KHALSA LABS

NETWORK SECURITY USING ESP32 & MICROPYTHON

HACK GUIDE - 1

WWW.KHALSALABS.COM

TABLE OF **CONTENTS**

Chapter 01 *GETTING STARTED*

Chapter 02 *CALL A REST API*

Chapter 03 *SEND EMAIL*

Chapter 04 *PORT SCANNING*

CHAPTER 01

GETTING STARTED

MicroPython is a lightweight version of Python that can run on microcontrollers, such as ESP32. MicroPython allows you to program the ESP32 board with Python code, making it easier and faster to develop applications for the Internet of Things (IoT) and sensor systems. IoT and sensor applications are rapidly growing fields that enable you to connect devices, collect data, and control the physical world with the power of the internet.

Let’s learn how to install micropython on the esp32 tutorial, we will download the official micropython binaries and flash them to esp32. Then verify the working of micropython on esp32.

What You Need:

- ESP32 Module: This is the hardware you'll be programming.
- Micro USB Cable: To connect ESP32 to your computer.
- Computer: With internet access to download necessary files and software.

Steps to install micropython on esp32:

Step 1: Download the Firmware

- Go to the MicroPython Website: Visit the MicroPython downloads page.
- Select Firmware for ESP32: Look for the section labeled 'Firmware for ESP32 boards'.
- Download the Latest Version: Click on the link to download the latest firmware (.bin file) to your computer.

Step 2: Install esptool.py

- Install Python: Make sure you have Python installed on your computer. You can download it from the Python official website.
- Open Command Line: Open Command Prompt (Windows) or Terminal (macOS/Linux).
- Install esptool: Type pip install esptool and press Enter.
- This tool is used to communicate with the ESP32 chip. Link to documentation

Step 3: Connect ESP32 to Computer

- Plug in Your ESP32: Use the micro USB cable to connect the ESP32 to your computer.
- Check the COM Port:
- On Windows, open Device Manager and look under 'Ports (COM & LPT)' to find out which COM port the ESP32 is connected to.
- On macOS/Linux, open Terminal and type ls /dev/tty.* to list the devices connected.

Step 4: Erase the Flash

- Open Command Line: If you closed it earlier, open it again.
- Go to the ESP32 Port: Type esptool.py --port COMx erase_flash (replace 'COMx' with the correct COM port).
- Execute the Command: Press Enter to erase the existing flash on the ESP32.

Steps to install micropython on esp32:

Step 5: Flash MicroPython

1. Locate the Downloaded Firmware: Find where you saved the .bin file (that you downloaded in step 1)
2. Flash the Firmware: Type esptool.py --port COMx --baud 460800 write_flash -z 0x1000 path-to-your-downloaded.bin (replace 'COMx' with your COM port and 'path-to-your-downloaded.bin' with the path to the downloaded .bin file).

Step 6: Verify Installation

Verify installation using Thonny

One simple way to verify the installation is install thonny and select your COM Port or tty connected device from settings. You will see the >>> sign on thonny's shell indicating that micropython is installed.

Verify installation using Putty on windows:

1. Install a Serial Terminal: Install a program like PuTTY or use the built-in terminal for communication. Open the serial terminal, select the COM port, set the baud rate to 115200, and connect.
2. Check for MicroPython Prompt: You should see the MicroPython prompt >>>. If you see this, congratulations, MicroPython is installed!

You've successfully installed MicroPython on your ESP32.

CHAPTER 02
HOW CALL A REST API FROM ESP32 USING MICROPYTHON

To call a REST API from an ESP32 using MicroPython, you'll typically need to connect the ESP32 to a Wi-Fi network and then use HTTP methods to interact with the API. Below is a step-by-step tutorial to achieve this.

We will use microphone in this tutorial to call REST API from ESP32:

What do you need

- ESP32 with MicroPython Installed
- Wi-Fi network Details (SSID and password).

Step 1: Code to Connect ESP32 to Wi-Fi

```
IMPORT NETWORK

# INITIALIZE WI-FI INTERFACE
WLAN = NETWORK.WLAN(NETWORK.STA_IF)
WLAN.ACTIVE(TRUE)

# WIFI CREDS
SSID = 'YOUR_WIFI_SSID'
PASSWORD = 'YOUR_WIFI_PASSWORD'

# CONNECT TO YOUR WI-FI:
WHILE NOT WLAN.ISCONNECTED():
   PASS
PRINT('CONNECTED TO WI-FI'))
```

Step 2: Install urequests Library (If Not Present)

The urequests library is used for making HTTP requests. It may or may not be included in your MicroPython firmware (its included in latest micropython).

If you are using older version of micropython, lets download the urequest with this simple step:
Download the file from here and save it on the root directory of your esp32 (using thonny or any file transfer tool like ampy). Using thonny it will be easier

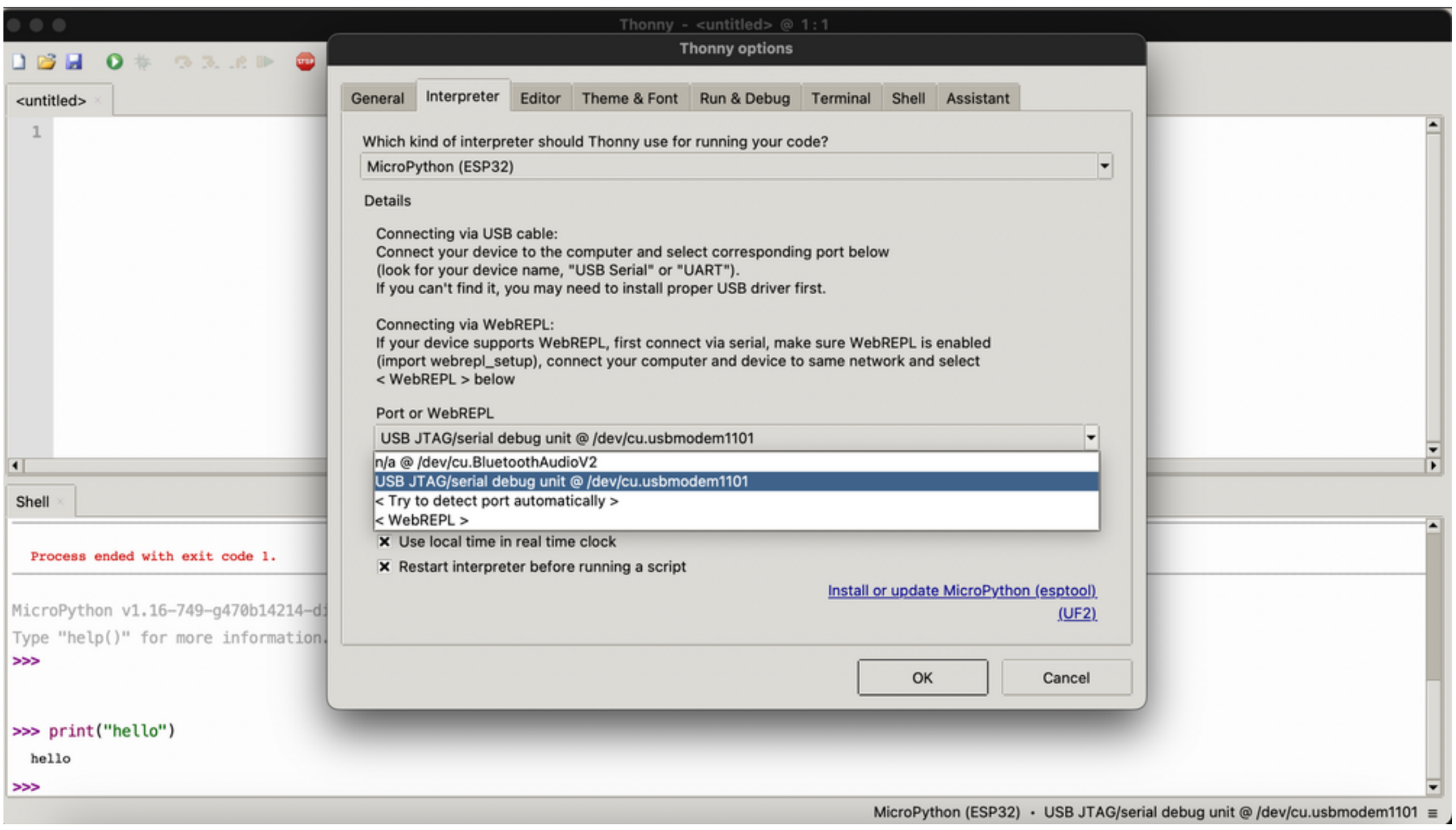
Step 3: Code to Call REST API

```
IMPORT UREQUESTS

# CALLING A PUBLICALY AVAILABLE REST API
URL = 'HTTP://API.EXAMPLE.COM/DATA'
RESPONSE = UREQUESTS.GET(URL)

# CHECKING THE RESPONSE
IF RESPONSE.STATUS_CODE == 200:
  PRINT(RESPONSE.TEXT) # PRINTING API RESPONSE
ELSE:
 PRINT('FAILED. ERROR CODE:', RESPONSE.STATUS_CODE)
```

Step 4: Run Your Code (complete code here)

```
IMPORT NETWORK
IMPORT UREQUESTS

# INITIALIZE WI-FI INTERFACE
WLAN = NETWORK.WLAN(NETWORK.STA_IF)
WLAN.ACTIVE(TRUE)

# WIFI CREDS
SSID = 'YOUR_WIFI_SSID'
PASSWORD = 'YOUR_WIFI_PASSWORD'

# CONNECT TO YOUR WI-FI:
WHILE NOT WLAN.ISCONNECTED():
   PASS
PRINT('CONNECTED TO WI-FI'))

# CALLING A PUBLICALY AVAILABLE REST API

URL = 'HTTPS://DOG.CEO/API/BREEDS/LIST/ALL' # YOU CAN USE
YOUR API URL HERE
RESPONSE = UREQUESTS.GET(URL)

# CHECKING THE RESPONSE
IF RESPONSE.STATUS_CODE == 200:
  PRINT(RESPONSE.TEXT) # PRINTING API RESPONSE
ELSE:
 PRINT('FAILED. ERROR CODE:', RESPONSE.STATUS_CODE)
REST API CALL FROM ESP32 USING MICROPYTHON
```

save above code file named as main.py and transfer to your esp32 (using thonny).
You can also directly save file using thonny to your esp32. (shown in image)



Reset esp32 and you will see the API call going from esp32, result will be printed on console (thonny' console).

Step 5:Verify the Output

Check the output on the serial console (thonny's serial console). You should see the response from the REST API!
Cheers

CHECK THE OUTPUT ON THE SERIAL CONSOLE (THONNY'S SERIAL CONSOLE). YOU SHOULD SEE THE RESPONSE FROM THE REST API! CHEERS

CHAPTER 03
HOW TO SEND EMAIL USING ESP32 WITH MICROPYTHON AND BREVO

CHAPTER 3
HOW TO SEND EMAIL USING ESP32 WITH MICROPYTHON AND BREVO

Brevo (formally Sendinblue) is widely used for its reliability and ease of integration, and offers a REST API for sending emails. In this tutorial, we will go step by step about how to send email using esp32 with micropython and brevo

What do you need

- ESP32 Module: With MicroPython firmware installed.
- Brevo Account: Sign up for a free brevo account here.
- Brevo API Key: Create an API key in your Brevo account.

Step 1: Setting Up Your Environment

1. Connect ESP32 to Your Computer
2. Use Thonny to write and save your code

Step 2: Code to Connect ESP32 to Wi-Fi

import network

```
IMPORT NETWORK

###### INITIALIZE WI-FI INTERFACE
WLAN = NETWORK.WLAN(NETWORK.STA_IF)
WLAN.ACTIVE(TRUE)


#### WIFI CREDS
SSID = '<YOUR-SSID>'
PASSWORD = '<YOUR-PASSWORD>'

PRINT(WLAN.ISCONNECTED())
TIME.SLEEP(0.5)

# CONNECT TO THE WI-FI NETWORK
WLAN.CONNECT(SSID, PASSWORD)
```

Step 3: Upload the urequests library to esp32

This library can be single file named urequests.py or a folder named "urequests" with a "__init__.py" file inside. In the code example attached I used the library with folder.

You can download the file from here and extract the init file to urequests folder inside root directory of esp32.

It should look like this:

/urequests/__init__.py

```
IMPORT UREQUESTS
IMPORT UJSON

BREVO_API_KEY = '<YOUR-API-KEY-HERE>'

URL = 'HTTPS://API.BREVO.COM/V3/SMTP/EMAIL'

HEADERS = {
   'ACCEPT': 'APPLICATION/JSON',
   'API-KEY': BREVO_API_KEY,
   'CONTENT-TYPE': 'APPLICATION/JSON',
}

DATA = {
   "SENDER": {
      "NAME": "ESP32 CLIENT",
      "EMAIL": "<YOUR-EMAIL-ID>"
   },
   "TO": [
      {
         "EMAIL": "<RECEIVER-EMAIL-ID>",
         "NAME": "<RECEIVER NAME>"
      }
   ],
   "SUBJECT": "HELLO WORLD FROM ESP32",
   "HTMLCONTENT": "<HTML><HEAD></HEAD><BODY><P>HELLO,
</P>THIS IS MY FIRST TEST EMAIL SENT FROM ESP32 USING BREVO.
</P></BODY></HTML>"
}

RESPONSE = UREQUESTS.POST(URL, DATA=UJSON.DUMPS(DATA),
HEADERS=HEADERS)
```

Complete Code for sending email from esp32

```
IMPORT UREQUESTS
IMPORT UJSON
IMPORT NETWORK
IMPORT TIME

# INITIALIZE WI-FI INTERFACE
WLAN = NETWORK.WLAN(NETWORK.STA_IF)
WLAN.ACTIVE(TRUE)

WLAN.DISCONNECT()

# WIFI CREDS
SSID = '<YOUR-SSID>'
PASSWORD = '<YOUR-PASSWORD>'

PRINT(WLAN.ISCONNECTED())
TIME.SLEEP(0.5)

# CONNECT TO THE WI-FI NETWORK
WLAN.CONNECT(SSID, PASSWORD)

PRINT('CONNECTED TO WI-FI')

BREVO_API_KEY = '<YOUR-API-KEY-HERE>'

URL = 'HTTPS://API.BREVO.COM/V3/SMTP/EMAIL'

HEADERS = {
   'ACCEPT': 'APPLICATION/JSON',
   'API-KEY': BREVO_API_KEY,
   'CONTENT-TYPE': 'APPLICATION/JSON',
}

DATA = {
   "SENDER": {
      "NAME": "ESP32 CLIENT",
      "EMAIL": "<YOUR-EMAIL-ID>"
   },
   "TO": [
      {
         "EMAIL": "<RECEIVER-EMAIL-ID>",
         "NAME": "<RECEIVER NAME>"
      }
   ],
   "SUBJECT": "HELLO WORLD FROM ESP32",
   "HTMLCONTENT": "<HTML><HEAD></HEAD><BODY><P>HELLO,
</P>THIS IS MY FIRST TEST EMAIL SENT FROM ESP32 USING BREVO.
</P></BODY></HTML>"
}

RESPONSE = UREQUESTS.POST(URL, DATA=UJSON.DUMPS(DATA),
HEADERS=HEADERS)


IF RESPONSE.STATUS_CODE == 201:
   PRINT("EMAIL SENT SUCCESSFULLY!")
ELSE:
   PRINT("FAILED TO SEND EMAIL. RESPONSE:", RESPONSE.TEXT)
RESPONSE.CLOSE()
```

Voila! this is all you need to send email from esp32. You can download this code with requests library in the file below and place on your esp32 (add your api key and emails) in places like <YOUR-API-KEY-HERE>

CHAPTER 04
PORT SCANNING USING ESP32 AND MICROPYTHON

In this chapter we will be coding our own port scanner for port scanning using esp32 and micropython. Please note that this is for educational purposes only. Scanning networks without permission can be illegal and unethical. Lets dive into a brief introduction of port scanning, why its unethical and then coding one for us.

What is Port Scanning?

Port scanning is a way to discover open spots, called ports, on a computer's network. This helps to understand which parts of the network or computer on network can talk to the outside world.

Why Port Scanning is Unethical?

Scanning ports on someone else's network is not only unethical but also illegal in many countries. Its similar to the methodically checking all doors and windows in a building to find which ones are open. Hackers usually use this method on a network or computer to see how far they can reach and what they can access on target computer or network.

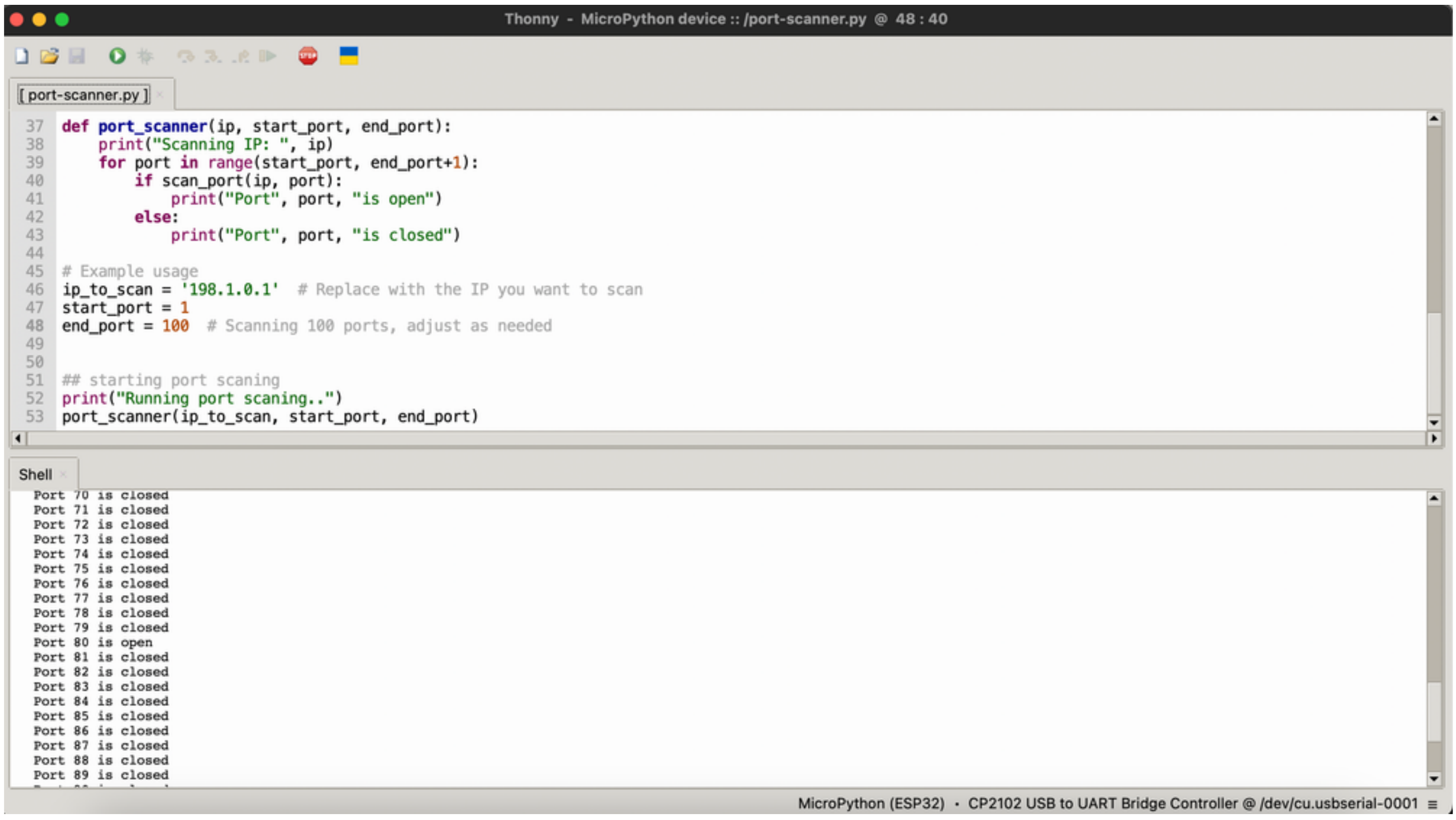
Other uses of Port Scanning

You might have question, if its unethical why the hell this information is available so easily and people like me are giving away the code to perform this scan. The one major use of port scanning is in vulnerability scanning. To perform an assessment about how vulnerable a network is to cyber attacks or security breaches.

Complete Code for Port Scanner using Micropython

```
IMPORT SOCKET
IMPORT TIME
IMPORT NETWORK

# INITIALIZE WI-FI INTERFACE
WLAN = NETWORK.WLAN(NETWORK.STA_IF)
WLAN.ACTIVE(TRUE)

# WIFI CREDS
SSID = '<YOUR WIFI SSID>'
PASSWORD = '<YOU WIFI PASSWORD>'

PRINT(WLAN.ISCONNECTED())
TIME.SLEEP(0.5)

# CONNECT TO THE WI-FI NETWORK
WLAN.CONNECT(SSID, PASSWORD)

PRINT('CONNECTED TO WI-FI')

###
PRINT(WLAN.IFCONFIG())
###

DEF SCAN_PORT(IP, PORT):
    S = USOCKET.SOCKET(USOCKET.AF_INET,USOCKET.SOCK_STREAM)
    TRY:
        S.SETTIMEOUT(5)
        S.CONNECT((IP, PORT))
        RETURN TRUE
    EXCEPT OSERROR AS E:
        # PRINT(E) ## UNCOMMENT FOR DEBUGGING
        RETURN FALSE
    FINALLY:
        S.CLOSE()

DEF PORT_SCANNER(IP, START_PORT, END_PORT):
    PRINT("SCANNING IP: ", IP)
    FOR PORT IN RANGE(START_PORT, END_PORT+1):
        IF SCAN_PORT(IP, PORT):
            PRINT("PORT", PORT, "IS OPEN")
        ELSE:
            PRINT("PORT", PORT, "IS CLOSED")

# EXAMPLE USAGE
IP_TO_SCAN = '198.1.0.1'  # REPLACE WITH THE IP YOU WANT TO SCAN
START_PORT = 1
END_PORT = 100  # SCANNING 100 PORTS, ADJUST AS NEEDED


## STARTING PORT SCANING
PRINT("RUNNING PORT SCANING..")
PORT_SCANNER(IP_TO_SCAN, START_PORT, END_PORT)
```

In above code, replace the value of ip_to_scan with the IP where you want to perform test and provide the range of ports in start_port and end_port. Save the code as main.py on you micropython enabled device.



THANK YOU
FOR DOWNLOADING OUR GUIDE!

FOR MORE EASY TIPS AND TUTORIALS
CHECK OUT OUR BLOG

WWW.KHALSALABS.COM