Thank You for Choosing the

Inventor Lab Kit

Embark on your electronics journey with the Inventor Lab Kit, designed for learners and enthusiasts. Centered around the Arduino Uno R3, this kit includes everything from basic components like LEDs and buzzers to advanced modules such as RFID systems and ultrasonic sensors, along with a mini-multimeter for circuit analysis.

The kit excels in educational clarity, gradually introducing beginners to programming and circuit design. Instead of pre-written code, it guides you step-by-step through writing your own scripts, enhancing understanding and retention. Projects range from simple tasks like lighting LEDs and using a joystick, to complex applications such as building a radar system and an automatic soap dispenser.

Ideal for both beginners and those looking to expand their skills, the Inventor Lab Kit makes learning both accessible and exciting, providing all the tools needed to explore and innovate in the world of electronics.

For any inquiries or support, please reach out to us at service@sunfounder.com. Dive into your learning journey with the Beginner's Lab Kit and start building, coding, and exploring the exciting world of electronics!

Contents

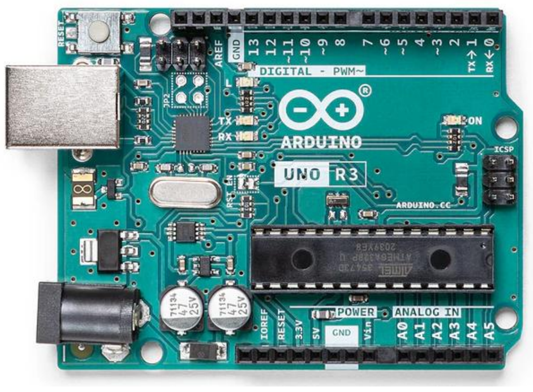
What’s Included in Your Kit

Inside our kit, you’ll find a variety of components and parts you’ll use throughout this course to build circuits. Here’s a quick guide to what’s included.

Basic Components

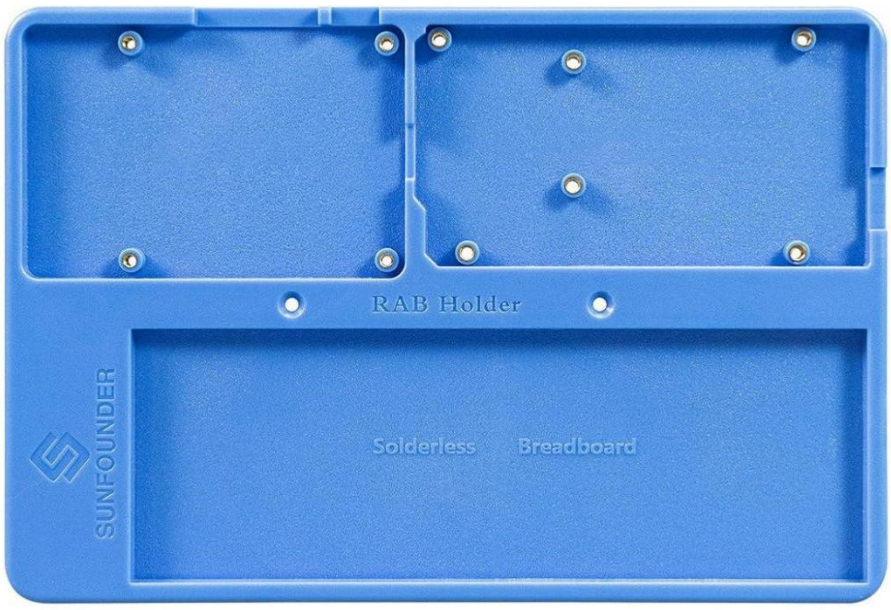
1 x Original Arduino Uno R3

A microcontroller board that’s the brain of your circuits. It has everything needed to support the microcontroller; simply connect it to your computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.



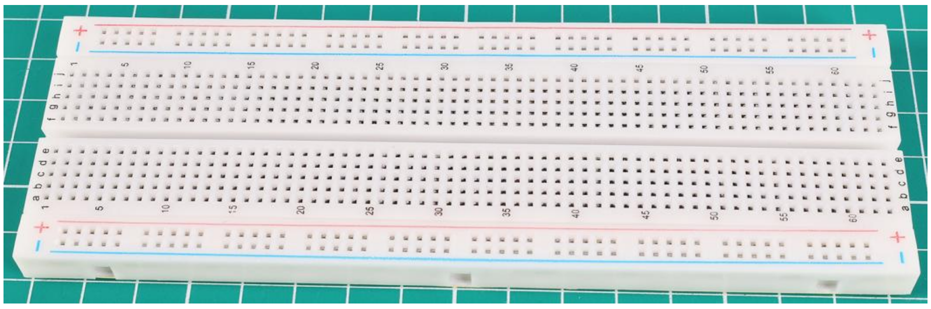
1 x RAB Holder

Designed for both Arduino and Raspberry Pi users, it ensures stability for all your devices. Beyond holding, it’s a tool for innovation, suitable for everyone.



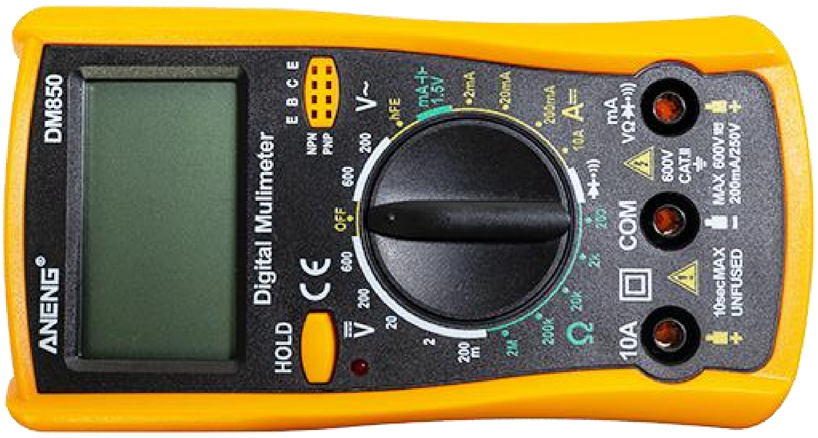
1 x 830-hole Breadboard

A solderless board that lets you easily build electronic circuits. It’s filled with rows of holes for connecting wires and components.



1 x Multimeter with Red & Black Leads

This is a versatile multimeter capable of measuring voltage, current, and resistance, as well as performing other electrical tests, making it an indispensable tool for electronics and electrical work.

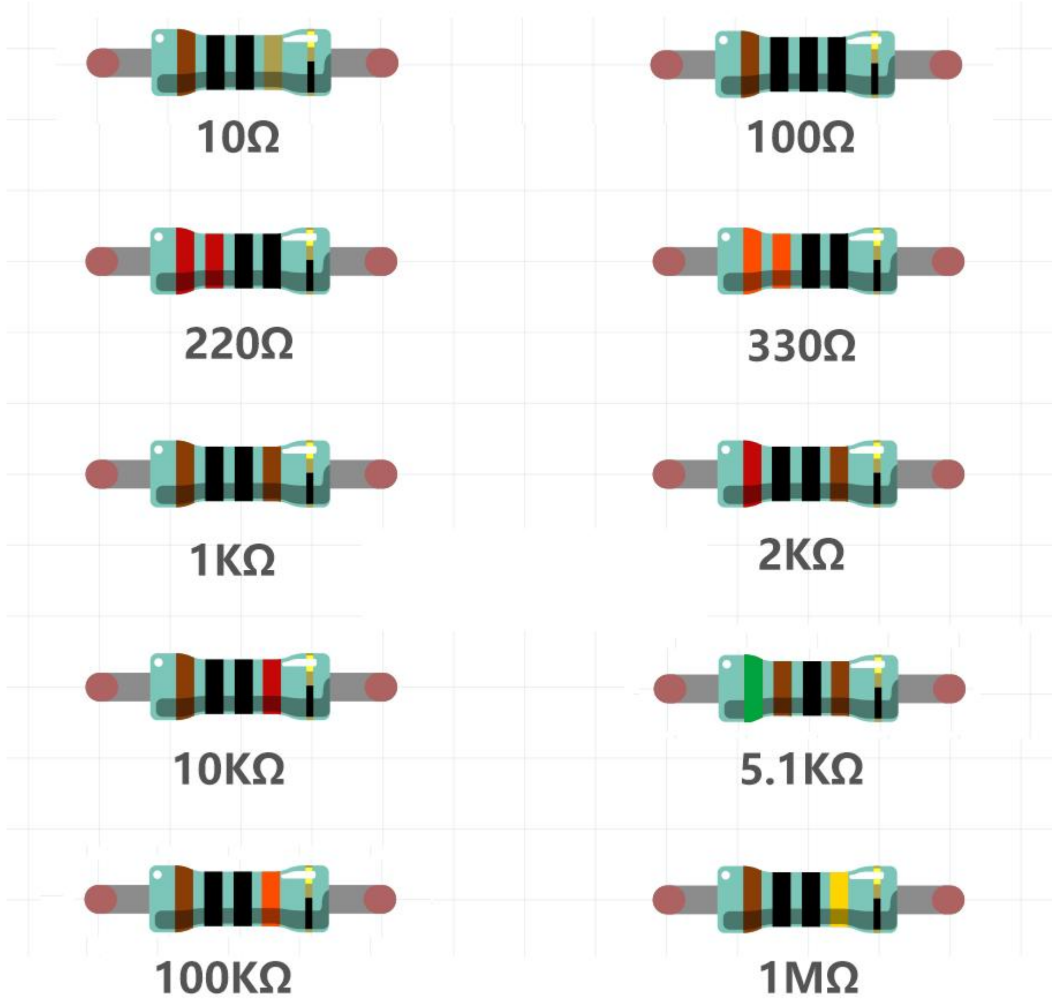


1 x Breadboard Power Module

The breadboard power module is a handy accessory for prototyping, providing a stable 3.3V or 5V power supply from a DC adapter or USB. It fits standard breadboards, plugs into power rails, and includes an on/off switch and voltage regulators for consistent output, making it essential for electronics projects.

120 x Resistors (10 of each, 30 of 220Ω resistor)

A resistor is a component that obstructs the flow of electric power, thereby altering the voltage and current within a circuit. The value of a resistor is measured in ohms, symbolized by the Greek letter omega (Ω). The colored stripes on a resistor indicate its resistance value and tolerance.

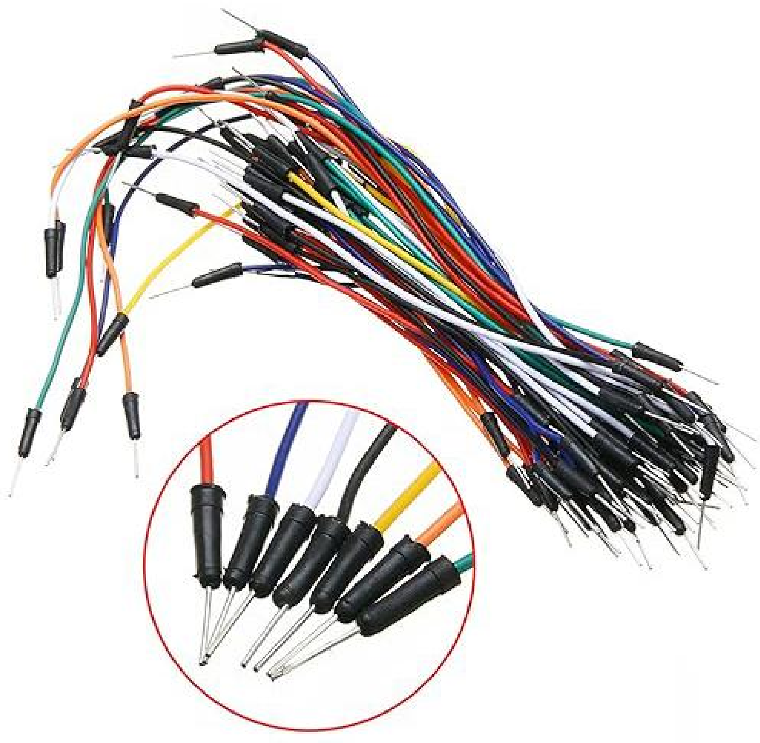


2 x 9V Battery

This is a non-rechargeable alkaline 9V battery. You need to install it in the multimeter, or use battery cable to power the Arduino Uno R3 or the breadboard power module.

65 x Jumper Wires

Connect components on the breadboard to each other and to the Arduino board.



20 x Male-to-female DuPont Wires

Male-to-female DuPont wires are specifically designed for connecting modules with male pin headers, like ultrasonic module, to breadboard. These wires are essential for interfacing different components in electronic projects, where breadboard-compatible male-to-female connections are needed.

1 x USB Cable

Connects the Arduino board to a computer. Allows you to write, compile, and transfer programs to the Arduino board. Also powers the board.

1 x Battery Cable

This cable connects a 9V battery to the DC input of a breadboard power module or an Arduino Uno R3. It provides a convenient and portable power source for your electronic projects.

Displays

25 x LEDs (5 of each color)

This colorful LED selection includes five colors: red, green, blue, yellow, and white, meeting various lighting and signaling needs. Suitable for applications ranging from simple status indicators to complex decorative lighting projects, these LEDs offer a rich color choice to enhance the visual appeal of any electronic project.

1 x RGB LED

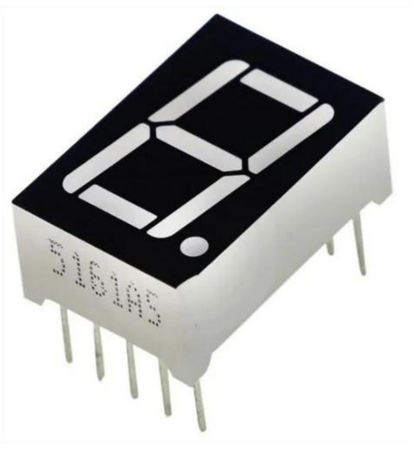
Combines red, green, and blue LEDs in one casing. It can display various colors by adjusting the input voltage, creating millions of colors.

1 x 74HC595 Chip

The 74HC594 is a shift register that is used to expand the input/output ports of digital circuits by converting serial input into parallel output, thus reducing the number of connection pins needed. This chip is suitable for controlling a large number of output devices, such as 7-segment Display, without occupying too many microcontroller pins.

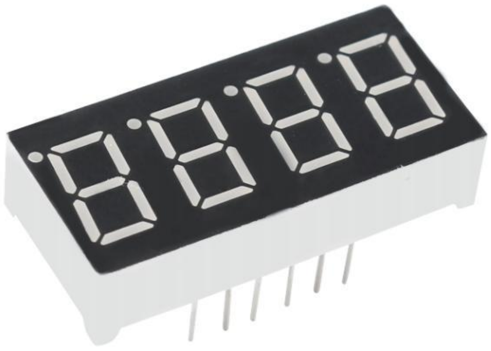
1 x 7-segment Display (Common Cathode)

A 7-segment display is an 8-shaped component which packages 7 LEDs. Each LED is called a segment - when energized, one segment forms part of a numeral to be displayed.



1 x 4-digit 7-segment Display (Common Cathode)

A 4-digit display combines four 7-segment displays, each representing a single digit. To reduce the number of pins needed, the segments of each display are multiplexed, meaning each segment pin is connected to all corresponding segment pins of the other displays.



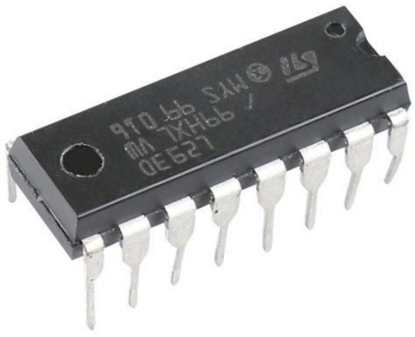
1 x I2C LCD1602

The I2C LCD1602 is a 16x2 character display module that uses the I2C communication protocol. This module is perfect for displaying text, such as sensor data or status messages, in your projects.

Actuators

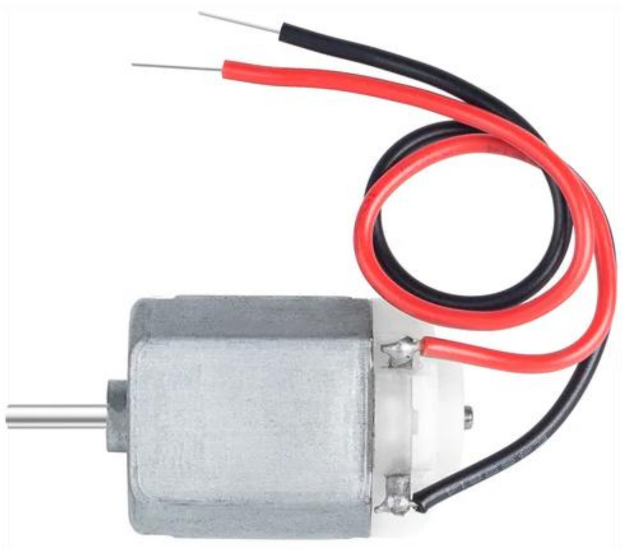
1 x L293D Chip

The L293D is a dual H-bridge motor driver IC that allows you to control the speed and direction of two DC motors simultaneously. It is ideal for robotics and automation projects, providing reliable and efficient motor control.



1 x Motor

The 3V motor is a compact and efficient DC motor designed for low-voltage applications. It is ideal for small electronics projects, toys, and hobbyist robotics, providing reliable performance with low power consumption.



1 x Pump

This is the DC 2.5-6V mini submersible water pump, ideal for small-scale projects like tabletop fountains, aquariums, and hydroponic systems. This pump employs centrifugal mechanics, using an electric motor to convert rotational energy into fluid dynamic energy, efficiently moving water through its system.

1 x Servo

A servo is a precise and versatile motor used for accurate control of angular or linear position, velocity, and acceleration. Commonly used in robotics, automation, and remote control systems, it ensures reliable and smooth movement for various applications.

1 x Tube

This is a 20cm long, 6mm diameter clear tube used to direct water from the outlet of a water pump.

1 x Stepper Motor

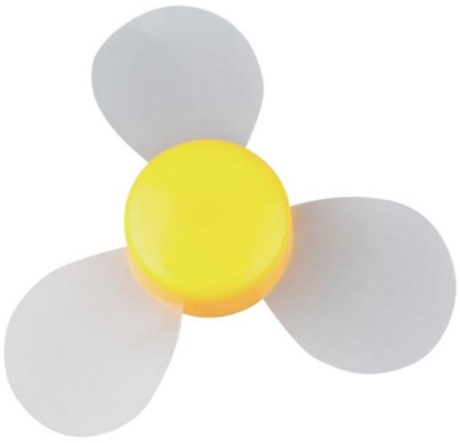
The 28BYJ-48 is a 5-wire unipolar stepper motor that operates at 5V. It is ideal for applications requiring precise control of rotation, such as robotics, 3D printers, and automation projects.

1 x ULN2003 Module

The ULN2003 module is a high-voltage, high-current Darlington transistor array used to drive stepper motors, relays, and other inductive loads. It features seven open-collector Darlington pairs, making it ideal for interfacing with TTL and CMOS logic levels in various control applications.

1 x 3-leaf Fan Blade

The soft 3-leaf fan blade is a flexible and safe fan accessory designed for use with 3V motors. Made from soft, durable materials, it minimizes the risk of injury.



Sounds

1 x Active Buzzer & 1 x Passive Buzzer

A buzzer, available in active and passive types, is an audio signaling device that emits sound when electric current is applied. It is commonly used in alarms, timers, and notification systems.

Sensors

1 x Photoresistor

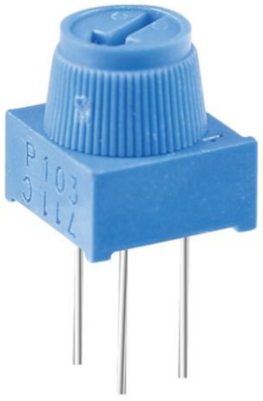
A photoresistor is a light-sensitive component that changes its resistance based on the intensity of light it is exposed to, ideal for creating light-activated controls and sensors in electronic projects.

1 x NTC Thermistor

A thermistor is a resistor sensitive to temperature changes. NTC thermistors decrease resistance as temperature rises, while PTC thermistors increase resistance with temperature.

1 x Potentiometer

A potentiometer is a variable resistor with three pins. Two pins connect to the ends of a resistor, while the middle pin attaches to a movable wiper, dividing the resistor into two parts. Potentiometers, often used to adjust voltage in circuits, are like the volume knobs on radios.

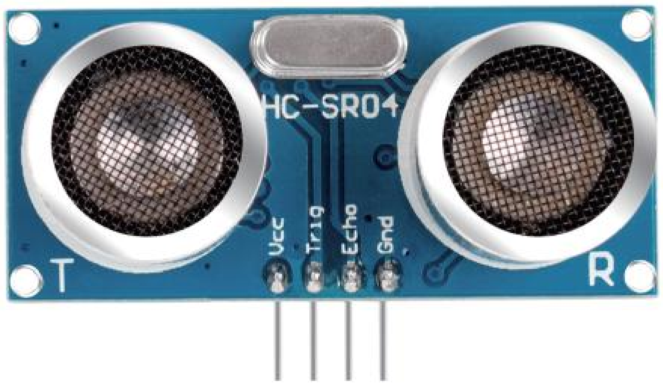


10 x Small Buttons

A small push-button is used to provide a physical response when pressed, commonly used in electronic devices to initiate actions or input commands.

1 x Ultrasonic Module

This is an ultrasonic module that uses ultrasonic waves to measure distances, accurately detecting and measuring the position and distance of objects. Widely used in robotics, obstacle avoidance systems, and automatic control fields, it is a key component for environmental perception and spatial navigation.

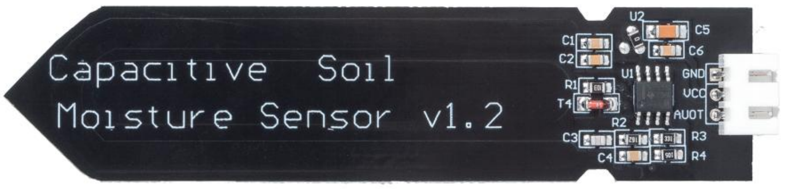


1 x Joystick Module

A joystick module, also known as a joystick sensor, is an input device that measures the movement of a knob in two directions, horizontal (X-axis) and vertical (Y-axis).

1 x Soil Moisture Module

A capacitive sensor for detecting soil moisture, corrosion-resistant, and operates at 3.3V to 5.5V. Outputs moisture value, with wetter soil giving a smaller analog value.



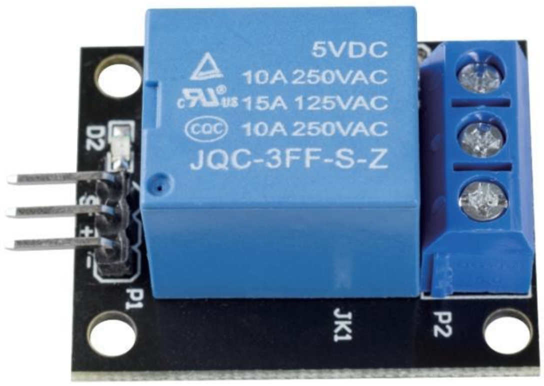
1 x RC522-RFID Module with a Tag and a White Card

The RC522 RFID reader module, operating at a frequency of 13.56MHz, is designed to communicate with RFID tags adhering to the ISO 14443A standard. This compact and versatile device is ideal for applications in access control, inventory tracking, and contactless payment systems due to its ability to interface with microcontrollers via a 4-pin SPI connection, supporting data rates up to 10 Mbps.

Others

1 x Relay Module

A relay module enables microcontrollers to control high-voltage devices by providing an electrically isolated switch. It is ideal for applications requiring the control of AC or high-current loads from low-voltage digital signals.



1 x IR Receiver

The SL838 infrared receiver is a component that receives infrared signals and can independently receive infrared rays and output signals compatible with TTL level. It is similar in size to a normal plastic-packaged transistor and is suitable for all kinds of infrared remote control and infrared transmission.

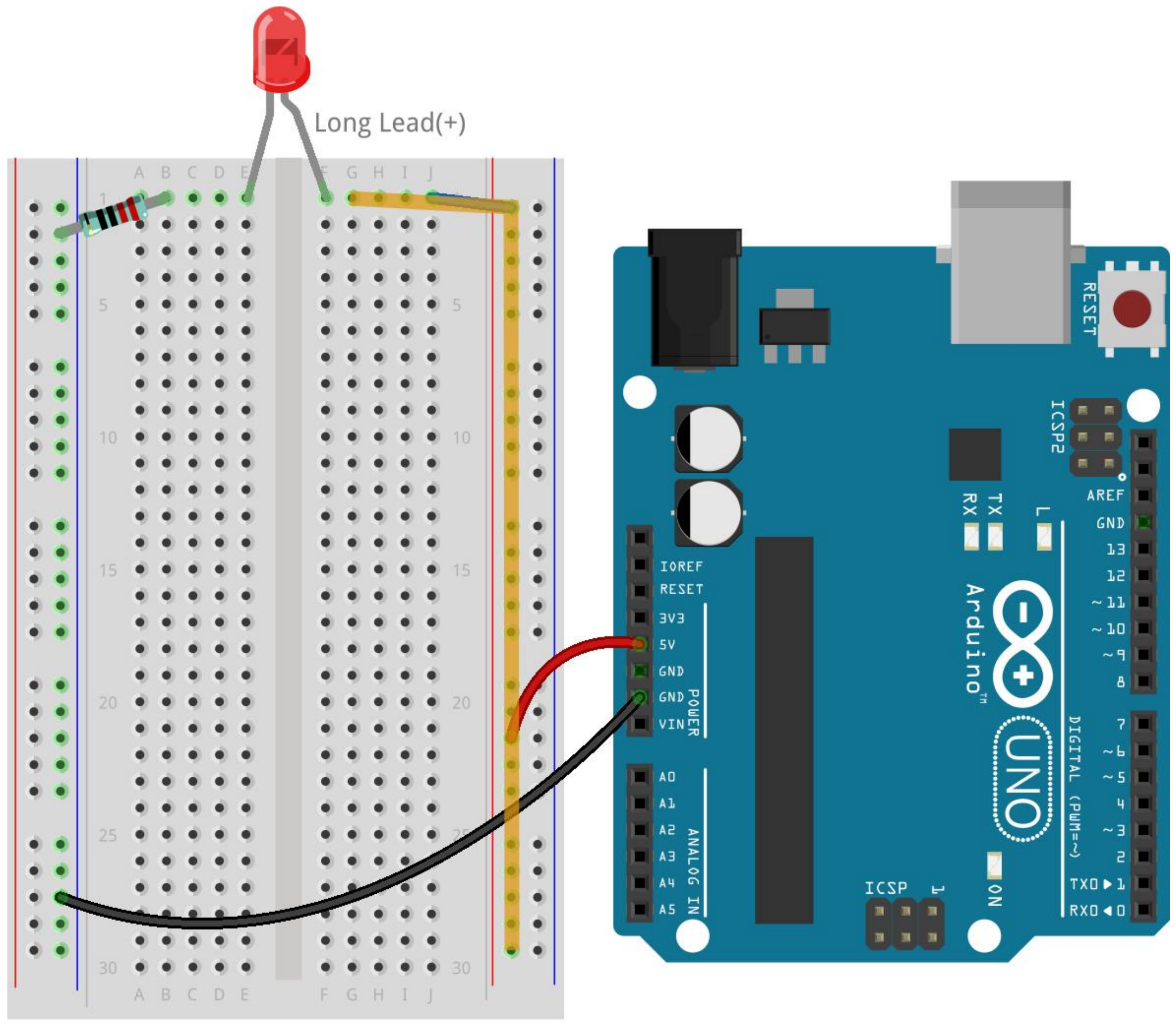
1 x Remote Control

This 21-key remote, compact at 85x39x6mm, has an 8-meter range and is powered by a 3V lithium battery. With a 38KHz infrared frequency and durable PET surface, it ensures over 20,000 uses, making it perfect for various devices.

Lesson 2: Your First Circuit

After completing the lesson, answer the following questions

1. Remove the red wire from the breadboard and experiment by placing it in different holes on the breadboard. Observe any changes in the LED. Sketch the hole positions that allow the LED to light up.



2. What happens if you reverse the pins of the LED? Will it light up? Why or why not?

The LED does not light up because it has unidirectional conductivity; the current must flow from the anode to the cathode in order to work.

SunFounder

Lesson 3: Measure with Multimeter

Answer this question after completing "**Know More about Multimeter!**"

Now that you have a detailed understanding of how to use a multimeter, consider which multimeter setting you would use to measure the following electrical values?

Measurement Object	Multimeter Setting
9V volts DC	*20V*
1K ohms	*2kΩ*
40 milliamps	*200mA*
110 volts AC	*200V~*

Fill out this table during "**Measuring with a Multimeter**"

Type	Units	Measurement Results	Notes
Voltage	Volts	*≈5.13 volts*	
Current	Milliamps	*≈13.54 milliamps*	
Resistance	Ohms	*≈378.88 ohms*	

Lesson4 : Ohm's Law

Fill out the following table during "**Exploring Ohm's Law with Practical Experiments**"

1. Substitute the 220-ohm resistor with other resistors of different values as listed below. Record the LED's brightness changes with each substitution to observe how resistance affects the current and, consequently, the light output.

Resistor	Observations
100Ω	*Brighter*
1KΩ	*Bright*
10KΩ	*Dimmer*
1MΩ	*Nearly Off*

You will notice that only with the 100Ω resistor is the LED brighter than with the previous 220Ω resistor. With higher resistances, the brightness of the LED diminishes until it completely turns off at 1MΩ. Why is this the case?

According to Ohm's Law (I = V/R), as resistance increases while the voltage is held constant, the current through the LED decreases, thus dimming the LED. At 1MΩ, the current is too small to light up the LED.

2. After observing the effects of changing resistance, maintain the resistor at 220 ohms and change the circuit's voltage supply from 5V to 3.3V. Record any changes in the LED's brightness.

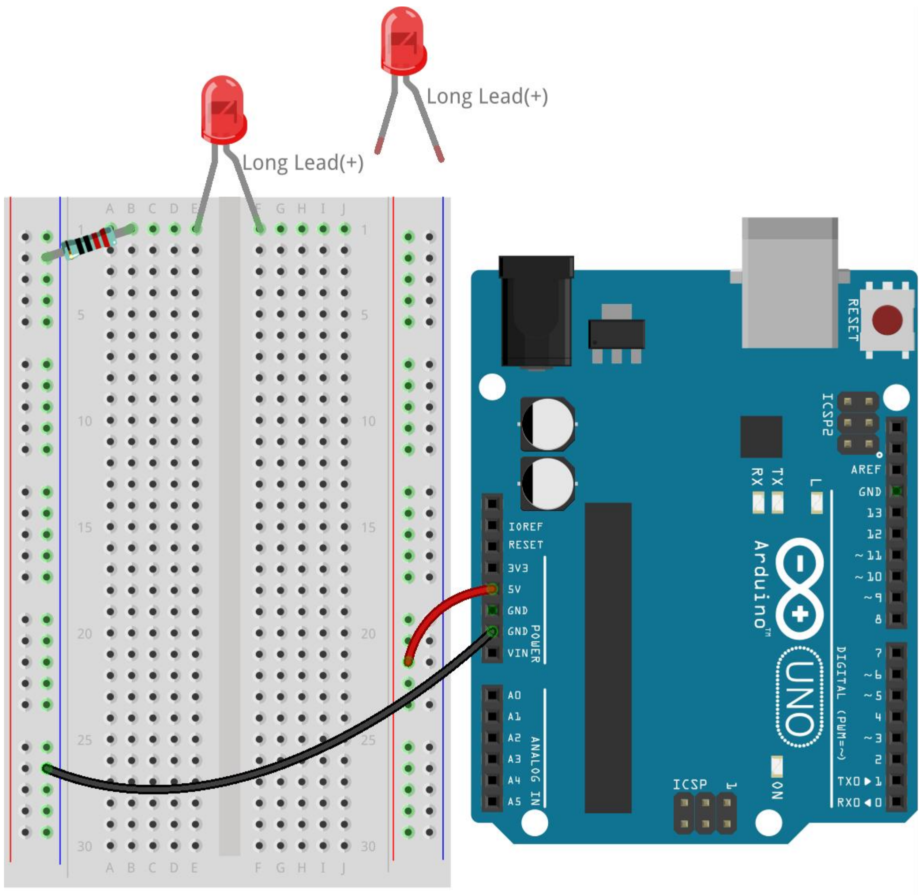
You will find that the LED is slightly dimmer at 3.3V than at 5V. Why is this?

With Ohm's Law, knowing the resistance and the new voltage, the current should be I = V/R. With a decrease in voltage while resistance stays the same, the current decreases, dimming the LED.

Lesson5 : Series Circuit vs. Parallel Circuit

Complete the following questions during "**Diving into Series Circuits**"

1. What happens if you remove one LED? Why does this occur?



In a series circuit, if you remove one LED, the other LED will not light up. This is because in a series circuit, the current must flow through every component in the path. Removing one LED breaks the circuit, preventing current from flowing through the remaining LED.

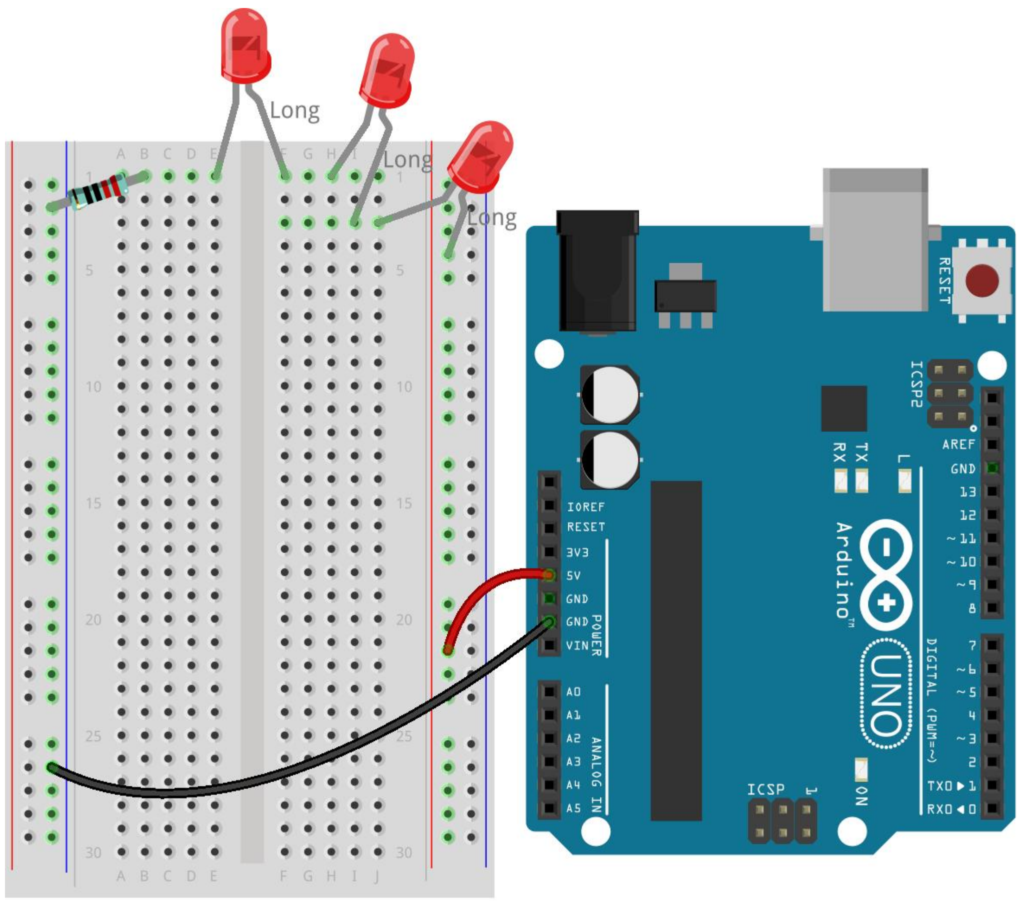
2. Measure the voltage of each component in the series circuit.

Circuit	Resistor Voltage	LED1 Voltage	LED2 Voltage	Total Voltage
2 LEDs	*≈1.13 volts*	*≈1.92 volts*	*≈1.92 volts*	*≈4.97 volts*

3. Measure the current of each component in the series circuit.

Circuit	LED1 Current	LED2 Current
2 LEDs	*≈4.43 milliamps*	*≈4.43 milliamps*

4. If another LED is added to this circuit, resulting in three LEDs, how does the brightness of the LEDs change? why? How do the voltages across the three LEDs change?



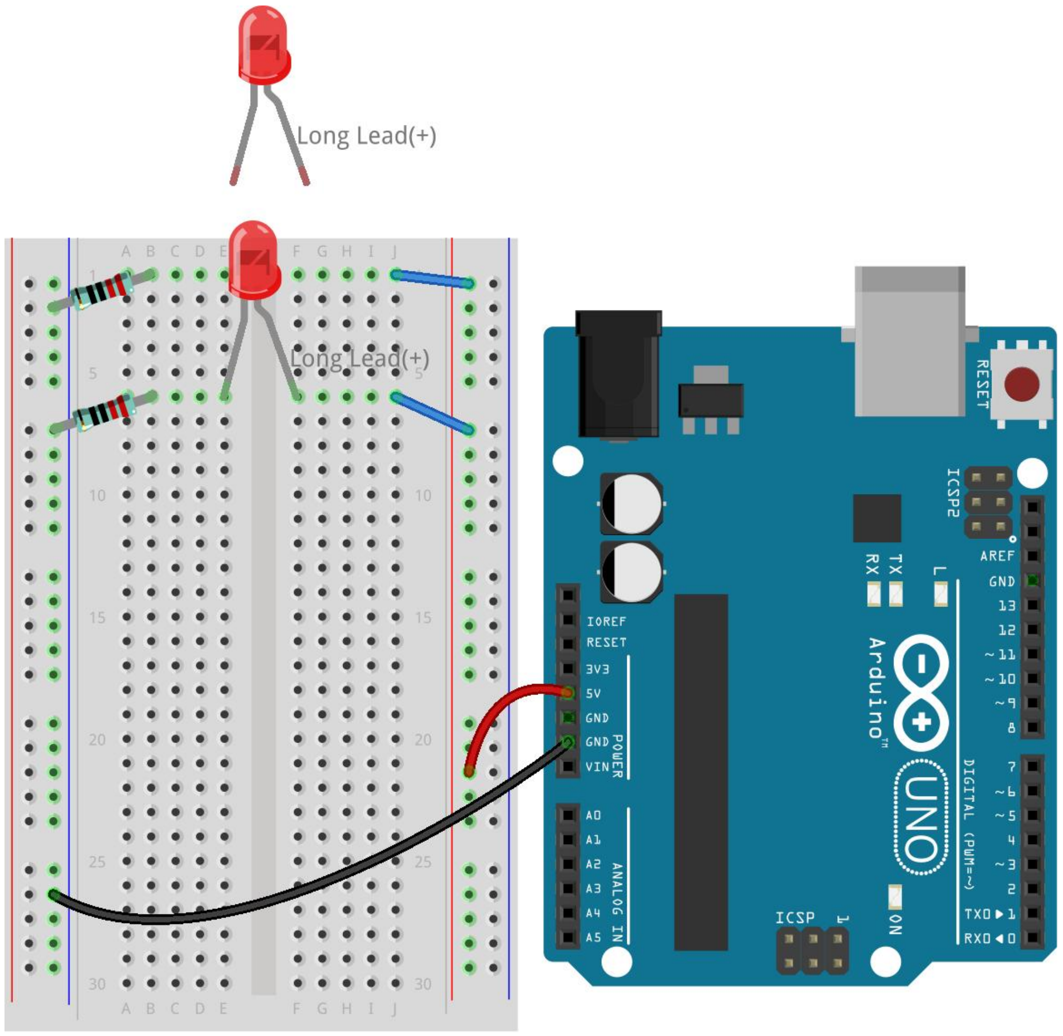
Adding another LED to a series circuit with already two LEDs will generally result in a decrease in the brightness of each LED. This happens because the total voltage of the power source is divided among more components, resulting in a lower voltage drop across each LED than when there were only two. Consequently, less current flows through each LED, reducing their brightness.

As for the voltages across the three LEDs, each LED will now have a smaller portion of the total circuit voltage across it. If the power source's voltage remains the same, this voltage is divided by three, assuming all LEDs have similar electrical characteristics. Therefore, the voltage across each LED in the circuit will be approximately one third of the total voltage provided by the power source.

SunFounder

Complete the following questions during "**Diving into Parallel Circuits**"

1. In this parallel circuit, what happens if one LED is removed? Why does this occur?



In a parallel circuit, if one LED is removed, the other LEDs in the circuit will continue to light up. This occurs because each LED in a parallel circuit has its own independent path to the power source. Removing one LED does not interrupt the current flow to the other LEDs, so they remain unaffected and continue to operate as normal. This setup allows each component in a parallel circuit to operate independently of the others.

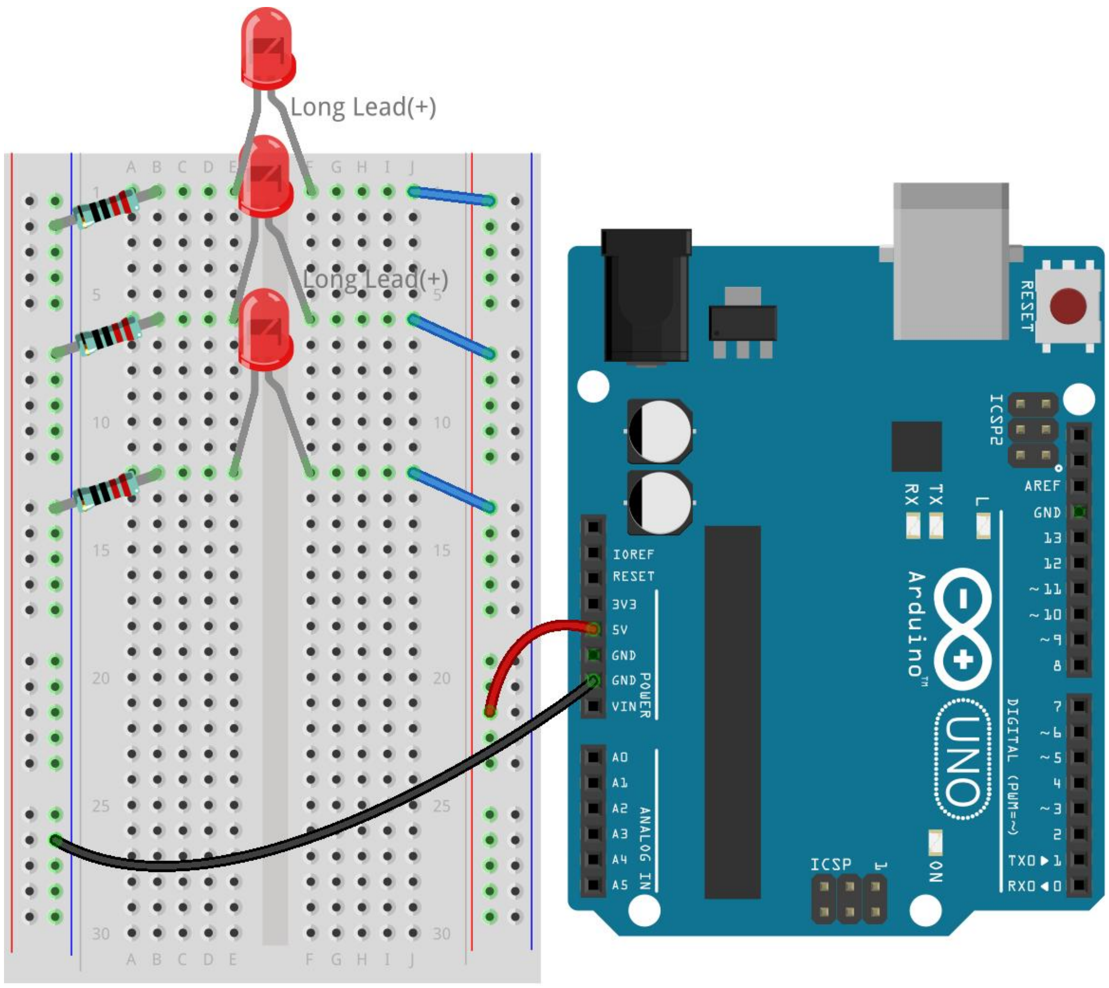
2. Record the measured voltage in the table.

Circuit	Path1 Voltage	Path2 Voltage
2 LEDs	*≈5.00 volts*	*≈5.00 volts*

3. Record the measured current in the table.

Circuit	LED1 Current	LED2 Current	Total Current
2 LEDs	*≈12.6 milliamps*	*≈12.6 milliamps*	*≈25.3 milliamps*

4. If another LED is added to this circuit, what happens to the brightness of the LEDs? Why? Record your answer in your handbook.



When another LED is added to a parallel circuit, the brightness of the existing LEDs typically remains unchanged. This is because each LED in a parallel circuit has its own direct path to the power source, so the voltage across each LED remains constant regardless of how many LEDs are added. Each LED gets the full voltage it requires to operate at its intended brightness. Therefore, adding more LEDs does not affect the brightness of the already present LEDs, provided the power supply can sustain the total current demand of the circuit. Make sure to record this in your handbook for future reference.

Lesson6 : Blink LED

Complete the following table during the "**Bringing LEDs to Life**"

1. Record the measured voltage in the table for Pin 3.

State	Pin 3 Voltage
HIGH	*≈4.95 volts*
LOW	*0.00 volts*

After completing the lesson, answer the following question

1. Upload the above code, and you'll find the LED repeatedly blinking at a 3-second interval. If you just want it to turn on and off once, what should you do?

You can move the commands that turn the LED on and off from the loop() function to the setup() function. The setup() function runs only once when the program starts, so this change will make the LED light up and turn off a single time. Here's how you can adjust your code:

```
void setup() {
  // Setup code here, to run once:
  pinMode(3, OUTPUT);  // set pin 3 as output

  digitalWrite(3, HIGH);  // Light up the LED on pin 3
  delay(3000);            // Wait for 3 seconds
  digitalWrite(3, LOW);   // Switch off the LED on pin 3
}

void loop() {
  // Main code here, to run repeatedly:
}
```

Lesson7 : Let's Make Traffic Lights!

Complete the following question during "**Writing Pseudo-code for a Traffic Light**"

Think about what needs to happen for your circuit to act like a traffic light. In the space provided in your log, write down the pseudo-code describing how your traffic light will function. Use plain English.

To simulate a traffic light using an Arduino, you would need a setup with three LEDs (red, yellow, and green) and a sequence that controls the lighting in a way that mimics real-world traffic lights. Here's a simple pseudo-code outline that you can write down in your log to describe how this traffic light circuit might function:

```
Setup:
    Define pins for the red, yellow, and green LEDs.
    Set all these pins as outputs.
Main Loop:
    Turn on the red LED for 5 seconds.
    Turn off the red LED.
    Turn on the yellow LED for 2 seconds.
    Turn off the yellow LED.
    Turn on the green LED for 5 seconds.
    Turn off the green LED.
    Repeat the cycle.
```

After completing the lesson, answer the following question

Take a look at the intersections around your home. How many traffic lights are there usually? How do they coordinate with each other?

In urban areas, intersections often have traffic lights to manage the flow of vehicles and pedestrians efficiently. The number of traffic lights at an intersection can vary widely depending on its size and complexity. A simple four-way intersection typically has at least four traffic lights, one facing each direction of traffic. More complex intersections may have additional lights for turn lanes, pedestrian crossings, and other traffic management needs.

Lesson8 : Traffic Light with Pedestrian Button

After completing "**Building the Circuit**", answer the following question

1. Your traffic light is a mix of series and parallel circuits. Discuss which parts of your circuit are in series and why. Then, explain which parts are in parallel and why.

In the circuit, the button and its 10K pull-down resistor are connected in series. This setup ensures that when the button is pressed, it properly changes the state of pin 8 by connecting it directly to ground when not pressed, preventing floating inputs.

The three LEDs connected to pins 3, 4, and 5 are in parallel with each other. Each LED operates independently because they are connected to separate control pins and share a common power supply. This setup allows each LED to function without affecting the others, which is crucial for a traffic light system.

Fill out this table during "**Code Creation**"

1. Fill in the table with the measured voltage at pin 8 when the Button is pressed and not pressed. Then fill in the corresponding high and low level states.

Button State	Pin 8 Voltage	Pin 8 State
Release	*0.00 volts*	*LOW*
Press	*≈4.97 volts*	*HIGH*

Complete the following question upon completion of this lesson

1. During testing, you may notice that the green LED only blinks while the pedestrian button is kept pressed, but pedestrians can't cross the road while continuously pressing the button. How can you modify the code to ensure that once the pedestrian button is pressed, the green LED lights up long enough for a safe crossing without requiring continuous pressing? Please write down the pseudo-code solution in your handbook.

To ensure the green LED lights up for pedestrians without requiring the button to be continuously pressed, and to continue the normal traffic light cycle afterwards, you can adjust your pseudocode to check for a button press and then change the state of operation based on that press. Here's an optimized and clearer version of the pseudocode that reflects these changes:

```
Setup:
    Define pins for red, yellow, and green LEDs as output
    Define the button pin as input
Main Loop:
    Check if the button is pressed
    If button is pressed:
        Turn off all LEDs
        Turn on green LED for pedestrians
        Delay 10 seconds
    Else:
        Execute normal traffic light cycle:
            Turn on green LED (for vehicles), turn off other LEDs
            Delay 10 seconds
            Turn on yellow LED, turn off other LEDs
            Delay 3 seconds
            Turn on red LED, turn off other LEDs
            Delay 10 seconds
```

Lesson9 : Dimmable Desk Lamp

Fill out this table during "**Build the Circuit**"

1. Rotate the potentiometer clockwise from position 1 to 3 and measure the resistance at each point, and record the results in the table.

Measurement Point	Resistance (kilohm)
1	*1.52*
2	*5.48*
3	*9.01*

2. How do you think the voltage at A0 would change when the potentiometer is turned clockwise and counterclockwise?

You can think of the potentiometer as consisting of two resistors connected in series within the circuit. According to the measurement of resistances, the resistance between A0 and GND increases as the potentiometer is turned clockwise. Since the current remains constant in a series circuit, according to Ohm's Law (voltage = current × resistance), an increase in resistance leads to an increase in the voltage at A0. Therefore, turning the potentiometer clockwise increases the voltage at A0, while turning it counterclockwise decreases it, as the resistance decreases.

Complete the following question upon completion of this lesson

1. If you connect the LED to a different pin, such as pin 8, and rotate the potentiometer, will the brightness of the LED still change? Why or why not?

If you connect the LED to pin 8 on an Arduino UNO and rotate the potentiometer, the brightness of the LED will not change. This is because pin 8 does not support PWM (Pulse Width Modulation), which is necessary for adjusting brightness levels using the `analogWrite()` function. On an Arduino UNO, the pins that support PWM and can thus be used to control the brightness of an LED through `analogWrite()` are pins 3, 5, 6, 9, 10, and 11.

Lesson10: Morse Code

Answer the following question during "**Building the Circuit**"

1. What will happen if you connect the cathode of an active buzzer directly to GND and the anode to 5V? Why?

If you connect the cathode of an active buzzer directly to GND and the anode to 5V, the buzzer will emit a continuous sound. This happens because the internal oscillator in the buzzer is activated by the 5V power, causing it to generate sound until the circuit is disconnected.

After completing the lesson, answer the following question

1. Using the Morse code table provided, write a code to send the message "Hello".

In Morse code, "Hello" would be encoded as follows based on the characters:

- H: ....
- E: .
- L: .-..
- L: .-..
- O: ---

Putting it together, the Morse code for "Hello" is:

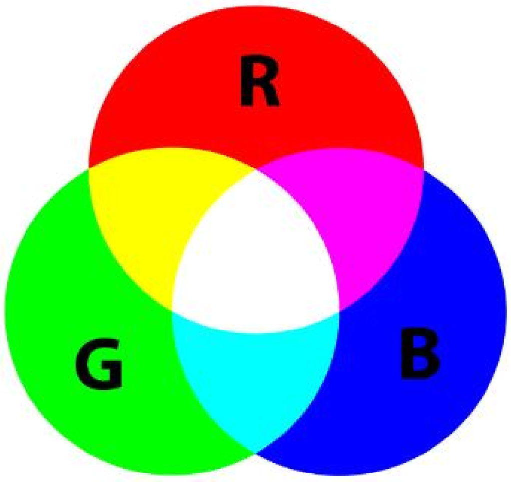
```
.... . .-.. .-.. ---
```

In practical communication, there's usually a longer pause between words to differentiate them clearly, but since "Hello" is a single word, the code is continuous with spaces only separating individual letters.

Lesson11: The Colors of the Rainbow

Fill out this table during "**Code Creation - Lighting Up an RGB LED**"

1. If you want other colors, what should you do? Refer to the diagram below and fill in your ideas in your handbook.



Color	Red Pin	Green Pin	Blue Pin
Red	*HIGH*	*LOW*	*LOW*
Green	*LOW*	*HIGH*	*LOW*
Blue	*LOW*	*LOW*	*HIGH*
Yellow	*HIGH*	*HIGH*	*LOW*
Pink	*HIGH*	*LOW*	*HIGH*
Cyan	*LOW*	*HIGH*	*HIGH*
White	*HIGH*	*HIGH*	*HIGH*

SunFounder

Fill out this table during "Code Creation - Displaying Colors"

1. Now you can adjust the values of pins 9, 10, and 11 separately, and record the observed colors in your handbook.

Red Pin	Green Pin	Blue Pin	Color
0	128	128	*Dark blue*
128	0	255	*Purple*
128	128	255	*Light blue*
255	128	0	*Orange*

Fill out this table during "**Code Creation - Parameterized Functions**"

1. Choose some of your favorite colors and fill in the table with their RGB values.

Color	Red	Green	Blue

Lesson12: Siren Sound

Answer the following question during "**Build the Circuit**"

1. What will happen if you connect the cathode of an passive buzzer directly to GND and the anode to 5V? Why?

If you connect the cathode of a passive buzzer directly to GND and the anode to 5V, unlike an active buzzer, the passive buzzer will not make any sound by itself because it does not have a built-in oscillator.

A passive buzzer requires an external signal to generate sound. Typically, you need to drive it with a square wave (oscillating voltage) at the desired frequency to create audible sounds.

Answer the following questions during "**Code Creation - Make the Passive Buzzer Sound**"

1. If you switch the code and circuit pins to 7 or 8, which are not PWM pins, will the buzzer still make a sound? You can test and then write your answer in the handbook.

Even though pin 8 is not a PWM pin, the `tone()` function can still generate a precise square wave on it, effectively driving a passive buzzer to produce sound. This flexibility allows you to use any digital pin for sound output without being limited to PWM-capable pins. When you call the tone(pin, frequency) function, Arduino configures a timer to toggle the pin state (from HIGH to LOW and back to HIGH) at the specified frequency, creating a square wave. This square wave drives the passive buzzer, making it emit sound at the frequency of the generated wave.

2. To explore how frequency and duration in tone(pin, frequency, duration) affect the sound of the buzzer, please modify the code under two conditions and fill in the observed phenomena in your handbook:

- Keeping frequency at 1000, gradually increase duration, from 100, 500, to 1000. How does the sound of the buzzer change, and why?
  - 100 ms duration: The sound is a short beep.
  - 500 ms duration: The sound is a longer beep, clearly audible and lasting half a second.
  - 1000 ms duration: The sound is even longer, lasting for a full second.

  As you increase the duration, the sound emitted by the buzzer lasts longer. The pitch or frequency of the sound remains constant (as it is set at 1000 Hz), which means the tone's "note" doesn't change, but the length of time you hear it increases. This is useful for signaling different durations of alerts where the urgency or type of alert can be distinguished by the length of the tone.

- Keeping duration at 100, gradually increase frequency, from 1000, 2000, to 5000. How does the sound of the buzzer change, and why?
  - 1000 Hz frequency: The sound is a medium-pitched beep.
  - 2000 Hz frequency: The sound has a higher pitch compared to 1000 Hz.
  - 5000 Hz frequency: The sound is much higher-pitched, likely perceived as sharper and possibly uncomfortable at close range.

    Increasing the frequency while keeping the duration constant results in a change in the pitch of the sound. Higher frequencies produce higher-pitched sounds. This principle is useful for distinguishing between different types of notifications or signals based on their urgency or importance, with higher pitches often used for more urgent alerts.

Lesson 13: Joystick LED Navigator

Answer the following question during "Code Creation- Read from Joystick Module"

1. The X and Y axes of the joystick module return analog values, while the SW pin returns a digital value. In previous steps, we've already seen these values on the Serial Monitor.

Please summarize the differences between digital and analog values in Arduino programming.

In Arduino programming:

- **Analog Values:** Continuous values ranging from 0 to 1023, corresponding to 0V to 5V, captured by the Arduino's analog-to-digital converter (ADC). These values provide detailed information about signal magnitude, as seen with the joystick's X and Y axes.
- **Digital Values:** Discrete values of 0 or 1, representing low (0V) or high (5V) states. The SW pin on your joystick uses a digital signal to indicate whether it's pressed or not, ideal for binary on/off decisions.

Analog inputs offer detailed signal intensity, while digital inputs are used for simple presence or absence detection.

Answer the following question during "Code Creation - Controlling LEDs Based on Joystick Movements"

1. In the last code, we controlled the corresponding LEDs based on the X and Y values of the joystick. How would you modify the code to adjust the brightness of these LEDs while they are lit?

To control the brightness of LEDs based on the X and Y values of the joystick, you should utilize PWM (Pulse Width Modulation) via the `analogWrite()` function instead of `digitalWrite()`. This approach allows you to set varying levels of LED brightness correlating with the joystick's position, rather than merely switching the LEDs on or off.

Here's an improved code snippet to demonstrate this concept:

Here's how you could modify the code:

```
void loop() {
  // Read the joystick values
  int xValue = analogRead(xPin);
  int yValue = analogRead(yPin);
  int swValue = digitalRead(swPin);

  // First, turn off all LEDs
  analogWrite(ledLeft, 0);
  analogWrite(ledRight, 0);
  analogWrite(ledUp, 0);
  analogWrite(ledDown, 0);

  // Check joystick positions and set LEDs accordingly
  if (yValue < 412) {
    // Joystick up
    analogWrite(ledUp, yValue / 4);
  } else if (yValue > 612) {
    // Joystick down
    analogWrite(ledDown, yValue / 4);
  } else if (xValue < 412) {
    // Joystick left
    analogWrite(ledLeft, xValue / 4);
  } else if (xValue > 612) {
    // Joystick right
    analogWrite(ledRight, xValue / 4);
  }
  // Add a small delay to stabilize readings
  delay(100);
}
```

2. What is the difference in behavior of the LED connected to pin 8 compared to those on other pins, and why?

The LED connected to pin 8 on the Arduino Uno R3 only toggles between on and off states without any variation in brightness.

This is because pin 8 does not support PWM (Pulse Width Modulation). On the Arduino Uno R3, PWM is only available on pins 3, 5, 6, 9, 10, and 11. The absence of PWM capability on pin 8 means that the analogWrite() function, which adjusts LED brightness through varying voltage levels, cannot be used on this pin. As a result, you can only use digitalWrite() to turn the LED fully on or off with no intermediate states.

SunFounder

Lesson 14: Play Dinosaur Game

Answer the following question during "2. Prepare the Servo"

1. If the servo is connected to pin 8 or another non-PWM pin on an Arduino, will it still operate correctly? Why or why not?

You can test this first before answering.

When you connect a servo motor to a pin on an Arduino, it's generally recommended to use a PWM (Pulse Width Modulation) pin.

However, the Arduino's Servo library is designed to facilitate servo control beyond just PWM pins by using software emulation to generate the necessary signals.

This software-based approach means that servo control pulses are managed internally by the library, not dependent on the microcontroller's hardware PWM capabilities. As a result, the Servo library enables the use of any digital pin—including non-PWM ones like pin 8—to effectively control a servo.

Answer the following question during "3. Ready the Photoresistor"

1. Read the resistance value under the current ambient light and record it in the table below.

Environment	Resistance (kilohm)
Normal Light	*≈5.48*
Bright Light	*≈0.16*
Darkness	*≈1954*

Lesson15: Cool or Warm Colors

Fill out this table during "**Code Creation**"

1. Open Paint or any color picking tool, find what you consider the warmest and coolest colors, and record their RGB values in your handbook.

Color Type	Red	Green	Blue
Warm Color	*246*	*52*	*8*
Cool Color	*100*	*150*	*255*

After completing the lesson, answer the following question

1. Note that the "lower bounds" of either range may be larger or smaller than the "upper bounds", so the map() function may be used to reverse a range of numbers, for example:

```
y = map(x, 1, 50, 50, 1);
```

The function also handles negative numbers well, so that this example is also valid and works well.

```
y = map(x, 1, 50, 50, -100);
```

For y = map(x, 1, 50, 50, -100);, if x equals 20, what should y be? Refer to the following formula to calculate it.

$$\frac{\text{Value - From Low}}{\text{From High - From Low}} = \frac{\text{Y - To Low}}{\text{To High - To Low}}$$

$$\text{Y} = \frac{\text{Value - From Low}}{\text{From High - From Low}} \times (\text{To High - To Low}) + \text{To Low}$$

For x=20 using the mapping formula y = map(x, 1, 50, 50, -100); the value of y would be approximately −8.16.

Lesson16: Summer Fan

Fill out this table during "**Build the Circuit**"

1. With 1,2EN connected to 5V, use a multimeter to measure the voltage at pin 3 (1Y) when pin 2 (1A) is connected to 5V and when it is connected to GND. Record the measurements in the table.

1,2EN	1A	1Y
5V	5V	≈5.04V
5V	0V	*≈0V*

After completing the lesson, answer the following question

1. How should the code be modified if you want to control the motor's direction as well?

- Modify the `motorRotate()` function to include direction control by using both motor control pins.
- Add a parameter to the `motorRotate()` function to specify direction.
- Adjust the calls to `motorRotate()` in the main loop to include direction control based on button presses.

```
void loop() {
  if (digitalRead(button1) == LOW) {
    motorRotate(0, 0);  // Turn off the motor
  } else if (digitalRead(button2) == LOW) {
    motorRotate(150, -1);  // Motor runs backward at low speed
  } else if (digitalRead(button3) == LOW) {
    motorRotate(200, 1);  // Motor runs forward at medium speed
  } else if (digitalRead(button4) == LOW) {
    motorRotate(250, 1);  // Motor runs forward at high speed
  }
}
void motorRotate(int speed, int direction) {
  if (direction >= 0) {
    analogWrite(motor1A, speed);
    analogWrite(motor2A, 0);
  } else {
    analogWrite(motor1A, 0);
    analogWrite(motor2A, speed);
  }
}
```

Lesson17: Exploring the I2C LCD1602 Display

After completing the lesson, answer the following question

1. If you wish to display 'Let's count' beginning from the fifth column of the first row on the I2C LCD1602, how should you adjust the code, and what visual effect would you observe?

To start displaying "Let's count" from the seventh column of the first row on the I2C LCD1602, you would adjust the `lcd.setCursor()` function by setting the cursor to the sixth column (since column indices start at 0). Here's how you can modify the code and what you might observe:

```
void loop() {
  lcd.setCursor(6, 0);       // Sets cursor to the fifth column (index
6) of the first row.
  lcd.print("Let's count");  // Displays "Let's count".
  lcd.setCursor(0, 1);       // Moves cursor to the first column of
the second line.
  lcd.print("Count: ");      // Displays "Count: ".
  lcd.print(count);          // Prints current count next to "Count:
".
  delay(1000);               // Pauses for one second.
  count++;                   // Increments counter.
  lcd.clear();               // Clears the display for the next
iteration.
}
```

Starting from the seventh column, the text "Let's count" will begin to display further right on the display. The text "Let's count" has 11 characters, so it will extend to the seventeenth column, but since the LCD1602 has only 16 columns, the last character will not be visible on the display. This truncation results in only "Let's coun" being visible, with the last "t" cut off due to the edge of the display.

Thus, careful planning of the display layout is essential to ensure all desired text is properly visible on the LCD screen.

Lesson18: ON/OFF Desk Lamp

Fill out this table during "**Build the Circuit**"

1. Now, use a multimeter to measure the continuity between the COM and NC to validate the working principle of the relay module.

State	NO or NC connected to the COM terminal?
Default	*NC*
S pin High	*NO*

Complete the following questions upon completion of this lesson

1. What would happen if you set digital pin 7 to INPUT only? Why?

```
void setup() {
  pinMode(9, OUTPUT);  // Set pin 9 as output
  pinMode(7, INPUT);   // Set pin 7 as input with an internal pull-up
resistor
  Serial.begin(9600);  // Serial communication setup at 9600 baud
}
```

Setting digital pin 7 to INPUT mode in your Arduino sketch, as opposed to INPUT_PULLUP, can lead to potential instability in the signal read from the pin. When a pin is configured as INPUT only and not connected to either a definitive high or low voltage through external circuitry, it becomes what is known as "floating." A floating pin is not in a stable high or low state; its state can fluctuate based on electrical noise or interference from the environment. This fluctuation can lead to unpredictable readings when you attempt to read the pin's state via digital input functions, resulting in erroneous or inconsistent data being received by the microcontroller.

2. If pin 7 is set only to INPUT, what adjustments would need to be made to the circuit?

If pin 7 on your Arduino is set to INPUT mode and you want to ensure stable and predictable readings, you should add an external pull-up resistor to the circuit. This involves connecting a 10kΩ resistor between pin 7 and the 5V power supply on the Arduino. The pull-up resistor ensures that the input pin is at a high state (logic level 1) when there is no other input signal present.

Lesson19: Smart Trash Can

Answer the following question during "Code Creation - Read the Distance"

1. If you want the distance detected by this device to be more accurate to decimals, how should you modify the code?

To enhance the accuracy of the distance measurements in your Arduino code and include decimal precision, you will need to adjust the calculation in the measureDistance() function. Currently, the distance is calculated using integer math, which truncates any decimal places. To achieve decimal accuracy, you can modify the calculation to use floating-point arithmetic instead of integer arithmetic.

Here's how you can modify the code:

- Change the data type of distance from long to float to allow for decimal values.
- Adjust the formula to ensure it performs floating-point calculations.

Here is the modified portion of the measureDistance() function:

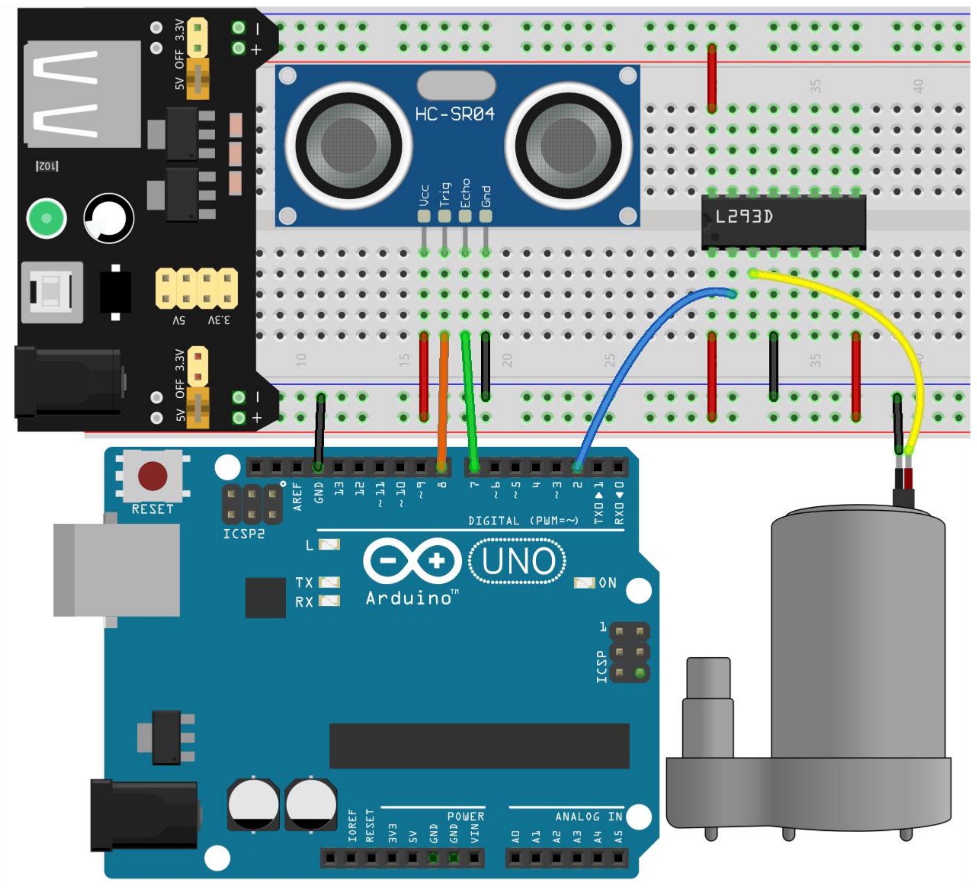
```
float measureDistance() {
  digitalWrite(TRIGGER_PIN, LOW);  // Ensure Trig pin is low before a pulse
  delayMicroseconds(2);
  digitalWrite(TRIGGER_PIN, HIGH);  // Send a high pulse
  delayMicroseconds(10);            // Pulse duration of 10 microseconds
  digitalWrite(TRIGGER_PIN, LOW);   // End the high pulse

  long duration = pulseIn(ECHO_PIN, HIGH);  // Measure the duration of high level on Echo pin
  float distance = duration * 0.034 / 2.0;   // Calculate the distance (in cm) with decimal precision
  return distance;
}
```

Lesson20: Automatic Soap Dispenser

Answer the following question during "Code Creation - Making the Water Pump Work"

1. In this project, you connected the water pump using a specific driver and setup. What do you think would happen if you reversed the connections of the pump? Would the pump work in reverse, stop working, or something else? Try this out and reflect on the outcome.



You will find that if you were to reverse the connections of the pump in your project, the pump would still operate effectively and continue to pump water. It would not work in reverse, stop working, or cause any damage.

Reversing the connections does not change the function of the pump due to its bidirectional capability. The pump will continue to move water from its inlet to its outlet as designed.

Lesson21: Temperature Alarm

Fill out this table during “**Building the Circuit**”

1. Read the resistance value under the different temperature and record it in the table below.

Environment	Resistance (kilohm)
Current temperature	*9.37*
Higher temperature	*6.10*
Lower temperature	*12.49*

After completing the lesson, answer the following question

1. In the code, Kelvin and Celsius temperatures are calculated. If you also want to know the Fahrenheit temperature, what should you do?

This is the standard method for converting Celsius to Fahrenheit and will give you the temperature in Fahrenheit based on the Celsius value you already have from your calculations.

```
F = C * 1.8 +32
```

2. Can you think of other situations or places where a temperature monitoring system like the one we built today could be useful?

Temperature monitoring systems are widely applicable in everyday situations and various environments. Here are a few simplified examples:

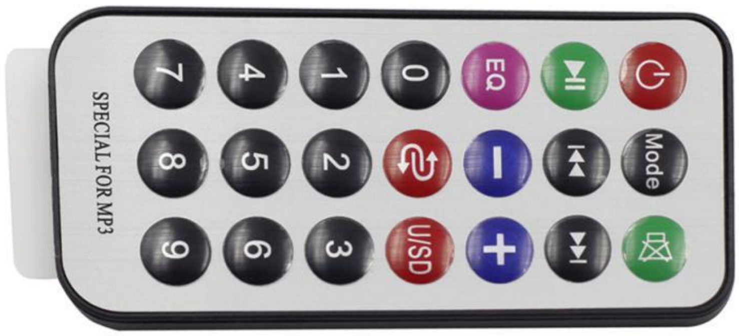
- **Home Comfort**: Automatically adjust your home heating or cooling based on real-time temperature readings to keep your living space comfortable.
- **Gardening**: Monitor greenhouse temperatures to ensure plants are growing in optimal conditions. Add automated systems to adjust temperatures as needed.
- **Food Safety**: Keep track of fridge and freezer temperatures to ensure food remains safe to eat, especially in restaurants or during food transport.

Healthcare: Monitor and log temperatures in storage areas for temperature-sensitive medicines and vaccines to ensure they remain effective.

Lesson22: Remote-Controlled Colorful Light

Fill out this table during "Code Creation - Getting the Key Values"

1. Please carefully press each key on the remote control and record the corresponding key values in the table in your manual.



Key Name	Key Value	Key Name	Key Value
POWER	0x45	0	0x16
MODE	0x46	1	0xC
MUTE	0x47	2	0x18
PLAY/PAUSE	0x44	3	0x5E
BACKWARD	0x40	4	0x8
FORWARD	0x43	5	0x1C
EQ	0x7	6	0x5A
-	0x15	7	0x42
+	0x9	8	0x52
CYCLE	0x19	9	0x4A
U/SD	0xD		

SunFounder

Lesson23: Play "Twinkle, Twinkle, Little Star"

Answer the following question during "Code Creation - Array"

1. You can also pertable operations on the elements in the array, such as changing to Serial.println(melody[i] * 1.3),What data will you get and why?

The number 1.3 is a floating-point number. When an integer from the melody array (which is of type int) is multiplied by 1.3, the result of the operation is automatically promoted to a floating-point number (float).

For each note frequency in this array, multiplying by 1.3 and then printing the result will yield:

```
340.6
340.6
509.6
509.6
572.0
572.0
509.6
...
```

After completing the lesson, answer the following question

1. If you replace the passive buzzer in the circuit with an active buzzer, can you positively play "Twinkle Twinkle Little Star"? Why?

If you replace the passive buzzer with an active buzzer to play "Twinkle Twinkle Little Star," it won't work as intended. Active buzzers can only produce a single tone because they have a built-in oscillator. Therefore, you cannot control the pitch to play the melody accurately; you would only hear a repetitive beep in the rhythm of the song, not the actual notes.

Lesson24: The Pomodoro Timer

Answer the following question during "Coding Creation - millis()"

1. If the delay(100) is changed to delay(1000), what will happen to the program? Why?

```
if (currentMillis - previousMillis >= interval) {
  previousMillis = currentMillis;  // Save the last time the buzzer beeped
  digitalWrite(buzzerPin, HIGH);   // Make a voice
  delay(100);
  digitalWrite(buzzerPin, LOW);  // silence
}
```

In the original code, the buzzer beeps for about 100 milliseconds every 1000 milliseconds (1 second, as set by the interval variable), followed by a silence of 900 milliseconds. After the modification, the buzzer will beep for 1000 milliseconds every 1000 milliseconds, and then almost immediately beep again, as the next interval starts almost right away. Thus, changing the delay from 100 to 1000 milliseconds turns the buzzer from emitting brief beeps to continuous sound, which becomes more annoying and unsuitable for the original intent.

Changing delay(100) to delay(1000) in your code will make the buzzer sound for a full second instead of just a short beep, as it increases the pause time when the buzzer is on. This results in longer buzzer noises and less frequent program cycles, potentially making the program less responsive to other tasks during these intervals.

After completing the lesson, answer the following question

Think about other places in your life where you can 'hear' time. List a few examples and write them in your handbook!

Hearing time is an intriguing concept, and there are several everyday scenarios where we can experience this. Here are a few examples you might consider noting down:

- **Clocks and Watches:** The ticking of analog clocks or the specific beeps from digital watches that signal each passing second or minute.
- **Kitchen Timers:** The ticking and final alarm of a mechanical or digital

kitchen timer counting down cooking or baking time.

- **School Bells:** The ringing of bells in schools that mark the beginning and end of periods or breaks.
- **Public Transport Announcements:** The beeping or chimes that precede announcements at train stations or on buses, marking the imminent departure or arrival.
- **Microwave Oven:** The beeping sound when the timer ends, signaling that the heating process is complete.
- **Fitness Trackers or Sports Watches:** The beeps or alarms that indicate the completion of a set time during workouts or intervals.

Lesson25: Reverse Radar System

After completing the lesson, answer the following question

1. In this project, we used an active buzzer to serve as an alert mechanism, but a passive buzzer could also be used to achieve similar functionality. If you were to replace the active buzzer with a passive buzzer, how should the code be modified?

If you decide to replace the active buzzer with a passive buzzer in your Arduino project, you will need to modify the code that controls the buzzer because passive buzzers require an alternating signal to produce sound, unlike active buzzers that only need a high signal to beep.

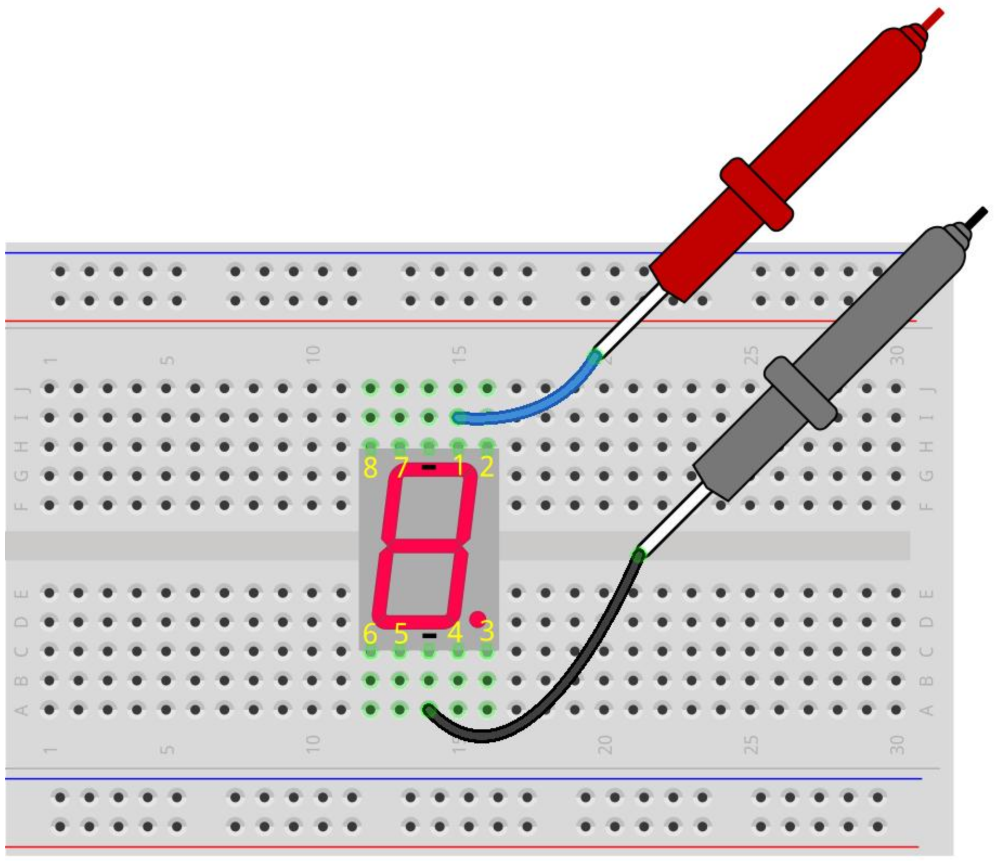
Here's how you can modify the beep() function in your code to work with a passive buzzer:

```
// Function to make passive buzzer beep
void beep() {
  // Generate a tone on the buzzer pin
  tone(BUZZER_PIN, 2000);  // 2000 Hz frequency
  delay(100);             // Beep duration: 100 milliseconds
  noTone(BUZZER_PIN);     // Stop the tone
}
```

Lesson26: Cyber Dice

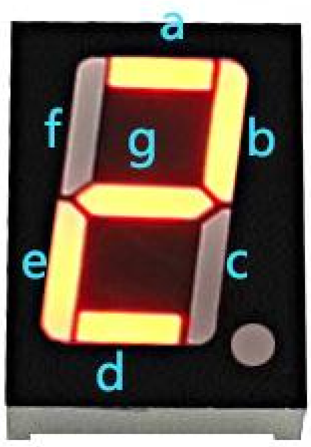
Answer the following questions during "**Learn the 7-Segment Display**"

1. If a segment lights up, refer to this diagram to record the segment's pin number and approximate position in the Handbook's table.



Pin	Segment Number	Position
1	*A*	*The top*
2	*B*	*The top right*
3	*C*	*The bottom right*
4	*D*	*The bottm*
5	*E*	*The bottm left*
6	*F*	*The top left*
7	*G*	*The middle*
8	*decimal point*	*The dot*

2. From the tests above, it is known that the display in the kit is common cathode, which means you only need to connect the common pin to GND and provide a high voltage to the other pins to light up the corresponding segments. If you want the display to show the number 2, which pins should be provided with a high voltage? Why?



For the number 2, the segments a, b, d, e, and g need to be activated (set to high voltage) because these are the segments that form the number 2 on the display. The segments f, c, and dp (decimal point, if present) should remain off (low voltage) as they are not part of the number 2 display.

So, the pins that should be provided with a high voltage are those connected to the a, b, d, e, and g segments to correctly display the number 2.

Lesson27: Flowing Light with 74HC595

Answer the following question during "Code Creation - Lighting Up LEDs"

1. What happens if we change MSBFIRST to LSBFIRST in shiftOut(DS, SHcp, MSBFIRST, B11101110);? Why?

If you change MSBFIRST to LSBFIRST, the order of the bits is reversed, and the byte is shifted out starting from the least significant bit (the rightmost one). If you are using the shift register to control LEDs, changing the bit order will reverse the order in which the LEDs light up. Instead of lighting up in the sequence originally programmed, they will light up in the reverse order.

After completing the lesson, answer the following question

1. If we want to have three LEDs lit at a time and have them appear to "flow," how should the elements of the dataArray[] array be modified?

You would start with the first three LEDs on and then move one LED to the right in each subsequent pattern until the last three LEDs are on. Here's how you could define these patterns in binary:

```
B11100000: LEDs 1, 2, 3 are on; others are off.
B01110000: LEDs 2, 3, 4 are on; others are off.
B00111000: LEDs 3, 4, 5 are on; others are off.
B00011100: LEDs 4, 5, 6 are on; others are off.
B00001110: LEDs 5, 6, 7 are on; others are off.
B00000111: LEDs 6, 7, 8 are on; others are off.
```

```
byte dataArray[] = { B11100000, B01110000, B00111000, B00011100,
B00001110, B00000111 };
```

SunFounder

Lesson 28: Show Number

Fill out this table during "**Binary Numbers for Digits 0 to 9**"

1. Now that we know the binary representations for digits 0 and 2, please fill in the binary numbers for the remaining digits in the table below.

Number	Binary
0	*B00111111*
1	*B00000110*
2	*B01011011*
3	*B01001111*
4	*B01100110*
5	*B01101101*
6	*B01111101*
7	*B00000111*
8	*B01111111*
9	*B01101111*

Fill out this table during "**Binary Conversion**"

1. Please convert the binary numbers representing digits 0 to 9 into decimal and hexadecimal numbers using a calculator, and fill in the table. This will give you a quick reference guide for base conversions.

Number	Binary	Decimal	Hexadecimal
0	B00111111	63	0x3F
1	B00000110	*6*	*0x06*
2	B01011011	*91*	*0x5B*
3	B01001111	*79*	*0x4F*
4	B01100110	*102*	*0x66*

5	B01101101	*109*	*0x6D*
6	B01111101	*125*	*0x7D*
7	B00000111	*7*	*0x07*
8	B01111111	*127*	*0x7F*
9	B01101111	*111*	*0x6F*

Lesson 29: Plant Monitor

Answer the following question during "Code Creation - Read Soil Moisture"

1. In the code provided, we understand that higher moisture content results in a lower sensor value, and moisture is typically expressed as a percentage. How can we modify the code to display the soil moisture level as a percentage?

To display the soil moisture level as a percentage, you first need to understand the range of values returned by your moisture sensor when it is completely dry and completely wet. These values can then be used to map the sensor's analog output to a percentage scale, where 0% indicates completely dry and 100% indicates completely wet.

Here's how you can modify your code to calculate and display the soil moisture as a percentage:

```
const int moisturePin = A1;  // Define the pin where the soil moisture
sensor is connected

void setup() {
  Serial.begin(9600);  // Initialize serial communication at 9600
baud rate
}

void loop() {
  int moistureValue = analogRead(moisturePin);  // Read the analog
value from the moisture sensor
  Serial.print("Raw Moisture Value: ");
  Serial.println(moistureValue);  // Output the raw sensor value to
the serial monitor for observation

  // Map the moisture value from the sensor's possible range (0 to 1023)
to a percentage (0% to 100%)
  float moisturePercent = map(moistureValue, 0, 1023, 100, 0);
  Serial.print("Moisture Percentage: ");
  Serial.print(moisturePercent);
  Serial.println("%");  // Output the calculated moisture percentage
  delay(1000);  // Delay for one second before the next reading to
reduce data flooding
}
```

Lesson 30: Arduino Radar System

Answer the following question during "3. Preparing the Ultrasonic Module"

1. In the above code, the ultrasonic module takes a reading every degree. If you feel that the readings are too frequent and want to take a reading every 5 degrees, how should the code be modified?

To modify the code so that the ultrasonic module takes a reading every 5 degrees instead of every degree, you would need to adjust the loop that controls the servo's movement. Specifically, you would change the increment in the for-loop from 1 to 5. This will cause the servo to move 5 degrees on each iteration of the loop, thus reducing the frequency of the readings.

Here's how you can adjust the loop in the code:

```
void loop() {
  // rotates the servo from 15 to 165 degrees in steps of 5 degrees
  for (int i = 15; i <= 165; i += 5) {
   ...
  }
  // rotates the servo from 165 to 15 degrees in steps of 5 degrees
  for (int i = 165; i >= 15; i -= 5) {
   ...
  }
}
```

Lesson 31: Guess Number

After completing the lesson, answer the following question

What additional components can be added to enhance the fun of the game? What roles do they play in the game?

To enhance the fun of the number guessing game, consider adding these components:

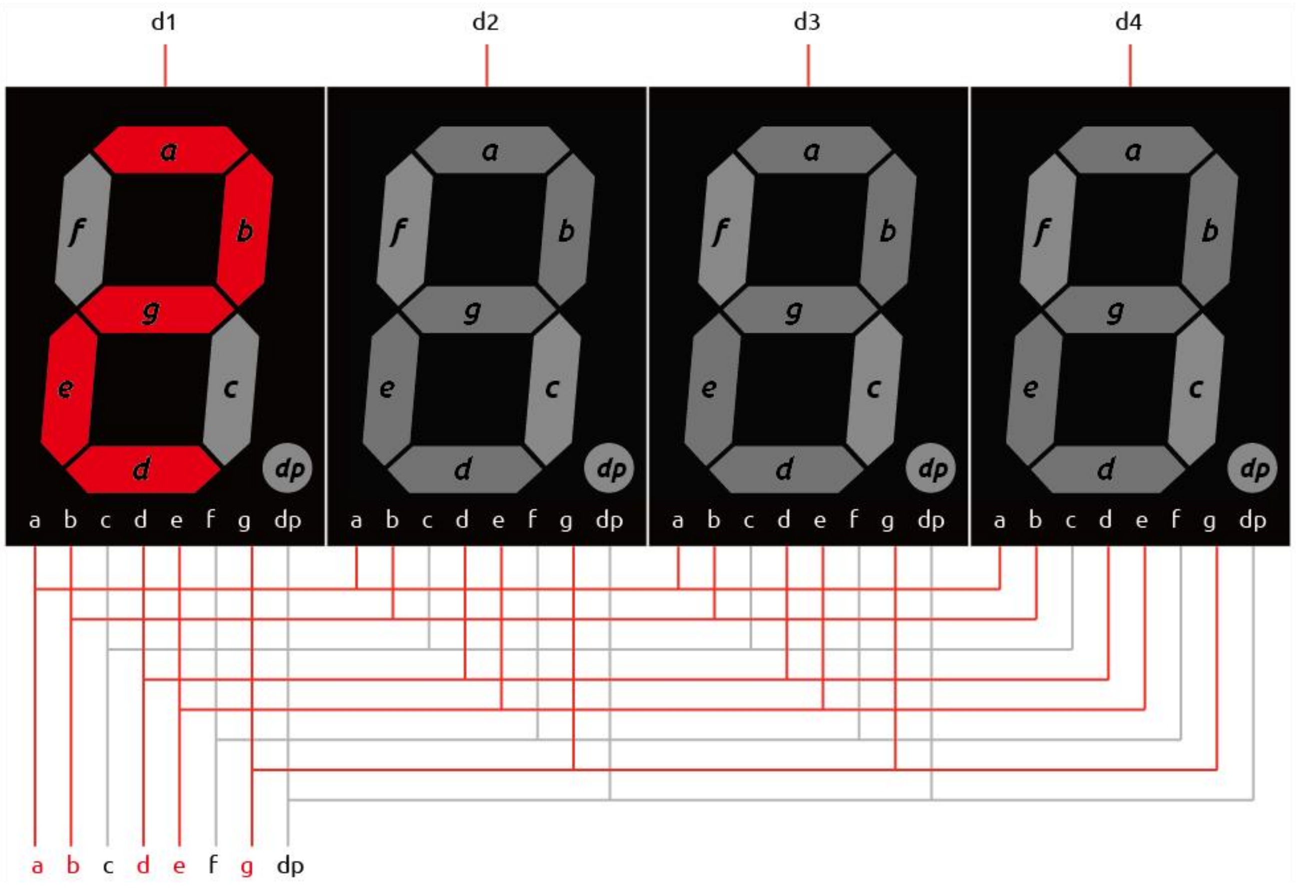
- **Sound Effects with a Speaker or Buzzer:** Provide auditory feedback for correct or incorrect guesses, making the game more engaging.
- **RGB LEDs or LED Strips:** Use visual effects like flashing colors for feedback, making the game visually exciting.
- **Scoreboard with an Additional Display:** Show scores or a timer to add competitiveness.
- **Wi-Fi or Bluetooth Module:** Enable remote play or multiplayer modes, allowing players to compete from different locations.
- **Touchscreen Display:** Upgrade the interface for a more interactive and modern gameplay experience.

These additions can make the game more interactive, competitive, and enjoyable.

Lesson 32: Stopwatch

Answer the following question during "Learn 4-Digit 7-Segment Display"

1. If you want the leftmost digit (d1) of the 4-digit 7-segment display to show "2", what should be the levels of d1~d4 and a~g pins?



Pin	HIGH or LOW	7-segment Display	HIGH or LOW
d1	*LOW*	a	*HIGH*
d2	*HIGH*	b	*HIGH*
d3	*HIGH*	c	*LOW*
d4	*HIGH*	d	*HIGH*
		e	*HIGH*
		f	*LOW*
		g	*HIGH*
		dp	*LOW*

Answer the following question during "**Code Creation - Scrolling Numbers on One Digit**"

In programming, bitwise operations like AND and OR are crucial for manipulating individual bits of data. The bitwise AND operation (&), compares each bit of its operands, resulting in 1 if both bits are 1, and 0 otherwise. Conversely, the bitwise OR operation (|), results in 1 if at least one of the bits is 1, and 0 only if both bits are 0.

Given this information, consider the expression (B01011011 >> 2) | 1. After right-shifting the binary number B01011011 by 2 positions, what is the result of applying the bitwise OR with 1?

1) **Right-Shifting:** The operation B01011011 >> 2 shifts the bits of B01011011 to the right by 2 positions:

- Original: 01011011
- After shifting: 00010110

2) **Bitwise OR with 1:** The bitwise OR operation (00010110 | 1) involves combining the right-shifted number with 1. Here's what happens:

- Before OR: 00010110
- After OR: 00010111

Conclusion

The final result of (B01011011 >> 2) | 1 is 00010111, which is 23 in decimal. The | 1 operation ensures that the least significant bit (LSB) is set to 1, which slightly increases the number if the LSB was initially 0. This shows how right-shifting decreases the value by reducing its magnitude (dividing by powers of 2), while the OR operation can be used to ensure a specific bit pattern, such as setting the LSB.

Lesson 33: Exploring the RF522-RFID Module

After completing the lesson, answer the following question

1. Now that you understand how to use the RC522-RFID module for reading or writing card or tag information and displaying it on an LCD, how would you design a common access control system for everyday use? Describe your design approach.

For designing a practical access control system for everyday use, my approach would focus on usability, security, and scalability. Here's a breakdown of my design strategy:

System Components

- **RFID Reader (RC522)**: To handle user identification through RFID cards or tags, which are easy to distribute and manage.
- **Microcontroller (Arduino)**: Acts as the central processing unit to control the RFID reader, process the input, and manage the output actions.
- **LCD Display (I2C LCD1602)**: To provide real-time feedback to users, such as access status or instructions.
- **Buzzer**: For auditory feedback, indicating access granted or denied, which enhances the user experience.
- **Keypad**: Allows for an alternative form of entry via pin code, offering a backup in case RFID tags are forgotten or lost.

User Interaction

- Users would either swipe their RFID card or enter a pin code. The system would immediately provide visual feedback on the LCD and audible feedback via the buzzer.
- For successful authentication, the door unlocks for a predetermined time and then relocks automatically.
- For denied access, the system displays a message and sounds an alert.

Lesson 34: Access Control System

Answer the following question during "Code Creation - Making the Stepper Motor Rotate"

1. If you want to achieve a full rotation in one direction and then a full rotation in the opposite direction, continuing in this cycle, how should the code be modified?

To modify the code so that the stepper motor completes a full rotation in one direction and then reverses to complete a full rotation in the opposite direction, continuously cycling between the two directions, you can adjust the loop() function in your Arduino sketch. Here's how you can do it:

- Set the number of steps to make a full revolution. Since your stepper motor requires 2048 steps per revolution (as defined by the STEPS constant), you'll use this number to move the motor a full turn.
- Use the stepper.step() function with positive and negative values. Positive values will rotate the motor in one direction, and negative values will rotate it in the opposite direction.

Here's the modified version of your loop() function:

```
void loop() {
  // Set the speed of the stepper
  stepper.setSpeed(5);

  // Rotate clockwise
  stepper.step(STEPS);  // Move 2048 steps (one full revolution)
  delay(1000);          // Wait for 1 second

  // Rotate counterclockwise
  stepper.step(-STEPS);  // Move -2048 steps (one full revolution in
the opposite direction)
  delay(1000);           // Wait for 1 second
}
```

Answer the following question during "**Code Creation - Access Control System**"

1. Now that a basic access control system has been set up, what additional components could be added to enhance its functionality and flexibility?

Here are some additional components that could be integrated into your access control system to enhance its functionality and flexibility:

- **Buzzer for Audio Feedback:** Provides audio feedback for access events, enhancing user interaction with signals for granted or denied access, and system alerts.
- **Matrix Keypad for Code Entry:** Allows pin code entry as an alternate authentication method, useful for users without RFID cards or for systems requiring dual-factor authentication.
- **Camera for Visual Identification:** Captures images or videos for additional security verification and maintains a visual log of access events.
- **Motion Sensors:** Detects presence before reaching the control point, activating the system or alerting security to unauthorized access attempts.