Capacitação para aulas de robótica

Conteúdos gerais:

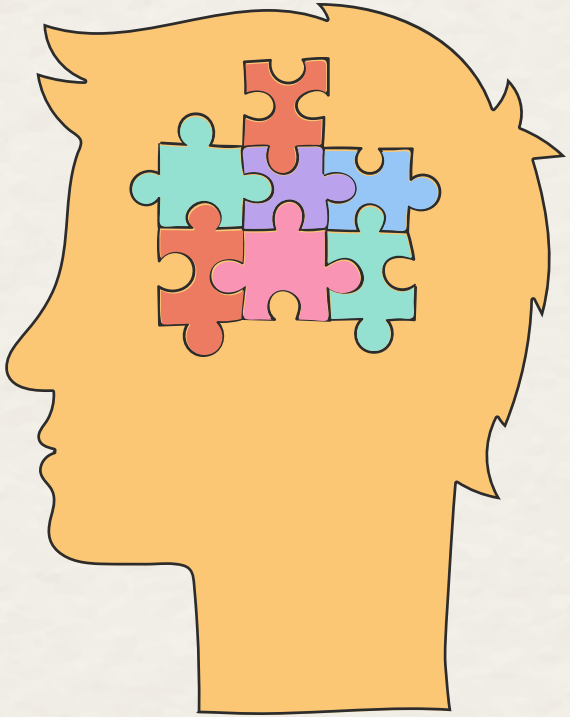
01 Conhecendo os componentes

02 Programação com Arduino

03 Projetos

01

Conhecendo os componentes



Arduino UNO

O que é o Arduino?

- Um Arduino é um microcontrolador, que funciona como o cérebro de um projeto eletrônico.
- Ele é programável e pode executar instruções para controlar diferentes dispositivos e realizar diversas ações.

Arduino UNO

Funções do Arduino

- Receber informações: O Arduino possui entradas, chamadas de "pinos", onde pode receber dados de sensores, botões, etc.
- Enviar comandos: O Arduino possui saídas, chamadas de "pinos de saída", onde pode enviar sinais para controlar motores, luzes, alto-falantes, etc.

Arduino UNO

Programação do Arduino

- Escrever código: Utiliza-se uma linguagem de programação (geralmente baseada em C/C++) para criar um conjunto de instruções, chamado de "sketch", que será executado pelo Arduino.

- Definir regras: No código, você especifica as ações que deseja que o Arduino execute com base nas entradas que recebe.
- Exemplo de regra: "Se alguém se aproximar, acenda as luzes".

Arduino UNO

Execução do projeto

- O Arduino interpreta o código e executa as instruções definidas, controlando o projeto conforme as regras estabelecidas.
- Ele pode interagir com o ambiente, respondendo a estímulos e realizando tarefas específicas conforme programado.

Pinos: Analógicos e Digitais

Funções dos pinos do Arduino

- Os pinos do Arduino são pontos de conexão que permitem interagir com diferentes componentes eletrônicos em um projeto.
- Existem dois tipos principais de pinos: digitais e analógicos.

Pinos: Analógicos e Digitais

Pinos Analógicos

- Funcionam como medidores que podem ter valores variáveis entre 0 e 1023.
- São úteis para medir grandezas contínuas, como luz, som ou temperatura.
- Permitem uma ampla gama de aplicações em projetos que exigem leitura de dados analógicos.
- Exemplo: Você pode conectar um sensor de luz a um pino analógico, e o Arduino pode ler os valores fornecidos pelo sensor para determinar a intensidade da luz ambiente. Isso pode ser usado para controlar automaticamente a luminosidade de uma luminária.

Pinos: Analógicos e Digitais

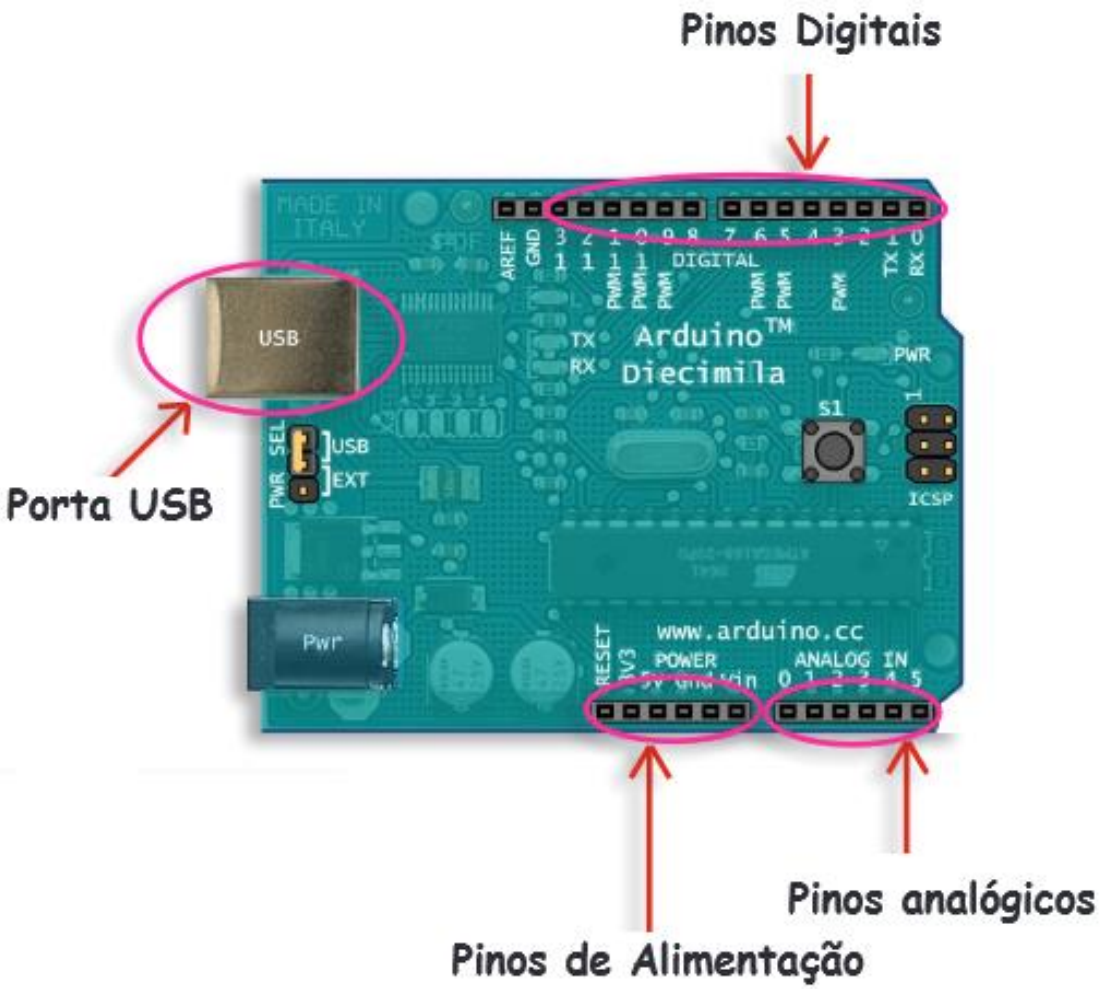
Pinos Digitais

- Funcionam como interruptores que podem estar ligados (HIGH) ou desligados (LOW).
- São ideais para controlar componentes que têm apenas duas condições, como LEDs, relés e botões.
- Exemplo: Você pode conectar um LED a um pino digital e, ao programar o Arduino, fazer o LED piscar ligando e desligando o pino digital.

Pinos: Analógicos e Digitais

Aplicações Criativas

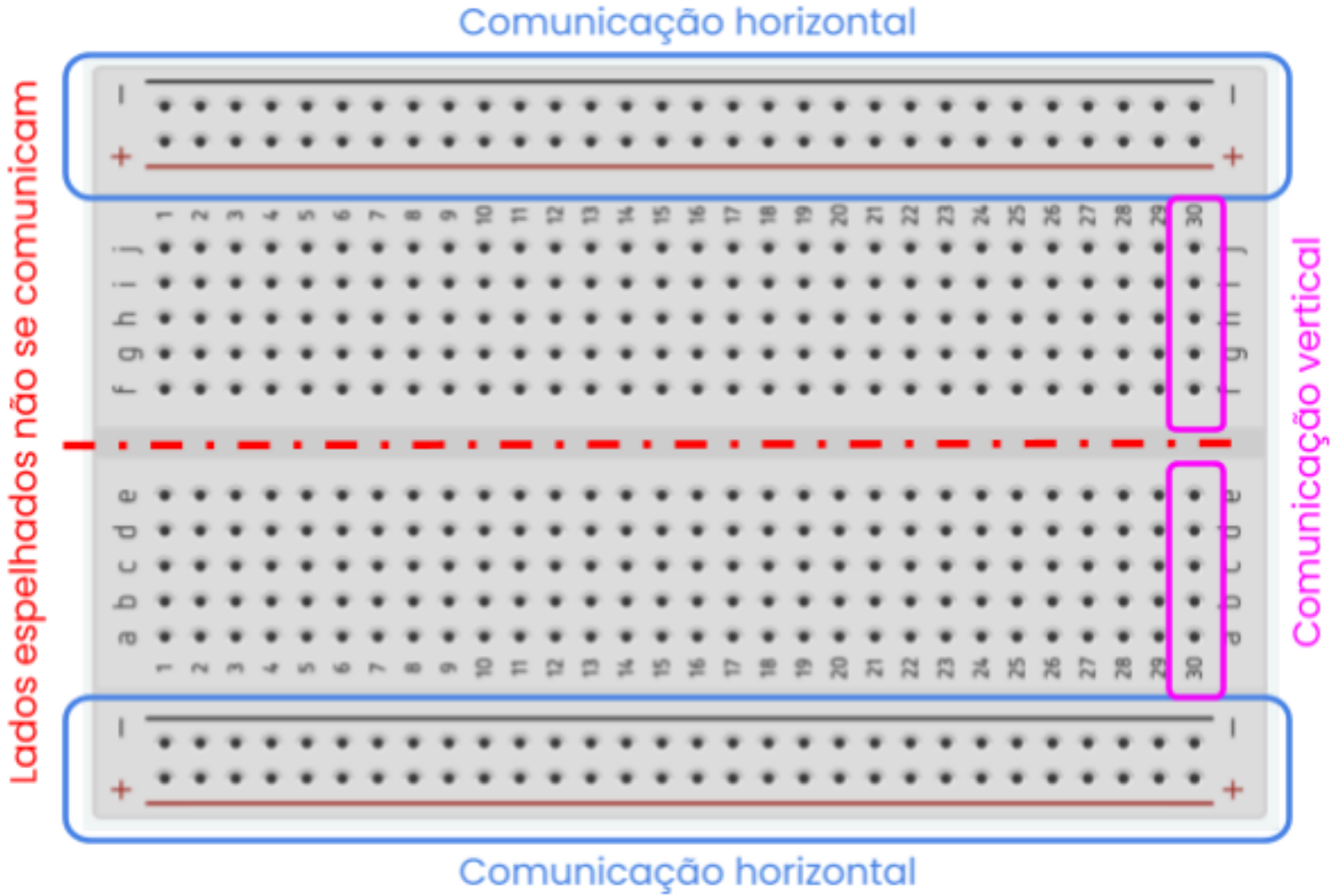
- Entendendo como usar esses pinos, você pode criar projetos eletrônicos incríveis e interagir com uma variedade de componentes de maneiras diversas.
- Os pinos digitais são úteis para controle binário (ligado/desligado), enquanto os pinos analógicos permitem uma ampla gama de valores, oferecendo flexibilidade em projetos criativos.


Pinos Digitais
Porta USB
Pinos de Alimentação
Pinos analógicos
USB
Pwr
Arduino™
Diecimila
DIGITAL
ANALOG IN
POWER
www.arduino.cc
PWR
ICSP
S1
RESET
AREF
GND
TX
RX
SEL
EXT

Protoboard (mini)

Descrição das mini-protoboards

- As mini-protoboards são placas de montagem que facilitam a conexão de fios e componentes eletrônicos em projetos.
- Elas possuem buracos onde é possível encaixar os fios e os componentes.
- Cada linha horizontal de buracos está conectada entre si, assim como as colunas verticais.


Comunicação horizontal
Lados espelhados não se comunicam
Comunicação vertical
Comunicação horizontal

Protoboard (mini)

Conexão de componentes

- A disposição dos buracos facilita a conexão de componentes elétricos sem a necessidade de solda.
- Para conectar dois componentes, basta colocar os fios nos buracos da mesma linha ou coluna.

Protoboard (mini)

Conexão entre linhas e colunas

- Não há conexão automática entre as linhas horizontais e as colunas verticais.
- Se desejar conectar um fio de uma linha a outro de uma coluna, é necessário usar um fio para fazer essa conexão.

Protoboard (mini)

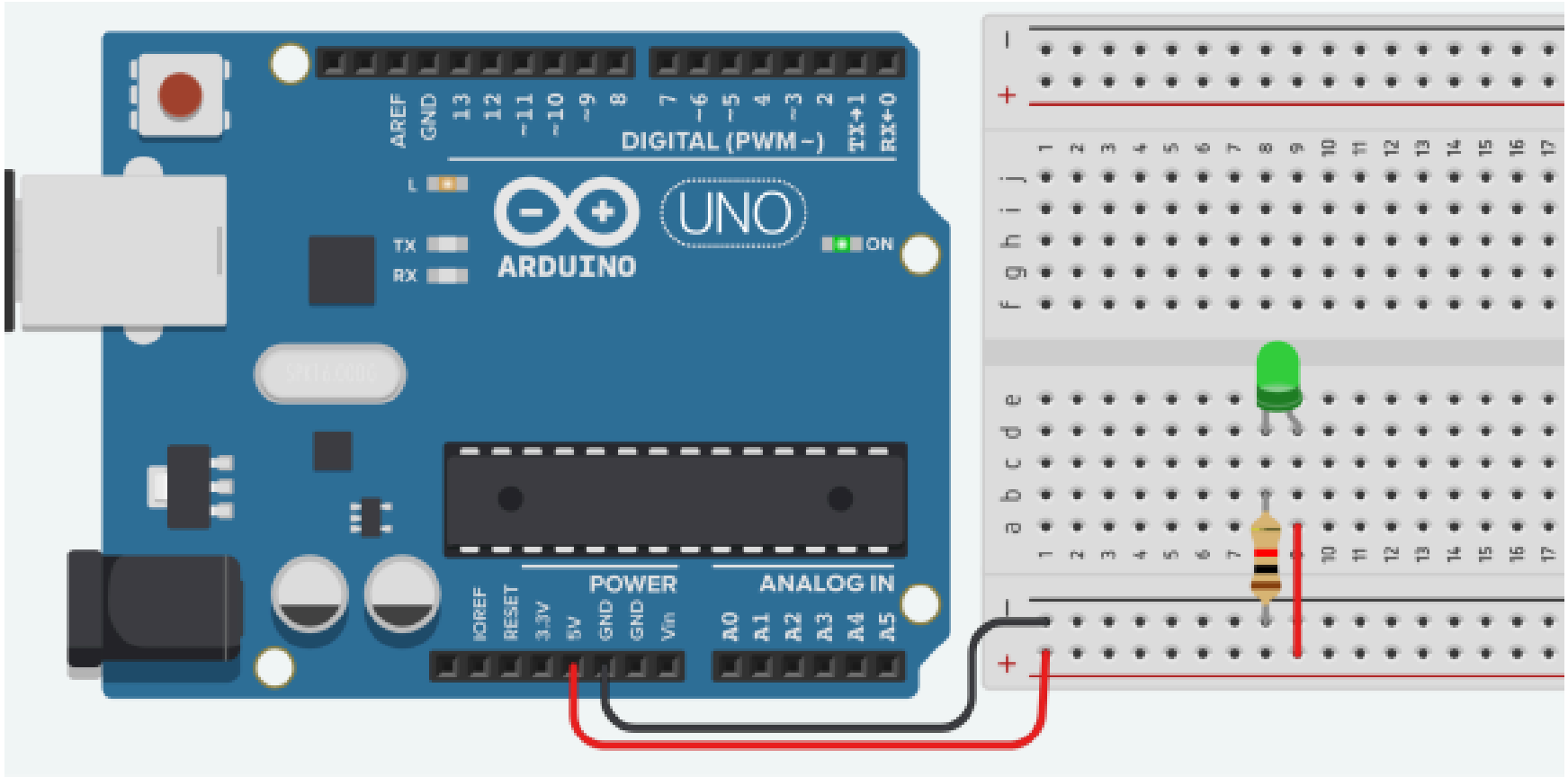
Espelhamento

- Em algumas mini-protoboards, as linhas são espelhadas em lados opostos.
- Isso significa que a linha superior de um lado está conectada com a linha inferior do lado oposto, e vice-versa.
- No entanto, as colunas geralmente não são espelhadas.
- Para conectar componentes entre os lados espelhados, é necessário usar um fio para fazer essa conexão.

LED

Descrição dos Leds

- Os LEDs são dispositivos eletrônicos que emitem luz quando uma corrente elétrica passa por eles.
- Eles possuem um lado longo (+) e um lado curto (-), também conhecido como ânodo e cátodo, respectivamente.


AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5
a b c d e
f g h i j
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

LED

Conexão ao Arduino

- Ao conectar um LED ao Arduino, certifique-se de que o lado longo esteja conectado ao fio positivo de 3V (+) e o lado curto ao fio negativo (-) para que a luz acenda.
- Experimente diferentes cores de LEDs para criar padrões de luzes divertidos em seus projetos.

LED

Uso de resistores

- Ao ligar um LED a um pino de 5V ou a uma saída digital do Arduino, é importante usar um resistor para limitar a corrente elétrica e proteger o LED.
- Este resistor pode ser ligado junto à perninha menor ou maior do LED, conectado ao GND (terra) ou VCC (5V) do sistema.

LED

Escolha do resistor

- Utilize a tabela de resistores por cor para escolher o resistor adequado para o seu projeto, considerando a tensão de operação do LED e a corrente desejada.
- A escolha correta do resistor ajuda a evitar danos ao LED e garante o funcionamento seguro do circuito.

Resistor (OHMS) por cor de LED a depender da tensão (3v, 5v ou 9v)							
Tensão	amarelo	laranja	vermelho	verde	azul	roxo	branco
3v	33 Ω	33 Ω	33 Ω	-	-	-	-
5V	150 Ω	150 Ω	150 Ω	82 Ω	82 Ω	82 Ω	82 Ω
9V	330 Ω	330 Ω	330 Ω	270 Ω	270 Ω	270 Ω	270 Ω

Resistores

Descrição dos resistores

- Os resistores são componentes eletrônicos que limitam a quantidade de corrente elétrica que passa por eles em um circuito.
- Eles são como "obstáculos" que ajudam a controlar o fluxo de eletricidade.

Resistores

Uso no Arduino

- Os resistores são usados em projetos com Arduino por várias razões.
- Uma das razões mais comuns é limitar a corrente que passa por LEDs, protegendo-os de receberem uma corrente excessiva que poderia danificá-los ou queimá-los.

Resistores

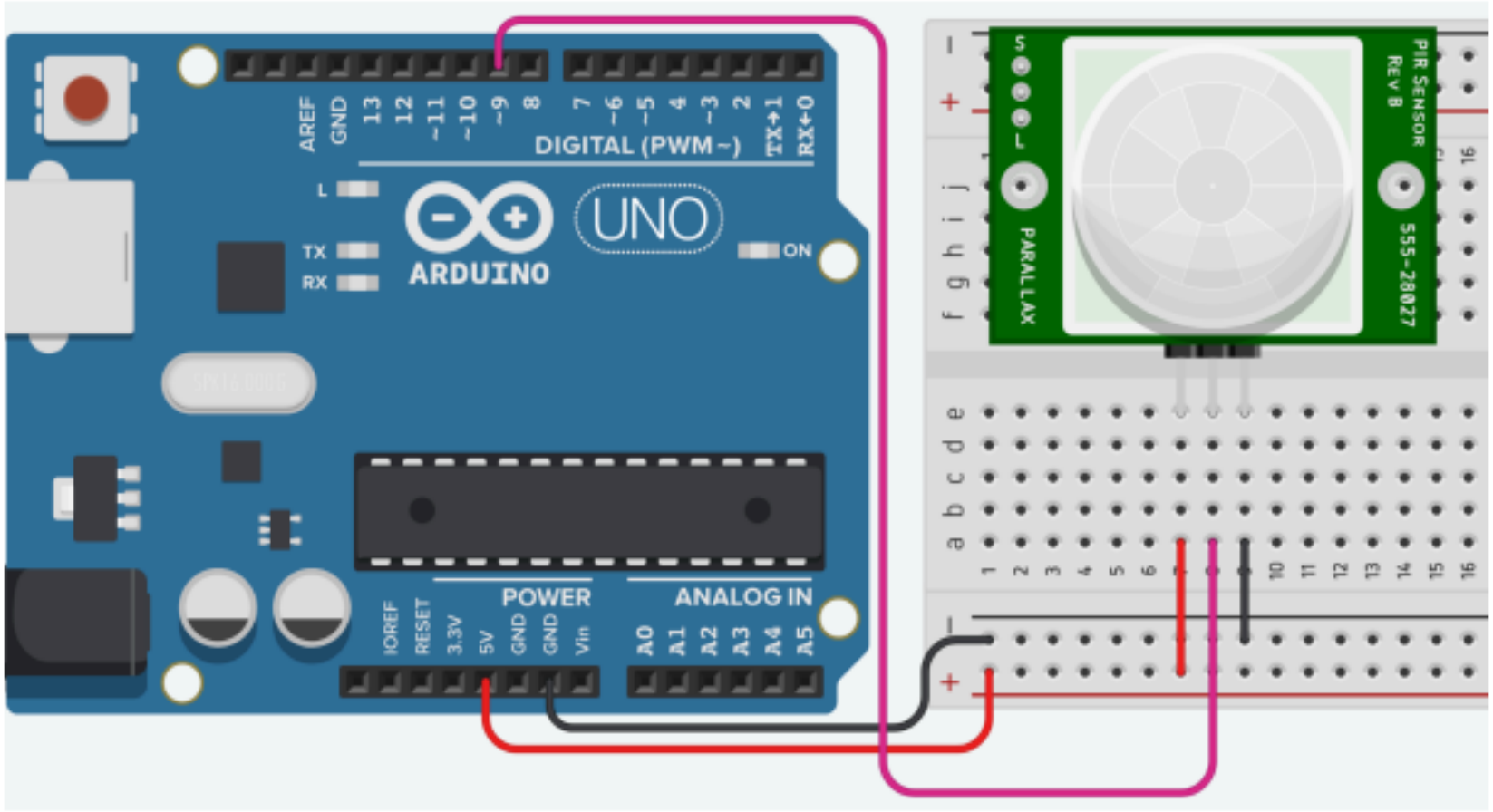
Uso dos resistores

- Além de limitar a corrente em LEDs, os resistores são usados em outros contextos, como em divisores de tensão, pull-up e pull-down em circuitos.
- Eles ajudam a garantir que os pinos do Arduino recebam sinais elétricos adequados, evitando danos aos componentes e garantindo o funcionamento correto do circuito.

Módulo sensor de movimento

Descrição do sensor PIR

- O sensor PIR (Passive Infrared Sensor) é capaz de detectar movimento por meio da detecção de variações no calor emitido por objetos em seu campo de visão.
- Geralmente, o módulo PIR possui três pinos: VCC (alimentação), OUT (saída) e GND (terra).


AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5
PARALLAX
PIR SENSOR
Rev B
555-28027
a b c d e
f g h i j
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Módulo sensor de movimento

Uso no Arduino

- Conecte o pino VCC do módulo PIR ao pino 5V do Arduino para fornecer alimentação.
- Conecte o pino GND do módulo PIR ao pino GND do Arduino para estabelecer a referência de terra.
- Conecte o pino OUT do módulo PIR a um pino digital do Arduino, como o pino 9, por exemplo, para receber o sinal de detecção de movimento.

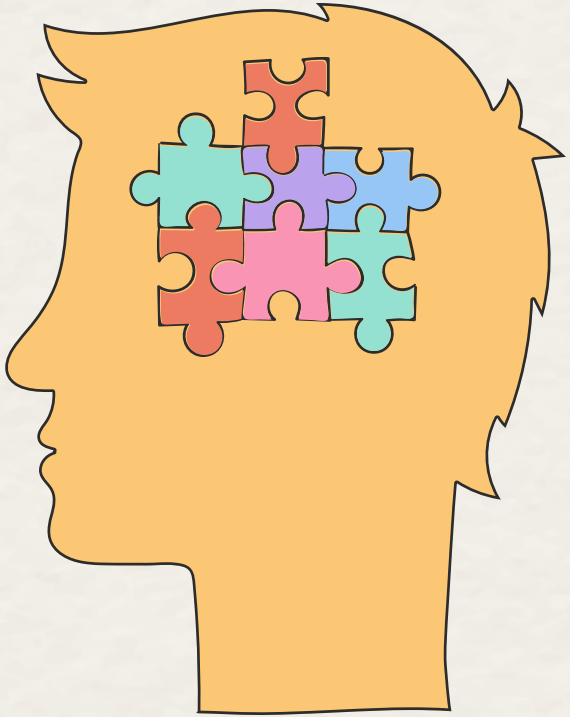
Módulo sensor de movimento

Funcionamento do sensor

- Quando o sensor PIR detecta algum movimento, ele envia um sinal de nível alto (HIGH) ao Arduino por meio do pino OUT.
- Isso permite que o Arduino detecte a presença de movimento e execute alguma ação programada em resposta, como acender uma lâmpada ou ativar um alarme.

02

Programação com Arduino



Ferramentas

Arduino IDE

- O Arduino IDE é como uma linguagem que você usa para contar uma história ao seu Arduino.
- É um software que facilita escrever, editar e enviar código para o Arduino.
- É como a ferramenta que você usa para dar instruções ao seu Arduino.

Como utilizar ?

Instalação

- Você pode baixar o Arduino IDE gratuitamente no site oficial: https://www.arduino.cc/en/software
- Ele está disponível para várias plataformas, como Windows, Mac e Linux. Escolha a versão adequada ao seu sistema operacional.

Como utilizar ?

Escrever o código

- No Arduino IDE, você escreve o código usando a linguagem de programação C/C++. Pode parecer complicado no início, mas você verá que é como contar uma história ao Arduino, dizendo o que você quer que ele faça.

Como utilizar ?

Conectar no computador

- Antes de enviar as instruções ao Arduino, você precisa conectar o Arduino ao computador usando um cabo USB. O Arduino IDE usará essa conexão para enviar o código que você escreveu para o Arduino.

Como utilizar ?

Selecionar a placa e a porta

- Conecte seu Arduino a uma porta USB do seu computador.
- No Arduino IDE, clique em "Ferramentas" (Tools) no canto superior direito.
- Escolha o tipo de placa Arduino que você está usando, como "Arduino Uno".
- Ao lado da opção da placa, você verá uma lista suspensa para a "Porta".
- A porta do Arduino será listada lá, identificada pelo nome do dispositivo e um número de porta.
- Em sistemas Windows, pode aparecer como "COM3". Em sistemas MacOS ou Linux, pode ser algo como "/dev/ttyUSB0" ou "/dev/cu.usbmodemxxxx".

Como utilizar ?

Enviar o código (Upload)

- Depois de escrever o código, você pressiona o botão "Upload". Isso faz com que o Arduino IDE envie o código para o Arduino. É como dar a história escrita para o robô seguir.

Como utilizar ?

Observar a execução

- O Arduino IDE tem uma ferramenta chamada Monitor Serial, onde você pode ver mensagens ou dados que o Arduino envia de volta para o computador. Isso é útil para entender o que está acontecendo no seu projeto.

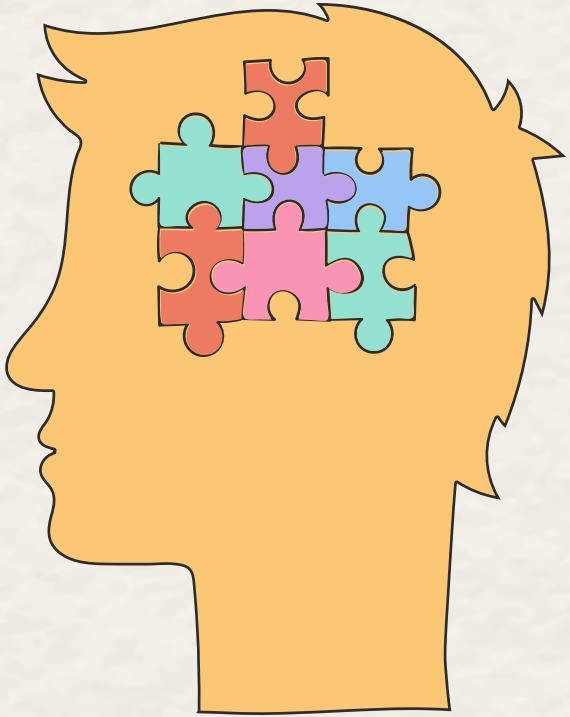
Como utilizar ?

Simulação

- Tinkercad é uma plataforma online onde você pode criar, simular e experimentar projetos de eletrônica e programação virtualmente.
- Não há necessidade de fios, componentes físicos ou um Arduino real.
- Você pode experimentar com eletrônica e programação de maneira virtual usando esta ferramenta online.
- Acesse pelo link: https://www.tinkercad.com/, caso não tenha uma conta e possível cria-la gratuitamente.

03

Projetos



Protótipo do semáforo

Crie um protótipo de semáforo de trânsito que simula a mudança de sinal usando LEDs.

MATERIAIS:

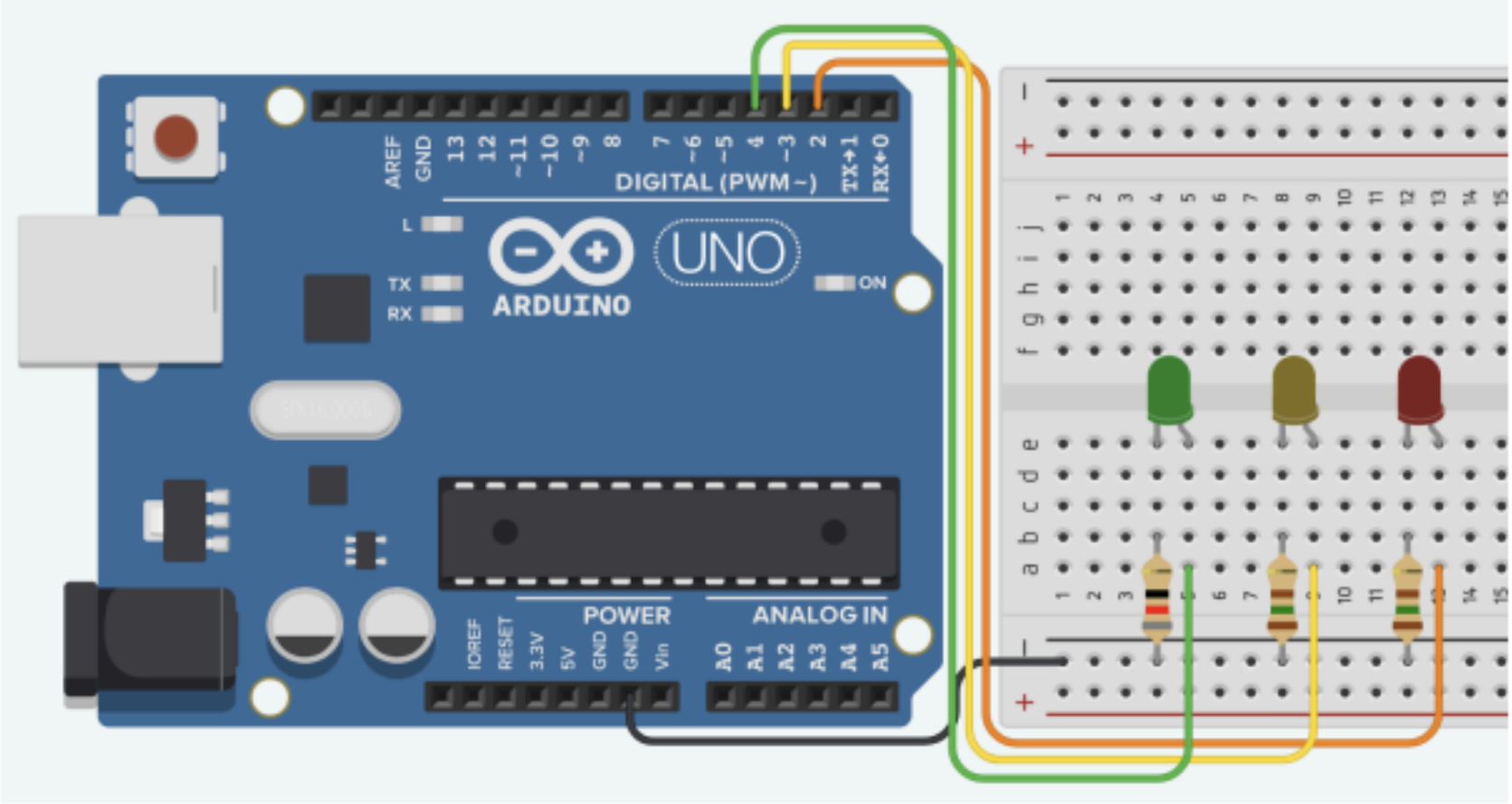
- ✓ 3 leds (verde, amarelo e vermelho)
- ✓ 1 protoboard
- ✓ 3 resistores
- ✓ Jumpers diversos

Protótipo do semáforo

Crie um protótipo de semáforo de trânsito que simula a mudança de sinal usando LEDs.

MONTAGEM:

- Conecte o LED Vermelho ao pino digital 2 do Arduino.
- Conecte o LED Amarelo ao pino digital 3 do Arduino.
- Conecte o LED Verde ao pino digital 4 do Arduino.
- Conecte um resistor 150 Ω no LED Vermelho e Amarelo.
- Conecte um resistor 82 Ω no LED Verde.
- Conecte um jumper da trilha negativa (-) da protoboard ao GND do Arduino.


AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5
a b c d e
f g h i j
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

CÓDIGO:

```
// Definindo os pinos para os LEDs
const int pinVermelho = 2;   // Pino digital 2 para o LED vermelho
const int pinAmarelo = 3;    // Pino digital 3 para o LED amarelo
const int pinVerde = 4;      // Pino digital 4 para o LED verde

void setup( ) {
  // Configura os pinos dos LEDs como saídas
  pinMode(pinVermelho, OUTPUT);
  pinMode(pinAmarelo, OUTPUT);
  pinMode(pinVerde, OUTPUT);
}

void loop( ) {
  // Ciclo de Semáforo
  // Sinal Verde por 30 segundos
  digitalWrite(pinVerde, HIGH);
  delay(30000); // 30 segundos

  // Sinal Amarelo por 3 segundos
  digitalWrite(pinVerde, LOW);
  digitalWrite(pinAmarelo, HIGH);
  delay(3000); // 3 segundos

  // Sinal Vermelho por 30 segundos
  digitalWrite(pinAmarelo, LOW);
  digitalWrite(pinVermelho, HIGH);
  delay(30000); // 30 segundos

  // Reinicia o ciclo
  digitalWrite(pinVermelho, LOW);
}
```

Explicando o Código

Definição dos Pinos: Usamos os pinos digitais 2, 3 e 4 do Arduino para controlar os LEDs vermelho, amarelo e verde, respectivamente.

Configuração dos LEDs:

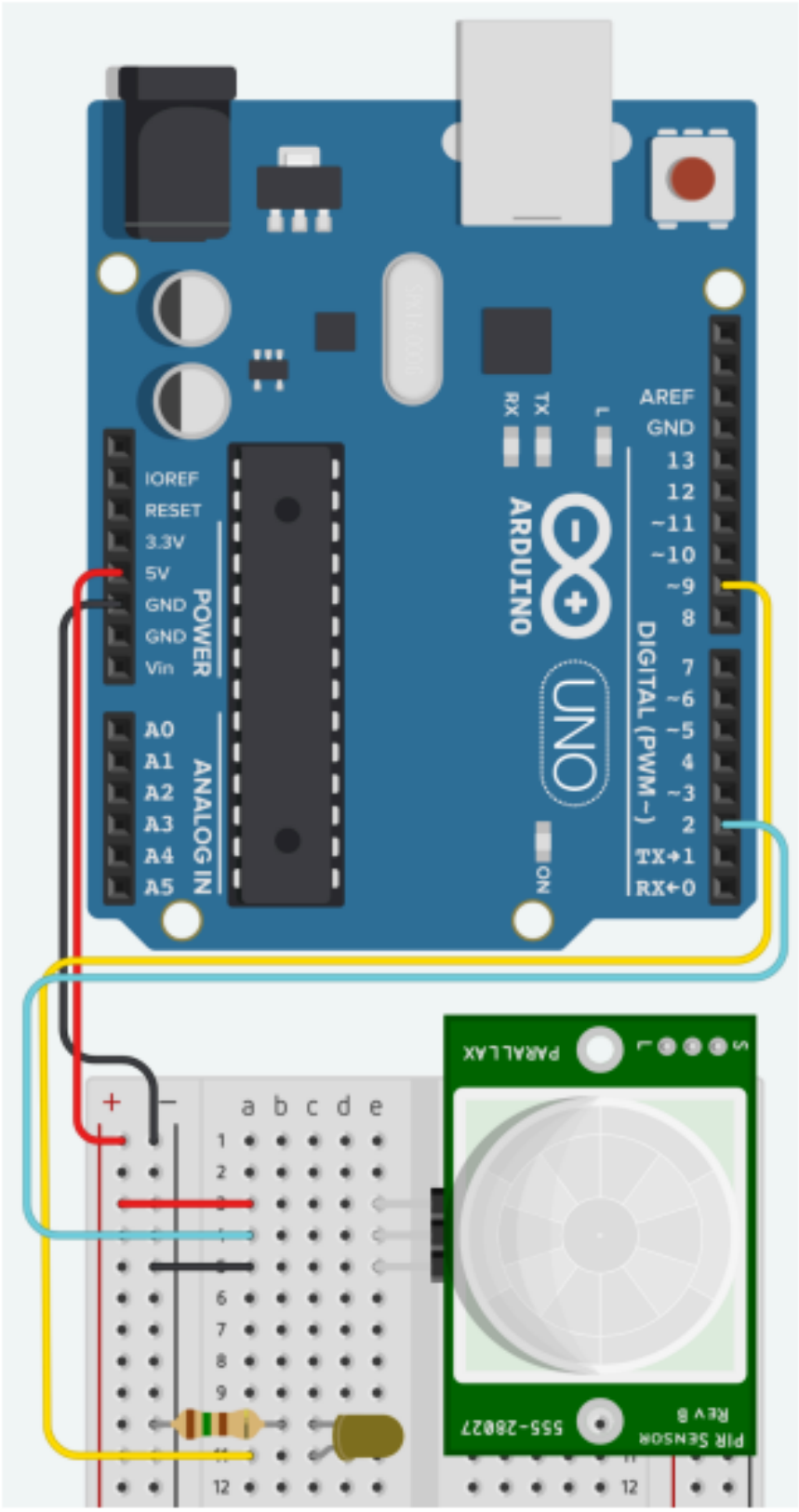
- No `setup()`, configuramos os pinos como saída (`OUTPUT`) para controlar os LEDs.
- Usamos **`digitalWrite(pino, HIGH)`** para ligar um LED (enviando sinal `HIGH`) e **`digitalWrite(pino, LOW)`** para desligá-lo (enviando sinal `LOW`).
- **Ciclo do Semáforo:**
- Utilizamos `delay` para criar intervalos de espera em milissegundos.
- Luz Verde: Acendemos o LED verde e esperamos 30 segundos.
- Luz Amarela: Apagamos o LED verde, acendemos o amarelo por 3 segundos.
- Luz Vermelha: Apagamos o LED amarelo, acendemos o vermelho por 30 segundos.
- Este ciclo se repete continuamente dentro do `loop()`.

Luminária Automática

Crie uma luminária inteligente que se acende automaticamente em ambientes quando detecta movimento, proporcionando praticidade e economia de energia.

MATERIAIS:

- ✓ Arduino Uno
- ✓ Protoboard
- ✓ Sensor de Movimento PIR (Passive Infrared)
- ✓ LED (qualquer cor)
- ✓ Resistor (limita a corrente do LED)
- ✓ Cabos Jumper (fios de conexão)


IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
RX
TX
L
ARDUINO
UNO
ON
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
a b c d e
1
2
6
7
8
9
12
PARALLAX
555-28027
PIR SENSOR
REV B

MONTAGEM:

1. Conectando o Sensor de movimento (PIR):

a. Conecte o pino de saída do sensor PIR ao pino digital 2 do Arduino.

b. Conecte o pino de alimentação do sensor PIR ao pino 5V do Arduino.

c. Conecte o pino de terra (GND) do sensor PIR ao GND.

2. Conectando um LED corretamente:

a. Conecte o anodo (perninha mais longa) do LED ao pino digital 9 do Arduino.

b. Conecte a catodo (perninha mais curta) do LED a um resistor (verifique qual o melhor resistor de acordo com a cor e a nossa tabela na seção LED do capítulo "Conhecendo os componentes").

c. Conecte outra extremidade do resistor ao terra (GND).

MONTAGEM:

3. Observações importantes:

a. Posicione o sensor PIR de tal forma que seja possível detectar a aproximação de alguém ou algum movimento.

b. O LED está fazendo o papel de uma lâmpada comum, imagine que estamos prototipando uma lâmpada que acenda sozinha ao detectar a presença de uma pessoa no corredor, por exemplo.

Código:

```
int pirPin = 2;  // Pino de entrada do sensor PIR
int ledPin = 9;  // Pino de controle do LED
int pirState = LOW;  // Variável para armazenar o estado do sensor de movimento
int val = 0;  // Variável para armazenar o valor lido do sensor PIR

void setup() {
  pinMode(pirPin, INPUT);  // Configura o pino do sensor PIR como entrada
  pinMode(ledPin, OUTPUT);  // Configura o pino do LED como saída
  Serial.begin(9600);  // Inicializa a comunicação serial (opcional, para depuração)
}

void loop() {
  val = digitalRead(pirPin);  // Lê o valor do sensor PIR

   if (val == HIGH) {   // Verifica se há movimento detectado
          digitalWrite(ledPin, HIGH);  // Se houver movimento, acende o LED
          if (pirState == LOW) {   // O estado pirState está LOW?
                  Serial.println("Movimento Detectado!"); // Escrevemos no nosso "diário"
                  pirState = HIGH; // Atualiza o estado do sensor
          }
  } else {    // Se não houver movimento, apaga o LED
          digitalWrite(ledPin, LOW);
          if (pirState == HIGH) {   // O estado pirState está HIGH?
                  Serial.println("Sem Movimento"); // Escrevemos no nosso "diário"
                  pirState = LOW; // Atualiza o estado do sensor
          }
  }
}
```

Explicando o Código

Definição de Pinos:

- pirPin: Pino conectado ao sensor de movimento PIR.
- ledPin: Pino conectado ao LED.

Variáveis de Controle:

- pirState: Armazena o estado atual do sensor de movimento (movimento ou sem movimento).
- val: Armazena o valor lido do sensor PIR (HIGH ou LOW).

Configuração (Setup):

- Configuramos os pinos como entrada (para o sensor PIR) e saída (para o LED).
- Inicializamos a comunicação serial para depuração (opcional).

Loop:

- Leitura do sensor PIR para detectar movimento.
- Se houver movimento, acendemos o LED e atualizamos o estado.
- Se não houver movimento, apagamos o LED e atualizamos o estado.
- Mensagens opcionais ("Movimento Detectado!" ou "Sem Movimento") são exibidas no Monitor Serial.

Jogo Genius

Construa um jogo onde os LEDs acendem em sequência e jogador deve pressionar o botão na ordem correta.

MATERIAIS:

- ✓ LEDs Coloridos (4)
- ✓ Botões de Pressão
- ✓ Resistores (8)
- ✓ Jumpers
- ✓ Arduino UNO

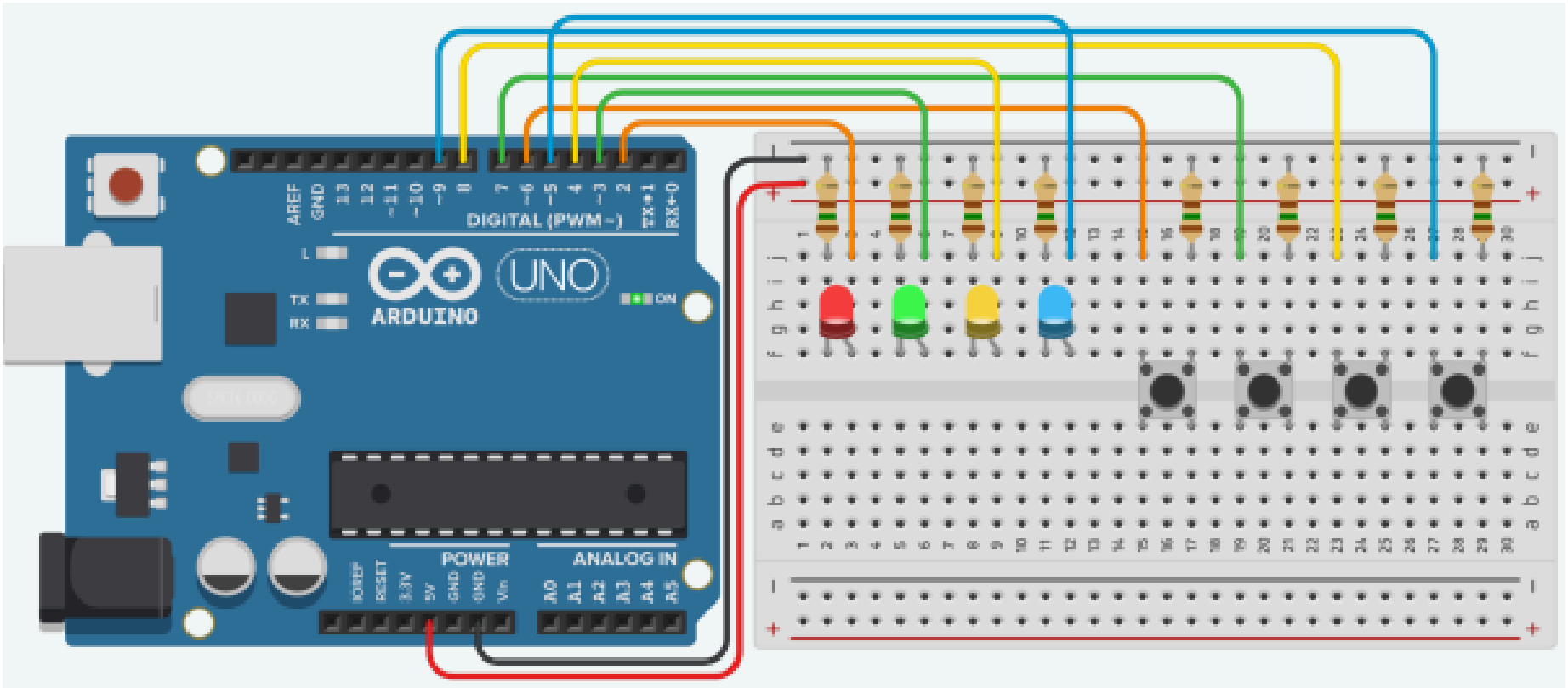
MONTAGEM:

Conectando os LEDS:

a. LED 1 (Vermelho): Pino digital 2, com resistor para o GND.
b. LED 2 (Verde): Pino digital 3, com resistor para o GND.
c. LED 3 (Amarelo): Pino digital 4, com resistor para o GND.
d. LED 4 (Azul): Pino digital 5, com resistor para o GND.

Conectando os Botões:

a. Botão 1 (Vermelho): Conectado ao pino digital 6.
b. Botão 2 (Verde): Conectado ao pino digital 7.
c. Botão 3 (Amarelo): Conectado ao pino digital 8.
d. Botão 4 (Azul): Conectado ao pino digital 9


AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5

Código:

```
// Definindo os pinos dos LEDs
const int ledPinos[] = {2, 3, 4, 5};
// Definindo os pinos dos botões
const int botaoPinos[] = {6, 7, 8, 9};

// Definindo a quantidade inicial de cores na sequência
int numCores = 2;
// Array para armazenar a sequência de cores
int sequenciaCores[10]; // Pode ajustar conforme necessário

// Função para piscar o LED vermelho indicando game over
void piscarGameOver() {
  for (int i = 0; i < 10; i++) {
    digitalWrite(ledPinos[0], HIGH);
    delay(100);
    digitalWrite(ledPinos[0], LOW);
    delay(100);
  }
}
```

```
// Função para gerar uma nova sequência de cores
void gerarNovaSequencia() {
  for (int i = 0; i < numCores; i++) {
    // Gerando número aleatório entre 0 e 3 (representando as cores)
    sequenciaCores[i] = random(4);
    // Acendendo o LED correspondente à cor gerada
    digitalWrite(ledPinos[sequenciaCores[i]], HIGH);
    delay(500); // Pode ajustar o tempo de exibição de cada cor
    digitalWrite(ledPinos[sequenciaCores[i]], LOW);
    delay(100); // Pausa entre as cores
  }
}

void setup() {
  // Configurando os pinos dos LEDs como saídas
  for (int i = 0; i < 4; i++) {
    pinMode(ledPinos[i], OUTPUT);
  }
  // Configurando os pinos dos botões como entradas
  for (int i = 0; i < 4; i++) {
    pinMode(botaoPinos[i], INPUT_PULLUP);
  }
}
```

```
void loop() {
  // Aguardando o jogador tentar a sequência atual antes de iniciar uma nova
  while (digitalRead(botaoPinos[0]) == HIGH &&
         digitalRead(botaoPinos[1]) == HIGH &&
         digitalRead(botaoPinos[2]) == HIGH &&
         digitalRead(botaoPinos[3]) == HIGH) {
    delay(100); // Aguardando jogador pressionar qualquer botão
  }

  // Resetando o jogo
  numCores = 2;
```

```
  while (true) {
    // Gerando uma nova sequência
    gerarNovaSequencia();

    // Aguardando o jogador tentar a sequência gerada
    for (int i = 0; i < numCores; i++) {
      // Aguardando o jogador pressionar o botão correspondente à cor
      while (digitalRead(botaoPinos[sequenciaCores[i]]) == HIGH) {
        delay(100); // Aguardando jogador pressionar o botão
      }
      delay(100); // Evitando leituras múltiplas do mesmo botão
      // Acendendo o LED correspondente à cor
      digitalWrite(ledPinos[sequenciaCores[i]], HIGH);
      delay(500); // Pode ajustar o tempo de exibição de cada cor
      digitalWrite(ledPinos[sequenciaCores[i]], LOW);
      delay(100); // Pausa entre as cores
    }

    // Incrementando o número de cores na próxima rodada
    numCores++;

    // Sinalizando que o jogador perdeu
    piscarGameOver();
  }
}
```

Explicando o Código

Definição dos pinos: Os LEDs e botões são conectados a pinos específicos no Arduino. Os LEDs estão nos pinos 2, 3, 4 e 5, enquanto os botões estão nos pinos 6, 7, 8 e 9.

Quantidade inicial de cores: A variável numCores armazena a quantidade inicial de cores na sequência. No início do jogo, são duas cores.

Array para armazenar a sequência de cores: O array sequenciaCores armazenará as cores geradas aleatoriamente para cada rodada. Em programação, um array é como uma série de caixas numeradas, cada uma contendo um item específico. Cada valor no array ocupa uma posição, identificada por um número chamado índice. Ao referenciar o índice desejado, podemos acessar e manipular o valor contido na posição correspondente do array. Por exemplo, se tivermos um array de números, ao usar o índice 1, acessamos o segundo número na sequência, pois o computador começa a contagem a partir do zero. Isso torna os arrays uma ferramenta poderosa para organizar e manipular conjuntos de dados de maneira eficiente em programação.

Explicando o Código

Função piscarGameOver: Essa função faz o LED vermelho piscar para indicar que o jogador perdeu.
Função gerarNovaSequencia: Essa função gera uma nova sequência de cores aleatórias. Ela utiliza um loop para atribuir valores aleatórios a cada elemento do array sequenciaCores, representando as cores dos LEDs. 6. Função setup: Configuração inicial dos pinos, definindo se são de entrada ou saída.

Explicando o Código

Loop principal (função loop):

Espera o jogador iniciar a partida: O jogo espera o jogador pressionar qualquer botão para começar.

Reinicia o jogo: Quando o jogador pressionar algum botão, o jogo é reiniciado com duas cores na sequência.

Loop do jogo principal:

■ Gera nova sequência: A função gerarNovaSequencia é chamada para criar a sequência de cores.

■ Jogador tenta a sequência gerada: O jogador deve pressionar os botões na ordem correta da sequência.

■ Exibe a sequência: Os LEDs acendem conforme a sequência gerada.

■ Incrementa a dificuldade: A cada rodada bem-sucedida, a quantidade de cores aumenta.

■ Sinaliza game over se o jogador errar: Se o jogador errar a sequência, o LED vermelho pisca para indicar game over.

Obrigada!

Alguma pergunta?

Grupo: Gabriela Ribeiro Batista
Gabrielle do Carmo Assunção
Geovana Silva Nogueira
Matheus Bragi Silva
Rayara Sousa Carvalho