OPEN ROBOTICS

INVENTOR 5.0

INSTALLATION & USE

OPENROBOTICS LTDA

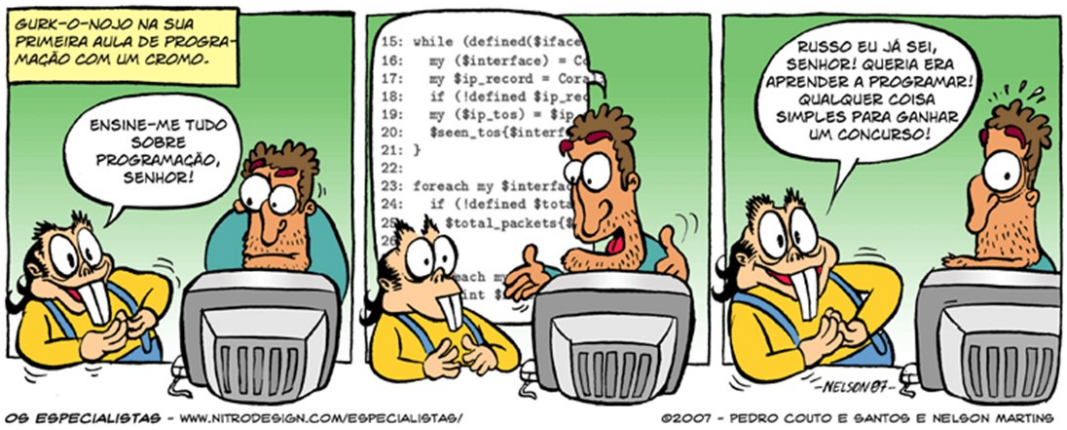
Rua Haia, 251—Bairro Santa Cruz Industrial

Contagem/MG—CEP 32340-260

GALILEO
EDITORA

Uma introdução a lógica de programação

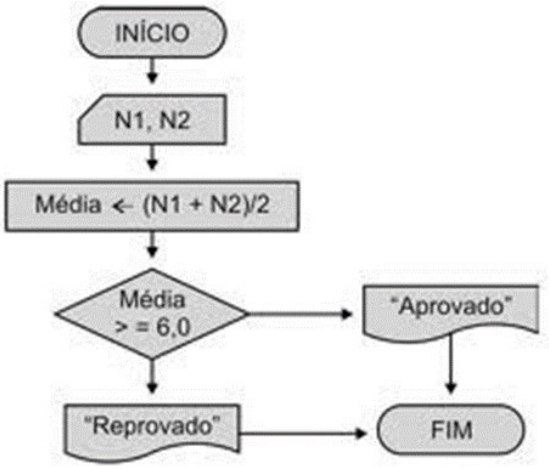
A lógica de programação é necessária para pessoas que desejam trabalhar com robótica, pois essa área do conhecimento envolve a criação de programas, que são utilizados para dar vida às estruturas criadas.



Podemos entender um programa como sendo “um conjunto de regras ou normas definidas para a realização ou emprego de algo”. Em informática um programa é a informação que indica a um computador uma **ação elementar a executar**. Convém ressaltar que uma ordem isolada não permite realizar o processo completo, para isso é necessário um conjunto de instruções colocadas em ordem sequencial lógica. Por exemplo, se quisermos fazer uma omelete, precisaremos colocar em prática **uma série de instruções**: descascar os, bater os ovos, colocar sal, etc... É evidente que essas instruções tem que ser executadas em uma ordem adequada – não se pode descascar as ovos depois de fritá-los.

Dessa maneira, uma **instrução tomada em separado não tem muito sentido**; para obtermos o resultado, precisamos colocar em prática o conjunto de todas as instruções, na ordem correta.

Vejamos um exemplo:



Algoritmos e programas

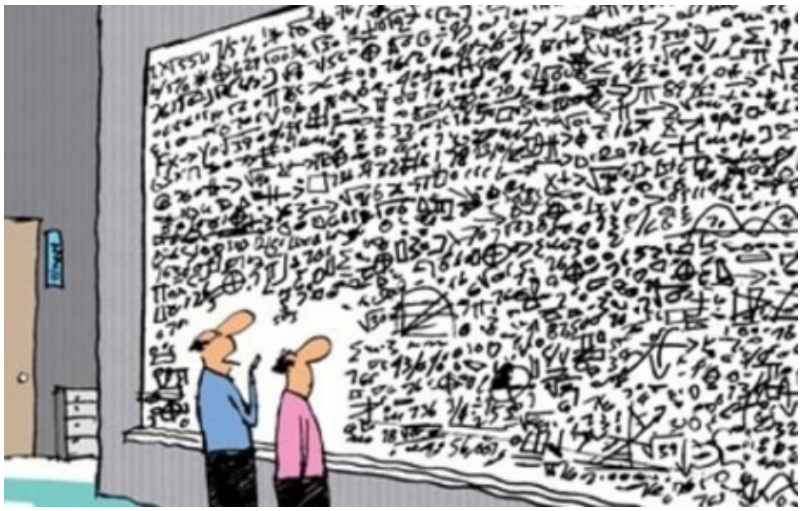
Um algoritmo é formalmente uma sequência finita de passos que levam a execução de uma tarefa. Podemos pensar em algoritmo como uma receita, uma sequência de instruções que dão cabo de uma meta específica. Estas tarefas não podem ser redundantes nem subjetivas na sua definição, devem ser claras e precisas.

Os programas de computadores nada mais são do que algoritmos escritos numa linguagem de computador (**Arduino, Pascal, C, Cobol, Fortran, Visual Basic entre outras**) e que são interpretados e executados por uma máquina, no caso um computador. Notem que dada esta interpretação rigorosa, um programa é por natureza muito específico e rígido em relação aos algoritmos da vida real.

```
Algoritmo Exemplo
     var x : numerico
Início
     Escreva "Escrevendo divisíveis por 2"
     x <- 0
     Enquanto x < 10 faça
          Se x%2 = 0
               Então
                    Escreva x
               Senão x <- x + 1
          Fim_Se
     Fim_Enquanto
Fim_Algoritmo
```

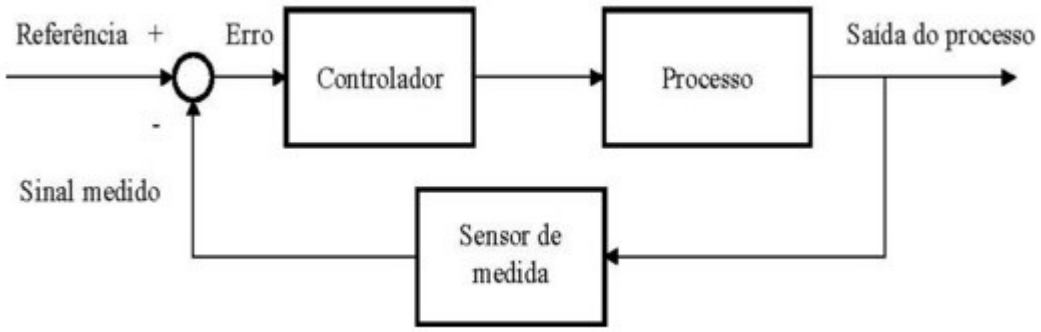
Acima temos um pequeno algoritmo que pode ser utilizado para escrever os números divisíveis por dois. No caso acima enquanto o valor de x for menor que 10 ele verifica se o resto da divisão deste número por 2 é zero (Se x%2 = 0) . Em caso positivo então ele escreve o valor (x). Em caso negativo ele passa para o próximo número (x <- x + 1)

Não se preocupe se estiver complexo, logo vamos aprender passo a passo como escrever um programa de computador para nosso **kit Inventor**.



Automação e controle

Para começar a entender os conceitos apresentados nesta tutorial e saber o que é um controlador lógico programável (CPU), é necessário entender o que é um controle, quais são seus elementos básicos e quais são os seus principais tipos. Uma estrutura para ser controlada deve dispor dos seguintes componentes básicos: Um sinal de referência (Entrada), um controlador, um processo, um sensor de medida. Para o entendimento de o que é um controle eletrônico é necessário o conhecimento destes componentes. O diagrama de um sistema de controle em malha fechada a seguir, apresenta estes elementos:



Referência:

É o sinal a ser obtido, ou seja, o valor alvo a ser atingido pelo sistema como um todo.

Controlador:

É a parte do sistema que faz o controle entre a referência e o sinal que está sendo medido nos sensores.

Processo:

É o equipamento ou recurso que será controlado pelo sistema

Sensor de medida:

São os dispositivos que realizam as leituras sobre o processo sendo executado.

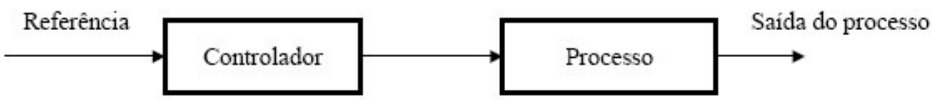
Um bom exemplo é uma geladeira. Em uma geladeira a temperatura interna é medida e comparada com um valor de referência pré-estabelecido pelo usuário. Caso a temperatura interna fique acima da temperatura pré-estabelecida (devido a temperatura externa ou a temperatura de algum alimento colocado na geladeira estar acima da temperatura interna), a geladeira é acionada e a temperatura começa a abaixar. Quando a temperatura atinge o valor de referência, o motor da geladeira é desligado. Desta maneira, a temperatura da geladeira tende a ficar em torno do valor de referência solicitado pelo usuário.

Ajustando o termostato de uma geladeira.

No controle em malha fechada, como no caso de uma geladeira, informações sobre como a saída de controle está evoluindo (no caso a temperatura) são utilizadas para determinar o sinal de controle que deve ser aplicado ao processo em um instante específico. Isto é feito a partir de uma realimentação da saída para a entrada. O diagrama básico de um sistema de controle em malha-fechada é mostrado na figura anterior. Em geral, a fim de tornar o sistema mais preciso e de fazer com que ele reaja a perturbações externas, o sinal de saída é comparado com um sinal de referência (chamado no jargão industrial de set-point) e o desvio (erro) entre estes dois sinais é utilizado para determinar o sinal de que deve efetivamente ser aplicado ao processo. Assim, o sinal de controle é determinado de forma a corrigir este desvio entre a saída e o sinal de referência. O dispositivo que utiliza o sinal de erro para determinar ou calcular o sinal de controle a ser aplicado à planta (sistema a ser controlado) é o controlador (ou compensador).

Circuito em malha aberta

Podemos ter também um sistema chamado de malha aberta, onde não temos o *feed-back* na saída do sinal, deixando o sistema menos robusto, mais simplificado. Veja na imagem abaixo:



Automatizar um sistema tornou-se muito mais viável à medida que a eletrônica avançou e passou a dispor de circuitos capazes de realizar funções lógicas e aritméticas com os sinais de entrada e gerar respectivos sinais de saída. Com este avanço, o controlador, os sensores e os atuadores passaram a funcionar em conjunto, transformando processo em um sistema automatizado, onde o próprio controlador toma decisões em função da situação dos sensores e aciona os atuadores.

Algoritmos de controle

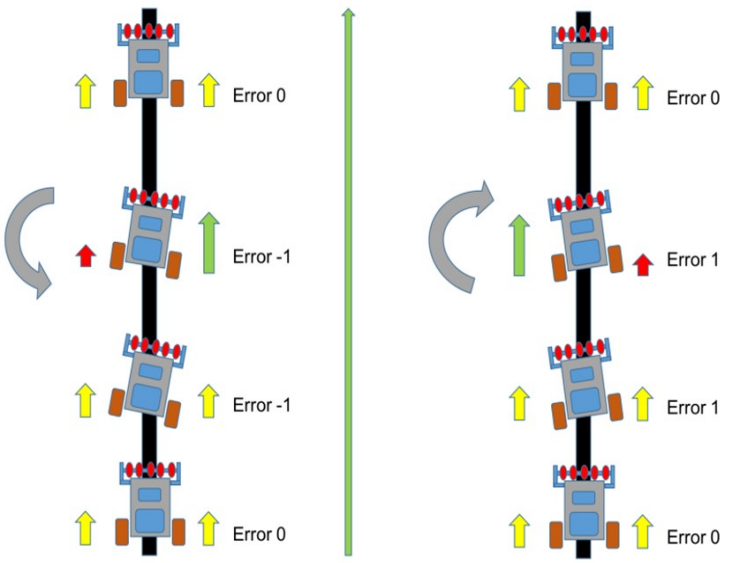
Conforme vimos, um problema de controle consiste em determinar sinais adequados para modificar ou não o valor de uma variável manipulada de um processo, a fim de se obter uma saída desejada. Para tanto, ao efetuarmos a medição de uma variável controlada, comparamos o valor obtido com o valor desejado. O erro é, então, processado em uma unidade denominada unidade de controle. O processamento é feito a partir de diferentes tipos de cálculos matemáticos (algoritmos) que determinam a **ação de controle** e, consequentemente, os efeitos corretivos no processo.

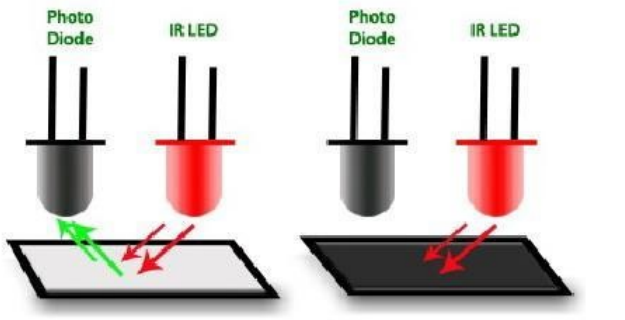
Existe uma enorme gama de controladores comerciais no mercado, cada uma com suas vantagens e desvantagens em relação a sua função. No nosso caso vamos utilizar o Inventor para realizar este controle, **através dos seus sensores**.

Controles do tipo liga e desliga

Em muitos sistemas básicos o controle pode ser efetuado a partir de uma simples chave liga-desliga que é ligada/desligada, por exemplo, a partir de uma determinada temperatura ou nível do reservatório. Nesse tipo de ação, o controlador compara o sinal de entrada com a realimentação e, se a saída superar a entrada, desliga o atuador; se a realimentação for menor, liga o atuador. É o caso por exemplo da nossa geladeira. Hora ela está ligada, e hora ela está desligada. Trata-se de uma chave que será acionada para **manter a temperatura** em determinado patamar.

As vantagens deste controlador são a simplicidade e o baixo custo. A desvantagem reside na contínua oscilação da saída entre os limites de atuação do controlador conhecida como histerese. Esta inerente instabilidade é devida à inexistência de uma realimentação negativa para diminuir o seu ganho que, teoricamente, é infinito. **A oscilação não garante precisão** e pode desgastar o controlador e o atuador pelo excesso de partidas. Um exemplo seria um robô simples que segue uma linha, como mostrado abaixo:

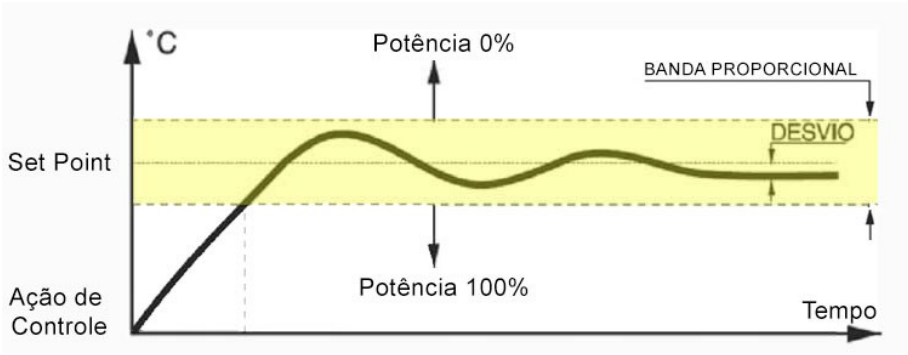




Nosso objetivo é manter o sensor de cor lendo a faixa preta fixa na estrada. Ele poderia sair para a esquerda ou para a direita e o sistema iria atuar para que a roda esquerda ou a roda direita seja modificada para que a direção seja ajustada. Veja que o sistema pode se tornar pouco eficiente, pois o carro poderia ficar virando para a **esquerda ou para a direita** de forma abrupta para manter o controle. Em um sistema mais avançado poderíamos ter um controle proporcional.

Controle proporcional

Foi visto anteriormente, que na ação liga-desliga, quando a variável controlada se desvia do valor ajustado, o elemento final de controle realiza um movimento brusco de ON (liga) para Off (desliga), provocando uma oscilação no resultado de controle, uma vez que ao se ligar ou desligar, temos um atraso grande até um novo ajuste no sistema. Para evitar tal tipo de movimento foi desenvolvido um tipo de ação no qual a ação **corretiva produzida por este mecanismo** é proporcional ao valor do desvio. Tal ação denominou-se ação proporcional.



Com os controles proporcionais a ação de controle seria “proporcional” ao desvio que foi gerado no sistema, ou seja, podemos atenuar a ação do controlador para que as variações não sejam abruptas como em um sistema do tipo **on-off**.

Instalando a placa Darwin

Nas páginas seguintes você vai aprender como fazer a instalação, configuração e primeiros testes da placa **CPU DARWIN**. Qualquer dúvida entre em contato com nossa central de atendimento através do nosso site **openrobotics.com.br.** Para receber dicas e novidades é importante que você faça a assinatura do nosso canal no Youtube. Com isso você sempre estará atualizado(a) pois receberá vídeos com novidades.

Sensores de entrada e saída

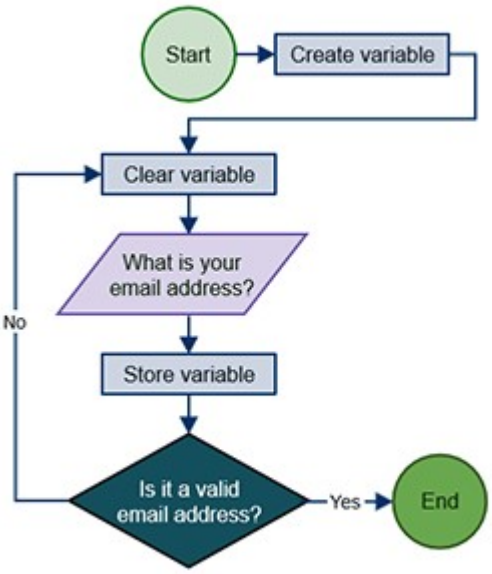
Em cada nova etapa deste tutorial você terá uma visão de algum dos sensores que acompanha o kit, ou seja, você vai aprender a lógica, pela prática , dos dispositivos de entrada e saída, os motores, e todos os outros hardwares que **compõem o seu kit**. É importante você perceber que o conjunto de sensores de entrada e saída, funcionando em conjunto com a parte mecânica, é que vai proporcionar o seu aprendizado. Neste tutorial você vai aprender como os sensores podem ser utilizados, e como programa-los para que façam o que você deseja.

O que é um programa de computador

Um programa de computador, ou software, é uma sequência de instruções que são enviadas para uma CPU (Unidade Central de Processamento). Cada tipo de microprocessador (cérebro) entende um conjunto diferente de instruções, ou seja, o seu próprio "idioma". Também chamamos esse idioma de linguagem de máquina.

As **linguagens de máquina** são, no fundo, as únicas linguagens que os computadores conseguem entender, só que elas são muito difíceis para os seres humanos entenderem. É por isso que nós usamos uma coisa chamada linguagem de programação.

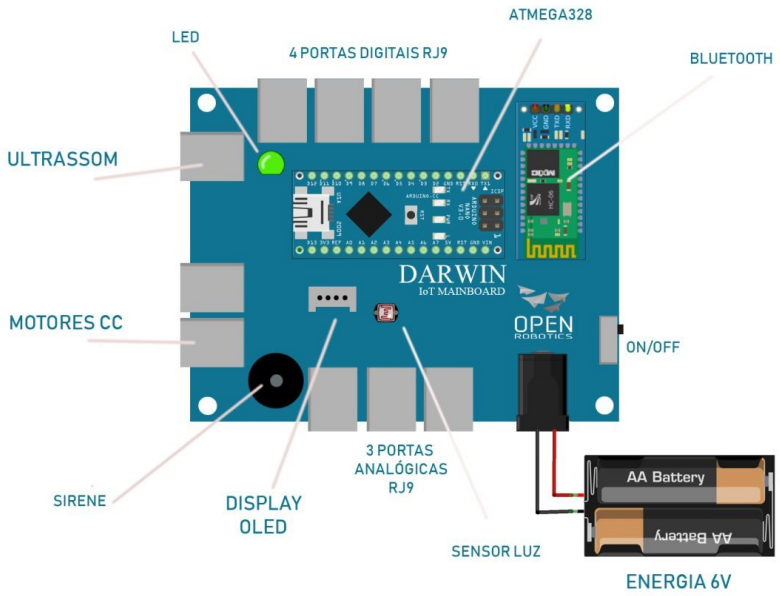
Um algoritmo, ou simplesmente programa, é um jeito de dizer para um computador o que ele deve fazer, de uma forma que nós humanos conseguimos entender facilmente. Os algoritmos normalmente são escritos em linguagens de programação de alto nível. Isso se aplica a praticamente qualquer computador, inclusive ao seu **KIT INVENTOR**.



Conhecendo a CPU DARWIN

Vamos agora conhecer melhor a nossa placa principal denominada **CPU DARWIN**, responsável por realizar todo o controle dos dispositivos que acompanham o seu kit.

O processador que acompanha essa placa chama-se **DARWIN**, o qual pode realizar dezenas de operações lógicas e matemáticas, além de ativar dispositivos de saída e receber informações do ambiente através de sensores. Este processador é chamado de **Arduino**, projetado na Itália em 2005 com o propósito de ser uma plataforma de ensino e aprendizagem de eletrônica, informática e robótica.



Nos conectores digitais você vai ligar a **SIRENES**, os **LEDS, e os SENSORES DE TOQUE**. O sensor de **distância** por **ultrassom** possui um conector próprio chamado **DIST**. Ao lado dele temos os conectores para os **Motores CC** denominados **M1 E M2.** Um pouco mais a direita da placa temos os conectores de Entrada de Energia. Perceba que a conexão Bluetooth já se encontra na placa. Abaixo da placa temos os leitores dos sensores de **Luz, som, cor, inclinação**, e são conhecidos como conectores **analógicos**. Nas portas digitais **D3** e **D2** podemos ligar os **servo motores**. Na placa temos ainda um LED, uma Sirene, um sensor de Luz (LDR), uma saída para display OLED (Não acompanha o kit) e a chave de liga e desliga.

A placa **DARWIN** trabalha com energia de 5 a 6 Volts conectados via USB ou de 9V conectados na entrada lateral da placa. **NUNCA** conecte 9 V diretamente na porta USB pois isso pode queimar o seu Kit.

Quando você adquire o **INVENTOR** ele já vem equipado com 4 pilhas AA do tipo Alcalinas, o que proporciona uma maior corrente de energia, melhorando o funcionamento. Porém, a maior parte do tempo você deverá utilizar a energia disponível via USB do seu computador, evitando gastar a energia das pilhas. Em geral os sensores gastam muita energia, então talvez o melhor seja posteri-

Instalando a placa DARWIN

Chegou a hora de preparar a sua placa para funcionamento. Veremos passo a passo todo o processo de instalação e configuração do seu computador para que você possa programar e enviar os comandos para seu kit INVENTOR.

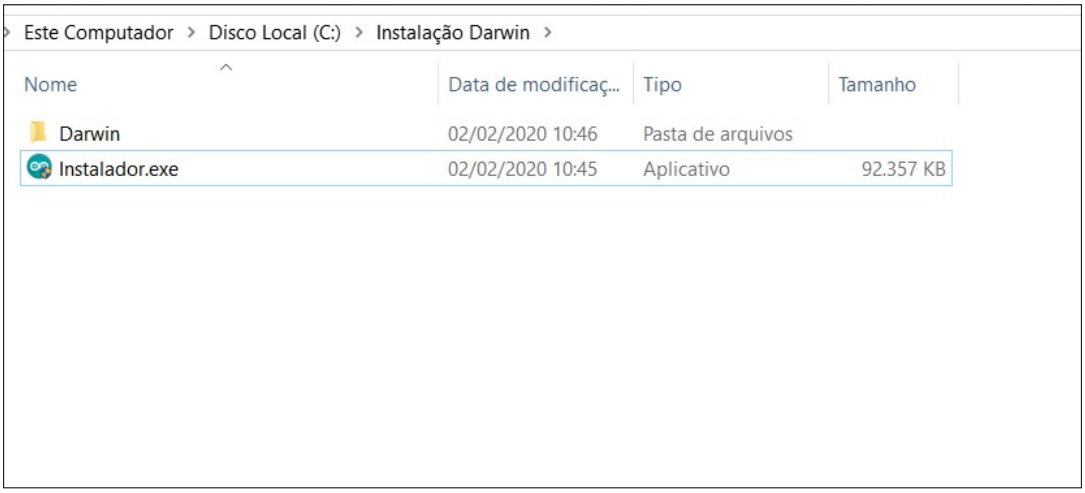
Passo 1: Download dos programas

Para baixar os programas de instalação entre no site **www.openrobotics.com.br.**

1. **Clique no link Downloads**
2. Vá até a opção **Ambiente de Programação Avançada** e clique **em Download**

O arquivo que será baixado está compactado. Dentro dele temos o programa instalador e uma pasta com uma biblioteca de recursos necessários ao correto funcionamento do seu INVENTOR.

Após baixar este arquivo, descompacte em algum lugar apropriado no seu computador. Veja como deverá ficar:

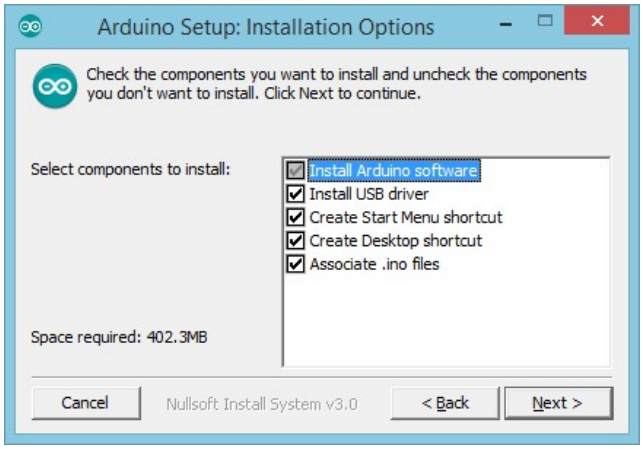


Passo 2: Disparando a instalação

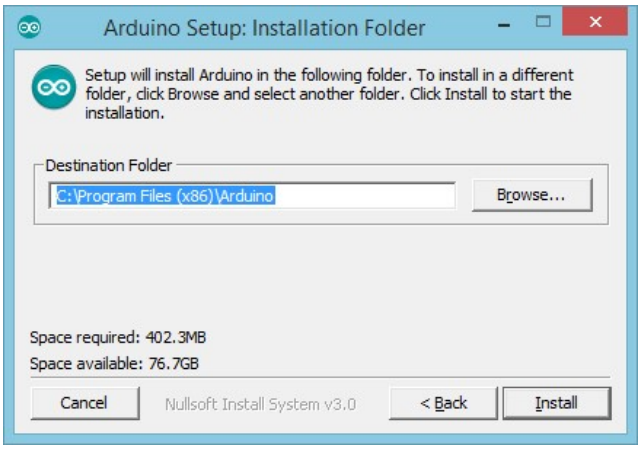
Agora que você já possui uma pasta com o programa de instalação, basta executar o **Instalador**, clicando 2 vezes sobre ele.

Você terá uma tela como mostrada ao lado, basta clicar na opção **I Agree**.

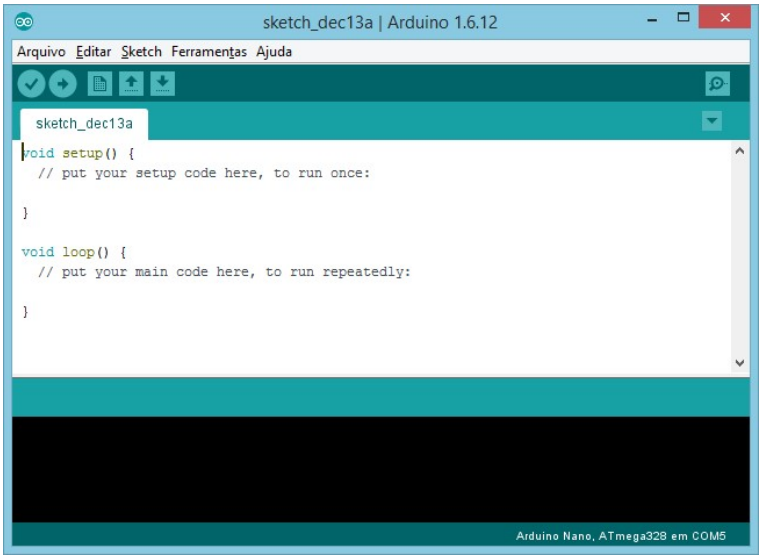
Na tela seguinte você pode clicar em NEXT para aceitar todas as opções de instalação.



Na sequencia vai aparecer uma tela com a opção de pasta onde você deseja instalar os programas do seu kit. Você pode deixar a opção padrão ou alterar conforme a sua necessidade. Clique no botão **Install** e aguarde o término da instalação.

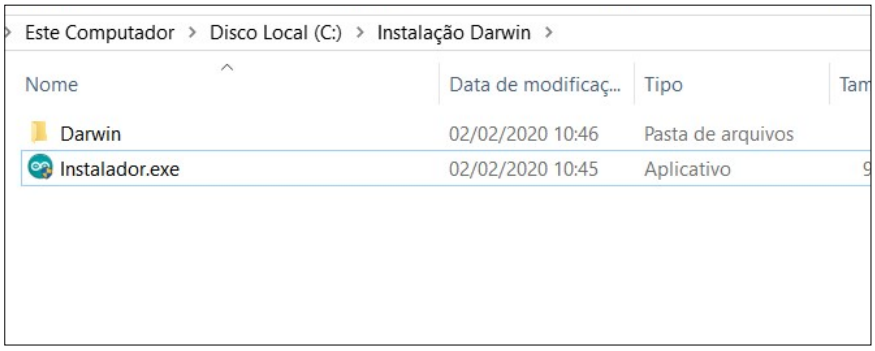


Ao término da instalação um ícone chamado ARDUINO será colocado em sua área de trabalho. Clicando duas vezes sobre ele você já consegue abrir o ambiente de programação.



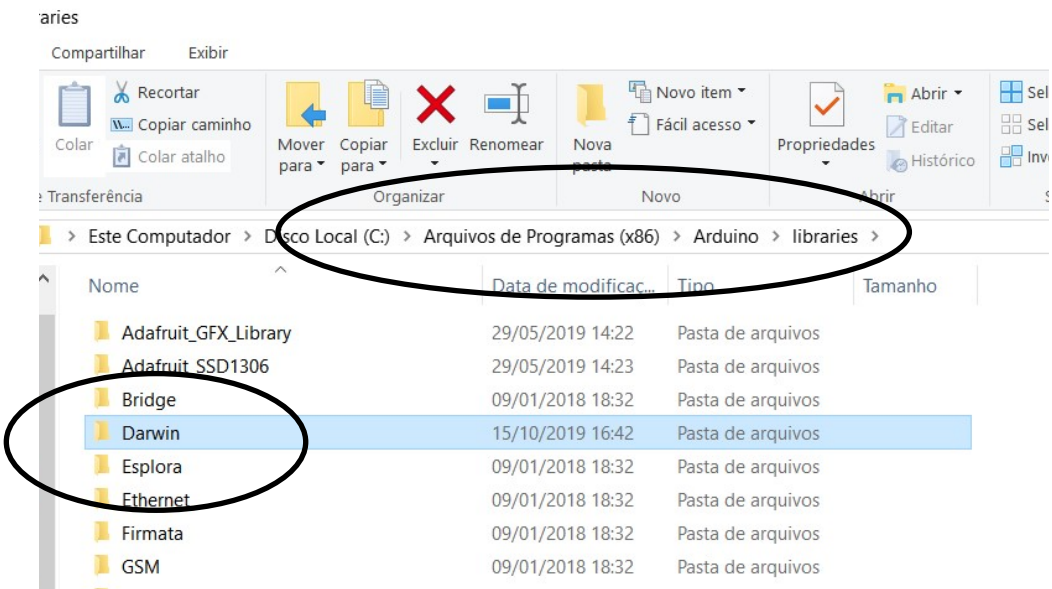
Passo 3: Copiando a biblioteca do Inventor

Agora que a instalação já está pronta, falta pouco para ficar tudo resolvido e você iniciar os seus primeiros testes. No local onde você salvou os seus arquivos tem uma pasta chamada **Darwin**. Ela contém a **biblioteca de comandos** necessários ao correto funcionamento do kit.



Faça o seguinte:

- Clique com o botão direito na pasta **Darwin** e escolha a opção **Copiar**.
- Vá até a pasta **Libraries** que está no local onde você instalou o Arduino, por exemplo: **C:\Program Files (x86)\Arduino\libraries**
- Cole a pasta **Darwin** dentro da pasta **Libraries**, para que fique como abaixo:

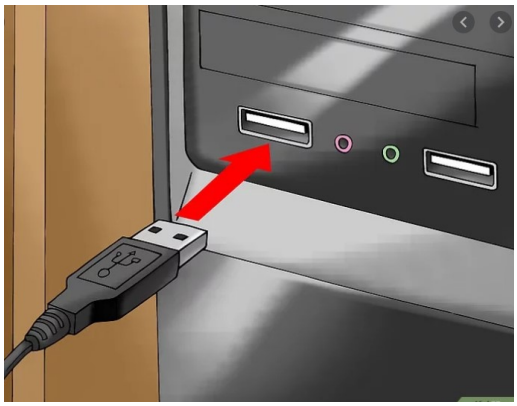


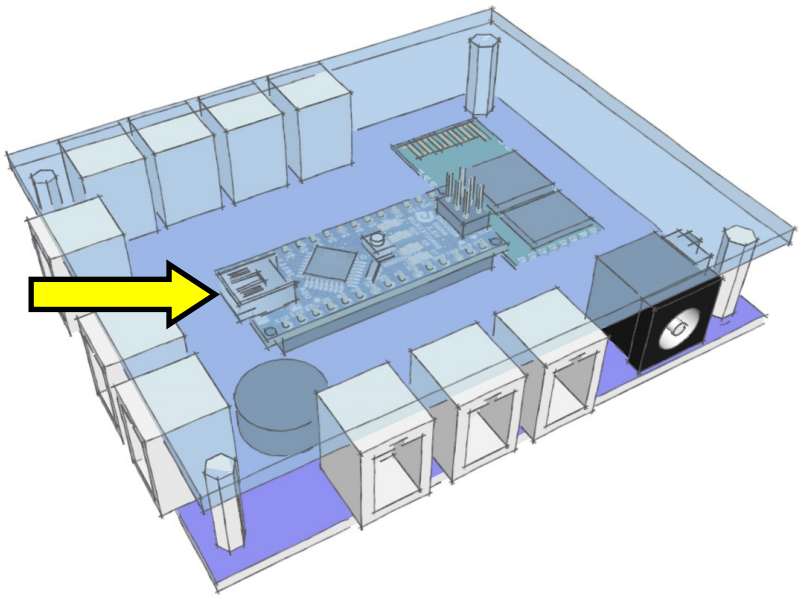
Resolvido! Agora já podemos rodar o nosso primeiro teste.

Passo 4: Conectando a CPU no computador.

Já estamos terminando. Agora vamos ligar o nosso computador na **CPU** e fazer o primeiro teste de programação. É muito importante que tudo dê certo neste momento, portanto, caso algo dê errado refaça os **procedimentos** ou acesse nossa central de atendimento através do nosso site.

· Ligue o cabo USB diretamente na sua placa como mostrado abaixo



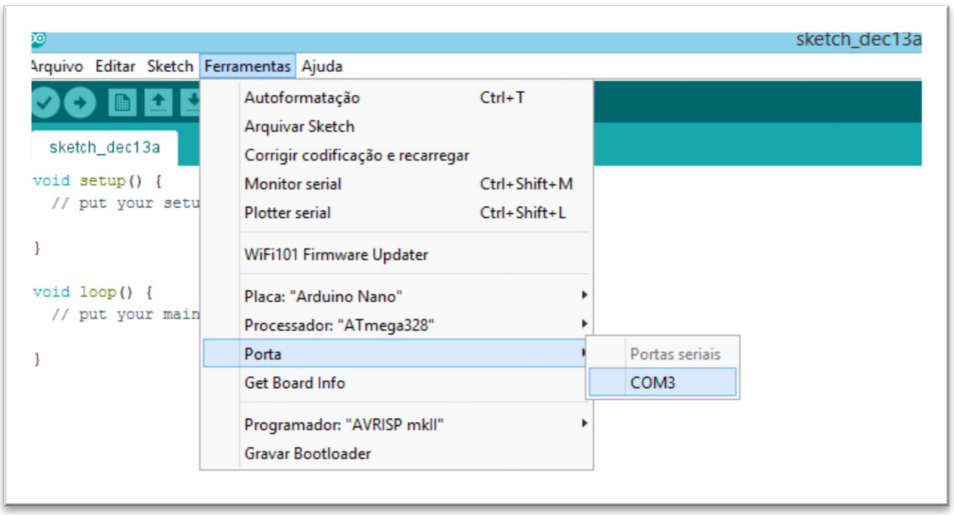


Agora vamos configurar o **Arduino**. Clique no menu **Ferramentas** e deixe tudo como mostrado abaixo:

* Placa: **Arduino Nano**

* Processador: **ATMega328**

* Porta: **COM3** (Ou outra que aparecer)



ATENÇÃO 1: Se a porta não aparecer repita a instalação. Se permanecer não aparecendo solicite a ajuda de um técnico. Pode ser um problema de configuração no seu computador.

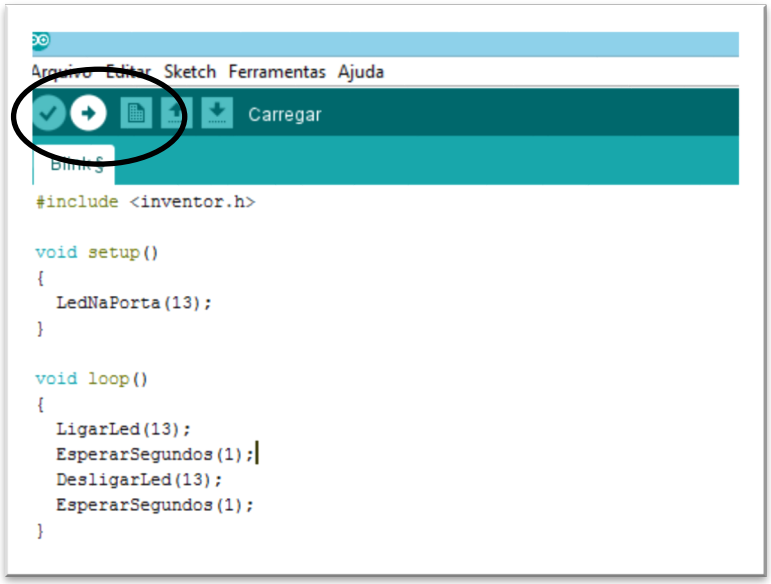
ATENÇÃO 2: A porta pode mudar de número quando você recolocar o cabo em uma próxima vez. Basta escolher a nova porta. As demais configuração não devem mudar.

Passo 5: Enviando seu primeiro programa

Agora vamos testar o envio de um primeiro programa. Vamos lá:

Digite o código abaixo no seu Arduino e salve-o com o nome de **TESTE1**. Atenção: as letras maiúsculas, minúsculas e **demais símbolos** devem ser digitados rigorosamente como mostrados abaixo.

```
#include <darwin.h>
void setup()
{
  LedNaPorta(12);
}
void loop()
{
  LigarLed(12);
  EsperarSegundos(1);
  DesligarLed(12);
  EsperarSegundos(1);
}
```



Após digitar o programa acima, envie para a CPU clicando no botão **Carregar**.

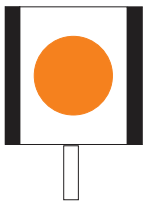
Se tudo der certo uma Luz irá piscar na CPU ligando e desligando com intervalo de 1 segundo. Caso algo dê errado, revise seu programa para conferir se tudo foi digitado corretamente. Um sinal de que tudo deu certo é uma mensagem em branco que aparece logo abaixo na tela do **Arduino**.

Nesta mesma tela pode aparecer a mensagem de erro alertando sobre um problema que pode ter acontecido. A primeira vez não é tão fácil, mas com o tempo você vai se acostumar a programar e enviar para a CPU.

Conhecendo os sensores

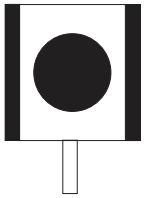
Os sensores compõem uma parte importante de qualquer robô. São eles os responsáveis em permitir que o robô receba dados do ambiente. É possível termos sensores para diversas situações, por exemplo, sensores de cor, de temperatura, de humidade, de distância, de ultrassom, e diversos outros.

Vejamos pelas imagens abaixo quais os sensores você está recebendo no seu INVENTOR. Atenção: a quantidade e o tipo de sensor podem variar de uma versão para outra.



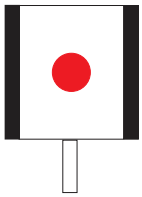
Sensor de Luz

Possui uma abertura frontal que permite a passagem de luz. Ele percebe se o ambiente está Claro ou Escuro. Você pode criar um robô que quando entra em algum local escuro, então ele acende uma lanterna.



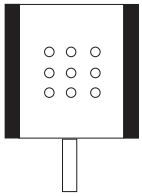
Sensor de Toque

Este sensor possui um interruptor chamado push-botton. Ou seja, quando o pressionamos, ele emite um sinal de positivo para o robô.



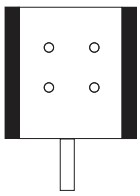
Lâmpada de Led

No seu INVENTOR você recebeu 3 lâmpadas de LED (Vermelho, verde, amarelo). Elas podem ser ligadas para sinalizar alguma situação importante, ou avisar sobre uma tomada de decisão pelo robô,



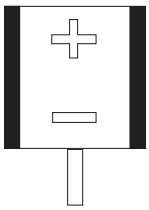
Sensor de som

Este sensor possui 9 furos frontais por onde pode entrar qualquer tipo de ruído. Ele pode ser utilizado para que o robô seja capaz de escutar o ambiente.



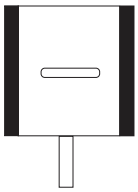
Sensor de hall

Este sensor possui 4 furos frontais por onde ele pode sentir a presença de campos magnéticos. Com ele você pode detectar materiais magnéticos como imãs.



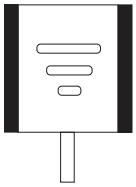
Energia 6v

O seu kit já vem com 4 pilhas totalizando 6 volts de energia. Você pode utilizar o Cabo USB diretamente no computador, ou a energia em separado. Nos dois casos o funcionamento da CPU permanecerá a mesma.



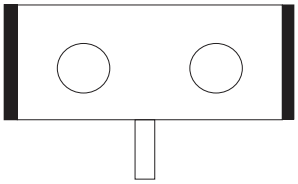
Sensor de inclinação

Este é um sensor muito interessante. Com ele podemos verificar se o robô está na vertical ou na horizontal. Isso pode ser muito útil quando você tiver que saber “como” o robô se encontra em um determinado momento. Pode ser que ele esteja de cabeça para baixo por exemplo.



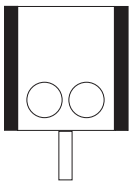
Sirene

Assim como os LEDs a sirene é um dispositivo de saída. Ou seja, uma sirene pode ser utilizada quando você precisar emitir algum sinal sonoro no ambiente.



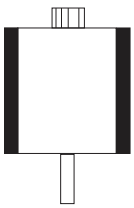
Sensor de distância por ultrassom

Utilizado para medir a distância de objetos, podendo variar de 1cm até 2m de distância. Você pode utilizar este sensor para criar veículos que se movimentam em um ambiente sem tocar nos objetos.



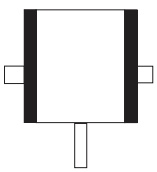
Sensor de infravermelho

Este sensor pode ser utilizado para verificar a presença de obstáculos, ou também para diferencias cores (preto ou branco) de objetos que estejam a aproximadamente 3cm de distância. Este sensor também pode ser utilizado para criar robôs que se deslocam por ambiente sem tocar nos objetos.



Servo motor

Este motor pode fazer giros precisos entre 0 e 180 graus. Geralmente são utilizados em articulações de robôs e em mecanismos que exigam precisam nos movimentos. Na sua placa você vai encontrar dois motores como este.



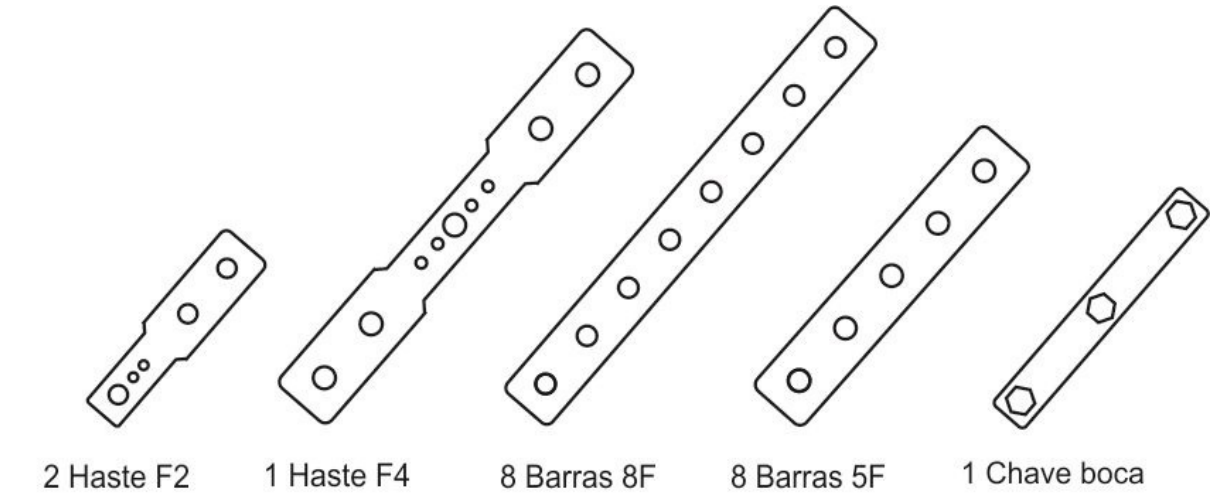
Motor CC

Este motor é o responsável por fazer as rodas de borracha funcionarem. Ele consome muita energia e como dito acima, você deverá liga-lo diretamente na ponte H.

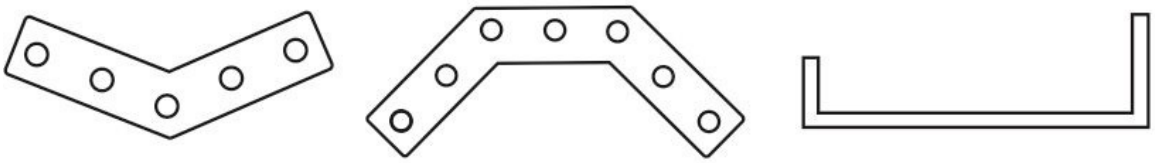
Conhecendo os conectores

Seu Kit INVENTOR possui várias peças conectores as quais podem ser utilizadas para dezenas de montagens diferentes. Essas peças foram projetadas para facilitar o encaixe uma nas outras e também nos sensores. Eles são feitos de um material conhecido como **polimetil-metacrilato** (**PMMA**) um material termoplástico rígido, transparente, colorido ou incolor; também pode ser considerado um dos polímeros (fibra sintética) mais modernos e com maior qualidade do mercado, por sua facilidade de adquirir formas, por sua leveza e alta resistência.

No kit INVENTOR ele possui várias formas e foram criados utilizando ferramenta de corte a lazer, portanto, todas as peças possuem um plástico de proteção durante o corte. Você pode manter esse plástico, mas também pode retira-lo durante o uso. **Você vai perceber que algumas peças estão escuras, aparentemente sujas, mas isso se deve ao calor do raio lazer que realizou o corte, portanto, basta remover este plástico e sua peça estará pronta para uso**. Vejamos quais são elas.



2 Haste F2 | 1 Haste F4 | 8 Barras 8F | 8 Barras 5F | 1 Chave boca



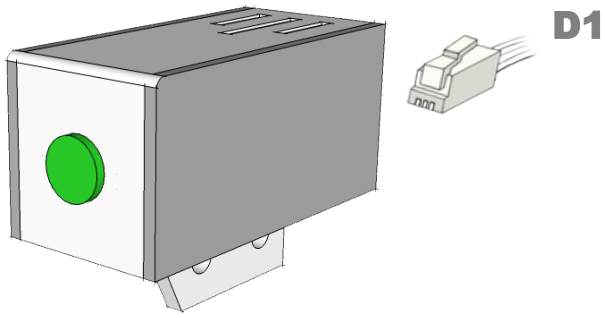
6 peças V 5F | 6 peças Cone 7F | 1 Chassi para veículo

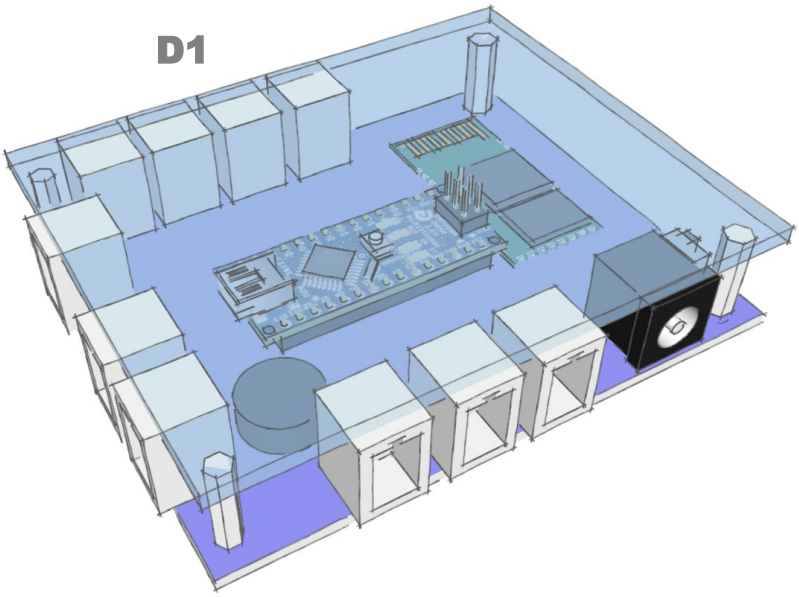
6 peças L 4F | 6 peças U 3F | 6 peças L 2F | 6 peças V 2F

Acionando lâmpadas de LED

As lâmpadas podem existir com diversos formatos e em **diversos tipos e funcionamento**. Uma das mais utilizadas hoje tanto pela **economia** quanto pela sua utilidade são as **lâmpadas de LED**. Vejamos como podemos fazer uma delas funcionar.

Fazendo um LED piscar





Programação:

```
#include <darwin.h>
void setup()
{
  //Informando que vamos ter um LED na porta Digital D1
  LedNaPorta(D1);
}
void loop()
{
  LigarLed(D1);
  EsperarSegundos(1);
  DesligarLed(D1);
  EsperarSegundos(1);
}
```

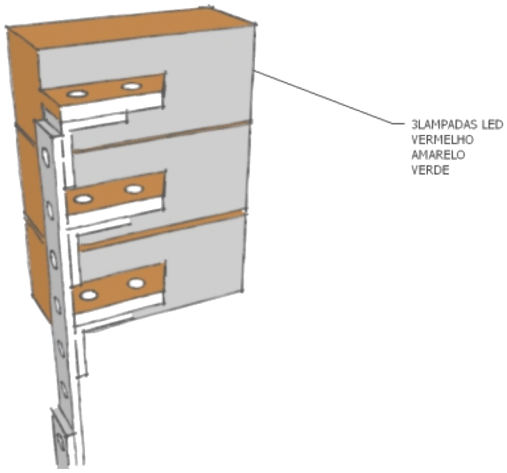
Desafios desta aula:

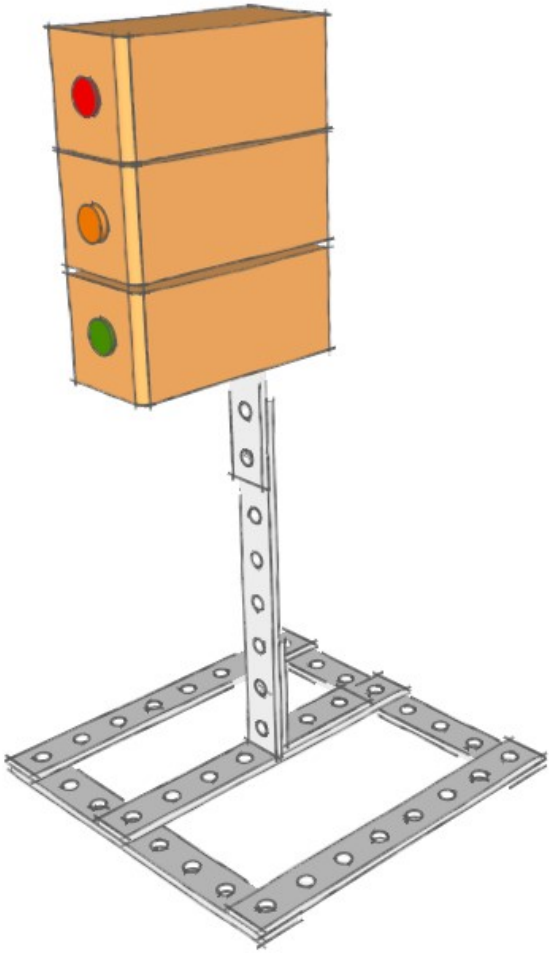
1. Coloque duas lâmpadas e faça elas piscarem alternadamente.
2. Faça com que a primeira lâmpada pisque duas vezes e a outra, de forma alternada, pisque uma única vez.

Montando nosso próprio semáforo

Vamos agora montar um semáforo (também conhecido popularmente como sinal, sinaleira, farol ou sinal luminoso) um instrumento utilizado para controlar o tráfego de veículos e de pedestres nas grandes cidades em quase todo o mundo. Utiliza uma linguagem simples e por isso de fácil assimilação. É composto geralmente por três círculos de luzes coloridas.

Geralmente o semáforo é controlado por um computador igual a este que você vai utilizar na sua montagem. O semáforo vai deixar os veículos passarem por um determinado tempo, e ficarem parados por um outro período de tempo. Tudo depende do fluxo desejado, seja de pedestres, seja de veículos.





Você vai precisar dos seguintes materiais: 1 LED vermelho, 1 LED amarelo, 1 LED verde, 7 barras de 8 furos, 1 peça L com 4 furos, 3 peças L com 2 furos, Parafusos e porcas.

Para montar o semáforo faça o seguinte:

1. Comece montando a base com **5 barras longas de 8 furos**. No centro acrescente uma peça **L com 4 furos**, e nela fixe duas barras longas com 8 furos cada.
2. Utilize as peças em **L de 2 furos** para fixar cada um dos LEDs no poste que sustenta o semáforo.
3. Os 3 LEDs deverão ser ligados na porta **D3, D1 e D2** respectivamente (**vermelho**, **amarelo**, **verde**).

Construindo o algoritmo do semáforo

Após a realização da **montagem** vamos programar a nossa estrutura e **enviar o código** para a CPU do **Inventor**. Para isto você vai utilizar o código do **Arduíno**.

1. Antes de digitar e enviar o programa, ligue **o LED vermelho na porta D3, o LED amarelo na porta D1 e o LED verde na porta D2**.
2. Na área de trabalho do seu computador dê dois cliques no ícone do **Arduino**.
3. Então crie o programa mostrado abaixo com a ajuda do professor.

Programação :

```
#include <darwin.h>
void setup() {
  //Adicionando os LEDS em cada porta
  LedNaPorta(D3);
  LedNaPorta(D1);
  LedNaPorta(D2);
}
void loop()
{
  LigarLed(D3);
  EsperarSegundos(8);
  DesligarLed(D3);

  LigarLed(D2);
  EsperarSegundos(5);
  DesligarLed(D2);

  LigarLed(D1);
  EsperarSegundos(2);
  DesligarLed(D1);
  }
```

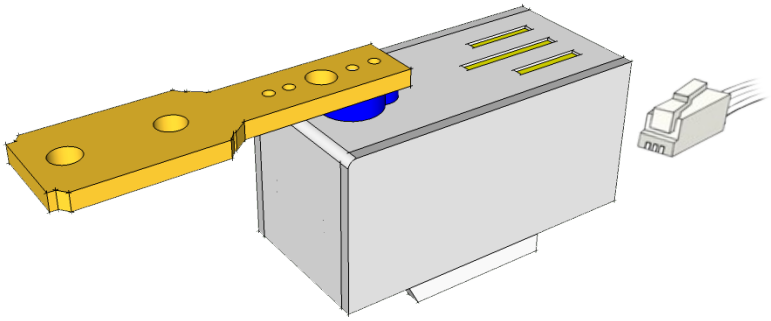
Entendendo o que o programa faz:

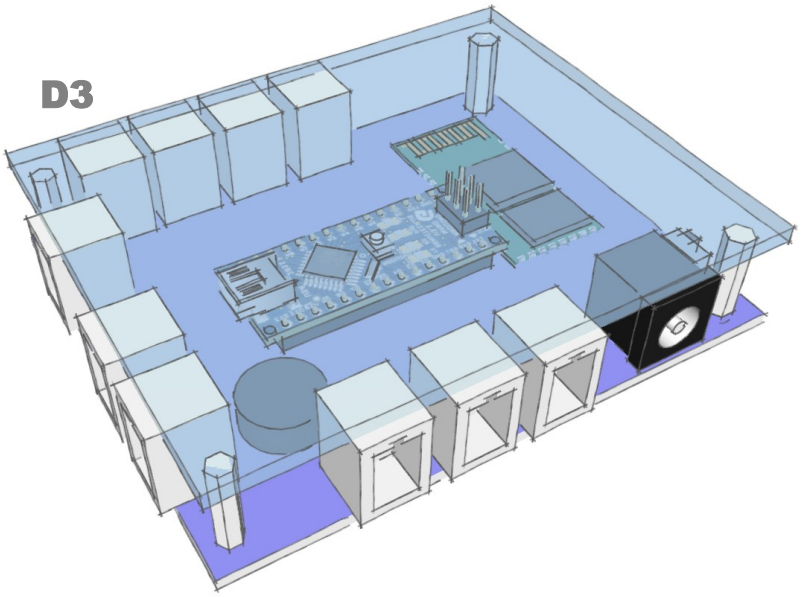
Nosso programa vai fazer com que o LED vermelho fique aceso por **8 segundos**, em seguida ele se apaga. Depois o LED verde vai ficar aceso por **5 segundos**, e se apaga. Ao final o LED amarelo fica aceso por **2 segundos** e também se apaga. Esta rotina vai ficar em execução até que a placa seja desligada, ou seja, vai ficar em **loop** enquanto houver energia. Experimente mudar estes tempos e veja os resultados.

Controlando servo motores

Os servo motores são dispositivos que podemos **girar** em um ângulo que varia entre **0 e 180 graus**. O professor irá lhe demonstrar como ele funciona. Na sequência faça **a montagem abaixo.**

Fazendo um servo girar





Programação:

```
#include <darwin.h>


void setup() {
  ServoNaPorta(D3);
}
void loop() {
   MoverServo(D3, 0);
   EsperarSegundos(2);
   MoverServo(D3, 180);
   EsperarSegundos(2);
}
```

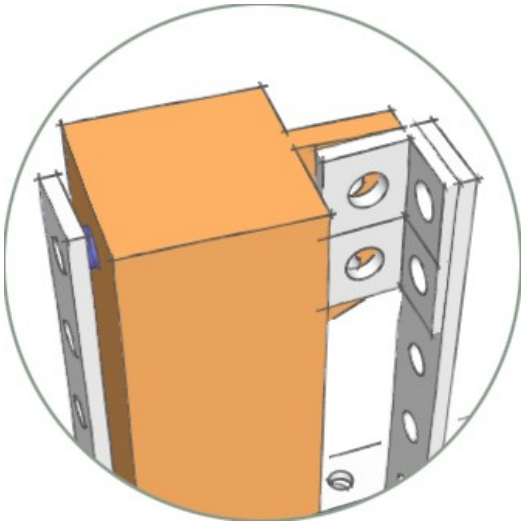
Desafios desta aula:

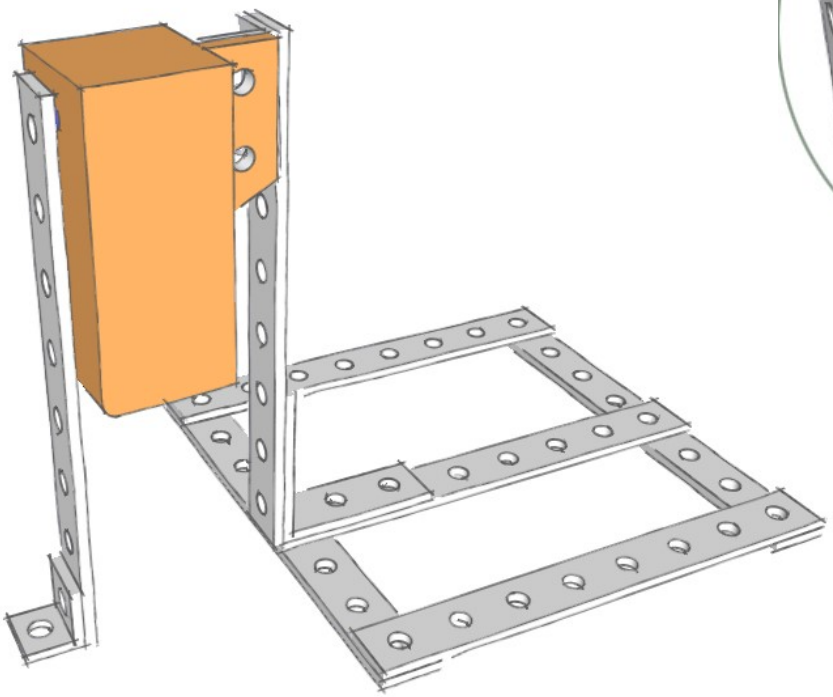
1. Ligue 2 leds de cores diferentes nas portas D1 e D2. Agora faça com que o LED em D1 fique aceso quando o servo estiver na posição 0 graus, e o LED em D2 fique aceso quando o servo estiver na posição 180 graus.

Montando nosso robô chutador

Vamos agora montar um robô que chuta a bola ao gol. Vamos utilizar uma **base** que dará **sustentação** ao nosso robô. Para completar essa montagem, você pode também fazer um gol, que deverá ser acertado pelo jogador.

Você vai precisar dos seguintes materiais: 1 servo





motor, 1 botão de push, 1 placa Darwin, 7 barras 8 furos, 2 conectores L com 2 furos, 1 haste para o servo motor, 1 conector L 4 furos, 1 bolinha de ping-pong, parafusos e porcas.

Para montar o robô chutador faça o seguinte:

1. Comece montando a **base com 5 peças de 8 furos**. Na peça central, **coloque um L** para conectar uma outra **peça com 8 furos**.
2. No topo desta peça faça a instalação do **servo motor** utilizando uma **peça L com 2 furos**.
3. Conecte **uma haste** no servo motor utilizando o parafuso para servo (**micro parafuso**).
4. Ligue o **botão de push na porta D2** e o **servo motor na porta D3**. O servo fará a movimentação da perna do robô para chutar a bolinha de ping-pong.

Programando o robô chutador

Após a realização da **montagem** vamos programar a nossa estrutura e **enviar o código** para a CPU do **Inventor**. Para isto você vai utilizar o código do **Arduíno**.

1. Na área de trabalho do seu computador dê dois cliques no ícone do **Arduino**,
2. Então crie o programa mostrado abaixo:

Programação:

```
#include <darwin.h>
int ValorDoBotao;
void setup() {
  ServoNaPorta(D3);
  BotaoNaPorta(D2);

}
void loop() {
  MoverServo(D3, 90);
  ValorDoBotao = LerBotao(D2);
  if(ValorDoBotao == Sim)
  {
    MoverServo(D3, 150);
    EsperarSegundos(1);

    MoverServo(D3, 90);
    EsperarSegundos(1);
  }
}
```

Entendendo o que o programa faz:

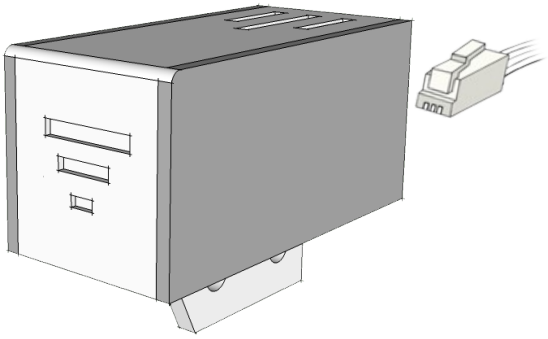
Na primeira parte criamos a **variável ValorDoBotao para armazenar a situação em que se encontra o botão** (apertado ou não), além de iniciar o **servo motor na posição 90 graus**.

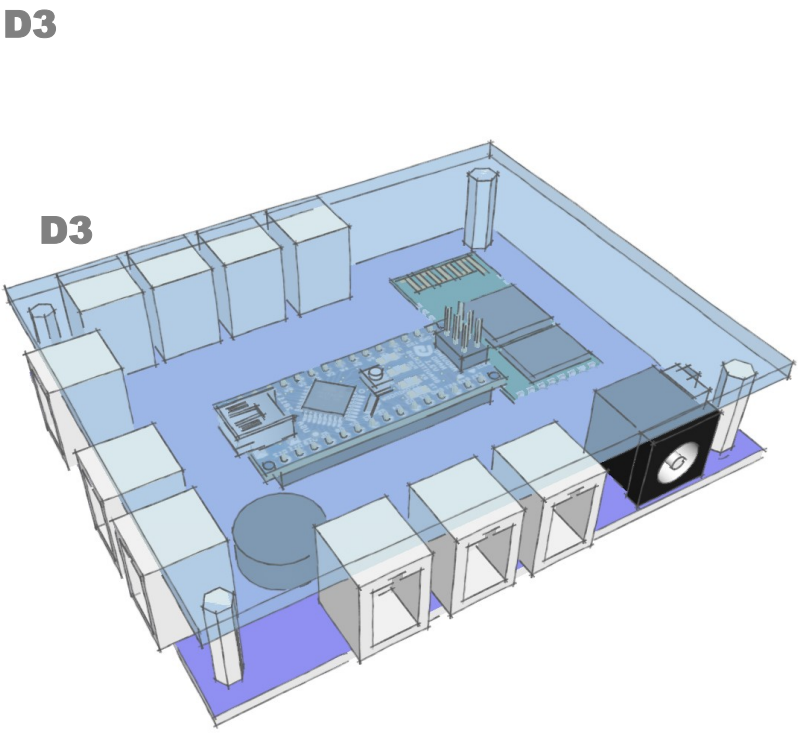
Na sequência existe um **teste para verificar se o botão** está pressionado, **em caso afirmativo (Sim) então o servo que está na porta D3 da placa irá se movimentar**. Veja que talvez você tenha que alterar os ângulos, dependendo de como ficou a posição do servo que você instalou na placa. Não se esqueça que o professor está aí para te ajudar.

Emitindo sons

O uso de **sirenes** pode se tornar útil quando precisamos emitir algum **sinal sonoro**. Utilizando uma sirene (também conhecida como **Buzzer**) podemos emitir um alerta ou reproduzir diversos tipos de **sons**.

Fazendo a sirene funcionar





Programação:

```
#include <darwin.h>
void setup() {
  //Informando que a sirene está em D3
  SireneNaPorta(D3);
}
void loop()
{
  LigarSirene(D3);
  EsperarSegundos(2);
  DesligarSirene(D3);
  EsperarSegundos(2);
}
```

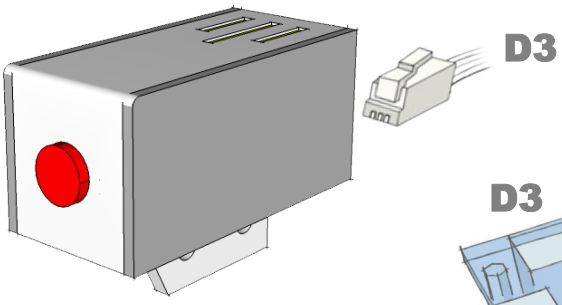
Desafios desta aula:

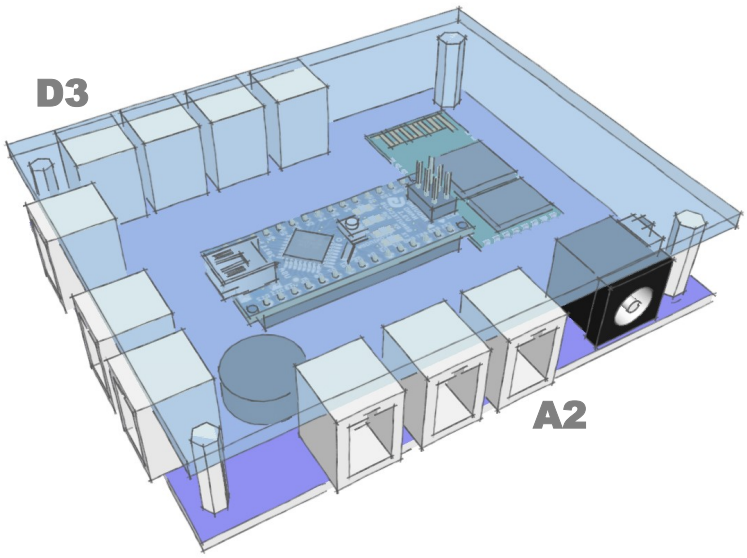
1. Instale um LED na porta D2 e faça ele ligar quando a sirene estiver desligada.
2. Faça o LED e a sirene tocarem juntos por um intervalo de 1 segundo, sem parar.

Sensores de LUZ ambiente

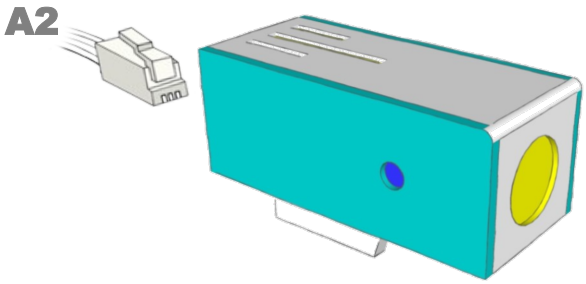
Agora vamos ver como podemos medir a **intensidade da luz** no ambiente onde estamos. Para isso vamos utilizar um **sensor de luz**, também chamado de **LDR (Light Dependent Resistor)**. Com ele vamos ativar um **LED** quando estiver escuro e apagaremos o LED quando estiver claro, **simulando** o que acontece com o poste de luz nas ruas.

Acionando o LED pelo LDR





Programação:




```
#include <darwin.h>
void setup() {
  //Informando que o LED está na porta D3
  LedNaPorta(D3);
}
void loop() {
 if ( LerLuz(A2) == Escuro )
     LigarLed(D3);
 else
     DesligarLed(D3);
}
```

Desafios desta aula:

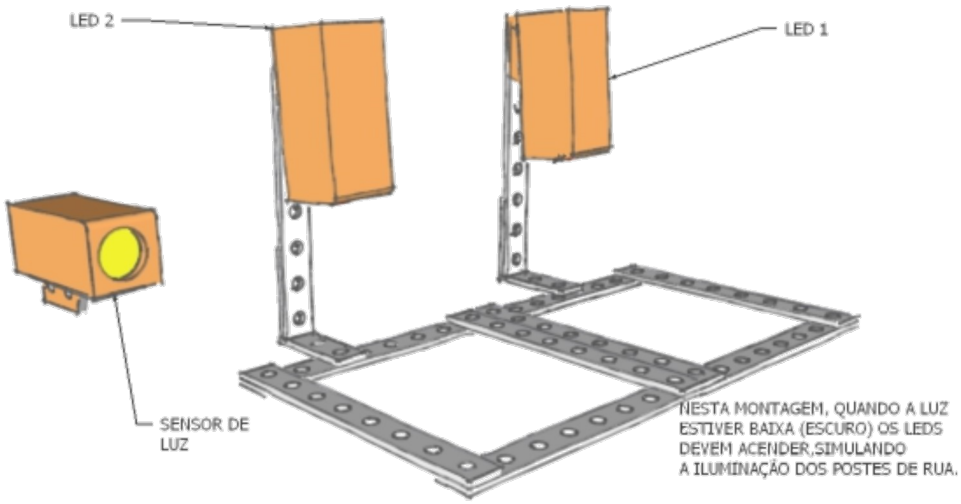
1. Instale mais um LED na porta D1. Faça com que ele acenda quando o ambiente estiver iluminado.
2. Utilizando um servo na porta D2, faça com que ele gire para a posição 0 graus quando estiver escuro e 180 graus quando estiver claro.

Construindo um sistema eficiente de iluminação pública

Uma das formas que os **habitantes do mundo** podem contribuir é justamente economizando energia, uma vez que para produzir energia elétrica, de uma maneira geral, **poluímos muito o meio em que vivemos**.

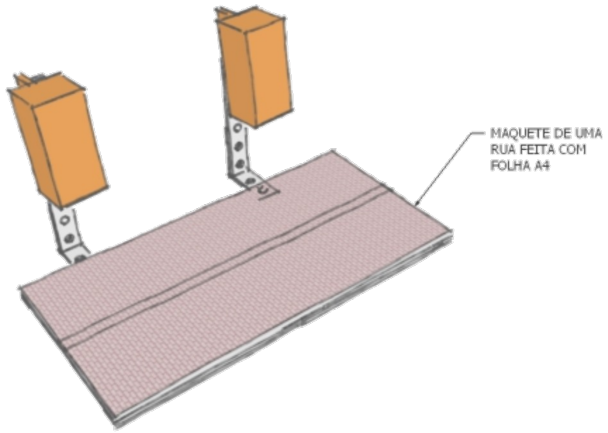
Podemos, por exemplo, **utilizar sensores** de luz em vias públicas, que farão com que as luzes dos postes se acendam apenas quando a noite chegar, ou quando o tempo estiver muito fechado durante o dia.

Vamos agora ver uma montagem simulando o sistema de iluminação pública, em que as luzes se acendem ao escurecer.



Você vai precisar dos seguintes materiais:

1 sensor de Luz, 2 lâmpadas LED, 10 barras 8 furos, 2 conectores L 4 furos, 2 conectores L 2 furos, placa Darwin, papelão ou papel A4.



Para montar o sistema de iluminação faça o seguinte:

1. Comece montando a **base com 8 peças de 8 furos**. Na peça central, coloque **duas L 4 furos**, para conectar duas outras peças com 8 furos.
2. Instale as **2 peças L4 em cada uma das bases** que você criou. Para os **postes**, utilize as outras **2 barras L 8 furos** restantes, e no alto delas instale os **2 leds utilizando as peças L 2**. Eles vão funcionar como as luminárias que estão no alto dos postes.
3. Instale o **LED 1 na porta D2** e o **LED2 na porta D3**. Instale o **sensor de luz na porta A1**.
4. Quando a incidência de luz no sensor estiver baixa (escuro) os LEDs devem acender. Você pode desenhar e forrar a base com o **papel A4 simulando uma avenida**.

Automatizando o sistema de iluminação

Após a realização da **montagem** vamos programar a nossa estrutura e **enviar o código** para a CPU do **Inventor**.

1. Na área de trabalho do seu computador dê dois cliques no ícone do **Arduino**.
2. Então crie o programa mostrado abaixo:

Programação:

```
#include <darwin.h>
int x;
void setup() {
  LedNaPorta(D2);
  LedNaPorta(D3);
}
void loop()
{
  x = LerLuz(A1);
  If ( x == Escuro )
  {
    LigarLed(D2);
    LigarLed(D3);
  }
  else
  {
    DesligarLed(D2);
    DesligarLed(D3);
  }
}
```

Entendendo o que o programa faz:

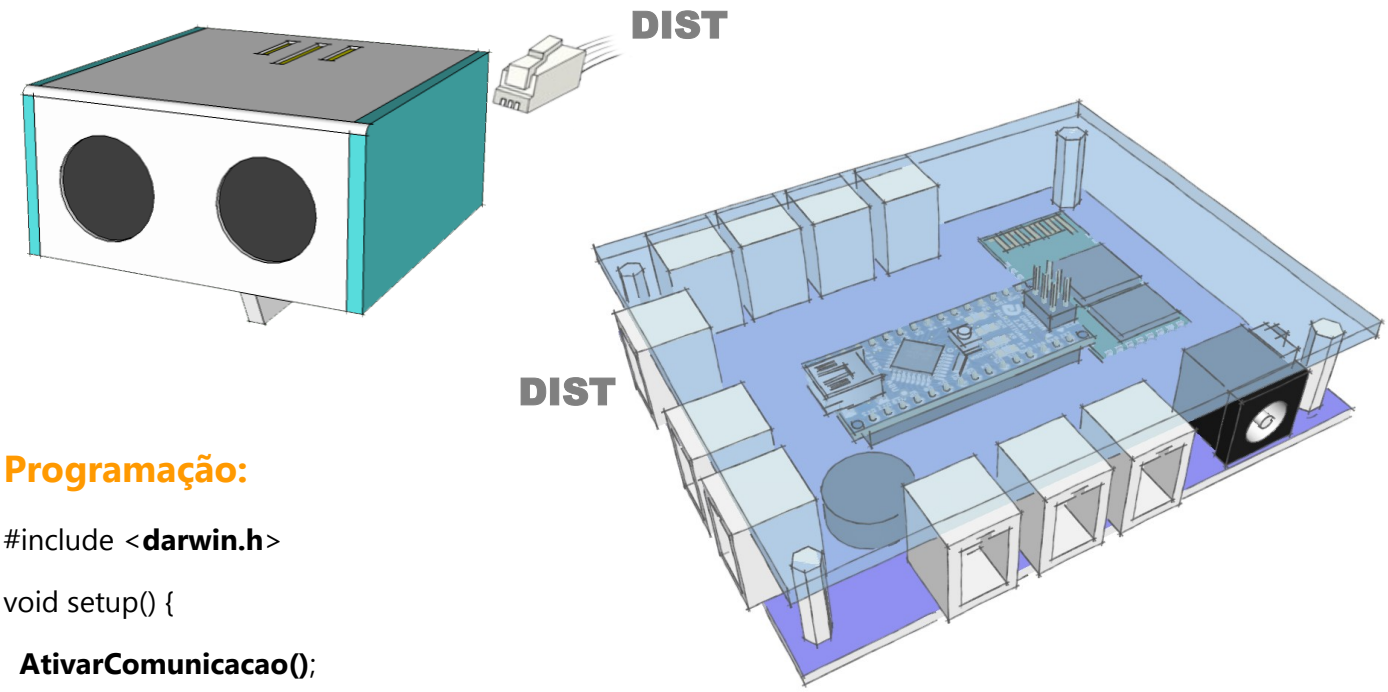
Após instalados os **LEDs nas portas D2 e D3**, fazemos a leitura da porta onde está o **sensor de Luz (A1)**, e colocamos esse valor na **variável x** que foi definida para isso.

SE a **luz** que entra pelo orifício do sensor estiver **fraca (Escuro),** então será dado o comando para **acender os LEDs**, caso contrário, ou seja, se a iluminação estiver **média ou alta (Claro)**, os **LEDs permanecerão apagados.**

Sensores de distância

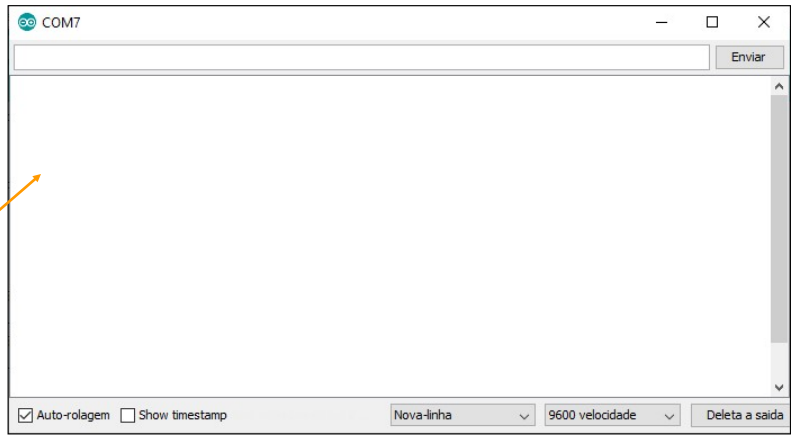
Alguns robôs possuem **sensores ultrassônicos** que são utilizados para medir a distância dos objetos. Este recurso torna-se muito útil quando ele precisa **desviar de obstáculos** ou seguir algum objeto até que o encontre em um **determinado ambiente**. Veremos agora como isso pode ser feito com o sensor de distância por ultrassom.

Medindo a distância em centímetros



Programação:

```
#include <darwin.h>
void setup() {
  AtivarComunicacao();
}
void loop() {
  EnviarParaComputador ( LerDistancia() ) ;
  EsperarSegundos(1);
}
```



Novidade na área:

Perceba que no programa acima utilizamos duas novas funções: **AtivarComunicacao()** e **EnviarParaComputador().** Com elas é possível receber e **monitorar o valor sendo lido por um sensor em tempo real**. Para isso, enquanto o programa estiver em execução, clique no **Ferramentas** do programa do Arduino e acione a opção **Monitor Serial**. Ao aproximar ou distanciar um objeto plano da frente do sensor, a distância será exibida **nesta área**. Experimente!

Desafios desta aula:

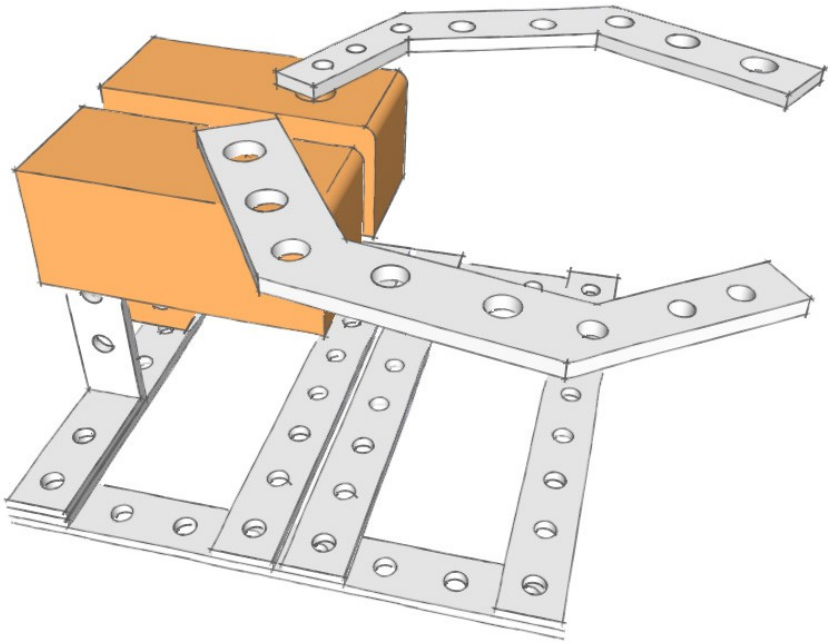
1. Instale um LED na porta D2. Faça com que ele acenda SE a distância for menor que 10 cm.
2. Instale a sirene na porta D1. Faça com que ela dê 1 apito quando a distância for menor que 10 cm; 2 apitos quando a distância estiver entre 11 e 20 cm e 3 apitos quando a distância estiver entre 21 e 30 cm.

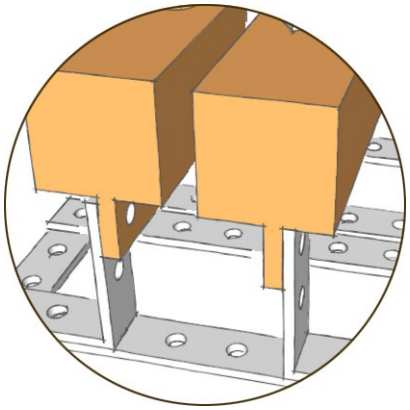
Montando uma garra robótica

Você já deve ter notado que formiga, escorpião, gafanhoto, alguns besouros e vários outros insetos possuem uma ou mais pinças em alguma parte do corpo.

A utilidade dessa garrinha para esses bichinhos deve ser enorme! Pense nas formigas cortando as folhas por exemplo...

Talvez por inspiração, alguns robôs são montados tendo como mãos as pinças, que alguns também chamam de garras. Vamos agora montar a garra de um robô.





Você vai precisar dos seguintes materiais:

1 sensor de ultrassom (distância), 2 servos motores, 2 conectores L 7 furos, 2 conectores L 4 furos, 6 barras longas B 8 furos, a placa Darwin, parafusos e porcas.

Para montar a garra robótica faça o seguinte:

1. Comece montando a **base com 6 barras longas de 8 furos**. Em uma das laterais acrescente **2 peças L com 4 furos** e então fixe **1 servo motor** em cada uma delas.
2. Nos eixos dos servos conecte **uma peça L de 7 furos** em cada. Depois coloque o sensor infravermelho em outra lateral da base.
3. Os 2 servos motores deverão ser ligados nas **portas D3 e D2**.
4. O sensor de distância deve ser ligado na **porta DIST.**

Fazendo a garra abrir e fechar

Após a realização da **montagem** vamos programar a nossa estrutura e **enviar o código** para a CPU do **Inventor**. Para isto você vai utilizar o código do **Arduíno**.

1. Na área de trabalho do seu computador dê dois cliques no ícone do **Arduino.**
2. Então crie o programa mostrado abaixo:

Programação:

```
#include <darwin.h>
int Distancia;
void setup()
{
 ServoNaPorta(D3);
 ServoNaPorta(D2);
}
void loop()
{
  Distancia = LerDistancia();
  if(Distancia < 20) {
    MoverServo(D3, 60);
    MoverServo(D2, 110);
    EsperarSegundos(5);
  }
  Else {
    MoverServo(D3, 180);
    MoverServo(D2, 0);
  }
}
```

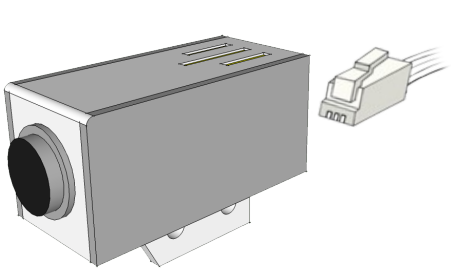
Entendendo o que o programa faz:

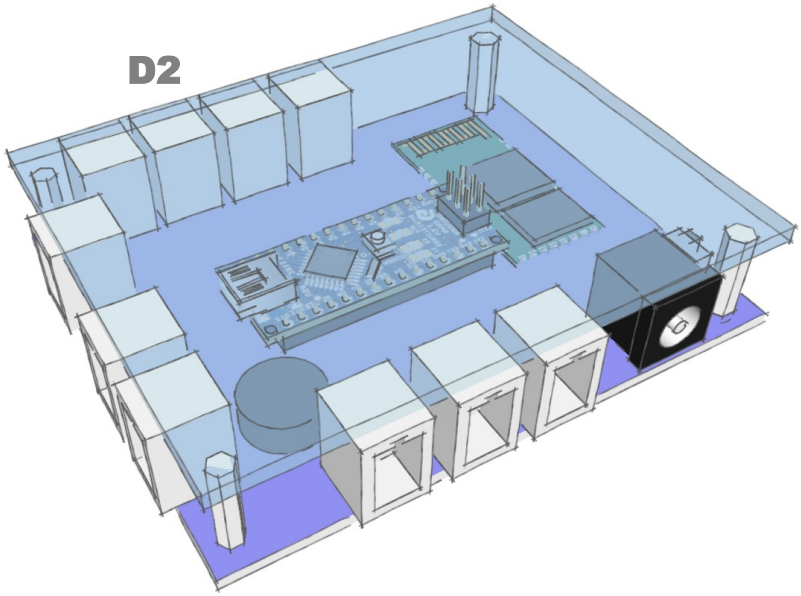
Esse programa faz com que as garras se fechem automaticamente toda vez que algo se aproximar do sensor de distância. No início pegamos as informações do sensor de distância conectado na **porta DIST** e passamos para a **variável Distancia**. Caso algo esteja próximo ao sensor de distância, os **servos motores (garras)** são posicionados em determinados ângulos para se fecharem. Então **aguardamos 5 segundos**, como se a garra estivesse segurando algo. Caso não tenha nada próximo, os servos motores são posicionados de maneira que as garras fiquem abertas.

Botão push button

Em diversas situações é importante contar com um **botão** para que uma pessoa possa apertar e disparar **alguma ação no robô**. Por exemplo, podemos utilizar um botão para enviar um sinal para que o robô possa rodar ou parar um motor; **ligar ou desligar um LED** e em dezenas de outras situações. Vejamos como isso pode ser feito.

Acionando a sirene da placa Darwin com o botão





Programação:

```
#include <darwin.h>
int x; //Criamos uma variável para
//ver a situação do botão
void setup() {
    BotaoNaPorta(D2);
    SireneNaPorta(D13);
}
void loop() {
  x = LerBotao(D2);
  if (x == Sim)
      LigarSirene(D13);
  else
      DesligarSirene(D13);
}
```

Desafios desta aula:

1. Agora use o LED da placa Darwin que está na **porta D12**. Caso o botão esteja pressionado esse LED deverá ser ligado.
2. Instale um servo motor na **porta D3**. Caso o botão seja pressionado, então o servo deverá girar para o ângulo 0 graus. Caso ele esteja solto, então o servo deverá girar até 180 graus.

Sensor de Infravermelho

Um sensor de **infravermelho** pode ser utilizado quando precisamos medir uma luz que não pode ser vista **pelo ser humano**. Os raios infravermelhos possuem diversos usos no nosso dia a dia, por exemplo, são utilizados em controle remotos de **aparelhos eletrônicos**.

Para esta aula você deverá possuir uma superfície preta e outra branca para testar o sensor de infravermelho, o qual deverá estar a uma distância de 1 cm destas superfícies.

Vejamos como funciona.

Acionando um LED pela leitura do infravermelho

Programação:

```
#include <darwin.h>
int x; //Criamos uma variável para ver a situação do infravermelho
void setup() {
   LedNaPorta(D1);
}
void loop() {
  x = LerInfra(A0);
  if (x == Preto)
     LigarLed(D1);
  else
     DesligarLed(D1);
}
```

Desafios desta aula:

1. Ligue um outro LED na porta D2 que deverá ficar aceso quando o Infravermelho estiver sobre a superfície branca.
2. Acione a sirene da placa Darwin quando o LED em D2 estiver ligado.

Montando uma cancela automática

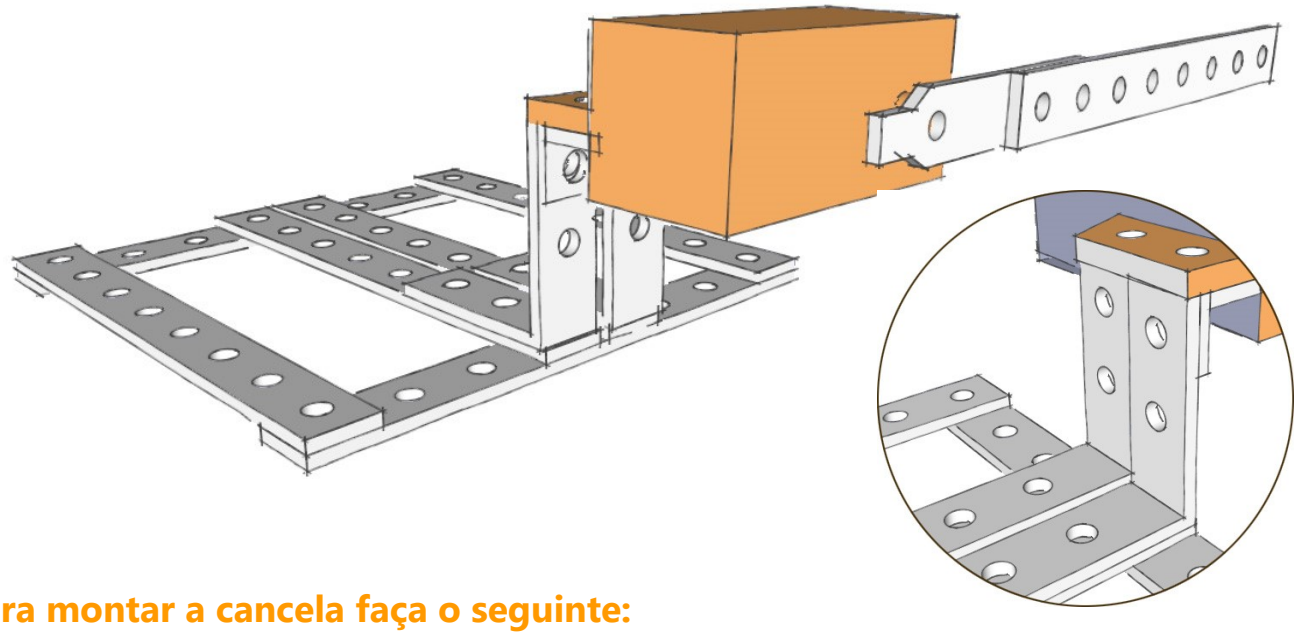
Há poucas décadas atrás não era assim, mas hoje **tem computador em tudo quanto é lugar que a gente vai**.

No escritório, na padaria, na lanchonete, no restaurante, na fábrica, na farmácia, no açougue, na igreja, no supermercado, nas lojas, nos bancos, na secretaria da escola e, algumas vezes, **até na sala de aula**.

Na cabine dos pedágios em rodovias também tem um computador para que o atendente possa registrar o pagamento e dar o **comando para levantar a cancela**. Será que o **computador levanta automaticamente a cancela** após a confirmação do pagamento?!

A seguir vamos montar e programar uma cancela automática para que você tenha a experiência com esse mecanismo. Acompanhe as orientações abaixo:

Você vai precisar dos seguintes materiais: 7 barras 8 furos, 1 servo motor, 1 haste do servo motor, 2 conectores L 4 furos, 2 conectores L 2 furos, parafusos e porcas.



Para montar a cancela faça o seguinte:

1. Comece montando a **base com 6 peças de 8 furos**. Nas peças centrais, coloque os **conectores L 4 furos** em uma das extremidades da base.
2. Em seguida fixe os **2 conectores L 2 furos** nas pontas que ficaram para cima dos **conectores L 4 furos**. Então acople o **servo motor aos 2 conectores L 2 furos** e fixe também sua **haste** com o parafuso apropriado.
3. Acople **a barra de 8 furos à haste** do servo motor. Ela servirá como **braço da cancela.**
4. Conecte o servo motor na **porta D2** da placa Darwin.

Automatizando a cancela

Após a realização da **montagem** vamos programar a nossa estrutura e **enviar o código** para a CPU do **Inventor**. Para isto você vai utilizar o código do **Arduíno**.

1. Na área de trabalho do seu computador dê dois cliques no ícone do **Arduino.**
2. Então crie o programa mostrado abaixo:

Programação:

```
#include <darwin.h>
void setup() {
 ServoNaPorta(D2);
}

void loop() {

 MoverServo(D2, 0);
 EsperarSegundos(2);
 MoverServo(D2, 90);
 EsperarSegundos(2);
}
```

Entendendo o que o programa faz:

O segredo nesse programa é fazer com que o **servo motor** se posicione em lugares diferentes para ser feito o **movimento de uma cancela automática**.

Primeiramente posicionamos o **servo na posição 0**, depois **esperamos 2 segundos** para o próximo movimento.

Neste ponto talvez seja necessário você **desparafusar a haste de acrílico do servo motor** e fazer algumas experiências para **descobrir onde fica a posição zero** em relação à sua montagem.

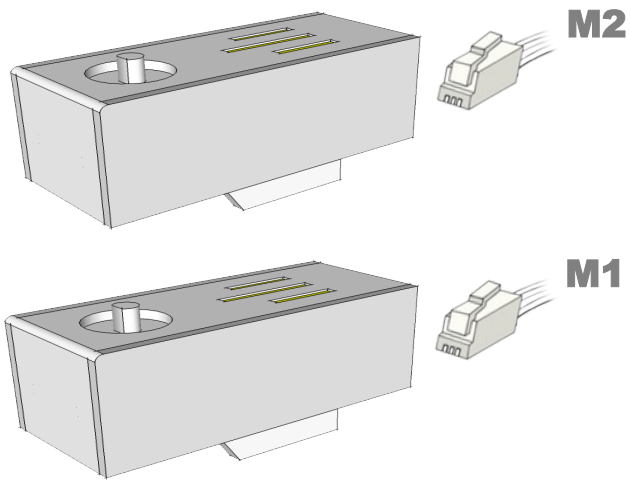
No próximo movimento iremos **abrir a cancela e esperar mais 2 segundos**. Por isso posicionamos o eixo do motor no **ângulo 90**.

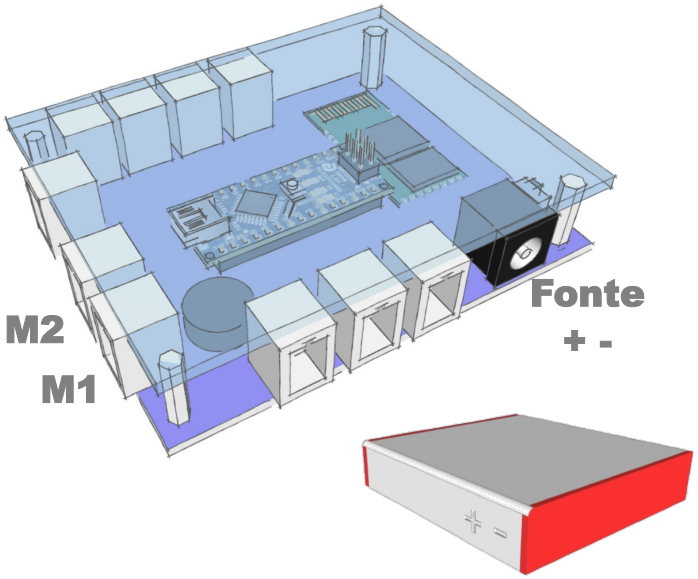
Caso a cancela não esteja abrindo adequadamente, **experimente mudar o ângulo de 90** para outros ângulos mais próximos até achar um que mais lhe agrade.

Acionando motores de Corrente Contínua (CC)

Os **motores** são dispositivos muito utilizados na robótica. Eles podem ser de vários tipos e formatos. Para que eles funcionem é necessário ligar um dispositivo **conhecido como ponte H**, o qual já vem embutido na placa **Darwin da Openrobotics**. Ela possui duas **entradas específicas para os motores CC** que estão com os nomes de **M1** e **M2** . Vejamos como funciona.

Acionando motores CC





Programação:

```
#include <darwin.h>
void setup() {
  AtivarMotores();
}
void loop() {
  MotorM1(Frente);
  MotorM2(Frente);
  EsperarSegundos(1);
  MotorM1(Parar);
  MotorM2(Parar);
  EsperarSegundos(1);
  MotorM1(Re);
  MotorM2(Re);
  EsperarSegundos(1);
 }
```

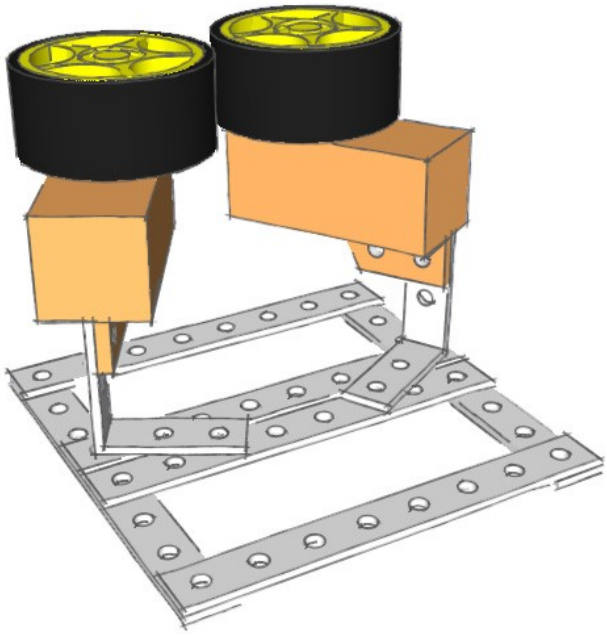
Desafio desta aula:

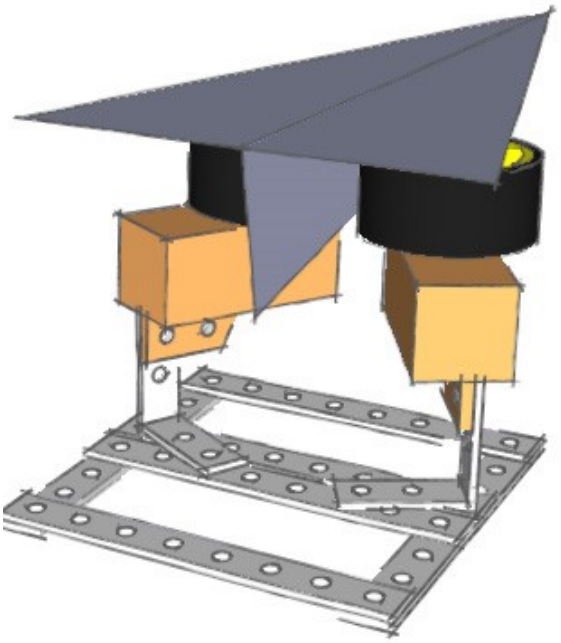
1. Ligue um botão na porta D2. Faça com que os dois motores girem apenas quando o botão estiver pressionado.

Montando um lançador de aviões de papel

Quando **lançamos um corpo verticalmente para cima** notamos que ele sobe até uma certa altura e depois cai, porque é **atraído pela Terra**. O mesmo acontece quando soltamos um corpo de determinada altura. Os corpos são atraídos pela Terra porque em torno dela há uma região chamada **campo gravitacional** exercendo atração sobre eles.

Assim sendo, se criarmos um **lançador robótico de aviões de papel**, em algum momento o aviãozinho vai pousar, ou melhor, cair. Vamos experimentar isso:





Você vai precisar dos seguintes materiais: 2 motores CC, 2 conectores L 4 furos, 6 barras B 8 furos, 2 rodas de borracha, 1 avião de papel, 1 botão de toque, placa Darwin, parafusos e porcas.

Para montar o lançador de aviões faça o seguinte:

1. Para esta montagem vamos ligar **dois motores em uma base com 6 barras de 8 furos**.
2. Perceba pela imagem que as **peças em L precisam formar um ângulo** de forma que as duas rodas se encostem, para que crie uma região de atrito onde o avião será posicionado.
3. A ideia é fazer com que, ao ligar os motores, eles possam ejetar o avião com uma força relativamente baixa, mas que possa lança-lo ao ar.
4. Talvez você irá **precisar de uma gominha** para fazer com que os dois motores fiquem pressionados um contra o outro.
5. Conecte um **botão push na porta D2.**
6. Conecte os motores cc **nas portas M1 e M2**.

Robotizando o lançador de aviões

Após a realização da **montagem** vamos programar a nossa estrutura e **enviar o código** para a CPU do **Inventor**.

Na área de trabalho clique no programa **Arduino**.

1. Então crie o programa mostrado abaixo:

Programação:

```
#include <darwin.h>
int ValorDoBotao;
void setup() {
  AtivarMotores();
  BotaoNaPorta(D2);
}
void loop() {
   ValorDoBotao = LerBotao(D2);
   if (ValorDoBotao == Sim){
     MotorM1(Frente);
     MotorM2(Re);
   }
   else{
     MotorM1(Parar);
     MotorM2(Parar);
   }
}
```

Entendendo o que o programa faz:

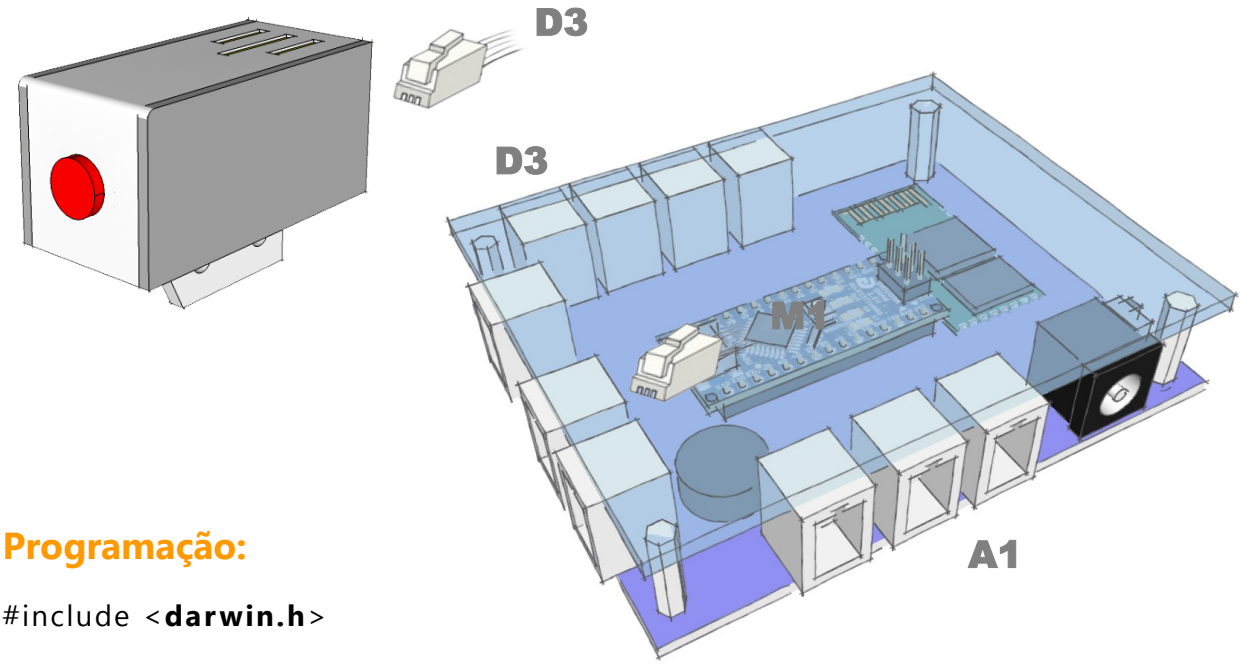
Logo após coletarmos **o estado do botão na variável ValorDoBotao**, perguntamos **SE** ele está **pressionado (Sim)**. Caso esteja, vai começar a **girar os motores CC em sentido contrário** para que o avião de papel, que está pressionado entre as rodas, seja lançado ao ar.

Caso tenha dificuldade em fazer os motores girarem em sentidos opostos, experimente **inverter as opções Frente e Re**.

Conhecendo os sensores de SOM

Os **sensores de som** são dispositivos muito importantes na robótica. Eles são capazes de detectar determinados tipos de sons no ambiente. Nesta aula vamos ver como **podemos ligar um LED** quando batemos **o microfone do sensor detectar um som**. Vejamos como ele funciona.

Ligando um LED com o som



Programação:

```
#include <darwin.h>
int LeituraDaPortaA1;
void setup() {
  LedNaPorta(D3);
}
void loop()
{
   LeituraDaPortaA1 = LerSom(A1);
   if ( LeituraDaPortaA1 == Sim)
      LigarLed(D3);
   else
      DesligarLed(D3);
}
```

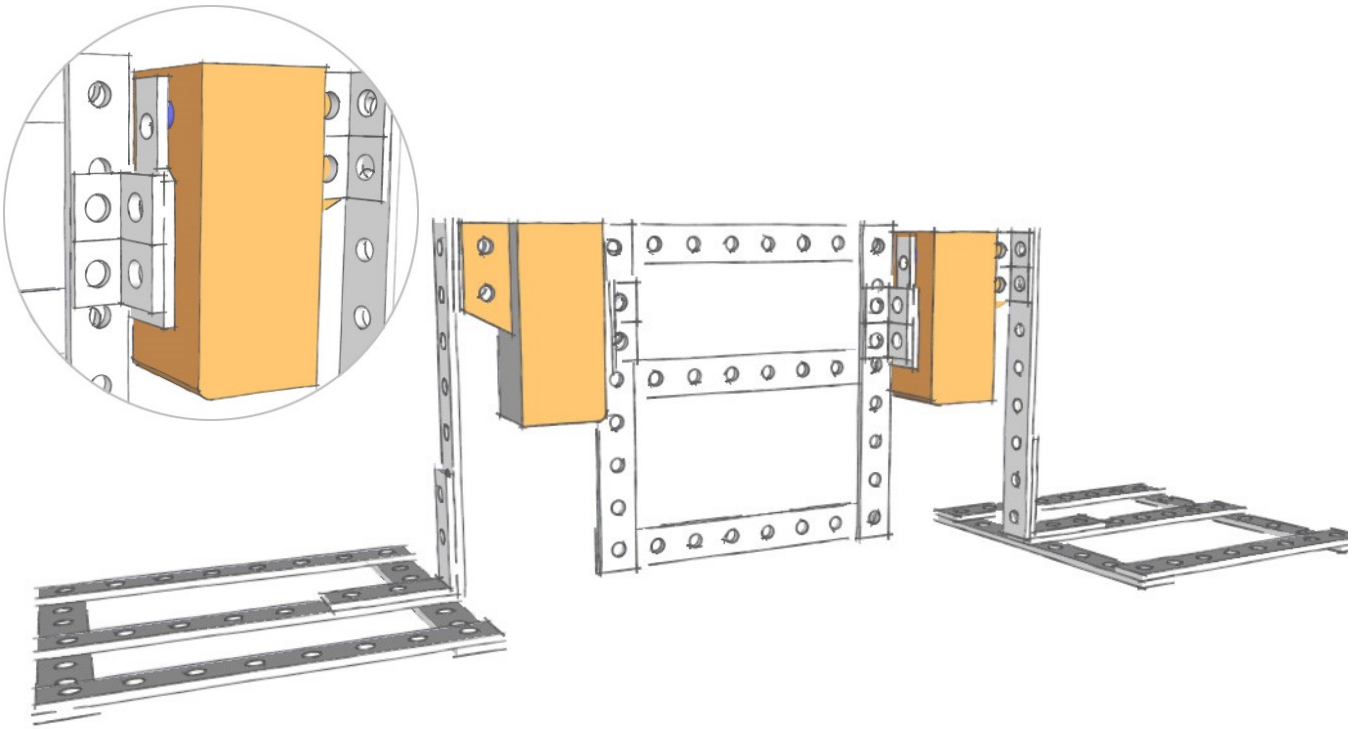
Desafios desta aula:

1. Altere o programa anterior para ligar um outro LED na porta D2 quando o ambiente estiver em silêncio.
2. Instale uma sirene na porta D1. Ela deverá ser acionada quando o sensor detectar algum som.

Montando um portão automático

A tecnologia é a aplicação da ciência. Para termos uma ideia mais clara a respeito dessa afirmação, que tal focarmos, por exemplo, nas **ondas sonoras**?! Elas são simplesmente um **fenômeno natural** que já vem sendo estudado pela **ciência chamada Física** há vários e vários anos. Se você estiver com seu **celular** agora, dentro do seu bolso, está com um belo exemplo de tecnologia, já que ele é sem dúvida **fruto dos diversos estudos que a ciência Física fez sobre as ondas sonoras**.

Outro exemplo seria um portão que se abre automaticamente, ou fecha, a partir de um sensor de som. Vejamos na prática!



Você vai precisar dos seguintes materiais: 17 barras longas 8 furos, 2 conectores L 4 furos, 8 conectores L 2 furos, 2 servos motores, 2 hastes para servo S 2 furos, 1 sensor de som, plca Darwin, parafusos e porcas.

Para montar o portão faça o seguinte:

1. Monte as duas bases, cada uma com **5 barras longas 8 furos**. Em cada base fixe o **conector L 4 furos** em uma das laterais e, na sequência, acople mais uma **barra de 8 furos** subindo os pilares do portão.
2. Instale os **servos motores** nas extremidades das barras de 8 furos usando **2 conectores L 2 em cada** e, depois, fixe as **hastes de 2 furos.** Depois disso, use mais 4 conectores L 2 furos nas hastes dos servos, onde posteriormente colocaremos o portão, o qual é montado facilmente usando mais **5 barras longas 8 furos restantes**.
3. Por fim acople o **sensor de som** em uma das bases e então ligue-o na porta analógica **A0.**
4. Conecte os servo motores **nas portas D2 e D3.**

Automatizando o portão

Após a realização da **montagem** vamos programar a nossa estrutura e **enviar o código** para a CPU do **Inventor**. Para isto você vai utilizar o código do **Arduíno**.

Como o **servo motor trabalha com ângulos de 0 a 180 graus**, pode ser que a posição inicial das hastes não reflita o movimento adequado de abrir e fechar o portão. Por esse motivo, **só fixe o portão nas hastes após ajustar os ângulos dos servos na programação abaixo**.

Programação:

```
#include <darwin.h>
void setup() {
  ServoNaPorta(D2);
  ServoNaPorta(D3);
}
void loop()
{
  if ( LerSom(A0) == Sim ){
    MoverServo(D2, 90);
    MoverServo(D3, 90);
    EsperarSegundos(5);
  }
  else
  {
    MoverServo(D2, 0);
    MoverServo(D3, 180);
  }
}
```

Entendendo o que o programa faz:

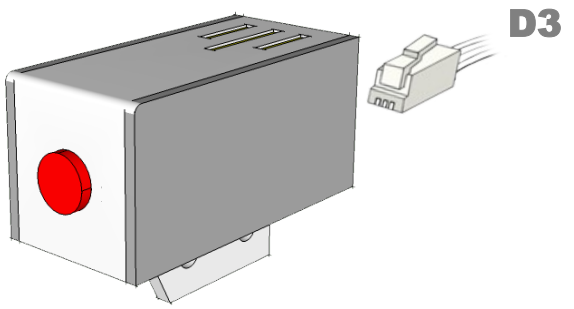
Após coletar a leitura do **sensor de som conectado na porta A0,** perguntamos **SE existe a leitura de algum som nesta porta [if (LerSom(A0) == Sim)]**. Caso positivo **(Sim)** os servos vão se **posicionar em 90 graus** fazendo com que o portão se abra, e **esperar 05 segundos** para voltar à posição inicial. Caso o sensor não detecte algum som, os servos motores se **posicionarão em 0 e 180 graus**, fazendo com que o portão se feche.

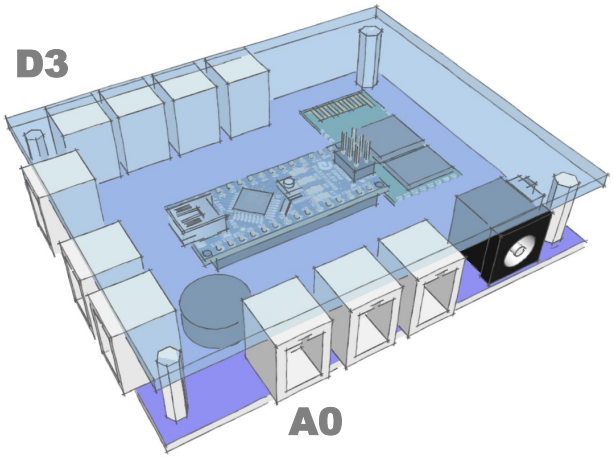
Neste ponto talvez seja necessário você desparafusar a haste de acrílico do servo motor e fazer algumas experiências para descobrir onde fica a posição 0 e 180 em relação à sua montagem.

Sensores de inclinação

Os **sensores de inclinação** podem ser utilizados sempre quando você precisar detectar se algo está posicionado na **vertical** ou na **horizontal**. Eles são muito utilizados para testar se o **robô em movimento** tombou ou não. Vejamos como podemos utilizar o **sensor de inclinação**:

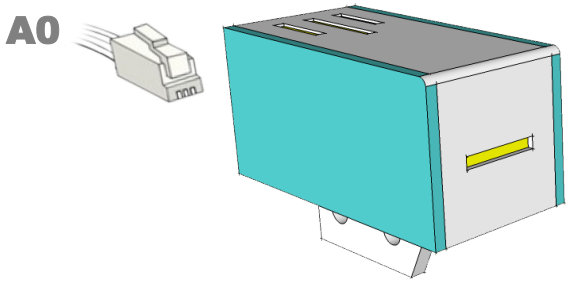
Ligando um LED com a inclinação

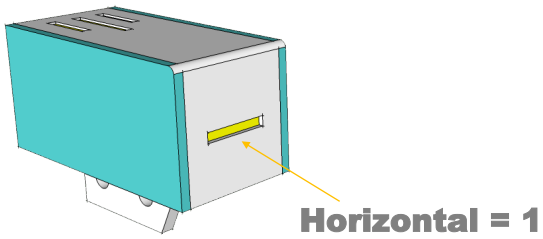


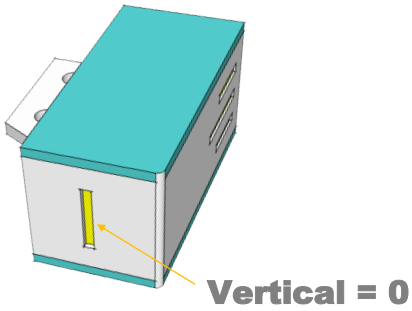


Programação:

```
#include <darwin.h>
int Inclinacao; //Variável para receber o valor da inclinação.
void setup() {
    LedNaPorta(D3);
    AtivarComunicacao();
}
void loop()
{
   Inclinacao = LerInclinacao(A0);
   EnviarParaComputador(Inclinacao);
   if (Inclinacao == Vertical)
      LigarLed(D3);
   if (Inclinacao == Horizontal)
     DesligarLed(D3);
}
```







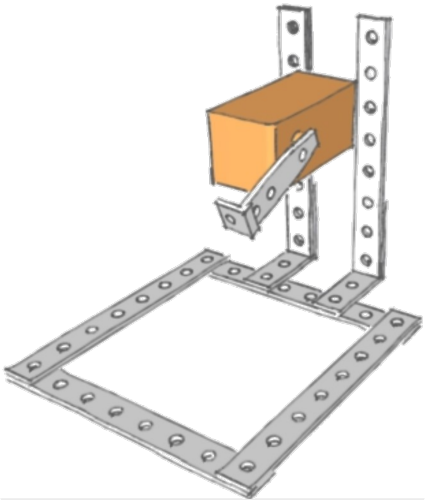
Novidade na área:

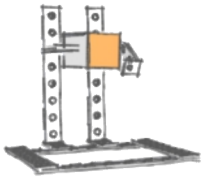
Perceba que no programa acima utilizamos novamente as duas funções: **AtivarComunicacao()** e **EnviarParaComputador().** Lembre-se de que com elas é possível receber e monitorar o valor sendo lido por um sensor em tempo real. Para isso, enquanto o programa estiver em execução, clique no **Ferramentas** do programa do Arduino e acione a opção **Monitor Serial**. Ao girar o sensor de inclinação, você vai perceber que **Horizontal equivale a 1** e **Vertical equivale a 0**.

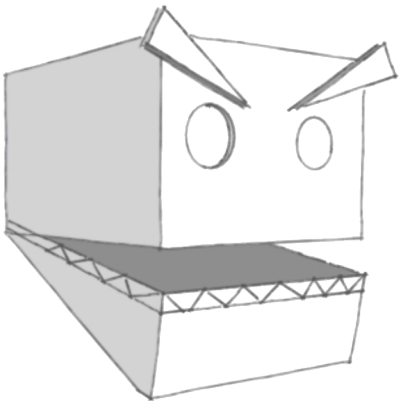
Montando uma cabeça de robô

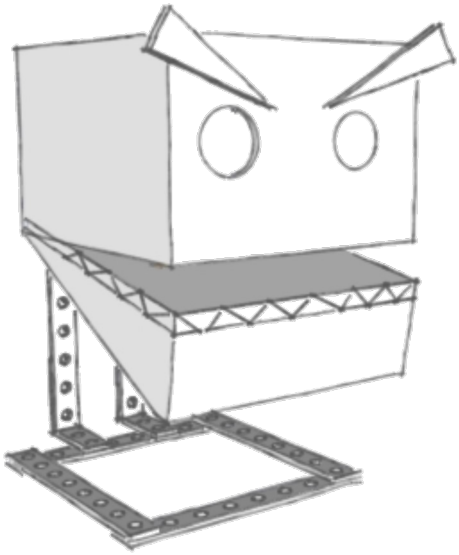
A mordida de um tubarão branco da espécie Carcharodon carcharias pode chegar a **1,8 tonelada** - cerca de metade da força da mordida de um **Tiranossauro Rex**, segundo um estudo de pesquisadores australianos. Um exemplar dessa espécime com 2,4 metros e 240 kg tem, relativamente a suas dimensões e sua massa corporal, **uma das mordidas mais fortes do mundo**.

Vamos agora montar a nossa própria cabeça de robô para simular a mordida de um **tubarão** ou mesmo do poderoso **T-Rex**. Acompanhe:









Você vai precisar dos seguintes materiais:

1 servo motor, 1 haste para o servo S 2 furos, 1 sensor de inclinação, 6 barras longas 8 furos, 2 conectores L 4 furos, 1 conector L 2 furos, papelão cortado segundo os desenhos, placa Darwin, parafusos e porcas.

Para montar a cabeça de monstro faça o seguinte:

1. Comece montando a base com **4 barras longas de 8 furos**. Em uma das laterais da base, acople os **2 conectores L 4 furos**.
2. Anexe mais duas barras longas de **8 furos aos conectores L 4 furos**. Depois instale o **servo motor** entre essas duas barras longas. Fixe **a haste S 2 furos** no servo motor com o parafuso apropriado. Coloque o **conector L 2 furos** na extremidade da haste.
3. Monte a cabeça com o papelão envolvendo o servo motor de forma que a **mandíbula** possa ser fixada **na ponta do conector L 2 furos.** Use parafusos para fixar a mandíbula de modo que ela fique móvel.
4. Conecte o sensor de inclinação na **porta A0**, mas deixe-o livre na montagem para que você possa virá-lo à vontade. Isso porque a ideia é fazer o monstro abrir e fechar a boca em função da inclinação lida pelo sensor.
5. O servo deve estar ligado na porta **D3**.

Programando a mandíbula do robô com o sensor de inclinação

Após a realização da **montagem** vamos programar a nossa estrutura e **enviar o código** para a CPU do **Inventor**. Para isto você vai utilizar o código do **Arduíno**.

1. Na área de trabalho do seu computador dê dois cliques no ícone do **Arduino.**

Programação:

```
#include <darwin.h>
int Inclinacao;
void setup() {
  ServoNaPorta(D3);
}
void loop() {

  Inclinacao = LerInclinacao(A0);

  if ( Inclinacao == Vertical )
    MoverServo(D3, 180);
  else
    MoverServo(D3, 90);
}
```

Entendendo o que o programa faz:

Primeiramente coletamos a leitura do **sensor de inclinação conectado na porta A0** e armazenamos na **variável Inclinacao.**

Se o sensor de inclinação estiver na posição **Vertical**, o servo motor vai se posicionar no ângulo 180, simulando a abertura da boca do monstro. Ajuste esse ângulo na programação em função da posição da haste que você fixou. Exatamente por esse motivo, **é interessante você fazer testes primeiro antes de fixar a mandíbula na haste**.

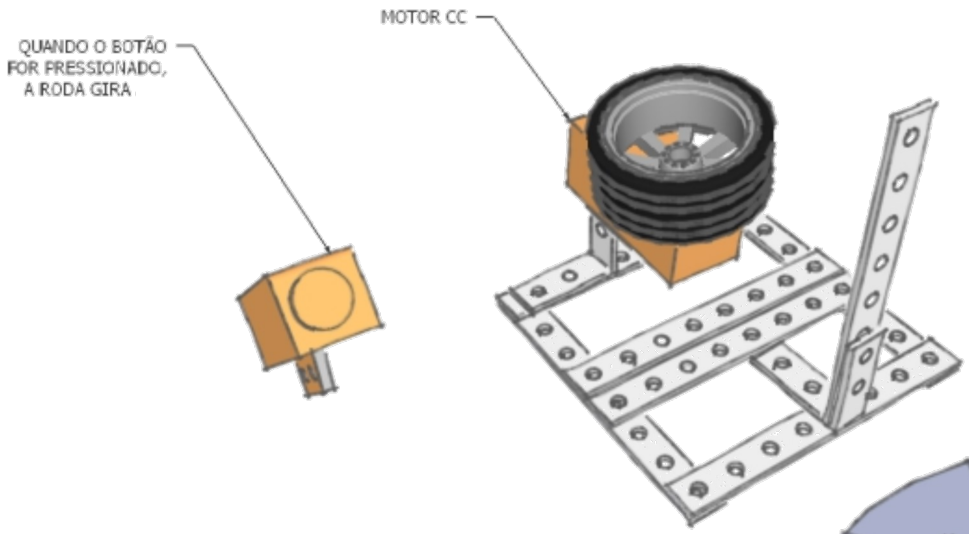
Se o sensor de inclinação estiver na posição **Horizontal**, o servo motor vai se posicionar no ângulo 90, simulando o fechamento da boca do monstro. Novamente, pode ser necessário ajustar esse ângulo na programação em função da posição observada da haste.

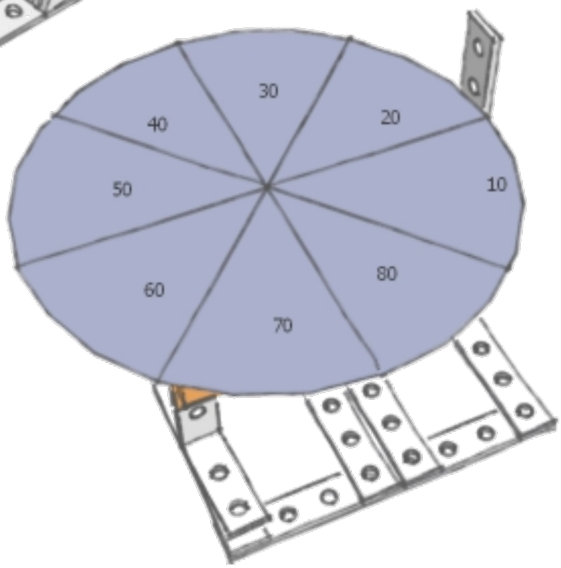
Criando o jogo da roleta

Se a gente tivesse uma **roleta com alguns números marcados**, montada sobre uma estrutura que permitisse girá-la para que ela parasse **selecionando um dos números aleatoriamente** a cada acionamento, daria para **brincar de Matemática de algumas formas diferentes**.

Adivinhar qual número será escolhido já seria uma brincadeira interessante.

Descobrir qual membro da equipe consegue multiplicar mentalmente mais rápido um primeiro número sorteado pelo segundo é outro bom exemplo. O mesmo vale para a subtração e a soma. A seguir você verá como montar essa estrutura. Acompanhe:





Você vai precisar dos seguintes materiais:

1 motor CC, 1 roda de borracha, 1 botão de toque, 7 barras 8 furos, 2 conectores L 4 furos, 1 roleta desenhada em papelão, placa Darwin, parafusos e porcas.

Para montar o jogo da roleta faça o seguinte:

1. Comece montando a base com **6 barras longas 8 furos**. Depois utilize um **conector L 4 furos** em uma das laterais da base para subir a **barra longa de 8 furos** restante. Ela será o **marcador** necessário para selecionar um número qualquer na roleta.
2. Na lateral oposta fixe outro **conector L 4 furos** e então acople o **motor CC** nele. Depois disso, anexe a **roda** ao eixo do motor CC que estiver para cima. Usando uma **fita adesiva fixe a roleta de papelão sobre a roda**.
3. O **botão de toque** pode ficar solto na montagem. Ele deve ser conectado **na porta D3**.
4. O motor CC deve ser conectado **na porta M1**.
5. Como vamos utilizar o motor CC, também é necessário conectar **a fonte de alimentação (+ -) na placa Darwin**.

Robotizando nosso jogo da roleta

Após a realização da **montagem** vamos programar a nossa estrutura e **enviar o código** para a CPU do **Inventor**. Para isto você vai utilizar o código do **Arduino**.

1. Na área de trabalho do seu computador dê dois cliques no ícone do **Arduino.**
2. Então crie o programa mostrado abaixo:

Programação:

```
#include <darwin.h>
void setup() {
  BotaoNaPorta(D3);
  AtivarMotores();
}

void loop() {
  if ( LerBotao(D3) == Sim )
  {
    MotorM1(Frente);
    EsperarSegundos ( SortearNumero(1, 5) );
    MotorM1(Parar);
  }
}
```

Entendendo o que o programa faz:

O segredo nesse programa é fazer com que **o motor CC gire a roda** (e consequentemente a roleta de números) assim que a **pessoa pressionar o botão**.

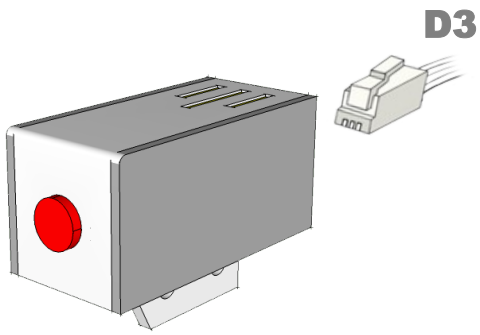
Primeiramente coletamos a leitura do **sensor de toque (botão) na porta D3.**

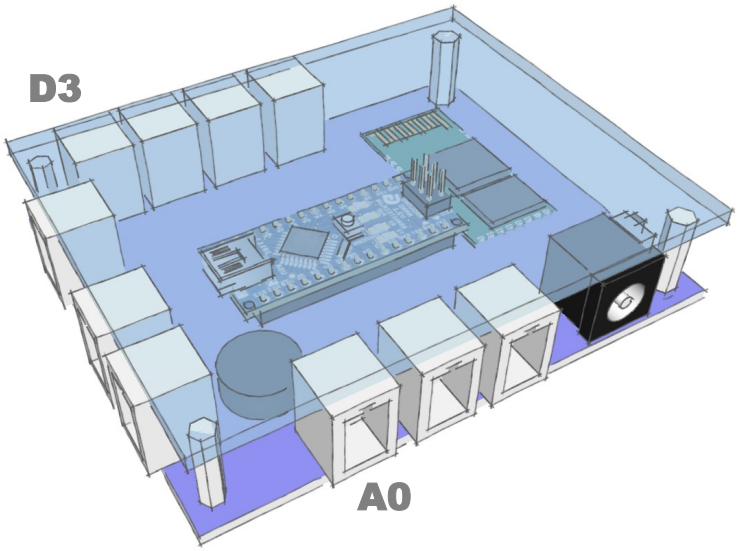
No próximo passo perguntamos **SE** o botão está pressionado. Caso esteja (**Sim**), vai começar a girar o motor CC por um **tempo sorteado entre 01 e 05 segundos**, para que não haja manipulação externa. Por fim, **paramos o motor CC** para vermos qual número da roleta ficou perto do **marcador**.

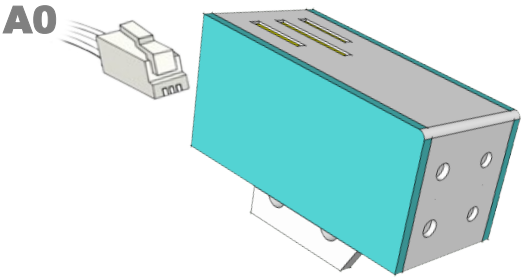
Utilizando o sensor magnético

Os **sensores magnéticos** devem ser usados quando você precisa detectar um **campo magnético**, como por exemplo, um **imã**. Eles podem ser muito úteis quando você precisar detectar algo sem ter que encostar neste objeto, uma vez que o imã gera um **campo ao seu redor**, invisível, mas que pode ser **detectado com este sensor**. Para essa montagem utilize **um imã**, que deve ser **posto em contato com a frente do sensor durante os testes**.

Ligando um LED quando há campo magnético







Programação:

```
#include <darwin.h>
int TemCampoMagnetico;
void setup() {
    LedNaPorta(D3);
    AtivarComunicacao();
}
void loop()
{
   TemCampoMagnetico = LerMagnetico(A0);
   EnviarParaComputador(TemCampoMagnetico);
   if (TemCampoMagnetico == Sim)
      LigarLed(D3);
   else
      DesligarLed(D3);
}
```

Desafios desta aula:

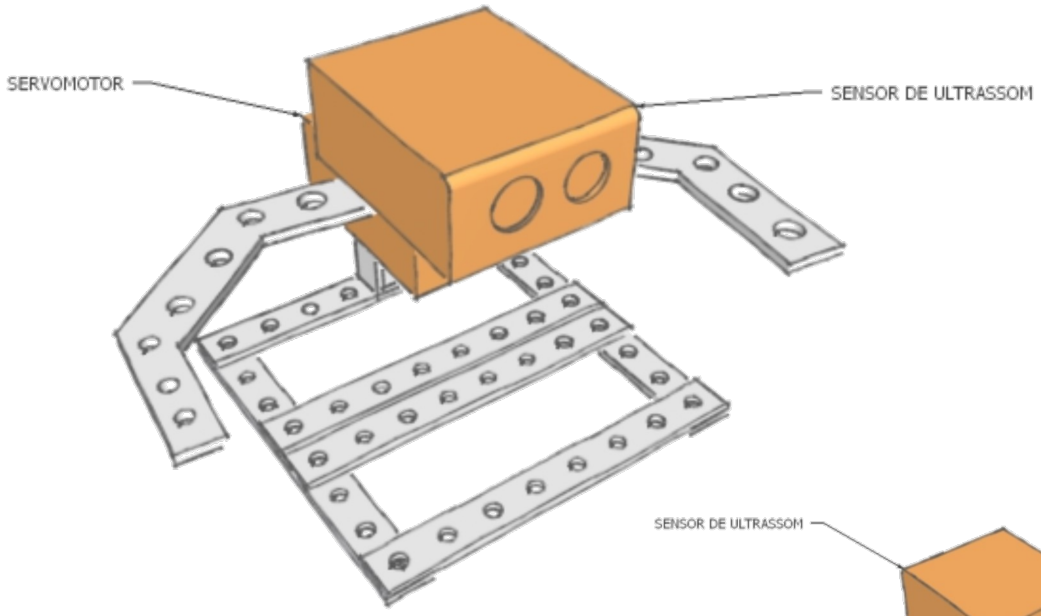
1. Crie uma cancela e faça com que ela se abra ou feche conforme a aproximação do imã.
2. Faça a sirene interna da placa Darwin apitar 3 vezes quando o imã aproximar.

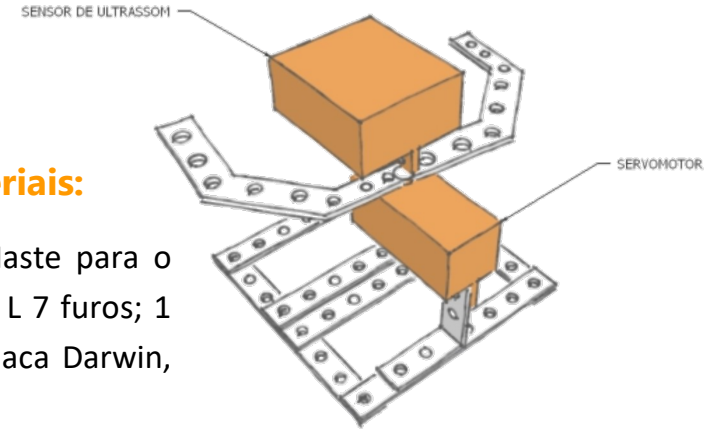
Montando um robô que busca por objetos

É muito bacana quando a gente pega uma bolinha para brincar com nosso cão e o bichinho fica louco olhando de um lado e de outro tentando acompanhar o movimento, mesmo estando a tão desejada esfera ainda em nossas mãos.

Depois da Robótica apareceu também a tal **Inteligência Artificial**. No meio das discussões entre o que é, e o que não é, possível transferir de intelecto para uma máquina, um grupo de pessoas gosta de dizer que os **robôs e máquinas podem se comportar de forma inteligente**.

No seu próximo desafio você construíra um belo exemplo de "comportamento inteligente", vamos criar uma cabeça de robô que procura por objetos próximos.





Você vai precisar dos seguintes materiais:

1 Sensor de ultrassom; 1 Servo motor; 1 Haste para o servo motor; 6 Barras 8 furos; 2 Conectores L 7 furos; 1 Conector L 4 furos; 1 Conector L 2 furos; Placa Darwin, Parafusos e porcas.

Para montar o robô faça o seguinte:

1. Comece montando a base com **6 barras longas de 8 furos**. Na lateral de uma delas, fixe a peça **L com 4 furos**., e nela conecte o servo motor. Ligue no servo motor a Haste pequena para servo com 2 furos, na ponta desta haste conecte uma peça em L pequena com 2 furos, e nela fixe o sensor de distância por ultrassom. Para os braços do robô utilize 2 conectores L 7 furos.
2. O servo motor deverá ficar ligado **na porta D3** da placa do Inventor.
3. Já o sensor de distância deverá ser colocado **na porta DIST**.

Programando nosso robô que procura objetos

Após a realização da **montagem** vamos programar a nossa estrutura e **enviar o código** para a CPU do **Inventor**. Para isto você vai utilizar o código do **Arduino**.

Programação:

```
#include <darwin.h>
int Distancia;
int Angulo;
void setup()
{
  ServoNaPorta (D3);
}
void loop()
{
   Distancia = LerDistancia ( );
   if (Distancia > 10)
   {
      Angulo = SortearNumero (50, 180);
      MoverServo (D3, Angulo);
      EsperarSegundos (0.5);
   }
}
```

Entendendo o que o programa faz:

Primeiramente são definidos dois locais na memória para armazenar dados (**variáveis**):

- Uma variável para armazenar a **distância** do objeto que será colocado na frente do robô;
- E a outra para armazenar um **ângulo** que será sorteado.

Depois informamos que vamos usar um servo motor **na porta D3**.

Em seguida fazemos a **leitura da distância** do objeto utilizando o sensor de ultrassom. **Se esta distância for maior do que 10 centímetros**, então o robô ficará procurando um objeto movendo a cabeça para um lado e para o outro. Isso é feito através do **sorteio de um número que é enviado como ângulo para o servo motor**.

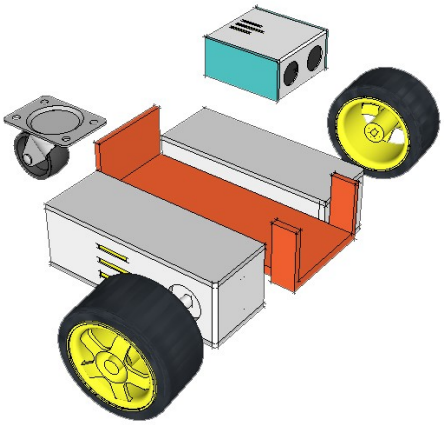
Perceba que você pode mudar os valores para o sorteio, mas eles **deverão variar entre 0 e 180**, pois estes são os ângulos possíveis para o servo motor.

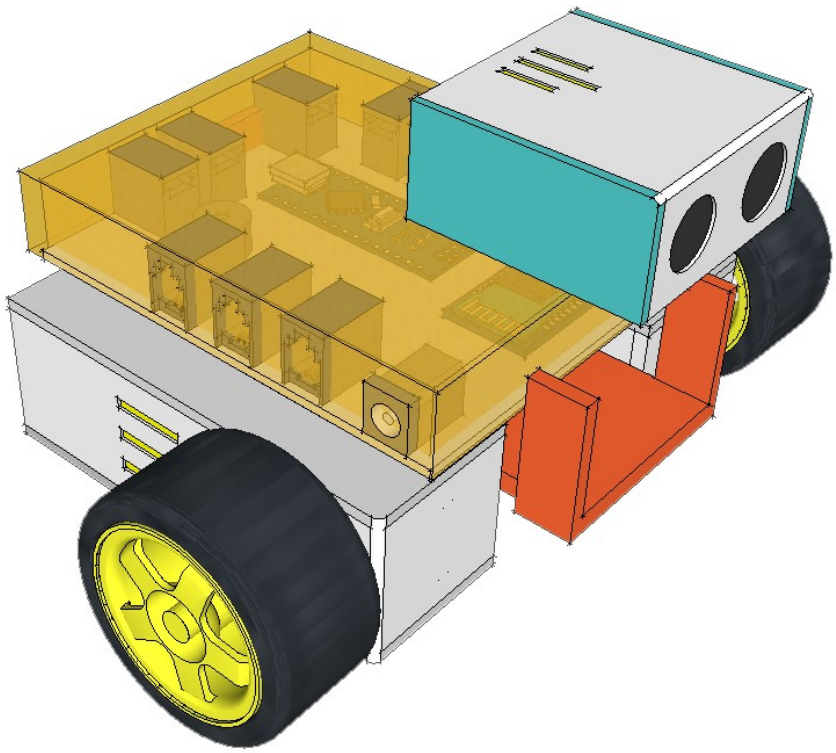
Criando um robô que reconhece e desvia de obstáculos

Um dos grandes desafios de um robô que esteja em movimento é que ele tenha a capacidade de se locomover com uma **certa autonomia**.

Este processo é importante em vários casos. Por exemplo: o robô que está atualmente explorando marte tem uma certa autonomia de navegação, para que ele possa desviar de pedras e buracos durante o percurso. **Esta autonomia é muito importante** neste caso, pois leva aproximadamente 15 minutos para que qualquer comando chegue da terra até o planeta vermelho, tempo suficientemente grande para que o robô se coloque em apuros.

Vamos montar agora um robô que pode **desviar de obstáculos**. Ele vai realizar essa tarefa usando um sensor de distância por ultrassom (semelhante aos que são usados por morcegos).





Agora vamos montar nossa estrutura:

Utilizando a peça de acrílico do chassi, comece montando os motores na base. Em seguida você pode ligar o conector **L2 na parte de cima da CPU Darwin** e nele você deve prender o sensor de ultrassom.

Faça a montagem de uma base sobre o chassi e na parte de cima fixe a placa Darwin, conforme ilustrado no desenho acima.

Conecte os motores CC **nas portas M1 e M2** da placa Darwin.

O sensor de distância deve ser ligado **na porta DIST** da placa.

Programando nosso veículo

Agora vamos ver como programar o nosso veículo para que ele possa reconhecer e desviar dos obstáculos que encontrar.

Programação:

```
#include <darwin.h>
int distancia;
void setup()
{
  AtivarMotores();
}
void loop()
{
  distancia = LerDistancia();
  if (distancia > 20)
  {
    MotorM1(Frente);
    MotorM2(Frente);
  }
  else
  {
    MotorM1(Re);
    EsperarSegundos(1);
  }
}
```

Entendendo o que o programa faz:

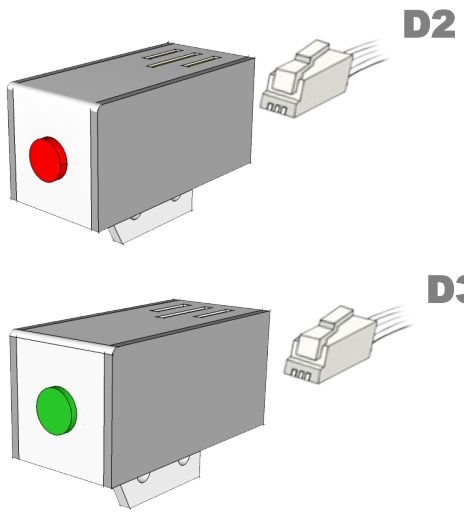
No código em Arduino a programação começa com a importação da biblioteca **<darwin.h>** que contém a programação que deixa a **sintaxe dos comandos** muito mais fácil.

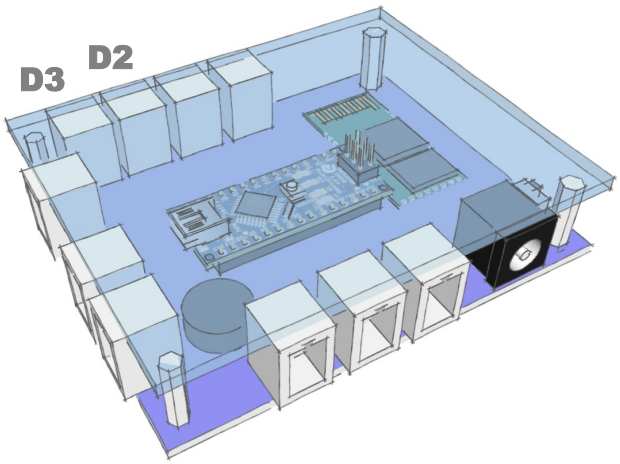
O programa basicamente faz a leitura da distância no sensor de ultrassom. Caso esta distância esteja **acima de 20 centímetros**, isso significa que **não tem qualquer objeto próximo do veículo**. Quando esta distância for **menor que 20 centímetros**, então **um dos motores faz o movimento contrário (Ré)**. Este processo faz com que o veículo faça um giro e se distancie do objeto encontrado, para na sequência voltar a andar normalmente.

Enviando comandos do computador para o robô

Em várias situações é necessário **enviar comandos do computador para o robô**, para que ele realize alguma atividade. Para isso podemos utilizar o **monitor serial**. Com ele podemos **digitar algum valor** no teclado, que ele será **lido pelo robô** e executado conforme a programação dada.

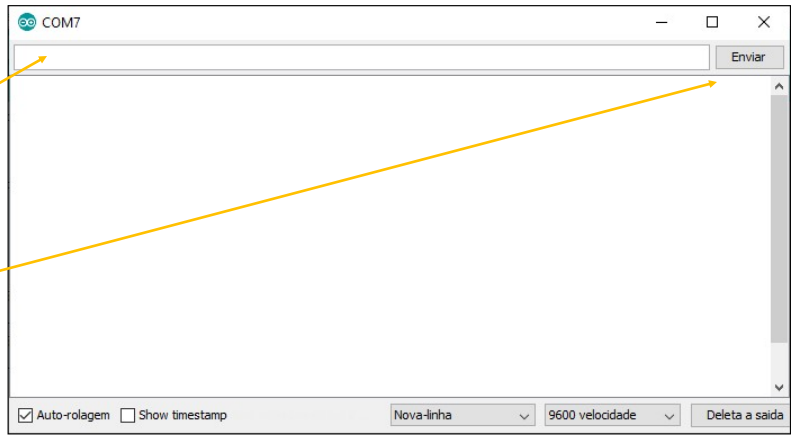
Acionando LEDs pelo computador





Programação:

```
#include <darwin.h>
int comando; //Variável para receber o valor do computador.
void setup() {
   LedNaPorta(D2);  LedNaPorta(D3) ;  AtivarComunicacao();
}
void loop()  {
  comando = ReceberNumeroPeloComputador();
  if (comando == 1)
     LigarLed(D2);
  if (comando == 2)
     LigarLed(D3);
}
```



Depois de carregar o programa abra o **Monitor serial** no menu **Ferramentas**. Então digite **nesta caixa** os **comandos 1 ou 2** e clique no botão **Enviar**.

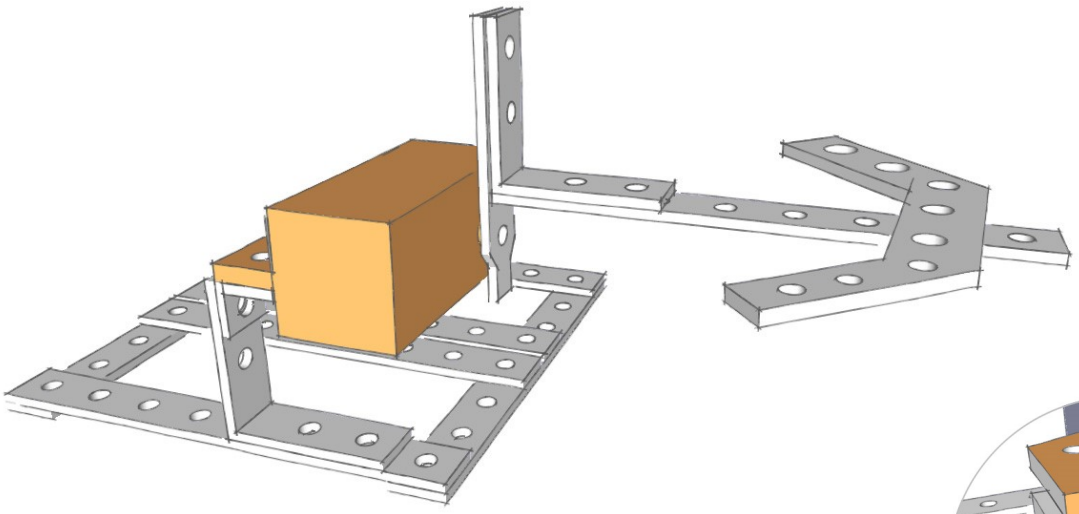
Desafios desta aula:

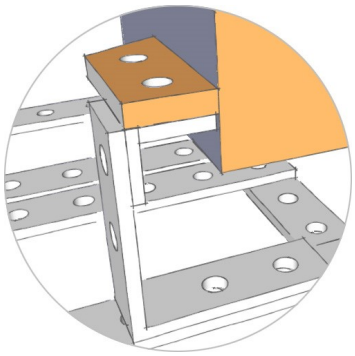
1. Altere o programa para que você possa também desligar os LEDs com comandos do computador.
2. Acrescente uma sirene que também deverá ligar ou desligar.

Controlando um avião por bluetooth

Vamos utilizar o celular para controlar o nosso avião

Vamos agora fazer uma montagem que precisa de um controle apropriado para pilotar aviões, embarcações e outros dirigíveis, **mais conhecido como manche**. A ideia é fazer com que a inclinação do celular controle a **inclinação da estrutura montada**, fazendo dele um **manche**.





Para esta montagem você vai precisar dos seguintes materiais:

1 servo motor; 1 haste para o servo S 2 furos; 7 barras 8 furos; 1 conector L 7 furos; 2 conectores L 4 furos; 1 conectores L 2 furos; 1 celular com o **aplicativo OpenController instalado**; placa Darwin, parafusos e porcas.

Onde obter o aplicativo?

O aplicativo **OpenController** para o celular pode ser baixado em nosso site **openrobotics.com.br** ou na **Play Store**..

Agora vamos montar nossa estrutura:

Comece montando a base com **6 peças de 8 furos**. Na peça central, coloque **uma L 4 furos**, para conectar a **peça L 2 furos**.

Instale o **servo motor** nas extremidades da **peça L 2 furos** e então conecte a **haste S 2 furos** com o parafuso apropriado.

Agora fixe o **conector L 4 furos** na extremidade da **barra 8 furos** que sobrou. Depois, na outra extremidade da barra, pule um furo e fixe o **conector L 4 furos** de forma a ficar centralizado na barra.

Falta ainda conectar esta última etapa montada à **haste S 2 furos**.

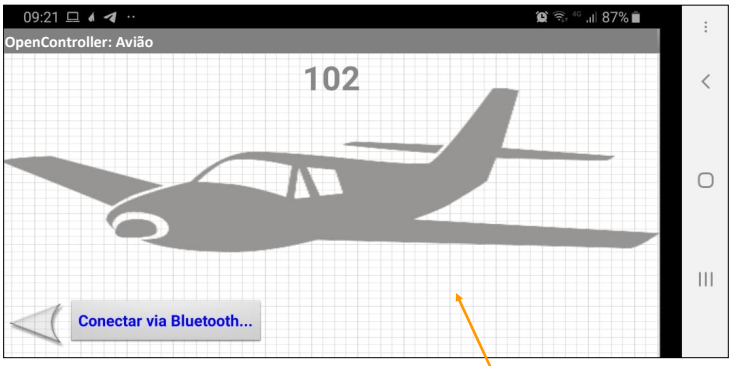
Não se esqueça de ligar o servo motor na porta **D3**.

Programando nosso avião

Agora vamos ver como podemos utilizar o celular para controlar o avião. Veja que você vai precisar do nosso aplicativo **OpenController**. Uma vez o aplicativo instalado, é necessário **parear o celular com o bluetooth** presente na placa Darwin, para depois abrir o aplicativo e então **fazer a conexão**. Além disso, a placa Darwin deve estar carregada com o **código Arduino** abaixo:

Programação:

```
#include <darwin.h>
int Inclinacao;
void setup() {
  AtivarBluetooth();
  ServoNaPorta(D3);
}
void loop() {
  if (BluetoothEstaRecebendo() == Sim)
  {
    Inclinacao = ReceberNumeroPeloBluetooth();
    if (Inclinacao >= 0)
    {
      MoverServo(D3, Inclinacao);
    }
  }
}
```



Entendendo o que o programa faz:

O algoritmo do **Arduino** recebe a inclinação vinda do sensor acelerômetro do celular e a repassa para a inclinação do servo motor. Então você deve navegar no aplicativo até chegar **nessa tela** para depois começar a inclinar o celular de um lado para o outro para ver o avião refletir o movimento.

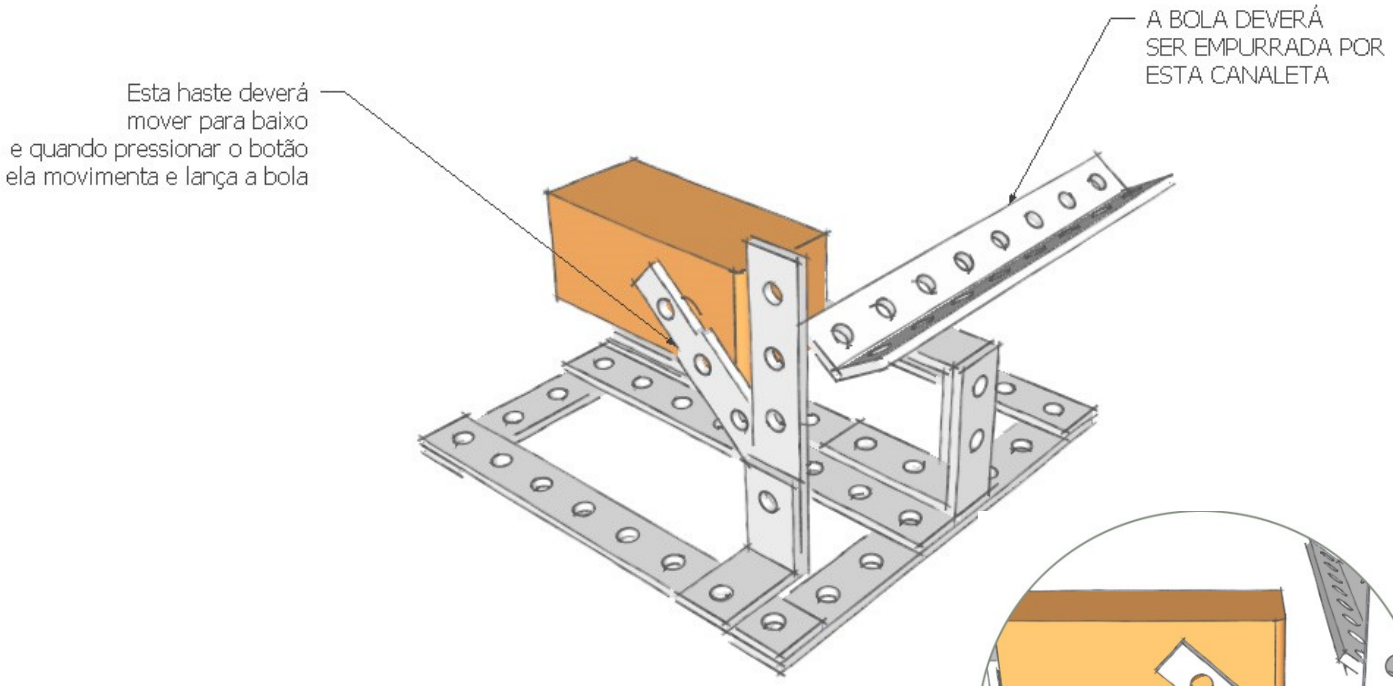
No programa **Arduino**, primeiramente criamos uma variável para receber a **inclinação** que virá via Bluetooth do celular ou tablet. Na sequência fazemos as configurações iniciais para receber os dados do Bluetooth, além de declarar em qual porta está o servo motor.

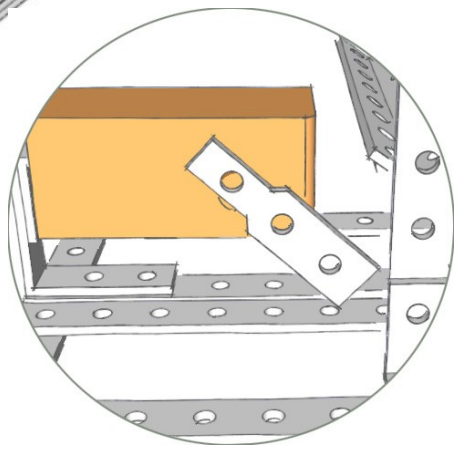
Se o sinal de Bluetooth estiver disponível, então converte o valor recebido em um número por meio da função **ReceberNumeroPeloBluetooth().** Uma inclinação igual ou maior do que zero grau pode ser passada para o servo motor, posicionando o seu eixo exatamente na inclinação recebida do celular ou tablet.

Simulando uma catapulta medieval

Uma máquina que lança bolinhas

Quer ficar bom em alguma coisa? Treine, treine, treine, treine, treine, treine, treine e depois, treine... A **evolução tecnológica** trouxe também algumas máquinas que ajudam as pessoas a treinarem. Hoje já tem diversas aplicações e uma das mais interessantes é a máquina que ajuda os goleiros dos times de futebol a treinar. Tem também aquela que **atira as bolinhas** para os tenistas poderem rebater. Na próxima montagem você vai criar uma maquinha dessas para ver como é simples o princípio delas.





Para esta montagem você vai precisar dos seguintes materiais:

1 Botão de toque; 1 Servo motor; 9 Barras 7 furos; 1 Barra 5 furos; 1 Haste para servo; 2 peças L 4 e 2 L 2 furos; Placa Darwin; Bolinha de ping-pong, Parafusos e porcas

Agora vamos montar nossa estrutura:

Comece montando a base com **6 peças de 8 furos**. Na peça central, coloque duas **L 4 furos**, Uma vai segurar o servo motor, a outra vai segurar a guia para lançar a bola.

Utilize um conector **L 4** para segurar o conector de **5 furos** que será o apoio para a bola que será lançada.

Para fazer a rampa onde a bola ficará esperando para ser lançada, utilize **2 peças em V com 2 furos**, e nelas conecte as **2 barras de 7 furos** para formar o caminho para a bola percorrer até ser lançada.

Conecte o **servo na porta D3** e o **botão de toque na porta D2**.

Programando a catapulta

Agora vamos programar a nossa catapulta. Depois de fazer a programação, experimente colocar energia extra na placa. Você vai perceber que isso traz muito mais força para o lançamento.

Programação:

```
#include <darwin.h>
int Botao;
void setup(){
  ServoNaPorta(D3);
  BotaoNaPorta(D2);
}
void loop(){
 Botao = LerBotao(D2);
 if (Botao == Sim)
 {
    MoverServo(D3, 180);
 }
 if (Botao == Nao)
 {
    MoverServo(D3, 90);
 }
}
```

Entendendo o que o programa faz:

No código em **Arduino** a programação começa com a importação da biblioteca **<darwin.h>** que contém a programação que deixa a **sintaxe dos comandos** muito mais fácil.

Depois vem a definição da **variável Botao** que vai armazenar a leitura feita pelo botão de verdade.

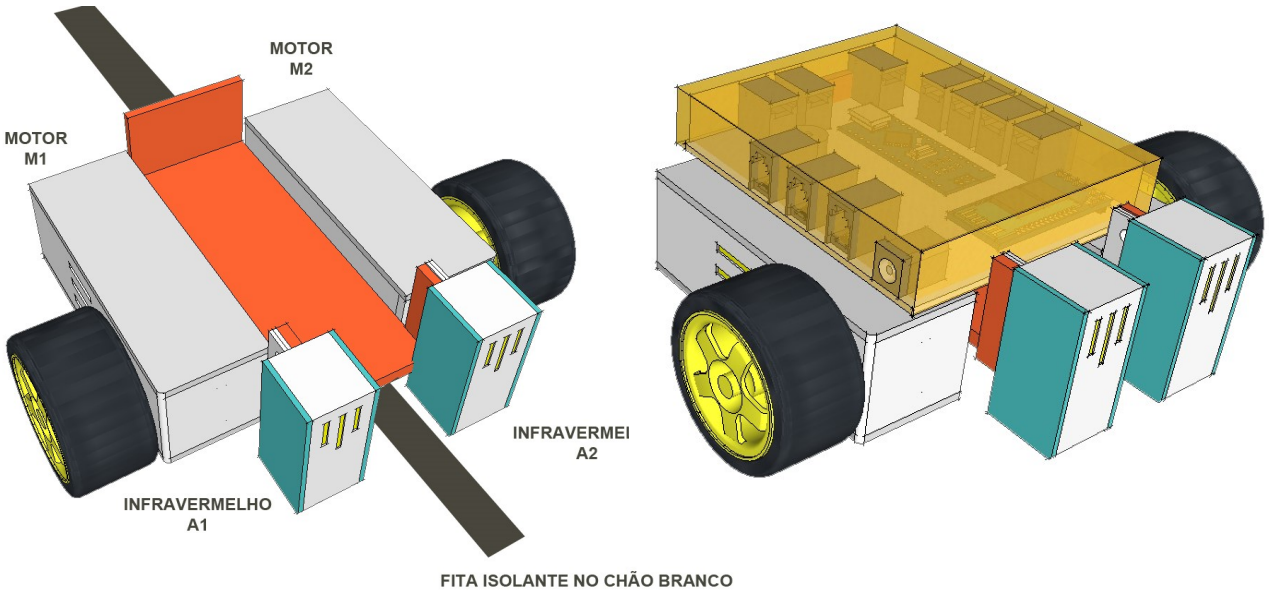
Então é realizada a leitura do botão que está **na porta D2** e o resultado é colocado na **variável Botao**.

Na sequência existe um **teste para verificar se o botão está pressionado**, em caso afirmativo (**Sim**) então o servo que está **na porta D3** da placa irá se movimentar.

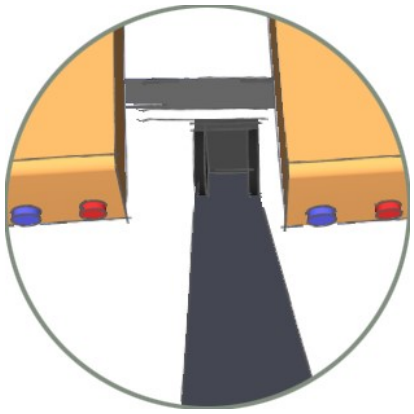
Veja que talvez você tenha que alterar os ângulos, dependendo de como ficou a posição do servo que você instalou na placa.

Montando um robô que segue linhas

Eu não sei você, mas eu, toda vez que leio, ouço ou vejo algo sobre carro autônomo, fico me imaginando entrando em um **veículo sem motorista** que vai me levar para algum lugar. Sinceramente, eu quase sinto antecipadamente aquele friozinho na barriga por causa do receio do veículo dar um problema qualquer comigo dentro. Parece que **esta realidade está ficando cada vez mais próxima**. Fabricantes de automóveis começaram a corrida para ver quem chega primeiro, softwares e aplicativos estão sendo construídos para ajudar e os especialistas já fazem suas previsões para o curto prazo. Vamos aproveitar para construir um **veículo autônomo** que segue uma linha? Você vai precisar de uma **superfície branca** e **fita isolante** para fazer o caminho.



Para esta montagem você vai precisar dos seguintes materiais:



1 Base (chassi) para o veículo; 2 Motores CC; 2 Rodas de borracha; 2 Sensores de infravermelho; Fita isolante, Roda boba; Placa Darwin; Parafusos e porcas.

Para montar o veículo faça o seguinte:

Na base do veículo você vai instalar todos os componentes. Comece fixando os motores na lateral do veículo, **mas deixe as rodas para colocar no final**.

Instale também a **bateria** e a **placa Darwin**.

O funcionamento baseia-se em identificar a cor branca em uma superfície. Quando algum dos sensores **detectar o preto** então ele para de rodar o motor por um tempo mínimo, fazendo com que o carro novamente acerte o traçado.

Conecte os **motores CC nas portas M1 e M2**, bem como os **sensores de infravermelho nas portas A1 e A2, conforme ilustra o desenho acima.**

Como estamos usando os motores, conecte também **a fonte de alimentação na placa Darwin**.

Programando nosso robô que segue linhas

IMPORTANTE: Ao conectar os motores e sensores **na placa Darwin**, siga o **desenho da página anterior** para que o **motor CC e o sensor infravermelho** que estiverem montados no mesmo lado do carro, usem também **a mesma numeração das portas**. Exemplo: **se você conectou o motor do lado esquerdo do carro na porta M1, conecte o sensor infravermelho do lado esquerdo na porta A1.**

Agora vamos programar o nosso veículo que segue linhas de forma autônoma. Para isto você vai utilizar o código do **Arduino**.

Programação:

```
#include <darwin.h>
int corEmA1; int corEmA2;
void setup() {
  AtivarMotores();
}
void loop() {
  corEmA1 = LerInfra(A1);
  corEmA2 = LerInfra(A2);
  MotorM1(Frente);
  MotorM2(Frente);
  if (corEmA1 == Preto)  {
      MotorM1(Parar);
      EsperarSegundos(0.5);
  }
  if (corEmA2 == Preto)  {
      MotorM2(Parar);
      EsperarSegundos(0.5);
  }
}
```

Entendendo o que o programa faz:

Neste programa começamos criando as variáveis **corEmA1** e **corEmA2** que irão receber a leitura dos sensores de infravermelho que estão conectados nas portas analógicas **A1** e **A2**. O carrinho deve iniciar com a faixa preta exatamente no meio dos dois sensores. A ideia basicamente é a seguinte: **os dois motores vão permanecer ligados o tempo todo**. Quando um dos sensores detectar a cor escura (**Preto**) significa que o veículo está **desviando para o lado**, então, basta desligar por meio segundo o motor correspondente que assim o veículo retoma a direção. Um detalhe importante, **caso os motores estejam girando para uma direção errada, substitua no programa o comando Frente por Re**.

Variáveis

Uma variável é um recurso utilizado para armazenar dados em um programa de computador. Todo computador possui algum tipo de memória, e uma variável representa uma região da memória usada para armazenar uma determinada informação. Essa informação pode ser, por exemplo, um número, um caractere ou uma sequência de texto. Para podermos usar uma variável em um programa **Inventor**, nós precisamos fazer uma declaração de variável, como você verá em alguns programas.

Tipos de variáveis

O tipo de dado de uma variável significa, como o próprio nome diz, o tipo de informação que se pode armazenar naquela variável. Em muitas linguagens de programação, como Java e C++, é obrigatório definir o tipo de dado no momento da declaração da variável. No caso dos módulos INVENTOR que usam processador **ATmega**, os tipos mais comuns de dados que utilizamos são:

boolean:	valor verdadeiro (true) ou falso (false)
char:	um caractere
byte:	um byte, ou sequência de 8 bits
int:	número inteiro de 16 bits com sinal (-32768 a 32767)
unsigned int:	número inteiro de 16 bits sem sinal (0 a 65535)
long:	número inteiro de 16 bits com sinal (-2147483648 a 2147483647)
unsigned long:	número inteiro de 16 bits sem sinal (0 a 4294967295)
float:	número real de precisão simples (ponto flutuante)
double:	número real de precisão dupla (ponto flutuante)
string:	sequência de caracteres
void:	tipo vazio (não tem tipo)

Não se preocupe em aprender agora, pois durante a **criação dos códigos da apostila** vamos falando sobre cada uma delas.

OPENROBOTICS INFORMÁTICA LTDA - CNPJ: 14.553.426/0001-87
(31) 3317.6345 (31) 98808.5938 - OPENROBOTICS.COM.BR - CONTATOS@OPENROBOTICS.COM.BR