Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 10, October 2024

A Comprehensive Survey of Operating Systems for Smart Home Networks Based on IOT

Suparna N, Manjaiah D H
Department of PG Studies and Research in CSc,
Mangalore University Mangalore, Karnataka

ABSTRACT
The entire world is leaping with the biggest technological steps and moving towards automated lifestyles with either Artificial Intelligence networks or IoT networks. The researchers in the field of Computer Science are steering extensive studies to provide various services and Smart Home Automation is one such service. This study guides those enquirers who want to choose a safe and secure smart home system with suitable operating system (OS) that assists the development of reliable application software for home automation by providing a convenient and safe abstraction of IoT devices. The study has taken up comparison of existing surveys and available Operating Systems for Smart Home Networks.

***KEYWORDS*: Smart Homes, Cooja, Constrained devices, TinyOS, Contiki.**

1 INTRODUCTION

Researchers in the fields of science and technology always focus on making the fruitions of their projects revolutionize various fields such as homes, health, education, construction, automobiles, a nation's infrastructure, and agriculture [1]. The field of IoT is such an affluence in technology, where anything and everything can be connected to the Internet and controlled remotely. The use of sensors to accumulate data without any human intervention has made IoT a ubiquitous field and has amplified the process of transforming the human lifestyle. Studies in this field fall into numerous categories. The The extensive survey of research papers in the field of IoT shows that most of the researchers have taken up common issues in the fields such as security, performance of IoT networks, performance analysis of various protocols, the IoT Eco system, the performance of Operating Systems etc. The OS plays a major role in IoT networks because most of the components and devices have constrained resources. IoT OS has varied hardware constraints such as low memory, less computational power, limited resources, and low battery life, and studies have shown that the IoT OS must be equipped to handle these constraints; however, the complexity of the OS must be kept low because the MCU will work at very low clock cycles. Although a number of Operating Systems for IoT are now available, many of them require optimization with reduced complexity. This study contributes to a distinct comparison perspective on IoT OSes.

2 THE IMPETUS FOR THE STUDY

Home automation is no longer a dream or part of a science-fiction movie. Smart homes have come to existence, and the number of families adapting smart home technology is exponentially increasing worldwide. IoT Home Automation controls the electrical or electronic appliances of our homes using Internet-connected systems. The Operating Systems used in IoT networks are called Smart Home Operating Systems and are designed to coalesce all the connected devices across the home and control them from a single platform. A comprehensive smart home OS must be able to create an intelligent system by providing technical support to all types and categories of IoT devices. An example of choosing a smart home OS is the need of the hour, and thus we conducted a critical survey of Smart Home Operating Systems. We also studied existing survey papers and listed the features of these surveys.

3 REVIEWS OF LITERATURES

3.1 IoT Ecosystem: A Survey on Devices, Gateways, Operating Systems, Middleware, and Communication [2]: In this article, Bansal et al. categorized IoT as high-end and low-end, and further formed sub-categories such as Linux-based OS and Non-Linux-based OS. In this study, the authors listed and briefly explained the design features for lightweight OS, such as Architecture, Scheduling, Memory management, interfaces and communication protocols, interfaces and communication protocols, simulation ability, Security, Development model, power management, and multimedia features.

3.2 Internet of Things (IoT): operating system, application and protocol design, and validation techniques [3]: In this paper, Zikria et al. stated that the important features to be considered when selecting a Lightweight OS are Energy Efficiency, Memory Footprint, Support for Heterogeneous Hardware, Network Connectivity, Interoperability and Security features. 3.3 WSN OPERATING SYSTEMS FOR INTERNET OF THINGS(IOT): A SURVEY [4]: In this study, Yaqoob et al. listed design features such as Architecture, Programming Model, Scheduling, Memory Management, Resource Sharing and Real-time Application Support for some popular OS such as TinyOS, Contiki, MANTIS, Nano-RK, LiteOS, and RIOT. They also prepared a comparative analysis table of these features.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 10, October 2024

3.4 Survey on Operating Systems for the Applications of the Internet of Things [5]: Kausar Parveen et al. studied TinyOS, Contiki, MANTIS, NanoRK, and RIOT, and summarized these operating systems for the IoT domain according to the properties of resource constraints, Architecture, Real-time requirements, Programming Model, Scheduling, Memory Management and Protection, Communication Protocol Support, Resource Sharing, Portability, Failure handling, safety, security, privacy, scalability, and upgrading for operating system software.

3.5 IoT Operating Systems and Security Challenges [6]: This paper presents a brief study of IoT operating systems and the current security challenges in IoT using RPL and 6LoWPAN (IPv6 over low-power WPAN) protocols.

3.6 Comparative Analysis Of Different Operating Systems Used For Low-End IoT Devices [7]: This study focused on lightweight operating systems designed for low-end IoT devices. This paper presents a comparative analysis of various operating systems and discusses the key strategies to consider in their design. These strategies include general models, scheduling approaches, hardware considerations, flexibility, and system capabilities.

3.7 Survey of Operating Systems for the IoT Environment [8]: This paper explores diverse operating systems designed for resource-constrained IoT environments. It delves into supported platforms and available developer tools, and enables communication protocols, offering a comprehensive overview.

3.8 A Comparative Study Between Operating Systems (OS) for the Internet of Things (IoT) [9]: Hicham et al. discussed the important features of OS for IoT, such as Architecture, Modularity, Communication Protocol Support. The authors listed the advantages and disadvantages of the well-known lightweight OS, namely TinyOS, Contiki, Nano-RK, LiteOS, FreeRTOS, and RIOT, and compared the features of these Operating Systems.

3.9 An Overview of the Internet of Things Closed-Source Operating Systems [10]: In this paper, the authors present an overview of the common and existing closed-source Oss for IoT. Each OS is described in detail based on the set of design and development aspects that we established. These aspects include architecture and kernel, memory management, scheduling, power consumption, networking protocol support, security, programming models, and multimedia support.

4 PARAMETERS FOR SELECTING SUITABLE IOT

OS An Operating System (OS) provides services to users to develop application software with a convenient and safe abstraction of hardware resources. In Servers and personal computers, the OS allocates threads to processors, virtual addresses to locations in memory, and operates storage devices, peripherals, network devices, and media on behalf of the user's application. Generally, IoT networks and devices make use of embedded Operating Systems, but many researchers have developed specific and specialized OSes for IoT. As IoT devices are built for specific usage, there is no one-size-fits-all approach to choosing the OS. If the OS chosen is just adequate for the time being, then it becomes tedious in future to change or alter the OS, if the user needs to add some more technologically advanced things or devices Therefore, the Operating System must provide all the necessary hardware, applications, and connectivity requirements of the product, now and in the future. When selecting the operating system for our Home Automation IoT project, it is vital to consider some key elements and common features of various currently available Operating Systems. Scalability, Portability, Memory Footprint, Modularity, Security, Compatibility, Simplicity, Flexibility, Reliability and Consistency are the key features that facilitate the IoT OS selection.

5 IOT ECOSYSTEM

A set of interconnected devices such as processors, sensors, actuators, and communication hardware enabled by Internet connection constitutes an IoT ecosystem. The basic functioning of IoT system is to acquire, transmit and perform some tasks on the data they obtain from their environments.[11] The Internet Engineering Task Force (IETF) has classified constrained devices used in the IoT field into different [12] according to the required memory for storing code and data[2]. Because the Operating System inhabits a major portion of memory, researchers are thriving to develop tiny Operating Systems suitable for resource-constrained devices.

6 ARCHITECTURES OF OPERATING SYSTEMS

In an effort to reduce the memory footprint, researchers are striving to lessen the complexity of the IoT OS, which has led to no consensus on the architecture for IoT, agreed universally. The structural design of IoT frameworks and protocols defines their architectures. This outline specifies the functions and principles of the physical components of IoT. Currently, researchers are using five well-known architectures of IoT OS: monolithic architecture, microkernel architecture, three- and five-layered architecture, service-oriented architecture, cloud, and fog-based architecture [2],[13]. Monolithic architecture is a combination of the necessary OS components and applications. The services are implemented separately, and each service has an interface for another service. The monolithic approach resulted in an underprivileged design choice for the OS. The micro-kernel architecture provides minimum functionality in the kernel. The application and the OS were built as a set of interacting modules. Therefore, the kernel size was reduced. Another type of OS architecture is virtual architecture, which works on the principle that a virtual machine is exported to user programs that resemble hardware[14]. The three-layered architecture consists of a perception, network, and application layers. As their names suggest, data are sensed and gathered at the perception layer, transported at the network layer, and processed, and the final product is provided at the application layer. In the five-layered architecture, along with the three basic layers of the three-layered

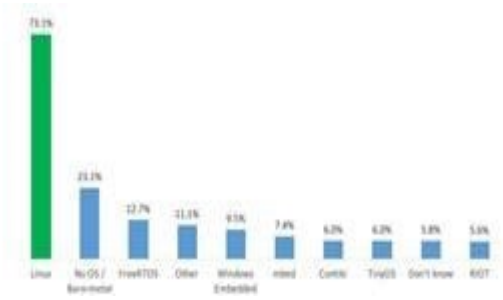
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 10, October 2024

architecture, two more layers were added to provide more abstraction to the IoT architecture. The five layers are perception, transport, processing, middleware, and application [2]. SOA, or service-oriented architecture, is a concept designed to build systems that provide services to applications. It is a design pattern and not restricted to any programming language A service is a well-defined self-contained function that represents a unit of functionality. A service can exchange information with another service. Here, the API does not change even if the inner technology and code are changed. This is not dependent on the state of the other services. It uses a loosely coupled message-based communication model to communicate with the applications and other services. In fog-based architecture, four layers are present between the physical and transport layers: monitoring, pre-processing, storage, and security. The monitoring layer observed and checked the data obtained from the sensors. The preprocessing layer performs operations on the sensed data. The storage layer gathers all the processed data. The security layer is responsible for the integrity and privacy of data.

7 OPEN-SOURCE VERSES CLOSED SOURCE OS IOT operating systems employ microcontroller units (MCUs) to perform fundamental computing tasks on Internet-connected devices. These operating systems fall into two main categories: open-source software (OSS) and commercial or closed operating systems, the latter of which are also referred to as proprietary operating systems. OSS code is accessible to everyone, enabling users to inspect, modify, and enhance it to meet their specific needs. Many prefer OSS to proprietary systems because they tend to be more secure, stable, easily upgradable, and offer greater user control. Some popular OSS include TinyOS, RIOT, Contiki, Mantis OS, Nano RK, LiteOS, FreeRTOS, Apache Mynewt, Zephyr OS, ARM mbed, Yocto, and Raspbian[15]. Linux has released several lightweight operating systems that are specifically designed for IoT and WSN networks. These operating systems can be categorized based on the storage space required for their installation. Versions such as Xubuntu, Zorin OS Lite, Arch Linux, Bunsen Labs Linux Lithium, Bodhi Linux, and Linux Lite require only 1GB storage space. In contrast, Porteus and Puppy Linux occupy less than 500MB. Notably, SliTaz and Tiny Core Linux are the most intriguing options, as they require a maximum of only 100MB of storage space. An Operating System whose source code is not accessible by the public is called CSS or closed Software OS. Only the individual or institution that has created the OS can change the source codes of the OS, and it needs a valid license before installation into any computer. The major names in CSS are Android Things, Windows 10 IoT, WindRiver VxWorks, Micrium µC/OS, Micro Digital SMX RTOS, MicroEJ OS, Express Logic ThreadX, TI RTOS, Freescale MQX, Mentor Graphics Nucleus RTOS, Green Hills Integrity, and Particle[10]. The IoT OS must ensure security, connectivity, interoperability, networking, storage management, and remote-device management.

Therefore, the development of the IoT OS has become a competitive task for developers, and several projects have been set up to perform this tedious task. Some of the projects are developed by individual researchers such as Richard Barry, J P Norair, Dave Hudson and Adam Dunkels. Several studies and surveys have been conducted, and forums and mail lists have been formed to improve the functionalities of these projects. In this study, we examined 18 important Operating Systems for IoT projects. Table 1 summarizes some IoT OS projects and papers published on these projects.

Fig 1: Survey results for operating systems used for IoT devices



(Source: IoT Developer Survey 2016)

IoT operating systems are designed to connect devices seamlessly. With built-in support for various communication protocols, they effectively bridge the gap between different wireless technologies, allowing for effortless connectivity. Furthermore, these systems are incorporated with highly secure communication technologies like Bluetooth, WiFi, Ethernet, WiMAX, LoRa, Z-Wave and Zigbee etc. and also improve security by implementing encryption, authentication, and data integrity measures. Thus, IoT OS ensures that IoT devices communicate securely, protecting them from unauthorized access and tampering.

Table 1: The Operating Systems for IoT projects and the papers published on those projects

Windows IoT	Microsoft	1999	[84], [85], [86], [87], [10], [88]
MicroC/OS-III	Micrium, Inc. Micro-Controller Operating Systems; designed by Jean J. Labrosse	1991	[89], [90], [91], [81], [92]
NutOS	Dave Hudson – Original project was Liquorice.	2000	[93], [94], [95], [96]

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 10, October 2024

Project Name	Organization	Year	Research papers published on the Project
Zephyr	Linux Foundation	2016	[16],[17],[18],[19],[20]
Tiny OS	EECS Department of U.C. Berkeley.	2007	[21],[22],[23],[24],[25],[26], [27]
RIOT	Free University of Berlin French Institute for Research in CSc and Automation Hamburg University of Applied Sciences	2013	[28], [29], [30], [31],[32] [33],[34]
Contiki	Adam Dunkels	2002	[28], [29], [30], [31]
Mantis OS	MANTIS Wireless Sensor Networking Project, University of Colorado at Boulder	2003	[35],[36],[37],[38]
LiteOS	Huawei	2007	[39], [40],[41],[42], [43]
FreeRTOS	Richard Barry ; Real Time Engineers Ltd.	2003	[34],[44],[45],[46],[47]
ARM mbed	ARM employees Simon Ford and Chris Styles	2007	[48], [49], [50], [51], [52], [53]
Yocto	The Linux Foundation	2010	[54], [55], [56], [57], [58], [59], [60], [61]
Raspbian	Mike Thompson and Peter Green at Raspberry Pi Foundation	2012	[62], [63], [64], [65]
Brillo	Google	2015	[66]
Android Things	Google	2018-2022	[67], [68], [69], [70], [71]
Erika Enterprise	Evidence Srl, ReTiS Lab	2002	[72], [73], [74], [75], [76], [77]
OpenTag	JP Norair	2011	[78], [79]
uClinux	D. Jeff Dionne and Kenneth Albanowski	1998	[80], [81], [82], [81], [83]

7 BRIEF STUDY OF LIGHTWEIGHT OS

7.1 Tiny OS: This is an application specific and component based Operating System that requires a memory footprint of 400bytes [97]. TinyOS written using the programming language nesC and available with BSD license. The SDK for TinyOS consists of TinyDT, TinyOS Eclipse Plugin – YETI 2 and Eclipse Editor plugin[8]. It provides excellent support for networking and has incorporated support for multiple wireless bands and standards [98].

7.2 Zypher: With a smallest memory footprint, Zephyr is a secure and flexible real time operating system best suited for smart home networks as it supports more than 100 developer boards. Zephyr requires only 8KB RAM and this suited for all types of home automation. With monolithic kernel, Zephyr supports various architectures such as RISC-V 32, ARM Cortex-M, , Tensilica Xtensa, NIOS-II, and Intel x86. This OS is programmed using Python using Kconfig and Devicetree as its configuration systems and thus can be ported to non-Linux operating systems. The project has multi threading services and priority based pre-emptive scheduling with round-robin time slicing.

7.3 RIOT: Real Time IoT is a microkernel-based operating system with a minimum RAM footprint of 1.5kB and ROM required is around 5KB[99].RIOT supports 16 and 32bit MCUs such as MSP430 or a ARM7[100]. It does not need a Memory Management Unit (MMU) nor a Floating Point Unit (FPU)[31].The RIOT project is developed with a tickles scheduler for energy efficiency and for real time scheduling it uses Deterministic O(1) scheduling [32]. It has a modular structure with low latency interrupt handling. It offers pre-emptive multithreading service with powerful IPC[31].

7.4 Mantis OS: MANTIS is a lightweight POSIX-like and energy efficient multithreaded operating system for Multimodal Networks of In-Situ micro sensor nodes. It is a cross-platform embedded OS with pre-emptive time-sliced scheduling. With a RAM requirement of 500KB,this OS is well suited for smart home networks. [35], [36], [38].

7.5 LiteOS: This lightweight OS has a Unix-like programming environment and it consists of three subsystems viz., LiteFS, LiteShell and Kernel. The user interacts with IoT devices from LiteShell using Unix like commands. The Kernel executes these commands and LiteFS File System provides support to file and directory related operations[41]. LiteOS runs on platforms such as MicaZ, with an 8MHz CPU and a memory footprint of 128K bytes of program flash, and 4K bytes of RAM[101].At the Kernel level LiteOS supports dynamic memory. This OS implements priority based and round-robin scheduling in the Kernel. LiteOS dos not have any in-built networking protocol stacks but it supports plug-and-play routing stack[102]

7.6 ARM mbed: ARM mbed is single-threaded, event-driven and modular. It's has good connectivity and low footprint

7.7 Yocto: With a layered architectural design, The Linux Foundation collaborative, Yocto Project is a platform to create customized OS for IoT networks. It has an excellent support for Raspberry Pi or the BeagleBone, or MinnowBoard. Yocto Project output can be transferred to other platform orto another platform. Usually Yocto uses 2GB RAM per virtual core and allows for easy re-use of code

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 10, October 2024

7.8 Apache Mynewt: Apache Mynewt requires 8KB of RAM and 64 KB of ROM. It's kernel takes up only 6KB. Communication protocols typically take up 50-100KB of ROM

7.9 Contiki: Contiki is a platform that provides software and hardware for Wireless Sensor Networks. Adam Dunkels created Contiki in 2002. Contiki platform has a pre-emptive multithreading architecture and an event-driven programming model, which uses Protothreads, Contiki requires only 2KB of RAM and 40KB of ROM. The Contiki OS features Cooja, a network simulator[8]

7.10 Brillo: Brillo is a new Android-based embedded OS for IoT launched in 2016 by Google Being a power frugal System, it should work with even the most basic hardware. Only 128Mb of storage and 32MB of Ram is Brillo's memory footprint. Brillo is accompanied with the full stack application framework with complete secure protocol stack called Weave. It is Open Source Version of Android OS, which is scaled down to suit resource constrained devices. Brillo supports connectivity like Wifi and BLE. It also supports the Thread protocol used in Google's Nest Devices and Android Things.

7.11 Android Things: In 2018 Google Launched its first Operating System built for IoT, called Android Things. To handle the communication with peripherals and drivers, This OS has Android Things Library which supports industry standard protocols such as GPIO, I2C,,PWM , UART and SPI.. Google dropped Android Things project in January 2021 and completely shutdown its console.[67]

7.12 Erika Enterprise: Started in 2002 by Evidence Srl, ReTiS Lab, Italy, Erika Enterprise is a Real Time OS with a support for multicore architecture. It is suitable for all kinds of micro controllers ranging from 8 to 64bits. It also supports hypervisors such as JailHouse and scaled up in 2018 by adding AUTOSAR and Graphical Editor and thus making it ready for the Vehicular Ad hoc Networks. The Erika Enterprise OS contains single image Kernel shared among various CPUs. Usually in embedded OS, multi-processor resource policy (MSRP) allows, tasks on a core to share single protocol stack. But being a multicore architecture Erica Enterprise needs a flexible spin-lock model. In 2014 Sara Afshar et al.,[77] proposed "a flexible spin-based model for locking global resources in a multiprocessor real-time system" and in 2018 implemented "the flexible spin-lock model(FSLM) in ERIKA Enterprise on a multi-core platform."[73] . S.Muthu N et al[74] published a hypothesis in 2017 introducing dual stack for FSLM in the Erica Enterprise.

7.13 OpenTag: Open Tag is a minimal exokernel, open-source RTOS. Exokernel is used to have a direct contact with the system architecture. This OS was developed in C programming language, having Exokernel system architecture; it has event driven programming and pre-emptive scheduling model. It is implemented with DASH7 protocol stack. It provides dynamic memory management and deep sleep mode for power management. DASH7 is a wireless standard designed for low-power and low-latency communication. OpenTag is a full featured exo-kernel with large API and Library. On MSP430 boards OpenTag entails 16-24KB ROM (Flash) and 2KB RAM[78].

7.14 µClinux: µClinux is a open source project developed by D. Jeff Dionne and Kenneth Albanowski in 1998. The name µClinux is pronounced as "you-see-Linux". But the name actually is the combination of the Greek Alphabet µ(mu) which stands for Micro, the English Capital C which is the abbreviation for Controller and the word Linux tells us the fact that µClinux is derived from the Linux 2.0/2.4 kernel and inherits some features of Linux which are suitable for embedded OS. This OS supports Motorola 68, ARM, Sparc, MIPS, Altera and NEC architectures. It is specifically aimed at CPUs without MMU (Memory Management Unit) and requires 32 MB and the size of bootable image starts from 0.8 MB.

7.15 MicroC/OS-III: This is an open-source project developed by Micrium, Inn and designed by Jean J. Labrosse. µC/OS-III is the acronym for Micro-Controller Operating Systems Version 3. It has a microkernel architecture and its highly portable and scalable. Its maximum ROM footprint is 24KB and only 1KB Ram is adequate to use it in microcontrollers and DSPs. µC/OS-III uses pre-emptive Round-Robin scheduling and is multitasking OS[103].

7.16 NutOS: Nut/OS is a modular, open source, real-time operating system with simple RTOS Kernel which provides services to run Ethernut, the TCP/IP stack. The Ethernut software network stack is called Nut/Net.[95] It provides a prevalent API for various protocols. It is easily configurable and highly scalable. It features Co-operative multithreading and dynamic memory management. The memory footprint of Nut/OS is 128/256k bytes Flash memory and 4K bytes on-chip EEPROM[96].

7.17 Windows IoT: Windows 10 IOT is a proprietary operating System developed by Microsoft. It is released in 3 different versions and Windows 10 IoT Core was first released by Microsoft in August 2015. The Core version is a light-weight version of Windows 10 and is optimised to run on small, constrained devices with a memory foot print of at least 256 MB of RAM memory and 2 GB of storage memory. Windows 10 IoT Core embedded devices need a minimum processing power of 400MHz and Windows 10 IoT OS uses pre-emptive scheduling with a Hybrid kernel Architecture. We have prepared a comparison table of this paper with 12 other survey papers and tabulated in Table 3

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 10, October 2024

Table 2 Study of various features of Lightweight Operating Systems

Paper		[97]	[104]	[105]	[106]	[107]	[102]	[16]	[10]	[14]	[108]	[12]
Discussed Features	Architecture	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
	Programming Model	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
	Scheduling	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Memory Management	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Resource Sharing	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y
	Real-time Application Support	Y	Y	Y	Y	Y	N	N	N	Y	N	Y
	Portability	N	Y	N	N	N	Y	Y	N	N	N	Y
	Upgradability	N	Y	N	N	N	N	N	N	N	N	N
	Energy Efficiency	N	Y	N	N	N	Y	Y	N	N	N	Y
	Resource constrained Computing	N	Y	N	N	Y	N	N	N	N	N	Y
	Failure handling	N	Y	N	N	N	N	N	N	N	N	N
	Simulation Support	N	N	N	N	N	N	N	Y	N	N	N
	Communication Protocol Support	N	Y	Y	N	Y	N	N	Y	N	Y	N
	Supported platforms	N	N	Y	N	Y	N	N	N	N	N	N
	Networking TechNlogies	N	N	N	N	N	N	N	N	Y	Y	N
	Licence	N	N	Y	N	N	N	N	N	N	N	N

8 CONCLUSIONS

Whenever an existing technology botches up, an innovative and advanced idea pops up and that technology starts trending. For the subsistence and development of a technology, its vital software part must be improved and made more reliable. The augmentation of Internet of Things is making the smart homes more secure, smarter and reliable. OS support is vital in facilitating the development and subsistence of IoT. In this paper, we first investigate the various IoT OS projects and its contributors. We provide a comprehensive study of the most used and state-of-art closed source OSs for IoT. Then, we provide an extensive overview of the survey papers on IoT OSes, where the various features of OS are studied in detail, based on the established designed and development aspects such as architecture and kernel models, memory management, scheduling, power consumption, security, development and programming model[109]

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 10, October 2024

Table 3: Comparison of this paper with other survey papers with respect to number of IoT OS studied

	Paper	[97]	[104]	[105]	[106]	[107]	[102]	[16]	[10]	[14]	[108]	[110]	[12]	This paper
Operating Systems	RIOT	☑	☑	☑	☑	☑	☑	☑	☑	☑	☒	☑	☑	☑
	LiteOS	☑	☒	☑	☑	☑	☑	☒	☑	☑	☒	☒	☒	☑
	Nano-RK	☑	☑	☑	☑	☒	☑	☒	☒	☑	☒	☒	☒	☑
	MANTIS	☑	☑	☑	☑	☑	☑	☒	☒	☑	☒	☒	☒	☑
	TinyOS	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☒	☑
	Contiki	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☒	☑
	FreeRTOS	☒	☒	☑	☑	☑	☑	☒	☑	☒	☑	☒	☒	☑
	mBED OS	☒	☒	☒	☑	☑	☑	☑	☑	☒	☒	☒	☒	☑
	SOS	☒	☒	☒	☒	☒	☑	☒	☒	☑	☒	☒	☒	☒
	NutOS	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☒	☒	☑
	MicroC/OS-III	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☒	☒	☑
	uClinux	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☒	☒	☑
	OpenTag	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☒	☒	☑
	Erika Enterprise	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☒	☒	☑
	Zypher	☒	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☑	☑
	Brillo	☒	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☒	☑
	Raspbian	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☑
	Android Things	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☑
	Mynewt	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒	☑
	RETOS	☒	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒	☒	☒
	Linux	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☑	☒	☒
	Contiki-NG	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☑	☒
Total		6	5	7	8	7	14	6	9	8	3	4	3	18

REFERENCES:

[1] G. L. P. Ashok, P. S. Akram, M. S. Neelima, J. Nagasaikumar, and A. Vamshi, “Implementation Of Smart Home By Using Packet Tracer,” vol. 9, no. 02, 2020.

[2] S. Bansal and D. Kumar, “IoT Ecosystem: A Survey on Devices, Gateways, Operating Systems, Middleware and Communication,” International Journal of Wireless Information Networks, vol. 27, no. 3, pp. 340–364, 2020, doi: 10.1007/s10776-020-00483-7.

[3] Y. Zikria, S. W. Kim, O. Hahm, M. Afzal, and M. Aalsalem, “Internet of Things (IoT) Operating Systems Management: Opportunities, Challenges, and Solution,” Sensors, vol. 8, pp. 1–10, 2019, doi: 10.3390/s19081793.

[5] K. Parveen, A. Ali, and G. Asadullah, “Survey on Operating Systems for the Applications of the Internet of Things”.

[6] M. Asim and W. Iqbal, “IoT Operating Systems and Security Challenges,” vol. 14, no. 7, 2016.

[7] Z. Riaz, “COMPARATIVE ANALYSIS OF DIFFERENT OPERATING SYSTEMS USED,” vol. 8, no. 1, pp. 64–72.

[8] T. Borgohain, U. Kumar, and S. Sanyal, “Survey of Operating Systems for the IoT Environment,” ArXiv, vol. abs/1504.0, 2015, [Online]. Available: http://arxiv.org/abs/1504.02517

[9] A. Hicham, A. Sabri, A. Jeghal, and H. Tairi, “A Comparative Study between Operating Systems (Os) for the Internet of Things (IoT),” Transactions on Machine Learning and Artificial Intelligence, vol. 5, 2017.

[10]A. Al-sakran, M. H. Qutqut, F. Almasalha, H. S. Hassanein, and M. Hijjawi, “An Overview of the Internet of Things Closed Source Operating Systems,” 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), pp. 291–297, 2018.

[11]P. Sethi and S. R. Sarangi, “Internet of Things : Architectures , Protocols , and Applications,” vol. 2017, 2017.

[12]M. Silva, D. Cerdeira, S. Pinto, and T. Gomes, “Operating Systems for Internet of Things Low-end Devices : Analysis and Benchmarking,” IEEE Internet of Things Journal, vol. PP, no. XX, p. 1, 2019, doi: 10.1109/JIOT.2019.2939008.

[13]P. P. Ray, “A survey on Internet of Things architectures,” Journal of King Saud University - Computer and Information Sciences, vol. 30, no. 3, pp. 291–319, 2018, doi: 10.1016/j.jksuci.2016.10.003.

[14]F. Javed, M. K. Afzal, S. Member, M. Sharif, and B. Kim, “Internet of Things (IoTs) Operating Systems Support , Networking Technologies , Applications , and Challenges : A Comparative Review,” IEEE Communications Surveys & Tutorials, vol. 20, no. c, pp. 1–39, 2018, doi: 10.1109/COMST.2018.2817685.

[15]M. H. Qutqut, A. Al-sakran, F. Almasalha, and H. S. Hassanein, “Comprehensive survey of the IoT open- source OSs,” 2018, doi: 10.1049/iet-wss.2018.5033.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 10, October 2024

[16]F. Jaskani, S. Manzoor, M. Amin, M. Asif, and M. Irfan, "An Investigation on Several Operating Systems for Internet of Things," EAI Endorsed Transactions on Creative Technologies, vol. 6, no. 18, p. 160386, 2019, doi: 10.4108/eai.13-7-2018.160386.

[17]R. K. Panta, S. Bagchi, and S. P. Midkiff, "Zephyr: Efficient incremental reprogramming of sensor nodes using function call indirections and difference computation," Proceedings of the 2009 USENIX Annual Technical Conference, pp. 411–424, 2019.

[18]A. Nyffenegger, "Connecting constrained devices to the cloud using Zephyr A prototype with nRF Connect Bachelor Thesis," 2020.

[19]A. Elvstam and D. Nordahl, "Operating systems for resource constraint Internet of Things devices : An evaluation," 2016.

[20]Y. K. Lee, Y. Kim, and J. N. Kim, "Implementation of TLS and DTLS on Zephyr OS for IoT Devices," 9th International Conference on Information and Communication Technology Convergence: ICT Convergence Powered by Smart Intelligence, ICTC 2018, pp. 1292–1294, 2018, doi: 10.1109/ICTC.2018.8539493.

[21]P. Levis et al., "TinyOS: An Operating System for Sensor Networks," in Ambient Intelligence, vol. 00, 2005, pp. 115–148. doi: 10.1007/3-540-27139-2_7.

[22]P. Thomas, K. Kuladinithi, M. Becker, P. Trenkamp, and C. Goerg, "Performance Evaluation of CoAP using RPL and LPL in TinyOS," pp. 2–6, 2012.

[23]R. Dor et al., "Student installation of TinyOS," pp. 1–7, 2014.

[24]P. Levis et al., "TinyOS : An Operating System for Sensor Networks".

[25]P. Levis and N. Lee, "TOSSIM : A Simulator for TinyOS Networks," pp. 1–17, 2003.

[26]P. Levis, "TinyOS : An Open Platform for Wireless Sensor Networks Moore ' s Law," 2007.

[27]P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM : Accurate and Scalable Simulation of Entire TinyOS Applications".

[28]E. Baccelli, O. Hahm, M. Wählisch, and M. Günes, "RIOT : One OS to Rule Them All in the IoT To cite this version : RIOT : One OS to Rule Them All in the IoT," 2013.

[29]E. Baccelli et al., "RIOT: An Open Source Operating System for Low-End Embedded Devices in the IoT," IEEE Internet of Things Journal, vol. 5, no. 6, pp. 4428–4440, 2018, doi: 10.1109/JIOT.2018.2815038.

[30]T. Eichinger, "The friendly operating system for the IoT," 2017.

[31]P. Kietzmann et al., "Connecting the World of Embedded Mobiles : The RIOT Approach to Ubiquitous Networking for the IoT," no. 2.

[32]K. Roussel, Y. Q. Song, and O. Zendra, "RIOT OS paves the way for implementation of high-performance MAC protocols," SENSORNETS 2015 - 4th International Conference on Sensor Networks, Proceedings, pp. 5–14, 2015, doi: 10.5220/0005237600050014.

[33]B. Karaduman, M. Challenger, R. Eslampanah, J. Denil, and H. Vangheluwe, "Platform-specific Modeling for RIOT based IoT Systems," Proceedings - 2020 IEEE/ACM 42nd International Conference on Software Engineering Workshops, ICSEW 2020, pp. 639–646, 2020, doi: 10.1145/3387940.3392194.

[34]T. Wroldsen, "A Real Time Operating System for embedded platforms," no. May, 2004.

[35]S. Bhatti et al., "MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms," Mobile Networks and Applications, vol. 10, no. August, pp. 563–579, 2005.

[36]H. Abrach et al., "Mantis," Network, no. October, pp. 50–59, 2003, doi: 10.1145/941350.941358.

[37]H. Abrach et al., "MANTIS : System Support For M ultimod A l N e T works of I n-situ S ensors Technical Report CU-CS-950-03 Department of Computer Science MANTIS : System Support for M ultimod A l N e T works of I n-situ S ensors," no. April, pp. 1–15, 2003.

[38]M. Rangan, "Mos (Mantis Os)," 2005.

[39]Q. Cao, T. Abdelzaher, J. Stankovic, and T. He, "The LiteOS Operating System: Towards Unix-Like Abstractions for Wireless Sensor Networks," in 2008 International Conference on Information Processing in Sensor Networks (ipsn 2008), 2008, pp. 233–244. doi: 10.1109/IPSN.2008.54.

[40]T. V. Chien, H. N. Chan, and T. N. Huu, "A comparative study on operating system for Wireless Sensor Networks," 2011 International Conference on Advanced Computer Science and Information Systems, pp. 73–78, 2011.

[41]V. Vanitha, V. Palanisamy, N. Johnson, and G. Aravindhbabu, "LiteOS based Extended Service Oriented Architecture for Wireless Sensor Networks," International Journal of Computer and Electrical Engineering, vol. 2, no. 3, pp. 432–436, 2010, doi: 10.7763/ijcee.2010.v2.173.

[42]LiteOS User ' s Guide. 2011. [Online]. Available: www.liteos.net

[43]"The LiteOS Operating System Kernel 2.0.1," 2011.

[44]N. Melot, "Study of an operating system: FreeRTOS," 2009.

[45]R. T. E. L. Barry, Mastering the FreeRTOS TM Real Time Kernel.

[46]T. Errors, "Getting Started with FreeRTOS on megaAVR® 0-series," pp. 1–29.

[47]C. Sabri, L. Kriaa, and S. L. Azzouz, "Comparison of IoT constrained devices operating systems: A survey," Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA, vol. 2017-Octob, pp. 369–375, 2018, doi: 10.1109/AICCSA.2017.187.

[48]S. Martin, "Arm Mbed – AWS IoT System," no. January, 2018.

[49]A. R. M. Limited, A. R. M. Limited, T. Instruments, C. C. Studio, and T. Instruments, EM BED D ED SY STEM S : INTRODUCTION TO ARM ® CORTEX -M MICROCONTROLLERS Volume 1 Fifth Edition June 2014 Jonathan W . Valvano Fifth edition 2nd printing. 2014.

[50]N. Sudheer, "Hardware Implementation of Mbed to Mbed Through Controller Area Network Using ARM Cortex Core," vol. 2, no. 8, pp. 2491–2498, 2013.

[51]P. Primiceri, P. Visconti, A. Melpignano, G. Colleoni, and A. Vilei, "Hardware and software solution developed in arm MBED environment for driving and controlling DC brushless motors based on ST X-Nucleo development boards," International Journal on Smart Sensing and Intelligent Systems, vol. 9, no. 3, pp. 1534–1562, 2016, doi: 10.21307/ijssis-2017-929.

[52]D. Ibrahim et al., Introduction, vol. 1999, no. December. 2018. doi: 10.1016/b978-0-08-102969-5.00001-x.

[53]Rob Toulson and T. Wilmshurst, Fast and Effective Embedded Systems Design Applying the ARM mbed. 2017.

[54]M. Swain and A. K. Srivastava, "Design of embedded Linux based voice calling device," International Journal of Applied Engineering Research, vol. 10, no. 15, pp. 35095–35102, 2015.

[55]H. Khandelwal, P. Mankodi, and R. Prajapati, "Enhancement of automation testing system using Yocto project," Proceedings of the International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017, vol. 2017-Janua, no. April 2017, pp. 697–700, 2017, doi: 10.1109/ICECA.2017.8203630.

[56]B. Linux, "Working with Yocto to Build Linux," 2018.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 10, October 2024

[57]S. Rifenbark, “Yocto Project Overview and Concepts Manual,” Yocto Project, 2018, [Online]. Available: https://www.yoctoproject.org/docs/2.5/overview-manual/overview-manual.html

[58]A. P. Navik and D. Muthuswamy, “Dual band WLAN gateway solutions in Yocto Linux for IoT platforms,” Internet of Things for the Global Community, IoTGC 2017 - Proceedings, pp. 1–3, 2017, doi: 10.1109/IoTGC.2017.8008968.

[59]A. Biswas, D. Biswas, S. S. Chauhan, and A. Borwankar, “Smart home equipment control system with raspberry Pi and Yocto,” Proceedings of the World Conference on Smart Trends in Systems, Security and Sustainability, WS4 2020, pp. 553–558, 2020, doi: 10.1109/WorldS450073.2020.9210376.

[60]N. X. P. Semiconductors, “Real-time Edge Yocto Project User Guide Overview,” 2021.

[61]D. Moseley, “Why the Yocto Project for my IoT Project?,” Embedded, 2017, [Online]. Available: https://www.embedded.com/why-the-yocto-project-for-my-iot-project/

[62]A. Kurniawan, Raspbian OS Programming with the Raspberry Pi. 2019. doi: 10.1007/978-1-4842-4212-4.

[63]G. R. Viswanath, J. A. Sadiq, V. R. Priya, and A. Professor, “Scrutiny Mechanism By Raspbian,” International Journal of Scientific Research and Engineering Development, vol. 2, [Online]. Available: www.raspberrypi.org

[64]W. Harrington and W. Harrington, Learning raspbian : get up and running with Raspbian and make the most out of your Raspberry Pi. 2015.

[65]G. Howser, “Raspberry Pi Operating System,” Computer Networks and the Internet, pp. 119–149, 2020, doi: 10.1007/978-3-030-34496-2_8.

[66]I. I. Pătru, M. Carabaș, M. Bărbulescu, and L. Gheorghe, “Smart home IoT system,” Networking in Education and Research: RoEduNet International Conference 15th Edition, RoEduNet 2016 - Proceedings, 2016, doi: 10.1109/RoEduNet.2016.7753232.

[67]L. Jordan and P. Greyling, “Android Projects”.

[68]S. Arslan, O. Dagdeviren, and G. Kardas, “An IoT LDR Bulb Application with Android Things Operating System for Smart Cities,” Proceedings - 2019 Innovations in Intelligent Systems and Applications Conference, ASYU 2019, pp. 1–5, 2019, doi: 10.1109/ASYU48272.2019.8946400.

[69]V. V. R. Dantu, V. V. S. Sai Dasaradha, and P. Sasikumar, “Unified Automotive Location Tracking Using Android Things (IoT),” Wireless Personal Communications, vol. 120, no. 1, pp. 63–79, 2021, doi: 10.1007/s11277-021-08434-y.

[70]T. Cho, H. Kim, and J. H. Yi, “Security Assessment of Code Obfuscation Based on Dynamic Monitoring in Android Things,” IEEE Access, vol. 5, no. X, pp. 6361–6371, 2017, doi: 10.1109/ACCESS.2017.2693388.

[71]S. Arslan, O. Dagdeviren, and G. Kardas, “An IoT LDR Bulb Application with Android Things Operating System for Smart Cities,” Proceedings - 2019 Innovations in Intelligent Systems and Applications Conference, ASYU 2019, no. January 2020, pp. 1–5, 2019, doi: 10.1109/ASYU48272.2019.8946400.

[72]A. Avanzini, P. Valente, D. Faggioli, and P. Gai, “Integrating Linux and the real-time ERIKA OS through the Xen hypervisor,” 2015 10th IEEE International Symposium on Industrial Embedded Systems, SIES 2015 - Proceedings, pp. 218–224, 2015, doi: 10.1109/SIES.2015.7185063.

[73]S. Afshar, M. P. W. Verwielen, P. Gai, M. Behnam, and R. J. Bril, “An implementation of the flexible spin-lock model in ERIKA Enterprise on a multi-core platform,” no. 2016, 2022.

[74] S. M. N. Balasubramanian, S. Afshar, P. Gai, M. Behnam, and R. J. Bril, “A dual shared stack for FSLM in Erika Enterprise A dual shared stack for FSLM in Erika Enterprise,” 2017.

[75]Evidence, “ERIKA Enterprise Manual,” 2012.

[76]P. Pagano et al., “ERIKA and open-ZB: An implementation for real-time wireless networking,” Proceedings of the ACM Symposium on Applied Computing, no. May 2014, pp. 1687–1688, 2009, doi: 10.1145/1529282.1529661.

[77]S. Afshar, M. Behnam, R. J. Bril, and T. Nolte, “Flexible spin-lock model for resource sharing in multiprocessor real-time systems,” Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems, SIES 2014, pp. 41–51, 2014, doi: 10.1109/SIES.2014.6871185.

[78]J. Norair, “Introduction to DASH7 technologies,” Dash7 Alliance Low Power RF Technical Overview, pp. 1–22, 2009.

[79]A. Vegendla, H. Seo, D. Lee, and H. Kim, “Implementation of an RFID Key Management System for DASH7,” Journal of information and communication convergence engineering, vol. 12, no. 1, pp. 19–25, 2014, doi: 10.6109/jicce.2014.12.1.019.

[80]G. Started, “AN 2119 Getting Started with uClinux for STR71x,” no. February, pp. 1–19, 2005.

[81]M. Wang and F. Liu, “Research & implementation of uCLinux-based embedded browser,” pp. 504–508, 2008, doi: 10.1109/apscc.2007.51.

[82]S. Martin, “uClinux CoreCommander BSP User Guide,” vol. 2009, no. January, 2009.

[83]E. Dodiu, A. Graur, and V. G. Gaitan, “Hard-soft real-time performance evaluation of linux RTAI based embedded systems,” Elektronika ir Elektrotechnika, vol. 104, no. 8, pp. 51–56, Oct. 2010.

[84]I. M. Culic and A. Radovici, “Extending the wyliodrin platform for windows 10 IoT core,” Networking in Education and Research: RoEduNet International Conference 15th Edition, RoEduNet 2016 - Proceedings, 2016, doi: 10.1109/RoEduNet.2016.7753246.

[85]J. M. C. Gómez, J. R. Gómez, J. C. Mondéjar, and J. L. M. Martínez, “Non-volatile memory forensic analysis in windows 10 IoT core,” Entropy, vol. 21, no. 12, 2019, doi: 10.3390/e21121141.

[86]A. C. Petrini and V. M. Ionescu, “Implementation of the huffman coding algorithm in windows 10 IoT core,” Proceedings of the 8th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2016, 2017, doi: 10.1109/ECAI.2016.7861103.

[87]P. Sabanal, “I NTO T HE C ORE : I N - DEPTH E XPLORATION O F W INDOWS 10 I O T C ORE”.

[88]C. Bell, Windows 10 for the Internet of Things. 2016. doi: 10.1007/978-1-4842-2108-2.

[89]J. J. Labrosse and W. Road, µC/OS-III.

[90]W. H. Li, J. Y. Zheng, and C. Yu, “Data acquisition system based on uC/OS-III embedded system,” Advanced Materials Research, vol. 619, pp. 85–89, 2013, doi: 10.4028/www.scientific.net/AMR.619.85.

[91]L. Peng, F. Guan, L. Perneel, and M. Timmerman, “Behaviour and performance comparison between FreeRTOS and µC/OS-III,” International Journal of Embedded Systems, vol. 8, no. 4, pp. 300–312, 2016, doi: 10.1504/IJES.2016.077774.

[92]J. J. Labrosse, µC/OS-II - The Real-Time Kernel - User’s Manual. 2015.

[93]M. Payer, “Implementation of a Bluetooth Stack for BTnodes and Nut/OS Version 0.9,” 2004.

[94]M. Strobl, N. Balbierer, and A. Schingale, “Rapid Prototyping Embedded Systems Using Ethernut Boards,” no. February, 2012.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 13 Issue 10, October 2024

[95]"Nut/OS Software Manual".

[96]M. Considerations, "Nut/OS".

[97]A. Yaqoob, M. A. Ashraf, F. Ferooz, A. H. Butt, and Y. Daanial Khan, "WSN Operating Systems for Internet of Things(IoT): A Survey," in 2019 International Conference on Innovative Computing (ICIC), 2019, pp. 1–7. doi: 10.1109/ICIC48496.2019.8966731.

[98]P. Levis et al., "TinyOS : An Operating System for Sensor Networks".

[99]E. Baccelli et al., "RIOT : an Open Source Operating System for Low-end Embedded Devices in the IoT," vol. 4662, no. c, pp. 1–12, 2018, doi: 10.1109/JIOT.2018.2815038.

[100] E. Baccelli, O. Hahm, and M. Gunes, "RIOT: One OS to Rule Them All in the IoT," 2012.

[101] Q. Cao, T. Abdelzaher, and J. Stankovic, "The LiteOS Operating System : Towards Unix-like Abstractions for Wireless Sensor Networks," pp. 233–244, 2008, doi: 10.1109/IPSN.2008.54.

[102] P. Gaur and M. P. Tahiliani, "Operating Systems for IoT Devices: A Critical Survey," in 2015 IEEE Region 10 Symposium, 2015, pp. 33–36. doi: 10.1109/TENSYMP.2015.17.

[103] M. Lv et al., "WCET analysis of the µC/OS-II real-time kernel," Proceedings - 12th IEEE International Conference on Computational Science and Engineering, CSE 2009, vol. 2, no. 2007, pp. 270–276, 2009, doi: 10.1109/CSE.2009.82.

[104] K. Parveen, A. Ali, and G. Asadullah, "Survey on Operating Systems for the Applications of the Internet of Things Introductio n," pp. 9–16, 2016.

[105] A. Jeghal, H. Aberbach, S. Abdelouahed, and H. Tairi, "A Comparative Study Between Operating Systems (Os) For The Internet Of Things (Iot) A Comparative Study between Operating Systems (Os) for the Internet of Things (IoT)," no. November 2019, 2017, doi: 10.14738/tmlai.54.3192.

[106] N. Al-taleb, "A Study on Internet of Things Operating Systems," 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), pp. 1–7, 2019.

[107] C. Sabri, K. Lobna, and S. L. Azzouz, "Comparison of IoT constrained devices operating systems : A Survey," 2017, doi: 10.1109/AICCSA.2017.187.

[108] A. Musaddiq et al., "A Survey on Resource Management in IoT Operating Systems," IEEE Access, vol. 6, pp. 8459–8482, 2018, doi: 10.1109/ACCESS.2018.2808324.

[109] O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes, "Operating Systems for Low-End Devices in the Internet of Things : a Survey," vol. 3, no. 1, pp. 1–16, 2016, doi: 10.1109/JIOT.2015.2505901.

[110] E. Baccelli, O. Hahm, E. Baccelli, and O. Hahm, "Operating Systems for the IoT – Goals , Challenges , and Solutions Operating Systems for the IoT – Goals , Challenges , and Solutions," no. January, 2013.