Bootloading Basics

Created by Bill Earl

https://learn.adafruit.com/bootloader-basics

Last updated on 2024-03-08 03:28:56 PM EST

Table of Contents

Overview 3

- What is a Bootloader?
- So, how do you load a program onto a brand-new microcontroller?
- What if you don't have one of those devices?
- So where do bootloaders come from?

Why not build them in? 5

- Some of the reasons you may want to forgo the convenience of a bootloader:

Types of Bootloaders 6

- Basic Bootloaders
- Advanced Bootloaders
- Multi-stage bootstraps

A Brief History of Bootstrapping 9

- Switches and Patch-Cables
- Punch-Cards and Paper Tape
- Programmable Read Only Memory
- EPROM, EEPROM and Flash
- Bootstraps for the Bootstraps

Bootloader Resources 12

Bootloader Usage 12

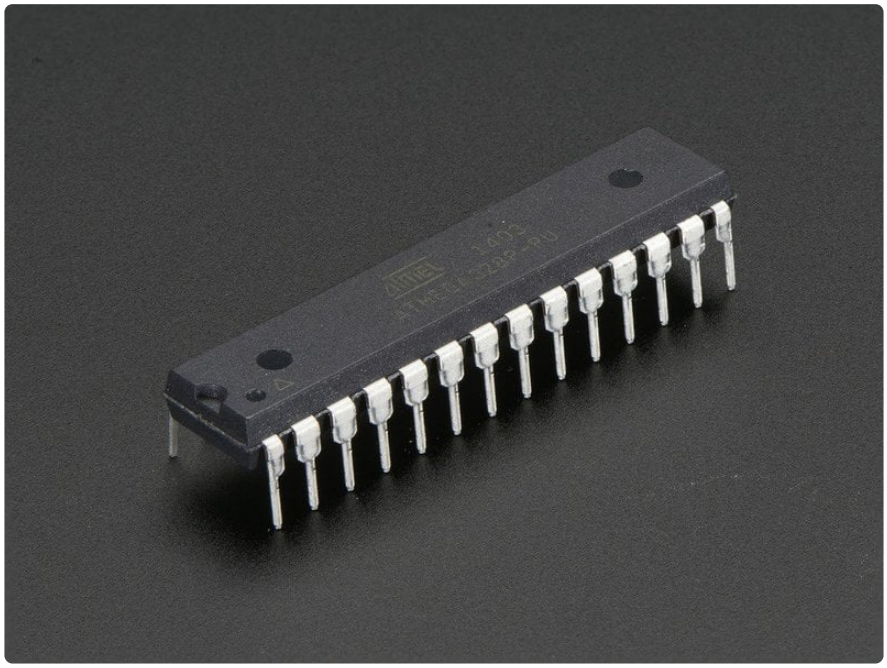
- CircuitPython and Bootloaders

BYOB! (Burn Your Own Bootloader) 14

- Bootloader Customization, Installation and Repair
- General Bootloader Programming and Repair
- Bootloader Programming Devices

Overview

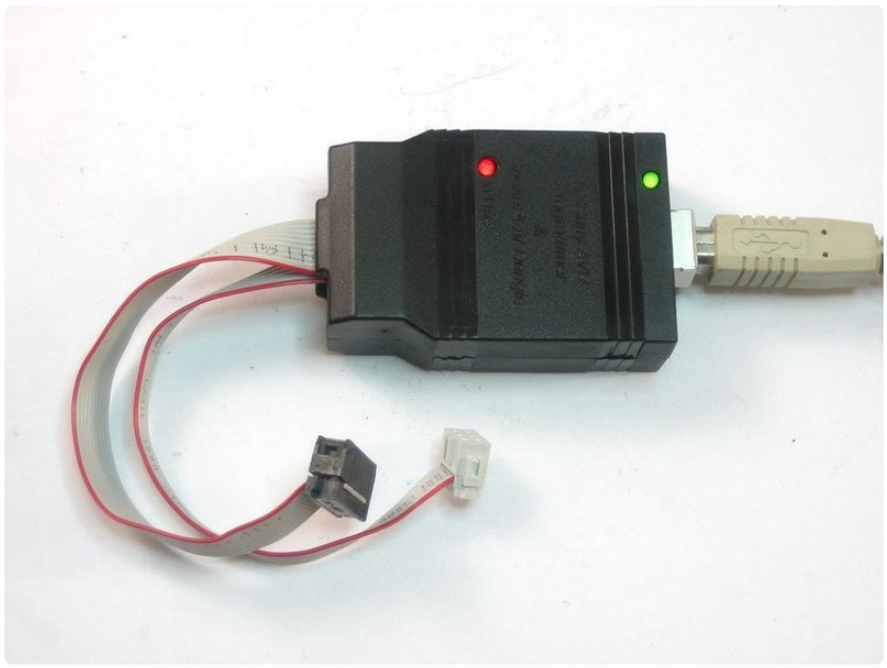
What is a Bootloader?



If you take a bare microcontroller chip fresh from the factory and apply power to it, the internal system clock will start ticking away. But nothing else will happen. Without a program to run, the chip will just sit there doing nothing.

So, how do you load a program onto a brand-new microcontroller?

Most microcontroller chips have a special programming interface such as JTAG or AVRISP that allows you to burn programs into Flash memory. Unfortunately, these programming interfaces generally require specialized tools like a USBTinyISP or a Segger J-Link.



What if you don't have one of those devices?

Most of the processor boards we carry have a pre-loaded program called a 'Bootloader'. A Bootloader is a program that allows you to load other programs via a more convenient interface like a standard USB cable. When you power-up or reset your microcontroller board, the bootloader checks to see if there is an upload request. If there is, it will upload the new program and burn it into Flash memory. If not, it will start running the last program that you loaded.

So where do bootloaders come from?

Here at the Adafruit factory, we use custom-built programming fixtures to burn a bootloader onto each board as it comes off the manufacturing line. So when you get your processor board, you can just plug it into your PC and upload your own programs via a USB cable.

Instragram Bootloader timelapse: https://www.instagram.com/p/BcAMCZ9DeDS

Why not build them in?

You might wonder why the chip manufacturers don't just bake a bootloader into every microcontroller chip at the factory.

Keep in mind that the vast majority of microcontroller chips wind up embedded in products ranging from toasters to traffic lights. A modern automobile may have 100 or more embedded microcontrollers that only need to be programmed once by the parts supplier. Cost, performance or other constraints can easily outweigh any considerations of programming convenience.

There have been a few chips that come with a "ROM" (permanent) bootloader. Such as the ESP32 or the LPC11xx series with their USB mass storage bootloader, but these are very few-and-far-between.

Some of the reasons you may want to forgo the convenience of a bootloader:

Memory

Bootloaders tend to be rather small programs. But they do take some space. For applications that need every last byte of program memory, the convenience of a bootloader may be an unnecessary luxury.

Embedded systems designers try to keep product costs down by using the smallest possible processor that can do the job. Investing in a proprietary programming interface for the manufacturing line is a one-time investment. Sizing up to a bigger processor adds cost to every unit sold.

Start-up Time

Most bootloaders incur some dead-time on power-up or reset as they look for a potential upload. If ‘instant on’ is a requirement for your application, a stock bootloader may not be an option.

Security

If you don’t want everyone with an Arduino IDE to be able to re-program your device, ditching the bootloader will add another obstacle for a would-be hacker.

(Although any hacker worth her salt will probably have an AVRISP and/or JLink in her toolbox!)

Types of Bootloaders

There are thousands of different bootloaders. The design and capabilities vary depending on the processor architectures, hardware resources and intended usage.

Basic Bootloaders

The simplest type of bootloader has just one function: Check for an upload at startup and load it.

Arduino UNO Bootloader

A good example of a basic bootloader is the bootloader programmed into the Arduino UNO.

On power-up or reset, the bootloader watches the serial receive pin for a few seconds - waiting to receive the special sequence of bytes that indicates an upload attempt from the IDE. If that magic byte sequence is received, the bootloader will respond with an acknowledgement and the IDE will start sending the code. The bootloader will receive the code from the IDE and burn it into flash memory.

If the bootloader does not see the magic byte sequence, after a few seconds it will proceed to start running whatever code is already burned into flash.

Advanced Bootloaders

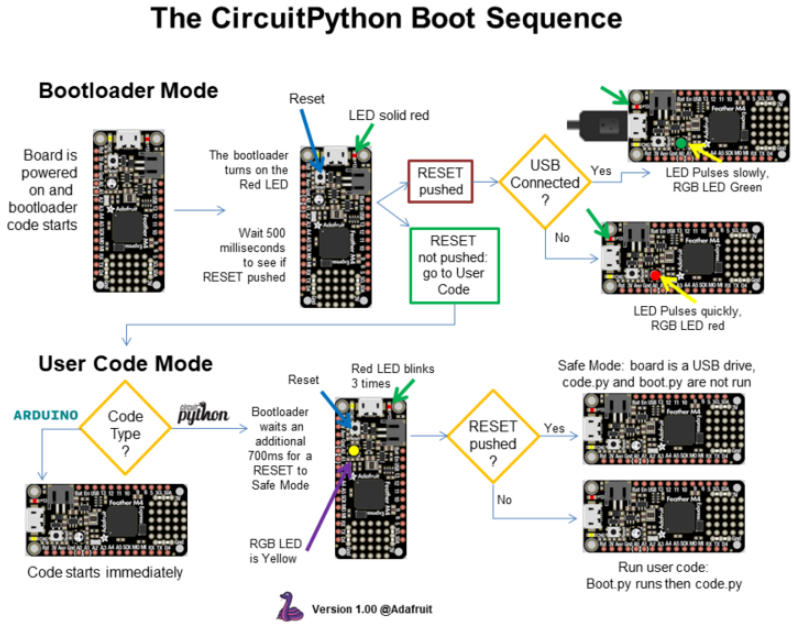
More advanced bootloaders add additional functionality such as choosing between operating modes and/or performing some low-level diagnostics.

CircuitPython UF2 Bootloader

A good example of an advanced bootloader is the UF2 bootloader used on our CircuitPython capable boards.

This bootloader can work like an Arduino and accept serial uploads via the USB connection. Or it can emulate a disk drive that lets you copy Python programs to it.

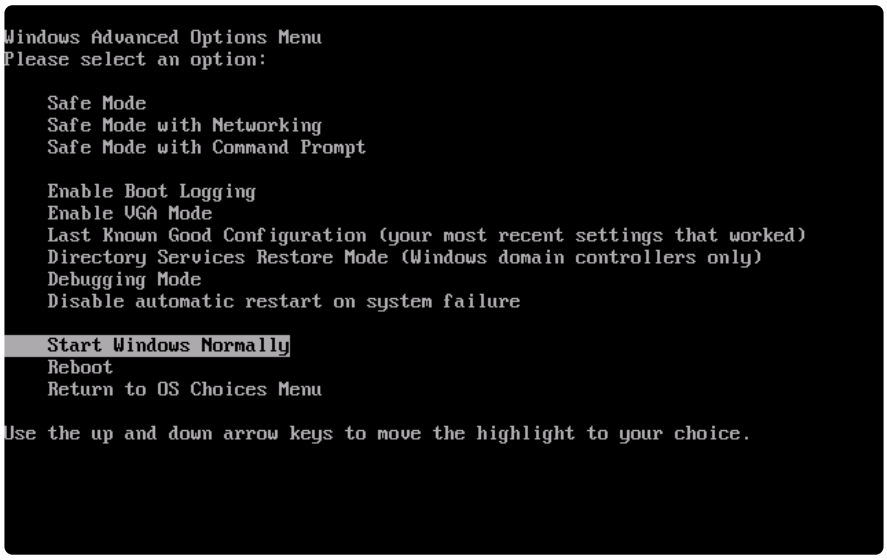
An on-board RGB LED is used to provide diagnostic feedback in the form of steady or flashing patterns in different colors to indicate the state of the bootloading process.



Multi-stage bootstraps

Your notebook, desktop computer and even your smartphone most likely have at least two stages of bootstrapping. This gives you options such as selecting different operating modes, or even an entirely different operating system.

The BIOS or UEFI is burned into flash memory on the motherboard. At startup it will look for a bootable drive and load an operating system bootloader from the drive. At that point it turns control over to the operating system's bootloader which takes care of loading your operating system. Once the operating system is up and running, you can just point and click to load and run any program you want.

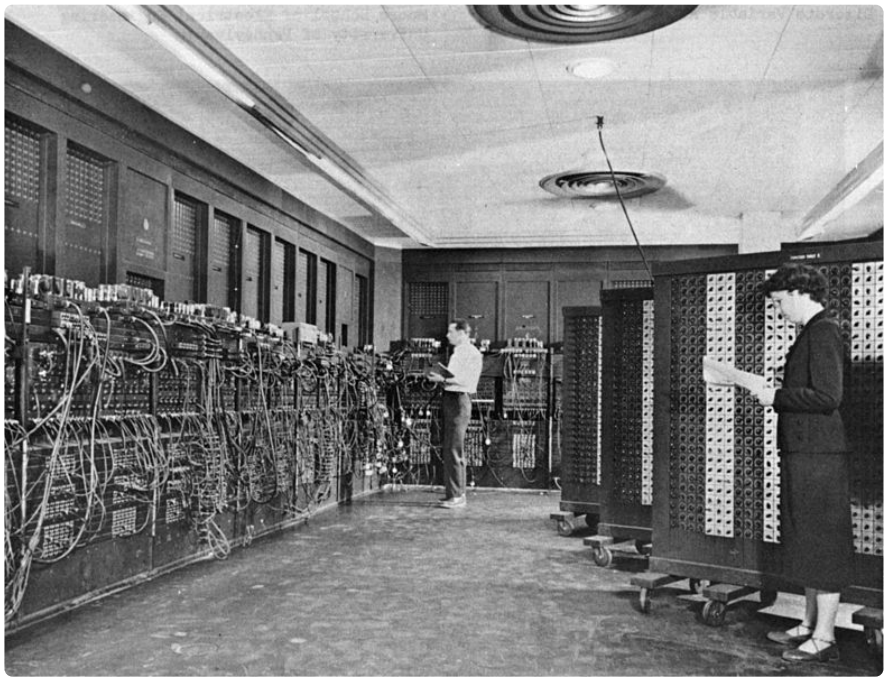


A Brief History of Bootstrapping

The word ‘Bootloader’ is a contraction of ‘bootstrap loader’. This is derived from the idea of ‘pulling yourself up by your bootstraps’. A computer without a program to run is a useless piece of machinery. So how do you get the program into the computer?

Switches and Patch-Cables

In the early days of computing, computers were programmed literally bit-by-bit using patch cables or an array of toggle switches on the front panel. As you might imagine, this is quite tedious to do for anything more than a trivial program.



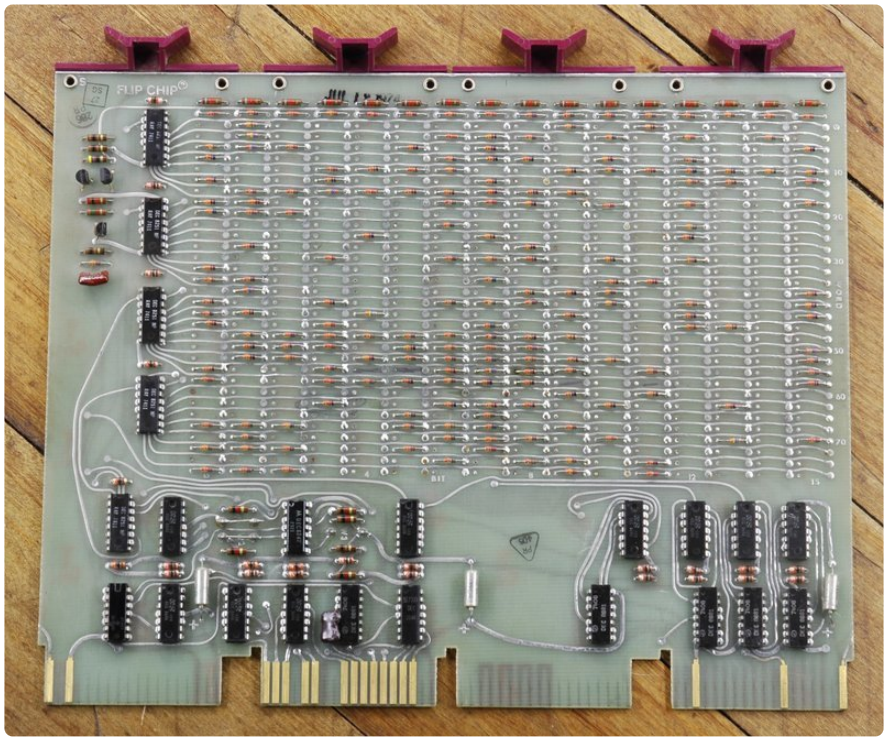
Punch-Cards and Paper Tape

One day, someone had a bright idea: Write a small program that knows how to load larger programs stored on punched cards or paper-tape. And the computer bootstrap loader was born.

Programmable Read Only Memory

Although the bootstrap program saved lots of time, it was still rather tedious to have to toggle in the instructions every time you restarted the machine. Eventually, another engineer had another bright idea: Build the bootloader into the computer.

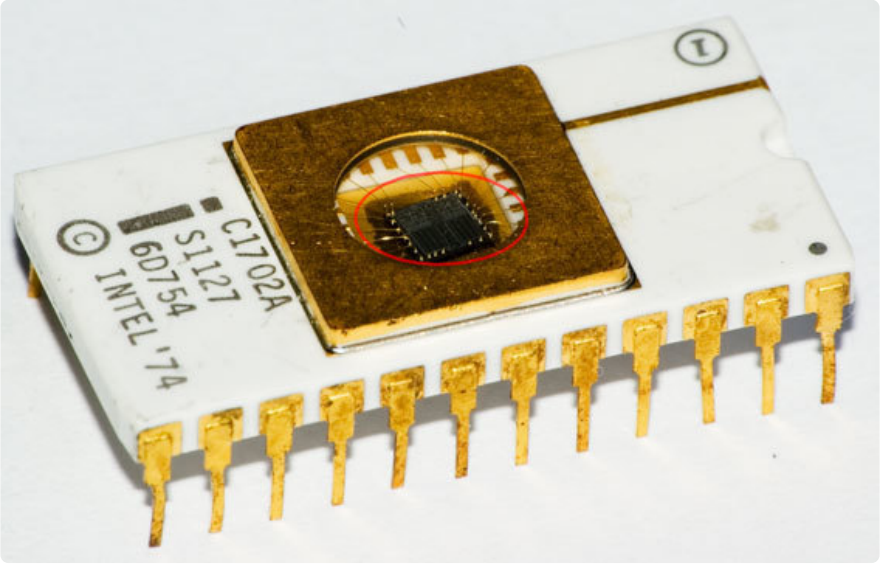
You could solder an array of diodes to program the 1's and 0's of the bootloader code at the required memory addresses to create a Read-Only Memory (ROM) for the bootloader. And the ROM boot was born.



EPROM, EEPROM and Flash

Moving on from the hard-wired diode boot, advanced in technology led to various forms of electrically programmable non-volatile memories.

These days, bootloaders are programmed into EEPROM or Flash memory. So you don't need a soldering iron to modify or repair your bootloader. For more complex systems, bootloading may be performed in multiple stages with each stage loading a more complex piece of software.



Bootstraps for the Bootstraps

Your notebook or desktop computer most likely has at least two stages of bootstraps. The BIOS or UEFI is burned into flash memory on the motherboard. At startup it will look for a bootable drive and load an operating system bootloader from the drive. At that point it turns control over to the operating system's bootloader which takes care

of loading your operating system. Once the operating system is up and running, you can just point and click to load and run any program you want.

Bootloader Resources

First time with an unfamiliar bootloader? Not sure what that blinking LED means? Looking for a way to speed up your restart times? Or just feel like building a better bootloader?

The following pages contain links to other tutorials regarding the care and feeding of common bootloaders.

Bootloader Usage

Most bootloaders are fairly simple to use. But the details vary somewhat depending on the platform and its capabilities. The links on this page describe the unique features of each one.

Gemma Bootloader

https://adafru.it/F56

Trinket Bootloader

https://adafru.it/F57

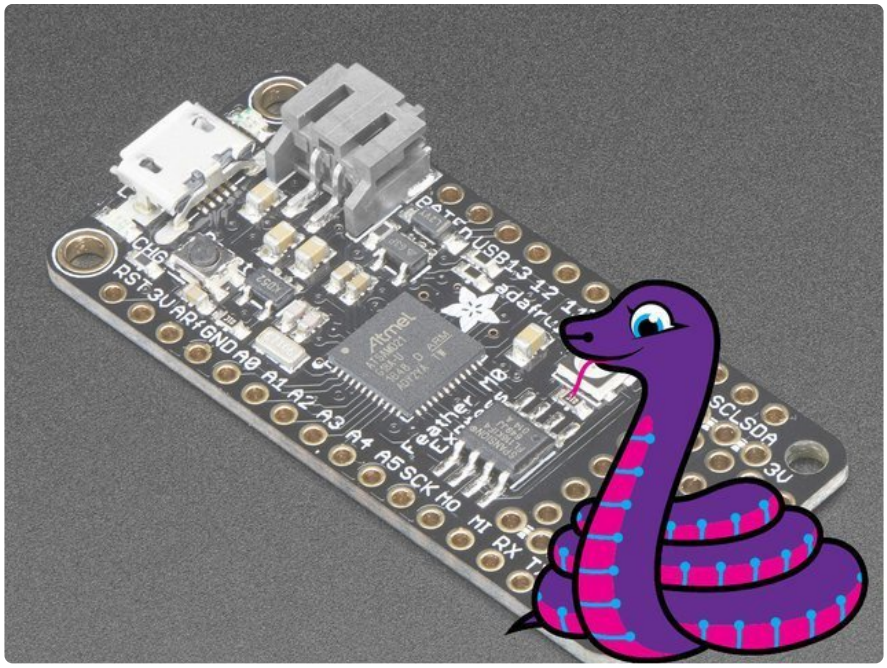
Pro Trinket Bootloader

https://adafru.it/eUv

NRF52 Bootloader

https://adafru.it/F58

CircuitPython and Bootloaders



Many people confuse the CircuitPython firmware with a bootloader. CircuitPython itself is not a bootloader. CircuitPython relies on the UF2 Bootloader to copy your Python program files to the board.

The UF2 bootloader emulates a USB mass storage device. When you connect a board with a UF2 bootloader to your computer, it looks like you plugged in a thumb-drive. It's not a real drive, so you can't use it to store all your favorite MP3s. But it will recognize certain file names as Python programs to load.

The UF2 loader can also operate in BOSSA compatible mode, so it can accept uploads of Arduino programs from the Arduino IDE.

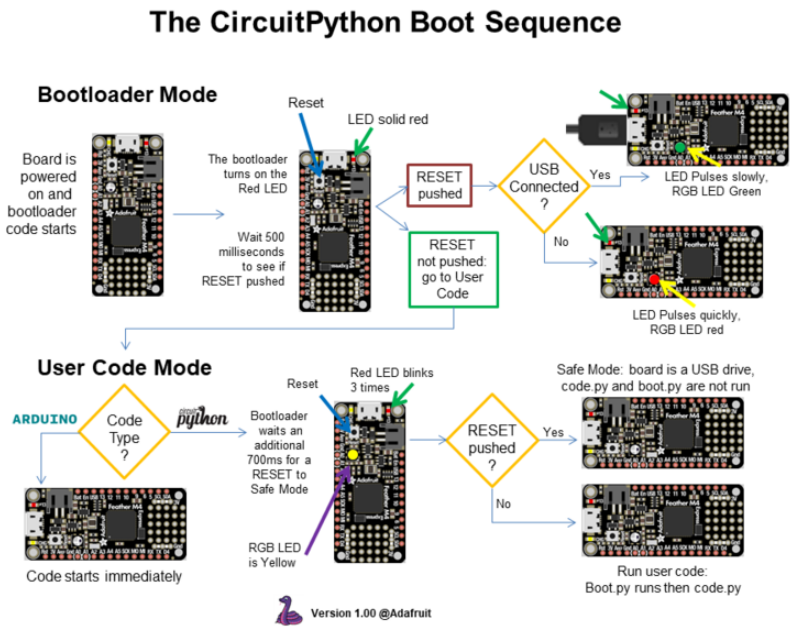
The links below describe the CircuitPython system and the UF2 Bootloader in depth and how they work together to load your Python or Arduino programs.

What is Circuit Python

https://adafru.it/F59

The UF2 Bootloader

https://adafru.it/F5a



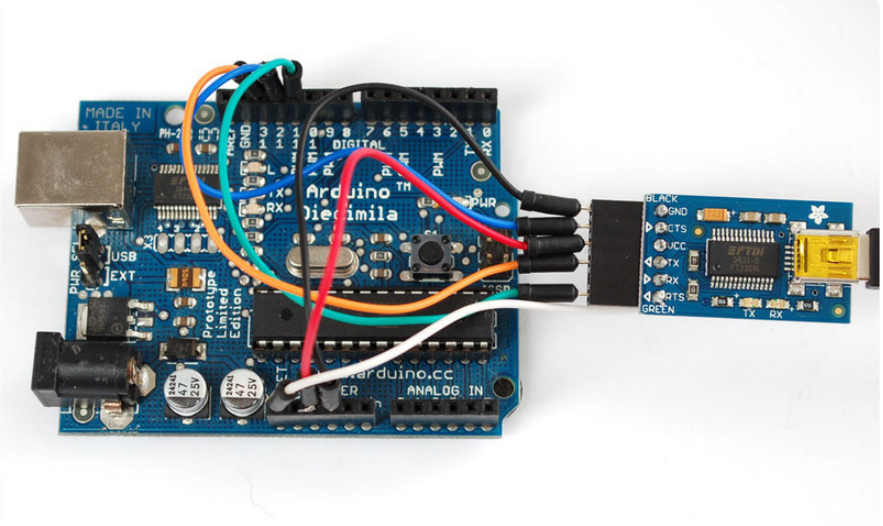
BYOB! (Burn Your Own Bootloader)

There are a few reasons why you might want to burn a bootloader yourself:

- You are starting with a bare chip, fresh from the factory
- You have 'bricked' a processor board and want to repair it
- You have special requirements and want to install a custom bootloader

This page has a number of links to tools and instructions for repairing, customizing and/or burning bootloaders on various microcontrollers:

Bootloader Customization, Installation and Repair



General Bootloader Programming and Repair

Arduino Bootloader Hacks

https://adafru.it/F5b

Burning Arduino Bootloaders with an FTDI Friend

https://adafru.it/Cfl

Burning Arudino Bootloaders with a Raspberry Pi

https://adafru.it/F5c

Pro Trinket Bootloader

https://adafru.it/F5d

Trinket/Gemma Bootloader

https://adafru.it/F5e

Programming SAMD Bootloaders

https://adafru.it/F5f

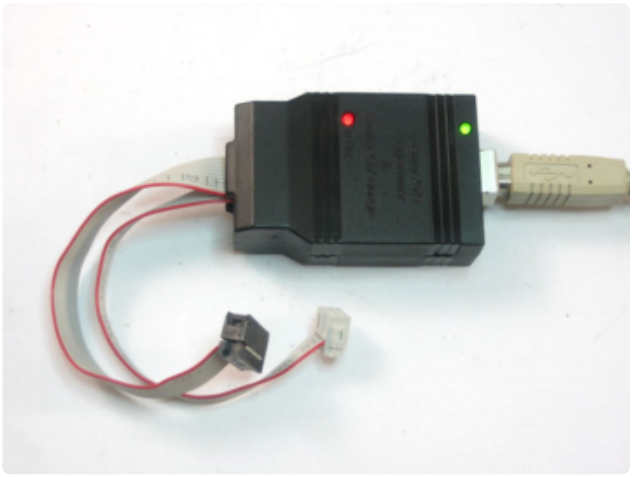
Compiling the ATSAMD21 Bootloader

https://adafru.it/F5g

Flashing the NRF52 Bootloader

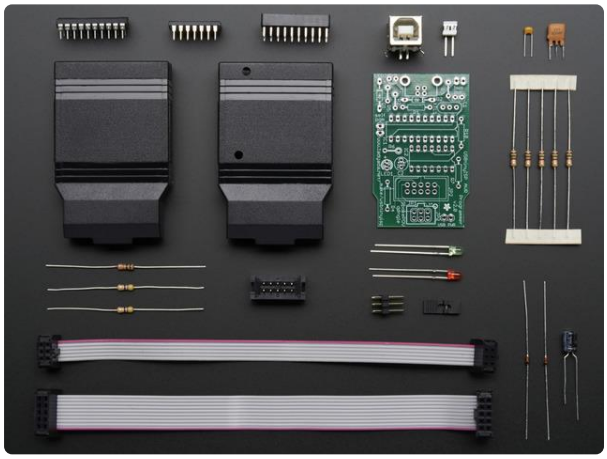
https://adafru.it/Dw9

Bootloader Programming Devices



USBTiny ISP

This device can be used to flash bootloaders or other programs onto AVR Microcontroller boards with ICSP headers.



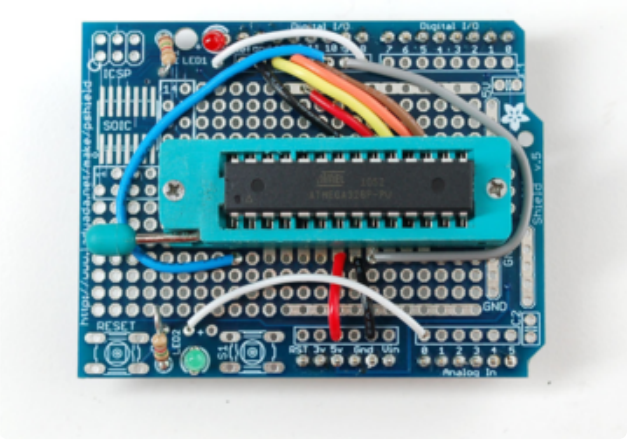
USBtinyISP AVR Programmer Kit (USB SpokePOV Dongle)

USBtinyISP is a simple open-source USB AVR programmer and SPI interface. It is low cost, easy to make, works great with avrdude, has both

https://www.adafruit.com/product/46

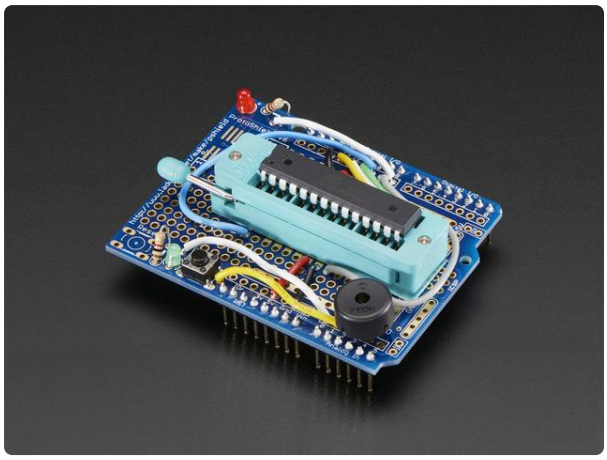
USBTiny ISP

https://adafru.it/dhl



Standalone AVR Chip Programmer

This board is designed for programming Atmel/Microchip AVR ATmega processors in 28 pin DIP packages.



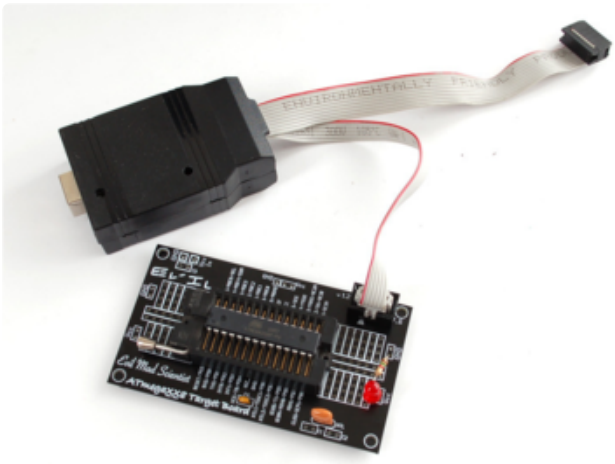
Standalone AVR ISP Programmer Shield Kit - includes blank chip!

This shield kit pack will allow you to turn any Arduino into an AVR chip burner! It is specifically designed for people who want to program Atmega328P's to turn them into Arduino...

https://www.adafruit.com/product/462

Standalone AVR Chip Programer

https://adafru.it/F5h



Mass AVR ISP Programmer

Another option for 28 pin ATmega processor chips.

Mass AVR ISP Programmer

https://adafru.it/F5i

Segger J-Link Programmer

Segger makes a variety of programmers designed for programming many different processor families and architectures.

Segger J-Link User Guide

http://adafru.it/3571080011

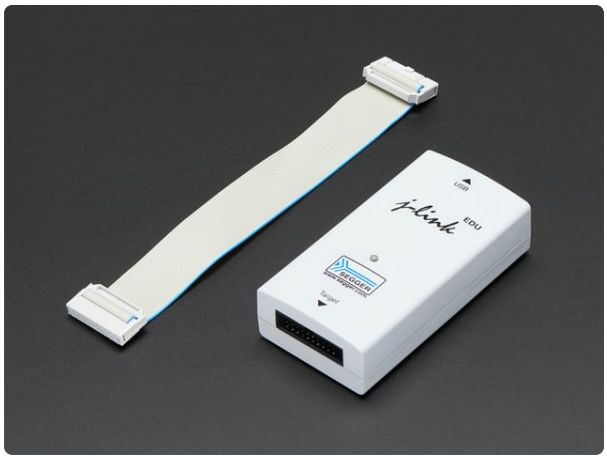
Segger Forums

https://adafru.it/F5k

SEGGER J-Link BASE - JTAG/SWD Debugger

The SEGGER J-Link BASE is identical to the cheaper J-Link EDU model except for the terms of...

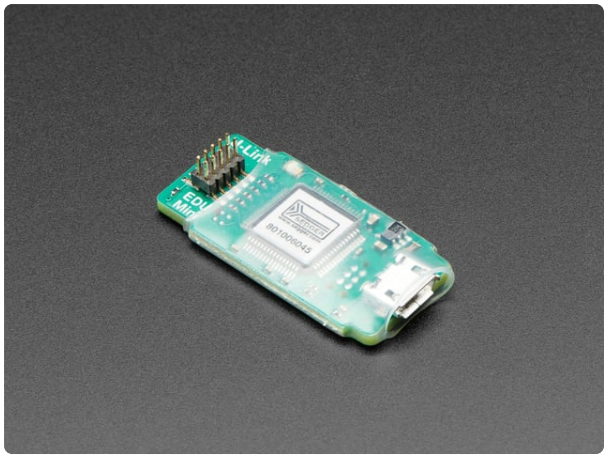
https://www.adafruit.com/product/2209



SEGGER J-Link EDU - JTAG/SWD Debugger

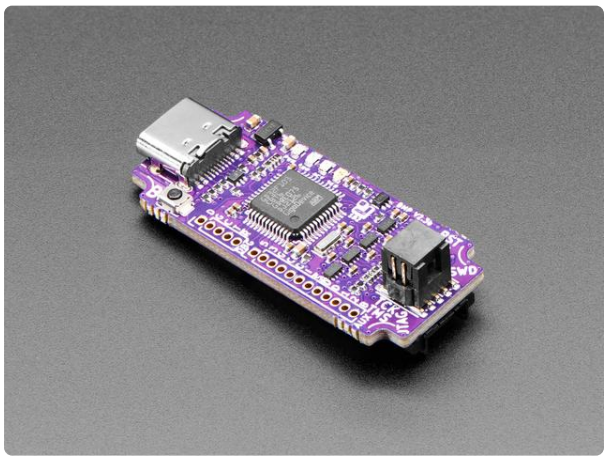
Discontinued - you can grab SEGGER J-Link EDU Mini - JTAG/SWD Debugger instead! The SEGGER J-Link EDU is...

https://www.adafruit.com/product/1369

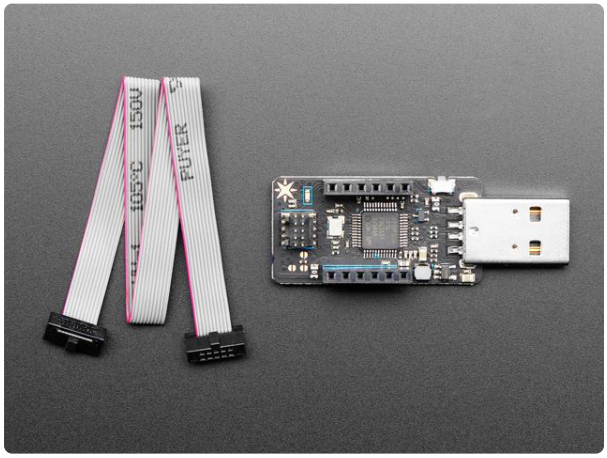


[SEGGER J-Link EDU Mini - JTAG/SWD Debugger]
Doing some serious development on any ARM-based platform, and tired of 'printf' plus an LED to debug? A proper JTAG/SWD HW debugger can make debugging more of a pleasure and...
https://www.adafruit.com/product/3571

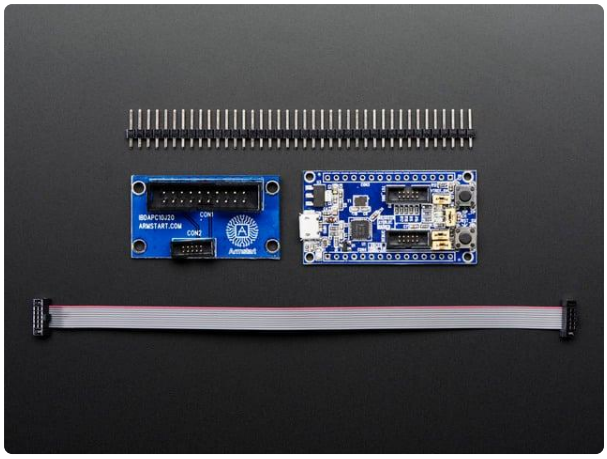
And a variety of others!



[Black Magic Probe with JTAG Cable and Serial Cable]
Toss away your boring old SWD/JTAG adapters! This Black Magic Probe, designed by 1BitSquared with
https://www.adafruit.com/product/3839



[Particle Debugger]
Discontinued - you can grab the Black Magic Probe with JTAG Cable and Serial Cable -...
https://www.adafruit.com/product/4001



IBDAP - CMSIS-DAP JTAG/SWD Debug Adapter Kit

A lot of debug adapters cost money that you'd much rather spend on tinkering. But not the IBDAP - CMSIS-DAP JTAG/SWD Debug Adapter Kit from armstart...

https://www.adafruit.com/product/2764