MEST Journal
DOI 10.12709/mest.08.08.02.06

TOP SEVEN IoT OPERATING SYSTEMS IN MID-2020

Zoran Cekerevac
Faculty of Business and Law, "Union – Nikola Tesla" University, Belgrade, Serbia

Zdenek Dvorak
University of Zilina, Faculty of Security Engineering, Zilina, Slovak Republic

Tamara Pecnik
Faculty of Business and Law, "Union – Nikola Tesla" University, Belgrade, Serbia

JEL Category: **L8, O33**

Abstract

The Internet of Things (IoT) is the close future of the Internet. In the introductory part of the paper, IoT and 5G networks are analyzed, and specifications of operating systems (OS) suitable for IoT are defined, as well as advantages and disadvantages of using Wi-Fi technology within IoT systems. During the preparation of the paper, 56 different IoT operating systems were analyzed according to different criteria, and, in Chapter 2, lists of the seven most popular IoT OS and seven operating systems suitable for IoT devices with small memories were created. The following is a selection of the most suitable operating systems in mid-2020. The ranking was performed based on similar ranking lists from the literature, based on the comments of the users of these operating systems, and the authors' analysis based on twelve additional criteria. Based on the obtained results of the analysis, it was shown that Contiki-NG, Riot OS, TinyOS, Raspberry Pi OS, Ubuntu Core, Zephyr IoT OS, and Windows 10 for IoT showed the best results. Each of the ranked OSs in the continuation of the paper is considered from the aspects of basic characteristics, architecture, basic pros, and cons, as well as the possibility of application, listing significant practical solutions. The third part of the paper presents a checklist that should be followed when choosing an OS for an IoT solution. Chapter 4 provides conclusions that can be drawn based on the performed analyzes and the presented work.

Keywords: *Internet of things, networks 5G, Contiki, RIOT, TinyOS, Raspberry Pi, Ubuntu Core, Ubuntu Enterprise, Zephyr IoT, Windows 10 for IoT*

Address of the corresponding author:
Zoran Cekerevac
zoran@cekerevac.eu

1 INTRODUCTION

Since the Internet of Things (IoT) involves many different physical devices connected to the Internet, and tasks related to data collection and/or sharing may occur, it is clear how much the

need for different software will expand. Also, certainly, the application of IoT to analyze a huge amount of related sensor data in the cloud will often require programs that use machine learning algorithms. IoT was initially mostly used for business and production, but now it can be found more and more often in people's homes and offices, with the prospect of mass application in smart cities and traffic and transport of goods. The creation of small super cheap computer chips has made it possible to turn any physical device into a part of the IoT. By connecting physical objects and adding sensors to them, a new level of digital intelligence is obtained. It is predicted that by 2025, there will be 41.6 billion connected IoT devices. (Ranger, 2020) Security is one of the biggest problems of the IoT. Sensors located on the devices collect information e.g. following what is said or done in the environment where the IoT devices are located. Danny Palmer's research (2017) showed that back in 2017, it was possible to easily hack over 100,000 webcams. There is no reason to believe that the situation is now more favorable. Hacking IoT devices in the real world can have unforeseeable consequences if someone successfully hacks the sensors of nuclear power plants or self-propelled vehicles in traffic. Areas in which IoT has the greatest application so far are smart manufacturing, smart cities, connected and smart logistics, smart homes, health systems, public sector…

1.1 IoT and 5G networks

The first step required for the realization of the concept of "Internet of Things" is the introduction of 5G technology. Each new generation of wireless technology has led to faster and more reliable networks. The evolution of 5G networks is shown in Figure 1. In the 1980s, first-generation technology-enabled communication over a mobile phone. The next generation, 2G, enabled more efficient and secure phone calls, and introduced text messaging. 3G has brought a new era of smartphones, and the 4G era has brought new speeds. 5G is a fifth-generation mobile network based on a combination of wireless technologies such as GSM, Wi-Fi, LTE. 5G allows one to connect almost everything and anything, including machines, objects, and devices. At the same time, the 5G network will enable faster data download and transfer, the easier flow of online content, higher quality voice and video calls, and more reliable mobile connections.

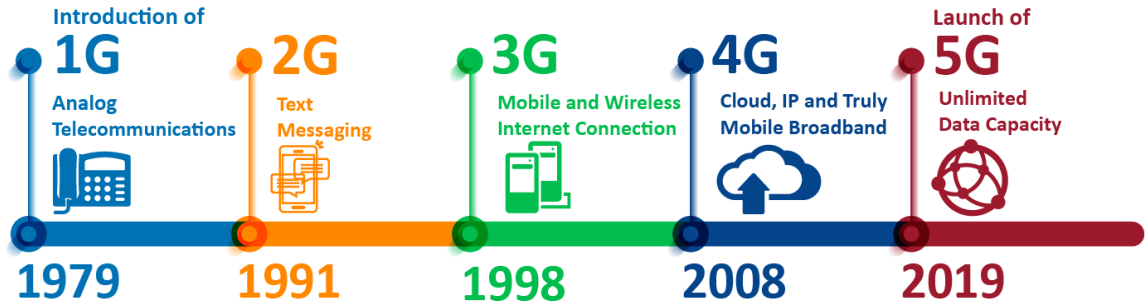

Fig. 1 The evolution of 5G

Source: (OKportal Technology, 2019)

Previous smartphones and other electronic devices used a frequency range from 30KHz (1G) to 8GHz (4G). But with the increase in the number of mobile phones and other devices that share this range, the speed of data transfer has decreased over time. The introduction of millimeter waves (1 to 10 mm) in the 5G network can achieve frequencies between 3 and 300 GHz. (Johansen, 2016) Frequency bands for 5G networks are grouped into two bands FR1 (<6GHz) and FR2 (24.25 - 52.6 GHz). (Craven, 2020). Until now, this technology has been used only by radar systems and satellites.

The main advantages of 5G technology are (Tutorialspoint, 2020) (Pecnik, 2020):

- High resolution and a high-bandwidth two-way connection
- Ability to gather all networks on one platform
- Effective and efficient
- Short response time
- Ability to transfer data at the Gigabit level in more than 60,000 connections.
- Easy management of previous generations of networks.
- Possibility of various services including private networks.

- Possibility of uniform, uninterrupted, and continuous connection around the world.
- Since millimeter waves have high frequencies and small wavelengths, the antennas used for them are smaller, so there is a possibility to build small base stations.
- 5G network can work based on "Massive MIMO" technology.

However, this technology also has its drawbacks (Tutorialspoint, 2020) (Pecnik, 2020):

- The waves are very short, and to "catch" the signal, it is necessary to be close to the cell, that is, the node that emits the signal.
- This technology is not able to pass through the walls of buildings, and it can be disturbed by rain.
- MIMO equipment contains two or four antennas. Installing many antennas is very convenient because in that way we can send and receive data through all of them at the same time. But to achieve Massive MIMO, the number of antennas must exceed the number of users.
- The expected speed seems difficult to achieve due to the level of technological support in most of the world.
- Old devices will likely need to be replaced with new ones, which is expensive.
- Huge investments in infrastructure development are needed.
- 5G technology is still in its infancy and its sustainability is still being explored.
- Privacy and security issues have yet to be addressed.

One of the main disadvantages of the 5G network is coverage. One node that emits a 5G network signal covers a significantly smaller area than a tower that emits a 4G network, which means that the infrastructure of a 5G network requires significantly more nodes that will emit a signal compared to the existing network. Another problem is that only a few phone and tablet models have built-in 5G modems that allow them to use the network. Older phone models that do not have a built-in 5G modem will not be able to use it. The 5G network puts a lot of strain on the processors, which causes the phone to heat up.

All the above significantly affects the creation of operating systems that can be or will be applied in IoT networks.

1.2 IoT Operating Systems' Specifications

Due to the characteristics of IoT devices, the operating systems (OS) provided for their operation differ from other existing operating systems. IoT operating systems are designed to perform tasks within constraints that are specific to IoT devices, such as e.g. limited memory, size, and power and processing capacity. IoT OS must support different hardware architectures and devices. While tablets, smartphones offer gigabytes or terabytes of memory, IoT devices have several kilobytes of memory available. That is why these OSs are essentially oriented towards data transfer via the Internet.

IoT OSs manage systems in smart cars, traffic lights, and streetlights, smart TVs, ATMs, elevators, smart meters, etc., so they should provide security and privacy for the entire IoT network. There should be established mechanisms for authentication, data integrity … Verifications that may be required for IoT OS are DO-178B for aircraft systems, IEC 61508 for industrial control systems, ISO 62304 for medical devices, or SIL3/SIL4 IEC for transport and nuclear systems. Most IoT OSs are open source as TinyOS, RIOT, Contiki, Mantis OS, LiteOS, Apache Mynewt, Zephyr OS, Ubuntu Core, Raspberry Pi OS, and others. When it comes to closed IoT OS some of the most popular are: Android Things, Windows 10 IoT, Micro Digital SMX RTOS, TI RTOS, Freescale MQX… (Mishra, 2019)

The parameters to pay attention to when choosing a suitable IoT OS are:

- **Small memory footprint** – the smaller the memory footprint of the OS the more favorable the OS.
- **Ability to work in real-time** – If the OS enables real-time operation, it is more favorable for the purposes that require it.
- **Energy-efficient operation** – If the OS does not require high energy consumption for its operation, it is advantageous because it saves battery maintenance and makes maintenance cheaper.
- **Portability** – the so-called hardware-agnostic work. It is advantageous if the OS can be transferred to different hardware platforms

and interfaces in BSP format in a standard way, such as. using POSIX calls.
- **Modularity** – The OS must have a kernel-core. All other functionalities can be added later if the application requires it.
- **Network connectivity and protocol support** – For IoT, the continuous connection of the device to the network is crucial, so the OS needs to support various connectivity protocols, such as Ethernet, Wi-Fi, BLE (Bluetooth Low Energy), IEEE 802.15.4 and many others.
- **Flexibility** – The OS must be scalable for any type of device. This means that developers and integrators need to know only one OS, which will be used for both nodes and gateways.
- **Reliability** – This is especially important because the devices are often in remote locations and have to work without error for years. Reliability means that the OS needs to meet all verifications for specific applications.
- **Security** – The OS has add-ons that meet security expectations, ensuring that the device is booted securely, using SSL support, components, and encryption drivers.
- **Eco-system and application development capability** are essential features of the OS, so OSs that come with development tools facilitate the development and implementation of IoT applications.

Gateways are particularly important components of IoT. They represent a bridge between real devices and the virtual IT environment. Therefore, requests for support at the gateway level are set before the OS (IoT Operating Systems, 2016):

- **Protocol support** – In addition to HTTP protocols, the OS should also support small data transfer protocols such as CoAP, MQTT, or UDP.
- **Data collection, local processing, and storage** – The OS should support real-time decision-making processes and reduce the amount of data going to the cloud (e.g. in the case of building management systems or video surveillance).
- **Security** – The OS should provide protection at the hardware and network level, provide crypto security, SSL / TLS certificate management, user authentication, VPN connectivity, firewalls, and a list of secure applications.
- **Reliable communication with IoT platforms in the cloud** and
- **Terminal Management** – The OS should enable remote upgrades and two-way communication between devices and the cloud.

The choice of an operating system is an extremely important step in creating an IoT-based system because it can ensure system performance, but it can also degrade system characteristics.

1.3 Wi-Fi and OS selection from a company perspective

Before choosing the OS, it is necessary to define the concept of the IoT network within the company. Wi-Fi provides many benefits that include the absence of cables and everything that goes with it, easy installation, a lot of cheap Wi-Fi sensors, and enough professionals who can support the system and the company's existing Wi-Fi infrastructure, but with the necessary peripheral power devices with electricity, also brings problems for network configuration. Many different devices can have different requirements for the network and OS. With successful IoT solutions that work well, problems can arise in the process, hardware, or firmware and it is good if the applied solutions are reliable because it is possible to focus on a narrower area and solve the problem more efficiently. If, in addition to these three different areas of problem research, the OS also appears as a possible cause of a problem, the number of variants in solving the problem increases many times over. Unfortunately, the company's Wi-Fi network becomes unreliable and inconsistent when an IoT solution is included. The reasons are:

- Each network is configured according to its previous needs and is not adapted to IoT.
- Network administrators and local IT teams cover certain tasks, but when hundreds and thousands of new devices are connected, and when new experts are not added to local people, problems are inevitable. The IoT may contribute to the improvement of the company's business, but it will burden the people, so they will not accept the change with enthusiasm.

- Lack of end-to-end control. Any, even small change in the local Wi-Fi environment can jeopardize the entire IoT solution.

These reasons say that using Wi-Fi as a backbone for an IoT solution is too risky and expensive. Whenever possible, it is far more convenient to set up the company's own network, such as a private LoRa network. This can provide greater reliability and consistency, maintain end-to-end control, and solve potential problems more efficiently. (Leverege, 2020).

There is no one solution and one operating system that will be suitable for all situations. Therefore, attention should be paid to the choice of OS. The needs of IoT system applications should be identified and based on them, technical requirements for the OS should be set.

2 SELECTION OF THE TOP SEVEN IoT OS

In preparing this paper, 147 operating systems that can be used in IoT were considered. Also, the literature dealing with their evaluation and ranking was considered. Based on a review of the literature, it was determined that this area is developing rapidly so that the same authors at different times give preference to a different OS. Some high-ranking operating systems in 2019 dropped out of the rankings in 2020, while some new ones appeared and were highly positioned. (Okoi, 2019) vs. (Okoi, 2020) Normally, it should be borne in mind that different rankings are made for different purposes within the IoT. The author's experience shows that an increasing number of investors demand software based on open-source software and that the impact of open-source operating systems is growing steadily.

Table 1 Analyzed operating systems

No.	IoT operating system
1	Alpine Linux
2	Amazon FreeRTOS
3	Android Things
4	Apache Mynewt
5	Arch Linux ARM
6	ARM Mbed OS
7	Balena Yocto Linux
8	Batocera.linux
9	Bedrock Linux
10	BMC64
11	Brillo
12	Chromium OS
13	Contiki
14	Device OS
15	Devuan GNU
16	DietPi
17	Domoticz
18	Embedded Linux
19	FreeBSD

No.	IoT operating system
20	Fuchsia
21	Gentoo
22	Huawei LightOS
23	Kali Linux
24	Kano
25	Lakka
26	LibreELEC
27	Linutop
28	Micrium uC/OS
29	Minibian
30	Mongoose OS
31	Nano RK IoT
32	Nucleus RTOS
33	NuttX Real-Time
34	OpenELEC
35	OpenMediaVault
36	OpenSUSE
37	OS-IoT
38	OSMC

No.	IoT operating system
39	Particle Device OS
40	Raspbian
41	RaspBSD
42	RetroPie
43	Riot OS
44	RISC OS
45	Rokos
46	SARPi
47	Siemens MindSphere
48	Snappy
49	TinyOS
50	TizenRT
51	Ubuntu Core
52	Ubuntu Mate
53	VxWorks 653
54	Wind River VxWorks
55	Windows 10 for IoT
56	Zephyr

Based on this research and papers (Mehedi, 2020), (Hawlader, 2020), (Okoi, 2019), (Okoi, 2020), (Dutta, 2020), (Kumar, 2018), (g2, 2020), (Mishra, 2019), (Pecnik, 2020) 56 operating systems were singled out. The ranks achieved by IoT operating systems in different ranking lists in this paper are weighted in accordance with Table 2 so that the first place on a ranking list brought 30 points, and the eleventh and lower places brought 2 points.

Table 2 Operating system ranking weighting

Rang	1	2	3	4	5	6	7	8	9	10	11
Weight	30	25	20	16	14	12	10	8	6	4	2

The rank of the operating system in each ranking list was multiplied by the appropriate weight, and then the obtained values were added up. The obtained results were then sorted from the largest sum to the smallest sum.

If the OS was chosen based on the popularity of operating systems, the results shown in Table 3 would be obtained. Linux operating systems dominate in number and popularity, and far behind are BSD type and Solaris operating systems, as well as some third-party operating systems (KolibriOS and KISS).

If the OSs were ranked by size from smallest to largest, the results would be as in Table 4.

Table 3 Ranking of the seven most popular operating systems suitable for IoT

Rank	Title	Based on	Architecture	Image size (MB)	Journaled File System
1	MX Linux	Debian, antiX	i686, x86_64	1500-1700	ext3, ext4, JFS, ReiserFS, XFS
2	Manjaro Linux	Arch	aarch64, x86_64	2600-3000	Btrfs, ext3, ext4, JFS, ReiserFS, XFS
3	Mint Linux	Debian, Ubuntu	i686, x86_64	1800-2000	Btrfs, ext3, ext4, JFS, ReiserFS, XFS
4	Ubuntu	Debian	armhf, i686, powerpc, ppc64el, s390x, x86_64	800-2400	Btrfs, ext3, ext4, JFS, ReiserFS, XFS
5	Debian	Independent	aarch64, armel, armhf, i386, i686, mips, mipsel, ppc64el, s390x, x86_64	300-3700	Btrfs, ext3, ext4, JFS, ReiserFS, XFS
6	Solus	Independent	x86_64	1600-1900	
7	Fedora	Independent	aarch64, armhfp, x86_64	500-1900	Btrfs, ext3, ext4, XFS

Source: (DistroWatch.com, 2020)

Table 4 Seven top-ranked operating systems suitable for IoT devices with small memories

Rank	Title	Based on	Architecture	Image size (MB)	Journaled File System
1	Bedrock Linux	Independent	aarch64, armv7, mips64, ppc64, s390x, x86_64	1 - 2	
2	MLL - Minimal Linux	Independent	i386, x86_64	10 - 18	ext3, ext4
3	Tiny Core Linux	Independent	I486, x86_64	14 - 206	
4	4MLinux	Independent	x86_64	14 - 952	
5	IPFire	Independent	armv5tel, i586, x86_64	200 - 300	
6	Fugulta	OpenBSD	aarch64, i386, x86_64	200 - 300	
7	Star	Debian	x86_64	200 - 600	Btrfs, ext3, ext4

Source: (DistroWatch.com, 2020)

Why it is difficult to choose the best OS for IoT?

When choosing the optimal operating system for a project, the designer faces several challenges.

1. The first criterion is whether the investor makes a request regarding the type of operating system, i.e. whether the software must be open source. If it is not demanded, the designer has more options at his disposal, which may or may not be an advantage.
2. The next criterion is whether the investor is willing to pay for an operating system license or requests a free operating system. Even if the investor wants a free operating system, a satisfactory solution can usually be found.
3. Many different operating systems is available, and it is unlikely that the designer knows them all well. That is why it is often the case that a designer must seek the advice of his colleagues or search for literature. An aggravating circumstance is that in the initial phase of design, the designer does not have fully defined technical requirements and does not know what he needs. In the literature, all

operating systems are relatively well described, in detail, but very often data are not given in the form that could be used for comparison. User comments are not of great help here either. The reasons are that different users have different expectations, so they can be both satisfied and dissatisfied, but also falsely satisfied and falsely dissatisfied, depending on their experiences. By analyzing 92 groups of responses with a total of 12,312 responses, we observed that the most popular operating systems with the most reviews usually have lower average scores than less popular programs. Less popular programs have fewer reviews, so the impact of each review on the result is significant. Therefore, in the analysis, we, as a rule, avoided grades with less than 10 reviewers. Another reason for lower ratings of popular operating systems is the degree to which user expectations are met. Many users very often means many different expectations, and it is possible that a great operating system is not great in all its features. This role can be played by seemingly insignificant features, from the method of installation to the multilingualism of the system and the quality of translation into certain languages. In addition, it is possible that reviewers have their favorites.

4. Software updates are a very significant factor. Operating systems are usually updated several times a year, and it is rare that an OS has not been updated for more than a year. Each update introduces a new disorder in the analysis, because very often errors noticed in the time between two versions of the software are corrected (which does not mean that no new errors are made). In this way, part of the reviews from the previous period becomes unusable.
5. For a particular application in IoT devices, one of the key factors may be the customer category. As the factors, they are the processor and memory capacities that the device has, the way of storing the operating system (RAM, SSD, Live medium, SD card,…), as well as the way of transferring data and receiving commands from the system (wire, Bluetooth, Wi-Fi ...). It is not the same whether it is an intelligent thermometer, an intelligent controller, an intelligent data acquisition unit, or a control unit, desktop computer, server, or Raspberry Pi-type computer. It is clear that the appropriate operating system should be chosen for each device while ensuring interoperability.
6. When choosing the OS, one should also keep in mind the market for which the installation is being prepared, because the habits of the users are not the same everywhere. In addition to user habits, the choice of OS is also influenced by the architecture of the desired system. Not all OSes are adapted to all possible architectures. Investors often have a request for certain systems to be modernized, which means that they also have older equipment. So the latest solution does not have to be a feasible solution. Normally, old equipment can be replaced with new ones, but this is not always economically viable.
7. The seventh, but one of the most important parameters that need to be analyzed is the protection of the system with the possible provision of privacy and, sometimes, anonymity. In this analysis, we will not consider the physical protection of the system, because it is implied, but we will keep in mind VPN, crypto protection, firewalls, and other types of software protection. Security is probably the biggest challenge for the IoT. Sensors collect a lot of sensitive data, and previous systems do not guarantee their protection. There are too many devices that don't even think about security. Even in mass-used software, security vulnerabilities are detected on a daily basis, and with many IoT devices, it is not even possible to update the software and enter security patches. Currently, routers and webcams are the most vulnerable, and smart children's watches connected to the Internet allow hackers to monitor the user's location, eavesdrop on conversations, and even communicate with the user. (Ranger, 2020). These problems are undoubtedly significant at the level of individual users, but the problems are increasing and more complex with the growth of the number of users. Connecting industrial plants to IoT networks increases the risk of hackers discovering and attacking devices. The readiness of hackers for such attacks is visible, but the readiness of companies to protect their devices and data is less visible.

No major involvement of companies in the field of IoT security planning has been observed. The prevailing impression is that many want to take advantage of the IoT, leaving security to be solved for later.

A special subgroup of protection problems is privacy. There is a real danger of a complete loss of privacy in the IoT sea. The one who has access to the data from the smart house also has data about each member of the household, from what he eats, what he drinks, when he sleeps, when he wakes up, who comes to visit him, all the way to who passes by the house. Virtually all users become participants in the reality show. It is not certain that the service sellers will give the collected data to someone, but it is likely that some of them will. By collecting seemingly harmless data such as electricity consumption, room temperature and humidity, and CO and CO2 concentration during the day, it is possible to determine what someone has for dinner. (Ranger, 2020).

Old, used devices also pose a great security risk. Many of them keep data in memory, so users must have a developed strategy for destroying used devices. Just because a user thinks the device is broken and he cannot retrieve data does not mean no one else can retrieve the data.

In addition to the analysis of IoT OS ranking based on the literature, twelve additional criteria shown in Table 5 were used in the analysis of the selection of the best IoT OS.

Table 5 Criteria for IoT OS assessment

No.	Criterion
1	Hardware agnostic operation
2	Energy efficient operation
3	Real-time capabilities
4	Minimum RAM
5	Minimum ROM
6	OS Footprint
7	Network connectivity and Protocol Support
8	Security
9	Programming model
10	Price and Market target
11	Way of installation
12	Maintenance

For each of the criteria, the observed OS was ranked, and then the ranks for all criteria were added up. In cases where two (or more) systems achieved the same result, everyone with the same result was assigned the most favorable rank, and the next lower-ranked OS was given a rank according to its actual position on the ranking list.

Based on the obtained results, seven "best" IoT OS were selected and ranked in Table 6.

Table 6 Ranking list of the seven best ranked IoT operating systems

Rank	Title	Score
1	Contiki	351
2	Riot OS	292
3	TinyOS	251
4	Raspbian	236
5	Ubuntu Core	202
5	Zephyr	199
7	Windows 10 for IoT	199

2.1 Contiki-NG

Contiki-NG is an operating system that is primarily oriented to small IoT devices with limited memory and power. It is the first operating system to enable IP communication using the uIP TCP/IP protocol stack. Thanks to that, with its minimalist design and wide possibilities of application, it gained great popularity.

Contiki-NG contains an RFC compatible IPv6 low-power communication package, which enables Internet connection. The system runs on various platforms based on energy efficient architectures such as ARM Cortek-M3/M4 and Texas Instruments MSP430. The code print is 100 kB in size, and memory usage can be reduced to 10 kB (Duquennoy, 2019). The source code is available as open source with a 3-clause BSD license (Tsiftes, 2017).

Although there are many similar operating systems like TinyOS, what sets Contiki apart from the competition is the complexity and flexibility it offers developers. It provides functionality for managing programs, processes, resources, memory, and communication, and to run Contiki OS only a few kilobytes are needed. A minimum 2kB of RAM and 30 kilobytes of ROM are required (RIOT, 2020). A complete system that includes a GUI (graphical user interface) requires about 30 kilobytes of RAM. The application enables

efficient use of hardware with simultaneous standardized low-power wireless communication for different hardware platforms.

Contiki-NG comes bundled with a range of applications such as: small web browsers, calculator, web server, email client and FTP.

Contiki OS supports IPv4 and IPv6 implementations based on TCP, UDP and HTTP protocols. All protocols supported by Contiki OS are: 6LoWPAN, RPL, CoAP, uIP (for IPv4), uIPv6 (for IPv6).

The basic language of this operating system is C language.

Before implementing a real-time IoT product, a simulator called Cooja tests each IoT product. Cooja Network Simulator facilitates the process of software development and troubleshooting.

The complete code is available on GitHub for use or further development. This project is under constant development by many developers around the world, and the software is also used in companies such as Google and Cisco.

Advantages

- It fully supports IPv6 and IPv4 standards, along with 6lowpan, RPL, and CoAP.
- Contiki-NG runs on a range of low-power wireless devices, many of which can be purchased online
- Contiki-NG is open source software - it can be used for both commercial and non-commercial purposes.
- It has a large community of developers who can help solve possible technical problems.

Disadvantages

- Contiki-NG is an event-based OS, meaning it does not use a scheduling algorithm.
- Contiki-NG partially supports real-time application development.

Application

Contiki is used in many key systems such as: street lighting, construction site monitoring, industrial surveillance, alarm systems, remote monitoring in the house.

IoT applications require a good knowledge of hardware, embedded system software, network protocols, low-power wireless communication, Internet server software and smartphone applications development. (Pearce, 2012) For Contiki, a Thingsquare platform has been created that allows applications to be created in an easy and fast way. (Thingsquare, n.d.). Some of the examples created using this platform are:

- **A hand wash sensor** built from off-the-shelf hardware to help the 20 seconds hand wash. The sensor detects that someone is standing in front of the sink. Then starts to blink:
  - A red blink for 20 seconds to indicate that we should keep washing
  - A quick green blink to indicate that we are done

  The idea is that this will help us keep the 20 second habit up. (Dunkels, 2020)
- **Retail shelf monitoring**

  Empty shelves in retail outlets repel customers. Up to 24% of Amazon's online sales refer to the sale of goods that customers could not find in stores in their area because the goods were sold out and not renewed. This solution consists of two parts: a wireless laser sensor and a smartphone app that is implemented on the phones of employees in charge of procurement. (Dunkels, 2019) The problem of stock monitoring can also be solved by applying RFID technology, reading RFID tags in the so-called smart shelves as shown in the paper 'SDD ITG Smart shelf RFID solution for the stocktaking of goods on remote shelves' (2010).
- **Street lighting**
- The cost of street lighting can reach 40% of the total energy consumption costs of a city. The greatest benefits from the application of this IoT solution are reflected in lower electricity consumption by using automated or manual dimming, and in automatic failure detection. The solution shown in (Dunkels, 2019) consists of two parts:
  - A wireless microcontroller on each lamp, and
  - A wireless mesh network that connects all the lamps in one huge network. The network has one or more access points that connect the wireless mesh to the back-end controller

  The system uses the Thingsquare IoT platform with sub-GHz networking. (Dunkels, 2019)

- **Office plants-as-a-service**
  Most employed people spend a lot of time indoors, about 90%. To make them feel better, the interior of the room is often enriched with various flowers and trees that require care. In order not to have to take care of plants, companies can rent plants from companies that deal with it. A plants-as-a-service IoT system allows the renter to monitor soil moisture for each individual plant and to define the plant maintenance regime based on a database and schedule. This can significantly extend the life of leased plants and increase the profit of the renter.
  The system consists of. (Dunkels, 2019):
  - A mobile app, for the plant care specialists and personnel
  - Wireless soil sensors, that keep track of the status of each plant
  - A backend database and scheduling system.

The possibilities of creating IoT applications are practically unlimited, from small and sometimes at first glance unnecessary, to technologically complex and economically crucial.

2.2 RIOT OS

RIOT supports IoT in the same way that Linux supports the Internet. RIOT is a free open source operating system that is widely supported by companies and academia, but also by individual hobbyists around the world. RIOT supports most low-power IoT devices and microcontrollers of various architectures (32-bit, 16-bit, 8-bit). It aims to implement all relevant open standards that support the Internet of Things in a secure, permanent and privacy-friendly manner. (RIOT, 2020)

RIOT OS requires ~1.5 kB of RAM and ~5 kB of ROM. RIOT supports C and C ++, Multi-Threading, MCU w/o MMU, modularity and real-time applications.

RIOT supports the 6LoWPAN (RFC6282 and RFC6775 compatible) IPv6 low-power communication package, which allows it to connect to the Internet. The system works on various platforms based on energy efficient architectures such as Airfy Beacon, Arduino, Microchip, Nordic, STM32X, Texas Instruments Zolertia and others. The kernel source code is available as open source with the GNU Lesser General Public License version 2.1 (LGPLv2.1). Some external libraries are subject to other licenses that are part of these packages.

RIOT OS comes bundled with a variety of drivers for a variety of devices, from radio transceivers, acceleration sensors, gyroscopes, to servomotors.

The basic languages of this operating system are C and C ++.

Prior to the implementation of IoT products, testing with tools such as Embunit was enabled.

The complete code is available on GitHub for use or further development.

Advantages

- It fully supports the IPv6 standard, along with 6lowpan, RPL, UDP, CoAP, and CBOR.
- Offers static and dynamic memory allocation.
- It is robust and works on a range of low-power wireless devices.
- RIOT is open source software - it can be used for both commercial and non-commercial purposes.
- It allows scheduling activities over a long period of time.
- It is supported by a large community of developers who can help solve specific problems.

Disadvantages

- RIOT sometimes requires increased memory consumption but is still within the available limits of IoT devices
- A little more work around the service because it is reimplementing "from scratch". (Baccelli, 2018)

Application

RIOT-OS provides a number of possibilities for use. Some of the better-known applications are:

- **CleverWeather with RIOT-OS** in which RIOT-OS applications collect data from sensors and deliver it to an Azure Iot hub using MQTT-SN and MQTT protocol. Details of this system are shown in (Catalano, 2020), and only its basic components will be shown here:
  - *RIOT-OS app* – creates values and publish them to a MQTT-SN channel.

  - *Mosquito RSMB* – ready-to-use broker
  - *PytonGateway* – allows RSMB to communicate with the IoT hub
  - *Azure IoT Hub* – it is used as a MQTT broker. Depending of user and the application, Azure can be free or payable.
  - *Nodejs* – can run an application for visualization the data in local browser.
- **Locha Mesh** solution is based on RIOT, and the idea is to enable the transmission of messages, data, Bitcoin transactions, and other services, decentralized, secure, using radio waves, bypassing the Internet, and even in the event of a power outage. Used radio transceivers have their own batteries. In addition to everyday use, their use is also possible in case of natural disasters. (Dudey, 2020)
- **AWS based IoT Virtual Environmental Station** was created with the goal of building a virtual environmental station based on Amazon Web Services (AWS). Two RIOT-OS native boards generate random environmental data and send them to AWS through MQTT-SN (Mosquitto) and an MQTT transparent bridge. (Zizzo, 2020)

2.3 TinyOS

TinyOS is open source, free software, with BSD (Berkeley Software Distribution) - licensed code. (Boral, 2019) It is designed for low-power wireless devices, such as those used in sensor networks, ubiquitous computing, personal networks, smart buildings, and smart meters. (Hicham, Jeghal, Sabrim, & Tairi, 2017)

It uses the nesC programming language, which is an extension of the C programming language and is intended for TinyOS and applications to control and manage wireless sensor networks. The differences between C and nesC are in the connection model (function). TinyOS has a monolithic architecture based on a combination of components. NesC supports TinyOS components (interface, modules, configurations, drivers). Each piece of code consists of simple functions housed in components and complex functions that integrate all components.

Sensor devices in such networks (particles, motes) are characterized by limited memory and low power. Low power sensors, due to their limited size, require efficient use of resources. TinyOS uses an event-based design, which allows the CPU to "rest" when there are no unresolved tasks. An example of such an event is the activation of an alarm when the temperature of the thermostat begins to change abruptly in relation to the set temperature. As soon as this task is completed, the sensor can go into sleep mode.

As of October 2019, TinyOS had over 35,000 downloads per year. (Kuppusamy, 2019) Its main application is in devices that use wireless sensor networks. TinyOS is also suitable for controlling the environment - it is useful for monitoring air pollution, forest fires, and preventing natural disasters. Its use in smart vehicles is also becoming more common — since smart vehicles are autonomous, one can view them as a network of sensors where all of these sensors communicate over low-power wireless networks known as LPWANs. This way of working fits TinyOS perfectly. TinyOS is also widely used in smart cities. In addition to the basic advantages that this operating system provides such as small code (400B), small requirements for RAM, another advantage is the built-in TOSSIM simulator. It allows one to develop, debug, and test the entire application on a PC and to transfer the unchanged code directly to the sensor nodes.

Advantages

- TinyOS was developed and is supported by the American University of Berkeley, which offers to download and monitor it under a BSD license.
- Based on event-based operation, TinyOS offers the user precise sensor control and allows him to better adapt to random wireless communication between physical interfaces.
- Compatibility with at least 9 hardware platforms and ease of adding or modifying platforms.
- TinyOS tracks hierarchical material abstraction in three layers.
- The preventive nature of an operating system determines whether it will allow an ongoing task to be interrupted. TinyOS does not manage this prevention but prefers hardware interrupts.
- TinyOS is designed to optimally manage power consumption: it puts the node on standby when there are no tasks to perform. This deactivates the radio device and

switches it to the "Low power listening" mode. (Hicham, Jeghal, Sabrim, & Tairi, 2017)
- It takes a little memory to run TinyOS. There is no need to buy larger memory devices to run it. TinyOS also creates small code, which reduces the power consumption for storing data in RAM.
- The code is optimized to run on any specific device. Due to the smaller size of the code, the devices run fast, and the OS does not load them.
- Modularity - TinyOS contains many different modules. Each of them performs its function. Modules include tasks, commands, events, microcontrollers, hardware, and software. Each of these modules communicates with everyone else so that the wireless devices can function properly.
- Low voltage - Due to the small memory and small space it takes up, TinyOS uses a weaker battery, so it can run on smaller devices with low voltage.
- Reuse - TinyOS can be reused on similar devices. This means that a change in the code is not necessary if the devices are of the same nature.

Disadvantages

- TinyOS does not support real-time applications. It provides a priority-based process planning algorithm (FIFO[1] and EDF[2]), but when the planned process is started, it is executed to completion. This may result in missing the deadline for activating the high-priority process that is next in line for implementation after the low-priority process. Tasks are not preempted by other tasks, but can be preempted on some other way.
- TinyOS in its current state is not an operating system that is in line with modern trends. It does not offer, for example, multitasking, multi-user options or a file system. It does not offer user mode and kernel mode. In fact, TinyOS is a set of routines available to the developer to simplify development.

Application

The main applications of this OS include different types of devices that use wireless sensor networks. Tiny OS is widely used for sensor nodes and is considered the most powerful, innovative, energy-efficient, and widely used OS.

- **Environmental Monitoring** - Since each TinyOS can be built into a small sensor, it is very useful for monitoring air pollution, forest fires, and preventing natural disasters.
- **Smart vehicles** can be viewed as a network of sensors. With this, the sensors communicate via low-power wireless networks known as LPVAN, which is very convenient for TinyOS.
- **Smart cities** install smart sensors characterized by high processing power and multi-layered IP capabilities. Sensors are placed everywhere. TinyOS is a sustainable solution for low-power sensors located everywhere in smart cities.

2.4 Raspberry Pi OS

Raspberry Pi OS is a free operating system, based on the latest version of 'Debian 10' (Buster), with Linux kernel version 4.19 and 8.3 GCC compiler. It is recommended and optimized for Raspberry Pi hardware. Debian is a free operating system that includes a set of programs that allow computers to work with thousands of other packages. Debian has a good reputation in the Linux community because it is remarkably high quality and stable. At the core of Debian is the Kernel. These features are copied to the Raspberry Pi OS.

However, the Raspberry Pi OS (formerly Raspbian) offers more than the OS itself. It comes with over 35,000 programs, with pre-assembled software for easy installation on the Raspberry Pi. Almost all the information that can be found about Debian will probably refer to the same version of Raspbian. Because Raspbian is closely related to Debian, a large amount of documentation is available for the Raspberry Pi OS.

As of May 2020, three versions of Raspberry Pi OS (32-bit) with 4.19 kernel are available:
- Lite – 432 MB
- With desktop – 1128 MB
- With desktop and recommended software – 2523 MB

In addition to the usual changes such as code error fixes, many options have been added from

[1] First-In, First-Out

[2] Earliest deadline first scheduling

which we would single out (Raspberry Pi OS (32-bit) Lite - Release notes, 2020):

- The Bookshelf application has been added
- The Raspberry Pi Diagnostics application has been added
- The Magnifier application has been added in the Recommended Software
- Internal audio outputs are enabled as separate ALSA devices
- The preinstallation for MagPi has been removed and replaced by the Beginner's Guide
- Chromium has created a default application for PDF files
- Focus behavior has changed so that the focus shifts to the desktop if the windows are not opened - improves the reliability of the Orca screen reader
- The disk ID is now regenerated on first boot
- i2cprobe: more flexible I2C / SPI alias mapping.

Buster is harder to hack, and most of the other differences from previous versions of Raspberry Pi OS (Raspbian) are mostly small. Python, Scratch, Sonic Pi, Java, and many other programs are also pre-installed in previous versions of Raspberry Pi OS. The latest versions of Raspberry also include a setup wizard.

There are two options when installing:

- the first is via an SD card, which has more than 2GB of memory (maximum 32 GB for Pi3, or 64 GB for Pi4, but with formatting in the exFAT file system) (Manske, 2019), heaving in mind that either Raspberry Pi Imager or NOOBS can be used. In both cases, after installing the OS on the SD card, inserting the SD card into the device will automatically activate the system.
- the other way is if Raspbian is already installed on the Raspberry Pi and it only needs to be upgraded. If one wants to update Raspbian and thus install all the latest application updates in the terminal, he needs to type the following commands:

  $ sudo apt-get update
  $ sudo apt-get upgrade
  $ sudo apt-get dist-upgrade

Advantages

- The main advantage of Raspbian Pi OS is that it is designed for a specific device and allows the best use of device performance. The device itself has aroused great interest and many additional devices are being made for it, which further increases its popularity.
- Another advantage is that it is based on a reliable Linux Debian operating system and that it is upgraded in parallel. This provided him with powerful support.

Disadvantages

- The main advantage is also the main disadvantage. Because it is intended for a specific device, its application on the Raspberry Pi platform is limited.
- Another disadvantage is the amount of memory it needs to operate, and in this connection the amount of electricity to power the device, but this is only a consequence of the basic disadvantage.

Application

Raspberry Pi OS contains software environments intended to expand the knowledge and skills of young people in the field of computer science interestingly. In addition to knowledge of computer science, it is also suitable for acquiring basic knowledge in the field of electronics.

The Scratch environment is one of the pre-installed environments under the Raspberry Pi operating system. This environment is suitable for children older than 8 years. Wolfram Language and Mathematica are integral parts of the Raspberry Pi operating system as part of a joint project and are free for non-commercial use. The first version of the Mathematica program was created in 1988 and has always been considered a very demanding programming environment. The version under the Raspberry Pi operating system executes most requests within a reasonable time, although it is 10 to 20 times slower than the average computer. The basis is a new pilot version of the core (Wolfram Engine) that handles all calculations.

If it is necessary to be able to use the media center on the TV set, via an HDMI cable a Raspberry Pi computer which has an operating system specially designed for these needs can be connected to the newer TV set. If one wants to add the ability to search the Internet or use certain programs, he can in the same way simply connect

a fully installed Raspberry Pi via an HDMI cable to the TV.

2.5 Ubuntu Core

Ubuntu Core is Ubuntu for IoT and embedding. It is optimized for security and reliable updates. It is easy to install, and resistant to unauthorized work and corruption. Its read-only root filesystem is built from the same packages used to build a wider set of Ubuntu distributions and differs only in the way packages are delivered and, essentially, in the way they are updated. All of this is an interrupt-driven, secure, confined, independent, cross-platform Linux packaging system. Snap packages ensure that there is always a clear separation between the base system and all other applications to be installed, as well as isolation between each application, their data, and even application version data. (Ubuntu, 2020)

Updates are transactional, meaning they are either 100% successful or not installed. If the changes are not installed, they do not leave any malfunction in the record. Only the details of the minutes remain. This means that the system remains fully operational and in a constantly well-defined condition during system implementation and updating. The system can also be recovered or restored if necessary, even if the system does not start. An unsuccessful update never leaves the system in an unpredictable state. (Ubuntu, 2020)

Ubuntu Core is designed to meet the requirements of IoT devices. It runs on many different hardware, including the Raspberry Pi, Intel NUC, Qalcomm Snapdragon 410c, and even the Kernel-based Virtual Machine (KVM).

Minimum requirements for Ubuntu Core operation include 500MHz single-core CPU, 256MB RAM, 512MB storage. If used on a Raspberry Pi 4, then it uses 4 cores and 1, 2 or 4 GB of RAM. On Qualcomm DragonBoard it uses 4 cores, 1 GB RAM and 8 GB eMMC flash storage. Intel NUC with Intel Core i3, i5, i7 64-bit enables >8 cores and >32 GB RAM, without built-in storage.

Security is provided by public and private key encryption, and two-step validation raises security to a higher level. For the system to remain secure, it needs to be constantly updated either automatically or after receiving a message about the availability of a new version.

Advantages

- Simple and consistent installation and application.
- A read-only filesystem allows applications to be activated independently of each other, and access to system resources is provided by specific transactional update permissions, so the system can handle any unforeseen situations, hardware, or network.
- The snap-based concept provides security, ease of construction and painless distribution, and public/private key validation ensures that exactly what needs to be run is running. (Ubuntu, 2020)

Disadvantages

- One of the main disadvantages of Ubuntu is the limited choice of applications although the OS is free, and many applications are also free to download.
- One of the main criticisms of Ubuntu is related to "seeming commercialization". With each update, Ubuntu moves away from the open-source identity, because the company is increasingly working independently and neglecting the open-source community. (Ivankov, 2019)

Application

Thanks to the long existence of Ubuntu, the possibilities of application are great. Due to its reliability, Ubuntu Core is suitable for industrial applications, to provide functional safety for critical embedded systems, for example in the automotive sector and robotics. Ubuntu Core is considered the safest choice for IoT because there are several layers that take care of system security, so it is suitable as a gateway (e.g. Dell Edge Gateway) and for business-critical equipment. It can provide full disk encryption. There are examples of using Ubuntu with Tunnel drones. Its application is growing in the areas of digital signage and smart displays. (Canonical, 2020)

2.6 Zephyr IoT OS

Zephyr is a "real-time" operating system used with connected, limited resources and embedded devices. The simple integration of different IoT architectures makes it popular among IoT professionals. The Zephyr project is supported by

the Linux Foundation, as one of their projects. The goal of this project is to build a secure and flexible "real-time" operating system (RTOS) for IoT. Intel, Linaro, Synopsys, Google, NXP, Acer, and Sony are also involved in this project.

Zephyr is highly configured and modular, with statically allocated memory and resources. Zephyr supports IoT in several hardware architectures, such as Intel x86, ARM Cortex-M, NIOS II, ARC, with more than 200 boards. It offers complete flexibility and freedom of choice, open-source code, neutral management and small footprint and scalability from small Cortex-M devices to multi-core 64-bit CPUs. (Zephyr, 2020)

Support for Zephyr now includes Bluetooth, Ethernet, Wi-Fi, IPv4/IPv6, 6LoWPAN, and NFC. Zephyr has excellent documentation and a system development kit.

C is the dominant language.

This OS requires at least 8kb of RAM and at least 512kb of ROM memory. Documentation can be found on Zephyr's website in the documentation section.

The source code of the project is licensed under the Apache v2.0 license, but there are also imported and reused components that are subject to other licenses. The source code can be divided into modules. Modules can be divided into three layers based on how close the module is to the hardware. In each layer, modules from the same and each lower level can be used (Pecnik, 2020):

- Kernel Layer - Contains modules for managing low-level processes related to hardware and task scheduling.
- OS Services layer - Provides access to all common operating system functions. The modules in this layer are viewed as application design blocks.
- Application Services layer - Users can create applications to implement certain functionalities in accordance with their own project requirements.

Zephyr IoT OS is a secure system, which offers support for memory protection. It implements configurable stack-overflow protection adapted to the specific system architecture, monitors "kernel object and device driver permission tracking, and thread isolation with thread-level memory protection on x86, ARC, and ARM architectures, user-space, and memory domains". (Z.P., 2020)

For platforms without MMU/MPU and devices with limited memory, Zephyr supports combining application-specific code with a custom kernel. Thus, the combined code is loaded and executed on the same address space of the device.

Zephyr uses a multi-purpose tool called “west”. It is written in Python 3 and is distributed via PyPI. Zephyr can be used without the west, but then the work is more uncomfortable. One of the more important purposes of the west tool is to manage multiple Git repositories. When installing Zephyr, a directory called the manifest repository is important, and it contains the west.yml file. This file, together with the west configuration files, controls the installation behavior.

Advantages

- Flexibility - Zephyr IoT OS can be tailored to a specific task so that it contains only those components that are needed to support the task's requirements. In this way, much more efficient use of memory and resources is achieved and everything that is not necessary for the application is removed.
- Zephyr IoT OS has integration with test automation tools.
- Some of the features that Zephyr supports are memory allocation, communication, and synchronization, power management, memory protection, POSIX…
- BLE (Bluetooth Low Energy) is one of the main functions supported in Zephyr.

Disadvantages

- Users suggest creating better default varieties of reporting.
- There were some bugs found after major releases (but these were always resolved very quickly).

Application

Some of the most popular products already using Zephyr IoT OS are:

1. HereO uses Zephyr IoT OS to control multiple modem devices. The goal of this project was to develop a software package that would enable the launch of multiple communication devices, using the same UART port. One of the main products is a watch for children. The

watch consists of an EPD screen that is touch-sensitive, and Bluetooth support. It is designed and intended for children aged 3 and over. Parents can track where their children are using the hereO smartphone app. There is also the possibility of setting up safe zones such as school or home, where the application provides notification when a child arrives or leaves a particular location. (Ochs, 2014) This system contains 3 UART[3] devices.

2. CommSolid – CSN130 the market's first integration-ready NarrowBand-IoT Intellectual Property (IP) solution, designed for 3GPP NarrowBand-IoT standard. It is built into the chip and thus allows sensors and actuators to be directly connected to the Internet. This is how smart applications for logistics, health, smart cities, and surveillance were created. Extremely low power consumption makes it very suitable for devices with batteries. CommSolid evaluated and studied different real-time OSes. The main characteristics that were crucial for the selection of the OS are performance and stability. Simple mechanisms are also needed to perform software and user application tasks. All these features are possessed by Zephyr, which offers ultra-low power consumption and long battery life.
3. Grush-Grush is the creator of the advanced Bluetooth toothbrush. This toothbrush differs in that children can have interactive games on their phones while brushing their teeth. All of these games can be found on Google Play and the App Store. (Connelly, 2018) The app also includes a "Parental Control Panel" that allows parents to monitor how their children brush their teeth. The brush wirelessly transmits data with the help of BlueTooth to Grush Games - an interactive mobile game that guides children through the entire process of brushing their teeth. It also uses the Cloud, which preserves the entire history of brushing teeth. Zephyr made it easy for Grush to develop the advanced algorithms on which this brush is based. They needed an OS that could collect data from sensors, process complex algorithms, and communicate with a smartphone. The Zephyr has all the necessary features.

2.7 Windows 10 for IoT

Windows 10 IoT belongs to the Windows 10 family and is intended for the Internet of Things. It enables organizations to create their IoT and fit into the cloud strategy effortlessly. It appears in two editions, Core and Enterprise. The Core version can run individual applications, and the Enterprise version is a complete version of Windows 10 with specialized features for creating a range of applications on peripherals. Accordingly, the differences are shown in Table 7.

Table 7 Differences between Windows 10 IoT Core and Windows 10 IoT Enterprise

	Windows 10 IoT Core	**Windows 10 IoT Enterprise**
User experience	One UWP app in the foreground at a time with supporting background apps and services.	Traditional Windows Shell with Advanced Lockdown Features
App architecture supported	UWP UI only	Full Windows UI support
Cortana	Cortana SDK	Yes
Domain join	AAD only	AAD and Traditional Domain
CPU Architecture support	x86, x64, and ARM	x86 and x64
Licensing	Online Licensing Agreement and Embedded OEM Agreements, Royalty-free	Direct and Indirect Embedded OEM Agreements
Usage scenarios	Digital Signage, Smart Building, IoT Gateway, HMI, Smart Home, Wearables	Industry Tablets, Retail Point of Service, Kiosk, Digital Signage, ATM, Medical Devices, Manufacturing Devices, Thin Client

Source: (Warwick & et al, 2018)

[3] UART stands for Universal Asynchronous Receiver-Transmitter

In addition to these differences, when booting, the Core version is booted to the default application, which allows the user to use open-sourced code for this application and their own applications.

The minimum hardware requirements for Windows 10 IoT OS are shown in Table 8.

Table 8 Minimum hardware requirements summary for Windows 10 IoT

	Components	IoT Core	IoT Enterprise and LTSB[4]	
1	Processor	400 MHz or faster x86, x64 processor or ARM SoC	1 GHz or faster processor or SoC	
2	RAM	256 MB available to the OS for devices without display support 512 MB available to the OS for devices with display support, depending on resolution	1 GB for 32-bit OS 2 GB for 64-bit OS	
3	Storage	2 GB	32-bit	16 GB or greater
4	Security	Optional	64-bit	20 GB or greater
5	Display	Optional	Yes	
6	Wireless	Optional	Yes	
7	Networking	Optional	Yes	

Source: (Microsoft, 2017)

Advantages

- Windows 10 is constantly updated, which is good for ordinary users, but not good for industrial customers because it can affect the functioning of the system. Therefore, while updating Windows 10 IoT industrial customers can choose which updates will be installed. Only security and safety sections are automatically updated.
- Each version of Windows 10 Enterprise IoT LTSC will be available at least 10 years from launch.

Disadvantages

- If users need to use Edge, Cortana, or the Windows Store, they need to look elsewhere, as none of these features are included with IoT Enterprise.
- While updating Windows 10 IoT industrial customers need to choose which updates will be installed.
- The price of Windows 10 IoT is not clear and it very depends on the feature set for an IoT device.

Application

Given the importance of the Windows family of operating systems and the compatibility of systems of this family, there are many different Windows 10 IoT applications. Some are listed below:

- **Dover Fueling Solutions** is described in detail in (Dover, 2020). The idea is to download data on the customer, fuel sold and purchases made by the customer at the gas station from each individual gas station, to send them to the cloud, process them there, and return the results to the place of use. The results contain many statistics important for the gas station owner. This is not a spectacular new solution, because a similar solution for trucking companies has been known for a long time (Cekerevac Z. , Matic, Djuric, & Celebic, 2006), but another technology has been applied here and the network of petrol stations is covered.
- **Idex Fire & Safety** system, the IoT gateway solution, it is intended to keep fire trucks ready for use. The Smart Truck Technology platform based on Windows 10 IoT Core and Azure connects different fire truck systems and components and provides real-time data to the personnel in charge of their maintenance. The activities are performed within the centralized solution for monitoring, processing, and distribution of vehicle data, AXIS. (Idex, 2019)
- **SmartHubs**, elegant multifunctional kiosks with a multitude of IoT devices that are

[4] Long-Term Servicing Branch

available to citizens and enable the execution of many applications and services. They can accommodate radio transceivers for 5G networks, motion sensors, security cameras, IoT gateways, audio-video devices, EV charging devices, etc. Each SmartHub can connect to super high-speed Internet. (Citybeacon, 2019)

- The **Hourfleet** system based on mobile devices using Windows 10 IoT Core installed on each vehicle allows users to control the vehicle if they have a valid code and smartphone. In that way, the application of car-sharing is enabled. The solution consists of a mobile device built into each vehicle that allows one to open the door and start the engine, as well as monitor the use of the vehicle. The back end of the system is hosted in Azure. An ultra-compact, system-on-a-module (SOM) ARM board manufactured by SolidRun, a small PC, which uses an NXP i.MX chip is used to reduce battery power consumption and simplify system updates. (Hourfleet, 2020)

3 CHECKLIST FOR OS SELECTION

Based on the considered needs of the application, a checklist should be created for the technical requirements for the selection of the operating system for a specific equipment configuration and specific purpose. This is explained in more detail in (IoT Operating Systems, 2016), and only the most important issues will be highlighted here:

1. Does the application need real-time or deterministic performance?
2. What the available resources (memory, processing capability, etc.)?
   a. Memory size: If the OS does not fit in the internal memory of the processing unit (MCU or MPU), expensive external memory is required.
   b. Memory management unit (MMU): Many of the small MCUs used in sensor nodes do not support MMUs to manage caching, memory allocation, and protection. The chosen RTOS should be able to provide efficient memory management to minimize extra coding requirements.
   c. Processing capability: The processor should have enough margins to comfortably support the OS as well as run-time applications.
3. What are the security requirements? Security is a top consideration and should be accounted for in the hardware, OS, and network layers.
4. How is the device powered? OSes that support power management features can efficiently manage applications and improve battery life.
5. What is the hardware chosen? The chosen OS should be able to support the various platforms.
6. What are the communication and networking requirements?
7. Is enterprise system interoperability needed? It is necessary to consider process and systems used for the integration of field devices with enterprise systems.

Complete and accurate answers to these questions can allow the designer to avoid the usual pitfalls and later costly system upgrades.

4 CONCLUSIONS

The IoT device market is constantly growing, and it is expected that this trend will continue, and thus more and more things will be connected to the Internet. This includes things that are used in our homes like smart devices, things in our communities like smart parking spaces, and things in offices and factories like smart printers and robotic equipment. As the number and complexity of smart things grow, so does the technology that supports them. Greater throughput, more efficient use of resources and greater coverage are needed for IoT growth to continue in the coming years. In recent years, we have seen that the IoT industry is making increasing progress and that large companies such as Microsoft, Amazon, Google, T-Mobile, AT&T, GE, and many other top brands are part of the IoT industry.

For the growing demands coming from the IoT, it will be suitable the 5G system together with intelligent connectivity, a concept that envisions a combination of 5G network, IoT, and AI. By applying 5G technology and IoT, society will be more efficient, and smart cities will live up to their name.

The next trend is the massive adoption of IoT with billions of devices. To achieve it, two conditions must be met:

- On the technical side, the IoT standard must offer both scalability and versatility, offering enough capacity and network efficiency to connect millions of devices. At the same time, it must offer advanced features such as longer battery life and a wider coverage area to increase the number of new uses of this concept.
- On the application side, many new use cases need to be developed and tested. It will take several years for the full adoption of the 5G network, according to the GSMA, by 2025, half of the mobile connections will be 5G, and the other half will use older technology.

The proposed IoT OS represents a narrow selection of possible IoT OS. Each of them has the specifics that give it an advantage at a certain time and in a certain area. Very frequent updates and additions allow them to be constantly improved, so those who survive will likely look more and more like each other.

WORKS CITED

Baccelli, E. (2018, Apr 4). *RIOT an Open Source Operating System for Low-end IoT Devices.* Retrieved from SILECS: www.silecs.net/wp-content/uploads/2018/04/2018-04FIT-IoT-Lab-Grid5000-school-Baccelli.pdf

Boral, S. (2019). *TinyOS: An Operating System for Wireless Sensors*. Retrieved from https://www.iottechtrends.com/tinyos-operating-system-for-wireless-sensors/

Canonical. (2020, 06). *Ubuntu for the Internet of Things.* Retrieved from Ubuntu IoT: https://ubuntu.com/internet-of-things

Catalano, D. (2020, Apr 2). *CleverWeather with RIOT-OS.* Retrieved from Hackster: https://www.hackster.io/domitix/cleverweather-with-riot-os-ada7fe

Cekerevac, Z., Matic, S., Djuric, D., & Celebic, D. (2006). ITGfdc-1 Fuel Dispenser Control System as the Technical Solution for Preventing of Non Authorized Fuel Tanking. *Proc. 11th International Scientific Conference devoted to Crises Situations Solution in Specific Environment.* Zilina: FSI.

Cekerevac, Z., Matic, S., Djuric, D., Celebic, D., & Dvorak, Z. (2010). SDD ITG Smart shelf RFID solution for the stocktaking of goods on remote shelves. *IMK-14 Istraživanje i razvoj*, 47-52.

Citybeacon. (2019, Aug 22). *Customer Stories - citybeacon.* Retrieved from Microsoft: https://customers.microsoft.com/en-us/story/749839-citybeacon

Connelly, L. (2018). *Smart toothbrush start-up wants to turn a chore into a game*. Retrieved from https://www.cnbc.com/2018/03/08/smart-toothbrush-start-up-wants-to-turn-a-chore-into-a-game.html

Craven, C. (2020, Jan 18). *What is the 5G Spectrum? Definition.* Retrieved from sdxcentral: https://www.sdxcentral.com/5g/definitions/what-is-5g-spectrum/

DistroWatch.com. (2020, Jun 26). *OS Distribution*. Retrieved from DistroWatch.com: https://distrowatch.com

Dover. (2020, Jan 9). *Dover Fueling Solutions.* Retrieved from customers.microsoft.com: https://customers.microsoft.com/en-us/story/775087-microsoft-country-corner-dover-fueling-solutions-oil-and-gas-azure

Dudey, J. P. (2020, Jun 11). *Locha Mesh.* Retrieved from github.com: https://github.com/btcven/locha#whats-in-the-code

Dunkels, A. (2019, Nov 14). *5 IoT Use Cases and Customer Projects (2019).* Retrieved from thingsquare: https://www.thingsquare.com/blog/articles/iot-use-cases-2019/

Dunkels, A. (2020, Mar 26). *A Hand Wash Sensor built from Off-the-Shelf Hardware.* Retrieved from thingsquere: https://www.thingsquare.com/blog/articles/handwash-sensor/

Duquennoy, S. (2019, Oct 18). *contiki-ng Home*. Retrieved from github.com: https://github.com/contiki-ng/contiki-ng/wiki

Dutta, P. (2020). *The 20 Best Raspberry Pi OS Available to Use in 2020.* Retrieved from Ubuntupit.com: https://www.ubuntupit.com/best-raspberry-pi-os-available/

g2. (2020). *Best IoT Operating Systems.* Retrieved from g2.com: https://www.g2.com/categories/iot-operating-systems

Hawlader. (2020). *Top 20 Best IoT Operating Systems for Your IoT Devices in 2020.* Retrieved from Fossguru.com: https://www.fossguru.com/best-iot-operating-systems-for-your-iot-devices/

Hicham, A., Jeghal, A., Sabrim, A., & Tairi, H. (2017, Aug). A Comparative Study Between Operating Systems (Os) for the Internet of Things (IoT). *Transactions on Machine Learning and Artificial Intelligence, 5*(4). doi:10.14738/tmlai.54.3192

Hourfleet. (2020, Jan 28). *Hourfleet.* Retrieved from customers.microsoft.com: https://customers.microsoft.com/en-us/story/779418-hourfleet-windows-iot

Idex. (2019, Oct 29). *IDEX Fire & Safety.* Retrieved from customers.microsoft.com: https://customers.microsoft.com/en-us/story/764579-idex-automotive-azure-iot-united-states

Ivankov, A. (2019, Aug 23). *Ubuntu Operating System: Advantages and Disadvantages.* Retrieved from Profolus: https://www.profolus.com/topics/ubuntu-operating-system-advantages-and-disadvantages/

Johansen, S. (2016, Sep 13). *1G, 2G, 3G, 4G, 5G.* Retrieved from its-wiki.no: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiag5PG-pXqAhVSDewKHZYgCowQFjALegQIARAB&url=https%3A%2F%2Fits-wiki.no%2Fimages%2Fc%2Fc8%2FFrom_1G_to_5G_Simon.pdf&usg=AOvVaw3B1Usth5vNQvj8I2XMSSUW

Kumar, A. (2018, 05 04). *Top 5 Open Source Operating Systems for IoT devices.* Retrieved from Technotification.com: https://www.technotification.com/2018/05/open-source-iot-operating-systems.html

Kuppusamy, K. S. (2019). *The Five Most Popular Operating Systems for the Internet of Things*. Retrieved from https://opensourceforu.com/2019/10/the-five-most-popular-operating-systems-for-the-internet-of-things/

Leverege. (2020, Apr 9). *WiFi for Enterprise IoT: Why You Shouldn't Use It.* Retrieved from iotforall: https://www.iotforall.com/wifi-enterprise-iot/

Manske, K. (2019, June 28). *Need a micro SD card for your Raspberry Pi 4? Here, we offer some suggestions around cost, speed, and storage amounts.* Retrieved from Maker Pro: https://maker.pro/raspberry-pi/tutorial/what-micro-sd-card-is-best-for-a-raspberry-pi-4

Mehedi, H. (2020). *Top 15 Best IoT Operating System For Your IoT Devices in 2020.* Retrieved from ubuntupit.com: https://www.ubuntupit.com/best-iot-operating-system-for-your-iot-devices/

Microsoft. (2017, Feb 05). *Minimum hardware requirements.* Retrieved from Microsoft Hardware Dev Center: https://docs.microsoft.com/en-us/windows-hardware/design/minimum/minimum-hardware-requirements-overview

Mishra, H. (2019). *IoT OS and RTOS for Internet of Things Devices*. Retrieved from https://medium.com/@harshhvm/iot-os-and-rtos-for-internet-of-things-devices-967c9b8077c6

n.d. (2016, 09 13). *IoT Operating Systems.* Retrieved from ARROW: https://www.arrow.com/en/research-and-events/articles/iot-operating-systems

n.d. (2020, May 27). *Raspberry Pi OS (32-bit) Lite - Release notes.* Retrieved from Raspberry Pi: http://downloads.raspberrypi.org/raspios_armhf/release_notes.txt

Ochs, S. (2014, Mar 18). *HereO's GPS watch tracks your kids anywhere they go*. Retrieved from PCWorld: https://www.pcworld.com/article/2109142/hereos-gps-watch-tracks-your-kids-anywhere-they-go.html

Okoi, M. D. (2019, 04 15). *5 Operating Systems For The Internet Of Things.* Retrieved from FOSSMint: https://www.fossmint.com/operating-systems-for-the-internet-of-things/

Okoi, M. D. (2020, 05 12). *20 Best Operating Systems You Can Run on Raspberry Pi in 2020.* Retrieved from FOSSMint: https://www.fossmint.com/operating-systems-for-raspberry-pi/

OKportal Technology. (2019, May 23). *The evolution of 5G.* Retrieved from Quora: https://www.quora.com/Will-the-incremental-gain-in-productivity-and-anything-else-be-as-high-for-5G-as-it-was-for-4G-and-3G

Palmer, D. (2017, July 31). *175,000 IoT cameras can be remotely hacked thanks to flaw, says security researcher.* Retrieved from ZDNet: https://www.zdnet.com/article/175000-iot-cameras-can-be-remotely-hacked-thanks-to-flaw-says-security-researcher/

Pearce, R. (2012, Oct 5). *Contiki: An operating system for the 'Internet of Things'.* Retrieved from Computerworld: https://www.computerworld.com/article/3470721/contiki-an-operating-system-for-the-internet-of-things.html

Pecnik, T. (2020). *7 najboljih operativnih sistema za IoT.* Beograd: Poslovni i pravni fakultet.

Ranger, S. (2020). *What is the IoT? Everything you need to know about the Internet of Things right now*. Retrieved from https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/

RIOT. (2020). Retrieved from riot-od.org: https://www.riot-os.org/

Thingsquare. (n.d.). *The Thingsquare IoT Platform*. Retrieved from https://www.thingsquare.com/iot-platform/#uses

Tsiftes, N. (2017, Oct 11). *License*. Retrieved from github.com: https://github.com/contiki-ng/contiki-ng/wiki/License

Tutorialspoint. (2020). *5G - Advantages & Disadvantages.* Retrieved from tutorialspoint.com: https://www.tutorialspoint.com/5g/5g_advantages_disadvantages.htm

Ubuntu. (2020). *Ubuntu Core documentation.* Retrieved from Ubuntu for IoT Developers documentation: https://docs.ubuntu.com/core/en/

Warwick, T., & et al. (2018, Jan 30). *An overview of Windows 10 IoT.* Retrieved from Microsoft: https://docs.microsoft.com/en-us/windows/iot-core/windows-iot

Z.P. (2020, Jun 25). *Introduction.* Retrieved from Zephyr Project: https://docs.zephyrproject.org/latest/introduction/index.html

Zephyr. (2020). *Zephyr - The Linux Foundation Project.* Retrieved from Zephyr: https://www.zephyrproject.org/

Zizzo, G. (2020, Apr 19). *AWS based IoT Virtual Environmental Station using RIOT-OS.* Retrieved from hackster.io: https://www.hackster.io/gianmarcozizzo/aws-based-iot-virtual-environmental-station-using-riot-os-1bd69d

Received for publication: 22.05.2020
Revision received: 05.06.2020
Accepted for publication: 01.07.2020

How to cite this article?

Style – **APA** *Sixth Edition:*

Cekerevac, Z., Dvorak, Z., & Pecnik, T. (2020, July 15). Top seven IoT operating systems in mid-2020. (Z. Cekerevac, Ed.) *MEST Journal, 8*(2), 47-68. doi:10.12709/mest.08.08.02.06

Style – **Chicago** *Sixteenth Edition:*

Cekerevac, Zoran, Zdenek Dvorak, and Tamara Pecnik. 2020. "Top seven IoT operating systems in mid-2020." Edited by Zoran Cekerevac. *MEST Journal* (MESTE) 8 (2): 47-68. doi:10.12709/mest.08.08.02.06.

Style – **GOST** *Name Sort:*

Cekerevac Zoran, Dvorak Zdenek and Pecnik Tamara Top seven IoT operating systems in mid-2020 [Journal] // MEST Journal / ed. Cekerevac Zoran. - Belgrade – Toronto : MESTE, July 15, 2020. - 2 : Vol. 8. - pp. 47-68.

Style – **Harvard** *Anglia:*

Cekerevac, Z., Dvorak, Z. & Pecnik, T., 2020. Top seven IoT operating systems in mid-2020. *MEST Journal,* 15 July, 8(2), pp. 47-68.

Style – **ISO 690** *Numerical Reference:*

Top seven IoT operating systems in mid-2020. **Cekerevac, Zoran, Dvorak, Zdenek and Pecnik, Tamara.** [ed.] Zoran Cekerevac. 2, Belgrade – Toronto : MESTE, July 15, 2020, MEST Journal, Vol. 8, pp. 47-68.