LITE Lightning Talks
Linaro LITE team
Zephyr board configuration

Erwan Gouriou
Why discussing about board configuration?

● Significant part of Zephyr
  ○ 85+ boards available today in Zephyr
  ○ Board porting PR + board configuration PR => lot of reviews

● Important matter:
  ○ First new comer’s contact with Zephyr: samples/tests easily or not on his board
  ○ Testing/CI: Easier “board <> test” matching => More tests run => More efficient CI

● What do I mean by configuration? Different levels
  ○ Driver selection (UART_STM32, …) > SoC dependant
  ○ Driver configuration (speed) > Application dependant
  ○ Other: Pin configuration
STM32 boards moves since last release

● Reduce use of configuration flags in /boards (1/2): STM32 specific
  ○ Move driver selection bits from boards/ to arch/ (most depends on SoC not board)
  ○ Factorized driver configuration when possible (beneficiate from abstraction provided by HAL/LL)

● Reduce use of configuration flags in /boards (2/2): Zephyr generic
  ○ Device tree deployment (I2C, SPI)
  ○ Move .fixup files (device tree generation artefact) from boards/ to arch/

● Recommended default board configuration (PR #5653)
  ○ Goal: Provide objective rules to minimize discussions
  ○ Enable use of embedded HW (sensors, LED, Buttons, comm i/f (USB, Eth), BT, VCP, ....)
  ○ When available: configure expansion boards connectors (Arduino, 96b, ....)
  ○ GPIO ports activated by default (deactivation is an application choice)
  ○ Provide a default networking interface
How to improve further?

● Continue device tree deployment:
  ○ pinctrl, clocks, ...
  ○ Remove/minimize use of .fixup files

● Promote use of dts aliases
  ○ Ease board <> test/sample compatibility
  ○ Define a set of zephyr aliases (eg: zephyr,led0 / zephyr,spi1)
  ○ test/samples: aliases should be used for driver configuration
  ○ /boards: provide aliases as default configuration (dts)
  ○ Prototyped in #5181
Specific configuration use case: shields

- Application: Micropython sensor dashboard
  - HW: board + sensors shield
  - SW: Zephyr (default board configuration) + Micropython (application configuration)

- Application configuration files
  - prj_base.conf: upython base conf: Enable & configure NETWORKING, CONSOLE, SENSOR, ...
  - prj_<board>.conf: drivers configuration
  - Shield specific configuration files (sensor configuration and activation): .overlay + dts.fixup
  - => Shield part needed?

- Add shield support within Zephyr: (#6017)
  - board + shield(s) = another board
  - Store shield.overlay, Kconfig.shield under boards/shields/xxxx
  - Activation via Kconfig: CONFIG_SHIELD_XXXXXX = y
**WiFi in Zephyr (Current State)**

- **Zephyr Network Protocols** (WebSock, MQTT, LWM2M, DNS, HTTP, SNTP, COAP)
- uPy modsocket & Zephyr samples
- No WiFi L2 Drivers
- No WiFi control plane APIs
- **Initial Idea**: add WiFi via offload chips/SoC coprocessors.

**Zephyr Native IP Stack**

- **L2 Interface**
  - Ethernet MAC (NXP, Atmel, ST, SLIP/TAP)
  - 802.15.4 MAC (TI, NXP, Nordic)
  - BlueTooth L2CAP (Nordic)
  - BSD sockets

```c
#ifdef CONFIG_NET_OFFLOAD
    Zephyr Native IP Stack
    L2 Interface
    #ifdef CONFIG_NET_OFFLOAD
        TCP/IP Offload Engine (SoC Coprocessor or WiFi chip)
```
WiFi in Zephyr (WIP)

Zephyr Network Protocols

- net_app
- wifi_mgmt
- net_mgmt
- net_context

Zephyr Native IP Stack

- L2 Interface
- Drivers

BSD sockets

- TLS stream
  PR #5985
- Async sockets

#ifdef CONFIG_SOCKET_OFFLOAD

TI SimpleLink WiFi Socket Provider

- Atmel Winc1500 WiFi driver
- ST DiscoIOT WiFi driver

#ifdef CONFIG_NET_OFFLOAD

Zephyr MQTT
  PR #5854

WiFi Offload Pending:
- wifi_mgmt (connect, disconnect, scan)
- net_buf redesign might enable buffer mgmt offload (?)
Zephyr WIP PRs / Links

- **PR #4711**: WINC1500 WiFi module support
- **PR #5075**: Disco l475 iot1 wifi
- **PR #4821**: [RFC] BSD Sockets API: Offloading Support
- **PR #5251**: [WIP/RFC] Wifi mgmt interface and wifi offload
- **PR #5028**: samples: boards: cc3220sf_launchxl: socket echo over WiFi
- **PR #6466**: SimpleLink WiFi Driver with wifi_mgmt functions
- **BUD17-112**: Enabling TCP/IP Offload in Zephyr with TI SimpleLink
Zephyr Networking/BSD Sockets update
Paul Sokolovsky
High-level changes in Zephyr releases

1.10
- Explicit loopback interface (good for testing)
- (Almost) all application-level protocols switched to net_app API
- New HTTP API, hopefully more flexible than the previous
- 15.4 6LoWPAN fixes, known to work on frdm_kw41z

1.11
- OpenThread (sup)port added (not yet tested on Linaro boards?)
- IPv6 Ethernet multicasting reqs implemented
- Static IPv4/DHCPv4 configuration enhancements
- New net buffer allocation code

Tons and tons of bugfixes in both releases!
Plans for 1.12

- Supposed to be LTS release (unclear if it will be)
- VLAN and traffic classification support (Intel team)
- Finalize APIs for WiFi support
- Work on making infrastructure for BSD Sockets to be a layer to implement Zephyr application-level protocols.
- Leverage new net buffer allocation features
- Continue to work on reliability and correctness of IP stack (deadlocks, concurrency issues, details of protocols)
Known issues

- Deadlocks: a) due to allocation issues (need to avoid no-timeout allocs); b) possibly, due to concurrency.
- Some alleged concurrency issues seen, possibly need to add more locking around
- TCP performance issues, and possibly driver issues under high packet load
- Correctness of application-level protocol implementations (would be automagically solved if switching to BSD Sockets API).
- Testing of all these is complicated, would need fault injection and/or external packet generator frameworks.
BSD Sockets API evolution

- Details of argument semantics for calls (e.g. addrlen of accept()/recvfrom())
- Better error handling (including resource leaks on errors)
- Non-blocking sockets behavior fixes (not seen if using poll(), seen otherwise)
- More tests, e.g. test to download a large file (few MBs or whatever) over HTTP and verify hash.

- Infrastructure around socket API.
  - The most important matter is TLS support. RFC PR posted implementing “network streams” API layer on top of sockets, abstracting away TCP vs TLS access (they are quite generic to work on top of other communication channels too).
  - Other helpers, e.g. to resolve address strings like “foo.com:1234”.

- Conversion of the MQTT library to sockets is posted
- A simple sensor webapp written in MicroPython using sockets will be shown at Demo Friday
Thank You

#HKG18
HKG18 keynotes and videos on: connect.linaro.org
For further information: www.linaro.org