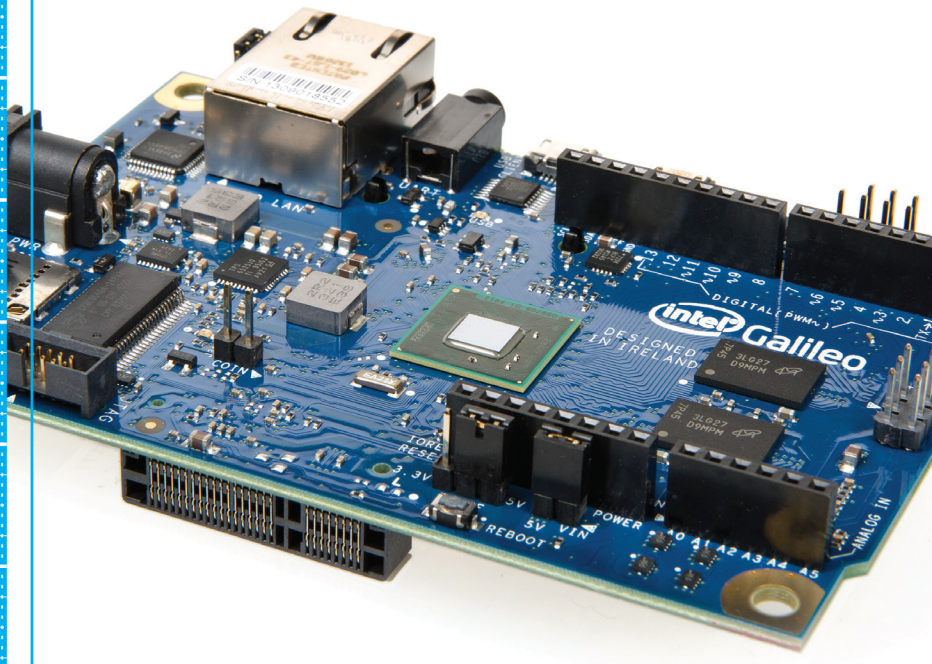
Make:
Getting Started with Intel Galileo



Electronic Projects With The Quark-Powered Arduino-Compatible Board

Matt Richardson

Getting Started with Intel Galileo

Matt Richardson

MAKER MEDIA™
SEBASTOPOL, CA

Getting Started with Intel Galileo

by Matt Richardson

Copyright © 2014 Awesome Button Studios, LLC. All rights reserved.

Printed in the United States of America.

Published by Maker Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Maker Media books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact O'Reilly Media's corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editor: Brian Jepson
Production Editor: Melanie Yarbrough
Proofreader: Gillian McGarvey
Cover Designer: Juliann Brown
Interior Designer: David Futato
Illustrator: Rebecca Demarest
Photographer: Matt Richardson
Cover Photographer: Jeffrey Braverman

March 2014: First Edition

Revision History for the First Edition:

2014-03-11: First release

See *http://oreilly.com/catalog/errata.csp?isbn=9781457183089* for release details.

The Make logo and Maker Media logo are registered trademarks of Maker Media, Inc. *Getting Started with Intel Galileo* and related trade dress are trademarks of Maker Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Maker Media, Inc., was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN: 978-1-457-18308-9

[LSI]

Contents

Preface . . . . . **vii**

1/Introduction to Galileo **1**
What Is Galileo? **2**
Inputs and Outputs **2**
Code **4**
Communication **4**
What Makes Galileo Different? **5**
Sketching in Hardware **8**

2/First Steps **9**
Tour of the Board **10**
Helpful Tools and Components **13**
Writing Programs to Control Your Galileo **16**
Getting Familiar with the Development Environment **17**
Connecting the Board **18**
Uploading Code **20**
Taking It Further **22**

3/Outputs **25**
Back to Blinking: Digital Output **26**
Setup and Loop **26**
Variables **27**
Pin Numbers **28**
Circuits and the Flow of Electricity **34**
pinMode() **37**
digitalWrite() **38**
delay() **39**
Code and Syntax Notes **39**
Going Further with Digital Output **41**
Analog Output **42**
analogWrite() **43**
Code and Syntax Notes **47**

Other Outputs. 50
Serial Data Output. 50
Controlling A/C Appliances with Relays. 54
Controlling Servos. 55
Looking at Linux. 60
Connecting via Telnet. 61
Working with Pins. 62
Taking It Further. 65

4/Inputs. 67
Switches: Digital Input. 68
digitalRead(). 73
Code and Syntax Notes. 74
Analog Input. 75
Potentiometers. 76
analogRead(). 80
Code and Syntax Notes. 81
Variable Resistors. 82
Code and Syntax Notes. 87
Going Further. 88

5/Going Further with Code. 91
Data Types. 92
int. 92
float. 92
long. 93
boolean. 93
char. 94
String Object. 94
millis(). 95
Other Loops. 95
while. 95
do... while. 95
for Loops. 96
More Serial. 99
Serial.available() and Serial.read(). 99
Taking It Further. 100

6/Getting Online. 103
Connecting and Testing an Ethernet Connection. 104

Connecting and Testing with a WiFi Connection. 105
Connecting Using Linux Commands. 107
system(). 108
Getting Galileo's IP Address Using system(). 108
Connecting to Servers. 110
How Many Days Until MAKE Comes Out?. 110
Serving a Web Page. 123
Serving a Web Page with Python. 126
Taking It Further. 129

A/ Arduino Code Reference. 131

B/ Breadboard Basics. 147

C/ Resistor Reference. 153

D/ Creating a MicroSD Image. 157

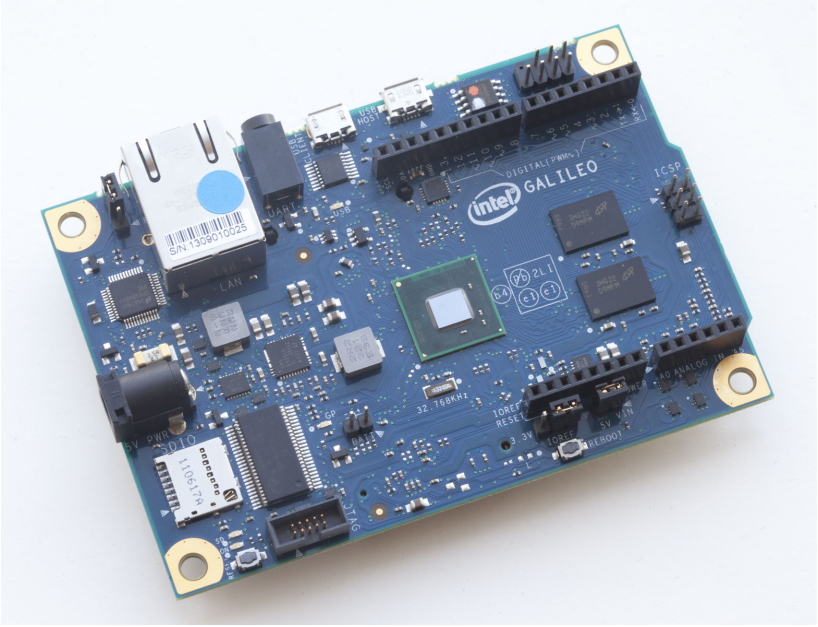
E/ Setting Up Galileo on Windows. 161

F/ Setting Up Galileo on Linux. 167

G/ Setting Up Galileo on Mac OS X. 171

H/ Connecting to Galileo via Serial. 175

Preface



Intel Galileo is a hardware development board that lets you write code and create electronic circuits to build your own projects. It's capable of acting as the brain in a robot, controlling haunted house special effects, uploading sensor data to the Internet, and much more.

The board doesn't do very much on its own, so it's up to you connect the right hardware and write the code to tell it what you want it to do. In that sense, Galileo is like a painter's canvas. It doesn't become anything remarkable until you start to work with it.

Luckily, since Galileo is Arduino-compatible, you have a vast amount of resources from the world of Arduino available to you. These include code

examples, libraries that help you do complex things, expansion shields that make it easy to connect circuits, and a simple development workflow—which means you spend more time being creative and less time figuring out how to get things to work. Not only that, but you also have access to the enormous community of Arduino users if you run into trouble.

Why Galileo?

When Intel announced Galileo at Maker Faire Rome in October of 2013, there was already an abundant selection of hardware development boards to choose from. At the time, there were so many boards available that an issue of MAKE magazine (Volume 36, Board Games) was released to take on the task of featuring the most interesting boards and helping readers choose the right one for them.

"We're now seeing an explosion of new boards coming to market," wrote Alasdair Allan in that issue of MAKE. "And there's no reason to expect the trend to slow in the next year or two." With so many boards out there, why did Intel decide to jump into this market?

After the announcement of Galileo, Intel CEO Brian Krzanich explained why Galileo came to be. "We wanted to be part of the Arduino ecosystem and maker community for two reasons," said Krzanich to Maker Media's founder, Dale Dougherty. "One was the pure innovation we see happening in the maker community around open source hardware, and we needed to be part of that innovation. Second, we saw that, in education, engineers and others were learning on non-Intel platforms and we wanted to change that, and in doing so, give them more capabilities."

Like the Galileo, the development boards that were gaining popularity had fairly powerful processors, similar to those found in cell phones and tablet computers. What they typically didn't have was an easy-to-use development environment, a good out-of-box experience, or an established community of users. With its strong Arduino compatibility, Galileo excels in these realms. Galileo also gives you the power of Linux under its hood.

Linux is a free and open source operating system that many people run on their desktops and on servers. It's also used in many consumer electronic devices. There's a lot to understand about Linux, but with Galileo, you can focus on bringing your creation to life without needing to know that Linux is there. This makes it easy for users to get more power and capabilities without sacrificing ease-of-use or community support. As you'll see later in this book, you can do some amazing things by poking around under the hood.

Intended Audience

The purpose of this book is to get you started with creating your own hardware projects with Intel Galileo. You won't need any experience wiring up circuits or writing code, but basic computer skills will be helpful so that you can move files around and install the software you'll need to develop projects.

Getting Started with Intel Galileo is written to give you a wide variety of experience and a basic understanding of the many different capabilities of Galileo. It won't dig into electrical engineering or computer science theory. I'll leave that for you to learn elsewhere should you want to pursue those

subjects in depth. Instead, I'll focus on how to get things done so that you can experiment, be creative, and make cool stuff with Intel Galileo.

Feedback

I encourage you to contact me with any feedback as you read this book. I hope to be able to incorporate your suggestions into future editions. My email address is *mattr@makezine.com*. You can also find me on Twitter with the name @MattRichardson.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This element signifies a tip, suggestion, or a general note.

This element indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *https://github.com/mrichardson23/GSW-Intel-Galileo*.

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code

from this book does not require permission. Selling or distributing a CD-ROM of examples from MAKE books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Getting Started With Galileo* by Matt Richardson (Maker Media). Copyright 2014, 978-1-4493-4537-2."

If you feel your use of code examples falls outside fair use or the permission given here, feel free to contact us at *bookpermissions@makermedia.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, get exclusive access to manuscripts in development, and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

Maker Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from MAKE and other publishers, sign up for free at http://my.safaribooksonline.com.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

MAKE
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

MAKE unites, inspires, informs, and entertains a growing community of resourceful people who undertake amazing projects in their backyards, basements, and garages. MAKE celebrates your right to tweak, hack, and bend any technology to your will. The MAKE audience continues to be a growing

culture and community that believes in bettering ourselves, our environment, our educational system—our entire world. This is much more than an audience, it's a worldwide movement that Make is leading—we call it the Maker Movement.

For more information about MAKE, visit us online:

MAKE magazine: *http://makezine.com/magazine/*
Maker Faire: *http://makerfaire.com*
Makezine.com: *http://makezine.com*
Maker Shed: *http://makershed.com/*

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://oreil.ly/getting_started_with_galileo

To comment or ask technical questions about this book, send email to:

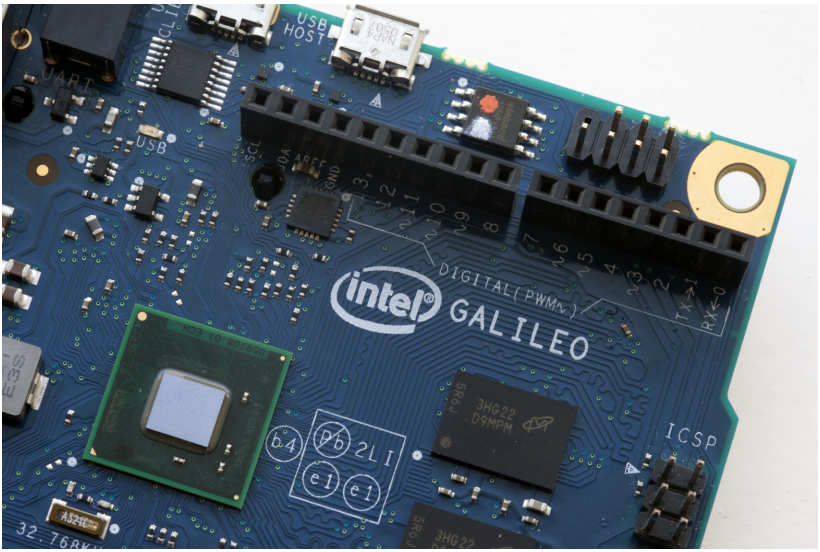
bookquestions@oreilly.com

Acknowledgements

I'd like to thank a few people who have provided their knowledge, support, advice, and feedback to *Getting Started with Galileo*:

Larry Barras
Julien Carreno
Michael Castor
Jez Caudle
Pete Dice
Seth Hunter
Tom Igoe
Brian Jepson
Jerry Knaus
Eiichi Kowashi
Mike Kuniavsky
Michael McCool
Jay Melican
Eric Rosenthal
Andrew Rossi
Mark Rustad
David Scheltema
Jim St. Leger

1/Introduction to Galileo



The purpose of the hardware and software that make up the Arduino platform is to reduce complexity when making an electronic project. It's meant to let you experiment, invent, and explore creative uses of technology rather than getting bogged down in technical mastery. By offering compatibility with Arduino hardware and software, Intel Galileo delivers an easy-to-use platform but has more power and features than typical Arduino boards.

What Is Galileo?

Galileo is a *hardware development board*, which is an electronic circuit board that helps you develop interactive objects by reading information from the physical world, processing it, and then taking action in the physical world. If it's connected to a network, it can also communicate to other devices like web servers. Galileo is an Arduino-compatible development board.

What Is Arduino?

There are a few answers to the question, "What is Arduino?" First and foremost, it's a hardware development board like Intel's Galileo. There are models of boards such as the Arduino Uno, Arduino Mega, and Arduino Yún. Each of these Arduino boards has different capabilities. The most basic board, the Arduino Uno, is typically what people are referring to when they say "an Arduino."

There's also the Arduino IDE software, which is the computer application that you use write code and upload it to the board. Arduino is also the name of the language used to program the board.

If you're entirely unfamiliar with Arduino and want to learn more about it, the Arduino website has many resources including getting started guides, reference information, communities, projects, and news updates. The book *Getting Started with Arduino* by Massimo Banzi (O'Reilly) was my first guide to using the popular development board. It covers the design philosophy of Arduino ("The Arduino Way") and walks you through the basics of using it. This book will cover a lot of that ground as well, but tailored for the Galileo board.

Galileo is an *Arduino-compatible* board, meaning that it can be programmed with the Arduino IDE using the Arduino programming language. It's also compatible with the Arduino 1.0 pinout, the design specification that says which pins go where on the board. Because it's compatible with the Arduino 1.0 pinout, you're able to attach most *Arduino shields*. A shield sits on top of the board and expands the functionality of it. Common circuits to drive motors, control many LEDs, or play sounds can come in the form of shields. The pin layout compatibility also makes it easy to use Galileo when you're following tutorials written for the other Arduino boards.

Inputs and Outputs

Like other hardware development boards, Galileo reads inputs and can control outputs. An *input* brings information from the physical world into the board's processor. It can be as simple as the state of a button or switch but can also be the position of a dial or slider like you see on a sound mixing board. Sensors can also be used as inputs (see Figure 1-1) to read information from

the physical world. There are plenty of sensors to choose from including temperature, light level, sound level, acceleration, and much more.

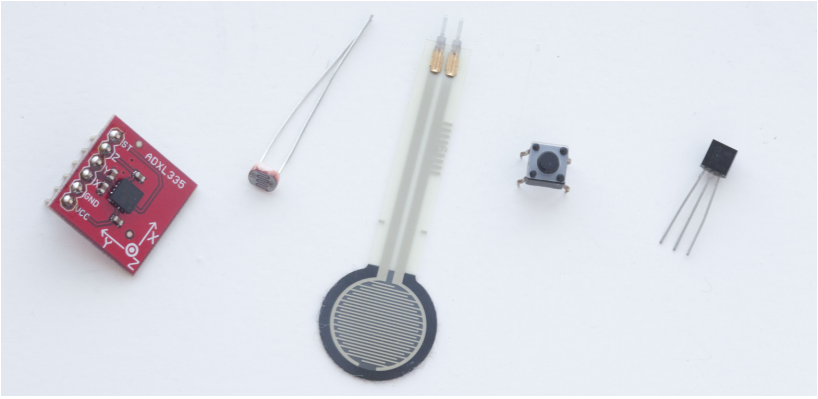


Figure 1-1. *A few possible inputs. From left to right: an accelerometer, a photo cell, a pressure sensor, a button, and a temperature sensor.*

An *output* is how a development board like the Galileo can affect the physical world. It can be as simple as a *light emitting diode*, or LED, which glows when electrical current runs through it. An LED might indicate whether the device is turned on, or if there's an error (a blinking red LED would be perfect for that). Outputs could also be motors that drive wheels on a robot, a text display for the temperature, or a speaker that plays musical tones. Figure 1-2 shows a few.

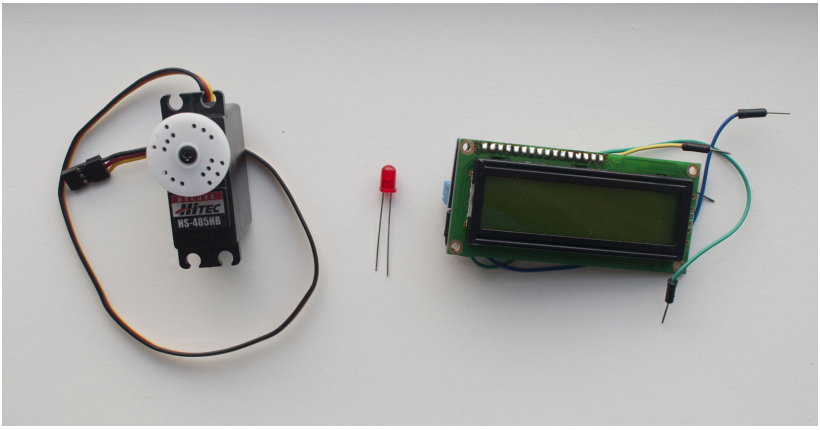


Figure 1-2. *A few possible outputs. From left to right: a servo motor, a light emitting diode, and an LCD character display.*

For example, a simple stopwatch has inputs and outputs. The start button would be considered an input. When you press the start button, it triggers a timer that keeps track of the time and outputs that information to the display on the face of the watch.

A digital voice recorder has a microphone for sound input, and a small speaker for sound output. Like the stopwatch, it also has input buttons to start or stop the recorder and a small display to output the amount of time that's left to record before you fill up the device's memory.

Code

Of course, it's not as simple as just wiring up inputs and outputs to a Galileo. You have to tell the board how you want it to respond to the inputs and how you want it to control the outputs. By programming the board, you'll be able to tell it what you want it to do.

For instance, a simple thermostat project will periodically check the value from a temperature sensor and compare it to the desired temperature that the user set using a dial control. If the temperature that the sensor measures is lower than the desired temperature, the board will activate a heater until the temperature gets close enough to the desired temperature. Logic like this will be defined by the code you write.

The Galileo can be programmed and reprogrammed over and over again. In fact, in the course of developing a project, you'll likely go through a cycle of writing code, uploading it to the board, checking how it works, finding problems, making adjustments to your code, and then uploading it again.

You may even find yourself using the board for one project, and then pulling the board out, reprogramming it, and using it for a completely different project a few weeks later.

Communication

The Galileo can also communicate with other devices in a few different ways. You can have it connect to your computer via USB to send and receive data. You might have Galileo send information about what it's doing to a console window running on your computer so that you can figure out why something isn't working right (this is known as *debugging*). Or you can have it send information about sensors to the computer so that it can display a live graph.

Galileo can also connect to other devices over the Internet using its built-in Ethernet (Figure 1-3) or an optional WiFi module. It can receive information about the weather or your email. It can search Twitter and much more. It can also use the Internet connection to send information such as temperature sensor data, the images from a webcam, or the state of your dog's water bowl.

Figure 1-3. *The Galileo's Ethernet port is just one way it can communicate with users or other devices.*

What Makes Galileo Different?

If you've used a typical Arduino like the Uno before, there are a few key differences between it and the Galileo (Figure 1-4). In fact, the specs on the Galileo make it seem like it's the product of cross-pollination between an Arduino Uno and a low-end computer.

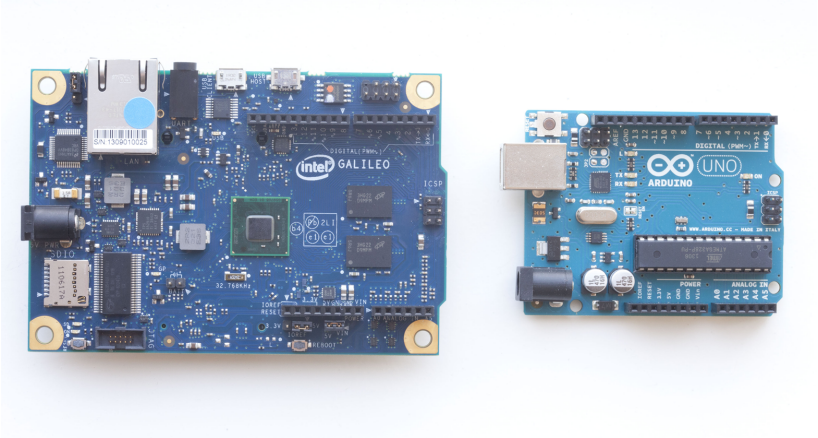


Figure 1-4. *An Intel Galileo next to an Arduino Uno*

The board itself is a little bit larger than an Arduino Uno, but along with that size, you get a more powerful processor (see Figure 1-5), more memory to store running programs, more data storage space, an Ethernet connector for connecting it to a network, and the ability to connect computer accessories through the USB port or the Mini PCI Express connector on the bottom.

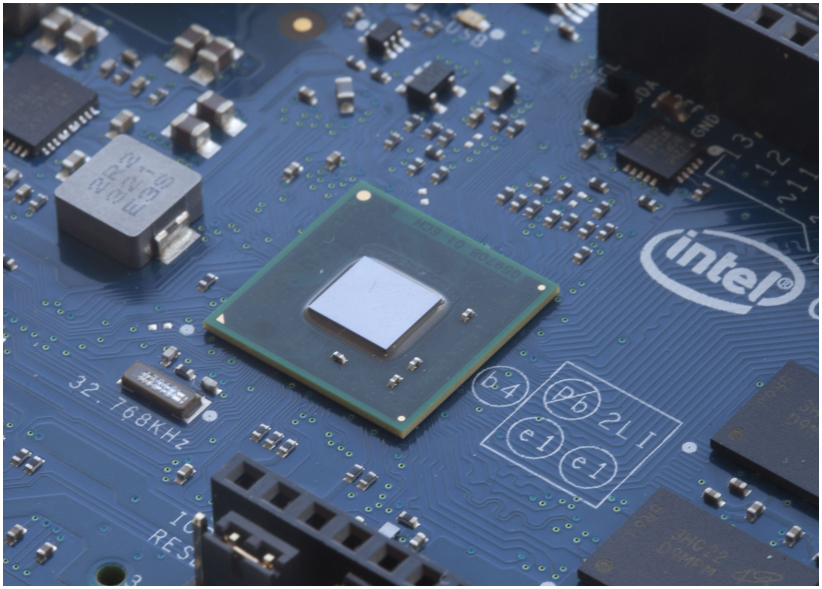


Figure 1-5. *Intel's Quark SoC X1000 is the processor at the heart of the Intel Galileo.*

The *firmware* that runs on the Galileo is much more advanced than what's on an Arduino Uno. On an Uno and most other Arduino devices, there is firmware on the board called the *bootloader* which is meant to help you upload and run your code on the board's processor. It only does that and not much else. The firmware on the Galileo, on the other hand, is much more advanced. Not only does it help you upload and run your code on the board, but it also keeps track of files, the date and time of day, and helps share the board's various resources between multiple programs running at the same time. In that way, it's more like a typical computer.

In fact, the firmware on the Galileo is a version of *Linux*, the free operating system that powers many desktop computers and servers these days. Galileo may not have a screen or desktop environment, but it still has much of the functionality that an operating system affords. And through your Arduino code, you'll be able to access this functionality, giving you much more capabilities than you'd have with a typical Arduino. For instance, if you want your project to take a picture from a web cam and email it, it's something that would be difficult to do with only Arduino code. But with the power of Linux, this could be done more easily.

Sketching in Hardware

Artists, engineers, designers, architects, and makers frequently start their work with a simple sketch on paper. Putting the idea down as a sketch helps by pushing something from being abstract towards something more concrete, more real. Sketching something out also helps you communicate your idea to your peers and collaborators. But you don't necessarily need to use a paper and pencil to create a sketch.

Having the power of a computer but the simplicity of Arduino development tools means that there's less to stand between you and your idea for an interactive object. It can help make the abstract idea a little more concrete. As a tool, the Galileo is meant to help you prototype early and iterate often so that you can get an idea of the look and feel of your project and refine it without hassle. I like to call this "sketching in hardware," a term I first heard from Mike Kuniavsky, who organizes a yearly conference by that name. According to Mike, the notion of this term originated with Bill Buxton's work on sketching in the realm of user experience design.

I don't want to be the one to stand between you and sketching in hardware, so let's jump right in.