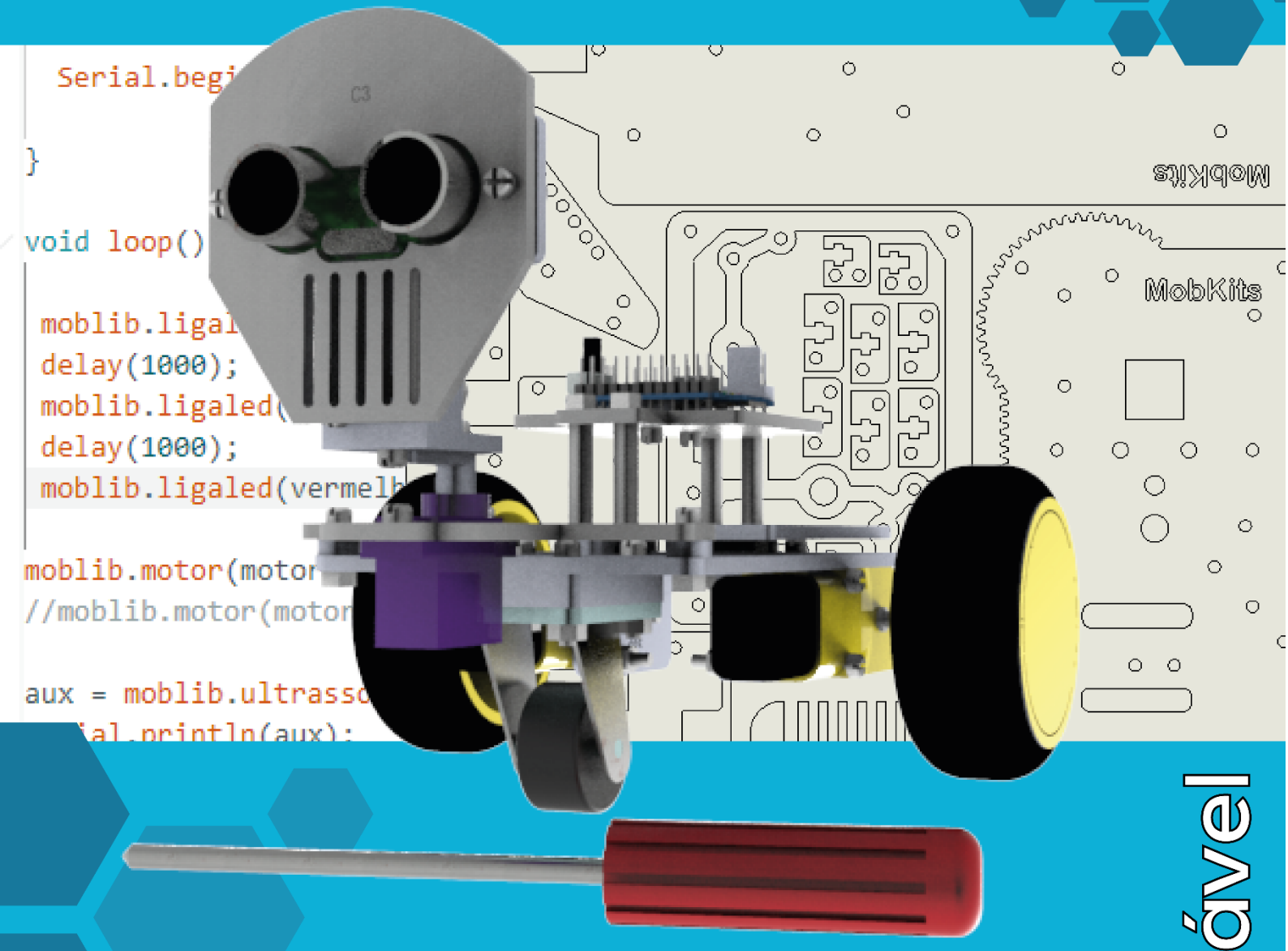
Manual do Usuário



Programável

MobKits®

Rev 1.0

Apresentação

Seja bem-vindo ao mundo da programação com MobKits! Neste módulo vamos ensinar como instalar a interface para programação, como gravar seus códigos na nossa plataforma e utilizar a nossa biblioteca.

Para facilitar o seu desenvolvimento, montamos uma série de funções para controlar de forma prática todos os periféricos da nossa placa de controle.

Claro, fique à vontade para modificar o que achar necessário e aproveite para nos mandar a sua atualização e fazer parte da nossa comunidade de desenvolvedores! Fale conosco pelo e-mail contato@mobkits.com.br.

Após os exemplos de funções ensinamos como deixar sua placa pré-programada novamente, ou seja, original de fábrica, pronta para uso sem precisar programar.

Divirta-se!

Sumário

Interface de Programação

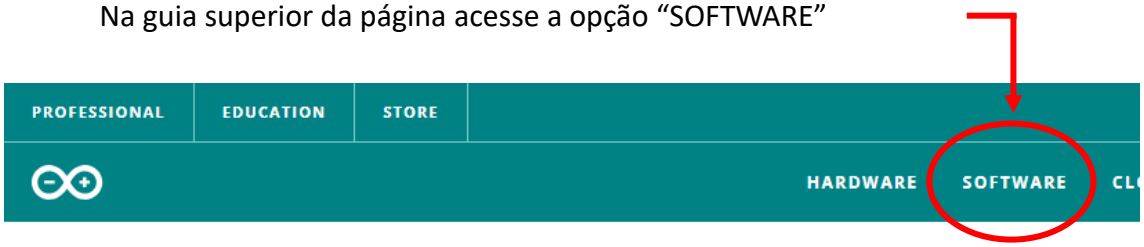
Não vamos reinventar a roda, existem ótimos IDE's, traduzindo do inglês, ambiente de desenvolvimento integrado, como o IDE Arduino, para programar, compilar e gravar os seus projetos.

Primeiro passo

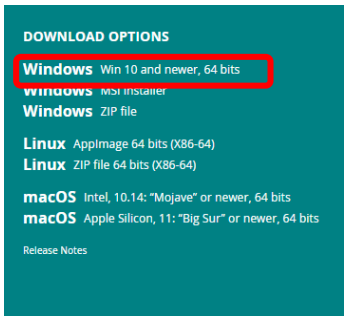
Acesse na internet a página oficial do Arduino em:

https://www.arduino.cc/

Na guia superior da página acesse a opção "SOFTWARE"



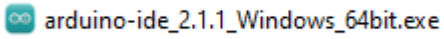
Escolha o arquivo adequado para o seu sistema operacional, na maioria dos casos, se você utiliza Windows a versão executável de 64 bits deve resolver.



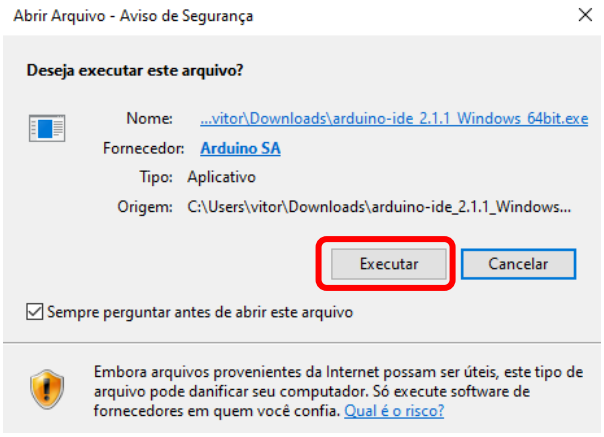
Na próxima página, será pedida uma doação, que não é necessária, mas fique à vontade para contribuir, caso contrário, clique apenas em

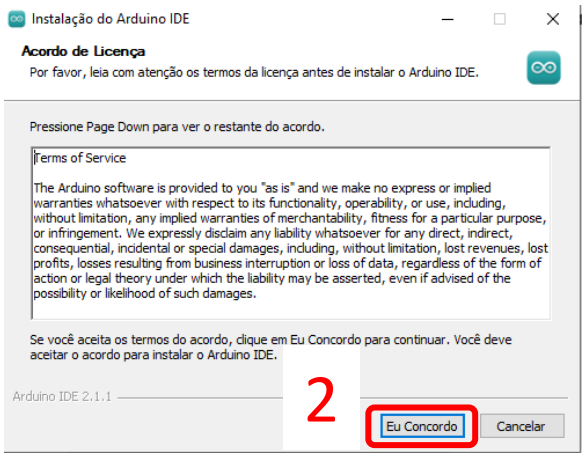
"JUST DOWNOAD".

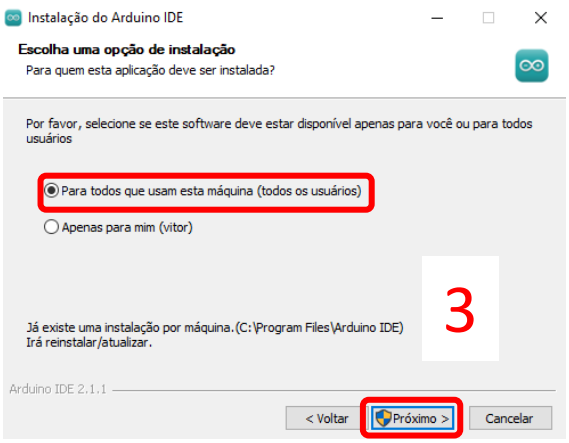
Após o download do arquivo, acesse a pasta em que foi salvo e execute o arquivo com um duplo clique sobre ele.

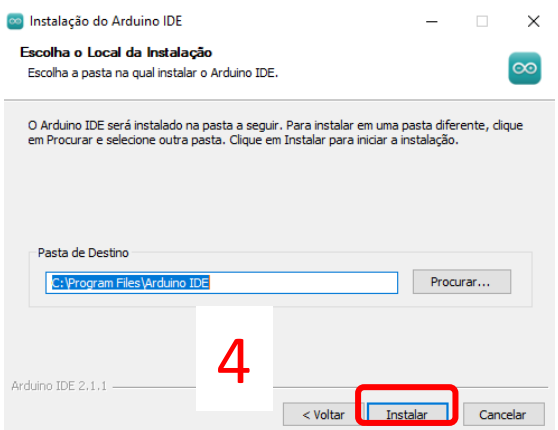


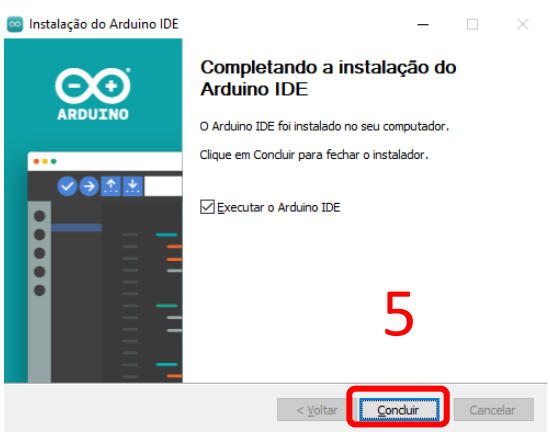
Siga a sequência de imagens para concluir a instalação:

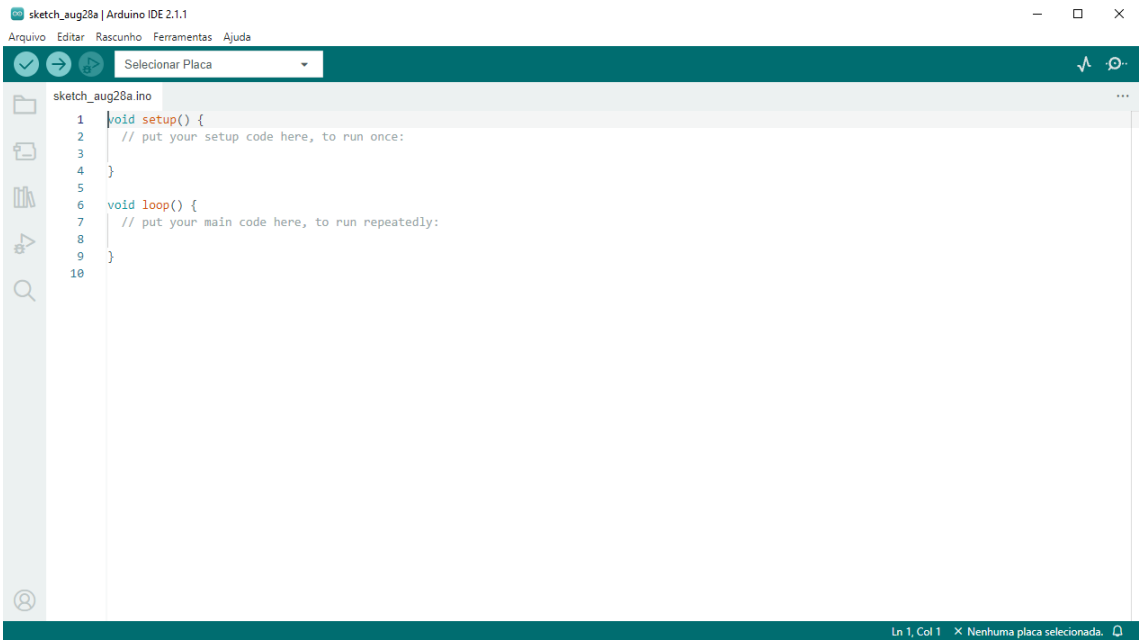












O seu IDE do Arduino está instalado, agora vamos para o processo de configuração.

Segundo passo

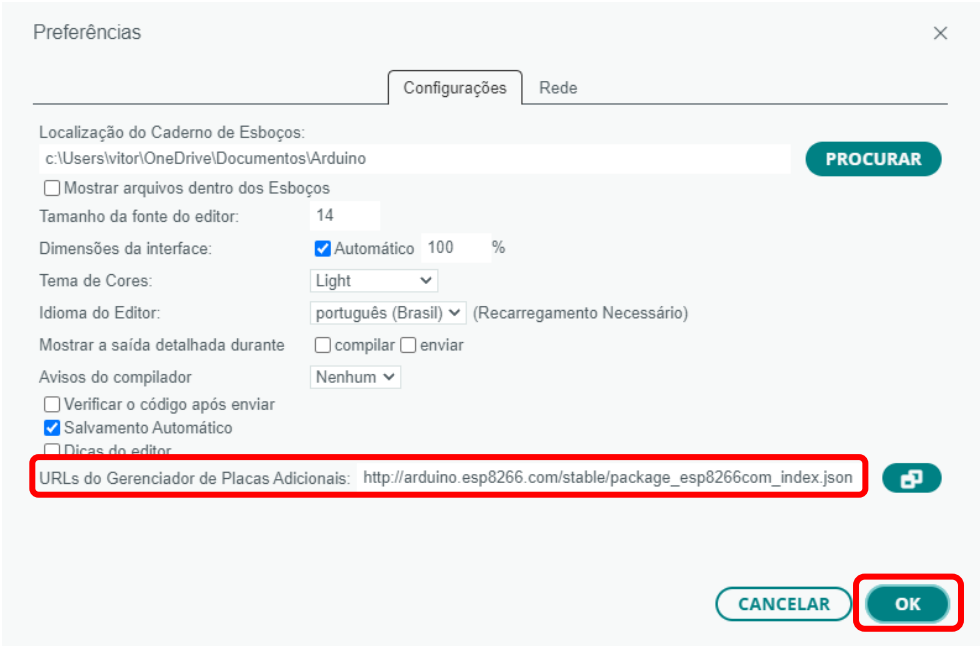
Agora que a nossa plataforma de desenvolvimento está instalada, vamos configurá-la para operar com a placa MobKits.

No IDE do Arduino acesse (canto superior esquerdo da tela) a opção “Arquivos”, depois “Preferências...”.

Copie a URL,

http://arduino.esp8266.com/stable/package_esp8266com_index.json

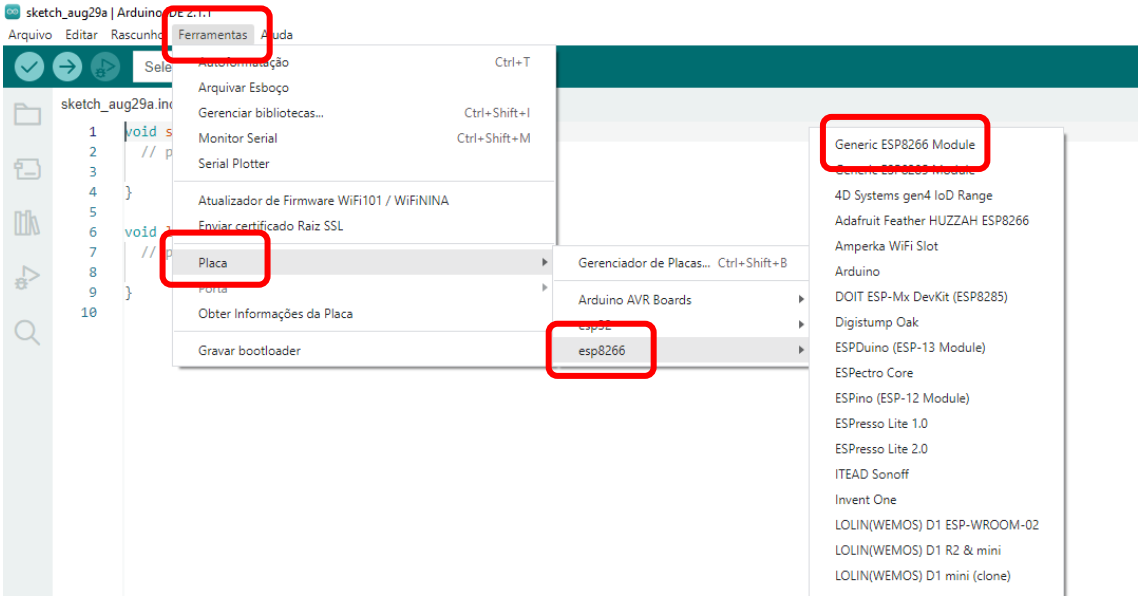
na tela de Preferências, cole (pressione Ctrl + v) no campo “URLs do Gerenciador de Placas Adicionais”, depois pressione OK.



Na tela principal (no canto esquerdo superior) clique na opção “Gerenciador de placas”

No campo de busca digite esp8266, logo abaixo deve aparecer a biblioteca para a nossa placa descrita como “esp8266 por ESP8266 community”, clique em “Instalar”. Para resetar o IDE do Arduino, após a instalação da biblioteca, feche o programa completamente e abra novamente.

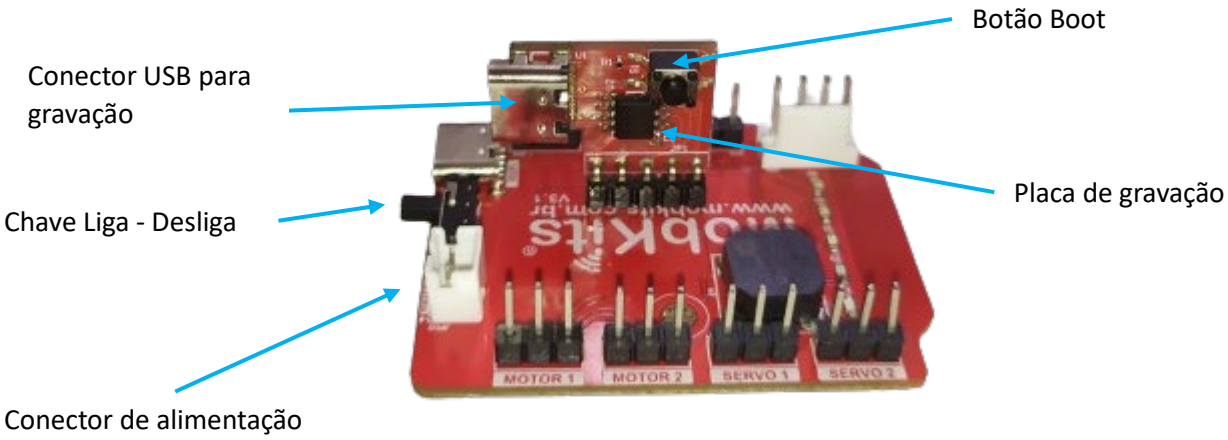
Agora vamos selecionar o microcontrolador da nossa plataforma, no IDE do Arduino clique em “Ferramentas”, → “Placa”, → “esp8266” e finalmente selecione a opção “Generic ESP8266 Module”.



Se você chegou nesse ponto podemos configurar a comunicação entre placa e computador.

Terceiro passo

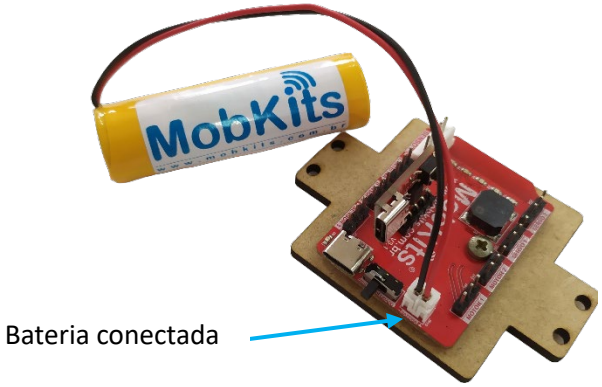
Identificando as conexões.



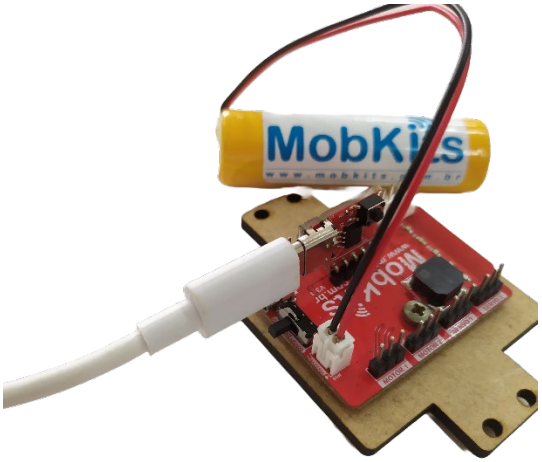
Quarto passo

Ligando o cabo USB e energizando a placa de controle MobKits para realizarmos a configuração que nos permitirá realizar a gravação do firmware (o firmware é o software que roda no microcontrolador após ser programado e compilado no IDE Arduino).

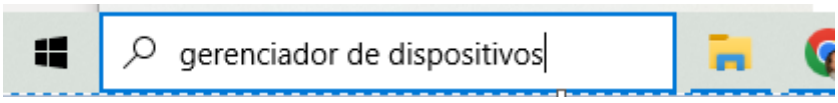
Conecte o cabo da bateria conforme imagem abaixo e certifique-se que a placa está desligada.



Conecte o cabo USB de acordo com a imagem abaixo.



Ligue a placa na chave liga-desliga e conecte a outra extremidade do cabo USB na porta USB de seu computador. O sistema operacional deve instalar um driver automaticamente. Para conferir se o cabo foi detectado, no Windows, no campo de pesquisa, digite “gerenciador de dispositivos” e clique ENTER.

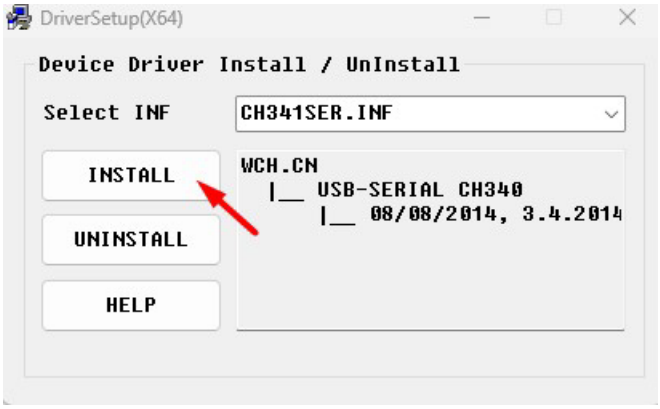


Na opção “Portas (COM e LPT), verifique se o cabo USB-Serial foi detectado e em qual COM foi mapeado.

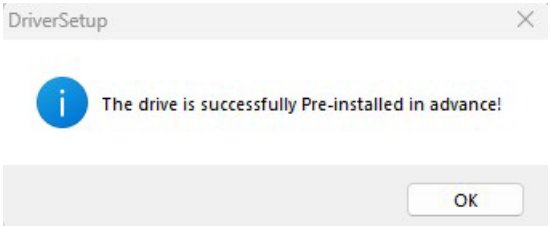
Caso o cabo não tenha sido detectado, baixe o driver do link (https://mobkits.com.br/pcb.html) e instale no seu computador.

Para instalar o Driver, após o download do arquivo, será necessário descompactar o arquivo. Acesse a pasta em que foi salvo e execute o arquivo com um duplo clique sobre ele.

Permita as alterações, após isso uma nova janela irá abrir. Clique no botão INSTALL.



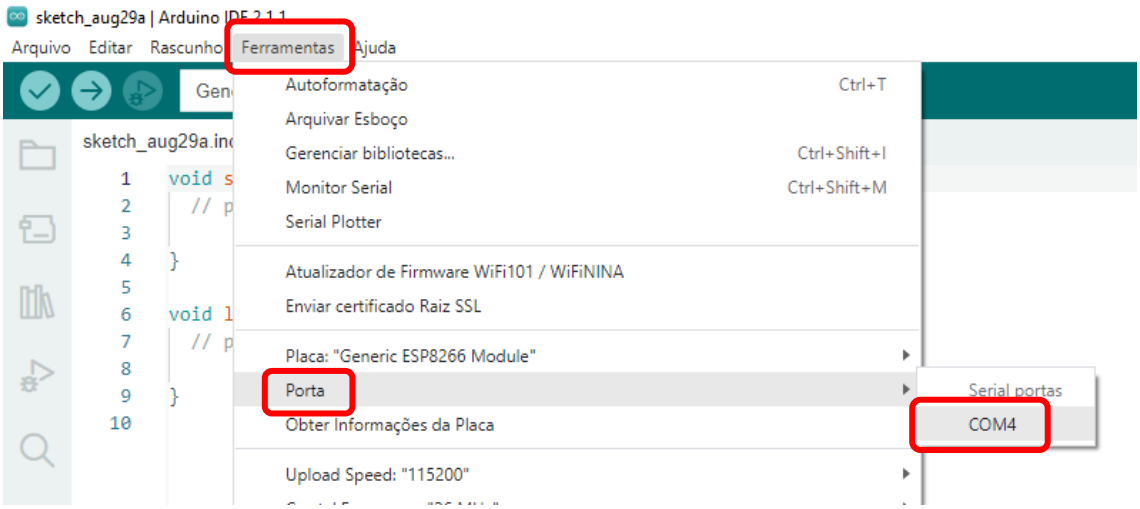
Ao final da instalação clique em OK se a seguinte mensagem aparecer.



Verifique novamente o Gerenciador de dispositivo, na opção “Portas (COM e LPT), verifique se o cabo USB-Serial foi detectado e em qual COM foi mapeado.

Quinto passo

Após configurar o cabo USB-Serial corretamente, com a placa energizada e com o cabo USB conectado conforme orientado no quarto passo, execute o IDE Arduino. Acesse a opção “Ferramentas” → “Portas”, finalmente selecione a COM mapeada para o seu cabo de gravação. É importante salientar que a numeração da porta COM pode variar a cada utilização.



Sexto passo

Gravando o código fonte (firmware) na placa Mobkits.

Para gravar um firmware na sua placa de controle é necessário ativar o modo de gravação. Com a placa desligada mantenha pressionado o Botão Boot enquanto liga a placa. Se o procedimento for feito corretamente os leds amarelo e vermelho deverão permanecer acesos.

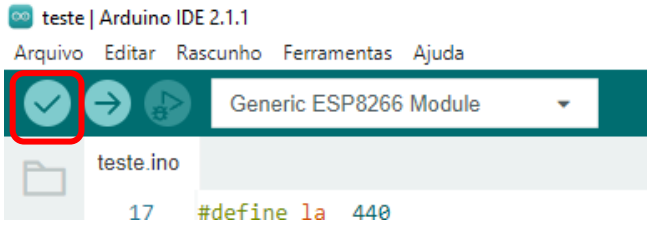
Atenção!! Se a bateria estiver fraca os leds ficarão com o brilho fraco e a porta COM não aparecerá.

Sempre que conectar a placa ao computador e acessar o IDE Arduino será necessário selecionar a porta conforme o quinto passo.

Vamos fornecer um código para ser utilizado como ponto de partida. Copie o código abaixo e cole na interface IDE do Arduino.

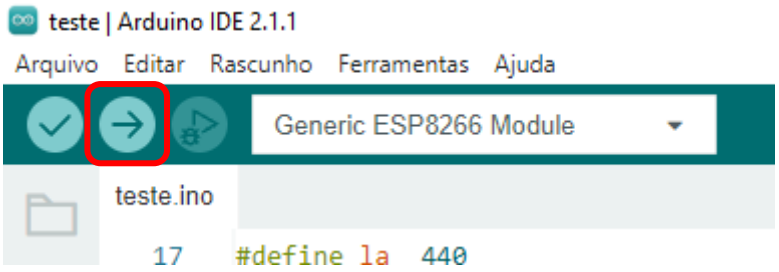
```
void setup() {

    pinMode(15, OUTPUT);
}
void loop() {
  digitalWrite(15, HIGH);
  delay(1000);
  digitalWrite(15, LOW);
  delay(1000);
}
```

Antes de gravar um firmware na sua placa de controle, é necessário que que o seu código esteja correto, sem erros lógicos, sintaxe ou digitação. É possível testar seu código fonte pressionando o botão “Verificar” no canto direito superior da interface IDE Arduino.

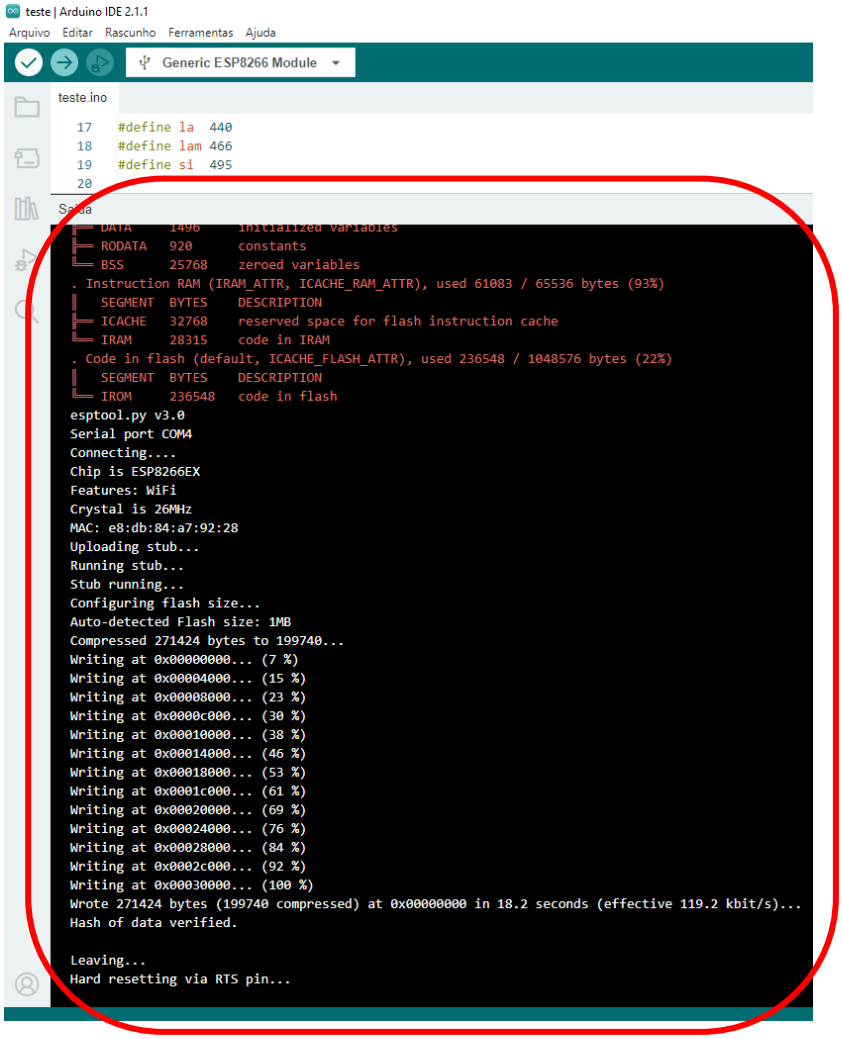


O seu código fonte será compilado e no final do processo não deve apresentar nenhuma mensagem de erro. Todas as mensagens podem ser verificadas na tela de “Saída”.

Vamos gravar o seu firmware na placa de controle! Após conferir seu código fonte e garantir que esteja livre de erros. Com a placa de controle em modo de gravação, no IDE Arduino pressione o botão de gravação “Enviar usando programador”.



A serial do computador deve sincronizar com a serial da placa MobKits e o programa iniciará a transferência. Você pode acompanhar a carga do firmware pela tela de “Saída”.



Após a carga do firmware, basta desligar a placa de controle e religar normalmente, sem pressionar o Botão Boot, assim poderá observar o comportamento da placa que foi alterado, pois agora o seu código está rodando.

Após a gravação, ao religar a placa, o led verde deve piscar com um intervalo de 1 segundo. Caso não esteja agindo desta forma repita com atenção o Sexto passo.

Funcionou!!!? Perfeito!

Agora que temos todo o nosso ferramental funcionando, vamos a programação!

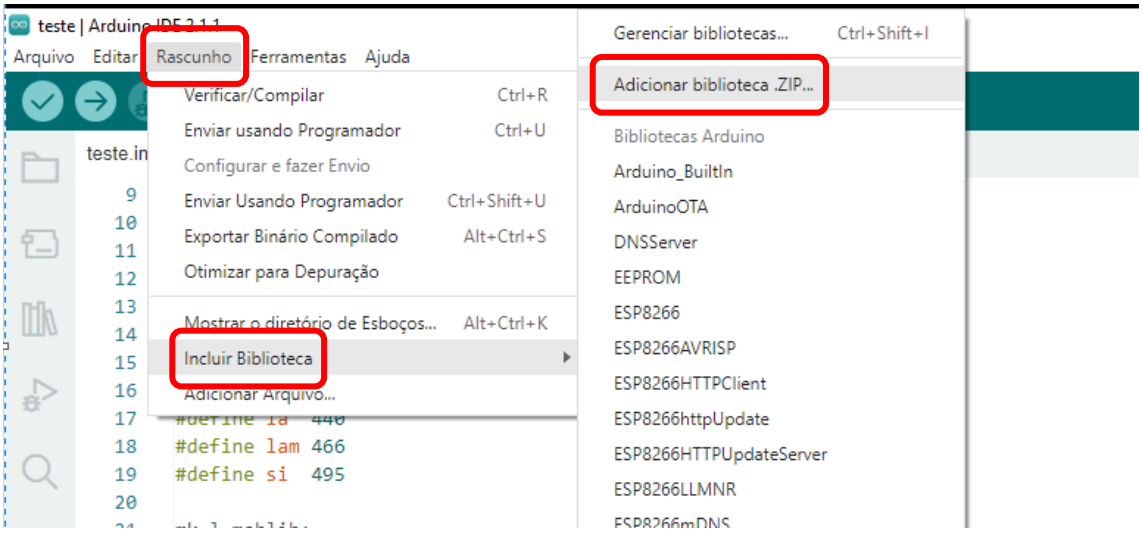
Biblioteca MobKits

Para facilitar a sua vida, a nossa equipe desenvolveu uma biblioteca para você acessar de forma prática todos os componentes da nossa plataforma.

Claro, fique a vontade para modificar as funções e criar novas, afinal de contas essa é a ideia, desenvolver e compartilhar!

Para instalar a biblioteca MobKits, primeiro baixe o arquivo “ZIP” da nossa página em (https://mobkits.com.br/pcb.html). Não descompacte o arquivo, ele será usado da forma em que está. Salve o arquivo em um diretório conhecido.

Na interface IDE do Arduino, acesse a opção “Rascunho” → “Incluir biblioteca” → “Adicionar biblioteca .ZIP...”.



Deve abrir um navegador do explorer para que você indique a pasta em que está salva a biblioteca “mobkits_lib.zip”. O arquivo será importado e instalado.

Agora estamos prontos para desenvolver e explorar todos os recursos que a plataforma MobKits tem a nos oferecer, na sequência descrevemos todas as funções, apresentando exemplos e desafios.

Como acesso as funções na biblioteca?

Todas as funções foram definidas dentro de uma classe, para facilitar o acesso. Dessa forma, no início do nosso projeto devemos incluir a chamada da nossa biblioteca “mobkits_lib.h” e definir o construtor da nossa classe, veja a descrição abaixo:

```
#include <mobkits_lib.h>

mk_l moblib;
```

Onde “mk_l” é o nome da nossa classe e “moblib” o construtor que dará acesso as funções dentro da classe.

Função piscaled

```
//-----------------------------------------------
// Função - piscaled
// Parâmetros da função
// Cor do led (verde, amarelo ou vermelho)
// Tempo ligado e desligado em (ms)
// 1000ms = 1segundo
//-----------------------------------------------
void piscaled (int cor_led, long tempo);
```

Com esta função é possível controlar de forma independente os 3 LED's (verde, amarelo e vermelho) presentes na plataforma. Passamos para a função a cor do led que desejamos controlar e o tempo de intermitência do pisca-pisca.

Exemplo:

```
#include <mobkits_lib.h>


mk_l moblib;

void setup() {

}

void loop() {

  moblib.piscaled(verde, 200); // pisca led verde a cada 200 ms
  moblib.piscaled(amarelo, 200); // pisca led amarelo a cada 200 ms
  moblib.piscaled(vermelho, 200); // pisca led vermelho a cada 200 ms

}
```

DESAFIO

Altere o campo "tempo" da função e verifique o que ocorre com os LED's na medida em que o tempo aumenta e diminui, qual a sua percepção?

Função ligaled

```
//-----------------------------------------------
// Função - ligaled
// Parâmetros da função
// Cor do led (verde, amarelo ou vermelho)
// estado (ligado ou desligado)
//-----------------------------------------------
void ligaled(int cor_led, bool estado);
```

Com esta função é possível controlar cada LED da plataforma, ligando e desligando da forma que lhe convir. Passamos para a função a cor do LED que desejamos controlar (verde, amarelo ou vermelho) e o estado (ligado, desligado).

Exemplo:

```
#include <mobkits_lib.h>

mk_l moblib;

void setup() {

}

void loop() {

  moblib.ligaled(verde, ligado); // liga o led verde
  delay(100); // tempo de espera em ms
  moblib.ligaled(amarelo, ligado); // liga o led amarelo
  delay(100);
  moblib.ligaled(vermelho, ligado); // liga o led vermelho
  delay(100);
  moblib.ligaled(verde, desligado); // desliga o led verde
  delay (100);
  moblib.ligaled(amarelo, desligado); // desliga o led amarelo
  delay(100);
  moblib.ligaled(vermelho, desligado); // desliga o led vermelho
  delay(100);
}
```

DESAFIO

Utilizando duas placas MobKits, ou mais, monte um sistema de semáforo, que funcione de forma sincronizada ao ligar todas as placas ao mesmo tempo.

Função dimmerled

```
//-----------------------------------------------
// Função - dimmerled
// Parâmetros da função
// Cor do led (verde, amarelo ou vermelho)
// estado (ligado ou desligado)
// tempo (em ms)
//-----------------------------------------------
void dimmerled(int cor_led, bool estado, int tempo);
```

Com esta função é possível ligar e desligar o LED de forma gradual, alterando a tensão média de alimentação. Dessa forma, conseguimos um efeito de esmorecimento. Passamos para a função a cor do LED (verde, amarelo ou vermelho), o estado (ligado e desligado) e o tempo expresso em ms.

Exemplo:

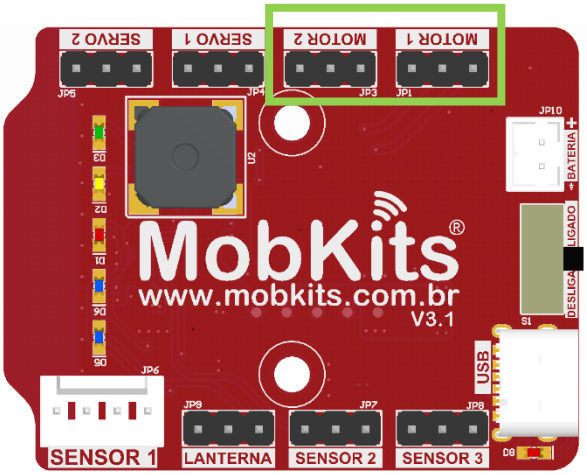
```
#include <mobkits_lib.h>

mk_l moblib;

void setup() {

}

void loop() {
  moblib.dimmerled(verde, ligado, 1);
  moblib.dimmerled(verde, desligado, 1);
  moblib.dimmerled(amarelo, ligado, 1);
  moblib.dimmerled(amarelo, desligado, 1);
  moblib.dimmerled(vermelho, ligado, 1);
  moblib.dimmerled(vermelho, desligado, 1);
}
```

DESAFIO

Altere a sequência do código de forma que os LED's liguem verde, amarelo e vermelho, permaneçam todos ligados e desliguem ao contrário, vermelho, amarelo e verde.

Função motor

```
//-------------------------------------------------
// Função - motor
// Parâmetros da função
// motor (motor1 ou motor 2)
// estado (ligado ou desligado)
// sentido (frente ou tras)
// Velocidade (minimo 10 , máximo 100)
//-------------------------------------------------
void motor(bool motor, bool estado, bool sentido, int velocidade);
```

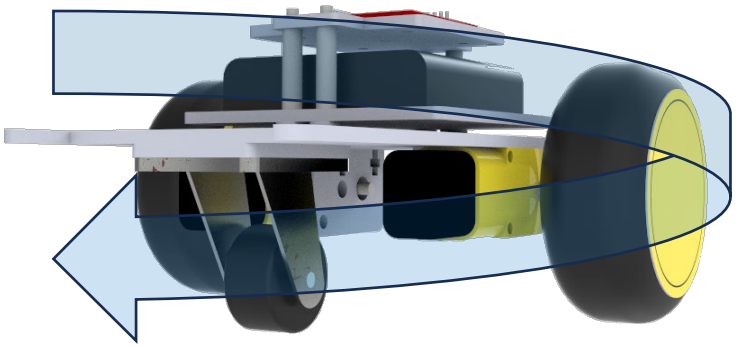


Com esta função é possível controlar as duas saídas de motores DC presentes na plataforma, “Motor 1” e “Motor 2”, de forma independente. Passamos para a função os seguintes parâmetros, inicialmente qual motor atuar, (motor1 ou motor2), o estado (ligado ou desligado), o sentido de rotação (frente ou trás) e finalmente a velocidade (mínimo de 10 e máximo de 100).

Exemplo:

```
#include <mobkits_lib.h>

mk_l moblib;

void setup() {

}

void loop() {
  moblib.motor(motor1, ligado, frente, 100);
  moblib.motor(motor2, ligado, tras, 100);
}
```

No exemplo acima, caso os motores estejam montados no carrinho básico, o carrinho vai girar com velocidade máxima no seu próprio eixo.

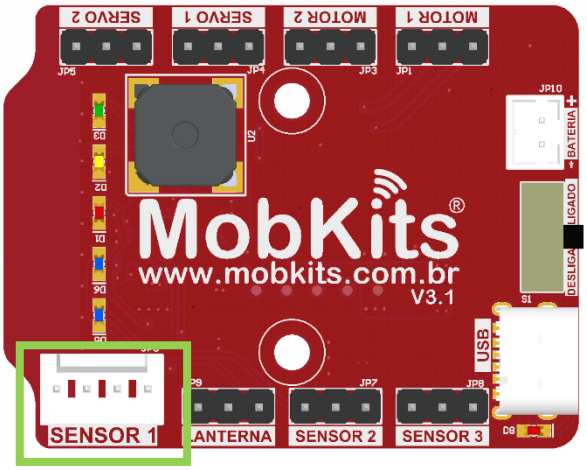


DESAFIO

Monte um percurso com uma largada e uma chegada, programe os movimentos do carrinho através dos motores para que de forma autônoma ele percorra o caminho corretamente.

Função Ultrassom

```
//------------------------------------------------
// Função - Ultrassom HC-SR04
// Parâmetros da função
// A função retorna a distância em cm
//------------------------------------------------
int ultrassom(void);
```

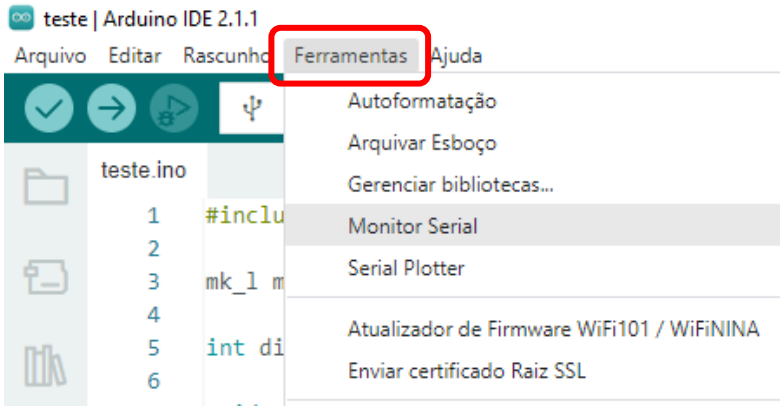


O ultrassom utilizado é o modelo HC-SR04, pode ser ligado na conexão indicada acima. A função “ultrassom” retorna uma variável do tipo inteira que representa a leitura instantânea do sensor em cm.

Exemplo:

```
#include <mobkits_lib.h>

mk_l moblib;

int distancia = 0;

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

void loop() {
  distancia = moblib.ultrassom();
  Serial.println(distancia);
  delay(1000);
}
```

Através do cabo de gravação, podemos acessar a serial do microcontrolador e usá-la em algumas aplicações ou apenas para debug do nosso código. No caso do exemplo acima, inicializamos a serial na velocidade de 9600bps, definimos uma variável do tipo inteira para receber o retorno da função ultrassom e escrevemos o resultado na serial a cada 1 segundo. Para visualizar o resultado no seu computador, o cabo serial (cabo de gravação) deve estar conectado na plataforma, na IDE do Arduino, abra o monitor serial e configure a velocidade de comunicação para 9600bps. A cada 1 segundo deve aparecer a informação de distância percebida pelo sensor ultrassônico.

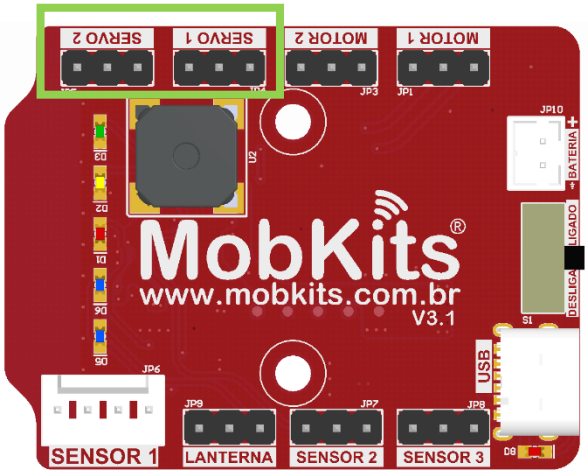


DESAFIO

Altere o exemplo anterior e implemente uma média de “n” medidas, onde n inicialmente seja 1, depois 100 e finalmente 1000. Analise o que acontece com os resultados médios na medida em que aumentam e diminuem a média?

Função Servo motor

```
//-------------------------------------------------
// Função - Servo motor
// Parâmetros da função
// estado (ligado ou desligado)
// angulo (0 a 180 graus)
// saida (servo1 ou servo2)
//-------------------------------------------------
void servo(bool estado, int angulo, int servo);
```



Com esta função é possível controlar as duas saídas PWM para os servos motores “Servo 1” e “Servo 2” de forma independente. Passamos para a função o estado (ligado ou desligado), o ângulo de rotação de (0 a 180 graus) e o servo que queremos controlar (servo1 ou servo2).

Exemplo:

```
#include <mobkits_lib.h>

mk_l moblib;

void setup() {

}

void loop() {
  moblib.servo(ligado, 0, servo1);
  delay(2000);
  moblib.servo(ligado, 180, servo1);
  delay(2000);
}
```

No exemplo acima, o “servo 1” é movimentado 180 graus a cada dois segundos.

DESAFIO

Movimente o servo 1 seguido pelo servo 2 de 0 a 180 graus, com um intervalo de tempo de meio segundo entre cada grau.

Função Buzzer

```
//-----------------------------------------------
// Função - Buzzer notas musicais
// Parâmetros da função
// tempo (tempo da nota em ms)
// nota (do, dom, re, rem, mi, fa, fam, sol, solm,
// la, lam, si)
//-----------------------------------------------
void buzznota(int nota, int tempo);
```

O buzzer é um alto falante com saída de som modulável, ou seja, podemos alterar a frequência de forma a modificar o som. Passamos para a função dois parâmetros, o "tempo" de perdura da nota (em ms) e a "nota" que desejamos tocar (do, dom, re, rem, mi, fa, fam, sol, solm, la, lam e si), as notas terminada com "m" indicam uma oitava acima.

Exemplo:

```
#include <mobkits_lib.h>

mk_l moblib;

int tempo_1 = 400;
int tempo_2 = 100;

void setup() {

}

void loop() {
  moblib.buzznota(mi, tempo_1);
  delay(tempo_2);
  moblib.buzznota(mi, tempo_1);
  delay(tempo_2);
  moblib.buzznota(fa, tempo_1);
  delay(tempo_2);
  moblib.buzznota(sol, tempo_1);
  delay(tempo_2);
  moblib.buzznota(sol, tempo_1);
  delay(tempo_2);
  moblib.buzznota(fa, tempo_1);
  delay(tempo_2);
  moblib.buzznota(mi, tempo_1);
  delay(tempo_2);
  moblib.buzznota(re, tempo_1);
  delay(tempo_2);
  moblib.buzznota(do, tempo_1);
  delay(tempo_2);
  moblib.buzznota(do, tempo_1);
  delay(tempo_2);
  moblib.buzznota(re, tempo_1);
```

```
  delay(tempo_2);
  moblib.buzznota(mi, tempo_1);
  delay(tempo_2);
  moblib.buzznota(mi, tempo_1);
  delay(tempo_2);
  moblib.buzznota(re, tempo_1);
  delay(tempo_2);
  moblib.buzznota(re, tempo_1);
  delay(tempo_2);
}
```

No exemplo acima, podemos encadear as notas musicais, mudando o tempo de perduração e o intervalo entre notas, criando padrões musicais. Também podemos modificar o código de forma mais otimizada, para facilitar a criação das nossas músicas. O exemplo abaixo executa a mesma música do exemplo acima (Ode a Alegria – Beethoven).

Exemplo:

```
#include <mobkits_lib.h>

mk_l moblib;

int tempo_1 = 400;
int tempo_2 = 100;
int notas[15] = {mi, mi, fa, sol, sol, fa, mi, re, do, do, re, mi, mi,
re, re};
int i;

void setup() {

}

void loop() {
  for(i=0; i<15; i++){
    moblib.buzznota(notas[i], tempo_1);
    delay(tempo_2);
  }
}
```

DESAFIO

Utilizando a função “buzznota”, crie sua própria música, dando personalidade a sua aplicação eletrônica!

Função Buzzer Sirene

```
//-----------------------------------------------
// Função - Buzzer sirene
// Parâmetros da função
// estado (liga ou desliga)
//-----------------------------------------------
void buzzsirene(bool estado);
```

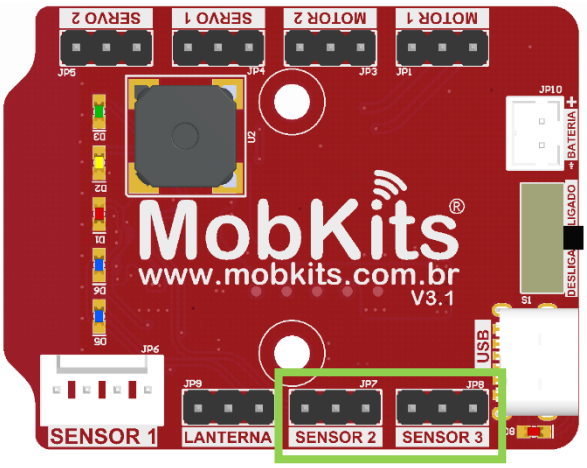
Esta função simula uma sirene e pode ser usada em vários contextos, sistemas de alarme, avisos de advertência ou apenas para diversão. Passamos para a função apenas o estado de (ligado ou desligado).

Exemplo:

```
#include <mobkits_lib.h>

mk_l moblib;

void setup() {

}

void loop() {
  moblib.buzzsirene(ligado);
}
```

Função Sensor

```
//-------------------------------------------------
// Função - Sensores
// Parâmetros da função
// sensor (sensor2 ou sensor3)
// IR --> Infrared, LDR --> Luminosidade
// retorna 1 na detecção e 0 sem detecção booleano
//-------------------------------------------------
bool sensor(int sensor);
```



Na plataforma Mobkits estão presentes duas entradas independentes para sensores. Podem ser utilizados sensores infravermelhos ou sensores de luminosidade. Passamos para a função a entrada a ser monitorada (sensor2 ou sensor3), a função retorna um booleano (1 ou 0),

Exemplo:

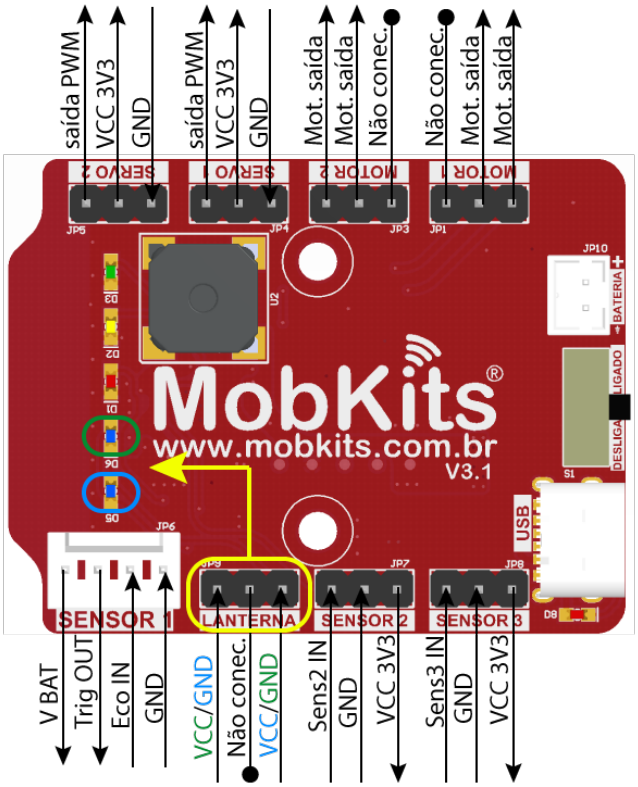
```
#include <mobkits_lib.h>

mk_l moblib;

bool alarme = 0;

void setup() {

}

void loop() {
  alarme = moblib.sensor(sensor2);
  if (alarme)
      moblib.buzzsirene(ligado);
  else
      moblib.buzzsirene(desligado);
}
```

No exemplo acima foi implementado um sistema de alarme simples. Na entrada sensor2, foi conectado um sensor infravermelho. Quando o sensor percebe algum objeto a sua frente, a sirene é ligada, indicando presença.

DESAFIO

Utilizando a função “sensor” implemente um sistema de contagem de entrada e saída de pessoas de uma loja. Caso a capacidade máxima da loja seja atingida, deve soar uma sirene.

Mapa elétrico da plataforma MobKits



IMPORTANTE:

A saída do sensor 1 (Trig OUT) é compartilhada com a entrada do sensor 2 (Sens2 IN);

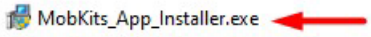
A entrada do sensor 1 (Eco IN) é compartilhada com a entrada do sensor 3 (Sens3 IN);

A entrada Lanterna é independente do microcontrolador, se alimentar os pinos com até +5VDC em um sentido liga um dos LED's, invertendo a alimentação liga o outro LED.

Retornando a Placa Pré-Programada

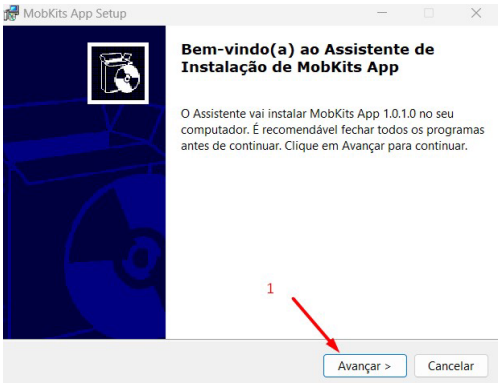
Instalando o aplicativo Windows

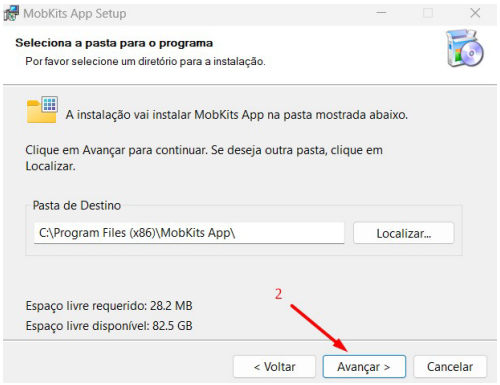
Para instalar o aplicativo Windows, primeiro baixe o arquivo “.ZIP” da nossa página em (https://mobkits.com.br/pcb.html). Descompacte o arquivo e execute com duplo clique.

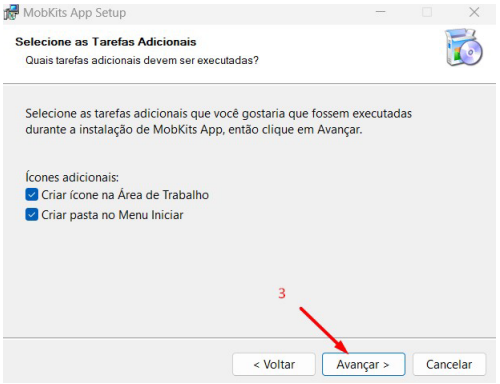


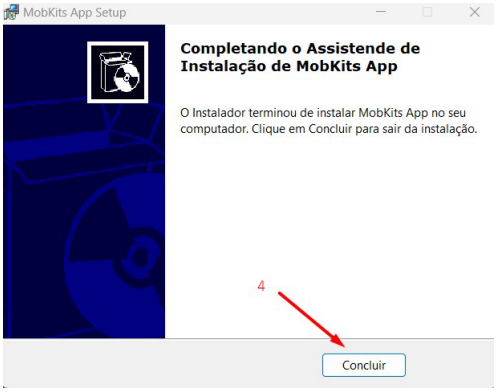
Caso o sistema operacional impeça a instalação ou apresente alguma mensagem informando que o aplicativo deseja realizar alterações basta permitir.

Após a permissão siga o passo a passo abaixo:



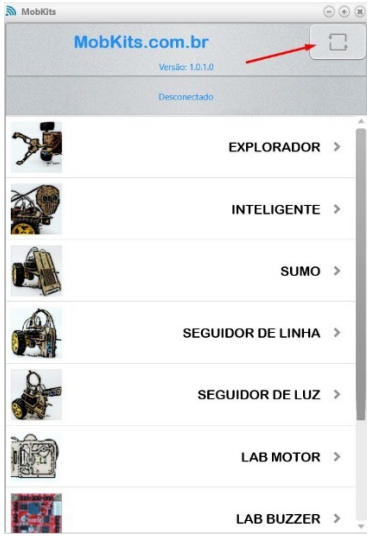






Na sua área de trabalho o ícone abaixo estará disponível.

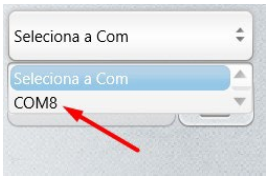
Neste passo a bateria precisa estar carregada e conectada na placa. Conecte a placa ao computador com o cabo USB-C, ligue a placa e abra o aplicativo. Clique no botão indicado na imagem abaixo.



Abrirá uma nova janela do aplicativo.

Nesta janela clique em Selecione a Com, abrirá uma lista com uma ou mais COMs selecione a COM relativa à conexão da sua placa. Caso não saiba qual a correta, verifique o quarto passo da Interface de Programação neste manual.

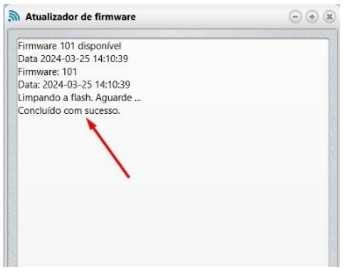
Obs.: o número que aparece ao lado da COM pode variar.



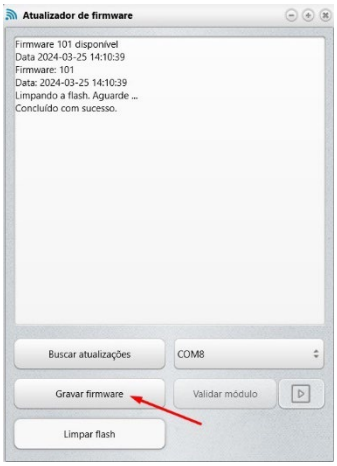
Após selecionar a Porta COM coloque a placa no modo de gravação: desligue a placa, mantenha pressionando o botão Boot e ligue a placa novamente. Os leds amarelo e vermelho devem estar acesos.

Nesta condição clique em Limpar flash, vamos tirar da memória todo o código gravado anteriormente.

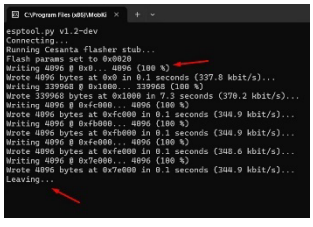
Uma nova janela abrirá e fechará rapidamente, após isso teremos a confirmação conforme imagem abaixo.



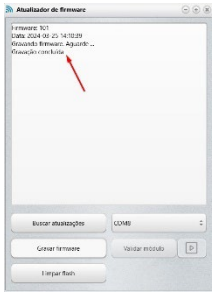
Desligue a placa e coloque no modo de gravação novamente. Confira os leds amarelo e vermelho acesos e clique em Gravar firmware.



Uma nova janela abrirá demonstrando a evolução da gravação e se fechará após o término da gravação.



Após isso teremos a confirmação conforme imagem abaixo:



Após a gravação precisamos validar nossa placa, clique no botão indicado na imagem abaixo.

Confira a mensagem conforme a imagem abaixo.

Clique em Validar módulo.

Confira a mensagem conforme a imagem abaixo. Se a placa foi validada com sucesso está tudo pronto. A programação original de fábrica foi restabelecida.

Desafios Avançados

DESAFIO 1

Automação residencial inteligente

Neste desafio será proposto a automação de uma cortina ou veneziana, o objetivo é aproveitar a luz do dia para iluminar a casa. O sistema deve monitorar a luminosidade externa da residência, existindo luz, a cortina ou veneziana deve ser aberta, na falta de luz, fechada.

DESAFIO 2

Sistema de detecção de obstáculos para deficientes visuais

Neste desafio será proposto o desenvolvimento de um sistema de detecção de obstáculos para deficientes visuais. Normalmente pessoas com esse tipo de deficiência, utilizam bengalas e até animais treinados para guiá-los. Mas os obstáculos percebidos são normalmente no nível do solo. A ideia deste projeto é perceber obstáculos mais elevados, e avisar de alguma forma para evitar colisões.

DESAFIO 3

Automação industrial

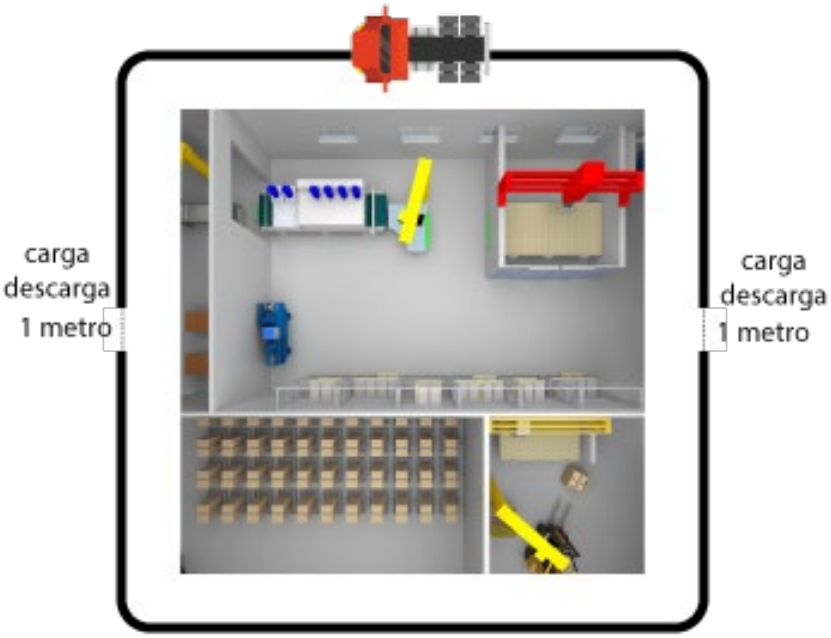
Neste desafio será proposto a automação de uma esteira, ela deve transportar caixas que são contadas em um ponto. Após “n” unidades, a esteira para por 10 segundos para a coleta das “n” caixas. Ao final da pausa o sistema volta a funcionar de forma recorrente.

DESAFIO 4

Veículo inteligente

Neste desafio será proposto a automação de um veículo inteligente, ele opera em uma fábrica e faz um caminho conhecido, seguindo um sistema de orientação baseado em uma referência pintada ao longo do seu caminho (linha). O veículo para em 2 postos também conhecidos para carga e descarga, a indicação de parada ocorre pela detecção de intermitência da linha em um espaço de 1 metro. Ao chegar ao posto de carga/descarga e o veículo estiver pronto para ser carregado ou descarregado, deve soar uma sirene. O veículo deve permanecer parado por 5 minutos, após esse tempo, deve soar novamente a sirene (com outro padrão de tom) indicando que o veículo

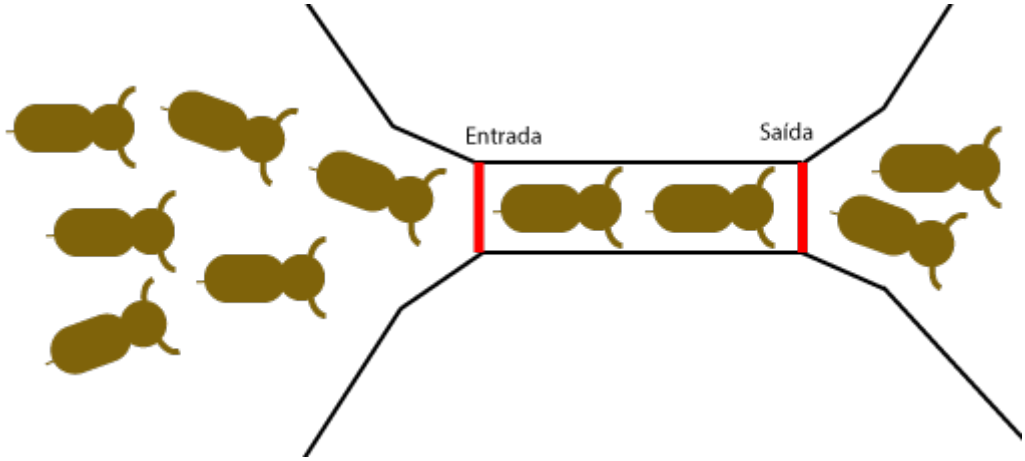
iniciará a jornada, transcorrido 1 minuto da sirene, o veículo novamente entra em movimento.



DESAFIO 5

Automação no campo

Neste desafio será proposto a criação de um sistema para controle de gado. Em uma fazenda, existe uma porteira de entrada controlada, onde passa apenas um boi por vez. O sistema conta a passagem de “n” animais e após alcançar esse limite não abre mais. Algum processo é feito nessa área de encarceramento e após 1 minuto, é aberto a porteira de saída. Essa porteira só fecha após o esvaziamento total dos bois. Para acelerar o processo de esvaziamento, também é permitido o uso de sirenes.



DESAFIO 6

Controle de nível d'água

Neste desafio será proposto a criação de um sistema para controle de nível de uma caixa d´água. É comum principalmente em condomínios falta d´água, seja por problemas de vazamento ou manutenções na rede de fornecimento. Esse sistema deve monitorar o nível mínimo e máximo da caixa e avisar através dos LEDs quando algum desses níveis for crítico. Além das indicações luminosas, também é necessário um sistema de alarme para chamar atenção.

DESAFIO 7

Controle de estacionamento

Neste desafio será proposto a criação de um sistema para controle de quantidade de veículos em um estacionamento. A capacidade máxima é de 100 veículos. O sistema controla uma cancela que dá acesso ao estacionamento, a entrada e saída é pela mesma cancela, ao atingir a capacidade máxima, não é permitida a entrada de outro veículo. Também deve haver uma indicação luminosa de que ainda existem vagas e de capacidade máxima atingida.

DESAFIO 8

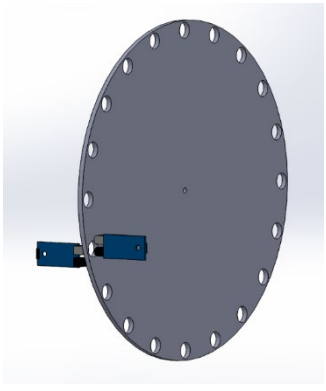
Alimentador para Pets

Neste desafio será proposto o desenvolvimento de um alimentador automático de ração. O sistema deve liberar a ração 2 vezes por dia e na mesma quantidade.

DESAFIO 9

Encoder

Neste desafio será proposto o desenvolvimento de um sistema de encoder. O encoder tem várias aplicações, como contagem ou controle de velocidade e distância. Nesta aplicação, podemos usar os infrared's em uma configuração específica e uma roda dentada para desenvolvimento do sistema.



Em um dos sensores, anulamos o emissor, e no outro o receptor. Para anular será necessário envolver com fita isolante o led transparente em uma das placas, na outra envolver com fita isolante o led preto. Então teremos uma fonte emissora de um lado e receptora do outro. Como a roda dentada tem furos espaçados e conhecida a distância entre furos, podemos implementar um simples contador de pulso, ou medir o tempo entre pulsos. Baseados na física de forma simples, imaginando um movimento contínuo sem aceleração temos que **velocidade = distância/tempo**. Logo podemos calcular a velocidade e medir distâncias.

DESAFIO 10

Dispenser de papel inteligente

Neste desafio será proposto o desenvolvimento de um dispenser de papel inteligente. Durante a pandemia de COVID, aprendemos que lavar as mãos é um hábito fundamental para evitar o contágio. Mas após lavar as mãos é necessário secá-las, para isso usamos lenços de papel que retiramos manualmente de algum dispenser. O dispenser inteligente, deve detectar a mão da pessoa que vai retirar o papel e o colocar para fora do dispenser, de forma automática sempre a mesma quantidade de papel.