Internet of Things (IoT)

- IoT system applications
- IoT system architectures

References: Wolf Text: Chapter 8
ARM SoC/IoT Presentations

ARM: Making Things Smart, Connected and Interactive

26 Billion installed units by 2020*
Internet of Things (IoT)

**What is it?**

- “The Internet of Things (IoT) is the interconnection of uniquely identifiable embedded computing devices within the existing Internet infrastructure.” [wikipedia.org](http://wikipedia.org)

- Buzzword, trend, convenient categorisation, industrial and consumer
Internet of Things (IoT)

Why Now?

- Embedded chips are becoming:
  - Cheaper (<50c)
  - Smaller (<1mm²)
  - Lower power (μW)
  - Commoditised HW and SW
- Communication is growing faster (broadband)
- New socio-economic demands (globalisation, competition, mobility)
Internet of Things (IoT)

Socio-Economic Benefits

• Automation (higher productivity)
• Smart monitoring, control and maintenance (higher efficiency, lower cost, higher quality, better optimisation/outcomes)
• Better safety (early warning)
• Higher responsiveness (dynamic response to varying demands)
• Huge and varied applications in industry, agriculture, health, transport, infrastructure, smart living, consumer etc.
IoT system applications

• Soft real-time networked embedded system.
  • Input devices: tags, sensors, etc.
  • Output devices: motor controllers, displays, etc.

• Examples:
  • Computer-readable identification code for objects.
  • Appliances controlled by cell phone interface.
  • Sensor network with analytics.
Devices

• People:
  • Implanted devices in the body.
  • Wearable devices on the body.
  • Environmental devices outside the body.

• Objects:
  • Interior: temperature sensor, etc.
  • Exterior: RFID, etc.
  • Environmental: camera, motion sensor, etc.

https://www.meddeviceonline.com/doc/how-revolutionizing-healthcare-0001

https://communicationandmediastudies.wordpress.com/2012/10/24/the-internet-of-things/
Connecting the Physical and Digital Worlds

Things ("Edge" Devices)

- Integrated sensors, memory and processing
- Low power systems
- Little Data

Wireless Network
- High throughput networks
- Low power wireless networks

Cloud
- High performance efficient servers
- High capacity storage
- Software as a service
- Big Data
IoT system architectures

- **Edge**: I/O devices.
- **Cloud**: centralized processing.
- **Smart appliance** = connected appliance + network + UI.
IoT system architectures, cont’d.

- **Monitoring system** = sensors + network + database + dashboard.
IoT system architectures, etc.

- **Control system** = sensors + database + controller + actuator.
Things: Basic Building Functional Blocks

- Sense
- Compute
- Control
- Store
- Communicate
Unlock a greater potential with custom SoCs

From PCB to custom SoC

Increase margins by reducing
- Cost
- Complexity
- Size

Enhance designs with greater
- Efficiency
- Reliability
- Differentiation
- IP protection
Cortex-M: Scalable, compatible and trusted

- **Cortex-M7**: Maximum performance, control and DSP
- **Cortex-M3**: Performance efficiency
- **Cortex-M4**: Mainstream control and DSP
- **Cortex-M0**: Lowest cost, low power
- **Cortex-M0+**: Highest energy efficiency
- **Cortex-M23**: TrustZone in smallest area, lowest power

**ARMv6-M**

**ARMv7-M**

**ARMv8-M**

**ARM TrustZone**

- High performance
- Performance efficiency
- Lowest power & area
ARM Cortex-M0 DesignStart Processor

• **Subset of the full ARM Cortex-M0**
  • Low gate count, 32-bit processor, 3-stage pipeline
  • Implements ARMv6-M architecture
  • Can achieve around 0.9 DIPS/MHz

• **Provided as synthesizable Verilog model**
  • CPU contained in top-level macro-cell “CORTEXM0DS” (instantiated in the SoC system model) and submodule “cortexm0ds_logic” (pre-configured, obfuscated)
  • Top-level macro-cell implements memory and system bus interface compatible with *AMBA 3 AHB-Lite* specification, including interrupt and event inputs, 3 status outputs, and an event output.

• **DesignStart Kit** includes:
  • Simulation testbench, a set of AHB-Lite peripherals, example SoC systems
The “THING”
Sense/Compute/Control/Store/Communicate

Sensor: Accelerometer
Sensor: Ambient Light
Low-power Cortex-M0 SoC
User Interface
Bluetooth Radio

The “NETWORK”
Bluetooth

The “CLOUD”
Server
Software, Storage

• Periodically sense, encrypt, send ambient light and acceleration data to “The Cloud”.
• Based on ARM SoC LiB and Cortex-M0 CPU.
• Platform: FPGA board
  (Digilent Nexys4 DDR, Numato Labs Mimas V2)
IoT SoC Application

• **Periodically capture sensor data**
  • Read ambient light sensor data
  • Read X-axis/Y-axis/Z-axis acceleration data
  • Sample at 1Hz frequency *(timer interrupt-driven)*

• **Encrypt sensor data**
  • Tiny Encryption Algorithm (TEA)
  • Encrypt before sending *(for debug - decrypt back to original data if Switch 1 on)*

• **Transmit data via Bluetooth to server**
  • Simulated wireless network and “Cloud” server
  • For debug (Switch 0 on) transmit via hard-wired USB to server

• **Display sensor data in server terminal window**
IoT Demo Sensors & Communication

• **ADXL362 3-axis Accelerometer**
  • 12-bit X/Y/Z axis values + 12-bit temperature
  • On Nexys4 DDR board
  • SPI interface

• **PmodALS Ambient Light Sensor**
  • Vishay Semiconductor TEMT6000X01 ambient light sensor
  • Texas Instruments ADC081S021 analog to digital converter
  • SPI interface

• **PmodBT2 Bluetooth Interface**
  • Roving Networks RN-42 Bluetooth (2.1, 2.0, 1.2, 1.0)
  • UART interface
IoT SoC Hardware

- ARM Cortex-M0 “Design Start” CPU
- Program and data in distributed/block RAM in FPGA
- Peripherals: basic I/O, timer, UART, SPI (all except SPI in the SoC LiB)
  - Sensors accessed via SPI
  - Wireless and wired communication via UARTs
- CPU and peripherals interconnected via *AHB-Lite* bus

```
System on Chip

ARM Cortex-M0 Processor

Control signals

32-bit Address bus

32-bit Data bus

Program Memory (On-Chip)

SPI Peripherals

Timer Peripheral

UART Peripherals

GPIO Peripheral

7-segment Peripheral

PmodALS Light Sensor

ADXL362 Accelerometer

UART to USB Converter

PmodBT2 Bluetooth Radio

LEDs Switches

7-Segment Display
```
Security important as more embedded devices become connected

Even the smallest of devices need to
- Safely store and process secrets
- Have secure communications (i.e., encryption)
- Offer trust in the integrity of the device and its software
- Be able to isolate trusted resources from non-trusted
- Reduce attack surface of key components

… without compromising on latency, determinism or footprint.
ARM TrustZone Technology

Bringing ARM security extensions to the embedded world

• Optional security extension for the ARMv8-M architecture
  • Security architecture for deeply embedded processors
  • Enables containerisation of software
  • Simplifies security assessment of embedded devices.

• Conceptually similar and compatible with existing TrustZone technology
  • New architecture tailored for embedded devices
    ▪ Preserves low interrupt latencies of Cortex-M
    ▪ Provides high performance cross-domain calling.
ARMv8-M Additional States

Existing handler and thread modes mirrored with secure and non-secure states

- Secure and Non-Secure code run on a single CPU
  - For efficient embedded implementation.

- Secure state for trusted code
  - New Secure stack pointers for robust operation
  - Addition of stack-limit checking.

- Dedicated resources for isolation between domains
  - Separate memory protection units for Secure and Non-secure
  - Private SysTick timer for each state.

- Secure side can configure target domain of interrupts.
Secure state view view

- Secure memory view permits access to Secure Flash, RAM, and peripherals.

- Load/store access to all regions is possible from Secure state.

- Security of regions can be configured using the Security Attribution Unit (SAU).
ARMv8-M Interrupt Security

Subject to priority, Secure can interrupt Non-secure and vice versa
- Secure can boost priority of own interrupts
- Uses current stack pointer to preserve context.

Uses ARMv7-M exception stacking mechanism
- Hardware pushes selected registers.

Non-secure interruption of Secure code
- CPU pushes all registers and zeroes them
  - Removes ability for Non-secure to snoop Secure register values.
Security Defined by Address

All transactions from core and debugger checked

- All addresses are either Secure or Non-secure.

- Policing managed by Secure Attribution Unit (SAU)
  - Internal SAU similar to MPU
  - Supports use of external system-level definition
    - E.g. based on flash blocks or per peripheral.

- Banked MPU configuration
  - Independent memory protection per security state.

- Load/stores acquire NS attribute based on address
  - Non-secure access attempts to Secure address = memory fault.
Cross-Domain Function Calls

An assembly code level example

- Guard instruction (SG) polices entry point
  - Placed at the start of function callable from non-secure code.
- Non-secure $\rightarrow$ secure branch faults if SG isn’t at target address
  - Can’t branch into the middle of functions
  - Can’t call internal functions.
- Code on Non-secure side identical to existing code.
Cortex-M23: Imagine the possibilities

- 32-bit performance
- TrustZone for ARMv8-M
- Long battery life
- Small area, low cost
Cortex-M23 enhancements over Cortex-M0+

Cortex-M0+
- NVIC (max 32 IRQs)
- MPU (PMSAv6)
- AHB Lite
- WIC
- Fast I/O bus
- MTB
- Serial wire / JTAG
- ARMv6-M

Cortex-M23
- TrustZone
- Stack limit checking
- Hardware divide
- Exclusive memory accesses
- Enhanced debug
- ETM
- NVIC (max 240 IRQs)
- MPU (PMSAv8)
- AHB5
- WIC
- Fast I/O bus
- MTB
- Serial wire / JTAG
- ARMv8-M baseline

New or updated

CoreMark®/MHz
- 2.46
- 2.50

DMIPS/MHz
- 0.95
- 0.98

Cortex-M23 enhancements over Cortex-M0+
Addressing diverse embedded and IoT opportunities

- Automotive
- Industrial
- Smart lighting
- Agriculture
- Wearables
- Smart building
- Healthcare
- Home automation

Cortex-M33

- 32-bit performance
- TrustZone for ARMv8-M
- Configurability, extensibility
- Digital signal processing
Cortex-M33 enhancements over Cortex-M4

<table>
<thead>
<tr>
<th>Cortex-M4</th>
<th>Cortex-M33</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETM</td>
<td>TrustZone</td>
</tr>
<tr>
<td>NVIC (max 240 IRQs)</td>
<td>Stack limit checking</td>
</tr>
<tr>
<td>MPU (PMSAv7)</td>
<td>Co-processor interface</td>
</tr>
<tr>
<td>AHB Lite</td>
<td>Enhanced debug</td>
</tr>
<tr>
<td>FPU</td>
<td>MTB</td>
</tr>
<tr>
<td>SIMD/ DSP</td>
<td>ETM</td>
</tr>
<tr>
<td>WIC</td>
<td>NVIC (max 480 IRQs)</td>
</tr>
<tr>
<td>Serial wire / JTAG</td>
<td>MPU (PMSAv8)</td>
</tr>
<tr>
<td>ARMv7-M</td>
<td>AHB5</td>
</tr>
<tr>
<td></td>
<td>FPU</td>
</tr>
<tr>
<td></td>
<td>SIMD/ DSP</td>
</tr>
<tr>
<td></td>
<td>WIC</td>
</tr>
<tr>
<td></td>
<td>Serial wire / JTAG</td>
</tr>
<tr>
<td></td>
<td>ARMv8-M mainline</td>
</tr>
</tbody>
</table>

- New or updated

<table>
<thead>
<tr>
<th></th>
<th>CoreMark®/MHz</th>
<th>DMIPS/MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cortex-M4</td>
<td>3.40</td>
<td>1.25</td>
</tr>
<tr>
<td>Cortex-M33</td>
<td>3.86</td>
<td>1.50</td>
</tr>
</tbody>
</table>

+13% +20%
IoT systems and networks

• OSI model for networks.
• Internet protocol.
• IoT networking concepts.
• Example networks:
  • Classic Bluetooth, Bluetooth Low Energy.
  • 802.15.4 and Zigbee.
  • Wi-Fi.
Network Abstractions: OSI model

• International Standards Organization (ISO) Open Systems Interconnection (OSI) to describe networks:
  • 7-layer model.
• Standard way to classify network components and operations.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>end-use interface</td>
</tr>
<tr>
<td>Presentation</td>
<td>data format</td>
</tr>
<tr>
<td>Session</td>
<td>application dialog control</td>
</tr>
<tr>
<td>Transport</td>
<td>connections</td>
</tr>
<tr>
<td>Network</td>
<td>end-to-end service</td>
</tr>
<tr>
<td>Data Link</td>
<td>reliable data transport</td>
</tr>
<tr>
<td>Physical</td>
<td>mechanical, electrical</td>
</tr>
</tbody>
</table>
OSI layers

- **Physical**: connectors, bit formats, etc.
- **Data link**: error detection and control across a single link (single hop).
- **Network**: end-to-end multi-hop data communication.
- **Transport**: provides connections; may optimize network resources.
- **Session**: services for end-user applications: data grouping, checkpointing, etc.
- **Presentation**: data formats, transformation services.
- **Application**: interface between network and end-user programs
PHY and MAC

• PHY = physical layer.
  • Circuitry to transmit and receive bits.

• MAC = media access control.
  • Provides link-level services.
Internet Protocol (IP)

Internet = network of networks: transports data from one network to another.
IoT networking concepts

• Edge device may not run IP protocol.
  • IP connection may be provided by hub or gateway.
  • Non-IP networks are known as edge networks.

• Ad hoc network is self-organized---not set up by system administrator.

• Ad hoc network services:
  • Authentication of eligibility to join network.
  • Authorization for access to given pieces of information on the network.
  • Encryption and decryption.
Network topologies

- Star
- Tree
- Mesh
Routing

- Routing discovery determines routes between source/destination pairs.
- Routing is driven by routing tables at the nodes.
QoS

• Many networks support synchronous and asynchronous communication.
  • Asynchronous: data records, etc.
  • Synchronous: voice, etc.

• Quality-of-service (QoS): bandwidth and periodicity characteristics.

• Admission control ensures that network can handle the QoS demands of a request.
Synchronization and beacons

• Many network operations require nodes to be synchronized.
• Synchronization can be performed using beacon.
  • Beacon transmission marks the beginning of a communications interval.
Communications energy

• Communications energy is a large part of node energy consumption.
• Comm energy consumption depends on many factors and parameters.
  • Generally evaluated for a set of use cases.
• We can use power state machine to model communications energy cost.
Communications power state machine example

<table>
<thead>
<tr>
<th>step</th>
<th>state</th>
<th>time</th>
<th>energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>sleep</td>
<td>1 ms</td>
<td>1 nJ</td>
</tr>
<tr>
<td>2</td>
<td>idle</td>
<td>10 μs</td>
<td>0.5 nJ</td>
</tr>
<tr>
<td>3</td>
<td>receive</td>
<td>50 μs</td>
<td>1.25 nJ</td>
</tr>
<tr>
<td>4</td>
<td>transmit</td>
<td>50 μs</td>
<td>1.75 nJ</td>
</tr>
<tr>
<td>5</td>
<td>receive</td>
<td>50 μs</td>
<td>1.25 nJ</td>
</tr>
<tr>
<td>6</td>
<td>transmit</td>
<td>50 μs</td>
<td>1.75 nJ</td>
</tr>
</tbody>
</table>

total = 7.5 nJ
Bluetooth

• Introduced in 1999, originally for telephony applications.
• Classic Bluetooth operates in instrumentation, scientific, and medical (ISM) band in the 2.4 GHz range.
• Bluetooth networks organized as piconet.
  • One master, several slaves.
  • Slave can be active or parked.
  • A device can be a slave on several networks simultaneously.
Bluetooth stack

• Transport protocol:
  • Radio, baseband layer, link manager, logical link control and adaptation protocol (L2CAP).

• Middleware:
  • RFCOMM for serial port, service discovery protocol, Internet Protocol, IrDA, etc.

• Applications.
Bluetooth protocol

• Every Bluetooth device has a 48-bit Bluetooth Device Address.
• Every device has a Bluetooth clock.
• Transmissions alternate between master and slave directions.
• Two types of packets:
  • Synchronous connection-oriented (SCO) packets for QoS-oriented traffic.
  • Asynchronous connectionless (ACL) packets for non-QoS traffic.
  • SCO traffic has higher priority than ACL packets.
Bluetooth Low Energy

• Designed for very low energy operation such as button-sized battery.
  • Goal: minimize radio on-time.
• Part of Bluetooth standard but deviates from Classic Bluetooth in several ways.
• Advertising transmissions can be used to broadcast, discover devices, etc.
• Connections can be established.
• Attribute Protocol Layer allows devices to create application-specific protocols.
• Generic Attribute Profile Layer (GATT) defines basic attributes for all BLUE devices.
• Pairing devices uses a short-term key to send a long-term key.
  • Bonding: storing long-term key in device database.
  • Optional data encryption using AES.
Bluetooth Low Energy (BLE)

Link-level state machine

- **scanning**
  - active
  - passive
- **advertising**
- **standby**
- **initiating**
- **connected**
  - master
  - slave
802.15.4 and ZigBee

• 802.15.4 defines MAC and PHY layers.
  • Supports full-function and reduced-function devices.
  • Either star or peer-to-peer topology.
  • Communication performed using frames.
  • Optional superframe provides a beacon mechanism and QoS.

• ZigBee is a set of application-oriented standards.
  • NWK layer provides network services.
  • APL layer provides application-level services.
  • Supports many different topologies.
Wi-Fi

• Originally designed for portable and mobile applications.
  • Has been adapted for lower-energy operation.

• Supports ad hoc networking.

• Network provides a set of services:
  • Distribution of messages from one node to another.
  • Integration delivers messages from another network.
  • Association relates a station to an access point.
IoT Systems Databases

- Database holds data about devices, helps to analyze data.
- Relational database management system:
  - Domain1 X domain2 X ... -> Range.
- Database organized into records or tuples:
  - Attribute: table column.
  - Record: table row.
- One column is the primary key---uniquely identifies a record.
### Database example

#### record

<table>
<thead>
<tr>
<th>name</th>
<th>id (primary key)</th>
<th>address</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>door</td>
<td>234</td>
<td>10.113</td>
<td>binary</td>
</tr>
<tr>
<td>refrigerator</td>
<td>4326</td>
<td>10.117</td>
<td>signal</td>
</tr>
<tr>
<td>table</td>
<td>213</td>
<td>11.039</td>
<td>MV</td>
</tr>
<tr>
<td>chair</td>
<td>4325</td>
<td>09.423</td>
<td>binary</td>
</tr>
<tr>
<td>faucet</td>
<td>2</td>
<td>11.324</td>
<td>signal</td>
</tr>
</tbody>
</table>

#### device_data

<table>
<thead>
<tr>
<th>signature (primary key)</th>
<th>device</th>
<th>time</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>256423</td>
<td>234</td>
<td>11:23:14</td>
<td>1</td>
</tr>
<tr>
<td>252456</td>
<td>4326</td>
<td>11:23:47</td>
<td>40</td>
</tr>
<tr>
<td>663443</td>
<td>234</td>
<td>11:27:55</td>
<td>0</td>
</tr>
</tbody>
</table>
IoT Management - Timewheels

- Used to manage timing of events in the system.
- Timewheel is a time-sorted set of events.
  - Event placed in proper spot in timewheel queue upon arrival.
  - When current time is equal to time of event at head, event is processed.

8:00 AM toaster on
7:45 AM lights on
7:30 AM radio on

7:28 AM

head pointer
Timewheel state diagram

get next event \[\rightarrow\] time = current_event.time() \[\rightarrow\] process event

get next event \[\Rightarrow\Rightarrow\] process event
Example: smart home

• Performs a variety of services:
  • Remote or automatic operation of lights and appliances.
  • Energy and water management.
  • Activity monitoring.

• Activity monitoring can help elderly, people with special needs:
  • Reports on daily activities.
  • Alerts for out-of-the-ordinary activity.
  • Recommendations.
Example smart home

• Cameras can identify resident and their activity.
• Faucet, door sensors can identify activity but not who performs the activity.
Use case: light control
Smart home object diagram

- sensor
- hub
- actuator
- console
- timewheel
- event queue
- database
- event queue
- event

The diagram illustrates the relationships between the components:
- Sensor connects to hub.
- Hub connects to timewheel.
- Timewheel connects to database and event queue.
- Event queue connects to event.
- Console is connected to timewheel.