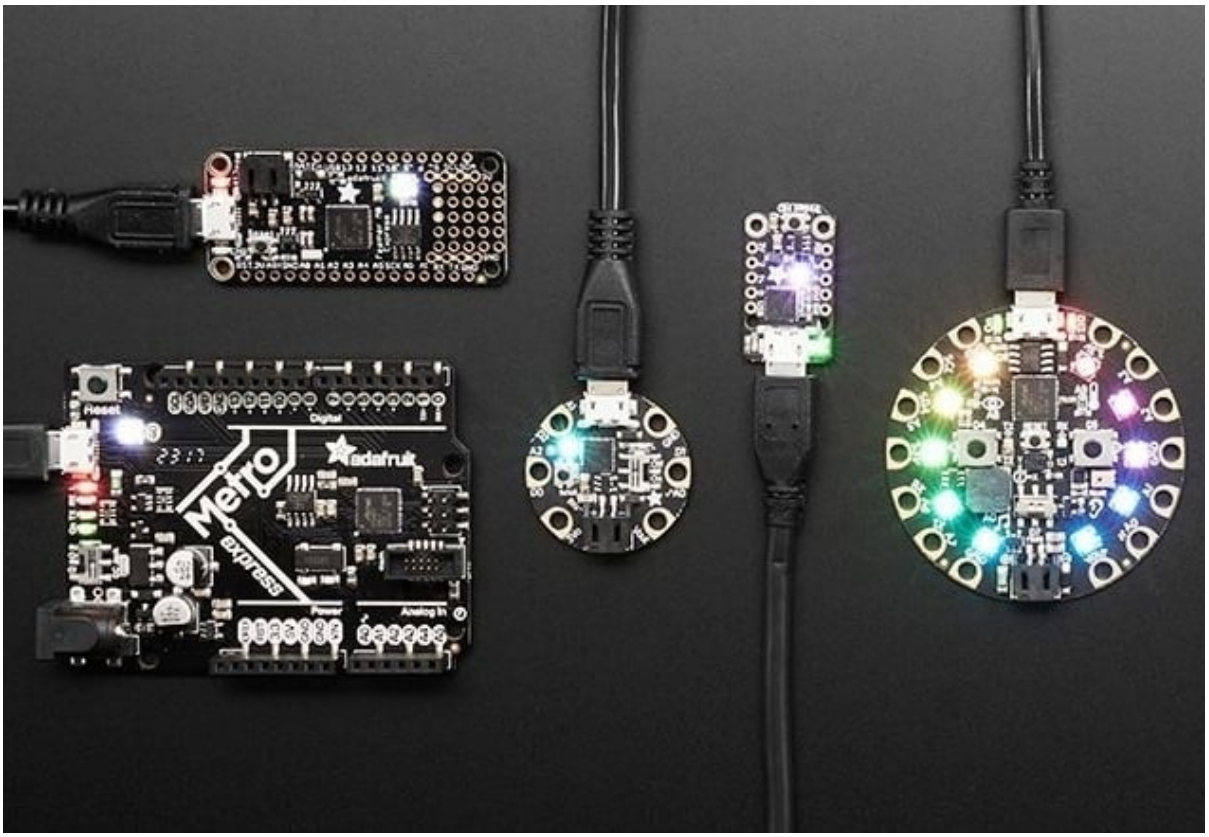
How to Choose a Microcontroller

Created by mike stone



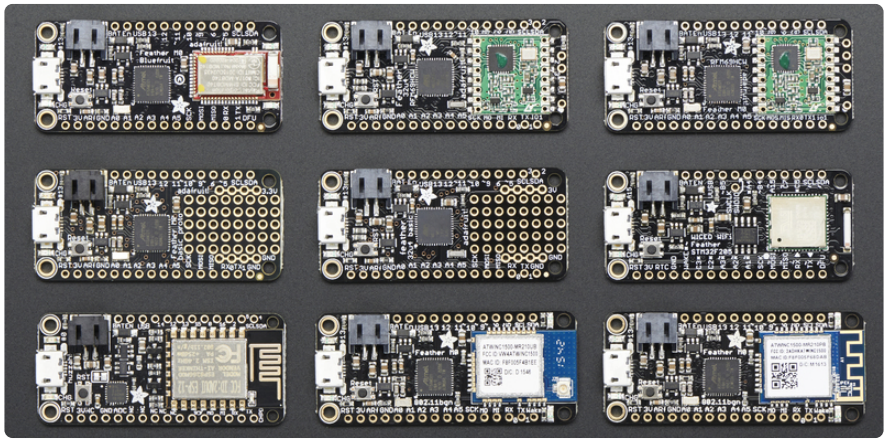
https://learn.adafruit.com/how-to-choose-a-microcontroller

Last updated on 2024-11-02 01:20:48 PM EDT

Table of Contents

Overview

Which Microcontroller is the Best?



We see that question in the Adafruit forums (https://adafru.it/dYq) all the time, especially from people who are just getting started. Adafruit has more than fifty boards in the Feather line alone, to say nothing of other platforms like the Metro, Circuit Playground, Arduino, and Teensy. Trying to choose one, especially if you haven't already worked with all of them, can be bewildering.

Our answer, predictably, is: "it depends what you are looking to do."

There's no such thing as a 'best' microcontroller... if there was, all the manufacturers would make that, and there wouldn't be any choice left.

Instead, you have to make tradeoffs between competing interests: The fastest chips tend to use more power than the slowest ones. The boards with the most IO pins are bigger (and usually cost more) than ones with fewer features.

This guide will explain which microcontroller boards are easiest to work with, which ones are more complex, and how to choose between them.

A reader asked for a concise table of boards. This doesn't exist to our knowledge. Besides this guide, see the Make: Boards Guide at https://makezine.com/comparison/boards/

How to Make a Bad Choice

H.L. Mencken said, "**there is always a well-known solution to every human problem—neat, plausible, and wrong**"

The microcontroller you really want has the most features you plan to use and the fewest that will get in your way. Even people who've worked with lots of different boards have to spend time thinking about that.

There are two easy ways to make a bad choice though:

- Pick the one that's cheapest.
- Pick the one that's newest/fastest/biggest.

Those boards have already made serious trade-offs to get where they are, and they have corresponding limits. If you don't know what the limits are, you don't know if they'll get in your way.

Only use the -est of something when you know you want that more than anything else.

Remember: a feature only makes a board better if you actually use it!

On the next page is information to assist in making informed choices when it comes to microcontrollers.

How to Make a Good Choice

Sketching out the features you might want

Sketch out on a piece of paper or in a notebook what you are looking to do in your project. Start with a general concept. Then list the general features that concept will use. Now you can start to search for parts that implement the features you have envisioned.

You definitely do not want fewer features than your design calls for. On the other side, do not pick a part that has a great deal more features than you are looking for (although a bit of expandability might be desirable).

A list of design considerations

Here are some considerations to think about:

- **Power** - does your project run on batteries or wall power/mains? If it only uses batteries, consider a design that conserves power to run longer between battery changes. Or consider if solar or other power sources might be desirable.

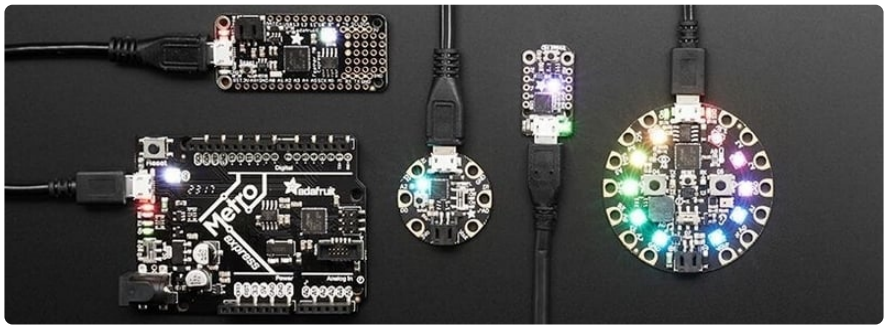
- **Speed** - does your project crunch a lot of numbers or gather huge amounts of data? Then speed may be good. Otherwise if the project is relatively simple, a smaller, slower board may be perfectly fine (and less expensive).

- **Price** - what is your budget? Keep this in mind when shopping but don't consider buying anything that will not fulfill the design of your project.

- **Communications** - does your project need to communicate information to you or other devices? Consider how to do that. Radio-enabled projects are great but they will require a transmitter and receiver so the amount of hardware may be double what you thought. You can communicate via USB to a computer often easily. Or maybe infrared signals?

- **Programming** - less of a consideration but important. Your time is valuable, so something easier to program may save you time/money even if its a little more expensive
  The easiest to program boards use Microsoft MakeCode (https://adafru.it/Co6), a block/Scratch like interface, and only applies to a limited board selection like the Adafruit Circuit Playground Express (http://adafru.it/3333).
  CircuitPython (https://adafru.it/cpy-welcome) is more broadly supported in Adafruit boards but not all.
  Finally, most boards can be programmed via the Arduino (https://adafru.it/Co7) code environment but there is a higher learning curve. The Adafruit Learning System (https://adafru.it/dlu) will have examples but you will have to be somewhat comfortable extending the code to your own needs if necessary.

Kinds of Microcontrollers

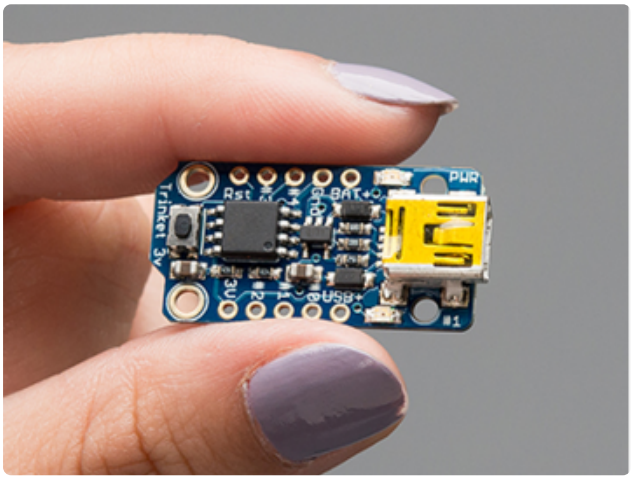


Microcontrollers are described in terms of the number of bits the CPU can process or move in and out of memory at once. The major flavors these days are 8-bit and 32-bit devices.

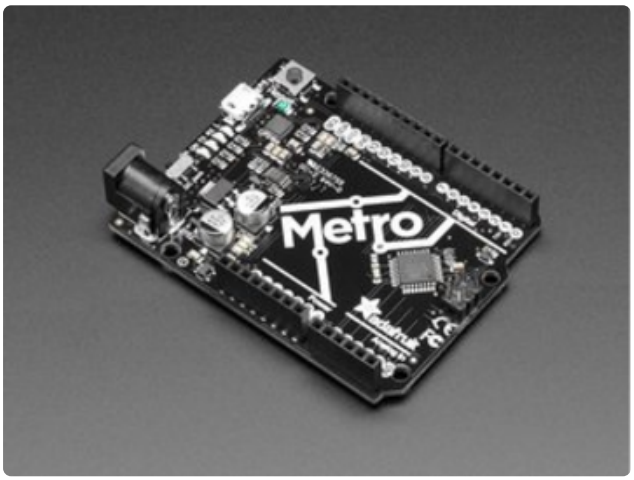
Those terms are just shorthand for more important differences in the way the chips are made though.

All microcontrollers contain a processor that executes code and a set of circuits called 'peripherals' that provide additional functions: USB, serial interfaces, ADCs, timers, and so on. The biggest difference between microcontrollers is the relationship between the processor, the peripherals, and the physical pins that come out of the package.

8-bit Microcontrollers



8-bit devices are the simplest kind available these days and have been around the longest. They're generally simple enough that you can learn everything about them in a reasonable amount of time and there may be more sample code available on the Internet.



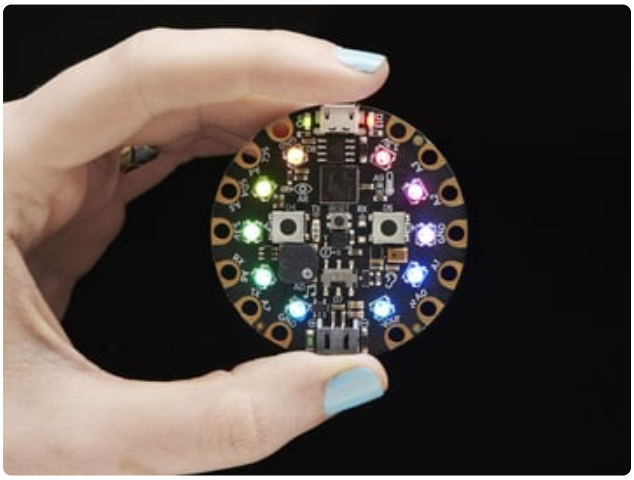
The Arduino Uno and the Adafruit Metro 328 (Uno compatible) are the most common boards in the 8-bit market with lots of software available. The tiny Adafruit Trinket is also 8-bit with fewer features.

An 8-bit microcontroller's peripherals are connected more or less directly to the processor and the physical pins.

If you change the voltage on a pin, the peripheral converts that to a value in the processor, then your code can use that value. Output from code to the physical pins works the same way.

Some 8-bit microcontrollers only have a few peripherals while others have many, but the overall design tends to remain the same: the peripherals connect the physical pins to data values in the processor.

32-bit Microcontrollers



A 32-bit device is more like a hardware store in a very small box. The actual microcontroller is only a small part of the chip, and usually isn't even discussed in the datasheet. Chip manufacturers tend to license generic processor designs like the ARM Cortex series.

The peripherals in a 32-bit microcontroller are more like separate devices connected to one or more data buses, and almost everything is a peripheral. In the SAMD21 (which we use in the Adafruit Circuit Playground Express (http://adafru.it/3333) and the Adafruit Feather M0 Express (http://adafru.it/3403)), the CPU clock, RAM, and Flash memory array are all peripherals.

More About Peripherals with 32-Bit Chips

Peripherals are the 'extras' beyond plain number crunching - like digital i/o pins, capacitive touch, ADC, SPI, I2C, UART, I2S, etc...

The peripherals don't have any direct connection to the processor. More traditional peripherals like the ADC and SPI hardware are connected to a separate data bus.

Connections between peripherals linked to the data buses are handled by an internal switching network, and that has two interesting side effects. First, peripherals can run from different clocks ticking at different rates. Second, peripherals can communicate with each other directly, without any control from code being executed in the processor. Chips with a Direct Memory Access (DMA) peripheral can move whole blocks of data from one peripheral to another. You can configure a chip to do many interesting things with the actual microcontroller completely shut down.

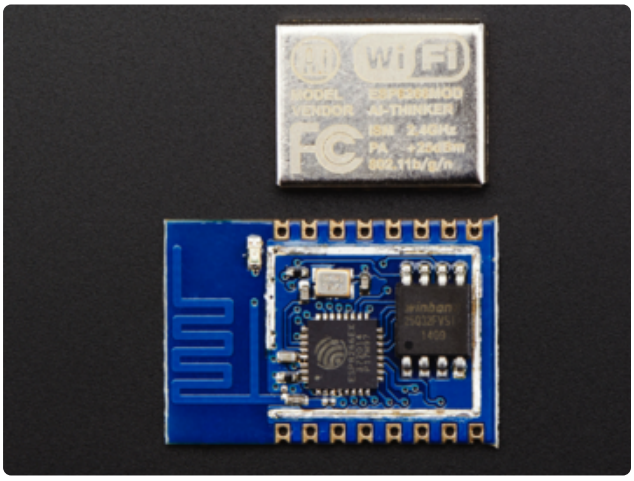
All of the physical pins are connected to a single peripheral called the pin multiplexer. In some chips, the multiplexer can connect any physical pin to any signal on any peripheral. The SAMD21 usually gives peripherals two or three pins that can handle a given signal, and a single physical pin can have as many as nine possible connections to peripherals.

The freedom to configure peripherals and move signals around like that makes 32-bit microcontrollers extremely flexible, but there are a couple of disadvantages as well.

The software written for an Adafruit product may not be 100% compatible with an Arduino or other manufacturer's product, even though they use the same chip.

So there may be less code on the Internet to show you how to use a 32-bit chip. Fortunately, Adafruit has invested a great deal to provide code examples, reusable code libraries and tutorials to easily use their more advanced products.

System On Chip Devices (SOC)



An SOC is usually a 32-bit microcontroller designed to do a specific thing. It's made the tradeoffs necessary to be good at that thing at the expense of all-around performance.

The ESP8266 is one example: it's designed to do WiFi at a very low cost. It has few input/output pins and peripherals, but one of the peripherals is a WiFi radio. It has large collection of built-in firmware, and a simple operating system that keeps the WiFi radio running more of less independently of the code you upload.

Some projects look to use this chip without WiFi solely due to cost. But the lack of input/output pins and peripherals often make it necessary to switch to a more capable chip.

The ESP32 and nRF52 are also SOCs. The ESP32 is intended to be a one-stop-shopping solution for any device with a wireless data connection. It can do WiFi, Bluetooth Classic, and Bluetooth Low Energy (BLE). The nRF52 focuses on BLE, and can operate as a BLE central device.

The microcontrollers in Adafruit products

With the general information out of the way, let's run through the microcontrollers used in Adafruit development boards, from the simplest to the most complex:

8-bit Microcontrollers:

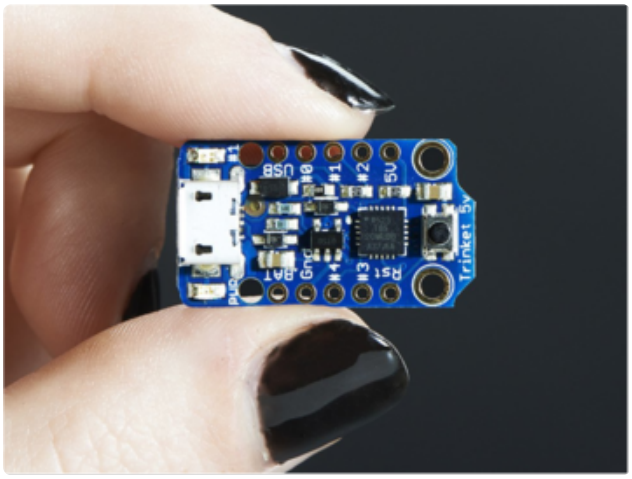
8-bit microcontrollers can only be programmed with the Arduino IDE. They are not powerful enough for MakeCode or CircuitPython.

Note the cost differential between 8-bit and the more capable 32-bit processors has narrowed quite a bit in the last two years such that the small cost difference may be worth the increased functionality 32-bit boards provide.

The ATtiny85

Small and simple, the ATtiny85 is an 8-bit CPU in an 8-pin package. It runs at 8MHz and has 6 GPIO pins.

> The ATtiny85 is low cost but we don't recommend it anymore because the bit-bang USB is annoying to use. Consider the Trinket M0 and Gemma M0 as 32-bit upgades!



It has 8 kilobytes of Flash to hold firmware, 512 bytes of RAM, and 512 bytes of EEPROM that you can read and write from code.

For peripherals, it has two 8-bit timers that can generate PWM signals, a Universal Serial Interface that can be configured to speak I2C or SPI, an analog voltage comparator, and a 10-bit ADC.

Adafruit uses it in the original Trinket (http://adafru.it/1501) and Gemma (http://adafru.it/1222).

The ATtiny85 a good chip for very small jobs like reading three or four inputs and deciding what output to produce, or for generating time delays. A soft power module for a Raspberry Pi would be one example: pushing the button once turns power on. When you push the button again, the microcontroller generates a signal that tells the

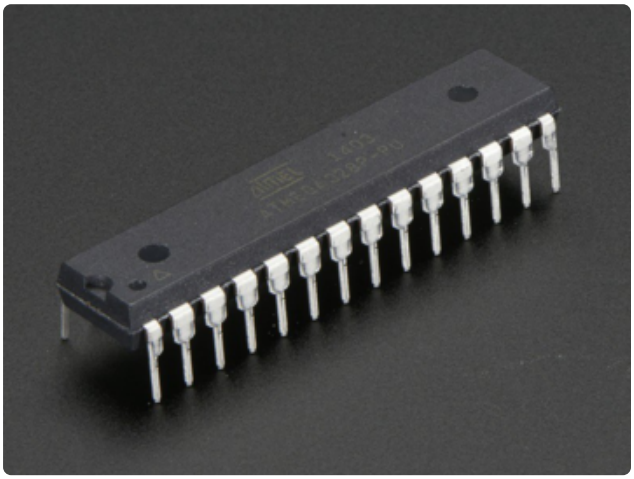
RasPi to do an orderly shutdown, waits a while, then cuts the power after the RasPi has shut down completely.

You could build that circuit from logic gates, but it would get cumbersome. You could also use a bigger microcontroller, but that's more money and physical space for a chip with lots of features you don't use. The ATtiny85 is smaller and less expensive than either option.

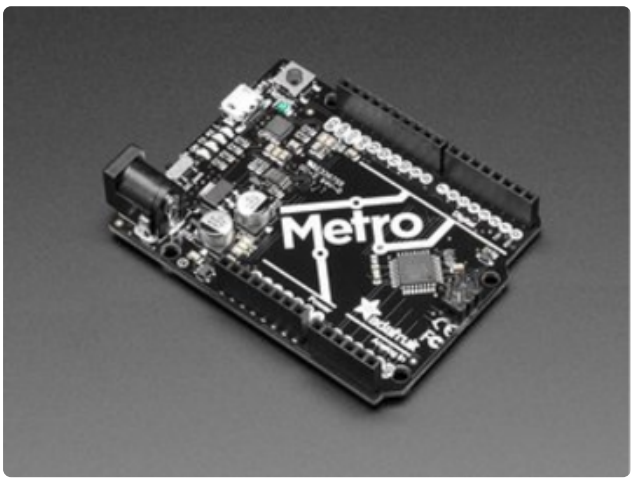
The ATmega328P

This is the chip the Arduino Uno made famous. It's an 8-bit microcontroller that normally runs at 16MHz and has 23 GPIO pins.

It has 32K of Flash program memory, 2K of RAM, and 1K of EEPROM.



For peripherals, it has two 8-bit timers and one 16-bit timer, six PWM channels, a 10-bit ADC, and separate hardware Serial, I2C, and SPI interfaces.



We use it in the Metro 328P (http://adafru.it/2488), Feather 328P (http://adafru.it/3458), and Pro Trinket (http://adafru.it/2000).

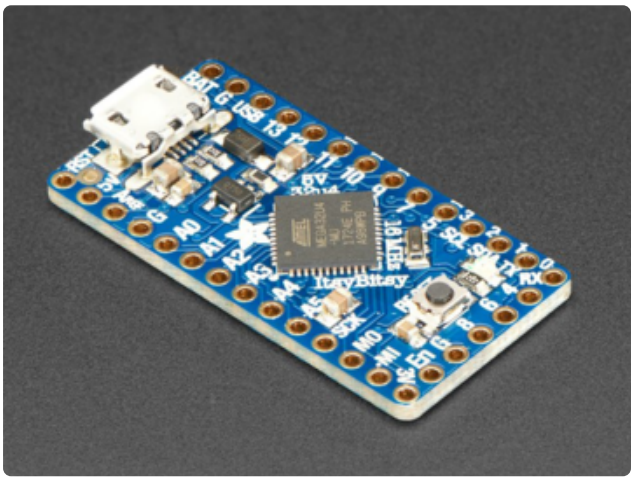
It's a good all-around chip for programmable hardware projects, as demonstrated by the thousands of projects people have built around the Arduino Uno. It has enough data pins to read several input signals, gather information from external I2C or SPI sensors, control an LCD or TFT display, and send information back to a computer through its USB serial port.

The ATmega32u4

The 32u4 is mostly comparable to the 328P, but is also a full-speed (12Mbps) USB device. It's an 8-bit microcontroller that usually runs at 16MHz and has 26 GPIO pins. The big benefit is the native USB, which makes it possible for the 32u4 to act like a USB mouse, keyboard, MIDI etc. The remaining overall peripherals are nearly identical to the 328P.

The 32u4 is used in the Arduino Leonardo and many compatibles.

It has 32K of program Flash and 1K of EEPROM like the 328P, but has 2.5K of RAM.



For peripherals, it has one 8-bit timer, one 10-bit timer, two 16-bit timers, fourteen PWM channels, a hardware Serial interface that can also be configured as an SPI interface, separate I2C and SPI interfaces, a 10-bit ADC, and (obviously) a 12Mbps USB interface.

Adafruit uses it in the Pro Trinket (http://adafru.it/2000), Flora (http://adafru.it/659), 32u4 Feather (http://adafru.it/2771), 32u4 ItsyBitsy (http://adafru.it/3677), and Circuit Playground Classic (http://adafru.it/3000).

The USB peripheral gives the ATmega32u4 two major differences from the ATmega328P:

1. The 32u4 bootloader doesn't need to use another hardware serial interface chip, so those pins are free to communicate with external devices or other microcontrollers.
2. You can program it to act like different kinds of USB devices, like a keyboard and mouse, or a MIDI controller.

That makes the 32u4 a good choice for projects that need to communicate with a computer and where you want to keep some '328P code-compatibilities

32-bit Microcontrollers:

32-bits is more than 8-bits! This makes these boards faster, and more powerful. Along with 32-bit math, you can also expect a lot more Flash memory, RAM memory, and clock speed.

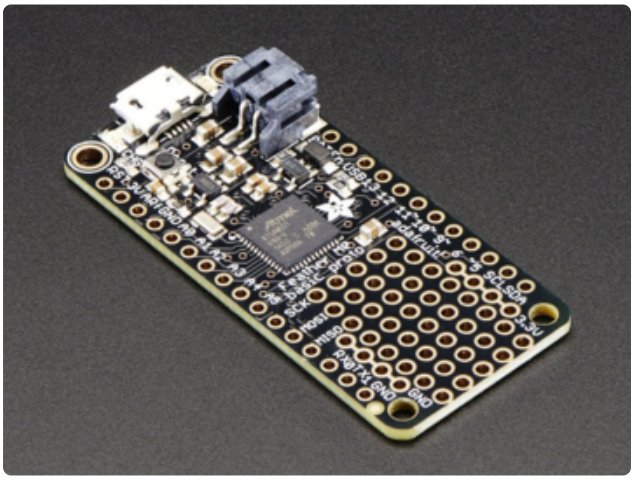
The world is turning to using 32-bit for all microcontrollers. Adafruit likes the ARM Cortex M0 and M4 series, these are very popular cores with good compiler support. (Compared to the 8-bit cores which are proprietary to Atmel only!)

The price of the M0 based boards have come down to the same levels as most 8-bit boards, so cost is often not a consideration.

These chips are fast and powerful so they can run MakeCode and CircuitPython as well as Arduino.

The SAMD21G

The SAMD21G uses an ARM Cortex M0+ microcontroller, which is about as small and simple as you get with 32-bit devices.



It runs at 48MHz, usually stabilized by an external 32,768Hz crystal oscillator, and has 38 GPIO pins.

It has 256K of Flash memory and 32K of RAM. It doesn't have any EEPROM, but you can read and write sections of the Flash array from code.

Unlike 8-bit microcontrollers, a 32-bit microcontroller can have multiple copies of a peripheral that are called 'instances'.

The SAMD21G has:

- three instances of a basic timer, each of which can be configured with 8. 16, or 24 bits and can generate two PWM waveforms.
- three instances of an advanced timer, two 24-bit and one 16-bit, each of which can generate up to eight PWM waveforms.

- six instances of a Serial Communication Interface (SERCOM), each of which can be configured as a Serial, I2C, or SPI interface.
- one I2S interface for digital audio.
- one 12-bit ADC that can take input from any of 14 pins
- one 10-bit DAC
- one Peripheral Touch Controller that can read a 12x10 matrix of capacitive touch sensors
- one full-speed (12Mbps) USB interface

It also has an Event system which allows peripherals to talk to each other directly, and a 12-channel Direct Memory Access (DMA) system that can copy data from one peripheral to another without making the CPU read and write the bytes directly.

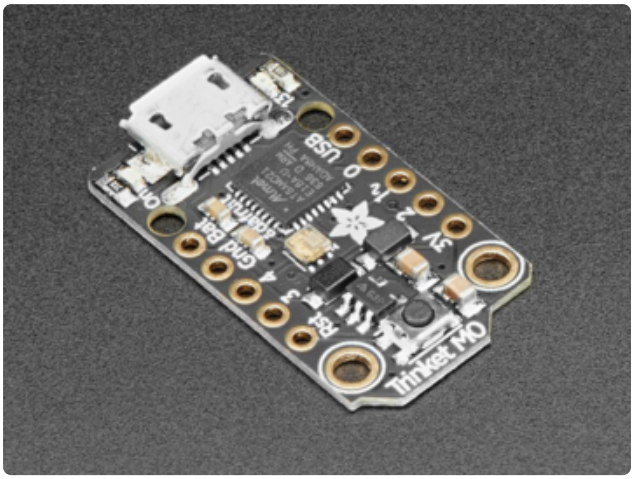
We use it in the Feather M0 (http://adafru.it/2772), Metro M0 Express (http://adafru.it/3505), ItsyBitsy M0 Express (http://adafru.it/3727), and Circuit Playground Express (http://adafru.it/3333).

It's a device made for crunching numbers and passing around large amounts of data. It's good for projects where you want to control a large number of programmable LEDs or do basic audio and video processing.

It also has enough processing power to run a Python interpreter, so it's our base platform for CircuitPython (https://adafru.it/BeW).

The SAMD21E

The SAMD21E is a slightly smaller version of the SADM21G, with 32 GPIO pins.



It has most of the same specs as the SAMD21G, but:

Its advanced counters can generate six PWM signals per instance instead of eight
It has four SERCOMs instead of six
Its ADC can read 10 pins instead of 14
Its PTC can handle a 10x6 matrix instead of 12x10

We use it in the Trinket M0 (http://adafru.it/3500) and Gemma M0 (http://adafru.it/3501), where almost none of the GPIO pins or their peripherals are broken out to the

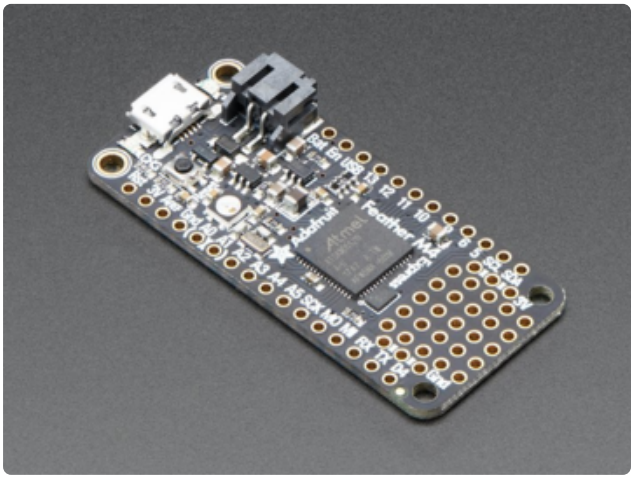
edge of the board. In those forms, they're primarily small CircuitPython (https://adafru.it/BeW) devices.

99% of code that runs on the SAMD21G18 will run on the 'E18 since its just a shrunken down version!

The SAMD51

The SAMD51 is a new device family from Atmel-now-Microchip, built around a very fast ARM Cortex M4 microcontroller. It's a super-powerful upgrade. At this time we have CircuitPython and Arduino support. Hopefully MakeCode will come around too, but as of this writing it isn't yet!

The '51 is an upgrade to the '21 so there are many similarities. Most code for the '21 will run on a '51, but 6x faster. However it is a different core so its not a simple upgrade.



It's more complex and faster than the M0+, and includes a hardware floating-point math unit and digital signal processing array (one command applies a math operation to a whole block of data).

It has 512K of Flash memory and 192K of RAM. It runs at 120MHz and has 51 GPIO pins.

For peripherals, it has:

- six 8/16/24-bit timer instances, each with two PWM outputs
- two 24-bit and three 16-bit advanced counter instances, each with eight PWM outputs
- six SERCOM instances
- two 12-bit ADC instances
- two 12-bit DAC instances
- one I2S interface
- one full-speed (12MBps) USB interface
- one quadrature Position Decoder (for reading rotary encoders)
- one QSPI interface (for using high-speed SPI Flash chips)
- one 10-bit Parallel Capture Controller (for reading video input)
- a set of cryptogtraphy devices including a true-random-number generator, an AES block, and a public key block

- four Configurable Custom Logic instances (basically mini-FPGAs)

It has 32 DMA channels and its PTC can read a 256x32 matrix of capacitive touch sensors.

We use it in the Metro M4 (http://adafru.it/3382), Feather M4 (http://adafru.it/3857), and ItsyBitsy M4 (http://adafru.it/3800), and the NeoTrellis M4 (http://adafru.it/3938).

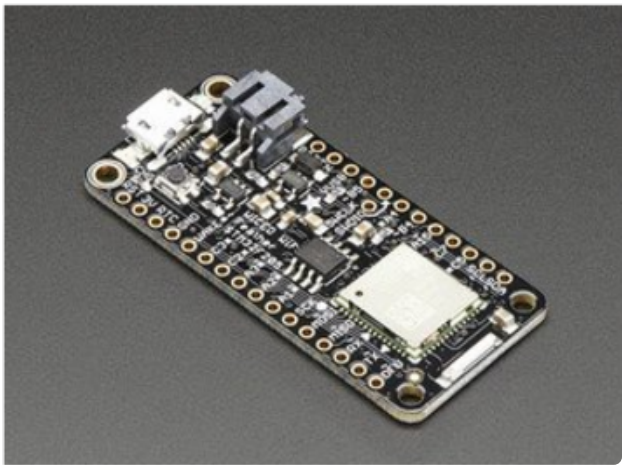
This chip is made for digital signal processing: calculating FFTs of audio input in real time, and generating/mixing audio or video signals. It has features like memory protection that would let it operate as a full-scale computer, but that would be kind of a waste of its abilities. Without the overhead of an operating system, it can outperform a Raspberry Pi Zero for some real-time signal processing.

System-On-a-Chip (SOCs)

The STM32F205

This is the main chip from the WICED Feather (http://adafru.it/3056). Its CPU is a 32-bit microcontroller designed by ST Microelectronics, one of the major chip manufacturers.

It runs at 120MHz and the raw microcontroller has 1MB of Flash and 128K of RAM. You can only program it with Arduino



You can't use most of that though, because the board comes preloaded with the WICED operating system for Wifi controllers. Your code gets 128K of Flash for program storage and 16K of RAM.

For peripherals, it has:

12 GPIO pins
one SPI interface
one I2C interface
three Serial interfaces
one 12-bit ADC
two 12-bit DACs
one Wifi radio

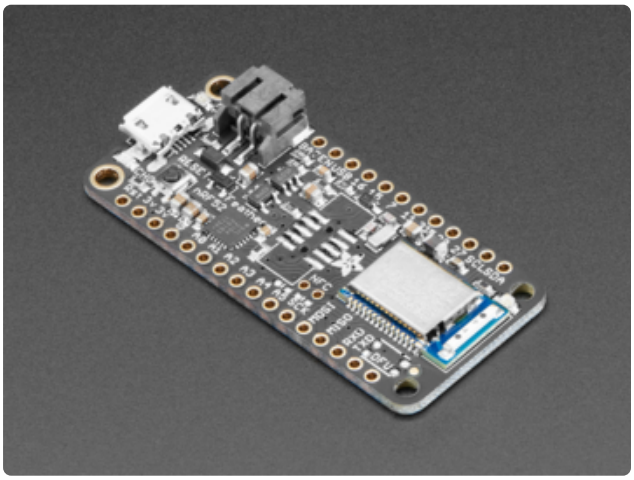
The Wifi interface and WICED operating system are the two biggest features of this board. It was designed to be the core of a Wifi-enabled IoT device.

The nRF52832

This is the main chip from the nRF52 Feather. Its CPU is an ARM Cortex M4 microcontroller running at 120MHz, with a 64MHz floating point unit (FPU).

It has 512K of Flash for program storage, 64K of RAM, and 19 GPIO pins.

This chip has Arduino support and also some CircuitPython support (at the time of this writing, it's Alpha but check to make sure we haven't gotten it to a stable release!)



For peripherals it has:

a 12-bit ADC
three timer instances that can each generate four PWM waveforms
three SPI interfaces
two I2C interfaces
one Serial interface
one I2S audio interface
one Pulse Density Modulation audio input
cryptography devices including a true-random-number generator and an AES block
one quadrature decoder

Those are all relatively minor features though. The big deal for this one is its **built-in Bluetooth Low Energy (BLE) radio**.

The nRF52 is the only microcontroller Adafruit carries that can operate as a BLE central device. Like the other SOCs in this list, it runs a simple operating system that swaps between your code and the built-in firmware that keeps the BLE radio working.

This one is designed to be the core of a BLE device.

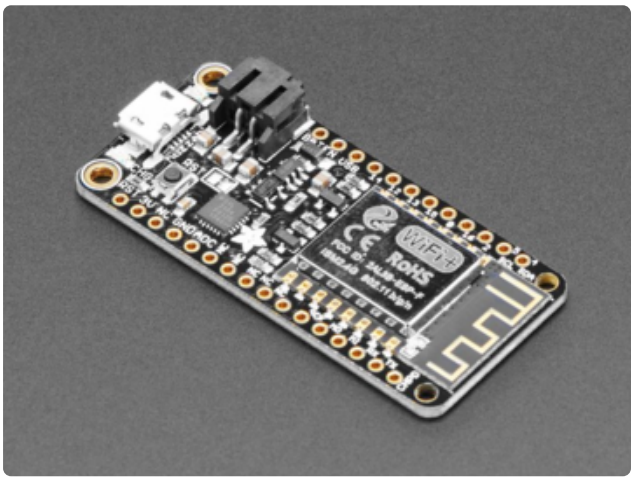
The similar nRF52840 is being designed into next-generation products and looks to be a very capable chip in designs.

The ESP8266

The ESP8266 was something of an internet sensation when it first came out: a microcontroller with built-in Wifi capacity that sold for about $6 in small quantities

(less than half what other non-microcontroller WiFi chips cost at the time). There was a catch though: there was absolutely no public documentation.

It also has the least number of GPIO available, special use GPIO, and a 'hidden' operating system that can make some projects and use difficult.



It didn't take long for the combined curiosity of the internet community to unravel most of the chip's mysteries. Then Espressif, the company that makes the ESP8266, began to talk to that community and release official documentation. It may go down as one of the most effective reverse-psychology viral marketing campaigns in history.

To reach its amazingly low price point, Espressif sacrificed almost everything else. The microcontroller part is a 32-bit Tenselica L106 Diamond architecture running at 80 MHz.

It has nine GPIO pins, three of which control the way the microcontroller behaves when it boots. User code gets less than 36K of RAM, but the chip has a whopping 4MB of Flash memory.

For peripherals, it has:

- one SPI interface
- one I2C interface
- one I2S interface
- one serial interface
- four PWM channels
- one 10-bit ADC that can read values between 0V and 1V

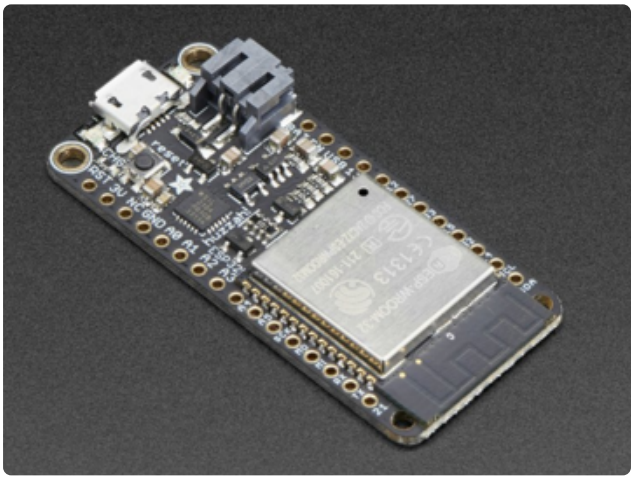
The big deal for this one is, obviously, its WiFi interface. This chip is made to be the core of simple WiFi IoT devices that don't need to make many physical connections to other devices.

The CPU has enough processing resources to run a Python interpreter. Adafruit carries the Feather HUZZAH with ESP8266 (http://adafru.it/2821) The 4MB Flash

makes the ESP8266 a suitable platform for CircuitPython. But, if CircuitPython is a primary consideration, Adafruit recommends more capable chips like the M0 boards for the best experience.

The ESP32

The ESP32 is the most recent chip from Espressif, who also developed the ESP8266.



It's designed to be a one-stop-shopping solution for IoT devices.

The microcontroller is a single or dual core Xtensa L6 32-bit microcontroller, running at 160 MHz or 240 MHz.

It uses an external fast SPI Flash chip for program storage (up to 16MB, Adafruit uses a 4MB chip). It has 540K of RAM and 34 GPIO pins.

For peripherals, it has:

- one 16-bit PWM generator with 16 output channels
- three Serial interfaces
- three SPI interfaces
- three I2C interfaces
- one 12-bit ADC that can read any of 18 pins
- two 8-bit DAC instances
- two I2S interfaces
- ten capacitive touch sensor instances
- temperature and magnetic field sensors
- one 8-channel IR interface
- one 10/100 Mbps Ethernet MAC
- one SDIO host interface that can read/write data from an SD card
- one SDIO secondary interface that can act like an SD card
- eight pulse counter instances, each of which can capture 4 signals

But the headline features are WiFi, Bluetooth Classic, and BLE radios. The dual-core processor means the chip doesn't have to spend as much time swapping between the code you upload and the built-in OS that keeps the radios running.

It's an ambitious design, but still a work in progress. There are lots of pieces, and sometimes they interact in unexpected ways. Espressif is finding, fixing, and adjusting those issues, but it's a process of testing options to see how they work in the field. That's normal for any new chip, but Epressif is doing it in public.

Adafruit carries the Espressif ESP32 Developers Board (http://adafru.it/3269), the Adafruit HUZZAH32 – ESP32 Feather Board (http://adafru.it/3405), and the Pycom SiPy 1.0 – ESP32 WiFi, BLE and +22dBm SigFox Radio (http://adafru.it/3534).

Simple Boards

Simple is Good

Engineers like simplicity. It has fewer ways to fail. A simple solution that works for a wide range of conditions is worth its weight in gold.. literally, if you add up the development, maintenance, and repair costs.

Simple, widely-used solutions have a rich engineering history: whatever you want to do, someone has probably done it before. You can learn from their months or years of experience instead of having to do it all on your own.

That doesn't mean learning new hardware or exploring new solutions is bad -- engineering history has to come from somewhere after all -- it just means exploration is a choice.

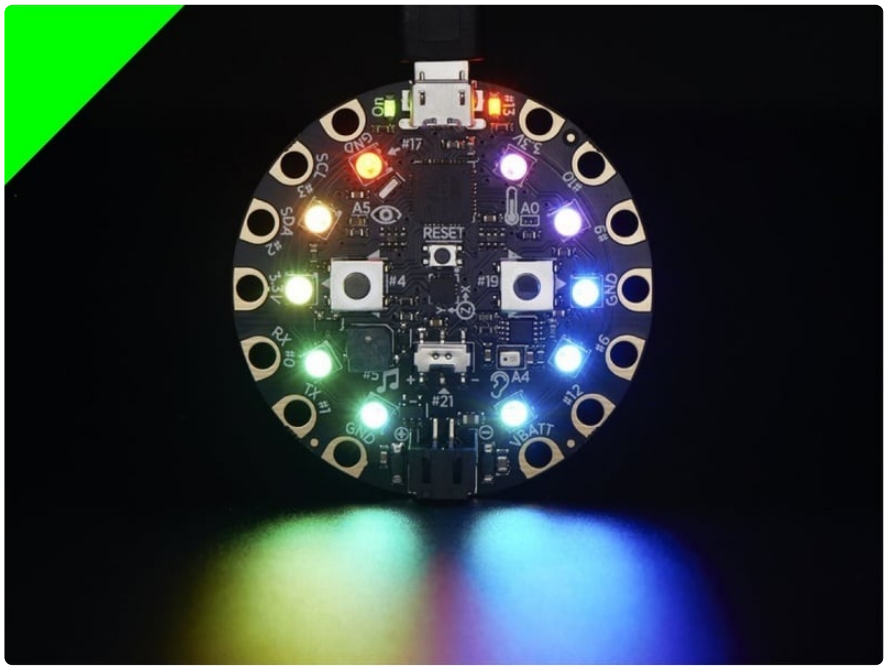
If you want to learn new hardware for the sake of learning new hardware, the failures and dead ends will be more interesting to you than the things that work on the first try. Simple boards probably won't be as interesting as more complex ones unless you're new to microcontrollers in general.

When you want to do a specific job though, exploration tends to be more of a nuisance than a reward. For those projects, you want the simplest board you can find.

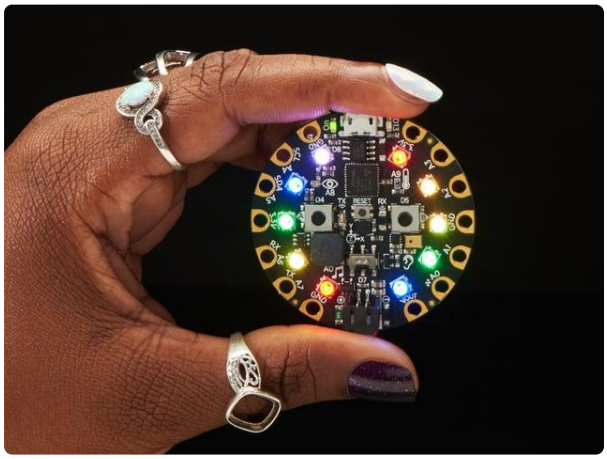
Where do I Start?

I want to learn microcontrollers, but don't want to buy a lot of extra stuff

Start with the Circuit Playground Express:



With the Circuit Playground, we've built the engineering history for you. It has a set of input and output devices that we've seen used most often, and we've written code to help you get started. Its job is to help you get familiar with the kind of code you'll use with any other microcontroller.



Circuit Playground Express

Circuit Playground Express is the next step towards a perfect introduction to electronics and programming. We've taken the original Circuit Playground Classic and...

https://www.adafruit.com/product/3333

You can use it with Arduino, CircuitPython, MakeCode and Code.org CS Discoveries, it has just about everything you want built in: sensors, lights, alligator clip pads. It's perfect for your first microcontroller board!

Resources

There are excellent tutorials for the Circuit Playground Express, free, in the Adafruit Learning System (https://adafru.it/dlu). The Introductory Guide (https://adafru.it/adafruit-cpx) is the place to start and then you can explore 100+ projects which use the board.

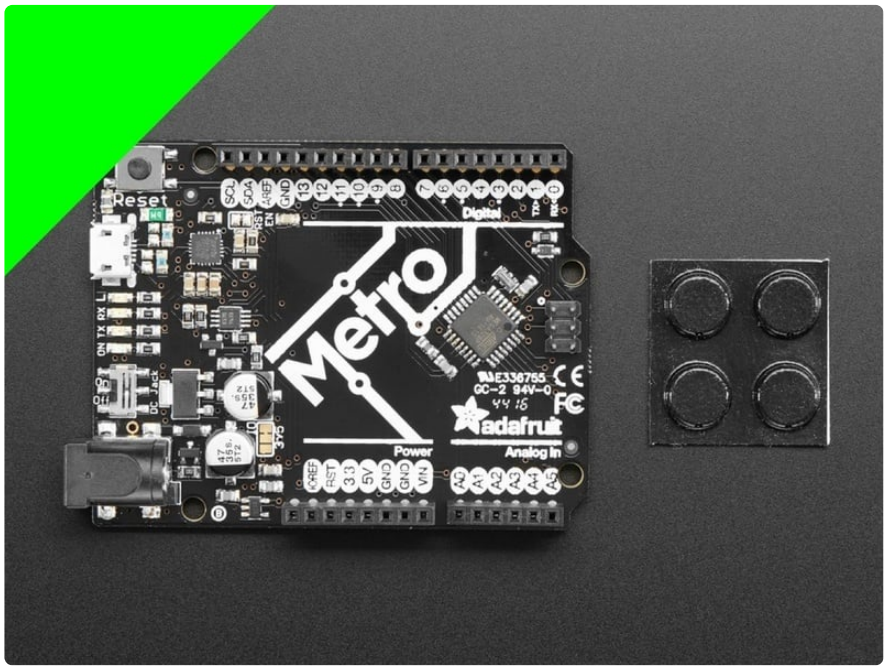
If you like physical books, Adafruit team member Mike Barela has recently published Getting Started with Adafruit Circuit Playground Express (https://adafru.it/CZf) which goes through using the Circuit Playground Express step by step.

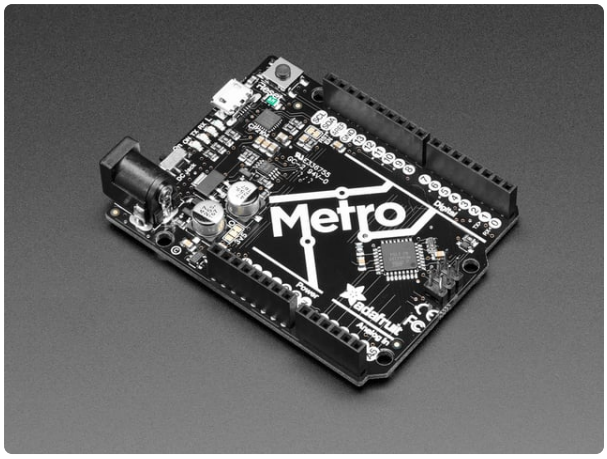
Arduino 328 Compatibles

If you are making something that is going to run existing Arduino UNO code, its handy to have a compatible board. These are your options that use the exact same chip, so you know it will work 100% the same

I want a microcontroller that is Arduino-Compatible

Start with the Metro 328P:





Adafruit METRO 328 - Arduino Compatible - with Headers

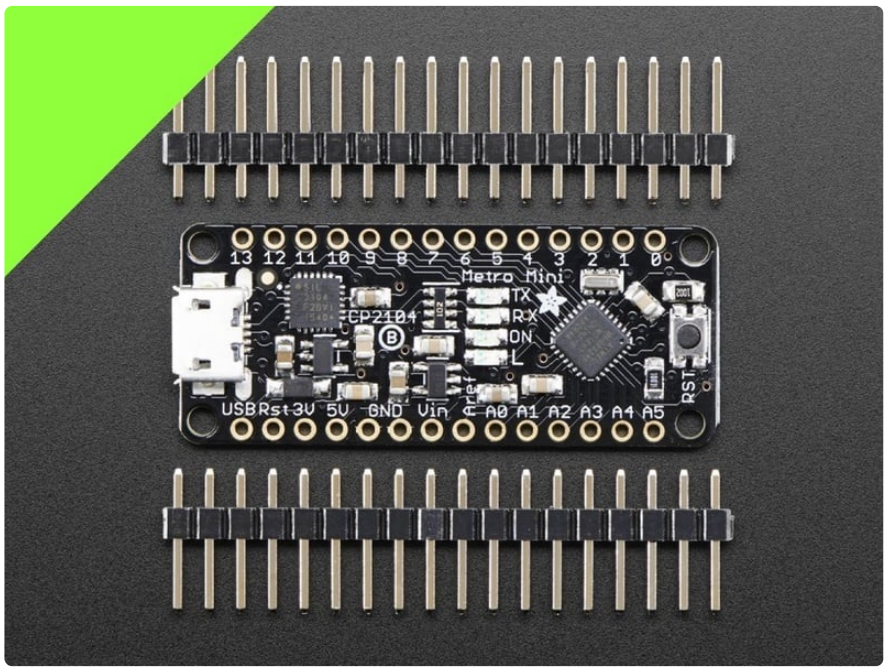
This is the Adafruit METRO Arduino-Compatible - with headers. It's a fully assembled and tested microcontroller and physical computing board with...

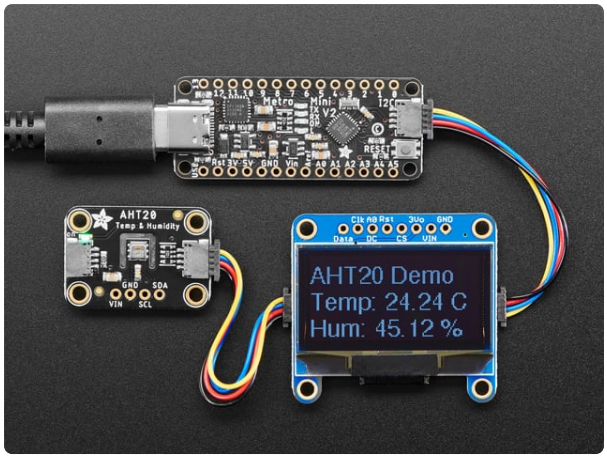
https://www.adafruit.com/product/2488

The Metro uses the same circuit design as the Arduino Uno, which has a huge engineering history. Its physical dimensions and headers make it good for breadboarding on the workbench.

I want to build an Arduino-compatible microcontroller into a project

Start with a Metro Mini:





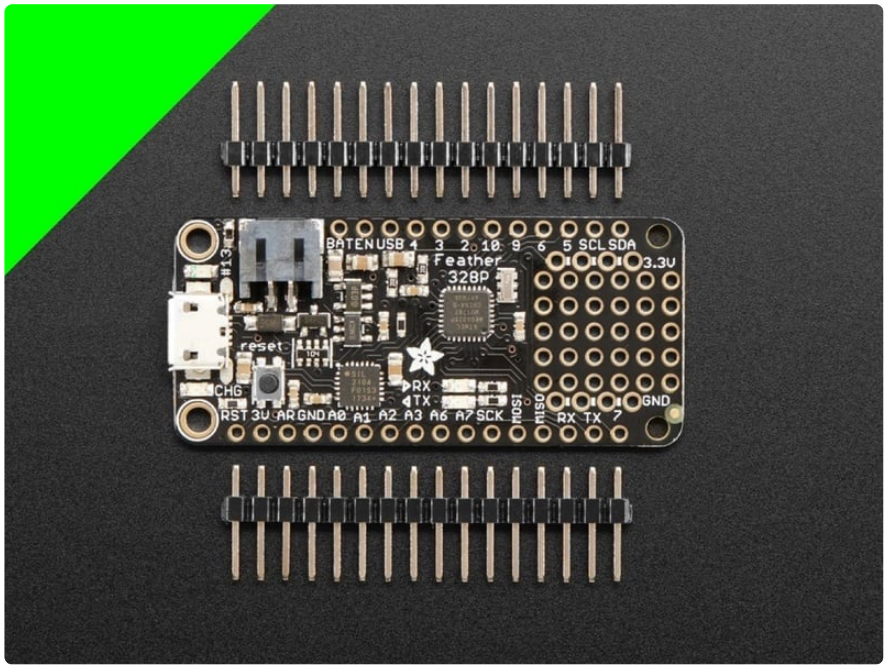
[Adafruit Metro Mini 328 V2 - Arduino-Compatible - 5V 16MHz]
One of our star development boards is the Adafruit METRO Mini 328, an excellent lil fellow that lets you make your Arduino-based project...
https://www.adafruit.com/product/2590

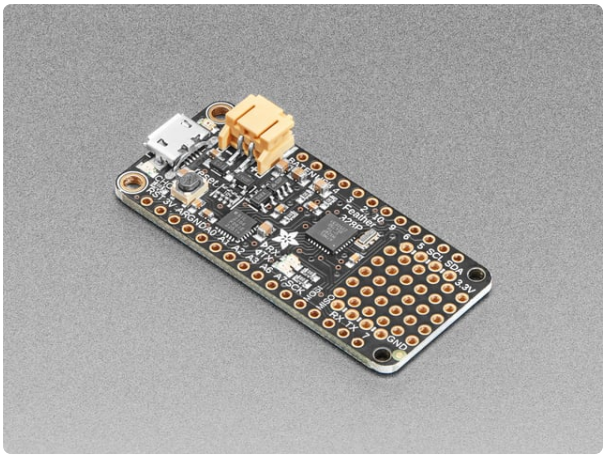
The Metro Mini has all the same parts and pin connections as its bigger sister, but is easier to build into a project.

You can design and test your project on the workbench using the Metro 328, then move the code and connections to the Metro Mini for the final product.

I want to build a battery-powered device

Consider the Feather 328P





[Adafruit Feather 328P - Atmega328P 3.3V @ 8 MHz](https://www.adafruit.com/product/3458)
With this Feather we're getting a little nostalgic for the ATmega328P - the classic 'Arduino' chip - with this Adafruit Feather 328P running a 3.3V and 8 MHz. Feather is...
https://www.adafruit.com/product/3458

The Feather 328P uses the same microcontroller as the Metro 328P and Metro Mini, but uses the footprint (the set of IO pins and how they're arranged) from our Feather line of development boards.

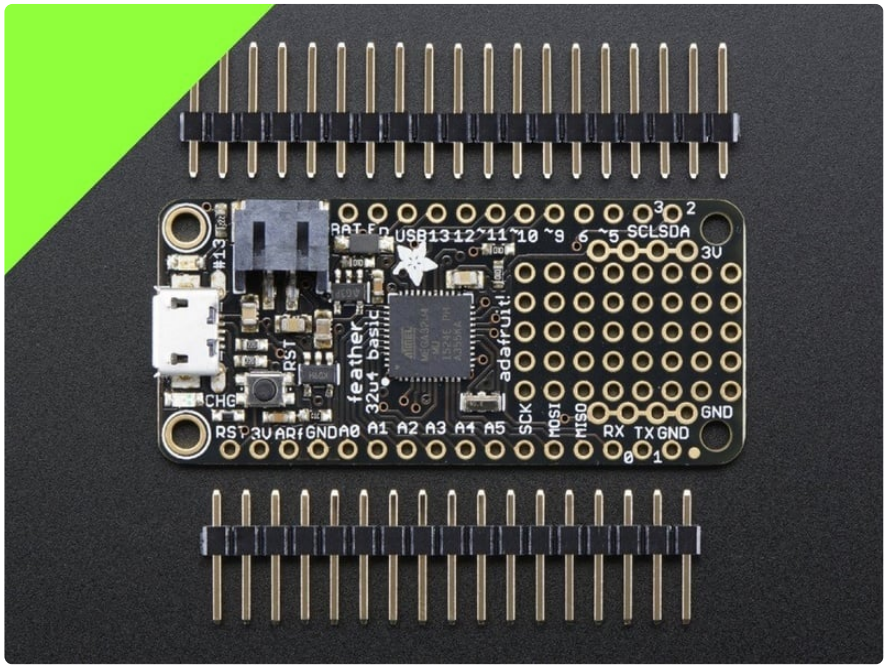
All Feathers can run from a rechargeable LiPo, and have built-in chargers to refill the battery when you plug in a USB cable.

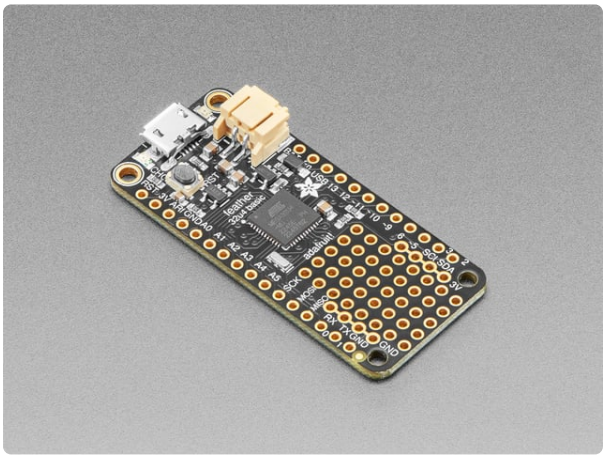
The Feather footprint makes it easier to move from one microcontroller to another as a project evolves, and we have FeatherWing add-on boards (https://adafru.it/BC6) to make connecting other devices simple.

The Feather 328P doesn't have all the pins available, and it runs at 3.3V 8MHz but it's the same chip so you can expect it to just be a little slower

Next Step - 32u4 Boards

The Feather 32u4 Basic:





Adafruit Feather 32u4 Basic Proto

Feather is the new development board from Adafruit, and like its namesake it is thin, light, and lets you fly! We designed Feather to be a new standard for portable...

https://www.adafruit.com/product/2771

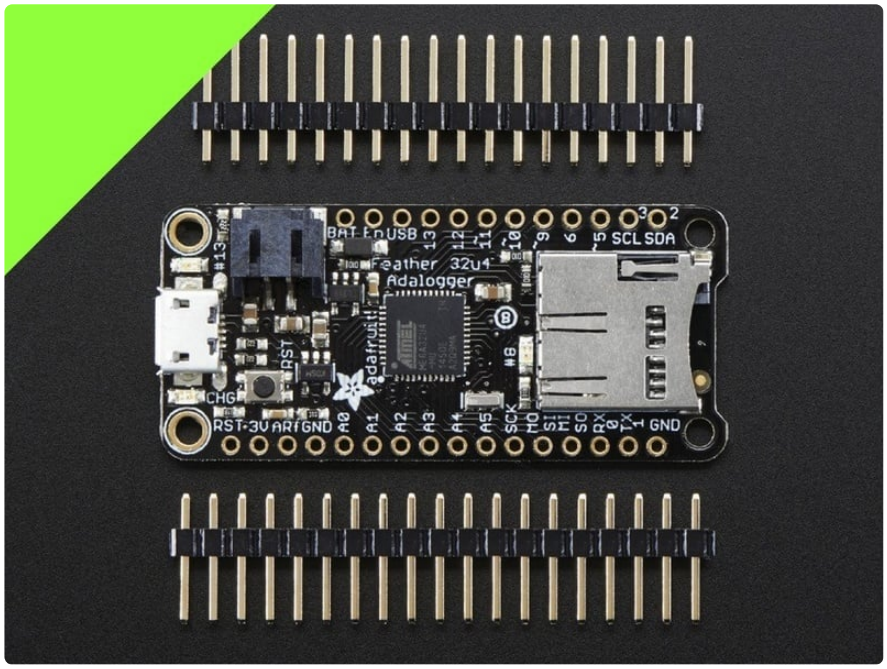
The ATmega32u4 belongs to the same family of 8-bit microcontrollers as the ATmega328P, but handles USB communication internally. The 328P doesn't know how to talk to a USB cable, so boards that use the 328P need a USB-to-Serial converter for programming.

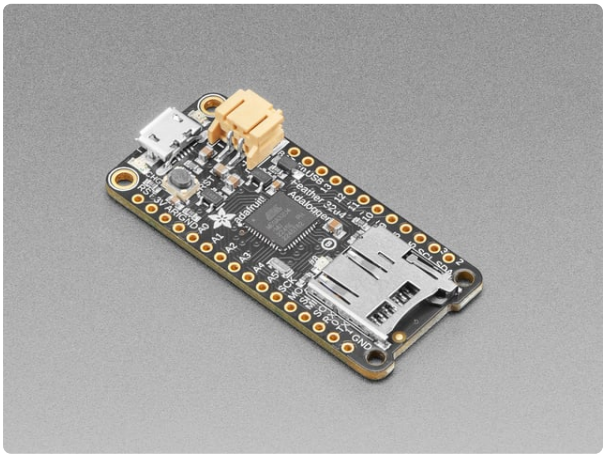
Controlling its own USB connection makes the 32u4 a bit more flexible than 328P-based boards. You can program it to act like a keyboard, mouse, or MIDI controller when it connects to a computer. 328P-based boards can't do that.

Most code that runs on a 328P Feather will also run on a 32u4 Feather, but you might have to make some minor changes here and there.

The Circuit Playground Classic also uses an ATmega32u4.

The Feather 32u4 Adalogger:

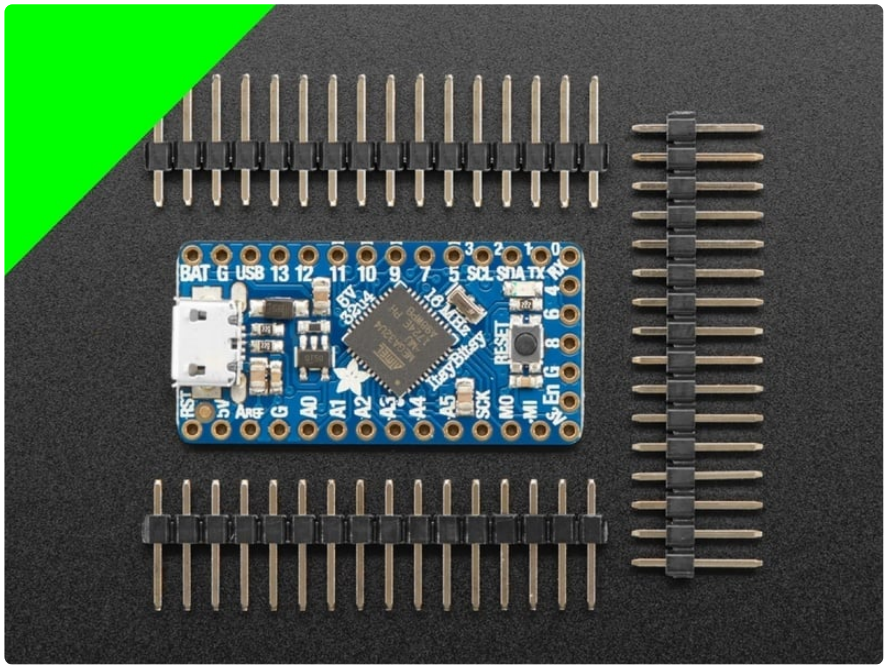




[Adafruit Feather 32u4 Adalogger](https://www.adafruit.com/product/2795)
Feather is the new development board from Adafruit, and like its namesake it is thin, light, and lets you fly! We designed Feather to be a new standard for portable microcontroller...
https://www.adafruit.com/product/2795

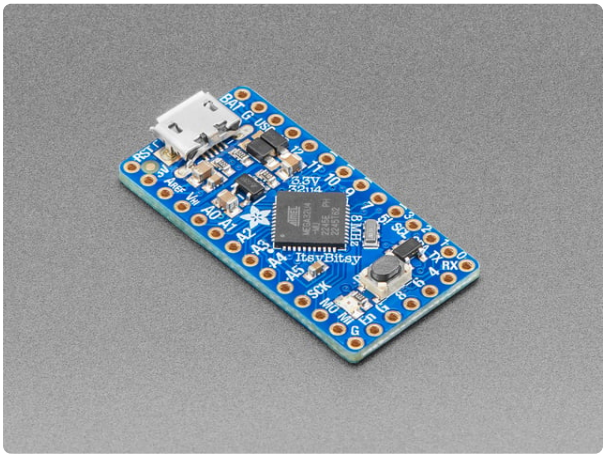
The Adalogger has a built-in SD card socket, which makes it perfect for projects where you want to collect data.

The ItsyBitsy 32u4:

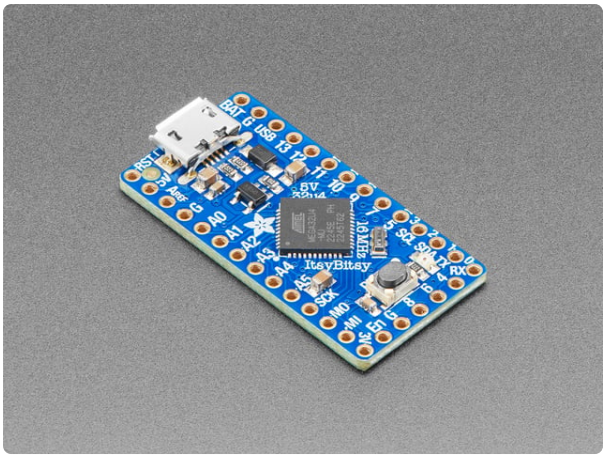


The ItsyBitsy 32u4 uses the same microcontroller as a Feather 32u4, but breaks out more GPIO pins than the Feather. It's best for embedded projects where you want the features of the ATmega32u4 microcontroller but need more pins than are available on the Feather.

There are two versions, one that is 8MHz 3.3V and one that is 16MHz 5.0V



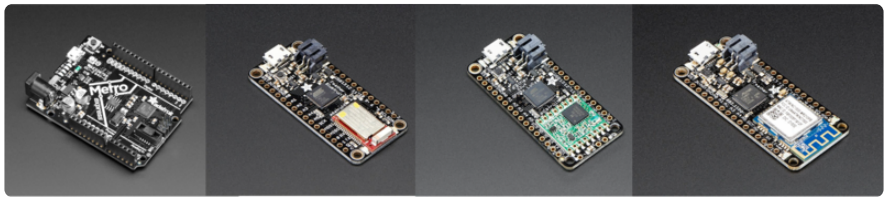
Adafruit ItsyBitsy 32u4 - 3V 8MHz
What's smaller than a Feather but larger than a Trinket? It's an Itsy Bitsy! Small, powerful, Arduino-compatible - this microcontroller board is perfect when you want something...
https://www.adafruit.com/product/3675



Adafruit ItsyBitsy 32u4 - 5V 16MHz
What's smaller than a Feather but larger than a Trinket? It's an ItsyBitsy! Small, powerful, Arduino-compatible - this microcontroller board is perfect when you want something...
https://www.adafruit.com/product/3677

Intermediate Boards

Branching Out

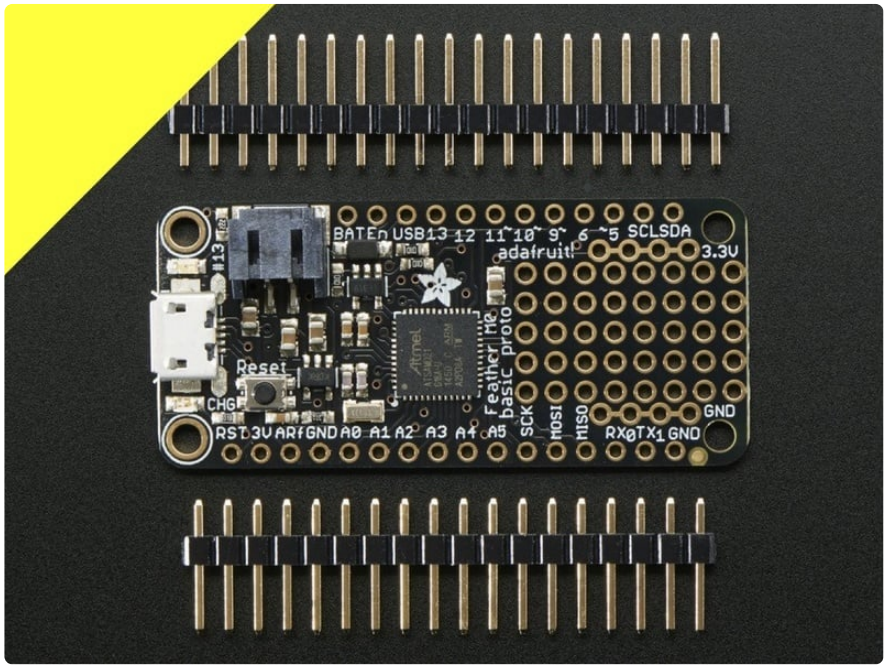


Once you've gotten comfortable with simple microcontrollers, you'll be ready to learn boards that have a few more features or are tuned for a specific kind of use.

Intermediate boards aren't that much harder to use than the simple ones, but they tend to be more specialized. They work well in some projects, but can get in your way for others. That means you need to know more about their strengths and limits when choosing one.

32-bit Boards

The Feather M0 Basic:



The Feather M0 series of development boards use the SAMD21G microcontroller, and the actual microcontroller part is an ARM Cortex M0+, one of the simplest 32-bit processor designs.

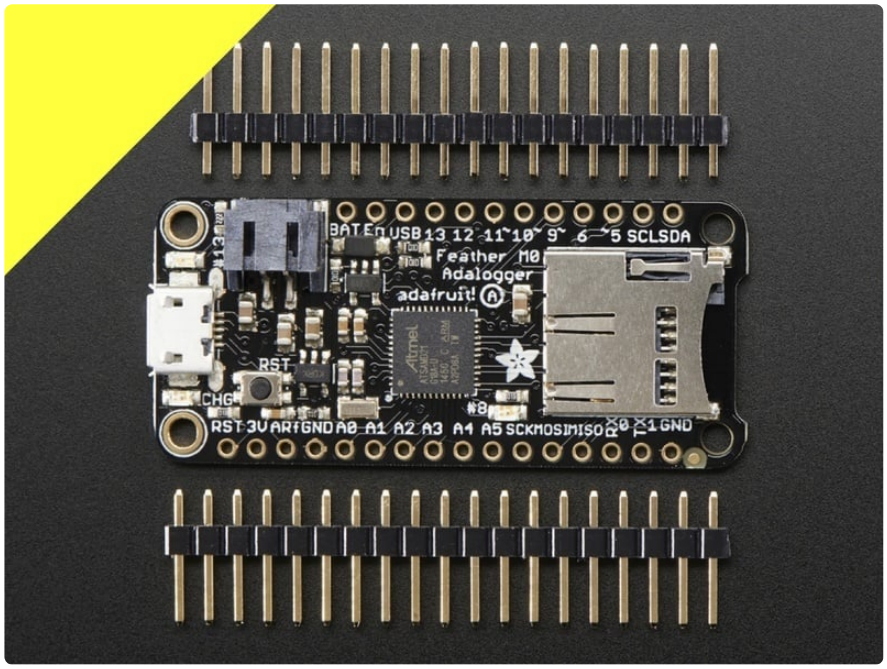
The Adafruit SAMD Board Support Package defines all the usual convenience functions from the Arduino environment, like digitalWrite() and analogRead(), and if you stick to those, you shouldn't see much difference between the M0 and an 8-bit board like the Metro 328P. The biggest differences you'll see are faster clock speed (48MHz), more program memory, and more RAM.

The SAMD21G has all the peripheral-bus-and-multiplexer features of a standard 32-bit device though, so if you want to start digging around under the hood, you'll have some reading to do.

That said, the SAMD21G is a good place to start learning the differences between 8-bit microcontrollers and 32-bit microcontrollers.

The Feather only uses a few of the SAMD21G's IO pins (there are about 40 of them), so it isn't a general purpose SAMD21G development board. It's limited to fit into the Feather ecosystem.

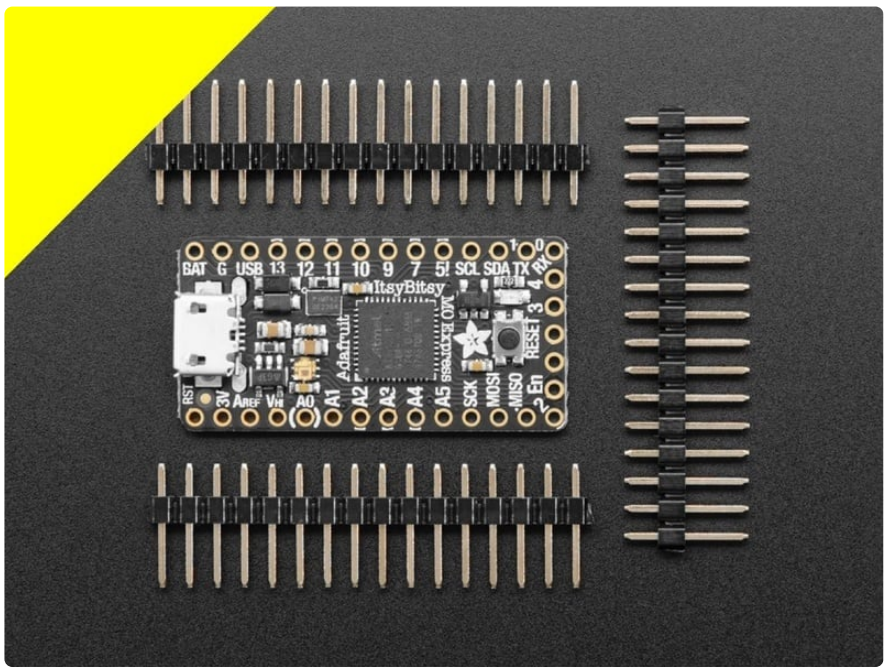
The Feather M0 Adalogger:



The Feather M0 Adalogger has the same microcontroller and pinout as the Feather M0 Basic, but there's a built-in slot for an SD card. As the name suggests, it's good for applications where you want to collect data.

The SD card is connected to the SPI pins at the edge of the board, so you need to be aware of that if you want to connect other SPI devices to the MOSI/MISO/SCK pins at the side of the board.

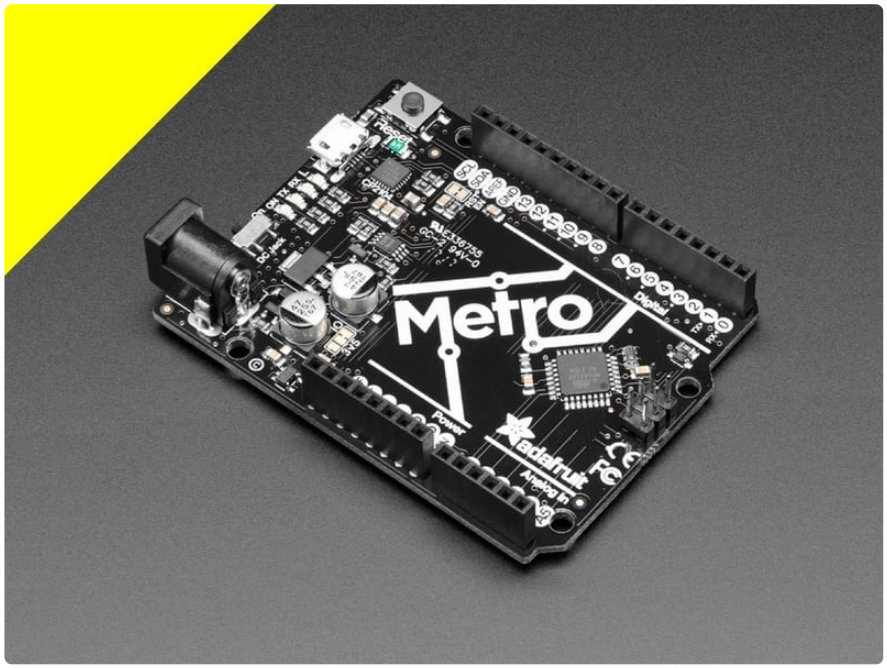
The ItsyBitsy M0:



The ItsyBitsy M0 also uses the SAMD21G, but brings more of the chip's IO pins out to the edge of the PCB. On the other hand, it doesn't have the Feather's built-in LiPo charger.

This one is more suited to embedded projects where you need pins more than portability.

The Metro M0:



What can we say? The Arduino Uno footprint has become a standard, but people keep wanting to do things the Uno's 8-bit microcontroller can't handle.

The Metro M0 gives you the power of a SAMD21G with the breadboard-friendly footprint of the Uno.

One point to note: different chips have different pin connections, so the Metro M0's SPI pins **do not** connect to digital pins 10-13 line on the Uno. They only connect to the 2x3 header near the microcontroller.

CircuitPython Boards

When you program an Arduino Uno in C or C++, the compiler takes the code you've written and turns it into chunks of machine code the microcontroller's processor can execute. The firmware you upload to the microcontroller has direct control of the processor.

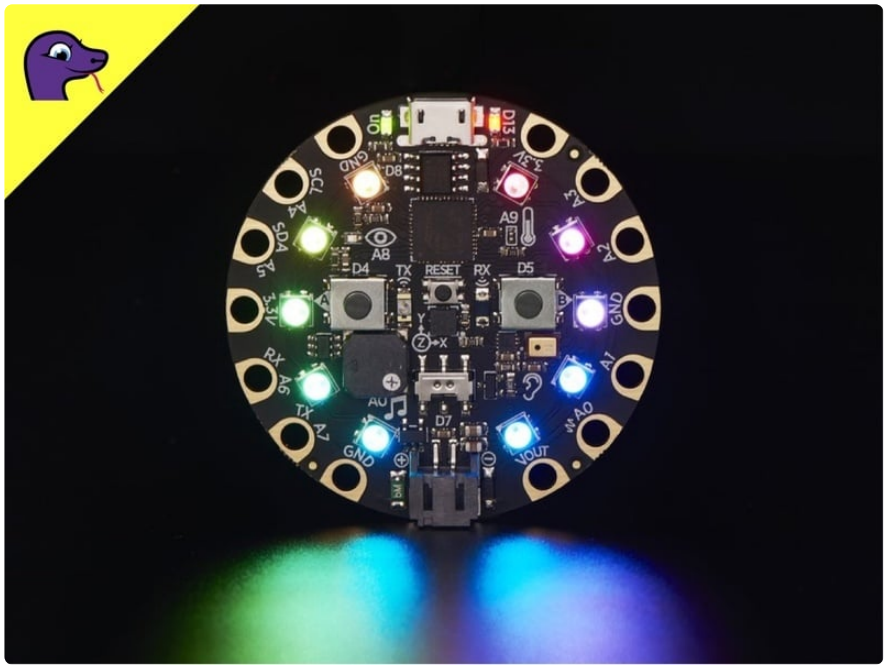
Python is different. It's an interpreted language, meaning the processor runs a program that speaks Python. The code you write doesn't have direct control of the processor, it's more like an elaborate set of button-presses that tell the code actually controlling the processor what to do.

Interpreted languages have some strong advantages over compiled languages: you can enter code on the fly and see the interpreter process the instructions as you type, and they handle a lot of nitpicky details like memory management so you don't have to.

There are always trade-offs though. An interpreted language might have to execute several hundred processor instructions for every line of code you type, and the behind-the-scenes stuff like memory management also takes work. You can assume that interpreted code will always be at least 10x slower than compiled code, and will require about 10x more programming resources like RAM and program storage.

The SAMD21G has enough RAM to run a simple Python interpreter, and we gave our Express boards a Flash chip where they can store code. The result is a set of boards you can program in Python on the fly.

The Circuit Playground Express:

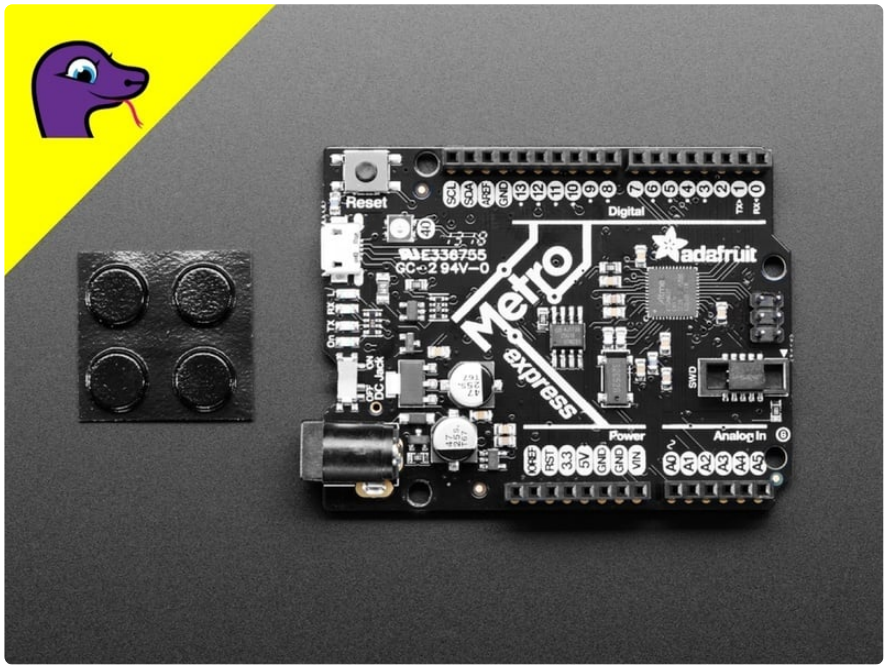


The Circuit Playground Express (http://adafru.it/3333) is the flagship of our Python-programmable line.

It's designed to be a lots-of-interesting-stuff-with-no-wiring-necessary learning tool like the Circuit Playground Classic, but it also has the microcontroller and external Flash that a Python interpreter needs.

We've written the low-level code that lets the Python interpreter talk to the hardware, and we've installed the UF2 bootloader (the program that lets a chip upload its own code). That makes getting Python code and libraries from your computer to the Flash chip as easy as dragging and dropping files on a USB thumb drive.

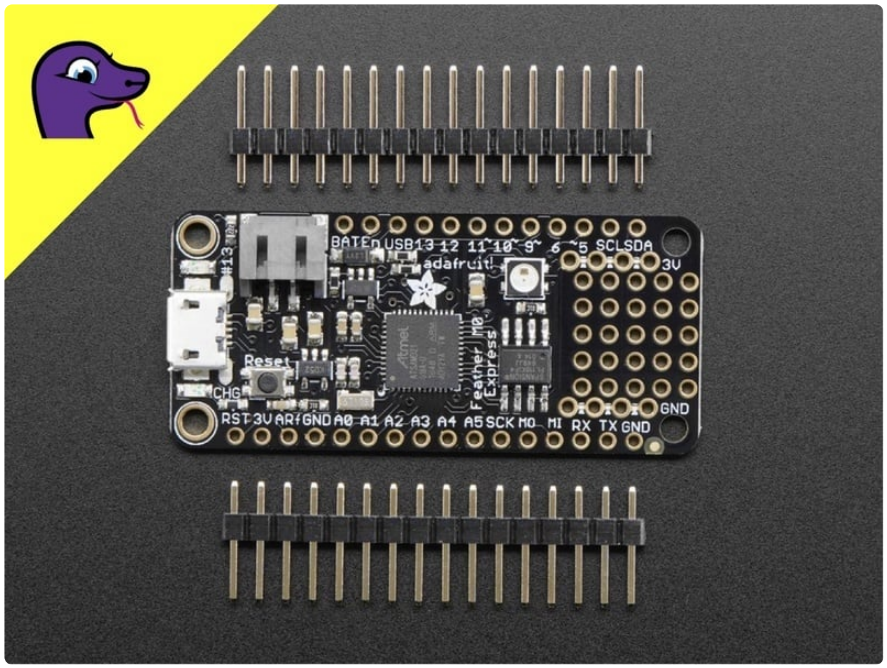
The Metro M0 Express:



All the Python in the Uno footprint.

The Metro M0 Express (http://adafru.it/3505) doesn't have all the connected devices of a Circuit Playground Classic, but is more friendly to breadboarding on the workbench or existing projects that are already designed for an Uno.

The Feather M0 Express:

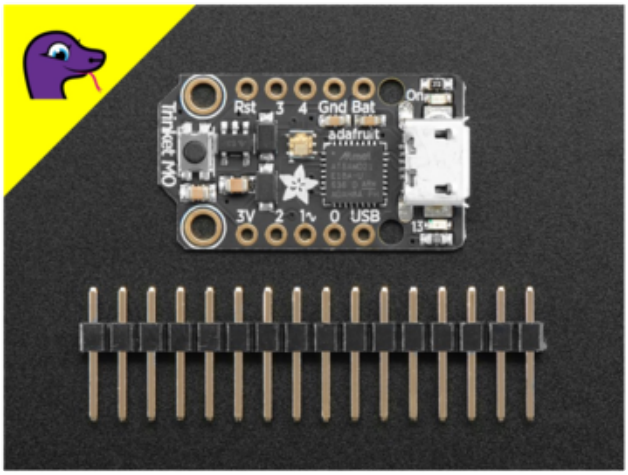


With this one, you can take your Python for a walk.

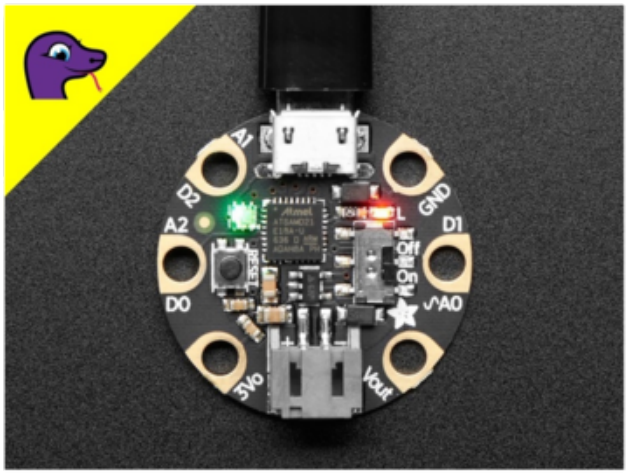
The Feather M0 Express (http://adafru.it/3403) is good for battery-powered projects and is compatible with all the boards in the Feather ecosystem.

The Trinket M0 and Gemma M0

Frankly, we designed these to replace the ATtiny85 Trinket and Gemma.



People would buy a Trinket or Gemma thinking it was a cheap and easy way to get started in microcontroller programming, not knowing that those boards were more of a challenge than a larger board like the Metro 328P. The result was an ongoing mismatch between expectations and experience.



The new M0 versions use a SAMD21E microcontroller, which is a bit smaller and more limited than the SAMD21G, but is far more capable than the ATtiny85. They handle their own USB communication and their GPIO pins are easier to work with.

They're still special-purpose boards best suited to embedded projects where space is limited, but they do work as cheap and simple entry-level boards.

Note: The book Getting Started with Adafruit Trinket does not cover the Trinket M0 or the Gemma M0.

BLE Boards

If you want a board to communicate with a cell phone or possibly a computer, you may want to consider a board which includes Bluetooth.

If you want wireless communication **between** microcontrollers, **you almost certainly do not want BLE**.

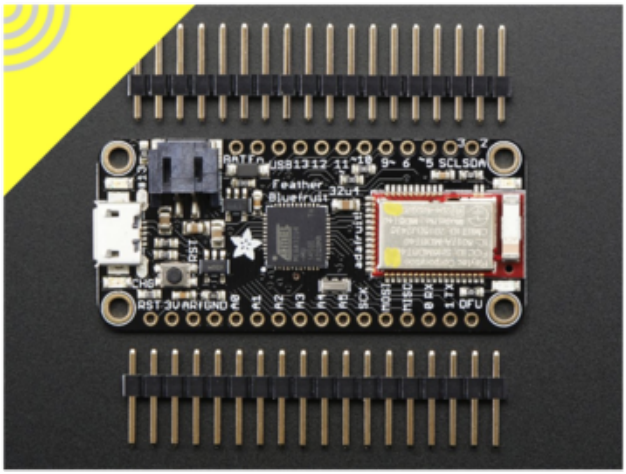
Bluetooth splits devices into 'central' and 'peripheral' categories. Central devices initiate and control all data connections, peripherals just wait for a central to tell them what to do. That arrangement puts most of the cost and complexity in the central device, allowing peripherals to be smaller, simpler, less expensive, etc.

There are three important facts to know about BLE when you're choosing a board for a project:

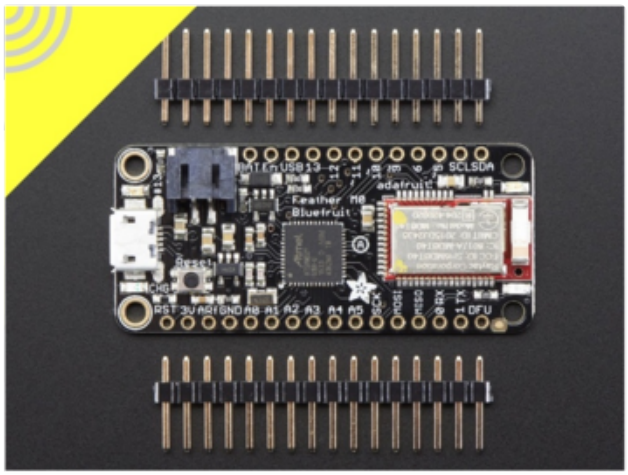
1. **BLE peripherals can not talk to each other. Period.**
2. **BLE has a maximum range of about 10m.**
3. **The best-case data rate for BLE is between 5 and 20 kilobytes per second**

The software stack for a BLE central is too big for the microcontrollers on this page. These boards are only BLE peripherals, and can't talk to each other.

nRF51822 Boards



The Bluefruit Feather 32u4 (http://adafru.it/2829) pairs the Adafruit Feather 32u4 with a BLE module. This chip has 32K of flash and 2K of RAM, with built in USB so not only does it have a USB-to-Serial program & debug capability built in with no need for an FTDI-like chip, it can also act like a mouse, keyboard, USB MIDI device, etc.



The Adafruit Feather M0 Bluefruit LE (http://adafru.it/2995) adds an ATSAMD21G18 ARM Cortex M0 processor, clocked at 48 MHz and at 3.3V logic, the same one used in the new Arduino Zero (http://adafru.it/2843). This chip has a whopping 256K of FLASH (8x more than the Atmega328 or 32u4) and 32K of RAM (16x as much)! This chip comes with built in USB so it has USB-to-Serial program & debug capability built in with no need for an FTDI-like chip.

Packet Radio Boards

If you want wireless communication between microcontrollers, you almost certainly want boards with integrated packet radio.

The RFM69 modules have a working range of 100m with a clear line-of-sight connection. The RFM95 LoRa modules work out to about 500m under normal conditions, and have been tested at 2km with focused antennas.

The data rate for both kinds of radio is about 19.2kbps (about 1.8 kilobytes per second).

Adafruit makes processor boards with on-board radios that work in the 433 MHz and 900 MHz bands.

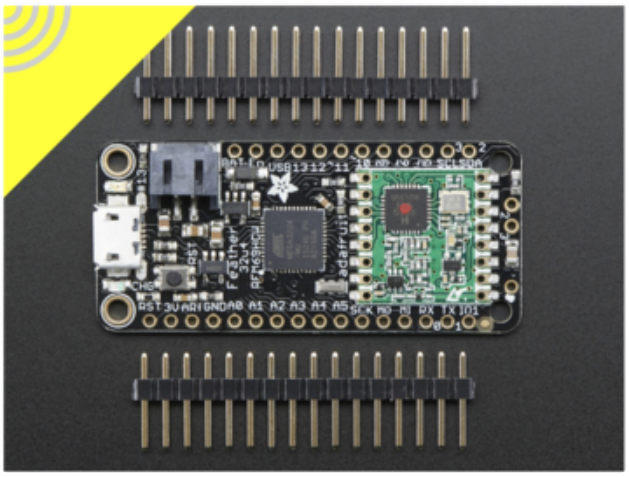
In general, higher frequencies are more prone to scattering and lose more energy passing through objects, so 433MHz radios are a bit better at long-distance connections and connections where you don't have a good line of sight from one antenna to the next.

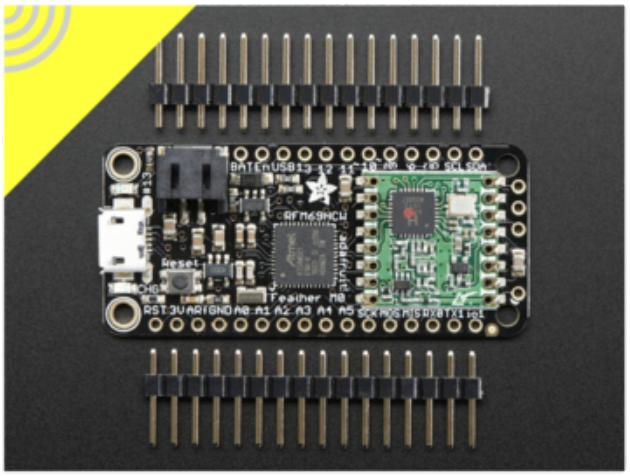
433MHz radios are more prone to interference from high-speed USB signals (operating at 480MHz) though, and high frequencies make it easier to modulate signals. That makes the 900MHz radios perform a bit better in areas where there are a lot of other signals.

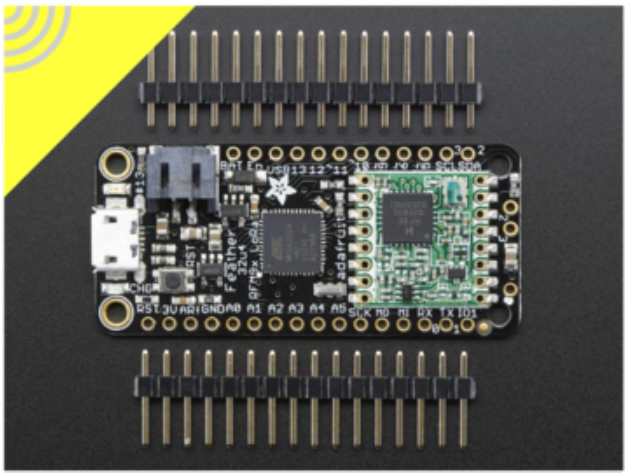
Adafruit has Feather boards with ATmega32u4 and SAMD21 microcontrollers for both frequencies and both kinds of modulation.

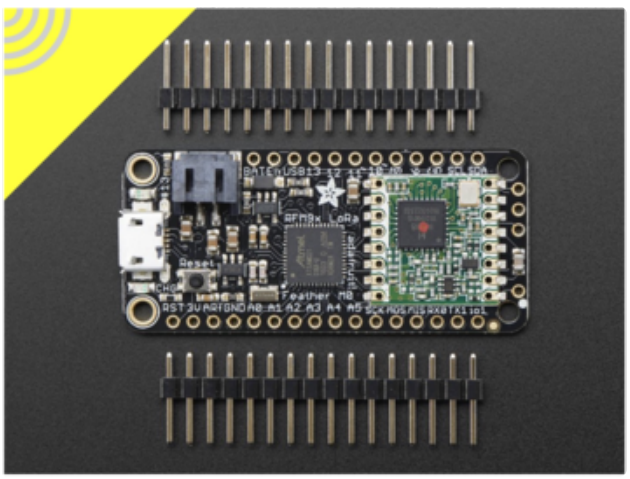
Double check the legal broadcast frequencies for data radios in your country. They may vary and you do not want to use radio signals in a frequency band where the authorities do not want you communicating or in a band used by cellular or emergency services only.

IMPORTANT NOTE: The packet radio modules use SPI to communicate with the microcontrollers. On these Feathers, the radio module's CS pin is tied to GND with a pull-down resistor so the radio is active by default. If you want to use another SPI device, like an SD card, YOU MUST SEND GPIO PIN 8 HIGH FIRST!





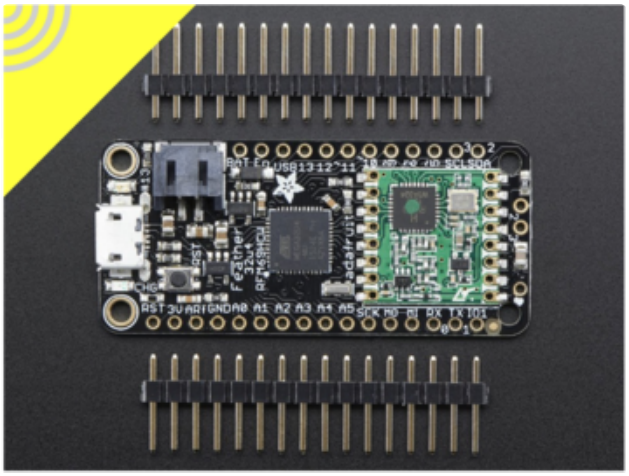


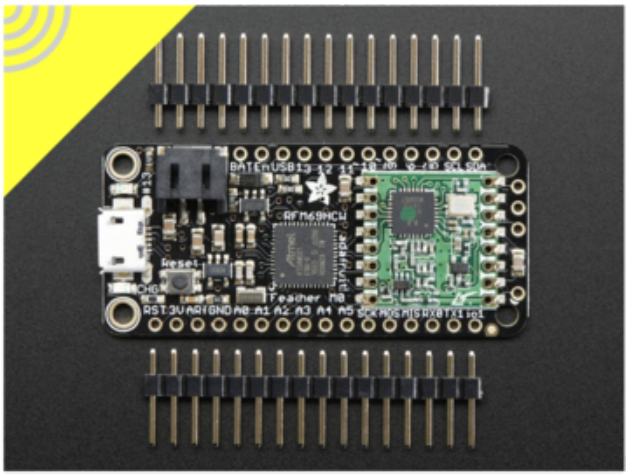


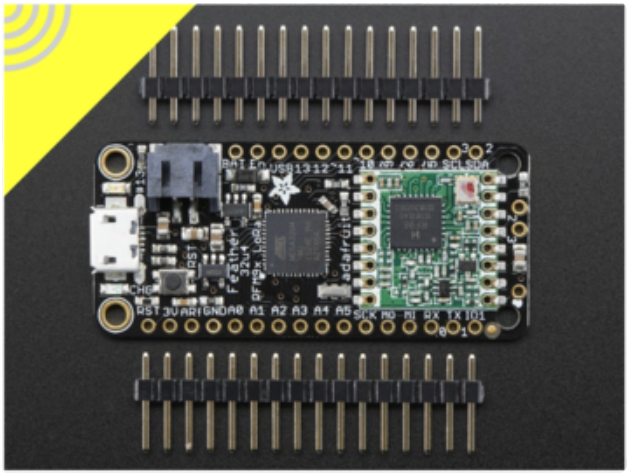
433 MHz Boards

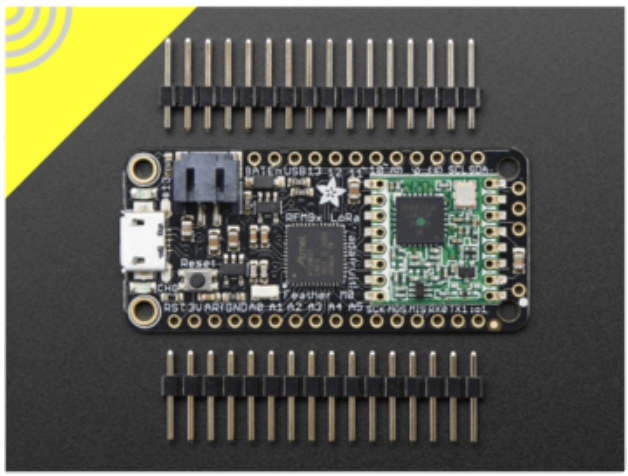
In general, higher frequencies are more prone to scattering than low frequencies, and lose more energy passing through objects.

433MHz radios are a bit better for long-distance connections and connections where you don't have a good line of sight from one antenna to the next.









900 MHz boards

High frequencies make it easier to modulate signals, and to identify signals in a noisy environment.

900MHz radios perform a bit better in environments with interference, especially signal leakage from 480MHz high-speed USB signals.

WiFi Boards

If you want to make a device that connects to the internet, you probably want WiFi.

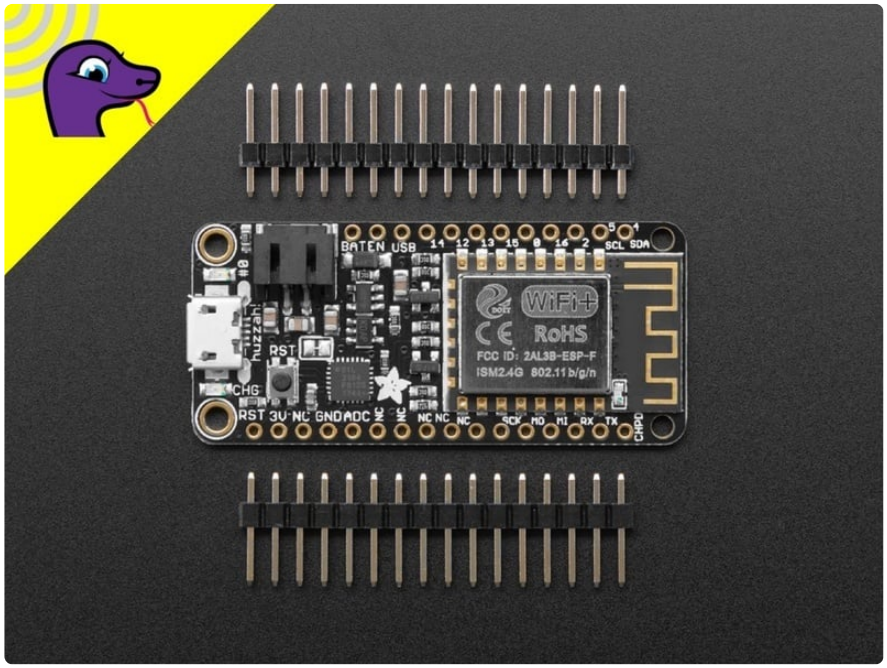
Back in the 1950s, air conditioners were new enough that stores, restaurants, and movie theaters would post signs in the summer to let people know they had one. It drew people in, and was called 'advertising air'.

Here in the 20-teens, we have 'advertising WiFi'. Many businesses offer WiFi to lure customers, making shopping more 'frictionless' and creating atmosphere.

Go to a hotel without WiFi: never!

Free wireless connections to the internet have become common enough that devices to use them are becoming popular. The boards below are good choices for portable, internet-connected projects.

The Feather HUZZAH with ESP8266:



The ESP8266 was made for WiFi, and is extremely popular in that space. It has a 32-bit microcontroller, 4MB of Flash for program and data storage, and a built-in operating system that keeps the Wifi radio running.

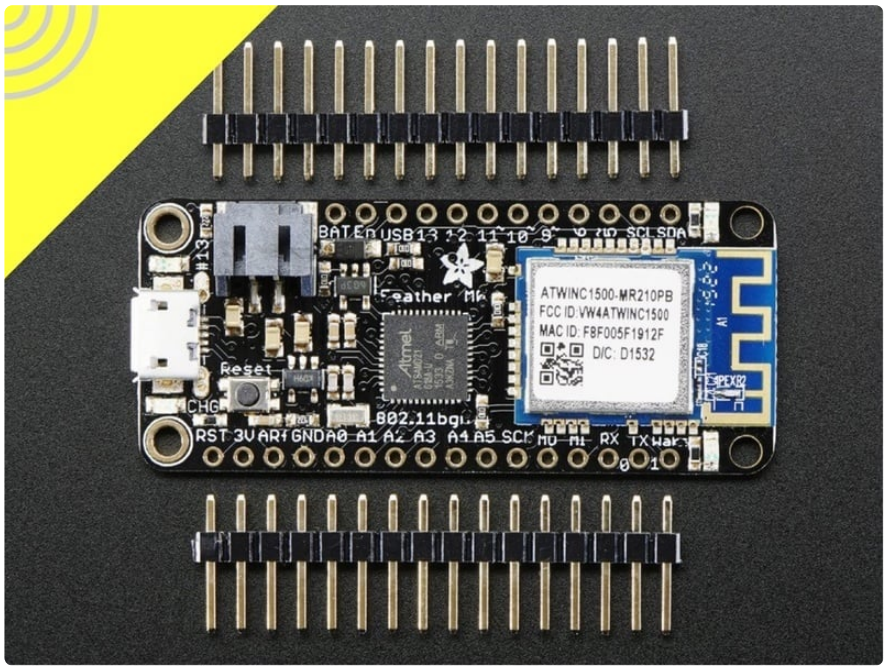
The OS handles all WiFi communication, so your code gets WiFi for free.. with some limits mentioned below. You don't have to write any low-level network code or allocate RAM for the WiFi hardware, you just call functions provided by the OS.

The Flash memory array is large enough to hold the support files used by Python code, so Adafruit has built CircuitPython to work on the ESP8266.

If you're trying to choose a WiFi development board, there are two points to remember:

1. **The ESP8266 has relatively few GPIO pins**. It can talk to a couple of external sensors and read a couple of buttons, but if you plan to pile on the hardware, you'll run into the hardware's limits quickly.
2. The ESP8266's operating system has to check the WiFi radio every few milliseconds. If it can't, its watchdog timer will crash the system and report a 'wdt error'. Unlike other microcontrollers, you don't have complete control of the hardware, and have to make your code cooperate with the rest of the system.

The Feather M0 with WINC1500:



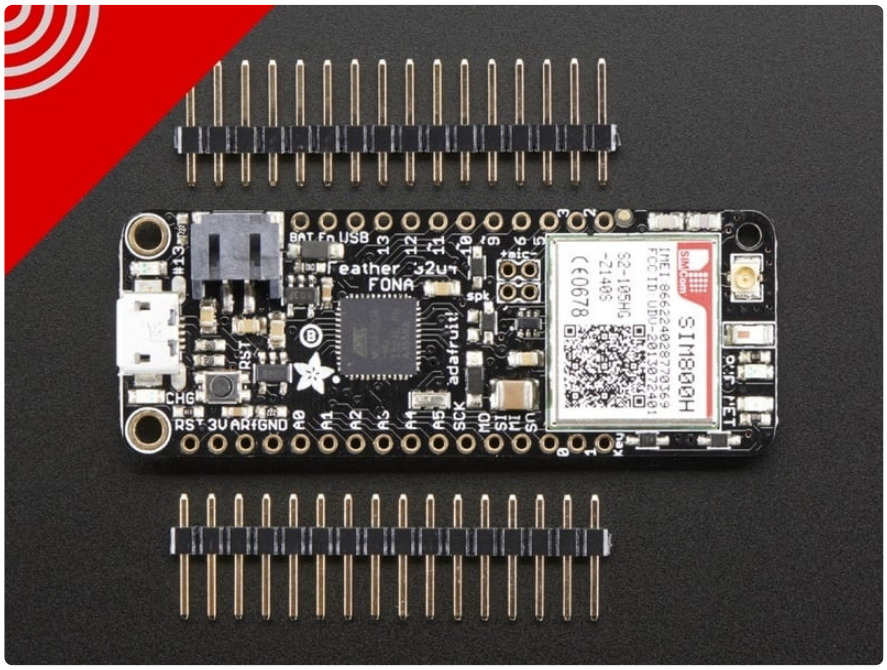
The WINC1500 is a Wifi peripheral that can talk to a microcontroller through its SPI interface. In the Arduino programming environment, it's supported by the Wifi101 library.

Unlike the ESP8266, a microcontroller connected to the WINC1500 has to run all the WiFi and low-level network code itself. You retain complete control of the hardware, but you do have to leave time for that code to run, and it does consume RAM.

The Feather pairs the WINC1500 with a 48MHz SAMD21G microcontroller, which has plenty of power and RAM for additional code.

Advanced Boards

The Feather 32u4 FONA:



FONA boards are made for remote data collection.

BLE connections go a few meters, WiFi can reach 100m or so, and packet radio modules can make a connection out to a few hundred meters.

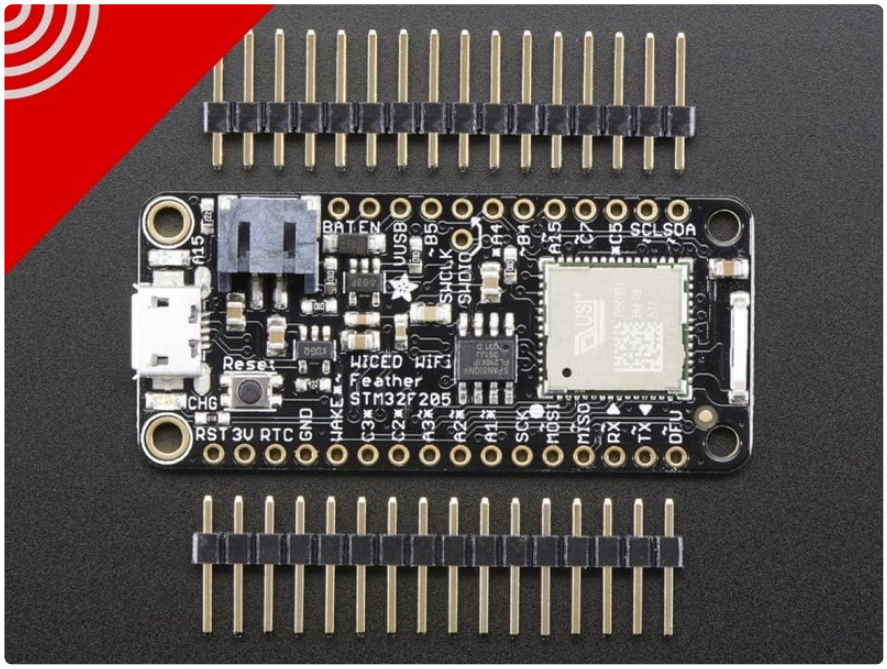
Cellular service covers the whole planet.. at least the parts of it within a few kilometers of a cell tower.

If you need to talk with something far away that has no connection to the internet, this is the board for you. If it moves around, you can add a GPS module and do your own remote asset tracking like the trucking and rail companies.

Or you could connect a headset and use it as a cellphone.

The microcontroller is an ATmega32u4, and the cellular module is the SIM800, which Adafruit also uses in the baseline FONA.

The WICED Feather:



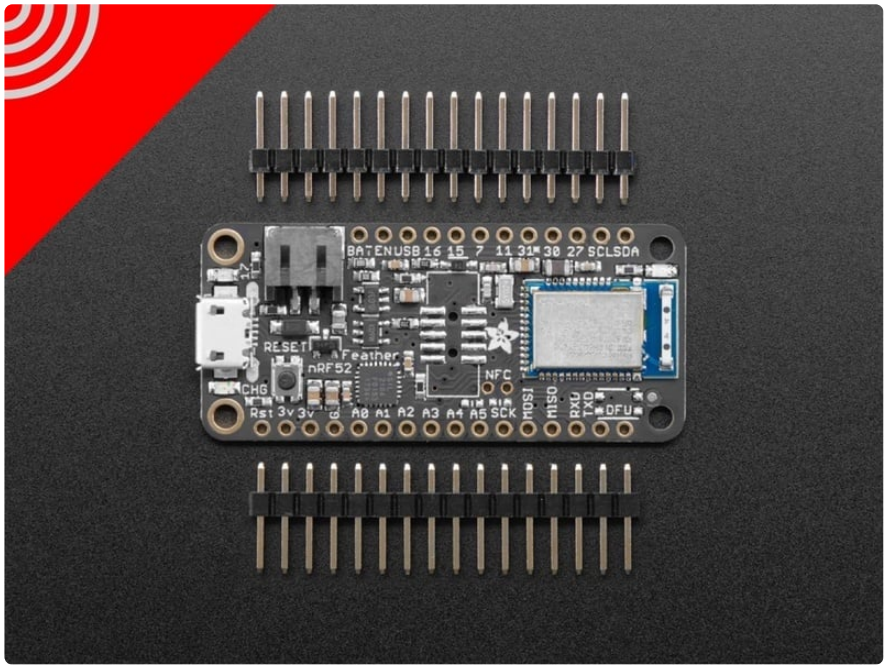
The WICED Feather is a microcontroller-and-WiFi device, similar to the ESP8266. The difference is that this one comes with Broadcom's truly excellent WICED (Wireless Internet Connectivity for Embedded Devices) platform.

It supports things like SSL connections and runs an RTOS (Real-Time Operating System), along with many other advanced features.

The catch is that the WICED software stack is proprietary and not all that compatible with the Arduino programming environment.

We developed our own layer of wrapper functions to make the two fit together a little better, but this board is still very different from other Arduino-friendly development boards. The main reason to choose this board is that you already know, or want to learn, the WICED platform.

The nRF52 Feather:



The nRF52 is a Bluetooth Low Energy SoC. It's also the only microcontroller we carry that can operate as a BLE central device.

If you're new to wireless communication and want to use BLE to make a group of microcontrollers talk to each other, we recommend you **STOP RIGHT NOW AND GO LOOK AT THE PACKET RADIO FEATHERS.**

99% of the time, BLE is a bad choice for peer-to-peer communication between microcontrollers.

The nRF52 is for the remaining 1% of use cases, and projects where you need a central device that can talk to existing BLE peripherals.

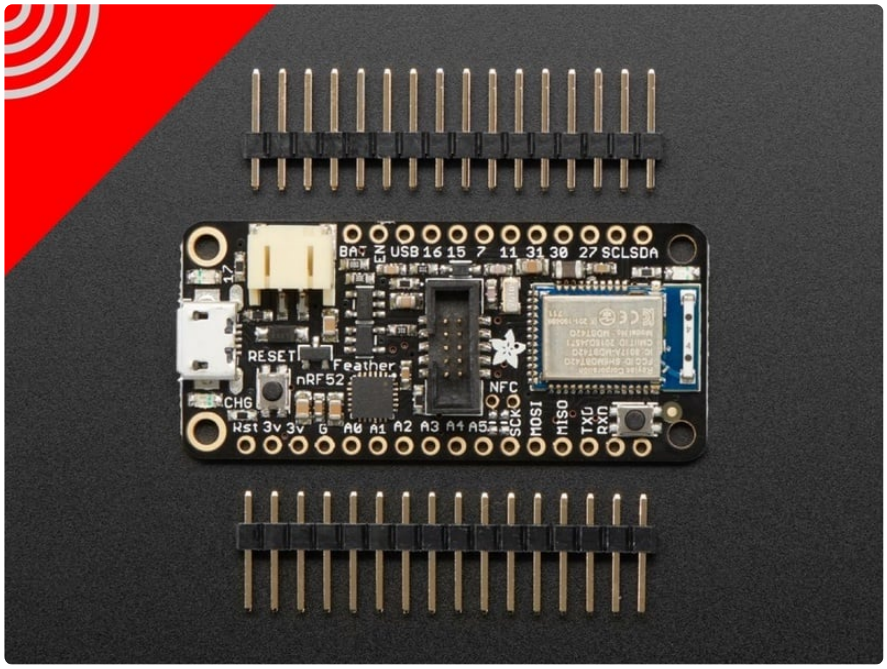
It's an impressive board by those standards, and works extremely well as a BLE peripheral if you're looking to develop that kind of device.

The nRF52 is only generally compatible with the Arduino programming environment, and has its own toolchain for uploading firmware. Installing the board support package can be a challenge, especially on Macs running OS X or Linux boxen.

If you're already familiar with BLE and know you want the features the nRF52 provides, the nRF52 Feather is a good development platform for you.

If you don't know you have a specific reason to use the nRF52, this is almost certainly not the board you want to use.

The nRF52 Feather with MyNewt OS:



If you scoffed at the warnings above, and you have no interest in programming an nRF52 from the Arduino environment, this is the board for you.

This one doesn't even pretend to be compatible with the Arduino IDE. Instead, it runs the Apache MyNewt operating system, and you program it on those terms.

The MyNewt environment has a number of compelling advantages for advanced developers, including professionally written BLE, networking, and cryptography stacks. That said, **THIS IS THE EXACT OPPOSITE OF A BEGINNER-FRIENDLY BOARD.**

Again, if you're familiar with BLE and MyNewt, and your JTAG pod is scuffed from frequent use, this board gives you a convenient selection of support hardware so you don't have to fuss around with the electronics before you start programming.

Cutting Edge Boards

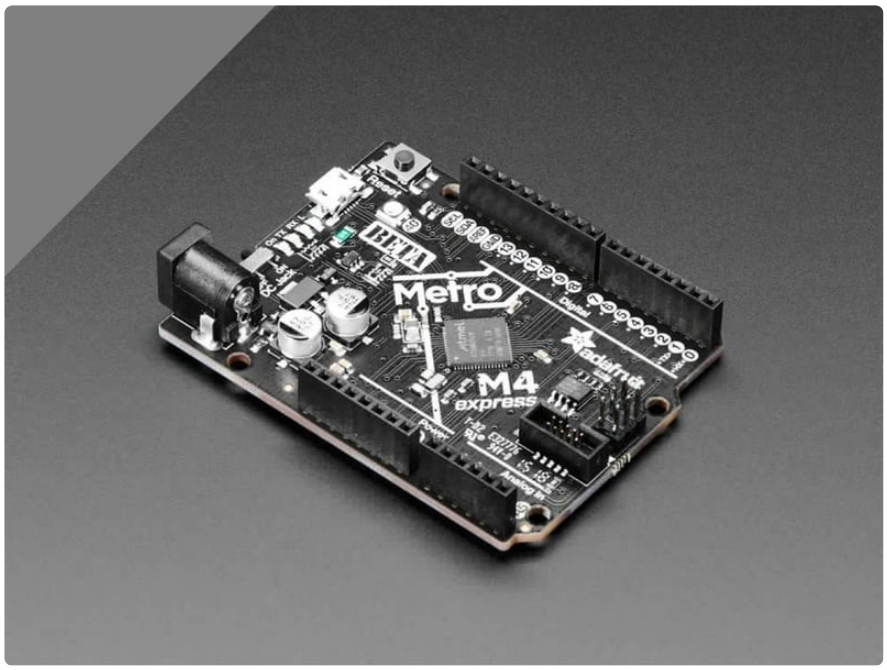
Get your R&D skills on

With cutting edge boards, you aren't just learning the future. You're helping to build it.

These boards don't have complete software support, and what does exist is still in active development. The companies that make the chips have released them so developers can help them find and fix bugs.

THESE ARE NOT GOOD BOARDS FOR BEGINNERS OR PEOPLE WHO NEED TO SHIP A STABLE DEVELOPMENT PLATFORM!

The Metro M4:



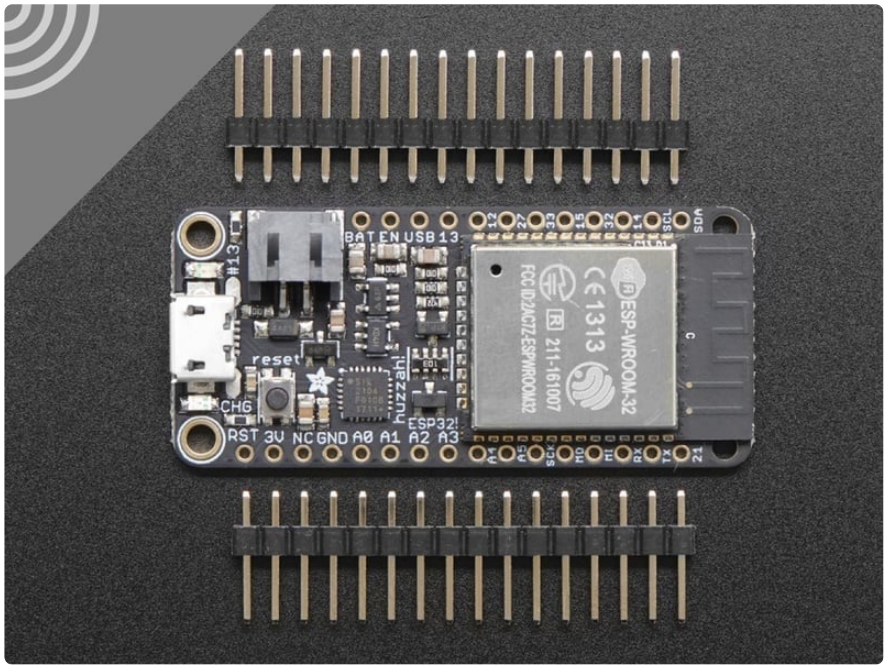
The Metro M4 is built around the SAMD51, one of the newest microcontrollers from Atmel/Microchip.

Its core is an ARM Cortex-M4 microcontroller, a 32-bit device with a hardware floating-point math engine and a digital signal processor (single math instructions that work on a whole block of memory).

SAMD51 peripherals include Ethernet and CAN interfaces, cryptography features like a true random number generator (TRNG), AES, and public-key processing circuits, and a block of configurable logic similar to a small FPGA.

The fact that the hardware can do those things doesn't mean code support exists yet. In most cases, if you want to use the SAMD51's most advanced features, you'll have to read the datasheets and develop the code yourself. Adafruit is working on getting the most out of the SAMD51, mainly through CircuitPython.

The Feather ESP32:



The ESP32 is an SOC made for wireless applications. It was designed by Espressif, the same company that makes the ESP8266, and has built-in radios to support WiFi, Bluetooth Classic, BLE, and NFC.

It's a new chip, and is essentially a public beta at this point. The firmware to support some of the low-level features is still being written, and bugs are still being found. Some parts may not work and others are unreliable.

The feature set is impressive, and it's worth working with the current hardware so you can be familiar with it when all the problems are fixed.

We don't recommend it for mission-critical projects, though.