Deep Science Publishing
https://doi.org/10.70593/978-93-49910-80-5_2

DeepScience
Open Access Books

Chapter 2: Fundamentals of embedded systems for the design of smart medical equipment

2.1. Introduction to Embedded Systems

Research on Embedded Systems has become increasingly common and important in recent years. The successful implementation and design of such systems have recently gained notoriety given the common integration of Embedded Processors in commercial Electronics. Although these were traditionally found only in specialized usage, the development of high performance programmable microcontrollers and the parallel advances in Sensor Technology have brought their use to areas such as Electronics for Robotics, Telecommunication and Medical Electronics. In these areas, Embedded Processors have shifted from having only a supportive function in dedicated circuitry to a more controlling role in complex innovative solutions.The Medical Electronics field has made a rapid and broad move to the architecture of Smart Medical Equipment. The move from traditional Electro-Medical Equipment has led to changes in design philosophy. Devices now possess high information content and rely on sophisticated Software Algorithms that often enable the visualization and storage of data (Ahmed et al., 2018; Jagadeeswari, 2018; Firouzi et al., 2022). The bundle of Electronics with other areas of knowledge in the design of a medical product may involve rehabilitation therapists, orthopedists, bio-mechanic engineers, dentists, physiologists, cardiologists, vascular surgeons, and neurologists, among other professionals. Interdisciplinary collaboration may lead to the successful completion of a medical product, or pieces of autonomous “high-tech” cure or diagnostic equipment can become available.An Embedded System is Computer Hardware and Software with a dedicated function within a larger mechanical or electrical system or Product. Embedded Systems have been essential in designing Smart Medical Equipment. Device smartness is expressed, for instance, by data Communication, User Interface, Storage, Display, Information Processing, and Diagnostic Capability. Typically, Embedded Systems contain

specialized hardware designed for devices that have pre-defined functions. Their Software enables the realization of auxiliary tasks inserted for coordination of the diverse system functionalities. With just a few exceptions, Smart Medical Devices are digital Electronics composed of Analog and Digital Processing Circuits connected to Microcontrollers or Microprocessors (Mathew, 2018; Liu et al., 2019).

2.1.1. Definition and Importance

From the very beginning, the design of embedded systems has accompanied man on his journey of technological evolution. The need to carry on monotonous, or dangerous tasks at a distance, led to the creation of machines that, by mimicking the cognition and behavior of a human being, could fill those gaps.

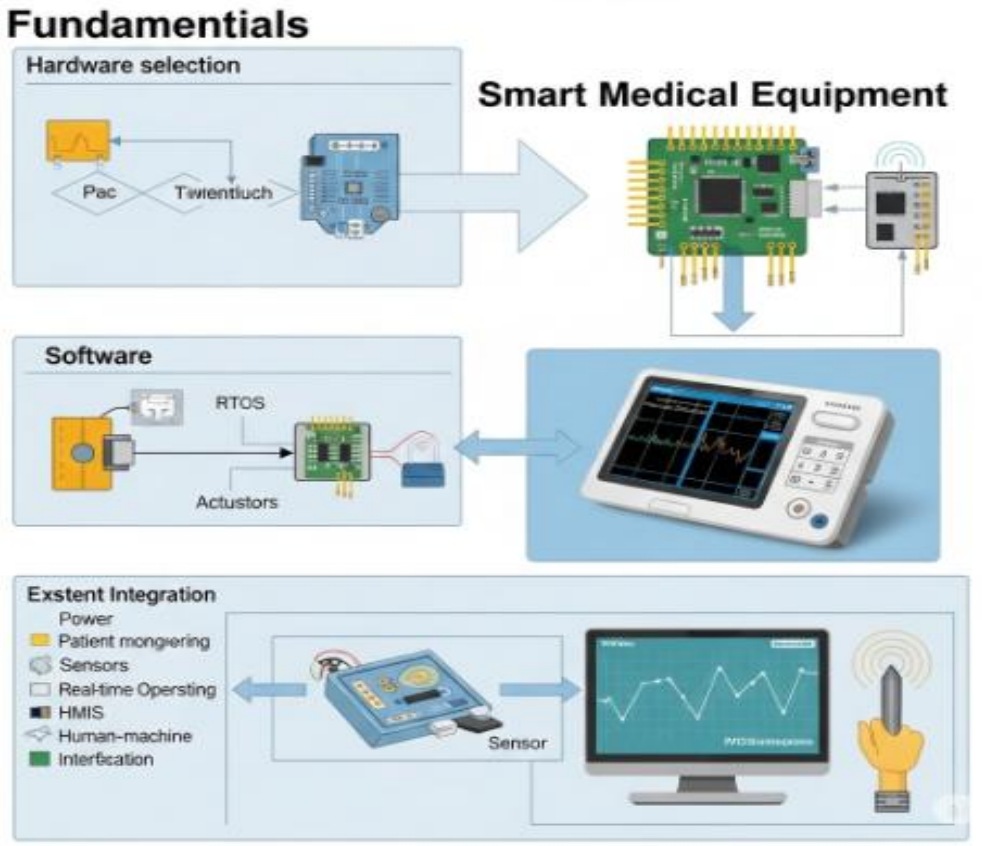

Fig 2.1: Fundamentals of Embedded Systems for the Design of Smart Medical Equipment.

These machines were created to work for a long time, with incredible precision, were able to withstand adverse environmental variations, and were often used outside the

facilities in companies or laboratories. In this sense, machines had a specific function to perform, with cycles that required to be repeated with high degree of regularity, following a programmed sequence. However, the passage of time, with an ever increasing demand for complexity and variability of the tasks to be performed, led to the need to broaden the range of capabilities of the machines used in preemption of human work. This led to the evolution from the original industrial automation, achieved through the use of fixed or programmable machines performing specific operations, to the formation of flexible or smart production systems or cells. Such systems and cells are able to combine the advantages of industrial automation with the need for diversity, variability and intelligence in the product performed.

The term 'embedded system' is commonly used to indicate special purpose computers designed to carry out specific tasks. In a more precise definition, it can be considered that an embedded system is a computer-based system designed to perform a dedicated function or functions within a larger mechanical or electrical system. According to this definition, embedded systems have, typically, the following three characteristics: - They are not programmable by the user, meaning that the internal programming is done at the time of manufacturing, - They perform only one dedicated function, with possible variations in function due to variations in input data; - They are not stand-alone systems, in that they must interact with other physical systems from which they normally receive their input information.

2.1.2. Historical Context

The term Embedded Systems used in the context of the title specifically applies to computerized control systems that are part of a larger system, such as a medical device. When this text refers to medical devices it could refer to diagnostic, monitoring, or therapeutic devices used in treatment of medical conditions. Computerized control systems have been in existence for a long time. Simple analog control systems were present in the 18th century when a steam engine that used a flyball governor to control the engine speed was developed.

The word computer however has a more specific meaning and is associated with a device that is able to perform arithmetic/logarithmic computations very quickly and can handle large amounts of data. The first computers were discussed by a pioneer who proposed an early computing engine. While this engine was not actually built, the first working computers were developed and rebuilt during the Second World War by various individuals. Medical devices were probably the earliest application of computing systems. Early computers were used to develop atomic bomb simulations during World War II, and radiation treatment plans were developed for patients that received radiation treatment for nose cancer. The first computers were large, heavy, expensive, and used a

lot of power. In spite of these problems, the speed and accuracy with which computers were able to perform tasks meant that when they were used for various applications, the cost of the systems was actually far less than when these tasks were performed manually.

2.2. Core Components of Embedded Systems

For designing dedicated systems something that is commonly used is Embedded Systems. Embedded systems deliver, very often, the best performance at the lowest cost, size and power consumption thanks to the high level of integration achieved with the integration of a computer with other digital or engine components like sensors and actuators. Embedded systems are a highly-integrated mix of electronic and firmware components including hardware, software, middleware, and operating systems that are developed to work in electronically-controlled devices. Embedded system applications often require to deal precisely with events happening outside the physical implementation of the systems and, very often, inside the device, at a very high speed and at a very low cost. Although highly flexible embedded systems offer best design tradeoffs for many consumer, industrial, and medical applications, they are not always the best solution possible from an overall performance perspective.

Microcontrollers

Traditionally, Embedded Systems were built with classical and stand-alone sensors interfaced to classical computer circuits. Their intelligence comes only from the processing elements available in the computer. The development of Embedded Systems has accelerated with the invention of microcontrollers that are basically integrated circuits that provide a digital computer and one or more appropriate sensors/actuators interfacing circuitry on one silicon chip. Microcontrollers have all the features of a computer, such as CPU, memory, and input/output capability and include processing elements targeted at the specific applications, such as read-only memories for storage of microprograms unique to the individual microcontrollers.

Sensors and Actuators

Sensors are transducers that convert physical signals of interest into electrical signals. They have become very powerful components of embedded systems, providing the systems with awareness of the external environment. They can measure several natural phenomena such as temperature, pressure, motion, and velocity. Actuators sense the existence and characteristics of information in other systems and use the information to control the other systems. The existing actuators include electromagnetic, thermal, light, liquid pressure, and piezoelectric actuators.

Power Management

Power management sections have started appearing in embedded systems due to a combination of applications and design methodologies. Five trends that influence design methodologies are the establishment of more diverse cores inside SoCs, the introduction of new packages with more cores and advanced thermal management techniques, support for more diverse application presenters to help different application performances, involvement of more process technologies, and the high cost in power for applications in badge technology.

2.2.1. Microcontrollers

Microcontrollers have become increasingly important in today's technological world. They are the core embedded systems responsible for the collection of data from the external environment, processing it for further actions, and finally commanding external devices to take actions. Microcontrollers are utilized in automatic feedback systems that have helped in taking over boring works. They have advantages over chips or FPGAs, being mid-level integrated circuits. They provide more on-chip integration of functions, which can replace many chips, while allowing non-reconfigurable software programming as compared to chips.

Microcontrollers consist of many elements, such as processors, memories, I/O peripherals, etc. A microcontroller consists of embedded CPU and on-chip memory and peripherals. In simpler words, microcontrollers can be defined as a single-chip computer, with a CPU, memory, and I/O peripheral devices integrated within it. Microcontrollers are continuously evolving to become chip solutions. Present-day microcontrollers are integrated solutions for specific market requirements. They help in reducing overall system costs and offer real-time performance by overcoming the bottlenecks of general-purpose microcontroller-based systems. Although additional functions like Bluetooth interface and Wi-Fi are included in the present-day microcontrollers, most of them still require custom hardware for various power management requirements. Low-power applications still require important solutions for logic voltage levels, which are not provided by general-purpose microcontrollers, but fabricated as specific solutions for a particular design.

Standard workhorse microcontrollers do not provide for low power requirements. They have become increasingly important in today’s advanced semiconductor technology with the capability of maintaining higher speed while dealing with very low voltages. They have become competitive not only in terms of their price but also in performance with special purpose standard microcontrollers.

2.2.2. Sensors and Actuators

Sensors play a crucial role in the interaction between embedded systems and the physical world. They are critical components that allow embedded systems to obtain data from the environment; in sensor networks, sensor nodes are capable of capturing and processing data from the environment and transmitting it to remote servers for further analysis or data fusion. A typical smart medical system incorporates multiple sensors to monitor different variables, including physiological sensors to monitor vital signs, pressure sensors to detect ventilation and assessment of sleep disorders, temperature sensors to monitor pathological variations, gas sensors for the detection and diagnosis of diseases, glucose sensors to control diabetes mellitus, and acceleration sensors for location-based diagnosis. For these reasons, sensors will be introduced in detail in this section. Actuators, which are parts of embedded systems capable of translating commands into physical actions, are essential components of embedded systems that interact with the world. The output processed by the embedded system generates reactions in the physical world through actuators. The most common external actions generated by actuators can be the infusion of prescribed drugs, the electric stimulation of biological tissues, and the generation of mechanical movements to react to physical events. The types of motors can vary according to the need, and might be DC motors, step-controlled motors, linear motors, or solenoids. These motors are usually paired with relays and brake drivers. In the case of intricate or precise movements, hydraulic or pneumatic actuators can be used. For some specific applications, the combination of heaters, LED lights, and loudspeakers can also be used to stimulate a reaction from the physical world.

2.2.3. Power Management

To cope with their limited energy storage capacity, smart medical equipment running on embedded systems require power management. Possible design flows to achieve this objective are presented in this section, ranging from hardware components to operating system techniques, combined or not.

There are basically three design options. The first one is to spend money and time at the design cycle to make the hardware as efficient as possible. This strategy is commonly required when the device is expected to reach its end of life without maintenance. The second option is to use specially developed components that allow us to harvest energy from the environment. This option is still in the early stages of development but there are expectations of an interesting progress. Energy harvesting can create a dedicated power supply for the device and allows its batteries to be slowly charged, prolonging the natural battery life. The third strategy is to spend a considerable time during the embedded operating system design and choice, addressing power management features

in a flexible and efficient way. The more common power saving strategies for hardware are reducing voltage levels and clock frequencies. At the OS level, the major options are transferring to a sleep state, suspending or shutting-off specific components, changing the execution frequency of an incoming signal processing algorithm, and defining dynamic rules to take these actions. Hardware features can be mixed and let the OS make use of them according to intended objectives.

Power management has to be accounted for in the whole system design, not only in the embedded OS. Avoiding or deliberately inserting specific operations, especially during the system boot process, can save a considerable amount of energy when not accounted for. Special functions at a determined frequency may demand to increase the component working frequencies and so the costs.

2.3. Software Development for Embedded Systems

Embedded systems comprise not only hardware but also firmware to yield a complete functional system. In many development cycles, only one hardware prototype is manufactured. Once the system specification is outlined and hardware is designed, engineers create firmware that is tested and verified on the single hardware prototype. After testing, the production system will use the primary hardware, but the firmware will change whenever software upgrades are necessary. Such programming is usually created in high-level languages. However, in cases where high performance and stringent resource utilization is a requirement, firmware is coded in assembly language. Even in high-level languages such as C or C++, chips that are not too complicated are programmed using assembly. This is because C is originally not a system programming language and is neutral to hardware configurations. In the case of C++, as a result of its features of objects, dynamic binding, and virtual functions, C++ has not been fully accepted as a programming language for embedded systems.

Nevertheless, the trend continues. Embedded systems manufacture is a process dominated by small companies and firms. But large companies have also begun to participate in the embedded market. These corporations offer a complete product line that covers the major domains of embedded systems. In addition, many offers are expansion kits that allow developers to create their own product using the complete package. The development software includes library functions that control the hardwired circuit. The functions facilitate easy and rapid implementation. Many embedded systems found in smart medical equipment, especially in low-cost and popular devices, are also developed with the help of expansion kits and supporting software.

2.3.1. Programming Languages

Introduction Various programming languages are available to build software for embedded systems. These languages are categorized based on their suitability for specific applications and audiences. Choosing an optimal language is crucial for the software project's efficiency, maintainability, and performance, and understanding the language's properties is a prerequisite for making the decision. Factors such as execution speed, memory footprint, portability, performance monitoring, and cost affect the choice of language. The development team also influences the development time, flexibility, and maintainability of the final product. Assembly Languages Mandated by resource limitations, assembly languages remain the optimal choice for embedded system application. The Achilles' heel of assembly languages lies in the delicate design flow and lack of testing and verification capabilities. Therefore, reliability-sensitive embedded applications cannot be developed with assembly languages. Nevertheless, the unique advantages of assembly languages lead to the development of embedded applications requiring fine-tuning on the engineering floor. For lifetime- and safety-critical systems, these applications reach production level and are validated with dedicated testing tools. Notably, assembly boards can be marketed. These boards are validated by embedded product software developers who write application programs in C, VHDL, Verilog, or PLM, in conjunction with analog and digital designers managing the hardware boards. However, industry experts warn that utilization of assembly language should be confined to commercial boards with no close competitors and well-defined market niches, such as application specific integrated circuits or complex programmable logic devices. In other cases, the investment in timing and debugging tools for board fabrication does not return any dividend. C and C++ C and C++ have multiple advantages including wide availability on multiple embedded platforms, ease of interfacing with hardware, pre-defined external libraries, and relative ease of debugging. They are therefore the primary languages for embedded development, and most commercial boards come with development tools to read and write in C or C++. With the current prevalence of field-programmable gate arrays on embedded boards, languages for hardware description come into play in coprocessor design. Hardware implementation refines and speeds up core functionalities of the embedded application and transfers the other functions to the possible software layers.

2.3.2. Development Environments

In principle, development environments are tools that help implement systems using only the programming and assembly languages available. However, the major advantage of development environments is that they help a lot in the most complex phase of an embedded software project, when no code at all has been written, using a concept in

software engineering: the concept of a software model. The idea is that drawing in a graphical way pieces of software will generate most of the code of the software project, including the code of the part of the project that is the most complex to implement and that has no code at all yet.

Development environments can be divided into five categories according to the support they provide: simple code writer tools for specialized embedded software; Integrated Development Environments; Graphic Interface Design Environments; Software Model Development Environments; Hardware Description Language Development Environments. Simple code writer tools for specialized embedded software are code editors for specific software implementation tools, such as cross assemblers, linkers, and loaders; but, those tools may have the advantage of being mini compilers based upon the model presented below as an example so as to allow us to visualize their simplified implementation.

Integrated Development Environments are a code editor for specific software implementation tools, such as cross compilers, assemblers, linkers, loaders, and debuggers, all together to facilitate the embedded software development, debugging, and maintenance phases. Graphic Interface Design Environments are IDEs for developing only the part of the code related to the Graphic User Interface. Software Model Development Environments implement on embedded systems the software model concept in software engineering, that is, sections of the embedded software can have their code generated by the tool just by drawing structures in a graphical way, as if they were flowcharts, making use of a software block library that is specific for the embedded software in development.

2.3.3. Debugging Techniques

A number of different methodologies are employed for effectively isolating logic errors in program code when debugging an embedded CPU and a user-developed application program. The methods fall into two general categories, hardware and software. The hardware-oriented methods rely to a great degree on the use of special tools, while the software-oriented methods depend mainly on the application program code. Of course, combinations of these different techniques are most often used in practice. Consequently, each programmer develops their own set of preferred debugging methods. Typically, users begin with trial-and-error techniques and then move onto more effective methods.

Hardware-oriented techniques are so called because they involve interacting with the CPU hardware while the application program is being executed. In general, the more sophisticated the external debugging tool, the less intrusive it is to the actual CPU

operation. Simple hardware-oriented debugging methods involve the use of displays to report variable values or program execution state. A series of predefined state or status values are usually stored in memory, as opposed to real time computation as a method of reducing timing overhead.

A more sophisticated hardware debugging technique involves the use of external debug timers which use hardware interrupts to automatically monitor the operation of the CPU while the program is running. These tools are useful for monitoring program execution state for all instruction cycles, or a specific set or range of instruction cycles. Timing analysis is essential for timing-critical tasks or when interacting with timing-sensitive peripherals. Simultaneous interaction with external peripheral devices via I/O lines while the program is being executed enables useful debugging information to be developed.

2.4. Design Principles for Medical Equipment

At its heart, the design of medical equipment seeks to restore or maintain health and improve safety and quality of life to the patient. Although it seems that nothing should be simpler than to design an efficient piece of hardware like a container or a vacuum, this task is made significantly more complex as the throughput capability increases to handle more patients, for example during surgery. Additionally, the need for sterilization hampers the use of commonly used materials like plastics, stainless steel or aluminum. The design also needs to take into consideration that a possible failure of any equipment in the OR may result in drastic consequences, for example by stressing a patient's cardiovascular system at an inappropriate time during surgery. Thanks to previous experiences both in the OR and in an environment closely related to medicine, for example aerospace, the following design principles have been defined and have been successfully integrated in medical devices.

User-Centric Design

Health professionals and surgeons strive to improve medical service quality and patient safety. As experts in the field, they offer high potential input into the equipment design and thus mold user-centric devices. However, often their busy hospital schedules do not allow time-consuming interactions with developers during long design and testing cycles. We propose to shorten these cycles, especially in the early phases of development, by assisting product designers with high-level user feedback early and often. Using a combination of sketching, physical mockups, and low-cost technology, fledgling products can gain direction quickly and implement iteratively in short cycles to yield devices that will see use and ultimately benefit both the healthcare team and the patients.

Safety and Compliance

The overwhelming need to prevent injuries to patients, care-givers and medical personnel ultimately comes from painful past experiences. Patients' misfortune by undergoing an unnecessary second surgery, parents grieving a loss of their child by a medical error, and the public outrage caught by journalists when the health system collapses are some incentives for medical equipment manufacturers to abide by the safety standard which governs the design of practically all medical devices used.

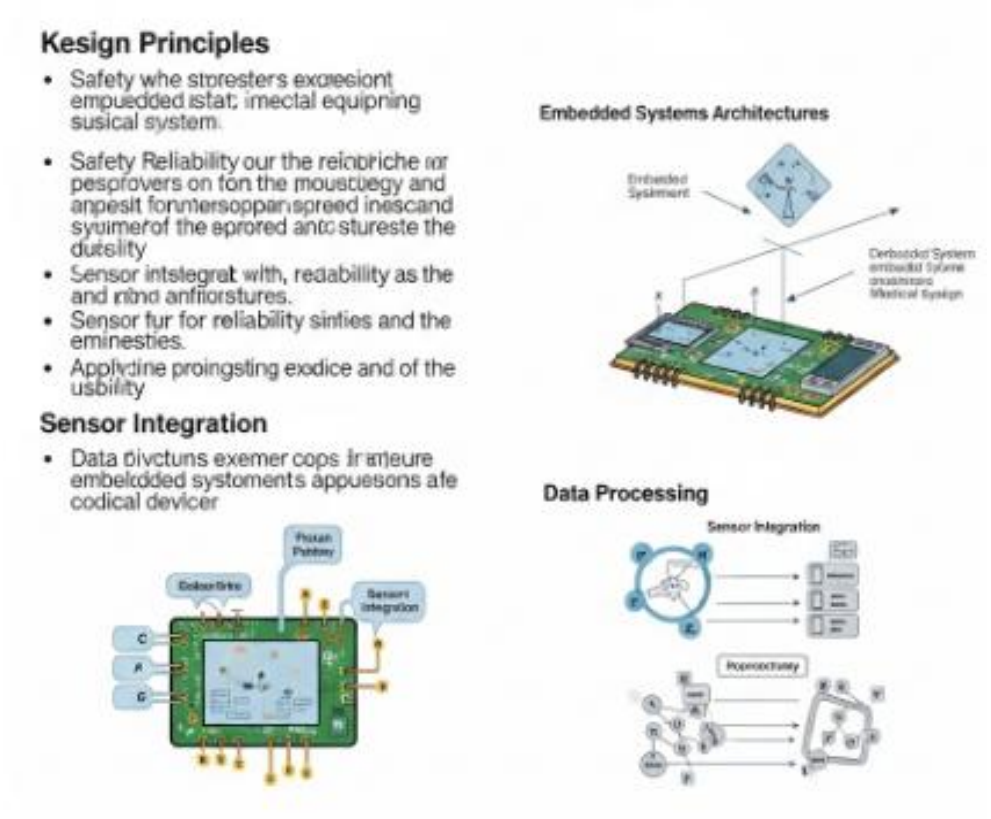

Fig 2.2 : Design Principles for Medical Equipment.

2.4.1. User-Centric Design

User-centric (or user-centered) design (UCD) is a design philosophy that aims to improve user experience (UX) and increase the usability of machines for the intended end user. This is accomplished by paying close attention to end-user needs and preferences during the design process, allowing them to evaluate prototypes, and making iterative changes to arrive at a final design. In particular, for medical equipment that has a direct effect on the well-being of patients and is operated by medical practitioners, a

UCD approach is vital to the successful design and deployment of the technology. User-centric design has many positive effects, such as increasing user happiness, leading to fewer user errors, and decreasing learning time; for medical equipment, this results in increased practitioner efficiency, reduced operating costs, and improved patient safety.

The concept of user-centric design has parallels with human factors engineering (HFE), user experience design, usability engineering, participatory design, and interaction design. However, UCD generally has a specific philosophy and process centered around the user and user feedback, while all the other concepts mentioned focus on specific aspects of design or specific ways of incorporating the user. UCD is also predicated on the understanding that design is an iterative process and not a linear path, as is normally emphasized in nominal engineering design processes. HFE and UX emphasize the psychological aspects of the user experience and are usually not part of the engineering design process. Because of its increased focus on the user, UCD makes use of extensive user input during the design process, typically with testing and feedback; many other concepts take that feedback and input as an assumption during the design process, using other validation or design methods. Therefore, it is important to use UCD at the very beginning of the design process, and work with it through subsequent revisions or iterations of that design.

2.4.2. Safety and Compliance

A medical device must meet a great plurality of requirements given the intentional medical purpose. Although some devices provide only diagnostic information or simply collect data, many will provide a diagnosis, or more importantly, lead to a medical action that will influence a person's health and degree of wellness. As a consequence, the role of a medical device within the ecosystem of care delivery is central. This will impose a huge number of constraints that a medical device must respond to. Most medical devices work together in a coordinated manner toward an overall purpose of enhancing the quality of the care delivered, while seeking to avoid iatrogenic consequences, and constantly aiming for the prevention of disease, delivery of treatment, and even cure. This involves another contradictory design principle – the company's commercial initiative linked to Engineering for the Delivery of Medical Solutions must be harmonized with economics.

The unintended consequences of a medical device being unsafe or effecting an unfavorable iatrogenic consequence can be severe. This was amplified in recent years by the large use of software-based approaches, increasing the complexity of embedded systems in medical devices. As a consequence, the requirements driven by Safety Assurance and Compliance become some of the most critical for the Design outfit of the embedded system, and also for the validation of its behavior and performance prediction.

This becomes even more challenging when we take into consideration that Safety, highly abstract and unintuitive cannot be designated into concrete architectural rules, but instead must be demonstrated at every level of the Safety Assurance and Validation process.

2.5. Communication Protocols in Medical Devices

Communication between devices, equipment, and systems can be both wired and wireless. Typically, equipment that is not portable or is in a controlled environment is connected with cables that can also provide power to the equipment. However, equipment that is portable or within free movement or the patient is connected through cableless protocols. The cables that are used in wired protocols should be insulated and covered to avoid exposure and damage. The same risk is not present in wireless protocols. However, careful consideration should be provided to security and privacy policies when implementing readily available wireless communication protocols in embedded systems of medical equipment.

Wired Protocols

General equipment interfaces implemented by wired communication protocols include Universal Serial Bus and RS-232 protocols. USB is used mainly for connecting portable data storage devices, but many data transferable medical devices connect using USB. For example, USB is used in digital microscopes. Serial Communication Protocol is present in a wide range of medical devices, especially less advanced diagnostic and therapeutic devices. Other wired interfaces include FireWire, Serial Peripheral Interface, Inter-Integrated Circuit, and Ethernet. With ever-advancing technology in low-power models, Ethernet connections have been implemented in portable medical devices too. Ethernet is the most common LAN and is implemented in computer networks in hospitals.

Wireless Protocols

There are many wireless communication protocols available, including Wi-Fi Direct, Bluetooth, Near Field Communication, Zigbee, Z-Wave, Insteon, WI-Sun, and ANT and RFID. There are quite a few health-related applications using Bluetooth and Zigbee embedded in mobile phones, tablets, and computers for health care. NFC is gaining popularity in the area of medical devices as it is a short-range communication system with an automatic connection approach.

2.5.1. Wired Protocols

This chapter introduces the main communication protocols used in medical systems, focusing on the wired industrial protocols applied in smart medical devices. The communication protocols presented are addressed in their usage in medical systems, the abstract model of execution and data flow, their basic frame structure, and the data types supported. After the protocol description, pertinent considerations are included to address relevant aspects such as energy efficiency, security, fault tolerance, and safety.

Although medical systems may make use of common wired and wireless communication protocols, such as RS-232, RS-485, Ethernet, IP, MQTT, WebSocket, and HTTP, the smart medical devices must use specific industrial protocols. In this chapter, the data link protocols are discussed. These specific protocols have been developed based on strict requirements, defined in standards. The industrial protocols guarantee deterministic execution and data transfer on time, security against external attacks, fault tolerance through redundancy of commands and data, safety by verifying the integrity of data communication, support for different data types, low processing overhead with possible use in low-cost smart medical devices, and low power consumption using sleep mode, allowing the device to operate for several years with a small battery. Due to the mentioned aspects, smart devices used in medical systems must use industrial wired protocols. Wireless protocols do not exhibit one or more of these properties, making them incapable of being safely used in medical applications.

2.5.2. Wireless Protocols

Wireless communications offer physical advantages in such applications as the data collection from capsules, sensors in the body, and distant non-immersible sensors. The proprietary protocols in use often incur difficulties in connecting to received nodes, and their range and data rate are inferior to the standards. Their use was justified due to their insurance of safety, low power (for the periodic waking up of nodes), and a small number of associated nodes. Most of the proposed wireless infrastructure use the standards: Bluetooth, Zigbee, and Wireless Fidelity. Their quality and cost of a transceiver is small enough to be used in Body Area Networks, and moved to a similar low-power and low data rate on-the-radio frequency for Wireless Personal Area Networks standard. The standard, adopted in 2003, is the bottom MAC level of the higher standards with off-the-shelf transceivers at low power and cost. It is an order of an elongation in the type of both the order frames and beacons for three levels of the priority traffic and a 2 ms wake up in each 100 ms are possibly issued. The protocol is an adaptation to infrastructures as Networks Medical Association and Open Systems Foundation Communication Protocol. ZigBee is a multilevel standard based on and proposes several security mechanisms when coexisting with standards to monitor and secure the other channels. Medical

implants usually have the following restrictions: low area, long battery life (usually on the order of years), and periodic use (ms). They use ISM channels below 1 GHz: MICS, 402–405 MHz, near the RFID: 13.56 MHz, and 860–960 MHz frequency-modulated, at 2.4 GHz: ZigBee, and at 5.8 GHz.

2.5.3. Data Security and Privacy

With wireless and Internet-enabled medical devices rapidly expanding, data security and privacy have taken a serious step forward. These are no longer simply technical niceties; they are now part of the regulatory requirements for the sale of devices in many larger markets and are often demanded by clinicians in charge of patients' care.

The two widely known compromises against user privacy are interception and misuse of the information transmitted from a medical device, for instance, a heart rate monitor, and injection of misleading information into the end-users' data streams. The challenges faced by manufacturers of commercial off-the-shelf medical devices operating on commercial platforms differ from those in other fields. The affordability, direct engagement with end users, access to substantial aggregate data, competitive forces, and regulatory requirements often underlying quality solutions with sufficient scrutiny mitigate against security challenges.

There are four cybersecurity goals for the production and deployment of medical devices: confidentiality to ensure that sensitive patient information is not disclosed inappropriately; integrity to allow the authorized party to control medical device data so that it is not modified inappropriately; availability to guarantee that a medical device can perform its intended function reliably; and authenticity to protect against unauthorized access that would permit compromise of confidentiality, integrity, or availability.

A typical critique of a specific medical device is use of an insecure communication protocol in a device that uses or transmits patient-facing data that are both sensitive and potentially injurious to that patient, and that is critical to the health of the patient.

2.6. Real-Time Operating Systems (RTOS)

Real-time systems are those that respond to events within a time that is determined according to the behavior of the system, that is, the response time must be as low as required. In some cases, even if the system is not in real-time, its response should happen as quickly as possible, for example, when the user is waiting. The important thing is that these deadlines are defined according to the user's requirement and must always be satisfied, especially in critical systems. Critical systems are those in which the absence

of a response, an erroneous response, or an untimely response creates some damage to their users. A well-known example of critical users is the airbag system, which needs to respond instantly so that it does not cause further damage. A non-critical system example is a printer connected to a computer.

Embedded systems are present in most smart devices currently developed, as the name implies, either to perform some ancillary function, which may be critical or non-critical according to the user's criteria. Given the great variety of applications (critical and noncritical), the description of these systems is quite broad. However, the characterization of real-time embedded system presents some specific requirements, some combine several features, such as: These systems have strict temporal requirements; They have a continuous, repetitive, periodic operation; Most of them require a predictable behavior; In general, they require deadline management; They often have more critical tasks than non-critical tasks; Generally, they require low-cost, small processors and low-power consumption; These systems present difficulties in changing the data and programs in the field; and Most are dedicated and reusable, but there are some non-dedicated.

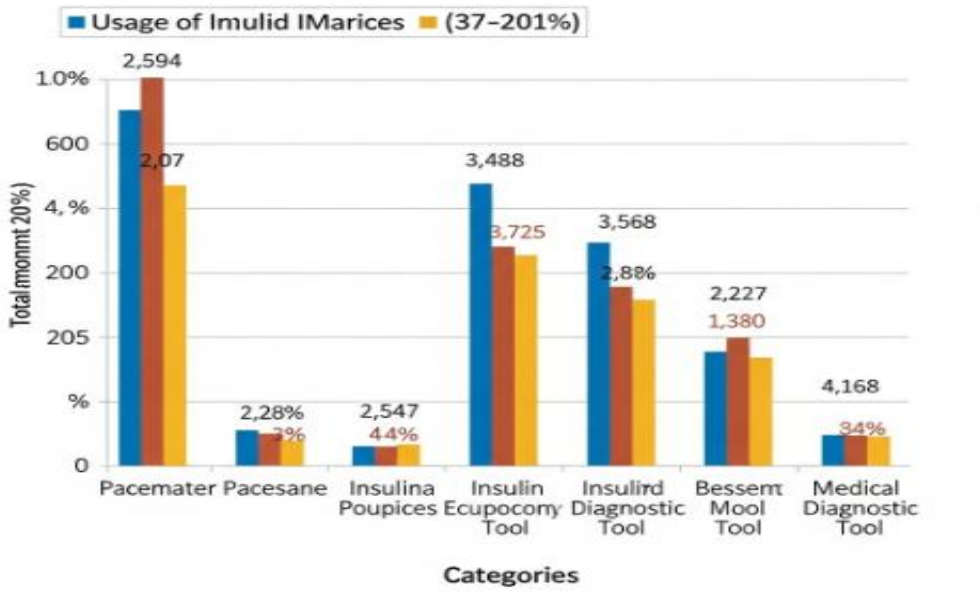

Fig : Medical Equipment of Fundamentals of Embedded Systems for the Design of Smart Medical Equipment.

There are embedded systems that do not require real-time characteristics, but given the answer time of these devices, we have a temporal response that we can consider as real time. An example is the electronic game console of endless games that can be produced and launched on the market. The need for a real-time response and the only reliability of

the system defines real-time systems. These deadlines have a physical form, such as the fact that the system must be effective for the display of a film. These characteristics presented above describe the strict real-time embedded system, which is the focus of this study.

2.6.1. RTOS Characteristics

Real-time operating systems (RTOS) are specialized software systems designed for embedded computing applications where the computing demands of a real-time process must be coordinated by the RTOS together with other embedded computing tasks. Unlike general-purpose operating systems (GPOS), where no deadline is imposed on the executing task of an application, an RTOS assures that for each periodic or aperiodic executable task its computations are completed within the specified worst-case response time. This is of extreme importance in critical embedded systems such as smart medical equipment, avionics control, financial banking, etc. Suppose the duty of a certain application task is to sample an analog sensor input every n seconds, filter the raw data, and publish it for use by other tasks. In that case, if the task takes much longer than n seconds to execute, the system will announce incorrect information, which is often fatal in smart medical applications.

By having control of scheduling of all the executing tasks according to specific timing regulations, RTOS can manage to meet the real-time deadlines of embedded applications. Therefore there are some particular technical features that distinguish RTOS from GPOS. The following are the more essential RTOS characteristics utilized in current-day embedded applications: (1) Guaranteed response time: for periodic tasks, response time can be predicted and specified. (2) Minimal interrupt latencies. (3) Minimal jitter: disturbance in the periodic task response time. (4) Multithreading support: while one application may be a sequential one requiring only a single task to react to a periodic event, more and more applications consist of many periodic segments requiring multithreading support. (5) Priority CDC: Embedded applications often predefine the sequence of executing application tasks. Therefore, efficient priority scheduling support is of utmost importance. (6) Kernel: A small kernel, often including just a thread management function, a scheduler function, and combinatorial functions for thread waiting, is adopted.

2.6.2. Scheduling Algorithms

In a real-time system, it is often necessary to execute jobs according to a defined temporal order and with a specific timing constraint. The latter can be either hard or soft. When a timing constraint needs to be met strictly, missing it means the job is said to be

tardy or missed and the system is considered to be in error, even if such tardiness does not affect the remaining jobs of the system. Hard real-time systems usually find application domains in safety-critical processes, such as the brake control system of automotive equipment. On the contrary, for some applications in specific domains, missing the timing deadline of some jobs may be acceptable, provided that the timing limitations of other jobs are satisfied. Besides, the system integrity is not at risk at all for those missing jobs. In that case, the system is considered to be soft real-time. Soft real-time systems are often used in non-critical applications, such as multimedia ones.

In this section, we will overview the scheduling algorithms used in hard and soft real-time systems. Some of the scheduling algorithms need to ensure that the timing constraints for all the jobs are satisfied. Other scheduling algorithms do not need to ensure that all the timing constraints are met. Such scheduling algorithms enable the system administrator to allocate a higher priority for defining the timing constraints of some jobs than others. For soft real-time systems, some scheduling algorithms will allow the missed jobs within certain limits, but ensure that the timing constraints for the other jobs in the job pool are satisfied. These scheduling algorithms are also known as best-effort algorithms. We assume that every task is a periodic task if there is no special mention. Task scheduling can be divided into two classes. These are periodic task scheduling and aperiodic task scheduling.

2.7. Integration of IoT in Medical Equipment

The adoption of the Internet of Things paradigm has opened even more opportunities in several domains, including the one for monitoring medical parameters, as what will be discussed in this chapter. Captain Medical has realized a spirometer instrument that aims to solve the use problem of spirometric measures in hospitals and clinics. The matrix from which this device arose represents a larger idea, focused on integrated medical devices capable of allowing telehealth in any situation. They are intended, first of all, for patients with chronic respiratory diseases, who need to evaluate their condition at home daily.

The advantage of the new design regards not only the ability to perform the measurements independently and the portability but also the remote control of the devices, making them easy to use and able to give early warnings in adverse situations. Yet this new device is not only a function for a smart spirometer. It is a platform to which it is easy to connect other medical devices targeted to the most disparate needs and parameters, during the possible exit of the patient from the monitoring with the logistic supervision of the clinic or authorized structure. In this phase, medical embedded IOT, also called e-IOT, enters in action. Considering this large new world of devices, we need

to understand if we can apply to them the known paradigms for conventional IoT, or if we need to modify them or adopt entirely new ones.

Therefore, we need to adopt a healthcare IoT or m-health IoT (where m stands for medical). To this aim, we review the three classic component segments for an IoT architecture: perception, transport, and application segments. The perception segment is responsible for acquiring environmental parameters and the object set of interest, and acting locally on the same object, using sensors, actuators, and microcontrollers. The transport segment transmits the information classified about the object of interest from the perception segment to the application segment. The application segment receives the information, processes them, and possibly sends back new instructions to the object of interest.

2.7.1. IoT Architecture

The integration of the Internet of Things (IoT) into various industries has made systems smarter and simpler. The IoT business is primarily focused on unifying all of the required devices under one umbrella and providing users with useful data to assist in making intelligent decisions. An IoT architecture is a conceptual structure that describes the organization of how IoT devices communicate with one another and with the cloud. Different devices are utilized across different IoT applications, and there are various ways in which devices can be integrated. Regardless of the type of devices being utilized, each IoT solution comprises hardware, connectivity, and software. Software, in addition to hardware, is what drives the recognition of an IoT solution. The core function of any IoT solution is data processing and analysis.

A simple IoT solution consists of four layers: perception, network, edge, and application. The perception layer possesses the devices that sense environment information; the network layer transports information between layers; the edge layer provides storage and computing service; and the application layer offers business functions to users. Smart cities, industries, and other checked areas make up an IoT application. The IoT perception layer consists of smart sensors, RFID tags, big/ultra HD cameras, and other sensing devices, which are capable of obtaining important status parameters from surrounding areas. Their abilities determine the coverage of the IoT system. The network layer consists of wireless networks, including cellular networks, Wi-Fi networks, LPWANs, and other modern communication techniques. Different data traffic, coverage, and latency requirements/preferences from users' applications cause a multi-tier and hybrid structure of the networking layer in most IoT applications, resulting in the challenges of data transmission for both reliability and latency requirements.

2.7.2. Data Analytics and Cloud Computing

The integration of smart medical equipment with data analytics and cloud computing permits the analysis of patients' health status, thus increasing quality achievement and assuring healthcare assistance costs' reduction and optimization, and their effectiveness. Data is transported to the cloud from the gateways, being processed through powerful Data Analytics frameworks allowing the coupling between processed medical data and patient information coming from the Electronic Health Records. Being deeply exploited, coupled, and correctly discussed by doctors, the listed health parameters allow personalized patient monitoring and allow their manual or automatic hospital services requesting or triggering alarm systems.

The data processing and analysis allow the detection of the specific patients' illnesses through pathology patterning, discovering depression by measuring the social avoidance level through phone log data and extracting behavioral patterns, and the promotion of connected health through the digital therapeutic.

2.8. Conclusion

Embedded systems have acquired a new dimension by allowing objects to be programmed, communicating, and historical data to be stored. Unprecedented amounts of computational power and memory, available at attractive prices, permit the implementation of features that were previously considered science fiction. Future products will be equipped with modular embedded systems with unique and innovative features. This abundance of hardware and embedded software offers tremendous opportunities to the developer. However, hardware and software have to be very carefully designed: they are at the same time system enablers and system obstacles: mistakes in the design of the system can make impossible to control costs or to optimize the reliability and the speed of the design process. Furthermore, possible errors in the design of the embedded software are at the same time very costly in terms of human resources and time and could trigger legal consequences due to the unintended action of the device. European and national regulations in the field of devices have been implemented or are in the process of being implemented in order to specify the requirements that a device has to satisfy, and to promote a uniform approach to the market throughout Europe in the field of certification.

The increase in the number of smart medical devices that are available on the market raises the problem of choosing one of the many available systems, as well as raises the requirements from the user. As technology becomes ubiquitous, additional layers of support and new functionalities are expected to be offered with new designs of embedded systems, fulfilling the specific needs of smart medical devices. The design of the system

should support the pack around the user, and the new wave of devices for every application, every age and every need, with.

2.8.1. Future Trends

Rising healthcare costs combined with technological advances have paved the way for provider and consumer interest in smart medical devices for health monitoring and patient record-keeping. Consumers, through their increasing access to health data online, are becoming more engaged in their own medical decisions and are communicating their needs and wants to physicians and other care providers. These changes have driven market demand for smart medical devices. Recent survey results indicate that health applications are the second most common functionality people want on their smartphone or tablet. For many, the smartphone is the preferred computing platform for health applications. Efforts are currently under way to enable the smartphone and the tablet as the preferred monitoring platforms for many smart medical devices. Advantages for using smartphones and tablets as smart medical device monitoring platforms include 1) ubiquity of the smartphone, 2) portability of the smartphone, 3) diverse sensor applications already built into the smartphone, 4) display functionality for alerts and graphics, 5) telecommunication capabilities, and 6) Web connectivity. Some of the many examples of smartphone platform development include blood-glucose sensor systems, electrocardiograms, and ultrasound imaging.

The smart medical device development effort must also pay attention to regulatory requirements. Although mobile-health applications must provide consumers with the data and information needed to make informed healthcare decisions and enable consumers to fulfill those decisions, consumers must also be assured of the reliability and functionality of the applications. The best way to provide consumers with this confidence is to develop smart medical devices that meet regulatory requirements. Fortunately, some capital-intensive device applications may have relatively low-volume produced systems, mainly because of the high costs associated with regulatory requirement compliance.

References

Firouzi, F., Farahani, B., & Marinšek, A. (2022). The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT). Information Systems, 107, 101840. SpringerOpen

Liu, Y., Zhang, L., Yang, Y., Zhou, L., Ren, L., Wang, F., Liu, R., et al. (2019). A novel cloud-based framework for the elderly healthcare services using digital twins. IEEE Access, 7, 49088–101. SpringerOpen

Ahmed, M. R., Mahmud, S. H., Hossin, M. A., Jahan, H., & Noori, S. R. H. (2018). A cloud based four-tier architecture for early detection of heart disease with machine learning algorithms. In 2018 IEEE 4th International Conference on Computer and Communications (ICCC) (pp. 1951–1955). IEEE. SpringerOpen

Mathew, P. (2018). Applications of IoT in healthcare. In Cognitive Computing for Big Data Systems Over IoT (pp. 263–288). Springer. SpringerLink

Jagadeeswari, V. (2018). A study on the medical Internet of Things and big data in the personalized healthcare system. Health Information Science and Systems, 6(14). SpringerLink