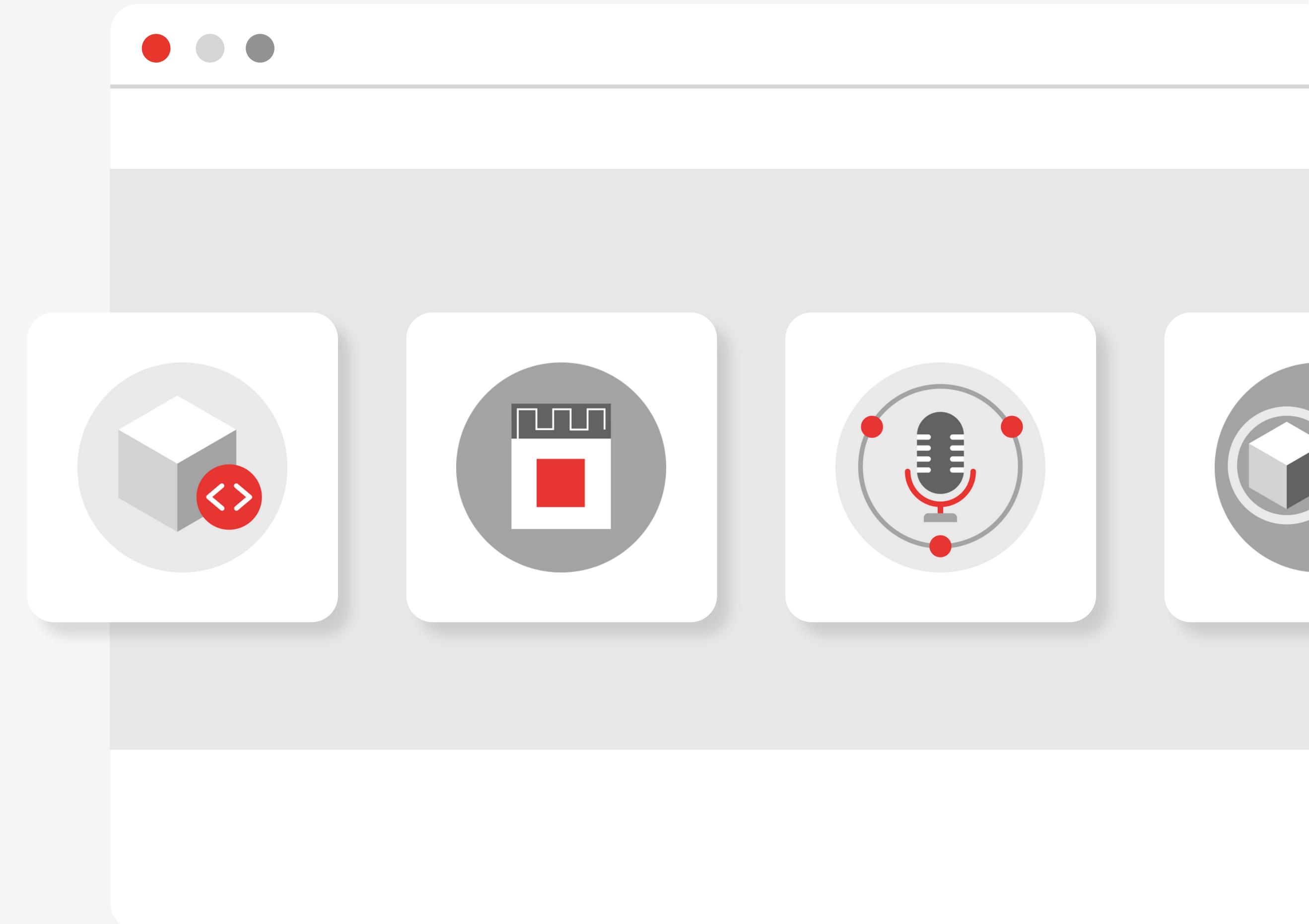


Espressif ESP-IDF IoT Workshop

Prepared for SUTD WTH 2023

24-25 August 2023





Jeroen Domburg
Senior Software &
Technical Marketing
Manager



Access to Slide Deck

- Access this slide deck from your device to follow along
- Get URLs to documentation and code

<https://tinyurl.com/wth-esp-idf/>



Contents

01 Introduction

02 What is AIoT?

03 Intro to ESP-IDF

04 ESP-IDF: Hello world (hands-on)

05 VScode: Blink (hands-on)

Contents

06 ESP RainMaker

07 VScode: Multi Device (hands-on)

08 Product Demonstration

01

Introduction

Our company, products, and what are we doing today?

Our Company

- Started in 2008 in China
- Offices in China, Czech Republic, India, Singapore and Brazil
- Well known for cutting-edge wireless communication, low-power, AIoT solutions
- Open-source code and documentation

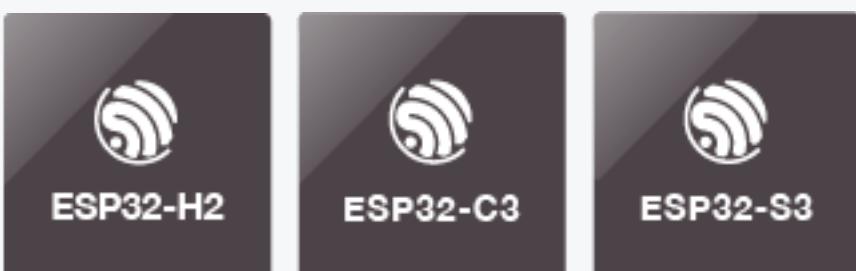


Our Products

Highly integrated, low-power, wireless connectivity, cost-effective AIoT solutions



Hardware



SDKs



ESP-IDF



Cloud
Connectivity



ESP
Jumpstart



ESP
SOLUTION
AUDIO



ESP
SDK
HomeKit



AI/DSP



ESP
Arduino



ESP
SDK
IoT



ESP
SDK
MESH



Solutions

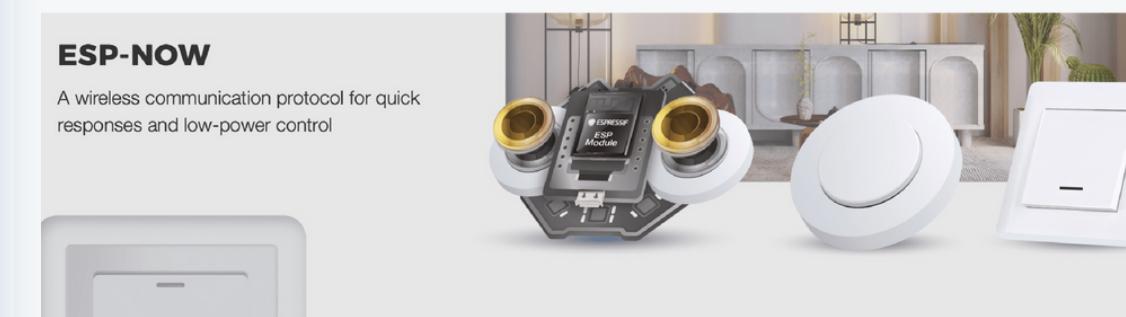


ESP-WHO Development Framework



ESP-NOW

A wireless communication protocol for quick responses and low-power control



Ecosystem



ESP
RAINMAKER®

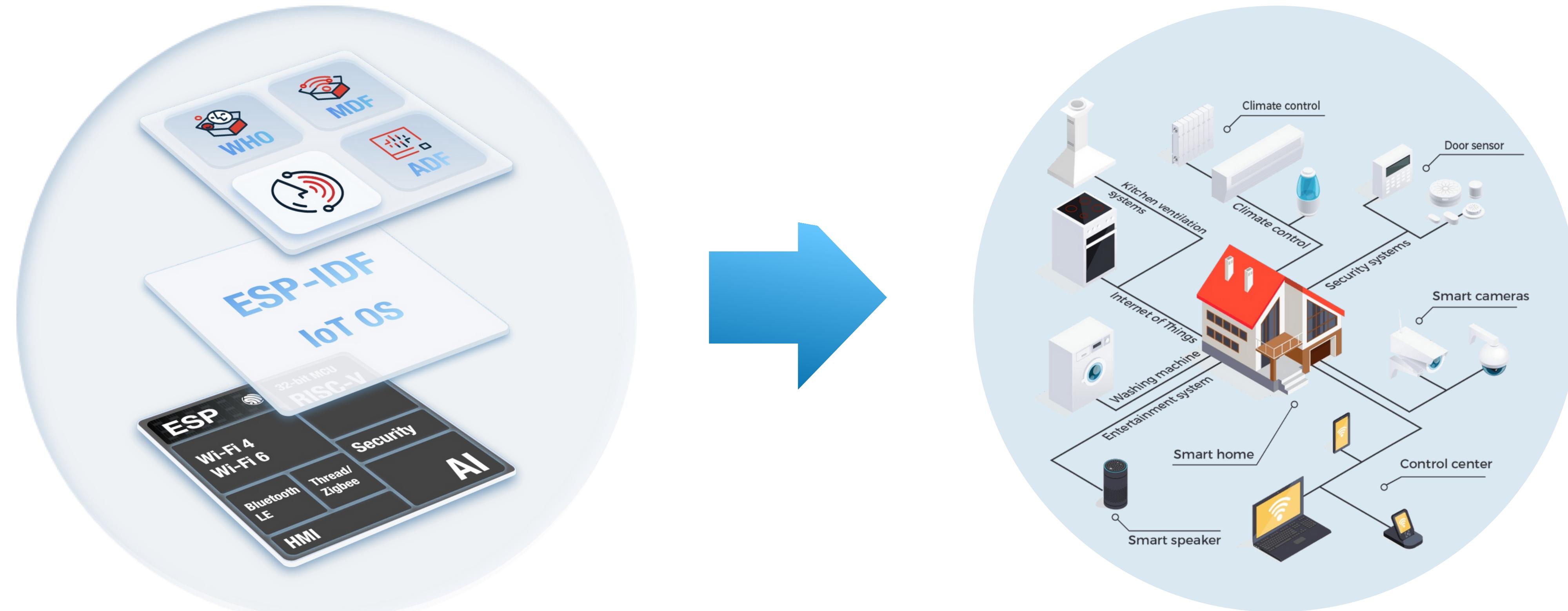


ESP
INSI
GHTS



Complete AIoT Solution Provider

From chip to embedded software to production-ready AIoT solutions



02

What is Alot?

AI + IoT = Alot

Face
recognition

Camera at your
house door
with live video
feed

Informs user's phone
when a family
member or a guest is
at the door

'Smart cities'
'Edge-based vs cloud-based'

'Autonomous vehicles'

'Smart homes'

03

Intro to ESP-IDF

Espressif's official IoT Development Framework

What is ESP-IDF?

- ESP-IDF is Espressif's official IoT Development Framework
- Provides self-sufficient SDK for any generic application
- Based on C

Do you know?

ESP-IDF currently powers millions of devices in the field and enables building a variety of network-connected products, ranging from simple light bulbs and toys to big appliances and industrial devices.





Open-source

Faster time to market without sacrificing customization.

Drastically simplify code complexity using smart-home products SDKs.

Supports large number of software components and SDKs

Support for VScode and Eclipse IDEs

Feature-Rich Software Components



ESP-IDF

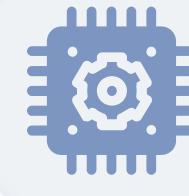
Production-Ready

Maintains stable release, well-defined release process and a support policy

Rigorous QA process to ensure production readiness

Extension documentation on software, usage and design level

Over 100 examples to learn and start with
Documentation and Examples



04

ESP-IDF: Hello world^(hands-on)

ESP-IDF Installation, connect ESP32-S3 Dev Kit,
flash ‘hello world’ example

Installing Driver

Google: Silicon Labs CP210x Driver

- Download driver
- Unzip folder
- Plug in your board (left port)
- Goes to device manager
- Right click
- Update driver
- Browse to the unzipped folder earlier

The screenshot shows the Silicon Labs website with the following details:

- Header:** SILICON LABS logo, followed by navigation links: Products, Applications, Ecosystems, Resources, and Company.
- Breadcrumbs:** Home // Developers // USB to UART Bridge VCP Drivers
- Section Headers:** OVERVIEW, DOWNLOADS (underlined), TECH DOCS, COMMUNITY & SUPPORT
- Section:** Download and Install VCP Drivers
- Description:** Downloads for Windows, Macintosh, Linux and Android below.
- Note:** *Note: The Linux 3.x.x and 4.x.x version of the driver is maintained in the current Linux 3.x.x and 4.x.x tree at www.kernel.org.
- Section:** Software Downloads
- Software Category:** Software (10) and Software · 10
- Software List:**
 - CP210x Universal Windows Driver (v11.3.0, 6/24/2023)
 - CP210x VCP Mac OSX Driver (v6.0.2, 10/27/2021)

ESP-IDF: Hello world(hands-on)

1. Go to website

<https://docs.espressif.com/projects/esp-idf/en/v5.1/esp32s3/get-started/index.html>



The screenshot shows the official ESP-IDF Programming Guide website. At the top, there's a navigation bar with the ESPRESSIF logo, dropdown menus for 'ESP32-S3' and 'stable (v5.1)', and a search bar. Below the header, a sidebar on the left contains links to 'Get Started', 'Introduction', 'What You Need', 'Installation', 'Build Your First Project', 'API Reference', 'Hardware Reference', 'API Guides', 'Migration Guides', 'Libraries and Frameworks', 'Contributions Guide', 'ESP-IDF Versions', 'Resources', 'Copyrights and Licenses', 'About', and 'Switch Between Languages'. The main content area is titled 'Get Started' and includes a note in Chinese. It explains that the document helps set up the development environment for ESP32-S3 hardware and provides examples for menu configuration, building, and flashing. A 'Note' section at the bottom states that this is for version v5.1 and other versions are available. Below the note, the 'Introduction' section is expanded, stating that ESP32-S3 is a system-on-chip with various features like Wi-Fi, Bluetooth, and multiple cores.

This document is intended to help you set up the software development environment for the hardware based on the ESP32-S3 chip by Espressif. After that, a simple example will show you how to use ESP-IDF (Espressif IoT Development Framework) for menu configuration, then for building and flashing firmware onto an ESP32-S3 board.

Note

This is documentation for stable version v5.1 of ESP-IDF. Other [ESP-IDF Versions](#) are also available.

Introduction

ESP32-S3 is a system on a chip that integrates the following features:

- Wi-Fi (2.4 GHz band)
- Bluetooth Low Energy
- Dual high performance Xtensa® 32-bit LX7 CPU cores
- Ultra Low Power co-processor running either RISC-V or FSM core
- Multiple peripherals
- Built-in security hardware
- USB OTG interface
- USB Serial/JTAG Controller

ESP-IDF: Hello world (hands-on)

2. Manual Installation

ESP-IDF Programming Guide

ESPRESSIF

ESP32-S3

stable (v5.1)

Search docs

Get Started

- Introduction
- What You Need
- Installation
- Build Your First Project

API Reference

Hardware Reference

API Guides

Migration Guides

Libraries and Frameworks

Contributions Guide

ESP-IDF Versions

Resources

Copyrights and Licenses

About

Switch Between Languages



Installation

To install all the required software, we offer some different ways to facilitate this task. Choose from one of the available options.

IDE

Note

We highly recommend installing the ESP-IDF through your favorite IDE.

- Eclipse Plugin
- VSCode Extension

Manual Installation

For the manual procedure, please select according to your operating system.

- Windows Installer
- Linux and macOS

Build Your First Project

If you already have the ESP-IDF installed and not using IDE, you can build your first project from the command line following the [Start a Project on Windows](#) or [Start a Project on Linux and macOS](#).

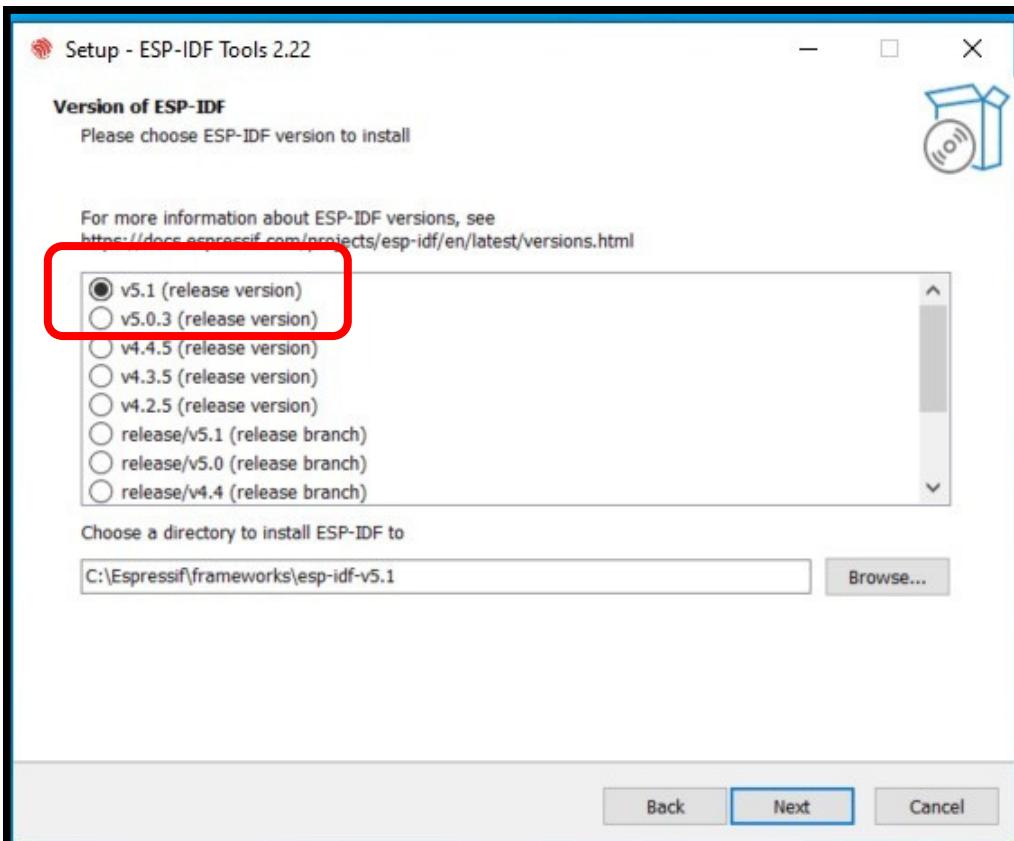
Provide feedback about this document

Previous

Next

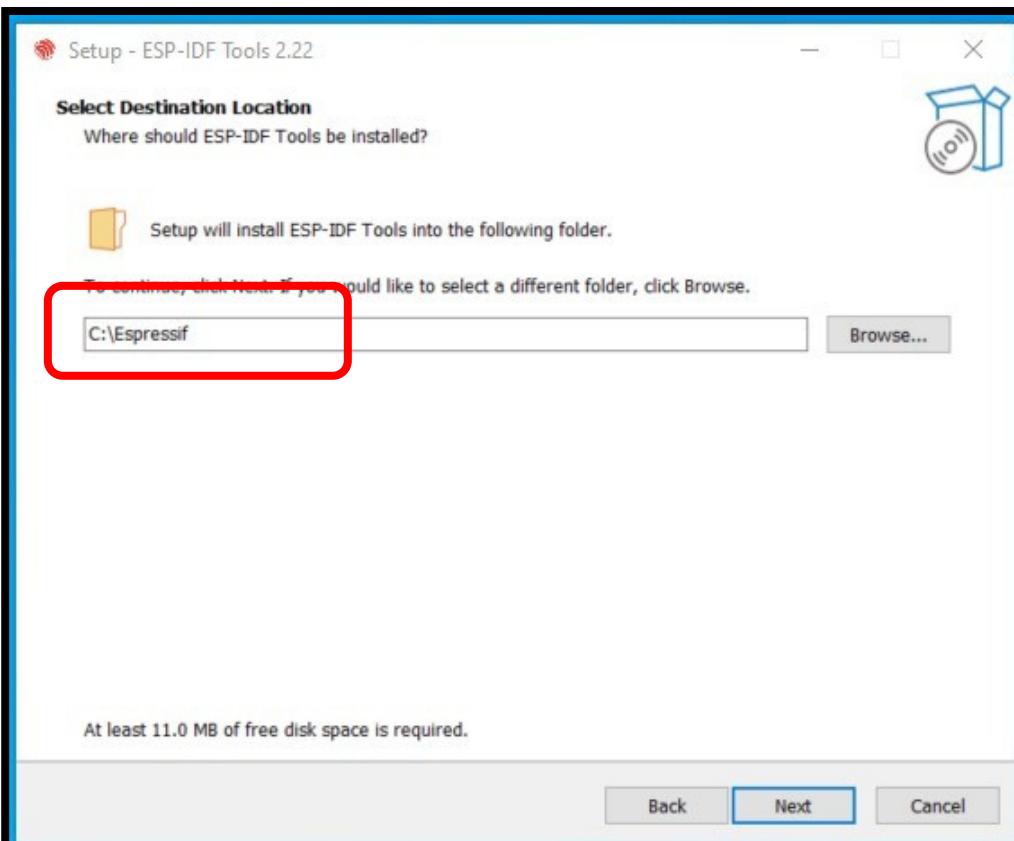
ESP-IDF: Hello world (hands-on)

3. Install on desktop



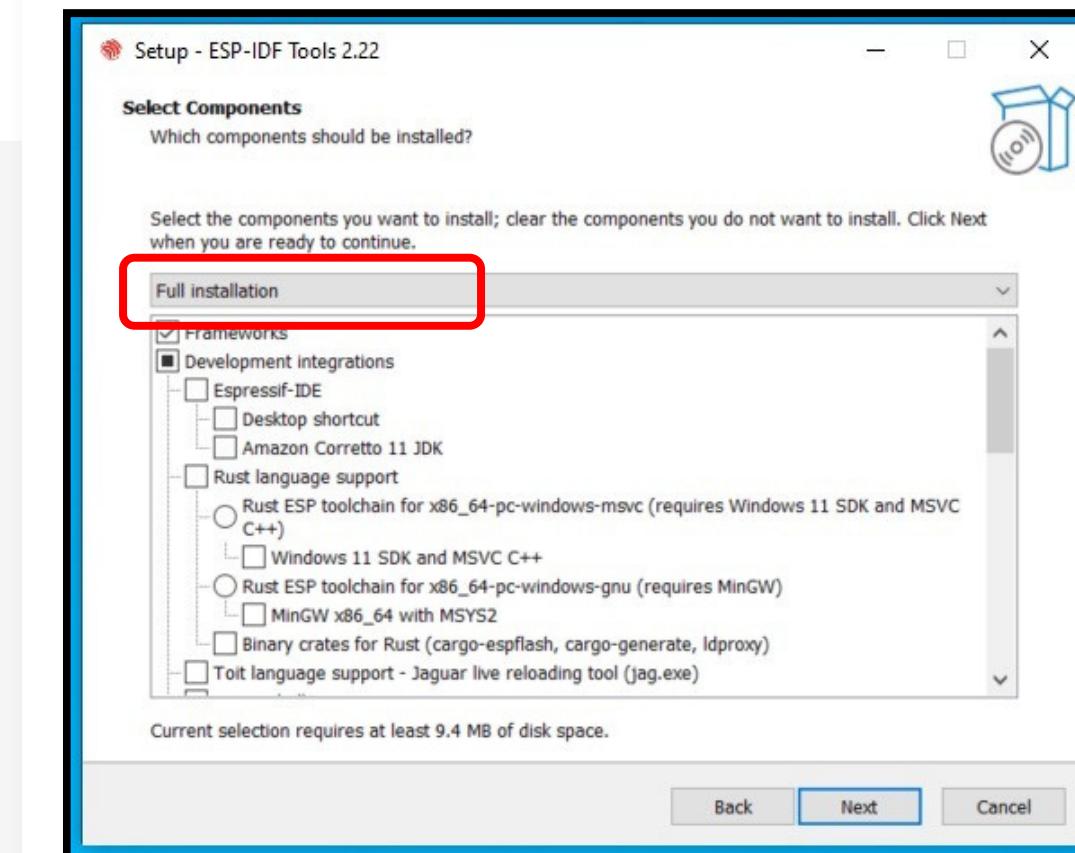
Select 'v5.1
(release version)'

Note
Do not choose the
release branch.

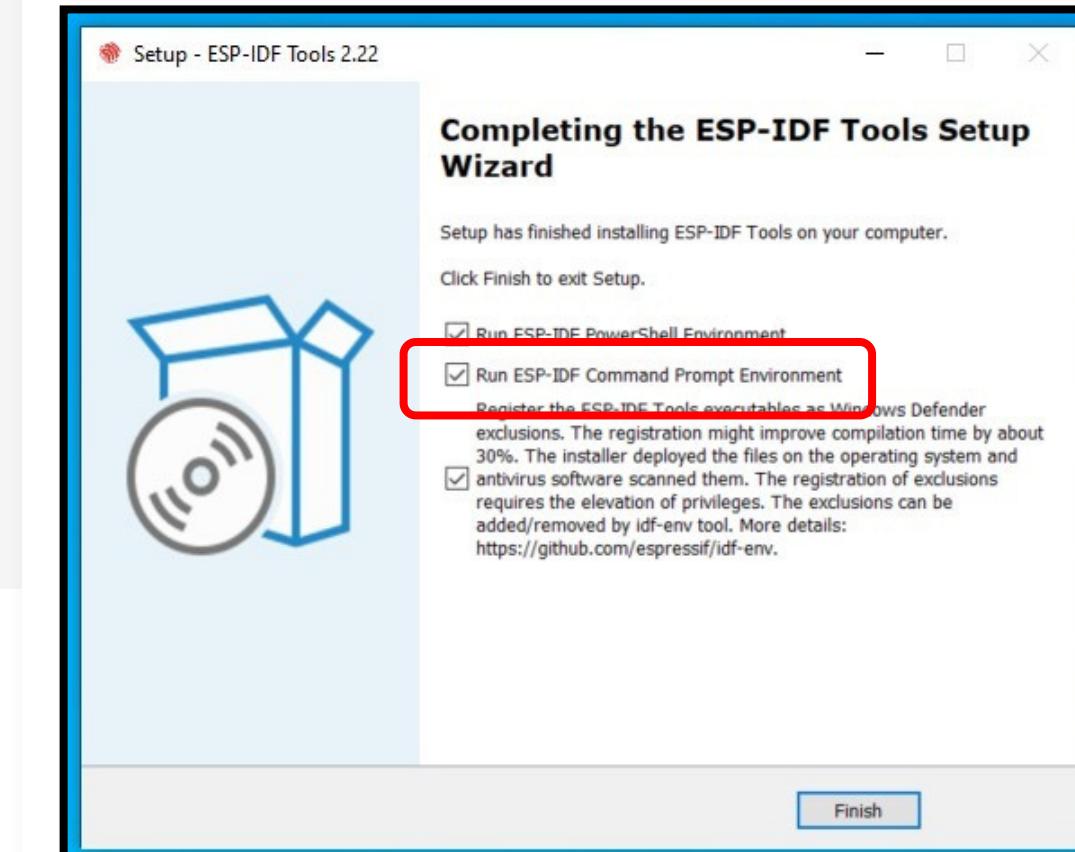


Select destination

Note
The path CANNOT have
any spaces!



Select
Full installation

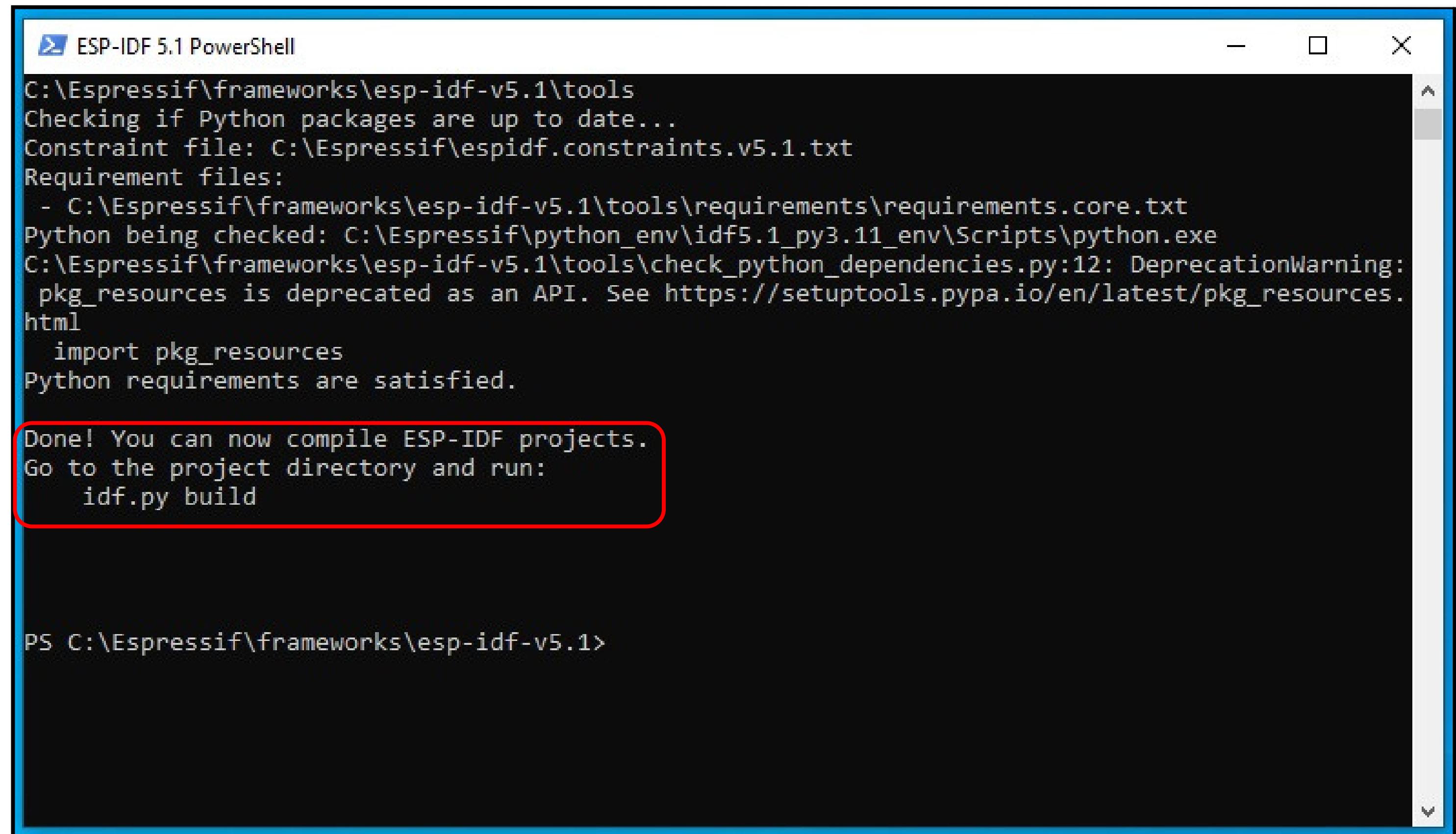


Uncheck
command prompt
(you only need to open either
one of the terminal)

ESP-IDF: Hello world (hands-on)

4. Launch ESP-IDF on PowerShell (terminal)

From the previous step,
ESP-IDF PowerShell
should run automatically



```
ESP-IDF 5.1 PowerShell
C:\Espressif\frameworks\esp-idf-v5.1\tools
Checking if Python packages are up to date...
Constraint file: C:\Espressif\espidf.constraints.v5.1.txt
Requirement files:
- C:\Espressif\frameworks\esp-idf-v5.1\tools\requirements\requirements.core.txt
Python being checked: C:\Espressif\python_env\idf5.1_py3.11_env\Scripts\python.exe
C:\Espressif\frameworks\esp-idf-v5.1\tools\check_python_dependencies.py:12: DeprecationWarning:
pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html
import pkg_resources
Python requirements are satisfied.

Done! You can now compile ESP-IDF projects.
Go to the project directory and run:
    idf.py build
```

Tips

From here, you can run various
ESP-IDF commands. Check out
the documentation here!

ESP-IDF: Hello world (hands-on)

Good to know!

From your terminal, you can access menuconfig to configure project specific variables.

We will skip this for now, as ‘hello world’ runs on default configuration.

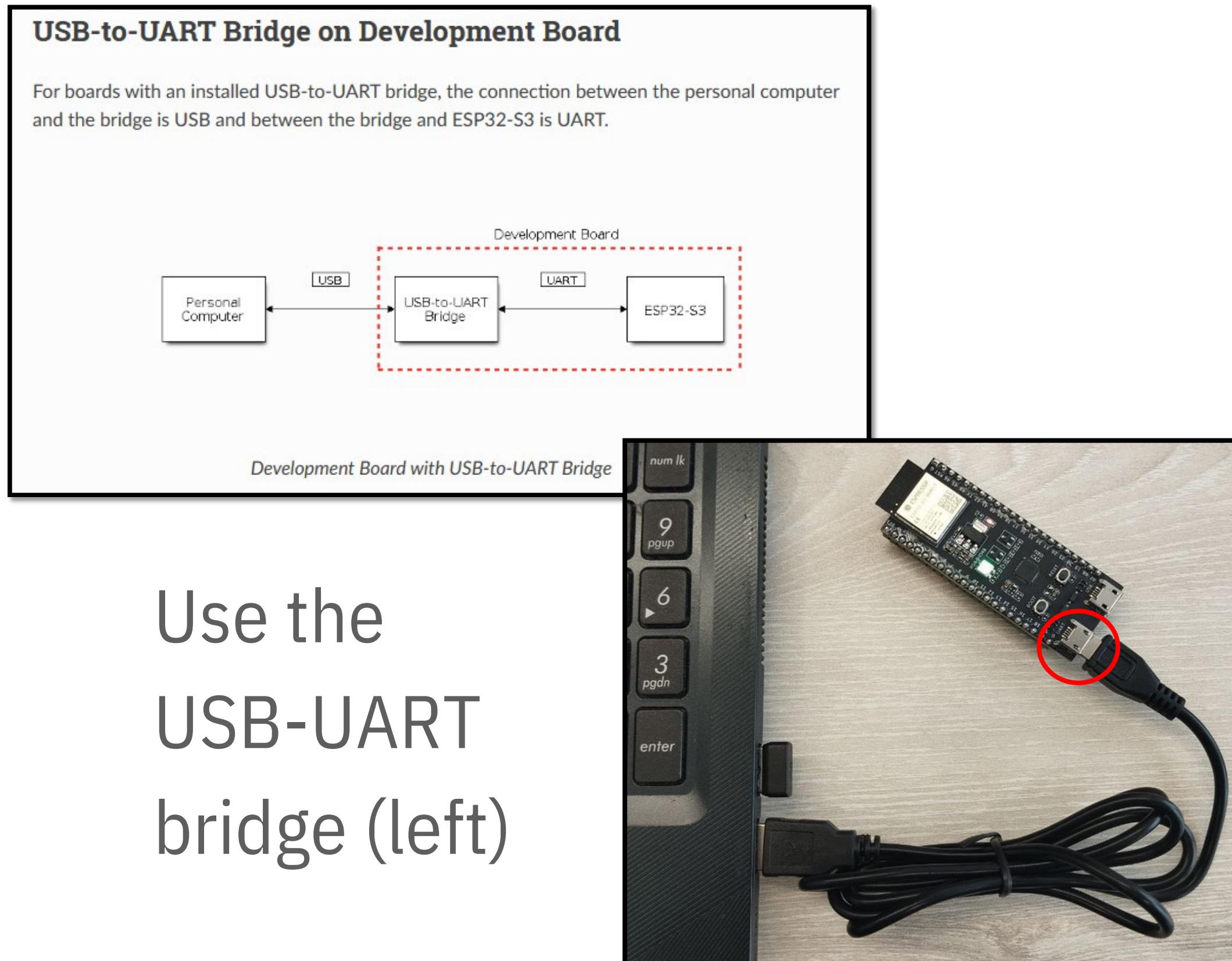
The screenshot shows the official ESP-IDF Programming Guide website. On the left, there's a sidebar with links like 'Get Started', 'Installation', 'Build Your First Project', 'API Reference', and 'Serial flasher config'. The main content area has a heading 'Configure Your Project' with instructions to navigate to the 'hello_world' directory and run 'menuconfig'. Below this, a terminal window shows the command:

```
cd %userprofile%\esp\hello_world  
idf.py set-target esp32s3  
idf.py menuconfig
```

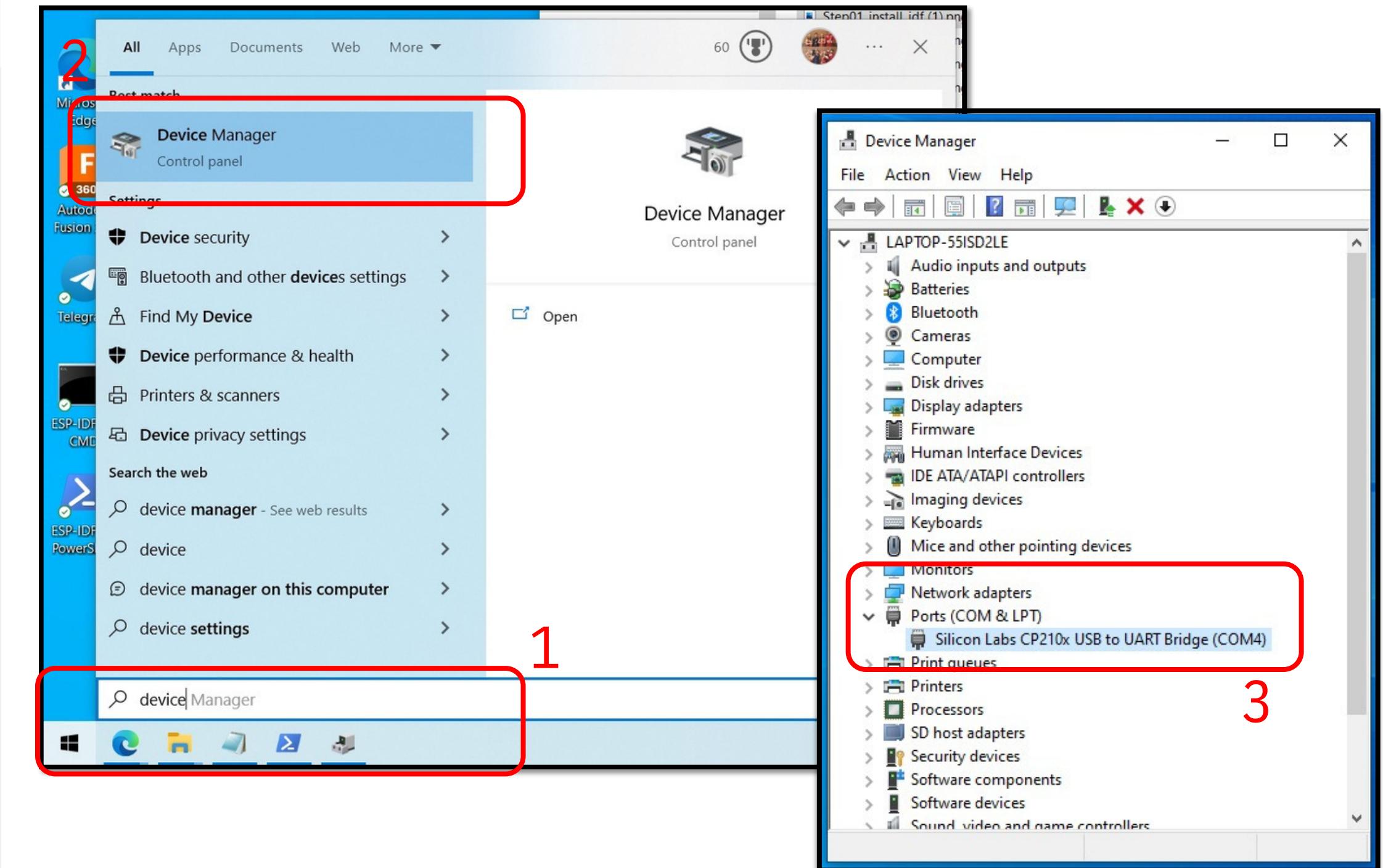
This terminal window is highlighted with a red border. To the right of the terminal, there's a note about setting the target chip in the environment variable. Below the terminal, it says 'If the previous steps have been done correctly, the following menu appears:' followed by a screenshot of the 'Serial flasher config' menu in the ESP-IDF configuration tool. The menu includes options like 'SDK tool configuration', 'Build type', 'Application manager', etc. At the bottom of the configuration window, there's a keymap for navigating the menu.

ESP-IDF: Hello world(hands-on)

5. Connect Devkit



6. Identify COM Port



Plug/unplug dev board to see which
COM port its connected to

ESP-IDF: Hello world (hands-on)

- 7a. Navigate to 'examples\hello_world' in PowerShell
- 7b. Setup builds and configuration for your chip

Use `idf.py set-target esp32s3`

```
PS C:\Users\kaitk\OneDrive\Desktop\myexamples> xcopy /e /i C:\Espressif\frameworks\esp-idf-v5.1\examples\get-started\hello_world hello_world  
C:\Espressif\frameworks\esp-idf-v5.1\examples\get-started\hello_world\CMakeLists.txt  
C:\Espressif\frameworks\esp-idf-v5.1\examples\get-started\hello_world\pytest_hello_world.py  
C:\Espressif\frameworks\esp-idf-v5.1\examples\get-started\hello_world\README.md  
C:\Espressif\frameworks\esp-idf-v5.1\examples\get-started\hello_world\ sdkconfig.ci  
C:\Espressif\frameworks\esp-idf-v5.1\examples\get-started\hello_world\main\ CMakeLists.txt  
C:\Espressif\frameworks\esp-idf-v5.1\examples\get-started\hello_world\main\hello_world_main.c  
6 File(s) copied  
PS C:\Users\kaitk\OneDrive\Desktop\myexamples> cd hello_world  
PS C:\Users\kaitk\OneDrive\Desktop\myexamples\hello_world> idf.py set-target esp32s3
```

- 7c. Build, and flash

Use `idf.py build`

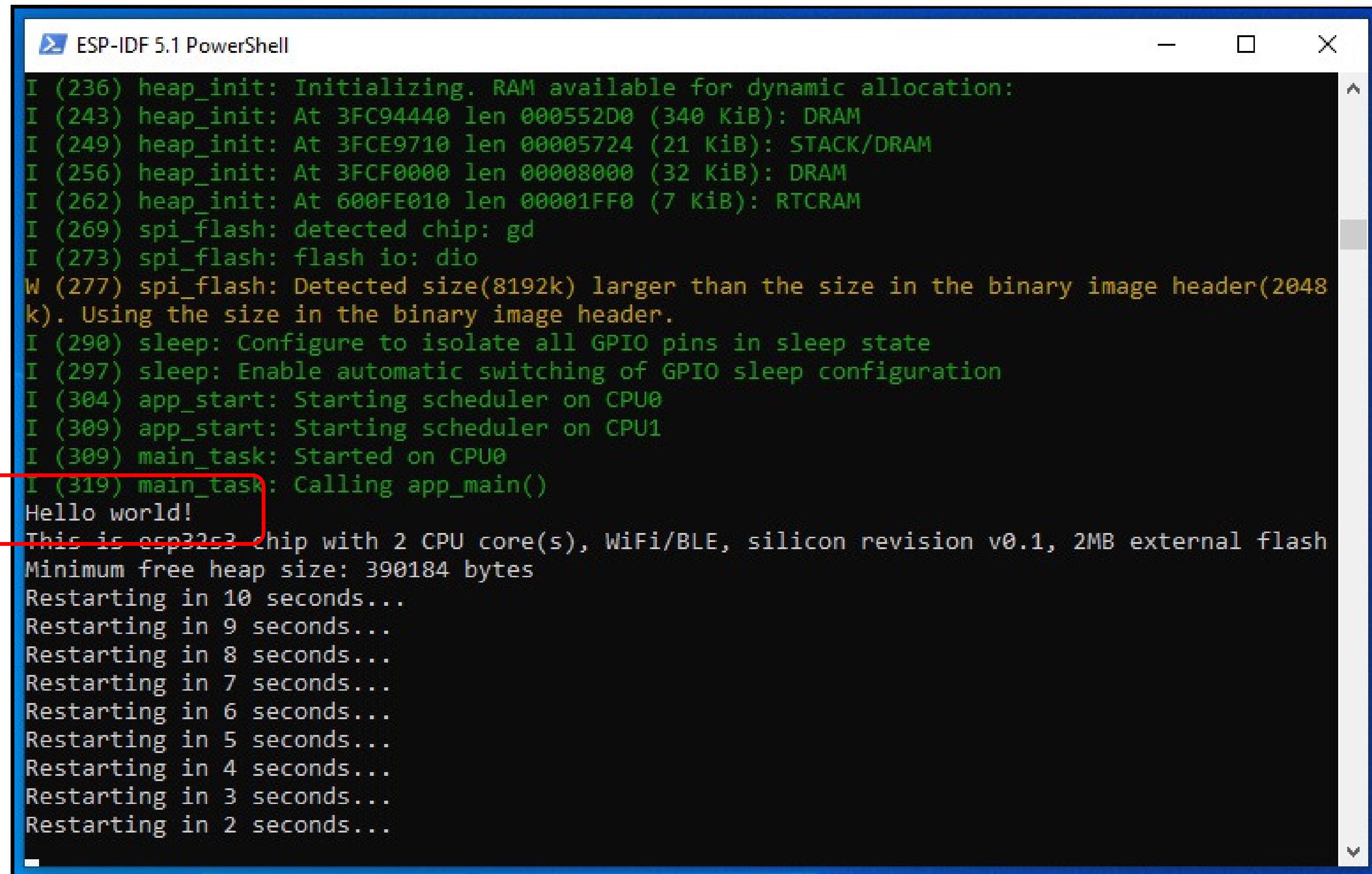
Use `idf.py -p (PORT) flash monitor`

Tips

Eg, `idf.py -p COM4 flash`

ESP-IDF: Hello world(hands-on)

DONE!



```
ESP-IDF 5.1 PowerShell
I (236) heap_init: Initializing. RAM available for dynamic allocation:
I (243) heap_init: At 3FC94440 len 000552D0 (340 KiB): DRAM
I (249) heap_init: At 3FCE9710 len 00005724 (21 KiB): STACK/DRAM
I (256) heap_init: At 3FCF0000 len 00008000 (32 KiB): DRAM
I (262) heap_init: At 600FE010 len 00001FF0 (7 KiB): RTCRAM
I (269) spi_flash: detected chip: gd
I (273) spi_flash: flash io: dio
W (277) spi_flash: Detected size(8192k) larger than the size in the binary image header(2048k). Using the size in the binary image header.
I (290) sleep: Configure to isolate all GPIO pins in sleep state
I (297) sleep: Enable automatic switching of GPIO sleep configuration
I (304) app_start: Starting scheduler on CPU0
I (309) app_start: Starting scheduler on CPU1
I (309) main_task: Started on CPU0
I (319) main_task: Calling app_main()
Hello world!
This is esp32c3 chip with 2 CPU core(s), WiFi/BLE, silicon revision v0.1, 2MB external flash
Minimum free heap size: 390184 bytes
Restarting in 10 seconds...
Restarting in 9 seconds...
Restarting in 8 seconds...
Restarting in 7 seconds...
Restarting in 6 seconds...
Restarting in 5 seconds...
Restarting in 4 seconds...
Restarting in 3 seconds...
Restarting in 2 seconds...
```

Hands-on Time!

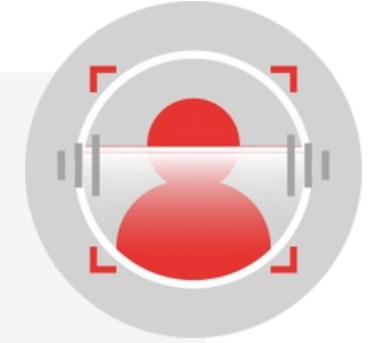


Download and install ESP-IDF

Navigate to ‘hello world’ example

Connect ESP32-S3-DevkitC-N8R8

Build, flash, monitor



05

VScode: Blink (hands-on)

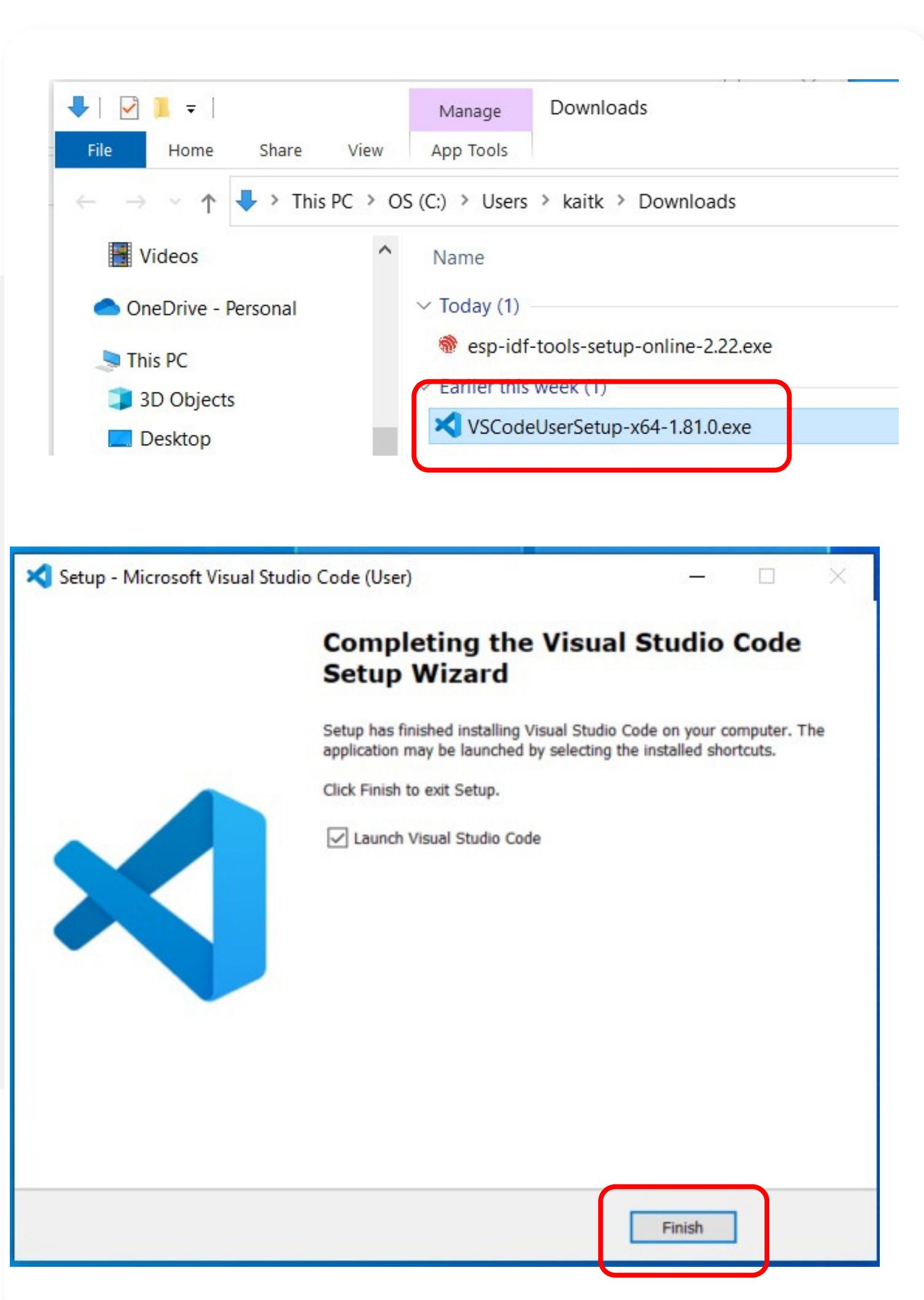
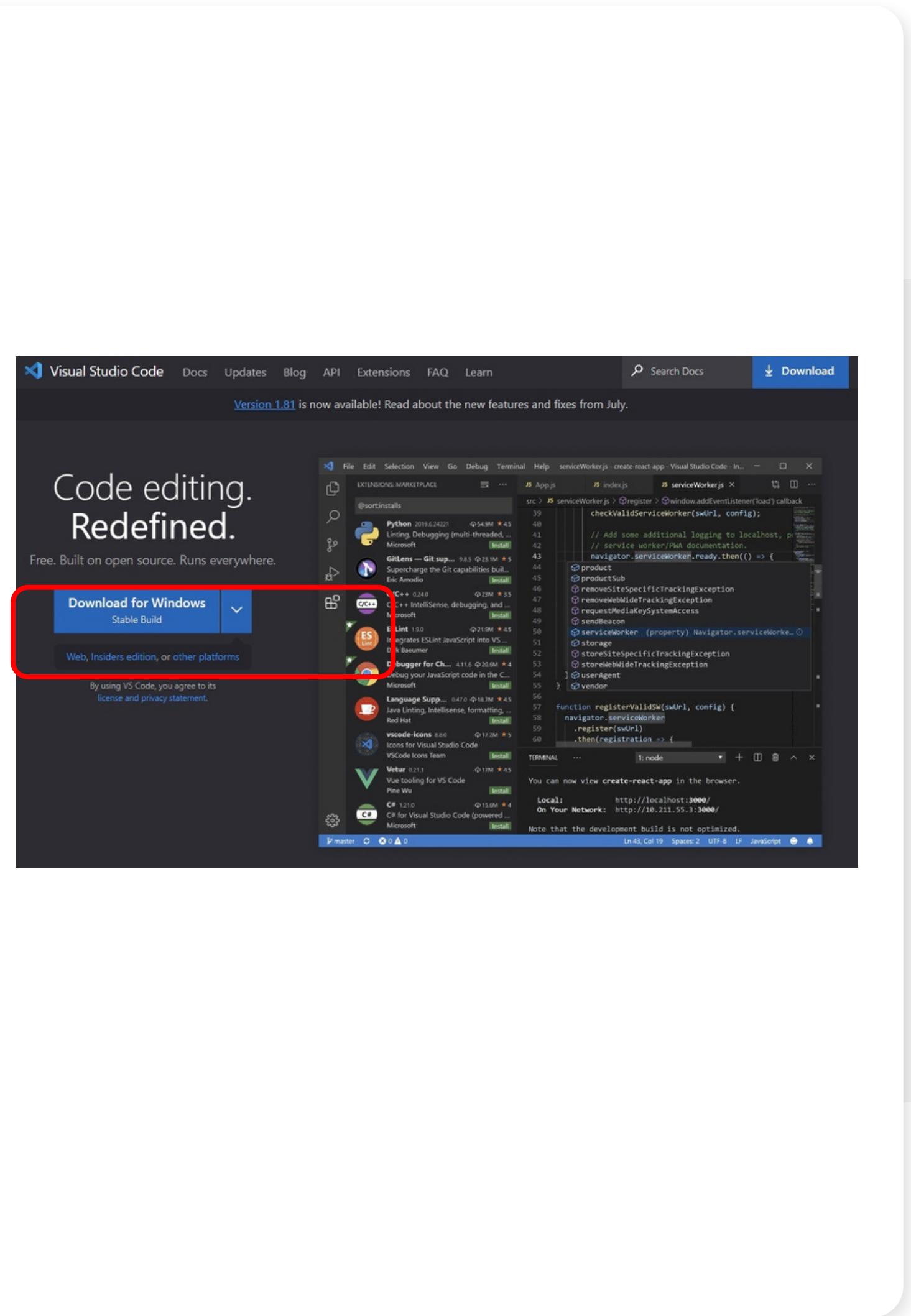
Download VScode, download Espressif IDF extension, Install, copy and flash ‘blink’ example

VScode: Blink (hands-on)

1. Download VScode

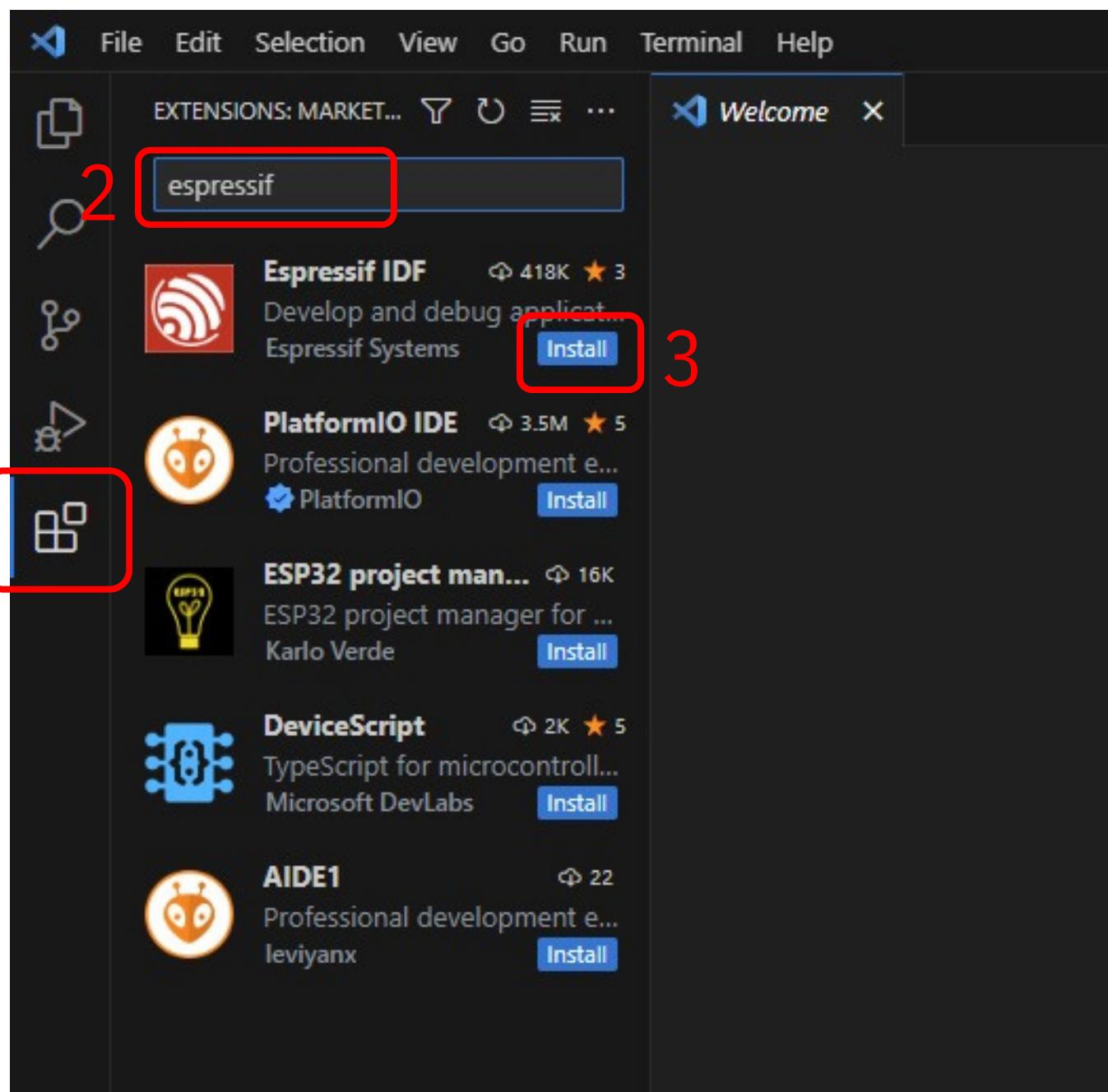


<https://code.visualstudio.com>

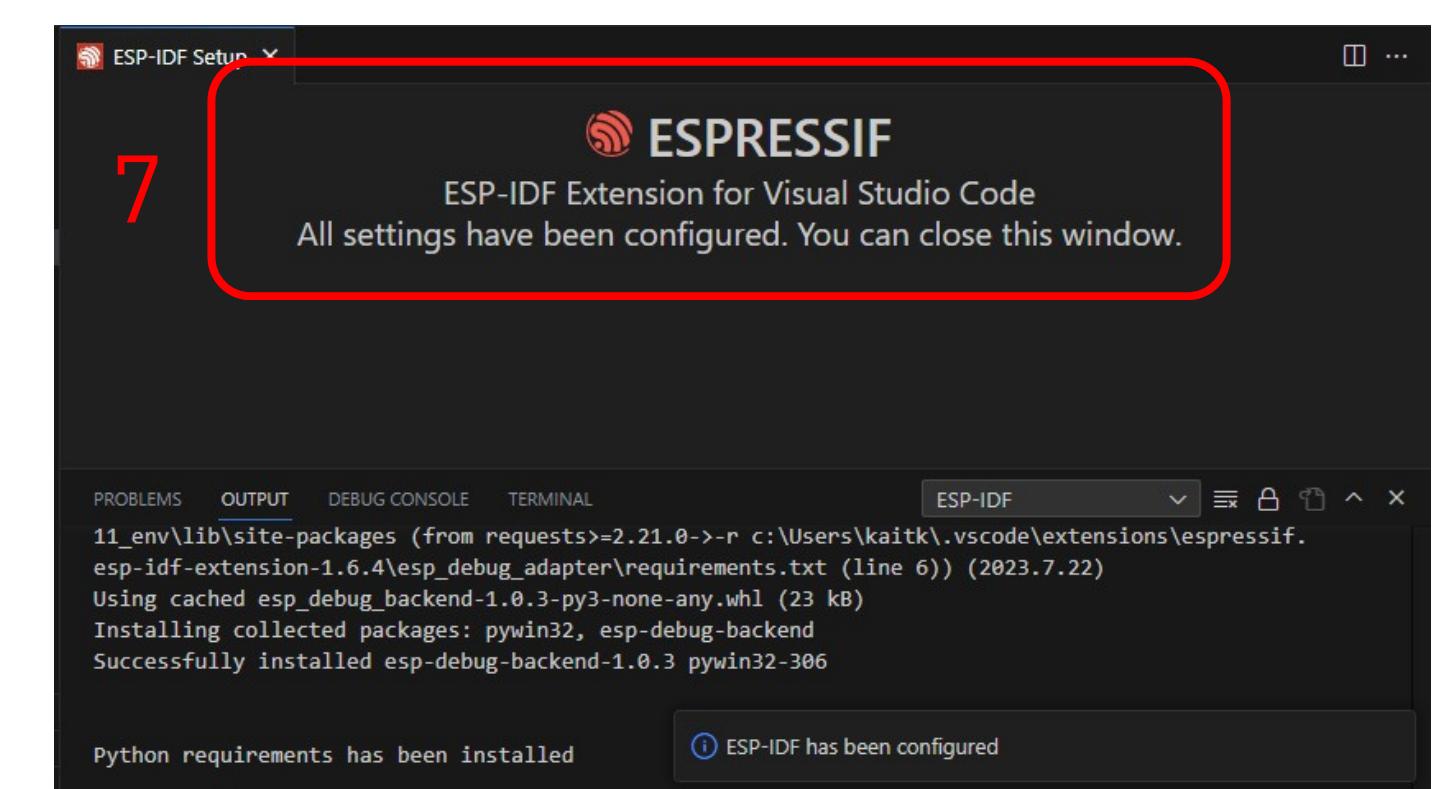
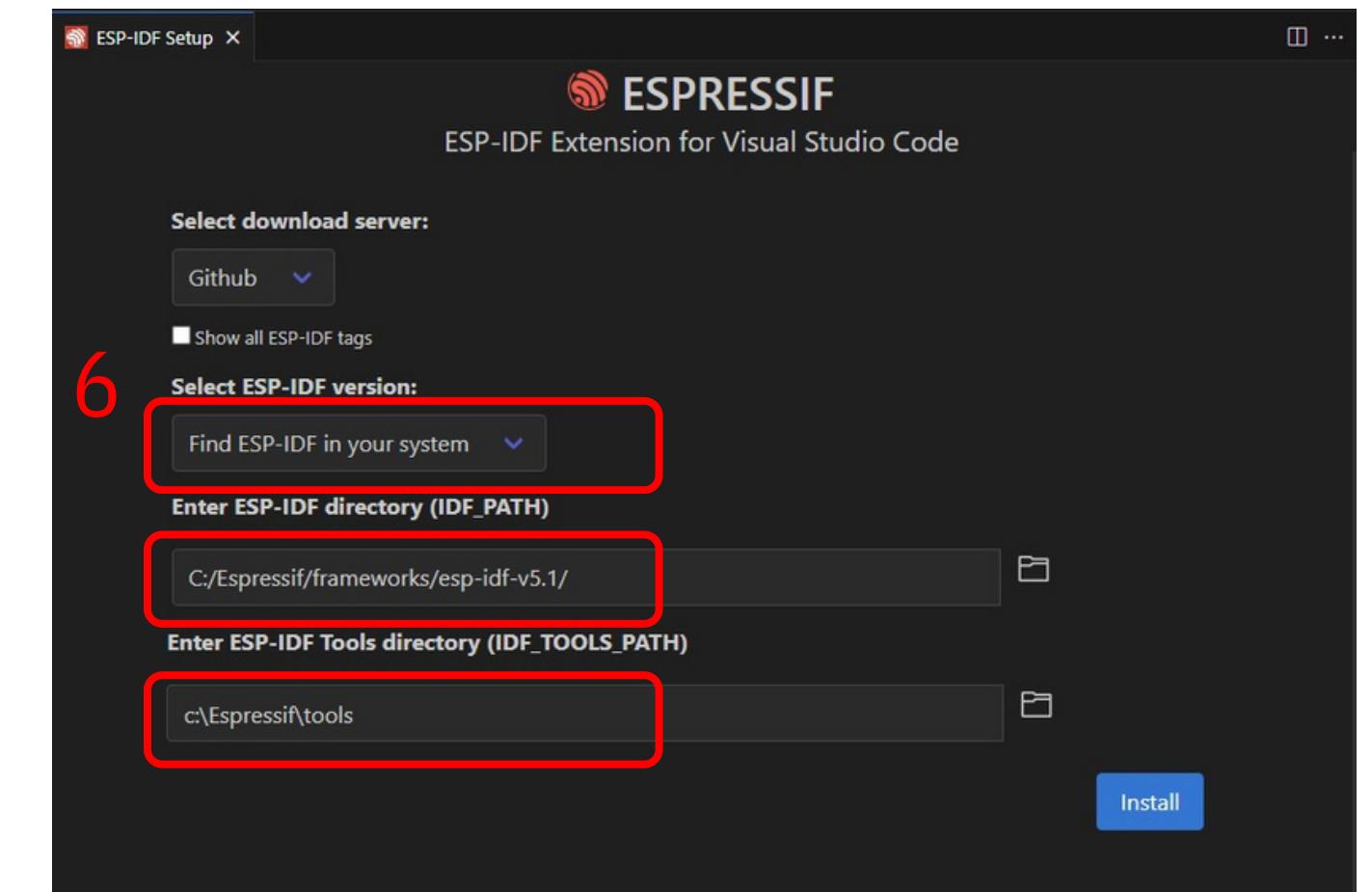
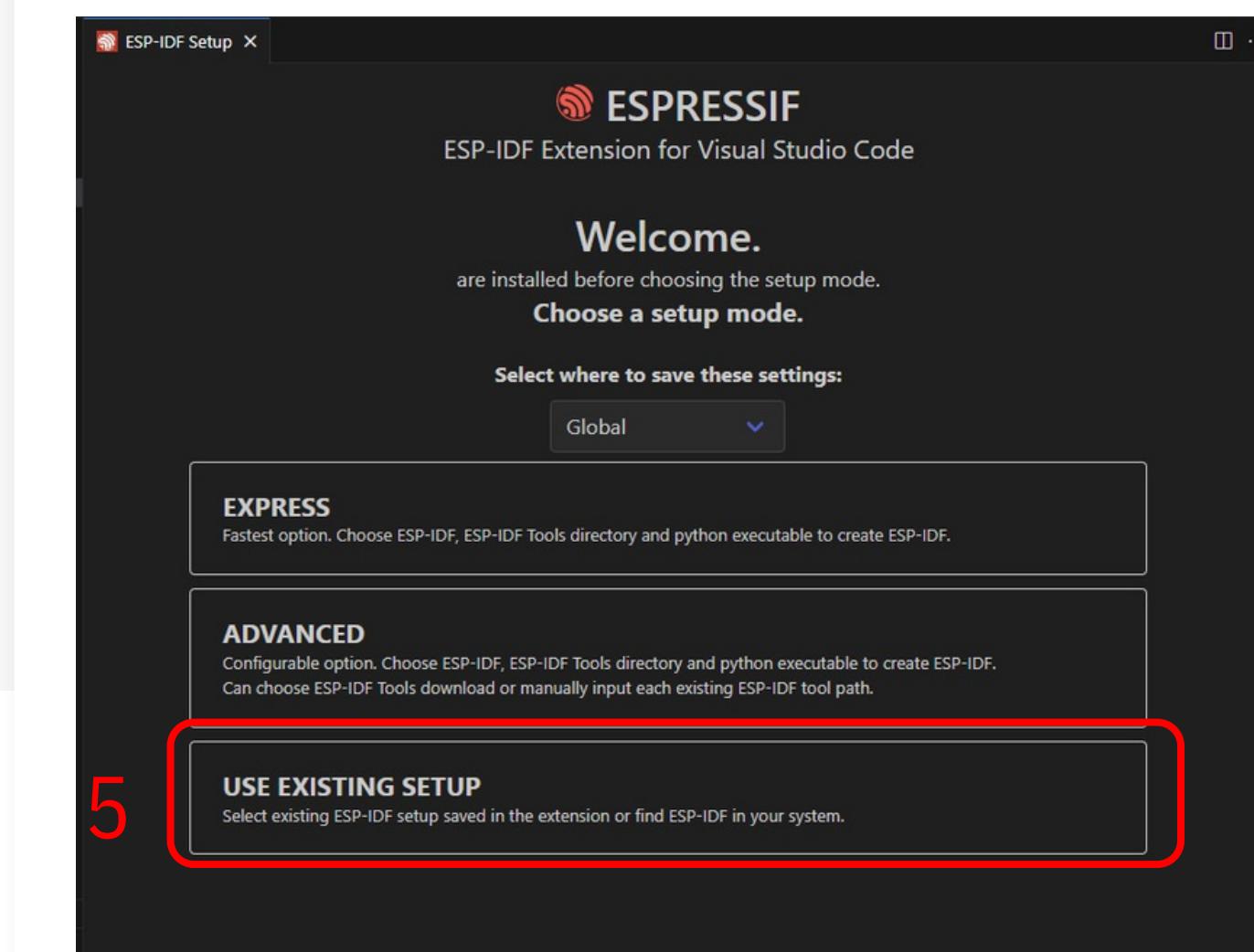
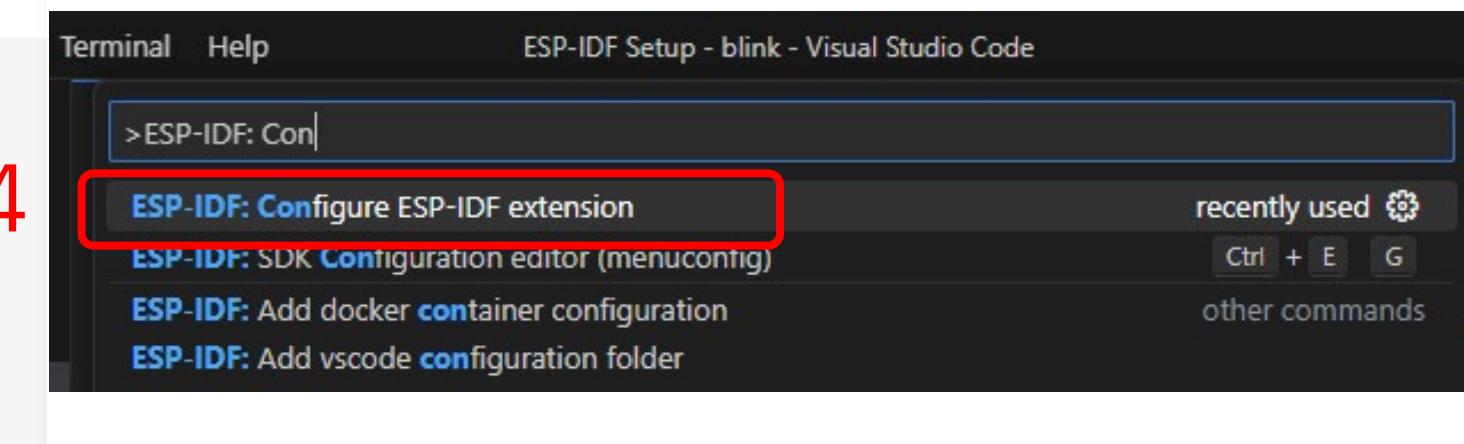


VScode: Blink (hands-on)

2. Download extension



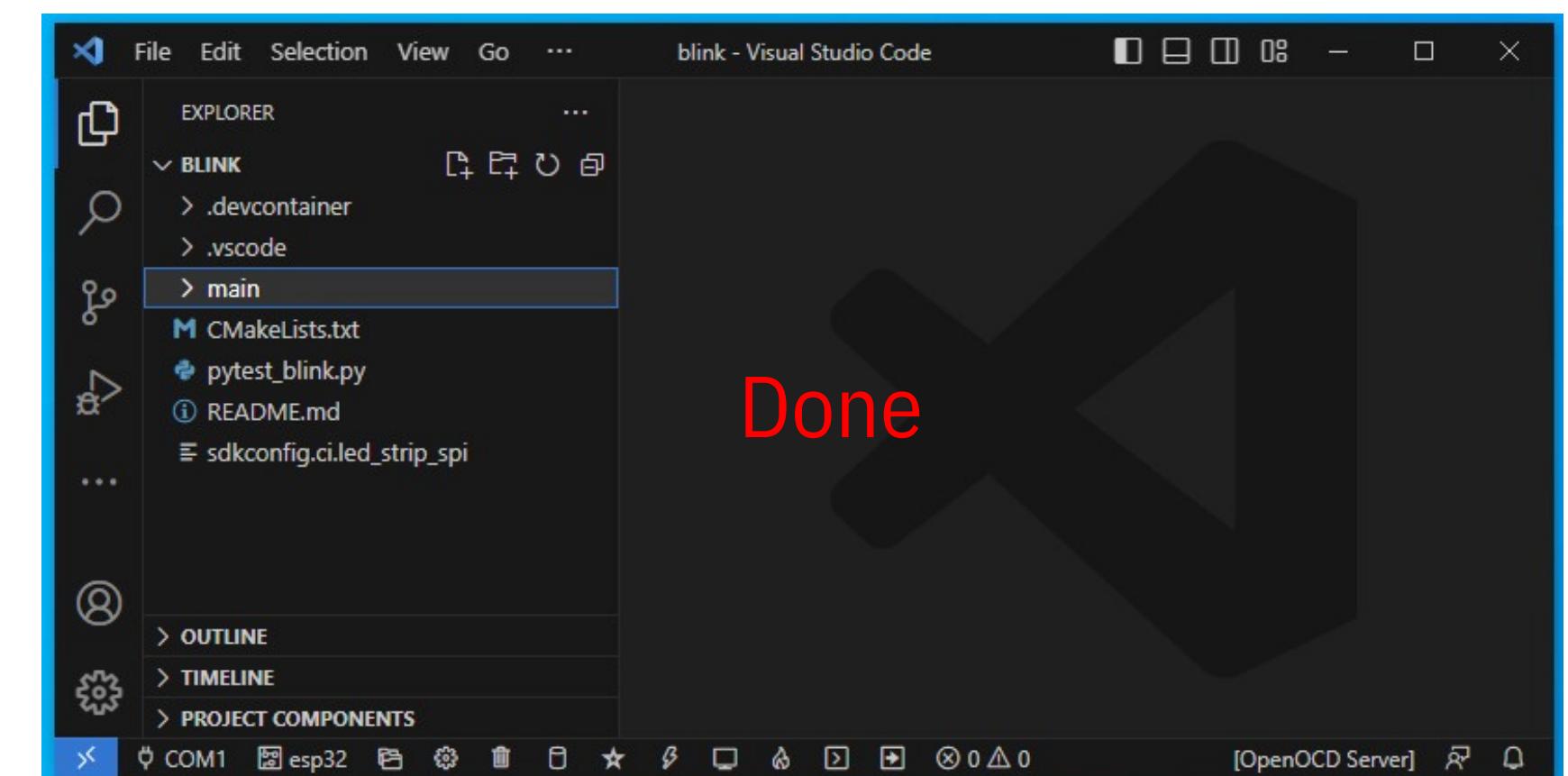
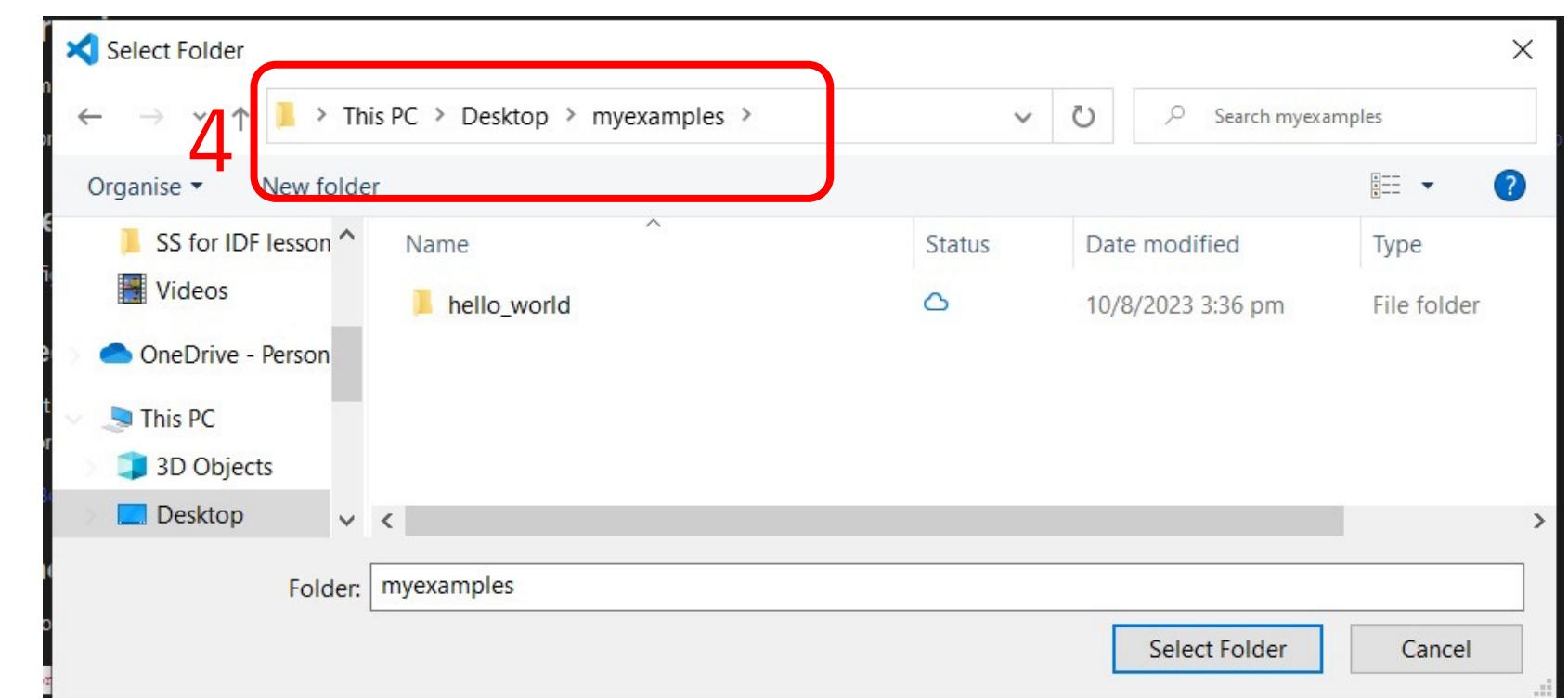
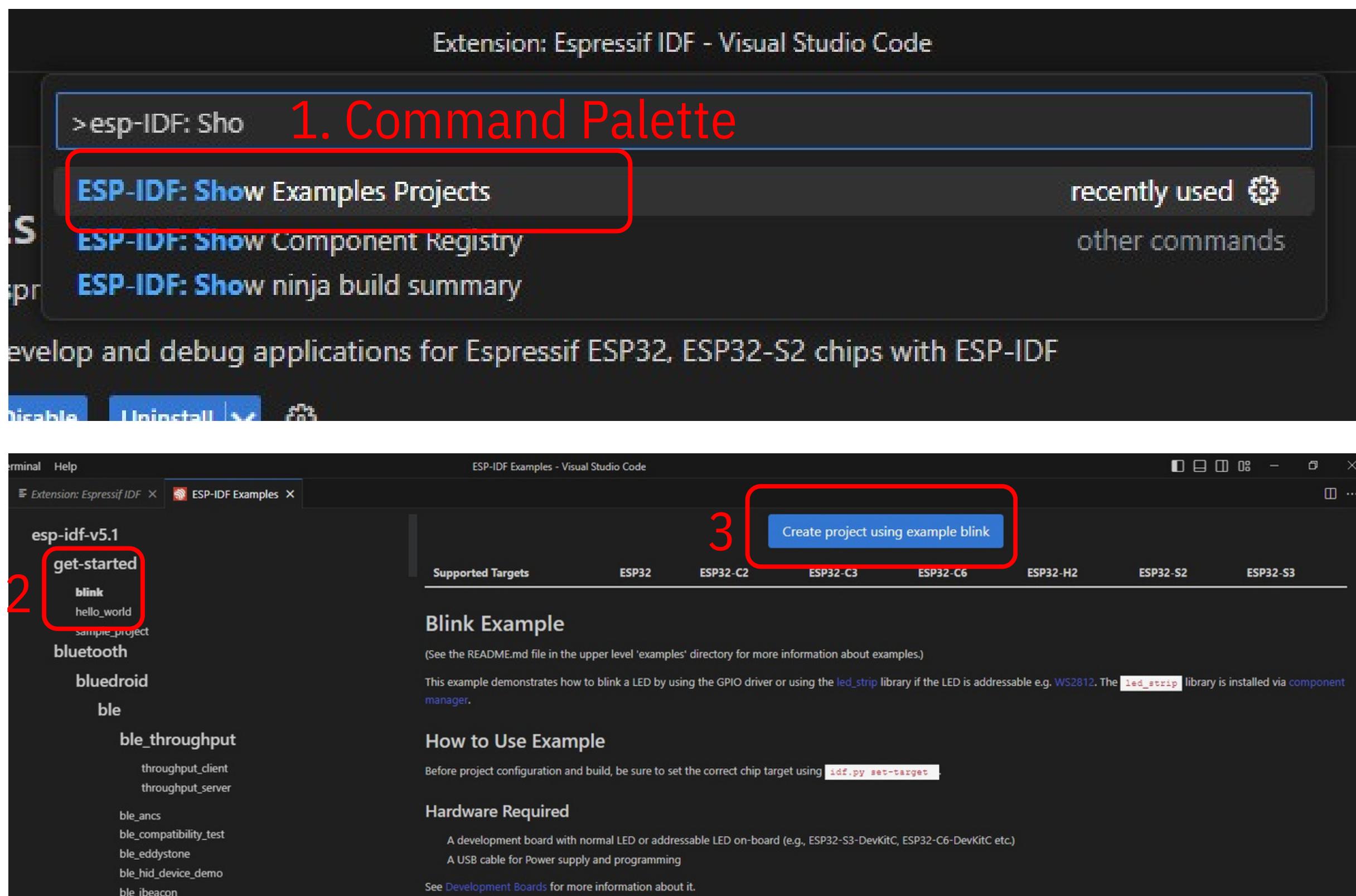
“Ctrl+Shift+P” or F1 to bring up Command Palette



Complete!

VScode: Blink (hands-on)

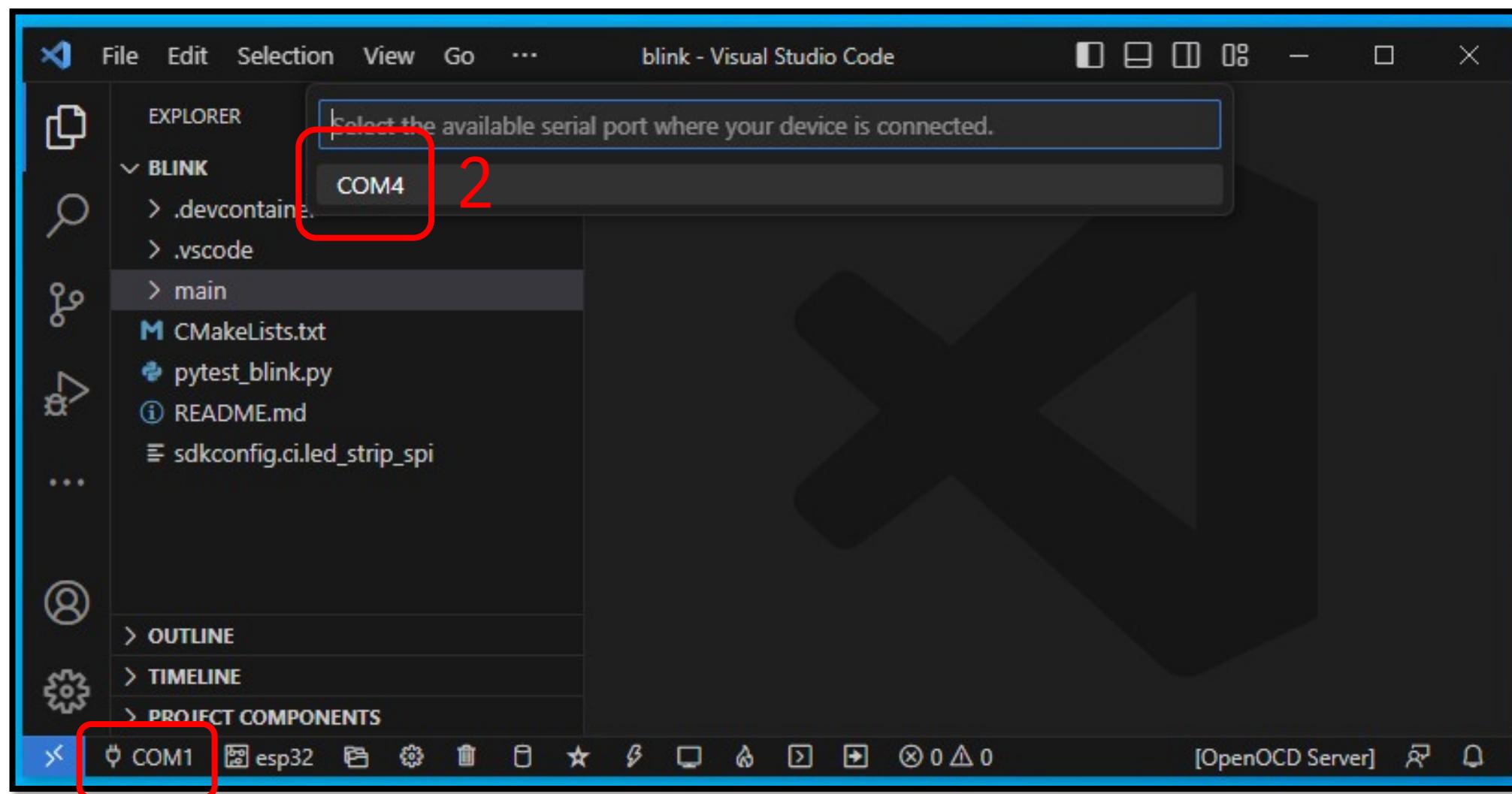
3. Copy 'blink' example



VScode: Blink (hands-on)

4. Set COM port

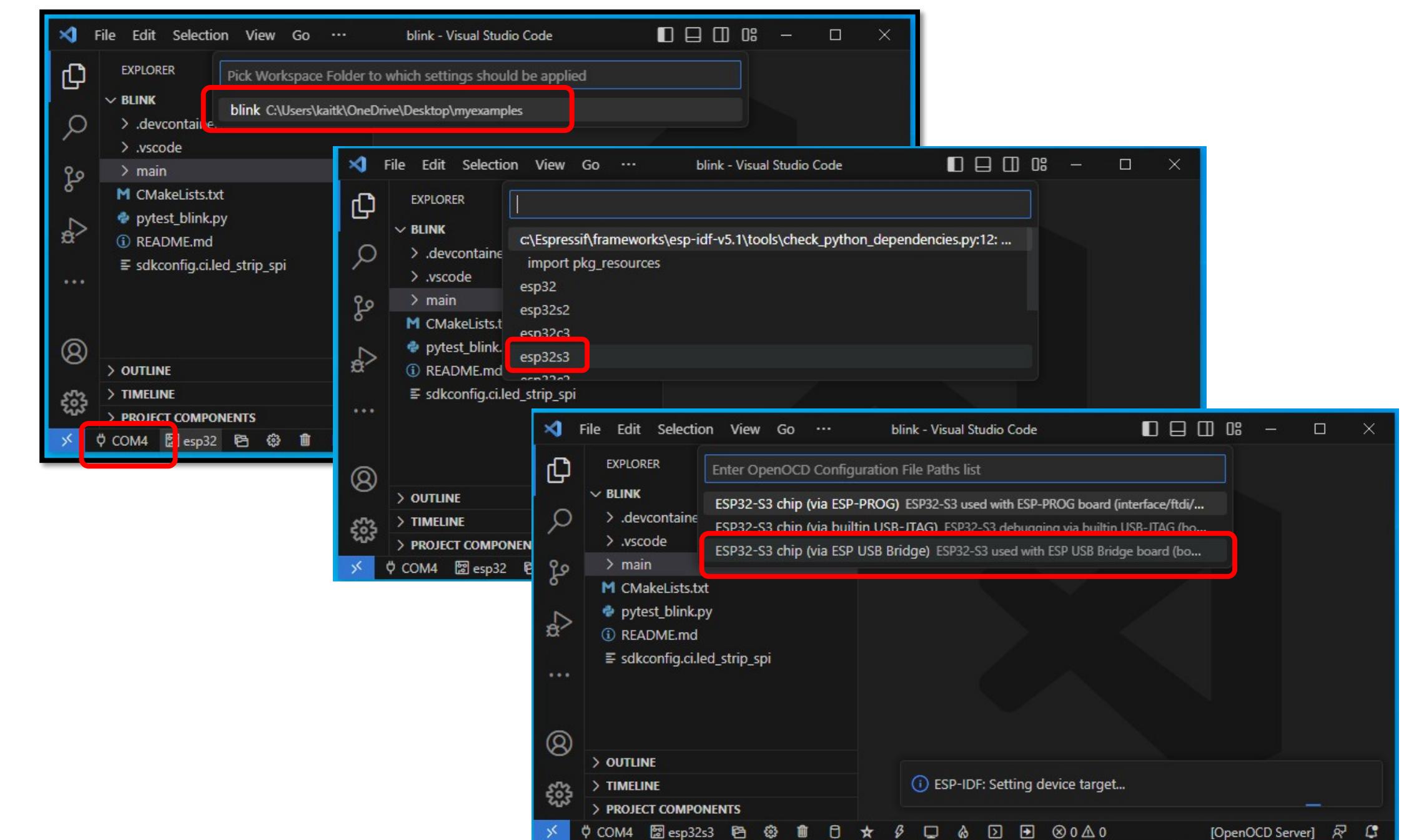
Set to your devkit's port



1

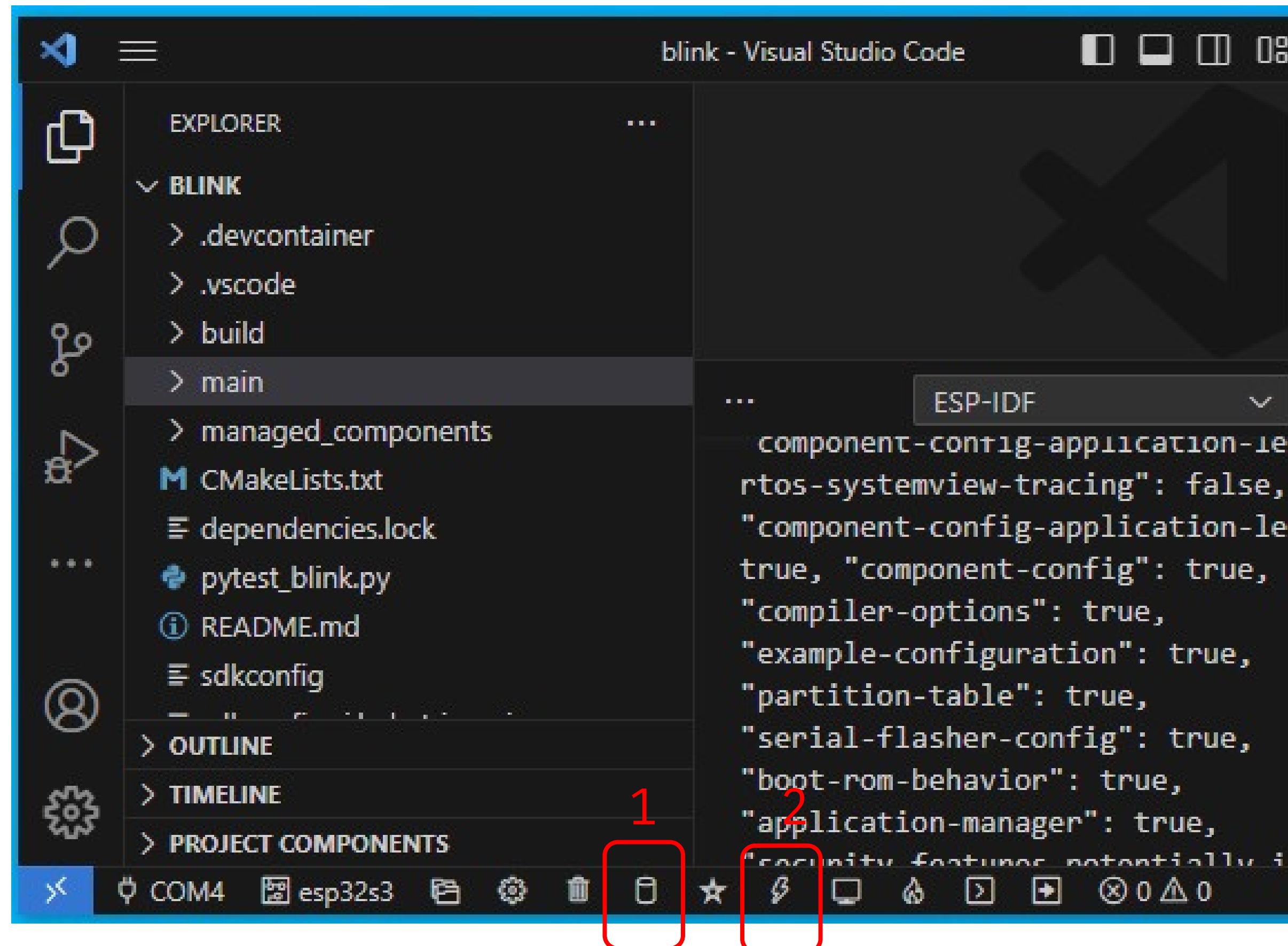
5. Set target chip

Set to esp32s3

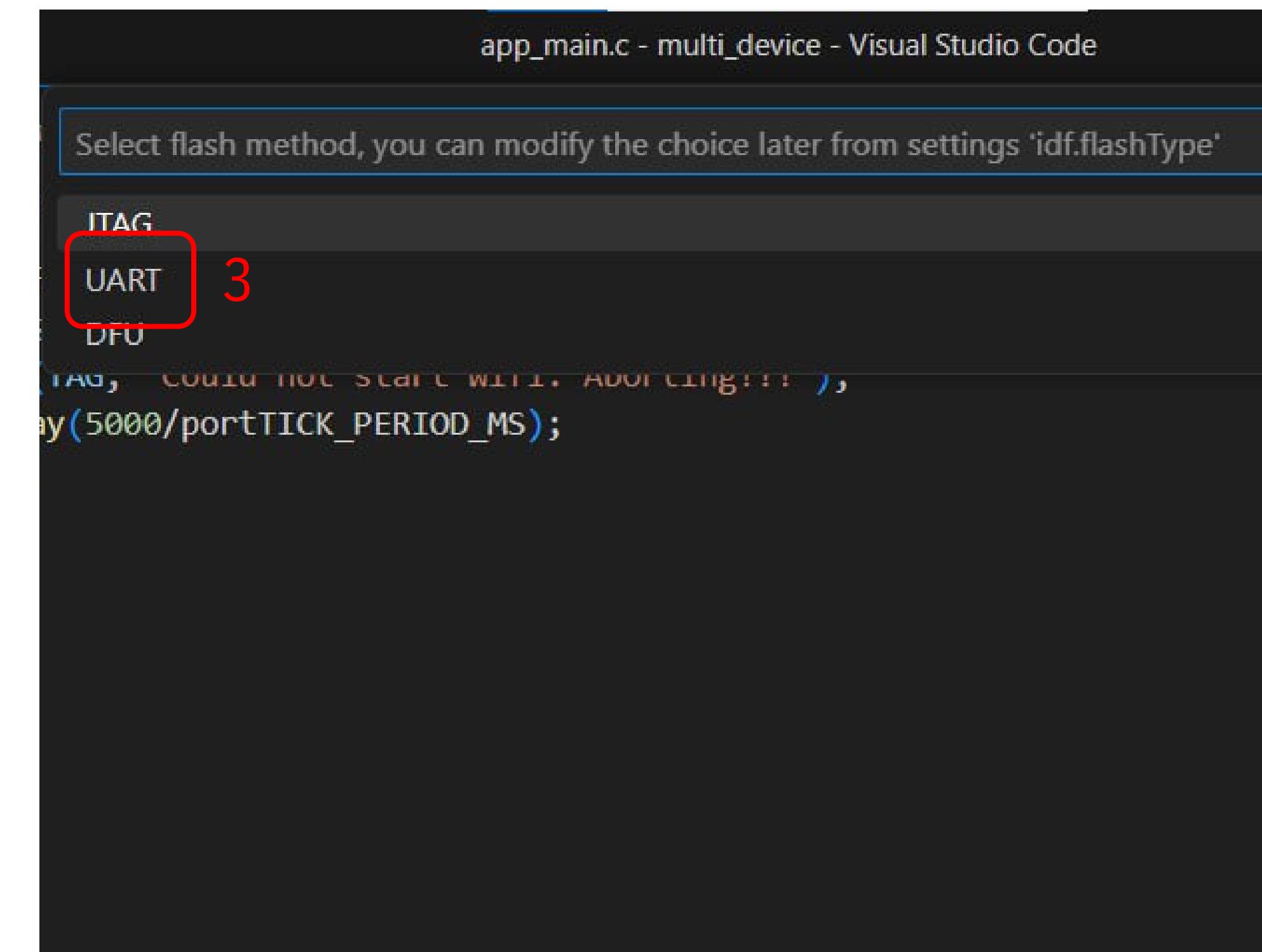


VScode: Blink (hands-on)

6. Build and flash



If ask for flash method, choose 'UART'



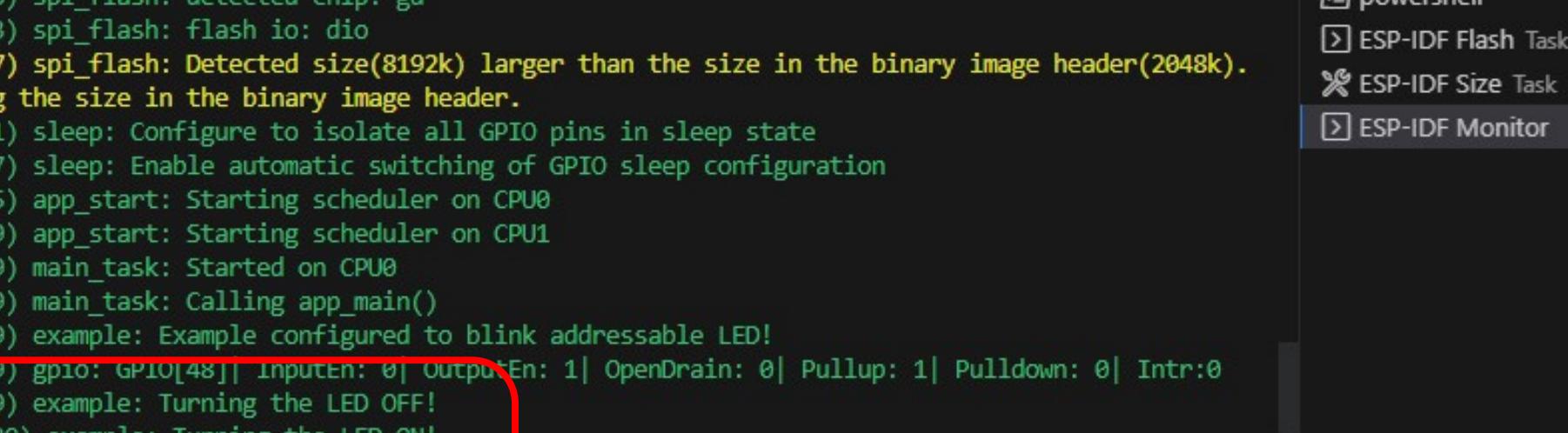
VScode: Blink (hands-on)

DONE!

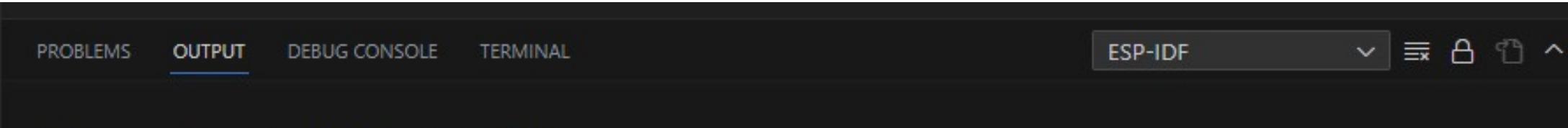
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL + - ... ^

.vectors size: 1027 bytes
Used stat D/IRAM: 12496 bytes ( 333360 remain, 3.6% used)
    .data size: 10344 bytes
    .bss size: 2152 bytes
Used Flash size : 160931 bytes
    .text: 113659 bytes
    .rodata: 47016 bytes
Total image size: 228997 bytes (.bin may be padded larger)

powershell
ESP-IDF Build Task
ESP-IDF Size Task
```



```
I (270) spi_flash: detected chip: gd
I (273) spi_flash: flash io: dio
W (277) spi_flash: Detected size(8192k) larger than the size in the binary image header(2048k).
Using the size in the binary image header.
I (291) sleep: Configure to isolate all GPIO pins in sleep state
I (297) sleep: Enable automatic switching of GPIO sleep configuration
I (305) app_start: Starting scheduler on CPU0
I (309) app_start: Starting scheduler on CPU1
I (309) main_task: Started on CPU0
I (319) main_task: Calling app_main()
I (319) example: Example configured to blink addressable LED!
I (329) gpio: GPIO[48]:| inputEn: 0| outputEn: 1| OpenDrain: 0| Pullup: 1| Pulldown: 0| Intr:0
I (339) example: Turning the LED OFF!
I (1339) example: Turning the LED ON!
I (2339) example: Turning the LED OFF!
I (3339) example: Turning the LED ON!
```



PROBLEMS **OUTPUT** DEBUG CONSOLE TERMINAL ESP-IDF

```
Python requirements has been installed
c:\Espressif\tools\tools\cmake\3.24.0\bin\cmake.exe

c:\Espressif\tools\tools\ninja\1.10.2\ninja.exe
```

[Flash]
Flash Done ⚡



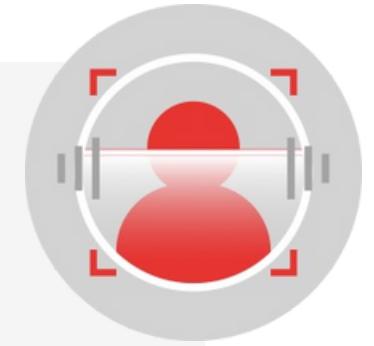
CHALLENGE YOURSELF!

Change the speed and colour of
the RGB LED

Hands-on Time!



- Download and install VScode & Espressif IDF
- Command Palette > Configure ESP-IDF extension
- Command Palette > Show Examples Project
- Copy ‘Blink’ into ‘myexamples’ folder
- Setup, build, flash



06

ESP RainMaker

Introduction, Features, Applications

ESP RainMaker

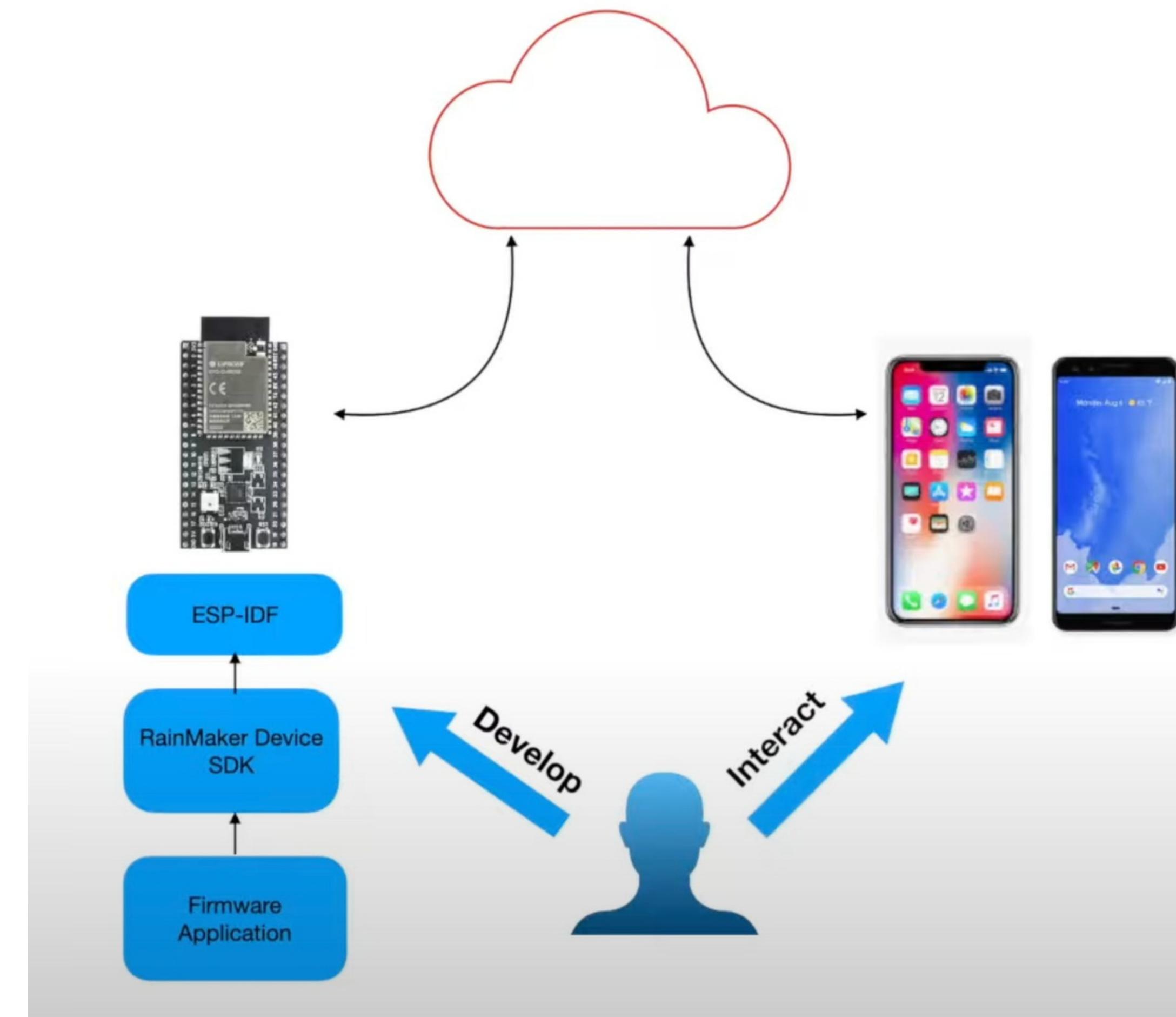
ESP RainMaker is a complete, yet light-weight, AIoTsolution that enables private Cloud deployment for your business in a simple, cost-effective and efficient manner.



ESP RainMaker

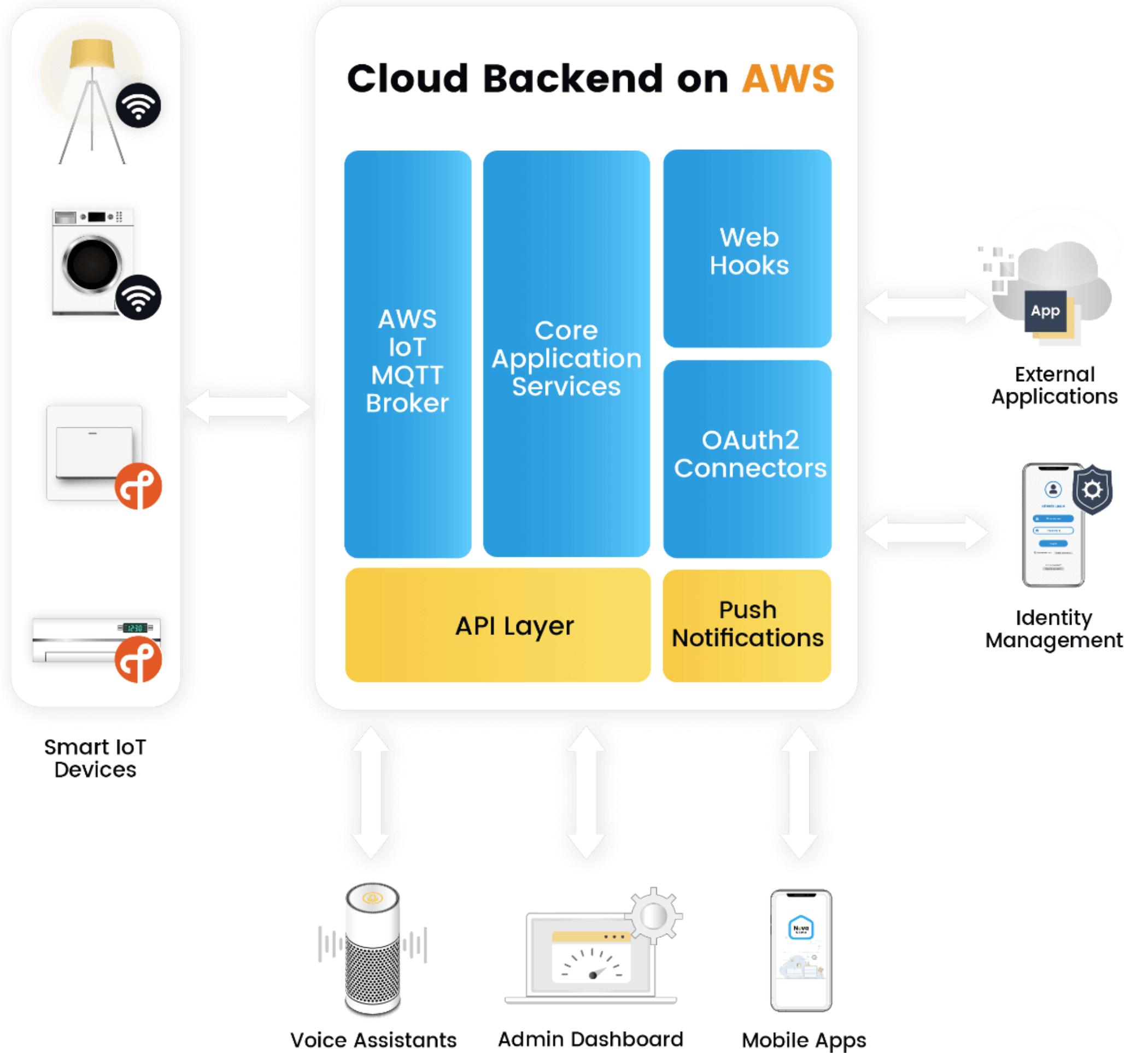
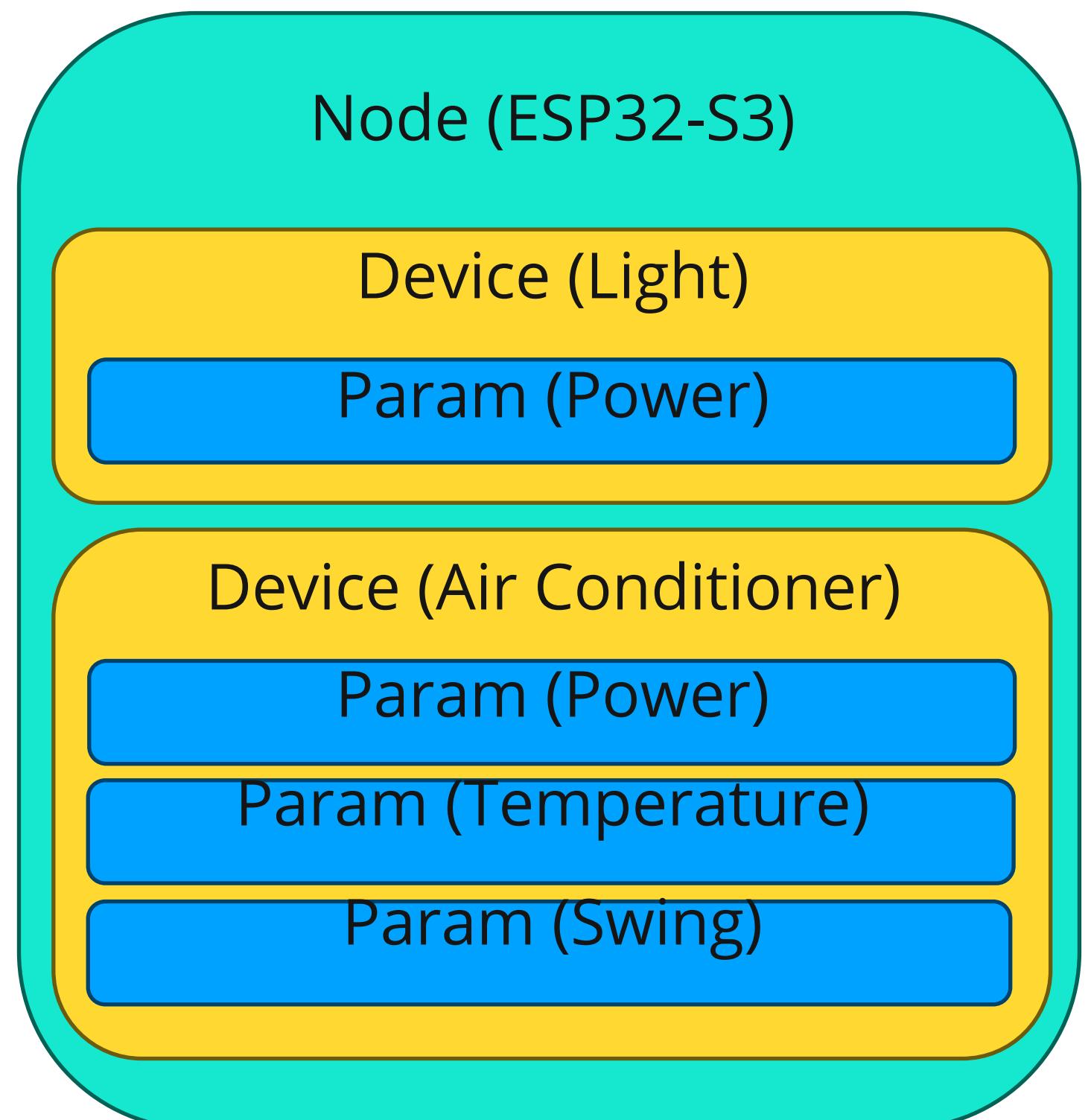
Features a self-adapting phone app and a transparent cloud middleware.

Makers can focus on developing their applications on top of RainMaker Device SDK, interact with the phone app, and forget about the rest of the infrastructure.



ESP RainMaker

Node, Device, Parameter



ESP RainMaker

Examples



Lightbulb



Plug



Lightstrip



Socket



Panel Light



Dimmer



Threadmill



Coffee Machine



Garage Door Controller



Thermostat



Remote Control Hub

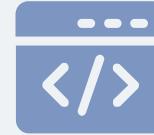


Faster development

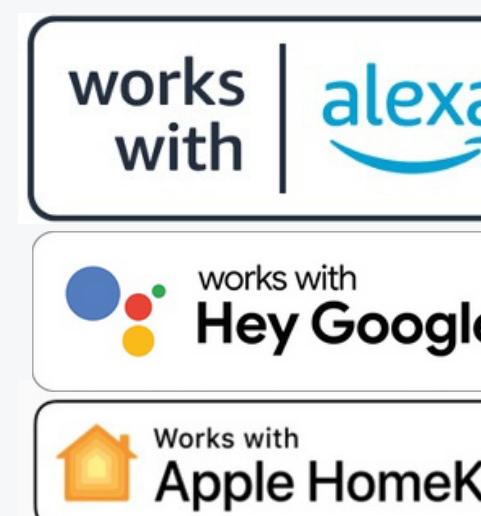
Faster time to market without sacrificing customization.

Drastically simplify code complexity using smart-home products SDKs.

Works with online Voice Assistant, as well as offline Voice Assistant ESP-Skainet.



Voice Assistant Integration



ESP RAINMAKER®

RainMaker App

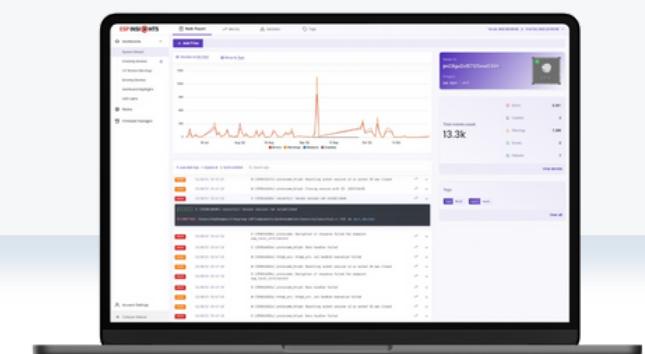
ESP RainMaker apps for iOS & Android are open-source, offering a reference design and development guidance.



Remote observability and diagnostics platform for in field-deployed devices.

Receive information about crashes, unintended reboots, error and warning logs, etc.

ESP Insights



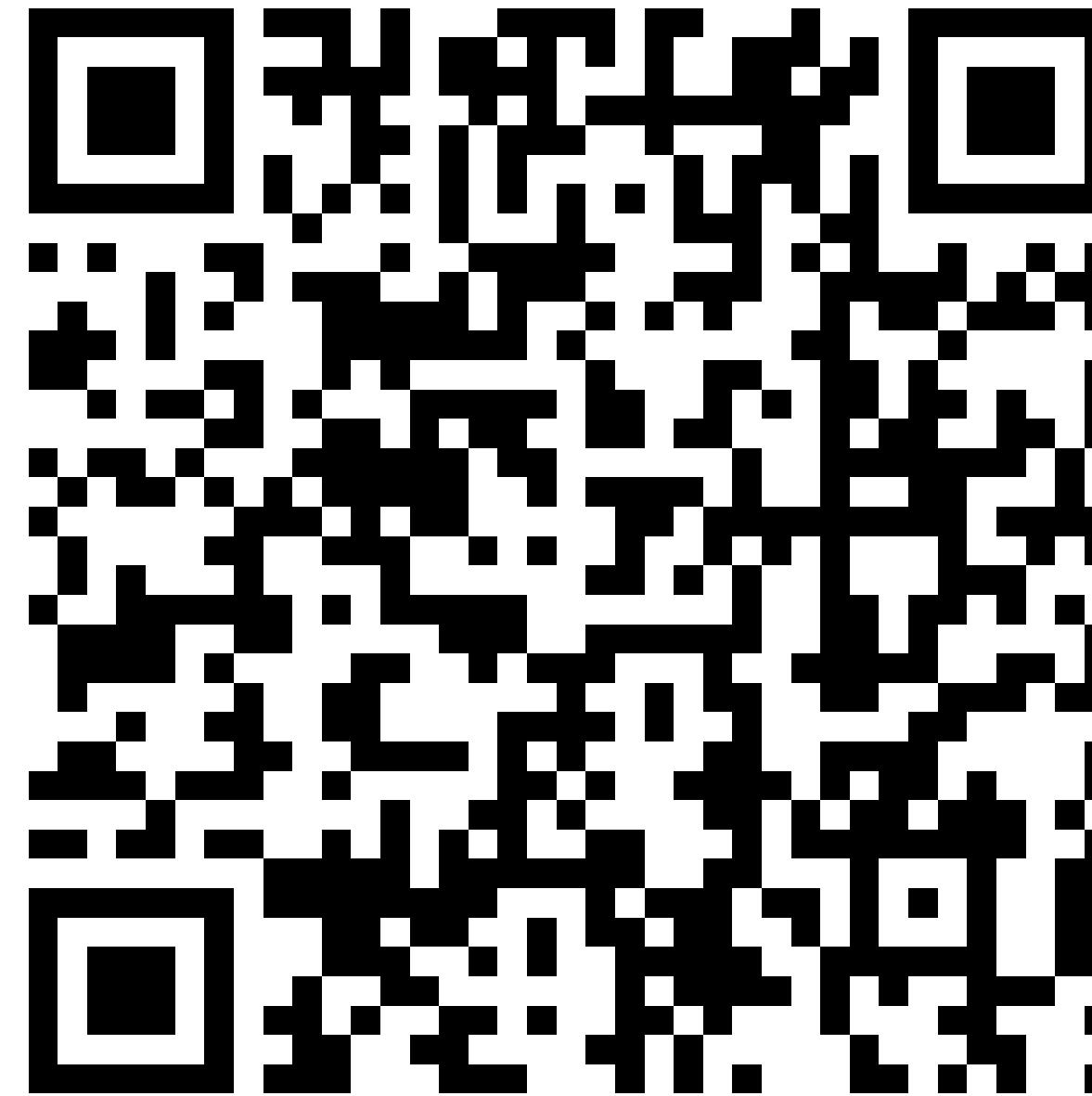
07

VScode: Multi Device (hands-on)

Download RainMaker App, download RainMaker framework, flash ‘Multi Device’ example

VScode: Multi Device (hands-on)

1. Download 'ESP RainMaker on smartphone

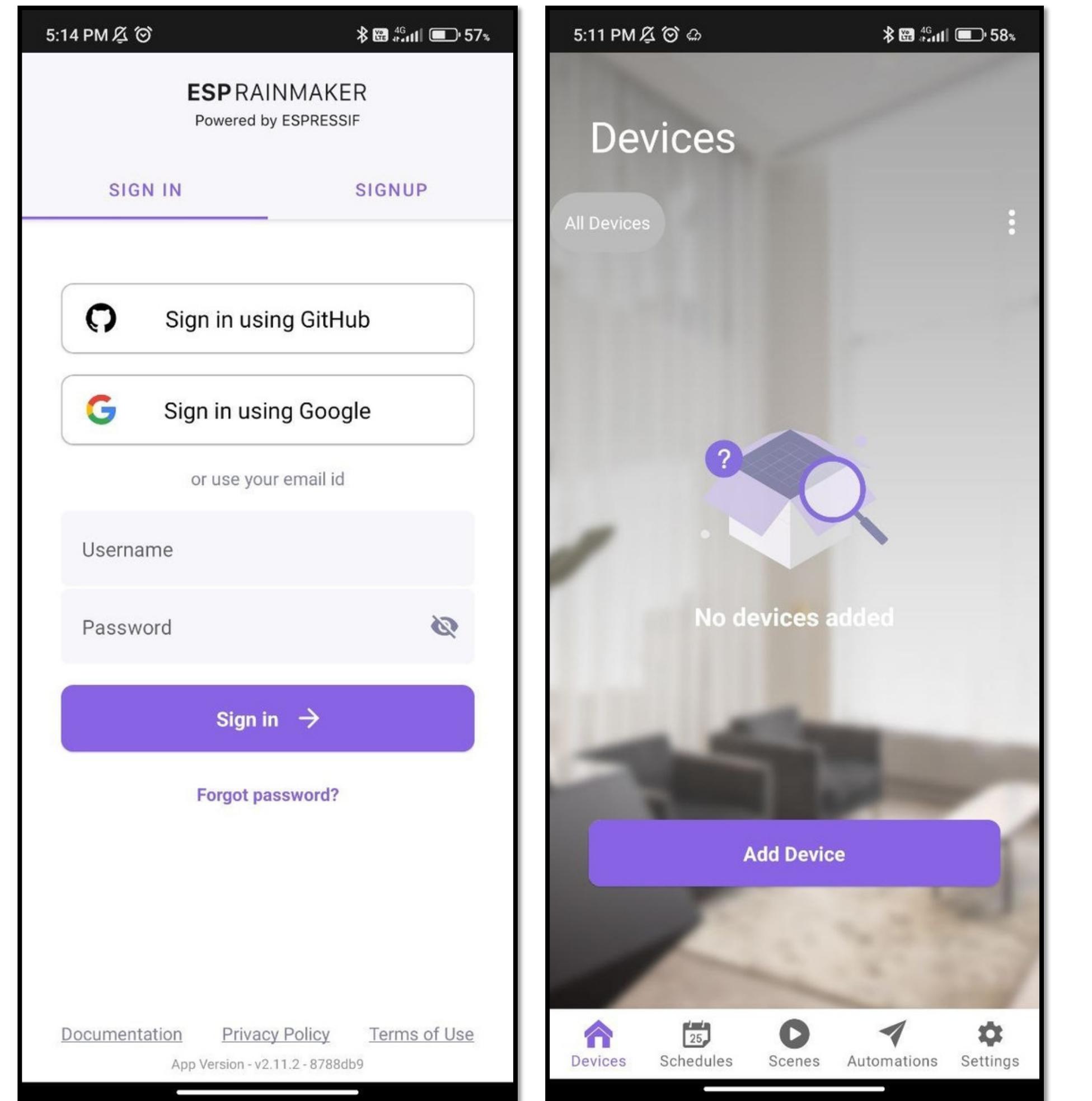


<https://play.google.com/store/apps/details?id=com.espressif.rainmaker>

<https://apps.apple.com/app/esp-rainmaker/id1497491540>

VScode: Multi Device (hands-on)

2. Make account and login



VScode: Multi Device (hands-on)

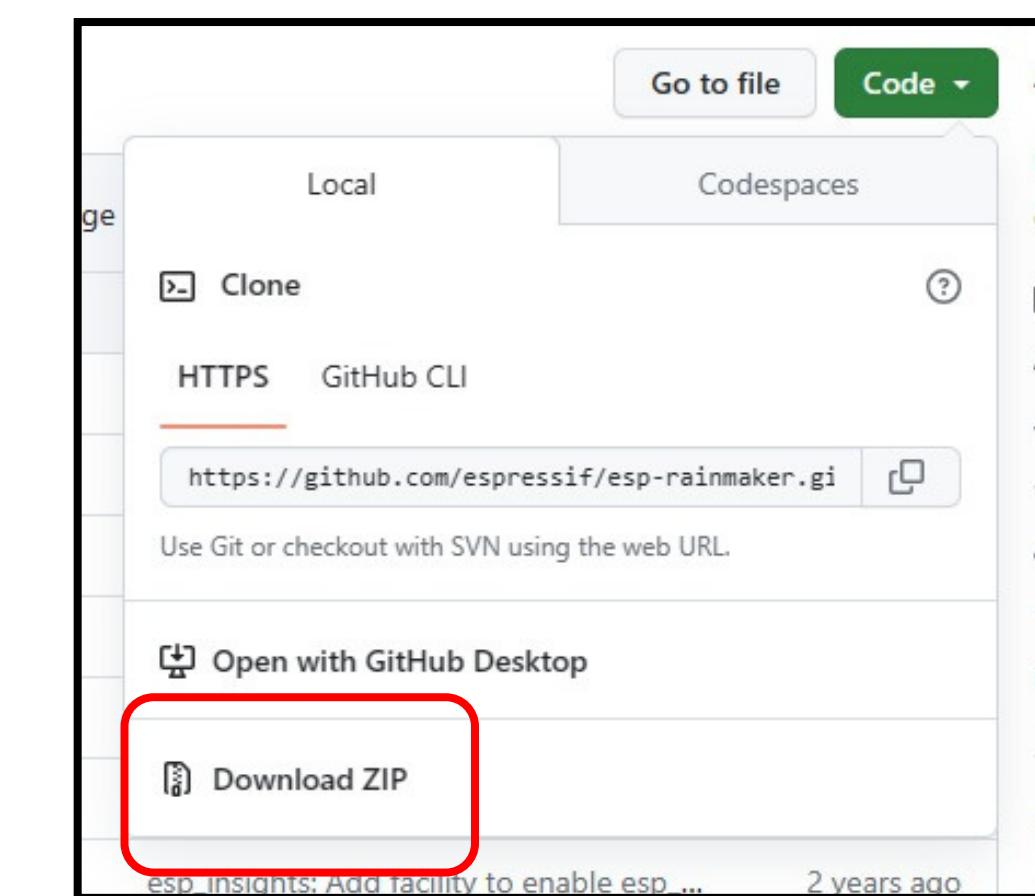
3a. Download RainMaker (2 options)

Via ESP-IDF PowerShell

1. Navigate to C:/../espressif/frameworks
2. Run git clone --recursive https://github.com/espressif/esp-rainmaker.git

Via website zip download

1. Go to <https://github.com/espressif/esp-rainmaker.git>
2. Download as zip
3. Extract in C:/../espressif/frameworks

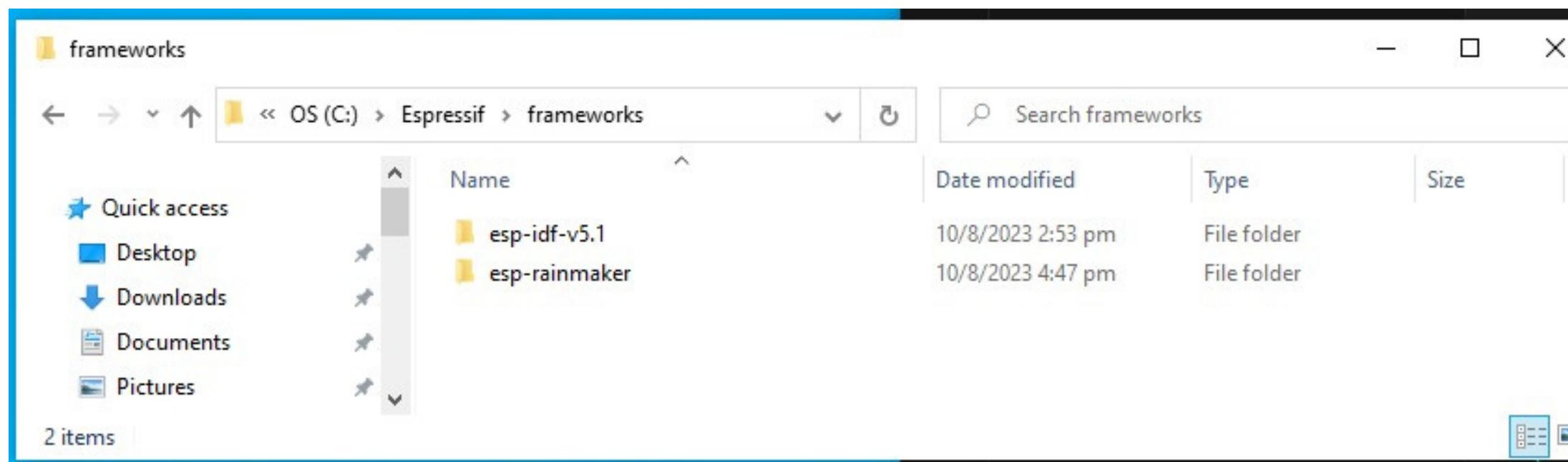


VScode: Multi Device

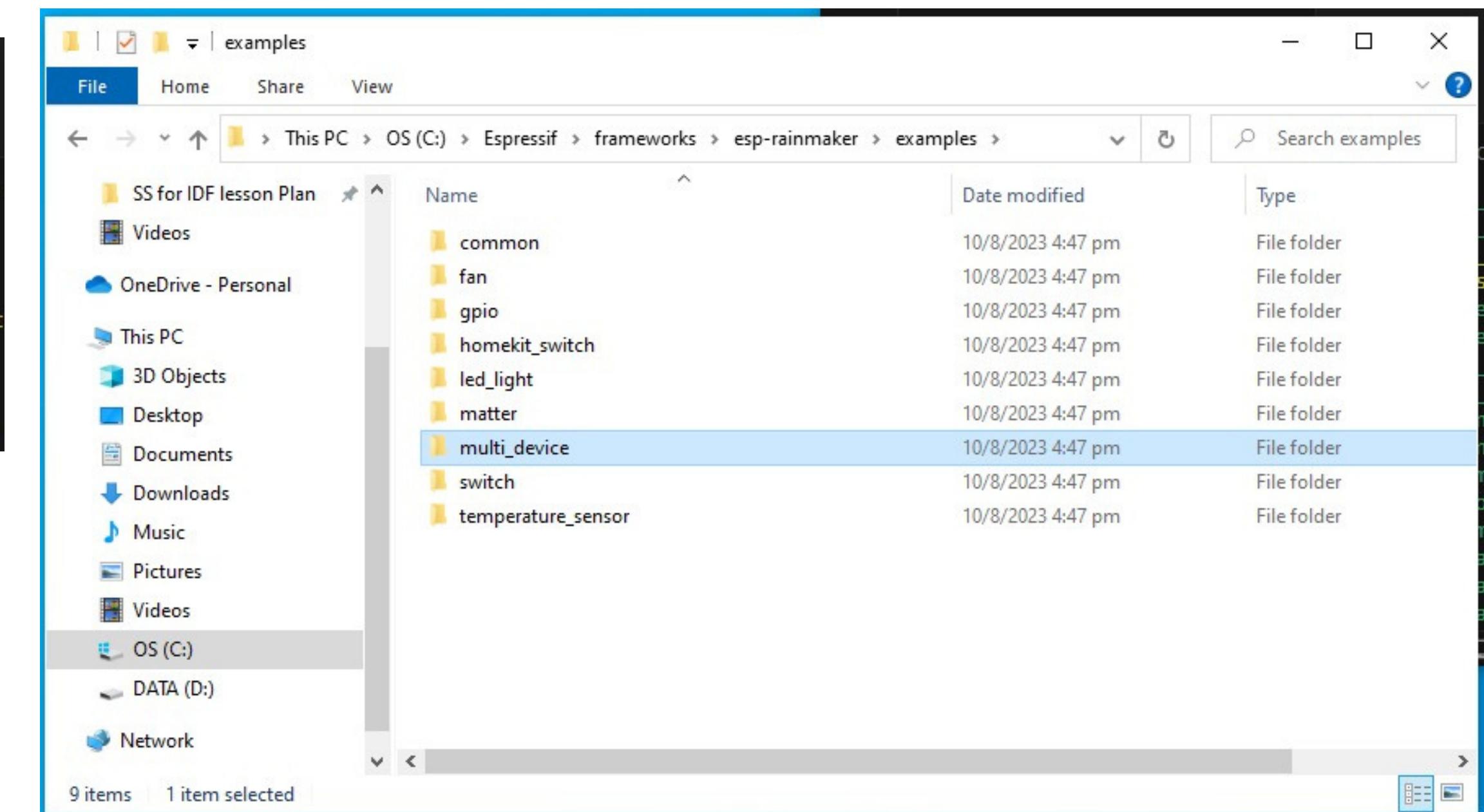
(hands-on)

3b. Check for RainMaker

Verify that you have ‘esp-rainmaker’ in the ‘frameworks’ folder.



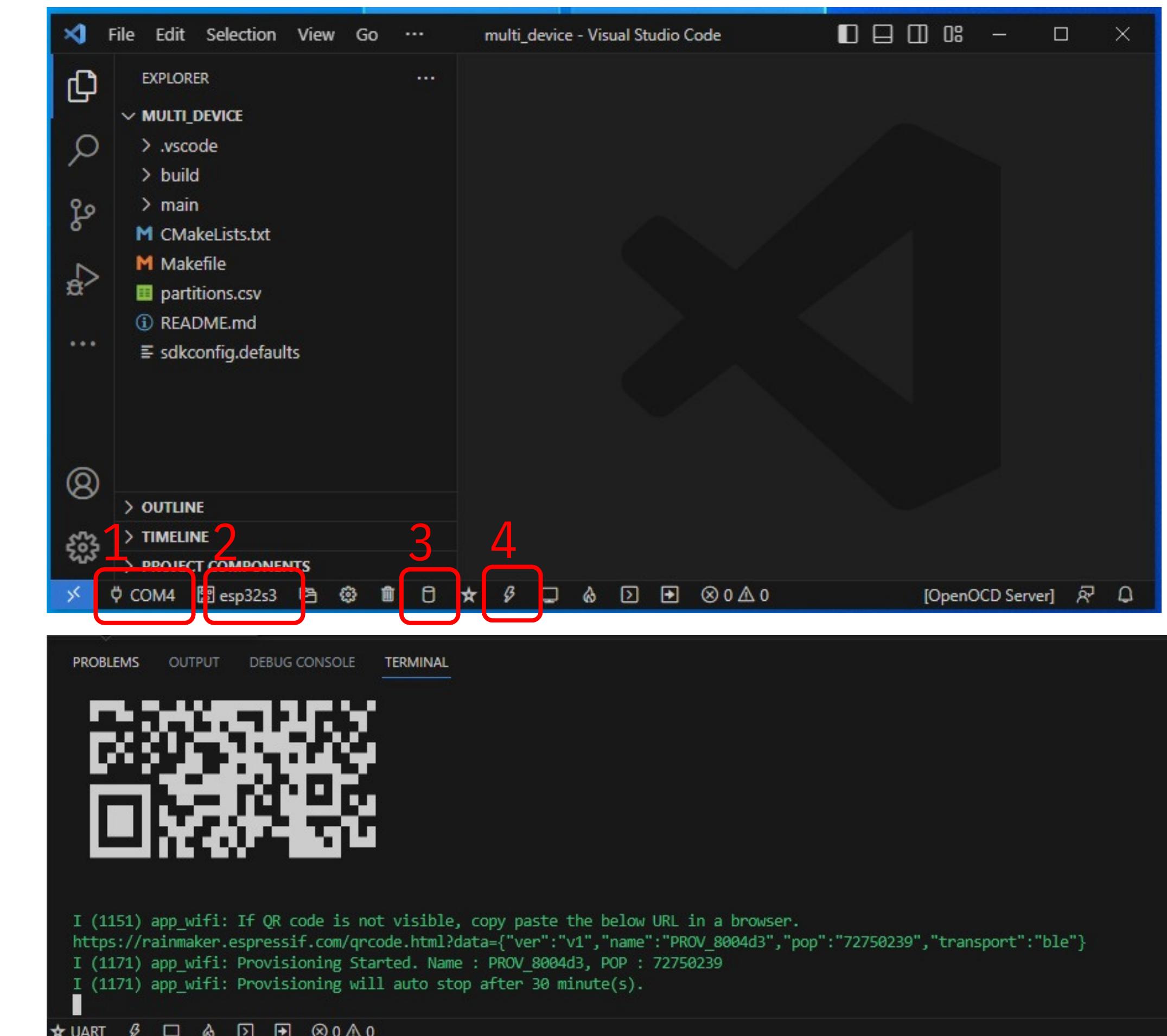
Check for ‘multi-device’ in
esp-rainmaker > examples



VScode: Multi Device (hands-on)

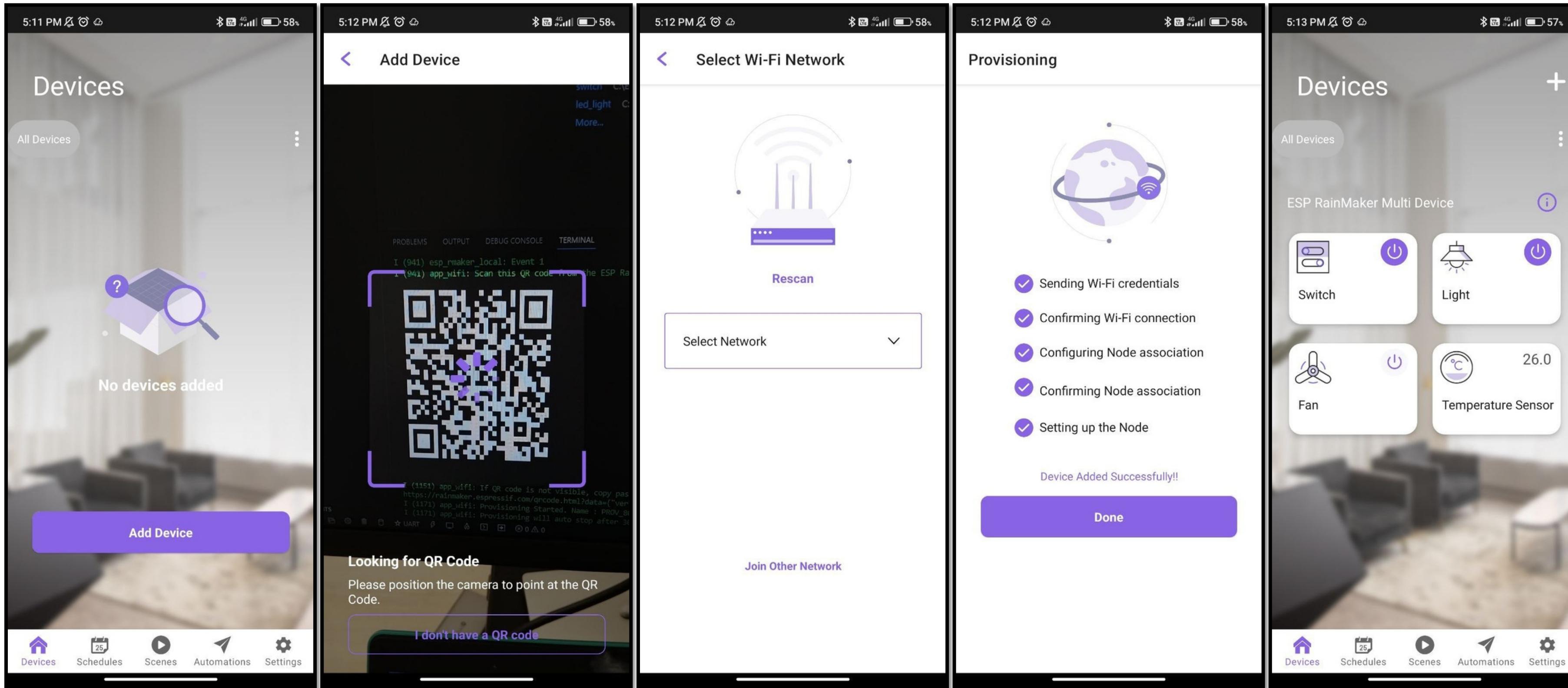
4. Flash ‘multi device’ into devkit

1. Open ‘multi device’ folder in VScode
2. Setup COM port and chip
3. Build project
4. Flash
5. You should see a QR code in terminal



VScode: Multi Device (hands-on)

5. Add device on phone's RainMaker App



Done

Hands-on Time!



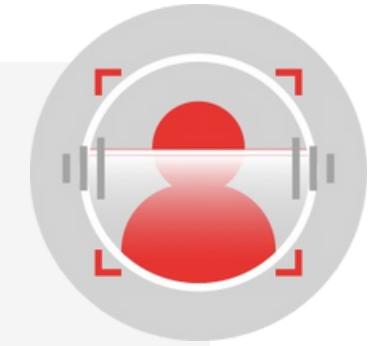
Download RainMaker App and log in

Download RainMaker framework

Flash ‘multi-device’ example into dev kit

On App, add device via QR code

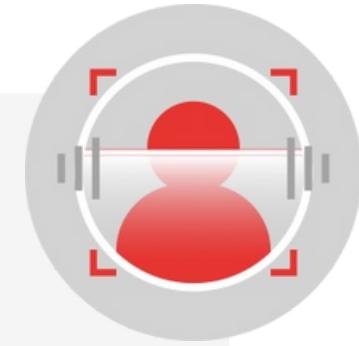
Control RGB LED



Smart Night Light Challenge!



1. ON/OFF control for an external LED via RainMaker App
2. See LDR sensor reading on RainMaker App
3. Using RainMaker App, set automation to ON external LED when it is dark

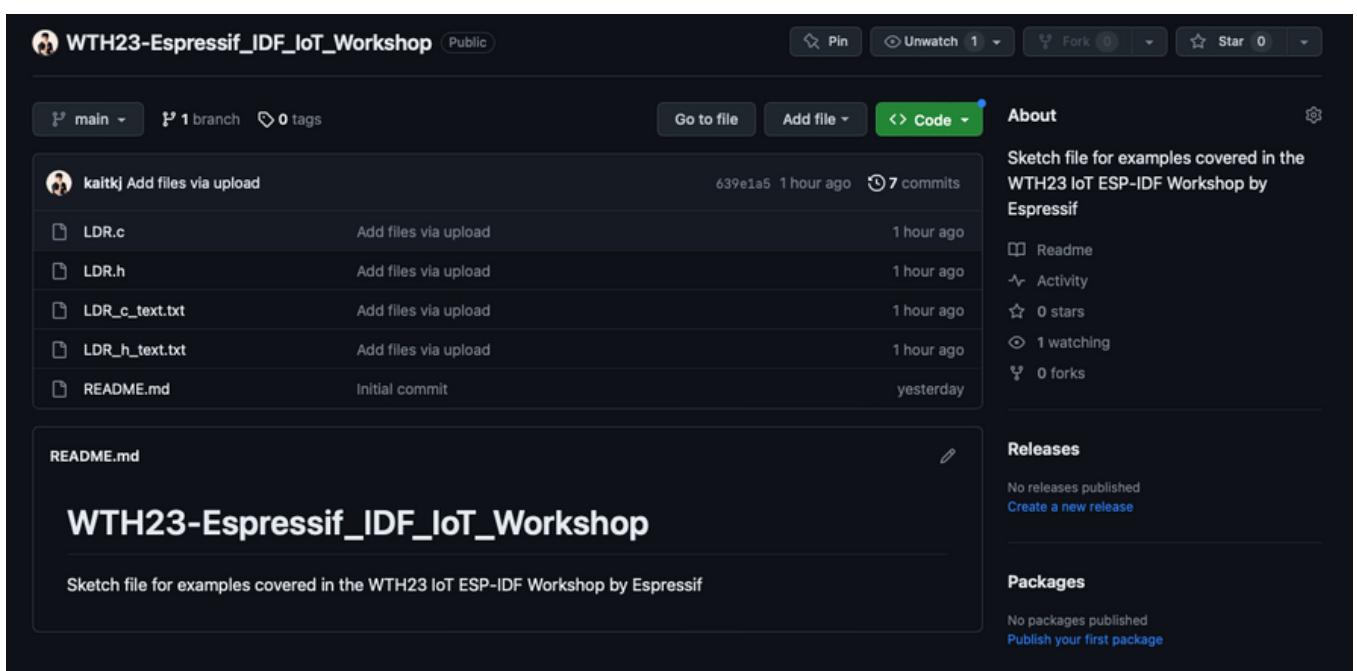


Smart Night Light Challenge Hints

Lost? Try start from a sample code

Read the code, and work on the taskings in the comments

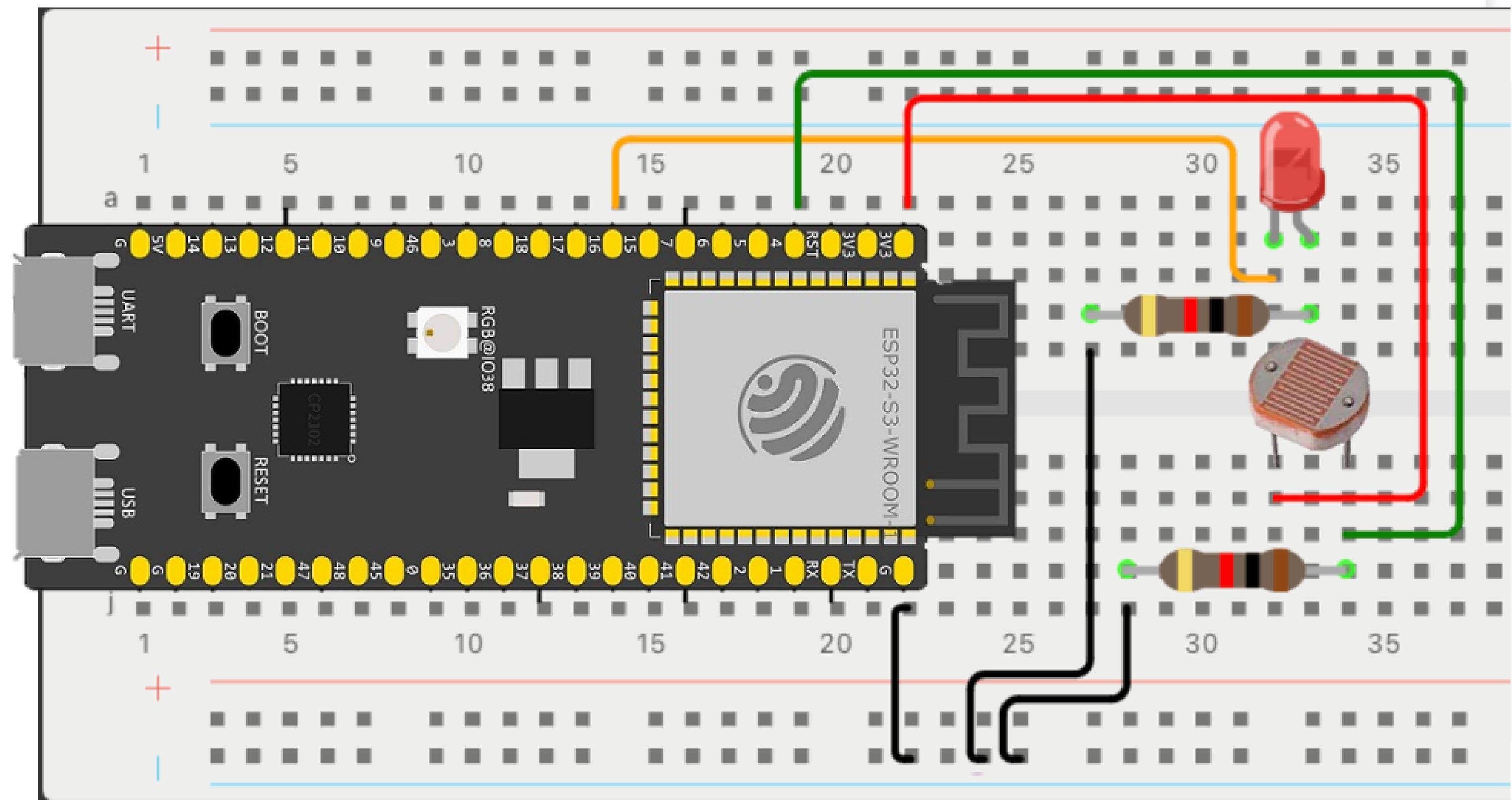
https://github.com/kaitkj/WTH23-Espressif_IDF_IoT_Workshop



Smart Night Light Challenge Hints

Electronics Refresher

1. Identify the components you need to fulfill each function
2. How to use the components
3. Which GPIO pins to use



08

Product Demonstration

Demonstration of ESP32-S3-EYE, ESP32-S3-EV-Board-2 ESP32-S3-BOX-Lite

Product Demonstration

ESP32-S3-EYE

Camera, 1.3' LCD,
MicroSD Card slot,
microphone



ESP32-S3-
EV- BOARD-
4.3' Touchscreen,
microphone, Audio
codec, GPIO



ESP32-S3-

BOX- Lite

2.3' LCD, microphone,
speaker, buttons,
GPIO



Further Learning

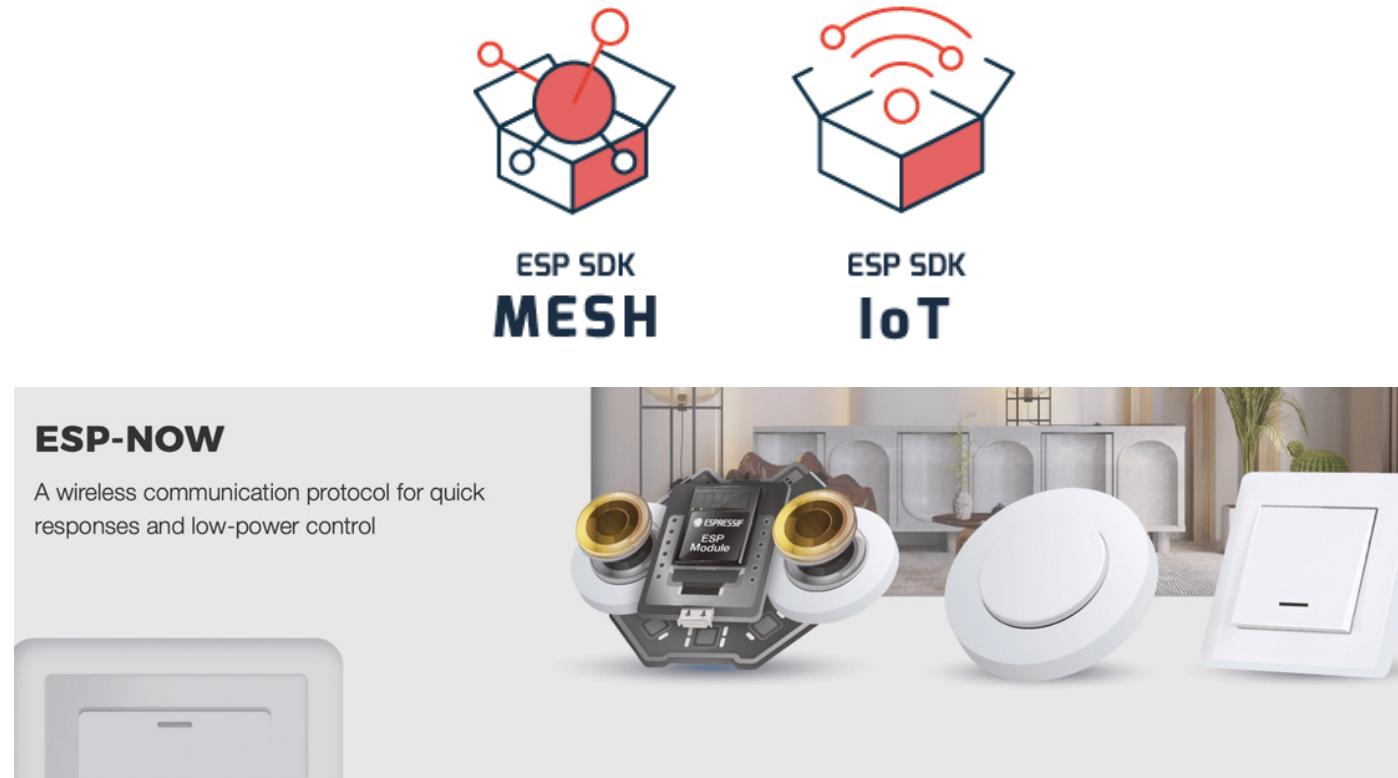


ESP-IDF



**ESP
Arduino**

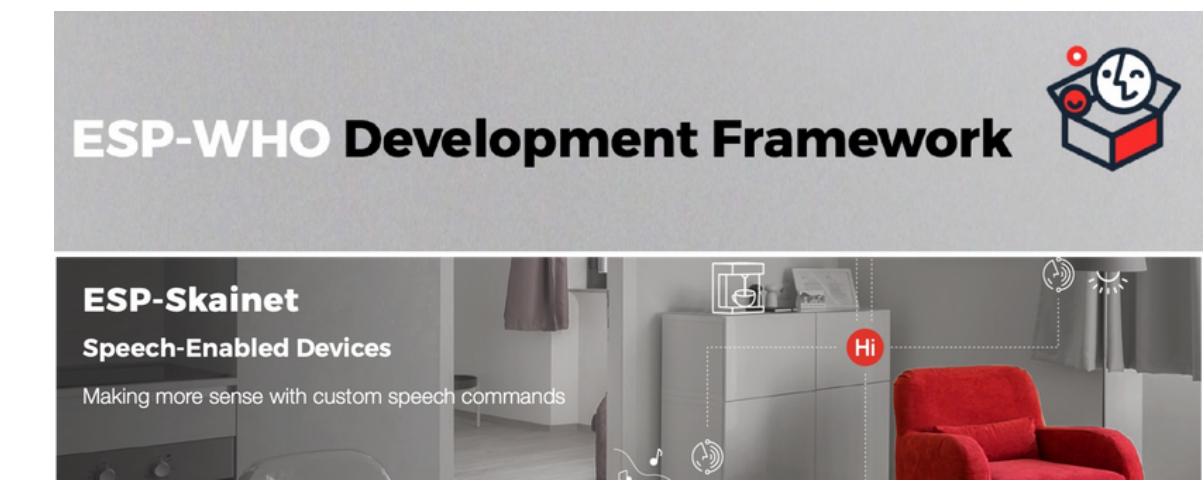
IDF/IDE



Wireless Comm



AI/DSP



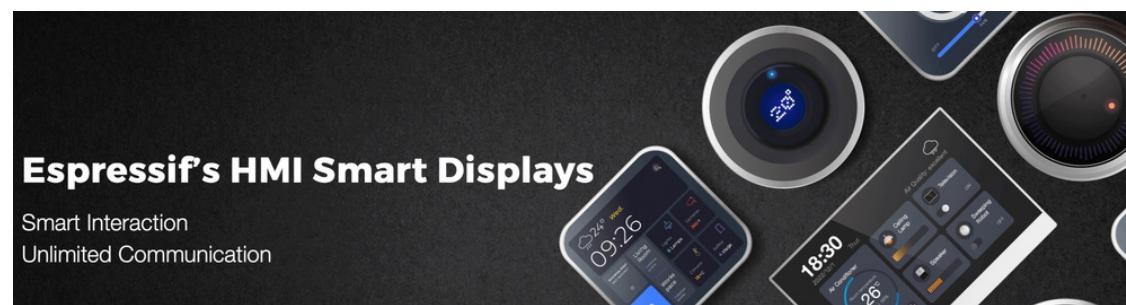
AI Features



ESP RAINMAKER®



Analytics and
OTA Updates



Visual and Audio



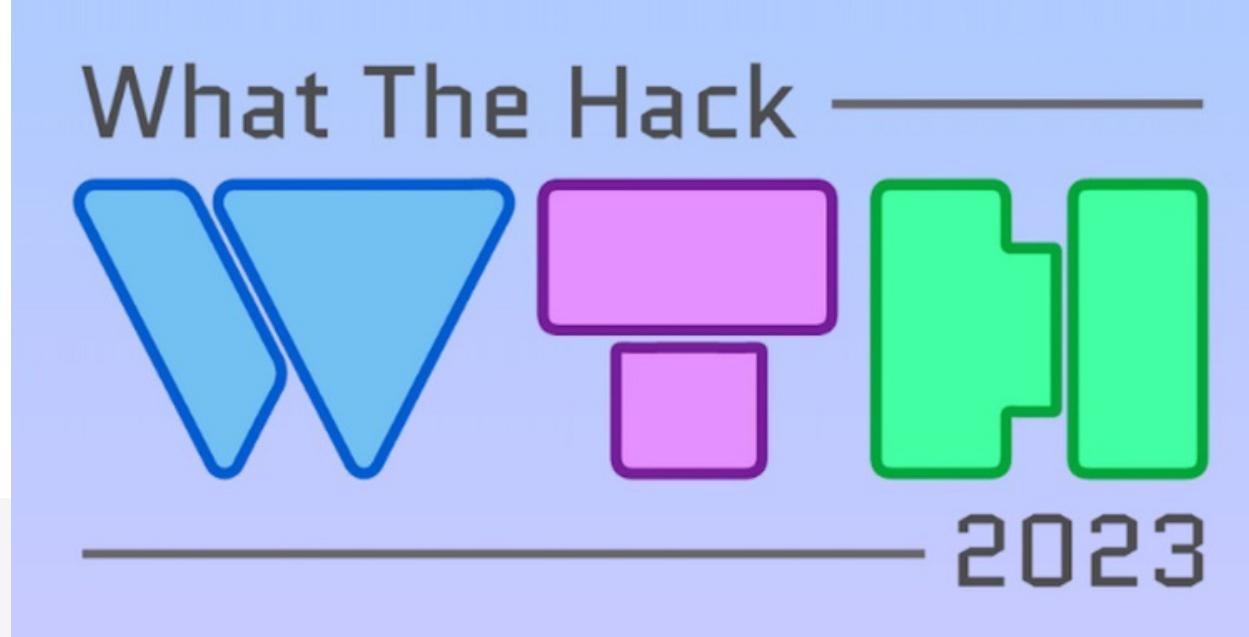
**ESP
SOLUTION
AUDIO**



**ESP
Jumpstart**

Learn ESP-IDF with
ESP Jumpstart

Fireside Chat and Human Library



Learn more about
AIoT from the
industry experts on
26th Aug 2023
12pm - 1.30pm



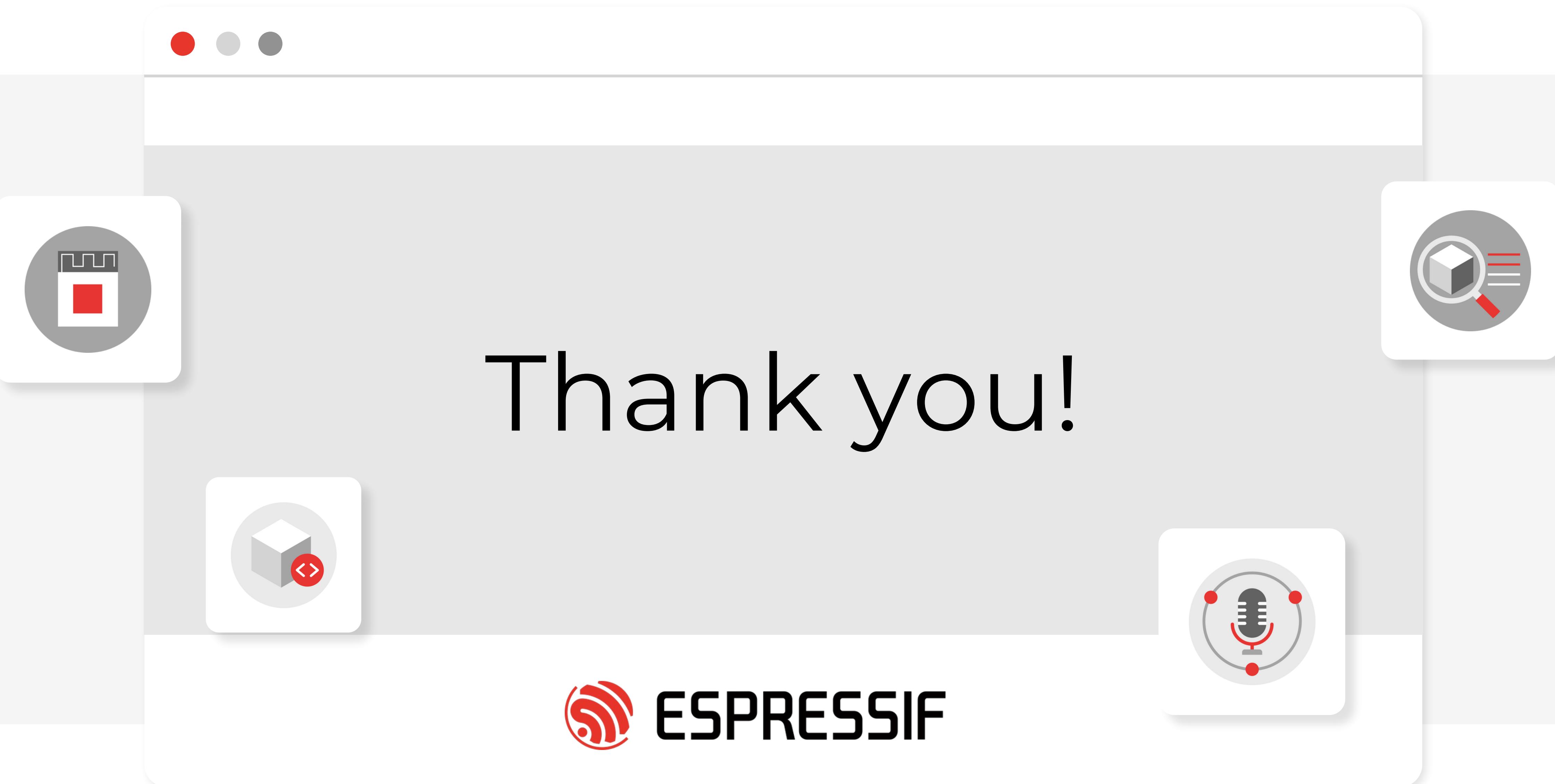
Amey Inamdar
Director of
Technical Marketing





Jeroen Domburg
Senior Software
&
Technical Marketing
Manager





Thank you!



Post Workshop Survey

Espressif x SUTD WTH23 - ESP-IDF
Workshop



<https://forms.office.com/r/5z9JmHNfZy>