Thank You for Choosing the Beginner's Lab Kit

Welcome to the Beginner's Lab Kit, a comprehensive starter pack designed specifically for newcomers to the world of electronics and programming. This kit includes an array of essential components such as LEDs, resistors, a buzzer, potentiometers, photoresistors, thermistors, push buttons, digital tubes, and an ultrasonic module. One of the standout features of this kit is the inclusion of a multimeter, an invaluable tool that allows you to measure current, voltage, and resistance within your circuits. This addition is particularly useful for deepening your understanding of how each component functions.

The course sequence provided with this kit is structured around the Arduino programming syntax, ensuring a logical and educational progression. This structure allows you to build circuits step-by-step while learning how to write the programs that control them. Throughout the course, you will encounter troubleshooting challenges that enhance your understanding of the material.

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

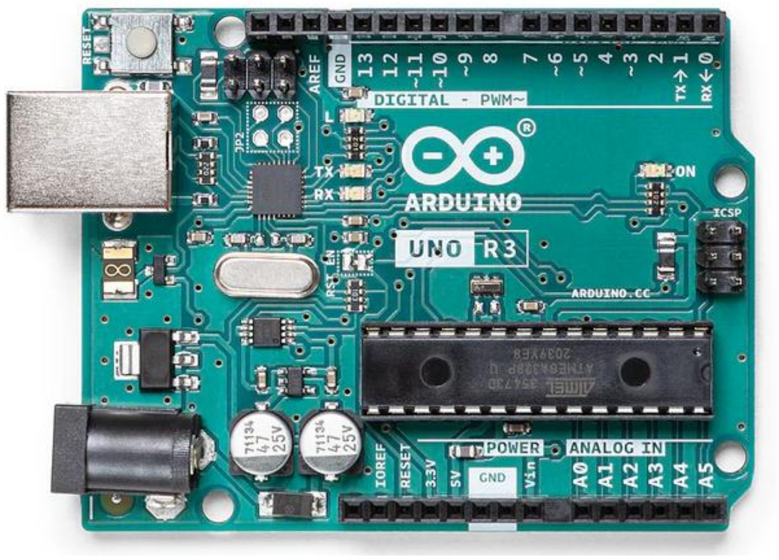
What's Included in Your Kit 1
Lesson 2: Your First Circuit 8
Lesson 3: Measure with Multimeter 9
Lesson4 : Ohm's Law 10
Lesson5 : Series Circuit vs. Parallel Circuit 11
Lesson6 : Blink LED 15
Lesson7 : Let's Make Traffic Lights! 16
Lesson8 : Traffic Light with Pedestrian Button 17
Lesson9 : Dimmable Desk Lamp 19
Lesson10 : ON/OFF Desk Lamp 20
Lesson11 : Controlling LED Arrays with Potentiometer 21
Lesson12 : The Colors of the Rainbow 22
Lesson13 : The Spectrum of Sight 23
Lesson14 : Random Colors 24
Lesson15 : Cool or Warm Colors 25
Lesson16 : Temperature Alarm 26
Lesson17 : Morse Code 27
Lesson18 : Light Alarm 28
Lesson19 : Reverse Parking Alarm System 29
Lesson20 : The Pomodoro Timer 30
Lesson21 : Siren Sound 32
Lesson22 : Play "Twinkle, Twinkle, Little Star" 34
Lesson23 : Cyber Dice 35
Lesson24 : Flowing Light with 74HC595 37
Lesson 25 Show Number 38

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.

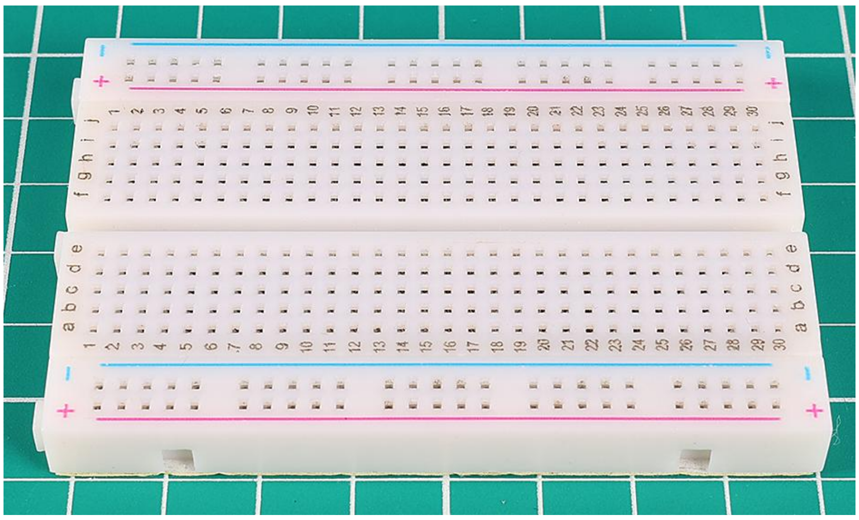
1 x Original Arduino Uno R3

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.



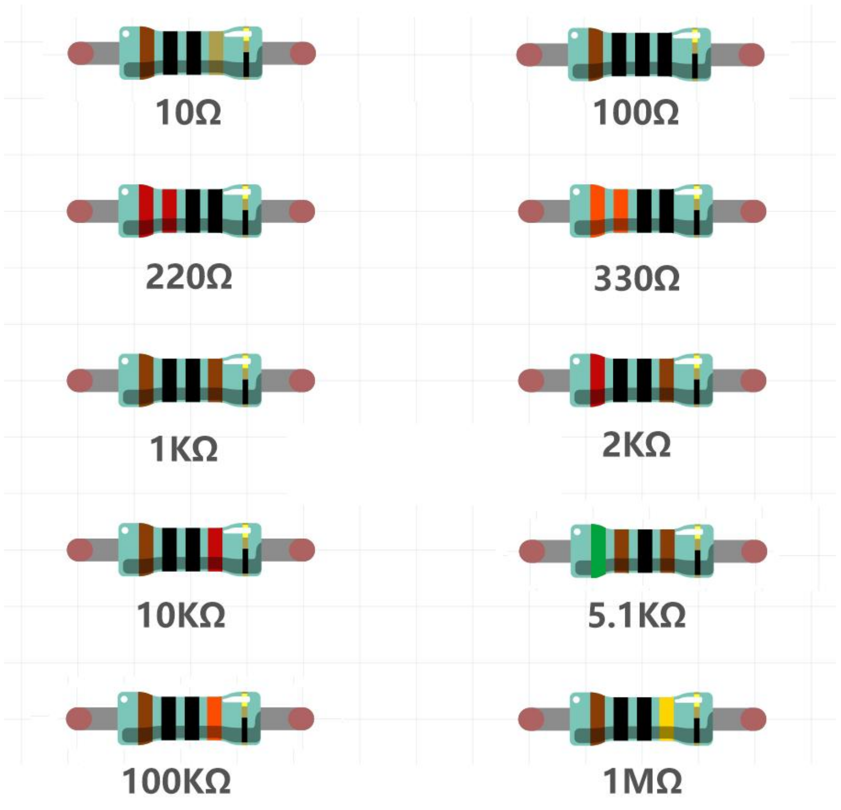
1 x 400-hole Breadboard

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



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.



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.

2 x RGB LEDs

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 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

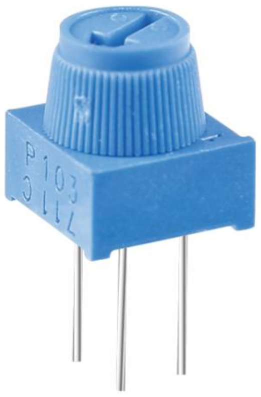
A thermistor is a resistor sensitive to temperature changes. NTC thermistors decrease resistance as temperature rises, while PTC thermistors increase resistance with temperature.

1 x 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.

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

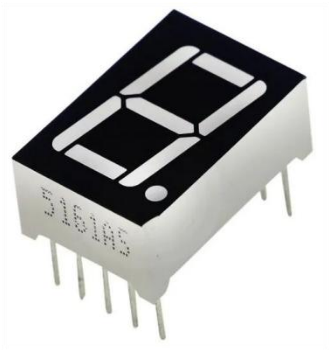
A small push-button is used to provide a physical response when pressed, commonly used in electronic devices to initiate actions or input commands.

1 x 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.

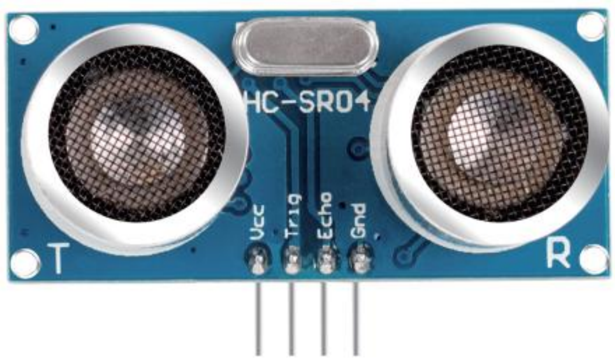
1 x 7-segment Display

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



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.



65 x Jumper Wires

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

10 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.

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 9V Battery

This is a non-rechargeable alkaline 9V battery. You need to install it on the multimeter.

1 x Multimeter with Red & Black Leads

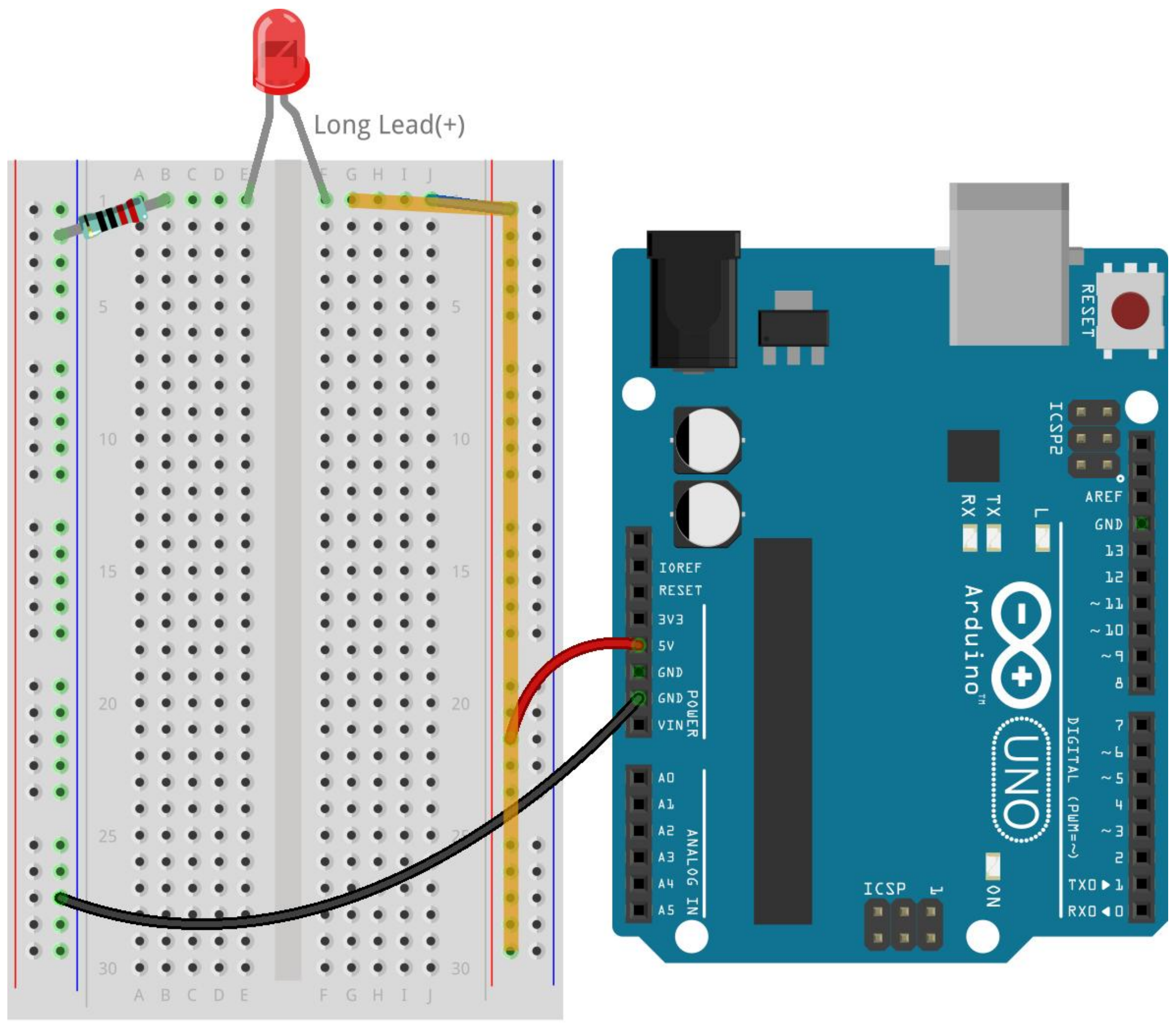
This is a versatile multimeter capable of measuring voltage, current, and resistance, as well as pertableing other electrical tests, making it an indispensable tool for electronics and electrical work.

SunFounder

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!**"

1. 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.

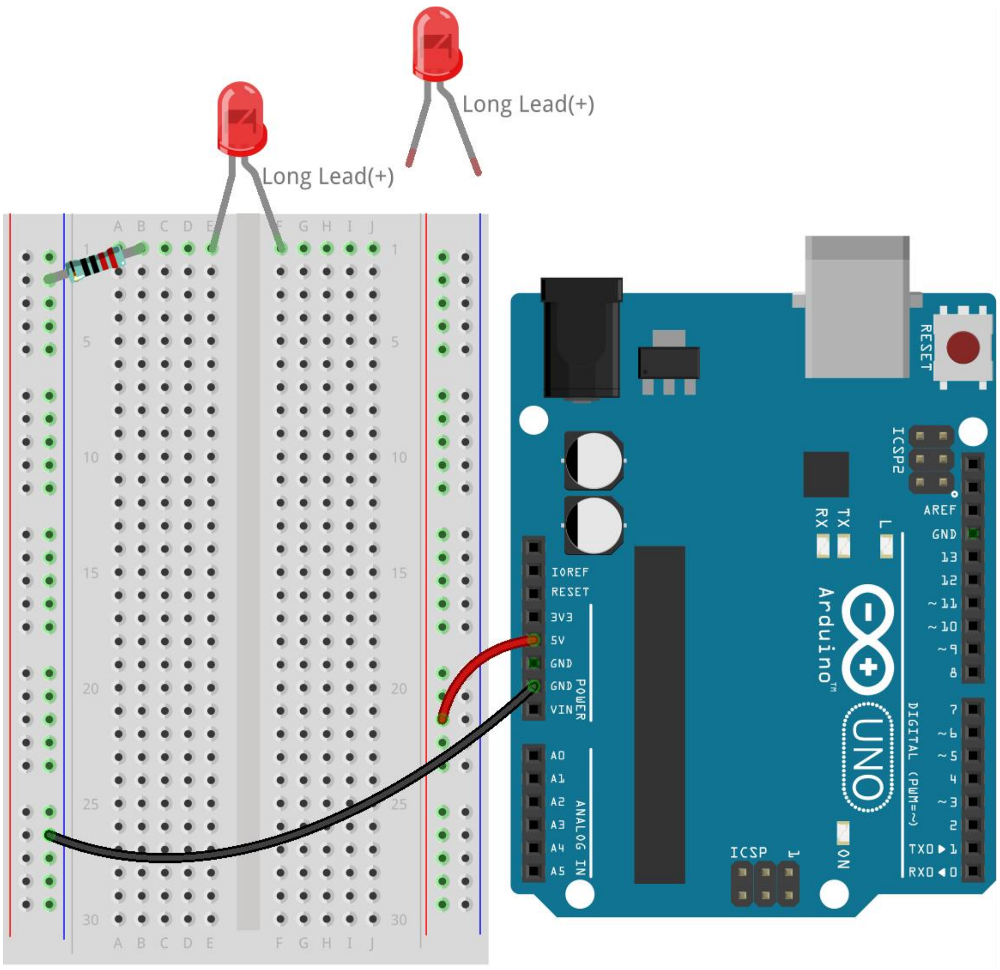
You will find that the LED is slightly dimmer at 3.3V than at 5V. Why is this?

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.

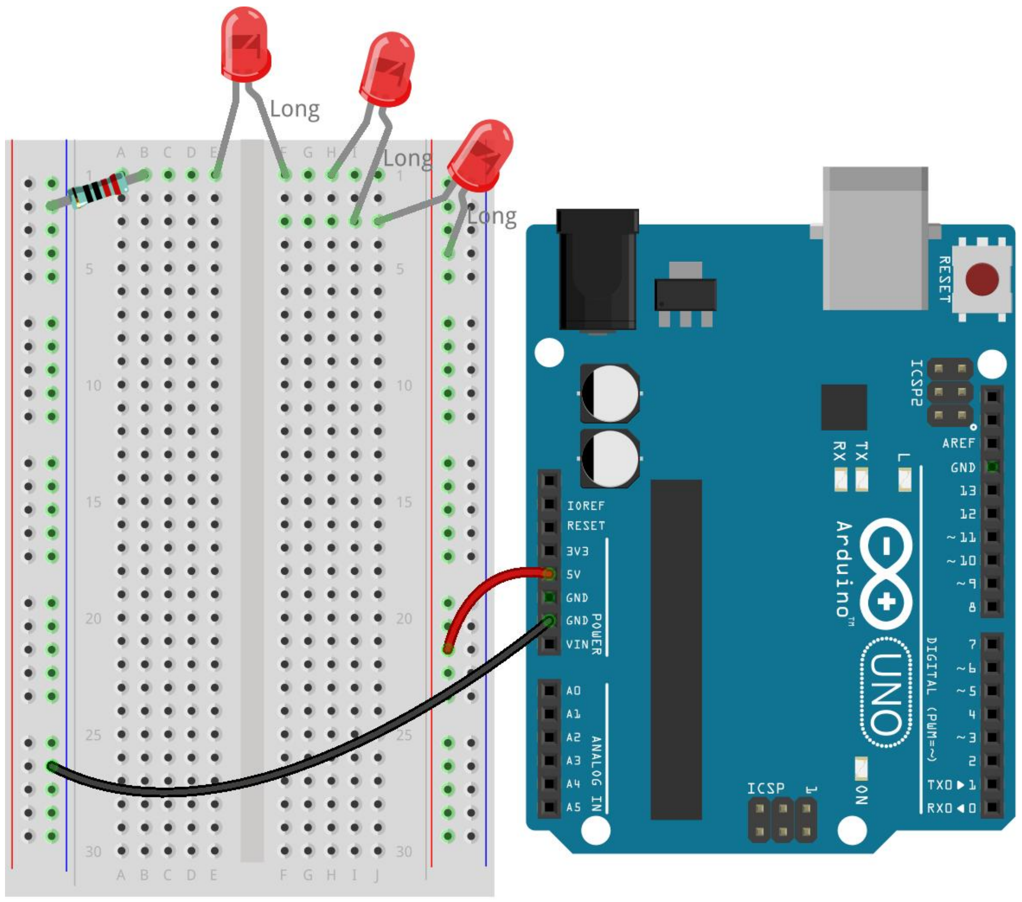
2. Measure the voltage of each component in the series circuit.

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*

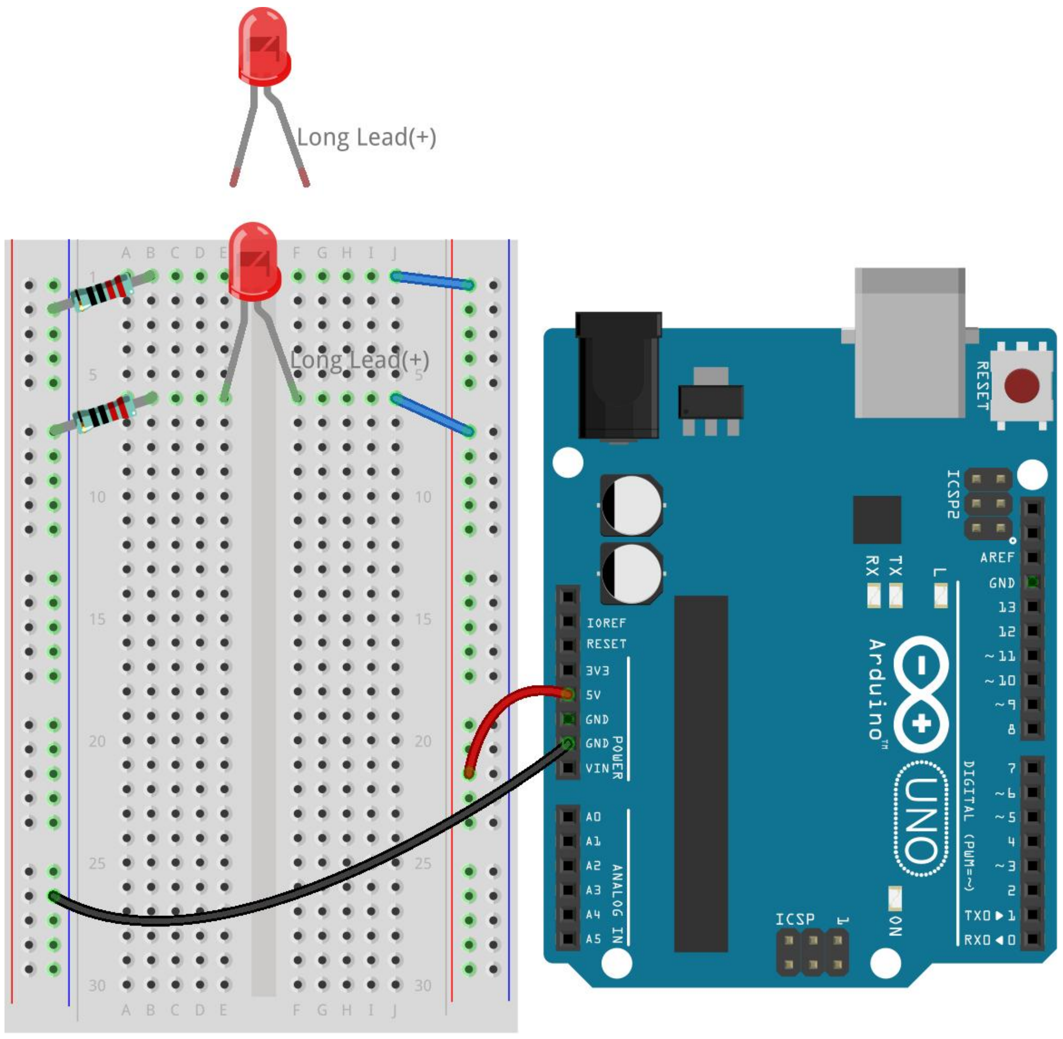
4. If another LED is added to this circuit, resulting in three LEDs, how does the brightness of the LEDs change? why? How do the voltages across the three LEDs change?



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.

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.

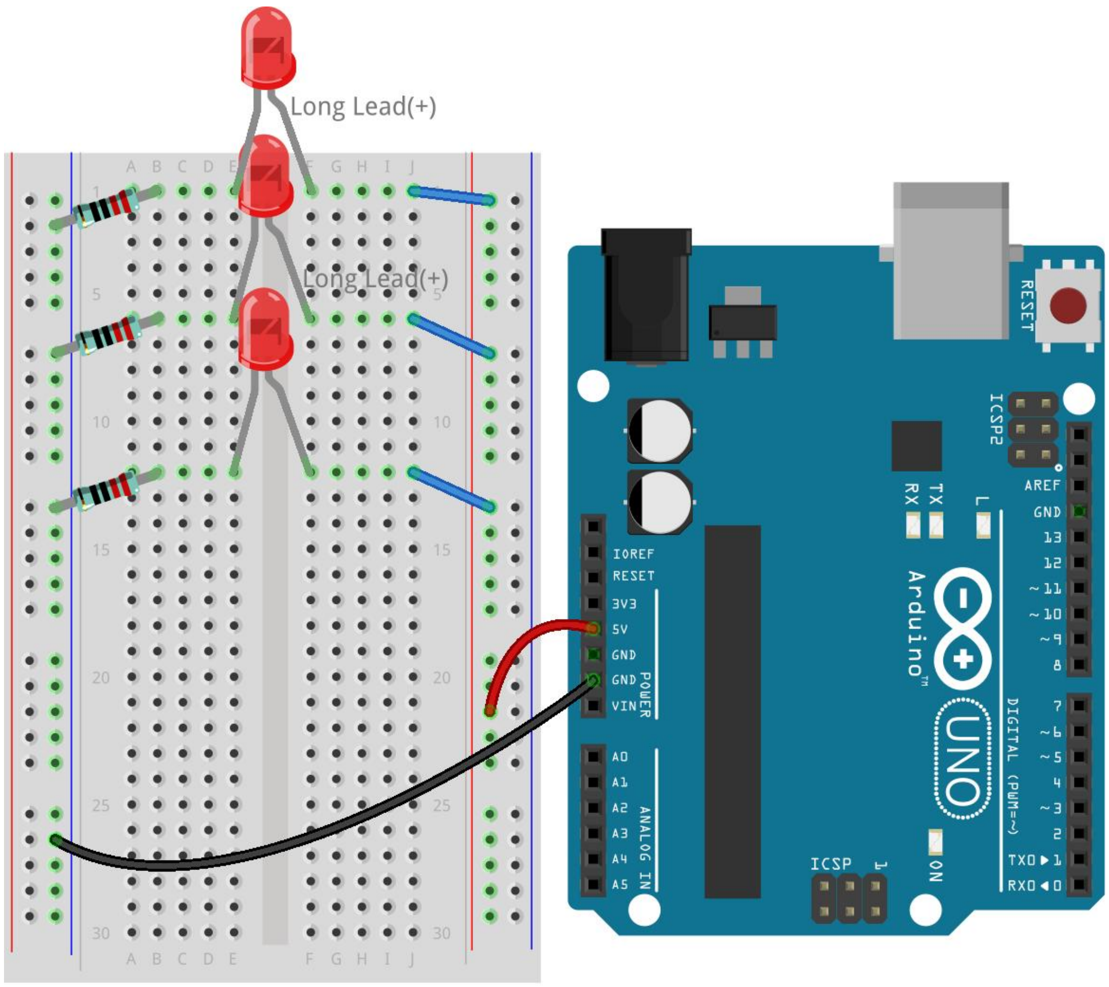
Circuit	Path1 Voltage	Path2 Voltage

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.
```

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

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 : ON/OFF Desk Lamp

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.

Lesson11 : Controlling LED Arrays with Potentiometer

Answer the following questions before proceeding to "**Code Creation**"

1. Write your pseudocode for the LED array.

Pseudocode serves as a program sketch, written in plain language to simplify understanding. Your task is to create pseudocode for an LED array that reacts to a potentiometer. As the potentiometer's value increases, more LEDs will light up.

Here's a simplified pseudocode that outlines the control of an LED array based on the input from a potentiometer:

```
Declare readValue variable.
Setup.
Declare 3 digital pin outputs.
Main Loop.
If the potentiometer's value is below 200, all LEDs should be off.
If the value is between 200 and 600, the first LED should be on.
If the value is between 600 and 1000, the first two LEDs should be on.
If the value exceeds 1000, all LEDs should be on.
Delay for a short time.
```

After completing the lesson, answer the following question

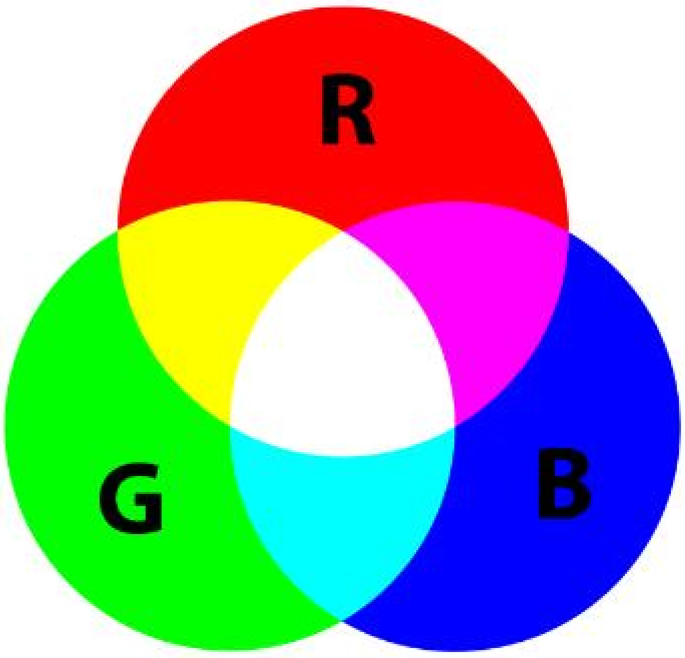
1. In the last code, we determine the number of LEDs to light up based on the value of the potentiometer. How can we modify the code so that, while lighting up the LEDs, their brightness changes in accordance with the potentiometer?

To modify your code so that the brightness of the LEDs changes in accordance with the potentiometer value, you can use the `analogWrite()` function instead of `digitalWrite()`. The `analogWrite()` function allows you to control the LED brightness through PWM (Pulse Width Modulation).

Lesson12 : The Colors of the Rainbow

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

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

Lesson13 : The Spectrum of Sight

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

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*

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

Color	Red	Green	Blue

Lesson14 : Random Colors

After completing the lesson, answer the following questions

1. If you change the code from randomSeed(analogRead(A0)) to randomSeed(0), how will the colors of the RGB LED change, and why?

If you change the code from randomSeed(analogRead(A0)) to randomSeed(0), the randomness of the RGB LED colors will be affected. The function randomSeed(seed) is used to initialize the pseudo-random number generator in Arduino, which influences the sequence of random numbers generated by functions like random().

Using analogRead(A0) as the seed value reads a somewhat random value from analog pin A0, which is typically affected by environmental noise and other factors, leading to different seed values each time the program starts. This means the sequence of random numbers (and thus the colors) will vary each time the Arduino is reset.

In contrast, setting the seed with a fixed value like randomSeed(0) initializes the random number generator with the same starting point every time the program runs. This results in the same sequence of random numbers, and therefore, the RGB LED will show the same color pattern every time you reset or power on the Arduino. This removes the randomness in the color changes of the LED.

2. What are some situations where randomness is used to solve problems in everyday life, aside from randomly picking colors for decoration and choosing lottery numbers?

Randomness is used in various everyday contexts to solve problems, including:

- **Board Games**: Rolling dice to determine moves, ensuring each game is different and fair.
- **Music Playlists**: Shuffling songs to keep your listening experience fresh and unpredictable.
- **Food Choices**: Randomly picking a restaurant or meal can make decision-making easier and more fun when you're unsure what to eat.
- **Seating Arrangements**: Drawing seats at random for events to mix guests and encourage social interactions.
- **Movie Nights**: Using a random draw to pick a movie when everyone has different preferences.

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 : 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.

Lesson17 : 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.

Lesson18 : Light Alarm

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

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

After completing the lesson, answer the following question

1. Cunning thieves might choose to steal at night, and if a painting disappears, the photoresistor might not be able to detect any change in light, thus failing to trigger an alarm. What can be done to improve this flaw?

One solution is to install a light source in front of the painting. This not only ensures that the painting is clearly visible and illuminated but also means that any disturbance or removal of the painting will immediately trigger the photoresistor's alarm by altering the light level detected.

Lesson19 : Reverse Parking Alarm System

After completing the lesson, answer the following question

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

To make the distance measurement more accurate to decimals, you can modify the calculation in the `measureDistance` function to use floating-point arithmetic instead of integer arithmetic. This will allow the result to include decimal values, providing more precise measurements.

```
float distance = duration * 0.034 / 2.0;
```

The `distance` variable in `measureDistance` function is changed to float to support decimal values.

The calculation for `distance` uses floating-point arithmetic by dividing by `2.0` instead of `2`.

Lesson20 : 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.

Lesson21 : 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.

Lesson22 : 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

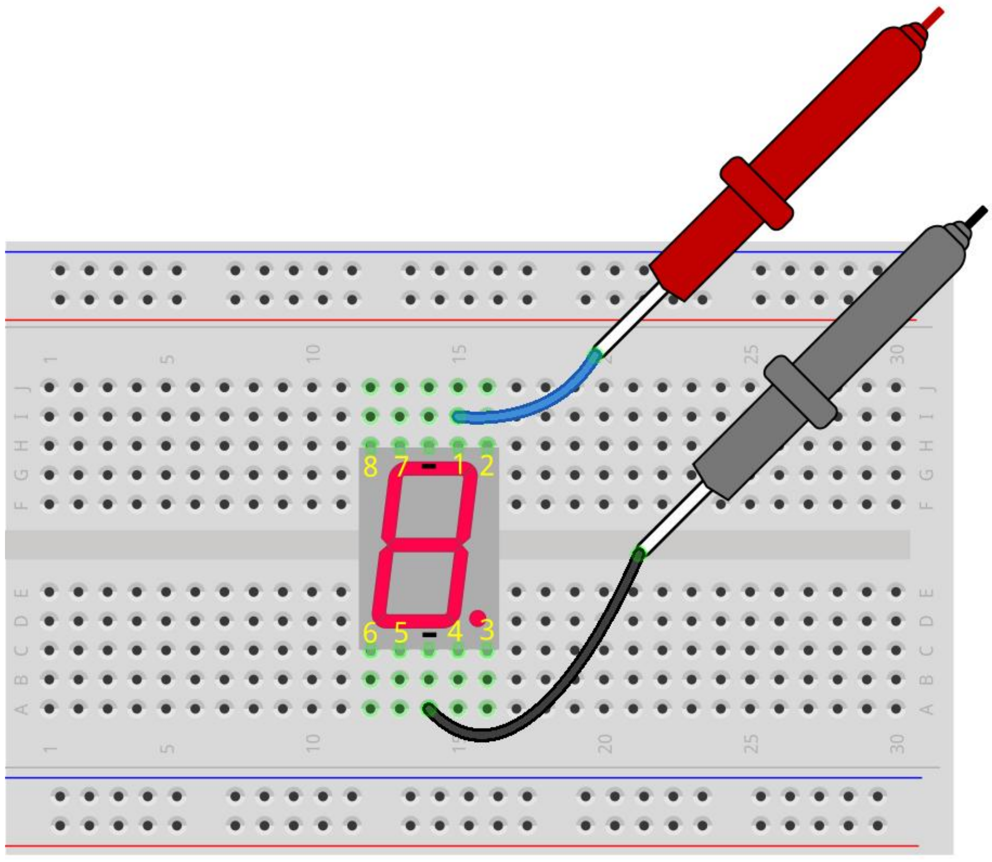
1. If you replace the passive buzzer in the circuit with an active buzzer, can you positively play “Twinkle Twinkle Little Star”? Why?

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.

Lesson23 : Cyber Dice

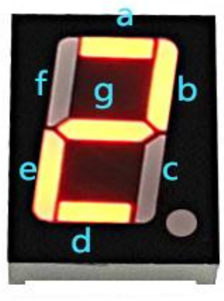
Answer the following questions during "**Understanding the 7-Segment Display**"

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.

Lesson24 : 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 };
```

Lesson 25 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*