Embedded Software for the IoT
Second Edition

Klaus Elk

© 2017, by Klaus Elk

Information provided in this book is derived from many sources, standards, and analyses - including subjective personal views. Any errors or omissions shall not imply any liability for direct or indirect consequences arising from the use of this information. The author make no warranty for the correctness or for the use of this information, and assume no liability for direct or indirect damages of any kind arising from technical interpretation or technical explanations in this book, for typographical or printing errors, or for any subsequent changes.

All rights reserved.

ISBN-13: 978-1545526033
ISBN-10: 1545526036

See also facebook.com/klauselkbooks and http://klauselk.com

Preface

The Internet of Things is here, and soon 50 Billion devices will be "connected". So we are told. This raises the question: "Who is going to program all these devices?"

According to a VDCResearch estimate back in 2011, there were 880k "embedded engineers" worldwide, with an annual growth rate of 7%. With an unchanged growth-rate, this amounts to 1.6 million embedded developers in 2020. Each will, by average, be responsible for approximately 30k devices, if all these numbers can be trusted.

We will probably not see 50 Billion connected devices, but we will need more embedded developers. In a major survey at "StackOverflow" in 2016, only 2.6% of the developers claimed to work in the embedded domain. Thus there is a large potential in attracting developers from the remaining 97.4%.

Surely, some developers will leave PC- and mobile-programming to work on all these embedded devices. Others will be new to the scene, self-taught or from universities. Common to all these developers is the overwhelming amount of new domains they need to get into, on top of their basic programming skills. The aim of this book is to help in this process. The author presents solid basic knowledge in the relevant domains in a structured way. This creates a strong foundation, onto which all the scattered details from the web may be attached.

Throughout the book the author draws on 30+ years of good and bad real-life experiences from the private industry as well as from university teaching, in an informal and relevant way.

What is new in this Second Edition?

The book has two new chapters in the "text-book part", which is now the third part. It may not be a surprise that one of these is about Internet Security. This area is growing in importance almost as fast as the Internet

of Things is growing in size. The other new chapter is about Statistical Process Control (SPC). This is a less obvious choice. However, as the new introduction to this book shows, SPC is an important part of "Industry 4.0", another term closely related to IoT.

On top of these two new chapters, the existing chapters are updated. The previous "Processes" chapter has been changed into "Code Maintenance" and now includes sections on Git and Yocto, while similar changes have been made in most other chapters. In terms of pages this edition is 50% longer than the first.

Finally the layout of the book has been tightened. The WireShark screenshots in the central network part are now easier to read, and the many new figures and tables improves the reading experience as well. This "tightening" has also affected the title: the long "Embedded Software Development for the Internet Of Things" has become the more user-friendly "Embedded Software for the IoT". Still, the basic aim is the same.

About The Author

Klaus Elk graduated as Master of Science in Electronics from the Danish Technical University in Copenhagen in 1984, with a thesis on Digital Signal Processing. Since then he has worked in the private industry within the domains of telecommunication, medical electronics and sound and vibration. He also holds a degree as Bachelor in Marketing. In a period of 10 years Klaus Elk - besides his R&D job - taught at the Danish Technical University. The subjects were Object Oriented Programming (C++ and Java), and the Internet Protocol Stack. Today he is R&D Manager in Instrumentation at Brüel & Kjær Sound & Vibration.

Acknowledgements

Thanks to my family for their patient listening. Especially I want to thank my daughter Amanda for her relentless proofreading, making me sound less like Yoda. I will also like to thank my colleagues for their support. In this case the special thanks goes to Camilla Travis for her great help with the cover.

Klaus Elk

Contents

1 Introduction **1**
1.1 The Tale of the Internet 1
1.2 The Cloud 3
1.3 Internet of Things 3
1.4 IoT related terms 5

I The Basic System **9**

2 How to select an OS? **11**
2.1 No OS and strictly polling 12
2.2 Co-routines 15
2.3 Interrupts 16
2.4 A small real-time kernel 19
2.5 A non-preemptive Operating System 21
2.6 Full OS 24
2.7 Open Source, GNU licensing and Linux 26
2.8 OS constructs 28
2.9 Further Reading 30

3 Which CPU to use? **31**
3.1 Overview 31
3.2 CPU Core 34
3.3 CPU Architecture 35
3.4 Word-Size 37
3.5 MMU – Memory Managed Unit 38
3.6 RAM 39
3.7 Cache 40
3.8 EEPROM and Flash 40
3.9 FPU – Floating Point Unit 41
3.10 DSP 41

3.11 Crypto-Engine 41
3.12 Upgrade Path 42
3.13 Second Sources 42
3.14 Price 42
3.15 Export Control 43
3.16 RoHS-Compliance 44
3.17 Evaluation Boards 44
3.18 Tool-chain 45
3.19 Benchmarking 45
3.20 Power-Consumption 46
3.21 JTAG Debugger 47
3.22 Peripherals 47
3.23 Make or Buy 52
3.24 Further Reading 55

II Best Practice **57**

4 Software Architecture **59**
4.1 Design for Performance 59
4.2 The fear of the white paper 61
4.3 Layers 63
4.4 Not just API's - more files 64
4.5 Object Model (Containment Hierarchy) 65
4.6 Case: CANOpen 66
4.7 Message Passing 69
4.8 Middleware 70
4.9 Case: Architectural Reuse in LAN-XI 71
4.10 Understanding C 74
4.11 Further Reading 78

5 Debug Tools **79**
5.1 Simulator 79
5.2 ICE – In-Circuit-Emulator 80
5.3 Background or JTAG debugger 80
5.4 Target Stand-In 81
5.5 Debugger 82
5.6 strace 84
5.7 Debugging without special tools 85
5.8 Monitoring Messages 86
5.9 Test Traffic 87

6 Code Maintenance **89**
6.1 Poor Mans Backup 89
6.2 Version Control - and git 90
6.3 Build and Virtualization 96
6.4 Static Code Analysis 97
6.5 Inspections 97
6.6 Tracking Defects & Features 100
6.7 Whiteboard 103
6.8 Documentation 104
6.9 Yocto 104
6.10 OpenWRT 107
6.11 Further Reading 108

III IoT Technologies **109**

7 Networks **111**
7.1 Introduction 111
7.2 Cerf & Kahn – internet as net of nets 111
7.3 Life of a packet 112
7.4 Life before the packet 119
7.5 Getting an IP address 121
7.6 DHCP 122
7.7 Network Masks, CIDR and Special Ranges 125
7.8 Reserved IP-Ranges 126
7.9 DNS 128
7.10 Introducing HTTP 131
7.11 REST 133
7.12 TCP sockets on IPv4 under Windows 135
7.13 IP Fragmentation 141
7.14 Introducing IPv6 addresses 146
7.15 TCP sockets on IPv6 under Linux 148
7.16 Data Transmission 153
7.17 UDP Sockets 155
7.18 A case of UDP on IPv6 159
7.19 Application Layer Protocols 164
7.20 Alternatives to the socket API 167
7.21 Ethernet Cabling 168
7.22 Physical Layer Problems 170
7.23 Further Reading 173

8 Network Tools **175**
8.1 Finding the IP-Address 175
8.2 The Switch as a Tool 178
8.2.1 Mirroring 178
8.2.2 Statistics 179
8.2.3 Simulating lost frames 179
8.2.4 Pause frames 179
8.3 Tap 182
8.4 SNMP 183
8.5 Wireshark 184
8.6 Network Commands 186
8.7 Further Reading 187

9 Wireless Networks **189**
9.1 Introduction 189
9.2 WiFi Basics 192
9.3 The Access Point as a Repeater 194
9.4 How is speed calculated? 197
9.5 Case: WiFi Data-transmission 198
9.6 Case: Beacons 200
9.7 Why so hesitant? 202
9.8 Aggregated Frames 203
9.9 Channel Assessment 207
9.10 Bluetooth Low Energy 207
9.11 Certification 210
9.12 Further Reading 212

10 Security **213**
10.1 Introduction 213
10.2 The goals of a Hacker 216
10.3 Network Security Concepts 217
10.4 Hash-function 218
10.5 Symmetric Key Encryption 221
10.6 Case Story: Enigma 222
10.7 Asymmetric Key Encryption 225
10.8 Digital Signature 227
10.9 Certificates 228
10.10 Message Authentication Code 230
10.11 Nonce 231
10.12 Secure Socket Communication 232
10.13 Case Story: Heartbleed 234

10.14 Case Story: WiFi Security 235
10.15 Software Crypto Libraries 238
10.16 Trusted Platform Module 239
10.17 Embedded Systems 240
10.18 Vulnerabilities in Embedded Systems 242
10.19 Export Control 246
10.20 Further Reading 249

11 Digital Filters **251**
11.1 Why Digital? 251
11.2 Why Filters? 253
11.3 Basic Signal Analysis 253
11.3.1 About the Sample Frequency 253
11.3.2 Time and Frequency Domains and their Duality . . . 254
11.3.3 Analog and Digital 255
11.3.4 More Duality 256
11.3.5 A Well-behaving System 257
11.4 IIR filter Basics 257
11.5 Implementing IIR 261
11.6 FIR Filter basics 262
11.7 Implementing FIR 269
11.8 Integers 272
11.9 Fixed Point Arithmetic 274
11.10 Q-Notation and Multiplication 276
11.11 Division 276
11.12 BCD 277
11.13 Further Reading 278

12 Statistical Process Control **279**
12.1 Introduction 279
12.2 Important terms 282
12.3 Control Charts 283
12.4 Finding the Control Limits 286
12.5 Subgroups 289
12.6 Case: Insulation Plates 290
12.7 EWMA Control Charts 294
12.8 Process Capability Index 294
12.9 Further Reading 297

Epilogue **299**

List of Figures

1.1 Internet Stack - an Hourglass 2
1.2 The Cloud Abstraction 4
1.3 An unusual - but architectural relevant - view 5

2.1 Finite State Machine in Main 15
2.2 Purely Interrupt Controlled System 16
2.3 Interrupt System with Finite State Machine in Main 17
2.4 OS with Preemptive Scheduling 20
2.5 Tasks and interrupts 21

3.1 AM335c - Courtesy Texas Instruments 33
3.2 System design is like Russian Dolls 34
3.3 Endian-ness 38
3.4 BeagleBone (left) and Raspberry Pi 45

4.1 Layering 64
4.2 CANOpen usage of the CAN-ID 67
4.3 ICPDAS User interface for CANOpen 68
4.4 Product-Line Reuse 72

5.1 BeagleBone Target using Host disk via TFTP and NFS 83
5.2 Cross Debugging BeagleBone using Eclipse 84

6.1 Main git actions 93
6.2 Oracle Virtual Box running Ubuntu 98
6.3 Prefast in Windows CE. 98
6.4 Simple Defect/Bug Workflow 101
6.5 Forecasting a project - and taking action 102
6.6 Incoming bugs versus fixed bugs in Atlassian Jira 103
6.7 Yocto Layers - Sample 105

7.1 The generic layers, typical variant and the data unit names . 113
7.2 Internet with some network elements 114

7.3 Transmission with HTTP using "Follow TCP-Stream" 115
7.4 A ping that provokes ARP 121
7.5 DHCP commands and options 123
7.6 Link-Local with Gratuitous ARP 125
7.7 A DNS request for google.com 130
7.8 Test-program showing an Object Model on REST 136
7.9 TCP State-Event Diagram 138
7.10 Webbrowser with response from simple server 144
7.11 Fragmented ICMP ping with 5000 bytes of data 145
7.12 TCP on Linux localhost with IPv6 153
7.13 Transmitting many bytes 154
7.14 Nice Stevens Graph of Sequence Number versus time 156
7.15 Stevens Graph with many retransmissions 156
7.16 Expert Info with lost data 157
7.17 UDP on IP6 with ICMPv6 error 163
7.18 Cheap Ethernet tester 171
7.19 A good Eye Diagram - outside the illegal zones 172

8.1 Angry IP scanner for Windows 176
8.2 iPhone Net Analyzer 177
8.3 Mirror Setup on Port 8 in switch 180
8.4 Receive Errors part of Statistics page in switch 180
8.5 Port Setup with Flow Control 181
8.6 Receive Counters part of Statistics page in switch 181
8.7 Ethernet Pause Frames 182
8.8 TAP Device 183
8.9 SNMP Client with lots of data from Windows CE 184

9.1 Multicast on WiFi is two-fold 196
9.2 Frame 2419 from STA to AP 196
9.3 Frame 2421 from AP to STA's 196
9.4 Full speed ahead on WiFi 200
9.5 WiFi spectrum of our transmission (in B/W) 201
9.6 Beacon from the Asus AP 202
9.7 Probe from the iPad STA 203
9.8 Wireshark Throughput graph 204
9.9 Wireshark RTT 204
9.10 iPad Power Nap. 13 breaks every 300s 205
9.11 A-MPDU's and their contents 206
9.12 Channel Assessment with RTS and CTS 207

10.1 Confidentiality by Symmetric Key Encryption 223
10.2 Enigma. 3 rings rotate like a clock after each letter. 224
10.3 Confidentiality by Public/Private key-pair 226
10.4 Integrity, Authentication and Non-repudiation 228
10.5 Certificates in Chrome Browser on Windows 229
10.6 Message Authentication Code without encryption 230
10.7 The Logo for the Heartbleed bug 236

11.1 Bode plot - Magnitude and Phase - of an IIR-filter 256
11.2 An everlasting sine of 1 Hz 258
11.3 A single square becomes $sin(x)/x$ in the Frequency Domain 258
11.4 Widest pulse - slimmest & highest $sin(x)/x$ 258
11.5 A square-wave duty-cycle 50% 259
11.6 Building up a square-wave - all Time Domain 259
11.7 Building up a square-wave - all Time Domain II 259
11.8 BiQuad IIR Filter Implementation 263
11.9 IIR-Filter Design by IowaHills 264
11.10 Pole-Zero Plot of the Elliptic Filter 265
11.11 2. Order FIR Filter 266
11.12 IowaHills FIR Designer 267
11.13 Coefficients from FIR design program 269
11.14 Buffer-usage for MIT's implementation 271

12.1 The Normal Distribution 283
12.2 Control Chart 284
12.3 X-R Control Chart on 10 subgroups of 5 objects 285
12.4 X-s Control Chart on 10 subgroups of 5 objects 295

List of Tables

2.1 Task Management - from simple to advanced 11
2.2 OS primitives for scheduling 29

3.1 Overview of concepts related to CPU-choice 32

4.1 Important CANOpen Message Types 67

6.1 Robocopy Parameters 89
6.2 Selected git commands 94

7.1 The most important reserved IP-address ranges 127
7.2 Sample NAT-table 128
7.3 The main HTTP commands, and what they do 131
7.4 REST Rules 133
7.5 How IPv4 and IPv6 Notation differs 146
7.6 Functional differences between IPv4 and IPv6 147
7.7 Examples on addresses in the two systems 148
7.8 The three types of -casts 155
7.9 TCP versus UDP 158
7.10 Parameters on application layer protocols 164
7.11 Names and terms used with Ethernet cables 169
7.12 EIA-568A Wiring scheme 170

8.1 Network Commands (Linux specific name in parantheses) . . 186

9.1 A selection of wireless networks 192
9.2 Parameters on application layer protocols 193
9.3 Wireshark terms for WiFi nodes 195
9.4 WiFi datarate calculation 198
9.5 Bluetooth Classic versus BLE 208
9.6 Bluetooth Smart Connectivity 209

10.1 Security Terms 219

10.2 Secure Hashes described in NIST FIPS-180-4 221
10.3 Crypto++ - an overview 239
10.4 Trusted Platform Module v2.0 240

11.1 Sample filter parameters 262
11.2 Examples of 16-bit signed integers 274

12.1 Statistical Symbols 283
12.2 Control Chart Definitions 284
12.3 Control Chart Alarm Rules 287
12.4 Control Chart Constants for X-Bar, R- and s-charts 289
12.5 10 days of testing - 5 samples each day 291
12.6 X-Bar and R Control Lines 292
12.7 X-Bar and s Control Lines 292
12.8 Subgroup Data with Means and Ranges 293
12.9 C_{pk} and related values 296

Chapter 1

Introduction

1.1 The Tale of the Internet

LAN's - Local Area Networks - became accessible during the seventies. This technology allowed computers within a building to connect to each other. The typical applications were printing and basic file-sharing, leaving the rest to the user. With the appearance of the IBM PC in the mid-eighties, the market grew, and we saw a regular war between network vendors with each their technology, e.g. Token-Ring, Token-Bus, Ethernet, Novell, AppleTalk, and DECnet. Especially the first three fought for the general market, whereas AppleTalk and DECnet respectively were specific to Apple and Digital. Novell quickly adopted Ethernet in its solution - successfully focusing on network-software for years. Even though Ethernet was not the technically most advanced solution, and only covered what we now know as the two lowest layers of the OSI-stack, it won the war. This is a great demonstration of how a solution which is available and "good enough" can gain traction, and become popular, leading to lowered prices, leading to more sales, leading to lower prices etc.

Before Ethernet became the clear winner, engineers and scientists were struggling with how to scale the LAN's to become more than local. Different physical layers, address-schemes and protocols, almost made this impossible. Vinton Cerf & Robert Khan from DARPA presented, in a paper in 1974, a concept of common virtual addresses on top of the various physical-rooted addresses. They presented a new common protocol, **T**ransmission **C**ontrol **P**rotocol, on top of these. TCP connected the existing networks, and they called the concept "the internet". The virtual addresses became known as **I**nternet **P**rotocol Addresses, or IP-addresses. Later on, TCP and IP became separate layers, making it possible to run e.g.

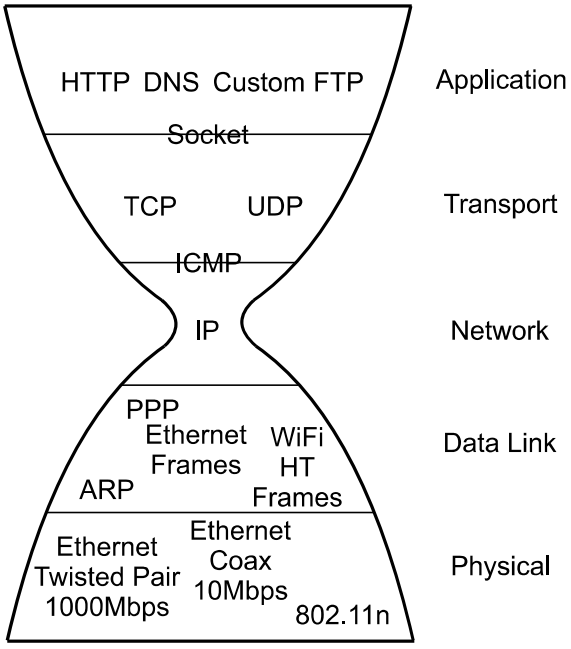

Figure 1.1: Internet Stack - an Hourglass

the simpler UDP on top of the IP network protocol. TCP and IP are treated in depth in Chapter 7.

Looking back, it is amazing how Ethernet has scaled from 10 Mbit/s to 10 Gbit/s, in affordable and still more robust solutions, **and** how TCP has been able to adapt to this. IP version 4 is growing out of addresses and IP version 6 is gradually taking over, as discussed in Chapter 7. The virtual addresses are hierarchical. This means that they are routable, and can change as we move. This is exactly like post-addresses and has proven extremely useful.

In 1989 Tim Berners-Lee of CERN presented the World-Wide-Web. Many people think of WWW as being the same as the internet, but WWW is some 20+ years younger than the internet, while being simply one of many applications on it. The "hourglass" architecture shown in Figure 1.1 with the single IP in the center, millions of applications on top, and many physical network solutions at the bottom, has also proven itself to be extremely powerful. Software architecture is discussed in Chapter 4.

In the same period as the internet was developed, Unix was developed at AT&T using the new machine-independent language "C". Unix and C were both created by Ken Thompson and Denis Ritchie. Unix was also one

of the first adopters of TCP/IP. In other words: many of the strengths Linux has today originates from the seventies. Operating Systems are discussed in Chapter 2.

1.2 The Cloud

While the early internet was mostly of academic interest[1], the World-Wide-Web brought it to the people with its easy clickable "hyper-links". Web-servers moved into companies and institutions, where they began to serve "fixed content", text and figures, from file-servers to us all.

With the help of script-languages such as PHP and CGI, the content became more dynamic, now often fetched from databases. This lasted for some years until we saw what was coined "Web 2.0". Before Web 2.0 it was possible for a web-server to update the content sent to the web-browser, at the cost of a complete "re-draw" of the page. Now the web-applications began to look like Windows or Mac applications, updating fields, graphs etc. This was also the breakthrough of Open-Source with the popular "LAMP", which stands for Linux (the OS), Apache (the web-server), MySQL (the database) and Python or PHP or Perl (all scripting languages).

As the complexity of web-servers grew and the internet became faster and more robust, it became economically wise for many companies and institutions to use "Web-Hotels". The web-hotels based their income on the scale: the constantly changing know-how required to run a single server is much better utilized running many. Now, not even the employees of a given company knew where their company web-site was hosted, and they didn't care. It was somewhere "in the cloud". The cloud-term became even more popular as we started to use smart-phones, clearly connecting over the air - through the clouds. See Figure 1.2.

1.3 Internet of Things

The first web-cam was used at Cambridge University to show a coffee-machine, so that no-one needed to go to an empty machine. A fun idea, but not the most practical. Instead you might consider building a web-server into the coffee-machine, telling the user how many cups are left. The problem with this solution is that it is not very *scalable*. This is an important term, that we will meet later. A web-server in the small electronics in a cheap coffee-machine would typically be too limited to serve many

[1] E-mail came before WWW and was driving a slow growth of the internet

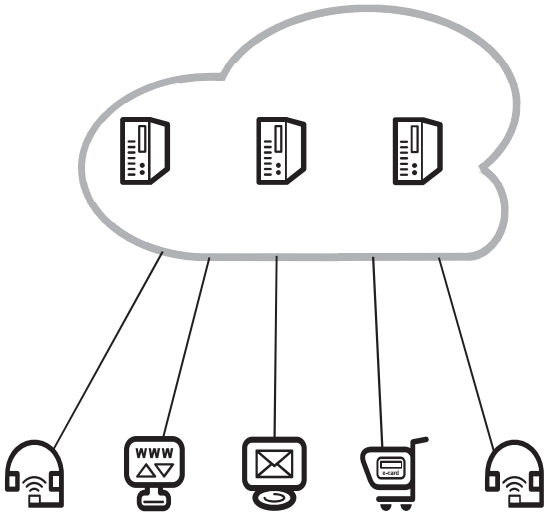

Figure 1.2: The Cloud Abstraction

web-browsers. At this point in history we actually did see a lot of "Web-Enabled" devices. It is practical that service technicians can connect to hardware using a standard web-browser, and we still use this concept. At the time however, the devices typically were not permanently connected to the internet, they just used the technology on rare occasions in a local connection, directly to the technician's PC.

The scalable solution to the coffee-machine problem would be a PC-based web-server, reading many coffee-machines, so that each machine only has a single client reading it e.g. twice a minute, and many users could connect to the web-server.

Let the coffee-machine be any small internet-attached device, place the web-server in the cloud, and let the users connect via smart-phone, PC or tablet, and you have the "Internet of Things", in short "IoT". This is another application of the hourglass architecture, shown in Figure 1.3, only now the hourglass is not the implementation in a single PC, but the entire internet. Rather unusual, the clients are placed above the cloud to demonstrate the architectural point. Another term sometimes used for all the devices "on the ground" is "The Fog".

The inventor of Linux, Linus Thorvalds, once stated in an interview that he created Linux for the desktop, and that it has overtaken the computerized world, except on the desktop. This refers to the Open Source first being extremely popular in the servers due to price, robustness and ease of

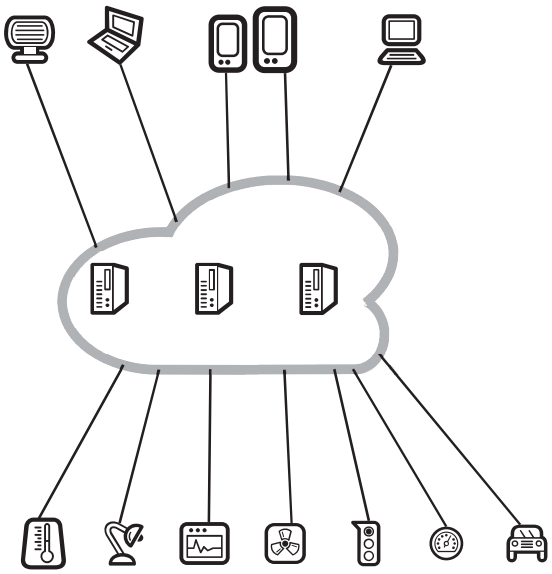

Figure 1.3: An unusual - but architectural relevant - view

programming, while now it is moving into the small devices for the same reasons, plus a few more. Common to the (web-)servers and devices is that they are only used directly by computer-professionals. These people are not relying on crisp graphical interfaces, but prefer programs and scripts featuring automation.

Devices such as the BeagleBone, Arduino and Raspbery Pi have opened the Linux-device world to new generations, but real-world devices typically need to be more robust. These are discussed in Chapter 3.

1.4 IoT related terms

- **Industry 4.0**
  Germany has been the most advanced European Country in the automotive market for a long time. The main player, VW, was shaken in 2015 by "DieselGate". Even before this, the Germans worried about *disruption* - the way new technologies may turn a thriving business into bankruptcy in no time. They knew they could not afford to rest on their laurels. As one of ten "Future Projects" in a 2020 plan, the German government publicized "Industrie 4.0" in 2013.

The "fourth industrial revolution" has been claimed several times in history. This time it was an elaborate plan, envisioning the connectivity of IoT with its ability to bring real-time factory-floor data from e.g. Brazil, to production control in Germany. These data are typically in the form of statistics relating to specific assembly lines, product type, production gear, day/night shift etc. Equally important, this plan focused on decentralization. This means more intelligent embedded systems at the assembly line, taking more autonomous and advanced decisions. These systems could possibly be calibrated remotely along the way, based on the statistics collected. Chapter 12 deals with Statistical Process Control for embedded systems, and in Chapter 4 we are looking at CANOpen. This is one of several standards in factory automation.

Today, this advanced embedded software will often communicate with the product itself. It is no surprise that a car may be queried for its type, serial no etc, and that calibrated production data may be written back to the car. However, via the car the plant software can talk to its internal CAN buses, and via these to e.g. brakes, airbags and other subsystems.

- **Industrial Internet of Things - IIoT**
  The "IIoT" term sounds very much like the "Industry 4.0" concept, but it has a less clear birth certificate. The "Industrial Internet Consortium" has taken the term to heart. IIC was formed in 2014 by AT&T, Cisco, General Electric, IBM and Intel, and since then companies from the whole world has joined. The vision is to assure the true interconnection of devices. The industries include manufacturing, energy, health care, transportation and smart cities.

- **Internet of Everything - IoE**
  Cisco is one of the members of the IIC. Cisco defines "the Internet of Everything" (IoE) as the "networked connection of people, process, data, and things". In this way it includes the Internet of Things, without forgetting the already existing internet of people. This makes sense, as Cisco's market is the infrastructure, it doesn't really matter whether a person or a device is generating or subscribing to data. By including the word "process", Cisco opens a door for expanding the current cloud with its web-servers and databases with actual applications. This is something that also interests major players like Google, Microsoft, Facebook and Amazon.

- **Big Data**
  Big data is clearly not the same as IoT. However, all these billions of internet-attached devices will create unprecedented amounts of data. Chapter 11 shows how some of these data may be filtered already at their birth in the devices. This will be essential in many scenarios. Processing power has grown at an impressive pace during the computer era, but the amounts of storage have grown even more. Big Data may be fantastic, but if we want fast algorithms, they need to work on less, but more relevant, data. We need to reduce data at the source and in the various process-steps, to a certain limit. To quote Einstein: "Make things as simple as possible, but not simpler". The subject of treating Big Data in the cloud is not in the scope of this book.

Should you want to browse through the amazing birth of the web and all the innovations that helped it along, take a look at "100 Ideas that Changed the Web" by Jim Boulton. This book spends only two pages per innovation, and one of these is typically a color-image. This makes it entertaining, but still very informative.