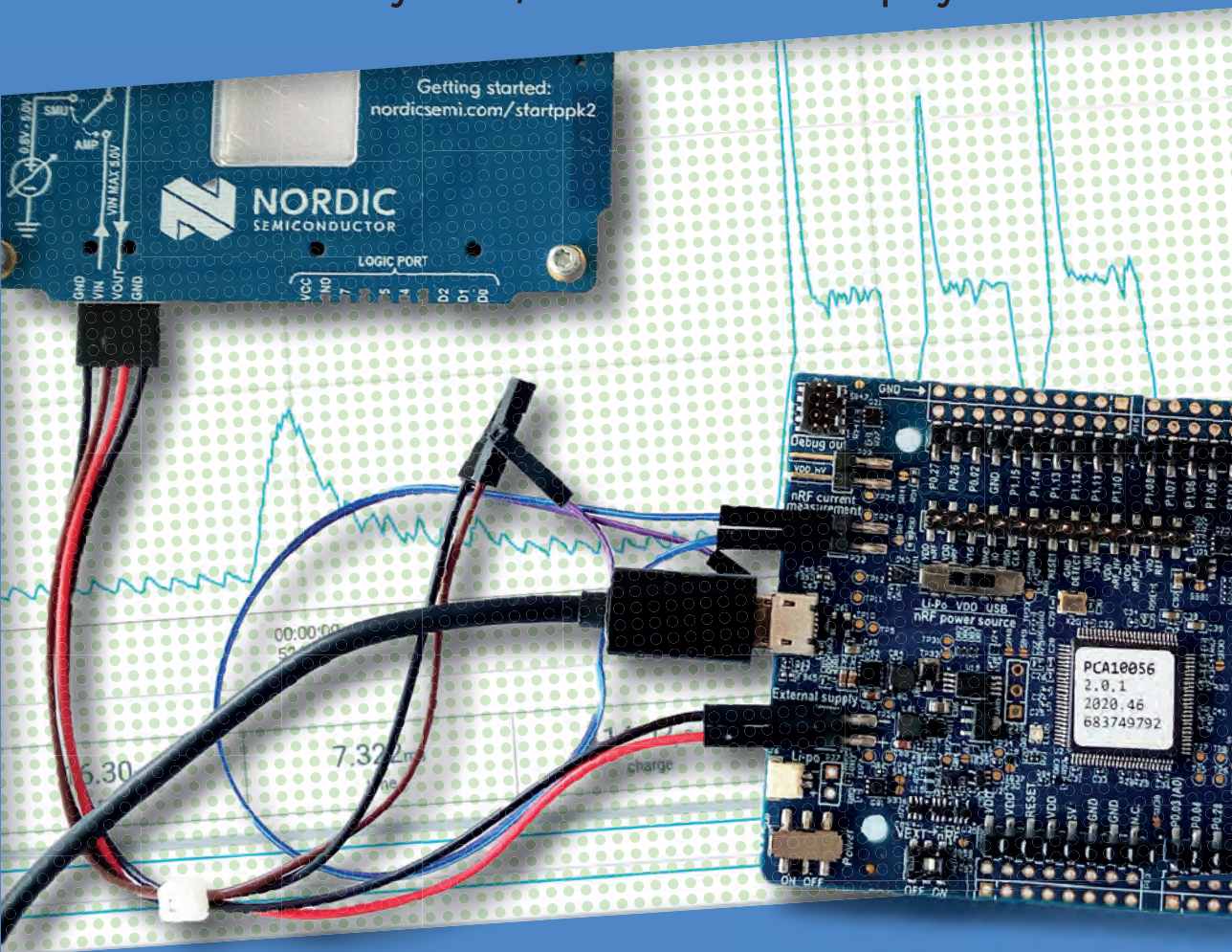




Develop your own Bluetooth Low Energy Applications

for Raspberry Pi, ESP32 and nRF52
with Python, Arduino and Zephyr



Koen Vervloesem

Develop your own **Bluetooth Low Energy Applications**

for Raspberry Pi, ESP32 and nRF52
with Python, Arduino and Zephyr



Koen Vervloesem

● This is an Elektor Publication. Elektor is the media brand of
Elektor International Media B.V.
PO Box 11, NL-6114-ZG Susteren, The Netherlands
Phone: +31 46 4389444

● All rights reserved. No part of this book may be reproduced in any material form, including photocopying, or storing in any medium by electronic means and whether or not transiently or incidentally to some other use of this publication, without the written permission of the copyright holder except in accordance with the provisions of the Copyright Designs and Patents Act 1988 or under the terms of a licence issued by the Copyright Licensing Agency Ltd., 90 Tottenham Court Road, London, England W1P 9HE. Applications for the copyright holder's permission to reproduce any part of the publication should be addressed to the publishers.

● **Declaration**

The Author and Publisher have used their best efforts in ensuring the correctness of the information contained in this book. They do not assume, and hereby disclaim, any liability to any party for any loss or damage caused by errors or omissions in this book, whether such errors or omissions result from negligence, accident, or any other cause.

● British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

● **ISBN 978-3-89576-500-1** Print
ISBN 978-3-89576-501-8 eBook

● © Copyright 2022: Elektor International Media B.V. (2022-05 / 1st)
Prepress Production: D-Vision, Julian van den Berg
Printed in the Netherlands by Ipskamp Printing, Enschede

Elektor is part of EIM, the world's leading source of essential technical information and electronics products for pro engineers, electronics designers, and the companies seeking to engage them. Each day, our international team develops and delivers high-quality content - via a variety of media channels (including magazines, video, digital media, and social media) in several languages - relating to electronics design and DIY electronics. www.elektormagazine.com

Preface	11
Chapter 1 • Introduction	12
1.1 What is Bluetooth Low Energy?	12
1.2 Layered architecture	13
1.3 How to communicate with BLE devices	15
1.3.1 Without a connection	15
1.3.2 With a connection	15
1.4 Advantages of BLE	16
1.4.1 Low power consumption	16
1.4.2 Ubiquitous	16
1.4.3 Low cost	16
1.5 Disadvantages of BLE	17
1.5.1 Short range	17
1.5.2 Limited speed	17
1.5.3 You need a gateway	17
1.6 Platforms used in this book	17
1.6.1 Python/Bleak (Raspberry Pi, PC)	18
1.6.2 C++/NimBLE-Arduino (ESP32)	18
1.6.3 C/Zephyr (nRF52)	19
1.7 How to use this book	20
1.8 Summary and further exploration	23
Chapter 2 • Preparing your development environment	24
2.1 Python and Bleak on your PC or Raspberry Pi	24
2.2 The Arduino platform with NimBLE-Arduino for the ESP32	25
2.2.1 Install Arduino CLI	26
2.2.2 Install the ESP32 Arduino core	27
2.2.3 Detect your ESP32 board	28
2.2.4 Install the NimBLE-Arduino library	29
2.3 The Zephyr development environment for nRF5 devices	30
2.3.1 Build a Zephyr application	30
2.3.2 Flash a Zephyr application	31

2.4 The nRF Connect for Desktop application	32
2.5 The nRF Connect mobile app.	33
2.6 The Bluetooth Low Energy app in nRF Connect for Desktop	34
2.7 Wireshark and a BLE sniffer dongle	35
2.7.1 Downloading Wireshark and the nRF Sniffer for Bluetooth LE	36
2.7.2 Installing the nRF Sniffer for Bluetooth LE firmware	36
2.7.3 Installing the nRF Sniffer capture tool	38
2.7.4 Installing the BLE profile	40
2.7.5 Testing a BLE packet capture	40
2.8 Summary and further exploration	41
Chapter 3 • Broadcasting data with advertisements	43
3.1 Device roles	43
3.2 Advertising packets	44
3.2.1 Advertising channels	44
3.2.2 Advertising packet structure.	46
3.3 Discovering advertisements with Bleak.	48
3.3.1 Scanning for devices	49
3.3.2 Detection callbacks	49
3.3.3 Active and passive scanning.	52
3.4 Public and random Bluetooth addresses	53
3.5 The iBeacon specification	55
3.6 Decoding iBeacon advertisements using Bleak	58
3.7 Discovering advertisements with NimBLE-Arduino	61
3.8 Decoding manufacturer-specific data using NimBLE-Arduino	66
3.8.1 Decoding iBeacon advertisements.	67
3.8.2 Decoding Microsoft advertising beacons.	69
3.9 Broadcasting iBeacon advertisements with Zephyr.	73
3.9.1 Advertising data structures in Zephyr	74
3.9.2 Enabling Bluetooth	76
3.9.3 Advertising.	77
3.9.4 Building and flashing the code	79

3.9.5 Investigating the advertised packets	80
3.10 Broadcasting sensor data as manufacturer-specific data with Zephyr	82
3.10.1 Hardware	82
3.10.2 Project structure	83
3.10.3 Source code	85
3.10.4 Decoding the BME280 sensor data	92
3.11 Advertise scan response data with Zephyr	94
3.12 Summary and further exploration	95
Chapter 4 • Connections and services	97
4.1 Device roles	97
4.2 Attributes	98
4.3 Services, characteristics, and descriptors	99
4.3.1 Services	100
4.3.2 Characteristics	100
4.3.3 Descriptors	101
4.4 Discovering services and characteristics with nRF Connect	101
4.5 A minimal GATT server	105
4.6 Discovering services and characteristics with Bleak	105
4.7 Reading and writing characteristics using Bleak	108
4.7.1 Reading characteristics	109
4.7.2 Reading characteristics by their handle	111
4.7.3 Writing characteristics	114
4.8 Notifications and indications	115
4.8.1 Read heart rate notifications	117
4.8.2 Read notifications from multiple devices	119
4.9 Creating a heart rate monitor with NimBLE-Arduino	124
4.10 Creating a GATT server with Zephyr	131
4.10.1 Exposing the Device Information service	131
4.10.2 Creating a BLE sensor with Zephyr	136
4.10.3 Reading the sensor characteristic	145
4.10.4 Sniffing packets in an unencrypted BLE connection	148

4.11 Receiving service data without a connection	149
4.11.1 Scanning for service data	149
4.11.2 Receiving Exposure Notification advertisements	151
4.12 Summary and further exploration	154
Chapter 5 • Securing BLE connections	155
5.1 BLE security architecture	155
5.2 Pairing and bonding	156
5.2.1 Phase 1: Exchange of pairing information	158
5.2.2 Phase 2: Pairing	159
5.2.2.1 LE Legacy Connection pairing	159
5.2.2.2 LE Secure Connection pairing	161
5.2.3 Phase 3: Bonding	163
5.3 Security modes and levels	164
5.4 Encrypting the BLE connection to a Zephyr sensor.	165
5.4.1 Implementing Security Mode 1 Level 2	166
5.4.2 Securely connecting to your sensor board	173
5.4.3 Sniffing the pairing procedure with Wireshark	175
5.5 Authenticating a BLE connection	176
5.5.1 Implementing Secure Connections Only Mode	177
5.5.2 Securely connecting with the board.	186
5.5.3 Sniffing the pairing procedure with Wireshark	188
5.6 Privacy	189
5.7 Summary and further exploration	190
Chapter 6 • Profiles and roles	192
6.1 Common BLE profiles	192
6.1.1 Generic profiles	192
6.1.2 GATT profiles	192
6.2 Understanding a profile specification	193
6.2.1 Introduction	195
6.2.2 Configuration	195
6.2.3 Proximity Reporter Requirements	196

6.2.4 Proximity Monitor Requirements	197
6.2.5 Connection Establishment	198
6.2.6 Security Considerations	199
6.2.7 GATT Interoperability Requirements	199
6.2.8 Acronyms and Abbreviations	200
6.2.9 References	200
6.3 Understanding a service specification	200
6.3.1 Introduction	200
6.3.2 Service Declaration	201
6.3.3 Service Characteristics	201
6.3.4 Service Behaviors	202
6.3.5 Acronyms and Abbreviations	203
6.3.6 References	203
6.4 Understanding the definition of a characteristic	203
6.4.1 Description	203
6.4.2 Definition	204
6.5 Implementing a Proximity Reporter in Zephyr	204
6.6 Implementing a Proximity Monitor in NimBLE-Arduino	210
6.7 Summary and further exploration	216
Chapter 7 • Reverse engineering BLE devices	217
7.1 Investigating the LED badge	217
7.2 Decompiling the mobile app	219
7.3 Sniffing BLE traffic between the LED badge and the mobile app	222
7.4 Writing arbitrary images to the LED badge using Bleak	225
7.4.1 Finding LED badges	225
7.4.2 Writing images to the LED badge	226
7.5 Summary and further exploration	231
Chapter 8 • Lowering power consumption	232
8.1 Measuring power consumption with the Nordic Semiconductor Power Profiler Kit II	232
8.1.1 Ampere Meter mode	233
8.1.2 Source Meter mode	234

8.2 Measuring an iBeacon's power consumption	236
8.3 Lowering power consumption by disabling hardware	237
8.4 Lowering the power consumption by using a larger advertising interval	238
8.5 Estimating battery life	239
8.6 Summary and further exploration	240
Chapter 9 • Conclusion.	242
9.1 Other BLE development platforms	242
9.2 More about Bluetooth Low Energy	243
9.3 Some ideas for further exploration.	244
Chapter 10 • Appendix.	246
10.1 Where to find BLE specifications	246
10.2 16-bit UUID ranges	247
10.3 Verifying a product's Bluetooth qualifications	247
10.4 Establishing a serial connection to a device over USB.	248
10.4.1 Check the port	248
10.4.2 Install the USB-to-serial driver	248
10.4.3 Give the user access	249
10.4.4 Start the serial connection	249
10.5 Sniffing BLE traffic on your Android device using the Bluetooth HCI snoop log. . .	250
10.5.1 Investigating the Bluetooth HCI snoop log file with Wireshark	250
10.5.2 Sniffing live BLE traffic in Wireshark with the Android Debug Bridge	251
10.6 Tips for specific hardware	251
10.6.1 Programming boards that have the Adafruit nRF52 bootloader	251
10.6.2 Programming boards with Arduino BOSSA	253
Index	254