ESP32 – Reconhecimento de voz com celular

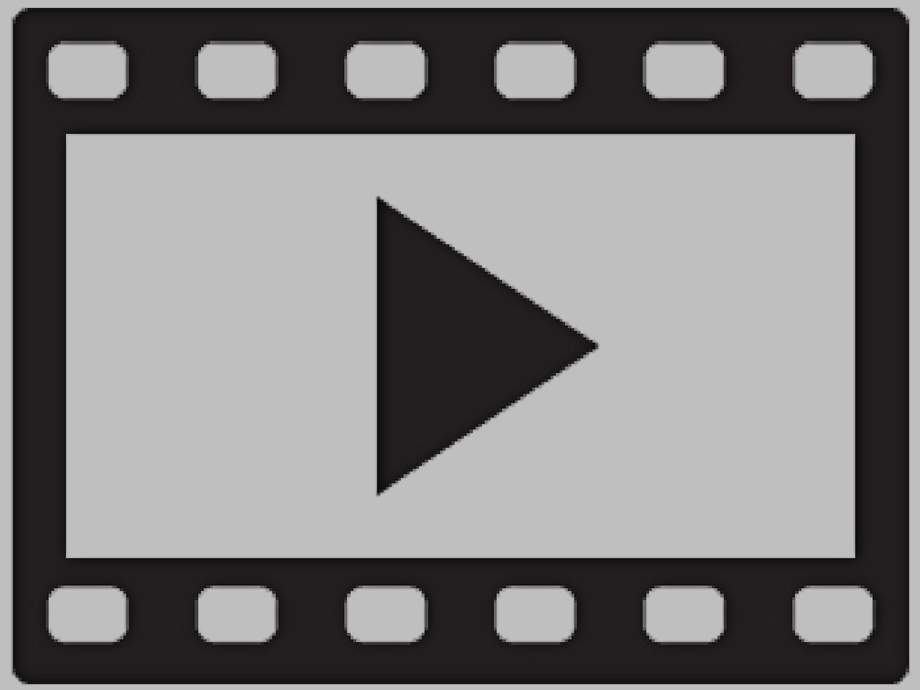
- AppInventor

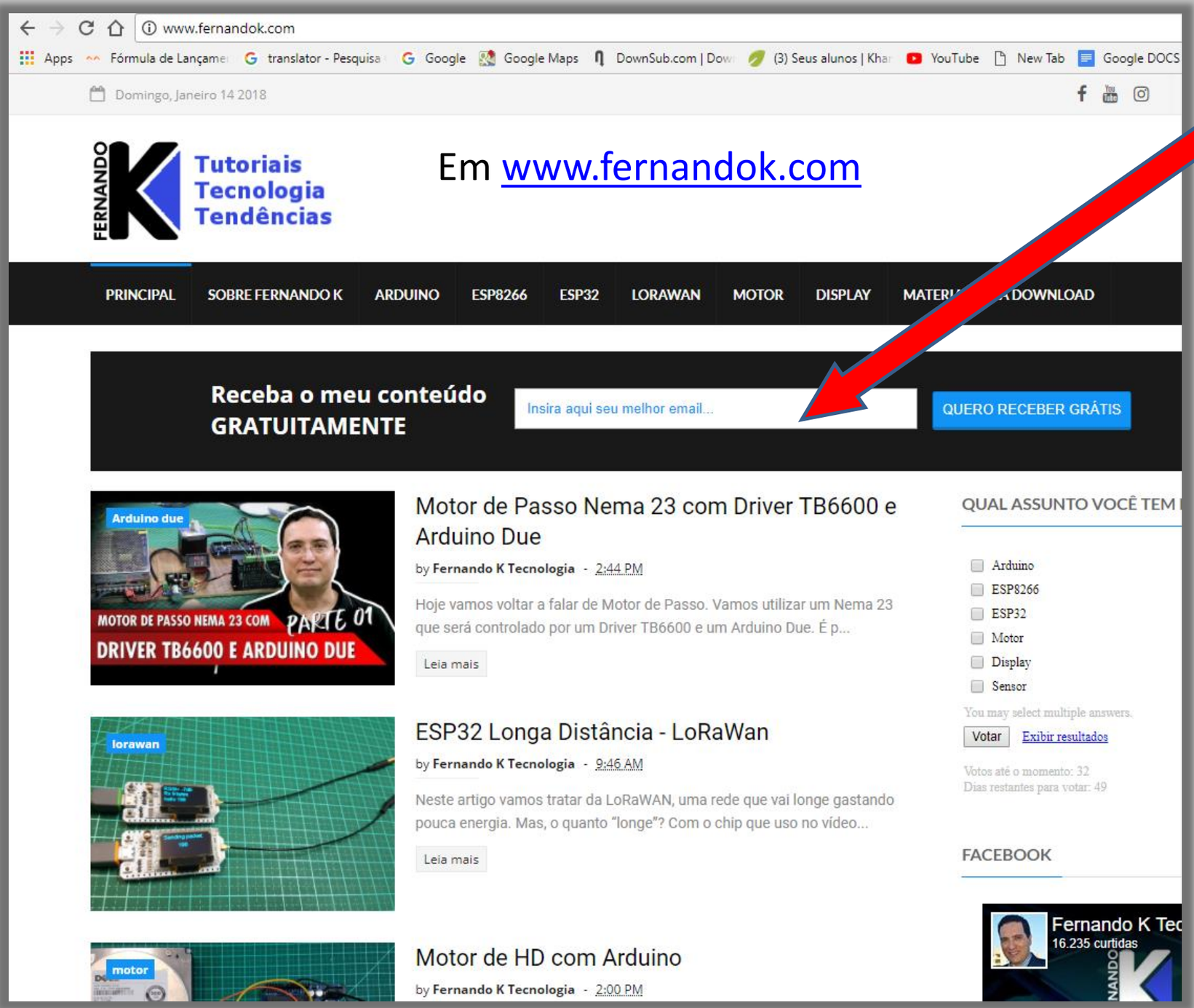
Por Fernando Koyanagi

Intenção dessa aula

1. Conectar ESP32 com celular
2. Utilizar reconhecimento de voz da Google
3. Entender código .ino do ESP32
4. Entender programa feito pelo AppInventor

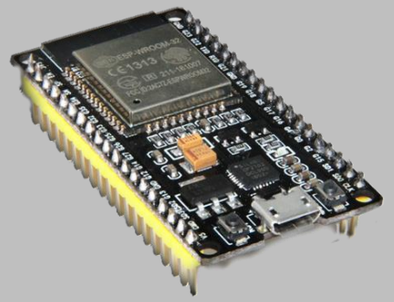
Demonstração




www.fernandok.com
Apps
Fórmula de Lançame
translator - Pesquisa
Google
Google Maps
DownSub.com | Dow
(3) Seus alunos | Kha
YouTube
New Tab
Google DOCS
Domingo, Janeiro 14 2018
FERNANDO K
Tutoriais
Tecnologia
Tendências
Em www.fernandok.com
PRINCIPAL
SOBRE FERNANDO K
ARDUINO
ESP8266
ESP32
LORAWAN
MOTOR
DISPLAY
DOWNLOAD
Receba o meu conteúdo GRATUITAMENTE
Insira aqui seu melhor email...
QUERO RECEBER GRÁTIS
Arduino due
MOTOR DE PASSO NEMA 23 COM DRIVER TB6600 E ARDUINO DUE
PARTE 01
Motor de Passo Nema 23 com Driver TB6600 e Arduino Due
by Fernando K Tecnologia - 2:44 PM
Hoje vamos voltar a falar de Motor de Passo. Vamos utilizar um Nema 23 que será controlado por um Driver TB6600 e um Arduino Due. É p...
Leia mais
lorawan
ESP32 Longa Distância - LoRaWan
by Fernando K Tecnologia - 9:46 AM
Neste artigo vamos tratar da LoRaWAN, uma rede que vai longe gastando pouca energia. Mas, o quanto "longe"? Com o chip que uso no vídeo...
Leia mais
motor
Motor de HD com Arduino
by Fernando K Tecnologia - 2:00 PM
QUAL ASSUNTO VOCÊ TEM
Arduino
ESP8266
ESP32
Motor
Display
Sensor
You may select multiple answers.
Votar
Exibir resultados
Votos até o momento: 32
Dias restantes para votar: 49
FACEBOOK
Fernando K Tec
16.235 curtidas

Seu e-mail

IDEs utilizadas

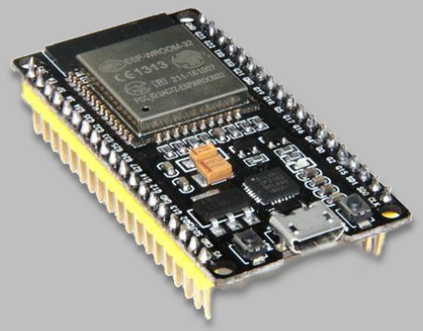


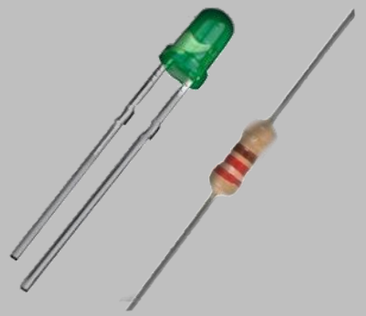
(Ferramenta online)

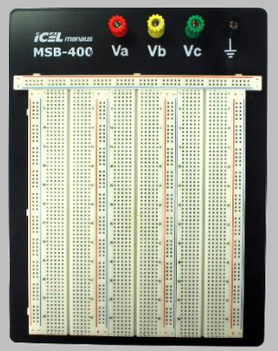
http://ai2.appinventor.mit.edu

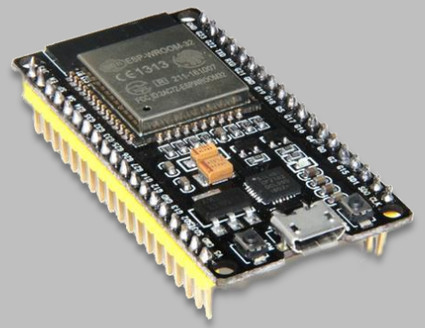
Recursos usados

- ESP32
- Smartphone
- Led
- Resistor de 220ohm
- Protoboard







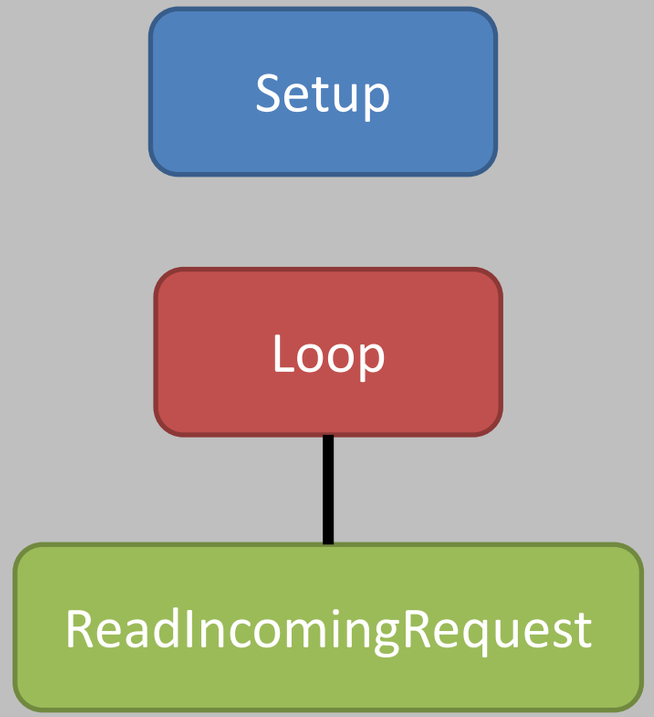


Código ESP32

ESP32 [Organização do código]

```
//Variáveis, definições e includes
//(...)

//função usada para a leitura do request
String ReadIncomingRequest()
{
   //(...)
}

void setup()
{
   //(...)
}

void loop()
{
   //(...)

   //obtém request utilizando a função local ReadIncomingRequest
   ClientRequest = (ReadIncomingRequest());

   //(...)
}
```

Além das funções “setup()” e “Loop()” foi criada uma chamada “ReadIncomingRequest()” para fins de melhor organização do código.



ESP32 [Definições e Variáveis]

```
//lib necessária para conectar o wifi
#include <WiFi.h>

//led conectado no pino 23
#define ledVerde 23

//mensagem enviada pelo client (aplicativo)
String ClientRequest;

//ip estático, o mesmo deve ser usado no app do smartphone
IPAddress staticIP(192,168,2,120);
//gateway, deixe aqui o gateway da rede em que está conectado
IPAddress gateway(192,168,2,255);
//máscara, deixe aqui a máscara da rede em que está conectado
IPAddress subnet(255,255,255,0);

//objeto do servidor, porta default 80
WiFiServer server(80);

//objeto do cliente
WiFiClient client;

//variável usada para obter o request do client
String myresultat;
```

ESP32 [ReadIncomingRequest - Leitura do Request]

```
//função usada para a leitura do request sem caracteres de quebra de linha como "\n" ou "\r"
String ReadIncomingRequest()
{
   //enquanto houver bytes enviados pelo client
   while(client.available())
   {
      //atribui para a variável String o comando enviado pelo cliente sem "\r"
      ClientRequest = (client.readStringUntil('\r'));

      //se existir "HTTP/1.1" na String então recebe comando, senão o comando não é aceito
      //isso verifica que a solicitação seja HTTP/1.1
      if ((ClientRequest.indexOf("HTTP/1.1")>0))
         myresultat = ClientRequest;
   }
   //retorna variável
   return myresultat;
}
```

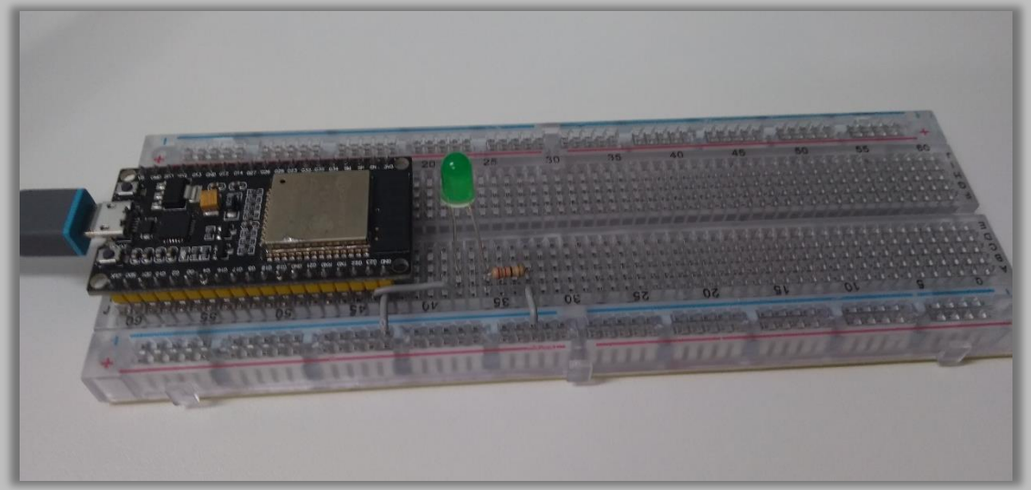
ESP32 [Setup]

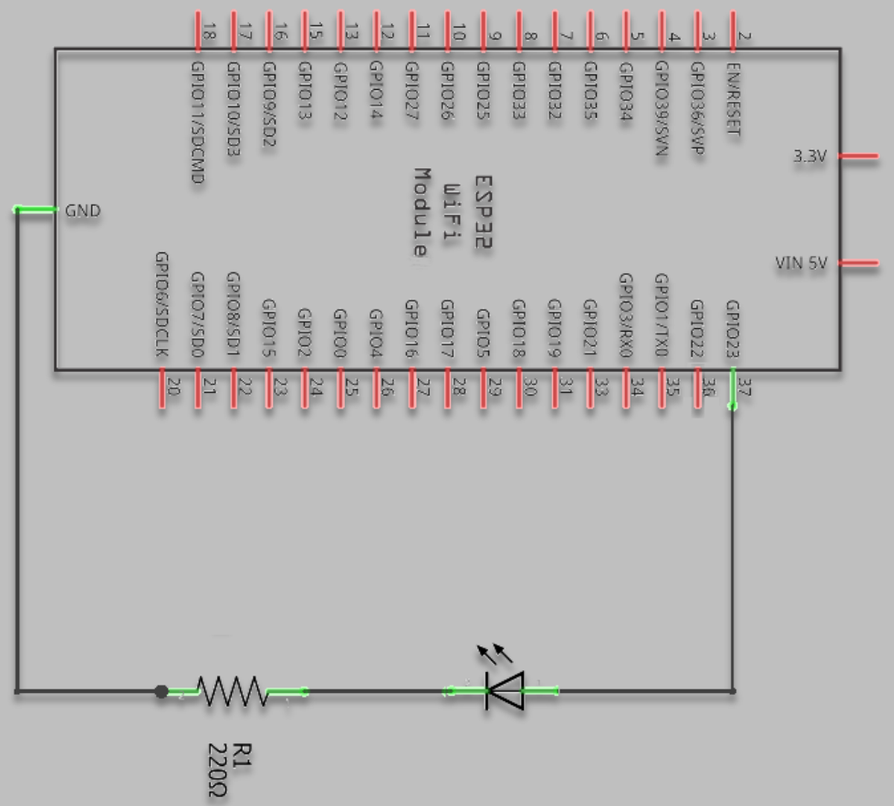
```
void setup()
{
   //inicializa varíavel como vazia
   ClientRequest = "";
   //define pino do led como saída
   pinMode(ledVerde,OUTPUT);

   //inicializa serial com 115200 bits por segundo
   Serial.begin(115200);
   //aguarda 10ms
   delay(10);

   //A partir daqui conecta wifi
   Serial.println("START");

   //configura ssid e senha da rede
   WiFi.begin(“robotica", “XXXXXXXX");

   //enquanto não conectar exibe "."
   while (WiFi.status() != WL_CONNECTED)
   {
      delay(500);
      Serial.print(".");
   }

   //exibe "conectado"
   Serial.println("Connected");

   //configura ip estático, gateway e máscara (definidos globais no início do código)
   WiFi.config(staticIP, gateway, subnet);

   //exibe ip utilizado pelo ESP
   Serial.println("Your IP is");
   Serial.println((WiFi.localIP()));
   //inicializa servidor
   server.begin();
}
```

ESP32 [Loop]

```
void loop()
{
   //obtém cliente
   client = server.available();

   //se ele for nulo, retorna até que ele realmente exista
   if (!client)
      return;

   //enquanto não existir request aguarda
   while(!client.available())
      delay(1);

   //obtém request utilizando a função local ReadIncomingRequest
   ClientRequest = (ReadIncomingRequest());
   //retira dados da página e obtém apenas o comando enviado
   ClientRequest.remove(0, 5);
   ClientRequest.remove(ClientRequest.length()-9,9);

   //controla led conforme o comando recebido
   if (ClientRequest == "acender")
      digitalWrite(ledVerde,HIGH);
   if (ClientRequest == "apagar")
      digitalWrite(ledVerde,LOW);
   if (ClientRequest == "piscar")
   {
      digitalWrite(ledVerde,HIGH);
      delay(500);
      digitalWrite(ledVerde,LOW);
      delay(500);
      digitalWrite(ledVerde,HIGH);
      delay(500);
      digitalWrite(ledVerde,LOW);
      delay(500);
   }
   //exibe na página a palavra "OK", caso acessado por um navegador
   //se estiver no aplicativo esta exibição não será feita
   client.println("HTTP/1.1 200 OK");
   client.println("Content-Type: text/html");
   client.println("");
   client.println("<!DOCTYPE HTML>");
   client.println("<html>");
   client.println("OK");
   client.println("</html>");
   client.flush();
   client.stop();
   delay(1);
}
```

Montagem ESP32

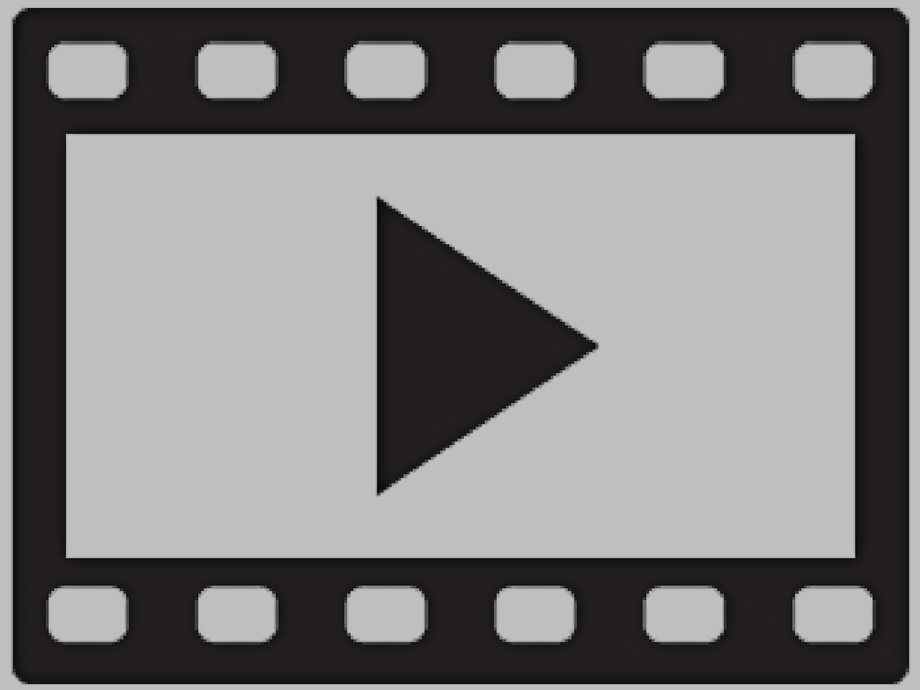




É necessário apenas um Led e um resistor
de 220ohm na porta 23 para este exemplo

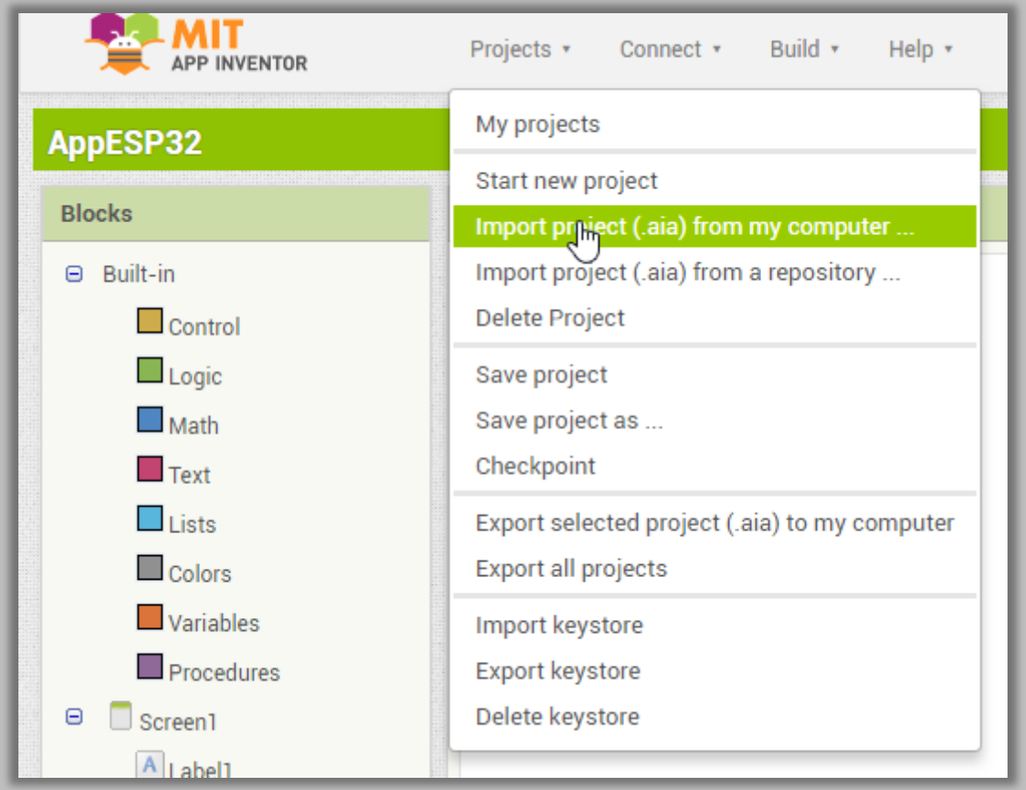
Programa AppInventor

Passo a passo - AppInventor

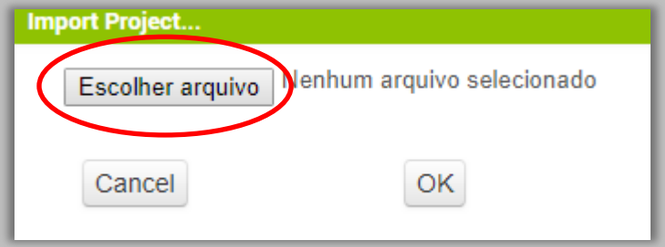


Importar arquivo [AppESP32.aia]

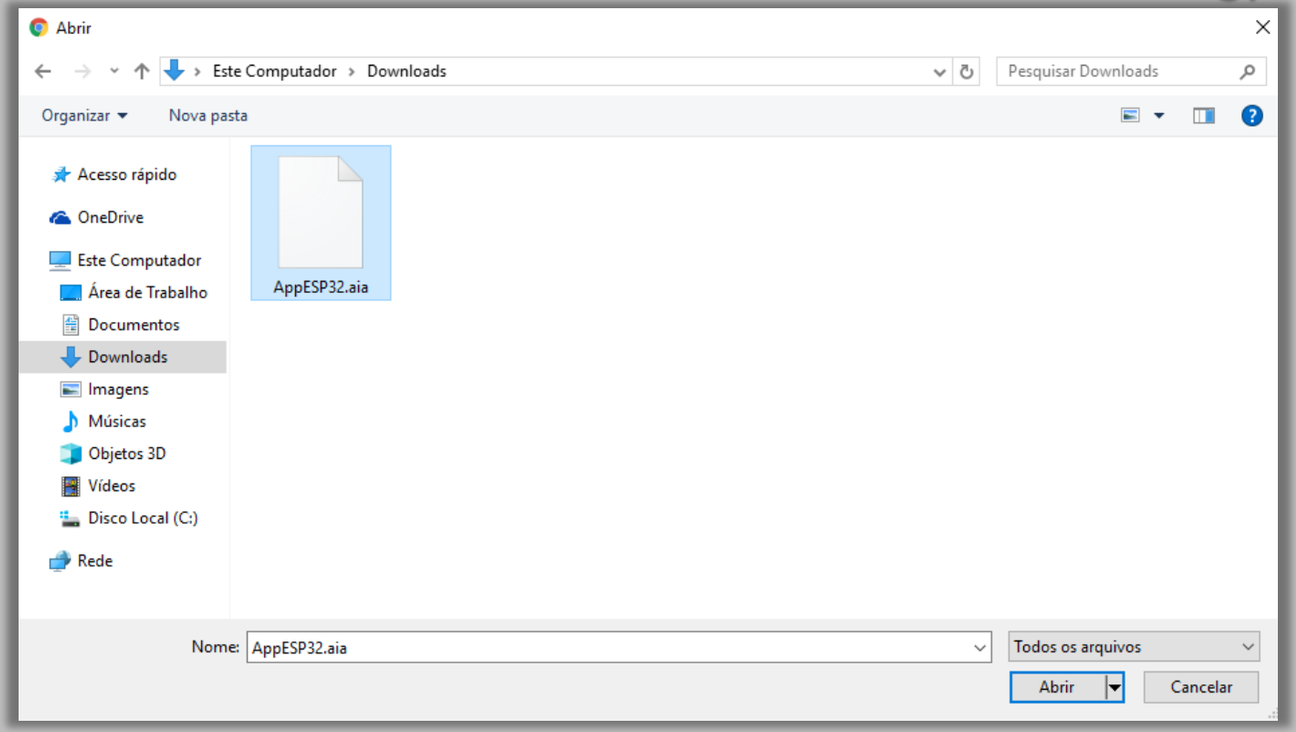
1.



2.

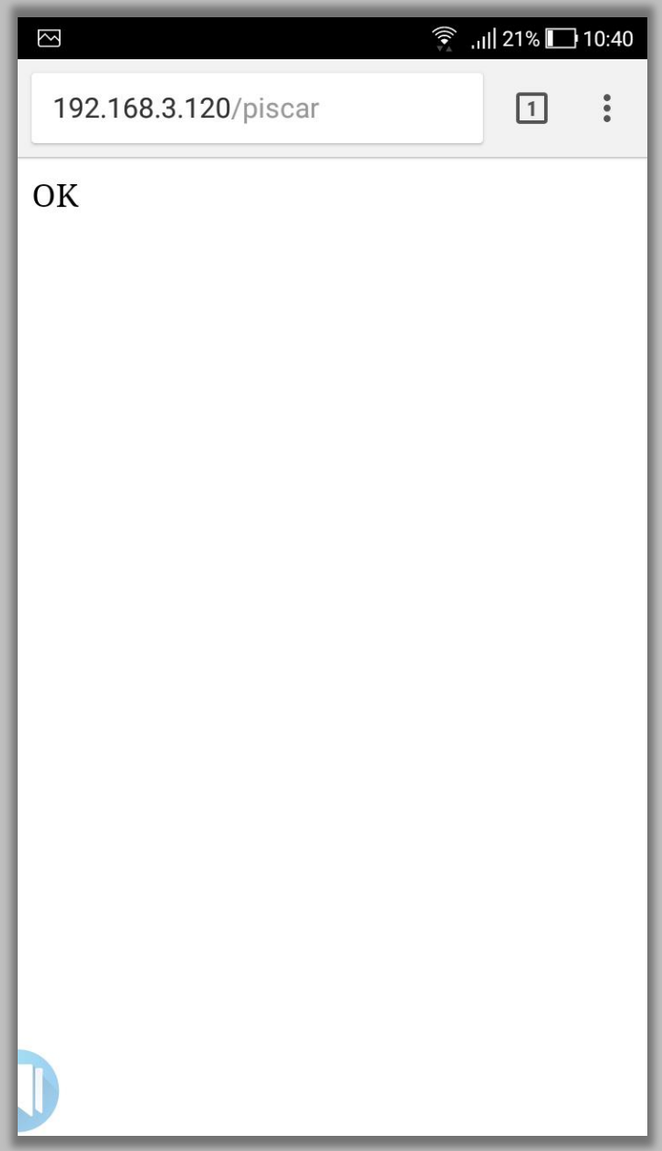


3.



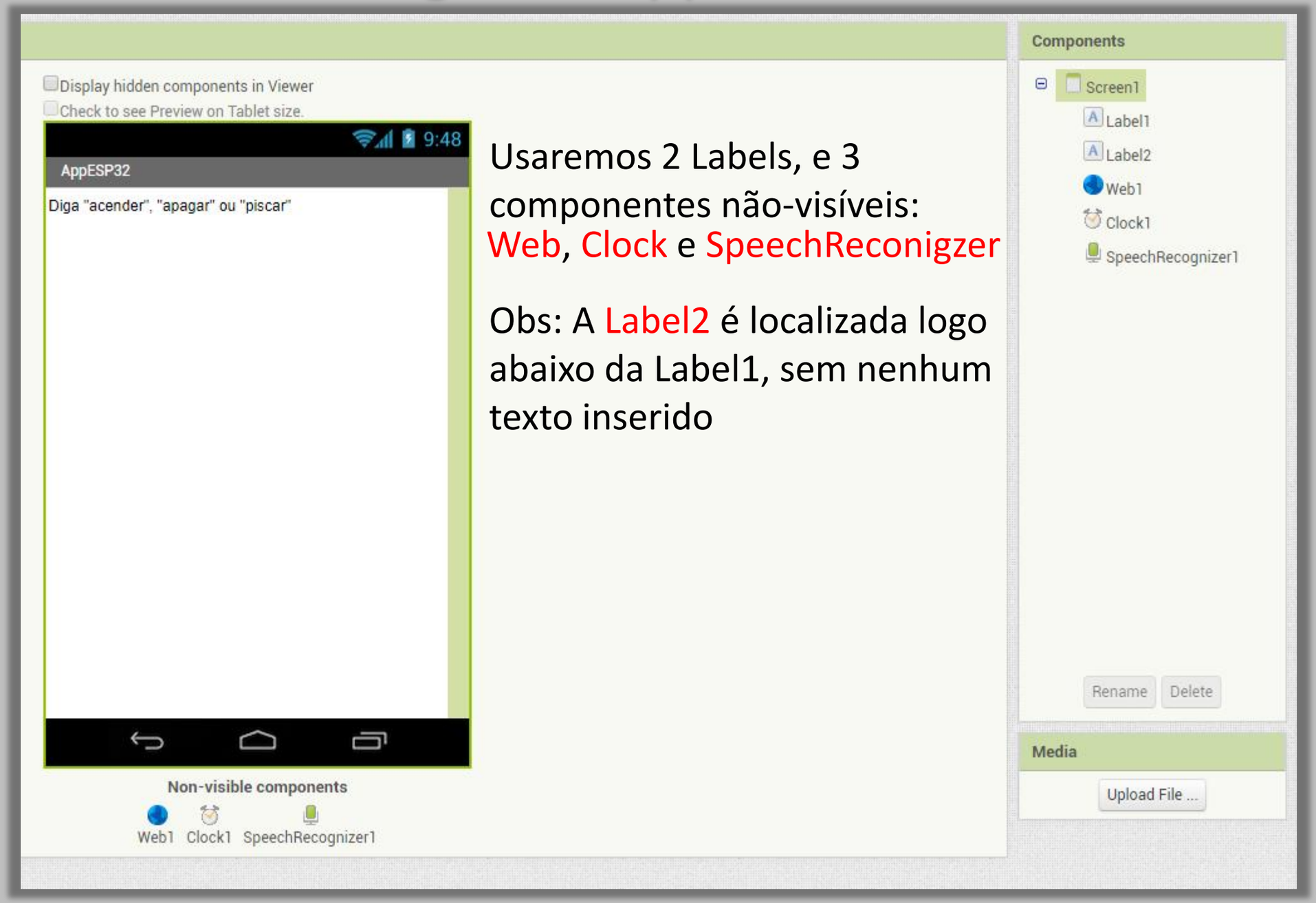
Baixe o arquivo .aia disponível no meu blog e importe para a sua conta do App Inventor seguindo estes passos.

Teste de comunicação do Smartphone com o ESP32



Para verificar se o seu celular pode se comunicar com o ESP32, abra o navegador e digite conforme a figura, lembrando que o ip deve ser o mesmo definido em “staticIP792” no código ino do ESP32.

Programa AppInventor [Designer]

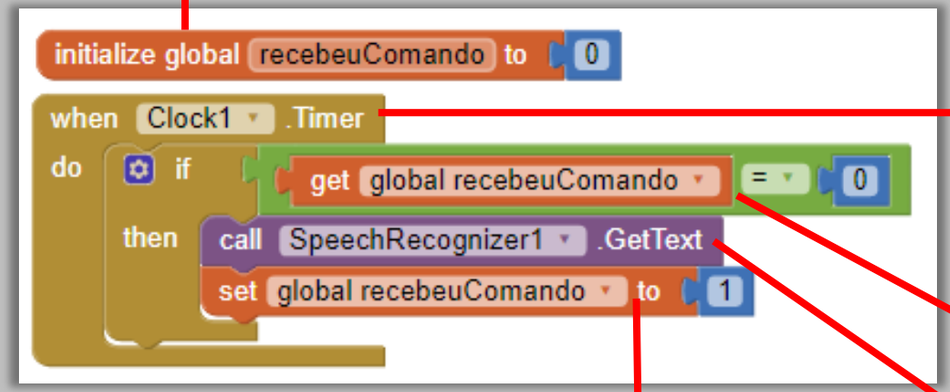


Usaremos 2 Labels, e 3 componentes não-visíveis: Web, Clock e SpeechReconigzer

Obs: A Label2 é localizada logo abaixo da Label1, sem nenhum texto inserido

Programa AppInventor [Blocks]

Inicializa variável global com “0”. Esta variável é usada para que a função de reconhecimento de voz seja chamada só uma vez, caso contrário a cada segundo essa função será chamada e ficará num loop “descontrolado”.



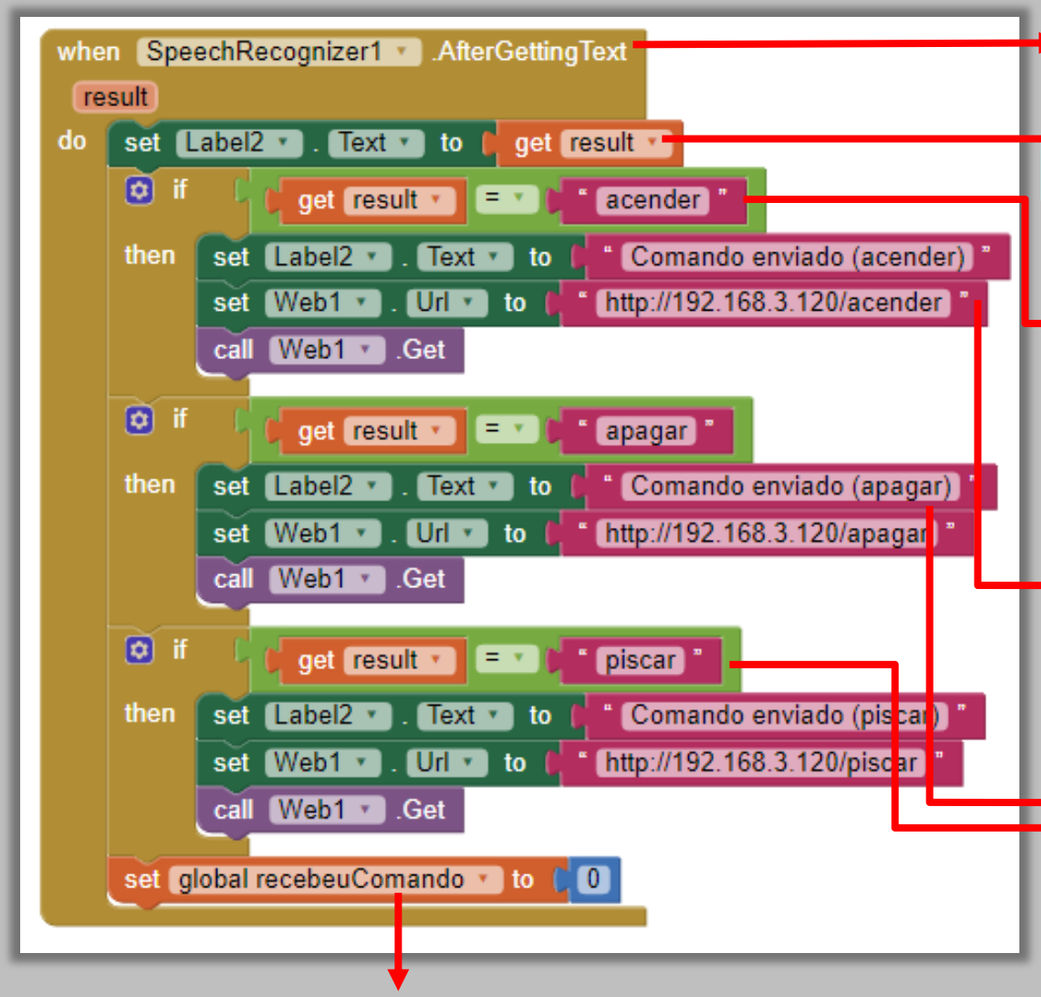
Este loop é executado assim que o aplicativo é aberto e ele funciona de tempo tempo (milissegundos). É como se fosse o loop do Arduino IDE, só que de acordo com o clock deste componente.

Se o comando não foi recebido ainda...

Então chama a função “SpeechRecognizer1” que é a função de reconhecimento de voz da Google.

Atribui à variável o valor “1”, que impede que a função seja chamada novamente neste loop.
(Até que ela seja setada como “0” novamente, veja no próximo slide)

Programa AppInventor [Blocks]



Após a voz for reconhecida e o texto já ter sido obtido...

Insere na etiqueta o texto que foi reconhecido

Se a palavra foi "acender", então...

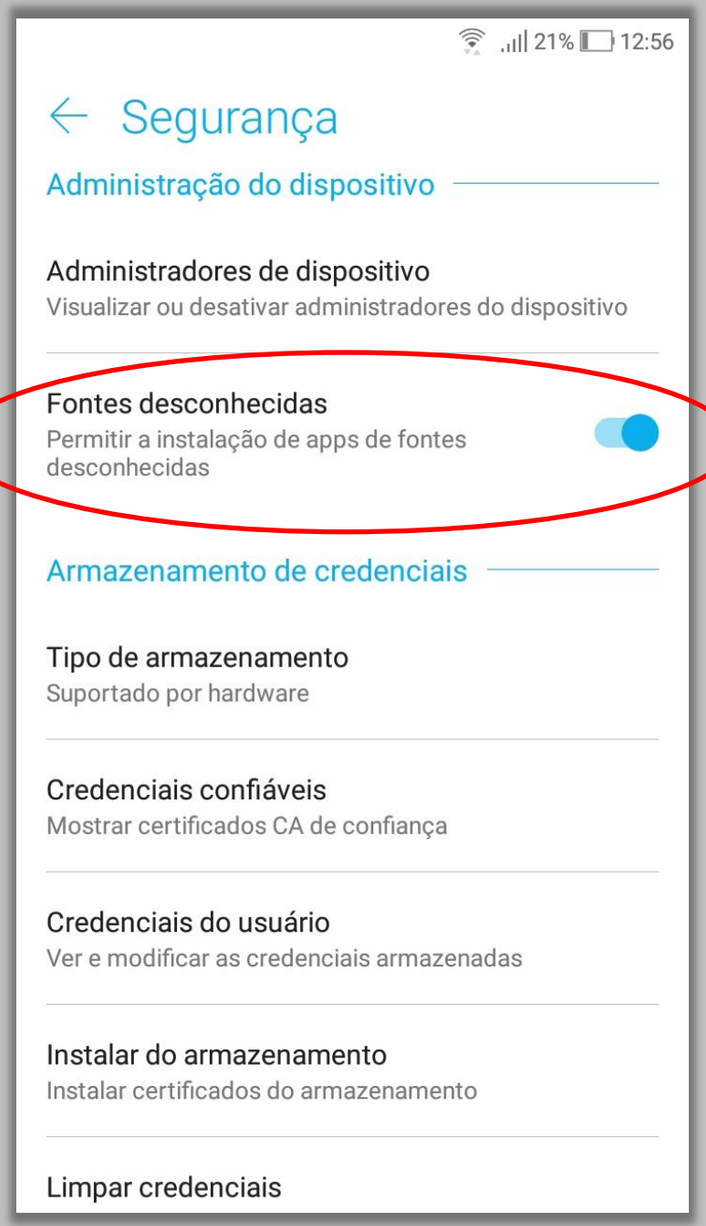
Insere na etiqueta o texto "Comando enviado" mais o comando entre parenteses

Envia por Url o comando "acender" (o ip deve ser o mesmo do ESP)

Repete para os demais comandos

Reseta variável para que o reconhecimento seja feito novamente

Instalação do aplicativo AppInventor [Permissão de instalação]

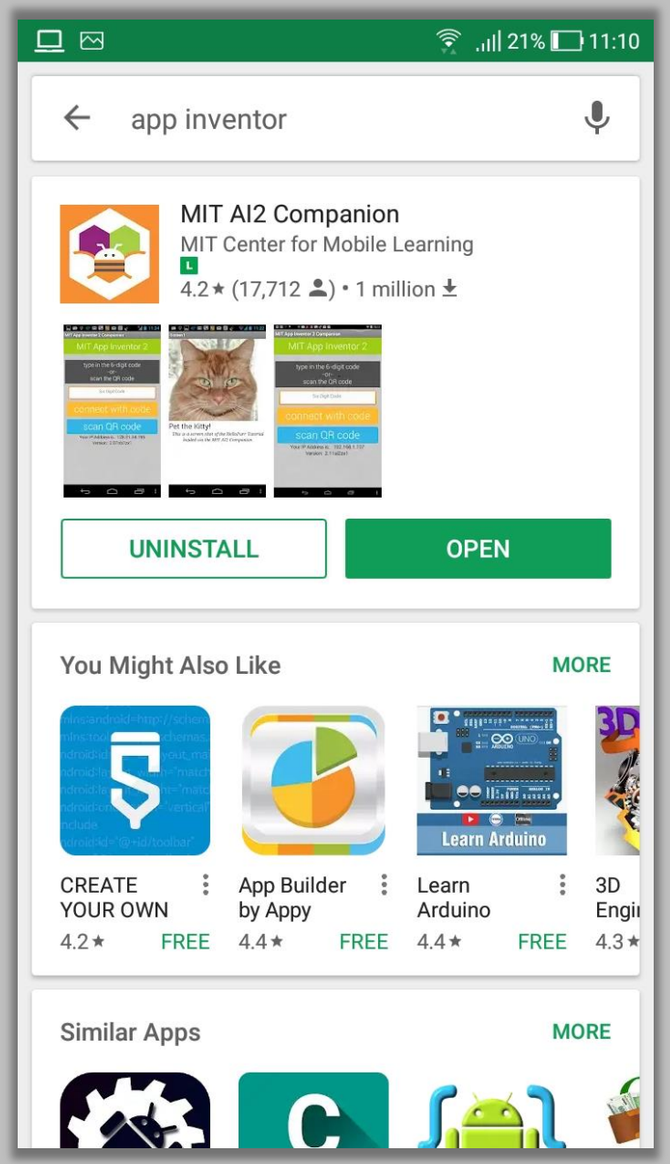


Antes de instalar o aplicativo habilite a permissão de instalação de apps de fontes desconhecidas.

No seu smartphone vá em: Configurações -> Segurança -> Administração do dispositivo

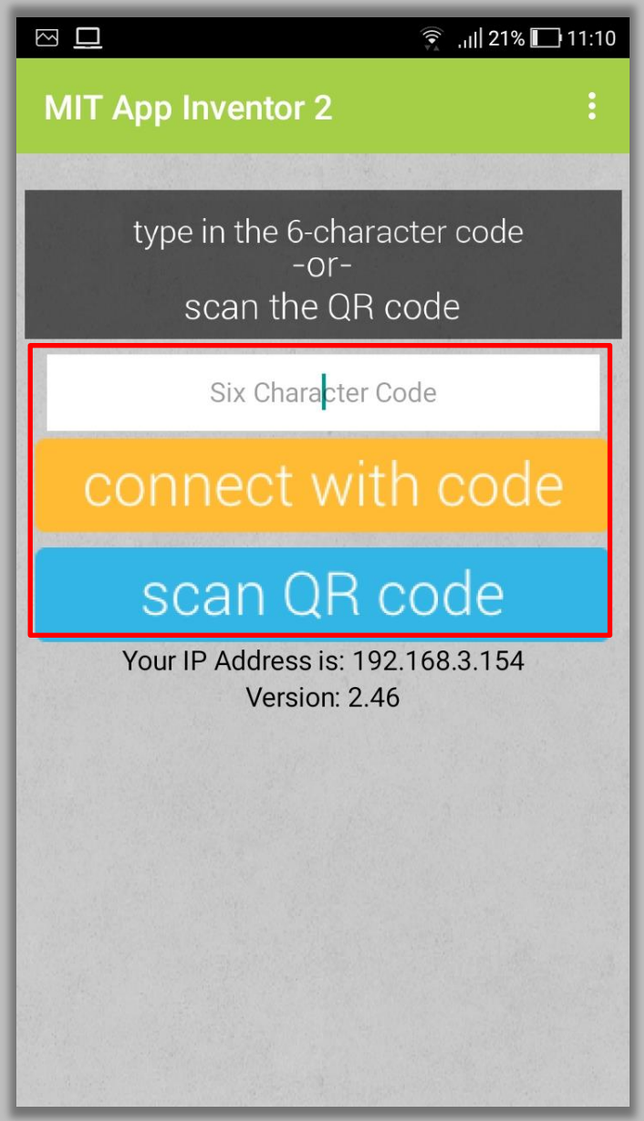
E marque a opção conforme a figura.

Instalação do aplicativo AppInventor [PlayStore]

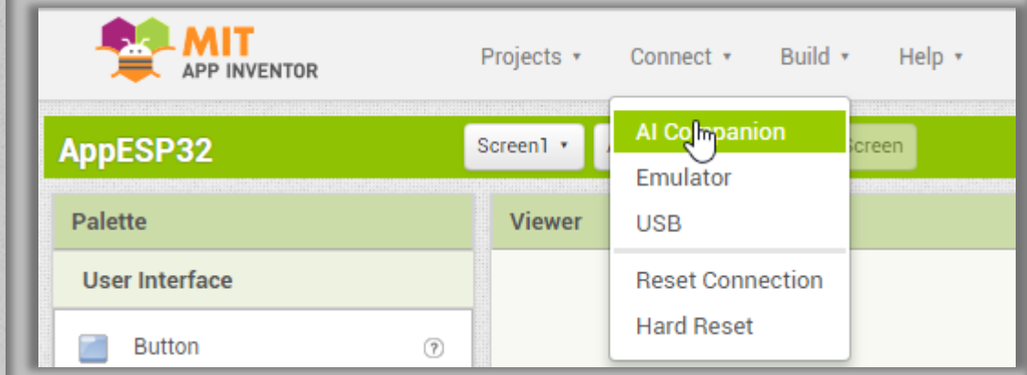


Abra a PlayStore e procure por “app inventor”, o aplicativo se chama “MIT AI2 Companion”.

Utilização e instalação do programa desenvolvido



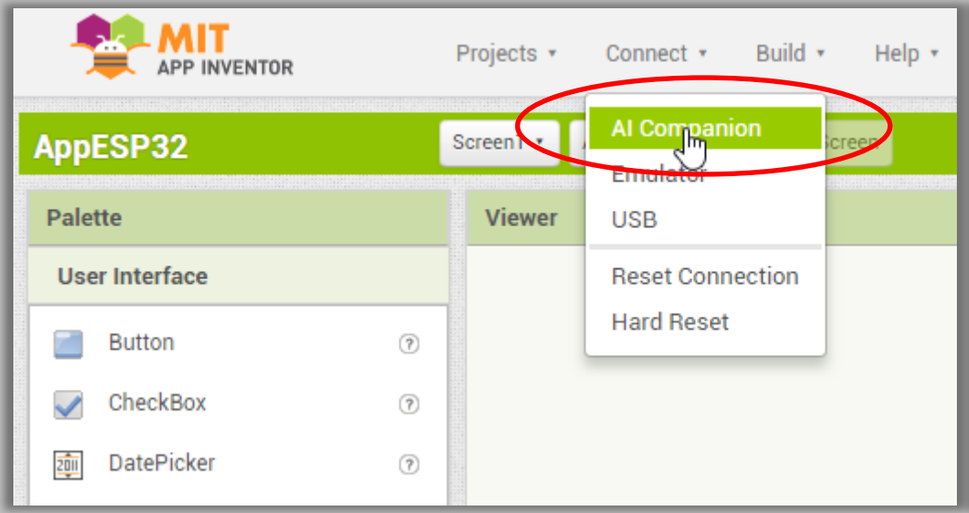
Abra o aplicativo baixado, esta é a tela principal.
É possível conectar com código ou escanear com QR code.



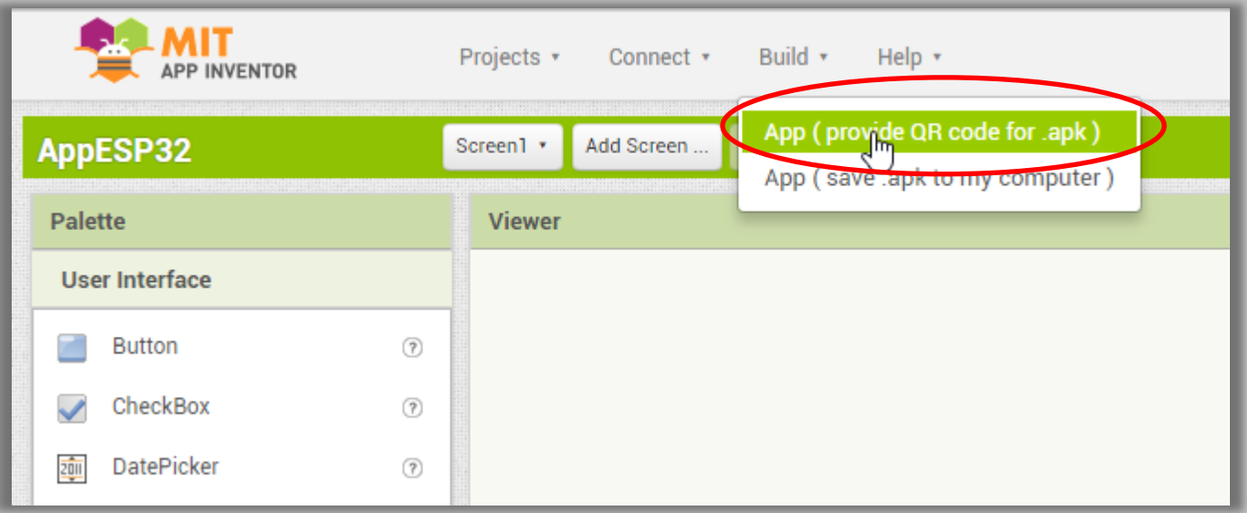
Site: http://ai2.appinventor.mit.edu

QRCode gerado pelo site

Utilização e instalação do programa desenvolvido



*Ao clicar são gerados um QR code e um código em texto



*Ao clicar é exibido um QR code

Opção usada para apenas utilizar o aplicativo sem que ele seja instalado. Detalhe: esta opção permite que o aplicativo seja atualizado em tempo real, assim que o “designer” ou “blocks” for alterado!

Opção usada para instalar o aplicativo no seu smartphone.

Em www.fernandok.com

Download arquivos PDF, INO e AIA do código fonte