Projeto final- LIP ARDUINO

Ciência da Computação

Lawer Böuch Soncin Rocha | Gabrielly Neres Barbara

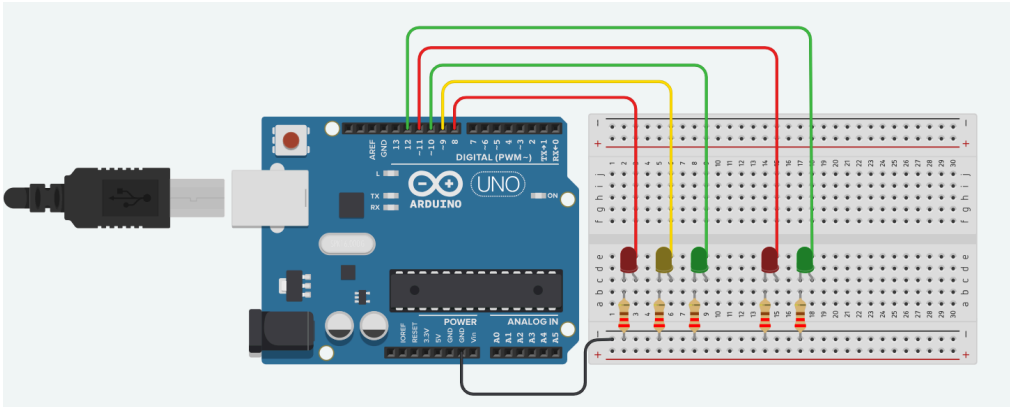
Ideia inicial:

Semáforo

Foi decidido desenvolver um projeto de semáforo inteligente utilizando Arduino. O sistema incluirá um botão de pedestre, um contador regressivo exibido em um display de sete segmentos e ajuste da intensidade das cores dos LEDs conforme a luminosidade ambiente, medida por meio de um sensor LDR. Além disso, serão implementadas duas funcionalidades adicionais: ao pressionar o botão, o semáforo abrirá instantaneamente; e, ao pressionar o botão cinco vezes, o semáforo entrará em modo de piscamento enquanto reproduz o som "Secret" do jogo Zelda.

Primeiro passo:

No primeiro dia do trabalho foi feito com que o sinal funcionasse, acendendo suas luzes na ordem vermelho, amarelo e verde. Quando o sinal de carros ficava verde, o de pedestres vermelho, quando o sinal de carros ficava vermelho, o de pedestres verde.



Segundo passo:

Adicionamos um botão de pedestres que fazia com que ao pedestre pressionar o botão o sinal abria automaticamente e se mantinha aberto por 5 segundos com um contador.

Terceiro passo:

Adicionamos a funcionalidade ao botão de pedestres que se, pressionado 5 vezes, tem uma funcionalidade secreta: toca um som, com o Buzzer.

Ideia final:

Foi desenvolvido um semáforo inteligente que inicia com o sinal verde (aberto) para os veículos. Ao pressionar o botão de pedestres, o semáforo automaticamente muda para o vermelho, permitindo que o sinal de pedestres fique verde. Após 5 segundos, o LED amarelo é ativado e, logo, o sistema retorna ao estado inicial do ciclo. Além disso, ao pressionar o botão de pedestres cinco vezes, uma funcionalidade surpresa é ativada: um som é emitido pelo Buzzer.

O código final foi:

```
#define a 2

#define b 3

#define c 4

#define d 5

#define e 13

#define f 1

#define g 0

// Definição dos pinos para os semáforos de carros

const int carro_verde = 10;

const int carro_amarelo = 9;

const int carro_vermelho = 8;

// Definição dos pinos para os semáforos de pedestres

const int pedestre_verde = 12;

const int pedestre_vermelho = 11;

// Definição do pino para o botão

const int botao_pino = 7;

// Definição do pino do LDR

const int ldr_pino = A0;
```

```
// Pino do buzzer
const int buzzer_pino = 6;
// Variáveis para armazenar o estado atual do semáforo
bool semaforo_carros_aberto = true;
bool semaforo_pedestres_aberto = false;
bool ultimo_estado_botao = false; // Variável para armazenar o estado anterior do botão
unsigned int contador_cliques = 0; // Contador de cliques no botão
void setup() {
  // Configura os pinos dos LEDs como saída
  pinMode(carro_verde, OUTPUT);
  pinMode(carro_amarelo, OUTPUT);
  pinMode(carro_vermelho, OUTPUT);
  pinMode(pedestre_verde, OUTPUT);
  pinMode(pedestre_vermelho, OUTPUT);
  pinMode(a, OUTPUT);
  pinMode(b, OUTPUT);
  pinMode(c, OUTPUT);
  pinMode(d, OUTPUT);
  pinMode(e, OUTPUT);
  pinMode(f, OUTPUT);
  pinMode(g, OUTPUT);
  // Garante que os LEDs dos semáforos iniciem apagados
  digitalWrite(carro_verde, LOW);
  digitalWrite(carro_amarelo, LOW);
  digitalWrite(carro_vermelho, HIGH); // Vermelho para carros inicialmente
  digitalWrite(pedestre_verde, HIGH); // Verde para pedestres inicialmente
```

```
  digitalWrite(pedestre_vermelho, LOW);
  // Configura o pino do botão como entrada com pull-up interno
  pinMode(botao_pino, INPUT_PULLUP);
  // Configura o pino do buzzer como saída
  pinMode(buzzer_pino, OUTPUT);
  // Inicializa o estado inicial dos semáforos
  atualizaSemaforos();
}
void display(int n){
  digitalWrite(a, LOW);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
  int x = n%10;
  if(x==0){
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
    digitalWrite(g, LOW);
  }
```

```
 else if(x==1){
   digitalWrite(a, LOW);
   digitalWrite(b, HIGH);
   digitalWrite(c, HIGH);
   digitalWrite(d, LOW);
   digitalWrite(e, LOW);
   digitalWrite(f, LOW);
   digitalWrite(g, LOW);
 }
 else if(x==2){
   digitalWrite(a, HIGH);
   digitalWrite(b, HIGH);
   digitalWrite(c, LOW);
   digitalWrite(d, HIGH);
   digitalWrite(e, HIGH);
   digitalWrite(f, LOW);
   digitalWrite(g, HIGH);
 }
 else if(x==3){
   digitalWrite(a, HIGH);
   digitalWrite(b, HIGH);
   digitalWrite(c, HIGH);
   digitalWrite(d, HIGH);
   digitalWrite(e, LOW);
   digitalWrite(f, LOW);
   digitalWrite(g, HIGH);
```

```
}
else if(x==4){
  digitalWrite(a, LOW);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}
else if(x==5){
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}
else if(x==6){
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
```

```
  digitalWrite(g, HIGH);
}
else if(x==7){
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
}
else if(x==8){
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}
else if(x==9){
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
```

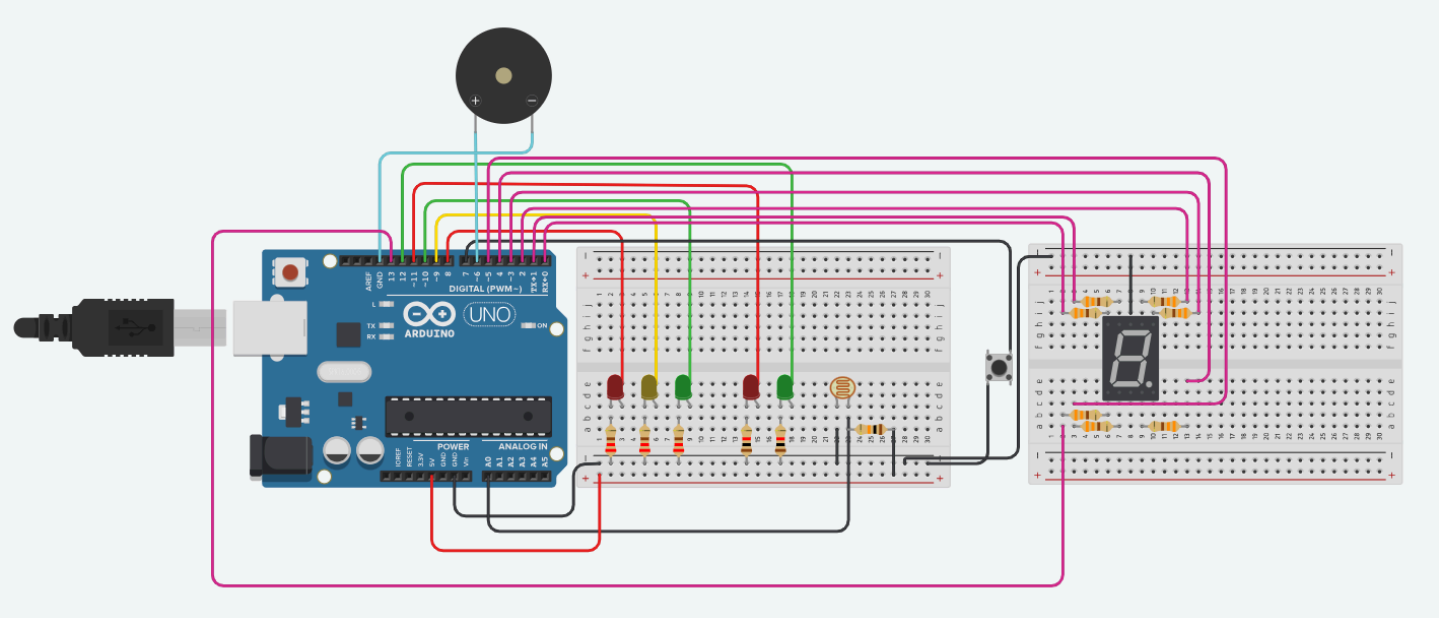
```
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
  }
}
void loop() {
  display(0);
  // Verifica se o botão foi pressionado para trocar os semáforos
  bool estado_botao = digitalRead(botao_pino);
  if (estado_botao == LOW && ultimo_estado_botao == HIGH) {
    // Apenas altera o estado se o botão foi pressionado
    trocaSemaforos();
    // Incrementa o contador de cliques no botão
    contador_cliques++;
    // Se atingiu 5 cliques, toca o easter egg
    if (contador_cliques == 5) {
      tocaSomSecreto();
      // Reinicia o contador de cliques
      contador_cliques = 0;
    }
  }
  ultimo_estado_botao = estado_botao;
  // Verifica o valor do LDR para ajustar a intensidade dos LEDs dos semáforos de pedestres
  int valor_ldr = analogRead(ldr_pino);
  // Ajusta a intensidade dos LEDs dos semáforos de pedestres com base no valor do LDR
  if (semaforo_pedestres_aberto) {
    // Exemplo de ajuste de intensidade (simplificado)
```

```
    int intensidade = map(valor_ldr, 0, 1023, 0, 255); // Mapeia o valor do LDR para o intervalo de 0 a 255
    analogWrite(pedestre_verde, intensidade);
  }
  delay(100); // Pequeno delay para estabilidade
}
// Função para atualizar os LEDs dos semáforos conforme o estado atual
void atualizaSemaforos() {
  if (semaforo_carros_aberto) {
    digitalWrite(carro_verde, HIGH);
    digitalWrite(carro_amarelo, LOW);
    digitalWrite(carro_vermelho, LOW);
    digitalWrite(pedestre_verde, LOW);
    digitalWrite(pedestre_vermelho, HIGH);
  } else {
    digitalWrite(carro_verde, LOW);
    digitalWrite(carro_amarelo, LOW);
    digitalWrite(carro_vermelho, HIGH);
    digitalWrite(pedestre_verde, HIGH);
    digitalWrite(pedestre_vermelho, LOW);
    display(5);
    delay(1000);
    display(4);
    delay(1000);
    display(3);
    delay(1000);
    display(2);
```

```
    delay(1000);
    display(1);
    delay(1000);
    display(0);
  }
}
// Função para trocar o estado dos semáforos
void trocaSemaforos() {
  // Troca o estado dos semáforos de carros e pedestres
  semaforo_carros_aberto = !semaforo_carros_aberto;
  semaforo_pedestres_aberto = !semaforo_pedestres_aberto;
  // Atualiza os LEDs conforme o novo estado
  atualizaSemaforos();
  // Aguarda 5 segundos antes de voltar ao estado normal
  delay(5000);
  // Troca para amarelo antes de voltar ao estado normal
  digitalWrite(carro_verde, LOW);
  digitalWrite(carro_amarelo, HIGH);
  delay(2000); // Sinal amarelo por 2 segundos
  digitalWrite(carro_amarelo, LOW);
  digitalWrite(carro_vermelho, HIGH);
  digitalWrite(pedestre_verde, LOW);
  digitalWrite(pedestre_vermelho, HIGH);
  // Inverte novamente para voltar ao estado normal após os 5 segundos
  semaforo_carros_aberto = !semaforo_carros_aberto;
  semaforo_pedestres_aberto = !semaforo_pedestres_aberto;
```

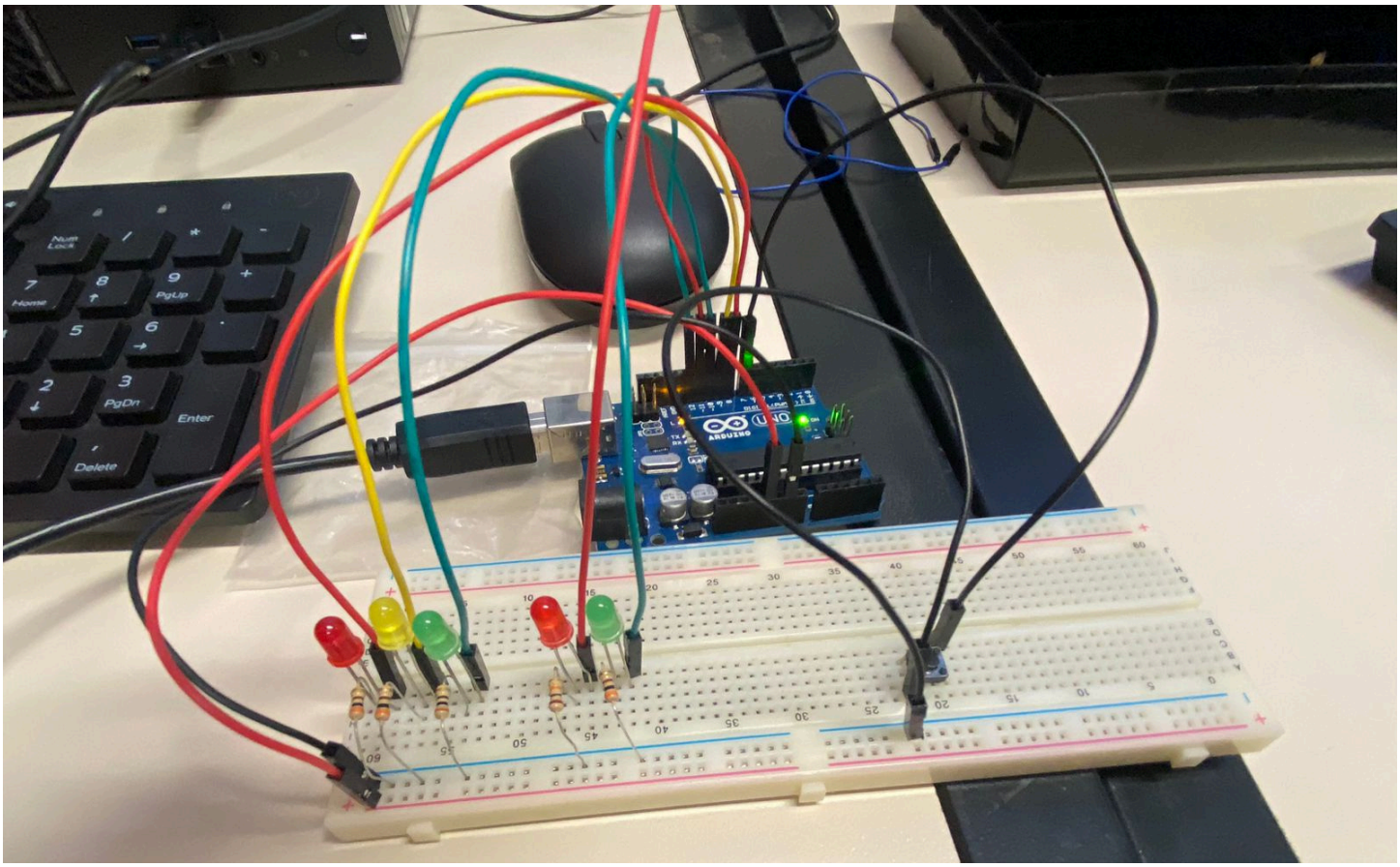
```
  // Atualiza os LEDs para refletir o estado normal
  atualizaSemaforos();
}
// Função para tocar o som secreto
void tocaSomSecreto() {
  int jingle[] = {262, 294, 330, 349, 392, 440, 494}; // Frequências das notas (C4, D4, E4, F4, G4, A4, B4)
  int duracao = 100; // Duração de cada nota em milissegundos
  for (int i = 0; i < 7; i++) {
    tone(buzzer_pino, jingle[i], duracao);
    delay(duracao + 50); // Pequeno intervalo entre as notas
    noTone(buzzer_pino);
  }
}
```

E o Tinkercad final:



Foi realizado a montagem física do Arduino, entretanto, alguns componentes necessários não estavam disponíveis para que a funcionalidade fosse completamente equivalente à simulação no

Tinkercad. Assim, o dispositivo apresenta apenas as funcionalidades básicas de abertura dos sinais de veículos e pedestres, bem como o botão para a abertura imediata do sinal de pedestres.



Conclusão:

O trabalho sofreu algumas alterações em relação à sua concepção inicial. Por exemplo, o botão de pedestres precisa ser pressionado apenas uma vez para abrir o sinal para os pedestres, entre outras modificações. Compreendemos a extrema importância deste projeto para a nossa formação, e é inegável que muitos aprendizados foram adquiridos ao longo do processo de concepção do mesmo.