

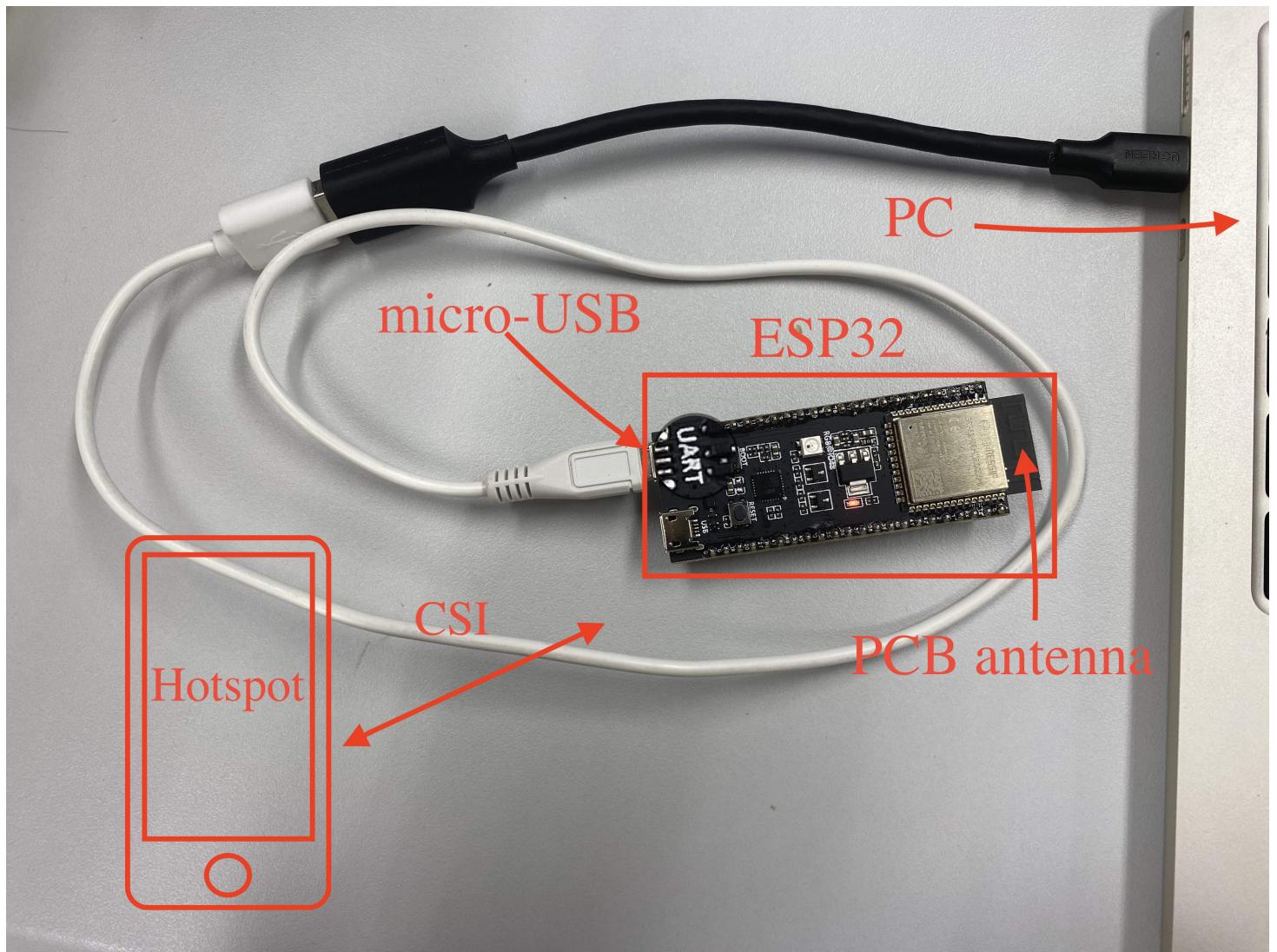
ESP32 Data Collection Manual

This tool is based on [Github repo: ESP32-CSI-Tool](#).

Requirement

We setup ESP in STA(station)-AP(Access Point) mode, by analysing CSI in-between, activities around STA-AP can be detected.

1. **STA:** ESP32-S3-DevKitC-1 v1.0
2. **AP:** A router with known password (e.g. your mobile phone hotspot)
3. micro-USB cable (connecting ESP32 to your computer).
4. Personal Computer (1 USB port)

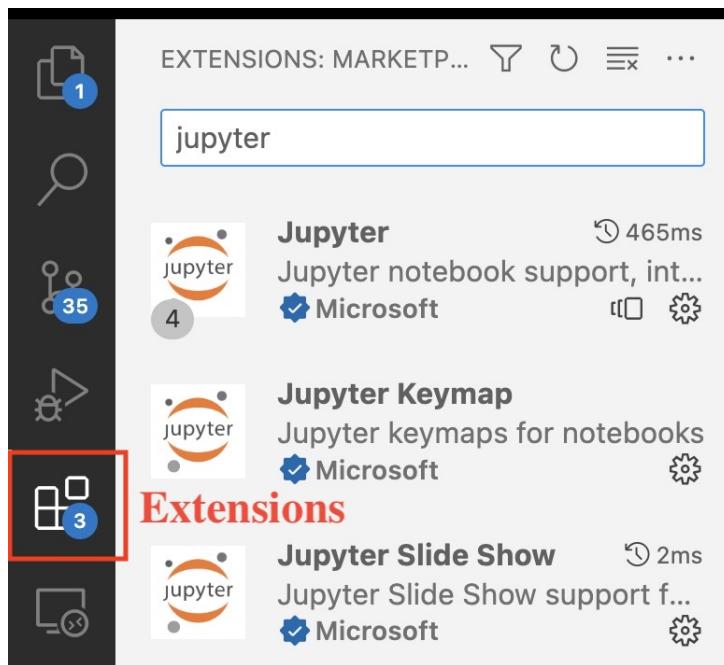


*Due to platform differences, some examples may only work in MacOS and Linux terminal. For Windows Users, please run all scripts in PowerShell .

ESPRESSIF Dev Kit installation (VS Code)

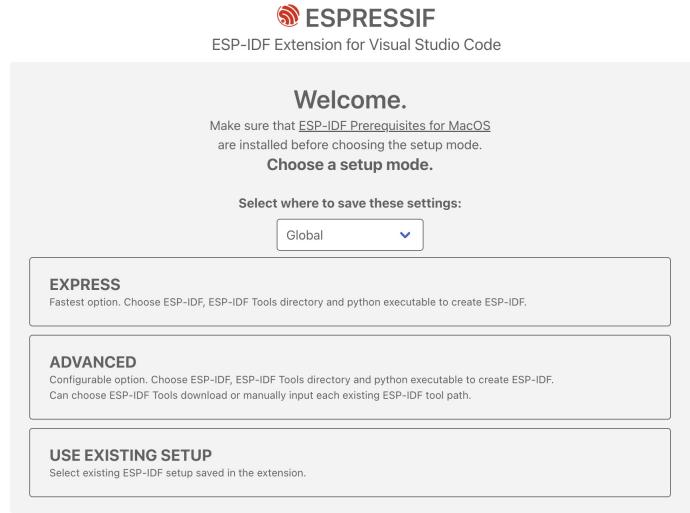
ESP-IDF on VS code is one of several methods to develop ESP32. Here are the steps:

1. Install **Visual Studio Code**, **git**, **python** on your computer.
2. Install required extensions for VS Code.
 1. Espressif IDF v1.6.0
 2. python, python environment manager, jupyter



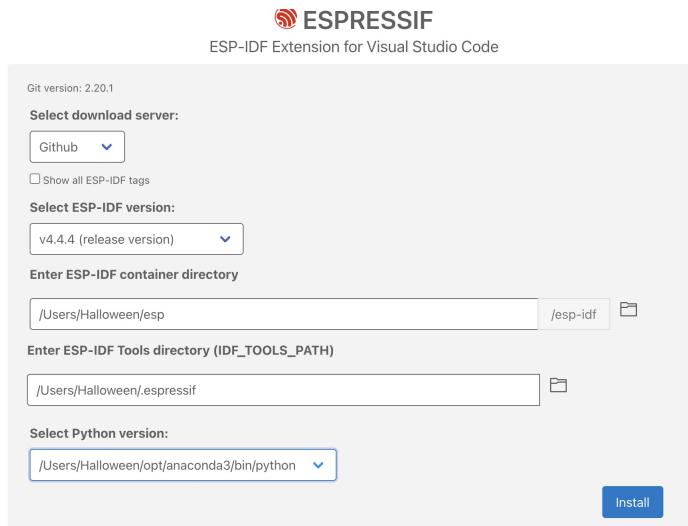
Configure ESP-IDF

1. Upon Installation, a new window for ESP-IDF will pop up. Or you can open it from View→Command Palette→*ESP-IDF: Configure ESP-IDF extension*.

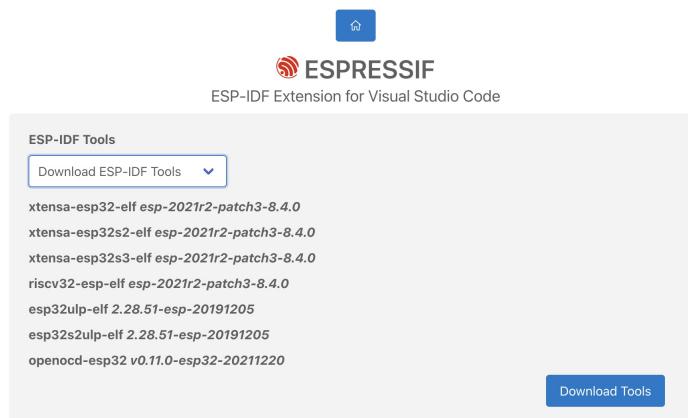


2. Click Advanced

3. For ESP-IDF version, **choose v4.4.4 (release version)** ; Leave other settings with default value.



4. Continue. Download tools.



5. Complete downloading. If there shows additional requirement for USB configuration, just do it.



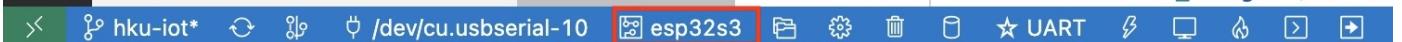
Setup VS code workspace

(Open folder:active_sto in VS code)

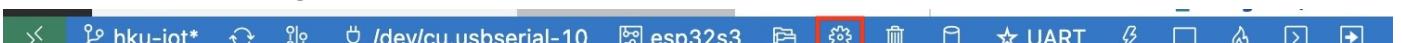
1. Connect ESP32 via USB port on your PC.
2. On the bottom bar of VSCode. Select ESP port. Its name may depend on your own device.



3. Select ESP Devices: active_sto(folder) > esp32s3 > ESP32-S3 Chip(via ESP-PROG)



4. Click ESP-IDF Configuration editor



1. enable CSI feature
2. set (SSID, password) of your personal Hotspot (or any WiFi with password, **CANNOT use HKU or WiFi.HKU via HKU**)
3. set TX rate(1000Hz)
4. enable CSI report
5. enable sending CSI to serial port. (All settings see below)

Component config

Wi-Fi

Max number of WiFi static RX buffers ⓘ

10

Max number of WiFi dynamic RX buffers ⓘ

32

Type of WiFi TX buffers ⓘ

Dynamic

Max number of WiFi dynamic TX buffers ⓘ

32

✓ WiFi CSI(Channel State Information) ⓘ

Select this option to enable CSI(Channel State Information) feature. CSI takes about CONFIG_ESP32_WIFI_STATIC_RX_BUFFER_NUM KB of RAM. If CSI is not used, it is better to disable this feature in order to save memory.

ESP32 CSI Tool Config

WiFi Channel ⓘ

6

WiFi SSID ⓘ

myssid

WiFi Password ⓘ

mypassword

Packet TX Rate ⓘ

1000

✓ Should this ESP32 collect and print CSI data? ⓘ

Allowing only a single ESP32 to collect CSI will likely increase the sampling frequency of your experiments.

✓ (Advanced users only) Should we only collect LLTF? ⓘ

✓ Send CSI data to Serial ⓘ

Send CSI data to SD ⓘ

5. Build, Compile and Flash! A monitor will show up if all work fine.



```
Generating ./partition_table/partition_table.bin
Partition table binary generated. Contents:
=====
# ESP-IDF Partition Table
# Name, Type, SubType, Offset, Size, Flags
nvs,data,nvs,0x9000,24K,
phy_init,data,phy,0xf000,4K,
factory,app,factory,0x10000,1M,
=====
[676/1057] Linking CXX static library esp-idf/mbedtls/library/libmbedtlscrypto.a
```

Building Project: Building project...

Source: Espressif IDF (Extension)

Cancel

```

esptool.py v3.3-dev
Serial port /dev/cu.usbserial-10
Connecting...
Chip is ESP32-S3
Features: WiFi, BLE
Crystal is 40MHz
MAC: 7c:df:a1:e8:8d:c0
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Auto-detected Flash size: 8MB
Flash will be erased from 0x00000000 to 0x00005fff...
Flash will be erased from 0x00010000 to 0x000defff...
Flash will be erased from 0x00008000 to 0x00008fff...
Compressed 20896 bytes to 13060...
Wrote 20896 bytes (13060 compressed) at 0x00000000 in 0.7 seconds (effective 238.8 kbit/s)...
Hash of data verified.
Compressed 846128 bytes to 488576...
Writing at 0x00091e97... (66 %)

```

Building Project: Flashing project into device...

Source: Espressif IDF (Extension)

Cancel

TERMINAL

```

I (531) pp: pp rom version: e7ae62f
I (531) net80211: net80211 rom version: e7ae62f
I (541) wifi:wifi driver task: 3fce5ae4, prio:23, stack:6656, core=0
I (541) system_api: Base MAC address is not set
I (541) system_api: read default base MAC address from EFUSE
I (561) wifi:wifi firmware version: 63017e0
I (561) wifi:wifi certification version: v7.0
I (561) wifi:config NVS flash: enabled
I (561) wifi:config nano formating: disabled
I (561) wifi:init data frame dynamic rx buffer num: 32
I (571) wifi:init management frame dynamic rx buffer num: 32
I (571) wifi:init management short buffer num: 32
I (581) wifi:init dynamic tx buffer num: 32
I (581) wifi:init static tx FG buffer num: 2
I (581) wifi:init static rx buffer size: 1600
I (591) wifi:Init static rx buffer num: 10
I (591) wifi:Init dynamic rx buffer num: 32
I (601) wifi_init: rx ba win: 6
I (601) wifi_init: tcpip mbox: 32
I (601) wifi_init: udp mbox: 6
I (611) wifi_init: tcp mbox: 6
I (611) wifi_init: tcp tx win: 5744
I (621) wifi_init: tcp rx win: 5744
I (621) wifi_init: top mss: 1440
I (621) wifi_init: WiFi IRAM OP enabled
I (631) wifi_init: WiFi RX IRAM OP enabled
I (631) phy_init: phy_version 501,94697a2,Apr 7 2022,20:49:08
I (761) wifi:mode : sta (7c:df:a1:e8:8d:c0)
I (761) wifi:enable tsf
I (761) wifi:Set ps type: 0

I (761) Active CSI collection (Station): connect to ap SSID:myssid password:mypassword
CSI will not be collected. Check `idf.py menuconfig` # > ESP32 CSI Tool Config` to enable CSIfwi not connected. waiting...
wifi not connected. waiting...
wifi not connected. waiting...
I (2811) Active CSI collection (Station): Retry connecting to the AP
wifi not connected. waiting...
wifi not connected. waiting...
I (4861) Active CSI collection (Station): Retry connecting to the AP

```

- zsh
- ESP-IDF Flash Task
- ESP-IDF Size Task
- ESP-IDF Monitor
- ESP-IDF Monitor
- ESP-IDF Monitor
- ESP-IDF Monitor

6. Make sure your hotspot is active when ESP32 finishes booting.

Once connected, ESP32 will dump CSI through serial monitor, i.e. you VS Code terminal.

If not, ESP32 will keep waiting

✓ TERMINAL

```
I (183001) Active CSI collection (Station): Retry connecting to the AP
wifi not connected. waiting...
wifi not connected. waiting...
I (185051) Active CSI collection (Station): Retry connecting to the AP
wifi not connected. waiting...
wifi not connected. waiting...
I (187091) Active CSI collection (Station): Retry connecting to the AP
wifi not connected. waiting...
wifi not connected. waiting...
I (189141) Active CSI collection (Station): Retry connecting to the AP
wifi not connected. waiting...
wifi not connected. waiting...
I (191191) Active CSI collection (Station): Retry connecting to the AP
wifi not connected. waiting...
wifi not connected. waiting...
I (193241) Active CSI collection (Station): Retry connecting to the AP
wifi not connected. waiting...
wifi not connected. waiting...
I (195281) Active CSI collection (Station): Retry connecting to the AP
wifi not connected. waiting...
□
```

7. Exit Monitor by `Ctrl+]`

Data Collection

Write a script to start terminal monitor

Previously, we are using VS code to help us bring up a monitor for ESP32 output. To collect data from ESP32 output, we can simply do as VS code does. First of all, we should know what VS code does when it starts a monitor.

Steps:

1. Click Monitor on bottom bar.



2. Exit Monitor mode when a few lines show up.

3. Copy the first few lines as a `start_monitor.sh` file.

```
export IDF_PATH=/Users/Halloween/esp/esp-idf
/Users/Halloween/.espresif/python_env/idf4.4_py3.7_env/bin/python /Users/Halloween/esp/esp-idf/tools/idf_monitor.py -p /dev/cu.usbserial-10 -b 115200 --toolchain-prefix xtensa-esp32s3-elf- --target esp32s3 /Users/Halloween/Documents/Octosense/esp32-csi-tool/active_sto/build/active-sta.elf
(base) + active_sta.git:(hku-iot) x export IDF_PATH=/Users/Halloween/esp/esp-idf
```

Run in any terminal by typing `sh start_monitor.sh`.

4. Collect CSI by running:

```
# sh start_monitor.sh | grep <keywords> > <your-filename>
# macOS or Linux
sh start_monitor.sh | grep "CSI_DATA" > data/my-experiment-file.csv

# Windows
sh start_monitor.sh | findstr "CSI_DATA" > data/my-experiment-file.csv
```

5. CSI data will be saved as CSV file with all available fields ([descriptions](#)), including type,role,mac,rssi,rate,sig_mode,mcs,bandwidth,smoothing,not_sounding,aggregation,stbc,fec_coding,sgi,noise_floor,ampdu_cnt,channel,secondary_channel,**local_timestamp**,ant,sig_len,rx_state,real_time_set,real_timestamp.

We will parse CSI data from it later in Jupyter notebook.

Basic data Processing

See `python_util/parse_csi.ipynb`

Other useful material

1. [ESP-IDF Official Doc for ESP32-S3](#)
 2. [ESP32-CSI-Tool Github](#)
 3. [ESP-IDF CSI Description](#)