Programmed Control with Arduino

Activity 3: The Momentary Pushbutton (Astable Operation)

Until now, our circuits only sent information to the outside world by turning on LEDs (output channels). In this activity, we will take a crucial step: we will learn to receive data from the outside using a pushbutton (input channel). We will design an intelligent program that makes a logical decision and turns on an LED only while we keep the pushbutton pressed down.

Learning Objectives

By the end of this session, you will be able to:

- Configure digital pins as data input channels using `pinMode INPUT` .
- Read and monitor digital states (logical HIGH and LOW values) with the `digitalRead` command.
- Apply conditional control structures using the `if else` instruction.
- Understand the electrical need for a protective resistor (Pull-Down) to prevent a floating state (electrical noise).

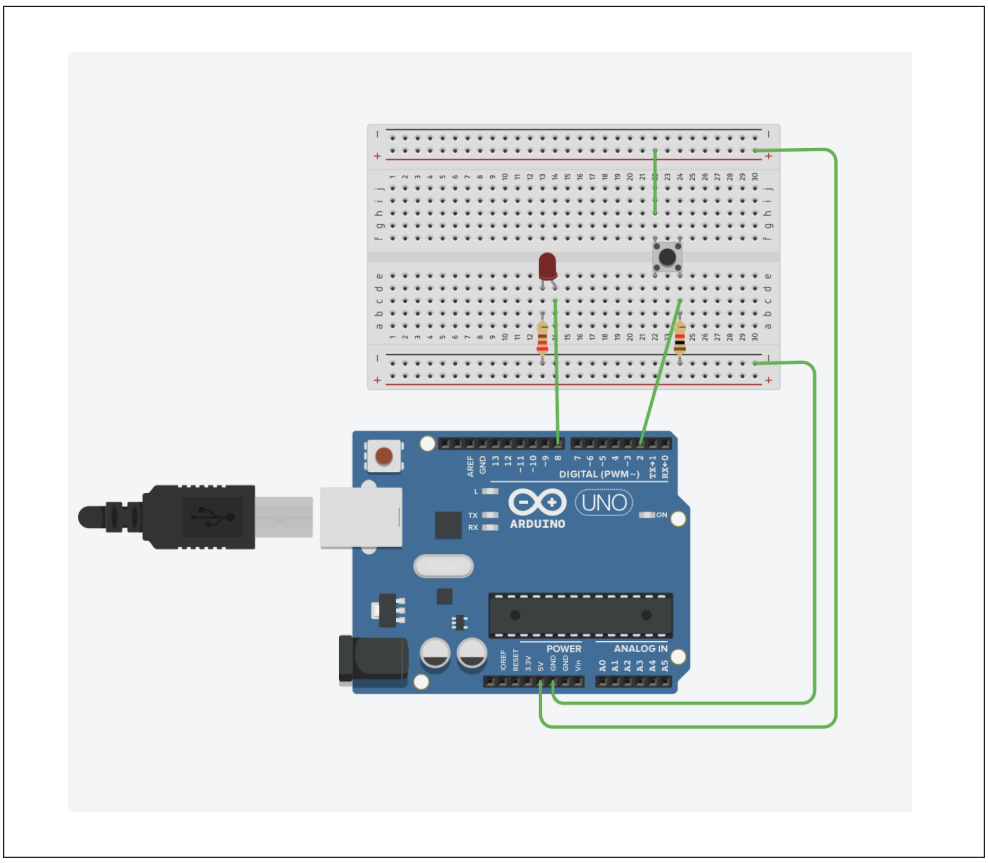
Required Components

Find and place the following components on your TinkerCad workspace:

- 1 Arduino Uno board.
- 1 Small breadboard.
- 1 LED and 1 $220\,\Omega$ resistor (for the LED).
- 1 Four-pin pushbutton.
- 1 $10\,\text{k}\Omega$ resistor (color code: brown, black, orange; to be used as a Pull-Down).
- Virtual jumper wires.

Breadboard Assembly Diagram

It is essential to connect the pushbutton properly. To ensure Arduino reads an electrical zero (0V) when nobody is pressing the button, we will use a **Pull-Down** resistor configuration.



Base Code: The if/else Conditional

Write the following program in the TinkerCad code editor. Pay close attention to how the code continuously decides the state of the LED by reading the voltage from the pushbutton.

```
// Variable declaration assigning the connection pins
int led = 8;         // The LED is connected to digital pin 8
int pushbutton = 2; // The pushbutton is connected to digital pin 2

void setup() {
  pinMode(led, OUTPUT);         // We configure the LED as a current OUTPUT
  pinMode(pushbutton, INPUT);    // We configure the pushbutton as a data INPUT
}

void loop() {
  // We create a local variable to store the pushbutton state (HIGH or LOW)
  int buttonState = digitalRead(pushbutton);

  // Conditional: If the pushbutton is pressed (receives 5V / HIGH)
  if (buttonState == HIGH) {
    digitalWrite(led, HIGH);  // Turns on the LED
  }
  // Otherwise (if it is not pressed / receives 0V / LOW)
  else {
    digitalWrite(led, LOW);    // Turns off the LED
  }
}
```

Listing 1: Code to control an LED with a momentary pushbutton.

How does the code work?

- `pinMode` `INPUT` : Tells the microcontroller that pin 2 will not supply current, but will instead passively "listen" to the voltage coming from the external circuit.
- `digitalRead` : Acts as a logical sensor. It reads the voltage at the specified pin. If it detects 5V, it will return the raw value `HIGH`, and if it detects 0V (GND), it will return `LOW`.
- **The comparison**: In programming, a single equals sign ( ) is used to store a value inside a variable. To check if two things are mathematically equal, you must use the double equals sign ( ).
- `if` `else` **(If / Otherwise)**: Evaluates a logical condition inside parentheses. If the condition is met, the lines enclosed within the `if` curly braces are executed. If the condition is false, the program skips that block and directly executes what is inside the `else` curly braces.

Activity 3 Challenge

Demonstrate your understanding of code execution flow by overcoming these three challenges:

1. **Inverse Logic**: Modify the program in TinkerCad so that the LED is always turned on by default, and turns **off** only while you keep the pushbutton pressed down.
2. **Two-Position Pedestrian Traffic Light**: Add a second LED (for example, green on pin 9 and red on pin 8). Make it so that, by default, the red LED is on and the green one is off. When the pushbutton is pressed, the red must turn off and the green must turn on.
3. *Question for reflection*: If we remove the $10\,\mathrm{k}\Omega$ resistor from the breadboard and connect the pushbutton directly to GND and 5V without it, what kind of electrical or physical anomaly do you think could happen to the Arduino board when the button is pressed?