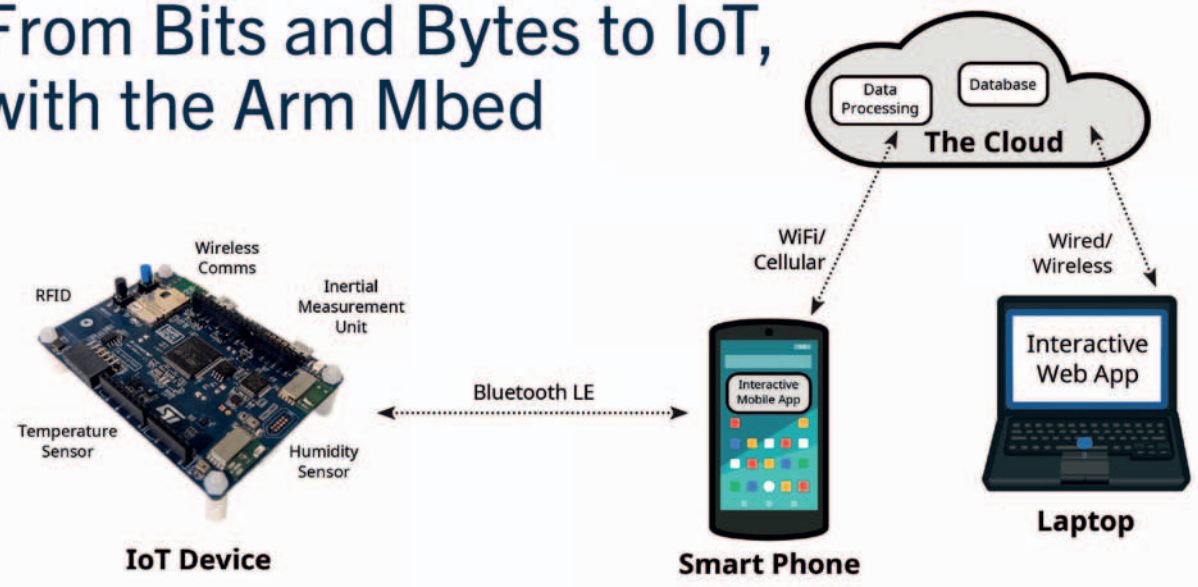
Newnes

Third Edition

FAST AND EFFECTIVE EMBEDDED SYSTEMS DESIGN

From Bits and Bytes to IoT, with the Arm Mbed



Tim Wilmshurst, Rob Toulson and Tom Spink

Fast and Effective Embedded Systems Design

This page intentionally left blank

Fast and Effective Embedded Systems Design

From Bits and Bytes to IoT, with the Arm Mbed

Third Edition

Rob Toulson
Tim Wilmshurst
Tom Spink

Amsterdam • Boston • Heidelberg • London • New York • Oxford • Paris
San Diego • San Francisco • Singapore • Sydney • Tokyo
Newnes is an imprint of Elsevier

Academic Press is an imprint of Elsevier
125 London Wall, London EC2Y 5AS, United Kingdom
525 B Street, Suite 1650, San Diego, CA 92101, United States
50 Hampshire Street, 5th Floor, Cambridge, MA 02139, United States
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, United Kingdom

Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

ISBN: 978-0-323-95197-5

Publisher: Katey Birtcher
Acquisition Editor: Stephen R. Merken
Editorial Project Manager: Sara Valentino
Production Project Manager: Nandhini Thanga Alagu
Cover Designer: Mark Rogers

Typeset by TNQ Technologies

Contents

Introduction ..... *xv*
Acknowledgments ..... *xix*

PART 1: Essentials of Embedded Systems, and Developing with Mbed .. 1

Chapter 1: Embedded Systems, Microcontrollers, and Arm 3
1.1 Introducing Embedded Systems 3
1.1.1 What is an Embedded System? 3
1.1.2 Fast Forward to the Internet of Things 5
1.1.3 The Electric Car—An Abundance of Embedded Systems 5
1.2 Microprocessors and Microcontrollers 8
1.2.1 Some Computer Essentials 8
1.2.2 The Microcontroller 10
1.3 Code Development in Embedded Systems 11
1.3.1 Machine Code, Assembler, and High-level anguages 11
1.3.2 Flow Diagrams and Simple Sequential Programming 13
1.3.3 The Development Cycle 14
1.4 The World of Arm 16
1.4.1 A Little History 16
1.4.2 Some Technical Detail—What Does This Word RISC Mean? 17
1.4.3 The Cortex Core 19
Chapter Review 22
Quiz 22
References 23

Chapter 2: Introducing the World of Mbed 25
2.1 Introducing the Mbed Environment 25
2.1.1 Mbed, Mbed Enabled, and the Mbed "Ecosystem" 25
2.1.2 Choosing Our Target Systems 26
2.2 Where Mbed Began—The Mbed LPC1768 27
2.2.1 Introducing the Mbed LPC1768 27
2.2.2 The Mbed LPC1768 Architecture 28
2.2.3 The LPC1768 Microcontroller 30
2.3 Nucleo and the F401RE Development Board 31

2.4 The STM32 Discovery Kit IoT Node 34
2.5 The Framework for Mbed Code Development 36
2.5.1 Using C/C++ 36
2.5.2 The Mbed API 36
2.5.3 The Mbed Operating System 37
2.5.4 The "Bare Metal" Profile 39
2.6 The Studio Development Environments, and a First Program 39
2.6.1 Using Mbed Studio/Keil Studio Cloud 40
2.6.2 Adding Software Libraries 44
2.7 The Mbed Application Board 44
Chapter Review 48
Quiz 49
References 49

Chapter 3: Digital Input and Output 51

3.1 Starting to Program 51
3.1.1 Thinking About the First Program 51
3.1.2 A First Word on Timing 53
3.1.3 Using the Mbed API 55
3.1.4 Exploring the *while* Loop 56
3.1.5 Using Development Board External Pins 58
3.2 Voltages as Logic Values 59
3.3 Digital Output 61
3.3.1 Using LEDs 61
3.3.2 Using a Breadboard to Connect External Components 62
3.4 Using Digital Inputs 64
3.4.1 Connecting Switches to a Digital System 64
3.4.2 The DigitalIn API 65
3.4.3 Using *if* to Respond to a Switch Input 65
3.5 Digital Input and Output with the Application Board 67
3.6 Interfacing Simple Opto Devices 71
3.6.1 Opto-reflective and Transmissive Sensors 71
3.6.2 Connecting an Opto-sensor to an Mbed Development Board 72
3.6.3 Seven-Segment Displays 73
3.6.4 Connecting a Seven-Segment Display to the Mbed Target 75
3.7 Switching Larger DC Loads 77
3.7.1 Applying Transistor Switching 77
3.7.2 Switching a Motor with the Mbed 79
3.7.3 Switching Multiple Seven-segment Displays 81
3.8 Mini-project: Letter Counter 82
Chapter Review 82
Quiz 82
References 85

Chapter 4: Analog Output and Pulse Width Modulation 87
4.1 Introducing Data Conversion 87
4.1.1 The Digital-to-Analog Converter 87
4.2 Analog Outputs on the Mbed LPC1768 89
4.2.1 Constant Output Voltages 90
4.2.2 Sawtooth Waveforms 90
4.2.3 Testing the DAC Resolution 93
4.2.4 Generating a Sine Wave 93
4.3 Another Way of Replicating Analog Output: Pulse Width Modulation 94
4.4 Pulse Width Modulation in the Mbed Environment 97
4.4.1 Using the Mbed PWM Sources 97
4.4.2 Some Trial PWM Outputs 97
4.4.3 Speed Control of a Small Motor 99
4.4.4 Generating PWM in Software 100
4.4.5 Servo Control 102
4.4.6 Producing Audio Output 104
Chapter Review 106
Quiz 107

Chapter 5: Analog Input 109
5.1 Analog-to-Digital Conversion 109
5.1.1 The Analog-to-Digital Converter 109
5.1.2 Range, Resolution, and Quantization 110
5.1.3 Sampling Frequency 113
5.1.4 Analog Input with the Mbed 114
5.2 Combining Analog Input and Output 114
5.2.1 Controlling LED Brightness by Variable Voltage 114
5.2.2 Controlling LED Brightness by PWM 117
5.2.3 Controlling PWM Frequency 117
5.3 Processing Data from Analog Inputs 119
5.3.1 Displaying Data on the Computer Screen—printf( ) and the Serial Terminal 119
5.3.2 Scaling ADC Outputs to Recognized Units 122
5.3.3 Applying Averaging to Reduce Noise 122
5.4 Some Simple Analog Sensors 123
5.4.1 The Light-Dependent Resistor 123
5.4.2 Integrated Circuit Temperature Sensors 125
5.5 Exploring Data Conversion Timing 126
5.5.1 Estimating Conversion Time and Applying Nyquist 126
5.6 Introducing the Debugger 128
5.6.1 A Motor Control Program 128
5.6.2 Simple Use of the Debugger 131
5.7 Mini-Projects 133
5.7.1 Two-Dimensional Light Tracking 133
5.7.2 Temperature Alarm 134

Chapter Review 134
Quiz 135
References 135

Chapter 6: Interrupts and Timers 137

6.1 Thinking about Time 137
6.2 Responding to External Events 138
6.2.1 Polling 138
6.2.2 Introducing Interrupts 139
6.3 Interrupts with the Mbed Operating System 140
6.4 Getting Deeper into Interrupts 143
6.4.1 Interrupts on the LPC1768 145
6.4.2 Testing Interrupt Latency 145
6.4.3 Disabling Interrupts 147
6.4.4 Interrupts from Analog Inputs 147
6.4.5 Conclusion on Interrupts 149
6.5 An Introduction to Timers 149
6.5.1 The Digital Counter 149
6.5.2 Using the Counter as a Timer 150
6.5.3 Timers in the Mbed Environment 150
6.6 Using the Mbed Timer 151
6.6.1 The Timer API 151
6.6.2 Using Multiple Mbed Timers 152
6.7 Using the Mbed Timeout 154
6.7.1 A Simple Timeout Application 154
6.7.2 Further Use of Timeout 156
6.7.3 Timeout Used to Test Reaction Time 157
6.8 Using the Mbed Ticker 159
6.8.1 Using Ticker for a Metronome 160
6.8.2 Reflecting on Multi-tasking in the Metronome Program 163
6.9 The RTC 163
6.10 Switch Debouncing 165
Chapter Review 167
Quiz 168

Chapter 7: Starting with Serial Communication 169

7.0 A Word on Inclusive Language 169
7.1 Introducing Synchronous Serial Communication 169
7.2 Serial Peripheral Interface 171
7.2.1 Introducing SPI 171
7.2.2 SPI in the Mbed Environment 173
7.2.3 Setting up an Mbed SPI Master 174
7.2.4 Creating an SPI Data Link 176
7.3 Intelligent Instrumentation and an SPI Accelerometer 180
7.3.1 Introducing the ADXL345 Accelerometer 181

7.3.2 Developing a Simple ADXL345 Program 182
7.4 Evaluating SPI 184
7.5 The Inter-integrated Circuit (I^2C) Bus 185
7.5.1 Introducing the I^2C Bus 185
7.5.2 I^2C with the Mbed OS 187
7.5.3 Setting Up an I^2C Data Link 187
7.6 Communicating with I^2C-Enabled Sensors 192
7.6.1 The TMP102 Sensor 192
7.6.2 The SRF08 Ultrasonic Range Finder 195
7.7 Evaluating I^2C 197
7.8 Asynchronous Serial Data Communication 198
7.8.1 Introducing Asynchronous Serial Data 198
7.8.2 Applying Asynchronous Communication in the Mbed Environment 199
7.8.3 Applying Asynchronous Communication with the Host Computer 202
7.9 USB 203
7.9.1 Introducing USB 204
7.9.2 USB Capability in the Mbed Environment 205
7.9.3 Using the Mbed to Emulate a USB Mouse 205
7.9.4 USB On-the-Go 207
7.10 Using Serial on the ST IoT Discovery Board 207
7.11 Mini-projects 210
7.11.1 Multi-node I^2C Bus 210
7.11.2 Vibrating Beam Acceleration Threshold Detection 210
Chapter Review 211
Quiz 212
References 213

Chapter 8: Liquid Crystal Displays 215

8.1 Display Technologies 215
8.1.1 Introducing Liquid Crystal Technology 215
8.1.2 Liquid Crystal Character Displays 216
8.2 Using the PC1602F LCD 218
8.2.1 Introducing and Connecting the PC1602F Display 218
8.2.2 Using Modular Coding to Control the LCD 220
8.2.3 Initializing the Display 220
8.2.4 Sending Display Data to the LCD 222
8.2.5 Calling the LCD Functions from main( ) 224
8.2.6 Adding Data to a Specified Location 225
8.3 Using the Mbed TextLCD Library 226
8.4 Displaying Analog Input Data on the LCD 228
8.5 Pixel Graphics—Implementing the NHD-C12832 Display 230
8.6 Color LCDs and the uLCD-144-G2 237

8.7 Mini-projects 241
8.7.1 Digital Spirit Level 241
8.7.2 A Self-contained Metronome 241
Chapter Review 242
Quiz 242
References 243

Chapter 9: Programming in Real Time 245

9.1 Multitasking and Real Time 245
9.2 Scheduling 247
9.2.1 Scheduling and Context Switching 247
9.2.2 Thread States 250
9.3 The Mbed RTOS 252
9.4 Writing Multi-threaded Programs 254
9.4.1 Defining and Writing Threads 254
9.4.2 Two-threaded Programs 255
9.4.3 A Multi-threaded Program 256
9.4.4 Setting Thread Priority 259
9.5 The Mutex 261
9.5.1 The Mutex Concept 261
9.5.2 A Mutex Program 263
9.6 The Semaphore 265
9.6.1 The Semaphore Concept 265
9.6.2 Programming with Semaphores 267
9.7 Using Interrupts with the RTOS 270
9.8 Queues and Memory Pools 272
9.8.1 The Queue Concept 272
9.8.2 Programming with Queues and Memory Pools 272
9.9 The "Bare Metal" Profile 276
Chapter Review 276
Quiz 277
References 280

Chapter 10: Memory and Data Management 281

10.1 A Memory Review 281
10.1.1 An Overview of Memory Technologies 281
10.1.2 Memory Mapping 285
10.2 Some Programming Techniques for Data and File Management 286
10.2.1 Using Addresses and Pointers 286
10.2.2 File Management in C/C++ 289
10.2.3 Using Data Files with the Mbed OS 289
10.3 The Mbed LPC1768 Local File System 291
10.3.1 Opening and Closing Files 291
10.3.2 Recovering a "Lost" Mbed LPC1768 292

10.3.3 Simple File Data Transfers 292
10.3.4 String File Access 294
10.3.5 Using Formatted Data 295
10.4 Using External SD Card Memory 297
10.4.1 Simple SD Card Block Transfers 297
10.4.2 Creating Files with the FAT File System 301
10.5 Mini Project: Accelerometer Data Logging on Exceeding Threshold 304
Chapter Review 305
Quiz 305
References 306

PART 2: Wireless and the Internet of Things 307

Chapter 11: Wireless Communication 309

11.1 Introducing Wireless Data Communication 309
11.1.1 Some Wireless Preliminaries 309
11.1.2 Wireless Networks 312
11.1.3 A Word on Protocols 313
11.2 Bluetooth Low Energy 314
11.2.1 Classic Bluetooth 315
11.2.2 Bluetooth Low Energy 315
11.2.3 Establishing a Connection 316
11.2.4 Exchanging Data 319
11.2.5 BLE on the ST IoT Discovery Board 319
11.2.6 Simple BLE: Sending Data to a Smart Phone 320
11.2.7 Evaluating BLE 326
11.3 Zigbee 328
11.3.1 Introducing Zigbee 328
11.3.2 Introducing XBee Wireless Modules 330
11.3.3 Introducing the XCTU Software 331
11.3.4 Configuring an XBee Pair 333
11.3.5 Implementing Zigbee Links 334
11.3.6 Introducing the XBee API 338
11.3.7 Applying the XBee API 339
11.3.8 Conclusion on Zigbee and Further Work 343
11.4 LoRa and LoRaWAN 343
11.5 Mini Projects 345
11.5.1 BLE Mini Project 345
11.5.2 Zigbee Mini Project 345
Chapter Review 346
Quiz 346
References 347

Chapter 12: Towards the Internet of Things 349
12.1 What is the Internet of Things? 349
12.1.1 An IoT Overview 349
12.1.2 Applications of the IoT 351
12.1.3 A Little IoT History 352
12.2 Core Components in the IoT 353
12.2.1 Sensors and Actuators 353
12.2.2 Connectivity 354
12.2.3 Processing: Edge Computing 356
12.2.4 Processing: Cloud Computing 357
12.2.5 Mobile and Web Applications 359
12.3 Case Studies 359
12.3.1 A Smart Lightbulb 359
12.3.2 A Wearable Fitness Tracker 361
12.4 Building a Full-stack IoT System: An Activity Tracker 362
12.4.1 Overview 363
12.4.2 Getting Started with Google Cloud 364
12.4.3 Firestore 365
12.4.4 Cloud Functions 367
12.4.5 Preparing the Embedded Device 373
12.4.6 Sending Movement Data 376
12.4.7 Building the Mobile App: Designing the User Interface 377
12.4.8 Building the Mobile App: Reading and Processing Activity Data 381
12.5 Summary 389
12.6 Mini Projects 390
12.6.1 Smart Lightbulb Simulation 390
12.6.2 Reconfiguring the Activity Tracker Example 390
Chapter Review 390
Quiz 391
References 393

Chapter 13: Further Aspects of the IoT 395
13.1 Ethernet 395
13.2 Wi-Fi 399
13.3 Getting Data from A to B 402
13.3.1 Routing and IP Addresses 403
13.3.2 The Role of the Gateway 405
13.4 Data Exchange 407
13.4.1 TCP and UDP 407
13.4.2 HTTP and HTTPS 408
13.4.3 Data Interchange Formats 410
13.5 Challenges and Risks in Building IoT Systems 412
13.5.1 Security and Privacy 413
13.5.2 Dealing with IoT Scale and Complexity 415

13.6 Powering the IoT "Things" 416
Chapter Review 418
Quiz 419
References 420

PART 3: Deeper Details 421

Chapter 14: Working Directly with the Control Registers 423
14.1 Introduction: Given the Mbed OS, Why Learn about Control Registers? 423
14.2 Control Register Concepts 424
14.3 Digital I/O 426
14.3.1 LPC1768 Digital I/O Control Registers 426
14.3.2 A Digital Output Application 427
14.3.3 Adding Further Digital Outputs 429
14.3.4 Digital Inputs 431
14.4 Getting Deeper into the Control Registers 433
14.4.1 Pin Select and Pin Mode Registers 433
14.4.2 Power Control and Clock Select Registers 435
14.5 Using the DAC 437
14.5.1 LPC1768 DAC Control Registers 438
14.5.2 A DAC Application 438
14.6 Using the ADC 440
14.6.1 LPC1768 ADC Control Registers 440
14.6.2 An ADC Application 442
14.6.3 Changing ADC Conversion Speed 445
14.7 A Conclusion on Using the Control Registers 447
Chapter Review 447
Quiz 448

Chapter 15: Some Hardware Insights: Clocks, Resets, and Power Supply 449
15.1 Hardware Essentials—Power Supply 449
15.1.1 Powering the LPC1768 Microcontroller 449
15.1.2 Powering the Mbed LPC1768 451
15.2 Clock Sources and Their Selection 453
15.2.1 Some Clock Oscillator Preliminaries 453
15.2.2 LPC1768 Clock Oscillators and the Mbed Implementation 454
15.2.3 Setting the LPC1768 Clock Configuration Register 456
15.2.4 Adjusting the LPC1768 PLL 458
15.2.5 Selecting the LPC1768 Clock Source 461
15.3 Reset 462
15.3.1 Power-On Reset 463
15.3.2 Other Sources of Reset 464
15.3.3 Reason for Reset 466
15.4 Toward Low Power 468

15.4.1 How Power Is Consumed in a Digital Circuit 468
15.4.2 A Word on Cells and Batteries 470
15.4.3 Microcontroller Low-Power Modes and OS Support 472
15.5 Exploring Mbed LPC1768 Power Consumption 473
15.5.1 LPC1768 Current Consumption Characteristics 474
15.5.2 Controlling Power to Microcontroller Peripherals 476
15.5.3 Manipulating the Clock Frequency 476
15.5.4 LPC1768 Low-power Modes 477
15.5.5 Applying Operating System Sleep Modes 477
15.6 Reviewing the Nucleo F401RE Clock and Power Features 479
15.7 Getting Serious about Low Power: The M0/M0+ Cores and the Zero Gecko 480
Chapter Review 482
Quiz 483
References 484
Appendix A: Some Number Systems 485
Appendix B: Some C Essentials, with a Little C++ 493
Appendix C: Mbed LPC1768 Technical Data 521
Appendix D: Parts List 527
Appendix E: Using a Host Terminal Emulator 531
Index 535

Introduction

It's now eleven years since the first edition of this book was published, and seven years since the second. Microprocessors are of course still everywhere, providing "intelligence" in cars, mobile phones, household and office equipment, TVs and entertainment systems, medical products, aircraft—the list is endless. Those everyday products, where a little computer is hidden inside to add intelligence, are called *embedded systems*.

It is still not so long ago that designers of embedded systems had to be electronics experts, or software experts, or both. Nowadays, with user-friendly and sophisticated building blocks available for our use, both the specialist and the beginner can quickly engage in successful embedded system construction and design. One such building block was the *Mbed*, a self-contained microcontroller development board, launched in 2009 by the renowned computer giant Arm. The Mbed was the central theme of the first edition of this book, and through it all the main topics of embedded system design were introduced.

Technology has continued its onward gallop since those earlier editions, and there is already much from the second edition which needs updating, rewriting, or replacing. The single Mbed device that we based the first book on has spawned an extended family of "Mbed-enabled" devices. An "ecosystem" of products, development tools, and community support has emerged, and the phrase "internet of things" (IoT) is now on everyone's lips. The creators of the Mbed, Arm, repositioned the Mbed concept, notably by placing it at the heart of their IoT developments. The original Mbed LPC1768 device, however, is still there, widely used and well established, in industry, among hobbyists, and in colleges and universities worldwide.

Alongside hardware developments, there has been stunning growth in software techniques. As part of the Mbed ecosystem, Arm have crafted a sophisticated operating system, an essential framework through which program development takes place. While in earlier editions we tried to restrict ourselves more or less to use of the C programming language, this is now no longer possible. The Mbed operating system requires some use of C++, as more advanced applications are encountered.

In light of all these changes, we as authors made a number of decisions regarding this new edition. We have retained the Mbed LPC1768, but recognize that it's approaching the end of its working life. Therefore, in parallel, we've introduced an alternative board, the Nucleo F401RE. You can work through the first 10 chapters of the book using either of these. As if two boards aren't enough, we introduce a third, the ST IoT discovery kit. We use this primarily for Chapters 11−13, the IoT part of the book.

In terms of content, we have dropped the audio and internet chapters to make way for a pair of explicitly IoT-focused chapters. There is also a completely new chapter on using the real time operating system, or RTOS.

Broadly, the book divides into three parts. Chapters 1 to 10 provide a wide-ranging introduction to embedded systems. These chapters aim to give full support to the reader, moving you through a carefully constructed series of concepts and exercises. They start from basic principles and simple projects, and move on to more advanced system design. The next three chapters, 11−13, sit clearly in the realm of wireless and the IoT. They start with the essential wireless techniques which underlie IoT capability, before building up a complex IoT example, linking sensors through intermediate devices right up to the cloud and back! Programming here also includes some Java and JavaScript. The final two chapters, mainly based on the Mbed LPC1768 but not depending on it, look inside the microcontroller to cover more advanced or specialist topics. They explain how some of the most fundamental microcontroller activities are undertaken and how they can be controlled—for example, setting clock speed or optimizing power consumption.

All this book asks of you at the very beginning is a basic grasp of electrical/electronic theory. The book adopts a "learning through doing" approach. To get started, you will need either the LPC1768 or the F401RE development board, an internet-connected computer, and at least some of the various additional electronic components identified. You won't need every single one of these if you choose not to do a certain experiment or book section. You'll also need a digital voltmeter, and ideally use of an oscilloscope.

Each chapter is based on a major topic in embedded systems. Each has some theoretical introduction, and may have more theory within the chapter. Most chapters then proceed as series of practical experiments. Have your board ready to connect up the next circuit, and download and compile the next example program. As your confidence grows, so will your creativity and originality; you will start to turn your own ideas into working projects.

You will find that this book rapidly helps you to

- Understand and apply the key aspects of embedded systems,
- Learn from scratch, or develop your skills in embedded C/C++ programming,
- Understand and apply the key aspects of the Arm Mbed operating system,

- Develop your understanding of electronic components and configurations,
- Produce designs and innovations you never thought you were capable of!

If you are a university or college instructor, then this book offers a "complete solution" for your embedded systems course. All three authors are experienced university lecturers, and had your students in mind when writing this book. The book contains a structured sequence of practical and theoretical learning activity. Ideally you should equip every student or student pair with an Mbed development board, prototyping breadboard, and component kit. These are highly portable, so development work is not confined to the college or university lab. Later in the course students will start networking their devices together. PowerPoint presentations for each chapter are available to instructors via the book web site, as well as answers to quiz questions.

Because the need for electronic theory is limited, this book is accessible to disciplines which would not normally aim to take embedded systems. The book is meant to be accessible to Year 1 undergraduates, though we expect it will more often be used in the years which follow. Students are likely to be studying one of the branches of engineering, physics, or computer science. The book will also be of interest to the practicing professional and the hobbyist.

The first ideas for this book came from Rob Toulson. When at Anglia Ruskin University in Cambridge, he started teaching with those new-fangled Mbed devices, working in close cooperation with the Mbed design team at Arm. Tim Wilmshurst—then at the University of Derby—joined to form a writing duo, and together Rob and Tim brought the first two editions to life. Tim then proposed, and has since led, development of the third edition. A new and welcome co-author, Tom Spink, joins the team for this edition. A lecturer in computer science at the University of St Andrews, Tom brings particular expertise in IoT and the software end of things. He and Tim are also authors of Arm Education MOOCs (massive open online courses) on embedded systems and IoT. Rob meanwhile has moved on from academia to become an active entrepreneur in digital music technologies (https://www.rt60.uk/), but has continued to play a supporting role in the new edition.

Because Tim has written several books on embedded systems in the past, a few background sections and diagrams have been taken from these and adapted for inclusion in this book. There seemed no point in "reinventing the wheel" where introductory explanations were needed.

This page intentionally left blank

Acknowledgments

The authors would like to thank staff from Arm, including Donatien Garnier, Robert Iannello, David MacKenzie, and Andy Powers, for support in the development of this edition, and in earlier work on the MOOCs. Thanks also to those who develop and maintain content on the Mbed OS web site (https://os.Mbed.com/). We have made repeated use of this site and benefited from the many example programs therein.

We would further like to thank most warmly the following for their comments on draft chapters. They read and evaluated with care and attention, making valued suggestions for improvements to content or code, exposing lurking typos or technical inaccuracy, and brought their shared teaching expertise to how ideas should be presented and should flow. In alphabetical order they are Hammam Alsafrjalani of the University of Miami, Haitham Abu Ghazaleh of Tarleton State University, John Larkin of Whitworth University, Erik Petrich of the University of Oklahoma, and Atoussa Tehrani of Florida International University. Despite the value of these inputs, any errors or weaknesses in the book remain the authors' responsibility!

Thanks to Liz A of sweetclipart.com for the use of the car image in Figure 1.3.

Companion Web Site

www.embedded-knowhow.co.uk

This page intentionally left blank

PART 1

Essentials of Embedded Systems, and Developing with Mbed

This page intentionally left blank

CHAPTER 1

Embedded Systems, Microcontrollers, and Arm

1.1 Introducing Embedded Systems

1.1.1 What is an Embedded System?

We are all familiar with the idea of a desktop or laptop computer, and the amazing processing that they can do. These computers are general-purpose; we can get them to do different things at different times, depending on the application or program we run on them. At the very heart of such computers, we would find one or more *microprocessors*, tiny and fantastically complicated electronic circuits that contain the core features of a computer. These are fabricated on a single slice of silicon, called an *integrated circuit* (IC). Some people, particularly those who are not engineers themselves, call these circuits *microchips,* or just *chips.*

What is less familiar to many people is the idea that instead of putting a microprocessor into a general-purpose computer, it can also be placed inside a product which has nothing to do with computing, like a washing machine, toaster, or camera. The microprocessor is then customized to control that product. The computer is there, inside the product; but it can't be seen, and the user probably doesn't even know it's there. Moreover, those add-ons which we normally associate with a computer, like a keyboard, screen, or mouse, are nowhere to be seen. We call such products *embedded systems,* because the microprocessor that controls them is embedded inside. Because such a microprocessor is developed to control the device, in many cases those used in embedded systems have different characteristics from the ones used in more general-purpose computing machines. We end up calling these embedded computers *microcontrollers.* Though much less visible than their microprocessor cousins, microcontrollers sell in far greater volume, and their impact has been enormous. To the electronic and system designer they offer huge opportunities.

Embedded systems come in many forms and guises. They are extremely common in the home, the motor vehicle, and the workplace. Most modern domestic appliances, like a washing machine, dishwasher, oven, central heating, or burglar alarm, are embedded systems. The motor car is full of them, in engine management, security (e.g., locking and anti-theft devices), crash sensing, air-conditioning, brakes, radio, and so on. They are

Fast and Effective Embedded Systems Design. https://doi.org/10.1016/B978-0-323-95197-5.00001-2

found across industry and commerce, in machine control, factory automation, robotics, electronic commerce, and office equipment. The list has almost no end, and it continues to grow.

Fig. 1.1 expresses the embedded system as a simple block diagram. A number of features appear; not every embedded system has all of them. There are one or more inputs from the external environment, for example, sensors providing voltages proportional to physical variables. The embedded computer runs a program dedicated to this application, permanently stored in its memory. Unlike the general-purpose desktop computer, which runs many programs, this is the only program it ever runs. Based on information supplied from the inputs, the microcontroller computes certain outputs, which may be connected to actuators and other devices within the system. Alternatively, that output may simply be data, transferred by cable or wireless to another subsystem elsewhere, or displayed to a user. The user may have further interaction—for example, via a keypad.

The actual electronic circuit, including the microcontroller and any electromechanical components, is often called the *hardware*; the program running on it is often called the *software*. There must be a supply of power to keep the circuits running. One other variable will affect all that we do in embedded systems, and this is time, represented as a dominating arrow which cuts across the figure. We will need to be able to measure time, make things happen at predetermined times, generate data streams or other signals with a strong time dependence, and respond to unexpected things in a timely fashion.

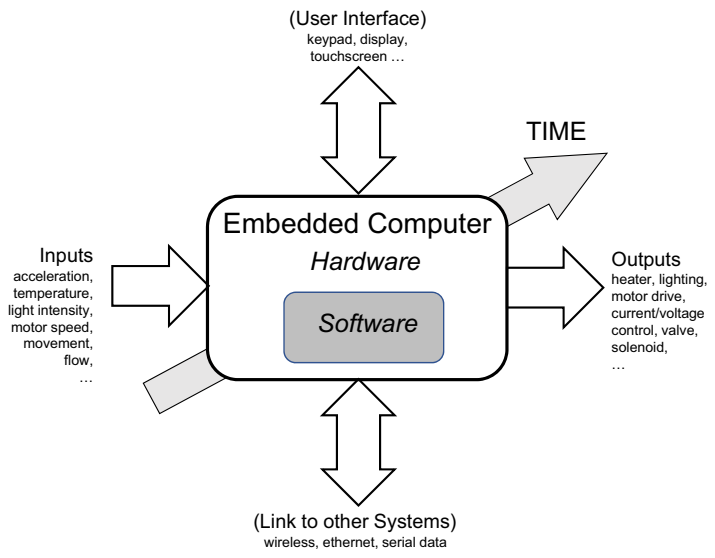


Figure 1.1
The embedded system.

A refrigerator controller provides us with an initial embedded system example. This has a temperature sensor in the main fridge compartment and one in the freezer. The user selects the required temperature or depends on default settings. Meanwhile, sensors are measuring the temperature in each fridge compartment. The embedded computer, a microcontroller, receives incoming signals from the sensors and control setting, compares these, and determines whether the compressor should be switched on or not. It will also display to the user the two temperatures measured, and may emit an alarm if a door is left open too long, or the temperature rises above a certain limit. In its traditional form the fridge does not have any external connection. However, a modern fridge may well be able to connect to the internet, so that the homeowner can check on its status while away—but more on this later.

1.1.2 Fast Forward to the Internet of Things

General-purpose computers, and now embedded systems, have become increasingly connected to the internet. The Internet of Things (IoT) is the name given to the resultant worldwide network of devices and artefacts—some everyday, some highly specialized—that are connected to the internet. These allow remote access to information that can be used to control and enhance diverse activities. The IoT generates huge quantities of real-time data that can potentially be accessed from anywhere in the world. IoT data now includes billions of sensors and databases that are made available for monitoring through the internet, from travel and transport data to environmental, medical, and industrial concerns large and small.

An integral part of the IoT is the concept of *cloud computing*, which uses servers and data memory storage locations that are only accessed through the internet. Companies such as Google, Microsoft, Apple, and Dropbox, among many others, provide their own cloud-based services, and it is anticipated that most, if not all, of our personal and professional data (and programs) will one day be stored in the cloud rather than on local computers and hard drives.

The IoT concept has expanded to include everyday objects attached to the internet. Examples include a washing machine that can alert the repair company to an impending fault, a vending machine that can tell the head office it is empty, a manufacturer who can download a new version of firmware to an installed burglar alarm, or a homeowner who can switch on the oven from the office or check that the garage door is closed.

1.1.3 The Electric Car—An Abundance of Embedded Systems

Technical developments in cars have been one of the driving forces behind the growth of embedded systems, with microcontrollers being applied in engine management, braking,

diagnostics, and many other things. Perhaps nowhere has this outpouring of embedded control been more evident than in the modern electric vehicle (EV), with battery replacing gas tank, and DC (direct current) electric motors replacing the internal combustion engine (see Fig. 1.2). Such a vehicle represents an excellent opportunity to apply and exploit embedded systems, using those little microcontrollers to their best advantage.

The possible categories of electronic control in the EV are shown in Fig. 1.3. Most of these will overlap, and other categorizations could be made. Each category contains numerous embedded systems, some controlling one or few variables, others managing a sophisticated array.

Imagine Fig. 1.1 implemented dozens of times within the EV, and think of the sensors and actuators in operation. The battery management will need to sense battery voltage and current, sometimes current flowing from battery to motors, at other times from motors back to battery due to kinetic energy recovery, then from charge point to battery when charging is taking place. Safety features will detect closure of seat belts in occupied seats, tire pressure, nearness of vehicle to potential obstacles, and lane following on a motorway or freeway. Environment control will sense internal and external temperatures and the temperature demanded by the driver and manage seat heating. Within most of these are actuators moving, heating, or lighting things they control. There is also an enormous flow of data, within and between the subsystems.

Figure 1.2
The electric car, rich in embedded systems.

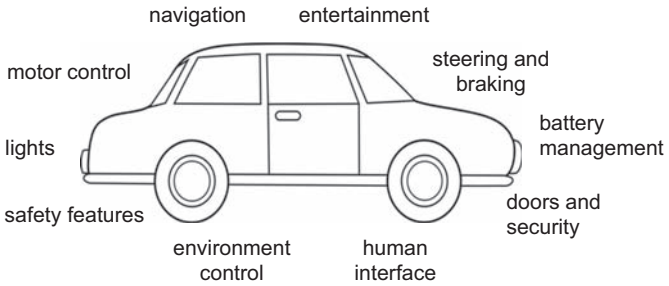


Figure 1.3
Electronic subsystems in the electric car.

Needless to say, data generated in the vehicle does not all stay within it. The owner is encouraged—indeed it's almost a necessity—to download a car-specific phone app, as seen in Fig. 1.4. This tells him or her many things, including the important features of whether the car is locked, the state of battery charge and available driving range, the

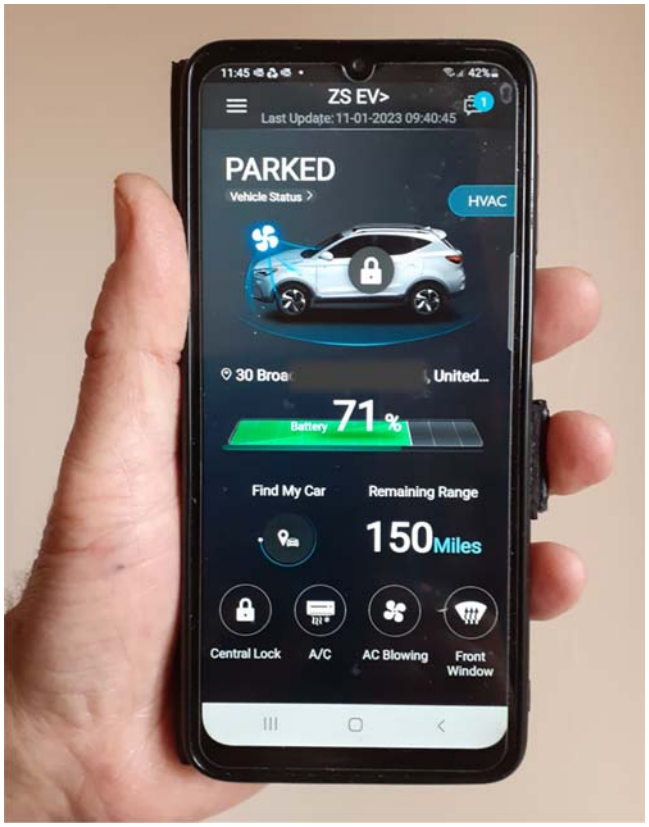


Figure 1.4
An electric car app.

vehicle state of health, and its location. From it the user can—if a charging cable is connected—start or stop charging. Also, on a cold day, the heating can be started and the windshield de-iced before a journey starts.

The actions just described place us firmly in the realm of the IoT. Data from the car has been transmitted wirelessly to the web, processed, and returned to the app, to be displayed in user-friendly form. Control data generated by the user with the app is then returned to the car for actuation.

This book takes us on a journey which—on completion—should allow you, the reader, to feel able to understand, apply, and even design the circuits and software found in embedded systems large and small. As each new chapter opens, we start at the beginning of that topic, initially with simple concepts.

This first chapter introduces or reviews many concepts relating to computers and embedded systems. It does this in overview form, to give a platform for further learning. We return to these concepts in later chapters, building on them and adding detail.

1.2 Microprocessors and Microcontrollers

Let's look more closely at the microcontroller, which sits at the heart of any embedded system. As the microcontroller is in essence a type of computer, it will be useful to get a grasp of basic computer details.

1.2.1 Some Computer Essentials

Fig. 1.5 shows the essential elements of any computer system. As its very purpose for existence, a computer can perform arithmetic or logical calculations. It does this in a

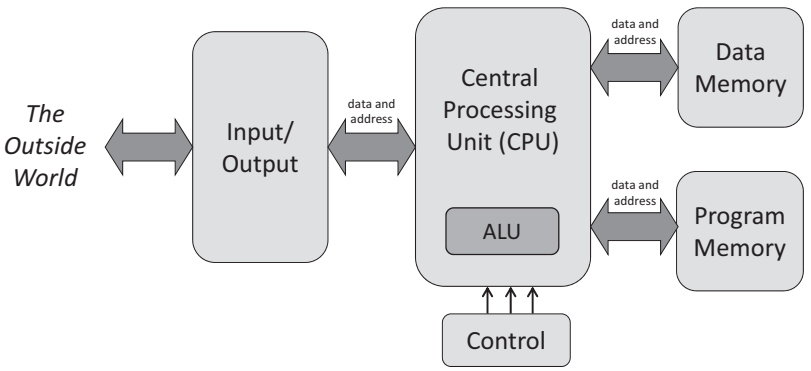


Figure 1.5
Essentials of a computer.