Pushbutton Digital Input and LED Status With Arduino

Sunday, February 2, 2025 5:14 PM

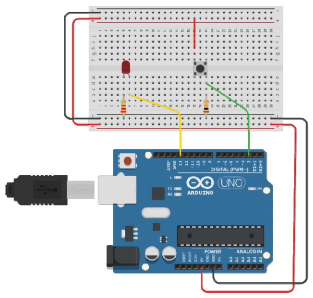
Fatemeh Nabidoust-Electronic & Electrical Department

How to Detect a pushbutton using Arduino's digital input.

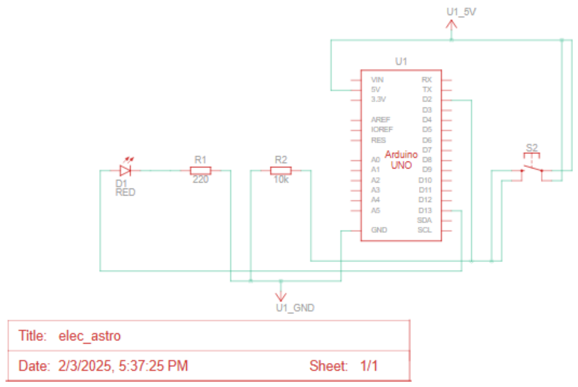
we'll create a bread circuit and compose a simple program to read the button and control a single LED.

we'll build upon the basic circuit for lighting up an LED connected to pin 13.

click to press the pushbutton. you can hold down shift while you click to keep the keep the button pressed down, and click again to release it.



add a pushbutton to your breadboard, straddling the center divider. one way to be sure you're wiring the switch properly is to connect any two diagonal legs. creat wires to connect one to 5 Volts and one to Arduino pin 2. that way, when the button is pressed, an electrical connection is made between power and the Arduino pin.
you'll also need to add a 10K or other larger resistor between the Arduino pin and ground. this is called a pull-down resistor. it gives the arduino pin a weak ground connection so the pin is never floating. Electricity takes the path of least resistance, so when the button is pressed, the pin will sense the connection to power stongly, and ignore the weak connection to ground. without this resistor, the floating pin cloud be affected by things like static electricity and give unreliable readings.



Code generate in Arduino IDE:

in the **setup()**, we use the **pinMode function** to establish digital pin2 as an input, and pin 13 as an output.

```
// C++ code
//
int buttonState = 0;
void setup()
{
  pinMode(2, INPUT);
  pinMode(13, OUTPUT);
}
```

you can see the code inside the loop uses a new function called digitalRead() to listen to a pin's state.

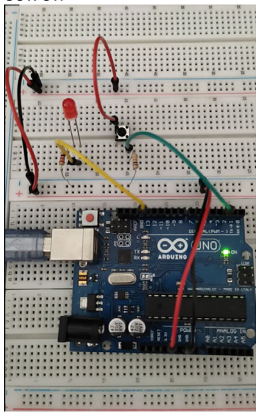
```
void loop()
{
  // read the state of the pushbutton
  buttonState = digitalRead(2);
```

after two more comment lines marked with double slashes, we find the if statement, which evaluates a condition using comparison operators like greater than, less than, or in out case, "equivalent to" noted with two equals signs.

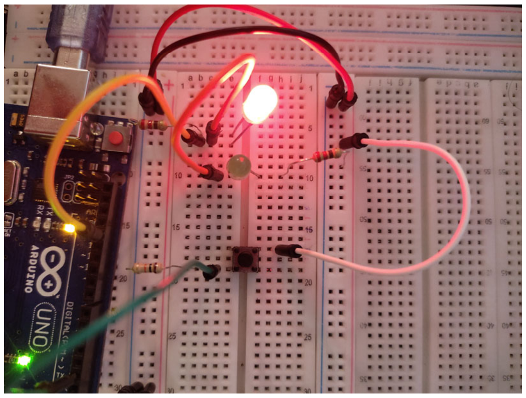
```
// check if pushbutton is pressed. if it is, the
  // button state is HIGH
  if (buttonState == HIGH) {
    digitalWrite(13, HIGH);
  } else {
    digitalWrite(13, LOW);
  }
  delay(10); // Delay a little bit to improve simulation performance
}
```

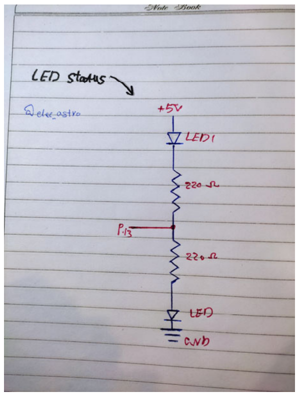
if the condition is met, the code inside the curly braces executes, setting the LED on.
if not, the code inside the else statement is executed instead, setting the LED off.

OUTPUT:



LED Status





```
// C++ code
//
int animationSpeed = 0;
void setup()
{
  pinMode(13, OUTPUT);
}
void loop()
{

  digitalWrite(13, HIGH);
  delay(200); // Wait for animationSpeed millisecond(s)
  digitalWrite(13, LOW);
  delay(200); // Wait for animationSpeed millisecond(s)
}
```