contiki os developed by 2003

what is contiki os:-

contiki os is an open-source, lightweight operating system design resource-constrained embedded system.

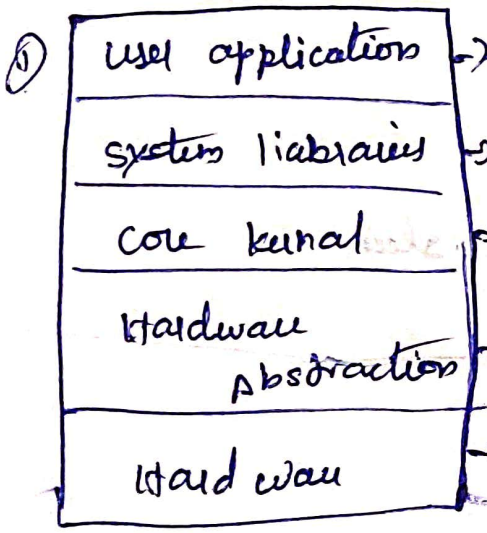
→ especially (IOT) internet of things

- ↳ wireless sesore node
- ↳ low power embedded device
- ↳ small device

you can run IPv6 and COAP on a node with 10kB ram

structure of contiki os

Ex



- user application → sensore app, COAP server, IPv6 stack
- system libraries → (TCP/IP)
- core kernal → event-driven kernal + protothread
- Hardware Abstraction → drivers, radio, timers.
- Hardware → MCU, radio transceiver, sensors, Peripheral.

repositories:-

OS → kernal, libraries, network stack

Platform → Board support file (MCU startup, drivers)

cpu → cpu specific code (timers)

example → sample application.

tool :- Build tools simulators (like cooja).

Working flow:

① initialization
- Hardware driver initialize.
- kernal set up event loop

② Process system (Process thread)

③ Event handling (Process wait event)

④ Networking
- Network Stack (IPV4/V6)
- Application use the stack API

⑤ Powa Management
- trigger MCU sleep
- Radio. duty cycle

⑥ loop
- handle event, Process data, sleep

Porting contiki OS File & step

① implement the CPU architecture:
- CPU/Nordic NRF 52840-dk
- startup code (startup.c (or) startup.s)
- clock driven : clock.c
- interrupt handling

② Add a platform folder
- Platform NRF5984O
- pin MAP (LED, uart, SPI, I2C
- Peripheral drivers.
- contiki-conf.h , Platform config.

③ write Radio driver

→ dev/radio/your device.c

↳ interface between contiki and the radio transceiver (Ex, CC2420, nRF24L01)

④ Build system

→ update Make file

→ Tool chain flag

→ linker script

→ Board definition

⑤ Test the board

②

if you want tiny code cork for a 16KB Ram device with simple event system and proven 6LoWPAN stack contiki is still Great

---

① Proto thread.

② cooja : simulates every instruction

↳ its not a simulator, its cycle-accurate emulator for MCU

③ Time Travelling debug.

↳ you can pause the simulation

④ Radio Duty cycling protocol

⑤ The rTimer vs etimer

etimer : coarse-grained, driven by the event loop (low precision)

rtime → High precision hardware interrupt → real time scheduling.

⑥ Dynamic code loading :-

⑦ Stack Pluggability

↳ contiki swap networking stack

↳ you can run uIP (classic IPV4/V6 / TCP/UDP) for internet style routing.

⑧ Tiny ~~foot~~ print but real time internet

↳ Full IPV6 over 802.15.4 using 6LOWPAN

↳ RPL (IPV6 Routing Protocol for low power and lossy networks)

↳ coap for Rest full IOT APIs.

⑨ cooja Mote type

↳ Emulated → MSP 430 → instruction level

↳ Native → Run as your code in linux.

↳ SKY, Z1, CC2650 → different MCU configuration

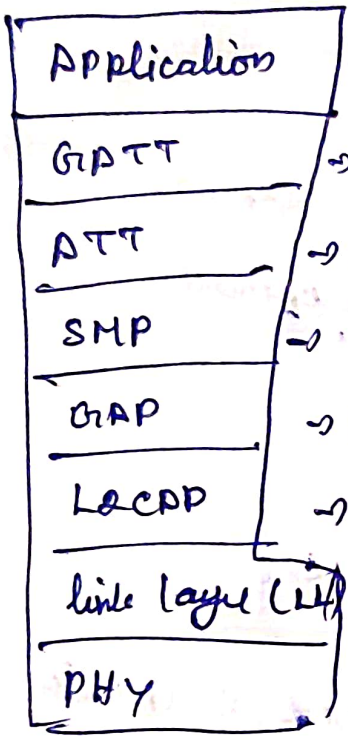
BLE

BLE Stack

A BLE stack (Bluetooth low Energy stack) is the complete software implementation of ble.

↳ From the radio layer and up to application layer.

core layer of BLE Stack



→ your app, data, notification

→ Generic attribute profile - how data is organized

→ Attribute protocol - (read/write)

→ Security Manager → pair, bond, encryption

→ Generic access profile → Advertising, connection

→ logical link control & Adaptation protocol

→ connection, Advertising, channel hopping

→ Physical Radio - actual Radio transmit)

Type of Stacks

Ⓐ Bluetooth Stack

① classic bluetooth stack (BR/EDR)

↳ For audio/ File transfer, legacy BT

↳ Big, support profile like A2DP, HFP, SPP

② Bluetooth low Energy (BLE) Stack

↳ For low-power device (sensors, wearable)

↳ Focuses on GAP, Gatt, ATT, SMP layer

Example

- Nordic Softdevice (Proprietary BLE Stack)
- Zephyr BLE Host & Controller (Open source)
- BlueZ (Linux Bluetooth stack)

(B) wifi Stack

- implement 802.11 MAC + PHY + higher layer like WPA2/WPA3
- provide TCP/IP on top

Example

→ Espressif ESP-IDF wifi stack (for ESP32)
→ Zephyr wifi subsystem (under development)
→ linux wifi stack in the kernal

(C) TCP/IP Stack

→ For internet connectivity.
→ Handle IPV4/IPV6 ITCP, UDP, DHCP, DNS, HTTP, etc.

Example :-

- uIP : Tiny TCP/IP stack (contiki OS)
- lwIP : Light weight IP.
- Nucleus NET, FreeRTOS+TCP

(D) 6LOWPAN Stack

- Adaptation layer for running IPV6 over low-power, lossy link like 802.15.4.
- used for Mesh Network.

Example

- Contiki 6LOWPAN Stack
- Zephyr 6LOWPAN subsystem
- Linux wpan tool for border router.

(E) Zigbee Stack

- For wireless Mesh Network
- Handle zigbee Profile, cluster, security and Mesh routing

Example

- Silicon lab Ember2Net
- TI Z-Stack
- Open zigbee implementation.

(F) Thread Stack

- IP-based Mesh standard for home IOT
- use 6Lowpan + IPV6 + COAP

Example :-

- Open thread (by Google)
- Nordic thread stack
- Zephyr has integration with Open Thread

(4) Security stack

→ Some protocol have a separate security layer

→ BLE uses SMP; wifi uses WPA2/3.

Example

→ MbedTLS is zephyr or lwIP

→ wolfSSL

→ TinyDTLS for constrained devices

Protocol	vendor stack	open source
BLE	Nordic soft Device	zephyr BLE, Nimble
Zigbee	TI-Z stack, Embuznet	zigbeeMQTT, open zigbee lib
Thread	Nordic thread	open Thread
wifi	ESP-IDE wifi	zephyr wifi, linux
TCP/IP	lwIP, uIP	zephyr Net, NUclues NET

Contiki is

CMake list and, SDK and, cooja,
↓
simulation

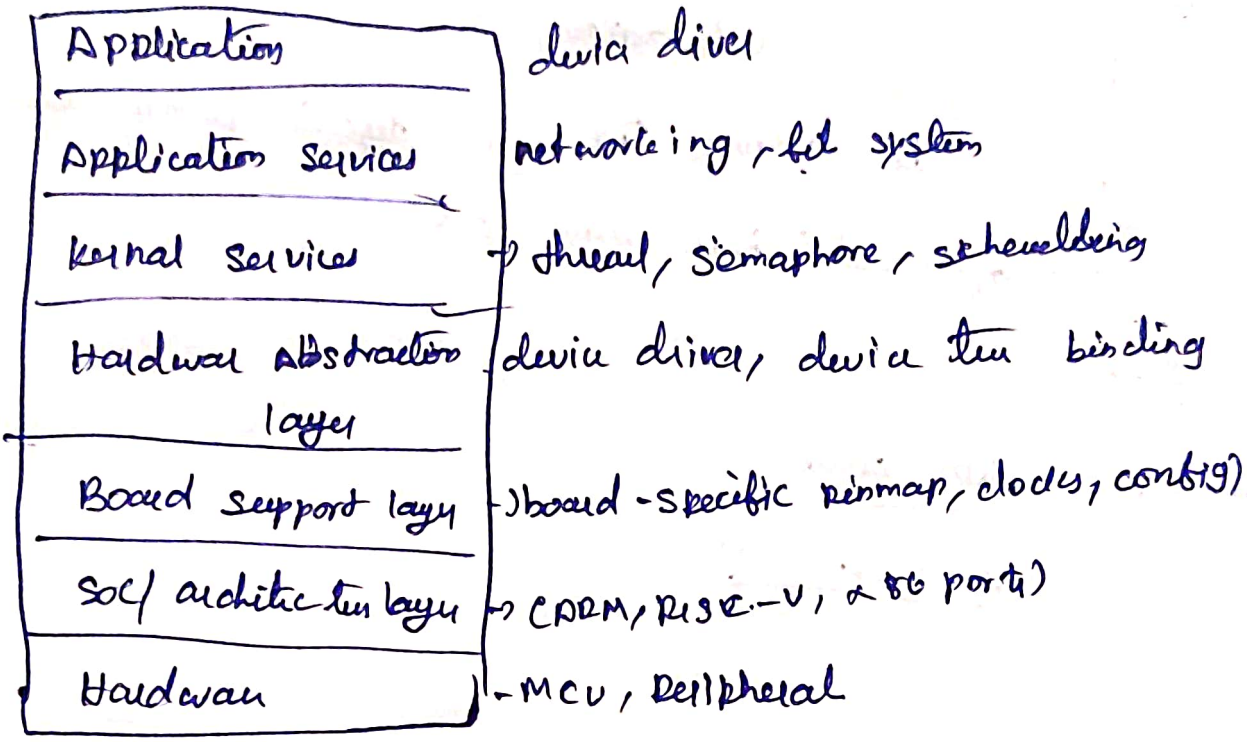
nrfProg method
↓
Hashing tool

Zephyr

CMake, West, dts, kconfig method
↓ ↓ ↓
build, flash tool devicetree kernal configuration

Zephyr

Zephyr RTOS architecture



key parts

- → Kernal
- → subsystem
- → Driver + Hal
- → BSP (Board support package) → pinmux, clocks, memory map
- → Build system → CMake + west
- → security → secure boot.

Porting Zephyr

① SOC port

↳ implement arch layer for CPU (eg: ARM cortex-M, RISC-V) context switch, ISR, atomic ops

② Board Port BSP

→ Add a new folder board/your board/

Provide

board.yaml : Meta data

board.dts : device tree source describes pinmaps, Peripheral

Kconfig.board : build time config

③ Driver

④ Test with board

use west child