IoT Devices Operating Systems Unveiled: An Analysis and Comparison of Operating System for Internet of Things

Maleeha Kanwal[1], Maryam Wajeeha[2], Nageen Khan[3]
[1,2,3]Department of Computer Science, University of Engineering and Technology, Lahore, Pakistan
[1]maleehakanwal582@gmail.com, [2]maryamwajeeha05@gmail.com, [3]knageen61@gmail.com

***Abstract*– The Internet of Things (IoT) is described by heterogeneous devices. For research objectives, the Internet of Things (IoT) presents many difficulties. IoT devices operating system are beneficial for this purpose. The low-end IoT devices are not reliable for outdated operating systems. A lot of effort is required to design operating systems for concerned devices. This paper compares operating systems for low-end IoT devices and examines essential characteristics in the majority of current IoT operating systems based on different resource management attributes. The comparison will focus on operating systems that are best for low-end devices on behalf of Architecture, Programming model, Scalability, Network performance, Energy Consumption and Scheduling. Operating systems that we will discuss in this paper are Contiki, TinyOS, LiteOS and freeRTOS, Zephyr, Tizen, UbuntuCore, OpenWSN etc. This paper can be beneficial for researchers interested in this field. It can provide an overview of the available IoT operating systems, their features, advantages, and limitations, as well as this paper can also help researchers identify gaps in the existing literature.**

***Keywords*– Low Ended Devices, Operating System, Tizen IoT, Zephyr and Comparison**

I. INTRODUCTION

THE Internet of Things is a network of smart devices that communicate and share information with one another via the internet. Implanted software, cameras, sensors, and actuators that can sense light, solids, separation, and development can all be found in the IoT environment [5]. With the development advancements we are quickly moving towards innovative time, where we discover the keen planet, brilliant urban areas, and shrewd homes all are outfitted with canny IoT devices fit for performing numerous assignments without anyone else. IoT devices are divided into two categories [12].

1- high ended devices
2- low ended devices

High ended devices such as smartphones while low ended devices such as regular operating system like Linux BSD (Berkeley Software Distribution).

In general, energy and RAM resources are limited in IoT devices. They are typically small and battery-operated, with a memory requirement of 100 kilobytes. These devices typically have 8-piece microcontrollers, which are no longer supported by modern Windows/Unix/Mac-based workstations and PCs. These unmistakable IoT features and requirements require a capable, adaptable, practical, and lightweight framework with minimal RAM and ROM impressions. For example, Linux, Windows 8.1, ARM, Arrayant, and IFTTT. The competition to structure IoT OS is furious [6].

These clever devices that truly interact with the physical world, such as by controlling motors or detecting the temperature, form one end of the Internet of Things (IoT). Amazing servers that serve as the backend, for example by providing a web interface for administration or a database to hold sensor data make the opposite end. However, the internet of things also poses a number of brand-new challenges for the network protocols and programming styles needed to operate on and among these smart devices [7].

IoT devices depend on the remote sensors, it's applications, which makes customary working framework unimportant due to IoT's low assets and calculation power, in such circumstance improvement of the lightweight working framework was important to fulfill asset limitation need of web of things (IoT).There are different OS for IOT environments are Contiki, RIOT, LiteOS,TinyOS, freeRTOS, and Mbed [7].

In this paper, we will study about different operating systems (Contiki, TinyOS, LiteOS and freeRTOS) for IoT on bases of their architecture, programming model and hardware support, etc. Fig. 1 describe the different types of IoT operating systems.

The organization of this paper is as follows. Section I describe introduction; section II is all about literature review. Section III is about comparison of IoT OS Section IV describe conclusion.

II. LITERATURE REVIEW

The operating system is an interconnection between the hardware and the user. The operating system provides us programming interfaces and manages all the timing of processes. IoT devices work in asset obliged conditions, and to deal with these simultaneous applications, a reasonable execution model must be given by OS. The execution model must give memory productivity to undertakings.

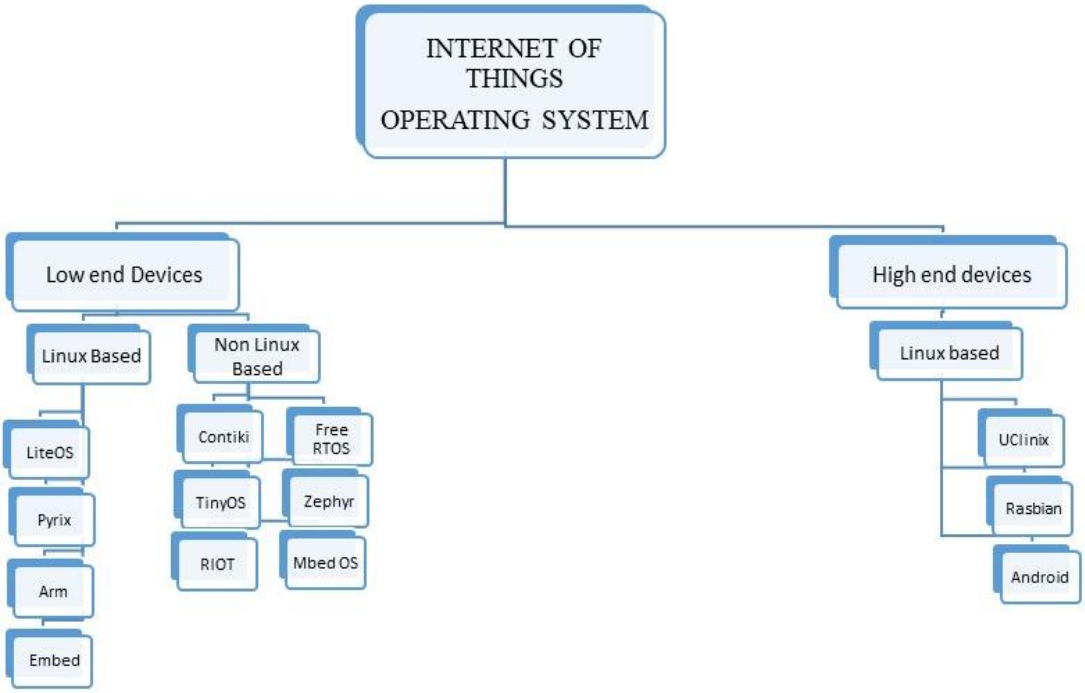

Fig. 1. Internet of things operating system Categories

Padmini Gaur, Mohit P. Tahiliani describe that the paper would assist specialists with understanding the Internet of Things, their character-qualities, and the methodologies received by OSes to deal with the shrewd constant IoT frameworks. Further, we have introduced a nonexclusive model for IoT working framework which may permit one to pick the best OS as indicated by their necessities [1].

Challouf Sabri, Kriaa Lobna, Saidane Leila Azzouz primary contribution is a comparison of the most current operating systems for low-end IoT devices. The comparison will concentrate on the main elements of the operating system, including architecture, scheduling, real-time capabilities, programming model, memory footprint, energy efficiency, hardware support, and programming model [2].

Muhammad Asim, Waseem Iqbal discuss discuss internet of things operating systems, current security issues in IoT, as well as potential solutions to these issues using RPL and 6LoWPAN (IPv6 over low-power WPAN) protocols [3].

Emmanuel Baccelli, Cenk Gundo gan, Oliver Hahm explained the initial thorough analysis of RIOT. The kernel, hardware abstraction, and software modularity are the three main topics of discussion for aspiring developers and users, and they are covered both theoretically and practically for a variety of example configurations. They explain operational features like network usage, battery management, timers, and system boot-up. The relevant APIs exposed by the operating system are discussed in the final section, along with RIOT's broader ecosystem, including its development and open-source community [4].

Arslan Musaddiq, Yousaf Bin Zikria, Oliver Hahm surveyed resource management of different IoT OS. They made an effort to show hidden patterns/features of several recommended approaches regarding the IoT OS Resource Management research. Give an in-depth insight into every Resource Management view of Contiki, FreeRTOS and TinyOS and their first approach, underlying idea, advantages, and limitations [7].

Farhana Javed examined the fundamental conditions for Internet of Things apps, including a suitable architecture, suggested algorithms, language support, and power and memory management. They claimed that among the IoT OS that have already been suggested, some use the most recent networking technologies and are real-time, while others do not. According to their investigation they found that Contiki and RIOT meet almost all of the criteria for an IoT OS, making them the most popular OS for IoT applications.OS should possess real-time capabilities and integrate 6loWPAN protocols for communication [11].

III. IOT OS COMPARISON

In this paper, the most used low ended devices are discussed. These Operating systems are Contiki, TinyOS, and freeRTOS. In this session, we will compare these OS on behalf of the programming model, architecture, and scheduling.

First, we will describe these operating systems for your understanding.

- ***FreeRTOS***

FreeRTOS is a class of RTOS (real-time operating system) that is intended to be sufficiently little to run on a microcontroller in spite of the fact that its utilization isn't constrained to microcontroller applications. Real-time Operating System, or an RTOS, is a product segment that switches between tasks rapidly to give the impression that several projects are being carried out concurrently on a single processing center [8].

- ***TinyOS***

TinyOS is an Open source developed by TinyOS Alliance for wireless sensor networks written in under the BSD license. SDK for TinyOS is a combination of Tiny DT and Eclipse Editor Plugin. TinyOS supports Multi-Path Routing, Geographical Routing, Routing Reliability-based, Broadcast based Routing and TDMA (time division multiplexing access) base Routing. Tiny Sec made TinyOS architecture Secure[9].

- ***Zephyr***

Zephyr is a real-time operating system, was created for IoT devices with limited resources. It is highly modular and adaptable to various devices' unique requirements.

In general, the creation of specialized operating systems for IoT gadgets is a crucial development in the advancement of this technology. IoT devices can function more effectively and efficiently thanks to these operating systems' highly effective and scalable design. We can anticipate more invention and advancement in this field as the number of IoT devices increases [28].

- ***Contiki OS***

Adam Dunkels founded Contiki in 2002. Contiki is an open-source operating system for memory-required frameworks and networking access, with a focus on low-power remote Internet of Things devices. Contiki is still being used for alerts, radiation monitoring, sound observing for bright cities, and road illumination frameworks. It is open-source code released under a BSD licence [9].

ContikiMAC is the name of the standard method for accomplishing the radio's low-power activity. With this MAC convention, hubs have the choice to receive and pass along radio messages even when they are operating in low-power mode. It utilizes an effective wake-up component to achieve powerful productivity: with a wake-up recurrence of 8 Hz, the inert radio obligation cycle is just 0.6% [29].

- ***LiteOS***

Like Contiki, Tiny OS, RIOT, LiteOS is an open source similar to Unix OS. Its development environment is UNIX based. LiteOS includes three Major components LiteFS, LiteShell, and the kernel. LiteOS has an Event Tracing mechanism to provide in depth knowledge of system. A buffer exists in processing applications to record initiated events. It has an advanced mechanism to achieve memory security that when the buffer runs out of space, it gets emptied in an external flash file. It supports plug and play routing stack. Various applications Like smart homes and smart cities using LiteOS [12].

Now we will compare them on behalf of some features:

- ***General***

Although they are all open-source operating systems, none of them have the same licence. FreeRTOS has an additional GPL license. TinyOS and Contiki have a BSD license. Contiki OS has a growth rate of 74%. FreeRTOS has a 63% growth rate between 2015 and 2017. These two are more effective than others.

- ***Architecture***

We have compared the operating systems on the bases of their architecture. The microkernel is a critical component of most operating systems due to its small size and high volume of context changes. However, the kernel architecture determines real-time capabilities. That's why FreeRTOS is very useful for real-time working because it has microkernel OS architecture.

- ***Scheduling***

A key element in the system's performance is the scheduling technique. Some IoT frameworks may have exacting real-time constraints and an IoT OS must have the option to give the planning required for these sorts of frameworks. RTOS is based on multitasking scheduling therefore the preemptive scheduler is mostly preferred.

Contiki uses a hybrid model for the purpose of attempting preemptive behavior due to the requirement for preemptive scheduling. This approach relies on a preemptive multithreading event-driven kernel for application libraries that are linked with programmers. Contiki includes a real timer library to perform real-time scheduling tasks in order to provide real-time capabilities. However, there is no real-time scheduling application compared to freeRTOS. FreeRTOS is therefore more reliable than others.

- ***Programming model***

The operating system's performance and productivity are significantly influenced by the programming model. It should help in expanding efficiency for engineers just as using abstraction to the underlying system.

As the Contiki and Tiny operating systems are event-driven, they use less memory than FreeRTOS. In this specific circumstance, the event driven Contiki kernel doesn't give multi-stringing without anyone else however their multithreaded applications use protothreads through optional libraries. TinyOS version 2.1 also provides support for multithreading through the use of TOS Threads, which employ a cooperative threading strategy, similar to how Contiki employs proto threads to join multithreading.

Table I provides a comparison of IoT operating systems based on different features, highlighting their advantages and disadvantages. Table II presents a comparison of various operating systems based on the evaluation criteria used in the research study, The table summarizes the key findings of the study and presents the conclusions drawn based on the evaluation results while Table III describe Operating systems representations based on performance.

Table I: IoT OS Advantages, disadvantages and Comparison based on different features.

OS	Architecture	Programming Model	Scheduling	Memory Management	File System Management	Advantage	Limitation
Contiki	Monolithic	Protothreads	Cooperative, Preemptive	Dynamic	Coffee flash	Support File System for a Flash and Low Module Interaction Cost	The file's size needs to be reserved in advance and Lack of memory protection unit.
Tiny OS	Monolithic	Event-driven, Threads	Cooperative	Static	Single file system	optimal competition without increasing resource consumption and reduce the total amount of energy used	Lack of Memory usage prediction. It cannot handle the large Number of files at a time.
RIOT	microkernel	Multithreading	Preemptive, Priority Based	Dynamic Static	FAT file system	It includes a method for real-time scheduling and is capable of supporting a filesystem designed for FAT embedded devices.	There is no MMU or Floating Point Unit
LiteOS	Modular	Multithreaded, Event-driven	Priority Based, RR	Dynamic	LiteFS	The file system is alike to the Unix file system and dynamic allocation creates a flexible system.	There are no built-in networking protocols
Tizen	ARM, x86	Object-oriented	Preemptive	Virtual Memory	EXT4, F2FS	Good performance, supports many device types	Limited community support, limited app ecosystem
Ubuntu Core	ARM, x86	Object-oriented	Preemptive	Virtual Memory	EXT4, BTRFS, SquashFS	Security-focused, supports containerization	Limited hardware support, limited commercial adoption
Mbed OS	ARM	Event-driven	Co-operative	Heap and Stack	FAT file system	Lightweight, supports low-power devices	Limited support for non-ARM architectures
FreeRTOS	ARM, x86	Event-driven	Co-operative	Heap and Stack	FAT, NTFS	Small footprint, suitable for embedded systems	Limited features compared to desktop OSes
Contiki	ARM, x86	Event-driven	Co-operative	Heap and Stack	Proprietary	Good for IoT applications, supports multiple platforms	Limited community support, not suitable for high-performance applications
Windows 10 IoT	x86, ARM	Object-oriented	Preemptive	Virtual Memory	NTFS, FAT32	Familiar interface for Windows users	Limited hardware support, not open-source
Zephyr	ARM, x86	Event-driven	Preemptive	Heap and Stack	FAT file system	Supports many platforms, easy to use	Limited community support, not suitable for high-performance applications
OpenWSN	ARM	Event-driven	Preemptive	Heap and Stack	Proprietary	Optimized for low-power networks	Limited community support, not suitable for general-purpose applications

NuttX	ARM, x86	Event-driven	Preemptive	Virtual Memory	FAT file system	Good for real-time systems, supports POSIX	Limited hardware support, not suitable for high-performance applications
Linux	ARM, x86	Object-oriented	Preemptive	Virtual Memory	EXT4, BTRFS, XFS	Supports many applications and platforms, open-source	May require more resources than other OSes, can be complex to configure

Table II: Summarization of key findings of the study and conclusions based on the evaluation results for various operating systems

Study Title	Operating Systems Compared	Evaluation Criteria	Key Findings Conclusion
Attack mapping for IoT" (2022) [24]	Tizen for IoT, UbuntuCore, Mbed OS, RIOT, Amazon FreeRTOS, Contiki, Windows 10 IoT	Real-time performance, connectivity, security,and battery consumption	Contiki and RIOT performed best in terms of power consumption, While Mbed OS and RIOT earned highly in terms of security and connectivity. Real-time performance for Windows 10 IoT was determined to be satisfactory.
"A Survey on Resource Management and Security Issues in IoT Operating Systems." (2022) [23]	Contiki, TinyOS, FreeRTOS, RIOT and Zephyr	Compatibility with sensors, reliability, energy economy, scalability, security	All five operating systems worked with various devices, Strong security elements were present in all five operating systems. TinyOS was the most sensor-friendly. The most reliable systems were Contiki BUT TinyOS had most compatibility while FreeRTOS was the most energy-efficient. Zephyr and RIOT were the most scalable.
" IoT Solutions with Eclipse IoT Technologies: An Open-Source Approach to Edge Computing " (2022) [25]	Zephyr, FreeRTOS, Contiki, and RIOT	Memory footprint, power consumption, real-time capabilities, security, development community	All four operating systems offered strong security features, Zephyr used smallest memory footprint , RIOT consumed least power , FreeRTOS had the best real-time performance. While Contiki had largest development community.
"Vision, Challenges and future perspective of low constrained devices IOT operating systems " (2020) [22]	Contiki, TinyOS, RIOT, FreeRTOS, Zephyr, NuttX, Linux	Memory usage, power consumption, network performance	RIOT was found to be the best in terms of memory usage and power consumption. TinyOS and Contiki performed well in network performance. Zephyr and FreeRTOS were found to have good scalability.
“Operating systems for Internet of Things low-end devices: Analysis and benchmarking”(2019)[27]	Contiki, FreeRTOS, RIOT, Zephyr	Memory requirements, power consumption, real-time skills, and security	All four operating systems had strong security features, but FreeRTOS had the best real-time performance and RIOT used the least memory and energy. while Zephyr and Contiki have high energy effectiveness.
" An investigation on several operating systems for internet of things " (2019)[21] [28]	Android Things, Contiki, FreeRTOS, RIOT, TinyOS	Connectivity, security, power consumption, real-time performance	Contiki and RIOT were found to have excellent power consumption, Android Things was found to have good connectivity and security while TinyOS was found to work well in real-time.
" A study on internet of things operating systems " (2019) [10]	Contiki, TinyOS, FreeRTOS, RIOT, Linux	Energy consumption, memory usage, network performance	It was discovered that TinyOS and FreeRTOS had excellent network performance while RIOT and contiki had greatest memory and energy
""Performance study of real-time operating systems for internet of things devices " (2018) [26]	Contiki, RIOT, TinyOS, FreeRTOS, OpenWSN, Linux	Energy consumption, memory usage, network performance	RIOT and Contiki were found to have the best energy consumption and memory usage. TinyOS and FreeRTOS were found to have good network performance.

Table III: Operating systems representations based on performance.

Operating System	Real-time Performance	Connectivity	Security	Battery Consumption	Compatibility with Sensors	Reliability	Energy Economy	Scalability	Memory Footprint	Network Performance	Development Community	Power Consumption
Tizen	✓	✓	✓	✓	×	×	×	×	×	×	×	×
Ubuntu Core	×	✓	✓	×	×	×	×	×	×	×	×	×
Mbed OS	×	✓	✓	×	×	×	×	×	×	×	×	✓
RIOT	✓	✓	✓	✓	×	×	×	✓	✓	×	✓	×
Amazon FreeRTOS	×	✓	✓	×	×	×	✓	×	×	×	×	×
Contiki	×	✓	✓	×	✓	✓	×	✓	✓	✓	✓	×
Windows 10 IoT	✓	✓	✓	✓	×	×	×	×	×	✓	×	×
Zephyr	✓	✓	✓	×	×	×	×	✓	✓	×	✓	×
TinyOS	×	×	✓	×	✓	✓	✓	✓	×	✓	×	×
FreeRTOS	×	×	✓	×	×	×	✓	✓	×	✓	×	✓
Open WSN	×	×	✓	×	✓	×	×	×	✓	✓	×	✓
NuttX	×		✓	×	×	×	×	×	✓	×	×	×
Linux	×	✓	✓	×	×	×	×	×	✓	✓	✓	×

✓=Feature is present or performs well ×= Feature is not present or performs poorly

Graphical representation:

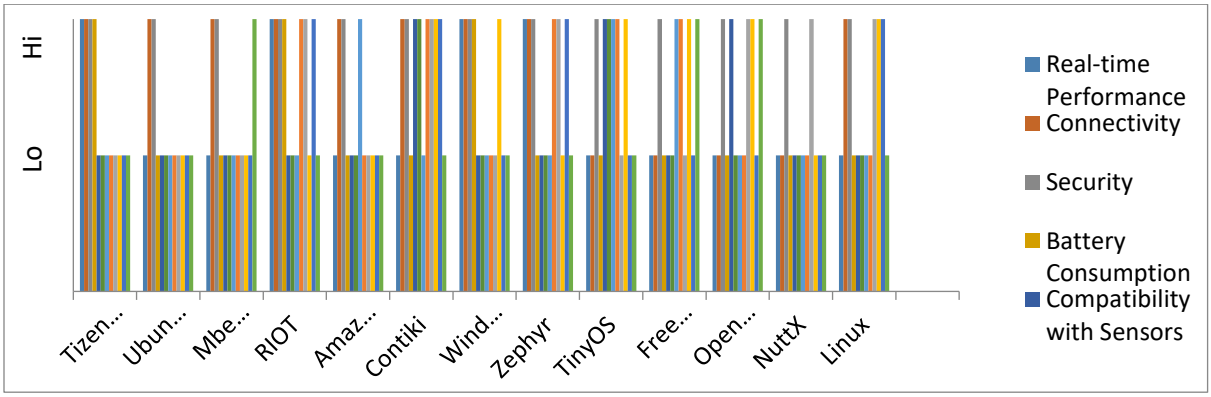

Fig. 2: Graphical Representation of IOT OS performances

High=Feature is present or performs well
Low= Feature is not present or performs poorly

IV. CONCLUSION

Different operating systems was selected and discuss according to their usage and features. On behalf of many challenges and newness, it is difficult to find which one is the best operating system. We present some major concerns for OS IoT which are OS architecture, programming model, and real-time capability. After that, we compare the Operating systems according to these all concerns. But it is difficult to say which one is best. We can choose the best operating system according to the requirement of IoT devices. Although the criteria used to assess the operating systems vary from study to study, all place a strong emphasis on factors relevant to Internet of Things (IoT) devices, such as power consumption, real-time capabilities, scalability, and security. Despite the fact that each operating system varies in strengths and weaknesses based on

the evaluation criteria, they were all created to handle the particular problems faced by IoT devices. This paper can help researchers in understanding the Internet of Things-IoT, their features, advantages, and Limitations.

REFERENCES

[1] Gaur, Padmini, and Mohit P. Tahiliani. "Operating systems for IoT devices: A critical survey." In *2015 IEEE region 10 symposium*, pp. 33-36. IEEE, 2015.

[2] Sabri, Challouf, Lobna Kriaa, and Saidane Leila Azzouz. "Comparison of IoT constrained devices operating systems: A survey." In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pp. 369-375. IEEE, 2017.

[3] Asim, Muhammad, and Waseem Iqbal. "Iot operating systems and security challenges." *International Journal of Computer Science and Information Security* 14, no. 7 (2016): 314.

[4] Baccelli, Emmanuel, Cenk Gündoğan, Oliver Hahm, Peter Kietzmann, Martine S. Lenders, Hauke Petersen, Kaspar Schleiser, Thomas C. Schmidt, and Matthias Wählisch. "RIOT: An open-source operating system for low-end embedded devices in the IoT." *IEEE Internet of Things Journal* 5, no. 6 (2018): 4428-4440.

[5] Zikria, Yousaf Bin, Heejung Yu, Muhammad Khalil Afzal, Mubashir Husain Rehmani, and Oliver Hahm. "Internet of things (IoT): Operating system, applications and protocols design, and validation techniques." *Future Generation Computer Systems* 88 (2018): 699-706.

[6] Baccelli, Emmanuel, Oliver Hahm, Mesut Günes, Matthias Wählisch, and Thomas C. Schmidt. "RIOT OS: Towards an OS for the Internet of Things." In *2013 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, pp. 79-80. IEEE, 2013.

[7] Musaddiq, Arslan, Yousaf Bin Zikria, Oliver Hahm, Heejung Yu, Ali Kashif Bashir, and Sung Won Kim. "A survey on resource management in IoT operating systems." *IEEE Access* 6 (2018): 8459-8482

[8] https://www.quora.com/What-is-the-difference-between-RTOS-and-FreeRTOS.

[9] https://en.wikipedia.org/wiki/Contiki

[10] Al-Taleb, Najla, and Nasro Min-Allah. "A study on internet of things operating systems." In *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pp. 1-7. IEEE, 2019.

[11] Javed, Farhana, Muhammad Khalil Afzal, Muhammad Sharif, and Byung-Seo Kim. "Internet of Things (IoT) operating systems support, networking technologies, applications, and challenges: A comparative review." *IEEE Communications Surveys & Tutorials* 20, no. 3 (2018): 2062-2100.

[12] Bansal, Sharu, and Dilip Kumar. "IoT ecosystem: A survey on devices, gateways, operating systems, middleware and communication." *International Journal of Wireless Information Networks* 27 (2020): 340-364.

[13] https://www.researchgate.net/publication/320000233_A_Comparative_Study_between_Operating_Systems_Os_for_the_Internet_of_Things_IoT

[14] Chandra, Tej Bahadur, Pushpak Verma, and A. K. Dwivedi. "Operating systems for internet of things: A comparative study." In *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*, pp. 1-6. 2016.

[15] Levis, Philip, Samuel Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay et al. "TinyOS: An operating system for sensor networks." *Ambient intelligence* (2005): 115-148.

[16] Adekotujo, Akinlolu, Adedoyin Odumabo, Ademola Adedokun, and Olukayode Aiyeniko. "A Comparative Study of Operating Systems: Case of Windows, UNIX, Linux, Mac, Android and iOS." *International Journal of Computer Applications* 176, no. 39 (2020): 16-23.

[17] Jaskani, Fawwad, Saba Manzoor, Muhammad Amin, Muhammad Asif, and Muntaha Irfan. "An investigation on several operating systems for internet of things." *EAI Endorsed Transactions on Creative Technologies* 6, no. 18 (2019).

[18] Silva, Miguel, David Cerdeira, Sandro Pinto, and Tiago Gomes. "Operating systems for Internet of Things low-end devices: Analysis and benchmarking." *IEEE Internet of Things Journal* 6, no. 6 (2019): 10375-10383.

[19] Umashankar, M. L., S. Mallikarjunaswamy, N. Sharmila, D. Mahesh Kumar, and K. R. Nataraj. "A Survey on IoT Protocol in Real-Time Applications and Its Architectures." In *ICDSMLA 2021: Proceedings of the 3rd International Conference on Data Science, Machine Learning and Applications*, pp. 119-130. Singapore: Springer Nature Singapore, 2023.

[20] Malallah, HayfaaSubhi, Subhi RM Zeebaree, Rizgar R. Zebari, Mohammed AM Sadeeq, Zainab Salih Ageed, Ibrahim Mahmood Ibrahim, Hajar Maseeh Yasin, and Karwan Jameel Merceedi. "A comprehensive study of kernel (issues and concepts) in different operating systems." *Asian Journal of Research in Computer Science* 8, no. 3 (2021): 16-31.

[21] Al-Boghdady, Abdullah, Khaled Wassif, and Mohammad El-Ramly. "The presence, trends, and causes of security vulnerabilities in operating systems of IoT's low-end devices." *Sensors* 21, no. 7 (2021): 2329.

[22] Rounaq, Sumera, and Muhammad Iqbal. "Vision, Challenges and Future Perspectives of Low Constrained Devices IOT Operating Systems: A Systematic Mapping Review." *European Journal of Engineering and Technology Research* 5, no. 12 (2020): 107-115.

[23] AlDossary, Noura, Sarah AlQahtani, and Haya AlUbaidan. "A Survey on Resource Management and Security Issues in IoT Operating Systems." In *2022 Fifth International Conference of Women in Data Science at Prince Sultan University (WiDS PSU)*, pp. 26-30. IEEE, 2022.

[24] Bathgate, William. "Attack Mapping for IoT." (2022).

[25] Desbiens, Frédéric. "Operating Systems." In *Building Enterprise IoT Solutions with Eclipse IoT Technologies: An Open-Source Approach to Edge Computing*, pp. 243-267. Berkeley, CA: Apress, 2022.

[26] Raymundo Belleza, Rafael, and Edison de Freitas Pignaton. "Performance study of real-time operating systems for internet of things devices." *IET Software* 12, no. 3 (2018): 176-182.

[27] Silva, Miguel, David Cerdeira, Sandro Pinto, and Tiago Gomes. "Operating systems for Internet of Things low-end devices: Analysis and benchmarking." *IEEE Internet of Things Journal* 6, no. 6 (2019): 10375-10383.

[28] Jaskani, Fawwad, Saba Manzoor, Muhammad Amin, Muhammad Asif, and Muntaha Irfan. "An investigation on several operating systems for internet of things." *EAI Endorsed Transactions on Creative Technologies* 6, no. 18 (2019).

[29] Homayouni, Sara, and Reza Javidan. "ERA-ContikiMAC: An adaptive radio duty cycling layer in Internet of Things." In *2018 9th International Symposium on Telecommunications (IST)*, pp. 74-79. IEEE, 2018.