
Sumera Rounaq, Muhammad Iqbal

Abstract — Now the far-fetched reality has become true with the prominence of IOT (Internet of Things) technology. Various individual devices get connected with each other to establish communication. These devices are built on a microcontroller which is responsible to receive and send information. These devices are very small and appropriate Operating Systems are required on the basis of particular device architecture, scheduling methods, network technologies and programming models. IOT Operating Systems are enormously facilitating low constrained devices to deliver their throughput efficiently in a timely manner. This concept helped a lot in emergence of IOT, which has translated our physical world into a digital cyber world. IOT devices consumes less power, less memory and less energy, therefore they need appropriate Operating Systems to act as interfaces. Low constrained Operating Systems are especially designed to provide support to these low constrained devices. Many researches have been conducted to discuss Operating Systems for these low constrained devices. In this study, capsuleization of Internet of Things and its building blocks, architecture of IOT Operating System and network stack architecture of state-of-the art IOT Operating Systems such as Contiki, Tiny OS, FreeRTOS, RIOT, Zephyr and Mbed OS is investigated. Moreover this, detailed overview of related work is presented with the comparative analysis of this study with the existing surveys. In addition, open research areas are discussed with recommendations.

Index Terms — Internet of Things, Low Constrained Devices, Operating System, RIOT, Zephyr, Mbed OS.

I. INTRODUCTION

Parallel with the expansion of Internet of Things, network of connected electronic devices is expanding across the globe. To facilitate users, these devices exchange data and information and provide value creation which is going to embrace proximate level of digitization.

Acclivity in wireless technologies has increased demand of IOT devices. These IOT devices interconnected with each other, utilizing embedded sensing and communication. The word ‘things’ in IOT is referred to as endpoints (devices and things). IOT is the connection of these endpoints through uniquely identifiable IP addresses. Moreover this, IOT bridges gap between virtual and physical realities by acting as an additional layer of information. In several industries and companies, IOT has created a tangible value. Therefore, by 2050 surge of IOT devices is expected.

According to IOT Analytics [2], in 2016, more than 4.7 billion devices were connected to the internet and by 2025 it is estimated that there will be birth of more than 21 billion devices [1]. As per [2], Fig. 1 depicts the expected growth of global number of connected IOT devices by 2025.

![Graph showing growth of global number of connected IOT devices by 2025](image)

II. CONTRIBUTION AND METHODOLOGY

A. Contribution to this Survey

As compare to recent review papers, this survey is summarized as follows:

- This survey presents contribution, structure and selected studies relevant to this study.
- We present discussion on IOT building components and their architecture.
- Compare to previous review in the same context we cover Contiki, Tiny OS, FreeRTOS, RIOT, Zephyr, Mbed OS for low constrained devices.
- We review IOT taxonomy along with discussion of key features and characteristics of major IOT OS.
- We also contribute towards Network Stack Architecture and also provide a detailed summary of existing surveys.
- Open issues are also discussed to facilitate future research studies.

B. Structure of this Survey

In this paper, we encapsulate details of low-end device IOT Operating Systems in frame of reference of architecture, scheduler, programming model, programming language and network stack architecture. We covered Contiki, Tiny OS, FreeRTOS, RIOT, Zephyr and Mbed OS. The contribution of this paper is structured as follows:

- It presents a detailed review of previous review studies in the section of Related work.
• It discusses network stack architecture with appropriate illustrations.
• It covers overview of IOT building blocks, components and its architecture.

Whole paper is structured as follows:

Section I provides introduction; Section II gives contribution and structure of this survey. Table I presents selected studies to define the inclusion criteria of our systematic mapping. Section III provides overview of IOT, building components, IOT taxonomy and IOT OS architectures. Next is Section IV which defines IOT Operating Systems used for this study. Section V focuses on architecture types and features of IOT Operating Systems. Section VI presents Network Stack architecture and Section VII gives comprehensive review of state of art review studies. Comparative analysis of this review with the existing studies is also discussed in this section. Section VIII is dedicated to open research issues followed by Section IX which is comprised of conclusion.

C. Studies selected for this Survey

Following are the studies taken into consideration to complete this survey. Table 1 presents peer review papers along with year of publication and authors.

![Table 1: Selected Studies](image)

III. OVERVIEW OF IOT

When it comes to no human – machine physical contact while solving problems related to science and engineering domain, only one name comes in our mind and that is IOT. IOT is a revolutionary technology that is gaining popularity by leaps and bounds. Due to advancements in network connectivity, real world objects have liberty to establish connectivity between them. Sharing of information is quite feasible and these objects are identified as nodes in IOT framework. Adding more to it, real world objects join their proficiencies and information storage, Cloud Infrastructure comes into an action to provide analytical and logical computing abilities. In addition to this, IOT devices must be comprised of a Physical Layer (PHY), an interface and an Internet Protocol (IP) address. Table 2 presents IOT building components, Associated devices and their features [3].

IOT. Here the term ‘things’ is a piece of equipment consists of sensing, actuating, storage or processing capability. Gateway act as an intermediate block between the things and Cloud Infrastructure. To ensure smooth and secure flow, Network Infrastructure (NI) comes into an action to provide control over the information. Imbued with computing proficiencies and information storage, Cloud Infrastructure allows analytical and logical computing abilities. In addition to this, IOT devices must be comprised of a Physical Layer (PHY), an interface and an Internet Protocol (IP) address. Table 2 presents IOT building components, Associated devices and their features [3].
TABLE 2: IOT BUILDING COMPONENTS, ASSOCIATED DEVICES AND THEIR FEATURES

<table>
<thead>
<tr>
<th>Ref.</th>
<th>IOT building components</th>
<th>IOT Associated Devices</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Things</td>
<td>Sensors and Actuators</td>
<td>Information collection and communication is possible from the objects, without any human intervention. Dataflow becomes secure and manageable. It acts as an intermediate block and establishes strong connection between the things and cloud infrastructure. It provides control of data flow between things and cloud infrastructure.</td>
</tr>
<tr>
<td></td>
<td>Gateways</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

B. IOT Architecture

Architecture is a skeleton that encompasses physical components with underlined principles. Effective IOT Architectures ensures best, fast, and reliable convergence of information technology. Different researchers have been proposed different architectures. Among those three-layer architecture is the basic one which consists of three layers - Perception Layer, Network Layer and Application Layer. Table 3-6 presents layers and their functionalities of three-layer architecture, five-layer architecture, Middleware Architecture and Service Oriented based architecture.

TABLE 3: THREE LAYER ARCHITECTURE

<table>
<thead>
<tr>
<th>Layers</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perception Layer</td>
<td>It is the ground layer and it deals with sensors and actuators.</td>
</tr>
<tr>
<td>Network Layer</td>
<td>It deals with the transmission and processing of information.</td>
</tr>
<tr>
<td>Application Layer</td>
<td>It ties to facilitate user with the application specific services. It is responsible to define various applications in which deployment of Internet of Things could be done.</td>
</tr>
</tbody>
</table>

TABLE 4: FIVE LAYER ARCHITECTURE

<table>
<thead>
<tr>
<th>Layers</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perception Layer</td>
<td>It is the ground layer and it deals with sensors and actuators.</td>
</tr>
<tr>
<td>Transport Layer</td>
<td>It takes data from perception layer and transfers it to the processing layer through various mediums such as LAN, 3G, NFC, RFID and Blue tooth. It is responsible to take data from transport layer and then it stores, analyze and process huge amount of data. It provides services to subsequent layers and also manage services related to databases, big data and cloud computing.</td>
</tr>
<tr>
<td>Processing Layer</td>
<td>It manages complete system and flow of data.</td>
</tr>
<tr>
<td>Middleware Layer</td>
<td>It is responsible to give interface to the user.</td>
</tr>
<tr>
<td>Application Layer</td>
<td>It also presents final view of IOT.</td>
</tr>
</tbody>
</table>

TABLE 5: MIDDLEWARE ARCHITECTURE

<table>
<thead>
<tr>
<th>Layers</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perception, Access, and Edge Layer</td>
<td>These layers deal with actuators and sensors.</td>
</tr>
<tr>
<td>Backbone Network Layer</td>
<td>It presents virtualized plane which consist of cloud and servers.</td>
</tr>
<tr>
<td>Middleware Layer</td>
<td>It is responsible to take data from transport layer and then it stores, analyze and process huge amount of data. It provides services to subsequent layers and also manage services related to databases, big data and cloud computing.</td>
</tr>
<tr>
<td>Coordination and Application Layer</td>
<td>These layers present application plane which provides interface to the user.</td>
</tr>
</tbody>
</table>

TABLE 6: SERVICE ORIENTED BASED ARCHITECTURE

<table>
<thead>
<tr>
<th>Layers</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise Layer</td>
<td>It manages to communicate with business processes sub-layer for Application Integration. It also communicates with Service Discovery, Service Election and Service Orchestration Layer.</td>
</tr>
<tr>
<td>Service Discovery, Selection and Orchestration Layer</td>
<td>These layers communicate with IOT Services, Cloud of Things and Cloud.</td>
</tr>
</tbody>
</table>

C. IOT Taxonomy

IOT taxonomy is a way in which devices are composed to deliver their specific functionalities. In IOT taxonomy, we discuss those layers which are always supposed to be a part of IOT ecosystem. We begin our discussion with perception layer, which is composed of sensors and actuators, where sensors are responsible to collect data while actuators perform actions on that data. These sensors and actuators are further categorized as low-end, middle end and high-end devices. Sensors, actuators, and motes come under the category of low constrained IOT devices. Constrained technologies are deployed on data preprocessing layer. At this layer, there are some security features which filter the data before processing it further to the middleware.

IOT taxonomy defines IOT ecosystem and IOT ecosystem is divided in to six elements. Starting with IOT devices which exist in all layers of IOT architecture can be divided into open source and proprietary IOT devices. IOT devices have limited capability in terms of memory, power, and storage. IOT gateways mediate between sensing networks and high end IOT devices. Gateways are...
responsible for collection of data from different sensors and then send data for high level processing.

Due to low constrained nature, IOT devices require efficient communication protocol to establish a network of devices. IOT devices cannot connect directly to the internet through IP stack because IP stack requires more power and here low power technologies like WSN, Bluetooth, Zigbee and WIFI came into existence. Fig. 3 presents elements of IOT.

IV. MAJOR IOT OPERATING SYSTEMS

To make IOT application efficient, reliable, and scalable, an Operating system plays a vital role. IOT OS can be categorized as High-End and Low-End Operating Systems. High End OS operate on devices with single board systems for example: Raspberry pi and on the other hand Low End IOT OS acts as an interface for small board with constrained resources for example Arduino. High End and Low-End OS are further classified into Linux based and non-Linux based category. Fig. 4 presents categorization of IOT Operating System.

A. Contiki

It is best suitable for low constrained devices. It was created by Adam Dunkels in 2002 and it was released under license of BSD as an open-source software. It supports light weight preemptive scheduling, and it is considered as most appropriate OS for low constrained devices due to its TCP/IP stack. It is developed with a modular architecture and it is written in C language. Hardware devices that are low constrained in terms of memory, power and communication bandwidth mostly go for Contiki OS. Moreover this, Contiki OS works on three network stack protocols, the uIP TCP/IP stack, which provides IPV4 networking, the uIPv6 stack, which provides IPV6 networking, and the Rime stack, which is comprised of light weight networking protocols especially designed for low power wireless networks.

B. Tiny OS

It is mainly aim for low power devices operate in wireless sensor networks (WSNs), ubiquitous computing, building automation and smart meters. It is written in nesC, which is a dialect of C programming language. It is an open-source software, and it was released under the license of BSD. Tiny OS provides interface to common abstractions like packet communication, routers, sensors, actuators, and storage.

C. RIOT

With a focus on low constrained devices, RIOT can be considered as a good choice. RIOT is developed by FU Berlin. It has micro kernel architecture, and it is written in C and C++. Its programming model is of Hybrid nature and it supports 6LoWPAN and real time scheduling.

D. RIOT

It is based on microkernel architecture and it is a small operating system for low constrained IOT devices. It is developed by Intel subsidiary wind driver. Its programming model supports multithreading, preemptive and non-preemptive scheduling. It has been developed in C and C++ programming language. It supports Blue tooth Low Energy (BLE) 5.0 because it provides network stack support with multiple protocols.
E. **Mbed OS**

It is based on monolithic kernel and it provides support for preemptive scheduling. It has written in C and C++ programming language. It is developed by ARM with the focus on low-constrained devices. It supports multithreading, 6LoWPAN, BLE, WiFi, Near Field Communication (NFC) and Radio Frequency Identification (RFID). Recently, it is considered as an epicenter to IOT research and development due to its multifaceted features.

F. **Free RTOS**

It is based on microkernel architecture and it provides support for preemptive priority based and cooperative scheduler. Its programming model features multithreading, mutexes and semaphores. It is written in C language and it also possesses set of assembly functions. It is built in tickles idle implementation. It uses Idle task hook which allows power saving and due to this nature, it can be considered for low powered IOT devices.

V. **ARCHITECTURE TYPES AND FEATURES OF IOT OPERATING SYSTEMS**

Categorization of Architecture of IOT Operating System is defined as follows:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Architecture</th>
<th>Scheduler</th>
<th>Programming Model</th>
<th>Programming Language</th>
<th>IOT Devices</th>
<th>OS Type</th>
<th>Ram (KB)</th>
<th>Rom (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TINYOS</td>
<td>Monolithic</td>
<td>Non-preemptive</td>
<td>Event-driven concurrency</td>
<td>NesC</td>
<td>Low</td>
<td>Non-Linux</td>
<td>10</td>
<td>4–8</td>
</tr>
<tr>
<td>CONTIKI</td>
<td>Modular</td>
<td>Preemptive FIFO</td>
<td>Preemptive priority based</td>
<td>C</td>
<td>Low</td>
<td>Non-Linux</td>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td>RIOT</td>
<td>Microkernel</td>
<td>Preemptive priority based</td>
<td>Hybrid</td>
<td>C</td>
<td>Low</td>
<td>Non-Linux</td>
<td>1.5</td>
<td>5</td>
</tr>
<tr>
<td>FREERTOS</td>
<td>Microkernel</td>
<td>Preemptive priority based</td>
<td>Multiple threads, mutexes, semaphore</td>
<td>C and assembly functions</td>
<td>Low</td>
<td>Non-Linux</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>ZEPHYR</td>
<td>Monolithic</td>
<td>Non-preemptive priority and preemptive priority based and cooperative scheduling</td>
<td>Multithreading</td>
<td>C</td>
<td>Low</td>
<td>Non-Linux</td>
<td>2 to 8</td>
<td>50</td>
</tr>
<tr>
<td>MBED OS</td>
<td>Monolithic</td>
<td>preemptive</td>
<td>Multithreading</td>
<td>C and C++</td>
<td>Low</td>
<td>Non-Linux</td>
<td>4</td>
<td>16</td>
</tr>
</tbody>
</table>

VI. **NETWORK STACK ARCHITECTURE OF IOT OS**

A. **Zephyr**

Native Network stack consists of layers which provide specific support according to their own functionalities. These layers include:

**NETWORK APPLICATION:**
- This layer communicates with the provided application-level protocols for example CoAP, LWM2M, MQTT.
- This layer may access BSD Socket AI for network connection.
- This layer is responsible of data transmission and managing connections.
- It can also use network management API for the configuration of network and setting network link options.

**NETWORK PROTOCOLS:**
- Provide implementation of application-level network protocols like CoAP, LWM2M and MQTT etc.
- It also provides support for core network protocols like IPv6, IPv4, UDP, TCP etc.

**NETWORK INTERFACE ABSTRACTION:**
It provides support that is common for all network like setting network interface down.

**L2 NETWORK TECHNOLOGIES:**
It implements API that is responsible for data communication to and from a device. These include Ethernet, Bluetooth and CANBUS etc.

**NETWORK DEVICE DRIVERS:**
Transmission of data packets over the net is taken care by low level device drivers.
B. Contiki

NETWORK STACKS:
It is comprised of three network stacks.
1. IPv6
2. IPv4
3. Rime

NETWORK LAYERS:
There are four layers.
1. Network layer
2. MAC (Medium Access Control) layer
3. RDC (Radio Duty Cycling) Layer
4. Radio Layer

NETWORK LAYER:
It is comprised of upper IPv6 layer and the lower adaptation layer.

MAC layer:
It is the simplest layer in IOT/IP stack. In case of any traffic, it helps in avoid collisions. It monitors the medium before sending and holds its operations when someone else is sending. This layer depends on RDC layer.

RDC (Radio Duty Cycling):
It provides facility of energy saving by keeping its radio transceiver off.

Contiki supports three cycling mechanisms: Contiki MAC, X-MAC, LPP (Low-Power Probing). These mechanisms are based on the principles of low power consumption and better power efficiency.

RADIO LAYER:
It is the ground layer in the Contiki Net Stack. Here interrupt handlers are responsible to fetch data. The input data is read into the buffer and then polling process sent the data to upper layers.

C. Mbed OS

Network Stack of Mbed Os is divided in to three layers:
- Transport Layer
- Network Layer
- Data link Layer

All IP connectivity methods shares Socket API. Socket API provides portability among various connectivity methods. Socket API relates to the transport layer and it supports TCP and UDP protocols. Network Stack layer supports IPV4 and IPV6. Data link relates to Network driver and it supports Ethernet, WiFi, Cellular and IEEE 802.15.4 drivers.

D. RIOT

It is very flexible as any network can be integrated in its network architecture. It provides two interfaces: Application programming interface and the device driver API netdev [26]. Network stack architecture of RIOT is composed of six layers. Network layer is loosely coupled with the hardware layer. Communication between Network Layer and hardware layer is established by means of netdev API. Application layer communicates with the network stack by means of Sock API. Separate thread is assigned to each device driver. Driver layer provides implementation of radio devices.
E. Tiny OS

Prime component of Tiny OS in communication is Active message. Active message is an extensively used protocol in parallel computing where each message consists of an identifier which is user handler name. The handler function is responsible to invoke on the target node using user level handler name to pass the active message. This mechanism give rise to event driven communication between nodes.

F. Free OS

It relies on third party tools to implement MAC layer. For example, IoT LAB. In addition to this, TCP and UDP operate as transport protocols. Free RTOS UDP Protocol is mainly aim for low constrained devices as it provides socket with compact code size. On the other hand, Free RTOS TCP is based on open-source stack which provides Ethernet based stack. TCP/IP protocol is best suitable for low constrained devices because it is based on IwIP. IwIP is mostly used in embedded systems where less resource usage is in focus. Network stack implementation of IwIP is mainly aim for systems which have 10kb of RAM and 40kb of ROM.

VII. RELATED WORK

By the passage of time Operating Systems for low constrained devices have emerged as predominant Operating Systems. This section discusses literature of previous studies and presents comparative analysis of this study with the existing surveys. Table 9 presents comparative analysis of this study with the existing surveys.

VIII. OPEN RESEARCH ISSUES AND RECOMMENDATIONS

A. Small Memory Footprint

To enhance the functionalities of low constrained devices more research is required in the area of small memory footprint. IOT devices operate on minimum RAM and ROM which contains few kilobytes. Hence, the core characteristic of low constrained IOT devices is to minimize the code size to ensure minimum memory utilization.

B. Less Energy Consumption

Sustenance in battery life of IOT devices is a key factor to reduce energy consumption. Designing of more efficient network protocols is highly required to ensure prolong battery life. In addition to this, more effort is required to manage hardware features in such an efficient manner which could result in less power consumption.

C. Reliability of IOT Devices

Another research direction led us to the reliability factor of IOT devices. While designing IOT complex architectures, OS reliability should be ensured by focusing on micro kernel architecture, memory protection units and static code analysis etc.

D. Scheduling Model

Sometimes scheduling model hit processor’s performance by affecting system’s energy and efficiency. Constraints during scheduling, affect processor’s performance and due to this reason IOT Operating Systems are not able to perform up to the mark.

E. Runtime Behavior of RTOS

Extensive research is required to deal with complex run time behavior of RTOS task. In a complex IOT system, proper task priorities and processor shared timings are required. Hence, while implementing RTOS special care should be taken because in case of RTOS predicting real time behavior is very difficult. Tasks may get failed or may result in delay during execution and this low point of RTOS Operating System cannot be tolerated while implementing time critical applications.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IOT OVERVIEW</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
</tr>
<tr>
<td>ARCHITECTURE</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
</tr>
<tr>
<td>TAXONOMY</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
</tr>
<tr>
<td>KEY CHARACTERISTICS AND FEATURES</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
</tr>
<tr>
<td>IOT DEVICES</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
</tr>
<tr>
<td>PROGRAMMING MODEL</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
</tr>
<tr>
<td>SCHEDULING MODEL</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
</tr>
<tr>
<td>MEMORY</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
</tr>
<tr>
<td>NETWORK STACK ARCHITECTURES</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
<td>TCZ</td>
</tr>
</tbody>
</table>

C: Contiki, T: TinyOS, F: FreeRTOS, R: RIOT, Z: Zephyr, M: MbedOS.
TABLE 10: CONTRIBUTIONS AND FUTURE DIRECTIONS OF EXISTING SURVEYS

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Discussed IOT Operating Systems</th>
<th>Discussed aspects</th>
<th>Future directions</th>
</tr>
</thead>
<tbody>
<tr>
<td>[6]</td>
<td>TinyOS, Contiki, RIOT, Zephyr, MbedOS and Brillo</td>
<td>In this paper, overview of different IOT Operating systems, supported hardware, and future research directions are presented. Moreover this, this study provides overview of the previous literature papers in Special Issue on IOT OS management: opportunities, challenges, and solution. Finally, this study concluded the whole survey.</td>
<td>• Future work should be focused on efficient techniques to acquire the acute motes of synchronization. • RDC (Radio Duty Cycling) can be another direction of research along with motes synchronization to work on. • More work is needed to achieve accuracy in execution of critical tasks of the IOT motes to enhance real time capabilities. • Critical systems such as health care, smart home, smart city is the flavor of time and their security is still a question mark. • Efficient mechanism is required to utilize the minimum memory. • Reliability of IOT devices requires more research. • Real time operating systems and execution of tasks without delay in real time is a great challenge. • Scheduling model constraints, limitations in network buffer management and programming model could be considered as future research direction. • More operating systems need to be explored for future research such as Zephyr, RIOT, Mbed OS.</td>
</tr>
<tr>
<td>[7]</td>
<td>Contiki, TinyOS, and FreeRTOS,</td>
<td>Different aspects of resource management including process management, memory management, energy management, communication management, and file management are investigated, and their advantages and disadvantages are presented.</td>
<td>• IOT operating Systems are still deficient in context of security. • Reduction in Operating System services is required in light Kernel architecture and major focus should be drawn towards small memory footprint. • Execution of tasks and real time compatibility should be equated in order to achieve accuracy. • IOT operating Systems are still deficient in context of security. • Reliable Communication, Bandwidth, Interoperability need to be addressed in detail for future challenges. • Small memory foot print needs to be addressed to facilitate low constrained IOT devices.</td>
</tr>
<tr>
<td>[8]</td>
<td>FreeRTOS, Mbed, Contiki, TinyOS and RIOT</td>
<td>This study is mainly focused on comparative analysis of the most recent IOT operating systems for low constrained IOT devices. This study discussed architecture, scheduling, real-time capabilities, programming model, memory footprint, network connectivity, hardware support and energy efficiency. This study addressed IOT operating Systems constraints with respect to their architecture, programming model, scheduler algorithms, networking and communication protocols. This study presented requirements and shortages in development. In addition to this, it also contributed towards summary of related work and detailed case studies are also illustrated.</td>
<td>• Security and Privacy are still open to research. • Significant research is required to stable communication protocols. • Security concerns need to be monitored such as bug fixes or updates and hardware constraints. • Small memory foot print is a big challenge to address while proposing new architecture for IOT. • Challenges in real time operating systems defines new direction for research. • Execution of tasks and real time compatibility should be equated in order to achieve accuracy. • Scheduling model constraints, limitations in network buffer management and programming model could be considered as future research direction. • IOT operating Systems are still deficient in context of security.</td>
</tr>
<tr>
<td>[9]</td>
<td>Contiki, TinyOS, RIOT, Nano-RK, LiteOS, MantisOS,ROS OS, RETOS</td>
<td>This study identified key parameters that needs to be focused on while selecting open-source project. Number of selected studies have been discussed in this survey to explore open-source system software projects and frameworks.</td>
<td>• IOT operating Systems are still deficient in context of security. • Reliable Communication, Bandwidth, Interoperability need to be addressed in detail for future challenges. • Small memory foot print needs to be addressed to facilitate low constrained IOT devices.</td>
</tr>
<tr>
<td>[10]</td>
<td>Android Things, Mbed OS, Contiki OS, RIOT OS, Zephyr</td>
<td>This study identifies key parameters that needs to be focused on while selecting open-source project. Number of selected studies have been discussed in this survey to explore open-source system software projects and frameworks.</td>
<td>• IOT operating Systems are still deficient in context of security. • Reliable Communication, Bandwidth, Interoperability need to be addressed in detail for future challenges. • Small memory foot print needs to be addressed to facilitate low constrained IOT devices.</td>
</tr>
<tr>
<td>[11]</td>
<td>Tiny OS, Contiki OS, FreeRTOS, and RIOT.</td>
<td>This study contributes in the dimensions of over view and evolution of IOT. Architectures of major IOT Operating systems are discussed. In addition to this, challenges of IOT Operating Systems and open research issues are discussed extensively.</td>
<td>• IOT operating Systems are still deficient in context of security. • Reliable Communication, Bandwidth, Interoperability need to be addressed in detail for future challenges. • Small memory foot print needs to be addressed to facilitate low constrained IOT devices.</td>
</tr>
<tr>
<td>[12]</td>
<td>Contiki TinyOS mbed OS Real Time Operating System (RTOS)</td>
<td>A comprehensive overview of IOT OS was discussed in this paper. Memory, Programmability Support, Network Protocols, Architecture, Scheduling, Modularity Support is discussed in view of major OS.</td>
<td>• IOT operating Systems are still deficient in context of security. • Reliable Communication, Bandwidth, Interoperability need to be addressed in detail for future challenges. • Small memory foot print needs to be addressed to facilitate low constrained IOT devices.</td>
</tr>
<tr>
<td>[13]</td>
<td>Contiki, RIOT, TinyOS, LiteOS, FreeRTOS, Mantis OS, Nano-RK, SOS, NutOS, uC/OS-III, uC/OS</td>
<td>This explanatory paper presents content on prevalent IOT operating systems. This study did comprehensive analysis and illustrate findings for future studies.</td>
<td>• IOT ecosystem is established with heterogeneous devices that work together and allow inter device communication. More research is required for heterogeneity among these devices. • Security and Privacy are still open to research. • Scheduling model constraints, limitations in network buffer management and programming model could be considered as future research direction. • IOT operating Systems are still deficient in context of security.</td>
</tr>
<tr>
<td>[14]</td>
<td>TinyOS, Contiki, RIOT, LiteOS, FreeRTOS, Mynext, uClinux, Raspbian, Android thing</td>
<td>This study serves to elucidate taxonomy of IOT ecosystem. Many technical aspects are illustrated such as architectures, devices, communication protocols and network stack architectures.</td>
<td>N/A</td>
</tr>
<tr>
<td>[15]</td>
<td>Contiki, RIOT, FreeRTOS .TinyOS, OpenWSN nuttx , eCos uClinux ,ChibiOS/RT CoOS, nanoRK,Nat/OS</td>
<td>Contribution of this study lies in exegetic discussions on specific requirements that should be fulfilled by an OS to cater low constrained devices. Several tradeoffs have been discussed regarding the constrains of IOT and hardware platforms.</td>
<td>N/A</td>
</tr>
</tbody>
</table>
IX. CONCLUSION

This paper provides an overview of IOT and its building components. This paper gives insights into architecture, programming model, schedular and network stack architecture of different IOT Operating Systems. The contributions are multifaceted. Firstly, we discussed contribution to this survey followed by the structure of this survey. Peer review papers are presented along with year of publication and authors. Secondly, an overview of IOT is presented along with IOT building components. IOT architecture is explained along with layers and its functionalities. In this section we discussed IOT elements and major IOT Operating Systems for low constrained devices with their key characteristics. Network Stack architecture of IOT OS is also given in this section. After this, we discussed related work and compare our work with the previous surveys. Finally, critical research areas are unfolded to facilitate future research studies in this domain.

ACKNOWLEDGMENT

The author is grateful to her instructor Dr. Muhammad Iqbal, Assistant Professor, Department of Computer Science, Bahria University Karachi, Pakistan for his immense support throughout this study.

REFERENCES


Sumera Rouana received BS degree in Software Engineering from UBIT, Karachi University, Pakistan. Currently, she is pursuing MS in Computer Science from Bahria University, Karachi Campus, Pakistan. Her research interests include IOT Operating Systems in Low end devices, Detection of patterns using Machine Learning, Deep Learning, Natural Language Processing and extraction and analysis of Data using Data Science.

Dr. Muhammad Iqbal is a Senior Assistant Professor and Cluster Head of Department of Computer Science, Bahria University, Karachi Campus, Pakistan. He received Ph.D. degree from South West Jiao tong University, Chengdu, China (SWJTU). He has 14 international peer-reviewed publications on his credit.