Chapter 1: Getting Started on the Raspberry Pi
Raspbian [RECOMMENDED]
A Debian wheezy port, optimised for the Raspberry Pi

OpenELEC [INSTALLED]
OpenELEC is a fast and userfriendly XBMC Mediacenter distribution.

RISC OS [INSTALLED]
RISC OS is a very fast and compact system

Raspbian - Boot to Desktop
A version of Raspbian that boots straight to a GUI

Arch
An Arch Linux port for ARM devices

RaspBMC
An XBMC media center distribution for Raspberry Pi

Disk space

Needed: 4161 MB
Available: 6881 MB
eth0  Link  encap:Ethernet  HWaddr  b8:27:eb:f6:fc:89
  inet6  addr:  fe80::734f:7460:ddaf:cc40/64  Scope:Link
  UP  BROADCAST  MULTICAST  MTU:1500  Metric:1
  RX  packets:0  errors:0  dropped:0  overruns:0  frame:0
  TX  packets:0  errors:0  dropped:0  overruns:0  carrier:0
  collisions:0  txqueuelen:1000
  RX  bytes:0  (0.0  B)  TX  bytes:0  (0.0  B)

lo     Link  encap:Local  Loopback
  inet  addr:127.0.0.1  Mask:255.0.0.0
  inet6  addr:  ::1/128  Scope:Host
  UP  LOOPBACK  RUNNING  MTU:65536  Metric:1
  RX  packets:209  errors:0  dropped:0  overruns:0  frame:0
  TX  packets:209  errors:0  dropped:0  overruns:0  carrier:0
  collisions:0  txqueuelen:1
  RX  bytes:17180  (16.7  KiB)  TX  bytes:17180  (16.7  KiB)

wlan0  Link  encap:Ethernet  HWaddr  b8:27:eb:a3:a9:dc
  inet  addr:192.168.0.10  Bcast:192.168.0.255  Mask:255.255.255.0
  inet6  addr:  fe80::f60:34f:49b8:5252/64  Scope:Link
  UP  BROADCAST  RUNNING  MULTICAST  MTU:1500  Metric:1
  RX  packets:18681  errors:0  dropped:14902  overruns:0  frame:0
  TX  packets:4620  errors:0  dropped:0  overruns:0  carrier:0
  collisions:0  txqueuelen:1000
  RX  bytes:3576248  (3.4  MiB)  TX  bytes:4214622  (4.0  MiB)
Would you like the SSH server to be enabled?

<Yes>  <No>
Chapter 2: Getting Up-and-Running with Web Development on the Raspberry Pi
Outside applications can only communicate with the sensor application through the server application.

Outside applications can communicate with the sensor application independently of the server application through its own APIs.
SELECT * FROM temperatures

<table>
<thead>
<tr>
<th>id</th>
<th>date</th>
<th>°C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01-03-2017</td>
<td>23</td>
</tr>
<tr>
<td>2</td>
<td>02-03-2017</td>
<td>21.5</td>
</tr>
<tr>
<td>3</td>
<td>03-03-2017</td>
<td>22</td>
</tr>
</tbody>
</table>

DELETE FROM temperatures WHERE id=1

<table>
<thead>
<tr>
<th>id</th>
<th>date</th>
<th>°C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01-03-2017</td>
<td>23</td>
</tr>
</tbody>
</table>

INSERT INTO temperatures VALUES (4, 04-03-2017, 24)

<table>
<thead>
<tr>
<th>id</th>
<th>date</th>
<th>°C</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>04-03-2017</td>
<td>24</td>
</tr>
</tbody>
</table>

UPDATE temperatures SET date=05-03-2017 WHERE id=4

<table>
<thead>
<tr>
<th>id</th>
<th>date</th>
<th>°C</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>05-03-2017</td>
<td>24</td>
</tr>
</tbody>
</table>
Chapter 3: Running a Node Server on the Pi

```
pi@raspberrypi:~/sensor-project/server $ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help json' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg> --save' afterwards to install a package and
save it as a dependency in the package.json file.

Press AC at any time to quit.
name: (server)
version: (1.0.0)
description: The server application for this project
entry point: (index.js)
test command: git repository:
keywords:
author:
license: (ISC)
About to write to /home/pi/sensor-project/server/package.json:

{
  "name": "server",
  "version": "1.0.0",
  "description": "The server application for this project",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" & & exit 1"
  },
  "author": ",",
  "license": "ISC"
}
```

Is this ok? (yes)

```
pi@raspberrypi:~/sensor-project/server $ node server
Server listening on port 3000
```

```
192.168.0.10:3000/temperature

24 °C
```
pm@raspberrypi:~ $ pm2 --version

--------------

PM2 process manager

Getting started

Documentation
http://pm2.io/

Start PM2 at boot
$ pm2 startup

Daemonize Application
$ pm2 start <app>

Monitoring/APM solution
https://app.keymetrics.io/

--------------

[PM2] Spawning PM2 daemon with pm2_home=/home/pi/.pm2
[PM2] PM2 Successfully daemonized
v2.4.5
```bash
pi@raspberrypi:~/sensor-project $ pm2 start server/
[PM2] Starting /home/pi/sensor-project/server in fork_mode (1 instance)
[PM2] Done.

<table>
<thead>
<tr>
<th>App name</th>
<th>id</th>
<th>mode</th>
<th>pid</th>
<th>status</th>
<th>restart</th>
<th>uptime</th>
<th>cpu</th>
<th>mem</th>
<th>watching</th>
</tr>
</thead>
<tbody>
<tr>
<td>server</td>
<td>0</td>
<td>fork</td>
<td>1944</td>
<td>online</td>
<td>0</td>
<td>0s</td>
<td>35%</td>
<td>16.1 MB</td>
<td>disabled</td>
</tr>
</tbody>
</table>
```

Use `pm2 show <id|name>` to get more details about an app.
Chapter 4: Extracting Information from the GPIO Pins
pi@raspberrypi:~/sensor-project/server $ node obtain-reading.js
temp: 25.0°C, humidity: 70.0%
pi@raspberrypi:~/sensor-project/server $
Chapter 5: Retrieving Sensor Readings from the Server
32%
Chapter 6: Creating a Web Page to Display Sensor Data
Temperature:
10 °C
Humidity:
43 %
<table>
<thead>
<tr>
<th>Sensor Dashboard</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Temperature</strong></td>
</tr>
<tr>
<td>10.0 °C</td>
</tr>
<tr>
<td><strong>Humidity</strong></td>
</tr>
<tr>
<td>20.0 °C</td>
</tr>
</tbody>
</table>
Chapter 7: Enhancing Our UI - Using Interactive Charts
Temperature
20.0 °C

Humidity
86.0 °C
# Chapter 8: SQLite - The Fast and Portable Database

<table>
<thead>
<tr>
<th>createdAt</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-06-18 12:26:08</td>
<td>16.9</td>
</tr>
<tr>
<td>2017-06-18 12:26:09</td>
<td>14.7</td>
</tr>
<tr>
<td>2017-06-18 12:26:10</td>
<td>22.0</td>
</tr>
<tr>
<td>2017-06-18 12:26:13</td>
<td>21.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>createdAt</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-06-18 12:26:08</td>
<td>16.6</td>
</tr>
<tr>
<td>2017-06-18 12:26:09</td>
<td>14.7</td>
</tr>
<tr>
<td>2017-06-18 12:26:10</td>
<td>22.0</td>
</tr>
<tr>
<td>2017-06-18 12:26:13</td>
<td>21.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>createdAt</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-06-18 12:26:10</td>
<td>22.0</td>
</tr>
<tr>
<td>2017-06-18 12:26:13</td>
<td>21.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>value</th>
<th>deviation</th>
<th>createdAt</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.6</td>
<td>2.0</td>
<td>2017-06-18 12:26:08</td>
</tr>
<tr>
<td>14.7</td>
<td>3.9</td>
<td>2017-06-18 12:26:09</td>
</tr>
<tr>
<td>22.0</td>
<td>-3.4</td>
<td>2017-06-18 12:26:10</td>
</tr>
</tbody>
</table>
Chapter 9: Integrating SQLite into Our Application
Sensor Dashboard

Temperature

19.9
°C

Humidity

49.6
%
Chapter 10: Making our Application Real Time with Web Sockets

Diagram showing the process of establishing a web socket connection, including headers with 'Connection: Upgrade' and 'Upgrade: Web socket', and the server sending a status code of 101.
io

function r()
Network activity with HTTP REST implementation

Network activity with web socket implementation

Sensor data now sent over socket connection

and counting...
Chapter 11: Deploying our application to Firebase
Welcome to Firebase

Tools from Google for developing great apps, engaging with your users, and earning more through mobile ads.

Learn more  Documentation  Support

Recent projects

Add project

Explore a demo project

sensor-project

sensor-project-5df04

Welcome to Firebase! Get started here.

Add Firebase to your iOS app  Add Firebase to your Android app  Add Firebase to your web app

Discover Firebase
Woohoo!
Firebase CLI Login Successful

You are logged in to the Firebase Command-Line interface. You can immediately close this window and continue using the CLI.

```bash
firebase git:(master) firebase init
```

You're about to initialize a Firebase project in this directory:
```
/home/minh/projects/sensor-project/Firebase
```

What Firebase CLI features do you want to setup for this folder?
- Database: Deploy Firebase Realtime Database Rules
- Functions: Configure and deploy Cloud Functions
- Hosting: Configure and deploy Firebase Hosting sites
--- Project Setup

First, let's associate this project directory with a Firebase project. You can create multiple project aliases by running `firebase use --add`, but for now we'll just set up a default project.

? What Firebase project do you want to associate as default? sensor-project (sensor-project-Sdf04)

--- Database Setup

Firebase Realtime Database Rules allow you to define how your data should be structured and when your data can be read from and written to.

? What file should be used for Database Rules? database.rules.json
✓ Database Rules for sensor-project-Sdf04 have been downloaded to database.rules.json. Future modifications to database.rules.json will update Database Rules when you run `firebase deploy`.

--- Hosting Setup

Your public directory is the folder (relative to your project directory) that will contain Hosting assets to be uploaded with `firebase deploy`. If you have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? public
✓ Configure as a single-page app (rewrite all urls to /index.html)? No
✓ Wrote public/404.html
✓ Wrote public/index.html

i Writing configuration info to firebase.json...
i Writing project information to .firebaserc...
✓ Firebase initialization complete!
✓ `firebase git:(master)`

Welcome to Firebase Hosting
You're seeing this because you've successfully set up Firebase Hosting. Now it's time to go build something extraordinary!

OPEN HOSTING DOCUMENTATION
Sensor Dashboard

**Temperature**

*Loading...*

°C

**Humidity**

*Loading...*

%
Add Firebase to your web app

Copy and paste the snippet below at the bottom of your HTML, before other script tags.

```html
<script src="https://www.gstatic.com/firebasejs/4.1.3/firebase.js"></script>

// Initialize Firebase
var config = {
  apiKey: "AIzaSyAKlNfMA1o9xMxW8T_6vc5Vv65",
  authDomain: "sensor-project-5df04.firebaseapp.com",
  databaseURL: "https://sensor-project-5df04.firebaseapp.com",
  projectId: "sensor-project-5df04",
  storageBucket: "sensor-project-5df04.appspot.com",
  messagingSenderId: "184513757436"
};

firebase.initializeApp(config);
</script>

Check these resources to learn more about Firebase for web apps:
- Get Started with Firebase for Web Apps (Opens in a new window)
- Firebase Web SDK API Reference (Opens in a new window)
- Firebase Web Samples (Opens in a new window)
Chapter 12: Using Firebase APIs to Update Our Application
Firebase Admin SDK

Your Firebase service account can be used to authenticate multiple Firebase features, such as Database, Storage and Auth, programmatically via the unified Admin SDK. Learn more

Firebase service account
firebase-adminsdk-bsu70@sensor-project-5df04.iam.gserviceaccount.com

Admin SDK configuration snippet

```javascript
var admin = require("firebase-admin");
var serviceAccount = require("path/to/serviceAccountKey.json");
admin.initializeApp(
  {
    credential: admin.credential.cert(serviceAccount),
    databaseURL: "https://sensor-project-5df04.firebaseio.com"
  });
```

[GENERATE NEW PRIVATE KEY]

Last key downloaded: Jul 25, 2017, 9:42:01 PM

---

Generate new private key

⚠️ Your private key gives access to your project's Firebase services. Keep it confidential and never store it in a public repository.

Store this file securely, because your new key can't be recovered if lost

[CANCEL] GENERATE KEY