DESIGN OF AN EMBEDDED SYSTEM: BEDSIDE PATIENT MONITOR

Ertan Ozturk[1], Ozan Emre Yapıcı[2], Mehmet Unal[2] and Osman Çakıcı[2]

[1]School of Electrical Engineering and Computer Science, University of North Dakota, Grand Forks, ND, USA
[2]ERETNA Medical LLC, Maltepe, Istanbul, TURKEY

ABSTRACT

Embedded systems in the range of from a tiny microcontroller-based sensor device to mobile smart phones have vast variety of applications. However, in the literature there is no up to date system-level design of embedded hardware and software, instead academic publications are mainly focused on the improvement of specific features of embedded software/hardware and the embedded system designs for specific applications. Moreover, commercially available embedded systems are not disclosed for the view of researchers in the literature. Therefore, in this paper we first present how to design a state of art embedded system including emerged hardware and software technologies. Bedside Patient monitor devices used in intensive cares units of hospitals are also classified as embedded systems and run sophisticated software and algorithms for better diagnosis of diseases. We reveal the architecture of our, commercially available, bedside patient monitor to provide a design example of embedded systemsrelating to emerged technologies.

KEYWORDS

Patient monitors, embedded systems, System on Module, cross-building, embedded hardware.

1. INTRODUCTION

Embedded systems are application specific computer systems such that include hardware and software components based on the target application such as household electronic equipment, automobile electronic systems, communication devices, defence and space equipment and medical devices, etc. Similarities between general purpose computers and embedded systems are; both have the basic computer architecture consisting of a processing unit, a memory and Input/Output (I/O) devices, both run a software, and can be monitored and may have human interface. On the other hand, embedded systems are application specific computer systems, so their hardware and software are tailored according to the target application. Due to the very wide range of applications, an embedded system can be from a very tiny Microcontroller (MCU) based device with a few kilo byte memory and storage, and several Mega Hertzprocessor speed to a System on Chip (SoC) based device having a Giga Byte (GB) level memory and storage as well as Giga Hertz (GHz) level processor speed. Embedded systemshave been a hot topic for industrial applications for the last decade, however instead of system level design of embedded hardware and software, the academic papers mainly focus on the improvement of specific features of embedded software/hardware [1-2] and the embedded system designs for specific applications [3-4].

Bedside Patient Monitor devices that provide momentarily display of the multiple physiological signals of patients are also considered as embedded systems. A generic Patient Monitor (PM) continuously monitors a patient's physiological signals which are electrocardiogram (ECG),

Respiration, Oxygen Saturation (SPO_2), Invasive Blood Pressure (IBP), Non-Invasive Blood Pressure (NIBP), Temperature and Carbon-dioxide (CO_2) in a waveform format (trace) and as numerical values [5]. The monitored physiological signals are interpreted by medical professionals for diagnostic purposes in intensive care units, emergency services, cardiological follow-up and post anaesthesia care units of hospitals. PMs are located next to the patient beds, so traditionally called bed-side patient monitor, the data (vital signals) collected by a bed-side monitor, is also transferred to a central system located in nurse rooms to monitor the conditions of all inpatients within a unit in a single big screen.

Early primitive vital signal monitors were started to be used in the 19th century, however modern PMs appeared in the mid-20th century following the birth of solid-state electronic, recently have been substantially improved parallel with advancements in digital electronics, software and communications technologies. High speed processors, high capacity memory and storage units used in modern PMs allow to run very sophisticated application software and advance diagnostic algorithms at the top. The usability of a PM and its interface by medical personal and the integration of multiple devices are very critical in critical care units [6], hence requires a user-friendly and adaptable software application running on it.

Monitor devices are also required to connect other devices to transfer their data and even to display transferred data within the same interface remotely. Hence, patient's vital signals are transferred to a central monitor device that is usually a high-power personal computer (PC) running a Central System software to display the vital signals of all patients away from the patient's bedside in a hospital unit. The advancement in communication technologies provide a high-data rate wire and wireless data transfer via Ethernet and Wi-Fi, respectively.Consequently, medical practitioners can observe their patients' physiological signals by using mobile devices which are connected to the central system.

Research on patient monitors is mostly driven by medical industry, but the design and the development of patient monitor devices by the industry are generally not disclosed due to business reasons. Manufacturers of PM mostly publish White Papers about their products which mainly focus on the functionality of the PM without getting into its design [7]. Universities also conduct researches on patient monitoring such as given in [8-13], however these works and others available in academic literature don't consider the hardware and software designs of a full PM for the monitoring of all vital signals. These works mainly focus on the monitoring of one or two vital signals by using simple off-the shelf-cards and implantation of simple software for receiving and sending signals.

On the other hand, remote-healthcare that requires a remote access to patient's monitored vital data has been the subject of research and development works for more than a decade [14]. The Internet of Things (IoT) has been widely identified as a potential solution for remote healthcare, and has thus been the focus of much recent research [14-16]. Obviously, these works more focus on the remote access technologies and its security.

In this paper, we present the hardware and software design of our patient monitor device which runs our developed application software consists of the back-end and the front-end to monitor all vital signals of a patient. Our monitor also has remote patient monitoring capability. Hence the main contribution of this paper is in two folds: The first is presenting an approach to design an embedded system which can run an embedded operating system (OS) and a sophisticated application software, over a main board that can be designed by using emerged electronic technologies. Then, the second is to present the state-of art architectural design for the hardware and software of a bed side patient monitor available commercially first in the literature. The rest of the paper is organized as: The section 2 reveals first, the design of a hardware hierarchy from a

SoC to an Embedded Board, then the attributes of embedded software design. The section 3 presents the architecture of our patient monitor device including its hardware, software and network communication. Section 4 concludes the paper.

2. DESIGNING EMBEDDED SYSTEM

2.1. Embedded Hardware

An embedded hardware includes a processing unit, memory and I/O devices based on the applications; for instance, a sensor board includes a sensor, Analog Digital Converter (ADC), and a communication interface, whereas an advanced embedded system may include keypads, display units, touch screens, various communication interfaces to communicate with external devices.

Although, there are off-the-shelf embedded boards available on the market to be used for various applications; an application specific embedded hardware/board may still need to be designed. From the processing point of view, there are two options; Microcontroller and System on Chip (Soc). Microcontrollers include a processor, memory power management circuits, timing resources, communication interfaces, and analogue interfaces in a single chip.

On the other hand, a SoC is a newer technology, and also include a processor/CPU, memory, and communication interfaces in a single chip. However, the speed of the processor in a SoC, typically in the level of several hundred MHz or even GHz, while it is typically from 20 MHz to 100 MHz for microcontrollers. In terms of the capacity of the memory; microcontrollers typically have Kbytes level memory, while SoCs may have GB level memory. SoC also offer more diverse communication interfaces compared to microcontrollers like the support for an USB interface.

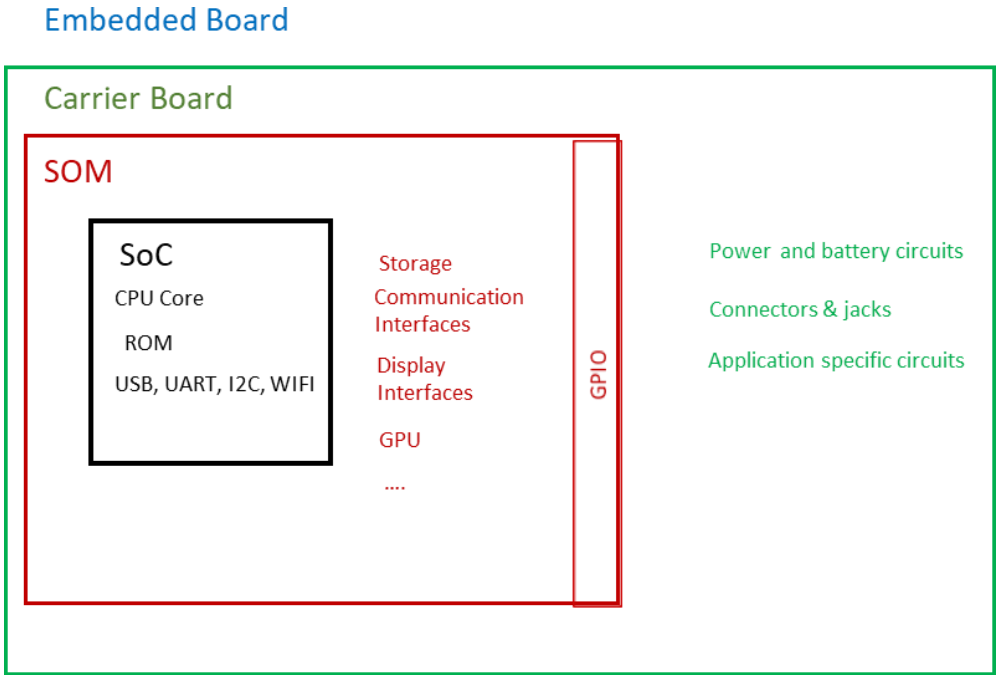

Figure 1. Hardware hierarchy from a SoC to an Embedded Board

Recently, another technology called System on Module (SOM) has emerged [17]. A SOM has more hardware units than a SoC such that it includes a large storage, audio and displayinterfaces, communication interfaces, camera interface. On the other hand, a SOM is not asingle chip,

instead it is a module in the size of a credit card, even smaller, but still needs external circuits to be a complete self-sufficient embedded board. Hence a SOM includes General Purpose Input Output (GPIO) pins as many as few hundreds in connection to driver circuits of the application designed in a carrier board. Consequently, a SOM and its carrierboard constitutes a self-sufficient embedded board as seen in Figure 1, where the hierarchy from a SoC to SOM, then to an embedded board is shown. The SOM module includes GB level storage, built-in communication and display interfaces, besides other processors like Graphical Processor Unit (GPU) etc. On the other hand, the application specific circuits such as display, Communication, audio circuits as well as power and battery circuits and the connector and jackscan be mounted on the carrier board.

2.2. Embedded Software

An embedded system whether it is a tiny microcontroller based or a powerful SoC based includes a software to perform a targeted task. The complexity of the embedded software based on the targeted task. For instance, a temperature sensing board requires just few lines of codes, whereas a medical device like patient monitoring systems requires very sophisticated software written preferably by object-oriented programming.

Requirements for an Embedded System Software are reliability, efficient memory use, low hardware requirement, tailored to hardware, and low power consumptions. The complexity of the application software also determines the required hardware, since the software of basic applications are not complex, they can be directly embedded into processing unit (microcontroller), hence they are called *Firmware*. Firmware programming is the writing codes directly into the processor without an operating system. This programming is written specific to hardware, which requires to consider the specific build of the hardware. High level languages, for example C programming, can be used for firmware programming, however in such case it is different than regular high-level C, and called Embedded C that includes specific syntax to control the ports and memory of hardware. Low level Assembly languages are also used for embedded firmware programming.

On the other hand, a complex software cannot be embedded directly into the processing units, instate an Operating System (OS) is used between the application software and embedded hardware. However, since the system is still application specific, the OS can also be tailored according to the hardware and the application requirements, so it is called Embedded Operating System.

An embedded OS consists of three parts: Boot-Loader, Kernel and Root File-System. These parts need to be compiled for the target embedded hardware in a powerful machine (Host) by cross-compiling, then can be flashed in to the target embedded system.

Therefore, the other unique feature of embedded software is cross-compiling that means compiling the developed application software and OS in a powerful host machine i.e., in a PC for the targeted hardware, described below.

2.3. Host and Target Systems

Embedded systems even SoC based ones have limited hardware and software capabilities than those of personal computers, Hence, Embedded Software Development is mostly done in a personal computer called a *Host System*. Then, the embedded system for which the software is developed is called a *Target System*. Host-Target method is used in the development of both MCU based and SoC based embedded systems. In a Host System, the developed software codes

whether it is written by using high level programming or assembly programming, needs to be built for the Target System, this is called *Cross-Building*. The application software can be built also for the host system, and executed on it. This is called *native build* compared to cross-build. Native build provides to run, debug and test the developed software first on the powerful host system. After cross-building the software, it is installed in the target system by using an external Hardware/software tool called flashing tools. The Figure 2 illustrates a cross-compiling of an application software in a host machine, then flashing it in to the target Embedded system.

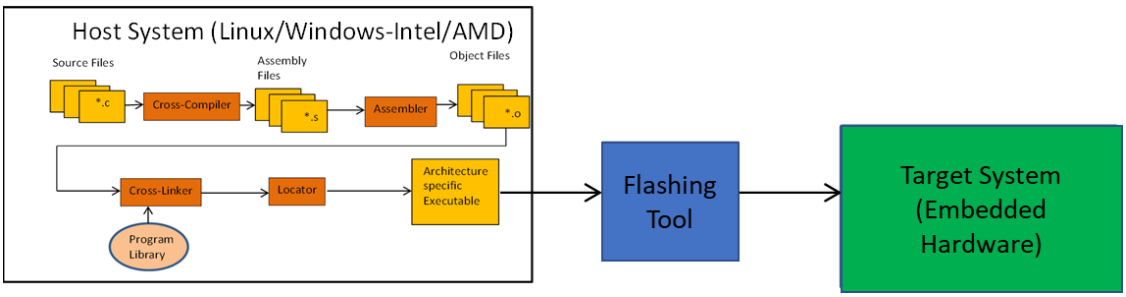

Figure 2. Cross-building of an embedded software

2.4. Embedded Communications

In an embedded hardware, the processing unit communicates with other Integrated Circuit (IC) or modules on the board or with external peripheral boards and devices attached to it. For example, the processor unit may communicate with on-board sensor IC, power- management IC, also with on-board modules such as GSM, WiFi, Bluetooth, SD-Card modules. The supported communication interfaces by SoC and MCU are serial parallel interface (SPI), Integrated-to-Integrated Circuit (I^2C) and Universal Asynchronous Receiver / Transmitter (UART), which are ideal for the above communications where there is no need for high speed and no long-distance data transfer.

3. HARDWARE AND SOFTWARE ARCHITECTURE OF OUR PATIENT MONITOR

Our bedside patient monitor shown in Figure 3 consists of our designed and third-party hardware components. Obviously, the third-party hardware units and the mechanical design of the patient monitor are not in the scope of this paper.

Our bedside monitor can continuously monitor and record ECG with 7 types of Arrhythmia, Respiration Rate (RR), SPO2, CO2, NIBP, End-tidal CO2 (EtCO2), up to 4 Channel IBP and dual-TEMP parameters. It has adaptive properties for adult, paediatric and new-born patients. It provides multi-parameter waveforms, alarms, and status messages. It allows the monitored functions to be recorded and to be monitored again when required. The monitor has the ability to connect to a local network Laser and thermal printer. Physiological signal waveforms, reports and tables generated by the monitor can be printed out via network printers. Our patient monitor device complies with IEC 60601-1 medical standard [18], hence it is CE certified and has been using recently in several different hospitals in Turkey [19].

Figure 3. ERETNA bed side patient monitor

3.1. The Embedded Hardware

The high-level block diagram for the hardware depicted in Figure 4 presents our designed hardware in green colour, whereas the third parties are in blue colour. Patient Monitors collect patient's physical signals via sensors attached to the patient body. All analogue data from sensors are feed to the Physiological Data Acquisition Module, in which analogue data is converted to digital and based on the Acquisition Module and relating vital signal. Digital dataat the output of acquisition module is fed to the main-card at the top of which our application software run.

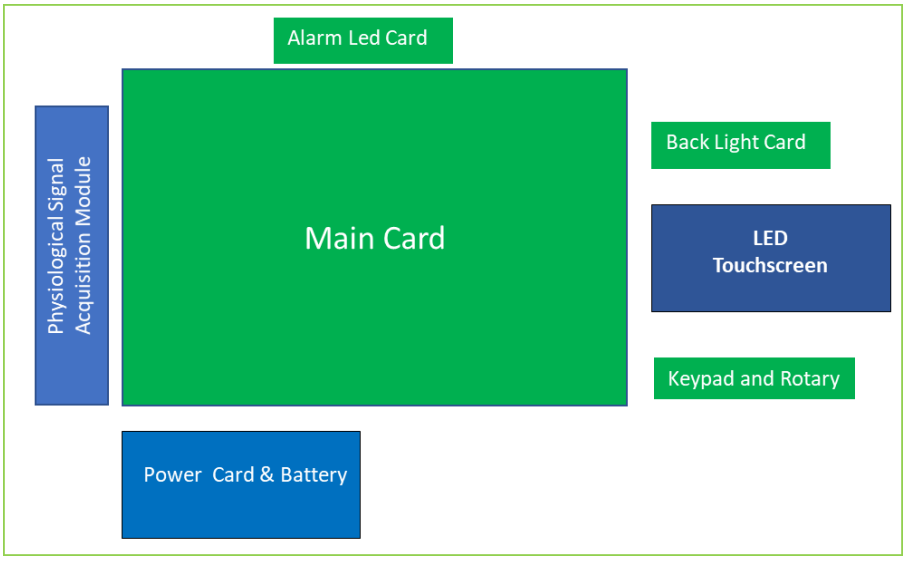

Figure 4. Hardware of the Bedside Patient Monitor

Our main card was design based on the system requirements and the software requirements, andit was built around an ARM processor-based SOM. The main card includes the drive circuits to interface the peripheral devices and modules such as an audio circuit for alarm soundings, a LVDS circuit that drives LED touch screen display, a network circuit for ethernet and Wife connections. In addition, the main card includes a power circuit to convert AC power to DC and a battery charger circuit, then a voltage regulator circuit to power the ICs with 3.3V, 5V voltages, also a circuit for the keypad. The drive circuits for data communication interfaces (UART, SPI, USB) are implemented in the main board to provide communication between the main card and the physiological signal modules, also provide extra USB ports. Finally, the connectors and jacks to connect the main card to external units of the system are embedded in our main card.

The Back-light card shown in Figure 5 was built to supply current to the LED touchscreen. The alarm led card including coloured LEDs was built to reflect the visual alarming feed by the main card. The keypad includes 10 buttons with their wirings, a red-green LED as on/off indication of the monitor, and an ambient light sensor to adapt the light of the screen according to the ambient lighting.

3.2. The Application Software

The development environment for our application software is a cross-platform design and development tool, Qt [20] running at the top of a regular PC with an Ubuntu Linux operating system (Host). The cross-platform design and development tool allows to cross-built our application software for our embedded Linux and ARM based architecture (Target) as seen in Figure 5.

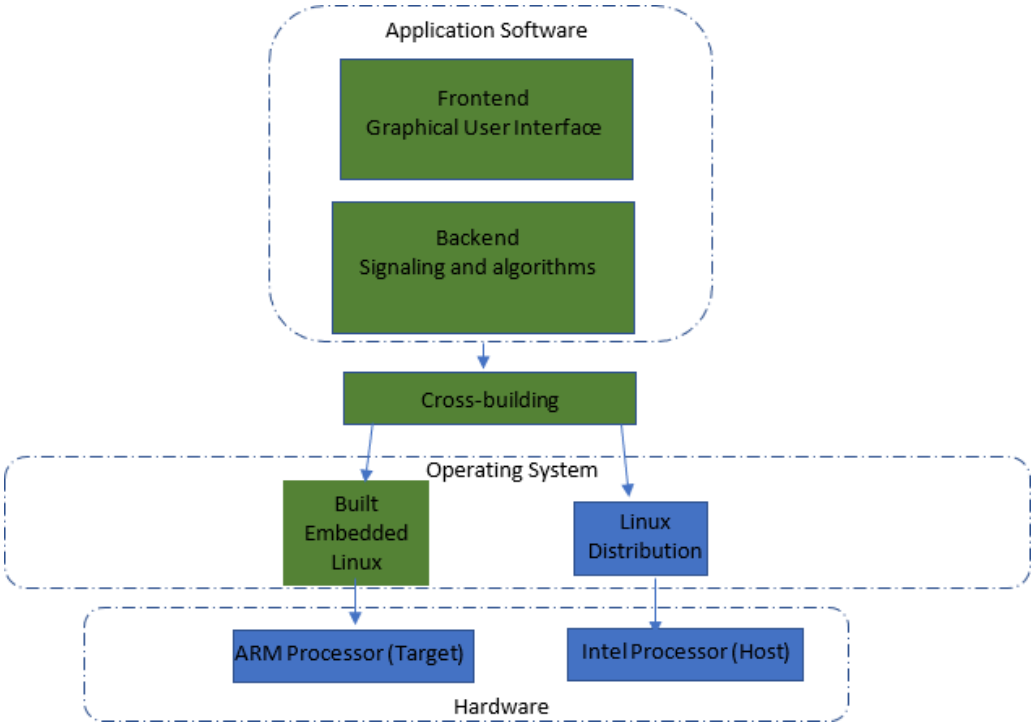

Figure 5. Cross-building of the system

Our application software was developed by using C++ on the Qt platform that includes tool-chains for cross building and compiling. The application software basically handles two parts, the backend and the frontend. The backend is the part that communicates with the hardware and includes the implementation of the physiological signal algorithms and calculations. It enables the data to be recorded or exported. There are several implemented communication modules in the backend such as the data transfer to other devices on the network, reading data from the monitor keypad or touch screen, screen brightness settings, battery information.

The data signals processed by the backend are sent to the graphical user interface, which is QML based frontend for the visualization, hence interfacing with users. The interface is aimedat being user-friendly, reducing the errors for busy medical personnel. Hence, the front-end was designed to make the interface simple and without any confusion for the users, while keeping the most significant vital parameters on the main screen and the others available based user's preference in the setting section menu. The screen view of the application software is seen in Figure 6.

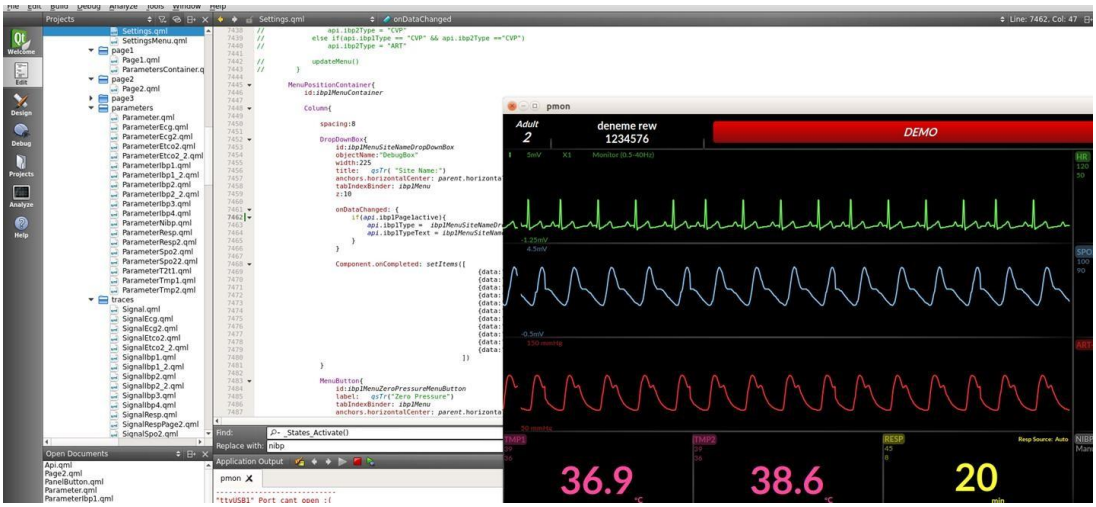

Figure 6. The application software

The most significant attribute of our application software is that it can be built to run on various operating system (Linux, Windows, iOS, Android) and hardware platforms (ARM, Intel, AMD processors based). Consequently, the central system software that collects all patient's data from bed-site monitors in a unit via a local network to display all in a single big screen, is the multi-windows version of our application software and it is cross-built for Intel based personal computers having Linux or Windows OS. Moreover, our application software is cross-built for Android and iOS operating system to run over a mobile device, tablet or smart phone.

3.3. The Embedded Linux OS

Pre-built Linux distributions are readily available; however, they are typically very large, so may not be proper for light hardware embedded systems, besides they are not available for all hardware architectures, and they are not easy to customize. Hence, we built our own Embedded Linux by using the *buildroot* tool available online [21]. That tool allows to include processor architecture, required modules and libraries in the built process. Hence, our embedded OS is much lighter than the available Linux distributions to relax the hardware of the bed side patient monitor device.

3.4. Network Communications of our Design

Our monitor is capable of Ethernet and Wi-Fi connectivity through a switch that makes up our local subnet. The data transmission rate between the bedside monitor and the central system and between the bedside monitor and the mobile device is over the UDP protocol with a rate of 80 packets per second. While the patient's data is transferred from the bedside monitor to the central monitor, the configuration commands from the central monitor are transmitted to the bedside monitor. In addition to UDP, multicast and broadcast transmissions are used to reach mobile devices in the network, hence the connection is unidirectional from the bedside to the mobile application. The size of a UDP packet as bytes per second depends on the transferred vital signal. In order to reduce the heavy traffic on the network, the bedside monitor does not send all signals. Only the signals of the active parameters are transmitted to the other party. Ifthe user wants to activate a parameter at any time, the activated parameters and signals are automatically transmitted to the other party.

Another network connection of our bedside monitor is with hospital information systems (HIS) that is done via HL7 protocol [22]. The vital data along with the patient's demographic data is sent once every 10 seconds in the form of an ORU message defined in HL7 version 2.3. Consequently, our patient monitor system is ready for remote healthcare due to the availability of network connection and its flexible cross-built software.

4. CONCLUSIONS

In this paper, we first explain the design of an embedded system to run sophisticated application software at the top of an embedded operating system, in order to provide a guide for researchers who are new in this field. Then, we present the state of art software and hardware architecture of our patient monitor that is commercially available. Obviously, the business confidential details of our design are not disclosed. Nevertheless, it provides an overview picture of the design relating to the recent hardware and software technologies, which is not available in the literature. Finally, the design of our patient monitor provides an easy expansion to remote patient monitoring within the concept of remote healthcare due to its flexible and the support of multi-platforms.

REFERENCES

[1] Perri, S. Spagnolo F., Frustaci, F, Corsenello P., (2023) "Design of Leading Zero Counters on FPGAs", *IEEE Embedded Systems Letters*, Vol. 15, No. 3, pp.149-152.

[2] Nagesh, K., Landsnes, O., Fuglestad, T., Svensen, N., Singhal, D., (2019) "Enhanced Embedded Linux Board Support Package Field Upgrade – A Cost Effective Approach", *International Journal of Embedded Systems and Applications (IJESA)*, Vol 9, No.1, pp.11-20.

[3] Lavanya, A., Jeevitha, M., Bhagyaveni, M. A., (2019) "IoT-Enabled Green Campus Energy Management System", *International Journal of Embedded Systems and Applications (IJESA)*, Vol 9, No.2, pp.21-35.

[4] Correa, C., Dujovne, D., Bolano F., (2023) "Design and Implementation of an Embedded Edge Processing Water Quality Monitoring System for Underground Waters", *IEEE Embedded Systems Letters*, Vol. 15, No. 2, pp.81-87.

[5] Khandpur, R.S. (2014), *Handbook of Biomedical Instrumentation, Chapter 6. Patient Monitoring System*, 3rd Edition, McGraw Hill Education (India) Private Limited.

[6] Andrade, E., Quinlan, L., R., Harte, R. Byrne, D., Fallon, E., Kelly, M., O'Connor, P., (2018), "State of the Art and Future Trends in the Usability of Patient Monitors", *International Conference on Human Systems Engineering and Design: Future Trends and Applications*, pp.338-344.

[7] Philips Patient Monitoring Solution, https://www.usa.philips.com/healthcare/solutions/patient-monitoring, [Online] Available, (accessed Oct 28,2023)

[8] Ramya, V., Palaniappan, B., Kumari, A., (2011) "Embedded Patient Monitoring System", *International Journal of Embedded Systems and Applications (IJESA)*, Vol.1, No.2, pp.51-63.

[9] Kang, J., Yoo S., Oh D., (2013), "Development of a Portable Embedded Patient Monitoring System", *International Journal of Multimedia and Ubiquitous Engineering*, Vol.8, No.6, pp.141-150

[10] Visvesvaran, C, Karthikeyan, N. K., Kumar, J. B., Kaviya, P., Kaviya S., (2022) "Advanced Patient Monitoring and Alert System with Auto Medicine Suggestion using Machine Learning", *Proceedings of the Third International Conference on Electronics and Sustainable Communication Systems.*

[11] Kanchikere, J. (2019), "Embedded patient monitoring system", International *Journal of Power Electronics and Drive System (IJPEDS)*, Vol. 10, No. 1, March 2019, pp. 388-397.

[12] Balakrishnan, (2022) "Sensor Based Health Monitoring System Using Embedded Technology", *8th International Conference on Advanced Computing and Communication Systems (ICACCS).*

[13] Jain N. P. (2012), "An Embedded, GSM based, Multiparameter, Realtime Patient Monitoring System and Control An Implementation for ICU Patients", *World Congress on Information and Communication Technologies*.

[14] Baker, S.B., Xiang, W., Atkinson, I. (2017), "Internet of Things for Smart Healthcare: Technologies, Challenges, and Opportunities", *IEEE Access*, Vol.5, pp. 26521-26544.

[15] Al-Fuqaha, A., Guizani, M., Mohammadi, M. Aledhari, M., Ayyash, M., (2015), "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications", *IEEE Communication Surveys & Tutorials*, Vol.17, No.4, pp. 2347-2376.

[16] Islam, R., Kwak D., Kabir, H., Hossain, M., Kwak, K., (2015), "The Internet of Things for Health Care: A Comprehensive Survey", *IEEE Access*, Vol.3, pp.678-708.

[17] Customized system on-modules and embedded electronics for when versatility and time to market matter, [Online] Available: https://beaconembedded.com/ (accessed Oct 28, 2023).

[18] IEC 60601-1-11:2015, Medical electrical equipment — Part 1-11: General requirements for basic safety and essential standard: Online https://www.iso.org/standard/65529.html, (accessed August 28, 2023).

[19] ERETNA Medical, [Online] Available: https://www.eretna.eu/home, (accessed Oct 28,2023).

[20] Qt Framework, [Online] Available: https://www.qt.io/product/framework, (accessed Oct 28, 2023).

[21] Buildroot Making Embedded Linux Easy, [Online] Available: https://buildroot.org/, (accessed Oct 28, 2023).

[22] HL7 International [Online] Available: https://www.hl7.org/ (accessed Oct 28,2023).

AUTHORS

Ertan OZTURK is a faculty member in the School of Electrical Engineering and Computer Science at University of North Dakota. He got his MS and PhD degrees in Electrical and Computer Engineering at Illinois Institute of Technology (IIT), Chicago. Previously, he was the technical leader of an embedded software and hardware team to develop and manufacture a patient monitor system at ERETNA Medical located in Istanbul, Turkey. He is the author 40 peer reviewed Journal and Conference papers, and he has supervised 2 PhD and 12 Master theses.

Ozan Emre YAPICI is recently the technical leader and senior embedded software engineer at ERETNA Medical, Istanbul. He got his BS degree in Computer Engineering from Gebze Technical University, Turkey. Previously he worked as a software engineer for image recognition and processing at Panel Yazilim, Istanbul.

Mehmet UNAL is the project manager at ERETNA Medical. He is also specialized in intensive care medical devices. He got his BS degree in Biomedical Technology from Electronics and Computer Department at Marmara University, Istanbul.

Osman ÇAKICI got his BS degree in Biomedical Technology from Electronics and Computer Department at Marmara University, Istanbul. He is the founder and the CEO of ERETNA Medical that is an R&D company for medical devices. He is also the founder of MESA Medical that has provided solutions and technical services to intensive care units of more than 100 hospitals in Turkey.