Robótica e Sistemas Digitais.

Apostila Arduino Básico

Vol.2

www.vidadesilicio.com.br

Estamos escrevendo as linhas do futuro, da mudança e da evolução.

A tecnologia está mudando a forma de o mundo enfrentar as dificuldades. Cada vez mais as pessoas se conectam através da internet para compartilhar conhecimentos e experiências, expandindo, de forma exponencial, a inteligência coletiva. A robótica assume posição privilegiada nessa evolução e queremos fazer parte dessa história.

Esperamos que goste dessa apostila, ela foi feita com carinho pensando em você. Divirtam-se!!

Atenciosamente,

Equipe Vida de Silício

Sumário

1. Introdução

1.1. Sistemas autônomos

Quando pensamos em um sistema autônomo, pensamos em um sistema que faça a tarefa designada sozinho, sem a necessidade de interferência humana. Para isso, nosso sistema deve ser dotado de alguma inteligência, tal como um computador ou sistema artificial. Em nosso caso, iremos usar nosso Arduino como o cérebro de nosso sistema.

Para compreender como essa inteligência funciona é interessante entendermos um pouco sobre sistemas de controle.

1.2. Sistemas de controle

No nosso cotidiano precisamos desempenhar várias funções que dependem de diversas variáveis. Por exemplo, em nossas casas, temos que controlar a temperatura da geladeira, do ar condicionado e do chuveiro. Em nossos carros, temos que controlar a velocidade, o nível de combustível e o nível de óleo. Na aviação, precisamos controlar a altitude, velocidade e pressão do avião. Em processos industriais, devemos controlar a vazão, nível e temperatura. Para controlar essas variáveis utilizamos sistemas de controle.

Imagine que você tenha que ir de bicicleta de uma cidade a outra o mais rápido possível. Para isso, você irá precisar controlar sua velocidade, mas também sua energia para que não se canse demais antes de chegar ao seu destino. Portanto, você precisará gerenciar seu consumo de energia de forma estratégica. Nisso, com o intuito de alcançarmos um objetivo, usamos um sistema de controle.

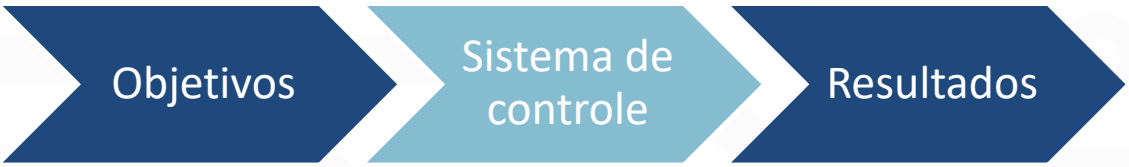
Com o avanço tecnológico, sistemas de controle usando dispositivos eletrônicos estão sendo cada vez mais comuns. Sua geladeira controla a temperatura dos alimentos de forma automática, basta você dizer qual temperatura que você deseja e ela fará o serviço pesado para você.

1.2.1. Componentes básicos de um sistema de controle

Os principais ingredientes de um sistema de controle são:

a. O objetivo do controle: Precisamos saber o que queremos fazer, quais resultados esperamos, ou seja, um motivo para criar nosso sistema de controle;
b. Componentes do sistema de controle: O sistema de controle possui componentes que serão responsáveis por produzir o resultado esperado;

c. Resultados ou saídas: É o produto de nosso sistema. Será resultado do processamento feito pelos componentes do sistema.



1.2.2. Classificação dos sistemas de controle

Sistemas de controle podem ser divididos em dois tipos, os dinâmicos e os de evento discreto:

- Sistema de controle dinâmico: Analisa constantemente a variável afim de controlá-la. Ex.: Controle de altitude de um avião;
- Sistema de controle de evento discreto: Só desempenha uma ação caso ocorra um evento. Ex.: Limpador de para-brisa só liga se chover.

1.2.3. Exemplo de um sistema de controle dinâmico

Para mostrar como são organizados sistemas de controle, iremos apresentar dois exemplos práticos.

Controle de temperatura de um chuveiro

Um exemplo clássico de sistema de controle é o de controle de temperatura de um chuveiro, quando temos duas válvulas, uma de água gelada e outra de água quente, podemos descrever o processo da seguinte forma:

1. Abrimos um pouco de cada torneira até chegar em um volume de agua por tempo (vazão) adequada;
2. Verificamos a temperatura:
    a. Se muito quente, abrimos mais a água fria ou diminuímos a água quente;
    b. Se muito frio, abrimos mais a água quente ou diminuímos a água fria;
3. Caso necessário, voltamos ao início do processo.

Veja que podemos definir cada claramente o objetivo, componentes do sistema de controle e a saída.

a. O objetivo do controle: Controlar a temperatura da água;
b. Componentes do sistema de controle: A válvula de água quente e a

válvula de água gelada, nossa mão, que irá abrir ou fechar as válvulas e irá verificar a temperatura, e nosso cérebro, que irá tomar as decisões.

c. Resultados ou saídas: A temperatura da água.

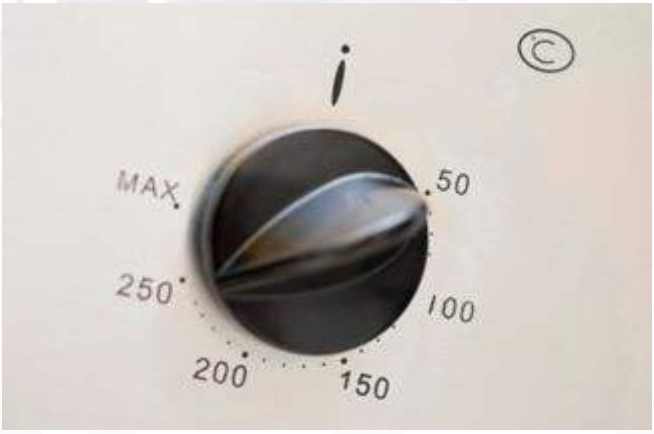
Vamos expandir o item b agora, seguindo um fluxo de controle, primeiro atuamos, depois verificamos e depois ajustamos.

Veja que existem 3 importantes componentes do sistema de controle:

- O atuador: Dispositivo que iremos utilizar para controle de alguma variável. No nosso exemplo, usamos a válvula das tomadas de água quente e fria para modificar a temperatura e a nossa mão;
- O sensor: Dispositivo que irá medir a variável que queremos controlar. No nosso caso queremos controlar a temperatura e usamos a mão como sensor da temperatura da água;
- O controlador: Dispositivo que irá tomar as decisões com base nos objetivos e irá atuar no processo por meio do atuador. No nosso exemplo, o nosso cérebro trabalha como controlador do processo.

Controle de temperatura de um forno

Um outro exemplo de um sistema de controle é o de temperatura de um forno. Quando queremos uma determinada temperatura, apenas giramos a válvula de gás até a posição que indica a temperatura desejada.



De alguma forma, a empresa que projetou o fogão verificou que a temperatura média em uma determinada posição seja 50ºC, porém, não existe na pratica um sensor que comprove que a temperatura seja de fato50ºC. Por exemplo, podemos obter temperaturas diferentes para a mesma posição caso usemos gases diferentes como combustível.

1.2.4. Sistemas de controle de malha aberta vs malha fechada

Podemos observar que, no exemplo do chuveiro, temos um fluxo cíclico. Verificamos a temperatura, tomamos uma decisão de quanto devemos ajustar, atuamos na válvula com o intuito de ajustar a temperatura, verificamos novamente a temperatura e o fluxo cíclico continua a rodar até que seja alcançada a temperatura desejada.

Já no exemplo do forno, não temos como verificar a temperatura e, por conta disso, não conseguimos ajustar de forma precisa a temperatura do equipamento. Desta forma temos um fluxo não-cíclico. Escolhe-se uma temperatura e depois atuamos na válvula.

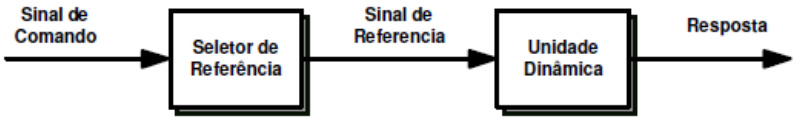
A principal diferença entre esses dois exemplos está no fato de no primeiro nós termos uma forma de mensurar o valor da variável que estamos controlando através de nosso tato. Dessa forma, conseguimos realimentar o nosso cérebro com uma informação de como nosso processo está na pratica, ajudando-o a tomar a próxima decisão.

Sistemas que possuem essa realimentação são denominados sistema de controle de malha fechada, enquanto os que não possuem essa realimentação são denominados sistemas de malha aberta.

Sistema de controle em malha aberta

Nessa topologia, os controladores em malha aberta, injetam no sistema sinais de controle pré-definidos, sem retorno do resultado obtido.

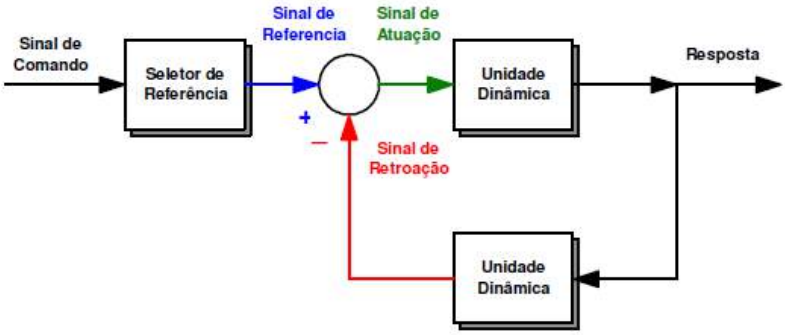
Por exemplo, ao selecionamos a posição do seletor de temperatura de um forno, que resultará em uma abertura da válvula de gás de nosso aparelho que, por fim, resultará em uma temperatura que não podemos afirmar ser a que escolhemos.



Sistema de controle em Malha Fechada

Nessa topologia, os sinais de controle injetados no sistema são dependentes dos valores de resposta de controle. Desta forma, o sinal controle é o resultado da relação entre a variável controlada e o valor desejado.

Por exemplo, escolhemos uma temperatura mentalmente para o chuveiro e atuamos nas válvulas de água, isso resultará em uma temperatura que iremos sentir com nosso tato e verificar a necessidade de ajuste.



Sabemos que sistemas de controle em malha aberta são mais baratos, porém, não possuem precisão. Portanto, os sistemas de malha fechados são cada vez mais usados. Para que um sistema seja autônomo é fundamental que ele seja adaptativo. Por isso, ele precisa enxergar as variáveis físicas que serão importantes para o desempenho da atividade. Logo, trabalharemos bastante com controle em malha fechada nesse tipo de sistema.

Pense em uma geladeira ou num ar condicionado. Neles, devemos definir o valor de temperatura desejado e, a partir desse valor, o equipamento deve controlar essa variável. Para que ele tenha capacidade de controlar a temperatura, é fundamental que o sistema tenha um termômetro para que haja a leitura da temperatura.

1.3. Sensores

Da famosa enciclopédia online Wikipédia, temos que:

"Um sensor é um dispositivo que responde a um estímulo físico/químico de maneira específica e mensurável analogicamente."

Ou seja, sensores são dispositivos capazes de ler variáveis físicas ou químicas do ambiente e transformar em informação.

Qualquer processo automático possui sensores. Em indústrias, onde temos processos de produção automatizados, há muitos tipos de sensores medindo as mais diversas variáveis do processo: temperatura, pressão, peso, pH, dentre muitos outros.

Devido à importância da leitura dessas variáveis, existe uma área responsável por instrumentos de medição industrial, a Instrumentação Industrial.

Nós, seres humanos, também somos feitos de vários sensores que nos ajudam a desenvolver nossas tarefas. Os nossos cinco sentidos: olfato, audição, paladar, tato e visão, nada mais são do que conjuntos de sensores que colhem informações do ambiente para que o cérebro possa tomar decisões adequadas em cada situação.

1.4. O uso dos sensores

Tão importante quanto medir uma variável física é transformá-la em uma informação legível pelo cérebro do sistema autônomo, como nosso Arduino, por exemplo. Pensando em eletrônica, essa informação pode ser digital ou analógica

- Digital: Quando a informação é passada através de valores lógicos alto (1) ou valores lógicos baixo (0).
- Analógico: Quando a informação pode assumir qualquer valor dentro de um máximo e um mínimo. Quando trabalhamos com eletrônica, geralmente essa informação é dada por um valor de corrente ou tensão.

Obs.: Leia mais sobre sinais analógicos e discretos na apostila Arduino Básico Vol 1.

1.5. O foco de nossa apostila

Sabendo da importância dos sensores para os sistemas autônomos, nessa apostila iremos introduzir alguns sensores básicos. São eles:

- O sensor de luz com LDR;
- O sensor de temperatura LM35;
- O sensor de temperatura e umidade DHT11;
- O sensor de distância ultrassônico HC-SR04.

Além disso, iremos ensinar a usar um emissor de som (buzzer), que funcionará como um atuador para nossas experiências. Aprenderemos também como usar um LCD de caractere 16x02 para visualizar as informações lidas pelo Arduino, como temperatura e distância.

1.6. Ajude-nos a criar um material de primeira

Antes de você começar a ler essa apostila, te convido para nos ajudar a criar conteúdos cada vez melhores. Caso tenha alguma sugestão para as novas apostilas ou encontrou algum erro nesse material, mande um e-mail para contato@vidadesilicio.com.br. Ficaremos muito felizes em te ouvir.

Caso você possua vontade, tal como nós, de ajudar a comunidade Arduino e tenha interesse em fazer tutoriais bacanas ou até nos ajudar a montar uma apostila para dividir seu conhecimento por meio da internet, mande um e-mail para contato@vidadesilicio.com.brcom um tutorial de sua autoria que ficaremos felizes em abrir um espaço para você em nosso blog.

2. Usando o buzzer com Arduino – Transdutor piezo elétrico

Que tal fazer um pouco de som com seu Arduino? Nesse tutorial iremos aprender a usar um dispositivo um tanto quanto barulhento, O sonorizador piezo.

Todo mundo se lembra daqueles cartões de natal que tocava Jingle Bells, não é? Eu gostava muito!! Era intrigante imaginar como seria possível sair som de um simples cartão natalino. Estava acostumado a ouvir sons saindo de grandes alto-falantes e, de repente, me deparava com um cartão que era capaz de tocar música.

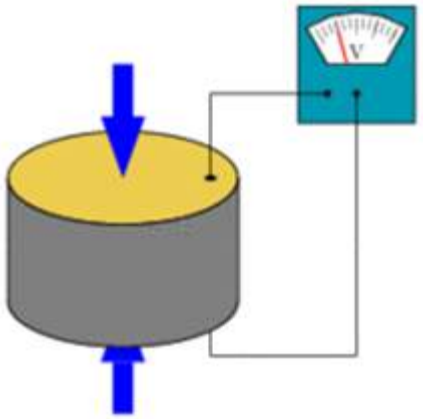
Alguns anos depois descobri como funcionavam esses cartões. Eles usam um dispositivo chamado transdutor piezoelétrico.

Transdutor piezoelétrico

A palavra piezo é de origem grega e significa pressionar, torcer. Piezos são elementos que utilizam o efeito piezoelétrico.

Do famoso portal wikipedia, temos que:

"Piezoeletricidade é a capacidade de alguns cristais gerarem tensão elétrica por resposta a uma pressão mecânica."



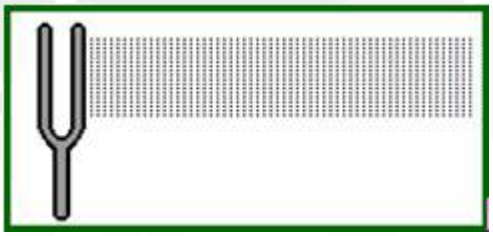
Sendo assim, alguns materiais possuem a capacidade de gerar tensão elétrica quando são submetidos a uma força externa. Uma das aplicações dessa propriedade é na forma de sensor de batida, como em baterias elétricas. Outro lugar comum de encontrá-los é em isqueiros, que transformam o movimento do ignitor em faísca.

O mais interessante é que o contrário também é verdade. Ao inserirmos um diferencial de tensão no cristal, ele transforma energia elétrica em energia mecânica. Dessa forma, o piezo se mexe.

Através desse efeito, podemos alimentar o transdutor com uma tensão variável com o intuito de obter sons.

O Som

O som é a propagação de uma vibração acústica em um meio material elástico, tais como o ar e a água, que seja perceptível pelo aparelho auditivo humano. Essa vibração gera o movimento das partículas desses materiais que, com isso, transmitem o som.

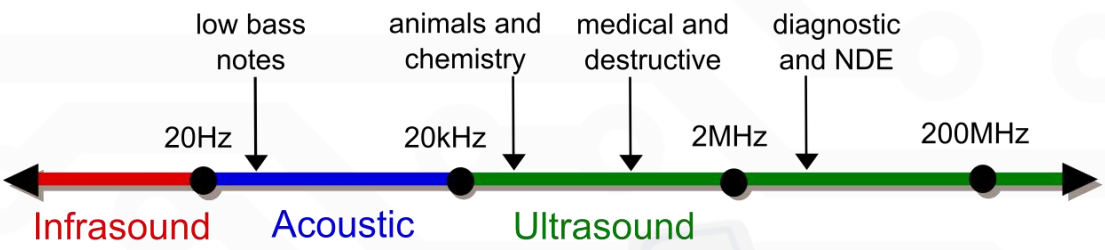


Como seres humanos, temos como um de nossos principais sensores o sistema auditivo. Além de nos permitir escutar, ele também está intimamente ligado ao equilíbrio do nosso corpo. (Recomendo a todos que estudem sobre os sistemas sensoriais do corpo humano. O que tentamos fazer com robótica já se encontra pronto na natureza).

Em resumo, o ouvido recebe o som e o transforma em impulsos nervosos, que são enviados para nosso cérebro, que, por sua vez, tem o trabalho de interpretar a informação.

O nosso aparelho auditivo é capaz de ouvir vibrações com frequência mínima de 20Hz, enquanto a frequência máxima chega a 20.000Hz. Sons cuja frequência sejam

maior que 20kHz são denominados ultrassons, enquanto que aqueles abaixo de 20Hz são chamados de infrassons.



Sabendo que o som é a propagação de uma vibração entre 20Hz e 20kHz, podemos usar a propriedade do transdutor piezoelétrico de se mexer ao aplicarmos uma tensão elétrica e gerar som.

Para isso, podemos colocar uma onda de tensão na faixa de frequência audível, assim o transdutor vibrará na mesma frequência, emitindo som. Veremos isso na prática durante a experiência.

Como eles são?

Em geral, o piezo é encontrado em formato de disco com dois fios. Um vermelho, o positivo, e um preto, o negativo. Veja um típico piezo na imagem a seguir.

Quando o assunto é Arduino, utilizamos muito o buzzer, que nada mais é que um disco piezo encapsulado em uma proteção plástica.

Podemos usar ele sozinho ou em um módulo, tal como o da GBK robotics

2.1. Experiência 1– Fazendo uma sirene com um buzzer

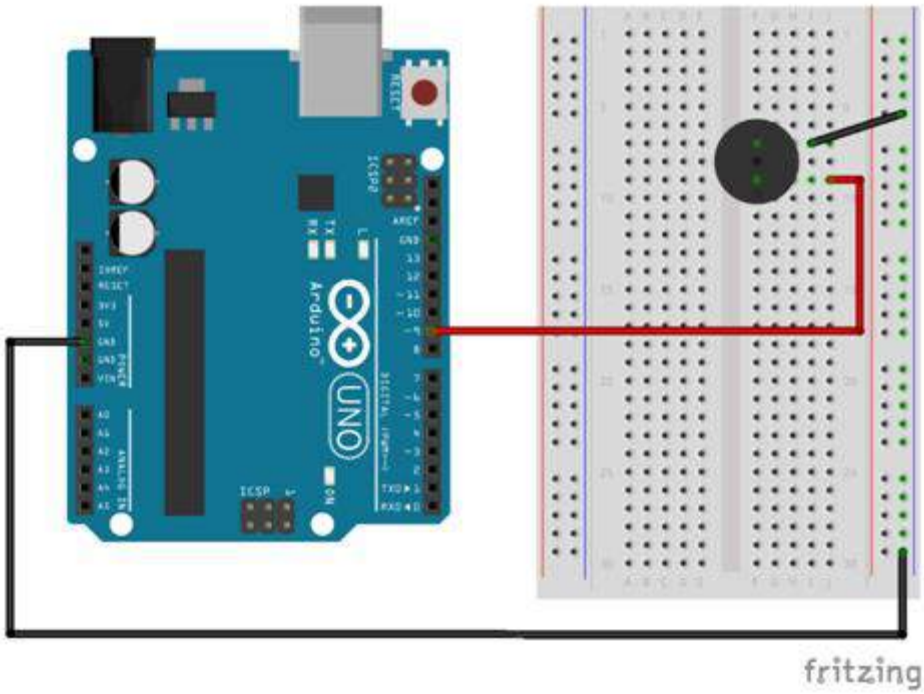
2.1.1. Ingredientes

Os itens que usaremos nessa experiência podem ser encontrados em nosso site (www.vidadesilicio.com.br). São eles:

- 2 Fios Jumper's
- Protoboard
- Arduino Uno Rev3
- Buzzer

2.1.2. Misturando os ingredientes

Agora vamos conectar os componentes do projeto. Para isso, desligue o cabo USB de seu Arduino e monte seu circuito conforme a figura a seguir.



2.1.3. Levando ao forno

Conecte seu Arduino ao computador e abra a IDE Arduino.

Antes de carregar um programa, você precisa selecionar qual porta você deseja usar para fazer carregar o programa no Arduino (upload). Dentro do Arduino IDE, clique no menu Ferramentas (tools) e abra o submenu Porta(Port). Clique na porta que seu Arduino está conectado, tal como COM3 ou COM4. Geralmente aparece o nome da placa Arduino: “COM3 (Arduino Uno)”.

Você também precisa garantir que o tipo de placa apropriado está selecionado em Ferramentas (Tools) no submenu Placa (Board).

2.1.4. Preparando a cobertura

Crie um programa (sketch) e salve com o nome de “programa_buzzer_sirene”.

Com o seu programa salvo, escreva nele o código escrito abaixo:

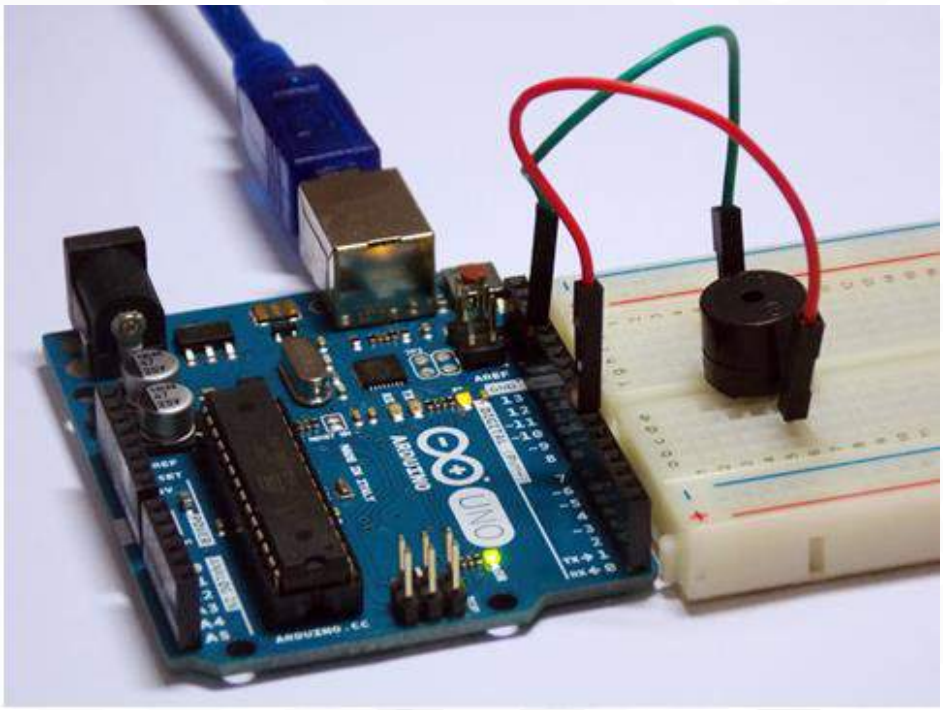
```
//Sensor de luz

float seno;
int frequencia;

void setup() {
 //define o pino 9 como saída
 pinMode(9,OUTPUT);
}

void loop() {
 for(int x=0;x<180;x++){
  //converte graus para radiando e depois obtém o valor do seno
  seno=(sin(x*3.1416/180));
  //gera uma frequência a partir do valor do seno
  frequencia = 2000+(int(seno*1000));
  tone(9,frequencia);
  delay(2);
 }
}
```

Após escrever o código, clique em Upload para que o programa seja transferido para seu Arduino.

2.1.5. Experimentando o prato

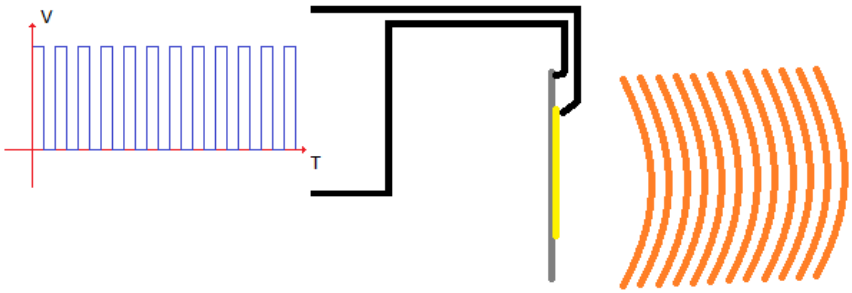
Caso tenha ocorrido tudo como esperado, o buzzer deve começar a “gritar” como uma sirene.

Cuidado: Apesar de surpreendente, não-entendedores podem ficar bravos com você devido ao barulho!!



2.2. Entendendo o Hardware

O som da sirene que ouvimos é resultado de um sinal digital de frequência variável na saída do pino 9, que, ao energizar a perna positiva do buzzer, faz com que o mesmo emita um som com frequência igual ao da saída do Arduino.



Para que o som seja o de uma sirene, aumentamos e diminuímos, através do programa, a frequência de saída do Arduino. Caso colocássemos uma frequência fixa, teríamos um som continuo.

2.3. Entendendo o programa

- *Bloco For*

Frequentemente em nossos programas precisamos executar uma sequência de comandos por um número determinado de vezes. Quando queremos algo do gênero,

usamos o bloco **for**.

O **for**, em português **para**, é um bloco clássico usado em programação. Em resumo, o for é um contador que executará as linhas de códigos que estão entre suas chaves "{}" enquanto a condição que ele definiu for verdadeira. Além disso, ele também define a variável que será usada como parâmetro, seu valor inicial e como ela será incrementada.

```
for(int x=0;x<180;x++){
 //linhas de código

}
```

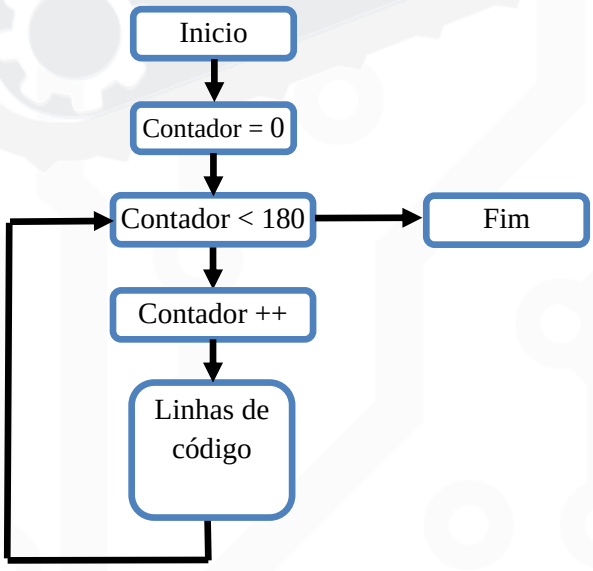
No nosso exemplo, temos como variável inteira de parâmetro, o x, que inicia com seu valor igual a zero (int x=0). Teremos x sendo incrementado de um em um (x++), a cada execução do bloco for. E, por fim, as linhas de código de for serão executadas enquanto o valor de x for menor que 180 (x<180).

Observe como o **for** trabalha como um contador, onde são definidos os valores iniciais e finais do contador e sua velocidade de incremento. No nosso exemplo ele contará de 1 em 1 a partir de 0 até chegar a 180.

Dessa forma, temos:

```
for(valor inicial; condição de execução; velocidade do contador){
 //linhas de código
}
```

Podemos desenhar o fluxograma de um for da seguinte forma: inicio



Uma vantagem de usar o for, é que podemos definir a variável usada como parâmetro, que será incrementada a cada novo ciclo do bloco for, como parte de alguma conta dentro do bloco.

É importante dizer que a variável parâmetro sempre será uma variável do tipo int, inteira.

- *Seno*

Para que o nosso buzzer emita o som de uma sirene, precisamos que sua frequência do som emitido aumente e diminua. Dessa forma, o buzzer irá variar de frequências mais graves até frequências mais agudas e vice-versa.

Para isso, usaremos a função seno em conjunto com a variável parâmetro de for, o x, que varia de 0 a 180. A ideia é que x simbolize 0 a 180 graus, como a função sin faz o cálculo usando valores em radianos. Precisamos converter graus para radianos. A variável seno receberá o valor de seno para o ângulo x convertido em radianos.

```
seno=(sin(x*3.1416/180));
```

Seno será um valor entre 0 e 1, visto que trabalharemos com um ângulo menor que 180 graus. Dessa forma, multiplicaremos por 1000 para que ele varie de 0 a 1000 e posteriormente somamos a 2000, que será a frequência mínima do som.

```
frequencia = 2000+(int(seno*1000));
```

Portanto, teremos a variável frequência variando entre 2000 e 3000 em função de x, que irá variar conforme o bloco for incrementá-lo.

- *tone()*

Por fim, no Arduino, temos uma função que gera uma onda pulsante na frequência desejada. O nome dessa função é tone.

```
tone(pino,frequência);
tone(pino,frequência,duração);
```

Para usar essa função, devemos definir o pino de saída, a frequência do tom e, caso queira, a duração em milissegundos da onda. Em nossa experiência usamos da seguinte forma:

```
tone(9,frequencia);
```

Veja que a frequência de saída irá variar entre 2000Hz e 3000Hz.

Vale lembrar que quando o x for 180, o programa irá sair de for, porem irá executá-lo novamente, devido ao fato de tudo que está dentro da função loop ser executado ciclicamente.

```
void loop() {
 for(int x=0;x<180;x++){
  //converte graus para radiando e depois obtém o valor do seno
  seno=(sin(x*3.1416/180));
  //gera uma frequência a partir do valor do seno
  frequencia = 2000+(int(seno*1000));
  tone(9,frequencia);
  delay(2);
 }
}
```

3. Sensor de Luz – Aprendendo a usar o LDR

Que tal construir um dispositivo que seja capaz de detectar luz? Você pode, por exemplo, fazer uma lâmpada que ligue sozinha durante a noite. Neste capítulo, iremos aprender a usar um componente simples e barato que pode ser usado em diversos projetos, o LDR.

Relembrar é viver

Quem nunca se perguntou como os postes ligam sozinhos de noite? Quantas vezes, quando éramos crianças, tentamos apagar as lâmpadas deles usando um laser? Possivelmente, muitos de nós já sabíamos qual era o dispositivo que acionava as lâmpadas.

O nome do dispositivo responsável por saber a hora certa de iluminar é conhecido como fotocélula ou relé fotocélula. Ele recebe a luz através dessa parte transparente. Como é possível ver na imagem abaixo, a fotocélula possui um LDR que irá medir a luminosidade.

Nessa fotocélula, quando o valor de luminosidade é menor do que o valor desejável, ele comuta uma chave que poderá ligar uma lâmpada, por exemplo.

LDR

O LDR, sigla em inglês de Light-Dependent Resistor, que significa resistor dependente de luz, nada mais é do que o que o próprio nome diz. Tipicamente, quanto maior a luz incidente nesse componente, menor será sua resistência.

O LDR é constituído de um semicondutor de alta resistência, que, ao receber uma grande quantidade de fótons oriundos da luz incidente, absorve elétrons, que melhoram sua condutibilidade, reduzindo assim sua resistência.

Dessa forma, esse semicondutor pode assumir resistências na ordem de megaOhm no escuro e resistência na ordem de poucas centenas quando exposto a luz.

3.1. Experiência2 - Mãos à obra - Fazendo um sensor de luz

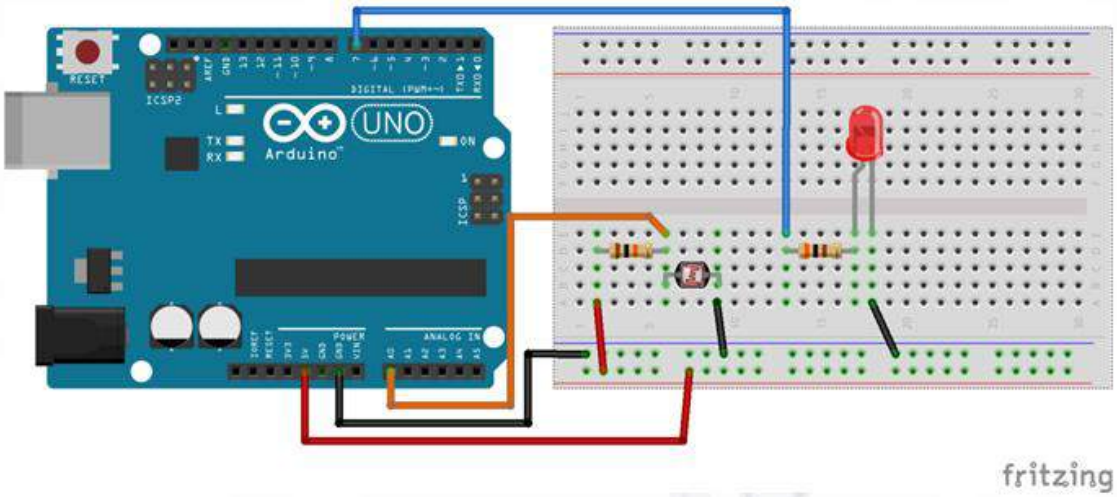
3.1.1. Ingredientes

Os itens que usaremos nessa experiência podem ser encontrados em nosso site (www.vidadesilicio.com.br). São eles:

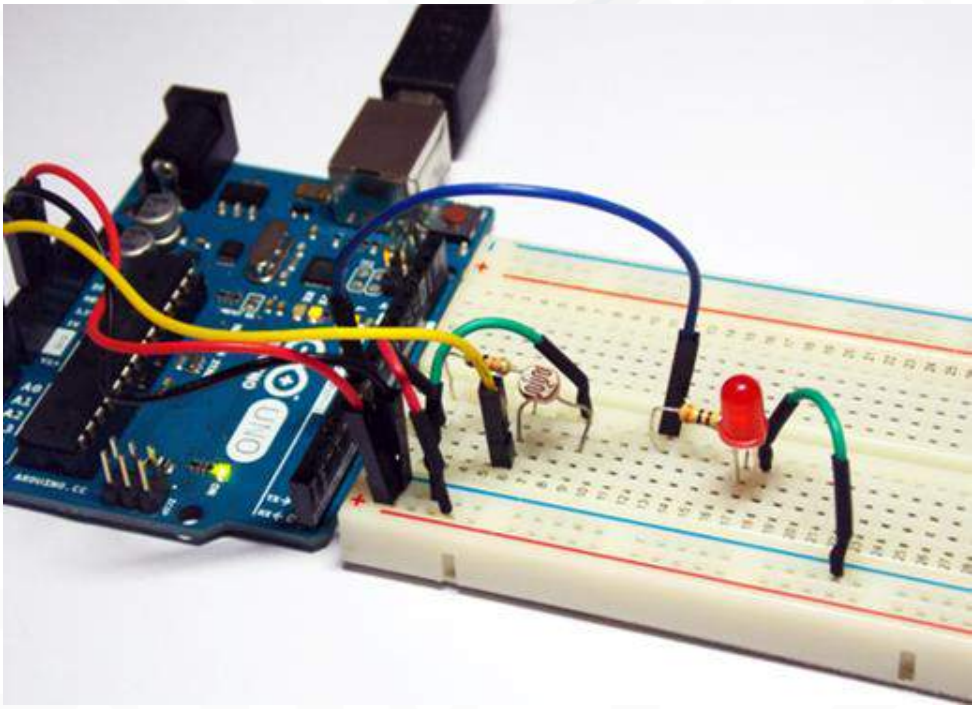
- Fios Jumper's
- Protoboard
- Arduino Uno Rev3
- 1x LED
- 1x Resistor 300Ohm
- 1x LDR
- 1x Resistor 10kOhm

3.1.2. Misturando os ingredientes

Agora vamos conectar os componentes do projeto. Para isso, desligue o cabo USB de seu Arduino e monte seu circuito conforme a figura a seguir.



Veja como ficou o nosso:



3.1.3. Levando ao forno

Conecte seu Arduino ao computador e abra a IDE Arduino.

Antes de carregar um programa, você precisa selecionar qual porta você deseja usar para fazer carregar o programa no Arduino (upload). Dentro do Arduino IDE, clique no menu Ferramentas (tools) e abra o submenu Porta(Port). Clique na porta que seu Arduino está conectado, tal como COM3 ou COM4. Geralmente aparece o nome da placa Arduino : “COM3 (Arduino Uno)”.

Você também precisa garantir que o tipo de placa apropriado está selecionado

em Ferramentas (Tools) no submenu Placa (Board).

3.1.4. Preparando a cobertura

Crie um programa (sketch) e salve com o nome de “programa_sensor_de_luz”.

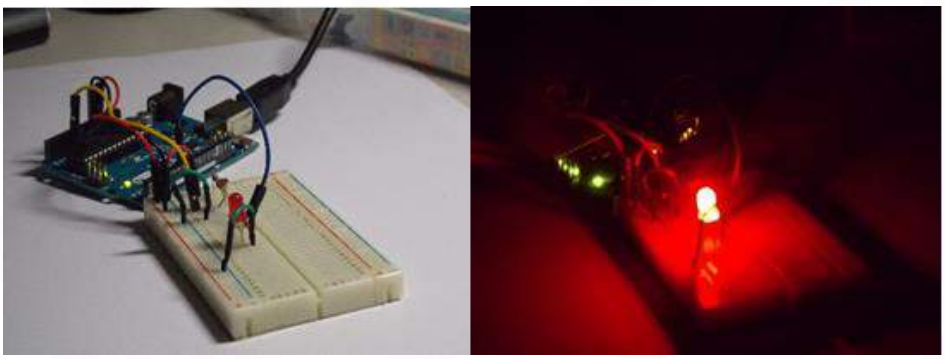
Com o seu programa salvo, escreva nele o código abaixo.

```
//Sensor de luz

int ledPin = 7; //Led no pino 7
int ldrPin = 0; //LDR no pino analígico 8
int ldrValor = 0; //Valor lido do LDR

void setup() {
 pinMode(ledPin,OUTPUT); //define a porta 7 como saída
 Serial.begin(9600); //Inicia a comunicação serial
}

void loop() {
 ///ler o valor do LDR
 ldrValor = analogRead(ldrPin); //O valor lido será entre 0 e 1023

 //se o valor lido for maior ou igual à 800, liga o led
 if (ldrValor>= 800) digitalWrite(ledPin,HIGH);
 // senão, apaga o led
 else digitalWrite(ledPin,LOW);

 //imprime o valor lido do LDR no monitor serial
 Serial.println(ldrValor);
 delay(100);
}
```

Após escrever o código, clique em Upload para que o programa seja transferido para seu Arduino.

3.1.5. Experimentando o prato

Caso tenha ocorrido tudo como esperado, ao cobrir o LDR, o LED irá acender. Abra o monitor serial para verificar o que está sendo lido na entrada A0.



3.2. Entendendo o Hardware

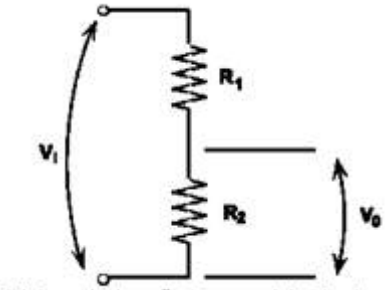
3.2.1. Divisor de tensão

Quando associamos resistênciasem série, temos o que é chamado de divisor de

tensão. Em um circuito divisor de tensão, temos uma queda de tensão em cada resistência igual ao produto da resistência com a corrente do circuito.

Como a corrente do circuito é calculada pela divisão da tensão sobre todos os resistores dividido pela soma dos resistores, teremos a tensão em cima de um resistor igual a resistência desse resistor vezes a tensão total dividida pela soma dos resistores.

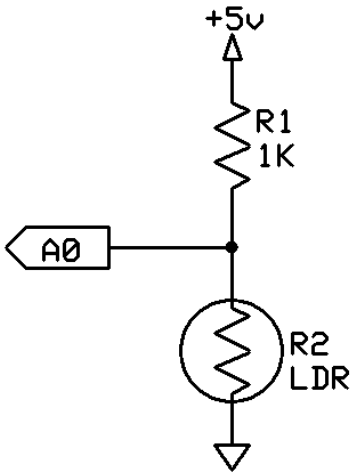
O exemplo a seguir mostra como funciona o cálculo o para dois resistores.



Divisor de tensão numa série de duas resistências

$$V_0 = \frac{R_2}{R_1 + R_2} \times V_i$$

Quando usamos um LDR, que é uma resistência foto-variável, podemos usufruir da propriedade do divisor de tensão para medir a variação da queda de tensão sobre o mesmo. Sabemos que a tensão total e a resistência total são fixas. Desta forma, o divisor de tensão irá variar com a resistência entre A0 e GND.



Levando em conta que, quanto menos luz incidir sobre o LDR, maior será sua resistência, teremos a tensão sobre o LDR e, por conseguinte, o valor de A0 maior com um índice de luminosidade incidente menor, isto é, num local mais escuro.

3.3. Entendendo o Software

3.3.1. Lendo da Entrada Analógica

A leitura da entrada analógica é feita com a função analogRead, que recebe como parâmetro o pino analógico a ser lido e retorna o valor digital que representa a tensão no pino. Como o conversor analógico-digital do Arduino possui uma resolução de 10 bits, o intervalo de tensão de referência, que no nosso caso é 5 V, será dividido em 1024 pedaços (2^10) e o valor retornado pela função será o valor discreto mais próximo da tensão no pino.

```
ldrValor = analogRead(ldrPin); //O valor lido será entre 0 e 1023
```

O código acima lê o valor analógico de tensão no pino A0 e guarda o valor digital na variável valorLido. Supondo que o pino está com uma tensão de 2V, o valor retornado pela conversão será:

$$2 \times 1024 / 5 = 409,6$$

O resultado deve ser inteiro para que nosso conversor consiga representá-lo, logo, o valor 410 será escolhido por ser o degrau mais próximo. Esse valor representa a tensão 2,001953125, inserindo um erro de 0,001953125 em nossa medida devido à limitação de nossa resolução.

Leia mais sobre sinais analógicos e discretos na apostila Arduino Básico Vol 1.

3.3.2. Lógica do programa

Em resumo, nosso programa lerá qual é o valor do sinal em A0 com o auxílio do comando analogRead(), que retornará um valor entre 0 a 1023, e o comparará com um valor de referência que em nosso caso é 800. Tendo em vista que, quanto mais escuro, maior será o valor de A0, caso A0 seja maior que o valor de referência o programa liga o LED conectado ao pino 7. Do contrário, ele apaga o LED. O programa também imprime o valor de A0 para que possamos verificar a faixa de valores e até mesmo calibrar nosso sensor.

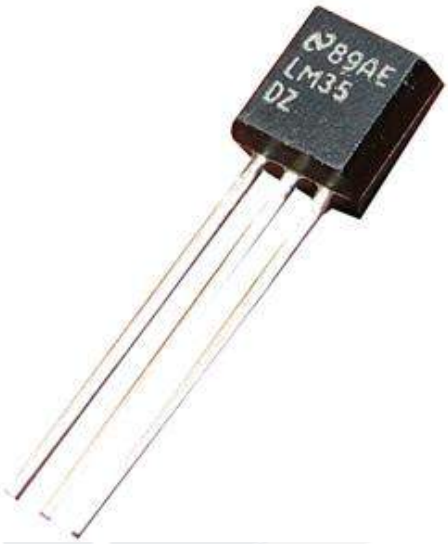
Tente mudar o valor de referência e veja que, quanto maior esse valor, menor será sua sensibilidade.

4. Sensor de temperatura - LM35

Em diversas aplicações temos que fazer a leitura de diversas variáveis físicas, tais como a temperatura, distância, entre outras. Neste capítulo, iremos aprender como fazer a leitura de temperatura usando o LM35, um sensor barato e de ótima precisão. Vamos lá?

4.1. LM35

O LM35 é um sensor de precisão que apresenta uma saída de tensão linear proporcional à temperatura em que ele se encontrar no momento, tendo em sua saída um sinal de 10mV para cada grau Célsius de temperatura.

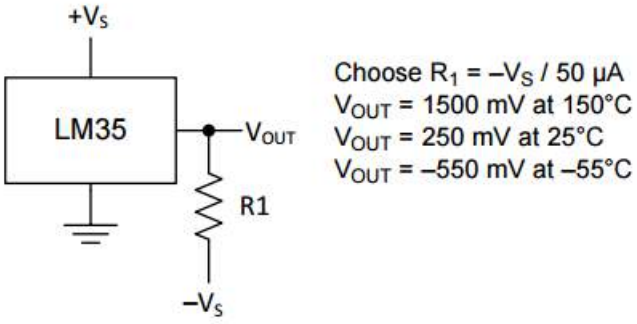


LM35

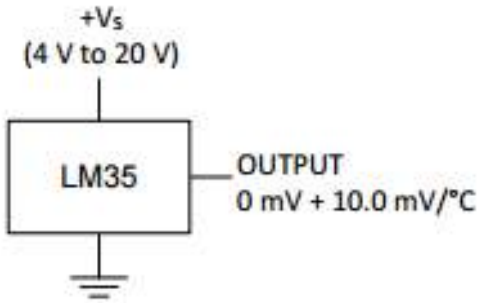
Esse sensor não necessita de qualquer calibração externa para fornecer com exatidão, valores temperatura com variações de ¼ºC ou até mesmo ¾ºC dentro da faixa de temperatura entre –55ºC e 150ºC.

Ele pode ser usado de duas formas: com alimentação simples ou simétrica, dependendo do que se desejar como sinal de saída, mas, independentemente disso, a saída continuará sendo de 10mV/ºC.

Em cada uma dessas duas formas de alimentação, o range de temperatura, ou seja, a temperatura máxima e mínima medida com exatidão, é diferente.



Modo escala completa (-55ºC a 150ºC)

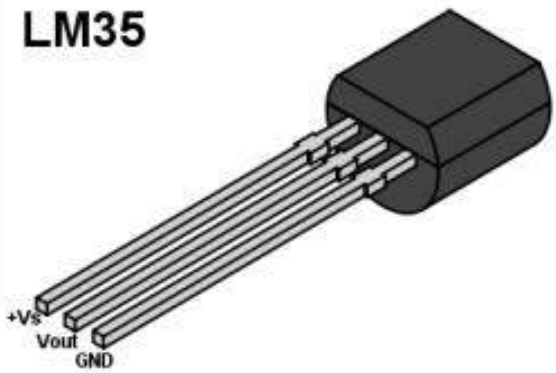


Modo básico (2ºC a 150ºC)

Uma vantagem é o fato desse sensor drenar apenas 60µA para essas alimentações. Desta forma, seu auto aquecimento é de aproximadamente 0.1ºC ao ar livre.

O sensor LM35 é apresentado com vários tipos de encapsulamentos, sendo o mais comum o TO-92, que mais se parece com um transistor, e oferece ótima relação custo-benefício, por ser o encapsulamento mais barato sem diferenças em seu uso ou exatidão.

Veja a folha de dados dele clicando aqui.



4.2. Experiência 3 - Mãos à obra – Medindo temperatura

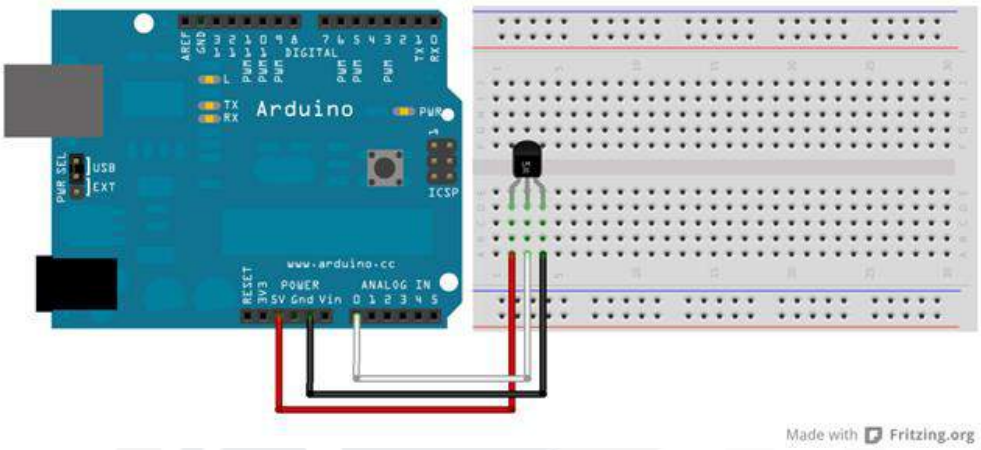
4.2.1. Ingredientes

Os itens que usaremos nessa experiência podem ser encontrados em nosso site (www.vidadesilicio.com.br). São eles:

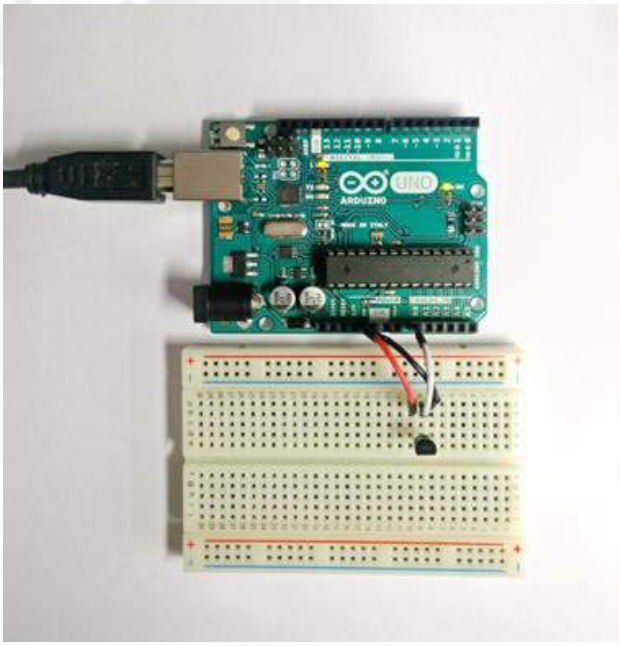
- LM35
- Fios Jumper's
- Protoboard
- Placa Uno R3

4.2.2. Misturando os ingredientes

Agora vamos conectar os componentes do projeto. Para isso, desligue o cabo USB de seu Arduino e monte seu circuito conforme a figura a seguir.



Veja como ficou o nosso:



4.2.3. Levando ao forno

Conecte seu Arduino ao computador e abra a IDE Arduino.

Antes de carregar um programa, você precisa selecionar qual porta você deseja usar para fazer carregar o programa no Arduino (upload). Dentro do Arduino IDE, clique no menu Ferramentas (tools) e abra o submenu Porta (Port). Clique na porta que seu Arduino está conectado, tal como COM3 ou COM4. Geralmente aparece o nome da placa Arduino : "COM3 (Arduino Uno)".

Você também precisa garantir que o tipo de placa apropriado está selecionado em Ferramentas (Tools) no submenu Placa (Board).

4.2.4. Preparando a cobertura

Crie um programa (sketch) e salve com o nome de "programa_sensor_de_luz".

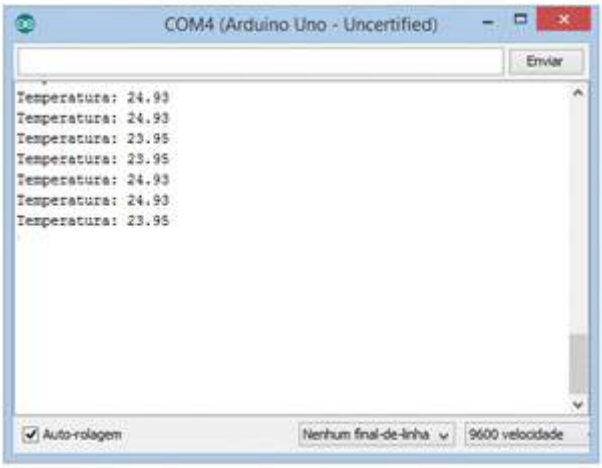
Com o seu programa salvo, escreva nele o código abaixo:

```
//Sensor de temperatura usando o LM35

const int LM35 = A0; // Define o pino que lera a saída do LM35
float temperatura; // Variável que armazenará a temperatura medida

//Função que será executada uma vez quando ligar ou resetar o Arduino
void setup() {
Serial.begin(9600); // inicializa a comunicação serial
}

//Função que será executada continuamente
void loop() {
temperatura = (float(analogRead(LM35))*5/(1023))/0.01;
Serial.print("Temperatura: ");
Serial.println(temperatura);
delay(2000);
}
```

Após escrever o código, clique em Upload para que o programa seja transferido para seu Arduino.

4.2.5. Experimentando o prato

Caso tenha ocorrido tudo conforme esperado, poderemos fazer a leitura da temperatura através do monitor serial. Abra o monitor serial para verificar o que está sendo lido na entrada A0.



4.3. Entendendo o Programa

Primeiramente, em nosso programa usamos o comando de leitura analógica, já estudado no capítulo anterior, para fazer a leitura do valor em A0. Além disso, usamos a comunicação serial, discutida em nossa última apostila e em nosso tutorial, Comunicação Serial Arduino. É importante que o leitor entenda como eles funcionam.

Em resumo, nosso programa lerá qual é o valor do sinal em A0, que varia de 0 a 1023, onde 0 corresponde a 0Volts e 1023 corresponde a 5Volts. Como sabemos, 1ºC é igual a 10mV. Teremos:

Tensão em A0 = (Valor lido em A0)*(5/1023)

Temperatura = Tensão em A0/10mV

Logo:

Temperatura = [(Valor lido em A0)*(5/1023)]/10mV

Em linguagem de programação, ficará:

```
temperatura = (float(analogRead(LM35))*5/(1023))/0.01;
```

Perceba que colocamos o comando de leitura do valor analógico, analogicRead, dentro de float(). Você saberia me dizer o motivo?

Quando o Arduino faz uma leitura analógica, ele converte o valor lido, que pode ser um valor de tensão entre 0V e 5V, em um número entre 0 e 1023. Ou seja, o Arduino divide 5Volts, que é o maior valor que ele é capaz de ler, em 1024 partes iguais e te informa quantas partes tem o valor que ele está medindo.

O número que o Arduino te informa é do tipo inteiro, contudo, o valor de temperatura é um número racional, que pode assumir valores decimais. Por conta disso,

no nosso programa, declaramos a temperatura como uma variável de tipo float.

Em programação, quando multiplicamos uma variável inteira por uma variável racional, o programa considera que o resultado deve ser inteiro, eliminando a parte decimal da variável. Assim, para que tenhamos um resultado racional, devemos transformar o número inteiro em um número racional.

Em virtude disso, em nosso código foi necessário colocar o comando de leitura do valor analógico, analogicRead, dentro de float().

```
numeroracional = float(numero);
temperatura = (float(analogRead(LM35))*5/(1023))/0.01;
```

Sempre que for necessário fazer um cálculo com o valor analógico, precisamos convertê-lo para uma variável do tipo float.

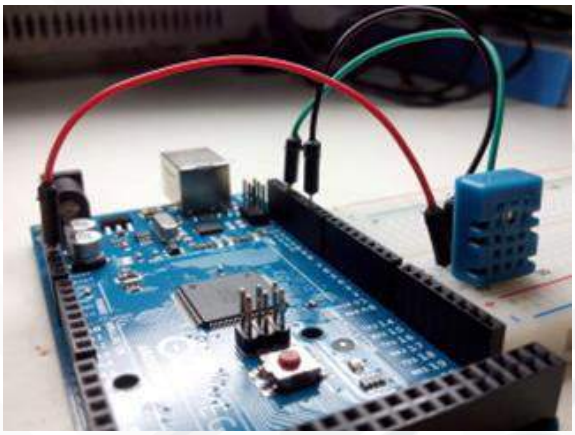
Em alguns casos, precisamos transformar uma variável qualquer para o tipo inteiro. O procedimento é o mesmo, ou seja, basta colocar o valor ou variável dentro dos parênteses de int();

```
numerointeiro= int(numero);
temperatura = int((float(analogRead(LM35))*5/(1023))/0.01);
```

Se usamos o int() no cálculo de temperatura, tal como mostrado acima, teremos um resultado sem os números decimais. Faça o teste.

5. Sensores DHT e Uso de Bibliotecas

O DHT é um sensor básico e de baixo custo que utiliza um termistor e um sensor capacitivo para medir a temperatura e a umidade do ar ambiente. Esse sensor é bastante simples de usar, mas requer cuidado com o tempo entre duas leituras consecutivas, uma vez que é necessário um intervalo de, no mínimo, 1 segundo entre uma leitura e outra.



Existem diferentes versões do DHT, similares na aparência e na pinagem, porém, com características diferentes. As características do DHT11 e DHT22, dois modelos populares desse sensor, são:

DHT11

- Baixíssimo custo
- Tensão de alimentação de 3V a 5V
- 2.5mA de corrente máxima durante a conversão
- Bom para medir umidade entre 20% e 80%, com 5% de precisão
- Bom para medir temperaturas entre 0 e 50°C, com ±2°C de precisão
- Taxa de amostragem de até 1Hz (1 leitura por segundo)
- Dimensões: 15.5mm x 12mm x 5.5mm
- 4 pinos com 0.1″ de espaçamento entre eles

DHT22

- Baixo custo
- Tensão de alimentação de 3V a 5V
- 2.5mA de corrente máxima durante a conversão
- Bom para medir umidade entre 0% e 100%, com 2% a 5% de precisão
- Bom para medir temperaturas entre -40 e 125°C, com ±0,5°C de precisão

- Taxa de amostragem de até 0,5Hz (2 leituras por segundo)
- Dimensões: 15.1mm x 25mm x 7.7mm
- 4 pinos com 0.1″ de espaçamento entre eles

Como pode ser observado, o DHT22 é um pouco mais preciso e trabalha em uma faixa um pouco maior de temperatura e umidade. Porém, ambos utilizam apenas um pino digital e são relativamente lentos, pois é necessário um intervalo de tempo relativamente grande entre cada leitura.

5.1. Incluindo bibliotecas

Uma grande vantagem das placas Arduino é a grande diversidade de bibliotecas disponíveis que podem ser usadas em seu programa. Isso faz com o trabalho pesado em programação seja abstraído e resumido em simples comandos.

Com isso, o desenvolvedor não precisa de conhecimento muito aprofundado em programação, podendo gastar menos tempo nisso,resultando em mais tempo para trabalhar com empenho na estratégia de controle.

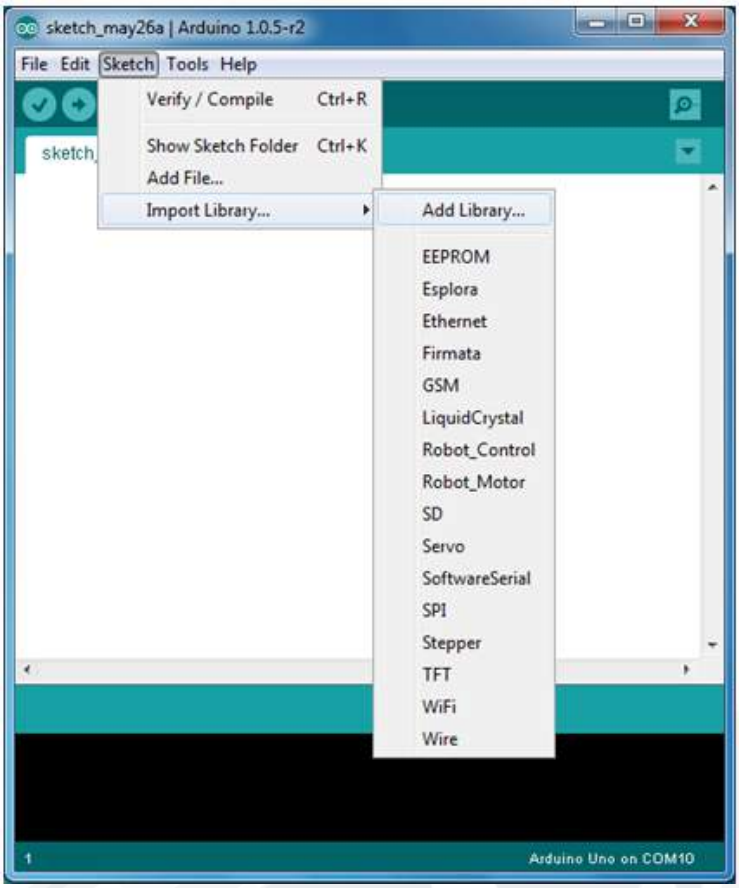
A seguir, iremos aprender como adicionar uma biblioteca em sua IDE.Esse mesmo procedimento será usado para outros sensores ou módulos.

5.1.1. Incluindo a biblioteca DHT

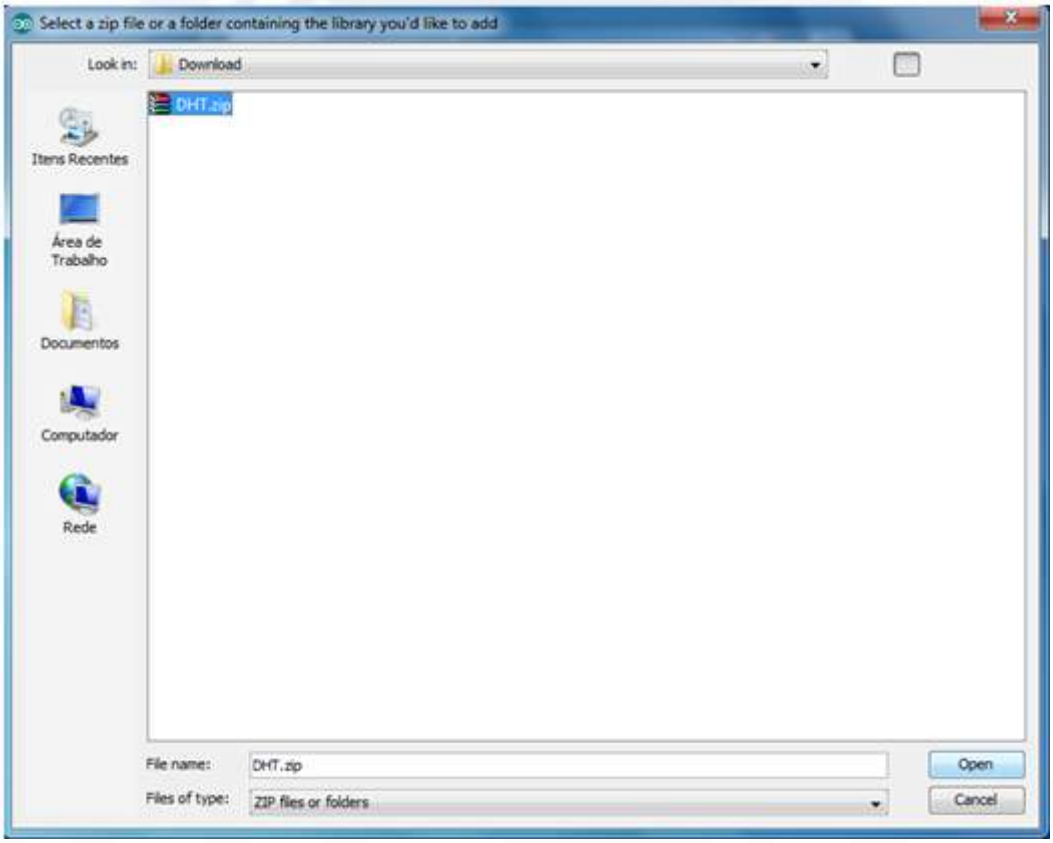
Para trabalhar de forma fácil com o DHT, podemos baixar uma biblioteca para ele no GitHub do Rob Tillaart (https://goo.gl/VkqZ5g). Há, basicamente, três formas de incluir bibliotecas no seu programa:

1ª opção) Instalar a biblioteca pelo IDE do Arduino

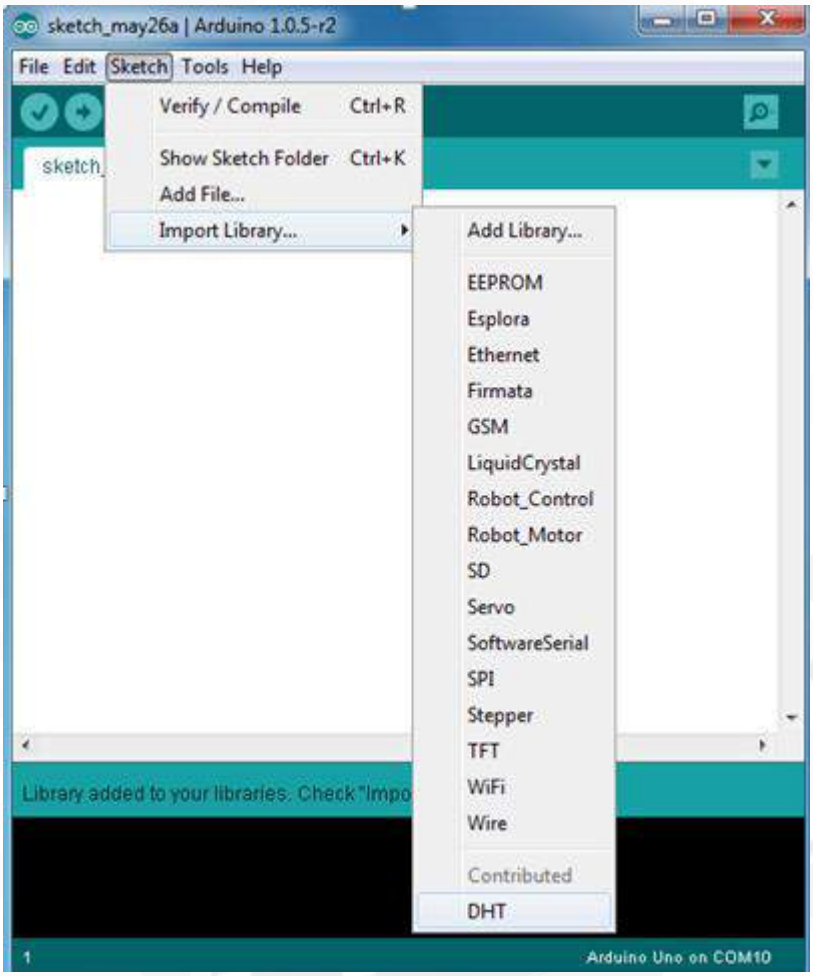
Esse é o método mais fácil. Primeiramente, faça o download dos arquivos da biblioteca compactados no formato zip. Geralmente, as bibliotecas já são distribuídas compactadas, porém, às vezes é necessário fazer o download dos arquivos separadamente e compactá-los à parte. Em seguida, basta abrir o IDE e ir em “Sketch -> Import Library… -> Add Library…”:



Na janela que abrir, selecione a biblioteca a ser adicionada:



Com isso, a nova biblioteca foi instalada. Para utilizá-la, basta ir em “Sketch -> Import Library…” e selecionar a biblioteca desejada:



Observe que o IDE adicionou no início do seu código a linha:

```
#include <dht.h>
```

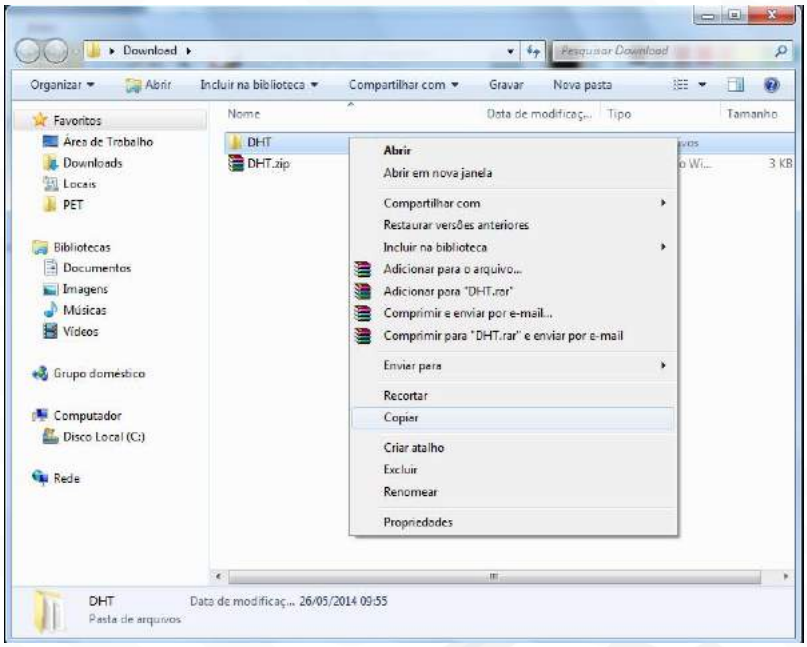
incluindo a biblioteca no seu programa.

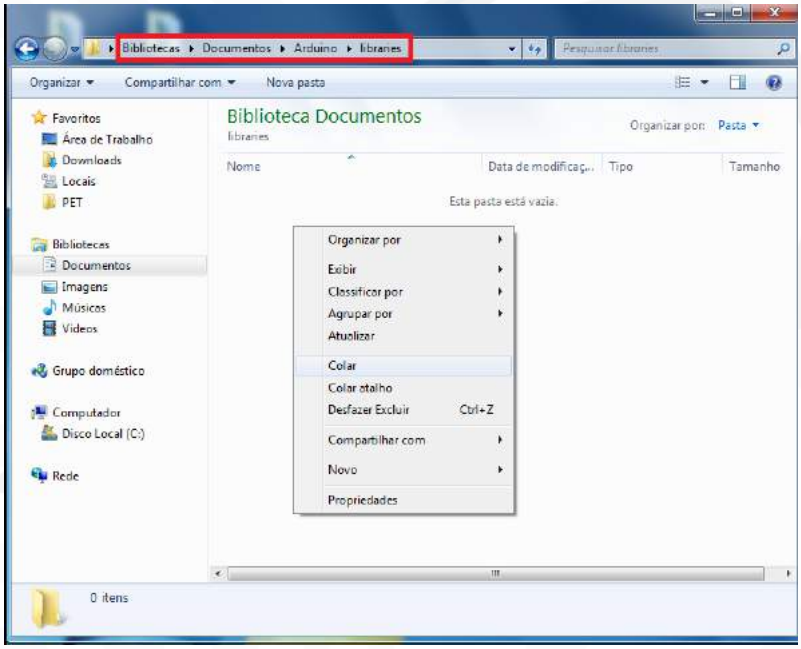
2ª opção) Instalar manualmente a biblioteca

Para instalar manualmente uma biblioteca, feche o IDE do Arduino e, em seguida, descompacte os arquivos da biblioteca. Se os arquivos .cpp e .h não estiverem dentro de uma pasta, crie uma e mova os arquivos para lá. Em seguida, basta mover a pasta para o local:

- Windows: “Meus documentos\Arduino\libraries”

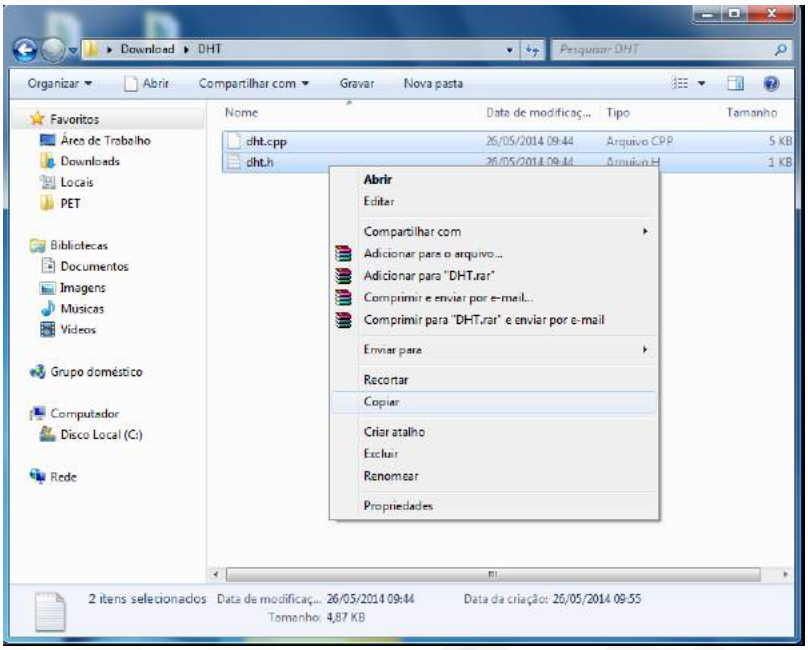
- Mac: “Documents/Arduino/libraries”

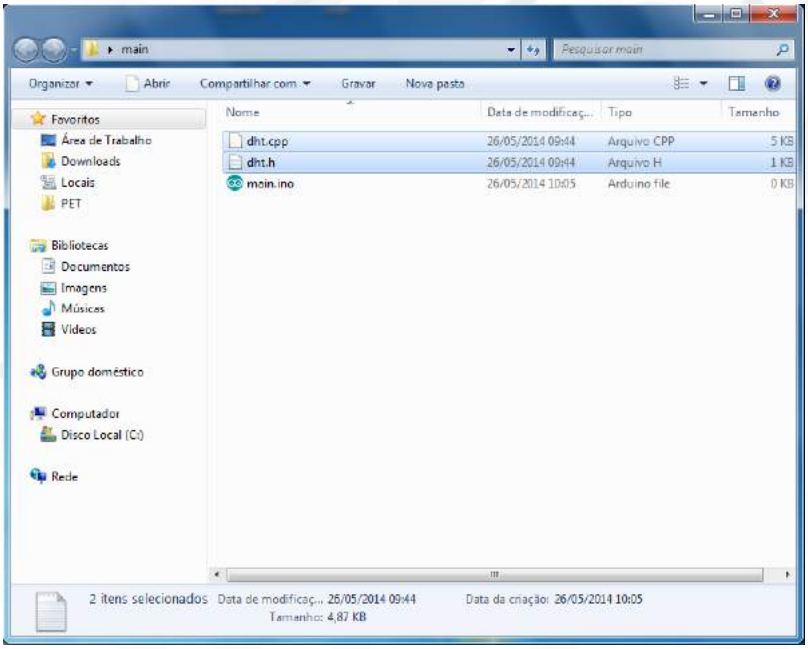




3ª opção) Incluir a biblioteca sem instalá-la

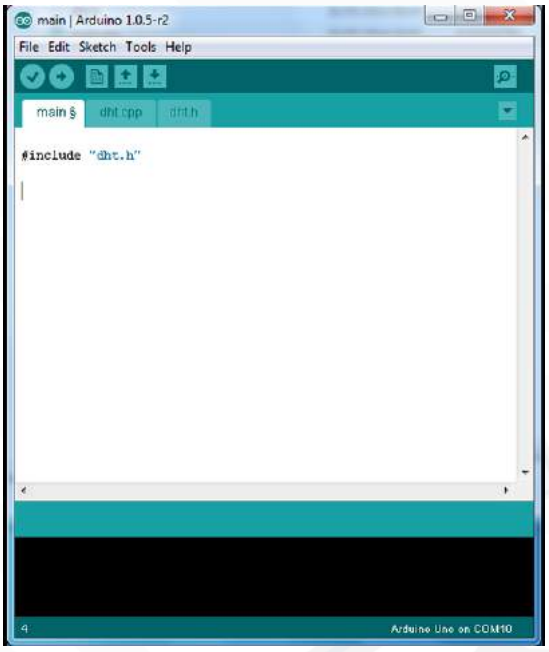
É possível também utilizar a biblioteca sem instalá-la no IDE do Arduino. Para isso, basta descompactar a biblioteca e colocar os arquivos .h e .cpp no mesmo diretório do programa.





Você terá que incluir no início do mesmo a linha:

```
#include "dht.h"
```



5.2. Experiência 4 - Mãos à obra – Usando o DHT11

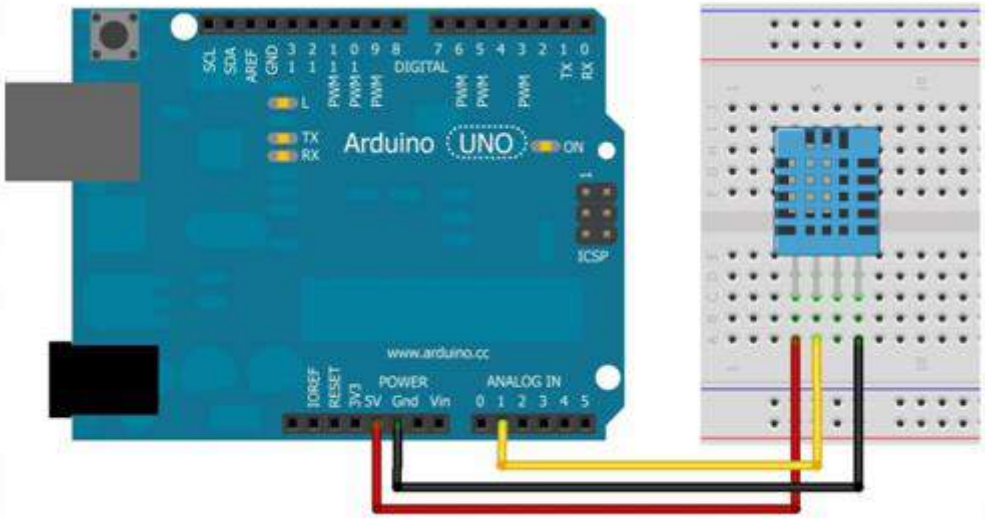
5.2.1. Ingredientes

Os itens que usaremos nessa experiência podem ser encontrados em nosso site (www.vidadesilicio.com.br). São eles:

- Fios Jumper's
- Protoboard
- Arduino Uno Rev3
- DHT11

5.2.2. Misturando os ingredientes

Agora vamos conectar os componentes do projeto. Para isso, desligue o cabo USB de seu Arduino e monte seu circuito conforme a figura a seguir.



5.2.3. Levando ao forno

Conecte seu Arduino ao computador e abra a IDE Arduino.

Antes de carregar um programa, você precisa selecionar qual porta você deseja usar para fazer carregar o programa no Arduino (upload). Dentro do Arduino IDE, clique no menu Ferramentas (tools) e abra o submenu Porta (Port). Clique na porta que seu Arduino está conectado, tal como COM3 ou COM4. Geralmente aparece o nome da placa Arduino : "COM3 (Arduino Uno)".

Você também precisa garantir que o tipo de placa apropriado está selecionado em Ferramentas (Tools) no submenu Placa (Board).

5.2.4. Preparando a cobertura

Crie um programa (sketch) e salve com o nome de "programa_dht11".

Com o seu programa salvo, escreva nele o código conforme escrito abaixo.

```
#include <dht.h> // Inclui a biblioteca no seu código

dht DHT; // Cria um objeto da classe dht
uint32_t timer = 0;

void setup()
{
  Serial.begin(9600); // Inicializa serial com taxa de transmissão de 9600 bauds
}

void loop()
{
  // Executa 1 vez a cada 2 segundos
  if(millis() - timer>= 2000)
  {

    DHT.read11(A1); // chama método de leitura da classe dht,
                    // com o pino de transmissão de dados ligado no pino A1

    // Exibe na serial o valor de umidade
    Serial.print(DHT.humidity);
    Serial.println(" %");

    // Exibe na serial o valor da temperatura
    Serial.print(DHT.temperature);
    Serial.println(" Celsius");

    timer = millis(); // Atualiza a referência
  }
}
```

Após escrever o código, clique em Upload para que o programa seja transferido para seu Arduino.

5.2.5. Experimentando o prato

Abra seu monitor IDE e veja o resultado. Casso tenha tudo ocorrido como esperado, você verá os valores lidos de umidade e temperatura.

5.3. Entendendo o Software

Nessa experiência usamos comandos já estudados anteriormente, então irei me ater apenas ao uso do millis.

- *millis()*

Retorna o número de milissegundos desde a placa Arduino começou a funcionar com programa atual. Este número irá saturar (voltar para zero), após, aproximadamente, 50 dias.

```
timer = millis();         //Atualiza a referência
```

Veja que usamos o millis como nossa referência de tempo. Toda vez que a diferença entre o millis e o timer for de2000 milissegundos, entraremos no if e o timer irá assumir o valor atual de millis.

Desta forma, o programa irá executar o que está dentro do if de 2000 em 2000 milissegundos, ou seja, 2 em 2 segundos.

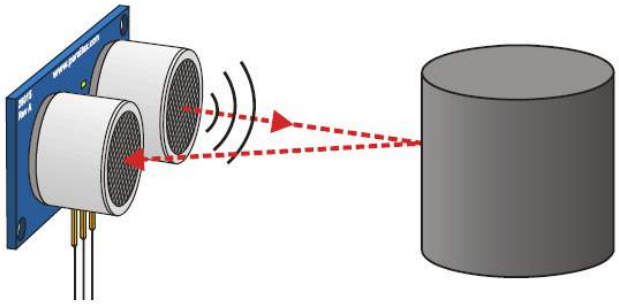
Esse tipo de estrutura é muito comum e será usada em outras experiências.

6. Sensor Ultrassônico

O sensor ultrassônico é amplamente utilizado em aplicações onde se deseja medir distâncias ou evitar colisões, como na robótica móvel e de reabilitação. Neste tutorial utilizaremos o sensor ultrassônico HC-SR04, que é o módulo ultrassom muito usado com Arduinos.

- *Como funciona?*

Tudo começa pela emissão de um pequeno pulso sonoro de alta frequência que se propagará na velocidade do som no meio em questão. Quando este pulso atingir um objeto, um sinal de eco será refletido para o sensor. A distância entre o sensor e o objeto pode então ser calculada caso saibamos o tempo entre a emissão e a recepção do sinal, além da velocidade do som no meio em questão. Afigura a seguir exemplifica o processo.



Para uma melhor medição da distância, a área do objeto na qual a onda será refletida deve ser de pelo menos 0,5 m^2.

6.1. Experiência5 – Medindo distância com o HC-SR04

Chega de conversa e vamos logo ao que interessa. Coloquemos o sensor para funcionar.

6.1.1. Ingredientes

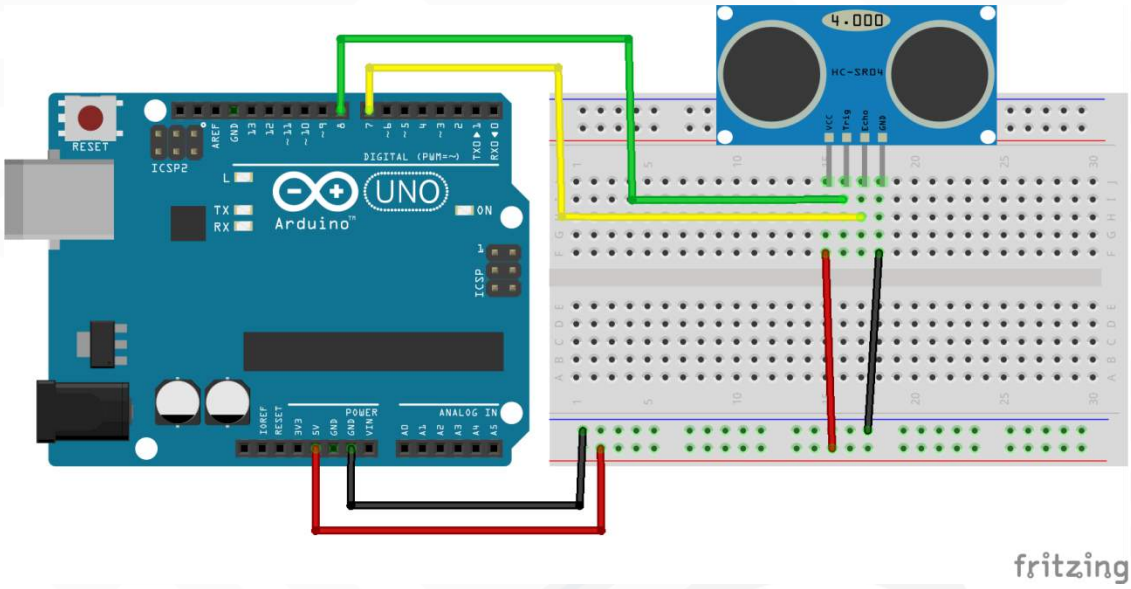
Os itens que usaremos nessa experiência podem ser encontrados em nosso site (www.vidadesilicio.com.br). São eles:

- HC-SR04
- Fios Jumper's
- Protoboard
- Arduino Uno

6.1.2. Misturando os ingredientes

Agora vamos conectar os componentes do projeto. Para isso monte seu circuito

conforme a figura a seguir. Garanta que seu Arduino esteja desligado durante a montagem.



O HC-SR04 possui 4 pinos sendo eles:

- Vcc – Deve ser conectado a um pino 5V do Arduino.
- Trig – Deve ser conectado a um pino digital configurado como saída. Utilizaremos o pino 8.
- Echo – Deve ser conectado a um pino digital configurado como entrada. Utilizaremos o pino 7.
- Gnd – Deve ser conectado a um pino GND do Arduino.

6.1.3. Levando ao forno

Conecte seu Arduino ao computador e abra a IDE Arduino. No menu Tools, certifique que a porta serial (serial port) está selecionada e se a placa configurada é a que você está usando (board).

6.1.4. Preparando a cobertura

Crie um no programa (sketch) e salve com o nome de “programa_ultrassom”.

Nesse exemplo utilizaremos a biblioteca Ultrasonic.h (Clique aqui para baixar). A utilização dessa biblioteca é bastante simples e você pode rever como adicioná-la no capítulo 5.

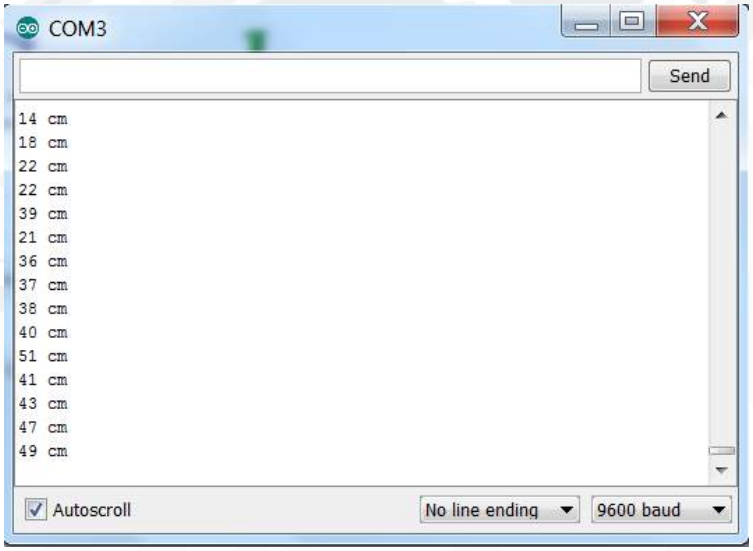
Com o seu programa salvo, escreva nele o código abaixo:

```
#include <Ultrasonic.h>
 Ultrasonic ultrassom(8,7); // define o nome do sensor(ultrassom)
//e onde esta ligado o trig(8) e o echo(7) respectivamente

// Esta função "setup" roda uma vez quando a placa e ligada ou resetada
 void setup() {
 Serial.begin(9600); //Habilita Comunicação Serial a uma taxa de 9600 bauds.
 long distancia;
 }

// Função que se repete infinitamente quando a placa é ligada
 void loop()
 {
   distancia = ultrassom.Ranging(CM);// ultrassom.Ranging(CM) retorna a distancia em
                                     // centímetros(CM) ou polegadas(INC)
   Serial.print(distancia); //imprime o valor da variável distancia
   Serial.println("cm");
   delay(100);
 }
```

Depois de escrever o código, clique em Upload para que o programa seja transferido para seu Arduino.

6.1.5. Experimentando o prato

Abra o Serial Monitor e coloque um obstáculo na frente do sensor ultrassônico. Se tudo deu certo, conforme você variar a distância do obstáculo em relação ao sensor, a distância medida aparecerá no serial monitor, tal como a figura a seguir.



6.2. Entendendo o Programa

- *Biblioteca Ultrasonic.h*

Na elaboração do software utilizaremos a biblioteca Ultrasonic.h. Essa biblioteca implementa as funcionalidades de um ultrassom, tornando sua utilização extremamente simples.

- *Declarando um Ultrassom*

Para o uso dessa biblioteca devemos definir o nome do sensor e em quais pinos estão conectados os pinos**trig** e **echo**.

```
Ultrasonic nomesensor(trig,echo);
```

No exemplo a seguir, o nome do sensor é **ultrassom**, o pino do trig está conectado na **porta 8** e o pino do echo na **porta 7**.

```
Ultrasonic ultrassom(8,7);
```

- *Medindo a distância*

Para ler a distância, basta chamar a função Ranging(CM). Para valor em centímetros, deve-se usar CM e para valor em polegadas, deve-se usar INC. Essa função retorna um valor de variável long.

```
long distancia = ultrassom.Ranging(CM); // distancia recebe o valormedido em cm
```

Observe que usamos o nome do sensor escolhido, no nosso caso **ultrassom**, seguido de ponto e o nome da função que queremos usar. Para o Arduino estamos falando que queremos usar determinada função do objeto mencionado, no nosso caso o **ultrassom**.

- *Imprimindo na porta serial*

Para imprimir o valor de distância lido usamos a função Serial.print().

```
Serial.print(distancia); //imprime o valor da variável distancia
Serial.println("cm");
```

Repare que na primeira linha imprimimos o valor da variável distância e na segunda linha imprimimos a palavra **cm**, que está entre aspas duplas. Sempre que quisermos imprimir um texto devemos colocá-lo dentro de aspas duplas, caso contrário o programa irá entender que o texto é o nome de uma variável.

Repare também que na segunda linha usamos o sufixo -ln depois de print. Esse sufixo informa que depois de escrito o texto, o programa deve pular uma linha.

6.3. Entendendo o Hardware

Como dito anteriormente, o sensor ultrassom mede a distância através da medição do tempo que uma onda leva para sair do emissor, colidir com um obstáculo e ser refletido, para, finalmente, ser detectado pelo receptor. Desta forma, podemos notar que nosso sensor possui dois cilindros metálicos que se assemelham a olhos em sua placa. São, na realidade, dois altos falantes: um trabalha como o emissor do sinal

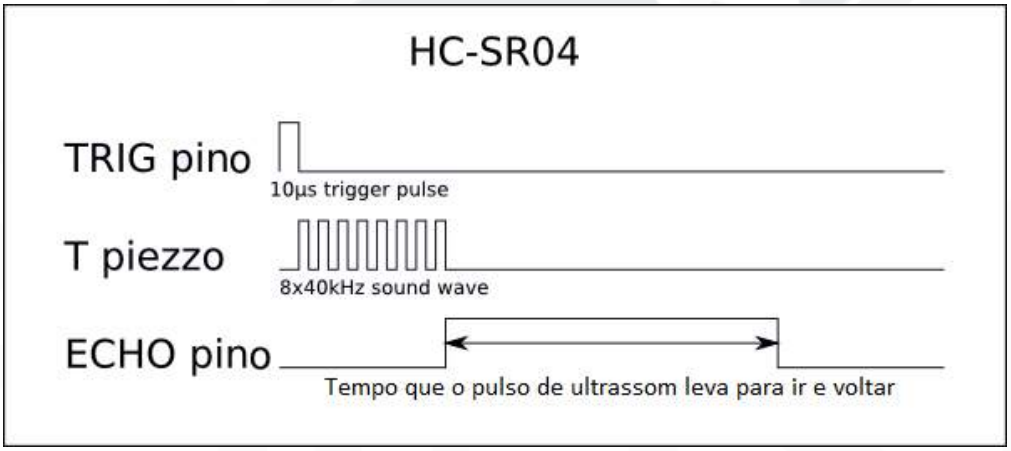
ultrassom e o outro como receptor.

- *Fazendo a leitura do tempo*

Para iniciarmos uma medição, o pino Trig, que funciona como gatilho do nosso sensor, deve receber um pulso de 5V por pelo menos 10 microssegundos. Isso fará com que o sensor emita 8 pulsos ultrassônicos em 40kHz (T piezzo) e o pino ECHO, que funcionará como nosso cronômetro, vai para 5V, iniciando assim a espera pelas ondas refletidas.

Assim que uma onda refletida for detectada, o pino Echo, que estava em 5V, será alterado para 0V. Desta forma, o período que o pino Echo fica em 5V é igual ao tempo que a onda emitida leva para ir até o obstáculo e voltar.

Veja a imagem abaixo para entender o que acontece no sensor.



Por fim, para sabermos quão longe nosso obstáculo está, basta contarmos quanto tempo (T) a tensão no pino Echo ficou em 5V. Em posse desse tempo, sabendo que ele é o dobro do tempo de ida e volta da onda do sensor até o obstáculo, e, considerando a velocidade do som igual a 340,29 m/s, temos:

$$Vsom = 340{,}29\ \frac{m}{s}$$

$$Vsom = \frac{\Delta S}{\Delta T} = 340{,}29\ \frac{m}{s}$$

$$\Delta T = T/2$$

$$\frac{\Delta S}{T/2} = 340{,}29$$

Desta forma, temos que a distância até o obstáculo é igual a ΔS. Assim:

$$\boldsymbol{\Delta S = \frac{340,29 * T}{2}}$$

A função Ultrasonic.h faz exatamente o procedimento descrito, retornando apenas o valor da distância.

6.4. Desafio

Sabendo como funciona um sensor ultrassônico, faça o programa que meça a distância sem o auxílio da biblioteca. Consulte a tabela de códigos em anexo para te ajudar na confecção do programa.

No nosso tutorial sobre o sensor em nosso blog, temos uma possível solução para esse desafio, mas só o use depois de tentar resolver sozinho. Esse desafio irá exercitar seu entendimento do uso da programação em paralelo com a eletrônica.

7. Display de Cristal Líquido (LCD)

- *As máquinas querem conversar*

Por muitas vezes precisamos coletar dados ou interagir com sistemas automatizados. Dessa forma, precisamos de um caminho que torne essa interação possível. O nome desse caminho de comunicação chama-se IHM (Interface Homem Máquina).

> IHM é um caminho pelo qual o homem e o computador podem se comunicar e interagir, visando atingir um objetivo comum.

Tipicamente, uma IHM é dada por algum tipo de saída (indicadores de velocidades, monitores, displays, alto-falantes, etc) e algum tipo de entrada (botões, touchscreen, microfone, câmera, entre outros). Quanto mais fácil for para coletar informações e introduzir entradas em um sistema automatizado, mais trivial será a interação dele com o usuário.

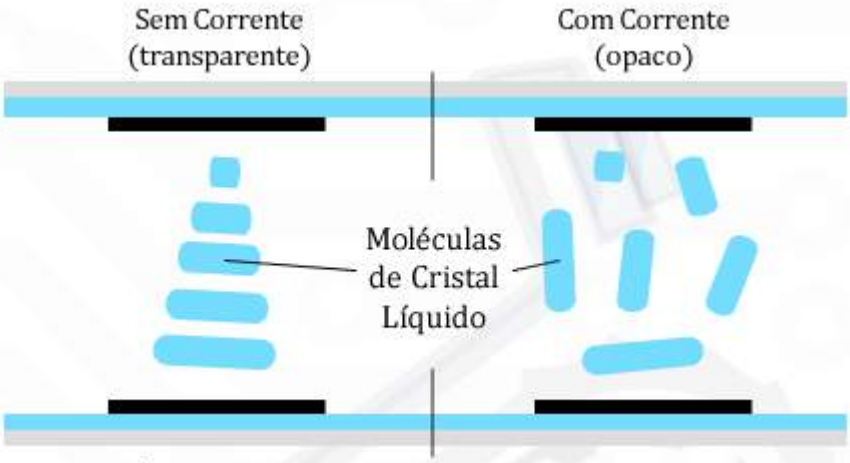
É aí que entram os displays! Boa parte dos dispositivos automatizados que utilizamos possuem ecrãs. O motivo é claro: Eles podem apresentar ao usuário várias informações de forma rápida. Além disso, um display pode apresentar vários tipos de saídas, desde textos ou números até imagens em movimento.

Até aqui, usamos o monitor serial como IHM para a visualização dos valores lidos nos sensores, porém, não é interessante dependermos de um computador para que possamos ler esses valores. Para essa função, podemos usar um LCD.

- *O que é um LCD*

O LCD (Display de Cristal Líquido) é um dispositivo gráfico muito poderoso na criação de interfaces com o usuário. Amplamente utilizado nos mais diversos tipos de

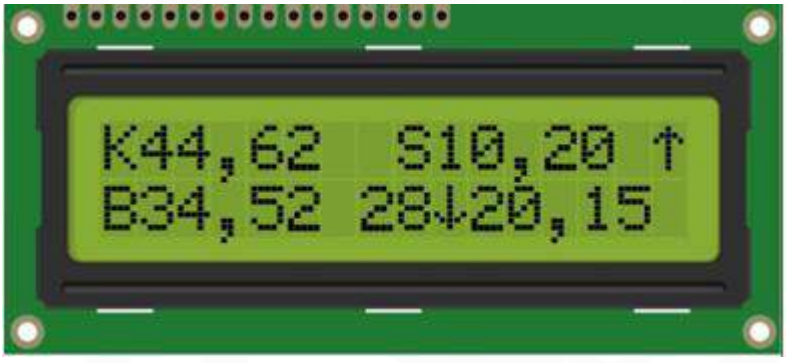
projetos, esses displays são formados por uma fina camada de cristal líquido entre duas placas de vidro, com uma fonte de luz fluorescente ou de LEDs por baixo de toda essa estrutura. A formação de caracteres e imagens ocorre devido ao fato do cristal líquido, naturalmente transparente, se tornar opaco ao receber uma carga elétrica, impedindo a passagem de luz.



Existem, basicamente, dois tipos de LCDs:

- **Caracter:** Esses tipos de LCD permitem apenas a escrita de caracteres, números e pequenos símbolos criados pelo usuário. Alguns tamanhos comuns para esse tipo de display são: 8×2, 16×2, 20×4, 24×4, 40×2, entre outros;
- **Gráficos:** Os LCDs gráficos possuem resoluções bem maiores e permitem a exibição de figuras e imagens. Alguns tamanhos comuns para esse tipo de display são: 122×32, 128×64, 240×64, 240×128, entre outros.

Neste tutorial será ensinado como utilizar um display LCD de caractere no Arduino por meio da biblioteca LiquidCrystal, que acompanha o seu IDE. Para isso, será utilizado um display 16×2 (2 linhas e 16 colunas).



A biblioteca LiquidCrystal possui diversas funções para utilização do LCD. Explicaremos as principais funções nesse tutorial, mas antes, iremos direto para a prática.

7.1. Experiência6 – Montando seu primeiro projeto com um LCD

Chega de conversa e vamos logo ao que interessa. Vamos colocar o sensor para funcionar.

7.1.1. Ingredientes

Os itens que usaremos nessa experiência podem ser encontrados em nosso site (www.vidadesilicio.com.br). São eles:

- Display LCD 1602
- Fios Jumper's
- Protoboard
- Potenciômetro 10kOhm
- Arduino Uno ou outra placa de desenvolvimento Arduino

7.1.2. Misturando os ingredientes

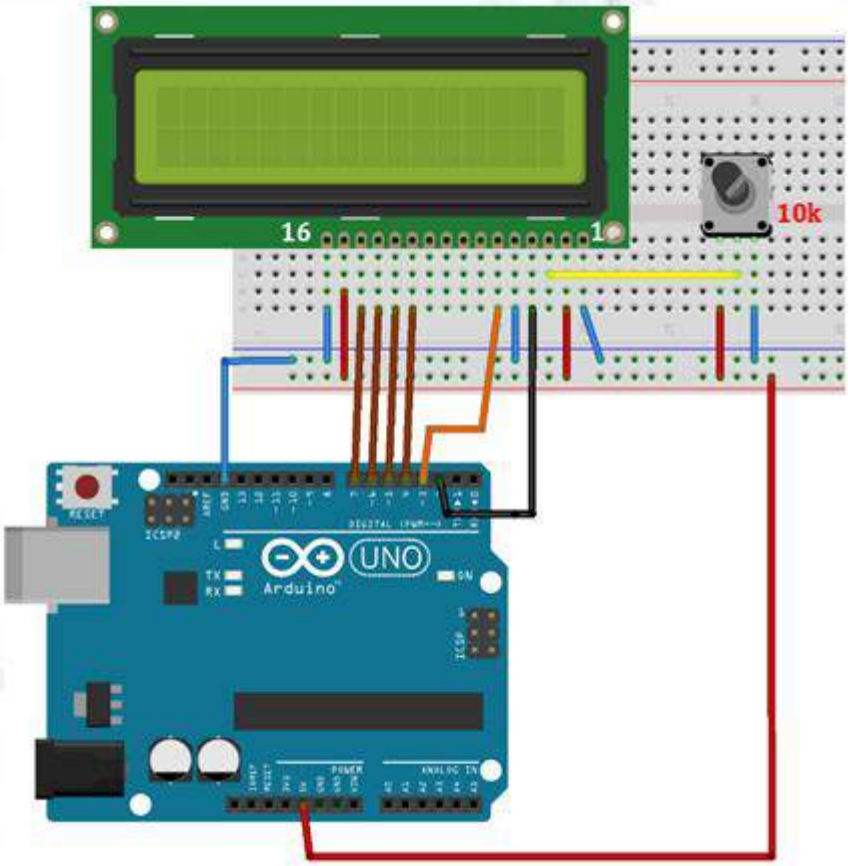
Agora vamos conectar os componentes do projeto. Para isso monte seu circuito conforme a figura a seguir. Garanta que seu Arduino esteja desligado durante a montagem.

O display LCD 1602 possui 16 pinos, sendo eles:

1	Vss
2	Vdd
3	V0 (Ajuste de Contraste)
4	RS
5	R/W (Leitura/Escrita)
6	Enable (Habilita escrita no LCD)
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5

13	DB6
14	DB7
15	Anodo – Luz de Fundo
16	Catodo – Luz de Fundo

No nosso exemplo não usaremos os pinos 7,8,9 e 10 do nosso módulo. Dessa forma, teremos os outros pinos conectados da seguinte forma:



Abaixo você pode conferir onde cada pino será conectado:

Pino LCD	Pino Arduino
1	GND
2	5V
3	Divisor de Tensão potênciometro
4	Porta digital 2
5	GND
6	Porta digital 3
7	-
8	-
9	-
10	-
11	Porta digital 4
12	Porta digital 5
13	Porta digital 6
14	Porta digital 7
15	5V
16	GND

7.1.3. Levando ao forno

Conecte seu Arduino ao computador e abra a IDE Arduino. No menu Tools, certifique que a porta serial (serial port) está selecionada e se a placa configurada é a que você está usando (board).

7.1.4. Preparando a cobertura

Crie um no programa (sketch) e salve com o nome de “programa_LCD”.

Neste exemplo, utilizaremos a biblioteca LiquidCrystal que já acompanha a IDE Arduino.

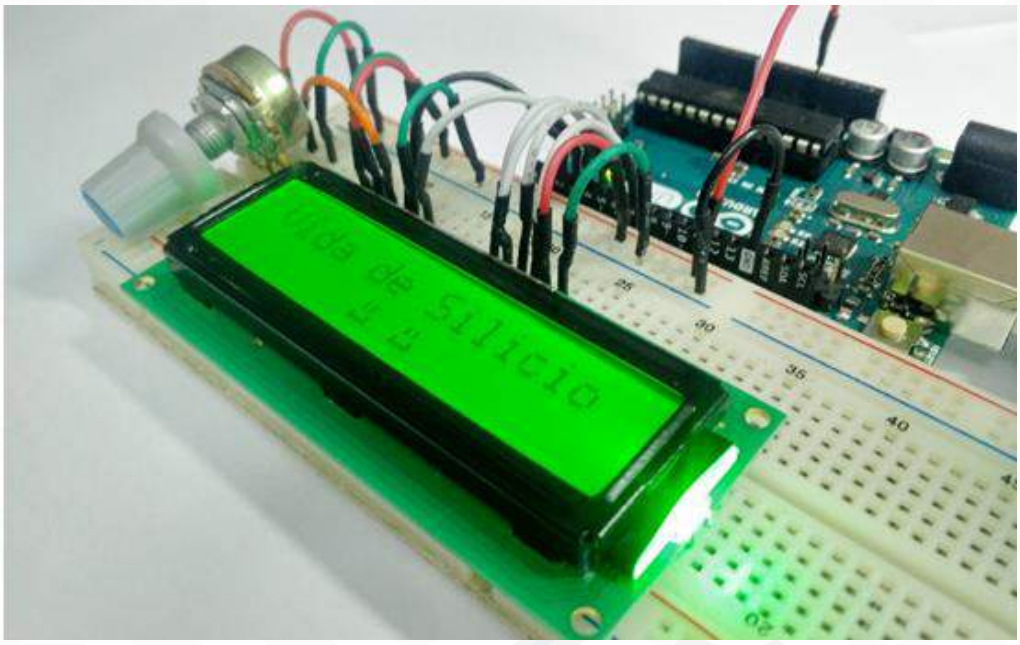
Com o seu programa salvo, escreva nele o código a seguir e depois de escrever o código, clique em Upload para que o programa seja transferido para seu Arduino.

```
#include <LiquidCrystal.h>     //inclui biblioteca no programa

LiquidCrystal lcd(2,3,4,5,6,7);
 /*Cria objeto lcd da classe LiquidCrystal
RS     2
Enable 3
DB4    4
DB5    5
DB6    6
DB7    7
*/

//Cria um smile
byte smile[8] = {
  B11111,
  B00000,
  B01010,
  B00000,
  B10001,
  B01110,
  B00000,
  B00100,
};

//Cria outro smile
byte smile2[8] = {
  B00000,
  B00000,
  B01010,
  B01010,
  B00000,
  B10001,
  B11111,
  B00000,
};

void setup(){
  lcd.begin(16,2);            //Inicializa display de 2 linhas x 16 colunas
  lcd.createChar(0, smile);   //Cria o smile e o associa ao 0
  lcd.createChar(1, smile2);  //Cria o smile2 e o associa ao 1
  lcd.home();                 //Posiciona cursor no canto superior esquerdo
  lcd.print("3 ");            //Executa uma pequena contagem regressiva
  delay(500);
  lcd.print("2 ");
  delay(500);
  lcd.print("1 ");
  delay(500);
  lcd.clear();                //Limpa a tela do LCD
  lcd.print("Vida de Silício");  //Escreve Vida de Silício
  lcd.setCursor(6,1);         //Posiciona o cursor na posição (6,1)
  lcd.write(byte(0));         //Escreve o smile
  lcd.setCursor(8,1);         //Posiciona o cursor na posição (8,1)
  lcd.write(1);               //Escreve smile2
}

void loop(){

}
```

7.1.5. Experimentando o prato

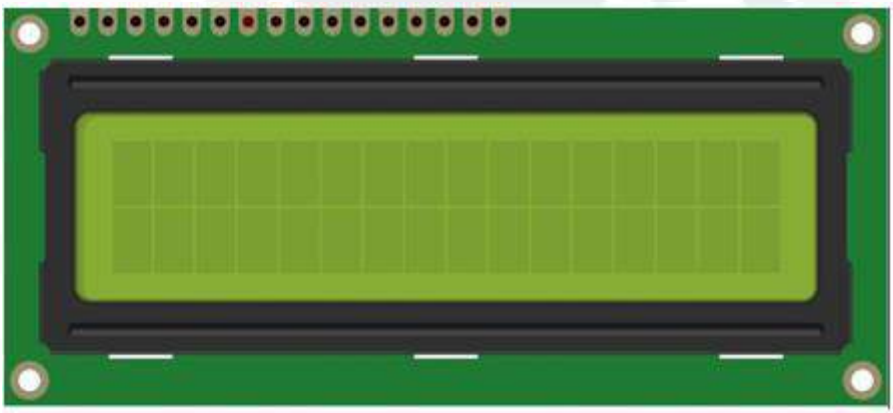
Depois que o Arduino carregar o programa, caso esteja tudo certo, você verá em seu display uma contagem regressiva e por fim o texto “Vida de Silício”.

O resultado será semelhante ao da imagem a seguir:

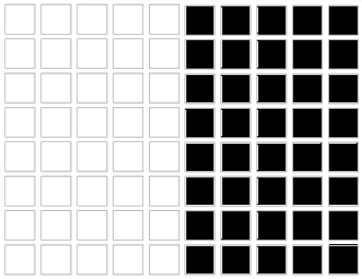


7.2. Entendendo o Hardware

Nesse exemplo usamos o LCD 1602, que possui 16 colunas e 2 linhas. Isso quer dizer que esse display pode apresentar 16x2, ou seja, 32 caracteres em sua tela. Na imagem abaixo podemos notar nitidamente a disposição dos caracteres na tela.

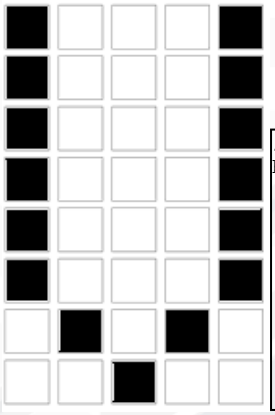


Cada caractere é composto de uma matriz de 5 colunas por 8 linhas de pixels. O estado desses pixels, “ligados” ou “desligados”, desenharam o caractere. Veja a seguir como estão dispostos esses pixels em um caractere.



Definimos os estados dos pixels através das entradas DB's do módulo LCD. O Arduino enviará 8 bytes através dessas portas para que seja definido o caractere.

A seguir você pode verificar como é formado a letra V.



```
//Letra V
byte V[8] = {
  B10001,
  B10001,
  B10001,
  B10001,
  B10001,
  B10001,
  B01010,
  B00100,
};
```

A biblioteca que usamos já possui essas combinações para caracteres alfanuméricos, dessa forma não é necessário definir no programa cada caractere que será usado. Porém, caso você queira criar um caractere personalizado, também será possível, tal como o smile que usamos.

7.3. Entendendo o Programa

Para o uso do módulo display LCD 1602 temos que entender o uso de sua biblioteca. Para esse exemplo, explicaremos a biblioteca LiquidCrystal.

7.3.1. Biblioteca LiquidCrystal

- *LiquidCrystal*

É o construtor que cria um objeto da classe LiquidCrystal. Ou seja, define em quais pinos do Arduino o módulo está conectado.

Ele é sobrecarregado de 4 maneiras diferentes:

```
LiquidCrystal tela(RS, Enable, DB4, DB5, DB6, DB7)
LiquidCrystal tela(RS, R/W, Enable, DB4, DB5, DB6, DB7)
LiquidCrystal tela(RS, Enable, DB0, DB1, DB2, DB3, DB4, DB5, DB6, DB7)
LiquidCrystal tela(RS, R/W, Enable, DB0, DB1, DB2, DB3, DB4, DB5, DB6, DB7)
```

Onde **tela**, é o nome escolhido para o seu LCD.

Caso o pino R/W não seja passado ao construtor, ele deve ser conectado ao pino terra para permitir a escrita no LCD. Observe que não é necessário enviar todos os pinos de dados ao construtor, pois o Arduino consegue controlar o display utilizando apenas os 4 últimos pinos.

Observe que em nosso exemplo usamos a primeira opção usamos o nome LCD:

```
LiquidCrystal lcd(2,3,4,5,6,7);
 /*Cria objeto lcd da classe LiquidCrystal
RS     2
Enable 3
DB4    4
DB5    5
DB6    6
DB7    7
*/
```

- *begin(colunas, linhas)*

Inicializa a interface com o LCD, recebendo como parâmetros o número de colunas e linhas do display. Deve ser chamada antes de qualquer outro método da classe.

```
lcd.begin(colunas,linhas);
```

Como nosso display possui 16 colunas e 2 linhas, temos:

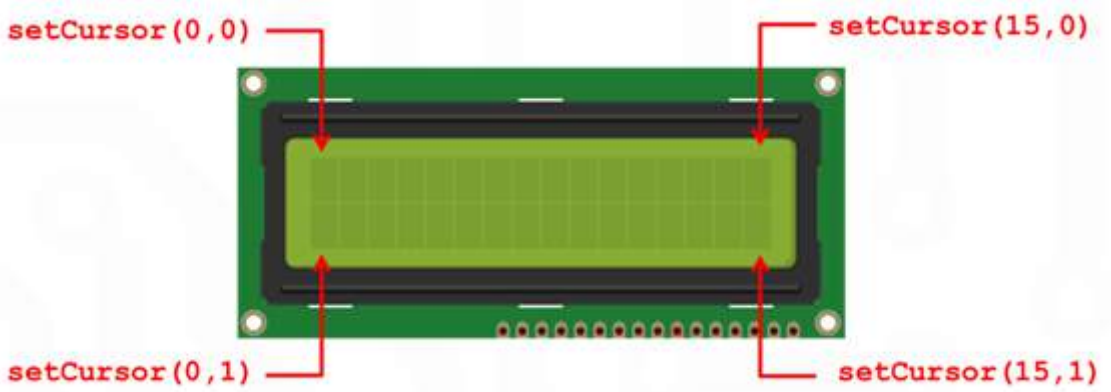
```
lcd.begin(16,2);            //Inicializa display de 2 linhas x 16 colunas
```

Observe que usamos o nome do display, no nosso caso **lcd**, seguido de ponto e o nome da função que queremos usar. Para o Arduino estamos falando que queremos usar determinada função do objeto mencionado, no nosso caso o **lcd**.

- *setCursor(coluna, linha)*

Posiciona o cursor do LCD nas coordenadas passadas pelo parâmetro, para que o texto possa ser escrito na posição desejada.

```
lcd.setCursor(6,1);         //Posiciona o cursor na posição (6,1)
```



- *home*

Posiciona o cursor no canto superior esquerdo do display. Equivale ao setCursor(0,0).

```
lcd.home();          //Posiciona o cursor na posição (0,0)
```

- *print(conteúdo, base)*

Escreve o conteúdo (char, int, string, byte ou long) na tela do LCD, na posição atual do cursor. O parâmetro base é opcional e indica apenas a base em que os números serão impressos (BIN para binário, DEC para decimal, OCT para octal e HEX para hexadecimal).

Ao término da escrita, o cursor é movido para a próxima posição.

```
lcd.print("Vida de Silício");  //Escreve Vida de Silício
```

- *clear()*

Limpa a tela do LCD e posiciona o cursor na extremidade superior esquerda.

```
lcd.clear();                 //Limpa a tela do LCD
```

- *createChar(número, caractere)*

Cria um caractere customizado para ser utilizado no display, até, no máximo, 8 caracteres. O símbolo criado é formado por um array de 8 bytes, onde os 5 bits menos significativos de cada byte determinam os pixels de cada linha.

```
nomedolcd.createChar(numero, caractere);
```

O parâmetro **número** recebe um inteiro de 0 a 7, que simboliza qual caractere está sendo criado.

O parâmetro **caractere** recebe um vetor de 8 bytes que desenham o caractere, tal como apresentado abaixo:

```
byte smile[8]={
  B11111,
  B00000,
  B01010,
  B00000,
  B10001,
  B01110,
  B00000,
  B00100,
};
```

1	1	1	1	1
0	0	0	0	0
0	1	0	1	0
0	0	0	0	0
1	0	0	0	1
0	1	1	1	0
0	0	0	0	0
0	0	1	0	0

Veja como ficou o no nosso exemplo:

```
lcd.createChar(0, smile);   //Cria o smile e o associa ao 0
lcd.createChar(1, smile2);  //Cria o smile2 e o associa ao 1
```

Onde temos os caracteres smile e smile 2 definidos da seguinte forma:

```
//Cria um smile
byte smile[8] = {
  B11111,
  B00000,
  B01010,
  B00000,
  B10001,
  B01110,
  B00000,
  B00100,
};

//Cria outro smile
byte smile2[8] = {
  B00000,
  B00000,
  B01010,
  B01010,
  B00000,
  B10001,
  B11111,
  B00000,
};
```

Para o caractere 0 é necessário que, no momento de imprimi-lo, faça-se um casting do número do caractere criado para um byte.

```
lcd.write(byte(0));          //Escreve o smile
```

O que não acontece com os demais.

```
lcd.write(1);          //Escreve smile2
```

- *write(caracter)*

Escreve um caractere no LCD na posição atual do cursor, movendo o cursor para a próxima posição.

```
lcd.write(byte(0));          //Escreve o smile
```

7.4. Experiência 8 – Medindo temperatura com um LCD

Agora que já sabemos usar o LCD, podemos mostrar um valor de uma variável medida. Para esse exemplo usaremos o LM35 já estudado anteriormente no capítulo 4.

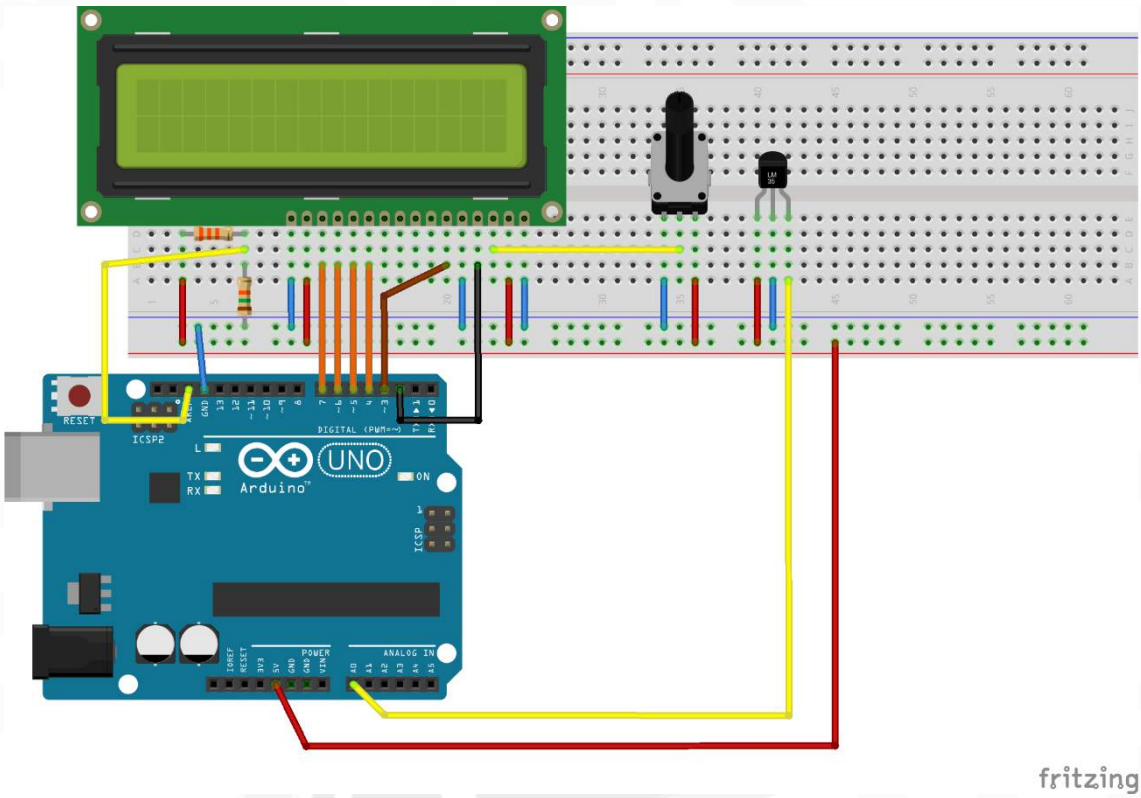
7.4.1. Ingredientes

Os itens que usaremos nessa experiência podem ser encontrados em nosso site (www.vidadesilicio.com.br). São eles:

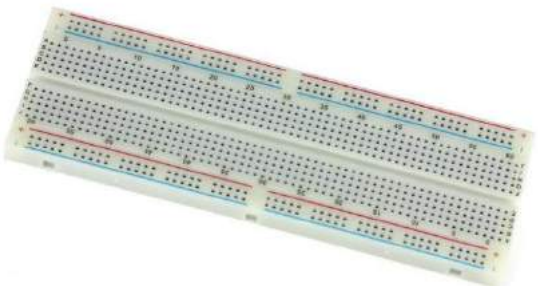
- Display LCD 1602
- Fios Jumper's
- LM35
- Protoboard
- Potenciômetro linear 10kOhm
- Arduino Uno ou outra placa de desenvolvimento Arduino

7.4.2. Misturando os ingredientes

Agora vamos conectar os componentes do projeto. Para isso poderemos aproveitar a montagem da experiência 6 e apenas acrescentar o LM35 (Referencia alterada)



Tome um especial cuidado em relação ao barramento vermelho e azul de sua protoboard. Muitas vezes ele não é continuo de um lado a outro tal como o do esquemático. Tal como pode ser observado na figura abaixo.



7.4.3. Levando ao forno

Conecte seu Arduino ao computador e abra a IDE Arduino. No menu Tools, certifique que a porta serial (serial port) está selecionada e se a placa configurada é a que você está usando (board).

7.4.4. Preparando a cobertura

Crie um no programa (sketch) e salve com o nome de “programa_LCD_LM35”.

Utilizaremos novamente a biblioteca LiquidCrystalque já acompanha a IDE Arduino, tal como na experiência anterior.

Com o seu programa salvo, escreva nele o código a seguir.

```
#include <LiquidCrystal.h>     //Inclui biblioteca no programa

uint32_t timer;

/*
Cria objeto lcd da classe LiquidCrystal
RS     2
Enable 3
DB4    4
DB5    5
DB6    6
DB7    7
*/
LiquidCrystal lcd(2,3,4,5,6,7);

void setup(){

  analogReference(EXTERNAL);  // Muda a referência para a tensão no pino AREF

  lcd.begin(16,2);            //Inicializa display de 2 linhas x 16 colunas

}

void loop(){

  //Executa a cada 1 segundo
  if(millis() - timer > 1000)
  {
    // Atualiza o valor lido pelo conversor
    float valorLido = analogRead(A0);

   // Converte a temperatura lida para graus Celsius
   float temp = valorLido/1024*1562.5/10;

   lcd.clear();     //Limpa o display

   //Imprime a temperatura no display
   lcd.print("Temperatura: ");
   lcd.print(temp);

   timer = millis();         //Atualiza a referência
}

}
```

Depois de escrever o código, clique em Upload para que o programa seja transferido para seu Arduino.

7.4.5. Experimentando o prato

Depois que o Arduino carregar o programa, caso esteja tudo certo, você verá em seu display o valor de temperatura medido pelo LM35.

7.5. Entendendo o Programa

Nessa experiência usamos comandos já estudados anteriormente. A estrutura do if contabilizando o tempo é a mesma usada na experiência do capitulo 5.

7.6. Desafio

A. Faça um sensor de ré usando um sensor ultrassom, um LCD e um buzzer.
B. Mostre o valor de distância no display LCD e, caso a distância seja menor que 20 cm, faça o buzzer alarmar.

8. Super desafio

Somando tudo que você aprendeu na apostila Arduino Básico vol.1 e 2, nós o desafiamos a fazer um sistema de automação residencial. O seu sistema deve:

A. Detectar a presença de uma pessoa usando o sensor ultrassom. (Você também poderá utilizar um sensor de presença PIR que, apesar de não ter sido explicado em nossa apostila, é muito simples de usar;
B. Detectar um incêndio através da temperatura muito alta da casa usando um LM35. (Existem outros sensores mais adequados para esse tipo de aplicação, tais como o sensor de chama. Caso queira, sinta-se à vontade para usá-lo);
C. Detectar se está de noite através de um sensor de luz LDR;
D. Possuir um botão para ligar e desligar um alarme;
E. Acender 1 LED de alta intensidade, caso seja noite;
F. Um alarme, usando o buzzer, deverá tocar, caso a temperatura do LM35 seja muito alta;
G. Esse alarme também deve tocar, caso acionado e uma pessoa seja detectada;
H. Medir umidade através do DHT11;
I. Ter um display LCD que mostrará os valores lidos em cada sensor, se o alarme está ligado ou desligado e o motivo do alarme, caso dispare.

9. Apêndice - Tabela de consulta

Funções Matemáticas e de tempo	
delay(t)	O programa tem uma pausa de t milissegundos
delayMicroseconds(t)	O programa tem uma pausa de t microssegundos
millis()	Retorna o tempo, milissegundos, desde que o programa começou a rodar (Reset 9h)
randomSeed(referecia)	Gera uma referencia para o primeiro numero aleatório (Função setup)
random(min,max)	Gera um valor pseudoaleatório int entre mín e máx (a função acima é necessária)
abs(x)	Retorna o módulo(valor absoluto) do numero real passado como parâmetro
map(valor,min1,max1,min1 ,max2)	Converte uma valor inserido em uma faixa de valores para um proporcional em uma nova faixa de valores
sin(x)	Retorna o seno de x(rad)

Entradas Analógicas	
anologRead(Pino)	Lê entrada analógica 0-5V transformando 10 bit's (resolução 4,9mV)
Pinos analógicos podem ser usados como porta digitais usando a função pinMode(), quando usado como porta analógica não necessitam de configuração .	

Saídas/entradas Digitais e PWM	
pinMode(porta,Tipo)	Define se a porta será uma entrada(TIPO=INPUT) ou uma saída(TIPO= OUTPUT).
digitalWriter (pino, VL)	Coloca 0V(VL =LOW) ou 5V(VL = HIGH) na saída.
digitalRead(pino)	Lê o sinal digital no pino citado.
analogWrite(pino, x)	saída PWM 500Hz (0 <= x <=255).
analogWrite(pino, x)	saída PWM 500Hz (0 <= x <=255).
tone(pino,frequência,duração)	Gera uma frequência no pino durante um determinado tempo.
tone(pino,frequência)	Gera uma frequência no pino até que ocorra um comando de mudança de Freq.
noTone(pino)	Cessa a geração do tom no pino.
pulseIn(pino,valor,espera)	Mede a largura em microssegundo de um pulso no pino digital, "valor" é o tipo de pulso a ser medido (LOW ou HIGH), espera (opcional) faz com que a medida do pulso só comece após o tempo em microssegundos especificado .
attachInterrupt(pino,função, modo)	É uma interrupção, ou seja, caso a condição "modo" ocorra no pino especificado a função é executada imediatamente.

LOW	Dispara a interrupção quando o pino está em 0
CHANGE	Dispara sempre q o pino muda de estado(borda 0-> 1 ou vice-versa)
RISING	Somente borda de subida (0 para 1)
FALLING	Somente borda de descida (1 para 0)

Variáveis		
Tipo de dado	**RAM**	**Intervalo numérico**
boolean	1 byte	0 a 1 (false ou true)
int	2 bytes	-32.768 a 32.767
unsigned int	2 bytes	0 a 65.535
Word	2 bytes	0 a 65.535
Char	1 byte	-128 a 127
usigned char	1 byte	0 a 255
Byte	1 byte	0 a 255
void keyword	N/A	N/A
Long	4 bytes	-2.147.483.648 a 2.147.483.647
Usigned long	4 byte	0 a 4.294.967.295
float	4 byte	-3,4028235e+38 a 3,4028235e+38
Double	4 byte	-3,4028235e+38 a 3,4028235e+38
string	1 byte + x	Sequência de caracteres
array (vetor)	1byte + x	Sequência de variáveis
tipo var nome Matriz[nº de posições]		

Comunicação Serial	
Serial.begin(TAXA)	Habilita a porta serial e fixa a taxa de transmissão (função setup)
Serial.end()	Desabilita a porta serial para permitir o uso dos pinos digitais
Serial.flush()	Libera caracteres que estão na linha serial, deixando-a vazia e pronta p/ ent/saídas
Serial.available()	Retorna o numero de bytes disponíveis para leitura no buffer da porta serial.
Serial.read()	Lê o primeiro byte que está no buffer da porta serial
Serial.print('valor',formato)	Envia para a porta serial um caractere ASCII
Serial.println('valor',formato)	O mesmo que o anterior, porem pula uma linha

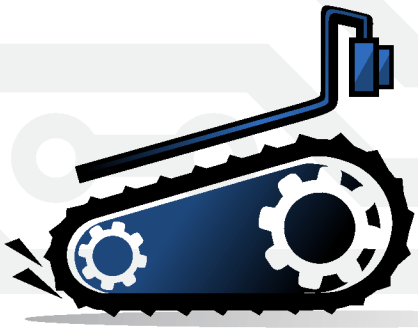
Operadores de Comparação	
==	Igual
!=	Diferente
<	Menor
>	Maior
>=	Maior ou igual
<=	Menor ou igual

Operadores Lógicos	
&&	AND
||	OR
!	NOT

Símbolos Compostos	
x++	Incrementa x
x--	Decrementa x
x+=y	x = x+y
x-=y	x = x-y
x*=y	x = x*y

x/=y	x = x/y
Símbolos	
{}	Entre as chaves fica o conteúdo da função
;	Final de um comando/linha
//	Linha de comentário
/* ... */	Comentário de varias linhas

Funções	
If(condição) **{}** **else** **{}**	Função Se e Se não
if(condição) **{}** **else if(condição 2)** **{}**	Função Se em cascata
switch(expressão){ **case expressão = x: Bloco1;** **break;** **case expressão = y: Bloco2;** **break;** **default: bloco3** **}**	Função Caso
while(condição){bloco funções}	Função enquanto
do{ **bloco de instruções** **}** **while(condição);**	Função enquanto, ela é executada pelo menos uma vez.
for(var;condição;incremento) **{}**	Função para
(condição) ? bloco1:bloco2;	Operador ternário '?' caso condição seja verdadeira ele executa o bloco 1, caso contrario, executa o bloco 2. Ex.: y = (x >10)? 15:20; // caso x>10 y=15, caso contrario, y = 20



Vida de Silício

Robótica e Sistemas Digitais

Fundada em março de 2014 com a filosofia de promover a mudança, o site Vida de Silício é uma empresa que trabalha no ramo de Robótica e Sistemas Digitais fornecendo produtos de excelente qualidade de diversas marcas para estudantes, hobbystas, Instituições de Ensino e Empresas de Tecnologia.

Temos como missão disseminar e promover a eletrônica e a robótica, vendendo produtos com preços justos e ensinando como usá-los de forma didática e organizada. Estimulando assim, que mais pessoas exercitem sua criatividade, produzindo tecnologias que possam mudar o cotidiano da humanidade.

Queremos estar presentes em cada nova ideia, em cada novo desafio e em cada conquista daqueles que promovem a mudança das mais variadas formas usando como ferramenta a tecnologia.

ARDUINO

adafruit INDUSTRIES

DISTRIBUIDOR OFICIAL

vidadesilicio.com.br

blog.vidadesilicio.com.br

contato@vidadesilicio.com.br

@vidadesilicio

fb.com/vidadesilicio