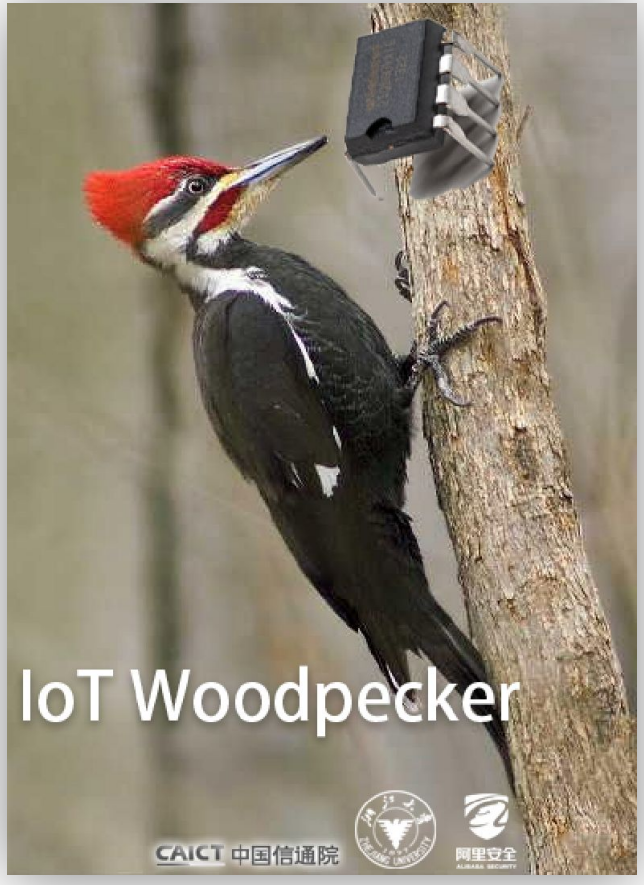


IoT Woodpecker

Intrusion-detection against Hardware Bus

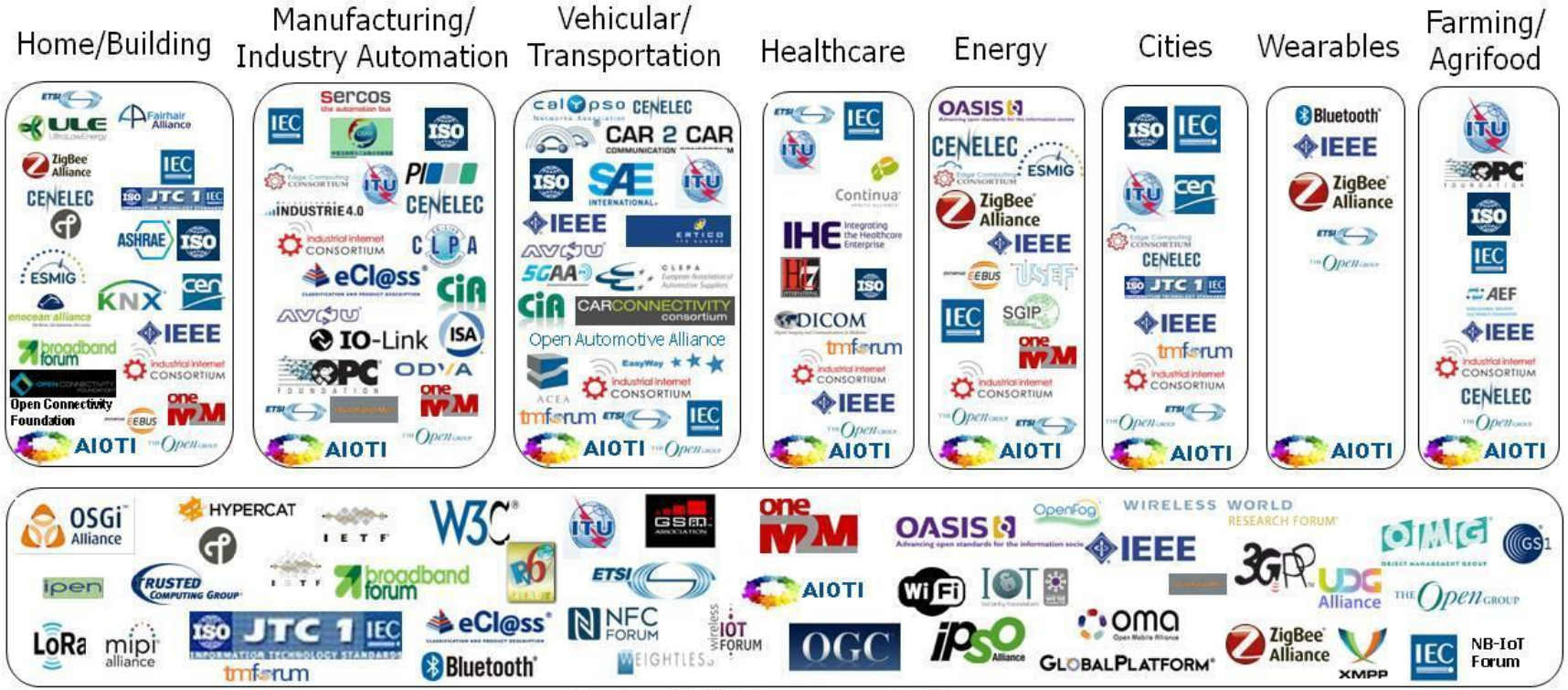
Outline

- IoT Fragmentation & Security Challenges
- IoT Woodpecker Principle & Prototype
- Analysis Against COTS Router
- Combined with artificial intelligence
- Scalable Deployment of IoT Woodpecker
- Future Work
- Takeaways
- *Trailer

Fragmentation of IoT

- Standard
- Hardware
  - CPU Architecture: MIPS/ARM
  - Storage Filesystem: JFFS2/SquashFS/UBI
- Characteristic
  - Characteristics of Home Kit
    - KhaosT/HAP-NodeJS: HomeKitTypes.js
    - brutella/hc: metadata.json
- Operating System
  - Yocto / OpenWrt / Android Things / FreeRTOS
- Manufacturer

IoT SDOs and Alliances Landscape (Vertical and Horizontal Domains)



Source: AIOTI WG3 (IoT Standardisation) – Release 2.8

25 Categories: Bridge, Fan, Garage Door Opener, Lightbulb, Door Lock, Outlet, Switch, Thermostat, Sensor, Security System, Door, Window, Window Covering, Programmable Switch, IP Camera, Video Doorbell, Air Purifier, Heater, Air Conditioner, Humidifier, Dehumidifier, Sprinklers, Faucets, Shower Systems, Other.

125 Characteristics: Accessory Flags, Active, Administrator Only Access, Air Particulate Density, Air Particulate Size, Air Quality, Audio Feedback, Battery Level, Brightness, Carbon Dioxide Detected, Carbon Dioxide Level, Carbon Dioxide Peak Level, Carbon Monoxide Detected, Carbon Monoxide Level, Carbon Monoxide Peak Level, Charging State, Color Temperature, Contact Sensor State, Cooling Threshold Temperature, Current Air Purifier State, Current Ambient Light Level, Current Door State, Current Fan State, Current Heater Cooler State, Current Heating Cooling State, Current Horizontal Tilt Angle, Current Humidifier Dehumidifier State, Current Position, Current Relative Humidity, Current Slat State, Current Temperature, Current Tilt Angle, Current Vertical Tilt Angle, Digital Zoom, Filter Change Indication, Filter Life Level, Firmware Revision, Hardware Revision, Heating Threshold Temperature, Hold Position, Hue, Identify, Image Mirroring, Image Rotation, In Use, Is Configured, Leak Detected, Lock Control Point, Lock Current State, Lock Last Known Action, Lock Management Auto Security Timeout, Lock Physical Controls, Lock Target State, Logs, Manufacturer, Model, Motion Detected, Mute, Name, Night Vision, Nitrogen Dioxide Density, Obstruction Detected, Occupancy Detected, On, Optical Zoom, Outlet In Use, Ozone Density, Pair Setup, Pair Verify, Pairing Features, Pairing Pairings, PM10 Density, PM2.5 Density, Position State, Program Mode, Programmable Switch Event, Relative Humidity Dehumidifier Threshold, Relative Humidity Humidifier Threshold, Remaining Duration, Reset Filter Indication, Rotation Direction, Rotation Speed, Saturation, Security System Alarm Type, Security System Current State, Security System Target State, Selected RTP Stream Configuration, Serial Number, Service Label Index, Service Label Namespace, Set Duration, Setup Endpoints, Slat Type, Smoke Detected, Status Active, Status Fault, Status Jammed, Status Low Battery, Status Tampered, Streaming Status, Sulphur Dioxide Density, Supported Audio Stream Configuration, Supported RTP Configuration, Supported Video Stream Configuration, Swing Mode, Target Air Purifier State, Target Air Quality, Target Door State, Target Fan State, Target Heater Cooler State, Target Heating Cooling State, Target Horizontal Tilt Angle, Target Humidifier Dehumidifier State, Target Position, Target Relative Humidity, Target Slat State, Target Temperature, Target Tilt Angle, Target Vertical Tilt Angle, Temperature Display Units, Valve Type, Version, VOC Density, Volume, Water Level.

40 Services: Accessory Information, Air Purifier, Air Quality Sensor, Battery Service, Camera RTP Stream Management, Carbon Dioxide Sensor, Carbon Monoxide Sensor, Contact Sensor, Door, Doorbell, Fan, Fan v2, Filter Maintenance, Faucet, Garage Door Opener, Heater Cooler, Humidifier Dehumidifier, Humidity Sensor, Irrigation System, Leak Sensor, Light Sensor, Lightbulb, Lock Management, Lock Mechanism, Microphone, Motion Sensor, Occupancy Sensor, Outlet, Security System, Service Label, Slat, Smoke Sensor, Speaker, Stateless Programmable Switch, Switch, Temperature Sensor, Thermostat, Valve, Window, Window Covering.

-- Home Kit

What can we trust?

What code can we trust? (1984)

Can't trust binary so check source →Compiler backdoor

→Inspect the compiler source → C compiler is written in C

What device can we trust? (2018)

Can't trust device so check “source” → Software & hardware backdoors

→ Inspect all the software and hardware design ? HUGE Challenges!

IoT Security Challenges

With the popularity of IoT devices, the difficulty of detecting their intrusions is also increasing. The traditional intrusion detection technology has encountered many new challenges in the IoT field, including:

a) limited computing, storage and power supply capabilities of the device

b) various hardware and software architectures, severe fragmentation

c) huge number of devices

d) (almost) always online

e) Encrypted traffics.

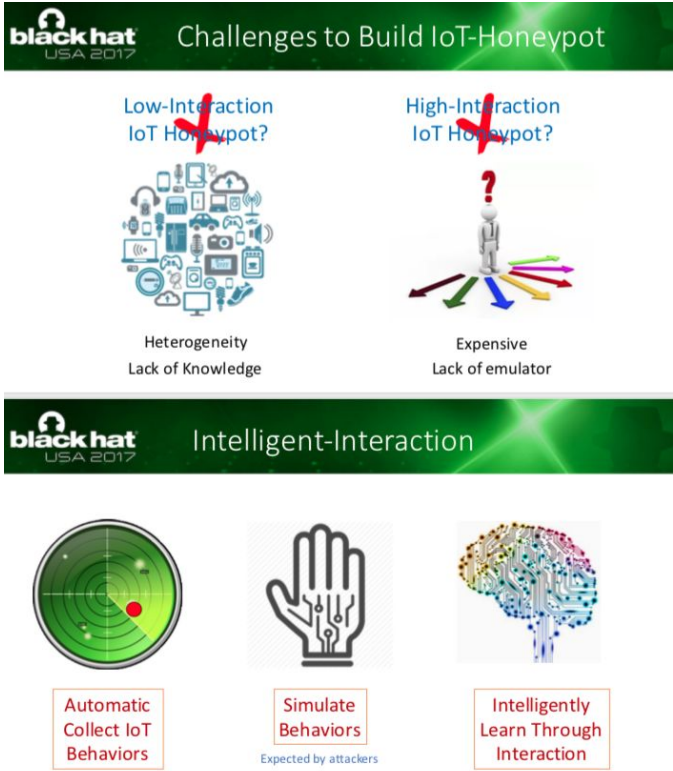
Therefore, traditional solutions, such as setting up monitoring agents on devices, are not applicable in the world of IoT.

IoT Security Events

- IoT becomes BoT (Botnet of Things)
- Mirai
- Two of the solutions are
  - Agent
  - Honeypot

Previous Work

- IoTPOT
  - Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, “IoTPOT: Analysing the Rise of IoT Compromises,”
- IoTCandyJar
  - T. Luo, Z. Xu, X. Jin, Y. Jia, and X. Ouyang, “IoTCandyJar: Towards an Intelligent-Interaction Honeypot for IoT Devices,”



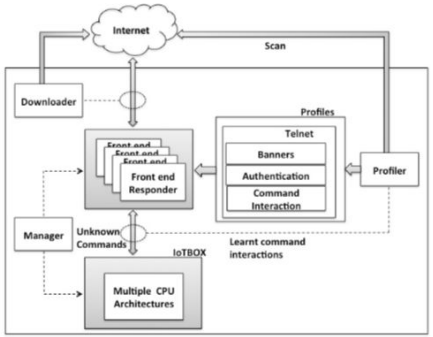


Figure 3 - Overview of IoTPOT

Challenges in traditional methods

Agents

- Endpoint deployment cost to adapt **fragmentation** of devices
- Extra memory, power consumption on constrained environment
- Encrypted information leakage risk

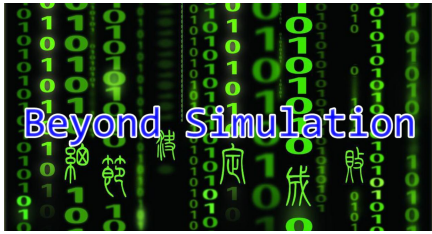
Honeypot

- “Smart Malware”
  - Simulator Detection, honeypot-resistant malwares
  - If attackers knows, can easily bypass them
- “Not an answer for system security”

Amazon IoT Device Defender

Ultimate Honey Pot: Beyond Simulation

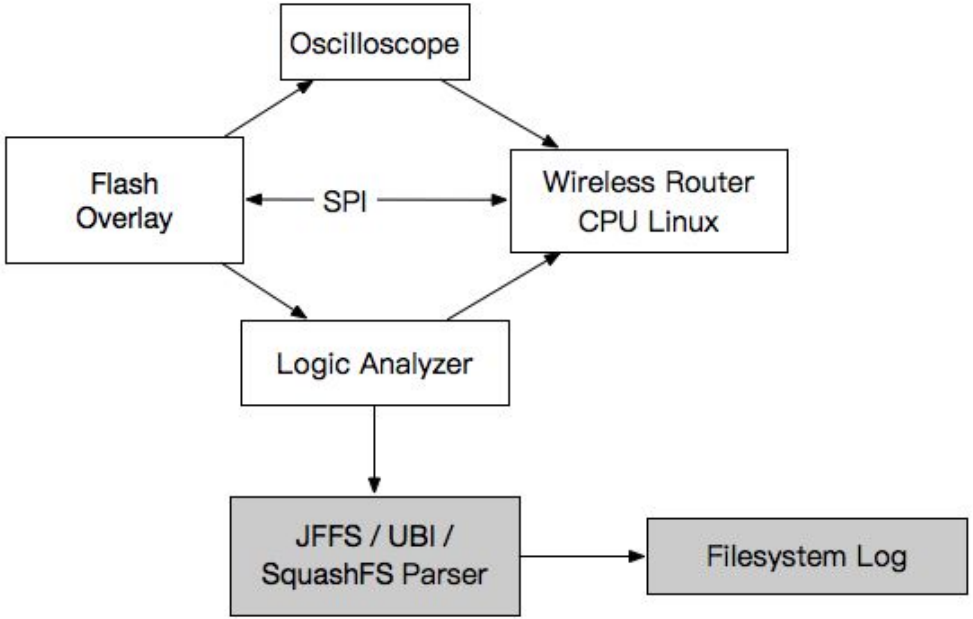
- Since the adversary will be smart enough to detect honeypot
- Why not directly use the device in the wild?
- Make minimal changes to the device (L.M.P.)



One Hypothesis

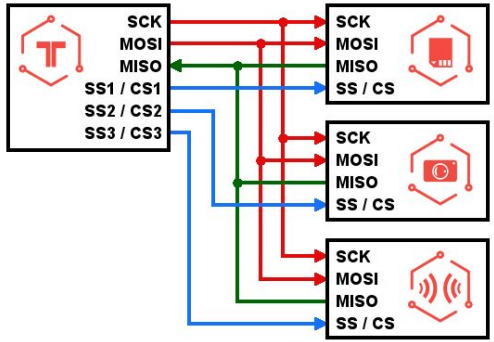
- The inevitability of memory accessing
- The attacker has to store the malware payload in a non-volatile memory (NVM)
- **If not: a simple reboot will solve every problem.**

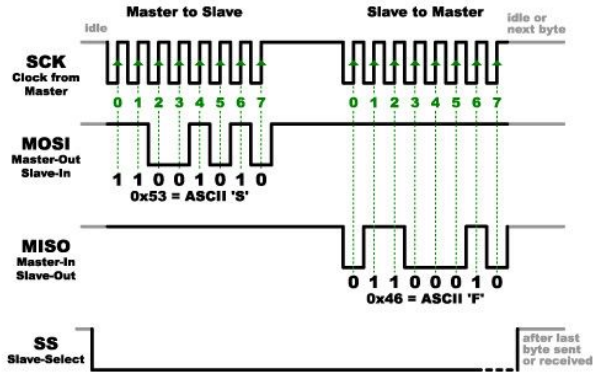
A Simple Prototype of IoT Woodpecker



NVM

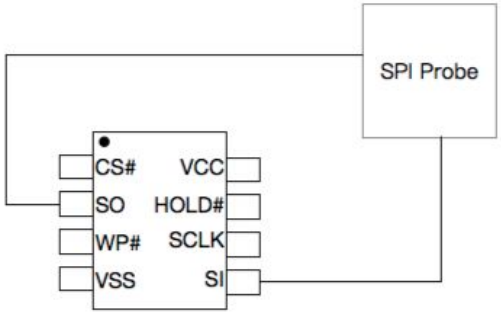
- Types of NVM
  - Serial Flash (SPI)
  - NAND Flash (ONFI)
  - ……
- Enumerable types. No fragmentation problem.

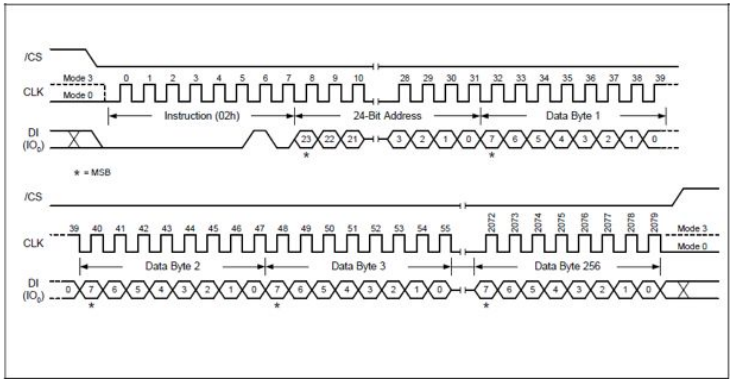




W25Q128BV (Serial Flash)

PAD NO.	PAD NAME	I/O	FUNCTION
1	/CS	I	Chip Select Input
2	DO (IO1)	I/O	Data Output (Data Input Output 1)*1
3	/WP (IO2)	I/O	Write Protect Input (Data Input Output 2)*2
4	GND		Ground
5	DI (IO0)	I/O	Data Input (Data Input Output 0)*1
6	CLK	I	Serial Clock Input
7	/HOLD (IO3)	I/O	Hold Input (Data Input Output 3)*2
8	VCC		Power Supply



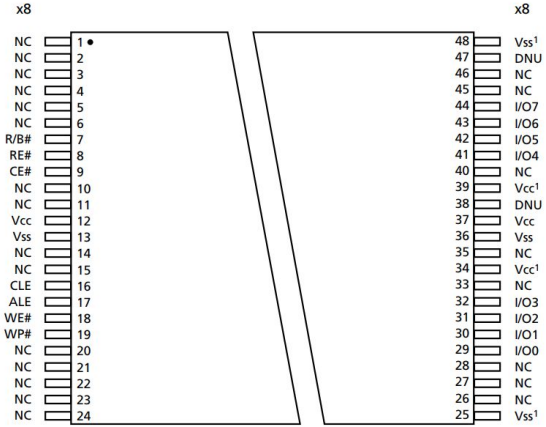


MT29F1G08ABADA (NAND Flash)

Table 1: Asynchronous Signal Definitions

Signal[1]	Type	Description[2]
ALE	Input	**Address latch enable:** Loads an address from I/O[7:0] into the address register.
CE#	Input	**Chip enable:** Enables or disables one or more die (LUNs) in a target.
CLE	Input	**Command latch enable:** Loads a command from I/O[7:0] into the command register.
RE#	Input	**Read enable:** Transfers serial data from the NAND Flash to the host system.
WE#	Input	**Write enable:** Transfers commands, addresses, and serial data from the host system to the NAND Flash.
WP#	Input	**Write protect:** Enables or disables array PROGRAM and ERASE operations.
I/O[7:0] (x8) I/O[15:0] (x16)	I/O	**Data inputs/outputs:** The bidirectional I/Os transfer address, data, and command information.
R/B#	Output	**Ready/busy:** An open-drain, active-low output that requires an external pull-up resistor. This signal indicates target array activity.
V_{CC}	Supply	**V_{CC}:** Core power supply
V_{SS}	Supply	**V_{SS}:** Core ground connection
NC	–	**No connect:** NCs are not internally connected. They can be driven or left unconnected.
DNU	–	**Do not use:** DNUs must be left unconnected.

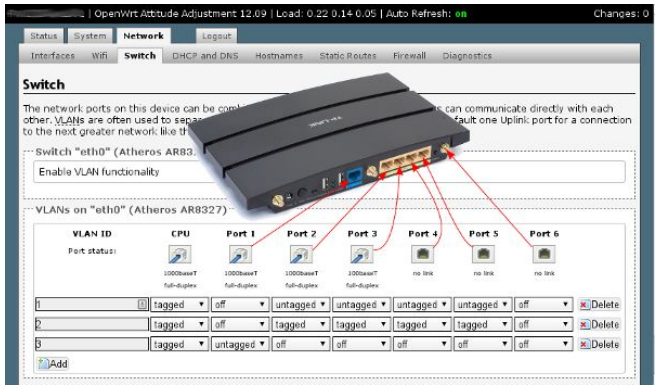
Notes: 1. See Device and Array Organization for detailed signal connections.
2. See Asynchronous Interface Bus Operation for detailed asynchronous interface signal de-



```
# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00040000 00020000 "u-boot"
mtd1: 00040000 00020000 "u-boot-env"
mtd2: 00040000 00020000 "caldata"
mtd3: 00080000 00020000 "pot"
mtd4: 00200000 00020000 "language"
mtd5: 00080000 00020000 "config"
mtd6: 00300000 00020000 "traffic_meter"
mtd7: 00200000 00020000 "kernel"
mtd8: 01700000 00020000 "ubiroot"
mtd9: 01900000 00020000 "firmware"
mtd10: 00040000 00020000 "caldata_backup"
mtd11: 06000000 00020000 "reserved"
mtd12: 001d1000 0001f000 "rootfs"
mtd13: 0118f000 0001f000 "rootfs_data"
```

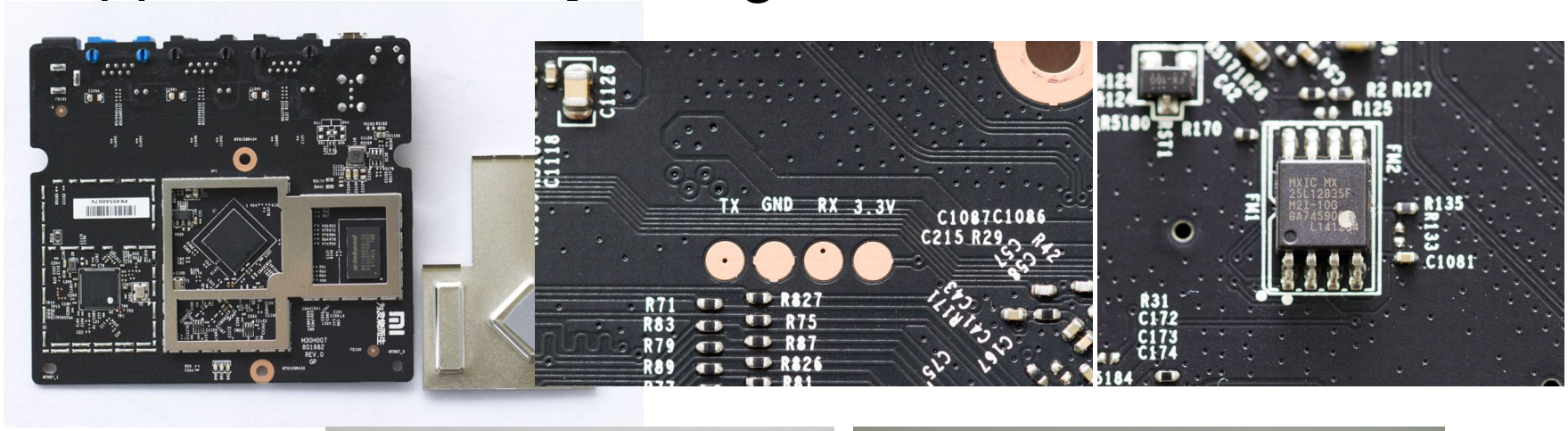
Filesystems in IoT

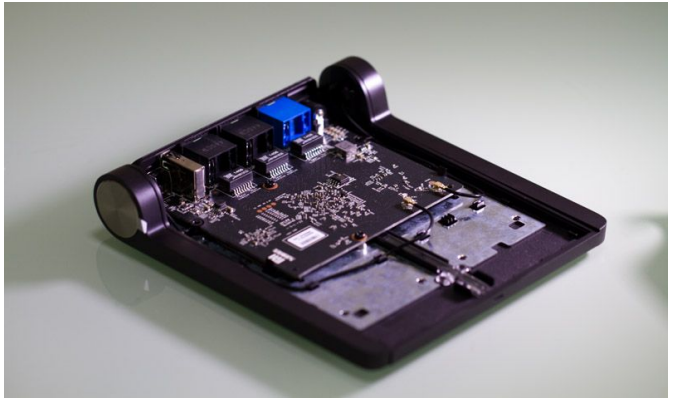
- Optimized for NVM
  - JFFS2
  - SquashFS: read only, compressed
  - UbiFS
  - CramFS
  - YAFFS2
- Overlay
  - Used to merge two filesystems, one read-only and the other writable
- Decompress then loaded into memory



Layer0	raw flash					
Layer1	bootloader partition(s)	optional SoC specific partition(s)	OpenWrt firmware partition			optional SoC specific partition(s)
Layer2			Linux Kernel	**rootfs** mounted: "/", OverlayFS with **/overlay**		
Layer3				**/dev/root** mounted: "**/rom**", SquashFS size depends on selected packages	**rootfs_data** mounted: "**/overlay**", JFFS2 "free" space	

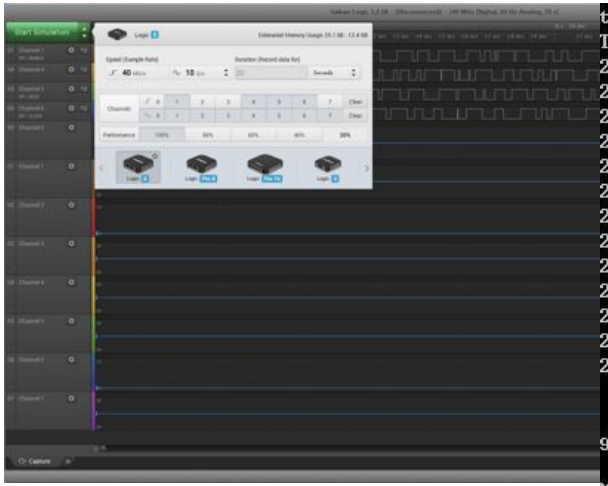
Approach 1: Analysis Against COTS Router





Approach 1: Analysis Against COTS Router

- With Logic Analyzer
- C++ Code with SDK



```
test:
Time [s], Analyzer Name, Decoded Protocol Result
2.053523992000000,SPI,MOSI: '5';  MISO: '0'
2.053524368000000,SPI,MOSI: '0';  MISO: '0'
2.053526744000000,SPI,MOSI: '3';  MISO: '0'
2.053527116000000,SPI,MOSI: '148';  MISO: '0'
2.053527496000000,SPI,MOSI:     ;  MISO: '0'
2.053527864000000,SPI,MOSI: '204';  MISO: '0'
2.053528236000000,SPI,MOSI: '0';  MISO: '194'
2.053528524000000,SPI,MOSI: '0';  MISO: '142'
2.053528856000000,SPI,MOSI: '0';  MISO: '192'
2.053529144000000,SPI,MOSI: '0';  MISO: '188'
2.053529476000000,SPI,MOSI: '0';  MISO: '25'
2.053529764000000,SPI,MOSI: '0';  MISO: '128'
2.053530056000000,SPI,MOSI: '0';  MISO: '0'


get success

MOSI:
05 00 03 94 09 cc 00 00 00 00 00 00 00
MISO:
00 00 00 00 00 00 c2 8e c0 bc 19 80 00
printf data success
```

Analysis Against COTS Router

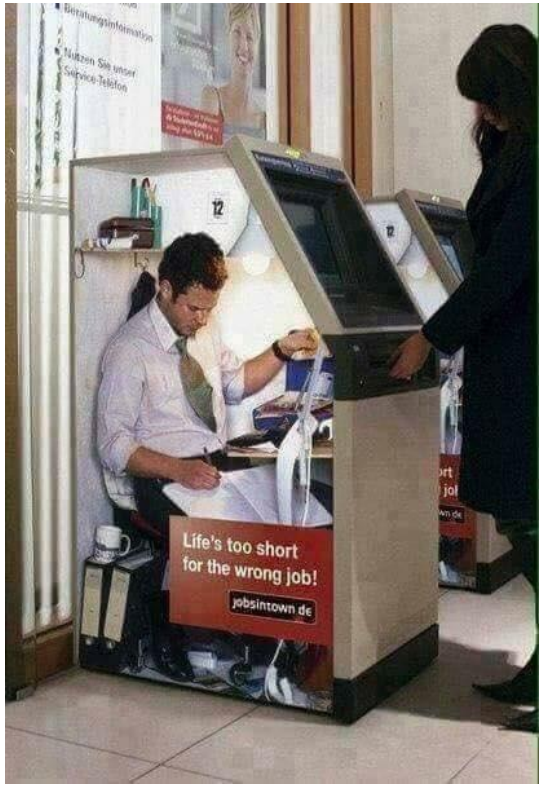
- Tested with malware samples from IoTPOT
- With a simple method based on malware file signature, we get a reasonable success.

Artificial Intelligence

Connected to the risk control system

Human intervention is not elegant

Fortunately, this is 2018, we have AI



What does the data flow look like?

Bidirectional byte streams

Natural NVM accessing log

SPI Example

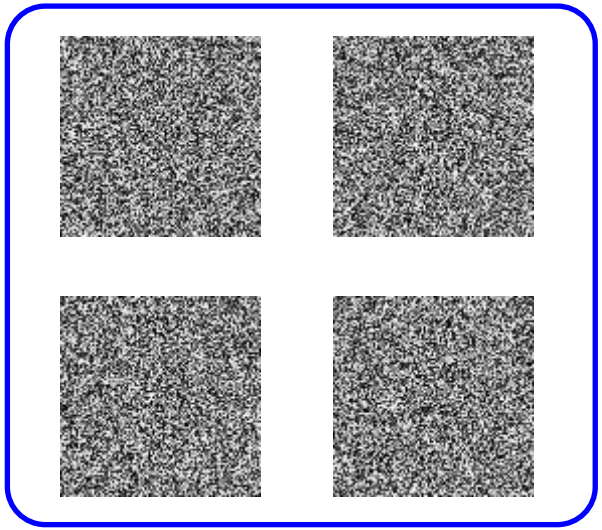
<table>
<tr><td>Layer0</td><td colspan="6">raw flash, 16MB</td></tr>
<tr><td>Layer1</td><td>mtd0
u-boot
192KiB</td><td>mtd1
u-boot-env
64KiB</td><td>mtd2
factory
64KiB</td><td colspan="3">mtd4
firmware
15872KiB</td></tr>
<tr><td>Layer2</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td>mtd5
Kernel</td><td colspan="2">mtd6
rootfs</td><td>mtd8
panic_oops</td></tr>
<tr><td>Layer3</td><td></td><td>/dev/root</td><td>mtd7
rootfs_data</td><td></td></tr>
</table>

Time [s]	Packet ID	MOSI	MISO
6.544088	2	0x00	0x03
6.544102	3	0x05	0xFF
6.544104	3	0x00	0x03
6.544115	4	0x05	0xFF
6.544117	4	0x00	0x03
6.544128	5	0x05	0xFF
6.544130	5	0x00	0x03
6.544140	6	0x05	0xFF
6.544143	6	0x00	0x03
6.544153	7	0x05	0xFF
6.544155	7	0x00	0x03
6.544166	8	0x05	0xFF
6.544168	8	0x00	0x03
6.544179	9	0x05	0xFF
6.544181	9	0x00	0x03
…	…	…	…

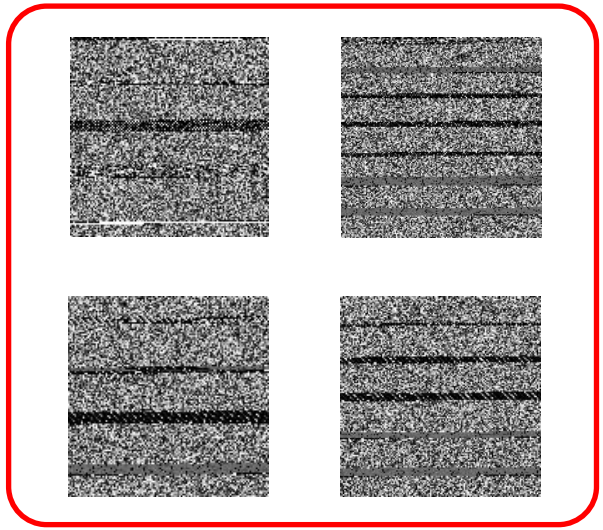
Coding them into images

Plot these data flows as images

Benign data flows



Malware data flows



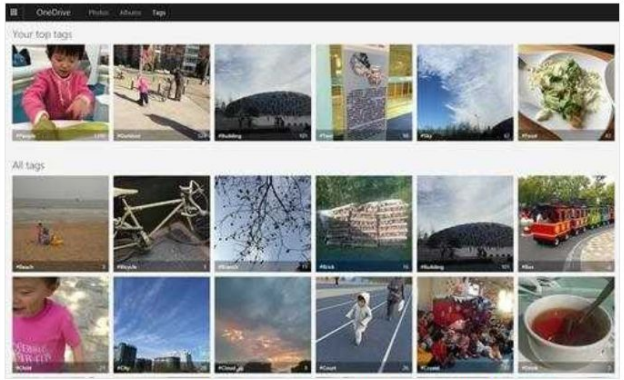
Why image?

AI is now better than humans at recognizing images

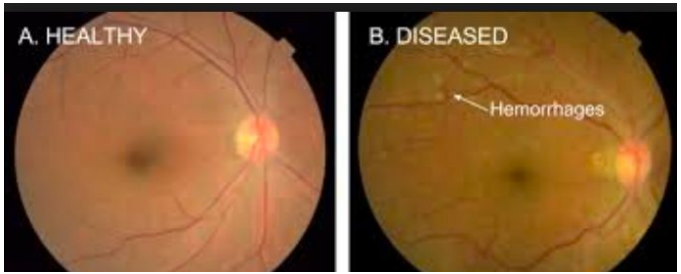
Microsoft, Google Beat Humans at Image Recognition

By R. Colin Johnson, 02.18.15 14

Share Post Share on Facebook Share on Twitter G+ in

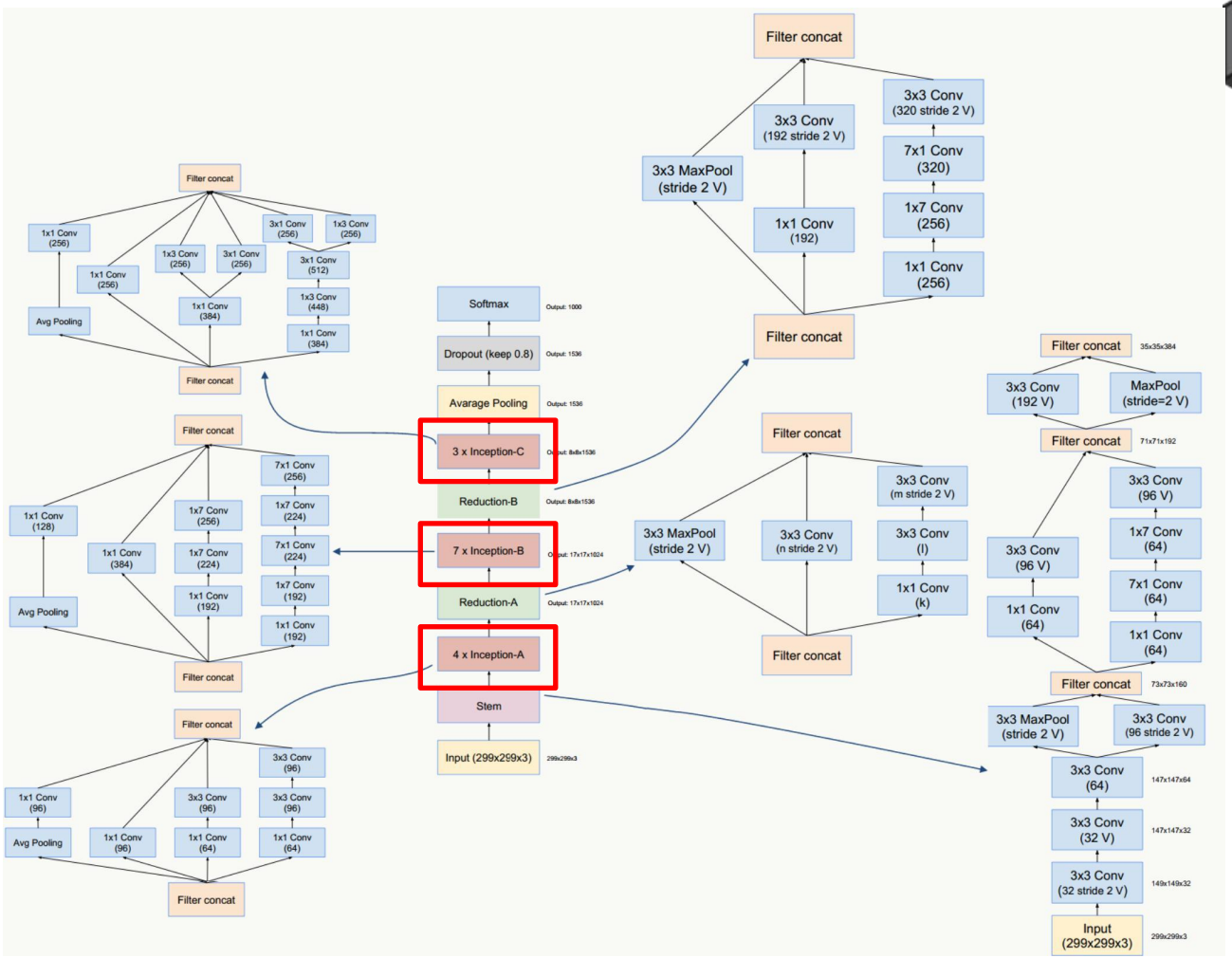


The top row is a representative of the categories that Microsoft's algorithm found in the database and the image columns below are examples that fit.
(Source: Microsoft)



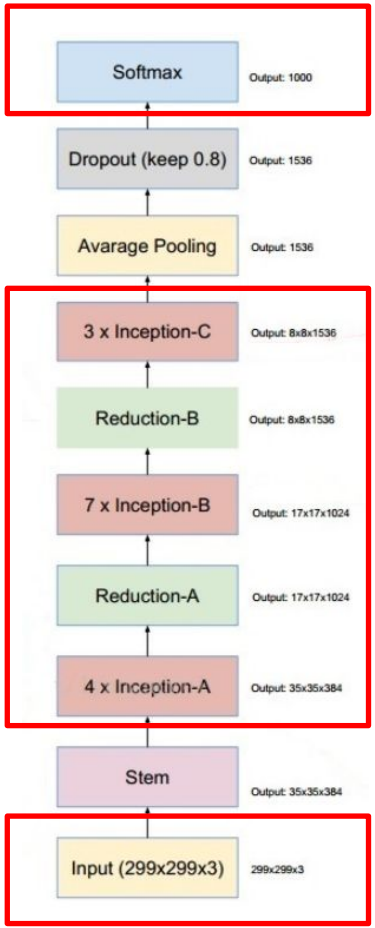
Architecture

Inception v4



Architecture

Transfer learning



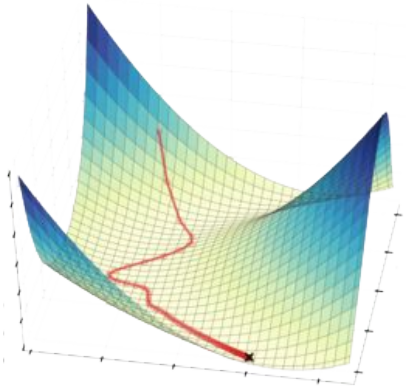
Replace Softmax with Sigmoid

Simplification

Change input size

Training with GPUs

Adam optimizer



```
INFO:tensorflow:Starting Queues.
INFO:tensorflow:global_step/sec: 0
INFO:tensorflow:Recording summary at step 0.
INFO:tensorflow:global step 10: loss = 1.4973 (0.831 sec/step)
INFO:tensorflow:global step 20: loss = 1.4011 (0.764 sec/step)
INFO:tensorflow:global step 30: loss = 1.4867 (0.737 sec/step)
INFO:tensorflow:global step 40: loss = 1.3802 (0.734 sec/step)
INFO:tensorflow:global step 50: loss = 1.3001 (0.744 sec/step)
INFO:tensorflow:global step 60: loss = 1.2600 (0.765 sec/step)
INFO:tensorflow:global step 70: loss = 1.2715 (0.802 sec/step)
INFO:tensorflow:global step 80: loss = 1.1036 (0.736 sec/step)
INFO:tensorflow:global step 90: loss = 1.2531 (0.765 sec/step)
INFO:tensorflow:global step 100: loss = 1.1818 (0.825 sec/step)
INFO:tensorflow:global step 110: loss = 1.2113 (0.751 sec/step)
INFO:tensorflow:global step 120: loss = 1.0533 (0.746 sec/step)
INFO:tensorflow:global step 130: loss = 1.1858 (0.729 sec/step)
INFO:tensorflow:global step 140: loss = 1.0852 (0.776 sec/step)
INFO:tensorflow:global step 150: loss = 1.2871 (0.756 sec/step)
INFO:tensorflow:global step 160: loss = 1.1551 (0.752 sec/step)
INFO:tensorflow:global step 170: loss = 1.0686 (0.738 sec/step)
INFO:tensorflow:global step 180: loss = 0.9936 (0.809 sec/step)
```

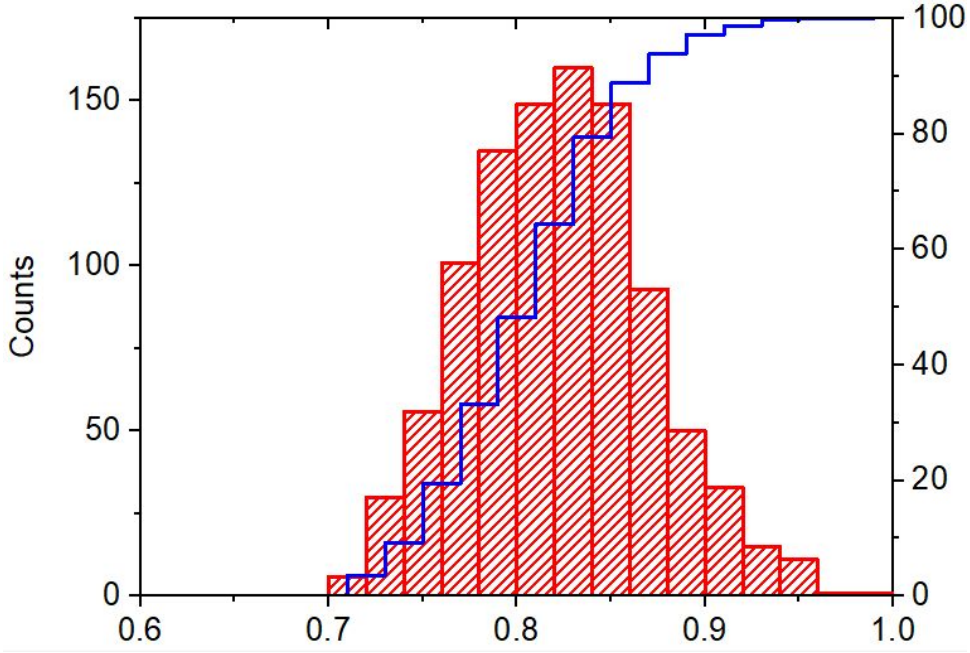
Loss function

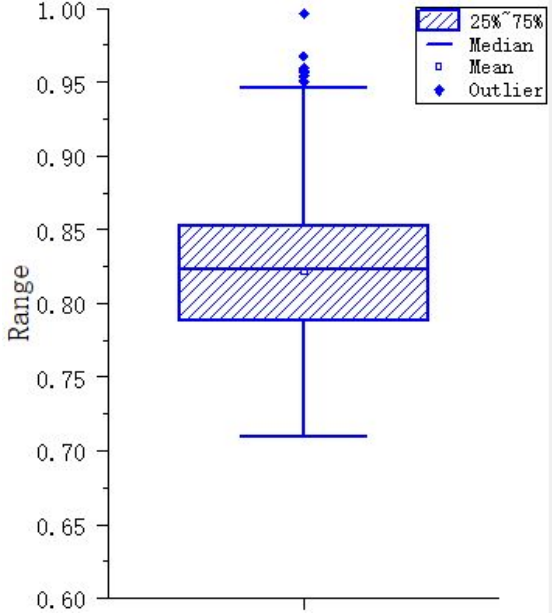
$$L(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

Results

Accuracy

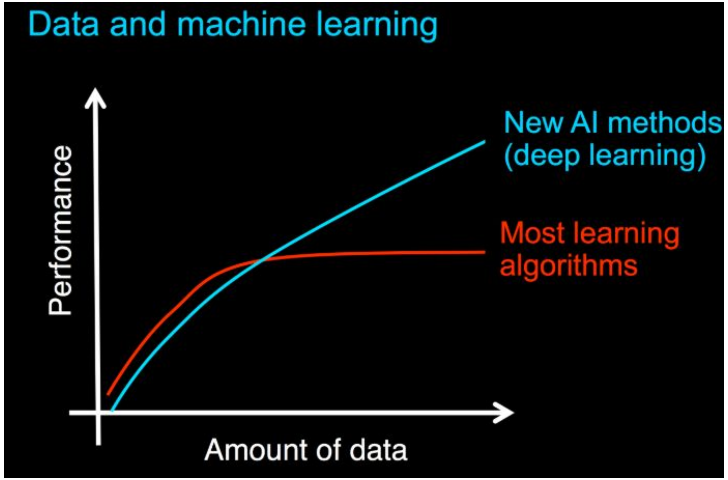
Average: 82.27%
Best: 99.63%
Worst: 71.03%

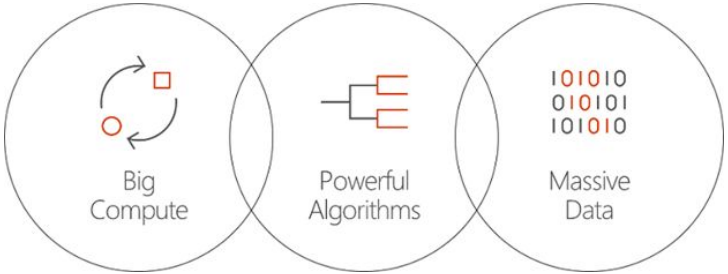




Future work with AI

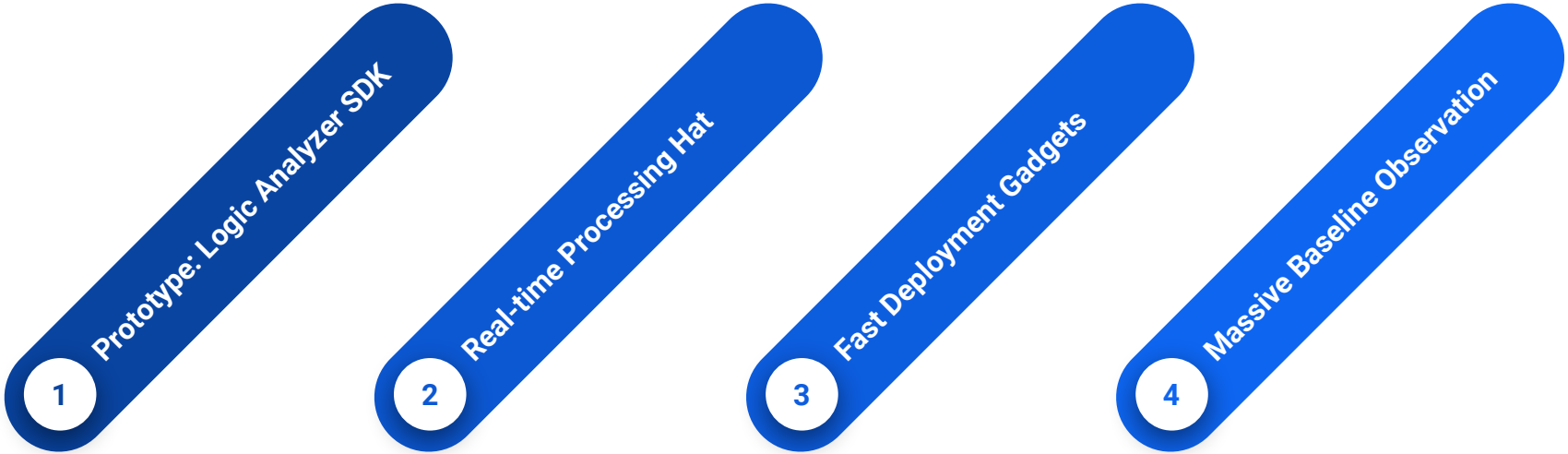
1. More data
2. More powerful algorithm
3. More machines



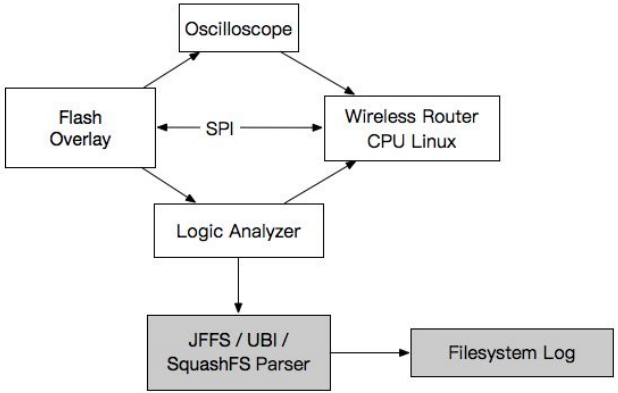


Towards Scalable Deployment of IoT Woodpecker

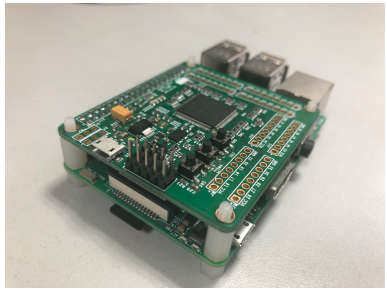
4 Steps


1
Prototype: Logic Analyzer SDK
2
Real-time Processing Hat
3
Fast Deployment Gadgets
4
Massive Baseline Observation

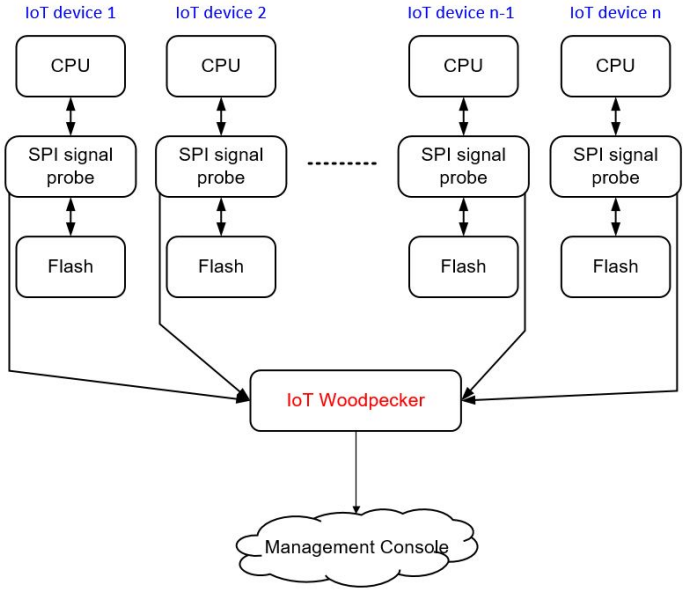
Scalable System Architecture



Prototype with Logic Analyzer SDK



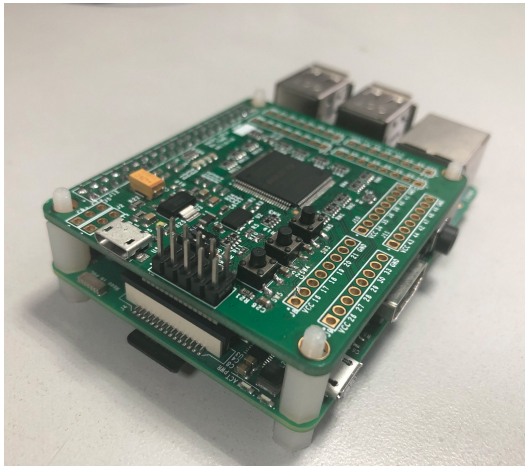
Real-time Processing Hat

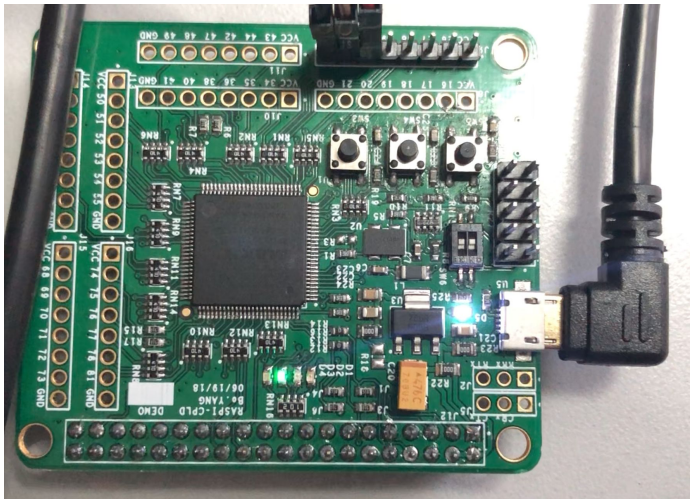


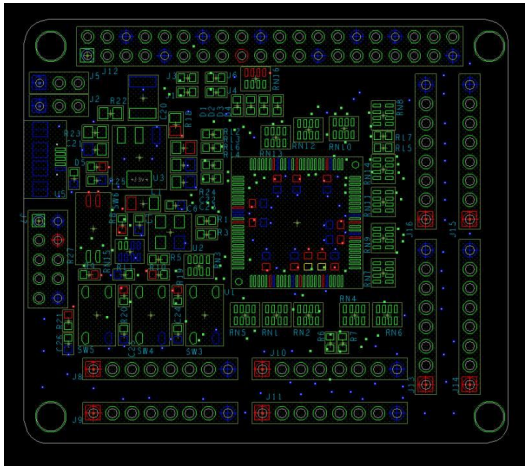
Massive Baseline Observation

Real-time Processing Daughterboard

- Custom Design with CPLD
- Feature:
  - Realtime On-board Analyze
  - Integrated as a Raspberry Pi Hat





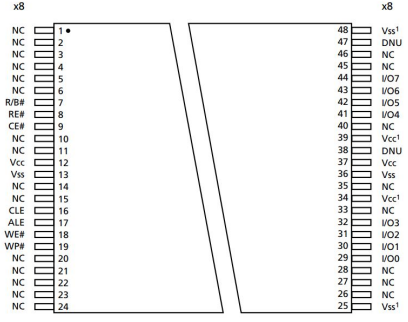


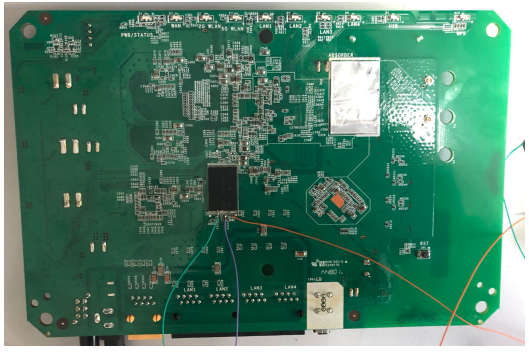
Real-time Processing - Use Case

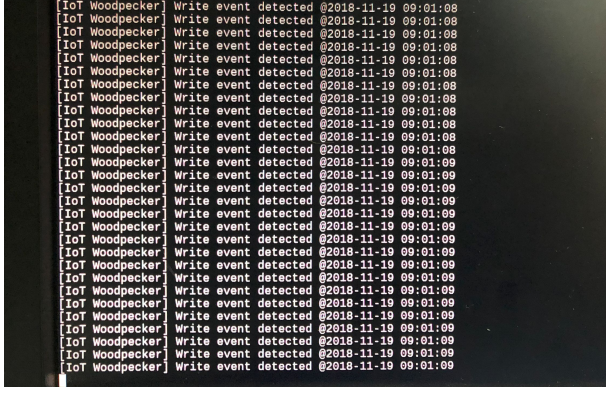
Table 1: Asynchronous Signal Definitions

Signal[1]	Type	Description[2]
ALE	Input	**Address latch enable:** Loads an address from I/O[7:0] into the address register.
CE#	Input	**Chip enable:** Enables or disables one or more die (LUNs) in a target.
CLE	Input	**Command latch enable:** Loads a command from I/O[7:0] into the command register.
RE#	Input	**Read enable:** Transfers serial data from the NAND Flash to the host system.
WE#	Input	**Write enable:** Transfers commands, addresses, and serial data from the host system to the NAND Flash.
WP#	Input	**Write protect:** Enables or disables array PROGRAM and ERASE operations.
I/O[7:0] (x8) I/O[15:0] (x16)	I/O	**Data inputs/outputs:** The bidirectional I/Os transfer address, data, and command information.
R/B#	Output	**Ready/busy:** An open-drain, active-low output that requires an external pull-up resistor. This signal indicates target array activity.
V_{CC}	Supply	**V_{CC}:** Core power supply
V_{SS}	Supply	**V_{SS}:** Core ground connection
NC	–	**No connect:** NCs are not internally connected. They can be driven or left unconnected.
DNU	–	**Do not use:** DNUs must be left unconnected.

Notes:
1. See Device and Array Organization for detailed signal connections.
2. See Asynchronous Interface Bus Operation for detailed asynchronous interface signal descriptions.

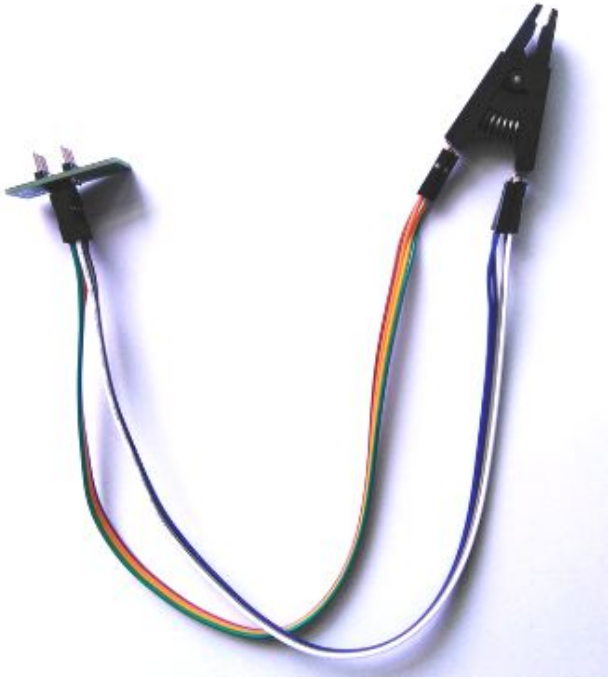






Live Demo

Fast Deployment Gadgets



Massive Baseline Observation

Movie: Blade Runner 2049 Baseline

Future Work

- Extend NVM interface to general electrical signal from test point
  - Analog Signal
  - Digital Signal: Intrusion Detection of Digital Signal Domain
    - I2C, I2S
  - Power Signal
    - Side Channel
- Automated Optical Inspection
  - Test Point/Chip Adaption
  - Hardware Supply Chain Risk

Takeaways

- IoT Woodpecker: Proposed a novel IDS for IoT devices, and verified on the real devices;
- State a hypothesis and verify it: the inevitability of memory accessing;
- Deep learning based artificial intelligence is introduced to detect anomalies;
- Roadmap of IoT Woodpecker.

Another finding by IoT Woodpecker

Hot Mic

November 28, 2018
10:45 am - 11:45 am
Conf Track 2
AUDITABLE & PROVABLE PRIVACY OF SMART SPEAKERS

Thank you