

Home / Hardware / UNO R4 Minima / [Arduino UNO R4 Minima Cheat Sheet](#)

## Arduino UNO R4 Minima Cheat Sheet

Learn how to set up the UNO R4 Minima, the fourth revision of our most popular and important development board.

Author · Hannes Siebeneicher

Last revision · 02/16/2024

The Arduino® UNO R4 Minima is a development board with the classic UNO form factor, based on the [RA4M1](#) microcontroller made by [Renesas](#). It now comes with 32 kB of RAM memory, a clock speed of 48 MHz, and a USB-C® port.

This is the first UNO board that uses a 32-bit architecture, being previously based on an 8-bit AVR architecture.

This article is a technical reference to your board, introducing the various components of the board, as well as resources to get started.

## Datasheet

The full datasheet is available as a downloadable PDF from the link below:

[Download the UNO R4 Minima datasheet](#)

## Power Supply

To power the UNO R4 Minima you may either use a USB-C® cable or the VIN pin.

The board can be powered via the VIN pin, supporting a range between 6-24 V. The VIN pin is also connected to the DC-jack (barrel plug connector).

When powered via the VIN pin, you are using the onboard regulator to bring down the voltage to 5V, which means that the 5 V pin can provide up to 1.2 A. Keep in mind that this voltage regulator also powers the rest of the circuit board, including the MCU, LEDs among other components.

**i** External devices with a high current draw (e.g. servo motors) should never be powered via the 5 V pin. It is mainly intended for devices drawing lower current such as sensor modules.

If you're using the USB-C® connector you must power it with 5 V.

When powered via USB, you are bypassing the onboard voltage regulator completely. In this case, the 5 V pin can provide up to 2 A without damaging the board.

## Board Package

The UNO R4 Minima is based on the [UNO R4 Board Package](#).

## Installation

The UNO R4 Minima can be programmed either via the Arduino IDE, Arduino Web Editor, or Arduino CLI.

### Arduino IDE

To use the board in the Arduino IDE, you need to install the latest version of the **Arduino UNO R4 Boards** package from the boards manager.

[Help](#)

Arduino Web Editor

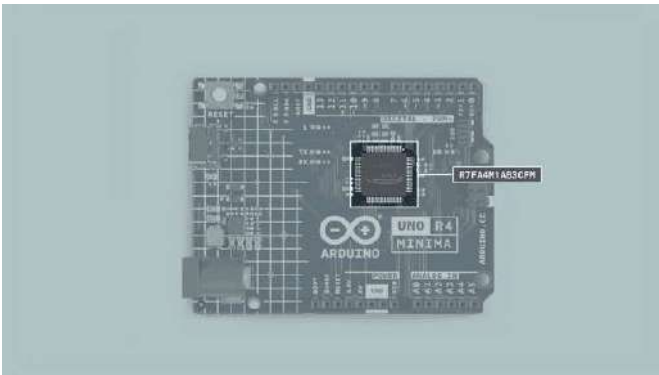
The Web Editor is an online IDE that includes all official boards, no need for installing the Board Package. You will need the Create Plugin installed on your computer to use the Web Editor.

Read more in the [Getting Started with the Web Editor](#) guide.

Renesas RA4M1

The UNO R4 Minima features the powerful and very robust Renesas microcontroller also found on the UNO R4 WiFi. Renesas microcontrollers are known for their high performance and robustness, including their built-in peripheral set.

These peripherals include analog-to-digital converters, timers, pulse width modulation (PWM) units, communication interfaces (such as UART, SPI, and I2C), and more.



Microcontroller on the UNO R4 Minima

← GO BACK

Hardware

UNO R4 Minima

Tutorials

Arduino UNO R4 Minima ADC Resolution

Arduino UNO R4 Minima CAN Bus

Arduino UNO R4 Minima Cheat Sheet

Arduino UNO R4 Minima Digital-to-Analog Converter (DAC)

Debugging the Arduino UNO R4 Minima

Arduino UNO R4 Minima EEPROM

Getting Started with Arduino UNO R4 Minima

Arduino UNO R4 Minima OPAMP

Arduino UNO R4 Minima Real-Time Clock

Arduino UNO R4 Shield Compatibility

Arduino UNO R4 Shield Guide

Memory

The board features

- 32 kB of SRAM
- 256 kB flash
- 8 kB data (EEPROM).

Pins

The UNO R4 Minima gives you access to many different pins and many of them have special features that will be accounted for in the upcoming sections of this article. Keep reading to learn what you can do with them.

This is a full table of all the IO pins on the UNO R4 Minima:

Pin	Type	Function
D0	Digital	UART Receive
D1	Digital	UART Transmit
D2	Digital	GPIO pin, Interrupt
D3	Digital	GPIO pin, Interrupt, PWM
D4	Digital	GPIO pin
D5	Digital	GPIO pin, PWM
D6	Digital	GPIO pin, PWM
D7	Digital	GPIO pin
D8	Digital	GPIO pin

ON THIS PAGE

- Datasheet
- Power Supply
- Board Package
- Installation +
- Renesas RA4M1 +
- Pins -
- Analog Pins
- OPAMP Pins
- PWM
- Digital Pins
- LED
- DAC
- RTC
- EEPROM
- SPI
- I2C
- USB Serial & UART +
- USB HID
- SWD Connector
- CAN Module
- Bootloader

D9	Digital	GPIO pin, PWM
D10	Digital	SPI (CS), GPIO pin, PWM
D11	Digital	SPI (CIPO), GPIO pin, PWM
D12	Digital	SPI (COPI), GPIO pin
D13	Digital	SPI (SCK), GPIO pin, Built-in LED
A0	Analog	Analog In, DAC
A1	Analog	Analog In, OPAMP +
A2	Analog	Analog In, OPAMP -
A3	Analog	Analog In, OPAMP OUT
A4	Analog	Analog In, SDA*
A5	Analog	Analog In, SCL*

 A4 and A5 pins are both connected to the same I2C bus.

Analog Pins

The UNO R4 Minima has six analog input pins (A0-A5) that can be read by using the `analogRead()` function.

Pin	Type	Function
A0	Analog	Analog In, DAC
A1	Analog	Analog In, OPAMP +
A2	Analog	Analog In, OPAMP -
A3	Analog	Analog In, OPAMP OUT
A4	Analog	Analog In, SDA*
A5	Analog	Analog In, SCL*

 A4 and A5 pins are both connected to the same I2C bus.

COPY

```
1 value = analogRead(pin);
```

The default reference voltage of these pins is 5 V, but this can be changed as follows:

```
analogReference(AR_DEFAULT) (Default reference of 5 V)
analogReference(AR_INTERNAL) (Built in reference of 1.5 V.)
```

The default resolution is set to 10-bit, but can be updated to 12 and 14-bit resolutions. To do so, use the following method in the `setup()` of your sketch.

```
analogReadResolution(10) (default)
analogReadResolution(12)
analogReadResolution(14)
```

To learn more about the ADC capabilities of the UNO R4 Minima, check out the [ADC-Resolution Guide](#).

OPAMP Pins

The **RA4M1** has an internal OPAMP that is exposed on the UNO R4 Minima as follows:

Pin	OPAMP
A1	OPAMP +

A2	OPAMP -
A3	OPAMP OUT

PWM

PWM (Pulse Width Modulation) capability allows a digital pin to emulate analog output by flickering on and off very fast letting you, among other things, dim LEDs connected to digital pins.

The UNO R4 Minima supports PWM on pins marked with ~ on the headers. Officially supported pins are:

Pin	RA4M1	Timer
D3	P104	GTIOC1B
D5	P102	GTIOC2B
D6	P106	GTIOC0B
D9	P303	GTIOC7B
D10	P112	GTIOC3B
D11	P109	GTIOC1A

You may use them as analog output pins with the function:

COPY

```
1 analogWrite(pin, value);
```

By default, the resolution is 8 bit (0-255), You can use `analogWriteResolution()` to change this, supporting up to 12 bit (0-4096) resolution.

COPY

```
1 analogWriteResolution(resolution);
```

Digital Pins

The UNO R4 Minima features a total of digital 14 pins. Though some of them serve another purpose and shouldn't be used for GPIO if you have other pins available.

Pin	Type	Function
D0	Digital	UART Receive
D1	Digital	UART Transmit
D2	Digital	GPIO pin, Interrupt
D3	Digital	GPIO pin, Interrupt, PWM
D4	Digital	GPIO pin
D5	Digital	GPIO pin, PWM
D6	Digital	GPIO pin, PWM
D7	Digital	GPIO pin
D8	Digital	GPIO pin
D9	Digital	GPIO pin, PWM
D10	Digital	SPI (CS), GPIO pin, PWM
D11	Digital	SPI (CIPO), GPIO pin, PWM
D12	Digital	SPI (COPI), GPIO pin
D13	Digital	SPI (SCK), GPIO pin, Built-in LED

In addition, analog pins A0-A5 can also be used as digital pins. Note that **A4/A5** are reserved for the I2C bus.

The reference voltage of all digital pins is 5 V.

The UNO R4 Minima has a total of four LEDs, three of which are programmable:

- ON** - power LED, cannot be programmed.
- LED\_BUILTIN** - classic "built-in LED", attached to pin 13.
- LED\_RX** - LED labelled "RX" on the board.
- LED\_TX** - LED labelled "TX" on the board.

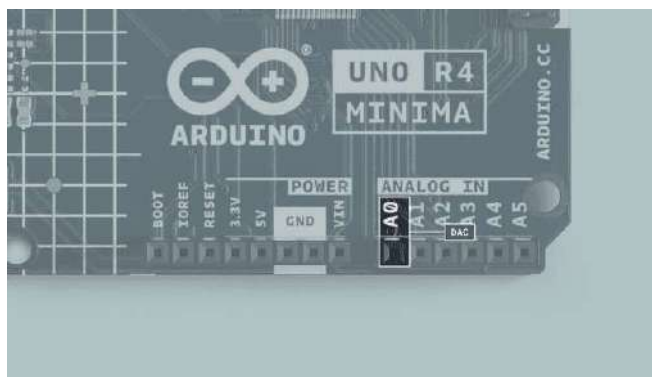
To control these, define them as outputs and write the desired state. The below example blinks each LED every second.

COPY

```

1 void setup(){
2   //define pins as output
3   pinMode(LED_BUILTIN, OUTPUT);
4   pinMode(LED_RX, OUTPUT);
5   pinMode(LED_TX, OUTPUT);
6 }
7
8 void loop(){
9   //turn on all LEDs
10  digitalWrite(LED_BUILTIN, HIGH);
11  digitalWrite(LED_RX, LOW);
12  digitalWrite(LED_TX, LOW);
13  delay(1000);
14
15  //turn off all LEDs
16  digitalWrite(LED_BUILTIN, LOW);
17  digitalWrite(LED_RX, HIGH);
18  digitalWrite(LED_TX, HIGH);
19  delay(1000);
20 }
```

## DAC



DAC Pin

The UNO R4 Minima has a DAC with up to 12-bit resolution, that can act as a genuine analog output pin which means it's even more capable than PWM pins.

COPY

```
1 analogWrite(pin, value);
```

This DAC pin has a default write resolution of 8 bits. This means that values that are written to the pin should be between 0-255.

However you may change this write resolution if you need to, to up to 12 bits, and in this case, the values you write to the pin should be between 0-4096.

COPY

```
1 analogWriteResolution(12);
```

## RTC

A real-time clock (RTC) is used to measure the time and is useful in any time-tracking application.

Below is a minimal example that shows how to obtain the date and time from the RTC:

[COPY](#)

```
1 #include "RTC.h"
2
3 void setup() {
4   Serial.begin(9600);
5
6   RTC.begin();
7   RTCTime mytime(30, Month::JUNE, 2023, 13, 37, 00, DayOfWeek::WE
8
9   RTC.setTime(mytime);
10 }
11
12 void loop() {
13   RTCTime currenttime;
14
15   // Get current time from RTC
16   RTC.getTime(currenttime);
17
18   // Print out date (DD/MM/YYYY)
19   Serial.print(currenttime.getDayOfMonth());
20   Serial.print("/");
21   Serial.print(Month2int(currenttime.getMonth()));
22   Serial.print("/");
23   Serial.print(currenttime.getYear());
24   Serial.print(" - ");
25
26   // Print time (HH/MM/SS)
27   Serial.print(currenttime.getHour());
28   Serial.print(":");
29   Serial.print(currenttime.getMinutes());
```

To learn more about the RTC capabilities of the UNO R4 Minima, check out the [RTC Guide](#).

## EEPROM

EEPROM, also referred to as 'data' memory, is a type of memory that can retain data even after the board has been powered off. The Arduino Uno R4 Minima has 8 kB EEPROM.

[COPY](#)

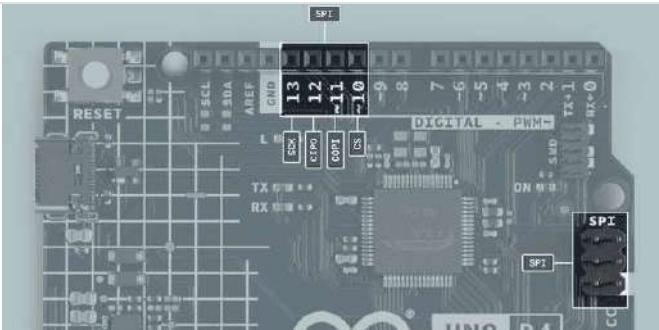
```
1 EEPROM.write(address, val);
2 EEPROM.read(address);
```

It has a limited amount of write cycles, meaning that it is best suited for read-only applications. Make sure to never use `write()` inside `void loop()` because you may use all write cycles for the chip.

Read more in the [Guide to EEPROM](#).

If you want to read more about the EEPROM check out [this article about Arduino UNO R4 Minima EEPROM](#).

## SPI



SPI Pins

The UNO R4 Minima features a Serial Peripheral Interface (SPI) bus. The bus (connector), 'SPI' uses the following pins:

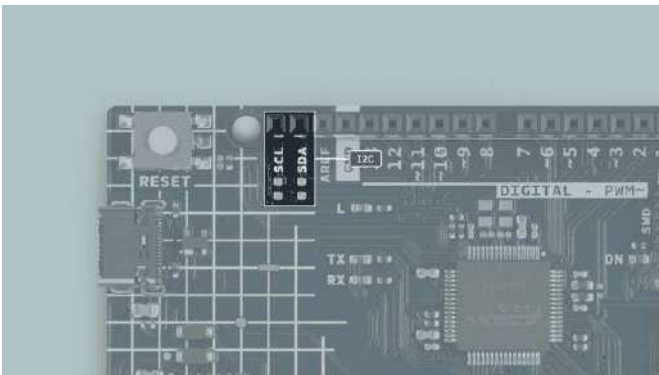
Pin	Type
D13	SCK
D12	MISO
D11	MOSI
D10	CS

The following example shows how to use SPI:

COPY

```
1 #include <SPI.h>
2
3 const int CS = 10;
4
5
6 void setup() {
7   pinMode(CS, OUTPUT);
8
9   SPI.begin();
10
11   digitalWrite(CS, LOW);
12
13   SPI.transfer(0x00);
14
15   digitalWrite(CS, HIGH);
16 }
17
18 void loop() {
19 }
```

I2C



I2C Pins

the technical limit of I2C devices on a single line is 128. Practically, you're never gonna reach 128 devices before other limitations kick in.

The UNO R4 Minima has one I2C bus which is marked with SCL and SDA. They are shared with A4 (SDA) and A5 (SCL) which owners of previous UNOs are familiar with. The pullups are not mounted on the PCB but there are footprints to do so if needed.

There are a couple of advantages to not mounting the pullup resistors from the factory:

As the pins used for I2C are directly connected to A4 and A5 respectively, they are also able to be used as digital input/output, and analog input pins. Mounting I2C pullup resistors to these pins would limit the functionality to only I2C, as they would be locally **HIGH** by default.

By choosing to mount different resistances, you are able to select if you want to operate a 3.3 V or a 5 V I2C device with these pins.

The pins used for I2C on the UNO R4 Minima are the following:

SDA - D18 or A4

SCL - D19 or A5

To connect I2C devices you will need to include the [Wire](#) library at the top of your sketch.

COPY

```
1 #include <Wire.h>
```

Inside `void setup()` you need to initialize the library and initialize the I2C port you want to use.

COPY

```
1 Wire.begin() //SDA & SDL
```

And to write something to a device connected via I2C, we can use the following commands:

COPY

```
1 Wire.beginTransmission(1); //begin transmit to device 1
2 Wire.write(byte(0x00)); //send instruction byte
3 Wire.write(val); //send a value
4 Wire.endTransmission(); //stop transmit
```

Learn more about the I2C protocol in our [I2C Protocol Guide](#)

## USB Serial & UART

The UNO R4 Minima board features two separate hardware serial ports.

One port is exposed via USB-C®, and

One is exposed via RX/TX pins.

This is one of the few things that are distinctly different from UNO R3 to UNO R4, as the UNO R3 only features one hardware serial port, that is connected to **both** the USB port and the RX/TX pins on the board.

### Native USB

Sending serial data to your computer is done using the standard `Serial` object.

COPY

```
1 Serial.begin(9600);
2 Serial.print("hello world");
```

To send and receive data through UART, we will first need to set the baud rate inside

`void setup()` .



The pins used for UART on the UNO R4 Minima are the following:

Pin	Function
D0	RX (Receive)
D1	TX (Transmit)

To send and receive data through UART, we will first need to set the baud rate inside `void setup()`. Note that when using the UART (RX/TX pins), we use the `Serial1` object.

COPY

```
1 Serial1.begin(9600);
```

To read incoming data, we can use a while loop() to read each individual character and add it to a string.

COPY

```
1 while(Serial1.available()){
2   delay(2);
3   char c = Serial1.read();
4   incoming += c;
5 }
```

And to write something, we can use the following command:

COPY

```
1 Serial1.write("Hello world!");
```

### Serial Event

The `SerialEvent()` method is supported on older revisions of the UNO board, but **not** on the UNO R4 boards (or any other newer Arduino boards).  
  
However, as this method is only used to detect serial data and execute a function, you can also use `Serial.available()` to detect when new data is available:

COPY

```
1 if(Serial.available() > 0) {
2   //code goes here
3 }
```

### SerialUSB

The UNO R4 Minima has an extended set of Serial methods:

- `Serial.baud()` - Returns the baud rate (**int**) currently used.
- `Serial.stopbits()` - Returns the number of stop bits (**int**) used in the communication.
- `Serial.paritytype()` - Returns the type of parity (**int**) used in the communication.
- `Serial.numbits()` - Returns the number of data bits (**int**) used in the communication.
- `Serial.dtr()` - Returns the status of the Data Terminal Ready (DTR) signal (**bool**) and also sets the `ignore_dtr` flag to true if the DTR signal is actively used.
- `Serial.rts()` - Returns the status of the Request to Send (RTS) signal (**bool**).

Supported links:

[SerialUSB.h](#) (Github).

This board can act as an HID (keyboard/mouse) and send keystrokes or coordinates to your computer via native USB.

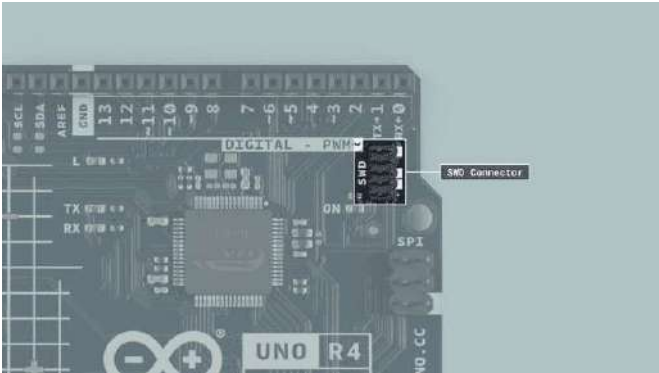
COPY

```
1 keyboard.press('W');
2 mouse.move(x,y);
```

This support is enabled by the [keyboard](#) and [mouse](#) libraries that you can install from the library manager in the IDE.

To learn more about the HID capabilities of the UNO R4 Minima, check out the [HID Guide](#).

## SWD Connector



SWD Connector

On the UNO R4 Minima, there is a debugging option available using the SWD connector pins, giving advanced debug functionalities for more advanced users.

## CAN Module

The UNO R4 Minima's RA4M1 has a built-in CAN module that complies with the CAN 2.0A/CAN 2.0B standard.

The pins CANRX and CANTX can be connected to a CAN transceiver, such as a MCP2551 or TJA1050 ICs.

Pin	Function
D4	CANTX
D5	CANRX

The built-in **Arduino\_CAN** library is used to communicate with other CAN devices.

COPY

```
1 //set CAN bit rate and init library at
2 //choose from BR_125k, BR_250k, BR_500k, BR_1000k
3 CAN.begin(CanBitRate: :BR_250k);
```


Construct a CAN message and send it:

COPY

```
1 uint8_t const msg_data[] = {0xCA, 0xFE, 0, 0, 0, 0, 0, 0};
2 memcpy((void *) (msg_data + 4), &msg_cnt, sizeof(msg_cnt));
3 CanMsg msg(CAN_ID, sizeof(msg_data), msg_data);
4 CAN.write(msg);
```

Read an incoming CAN message.

```
CanMsg const msg = CAN.read(); //read
```

 Please note that without a CAN transceiver it is not possible to communicate with other CAN devices.

## Bootloader

In case you need to flash the bootloader on the UNO R4 Minima, you can follow the steps below:

**Step 1** Install the [UNO R4 Board Package](#) as described in the [Getting Started Guide](#).

**Step 2** Navigate to:  
"C:\Users\YourWindowsUserName\AppData\Local\Arduino15\packages\arduino\hardware\renesas\_uno\1.X\bootloaders\UNO\_R4"

**Step 3** Identify the `dfu_minima.hex`

**Step 4** Install the Renesas flash programmer ([download page](#))

 The Renesas flash programmer is currently only available on Windows.

**Step 5** Flash the bootloader using the Renesas programmer:

Select `dfu_minima.hex` .  
Connect your board.  
Short the BOOT and GND pin found on the UNO R4 Minima.  
Go to the Connect Settings tab.  
Select the COM port in the Tool > select the port shown in the IDE.  
Press start.

 For more details check the `README.md` or the [GitHub page](#).

### Suggested changes

The content on docs.arduino.cc is facilitated through a public [GitHub repository](#). You can read more on how to contribute in the [contribution policy](#).

### Need support?

Help Center  
Ask the Arduino Forum  
Discover Arduino Discord

### License

The Arduino documentation is licensed under the [Creative Commons Attribution-Share Alike 4.0](#) license.