

Contents

	<i>List of Code examples</i>	page xi
	<i>Preface</i>	xvii
	<i>Acknowledgements</i>	xix
	<i>Programming hints, condensed</i>	xxi
	Part I TinyOS and nesC	1
1	Introduction	3
	1.1 Networked, embedded sensors	3
	1.1.1 Anatomy of a sensor node (mote)	4
	1.2 TinyOS	5
	1.2.1 What TinyOS provides	6
	1.3 Example application	7
	1.4 Compiling and installing applications	8
	1.5 The rest of this book	8
2	Names and program structure	10
	2.1 Hello World!	10
	2.2 Essential differences: components, interfaces, and wiring	13
	2.3 Wiring and callbacks	15
	2.4 Summary	16
	Part II Basic programming	19
3	Components and interfaces	21
	3.1 Component signatures	21
	3.1.1 Visualizing components	22
	3.1.2 The “as” keyword and clustering interfaces	23
	3.1.3 Clustering interfaces	24
	3.2 Interfaces	24
	3.2.1 Generic interfaces	27
	3.2.2 Bidirectional interfaces	28

vi	Contents	
		<hr/>
	3.3 Component implementations	29
	3.3.1 Modules	30
	3.3.2 A basic configuration	31
	3.3.3 Module variables	32
	3.3.4 Generic components	33
	3.4 Split-phase interfaces	34
	3.4.1 Read	36
	3.4.2 Send	36
	3.5 Module memory allocation, avoiding recursion, and other details	36
	3.5.1 Memory ownership and split-phase calls	38
	3.5.2 Constants and saving memory	41
	3.5.3 Platform-independent types	42
	3.5.4 Global names	44
	3.5.5 nesC and the C preprocessor	46
	3.5.6 C libraries	47
	3.6 Exercises	48
4	Configurations and wiring	49
	4.1 Configurations	50
	4.1.1 The \rightarrow and \leftarrow operators	51
	4.1.2 The = operator	52
	4.1.3 Namespace management	53
	4.1.4 Wiring rules	54
	4.1.5 Wiring shortcuts	56
	4.2 Building abstractions	57
	4.2.1 Component naming	58
	4.2.2 Component initialization	59
	4.3 Component layering	60
	4.3.1 Extensibility	61
	4.3.2 Hardware specificity	61
	4.4 Multiple wirings	63
	4.4.1 Fan-in and fan-out	64
	4.4.2 Uses of multiple wiring	65
	4.4.3 Combine functions	66
	4.5 Generics versus singletons	68
	4.5.1 Generic components, revisited	68
	4.5.2 Singleton components, revisited	70
	4.6 Exercises	70
5	Execution model	71
	5.1 Overview	71
	5.2 Tasks	72
	5.2.1 Task timing	74

	Contents	vii
5.2.2	Timing and event handlers	75
5.3	Tasks and split-phase calls	75
5.3.1	Hardware versus software	75
5.3.2	Tasks and call loops	76
5.4	Exercises	78
6	Applications	79
6.1	The basics: timing, LEDs, and booting	79
6.1.1	Deadline-based timing	81
6.1.2	Wiring AntiTheftC	83
6.2	Sensing	83
6.2.1	Simple sampling	84
6.2.2	Sensor components	85
6.2.3	Sensor values, calibration	86
6.2.4	Stream sampling	87
6.3	Single-hop networking	89
6.3.1	Sending packets	90
6.3.2	Receiving packets	93
6.3.3	Selecting a communication stack	94
6.4	Multi-hop networking: collection, dissemination, and base stations	95
6.4.1	Collection	96
6.4.2	Dissemination	97
6.4.3	Wiring collection and dissemination	97
6.4.4	Base station for collection and dissemination	98
6.5	Storage	101
6.5.1	Volumes	102
6.5.2	Configuration data	103
6.5.3	Block and Log storage	105
6.6	Exercises	111
7	Mote-PC communication	112
7.1	Basics	112
7.1.1	Serial communication stack	113
7.2	Using mig	114
7.2.1	Sending and receiving mig-generated packets	116
7.3	Using ncg	118
7.4	Packet sources	119
7.5	Example: simple reliable transmission	120
7.5.1	Reliable transmission protocol	121
7.5.2	Reliable transmission in Java	121
7.5.3	Reimplementing TestSerial	125
7.6	Exercises	125

Part III	Advanced programming	127
8	Advanced components	129
	8.1 Generic components review	129
	8.2 Writing generic modules	131
	8.2.1 Type arguments	132
	8.2.2 Abstract data types as generics	133
	8.2.3 ADTs in TinyOS 1.x	134
	8.3 Parameterized interfaces	135
	8.3.1 Parameterized interfaces and configurations	137
	8.3.2 Parameterized interfaces and modules	139
	8.3.3 Defaults	141
	8.4 Attributes	142
	8.5 Exercises	144
9	Advanced wiring	145
	9.1 <code>unique()</code> and <code>uniqueCount()</code>	145
	9.1.1 <code>unique</code>	146
	9.1.2 <code>uniqueCount</code>	147
	9.1.3 Example: <code>HilTimerMilliC</code> and <code>VirtualizeTimerC</code>	147
	9.2 Generic configurations	150
	9.2.1 <code>TimerMilliC</code>	150
	9.2.2 <code>CC2420SpiC</code>	152
	9.2.3 <code>AMSenderC</code>	156
	9.2.4 <code>BlockStorageC</code>	160
	9.3 Reusable component libraries	162
	9.4 Exercises	165
10	Design patterns	166
	10.1 Behavioral: Dispatcher	166
	10.2 Structural: Service Instance	170
	10.3 Namespace: Keyspace	174
	10.4 Namespace: Keymap	177
	10.5 Structural: Placeholder	180
	10.6 Structural: Facade	183
	10.7 Behavioral: Decorator	186
	10.8 Behavioral: Adapter	189
11	Concurrency	192
	11.1 Asynchronous code	192
	11.1.1 The <code>async</code> keyword	192
	11.1.2 The cost of <code>async</code>	193
	11.1.3 Atomic statements and the <code>atomic</code> keyword	195

	11.1.4 Managing state transitions	197
	11.1.5 Example: CC2420ControlP	197
	11.1.6 Tasks, revisited	199
	11.2 Power locks	200
	11.2.1 Example lock need: link-layer acknowledgements	200
	11.2.2 Split-phase locks	201
	11.2.3 Lock internals	202
	11.2.4 Energy management	203
	11.2.5 Hardware configuration	204
	11.2.6 Example: MSP430 USART	204
	11.2.7 Power lock library	205
	11.3 Exercises	205
12	Device drivers and the hardware abstraction architecture (HAA)	206
	12.1 Portability and the hardware abstraction architecture	206
	12.1.1 Examples	208
	12.1.2 Portability	210
	12.2 Device drivers	210
	12.2.1 Access control	211
	12.2.2 Access control examples	212
	12.2.3 Power management	215
	12.2.4 Microcontroller power management	218
	12.3 Fitting in to the HAA	219
13	Advanced application: SoundLocalizer	221
	13.1 SoundLocalizer design	221
	13.1.1 Time synchronization	222
	13.1.2 Implementing SoundLocalizer in TinyOS	223
	13.2 SynchronizerC	225
	13.3 DetectorC	230
	13.4 MicrophoneC	233
	13.5 Wrap-up	237
Part IV Appendix and references		239
A	TinyOS APIs	241
	A.1 Booting	241
	A.2 Communication	241
	A.2.1 Single-hop	242
	A.2.2 Multi-hop collection	243
	A.2.3 Multi-hop dissemination	244
	A.2.4 Binary reprogramming	245
	A.3 Time	245
	A.4 Sensing	245

A.5	Storage	246
A.6	Data structures	247
A.6.1	BitVectorC	247
A.6.2	QueueC	247
A.6.3	BigQueueC	248
A.6.4	PoolC	248
A.6.5	StateC	249
A.7	Utilities	249
A.7.1	Random numbers	249
A.7.2	Leds	249
A.7.3	Cyclic redundancy checks	250
A.7.4	Printf	250
A.8	Low power	251
	<i>References</i>	252
	<i>Index</i>	254