

"Loaded with helpful information and coding tutorials, this exploration around the extensive capabilities of BeagleBone Black has me excited to connect everything I encounter to the Internet."

—Christine Long, BeagleBoard.org Foundation

**DEREK MOLLOY**

# EXPLORING BEAGLEBONE®

TOOLS AND TECHNIQUES FOR  
BUILDING WITH EMBEDDED LINUX®

SECOND EDITION



WILEY



# **Exploring BeagleBone<sup>®</sup>**

**Second Edition**





# **Exploring BeagleBone<sup>®</sup>**

Tools and Techniques for Building with  
Embedded Linux<sup>®</sup>

**Second Edition**

Derek Molloy

**WILEY**

**Exploring BeagleBone®: Tools and Techniques for Building with Embedded Linux®, Second Edition**

Published by

**John Wiley & Sons, Inc.**

10475 Crosspoint Boulevard

Indianapolis, IN 46256

[www.wiley.com](http://www.wiley.com)

Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-1-119-53316-0

ISBN: 978-1-119-53315-3 (ebk)

ISBN: 978-1-119-53317-7 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

**Limit of Liability/Disclaimer of Warranty:** The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Web site is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or website may provide or recommendations it may make. Further, readers should be aware that Internet websites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services please contact our Customer Care Department within the United States at (877) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at <http://booksupport.wiley.com>. For more information about Wiley products, visit [www.wiley.com](http://www.wiley.com).

**Library of Congress Control Number:** 2018962584

**Trademarks:** Wiley and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. BeagleBone is a registered trademark of BeagleBoard.org Foundation. Linux is a registered trademark of Linus Torvalds. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

*To Sally, Daragh, Eoghan, Aidan, and Sarah*







## About the Author

**Dr. Derek Molloy** is an associate professor in the Faculty of Engineering and Computing's School of Electronic Engineering at Dublin City University, Ireland. He lectures at undergraduate and postgraduate levels in object-oriented programming with embedded systems, digital and analog electronics, and connected embedded systems. His research contributions have largely been in the fields of computer and machine vision, embedded systems, 3D graphics/visualization, and e-learning.

Derek produces a popular YouTube video series that has introduced millions of people to embedded Linux and digital electronics topics. In 2013, he launched a personal web/blog site that is visited by thousands of people every day and that integrates his YouTube videos with support materials, source code, and user discussion. In 2015, he published the first edition of this book on the BeagleBone platform, *Exploring BeagleBone*, and followed up in June 2016 with *Exploring Raspberry Pi*. Both of these books have received strong acclaim for both their depth of coverage and accessibility.

Derek has received several awards for teaching and learning. He was the winner of the 2012 Irish Learning Technology Association (ILTA) national award for Innovation in Teaching and Learning. The award recognizes his learning-by-doing approach to undergraduate engineering education, which utilizes electronic kits and online video content. In 2012, as a result of fervent nominations from his students and peers, he was also awarded the Dublin City University President's Award for Excellence in Teaching and Learning. This learning-by-doing approach is strongly reflected in his books.

You can learn more about Derek, his work, and his other publications at his personal website, [www.derekmolloy.ie](http://www.derekmolloy.ie).





## About the Technical Editor

**Marcia K. Wilbur** is a technical communicator consulting in the semiconductor field, focusing on industrial IoT (IIoT). Marcia holds degrees in computer science, technical communication, and information technology. As the Copper Linux User Group interim president, she is heavily involved in the East Valley maker community, leading regular Raspberry Pi, BeagleBone, Banana Pi/Pro, and ESP8266 projects, including home automation, gaming consoles, surveillance, network, multimedia and other “pi fun.”

In addition to tinkering, she volunteers to aid disaster-stricken areas in getting access to public domain content to enable students to continue learning. For fun, she serves the community as the lead Debian developer for Linux Respin, a backup and distro customization tool.





## Credits

**Acquisitions Assistant**

Devon Lewis

**Project Editor**

Adaobi Obi Tulton

**Technical Editor**

Marcia K. Wilbur

**Production Editor**

Barath Kumar Rajasekaran

**Copy Editor**

Kim Wimpsett

**Production Manager**

Katie Wisor

**Content Enablement and****Operations Manager**

Pete Gaughan

**Marketing Manager**

Christie Hilbrich

**Associate Publisher**

Jim Minatel

**Project Coordinator, Cover**

Brent Savage

**Proofreader**

Debbye Butler

**Indexer**

Johnna VanHoose Dinse

**Cover Designer**

Wiley

**Cover Image**

Courtesy of Derek Molloy





# Acknowledgments

Many thanks to everyone at John Wiley & Sons, Inc once again for their exceptional work on this project: to Jim Minatel for encouraging me to take on the revision of this book and for supporting the enhancement of a book that engages in deeper learning; to Devon Lewis for guiding the project forward and for his expert support and help throughout the development of this book; to Adaobi Obi Tulton, the project editor, for driving this project to completion in the most efficient way possible—it was a real pleasure to work with such an accomplished and adept editor for the third time; to Kim Wimpsett, the copy editor, for translating this book into readable U.S. English; to Barath Kumar Rajasekaran, the production editor, for bringing everything together to create a final, polished product. Thanks to the technical editor, Marcia Wilbur, for her careful review and constructive feedback on the technical content in this book. Continued thanks to the technical editors from my previous titles, Tom Betka, Robert Zhu (Microsoft), and Jason Kridner (BeagleBoard.org Foundation), on whose advice this work is based. Thanks also to Cathy Wicks (Texas Instruments) for her advice and support in the development of this book.

Continued thanks to the thousands of people who take the time to comment on my YouTube videos, blog, and website articles. I always appreciate the feedback, advice, and comments—it has really helped in the development of the topics in all of my books.

The School of Electronic Engineering at Dublin City University is a great place to work, largely because of its *esprit de corps* and its commitment to rigorous, innovative, and accessible engineering education. Thanks again to all of my colleagues in the school for supporting, encouraging, and tolerating me in the development of this book. Thanks in particular must go to Noel Murphy and Conor Brennan for sharing the workload of the school executive with me

while I was once again absorbed in a book. Thanks again to (my brother) David Molloy for his expert software advice and support. Thanks to Jennifer Bruton, Martin Collier, Pascal Landais, Michele Pringle, Robert Sadleir, Ronan Scaife, and John Whelan for their ongoing expertise, support, and advice on the various titles I have written.

The biggest thank-you must of course go to my own family once again. This revision was written over six months, predominantly at night and on weekends. Thanks to my wife, Sally, and our children, Daragh, Eoghan, Aidan, and Sarah for putting up with me (once again) while I was writing this book. Thank you, Mam, Dad, David, and Catriona for your endless lifelong inspiration, support, and encouragement. Finally, thank you to my extended family for your continued support, understanding, and constancy.





# Contents at a Glance

Introduction		xxix
Part I	Beagle Board Basics	1
Chapter 1	The Beagle Hardware Platform	3
Chapter 2	Beagle Software	31
Chapter 3	Exploring Embedded Linux Systems	71
Chapter 4	Interfacing Electronics	139
Chapter 5	Practical Beagle Board Programming	185
Part II	Interfacing, Controlling, and Communicating	245
Chapter 6	Interfacing to the Beagle Board Input/Outputs	247
Chapter 7	Cross-Compilation, Eclipse, and Building Linux	307
Chapter 8	Interfacing to the Beagle Board Buses	341
Chapter 9	Interacting with the Physical Environment	401
Chapter 10	Real-Time Interfacing Using External Slave Processors	455
Part III	Advanced Beagle Board Systems	495
Chapter 11	The Internet of Things	497
Chapter 12	Wireless Communication and Control	555
Chapter 13	Beagle Board with a Rich User Interface	599

<b>Chapter 14</b>	<b>Images, Video, and Audio</b>	<b>643</b>
<b>Chapter 15</b>	<b>Real-Time Interfacing with the PRU-ICSS</b>	<b>673</b>
<b>Chapter 16</b>	<b>Embedded Kernel Programming</b>	<b>717</b>
<b>Index</b>		<b>745</b>



# Contents

<b>Introduction</b>	<b>xxix</b>
<b>Part I      Beagle Board Basics</b>	<b>1</b>
<b>Chapter 1    The Beagle Hardware Platform</b>	<b>3</b>
Introduction to the Boards	3
Who Should Use the Beagle Platform	6
When to Use Beagle Boards	7
When Should You Not Use the Beagle Boards	7
BeagleBone Documentation	8
The Beagle Hardware	10
BeagleBone Versions	10
The Beagle Hardware	12
Beagle Accessories	19
Highly Recommended Accessories	19
Headers for the PocketBeagle	20
Micro-SD Card (for Booting or Flashing eMMCs)	20
External 5V Power Supply (for Peripherals)	22
Ethernet Cable (for Wired BBB Network Connection)	22
HDMI Cable (for Connection to Monitors/Televisions)	22
USB to Serial UART TTL 3.3 (for Finding Problems)	23
Optional Accessories	24
USB Hub (to Connect Several USB Devices to a USB Host)	25
Micro-HDMI to VGA Adapters (for VGA Video and Sound)	25
Wi-Fi Adapters (for Wireless Networking)	25
USB Webcam (for Capturing Images and Streaming Video)	25
USB Keyboard and Mouse (for General-Purpose Computing)	26
Capes	26
How to Destroy Your Board!	27
Summary	29
Support	29

<b>Chapter 2</b>	<b>Beagle Software</b>	<b>31</b>
	Linux on the Beagle Boards	32
	Linux Distributions for Beagle Boards	32
	Create a Linux Micro-SD Card Image	33
	Communicating with the Boards	34
	Installing Drivers	34
	Wired Network Connections	35
	Internet-over-USB (All Boards)	36
	Regular Ethernet (BBB and BeagleBoard Only)	39
	Ethernet Crossover Cable (BBB and BeagleBoard Only)	40
	Communicating with Your Board	42
	Serial Connection over USB	42
	Serial Connection with the USB-to-TTL 3.3 V Cable	43
	Connecting Through Secure Shell	44
	Secure Shell Connections Using PuTTY	45
	Chrome Apps: Secure Shell Client	45
	Transferring Files Using PuTTY/psftp over SSH	46
	Controlling the Beagle Board	48
	Basic Linux Commands	48
	First Steps	49
	Basic File System Commands	50
	Environment Variables	52
	Basic File Editing	53
	What Time Is It?	54
	Package Management	56
	Beagle-Specific Commands	58
	Expand the File System on an SD Card	59
	Update the Kernel	60
	Interacting with the On-Board LEDs	61
	Shutdown	63
	Node.js, Cloud9, and BoneScript	64
	Introduction to Node.js	64
	Introduction to the Cloud9 IDE	66
	Introduction to BoneScript	67
	Summary	69
	Further Reading	69
<b>Chapter 3</b>	<b>Exploring Embedded Linux Systems</b>	<b>71</b>
	Introducing Embedded Linux	72
	Advantages and Disadvantages of Embedded Linux	73
	Is Linux Open Source and Free?	74
	Booting the Beagle Boards	74
	Bootloaders	74
	Kernel Space and User Space	83
	The systemd System and Service Manager	85
	Managing Linux Systems	90
	The Superuser	90

System Administration	92
The Linux File System	92
Links to Files and Directories	94
Users and Groups	95
File System Permissions	98
The Linux Root Directory	102
Commands for File Systems	103
The Reliability of SD Card/eMMC File Systems	111
Linux Commands	113
Output and Input Redirection (>, >>, and <)	113
Pipes (  and tee)	114
Filter Commands (from sort to xargs)	115
echo and cat	117
diff	118
tar	119
md5sum	120
Linux Processes	121
How to Control Linux Processes	121
Foreground and Background Processes	122
Other Linux Topics	124
Using Git for Version Control	124
A Practice-Based Introduction	126
Cloning a Repository (git clone)	126
Getting the Status (git status)	128
Adding to the Staging Area (git add)	128
Committing to the Local Repository (git commit)	129
Pushing to the Remote Repository (git push)	129
Git Branching	130
Creating a Branch (git branch)	130
Merging a Branch (git merge)	132
Deleting a Branch (git branch -d)	132
Common Git Commands	133
Desktop Virtualization	134
Code for This Book	135
Summary	136
Further Reading	136
Bibliography	137
<b>Chapter 4    Interfacing Electronics</b>	<b>139</b>
Analyzing Your Circuits	140
Digital Multimeter	140
Oscilloscopes	141
Basic Circuit Principles	143
Voltage, Current, Resistance, and Ohm's Law	143
Voltage Division	145
Current Division	146

Implementing Circuits on a Breadboard	147
Digital Multimeters and Breadboards	149
Example Circuit: Voltage Regulation	150
Discrete Components	152
Diodes	152
Light-Emitting Diodes	153
Smoothing and Decoupling Capacitors	156
Transistors	158
Transistors as Switches	159
Field Effect Transistors as Switches	162
Optocouplers/Optoisolators	164
Switches and Buttons	166
Hysteresis	168
Logic Gates	169
Floating Inputs	173
Pull-Up and Pull-Down Resistors	173
Open-Collector and Open-Drain Outputs	174
Interconnecting Gates	175
Analog-to-Digital Conversion	177
Sampling Rate	177
Quantization	178
Operational Amplifiers	178
Ideal Operational Amplifiers	178
Negative Feedback and Voltage Follower	181
Positive Feedback	181
Concluding Advice	182
Summary	182
Further Reading	183
<b>Chapter 5 Practical Beagle Board Programming</b>	<b>185</b>
Introduction	186
Performance of Different Languages	186
Setting the CPU Frequency	190
Scripting Languages	192
Scripting Language Options	192
Bash	193
Lua	196
Perl	197
Python	198
Dynamically Compiled Languages	201
JavaScript and Node.js on the Beagle boards	201
Java on the Beagle Boards	203
C and C++ on the Beagle Boards	207
C and C++ Language Overview	210
Compiling and Linking	211
Writing the Shortest C/C++ Program	213
Static and Dynamic Compilation	215
Variables and Operators in C/C++	215

Pointers in C/C++	219
C-Style Strings	221
LED Flashing Application in C	223
The C of C++	224
First Example and Strings in C++	225
Passing by Value, Pointer, and Reference	226
Flashing the LEDs Using C++ (non-OO)	227
Writing a Multicall Binary	228
Overview of Object-Oriented Programming	229
Classes and Objects	229
Encapsulation	230
Inheritance	231
Object-Oriented LED Flashing Code	233
Interfacing to the Linux OS	236
Glibc and Syscall	237
Improving the Performance of Python	239
Cython	239
Boost.Python	242
Summary	244
Further Reading	244
Bibliography	244
<b>Part II</b>	<b>Interfacing, Controlling, and Communicating</b>
<b>Chapter 6</b>	<b>Interfacing to the Beagle Board Input/Outputs</b>
General-Purpose Input/Outputs	248
Introduction to GPIO Interfacing	248
GPIO Digital Output	250
GPIO Digital Input	255
GPIO Configuration	257
Internal Pull-Up and Pull-Down Resistors	258
GPIO Pin Configuration Settings	258
Interfacing to Powered DC Circuits	265
C++ Control of GPIOs	267
The Linux Device Tree	271
Flattened Device Tree on the Beagle Boards	272
Modifying a Board Device Tree	276
Boot Configuration Files	278
Analog Inputs and Outputs	280
Analog Inputs	280
Enabling the Analog Inputs	280
Analog Input Application—A Simple Light Meter	282
Analog Outputs (PWM)	285
Output Application—Controlling a Servo Motor	289
BoneScript	290
Digital Read and Write	290
Analog Read	292

Analog Write (PWM)	293
GPIO Performance	294
Advanced GPIO Topics	295
More C++ Programming	295
Callback Functions	295
POSIX Threads	297
Linux poll (sys/poll.h)	298
Enhanced GPIO Class	299
Using GPIOs without Using sudo	302
Root Permissions with setuid	304
Summary	306
Further Reading	306
<b>Chapter 7 Cross-Compilation, Eclipse, and Building Linux</b>	<b>307</b>
Setting Up a Cross-Compilation Toolchain	308
Cross-Compiling Under Debian	309
Testing the Toolchain	311
Emulating the armhf Architecture	312
Cross-Compilation with Third-Party Libraries (Multiarch)	314
Cross-Compilation Using Eclipse	315
Installing Eclipse on Desktop Linux	315
Configuring Eclipse for Cross-Compilation	316
Remote System Explorer	318
Integrating GitHub into Eclipse	322
Remote Debugging	322
Automatic Documentation (Doxygen)	328
Adding Doxygen Editor Support in Eclipse	330
Cross-Building Linux	330
Downloading the Kernel Source	331
Building the Linux Kernel	332
Building a Poky Linux Distribution (Advanced)	335
Summary	340
<b>Chapter 8 Interfacing to the Beagle Board Buses</b>	<b>341</b>
Introduction to Bus Communication	342
I <sup>2</sup> C	343
I <sup>2</sup> C Hardware	343
I <sup>2</sup> C on the Beagle Boards	344
I <sup>2</sup> C Devices on the Beagle Boards	345
An I <sup>2</sup> C Test Circuit	346
A Real-Time Clock	346
The ADXL345 Accelerometer	347
Wiring the Test Circuit	348
Using Linux I2C-Tools	348
i2cdetect	348
i2cdump	349
i2cget	353
i2cset	354



I <sup>2</sup> C Communication in C	356
Wrapping I <sup>2</sup> C Devices with C++ Classes	358
SPI	360
SPI Hardware	361
SPI on the Beagle Boards	363
Testing an SPI Bus	363
A First SPI Application (74HC595)	365
Wiring the 74HC595 Circuit	366
SPI Communication Using C	367
Bidirectional SPI Communication in C/C++	370
The ADXL345 SPI Interface	370
Connecting the ADXL345 to the Beagle Boards	372
Wrapping SPI Devices with C++ Classes	373
Three-Wire SPI Communication	375
Multiple SPI Slave Devices	376
UART	377
The Beagle Board UART	378
UART Examples in C	380
Beagle Board Serial Client	381
LED Serial Server	383
UART Applications: GPS	386
CAN Bus	388
Beagle Board CAN Bus	389
SocketCAN	390
A CAN Bus Test Circuit	392
Linux CAN-utils	393
A SocketCAN C Example	394
Logic-Level Translation	396
Summary	398
Further Reading	399
<b>Chapter 9    Interacting with the Physical Environment</b>	<b>401</b>
Interfacing to Actuators	402
DC Motors	403
Driving Small DC Motors (up to 1.5A)	406
Controlling a DC Motor Using sysfs	407
Driving Larger DC Motors (Greater Than 1.5 A)	409
Controlling a DC Motor Using C++	411
Stepper Motors	412
The EasyDriver Stepper Motor Driver	413
A Beagle Board Stepper Motor Driver Circuit	414
Controlling a Stepper Motor Using C++	415
Relays	417
Interfacing to Analog Sensors	418
Protecting the ADC Inputs	420
Diode Clamping	421
Op-Amp Clamping	422

	Analog Sensor Signal Conditioning	427
	Scaling Using Voltage Division	427
	Signal Offsetting and Scaling	428
	Analog Interfacing Examples	431
	Infrared Distance Sensing	431
	ADXL335 Conditioning Example	436
	Interfacing to Local Displays	438
	MAX7219 Display Modules	438
	Character LCD Modules	441
	Building C/C++ Libraries	445
	Makefiles	446
	CMake	447
	A Hello World Example	448
	Building a C/C++ Library	449
	Using a Shared (.so) or Static (.a) Library	452
	Summary	453
	Further Reading	454
<b>Chapter 10</b>	<b>Real-Time Interfacing Using External Slave Processors</b>	<b>455</b>
	Real-Time Beagle Board	456
	Real-Time Kernels	456
	Real-Time Hardware Solutions	458
	Extended GPIO Availability	458
	The MCP23017 and the I <sup>2</sup> C Bus	460
	Controlling the GPIO LED Circuit	461
	Reading the GPIO Button State	462
	An Interrupt Configuration Example (Advanced)	463
	The MCP23S17 and the SPI Bus	464
	A C++ Class for the MCP23x17 Devices	465
	Adding External UARTs	468
	The Arduino	471
	An Arduino Serial Slave	474
	A UART Echo Test Example	475
	UART Command Control of an Arduino	478
	An Arduino I <sup>2</sup> C Slave	481
	An I <sup>2</sup> C Test Circuit	481
	I <sup>2</sup> C Register Echo Example	482
	I <sup>2</sup> C Temperature Sensor Example	484
	I <sup>2</sup> C Temperature Sensor with a Warning LED	486
	Arduino Slave Communication Using C/C++	488
	An I <sup>2</sup> C Ultrasonic Sensor Application	490
	Summary	493
	Further Reading	493
<b>Part III</b>	<b>Advanced Beagle Board Systems</b>	<b>495</b>
<b>Chapter 11</b>	<b>The Internet of Things</b>	<b>497</b>
	The Internet of Things	498
	A Beagle Board IoT Sensor	499
	The Beagle Board as a Sensor Web Server	501

Installing and Configuring a Web Server	502
Configuring the Apache Web Server	503
Creating Web Pages and Web Scripts	503
PHP on the Beagle Board	506
GNU Cgicc Applications (Advanced)	508
Replacing Bone101 with Apache	511
A C/C++ Web Client	512
Network Communications Primer	513
A C/C++ Web Client	514
Secure Communication Using OpenSSL	516
A Beagle Board as a “Thing”	518
ThingSpeak	518
The Linux Cron Scheduler	521
System crontab	521
User crontab	523
Sending E-mail from the Beagle Board	524
If This Then That	526
IoT Frameworks	528
MQ Telemetry Transport	529
MQTT Server/Broker	531
MQTT Publisher/Subscriber on a Beagle Board	533
The mqtt-spy Debug Tool	534
Writing MQTT Code	535
A Paho MQTT Publisher Example	535
A Paho MQTT Subscriber Example	537
Adafruit IO	539
Configuring the Adafruit IO Account	540
Connecting to Adafruit IO with MQTT	542
An MQTT Node.js Publish Example	543
The C++ Client/Server	545
IoT Device Management	548
Remote Monitoring of a Beagle Board	548
Beagle Board Watchdog Timers	549
Static IP Addresses	551
Power over Ethernet	551
PoE Power Extraction Modules (Advanced Topic)	553
Summary	554
<b>Chapter 12    Wireless Communication and Control</b>	<b>555</b>
Introduction to Wireless Communications	556
Bluetooth Communications	557
Installing a Bluetooth Adapter	558
Checking the LKM	559
Configuring a Bluetooth Adapter	560
Making the Beagle Board Discoverable	561
Android App Development with Bluetooth	563
Wi-Fi Communications	564
Installing a Wi-Fi Adapter	564

The NodeMCU Wi-Fi Slave Processor	568
Flashing with the Latest Firmware	569
Connecting the NodeMCU to Wi-Fi	570
Programming the NodeMCU	571
The NodeMCU Web Server Interface	574
JSON	575
The NodeMCU and MQTT	577
ZigBee Communications	579
Introduction to XBee Devices	579
AT versus API Mode	581
XBee Configuration	582
XCTU	582
Configuring an XBee Network Using XCTU	583
An XBee AT Mode Example	584
Setting Up the Arduino XBee Device (XBeeA)	584
Setting Up the PocketBeagle XBee Device (XBeePB)	586
An XBee API Mode Example	589
Setting Up the PocketBeagle XBee Device (XBee1)	589
Setting Up the Stand-Alone XBee Device (XBee2)	589
XBee API Mode and Node.js	590
XBee and C/C++	592
Near Field Communication	593
Summary	596
<b>Chapter 13 Beagle Board with a Rich User Interface</b>	<b>599</b>
Rich UI Beagle Board Architectures	600
Beagle Boards as General-Purpose Computers	601
Connecting a Bluetooth Input Peripheral	603
BeagleBone with a LCD Touchscreen Cape	604
Virtual Network Computing	605
VNC Using VNC Viewer	605
VNC with Xming and PuTTY	606
VNC with a Linux Desktop Computer	607
Fat-Client Applications	608
Rich UI Application Development	608
Introduction to GTK+ on the Beagle Boards	609
The “Hello World” GTK+ Application	609
The Event-Driven Programming Model	610
The GTK+ Temperature Application	611
Introduction to Qt for the Beagle Board	612
Installing Qt Development Tools	613
The “Hello World” Qt Application	613
Qt Primer	615
Qt Concepts	615
The QObject Class	617
Signals and Slots	617
Qt Development Tools	618

A First Qt Creator Example	620
A Qt Temperature Sensor GUI Application	621
Remote UI Application Development	625
Fat-Client Qt GUI Application	626
Multithreaded Server Applications	629
A Multithreaded Temperature Service	632
Parsing Stream Data	634
The Fat Client as a Server	635
Parsing Stream Data with XML	638
The Beagle Board Client Application	639
Summary	641
Further Reading	641
<b>Chapter 14    Images, Video, and Audio</b>	<b>643</b>
Capturing Images and Video	644
USB Webcams	644
Video4Linux2 (V4L2)	646
Image Capture Utility	647
Video4Linux2 Utilities	648
Writing Video4Linux2 Programs	650
Streaming Video	652
Image Processing and Computer Vision	654
Image Processing with OpenCV	654
Computer Vision with OpenCV	656
Boost	659
BeagleBone Audio	660
Core Audio Software Tools	661
Audio Devices for the Beagle Boards	661
HDMI and USB Audio Playback Devices	661
Internet Radio Playback	664
Recording Audio	664
Audio Network Streaming	666
Bluetooth A2DP Audio	666
Text-to-Speech	669
Summary	670
Further Reading	670
<b>Chapter 15    Real-Time Interfacing with the PRU-ICSS</b>	<b>673</b>
The PRU-ICSS	674
The PRU-ICSS Architecture	674
The Remote Processor Framework	675
Important Documents	676
Development Tools for the PRU-ICSS	676
The PRU Code Generation Tools	677
The PRU Debugger	677
Using the AM335x PRU-ICSS	679
Setting Up the Board for Remoteproc	679
Testing Remoteproc under Linux	680

A First PRU Example	683
PRU-ICSS Enhanced GPIOs	683
A First PRU Program	686
A First PRU Program in C	686
A First PRU Program in Assembly	688
The PRU-ICSS in Detail	691
Registers	691
Local and Global Memory	692
PRU Assembly Instruction Set	696
PRU-ICSS Applications	698
PRU-ICSS Performance Tests	698
Utilizing Regular Linux GPIOs	702
A PRU PWM Generator	704
A PRU Sine Wave Generator	708
An Ultrasonic Sensor Application	709
Summary	714
Further Reading	714
<b>Chapter 16 Embedded Kernel Programming</b>	<b>717</b>
Introduction	718
Why Write Kernel Modules?	718
Loadable Kernel Module Basics	719
A First LKM Example	720
The LKM Makefile	722
Building the LKM on a Beagle Board	723
Testing the First LKM Example	724
Testing the LKM Parameter	726
An Embedded LKM Example	727
Interrupt Service Routines	729
Performance	733
Enhanced Button GPIO Driver LKM	733
The kobject Interface	734
Enhanced LED GPIO Driver LKM	741
Kernel Threads	742
Conclusions	744
Summary	744
<b>Index</b>	<b>745</b>



# Introduction

The Beagle platform continues to amaze! Given the proliferation of smartphones, the idea of holding in one hand a computer that is capable of performing two billion instructions per second is easy to take for granted—but the fact that you can modify the hardware and software of such small yet powerful devices and adapt them to suit your own needs and create your own inventions is nothing short of amazing. Even better, you can purchase a board for as little as \$25 in the form of a PocketBeagle.

The Beagle boards on their own are too complex to be used by a general audience; it is the capability of the boards to run Linux that makes the resulting platform accessible, adaptable, and powerful. Together, Linux and embedded systems enable ease of development for devices that can meet future challenges in smart buildings, the Internet of Things (IoT), robotics, smart energy, smart cities, human-computer interaction (HCI), cyber-physical systems, 3D printing, smart manufacturing, interactive art, advanced vehicular systems, and many, many more applications.

The integration of high-level Linux software and low-level electronics represents a paradigm shift in embedded systems development. It is revolutionary that you can build a low-level electronics circuit and then install a Linux web server, using only a few short commands, so that the circuit can be controlled over the internet. You can easily use a Beagle board as a general-purpose Linux computer, but it is vastly more challenging and interesting to get underneath the hood and fully interface it to electronic circuits of your own design—and that is where this book comes in!

This book should have widespread appeal for inventors, makers, students, entrepreneurs, hackers, artists, dreamers—in short, anybody who wants to bring the power of embedded Linux to his or her products, inventions, creations, or projects and truly understand the Beagle platform in detail. This is not a recipe

book—with few exceptions, everything demonstrated here is explained at a level that will enable you to design, build, and debug your own extensions of the concepts presented here. Nor is there any grand design project at the end of this book for which you must purchase a prescribed set of components and peripherals to achieve a specific outcome. Rather, this book is about providing you with enough background knowledge and “under-the-hood” technical details to enable and motivate your own explorations.

I strongly believe in learning by doing, so I present examples using low-cost, widely available hardware so that you can follow along. Using these hands-on examples, I describe what each step means in detail so that when you substitute your own hardware components, modules, and peripherals you will be able to adapt the content in this book to suit your needs. As for that grand project or invention—that is left up to you and your imagination!

When writing this book, I had the following aims and objectives:

- To explain embedded Linux and its interaction with electronic circuits—taking you through the topics from mystery to mastery!
- To provide in-depth information and instruction on the Linux, electronics, and programming skills that are required to master a pretty wide and comprehensive variety of topics in this domain.
- To create a collection of practical “Hello World” hardware and software examples on each and every topic in the book, from low-level interfacing, general-purpose input/outputs (GPIOs), analog-to-digital converters (ADCs), buses, and UARTs, to high-level libraries such as OpenCV, Qt, and complex and powerful topics, such as real-time interfacing with the PRU-ICSS, and Linux kernel programming.
- To ensure that each circuit and segment of code is specifically designed to work with a Beagle board. Every circuit and code example in this book was built and tested on the BeagleBone Black wireless and PocketBeagle boards.
- To use the “Hello World” examples to build a library of code that you can use and adapt for your own Beagle projects.
- To make all of the code available on GitHub in an easy-to-use form.
- To support this book with strong digital content, such as the videos on the DerekMolloyDCU YouTube channel, and a custom website, [www.exploringbeaglebone.com](http://www.exploringbeaglebone.com).
- To ensure that by the end of this book you have everything you need to imagine, create, and build *advanced* Beagle board projects.



I wrote this second edition because of the popularity of the first edition of *Exploring BeagleBone*. The number of pages in this edition is more than 20 percent of the first edition, increased to include the following major additions:

- Full coverage of new Beagle boards, with a particular emphasis on the PocketBeagle and BeagleBone Black wireless boards
- Updated content to account for all recent changes to the Linux kernel and operating system
- Inclusion of electronics interfacing approaches, such as protection of I/O pins using optocouplers, the CAN bus, and many additional interfacing application examples using external I/O circuits
- New work on real-time interfacing using external slave processors, with a particular emphasis on building I<sup>2</sup>C digital sensors
- A full account of new Internet of Things (IoT) full-stack frameworks, with an emphasis on MQTT and interfacing to Adafruit IO
- Full coverage of building wireless sensor networks using technologies such as Wi-Fi, Bluetooth, NFC, and ZigBee
- A complete rewrite of the PRU-ICSS chapter to account for Texas Instruments' decision to move away from UIO to Linux Remoteproc
- Inclusion of new work on writing Linux loadable kernel modules (LKMs)

## Why the BeagleBone and PocketBeagle?

---

The Beagle boards are powerful single-board computers (SBCs), and while there are other SBCs available on the market, such as the Raspberry Pi and Intel NUC boards, the Beagle platform has one key differentiator—it was built to be interfaced to! For example, the Beagle board's microprocessor package even contains two additional on-chip microcontrollers that can be used for real-time interfacing—an area in which other Linux SBCs have significant difficulty.

Unlike most other SBCs, the Beagle boards are fully open-source hardware. The BeagleBoard.org Foundation provides source schematics, hardware layout, a full bill of materials, and comprehensive technical reference manuals, enabling you to modify the design of the Beagle platform and integrate it into your own product. In fact, you can even fork the hardware design onto Upverter ([www.upverter.com](http://www.upverter.com)) under a Creative Commons Attribution-ShareAlike license (see [tiny.cc/beagle001](http://tiny.cc/beagle001) for the full schematics). This is a useful feature should you decide to take your newest invention to market!

## How This Book Is Structured

---

There is no doubt that some of the topics in this book are quite complex—the Beagle boards are complex devices! However, everything that you need to master the devices is present in the book within three major parts.

- Part I, “Beagle Board Basics”
- Part II, “Interfacing, Controlling, and Communicating”
- Part III, “Advanced Beagle Board Systems”

In the first part in the book, you learn about the hardware and software of the Beagle board platform in Chapters 1 and 2 and subsequently gain more knowledge through these three primer chapters:

- Chapter 3, “Exploring Embedded Linux Systems”
- Chapter 4, “Interfacing Electronics”
- Chapter 5, “Practical Beagle Board Programming”

If you are a Linux expert, electronics wizard, and/or software guru, then feel free to skip the primer chapters; however, for everyone else, you’ll find a concise but detailed set of materials to ensure that you gain all the knowledge required to effectively and safely interface to your Beagle boards.

The second part of the book, Chapters 6 to 10, provides detailed information on interfacing to the Beagle board GPIOs, analog inputs, buses (I<sup>2</sup>C, SPI, CAN bus), UART devices, USB peripherals, and real-time interfacing to slave processors. You’ll learn how you can configure a cross-compilation environment so that you can build large-scale software applications. This part also describes how you can combine hardware and software to provide your board with the ability to interact effectively with its physical environment.

The final part of the book, Chapters 11 to 16, describes how the Beagle board can be used for advanced applications such as Internet of Things (IoT); rich user interfaces; images, video, and audio; real-time interfacing using the PRU-ICSS; and kernel programming. Along the way you will meet many technologies, including TCP/IP, ThingSpeak, Adafruit IO, PoE, Wi-Fi, Bluetooth, Zigbee, RFID, MQTT, cron, Apache, PHP, e-mail, IFTTT, VNC, GTK+, Qt, XML, JSON, multi-threading, client/server programming, V4L2, video streaming, OpenCV, Boost, USB audio, Bluetooth A2DP, text-to-speech, and Remoteproc.

## Conventions Used in This Book

---

This book is filled with source code examples and snippets that you can use to build your own applications. Code and commands are shown as follows:

```
This is what source code looks like.
```

When presenting work performed in a Linux terminal, it is often necessary to display both input and output in a single example. A bold type is used to distinguish the user input from the output. Here's an example:

```
debian@ebb:~$ ping www.exploringbeaglebone.com
PING lb1.reg365.net (195.7.226.20) 56(84) bytes of data.
64 bytes from lb1.reg365.net (195.7.226.20): icmp_req=1 ttl=55 time=25.6 ms
64 bytes from lb1.reg365.net (195.7.226.20): icmp_req=2 ttl=55 time=25.6 ms
...
```

The `$` prompt indicates that a regular Linux user is executing a command, and a `#` prompt indicates that a Linux superuser is executing a command. The ellipsis symbol (...) is used whenever code or output not vital to understanding a topic has been cut. I've edited the output like this to enable you to focus on only the most useful information. You are encouraged to repeat the steps in this book yourself, whereupon you will see the full output. In addition, the full source code for all examples is provided along with the book.

There are some additional styles in the text. Here are some examples:

- New terms and important words appear in *italics* when introduced.
- Keyboard strokes appear like this: Ctrl+C.
- All URLs in the book appear in this font: `www.exploringbeaglebone.com`.
- A URL-shortening service is used to create aliases for long URLs that are presented in the book. These aliases have the form `tiny.cc/beagle102` (e.g., link 2 in Chapter 1). Should the link address change after this book is published, the alias will be updated.

There are several features used in this book to identify when content is of particular importance or when additional information is available.

**WARNING** This type of feature contains important information that can help you avoid damaging your Beagle board.

**NOTE** This type of feature contains useful additional information, such as links to digital resources and useful tips, which can make it easier to understand the task at hand.

#### FEATURE TITLE

This type of feature goes into detail about the current topic or a related topic.

## What You'll Need

---

Ideally you should have a Beagle board before you begin reading this book so that you can follow along with the numerous examples in the text. If you do not yet have a board, it would be worth reading Chapter 1 before placing an order. Currently the board is manufactured by both CircuitCo and Embest—the boards from either manufacturer are compatible with the designs and operations in this book. You can purchase one of the boards in the United States from online stores such as Adafruit Industries, Digi-Key, Mouser, SparkFun, and Jameco Electronics. They are available internationally from stores such as Farnell, Radionics, Watterott, and Tegal.

A full list of recommended and optional accessories for the Beagle platform is provided in Chapter 1. In addition, each chapter contains a list of the electronics components and modules required if you want to follow along with the text. The book website provides details about where these components can be acquired.

## Errata

---

I have worked really hard to ensure that this book is error free; however, it is always possible that something was overlooked. A full list of errata is available on each chapter's web page at the companion website. If you find any errors in the text or in the source code examples, I would be grateful if you could send the errors using the companion website so that I can update the web page errata list and the source code examples in the code repository.

## Digital Content and Source Code

---

The primary companion site for this book is [www.exploringbeaglebone.com](http://www.exploringbeaglebone.com). It contains videos, source code examples, and links to further reading. Each chapter has its own individual web page. In the unlikely event that this website is unavailable, you can find the code at [www.wiley.com/go/exploringbeaglebone2e](http://www.wiley.com/go/exploringbeaglebone2e).

All the source code is available through GitHub, which allows you to download the code to your Beagle board with one command. You can also easily view the code online at [tiny.cc/beagle002](http://tiny.cc/beagle002). Downloading the source code to your board is as straightforward as typing the following at the Linux shell prompt:

```
debian@ebb:~$ git clone https://github.com/derekmolloy/exploringbb.git
```

If you have never used Git before, don't worry—it is explained in detail in Chapter 3. Now, on with the adventures!

---

# **Beagle Board Basics**

---

## **In This Part**

**Chapter 1:** The Beagle Hardware Platform

**Chapter 2:** Beagle Software

**Chapter 3:** Exploring Embedded Linux Systems

**Chapter 4:** Interfacing Electronics

**Chapter 5:** Practical Beagle Board Programming



# The Beagle Hardware Platform

In this chapter, you are introduced to the BeagleBone platform hardware and its variant boards. The chapter focuses in particular on the BeagleBone and PocketBeagle boards and the various subsystems and physical inputs/outputs of these boards. In addition, the chapter lists accessories that can be helpful in developing your own Beagle-based projects. By the end of this chapter, you should have an appreciation of the power and complexity of this computing platform. You should also be aware of the first steps to take to protect your boards from physical damage.

## Introduction to the Boards

---

Beagle boards are compact, low-cost, open-source Linux computing platforms that can be used to build complex applications that interface high-level software and low-level electronic circuits. These are ideal platforms for prototyping project and product designs that take advantage of the power and freedom of Linux, combined with direct access to input/output pins and buses, allowing

you to interface with electronics components, modules, and USB devices. The following are some characteristics of the single-board computing (SBC) boards:

- They are powerful, containing a processor that can perform up to 2 billion instructions per second.
- They are widely available at relatively low-cost, as little as \$25–\$90 depending on the board chosen.
- They support many standard interfaces for electronics devices.
- They use little power, running at between 1W (idle) and 2.3W (peak).
- They are expandable through the use of daughter boards and USB devices.
- They are strongly supported by a huge community of innovators and enthusiasts.
- They are open-hardware and support open-software tools and applications for commercial and noncommercial applications.

The BeagleBone and PocketBeagle boards run the Linux operating system, which means you can use many open-source software libraries and applications directly with them. Open-source software driver availability also enables you to interface devices such as USB cameras, keyboards and Wi-Fi adapters with your project, without having to source proprietary alternatives. Therefore, you have access to comprehensive libraries of code that have been built by a talented open-source community; however, it is important to remember that the code typically comes without any type of warranty or guarantee. If there are problems, then you have to rely on the good nature of the community to resolve them. Of course, you could also fix the problems yourself and make the solutions publicly available.

**NOTE** The BeagleBone and PocketBeagle boards are quite different in physical appearance, as displayed in Figure 1-1, but they are similar devices under the hood. To illustrate this, both boards are typically booted with the same Linux image on a micro-SD card. The Linux image will automatically detect and configure the differing hardware during the boot sequence depending on the board it is booting.

The BeagleBoard.org Foundation is a U.S. nonprofit corporation that aims to provide embedded systems education in open-source hardware and software. Over the last ten years, the Foundation has developed high-quality boards that are renowned in the open-source community for their detailed documentation, for their extensive support, and for providing a strong bridge between idea prototyping and commercial product design.

The platform boards are formed by the integration of a high-performance microprocessor on a printed circuit board (PCB) and an extensive software ecosystem. The physical PCB is not a complete product; rather, it is a