Manafish Ultimate Guide to Microcontrollers and SBCs

In this guide I'll try my best to map out different Microcontrollers and SBCs, to help you find which one to use for your project. When researching for writing this, I was surprised to find that there were some options I had never even heard of that had some super cool features. So buckle up, hopefully you'll learn something new!

Microcontroller vs. SBC

Before we begin, I want to clarify the difference between microcontrollers and single board computers (SBCs), as these are often confused and the list will include both.

The main difference is whether the processor runs a full operating system like Linux, or just simple code that you upload to it.

Microcontrollers
Arduino Uno
ESP32
Raspberry Pi Pico 2
Teensy 4.1
STM32
Particle Photon

Single Board Computers
Raspberry Pi
BeagleBone Black
Orange Pi / Banana Pi / Radxa / Odroid
Nvidia Jetson

Single board computers run a full operating system. That means you'll be able to do things like browse their files, start and run different programs on them, connect them to a monitor, use them to browse the internet, use them to process video streams and display graphics. More flexibility, but more overhead too.

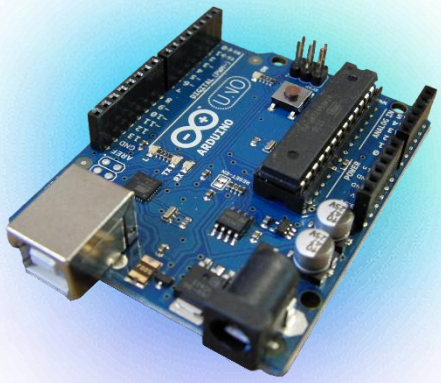
Microcontrollers are far simpler. They do not run a full linux-based operating system, they just execute the exact code you upload on them. The benefit is that they run in a more deterministic way, which allows for faster loops and more precise signals.

Think of a microcontroller as a super fast customizable calculator, and a SBC as mini size computer.

With that out of the way, let's begin!

MICROCONTROLLERS

Arduino Uno – The OG that's getting old



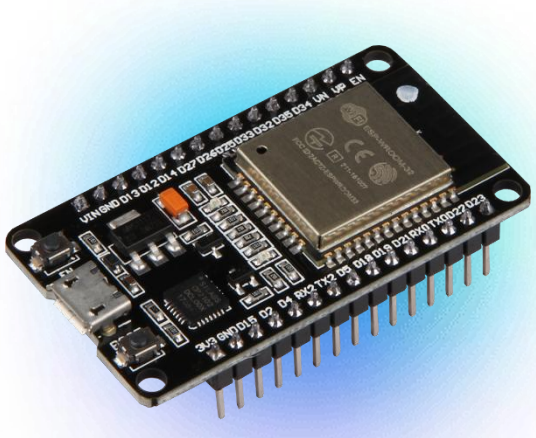
Price: $20 - $27
Difficulty: Easy
CPU: Single-core 16 MHz
RAM: 2 KB SRAM
Notable Features: Huge library
GPIO: 14 digital + 6 analog
Power consumption: ~50 mA
Operating voltage: 5V

Arduiono Uno is the OG microcontroller. When it was released, it was a big deal because it was the first microcontroller (as far as I'm aware) that was easy to program. Before Arduino, setting up any sort of custom electronics device would require advanced enterprise level tools. Arduino completely changed that by making the Arduino IDE, which allows anyone to easily program the microcontroller.

However, there has been a lot of progress since Arduino was initially released. Even though Arduino has updated their models and made them more powerful, all things considered in 2026 there are microcontrollers like ESP32 that are way cheaper, have more functionality and also run *way* faster than Arduino.

***When to use**:* Arduino UNO used to be the king of microcontrollers, but in 2026 there are options that are cheaper and better in almost every way. I'd avoid unless you are using someone else's code made specifically for Arduino UNO, or some tool that requires it (Like BLHeli S Configurator).

ESP32 – The new default



Price: $5 - $10
Difficulty: Easy
CPU: Dual-core 240 MHz
RAM: 520 KB SRAM
Notable Features: WiFi and Bluetooth
GPIO: ~34 usable GPIO pins
Power consumption: ~20 mA
Operating voltage: 3.3V

ESP32 is like the new Arduino. It is way faster. It also supports WiFi and Bluetooth. It can be programmed with the Arduino IDE with the exact same workflow as an Arduino, or it can run MicroPython and CurcuitPython, which are versions of Python made for microcontrollers.

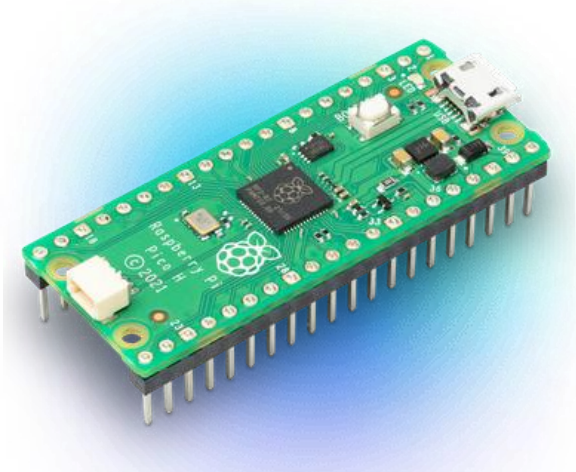
The cleanly integrated WiFi and Bluetooth support makes this a go-to for internet of things applications – things like smart home setups where you need many small systems and sensors to communicate with one another.

It is also possible to get ESP32 as just a chip that you can put on your own PCB, so if you develop anything cool with it and want to take it one step further and start shipping units, taking the next step is easy.

One little drawback is that it only runs at 3.3V, not 5V like many other microcontrollers. This might be a problem for some sensors, but for the most part it should be okay.

***When to use**:* In many ways the new default microcontroller. Fast. Easy to use. Integrated WiFi and Bluetooth makes it really shine for projects that require wireless control, like smart home appliances or anything remote controlled.

Raspberry Pi Pico 2 – ESP32's precise brother



Price: $5 - $8
Difficulty: Easy
CPU: Dual-core 150 MHz
RAM: 520 KB SRAM
Notable Features: 3 PIO blocks
GPIO: 26 usable GPIO pins
Power consumption: ~25 mA
Operating voltage: 3.3V

On the surface, Raspberry Pi Pico has very similar capabilities to the ESP32. It's the same price range, and also easy to set up with a variety of code languages. The "W" version of the Pico also supports WiFi and Bluetooth.

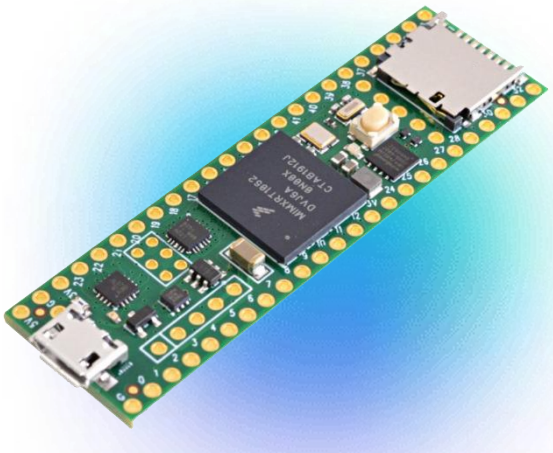
However, the Pico has one trick up it's sleeve, it has two PIOs. A PIO (Programmable I/O) is basically a mini-CPU inside the main CPU whose only job is to read or write bits. Since they are separate from the main CPU, they are able to just focus on the output they're making, without any waiting for other parts of the code, and thus create super precise signals.

These super precise signals can be useful in some cases, like:

- Controlling LED strips (WS2812)
- Controlling BLDC motors with ESC/DShot motor control
- Making unsupported sensor protocols work without a driver

When to use: Very similar use cases to ESP32, most projects where you'd choose an ESP32 work with this too. PIO makes it great if you need precise signals.

Teensy 4.1 – The premium microcontroller



Price: $30 - $35
Difficulty: Easy
CPU: Single-core 600 MHz
RAM: 1 MB
Notable Features: Ethernet PHY + SD card slot
GPIO: 55 digital pins
Power consumption: ~100 mA
Operating voltage: 3.3V

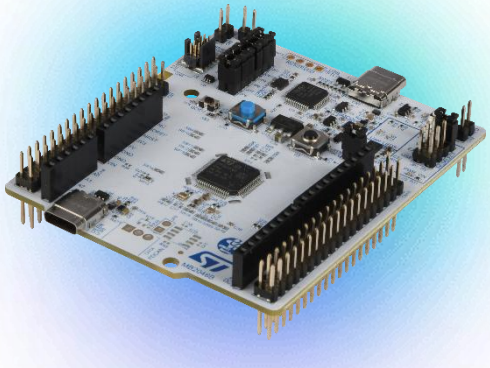
Teensy 4.1 runs at a whooping 600MHz, way faster than the alternatives we've covered so far. It also has some nice extra features like an SD-card slot for way more storage and support for ethernet communication. It is the better but more expensive microcontroller.

It's also known for having good documentation, and is easy to use with Arduino IDE or Micropython.

It does not support WiFi or Bluetooth, so if you need wireless communication, stay clear.

***When to use**:* Use this when you need more performance than what the other microcontrollers offer. Some examples are super fast control loops and audio synths.

STM32 – Built for scaling



Price: $3 - $20+ per chip
Difficulty: Hard
CPU: Cortex-M0+ to M7, 48 - 600 MHz
RAM: 8 KB - 4 MB
Notable Features: 100+ variants for every use case
GPIO: 20 - 144 pins
Power consumption: Varies, some are ultra low power
Operating voltage: 1.8V - 3.6V

STM32 is not technically one microcontroller – it's a family of MCU chips covering a wide range of performance categories.

STM32 microcontrollers are not really aimed at hobby users. They are rather intended for professional embedded engineers working on projects that will get mass produced and shipped in thousands of units.

They are way harder to program and use than the more hobby-friendly microcontrollers like ESP32 and Pico. On the other hand, the wide variety of different options makes them very flexible, and they can easily be added to custom PCBAs.

***When to use**:* Probably not the best option for a hobby project. Great choice if you're making production electronics that will be shipped on thousands of units and needs to work reliably for a cheap price.

Particle Photon 2 – Cloud connected by default

Price: $15 - $18
Difficulty: Easy
CPU: Single-core 200 MHz
RAM: 3 MB
Notable Features: Built-in cloud platform
GPIO: ~18 usable GPIO pins
Power consumption: ~80 mA
Operating voltage: 3.3V

Photon 2 is fairly standard microcontroller when it come to the specs. However, it has one unique feature that separates it from the others: Particles Device OS and cloud platform. Every Photon will be automatically linked to your account, and you can access and modify them from your PC without any additional setup.

Let's say you have 100+ devices shipped to customers. Updating the firmware on all these devices would normally be a huge pain. With Particles cloud platform, updating every device can be done with a few simple commands instead.

The Particle OS also makes things like communication between devices far simpler.

***When to use**:* Use when you will have a lot of wirelessly connected devices and want to be able to control all via a cloud platform without additional setup.

SINGLE BOARD COMPUTERS

Raspberry Pi (3, 3B, 4, 5 etc.) – The default SBC

Price: $35-$120
Difficulty: Easy
CPU: Quad-core 2.4 GHz
RAM: 1 GB - 16
Notable Features: Built-in cloud platform
GPIO: 40-pin header
Power consumption: ~5 - 12 W
Operating voltage: 5V

Raspberry Pi is by far the most widely used single board computer, and for good reason. It's a solid SBC running an easy-to-use operating system, with a lot of ports (4x USB, HDMI, camera port, dispay port, AUX, RJ45 Ethernet), and a huge ecosystem with lots of libraries and projects. It is basically the jack of all trades for hobby projects, almost any project can be made to work using a Raspberry Pi.

It also has integrated WiFi, making it a popular choice to use as a home server for all sorts of network applications, like ad-blockers, hosting websites or cloud storage. For smart home applications, a Raspberry Pi is often used as the hub that connects all the other devices with cheaper microcontrollers together.

Over the years, a lot of versions have been released. The general trend is that the newer versions get more and more power, at the cost of a much higher price. The newest versions like Raspberry Pi 5 are made for projects needing an AI algorithm or something else heavy running. If your project only requires basic computing, I'd recommend going for 3B+, as it is a lot cheaper and still has all the things you'll probably need.

The biggest drawback of Raspberry Pi is that it can't produce fast precise signals like the simpler microcontrollers. If you want to precisely control pretty much anything that requires a PWM signal, like a servo, LED light or BLDC controller, chances are that Raspberry Pi won't control it in a smooth way. This is normally solved by using the Raspberry Pi in combination with a simpler microcontroller like a Pico to generate precise signals, but there is no seamless way to get the Raspberry Pi and Pico to communicate with one another and it can be a huge pain to get working.

***When to use**:* Raspberry Pi is extremely versatile, and is often the first choice for projects that require more capabilities than a cheap microcontroller can provide.

Beaglebone Black – SBC and microcontroller in one

Price: $55-$70
Difficulty: Medium
CPU: Single-core 1 GHz + 2 PRUs
RAM: 512 MB DDR3
Notable Features: 2 real-time PRU cores
GPIO: 65+ usable pins
Power consumption: ~1 - 2.3 W
Operating voltage: 5V (3.3V logic)

You know how I said earlier that sending precise signals with a Raspberry Pi usually means you have to connect it to another microcontroller like a Pico? Beaglebones main advantage is that it solves this problem.

It has two programmable real-time units, PRUs for short. These are essentially like two small additional microcontrollers in addition to the main CPU, that can be used to handle loops that need to run fast or send precise signals. This means that you can run fast control loops, control LEDs, BLDC motors or servos without needing any additional microcontrollers.

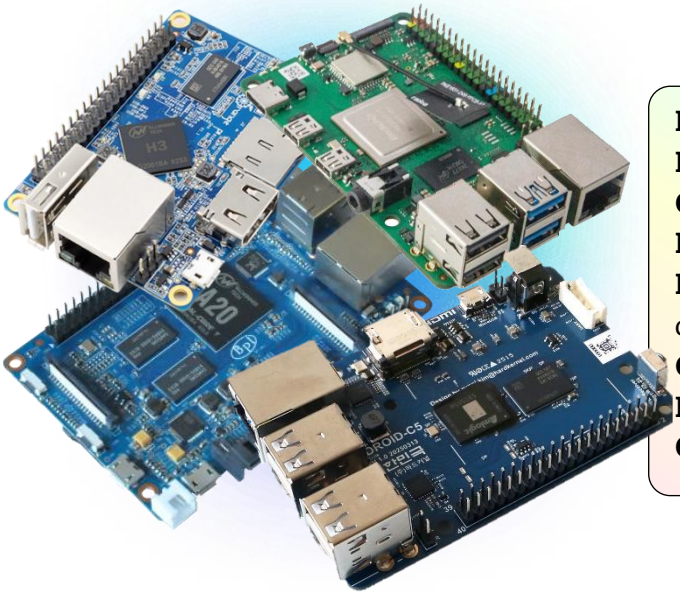
So you get the best of both worlds: the fast deterministic controls of microcontrollers and the advanced capabilities of single board computers.

The cost is that the ecosystem is a lot smaller than it is for Raspberry Pi, and it is generally harder to set up and use.

The PRUs makes it a popular choice in industrial automation and control systems. If you for example want to build your own robot arm, 3d-printer or CNC machine, this is one of the most used options.

***When to use**:* When you need a single board computer but also need real time precision. Let's you use one device instead of a SBC and microcontroller. Popular choice in industrial automation and robotics.

Orange Pi / Banana Pi / Radxa / Odroid – Cheaper and better Raspberry Pi?



Price: $15-$80+
Difficulty: Medium
CPU: Varies
RAM: Varies
Notable Features: More performance per dollar, lots of unique features
GPIO: Varies
Power consumption: Varies
Operating voltage: 5V (usually)

After Raspberry Pi was released, a bunch of other companies have made their own SBCs that have basically identical capabilities, but more performance at a lower price. Orange Pi, Banana Pi, Radxa and Odroid are some of the biggest brands in this space. Generally, these brands do give you more performance per dollar compared to Raspberry Pi.

The drawback is that these devices have way smaller ecosystems, and the OS is usually less polished and harder to use. All in all, these devices are less consistent and will probably require more tinkering to get working properly.

These companies have many different versions each, and new versions are released all the time - there are dozens of concurrent SBCs targeting very different niches. I won't even pretend to know what versions are best to choose, but if you invest some time, chances are you can fins an SBC that fits well with your exact needs.

***When to use**:* If you want maximum performance per dollar and don't mind the additional complexity and tinkering to get it working, these brands are probably a better option than Raspberry Pi.

Nvidia Jetson – The AI SBC

Price: $55-$70
Difficulty: Medium
CPU: Single-core 1 GHz + 2 PRUs
RAM: 512 MB DDR3
Notable Features: 2 real-time PRU cores
GPIO: 65+ usable pins
Power consumption: ~1 - 2.3 W
Operating voltage: 5V (3.3V logic)

Nvidia Jetson is built for projects that need serious AI computing. Instead of optimizing for the cheapest, most flexible general-purpose computer, Jetson boards are optimized to run AI models. They can do tasks like object detection, image recognition, or robot navigation directly on the device, in real time, without needing an internet connection or a cloud server to do the heavy lifting.

The reason they can do this is that Jetson boards pack a genuine Nvidia GPU - the same kind of chip that powers gaming PCs and AI datacenters, just scaled down.

That extra capability comes with tradeoffs. Jetson boards cost noticeably more than a Raspberry Pi or Orange Pi, use more power, and their pin-header ecosystem for hooking up simple sensors and switches isn't nearly as mature or beginner-friendly.

***When to use**:* If you want to make anything that requires advanced AI capabilities, like an autonomous vehicle or humanoid robot, this is probably the best choice.

P.S.
I hope you enjoyed the guide! It was manually written, because for things like this the AIs often yap a lot without cutting through the noise and actually showing the unique quirk of each thing. So it took a while to make, hopefully you found it useful <3

Since you downloaded this, you're probably somewhat interested in electronics and robotics and DIY projects and whatnot.

We make kits so you can make your own 3d-printed underwater drone that goes 100m deep, check it out if you want: https://www.manafishrov.com