HCMC University Of Technology, VNU-HCM
Faculty of Computer Science & Engineering

– Embedded System Lab 01 –

Introduction to ESP32 and ESP-IDF

Pham Hoang Anh & **Huynh Hoang Kha**

Goal In this lab, students are expected to be familiar with the NodeMCU ESP32 embedded board and the Espressif IoT Development Framework (esp-idf).

Content

- Introduction to ESP32
- Introduction to NodeMCU ESP32 board
- Introduction to FreeRTOS
- Introduction to ESP-IDF
- Getting started with ESP32

Prerequisite Requirement Have basic knowledge of operating systems and microcontroller programming.

Grading policy

- 40% in-class performance
- 60% report submission

1 Introduction to ESP32

The ESP32 is a dual-core system with two Harvard Architecture Xtensa LX6 CPUs. All embedded memory, external memory and peripherals are located on the data bus and/or the instruction bus of these CPUs.

ESP32 is a single 2.4 GHz Wi-Fi-and-Bluetooth combo chip designed with the TSMC ultra-low-power 40 nm technology. It is designed to achieve the best power and RF performance, showing robustness, versatility and reliability in a wide variety of applications and power scenarios.

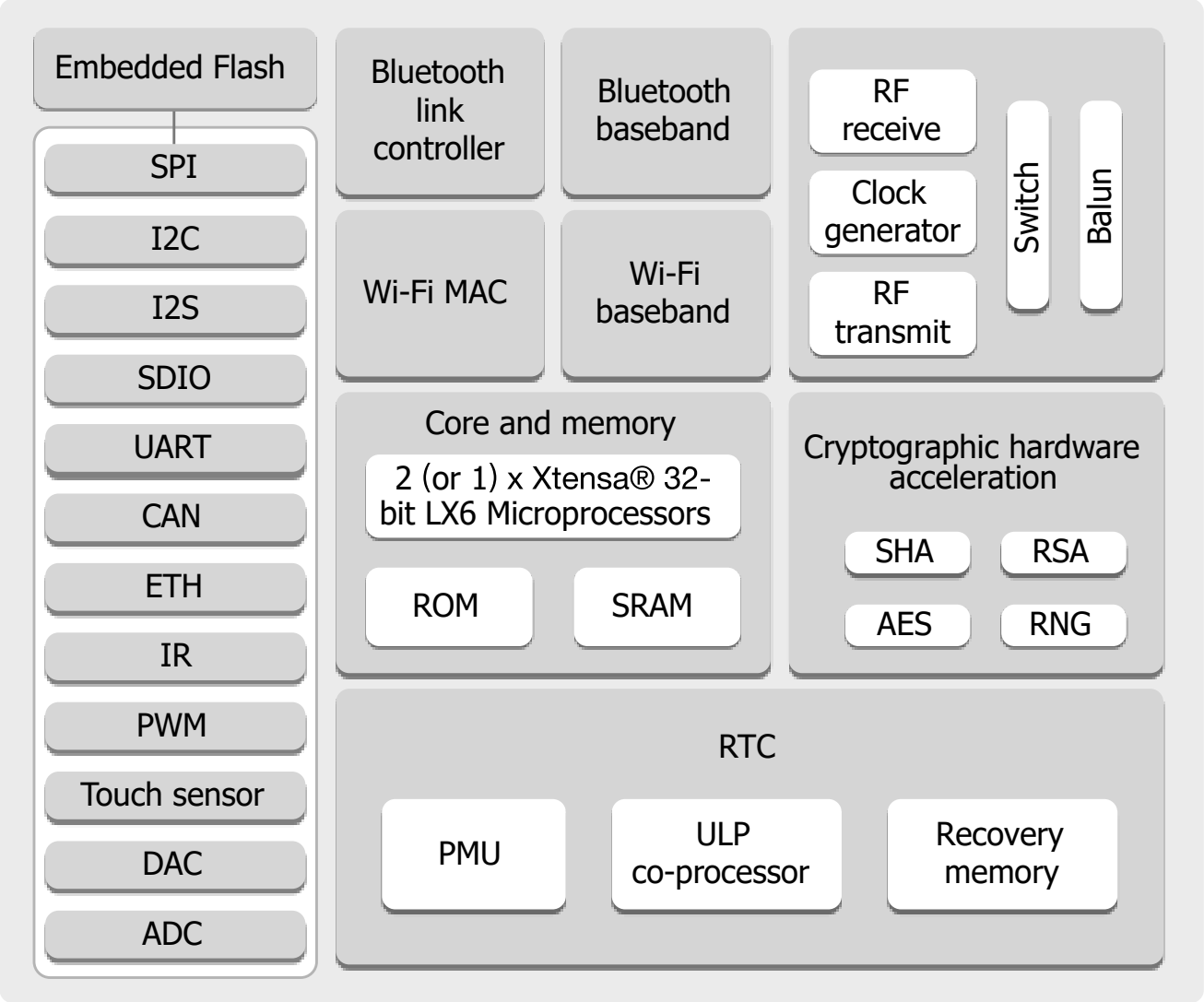

Figure 1: ESP32 functional block diagram

The two CPUs are named "PRO_CPU" and "APP_CPU" (for "protocol" and "application"), however, for most purposes the two CPUs are interchangeable.

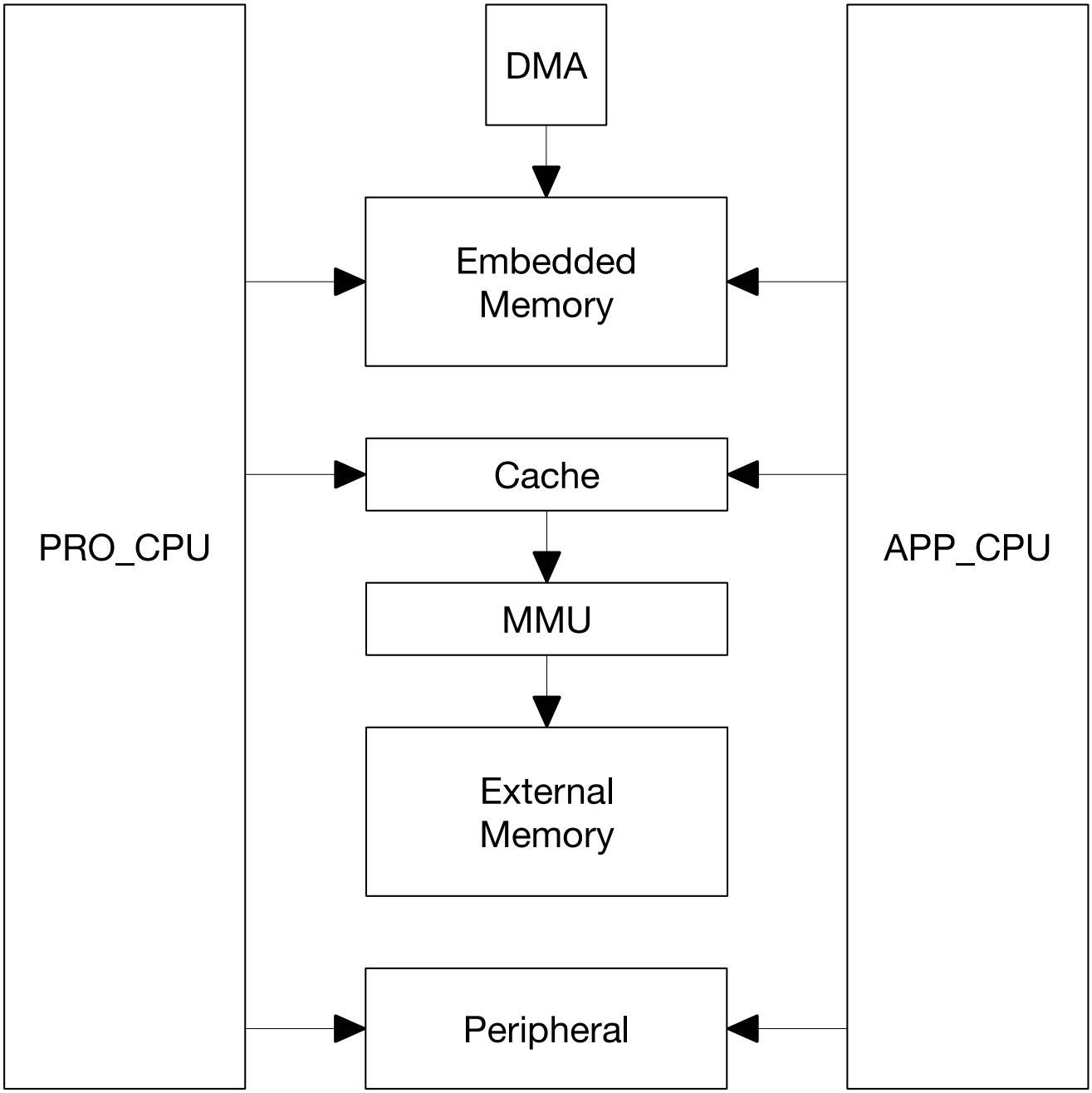

Figure 2: ESP32 system structure

2 Introduction to NodeMCU ESP32 board

NodeMCU is an open source Lua based firmware for the ESP32 and ESP8266 WiFi SOC from Espressif and uses an on-module flash-based SPIFFS file system. NodeMCU is implemented in C and is layered on the Espressif ESP-IDF.

The firmware was initially developed as is a companion project to the popular ESP8266-based NodeMCU development modules, but the project is now community-supported, and the firmware can now be run on any ESP module.

Figure 3 shows a snapshot of an ESP32 development board that will be used

in further labs later on.

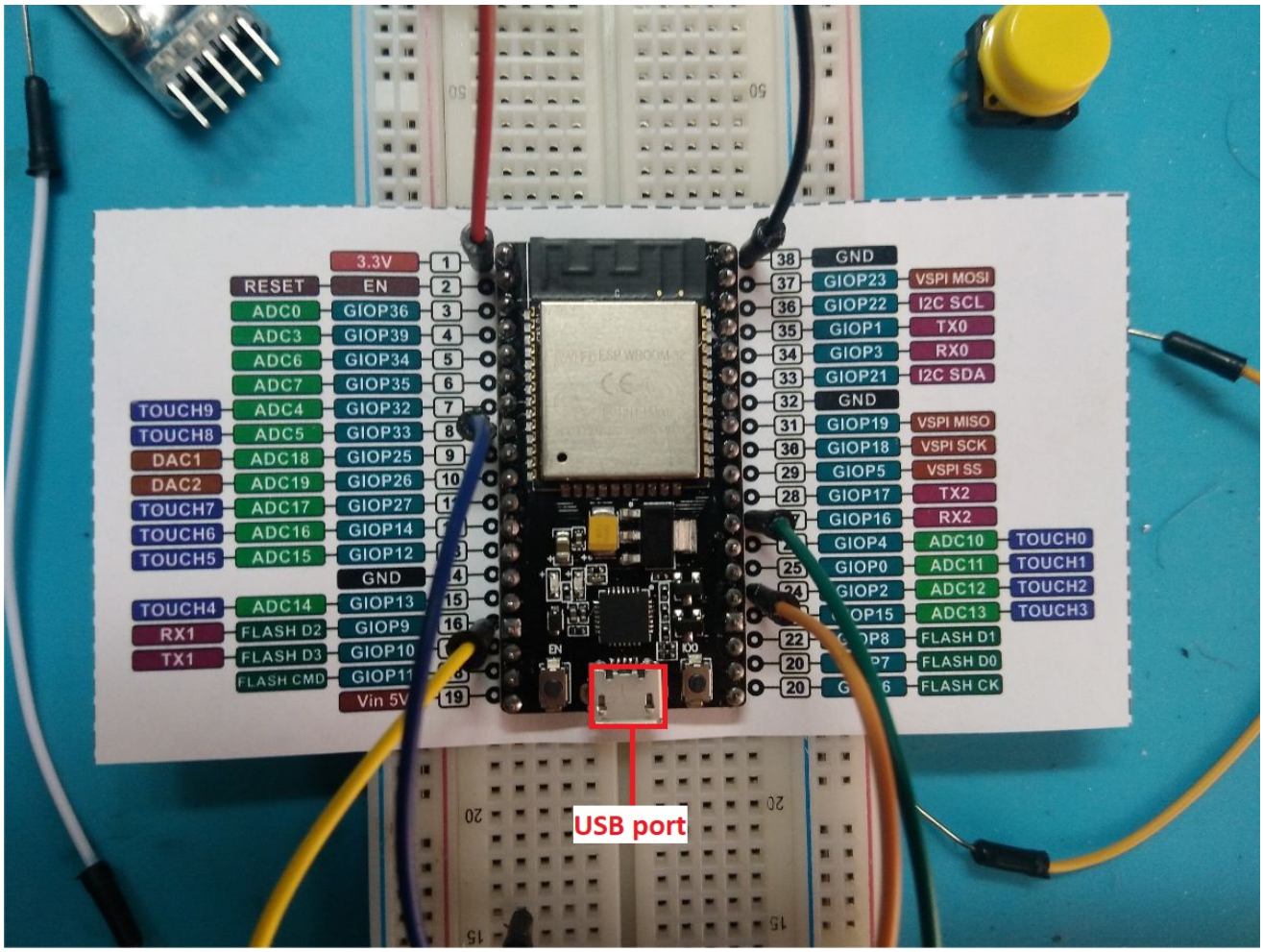

Figure 3: An ESP32 development board

3 Introduction to FreeRTOS

FreeRTOS is an free and open-source Real-Time Operating system developed by Real Time Engineers Ltd. Its design has been developed to fit on very small embedded systems and implements only a very minimalist set of functions: very basic handle of tasks and memory management, just sufficient API concerning synchronization, and absolutely nothing is provided for network communication, drivers for external hardware, or access to a filesystem.

FreeRTOS is a real-time kernel (or real-time scheduler) on top of which embedded applications can be built to meet their hard real-time requirements. It allows applications to be organized as a collection of independent threads of execution. On a processor that has only one core, only a single thread can be executing at any one time. The kernel decides which thread should be executing by examining the priority assigned to each thread by the application designer. In the simplest case, the application designer could assign higher priorities to threads

that implement hard real-time requirements, and lower priorities to threads that implement soft real-time requirements. This would ensure that hard real-time threads are always executed ahead of soft real-time threads, but priority assignment decisions are not always that simplistic.

Visit the FreeRTOS home page at https://www.freertos.org for further information about the FreeRTOS,

4 Introduction to ESP-IDF

For the ESP32, a framework has been developed by Espressif called the IoT Development Framework which has become commonly known as "ESP-IDF". It can be found at https://github.com/espressif/esp-idf

Espressif provides basic hardware and software resources to help application developers realize their ideas using the ESP32 series hardware. The software development framework by Espressif is intended for development of Internet-of-Things (IoT) applications with Wi-Fi, Bluetooth, power management and several other system features.

The ESP-IDF is built based on the FreeRTOS platform.

5 Getting started with ESP32

5.1 What You Need

Hardware:

- An ESP32 board
- USB cable - USB A / micro USB B
- Computer running Windows, Linux, or macOS

Software:

- Toolchain to compile code for ESP32
- Build tools - CMake and Ninja to build a full Application for ESP32
- ESP-IDF that essentially contains API (software libraries and source code) for ESP32 and scripts to operate the Toolchain
- Text editor to write programs (Projects) in C, e.g., Eclipse

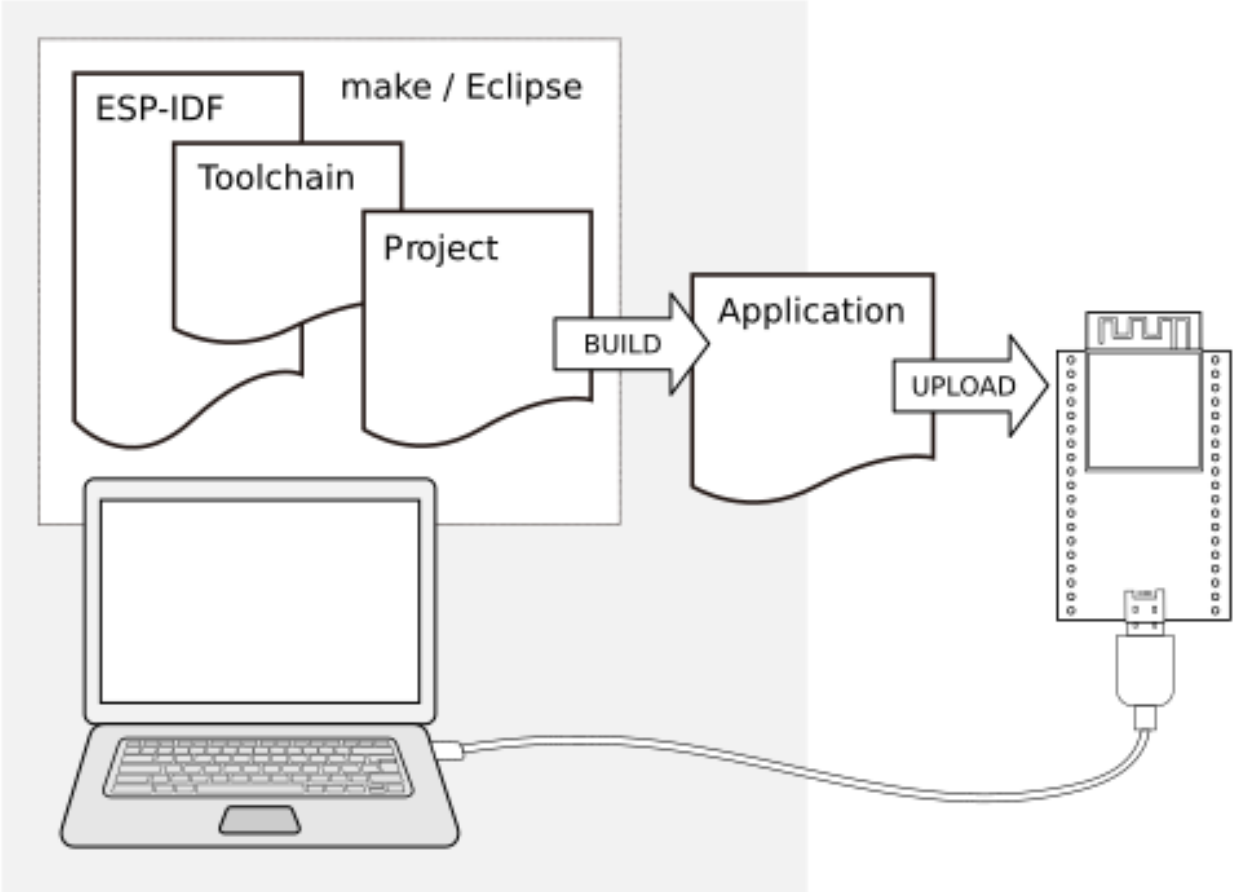

Figure 4: Development of applications for ESP32

5.2 Installation

ESP-IDF requires some prerequisite tools to be installed so you can build firmware for supported chips. The prerequisite tools include Python, Git, cross-compilers, CMake and Ninja build tools.

Step-by-step installation guide can be found in Espressif's documentation on how to get started with ESP-IDF:
https://docs.espressif.com/projects/esp-idf/en/latest/get-started/index.html

6 In-class Requirements

6.1 Build and flash

Students build and flash the "Hello world!" example to the ESP32 development board.

6.2 Check the result

Students can use any serial monitor to check whether the ESP32 development board sends the text: "Hello world!" or not. If everything goes OK, the output should be:

```
...
Hello world!
Restarting in 10 seconds...
I (211) cpu_start: Starting scheduler on APP CPU.
Restarting in 9 seconds...
Restarting in 8 seconds...
Restarting in 7 seconds...
```