



instructables

Controle Acionamento De ESP32 Por Comando De Voz

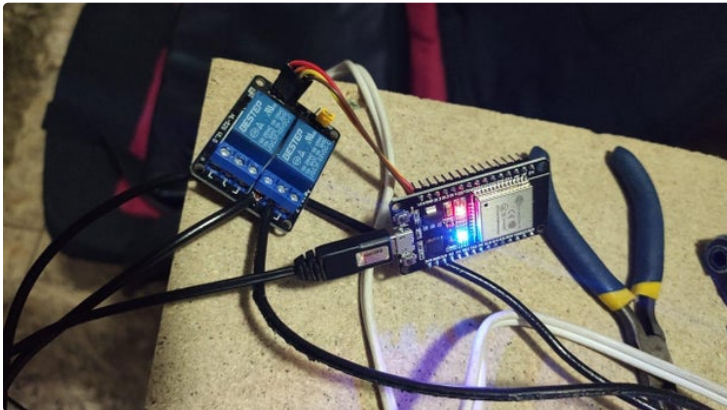


by mathbarc

O presente trabalho foi desenvolvido por **Matheus Barcelos de Oliveira** durante a aula de Internet das Coisas (IoT) lecionada pelo professor **Ilo Amy Saldanha Rivero**, no curso de Pós Graduação MBA em Indústria 4.0 da Instituição PUC Minas Unidade Coração Eucarístico.

Supplies:

- Um computador para hospedar um servidor **Mosquitto** (broker MQTT);
- Uma dispositivo **ESP32**;
- Uma placa contendo dois relés com acionamento de 5 volts;
- Um computador para realizar captura e conversão de áudio de áudio para texto.



Step 1: Instalação Do Servidor MQTT

Para hospedar o broker MQTT foi utilizada uma máquina Ubuntu 20.04 e para instalá-lo o seguinte comando foi utilizado:

- `sudo apt install mosquitto`

Obtendo uma resposta conforme imagem acima.

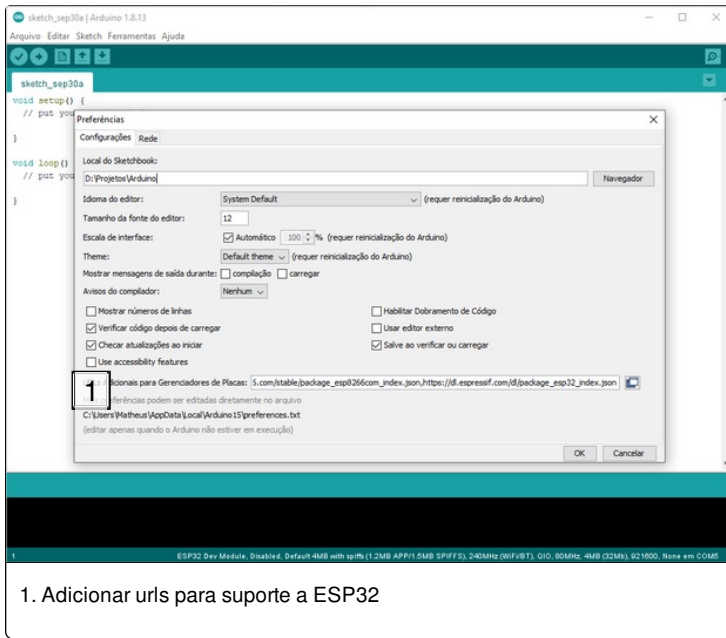
```
MINGW64/c/Users/Matheus
matheus@Gray:~$ sudo apt install mosquito
[sudo] senha para matheus:
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
Os seguintes pacotes foram instalados automaticamente e já não são necessários:
  arduino-core avr-libc avrdude binutils-avr espeak-ng-data extra-xdg-menus
  gcc-avr glib-networking:i386 gstreamer1.0-x:i386 libaal:i386
  libavc1394-0:i386 libcaca0:i386 libdv4:i386 libespeak-ng1 libftdi1
  libgstreamer-plugins-good1.0-0:i386 libgudev-1.0-0:i386 libhidapi-libusb0
  libiec61883-0:i386 libpcaudio0 libproxy1v5:i386 libraw1394-11:i386
  librtx-java librsamplerate0:i386 libshout3:i386 libslang2:i386
  libsoup2.4-1:i386 libtag1v5:i386 libtag1v5-vanilla:i386 libusb-0.1-4
  libxvi:i386 linux-headers-5.4.0-45 linux-headers-5.4.0-45-generic
  linux-image-5.4.0-45-generic linux-modules-5.4.0-45-generic
  linux-modules-extra-5.4.0-45-generic
Utilize 'sudo apt autoremove' para os remover.
The following additional packages will be installed:
  libdlt2 libev4 libwebsockets15
Os NOVOS pacotes a seguir serão instalados:
  libdlt2 libev4 libwebsockets15 mosquito
0 pacotes atualizados, 4 pacotes novos instalados, 0 a serem removidos e 0 não a
tualizados.
E preciso baixar 394 kB de arquivos.
Depois desta operação, 1.147 kB adicionais de espaço em disco serão usados.
Você quer continuar? [S/n] S
Obter:1 http://br.archive.ubuntu.com/ubuntu focal/universe amd64 libdlt2 amd64 2
.18.4-0.1 [50,4 kB]
Obter:2 http://br.archive.ubuntu.com/ubuntu focal/universe amd64 libev4 amd64 1:
4.31-1 [31,2 kB]
Obter:3 http://br.archive.ubuntu.com/ubuntu focal/universe amd64 libwebsockets15
amd64 3.2.1-3 [152 kB]
Obter:4 http://br.archive.ubuntu.com/ubuntu focal/universe amd64 mosquito amd64
1.6.9-1 [160 kB]
Baixados 394 kB em 1s (447 kB/s)
A seleccionar pacote anteriormente não seleccionado libdlt2:amd64.
(Lendo banco de dados ... 554542 ficheiros e directórios actualmente instalados.
)
A preparar para descompactar .../libdlt2_2.18.4-0.1_amd64.deb ...
A descompactar libdlt2:amd64 (2.18.4-0.1) ...
A seleccionar pacote anteriormente não seleccionado libev4:amd64.
A preparar para descompactar .../libev4_1%3a4.31-1_amd64.deb ...
A descompactar libev4:amd64 (1:4.31-1) ...
A seleccionar pacote anteriormente não seleccionado libwebsockets15:amd64.
A preparar para descompactar .../libwebsockets15_3.2.1-3_amd64.deb ...
A descompactar libwebsockets15:amd64 (3.2.1-3) ...
A seleccionar pacote anteriormente não seleccionado mosquito.
A preparar para descompactar .../mosquito_1.6.9-1_amd64.deb ...
A descompactar mosquito (1.6.9-1) ...
Configurando libev4:amd64 (1:4.31-1) ...
Configurando libdlt2:amd64 (2.18.4-0.1) ...
Configurando libwebsockets15:amd64 (3.2.1-3) ...
Configurando mosquito (1.6.9-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/mosquito.service =
/lib/systemd/system/mosquito.service.
A processar 'triggers' para systemd (245.4-4ubuntu3.2) ...
A processar 'triggers' para man-db (2.9.1-1) ...
A processar 'triggers' para libc-bin (2.31-0ubuntu9.1) ...
matheus@Gray:~$
```

Step 2: Configurando Arduino IDE

Para programar o dispositivo ESP32 foi escolhida a **Arduino IDE** devido à sua simplicidade.

Após o download e instalação da ferramenta é necessário adicionar a seguinte linha ao item presente em *Arquivo->Preferências->Urls Adicionais para Gerenciadores de Placas*.

- https://arduino.esp8266.com/stable/package_esp8266com_index.json,https://dl.espressif.com/dl/package_esp32_index.json



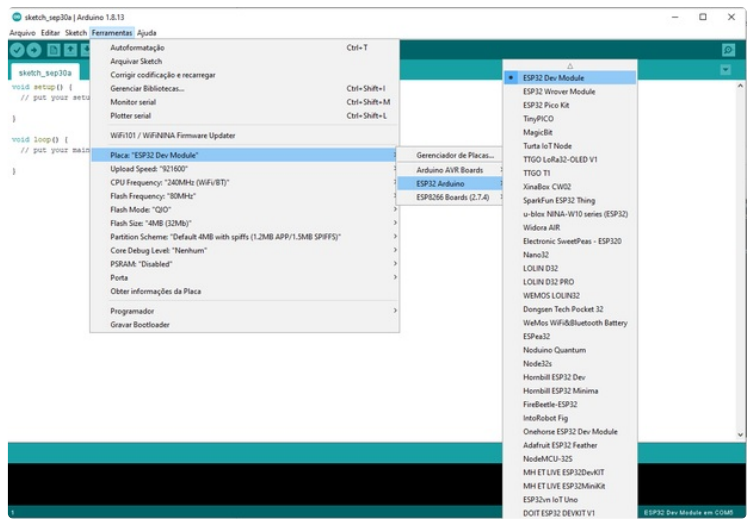
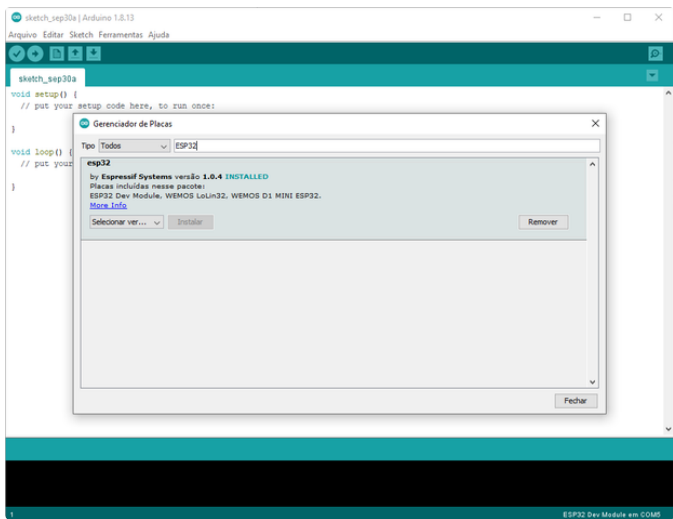
Step 3: Instalando Suporte À ESP32 No Arduino IDE

Com as urls configuradas do passo anterior, é possível acessar o gerenciador de placas no caminho *Ferramentas->Placa->Gerenciador de Placas* o que abrirá a tela conforme imagem acima.

Escrevendo ESP32 no filtro conforme a imagem, aparecerá o item esp32 by Espressif Systems, sendo necessário apenas instalá-lo.

Após instalá-lo, é necessário selecionar a placa em utilização clicando no seguinte caminho: *Ferramentas->Placa->ESP32 Arduino->ESP32 Dev Module*

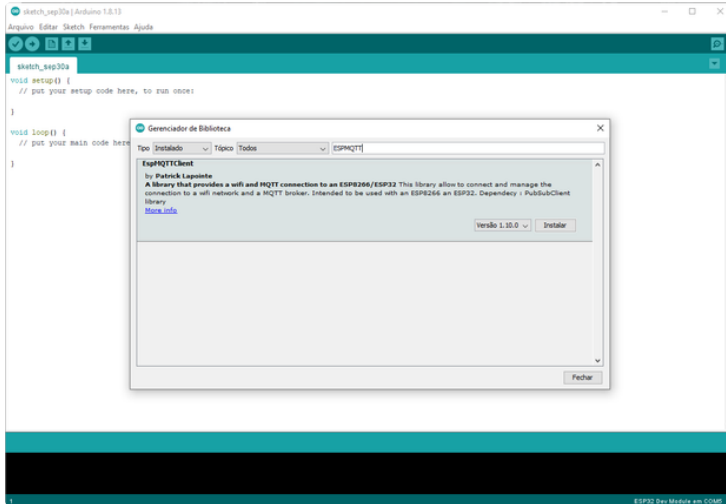
Após esta sequencia de passos já será possível programar o ESP32 utilizando Arduino IDE e conectando a placa ao computador por meio do cabo USB.



Step 4: Programando ESP32 Para Acionamento a Partir De MQTT

Após configurada a IDE, é necessário instalar a biblioteca EspMqttClient, para isso é necessário navegar para o seguinte caminho: *Ferramentas->Placa->ESP32 Arduino->ESP32 Dev Module*, o que fará a tela na imagem acima surgir.

Após a instalação já é possível desenvolvermos o firmware que será executado no dispositivo.



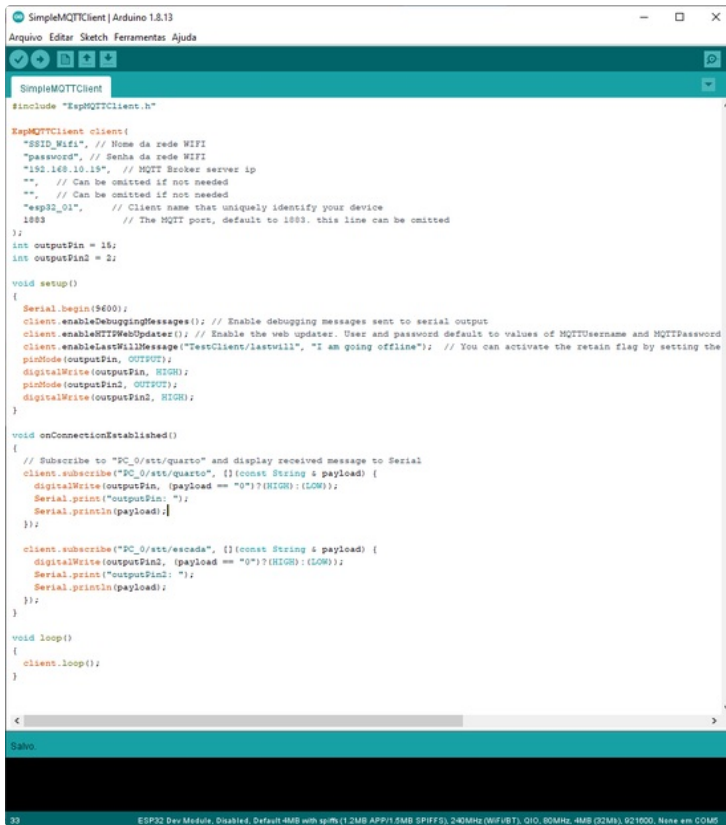
Step 5: Desenvolvimento Do Firmware

O programa de modelo SimpleMQTTClient foi modificado para permitir a subscrição em dois tópicos que serão publicados pelo dispositivo que interpretará os comandos de voz. A versão final utiliza no dispositivo ESP32 está contida na imagem.

Neste código foi criada um objeto da classe EspMQTTClient denominado client. Este cliente será responsável por buscar novas publicações disponibilizadas pelo broker. Além da criação do objeto client, é importante definir a função "*void onConnectionEstablished()*", pois esta função é responsável por realizar inscrições em assuntos disponíveis no broker ao se realizar uma conexão com sucesso. No caso foram realizadas inscrições nos assuntos

- PC_0/stt/quarto
- PC_0/stt/escada

e definidas funções lambda que tratarão a atualização dos estados de cada saída digital (pinos D15 e D2) conectadas à placa de acionamento de relés.



Step 6: Script Python Para Comandos De Voz

Após a programação do dispositivo ESP32, é necessário criar uma entidade que publique os estados a serem atualizados no dispositivo. Para isso foi criado um script Python que utiliza Speech to Text para transcrever áudio captado por um microfone instalado a um computador e quando reconhecido o comando, o estado é publicado via MQTT para o dispositivo ESP32.

Para isso serão necessários pacotes que podem ser instalados com os seguintes comandos:

- `sudo pip install speech_recognition pyaudio paho_mqtt`

ou para sistemas Windows:

- `pip install pipwin`
- `pipwin install pyaudio`
- `pip install speech_recognition paho_mqtt`

Após instalados os pacotes o script contido na imagem acima cria um objeto que encapsula um cliente MQTT e utilizando a ferramenta de transcrição de áudio do GCP (Google Cloud Platform) verifica se no áudio os comandos foram ditos e assim realiza publicações nos respectivos assuntos MQTT.

```

1 import speech_recognition as sr # import the library
2 import paho.mqtt.client as mqtt
3 from struct import pack
4
5 #print(sr.Microphone.list_microphone_names())
6 clientid = 'PC_0'
7 client = mqtt.Client(client_id = clientid,
8                     protocol = mqtt.MQTTv31)
9
10 client.connect(("192.168.18.19", 1883))
11 deviceId = clientid + '/stt/'
12
13 r = sr.Recognizer() # initialize recognizer
14 with sr.Microphone() as source: # mention source it will be either Microphone or audio files.
15     print("Diga o comando: ")
16
17     while True:
18         r.adjust_for_ambient_noise(source)
19
20         audio = r.listen(source, phrase_time_limit=5) # listen to the source
21         #with open("microphone-results.wav", "wb") as f:
22         #    f.write(audio.get_wav_data())
23         #audio.play()
24         try:
25             text = r.recognize_google(audio, language="pt-BR") # use recognizer to convert our audio into
26             print("{}\n".format(text))
27             if text == 'sair' or text == 'encerrar':
28                 break
29             if text == 'liga o abajur':
30                 print("ligando")
31                 client.publish(deviceId+"abajur","1",qos=0)
32             if text == 'desliga o abajur':
33                 print("desligando")
34                 client.publish(deviceId+"abajur","0",qos=0)
35             if text == 'liga a luz da escada':
36                 print("ligando")
37                 client.publish(deviceId+"escada","1",qos=0)
38             if text == 'desliga a luz da escada':
39                 print("desligando")
40                 client.publish(deviceId+"escada","0",qos=0)
41
42         except:
43             print("Sorry could not recognize your voice") # In case of voice not recognized clearly
44
45

```

Step 7: Resultado

Após todos os passos anteriores foi possível atingir o resultado esperado e realizar o acionamento de um abajur utilizando comandos de voz conforme o seguinte vídeo:

https://www.youtube.com/embed/SGRybBWM_gc