CS144r/244r
Network Design Project on
Secure and Intelligent Internet of Things
(Lowpower SoC)
2/5/2014

Instructor: Professor HT Kung
Harvard School of Engineering and Applied Sciences
1. We use classroom MD 119 starting today
2. Quiz results will be emailed today
3. Next set of questions due Sunday night (Feb 9) will be posted Saturday evening
   – They will cover materials in today’s class and readings for next Monday’s class. Readings for next Monday’s class will be posted by Friday evening
4. At the end of the semester, we will discard the bottom two quiz scores for each student. In this way, students will have some flexibility in working on these quizzes
5. We will hold a “Make Your Own Nest” information and organizational meeting next Monday after class for those students who are interested in starting their course projects earlier in this area. Time and place will be posted later this week. Everyone is welcome to attend the meeting
1. “Satya Nadella named Microsoft CEO” (2/4/2014)
   • “… but he will now need to prove he can complete Microsoft’s vision for **devices** and its ongoing **mobile push**”

   • AllJoyn is a key part of the AllSeen Alliance, which was formed in December 2013 by The Linux Foundation and industry leaders to drive an open, universal framework to connect devices and systems. AllJoyn was originally developed by and has been contributed to the Alliance by Qualcomm.
Recap: How Can A Thermostat Company Be Worth Billions?

1. Google bought Nest for $3.2 billion in cash (January 13, 2014)
   • "If your house is burning down you'll now get gmail ads for fire extinguishers," someone tweeted

2. We can speculate reasons behind the acquisition (see a good Wired article at [http://www.wired.com/business/2014/01/googles-3-billion-nest-buy-finally-make-internet-things-real-us/](http://www.wired.com/business/2014/01/googles-3-billion-nest-buy-finally-make-internet-things-real-us/))
   • What’s after smartphones is **wearables**
   • What’s after human-centric Web is info directly coming from **physical things** generated by sensors
   • Having a means to **bring a device to homes** is a big deal. In the past only a few industries such as carriers, cable operators and utility companies could put such boxes at home

3. Note that the Nest founder, Tony Fadell, led the teams at Apple which created the iPod and the iPhone. Making beautiful, intelligent and cloud-connected artifacts with the Apple-like DNA can get people really excited (emotion is powerful)
Recap: Making Your Own Nest

1. You can start your course project early by building a Nest-like device or better functionality (and have something to talk about when you go home for spring break)

2. You can use breadboarding or a small computer to connect off-the-shelf sensors

3. There are open source kits available. See, e.g., “Building an open source Nest” at http://blog.spark.io/2014/01/17/open-source-thermostat/

4. Then let’s experiment with various machine learning methods allowing user feedback to suit your target device applications

5. Please let TFs know if you are interested in attending an organizing meeting next Monday after class
Recap: Principle of Stereo Cameras: Deriving Depth from Left and Right Images

One major problem is to locate $x'_l$ and $x'_r$: **The correspondence problem**

Distinguishing textual features around the object will help. We can then use neighborhood block matching to locate $x'_l$ and $x'_r$.
Recap:
The Kinect Pose Estimation Pipeline

1. capture depth image

Infrared dots projected onto objects

2. infer body parts per pixel

3. infer body joint positions

4. track skeleton
Today’s Topic: Lowpower SoC

- We usually use **embedded systems** to implement IoT devices
  - Embedded systems are designed to do some specific task (e.g., routers, set-top boxes, factory controllers and MP3/video players) rather than be a general-purpose computer for multiple tasks
  - May have high reliability and real-time performance constraints that must be met, for reasons such as safety and usability. They often don’t have hard drives which could fail after a few year’s use
  - May have low performance requirements, allowing the system to be simplified to reduce costs and power consumption

- We usually use **System-on-Chip (SoC)** to make massively used embedded systems
  - CPU cores, flash memory, accelerators, GPU, DSP, etc. all on a single chip. Often license third-party IPs
  - Power and energy are primary design and management concerns
Readings on Lowpower SoC

1. Power Struggles: Revisiting the RISC vs. CISC Debate on Contemporary ARM and x86 Architectures, 2013

2. Example SoCs with varying capabilities/power consumptions and target applications
   - Arduino microcontroller (based on ARM, MIPS, etc.), Galileo/Edison (Intel’s Quark chip), and Snapdragon (Qualcomm SoC)
The CPU Debate (Past and Current)

- Current CPU competition landscape
  - Intel CPU dominates laptop, desktop and server markets
  - ARM CPU dominates mobile smartphone and tablet markets

- Players’ ambition
  - ARM hopes to enter Intel dominated markets
  - Intel hopes to increase its market share in mobiles and dominate emerging IoT markets

- There are two computer architectures: CISC (Complex Instruction Set Computers) vs. RISC (Reduced Instruction Set Computers). There was a debate in the 1980s. The debate was over ten years later when Intel CISC won over almost all RISC machines. (Intel had an advantage on supporting legacy x86 applications)

- For mobile and IoT, we now have two camps again:
  - Intel is CISC
  - ARM is RISC

- A natural question to ask now:
  - Can the so-called “complex” architecture (CISC) win the battle given that mobile and IoT devices are mostly small and low power?
Recall the Rationale for CISC

Complex instruction sets can:

• Ease the task of the compiler writer

• Improve execution efficiency, because complex sequences of operations can be implemented in microcode

• Provide support for sophisticated high-level programming languages (e.g., security functionalities)
Recall the Rationale for RISC

• Characteristics of RISC
  – One instruction per cycle
  – Register-to-register operation
  – Simple addressing modes (register addressing)
  – Simple instruction formats (fixed at word boundaries)

• Potential benefits of RISC
  – Can make optimizing compilers more effective (e.g., instruction pipelining)
  – The control unit has simpler design and implementation, and thus has smaller footprint on chip
What Role if Any Does RISC vs. CISC Play in This Power Struggle?

- Reading 1 argues that Instruction Set Architecture (ISA) being RISC or CISC does not matter for modern microprocessors.
- Instead, it is the intended performance objective for the SoC that matters.
- This is analogous to Maxwell Dworkin vs. Northwest Building. They both use the open-space architecture in the building center. But the former is more than economical than the latter in energy consumption. This results from different target usages.
Performance, Power, Energy and Trade-offs Between Them

• **Performance** is billions of instructions per second (BIPS)

• **Power** is the rate of consumption of energy, measured in watts, milliwatt, microwatts, etc.

• **Energy** is product of power and time

Using a high-performance SoC we complete a computation task earlier. But the required power is higher, and total energy used for the entire task duration may be lower or higher (usually higher!)

The **trade-off** is that we like short task completion time, but don’t like high energy consumption
Power-Performance and Energy-Performance Trade-offs

- Note **super-linear** trade-off curves (red)
- Where do we pay in power/energy for high performance under the same clock frequency?
  - Larger caches
  - Larger branch prediction tables
  - Faster TLB (Translation Lookaside Buffer)
  - Wider instruction and data paths
  - OOO execution
  - Larger active chip area (and higher leakage)
- These features have little to do with CISC or RISC
- Instead, the issue is about how to reduce significantly all these energy consuming features
IoT Devices

1. Major Players: Raspberry Pi, Arduuno Family, Intel Galileo/Edison, BeagleBone
2. Based on MIPS, x86, ARM
3. Low-end, Mid-end and Hi-end Applications
4. All have networking: WiFi, Bluetooth, Ethernet
## Example IoT Devices (1/2)

<table>
<thead>
<tr>
<th></th>
<th>Galileo</th>
<th>Edison</th>
<th>Arduino Yûn</th>
<th>Arduino TRE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Release Date</strong></td>
<td>Early 2014</td>
<td>Late 2014</td>
<td>Late 2013</td>
<td>Mid 2014</td>
</tr>
<tr>
<td><strong>Form Factor</strong></td>
<td>4.2&quot; x 2.8&quot;</td>
<td>0.9&quot; x 1.2&quot;</td>
<td>2.7&quot; x 2.1&quot;</td>
<td>2.4&quot; x 2.4&quot;</td>
</tr>
<tr>
<td><strong>Processor</strong></td>
<td>Quark SoC X1000</td>
<td>Quark SoC X1000</td>
<td>Atheros AR9331</td>
<td>ARM Cortex-A8</td>
</tr>
<tr>
<td><strong>ISA</strong></td>
<td>x86 @400 Mhz</td>
<td>x86 @2x400 Mhz</td>
<td>MIPS @400MHz</td>
<td>ARM @ 1GHz</td>
</tr>
<tr>
<td><strong>RAM</strong></td>
<td>256 MB DRAM</td>
<td>?</td>
<td>64 MB DDR2</td>
<td>512 MB DDR</td>
</tr>
<tr>
<td><strong>Microcontroller</strong></td>
<td>No dedicated microcontroller</td>
<td>No dedicated microcontroller</td>
<td>ATmega32u4 @ 16Mhz</td>
<td>ATmega32u4 @ 16Mhz</td>
</tr>
<tr>
<td><strong>OS</strong></td>
<td>Yocto Linux</td>
<td>?</td>
<td>Lilino Linux (OpenWRT)</td>
<td>Linux</td>
</tr>
<tr>
<td><strong>Networking/Connectivity</strong></td>
<td>Ethernet, USB</td>
<td>WiFi, Bluetooth</td>
<td>Ethernet, USB</td>
<td>Ethernet, USB</td>
</tr>
<tr>
<td></td>
<td>Micro SD</td>
<td>Micro SD</td>
<td>Micro SD</td>
<td>Micro SD</td>
</tr>
<tr>
<td><strong>IDE</strong></td>
<td>Galileo Sketch IDE</td>
<td>?</td>
<td>Arduino Sketch IDE Loaded with Tempo for web-based resources &amp; services</td>
<td>Arduino Sketch IDE</td>
</tr>
<tr>
<td><strong>Max Current Draw</strong></td>
<td>800 mA</td>
<td>?</td>
<td>50 mA</td>
<td>?</td>
</tr>
</tbody>
</table>
### Example IoT Devices (2/2)

<table>
<thead>
<tr>
<th>Peripherals</th>
<th>Galileo</th>
<th>Edison</th>
<th>Arduino Yûn</th>
<th>Arduino TRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 Digital I/O</td>
<td>14 Digital I/O</td>
<td>20 Digital I/O</td>
<td>14 Digital I/O</td>
<td>7 PWM Channels</td>
</tr>
<tr>
<td>6 PWM (shared)</td>
<td>6 PWM (shared)</td>
<td>7 PWM Channels</td>
<td>7 PWM Channels</td>
<td>7 PWM Channels</td>
</tr>
<tr>
<td>6 Analog Input</td>
<td>6 Analog Input</td>
<td>12 Analog Input</td>
<td>12 Analog Input</td>
<td>6+6 (shared) Analog Input</td>
</tr>
<tr>
<td>Support Arduino Shields</td>
<td>Support Arduino Shields</td>
<td>Support Arduino Shields</td>
<td>Support Arduino Shields</td>
<td>Support Arduino Shields</td>
</tr>
<tr>
<td>Mini PCI Express</td>
<td>Mini PCI Express</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UART RS-232 serial port</td>
<td>UART RS-232 serial port</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Galileo</th>
<th>Edison</th>
<th>Arduino Yûn</th>
<th>Arduino TRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to make Sketch interact with other processes running on GNU/Linux</td>
<td>Easy to make Sketch interact with other processes running on GNU/Linux</td>
<td>Dedicated microcontroller</td>
<td>Dedicated microcontroller</td>
<td>Security</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application Area</th>
<th>Galileo</th>
<th>Edison</th>
<th>Arduino Yûn</th>
<th>Arduino TRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Higher Security (Intel Quark)</td>
<td>Higher Security (Intel Quark)</td>
<td>?</td>
<td>?</td>
<td>Mid-End</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>Lo-End</td>
<td>Hi-End</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Examples</th>
<th>Galileo</th>
<th>Edison</th>
<th>Arduino Yûn</th>
<th>Arduino TRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>HVAC system with security on a rooftop</td>
<td>HVAC system with security on a rooftop</td>
<td>Mimo Baby Monitor</td>
<td>Networks of home sensors</td>
<td>OpenCV Computer Vision/Machine Learning</td>
</tr>
</tbody>
</table>
A Fancy SoC: Snapdragon

Multicore CPU
- 1.5 – 2.5 GHz
- ARM ISA
- L1/L2 cache
- SIMD

Networking and Wireless
- WiFi, 3G/4G, LTE, GPS, Bluetooth 4.0

GPU
- Unified shader arch
- Programmable OpenGL

DSP accelerator
- Interleaved multi-thread programming toolchain

Low power techniques
- Dynamic frequency voltage scaling
- Asynchronous symmetrical multi-processor system
- Average Power Tracking (for COMM)
Shields (Add-ons)

1. Xbee Zigbee (300 ft)
2. Motor Control
3. IR remote
4. VGA camera
5. LCD, OLED, Touch Screen
6. Battery
7. GPS
8. Random Number Generator (RNG)
9. Pressure, temperature and relative humidity, light motion, temperature, sound sensors
10. GSM, WiFi, Bluetooth, RFID
1. All can be used for IoT projects
2. For applications that require more processing power, you should go with TRE (Beaglebone Black is a comparable alternative)
3. If you need to use UART RS-232 Serial Interface or PCI Express, Galileo is the way to go
4. Yûn comes with Wi-Fi which is more convenient
5. Yûn and Edison are suitable for low-power wearable devices.
6. Galileo has security support (Intel Quark)
7. Project Ideas:
   http://www.instructables.com/id/Arduino-Projects/
Project Examples

Text clock

LED light that reflects Twitter trends

Jacket for Cyclists