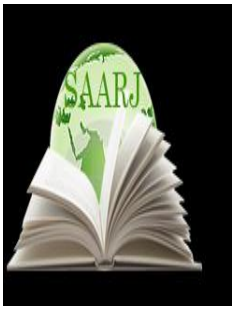
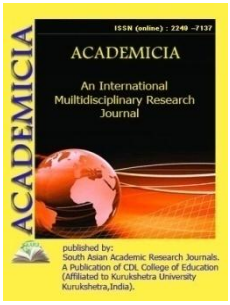


ACADEMICIA
An International Multidisciplinary Research Journal
(Double Blind Refereed & Peer Reviewed Journal)



DOI: 10.5958/2249-7137.2021.02184.4

AN ANALYSIS OF EMBEDDED SYSTEM DESIGN ASPECTS

Mr Mrinal Paliwal*

*SOEIT, Sanskriti University,
Mathura, Uttar Pradesh, INDIA
Email id: mrinalpaliwal.cse@sanskriti.edu.in

ABSTRACT

The use of embedded systems has increased at an exponential rate in virtually every area, from cars to household appliances to ICT(Information Communication Technology). Embedded systems and embedded computer processors have received a lot of attention in recent years, even in desktop application environments. The inherent advantages of embedded systems over conventional desktop computers, as well as the rapid growth in the computing capacity of embedded processors, are driving this trend. Rural application platforms are designed to run applications that are needed to provide various e-services and self-help services in rural regions. In terms of power outages, irregular grid power circumstances, high temperature ranges, changing humidity, dusty atmosphere, and so on, the eco system existing in rural regions, particularly in developing nations, is mainly difficult. Although embedded systems have many benefits, they can have significant drawbacks, especially in the context of application platforms. We explore and evaluate many elements and difficulties related to the design of embedded systems for rural application platforms in this article. We also talk about how to deal with the difficulties that come with designing an embedded system for a rural application platform.

KEYWORDS: *Application Platform Embedded System, Hardware, Network, Software.*

1. INTRODUCTION

In virtually all aspects of ICT (Information Communication Technology), there has been extraordinary development and growth in the past few of decades. Many public and commercial organizations have launched e-services projects throughout the globe, and many of them have been successful. After a closer examination, it becomes clear that the bulk of these effective efforts are limited to metropolitan regions, with success rates in rural areas being much lower. This is due to a variety of technological and commercial constraints. Designing an effective

application platform that is appropriate for the rural eco-system is one of the main technological difficulties to overcome for a successful e-service project. A relevantly optimized embedded system may be developed to meet the needs of rural application platforms[1].

1.1 Embedded system:

An incorporated system is a computer-hardware system with software embedded as one of its most significant components. It's a computer-based system devoted to a certain application or product. It may be a stand-alone system or a component of a bigger system. It does not need secondary memory like a computer since its software is typically stored in ROM (Read Only Memory). Embedded systems are categorized depending on their complexity, functions, and other factors. Small scale (or low end) embedded systems, medium size embedded systems, and high end embedded systems are the three main categories of embedded systems. Small scale systems are built around a single 8- or 16-bit microcontroller, with simpler hardware and software, internal memory (both ROM and RAM), and no operating system. They run on a single thread. These are usually low-power, and some are even battery-powered. A single or a few 16- or 32-bit microcontrollers, DSPs, or microprocessors are often used in medium-scale embedded systems. These are complicated in terms of both hardware and software. The majority of them feature some kind of external memory. High-end embedded systems contain a lot of hardware and software complexity, thus they may require a cluster of processors or customizable processors and programmable logic arrays.

These are used for complicated applications that need co-designing and integrating hardware and software in the final system. These high-end computers will feature several Megabytes of external high-speed RAM (such as DDRs) and solid-state storage memory. To achieve better speed performance, some functionalities such as cryptographic algorithms, graphic processing algorithms, video decoders, discrete cosine transformation and inverse transformation algorithms, network protocols, and network drivers functions are implemented in hardware (via specialized co-processing units). These run a full-fledged operating system (in certain instances, Real Time OS) and, in many situations, can link to high-end display subsystems. As applications become increasingly computationally complex and sophisticated, there has been a lot of emphasis and research interest in this type of embedded system in recent years[2].

1.2 Application platform:

An application platform, by definition, offers services to apps. The application platform makes accessible everything needed to run/execute a program effectively. This usually comprises hardware, device drives, and operating systems, among other things. The services provided by the application platform vary depending on the application. Display support, network connection, graphic and video capabilities are all required by certain applications, such as e-services. The appropriateness of an application platform for a particular collection of apps in a certain eco system or deployment scenario is also critical to a system's success. Because an application platform has so many characteristics and so many competing needs, optimization is a big problem[3].

1.3 Requirement specifications of rural application platform:

There are a number of criteria that a system must meet in order to be used as a rural application platform. The rural eco system scenarios are very demanding, which makes the application

platform requirements highly stiff and, in many instances, contradictory. If we attempt to improve one parameter, it's possible that other values may drift out of the optimum range. As a result, in order to optimize the system as a whole, all of the parameters must be balanced in such a manner that overall system peak performance may be reached. This scenario may be accomplished by weighting and balancing the parameters properly; however, this does not necessarily imply that all of the individual parameters are at their optimal levels when examined separately. Table 1 lists the general criteria that the application platform must meet. In order to meet the overall system performance, all of these criteria are stated in broad terms. When needs are stated at a greater level of granularity, there are more scenarios that conflict, and overall performance goals suffer. Computing power is often stated in terms of MIPS (Million Instructions Per Second) or MFLOPS (Million Floating Point Operations Per Second), however these low-level metrics do not guarantee that the necessary functional requirements are fulfilled.

This is because the embedded system in question may include a SoC (System on Chip) and other components with specialized video decoding engines and Ethernet MAC engines. Even if the CPU row horse power is low, this would be able to play YouTube videos in such situations. On the other hand, if the embedded system does not have specialized video decoding and Ethernet MAC engines, a higher CPU MIPS will not sufficient, and the high-level functional requirements will not be fulfilled. Some of the criteria listed in Table 1 are incompatible with others. Higher processing capacity, for example, is directly related to electricity consumption. CPUs with high computational power use more energy than CPUs with lower computational power (Pentium4 is around 60W vs Atom D510 is around 13 W)[4]. Point 8 (Software compatibility / Porting effort) in table 1 clashes with 4,5. Point 8 is readily satisfied by x86 architecture based CPUs (usually desktops), however points 4,5 are not. 1&8 are satisfied by an Atom D510 system, however some of the other criteria are not.

1.4 Evolution of embedded system from past to the present:

Even if there are an unlimited number of embedded systems, the principles of functioning of system components and design methods are basically the same in all of them.

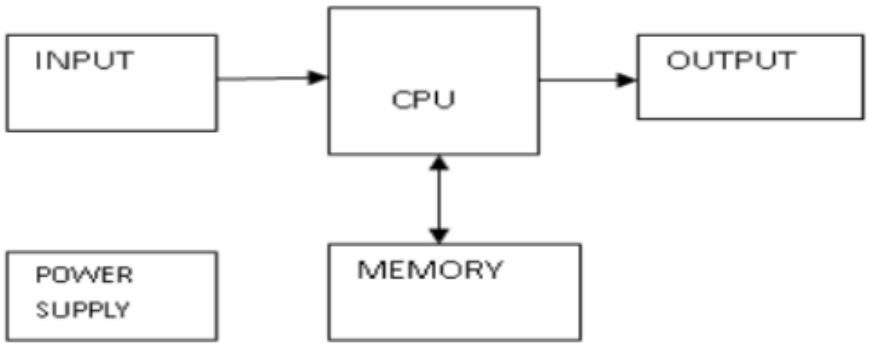


Figure 1: Representation of Generic Block Diagram of Embedded System.

Figure 1 represents generic block diagram of embedded system. Every embedded system, for example, includes a CPU and software that runs on the processor [2]. A microcontroller or a microprocessor may be used as the processor. Additionally, in order to have software, there must be a location to store the executable code as well as temporary storage for run-time data

manipulations, which will be in the form of ROM and RAM, respectively. Small memory may be housed on the same chip as the CPU in certain cases. Such configurations are common in microcontrollers. If this is not the case, one or both kinds of memory will be stored on external memory chips. Additionally, all embedded systems have some kind of inputs and outputs. Sensors and probes, communication signals, and control knobs and buttons are common inputs to the system. Displays, communication signals, and changes to the physical environment are all examples of outputs[5].

1.4.1. Evolution:

Embedded systems are made up of a CPU (which may be a microprocessor, microcontroller, DSP, or a mix of these), memory, and input/output subsystems. Every component of the embedded system has undergone a massive update. The Apollo Guidance Computer, developed by Charles Stark Draper at the MIT Instrumentation Laboratory in 1966, was one of the earliest widely acknowledged embedded systems. The Intel 4004 microprocessor was the first to be utilized in calculators. There has been a lot of change since then[6].

1.5 Design challenges:

Higher processing capacity is almost always inversely related to increased energy usage. High-performance computational CPUs use more electricity than low-performance computational CPUs. The Pentium 4 consumes approximately 60 watts and provides 9700 MIPS. The Atom D510 consumes approximately 13 W and provides 8400 MIPS, whereas the ARM Cortex A8 consumes less than a watt and provides 2000 MIPS. Desktop computers aren't built to withstand the elements mechanically or electrically. It has moving components in the hard drive and upgradeable RAM slots, among other things. Such components and designs are unacceptable in an embedded system. RAM slots are given in desktop computers in order to enable future memory upgrades, however this cannot be done in an embedded system since placing the RAM in slots is not a mechanically robust configuration and therefore lowers the mechanical strength of the system. Since a result, the memory needs of an embedded system must be determined ahead of time, as field upgrades are not feasible. There is almost no realistic system that meets all of the criteria in their entirety. As a result, it is critical that the parameters/requirements be properly weighted. The importance of each metric is determined by the application and product/project contexts. Power consumption may not be as important as cost or program compatibility in certain situations. As a result, no set weighting factors exist, and they are context-dependent. The difficulty is in correctly balancing the factors and making engineering compromises as a result[7].

1.6 Dependency on eco system:

Building a sophisticated embedded system, especially for a rural application, is a difficult job in terms of both hardware and software architecture. The main design challenge here is to create an incredibly powerful system (in terms of functionality offered by the system to make it easy to use for rural people) with very low power consumption (less than 10 watts), minimal NRE cost, and while fulfilling all of the criteria stated in Table 1. In reality, in every embedded system, both the hardware and software must be co-designed. The main variables that influence the SoC selection are the total system cost, power consumption, performance, size, component availability, adaptability, software tools, and time-to-market, among others. Eco-system support, as well as

SoC vendor support for both hardware and software components, are also essential for building a highly efficient system and play a critical part in the system design and development life cycle. One of the most essential elements in designing a successful rural application platform that meets the difficult and, at times, contradictory criteria is environmental support. The following sections address the eco system's contribution to the design of an embedded system for a rural application platform[8]:

- *Dependency on eco system for Hardware design aspects:*

Various embedded system components, such as the processor or microcontroller, memory (ROM and RAM), and hardware and software interfaces (or environment peripherals), are key aspects that influence the overall design approach. In the embedded sector, the current tendency is for processors to evolve toward fully featured system on chips (SoC) that may be customized for a particular application situation. These SoCs are tailored to particular needs in order to create complicated systems with the necessary interface components. The design of an embedded system based on these SoCs requires extensive assistance from the eco system, most notably from the SoC vendor. The peripherals that are compatible with these SoCs are critical to the design's success. The vendor must supply all hardware interface information, routing instructions for high-speed important signals, packaging specifications, and assembly directions in sufficient detail. Because modern SoCs have such a high degree of integration, these principles are critical in ensuring the design of a fault-free system. Additionally, the PCB design tools needed to support SoCs are critical and play a key part in the entire hardware design cycle for an embedded system. It is necessary to confirm the availability of hardware interfaces that are particularly needed for an application platform[9].

- *Dependency on eco system for OS/Device Drivers/ BSP:*

Embedded system components affect and define how embedded software or firmware is planned and developed, regardless of whether the system is CPU or SOC based [6-8]. One essential feature of the Kernel in an embedded system environment is that it handles the board's startup and setup, memory and I/O resource management, and required drivers for both on-board hardware devices and external peripherals. For the whole embedded system, the Root File-system supplies all of the necessary run-time user-space as well as kernel-space libraries, system binaries, start-up scripts (including DDR initialization), and feature-specific configuration files. Most silicon manufacturers offer board support packages (BSPs) for their SoCs, which include boot-loaders, kernels, and rootfs. In an embedded system, these BSPs must be modified and ported in accordance with their unique hardware design in order to satisfy their specific needs. The Linux kernel, an open source operating system, supports the majority of processor architectures (x86, PowerPC, MIPS, ARM, SPARC, SuperH, and others) used in the industry for a variety of applications, with the ARM architecture having a large presence in the embedded sector. For improved system performance while using less power, Linux kernel sources provide architecture-independent, highly optimized device drivers, resource, and power management modules. Reusing the aforementioned software components is very helpful for building an optimized system since Linux offers eco system support. Only the architectural and board-specific codes must be changed to meet their design requirements. Proprietary drivers must be incorporated into the firmware; proprietary vendor support is essential and, in most instances, supplied by the vendor[10].

1.7 Approaches towards meeting the design challenges:

There are many obstacles to overcome while designing a rural application platform. There are a variety of ways to addressing the issues, however most of the time an engineering compromise is needed. The computer module (microprocessor/SoC) must be carefully chosen. By architectural design, RISC-based SoC/microprocessors are power efficient. The architecture choices are ARM and PowerPC, with ARM being the most popular in terms of penetration and eco system compatibility. The availability of necessary built-in interfaces also influences the computer module selection. Display interfaces are required for the rural application platform. As a result, SoCs with built-in display interfaces (VGA/HDMI) are an obvious option. The processing capacity of the microprocessor/SoC, on the other hand, must be adequately validated. The processing power of the row CPU in terms of MIPS/MFLOPS is not necessarily a reliable indication of performance and does not guarantee that the performance criteria will be met. The built-in hardware modules (for example, video decoders) have a significant influence in the overall performance. As a result, while choosing a microprocessor/SoC, the presence of built-in hardware engines/co-processing modules must be checked against the necessary capabilities.

It's also crucial to make sure that the necessary driver support for such co-processing modules is accessible and simple to incorporate. The memory section's design is similarly important. Because speedier RAMs (SRAM) are more expensive and power demanding, their use in designs must be restricted. For optimal performance and power consumption, low power DDR is an excellent option. The design of the power section has a significant impact on the system's total power efficiency. The use of more efficient regulators increases the system's overall power efficiency. Systems that allow sleep modes/power saving modes provide benefits in terms of power usage, however system reaction times must be checked and guaranteed to be within the necessary requirements in such designs. In an embedded system, there are also design difficulties with operating systems. In terms of the capabilities it offers and the architectures it supports, the Linux kernel has progressed considerably.

Despite the fact that Linux is monolithic in terms of architecture, all drivers may be dynamically loaded. The program sources are portable between platforms thanks to its system call interface (kernel-to-user space API). As a result, in order to use the linux kernel in any embedded system design, the kernel must be heavily customized in terms of the functionality it offers in order to eliminate resources for superfluous processes that may slow down the system. Kernel must be trimmed/customized by adjusting the whole Software interface and its modules both within and outside the SoC while keeping the HW design in mind. The necessary driver or resource management modules inside the kernel must be replaced with proprietary modules or open-source modules must be adapted for the target SoC and capabilities, according to the HW design requirements.

All modules, such as interrupt handlers, GPIO, and board-specific initialization sources for busses and peripheral interfaces like USB, I2C, SPI, and device drivers, must be modified, which presents a major barrier in effective driver and low-level software porting. The efficiency of these modules has a significant impact on the overall performance of the embedded system. There are architecture-dependent memory management and CPU-related codes; these are critical components that determine the system's functionality and performance. User-space programs and even libraries must be modified and developed for the necessary target in accordance with the

HW environment. To minimize communication overhead, suitable protocols and networks/bus-types must be carefully selected, as stated in. This allows the system to meet speed requirements while simultaneously limiting power consumption.

2. DISCUSSION

A programmed controlling and operating system with a specific purpose inside a larger mechanical or electrical system, typically with real-time processing limitations, is known as an embedded system. It's typically found as part of a larger gadget that includes physical and mechanical components. Many modern gadgets are controlled by embedded systems. Ninety-eight percent of all microprocessors are made as embedded system components. Cheap power consumption, compact size, robust working ranges, and low per-unit cost are some of the characteristics of common embedded computers as compared to general-purpose equivalents. This comes at the cost of restricted processing resources, which makes programming and interacting with them much more challenging. By building intelligence mechanisms on top of the hardware, taking advantage of possible existing sensors and the existence of a network of embedded units, one can both optimize available resources at the unit and network levels, as well as provide augmented functions far beyond what is currently available.

Intelligent methods may be used to control the power consumption of embedded systems, for example. Microcontrollers (i.e. CPUs with integrated memory or peripheral interfaces) are popular in modern embedded systems, although conventional microprocessors (using separate chips for memory and peripheral interface circuits) are also widespread, particularly in more sophisticated systems. The processor(s) utilized in each instance may be general-purpose, specialized in a certain class of calculations, or even custom-designed for the application at hand. The digital signal processor is a popular kind of specialized processor (DSP).Because the embedded system is devoted to a single job, design engineers may optimize it to decrease the product's size and cost while increasing its dependability and performance. Embedded systems are sometimes mass-produced to take advantage of economies of scale.Embedded systems include anything from small portable gadgets like digital watches and MP3 players to huge permanent installations like traffic lights and industrial controls, as well as more sophisticated systems like hybrid cars, MRI, and avionics.

3. CONCLUSION

Embedded system design presents a number of difficulties, particularly for rural application platforms. We addressed the criteria that the rural application platform must satisfy, the difficulties that must be overcome in order to meet those needs, and future trends in embedded system design in this article. According to our findings, needs are inherently contradictory, and it is thus preferable to represent them at a higher degree of functional abstraction rather than at a highly granular level. This will be helpful in the design of embedded systems as well as in narrowing down optimization methods and analysis in the context of rural application platforms. In the future, we want to use statistical techniques and feature selection methods such as Principal Component Analysis to conduct quantitative analysis of optimization approaches for embedded systems.

REFERENCES:

1. M. Rashid, M. W. Anwar, and A. M. Khan, "Toward the tools selection in model based system engineering for embedded systems - A systematic literature review," *J. Syst. Softw.*, 2015, doi: 10.1016/j.jss.2015.04.089.

2. M. Seo and R. Lysecky, "Non-intrusive in-situ requirements monitoring of embedded system," *ACM Trans. Des. Autom. Electron. Syst.*, 2018, doi: 10.1145/3206213.

3. G. Martin, L. Lavagno, H. Hansson, M. Nolin, T. Nolte, and K. Thramboulidis, "Embedded systems," in *Systems, Controls, Embedded Systems, Energy, and Machines*, 2017.

4. C. Manifavas, K. Fysarakis, A. Papanikolaou, and I. Papaefstathiou, "Embedded Systems Security: A Survey of EU Research Efforts," *Secur. Commun. Networks*, 2015, doi: 10.1002/sec.1151.

5. A. Malinowski and H. Yu, "Comparison of embedded system design for industrial applications," *IEEE Trans. Ind. Informatics*, 2011, doi: 10.1109/TII.2011.2124466.

6. H. Mao, S. Yao, T. Tang, B. Li, J. Yao, and Y. Wang, "Towards Real-Time Object Detection on Embedded Systems," *IEEE Trans. Emerg. Top. Comput.*, 2018, doi: 10.1109/TETC.2016.2593643.

7. S. Parameswaran and T. Wolf, "Embedded systems security-an overview," *Des. Autom. Embed. Syst.*, 2008, doi: 10.1007/s10617-008-9027-x.

8. J. A. Back, L. P. Tedesco, R. F. Molz, and E. O. B. Nara, "An embedded system approach for energy monitoring and analysis in industrial processes," *Energy*, 2016, doi: 10.1016/j.energy.2016.09.045.

9. P. Axer *et al.*, "Building timing predictable embedded systems," *ACM Trans. Embed. Comput. Syst.*, 2014, doi: 10.1145/2560033.

10. B. H. Sababha, Y. A. Alqudah, A. Abualbasal, and E. A. Q. Al, "Project-based learning to enhance teaching embedded systems," *Eurasia J. Math. Sci. Technol. Educ.*, 2016, doi: 10.12973/eurasia.2016.1267a.