TinyOS Programming

PHILIP LEVIS
Stanford University

and

DAVID GAY
Intel Research
Contents

List of Code examples xi
Preface xvii
Acknowledgements xix
Programming hints, condensed 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
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>Component implementations</td>
<td>29</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Modules</td>
<td>30</td>
</tr>
<tr>
<td>3.3.2</td>
<td>A basic configuration</td>
<td>31</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Module variables</td>
<td>32</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Generic components</td>
<td>33</td>
</tr>
<tr>
<td>3.4</td>
<td>Split-phase interfaces</td>
<td>34</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Read</td>
<td>36</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Send</td>
<td>36</td>
</tr>
<tr>
<td>3.5</td>
<td>Module memory allocation, avoiding recursion, and other details</td>
<td>36</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Memory ownership and split-phase calls</td>
<td>38</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Constants and saving memory</td>
<td>41</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Platform-independent types</td>
<td>42</td>
</tr>
<tr>
<td>3.5.4</td>
<td>Global names</td>
<td>44</td>
</tr>
<tr>
<td>3.5.5</td>
<td>nesC and the C preprocessor</td>
<td>46</td>
</tr>
<tr>
<td>3.5.6</td>
<td>C libraries</td>
<td>47</td>
</tr>
<tr>
<td>3.6</td>
<td>Exercises</td>
<td>48</td>
</tr>
</tbody>
</table>

### 4 Configurations and wiring

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Configurations</td>
<td>50</td>
</tr>
<tr>
<td>4.1.1</td>
<td>The ( \rightarrow ) and ( \leftarrow ) operators</td>
<td>51</td>
</tr>
<tr>
<td>4.1.2</td>
<td>The ( = ) operator</td>
<td>52</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Namespace management</td>
<td>53</td>
</tr>
<tr>
<td>4.1.4</td>
<td>Wiring rules</td>
<td>54</td>
</tr>
<tr>
<td>4.1.5</td>
<td>Wiring shortcuts</td>
<td>56</td>
</tr>
<tr>
<td>4.2</td>
<td>Building abstractions</td>
<td>57</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Component naming</td>
<td>58</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Component initialization</td>
<td>59</td>
</tr>
<tr>
<td>4.3</td>
<td>Component layering</td>
<td>60</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Extensibility</td>
<td>61</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Hardware specificity</td>
<td>61</td>
</tr>
<tr>
<td>4.4</td>
<td>Multiple wirings</td>
<td>63</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Fan-in and fan-out</td>
<td>64</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Uses of multiple wiring</td>
<td>65</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Combine functions</td>
<td>66</td>
</tr>
<tr>
<td>4.5</td>
<td>Generics versus singletons</td>
<td>68</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Generic components, revisited</td>
<td>68</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Singleton components, revisited</td>
<td>70</td>
</tr>
<tr>
<td>4.6</td>
<td>Exercises</td>
<td>70</td>
</tr>
</tbody>
</table>

### 5 Execution model

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Overview</td>
<td>71</td>
</tr>
<tr>
<td>5.2</td>
<td>Tasks</td>
<td>72</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Task timing</td>
<td>74</td>
</tr>
</tbody>
</table>
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

8 Advanced components
8.1 Generic components review
8.2 Writing generic modules
  8.2.1 Type arguments
  8.2.2 Abstract data types as generics
  8.2.3 ADTs in TinyOS 1.x
8.3 Parameterized interfaces
  8.3.1 Parameterized interfaces and configurations
  8.3.2 Parameterized interfaces and modules
  8.3.3 Defaults
8.4 Attributes
8.5 Exercises

9 Advanced wiring
9.1 unique() and uniqueCount()
  9.1.1 unique
  9.1.2 uniqueCount
  9.1.3 Example: HilTimerMilliC and VirtualizeTimerC
9.2 Generic configurations
  9.2.1 TimerMilliC
  9.2.2 CC2420SpiC
  9.2.3 AMSenderC
  9.2.4 BlockStorageC
9.3 Reusable component libraries
9.4 Exercises

10 Design patterns
10.1 Behavioral: Dispatcher
10.2 Structural: Service Instance
10.3 Namespace: Keyspace
10.4 Namespace: Keymap
10.5 Structural: Placeholder
10.6 Structural: Facade
10.7 Behavioral: Decorator
10.8 Behavioral: Adapter

11 Concurrency
11.1 Asynchronous code
  11.1.1 The async keyword
  11.1.2 The cost of async
  11.1.3 Atomic statements and the atomic keyword
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