For Loop Iteration:

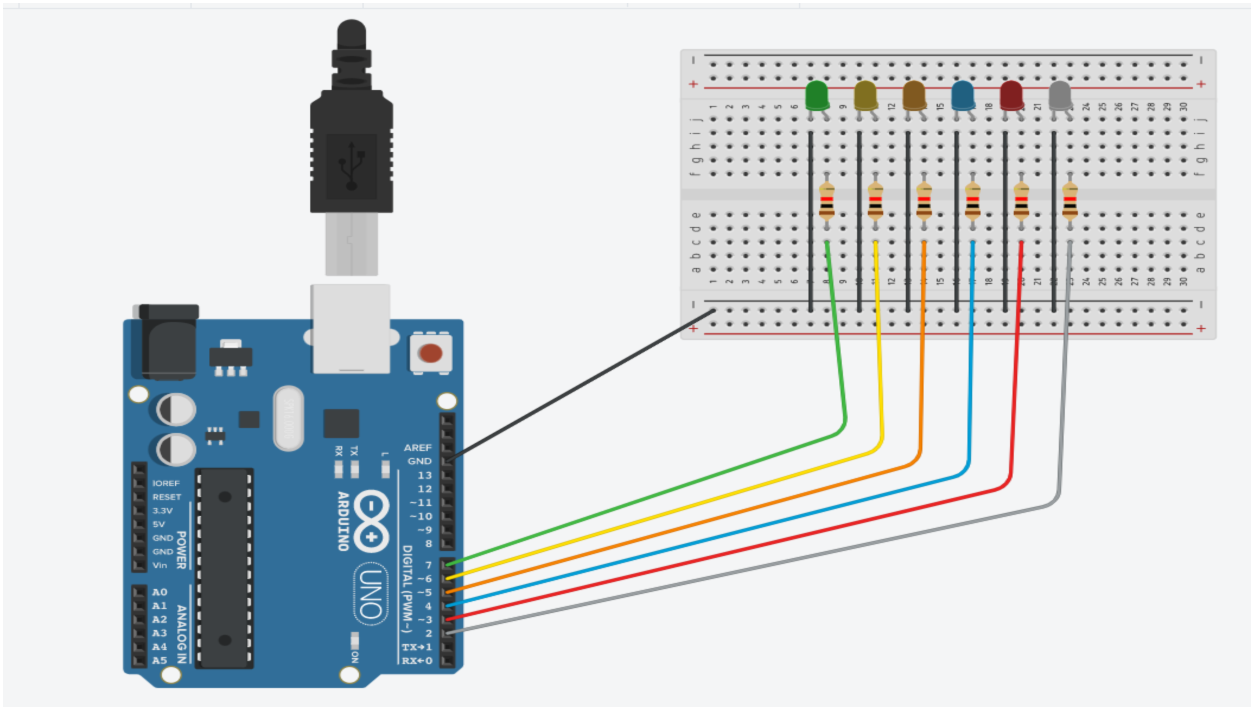
Often you want to iterate over a series of pins and do something to each one. For instance, this example blinks 6 LEDs attached to the Arduino by using a **for()** loop to cycle back and forth through digital pins 2-7. The LEDS are turned on and off, in sequence, by using both the digitalWrite() and delay() functions .
We also call this example "Knight Rider" in memory of a TV-series from the 80's where David Hasselhoff had an AI machine named KITT driving his Pontiac. The car had been augmented with plenty of LEDs in all possible sizes performing flashy effects. In particular, it had a display that scanned back and forth across a line, as shown in this exciting fight between KITT and KARR. This example duplicates the KITT display.

Hardware Required

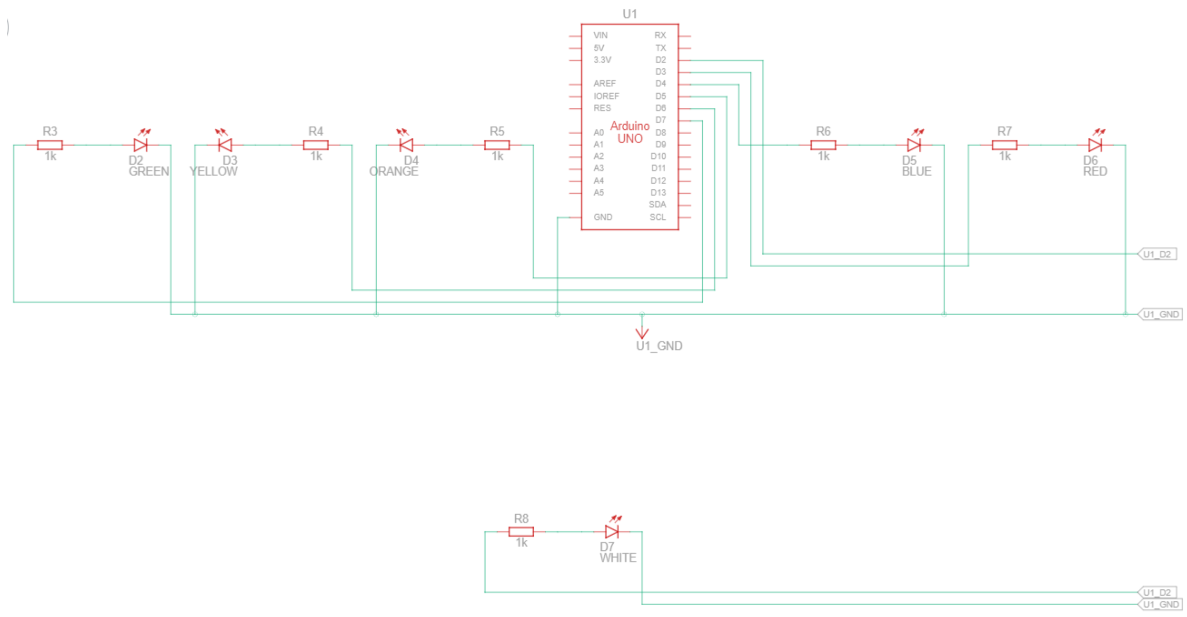
- Arduino Board
- 6 220 ohm resistors
- 6 LEDs
- hook-up wires
- breadboard

CIRCUIT:

Connect six LEDS, with 220 ohm resistors in series, to digital pins 2-7 on your Arduino.



SCHEMATIC:



CODE:

The code below begins by utilizing a for () loop to assign digital pins 2-7 as outputs for the 6 LEDs used.
In the main loop of the code, two for () loops are used to loop incrementally, stepping through the LEDs, one by one, from pin 2 to pin seven. Once pin 7 is lit, the process reverses, stepping back down through each LED.

```
int timer = 100;           // The higher the number, the slower the timing.
void setup()
{
  // use a for loop to initialize each pin as an output:
  for (int thisPin = 2; thisPin < 8; thisPin++) {
    pinMode(thisPin, OUTPUT);

  }
}
void loop()
{
  // loop from the lowest pin to the highest:
  for (int thisPin = 2; thisPin < 8; thisPin++) {
    // turn the pin on:
    digitalWrite(thisPin, HIGH);
    delay(timer);
    // turn the pin off:
    digitalWrite(thisPin, LOW);
  }
  // loop from the highest pin to the lowest:
```

```
  for (int thisPin = 7; thisPin >= 2; thisPin--) {
    // turn the pin on:
    digitalWrite(thisPin, HIGH);
    delay(timer);
    // turn the pin off:
    digitalWrite(thisPin, LOW);

  }
}
```

How to Use Arrays:

A variation on the for Loop example that demonstrates how to use an array.

This variation on the For Loop Iteration example shows how to use an **array**. An array is a variable with multiple parts. If you think of a variable as a cup that holds values, you might think of an array as an ice cube tray. It's like a series of linked cups, all of which can hold the same maximum value.
The For Loop Iteration example shows you how to light up a series of LEDs attached to pins 2 through 7 of the Arduino board, with certain limitations (the pins have to be numbered contiguously, and the LEDs have to be turned on in sequence).
This example shows you how you can turn on a sequence of pins whose numbers are neither contiguous nor necessarily sequential. To do this is, you can put the pin numbers in an **array** and then use **for loops** to iterate over the array.

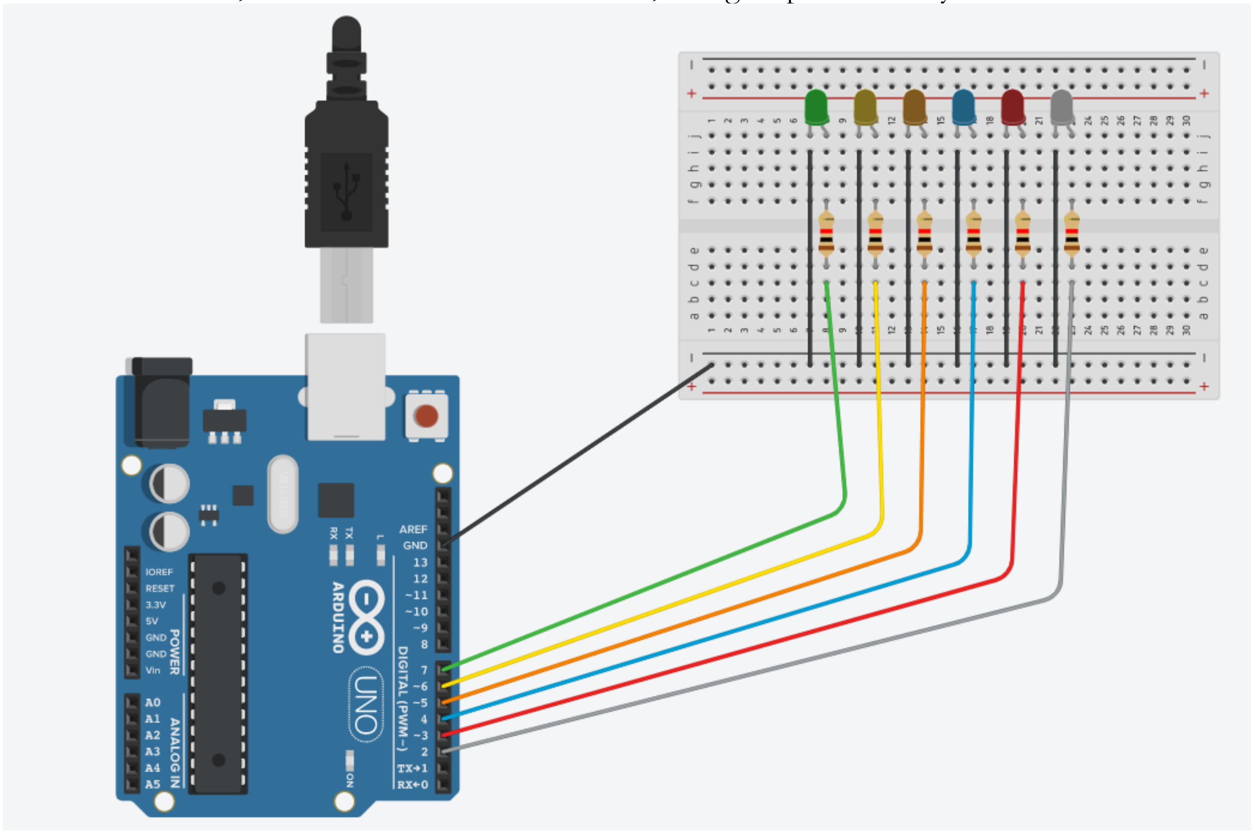
This example makes use of 6 LEDs connected to the pins 2 - 7 on the board using 220 ohm resistors, just like in the For Loop. However, here the order of the LEDs is determined by their order in the array, not by their physical order.
This technique of putting the pins in an array is very handy. You don't have to have the pins sequential to one another, or even in the same order. You can rearrange them in any order you want.

Hardware Required

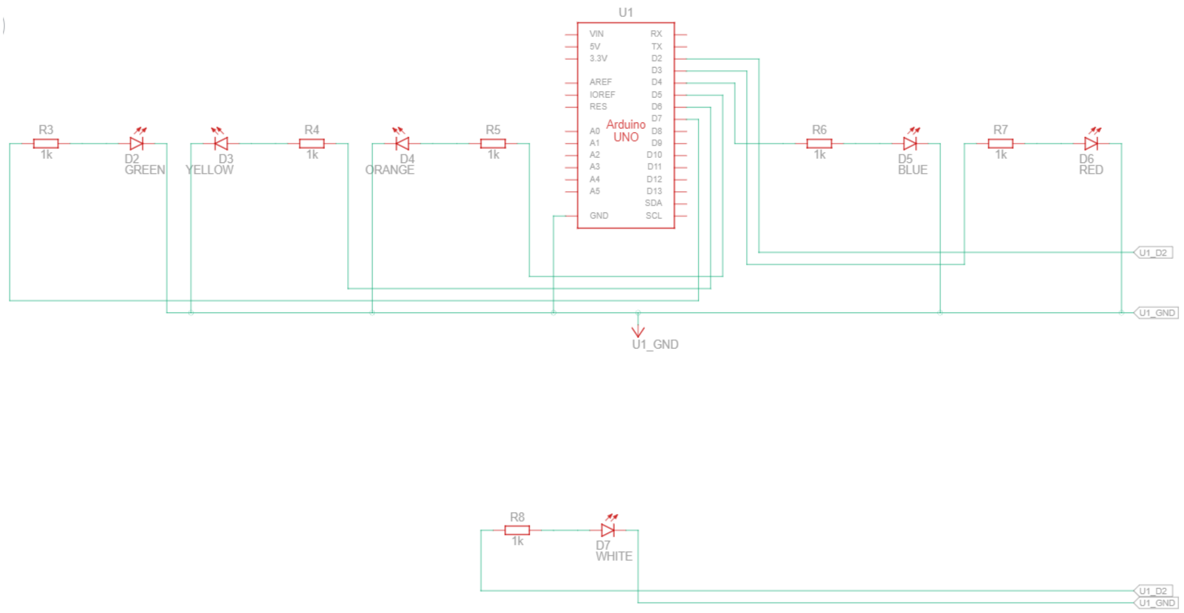
- Arduino Board
- 6 LEDs
- 6 220 ohm resistors
- hook-up wires
- breadboard

CIRCUIT:

Connect six LEDs, with 220 ohm resistors in series, to digital pins 2-7 on your board.



SCHEMATIC:



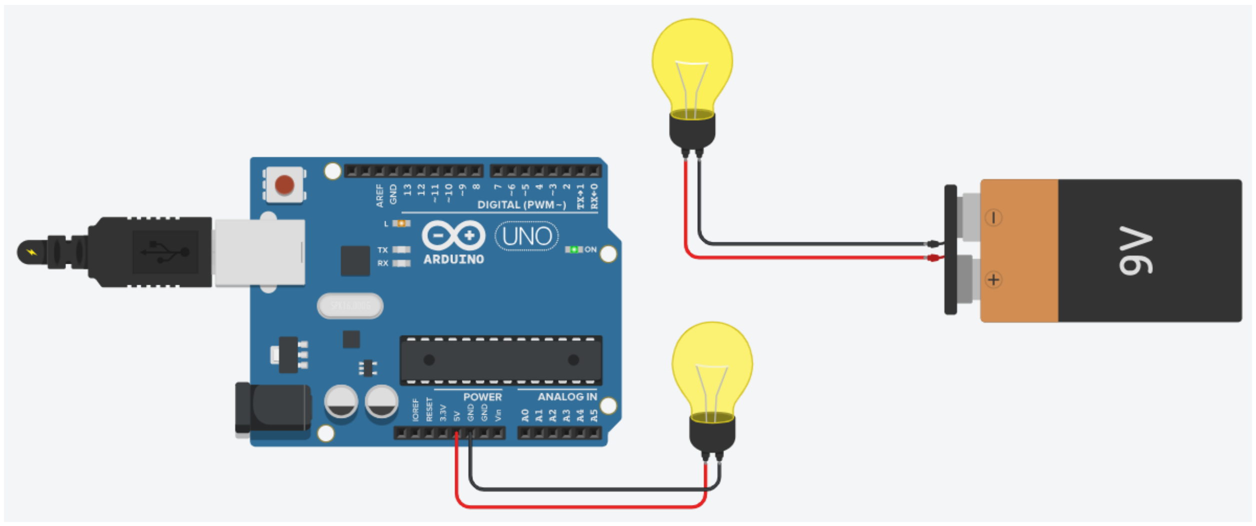
CODE:

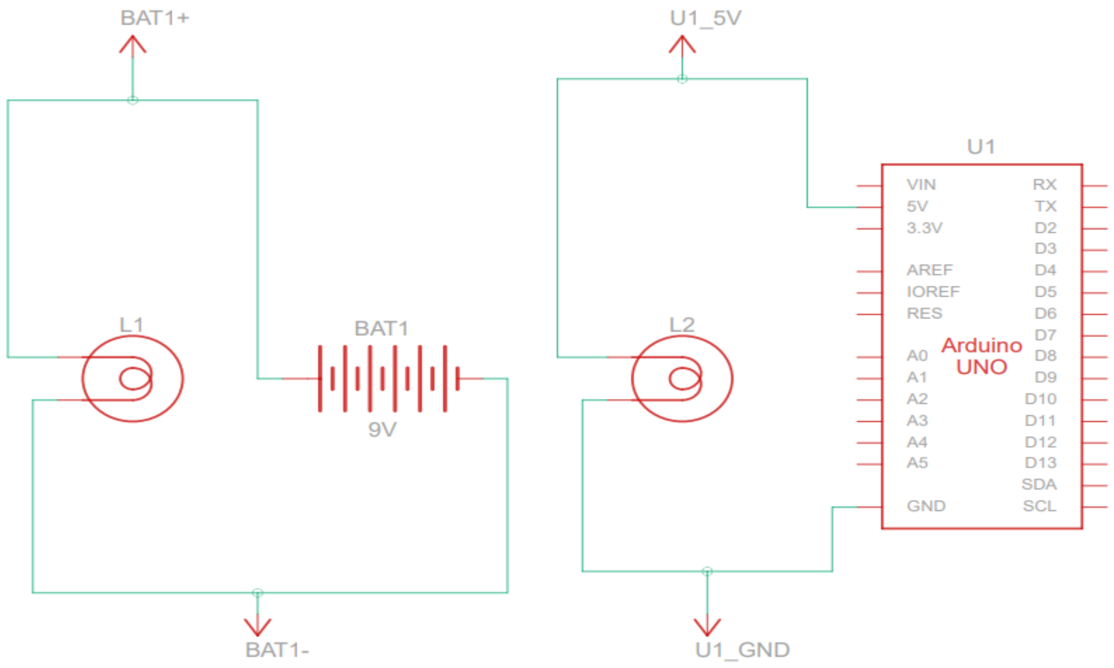
```
int timer = 100;// The higher the number, the slower the timing.
int ledPins[] = {2, 7, 4, 6, 5, 3};// an array of pin numbers to which LEDs are attached
int pinCount = 6;// the number of pins (i.e. the length of the array)

void setup()
{
  // the array elements are numbered from 0 to (pinCount - 1).
  // use a for loop to initialize each pin as an output:
  for (int thisPin = 0; thisPin < pinCount; thisPin++)
  {
    pinMode(ledPins[thisPin], OUTPUT);

  }
}
void loop() {
  // loop from the lowest pin to the highest:
  for (int thisPin = 0; thisPin < pinCount; thisPin++) {

    // turn the pin on:
    digitalWrite(ledPins[thisPin], HIGH);
    delay(timer);

    // turn the pin off:

    digitalWrite(ledPins[thisPin], LOW);

  }

  // loop from the highest pin to the lowest:

  for (int thisPin = pinCount - 1; thisPin >= 0; thisPin--) {

    // turn the pin on:

    digitalWrite(ledPins[thisPin], HIGH);

    delay(timer);

    // turn the pin off:

    digitalWrite(ledPins[thisPin], LOW);

  }
}
```

Simple Circuit with Battery & Arduino.

A simple circuit consisting of an Arduino Uno board, a 9V battery, and an LED connected to Arduino Board.

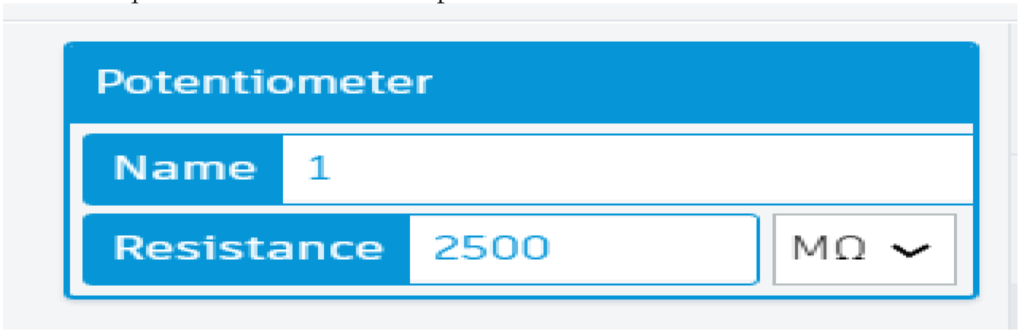
- Arduino Uno: This is the microcontroller board that controls the LED.
- 9V battery: This provides power to the Arduino Uno board and the LED.
- LED: This is a light-emitting diode that is connected to digital pin 9 of the Arduino Uno board. The positive leg of the LED (longer leg) is connected to a resistor, and the negative leg is connected to ground.





Multi voltage power supply in tinkercad.

1 volt is equivalent to 1000 milliamperes.

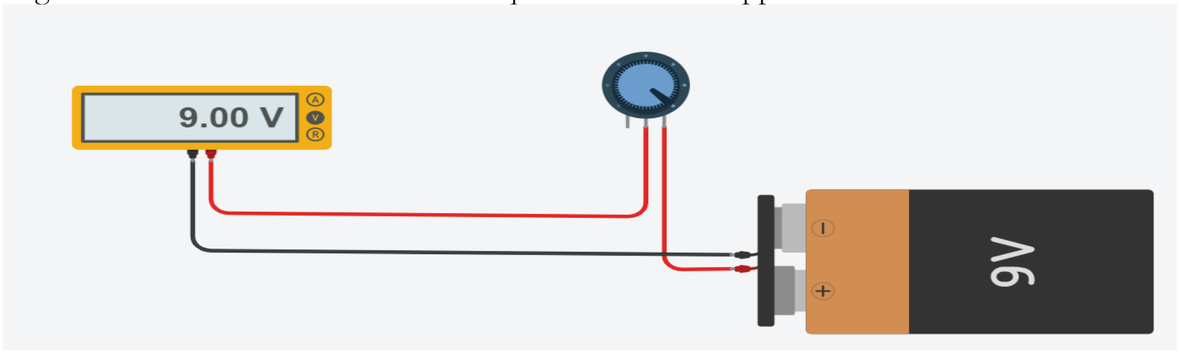


Potentiometer

Name: 1

Resistance: 2500 MΩ

In a specific application, such as controlling a motor or LED driver circuit, the potentiometer might indirectly control the current by adjusting voltage or resistance. In this case, the value of 2500 mA might have been chosen based on the requirements of the application.



- **Battery:** This is a 9V battery that provides power to the circuit.
- **Multimeter:** This is a device that can measure voltage, current, and resistance. In this diagram, the multimeter is set to measure voltage.
- **Variable resistor (potentiometer):** This is a component that can change its resistance, and therefore the voltage output, depending on the position of its knob. In this diagram, the potentiometer is connected between the battery and the output terminals.
- **Output terminals:** These are the terminals where the output voltage of the circuit can be measured or connected to other devices.

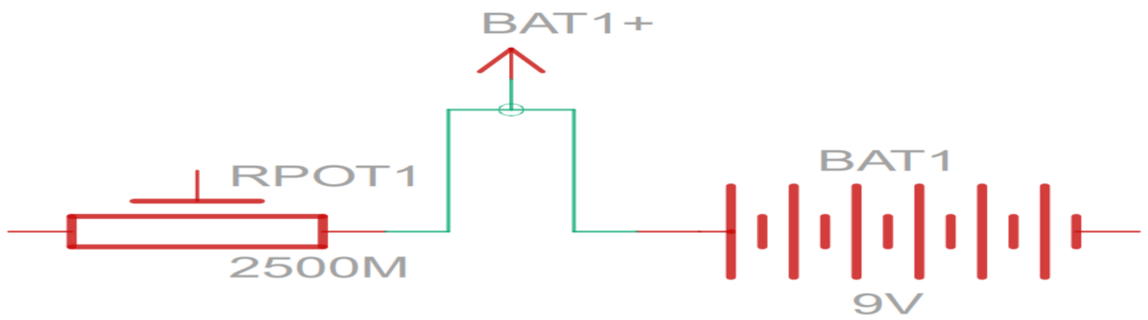
The way the circuit works is as follows:

1. The battery provides power to the circuit.
2. The voltage from the battery is passed through the variable resistor.
3. The position of the knob on the variable resistor determines how much of the voltage is dropped across it.
4. The remaining voltage is available at the output terminals.
5. The multimeter is used to measure the voltage at the output terminals.

By turning the knob on the variable resistor, you can adjust the output voltage of the circuit to any value between 0V and 9V.

Here are some additional things to keep in mind:

- The maximum current that the circuit can provide is limited by the battery and the variable resistor.
- The accuracy of the output voltage will depend on the accuracy of the variable resistor and the multimeter.



In the context of a schematic view in Tinkercad or any electronic schematic diagram, "RPOT1 2500M, BAT1+, BAT1 9V" represents various components and their properties.

- RPOT1 2500M: A potentiometer component labeled RPOT1. The "2500M" could represent a value or a setting associated with this potentiometer. "M" stands for milliamps (mA), but without further context, it's uncertain. Typically, potentiometers are used to adjust voltage or resistance, not directly set current.
- BAT1+: This represent the positive terminal of a battery labeled as BAT1. In electronic schematics, battery symbols are commonly labeled with a plus sign (+) to indicate the positive terminal.
- BAT1 9V: This represents a battery component labeled BAT1 with a voltage rating of 9V. This indicates that the battery provides a voltage of 9 volts.

In Tinkercad's schematic view, components are represented by symbols with labels, and connections between components are shown with lines. Each component has properties associated with it, such as resistance for resistors, capacitance for capacitors, etc. These properties can be set or adjusted depending on the component's specifications and the requirements of the circuit being designed.