Tiny Linux for IoT

Grant Likely
Manish Pandey
20 Sept 2018

OS Options For IoT Devices

- Linux
- Zephyr
- FreeRTOS
- mBED
- *BSD

Tradeoffs

- Cost
- Complexity
- Functionality
- Security

This Presentation is about Linux

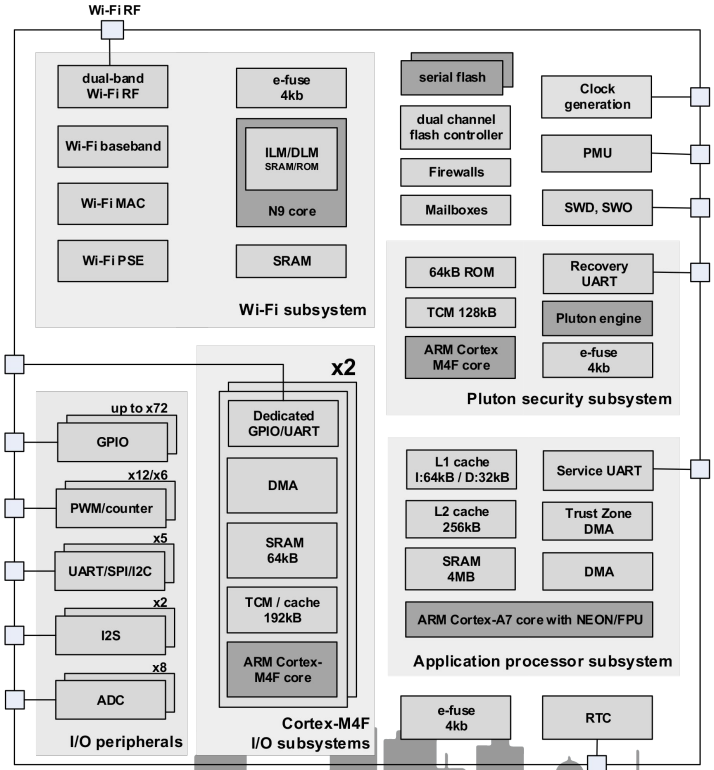
- Typically an RTOS is assumed for IoT devices
  - Zephyr, mBed, FreeRTOS
  - Small footprint
  - Easy to port
  - Many embedded SoCs - Low cost, high volume, secure
- Sometimes Linux is preferred
  - Robust networking stack
  - Wide application support
- Still need to be low cost, high volume and secure
  - How does Linux fit in this model?
- Loads of work done by Nicolas Pitre.

Sadly, limited uptake :-(

Vindication!

Announced earlier this year: MT3620

- 4MB SRAM, 16MB Flash
- Linux on A7 cores
- Single Chip
- Secure OTA updates
- Integrated into Microsoft Azure cloud service



Project to support Tiny IoT Linux

- Whole class of devices suitable for Linux IoT
- Mainline support in Linux kernel
  - Link Time Optimization
  - XIP
  - CRAMFS w/ XIP
  - Minimized configuration
- Userspace pieces still needed
  - Crypto libraries
  - Device access (gpio, spi, i2c, etc)
  - IoT Client Applications
- Spin up project to integrate all things needed for IoT Linux
  - Running on Arm system guidance platform
  - Starting point for building product
  - Opportunity to collaborate on tiny Linux devices

Define “Tiny Linux for IoT”

Targeting a specific subset of devices

General Characteristics:

- <4MB SRAM
- <8MB Flash
- Mixed A+M design
  - A5 running tiny Linux
  - M-class cores for real time and network offload
- Single chip solutions
- Separate secure enclave to manage updates, crypto and image validation

- Common baseline for IoT Agents
  - Connectivity
  - Common libraries (libssl, etc)
  - Busybox
  - Customizable for any IoT agent
    - E.x. Google IoT, Amazon, Azure, IBM, ...

A5 Tiny Linux Project

- Initial prototype up on git.linaro.org
- Runnable on A5 FVP model

https://git.linaro.org/landing-teams/working/arm/manifest.git/tree/readme.txt?h=tinylinux_giot_demo

1. Software Components:

    1. Boot-wrapper: bare minimum bootloader.
    2. Linux: Based on kernel 4.15 with some patches.
    3. Busybox
    4. Device tree
    5. ramdisk
    6. Build scripts: Building SW stack and packaging ramdisk image.
    7. Run scripts: Simple script to run the fast model.
    8. iot-client: Simple C application to send message to google IoT core cloud and dependent libraries.
    9. tools: ARM cross compile toolchain

Memory Numbers

- **Static Memory:**
- Kernel

Kernel/total	init	data	bss
2.8MB	2.6MB	150KB	72KB

- File System Size - 4.4 MB containing Busybox + GIoT sample application and dependent libraries.

Dynamic Memory:

Total used memory around 12MB (restricted by “mem = 12M”) we can run Client application successfully without any OOM.

# free:	total	used	free	shared	buffers	cached
	12860	11268	1592	0	0	6108

Future Work

- Currently CONFIG_THUMB2 is disabled in our config, enabling it will improve on our Flash size requirement.
- Replacing Boot wrapper with TF-A.
- GCC does not allow THUMB2 and LTO_MENU config together.
- Dynamic memory optimization.
- Making user space XIP and optimizing it.
- Upstreaming kernel patches.
- Fine tune optimization for both static and dynamic memory.

Questions?