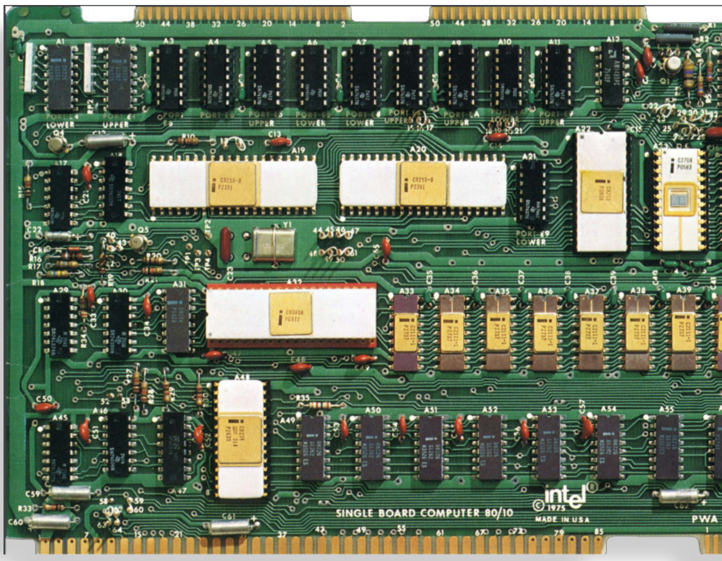
CHOOSING THE RIGHT DEVELOPMENT BOARD

The zeitgeist of 1976 may well have been Punk Rock, but in March of that year a small electronics company called Intel released a single-board computer product called the SBC 80/10 that integrated all the support components required by the 8080 microprocessor; including 1kbyte RAM, 4kbytes user-programmable ROM and 48 lines of parallel digital I/O with line drivers, plus a bus connector for expansion onto a single board. There was a Software Development Kit hosted on an Intel Intellec MDS 800 microcomputer development system, which also supported the 'ICE-80' in-circuit emulator for debugging.

What Intel were providing was both the first OEM single-board computer (SBC) and a development board with software development kit (SDK) all rolled into one. Which goes to show that the development-kit zeitgeist has been around longer than you may have expected: since the beginning of the microcomputer revolution!



Why Use A Development Board?

So why have development kits been around for so long? Why would you use one? To answer these questions, we need to take a step back and define what we mean by a development board.

When we talk about a development board, we mean a PCB built around a target device, usually a microcontroller for our purposes, designed to give a user easy access to the myriad features the device may incorporate. These boards will almost always include additional memory, Bluetooth/WiFi, input-output headers and ports for I2C, SPI, cameras, etc., LEDs, switches and a programming device connected to USB; allowing the board to be powered and firmware to be downloaded and debugged from a development PC.

Variations on the theme are evaluation boards, which tend to be simpler boards (incorporating fewer extras and sometimes just a peripheral device) and demo boards, which are usually set up to highlight specific capabilities using a demonstration application. There's actually no hard distinction between these board types, or indeed, a dev board and an SBC, though the latter will tend to sport a microprocessor and memory capable of running a full-blown operating system.

Why do manufacturers make these boards? It's no secret: chip makers want to sell silicon. As much as they can. To get you to choose their silicon over anyone else's for what (hopefully) will be the next big thing, they want to make the decision to choose them as frictionless as possible.

The aim of the development board is to flatten the learning curve for deploying an application on the target device by as much as humanly possible; allowing the user to go from zero to hero in a matter of hours rather than weeks.

Advantages

Modern dev kits give developers a host of advantages unavailable to their engineering forebears, not least of which is the ability to rapidly put together a proof-of-concept prototype for a minimum viable product (MVP) without ever touching a breadboard, let alone a soldering iron.

This is because most boards have an array of on-board peripherals and at least one (and usually multiple) standardised ecosystem expansion connectors that allow users to simply plug-in their choice(s) from a bewildering array of sensors, actuators and other analogue or digital peripheral devices. These are well-designed and fully-tested functional modules that use the likes of MikroElektronika™ mikroBUS connectors or Digilent Pmod™, SparkFun® Qwiic® and SeeedGrove® system connectors for expansion.

Don't have enough expansion connectors? If you're using a popular dev board, you can probably get a 'shield' board that will connect onto the GPIO header and provide extra expansion slots through GPIO lines.

Rapid prototyping has multiple benefits in itself. The most obvious is a reduced time to market for promising ventures. The flip-side is the ability to fail quickly: any projects that turn out to have been 'a good idea at the time' but are really on a hiding to nothing, can be retired early with a minimal loss of investment in time and resources.

This is also an early opportunity to evaluate candidate microcontrollers and peripherals for their suitability in the intended application. For example, a sensor may turn out to have much lower resolution in the down-and-dirty of real life than the datasheet might suggest. Maybe the processor is struggling to meet the demands of the MVP, offering no ability to scale the capabilities of the application? Alternatively, the processor may be way over spec, meaning you could save 50₵ per device by using its little brother – that will add up over a million units. All of these kinds of discovery allow course corrections to be made early, at minimal cost.

Hardware

It's unlikely that a user will incorporate a development board into their final application: they are generally too bulky, power-hungry and insecure with all those extraneous bells and whistles that aren't being used. After all, the point of the board is to give a user access to as many of the target device's features as possible. But that doesn't stop the development board from still being useful to the final application.

In the pursuit of making things easy for the user, chip makers will generally publish all aspects of the board design, from schematics and part lists, to gerber files for the layout. This can make many aspects of the final design 'cut and paste' from the original dev board design. Even if the user intends to incorporate different supporting components (like DRAM or flash) from those found on the development board, being able to compare against an existing design can speed up the design cycle. It may also spare the user some wire mods on their first PCB spin, by highlighting a design foible for an unfamiliar component that isn't obvious from the part documentation.

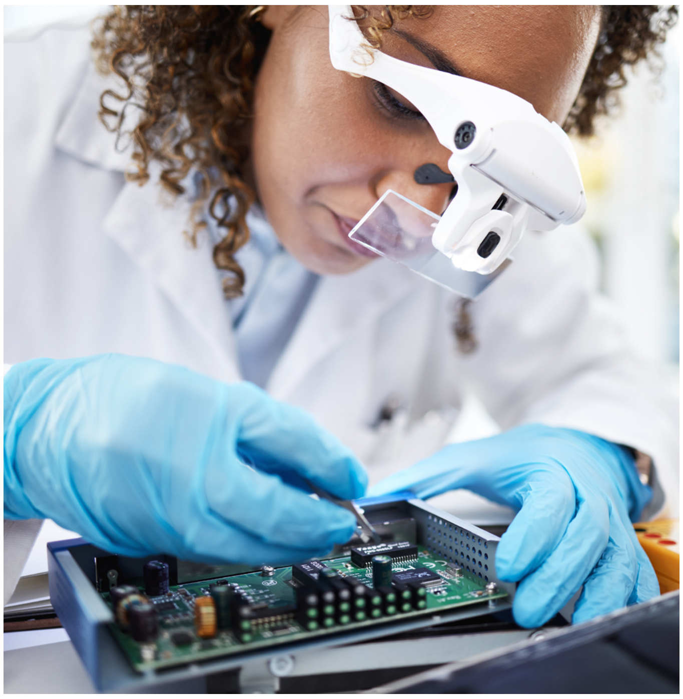
Software

The reality for many applications is that, aside from the seductive casing, the main differentiator between competitors is not the hardware (which often has near-identical capabilities) but the firmware deployed on the microcontroller. Whether it's the richness of the features or their novelty that separate a product from the also-rans, the speed and ease of deployment depends heavily on the software development kit (SDK) for the device and, by extension, the dev board.

While it's still possible for a hardcore old-schooler to build cross-compiled projects, using makefiles, from the Linux command line, most chip manufacturers have invested heavily in visually appealing integrated development environments (IDEs). Some are based on existing extensible IDE's with wide acceptance, like Eclipse; where others are ambitious projects, unique and exclusive to the particular manufacturer.

Many of the larger manufacturers are also supported by third party development software suppliers like Kiel, Segger, IAR Systems and Softlog Systems. From a user perspective, using an Eclipse based IDE or third-party toolset has the advantage of transferability: when you move from one device to another, you won't have to learn a new programming environment as all the tools are right where you expect to find them. This saves a lot of time as you get to grips with the different register set and memory map on your new device.

Whether the application is going to run on bare metal, an RTOS or embedded Linux, trialling a dev board's SDK is an early opportunity to decide if the libraries, drivers, APIs, debugger and any hardware abstraction layers are a fit for what you are trying to accomplish. If nothing else, you can decide whether you like them or not – and let's be real: you're going to be using these tools for some time, so it's a good idea to actually like using them.



Examples

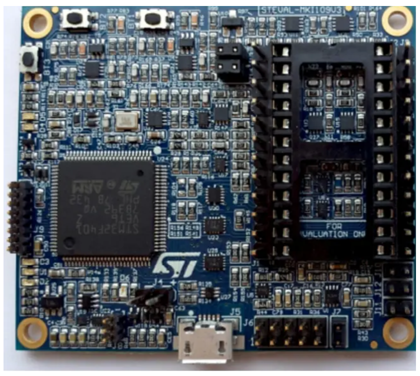
The scope of development kits available is truly mind boggling but to illustrate a little of what we have been discussing, it seems reasonable to shine a light on a trio of examples.

ST Micro STEVAL-MKI235KA

First up is an evaluation kit from ST Micro, who are well known in the industry for the quality of their Discovery and Nucleo development Kits for their ARM based microcontrollers. However, we're going to look at the STEVAL-MKI235KA Accelerometer Sensor Evaluation Kit.

This is an eval kit parred down to the components needed to highlight the functionality of a sensor device. In this case, the sensor is the LIS2DUXS12 which is a MEMS 3-axis linear accelerometer. But the device also includes an ultra-low power ASIC that provides an always-on antialiasing filter, a finite state machine (FSM) and machine learning core (MLC) with adaptive self-configuration (ASC), and an analogue hub / Qvar sensing channel. This kit especially highlights Qvar sensing and gives the user a lot of scope to experiment with the technology.

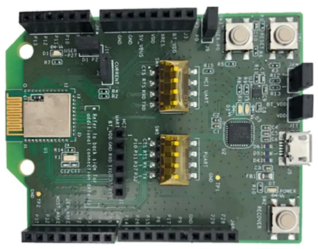
To break-out all this functionality, the eval board is laid out to fit a standard DIL24 socket, like the one found on its intended host, the STEVAL-MKI109V3 which is a MEMS Adapter Motherboard compatible with all ST Micro MEMS eval modules.



The motherboard is based around an ST Micro STM32F401VE ARM Cortex-M4 microcontroller with DSP and FPU, allowing a user to put any of the available sensors through their paces. You could also use a X-NUCLEO-IKS01A3 expansion board, if you already have a STM32 Nucleo board.

This is all supported by the X-CUBE-MEMS1 expansion software package for ST's STM32Cube software development tools and includes drivers that recognize the sensors and can collect temperature, humidity, pressure, and motion data.

Infineon CYBT-243053-EVAL



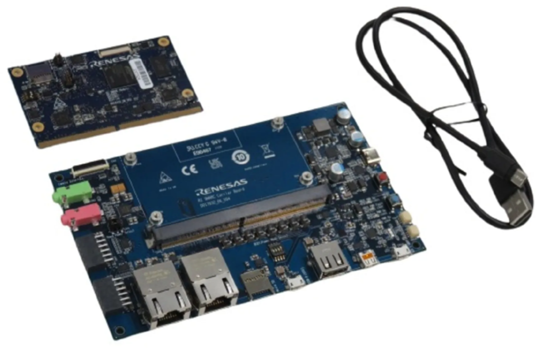
Moving up a little in terms of eval board complexity, we have the CYBT-243053-EVAL board which is an Arduino Uno sized module (complete with Arduino-compatible headers that can be used with a compatible shield) that showcases the AIROC™ CYBT-243053-02 system on a chip (SoC).

This SoC is a Bluetooth® 5.0 module which has support for Basic Rate (BR), Enhanced Data Rate (EDR @ 2/3 Mbps) and extended SCO (synchronous connection orientated) links. There is an Integrated onboard crystal oscillator and Arm® Cortex®-M4 core with 256-KB on-chip Flash and 176-KB on-chip RAM to run the royalty-free Bluetooth® 5.0 stack. All the usual I/O is supported: 22 GPIOs plus I2C, SPI, I2S, UART, and PCM interfaces. It also includes a Bluetooth® SIG qualified, integrated PCB trace antenna.

The eval board provides an instant way to connect the AIROC™ CYBT-243053-02 to a development PC and the AIROC™ Bluetooth® SDK within ModusToolbox™ Software which has a lot of example code to get you up to speed in record time.

Renesas Electronics RZ/Five Evaluation Kit

Our final example is the RZ/Five Evaluation Kit, which is at the higher end of both cost and complexity. This kit consists of a carrier board designed to break out the connections from a SMARC v2.1 module board.



SMARC modules are small form factor, low power computer Modules. In this case the module supports a Renesas RZ/Five (64-bit RISC-V CPU running at 1.0 GHz) with 1GB DDR4 main memory, 64GB eMMC Memory, 16MB QSPI NOR FLASH and a microSD connector. The module has an edge connector that fits to the carrier board where a number of standard connectors can break out the signals. These include Gigabit Ethernet (x2), USB2.0, USC-C (power), PMOD (x2), audio in/out and more. This is all supported by the RZ/Five Board Support Package to get the software development up an running.

The carrier board can also be used to evaluate Renesas ARM based SMARC modules.

Conclusion

Development boards offer the shortest path to application success that any chip maker can provide. In many ways though, when comparing similar target devices, the choice between manufacturers is going to be subjective for the user. Having used similar devices before is often the biggest influence. Then there is the subjectivity of whose development tools 'feel' best to you.

The great thing is that dev boards give you the opportunity to easily try various options for their functional fit and 'feel' – often at very low cost – before making that grand commitment to a product and software tool chain.