The Many IoT Roles Of Embedded Linux

SCALE15X

March 2017

Rev 1.2

http://www.hy-research.com/

Agenda

- What is IoT?
- IoT Connections and Roles
- Embedded Linux
- Embedded Linux Roles
- Linux vs Bare Metal
- Sample IoT devices/Demo
- Security
- Conclusion

What is IoT?

- Internet of Things, new buzz word
- Connected devices
  - IP (Internet Protocol)
  - Includes both **I**nternet (public) and **i**nternet (private) aka as intranet
- Viewable from the internet
  - Interaction from anywhere. i.e. using a smart phone
- Control/configureable from the internet
- Sensors, Switches, and other devices
  - Home automation
  - Weather station
  - “Smart devices”

Connections

- Ultimately via IP (either IPv4 or IPv6)
  - Wired: Ethernet, USB
  - Wireless: WiFi, Bluetooth Classic (PAN/DUN profiles)
- Wireless
  - Zigbee
  - Custom/proprietary
  - Bluetooth Low Energy
  - Phone (GPM/GPRS/EDGE/3G/LTE/etc)

IoT Roles

- IoT sensors and devices
  - Self contained
  - May have IP precluding constraints
- Gateway
  - Data aggregation
  - Translation to/from IP

http://www.hy-research.com/

What is Embedded Linux?

- Same code base as regular desktop Linux
- Different user land pairing
- Tuned for a specific use.
  - Options hard coded for specific set of HW
- Targeted for embedded hardware

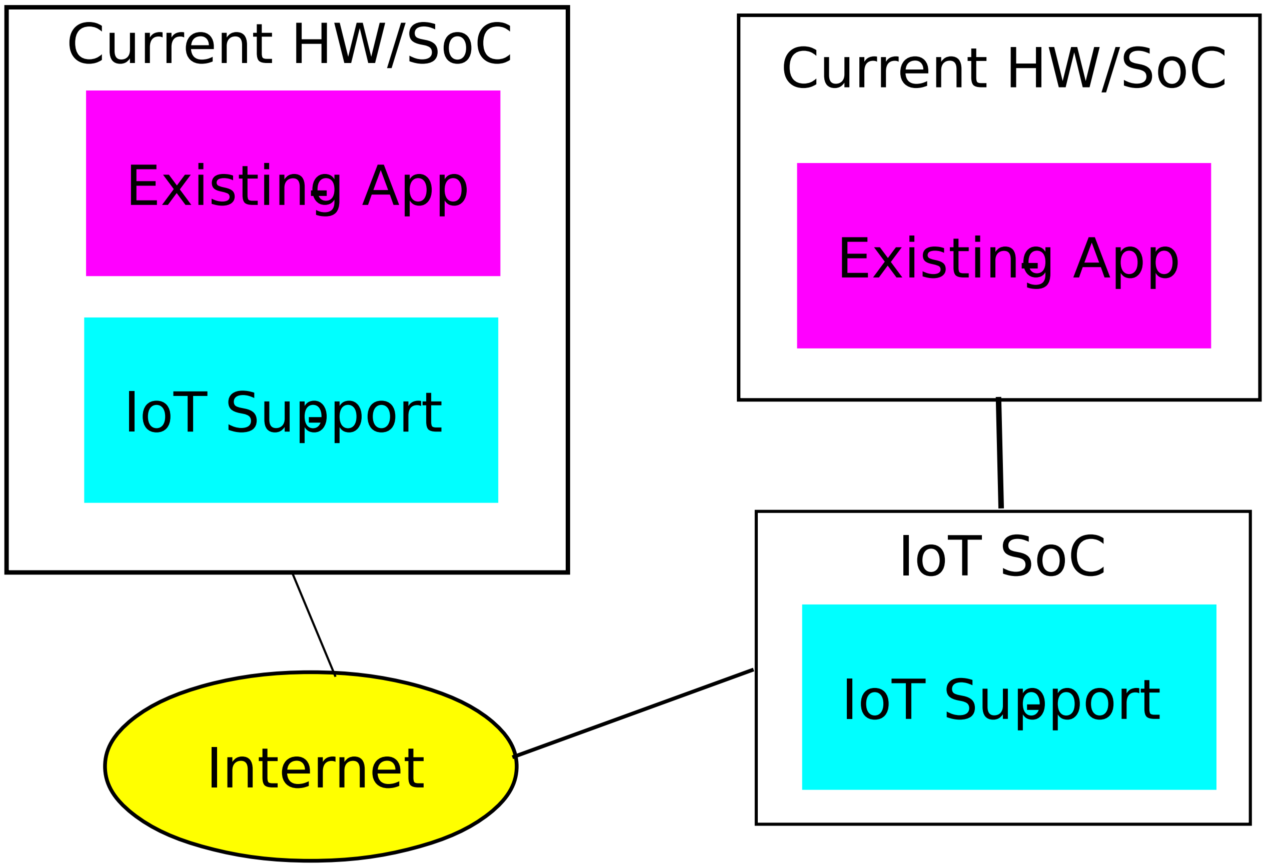
Embedded Linux IoT Roles

- Gateway/aggregator
  - Addresses resource issues
- Self contained device
  - IP Camera
- Augmenting existing device for IoT
  - SW: Added features + connection on existing platform
  - HW: In line with existing connection

Linux Gateway Role

- Mature IP stack with routing support
  - Both IPv4 and IPv6
- Ethernet/WiFi support
  - Utilized by many routers
- Bluetooth wireless support (multiple stacks with varying licenses, default is BlueZ)
  - Bluetooth Classic
  - Bluetooth Low Energy
- Other protocols in userland

Adding IoT with Linux



Partitioning can be virtual or physical

http://www.hy-research.com/

Partitioning

- Processes
  - Memory/resource isolation
- Different Linux users
  - Sharing via permissions
- Containers
  - Shared kernel but more specific limitations
  - Grouping/hiding of resources
- Virtualization
  - Total isolation
  - Multiple OS (mix of bare metal and Linux)
  - More resources

Why Embedded Linux? (vs Baremetal/RTOS) – Pro's

- Large library of drivers (network, sensors, etc)
- Vast range of source code to leverage
- Wide processor/SoC support
- Application support (userland libraries)
- Tested application stacks (LAMP, etc)
- Mature IP stack
- Kernel-user isolation
- Inter-user isolation

Why Embedded Linux? (Con't)

- Open Source!
- Simple to simulate/can leverage desktops
- Development resource availability

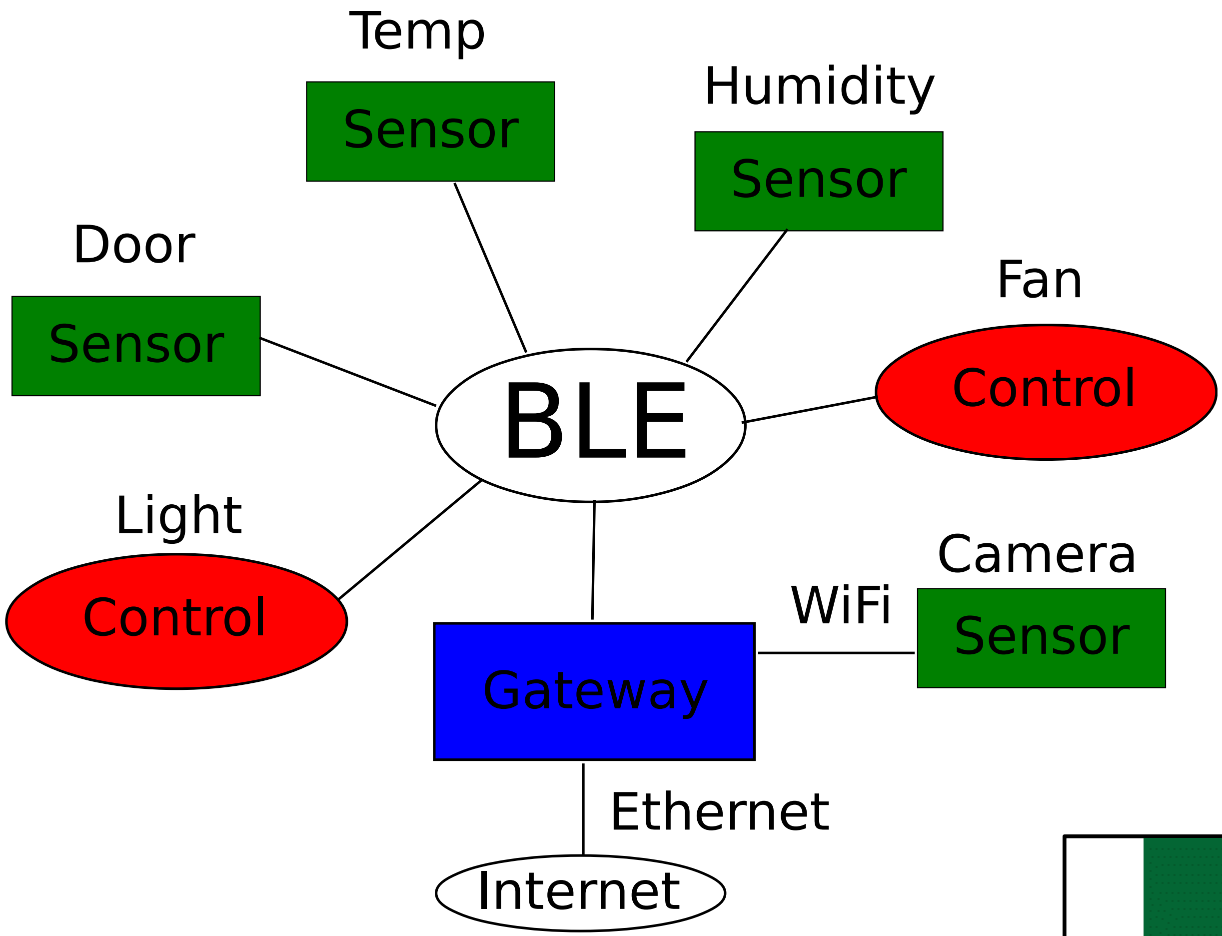
Drawbacks (vs Bare Metal/RTOS)

- Larger footprint then bare metal
  - RAM requirements
  - Flash/image requirements
- Boot time
- Not RT
- Bigger SoC (MMU)
  - NOMMU/uCLinux
- Potentially higher power consumption
  - Code maturity is a factor
  - More complex to tune
- Licensing
  - See license for details

Pairing Baremetal with Linux

- Best of both worlds
- Linux for processing and gateway functions
  - Compression, filtering, and other intelligence
- Baremetal for ultra low power/low cost
  - Sensors arrays
- Baremetal for real time
- Embedded in SoC or separate
  - On chip co-processors
  - Seperate chip
  - Virtualization

Sample IoT (Home Automation)



http://www.hy-research.com/

Examples in the Market

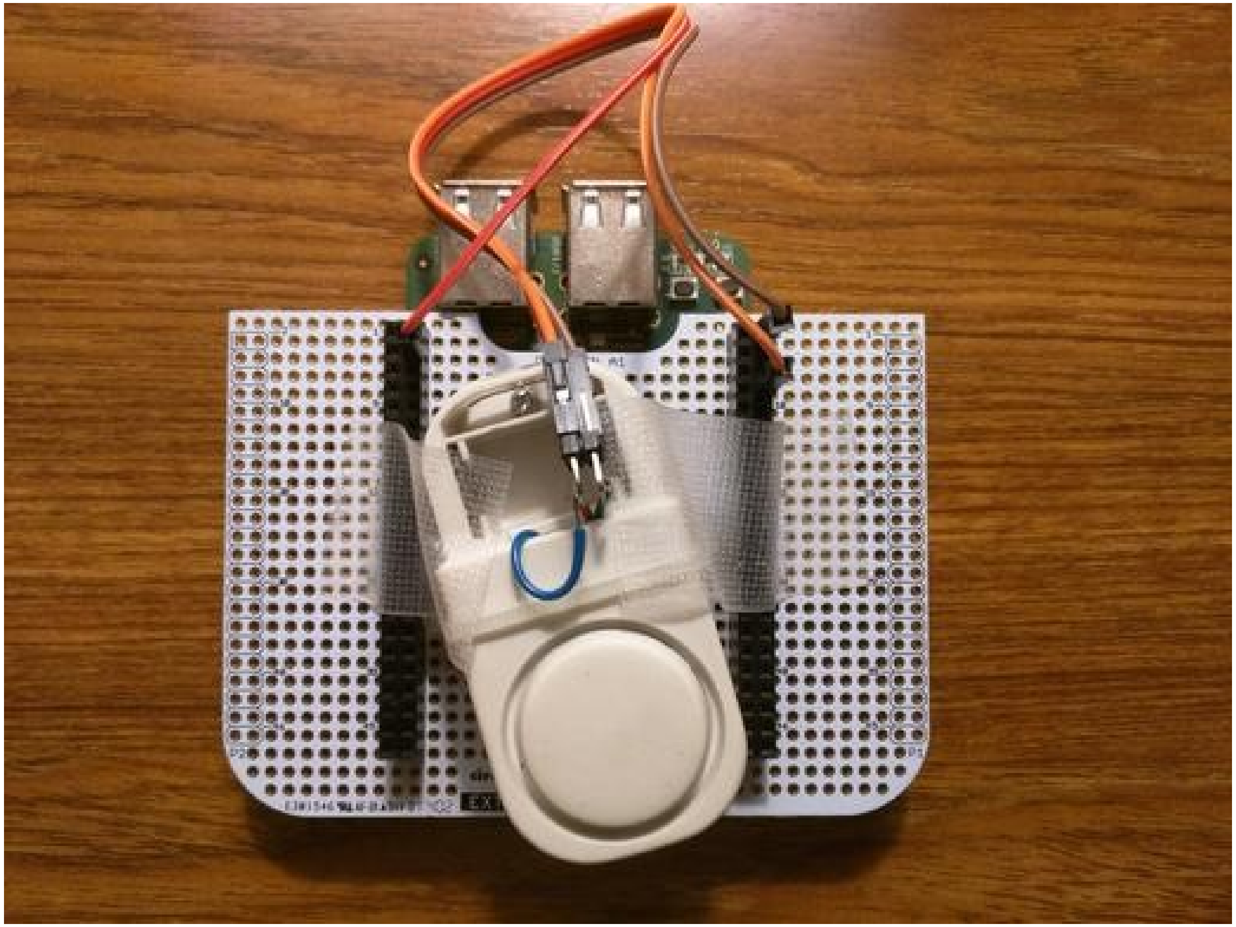
- Gateway: Wink Hub
  - Embedded Linux based Hub provides ZigBee/ZwaveWiFi/ethernet/etc home automation gateway

- IoT DLink IP Camera
  - DCS-93x
  - Ethernet and WiFi

DCS-932 image from http://support.dlink.com/
Wink Hub image from https://www.wink.com

Demo

- IoT Connected Siren
- Built around a BeagleBone Green Wireless
  - BeagleBone with a Wireless module on board
  - Comes with a Linux image and acts as a wireless access point
- Implemented with less then 20 additional lines of code to the stock image
- Write up at: http://www.hy-research.com/iotdemo.html



http://www.hy-research.com/

Security

- IoT devices often on public/open network
- Compromise can expose device to malicious use.
  - Determine real world info like occupancy
  - Control real world things
- IoT device can become a drone (“botnet”) for DDoS

Securing Embedded Linux

- Covers a lot of things
- Start with desktop Linux security
  - Up to date patches
  - Proper configuration
    - Provide only needed services
    - Restrict access (i.e. local network only)
  - Minimal tools on deployed devices.
- Determine needed patches and deployment plan

Brute force Security

- Should not be needed
- SELinux
  - Complicate policies. Can obscure holes
- IPTables (Packet filter)
  - Does not replace disabling services

Embedded Linux Security Issues

- Open services (userland)
- Open services (kernel config)
- Left over bits from sample/demo rootfs
  - Open users
  - Passwords (or lack of)
  - Default passwords/well known passwords
  - sudoers file
- Rootfs/file system

Bad Practices

- Passing things directly to any calls that is expanded by the shell (i.e. system())
- Failing to sanitize any data coming from the outside.
  - Avoid SQL injection
  - Buffer overruns
  - IP addresses/ports
  - DNS names
- Running everything as root

Bad Practices (con't)

- Not using HTTPS. (or at least not considering it)
  - Several choices on embedded Linux
- Using well known passwords for services
  - Obscurity doesn't help
  - Recent IoT DdoS

IoT Data issues

- Privacy
- Cloud vs private/local processing

Conclusions

- Embedded Linux as a rapid path to IoT
  - New devices
  - Added IoT Functionality
- Embedded Linux as a gateway
- Be aware of security

Questions?