Programmed Control with Arduino

Activity 5: The Basic Traffic Light (Advanced Sequential Control)

In previous activities, we learned how to interact with buttons and log logical states. Today, we will apply a classical engineering approach to solve a system coordinated sequentially over time: a road traffic light for vehicles. We will connect three LEDs (Red, Yellow, and Green) and program their timing behavior to simulate the real cycle found on our city streets.

Learning Objectives

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

- Coordinate multiple digital output channels synchronously and sequentially.
- Structure differentiated delay times based on the priority or danger of the physical state being controlled.
- Organize a sequential algorithm in a closed-loop system, avoiding undefined states or dangerous overlapping signals.
- Practice the multiple declaration of hardware pin assignment variables.

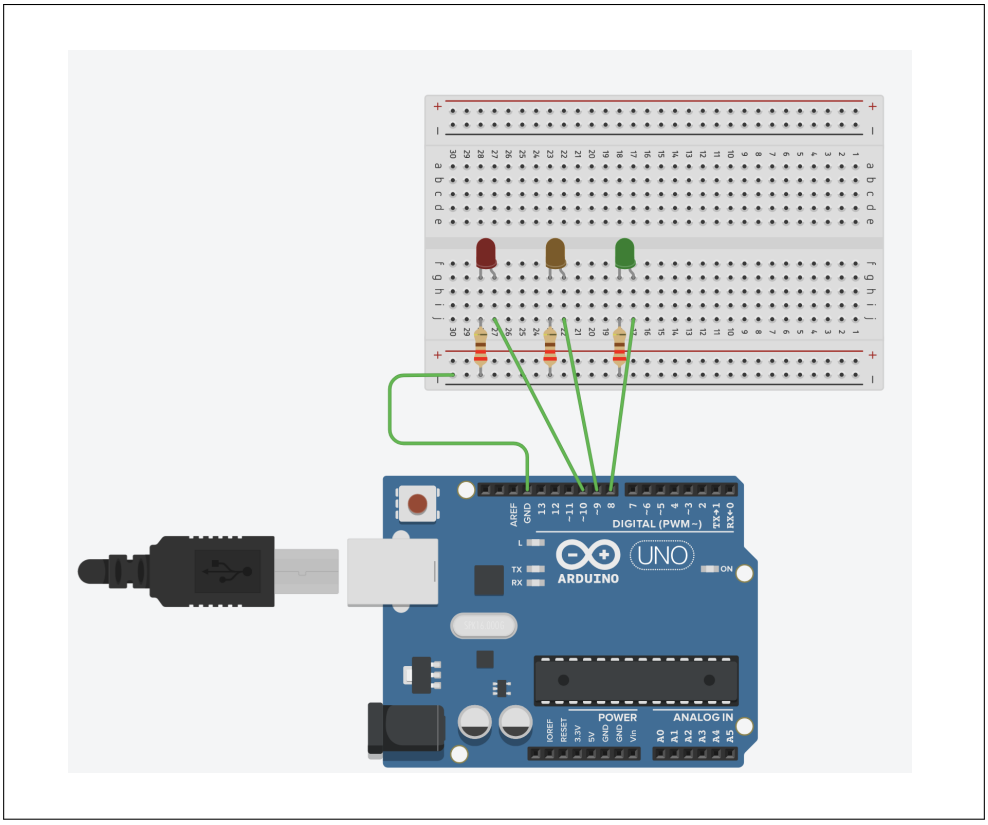
Required Components

Find and place the following components on your TinkerCad workspace:

- 1 Arduino Uno board.
- 1 Small breadboard.
- 3 LEDs (mandatory colors: 1 Red, 1 Yellow, and 1 Green).
- 3 $220\,\Omega$ resistors (one for each LED).
- Virtual jumper wires.

Breadboard Assembly Diagram

To control the traffic light for vehicles, we will associate each color with a sequential digital pin on the Arduino.



Base Code: Strict Timing Sequence

Write the following program in your TinkerCad text editor. Pay close attention to how, in each state of the traffic light, we explicitly turn off the inactive LEDs to prevent accidental simultaneous activation.

```
// Physical pin declaration for traffic light LEDs
int redLed = 10;
int yellowLed = 9;
int greenLed = 8;

void setup() {
  % Configure the three pins as output channels
  pinMode(redLed, OUTPUT);
  pinMode(yellowLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
}

void loop() {
  // STATE 1: Green active (Vehicles allowed to pass)
  digitalWrite(greenLed, HIGH);
  digitalWrite(yellowLed, LOW);
  digitalWrite(redLed, LOW);
  delay(5000); // Green stays active for 5 seconds

  // STATE 2: Yellow active (Warning of imminent stop)
  digitalWrite(greenLed, LOW);
  digitalWrite(yellowLed, HIGH);
  digitalWrite(redLed, LOW);
  delay(2000); // Yellow lasts for only 2 seconds

  // STATE 3: Red active (Vehicles stopped)
  digitalWrite(greenLed, LOW);
  digitalWrite(yellowLed, LOW);
  digitalWrite(redLed, HIGH);
  delay(5000); // Red stays active for 5 seconds
}
```

Listing 1: Code for vehicle traffic light timing.

How does the code work?

The flow of this program is purely sequential and linear. Being inside the `loop` function, it repeats endlessly in this exact order:

1. Green turns on and the others turn off $\rightarrow 5$-second delay.
2. Green turns off, Yellow turns on, and Red turns off $\rightarrow 2$-second delay.
3. The previous two turn off, Red turns on $\rightarrow 5$-second delay.
4. The cycle ends and immediately starts over at step 1 (Green).

Electrical and Logical Safety

In real-world industrial systems, a software bug must never allow contradictory lights (such as Green and Red) to be lit simultaneously. For this reason, at every phase change, we explicitly turn off the inactive outputs using `digitalWrite(pin, LOW)` instructions.

Activity 5 Challenge

Improve the behavior of the traffic light by applying these logical variations to your code:

1. **Safety Transition Phase**: In many countries, before transitioning from Red to Green, both the Red and Yellow lights turn on simultaneously for one second to signal drivers to ready their engines. Modify your code to introduce this intermediate phase between State 3 (Red) and State 1 (Green).
2. **Night Mode (Flashing Yellow)**: Modify the code to simulate the behavior of a traffic light at midnight: the Green and Red lights must remain completely inactive, while the Yellow light continuously turns on and off every 500 ms.
3. *Question for reflection*: If you wanted to make the traffic light interactive so a pedestrian could change the light cycle by pressing a button, what problems would using the `delay(5000)` instruction pose? Is the Arduino capable of detecting a button press while executing that time pause?