# Arduino UNO R4 minima Introduction

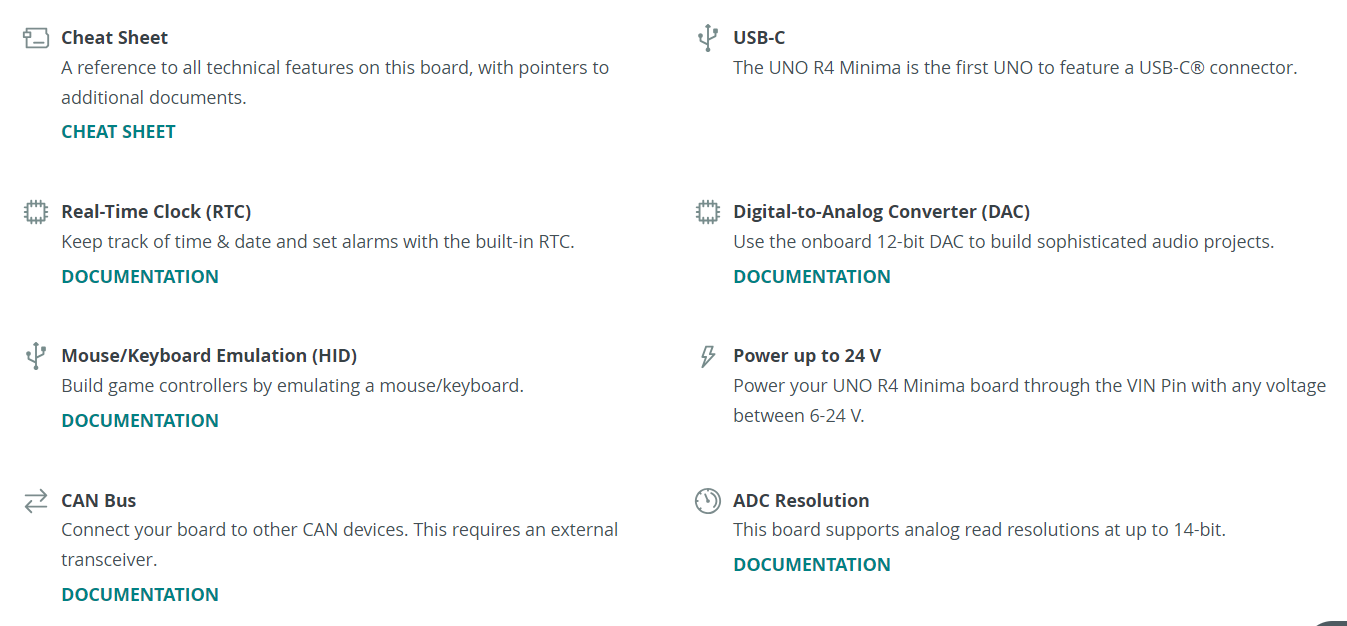
## Description

The Arduino UNO R4 Minima is the first UNO board featuring a 32-bit microcontroller, the RA4M1 from Renesas. It is faster, has more memory and has a number of built-in features such as a DAC, RTC and HID. The UNO R4 Minima is a **5V only** board.

## Features

The UNO R4 Minima features a microcontroller based on the Renesas RA4M1 (Arm® Cortex®-M4) with an operating voltage of 5 V. It has 14 digital I/O, 6 analog inputs with up to 14-bit resolution, a clock speed of 48 MHz, and 32 kB SRAM, 256 kB flash memory & 8 kB of EEPROM.

It features a DAC for audio projects, RTC for accurate time tracking and HID for emulating a keyboard/mouse. The barrel jack plug (VIN) supports voltages at up to 24V, making it compatible with common higher voltage adapters.



### Arduino UNO R4 Minima Cheat Sheet

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:

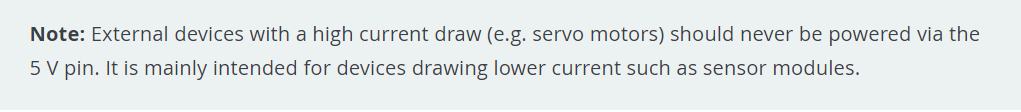
<https://docs.arduino.cc/resources/datasheets/ABX00080-datasheet.pdf>

#### 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.



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.

<https://docs.arduino.cc/tutorials/uno-r4-minima/minima-getting-started>

#### 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.

Read more in the Getting Started with the UNO R4 Minima guide.

<https://docs.arduino.cc/tutorials/uno-r4-minima/minima-getting-started>

#### 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.

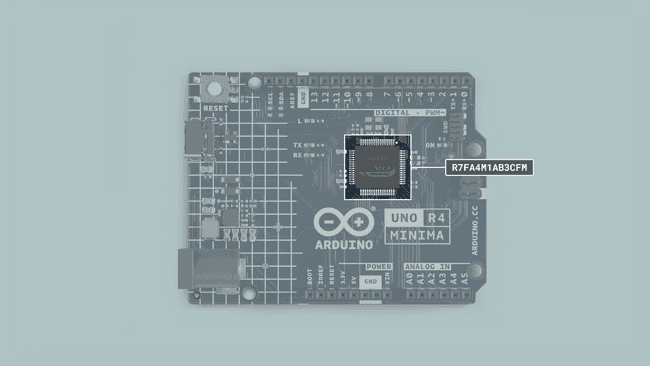
Read more in the Getting Started with the Web Editor guide.

<https://docs.arduino.cc/arduino-cloud/getting-started/getting-started-web-editor>

#### 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

#### Memory

The board features

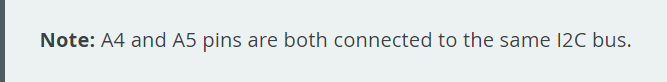
* 32 kB of SRAM
* 256 kB flash
* 8 kB data (EEPROM).

#### 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:

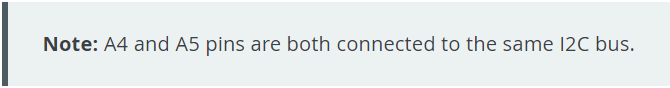
| **Pin** | **Type** | **Function** |
| --- | --- | --- |
| D0 | Digital | UART Receive |
| D1 | Digital | UART Transmit |
| D2 | Digital | GPIO pin, Interrupt |
| D3 | Digital | GPIO pin, Interrupt, PWM |
| D4 | Digital | GPIO pin |
| D5 | Digital | GPIO pin, PWM |
| D6 | Digital | GPIO pin, PWM |
|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



#### 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\* |



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:

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.

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.

#### LED

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.

void setup(){

  //define pins as output

  pinMode(LED\_BUILTIN, OUTPUT);

  pinMode(LED\_RX, OUTPUT);

  pinMode(LED\_TX, OUTPUT);

}

void loop(){

  //turn on all LEDs

  digitalWrite(LED\_BUILTIN, HIGH);

  digitalWrite(LED\_RX, LOW);

  digitalWrite(LED\_TX, LOW);

  delay(1000);

  //turn off all LEDs

  digitalWrite(LED\_BUILTIN, LOW);

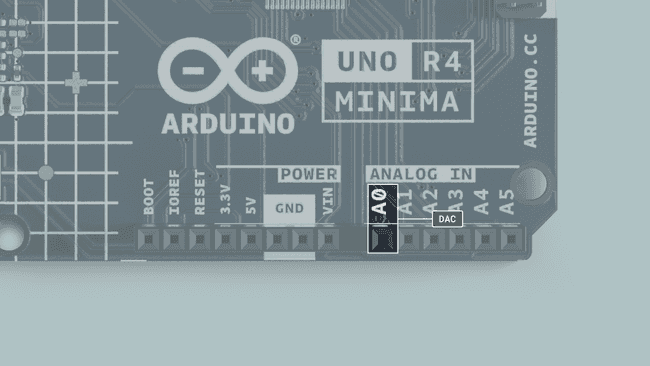
  digitalWrite(LED\_RX, HIGH);

  digitalWrite(LED\_TX, HIGH);

  delay(1000);

}

#### DAC



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.

**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.

**analogWriteResolution(12);**

To learn more about the DAC capabilities of the UNO R4 Minima, check out the DAC guide: <https://docs.arduino.cc/tutorials/uno-r4-minima/dac>

#### 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:

#include "RTC.h"

void setup() {

  Serial.begin(9600);

  RTC.begin();

  RTCTime mytime(30, Month::JUNE, 2023, 13, 37, 00, DayOfWeek::WEDNESDAY, SaveLight::SAVING\_TIME\_ACTIVE);

  RTC.setTime(mytime);

}

void loop() {

  RTCTime currenttime;

 // Get current time from RTC

  RTC.getTime(currenttime);

  // Print out date (DD/MM//YYYY)

  Serial.print(currenttime.getDayOfMonth());

  Serial.print("/");

  Serial.print(Month2int(currenttime.getMonth()));

  Serial.print("/");

  Serial.print(currenttime.getYear());

  Serial.print(" - ");

  // Print time (HH/MM/SS)

  Serial.print(currenttime.getHour());

  Serial.print(":");

  Serial.print(currenttime.getMinutes());

  Serial.print(":");

  Serial.println(currenttime.getSeconds());

  delay(1000);

}

To learn more about the RTC capabilities of the UNO R4 Minima, check out the RTC guide:

<https://docs.arduino.cc/tutorials/uno-r4-minima/rtc>

#### 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.

EEPROM.write(address, val);

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.

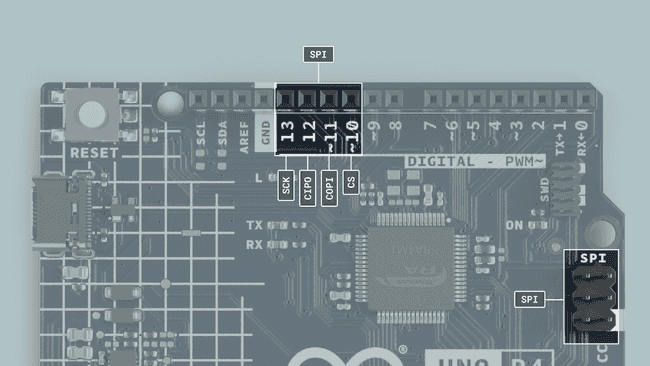
Read more in the Guide to EEPROM.

URL: <https://docs.arduino.cc/learn/programming/eeprom-guide>

If you want to read more about the EEPROM check out this article about Arduino UNO R4 Minima EEPROM.

URL: <https://docs.arduino.cc/tutorials/uno-r4-minima/eeprom>

#### SPI



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

| Pin | Type |
| --- | --- |
| D13 | SCK |
| D12 | CIPO |
| D11 | COPI |
| D10 | CS |

The following example shows how to use SPI:

#include <SPI.h>

const int CS = 10;

void setup() {

  pinMode(CS, OUTPUT);

  SPI.begin();

  digitalWrite(CS, LOW);

  SPI.transfer(0x00);

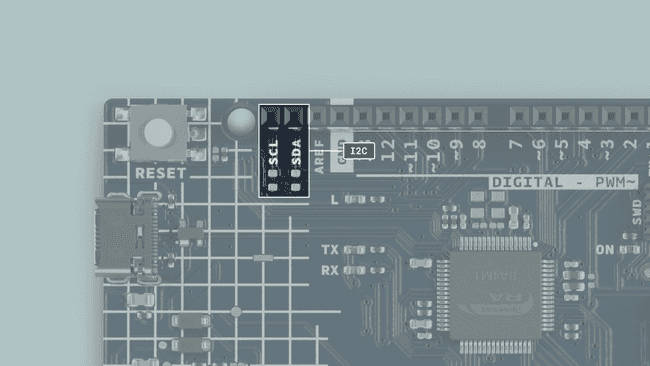
  digitalWrite(CS, HIGH);

}

void loop() {

}

#### I2C



I2C lets you connect multiple I2C-compatible devices in series using only two pins. The controller will send out information through the I2C bus to a 7-bit address, meaning that the technical limit of I2C devices on a single line is 128. Practically, you're never going to 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 pull-ups 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 pull-up 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 pull-up resistors to these pins would limit the functionality to only I2C, as they would be logically **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.

#include <Wire.h>

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

Wire.begin() //SDA & SDL

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

Wire.beginTransmission(1); //begin transmit to device 1

Wire.write(byte(0x00)); //send instruction byte

Wire.write(val); //send a value

Wire.endTransmission(); //stop transmit

Learn more about the I2C protocol in our I2C Protocol Guide:

<https://docs.arduino.cc/learn/communication/wire>

#### 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.

Serial.begin(9600);

Serial.print("hello world");

To send and receive data through UART, we will first need to set the baud rate inside void setup().

##### UART

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.

Serial1.begin(9600);

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

while(Serial1.available()){

    delay(2);

    char c = Serial1.read();

    incoming += c;

  }

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

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:

if(Serial.available() > 0) {

  //code goes here

}

##### 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).

##### USB HID

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

keyboard.press('W');

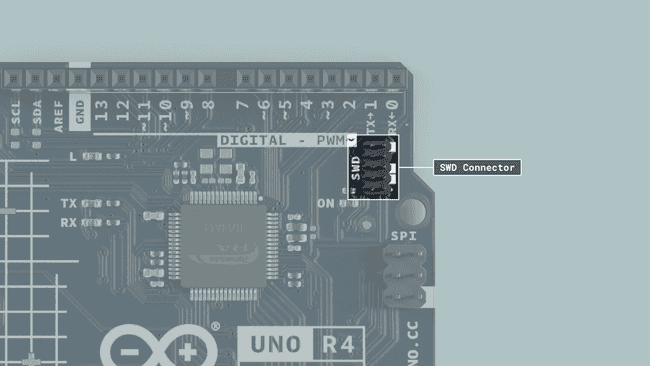
mouse.move(x,y);

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

* Keyboard: <https://www.arduino.cc/reference/en/language/functions/usb/keyboard/>
* Mouse: https://www.arduino.cc/reference/en/language/functions/usb/mouse/

To learn more about the HID capabilities of the UNO R4 Minima, check out the HID Guide: <https://docs.arduino.cc/tutorials/uno-r4-minima/usb-hid>

#### 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.

//set CAN bit rate and init library at

//choose from BR\_125k,BR\_250k,BR\_500k,BR\_1000k

CAN.begin(CanBitRate::BR\_250k);

Construct a CAN message and send it:

uint8\_t const msg\_data[] = {0xCA,0xFE,0,0,0,0,0,0};

memcpy((void \*)(msg\_data + 4), &msg\_cnt, sizeof(msg\_cnt));

CanMsg msg(CAN\_ID, sizeof(msg\_data), msg\_data);

CAN.write(msg);

Read an incoming CAN message.

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

**NOTE:** **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.

URL: <https://github.com/arduino/ArduinoCore-renesas>

Getting Started Guide: <https://docs.arduino.cc/tutorials/uno-r4-minima/minima-getting-started>

* **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 in following URL)

<https://www.renesas.com/us/en/software-tool/renesas-flash-programmer-programming-gui>

**NOTE: 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.

**NOTE: For more details, check the README.md or the GitHub page:**

[**https://github.com/arduino/ArduinoCore-renesas/tree/main/bootloaders/UNO\_R4**](https://github.com/arduino/ArduinoCore-renesas/tree/main/bootloaders/UNO_R4)

Alright, if you've finished reading this document completely, then you're ready to embark on the magical journey ahead. In the following chapters, we'll show you how to start from the very basics and gradually delve into deep learning. Let's get started!