# A Survey on small IoT Device Operating Systems

##### **Bo Broyles**

Whitworth University

Operating Systems Final Paper

5/16/2023

***Abstract*-** IoT, internet of things, is a recent concept that has begun to take over the technological world. IoT devices have actively been revolutionizing the world as we know if from behind the scenes. From everyday household items to oil rigs off the coast, these devices have grown to hold massive importance over the world’s current ability to function properly. In this paper I will be introducing IoT devices, their capabilities, and their handicaps. I will focus on three of the most prominent operating systems for these devices, TinyOS, Contiki, and FreeRTOS. All built with their own inherent strengths and weaknesses. In particular, I will be taking a look at each OS’s memory management, energy efficiency, and various security concerns. The in-depth analysis on each OS is aimed at increasing awareness of IoT OS requirements and the challenges associated with implementation. In addition, I will touch on related work being done on hardware security measures.

***Index Terms***- Internet of Things, operating systems, TinyOS, Contiki, FreeRTOS, energy efficiency, memory management, Random Access Memory, Read Only Memory

1. Introduction

T

As the internet of things (IoT) continues to grow in prominence, it has been more important than ever to understand what these devices are and how they work. IoT refers to any type or piece of equipment that has an ability to sense, monitor, store, or process information. These devices are all around us whether we recognize it or not. Some common household IoT devices include smart thermostats, ring doorbells, smart locks, smart lightbulbs, and many universal remotes. These devices are also found in many industrial locations as well. Oil drills, wind turbines, car manufacturing plants, and in nearly all medical facilities. By 2025 there are projected to be upwards of 40 billion of these IoT devices worldwide (6). As expected, there are many potential concerns with devices this powerful, influential, and this small, these include but aren’t limited to; hardware constraints, energy constraints (many are battery powered without an ability easily replace), small memory footprints, interoperability, and network connectivity (10). The need for specialized operating systems is exacerbated by the problems listed above. Many of the current solutions are focused on perfecting three of the problems listed above, the three areas are: memory management, energy efficiency, and security concerns. Failing to adhere to the above requirements by the OS can lead to catastrophic attacks and failures. In this paper I will begin by briefly overviewing some of the challenges with creating small IoT device OSs. Then, I will briefly mention some currently proposed hardware solutions. Finally, I will focus on three small IoT OSs, TinyOS, Contiki, and FreeRTOS. More specifically I will take a look at how each of them approaches memory management, energy efficiency, and their security concerns.

1. Inherent Challenges with IoT Devices and Operating System Design

It should come as no surprise that IoT devices designed to operate in small and often isolated locations lead to logistical and design problems. The first being that these devices are often extremely resource constrained. For the purpose of this paper, I am focusing on small IoT devices, containing <10kB of Random Access Memory (RAM) and <100 kB of Read Only Memory (ROM) (from first note source). For reference, the average computer contains 8GB of RAM and 8MB of ROM. The average iPhone contains 4GB of RAM and 4MB of ROM (3). Often times these devices are located in areas where their environment can’t be controlled, so changes in temperature and weather may affect the hardware. Also, their operational environment may be unpredictable so variables such as supply voltage may be variable and must be accounted for. Both variables contribute to the importance of the OSs ability to operate efficiently. If a device has limited battery life, then the efficiency of the OS is of upmost importance. Another issue is the limited capacity of many device memories. A small memory stack means mundane OS tasks must be allocating their memory efficiently and without leakage or wasted space. These devices are often in charge of multiple tasks, with an already limited hardware capacity, variable environmental conditions, and a minimal memory, the implementation of all the required use cases becomes challenging. TinyOS, Contiki, and FreeRTOS all have multiple publicly available packages that are optimized for various use cases. Owners of IoT devices are able to choose which package best fits their needs and in turn can (hopefully) build their OS to possess optimal abilities and efficiency. Because of this another of the major OS responsibilities is the support of mechanisms to use code from third parties. Finally, protecting the security of the device inputs and outputs offers a challenge to efficient OS development. These IoT devices often have specified tasks but must communicate with devices of differing uses, the transfer of data in a secure manner must be considered by the OSs implemented. TinyOS, Contiki, and FreeRTOS all have their own security protocols and potential vulnerabilities. All three of the OSs to be discussed later are open source which means testing is being done on them consistently to find new bugs quicker than attacks can occur. There have been several attacks on small IoT devices in recent years that are important to mention for context. In 2017, Virgin media routers was subject to one of these attacks. A fault in the default password allocation was exploited and the attackers were able to gain access to the IoT devices located on the router’s networks. Resulting in more than 800,000 of their customers left vulnerable and compromised (2). Section 3 will go over some proposed hardware solutions aimed at preventing these attacks.

1. Proposed Hardware solutions/Related Work

When designing security protocols and attack prevention it is important to keep in mind that the lowest level of security represents the security level of the system as a whole. The resource constraints inherent with small IoT devices opens discussion for the implementation of advanced hardware security solutions. Physical and side-channel attacks are a reality for many of the IoT devices and connected networks. Proponents of a hardware-based approach to device security cite the devices often inaccessibility and response time as a cause for needing hardware solutions. Another cause is the resource capacities inherent in the devices. Encoded security measures are therefore limited and hardware solutions offer greater flexibility for various attacks that coded solutions may not have room for. Two proposed hardware solution overviews below implement public physical unclonable function (PPUF)s. PPUFs are physical objects that allow unique in that for every set of inputs they receive the output can be used as a type of digital footprint. These footprints allow for monitoring, and the specific requirement of inputs allows for the implementation of strict security protocols (3). Two solutions below both build off the idea and use of PPUFs.

## A. XOR Network Delay PPUF

The XOR Network Delay PPUF idea is a system of XOR gates set up in a gridded network. The great thing about PPUF is that the intrinsic differences in the manufacturing hardware means that every instance will have it’s own uniqueness for the digital footprints. For this to work properly the system must have extremely precise clocks.

## B. Differential PPUF

The second of the proposed hardware solutions is a Differential PPUF. This solution is also contingent on the manufacturing having variability in it, more specifically, the logic gates having varying response times and delays. Instead of XOR gates this solution incorporates NAND gates and bounces inputted signals in ways that can’t be anticipated. The goal is for the signals to be so random that even the architect can not predict them. A predictable system would be insecure (2).

1. TinyOS, Contiki, FreeRTOS Overview

To further understand where the IoT OS market is and how the operating systems are being implemented in the world I want to further analyze three of the most common OSs for small IoT devices. TinyOS, Contiki, and FreeRTOS all found themselves in the top 7 IoT operating systems in 2020 (6). Three areas that are at the pinnacle of quality small IoT OS design are energy efficiency, memory management, and security. So those are the three areas I will focus on below, starting with memory management.

## A. Memory Management

The importance of memory management in IoT devices can not be overstated. Small devices are expected to have less than 10kB of RAM and <100 kB of ROM.

TinyOS is an event driven monolithic operating system written in nesC (based on C, optimized for memory limits). There are many important memory components hard coded. TinyOS uses unstacked C. Unstacked C allows for the translation of a multithreaded program to be broken into stackless threads. In turn memory is freed up without the presence of the thread stacks (8). Memory can also be allocated dynamically through the use of TinyAlloc, which is a built in component accessed through the MemAlloc interface (8). TinyAlloc double allocates memory in flash memory. Then a pointer handle for the virtual address that is being allocated is returned. There is also a dereferencing function that dereferences the virtual address and frees the actual address for memory. Both components are used to utilize all available memory as efficiently and securely as possible.

Contiki comes with a C library which provides a set of functions that can be used for allocation and de-allocation of memory (8). The functions, memb\_alloc() and memb\_free() are used for the allocating and deallocating. While the function mmem() is used to handle the potential memory fragmentation that can be detrimental to small IoT devices. Contiki also supports the use of protothreads. These allow for threads to be created without stacks, drastically decreasing the memory requirements of the OS. One of the reasons Contiki has become so popular is because of the ability to use protothreads to minimize memory overhead and fragmentation.

FreeRTOS is simpler in it’s approach to memory management. By utilizing three heap functions, FreeRTOS is able to dynamically allocate memory depending on the needs of the system. FreeRTOS also has very small memory minimum requirements. However, the OS uses multithreading, and in-turn allocates a stack to each thread in the system. Contiki and TinyOS operate on a FCFS basis, FreeRTOS is the only OS that uses a priority based round robin method. Because of this there are times when threads requiring large burst times are retaining memory stacks causing the system to get low on memory and fragmentation to occur. Below is a chart that compares the minimum memory overhead that each of the three operating systems require.

A picture containing text, screenshot, diagram, plot

Description automatically generated

(11)

## B. Energy Efficiency

The small nature of Iot devices and resource constraints, coupled with the often mass implementation in difficult to reach places intensifies the need for proper energy efficiency. your paper receives number of critical remarks.

TinyOS is unique in that it takes advantage of software thread integration (STI) to transform idle time into busy time. This approach minimizes the time the OS spends waiting for things such as I/O and allows for the completion of various waiting tasks. By completing tasks quicker the device can enter low power mode sooner. Because TinyOS is preemptive with an event scheduler, threads are able to run concurrently which also mitigates idle time and preserves energy that may be very important for the device’s battery life.

Contiki has a different approach involving the event queue. The only built in method to enter low power mode in Contiki is by checking the size of the event queue and entering a sleep mode if empty. Because the devices are consistently receiving and interacting with data the event queue is always changing in size and the few moments when the queue is empty the system gets put in low power mode. With the long life spans expected out of most IoT devices, the low power mode time becomes very important.

FreeRTOS is similar to Contiki in how it handles energy efficiency. FreeRTOS is the only of the three OS I’m looking at that allows for processes to be interrupted (this is because it takes into account priority). The idle-task used by the OS to put the device in low-power mode is given the lowest priority. If any threads arrive then they are given a priority higher than the idle-task. The idle-task will always have the lowest priority.

## C. Security

There is hardly a talk about modern technology that doesn’t involve security. The resource constraints and vulnerability to physical attacks. TinyOS, Contiki, and FreeRTOS all have different approaches and vulnerabilities. There are some proposed solutions that are applicable regardless of OS. One of these is diversification. The high-level idea is creating uniqueness among interfaces to prevent one attack from compromising an entire network of devices (9). The following operating system specific security information is below:

Many of the security vulnerabilities with IoT involve the physical sensors they employ. TinyOS has a strict event-based OS that allows for tightly monitored sensor control to tighten up security. Another layer of security is the compatibility with hardware that TinyOS allows. Many security measures can be taken on a hardware level to prevent physical and side-channel attacks.

Extensive testing has been done on the security of Contiki. One static analysis was run and found that Contiki had a bug in their memory deallocation that caused the system to crash (2). Bugs such as these are constantly getting fixed. Being open-source, the OSs being analyzed are having tests run on them frequently to try and fix bugs such as these. Contiki also offers a tool called contikisec which adds an additional layer of network level security. It is important for the OSs to be secure against several methods of attack.

FreeRTOS comes with many libraries that provide security measures for implementors. RTOS is also one of the most popular IoT operating systems so the packages that come with it are always improving. The great thing about the open source OSs is the ability for people all over the world to provide security flaws and solutions.

1. CONCLUSION

In conclusion, as the Internet of Things (IoT) continues to expand, the need for specialized operating systems for small IoT devices becomes crucial. These devices face inherent challenges such as limited resources, unpredictable environments, and the requirement for efficient memory management, energy efficiency, and security. To address these challenges, I analyzed three prominent IoT operating systems: TinyOS, Contiki, and FreeRTOS. In terms of memory management, TinyOS utilizes unstacked C and dynamic memory allocation through TinyAlloc to optimize memory usage. Contiki uses a C library with functions for memory allocation and deals with potential fragmentation using mmem(). FreeRTOS dynamically allocates memory with three heap functions but faces some fragmentation issues due to its priority-based round-robin scheduling. For energy efficiency, TinyOS utilizes software thread integration (STI) to minimize idle time, while Contiki focuses on entering low power mode during moments of empty event queues. FreeRTOS allows for process interruption and employs an idle-task with the lowest priority for low-power mode. In terms of security, TinyOS emphasizes event-based control and compatibility with hardware security measures. Contiki has undergone extensive testing and offers additional network-level security tools. FreeRTOS provides security libraries and benefits from community contributions in identifying and addressing vulnerabilities. Overall, the analysis of these operating systems highlights their approaches to memory management, energy efficiency, and security in the context of small IoT devices.

Acknowledgment

This paper was written in accordance with the expectations expressed by Professor Qian Mao for the final paper in CS475W Operating Systems. Information was gathered from the resources listed below and organized to emphasize the importance of IoT devices and the difficulties in building and maintaining their operating systems.

References

1. Celik, Z. B., Tan, G., & McDaniel, P. (2018). IOTGUARD: Dynamic Enforcement of Security and Safety Policy in Commodity IoT. In Proceedings/Conference Name.
2. McBride, J., Arief, B., & Hernandez-Castro, J. (Year). Security Analysis of Contiki IoT Operating System. In Proceedings/Conference Name (2018).
3. Xu, T., Wendt, J. B., & Potkonjak, M. (Year). Security of IoT Systems: Design Challenges and Opportunities. In Proceedings/Conference Name.
4. Javed, F., Afzal, M. K., & Majeed, M. U. (2018). Internet of Things (IoT) Operating Systems Support, Networking Technologies, Applications, and Challenges: A Comparative Review. IEEE Communications Surveys & Tutorials, 20(3), 2062-2100.
5. Antony, A., & Sarika, S. (2020). A Review on IoT Operating Systems. International Journal of Computer Applications, 176(24), 33. ISSN 0975-8887
6. Cekerevac, Z., Dvorak, Z., & Pecnik, T. (2020). TOP SEVEN IoT OPERATING SYSTEMS IN MID-2020. MEST Journal, 8(8), 47. DOI: 10.12709/mest.08.08.02.06.
7. Abuserrieh, L., & Alalfi, M. H. (Year). A Survey of Analysis Methods for Security and Safety Verification in IoT Systems. In Proceedings/Conference Name
8. Musaddiq, A., Zikria, Y. B., Hahm, O., Yu, H., Bashir, A. K., & Kim, S. W. (Year). A Survey on Resource Management in IoT Operating Systems. [Journal/Conference Name], Volume(Issue)
9. Koivunen, L., Rauti, S., & Leppanen, V. (Year). Applying Internal Interface Diversification to IoT Operating Systems. [Journal/Conference Name], Volume(Issue)
10. Zhang, Z.K., Cho, M.C.Y., Wang, C.W., Hsu, C.W., Chen, C.K., Shieh, S., IEEE Fellow. (2014). IoT Security: Ongoing Challenges and Research Opportunities. [Journal/Conference Name], Volume(Issue)
11. Challouf, S., Kriaa, L., & Saidane, L. A. (2017). "Comparison of IoT Constrained Devices Operating Systems: A Survey." In Proceedings of the 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications.

Author

**Author** – Bo Broyles, Computer Science Major, Class of 2023, Whitworth University.