THE EXPERT'S VOICE® IN ARDUINO

Building Arduino Projects for the Internet of Things

Experiments with Real-World Applications

A guidebook for the eager-to-learn Arduino enthusiast

Adeel Javed

Apress®

Building Arduino Projects for the Internet of Things

Experiments with Real-World Applications

Adeel Javed

Apress®

Building Arduino Projects for the Internet of Things: Experiments with Real-World Applications

Adeel Javed
Lake Zurich, Illinois, USA

ISBN-13 (pbk): 978-1-4842-1939-3 ISBN-13 (electronic): 978-1-4842-1940-9
DOI 10.1007/978-1-4842-1940-9

Library of Congress Control Number: 2016943433

Copyright © 2016 by Adeel Javed

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director: Welmoed Spahr
Lead Editor: Jonathan Gennick
Development Editor: James Markham
Technical Reviewer: Jeff Tang
Editorial Board: Steve Anglin, Pramila Balen, Louise Corrigan, James DeWolf,
Jonathan Gennick, Robert Hutchinson, Celestin Suresh John, Nikhil Karkal,
James Markham, Susan McDermott, Matthew Moodie, Douglas Pundick,
Ben Renow-Clarke, Gwenan Spearing
Coordinating Editor: Melissa Maldonado
Copy Editor: Kezia Endsley
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springer.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/.

Printed on acid-free paper

To my wife Naila, for supporting me throughout the process.

Contents at a Glance

Contents

About the Author

Adeel Javed is a Solutions Architect with over 11 years of software development, design, and systems-architect experience in enterprise-wide business process management (BPM) and service-oriented architecture (SOA) solutions. He helps organizations from diverse global-industry domains with process improvements and implementation initiatives. Adeel Javed regularly writes about BPM, SOA, IoT, cloud, and all things process-oriented on his blog, `ProcessRamblings.com`, as well as for other major industry sites such as `BPMLeader.com`, `BPTrends.com`, and IBM developerWorks.

In his time off, Adeel is an avid—and process-driven—Arduino enthusiast and device developer.

About the Technical Reviewer

Jeff Tang worked on enterprise and web app development for many years before reinventing himself to focus on building great iOS and Android apps. He had Apple-featured, top-selling iOS apps with millions of users and was recognized by Google as a Top Android Market Developer. He's the author of the *Beginning Google Glass Development* book published by Apress in 2014. His current passion is in IoT and AI and he actually received his master's degree in AI.

Preface

Analysts are forecasting that by the year 2020 there will be more than 50 billion connected things (devices) and the total revenue from the Internet of things (IoT) will easily surpass $1.5 trillion.

The numbers look phenomenal, but what exactly is IoT? Is it simply things connected to the Internet? Why do connected things matter?

IoT is much more than things connected to the Internet. IoT is about making dumb things smarter by giving them the ability to sense, communicate, and respond. We have five senses—we can see, hear, taste, smell, and touch. Similarly if you add these sensors to things they can do the same as well. For example, using a camera things can see, using a sound detector things can hear, and using a speaker things can talk. There are so many other sensors that things can use to do so much more than us. By connecting these things to the Internet, they can communicate with us, with other things, and the next frontier where they can use artificial intelligence to think as well. There are numerous applications of IoT, but here are a couple of examples to further understand how IoT is being used to improve our lives:

- A wristband with the ability to monitor your vitals. If it finds anything out of the ordinary, it can alert you and your doctor immediately.
- A security system that monitors the premises of your house for any intrusions and alerts you and any security agencies.

What This Book Covers

This book is based on my personal experience of getting started with IoT. It is divided into two logical sections. The first one teaches the basics of building IoT applications and the second section follows a project-based approach. At the end of each chapter you will have a working prototype of an IoT application.

Part 1: Building Blocks

Chapters 1-3 cover the building blocks of IoT:

- Chapter 1, “Arduino Basics,” introduces the Arduino prototyping platform, which is used throughout the book.
- Chapter 2, “Internet Connectivity,” discusses the different options available for connecting things to the Internet.
- Chapter 3, “Communication Protocols,” teaches you what communication protocols are and which ones are available for IoT.

Part 2: Prototypes

Chapters 4-12 use the information covered in Part 1 to build prototypes of IoT applications.

- Chapter 4, “Complex Flows: Node-RED,” introduces Node-RED, which is a visual designer that helps reduce the amount of code required for IoT applications.
- Chapter 5, “IoT Patterns: Realtime Clients,” talks about components required for building IoT applications that provide data to users in real time and shows you how to build an intrusion detection system as an example.
- Chapter 6, “IoT Patterns: Remote Control,” discusses components of IoT applications that can remotely control things, such as a lighting control system.
- Chapter 7, “IoT Patterns: On-Demand Clients,” shows you different components involved in building an on-demand IoT application. You’ll build a smarter parking system in this chapter.
- Chapter 8, “IoT Patterns: Web Apps,” teaches you scenarios where web clients are preferred and uses a temperature monitoring system as an example.
- Chapter 9, “IoT Patterns: Location-Aware Devices,” discusses importance of location-aware devices. You’ll develop a livestock tracking system as an example.
- Chapter 10, “IoT Patterns: Machine to Human,” talks about scenarios where human response is needed; you’ll build a waste management system as an example.
- Chapter 11, “IoT Patterns: Machine to Machine,” discusses a pattern of IoT that is going to be very popular as things get smarter. The example is an energy conservation system.
- Chapter 12, “IoT Platforms,” wraps up the book by introducing you to IoT platforms that help expedite entry into IoT. The example in this chapter builds a soil moisture control system.

What You Need for This Book

IoT applications require hardware and software and can span different technologies, so this book uses quite a few technologies. However, we have tried to keep them as simple and minimal as possible.

Required Hardware

Read the complete instructions provided in each chapter because, based on your device, you may or may not need additional components.

- Arduino Uno or Arduino Yún
- Ethernet shield
- WiFi (wireless) shield
- Breadboard
- Jumper cables (male-male, male-female)
- Light sensor
- Motion sensor (HC-SR501)
- LED
- 220Ω resistor
- Proximity sensor (Ultrasonic Rangemeter HC-SR04)
- Temperature sensor (TMP36)
- GPS module (NEO6MV2)
- Soil moisture sensor

Software

- Arduino IDE
- Node-RED
- MQTT broker (book uses free and publicly available broker from Eclipse Foundation)
- Android Studio
- Xcode/Swift
- PHP server
- MySQL server
- Text editor
- Effektif BPM (cloud-based, free account required)
- Xively (cloud-based, free account required)
- Zapier (cloud-based, free account required)

To further help you, we have also created a web site at `http://codifythings.com` dedicated to the book. The web site contains variations and enhancements to prototypes developed in this book along with additional prototypes.

Who This Book Is For

This book is for hobbyists and professionals who want to enter the world of IoT.

The material in this book requires some prior knowledge of Arduino or similar devices and programming experience. We have used basic hardware components and provided step-by-step instructions for building circuits. We kept the code simple, readable, and minimal to help newbies understand concepts and develop useable prototypes. Throughout the book, the code is consistent and, wherever needed, is explained in detail.

PART 1

Building Blocks

CHAPTER 1

Arduino Basics

Arduino is an open-source platform that's composed of very simple and easy-to-use hardware and software. In a nutshell your Arduino can read sensor data and control components such as lights, motors, thermostats, and garage doors. It has mainly been developed for prototyping purposes, so it is a great fit for this IoT beginner's book.

Learning Objectives

At the end of this chapter, you will be able to:

- Use Arduino hardware
- Use the Arduino IDE
- Write, upload, and execute basic Arduino programs

Hardware Requirements

Arduino comes in various models (also known as *boards*). Each board has different specifications. If your board does not come built-in with the features you are looking for, then you always have an option to add a shield that supports required features. In the Arduino world, a shield is very similar to a board, but it only supports specific functionality such as the ability to connect to a WiFi network or the ability to control servo motors. A shield acts as an add-on; that is, it is physically attached to the top of an Arduino board. Once attached, the Arduino board becomes capable of handling shield features as well.

Figure 1-1 shows a diagram of Arduino Uno, while Figure 1-2 shows a diagram of an Ethernet shield.

Electronic supplementary material The online version of this chapter (doi:10.1007/978-1-4842-1940-9_1) contains supplementary material, which is available to authorized users.

© Adeel Javed 2016

A. Javed, *Building Arduino Projects for the Internet of Things*,
DOI 10.1007/978-1-4842-1940-9_1

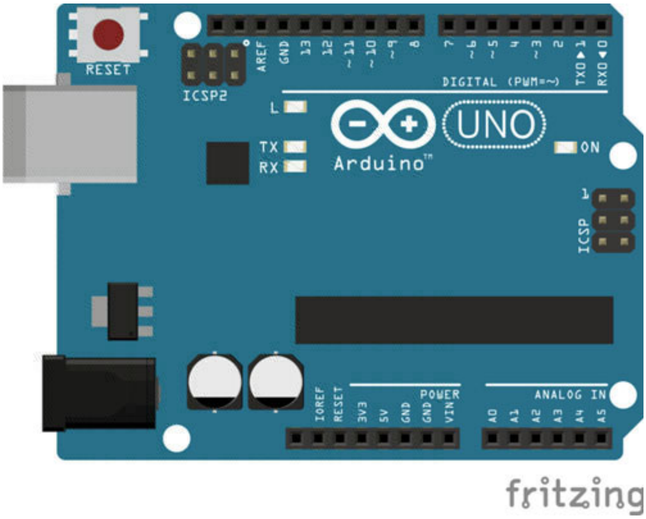


Figure 1-1. *Arduino Uno*

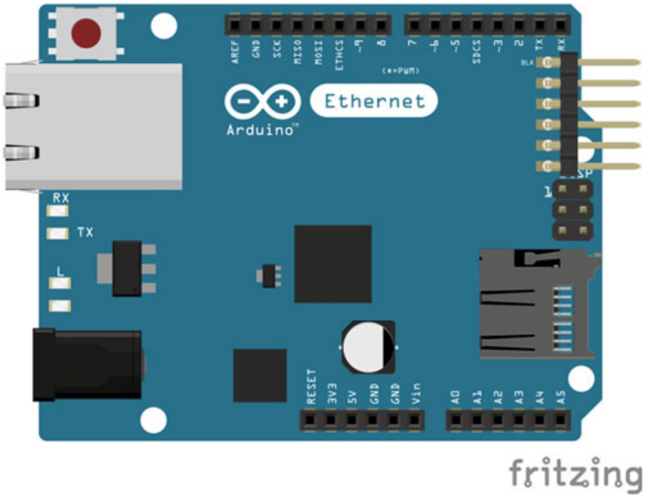


Figure 1-2. *Ethernet shield*

The following list summarizes some of the important parts of the board that have been used in projects throughout the book.

■ **Note** Parts will vary based on the Arduino board you choose.

- **Digital pins:** In total there are 14 digital pins on Arduino Uno. Digital pins can be both INPUT and OUTPUT, but their state can only be HIGH or LOW. HIGH means there is current while LOW means no current. An example of digital pin usage is turning an LED light on or off. To turn it on, the digital pin should be set to HIGH and to turn it off the digital pin should be set to LOW.
- **Analog pins:** Arduino Uno supports six analog pins, A0 through A5. Unlike digital pins, the readings of analog pins can range from 0 to 1023. A good example of a sensor that provides analog readings is a soil moisture sensor. The range helps identify how much moisture is left in the soil.
- **USB connector:** A USB connector lets you connect Arduino to the computer, power the board, upload code, and receive logs on a serial monitor.
- **Battery power:** IoT applications that need to be placed in remote locations will need their own power source. You can use the battery power connector to power the board.

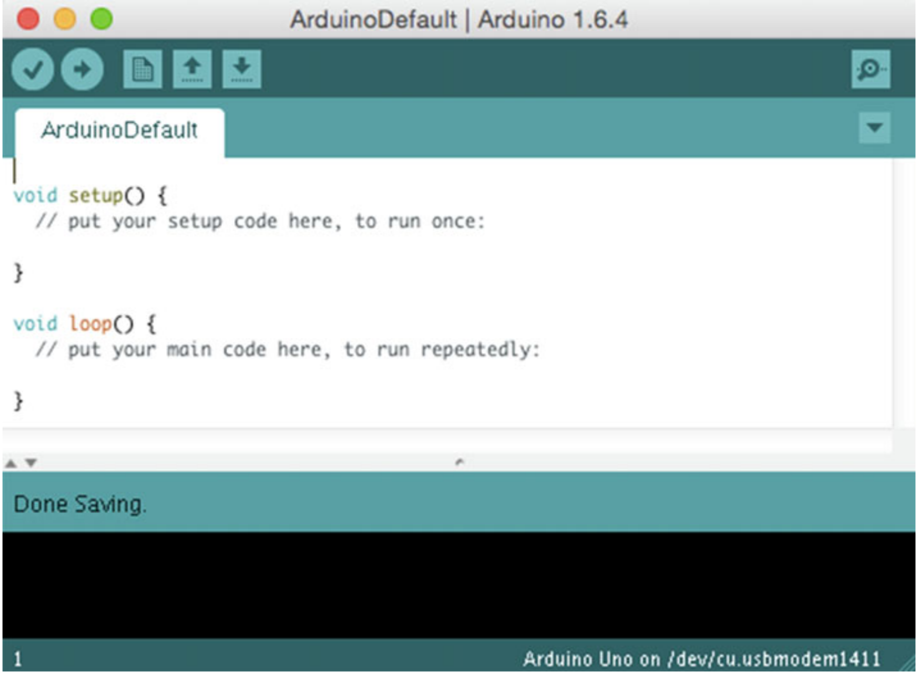
This book uses Arduino Uno for all projects. Arduino Uno is categorized as an entry-level board most suited for beginners. Even though the book uses Arduino Uno, you are not required to use it; you can choose any of the Arduino boards to complete projects in this book. Since this book is about the Internet of things, Internet connectivity is an important requirement. Whichever Arduino board you decide to use, just make sure that it supports Internet connectivity in some form. The Arduino board should either come with a built-in Internet connectivity option or you should have the required Internet connectivity shield.

■ **Note** Arduino Uno does not come with built-in Internet connectivity support, so in the book both Ethernet and WiFi shields have been used. On the other hand, a more advanced model of Arduino called *Yún* does support built-in Ethernet and WiFi connectivity. Chapter 2 discusses Internet connectivity in more detail.

Software Requirements

Arduino provides a C-like language for programming Arduino boards. You will be using the Arduino IDE for writing code and uploading it to an Arduino board. You can install the latest version of Arduino IDE from `https://www.arduino.cc/en/Main/Software`.

Once Arduino IDE has been installed on your machine, open it and, as shown in Figure 1-3, it will load with default code.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Done Saving.

1 Arduino Uno on /dev/cu.usbmodem1411

Figure 1-3. *Default view of Arduino IDE*

There are three components of Arduino IDE that are referenced in every chapter of this book.

- Toolbar
- Status window
- Serial Monitor window

Toolbar

The toolbar on top of the IDE, as shown in Figure 1-4, provides easy access to frequently used options.

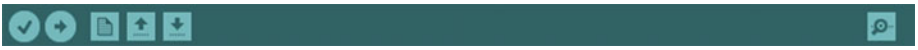
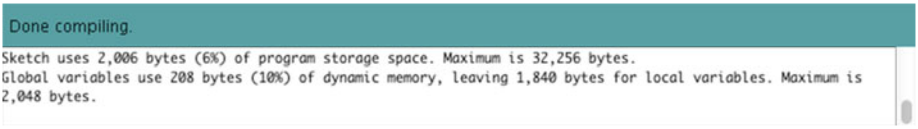


Figure 1-4. *Arduino IDE toolbar*

- **Verify/Compile:** This is the first button from the left (the tick mark). Click this button to verify and compile your code for correctness. You can view the results in the Status window at the bottom.
- **Upload:** This is the second button from left (right-pointing arrow). If your Arduino board is connected to your machine that is running the Arduino IDE, this will upload the code on the Arduino board. You can view the deployment results in the Status window at the bottom.
- **New/Open/Save:** The next three buttons, as their names suggest, let you open a new code window, open an existing code file, or save the currently open code. Arduino code files have an `*.ino` extension.
- **Serial/Monitor:** The last button on the right lets you open the Serial Monitor window.

Status Window

When you verify the code or upload it to a board, the Status window shown in Figure 1-5 lists all the results. Any errors that occur during code verification or uploading will be shown in the Status window.



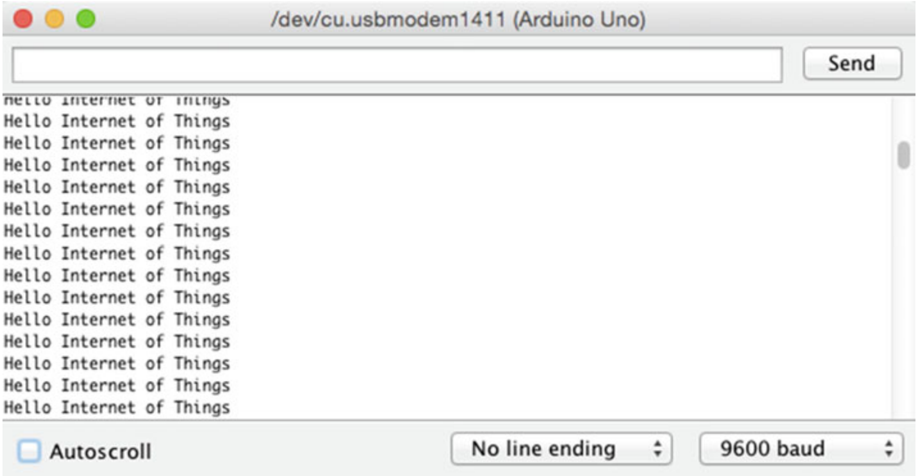
Done compiling.

```
Sketch uses 2,006 bytes (6%) of program storage space. Maximum is 32,256 bytes.
Global variables use 208 bytes (10%) of dynamic memory, leaving 1,840 bytes for local variables. Maximum is
2,048 bytes.
```

Figure 1-5. *Arduino IDE Status window*

Serial Monitor Window

The Serial Monitor window shown in Figure 1-6 prints all log messages generated by the `Serial.print()` and `Serial.println()` functions in the code. In order to print any messages on the Serial Monitor window, you first need to initialize the message in the code (discussed later).



```
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
```

No line ending

9600 baud

Figure 1-6. *Log messages on the Serial Monitor window*

Arduino Programming Language Reference

The Arduino programming language has quite a few constructs. However, this chapter provides the basics that have been used throughout the projects in this book; see Table 1-1.

Table 1-1. *Language Reference*

Code Construct	Description
`int`	Integer values, such as 123
`float`	Decimal values, such as 1.15
`char[]`	String values, such as "`Arduino`"
`HIGH`	Digital pin with current
`LOW`	Digital pin with no current
`INPUT`	Pin can only be read
`OUTPUT`	Pin can only be set
`A0 - A7`	Constants for analog pins; varies by board
`0 - 13`	Value for digital pins; varies by board
`analogRead()`	Returns analog pin value (0 – 1023)
`analogWrite(...)`	Sets analog pin value
`digitalRead()`	Returns digital pin value (HIGH or LOW)

(*continued*)