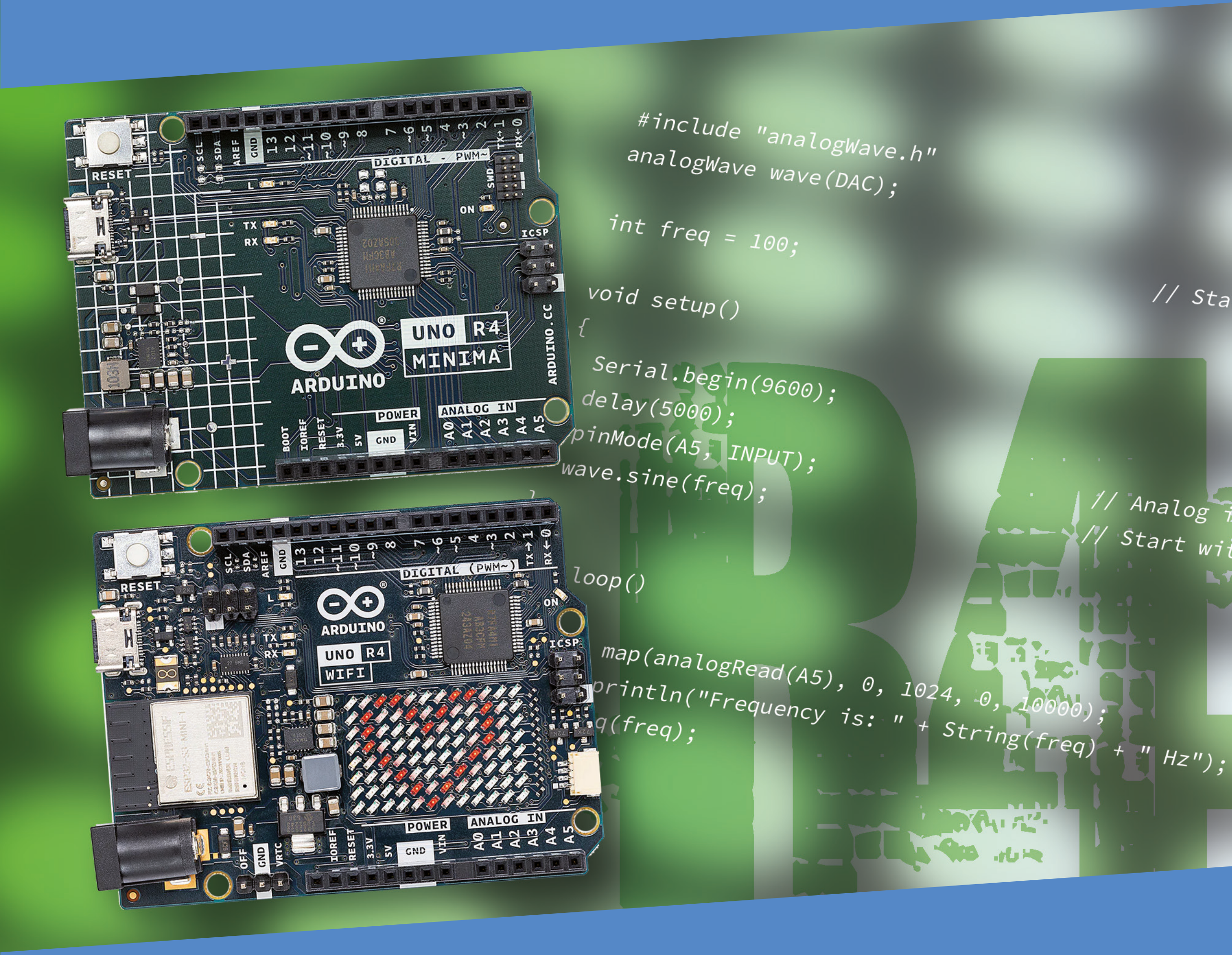
elektor books

Mastering the Arduino Uno R4

Programming and Projects for the Minima and WiFi



Dogan Ibrahim

elektor
design > share > earn

Mastering the Arduino Uno R4

Programming and Projects for the Minima and WiFi

●

Dogan Ibrahim

elektor
design > share > earn

● 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

● British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

● **ISBN 978-3-89576-578-0** Print
ISBN 978-3-89576-579-7 eBook

● © Copyright 2023: Elektor International Media B.V.
Editors: Alina Neacsu; Jan Buiting MA
Prepress Production: D-Vision, Julian van den Berg
Print: Ipskamp Printing, Enschede (NL)

Contents

Preface

"Arduino" is an open-source microcontroller development system that incorporates system hardware, an Integrated Development Environment (IDE), and a large number of libraries. Arduino is supported by a large community of programmers, electronic engineers, enthusiasts, and academics. There are several distinctive designs of the basic Arduino board. Although they are intended for diverse types of applications, they can all be programmed using the same IDE, and, in general, programs can be transported ("ported") between different boards. This is probably one of the reasons for the popularity of the Arduino family. Arduino is also supported by a large number of software libraries for many interface devices that can easily be included in your programs. Using these libraries makes programming relatively easy and speeds up the programming time. Using libraries also makes it easier to test your programs since most libraries have already been fully tested and working.

The **Arduino Uno R3** board probably ranks as the most popular Arduino family member to date and has been with us for many years. Based on the low-cost 8-bit ATmega328P processor, the Uno R3 has been used by students and hobbyists in many beginning and intermediate-level, low-speed projects requiring small to medium amounts of memory. Perhaps one of the attractive points of the Uno R3 was its powerful IDE and the simplicity of using it to develop projects in relatively short times.

Recently, the new Arduino Uno R4 was announced. This new board is compatible with the earlier Uno R3 but offers highly improved specifications compared to Uno R3. The new **Arduino Uno R4** is based on a 48 MHz 32-bit Cortex-M4 processor with a large amount of SRAM and flash memory. Additionally, a higher-precision ADC and a new DAC are added to the design. The contemporary design also supports the CAN bus interface. Two versions of the board are available: **Uno R4 Minima**, and **Uno R4 WiFi**.

This book is about using these new boards to develop many different and interesting projects. The projects given in the book have been fully tested with just a handful of parts and external modules, which are available as a kit from Elektor. The block diagrams, circuit diagrams, full program listings, and complete program descriptions are given for all the projects in the book. You should find it easy to build the project hardware and then follow the software descriptions given for the projects. It should also be relatively easy to modify the hardware and software for your own project applications.

I hope that you enjoy reading the book and at the same time learn how to use the Arduino Uno R4 Minima or the R4 WiFi models in your innovative projects.

Dogan Ibrahim
London, 2023

Publisher's Notice: All programs discussed in this Guide are contained in an archive file you can download free of charge from the Elektor website. Head to www.elektor.com/books and enter the book title in the Search box.

Chapter 1 • The Arduino Uno R4

1.1 Overview

The Arduino project started out as a tool for students at the **Interactive Design Institute**, **Ivrea** in Italy back in 2005. The aim of this project was to provide low-cost and easy-to-use hardware and software to beginner students and hobbyists to create simple projects using sensors and actuators. The name "Arduino" comes from the bar named *Arduin of Ivrea* where the project's founders used to meet for drinks. The name of this bar came from the Margrave *Arduin of Ivrea*, who was King of Italy from 1002 to 1014.

The initial Arduino project team consisted of Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis. In 2003 Hernando Barragan created the development platform *Wiring* as a master's thesis at the Institute Ivrea. Wiring was an open-source electronics platform consisting of a programming language, an integrated development environment (IDE), and a single-board microcontroller. This project was developed under the supervision of Massimo Banzi and Casey Reas. The Wiring platform included a printed circuit board with an ATmega128 microcontroller, an IDE, and some library functions. Later in 2005, Massimo Banzi, David Mellis, and David Cuartielles extended the Wiring platform by adding support for the ATmega8 microcontroller which was cheaper. This new project was named Arduino. The project was so successful that after developing less expensive versions, by mid-2011 over 300,000 copies of Arduino were produced commercially and by 2013 this number increased to over 800,000.

Arduino is an open-source hardware where the designs are distributed under a Creative Commons license and are freely available. The only point is that the developers are requested the name Arduino to be reserved for the official product and not be used for similar copy work without permission.

One of the nice features of the Arduino series is that a pre-programmed boot loader is used on the on-board processor. Users can develop their programs using the IDE and then upload their programs to the Arduino processor with the help of this boot loader program. The I/O pins are available at female headers located on either side of the board. This makes the hardware development very easy as jumper wires can be used to make connections to the board.

The original Arduino board was manufactured by the Italian company called Smart Projects. Many versions of the Arduino have been developed over the years by several companies. Some versions are:

- Arduino Diecimila
- Arduino Uno R2
- Arduino Leonardo
- Arduino RS232
- Arduino Pro
- Arduino Mega
- Arduino LilyPad

- Arduino Robot
- Arduino Esplora
- Arduino Yun
- Arduino Fio
- Arduino Ethernet
- Arduino Due
- Arduino Nano
- Arduino Uno SMD R3
- Arduino Uno R3
- Arduino MKR Zero
- Arduino Zero
- … and many more

The Arduino family has been so popular that in 2022, its revenue amounted to over US$237 million, including a large portion of online sales via the Internet.

Two new versions of the Arduino have recently been announced: **Arduino Uno R4 Minima**, and **Arduino Uno R4 WiFi**. In this book, you will be developing projects using both the Arduino Uno R4 Minima and the Arduino Uno R4 WiFi. The new two boards are similar to the very popular Arduino Uno R3 board but they have been expanded in many ways.

1.2 The Arduino Uno R4 against Uno R3

A comparison of the Uno R3 and Uno R4 is given in Table 1.1. Notice that almost all the projects and libraries used with the Uno R3 can be used with the Uno R4 without any modifications. It is recommended however to upgrade any libraries which may have been modified specifically for the Uno R4.

Feature	Arduino Uno R3	Arduino Uno R4
Processor	ATmega328P	Renesas RA4M1
Word length	8	32
Clock speed	16 MHz	48 MHz
SRAM	2 KB	32 KB
Flash memory	32 KB	256 KB
EEPROM	1 KB	8 KB
Operating voltage	5 V	5 V
Timers	3	10
Capacitive touch sensing	None	27 channels
Temperature sensor	None	1
USB connector	Type B	USB-C
ADC	10-bit	14-bit
DAC	None	12-bit
SPI	1	1

I²C	1	2
Qwiic I²C	None	1 (WiFi version only)
Operating voltage	5 V	5 V
Wi-Fi	None	WiFi version only
RTC	None	1 (WiFi version only)
Human Interface Device	None	Yes
SWD debug	None	1
Bluetooth BLE	None	WiFi version only
CAN bus support	None	1
Op Amp	None	1
128x8 LED matrix	None	WiFi version only
Input voltage	7–12 V	6–24 V
Analog inputs	6	6
PWM pins	6	6
USB	USB-B	USB-C

Table 1.1: Comparison of the Arduino Uno R3 and Uno R4.

The Arduino Uno R4 features the Renesas RA4M1 processor, which is an Arm 32-bit Cortex-M4 processor, running at 48 MHz. The Uno R3 had an ATmega328P processor with only a 16 MHz clock. This is a 3 times increase of the clock speed over the Uno R3. Additionally, Uno R4 has 32 KB SRAM memory compared to 2 KB on the Uno R3. The flash memory of the R4 is 256 KB, compared to only 32 KB on the Uno R3. As a result, more complex projects requiring more memory can be developed with the Uno R4. The USB port on the Uno R3 has been replaced with the currently standard USB-C and the maximum power supply voltage has been increased to 24 V with improved thermal design. The processor operating voltage is still 5 V. Arduino Uno R4 provides a CAN bus interface, allowing devices to be connected and programmed on a CAN bus environment. The ADC converter capacity has been increased from 12 bits to 14 bits on the Uno R4. SPI and I²C bus interfaces have been increased from 1 to 2. The Uno R4 supports Human Interface Device (HID) which enables users to simulate a mouse or keyboard when connected to a computer via a USB cable. Users can send mouse movements or keystrokes to a computer. Additionally, Uno R4 includes a true 12-bit DAC converter. The analogWave library was added to make using the DAC easy. Generating a sine, saw or square wave is as easy as calling a library function. Of course, you can do much more with it. The Uno R4 PCB is hardware compatible with the Uno R3. The pinout, voltage, and form factor are unchanged so that the Uno R4 can easily replace designs that use the Uno R3. The software IDE is also compatible between the Uno R3 and Uno R4, where an effort was made to maximize backward compatibility of the Arduino libraries so that users can use the existing libraries without any modifications. Some libraries which depend heavily on the AVR architecture may need to be re-loaded into the IDE. A public list of such libraries will be provided by Arduino.

Compared to other Cortex-based boards such as the Raspberry Pi Pico, which uses the Cortex M0+, the Cortex-M4 performance is about 6 times better (just to remind you, the Raspberry Pi Pico clock runs at 125 MHz by default). As a result, the Arduino Uno R4 can be used in highly complex real-time projects (e.g. digital control, DSP, AI, etc.) requiring large memory and fast throughput.

As mentioned earlier, Arduino Uno R4 is available in two versions: **WiFi** and **Minima**. The WiFi version is equipped with an Espressif S3 Wi-Fi module, making the board ideal in IoT-based network and Bluetooth BLE applications, as well as in applications requiring Internet connectivity. Additionally, a 128×8 LED matrix is included on the board. The Minima version offers a cost-effective option with no WiFi or Bluetooth connectivity. **In this book, all the projects compile and run on both versions, except that the WiFi and LED Matrix projects only compile and run on the R4 WiFi version.**

In summary, the Uno R4 is a giant leap forward for Arduino as it is a truly remarkable board that will take your microcontroller project experience to the next level. Perhaps one disadvantage of the Uno R4 compared to Uno R3 is its increased price.

1.3 The Arduino Uno R4 Minima hardware

Figure 1.1 shows the Arduino Uno R4 PCB layout. The PCB footpath and the placement of the headers on the PCB are the same as the Uno R3. In the center of the PCB is the RA4M1 microcontroller. On the left-hand side of the PCB, you can see the Reset button, USB-C connector for connecting to a PC, and the barrel connector for supplying external power. At the rear of the PCB, you can see the 6-pin ICSP pins (SPI) and the 10-pin SWD/JTAG pins. An on-board LED is connected to port 13 as in Uno R3. Additionally, yellow LEDs are connected to the serial TX and RX pins to indicate serial data transmission, a green LED indicates power to the board, and a yellow LED indicates the state of the SCK pin.

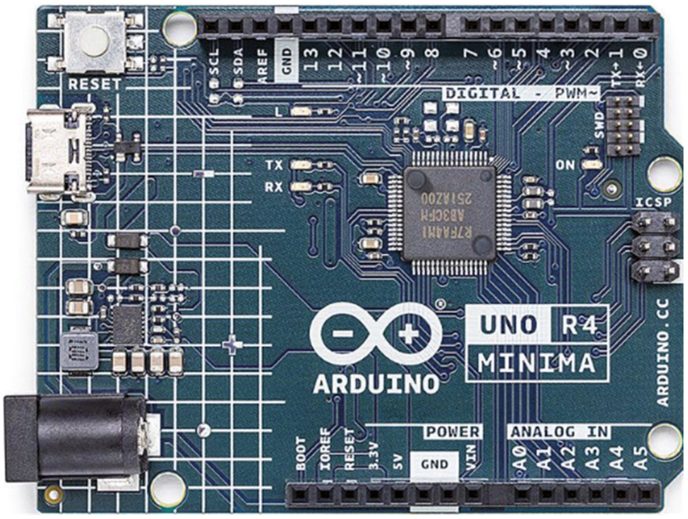


Figure 1.1: Arduino Uno R4 Minima PCB layout.

On the two sides of the PCB, you have the header connectors as in the Uno R3. The headers have the following pins (see Figure 1.2):

Short header connector:

- 6× 14-bit analog input pins (A0–A5)
- 1× 12-bit DAC (A0)
- 1× Opamp+ (A1), Opamp– (A2), Opamp Out (A3)
- I²C SDA (A4), I²C SCL (A5)
- Vin
- GND
- +5 V
- +3.3 V output (output from the RA4M1 VCC_USB pin)
- RESET
- IOREF (reference for the digital logic V. Connected to + 5 V)
- BOOT (mode selection)

Note: Analog pins A0–A5 can also be used as digital pins.

Long header connector:

- 14 × digital pins (D0–D13)
- External interrupt (IRQ00: D2, IRQ01: D3)
- 6 × PWM pins (D3, D5, D6, D9, D10, D11)
- UART (RX:D0, TX:D1)
- SPI (same as on ICSP header. D13: SCK, MISO (CIPO): D12, MOSI (COPI): D11, CS: D10)
- CAN (RX:D5, TX: D4 on Minima, RXD13, TX:D10 on WiFi, **Note**: external transceiver required)
- GND
- AREF (analog reference voltage, connected to +5 V through a 5.1-kΩ resistor)
- I²C SDA (pullups not mounted)
- I²C SCL (pullups not mounted)

ICSP connector:

- See Table 1.2

SWD/JTAG connector:

- See Table 1.3

Pin	Function	Type	Description
1	CIPO	Internal	Controller In Peripheral Out
2	+5V	Internal	Power Supply of 5 V
3	SCK	Internal	Serial Clock
4	COPI	Internal	Controller Out Peripheral In
5	RESET	Internal	Reset
6	GND	Internal	Ground

Table 1.2: ICSP connector pins (source: Product Reference Manual, SKU: ABX00080).

Pin	Function	Type	Description
1	+5V	Internal	Power Supply of 5 V
2	SWDIO	Internal	Data I/O pin
3	GND	Internal	Ground
4	SWCLK	Internal	Clock Pin
5	GND	Internal	Ground
6	NC	Internal	Not connected
7	RX	Internal	Serial Receiver
8	TX	Internal	Serial Transmitter
9	GND	Internal	Ground
10	NC	Internal	Not connected

Table 1.3: SWD/JTAG connector pins (source: Product Reference Manual, SKU: ABX00080).

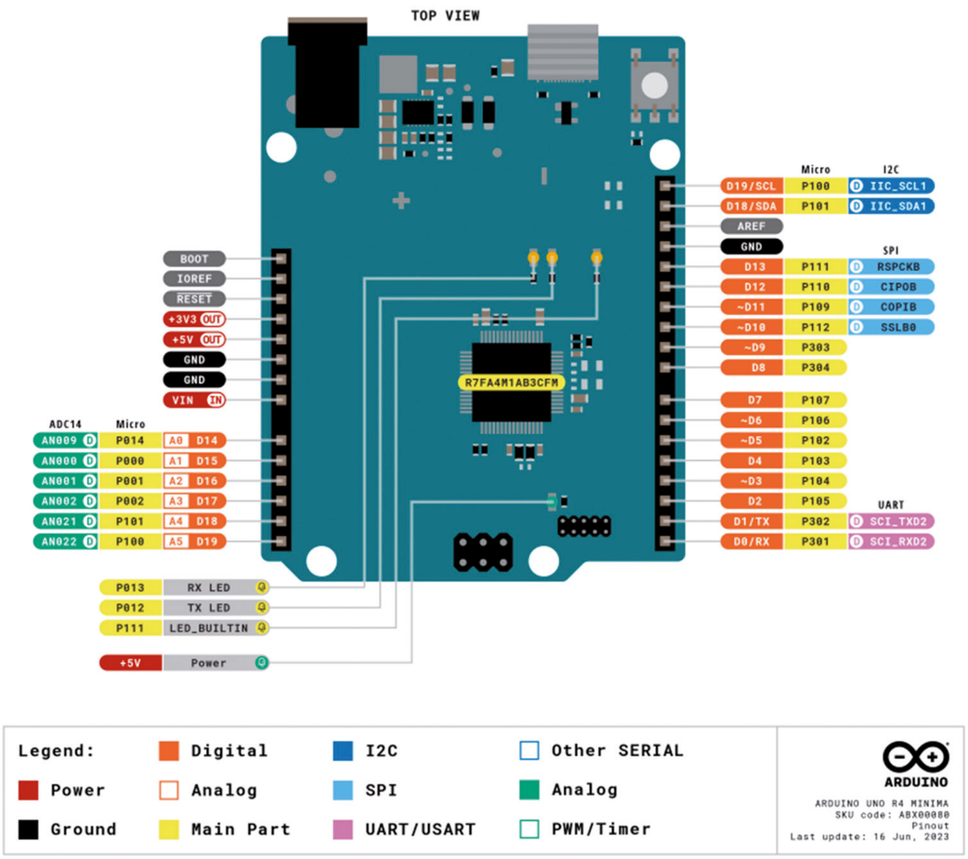


Figure 1.2: Header pins (source: Product Reference Manual, SKU: ABX00080).

The recommended operating conditions are shown in Table 1.4. Schottky diodes are used for overvoltage and reverse polarity protection. Power can either be supplied via the VIN pin, the barrel jack (DC jack), or via USB-C connector. If power is supplied via VIN, a buck converter steps the voltage down to 5 V. Power via USB supplies about 4.7 V (due to Schottky barrier forward bias) to the RA4M1 microcontroller. The RA4M1 processor can operate from +1.6 to +5.5 V and is connected to +5 V on the Arduino board. **The digital GPIOs on the RA4M1 microcontroller can handle currents of IOH = 4 mA and IOL = 8 mA (assuming middle pin drive) current. Remember that IOL is the current into the pin (sinking) when the pin is at logic 0, and IOH is the current from the pin (sourcing) when the pin is at logic 1. Care must be taken not to exceed the recommended current drives of the GPIO ports.** Figure 1.3 shows a simplified power supply connection of the Arduino Uno R4.

Symbol	Description	Min	Typ	Max	Unit
V_{IN}	Input voltage from VIN pad / DC Jack	6	7.0	24	V
V_{USB}	Input voltage from USB connector	4.8	5.0	5.5	V
T_{OP}	Operating Temperature	-40	25	85	°C

Table 1.4: Recommended operating conditions (source: Product Reference Manual, SKU: ABX00080).

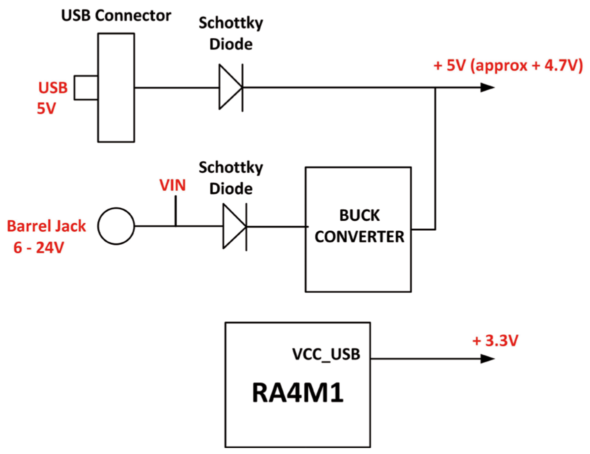


Figure 1.3: Simplified power supply connections.

Figure 1.4 shows the pin layout of the Arduino Uno R4 Minima board. The component layout is shown in Figure 1.5 (taken from Product Reference Manual: SKU: ABX00080) with the component descriptions shown in Table 1.5.

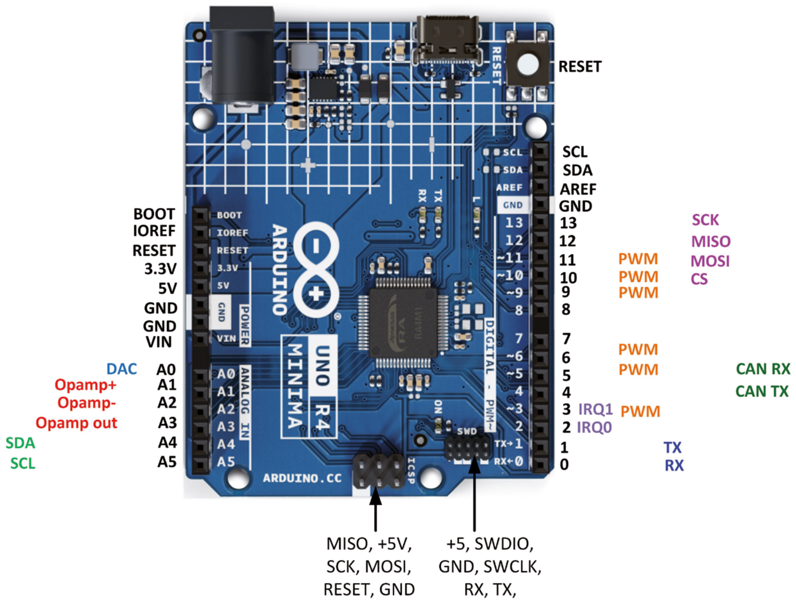


Figure 1.4: Arduino Uno R4 Minima pin layout.

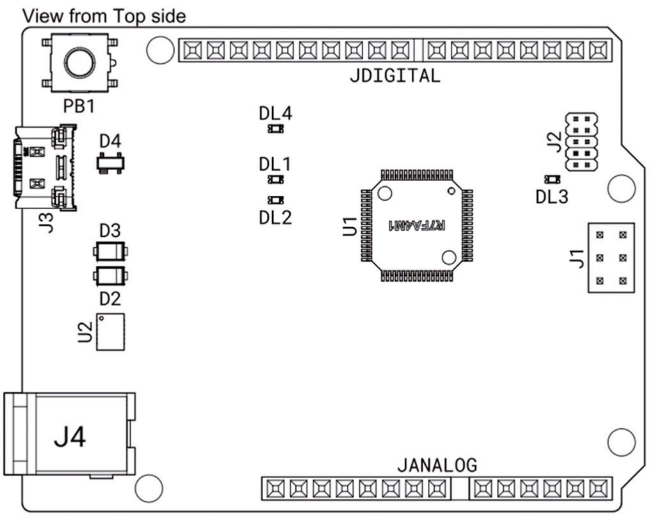


Figure 1.5: Arduino Uno R4 Minima component layout.

Ref.	Description	Ref.	Description
U1	R7FA4M1AB3CFM#AA0 Microcontroller IC	J4	DC Jack
U2	ISL854102FRZ-T Buck Converter	DL1	LED TX (serial transmit)
PB1	RESET Button	DL2	LED RX (serial receive)
JANALOG	Analog input/output headers	DL3	LED Power
JDIGITAL	Digital input/output headers	DL4	LED SCK (serial clock)
J1	ICSP header (SPI)	D2	PMEG6020AELRX Schottky Diode
J2	SWD/JTAG Connector	D3	PMEG6020AELRX Schottky Diode
J3	CX90B-16P USB-C® connector	D4	PRTR5V0U2X,215 ESD Protection

Table 1.5: Arduino Uno R4 Minima component layout.

1.4 The Arduino Uno R4 Projects Kit

The Arduino Uno Experimenting (SKU 20339) Kit available from Elektor (www.elektor.com) includes a large number of sensors, actuators, buttons, LEDs, plus a breadboard, stepper motor, jumper wires etc. In detail, the kit includes the following components:

- 1× RFID reader module
- 1× DS1302 clock module
- 1× 5 V stepper motor
- 1× Stepper motor 2003 drive board
- 5x Green Led
- 5× Yellow LED
- 5× Red LED
- 2× Rocker switch
- 1× Flame sensor
- 1× LM35 sensor module
- 1× Infrared receiver
- 3× Light-dependent resistor
- 1× Remote controller
- 1× Breadboard
- 4× Pushbutton (with four caps)
- 1× Buzzer
- 1× Piezo sounder
- 1× Adjustable resistor
- 1× 74HC595 shift register
- 1× 7-segment display
- 1× 4-digit 7-segment display
- 1× 8×8 Dot-matrix display
- 1× 1602 / I²C LCD module
- 1× DHT11 Temperature and humidity module
- 1× Relay module
- 1× Sound module
- Set of Dupont cables
- Set of Breadboard cables
- 1× Water sensor
- 1× USB cable

- 1× PS2 Joystick
- 5× 1 kΩ resistor
- 5× 10 kΩ resistor
- 5× 220Ω resistor
- 1× 4×4 keypad module
- 1× SG90 Servo
- 1× RFID card
- 1× RGB module
- 2× jumper cap
- 1× 9 V Battery DC clip-on cable

The kit is supplied in a plastic box with a lid as shown in Figure 1.6 (note: the actual packaging and contents of the kit as received may differ from the photo).

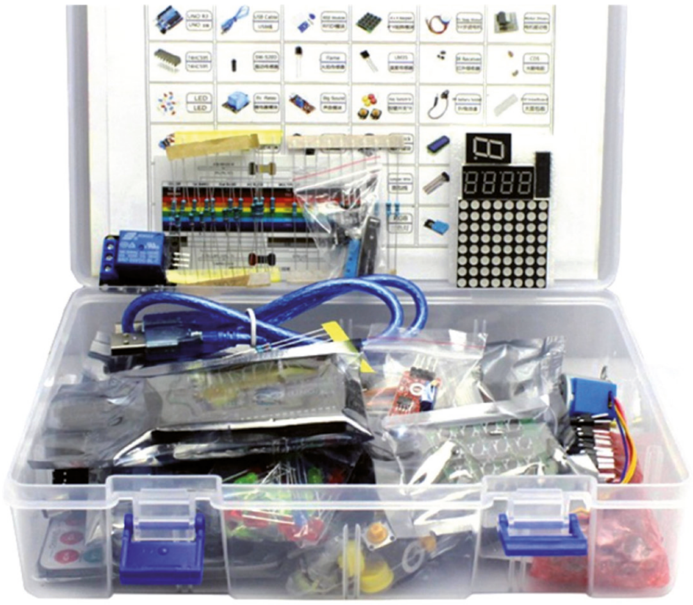


Figure 1.6: The kit supplied in a plastic box.

Figure 1.7 shows the supplied components and the Arduino Uno R4, which is not included and must be purchased separately. A close-up picture of the included sensors, actuators, and displays is shown in Figure 1.8. Figure 1.9 shows a close-up picture of the supplied LEDs, resistors, buttons, buzzers, breadboard and wire jumpers.

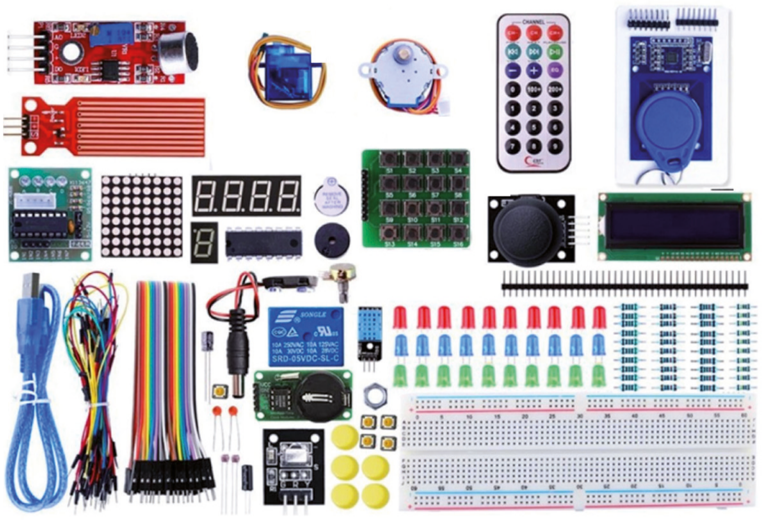


Figure 1.7: Supplied components.

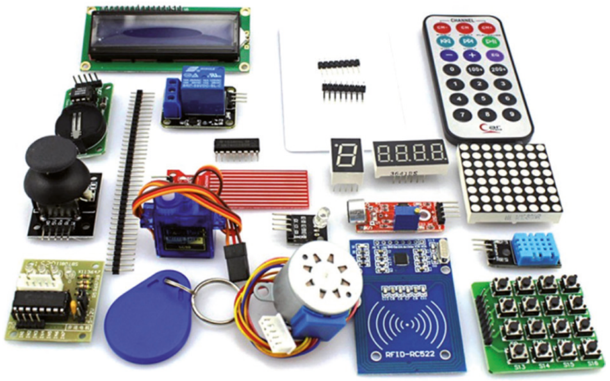


Figure 1.8: Close-up picture of sensors, actuators, and displays.

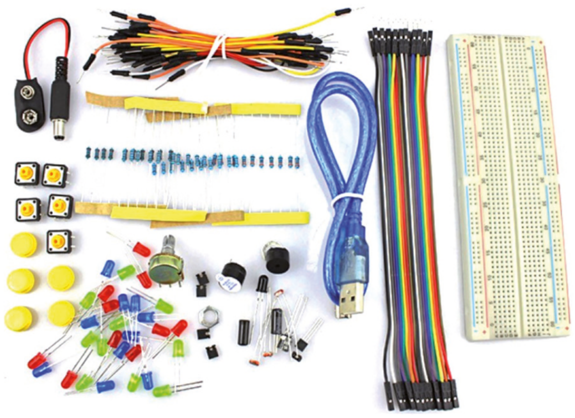


Figure 1.9: Close-up picture of the LEDs, resistors, buttons, wires, etc.

Chapter 2 • Arduino Uno R4 Program Development

2.1 Overview

The Arduino code is written in the C++ language, which is currently one of the most popular languages used to program microcontrollers. Arduino can be programmed using either the **Desktop Arduino IDE** or the **Arduino Web Editor**. The IDE requires the software to be installed on your PC. This is an Integrated Development Environment (IDE) that consists of an editor, compiler, debugger, and tools to upload the compiled code to the processor on the development board. An Arduino program is called a **sketch**, which is compiled into machine code and uploaded to the target processor. Arduino Web Editor enables the programmer to develop, compile, and upload programs using an online browser with the advantage that the online tool is always up-to-date and includes the latest libraries and features. In this book, you will be using the IDE. Interested readers can search for **Arduino Web Editor** in Google and create an account to sign in and use it.

The Arduino IDE has been developed over a decade and there are several versions of it. The latest stable version is **2.1** released in March 2022. Version 1.8.19 has been popular for many years and is still used by many programmers. New version 2.1.0 is the recommended version since it is faster and easier to use than version 1.8.19. The author has used version 2.1.0 in all the projects in this book. Readers may prefer to use the same version in their projects.

In this chapter, you will learn how to install version 2.1.0 of the IDE, which was the latest version at the time of authoring this book. Simple software-only programs are given in this chapter to review the principles of programming using the Arduino IDE. In the next chapters, you will be using the newly released **Arduino Uno R4 Minima/WiFi** development boards together with the supplied components of the bundle in many real-time project applications.

Further information on Arduino IDE, Web Editor, and related tools can be obtained from the links given in Table 2.1.

Ref	**Link**
Arduino IDE (Desktop)	https://www.arduino.cc/en/Main/Software
Arduino IDE (Cloud)	https://create.arduino.cc/editor
Cloud IDE Getting Started	https://docs.arduino.cc/cloud/web-editor/tutorials/getting-started/getting-started-web-editor
Arduino Project Hub	https://create.arduino.cc/projecthub?by=part&part_id=11332&sort=trending
Library Reference	https://github.com/arduino-libraries/
Online Store	https://store.arduino.cc/

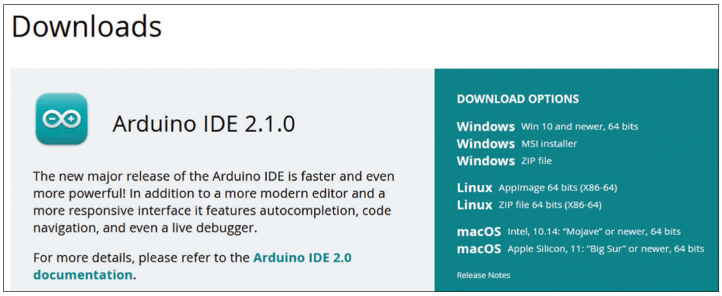
Table 2.1: Links to Arduino IDE, Web Editor, and related tools (Product Reference Manual, SKU: ABX00080).

2.2 Installing the Arduino IDE 2.1.0

The latest version of the Arduino IDE can be installed from the following website:

https://www.arduino.cc/en/software

Select your processor from the **DOWNLOAD OPTIONS** at the right (Figure 2.1). Click **JUST DOWNLOAD** (unless you want to support by contributing). The author installed it on a Windows 10 laptop and at the time of drafting this book the latest version file name was: **Arduino-ide_2.1.0-Windows_64bit.exe**.



Downloads

Arduino IDE 2.1.0

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the **Arduino IDE 2.0 documentation**.

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file

Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)

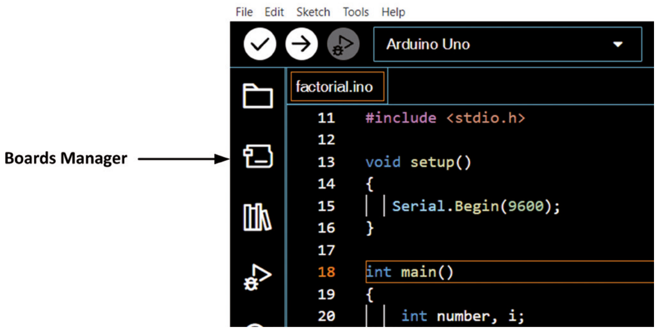
macOS Intel, 10.14: "Mojave" or newer, 64 bits
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

Release Notes

Figure 2.1: Select your processor to install.

You now have to install the Board Package for your Arduino Uno R4 Minima. The steps are:

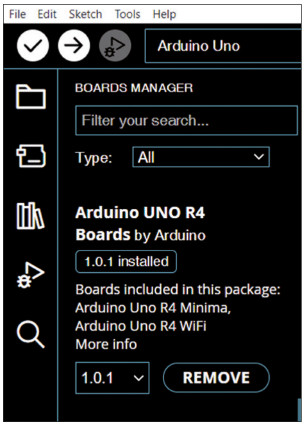
- Start the Arduino IDE.
- Click to Open the **Boards Manager** at the top left of the screen (Figure 2.2).



```
#include <stdio.h>

void setup()
{
  Serial.Begin(9600);
}

int main()
{
  int number, i;
```

Figure 2.2: Open the Boards manager.

- Search for ARDUINO UNO R4 (Figure 2.3) and click INSTALL to install it. At the time of drafting this book, the version was: 1.0.1.



Arduino UNO R4
Boards by Arduino
Boards included in this package:
Arduino Uno R4 Minima,
Arduino Uno R4 WiFi
More info

Figure 2.3: Click INSTALL to install the Uno R4 boards.

- Click **Boards Manager** to close the left window.
- You should be able to select the Arduino Uno R4 board from the board selector at the top left of the screen (Figure 2.4). Connect your Arduino Uno R4 Minima (or WiFi) to your PC via a USB-C cable.
- Click **Tools** → **Por**t and select the serial port connected to your Uno R4.

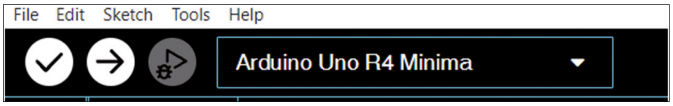


Figure 2.4: Select: Arduino Uno R4 Minima.

You are now ready to develop programs and upload them to your Arduino Uno R4 Minima/ WiFi processor on your development board.

Before looking at some example programs, it is worthwhile to learn some of the *commonly used* menu options offered by the IDE.

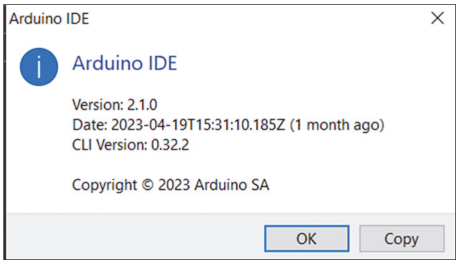
File: with this menu option you can open existing programs, save programs, open example programs, and set the IDE working environment (e.g., Preferences…).

Edit: with this menu option you can cut, paste, select, go to a specified line, change indentation and font size, and find text in a file.

Sketch: with this menu option you can compile your program, upload the compiled program to the target processor, include libraries, add files and some other options that you will not be using.

Tools: with this menu option you can manage libraries, configure the serial monitor and serial plotter, select and configure the development board that you will be using, and burn a new bootloader.

Help: this menu option displays various help items on selected topics. Additionally, it displays the version number of the currently used IDE (Figure 2.5). e.g., Version: 2.1.0


Arduino IDE

Arduino IDE

Version: 2.1.0
Date: 2023-04-19T15:31:10.185Z (1 month ago)
CLI Version: 0.32.2

Copyright © 2023 Arduino SA

Figure 2.5: Displaying the IDE version number.

2.3 Software-only programs

In this chapter, simple software-only programs are given to review the basic principles of programming in C language using the Arduino IDE. The human interface (e.g. display and keyboard) to these programs is the **Serial Monitor** of the IDE. The aim here has been to review the C language programming concepts by developing simple programs, and then uploading and running them on your development board. Readers who have a good working knowledge of the C language and who are familiar with the Serial Monitor can skip this chapter.

2.3.1 Example 1: Sum of integer numbers

Write a program to read an integer number N from the keyboard and then calculate and display the sum of all the integer numbers from 1 to N.

Solution 1

Figure 2.6 shows the program listing (Program: **sumN**). Comments are used at the beginning of the program to describe the function of the program. Also, the names of the author and the program and the date of development of the program are all listed here. It is strongly recommended by the author to include comments in your programs to make them easy to follow and also easy to modify in the future.

The **setup()** function is executed only once at the beginning of the program. Inside this function, the Serial Monitor is configured to run at 9600 baud (you may choose a different baud rate if you wish).

The main program runs inside the function **loop()**. Here, variables **i**, **N**, and **Sum** are declared as integers and **Sum** is cleared to 0. The program prompts the user to enter **N** which is read using the built-in function **parseInt()**. The program checks if data is available before reading from the keyboard. Then, a **for** loop is formed where the sum of all the integer numbers from 1 to N are calculated and stored in variable **Sum**. The sum is finally displayed as an integer using a **println()** function. Notice that the **println()** function prints a carriage return and line feed after displaying the data. The program is stopped by using a **while()** statement at the end, otherwise the **loop()** function will repeat forever.

The steps to test the program are as follows:

- Connect your Arduino Uno R4 development board to the PC and configure the serial link.

- Type your program as in Figure 2.6 (or load from the Elektor website of the book) and then save it with a suitable name, e.g., **sumN**.

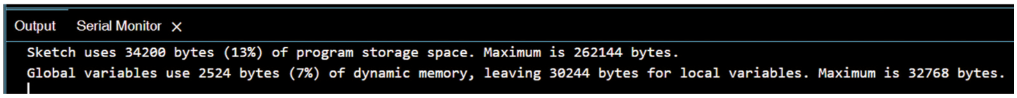
- Click **Sketch → Verify/Compile** to compile the program. The status of the compilation will be displayed in the bottom panel as **Compiling sketch....** If there are any errors, you should go back to your program to correct the errors. If there are no errors, then the bottom panel will display as shown in Figure 2.7 (Click **Output** at the bottom panel to see this message).

- Click **Sketch → Upload** to upload the correctly compiled code to the processor on your development board. You should see the message **Done uploading** at the bottom part of the screen.

- Make sure the Baud rate is set to 9600. Click **Serial Monitor** at the top of the bottom panel. (Figure 2.8). If Serial Monitor is not available, click the **Serial Monitor** icon at the top right corner of the display.

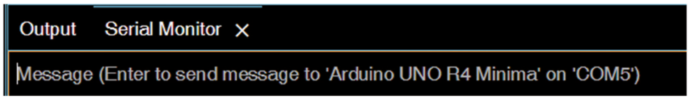
```
//----------------------------------------------------------------------
//           SUM OF INTEGER NUMBERS FROM 1 to N
//           ==================================
//
// This program calculates and displays the sum of integer numbers
//
// Author: Dogan Ibrahim
// File  : sumN
// Date  : June, 2023
//----------------------------------------------------------------------
void setup()
{
  Serial.begin(9600);
  delay(5000);
}

void loop()
{
  int i, N, Sum = 0;
  Serial.print("How many numbers are there ? ");
  while(Serial.available() <= 0);
  N = Serial.parseInt();
  Serial.print(N);

  for (i = 1; i <= N; i++)                     // Do for 1 to N
  {
    Sum = Sum + i;                             // Calculate the sum
  }

  Serial.println();                            // Print new line
  Serial.print("Sum of numbers from 1 to ");  // Display "Sum = "
  Serial.print(N);                             // Display N
  Serial.print(" are: ");                      // Display " are: "
  Serial.println(Sum);                         // Display the sum

  while(1);                                    // Stop the program
}
```

Figure 2.6: Program: ***sumN****.*



```
Sketch uses 34200 bytes (13%) of program storage space. Maximum is 262144 bytes.
Global variables use 2524 bytes (7%) of dynamic memory, leaving 30244 bytes for local variables. Maximum is 32768 bytes.
```

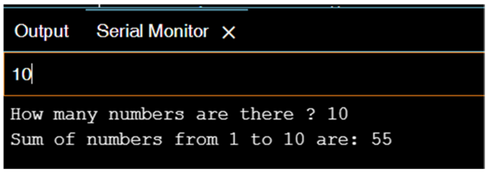
Figure 2.7: Successful compilation.



Message (Enter to send message to 'Arduino UNO R4 Minima' on 'COM5')

Figure 2.8: Serial Monitor window.

- You should see the text **How many numbers are there ?** displayed at the Serial Monitor window. For example, enter 10 where it says**: Message (Enter to send message to 'Arduino UNO R4' Minima...)** and press the **Enter** key on your PC. You should see 55 displayed which is the sum of integer numbers from 1 to 10 as shown in Figure 2.9.



```
10
How many numbers are there ? 10
Sum of numbers from 1 to 10 are: 55
```

Figure 2.9: Displaying the sum of numbers from 1 to 10.

You should follow the steps given in this program in order to test the other programs given in the following sections of this chapter.

2.3.2 Example 2: Table of squares

Write a program to tabulate the squares of integer numbers from 1 to 10.

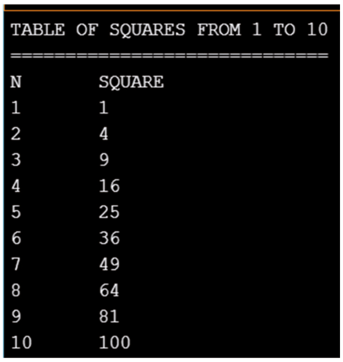
Solution 2

Figure 2.10 shows the program listing (Program: **squares**). The Serial Monitor is initialized as in the previous example. A **for** loop is set up in the main program loop, which tabulates the squares of numbers from 1 to 10. The display items are separated with a tab (i.e. "\t").

```
//--------------------------------------------------------------------
//                TABLE OF SQUARES
//                =================
//
// This program displays table of squares of integers from 1 to 10
//
// Author: Dogan Ibrahim
// File  : squares
// Date  : June, 2023
//--------------------------------------------------------------------
void setup()
{
  Serial.begin(9600);
  delay(5000);
}

void loop()
{
  int i, N;

  Serial.println("TABLE OF SQUARES FROM 1 TO 10");
  Serial.println("=============================");
  Serial.println("N\tSQUARE");

  for (i = 1; i <= 10; i++)                   // Do for 1 to N
  {
    N = i * i;                                // Calculate the square
    Serial.print(i);
    Serial.print("\t");
    Serial.println(N);
  }

  while(1);                                   // Stop the program
}
```

Figure 2.10: Program: ***squares****.*

Figure 2.11 shows the output from the program, displayed on the Serial Monitor.



```
TABLE OF SQUARES FROM 1 TO 10
=============================
N       SQUARE
1       1
2       4
3       9
4       16
5       25
6       36
7       49
8       64
9       81
10      100
```

Figure 2.11: Output from the program.

2.3.3 Example 3: Volume of a cylinder

Write a function to return the volume of a cylinder where the radius and height should be passed as arguments to the function. Use the function in a program to calculate and display the area of the cylinder whose radius is 10 cm and height 15 cm.

Solution 3

The volume of a cylinder whose radius and height are *r* and *h* respectively is given by:

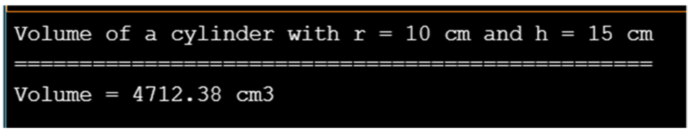
$$\text{Area} = \pi r^2 h$$

Figure 2.12 shows the program listing (Program: **cylarea**). Function **area** receives the radius and height of the cylinder as floating point numbers and returns the volume to the main program which then displays the volume on the Serial Monitor.

```
//---------------------------------------------------------------------
//                VOLUME OF A CYLINDER
//                ====================
//
// This program calculates and displays the volume of a cylinder
//
// Author: Dogan Ibrahim
// File  : cylarea
// Date  : June, 2023
//---------------------------------------------------------------------
#define pi 3.14159
float r = 10.0;
float h = 15.0;

void setup()
{
  Serial.begin(9600);
  delay(5000);
}

//
// Function to calculate the volume
//
float volume(float radius, float height)
{
  float vol;
  vol = pi * radius * radius * height;
  return vol;
}

void loop()
{
  float cylinder;

  Serial.println("Volume of a cylinder with r = 10 cm and h = 15 cm");
  Serial.println("=================================================");

  cylinder = volume(r, h);
  Serial.print("Volume = ");
  Serial.print(cylinder);
  Serial.println(" cm3");

  while(1);                                    // Stop the program
}
```

Figure 2.12: Program: **cylarea***.*

Figure 2.13 shows the output from the program where the radius and height are set to 10 cm and 15 cm, respectively at the beginning of the program.



```
Volume of a cylinder with r = 10 cm and h = 15 cm
=================================================
Volume = 4712.38 cm3
```

Figure 2.13: Output from the program.

2.3.4 Example 4: Centigrade to Fahrenheit

Write a program to receive the temperature as Centigrade, convert it to Fahrenheit, and then display it on the Serial Monitor. You should read the temperature from the keyboard.

Solution 4

Given the temperature in degrees C, it can be converted into degrees F using the following formula:

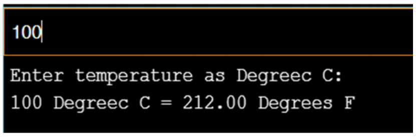
$$F = 1.8 \times C + 32$$

Figure 2.14 shows the program listing (Program: **CtoF**). Function **ToF** receives degrees C as its argument, converts it in degrees F and returns to the main program. The temperature in degrees Centigrade is read from the keyboard.

```
//----------------------------------------------------------------------
//                CELSIUS TO FAHRENHEIT
//                =====================
//
// This program converts Celsius to Fahrenheit
//
// Author: Dogan Ibrahim
// File  : CtoF
// Date  : June, 2023
//----------------------------------------------------------------------
void setup()
{
  Serial.begin(9600);
  delay(5000);
}

//
// Function to convert Degrees C to Degrees F
//
float ToF(float C)
{
  return (1.8 * C + 32);
}

void loop()
{
  float F;
  int C;

  Serial.println("Enter temperature as Degreec C: ");
  while(Serial.available() <= 0);
  C = Serial.parseInt();                        // Read Degrees C

  F = ToF(C);                                   // COnvert to F
  Serial.print(C);
  Serial.print(" Degreec C = ");
  Serial.print(F);
  Serial.println(" Degrees F");

  while(1);                                    // Stop the program
}
```

Figure 2.14: Program: ***CtoF****.*

Figure 2.15 shows the output from the program where 100 degrees centigrade is converted into Fahrenheit and displayed on the Serial Monitor.



```
100
Enter temperature as Degreec C:
100 Degreec C = 212.00 Degrees F
```

Figure 2.15: Output from the program.

2.3.5 Example 5: Times table

Write a program to read an integer number and then tabulate the times table from 1 to 12 for the given number.

Solution 5

Figure 2.16 shows the program listing (Program: **times**). An integer number is read from the keyboard and stored in variable **N**. Then a **for** loop is set up that runs from 1 to 12. Number **N** is multiplied by 1 to 12 inside this loop and is then displayed on the Serial Monitor.

```
//----------------------------------------------------------------------
//                        TIMES TABLE
//                        ===========
//
// This program reads an integer number and then tabulates the time table
//
// Author: Dogan Ibrahim
// File  : times
// Date  : June, 2023
//----------------------------------------------------------------------
void setup()
{
  Serial.begin(9600);
  delay(5000);
}

void loop()
{
  int N, i;
```