

CONTENTS

01 Getting started with ESP32

02 Getting used to ESP-IDF

03 Connecting to WiFi

04 TCP/IP sockets

Session's content:

- Introduction to the ESP32 and its' variants
- Comparing ESP32 to Arduino UNO
- Getting started with IoT Development Framework
- Understand the “blink” example code



01

Getting Started with ESP32

Introduction

ESP32 is a low-cost dual core microcontroller developed by Espressif Systems. It has both WiFi and Bluetooth integrated with various peripherals.

ESP32 vs Arduino UNO

Specs	ESP32	Arduino UNO
CPU	Dual Core 160 MHz	Single Core 16MHz
Flash Memory	4MB	32KB
SRAM	520KB	2KB
GPIOs	38	20
WiFi/Bluetooth	Yes	No
UART (Hardware Serial)	3	1
SPI	3	1
I ² C	2	1
Operating Voltage	3.3V	5V
Price	2500DA/ 3.5\$	2800DA/ 4\$

Other ESPs



ESP8266



ESP32 Wroom



ESP32 Wrover



ESP32-S2



ESP32 PICO D4



ESP32-S3

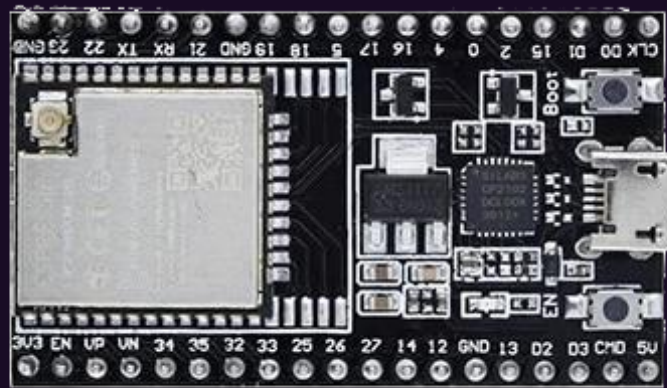
Development boards



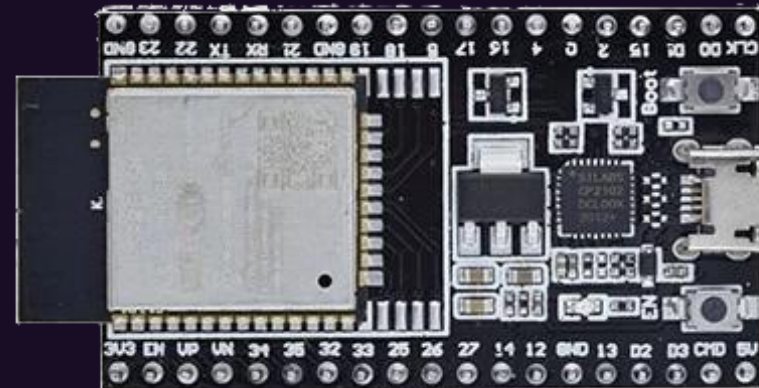
ESP-32 30PIN



ESP-32 38PIN



ESP32-WROOM-32U



ESP32-WROOM-32D

How to program the ESP32

Arduino Framework

Beginner friendly but thats it.
Very limited, lacks many features and
bad coding mindset (copy-paste)

IoT Dev Framework

Official framework supported by
Espressif. Extremely reliable and
configurable. Well documented with
growing community.

micropython, lua...

Just do not use this.
You do not want a high level language
when dealing with microcontrollers.
C/C++ is the way to go with
embedded systems.

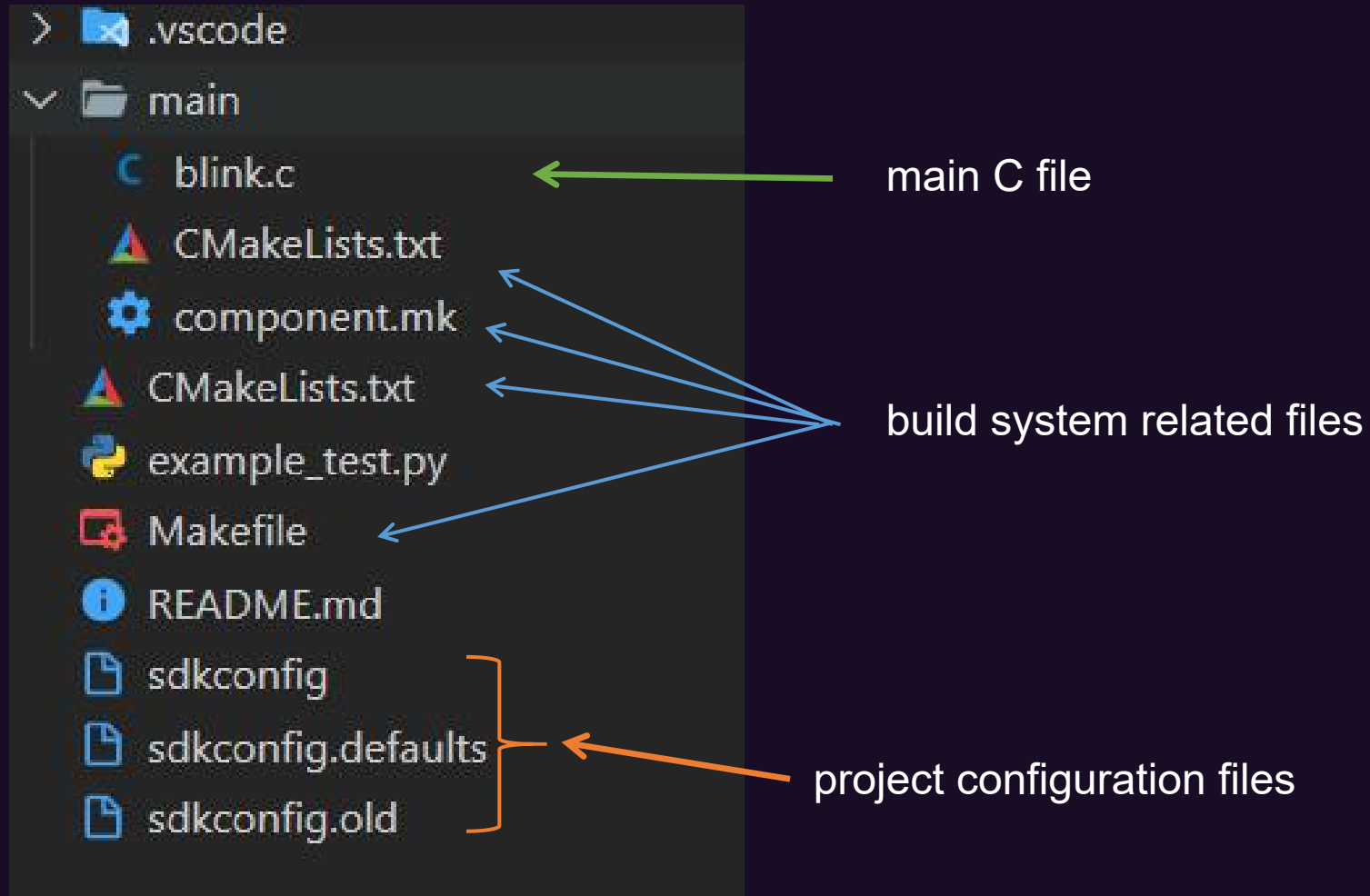
Setting up ESP-IDF

1. Install ESP-IDF with the esp-idf-tools-setup-offline-4.4.exe
2. Make sure the path you install esp-idf does not have space in it
3. Install VSCode
4. Install the Espressif IDF Extensions in VSCode
5. F1 or CTRL+Shift+P and search for “Configure ESP-IDF extension”
6. Choose “Use existing setup”

Your first ESP-IDF project

1. F1 or CTRL+Shift+P and search for “new project”
2. Set project name and directory then choose template
3. Pick the “blink” example as your project template

ESP-IDF Project Structure



ESP-IDF Extension status bar

Port

Target platform

Menu Config

Build

Flash

Monitor

New IDF
terminal



esp32



Project directory

Clean project

Build, Flash and Monitor

Arduino Framework vs ESP-IDF

Arduino	ESP-IDF
<code>pinMode(pin, mode)</code>	<code>gpio_set_direction(pin, mode)</code>
<code>digitalWrite(pin, state)</code>	<code>gpio_set_level(pin, state)</code>
<code>Serial.printf()</code>	<code>printf()</code>
<code>delay()</code>	<code>vTaskDelay()</code>
OUTPUT	<code>GPIO_MODE_OUTPUT</code>
INPUT	<code>GPIO_MODE_INPUT</code>

Thank you



Wameedh Scientific Club



02

Getting used to ESP-IDF

Session's content:

- Understand the “Hello World” code example
- Learn how to use the ESP-IDF documentation and ESP32 datasheet
- Use “ESP_LOGx” instead of printf to make log statements
- A bit on freeRTOS and menuconfig

Resources you need

Example codes: Example codes are the best way to understand how to use any feature of the ESP32

Source code of esp-idf: You can open the source code by right-clicking and click on “go to definition”

Docs: docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/

Datasheet: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

Logging in general:

Log is outputting informations about a software or microcontroller to the console. This can be done using simple print statements. However, it gets really hard to debug big projects without organizing logs. One logging convention is to split your logs into 5 log levels: Verbose, Debug, Info, Warn, Error

	Verbose Statement	Debug Statement	Info Statement	Warn Statement	Error Statement
Verbose Level	Included	Included	Included	Included	Included
Debug Level	Excluded	Included	Included	Included	Included
Info Level	Excluded	Excluded	Included	Included	Included
Warn Level	Excluded	Excluded	Excluded	Included	Included
Error Level	Excluded	Excluded	Excluded	Excluded	Included
Off	Excluded	Excluded	Excluded	Excluded	Excluded

Logging in ESP-IDF:

Logs in ESP-IDF is achieved with esp_log.h library. The output has the following format:


Log Level (Time) TAG: Log Content

```
I (301) TAG: Information log: hello world
W (301) TAG: Warning log: hello world
E (311) TAG: Error log: hello world
```

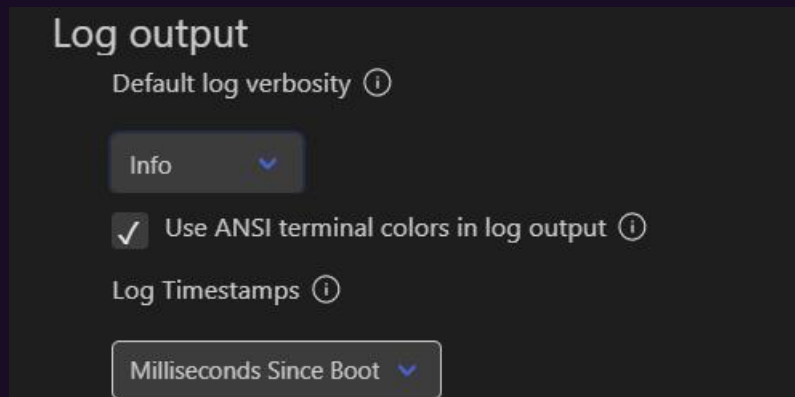
This is achieved with the following code

```
char data[] = "hello world";
ESP_LOGI("TAG", "Information log: %s", data);
ESP_LOGW("TAG", "Warning log: %s", data);
ESP_LOGE("TAG", "Error log: %s", data);
```

Introduction to MenuConfig:

MenuConfig is simply a configuration interface. You can find many options related to both the build system and esp-idf components. It can be accessed using the menuconfig icon in the status bar 

One of the things you can with menuconfig is change settings related to the serial monitor of the esp-idf. You can change log settings like log level (default is INFO) or the baudrate (default is 115200)



Introduction to freeRTOS:

freeRTOS is one of the most popular real-time operating systems for embedded systems. It is used in 35 different microcontrollers. An operating system is used to manage the execution of processes and provides the necessary resources to them (memory and cpu-time).

One of the benefits of using freeRTOS is the ability of multi-threading in embedded systems. Multi-threading is essentially running two different piece of code (functions) in parallel. They are called tasks with each task have its own priority.