Embedded System Programming

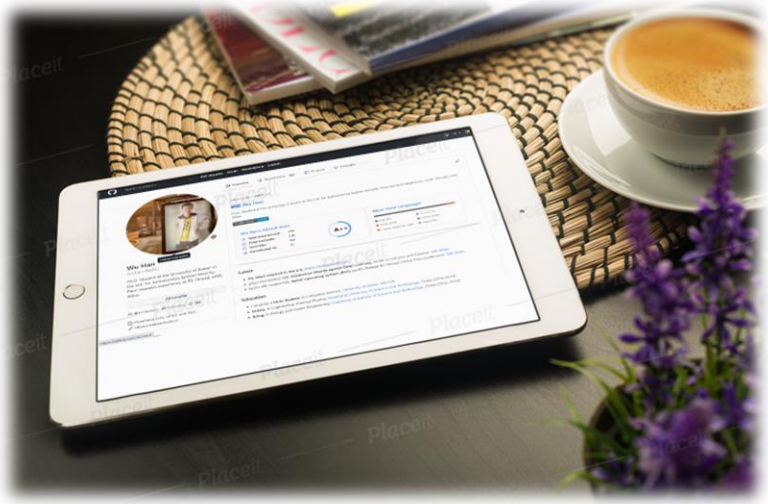
An introduction, Applications and Career Opportunities

Han Wu

Ph.D. Student at the University of Exeter, the U.K.

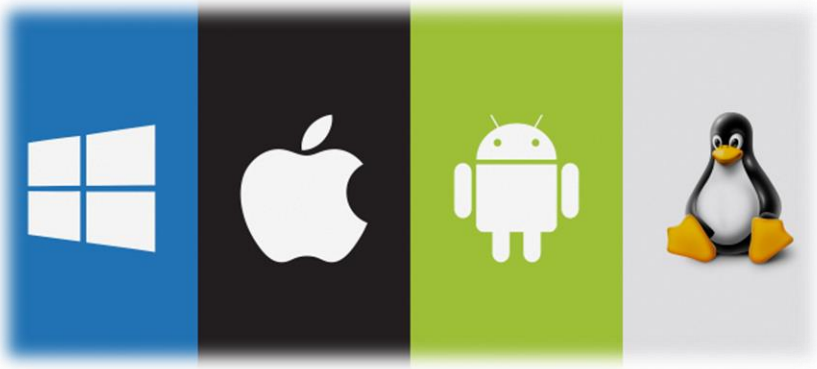
Is Deep Learning secure for Robots?

What is Embedded Programming?

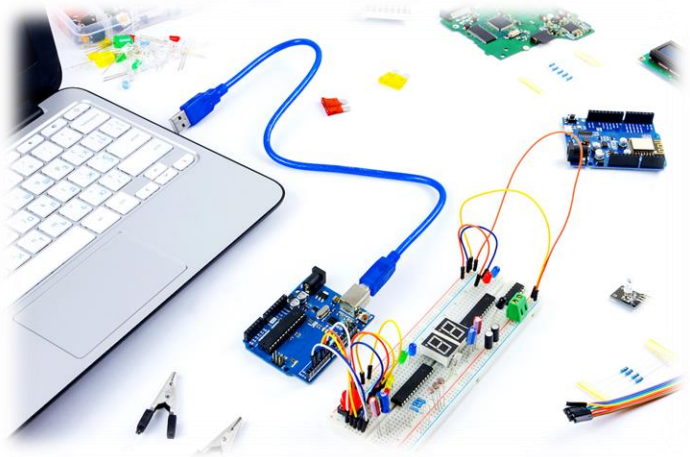


Web Development

App Development



OS Development

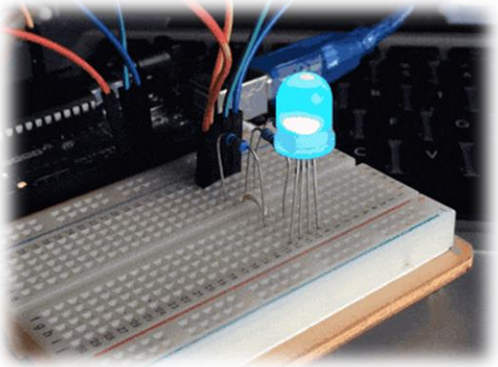


We program **Hardware**

What is Embedded Programming?

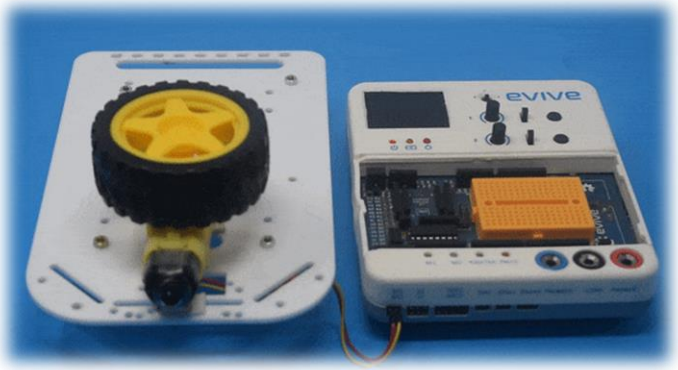
- Achieve a given task set with constrained resources
- Real-time performance

Microcontroller (MCU) & Peripherals

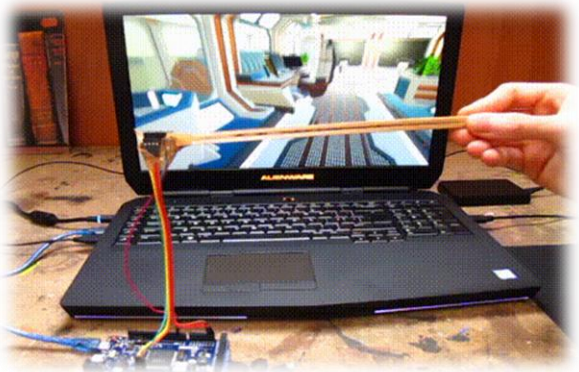


LED

OLED Display



DC Motor

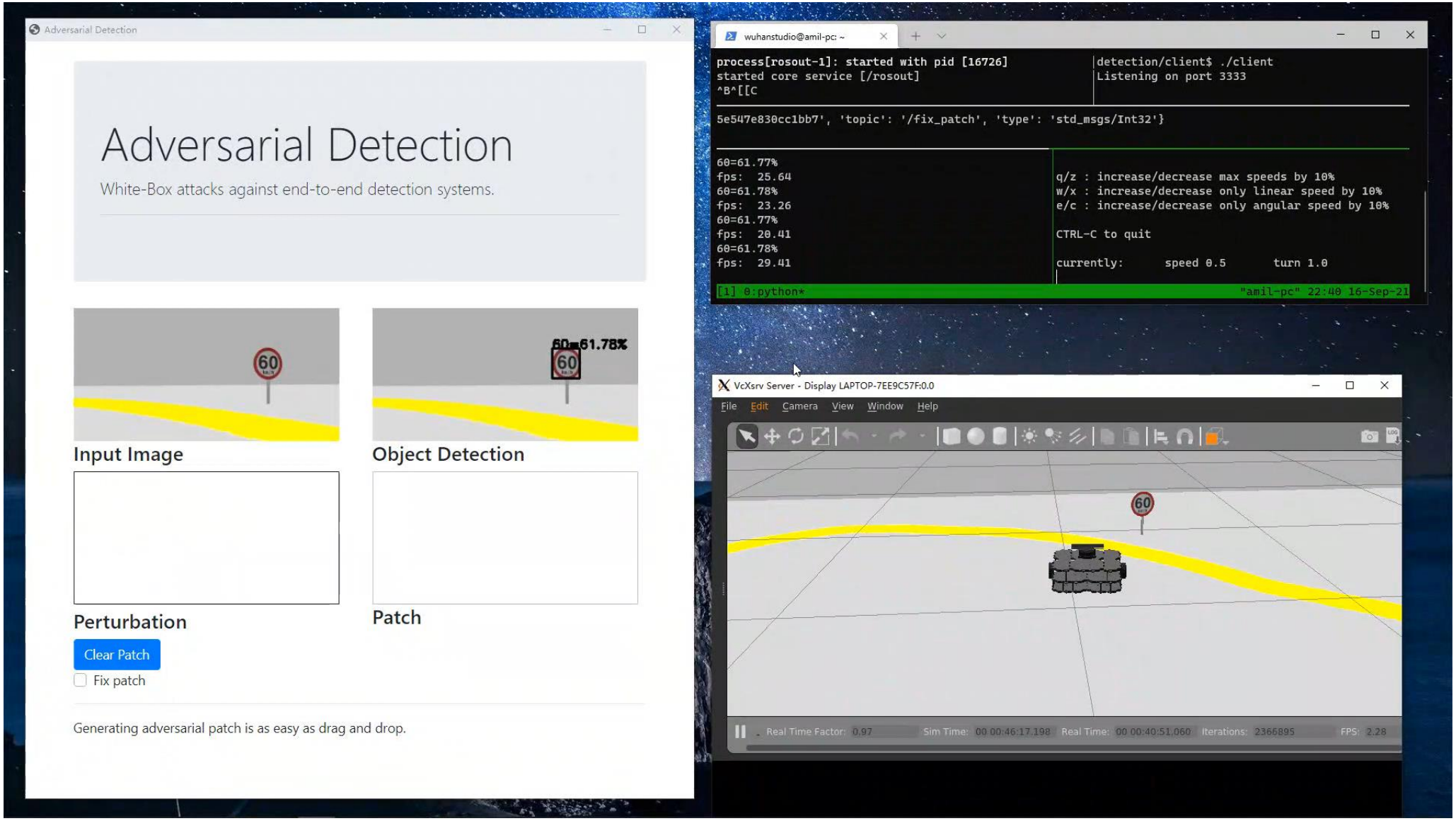


Accelerometer Gyroscope

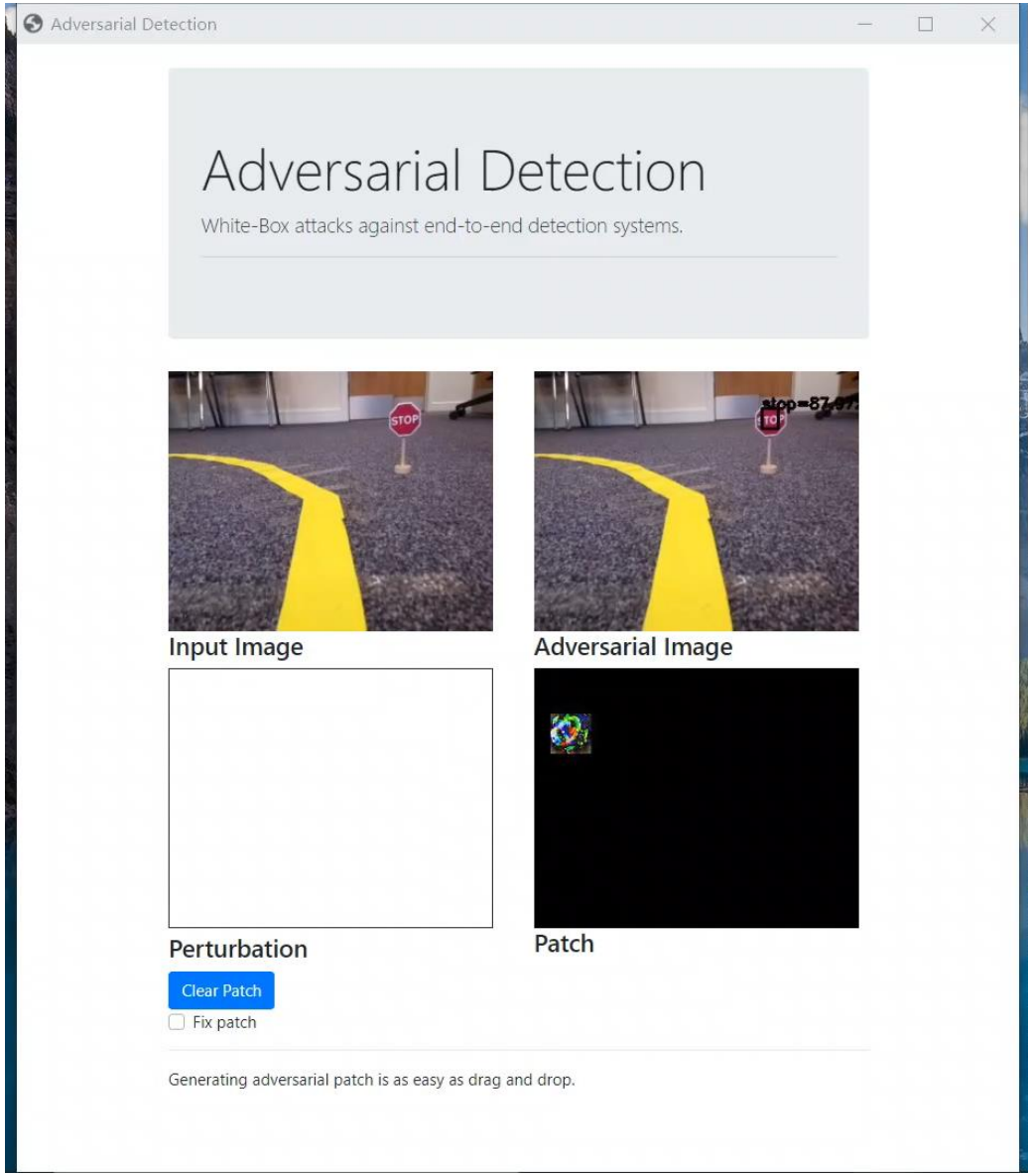
Is Deep Learning Secure for Robots?

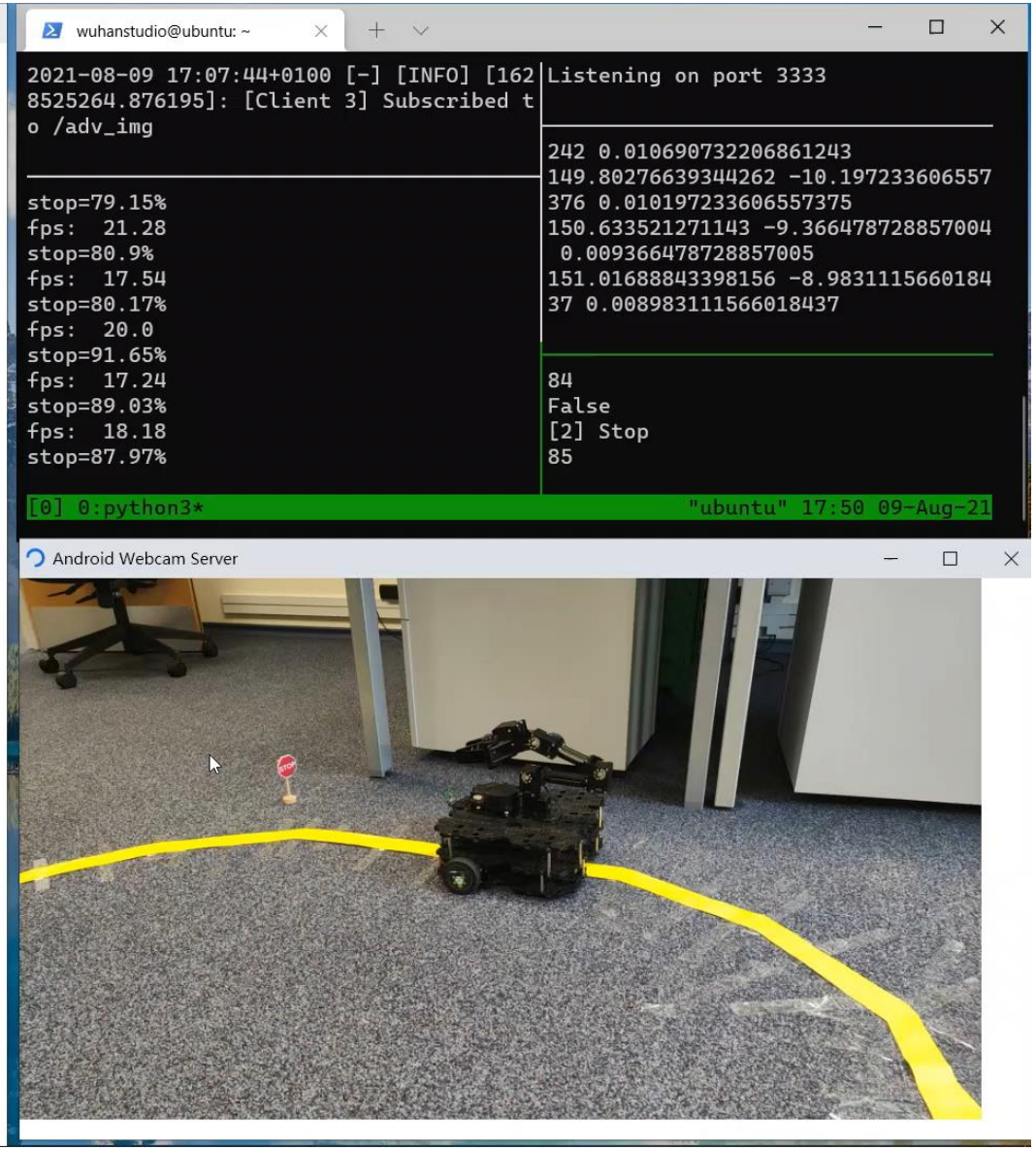
Deep Learning on Embedded Systems

Is Deep Learning Secure for Robots



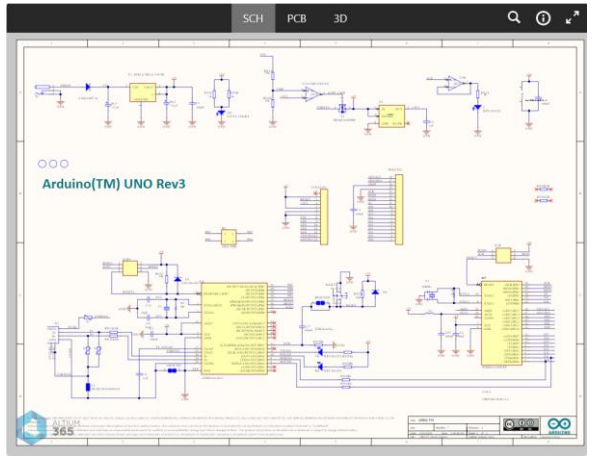
Is Deep Learning Secure for Robots



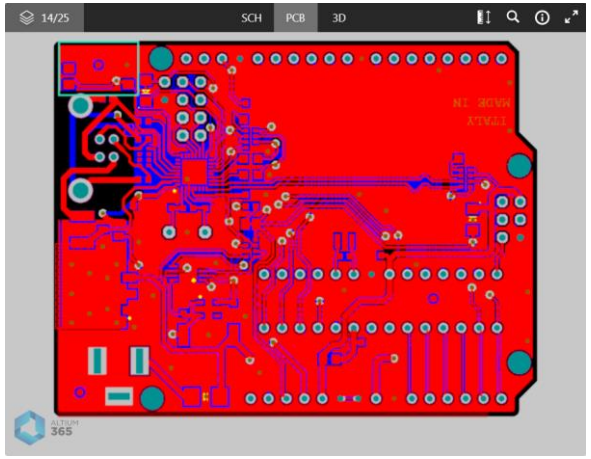


How do we program hardware?

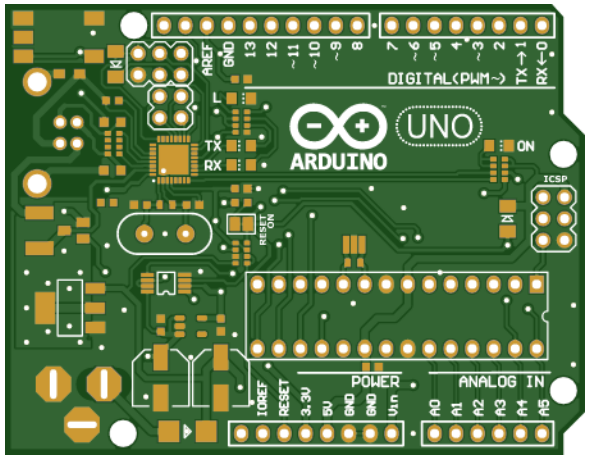
Circuits Design



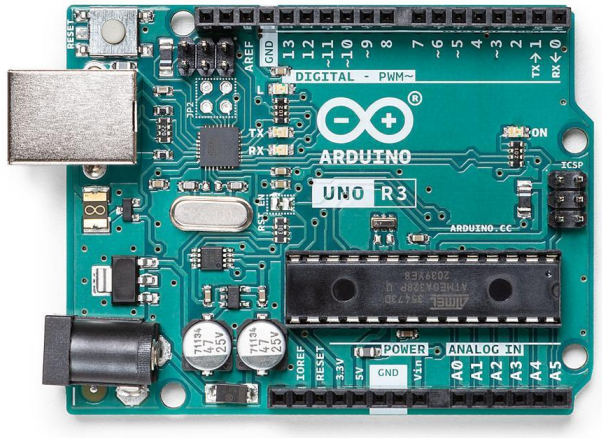
Schematic



PCB Routing

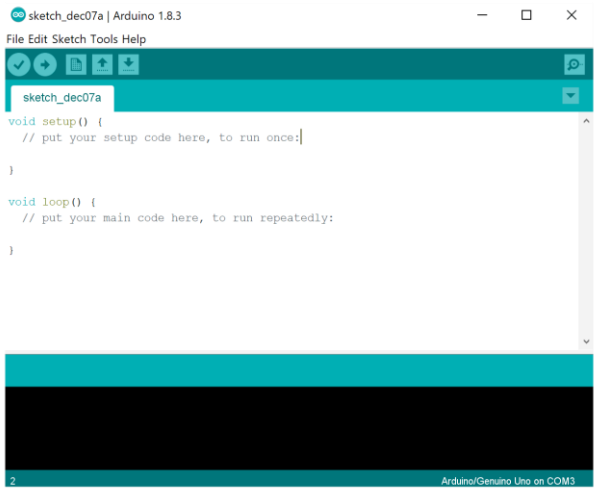


PCB Manufacturing



Development Board

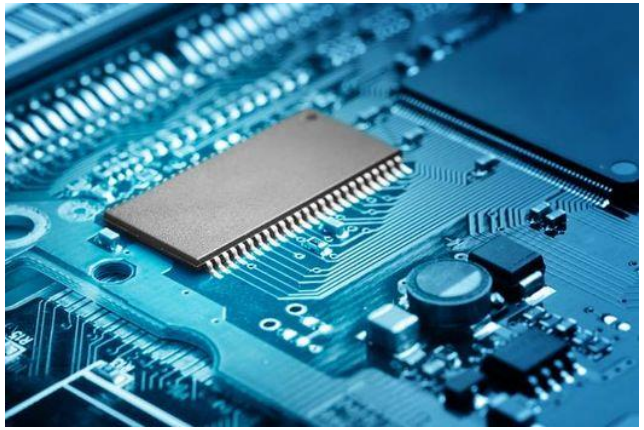
Embedded Programming



① BareMetal

RT-Thread

② RT-Thread RTOS

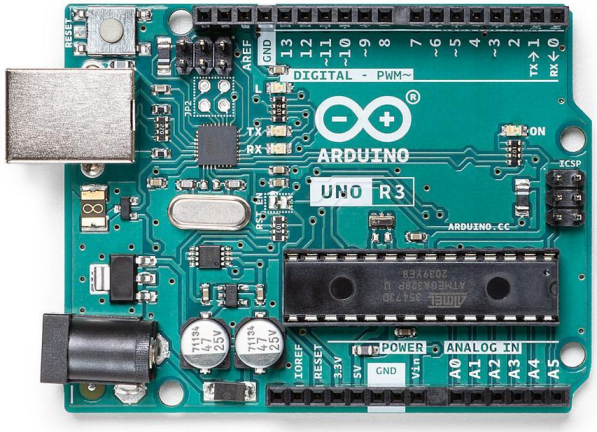


③ Future Career

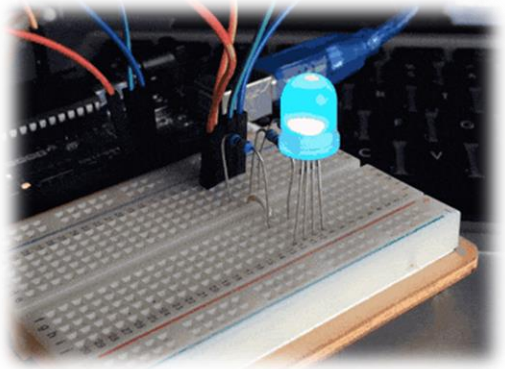
BareMetal Programming

How do we program hardware ?

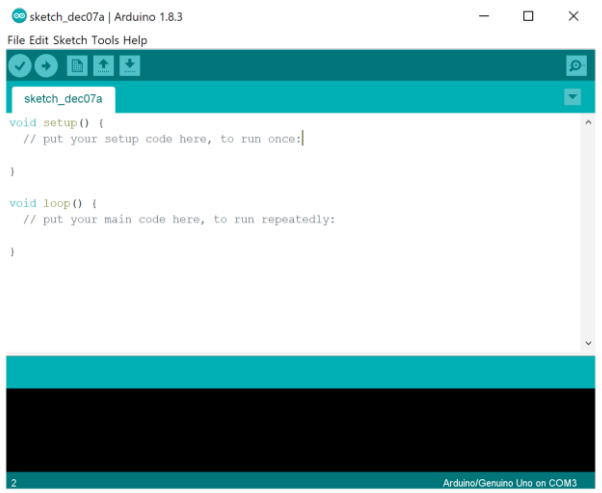
BareMetal Programming



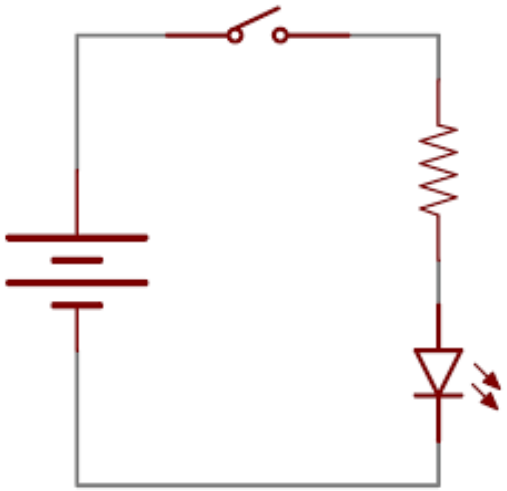
MCU Development Board

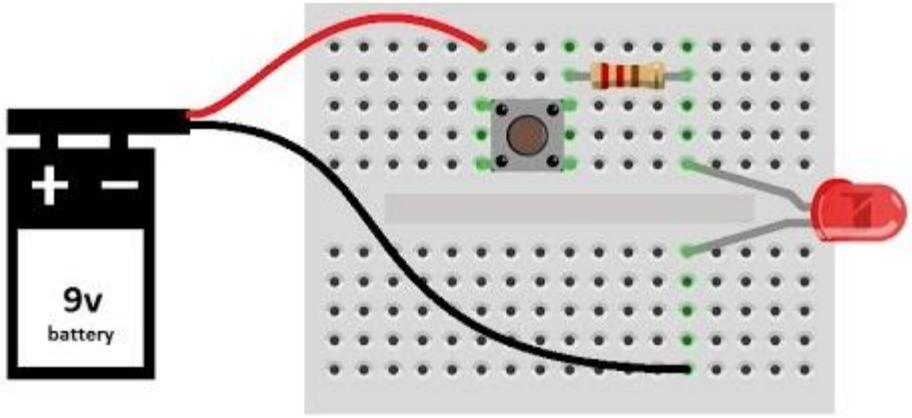


LED 101

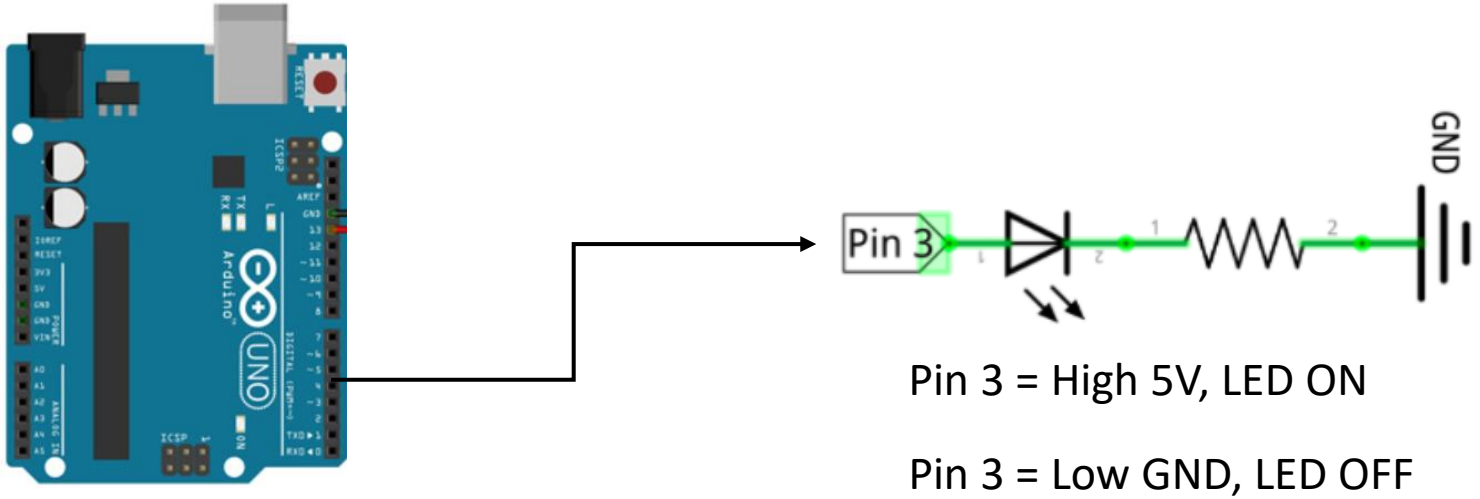


How to write a C program to control the LED?





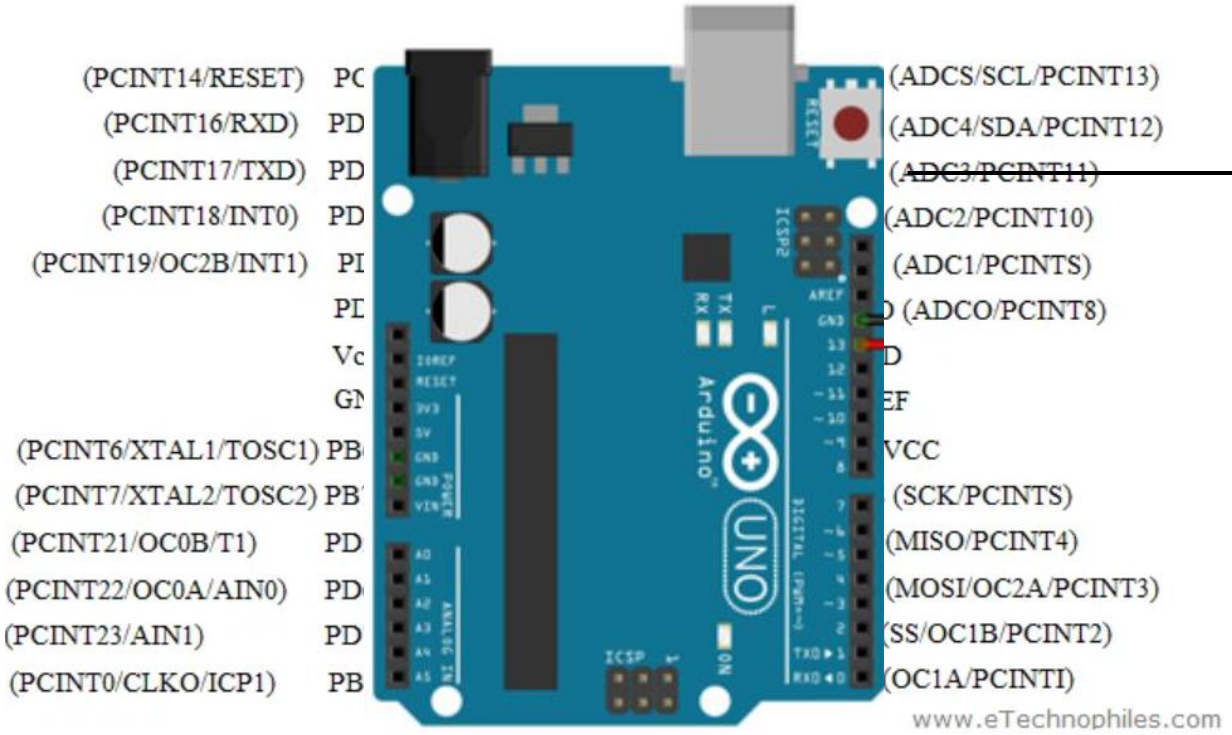
Control the LED using a Switch / Button

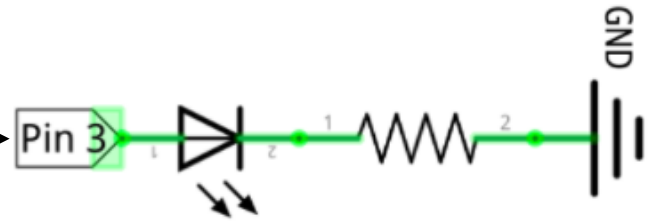


Pin 3 = High 5V, LED ON

Pin 3 = Low GND, LED OFF

BareMetal Programming





Pin 3 = High 5V, LED ON

Pin 3 = Low GND, LED OFF

[ATmega328p Datasheet]

13.4.8 PORTD – The Port D Data Register

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

C Language (SDK)

digitalWrite(3, HIGH); // LED ON

digitalWrite(3, LOW); // LED OFF

Assembly Language

SBI PORTD, PIND3 ; 1, HIGH, LED ON

CBI PORTD, PIND3 ; 0, LOW, LED OFF

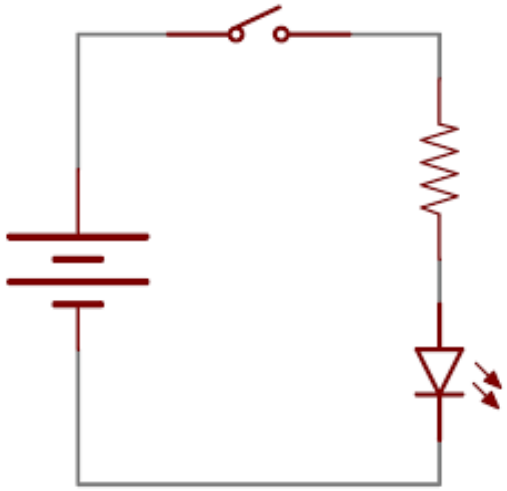
Register (Configurations in MCU)

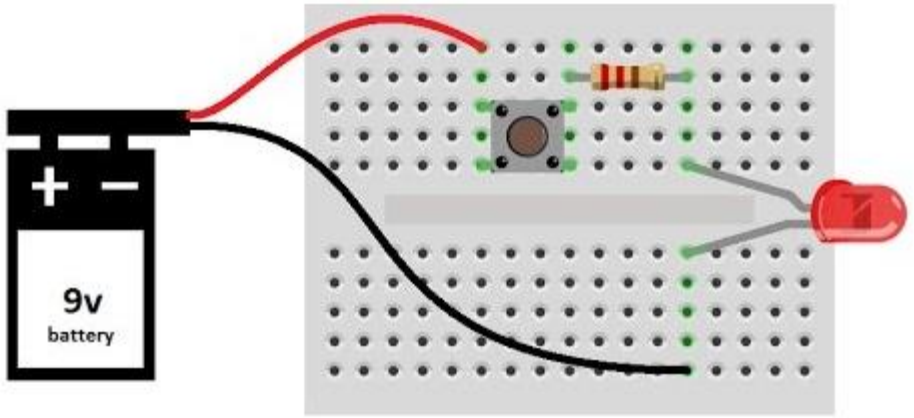
PORTD3 = 1, HIGH, LED ON

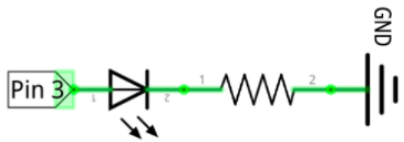
PORTD3 = 0, GND, LED OFF

LED 101 Demo

BareMetal Programming



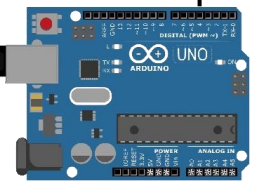




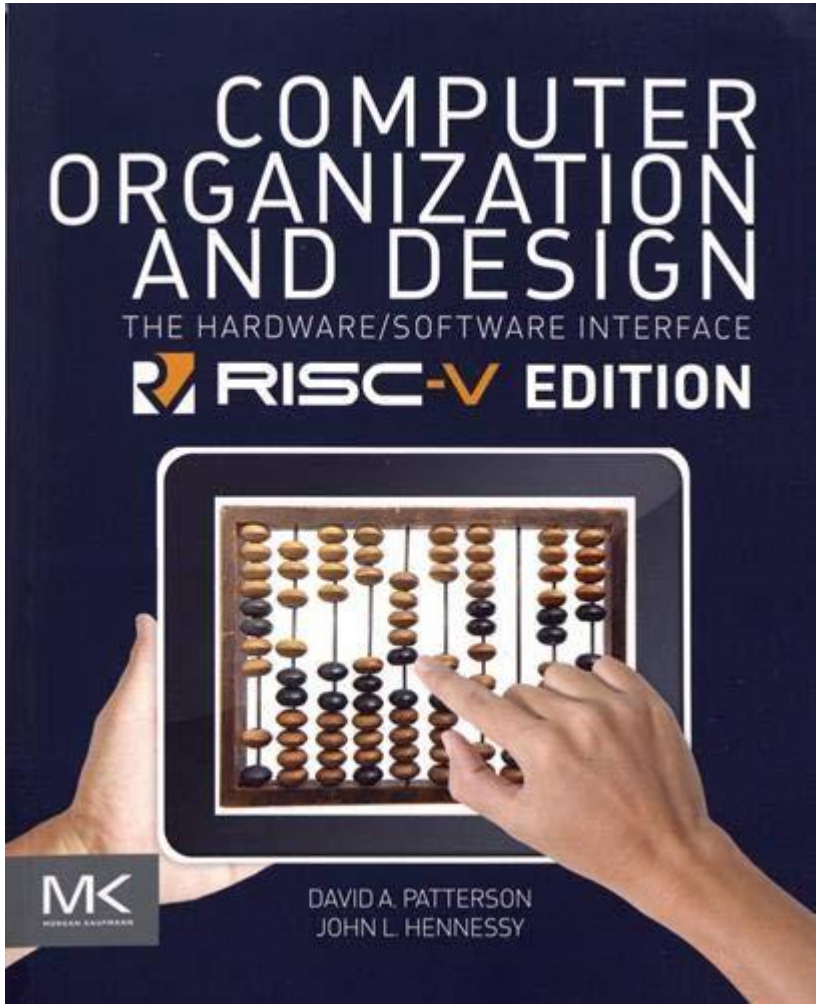
Pin 3 = High 5V, LED ON

Pin 3 = Low GND, LED OFF

Write a C program



Upload the program to MCU



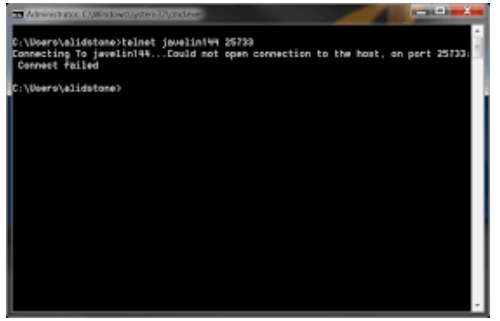
Introduction to RT-Thread (RTOS)

From BareMetal to Real-Time Operating System

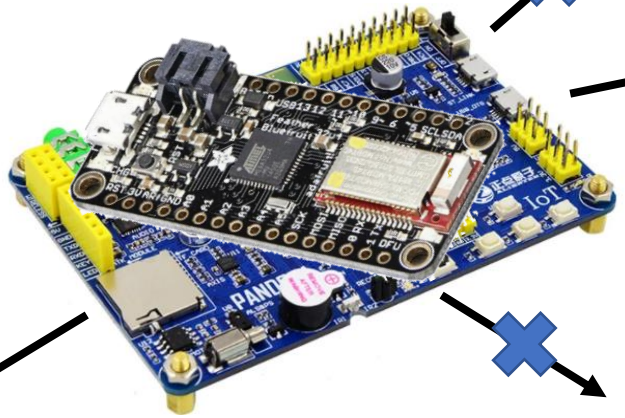
Why do we need RTOS?

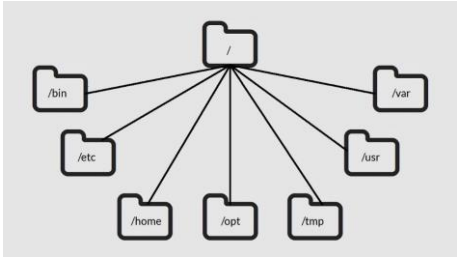
Web Server

Wireless Network

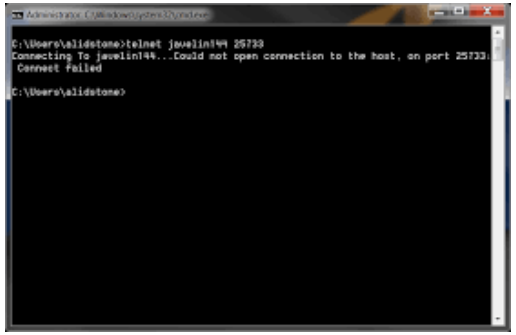


Telnet Server

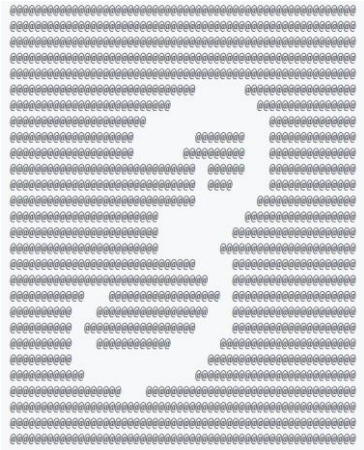




File System



Serial Terminal



Deep Learning

Why do we need RTOS?

Operating System manages things for you

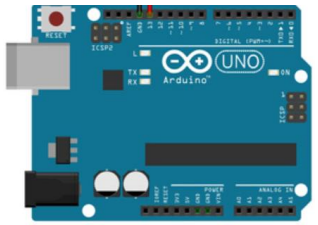
Your Application

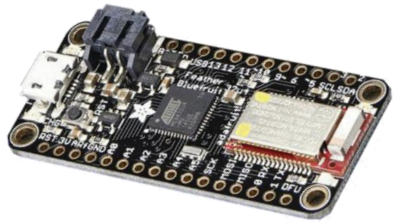
Real-Time Operating System (RTOS)

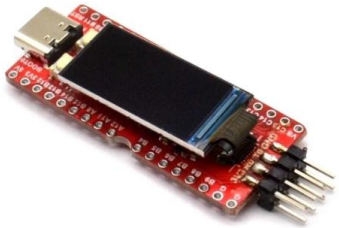
File System | Networking | POSIX | Shell

Hardware Abstraction Layer (HAL)

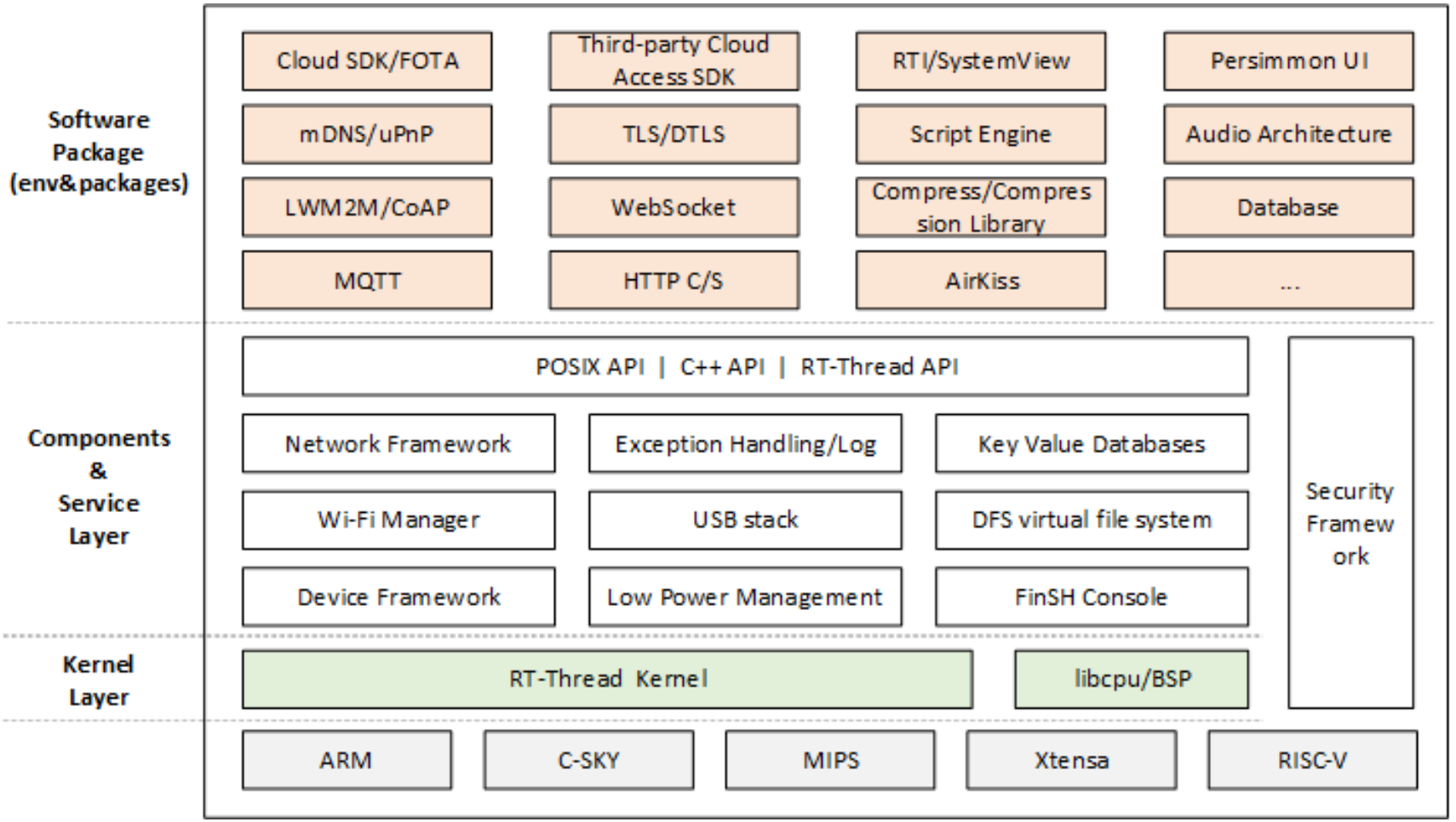
AVR | STM32 | ESP32 | RISC-V







Open Source RTOS



FreeRTOS

RT-Thread

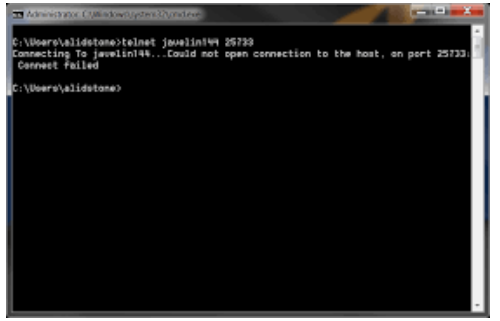
Zephyr

Nuttx

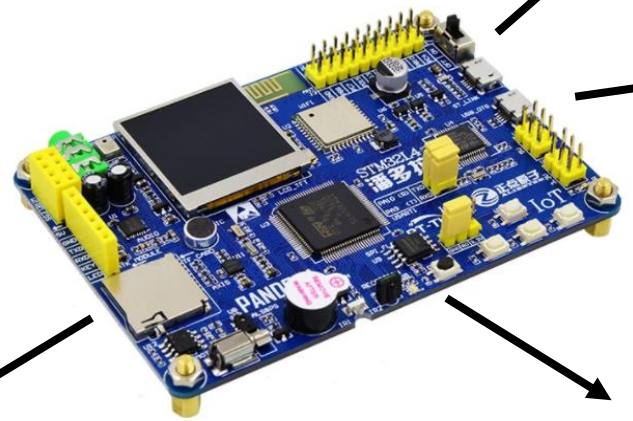
Why do we need RTOS?

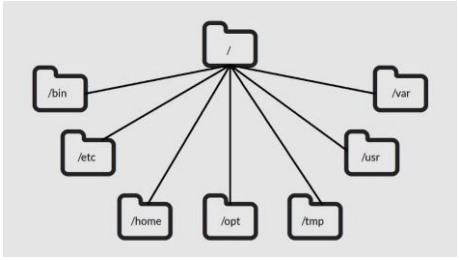
Software Packages
(Web Server)

Networking

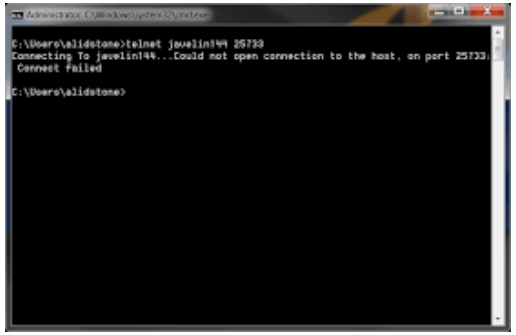


Software Packages
(Telnet)

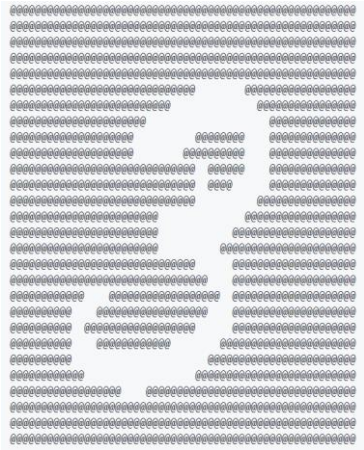




File System

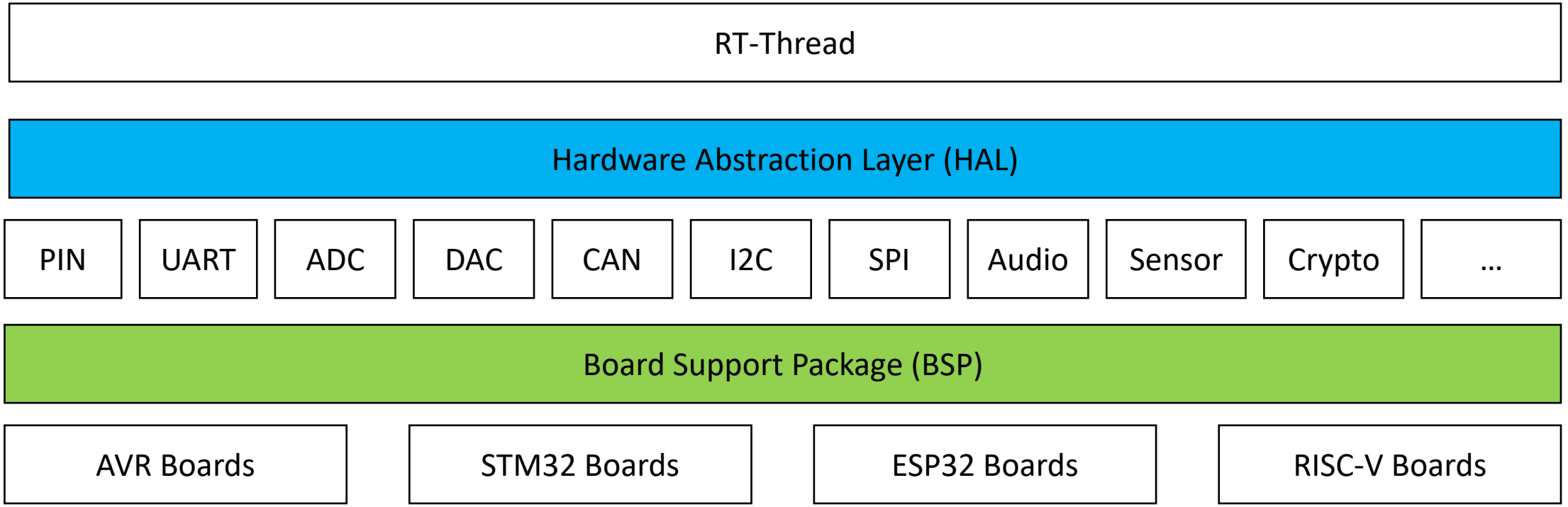


HAL (UART, I2C ...)



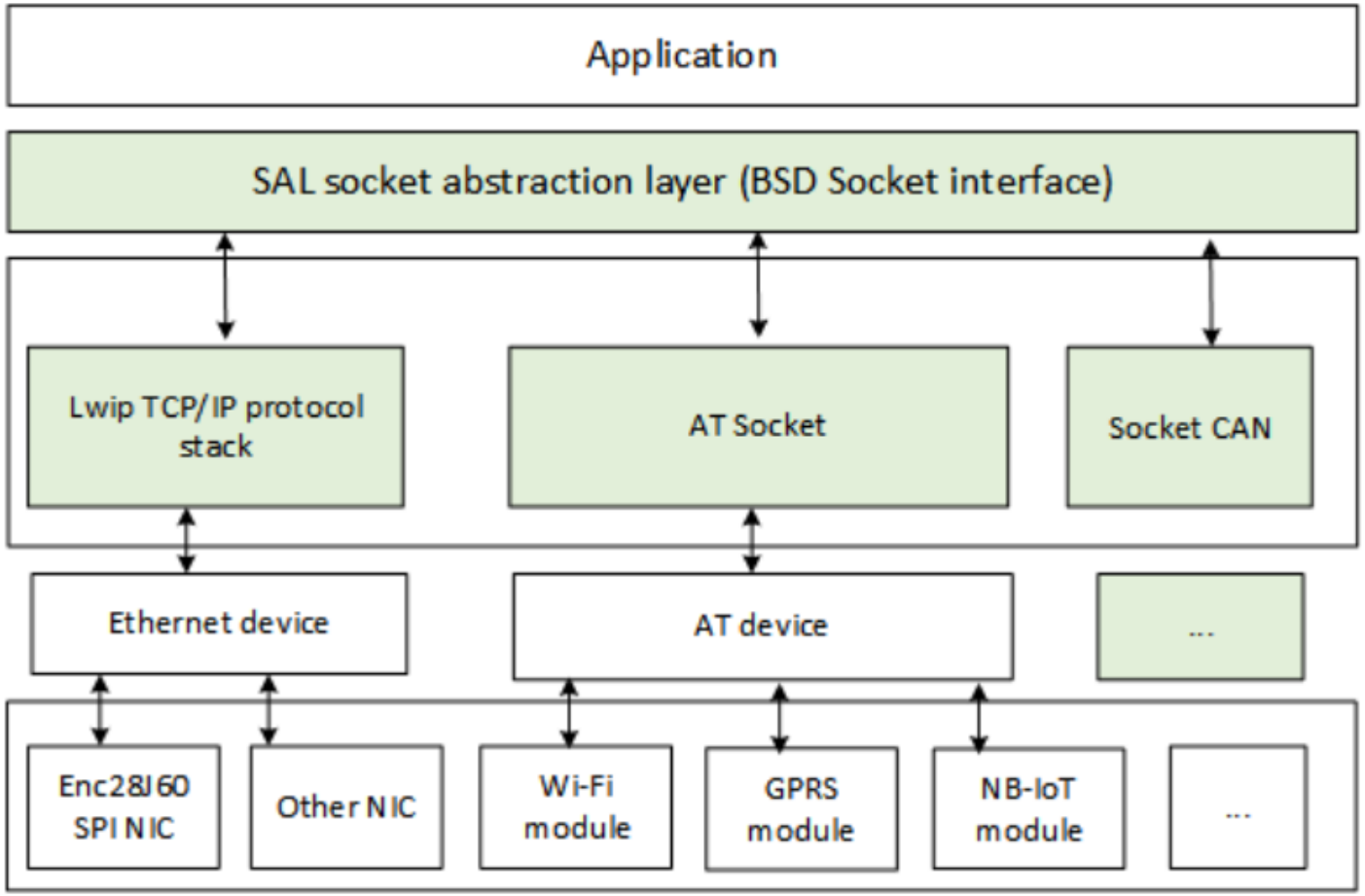
POSIX Layer

RT-Thread HAL



The same code runs on different hardware

RT-Thread Networking



MSH Command

> list_device

> wifi scan

> wifi status

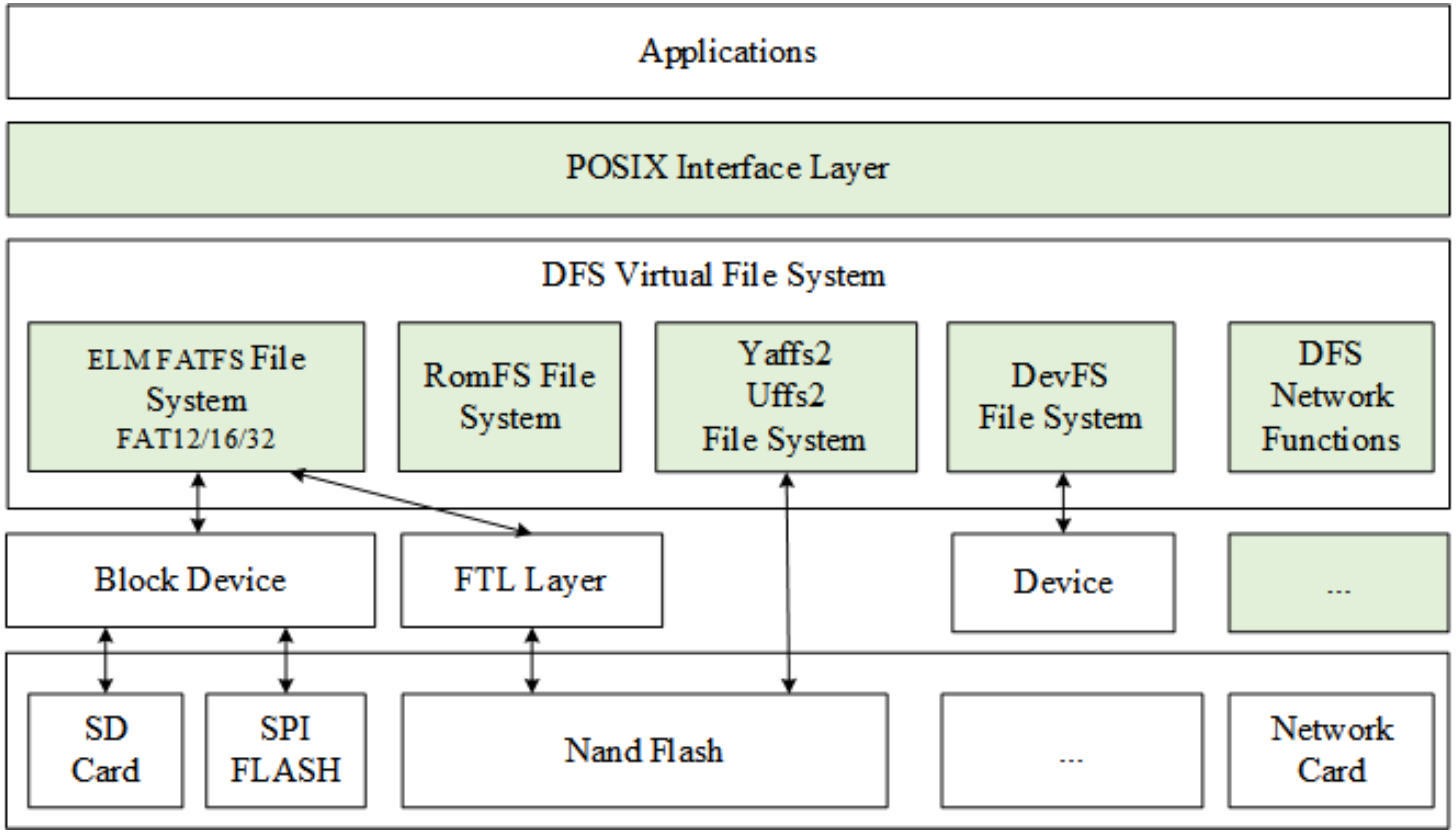
> Ifconfig

> dns

> ping

> telnet

RT-Thread File System



MSH Command

```
> list_device
> pwd
> ls
> mkdir
> cd
> rm
> cat
> echo
```

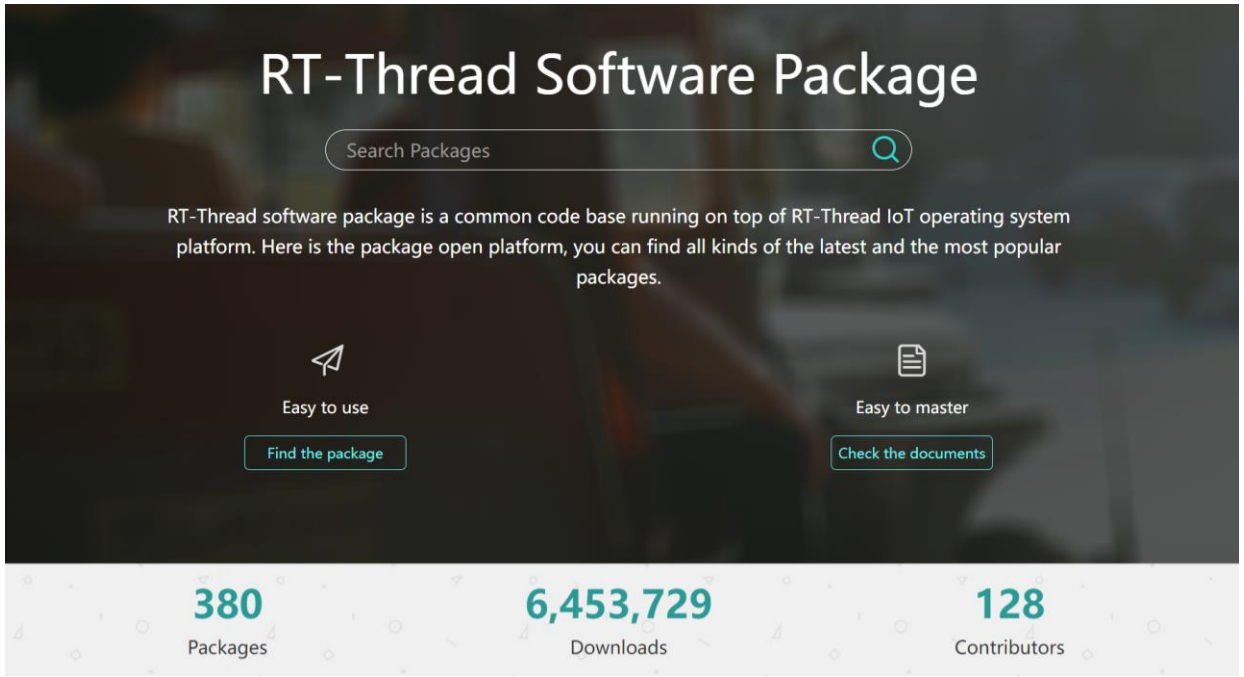
RT-Thread POSIX

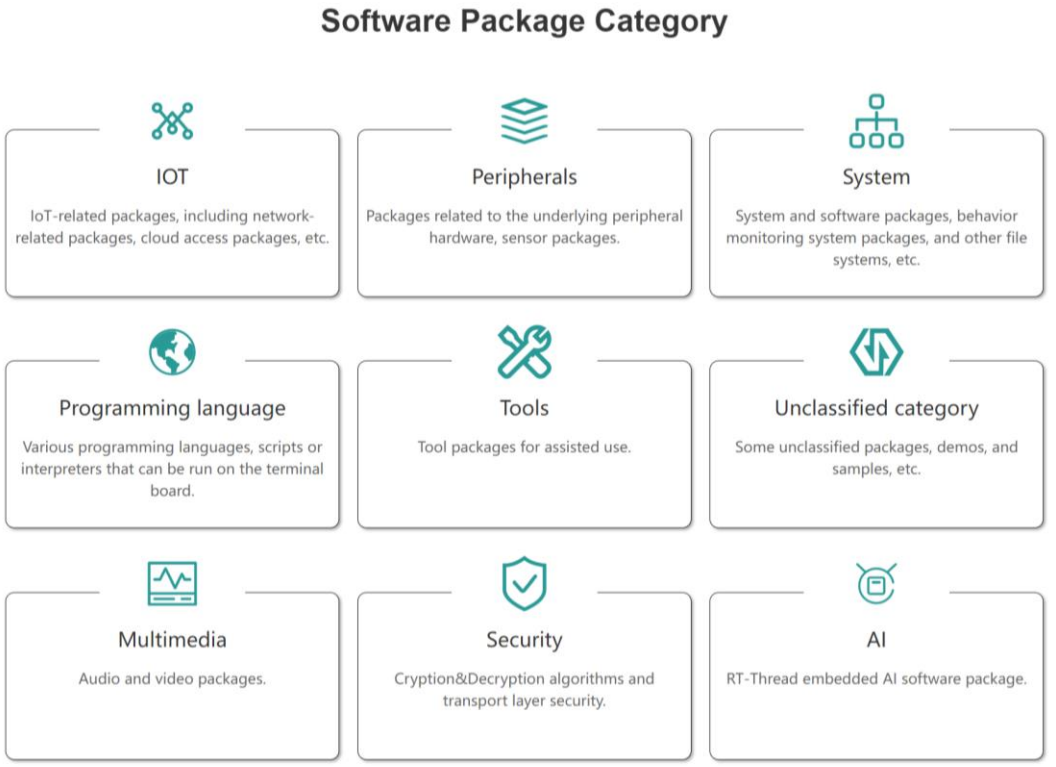
- POSIX = **Portable Operating System**
  - Standard Enforced by IEEE
  - The purpose of POSIX was to **improve portability**. When your source code follows the standard, you can compile and **run the code on a different machine easily**.

- **Thread management**: Thread creating, detaching, joining, and setting and querying thread attributes.
- **Mutex**: Abbreviation for "mutual exclusion", which restricts thread access to shared data and protects the integrity of shared data. This includes creating, destroying, locking, and unlocking mutex and some functions for setting or modifying mutex properties.
- **Condition variable**: Communication between threads used to share a mutex. It includes functions such as creation, destruction, waiting condition variables, and sending signal.
- **Read/write locks and barriers**: including the creation, destruction, wait, and related property settings of read-write locks and barriers.

The same code runs on different Operating Systems

RT-Thread Packages





Future Career

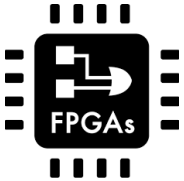
Embedded System Developer

Embedded Linux

Rust Lang

Security

FPGA



From BareMetal to RTOS

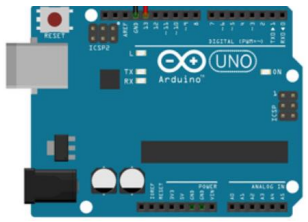
FreeRTOS

RT-Thread

Zephyr

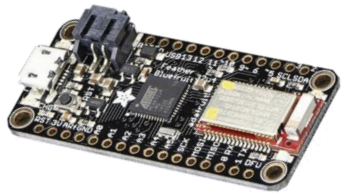
Nuttx

BareMetal Programming

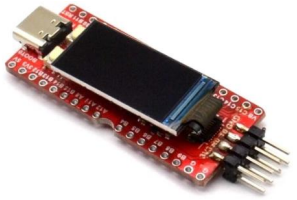


Arduino

STM32



ESP32



RISC-V

Thanks

@wuhanstudio

https://wuhanstudio.cc