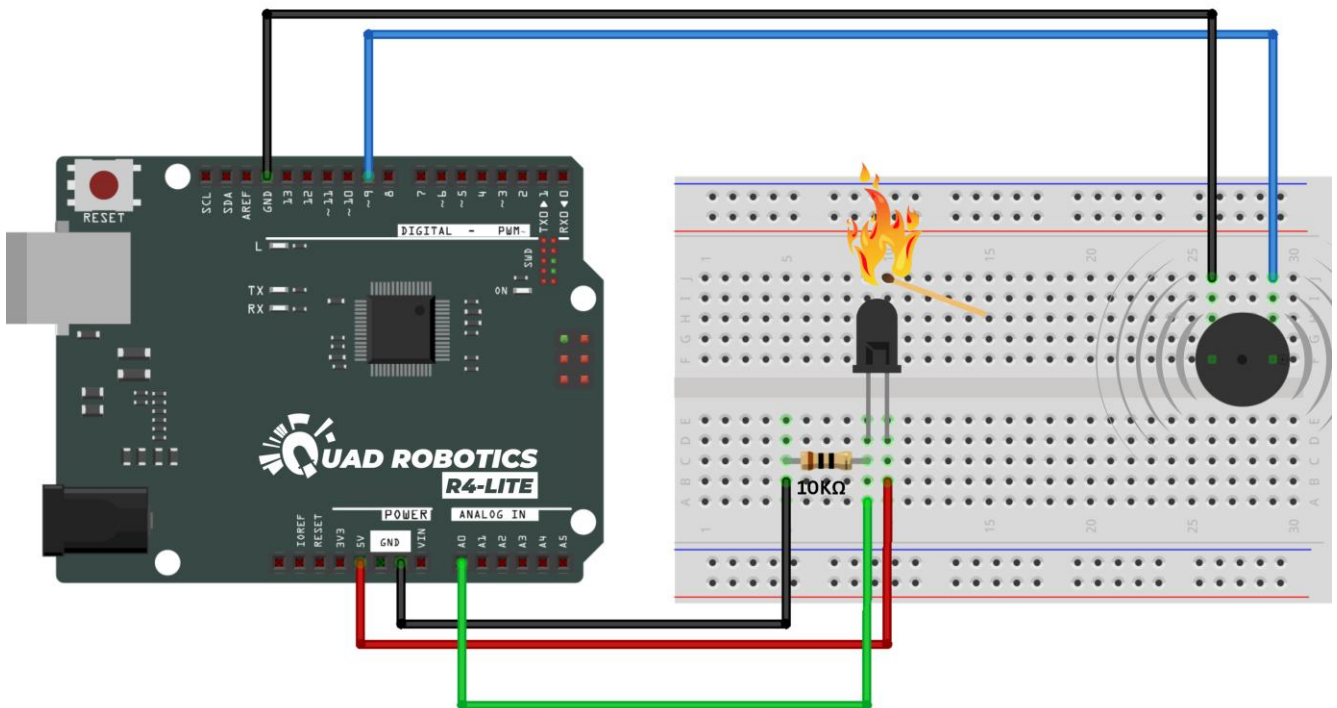
🔥 Objective: Detect fire and trigger buzzer.

📦 Components Required

- ✓ UNO R4 board
- ✓ Fire Sensor Module
- ✓ Buzzer
- ✓ Breadboard
- ✓ Jumper wires

🔌 Circuit Diagram

Component	Connection Method	Uno R4
Fire Sensor (One Leg)	Direct Connection	5V
Fire Sensor (Other Leg)	Through 10 kΩ resistor	GND
Fire Sensor + Resistor Junction (Middle Point)	Direct Connection	A0
Buzzer (Positive +)	Direct Connection	Pin 9
Buzzer (Negative -)	Direct Connection	GND



Code

```
const int PD_PIN = A0;
const int LED_PIN = 9;
const int THRESH = 30; // tune after testing (0-1023)

void setup() {
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int v = analogRead(PD_PIN);
  Serial.println(v);
  if (v >= THRESH) {      // lower = more IR light (photodiode conducting)
    digitalWrite(LED_PIN, HIGH);
  } else {
    digitalWrite(LED_PIN, LOW);
  }
  delay(100);
}
```

Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **13.FireAlarm.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Explanation & Output

Fire sensor detects flames or strong IR light.

If fire detected, buzzer turns ON.

DIY Extension

Use as a safety alarm.

Project 14: I2C LCD Display Basics

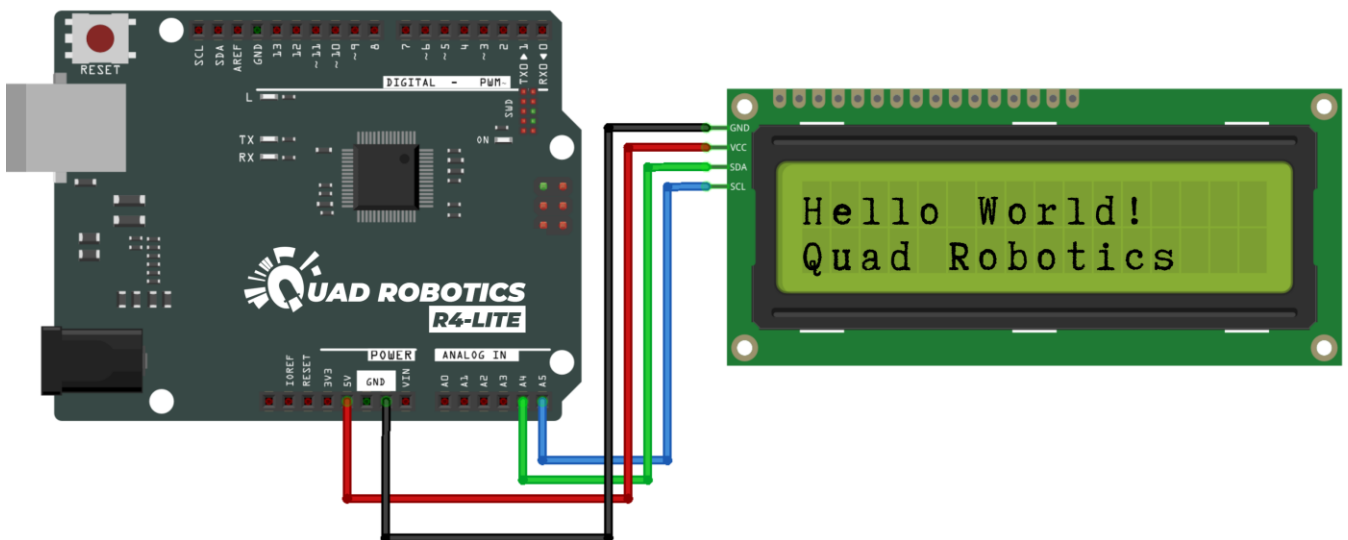
🌟 Objective: Display text on I2C LCD display.

📦 Components Required

- ✓ UNO R4
- ✓ 0.96 inch OLED Display (I2C)
- ✓ Male to Female Jumper wires

💡 Circuit Diagram

Component	Connection Method	Uno R4
I2C LCD – VCC	Direct Connection	5V
I2C LCD – GND	Direct Connection	GND
I2C LCD – SDA	Direct Connection	A4
I2C LCD – SCL	Direct Connection	A5



✅ Required Library

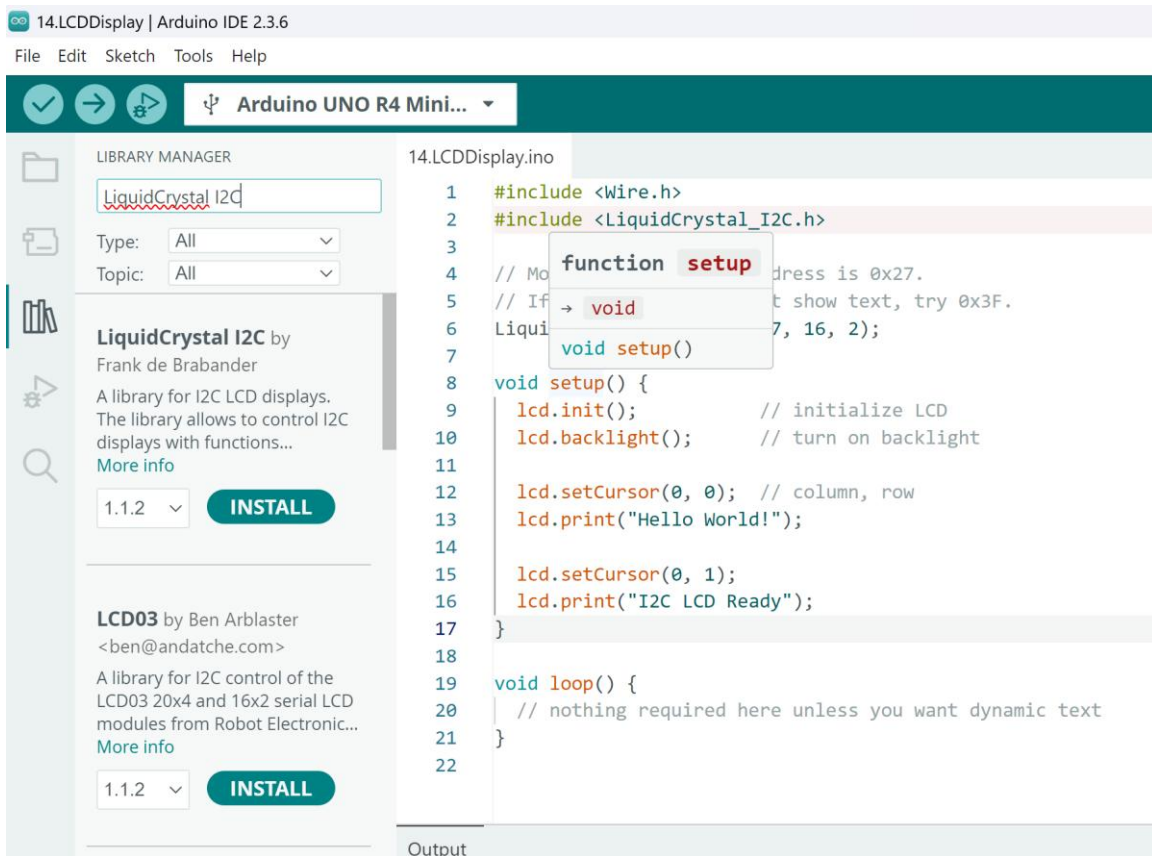
Install this library in Arduino IDE:

LiquidCrystal_I2C

(Author: Frank de Brabander)

Steps:

Sketch → Include Library → Manage Libraries → search "**LiquidCrystal I2C**" → Install



Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **14.LEDDispaly.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Code

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

// Most common I2C LCD address is 0x27.
// If your display doesn't show text, try 0x3F.
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  lcd.init();           // initialize LCD
  lcd.backlight();      // turn on backlight

  lcd.setCursor(0, 0);  // column, row
  lcd.print("Hello World!");

  lcd.setCursor(0, 1);
  lcd.print("Quad Robotics");
}

void loop() {
  // nothing required here unless you want dynamic text
}
```

Explanation & Output

The LED uses I2C communication (SDA, SCL).

Output: We display simple text “Hello World” on the first line of screen and “Quad Robotics” text on the second line.

DIY Extension

Show sensor values on LED.

Make a mini weather station display.

Project 15: Stepper Motor

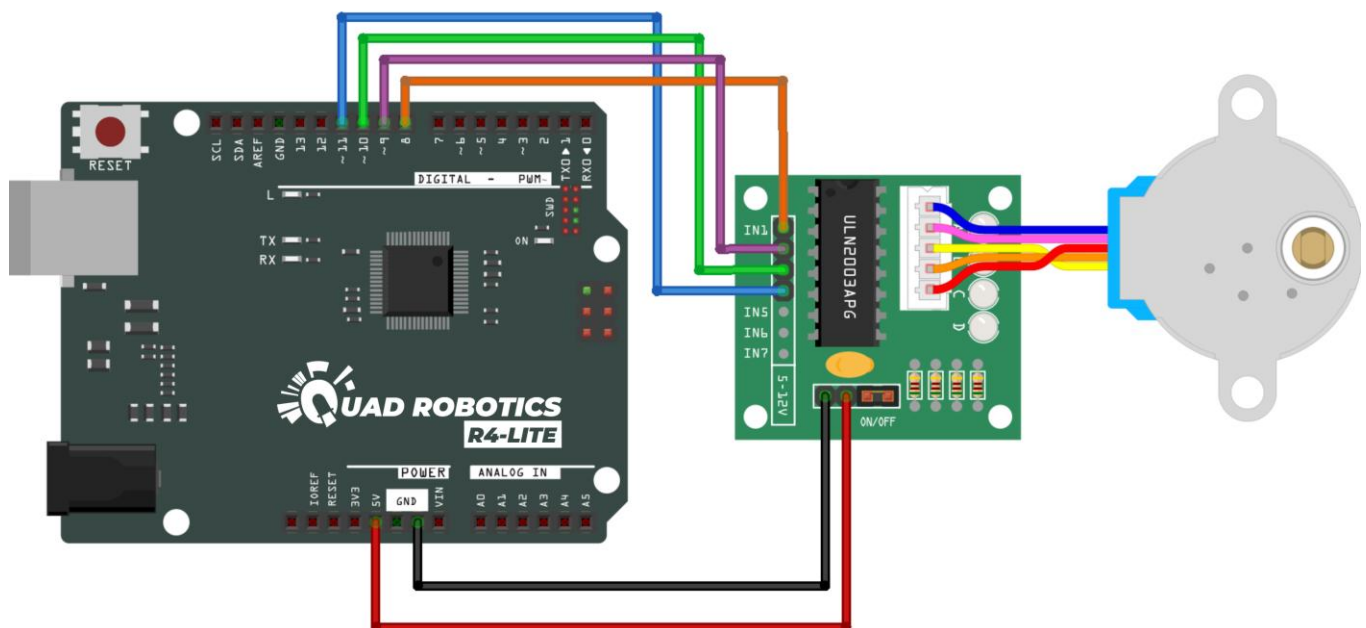
🌟 Objective: To interface a **28BYJ-48 stepper motor** with an **UNO R4** using the **ULN2003 motor driver** and control the motor to rotate clockwise and anticlockwise.

📦 Components Required

- ✓ UNO R4 board
- ✓ 28BYJ-48 Stepper Motor
- ✓ ULN2003 Stepper Motor Driver Board
- ✓ Jumper wires (Male to Female)
- ✓ USB cable for Arduino

🔌 Circuit Diagram

Component	Connection Method	Uno R4
ULN2003 – IN1	Direct Connection	Pin 8
ULN2003 – IN2	Direct Connection	Pin 9
ULN2003 – IN3	Direct Connection	Pin 10
ULN2003 – IN4	Direct Connection	Pin 11
ULN2003 – VCC (+)	Direct Connection	5V
ULN2003 – GND (-)	Direct Connection	GND
Stepper Motor (5-pin JST)	Plug into ULN2003 socket	—



Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **15.StepperMotor.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Code

```
#include <Stepper.h>

const int STEPS_PER_REV = 2048;    // 28BYJ-48 full revolution steps

// Pin sequence for ULN2003: IN1-IN3-IN2-IN4 or IN1-IN2-IN3-IN4
Stepper myStepper(STEPS_PER_REV, 8, 10, 9, 11);

void setup() {
  myStepper.setSpeed(12);    // Speed in RPM (recommended: 5-15)
  Serial.begin(9600);
  Serial.println("Stepper Motor Test Starting...");
}

void loop() {
  Serial.println("Rotating Clockwise...");
  myStepper.step(2048);    // one full rotation clockwise
  delay(1000);

  Serial.println("Rotating Anti-Clockwise...");
  myStepper.step(-2048);    // one full rotation anticlockwise
  delay(1000);
}
```

Output

After uploading the code:

- The motor rotates one full turn clockwise
- Pauses for 1 second
- Then rotates one full turn anticlockwise
- Repeats continuously

✂️ DIY Extension: Button-controlled Rotation: Use two push buttons — one for clockwise, one for anticlockwise.

Project 16: 1-Digit 7 Segment Display

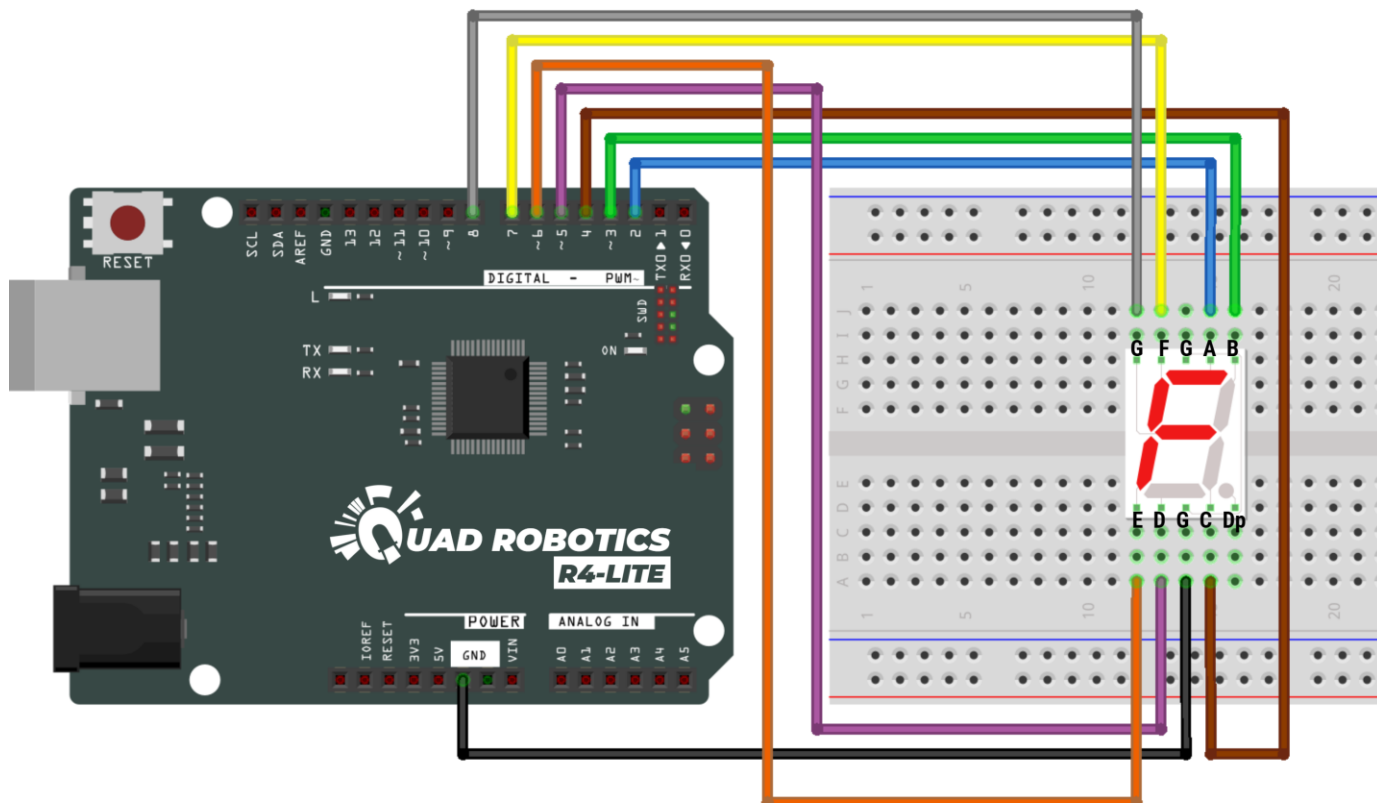
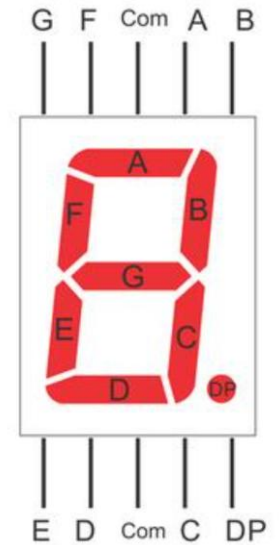
🌟 Objective: To connect a **1-digit 7-segment display** to an **Arduino UNO R3** and display numbers from 0 to 9 by controlling each segment using digital output pins.

🧰 Components Required

- ✓ UNO R4 board
- ✓ 1-digit 7-segment display (Common Cathode)
- ✓ 8 × 220Ω resistors (optional for safety)
- ✓ Jumper wires
- ✓ Breadboard

💡 Circuit Diagram

Component	Connection Method	Uno R4
Segment A	Direct Connection /Optional 220Ω resistor	Pin D2
Segment B	Direct Connection /Optional 220Ω resistor	Pin D3
Segment C	Direct Connection /Optional 220Ω resistor	Pin D4
Segment D	Direct Connection /Optional 220Ω resistor	Pin D5
Segment E	Direct Connection /Optional 220Ω resistor	Pin D6
Segment F	Direct Connection /Optional 220Ω resistor	Pin D7
Segment G	Direct Connection /Optional 220Ω resistor	Pin D8
Common Pin (COM)	Direct Connection	GND
Decimal Point (DP)	<i>Not connected</i>	—



Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **16.7Segment1Digit.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Code

```
// Pin connections for each segment
int a = 2;
int b = 3;
int c = 4;
int d = 5;
int e = 6;
int f = 7;
int g = 8;
int dp = 9;

// Array of segments for easy looping
int segments[] = {a, b, c, d, e, f, g};

void setup() {
  // Set all segment pins as outputs
  for (int i = 0; i < 7; i++) {
    pinMode(segments[i], OUTPUT);
  }
  pinMode(dp, OUTPUT);
}

void displayDigit(int digit) {
  // Segment patterns for digits 0-9 (common cathode)
  const byte numbers[10][7] = {
    // a b c d e f g
    {1,1,1,1,1,1,0}, // 0
    {0,1,1,0,0,0,0}, // 1
    {1,1,0,1,1,0,1}, // 2
    {1,1,1,1,0,0,1}, // 3
    {0,1,1,0,0,1,1}, // 4
    {1,0,1,1,0,1,1}, // 5
    {1,0,1,1,1,1,1}, // 6
    {1,1,1,0,0,0,0}, // 7
    {1,1,1,1,1,1,1}, // 8
  }
```



```

    {1,1,1,1,0,1,1}    // 9
};

// Write segment values
for (int i = 0; i < 7; i++) {
    digitalWrite(segments[i], numbers[digit][i]);
}
}

void loop() {
    for (int i = 0; i < 10; i++) {
        displayDigit(i);
        delay(1000);    // show each number for 1 second
    }
}

```

🔍 Explanation & Output

After uploading the code:

- The 7-segment display will show:
0 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9
- Each number appears for 1 second before switching.
- All segments glow clearly using individual resistors.

✂️ DIY Extension

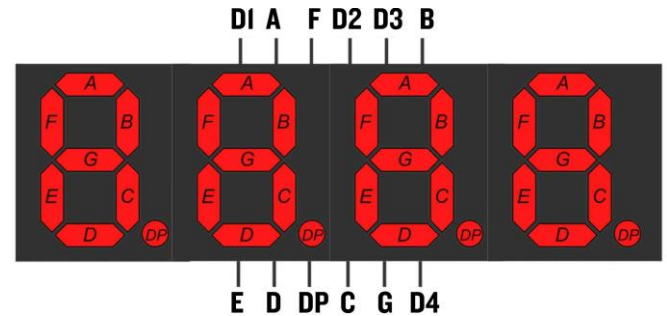
Display a custom pattern: Show letters like A, b, C, d using custom segment mappings.

Project 17: 4-Digit 7 Segment Display

🌟 Objective: Increment the 4-digit 7-segment display by **one** on every push-button press (0000 → 0001 → to → 9999 → 0000).

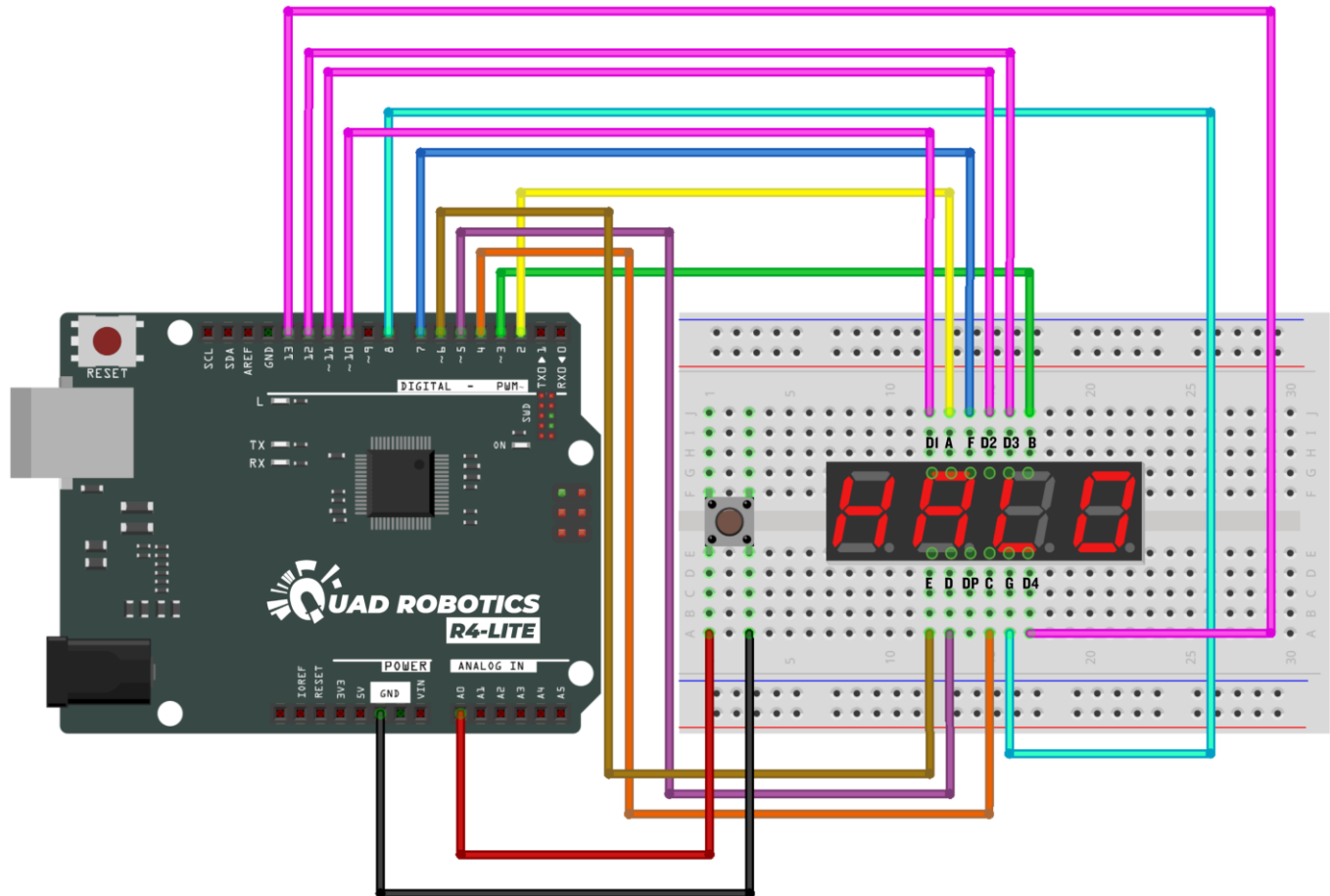
🧰 Components Required

- ✓ UNO R4 board
- ✓ 4-digit 7-segment display (Common Anode)
- ✓ 8 × 220Ω resistors (optional for safety)
- ✓ Jumper wires
- ✓ Push Button Switch
- ✓ Breadboard



🔌 Circuit Diagram

Component	Connection Method	Uno R4
Segment A	Direct Connection /Optional 220Ω resistor	Pin D2
Segment B	Direct Connection /Optional 220Ω resistor	Pin D3
Segment C	Direct Connection /Optional 220Ω resistor	Pin D4
Segment D	Direct Connection /Optional 220Ω resistor	Pin D5
Segment E	Direct Connection /Optional 220Ω resistor	Pin D6
Segment F	Direct Connection /Optional 220Ω resistor	Pin D7
Segment G	Direct Connection /Optional 220Ω resistor	Pin D8
Decimal Point (DP)	Direct Connection /Optional 220Ω resistor	Pin D9
Digit D1 (Leftmost)	Direct Connection	Pin D10
Digit D2	Direct Connection	Pin D11
Digit D3	Direct Connection	Pin D12
Digit D4 (Rightmost)	Direct Connection	Pin D13
Component	Connection Method	Uno R4
Push Button (One Pin)	Direct Connection	Pin A0
Push Button (Other Pin)	Direct Connection	GND



Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **17.7Segment4Digit.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Code

```
// 4-digit 7-seg increment on button press (COMMON ANODE)
// Segments: D2..D9 (a,b,c,d,e,f,g,dp), Digits: D10..D13 (DIG1..DIG4), Button:
// A0 (INPUT_PULLUP)

const uint8_t segPins[8] = {2,3,4,5,6,7,8,9}; // a,b,c,d,e,f,g,dp
const uint8_t digitPins[4] = {10,11,12,13}; // DIG1..DIG4 (common anode:
HIGH = ON)

const uint8_t digitPatterns[10][7] = {
    {1,1,1,1,1,1,0}, //0
    {0,1,1,0,0,0,0}, //1
    {1,1,0,1,1,0,1}, //2
    {1,1,1,1,0,0,1}, //3
    {0,1,1,0,0,1,1}, //4
    {1,0,1,1,0,1,1}, //5
    {1,0,1,1,1,1,1}, //6
    {1,1,1,0,0,0,0}, //7
    {1,1,1,1,1,1,1}, //8
    {1,1,1,1,0,1,1} //9
};

volatile unsigned int numberToDisplay = 0; // 0..9999

// Button (INPUT_PULLUP)
const uint8_t buttonPin = A0;
int lastButtonReading = HIGH;
unsigned long lastDebounceTime = 0;
const unsigned long debounceDelay = 50UL; // ms
int buttonStateStable = HIGH;

void setup() {
    // segments as outputs
    for (uint8_t i = 0; i < 8; ++i) pinMode(segPins[i], OUTPUT);
    // digit control pins as outputs
    for (uint8_t i = 0; i < 4; ++i) pinMode(digitPins[i], OUTPUT);
    // turn all digits OFF (LOW for CA)
    for (uint8_t i = 0; i < 4; ++i) digitalWrite(digitPins[i], LOW);

    // button input with internal pull-up
    pinMode(buttonPin, INPUT_PULLUP);

    // initialize segments to OFF (for CA: HIGH = OFF)
    for (uint8_t i = 0; i < 8; ++i) digitalWrite(segPins[i], HIGH);
}

void displayOneDigit(uint8_t whichDigitPin, uint8_t num) {
    // Turn OFF all digits first (LOW for CA)
    for (uint8_t i = 0; i < 4; ++i) digitalWrite(digitPins[i], LOW);

    // Set segments for 'num' – NOTE: for COMMON ANODE, segment LOW = ON
    for (uint8_t s = 0; s < 7; ++s) {
        if (digitPatterns[num][s]) digitalWrite(segPins[s], LOW); // ON
        else digitalWrite(segPins[s], HIGH); // OFF
    }
    // decimal point left OFF
    digitalWrite(segPins[7], HIGH);
}
```

```

// Enable this digit (HIGH for CA)
digitalWrite(whichDigitPin, HIGH);
delay(3); // short on-time for persistence
// Disable digit again
digitalWrite(whichDigitPin, LOW);
}

void loop() {
  // Multiplex refresh - show all 4 digits quickly
  uint8_t thousands = numberToDisplay / 1000;
  uint8_t hundreds  = (numberToDisplay / 100) % 10;
  uint8_t tens       = (numberToDisplay / 10) % 10;
  uint8_t ones       = numberToDisplay % 10;

  displayOneDigit(digitPins[0], thousands);
  displayOneDigit(digitPins[1], hundreds);
  displayOneDigit(digitPins[2], tens);
  displayOneDigit(digitPins[3], ones);

  // Read button and debounce
  int reading = digitalRead(buttonPin);

  if (reading != lastButtonReading) {
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != buttonStateStable) {
      buttonStateStable = reading;
      // button pressed = LOW (because of INPUT_PULLUP)
      if (buttonStateStable == LOW) {
        numberToDisplay++;
        if (numberToDisplay > 9999) numberToDisplay = 0;
      }
    }
  }
  lastButtonReading = reading;
}

```

Output

After uploading the code:

Each press of the push button increments the displayed number by 1 (0000 → 0001 ... → 9999 → 0000)..

DIY Extension

- Add two buttons: one to increment and one to decrement.
- Long-press to auto-repeat increment (hold to speed up).
- Add buzzer click on each increment.

Project 18: Infrared Remote controlling LED and brightness.

✦ Objective: To control the **red, green, and blue LED's and their brightness** using an **IR remote**, where:

- Press button 1 → Red ON and 2 → Red OFF
- 3 / 4 → Green ON / OFF
- 5 / 6 → Blue ON / OFF
- VOL+ → Increase brightness (all colors)
- VOL- → Decrease brightness (all colors)

Components Required

- ✓ UNO R4 board
- ✓ Red, Green, Blue LED
- ✓ 3 × 220Ω resistors (one for each led)
- ✓ Jumper wires
- ✓ IR remote (Note: you need to insert CR2032 coin type battery into the remote which has to be purchased separately. **Not included in the kit**)
- ✓ IR Receiver
- ✓ Breadboard

Circuit Diagram

Component	Connection Method	Uno R4
Red LED (Anode, Long Leg +)	Through 220Ω resistor	Pin 9
Red LED (Cathode, Short Leg -)	Direct Connection	GND
Green LED (Anode, Long Leg +)	Through 220Ω resistor	Pin 10
Green LED (Cathode, Short Leg -)	Direct Connection	GND
Blue LED (Anode, Long Leg +)	Through 220Ω resistor	Pin 11
Blue LED (Cathode, Short Leg -)	Direct Connection	GND
Component	Connection Method	Uno R4
IR Sensor – OUT	Direct Connection	Pin 2
IR Sensor – VCC	Direct Connection	5V
IR Sensor – GND	Direct Connection	GND

Steps to Upload Code


```

#include <IRremote.h>

const int RECV_PIN = 2;

// RGB pins (use PWM pins)
const int RED_PIN   = 9;
const int GREEN_PIN = 10;
const int BLUE_PIN  = 11;

bool COMMON_ANODE = false; // set true if your RGB is common anode

// brightness values (0..255)
int redVal = 0;
int greenVal = 0;
int blueVal = 0;
const int STEP = 25; // brightness step

// ----- REPLACE THESE WITH YOUR REMOTE HEX CODES -----
// After running the scanner, paste the HEX values below (include 0x prefix).
// Example: const unsigned long CODE_1 = 0xFF6897;
const unsigned long CODE_1 = 0xF30CFF00; // BUTTON 1 - RED ON
const unsigned long CODE_2 = 0xE718FF00; // BUTTON 2 - RED OFF
const unsigned long CODE_3 = 0xA15EFF00; // BUTTON 3 - GREEN ON
const unsigned long CODE_4 = 0xF708FF00; // BUTTON 4 - GREEN OFF
const unsigned long CODE_5 = 0xE31CFF00; // BUTTON 5 - BLUE ON
const unsigned long CODE_6 = 0xA55AFF00; // BUTTON 6 - BLUE OFF
const unsigned long CODE_CHP = 0xEA15FF00; // VOL+ (increase)
const unsigned long CODE_CHM = 0xF807FF00; // VOL- (decrease)
// -----

void setup() {
  Serial.begin(115200);
  while (!Serial) {}
  Serial.println("IR RGB controller starting...");
  IrReceiver.begin(RECV_PIN, ENABLE_LED_FEEDBACK); // initialize receiver

  pinMode(RED_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);

  applyRGB(); // make sure outputs initialized
}

int clampInt(int v, int lo, int hi) {
  if (v < lo) return lo;
  if (v > hi) return hi;
  return v;
}

void applyRGB() {
  // If common anode, invert values
  if (COMMON_ANODE) {
    analogWrite(RED_PIN, 255 - clampInt(redVal, 0, 255));
    analogWrite(GREEN_PIN, 255 - clampInt(greenVal, 0, 255));
    analogWrite(BLUE_PIN, 255 - clampInt(blueVal, 0, 255));
  } else {
    analogWrite(RED_PIN, clampInt(redVal, 0, 255));
    analogWrite(GREEN_PIN, clampInt(greenVal, 0, 255));
  }
}

```

```

    analogWrite(BLUE_PIN, clampInt(blueVal, 0, 255));
}

Serial.print("RGB = ");
Serial.print(redVal); Serial.print(", ");
Serial.print(greenVal); Serial.print(", ");
Serial.println(blueVal);
}

void loop() {
    if (IrReceiver.decode()) {
        // prefer decodedRawData for full raw value (works for many remotes)
        unsigned long code = IrReceiver.decodedIRData.decodedRawData;

        // Some remotes report repeat as 0xFFFFFFFF; ignore repeat-only values
        if (code == 0xFFFFFFFFUL) {
            IrReceiver.resume();
            return;
        }

        Serial.print("IR code: 0x");
        Serial.println(code, HEX);

        // ----- Mapping actions -----

        // RED ON / OFF
        if (code == CODE_1) {
            redVal = 255;
            Serial.println("RED ON");
        } else if (code == CODE_2) {
            redVal = 0;
            Serial.println("RED OFF");
        }

        // GREEN ON / OFF
        else if (code == CODE_3) {
            greenVal = 255;
            Serial.println("GREEN ON");
        } else if (code == CODE_4) {
            greenVal = 0;
            Serial.println("GREEN OFF");
        }

        // BLUE ON / OFF
        else if (code == CODE_5) {
            blueVal = 255;
            Serial.println("BLUE ON");
        } else if (code == CODE_6) {
            blueVal = 0;
            Serial.println("BLUE OFF");
        }

        // Vol+ increase brightness for ALL channels
        else if (code == CODE_CHP) {
            redVal = clampInt(redVal + STEP, 0, 255);
            greenVal = clampInt(greenVal + STEP, 0, 255);
            blueVal = clampInt(blueVal + STEP, 0, 255);
            Serial.println("BRIGHTNESS +");
        }
    }
}

```

```

    // Vol- decrease brightness for ALL channels
    else if (code == CODE_CHM) {
        redVal  = clampInt(redVal  - STEP, 0, 255);
        greenVal = clampInt(greenVal - STEP, 0, 255);
        blueVal  = clampInt(blueVal - STEP, 0, 255);
        Serial.println("BRIGHTNESS -");
    }

    applyRGB();

    IrReceiver.resume(); // ready for the next code
    delay(120);          // small debounce
}
}

```

Explanation & Output

After uploading the code:

- Turns each color ON/OFF using buttons 1 to 6 in IR remote.
- Changes brightness for ALL colors using **VOL+** / **VOL-**.

Brightness Logic

- VOL+ adds STEP (default 25) to red, green, blue.
- VOL- subtracts STEP.
- Values stay within 0-255.

DIY Extension

◆ 1. Preset Mood Colors

Map remote buttons to:

Cyan
Magenta
Yellow
White

◆ 2. Fade Effect Mode

Use a remote button to start a smooth color fade animation.

Project 19: Dot Matrix 8x8 Display

🌟 Objective: To interface a **MAX7219 8×8 LED Dot Matrix** module with an **UNO R4**, and toggle between a **Smiley Face** and **Sad Face** using **two push buttons**: Press button 1 → Red ON and 2 → Red OFF

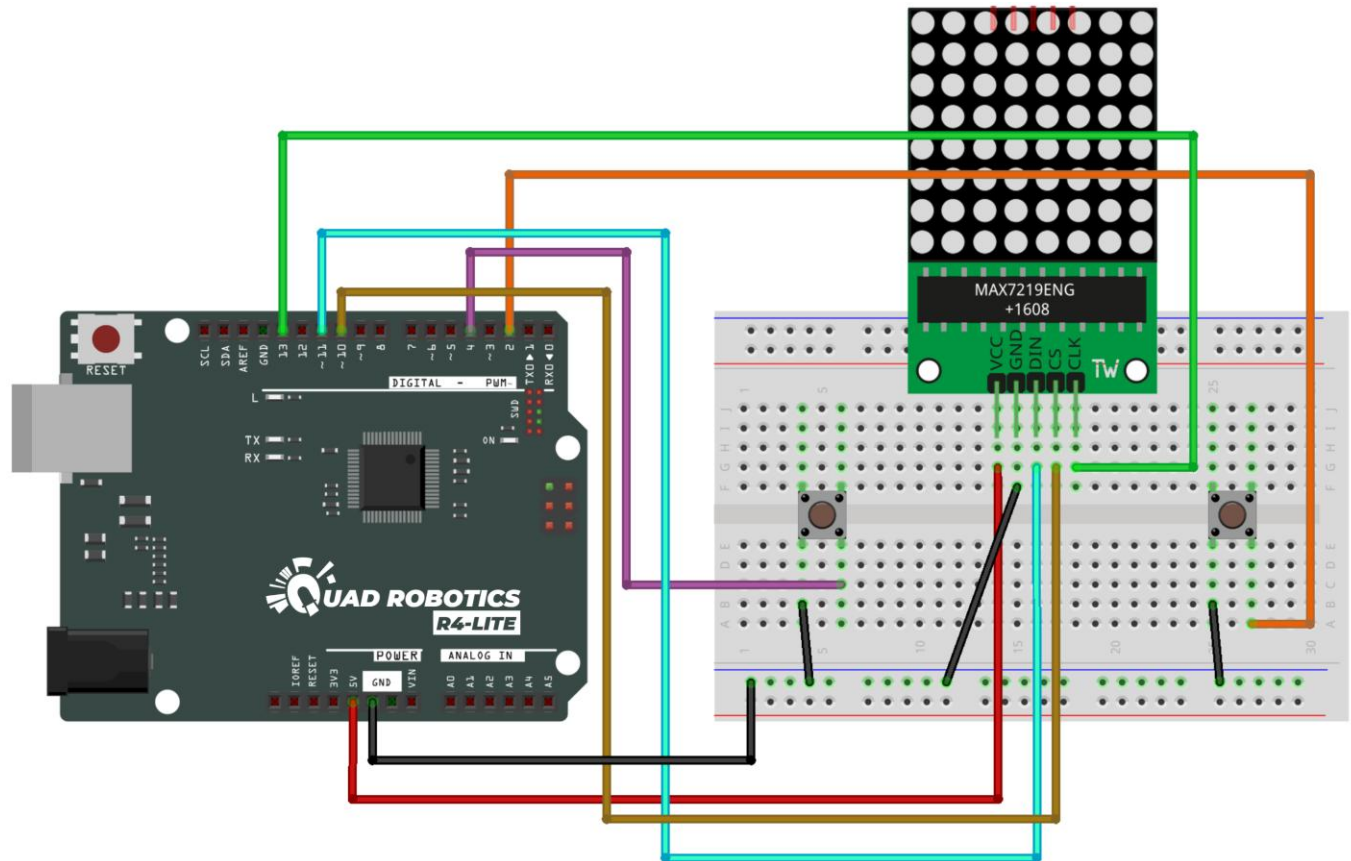
- Button 1 → Show Smiley 😊
- Button 2 → Show Sad Face 😞

🛒 Components Required

- ✓ UNO R4 board
- ✓ MAX7219 8×8 Dot Matrix Module
- ✓ 2 × Push Buttons
- ✓ Breadboard
- ✓ Male to Male Jumper wires

🔌 Circuit Diagram

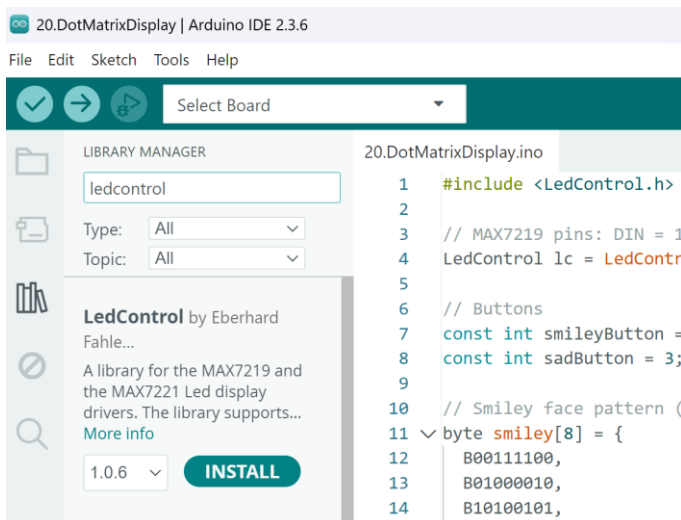
Component	Connection Method	Uno R4
MAX7219 – VCC	Direct Connection	5V
MAX7219 – GND	Direct Connection	GND
MAX7219 – DIN	Direct Connection	Pin D11
MAX7219 – CS	Direct Connection	Pin D10
MAX7219 – CLK	Direct Connection	Pin D13
Component	Connection Method	Uno R4
Push Button 1 (One leg)	Direct Connection	Pin 2
Push Button 1 (Other leg)	Direct Connection	GND
Push Button 2 (One leg)	Direct Connection	Pin 4
Push Button 2 (Other leg)	Direct Connection	GND



✓ Install Library:

Uses the **LedControl** library.

Arduino IDE → **Sketch** → **Include Library** → **Manage Libraries** → search **LedControl** → Install.



Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **19.DotMatrixDisplay.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Code

```
#include <LedControl.h>

// MAX7219 pins: DIN = 11, CLK = 13, CS = 10
LedControl lc = LedControl(11, 13, 10, 1);

// Buttons
const int smileyButton = 2;
const int sadButton = 4;

// Smiley face pattern (8x8)
byte smiley[8] = {
  B00111100,
  B01000010,
  B10100101,
  B10000001,
  B10100101,
  B10011001,
  B01000010,
  B00111100
};

// Sad face pattern (8x8)
byte sad[8] = {
  B00111100,
  B01000010,
  B10100101,
  B10000001,
  B10011001,
  B10100101,
  B01000010,
  B00111100
};

void setup() {
  lc.shutdown(0, false); // Wake up MAX7219
  lc.setIntensity(0, 8); // Brightness 0-15
  lc.clearDisplay(0);
```

```

pinMode(smileyButton, INPUT_PULLUP);
pinMode(sadButton, INPUT_PULLUP);

showSmiley(); // Default on startup
}

void showPattern(byte pattern[8]) {
  for (int row = 0; row < 8; row++) {
    lc.setRow(0, row, pattern[row]);
  }
}

void showSmiley() { showPattern(smiley); }
void showSad()    { showPattern(sad);   }

void loop() {
  if (digitalRead(smileyButton) == LOW) {
    delay(50); // debounce
    showSmiley();
  }

  if (digitalRead(sadButton) == LOW) {
    delay(50); // debounce
    showSad();
  }
}

```

Output

After uploading the code:

- On power-up, the smiley face 😊 appears.
- When Button 1 is pressed → Smiley face appears.
- When Button 2 is pressed → Sad face 😞 appears.

DIY Extension

◆ Add more emotions

Buttons for:

- Wink 😜
- Heart ❤️
- Angry 😡
- Neutral 😐

Mini Project -1: Automated Gate/Smart Dustbin

✦ Objective: To make an automated gate for a miniature model using ultrasonic distance indicator so that when an object comes within a threshold the system:

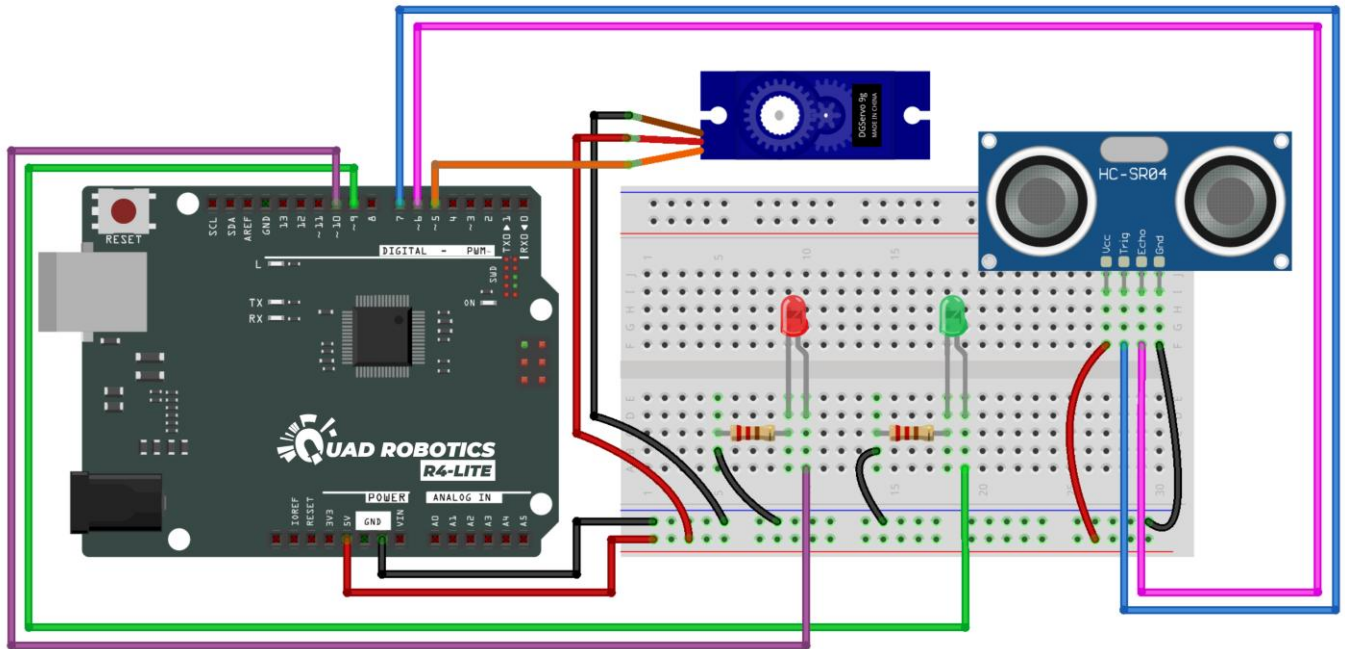
- turns Red LED ON,
- moves a servo to *open* the gate (e.g., rotate to opening angle), and when the object is away:
- turns Green LED ON,
- moves the servo to *close* the gate.

Components Required

- ✓ UNO R4
- ✓ HC-SR04 ultrasonic sensor
- ✓ 1 × Red LED + 220Ω resistor
- ✓ 1 × Green LED + 220Ω resistor
- ✓ 1 × SG90 Servo Motor
- ✓ Jumper wires
- ✓ Breadboard
- ✓ USB cable for Arduino

Circuit Diagram

Component - Ultrasonic Sensor	Connection Method	Uno R4
Ultrasonic Sensor – VCC	Direct Connection	5V
Ultrasonic Sensor – GND	Direct Connection	GND
Ultrasonic Sensor – TRIG	Direct Connection	Pin D7
Ultrasonic Sensor – ECHO	Direct Connection	Pin D6
Component - LED	Connection Method	Uno R4
Green LED (Anode, Long Leg +)	Through 220Ω resistor	Pin D9
Green LED (Cathode, Short Leg –)	Direct Connection	GND
Red LED (Anode, Long Leg +)	Through 220Ω resistor	Pin D10
Red LED (Cathode, Short Leg –)	Direct Connection	GND
Component - Servo Motor	Connection Method	Uno R4
Servo Signal (Orange wire)	Direct Connection	Pin D5
Servo VCC (Red wire)	External 5V supply (<i>recommended</i>)	5V
Servo GND (Brown wire)	Common Ground	GND



Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **1.MiniProject1.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Code

```
// UNO R4 - Ultrasonic + LEDs + Servo Gate
#include <Servo.h>

// Pins
const uint8_t TRIG_PIN = 7;
const uint8_t ECHO_PIN = 6;
const uint8_t GREEN_PIN = 9;
const uint8_t RED_PIN = 10;
const uint8_t SERVO_PIN = 5;

// Servo settings
Servo gateServo;
const int CLOSED_ANGLE = 0;    // servo angle when gate closed
```

```

const int OPEN_ANGLE    = 90;    // servo angle when gate open
const unsigned int SERVO_STEP_DELAY = 12; // ms between small steps for smooth
motion

// Ultrasonic settings
const unsigned long SENSOR_TIMEOUT = 30000UL; // microseconds
const float CM_PER_US = 0.0343 / 2.0; // speed of sound factor (cm/us /2)
const float THRESHOLD_CM = 15.0; // trigger distance in cm

// State tracking
bool gateOpen = false;
float lastDistance = -1;

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(RED_PIN, OUTPUT);

  digitalWrite(TRIG_PIN, LOW);
  digitalWrite(GREEN_PIN, LOW);
  digitalWrite(RED_PIN, LOW);

  Serial.begin(115200);
  delay(100);
  Serial.println("Ultrasonic + Servo Gate starting...");

  gateServo.attach(SERVO_PIN);
  // initialize gate closed
  setServoAngleSmooth(CLOSED_ANGLE);
  gateOpen = false;
  showGreen();
}

float readDistanceCM() {
  // Trigger 10us pulse
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  unsigned long duration = pulseIn(ECHO_PIN, HIGH, SENSOR_TIMEOUT);
  if (duration == 0) return -1.0; // timeout/no echo
  float dist = duration * CM_PER_US;
  return dist;
}

void showGreen() {
  digitalWrite(GREEN_PIN, HIGH);
  digitalWrite(RED_PIN, LOW);
}

void showRed() {
  digitalWrite(GREEN_PIN, LOW);
  digitalWrite(RED_PIN, HIGH);
}

void setServoAngleSmooth(int targetAngle) {
  int current = gateServo.read(); // get current angle

```

```

// If servo returns 255 (unknown), set it to target immediately
if (current == 255) {
    gateServo.write(targetAngle);
    delay(300);
    return;
}
if (current < targetAngle) {
    for (int a = current; a <= targetAngle; a++) {
        gateServo.write(a);
        delay(SERVO_STEP_DELAY);
    }
} else if (current > targetAngle) {
    for (int a = current; a >= targetAngle; a--) {
        gateServo.write(a);
        delay(SERVO_STEP_DELAY);
    }
}
// small settle delay
delay(80);
}

void openGate() {
    if (!gateOpen) {
        Serial.println("Opening gate...");
        setServoAngleSmooth(OPEN_ANGLE);
        gateOpen = true;
    }
}

void closeGate() {
    if (gateOpen) {
        Serial.println("Closing gate...");
        setServoAngleSmooth(CLOSED_ANGLE);
        gateOpen = false;
    }
}

void loop() {
    float dist = readDistanceCM();

    if (dist < 0) {
        Serial.println("No echo");
        // optional: indicate unknown by blinking both LEDs briefly
        digitalWrite(GREEN_PIN, LOW);
        digitalWrite(RED_PIN, LOW);
    } else {
        Serial.print("Distance: ");
        Serial.print(dist, 1);
        Serial.println(" cm");
        lastDistance = dist;

        if (dist <= THRESHOLD_CM) {
            // object near -> red + open gate
            showRed();
            openGate();
        } else {
            // clear -> green + close gate
            showGreen();
            closeGate();
        }
    }
}

```

```
}  
  
delay(120); // measurement pacing  
}
```

Output

After uploading the code:

- On power-up: gate closed, Green LED ON.
- When an object moves within the threshold (e.g., car toy or hand): Red LED ON, servo rotates to open gate (smooth motion).
- When object moves away: Green LED ON, servo rotates back to closed position.
- Serial monitor prints distances and actions.

NOTE: The same wiring setup and code can be used for Smart Dustbin project as well. The concept is the same.

DIY Extension

◆ Add more emotions

- **Manual override:** add a push button to manually open/close the gate.
- **Safety:** add a limit switch to detect fully open/closed positions.
- **Add buzzer:** beep when gate is opening or when object detected.

Mini Project -2: Digital Locker

🌟 Objective: Build a digital password-protected smart safe where:

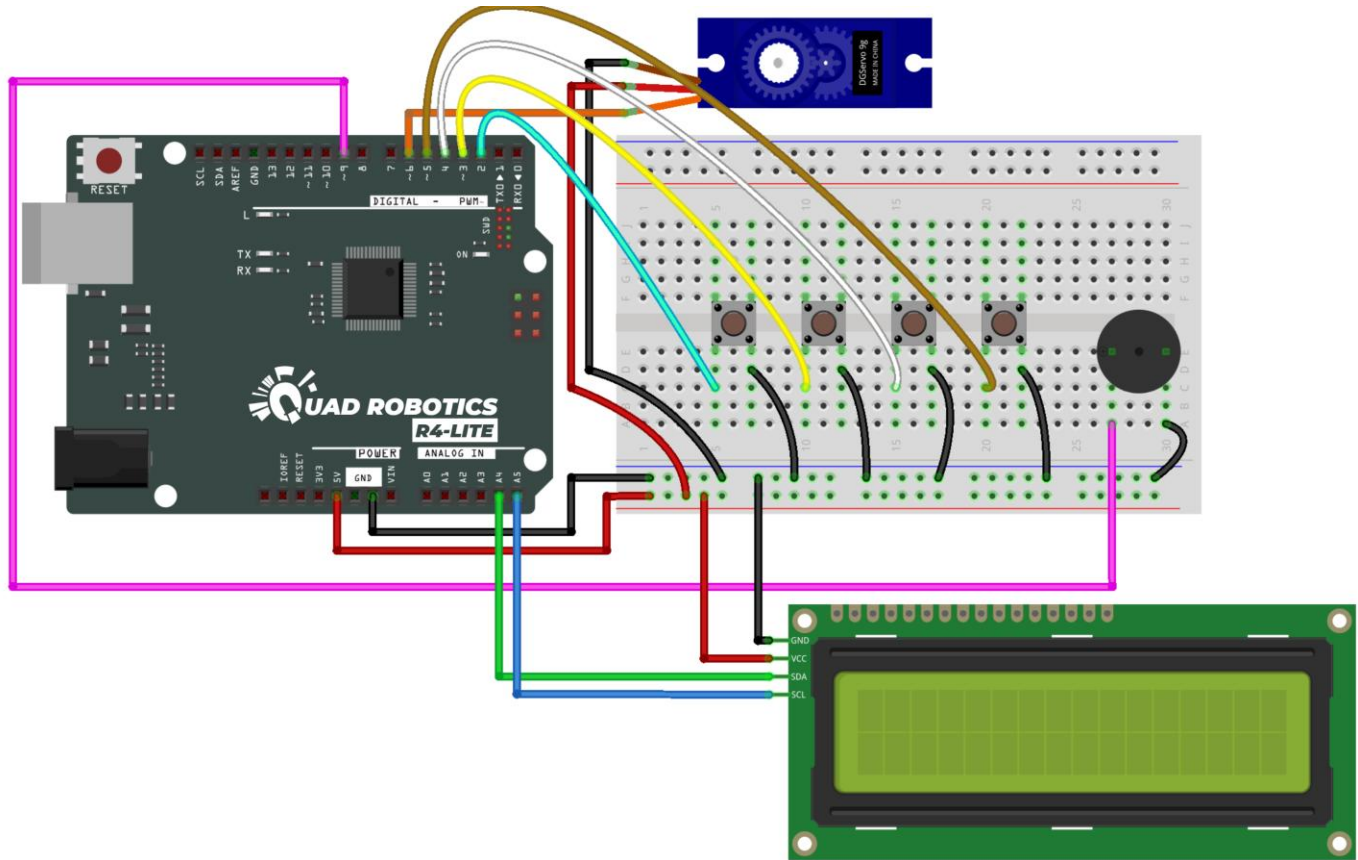
- User enters a 4-digit password using push buttons
- LCD guides the user
- Correct password → Servo unlocks the box
- Wrong password → buzzer alert + message on LCD

Components Required

- ✓ UNO R4
- ✓ 4 × Push buttons
- ✓ I2C 1602 LCD
- ✓ Servo motor (SG90)
- ✓ Active buzzer
- ✓ Jumper wires
- ✓ Breadboard

Circuit Diagram

Component - Push Buttons	Connection Method	Uno R4
Button B1	Direct Connection	Pin D2
Button B1 (Other Leg)	Direct Connection	GND
Button B2	Direct Connection	Pin D3
Button B2 (Other Leg)	Direct Connection	GND
Button B3	Direct Connection	Pin D4
Button B3 (Other Leg)	Direct Connection	GND
Button B4	Direct Connection	Pin D5
Button B4 (Other Leg)	Direct Connection	GND
Component - LCD	Connection Method	Uno R4
I2C LCD – VCC	Direct Connection	5V
I2C LCD – GND	Direct Connection	GND
I2C LCD – SDA	Direct Connection	A4
I2C LCD – SCL	Direct Connection	A5
Component - Servo Motor	Connection Method	Uno R4
Servo Signal (Orange wire)	Direct Connection	Pin D6
Servo VCC (Red wire)	Direct Connection	5V
Servo GND (Brown wire)	Common Ground	GND
Component - Buzzer	Connection Method	Uno R4
Buzzer Signal (+)	Direct Connection	Pin D9
Buzzer GND (–)	Direct Connection	GND



📖 Steps to Upload Code

1. Connect your **UNO R4** board to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **2.MiniProject2.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Code

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

LiquidCrystal_I2C lcd(0x27, 16, 2);
Servo lockServo;

int buttons[4] = {2, 3, 4, 5};
int password[4] = {1, 2, 3, 4};
int entered[4];
int indexPos = 0;

const int buzzer = 9;
const int servoPin = 6;

void setup() {
  // LCD
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("SMART SAFE BOX");
  lcd.setCursor(0, 1);
  lcd.print("ENTER CODE...");
  delay(1500);
  lcd.clear();

  // Buttons
  for (int i = 0; i < 4; i++)
    pinMode(buttons[i], INPUT_PULLUP);

  // Servo
  lockServo.attach(servoPin);
  lockServo.write(0); // locked

  // Buzzer
  pinMode(buzzer, OUTPUT);
}

bool checkPassword() {
  for (int i = 0; i < 4; i++) {
    if (entered[i] != password[i]) return false;
  }
  return true;
}

void loop() {
  lcd.setCursor(0, 0);
  lcd.print("ENTER CODE:  ");
  lcd.setCursor(0, 1);
  lcd.print("Keys: ");
  lcd.print(indexPos);
  lcd.print("    ");

  for (int i = 0; i < 4; i++) {
    if (digitalRead(buttons[i]) == LOW) {
      delay(200);
      entered[indexPos] = i + 1;
    }
  }
}
```

```

        indexPos++;

        if (indexPos == 4) {
            lcd.clear();
            if (checkPassword()) {
                lcd.setCursor(0, 0);
                lcd.print("CORRECT CODE!");
                lcd.setCursor(0, 1);
                lcd.print("SAFE UNLOCKED");
                lockServo.write(90); // unlock
            } else {
                lcd.setCursor(0, 0);
                lcd.print("WRONG CODE");
                lcd.setCursor(0, 1);
                lcd.print("ACCESS DENIED");
                tone(buzzer, 600, 400);
                lockServo.write(0); // ensure locked
            }

            delay(2000);
            lcd.clear();
            indexPos = 0;
        }
    }
}
}

```

Output

After uploading the code:

- 4 push buttons simulate numeric keypad digits as 1, 2, 3, 4.
- User enters 4 digits → stored in entered[].
- Program compares entered digits with password {1,2,3,4}.
- LCD displays real-time prompts and status.
- Correct password → servo rotates to open the safe (90°).
- Wrong password → error message + buzzer alert.

DIY Extension

◆ Add more emotions

- Add "change password" option
- Add IR remote password input
- Add attempt counter → lockout after 3 wrong tries
- Add a vibration sensor → tamper detection
- Add MAX7219 matrix for fancy animations
- Add buzzer beep for each button press.

Mini Project -3: Smart Weather Station

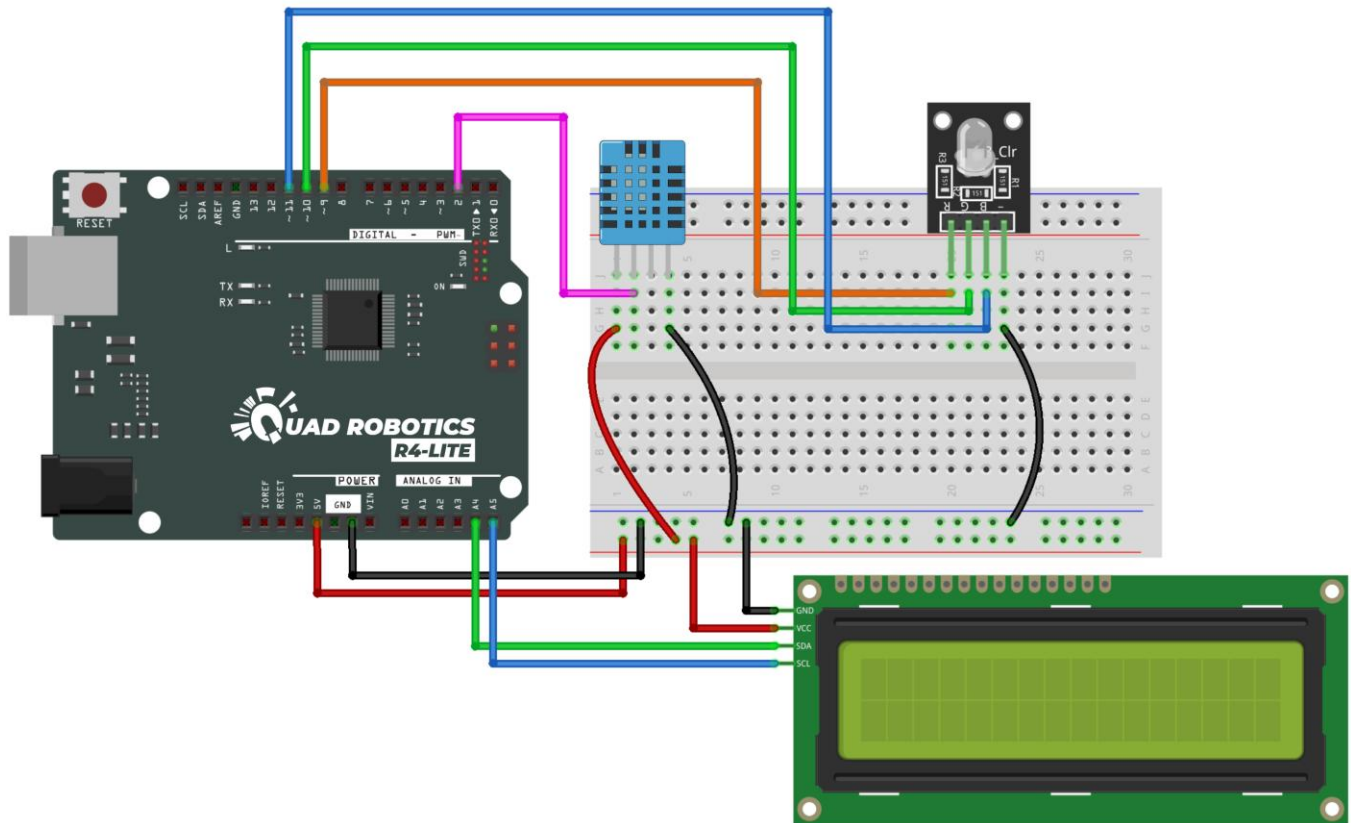
✦ Objective: Build a mini weather station that shows temperature & humidity on the LCD and uses RGB LED to show climate condition.

Components Required

- ✓ UNO R4
- ✓ DHT11 sensor
- ✓ I2C 1602 LCD
- ✓ RGB LED
- ✓ Jumper wires

Circuit Diagram

Component - DHT11 Temperature Sensor	Connection Method	Uno R4
DHT11 – Signal	Direct Connection	Pin D2
DHT11 – VCC	Direct Connection	5V
DHT11 – GND	Direct Connection	GND
Component - LCD Display	Connection Method	Uno R4
I2C LCD – VCC	Direct Connection	5V
I2C LCD – GND	Direct Connection	GND
I2C LCD – SDA	Direct Connection	A4
I2C LCD – SCL	Direct Connection	A5
Component - RGB Led	Connection Method	Uno R4
RGB LED – Red (R)	Direct Connection	Pin D9
RGB LED – Green (G)	Direct Connection	Pin D10
RGB LED – Blue (B)	Direct Connection	Pin D11
RGB LED – Common Cathode (-)	Direct Connection	GND



📖 Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **3.MiniProject3.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

💻 Code

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

LiquidCrystal_I2C lcd(0x27, 16, 2);

#define DHTPIN 2
#define DHTTYPE DHT11
```

```

DHT dht(DHTPIN, DHTTYPE);

int R = 9, G = 10, B = 11;

void setColor(int r, int g, int b) {
  analogWrite(R, r);
  analogWrite(G, g);
  analogWrite(B, b);
}

void setup() {
  lcd.init();
  lcd.backlight();
  dht.begin();
  pinMode(R, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(B, OUTPUT);
}

void loop() {
  float t = dht.readTemperature();
  float h = dht.readHumidity();

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temp: "); lcd.print(t); lcd.print("C");
  lcd.setCursor(0, 1);
  lcd.print("Hum : "); lcd.print(h); lcd.print("%");

  if (t < 20) setColor(0, 0, 255);      // cold - blue
  else if (t > 30) setColor(255, 0, 0); // hot - red
  else setColor(0, 255, 0);            // ideal - green

  delay(1500);
}

```

Output

After uploading the code:

- LCD updates every 1.5 seconds.
- RGB LED shows climate condition.

DIY Extension

◆ Add more emotions

- Add buzzer alarm for extreme heat
- Add MAX7219 scrolling temperature
- Add humidity-controlled fan

Mini Project -4: Reaction Time Game

✦ Objective: Measure a player's reaction time when the dot-matrix flashes; show the result on the **I2C 16×2 LCD** and give audio feedback via the buzzer. Matrix provides the visual start cue.

Components Required

- ✓ UNO R4
- ✓ 1 × Push buttons
- ✓ MAX7219 dot matrix display
- ✓ I2C 1602 LED display
- ✓ Active Buzzer
- ✓ Jumper wires
- ✓ Breadboard

Circuit Diagram

Component - MAX7219 8x8 LED Matrix	Connection Method	Uno R4
MAX7219 – VCC	Direct Connection	5V
MAX7219 – GND	Direct Connection	GND
MAX7219 – DIN	Direct Connection	Pin D11
MAX7219 – CS	Direct Connection	Pin D10
MAX7219 – CLK	Direct Connection	Pin D13
Component - LCD Display	Connection Method	Uno R4
I2C LCD – VCC	Direct Connection	5V
I2C LCD – GND	Direct Connection	GND
I2C LCD – SDA	Direct Connection	A4
I2C LCD – SCL	Direct Connection	A5
Component - Push Buttons	Connection Method	Uno R4
Push Button (One Leg)	Direct Connection	Pin D2
Push Button (Other Leg)	Direct Connection	GND
Component - Buzzer	Connection Method	Uno R4
Buzzer (Positive +)	Direct Connection	Pin D9
Buzzer (Negative –)	Direct Connection	GND

- ## Steps to Upload Code

Code

```
// Reaction Time Game: MAX7219 + I2C 1602 LCD + Button + Buzzer
// Libraries: LedControl, LiquidCrystal_I2C
#include <LedControl.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LedControl lc = LedControl(11, 13, 10, 1); // DIN=11, CLK=13, CS=10, 1 device
LiquidCrystal_I2C lcd(0x27, 16, 2);        // change address if your module
differs

const int buttonPin = 2;    // player button (INPUT_PULLUP)
const int buzzerPin = 9;

const unsigned long MIN_DELAY = 1500;
const unsigned long MAX_DELAY = 4000;

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(buzzerPin, OUTPUT);
  digitalWrite(buzzerPin, LOW);

  // Initialize MAX7219
  lc.shutdown(0, false);
  lc.setIntensity(0, 8);    // 0..15
  lc.clearDisplay(0);

  // Initialize LCD
  lcd.init();
  lcd.backlight();

  // Welcome screen
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Reaction Game");
  lcd.setCursor(0,1);
  lcd.print("Press button...");
  delay(1500);
  lcd.clear();
  randomSeed(analogRead(A0));
}

// small test pattern: flash center cross
void showReadyCue() {
  byte cue[8] = {
    B00011000,
    B00011000,
    B00111100,
    B01111110,
    B01111110,
    B00111100,
    B00011000,
    B00011000
  };
  for (int r=0; r<8; r++) lc.setRow(0, r, cue[r]);
}

void clearMatrix() {
```

```

    lc.clearDisplay(0);
}

// map reaction (ms) to 0..8 bars (faster -> more bars)
int reactionToBars(unsigned long reactionMs) {
    const unsigned long FAST_MS = 100;    // bright best time
    const unsigned long SLOW_MS = 2000;   // slow worst time
    if (reactionMs <= FAST_MS) return 8;
    if (reactionMs >= SLOW_MS) return 0;
    // linear mapping
    float frac = 1.0 - (float)(reactionMs - FAST_MS) / (SLOW_MS - FAST_MS);
    int bars = (int)round(frac * 8.0);
    if (bars < 0) bars = 0;
    if (bars > 8) bars = 8;
    return bars;
}

// draw bars on leftmost columns of matrix
void drawBars(int count) {
    // Light columns 0..(count-1) full height
    clearMatrix();
    if (count > 8) count = 8;    // safety

    for (int c = 0; c < count; c++) {
        for (int r = 0; r < 8; r++) {
            // setLed(device, row, column, state)
            lc.setLed(0, r, c, true);
        }
    }
}

void showFalseStart() {
    clearMatrix();
    // small cross to indicate false start
    byte cross[8] = {
        B10000001,
        B01000010,
        B00100100,
        B00011000,
        B00011000,
        B00100100,
        B01000010,
        B10000001
    };
    for (int r=0; r<8; r++) lc.setRow(0, r, cross[r]);
}

void loop() {
    // Wait a random time, but watch for false starts
    unsigned long waitTime = random(MIN_DELAY, MAX_DELAY);
    unsigned long startWait = millis();
    lcd.setCursor(0,0);
    lcd.print("Get Ready...    ");
    lcd.setCursor(0,1);
    lcd.print("Don't press early ");
    clearMatrix();

    // During waiting period detect false start
    bool falseStart = false;

```

```

while (millis() - startWait < waitTime) {
    if (digitalRead(buttonPin) == LOW) {
        // debounce simple
        delay(30);
        if (digitalRead(buttonPin) == LOW) {
            falseStart = true;
            break;
        }
    }
}

if (falseStart) {
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("False Start!");
    lcd.setCursor(0,1);
    lcd.print("Wait for cue...");
    tone(buzzerPin, 400, 250);
    showFalseStart();
    delay(1000);
    clearMatrix();
    lcd.clear();
    return; // restart loop
}

// Show the cue
showReadyCue();
lcd.clear();
lcd.setCursor(0,0);
lcd.print("GO! Press NOW");
unsigned long cueTime = millis();

// Wait for button press and record reaction
while (true) {
    if (digitalRead(buttonPin) == LOW) {
        // debounce
        delay(20);
        if (digitalRead(buttonPin) == LOW) {
            unsigned long reaction = millis() - cueTime;
            // beep
            tone(buzzerPin, 1000, 120);
            // Display on LCD
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("Reaction:");
            lcd.setCursor(10,0);
            lcd.print(reaction);
            lcd.print("ms");
            // show small bar graph on matrix
            int bars = reactionToBars(reaction);
            drawBars(bars);
            // also show feedback message
            lcd.setCursor(0,1);
            if (bars >= 6) lcd.print("Excellent!      ");
            else if (bars >= 3) lcd.print("Good          ");
            else lcd.print("Try again!      ");

            // wait until button released to avoid multiple readings
            while (digitalRead(buttonPin) == LOW) delay(10);
        }
    }
}

```



```
        delay(1200); // let player see result
        clearMatrix();
        lcd.clear();
        break;
    }
}

// small pause between rounds
delay(400);
}
```

Output

After uploading the code:

On the MAX7219 matrix: a bar with e.g. 6/8 columns lit indicating a pretty quick reaction. If the player presses early, LCD shows False Start! Wait for cue... and the matrix shows a cross.

DIY Extension

◆ Add more emotions

- wo-player mode: alternate cues and show both players' times.
- Use buzzer patterns or MAX7219 animations corresponding to performance tiers.
- Add a secondary button for player start/ready instead of auto-wait.

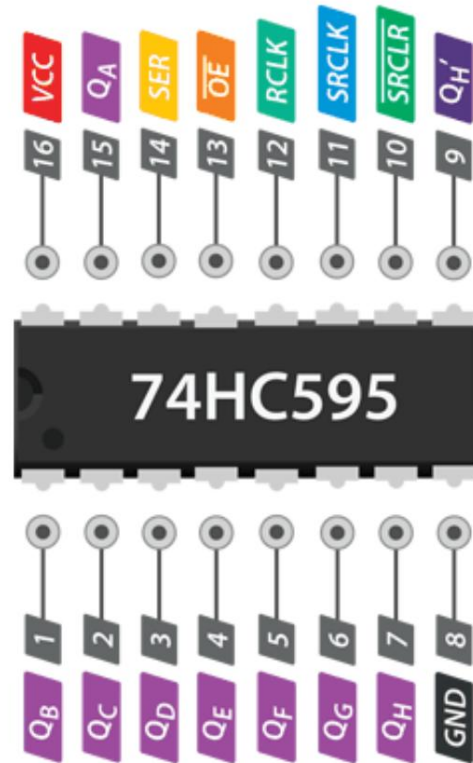
Mini Project -5: LED Direction Bar

✦ Objective: Display a smooth bar-graph animation (LEDs light progressively from 1→7 then back 7→1) using a single 74HC595 shift register driven by an UNO R4. Good demo for shift-register concepts and efficient use of IO pins.

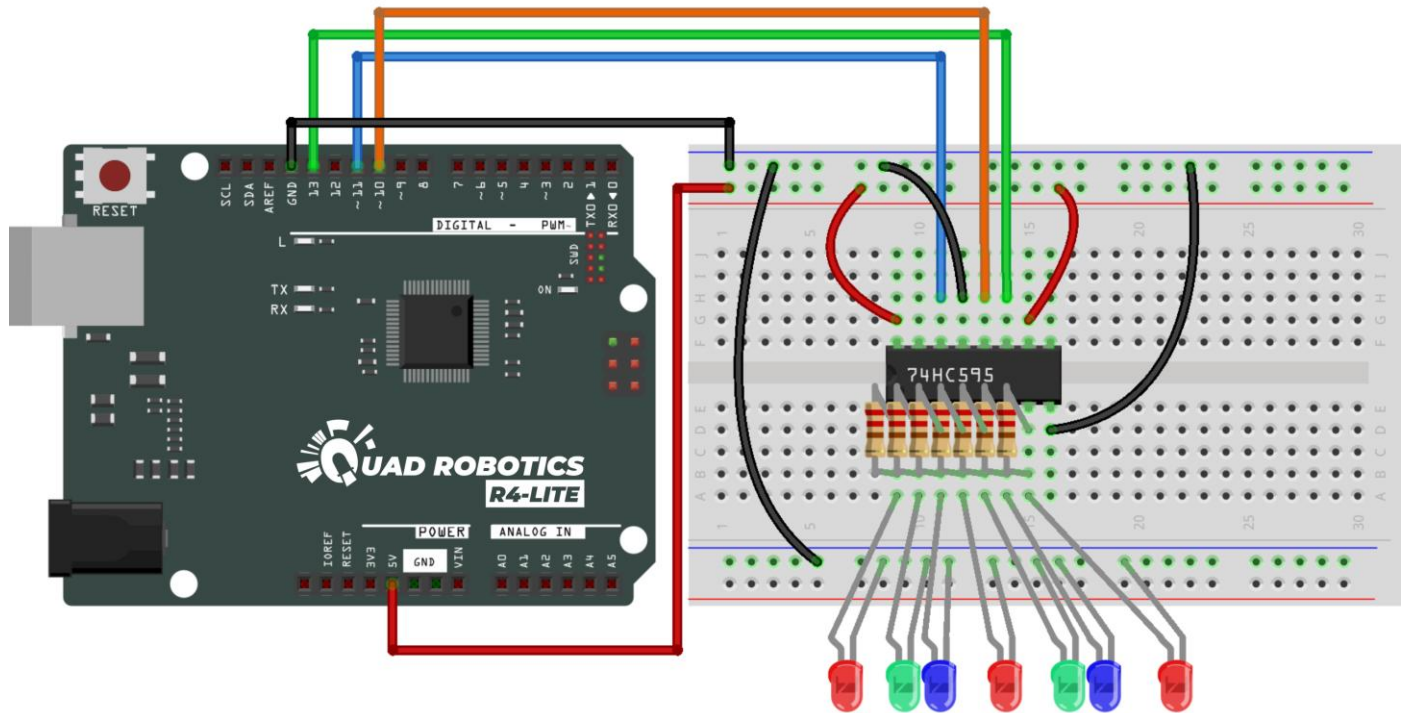
📦 Components Required

- ✓ UNO R4 board
- ✓ 1 × 74HC595 (8-bit shift register)
- ✓ 7 × LEDs (any color)
- ✓ 7 × 220 Ω resistors (one per LED)
- ✓ Breadboard & jumper wires

💡 Circuit Diagram



Component - 74HC595 Pin	Connection Method	Uno R4
74HC595 VCC (Pin 16)	Direct connection (+5V)	5V
74HC595 GND (Pin 8)	Direct connection (Ground)	GND
SER / DS (Pin 14)	Serial data input	D11
SRCLK / SH_CP (Pin 11)	Shift clock	D13
RCLK / ST_CP (Pin 12)	Latch clock	D10
SRCLR / MR (Pin 10)	Tie HIGH to disable reset	5V
OE (Pin 13)	Tie LOW to enable outputs	GND
QH' (Pin 9)	Cascade output (optional)	Not connected
Component - Led's	Connection Method	74HC595 Pin
LED 1 (Anode +)	Through 220Ω resistor	QB (Pin 1)
LED 2 (Anode +)	Through 220Ω resistor	QC (Pin 2)
LED 3 (Anode +)	Through 220Ω resistor	QD (Pin 3)
LED 4 (Anode +)	Through 220Ω resistor	QE (Pin 4)
LED 5 (Anode +)	Through 220Ω resistor	QF (Pin 5)
LED 6 (Anode +)	Through 220Ω resistor	QG (Pin 6)
LED 7 (Anode +)	Through 220Ω resistor	QH (Pin 7)
All LEDs (Cathode -)	Direct Connection	GND (Pin 8)



Steps to Upload Code

1. Connect your **UNO R4 board** to the computer using a **USB cable**.
2. Open **Arduino IDE** on your computer.
3. Go to **Tools > Board** and select **Arduino UNO R4**.
4. Go to **Tools > Port** and choose the correct **COM port**.
5. Either **Copy & Paste above code** into the Arduino IDE (OR) open file **5.MiniProject5.ino** from the provided **CODE** folder.
6. **NOTE:** If you open the file directly from CODE folder, you will need to select the Board again from the drop-down list. Select as **Arduino Uno R4 minima** from drop down list.
7. Click on the **Upload button** (right arrow icon).
8. Wait for the code to **compile and upload**. The board will start running the program automatically.

Code

```
// Bar-graph mode using 74HC595
// Arduino pins
const int dataPin = 11; // DS (74HC595 pin 14)
const int clockPin = 13; // SH_CP (74HC595 pin 11)
const int latchPin = 10; // ST_CP (74HC595 pin 12)

// Optional: push button to toggle behavior (not required)
const int buttonPin = 2; // use INPUT_PULLUP if you wire a button

const int STEP_DELAY = 180; // ms between steps

// Write one byte to the 74HC595 (LSB -> Q0)
void shiftWrite(byte value) {
    digitalWrite(latchPin, LOW);
    // Use LSBFIRST so bit0 -> Q0 (LED1), bit7 -> Q7 (LED8)
    shiftOut(dataPin, clockPin, LSBFIRST, value);
    digitalWrite(latchPin, HIGH);
}

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

    // Optional button
    pinMode(buttonPin, INPUT_PULLUP);

    // Clear outputs
    shiftWrite(0x00);
}

void loop() {
    // Grow from 0 to 8 LEDs
    for (int i = 1; i <= 8; i++) {
        byte pattern = 0;
        for (int b = 0; b < i; b++) {
            pattern |= (1 << b); // set bit b (LSB = LED1)
        }
        shiftWrite(pattern);
        delay(STEP_DELAY);
        // If button pressed, you could break/modify behavior here (optional)
    }

    // Pause at full
    delay(300);

    // Shrink from 8 down to 0
    for (int i = 8; i >= 0; i--) {
        byte pattern = 0;
        for (int b = 0; b < i; b++) {
            pattern |= (1 << b);
        }
        shiftWrite(pattern);
        delay(STEP_DELAY);
    }

    // Optional pause
```

```
delay(300);  
}
```

Output

After uploading the code:

- LEDs light progressively from LED1 → LED7 one by one until all are ON.
- After a brief pause the LEDs turn off one by one (shrinking).
- The animation repeats continuously.
- Step speed is controlled by STEP_DELAY (increase to slow down, decrease to speed up).

DIY Extension

Add more emotions

- **Mode button:** Press to switch between Bar Graph, Running Dot, and Knight Rider styles.
- **Ultrasonic input:** Use HC-SR04 distance reading to display the proximity as bar height (mapping distance to 0..8).