Real-Time LED Control System Using Simulator

Introduction

The Real-Time LED Control System project is designed to simulate a simple yet effective embedded system that controls LEDs using an Arduino Uno. This project uses the Tinkercad Circuits simulator to create a virtual environment where LEDs are controlled by push buttons. The primary objectives include simulating hardware components, handling interrupts, and managing LED blink rates and colors.

Objectives

- Simulate an Arduino Uno environment using Tinkercad Circuits.
- Implement control of two LEDs of different colors.
- Use two push buttons to control the blink rate and color of the LEDs.
- Demonstrate effective handling of interrupts.
- Implement the functionalities without using RTOS due to simulator constraints.

Materials and Tools

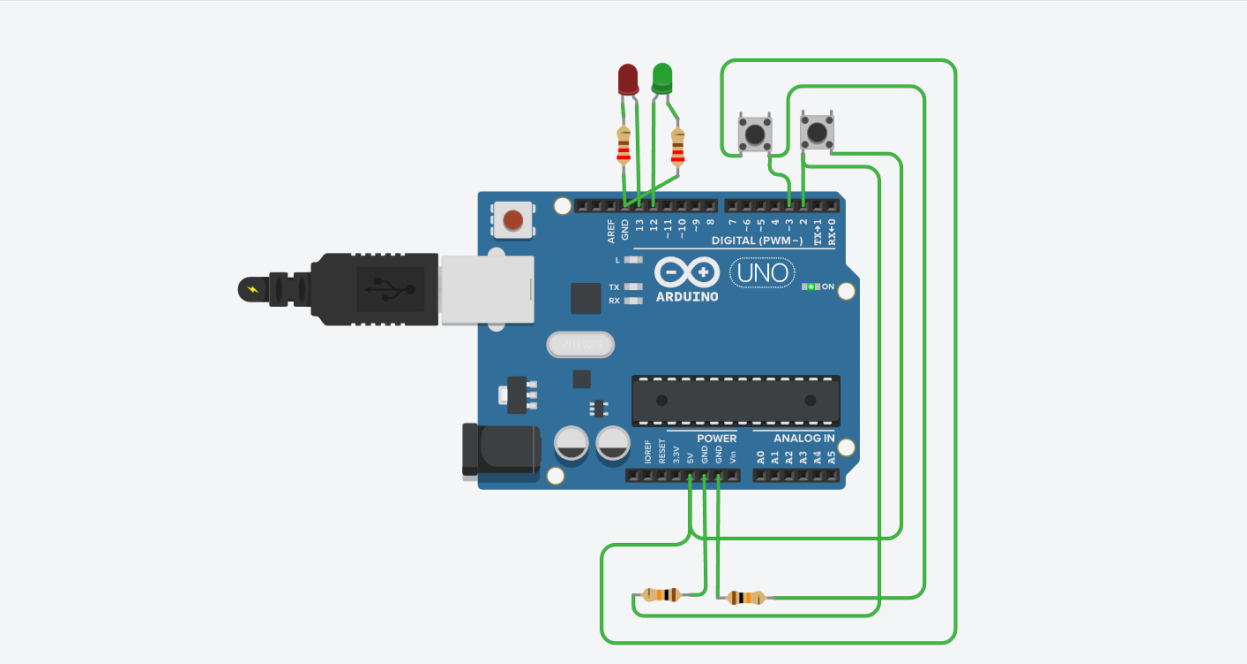
- Tinkercad Circuits Simulator
- Arduino IDE
- Virtual Components:
    - Arduino Uno
    - 2 LEDs (Red and Green)
    - 2 Push Buttons
    - 220-ohm Resistors
    - 10k-ohm Resistors

Circuit Diagram

Connections

- **Red LED (ledPin1)**:
    - Anode (long leg) to digital pin 13 on the Arduino.
    - Cathode (short leg) to one terminal of a 220-ohm resistor.
    - Other terminal of the 220-ohm resistor to GND.
- **Green LED (ledPin2)**:
    - Anode (long leg) to digital pin 12 on the Arduino.
    - Cathode (short leg) to one terminal of a 220-ohm resistor.
    - Other terminal of the 220-ohm resistor to GND.

- **Button 1 (buttonPin1)**:
  - One terminal to 5V.
  - Other terminal to digital pin 2 on the Arduino.
  - Same terminal to GND through a 10k-ohm pull-down resistor.
- **Button 2 (buttonPin2)**:
  - One terminal to 5V.
  - Other terminal to digital pin 3 on the Arduino.
  - Same terminal to GND through a 10k-ohm pull-down resistor.



Software Implementation

Code

```
// Pin definitions
const int ledPin1 = 13; // Red LED
const int ledPin2 = 12; // Green LED
const int buttonPin1 = 2;
const int buttonPin2 = 3;

// Variables
int blinkRate = 500; // Default blink rate in milliseconds
volatile bool colorToggle = false;
unsigned long previousMillis = 0;
int currentLedPin = ledPin1;

void setup() {
  // Initialize pins
```

```
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(buttonPin1, INPUT_PULLUP);
  pinMode(buttonPin2, INPUT_PULLUP);

  // Attach interrupts
  attachInterrupt(digitalPinToInterrupt(buttonPin1), changeBlinkRate,
FALLING);
  attachInterrupt(digitalPinToInterrupt(buttonPin2), changeColor, FALLING);
}

void loop() {
  unsigned long currentMillis = millis();

  // Blink LED at the current rate
  if (currentMillis - previousMillis >= blinkRate) {
    previousMillis = currentMillis;
    digitalWrite(currentLedPin, !digitalRead(currentLedPin));
  }
}

void changeBlinkRate() {
  blinkRate = (blinkRate == 500) ? 250 : 500; // Toggle between 1 Hz and 2 Hz
}

void changeColor() {
  // Toggle LED pin
  currentLedPin = (currentLedPin == ledPin1) ? ledPin2 : ledPin1;
}
```

1 (Arduino Uno R3)

```
// Pin definitions
const int ledPin1 = 13; // Red LED
const int ledPin2 = 12; // Green LED
const int buttonPin1 = 2;
const int buttonPin2 = 3;

// Variables
int blinkRate = 500; // Default blink rate in milliseconds
volatile bool colorToggle = false;
unsigned long previousMillis = 0;
int currentLedPin = ledPin1;

void setup() {
  // Initialize pins
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(buttonPin1, INPUT_PULLUP);
  pinMode(buttonPin2, INPUT_PULLUP);

  // Attach interrupts
  attachInterrupt(digitalPinToInterrupt(buttonPin1), changeBlinkR
  attachInterrupt(digitalPinToInterrupt(buttonPin2), changeColor,
}

void loop() {
  unsigned long currentMillis = millis();

  // Blink LED at the current rate
  if (currentMillis - previousMillis >= blinkRate) {
    previousMillis = currentMillis;
    digitalWrite(currentLedPin, !digitalRead(currentLedPin));
  }
}

void changeBlinkRate() {
  blinkRate = (blinkRate == 500) ? 250 : 500; // Toggle between 1
}

void changeColor() {
  // Toggle LED pin
  currentLedPin = (currentLedPin == ledPin1) ? ledPin2 : ledPin1;
}

```

Explanation

- **Setup Function**: Initializes the pins for LEDs and buttons and attaches interrupt service routines (ISRs) for the buttons.
- **Loop Function**: Manages the blinking of the LEDs based on the current blink rate.
- **ISR for Button 1**: Changes the blink rate between 500 ms (1 Hz) and 250 ms (2 Hz) when Button 1 is pressed.
- **ISR for Button 2**: Toggles the LED color between red and green when Button 2 is pressed.

Results

Upon running the simulation in Tinkercad:

- The red LED blinks at a default rate of 1 Hz.
- Pressing Button 1 changes the blink rate to 2 Hz and back to 1 Hz on subsequent presses.
- Pressing Button 2 switches the blinking LED from red to green and vice versa.

Conclusion

This project successfully simulates a Real-Time LED Control System using the Tinkercad Circuits simulator. The use of interrupts for handling button presses ensures real-time response to user inputs. Although RTOS was not used due to simulator constraints, the project demonstrates key concepts of real-time systems and interrupt handling in an embedded environment.

Future Work

For future enhancements:

- Implement RTOS in a hardware setup to manage tasks more efficiently.
- Add more LEDs and buttons to expand functionality.
- Integrate more complex user inputs and outputs to create a more sophisticated control system.