SDK ESP Docs Documentation

Release 1.0.0

Automacao IOT

fev 09, 2019

Getting Started:

If you are looking for the english documentation go to here.

CAPÍTULO 1

Guia Básico

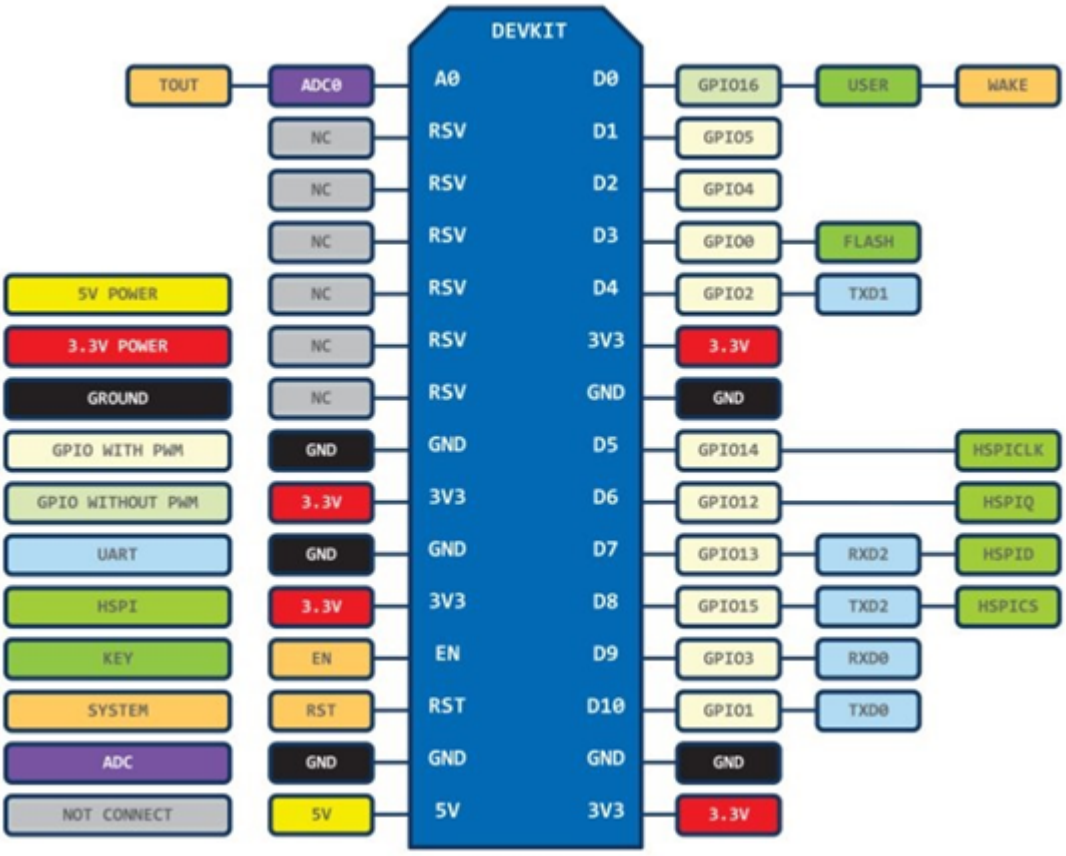
SDK ESP é o kit de desenvolvimento de software pra os projetos relativos a Automação-IOT, baseados no hardware ESP8266 ou ESP32 Nodemcu. Este SDK foi baseado no projeto MicroPython (http://micropython.org/).

1.1 ESP8266

ESP8266 é um módulo WIFI de baixo custo, adequado para projetos de micontroladores, totalmente reprogramável. Possui como lista de recursos, os seguintes itens:

- Protocolo 802.11 b/gn
- Wi-fi direct (P2P)
- Pilha de protocolo TCP/IP integrado

1.2 Pinagem ESP8266



Pinagem :

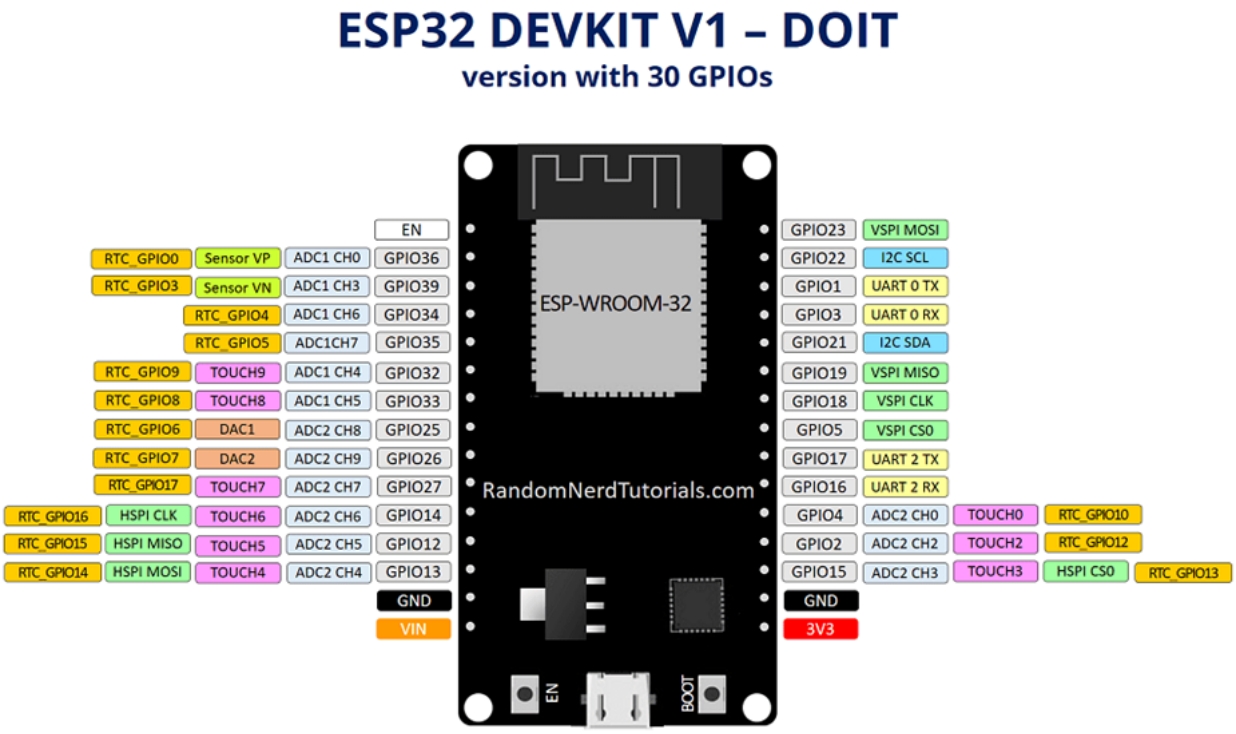
PINO	GPIO
D0	16
D1	05
D2	04
D3	00
D4	02
D5	14
D6	12
D7	13
D8	15
D9	03
D10	01

1.3 ESP32

- Protocolo 802.11 b/gn
- WiFi: 2,4 GHz, 802.11 b/g/n

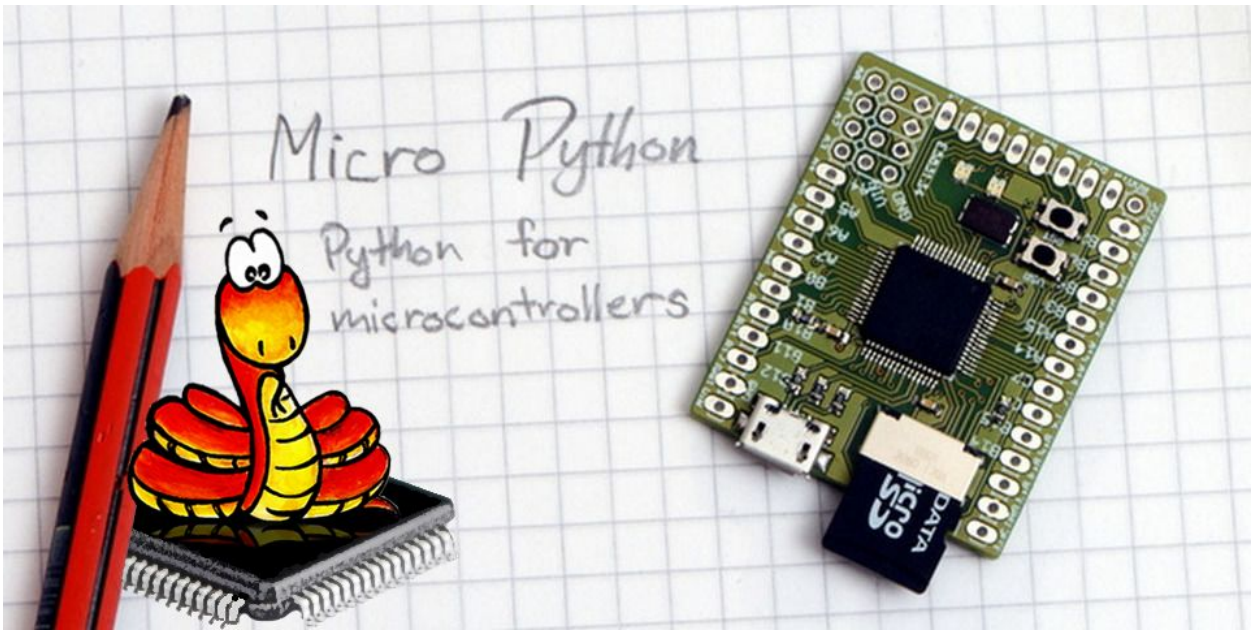
- Bluetooth: Bluetooth Low Energy v4.2 (BLE)
- 1 Sensor de temperatura interno
- 1 Sensor de efeito Hall

1.4 Pinagem ESP32



1.5 Projeto MicroPython

Projeto Micropython, que possui como foco principal a implementação do Python 3.X em pequenos sistemas embarcados.


Micro Python
Python for
microcontrollers

CAPÍTULO 2

Bibliotecas SDK-ESP

2.1 Biblioteca Common

É representado pelo conjunto de classes e métodos comum a todas as classes da biblioteca Resources.

Importante: Bibliotecas Common. Relação de bibliotecas common

- Api
- Device
- Resource
- Wifi

2.2 Biblioteca Resources

É representado pelo conjunto de recursos de entrada/saída que possuam capacidade de integração e comunicação com dispositivo.

Importante: Bibliotecas Resources. Relação de bibliotecas resources

- Dht
- Ds1820
- Led
- Rele

Importante: Classe DHT

- DHT11;
- DHT21; e
- DHT22.

CAPÍTULO 3

Exemplos SDK-ESP

3.1 Exemplos de uso do SDK-ESP

Recurso Rele ESP8266

```
from Device import Device
from Rele import Rele
from Led import LedRGB


device = Device("PUBLIC_KEY","SECRET_KEY",1000)
device.SYS_CPU_160MHZ()
device.presenceLed(device.GPIO14,device.GPIO12,device.GPIO13,LedRGB.CATODO)
device.setNetworkConfig('SSID','PWD')
device.setDebug(True)

rele = Rele(idRELE,device.GPIOrele,Rele.OPEN,1000)

device.start()
```

Recurso Rele ESP32

```
from Device import Device
from Rele import Rele
from Led import LedRGB


device = Device("PUBLIC_KEY","SECRET_KEY",1000)
device.SYS_CPU_240MHZ()
device.presenceLed(device.GPIO25,device.GPIO26,device.GPIO27,LedRGB.CATODO)
device.setNetworkConfig('SSID','PWD')
device.setDebug(True)

rele = Rele(idRELE,device.GPIOrele,Rele.OPEN,1000)
```

(continues on next page)

(continuação da página anterior)

```
device.start()
```

Recurso Dht ESP8266

```
from Device import Device
from Led import LedRGB
#from Dht import Dht

device = Device("PUBLIC_KEY","SECRET_KEY",1000)
device.SYS_CPU_160MHZ()
device.presenceLed(device.GPIO14,device.GPIO12,device.GPIO13,LedRGB.CATODO) # 14 (D5-
↪RED) 12 (D6 - GREEN) 13 (D7 - BLUE)
device.setNetworkConfig('SSID','PWD')
device.setDebug(False)

dht = Dht(idTemperature,idHumidity,device.GPIOdht,Dht.type,True,1000)

device.start()
```

Recurso Ds1820 ESP32

```
from Device import Device
from Ds1820 import Ds1820
from Led import LedRGB

device = Device("PUBLIC_KEY","SECRET_KEY",1000)
device.SYS_CPU_240MHZ()
device.presenceLed(device.GPIO25,device.GPIO26,device.GPIO27,LedRGB.CATODO)
device.setNetworkConfig('SSID','PWD')
device.setDebug(True)

rele = Rele(idRELE,device.GPIOrele,Rele.OPEN,1000)
ds = Ds1820(idDS1820,device.GPIOds1820,Ds1820.type,device.escala,True)

device.start()
```

CAPÍTULO 4

Criando Classe SDK

4.1 Estrutura Classe SDK

O módulo de SDK poderá ser expandido a outros sensores (recursos), para isto basta seguir o modelo de estrutura abaixo, para garantir a integração junto ao sistema da Automação IOT.

Classe Hipotética

Este recurso (hipoteticamente) lê dados de uma determinada GPIO e envia para a Base de Dados IoT:

LeGPIO.py

```
from Device import Device
from Resource import Resource


class LeGPIO(Resource):

        def __init__(self, *args):

                self.refresh = 1000
                self.pin = None
                self.state = None

                if len(args) == 3:
                        self.id = args[0]
                        self.gpio = args[1]
                elif len(args) == 4:
                        self.id = args[0]
                        self.gpio = args[1]
                        self.refresh = args[3]

                Device.addResource(self,self)
                self.tsLeGPIOBegin = Device.getTime(self)
```

(continues on next page)

(continuação da página anterior)

```
            @staticmethod
            def start(cls, protocol, url, keyPublic, keySecret,debug):
                    try:
                            if(Resource.resourceTime(Device.getTime(cls),cls.
↪tsLeGPIOBegin,cls.refresh)):
                                    cls.actionStart(protocol, url, keyPublic,␣
↪keySecret, debug)
                                    cls.tsLeGPIOBegin = Device.getTime(cls)
                    except:
                            pass
                    return

            def actionStart(self,protocol, url, keyPublic, keySecret,debug):
                    try:
                            self.pin = Pin(self.gpio, Pin.IN)
                            valuePin = self.pin.value()

            value = Resource.createFeed(self.id, protocol, url, keyPublic,␣
↪keySecret,valuePin, debug)
                    except:
                            pass
                    return
```

CAPÍTULO 5

Links Externos

5.1 Links

Documentação API Automação-IOT.

Documentação Site Automação-IOT.

Documentação Micropython.