Embedded systems engineering
Distributed real-time systems

David Kendall

Introduction

Internet of Things (IoT)

- billions of interconnected devices - household appliances, medical devices, industrial controllers, automobiles, cloud servers, mobile phones etc. - offering opportunities for monitoring, control and big data analysis

Organisation of the lecture

- IoT applications are complex, integrating many components:
  - hardware platforms, operating systems, software development tools, languages, middleware, applications, communication protocols, and data representations.
- Approaches to IoT are many and various; no definitive answers yet in any areas of development.
- This lecture describes a simple but complete IoT system which
  - illustrates, explains and clarifies the many components and their interactions;
  - offers a "strawman" proposal – definitely not the final word.

IoT opportunities

Projections for 2020, according to (Gartner, 2013) ...

- 26 billion connected devices in the IoT
- $>$ \$300 billion incremental revenue from IoT services
- growth in IoT will far exceed growth in other connected devices, e.g. the number of PCs, smartphones and tablets will reach about 7.3 billion units

Why this growth in IoT?

- ...because we want it
  - 'things' can have greater functionality and become more 'intelligent'
  - 'things' can be managed more easily
  - 'things' can provide us with more information
- ...and because we can
  - embedded chips are becoming: cheaper, smaller, lower power
  - communication is becoming faster

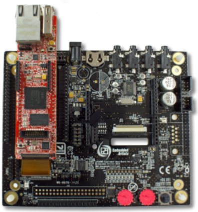
The first 'Thing'?

The Trojan Room Coffee Machine

- University of Cambridge Computer Laboratory - 1991
  - Camera pointed at coffee machine (no point walking all the way to the machine if it's empty!)
  - Wired to Acorn Archimedes with video capture board
  - Communication: MSNL (Multi-Service Network Layer) over ATM
  - Server requested a frame every few seconds (software by Paul Jardetzky)
  - Client acquired updates from the server (about 3 per minute) (software by Quentin Stafford-Fraser)

- Move to WWW - 1993
  - Daniel Gordon and Martyn Johnson
  - The birth of the Internet of Things

A new(-ish) ‘Thing’?

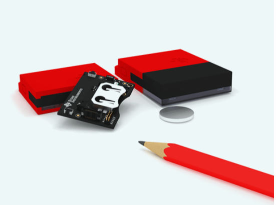


ARM Embedded Systems Education Kit

- NXP LPC4088 QSB - 120 MHz ARM Cortex M4; RAM: 32 MB SDRAM + 96 KB on-chip SRAM; Flash: 8 MB QSPI + 512 KB on-chip; ROM: 4 KB on-chip E2PROM; 4 LEDs, push button, USB, Ethernet, RF connectors
- Experiment base board - RGB LED, joystick, accelerometer, temperature sensor, potentiometer, 4.3 inch (480x272 pixel) TFT LCD
- Total cost - approx. £95.00
- See (Embedded Artists, 2016)

Some other ‘Things’

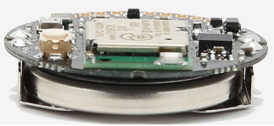
TI SensorTag



(Texas Instruments, 2015)

- 48MHz TI CC2650 SOC with ARM Cortex M3 + BLE, Zigbee, 6LoWPAN;
- Sensors: infrared and ambient temperature; ambient light; humidity; barometric pressure; 9-axis motion tracking - accelerometer, gyroscope, compass; magnetic proximity; microphone
- Cost approx. £20

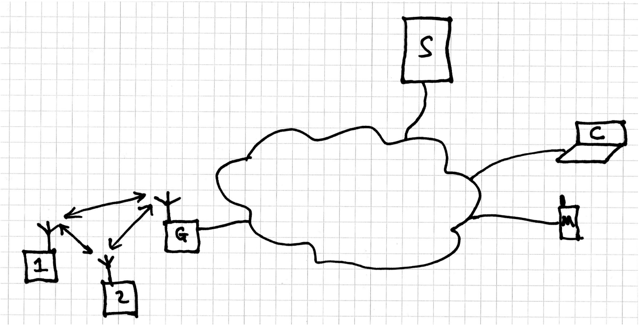
MetaWear C



(MBIENTLAB, 2016)

- Nordic nRF51822 SOC with ARM Cortex M0 CPU + BLE
- Sensors: BMI160 accelerometer + gyroscope; temperature sensor
- Cost approx. £35

Demonstration configuration



G	gateway	S	server
1	sensor node	C	computer
2	sensor node	M	mobile device

Demonstration functionality

Sensor Node

Monitor: Sample the sensors on experiment baseboard at 5 Hz

- Accelerometer [X,Y,Z]
- Potentiometer
- Temperature

Control: Accept commands via the gateway

- LEDs: turn on and off
- Display: change background colour; print message

Gateway

- Relay data from sensor nodes to server
- Relay commands from server to sensor nodes

Server

- Relay sensor node data from gateway to web clients
- Relay commands from web clients to gateway

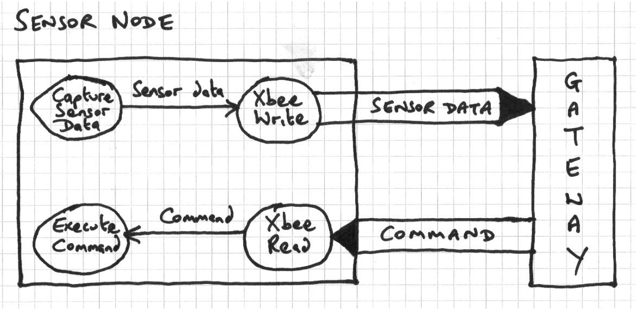
Browser

- Receive sensor node data from server; display it to user
- Receive commands from user; send to server

Demonstration

LET'S SEE A DEMONSTRATION

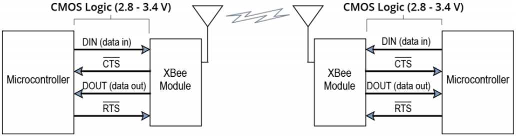
How it works: Sensor Node



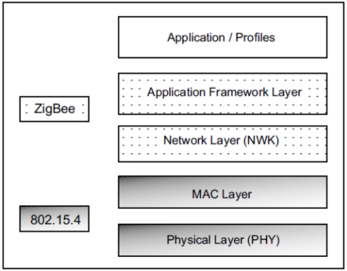
- LPC4088QSB + Xbee ZB RF module, managing an experiment baseboard
- Running a simple, multi-tasking RTOS (uC/OS-II)
- Essentially, 4 simple uC/OS-II tasks running concurrently
- All devices handled using the mbed libraries + some extensions from Embedded Artists for the experiment baseboard.

Xbee ZB RF module

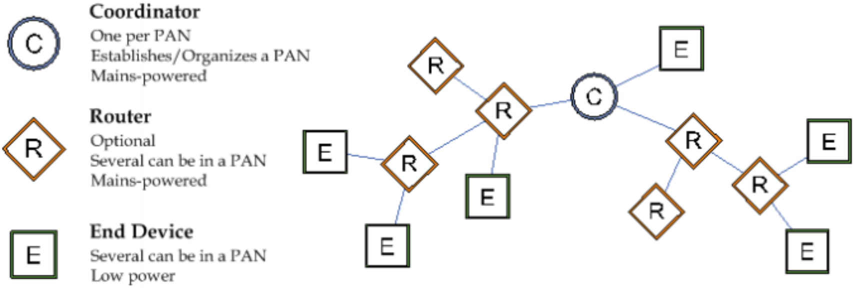
- RF module by Digi International – ISM 2.4 GHz band
- Range: up to 40 m indoors/urban; up to 120 m outdoor line-of-sight
- Low power: TX peak current and RX current 40 mA (@3.3V); power down current $< 1\mu A$
- Implements ZigBee for wireless mesh networking (WPAN), see (ZigBee Alliance, 2016)
- Data throughput: variable, from 35 kb/s to 5 kb/s



(Digi International, 2015)



ZibBee Network



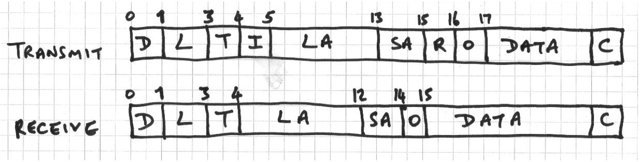
(Digi International, 2015)

Data Format

Sensor data is represented simply using JSON, e.g.

```
{
  "type":"DATA","id":"SN01","ax":1,"ay":2,"az":24,
  "pt":58,"tm":24.2
}
```

It's packed into an Xbee transmit frame and received at the gateway in a receive frame:



D – Delimiter (0x7E), L – Length, T – Frame type (0x10 Transmit, 0x90 Receive), I – Identifier, LA – Long address, SA – Short address, R – Broadcast radius, O – Options, DATA – JSON DATA, C – Checksum

How it works: Gateway

- Hardware:
  - Raspberry Pi Model B (revision 0002), single-core 700 MHz BCM2835 (ARM11), 256 MB RAM + Xbee ZB RF Module + WiFi dongle
- Software:
  - PyPy (PyPy, 2016)
    - fast Python implementation with JIT compiler
  - Twisted (Twisted Matrix Labs, 2016)
    - open source, event-driven networking engine written in Python
  - Autobahn|Python (Tavendo GmbH, 2015)
    - high-performance, fully asynchronous, scalable implementation of Websockets
- Implementation:
  - Asynchronously receive RF data via serial link; when a complete frame is received, extract the JSON data and send it to the server via a websocket
  - Receive commands from the server via a websocket and send via the serial link to RF module for transmission to the sensor node.

The Websocket Protocol

- provides a bi-directional communication channel over a single TCP connection
- the initiating handshake is an HTTP Upgrade request
- once the upgrade occurs, HTTP is no longer involved – message-based protocol over TCP
- typically uses port 80 (`ws://myserver.com/`) or port 443 `wss://myserver.com/` for a secure channel)
- allows a server to *push* data to a browser, and the browser to send requests to the server without requiring multiple HTTP connections, reloading or polling
- So your browser can display *current information* efficiently
- supported by most major modern browsers: Firefox, Google Chrome, Safari, Internet Explorer, Opera.
- can also be used by applications outside the browser
- defined in RFC6455 (Fette and Melnikov, 2011)

Simple Websocket Client

```
class MyClientProtocol(WebSocketClientProtocol):

   def onOpen(self):
      self.sendMessage(u"Hello, world!".encode("utf8"))

   def onMessage(self, payload, isBinary):
      if isBinary:
         print("Binary message received: {0} bytes".format(
                                              len(payload)))
      else:
         print("Text message received: {0}".format(
                            payload.decode("utf8")))
```

- runs in an event loop provided by the Twisted framework
- on a websocket open event, the `onOpen()` method runs
- on a websocket message received event, the `onMessage()` method runs

How it works: Server

- Hardware:
  - HP ProLiant DL320e Gen8, Quad core Intel Xeon CPU E3-1220 V2 @ 3.10GHz, 8 GB RAM
- Software:
  - Tornado (The Tornado Authors, 2016)
    - a Python web framework and asynchronous networking library
    - by using non-blocking network I/O, Tornado can scale to tens of thousands of open connections
- Implementation:
  - Accepts websocket connection requests from gateways and from web clients (browsers)
  - Acts as a publish/subscribe broker
    - Gateways act as publishers of sensor data
    - Browsers act as subscribers to 'topics' – data from some particular sensor node
    - Topics are identified by sensor node id, e.g. `SN01`, i.e. when you subscribe to data from a sensor node, you subscribe to *all* its data, you can't choose just the temperature data, for example

Data structure for a primitive publish/subscribe broker

```
topics = {
  "SN00" : {
    "publisher" : <websocket gateway1>,
    "subscribers" : [
      <websocket pc1>,
      <websocket mobile1>
      <websocket tablet1>
    ]
  },
  "SN01" : {
    "publisher" : <websocket gateway1>,
    "subscribers" : [
      <websocket pc1>,
      <websocket mobile2>
    ]
  }
  "SN02" : {
    "publisher" : <websocket gateway2>,
    "subscribers" : [
      <websocket pc1>,
      <websocket tablet2>
    ]
  }
}
```

How it works: Browser

- Hardware:
  - Anything that can run a modern browser ...
- Software:
  - Any modern browser, supporting websockets, e.g. Google Chrome, Firefox, Safari, Internet Explorer etc.
  - Javascript libraries:
    - Smoothie Charts for accelerometer data
    - JustGage for generating gauges: potentiometer, temperature
  - Bootstrap for responsive web page
- Implementation:
  - Loads page and libraries
  - Creates chart and gauges
  - Opens websocket to server
  - Extracts sensor node id from URL and sends a SUBSCRIBE message to the server
  - Implements an `onmessage()` function that receives PUBLISHed data for the sensor node and updates the chart and gauges accordingly.
  - Implements various functions to send COMMANDs to the server

References I

ARM Ltd. (2016a). ARM mbed. https://developer.mbed.org/.

ARM Ltd. (2016b). Cortex-M series. https://www.arm.com/products/processors/cortex-m/index.php.

ARM Ltd. (2016c). Internet of Things (IoT). https://www.arm.com/markets/internet-of-things-iot.php.

ARM Ltd. (2016d). mbed IoT Device Platform. https://www.arm.com/products/internet-of-things-solutions/mbed-IoT-device-platform.php.

Banks, A. and Gupta, R. (2014). MQTT Version 3.1.1 OASIS Standard. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

Collina, M. (2016). Mosca: MQTT broker as a module. http://www.mosca.io/.

Contiki (2016). Contiki: The Open Source OS for the Internet of Things. http://contiki-os.org/.

Digi International (2015). Zigbee RF modules – User guide. http://ftp1.digi.com/support/documentation/90000976.pdf.

Eclipse (2016a). Mosquitto: An open source mqtt v3.1/v3.1.1 broker. http://mosquitto.org/.

Eclipse (2016b). Paho. http://www.eclipse.org/paho/.

Embedded Artists (2016). LPC4088 experiment bundle. http://www.embeddedartists.com/products/boards/lpc4088_exp_bb_bundle.php.

References II

Fette, I. and Melnikov, A. (2011). RFC6455: The Websocket Protocol. http://tools.ietf.org/html/rfc6455.

Fullstack (2016). Fullstack 2016 - the conference on Javascript, Node & Internet of Things. https://skillsmatter.com/conferences/7278-fullstack-2016-the-conference-on-javascript-node-and-internet-of-

Gartner (2013). Forecast: The internet of things, worldwide, 2013. http://www.gartner.com/document/2625419?ref=QuickSearch&sthkw=G00259115.

Hunt, T. (2016). Controlling vehicle features of Nissan LEAFs across the globe via vulnerable APIs. https://www.troyhunt.com/controlling-vehicle-features-of-nissan/.

IBM (2016). Watson Internet of Things. http://www.ibm.com/internet-of-things/.

mbed OS (2016). mbed OS. https://www.mbed.com/en/development/software/mbed-os/.

MBIENTLAB (2016). MetaWear C. https://store.mbientlab.com/product/metawear-c/.

PyPy (2016). Welcome to PyPy. http://pypy.org/.

Riot (2016). Riot: the friendly Operating System for the Internet of Things. https://www.riot-os.org/.

Shelby, Z., Hartke, K., and Bormann, C. (2014). RFC7252: The Constrained Application Protocol (CoAP). https://www.rfc-editor.org/rfc/rfc7252.txt.

References III

Tavendo GmbH (2015). Autobahn|Python. http://autobahn.ws/python/.

Texas Instruments (2015). The SimpleLink SensorTag. http://www.ti.com/ww/en/wireless_connectivity/sensortag2015/.

The Tornado Authors (2016). Tornado. http://www.tornadoweb.org/.

Twisted Matrix Labs (2016). Twisted Home Page. https://twistedmatrix.com/.

ZigBee Alliance (2016). ZigBee Alliance website. http://www.zigbee.org/.