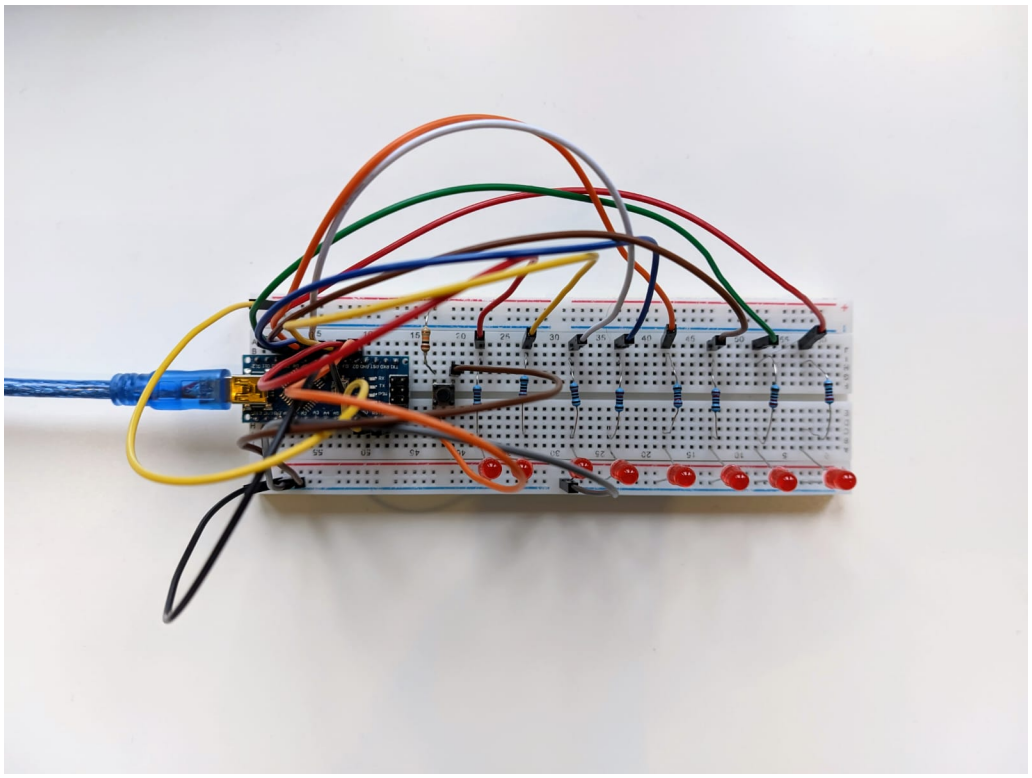
IRO 1 - Microcontroller Lab Report

Lab 4, 8-bit Binary LED Counter



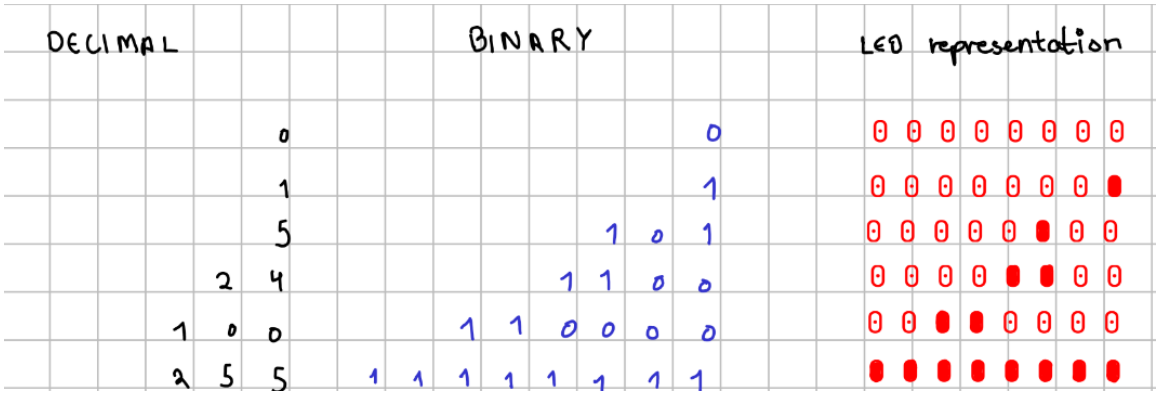
Lab 4, 8-bit Binary LED Counter

Name: Ishaan Bhimwal
Matriculation Number: 4223008
University of Applied Sciences Würzburg-Schweinfurt
Date: 31.10.2023

1. Introduction:

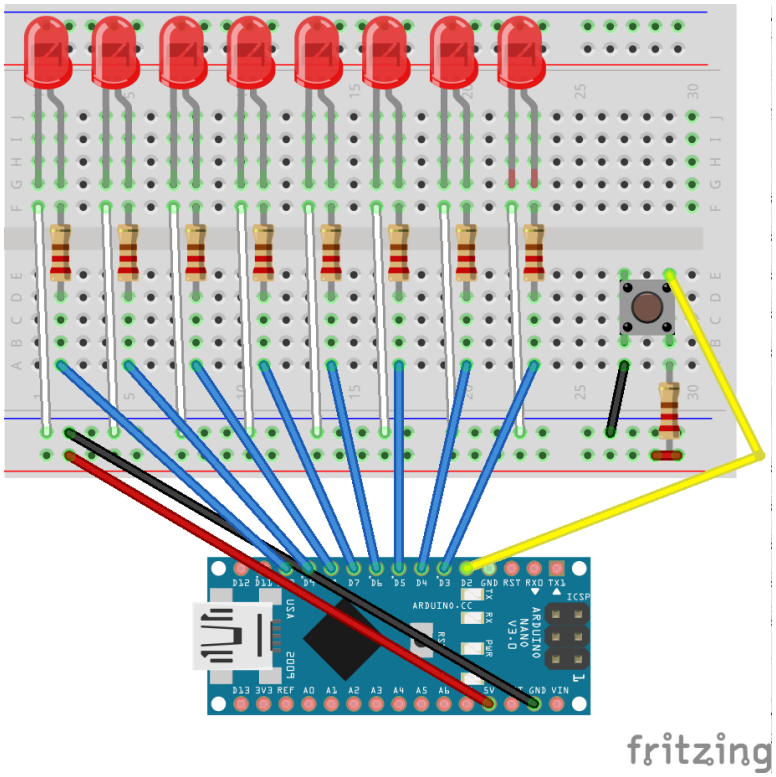
This project aims to visualize binary counting using LEDs, wherein each LED represents a binary bit. The project utilizes simple and easily available components like LEDs, 220 Ohms resistors, 10 kOhms resistor, push button and an Arduino Nano.

The initial state starts from 0 and goes till 255 each time the push-button is pressed. For example, the number 0 is `00000000` in binary so no LEDs must be ON, number 1 is `00000001` so only the rightmost led will be on, number 2 is `00000010` so only the second led from the right will be on and so on.

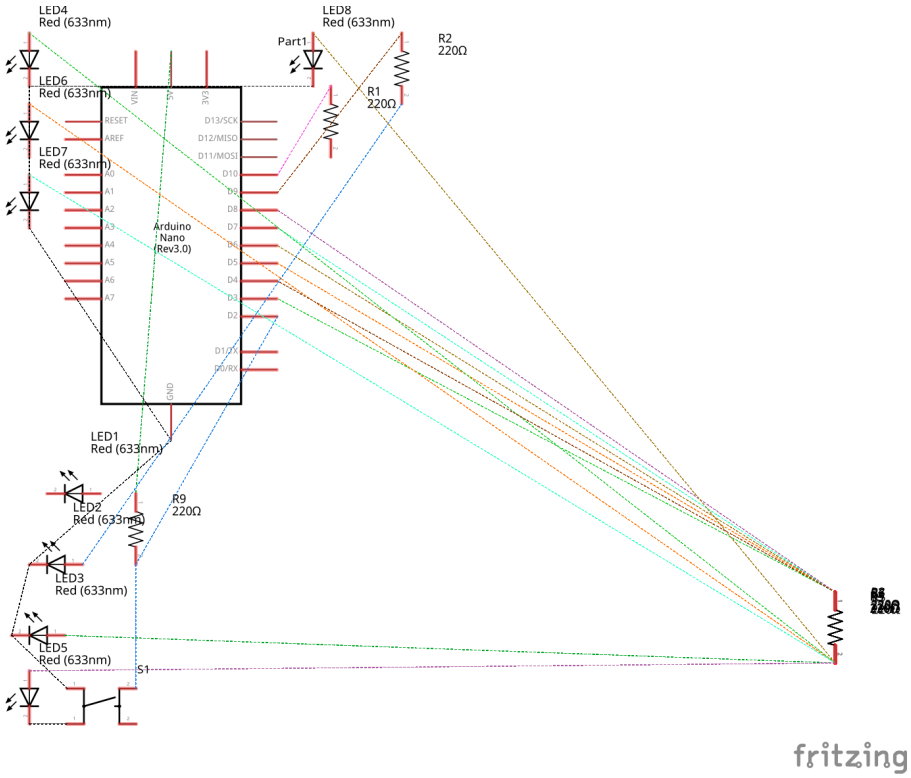


The representation cannot go beyond decimal number 255 or `11111111` as overflow will occur. In this case we don't have an extra LED to represent the ninth bit (in case of 256 or, `100000000`) hence the answer we'll get is simply wrong.

2. Circuit diagram:



Breadboard layout. Made in Fritzing.



Schematic. Made in Fritzing.

3. Program code:

task_4

```
// Pin configuration
const int buttonPin = 2;  // Pin for the push button
const int ledPins[] = {3, 4, 5, 6, 7, 8, 9, 10}; // Pins for the 8 LEDs

int counter = 0;  // Counter variable to store the current count
int lastButtonState = LOW;  // Variable to store the previous state of the button

void setup() {
  for (int i = 0; i < 8; i++) {
    pinMode(ledPins[i], OUTPUT); // Initialize LEDs as outputs
    digitalWrite(ledPins[i], LOW);  // Initially turn off all LEDs
  }

  // Initialize push button pin
  pinMode(buttonPin, INPUT);

  // Set the initial state of the LEDs
  updateLEDs();
}

void loop() {
  // Read the state of the button
  int buttonState = digitalRead(buttonPin);

  // Check for a button press
  if (buttonState == HIGH && lastButtonState == LOW) {
    // Increment the counter on button press
    counter = (counter + 1) % 256;
    updateLEDs();
  }

  // Update the previous button state
  lastButtonState = buttonState;
}

void updateLEDs() {
  // Update the LEDs based on the binary representation of the counter
  for (int i = 0; i < 8; i++) {
    digitalWrite(ledPins[i], (counter & (1 << i)) ? HIGH : LOW);
  }
}
```

4. Description of the program code:

- First, we define the pins for the push button and the 8 LEDs. The `counter` variable keeps track of the current count value, and `lastButtonState` is used to store the previous state of the push button.

- `setup()` Function: This function runs only once during the setup. It sets up the LEDs as outputs, initializes the push button pin with an internal pull-up resistor, and sets the initial state of the LEDs to off.

- `loop()` Function: This function runs repeatedly. It reads the state of the push button and, if a button press is detected, increments the counter (with wraparound at 255) and updates the LEDs accordingly.

- `updateLEDs()` Function: This function is responsible for updating the LEDs based on the binary representation of the counter. It uses bitwise operations to extract each bit of the counter and sets the corresponding LED accordingly.

5. Summary:

I had fun working on this project by myself. I am glad that I could use knowledge from my Computer Engineering and Programming classes and make a working setup here.