Vida de Silício

Robótica e Sistemas Digitais

Apostila Arduino Básico

Vol.3

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. Divirta-se!!

Atenciosamente,

Equipe Vida de Silício

Sumário

1. Introdução

1.1. O que é um Robô?

> *"E disse Deus: Façamos o homem à nossa imagem, conforme a nossa semelhança; e domine sobre os peixes do mar, e sobre as aves dos céus, e sobre o gado, e sobre toda a terra, e sobre todo o réptil que se move sobre a terra. "*
>
> *Bíblia - Gênesis 1:26*

Analisando esse versículo da Bíblia de um ponto de vista filosófico, podemos criar uma analogia dele conosco e nossos robôs. Hoje, no papel de criadores, construímos robôs que desempenham funções que são naturalmente executadas por nós ou por outros seres vivos, ou seja, semelhantes a nós. Em suma, a natureza é nossa maior fonte de inspiração.

Pense bem, os seres vivos povoam a terra a milhares de anos, por que não copiar o que já está dando certo na natureza? Vamos aproveitar todo esse conhecimento e tecnologia desenvolvidos a partir de milênios de seleção natural.

Portanto, se queremos construir dispositivos dotados de alguma inteligência o primeiro passo é olharmos para nós mesmos, o próprio ser humano.

1.1.1. Homo Sapiens

O homem, ou homo sapiens, do latim "homem sábio", é a denominação dada à espécie animal que somos. Com a vida corrida que vivemos, por muitas vezes não lembramos de que somos animais, dotados de uma máquina biológica incrível, o corpo humano.

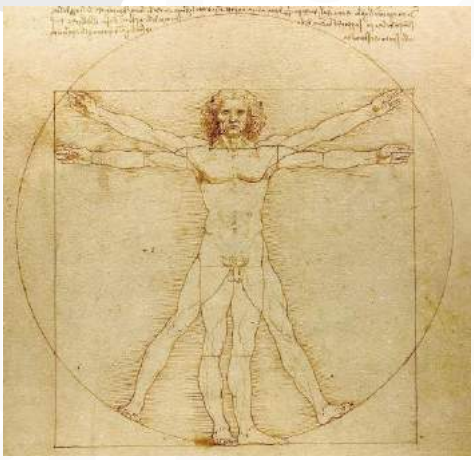


Figura 1 - Corpo Humano e sua simetria.

Estamos a todo momento pensando, sentindo e fazendo. Para pensar temos o nosso maior diferencial dentre todas as espécies de animais encontrados na terra, nosso poderoso cérebro.

Cérebro

O cérebro é responsável por receber toda a informação colhida por nossos sensores, tomando decisões lógicas conforme a necessidade. Por exemplo, se seu olho vê que está chovendo, você irá processar a informação e tomará a decisão de tirar as roupas no varal. Se você estiver lendo esse texto e alguém chamar pelo seu nome, você irá processar a informação e provavelmente tomará a decisão de verificar o que é.

Portanto, nosso cérebro é responsável pela tomada de decisões de como iremos interagir com o ambiente em que estamos.

Os sensores

Para interagir com o mundo, temos que coletar informações dele. Para isso nosso corpo conta com os cinco sentidos: Audição, Visão, Tato, Olfato e Paladar. Nossos sentidos são conjuntos de sensores que coletam informações do meio ambiente e transformam em informação que será processada por nosso cérebro.

Os atuadores

Os músculos são nossos dispositivos de atuação, eles são responsáveis por nossos movimentos. Desde um piscar de olhos até os mais inusitados movimentos que o homem pode fazer. Todos são resultados da contração e relaxamento dos nossos aproximados 660 músculos.

Desse modo, o cérebro tem o poder de interagir mecanicamente com o meio a partir de suas decisões.

A manutenção

Toda essa máquina biológica precisa de suprimentos tais como água e energia. Para isso possuímos sistemas que são responsáveis pela distribuição, em todo o corpo, dos suprimentos necessário para sua operação.

Vale ressaltar que podemos observar os mesmos sistemas em todos seres vivos que detenham um sistema nervoso. As formigas por exemplo, apesar de seu sistema nervoso primitivo, desempenham de forma extraordinária seu objetivo, a sobrevivência.

1.1.2. Robôs

Vamos chamar de robô qualquer sistema que seja dotado de alguma inteligência e que possa interagir com o mundo em que vivemos. Se queremos um sistema inteligente, já sabemos por onde começar. Precisaremos de:

- Um cérebro, onde será processada as informações e tomada as decisões;
- Sensores, que coletarão informações do meio e enviarão para o cérebro;
- Atuadores, que serão o meio de o cérebro executar tarefas;
- Sistemas auxiliares, que garantirão o funcionamento do robô.

Figura 2 - Robô jogando futebol.

No robô, teremos como cérebro um processador, no caso dessa apostila, um Arduino. Ele será responsável por processar as informações vindas dos sensores, que podem ser dos mais variados tipos, como sensores de temperatura, sensores de distância, entre outros. Depois de processada a informação, o cérebro do robô deverá tomar decisões lógicas e caso necessário exercer uma ação que permita alcançar um objetivo.

Figura 3 - Sistema autônomo de irrigação

Pense em um sistema inteligente que tem como objetivo molhar as plantas de um jardim. Para que ele seja autônomo e consiga desempenhar a sua atividade sozinho, ele tem que saber qual é o momento certo de molhar as plantas desse jardim, correto? Para isso, precisamos de um sensor que identifique se devemos ou não irrigar. Vamos supor que nosso sistema usa um sensor de umidade de solo.

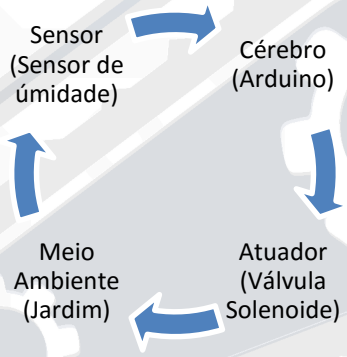
Agora que temos um sensor, precisamos de um cérebro que receba essa informação e seja capaz de tomar uma decisão de acordo com o objetivo do sistema. Vamos adotar como nosso cérebro um Arduino. Assim, o Arduino recebe as informações

do sensor e consegue saber se o solo está úmido ou seco e, portanto, consegue decidir se é necessário irrigar ou não o jardim.

O que adianta sabermos que o jardim precisa ser molhado se não fizermos nada? É nesse momento que precisamos de um atuador. Nesse exemplo podemos usar como atuador uma válvula solenoide, que irá liberar ou não a água para a irrigação.

Quando o Arduino perceber que o solo está seco demais, através do sensor, ele irá irrigar o jardim liberando a água através da válvula solenoide. Quando for identificado que o solo está úmido o suficiente, o Arduino, através da válvula solenoide, irá cortar o fluxo de água de irrigação.

Veja como o sistema inteligente interage com o meio ambiente formando um ciclo de informação. Primeiro ele sente o que está acontecendo no meio ambiente, depois o cérebro toma uma decisão baseada no seu objetivo e interage com o meio ambiente por meio do atuador



1.2. Hora de usar nossa força

Na apostila Arduino Básico Vol. 2, aprendemos a importância dos sensores e como usá-los, processando a informação coletada e tomando decisões.

No entanto, para desenvolvermos um sistema autônomo precisamos saber como atuar, como garantir ao nosso robô a capacidade de modificar as variáveis do meio para que alcance o estado desejado. Por exemplo: se a geladeira está gelada demais, o "cérebro" dela deve ser capaz de desligar o sistema responsável por gelar o seu interior.

O assunto dessa apostila é justamente esse, atuadores. Vamos aprender como exercer uma ação com o meio e como interagir com o ambiente, usando as informações coletadas pelos sensores e modificando as variáveis do meio através dos atuadores.

1.3. Atuadores

Atuadores são o meio pelo qual nossos robôs interagem com o mundo. Eles podem usar motores para se movimentar, LED's para indicar uma situação de perigo e resistores

para aquecer um liquido, entre muitas outras possibilidades de interação.

Em geral, nossos atuadores irão transformar energia elétrica em energia mecânica, luminosa ou térmica. Temos como exemplos de atuadores:

- Energia mecânica: Motores;
- Energia luminosa: LED's, LCD's e Lâmpadas;
- Energia térmica: Resistores.

Como estamos tratando de transforma energia elétrica em outras energias, temos que tomar um especial cuidado com consumo de energia de nosso sistema para que não sobrecarreguemos nossos dispositivos.

É importante saber que cada dispositivo tem uma capacidade máxima de fornecimento de energia. Por exemplo, uma porta digital do Arduino pode fornecer até 40mA de corrente, não podemos drenar mais que isso, caso contrário, podemos queimá-lo.

1.4. O foco dessa apostila

Sabendo da importância dos atuadores para os sistemas autônomos, nessa apostila iremos aprender como usá-los de forma segura com o Arduino.

Os atuadores abordados aqui são:

- Transistores
- Relés
- Servomotores
- Ponte H e motores DC

1.5. Ajude-nos a criar um material de primeira

Antes de você começar a ler essa apostila, te convido para nos ajudar a criar materiais 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.br com um tutorial de sua autoria que ficaremos felizes em abrir um espaço para você em nosso blog.

2. Módulo Relé

Os microcontroladores, tais como Atmega, PIC e MSP, são dispositivos lógicos. Eles são usados com o intuito de ser a inteligência do circuito, o cérebro de nosso sistema inteligente.

Por isso, um microcontrolador usa tensões e correntes baixas para funcionar e não são produzidos para suportar grandes tensões e correntes. O Arduino UNO, por exemplo, que usa o Atmega328, suporta um máximo de 40mA em suas portas I/O e fornece uma tensão de 5V. Além disso, é sempre aconselhável separar o sistema de inteligência do sistema de acionamento, para que curtos ou problemas no sistema de acionamento não danifique sua placa Arduino.

Quando queremos acionar cargas com valores de corrente ou tensão maiores precisamos utilizar algum dispositivo que suporte tal carga. Por exemplo: Queremos acender uma lâmpada de 127V quando estiver escuro. Sabemos que o Arduino não é capaz de fornecer tal tensão, para isso iremos precisar de algum dispositivo capaz de ligar e desligar a lâmpada através do comando do nosso Arduino, uma boa pedida é o relé.

Interruptor

Diariamente ligamos e desligamos lâmpadas através de um dispositivo muito eficaz, o interruptor. Tal como o nome já diz, ele tem a capacidade de interromper o fluxo de energia e vice-versa. Veja na figura abaixo como um interruptor trabalha. O fio vermelho, que é o positivo e traz a energia, passa pelo interruptor e depois vai para a lâmpada que já possui o negativo conectado. Quando o interruptor está aberto, não temos passagem de energia, quando mudamos o estado dele e fechamos o contato, a lâmpada liga.

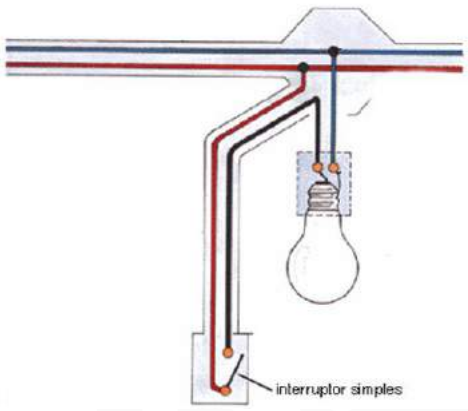


Figura 4 - Interruptor residencial.

2.1. Relé

Um relé é semelhante a um interruptor, com uma diferença, não precisamos usar nosso dedo para fechar ou abrir o contato. Para essa função, usamos um sistema engenhoso que quando alimentado, fecha o contato.

Esse sistema engenhoso possui um eletroímã, ou seja. Uma bobina que quando energizada criará um campo eletromagnético, se tornando um imã. Devido a força de atração magnética do eletroímã teremos a movimentação do interruptor. Veja a imagem a baixo para que possa entender melhor:

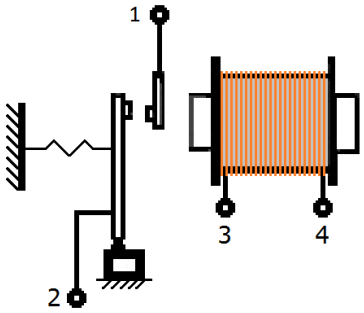


Figura 5 - Olhando um relé por dentro.

Quando alimentamos os terminais 3 e 4 da bobina, ela gera um campo eletromagnético que atrai a haste conectada ao terminal 2, que vence a força exercida pela mola conectada a ela. Isso faz com que essa haste se mova até encostar na haste conectada ao terminal 1. Com isso, teremos o contato fechado.

Quando paramos de alimentar a bobina através dos terminais 3 e 4, a força magnética exercida sobre a haste conectada ao terminal 2 deixa de existir e a mola puxa a haste de volta ao seu lugar inicial, abrindo o contato. Portanto, agora temos um interruptor acionado por uma bobina.

Usando um relé, podemos controlar um dispositivo, tal como a lâmpada de 127V que possui uma corrente e tensão alta, através da alimentação de uma bobina. A corrente e a tensão que essa bobina precisa varia com a especificação do relé.

2.2. O que é um módulo relé?

Apesar de a bobina relé demandar uma corrente e uma tensão baixa, ainda assim o relé precisa de mais energia do que uma porta digital do seu Arduino Uno pode fornecer.

Por conta disso, precisamos de um circuito auxiliar para o acionamento do relé. Esse circuito, que usa uma fonte de alimentação que pode ser a própria saída de 5V do seu Arduino, não é trivial, mas já pode ser encontrado pronto nos módulos relés muito

usados com Arduino. Esse tipo de módulo poupa tempo e é muito simples de usar.

Figura 6 - Módulo relé.

No próximo capítulo dessa apostila abordaremos sobre transistores e aprenderemos como montar o circuito para acionar um relé. Porém, nesse capítulo nos limitaremos a usar o módulo devido a sua praticidade na montagem do circuito.

2.3. Mãos à obra – Experiência 1

Então vamos fazer uma experiência e olhar o funcionamento de um módulo relé na prática?

2.3.1. Ingredientes

- Arduino
- Módulo Relé
- Fios Jumper's
- Extensão
- Botão
- Resistor de 10kOhm

2.3.2. Misturando os ingredientes

Cuidado!

Nesse tutorial iremos mexer com tensões e correntes maiores. Tome cuidado ao manusear seu projeto, risco de choque!!

Tome cuidado para não energizar seu Arduino com a tensão da tomada. Isso irá queimá-lo.

Vamos separar essa montagem em duas partes:

1. *Inteligência*

Essa parte do projeto representa o Arduino e seus periféricos de baixa potência. Para montar seu circuito, use a figura a seguir. Garanta que seu Arduino esteja desligado

durante a montagem.

O pino de entrada do botão é o 8 e o pino de saída do relé é o 13.

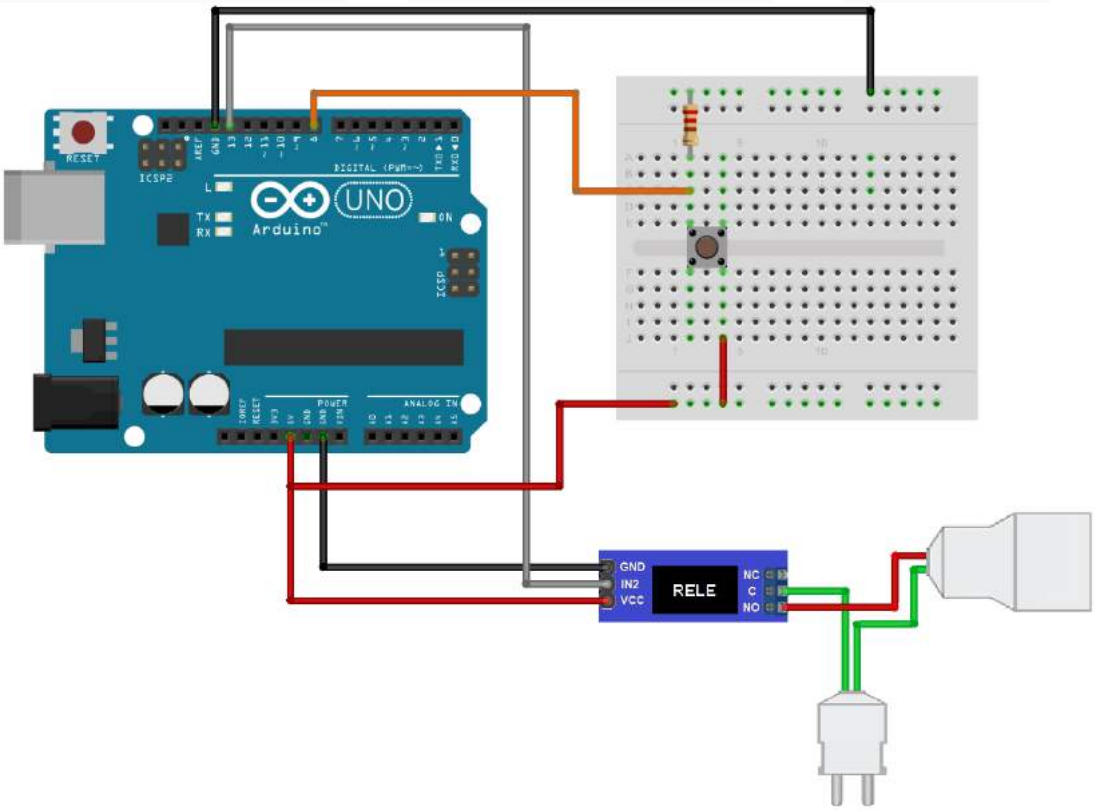


Figura 7- Esquemático módulo relé

2. *Potência*

Nessa parte iremos montar a parte de maior potência do nosso circuito. Para esse projeto iremos precisar de uma extensão que você não usará mais, ou você pode fazer a sua.

Quando lidamos com energia elétrica temos que tomar muito cuidado! Então monte seu projeto com segurança. Não queremos transformar nossa diversão em dor de cabeça, não é mesmo?

Por isso, vamos montar uma extensão segura. Caso você já tenha uma que possa cortar os seus fios, você poderá usá-la. Basta executar apenas os três últimos passos.

Para nossa extensão você vai precisar de:

- Um adaptador de tomada macho
- Um adaptador de tomada fêmea
- 4 metros de Fio de cobre (Fique à vontade em relação ao tamanho).

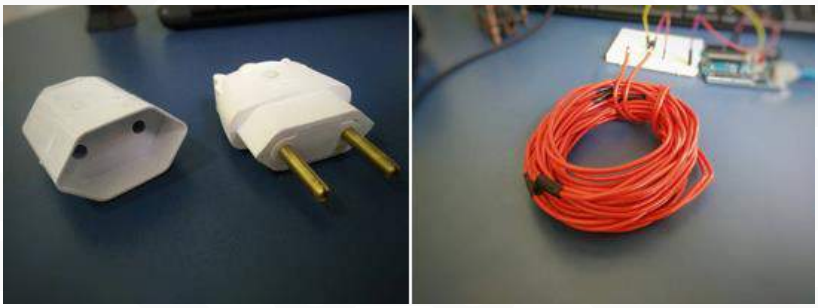


Figura 8 - Adaptadores e fio para fazer a extensão.

Vamos montar nossa extensão:

- Divida seu fio de cobre em dois fios de 2 metros (ou metade do tamanho que você tiver). Dessa forma, você terá dois fios iguais;
- Com seu alicate, descasque a ponta de cada fio;

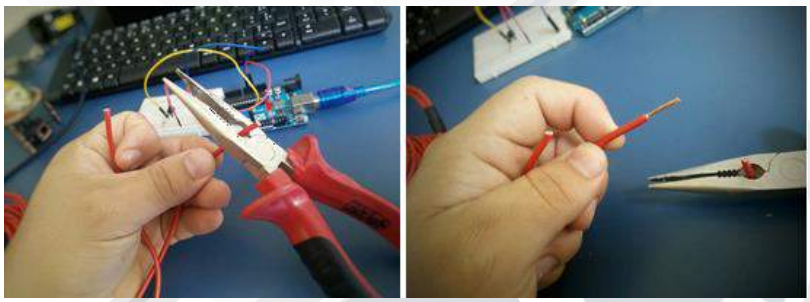


Figura 9 - Descasque a ponta do fio.

- Parafuse a ponta dos dois no adaptador macho;

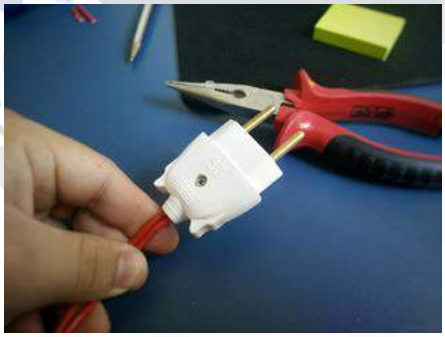


Figura 10 - Parafuse os fios dentro dos adaptadores.

- Parafuse a outra ponta dos dois no adaptador fêmea;

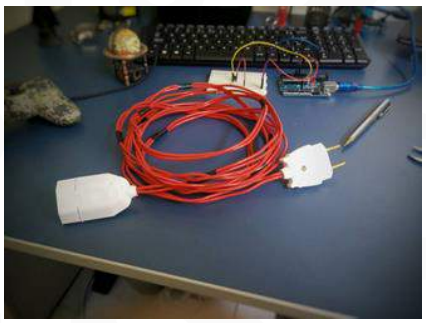


Figura 11 - Extensão pronta.

- Agora vamos conectá-la ao nosso circuito.
- Corte um dos dois fios.

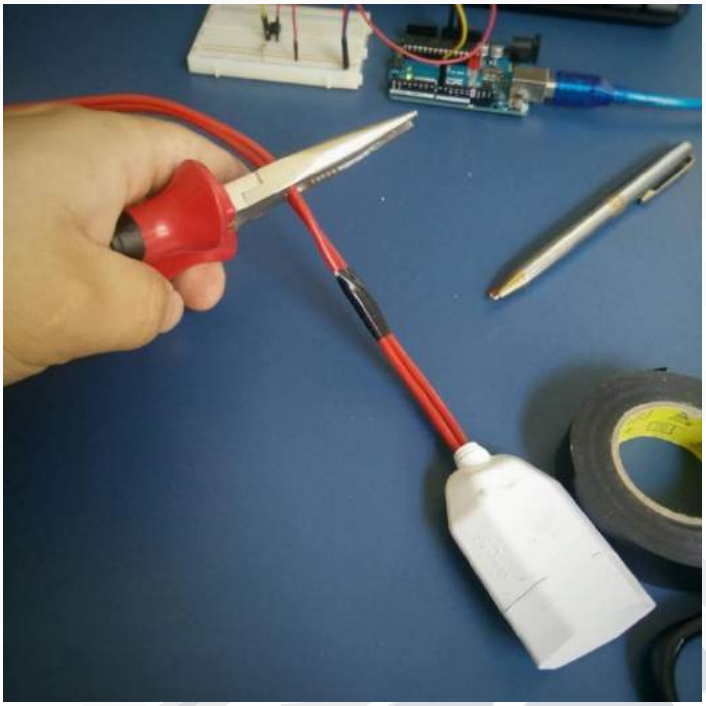


Figura 12 - Corte um dos fios.

- Descasque as das pontas
- Parafuse uma no comum (COM) e outra no contato normalmente aberta (NA ou NO – Caso esteja em inglês)

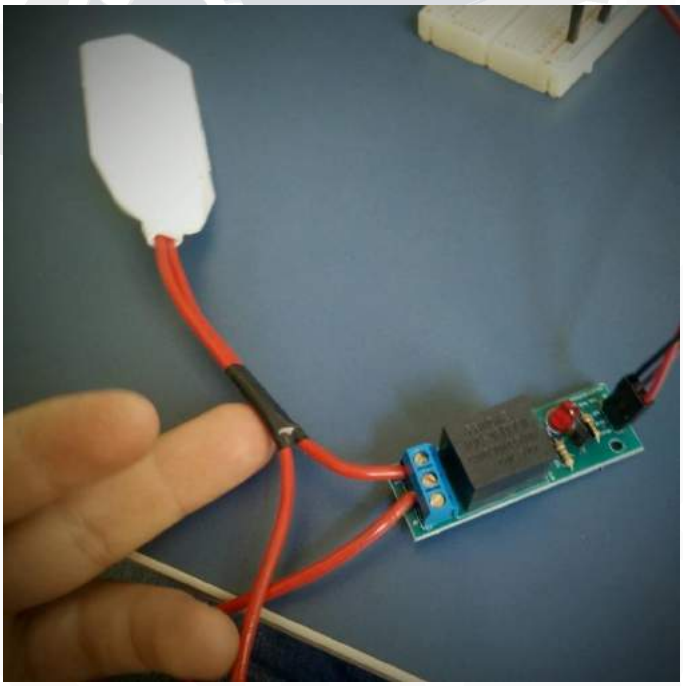


Figura 13 - Parafuse os fios no módulo relé.

Agora, volte, e confira se você realmente montou tudo como explicado.

Seu circuito ficará semelhante ao da foto a seguir:

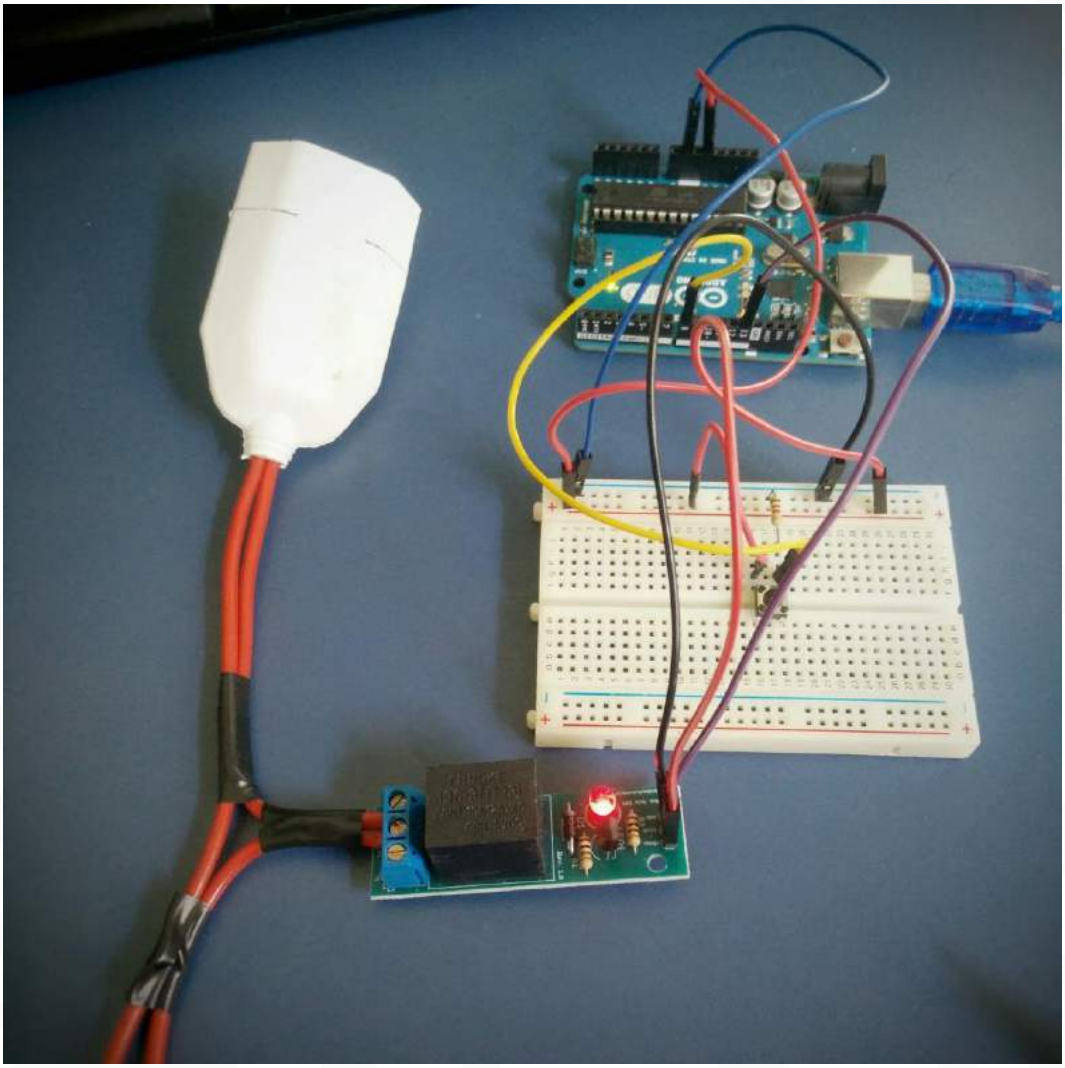


Figura 14 - Sistema montado.

2.3.3. 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 que a placa configurada é a que você está usando (board).

2.3.4. Preparando a cobertura

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

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

```
bool stable;    // Guarda o último estado estável do botão;
bool unstable;  // Guarda o último estado instável do botão;
uint32_t bounce_timer;
uint8_t counter = 0;
int rele=13;
int botao=8;
bool set;

bool changed() {
  bool now = digitalRead(botao);   // Lê o estado atual do botão;
  if (unstable != now) {       // Checa se houve mudança;
bounce_timer = millis();   // Atualiza timer;
unstable = now;            // Atualiza estado instável;
  }
  else if (millis() - bounce_timer > 10) {  // Checa o tempo de trepidação acabou;
    if (stable != now) {           // Checa se a mudança ainda persiste;
      stable = now;                  // Atualiza estado estável;
      return 1;
    }
  }
  return 0;
}

void setup() {
  Serial.begin(9600); // configura comunicação serial a uma taxa de 9600 bauds.
  pinMode(botao, INPUT_PULLUP);  // Configura pino 8 como entrada e habilita pull up interno;
  pinMode(rele, OUTPUT);  // Configura pino 8 como entrada e habilita pull up interno;
stable = digitalRead(botao);
  set=0;
}

void loop() {
if (changed()) { //verifica se houve mudança de estado
     if (digitalRead(botao)) set=!set; //se mudou o estado e o botão está pressionado muda o
valor de set
  }
 digitalWrite(rele,set);
  // Outras tarefas;
}
```

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

2.3.5. Experimentando o prato

Conecte algum aparelho em sua extensão (que tenha uma coerente menor que 10A), um ventilador por exemplo. Então ligue seu Arduino e em seguida ligue a tomada de sua extensão.

Cuidado!

Só ligue seu sistema na tomada quando tiver a certeza que está tudo corretamente conectado.

Tome cuidado para não tomar um choque ou energizar seu Arduino com a tensão da tomada. Isso irá te machucar ou queimar seu Arduino.

Se tudo deu certo, quando você apertar o botão uma vez, o dispositivo que estiver ligado na tomada deverá ligar.

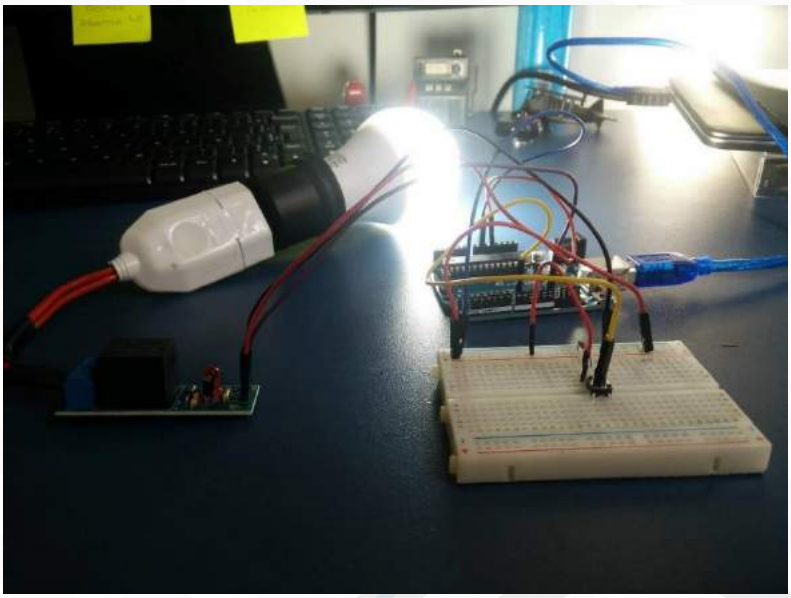


Figura 15 - Sistema depois de apertado o botão 1 vez.

Quando apertar uma segunda vez, o dispositivo deverá desligar.

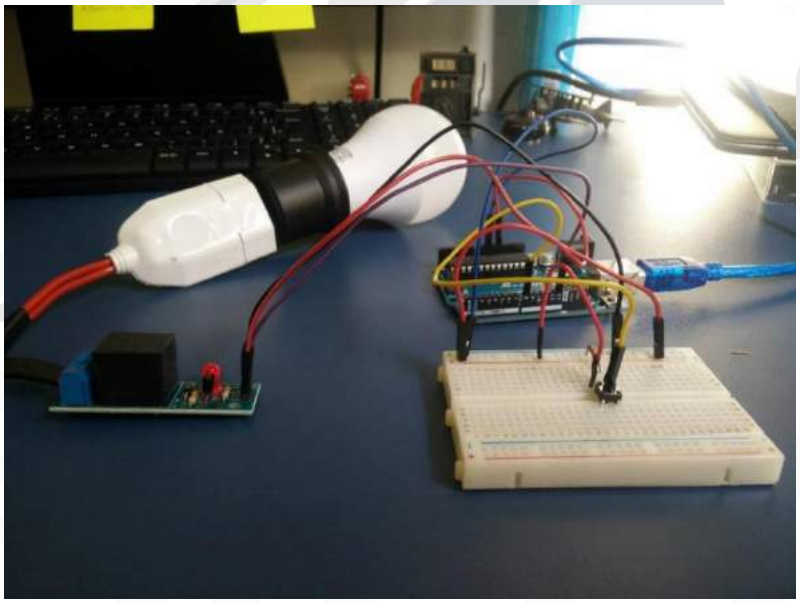


Figura 16 - Sistema depois de apertado o botão pela segunda vez.

2.4. Entendendo o Hardware

Nessa experiência usamos um botão para ligar e desligar uma carga através do relé.

Quando pressionado o botão pela primeira vez, o Arduino detecta essa entrada e acionar o rele. Quando pressionado novamente, o Arduino detecta e desliga a carga e assim por diante.

O funcionamento do relé foi explicado no começo do capítulo. Em resumo, o relé

funciona como um interruptor comandado pelo Arduino. Quando energizado ele aciona a carga, quando desenergizado ele desliga a carga. Sendo a carga em nosso caso uma lâmpada.

Um ponto importante para ser ressaltado aqui é que esse é o tipo de projeto que se é fundamental o tratamento do Bounce.

- *O Bounce*

Ao pressionar um botão nós fazemos com que os seus contatos se choquem o que fará com que o botão trepide um pouco antes de se estabilizar. Esse fenômeno é similar ao que acontece com uma bolinha de ping pong quando jogada contra o chão, que quica algumas vezes antes de se estabilizar. A figura a seguir mostra a forma de onda do sinal produzido pelo botão ao ser pressionado.

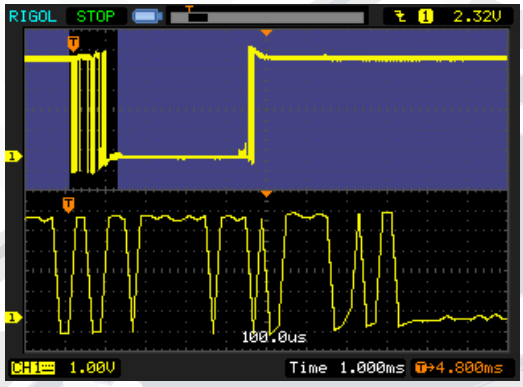


Figura 17 - Olhando o Bounce em um osciloscópio.

Podemos observar que o botão varia entre nível alto e baixo várias vezes antes de se estabilizar em 0V, ou seja, quando ligamos um LED por exemplo, ele pisca várias vezes cada vez que o botão é pressionado, porém isso acontece tão rápido que nosso olho não consegue perceber.

Como nosso programa liga quando pressionado uma vez o botão e desliga quando pressionado novamente, teremos um grande problema aqui, porque nossa carga ligaria e desligaria muitas vezes em um curto espaço de tempo e o estado final seria aleatório.

A solução para isso é ignorar essa trepidação através do nosso programa.

Vamos nos ater aqui em tratar o Bounce através de uma simples função que será explicada no próximo item.

Você pode saber mais sobre esse efeito na postagem em nosso blog sobre botões nesse link: Leitura de Botões e o Bounce. Lá você encontrará mais informações sobre o assunto.

2.5. Entendendo o programa

Nesse programa usamos duas coisas muito interessantes que podem ser usadas sempre que você tiver problemas semelhantes. Um botão momentâneo para ligar e desligar a mesma carga e o tratamento de Bounce desse botão.

O bacana que também podemos adaptar um sensor no lugar do botão para criar alguma lógica afim de controlar o nosso módulo relé.

- *Tratando o Bounce*

A primeira coisa que precisamos falar é sobre o tratamento de Bounce que ocorre devido a trepidação inicial dos contatos do botão. Esse efeito foi explicado no item anterior. Aqui vamos aprender como fizemos para tratá-lo.

A ideia é simples! Precisamos saber se a trepidação acabou. Para isso medimos quanto tempo temos desde a última alteração de estado. Se ela for maior que um determinado tempo, consideramos que nosso estado está estável.

Veja essa função que nos retorna 1, caso o botão tenha mudado de estado, e 0, caso não tenha mudado.

```
bool changed() {
  bool now = digitalRead(botao);   // Lê o estado atual do botão;
  if (unstable != now) {       // Checa se houve mudança;
bounce_timer = millis();   // Atualiza timer;
unstable = now;            // Atualiza estado instável;
  }
  else if (millis() - bounce_timer > 10) {  // Checa o tempo de trepidação acabou;
       if (stable != now) {           // Checa se a mudança ainda persiste;
              stable = now;                  // Atualiza estado estável;
return 1;
    }
  }
  return 0;
}
```

Primeiro lemos o valor de entrada

```
  bool now = digitalRead(botao);   // Lê o estado atual do botão;
```

Verificamos se ouve mudança, ou seja, se o valor lido é diferente do último.

```
  if (unstable != now) {       // Checa se houve mudança;
```

Armazena o tempo de millis atual.

```
bounce_timer = millis(); // Atualiza timer;
```

Muda o estado armazenado na variável usntable.

```
unstable = now;            // Atualiza estado instável;
```

Caso não tenha mudado de estado, verifique se já se passou mais de 10 milissegundos desde a última alteração de entrada.

```
else if (millis() - bounce_timer > 10) {  // Checa o tempo de trepidação acabou;
```

Se já se passaram 10 milissegundos ele verificará se o estado atual estável é diferente do último estado estável.

```
if (stable != now) {  // Checa se a mudança ainda persiste;
```

Atualiza o estado estável.

```
stable = now;                  // Atualiza estado estável;
```

Retorna 1, simbolizando que ouve mudança de estado.

```
return 1;
```

Caso não tenha dado os 10 milissegundos, retorna 0, simbolizando que ouve mudança de estado.

```
return 0;
```

Essa função é chamada um "se" dentro da função loop como condição lógica de um if. Caso a função retorne 1 (ouve mudança de estado), nós iremos verificar se essa mudança foi para pressionado, caso positivo ele muda o valor da variável set, que pode assumir o valor 0 ou 1 por ser do tipo bool.

```
void loop() {
if (changed()) { //verifica se houve mudança de estado
     if (digitalRead(botao)) set=!set; //se mudou o estado e o botão está pressionado
muda o valor de set
  }
 digitalWrite(rele,set);
  // Outras tarefas;
}
```

Dessa forma, se é a primeira vez que pressionamos o botão, teremos set indo de 0 para 1. Na segunda vez teremos set indo de 1 para 0. Assim por diante.

Depois pegamos o valor de set e escrevemos no pino de saída que está conectado o relé (pino 13).

2.6. Desafio

Agora que sabemos usar um módulo relé. Que tal usar um sensor de presença para acender uma lâmpada ou um sensor de temperatura para ligar seu ventilador? Seria uma boa ideia para automatizar o seu quarto, o que acha?

Faça isso:

- Ligue uma lâmpada quando detectar movimento no ambiente;
- Ligue um ventilador caso a temperatura esteja muito alta.

3. Usando Transistores no Arduino

No capítulo anterior ressaltei a importância de um circuito auxiliar para o relé, nesse capítulo aprenderemos como montá-lo. Isso mesmo, você poderá montar seu próprio módulo relé. Para isso, iremos aprender a usar um componente muito importante. O Transistor.

3.1. Por que usar transistores?

Já falamos que os microcontroladores, tais como Atmega, PIC e MSP, são dispositivos lógicos que são usados com o intuito de ser a inteligência do circuito. Dessa forma, esses componentes não são produzidos para suportar grandes correntes. Para muitas aplicações isso não é o suficiente. Segue alguns exemplos:

- Motores DC;
- Fitas de LED;
- Relé;
- Ou qualquer componente que precise de mais de 5V ou 40mA.

Quando queremos acionar cargas com valores de corrente ou tensão maiores precisamos de utilizar algum dispositivo que suporte tal carga. Uma das soluções mais práticas são os Transistores.

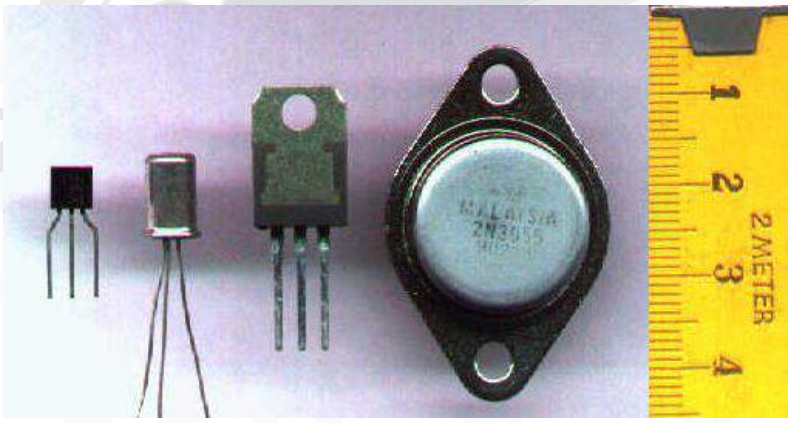


Figura 18 - Alguns encapsulamentos de transistores.

Nesse capítulo iremos focar nos transistores BJT (junção bipolar), tentarei ser o mais prático possível, pensando em alguém que não saiba nada de eletrônica. Caso queira saber mais sobre transistores, existe bastante material na internet explicando os vários tipos e suas propriedades de forma mais detalhada.

3.2. Um pouco de história!

Uma das invenções mais importantes do Milênio, os transistores possibilitaram uma revolução tecnológica inimaginável. Agora mesmo, usando o seu computador ou qualquer aparelho eletrônico você está colhendo os frutos dessa invenção.

Antes dos transistores, os computadores funcionavam a partir do uso de válvulas termiônicas, elas até que funcionavam bem, porém ocupavam muito espaço fazendo com que computadores ocupassem grandes áreas. Dessa forma, surgiu a necessidade de descobrir uma forma de utilizar um substituto menor.

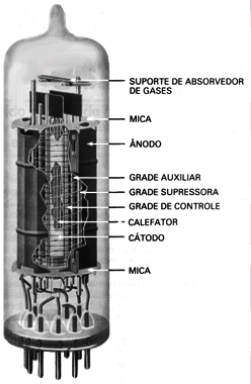


Figura 19 - Válvula.

Em 1947, nos Laboratórios da Bell Telephone, os pesquisadores John Bardeen e Walter Houser Brattain inventaram o primeiro transistor feito de germânio. Em 23 de Dezembro de 1948, foi demonstrado para o mundo por John Bardeen, Walter Houser Brattain e William Bradford Shockley, que ganharam o Nobel de Física em 1956.

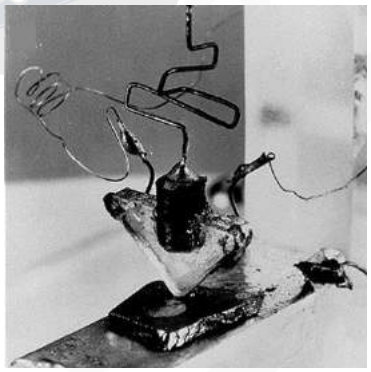


Figura 20 - Primeiro transistor.

Hoje, o material semicondutor mais usado na fabricação de transistores é o silício. O silício é preferível porque possibilita o funcionamento a temperaturas mais elevadas (175 ºC, quando comparado com os ~75ºC dos transistores de germânio) e também porque apresenta correntes de fuga menores. Com a evolução tecnológica, surge a necessidade de diminuir cada vez mais o tamanho dos transistores para que se diminua o

tamanho dos equipamentos e que se aumente a capacidade de processamento. Uma das novas tecnologias é o chamado transistor 3D que tem dimensões nanométricas.

3.3. Como funcionam?

Imagine uma válvula hidráulica, a do seu chuveiro por exemplo, ela tem a função de controlar o fluxo de água que sairá pelo chuveiro, correto? Podemos ter a válvula totalmente fechada, totalmente aberta ou em uma abertura especifica, limitando a corrente de água.

Figura 21 - Válvula de chuveiro.

Nessa válvula temos:

- Uma entrada, onde entra a corrente de água;
- Uma saída, nesse caso vai para o chuveiro
- Um elemento de controle de fluxo, no caso o volante da válvula.

O transistor é muito semelhante, porém tratamos de uma corrente de elétrons. O transistor atua como uma válvula. No transistor NPN, temos:

- Uma entrada, chamada coletor, por onde entra a corrente de elétrons;
- Uma saída, chamada emissor, por onde sai a corrente de elétrons;
- E uma entrada de controle, chamada base, que no caso é controlada a partir de uma corrente de controle.

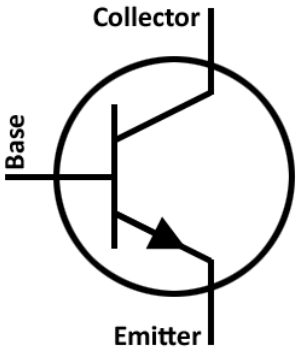


Figura 22 - Símbolo de um transistor PNP.

Como assim, corrente de controle?

Quando trabalhando no estado **Ativo**, o transistor aumenta ou reduz o fluxo de corrente entre o coletor e o emissor conforme à corrente de base aumenta ou reduz. Dessa forma, existe uma relação entre corrente do emissor e coletor com a corrente de base. Ela é:

$$Icoletor \cong Iemissor \cong Ibase.hfe$$

Repare que a corrente de emissor e coletor são aproximadamente iguais (em breve explicaremos por que) e que as duas são proporcionais a corrente de base.

Mas quem é esse hfe?

O hfe, também conhecido como β **(**beta), é o coeficiente de ganho de corrente. Ele relaciona a corrente de base com a corrente de coletor e emissor. Todo transistor tem o seu valor, esse pode ser consultado na folha de dados do componente (datasheet). Esse valor costuma ser na ordem de centenas. Assim, a corrente de base é muito pequena, em relação às correntes de coletor e de base.

Contudo, a corrente de base tem que fluir para algum lugar. No caso do transistor NPN, ela se junta a corrente de coletor, fluindo em direção ao emissor. Logo, no transistor NPN:

$$Iemissor = Icoletor + Ibase = Ibase.hfe + Ibase$$

$$Iemissor = Ibase.(hfe + 1) \cong Ibase.hfe$$

$$Icoletor \cong Iemissor$$

Como hfe >> 1 (muito maior que 1), podemos aproximar o termo (hfe+1) para hfe e assim considerar que as correntes de emissor e coletor são aproximadamente iguais, cometendo um erro inferior a 3% em transistores típicos.

Mas o que acontece se a corrente de base for muito grande?

O transistor entrará em um estado de **Saturação**. Chega um instante em que a corrente de base é tão grande que a corrente de coletor não consegue ser proporcional a ela. Nesse instante, o transistor libera o máximo de corrente de coletor que pode. É como um registro totalmente aberto.

$$Icoletor < Ibase \,.\, hfe$$

(Condição de Saturação)

E como restringir toda a corrente?

Quando nenhuma corrente flui entre coletor (c) e emissor (e), pode-se dizer que o

transistor está em corte. Para que possamos entrar nesse estado, algumas condições são necessárias. No transistor NPN as condições são:

- Vb < Vc, Tensão de coletor maior que tensão de base;
- Vb - Ve < 0.7, Tensão de base deve ser 0,7V maior que tensão de Emissor.

Observe que caso o emissor tenha uma tensão de 0V e colocarmos uma tensão de 0V na base, é o suficiente para que o transistor não conduza. Já que a tensão de base não será 0,7V maior que a tensão de emissor.

Será que entendi?

O transistor possui 3 estados de operação, são eles:

- Ativo, *Icoletor ≅ Iemissor ≅ Ibase.hfe*;
- Corte, *Icoletor ≅ Iemissor ≅ 0*;
- Saturado, *Icoletor < Ibase.hfe* (deixa toda a corrente fluir).

Para o transistor NPN funcionar em estado ativo é necessário que:

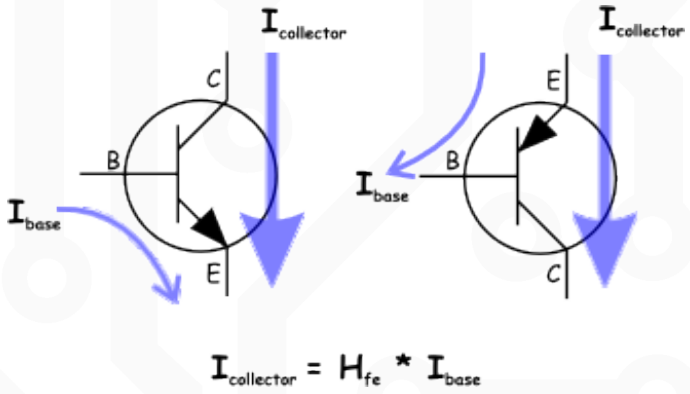
- Vb < Vc, Tensão de coletor maior que tensão de base;
- Vb - Ve < 0.7, Tensão de base 0,7V maior que tensão de Emissor;

Transistor PNP

Além do transistor NPN, existe também o PNP. Os dois possuem configurações internas diferentes que mudam um pouco o funcionamento deles.

- No transistor NPN a corrente flui do coletor para o emissor, e a corrente de base entra somando junto a corrente de emissor, fluindo da base para o emissor.
- No transistor PNP a corrente flui do emissor para o coletor e a corrente de base flui do emissor para a base.

Veja a imagem a seguir:



Além disso, os parâmetros para que não entre em Corte mudam. Para o transistor

PNP funcionar, precisamos que:

- Tensão de Base seja maior que tensão de Coletor;
- Tensão de Emissor 0,7V maior que tensão de Base;

Observe que a corrente de base contínua sendo muito pequena, considerando Icoletor ≅ Iemissor.

3.4. Qual transistor usar?

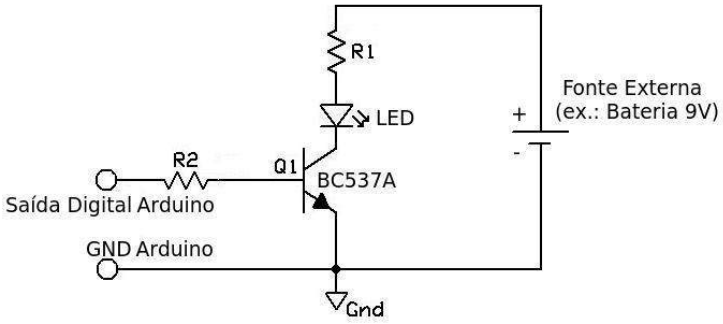
Para selecionar um transistor, é importante verificar os níveis de corrente, tensão e dissipação de potência em que ele irá trabalhar. Isso é feito na etapa de projeto, utilizando teoria de circuitos ou (quando possível) simulando o comportamento do circuito em softwares específicos. Então, deve--se escolher um dispositivo com capacidade de suportar tais correntes, tensões. Para isso existem duas abordagens principais. A mais comum e simples, é utilizar dispositivos de projetos parecidos (amplamente disponíveis na internet) pois já foram testados por outras pessoas. A abordagem ideal, porém, mais trabalhosa é consultar a folha de dados (datasheet) dos transistores. Nos datasheets é possível descobrir a SOA (Safety Operation Area) ou 'área de operação segura' dos transistores e assim é possível confiar no seu funcionamento correto e na sua durabilidade.

O que devemos verificar:

- Pol: polarização; negativa quer dizer NPN e positiva significa PNP.
- VCEO: tensão entre coletor e emissor com a base aberta.
- VCER: tensão entre coletor e emissor com resistor no emissor.
- IC: corrente máxima do coletor.
- PTOT: é a máxima potência que o transistor pode dissipar (Corrente Máxima de coletor vezes Tensão máxima entre coletor e base)
- hFE: ganho (beta).
- Encapsulamento: a maneira como o fabricante encapsulou o transistor, nos fornece a identificação dos terminais.

3.5. Como usar transistores no Arduino?

Veremos agora como usar um transistor NPN junto ao Arduino. Existem várias configurações de circuitos usando transistores, mas vamos nos ater a uma das mais simples. Ela é apresentada na figura a seguir. Nesse caso, estamos controlando um LED através do Arduino com o auxílio do transistor:



Saída Arduino

A saída do Arduino será:

- 5V, quando em nível lógico alto;
- 0V, quando em nível lógico baixo.

Para que o LED acenda, precisamos que quando a saída do Arduino estiver em nível lógico alto, o transistor conduza (Estado Ativo ou Saturado) e que quando estiver em nível lógico baixo, não conduza (Estado de Corte).

Relembrando

O transistor possui 3 estados de operação, são eles:

- Ativo, Icoletor ≅ Iemissor ≅ Ibase.hfe;
- Corte, Icoletor ≅ Iemissor ≅ 0;
- Saturado, Icoletor < Ibase.hfe (deixa toda a corrente fuir).

Para o transistor NPN conduza precisamos que:

- Vb < Vc, Tensão de coletor maior que tensão de base;
- Vb - Ve < 0.7, Tensão de base 0,7V maior que tensão de Emissor;

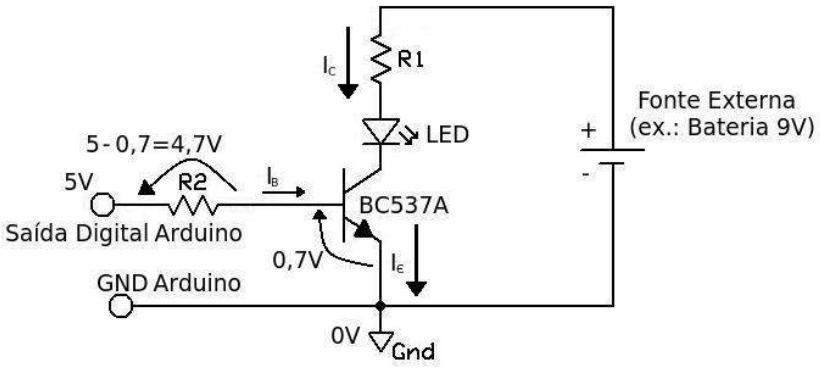
Transistor em Corte

Como o emissor está conectado ao GND, sua tensão é de 0V. Assim, quando a saída do Arduino estiver em 0V não atenderá um dos requisitos para que o transistor conduza (*Tensão de Emissor 0,7V maior que tensão de Base*). Dessa forma, o transistor entrará em estado de Corte.

Transistor Conduzindo

Quando o Arduino estiver com 5V em sua saída, sendo a tensão de coletor maior que a tensão de base, teremos o transistor na região ativa ou saturada. Para garantimos que quando em região ativa teremos a corrente de coletor suficiente para acionar o LED

precisamos dimensionar os resistores de base (R2) e de coletor (R1) corretamente.



Resistor R1

O resistor R1 tem função de limitar a corrente de coletor. Para acionar um LED, precisamos de uma corrente de pelo menos 10mA. Para uma tensão de 9V um resistor de 150 Ohm atende muito bem, veja os cálculos a seguir.

Aplicando a 1ª lei de Ohm considerando uma queda de tensão de 2,5V no LED e desconsiderando a queda de tensão no transistor, temos:

Icoletor-máx = (Vfonte – Queda de tensão no componente) /R

Icoletor-máx = (9-2,5)V/150 Ohm = 43mA

O que é suficiente para alimentar o LED.

Resistor R2 – O mais importante

O resistor R2 tem a função de limitar a corrente de base. Sendo ela responsável por controlar a corrente que irá fluir do coletor para o emissor, temos que escolhê-la com cuidado para garantir a mínima corrente no coletor que conseguirá acender o LED.

Aplicando a 1ª lei de Ohm temos que a corrente de base é:

Ibase = (5-0,7)/R2 = 4,3/R2-max

Mas precisamos que a corrente de coletor seja pelo menos de 10mA, para isso temos que:

Icoletor-min ≅ Iemissor-min ≅ 10mA = 0,01A = Ibase.hfe

0,01A = Ibase-min.hfe

Considerando que estamos usando o transistor BC337, temos que seu hfe = 100 (Consultado na Folha de Dados – datasheet), dessa forma:

0,01A = Ibase.110

$$Ibase\text{-}min=0,01/110 = 0,00009= 0,09mA$$

Logo:

$$Ibase\text{-}min= 0,00009 = 4,3/R2$$

$$R2\text{-}max = 4,3/0,00009 =52,2KOhm$$

Dessa forma, devemos usar um resistor menor que o calculado. Vamos adotar 4,7KOhm. Com esse resistor, teremos:

$$Ibase=4,3/4,7KOhm=0,9mA$$

$$Icoletor'=Ibase.hfe$$

$$Icoletor'=0,9mA.110=99mA$$

$$Como: Ibase.hfe > Icoletor \rightarrow 99mA > 43mA, então:$$

Transistor Saturado

3.6. Resumo dos cálculos

Requisito

Icoletor > 10mA -> Corrente de coletor deve ser maior que 10mA quando o transistor estiver conduzindo para que o LED acenda.

Componentes

- Fonte externa de 9V;
- Transistor BC337-> hfe=100;
- R1 -> resistor que limitará a corrente de coletor, definindo a corrente máxima;
- R2-> resistor que define a corrente de base que por sua vez define a máxima corrente de coletor.

Especificações

- R1=150 Ohm, Limita a corrente de coletor em 43mA
- R2 < 52,2 kOhm, adotamos R2=4,7 kOhm para corrente de base mínima de 0,09mA, logo a corrente de coletor mínima é 10mA

3.7. O que você precisa saber?

Ao projetar um circuito usando o transistor NPN, devemos:

- Escolher qual transistor usar a partir das tensões e correntes máximas;
- Dimensionar os resistores R1 e R2.

Para dimensionar R1, que é o resistor responsável por limitar a corrente de coletor, precisamos saber:

- Qual a tensão da fonte;
- Qual a queda de tensão no componente;
- Qual a corrente necessária para que o componente controlado funcione corretamente (Icoletor-min).

Fórmula para cálculo do R1:

Icoletor-min =(Vfonte – Queda de tensão no componente) /R1-max

R1 não deve ser muito menor que R1-max.

Para dimensionar R2, que é responsável por controlar a corrente de coletor através da corrente de base (Icoletor max = Ibase.hfe), precisamos saber:

- Qual a corrente necessária para que o componente controlado funcione corretamente (Icoletor-min);
- hfe do transistor escolhido.

Fórmula para cálculo do R2:

Ibase = 4,3/R2-max

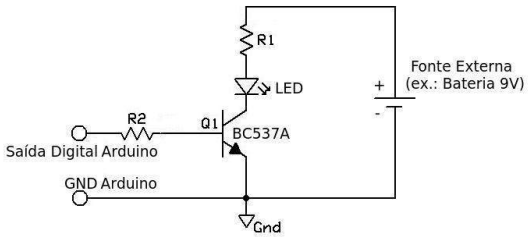
Ibase = Icoletor/hfe

Obs.: É importante dimensionar bem esse resistor para que não se restrinja demais a corrente de coletor do transistor. O ideal para esse tipo de aplicação é trabalharmos com o transistor saturado (Icoletor < Ibase.hfe). Para isso, geralmente escolhemos um resistor bem menor que o R2 calculado.

3.8. DICAS

Para esse tipo de aplicação, na maioria dos casos, podemos adotar:

- R1 = 150 Ohm;
- R2 = 1 kOhm a 4,7 kOhm
- Fonte externa até 12V para o BC337 (ou BC537A)



No lugar do LED podemos colocar um relé ou outro componente que consuma até 100mA, caso esteja usando o BC337

Costumo usar na maioria das vezes os seguintes transistores NPN:

- BC337 (Tensão de coletor máx =45V; Corrente de coletor máx = 500mA ; hfe=100)
- BD135 (Tensão de coletor máx =45V; Corrente de coletor máx =1,5A ; Corrente de Base máx = 500mA; hfe=25)
- TIP122 (Tensão de coletor máx =100V; Corrente de coletor máx = 5A ; Corrente de Base máx = 120mA; hfe= 1000)

3.9. Mãos à obra – Experiência 2

Então vamos fazer uma experiência e olhar o funcionamento de um módulo relé na prática?

3.9.1. Ingredientes

- Arduino
- Relé 5V GS-SH-205T
- Fios Jumper's
- Transistor BC337
- Botão
- Resistor 10kOhm a 15kOhm
- Resistor de 560Ohm ou próximo
- Resistor de 1kOhm´
- 2 Leds

3.9.2. Misturando os ingredientes

Nessa experiência iremos usar um transistor para acionar um relé. Mais especificamente um relé de 5V modelo GS-SH-205T. Como esse Rele exige pouca corrente, não iremos usar o resistor de emissor R1.

Iremos usar esse rele para chavear dois LEDs. Eles irão alternar conforme a bobina ligar e deligar. Esse acionamento será feito pelo botão, tal como a experiência anterior.

Figura 23 - Relé 5V

Esse relé possui 2 canais, ou seja, duas chaves internas. Como podemos ver no esquemático a seguir:

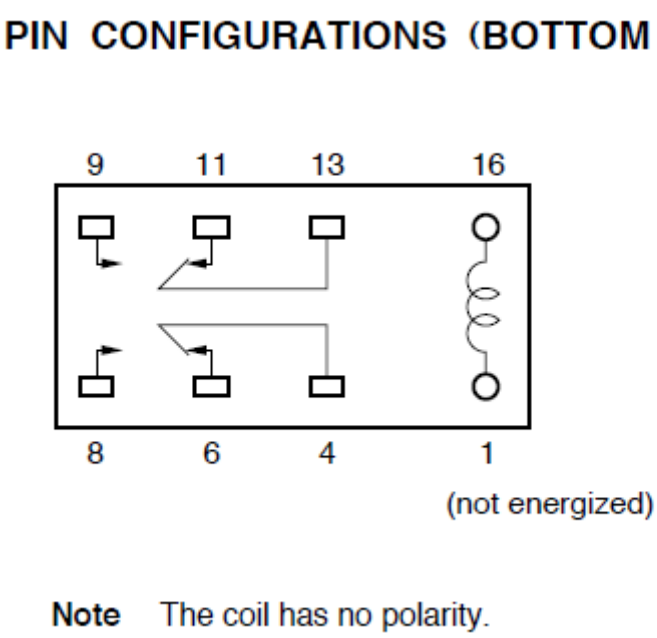


Figura 24 - Configuração dos pinos do relé 5V

Veja que temos apenas uma bobina e duas chaves. Quando a bobina (1 e 16) está sem energia, temos o pino 13(COM) conectado ao pino 11 (NF) e o pino 4 (COM) conectado ao pino 6 (NF).

Quando a bobina está energizada, temos o pino 13(COM) conectado ao pino 9 (NA) e o pino 4 (COM) conectado ao pino 8 (NA).

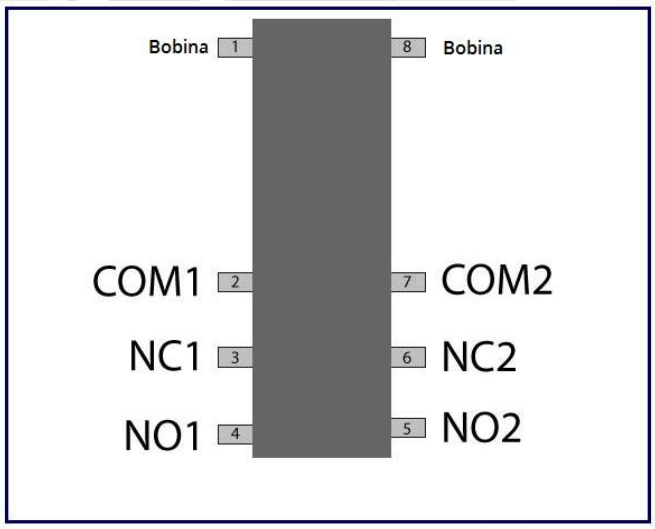


Figura 25 - Configuração dos pinos do relé 5V.

Para esse experimento iremos usar apenas um canal. No caso eu usei os pinos 4, 6 e 8. Onde coloquei no comum um resistor de 560ohm ligado ao 5V. No 6 eu coloquei um LED vermelho e no 8 eu coloquei um LED verde.

Dessa forma, quando a bobina estiver desligada. Teremos o led vermelho ligado. Quando ela estiver ligada, teremos o led verde ligado.

Vamos a montagem: Garanta que seu Arduino esteja desligado durante a montagem.

O pino de entrada do botão é o 8 e o pino de saída para a base do transistor é o 13.

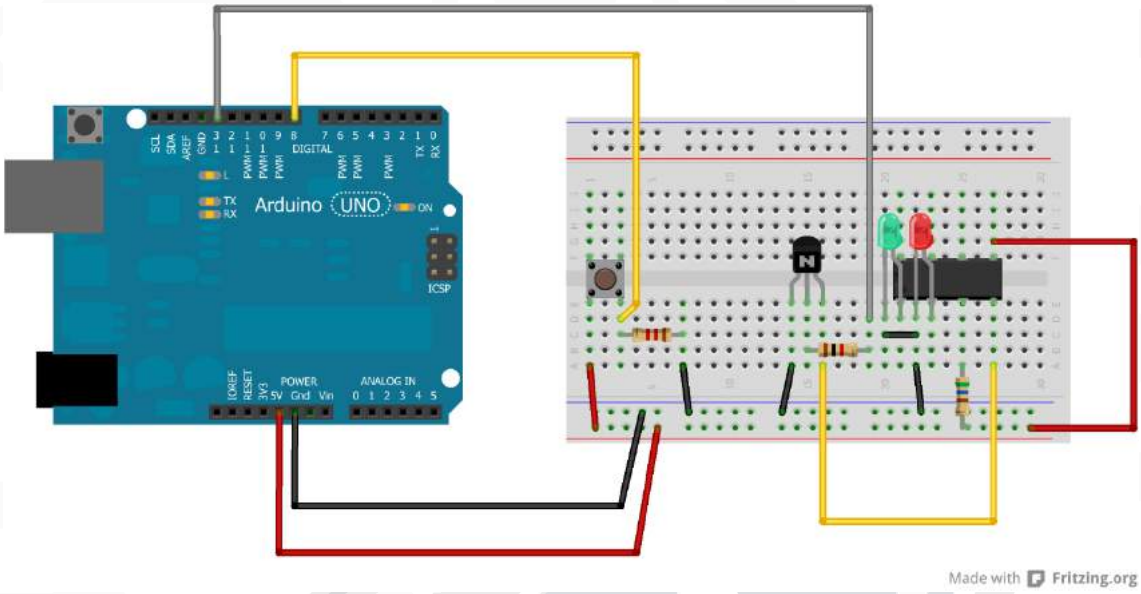


Figura 26- Esquemático módulo relé.

Uma visão mais próxima da protoboard:

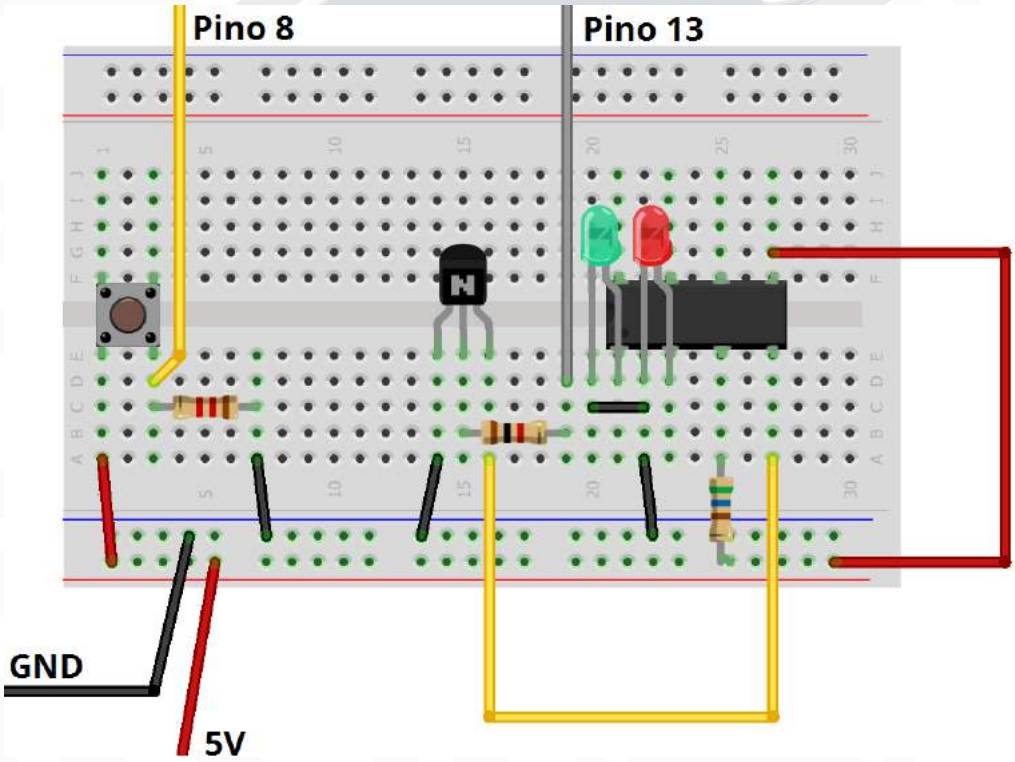


Figura 27 - Esquemático ligação relé.

Para ficar mais fácil de visualizar o circuito do transistor, veja o esquemático a

seguir:

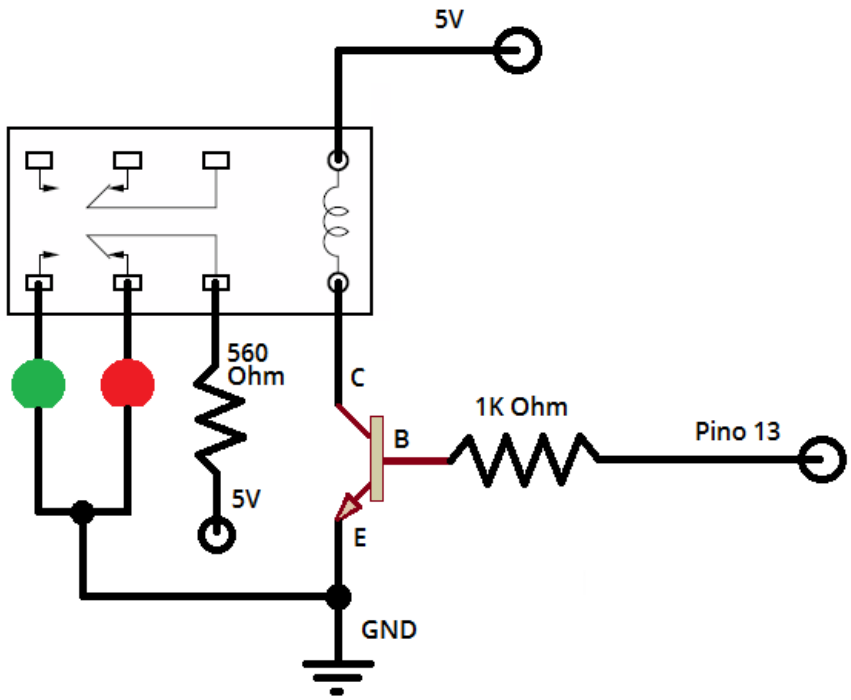


Figura 28 – Esquemático de ligação do relé.

Lembre-se de verificar a polaridade do LED. O lado negativo tem a perna menor e possui um chanfre reto no LED.

Para identificar os pinos do transistor, use a seguinte referência:

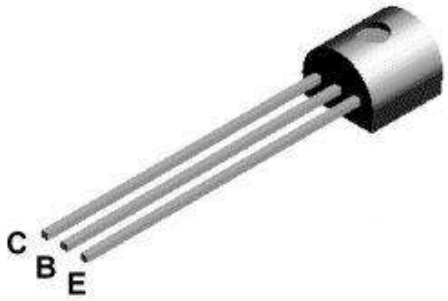


Figura 29 - Pinos do transistor BC337

3.9.3. 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 que a placa configurada é a que você está usando (board).

3.9.4. Preparando a cobertura

Esse é o mesmo programa utilizado no exemplo anterior. Caso já tenha ele salvo, use o mesmo.

Mas caso contrário, crie um programa (sketch) e salve com o nome de "programa_modulo_rele".

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

```
bool stable;    // Guarda o último estado estável do botão;
bool unstable;  // Guarda o último estado instável do botão;
uint32_t bounce_timer;
uint8_t counter = 0;
int rele=13;
int botao=8;
bool set;

bool changed() {
  bool now = digitalRead(botao);   // Lê o estado atual do botão;
  if (unstable != now) {       // Checa se houve mudança;
bounce_timer = millis();   // Atualiza timer;
unstable = now;            // Atualiza estado instável;
  }
  else if (millis() - bounce_timer > 10) {  // Checa o tempo de trepidação acabou;
    if (stable != now) {           // Checa se a mudança ainda persiste;
      stable = now;                  // Atualiza estado estável;
      return 1;
    }
  }
  return 0;
}

void setup() {
  Serial.begin(9600); // configura comunicação serial a uma taxa de 9600 bauds.
  pinMode(botao, INPUT_PULLUP);  // Configura pino 8 como entrada e habilita pull up interno;
  pinMode(rele, OUTPUT);  // Configura pino 8 como entrada e habilita pull up interno;
stable = digitalRead(botao);
  set=0;
}

void loop() {
if (changed()) { //verifica se houve mudança de estado
     if (digitalRead(botao)) set=!set; //se mudou o estado e o botão está pressionado muda o
valor de set
  }
 digitalWrite(rele,set);
  // Outras tarefas;
}
```

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

3.9.5. Experimentando o prato

Se tudo deu certo, quando você ligar seu Arduino o led Vermelho deverá estar ligado e o verde desligado. Quando apertar o botão uma vez, o led vermelho irá apagar e o verde acenderá. Apertado mais uma vez o botão, teremos a condição inicial novamente. E assim por diante.

Na imagem a seguir temos o estado inicial, bonina desligada e por isso o LED vermelho ligado (contato normalmente fechado):

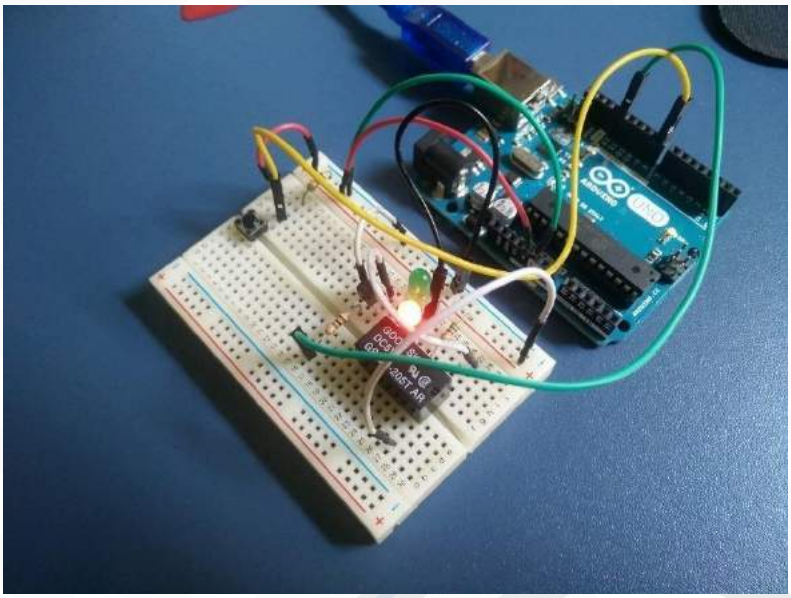


Figura 30 - LED vermelho aceso

Apertando o botão uma vez, temos o a bobina energizada. Logo o LED vermelho apaga e o led verde acende.

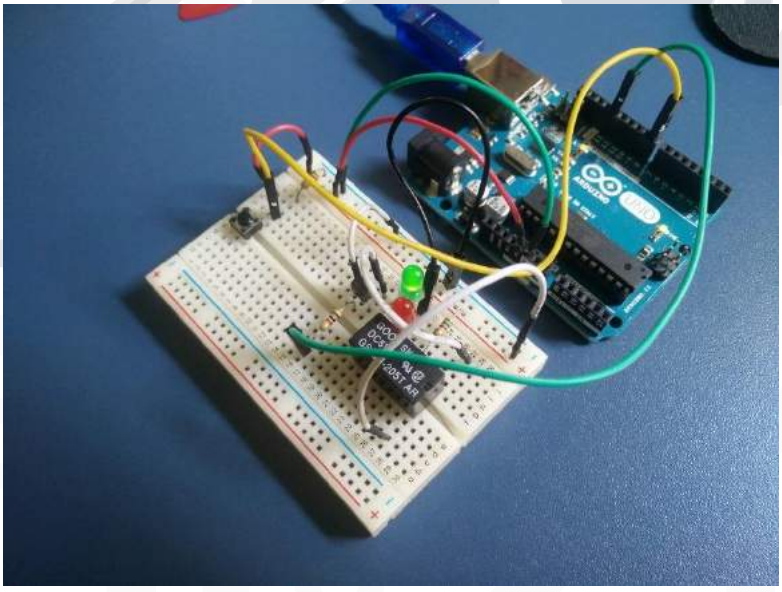


Figura 31 - LED verde aceso.

Podemos usar esse padrão de circuito com transistores para vários tipos de componentes levando em conta as explicações básicas sobre transistores feita no início desse capítulo.

4. Motores

Uma das formas de robôs e sistemas automatizados interagirem com o meio ambiente é através de movimentos físicos. Para essa tarefa encontramos com frequência motores elétricos.

Existem diversas forma de classificar motores elétricos, uma delas é quanto ao tipo de corrente: continua ou alternada. Em geral, para projetos de robótica iremos usar com mais frequência os motores de corrente contínua que no qual iremos aprender a usá-los aqui.

Em resumo, um motor de corrente contínua (motor DC ou motor CC) utiliza propriedades magnéticas para transformar energia elétrica em movimentos através de um sistema engenhoso.

Vamos considerar um modelo básico de um motor para que possamos entender rapidamente o conceito por trás de seu funcionamento.

O motor é dividido em duas partes importantes. O rotor, que na maioria dos casos é a parte girante, onde temos o eixo do motor. E o estator, que é a parte fixa do motor na maioria dos casos, ou seja, a parte externa.

Veja na nossa figura abaixo que o rotor é representado pela parte girante no centro do motor e o estator é a parte fixa externa. Veja também que o rotor tem um lado vermelho (polo norte) e um lado azul (polo sul). O mesmo vale para o estator.

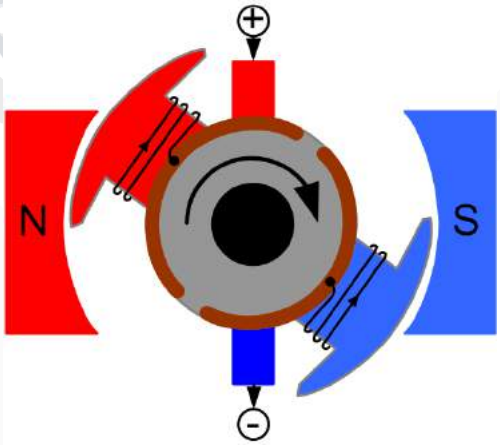


Figura 32 - Campo magnético motor DC

Como sabemos, os opostos se atraem. O campo magnético cria uma força de atração entre os pólos sul e norte. Essa força faz com que o eixo se mova até que o pólo sul do rotor fique o mais próximo possível de pólo norte do estator. E o pólo norte do rotor fique o mais próximo possível de pólo sul do estator. Até que consiga chegar a esse ponto, o eixo do motor irá se movimentar.

Até aqui estamos entendidos, o rotor rodar até que o seu pólo sul esteja o mais próximo quanto possível do pólo norte do estator. E quando chegarmos a esse ponto, como que o movimento continua? Simples. Existe um sistema de comutação no eixo. Como assim?

O campo magnético gerado no rotor é resultado da corrente elétrica que passa pelo rotor. Essa corrente gera um campo eletromagnético que vai variar o sentido dos pólos conforme variar o sentido da corrente.

Veja na figura anterior que temos um ponto de alimentação positiva, sinalizado por um sinal de mais, e um ponto de alimentação negativa, sinalizado pelo sinal menos.

Quando alimentamos o rotor, ele irá criar um campo magnético e irá gerar um pólo sul de um lado e o pólo norte do outro. Para que o rotor se movimente sem parar.

A grande engenhosidade é fazer com que conforme o eixo gire, o sentido de alimentação mude, invertendo o sentido da corrente e por consequência, também os pólos.

Veja a figura abaixo. No centro do rotor podemos ver que temos dois contatos laranjas, e dois contatos ligados a uma fonte, um vermelho (positivo) e outro azul (negativo). Observe que ao girar o rotor, os contatos laranjas vão mudar de positivo para negativo e de negativo para positivo.

Isso fará com que a corrente dentro da bobina mude de direção e por fim o campo eletromagnético também terá seus pólos invertidos.

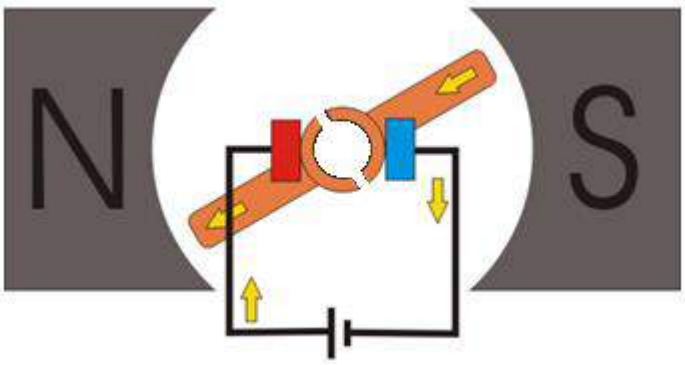


Figura 33 - Comutador motor DC

Então vamos resumir o que acontece:

1. A corrente que passar pelo rotor cria um campo eletromagnético com pólos sul e norte;
2. O pólo sul do rotor será atraído pelo pólo norte do estator, provocando um movimento circular;
3. Esse movimento circular do rotor fará com o pólo sul do rotor se aproxime do pólo norte do estator;

4. Com essa aproximação e movimento, haverá uma comutação dos pólos que alimentam o rotor, fazendo com que a corrente dentro dele mude de sentido e por consequência os pólos se invertam;
5. O pólo que antes era sul, agora será norte e será atraído para o pólo sul do outro lado do estator fazendo com que o movimento continue;
6. E assim continua o movimento do motor.

Acredito que conseguimos entender um pouco do funcionamento de um motor. Recomendo que veja vídeos no youtube explicando o funcionamento de um motor DC. É um assunto interessante.

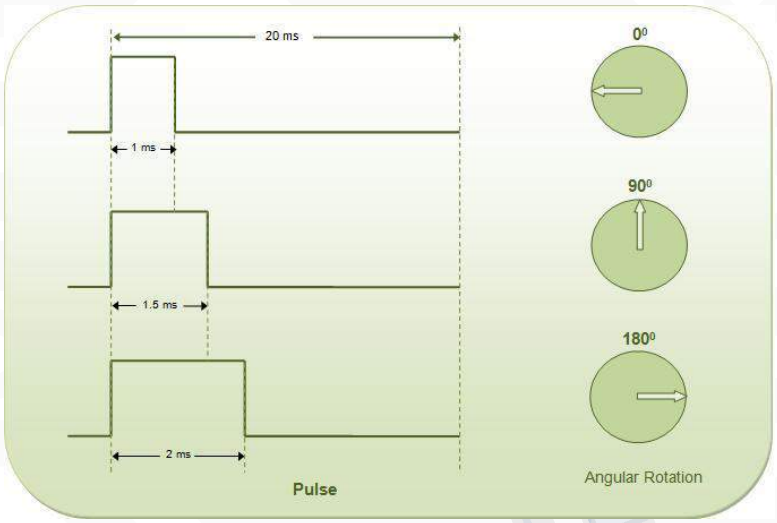
Vamos então começar a usar alguns tipos de motores DC.

5. Servomotores

Entre os atuadores temos um motor bem especial. Os servomotores são muito utilizados quando o assunto é robótica. De forma simplificada, um servomotor é um motor na qual podemos controlar sua posição angular através de um sinal PWM (Leia mais sobre PWM na postagem Grandezas digitais e analógicas e PWM.).

Figura 34 - Servo

Dessa forma, um servomotor é um atuador eletromecânico utilizado para posicionar e manter um objeto em uma determinada posição. Para isso, ele conta com um circuito que verifica o sinal de entrada e compara com a posição atual do eixo.



Como você pode ver na figura anterior, o ângulo do servo é proporcional ao Duty Cycle (tempo que o sinal é positivo) do sinal PWM.

Diferentemente dos motores CC ou motores de passo que podem girar indefinidamente, o eixo dos servomotores possui a liberdade de apenas 180º. Existem ainda alguns servos que são adaptados para girar indefinidamente, mas não entraremos nesse mérito aqui.

Servomotores geralmente possuem 3 pinos:

- Alimentação positiva (vermelho) – 5V;
- Terra (Preto ou Marrom) – GND;
- Controle. (Amarelo, Laranja ou Branco) – Ligado a um pino digital de entrada e saída;

Atenção!!

Servomotores consomem uma corrente significativa ao se movimentarem. A utilização de uma fonte externa pode ser necessária e é recomendada. Lembre-se de conectar o pino GND da fonte externa ao GND do Arduino para que a referência seja a mesma.

Apesar de sua posição ser controlada através do duty cycle de um sinal PWM enviado ao pino de controle não é necessário a conexão do pino de controle a um pino que possua PWM, pois utilizaremos a biblioteca Servo.h.

A utilização de analogWrite (PWM) produzirá um controle de menor precisão e poderá até danificar alguns servos por sua frequência (490 Hz) ser 10 vezes superior a frequência típica de controle de alguns servos.

Além de mais preciso e recomendado, o uso da biblioteca Servo.h faz com que o uso do servo seja mais fácil. Isso se dá pelo fato de você só precisar definir o ângulo que você deseja, não necessitando o uso dos valores de PWM (0 a 255).

5.1. Experiência 1

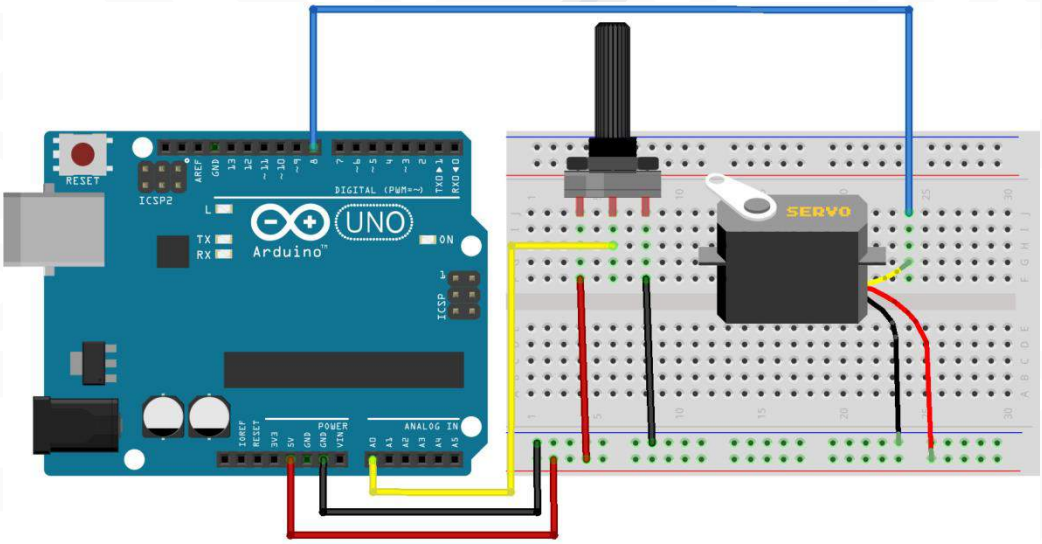
Então vamos fazer uma experiência e testar como funciona o servo?

5.1.1. Ingredientes

- Potenciômetro
- MicroServo 9g
- Fios Jumper's
- Protoboard

5.1.2. 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 microservo tem três fios saindo dele. Um será vermelho e irá para os +5 V. Outro, preto ou marrom, irá para o GND. O terceiro, branco, amarelo ou laranja, será conectado ao pino digital 8.

Monte o potenciômetro com os seus pinos externos conectados um no +5 V e o outro no GND e o pino do meio ao pino analógico A0.

5.1.3. 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 que a placa configurada é a que você está usando (board).

5.1.4. Preparando a cobertura

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

Nesse exemplo utilizaremos a biblioteca Senvo.h que deve ser adicionada em: Sketch > Import Library > Servo.

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

```
#include <Servo.h>
Servo servo1; // cria um objeto servo

// Esta função "setup" roda uma vez quando a placa e ligada ou resetada
void setup() {
servo1.attach(5); // anexa o servo (físico), no pino 5, ao objeto servo (lógico)
}

void loop() {
int angulo = analogRead(0); // Lê o valor do potenciômetro
angulo = map(angulo, 0, 1023, 0, 180); // Mudança de Escala
servo1.write(angulo); // Escreve o ângulo para o servo
delay(20); // Espera de 20ms, Suficiente para que o servo atinja a posição
}
```

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

5.1.5. Experimentando o prato

Se tudo deu certo, conforme você variar a resistência do potenciômetro o servo irá se mover.

5.2. Entendendo o Software

- *Biblioteca Servo.h*

Na elaboração do software utilizaremos a biblioteca Servo.h. Esta biblioteca implementa as funcionalidades de um servomotor tornando sua utilização extremamente simples. Entretanto alguns cuidados devem ser tomados.

> A biblioteca suporta a ligação de até 12 servomotores na maioria das placas Arduino e 48 no Arduino Mega. O uso da biblioteca desabilita o uso da função analogWrite nos pinos 9 e 10 (*exceto no Arduino Mega). No Mega o uso de 12 a 23 servomotores desabilitará o a função analogWrite nos pinos 11 e 12.

- *Declarando um Servo*

Ao usar essa biblioteca trataremos cada servo como um objeto, dessa forma precisamos declará-lo no início do código.

```
Servo servo1; // Cria um objeto servo
```

Depois de declarado, sempre que quisermos mexer em alguma função desse servo, devemos usar o nome da função precedida do nome do servo e ponto.

```
servo1.exemplo(); // chama função exemplo() para o objeto servo1
```

Você poderá declarar quantos servos for usar, levando em conta a limitação física de sua placa Arduino. Cada servo pode ter qualquer nome, mas é aconselhável que se use nomes intuitivos.

- *Declarando porta de controle do Servo*

Agora é preciso definir em que porta esta conectado o fio de controle do servo, para isso usamos a função attach(pino).

```
servo1.attach(5); // Anexa o servo (físico), no pino 5, ao objeto servo (lógico)
```

- *Controlando a Posição do Servo*

A função write define em um servo padrão o ângulo em graus na qual ele deve se posicionar.

```
servo1.write(angulo); //angulo: posição em graus para servos comuns
```

6. Módulo Ponte H – Controlando o sentido de um motor DC

Se você curte robótica, provavelmente deve estar louco para montar seu próprio robô. A ponte H é uma peça chave quando o assunto é robótica.

6.1. O que é uma Ponte H?

Os motores DC (direct current ou corrente continua) são cargas indutivas que, em geral, demandam uma quantidade de corrente superior à que as portas do Arduino conseguem fornecer.

Figura 35 - Exemplo de motor DC

Sendo assim, não devemos ligar estes motores diretamente nas portas do Arduino pois se o motor demandar uma corrente acima de 40mA nas portas digitais (máxima fornecida pelo Arduino) pode queimar a porta e danificar a placa.

Para solucionar a questão da alta corrente poderíamos usar transistores, porém é importante que seja possível controlar o sentido de giro do motor, função que não se faz possível usando apenas um transistor já que para inverter o sentido de giro devemos inverter a polaridade da alimentação do motor (onde era positivo se põe negativo e vice-versa). Um transistor só seria suficiente para ligar e desligar o motor.

Para resolver nosso problema utilizamos um famoso circuito conhecido como Ponte H que nada mais é que um arranjo de 4 transistores. Este circuito é uma elegante solução por ser capaz de acionar simultaneamente dois motores controlando não apenas seus sentidos, como também suas velocidades.

6.1.1. Mas como funciona a Ponte H? Por que este nome?

As pontes H possuem este nome devido ao formato que é montado o circuito, semelhante a letra H. O circuito utiliza quarto chaves (S1, S2, S3 e S4) que são acionadas de forma alternada, ou seja, (S1-S3) ou (S2-S4), veja as figuras abaixo. Dependendo da configuração entre as chaves teremos a corrente percorrendo o motor hora por um sentido, hora por outro.

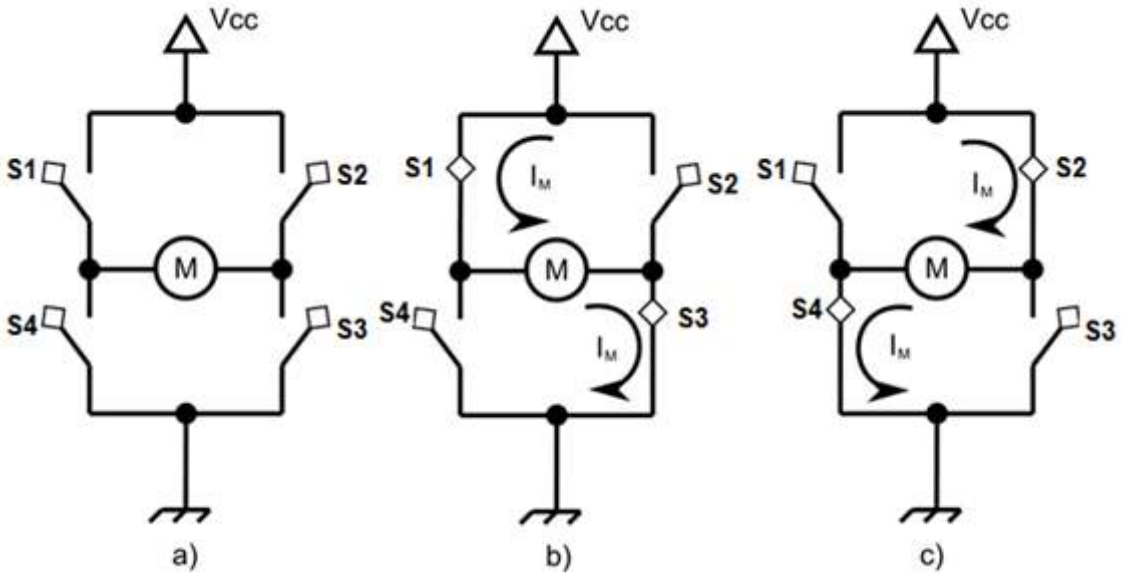


Figura 36 - Circuito Ponte H

Quando nenhum par de chaves está acionado, o motor está desligado (a). Quando o par S1-S3 é acionado a corrente percorre S1-S3 fazendo com que o motor gire em um sentido (b). Já quando o par S2-S4 é acionado a corrente percorre por outro caminho fazendo com que o motor gire no sentido contrário (c).

6.1.2. Circuito integrado L2398N

O CI L298N é muito utilizado para o propósito de controle de motores, ele nada mais é que uma ponte H em um componente integrado. Uma das vantagens do uso desse CI é o menor espaço ocupado, a baixa complexidade do circuito e o fato de ele já possuir duas pontes H, podendo assim, controlar dois motores. Na figura a seguir você pode conferir o diagrama de blocos do CI L298N retirado de sua folha de dados (folha de dados L298N):

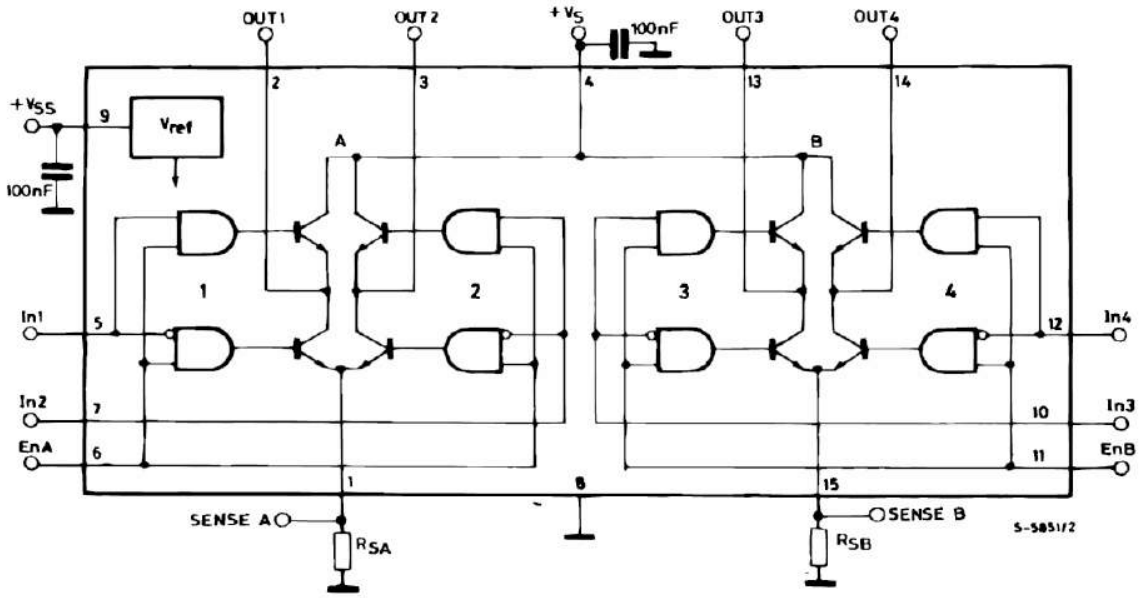


Figura 37 - Esquema dos circuitos internos do CI L298n

As funções dos principais pinos desse CI são descritas na tabela a seguir:

Nome	Função
Sense A; Sense B	Entre esse pino e o Terra é conectado um resistor sensitivo para controlar a corrente da carga.
Out 1; Out 2	Saídas da Ponte A. A corrente que flui através da carga conectada entre esses dois pinos, é monitorada pelo pino 1.
Input 1; Input 2	Entradas da Ponte A (compatíveis com nível TTL).
Enable A; Enable B	Entradas (compatíveis com nível TTL); nível baixo desabilita a Ponte A (*enable A*) e/ou a Ponte B (*enable B*).
Input 3; Input 4	Entradas da Ponte B (compatíveis com nível TTL).
Out 3; Out 4	Saídas da Ponte B. A corrente que flui através da carga conectada entre esses dois pinos, é monitorada pelo pino 15.

Figura 38 - Funções dos principais pinos da ponte H L298N [2]

Outra vantagem da ponte H L298N é a resposta a sinais de PWM. Se no lugar de usar sinais lógicos TTL for usado sinais de PWM, é possível regular a tensão de saída, e dessa forma regular a velocidade dos motores.

O PWM, Pulse Width Modulation (Modulação por Largura de Pulso), consiste basicamente em aplicar uma onda quadrada de amplitude Vcc e frequência alta no lugar da tensão continua Vcc. Leia mais sobre PWM na postagem Grandezas digitais e analógicas e PWM.

Ao usar um sinal de PWM nas entradas IN1 e IN2, por exemplo, teremos uma tensão de saída nos pinos OUT1 e OUT2 em PWM que será igual à *Duty Cycle**Vcc. Dessa forma, podemos regular a diferença de potencial média aplicada nos motores, controlando as suas velocidades.

Existem outras opções de CI's de ponte H no mercado, é importante consultar as

especificações deles em suas folhas de dados (Datasheet) para saber qual irá lhe atender melhor. Veja algumas opções de ponte H:

- L293D Folha de Dados;
- LMD18200 Folha de dados.

6.1.3. Módulos de Ponte H

Módulos de ponte H são muito utilizados em aplicações de robótica. Esses módulos possuem dimensões pequenas e já possuem o circuito básico para o uso do CI, o que facilita na acomodação do módulo no robô ou em outros projetos e a sua utilização.

Existem varias opções disponíveis no mercado, com tamanhos e especificações diferentes. Algumas especificações são importantes ao escolher seu módulo de ponte H, são elas:

- Especificação de potência máxima fornecida;
- Tensão máxima suportada;
- Corrente máxima suportada;
- Tensão lógica.
- Ci L298N;
- Tensão para os motores: 5 – 35V;
- Corrente máxima para os motores: 2A;
- Potência máxima: 25W;
- Tensão lógica: 5V;
- Corrente lógica: 0-36mA;

Figura 39 - Módulo Ponte H com o CI L298N

6.2. Módulo Ponte H com CI L298N

Agora que já sabemos como a Ponte H funciona, vamos entender na prática como podemos usá-la em conjunto com o Arduino. Para isso iremos usar o módulo Ponte H com CI L298N.

6.2.1. Entradas e saídas

Para começa vamos entender função de cada pino bem como deve ser utilizado.

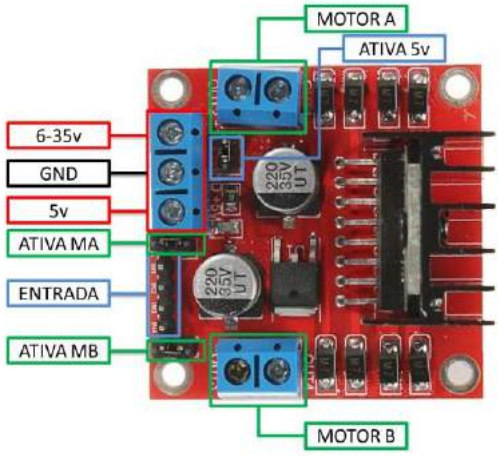


Figura 40 - Entradas e saídas do módulo Ponte H

- **Motor A e Motor B**: Conectores para os dois motores
- **6-35V**: Porta para alimentação da placa com tensão entre 6 a 35V.
- **Ativa 5V**: Quando jumpeado, a placa utilizará o regulador de tensão integrado para fornecer 5v (na porta 5v) quando a porta 6-35V estiver sendo alimentada por uma tensão entre 6 e 35V. Neste caso, não se deve alimentar a porta 5V pois pode danificar os componentes. A tensão fornecida na porta 5V pode ser usada para alimentar o Arduino, por exemplo.
- **5v**: Em casos de não haver fonte de alimentação com mais de 6V podemos alimentar a placa com 5V por esta porta.
- **Ativa MA**: Quando jumpeado aciona o motor A com velocidade máxima. Para controlar a velocidade do motor A basta remover o jumper e alimentar o pino com uma tensão entre 0 e 5v, onde 0V é a velocidade mínima (parado) e 5V a velocidade máxima.
- **Ativa MB**: Quando jumpeado aciona o motor B com velocidade máxima. Para controlar a velocidade do motor B basta remover o jumper e alimentar o pino com

uma tensão entre 0 e 5v, onde 0V é a velocidade mínima (parado) e 5V a velocidade máxima.

- **IN1 e IN2:** são utilizados para controlar o sentido do motor A;
- **IN3 e IN4**: são utilizados para controlar o sentido do motor B;

Veja que agora, no lugar das chaves S1-S3 e S2-S4 temos os pinos IN1 e IN2. Onde IN1 corresponde às chaves S1-S3 e a IN2 às chaves S3-S4.

Para controlar o sentido, temos as seguintes combinações para o motor A(IN1 e IN2)

IN1	IN2	Estado
0V	0V	Desligado
0V	5V	Sentido 1
5V	0V	Sentido 2
5V	5V	Freio

Figura 41 - Tabela de combinações

Para o motor B (IN3 e IN4), a tabela funciona da mesma forma.

6.3. Experiência3

Vamos fazer um exemplo para testar na pratica a ponte h. Neste primeiro exercício queremos testar o controle do sentido de giro dos motores A e B através do Arduino.

6.3.1. Ingredientes

- Arduino UNO
- Ponte H
- 2 Motores DC 12V (pode ser feito com apenas 1)
- Fonte alimentação de 12V

6.3.2. Misturando os ingredientes

Prossiga com a montagem conforme esquema abaixo (caso você use apenas um motor, basta desconsiderar o motor B):

Garanta que seu Arduino e a fonte externa estejam desligados durante a montagem.

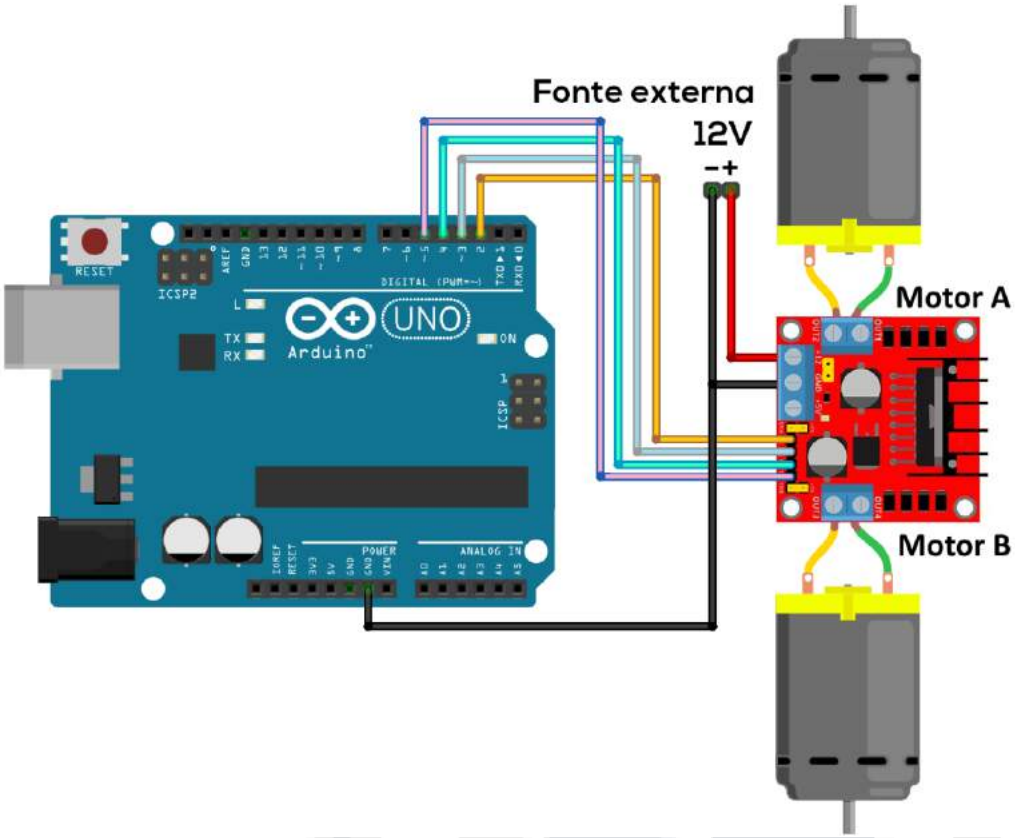


Figura 42 - Esquema de montagem exemplo 1

6.3.3. 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).

6.3.4. Preparando a cobertura

Agora vamos à implementação do programa. Dessa forma, dentro da IDE Arduino: crie um programa (sketch) e salve com o nome de "programa_ponte_H".

Com o seu programa salvo, escreva o seguinte código e ao final clique em Upload para que o programa seja transferido para seu Arduino.

Observe que nos comentários escrevemos "sentido horário" e "sentido anti-horário". Porem, você só poderá identificar isso com a execução do programa.

Para que você saiba qual o sentido de rotação quando for usar a ponte H, caso o motor gire em sentido diferente ao comentário do programa, você pode inverter os comentários.

```
/*Pinagem do arduino*/

//motor A
int IN1 = 2 ;
int IN2 = 3 ;

//motor B
int IN3 = 4 ;
int IN4 = 5 ;

// Esta função "setup" roda uma vez quando a placa e ligada ou resetada
void setup(){
 //Inicializa Pinos
 pinMode(IN1,OUTPUT);
 pinMode(IN2,OUTPUT);
 pinMode(IN3,OUTPUT);
 pinMode(IN4,OUTPUT);
}

// Função que se repete infinitamente quando a placa é ligada
void loop(){

 /*Inicio dos Estados do motor A*/

 //Sentido Horario
 digitalWrite(IN1,HIGH);
 digitalWrite(IN2,LOW);
 delay(5000);

 //Freia Motor A
 digitalWrite(IN1,HIGH);
 digitalWrite(IN2,HIGH);
 delay(5000);

 //Sentido Anti-Horario
 digitalWrite(IN1,LOW);
 digitalWrite(IN2,HIGH);
 delay(5000);

//Freia Motor A
 digitalWrite(IN1,HIGH);
 digitalWrite(IN2,HIGH);
 delay(5000);

 /*Fim dos Estados do motor A*/

 /*Início dos Estados do motor B*/

 //Sentido Horario
 digitalWrite(IN3,HIGH);
 digitalWrite(IN4,LOW);
 delay(5000);

 //Freia Motor B
 digitalWrite(IN3,HIGH);
 digitalWrite(IN4,HIGH);
 delay(5000);

 //Sentido Anti-Horario
 digitalWrite(IN3,LOW);
 digitalWrite(IN4,HIGH);
 delay(5000);

 //Freia Motor B
 digitalWrite(IN3,HIGH);
 digitalWrite(IN4,HIGH);
 delay(5000);

 /*Fim dos Estados do motor B*/
}
```

6.3.5. Experimentando o prato

Se tudo der certo, o motor A irar girar para um lado, parar, girar para o outro e depois parar. Em seguida a mesma coisa irá acontecer para o motor B.

Como comentado anteriormente, caso os seus motores girem em sentido diferente, altere os comentários para que fiquem devidamente organizados.

6.4. Entendendo o programa

Como podemos ver, esse é um programa simples. Nós apenas alteramos as combinações de IN1 e IN2 para o motor A e IN3 e IN4 para o motor B como na tabela a seguir.

IN1	IN2	Estado
0V	0V	Desligado
0V	5V	Sentido 1
5V	0V	Sentido 2
5V	5V	Freio

Figura 43 - Tabela de combinações

7. Ponte H – Controlando a velocidade de um motor DC

No último capítulo vimos como se faz para ligar um motor DC no Arduino com o auxílio de um módulo ponte H, usando suas entradas para variar o sentido de giro. Mas na robótica é comum termos que controlar, além do sentido, a velocidade do motor. Nesse capítulo iremos ensinar, de forma simples e didática, como podemos controlar a velocidade de um motor DC.

Como controlar a velocidade de um motor?

Um motor DC gira baseado em campos magnéticos gerados pela corrente que passa em suas bobinas. Para variar a velocidade do motor podemos alterar essa corrente que é diretamente proporcional a tensão sobre elas.

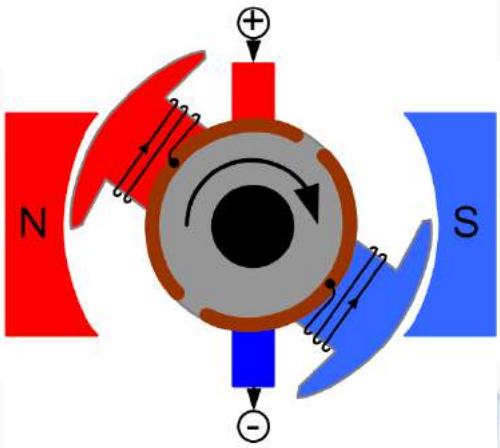


Figura 44 - Campo magnético motor DC

Dessa forma, com a mudança da tensão em cima do motor, teremos uma alteração de velocidade. **Mas como podemos fazer isso usando o Arduino e a Ponte H?** A solução é simples e eficiente e se chama PWM.

Modulando a ponte H

No módulo Ponte H com CI L298N cada ponte H possui um pino que ativa ou não a ponte H. Caso tenha um sinal de 5V inserido nele, a ponte estará ligada, caso seja 0V a ponte estará desligada. Como temos 2 pontes H, temos o Enable A(Ativa A) e o Enable B (Ativa B).

Normalmente o Enable A e B fica em curto com um sinal de 5V da placa através de um jumper.

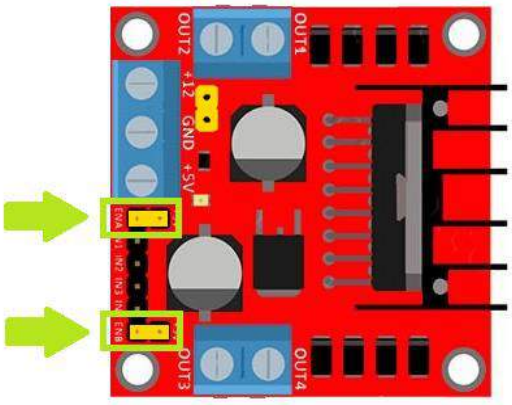


Figura 45 - Jumpers Enable A e B

Se retirarmos esse jumper e inserimos um sinal PWM nessa entrada, modularemos a tensão que é enviada para o motor no mesmo formato. Isso ocorre porque a ponte H só ira "funcionar" enquanto o sinal de Enable estiver com 5V.

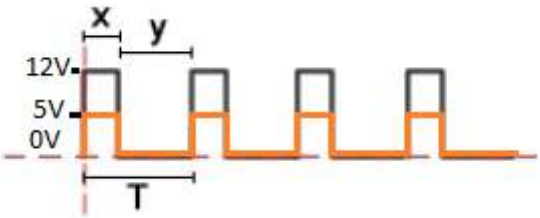


Figura 46 - Sinal PWM no Enable A em orange (5V) e a saída para o motor A em preto (12V).

Sendo assim, a saída para o motor será um sinal PWM com um Duty Cycle igual ao do Enable e terá tensão média calculada pela seguinte fórmula.

Vmédio = Vmax(tensão PonteH)*Duty Cycle(%)

Com essa modulação, podemos variar a velocidade do motor através de PWM.

7.1. Experiência 4

Neste segundo exemplo, vamos verificar o controle de velocidade dos motores A e B.

7.1.1. Ingredientes

- Arduino UNO
- Ponte H
- 2 Motores DC 12V (pode ser feito com apenas 1)
- Fonte alimentação de 12V

7.1.2. Misturando os ingredientes

Prossiga com a montagem conforme esquema abaixo (caso você use apenas um motor, basta desconsiderar o motor B):

Garanta que seu Arduino e a fonte externa estejam desligados durante a montagem.

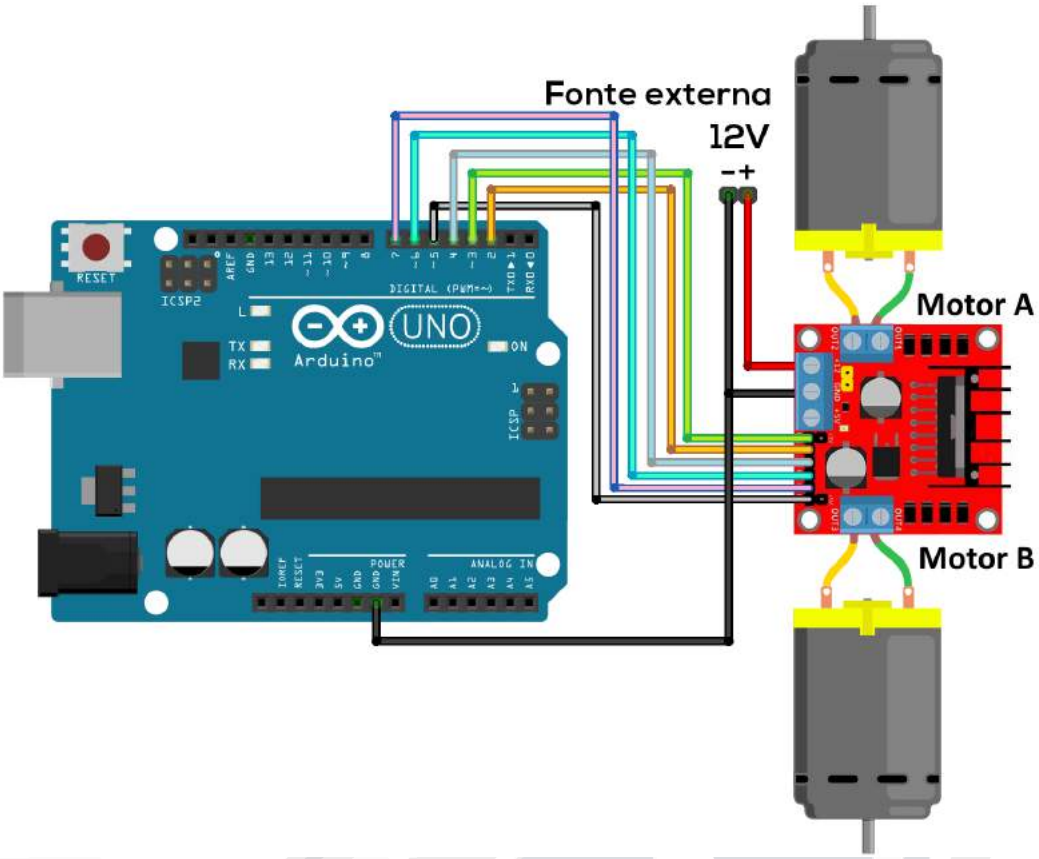


Figura 47 - Esquema de montagem exemplo 1

7.1.3. Preparando a cobertura

Agora vamos à implementação do programa. Dessa forma, dentro da IDE Arduino: crie um programa (sketch) e salve com o nome de "programa_ponte_H_velocidade".

Com o seu programa salvo, escreva o seguinte código e ao final clique em Upload para que o programa seja transferido para seu Arduino.

```
/*Pinagem do arduino*/

//motor A
int IN1 = 2 ;
int IN2 = 4 ;
int velocidadeA = 3;

//motor B
int IN3 = 6 ;
int IN4 = 7 ;
int velocidadeB = 5;

//variavel auxiliar
int velocidade = 0;

// Esta função "setup" roda uma vez quando a placa e ligada ou resetada
void setup(){
 //Inicializa Pinos
 pinMode(IN1,OUTPUT);
pinMode(IN2,OUTPUT);
pinMode(IN3,OUTPUT);
pinMode(IN4,OUTPUT);
pinMode(velocidadeA,OUTPUT);
 pinMode(velocidadeB,OUTPUT);
}

// Função que se repete infinitamente quando a placa é ligada
void loop(){

/*Exemplo de velocidades no motor A*/

//Sentido Horario
digitalWrite(IN1,HIGH);
digitalWrite(IN2,LOW);

//Velocidade Alta
analogWrite(velocidadeA,230);

//Intermediaria
analogWrite(velocidadeA,150);

// Velocidade Baixa
analogWrite(velocidadeA,80);

//Freia Motor A
digitalWrite(IN1,HIGH);
digitalWrite(IN2,HIGH);

/*Exemplo de variacao de velocidade no motor B*/
 //Sentido Horario
digitalWrite(IN3,HIGH);
digitalWrite(IN4,LOW);

//velocidade de 0 a 255
while (velocidadeB < 255){
analogWrite(velocidadeB,velocidade);
velocidade = velocidade + 10;
delay(50);
}

//velocidade de 255 a 0
while (velocidadeB > 0){
analogWrite(velocidadeB,velocidade);
velocidade = velocidade - 10;
delay(50);
 }

 //Freia Motor B
digitalWrite(IN3,HIGH);
 digitalWrite(IN4,HIGH);

}
```

7.1.4. Experimentando o prato

Se tudo der certo, o motor A irar girar para um lado com uma velocidade Alta, depois intermediaria e finalmente devagar. Depois o motor B irá começar a girar variando a velocidade da menor velocidade até a velocidade máxima e vice-versa.

O programa irá repetir esse procedimento continuamente.

7.2. Entendendo o programa

Se você observar bem, a única diferença desse programa para o programa do capítulo anterior é o uso do PWM no Enable A, para o motor A, e no Enable B, para o motor B.

Ao executar o programa é fácil reparar que quanto menor o valor de PWM que colocamos no Enable da ponte H, menor a velocidade.

É importante ressaltar, que devido ao fato de o motor ser um dispositivo que envolve mecânica, existe a necessidade de uma força mínima para que ele comece a rodar. Para valores de PWM pequenos você irá reparar que não há qualquer movimento do motor. Dessa forma, existe um valor mínimo de PWM para que ele gire. Chamamos isso de zona morta.

A zona morta vai variar de motor para motor, mesmo que seja de mesmo modelo. Alguns motivos para a variação da zona morta:

- Lubrificação;
- Torque no eixo;
- Especificação do motor.

Tal como a zona morta, cada motor tem uma curva de velocidade, isso pode ser um problema na hora de fazer seu robô. Mas não é nada que será prejudicial em projetos amadores.

Para projetos mais profissionais é necessário traçar a curva de velocidade por PWM do motor para que possamos implementa uma função que aplique o PWM correto para a velocidade desejada.

Outra solução é o uso de encoders, que são sensores que medem a velocidade do motor. Assim podemos implementar um sistema de controle PID, por exemplo, que corrija a o PWM para que a velocidade da roda seja a que desejamos.

8. Super desafio

Agora que chegamos ao final desta apostila com tanta coisa aprendida, que tal um desafio?

Passarei dois, que julgo muito interessante e que vocês serão capazes de fazer com tudo o que foi aprendido e mais alguma consulta na internet.

8.1. Robô Autônomo

Sim, você é capaz! E vai ser muito divertido! Imagine a felicidade e orgulho que é dar vida a um robô. É emocionante!

Vou começar com um projeto simples, e você vai melhorando caso queira seguir os desafios. Recomendo comprar um chassi 2wd pronto, para que você possa focar no que realmente importa.

Vamos lá:

- Tarefa 1: Construa um robô seguidor de linha. Para isso você vai precisar de alguns módulos sensor de linha (geralmente usando o TCRT5000);
- Tarefa 2: Construa um robô que desvie de obstáculos usando dois sensores ultrassônico HCSR04;
- Tarefa 3: Construa um robô que desvie de obstáculos (sensor ultrassônico - HCSR04) e não caia em buracos (sensor seguidor de linha);
- Tarefa 4: Faça um robô que siga a luz! (Dica, use 3 sensores de luz LDR apontados para direções diferentes)

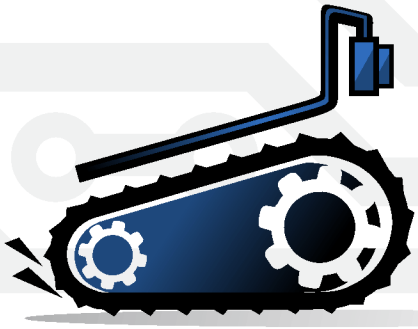
8.2. Automação residencial

Na apostila Vol 2, no super desafio, sugeri que fizesse um sistema de automação residencial. Você fez? Vou colocar aqui novamente e acrescentar mais algumas coisas!!

Somando tudo que você aprendeu na apostila Arduino Básico vol.1,2 e 3, 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 (HC SR501) 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. Usar um teclado matricial para configurar o sistema e ativar e desativar o alarme com senha;
E. Acender 1 lâmpada, caso seja noite, usando um módulo relé;
F. Um alarme, de preferência uma sirene de 127V ligada a um módulo relé, 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;
J. Use um módulo RTC para acionar sua cafeteira de manhã.



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, entusiastas, 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