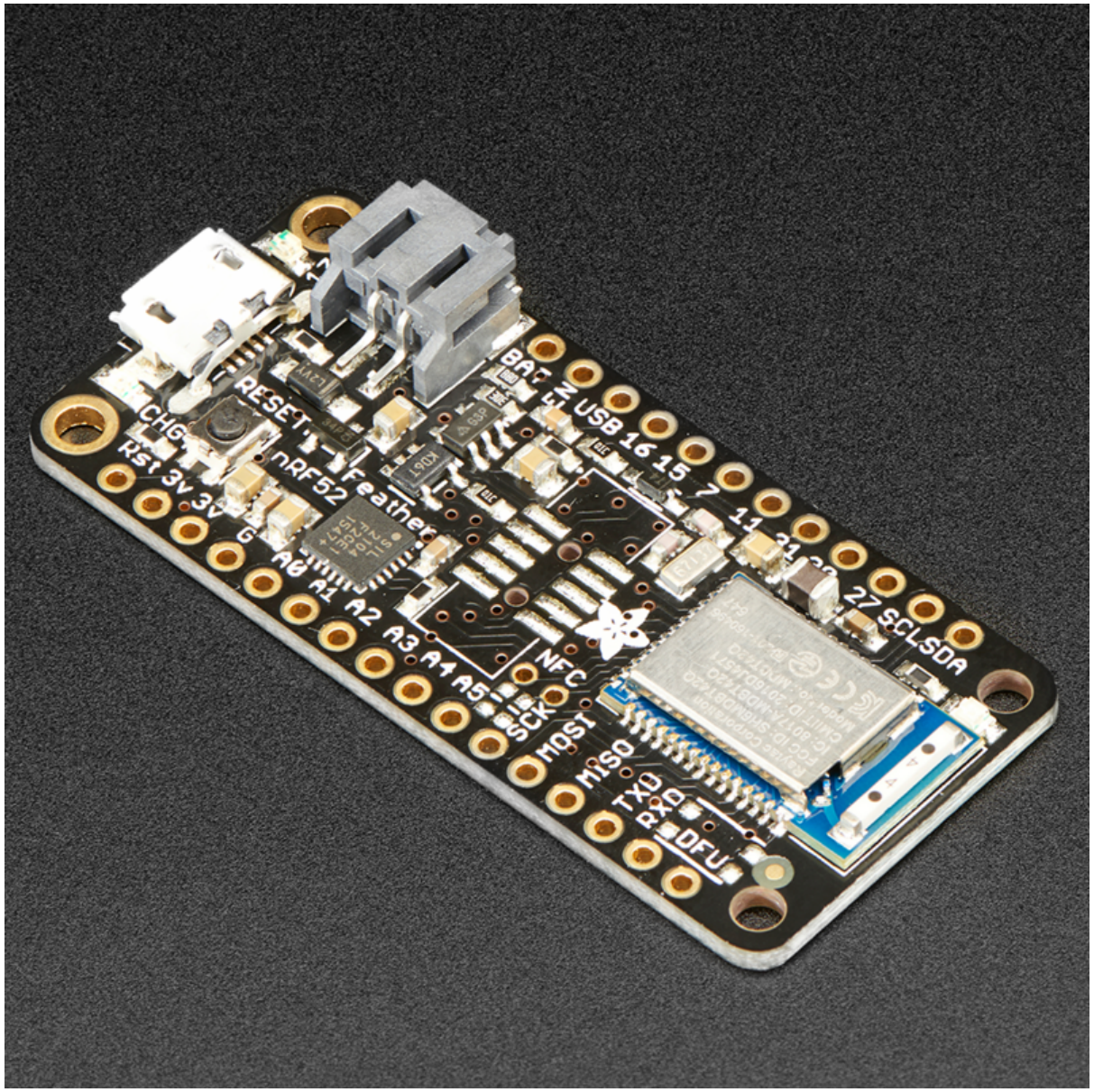
Adafruit will not be shipping orders Labor Day, Monday September 3rd. Expedited orders placed after 11am ET Friday August 31st will go out Tuesday September 4th.



Bluefruit nRF52 Feather Learning Guide

Get started now with our most powerful Bluefruit board yet!

Contributors

Kevin Townsend
Thach Ha

Feedback? Corrections?
MICROCONTROLLERS FEATHER / FEATHER BOARDS INTERNET OF THINGS - IOT / BLUEFRUIT / BLE

FAQs

by Kevin Townsend

NOTE: For FAQs relating to the **BSP**, see the dedicated BSP FAQ list.

What are the differences between the nRF51 and nRF52 Bluefruit boards? Which one should I be using?

The two board families take very different design approaches.

All of the nRF51 based modules are based on an AT command set (over UART or SPI), and require two MCUs to run: the nRF51 hosting the AT command parser, and an external MCU sending AT style commands.

The nRF52 boards run code directly on the nRF52, executing natively and calling the Nordic S132 SoftDevice (their proprietary Bluetooth Low Energy stack) directly. This allows for more efficient code since there is no intermediate AT layer or transport, and also allows for lower overall power consumption since only a single device is involved.

The nRF52 will generally give you better performance, but for situation where you need to use an MCU with a feature the nRF52 doesn't have (such as USB), the nRF51 based boards will still be the preferable solution.

Can I run nRF51 Bluefruit sketches on the nRF52?

No. The two board families are fundamentally different, and have entirely separate APIs and programming models. If you are migrating from the nRF51 to the nRF52, you will need to redesign your sketches to use the newer API, enabling you to build code that runs natively on the nRF52832 MCU.

Can I use the nRF52 as a Central to connect to other BLE peripherals?

The S132 Soft Device and the nRF52832 HW support Central mode, so yes this is *possible.* At this early development stage, though, there is only bare bones support for Central mode in the Adafruit nRF52 codebase, simply to test the HW and S132 and make sure that everything is configured properly. An example is provided of listening for incoming advertising packets, printing the packet contents and meta-data out to the Serial Monitor. We hope to add further Central mode examples in the future, but priority has been given to the Peripheral API and examples for the initial release.

How are Arduino sketches executed on the nRF52832? Can I do hard real time processing (bit-banging NeoPixels, etc.)?

In order to run Arduino code on the nRF52 at the same time as the low level Bluetooth Low Energy stack, the Bluefruit nRF52 Feather uses FreeRTOS as a task scheduler. The scheduler will automatically switch between tasks, assigning clock cycles to the highest priority task at a given moment. This process is generally transparent to you, although it can have implications if you have hard real time requirements. There is no guarantee on the nRF52832 to meet hard timing requirements when the radio is enabled an being actively used for Bluetooth Low Energy. This isn't possible on the nRF52832 even without FreeRTOS, though, since the SoftDevice (Nordic's propietary binary blob stack) has higher priority than any user code, including control over interrupt handlers.

Can I use GDB to debug my nR52832?

You can, yes, but it will require a Segger J-Link (that's what we've tested against anyway, other options exist), and it's an advanced operation. But if you're asking about it, you probably know that.

Assuming you have the Segger J-Link drivers installed, you can start Segger's GDB Server from the command line as follows (OSX/Linux used here):

```
$ JLinkGDBServer -device nrf52832_xxaa -if swd -speed auto
```

Then open a new terminal window, making sure that you have access to `gcc-arm-none-eabi-gdb` from the command line, and enter the following command:

```
$ ./arm-none-eabi-gdb something.ino.elf
```

`` `something.ino.elf` `` is the name of the .elf file generated when you built your sketch. You can find this by enabling 'Show verbose output during: [x] compilation' in the Arduino IDE preferences. You CAN run GDB without the .elf file, but pointing to the .elf file will give you all of the meta data like displaying the actual source code at a specific address, etc.

Once you have the `(gdb)` prompt, enter the following command to connect to the Segger GDB server (updating your IP address accordingly, since the HW isn't necessarily local!):

```
(gdb) target remote 127.0.0.1:2331
```

If everything went well, you should see the current line of code where the device is halted (normally execution on the nRF52 will halt as soon as you start the Segger GDB Server).

At this point, you can send GDB debug commands, which is a tutorial in itself! As a crash course, though:

- To continue execution, type '`monitor go`' then '`continue`'
- To stop execution (to read register values, for example.), type '`monitor halt`'
- To display the current stack trace (when halted) enter '`bt`'
- To get information on the current stack frame (normally the currently executing function), try these:
  - `info frame`: Display info on the current stack frame
  - `info args`: Display info on the arguments passed into the stack frame
  - `info locals`: Display local variables in the stack frame
  - `info registers`: Dump the core ARM register values, which can be useful for debugging specific fault conditions

Are there any other cross platform or free debugging options other than GDB?

If you have a Segger J-Link, you can also use Segger's OZone debugger GUI to interact with the device, though check the license terms since there are usage restrictions depending on the J-Link module you have.

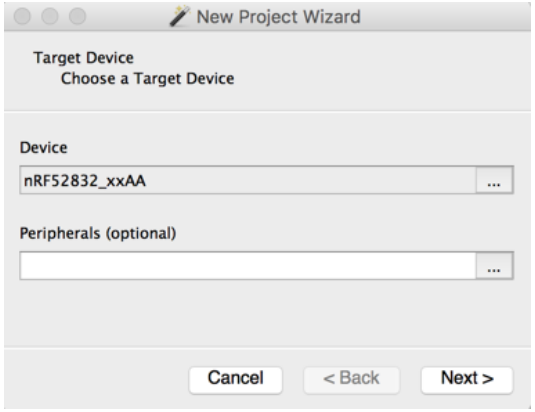
You will need to connect your nRF52 to the J-Link via the SWD and SWCLK pins on the bottom of the PCB, or if you are OK with fine pitch soldering via the SWD header.

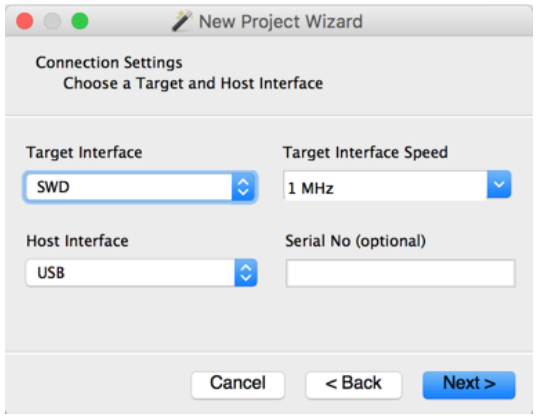
You can either solder on a standard 2x5 SWD header on the pad available in the board, or you can solder wires to the SWD and SWCLK pads on the bottom of the PCB and use an SWD Cable Breakout Board, or just connect cables directly to your J-Link via some other means.

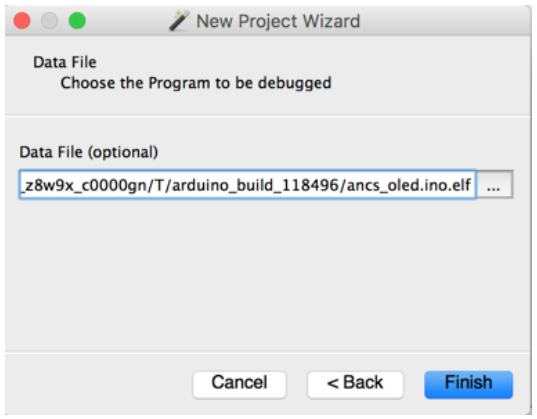
You will also need to connect the **VTRef** pin on the JLink to **3.3V** on the Feather to let the J-Link know what voltage level the target has, and share a common GND by connecting the GND pins on each device.

Before you can start to debug, you will need to get the .elf file that contains all the debug info for your sketch. You can find this file by enabling **Show Verbose Output During: compilation** in the **Arduino Preferences** dialogue box. When you build your sketch, you need to look at the log output, and find the .elf file, which will resemble something like this (it will vary depending on the OS used): `/var/folders/86/hb2vp14n5_5_yvdz_z8w9x_c0000gn/T/arduino_build_118496/ancs_oled.ino.elf`

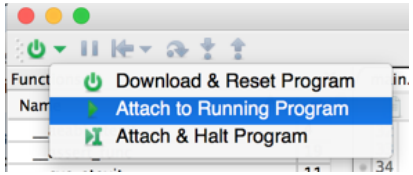
In the OZone New Project Wizard, when prompted to select a target device in OZone select **nRF52832_xxAA**, then make sure that you have set the Target Interface for the debugger to **SWD**, and finally point to the .elf file above:

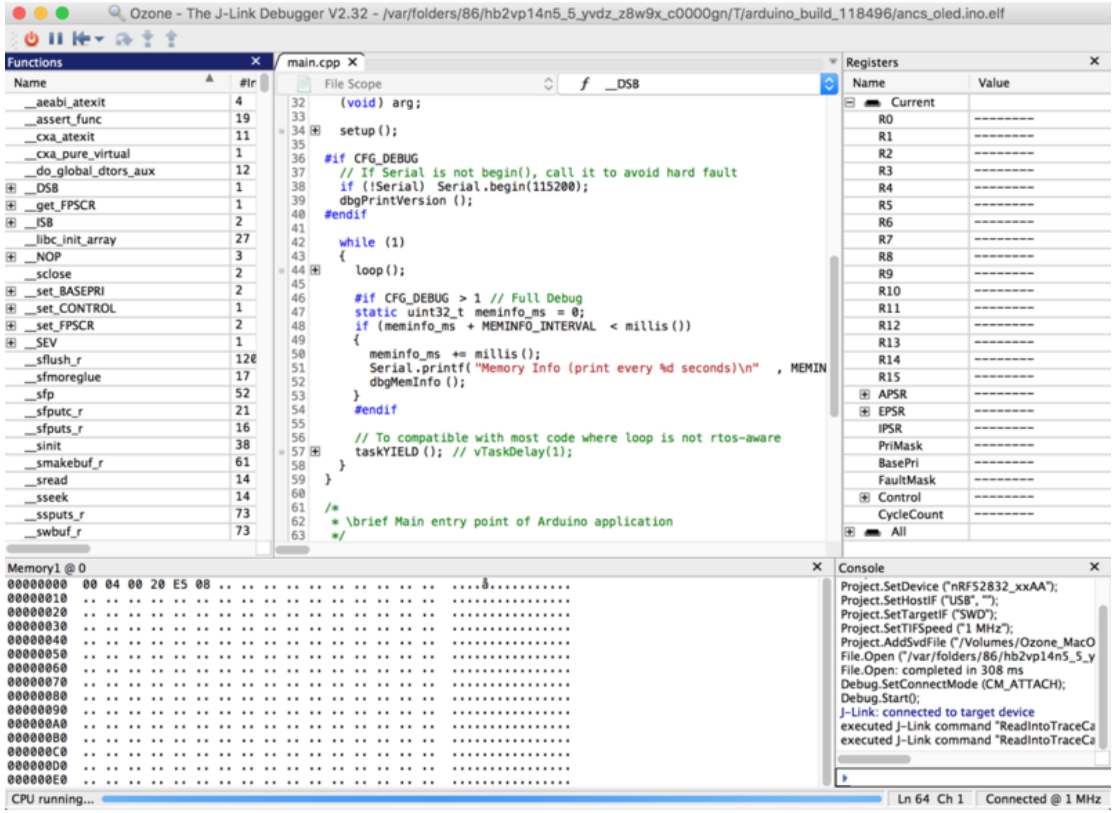




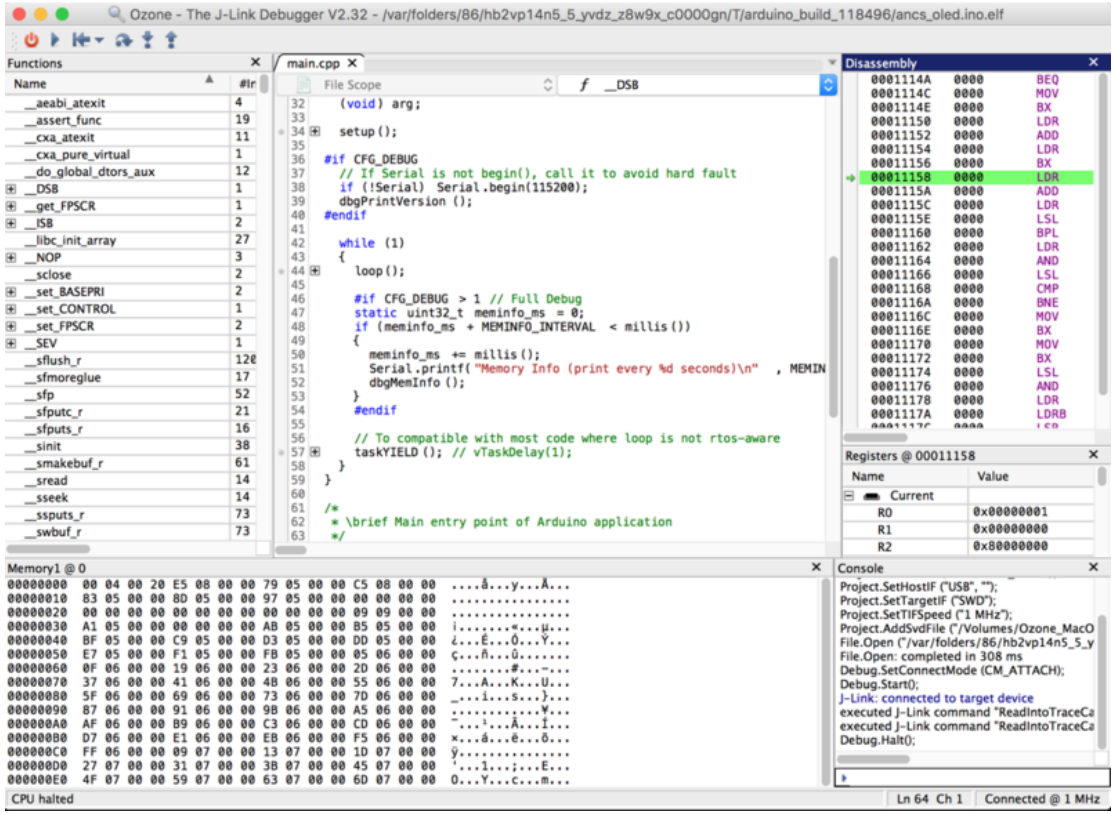


Next select the **Attach to running program** option in the top-left hand corner, or via the menu system, which will cause the debugger to connect to the nRF52 over SWD:

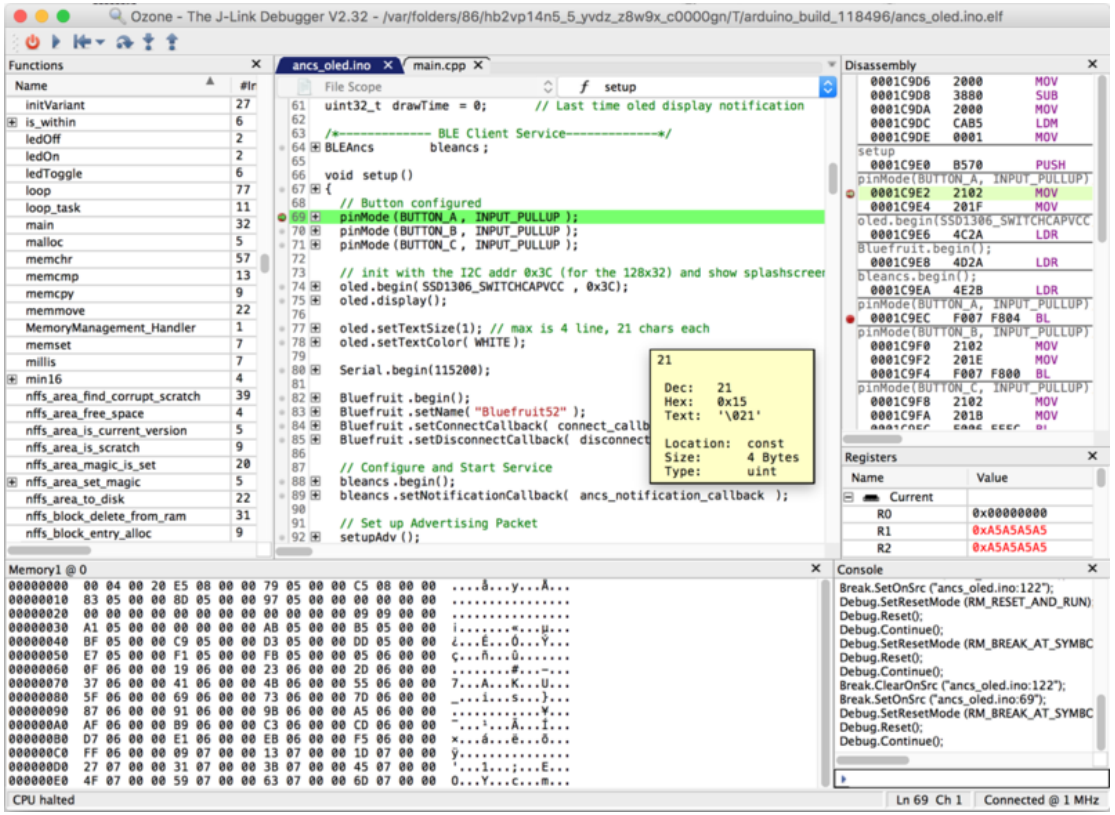




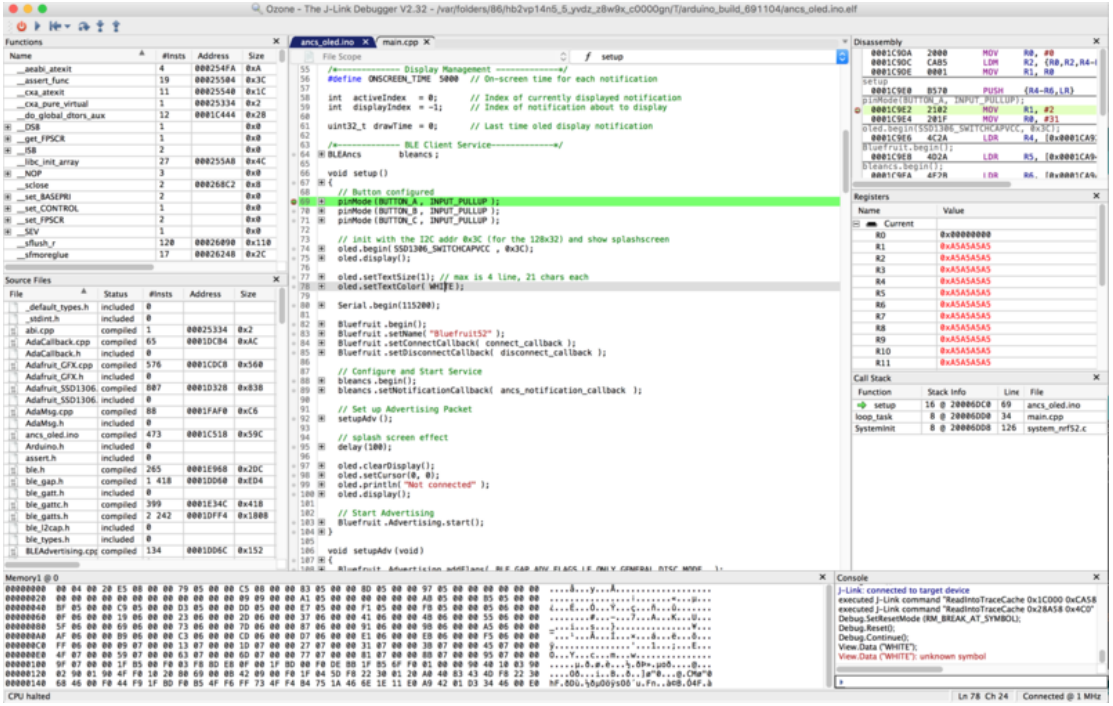
At this point, you can click the **PAUSE** icon to stop program execution, and then analyze variables, or set breakpoints at appropriate locations in your program execution, and debug as you would with most other embedded IDEs!



Clicking on the left-hand side of the text editor will set a breakpoint on line 69 in the image below, for example, and the selecting **Debug > Reset > Reset & Run** from the menu or icon will cause the board to reset, and you should stop at the breakpoint you set:

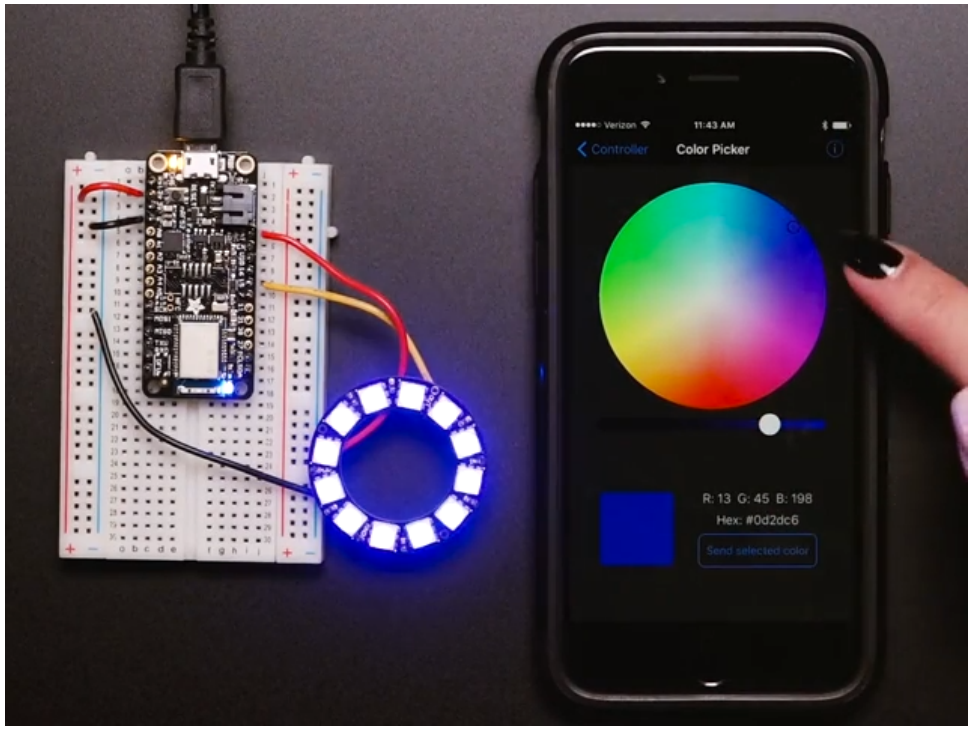


You can experiment with adding some of the other debug windows and options via the **View** menu item, such as the **Call Stack** which will show you all of the functions that were called before arriving at the current breakpoint:

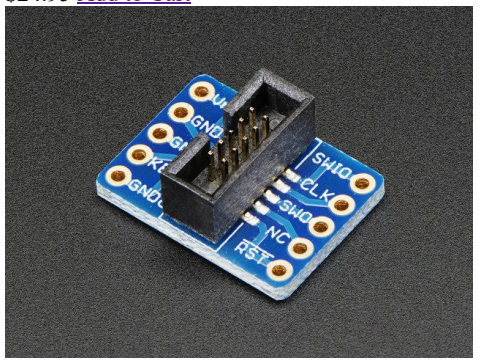


DOWNLOADS

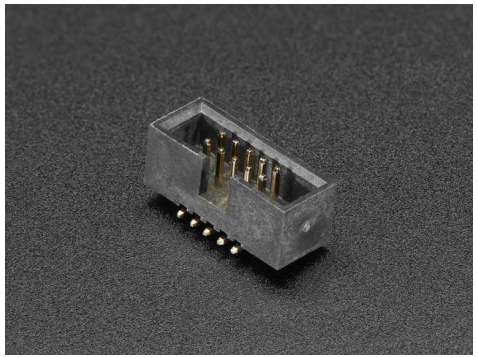
Last updated on Jan 03, 2018 Published on Mar 22, 2017



Adafruit Feather nRF52 Bluefruit LE
$24.95 Add to Cart



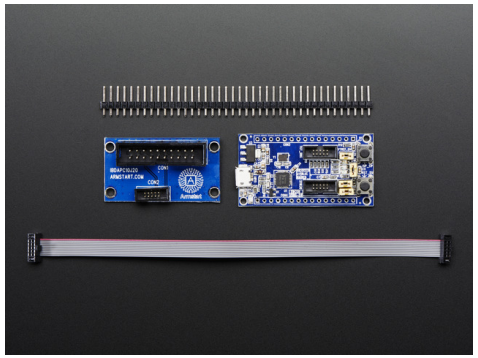
SWD (2x5 1.27mm) Cable Breakout Board
$1.95 Add to Cart



SWD 0.05" Pitch Connector - 10 Pin SMT Box Header
$1.50 Add to Cart

SEGGER J-Link BASE - JTAG/SWD Debugger
$399.95 Add to Cart

SEGGER J-Link EDU - JTAG/SWD Debugger
$69.95 Add to Cart



IBDAP - CMSIS-DAP JTAG/SWD Debug Adapter Kit
$29.95 Add to Cart

10-pin 2x5 Socket-Socket 1.27mm IDC (SWD) Cable - 150mm long
$2.95 Add to Cart

RELATED GUIDES

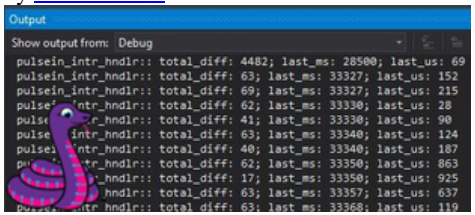
Adafruit seesaw

An I2C to ... whatever! interface
by Dean Miller

PWM, ADC, NeoPixels...You want to use em but if you don't have hardware support it's impossible! Glue an Arduino to the side? No longer! Seesaw gives you multiple hardware interfaces over I2C for robotics, sensing, lighting and more!

Debugging CircuitPython On SAMD w/Atmel Studio 7

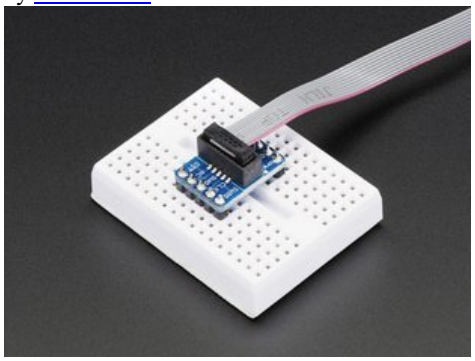
Un-Constricted CircuitPython Debugging in a Windows Environment!
by M. Schroeder



This guide will help you to start debugging CircuitPython on the SAMD21 or SAMD51, using Windows and Atmel Studio 7. Take the squeeze off virtual environments and do some work in your native operating system!

Programming an M0 using an Arduino

computers are overrated anyhow
by Dean Miller



Programming ARM core microcontrollers can be bulky and time consuming. This guide hows how to do it simply using an ATSAMD21 based arduino-compatible board.

Adafruit seesaw

An I2C to ... whatever! interface
by Dean Miller

PWM, ADC, NeoPixels...You want to use em but if you don't have hardware support it's impossible! Glue an Arduino to the side? No longer! Seesaw gives you multiple hardware interfaces over I2C for robotics, sensing, lighting and more!

×

OUT OF STOCK NOTIFICATION

YOUR NAME
YOUR EMAIL
NOTIFY ME
×

Create Wishlist

Title

Description
Create Wishlist CLOSE

- CONTACT
- SUPPORT
- DISTRIBUTORS
- EDUCATORS
- JOBS
- FAQ
- SHIPPING & RETURNS
- TERMS OF SERVICE
- PRIVACY & LEGAL
- ABOUT US

ENGINEERED IN NYC Adafruit ®
"You're only given one little spark of madness. You mustn't lose it" - Robin Williams