DOI 10.5162/EUROSENSORS2025/MP58

Power consumption analysis of an OpenThread border router in a Wireless Sensor Network for Precision Agriculture

Andrea Buratti [1], Davide Polese[2] and Alessandro Checco[1]
[1] Computer Science Department, University of Rome "La Sapienza", Italy
[2] Istituto per la Microelettronica e Microsistemi (IMM) Consiglio Nazionale delle Ricerche, Italy

Summary: We performed an energy consumption analysis of a Wireless Sensor Network (WSN) for precision agriculture using the OpenThread protocol. Focusing on the Raspberry Pi as a border router, we examine strategies for optimizing power usage through peripheral deactivation and CPU frequency adjustments. A custom real-time monitoring system based on an Arduino board and Hall-effect sensor measures power consumption under varying network loads. Experimental results show that lower CPU frequencies do not necessarily reduce power draw and that increased network size raises energy demands. Findings support energy-aware configurations that prioritize sleep scheduling and optimized CPU usage for long-term WSN deployment.

Keywords: Precision Agriculture, OpenThread, energy management.

Introduction

Wireless Sensor Networks (WSNs) represent an efficient and low-cost technology for the monitoring of environmental parameters in agronomical applications and environmental monitoring [1]. Minimizing the power draw of embedded systems is critical, as these architectures are typically intended to operate using renewable energy sources and potentially in remote areas. Effective energy management is particularly critical in battery-powered contexts, where even modest consumption can significantly affect system performance and longevity. The analysis focused on the Raspberry Pi configured as a border router, which accounts for a disproportionately high share of total system energy usage. While identifying power-consuming components is relatively straightforward, developing energy-efficient strategies that preserve system performance is considerably more complex. In this work, we will investigate the energy consumption performance of a WSN based on the OpenThread networking protocol for short-range wireless communication, which claims to provide lower latency, more efficient routing algorithms, and easier interconnectivity with other IP-based networks than other protocols [2].

Experimental setup

Our WSN based on the OpenThread networking protocol is composed by one border router that acts as a bridge between the WSN and the external world and collect all sensor data in a local database, and an increasing number of sensor nodes that can act as router-nodes, i. e. nodes that collect sensor data and route the network messages and end-nodes i. e. nodes that just send their sensor data to the border router by the router-nodes. Router and end nodes are implemented by custom hardware [2], whereas the border router is implemented using a Raspberry Pi with Raspbian OS. This configuration without any optimization will be referred as "baseline".

Peripheral Management

To optimize energy consumption, a targeted strategy was adopted to deactivate all non-essential components required for the operation of the OpenThread daemon. Specifically, the HDMI ports were disabled, given their redundancy in a headless setup, alongside the Wi-Fi interface, which was used solely during the development phase for SSH access.

CPU Configuration and Energy Dynamics

Besides to deactivate unused board peripherals, the optimization of the CPU loads can improve the power consumption. It is generally assumed that higher clock frequencies lead to increased power consumption, empirical evidence suggests that this relationship is not strictly linear. However, Kaup, Gottschling, and Hausheer [3] suggest that reducing CPU frequency may, in some scenarios, results in a greater overall energy usage. To this purpose, we analyzed two power configuration "baseline" and "low-power" in the first configuration we disable all unused peripheral but we do not modify the CPU clock (1.5 GHz) and governor; in the second configuration, we set CPU clock at 600 MHz, the CPU governor has be modified to energy saving mode, and the arm_boost parameter of the boot file has be set to avoid the automatic overclock of CPU and GPU as usual in the Raspberry Pi models.

Profiling Tools and Methodology

A custom monitoring system was developed, comprising an Arduino board and a Hall-effect

sensor that allows high-resolution real-time profiling of absorbed power and current (with a resolution of 0.01 W and 0.01 A respectively), and CPU load percentage on a per-second basis. This solution enables direct, real-time measurement of the Raspberry Pi's current consumption and generates corresponding visualizations, with the capability for live display.

Two scripts were developed to manage the Raspberry Pi's power-saving configuration. The first script reduces CPU frequency and disables all non-critical interfaces, while the second restores the system's default behavior. Additional optimizations at boot include disabling the *arm_boost* parameter, onboard LEDs, and the *hdmi_blanking* option, which otherwise persistently attempts to detect a display interface.

Increasing network configuration

To evaluate the power consumption as function of the number of the nodes composing the network, a profiling script was initiated, and sensors were added incrementally. After the arrival of the first CoAP packet from each newly added sensor, a waiting period of approximately one minute was used to ensure system stabilization. This setup allows us to analyze power consumption as a function of both time and network size.

Results

In the "baseline" configuration, the Raspberry Pi consumes 4.4 W on average (σ = 0.01 W), while "low-power" mode slightly increases usage to 4.46 W (σ = 0.01 W). As shown in Fig. 1, lower CPU frequencies demand more current for equivalent workloads, highlighting the need for sleep-optimized CPU scheduling.

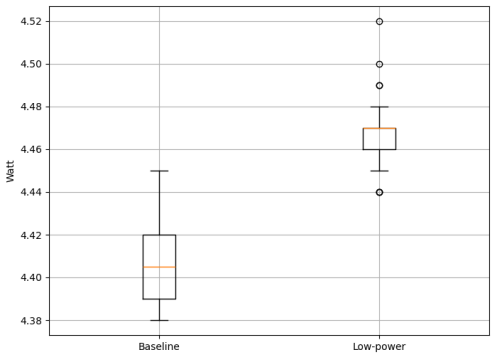

Fig. 1: Power consumption of two energy profiles.

After establishing this baseline and confirming better efficiency at higher frequencies, we evaluated power use as the network size increased. Each sensor sends a CoAP request every 2–3 s. Fig. 2 shows power varying from 4.9 W to 5.10 W with six sensors, and standard deviation rising from 0.04 W to 0.11 W, reflecting, as expected, the router's growing workload in terms of data collection and network management.

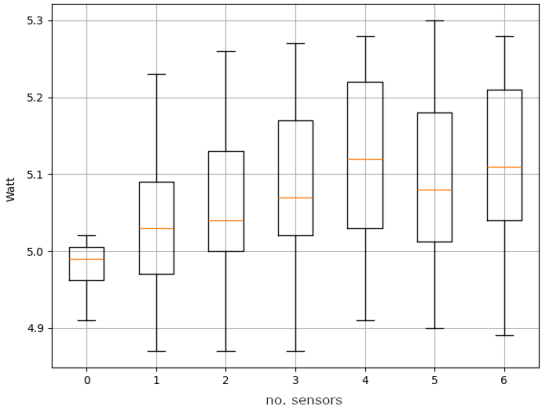

Fig. 2: Power consumption vs number of nodes.

Conclusions

Contrary to common assumptions, operating the Raspberry Pi border router at lower CPU frequencies does not necessarily yield energy savings. In fact, higher frequencies offered better energy efficiency. Additionally, power consumption scaled with network size, as increased communication demands from additional sensors led to measurable current fluctuations. Future work will focus on profiling the energy consumption of individual sensor nodes and comparing the Raspberry Pi's performance with alternative hardware platforms for the border router, as well as further investigating optimized peripheral management and CPU sleep strategies, aiming to identify more power-efficient solutions for long-term WSN deployment in agriculture.

Acknowledgments

We thank the PlumJuice team and Vladimiro Paschali for their invaluable help. This work has been partially funded by the European Union, Next Generation EU programme, PRIN project 202293CWWS.

References

[1] D. Thakur, Y. Kumar, A. Kumar, and P. K. Singh, *Applicability of Wireless Sensor Networks in Precision Agriculture: A Review*, vol. 107. Springer US, 2019.

[2] A. Checco, V. Fabiani, M. Palmisano, and D. Polese, "Network connectivity and sensor response of an openthread wsn platform for crop monitoring," *IEEE Transactions on AgriFood Electronics*, 2024.

[3] F. Kaup, P. Gottschling, and D. Hausheer, "Powerpi: Measuring and modeling the power consumption of the raspberry pi," in *39th Annual IEEE Conference on Local Computer Networks*, pp. 236–243, IEEE, 2014.