# RedBoard Artemis Quick Start Guide

Perhaps you've been using Arduino Uno or Mega boards, or their many clones and derivatives such as the SparkFun RedBoard-Qwiic, and have run out of program memory, data memory, speed, or I/O features, now might be the time to move several rungs on the ladder and try either the RedBoard Artemis (to replace Uno), or the RedBoard Artemis ATP (to replace Mega). The SparkFun Artemis module on these boards uses the very powerful yet low power Ambiq Micro Apollo3 MCU which is an ARM Cortex-M4F.

To compare the boards, take a look at the products online:

RedBoard Qwiic: <https://www.sparkfun.com/products/15123>

RedBoard Artemis: <https://www.sparkfun.com/products/15444>

Redboard Artemis ATP: <https://www.sparkfun.com/products/15442>

The two Artemis boards (there are more but I'm only looking at two in this article), have more ... well, more of everything:

* More program memory (1 MB)
* More data memory (384 kB)
* More bits (32 vs 8)
* More speed (48 MHz clock, boostable to 96 MHz)
* More interesting peripherals, including Bluetooth low power (BLE), and a microphone.

All the Artemis line are 3.3 V boards making it very easy to connect a lot of different modules via the IIC bus. The boards come with a handy QWIIC connector making some additional parts a plug-and-play option such as my favorite display for small projects: the SparkFun 20x4 LCD (See: <https://www.sparkfun.com/products/16398>).

Another feature that I like very much is the USB-C connector which is both less fiddly than older USB connectors, and also allows your project to draw more current from the USB connection - up to 3 A per the USB-C specification.

If you look at the prices, the Artemis boards are in the same price range as the AVR-based (conventional) Arduino boards making them very attractive for new projects.

So what's not to like?

If you are programming your Arduino boards with the Arduino IDE (I'm running version 1.8.19), then you will need to install a new board manager to get up and running on an Artemis board. To use the BLE module, you'll also need a new library. I'm going to cover that in a separate article.

If you want all the details, you can start here: <https://learn.sparkfun.com/tutorials/hookup-guide-for-the-sparkfun-redboard-artemis>. It's very complete but requires some perseverance to work through the numerous links.

If you have experience with some coding using the Arduino IDE, then you should be able to get up and running in 10 to 15 minutes. I certainly did with the RedBoard Artemis. I had my usual pin 13 blue LED flasher running with no code changes at all. The RedBoard Artemis ATP took me a little longer. We'll see why a bit later.

Here are all the steps you need to take to get your Arduino IDE running. I did this on a Mac, but there is no difference on a PC other than the possibility that you have to install a USB driver for the CH340C USB interface chip. SparkFun has instructions for this here: <https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers/all>

## Installation Steps

Here are the steps I took:

* **STEP 1: Install the Artemis board definitions**
  + Open the Arduino IDE
  + Menu: Arduino, Preferences...
    - Additional Boards Manager URLS:
      * Enter: <https://raw.githubusercontent.com/sparkfun/Arduino_Apollo3/main/package_sparkfun_apollo3_index.json>
      * Click the OK button
    - Be patient while it gets the board configuration data
  + Menu: Tools, Board, Boards Manager...
    - Type the word: spark in the search window
    - Select the SparkFun Apollo3 Boards entry
    - Click the Install button
  + Be a lot more patient. The tools installation takes quite a while.
  + Really, just be patient. Make tea or something.
* **STEP 2: Do a quick test**
  + Close the Arduino IDE and restart it
    - (You don't absolutely have to do this, but it's menus get messed up during the install)
  + Plug a RedBoard Artemis into the computer with a USB-C cable
    - The board should power up (there is a green power LED)
    - You should also see the blue LED flashing (code installed before you got the board)
  + Menu: Tools, Board, SparkFun Apollo3, RedBoard Artemis
  + Menu: Tools, Port, <the USB port you plugged into>
  + Open the example app here called blue\_led\_flasher
  + Compile
    - Be patient. It takes a while first time
    - If you want to see what's going during compiles, open the Arduino preferences and check the box that says: Show verbose output during compilation. I usually leave this on.
  + If the compile went OK, click the upload button
  + You should see the blue LED flashing with a more interesting pattern.
* **DONE!**

If you have chosen to buy the RedBoard Artemis ATP (so called because it has All The Pins of the Apollo3 brought out), then the instructions above work exactly the same. What is a little different is that the ATP version does not have the blue LED on pin 13. Most other Arduino family boards I've played with have the pin 13 LED which is what makes it such a good thing to try first on a new board. The RedBoard Artemis ATP has the blue LED on pin 5. To avoid this issue in your code just use the *LED\_BUILTIN* define for the LED. It's value is 13 on most boards, but 5 on the ATP, which is just what you need.

## Performance

If you're happy now and keen to get on with your own project you can skip the rest. If you'd like to see how fast the Artemis is compared to a Uno-like board, keep reading.

I put together a simple app to do a load of multiply and divide operations on long ints and floats. I added a test of the *sqrt* function. There is nothing fancy here, I just wanted to get an idea of how fast the RedBoard Artemis is compared to a RedBoard Qwiic.

Here are the results from the RedBoard Qwiic:

Math on long variables...

Elapsed time: 5441 ms

Math on float variables...

Elapsed time: 1641 ms

Math with sqrt...

Elapsed time: 5967 ms

Here are the results from the same app run on the RedBoard Artemis:

Math on long variables...

Elapsed time: 43 ms

Math on float variables...

Elapsed time: 73 ms

Math with sqrt...

Elapsed time: 217 ms

You can see the happy news. The RedBoard Artemis with its Apollo 3 MCU is a lot faster than a traditional Arduino Uno with an ATmega328P.

## A Small Issue

While I was doing my initial tinkering with the Artemis boards I ran into a problem with printing float values. The normal Arduino support libraries don't have much support for printing floating point values. This is deliberate to keep the compiled code smaller. The usual way to print out a floating point value is to use the String class something like this:

String s(value, 3);

Serial.println(s);

When I ran this code, I got this:

%5.3f

which is the *sprintf* formatting string I'd normally use if we could use *sprintf* for floats (which is not generally supported on Arduino).

The *String* class for Artemis builds uses *dtostrf* which itself is implemented with *sprintf*.

It seems that the implementation of *sprintf* is broken and is ignoring the formatting string for floats (%f) and so the formatting string gets inserted into the formatted output.

## There's More to Artemis

There is a lot to discover about the Artemis boards such as the BLE interface, the on-board microphone, stereo audio support in the MCU, the real-time clock, and much more. You can find the full Ambiq Micro Apollo 3 specification here: <https://cdn.sparkfun.com/assets/1/5/c/6/7/Apollo3-Blue-MCU-Datasheet_v0_15_0.pdf>. It's only 940 pages so perfect for bedtime reading.

Happy coding.