Atividade Prática 2

Luís Felipe Valença de Albuquerque

10 de dezembro de 2023

1 Instruções

O projeto proposto visa a criação de um sistema de iluminação natalina decorativa, utilizando o microcontrolador ATmega328p e linguagem de programação C++. O painel será desenvolvido para que atenda às especificações requisitadas, sendo: uma matriz LED 5x5 que proporcione diferentes combinações de exibições.

Ademais, pode-se definir com mais detalhes como seguem os seguintes itens:

- Usar LEDs com cores diferentes.
- Usar botões.
- Usar botões para fazer a alteração da configuração das imagens.
- Usar Arduino uno R3.
- **Observação**: Será analisada a organização da montagem, escolha das imagens que serão formadas no paiel, bem como o código de cores dos jumpers utilizados para representação do GND e do Vcc.

2 Desenvolvimento experimental

Neste trabalho foi-se utilizando o Tinkercad para a simulação do painel natalino requisitado. Nesse sentido, é possível elencar os materiais utilizados como segue na tabela:

Nome	Quantidade	Componente
D1 D4 D6 D8 D19	5	Azul LED
D2 D7 D12 D18 D20	5	Vermelho LED
D3 D10 D11 D15 D17	5	Amarelo LED
D5 D9 D13 D14 D16	5	Verde LED
U1	1	Arduino Uno R3
S1 S2	2	Botão
R1 R2	2	2 kΩ Resistor
R3 R4 R5 R6	4	1 Ω Resistor

Tabela 1: Especificação dos Materiais

3 Lista de Componentes Eletrônicos

- **LEDs:**

  D1-D20 5 de cada cor (Azul, Vermelho, Amarelo, Verde)

  Função: Dispositivos semicondutores que emitem luz quando uma corrente elétrica passa por eles.

  Aplicação: Indicadores visuais em circuitos eletrônicos.

- **Arduino Uno R3 (U1):**

  Quantidade: 1

  Função: Placa de desenvolvimento com microcontrolador ATmega328, programável para controle de dispositivos.

- **Botões (S1-S2):**

  Quantidade: 2

  Função: Interruptores manuais que podem ser pressionados para fechar ou abrir um circuito elétrico.

- **Resistores (R1-R6):**

  R1-R2 2 de 2 kΩ

  R3-R6 4 de 1 Ω

  Função: Limitam o fluxo de corrente em um circuito.

  Aplicação: R2 pode ser usado para limitar a corrente em certas partes do circuito, enquanto R6 pode ser usado para limitar corrente em circuitos de baixa resistência.

3.1 Execução Experimental

A execução experimental deste projeto foi realizada seguindo as diretrizes da atividade 2, que consiste na criação de um sistema de iluminação decorativa para o Natal. O objetivo principal era construir um painel de iluminação programável, controlado pelo microcontrolador Atmega328p por meio de um código C++. Sua arquitetura ficou da seguinte forma:

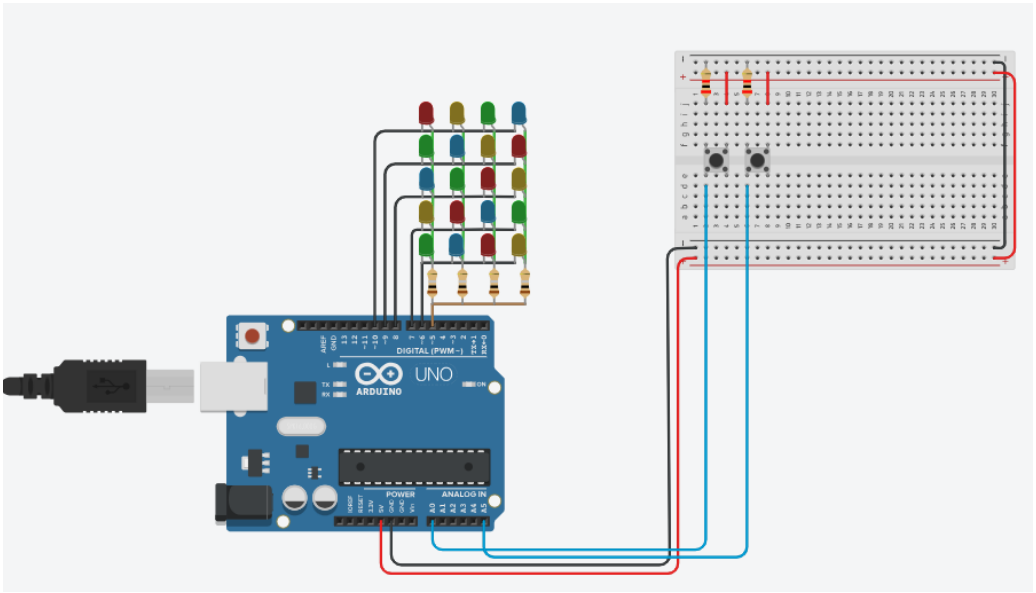


Figura 1: Topologia do Projeto

4 Explicação do Código

4.1 Definições e Constantes

O código começa com a definição de pinos para as linhas, colunas e botões. Parâmetros como o tempo de linha e o tempo do quadro são definidos, bem como constantes para as direções.

5 Explicação do Código

5.1 Definições e Constantes

O código começa com a definição de pinos para as linhas, colunas e botões. Parâmetros como o tempo de linha e o tempo do quadro são definidos, bem como constantes para as direções.

Listing 1: Definições e Constantes

```
// CONEXOES
#define LINHA1 6
#define LINHA2 7
#define LINHA3 8
#define LINHA4 10
#define LINHA5 9

#define COLUNA1 5
#define COLUNA2 4
#define COLUNA3 3
#define COLUNA4 2
#define COLUNA5 A4

#define LBTN A0
#define RBTN A5

// PARAMETROS
#define TEMPO_LINHA 1       // em milissegundos
#define TEMPO_QUADRO 16     // em milissegundos

// CONSTANTES
#define DIREITA 0
#define BAIXO   1
#define ESQUERDA  2
#define CIMA    3
```

5.2 Matriz e Variáveis

Uma matriz `tela` de 5x5 é declarada para representar o painel LED. Variáveis booleanas `ultimoEstadoLBTN` e `ultimoEstadoRBTN` são utilizadas para acompanhar o estado anterior dos botões.

Listing 2: Matriz e Variáveis

```
byte tela[5][5];  // Alterado para 5x5

// VARIAVEIS
bool ultimoEstadoLBTN = LOW;
bool ultimoEstadoRBTN = LOW;
```

5.3 Função `limparTela()`

A função `limparTela()` percorre a matriz e a zera, apagando todos os LEDs.

Listing 3: Função `limparTela()`

```
void limparTela() {
  for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 5; j++) {
      tela[i][j] = 0;
    }
  }
}
```

5.4 Função `exibirTela()`

A função `exibirTela()` é responsável por exibir a matriz no painel LED, utilizando ciclos para acender os LEDs de acordo com a matriz.

Listing 4: Função `exibirTela()`

```
void exibirTela(byte matriz[5][5]) {
  for (byte c = 0; c < 5; c++) {
    digitalWrite(colunas[c], HIGH);
    for (byte r = 0; r < 5; r++) {
      digitalWrite(linhas[r], matriz[r][c] ? 0 : 255);
      delay(5);
    }
    for (byte r = 0; r < 5; r++) {
      digitalWrite(linhas[r], HIGH);
      delay(5);
    }
    digitalWrite(colunas[c], LOW);
  }
}
```

5.5 Configuração Inicial (`setup()`)

No `setup()`, os pinos das linhas e colunas são configurados como saídas, e os pinos dos botões como entradas. O estado inicial da tela é inicializado chamando `limparTela()`.

Listing 5: Configuração Inicial (`setup()`)

```
void setup() {
  for (int i = 0; i < 5; i++) {
    pinMode(linhas[i], OUTPUT);
    digitalWrite(linhas[i], LOW);

    pinMode(colunas[i], OUTPUT);
    digitalWrite(colunas[i], HIGH);
  }
```

```
  pinMode(LBTN, INPUT);
  pinMode(RBTN, INPUT);

  // Inicializar o estado inicial da tela
  limparTela();
}
```

5.6 Loop Principal (`loop()`)

Dentro do loop principal, há uma lógica para reagir aos botões. Se um botão é pressionado, a matriz de LEDs é aleatoriamente acesa por um curto período. A função `exibirTela()` é chamada para mostrar a matriz no painel LED.

Listing 6: Loop Principal (`loop()`)

```
void loop() {
  // Logica para reagir aos botoes
  if (digitalRead(LBTN) == HIGH && !ultimoEstadoLBTN) {
    // Piscar LEDs aleatoriamente
    for (int i = 0; i < 10; i++) {
      tela[random(5)][random(5)] = 1;
    }
    exibirTela(tela);
    delay(200);  // Tempo de exibicao
    limparTela();
  }

  if (digitalRead(RBTN) == HIGH && !ultimoEstadoRBTN) {
    // Piscar LEDs aleatoriamente
    for (int i = 0; i < 10; i++) {
      tela[random(5)][random(5)] = 1;
    }
    exibirTela(tela);
    delay(200);  // Tempo de exibicao
    limparTela();
  }

  ultimoEstadoLBTN = digitalRead(LBTN);
  ultimoEstadoRBTN = digitalRead(RBTN);

  delay(TEMPO_QUADRO);
}
```

5.7 Delay e Atualização do Estado dos Botões

Um pequeno atraso (`delay(TEMPO_QUADRO)`) é introduzido para controlar a taxa de atualização. O estado dos botões é atualizado no final do loop para detectar pressionamentos.

6 Resultados e Considerações

Após o desenvolvimento do código, bem como a organização dos componentes de hardware, é possível inicializar a simulação desejada. As imagens a seguir apresentação as exibições feitas no painel:

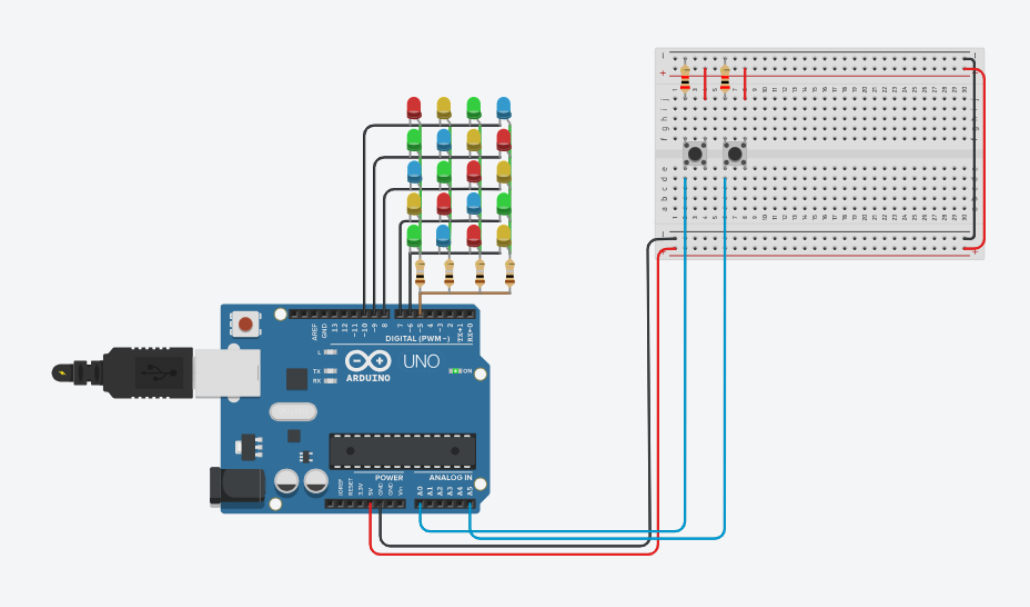


Figura 2: Primeira imagem após pressionar o botão

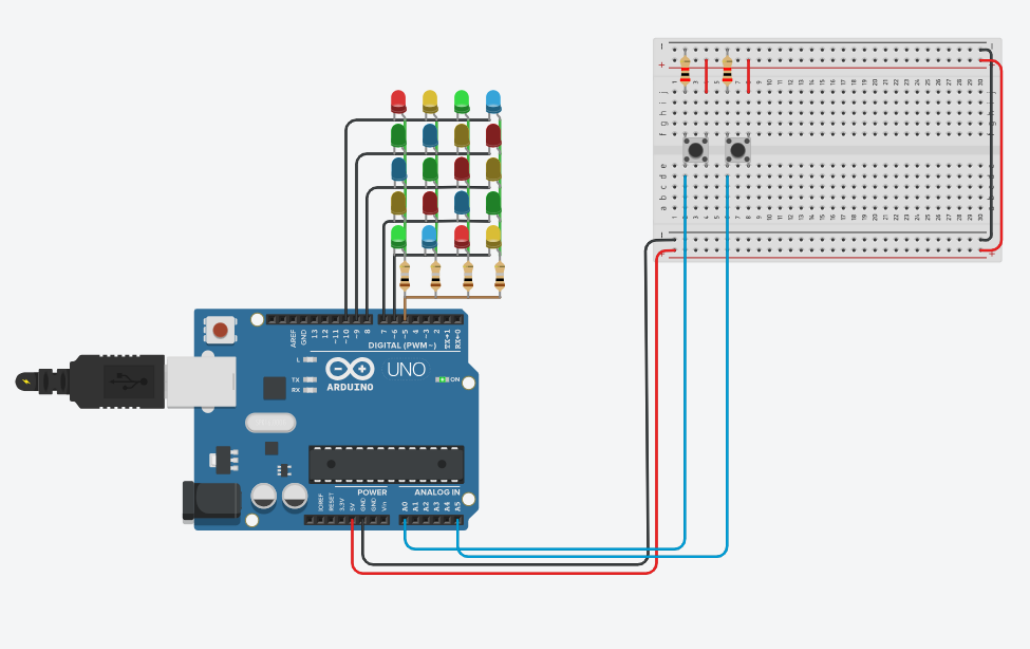


Figura 3: Segunda imagem após pressionar o botão

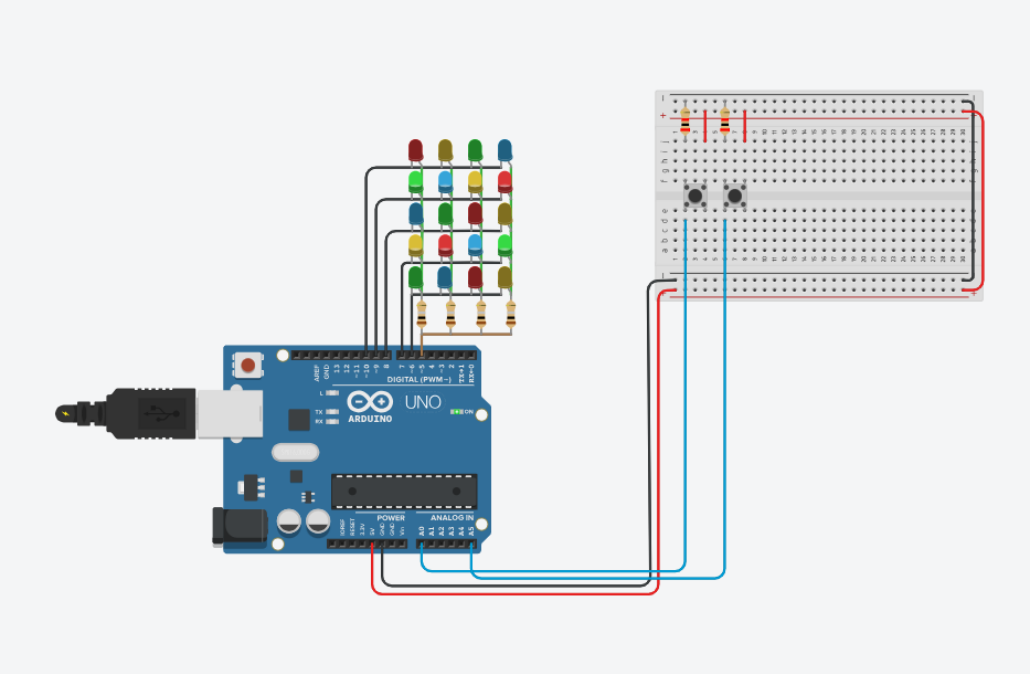


Figura 4: Terceira imagem após pressionar o botão

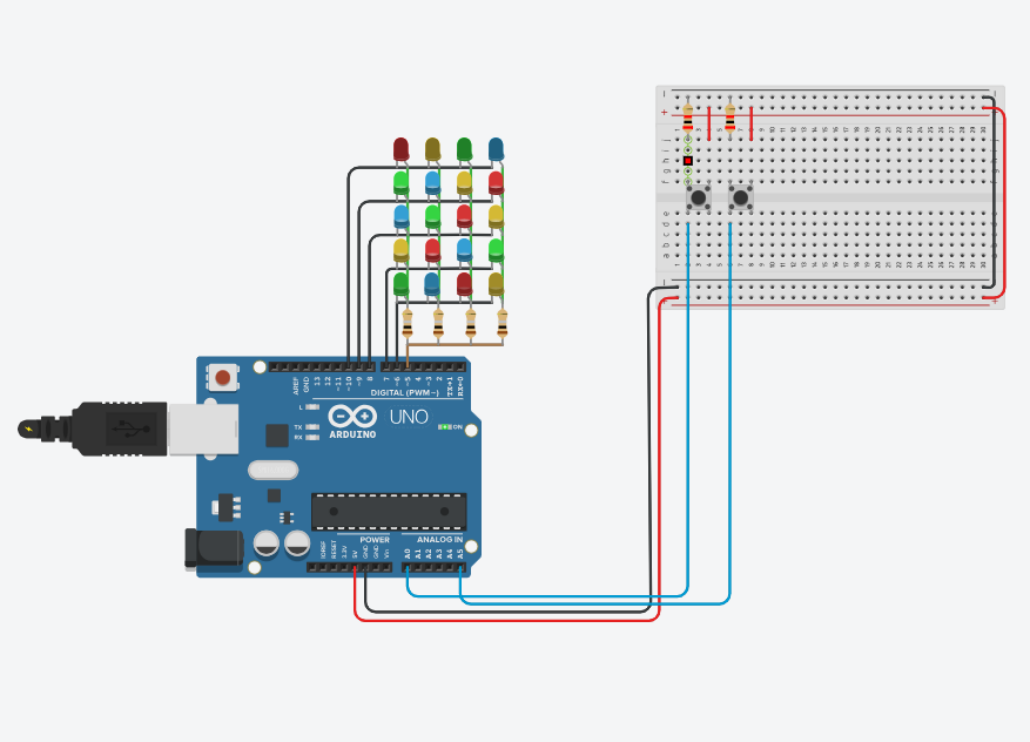
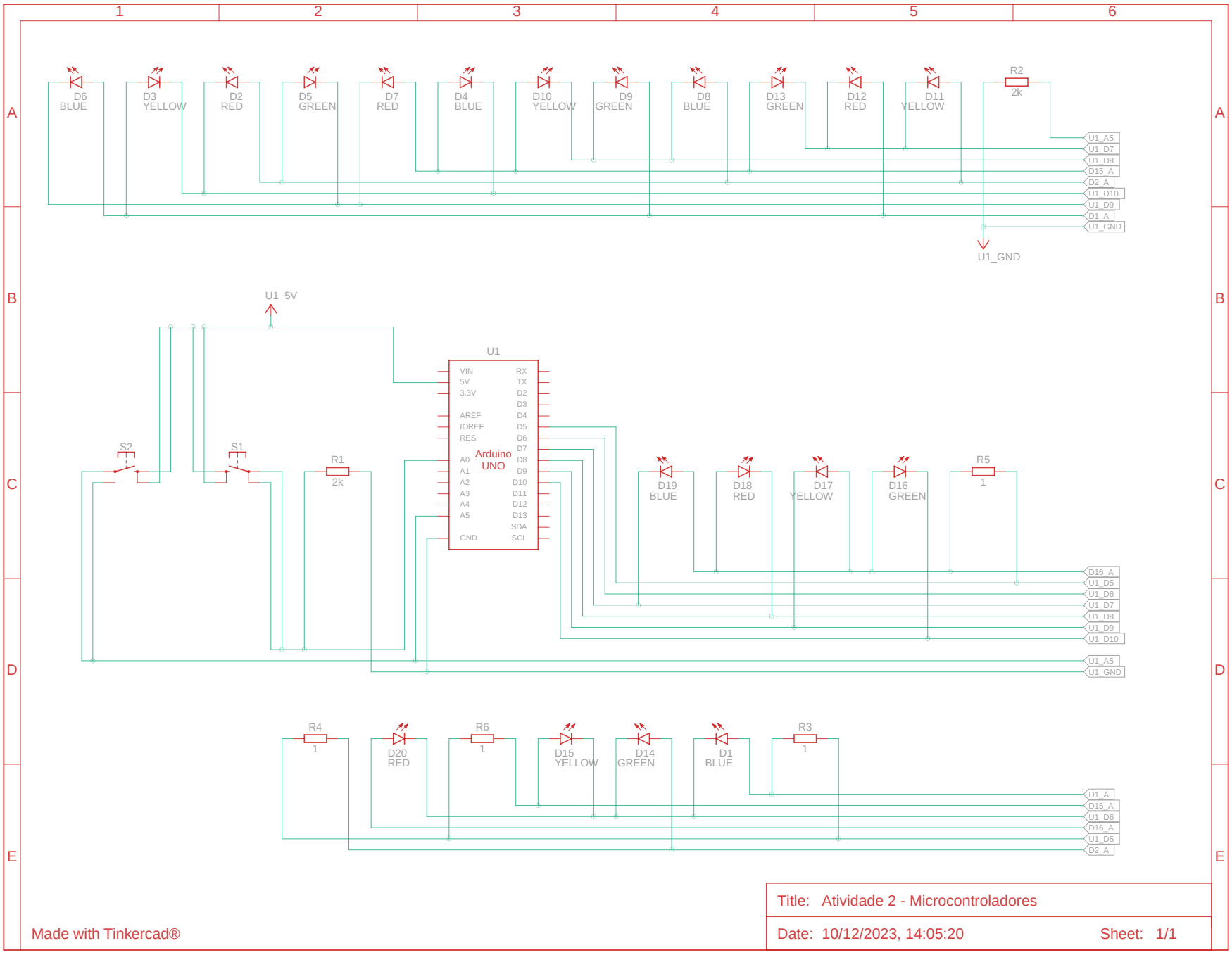


Figura 5: Quarta imagem após pressionar o botão

Link de acesso ao projeto: https://www.tinkercad.com/things/9BoG6OwEn2S-atividade-2-microcontroladores

E em anexo, segue sua representação esquemática.


1
2
3
4
5
6
A
B
C
D
E
D6
BLUE
D3
YELLOW
D2
RED
D5
GREEN
D7
RED
D4
BLUE
D10
YELLOW
D9
GREEN
D8
BLUE
D13
GREEN
D12
RED
D11
YELLOW
R2
2k
U1_A5
U1_D7
U1_D8
D15_A
D2_A
U1_D10
U1_D9
D1_A
U1_GND
U1_GND
U1_5V
U1
VIN
5V
3.3V
AREF
IOREF
RES
A0
A1
A2
A3
A4
A5
GND
RX
TX
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
SDA
SCL
Arduino
UNO
S2
S1
R1
2k
D19
BLUE
D18
RED
D17
YELLOW
D16
GREEN
R5
1
D16_A
U1_D5
U1_D6
U1_D7
U1_D8
U1_D9
U1_D10
U1_A5
U1_GND
R4
1
D20
RED
R6
1
D15
YELLOW
D14
GREEN
D1
BLUE
R3
1
D1_A
D15_A
U1_D6
D16_A
U1_D5
D2_A
Made with Tinkercad®
Title: Atividade 2 - Microcontroladores
Date: 10/12/2023, 14:05:20
Sheet: 1/1