This book starts by teaching you the essentials of the Intel Galileo board, its components, how to wire it, and how to use it safely. The book will teach you how to use and combine simple sensors to build more complex connected objects with the help of an Internet connection. You’ll also learn how to control and read from your sensors by building a number of interesting projects. Finally, the book will familiarize you with the art of controlling your objects using mobile devices.

By the end of the book, you’ll be able to understand the key concepts of the Internet of Things, and what a "Thing" truly is. This book will make you ready and also more aware of what you can do with a Galileo board, while inspiring you with more ideas to build your own home projects.

Who this book is written for
This book employs an incremental, step-by-step approach to get you familiarized with everything from the basic terms, board components, and development environments to developing real projects. Each project will demonstrate how to use specific board components and tools. Both Galileo and Galileo Gen 2 are covered in this book.

What you will learn from this book
- Understand the concept of the Internet of Things
- Wire and power a Galileo board safely
- Develop code using the Arduino IDE, Intel IoT Dev Kit, and Intel XDK environments
- Build and start a Galileo board from a bootable image
- Use the Internet to feed, control, and gather data from sensors
- Create connected modules
- Combine multiple sensors to build complex projects

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Introducing Galileo'
- A synopsis of the book’s content
- More information on Internet of Things with Intel Galileo
Miguel de Sousa is a core engineer and full-stack developer at Muzzley. He holds an MSc in computing and telecommunications engineering from ISCTE-IUL Lisbon University Institute, Portugal. He has worked in backend networking, protocols implementation, and systems integration. As a maker, over the last 2 years, he has been developing several projects using Raspberry Pi, Arduino, and Intel Galileo/Edison boards in the field of Internet of Things.
As technology is evolving, things that surround us in our daily lives are starting to have the ability to share data over the Internet. With this evolution, it is no longer the case that only humans can operate the devices connected to the Internet. These devices are now able to collect and share sensorial data that can be controlled by sensor inputs. They also help you power up big data analysis, monitor systems, and even make devices work together for a common purpose. A new era has begun, the era of Internet of Things!

Following this vision, Intel presented the Galileo board, a board that packs together many of the most common components that are usually purchased separately for most development boards. An Intel Galileo Board can be programmed to read and control sensors and actuators, being an interesting tool for sensorial data collection. The possibility of connecting it straight to the Internet using an Ethernet cable or a wireless card in its mini PCI-express slot enables it to share the collected data over the Internet. Another great feature is that being hardware and software compatible with Arduino, it will make you have a very familiar development environment. If you prefer using other development tools, you can also do that by booting your board from a custom Linux image.

This book will give you the right tools to help you start developing your own IoT projects using an Intel Galileo board.

What this book covers

Chapter 1, Introducing Galileo, introduces you to the Intel Galileo boards by explaining their components, main differences when compared to other boards, and the other interesting boards for building IoT projects.

Chapter 2, Rediscovering the Arduino IDE, will guide you through the Arduino IDE from the process of setting up your board to uploading and running your first sketches.
**Preface**

*Chapter 3, Monitoring the Board Temperature,* will help you develop your first IoT project with the Arduino IDE. You'll create and collect temperature data samples from your own board CPU temperature and plot it to an online chart using Galileo as a web client.

*Chapter 4, Creating a Motion Sensing Light,* will show you how to create a web server with the Arduino IDE and use it to display a web page, allowing you to switch the lighting system operation mode, which is controlled by motion sensor, either by luminosity or manually.

*Chapter 5, Intel IoT Developer Kit Tools,* introduces you to the Intel ecosystem, giving you an overview of its perks and how you can use its main libraries to read and control your sensors and actuators in other development languages.

*Chapter 6, Building an Irrigation System,* will guide you through the process of monitoring your sensor data using Wyliodrin, and the creation of rules to control actuators.

*Chapter 7, Creating Christmas Light Effects,* will teach you how to create remotely controlled animations using an LED strip and a YouTube player.

*Chapter 8, The Intel XDK IoT Edition,* will explain how you can use the Intel XDK IoT Edition IDE to develop Node.js projects for Galileo.

*Chapter 9, Developing an IoT Quiz,* will help you develop a quiz game played with mobile devices.

*Chapter 10, Integrating with Muzzley,* will show you how you can integrate Galileo in your daily life. You'll be creating a building door unlocking system using your Galileo board and integrating it with market-available smart devices using the Muzzley cloud-based ecosystem.
The **Internet of Things (IoT)** is a hot topic nowadays. It is a vision where everyday objects are connected, and share data over the Internet. It is believed that it will have a huge impact in our lives by changing the way we interact with the things that are present in our daily lives.

In this context, many development boards have been developed for the makers' community over the last few years. Most of them required some of the essential components to be bought separately, such as the Ethernet socket. Intel offered a different solution packing the most common components together onboard and putting the Arduino and Linux worlds together, all in one board — the Galileo.

In this chapter, you will be able to understand what the concept of the Internet of Things is, what a Thing is, and how Galileo may help you develop your Things.

In this chapter, we'll be covering the following topics:

- The Internet of Things vision
- Galileo board and its components
- Galileo Gen 2
- Popular boards comparison

**The Internet of Things vision**

The Internet of Things concept is neither entirely new nor is a futuristic distant technology. It is being built today with today's technology, and you can find it in some of your own home devices, big data clouds, and sensors. It started with wireless technologies converging progressively with **micro-electromechanical systems (MEMS)** and the Internet.
The initial concept suggested that it were the persons who should share the data. Today, it can be defined as a network of sensing and actuating devices with the ability to share information.

The first time the term "Internet of Things" was officially used in a publication back in 1999, where Kevin Ashton published his vision in the RFID Journal:

You can't eat bits, burn them to stay warm or put them in your gas tank. Ideas and information are important, but things matter much more. Yet today's information technology is so dependent on data originated by people that our computers know more about ideas than things. If we had computers that knew everything there was to know about things — using data they gathered without any help from us — we would be able to track and count everything, and greatly reduce waste, loss and cost. We would know when things needed replacing, repairing or recalling, and whether they were fresh or past their best. The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so.

As you can see, devices sharing data is the real concept behind IoT. Such devices could either be living or inanimate. A Thing in the IoT context can be a person wearing a pulse monitor, a dog carrying a tracking device, a garbage bin that notifies it needs to be emptied, or a thermostat that adjusts itself automatically to help you lower your electricity bills.

IoT assumes that Things must be uniquely identifiable and able to gather data recurring to sensors. They must also have the ability to communicate and transfer data over a network. Such data could be used for monitoring purposes, big data processing, or even to control that same Thing.

Things supporting this machine-to-machine communication are usually known as smart devices. An example of a smart device is the famous Google Nest thermostat (https://nest.com/thermostat). Being more than a simple thermostat, it shares its usage data to help you save on your home's energy, while keeping you cozy. It can also work together with other smart devices such as some Mercedes-Benz cars (https://nest.com/works-with-nest/). The car GPS system shares data with the Nest cloud, making it possible to start heating or cooling your home, based on the expected arrival time.
Introducing Intel Galileo

Intel® Galileo is a development board based on Intel x86 architecture; it was designed mostly for makers and complies with open source software and hardware licenses. If you are familiar with the Arduino boards, you’ll find this board somewhat similar; the reason being this board was designed to be hardware and software compatible with the Arduino shield ecosystem. It combines Intel technology with support for Arduino shields and libraries. It is even possible to write code using the same Arduino development environment.

The expansion header is similar to the Arduino ones. It has 14 digital I/O pins (where six of them can be used as PWM), six analog inputs, a serial port, and an ICSP header. It supports shields operating at either 3.3 V or 5 V. A jumper on the board enables voltage translation at the I/O pins from 5 V to 3.3 V, and vice versa, providing compatibility with the Arduino shields.

Galileo runs over a very light open source Linux OS in its 8 MB flash memory. However, do take into consideration that Arduino is being emulated using Linux, and your code will be running in a separate process.

This board includes a 10/100 Ethernet connector port, and if you wish to use Wi-Fi, you can add a card to the Mini PCIe socket on the back side of the board.
Introducing Galileo

Breaking down the board, you can find the following major components:

1. **I2C jumper**: This jumper allows you to change the I2C address of some on-board components. You may need to do this if you are using I2C components that conflict with other components on the board.

2. **Ethernet port**: This port allows you to connect your board to a wired network, allowing you to communicate with other devices and also access the Internet. If you wish to use Wi-Fi, in the backside of the board, you'll find a Mini PCI Express slot where you can connect your Wi-Fi card. It also enables another possible storage device, USB host, bluetooth, or GSM card.

3. **Serial port**: There is a serial port for connecting to the Galileo Linux command line from your computer. Although this port looks like an audio jack, it is only used for serial communication.

4. **USB client**: When developing with the Arduino IDE, you'll need to connect your USB cable here, so that you can upload your project's code on the board.

5. **USB host**: Do not mistake this port for the USB client. This one is not intended to be used to upload your project's code, but to allow you to connect more peripheral devices, such as webcams and extra storage.

6. **Flash memory**: This type of memory is persistent and it is where the board firmware is stored, taking most of the available 8 MB of space.

7. **Random Access Memory (RAM)**: This is where your sketches are stored while running. Galileo has 512 KB of in-built SRAM and an additional 256 MB of external DRAM. Since it is a volatile type of memory, when you reboot your board, your sketch will be lost. If you wish to keep it persistent, you'll need to save it to a microSD card.

8. **Arduino expansion header**: It has 14 digital I/O pins (IO2-IO13, TX, RX); all of them can be used as input or output and six of them can be used as Pulse Width Modulation (PWM) outputs. The RX and TX pins control the programmable speed UART port. At the bottom-right side of the expansion header, you'll find six available analog pins with a 12-bit resolution. The pins at the bottom-left of the board are power pins (IOREF, VIN, RESET, 3.3 V, 5 V, and 2 GND).

9. **VIN jumper**: This jumper connects the Galileo VIN pin to the 5 V regulator. When using shields that require more voltage than this, you must pull out this jumper to avoid damaging the board.

10. **IOREF jumper**: In order to support 3.3 V shields, you can use this jumper to change your board voltage level from 5 V to 3.3 V.

11. **Reboot button**: This button reboots the board, including the OS.
12. **Intel Quark SoC X1000 Application Processor:** This is the board's processor; it is responsible for processing your code. It is a 32-bit, single core, single-thread, Pentium (P54C/i586) instruction set architecture (ISA)-compatible CPU. It is capable of operating at speeds up to 400 MHz.

13. **Clock battery power:** With this inclusion, you won't need to get the date and time from the Internet every time you reboot your board. By connecting a 3 V coin cell battery to the board, you'll be able to keep track of time, even when the board is powered off.

14. **On board LED:** This is an on-board LED, directly connected to the pin 13. You can use it to test and run basic sketches.

15. **JTAG header:** This is used to debug boards. It should be used with an in-circuit debugger.

16. **Reset button:** Pressing this button will restart your code and send the reset signal to the connected shields. It won't restart the OS.

17. **MicroSD card slot:** You'll definitely need more space to store your sketches or other apps. Here, you can insert your microSD card and store your persistent sketches or even use it to boot an operating system instead of the on-board one. You'll need to do this if you want additional functionalities, such as Wi-Fi, since those drivers can't fit in the 8 MB of the board's flash memory. It can be done using a card up to 32 GB.

18. **Power input:** This is where you must connect your power adapter. The 5 V, 2A feed is the only official way to power the board. Intel recommends you power the board through its power supply before connecting it via USB to your computer, otherwise, you might damage your board.

As you can see, this board is more suitable to work with sensors. The Arduino shield compatibility, familiar IDE, real-time clock and, possibility of using the PCI express connector are some of its best features.

**Galileo Gen 2**

Many makers found the 400 MHz processor a bit slow for their projects, mostly because of Arduino being emulated with Linux. Intel addressed the community issues, made some changes to the original board, and presented a new one named Galileo Gen 2.

Galileo Gen 2 is still powered by the same processor, but its performance has been considerably increased. The Arduino shields compatibility has also been improved with 12 GPIOs, now made fully native by being connected directly to the Quark X1000 SoC, and 12-bit PWM resolution allowing faster and smoother responses.
This board is a bit bigger than the original one, and the 3.5 mm serial port jack has been replaced by a six-pin 3.3 V USB TTL UART header, now making it compatible with the standard FTDI to USB serial cable. Also, the USB host port was replaced with a full size Type A receptacle 2.0 USB port.

The power regulation system has been changed to accept power supplies from 7 V to 15 V. The power supply jack isn't the only powering option available; it is now possible to power this board through the Ethernet cable by connecting it to a **Power over Ethernet (PoE)**-enabled Ethernet switch. It is also possible to power it from a connected shield, as long as the input voltage applied to the Galileo's VIN pin is in the 7 V – 15 V range:

![Intel Galileo Gen 2 board](image)

For the demos in this book, you can either use Galileo or Galileo Gen 2. The projects will run on both.
A comparison of Galileo with the most popular boards

Along with Galileo, Raspberry Pi and Arduino Yún are very popular boards among makers. All of them are Linux-embedded and have open source hardware design.

The following is a table with some features of each board:

<table>
<thead>
<tr>
<th>Features</th>
<th>Intel Galileo</th>
<th>Arduino Yún</th>
<th>Raspberry Pi model B</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU speed</td>
<td>400 Mhz</td>
<td>400 Mhz</td>
<td>700 Mhz</td>
</tr>
<tr>
<td>Memory</td>
<td>256 MB</td>
<td>64 MB (AR9331) and 2.5 KB (ATmega)</td>
<td>512 MB</td>
</tr>
<tr>
<td>Internal storage</td>
<td>8 MB</td>
<td>16 MB (AR9331) and 32 KB (ATmega)</td>
<td>-</td>
</tr>
<tr>
<td>External storage</td>
<td>MicroSD</td>
<td>MicroSD</td>
<td>SD card</td>
</tr>
<tr>
<td>Networking</td>
<td>Ethernet and Wi-Fi (Wi-Fi adapter is bought separately)</td>
<td>Ethernet and Wi-Fi</td>
<td>Ethernet and Wi-Fi (Wi-Fi dongle is bought separately)</td>
</tr>
<tr>
<td>Video output</td>
<td>-</td>
<td>-</td>
<td>HDMI and 1080p composite RCA</td>
</tr>
<tr>
<td>Audio output</td>
<td>-</td>
<td>-</td>
<td>HDMI and 3.5 mm audio jack</td>
</tr>
<tr>
<td>Digital I/O pins</td>
<td>14 at 3.3 V or 5 V</td>
<td>20 at 5 V</td>
<td>17 at 3.3 V</td>
</tr>
<tr>
<td>Analog input</td>
<td>6 (12-bit ADC)</td>
<td>12 (10-bit ADC)</td>
<td>-</td>
</tr>
<tr>
<td>PWM output</td>
<td>6</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Real-time clock</td>
<td>Optional</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SPI</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>I2C</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Introducing Galileo

Here's the brief comparison of the boards:

- **Arduino Yún**: At first glance, we can say that Yún and Galileo have more in common than with the Raspberry Pi. The number of available PWM, analog and digital pins make them good boards for projects with sensors. Arduino Yún is compatible with most Arduino Leonardo shields. Although Galileo shares a look alike development environment and board setup, not all the shields are compatible because some of the Arduino libraries are heavily bound to the Arduino architecture. Galileo runs a custom Yocto-based Linux in its 8 MB SPI flash by default. To be able to install stronger tools, it requires to be booted from a microSD card image provided by Intel. Yún runs an OpenWrt distribution, and has 16 MB of space available for the entire operating system. It is not possible to boot load from the microSD card, but is possible to increase its disk space by plugging in a microSD card and configuring it to become the new Linux file system. An advantage of Yún is that it already brings Wi-Fi on its board. Galileo requires connecting a Wi-Fi adapter.

- **Raspberry Pi**: Raspberry Pi, the board with the biggest community, is the only board in our table that has video and audio output, making it more interesting for multimedia projects. However, the lack of analog inputs makes this board less interesting to work with analog sensors.

Galileo is compatible with Arduino in the way that it can run 3.3 and 5 V shields, but it has some restrictions because of the Arduino's AVR libraries dependencies. Before buying a shield, you should check whether it is supported by Galileo.

Some of the advantages of the Galileo board when compared to the mentioned ones are the possibilities of working with PCI Express mini cards and using a real-time clock. Besides these, Galileo comes fully ready to work with sensors; this makes it an interesting tool for data collection.

Like the Raspberry Pi, the possibility of booting from a stronger Linux image makes it possible for projects developed in some of the most popular languages such as Python or Node.js (Javascript).
Summary
Galileo is a good option if you have a project requiring sensors, monitoring, or device control. It is an interesting board to develop ambitious projects in the scope of the Internet of Things, where you can develop your unique Things and make them share data with each other.

In this chapter, you've learned about the IoT concept, your board components, and where you should connect what. By now, you may already be able to imagine what you are able to use and connect to your board.

In the next chapter, you'll get familiar with the Arduino development environment, learning how you can develop code and run it in your Galileo.
Where to buy this book

You can buy Internet of Things with Intel Galileo from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.