Open Source Projects Have the Power to Change the World

By **Mark Patrick** — Mouser Electronics

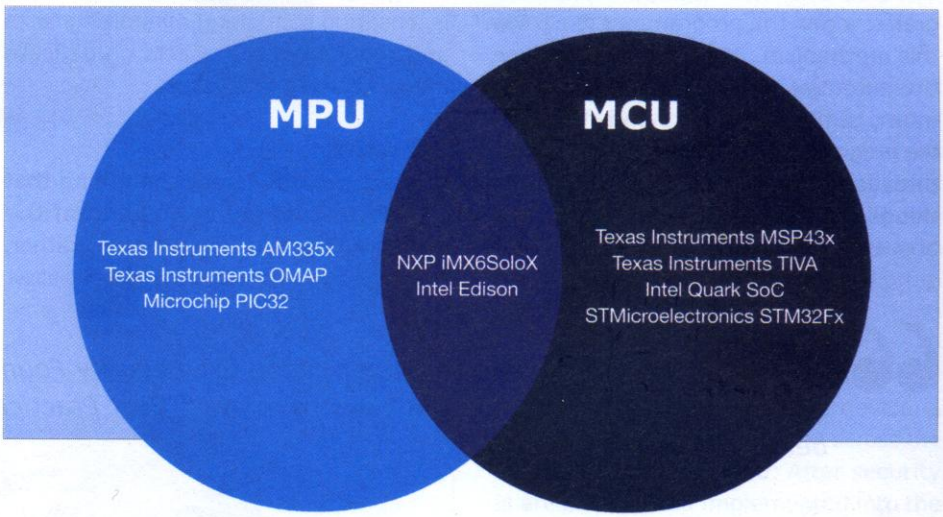


Figure 1: The MPU versus MCU divide.

Creativity has a new champion; open source. Access to 'free' software, along with low cost hardware is enabling anyone to realise things that they could have only previously dreamt of and this has the potential to help change society for the better. The realm of open source can be both inviting and confusing; inviting because it really is as good as it sounds and confusing because, well if something sounds too good to be true, then it normally is, right?

The basic concept started within the software sector through the work of altruistic communities, such as the Free Software Foundation [1] and the GNU Project [2]. Today, that concept has spread to hardware and to the system level in general. The idea of having access to intellectual property (IP) without an associated cost on which to base a commercial product is still largely limited to technology-oriented sectors, but the momentum behind it continues to grow.

There is now, of course, an unapologetically commercial aspect to some open source projects and each pursues its own objectives, but it remains true that if something is offered as open source it gives one the liberty to use, change and distribute that inherently valuable IP at essentially no cost. Developers should not ignore the issue of licensing, the way in which the underlying IP is protected for the good of the community. Licenses aren't, however, intended to impede innovation and it is in innovation that open source excels.

The platform sells the project

Licensing issues aside, open source projects appeal to the maker community for a number of reasons, one of which will be their extensibility. They provide the perfect platform on which to build an application and it's really the concept of a platform that sells the project.

An appropriate analogy is the smartphone app store; an app developed for a smartphone using the manufacturer's development environment and programming framework (the 'platform') removes a large part of the effort required in realising an app (the 'project') which can then be distributed through the store. The same is true for open source projects. It's not enough to develop a single-board computer and make it freely available, although that's how many projects started. Makers flock towards an environment that supports the development of their own project, without requiring them to build the platform. This is a sliding scale, however, as some open source projects are more complete platforms than others.

One thing that is undeniably true is that open source projects lower the barriers to entry for everyone, which is encouraging people with completely unrelated skill-sets to start programming and developing projects not just for their own amusement but to meet their specific requirements. This could be described as the evolution of open source projects and the democratisation of technology. There are many examples of people using popular projects like Arduino and Raspberry Pi to realise any number of projects. The ones that grab most headlines are those that further progress research in a particular area of science. These often involve some form of data collection, actuation and processing - the 'perfect storm' of open source projects.

It is significantly easier for someone to develop an experiment based on a project like Arduino that uses actuators and sensors, than to design something that needs to be controlled by a regular PC. It is this openness that is responsible for the popularity of open source projects today.

Getting started

The cost of hardware targeting the maker community is intentionally low; the whole point of the projects is to encourage people to experiment with them, so price shouldn't be an obstacle. The real investment needed to start experimenting is time, much of which will be spent on building the knowledge needed to do something meaningful.

Software and hardware are inextricably linked in open source projects; in many ways it defines them. It is inevitable that getting a concept into the real world using open source solutions will require low-level code at some point; setting bits in a register to turn LEDs on or start a motor. This kind of low-level activity can only be achieved using software that is 'aware' of the hardware and, in turn, requires the maker to be familiar and comfortable with working in both domains.

How low-level code is implemented is probably the best way to differentiate between open source solutions, and is therefore a good method for a maker to decide which project is most appropriate for them. If the maker is comfortable with using or learning the Linux environment it opens up a range of possibilities based on highly capable microprocessors (MPUs). If, on the other hand, the idea of building a Linux distribution does not appeal to the maker, then the alternative is to focus on the projects based on less powerful but no less capable microcontrollers (MCUs).

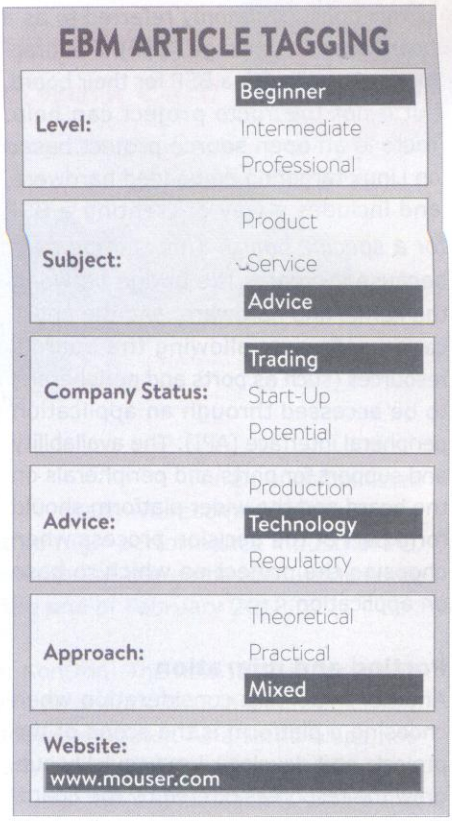
The MPU *vs.* MCU divide is illustrated in **Figure 1**, which categories some of the boards available based on the processor employed. In short, if a board uses an MPU then it's likely that Linux will be the (suggested) operating system at the heart of the software platform. Developing applications is, at least in theory, simpler; they can be written using a host system (such as a PC running Linux) and ported to the target. However, because the target may not be based on an MPU with the same architecture as the host it introduces the need for a cross-compiler, which takes an application written for one MPU and compiles it to execute on another (the target) processor architecture.

If it is an MCU then the options are slightly wider; it may be based on an open source project such as Arduino or mbed, or it may use the manufacturer's own software platform. In each case, the software development environment will be specific to the project.

In some cases, it can be a combination of several options. The Intel Edison Development Platform [3] a good example; based on the Edison SoC it supports both Arduino and non-Arduino breakout boards, which means the low-level code can be created natively in C or generated using the Arduino programming language and development environment. An overview of currently popular boards is given in **Table 1**.

MANUFACTURER	BOARD	PROCESSOR
BeagleBoard.org	BeagleBoard-xM	Texas Instruments OMAP3530
BeagleBoard.org	BeagleBone Black	Texas Instruments AM3358
Diligent	ChipKit uC32	Microchip PIC32MX340F512H
UDOO	UDOO Neo	NXP i.MX6SoloX
Intel	Edison	Intel Edison SoC
Texas Instruments	Tiva C LaunchPad	Texas Instruments Tiva C TM4C123GH6PM
STMicroelectronics	STM32F4 Nucleo	STMicroelectronics STM32F4
Texas Instruments	MSP430 LaunchPad	Texas Instruments MSP430G2
Intel	Galileo	Intel Quark SoC X1000

Table 1: Overview of development boards and platforms currently popular in the maker scene.



Ports and peripherals

A fundamental part of the maker movement is using technology to interact with the real world, which translates to extending the platform through its ports and peripherals. How this is achieved depends partly on what the platform offers and what the community supporting the platform can provide.

Arduino [4] is an extensible platform that uses shields (add-on boards) and sketches (software) to create bespoke solutions. Arduino boards come in various forms offering their own peripherals, but when that's not sufficient the board can be extended using shields that connect via on-board headers. The development environment recognises and supports official shields making it easier to exploit the added functionality.

Boards based on MPUs running Linux will also include standard features such as wireless (Wi-Fi, Bluetooth) and wired (Ethernet) connectivity. Part of the appeal of Linux is that it already supports many such standard features, making it easy for applications to access them. For non-standard hardware features support will need to be provided using a library

of functions, commonly referred to as a board support package (BSP). Manufacturers may provide a BSP for their board, but if not the Yocto project can help. Yocto is an open source project based on Linux targeting embedded hardware, and includes a way of creating a BSP for a specific board. This is important because it creates the bridge between the kernel and hardware, and the application software; allowing the board's resources (such as ports and peripherals) to be accessed through an application peripheral interface (API). The availability and support for ports and peripherals on the board and the wider platform should form part of the decision process when choosing the project on which to base an application.

Porting and migration

Another important consideration when choosing a platform is the scope of the project; as it develops it may quickly outgrow the resources offered by the board. This can present more of a challenge if the project is a board based on an MCU, less if the board is MPU based. Typically, MPUs have considerably more processing power than MCUs, but more importantly the MPU's architecture is likely to support Linux - making it easier to port the application to a more powerful processor. This is one of the key reasons to choose Linux as the software platform.

The instruction architectures used in MCUs varies by manufacturer, but two tend to dominate in the maker environment; AVR and ARM. While Atmel owns the AVR architecture, many semiconductor manufacturers have licensed ARM's IP to create a much wider range of MCUs. Arduino was conceived using the AVR core architecture and includes boards based on both 8-bit and 32-bit cores, but in recent times has extended its scope to include MCUs based on other architectures - including ARM.
Beyond Arduino, open source MCU boards are more likely to be based on the ARM architecture and many will support ARM's own software development environment, mbed. A common hardware and software platform makes it much simpler to port or migrate an application to a board that offers higher performance or more peripherals.

Taking the next step

The number of open source projects now available can be bewildering but choice is never a bad thing. Mouser supports open source projects from communities such as Arduino and UDOO, commercial suppliers such as Adafruit and Seeed Studio [5], and manufactures including Intel, ST and NXP. Some projects, like BeagleBoard [6], are championed by a single manufacturer supporting a community focused on a particular product family. Having a high, 'system-level' vision of what an application entails is a great place to start, as it will help decide which platform to choose. For example, if the application calls for cellular connectivity with GPS it may suggest using the Adafruit FONA 808 Cellular & GPS Shield [7], which is compatible with Arduino baseboards.

Compatibility with the Arduino 'standard' is now commonplace, as it has the ability to bring together multiple open source projects in a single application. A great example is the UDOO Neo [8]. Based on an i.MX 6SoloX application processor, it runs Android on Linux while also supporting Arduino sketches and shields. Open source boards are now being used as a low cost alternative to expensive test and measurement equipment, in a range of environments. Their affordability means they can be more cost-effective to deploy than their industrial-grade counterparts, making it possible to put electronic sensing/data acquisition equipment practically anywhere. The caveat here isn't just durability but, perhaps more significantly, it can be one of precision. Professional test equipment is calibrated to give a guaranteed degree of accuracy, while a maker version may not be able to offer the same levels of reliability or repeatability. However, some data is better than no data at all and in this respect the prospect of creating application-specific equipment is of real value to the next generation of scientists — enabling them to discover new things about this world and beyond.

Conclusion

The rise of low cost open source hardware and software projects is already leading to new discoveries. It is making it accessible to anyone in a way that has never happened before and has the potential to literally change lives. Mouser is supporting the maker community [9], not only by stocking a wide range of hardware components but by also offering a vast amount of information and educational content to accompany them. This is allowing makers of all abilities to develop their knowledge quicker, helping them realise their application faster. ⏮

The Author

Mark Patrick joined Mouser Electronics in July 2014 having previously held senior marketing roles at RS Components. Prior to RS, Mark spent 8 years at Texas Instruments in Applications Support and Technical Sales roles. He holds a first class Honours Degree in Electronic Engineering from Coventry University.

Web Links

[1]. Free Software Foundation: www.fsf.org

[2]. GNU Project: www.gnu.org/gnu/thegnuproject.en.html

[3]. Intel Edison Development Platform: www.mouser.co.uk/new/Intel/intel-edison

[4]. Arduino products: www.mouser.co.uk/arduino

[5]. Seedstudio products: www.mouser.co.uk/seeedstudio

[6]. BeagleBone products: www.mouser.co.uk/beagleboardorg

[7]. Adafruit products: www.mouser.co.uk/adafruit

[8]. UDOO Neo products: www.mouser.co.uk/udoo

[9]. Maker community support: www.mouser.co.uk/applications/open-source-hardware