Programmed Control with Arduino

Activity 2: Alternating Blinking (Variables and Sequential Control)

In the previous activity, we learned how to keep a single LED continuously turned on[cite: 31]. In this session, we will take a step further by introducing a second LED to create an alternating blinking effect (like railroad crossing lights or an emergency vehicle)[cite: 32]. To achieve clean and scalable code, you will learn how to declare and initialize multiple independent variables[cite: 33].

Learning Objectives

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

- Declare and utilize multiple integer (`int`) variables in your code[cite: 34].
- Design logical and orderly timing sequences using the `delay` instruction[cite: 35].
- Understand the physical behavior of independent outputs connected in parallel[cite: 36].
- Optimize the maintenance of your program by using descriptive variable names[cite: 37].

Required Components

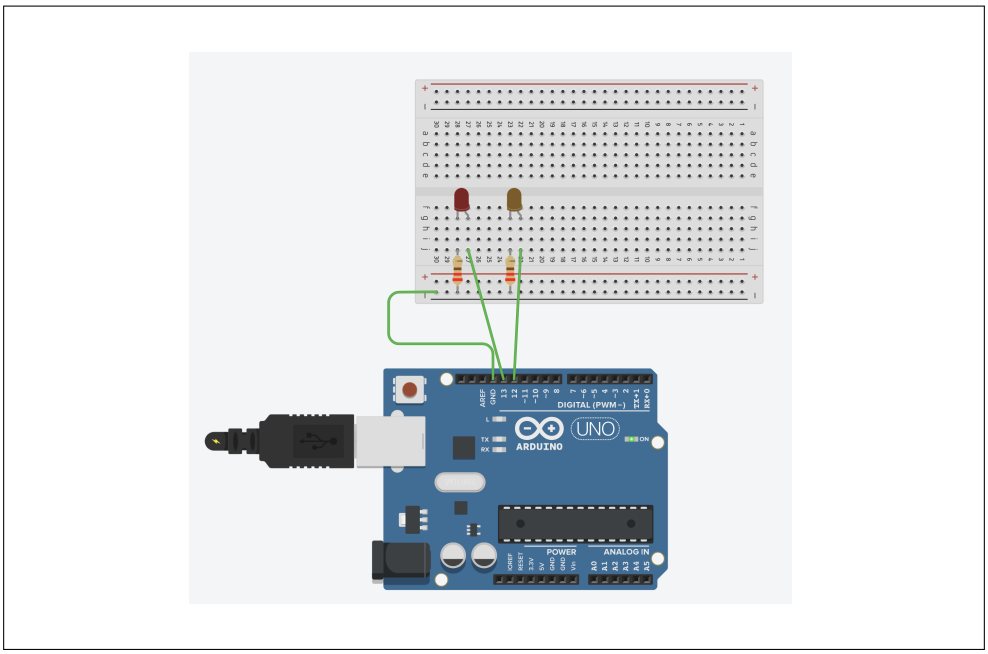
Find and place the following components on your TinkerCad workspace:

- 1 Arduino Uno board[cite: 38].
- 1 Small breadboard[cite: 39].
- 2 LEDs (we recommend using two different colors, e.g., Green and Red)[cite: 39].
- 2 $220\,\Omega$ resistors (one to protect each LED)[cite: 40].
- Virtual jumper wires[cite: 40].

Breadboard Assembly Diagram

We will add a second LED to the circuit you assembled in Activity 1. Make sure to connect each anode to an independent digital control pin[cite: 41].

Initial configuration of the Terminal object



Base Code: Variable Declaration and Alternation

Open the code editor in TinkerCad in text mode[cite: 42]. Clear the previous program and write the following sequential code[cite: 43]. Analyze how the state of each digital output is synchronously coordinated over time[cite: 44].

```
// 1. VARIABLE DECLARATION
// We reserve memory space to store the LED pins
int redLed = 13;
int greenLed = 12;

void setup() {
  // We configure both pins as current outputs
  pinMode(redLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
}

void loop() {
  // STATE 1: Red on and Green off
  digitalWrite(redLed, HIGH);
  digitalWrite(greenLed, LOW);
  delay(1000); // 1-second pause in this state

  // STATE 2: Red off and Green on
  digitalWrite(redLed, LOW);
  digitalWrite(greenLed, HIGH);
  delay(1000); // 1-second pause in this state
}
```

Listing 1: Code to control the alternating blinking of two LEDs.

How does the code work?

In this program, the Arduino executes instructions in a strictly linear fashion within the `loop` function:

1. **Global variables**: We declare `int` and `int` at the beginning[cite: 49]. This allows us to use clear, readable words instead of writing fixed (hardcoded) numbers throughout the code[cite: 50].
2. **Synchronous alternation**: To make one LED turn off while the other turns on, we send opposing signals simultaneously[cite: 51]. In State 1, we apply `HIGH` to the red and `LOW` to the green[cite: 52]. After a one-second pause, we invert the values in State 2.
3. **Continuous loop**: Upon finishing State 2, execution immediately returns to the beginning of `loop`, turning the red back on and the green off, thus creating an infinite loop of alternation[cite: 53].

Importance of logical order

If we did not explicitly turn off the opposing LED when changing states (i.e., by omitting the `LOW` instructions), the program would simply turn both LEDs on sequentially and leave them lit indefinitely, losing the alternating blinking effect[cite: 54].

Activity 2 Challenge

Verify that you understand the timing flow and the use of variables by solving these three proposals in your simulator:

1. **Police Effect (Stroboscopic Flashing)**: Modify the delay times (`delay`) to achieve a very fast alternation that emulates the emergency lights of a patrol car (e.g., a 150 ms pause)[cite: 55].
2. **The power of variables**: Change the wire of the Green LED to **digital pin 8** in TinkerCad[cite: 56]. Make a single change in your code (taking advantage of variables) so that your program continues to work perfectly without altering `setup` or `loop` [cite: 57].
3. **Full Off Phase**: Modify the program so that, before each light change, there is a brief intermediate phase where both LEDs remain turned off at the same time for half a second (500 ms)[cite: 58].