International Journal of Pure and Applied Mathematics
Volume 117 No. 16 2017, 719-726
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
Special Issue
ijpam.eu

DESIGN AND IMPLEMENTATION OF RECONFIGURABLE RTOS SMART IP SURVEILLANCE CAMERA WITHIN INTERNET CONNECTED MINE MONITORING SYSTEM

[1]R.Elavarasi, [2]M.Anand, [3]S.Sathya
[1]Research Scholar, Dr.M.G.R Educational and Research Institute University, Chennai-95.
[2]Professor, Dr.M.G.R Educational and Research Institute University, Chennai-95.
[3]Research Scholar, Dr.M.G.R Educational and Research Institute University, Chennai-95.

Abstract: The time of delivering the embedded system solution is very important, because of the highly competitive market and demand of the consumer needs. In Real Time Operating System (RTOS) based embedded systems, the consistently modify yield inside the predetermined time. The virtual stage gives the huge preferred standpoint in creating; testing and troubleshooting the RTOS based embedded system. This work described the design and improvement of a reconfigurable virtual stage for ongoing part. High tech Mine monitoring system inserted in camera and sensors interfaced with microcontroller. FreeRTOS is used to reconfigure the system. So Mine monitoring system additionally adds the IP Surveillance Camera is monitor the place is an inserted web server to provide smart web based services. To the monitoring and management services and other emergency handling facilities is called as multitasking so that in this method based on FreeRTOS. FreeRTOS utilized will plan the undertaking, for example, IP surveillance camera task; Mine monitoring task, Web server task and Ethernet task. FreeRTOS is used to reconfigure the system. From the execution of this design structure on real system, it can build the productivity of the FreeRTOS based applications.

Keywords: FreeRTOS, IP Surveillance Camera, Ethernet, Web server, multitasking.

1. Introduction

Embedded systems are normally intended for different purposes, for example, to control or to process information. Characteristics of real-time system include meeting certain deadlines at the right time. To achieve this purpose, real-time operating system (RTOS) is often used. FreeRTOS is a part of software with a set of APIs for users to develop applications. Utilizing a RTOS does not ensure that the system will constantly meet its deadlines, as it relies upon how the general system is composed and organized. While FreeRTOS for embedded systems are dominatingly used in top of the microchips or microcontrollers with 32-bit central processing unit (CPU), there is an expanding pattern to give these highlights in the mid-extend (16-bit and 8-bit) processor systems. An Operating System (OS) is a bit of programming that deals with the sharing of assets in a PC system. FreeRTOS is frequently separated from nonexclusive OS as it is particularly intended for booking to accomplish constant reactions. There are a number of variants of RTOSs available nowadays; they range from commercial, proprietary, to open-source RTOSs. For small-scaled embedded systems designed using small microcontrollers (i.e. microcontrollers with typical ROM of 128Kbytes and RAM of 4Kbytes)it is common perception that there is no need to have an RTOS. RTOS for this range of devices such as:

- Better and more secure synchronization: In small embedded system advancement without using any RTOS, worldwide factors are regularly used for synchronization among modules/capacities. However, especially in highly interrupt-driven system, using global variables lead to bugs and software safety issues. These global variables are often shared and accessed among the functions, and there is high chance of them being corrupted any time during the program execution. With a FreeRTOS in place, synchronization is safely managed and tasks can pass messages or synchronize with each other without any corruption problem.
- Resource management: Most RTOS provide APIs for developers to manage the system resources These include task management, memory management, time management, interrupt management, communication and synchronization methods. These features provide the abstraction layer for developers to freely structure the software to achieve the code.

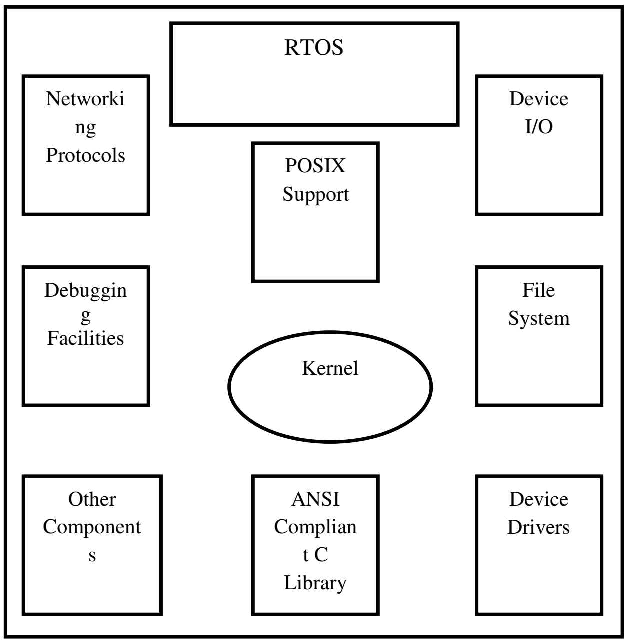


Figure 1. General Architecture of RTOS Kernel

The architecture of an RTOS is dependent on the complexity of its deployment. FreeRTOSs are scalable to meet different sets of requirements for different applications. For simple applications, an RTOS usually comprises only a kernel. For more complex embedded systems, an RTOS can be a combination of various modules, including the kernel, networking protocol stacks, and other components as illustrated in Figure 1.

An operating system generally comprises of two sections: kernel space (kernel mode) and user space (user mode). Kernel is the smallest and central component of an operating system. Its services include managing memory and devices and also to provide an interface for software applications to use the resources. Extra administrations, for example, overseeing protection of projects and multitasking might be incorporated relying upon design of working system.

2. Literature Survey

An operating system (OS) is a software that handles all the sources of a computer, both hardware and software, and provides an environment in which a user can implement programs efficient manner [1]. However, the principles of operating systems are Real time. In fact, operating systems have been developing through the years [2]. There were no operating systems in the early computers. In individuals systems, every program necessary for full hardware requirement is implementing properly and executes each minor task. Operating systems is necessary for growing difficulty of the computer hardware and the application programs. Initially, operating systems were not fully automatic is [3] defined an operating system as a set of manual and automatic procedures that allow a group of people to share a computer installation efficiently. Now a day's modern operating systems are fully automatic.

An embedded system is a real-time computer system. Some examples of embedded systems are: the microcontroller used to control the fuel in automobiles, software embedded in airplanes, industrial machines, microwave ovens, dryers, vending machines, medical equipment, and cameras [4]. Designing a real-time system involves the task partitioning and merging, and assigning priorities to manage response times [5-6]. Depending upon scheduling tasks, parallelism and communication [7] may be balanced. Merging highly reliable and parallel tasks for sequential execution may reduce overheads of context switches and inter-task communications.

The designer must determine critical tasks and assign them high priorities. When higher priority tasks are always ready to run, resulting in insufficient processor time for lower priority tasks [8-9]. Priority is an important part of RTOS. Tasks are scheduled by priority. Furthermore, good response times may require memory in resource-impoverished systems [10]. Clearly the choice of an RTOS in the design process is important for support of priorities, interrupts, timers, inter-task communication [11], synchronization, multiprocessing [12] and memory management.

3. Overview

The proposed approach is to focus on providing FreeRTOS on Embedded System. As mentioned earlier Embedded System is just a prototype and used to check the circuit functionality. The proposed method is to make the embedded system of the proprieties by using in FreeRTOS in simple application as a test model. Research is going ahead in executing the reconfigurable application as it is a perplexing task. Modified RTOS kernel is adopted to build the RTOS along with the system. It is the standard system used for commercial market and this system is built under the user define application. In this proposed method, FreeRTOS mainly deals Inter Process Communication and has the following mechanisms:

- Semaphores
- Message Queues
- Shared Memory Segments

Among IPC it deals with the concepts of Task Management, Semaphores and Preemptive Scheduling.

2.1 Task Management

• Creating a Real Time task and memory without delay: this is difficult in light of the fact that memory must be dispensed and a considerable measure of information structures, code section must be duplicated/introduced.

• The memory obstructs for Real Time tasks must blocked in fundamental memory to keep away from get to latencies because of swapping.

• Changing run-time needs is priorities are dangerous. It might change the run-time conduct and consistency of the entire system. The Task management is finished by using many tasks. There are, first task is Ethernet task. Second task is Web server task and Third task is IP surveillance task.

2.2 Task States

A task can exist in one of the following states shown in Figure 2.

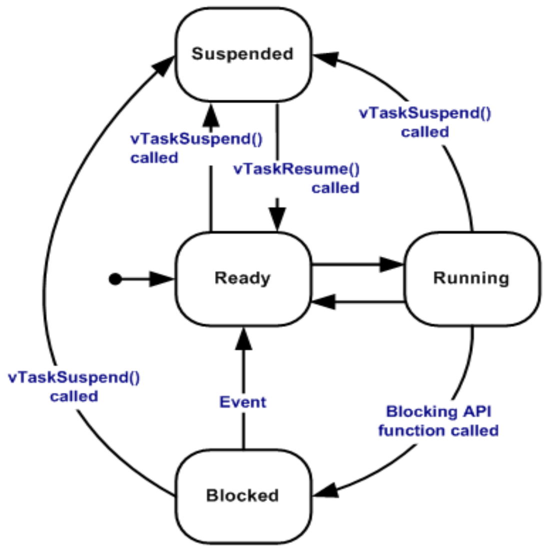


Figure 2. Task Management

- Running

When a task is actually executing it is said to be in the Running state. It is currently utilizing the processor. If the processor on which the RTOS is running only has a single core then there can only be one task in the Running state at any given time.

- Ready

Ready tasks are those that are able to execute but are not currently executing because a different task of equal or higher priority is already in the Running state.

- Blocked

A task is said to be in the Blocked state in the event that it is at currently waiting for either a temporal or external event. For example, if a task calls vTaskDelay () it will block until the delay period has expired - a temporal event. Tasks can also block to wait for queue, semaphore, event group, notification or semaphore event. Tasks in the Blocked state regularly have a "timeout" period, after which the undertaking will be timeout, and are unblocked, regardless of the possibility that the time event the task was waiting for has not occurred.

Tasks in the Blocked state do not use any processing time and cannot be selected to enter the Running state.

- Suspended

Like tasks that are in the Blocked state, tasks in the Suspended state cannot be selected to enter the Running state, but tasks in the Suspended state do not have a time out. Instead, tasks only enter or exit the Suspended state when explicitly commanded to do so through the vTaskSuspend () and xTaskResume () API calls respectively.

4. Real Time Hardware Task Model

The Microcontroller has facility to perform computations in hardware to increase the system performance, while maintaining much of the flexibility of software solutions. In modern development, the real-time embedded system is increasingly being built with RTOS namely Real Time Operating System by reducing cost and performance improvement. In order to verify the proposed approach designed with ARM Microcontroller.

The project aim is to design a mine monitoring system consists of sensor based monitoring add an IP Surveillance Camera that runs an embedded web server to provide smart web based services to the monitoring services and other emergency handling facilities. FreeRTOS used will schedule the task such as IP Surveillance Camera task, Mine monitoring task, Ethernet task and Web server task. A real time operating system is necessary to handle the timely events and other multitasking requirements of the project. Figure 3 shows that the blocks diagram.

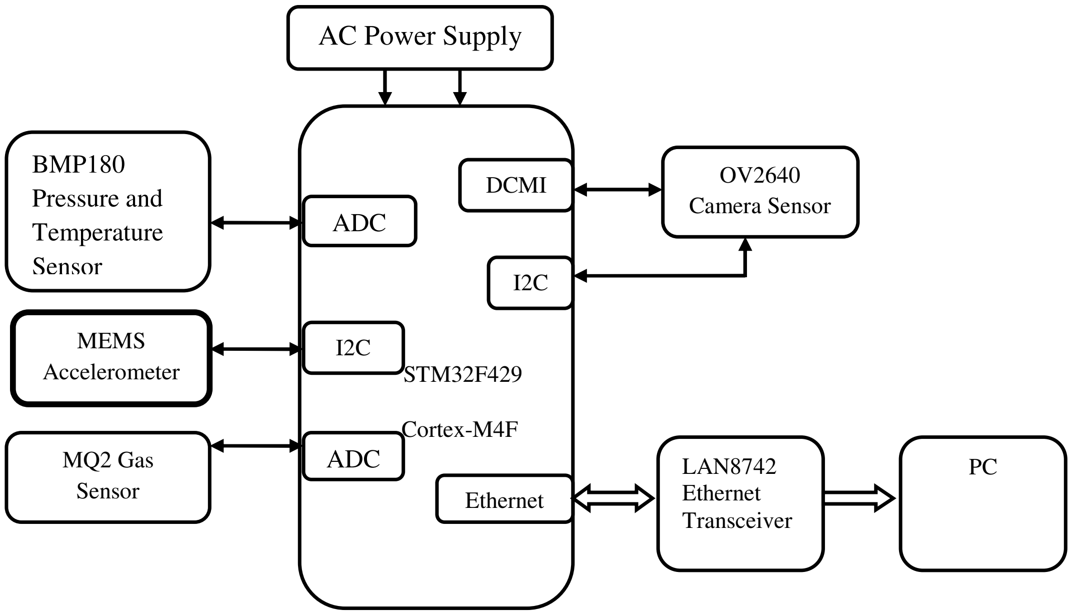


Figure 3. Block diagram of Street Lighting System

1) **IP surveillance camera** – allows a resident on the Mine to view the live video feed from the camera. Any internet browser can be used to view the video. This also helps the government authorities to use it for security and surveillance purposes.
2) **Live Video Feed** – The system consists of a camera sensor to stream a live video feed. When request is made the onboard microcontroller captures the JPEG images from the camera using the built-in DCMI peripheral and starts to stream it over the web in MJPEG compression format at an acceptable rate between. The image resolution is fixed at 470 x 272. The microcontroller has a large RAM memory area, about 256KB, which is a must for this kind of application.
3) **Environment Monitoring:** The device consists of different sensors to capture the local environmental conditions. Sensors are Gas sensor, Pressure sensor, Temperature sensor and MEMS Accelerometer. All these sensor readings are available on the web page for the control person to visualize.
4) **Microcontroller -** A project of this sort needs a very capable microcontroller with large amount of RAM memory. Thus STM32F429 from STMicroelectronics is chosen as the main MCU, which is an ARM Cortex-M4 based microcontroller that can run up to 180 MHz. It has 2MB of Flash memory and 256 KB RAM memory.
5) **IoT -** The Internet of Things (IoT) is the inter-networking of physical devices, vehicles, buildings and other items embedded with electronics, software, sensors, actuators, and network connectivity which enable these objects to collect and exchange data.
6) **Camera Image Acquisition using DCMI -** The system includes of a camera sensor to stream a live video feed. At the point when ask for is made the onboard microcontroller catches the JPEG pictures from the camera utilizing the implicit DCMI peripheral and begins to stream it once again the web in MJPEG pressure arrange at an sufficient rate between. The picture purpose is settled at 470 x 272. The microcontroller has a wide RAM memory region, around 256KB, which is a conclusive necessity for this sort of use.
7) **Security -** Before associating with the framework the client needs to enter the login username and secret word which is a genuinely essential safety effort to keep others from getting to the substance. Once logged in, the user allowed accessing all the data including the video feed.

In this proposed system design a real time IP surveillance Camera system. In this surveillance camera capture the image and the entire image sent through the

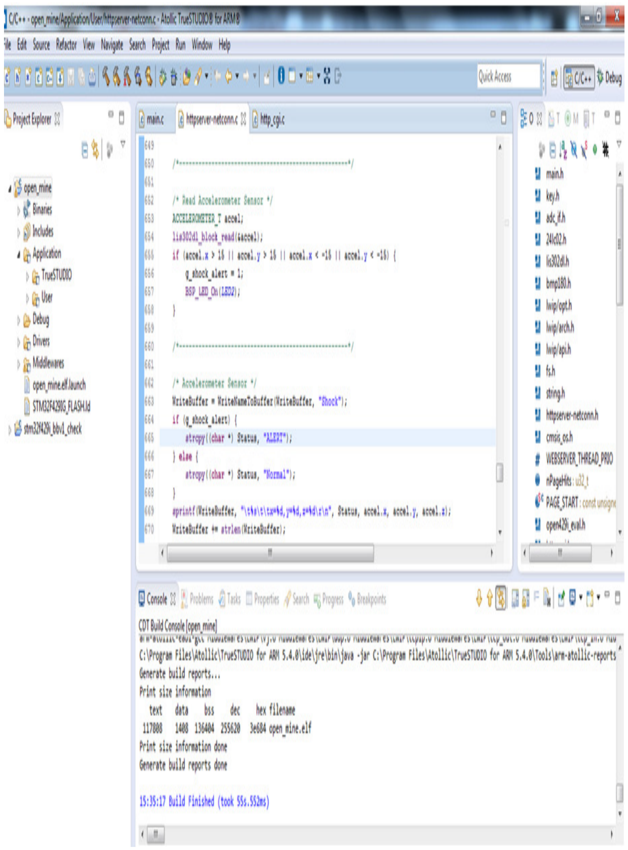
Ethernet. Finally video monitoring is possible through the internet. So in this proposed system have many tasks. There are Ethernet task, Web server task and IP Surveillance Camera task and sensor task. So FreeRTOS manage the multiple numbers of tasks. So that FreeRTOS scheduler ensures that tasks in the Ready or Running state will always be given processor (CPU) time in preference to tasks of a lower priority that are also in the ready state. In other words, the task placed into the Running state is always the highest priority task that is able to run. Ethernet task is a High priority task, Web server task is a Medium priority task and IP Surveillance Camera and sensor task is a Lower priority task.

5. Results And Discussions

Testing includes operation of a system under controlled conditions and assessing the results. Testing is done to discover the error in the program. It will increase the reliability of the software and improves the quality of the software. It finds the differences between the expected and observed behavior of the systems. The Attolic Studio software will be tested for its re-configurability using FreeRTOS. Figure 4 represents the software implementation.

Additional features

- Direct download into flash memory of most popular micro-controllers supported
- Full-speed USB 2.0 interface
- Serial Wire Debug supported
- Serial Wire Viewer supported
- Download speed up to 1 MBytes/second*
- Debug interface (JTAG/SWD/...) speed up to 15 MHz
- RDI interface available, which allows using J-Link with RDI compliant software



```
/*-------------------------------------------------*/

/* Read Accelerometer Sensor */
ACCELEROMETER_T accel;
lis302dl_block_read(&accel);
if (accel.x > 15 || accel.y > 15 || accel.x < -15 || accel.y < -15) {
    g_shock_alert = 1;
    BSP_LED_On(LED2);
}

/*-------------------------------------------------*/

/* Accelerometer Sensor */
WriteBuffer = WriteNameToBuffer(WriteBuffer, "Shock");
if (g_shock_alert) {
    strcpy((char *) Status, "ALERT");
} else {
    strcpy((char *) Status, "Normal");
}
sprintf(WriteBuffer, "\t%s\t\tx=%d,y=%d,z=%d\r\n", Status, accel.x, accel.y, accel.z);
WriteBuffer += strlen(WriteBuffer);
```

```
C:\Program Files\Atollic\TrueSTUDIO for ARM 5.4.0\ide\jre\bin\java -jar C:\Program Files\Atollic\TrueSTUDIO for ARM 5.4.0\Tools\arm-atollic-reports
Generate build reports...
Print size information
   text    data     bss     dec     hex filename
 117808    1408  136404  255620   3e684 open_mine.elf
Print size information done
Generate build reports done

15:35:17 Build Finished (took 55s.552ms)
```

Figure 4. Software Implementation

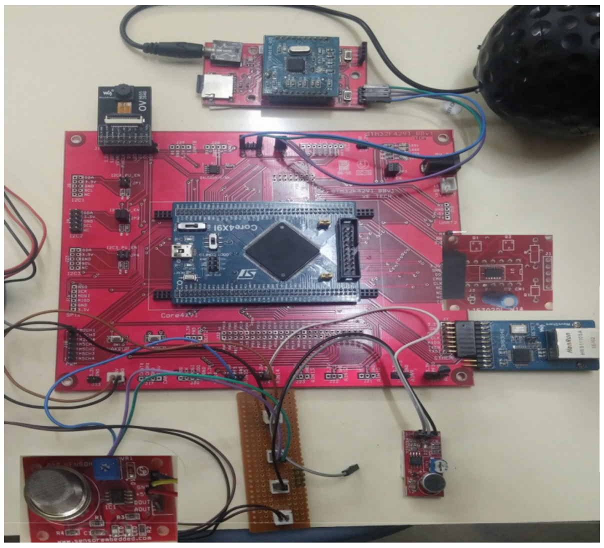
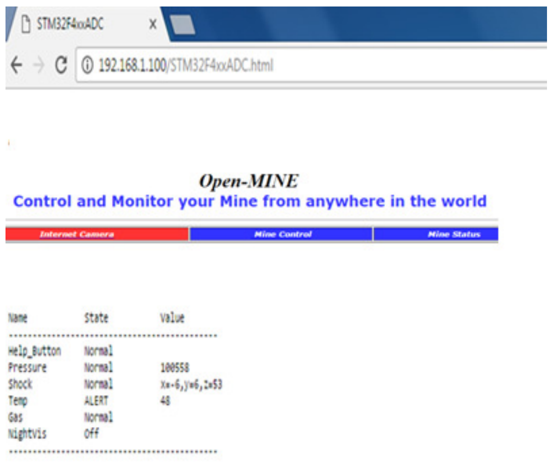


Figure 5. Hardware Implementation

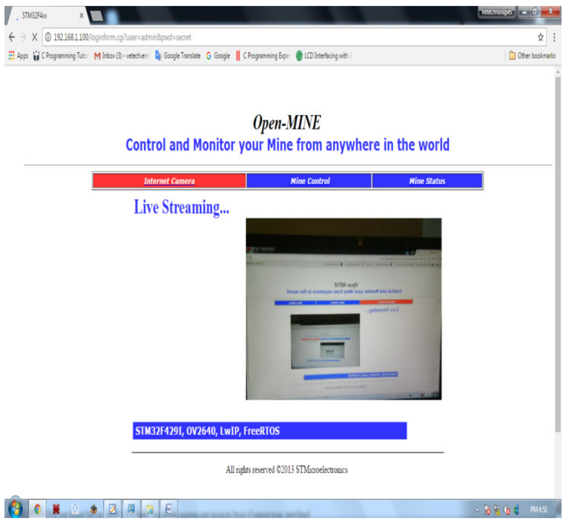
The design aim is to design an IP Surveillance Camera system that runs an embedded web server to provide smart web based services to mine monitoring in addition to the emergency handling facilities. FreeRTOS used will schedule the task such as IP Surveillance Camera task, Sensor task, Ethernet task and Web server task. A real time operating system is necessary to handle the timely events and other multitasking requirements of the project. Multitasking is a major part of FreeRTOS. FreeRTOS used to reconfigure the Mine monitoring system. FreeRTOS is used to reconfigure the system, for example first enable the Ethernet task and Web server task next enable the Camera task and Sensor task. Figure 5 shows that the hardware implementation.



Control and Monitor your Mine from anywhere in the world

```
Name          State         Value
------------------------------------------
Help_Button   Normal
Pressure      Normal        100558
Shock         Normal        Xe-6,y=6,z=53
Temp          ALERT         48
Gas           Normal
Nightvis      Off
------------------------------------------
```

Figure 6. Mine Monitoring Based Sensor Task


STM32F4xx
192.168.1.100/loginform.cgi?user=admin&pwd=secret
Open-MINE
Internet Camera
Mine Control
Mine Status
Live Streaming...
STM32F429I, OV2640, LwIP, FreeRTOS

Control and Monitor your Mine from anywhere in the world

All rights reserved ©2013 STMicroelectronics

Figure 7. Video Monitoring

Figure 6 and Figure 7 represents the Mine status and video monitoring. The term multitasking means that several sequential tasks are processed in parallel. However, on single processor systems, several tasks cannot run at the same time; therefore, task switches must be performed. This is the job of a multitasking system like RTKernel-32. So the user provides priorities. In this design higher priority is Ethernet task, Middle Priority is Web server task and Low Priority is IP Surveillance Camera task.

6. Conclusion

FreeRTOS allows an unlimited number of tasks to be run as long as hardware and memory can handle it. As a real time operating system, FreeRTOS is able to handle both cyclic and acyclic tasks. In RTOS, several functions are available to manage tasks: task creation, destruction, priority management and delay. More options are available to user, for instance to create a critical sequence or monitor the task for debugging purpose. For RTOSs preemptive multitasking is usually used. Preemptive multitasking uses priority based scheduling. This means a task can be stopped or suspended in order to allow a task of higher priority to run. In the case of two tasks of the same high priority each is given an equal time. A task is said to 'preempt' another task, when that task is interrupted and the context switched to a higher priority task. In the future it might even be possible to run efficiently re-configurable FreeRTOS. Then self-adaptation to evolving requirements of applications during their life time may become possible. The period of the Internet of Things requires a modular, configurable, and flexible FreeRTOS. The FreeRTOS will add better scalability, connectivity, security, safety, and an extended feature set to the solid real-time performance, low latency, and multi-core processor support of the RTOS of today. In future work FreeRTOS manage more number of tasks and reconfigure the application.

Reference

[1] Deng, Q., Wei, S., Xu, H., Han, Y. and Yu, G., 2005, December. A reconfigurable RTOS with HW/SW co-scheduling for SOPC. In *Embedded Software and Systems, 2005. Second International Conference on* (pp. 6-pp). IEEE.

[2] Gammoudi, A., Benzina, A., Khalgui, M. and Chillet, D., 2016, October. New Reconfigurable Middleware for Adaptive RTOS in Ubiquitous Devices. In *10th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*.

[3] Zaykov, P.G., Kuzmanov, G., Molnos, A. and Goossens, K., 2016. RTOS acceleration in an MPSoC

with reconfigurable hardware. *Computers & Electrical Engineering*, *53*, pp.89-105.

[4] Stahl, K., Stöcklein, J. and Li, S., 2015, August. Evaluation of Autonomous Approaches Using Virtual Environments. In *International Conference on Virtual, Augmented and Mixed Reality* (pp. 499-512). Springer International Publishing.

[5] Snehalatha, G., 2016. Implementation of High Speed Secure Communication Between Multiple FPGA Systems Using RTOS. *Networking and Communication Engineering*, *8*(4), pp.106-113.

[6] Muller, F. and Muhammad, F., 2009. Virtual Platform for Hw RTOS–Multiprocessor Hardware RTOS. *IEEE in Proc. Design Automation and Test in Europe,(DATE 2009), University Booth, Nice, France*, pp.21-23.

[7] Nedjah, N. and Figueroa, M., 2010. Modern development methods and tools for embedded reconfigurable systems: A survey. *Integration, the VLSI Journal*, *43*(1), pp.1-33.

[8] Xu, H., Han, Y., Wei, S., Deng, Q., Cui, J., Wang, S. and Yu, G., 2005, December. A remote wire/wireless video monitor system using HW/SW co-scheduling RTOS. In *Embedded Software and Systems, 2005. Second International Conference on* (pp. 6-pp). IEEE.

[9] Voros, N.S., Hübner, M., Becker, J., Kühnle, M., Thomaitiv, F., Grasset, A., Brelet, P., Bonnot, P., Campi, F., Schüler, E. and Sahlbach, H., 2013. MORPHEUS: A heterogeneous dynamically reconfigurable platform for designing highly complex embedded systems. *ACM Transactions on Embedded Computing Systems (TECS)*, *12*(3), p.70.

[10] Ghribi, I., Abdallah, R.B., Khalgui, M. and Platzner, M., 2016, July. I-Codesign: A Codesign Methodology for Reconfigurable Embedded Systems. In *International Conference on Software Technologies* (pp. 153-174). Springer, Cham.

[11] Gadalla, M. and Xue, D., 2017. Recent advances in research on reconfigurable machine tools: a literature review. *International Journal of Production Research*, *55*(5), pp.1440-1454.

[12] Janßen, B., Schwiegelshohn, F. and Hu, M., 2015, June. Adaptive computing in real-time applications. In *New Circuits and Systems Conference (NEWCAS), 2015 IEEE 13th International* (pp. 1-4). IEEE.

[13] S.V.Manikanthan and T.Padmapriya "Recent Trends In M2m Communications In 4g Networks And Evolution Towards 5g", International Journal of Pure and Applied Mathematics, ISSN NO:1314-3395, Vol-115, Issue -8, Sep 2017.

[14] M. Rajesh, Manikanthan, "ANNOYED REALM OUTLOOK TAXONOMY USING TWIN TRANSFER LEARNING", International Journal of Pure and Applied Mathematics, ISSN NO:1314-3395, Vol-116, No. 21, Oct 2017.