Polling-Based Implementation

How It Works

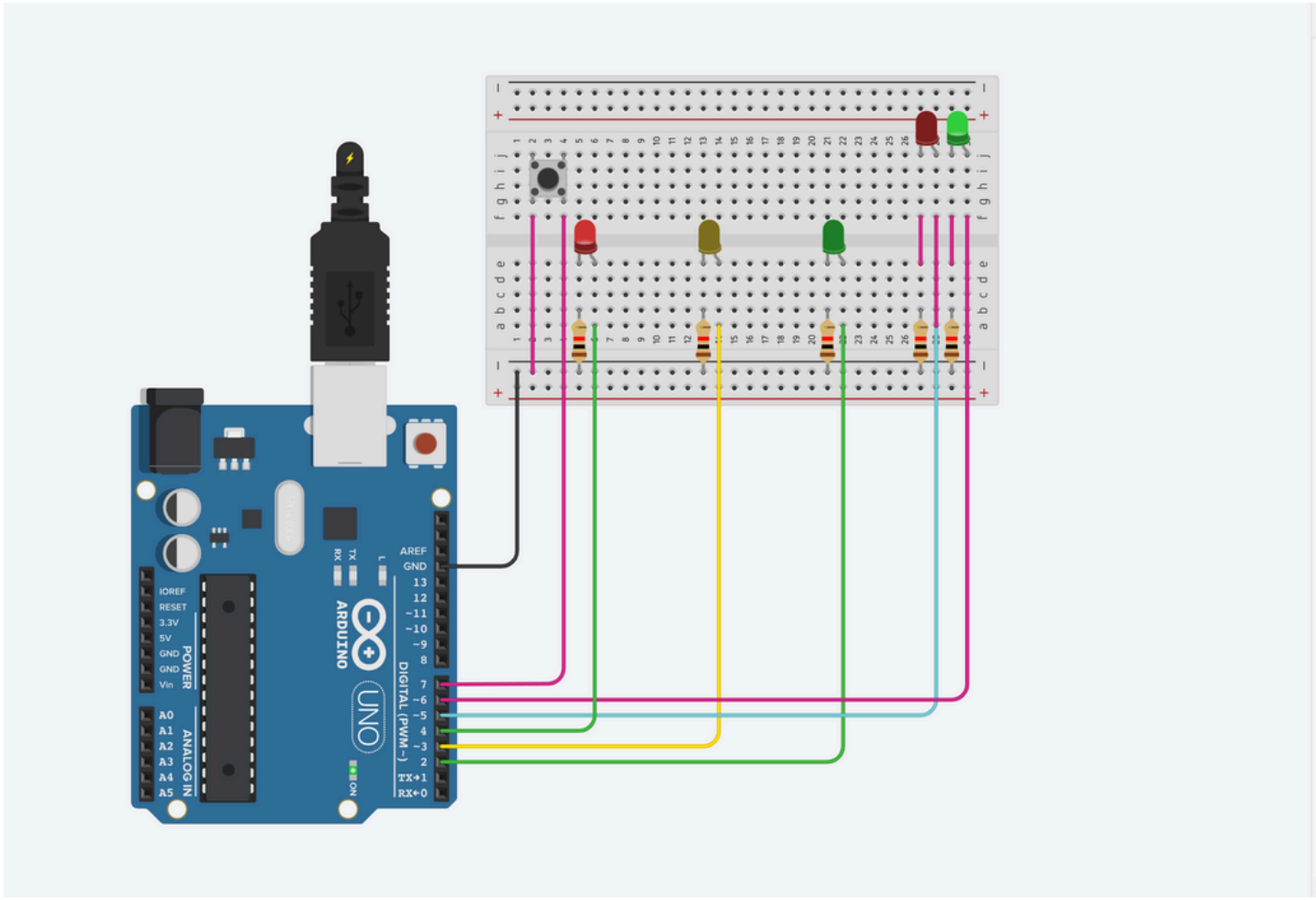
The loop() function continuously checks the state of the button using digitalRead(). If the button is pressed, the state is updated.

Characteristics

- Simple: Easy to implement and understand.
- Inefficient: Continuously checking the button consumes CPU resources, even when no button is pressed.
- Slower Response: Delays in detecting button presses can occur if the loop contains time-consuming tasks.

Use Case

Suitable for simple systems where the program doesn't have to handle multiple tasks or real-time constraints.



1 (Arduino Uno R3)

```
const int greenLight = 2;
const int yellowLight = 3;
const int redLight = 4;
const int pedRed = 5;
const int pedGreen = 6;
const int button = 7;

enum TrafficState { GREEN, WAIT, YELLOW, RED };
TrafficState currentState = GREEN;

unsigned long stateTimer = 0;
bool buttonPressed = false;

void setup() {
  pinMode(greenLight, OUTPUT);
  pinMode(yellowLight, OUTPUT);
  pinMode(redLight, OUTPUT);
  pinMode(pedRed, OUTPUT);
  pinMode(pedGreen, OUTPUT);
  pinMode(button, INPUT_PULLUP);

  digitalWrite(greenLight, HIGH);
  digitalWrite(pedRed, HIGH);
  stateTimer = millis();
}

void loop() {
  if (digitalRead(button) == LOW) {
    buttonPressed = true;
  }

  switch (currentState) {
    case GREEN:
      if (buttonPressed || millis() - stateTimer >= 10000) {
        buttonPressed = false;
        stateTimer = millis();
        currentState = WAIT;
      }
      break;

    case WAIT:
      if (millis() - stateTimer >= 3000) {
        stateTimer = millis();
        currentState = YELLOW;
      }
      break;

    case YELLOW:
      if (millis() - stateTimer >= 2000) {
        stateTimer = millis();
        currentState = RED;
      }
      digitalWrite(greenLight, LOW);
      digitalWrite(yellowLight, HIGH);
      break;

    case RED:
      if (millis() - stateTimer >= 10000) {
        stateTimer = millis();
        currentState = GREEN;
      }
      digitalWrite(yellowLight, LOW);
      digitalWrite(redLight, HIGH);
      digitalWrite(pedRed, LOW);
      digitalWrite(pedGreen, HIGH);

      if (currentState == GREEN) {
        digitalWrite(pedGreen, LOW);
```

Serial Monitor

Interrupt-Based Implementation

How It Works

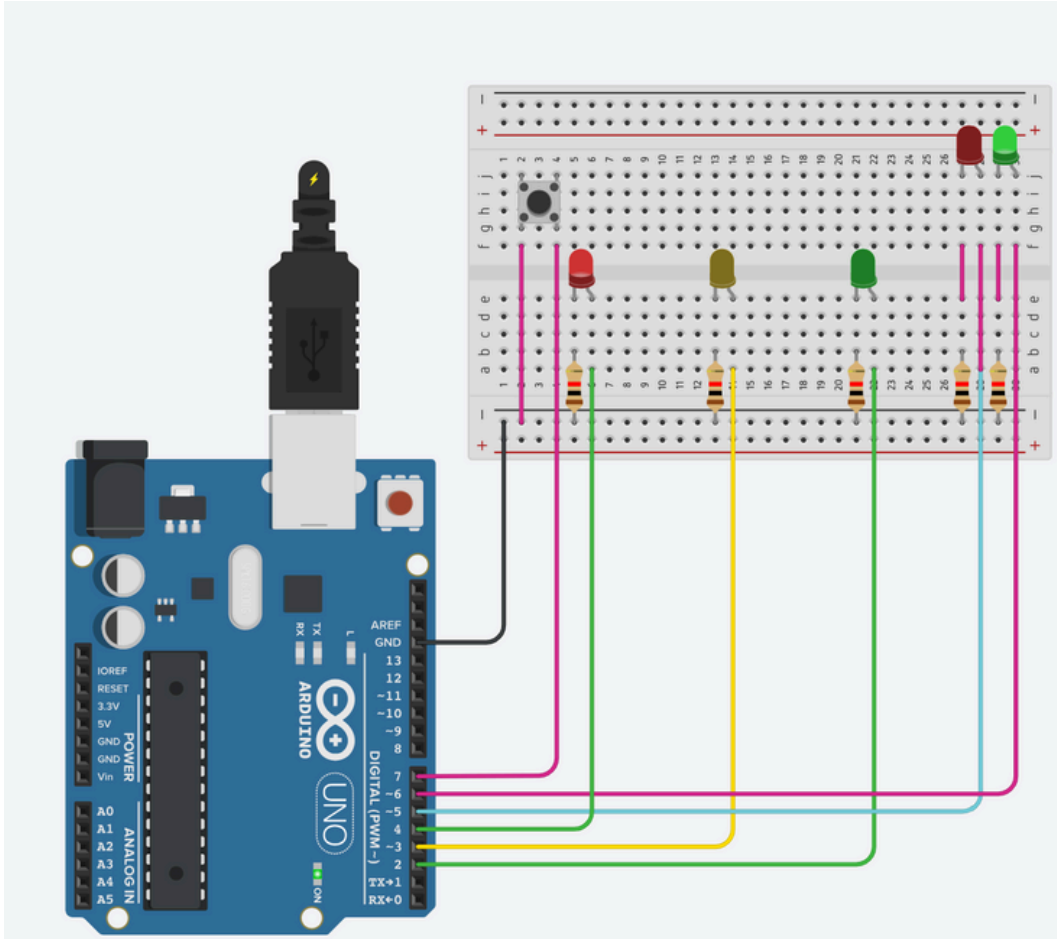
- The attachInterrupt() function links the button pin to an Interrupt Service Routine (ISR).
- The ISR is triggered automatically when the button is pressed (FALLING edge)

Characteristics

- Efficient: The CPU only reacts when the button is pressed, freeing resources for other tasks.
- Fast Response: Button presses are detected immediately, regardless of what the loop is doing.
- More Complex: Requires managing volatile variables and keeping the ISR short.

Use Case

Ideal for systems that need real-time response or must perform multiple tasks concurrently..



1 (Arduino Uno R3)

```ino
const int greenLight = 2;
const int yellowLight = 3;
const int redLight = 4;
const int pedRed = 5;
const int pedGreen = 6;
const int button = 7;

enum TrafficState { GREEN, WAIT, YELLOW, RED };
TrafficState currentState = GREEN;

unsigned long stateTimer = 0;
volatile bool buttonPressed = false;

void setup() {
  pinMode(greenLight, OUTPUT);
  pinMode(yellowLight, OUTPUT);
  pinMode(redLight, OUTPUT);
  pinMode(pedRed, OUTPUT);
  pinMode(pedGreen, OUTPUT);
  pinMode(button, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(button), buttonISR, FALL

  digitalWrite(greenLight, HIGH);
  digitalWrite(pedRed, HIGH);
  stateTimer = millis();
}

void loop() {
  if (buttonPressed) {
    buttonPressed = false;
    if (currentState == GREEN) {
      stateTimer = millis();
      currentState = WAIT;
    }
  }

  switch (currentState) {
    case GREEN:
      if (millis() - stateTimer >= 10000) {
        stateTimer = millis();
        currentState = WAIT;
      }
      break;

    case WAIT:
      if (millis() - stateTimer >= 3000) {
        stateTimer = millis();
        currentState = YELLOW;
      }
      break;

    case YELLOW:
      if (millis() - stateTimer >= 2000) {
        stateTimer = millis();
        currentState = RED;
      }
      digitalWrite(greenLight, LOW);
      digitalWrite(yellowLight, HIGH);
      break;

    case RED:
      if (millis() - stateTimer >= 10000) {
        stateTimer = millis();
        currentState = GREEN;
      }
      digitalWrite(yellowLight, LOW);
```

Serial Monitor