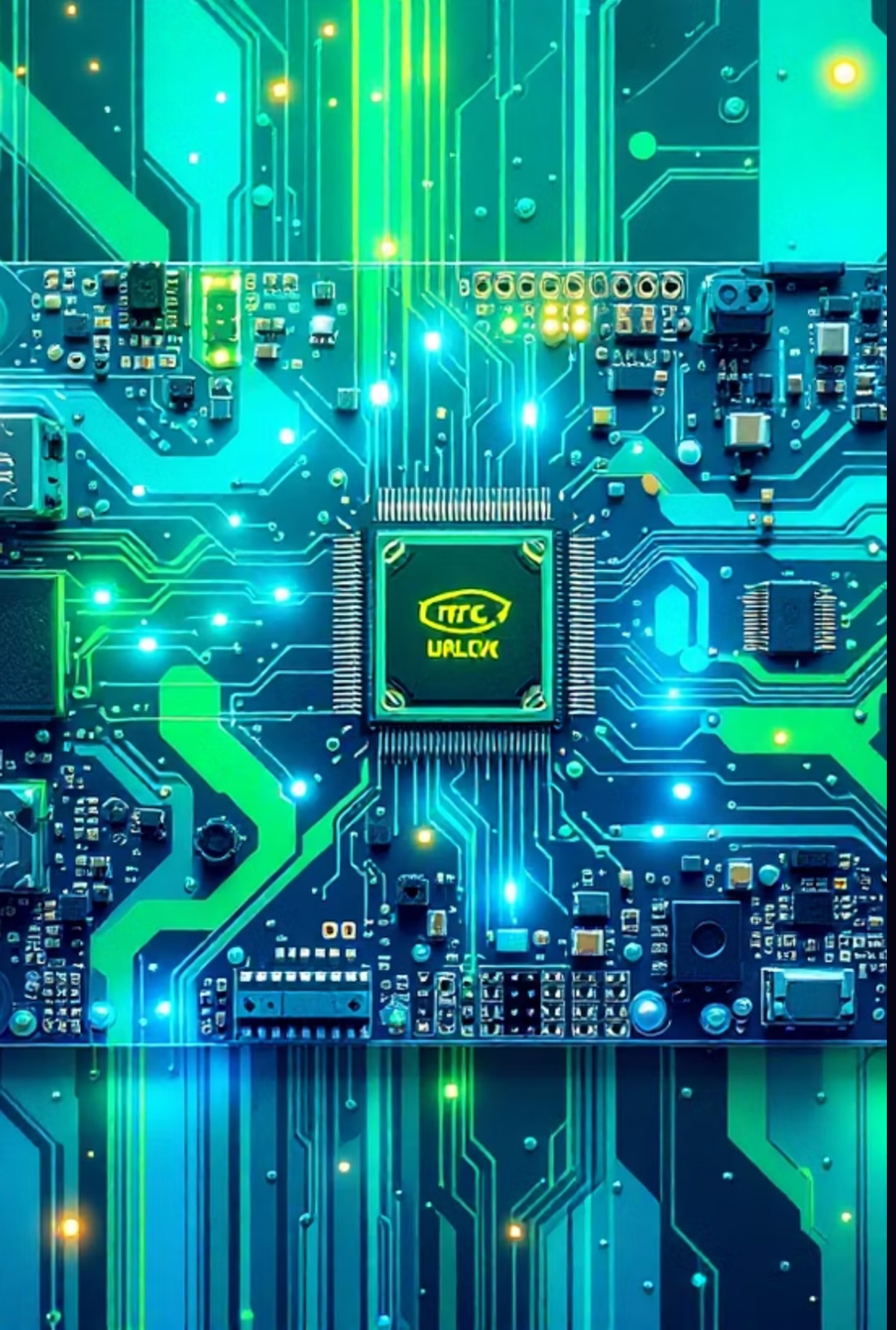


Zephyr Workshop – Kconfig + DeviceTree

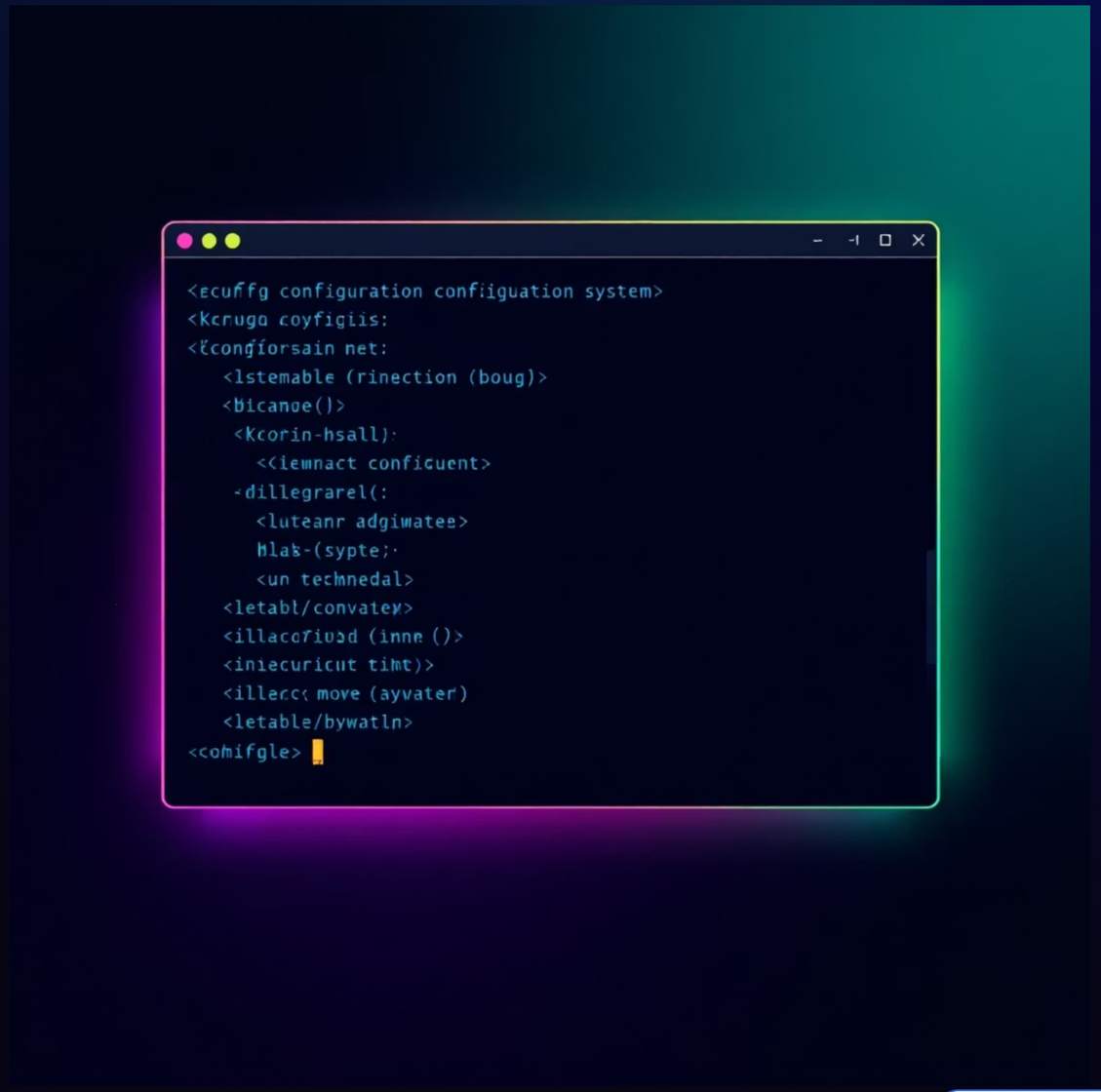
Understanding how Zephyr RTOS configures features and hardware

What is Kconfig?

Configuration system for:

- System features
- Drivers
- RTOS options

Options start with 'CONFIG_'



Kconfig – How it Works

prj.conf

CONFIG_PRINTK=y

CMake + Kconfig

Build system processing

Header Files

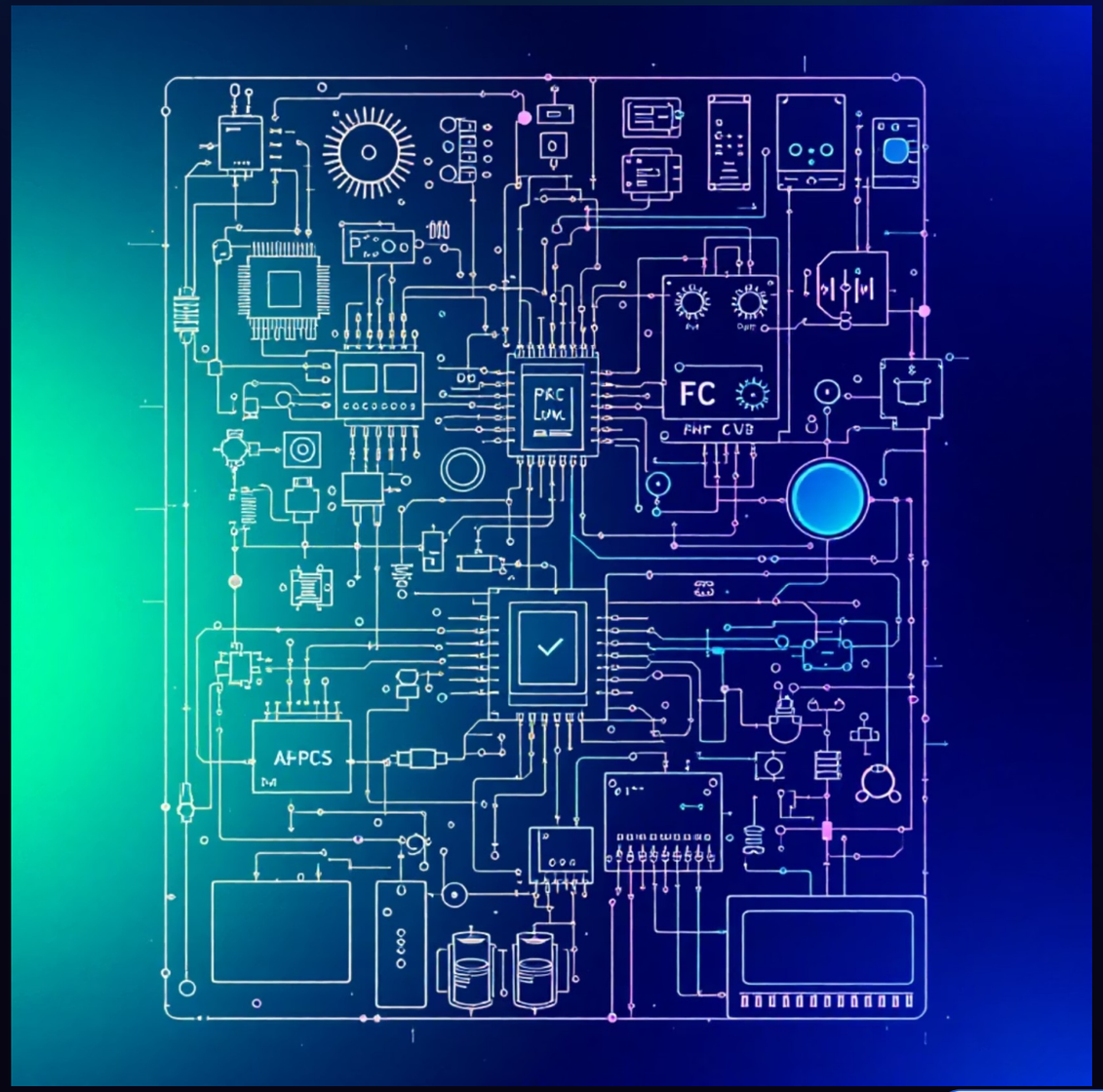
Generated configuration

Use 'west build -t menuconfig' to explore

What is DeviceTree?

Hardware description language that:

- Describes hardware layout and connections
- Used by Zephyr at build time to generate hardware abstraction code.
- Files: *.dts (base), *.overlay (app-level override)



DeviceTree

Nodes

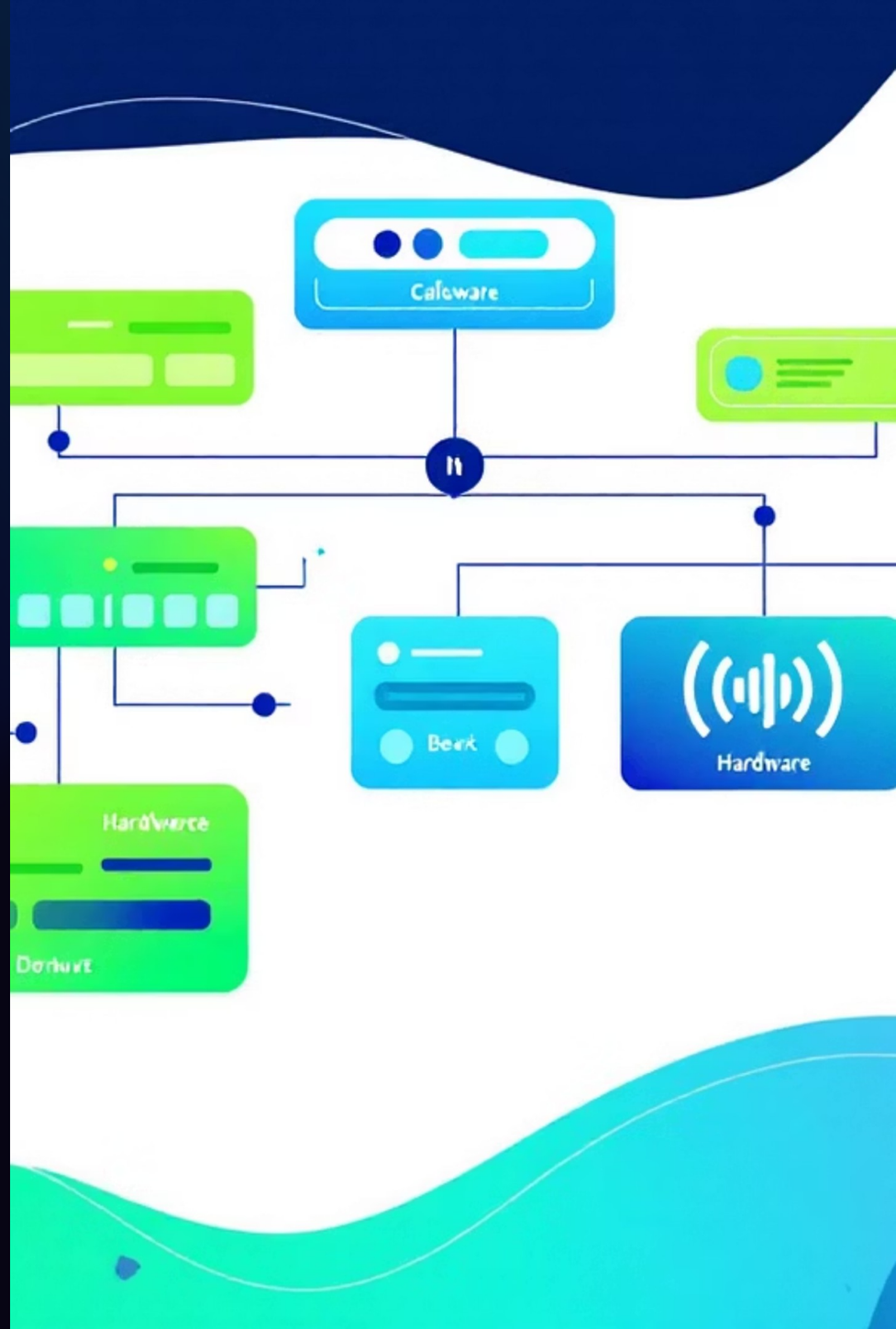
Hardware blocks (e.g., &gpio0)

Properties

Configuration data (pins, levels)

Labels & Aliases

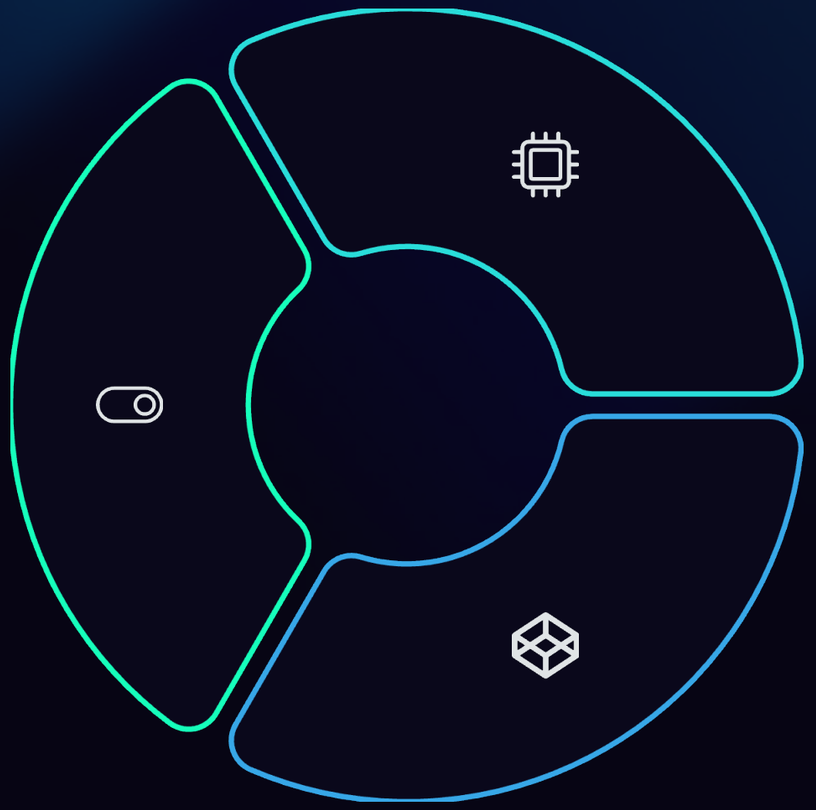
Simplified code access



How Kconfig & DeviceTree Work Together

Kconfig

Enables drivers (CONFIG_GPIO=y)



DeviceTree

Defines hardware (LED on GPIO0, pin 13)

Application

Uses enabled, defined hardware

Live Example – LED Blink

Start with Hello World

Base project structure

Edit Files

prj.conf and main.c

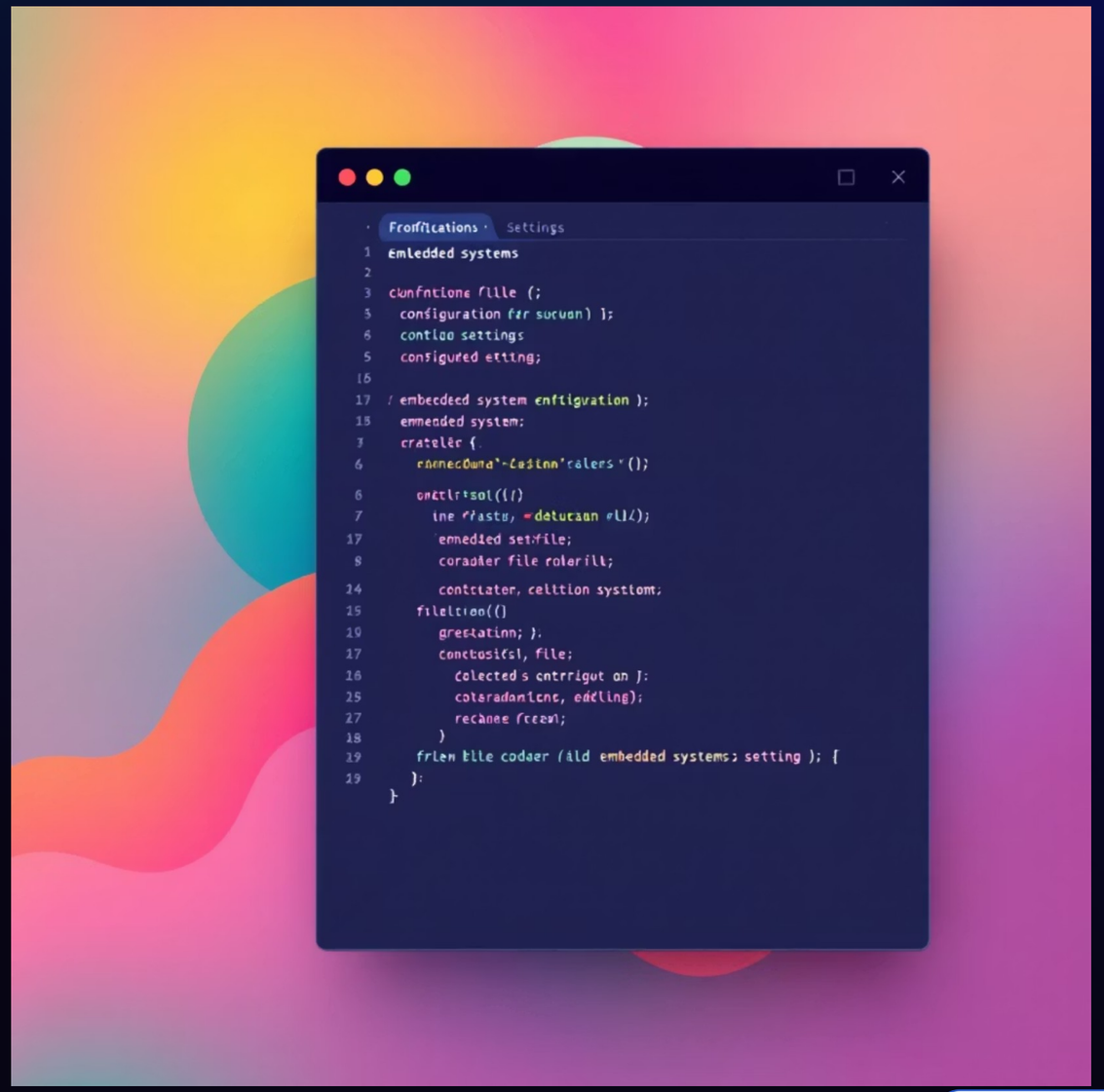
Add app.overlay

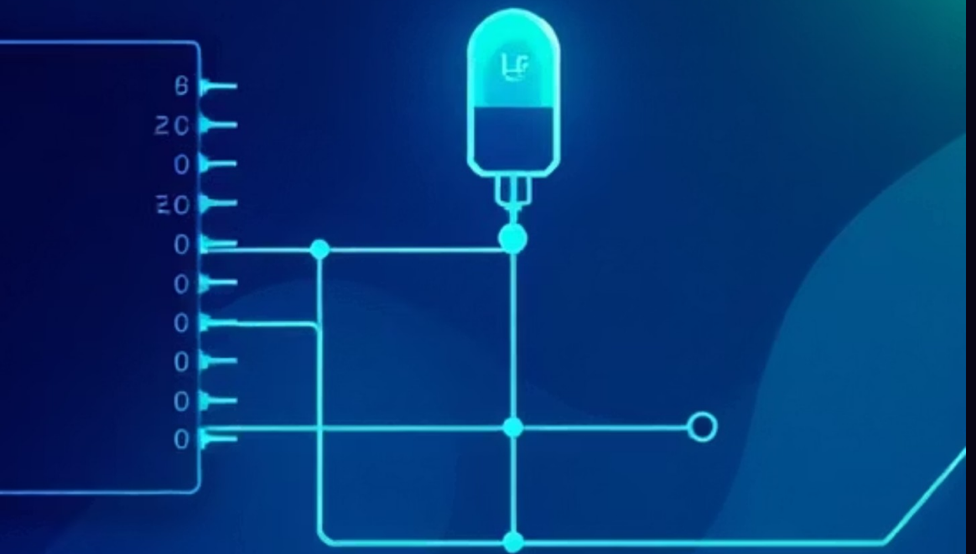
Define LED alias

prj.conf – Example

```
CONFIG_GPIO=y
```

Enables the GPIO driver subsystem





app.overlay –

```
/ {
        aliases     {
                led0 = &led0;
            };
};
```

Defines hardware reference for LED

main.c – DeviceTree Usage

Add the code from blinky led sample to main.c

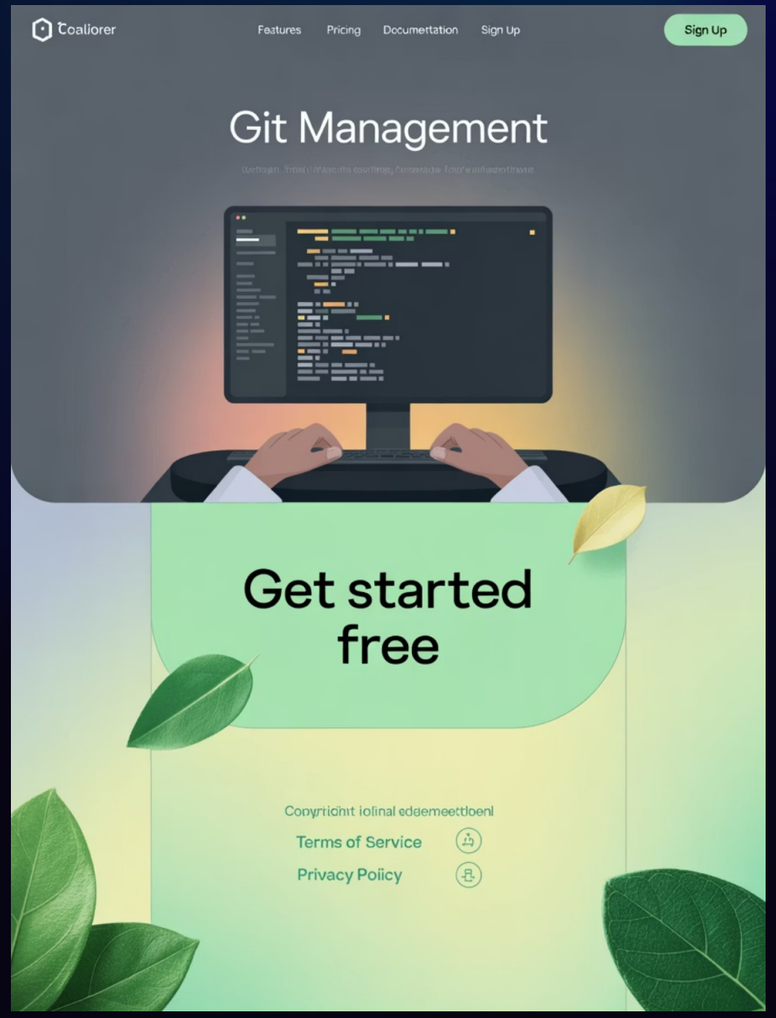
Run the sample

- build and flash on nrf52dk

  files .dts and _defconfig from the folder

  boards/nrf52dk generates the configuration

- Check the file build/zephyr/.config
- Check the file build/zephyr/zephyr.dts



Explore menuconfig

Run: west build -t menuconfig

Enable or disable options

Compare output with build/zephyr/.config now