

# An Introduction to the Operating Systems of the IoT

Elahe Jalalpoor and Parham Alvani

Amirkabir University of Technology

`elahejalalpoor@gmail.com`

`parham.alvani@gmail.com`

Jun 22, 2015

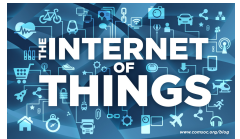


## What is IoT...

The Internet of Things (IoT) is a scenario in which objects, animals or people are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

## What is IoT...

The Internet of Things (IoT) is a scenario in which objects, animals or people are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.



## Open Source Operating Systems for the IoT

- ▶ FreeRTOS
- ▶ RIOT
- ▶ Contiki
- ▶ TinyOS
- ▶ Embedded Linux
- ▶ OpenWSN



# FreeRTOS

- ▶ FreeRTOS is designed to be **small** and **simple**.
- ▶ The kernel itself consists of only three or four C files.
- ▶ It provides methods for multiple threads or tasks, mutexes, semaphores and software timers.
- ▶ Key features are **very small memory footprint**, **low overhead**, and **very fast execution**.



- ▶ RIOT is a **real-time multi-threading** operating system.
- ▶ RIOT is based on design objectives including:
  - Energy-Efficiency
  - Reliability
  - Real-Time Capabilities
  - Small Memory Footprint
  - Modularity
  - Uniform API Accessindependent of the underlying hardware  
(this API offers partial POSIX compliance)



- ▶ Contiki is an open source operating system for **networked**, **memory-constrained** systems
- ▶ Contiki provides three network mechanisms:
  - The uIP stack, which provides IPv4 networking,
  - The uIPv6 stack, which provides IPv6 networking,
  - The Rime stack, which is a set of custom lightweight networking protocols designed specifically for low-power wireless networks.

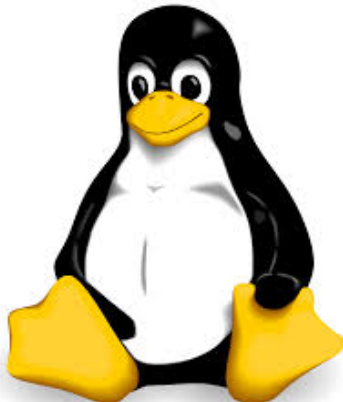


- ▶ TinyOS is a **component-based** operating system and platform targeting wireless sensor networks.
- ▶ TinyOS is an embedded operating system written in the **nesC programming language** as a set of cooperating tasks and processes.



# Embedded Linux

- ▶ Embedded Linux is created using OpenEmbedded, the build framework for embedded Linux.
- ▶ OpenEmbedded offers a best-in-class cross-compile environment.





- ▶ The goal of the OpenWSN project is to provide open-source implementations of a complete protocol stack based on Internet of Things standards, on a variety of software and hardware platforms.



# Comparison

OS	Min RAW	Min ROM	C Support	C++ Support
Contiki	$< 2kB$	$< 30kB$	Partial support	No support
Tiny OS	$< 1kB$	$< 4kB$	No support	No support
Linux	$\sim 1MB$	$\sim 1MB$	Full support	Full support
RIOT	$\sim 1.5kB$	$\sim 5kB$	Full support	Full support



# Comparison

OS	Multi-Threading	Modularity	Real-Time
Contiki	Partial support	Partial support	Partial support
Tiny OS	Partial support	No support	No support
Linux	Full support	Partial support	Partial support
RIOT	Full support	Full support	Full support



# Why Not Linux?

## Real-Time Linux

Controlling a laser with Linux is crazy, but everyone in this room is crazy in his own way. So if you want to use Linux to control an industrial welding laser, I have no problem with your using `PREEMPT_RT`.

- Linus Torvalds



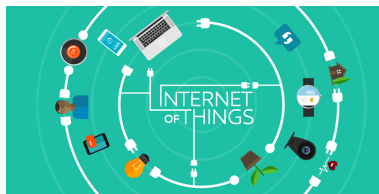
# Why Not Linux?

- ▶ Linux certainly is a robust, developer-friendly OS
- ▶ Linux has a disadvantage when compared to a real-time operating system:
  - Memory footprint
  - It simply will not run on 8 or 16-bit MCUs



## Close Source Operating Systems for the IoT

- ▶ ARM mbed
- ▶ Huawei LiteOS
- ▶ Google Brillo



- ▶ Automation of power management
- ▶ Software asset protection and secure firmware updates for device security & management
- ▶ Connectivity protocol stack support for Bluetooth low energy, Cellular, Ethernet, Wi-fi, Zigbee IP, Zigbee NAN, 6LoWPAN



- ▶ The company says that its LiteOS is the lightest software of its kind and can be used to power a range of smart devices



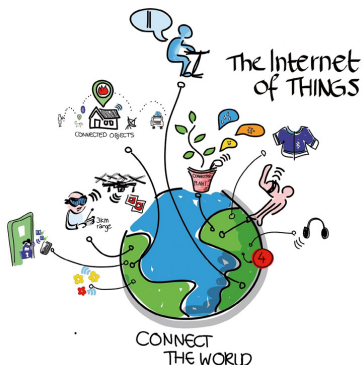


- ▶ Brillo is **derived** from Android but **polished** to just the lower levels.
- ▶ It supports Wi-Fi, Bluetooth Low Energy, and other Android things.



## Requirements for IoT

- ▶ Scalability
- ▶ Modularity
- ▶ Connectivity
- ▶ Reliability



- ▶ Can you build an IoT system with familiar Web technologies?

# Internet Usage and Protocols for the IoT

- ▶ Can you build an IoT system with familiar Web technologies?
- ▶ Yes you can, although the result would not be as **efficient** as with the **newer protocols**.

# Internet Usage and Protocols for the IoT

## Web Hundreds / thousands of bytes



- Inefficient content encoding
- Huge overhead, difficult parsing
- Requires full Internet devices

## Internet of Things Tens of bytes



- Efficient objects
- Efficient Web
- Optimized IP access

# Internet Usage and Protocols for the IoT

<i>Protocol</i>	<i>Transport</i>	<i>Messaging</i>	<i>2G,3G,4G (1000's)</i>	<i>LowPower and Lossy (1000's)</i>	<i>Compute Resources</i>	<i>Security</i>	<i>Success Stories</i>	<i>Arch</i>
<b>CoAP</b>	UDP	Rqst/Rspnse	Excellent	Excellent	10Ks/RAM Flash	Medium - Optional	Utility field area ntwns	Tree
<b>Continua HDP</b>	UDP	Pub/Subsrbr Rqst/Rspnse	Fair	Fair	10Ks/RAM Flash	None	Medical	Star
<b>DDS</b>	UDP	Pub/Subsrbr Rqst/Rspnse	Fair	Poor	100Ks/RAM Flash +++	High- Optional	Military	Bus
<b>DPWS</b>	TCP		Good	Fair	100Ks/RAM Flash ++	High- Optional	Web Servers	Client Server
<b>HTTP/ REST</b>	TCP	Rqst/Rspnse	Excellent	Fair	10Ks/RAM Flash	Low- Optional	Smart Energy Phase 2	Client Server
<b>MQTT</b>	TCP	Pub/Subsrbr Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	Medium - Optional	IoT Msging	Tree
<b>SNMP</b>	UDP	Rqst/Response	Excellent	Fair	10Ks/RAM Flash	High- Optional	Network Monitoring	Client- Server
<b>UPnP</b>		Pub/Subsrbr Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	None	Consumer	P2P Client Server
<b>XMPP</b>	TCP	Pub/Subsrbr Rqst/Rspnse	Excellent	Fair	10Ks/RAM Flash	High- Mandatory	Rmt Mgmt White Gds	Client Server
<b>ZeroMQ</b>	UDP	Pub/Subsrbr Rqst/Rspnse	Fair	Fair	10Ks/RAM Flash	High- Optional	CERN	P2P

- ▶ IoT device will require a modular operating system that separates the core kernel from middleware, protocols, and applications.

# Multi-Tasking, Thread Model

- ▶ Most RTOS products on the market are thread model.
- ▶ **Tasks** are now called **threads**.
- ▶ All the **tasks** code and data occupy **the same address space**, along with that of the RTOS itself.





# Event-Driven, Non-Blocking I/O Model



# Questions?