ТЕХНИЧЕСКИЕ НАУКИ
И ТЕХНОЛОГИИ

АВТОМАТТАНДЫРУ ЖӘНЕ БАСҚАРУ
АВТОМАТИЗАЦИЯ И УПРАВЛЕНИЕ
AUTOMATION AND CONTROL

DOI 10.51885/1561-4212_2023_3_126
IRSTI 50.43

**S.V. Vyazigin[1], N. Kunicina[1,2], M.E. Mansurova[1], Zh.E. Baigarayeva[1]**
[1]Al-Farabi Kazakh National University, Almaty, Kazakhstan
*E-mail: wismas1996@gmail.com**
[2]Riga Technical University, Riga, Latvia
*E-mail: nadezda.kunicina@rtu.lv*
*E-mail: mansurova.madina@gmail.com*
*E-mail: zhanel.baigarayeva@gmail.com*

# USING A DISTRIBUTED VIRTUAL MEMORY SYSTEM TO MANAGE A SMART HOME

# АҚЫЛДЫ ҮЙДІ БАСҚАРУ ҮШІН ТАРАТЫЛҒАН ВИРТУАЛДЫ ЖАДЫ ЖҮЙЕСІН ПАЙДАЛАНУ

# ИСПОЛЬЗОВАНИЕ СИСТЕМЫ С РАСПРЕДЕЛЕННОЙ ВИРТУАЛЬНОЙ ПАМЯТЬЮ ДЛЯ УПРАВЛЕНИЯ УМНЫМ ДОМОМ

***Аңдатпа.*** *Ақауларға төзімділік мәселесін шешу, «ақылды үй» бағдарламалық-аппараттық кешенінің ыңғайлылығы мен қауіпсіздігін арттыру үшін авторлар жүйенің әртүрлі түйіндерінде деректерді параллель өңдейтін кластер түріндегі SVM (Shared Virtual Memory) жүйесінің жеңілдетілген нұсқасын пайдалануды ұсынды. Бұл жұмыста өнеркәсіптік өндірілген Raspberry Pi 4 Model B құрылғыларына және LAN және Wi-Fi коммутациялық желісіне негізделген SVM жүйесін құру мысалы келтірілген, сонымен қатар алынған жүйенің ақауларға төзімділігін тексеру нәтижелері келтірілген. Бұл жұмыс жүйенің жұмысына талдау жасайды және кэштің жаңа түрін қосу арқылы өнімділікті жақсарту туралы шешім ұсынады.*

***Түйін сөздер:*** *SVM жүйесі, ақылды үй, ақауларға төзімділік, параллельді есептеу, SVM жүйесінің өнімділігін арттыру, жедел кэш, ақауларға төзімді кластер, Raspberry Pi..*

***Аннотация.*** *Для решения проблемы отказоустойчивости, повышения удобства и безопасности программно-аппаратного комплекса «Умный дом» авторы предложили использовать упрощенную версию SVM (Shared Virtual Memory) системы в виде кластера с параллельной обработкой данных на различных узлах системы. В данной работе приведен пример создания SVM системы на основе промышленно изготовляемых устройств Raspberry Pi 4 model B и коммутационной сети LAN и Wi-Fi, а также представлены результаты тестирования отказоустойчивости полученной системы. В данной работе приведен анализ производительности системы, и предложено решение по улучшению производительности за счет добавления нового вида кэша.*

***Ключевые слова:*** *SVM система, умный дом, отказоустойчивость, параллельные вычисления, повышение производительности SVM системы, оперативный кэш, отказоустойчивый кластер, Raspberry Pi.*

***Abstract.*** *To solve the problem of fault tolerance, increase the convenience and security of the Smart Home hardware and software complex, the authors proposed using a simplified version of the SVM (Shared Virtual Memory) system in the form of a cluster with parallel data processing on various nodes of the system. This paper provides an example of creating an SVM system based on industrially manufactured Raspberry Pi 4 model B devices and a LAN and Wi-Fi switching network, and also presents the results of testing the*

*fault tolerance of the resulting system. This paper provides an analysis of system performance, and a solution is proposed to improve performance by adding a new type of cache.*

***Keywords:*** *SVM system, Smart Home, fault tolerance, parallel computing, SVM system performance improvement, operational cache, fault-tolerant cluster, Raspberry Pi.*

*Introduction.* Due to the development of computer and information technology, people have started devoting more time to enhancing their personal spaces. Traditional ways of living and working have undergone significant changes due to information technologies. Nowadays, individuals actively incorporate Smart Home technologies into their lives. A "Smart Home" refers to an automated software and hardware complex that can control all connected household appliances and audio/video devices [1]. It optimizes life support procedures in modern buildings by combining essential components into a single unit. Users can remotely change or configure their home's internal equipment at any time using a remote control, wall panels, or a smartphone app.

The engineering of a cutting-edge Brilliant home includes a head unit and an exchanging climate that facilitates data exchange between the head unit, sensors, and actuators. This architecture offers advantages such as relatively low device prices and easy configuration and updates. However, a system with a centralized architecture has poor fault tolerance, which is a significant disadvantage. If the head unit fails or catches fire, the system ceases to function, thereby failing to alert occupants about a fire or other emergency situations that may jeopardize human life or well-being.

There are several traditional approaches still in use for solving Smart Home systems. The paper [2] discusses various architectures, technologies, and algorithms utilized in Smart Home energy management systems. It also evaluates the efficiency of current systems in use. Another article [3] focuses on the home's energy management systems in response to demand, covering aspects such as energy management, load management, and energy preservation. Additionally, studies [4] provide an overview of energy management systems in a smart home, including various optimization and management algorithms, as well as addressing security and data privacy. Moreover, IoT technologies are commonly employed to address Smart Home issues, as highlighted in article [5]. This article specifically examines the energy management system in a Smart Home and various aspects of its development and implementation, discussing control algorithms, optimization techniques, sensors, control devices, and monitoring systems.

Remote control methods utilizing Internet of Things technology are considered optimal and cost-effective for Smart Homes due to their inclusion. Numerous Smart Home applications used to control energy consumption are described in studies [6]. These applications cover topics such as lighting, temperature control, electrical appliances, and energy consumption. Another article [7] explores a novel approach to Smart Home management, incorporating a network protocol into the architecture of the Internet of Things. Additionally, work [8] presents innovative ideas for Smart Home management by altering energy consumption. For example, in article [9], the Arrowhead Framework core systems - Event Handler were utilized. They implemented the Event Handler system as a MQTT-enabled service broker and deployed the Node-RED data flow programming tool to connect diverse hardware devices and nodes, along with APIs for online services.

The main objective of this research is to design and develop a Smart Home system that continues to function even if one or more devices fail. One potential solution proposed in this paper is the development of a Smart Home based on a system with distributed virtual memory (Shared virtual Memory). This approach involves processing the same data on multiple devices simultaneously, allowing for timely alerts about any device failures in the system, as well as in emergency situations. It can also automatically call for emergency services such as ambulances

or firefighters. Another crucial aim of this research is to enhance the system's performance, resulting in reduced expenses related to purchasing necessary Smart Home equipment.

The scientific novelty of this article lies in the implementation of a Smart Home system based on fault-tolerant clusters and the introduction of a new caching mechanism in distributed virtual memory systems. Specifically, the focus is on caching memory pages in RAM within a Smart Home cluster. By integrating fault-tolerant clusters into the Smart Home infrastructure, the system becomes more resilient to failures, ensuring continuous operation and enhanced reliability. Furthermore, introducing memory page caching in distributed virtual memory systems optimizes data access, improving overall performance and reducing latency. This innovative approach demonstrates the potential for enhancing Smart Home management through advanced cluster-based architectures and caching techniques in distributed virtual memory systems.

The article is organized as follows: The requirements for Smart Homes are briefly discussed at the start of the section "Smart Home Management Requirements." The following section suggests creating a cluster-based SVM system for a Smart Home. The technical specifications of the proposed SVM system are described in the section "Smart Home architecture based on SVM system." The results of fault tolerance testing and one possible approach to improving the system's performance are discussed in the sections "Implementation of a Smart Home based on SVM systems and computational experiments" and "Improving the performance of the SVM system for a Smart Home." The conclusion and plans for future research are presented in the concluding section.

*Smart Home Management Requirements.* The design of a Smart Home is determined by the person or group for whom the solution is intended. The Smart Home system must prioritize meeting the needs of its users [10].

When designing a Smart Home, there are fundamental principles and requirements that must be followed, despite the variety of solutions:

Data transmission standards: Different devices with various capabilities and functions exist in a house, including furniture, home appliances, smart devices, and sensors. Integrating these devices requires standardized network data transmission and a seamless platform for their interaction, ensuring maximum functionality and comfort.

Apps for smart device management: The Smart Home system should support the addition of new devices and their accompanying apps. Existing applications may need updates for added security support and additional features. Tools for managing Smart Home applications should be available to easily add new devices and apps.

Dependable and secure functionality: Smart Home users may lack sufficient experience and knowledge to manage smart devices. Smart Home applications should operate reliably at all times, without delays or interruptions. These applications should ensure the reliability, security, and privacy of data, even without a head unit or manager.

User-centric design: Smart Home applications serve various purposes for users. These applications must fulfill the requirements and convenience of all user groups and deliver the expected outcomes through logical inference.

User-centered design: While there are numerous Smart Home apps available, users should be given priority. These applications should be designed with the user's needs in mind, meet their requirements, and provide a user-friendly interface for easy interaction.

One critical component of a Smart Home system is the head unit, which comes in a wide selection from different companies and manufacturers.

In a Smart Home system, the head unit serves as a computer that controls the entire system, connecting sensors, controllers, and control devices to each other. It processes data from various sensors and controls the devices in the house based on preset schedules and settings.

A major advantage of the head unit in a Smart Home system is the centralized control it provides over all devices in the house. This simplifies the management process and allows users to quickly configure and modify system parameters.

Additionally, the head unit plays a crucial role in the security of the Smart Home system. It can control system access and send notifications to mobile devices in the event of unforeseen events, such as hacking attempts or unauthorized access.

The head unit of a Smart Home system can be a standalone device or integrated into control devices like smartphones or tablets. In some cases, it can be implemented as a cloud service that manages the Smart Home system remotely.

Modern head units in Smart Home systems offer numerous functions and capabilities, enabling users to configure and control the system with precision. They provide a high level of automation and user-friendliness, enhancing comfort and safety in a Smart Home.

However, each model has its advantages and disadvantages, significantly impacting the overall reliability of the Smart Home system. For example, cloud-based systems aimed at cost-saving are often located on remote servers, causing modules to go offline when the internet connection is lost within the Smart Home network. On the other hand, using local head units such as those from Aqara, Xiaomi, Mijia, etc., ensures the Smart Home remains functional even without internet access. Nonetheless, risks exist in such situations, such as head unit malfunctions. Budget headsets, typically priced up to $300, often have a centralized control system where sensors and actuators connect to one central master head unit, with other head units as slave devices. In the event of a master head unit failure, the components of the Smart Home system go into offline mode. This limitation is addressed by more expensive headsets like those from Yandex and Xiaomi, which implement a protected cluster of multiple head units. The proposed solution in this article suggests implementing a cluster using three Raspberry Pi 4 Model B devices, each priced at $60.

*Designing a Smart Home based on an SVM system.* A wide variety of SVM systems are currently used in various industries, including personal computer work and scientific research conducted in computing centers. Despite the extensive range of applications (Figure 1), the overall appearance, operating principles, and limitations of SVM systems remain unchanged.

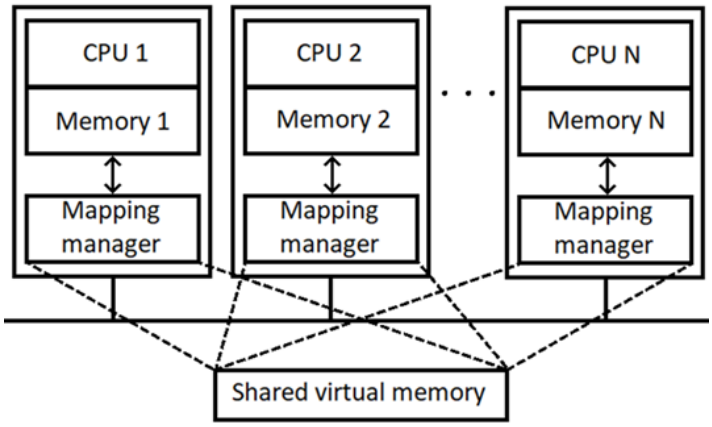



**Figure 1.** General view of the SVM system

As illustrated in Figure 1, the SVM system comprises multiple devices, each equipped with its own processor and memory. These devices are interconnected through a switching network, forming a unified computing complex.

The SVM system has several key limitations, including:

1) Each block of code is assigned to only one page.
2) The total length of blocks on a page does not exceed the page length.
3) The length of any working set does not exceed a predetermined system constant.

The functionality of Shared Virtual Memory involves multiple processors or computing processes sharing a single address space. Each processor can directly access any memory cell within this address space. The SVM System Manager is responsible for establishing the connection between local memory and the shared virtual memory address space. It also ensures the integrity of the address space, guaranteeing that the value returned by a read operation matches the value written by the most recent write operation at the same address.

There are several types of SVM systems:

1) Computer-computer: This type is commonly used in servers.
2) Processor-processor or core-core: Also prevalent in server applications.
3) Graphics processor-graphics processor: Primarily used for working with video data and neural networks.
4) Processor-graphics processor: Also utilized in video data processing and neural network applications.

To enhance the reliability and security of a Smart Home, this article proposes implementing an SVM system based on computer-computer communication. This type is often referred to as "fault-tolerant clusters" or "high-availability clusters." A key feature of this type is the automatic transfer of functions to other nodes within the cluster in the event of a node failure (known as failover). Additionally, cluster roles are proactively monitored to ensure their proper operation, with rebooting or migration to another node performed if necessary.

The home network is a crucial aspect of any Smart Home system. It facilitates communication between devices and contributes to energy management and household consumption reduction [11]. The home network can be either wired or wireless, as depicted in Figure 2. Wireless setups offer easier installation and maintenance compared to traditional wired configurations. They also provide improved scalability. However, wireless networks have their downsides, such as lower bandwidth, reduced security levels, and susceptibility to interference.

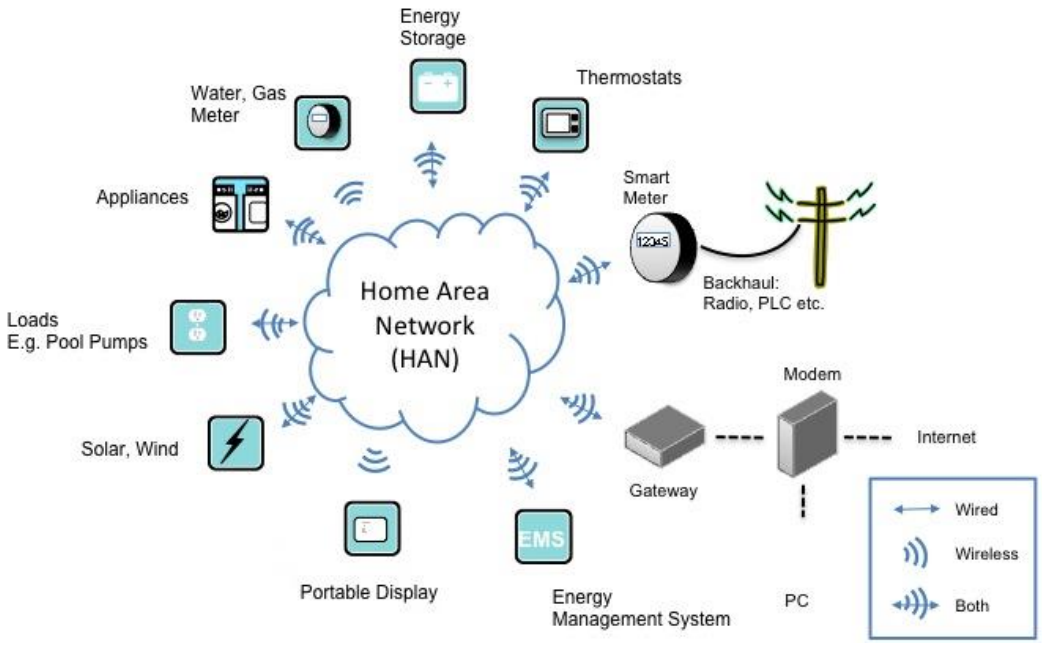



**Figure 2.** Home network [12]

It is important to highlight that for the implementation of an SVM system in a Smart Home, a high data transfer rate between most nodes is not necessary. This is because sensor data is not transmitted frequently, making little difference between using a wired or wireless network. Additionally, the SVM system offers a "terminal server" functionality, allowing users to manage their Smart Home, continue working, or watch movies on any connected terminal, regardless of whether the terminal is directly connected to the home network or accessed through a mobile application. Further details about this feature will be provided in the following module.

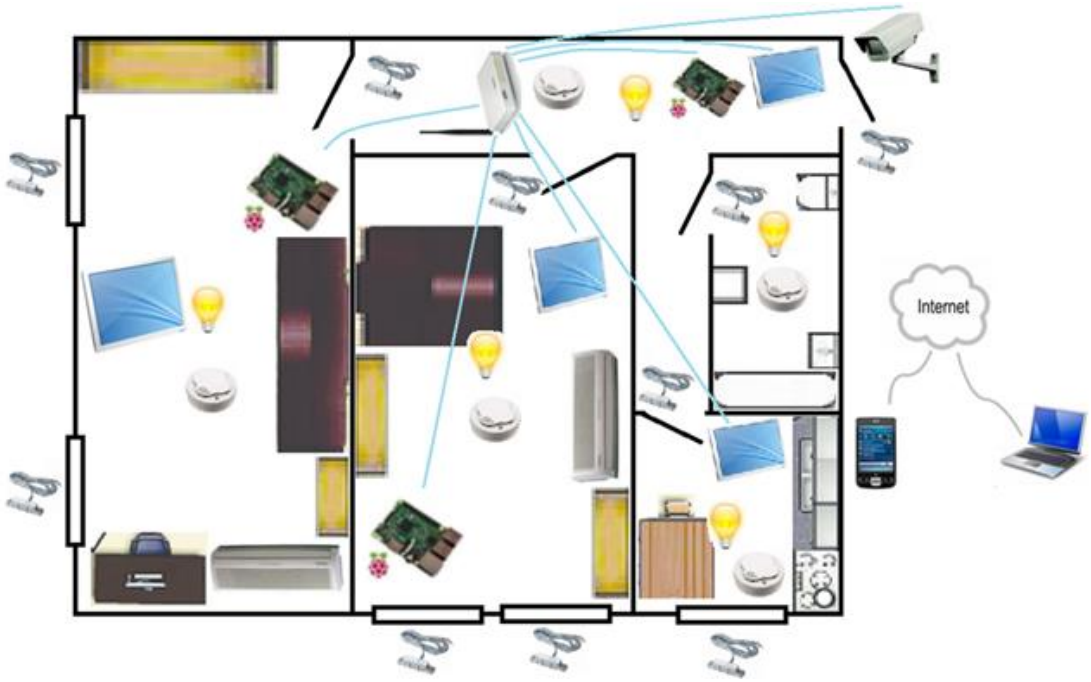


**Figure 3.** Smart Home Architecture

*Smart Home architecture based on SVM system.* Figure 3 depicts the architectural diagram of a Smart Home that utilizes an SVM system. The head unit of the Smart Home system is based on the Raspberry Pi 4 Model B, which possesses the following specifications:

1) A 4-core, 64-bit Cortex-A72 processor (ARM v8) with a clock speed of 1.5 GHz.
2) 4 GB of RAM.
3) A Gigabit network port.

For the switch, an 8-port TP-Link TL-SG1008MP is employed. This switch is connected to three Smart TVs, three Raspberry Pi 4 Model Bs, an outdoor surveillance camera, and an Asus RT-AX82U Wi-Fi router. These devices can establish wireless connections with sensors and actuators, allowing them to communicate with the Smart Home system via the Internet. The gauss Smart Home 1170112 lamps, connected through Wi-Fi to the SVM system, provide illumination for each room. The system is also connected to sensors such as the Ps-Link WIFI-818 wireless gas sensors, the PS-WD001 wireless opening sensors, the Rovex RS-07CST4 wireless on/off air conditioners, and Raspberry Pi 4 Model B devices with USB Wi-Fi and GSM receivers for emergency notifications to firefighters and ambulances. Additionally, actuators for automatic window control are incorporated. The system is built on the foundation of Raspberry Pi 4 Model B devices with USB Wi-Fi modules, as depicted in Figure 4.

**Figure 4.** Actuator for automatic window control

The foundation of the SVM system for a Smart Home is based on Classic Swarm [13], which runs on the Raspberry Pi OS Lite operating system. This Classic Swarm has been uniquely modified to organize a cluster SVM system, as illustrated in Figure 5. The cluster forming the SVM system consists of three Raspberry Pi devices connected through a fault-tolerant cluster scheme, serving as the head unit for the Smart Home. Additionally, six additional Raspberry Pi devices are connected to this cluster using a computing cluster scheme, and they are situated within the actuators and GSM receiver. This plan combines a fault-tolerant cluster with a computing cluster, resulting in increased system reliability and enhanced performance of the SVM system. Consequently, the implementation of a terminal server for the Smart Home becomes possible. By installing the Docker Engine version 19.03 application on the 32-bit Raspberry Pi OS Lite operating system, each Raspberry Pi node is transformed into a Docker Machine.

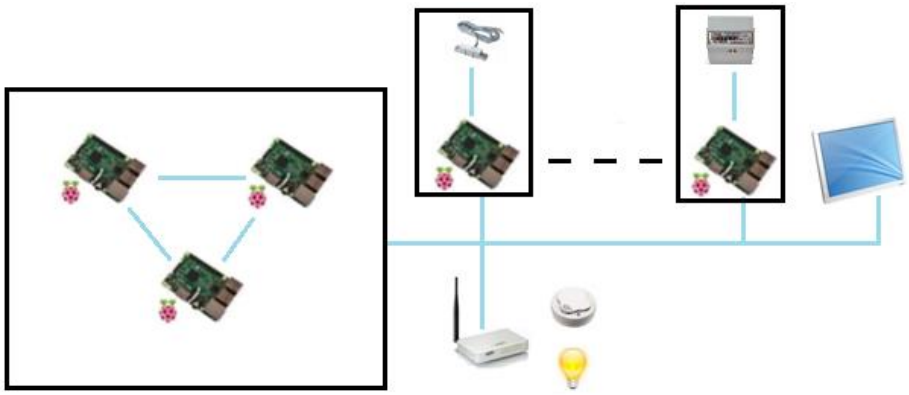


**Figure 5.** SVM System Connection Diagram for Smart Home

*Implementation of a Smart Home based on SVM systems and computational experiments.* System performance is evaluated based on throughput and response delay parameters generated by the system during load testing. The request load is simulated using Apache JMeter application [14] version 5.5, executed from a client system consisting of a desktop PC equipped with a Ryzen 9 5950X processor (4.8 GHz clock speed), 32 GB of RAM, and running on the MS Windows 10 64-bit operating system. To facilitate monitoring of cluster behavior, a visual image is utilized - the Visualizer container [15]. This container is launched on the cluster system manager node, functioning as a web service accessible through a browser on the client machine.

Below are some of the completed test scenarios.

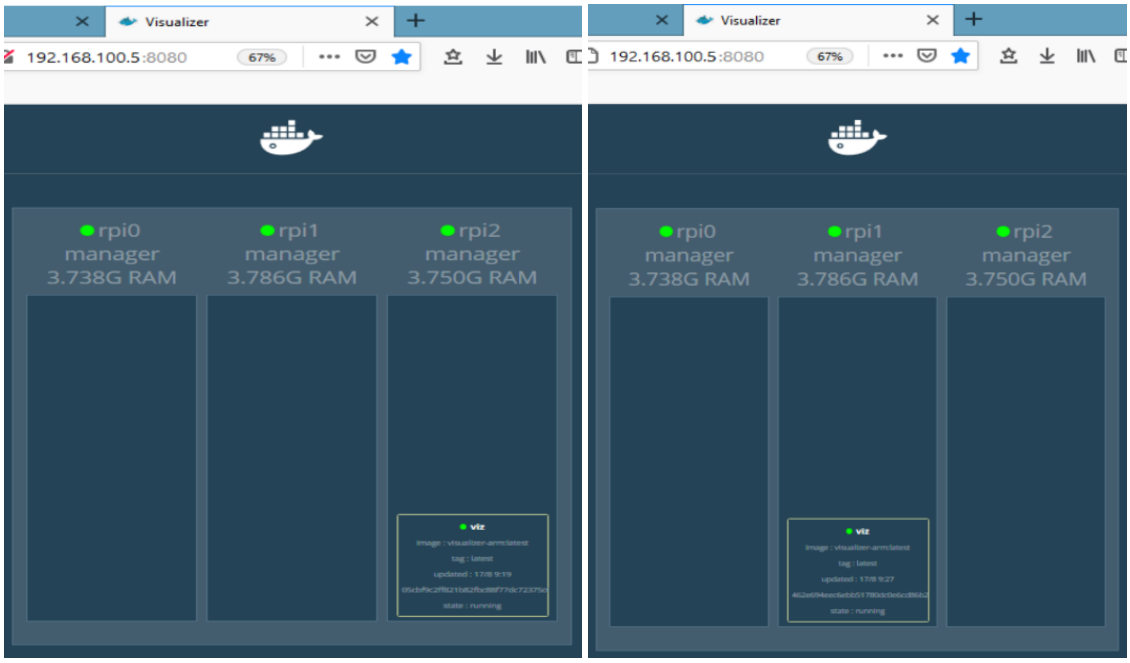



**Figure 6.** Cluster visualization: (a) initial state; (b) rpi2 reboot status

1. Fault tolerance testing

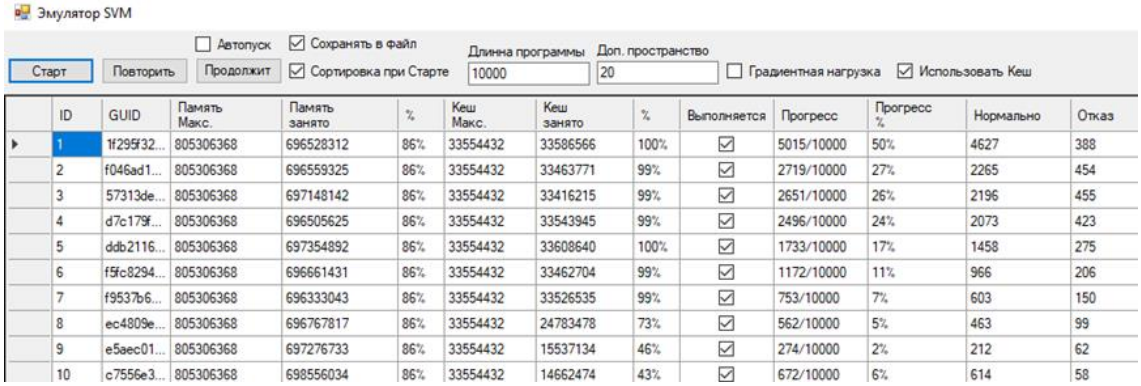
First and foremost, a fault tolerance test was conducted to assess the system's reliability in

handling failures caused by resource unavailability. For this experiment, simply disconnecting the power supply or rebooting the device is sufficient. The purpose of this test is to monitor and measure the system's availability by creating a scenario where one or more machines become unavailable. We attempt to access the Visualizer application service using the same address to verify if the cluster system can still run the Visualizer application service (Figure 6).

In this experiment, the initial state of the visualizer service was set on the RPI2 server. Next, we rebooted the RPI2 server and attempted to access the Visualizer application at the same address. As shown in Figure 6, the Visualizer application service is still running, but it has been automatically moved to the RPI1 server. Therefore, the test confirmed that failover in this cluster system occurs quickly and without downtime. This feature is essential in systems with high fault tolerance.

2. Execution testing

Framework performance testing involves gathering data from various components of the system within a short period of time. The objective of this test is to monitor and measure the availability of SVM framework pages by tracking the number and duration of page failures. This scenario assesses how the system handles consecutive requests. To conduct this test, we have developed a program that generates a significant load on the entire SVM framework in a Smart Home by sending a large number of data page requests to each component of the system. Additionally, this program records the execution time of each individual request. The test program for the SVM framework is designed as a Windows application, as depicted in Figure 7.




| | ID | GUID | Память Макс. | Память занято | % | Кеш Макс. | Кеш занято | % | Выполняется | Прогресс | Прогресс % | Нормально | Отказ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ▸ | 1 | 1f295f32... | 805306368 | 696528312 | 86% | 33554432 | 33586566 | 100% | ☑ | 5015/10000 | 50% | 4627 | 388 |
| | 2 | f046ad1... | 805306368 | 696559325 | 86% | 33554432 | 33463771 | 99% | ☑ | 2719/10000 | 27% | 2265 | 454 |
| | 3 | 57313de... | 805306368 | 697148142 | 86% | 33554432 | 33416215 | 99% | ☑ | 2651/10000 | 26% | 2196 | 455 |
| | 4 | d7c179f... | 805306368 | 696505625 | 86% | 33554432 | 33543945 | 99% | ☑ | 2496/10000 | 24% | 2073 | 423 |
| | 5 | ddb2116... | 805306368 | 697354892 | 86% | 33554432 | 33608640 | 100% | ☑ | 1733/10000 | 17% | 1458 | 275 |
| | 6 | f5fc8294... | 805306368 | 696661431 | 86% | 33554432 | 33462704 | 99% | ☑ | 1172/10000 | 11% | 966 | 206 |
| | 7 | f9537b6... | 805306368 | 696333043 | 86% | 33554432 | 33526535 | 99% | ☑ | 753/10000 | 7% | 603 | 150 |
| | 8 | ec4809e... | 805306368 | 696767817 | 86% | 33554432 | 24783478 | 73% | ☑ | 562/10000 | 5% | 463 | 99 |
| | 9 | e5aec01... | 805306368 | 697276733 | 86% | 33554432 | 15537134 | 46% | ☑ | 274/10000 | 2% | 212 | 62 |
| | 10 | c7556e3... | 805306368 | 698556034 | 86% | 33554432 | 14662474 | 43% | ☑ | 672/10000 | 6% | 614 | 58 |

**Figure 7.** A test program for evaluating system performance

This program connects to the cluster over the network to exchange memory pages. It serves as an external manager for the SVM system. This program's primary objective is to organize, control, and monitor the entire SVM system. Notwithstanding the abovementioned, this program likewise makes a heap on the SVM framework as an enormous number of solicitations for pages. There are a lot of settings and functions in this program. For instance:

– Program duration: Decides the quantity of irregular virtual memory locations to get pages from the SVM framework.

– More floor space: This is the additional space that can be added by this program to the SVM system in order to simulate the computational process.

– Sorting loads: allows you to display the system's most loaded node.

– Loading gradient: responsible for adjusting additional load on a particular system node selectively.

– Export to File: The job of this function is to save interim totals to files for later analysis.

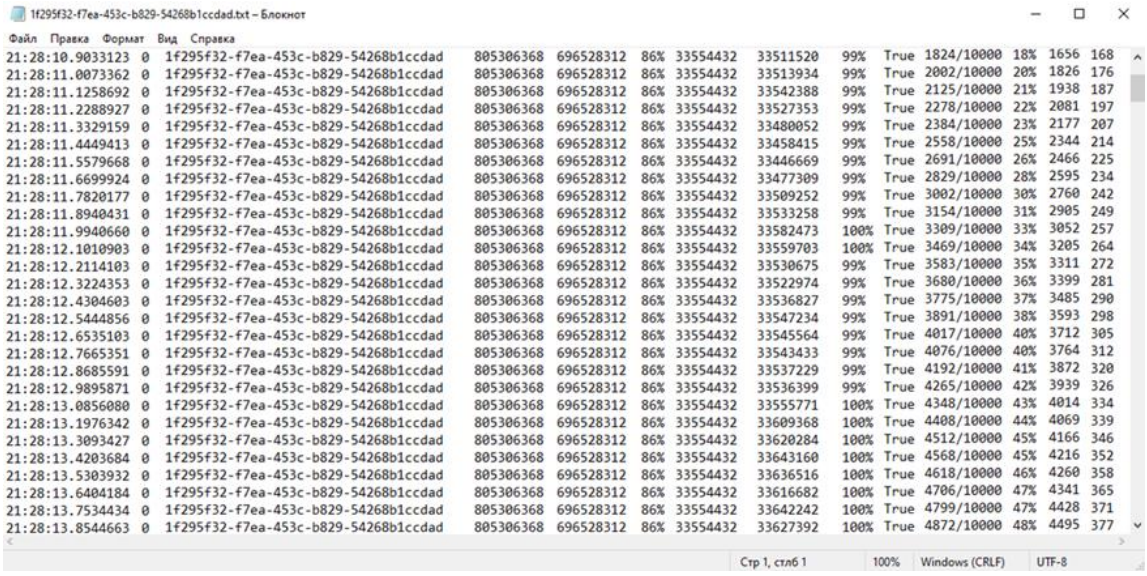
Figure 7 shows how this program displays computing process information in real time and

saves data when the "save to file" option is selected (Figure 8). Each process's output data are stored separately in files that can later be analyzed with third-party software like Microsoft Excel.

3. Smart Home management

To enhance the usability of the Smart Home system and enable access from mobile devices [16], an LTSP terminal server was incorporated into the SVM system. With LTSP, local network computers can boot from a single server, reducing equipment costs and minimizing maintenance time. Since most processes are executed on the server, clients do not require powerful processors or video cards. Servicing only the server instead of multiple local network computers decreases service time, and software installations and updates only need to be performed once. The decreasing cost of hardware further contributes to the time-saving benefits of LTSP [17].

LTSP also offers advantages in administration and communication. Features like screen/desktop sharing and centralized user accounts allow users to access their accounts from any terminal. Furthermore, LTSP is widely used and compatible across different platforms. This strategy enables access to the Smart Home system via thin-client technology from Android-powered smartphones, tablets, and personal computers. The aTMC application was utilized to connect Android devices to the Smart Home SVM system, while the VNC Viewer 6.22.315 64-bit program was used for connections from MS Windows OS. An example illustrating the functionality of the terminal server in this system is watching movies. If a movie is started on one device, such as a TV in one room, it can be seamlessly continued on another device, like a tablet in the kitchen. Users simply need to turn on the device and log in as their user, similar to any operating system.


1f295f32-f7ea-453c-b829-54268b1ccdad.txt – Блокнот

Файл Правка Формат Вид Справка

```
21:28:10.9033123 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33511520 99% True 1824/10000 18% 1656 168
21:28:11.0073362 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33513934 99% True 2002/10000 20% 1826 176
21:28:11.1258692 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33542388 99% True 2125/10000 21% 1938 187
21:28:11.2288927 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33527353 99% True 2278/10000 22% 2081 197
21:28:11.3329159 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33480052 99% True 2384/10000 23% 2177 207
21:28:11.4449413 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33458415 99% True 2558/10000 25% 2344 214
21:28:11.5579668 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33446669 99% True 2691/10000 26% 2466 225
21:28:11.6699924 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33477309 99% True 2829/10000 28% 2595 234
21:28:11.7820177 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33509252 99% True 3002/10000 30% 2760 242
21:28:11.8940431 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33533258 99% True 3154/10000 31% 2905 249
21:28:11.9940660 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33582473 100% True 3309/10000 33% 3052 257
21:28:12.1010903 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33559703 100% True 3469/10000 34% 3205 264
21:28:12.2114103 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33530675 99% True 3583/10000 35% 3311 272
21:28:12.3224353 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33522974 99% True 3680/10000 36% 3399 281
21:28:12.4304603 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33536827 99% True 3775/10000 37% 3485 290
21:28:12.5444856 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33547234 99% True 3891/10000 38% 3593 298
21:28:12.6535103 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33545564 99% True 4017/10000 40% 3712 305
21:28:12.7665351 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33543433 99% True 4076/10000 40% 3764 312
21:28:12.8685591 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33537229 99% True 4192/10000 41% 3872 320
21:28:12.9895871 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33536399 99% True 4265/10000 42% 3939 326
21:28:13.0856080 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33555771 100% True 4348/10000 43% 4014 334
21:28:13.1976342 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33609368 100% True 4408/10000 44% 4069 339
21:28:13.3093427 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33620284 100% True 4512/10000 45% 4166 346
21:28:13.4203684 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33643160 100% True 4568/10000 45% 4216 352
21:28:13.5303932 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33636516 100% True 4618/10000 46% 4260 358
21:28:13.6404184 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33616682 100% True 4706/10000 47% 4341 365
21:28:13.7534434 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33642242 100% True 4799/10000 47% 4428 371
21:28:13.8544663 0 1f295f32-f7ea-453c-b829-54268b1ccdad 805306368 696528312 86% 33554432 33627392 100% True 4872/10000 48% 4495 377
```

Стр 1, стлб 1 100% Windows (CRLF) UTF-8

**Figure 8.** Calculation result

As a result, the Smart Home system becomes accessible from any preferred device, allowing uninterrupted work without the need to start over.

*Improving the performance of the SVM system for a Smart Home.* SVM systems are typically implemented using particular server technologies or emulated using FPGAs, as shown in [18]. However, not the most powerful single-board computers, the Raspberry Pi 4 model B, were used as a foundation for the SVM system for a Smart Home in order to maximize energy savings and reduce costs. As a result of its small size and poor performance, the SVM system for a Smart

Home frequently began to function but then froze. Using the testing program we developed, which can be seen in Figure 7, a number of tests were carried out to determine the causes of freezes in the SVM system. We were able to create a schedule of delays in receiving pages in the SVM system and observe the response times of each system device thanks to this program. In this experiment, data page addresses were selected based on the worst-case access scenario, namely accessing a memory address randomly. Figure 9a depicts a 32-second delay graph of the entire system with 10 Raspberry Pi, while Figure 9b depicts a 4-second delay graph of just one Raspberry Pi device that is distinct from the overall graph in Figure 9a.

Figure 9 illustrates that the system exhibited a maximum response time of 68 milliseconds and an average response time of 15 milliseconds. To address the occurrence of system freezes, an additional form of caching called "Operational Cache" was implemented on each system node. The size of this cache was set to 5% of the RAM. Instead of swapping pages in the SVM system, the Operational Cache serves as a software-based page caching mechanism, allowing pages to be cached in local memory for future read operations. These cached pages are marked as "read-only." If there is a need to write data to a cached page, the requesting device sends a request to invalidate the page on other nodes. As a result, the SVM system ensures that only one instance of the page exists, satisfying the requirements for page editing.

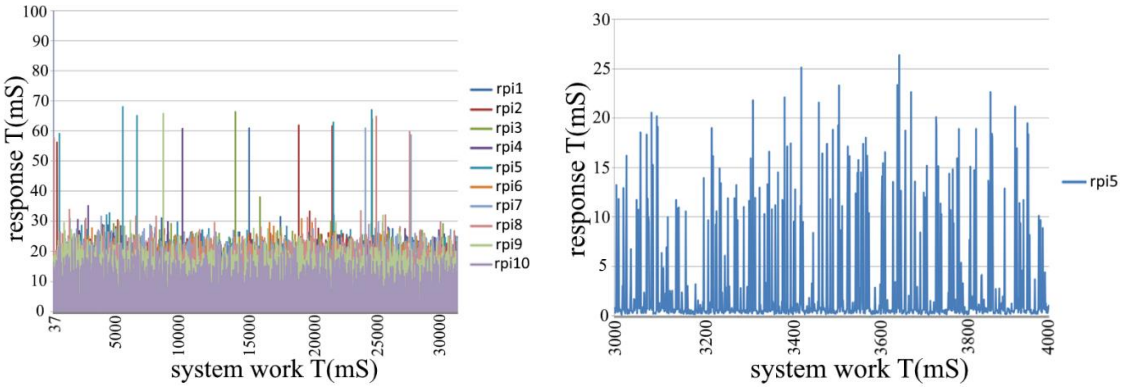



**Figure 9.** Delay schedule: (a) the delay schedule of the entire system in 32 seconds; (b) the delay schedule of rpi5 in 4 seconds

It is important to note that complete elimination of page failures is nearly impossible. However, this strategy has significantly reduced both the frequency and duration of page failures in the system. This is attributed to the fact that the Smart Home system does not involve a large number of mathematical calculations that constantly access the pages of the SVM system. After the incorporation of the Operational Cache, the delay schedule of the entire system is presented in Figure 10a for a duration of 32 seconds. Figure 10b shows the delay schedule of an individual Raspberry Pi, represented separately from the overall graph in Figure 10, a.

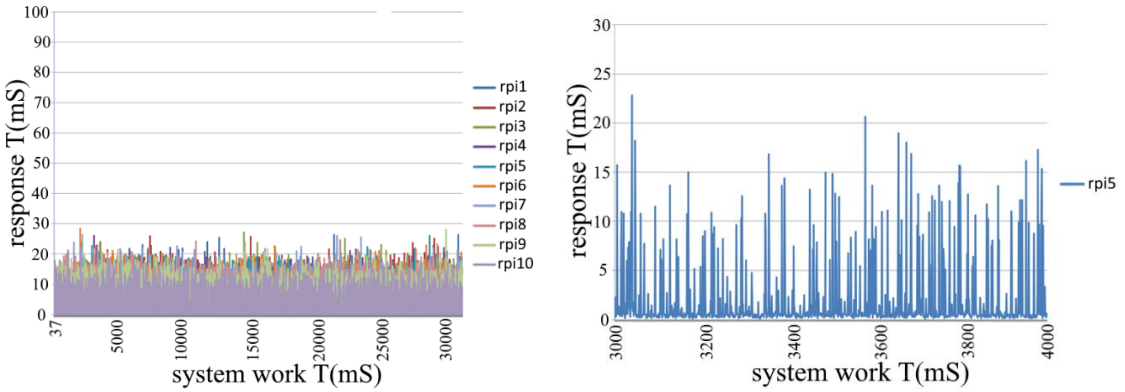



**Figure 10.** Schedule of delays with an "Operational cache":
(a) schedule of delays of the entire system in 32 seconds; (b) schedule of delays of rpi5 in 4 seconds

As depicted in Figure 10, the implementation of the "Operational Cache" resulted in a decrease in the average page failure time from 15 milliseconds to 12 milliseconds. Furthermore, the occurrence of very long page failures that previously took up to 65 milliseconds was completely eliminated. This improvement in the system's performance led to an overall boost of up to 20%.

In conclusion, this paper introduces a Smart Home system based on a distributed virtual memory SVM system that can maintain functionality during emergency situations even when one or more devices, including the Smart Home's head unit, become disabled. The paper also presents a method for integrating various commercially available devices into a unified SVM-based computing system for Smart Home management. By utilizing "Thin Client" technology, this approach enhances the convenience of using a Smart Home. Additionally, the paper discusses how increasing the amount of RAM utilized by the SVM system's components can result in improved performance for a Smart Home. Experimental tests involving data block and page transfers between devices support this concept. The results demonstrate a reduction in the average page failure time from 15 milliseconds to 12 milliseconds, with the elimination of very long page failures that previously took up to 65 milliseconds. Consequently, the system's performance was enhanced by more than 20%.

Future plans involve conducting experiments with a neural network as a caching algorithm, as it is anticipated to further reduce the occurrence of page failures by accurately predicting page requests.

References

1. Marlon Buchanan. The Smart Home Manual: How to Automate Your Home to Keep Your Family Entertained, Comfortable, and Safe (Home Technology Manuals). HomeTechHacker. 2020
2. M. Devi, Muralidharan S., Elakiya R., Monica M.. (2023) Design and Implementation of a Smart Home Energy Management System Using IoT and Machine Learning. International Conference on Smart Engineering for Renewable Energy Technologies (ICSERET-2023). 10.1051/e3sconf/202338704005
3. Shareef, Hussain & Ahmed, Maytham & Mohamed, Azah & Al Hassan, Eslam. (2018). Review on Home Energy Management System Considering Demand Responses, Smart Technologies, and Intelligent Controllers. IEEE Access. PP. 1-1. 10.1109/ACCESS.2018.2831917.
4. Kamran Taghizad-Tavana, Mohsen Ghanbari-Ghalehjoughi, Nazila Razzaghi-Asl, Sayyad Nojavan, As'ad Alizadeh. (2022) An Overview of the Architecture of Home Energy Management System as Microgrids, Automation Systems, Communication Protocols, Security, and Cyber Challenges. Renewable Energy Technologies and Microgrids. 10.3390/su142315938.
5. Stavros Mischos, Eleanna Dalagdi, Dimitrios Vrakas. (2023) Intelligent energy management systems: a review. Artificial Intelligence Review. 10.1007/s10462-023-10441-3
6. Marco Esposito, Alberto Belli, Lorenzo Palma, Paola Pierleoni. Design and Implementation of a

Framework for Smart Home Automation Based on Cellular IoT, MQTT, and Serverless Functions. Internet of Things for Smart Homes III. Volume 23, Issue 9, 2023. 10.3390/s23094459.
7. Mendes, T.D.P. & Godina, Radu & Rodrigues, Eduardo & Matias, João & Catalão, João. (2015). Smart Home Communication Technologies and Applications: Wireless Protocol Assessment for Home Area Network Resources. Energies. 8. 7279-7311. 10.3390/en8077279
8. Bin Zhou, Wentao Li, Ka Wing Chan, Yijia Cao, Yonghong Kuang, Xi Liu, Xiong Wang, Smart Home energy management systems: Concept, configurations, and scheduling strategies, Renewable and Sustainable Energy Reviews, Volume 61, 2016, Pages 30-40, ISSN 1364-0321, https://doi.org/10.1016/j.rser.2016.03.047.
9. A. Zabasta, N. Kunicina, K. Kondratjevs, A. Patlins, L. Ribickis and J. Delsing, "MQTT Service Broker for Enabling the Interoperability of Smart City Systems," 2018 Energy and Sustainability for Small Developing Economies (ES2DE), Funchal, Portugal, 2018, pp. 1-6, doi: 10.1109/ES2DE.2018.8494341.
10. Kim MJ, Cho ME and Jun HJ (2020). Developing Design Solutions for Smart Homes Through User-Centered Scenarios. Front. Psychol. 11:335. doi: 10.3389/fpsyg.2020.00335
11. M. Usman Saleem , Mustafa Shakir, M. Rehan Usman, M. Hamza Tahir Bajwa, Noman Shabbir, Payam Shams Ghahfarokhi, Kamran Daniel. (2023) Integrating Smart Energy Management System with Internet of Things and Cloud Computing for Efficient Demand Side Management in Smart Grids. Energies. Volume 16. doi: 10.3390/en16124835
12. Rick Lehrbaum, “ZigBee-certified software supports Smart Grid devices” [Online]. Available: https://linuxgizmos.com/zigbee-support-for-smart-grid-apps/. [Accessed: 09- july-2023].
13. Justincormack,et.all, “GitHub - docker-archive/classicswarm: Classic Swarm: a Docker-native clustering system.” [Online]. Available: https://github.com/docker-archive/classicswarm. [Accessed: 20-May-2022].
14. Apache, “Apache JMeter - Apache JMeter TM,” Apache Jm., p. 2019, 2014.
15. Miranda,et.all, “GitHub - dockersamples/docker-swarm-visualizer: A visualizer for Docker Swarm Mode using the Docker Remote API, Node. JS, and D3.” [Online]. Available: https://github.com/dockersamples/docker-swarm-visualizer. [Accessed: 20-May-2022].
16. alkisg,et.all, “GitHub - ltsp/ltsp: Linux Terminal Server Project.” [Online]. Available: https://github.com/ltsp/ltsp. [Accessed: 20-May-2022].
17. Dashamir Hoxha, “LTSP Use Cases”, Researchgate, December 2018.
18. Vyazigin, S., Dyusembaev, A., Mansurova, M.: Emulation of x86 Computer on FPGA, 2020 IEEE 8th Workshop on Advances in Information, Electronic and Electrical Engi-neering, AIEEE 2020. Latvia, Riga, Riga Technical University (2021).