

# Programação de Arduino



9 dezembro 2021

# O Que É o NEEC?

- » **Associação sem fins lucrativos.**
- » **Foi fundado a 19 de Setembro de 2003.**
- » **Constituído por estudantes de MEEC do IST**



# O Que Fazemos?

## As atividades incluem:

- Workshops
- Eventos
- Receção aos novos alunos
- Estágios de Verão (IST-SI)
- Contacto Empresarial
- Podcast (NEECTalks)

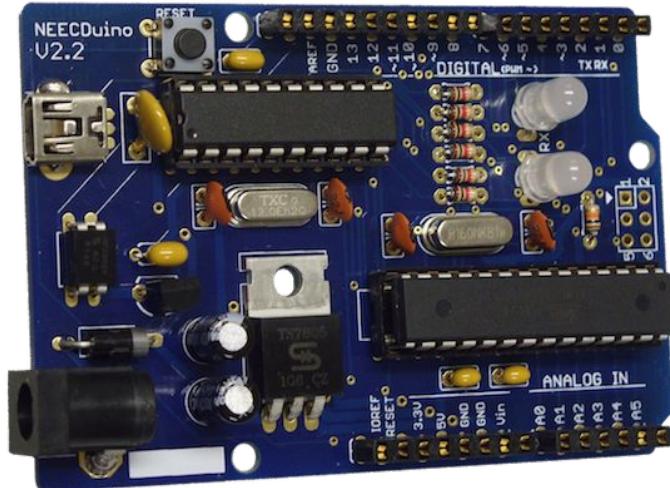
# Alguns dos Workshops



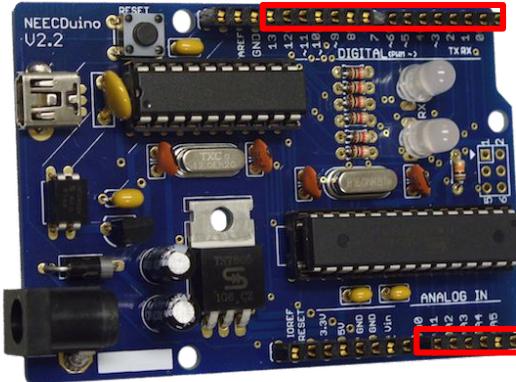
# O que é o Arduino (NEECduino)?

## Site oficial:

- » “Plataforma open-source de prototipagem eletrônica com hardware e software flexíveis e fáceis de usar, destinado a artistas, designers, hobbistas e qualquer pessoa interessada em criar objetos ou ambientes interativos.”



# NEECduino



## Digitais

- » Informação transmitida: 0 ou 1
- » Pin 0 até ao Pin 13

## Analógicos

- » Informação transmitida: 0 ao 255 (8-bit)
- » Pin A0 até ao Pin A5

## Alimentação e terra

## Memória

- » Flash 32k bytes (of which .5k is used for the bootloader)
- » SRAM 2k bytes
- » EEPROM 1k bytes

## Nota:

1 Byte = 8 bit  
1 kB = 1024 B

# Introdução à Programação(1)

Há vários tipos de variáveis. Exemplos:

- » int – guardam números inteiros; (2 Bytes)
- » float – guardam números reais; (4 Bytes)
- » char – guardam caracteres (letras e símbolos); (1 Byte)

```
int o_meu_inteiro = 1;
```

```
char my_first_char = 'a';
```

```
float this_is_uncommon = 212.4731;
```

# Introdução à Programação(2)

- » O ponto e vírgula (;) é utilizado para separar as instruções que constituem o programa.
- » Para definir o comportamento do programa consoante as ações do utilizador usamos condições **if** e condições **else** .

## Exemplo:

Caso a variável winner for **igual a 1**,  
o programa segue pelo código A;

Caso a variável winner for **igual a 2**,  
o programa segue pelo código B;

- Caso **nenhuma** das condições anteriores se realize,  
o programa segue pelo código C.



```
9  if(winner == 1) {  
10     ...A...  
11 }  
12  
13 else if(winner == 2) {  
14     ...B...  
15 }  
16 else {  
17     ...C...  
18 }  
19  
20  
21  
22  
23  
24  
25 }
```



# Introdução à Programação(3)

- » Um ciclo **while** faz com que enquanto a expressão for verdadeira o código dentro do bloco seja executado indefinidamente até que a expressão se verifique falsa.

```
3  while(life_p1 != 0 && life_p2 != 0) {  
4  
5  ...  
6  
7  }
```

- » Um ciclo **for** faz com que o bloco seja executado, enquanto a expressão for verdadeira, num número definido de iterações.

```
29  for(i = 0; i < 3; i++) {  
30  
31  ...  
32  
33  }
```

# Introdução à Programação(4)

- » **Um array** é uma maneira de armazenar vários dados numa mesma variável através do uso de índices numéricos.

```
35 int i;  
36 int led_p1[] = {12, 11, 10};  
37  
38 i = led_p1[1];
```

Qual o valor de i?

Qual o valor de led\_p1[2]?

# Introdução à Programação(5)

- » De modo a ter o código estruturado e organizado, é possível criar vários “blocos de código” chamados **Funções**. Estas podem ser “chamadas” dentro de outras, de modo a evitar a repetição excessiva de código.

## Exemplo:

O programa começa na função MainGame(),  
“seguindo” a parte de código A;

De modo a obter um valor para a variável winner,  
o programa “chama” a função Round\_Judge(),

```
46 int MainGame() {  
47     ... A ...  
48     winner = Round_Judge(rand_led);  
49     ... B ...  
50 }  
51  
52 int Round_Judge(int rand_led) {  
53     ... C ...  
54     return winner;  
55 }
```

# Introdução à Programação(6)

## » Estrutura de um programa em arduino (.ino)

setup()

loop()

function1()

function2()

etc...

```
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
    // set up the LCD's number of columns and rows:
    lcd.begin(16, 2);
    // Print a message to the LCD.
    lcd.print("hello, world!");
}

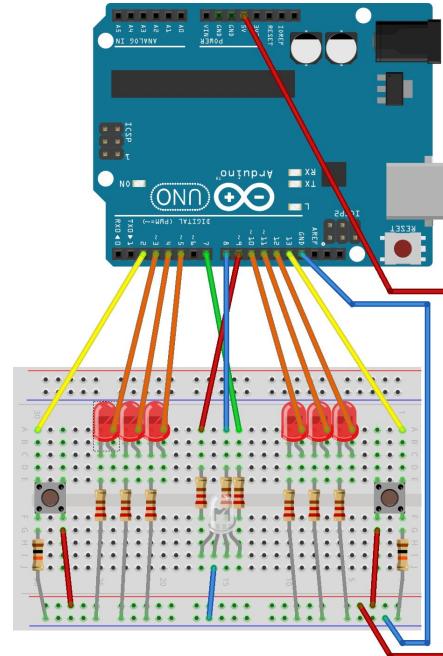
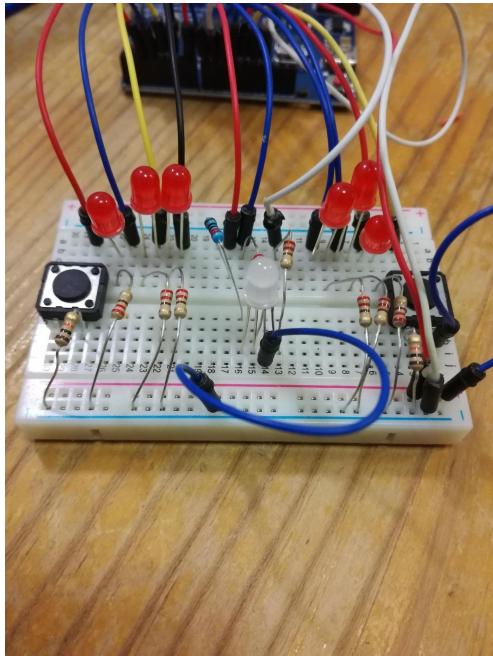
void loop() {
    // set the cursor to column 0, line 1
    // (note: line 1 is the second row, since counting begins with 0):
    lcd.setCursor(0, 1);
    // print the number of seconds since reset:
    lcd.print(millis() / 1000);
}
```

# Introdução à Programação(7)

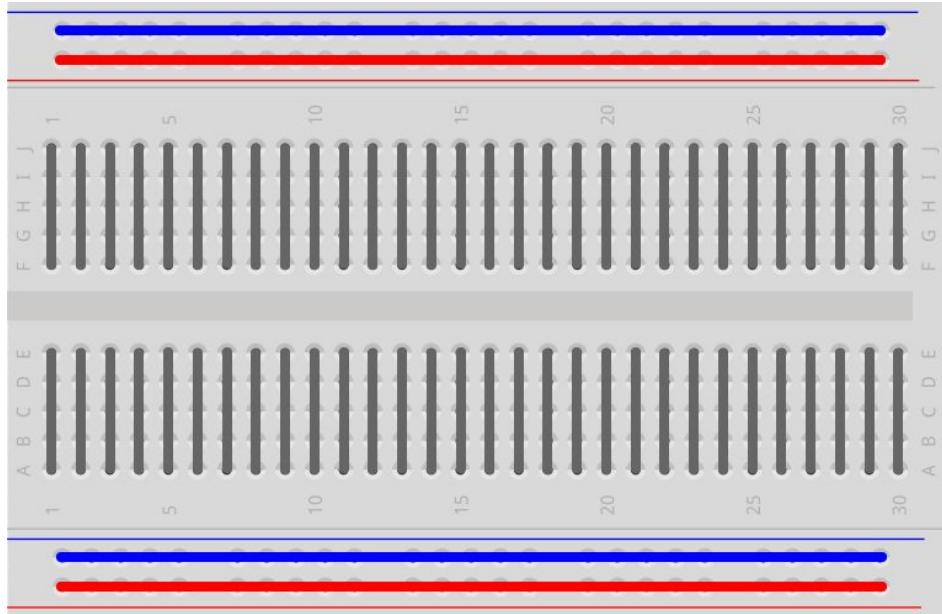
» **Funções** específicas do arduino que vamos usar:

```
void pinMode(pin, mode);
int digitalRead(pin);
void digitalWrite(pin, value);
int analogRead(pin);
void analogWrite(pin, value);
```

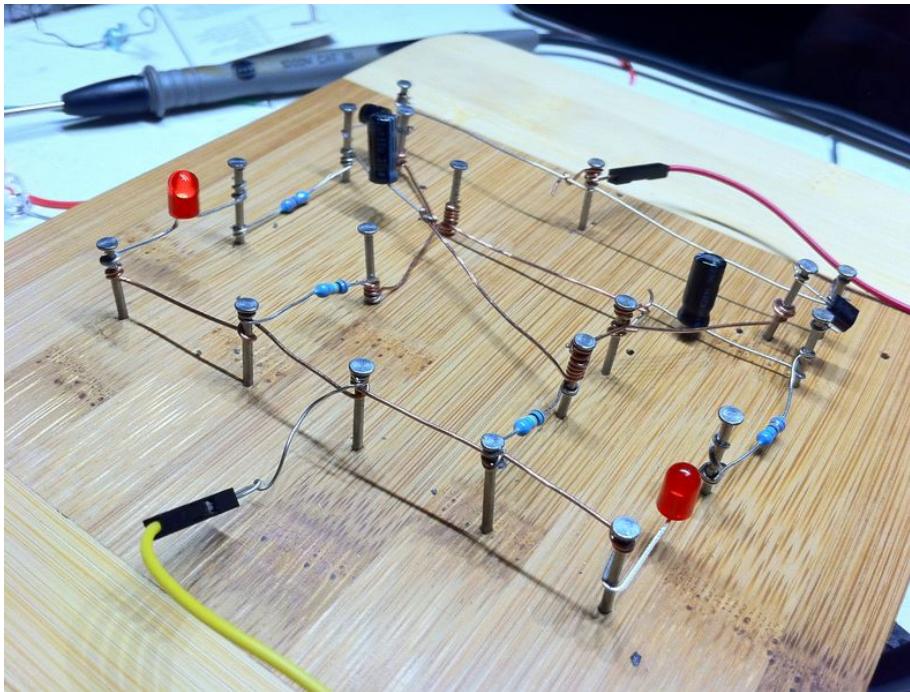
# NEECReact



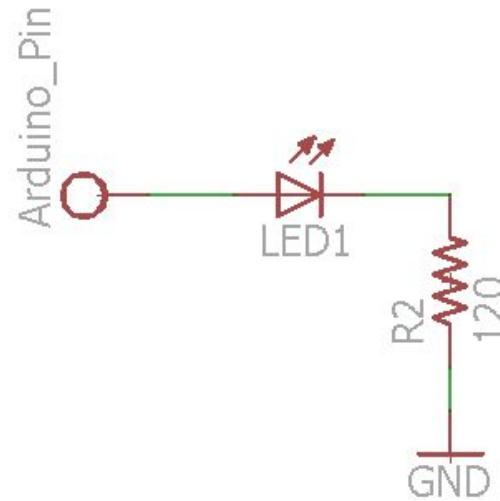
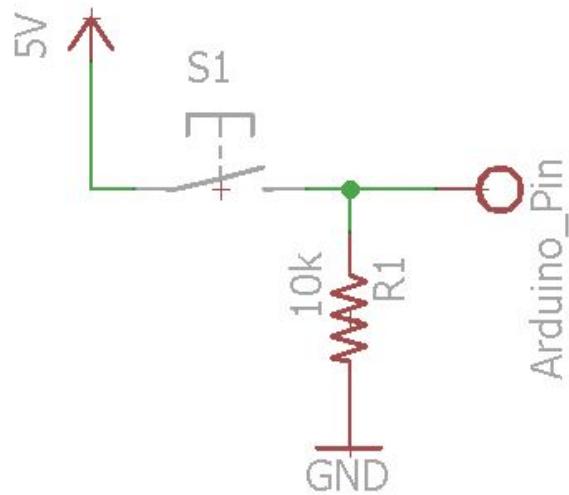
# Breadboard



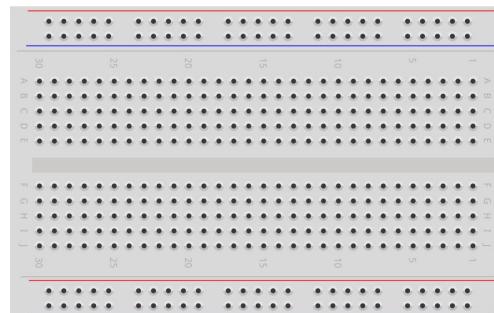
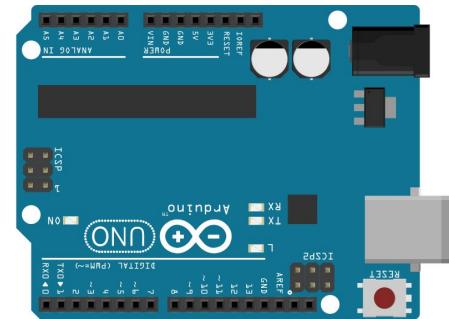
# Breadboard



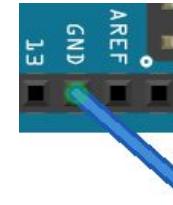
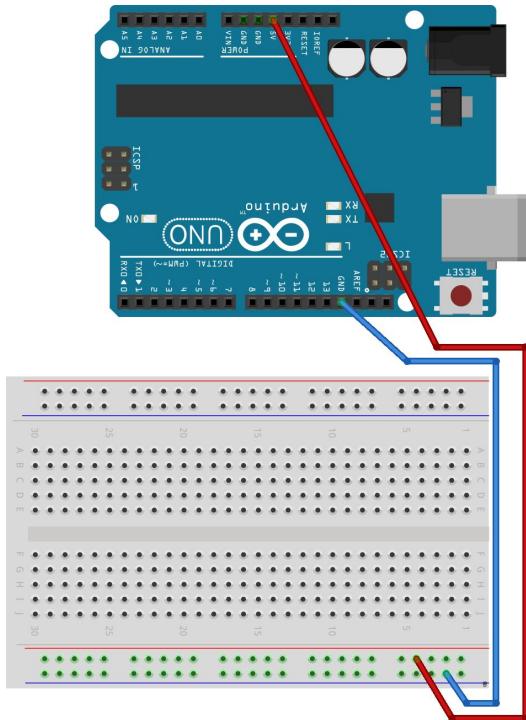
# Esquema elétrico



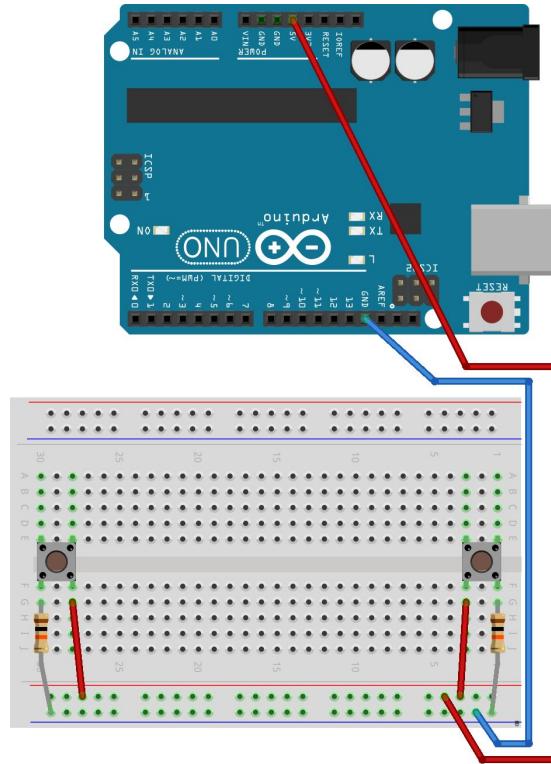
# Montagem do circuito (1)



# Montagem do circuito (2)

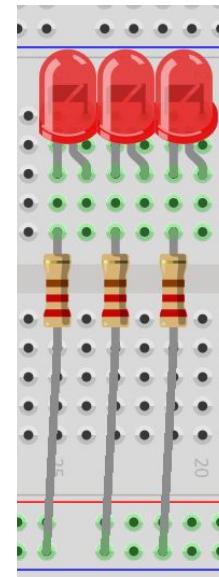
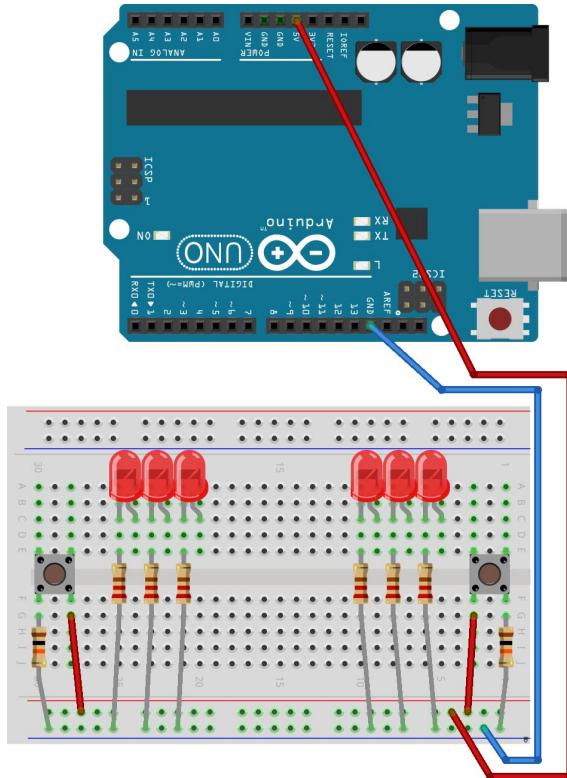


## Montagem do circuito (3)



Castanho  
Preto  
Laranja

# Montagem do circuito (4)

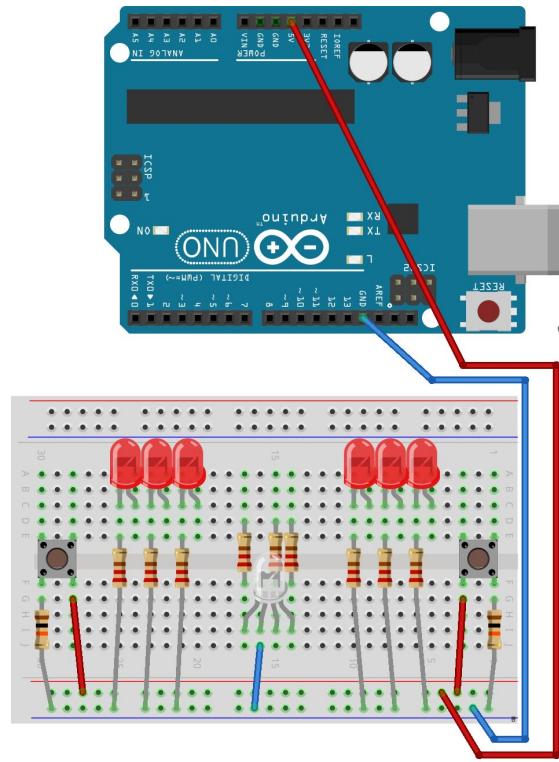


Castanho  
Vermelho  
Vermelho

# LED RGB

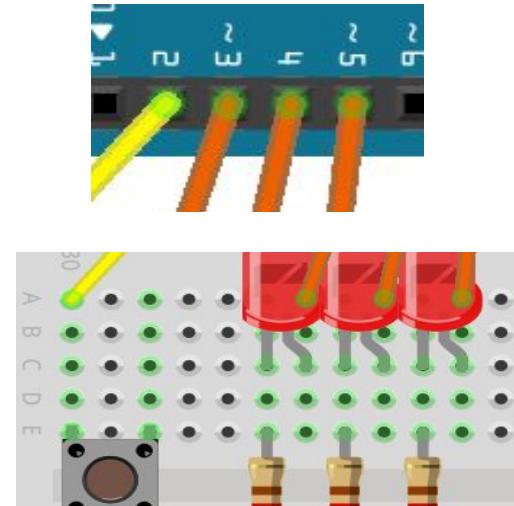
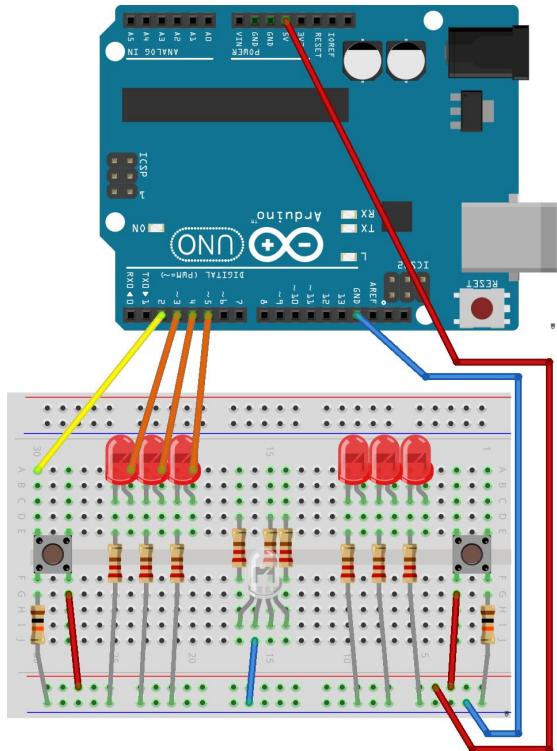


# Montagem do circuito (5)

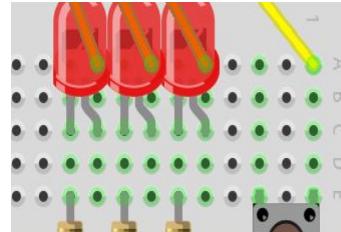
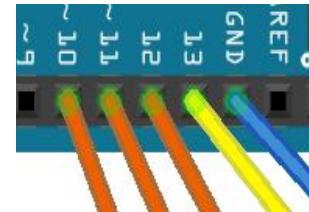
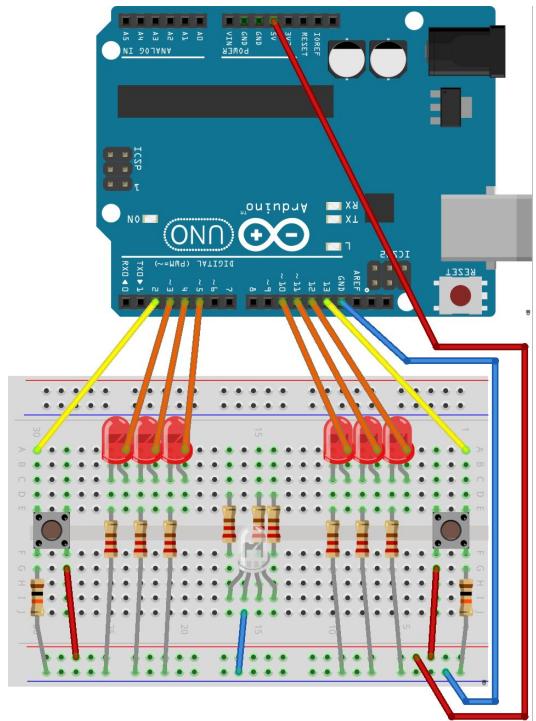


Castanho  
Vermelho  
Vermelho

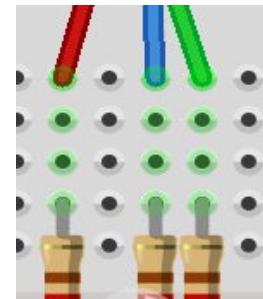
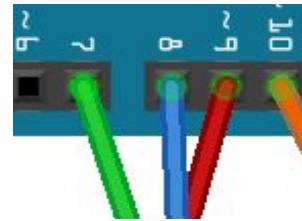
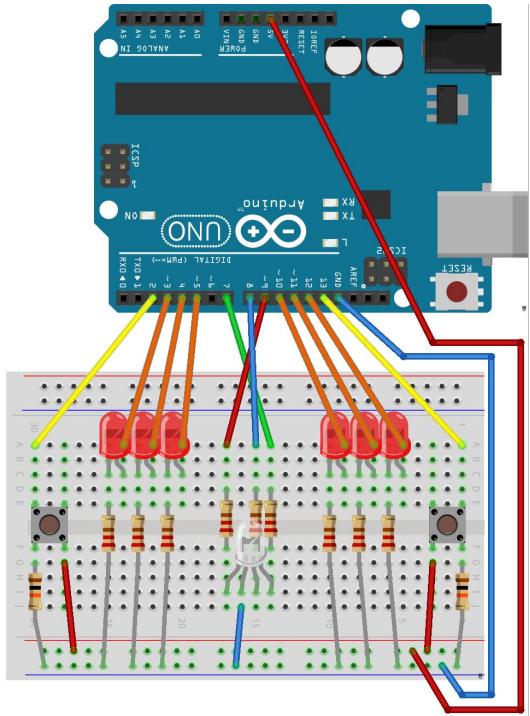
# Montagem do circuito (6)



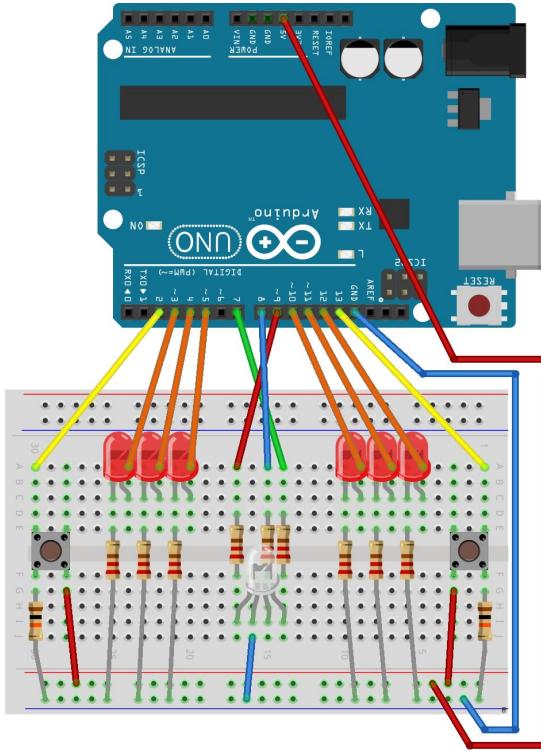
## Montagem do circuito (7)



# Montagem do circuito (8)



# Montagem do circuito (FINAL)



# Código

<http://bit.ly/NEECReactInc>

# Código (1)

```
Jogo React
-----
/*variáveis para botões*/
int button_p1 = 13;
int button_p2 = 2;

/*variáveis para leds de vida*/
int led_p1[] = {12, 11, 10};
int led_p2[] = {3, 4, 5};

/*variáveis para led rgb*/
int RGB_green = 7;
int RGB_blue = 8;
int RGB_red = 9;

/*variáveis de delay*/
int delay_max = 1000;
int delay_min = 300;
int delay_max_no_press = 900;
int delay_flash_init = 80;
int delay_end_flash = 100;

/* vidas de cada jogador */
int life_p1;
int life_p2;
```

# Código (2)

```
/*-----/setup() /-----*/  
void setup() {  
  
    // Input para os pins dos botões  
    pinMode(button_p1, INPUT);  
    pinMode(button_p2, INPUT);  
  
    // Output para os pins dos leds  
    for(int i = 0; i < 3; i++) {  
        pinMode(led_p1[i], OUTPUT);  
        pinMode(led_p2[i], OUTPUT);  
    }  
  
    // Output para os pins do RGB  
    pinMode(RGB_green, OUTPUT);  
    pinMode(RGB_red, OUTPUT);  
    pinMode(RGB_blue, OUTPUT);  
    // Utilizar o valor de um pin analógico não utilizado para o randomize  
    randomSeed(analogRead(A0));  
}  
/*-----/setup() /-----*/
```

# Código (3)

```
/*-----/loop () /-----*/  
void loop() {  
    int winner;  
  
    life_p1 = 3, life_p2 = 3;  
  
    // Chamar função que inicializa o jogo  
    Game_init();  
    delay(delay_max);  
  
    // Chamar função do jogo principal, a retornar o vencedor  
    winner = MainGame();  
  
    // Chamar função que acaba o jogo com a vitória de um player  
    EndGame(winner);  
    delay(delay_max);  
}  
/*-----/loop () /-----*/
```

# Código (4)

```
/*
-----  
função de inicialização (LEDS A PISCAR)
-----*/  
  
void Game_init() {  
  
    // Por três iterações (vezes)  
    for(int j = 0; j < 3; j++) {  
  
        // Piscar os leds (acender e apagar)  
        // Apagar primeiro para ficarem as vidas "prontas"  
        for(int i = 0; i < 3; i++) {  
            digitalWrite(led_p1[i], LOW);  
            digitalWrite(led_p2[i], LOW);  
            delay(delay_flash_init);  
        }  
    }  
}
```

# Código (5)

```
função principal para cada jogo
-----*/
int MainGame() {
    int i = 2, j = 2, rand_led, winner, rand_delay, w_max;

    while(life_p1 != 0 && life_p2 != 0) {

        w_max = random(3, 10);

        for(int w = 0; w < w_max; w++) {
            [REDACTED]
            digitalWrite(rand_led, HIGH);
            delay(50);
            [REDACTED]
            delay(50);
        }

        rand_led = random(RGB_green, RGB_red + 1);
        digitalWrite(rand_led, HIGH);

        winner = Round_Judge(rand_led);
        digitalWrite(rand_led, LOW);
```

# Código (6)

```
if(winner == 1) {  
    /* jogador 1 ganhou; jogador 2 perde 1 vida */  
    digitalWrite(led_p2[i], LOW);  
    i--;  
}  
  
else if(winner == 2) {  
    /* jogador 2 ganhou; jogador 1 perde 1 vida */  
}  
}  
  
else if(winner == 3) {  
    /* jogador 1 ganha 1 vida */  
    if(j < 2) {  
        j++;  
        digitalWrite(led_p1[j], HIGH);  
    }  
}
```

# Código (7)

```
else if(winner == 4) {
    /* jogador 2 ganha 1 vida */
    if(i < 2) {
        [REDACTED]
    }

    rand_delay = random(delay_min, delay_max);
    delay(rand_delay);
}

/* retorna sempre 1 ou 2 pois nao e fim do jogo se alguem ganhar 1 vida */
return winner;
}
```

# Código (8)

```
/*
função que calcula o vencedor de cada ronda
*/
int Round_Judge(int rand_led) {
    int over = 0, no_press = 0;
    int buttonState_p1, buttonState_p2, winner, no_press_limit;

    no_press_limit = random(500, delay_max_no_press);

    while(over == 0) {

        buttonState_p1 = digitalRead(button_p1);
        if(buttonState_p1 == HIGH) {
            if(rand_led == RGB_green) {
                life_p2--;
                winner = 1;
                over++;
            }
            else if(rand_led == RGB_red) {
}
            else if(rand_led == RGB_blue) {
                if(life_p1 < 3)
}
            }
        }
    }
}
```

# Código (9)

```
else if(buttonState_p2 == HIGH) {  
  
    if(rand_led == RGB_green) {  
        life_p1--;  
        winner = 2;  
        over++;  
    }  
    else if(rand_led == RGB_red) {  
        life_p2--;  
        winner = 1;  
        over++;  
    }  
    else if(rand_led == RGB_blue) {  
        if(life_p2 < 3)  
            life_p2++;  
        winner = 4;  
        over++;  
    }  
}  
  
else  
    /* truque ninja */  
    delay(4);  
}  
}
```

# Código (10)

```
/k-----  
função para o pisca-pisca no final de cada jogo  
-----/k/  
void EndGame(int winner) {  
    if(winner == 1) {  
        for(int i = 0; i < 5; i++) {  
            digitalWrite(led_p1[0], HIGH);  
            digitalWrite(led_p1[1], HIGH);  
            digitalWrite(led_p1[2], HIGH);  
            delay(delay_end_flash);  
            digitalWrite(led_p1[0], LOW);  
            digitalWrite(led_p1[1], LOW);  
            digitalWrite(led_p1[2], LOW);  
            delay(delay_end_flash);  
        }  
    }  
    else {  
        for(int i = 0; i < 5; i++) {  
            digitalWrite(led_p2[0], HIGH);  
            digitalWrite(led_p2[1], HIGH);  
            digitalWrite(led_p2[2], HIGH);  
            delay(delay_end_flash);  
            digitalWrite(led_p2[0], LOW);  
            digitalWrite(led_p2[1], LOW);  
            digitalWrite(led_p2[2], LOW);  
            delay(delay_end_flash);  
        }  
    }  
}
```

# Código Completo (em caso de problemas)

<http://bit.ly/NEECReactCompleto>

# Contactos

- ❑ [neecist.org](http://neecist.org)
- ❑ [facebook.com/NEECIST](https://facebook.com/NEECIST)
- ❑ [@neecist](https://twitter.com/@neecist)
- ❑ [NEECIST](#)

