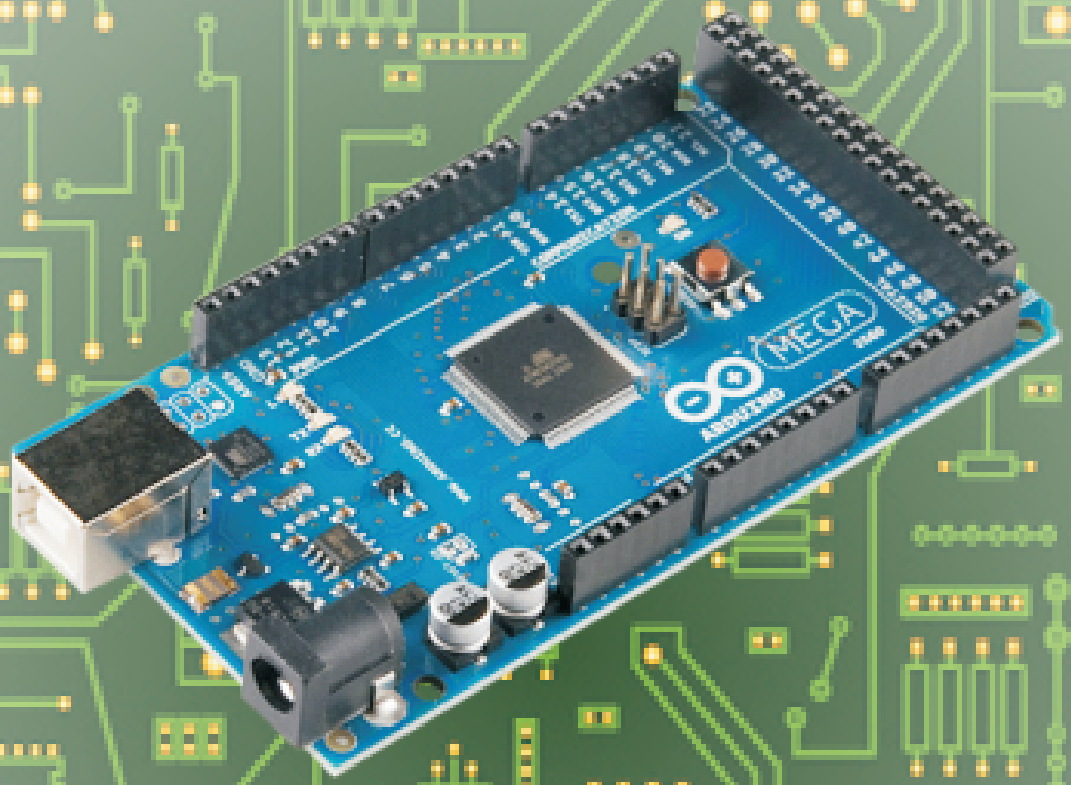
APRENDA ROBÓTICA COM O

ARDUINO



CURSO DE INTRODUÇÃO

CARGA HORÁRIA: 12 h

NOME:

Realização:
COLÉGIO SANTA EMÍLIA

O que é o Arduino?

O Arduino é um projeto totalmente aberto de protótipos de eletrônica baseados numa plataforma de hardware e software flexível e de fácil utilização. É destinado a artistas, designers, hobbyistas e qualquer tipo de pessoa interessada em criar objetos ou ambientes interativos. É um projeto que engloba software e hardware e tem como objetivo fornecer uma plataforma fácil para prototipação de projetos interativos, utilizando um microcontrolador. Ele faz parte do que chamamos de computação física: área da computação em que o software interage diretamente com o hardware, tornando possível integração com sensores, motores e outros dispositivos eletrônicos.

O Arduino pode perceber o ambiente por receber informação de uma grande variedade de sensores, e pode estimular o ambiente controlando luzes, motores, e outros atuadores.

A parte de hardware do projeto, uma placa que cabe na palma da mão, é um computador como qualquer outro: possui microprocessador, memória RAM, memória flash (para guardar o software), temporizadores, contadores, dentre outras funcionalidades. Atualmente, o projeto está na versão Uno, porém muitos Arduinos encontrados hoje são da versão Duemilanove (2009, em italiano), que possui um clock de 16 MHz, 2 Kb de memória RAM, 32 Kb de memória flash, 14 portas digitais e 6 entradas analógicas.

O microcontrolador em que se baseia (ATMEL) é programável usando a linguagem Arduino (baseada em C/C++), e também aceita código diretamente em C/C++, bem como o ambiente do Arduino que é baseado em Processing.

Normalmente, é necessário construir os circuitos para as entradas e saídas do Arduino, o que permite flexibilidade e uma grande variedade de soluções para um mesmo problema. Muitos projetos para Arduino estão disponíveis na internet (o próprio site do Arduino mantém um fórum e um blog para os usuários do sistema), facilitando o aprendizado e a troca de informações entre os construtores.

Os projetos em Arduino podem ser únicos ou podem comunicar com outros circuitos, ou até mesmo com outros softwares em um computador (por exemplo, Java, Flash,Processing, MaxMSP).

As placas podem ser montadas à mão ou serem compradas montadas, e o software pode ser obtido gratuitamente.

Como o Arduino é um projeto aberto, diversas empresas fabricam e disponibilizam suas placas no mercado, como o Freeduino, Seeduino, Roboduino, Pinguino e os brasileiros Severino e Brasuino.

Esta apostila foi baseada em diversos projetos disponíveis na internet.

São referências importantes:

www.arduino.cc

Site do projeto Arduino. Permite o download do ambiente de programação e contém referências e tutoriais para iniciantes, além de manter um fórum e um blog

www.viaspositronicas.blogspot.com

Blog mantido por diversas pessoas ligadas À robótica educacional, de várias localidades do país. Contém informações sobre diversas plataformas e campeonatos.

www.earthshineEelectronics.com

Disponibiliza para download o “The complete beginners guide to the Arduino”, uma das principais referências para iniciantes (em inglês).

Arduino Mega

Visão geral

O Arduino Mega é uma placa microcontroladora baseada em ATmega1280 (http://www.atmel.com/dyn/resources/prod_documents/doc2549.pdf). Possui 54 pinos digitais de entrada/saída (dos quais 14 podem ser utilizados como saídas PWM), 16 entradas analógicas, 4 UARTs (portas seriais de hardware), um cristal com oscilação de 16 MHz, uma conexão USB, uma entrada para fonte externa, um ICSP header, e um botão de reinício. Contém todos os elementos necessários para suportar o microcontrolador; bastando conectar a placa a um computador através de um cabo USB ou a uma fonte externa com um adaptador AC/DC ou bateria para iniciar. O Mega é compatível com muitos shields desenvolvidos para Arduino Duemilanove ou Diecimila.

Sumário

Microcontrolador	ATmega1280
Tensão de operação	5V
Tensão de entrada (recomendada)	7-12V
Tensão de entrada (limites)	6-20V
Pinos digitais (E/S)	54 (dos quais 14 suportam saída PWM)
Pinos de entradas analógicas	16
Corrente DC para pinos E/S	40 mA
Corrente DC para pino 3.3V	50 mA
Memória Flash	128 KB dos quais 4 KB são usados para bootloader
SRAM	8 KB
EEPROM	4 KB
Frequência do relógio	16 MHz

Energia

O Arduino Mega pode receber energia via conexão USB ou de uma fonte externa. A fonte é selecionada automaticamente.

Energia de uma fonte externa (não-USB) pode ser proveniente de um adaptador AC/DC ou bateria. O adaptador deve ser conectado a um plug de 2.1 mm, sendo carregado positivamente no centro. Cabos de uma bateria devem ser inseridos nos pinos Gnd e Vin do conector de energia.

A placa pode operar com uma tensão de 6 a 20 Volts. Se receber menos que 7 V, entretanto, o pino de 5 V poderá fornecer menos que cinco Volts e a placa pode ficar instável. Se utilizar mais que 12 V, o regulador de tensão pode superaquecer e danificar a placa. A faixa recomendada é entre 7 e 12 volts.

Os pinos de energia são os seguintes:

- **VIN.** Equivale à tensão de entrada da placa Arduino quando utiliza uma fonte externa (diferentemente dos 5 volts da conexão USB ou outras fontes reguladas). Você pode fornecer energia através deste pino, ou, se fornecer uma tensão através do plug, obter a mesma tensão através deste pino.
- **5V.** Fornece a tensão regulada para o microcontrolador e outros componentes da placa.
- **3V3.** Fornece uma tensão de 3.3 Volts gerada pelo chip FTDI (on-board). A corrente máxima é de 50 mA.
- **GND.** Pinos-terra.

Memória

O ATmega1280 possui 128 KB de memória flash para armazenar códigos (dos quais 4 KB são utilizados para bootloader), 8 KB de SRAM e 4 KB of EEPROM (os quais podem ser acessados utilizando a EEPROM library).

Entradas e Saídas

Cada um dos 54 pinos digitais do Mega pode ser usado como uma entrada ou saída, usando as funções pinMode (), digitalWrite (), e digitalRead (). Eles operam com 5 volts. Cada pino pode fornecer ou receber um máximo de 40 mA e tem um resistor pull-up interno (desconectado por padrão) de 20-50 kOhms. Além disso, alguns pinos têm funções especializadas:

- **Serial: 0 (RX) e 1 (TX); Serial 1: 19 (RX) e 18 (TX); Serial 2: 17 (RX) e 16 (TX); Serial 3: 15 (RX) e 14 (TX).** Usado para receber (RX) e transmitir dados (TX) TTL serial. Pinos 0 e 1 são também conectados aos pinos correspondentes do chip FTDI USB-to-TTL Serial.
- **Interrupções externas: 2 (interrupção 0), 3 (interrupção 1), 18 (interrupção 5), 19 (interrupção 4), 20 (interromper 3), e 21 (interrupção 2).** Estes pinos podem ser configurados para disparar uma interrupção por um valor baixo, uma borda de subida ou queda, ou uma alteração no valor.
- **PWM: 0 até 13.**Fornecer de 8 bits de saída PWM com a função analogWrite ().
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** Estes pinos suportam comunicação SPI, que, embora fornecida pelo hardware subjacente, não está incluída na linguagem Arduino. Os pinos SPI também são divididas sobre o cabeçalho ICSP, que é fisicamente compatível com a Duemilanove e Diecimila.
- **LED: 13.** Há um built-in LED conectado ao pino digital 13. Quando o pino está em HIGH o LED está ligado, quando o pino é LOW, ele está desligado.
- **I^2C: 20 (SDA) e 21 (SCL).** Suporte de comunicação (TWI) I2C usando a biblioteca Wire (documentação no site Fiação). Note que estes pinos não estão no mesmo local que os pinos I2C no Duemilanove ou Diecimila.

A Mega tem 16 entradas analógicas, cada uma das quais com 10 bits de resolução (isto é, 1024 valores diferentes). Por padrão, eles medem até 5 volts, embora seja possível mudar o limite superior de sua faixa usando o pino AREF e função analogReference ().

Há um par de pinos diferentes na placa:

- **AREF.** Tensão de referência para as entradas analógicas. Usado com analogReference ().
- **Reset.** Trazer esta linha LOW para resetar o microcontrolador. Tipicamente usado para adicionar um botão de reset para escudos que bloqueiam a uma no tabuleiro.

Comunicação

O Arduino Mega tem uma série de facilidades para se comunicar com um computador, outro Arduino ou outros microcontroladores. O ATmega1280 fornece quatro UARTs hardware para comunicação serial TTL (5V). Um FT232RL FTDI nos canais de bordo de uma dessas através de USB e os drivers FTDI (incluído com o software Arduino) fornece uma porta COM virtual para software no computador. O software Arduino inclui um monitor serial que permite que dados simples de texto a ser enviados de e para a placa Arduino. O RX e TX LEDs na placa pisca quando os dados estão sendo transmitidos pelo chip FTDI e conexão USB para o computador (mas não para a comunicação serial nos pinos 0 e 1).

Uma biblioteca Software Serial permite comunicação serial em qualquer um dos pinos digitais do Mega.

O ATmega1280 também suporta comunicação I2C (TWI) e SPI. O software Arduino inclui uma

biblioteca Wire para simplificar o uso do barramento I2C.

Programação

O Arduino Mega pode ser programado com o software Arduino (download).

O ATmega1280 sobre o Arduino Mega vem com um bootloader que permite envio de novos códigos sem o uso de um programador de hardware externo. Ele se comunica através do protocolo STK500 original (de referência, arquivos de cabeçalho C).

Você também pode ignorar o bootloader e programar o microcontrolador através do ICSP (In-Circuit Serial Programming) cabeçalho.

Reinício automático (software)

O Arduino Mega é projetado de uma maneira que permite que ele seja reiniciado pelo software rodando em um computador conectado. Uma das linhas de hardware de controle de fluxo (DTR) do FT232RL é conectado à linha de reset do ATmega1280 através de um capacitor de 100 nanofarad. Quando esta linha é afirmada (rebaixada), a linha de reset cai o tempo suficiente para repor o chip. O software Arduino usa esse recurso para permitir que você envie o código, simplesmente pressionando o botão de upload no ambiente Arduino. Isto significa que o bootloader pode ter um tempo mais curto, como o rebaixamento do DTR pode ser bem coordenado com o início do upload.

Essa configuração tem outras implicações. Quando a Mega é conectado a um computador rodando Mac OS X ou Linux, ele redefine a cada vez que uma conexão é feita com o software (via USB). Para o seguinte meio segundo ou menos, o bootloader está sendo executado no Mega. Embora seja programado para ignorar dados mal formados (ou seja, nada além de um upload de um novo código), que irá interceptar os primeiros bytes de dados enviados para o conselho depois que uma conexão é aberta. Se um programa rodando na placa recebe um tempo de configuração ou outros dados quando ele começa a, certifique-se que o software com o qual comunica espera um segundo depois de abrir a conexão e antes de enviar esses dados.

A Mega tem uma trilha que pode ser cortada para desabilitar o auto-reset. As almofadas de cada lado do traço pode ser soldada em conjunto para reativá-lo. Está identificada como "RESET-PT". Você também pode ser capaz de desativar o auto-reset conectando um resistor 110 ohm de 5 V para a linha de reset, veja este tópico do fórum para mais detalhes.

Proteção de sobrecorrente USB

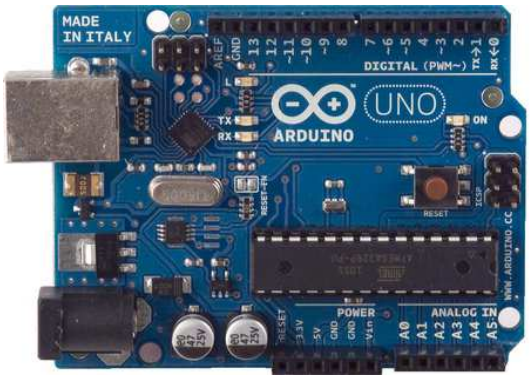
O Arduino Mega tem um polifusível reajustável que protege as portas USB do seu computador a partir de shorts e sobrecorrente. Embora a maioria dos computadores forneça sua própria proteção interna, o fusível fornece uma camada extra de proteção. Se mais de 500 mA é aplicada à porta USB, o fusível irá automaticamente quebrar a ligação até a curto ou sobrecarga seja removida.

Características Físicas e de Compatibilidade

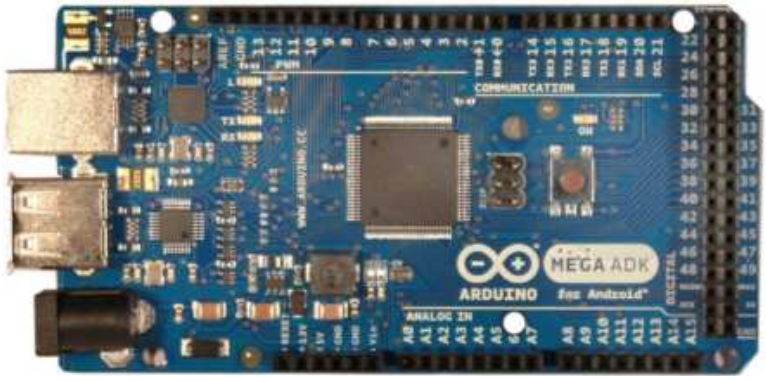
O comprimento máximo e a largura do PCB Mega são 4 e 2.1 polegadas respectivamente, com o conector USB e conector de energia, ultrapassam a dimensão anterior. Três furos permitem que a placa seja ligada a uma superfície ou case. Note que a distância entre os pinos digitais 7 e 8 é de 3 mm, e não um múltiplo do espaçamento de 2 mm dos outros pinos.

A Mega é projetado para ser compatível com a maioria dos escudos projetado para o Diecimila ou Duemilanove. Pinos digitais 0-13 (e as adjacentes AREF e pinos GND), entradas analógicas 0-5, o cabeçalho do poder, e cabeçalho ICSP estão todos em locais equivalentes. Além disso, o UART principal (porta serial) está localizado nos mesmos pinos (0 e 1), assim como as interrupções externas 0 e 1 (pinos 2 e 3, respectivamente). SPI está disponível através do cabeçalho ICSP em ambos os Mega e Duemilanove / Diecimila. Note-se que I2C não está localizado nos mesmos pinos no Mega (20 e 21) como o Duemilanove / Diecimila (entradas analógicas 4 e 5)

Outros tipos de Arduino

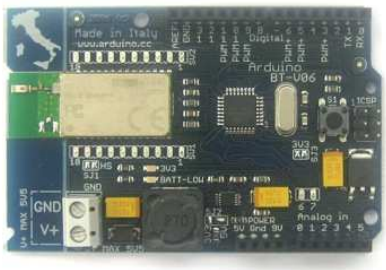


Arduino UNO



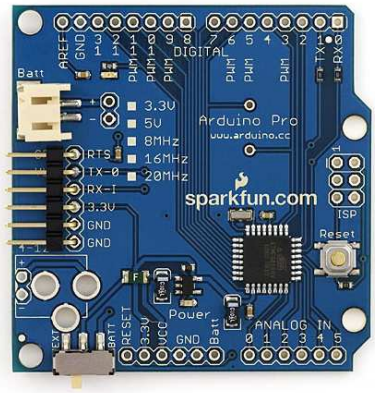
Arduino ADK

Arduino Mega 2560

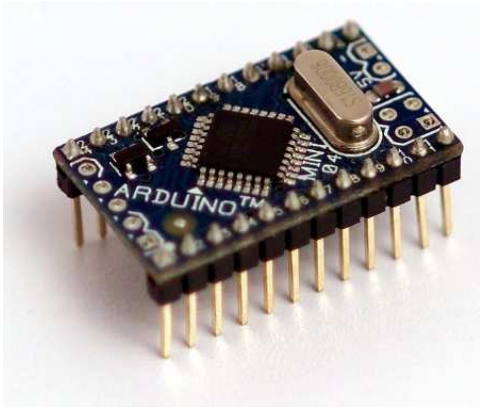


Arduino BT400

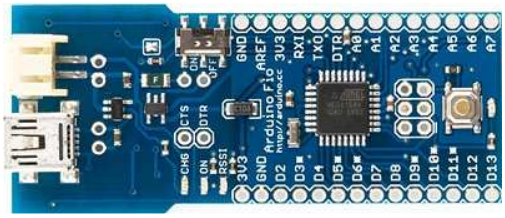
Arduino Nano



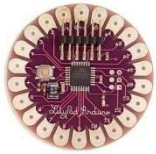
Arduino Pro



Arduino Mini



Arduino Fio



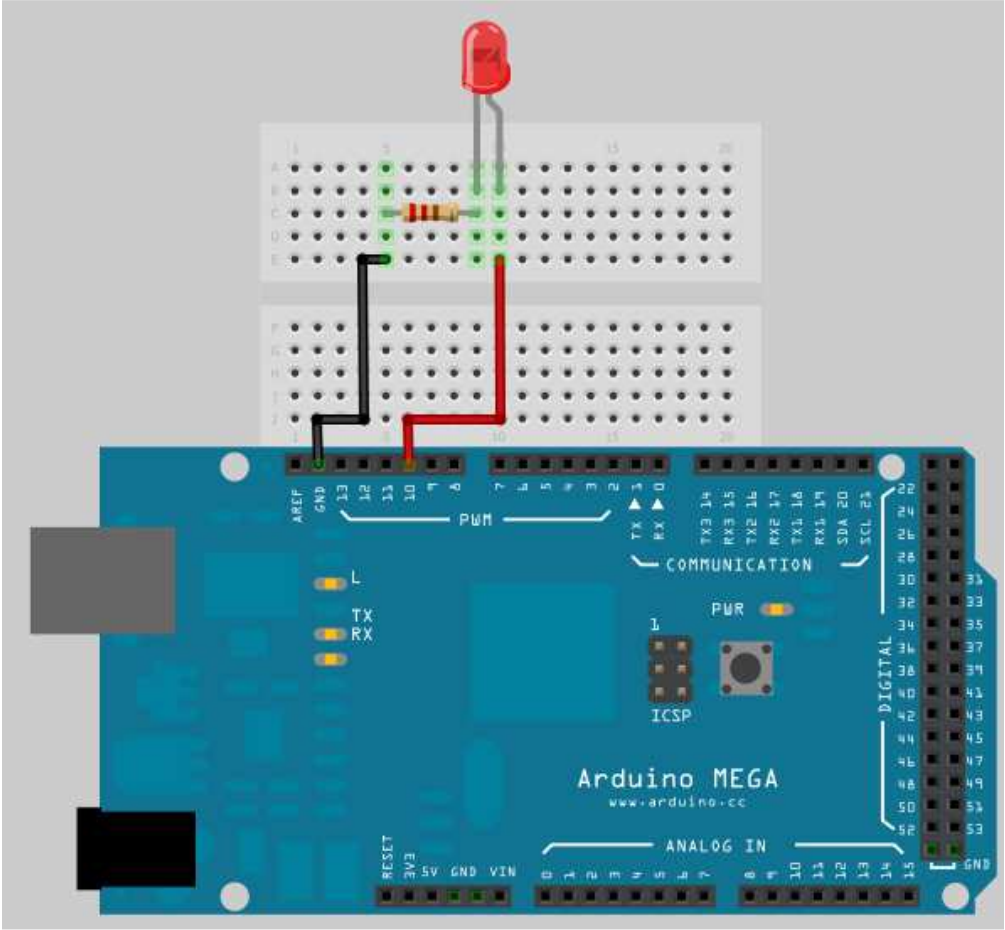
ArduinoLilyPad

01. LED Piscante

Neste projeto, você aprenderá a criar um circuito simples usando um LED e um resistor e o fará piscar.

O que você vai precisar:
01 placa Arduino Mega
02 cabos jumpers
01 resistor 220 Ω
01 LED

Montagem sugerida



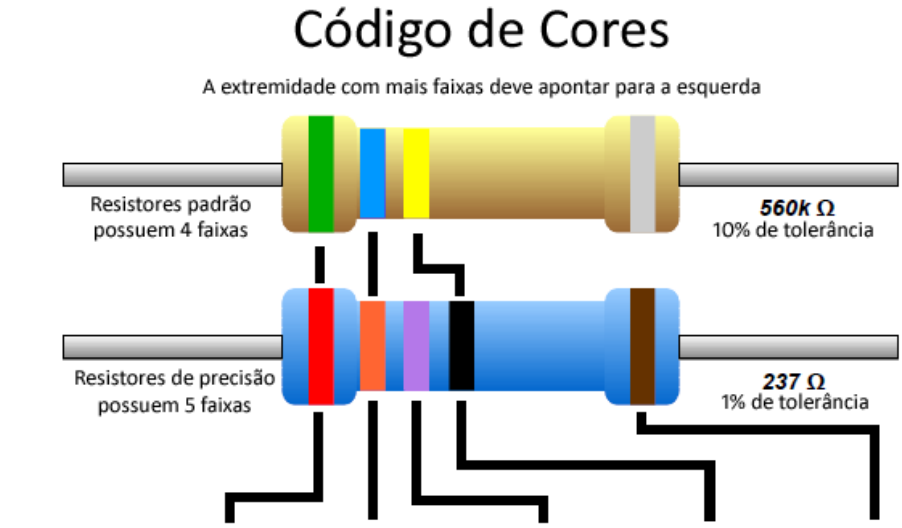
Programação sugerida

```
int ledPin = 10; // define a porta em que o LED está conectado

void setup()
{
        pinMode (ledPin, OUTPUT); // define o LED como atuador
}

//essa parte da programação faz com que ela se repita
void loop()
{
        digitalWrite (ledPin, HIGH); // liga o LED
        delay (1000);           // espera por 1000 milisegundos
        digitalWrite (ledPin, LOW);  // desliga o LED
        delay (1000);       // espera por 1000 milisegundos
}
```

Detalhando um pouco mais

Resistor

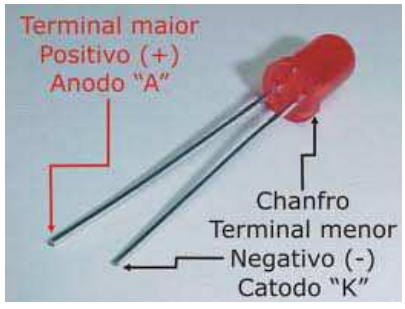
Um resistor é um dispositivo eletrônico capaz de limitar a passagem de uma corrente elétrica por um circuito elétrico. Os resistores são identificados através de um código de linhas coloridas.



Cor	1ª Faixa	2ª Faixa	3ª Faixa	Multiplicador	Tolerância
Preto	0	0	0	x 1 Ω	
Marrom	1	1	1	x 10 Ω	+/- 1%
Vermelho	2	2	2	x 100 Ω	+/- 2%
Laranja	3	3	3	x 1K Ω	
Amarelo	4	4	4	x 10K Ω	
Verde	5	5	5	x 100K Ω	+/- .5%
Azul	6	6	6	x 1M Ω	+/- .25%
Violeta	7	7	7	x 10M Ω	+/- .1%
Cinza	8	8	8		+/- .05%
Branco	9	9	9		
Dourado				x .1 Ω	+/- 5%
Prateado				x .01 Ω	+/- 10%

LED (Light Emitting Diode)

LED's são dispositivos eletrônicos que emitem luz pela passagem de uma pequena corrente elétrica. Por isso, é necessário conectar um resistor ao LED para evitar sobrecorrente. Um LED só permite passagem de luz em um sentido. Portanto, é necessário identificar os pólos para conectar o LED corretamente dentro do circuito eletrônico.



Alterando o código

1 – Altere o intervalo de ligar e desligar o LED para um valor definido por você.
2 – Altere o valor do delay para 50 milisegundos. O LED estará desligando e acendendo?

02. Sinalizador de Código Morse

Neste projeto nós vamos fazer o mesmo circuito do projeto anterior, mas usaremos alguns códigos diferentes para fazer o LED transmitir uma mensagem em Código Morse. Nesse caso, vamos fazer o LED sinalizar as letras S.O.S., que é o sinal mais conhecido deste código (foi utilizado pelos tripulantes do Titanic para transmitir seu pedido de socorro).

(SOS: A letra “S” consiste de três sinais breves e letra “O”, de três sinais longos)

O que você vai precisar saber:

- **Sinais de comparação**
  == (semelhante a)
  != (diferente de)
  < (menor que)
  > (maior que)
  <= (menor ou igual a)
  >= (maior ou igual a)

Programação Sugerida

```
int ledPin = 10; // LED conectado ao pino 10

void setup()
{
	pinMode (ledPin, OUTPUT); // define a porta digital como output
}

void loop()
{
	for (int x=0; x<3; x++)
	{
		digitalWrite (ledPin, HIGH); // liga o LED
		delay (150); // espera por 150 milisegundos
		digitalWrite (ledPin, LOW); // desliga o LED
		delay (100); // espera por 100 milisegundos
	}

	delay (200);// intervalo de 200 milisegundos para fazer a pausa entre as letras
	for (int x=0; x<3; x++)
	{
		digitalWrite (ledPin, HIGH); // liga o LED
		delay (400); // espera por 400 milisegundos
		digitalWrite (ledPin, LOW); // desliga o LED
		delay (100); // espera por 100 milisegundos
	}
	delay (200);// intervalo de 200 milisegundos para fazer a pausa entre as letras
	for (int x=0; x<3; x++)
	{
		digitalWrite (ledPin, HIGH); // liga o LED
		delay (150); // espera por 150 milisegundos
		digitalWrite (ledPin, LOW); // desliga o LED
		delay (100); // espera por 100 milisegundos
	}
	delay (5000); // espera por 5 segundos para repetir a programação
}
```

Detalhando um pouco mais

A função *for*

A função *for* estabelece um loop enquanto um teste de comparação for verdadeiro.
Observe a declaração da função no programa trabalhado:

```
for (int x=0; x<3; x++)
        {
    comando 1;
    comando 2;
                }
```

A função *for* segue o esquema geral:

for (declara variável; condição de comparação; operação sobre a variável)

No exemplo acima, uma variável *x* é declarada como um número inteiro e é atribuído o valor 0, inicialmente. Em seguida, testa-se se o valor de *x* é menor que *3*. Caso o teste resulte verdadeiro, são executados os comandos 1 e 2 e, no final, acrescenta-se uma unidade à variável *x* (*x*++).

Alterando o código

1 – Escreva uma palavra em código Morse, de acordo com os dados em tabela.

A	·−	J	·−−−	S	···	2	··−−−
B	−···	K	−·−	T	−	3	···−−
C	−·−·	L	·−··	U	··−	4	····−
D	−··	M	−−	V	···−	5	·····
E	·	N	−·	W	·−−	6	−····
F	··−·	O	−−−	X	−··−	7	−−···
G	−−·	P	·−−·	Y	−·−−	8	−−−··
H	····	Q	−−·−	Z	−−··	9	−−−−·
I	··	R	·−·	1	·−−−−	0	−−−−−

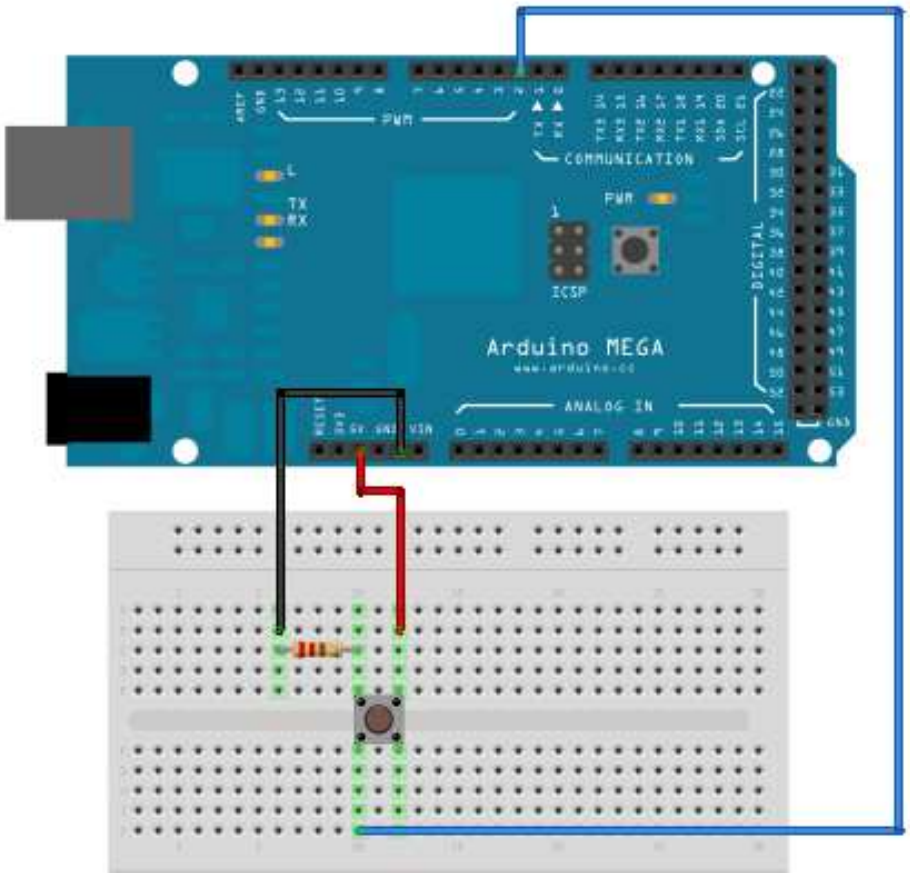
03. Leitura de um sinal digital no computador

Neste projeto, você deve construir um circuito que possibilite a leitura do estado de um interruptor (ligado ou desligado), enviando-o através de uma comunicação serial ao computador.

A comunicação serial no computador é vista em uma tela à parte, que pode ser acessada pelo atalho Ctrl+Shift+M (Serial monitor).

O que você vai precisar:
01 placa Arduino Mega
03 cabos jumpers
01 interruptor
01 resistor 220 Ω

Montagem sugerida



Programação sugerida

```
void setup()
{
  Serial.begin (9600); // inicia a comunicação serial
  pinMode (2, INPUT); // define o pino 2 como entrada de dados
}

void loop()
{
  int sensorValue = digitalRead(2); // define variável e armazena a leitura do sensor
  Serial.println (sensorValue, DEC); // envia valor para o serial monitor
  delay (1000);
}
```

Explorando a programação

A linha *Serial.begin (9600);* inicia a leitura serial, a 9600 bauds.

A linha *Serial.println (sensorValue, DEC);* envia o valor da leitura do sensor para o monitor serial e salta linha ao final. Para que as entradas fossem publicadas em uma mesma linha, deve-se utilizar o comando *Serial.print* em vez de *Serial.println*.

Neste projeto, o monitor serial deverá exibir 0 quando o sensor estiver solto e 1, quando estiver pressionado.

Baud deriva do sobrenome de J.M.E. Baudot, francês inventor do código telegráfico Baudot. Um **Baud** é uma medida de **velocidade de sinalização** e representa o número de mudanças na linha de transmissão (seja em frequência, amplitude, fase etc...) ou eventos por segundo. Para se determinar a taxa de transmissão de um canal em bits por segundo - bps, deve ser levado em consideração o tipo de **codificação** utilizada, além da velocidade de sinalização do canal de comunicação.

Fonte: http://pt.wikipedia.org/wiki/Baud

Alterando o código

1 – Adicione um LED à porta 2 do Arduino Mega e altere o código para que o LED acenda quando o botão do interruptor for pressionado.

2 – Observe o seguinte código:

```
char nome[] = "ARDUINO";

void setup()
{
  Serial.begin (9600);
}

void loop()
{
  for (int x=0; x < 8; x++)
  {
    Serial.print (nome[x]);
    delay (500);
  }
}
```

observe a diferença entre os comandos *Serial.print* e *Serial.println*

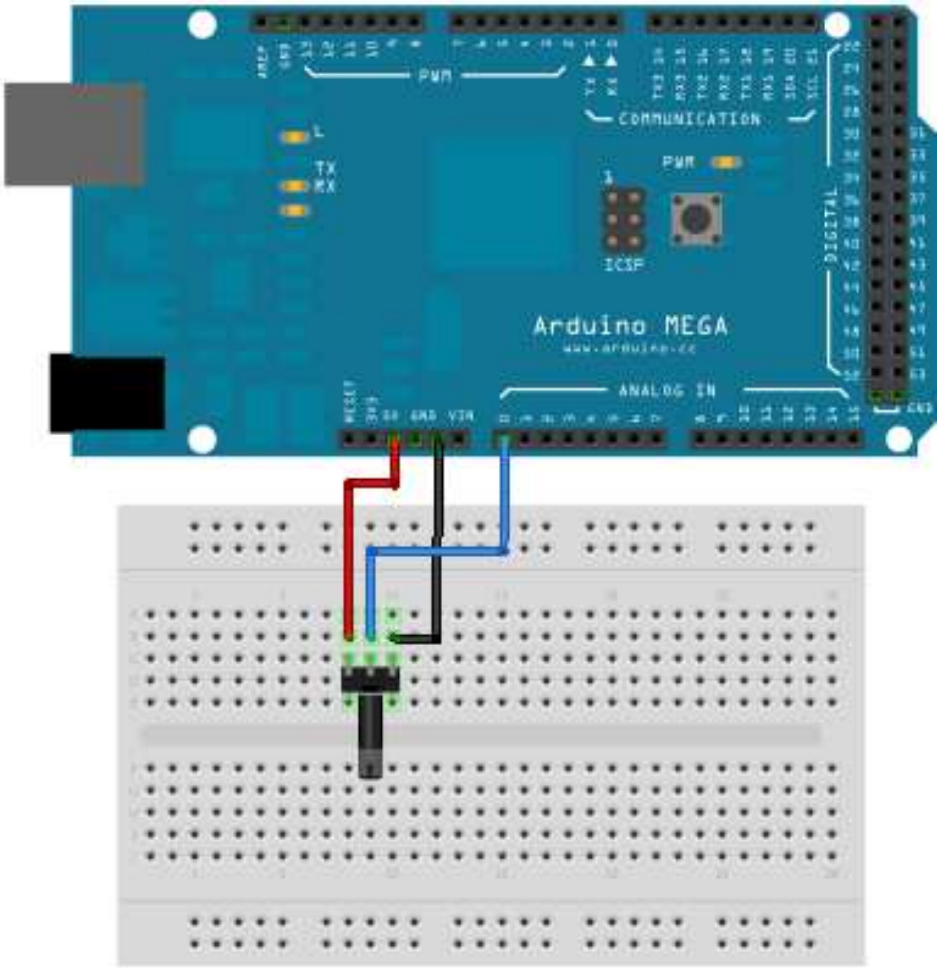
04. Leitura de um sinal analógico no computador

Neste projeto, você deve construir um circuito que possibilite a leitura de um valor analógico (0 a 1023) fornecido por um potenciômetro, enviando-o através de uma comunicação serial ao computador.

A comunicação serial no computador é vista em uma tela à parte, que pode ser acessada pelo atalho Ctrl+Shift+M (Serial monitor).

O que você vai precisar:
01 placa Arduino Mega
03 cabos jumpers
01 potenciômetro

Montagem sugerida



Programação sugerida

```
void setup()
{
  Serial.begin (9600);
}
void loop()
{
  int sensorValue = analogRead (A0);
  Serial.println (sensorValue, DEC);
  delay (1000);
}
```

Explorando a programação

A linha *intsensorValue = analogRead(A0);* define a variável sensorValue e armazena o valor da porta analógica 0 na variável sensorValue

Ao girar o potenciômetro, o novo valor do sinal analógico será exibido em tela.

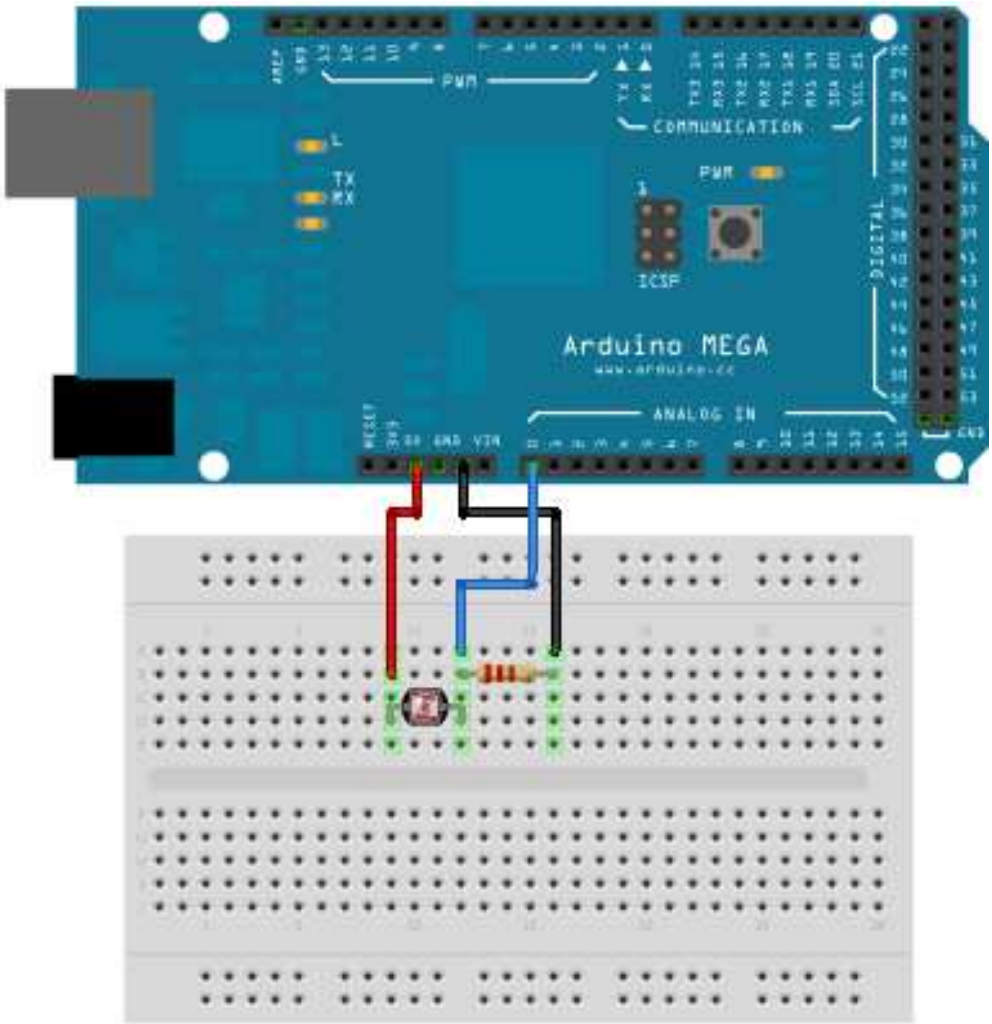
Alterando o código

Altere o código para que além do valor analógico, seja exibido o valor da resistência, em ohms. No final, ajuste o potenciômetro para que ele exerça resistência de 220 Ω. (*dica: determine a relação entre a resistência e uma unidade analógica – lembre-se de que o Arduino utiliza 1024 unidades analógicas*).

Alterando a montagem

Substitua o potenciômetro por um LDR (*Light Dependent Resistor*) e efetue medidas da luminosidade ambiente.

A montagem deve ficar com o seguinte aspecto:



Detalhando um pouco mais

LDR (Light Dependent Resistence)

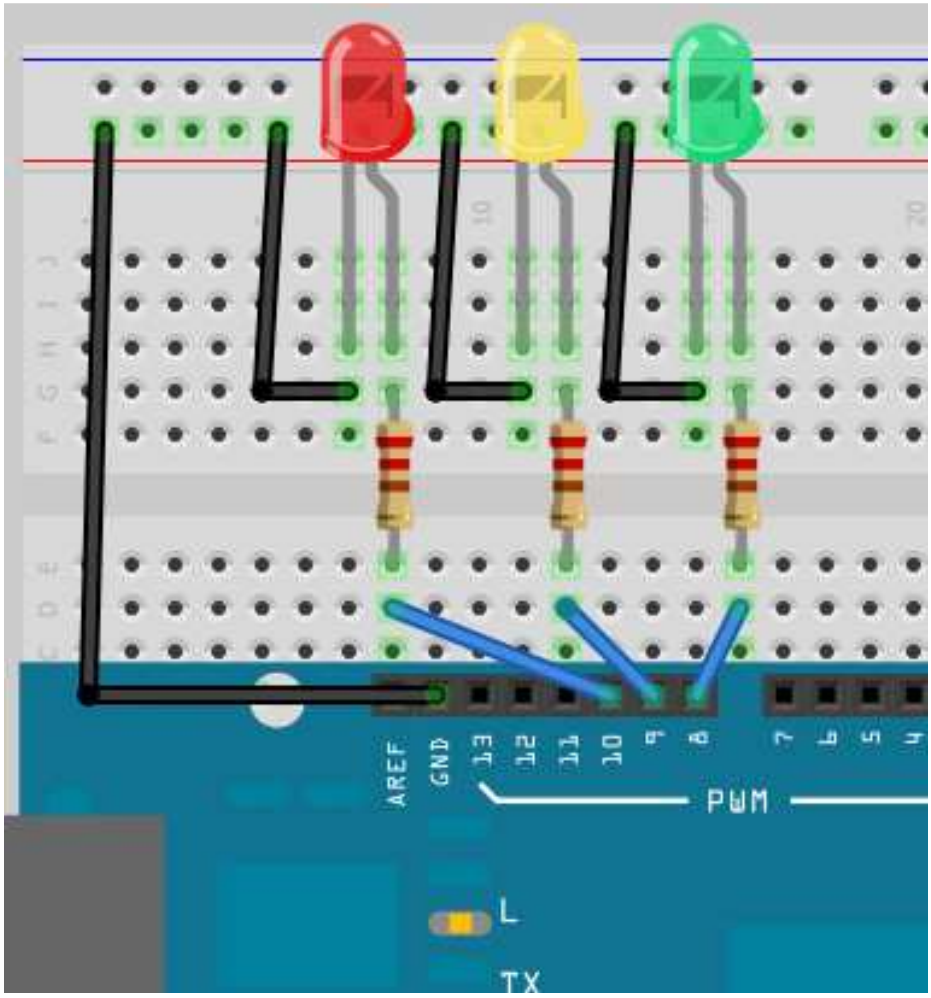
Dispositivo eletrônico que apresenta como uma resistência variável, cujo valor depende da luz incidente. É necessário conectá-lo a um resistor para que possa atuar como sensor analógico.

05. Semáforo

Agora nós iremos criar um circuito para simular um semáforo de trânsito. O semáforo será constituído por três LED's: um vermelho, um amarelo e um verde.

O que você vai precisar:
01 Placa Arduino Mega
01 Protoboard
01 LED Vermelho
01 LED Amarelo
01 LED Verde
03 Resistores 220 Ω

Montagem Sugerida



Programação Sugerida

```
int ledDelay = 10000; // espera entre as trocas de cores
int vermelho = 10;
int amarelo = 9;
int verde = 8;
void setup()
{
        pinMode (vermelho, OUTPUT);
        pinMode (amarelo, OUTPUT);
        pinMode (verde, OUTPUT);
}
```

```
void loop()
{
        digitalWrite (vermelho, HIGH); // liga o led vermelho
        delay (ledDelay); // espera o tempo determinado na variável “ledDelay”
        digitalWrite (amarelo, HIGH); // liga o led amarelo
        delay (2000); // espera 2 segundos
        digitalWrite (verde, HIGH); // liga o led verde
        digitalWrite (vermelho, LOW); // desliga o led vermelho
        digitalWrite (amarelo, LOW); // desliga o led amarelo
        delay (ledDelay); // espera o tempo determinado na variável “ledDelay”
        digitalWrite (amarelo, HIGH); // liga o led amarelo
        digitalWrite (verde, LOW); // desliga o led verde
        delay (2000); // espera 2 segundos
        digitalWrite (amarelo, LOW); // desliga o led amarelo
}
```

Alterando o código

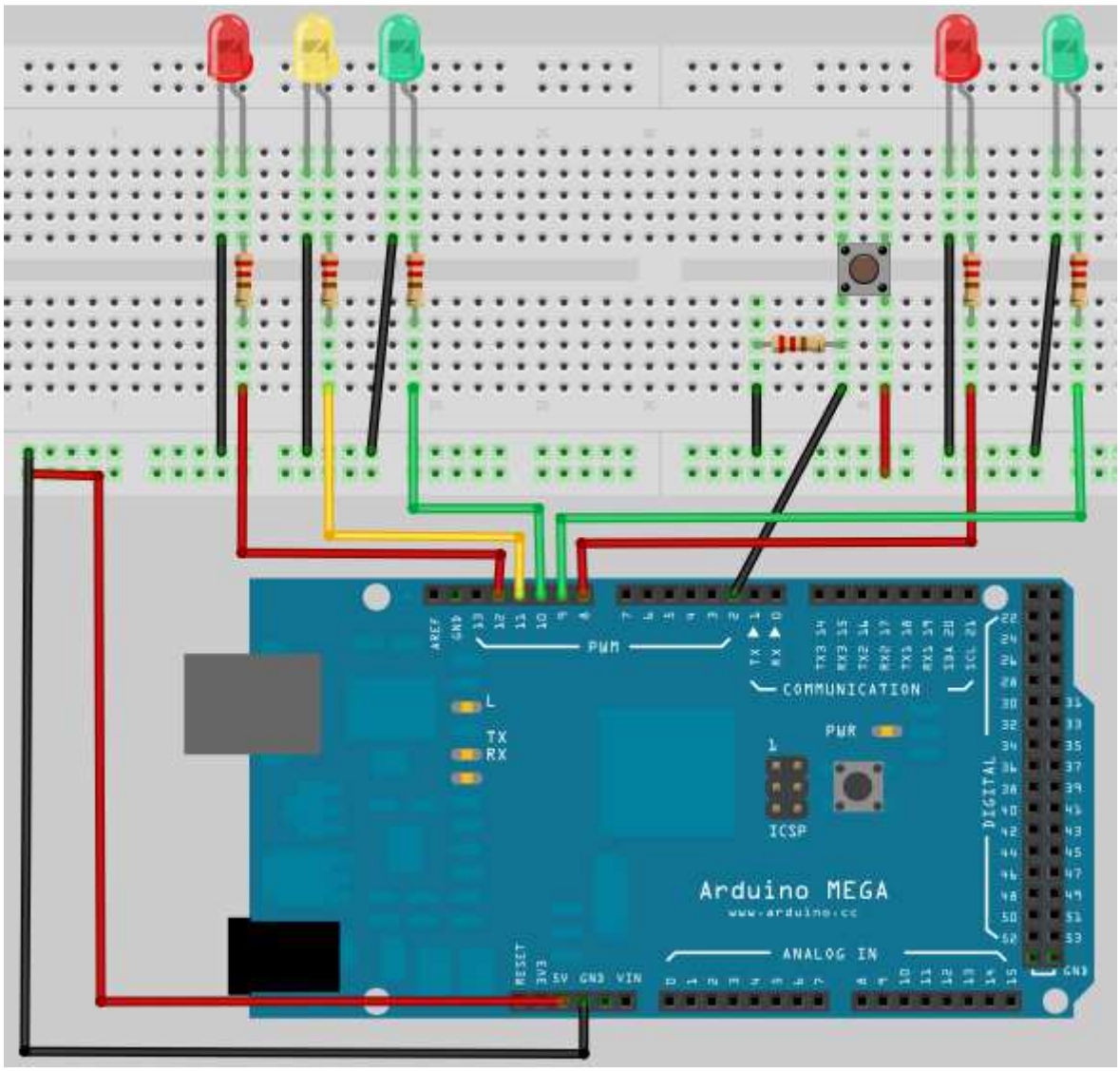
1 – Você deve ter percebido que o código do semáforo é ligeiramente diferente do código utilizado nos semáforos brasileiros. Adapte o código para torná-lo semelhante à programação de nossos semáforos.

06. Semáforo Interativo

Este projeto é uma extensão do projeto anterior, onde iremos incluir um semáforo para pedestres e um botão para solicitar a parada dos carros.

O que você vai precisar:
02 LED's vermelhos
01 LED amarelo
02 LED's verdes
06 Resistores de 220 Ω
01 Interruptor

Montagem sugerida



Programação sugerida

```
int carroVermelho = 12;
int carroAmarelo = 11;
int carroVerde = 10;
int pedVermelho = 8;
int pedVerde = 9;
int botao = 2; // pino do interruptor
int tempoTravessia = 5000; // tempo para atravessar a rua
```

```
unsigned long changeTime;

void setup()
{
	pinMode (carroVermelho, OUTPUT);
	pinMode (carroAmarelo, OUTPUT);
	pinMode (carroVerde, OUTPUT);
	pinMode (pedVermelho, OUTPUT);
	pinMode (pedVerde, OUTPUT);
	pinMode (botao, INPUT);
	// liga a luz verde dos carros e a vermelha para os pedestres
	digitalWrite (carroVerde, HIGH);
	digitalWrite (pedVermelho, HIGH);
}

void loop()
{
	int state = digitalRead (botao);
	if (state == HIGH && (millis() - changeTime) > 5000)
	{
	// Ativa a função para mudar as luzes
	changeLights(); // executa o bloco changeLights()
	}
}

void changeLights()
{
	digitalWrite (carroVerde, LOW); // desliga o verde
	digitalWrite (carroAmarelo, HIGH); // liga o amarelo
	delay (2000); // espera 2 segundos
	digitalWrite (carroAmarelo, LOW); // desliga o amarelo
	digitalWrite (carroVermelho, HIGH); // liga o vermelho
	delay (1000); // espera 1 segundo
	digitalWrite (pedVermelho, LOW); // desliga o vermelho do pedestre
	digitalWrite (pedVerde, HIGH); // liga o verde do pedestre
	delay (tempoTravessia);

	for (int x=0; x<10; x++)
	{
		digitalWrite (pedVerde, HIGH); // liga o verde do pedestre
		delay (250); // espera 250 milisegundos
		digitalWrite (pedVerde, LOW); // desliga o verde do pedestre
		delay (250); // espera 250 milisegundos
	}
	digitalWrite (pedVermelho, HIGH); // liga o sinal vermelho do pedestre
	delay (500); // espera 500 milisegundos
	digitalWrite (carroAmarelo, HIGH); // liga o vermelho
	digitalWrite (carroVermelho, LOW); // desliga o vermelho
	delay (1000); // espera 1 segundo
	digitalWrite (carroVerde, HIGH); // liga o verde
	digitalWrite (carroAmarelo, LOW); // desliga o amarelo
	changeTime = millis(); // registra o tempo desde a última alteração de luzes
}
```

Detalhando um pouco mais

millis()

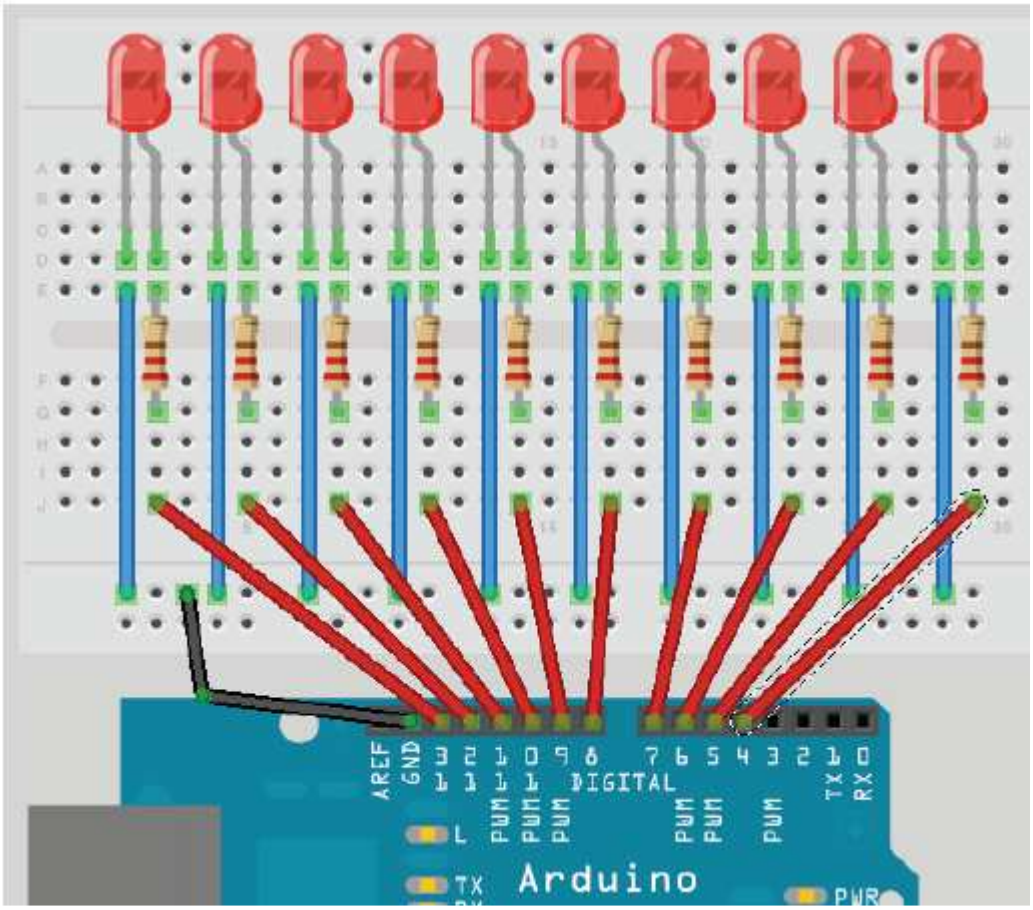
Função que retorna o intervalo de tempo, em milisegundos, decorrido desde o início do programa em execução.

07. Efeito Perseguição

Neste projeto, nós vamos usar uma série de LED's (10, no total) para criar um efeito de perseguição introduzindo o conceito de matrizes.

O que você vai precisar
01 Arduino Mega
01 placa protoboard
10 LED's vermelhos
10 Resistores 220 Ω
21 cabos jumpers

Montagem Sugerida



Programação Sugerida

```
byte ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13}; // define uma matriz
int ledDelay(65); // intervalo entre as trocas
int direction = 1;
int currentLED = 0;
unsigned long changeTime;

void setup()
{
        // define todas as portas como saída
        for (int x=0; x<10; x++)
        {
                pinMode (ledPin[x], OUTPUT);
        }
        changeTime = millis();
}
```

```
void loop()
{
	if ((millis() - changeTime) > ledDelay)
	{
		changeLED();
		changeTime = millis();
	}
}

void changeLED()
{
	// desliga todos os LED's
	for (int x=0; x<10; x++)
	{
		digitalWrite (ledPin[x], LOW);
}

digitalWrite (ledPin[currentLED], HIGH); // desliga o LED atual
currentLED += direction; // incremento pelo valor da duração
// muda a direção se chegar ao fim
if (currentLED == 9)
{
	direction = -1;
}

if (currentLED == 0)
{
	direction = 1;
}
}
```

Detalhando um pouco mais

O conceito de matriz

Uma matriz (array, em inglês) é um conjunto de variáveis que podem ser acessadas através de um índice. As matrizes podem ser declaradas de vários modos:

```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};
char message[6] = "hello";
```

O número entre colchetes indica o número de elementos da matriz. Lembre-se que o primeiro elemento é acessado pelo índice 0.

Alterando o código

1 – Altere o código para que os LED's acendam alternadamente (números pares e números ímpares).

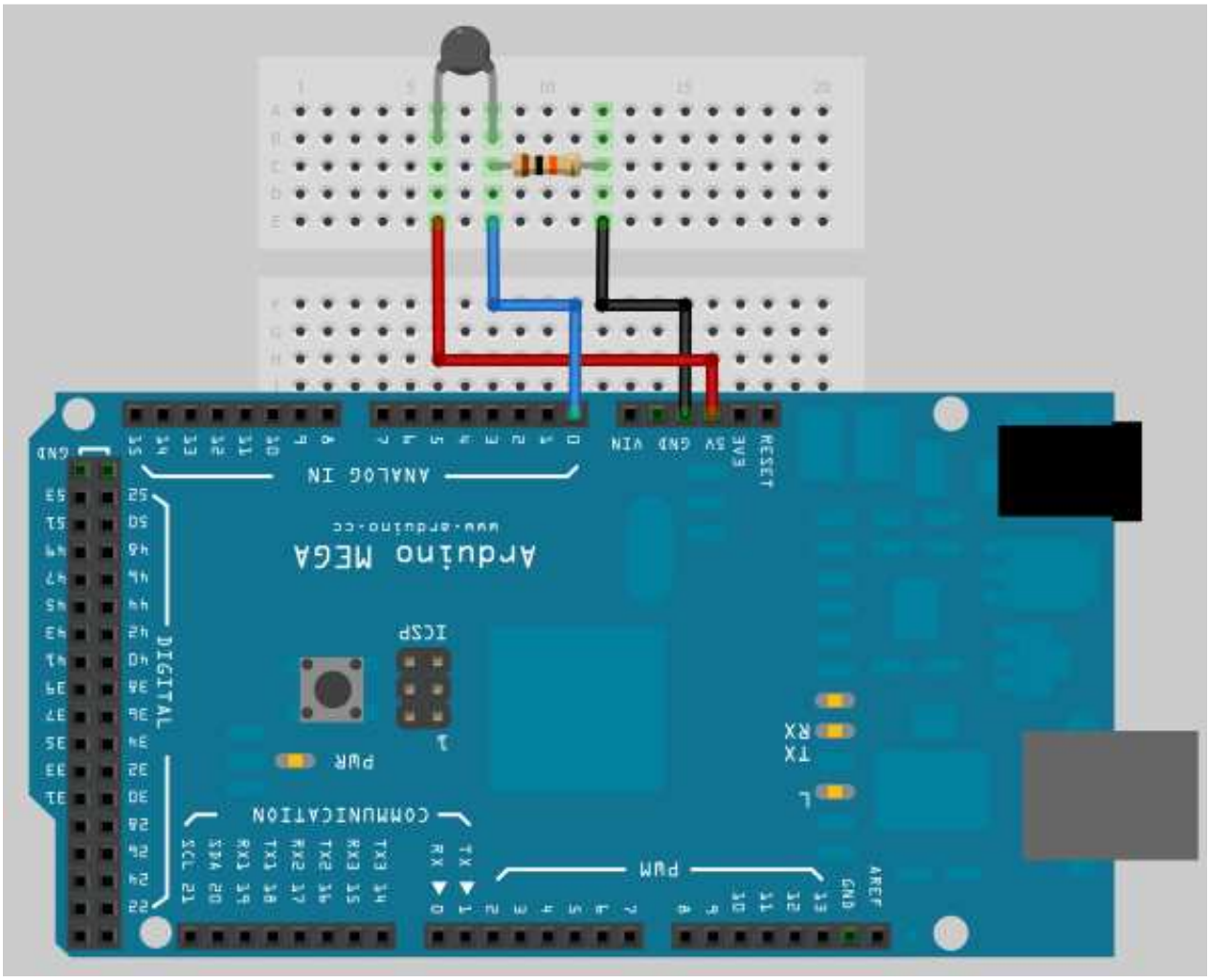
08. Usando um termistor para calcular a temperatura

Algumas aplicações industriais necessitam da leitura da temperatura para manterem seus processos. O Arduino permite fazer a leitura da temperatura através de um termistor conectado a uma porta analógica.

O que você vai precisar

01 Arduino Mega
01 protoboard
01 termistor NTC 10 KΩ
01 resistor 10 KΩ
03 cabos jumpers

Montagem sugerida:



Programação sugerida

```
#include <math.h>

double Termistor (int RawADC)
{
double Temp;
Temp = log(((10240000/RawADC) - 10000)); // Considerando resistência de 10K
Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp ))* Temp );
 // Equação de Steinhart-Hart para o termistor - temperatura em Kelvin
```

```
Temp = Temp - 273.15;           // Converte Kelvin para Celsius
return Temp;
}

void setup()
{
Serial.begin (115200);
}

void loop ()
{
Serial.print(int (Termistor (analogRead (0))));  // Exibe temperatura em Celsius
Serial.println (" graus Celsius");
delay (1000);
}
```

Explorando o código

Bibliotecas

Bibliotecas são conjuntos de dados pré-formatados que possibilitam funcionalidade extra a um programa (por exemplo, funções matemáticas mais complexas). Para incluir uma biblioteca, deve-se utilizar a palavra-chave *#include*.

Exemplo:

#include <math.h>

As bibliotecas para o Arduino podem ser encontradas em *http://arduino.cc/en/Reference/Libraries*

Double

Função que dobra a precisão do ponto decimal, quando se trabalha com outras plataformas. No código do Arduino, possui exatamente a mesma função que *float*.

Return

Finaliza a função, retornando o valor designado.

Detalhando um pouco mais

Termistor

Um termistor é um controlador térmico, que consiste numa resistência sensível cuja função principal é exibir uma mudança grande, previsível e precisa em resistência elétrica quando um equipamento ou produto sofrer uma mudança na temperatura de corpo. Um termistor de Coeficiente de Temperatura Negativo (**NTC**) (Negative Temperature Coefficient) exibe uma diminuição em resistência elétrica quando submetido a um aumento de temperatura do equipamento e um termistor de Coeficiente de Temperatura Positivo **(PTC**) (Positive Temperature Coefficient) exibe um aumento em resistência elétrica existe um aumento da temperatura do equipamento.

Alterando o código

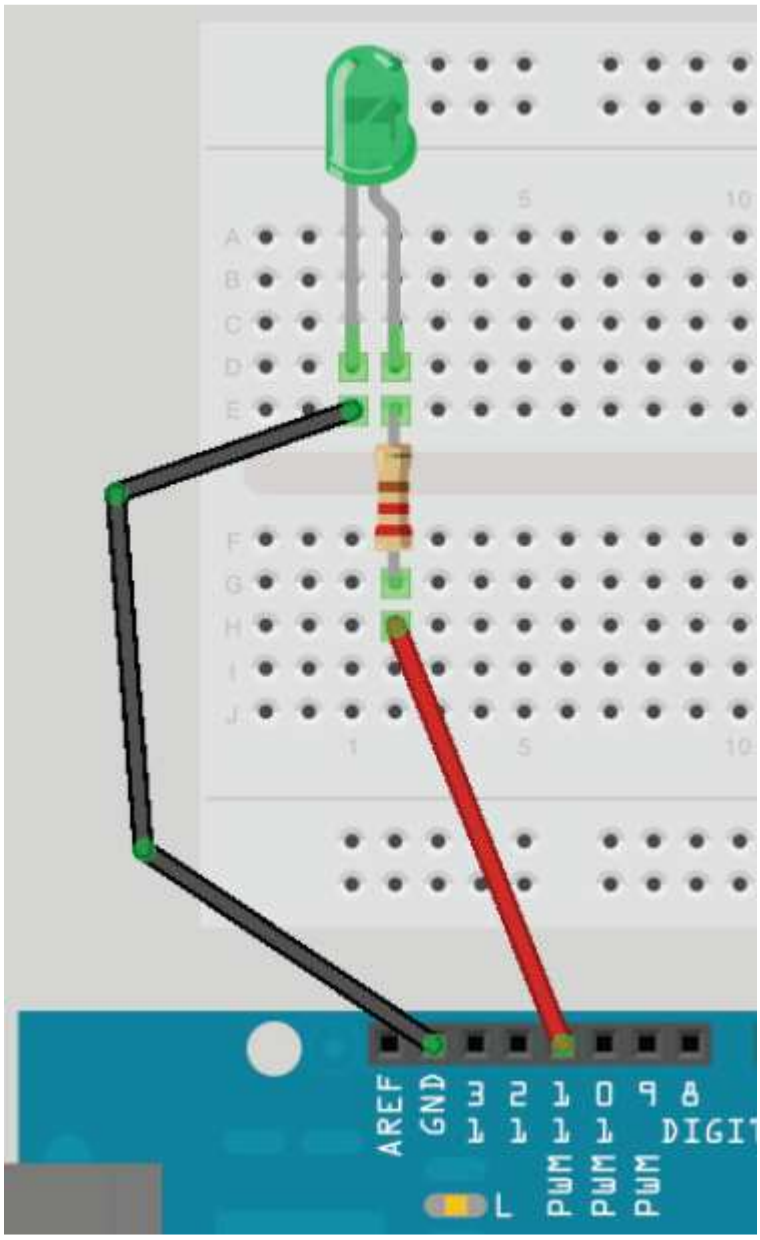
1 – Adicione um LED ao circuito e altere a programação para que o LED acenda quando a leitura da temperatura indicar 30 ºC.

09. Um LED pulsante

Neste projeto, o brilho de um LED irá variar de acordo com uma onda senoidal, a partir do controle de uma porta de saída de pulso modulado (PWM).

O que você vai precisar
01 Arduino Mega
01 placa protoboard
01 LED
01 resistor de 220 Ω
02 cabos jumpers

Montagem sugerida



Programaçãosugerida

```
int ledPin = 11;
float sinVal;
int ledVal;

void setup()
{
        pinMode (ledPin, OUTPUT);
```

```
}

void loop()
{
        for (int x=0; x<180; x++)
        {
                sinVal = (sin(x*(3.1415/180))); // convertegrausa radianos e obtém o valor do seno
                ledVal = int(sinVal*255);
                analogWrite (ledPin, ledVal); // define a saída PWM
                delay(25);
        }
}
```

Alterando o código

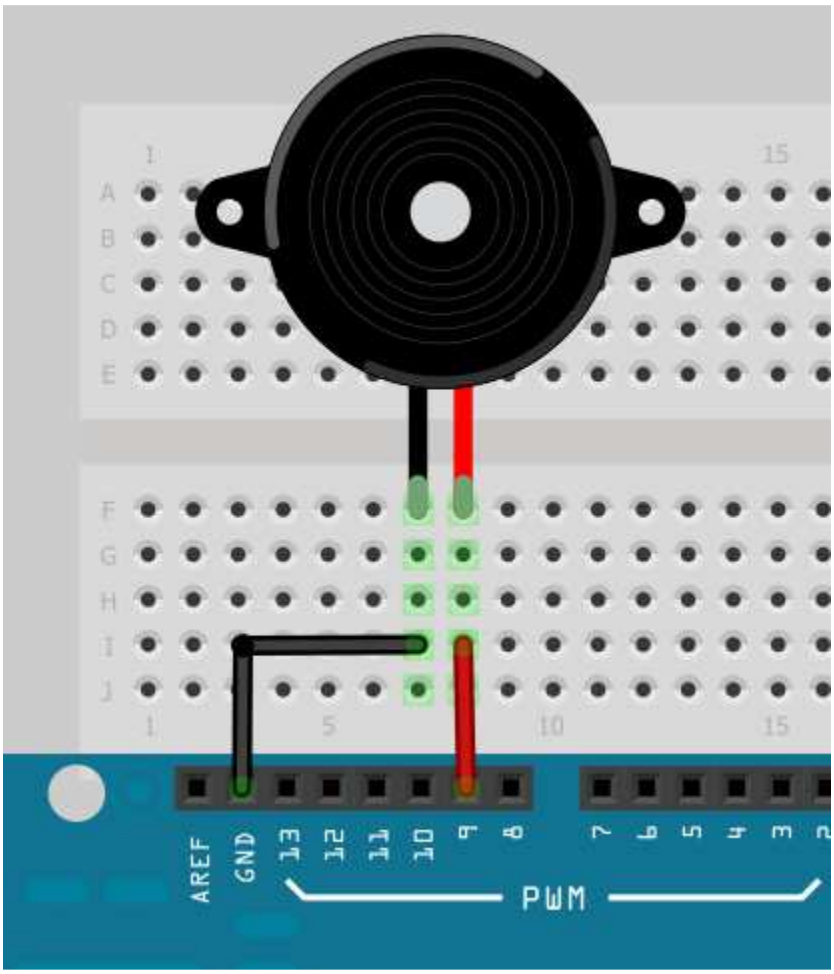
1 – Insira um potenciômetro no circuito e o conecte a uma porta analógica. Em seguida, altere seu código para que, a partir da rotação do potenciômetro, o brilho do LED seja alterado.

10. Tocando uma melodia

"Brilha, brilha, estrelinha..."

O que você vai precisar
01 Arduino Mega
01 placa protoboard
01 piezo elemento

Montagem sugerida



Programação sugerida

```
int speakerPin = 9;
int length = 15; // define o número de notas
char notes[] = "ccggaagffeeddc "; // o espaço representa uma pausa
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 }; // define o tempo das notas
int tempo = 300;

void playTone (int tone, int duration)
{
        for (long i = 0; i < duration * 1000; i += tone * 2)
        {
                digitalWrite (speakerPin, HIGH);
                delayMicroseconds (tone);
                digitalWrite (speakerPin, LOW);
                delayMicroseconds (tone);
        }
}

void playNote (char note, int duration)
{
```

```
	char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
	int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };// toca o som correspondente à nota
	for (int i = 0; i < 8; i++)
	{
		if (names[i] == note)
		{
			playTone (tones[i], duration);
		}
	}
}

void setup()
{
	pinMode (speakerPin, OUTPUT);
}

void loop()
{
	for (int i = 0; i < length; i++)
	{
		if (notes[i] == ' ')
		{
			delay(beats[i] * tempo); // pausa
		}
		else
		{
			playNote (notes[i], beats[i] * tempo);
		}
		delay (tempo / 2); // pausa entre as notas

	}
}
```