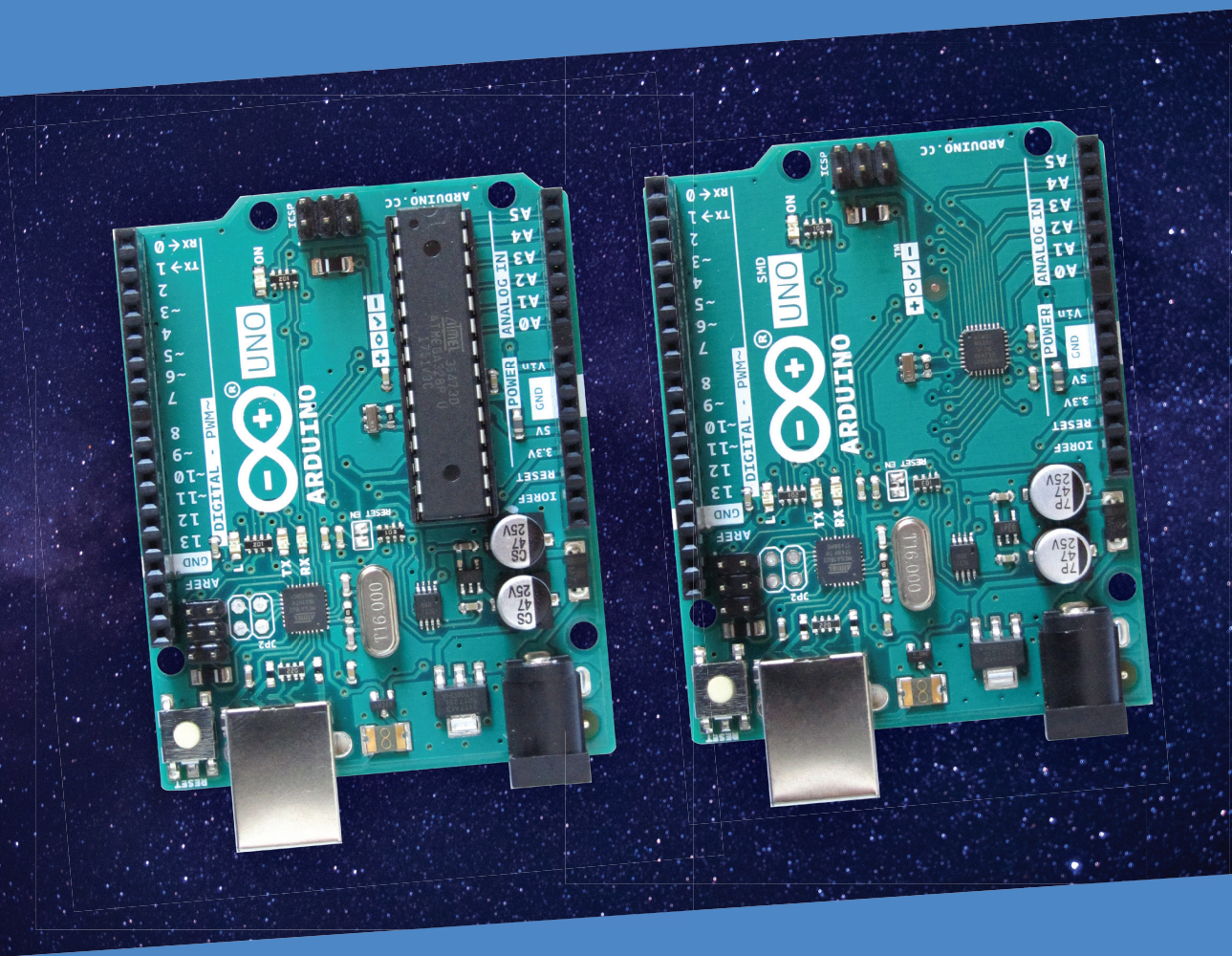
elektor books

Revised & Updated

A Reference and User Guide for the Arduino Uno Hardware and Firmware

ULTIMATE Arduino Uno Hardware Manual



Warwick A. Smith

elektor
design > share > sell

Ultimate Arduino Uno Hardware Manual

A Reference and User Guide for the Arduino Uno Hardware and Firmware

Warwick A. Smith

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

Declaration

The Author and the 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: 978-3-89576-434-9 (print) ISBN: 978-3-89576-435-6 (e-book)

Table of Contents

Introduction 13

Why Buy this Arduino Uno Hardware Manual? 13

Target Audience 14

Prerequisites 14

Hardware Requirements 15

Software Requirements 15

What is Covered and What's Not Covered 16

How to Use this Book 16

Accompanying Resources 17

Disclaimer, Errors and Corrections 17

A Note from the Author 17

Chapter 1 • Arduino Uno Overview 19

1.1 Arduino Uno Description and Functionality 20

1.1.1 Arduino Uno 20

1.1.2 Arduino Uno SMD 20

1.1.3 Uses of the Arduino Uno 21

1.1.4 Arduino Uno Main Parts 22

1.1.4.1 Main Microcontroller 22

1.1.4.2 USB Connector 23

1.1.4.3 External Power In 23

1.1.4.4 Reset Button 23

1.1.4.5 Header Sockets with Arduino Pins 23

1.1.4.6 ON LED 24

1.1.4.7 L LED 24

1.1.4.8 TX LED (Transmit) 24

1.1.4.9 RX LED (Receive) 24

1.1.4.10 ATmega16U2 Microcontroller 24

1.1.4.11 ICSP Header 24

1.1.4.12 ICSP Header for ATmega16U2 25

1.1.4.13 Mounting Holes....26
1.1.5 Programming....26
1.1.6 Extending the Hardware....28
1.1.6.1 Add-on Boards: Shields....28
Stacking Shields....28
Shield Reverse Connection Protection....29
1.1.6.2 Prototype Shields....29
1.1.6.3 Strip-board....30
1.1.6.4 Electronic Breadboard....30
1.1.6.5 Custom PCB....31
1.1.7 Open-Source, Licensing and Logo....32
1.1.8 Third Party Compatible Boards....32
1.1.9 Build Quality, Warranty and Safety....33
1.1.10 Arduino and Genuino....34
1.2 Arduino Uno Firmware....35
1.2.1 USB Bridge Firmware....35
1.2.2 Bootloader....35
1.3 Precautions During Handling and Usage....36
1.3.1 Static Electricity....36
1.3.2 Work Surface....38
1.3.3 Power....38
1.3.4 Voltage....38
1.3.5 Handling....39
1.3.6 Interfacing Precautions....39
1.4 Arduino Uno History and Revisions....40
1.5 First Time Use and Basic Testing....41
1.5.1 New Arduino Uno Default Behavior....41
1.5.1.1 Computer Drivers....42
1.5.1.2 Arduino Uno Hardware Behavior....43
On LED....43
LED Flashed by Bootloader....43
Factory or User Loaded Sketch Running....43
1.5.2 Loading a Sketch to an Arduino Uno....43
1.5.2.1 Select Board and Port....44
1.5.2.2 Loading a Test Sketch....45
Open the Test Sketch....45

Upload the Sketch....45
Modify the Sketch....45
1.5.2.3 Serial Port Demonstration....46
Serial Port Sketch Example....46
Serial Monitor Window....46
1.5.3 Basic Testing....48
1.5.3.1 Visually Inspect the Board....48
1.5.3.2 Power LED and Voltages....48
1.5.3.3 Check Expected Default Behavior....49
1.5.3.4 Is the Board Recognized by the Host Computer?....49
1.5.3.5 Load a Test Sketch....49
1.6 Arduino Uno References and Help....49
1.6.1 Installing Software....50
1.6.1.1 Windows....50
1.6.1.2 MAC OS X....50
1.6.1.3 Linux....50
1.6.2 Getting Started, Examples and Reference....50
1.6.2.1 Getting Started Guides....50
1.6.2.2 Arduino Examples and Tutorials....50
1.6.2.3 Building Breadboard Circuits....50
1.6.2.4 Arduino Software Reference....50
1.6.3 Getting Help....51
1.6.4 Related Open-Source Projects....51
1.6.4.1 Fritzing....51
1.6.4.2 Wiring....52
1.6.4.3 Processing....52
1.6.5 Arduino Uno Boards on the Web....52
1.6.5.1 Arduino Uno Web Page....53
1.6.5.2 Arduino Uno SMD Web Page....53

Chapter 2 • Hardware Technical Information 55

2.1 Microcontroller....56
2.2 Atmel, Microchip and AVR....57
2.3 Memory....57
2.3.1 Flash Memory....57
2.3.1.1 Flash Memory Size....58

2.3.1.2 Flash Wear....58
2.3.1.3 Data Retention....58
2.3.2 SRAM....58
2.3.2.1 Volatile Memory....58
2.3.2.2 SRAM Size....58
2.3.3 EEPROM....59
2.3.3.1 EEPROM Programming....59
2.3.3.2 EEPROM Size....59
2.3.3.3 EEPROM Wear....59
2.3.4 Adding External Memory....59
2.3.4.1 SD Cards (SPI Interface)....59
2.3.4.2 Flash and EEPROM Chips (SPI / TWI Interface)....61
2.3.4.3 SPI Devices....61
2.3.4.4 TWI and I²C Devices....61
2.4 Power and USB....61
2.4.1 USB Power....62
2.4.2 USB Connection and Cable....62
2.4.3 External Power....63
2.4.4 Battery Power....64
2.4.5 Operating Voltage....66
2.5 Operating Frequency....66
2.6 LED Indicators and Reset Button....66
2.6.1 ON LED....67
2.6.2 L LED....67
2.6.3 TX LED....68
2.6.4 RX LED....68
2.6.5 Reset Button....68
2.7 User Pin Headers....68
2.7.1 Power Pins....69
2.7.1.1 GND Pins....69
2.7.1.2 5V Pin....69
2.7.1.3 3.3V Pin....69
2.7.1.4 Vin Pin....69
2.7.2 IOREF Pin and Unconnected Pin....69
2.7.3 RESET Pin....70
2.7.4 Digital and PWM Pins....70

2.7.4.1 Output Pins....70
2.7.4.2 Pin Current Rating....72
2.7.4.3 Input Pins....73
2.7.4.4 PWM Pins....74
PWM Example Sketch....75
Calculating PWM Duty Cycle....76
PWM Frequency....76
PWM LED Control Example....76
2.7.4.5 Serial Port / UART Pins....77
2.7.5 Analog In Pins....78
2.7.5.1 Analog In Example Sketch....78
2.7.5.2 Floating Analog Input Pin....78
2.7.5.3 Calculating Analog In Voltage....79
2.7.5.4 Analog In Pins Used for TWI....79
2.7.5.5 Analog In Pins Used as Digital I/O....79
2.7.6 AREF Pin....80
2.8 Programming Headers....81
2.8.1 Programming the Arduino Uno with External Programmer on ICSP....81
2.8.2 Restore the Bootloader with IDE and External Programmer....82
2.8.3 ATmega16U2 ICSP Header....82
2.8.4 ICSP Programming Resources....82
2.8.5 Using an Arduino as an In-System Programmer....83
2.9 Shared Pins....83
2.9.1 Serial Port Pins....83
2.9.2 L LED Pin....83
2.9.3 TWI or I²C Pins....83
2.9.4 ICSP SPI Pins and Reset....84

Chapter 3 • Pin Reference and Interfacing 87

3.1 Pin Default and Alternate Functions....88
3.1.1 Shared TWI Pins....88
3.1.2 Shared SPI Pins....88
3.2 ATmega328P to Arduino Uno Pin Mapping....90
3.2.1 ATmega328P Ports....92
3.2.2 ATmega328P Alternate Pin Functions....92
3.3 Pin Types and Interfacing....93

3.3.1 Digital Input / Output Pins....93
3.3.1.1 Pins as Outputs....93
Why an LED Needs a Series Resistor....94
How to Calculate a LED Current Limiting Series Resistor....95
Current Sourcing and Current Sinking....97
Current Sourcing....97
Current Sinking....98
Current Limitation Per Pin....99
I/O Port Current Source Limits....99
I/O Port Current Sink Limits....101
Switching Heavier Loads with Transistors and Relays....103
3.3.1.2 Pins as Inputs....105
Pull-down Resistor....106
Pull-up Resistor....107
Internal Pull-up Resistors....108
3.3.2 PWM Pins....109
3.3.3 Analog Pins....110
3.3.4 TWI Bus Pins....111
3.3.4.1 TWI Interfacing Example....111
Pull-up Resistors....112
3.3.4.2 Accessing TWI Devices in Software....113
3.3.5 SPI Bus Pins....114
3.3.5.1 SPI Bus Interfacing Example....114
3.3.5.2 Accessing SPI Devices in Software....116
3.3.5.3 Accessing SD Cards in Software....116
3.3.6 Serial / UART Pins....116
3.3.6.1 Hardware Serial Port....117
3.3.6.2 Software Serial Port....117
3.3.7 Power Pins....118
3.3.7.1 GND Pins....118
3.3.7.2 5V Pin....118
USB 5V....118
External Power to 5V Regulator....118
3.3.7.3 3.3V Pin....119
3.3.7.4 Vin Pin....119
3.3.8 Reset Pin....119
3.3.9 IOREF Pin....120

3.3.10 AREF Pin....120
3.4 ICSP Header on Main Microcontroller....121
3.5 ICSP Header on USB Microcontroller....122
3.6 JP2 Header on USB Microcontroller....123
3.7 Finding the Datasheets....125
3.7.1 ATmega328P Datasheet....125
3.7.2 ATmega16U2 Datasheet....125
3.7.3 Datasheets for Other Components....125

Chapter 4 • Power Reference 127

4.1 Power Supply Specification....128
4.1.1 Operating Voltage....128
4.1.2 USB Power Input....128
4.1.3 External Power Jack Input....128
4.1.4 External Power Jack Pinout....129
4.1.5 Vin Pin as Power Input....129
4.2 Power Circuit....129
4.2.1 External Power In and 5V Regulator....129
4.2.2 Power On Indicator LED....130
4.2.3 USB Power In....130
4.2.4 Automatic Switch....131
4.2.5 3.3V Regulator....131
4.2.6 Power Header Socket....131
4.3 Power Supply Protection....132
4.3.1 Reverse Polarity Protection....132
4.3.2 5V Regulator Protection Features....132
4.3.3 3.3V Regulator Protection Features....132
4.3.4 USB Overload Protection....132

Chapter 5 • Arduino Uno Firmware and Bootloader 133

5.1 Updating the USB to Serial Bridge Firmware using DFU....134
5.2 Microchip Studio....135
5.3 USB Microcontroller Firmware....137
5.3.1 Backing up the ATmega16U2 Firmware with Microchip Studio....137
5.3.2 DFU Bootloader Firmware....139

5.3.3 USB to Serial Bridge Firmware....139
5.3.4 Programming the USB Microcontroller using ICSP....140
5.3.5. ATmega16U2 Fuse Settings....141
5.4 Main Microcontroller Bootloader....142
5.4.1 Backing up the ATmega328P Firmware with Microchip Studio....142
5.4.2 Optiboot Bootloader Firmware....143
5.4.3 Restoring the Bootloader....144
5.4.4 ATmega328P Fuse Settings....144
5.5 The RESET-EN Solder Jumper....148
5.6 Alternative Firmware Programming Methods....149

Chapter 6 • Circuit Diagram and Components 151

6.1 Circuit Diagram....152
6.1.1 Block Diagram....152
6.1.2 Main Microcontroller Circuit....153
6.1.3 USB Microcontroller Circuit....154
6.1.4 Power Supply Circuit....155
6.2 Component List....156
6.3 Component Positions on the Board....159
6.4 Getting an Electronic Copy of the Circuit Diagram....160

Chapter 7 • Fault Finding and Measurement 161

7.1 Basic Fault Finding and Repair....162
7.2 Replacing the Main Arduino Uno Microcontroller....162
7.2.1 Replacement Part....163
7.2.2 Removing the Old Microcontroller....163
7.2.3 Inserting the New Microcontroller....163
7.2.4 Loading the Bootloader....165
7.3 Voltage Measurements....165
7.3.1 Powered from USB....166
7.3.1.1 5V Test Points....166
7.3.1.2 USB Power and 3.3V Circuit Test Points....166
7.3.1.3 Measuring Voltage on the L and ON LEDs....167
7.3.2 Powered from External Power Supply....167

7.4 Waveform Patterns and Measurement....169
7.4.1 Testing for Presence of Microcontroller Clocks....170
7.4.2 Testing for PWM Waveforms....171
7.4.3 Testing the UART Output....172
7.4.4 TWI Signals....173
7.4.5 SPI Signals....175

Chapter 8 • Mechanical Dimensions and Templates 177

8.1 Measurements, Tolerance and Scale....178
8.2 Length, Width and Mass....178
8.3 Mounting Hole Spacing and Size....180
8.4 Shape Dimensions....181
8.5 Header Positions....182
8.6 Shield Reverse Connection Protection....183
8.7 Using Strip-board as a Shield....184
8.8 Drill Template and KiCad Template....184

Appendix A • Specifications Quick Reference 185

Index 189

Introduction

Welcome to the Arduino Uno Hardware Manual, a reference and user guide for the Arduino Uno hardware and firmware. This manual provides up to date hardware information for the popular Arduino Uno, the easy to use open-source electronics platform used by hobbyists, makers, experimenters, educators and professionals.

Why Buy this Arduino Uno Hardware Manual?

There are four main reasons to buy this book besides just wanting a handy reference manual for the Arduino Uno hardware and firmware for use on the workbench. These reasons are discussed below.

1) All the Hardware Information in One Place

Although there is a lot of information about the Arduino Uno hardware available online, it is spread out over many pages and websites. Having all of the Arduino Uno hardware information in one place, in this concise guide, is convenient and a time saver. This is especially true for new Arduino users who may not even know what to look for when getting familiar with the Arduino Uno hardware, for example, how to extend the hardware, how to add external data memory, how to connect hardware to Arduino pins in current sinking and current sourcing configuration. It also includes practical information such as how to replace the main microcontroller, which firmware needs to be loaded to a new microcontroller and which other settings must be changed for it to work. In addition, this manual has detailed hardware technical information, a pin reference chapter, basic interfacing information, a power reference chapter, information on all firmware programs loaded on an Arduino Uno, the circuit diagram and component list, fault finding and testing steps, as well as hardware mechanical dimensions and measurements.

2) Integrity of Information

Information on the internet about Arduino or any other subject may or may not be technically correct. Discerning which information is correct and which is incorrect is a problem, especially for those users who are new to electronics. Even hardware specifications from Arduino themselves are often badly specified or even incorrectly specified. This manual provides hardware information on the Arduino Uno that has been carefully checked and verified to be correct.

3) Presentation of Information

Each topic or subject in this book has been broken down into smaller parts and carefully explained and described. For this reason, each chapter contains many sections, each with its own sub-heading and section number for easy reference and cross-reference. Numerous illustrations and figures are included for easier understanding of explanations and for quick and easy reference.

4) Information Not Available Elsewhere

This book contains some information that is not available online, or is hard to find. It also offers unique views and illustrations of the hardware that reveal some interesting aspects of the Arduino Uno hardware that is not obvious when looking at the circuit diagram or other sources of information.

Target Audience

This manual has been written for anyone interested in the Arduino Uno who would like an easy to use hardware reference, including hobbyists, makers, experimenters, teachers, students, and professionals such as electronic engineers.

Prerequisites

It is assumed that the reader of this manual has at least used the Arduino Uno on a basic level and has some basic knowledge of electronics. This is not a book that explains the basics of electronics, but assumes that the reader already has the knowledge to understand terms such as voltage and current, and knows what electronic components such as resistors and transistors are. In any case, this is the basic knowledge that is required to use an Arduino. Those readers who do not have this basic knowledge must at

least be willing to learn, by using additional resources, be it via the internet or by using other books on the subject.

Hardware Requirements

At a minimum, an Arduino Uno will obviously be required, as well as a computer with which to program the Arduino board. This book mainly references the Arduino Uno revision 3 (REV3 or R3) board which is the latest Arduino Uno at the time of writing. It also mentions earlier versions of the Arduino Uno when necessary and makes some reference to clone or compatible boards. For advanced use and programming firmware to the Arduino Uno, one of several USB programmers can be used – see the text for more details, but most users will not need to do this, unless updating firmware on a new microcontroller, and even then there are cheaper alternatives such as using one Arduino to update the firmware on a second Arduino.

In the pin reference and interfacing chapter, an electronic breadboard and jumper wires are used to show how to interface LEDs, transistors, a relay, an external TWI memory chip and an SD card adapter. These are just interfacing examples and users will only need the components for a particular project or interfacing example that they are interested in. Hardware testing requires at least a multimeter. Some tests require an oscilloscope, but it is not essential for every user and is only used in a small section of the book.

Software Requirements

Without software, an Arduino Uno does absolutely nothing. For this reason, although this is a hardware manual, the free Arduino IDE programming environment must be installed. This enables Arduino programs or sketches to be loaded to the Arduino Uno to operate and test the hardware.

All software used in this book is either free and open-source, such as the Arduino IDE software, or free software that is not open-source. All software used can easily be downloaded from the internet at no cost. Other software, besides the Arduino IDE, is optional and will only need to be installed if required for certain tasks.

What is Covered and What's Not Covered

This book is primarily about hardware on an Arduino Uno board, including technical information, a pin reference, power supply information, firmware details, circuits and parts list, fault finding and measurement, as well as mechanical information, but has to include some software, in the form of user programs or sketches, in order to get the hardware to work. That being said, it does not cover software programming in great detail, but does provide references to the appropriate example sketches in the Arduino IDE and online documentation where necessary. Most of the functionality of an Arduino Uno comes from the main microcontroller on the board. This microcontroller is covered by the text in the context of Arduino and how it is set up and used by the Arduino IDE and its libraries. Advanced use of some of the internal hardware in the main microcontroller that is not available in the Arduino IDE and libraries is not covered, except to provide a pin reference for these advanced features that shows which Arduino pins are mapped to advanced hardware. Users who want to use the advanced hardware that is not available in the Arduino IDE or libraries must reference the datasheet for the main microcontroller.

How to Use this Book

It is suggested that this book is first read through once to get a good idea of its contents and a better understanding of the Arduino Uno hardware, it can then be used as a reference manual. Although this book has been written as a manual, it is not dry like some manuals or textbooks. Anyone interested in Arduino should find it an enjoyable read.

For electronic engineers or competent hobbyists who have used other microcontroller boards, but not an Arduino Uno, this book provides an easy introduction to the Arduino Uno hardware that will get new users started quickly with the board. The first two chapters of the book are basically a hardware user manual, while the remaining chapters are more like a reference manual, although there is some overlap between the two.

When using the book for reference, use the index to find specific topics and use appendix A to quickly find technical specifications. Appendix A has a cross reference for each technical specification that references the same specification in the main text, which provides detailed information. Each section in the book cross references related material in other sections and chapters. The table of contents is also useful for referencing information, as it is a list of every section in the book.

Accompanying Resources

Accompanying resources can be found on the Elektor (www.elektor.com) web page for this book, and on the author's supporting website (wspublishing.net) which includes links to all of the online resources so that they do not need to be typed in manually. Arduino software sketches used in the book are also available from the supporting website.

Disclaimer, Errors and Corrections

Although this manual is not an official Arduino manual, or endorsed by Arduino in any way, it has been written and thoroughly checked by a competent electronic engineer. The information in this manual has been carefully checked and proof read, but this is not a claim of infallibility, and this manual, like any work, is subject to typos and small errors that may have been missed during the proof reading process. In the event that you do find an error or typo, please use the accompanying website (wspublishing.net) to report it. This will be a big help to others as any corrections will be published on the supporting website.

A Note from the Author

Thank you for purchasing this manual! I hope that you thoroughly enjoy reading this book and using it as a reference. Please give an honest review of this book on the web page for this book at the Elektor website (www.elektor.com). This will provide feedback to the author that can be used to improve future versions of this manual. When writing a review, bear in mind that this book is an Arduino Uno hardware manual and not a software or electronics book. Review the book contents in light of its title and stated purpose. Also bear in mind that some information in this book is repeated because of the layout of the book – with chapter 1 being an overview of the Arduino Uno, chapter 2 containing technical information and chapter 3 a pin reference, some information is repeated, but more details are added in each chapter. The alternative to this is to lump everything related together, in which case the book can not be laid out in an easier to read format that builds the information up from a top level view of the hardware followed by more details on each aspect of the hardware.

Chapter 1 • Arduino Uno Overview

Before taking a closer look at each aspect of the Arduino Uno hardware in greater depth, it is necessary to get a top level overview and general understanding of the Arduino Uno. This chapter, together with chapter 2, can be thought of as a “hardware user manual” for the Arduino Uno and covers all of the basics that are needed before a more comprehensive study of the Arduino Uno hardware that is contained in the chapters that follow.

Included in this chapter is a general overview of the Arduino Uno, its main parts, and how to extend its hardware.

Basics of the firmware, the software that comes factory loaded on a new Arduino Uno board, is explained. The chapter wraps up with board handling precautions, a brief history of the Arduino Uno and the hardware revisions that it has been through, some information on first time use, and basic testing.

In this Chapter

- A top level view of the Arduino Uno hardware and functionality
- A brief look at the Arduino Uno firmware and what it does
- Board handling and usage precautions
- Arduino Uno history and revisions
- First time use and basic testing
- References to quickly find information on the Arduino Uno

1.1 Arduino Uno Description and Functionality

The Arduino Uno is a single board computer that uses a microcontroller as its main processor to run software loaded via the Arduino IDE or other programming environment. It comes in two different versions, namely, the Arduino Uno and Arduino Uno SMD. These boards are identical except for the physical package of the main microcontroller.

1.1.1 Arduino Uno

Most users will want to get the Arduino Uno (Figure 1.1) rather than the Arduino Uno SMD board because it has a socket that houses the main microcontroller. This allows easy replacement of the microcontroller should it be damaged or bricked up by software. The main microcontroller is the 28-pin chip that can be seen above the text “POWER” and “ANALOG IN” in Figure 1.1.

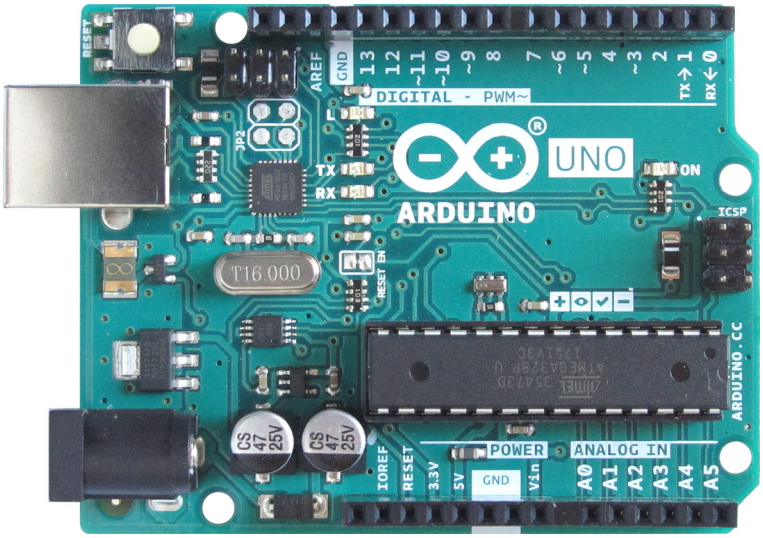


Figure 1.1: Top View of an Arduino Uno

1.1.2 Arduino Uno SMD

On the Arduino Uno SMD board, the main microcontroller is a surface mount device (SMD) which can be seen above the word “POWER” in Figure 1.2. This means that it can not easily be replaced if it is ever damaged, as it is directly soldered to the top of the board. It is almost impossible to replace this microcontroller without using a professional soldering rework station.

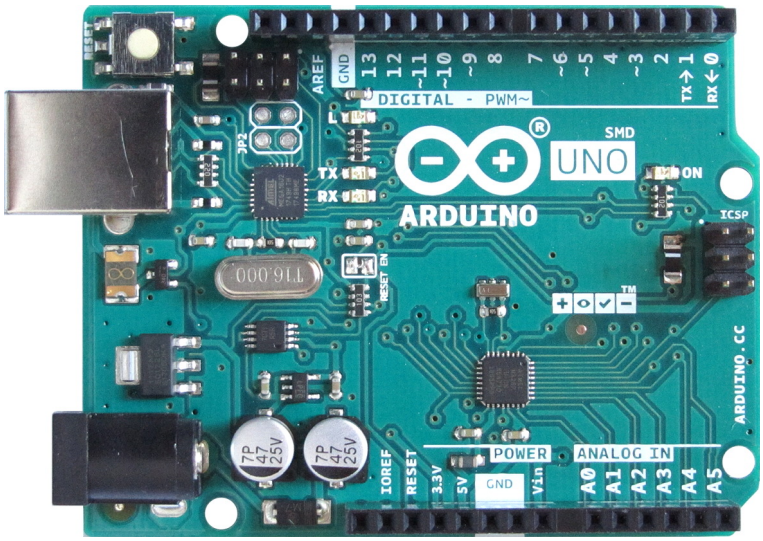


Figure 1.2: Arduino Uno SMD Board

According to the Arduino website, the Arduino Uno SMD board was made in response to a shortage in supply of the through-hole version of the microcontroller. Through-hole here refers to the way the microcontroller is mounted on the Arduino Uno board of Figure 1.1 where the pins of the microcontroller socket go through holes in the Arduino board and are soldered underneath. The through-hole microcontroller then plugs into the socket.

1.1.3 Uses of the Arduino Uno

Arduino is based on the open-source Wiring project (wiring.org.co) which was originally intended for art and design students who wanted to use electronics in their projects. Originally the Wiring board and the first Arduino (before the Arduino Uno) were used in education for student projects. Arduino has become popular beyond its original design purpose and is now used by everyone from hobbyists, makers, hackers and educators to engineers and other professionals.

Some examples of Arduino projects are robots (many different kinds, including drawing robots and plotters), colored light sequencers and other light displays, clocks, electronic games, devices for measuring temperature and humidity, vending machines and many more. A good place to look for Arduino projects, other than doing an internet search, is at the Arduino Project Hub (create.arduino.cc/projecthub) where projects that use the Arduino

Uno can be filtered using the products selection menu. Projects can also be filtered by category, difficulty and type.

Other uses of an Arduino Uno are for quick prototyping and testing of electronic components. For example, a temperature sensor can easily be wired to an Arduino Uno and tested by writing a simple software sketch using the Arduino software programming environment (called the Arduino IDE). Temperature readings can then be displayed in the Serial Monitor window of the Arduino IDE to see whether the sensor is working.

1.1.4 Arduino Uno Main Parts

Figure 1.3 shows the main parts of an Arduino Uno that will be of interest to a normal user of the board. Each of these parts is described in more detail in the sections that follow.

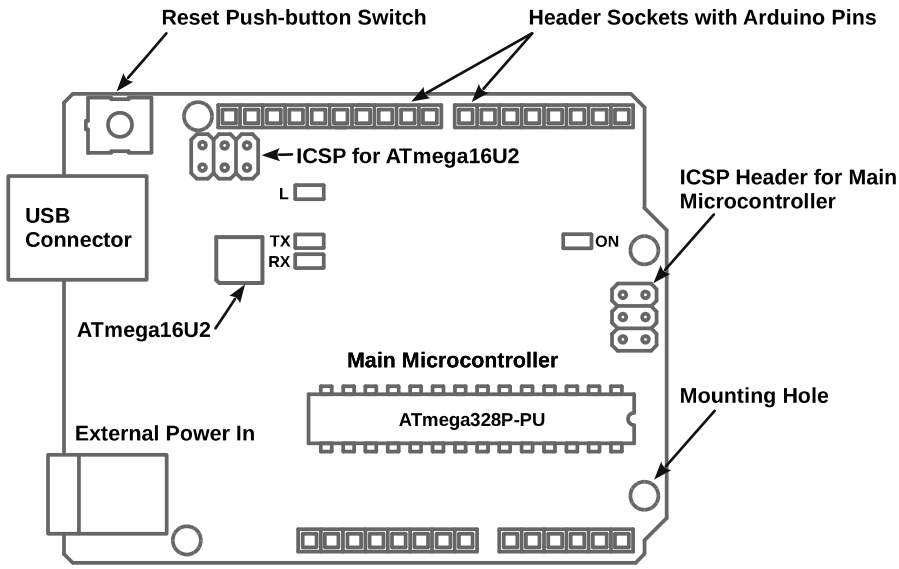


Figure 1.3: Main Parts of the Arduino Uno

1.1.4.1 Main Microcontroller

The main microcontroller (part number ATmega328P) runs sketches, or user programs, that are loaded to the Arduino from the Arduino IDE programming environment. Section 2.1 of the next chapter has more information on the main microcontroller of both the Arduino Uno and Arduino Uno SMD boards.

1.1.4.2 USB Connector

The USB connector can be used to both power the Arduino Uno and program it. USB programming is the default method of programming an Arduino Uno when using the Arduino IDE software. The USB connector is a standard type B connector. Use an A to B USB cable for connecting the Arduino Uno to a computer.

A user program, called a sketch, can use the USB connection to send and receive serial data. This allows user programs to send data to, and receive data from a computer. The Arduino IDE has a utility called the Serial Monitor, which is one method that can be used to send and receive data over the USB link. Alternatively other serial terminal programs or custom software running on a computer can use the same link to communicate with the Arduino via USB.

1.1.4.3 External Power In

External DC (Direct Current) power with a voltage of between 7V to 12V can be applied to the external power connector of the Arduino Uno, allowing it to run stand-alone without the need of a computer connected to the USB port. When external power is supplied to the Arduino Uno and the USB cable is plugged in for programming, the Arduino will automatically switch to using external power.

External power can be in the form of a DC power supply that converts mains power to low voltage DC, for example a “wall wart” power supply. Alternatively, a battery can be used, as long as the voltage is in the correct range of 7 to 12V. When connecting external power it is very important to get the polarity of the supply connected to the Arduino correctly. The center pin of the external power connector is positive on the Arduino Uno.

1.1.4.4 Reset Button

A reset button next to the USB connector allows manual resetting of the Arduino. Pressing the reset button will reset the main microcontroller, causing the software that is loaded on the main microcontroller to start running from the beginning.

1.1.4.5 Header Sockets with Arduino Pins

Header sockets are found on either side of the Arduino Uno, along its length. Each hole in the socket is usually referred to as an Arduino pin. There are several types of pins, including power pins, analog input pins, digital pins and a few special pins. Section 2.7 of chapter 2 has more information on the pin types and functions.

1.1.4.6 ON LED

When power is supplied to the Arduino Uno, either from the USB connection, or from an external power supply, the ON LED will turn on, indicating that the Arduino Uno is powered up.

1.1.4.7 L LED

The L LED is a user programmable LED connected to digital pin 13 on an Arduino Uno. In other words it can be switched on and off by instructions or statements in any user sketch by controlling digital pin 13.

1.1.4.8 TX LED (Transmit)

When data is transmitted from the main microcontroller over the serial to USB link, the TX LED will flash or blink.

1.1.4.9 RX LED (Receive)

When data is received by the main microcontroller over the serial to USB link, the RX LED will flash or blink.

1.1.4.10 ATmega16U2 Microcontroller

An ATmega16U2 microcontroller in a tiny surface mount package can be found near the USB connector (see Figure 1.3). This microcontroller acts as a bridge between the USB connection and the main microcontroller for either programming the main microcontroller, or when a user sketch sends and receives serial data over the serial to USB link.

This microcontroller comes with factory loaded software, usually referred to as firmware, that gives it its functionality as a USB bridge. For most normal use, this microcontroller is never programmed by the end user.

1.1.4.11 ICSP Header

Most Arduino users won't need to use the ICSP header, except in the case where an extension board is used with the Arduino that happens to connect to the ICSP header. Some advanced operations are available through the ICSP header and are described further in this section. New Arduino users may just want to browse through the rest of this section to see what the ICSP capabilities are, but it is not necessary to know about them for normal use. ICSP stands for In-Circuit Serial Programming.

The ICSP header is found at the end of the board, on the opposite side to the USB connector, and is wired to the main microcontroller. This header has two main uses. It is used by some extension boards (known as shields), such as Ethernet shields which have a connector under them that connects to this header in addition to the normal header sockets down the sides of the board. The second use of the ICSP header is to connect an external USB programmer such as an Atmel-ICE, or any number of homemade or third party programming devices. An external programmer can be used to program the Arduino Uno through the ICSP header as an alternative to using the default USB programming. This allows the full capacity of the program memory to be used – factory loaded bootloader software that is necessary for USB programming to work uses up some of the program memory (where sketches are loaded to). When using an external programmer, software sketches overwrite the bootloader and make use of the memory that it was occupying.

External programmers also allow the internal programmable fuse settings of the main microcontroller to be easily changed using software such as Microchip Studio (which used to be called Atmel Studio, but was rebranded to Microchip Studio). For normal use these fuses are left at their Arduino factory set values. A new bootloader can easily be programmed to the main microcontroller using an external programmer. This means that a board on which the bootloader was overwritten, after using an external programmer to load user sketches, can have its bootloader restored by an external programmer. The board will then be able to use USB programming again without the need of an external programmer.

Alternative programming tools such as Microchip Studio and others can be used to program the Arduino using plain C or C++. Some external programming tools that connect to the ICSP header have debugging capabilities, and when used in combination with Microchip Studio or similar tools, allow single-step debugging and examination of the internal registers and variables of the main microcontroller. Those interested in learning the C language to program Arduino boards may be interested in the book *C Programming with Arduino ISBN 978-1-907920-46-2* by the same author, also published by Elektor.

1.1.4.12 ICSP Header for ATmega16U2

A second ICSP header can be found near the USB connector that connects to the ATmega16U2 microcontroller. Most Arduino users will never need to use this header.

This header can be used to load software to the ATmega16U2 USB bridge microcontroller using the same tools described for the main microcontroller ICSP header. It can be used to

update the ATmega16U2 firmware or for advanced users who are developing their own firmware for this microcontroller. An alternative to using this ICSP header to update the firmware on the ATmega16U2 is available, called DFU (Device Firmware Update) programming. DFU programming allows the ATmega16U2 to be programmed using the USB connection.

1.1.4.13 Mounting Holes

Four mounting holes can be found near the edges of the Arduino Uno. These holes can be used to mount the Arduino to a base plate or enclosure using screws, or bolts and PCB stand-offs.

1.1.5 Programming

Arduino programming software is available for Windows, Mac OS X and Linux, and is known as the Arduino IDE (Integrated Development Environment). Although this is a manual on the Arduino Uno hardware, it is still necessary to have the Arduino IDE programming software installed and ready, as it is used to run various programs for testing the hardware and for demonstrating how the hardware works. Without software, the hardware of the Arduino Uno does nothing.

User written programs that are typed into the Arduino IDE are known as *sketches*. Sketches are the equivalent of source code files found in other programming languages. Sketches are written in the editor window of the Arduino IDE where they can be verified to see if they contain valid code by clicking the Verify button. Valid sketches can be uploaded to the Arduino using the Upload button in the Arduino IDE. Whenever the Verify or Upload button is clicked in the Arduino IDE, the build process of the sketch is started, which goes through preprocess, compile and link steps to convert the sketch into a file usable by the main microcontroller.

The Arduino IDE is free to download from www.arduino.cc/en/software where the correct download must be selected for the desired operating system. For Windows, there are three options for installing the Arduino IDE: by downloading and running the Windows installer, from the Microsoft Store, or via a ZIP file. If using a Linux or Apple computer, refer to section 1.6.1 near the end of this chapter for links to online installation guides.

It is suggested to just download the ZIP file and extract the contents of the ZIP file (the folder found inside) to a convenient location, such as the desktop. Figure 1.4 shows the

contents of the folder that was extracted from the ZIP file and then copied to the Windows desktop. Make sure to extract the folder from the ZIP file and not just open the ZIP file – first open the ZIP file and then drag and drop the folder from the ZIP file to the desktop.

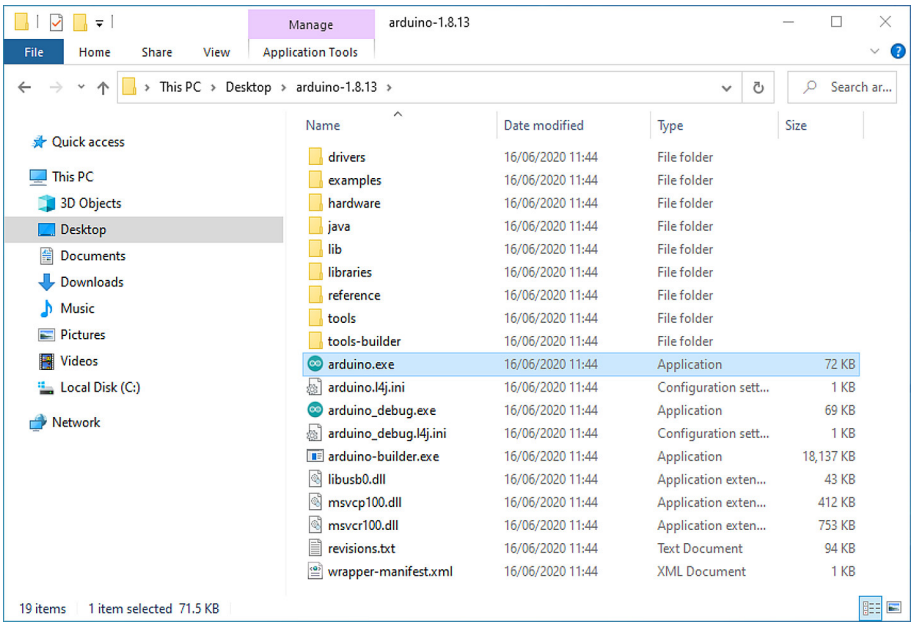


Figure 1.4: Arduino IDE Software Unzipped to a Folder on the Desktop

For those readers who may have already installed the Arduino IDE software using one of the other methods, this is not a problem. Having the Arduino software folder on the desktop does offer some advantages, but is mostly the preference of the author. One advantage is that it allows multiple versions of the IDE to be available on a single computer. It is possible that a newer version of an IDE can cause sketches of an older project to stop working. Building the same project using an older version of the IDE can resolve the problem. Although this type of software bug occurs rarely, it can and does still happen, hence the advantage of having different versions of the Arduino IDE on the same computer. For the purposes of this book, having the Arduino IDE software in an easy to find location on the desktop is convenient when we look at the firmware files that are located in this folder, rather than trying to find them in the file system. Users who have already installed the Arduino IDE using one of the other methods still have the option of downloading the ZIP file for the purpose of examining the software and easily finding the firmware files.

To start the Arduino IDE from the folder extracted from the ZIP file, simply open the folder and double-click the Arduino application file to run it. The Arduino application file can be seen highlighted in Figure 1.4. To remove the software from the computer, just delete the Arduino IDE folder. When a newer version of the software becomes available, the old version can either be left in place or deleted.

Alternative programming environments to the Arduino IDE are available, such as Microchip Studio, other free and commercial professional programming tools, and command line tools. This book uses the Arduino IDE software as the main method of programming the Arduino Uno. The Arduino IDE is the main or default way of programming Arduino boards, especially for new users.

1.1.6 Extending the Hardware

There are several ways to extend the Arduino hardware including: buying add-on boards, using prototype boards, building circuits on strip-board, using an electronic breadboard and making a custom PCB (Printed Circuit Board). A description of each of these methods for extending Arduino Uno hardware can be found in the sections that follow. Ultimately whichever method is used to extend the hardware, they all make use of the connector sockets found on each side of the Arduino Uno down its length. Hardware can be connected directly to any of the Arduino pins from the connector sockets, or use specific pins such as those that provide serial bus interfaces like SPI or TWI. SPI and TWI are explained further in section 2.3.4 – Adding External Memory, in the next chapter.

1.1.6.1 Add-on Boards: Shields

External add-on boards called *shields* that plug into the header sockets of the Arduino Uno are used to extend the hardware of the board. Shields are available from Arduino as well as third party vendors, and include such functionality as Ethernet, WiFi, motor control, USB host support, relays, data logging and more.

Stacking Shields

Shields can be stacked one on top of the other to add multiple hardware functionality. An example of an Ethernet shield plugged into an Arduino Uno board is shown in Figure 1.5 at the left. A stack of two shields is shown at the right of the figure where the first shield is plugged into the Arduino Uno and a second shield is plugged into the first shield.

When connecting more than one shield to an Arduino, it is important to know which pins of the Arduino connectors are used by the shields. In some cases the pins can be shared, such as when the pins are used as a serial bus such as SPI or TWI. In other cases the shields may not be compatible if they both use one or more of the same pins.

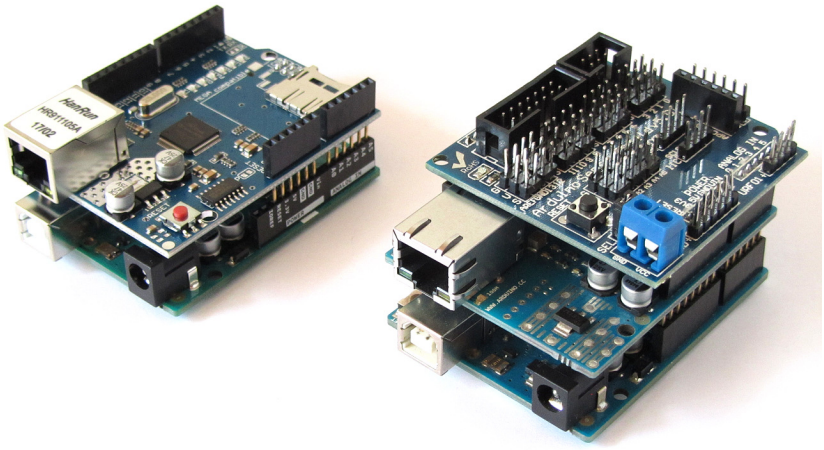


Figure 1.5: Arduino Uno with Ethernet Shield Plugged in (left) and Shields Stacked on an Arduino Uno (right)

Shield Reverse Connection Protection

To prevent shields from being inserted into an Arduino the wrong way around, one of the Arduino headers is spaced so that it is slightly offset from the headers on the other side of the board. This is essential for a classroom environment where hardware could easily be destroyed by students plugging in shields the wrong way around. For more information on the spacing of the headers on an Arduino Uno, see Chapter 8, section 8.6.

1.1.6.2 Prototype Shields

Besides ready made and assembled shields, prototype shields are also available that allow custom circuits to be built on them by soldering the desired components to the board. These prototype shields are available from Arduino and from third party manufacturers.

Figure 1.6 shows two examples of blank prototype shields. At the left of the figure is a third party prototype shield, and at the right is a Proto Shield Rev3 from Arduino, part number TSX00083 (store.arduino.cc/proto-shield-rev3-uno-size), which includes long pin

connectors. As can be seen in the figure, prototype shields are blank circuit boards, to which connectors can be soldered so that they will plug into an Arduino Uno like a shield does. They have holes and tracks to which components can be soldered to build a circuit.

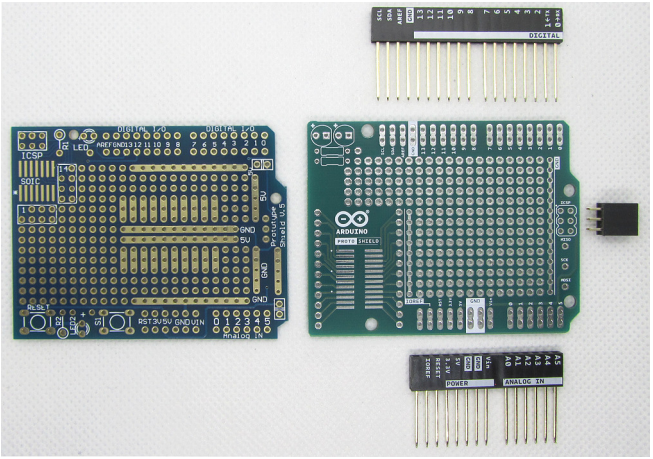


Figure 1.6: Prototype Shields for Arduino Uno

1.1.6.3 Strip-board

Strip-board, such as Veroboard, that has been available to electronic hobbyists for many years can be used to make custom shields. The main problem with strip-board is that one of the Arduino Uno headers is offset from the headers on the opposite side of the board, which means that the holes in the strip-board won't line up with the holes in the offset Arduino header, see chapter 8, sections 8.6 and 8.7 for more details. One option that allows strip-board to be used with an Arduino Uno is to not use any pins from the offset header. Another option is more of a hack where long pins are soldered in the strip-board and bent to fit into the offset header.

1.1.6.4 Electronic Breadboard

Temporary circuits can be built on electronic breadboards which allows electronic components to be plugged into and removed from them. Breadboard circuits can be connected to the Arduino Uno pins via the header sockets using jumper wires, as can be seen in Figure 1.7, which shows two LEDs with series resistors connected to digital pins of an Arduino Uno. A software sketch can be written to control the two LEDs in the circuit, switching them on and off. Components and wires can be unplugged from a breadboard and reused, which is ideal for prototyping and educational use.

In addition to discrete electronic components, various modules are available that can also be plugged into a breadboard to build circuits. Packs of modules are available that include, for example, 30 modules of various types, such as LED modules, light sensors, tilt switches, temperature and humidity sensors, relays and others. Each module consists of a small circuit board with components soldered to it to make up the circuit. Modules have pins that are used to plug them into a breadboard.

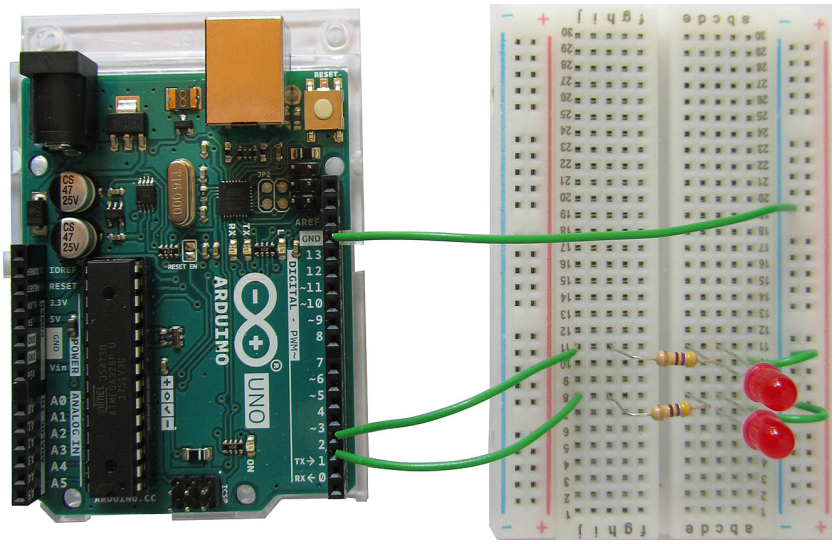


Figure 1.7: A Simple Breadboard Circuit Interfaced to an Arduino Uno

1.1.6.5 Custom PCB

A shield in the form of a custom made PCB (Printed Circuit Board) can be designed and built. This method is more suited to those who have some experience with electronics, as it requires some circuit design skills and ability to use a suitable electronic CAD (Computer Aided Design) package. Also see Fritzing in section 1.6.4.1 in this chapter.

Options here are to either etch a board at home or in a work laboratory, or send the PCB design out to be professionally manufactured. KiCad (kicad.org) is an open-source EDA (Electronic Design Automation) software package that can be used to draw circuit diagrams and lay out PCBs. A KiCad template for an Arduino shield is provided with this book – see Chapter 8, section 8.8 for details. Figure 1.8 shows an example of a custom homemade Arduino shield that was designed using KiCad.

Figure 1.8: Homemade Custom PCB Shield

1.1.7 Open-Source, Licensing and Logo

Arduino boards are open-source hardware which means that the design files for the hardware are available to examine and modify by anyone. The design files are licensed under a Creative Commons Attribution Share-Alike license which allows personal and commercial derivative works, but they must credit Arduino and release the derivative works under the same license. See creativecommons.org/licenses for more information on the various Creative Commons licenses.

The Arduino IDE software is also open-source which means that anyone can have access to the source code which can be examined and modified. Source code for the Java environment (used by the Arduino IDE software) is released under the GPL (General Public License). The C/C++ libraries are released under the LGPL (Lesser General Public License).

Arduino logos are not open-source, but are trademarks of Arduino. Use of the Arduino name is restricted and Arduino logos may not be copied. For more details, see the page www.arduino.cc/en/trademark on the Arduino website.

1.1.8 Third Party Compatible Boards

Because Arduino is open-source, many third party Arduino boards are available including third party clone boards and boards that are compatible with the Arduino Uno. Figure 1.9

shows a board that is compatible with the Arduino Uno. Although this board works like an Arduino Uno, it has some slight differences such as the physical package that the main microcontroller is housed in, which differs from both the Arduino Uno and Arduino Uno SMD boards manufactured by Arduino. Some other differences are the layout of components and the color of the various indicator LEDs on the board. Additionally this board does not use an ATmega16U2 for the USB bridge chip, but uses a cheaper alternative chip. Clone or compatible boards may offer some additional features to the official Arduino boards, such as the extra holes in the board of Figure 1.9 that can be used to solder pins, connectors or wire for the purpose of connecting circuits to the Arduino pins.

Third party Arduino clones and Arduino compatible boards vary in quality from excellent to poor quality. Some of the cheaper boards, usually from China, are quite usable, but there is always the risk of getting a dud. Note the Chinese board in Figure 1.9 has the ICSP header at the right incorrectly labeled “ISCP”. For more information on the various spin-off boards including official boards, clones, derivatives, compatibles and even counterfeits, see the article blog.arduino.cc/2013/07/10/send-in-the-clones/ on the Arduino blog.

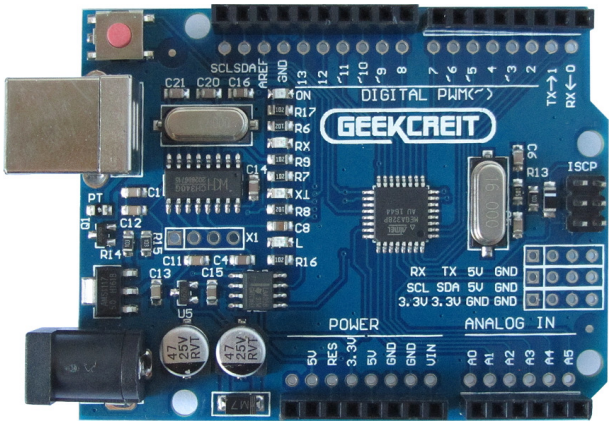


Figure 1.9: Arduino Uno Compatible Board from a Third Party Manufacturer

1.1.9 Build Quality, Warranty and Safety

Buying an official Arduino board manufactured by Arduino or one of its authorized manufacturers guarantees that the board will be a quality product, and that some of the money paid for the board will go back into supporting the Arduino community, such as for

further hardware development, documentation and hosting of the support forums. A full list of official Arduino boards can be found at www.arduino.cc/en/Main/Products on the Arduino website.

For more information on Arduino board FCC compliance and for the warranty statement, see www.arduino.cc/en/Main/warranty where probably the most important clause is with regards to safety-critical applications. This clause (clause 1.5) is included below, verbatim as it was found on the Arduino website at the above link at the time of writing.

" 1.5 Arduino LLC products are not authorized for use in safety-critical applications where a failure of the Arduino LLC product would reasonably be expected to cause severe personal injury or death. Safety-critical applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Arduino LLC products are neither designed nor intended for use in military or aerospace applications or environments, nor for automotive applications or the automotive environment. The Customer acknowledges and agrees that any such use of Arduino LLC products is solely at the Customer's risk, and that the Customer is solely responsible for compliance with all legal and regulatory requirements in connection with such use. "

1.1.10 Arduino and Genuino

When starting the Arduino IDE software, the splash screen shows both the name Arduino and *Genuino*. The reason for the Genuino name is because of a trademark dispute. The dispute has since been resolved.

A dispute over the trademark "Arduino" caused Arduino to split into two companies. A new Arduino company was started with a new website at **arduino.org** and the original Arduino company with the original **arduino.cc** website. The folks at arduino.cc owned the Arduino trademark in the USA, but not outside of the USA. They then came up with the name *Genuino* to use for boards that sold outside the USA, in order to sidestep the trademark issue, and the name *Arduino* for boards sold in the USA. Genuino was just a re-branding in order for the original team to be able to sell Arduino boards outside the USA. The whole legal battle has since been resolved and there is now only one Arduino website – arduino.cc and they own all of the trademark rights to the Arduino brand name.

elektorbooks

A Reference and User Guide for the Arduino Uno Hardware and Firmware

ULTIMATE Arduino Uno Hardware Manual

A manual providing up-to-date hardware information for the popular Arduino Uno, the easy to use open-source electronics platform used by hobbyists, makers, hackers, experimenters, educators and professionals.

Get all the information that you need on the hardware and firmware found on Arduino Uno boards in this handy reference and user guide.

- Ideal for the workbench or desktop.
- Contains all of the Arduino Uno hardware information in one place
- Covers Arduino / Genuino Uno revision 3 and earlier boards
- Easily find hardware technical specifications with explanations
- Pin reference chapter with interfacing examples
- Diagrams and illustrations for easy reference to alternate pin functions and hardware connections
- Learn to back up and restore firmware on the board, or load new firmware
- Basic fault finding and repair procedures for Arduino Uno boards
- Power supply circuits simplified and explained
- Mechanical dimensions split into five easy to reference diagrams
- Contains circuit diagrams, parts list and board layout reference to easily locate components

Warwick A. Smith is an electronics engineer and embedded programmer from South Africa with experience in industrial, commercial and aerospace related industries. Warwick has a wide interest in various fields of technology, including embedded systems hardware, software, and information technology. His writing style has been described as "clear and concise" as well as "conversational and friendly".

Elektor International Media BV
www.elektor.com

ISBN 978-3-89576-434-9
9 783895 764349

elektor
design > share > sell