Blinking LED With Arduino

Saturday, January 25, 2025 12:19 PM

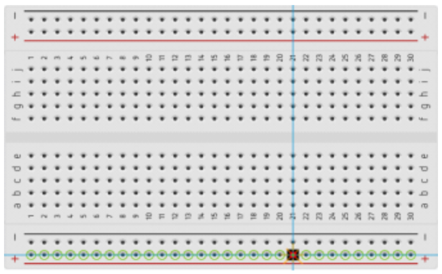
Fatemeh Nabidoust
Instagram: @elec_astro

- **how to control multiple LEDs using Arduino digital outputs and a breadboard.**

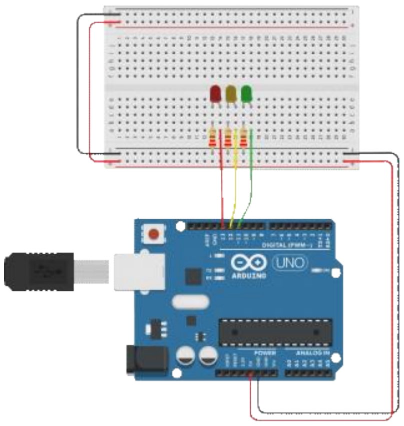
GOAL:
we'll connect some LEDs to the Arduino UNO and compose a simple program to light them up in a pattern

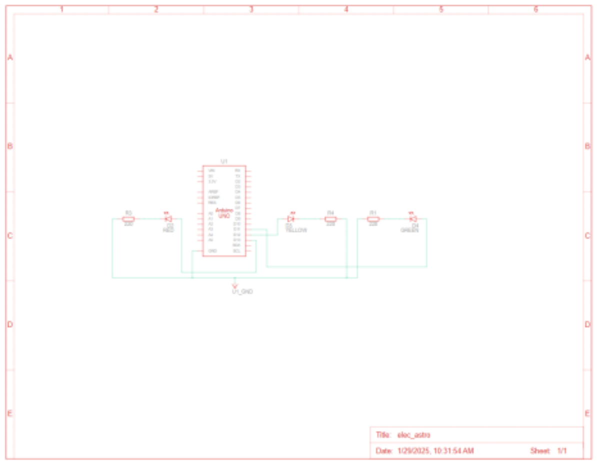
take a look at the workplane, where i'll add a breadboard from the components panel.

the rows of a solderless breadboard are connected inside, allowing you to connect components by plugging them into the same row as each other. the special long rails along the edges are for easy access to power and ground.



1.it's a best practice to always connect 5V and ground to these long rails as a starting point for any Arduino circuit. wire connections to 5V are typically red, and those connected to ground are typically black.



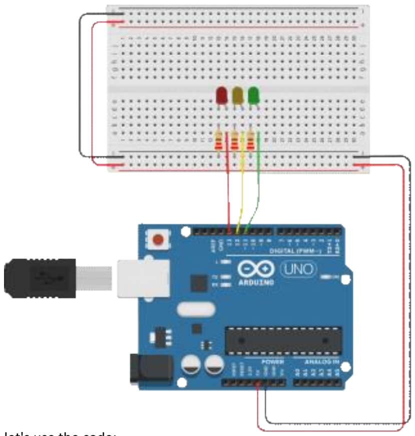


Name	Quantity	Component
U1	1	Arduino Uno R3
D2	1	Red LED
D3	1	Yellow LED
D4	1	Green LED
R3 R4 R1	3	220 Ω Resistor

2.add an LED to the breadboard so its legs go into two different rows of the breadboard.

attach wires to any of the holes in the same row to make an electrical connection. just like before, we want to connect the LED and resistor in series to pin 13 and ground. this breadboard circuit is electrically identical to this free-wired circuit.

3.let's add a few more LEDs to this circuit, along with their companion resistors. resistors are 220 ohm.
for each, connect one end to ground, in this case through the resistor, and the other to a different digital input pin on the Arduino board.



4.let's use the code:

before the setup(), we can use pure a variable is created.
it's called int because it's an integer, or any whole number.

```
int animationSpeed = 0;
```

inside the setup, just like last time, the pins are configured to be outputs, rather than inputs, using the pinMode() function.

```
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
}
```

the code inside the loop looks familiar, too, using digitalWrite() to set the pins HIGH and LOW, on and off, and pausing in between for a number of milliseconds.

and see, because we created the variable animationSpeed, if we change that number once at the start of the program, it will affect all of the places in the program that reference it.

so in this case, changing the variable animationSpeed will control the pauses, and therefore the overall speed of the animation.

```
void loop()
{
  animationSpeed = 400;
  digitalWrite(13, HIGH);
  delay(animationSpeed); // Wait for animationSpeed millisecond(s)
  digitalWrite(13, LOW);
  delay(animationSpeed); // Wait for animationSpeed millisecond(s)
  digitalWrite(12, HIGH);
  delay(animationSpeed); // Wait for animationSpeed millisecond(s)
  digitalWrite(12, LOW);
  delay(animationSpeed); // Wait for animationSpeed millisecond(s)
  digitalWrite(11, HIGH);
  delay(animationSpeed); // Wait for animationSpeed millisecond(s)
  digitalWrite(11, LOW);
  delay(animationSpeed); // Wait for animationSpeed millisecond(s)
}
```

Output: