UNIVERSITÉ CÔTE D'AZUR

POLYTECH NICE SOPHIA

Part 1
IoT Framework

Fabrice MULLER

Fabrice.Muller@univ-cotedazur.fr

2021 - 2022

itii PROVENCE ALPES CÔTE D'AZUR

What we will design … an IoT system

Photoresistor

Resistor

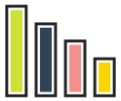
LED

Push button

Pressure/Temperature sensor

potentiometer

Water level sensor



Saving power !

SPI Reader

ESP NOW

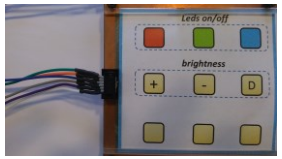
ESP32

Wi-Fi

USB

BLE-
Nimble

Touch Keyboard



LCD OLED SPI (SSD1306)

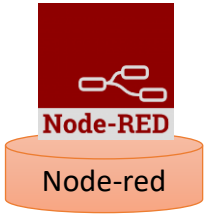
ESP32

versions of program

Internet

Google/One
Drive

Hot Spot AP

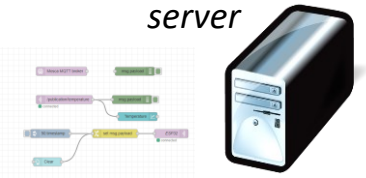


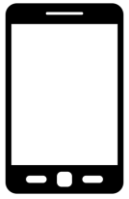
Node-RED

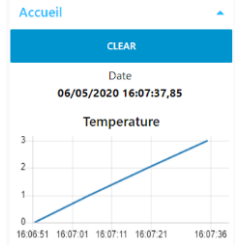
Node-red

Internet

server







What we will use … Software & Communication

Development Tool

Visual Studio Code

Node-Red

Microsoft GitHub

git

ESPRESSIF

CMake

Network/Cloud

Postman

mosquitto

MQTTBox

One Drive

Google Drive

Operating System

FreeRTOS

Ubuntu

Languages

THE C PROGRAMMING LANGUAGE

Java Script

CSS

HTML

C++

JSON

Doxygen

Android Tool

Beacon Scanner

BLE Scanner

LightBlue

Wireless Communication

Bluetooth LE - Nimble

WI-FI

ESP NOW ESP Now

Part 1 - IoT Framework

Lab 1 : Framework

- Working on Linux
- Espressif IoT Development Framework
- FreeRTOS
- Visual Code Studio
- GitHub
- Doxygen

Lab 2 : ESP32 Debug

- Debugging ESP32 program with JTAG

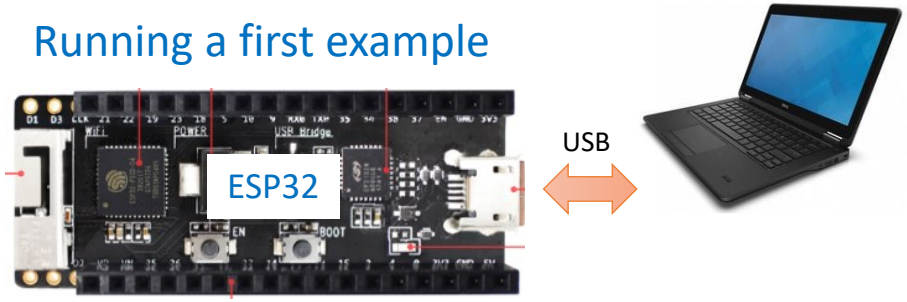
Lab 3 : Working with C and IDF framework

- Macro, header file
- Pointer, memory allocation, linked list
- Doxygen documentation

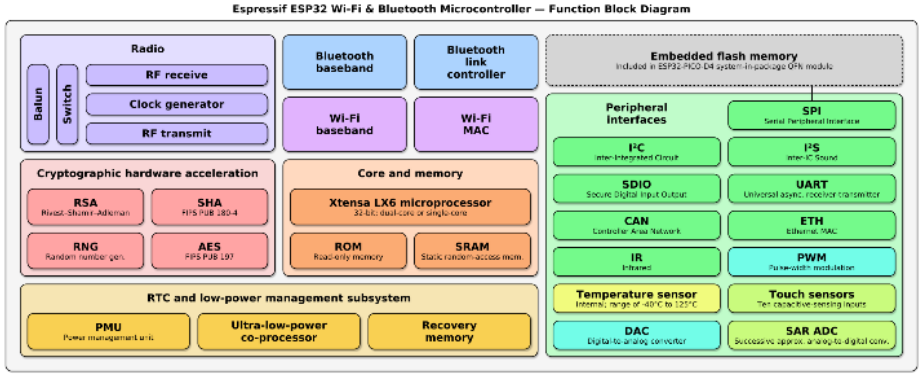
Lab 4 : Components & Configuration

- Components for ESP32
- Custom menu configuration
- Default configuration

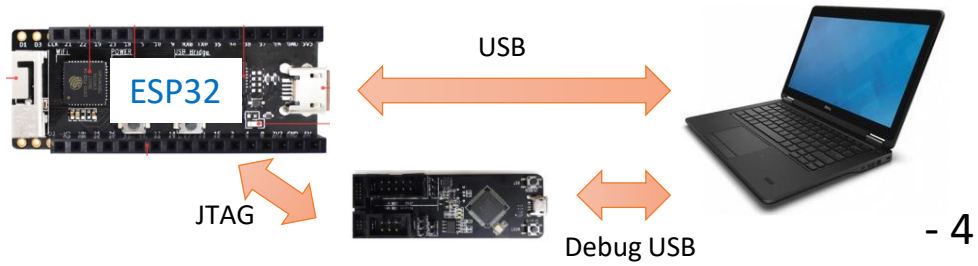
Running a first example



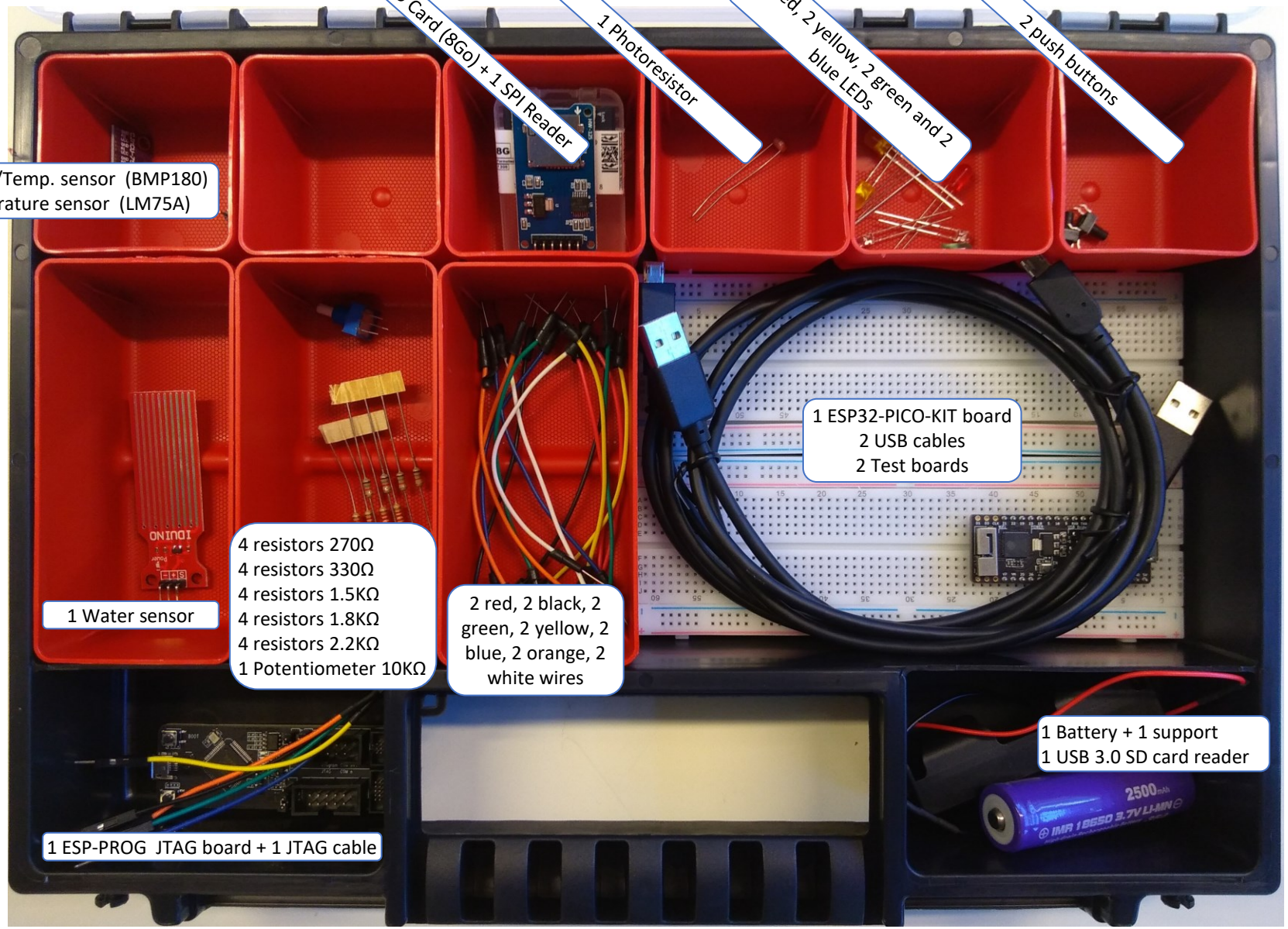
ESP32 Architecture

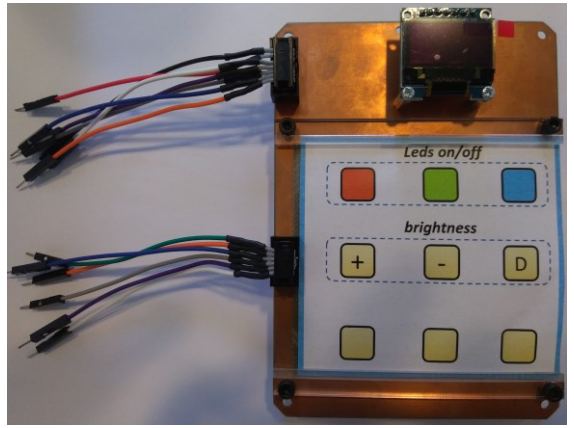


Debug example



Material


1 SD Card (8Go) + 1 SPI Reader
1 Photoresistor
2 red, 2 yellow, 2 green and 2 blue LEDs
2 push buttons
- Pressure/Temp. sensor (BMP180)
- Temperature sensor (LM75A)
1 ESP32-PICO-KIT board
2 USB cables
2 Test boards
1 Water sensor
4 resistors 270Ω
4 resistors 330Ω
4 resistors 1.5KΩ
4 resistors 1.8KΩ
4 resistors 2.2KΩ
1 Potentiometer 10KΩ
2 red, 2 black, 2 green, 2 yellow, 2 blue, 2 orange, 2 white wires
1 Battery + 1 support
1 USB 3.0 SD card reader
1 ESP-PROG JTAG board + 1 JTAG cable

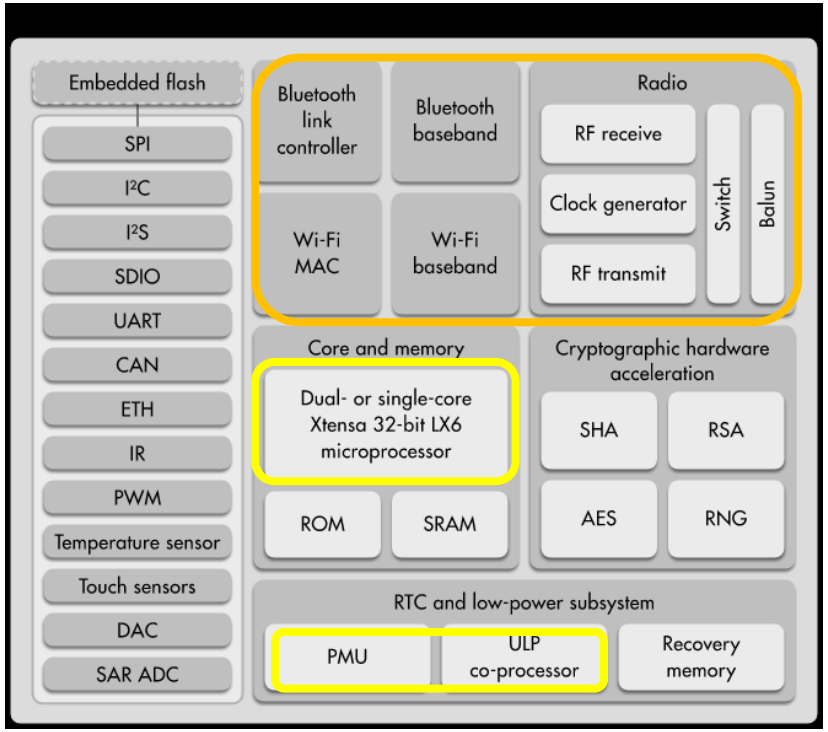

Leds on/off
brightness
+
-
D

ESP32 Features

ESP32 architecture

- Processors
  - Tensilica Xtensa 32-bit LX6 microprocessor
  - 1 or 2 Cores
  - up to 240 MHz for clock frequency
  - up to 600 DMIPS (Dhrystone MIPS)
  - Ultra Low Power (ULP) Coprocessor
  - Phasor measurement unit (PMU)
- Wireless connectivity
  - Wi-Fi: 802.11 b/g/n/e/i (802.11n @ 2.4 GHz up to 150 Mbit/s)
  - Bluetooth: v4.2 BR/EDR and Bluetooth Low Energy (BLE)

ESP32



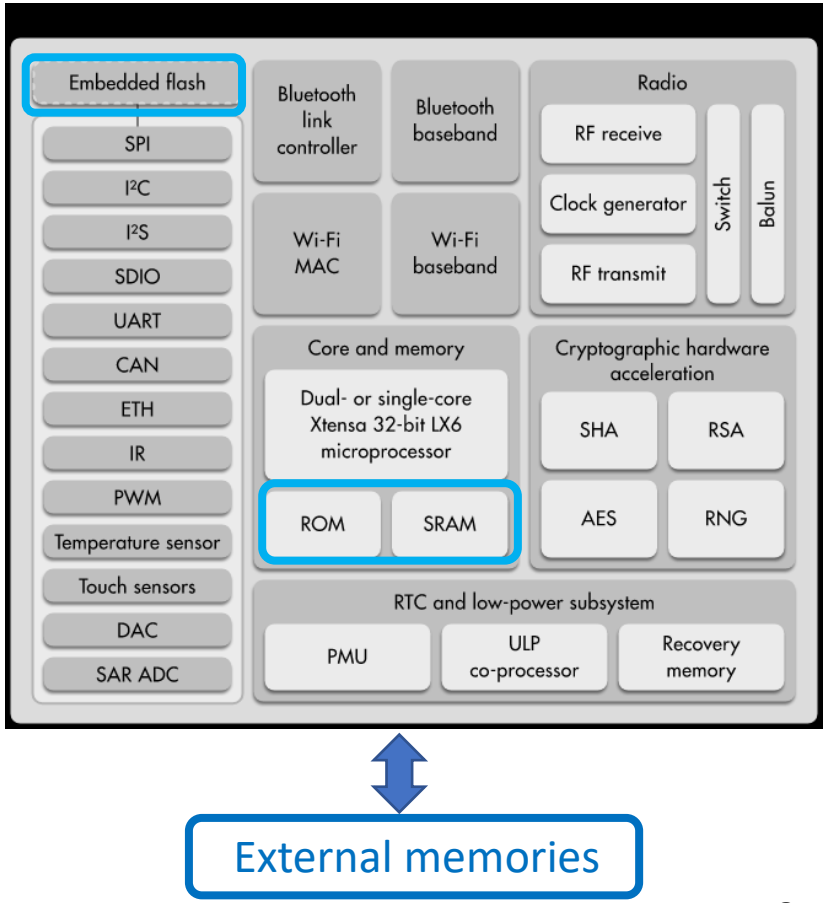
Tensilica Xtensa 32-bit LX6 microprocessor

- 7-stage pipeline to support clock frequency up to 240MHz
- 16/24-bit Instruction Set
- Floating Point Unit (FPU)
- DSP instructions, such as a 32-bit multiplier, a 32-bit divider, and a 40-bit MAC
- 32 interrupt vectors from about 70 interrupt sources
- Interfaces
  - Xtensa RAM/ROM Interface for instructions and data
  - Xtensa Local Memory Interface for fast peripheral register access
  - External and internal interrupt sources
  - JTAG for debugging

ESP32 architecture

- Internal memories
  - ROM: 448 KiB
  - SRAM: 520 KiB
  - RTC fast SRAM: 8 KiB
  - RTC slow SRAM: 8 KiB
  - eFuse: 1 Kibit
  - Embedded flash: 0/2/4 MiB
- External memories
  - Up to 16 MiB of external flash
  - Up to 8 MiB of SRAM memory

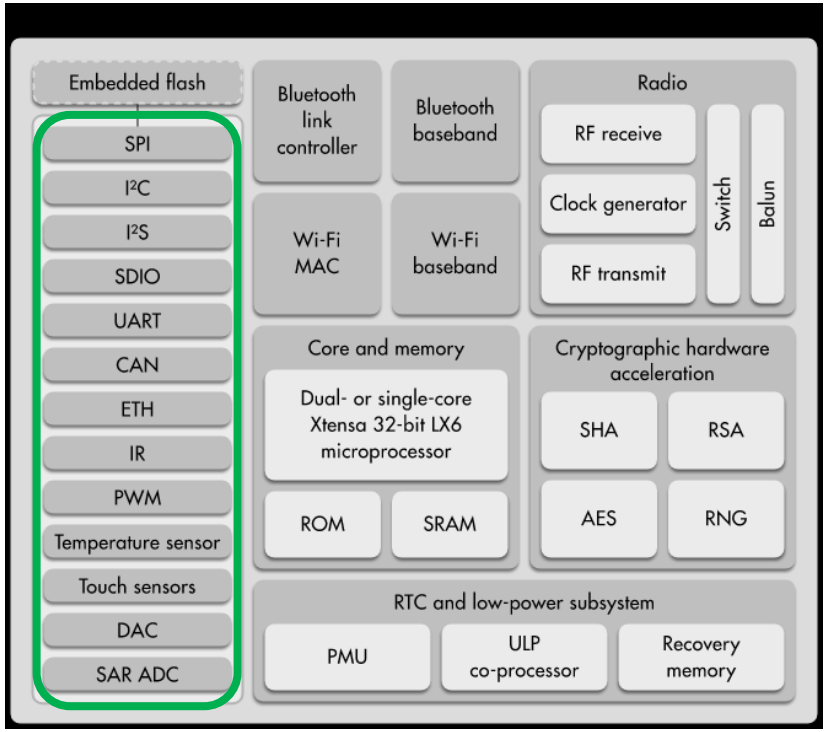
ESP32



ESP32 architecture

- Peripheral input/output
  - UART (universal asynchronous receiver/transmitter)
  - I²C (Inter-Integrated Circuit)
  - SPI (Serial Peripheral Interface)
  - ADCs (analog-to-digital converter),
  - DACs (digital-to-analog converter)
  - PWM (pulse width modulation)
  - Capacitive touch
  - I²S (Integrated Inter-IC Sound)
  - CAN 2.0 (Controller Area Network)
  - …

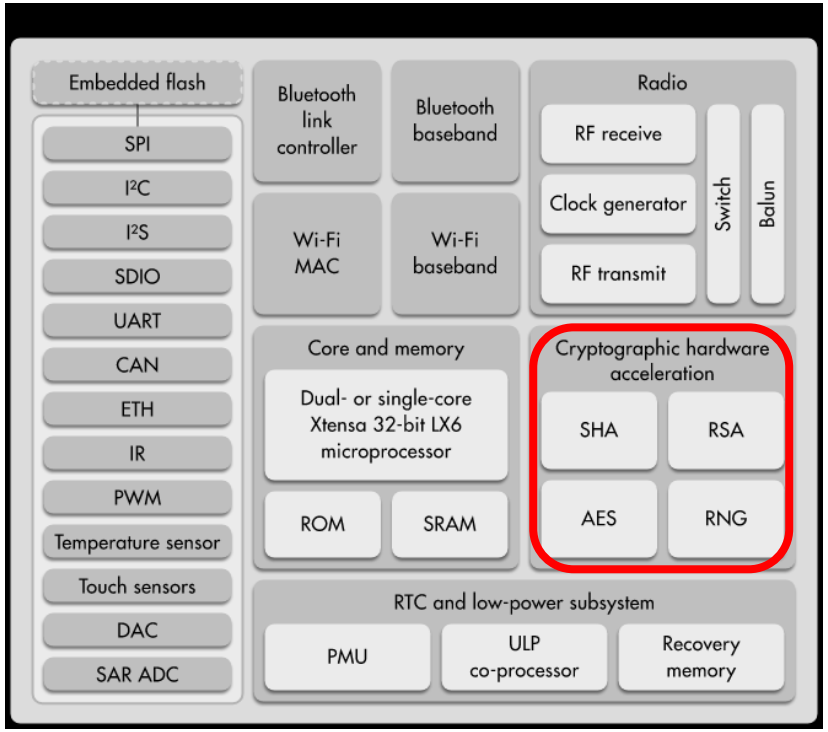
ESP32



ESP32 architecture

- Security
  - IEEE 802.11 standard security features (WFA, WPA/WPA2 and WAPI)
  - Secure boot
  - Flash encryption
  - 1024-bit OTP (One Time Programmable), up to 768-bit for customers
  - Cryptographic hardware acceleration
    - Random number generator (RNG)
    - AES, SHA-2, RSA
    - Elliptic curve cryptography (ECC)

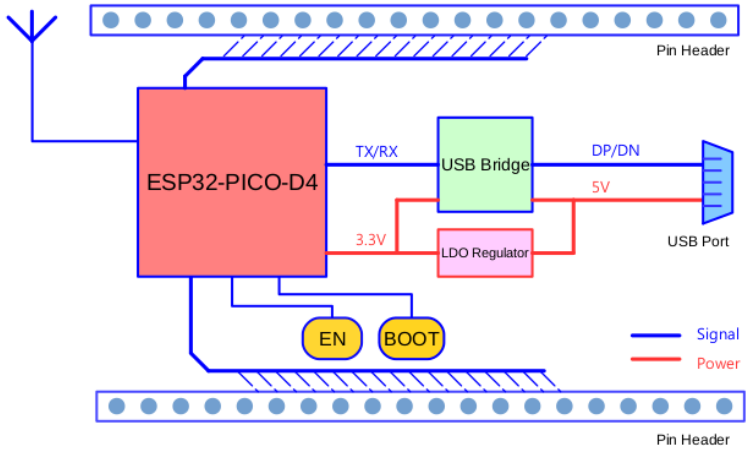
ESP32

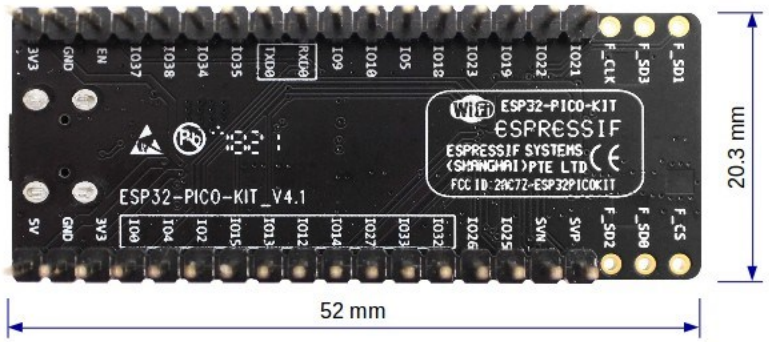


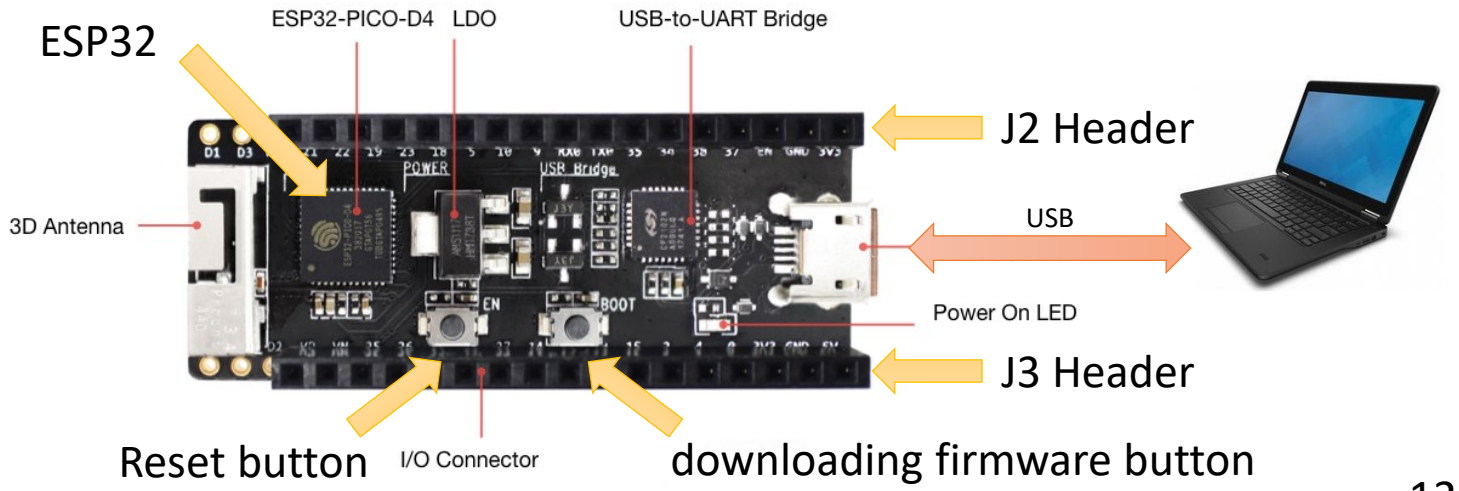
ESP32 Board - ESP32-PICO-KIT

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-pico-kit.html

- System-in-Package (SiP) : ESP32-PICO-D4
- Including
  - 40 MHz crystal oscillator
  - 4 MiB flash
  - Filter capacitors and RF matching links in
- USB-UART bridge (up to 3 Mbps transfers rates)
- Buttons
  - BOOT : press for downloading firmware through the serial port.
  - EN: Reset







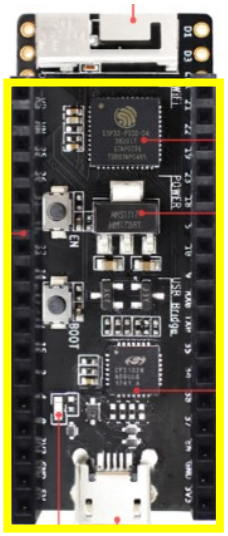
ESP32-PICO-KIT - Pin Descriptions

J3 Header

No.	Name	Type	Function
1	FLASH_CS (FCS)	I/O	GPIO16, HS1_DATA4 (See 1) , U2RXD, EMAC_CLK_OUT
2	FLASH_SD0 (FSD0)	I/O	GPIO17, HS1_DATA5 (See 1) , U2TXD, EMAC_CLK_OUT_180
3	FLASH_SD2 (FSD2)	I/O	GPIO11, SD_CMD, SPICS0, HS1_CMD (See 1) , U1RTS
4	SENSOR_VP (FSVP)	I	GPIO36, ADC1_CH0, RTC_GPIO0
5	SENSOR_VN (FSVN)	I	GPIO39, ADC1_CH3, RTC_GPIO3
6	IO25	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
7	IO26	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1
8	IO32	I/O	32K_XP (See 2a) , ADC1_CH4, TOUCH9, RTC_GPIO9
9	IO33	I/O	32K_XN (See 2b) , ADC1_CH5, TOUCH8, RTC_GPIO8
10	IO27	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17 EMAC_RX_DV
11	IO14	I/O	ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
12	IO12	I/O	ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI (See 4) , HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
13	IO13	I/O	ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
14	IO15	I/O	ADC2_CH3, TOUCH3, RTC_GPIO13, MTDO, HSPICS0 HS2_CMD, SD_CMD, EMAC_RXD3
15	IO2	I/O	ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
16	IO4	I/O	ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
17	IO0	I/O	ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1 EMAC_TX_CLK
18	VDD33 (3V3)	P	3.3V power supply
19	GND	P	Ground
20	EXT_5V (5V)	P	5V power supply

3. This pin is connected to the pin of the USB bridge chip on the board.
4. The operating voltage of ESP32-PICO-KIT's embedded SPI flash is 3.3V. Therefore, the strapping pin MTDI should hold bit zero during the module power-on reset. If connected, please make sure that this pin is not held up on reset.

ESP32-PICO-D4



USB

J3 Header

J2 Header

J2 Header

No.	Name	Type	Function
1	FLASH_SD1 (FSD1)	I/O	GPIO8, SD_DATA1, SPID, HS1_DATA1 (See 1) , U2CTS
2	FLASH_SD3 (FSD3)	I/O	GPIO7, SD_DATA0, SPIQ, HS1_DATA0 (See 1) , U2RTS
3	FLASH_CLK (FCLK)	I/O	GPIO6, SD_CLK, SPICLK, HS1_CLK (See 1) , U1CTS
4	IO21	I/O	GPIO21, VSPIHD, EMAC_TX_EN
5	IO22	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
6	IO19	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
7	IO23	I/O	GPIO23, VSPID, HS1_STROBE
8	IO18	I/O	GPIO18, VSPICLK, HS1_DATA7
9	IO5	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
10	IO10	I/O	GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD
11	IO9	I/O	GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD
12	RXD0	I/O	GPIO3, U0RXD (See 3) , CLK_OUT2
13	TXD0	I/O	GPIO1, U0TXD (See 3) , CLK_OUT3, EMAC_RXD2
14	IO35	I	ADC1_CH7, RTC_GPIO5
15	IO34	I	ADC1_CH6, RTC_GPIO4
16	IO38	I	GPIO38, ADC1_CH2, RTC_GPIO2
17	IO37	I	GPIO37, ADC1_CH1, RTC_GPIO1
18	EN	I	CHIP_PU
19	GND	P	Ground
20	VDD33 (3V3)	P	3.3V power supply

1. This pin is connected to the flash pin of ESP32-PICO-D4.
2. 32.768 kHz crystal oscillator: a) input b) output

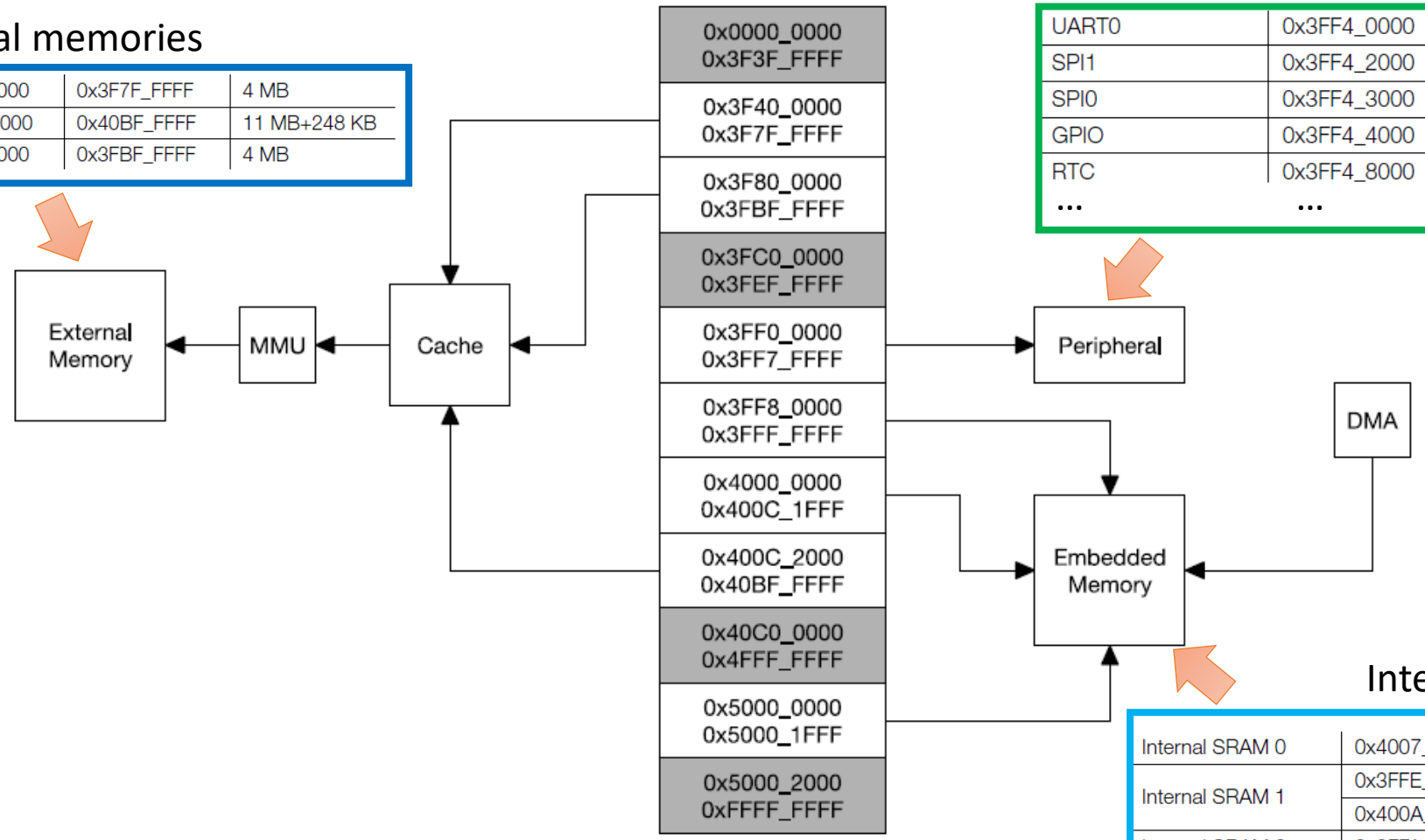
Tensilica Xtensa 32-bit LX6 microprocessor Memory Map

External memories

External Flash	0x3F40_0000	0x3F7F_FFFF	4 MB
	0x400C_2000	0x40BF_FFFF	11 MB+248 KB
External RAM	0x3F80_0000	0x3FBF_FFFF	4 MB

Peripheral input/output

UART0	0x3FF4_0000	0x3FF4_0FFF	4 KB
SPI1	0x3FF4_2000	0x3FF4_2FFF	4 KB
SPI0	0x3FF4_3000	0x3FF4_3FFF	4 KB
GPIO	0x3FF4_4000	0x3FF4_4FFF	4 KB
RTC	0x3FF4_8000	0x3FF4_8FFF	4 KB
...	...	...	...



Internal memories

Internal SRAM 0	0x4007_0000	0x4009_FFFF	192 KB
Internal SRAM 1	0x3FFE_0000	0x3FFF_FFFF	128 KB
	0x400A_0000	0x400B_FFFF	
Internal SRAM 2	0x3FFA_E000	0x3FFD_FFFF	200 KB

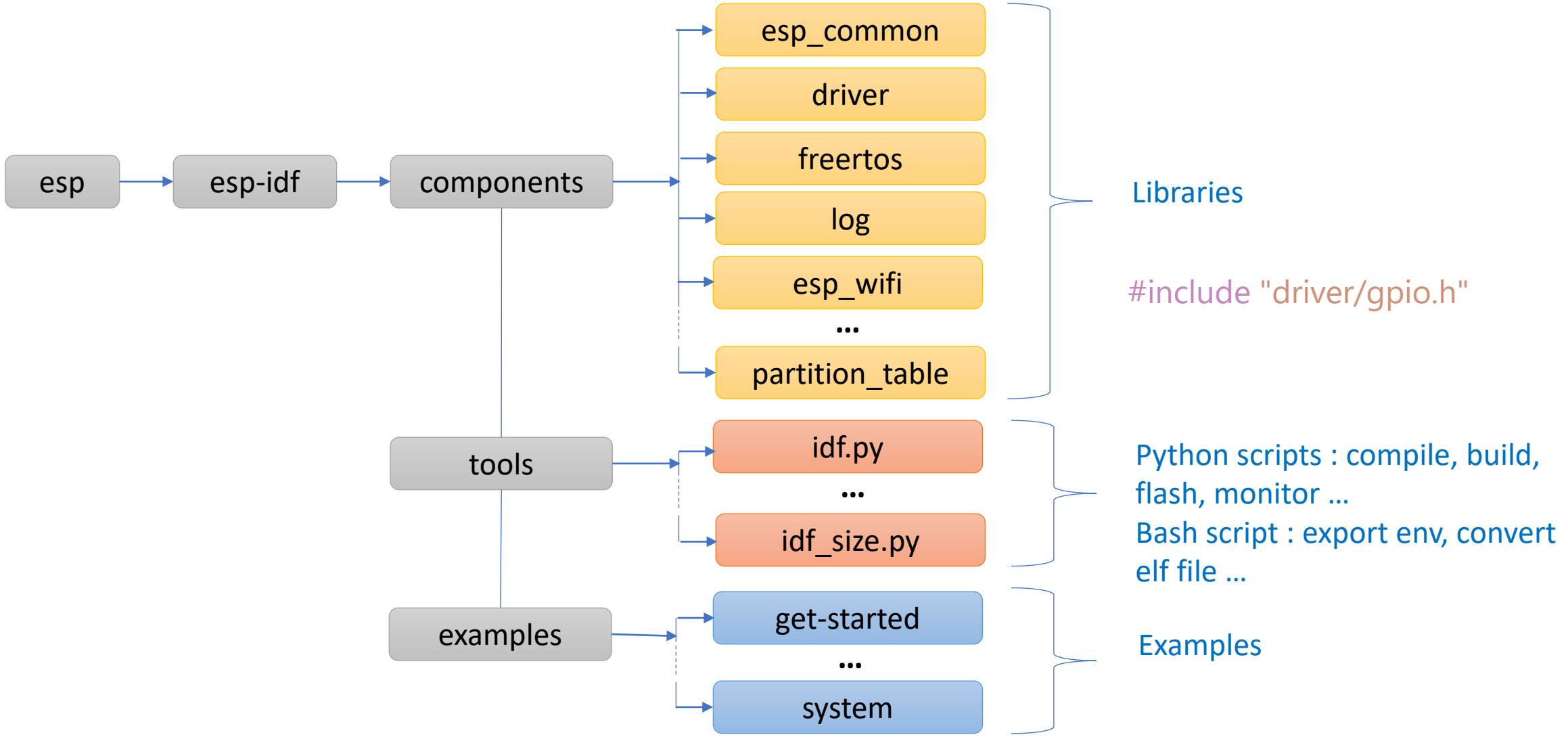
Development tool

Espressif IoT Development Framework

Espressif IoT Development Framework

- **Esp**ressif **I**oT **D**evelopment **F**ramework = **ESP-IDF**
- Included
  - Libraries
  - Tools
  - Examples
- ESP-IDF Programming Guide
  - https://docs.espressif.com/projects/esp-idf/en/latest/esp32/

ESP-IDF folder structure



Development tool

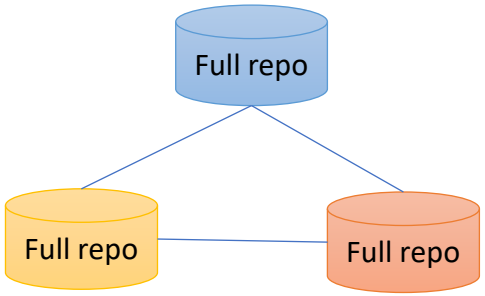
Git & GitHub / Visual Studio Code / Doxygen

Version Control System & Git

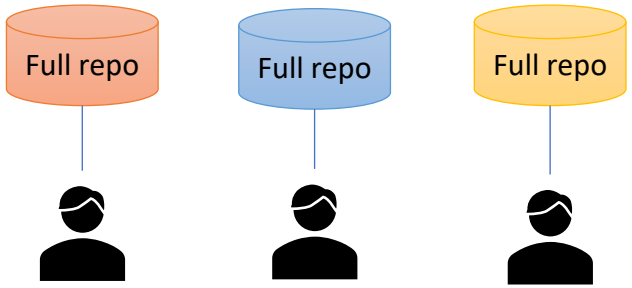
- Decentralized Version Control System (DVCS)
- Helps a development team to manage the changes to source code
- Integrated in lot of IDE
- Benefits
  - A complete history of long-term changes to each file
  - Team members can work simultaneously by creating branches
  - Traceability. Be able to track each change made to the software and connect it to project management and bug tracking software
  - Developers can go back and compare earlier versions of the code
- Goals
  - Performance: file content, differential encoding, compression, decentralization, remote repository
  - Security: hash algorithm (SHA1), protects code and change history against accidental or malicious modification
  - Flexibility: support for various development workflows, compatibility with many existing systems and protocols

Git - Principle

each developer gets their own local repository, complete with a full history of commits.



Having a full local history, it means you don't need a network connection to create commits.



Some commands

- Git global setup

  ```
  git config --global user.name "login"
  git config --global user.email "prenom.nom@univ-cotedazur.fr"
  ```

- Clone the central repository

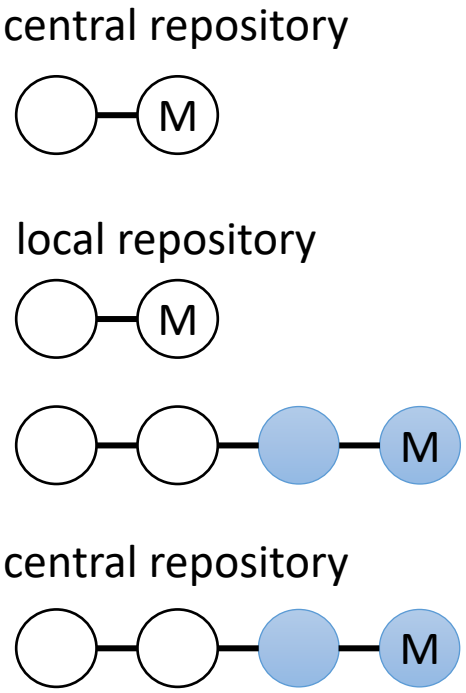
  ```
  git clone https://github.com/fmuller-pns/esp32-vscode-project-template.git
  cd esp32-vscode-project-template
  ```

- Make changes and commit

  ```
  git status                      View the state of the repo
  git add <files>                 Stage a file
  git commit -m "my comment"      Commit a file <files>
  ```

- Push new commits to central repository

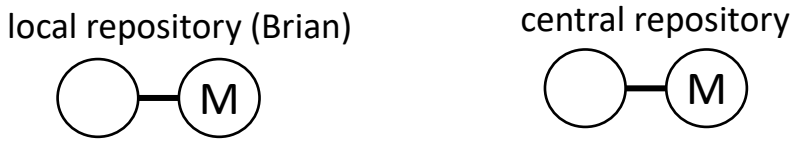
  ```
  git push -u origin master
  ```



Managing conflicts - Example

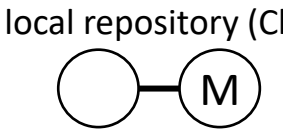
- Brian clones the PROJECT in his local repository

  git clone https://github.com/PROJECT.git



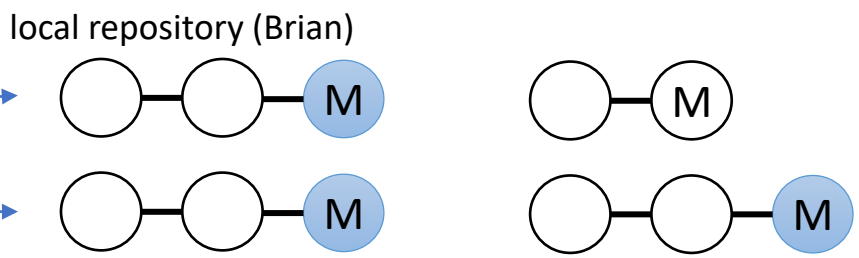
- Clara clones the PROJECT in her local repository

  git clone https://github.com/PROJECT.git



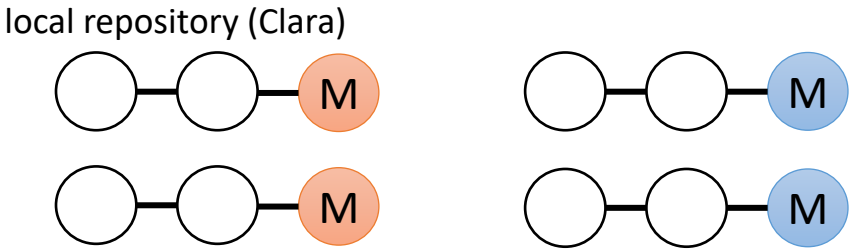
- Brian modifies and update the PROJECT in central repository

  git add <files>
  git commit -m "my comment“
  git push -u origin master



- Clara modifies and tries updating the PROJECT in central repository

  git add <files>
  git commit -m "my comment“
  git push -u origin master => ERROR

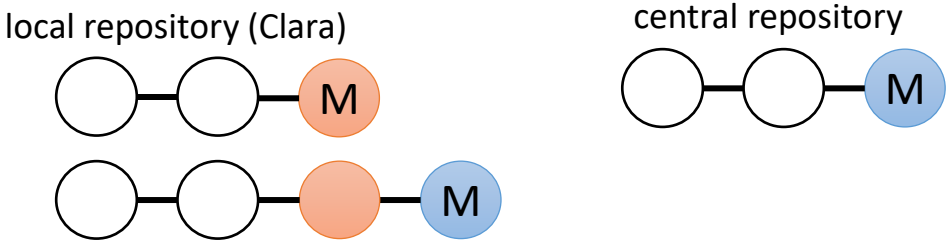


Updates were rejected because the tip of your current branch is behind

Managing conflicts - Example

- Clara incorporates changes into her local repository

  git pull --rebase origin master



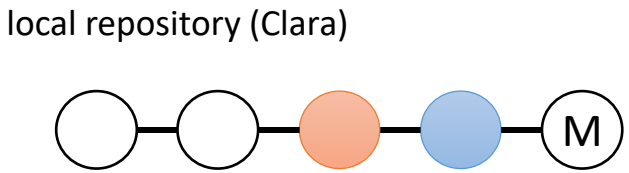
- Clara resolves a merge conflict

  Edit files …

  git add <some-file>

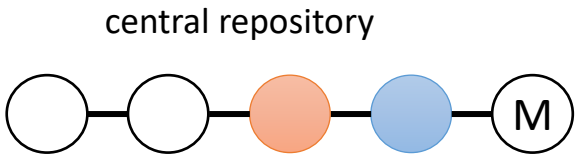
  git rebase --continue

  git status



- Clara publishes to central repository

  git push origin master



.gitignore file (1)

- Skip some unnecessary files
  - the executable (.exe), the temporary files
  - Object files .o .obj
- The ".gitignore" file
  - Add the ".gitignore" file to the root of your git
- How do I specify default content?
  - Web site: gitignore.io (https://www.gitignore.io/)
  - Specify, for example, "C"
  - Copy and save the *.gitignore* file
  - Specify a folder to ignore
  - Add for example “build/" to ignore the “build" folders

.gitignore file (2)

- If you forgot the .gitignore file
  - Add the .gitignore file on your local repository
  - Enter the following commands

```
git rm -r --cached .
git add .
```

  - Update the local repository

```
git commit -am "Remove skipped files"
```

  - Update the central repository

```
git push origin master
```

GitHub – For what ?

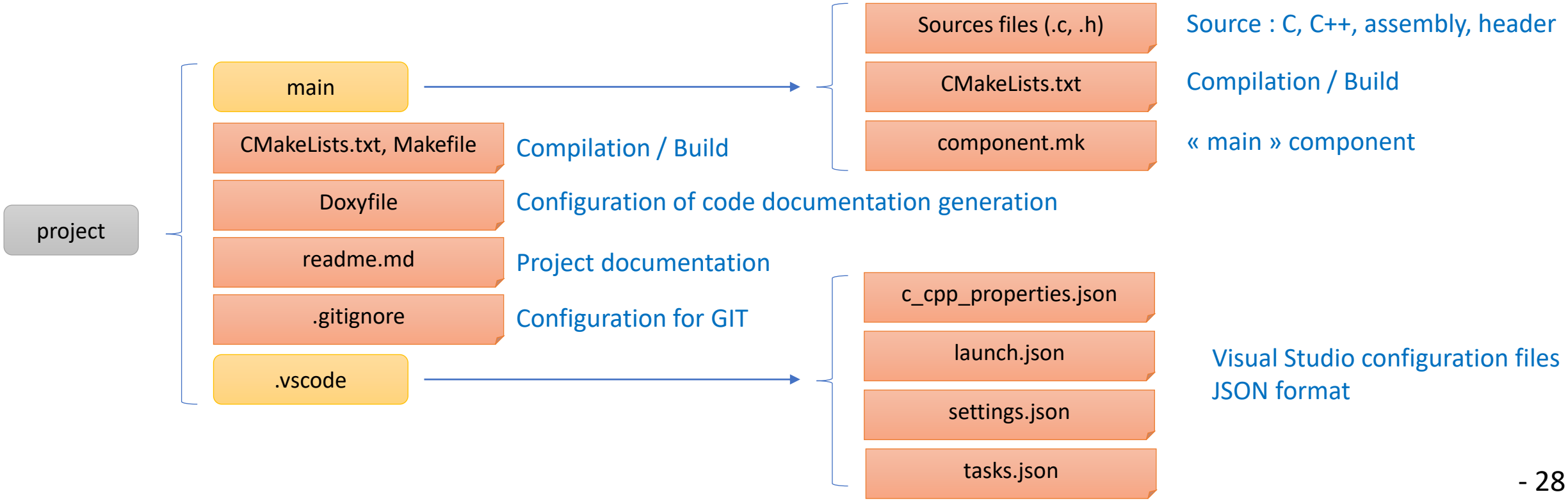
- GitHub provides hosting for software development and version control using Git.
- It has been a subsidiary of Microsoft since 2018
- Projects
  - https://github.com/
- Documentation
  - https://docs.github.com/
- Repository visibility changes
  - Public / Private
- Files
  - Source code (C, C++, Java …)
  - Documentation (including readme.md)
  - Configuration (tools …)
  - Images, video
  - …

GitHub - Markdown format

- Markdown is often used to format *readme files*
  - Lightweight markup language
  - Control the display of the document
  - Format words as bold or italic
  - add images
  - create lists
- File with the *.md or .markdown* extension
- Example
  - *readme.md* of *esp32-vscode-project-template* GitHub project
  - https://github.com/fmuller-pns/esp32-vscode-project-template
- Documentation
  - https://guides.github.com/features/mastering-markdown/

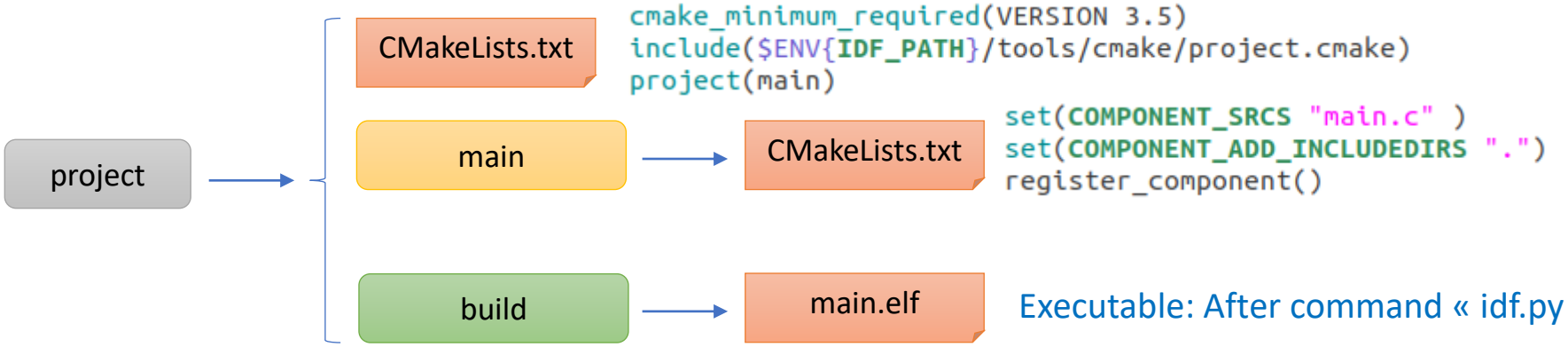
ESP32 project template

- For Visual Studio Code
- Located in « esp32-vscode-project-template » project
  - https://github.com/fmuller-pns/esp32-vscode-project-template



CMakeLists.txt & CMake

- CMake (cmake.org)
  - Cross-platform family of tools
  - Designed to build, test and package software
  - Used to control the software compilation process using simple platform and compiler independent configuration files
  - Generate native makefiles
  - Open-source
- File configuration : *CMakeLists.txt*
- ESP32 guide
  - https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/build-system.html#project-cmakelists-file
- idf.py (Python script) is a wrapper around CMake
  - idf.py build



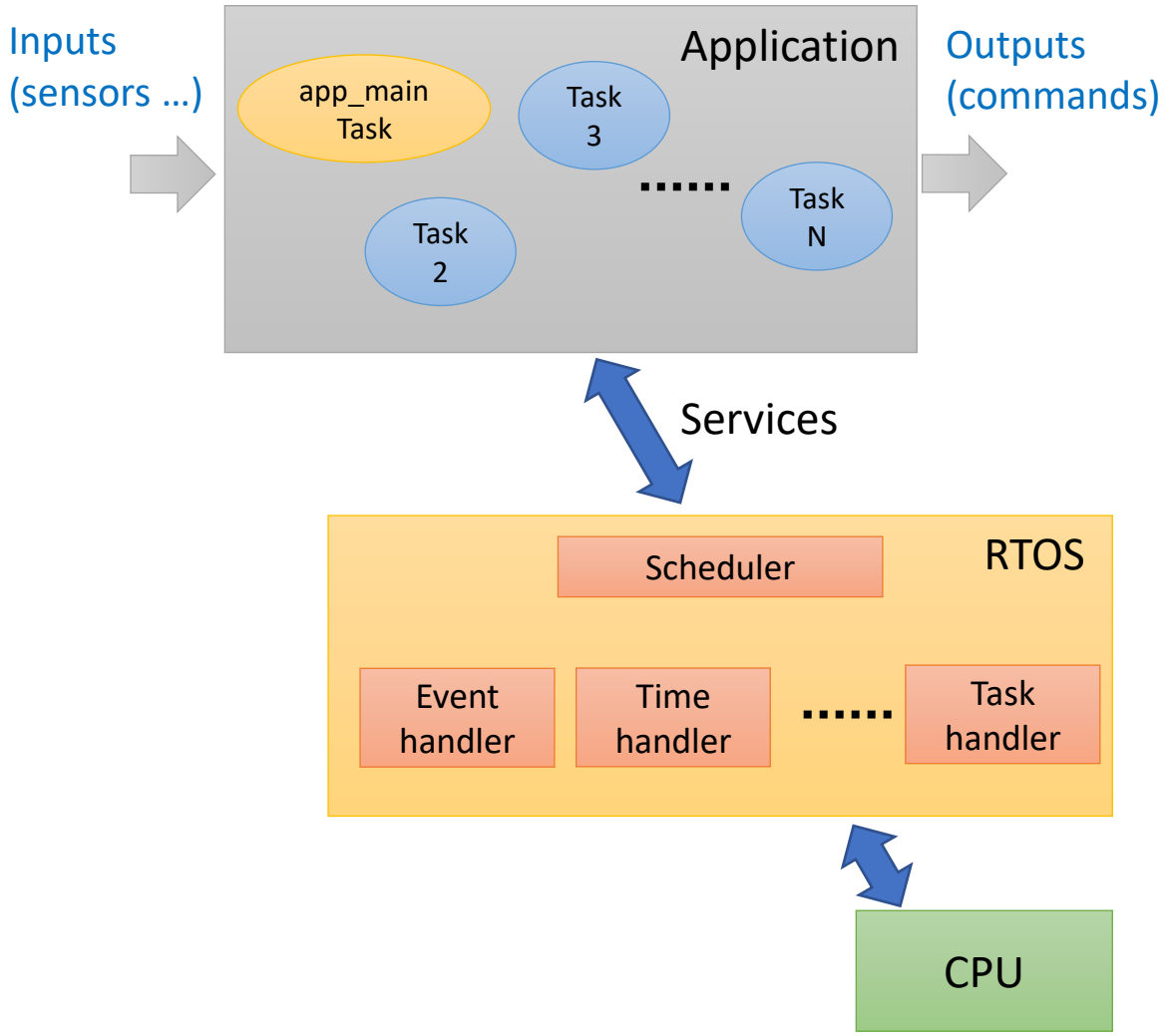
Visual Studio Code

- *.vscode* folder including configuration files
- JSON format (JavaScript Object Notation)
- Environment
  - IDF_TOOLS
  - IDF_PATH
- Configuration : esp32
  - name, browse
  - **includePath: important for components!**
- Miscellaneous
  - cStandard : c11 (ISO/IEC 9899:2011)
  - cppStandard : c++17 (ISO/IEC 14882)

```
{
    "env": {
        "IDF_TOOLS": "~/.espressif/tools",
        "IDF_PATH": "~/esp/esp-idf"
    },
    "configurations": [
        {
            "name": "esp32",
            "browse": {
                "path": [
                    "${workspaceFolder}",
                    "${IDF_PATH}",
                    "${IDF_TOOLS}"
                ],
                "limitSymbolsToIncludedHeaders": true
            },
            "includePath": [
                "${workspaceFolder}",
                "${workspaceFolder}/build/config",
                "${workspaceFolder}/build/bootloader/config",
                "${IDF_TOOLS}/xtensa-esp32-elf/esp-2019r2-8.2.
                "${IDF_TOOLS}/xtensa-esp32-elf/esp-2019r2-8.2.
                "${IDF_TOOLS}/xtensa-esp32-elf/esp-2019r2-8.2.
                "${IDF_TOOLS}/xtensa-esp32-elf/esp-2019r2-8.2.
                "${IDF_PATH}/components/newlib/include",
                "${IDF_PATH}/components/esp32/include",
                "${IDF_PATH}/components/soc/esp32/include",
            ],
            "defines": [],
            "cStandard": "c11",
            "cppStandard": "c++17",
            "intelliSenseMode": "clang-x64"
        }
    ],
    "version": 4
```

Using FreeRTOS on ESP32 boards

- RTOS = Real Time Operating System
- Starting point
  - *app_main()* task
- Input/output management
  - Input/output handler
  - Interrupt handler
- Task scheduling
  - Organization of functioning in tasks
  - Scheduling policy
  - Time handler
- Inter task communications
  - Synchronization (event)
  - Communication (data)
  - Access to a shared resource (data)
  - Time (counter, watchdog)



Part1 - Lab 1 - Framework

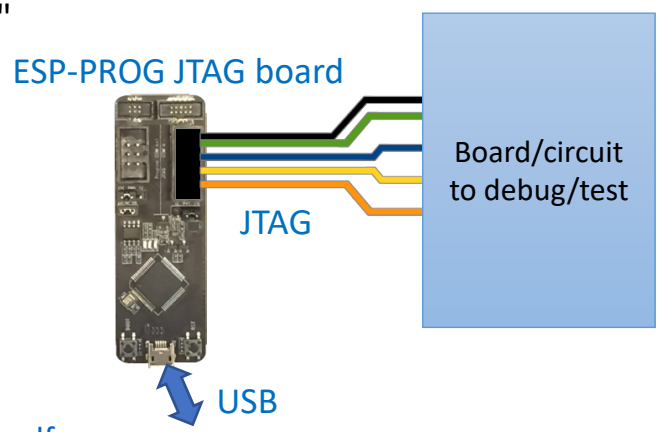
- Take an example of ESP-IDF : « hello_world »
- Visual Studio Code with ESP-IDF

- Help to create an new project
  - https://github.com/fmuller-pns/esp32-vscode-project-template

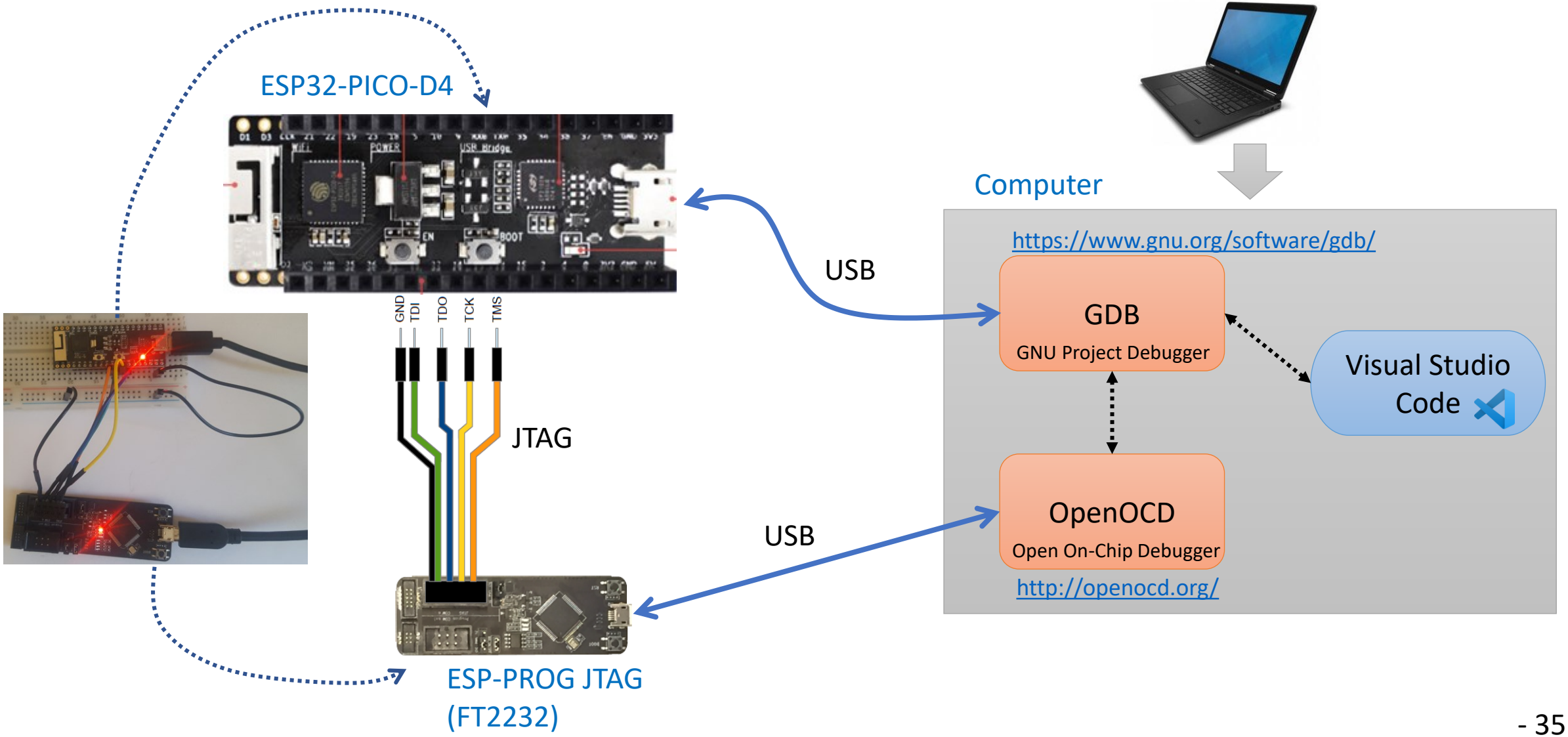
Debugging with OpenOCD

Overview - JTAG & FT2232

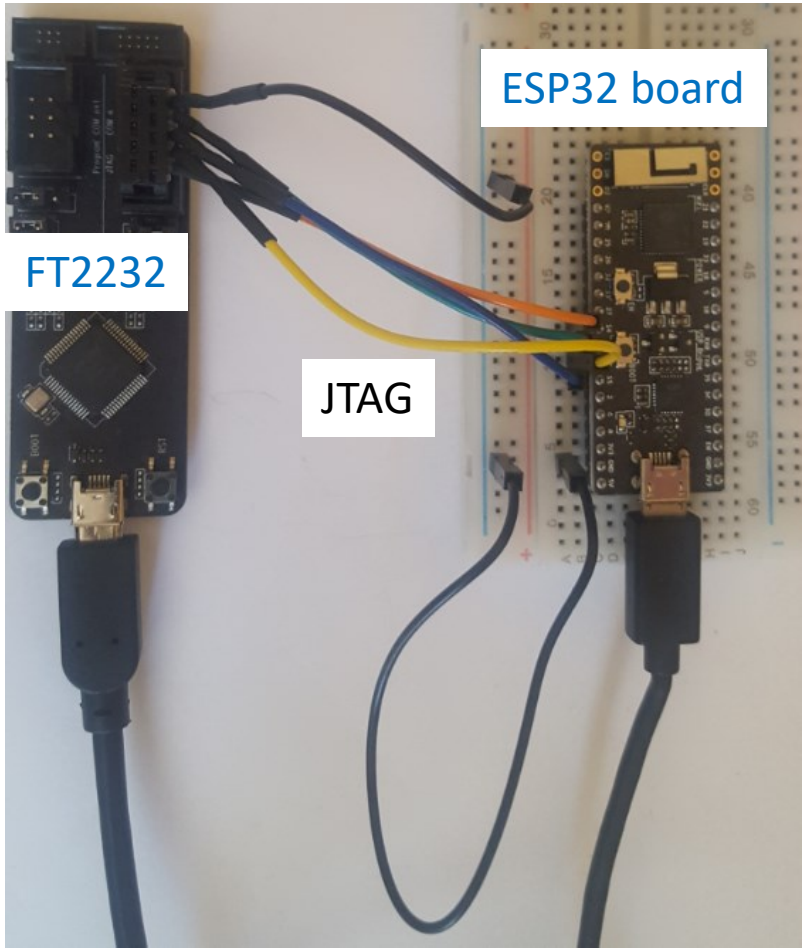
- JTAG = Joint Test Action Group
- Industry standard IEEE 1149.1 titled "Standard Test Access Port and Boundary-Scan Architecture"
- Test Access Port (TAP)
  - Test Data In (TDI)
  - Test Mode Select (TMS)
  - Test Clock (TCK)
  - Test Data Out (TDO)
  - Test Reset (TRST), Optionnal
- ESP-PROG JTAG board
  - FTDI Chip (Future Technology Devices International Ltd.), https://www.ftdichip.com/
  - Based on FT2232D circuit, https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT2232D.pdf
  - USB JTAG Programming, USB to SPI Bus Interfaces , USB to Dual Port RS232 Converters, …
- Use
  - **Debugging (breakpoint, step by step, state of variables, …)**
  - Transfer data into internal non-volatile device memory : CPLD, FPGA, Flash memory for storing firmware
  - Access to many logic signals of an integrated circuit, including the device pins Boundary scan testing
- JTAG Documentation
  - https://www.jtag.com/what-is-jtag-boundary-scan/
  - https://en.wikipedia.org/wiki/JTAG



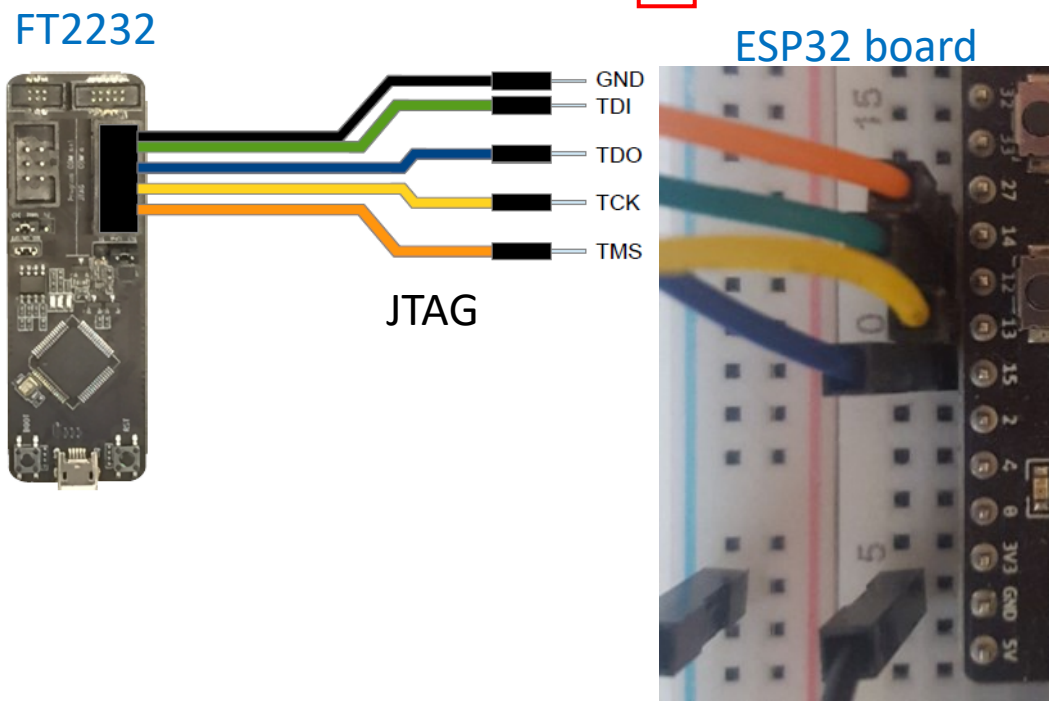
Overview – ESP32 workflow



ESP32/FT2232 JTAG connections

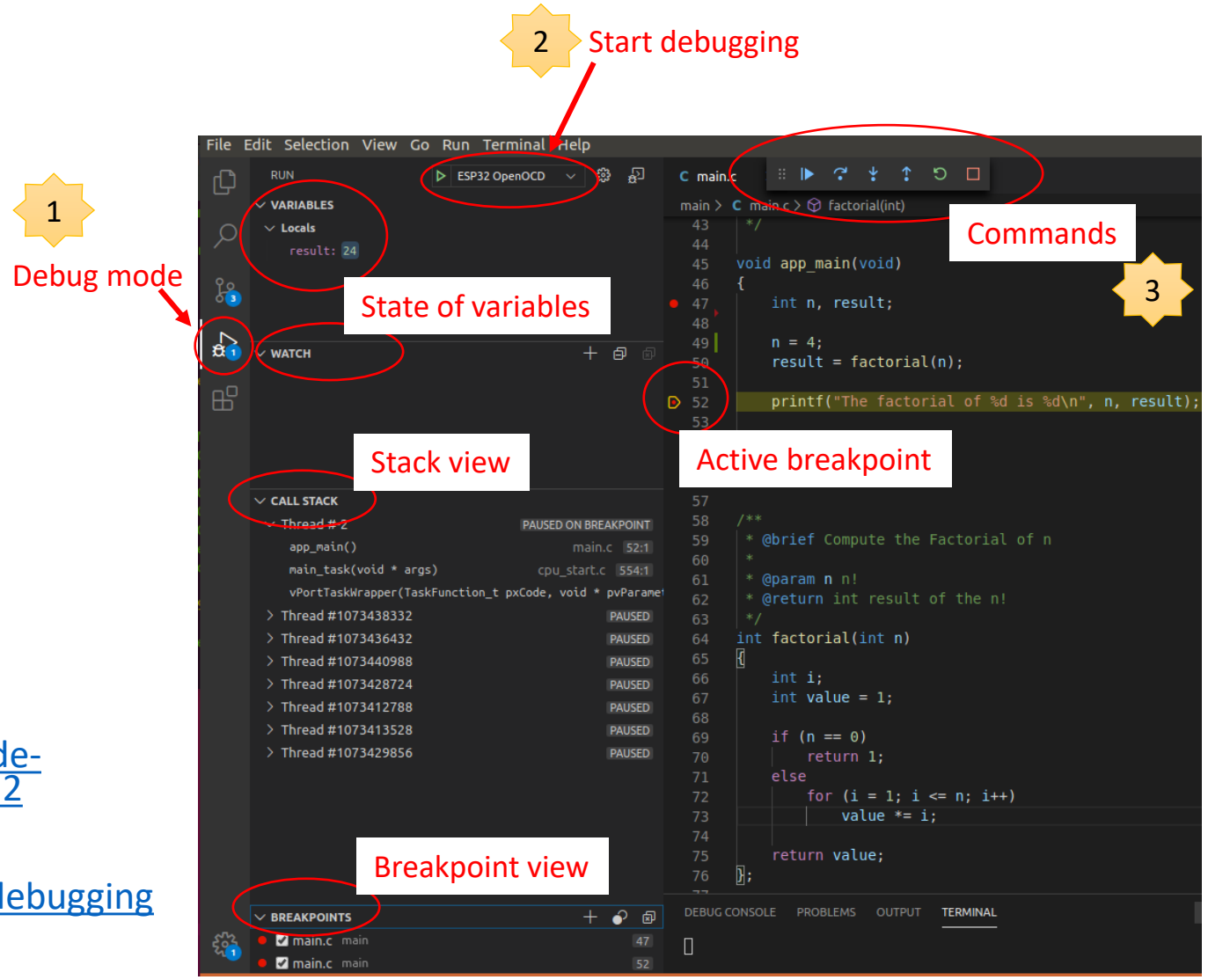


11	IO14	I/O	ADC2_CH6, TOUCH6, RTC_GPIO16, **MTMS**, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
12	IO12	I/O	ADC2_CH5, TOUCH5, RTC_GPIO15, **MTDI** (See 4) , HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
13	IO13	I/O	ADC2_CH4, TOUCH4, RTC_GPIO14, **MTCK**, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
14	IO15	I/O	ADC2_CH3, TOUCH3, RTC_GPIO13, **MTDO**, HSPICS0 HS2_CMD, SD_CMD, EMAC_RXD3



Debugging with Visual Studio Code

- Commands
  - Continue till next break
  - Step over
  - Step into
  - Step out
  - Restart
  - Stop
- Views
  - Variables
  - Stack
  - Breakpoint
- Guide
  - https://github.com/fmuller-pns/esp32-vscode-project-template#debugging-with-jtag-ft2232
- Documentation
  - https://code.visualstudio.com/docs/editor/debugging



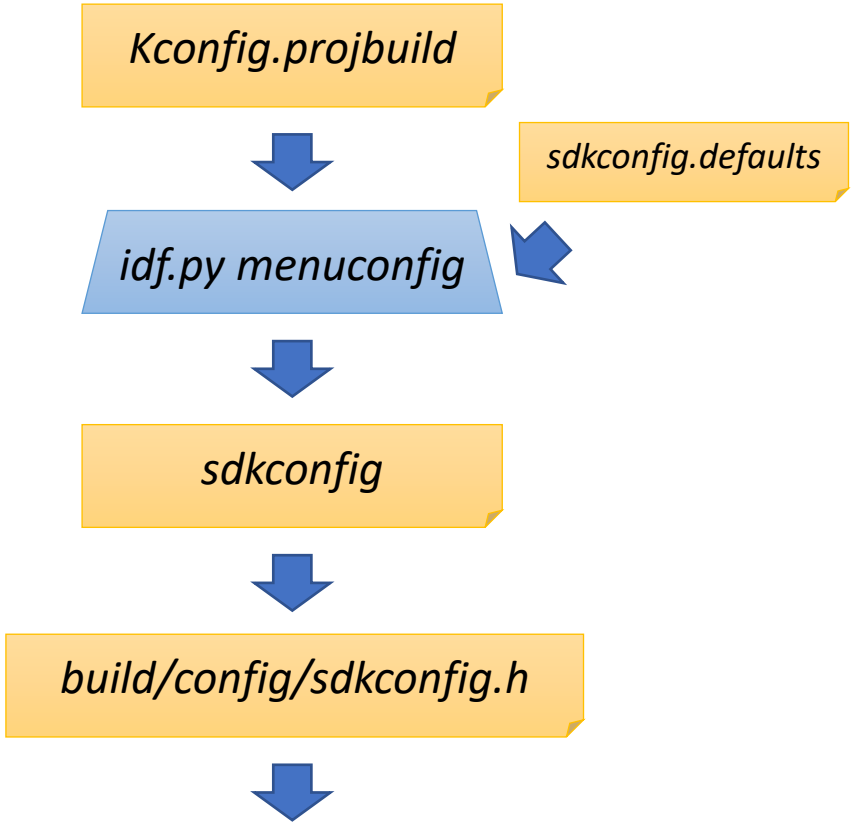
Project Configuration & IDF Components

How to configure complex project ?

- Compilation directives
  - #define, #ifdef …
  - Limited …
- Complex project
  - What action does each directive perform ?
  - Especially if they have dependencies between them.
  - Example: Linux Kernel, more than 4000 symbols!
- Solution
  - Using **kconfiglib** which is a Python-based extension to the **Kconfig** system
- More information
  - https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/kconfig.html
  - https://www.kernel.org/doc/Documentation/kbuild/kconfig-language.txt
  - https://docs.zephyrproject.org/latest/guides/kconfig/tips.html

Principle

- Create a configuration file
  - Create a *Kconfig.projbuild* located in main folder
  - Call python script
    - idf.py menuconfig
  - Navigate and modify configuration
- Create a default configuration file
  - Optional file
  - *sdkconfig.defaults*
- Generate *sdkconfig* file
- When building your project
  - Post-processes generate *sdkconfig.h*
  - Located in *build/config* folder



How to use the configuration?

DEFINED NAME in Header file = CONFIG_ + « CONFIG NAME »

Example of *Kconfig.projbuild*

```
menu "My Custom Menu"

config CUSTOM_FUNCTIONS
   bool "select for customSensorSet"
   default "n"

config GENERATE_RANDOM_VALUE
   bool
   choice
      prompt "select a choice to generate random values"
      default ABS_RANDOM

      config CONSTANT
         bool "Use constant: 2000"
      config ABS_RANDOM
         bool "abs(random())"
      config ONLY_RANDOM
         bool "random()"
   endchoice

endmenu
```

CONFIG_CUSTOM_FUNCTIONS

```
uint32_t h3 = esp_get_free_heap_size();

#ifdef CONFIG_CUSTOM_FUNCTIONS
customSensorSet->Update = updateSensorValues;
customSensorSet->Display = printSensorValues;
#endif

for (int j=0; j<5; j++) {
```

CONFIG_CONSTANT

CONFIG_ ABS_RANDOM

CONFIG_ ONLY_RANDOM

```
while (sensor) {

#ifdef CONFIG_CONSTANT
int random = 2000;
#elif ABS_RANDOM
int random = abs(esp_random());
#else
int random = esp_random();
#endif

switch (sensor->type) {
```

default configuration file (*sdkconfig.defaults*)

```
# User paramaters
CONFIG_CUSTOM_FUNCTIONS=y
GENERATE_RANDOM_VALUE=ONLY_RANDOM
```

IDF Components

- Modular pieces of standalone code
  - Compiled into static libraries
  - Linked into an app
- **Main directory** is a special component
- **Component directory** (optional)
  - Useful for structuring reusable code
  - EXTRA_COMPONENT_DIRS can be set in the top-level CMakeLists.txt to look for others components
- More information
  - https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/build-system.html#adding-conditional-configuration