DOI 10.5162/ettc2020/2.2

Using Single Board Computers in FTI console

Óscar Gigato Rodríguez
Airbus, Paseo de John Lennon S/N, Getafe, Madrid (Spain)
oscar.gigato@airbus.com

Abstract:

The born of the first Raspberry PI computer in 2012 created a new market for low cost ARM based single board computer (SBC), initially for teaching purposes; but nowadays, after an amazing increase of computing power while maintaining costs, makes possible their use for many purposes. This incredible evolution allows their use for many tasks, including certain data processing workloads.

A ***cheaper***, scalable, ***modular*** and more ***energy efficient*** system is proposed for onboard FTI consoles. It provides also an easy to use frontend dynamic web ***interface***, usable from a laptop or a tablet when A/C is on ground.

Several services will be provided with this solution, like online/offline monitoring through FxS dataservers, data recording and so on. Anyway, almost every service necessary onboard can be run in these devices, which will be hosted in a distributed environment.

Key words: scalable, modular, energy efficient, web interface, cheaper

Current FTI consoles needs/services

Today's aircraft FTI consoles provide multiple services, such as data recording, monitoring and so on. In Airbus Defence and Space, several console models are designed, most of them taking into account many criteria, like monitoring displays use, data analysis, FTI data recording, either in RAW or processed mode, etc. Cameras control (when present) is also a typical work of these consoles.

Although there are many approaches to provide FTI consoles services, all solutions normally include the use of one or various PCs. These PCs run several services, like data recording, data exploitation utilities, FxS data servers and some adoc processes.

Advantages of changing to Single Board Computers

The two main advantages of using these mini PCs are cost and energy efficiency. Other benefits for using this technology are: heat dissipation, quick replacement and upgrade, modular design approaches, etc.

- Cost: SBCs are really cheap. For example, talking about the successful *Raspberry PI*, a board with 4 GBytes of RAM, 4 ARM Cortex A72 processing cores, WiFi, Bluetooth LE, two USB 3.0 ports, etc, costs less than 100€, taxes included.
- Energy efficiency: Carrying on with the Raspberry PI example, last model (4B) consumes a maximum of 15W, but it normally drains, under load, a bit more than 7W. This includes peripherals, as CPU only drains as much as 7,6W.
- Heat dissipation: Although last RPi version normally need a fan, previous versions like 3B+, or other SBCs do not need it. Only with a heat sink is enough. This is not a really huge advantage, because many low power PCs can be found in the market that do not need active cooling (normally, ultra-low voltage versions).
- Modular design approaches: The little size of these computers allows the use from one to 2 or 4 devices (or even more). As these SBCs have at least 4 processor cores, with 4 devices installed a total of 16 cores are available,

which is indeed a very powerful distributed computer.

Drawbacks of SBCs

Of course, using these devices also have some disadvantages. We are going to enumerate some of them here:

- Compatibility: Although there are SBCs based on x86 architecture, the vast majority of them are based on ARM. This fact has some advantages, as mentioned in the previous paragraph, but existing legacy software has to be recompiled and tested.
- OS availability: As mentioned above, most of SBCs are based on ARM architecture. Although Microsoft is developing an ARM version of Windows 10, is only available on certaing SOCs of ARM (Qualcomm Snapdragon). A speacial version of Windows, Windows IOT, is present in Raspberry PI board. Without graphical envirnment, is suitable for certain projects. When using a x86 platform, OS availability is higher.
- Hardware limitations: Some SBC models are limited to 1 Gbyte of RAM memory. This, that should be enough for a majority of applications run in a FTI consoles, is a limit for certain software.
- CPU raw power: Compared to x86-x64 models, ARM processors are more efficiency oriented.
- Ethernet connections requirement: Each node of this system requires its own network connection. So, if we configure a four node system, we will need four network sockets available.

Use of GPU for parallel computing

A common point for all SBCs is that, based on a SOC (System On a Chip), is the integrated graphics core, with 3D specialized hardware. As this chip is designed for massive parallel operations, and can be used by programming standards like OpenCL for performing this kind of calculations. Some libraries for C++, for example, boost::compute, can bring this power closer to the programmer. Of course, it's necessary to install a driver first.

Although there are other hardware acceleration libraries for GPUs, like nVidia CUDA or ATI Stream, this one (OpenCL) is a multiplatform and open solution, so it availability is more spreaded.

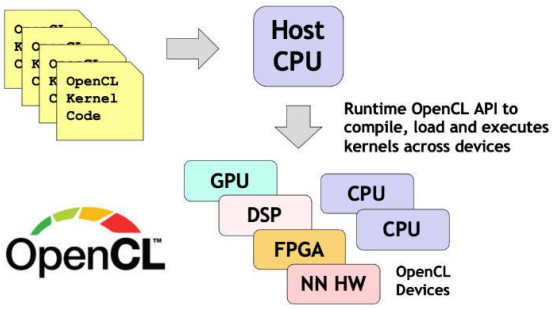

Figure 1: OpenCL task dispatching

This library allows parallel code to be executed in CPU, GPU or other devices, like FPGAs, DSP, etc.

The possibility of using this hardware acceleration is also available in classic PCs, but is mentioned here to realise that there is a lot of power available in these little SBCs. For example, there is an implementation available for RPI 3B+, and another one in development for RPI 4 (GPU of this model is different, and more powerful, than GPU of RPI 3B+).

In this use case, we propose (initially) to use OpenCL for certain heavy calculations, that are some times performed when using calculated parameters. Of course, software has to be developed taking in mind this approach, to make the most of all available execution units, that is, CPUs + GPU.

Modularized system approach

To fit FTI consoles needs, for Airbus Defence and Space, the initial proposal is based on a modularized design. That is, a system built of one, two or four SBCs working together like a unique system. This allows a great amount of scalability and flexibility that eases adaptation to a variable number of scenarios. For example, a basic system of only one SBC is possible for giving basic services, and grow when needs scale up.

As a common denominator, a web server must be present in at least one node, just to host the configuration service.

Advantages of this approach are crystal clear. System can grow according to project needs, and it can also decrease when too much power is not necessary.

The main disadvantage of using this design is the increased complexity, but it is totally affordable and it really worths. This kind of approach can also be adopted when using classical PCs, although this is out of the scope of this paper.

Hardware configuration

Many SBC devices are available in the market, and most of them can fit the requirements for their use in FTI consoles, as today regular PCs are performing these tasks. It is true that there are some devices specially designed for their use in industrial environments, like NxP products. But, as commented above, regular PCs accomplish these tasks until now, so there is no special requirement for industrial SBCs.

Three different devices have been tested by us:

- Raspberry PI 3B+: With Broadcom BCM2837B0, 4x ARMv8 Cortex A53 at 1.4 GHz. 1 GByte of LPDDR2 RAM. Gigabit Ethernet over USB2 (max 300 mbps), WiFI 802.11ac, Bluetooth, microSD of 8 gigabytes. USB 2.0.

Figure 2: A raspberry PI 3B+

- Odroid C2: Amlogic ARMv8 Cortex A-53 quad core CPU at 1.5 GHz, gigabit Ethernet, 2 GBytes DDR3 RAM. eMMC module. USB 2.0.

Figure 3: An Odroid C2

- Raspberry PI 4: Broadcom BCM2711, 4x ARMv8 Cortex A72 at 1,5 GHz, 2 GBytes of LPDDR4 RAM, USB 2.0 & 3.0, WiFi 802.11ac, Bluetooth 5.0 BLE, microSD of 8 GBytes.

Figure 4: A raspberry PI 4B

- Raspberry PI Compute Module 3+: Not tested yet, but one of our favorites. This is a special version of the 3B+ model, but compacted to fit in a SO-DIMM module. This allows a custom made designed board to fit one, two on up to four of these modules. Moreover, it comes with a size configurable eMMC of 8, 16 or 32 GBytes. eMMC modules are much faster than microSD, so this option is really very attractive, although board development cost has to be taken into account.

Figure 5: A raspberry pi Compute Module 3+

Software configuration

In this chapter, a description of all environment, including operating system, base and management software,etc.

Operating system

Nowadays, there are a general availability of various flavours of Linux distributions for these machines, although a special version of Microsoft Windows, Windows IoT, is also an option. As we have tested many devices, a description of different choices will be explained here.

- Raspbian: A Linux distribution, now known as “Raspberry Pi OS”, is a debian based distribution for Raspberry PI. This is the more extended option for Raspberry PI. Backwards compatible, it runs from original Raspberry PI model

B to latest 4 one. A new 64 bits version is available, but is still beta version.

- Ubuntu 16.04: This version of Linux was installed in Odroid C2. It is also based in Debian. It's a 64 bits version, compiled for ARMv8 architecture (includes many instructions for specific purposes).
- Windows IoT Core: This version of Microsoft Windows was created for Raspberry PI 2, and it has been updated to run in newer versions of this hardware. It is designed for running only one application in the foreground. The main disadvantage of this proposal is that it has a license for commercial purposes.

Once described the three alternatives tested in these devices, let's speak about our conclusions.

- Advantages of choosing Windows IoT Core:
    - Existing WIN32 applications are easy to port.
    - DotNet apps are also portable. In boths cases, not all desktop API is available in Windows IoT Core.
- Disadvantages of Windows IoT Core:
    - Has a cost per license.
    - Only one foreground application is possible.
    - Closed source product.
- Advantages of choosing Linux:
    - It is a full desktop OS. Many applications can be run in foreground.
    - Open Source project with great support. This is especially true in Raspberry PI OS.
    - Hundreds of packets that can be installed for almost any purpose.
    - DotNet Core is available in Linux environmnent, and future developments seem to follow this path.
    - It is a very lightweight Operating System is well configured, ideal for this kind of devices.
- Disadvantages of Linux:
    - Applications need to be port to this operating system.
    - Additionally, when working with ARM architecture, some other kinds of changes have to be performed. For example, ARVv7 architecture does not permit an indirect access to memory.
    - Developing tools are more powerful in Windows environment. For example, Microsoft Visual Studio is a reference for developers. Although last versions are capable of developing and debugging using a Linux host, there is still a gap between the two platforms.

Comparison between Raspbian and Ubuntu has no sense attending to the orientation of this analysis.

Master/Slave

In this system, one node takes on the role of master. It will host the responsibility of managing the processes run in other nodes, and gather for the status of them. It will also host the web server for the managing application (commented ahead).

Services

The common services to be run in these consoles are the following:

- Data recording in RAW format: *dumpcap* (or similar). This is a very light application, whose main requirement is enough bandwidth to storing system. A USB or microSD can be used.
- Data recording in PFF (Parameter Fast File) format: Performed by our *pffrecorder* application, written in C++, requires a high IOPS storing device. Performed in a fast USB3 stick memory, or external USB3 SSD disk.
- *Online* FxS Dataserver: An application that reads IENA packets and serves parameters to any client that asks for them. Written in C++, runs smoothly in a Raspberry PI 3B+, managing a data flow of 60 Mbits per second.
- *Offline* PFF Dataserver: An application that access PFF files (recorded by *pffrecorder*) and serves data to clients. Is very lightweight.
- Other little tools.

Future Services (in development)

- Web server: Hosts the managing/configuration service. Uses dotNet Core technology.
- Web client: Will use *dotNet Blazor* technology. *Blazor* is a new framework from Microsoft that enables the use of C# code directly in a web page. Dynamic web pages can be built easily with this tool, and the use of dotNet Core, an open source implementation, available in many platforms, as x86/x64 Linux, ARM Linux for raspberry PI and others, guarantees a long term support to this technology. Code can be run directly in the browser (with WebAssembly), or in the server.

The optimal use of this infrastructure is using a WiFi connection. The ideal scenario would be turning on the WiFi when the aircraft is on ground. Then, the user will come to the aircraft with a tablet/cellular phone (a laptop would work too, of course). Opening the internet browser and connecting to the server would show the user interface of all the system. Different pages/tabs are planned for this application:

- System status: Will show a brief description of the system: number of nodes, active services, configuration in use, system alerts, etc.
- Services configuration: Will show services available, services configured to run, and associated node to run each service.
- Configuration database: For checking, loading and applying configurations (FTI parameters specification).
- Profiles/modularization: Each node can assume one or more roles. Depending on system expected load, services will be distributed among existing nodes.
- Software update: This tab will allow an update of software versions running in the system (binaries). It will also show current versions.
- Logs/restart: A tab where system logs can be viewed, downloaded and deleted. It will also show an option for system restart and/or shutdown.

The idea is to simplify the management of all the system in a user friendly application, easy to use and powerful at the same time.

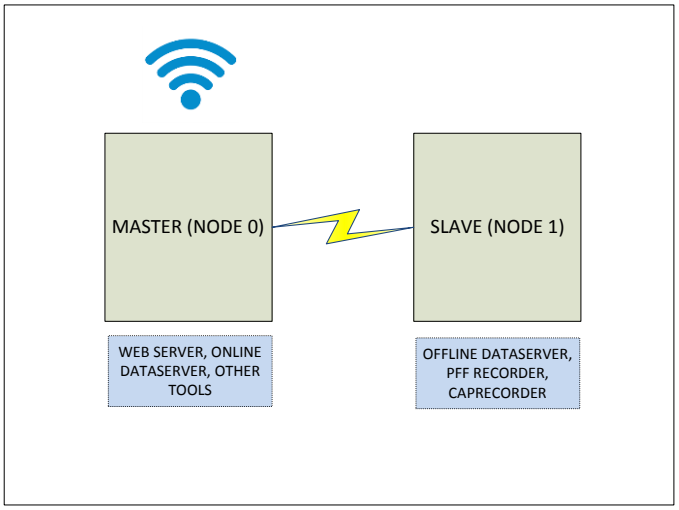

Figure 6: A typical system configuration

Test results

As we have port all base services to linux/ARM environment, we can resume our personal experience with these firsts testing stages.

- Speed: Several improvements have been made to our C++ source tree, in order to be more stable and faster. We have not seen any problem with speed, so our *Online* data server starts in about 9 seconds in a raspberry pi 3, with an A400M configuration (really big, more than 300k parameters). *Odroid C2* is a bit quicker, but barely noticeable. Raspberry PI 4 is much faster,

starting our application in a bit more than 5 seconds. Regarding CPU use, raspberry PI 3 managed with no problem a data flow of almost 60 Mbps.

- Memory utilization: Continuing with our example, *online* data server uses 274 Mbytes of RAM memory with no clients are connected (more than 300k parameters). So, thinking about its use in an onboard console, where clients demand no more than 3000 parameters, one of these SBCs will manage with *no suffering* all clients' requirements. As all these devices have 4 core processors, multithreading applications, like this one, will benefit of this configuration. There is still room for more processes in a raspberry PI 3 running the online data server.
- Multithreading: As commented just above, one of the key factors for the viability of using these devices is the multithreading programming approach. As IPC (Instructions Per Clock) of ARM Cortex A53 is much lower than regular PCs processors, due to many reasons, like in-order design vs out of order one, cache sizes and so on, squeezing each core is possible when multiples threads are running. Our services design are heavily multithreaded, and have been optimized for a slow footprint in both CPU and memory use.
- Energy consumption: This is where these devices bright when compared to x86 alternatives. As ARM architecture is designed primarily for energy efficiency, these processors have a really small footprint in energy requirements. For example, raspberry PI 4, the hungriest device of all tested, demands a *peak power* of less than 8W. This device is really powerful, as it features a 4 core out of order processor, and up to 8 GBytes of RAM (2 GBytes our sample). Two of these devices should be enough to running all of our processes smoothly and would consume a maximum of 30W, including all necessary devices connected to them. Moreover, two raspberry PI 3b+ or two compute modules should be enough too, with a bit less power requirements.
- Access to disk: There is only one application that requires a lot of IOPS to run properly. This is *pffrecorder*, and this is true specially when too many parameters at high bit rate are being recorded (like A400M). If we are using compute modules, internal eMMC will cover our requirements. Also, with Odroid solution, which uses eMMC, will also be enough. Raspberry PI 4, with their USB3 ports, will also perform well. The only case that would cause any kind of problem is Raspberry PI 3B+, because it only features USB2 ports.
- Upgradeable: Upgrading this kind of design is very easy, as all Raspberry PI have the same size since their first model, Raspberry PI Model B, born in 2012.

Cost analysis

SBC devices are really cheap. You can buy a Raspberry PI 4 with 2 GBytes of RAM for only $35. You can also buy a Compute Module 3+ with 32 GBytes of eMMC for only $40. But, this is not the only cost of the system. For building it, you need to be concerned by:

- Peripherals: microSD or external SSD/USB stick.
- Development costs: As our software was already port to linux, base software adaptation cost has been very low (mainly configuring environment and a few architectural bugs over ARM). Managing service development is pending, but this will benefit both PCs and SBCs architectures.
- Ad hoc hardware design: If Compute Modules 3+ (or more recent models when available) is going to be used, designing, engineering and testing custom boards will be the highest cost. But this development could be very useful, as it would be very compact and expandable.

Taking into account all these aspects, we can estimate the cost of a system based on SBCs:

Two Raspberry PI 4 + two microSD for booting + one external SSD (500 GBytes): aprox (including taxes in Spain): 188€.

Using a PC for these purposes is much more expensive (more than 1000€).

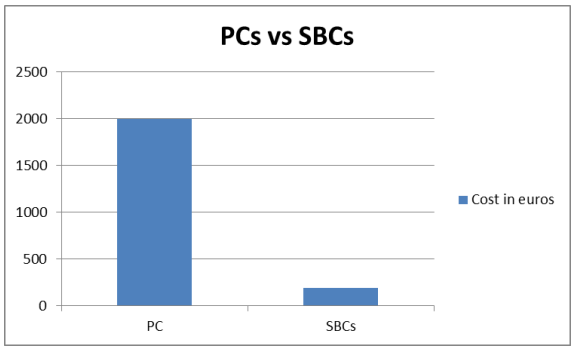

Figure 7: Cost comparison between PCs and SBCs

Conclusions

Single Computer Board devices have been a revulsive for “cheap” computation. Raspberry PI has founded a new category in computer devices. Originally thought as a teaching platform, its evolution has converted this “toy” into a real computer, powerful and cheap enough to handle very hard work. It is very cheap building a multi device system with a great power as a real alternative to classic PCs.

Last version of Raspberry PI can be configured with up to 8 GBytes of RAM, enough for storing medium databases. For our use cases, 2 GBytes is enough for handling with our processes. Even 1 GByte should work.

Using these SBCs can drop the cost of building a console for FTI, at least the PCs for computing tasks. The scalability of these devices is also a plus to have into account.