Energy Consumption of Common Sorting Algorithms

Daniel Mayr
dg.mayr@gmail.com

***Abstract*—With the pervasive presence of global warming and the global energy crisis, individuals are becoming increasingly cognizant of their energy consumption and exploring methods to save energy. This research study undertakes an analysis of the energy consumption associated with commonly used sorting algorithms, with the objective of discerning potential energy savings at a low implementation level.**

The experiments were carried out on microcontrollers featuring diverse architectures. A comprehensive evaluation was conducted over an extended duration to gauge the energy consumption, number of iterations over time for each algorithm, thereby enabling the determination of the average energy consumption.

Among the algorithms subjected to evaluation, Quick Sort emerged as the most energy-efficient option, demonstrating a significantly lower energy consumption compared to other algorithms.

I. Introduction

The global energy crisis, which began in 2021, has increased consumer and corporate focus on energy usage. This has resulted in pressure on governments, the public, and researchers [1], [2] to develop and discover new energy sources, infrastructure, and ignited substantial investments in renewable energy sources such as wind and solar power as alternatives to fossil fuels. This positive effect has the potential to minimize our environmental impact while also creating sustainable energy resources for the future and long tail business opportunities.

IT systems consume a significant amount of electricity on a daily basis because their applications are pervasive in a variety of settings, including homes, businesses, healthcare, and science. As a result, there is a considerable amount of publicly funded research being conducted to improve IT system efficiency, as well as the effective use and allocation of processing power.

Most research in this area has concentrated on large-scale or black box systems, which undoubtedly provide significant potential savings. However, we aim to explore potential energy savings at the implementation level of algorithms. These algorithms we investigated are widely used and could significantly impact energy consumption through small changes in their software implementation.

Common Sorting Algorithms

In the realm of computer science, sorting data using algorithms poses a persistent challenge [3]. There is no universal, perfect algorithm for sorting as they often vary in run time and memory consumption and can only be approximated. Therefore, multiple implementations of sorting strategies exist and here is a listing of commonly used sorting strategies which we will also evaluate on their energy consumption.

TABLE I
Complexity of Sorting Algorithms

Algorithm	Best	Average	Worst	Memory
Bubble Sort	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$
Selection Sort	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$
Insertion Sort	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$
Quick Sort	$\mathcal{O}(n \log n)$	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(\log n)$

II. Runtime Complexity and Energy Efficiency

Table I presents various widely used implementations of sorting algorithms. These algorithms are frequently chosen due to their relatively straightforward implementation, overall good performance, and/or their limited memory consumption. Some of the selected algorithms also provide a stable result, indicating that the best/worst and average run time and memory consumption do not fluctuate significantly. Consequently, the choice of an appropriate sorting algorithm depends on factors such as the list size, the inherent characteristics of the data, and the desired efficiency requirements in terms of time and space. However, what if the primary objective is to minimize energy consumption? When energy efficiency is the paramount concern, traditional sorting algorithms may not be the optimal solution. These algorithms are typically designed to prioritize speed and memory usage, with less emphasis on energy consumption. However, in certain applications, such as battery-powered devices or embedded systems, reducing energy consumption is crucial to prolonging battery life and enhancing overall system efficiency [4]. If the overall efficiency of the system can be improved, it may lead to potential long-term energy savings for the appliance, including but not limited to IoT [5].

Evaluation Method

In this research, we want to analyze whether the commonly used sorting algorithms show a difference in their energy consumption. For this, we have designed a real world evaluation setup which will use several iterations of the same sorting algorithm on randomly filled arrays with fixed lengths. With a high number of iterations, we expect to reach the average run time complexity of the given algorithm.

To reduce the impact of factors like operating system overhead, multi-threading, and preemption, we opted to measure

the energy consumption of the algorithms using microcontrollers. In this study, we will use an Arduino UNO R3 (ATMega328P) [6] and an Arduino Nano ESP32 (Espressif32) [7]. The two devices are built on different platforms, each with its own unique architecture and clock speed for processor as well as memory. Both devices can be programmed with C++ for our purpose with the same implementation of the sorting algorithms with the Arduino tool-chain. We will use the *bxparks/AceSorting* [8] library which can be simply integrated, is very well tested and has already done a very good assessment of run time and space complexity on a variety of devices.

To identify the average energy consumption of a given algorithm A on a given device D, we generate a random integer array with the length $k = 500$, sort it with the given algorithm in a loop. We will loop for an hour and we will track the iterations n regularly. The length $k = 500$ is set to overcome the flash memory limitations of the Arduino Uno and we aim to use the same parameters also on the Arduino Nano ESP32.

To measure the energy consumption of the algorithm, we can measure the whole device as black box, as both devices are supposed to only execute our code and therefore should be free of any side effects of other executions. For this, we will connect a USB power meter [9] between power supply and mirocontroller, as both devices can be powered via USB. We will track the energy W regularly throughout the hour.

With the described setup, we aim to answer the following questions:

(1) Will there be algorithms which consume more or less energy on average and how does this relate to their average run time complexity?

(2) Will energy W and number of iterations n be linear? There could be side effects like overheating, not enough power supply etc. which might slow down execution or decrease the throughput of iterations over time.

(3) Are there differences in energy consumption related to device and device architecture?

For the experiment, we ran each sorting algorithm of the table I on Arduino UNO first, each of at least one hour and regularly tracked energy W and iterations n. After this, we repeated the same with the Arduino Nano ESP32. For each execution, we compiled and uploaded the code from scratch and the device waits for a new line input via serial to start running the sorting loop. We set iterations, timer and energy to 0 in parallel to sending the new line character. This ensured that we had no boot up or upload energy consumption mixed into the measurements. The full code is published and can be found on GitHub [10].

III. Results and Discussion

For each execution per device and per sorting algorithm, we produced a data set (also on GitHub [10]) which records the energy consumption of the device during the loop at $t = \{5min, 10min, 15min, 20min, 30min, 45min, 60min\}$. We merged in the data with timestamps when an adequate step milestone of iterations has been looped and logged by the device on the serial output with the given timestamp. In figure 1 is a visualization of the Selection Sort measures on the Arduino UNO.

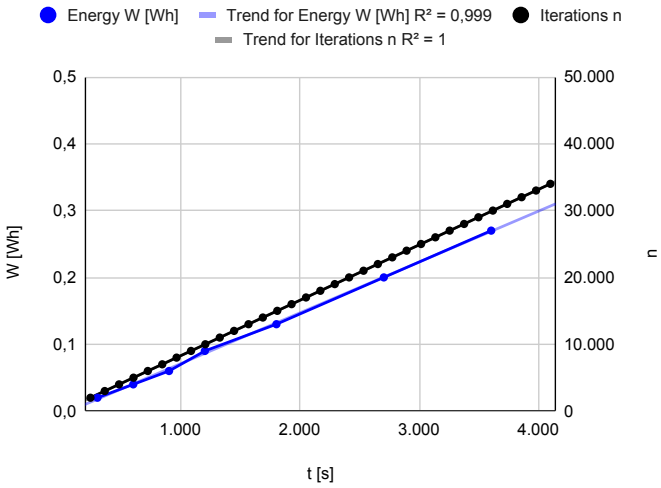

Fig. 1. Energy W and Iterations n on Arduino UNO with Selection Sort

When looking at the data, the number of iterations n shows linear growth with $R^2 = 1$, as well as the consumption of energy W which analogously seems to grow on a linear scale with $R^2 = 0,999$. This indicates that the average energy consumption and number of iterations over time remains stable.

This upholds for all sorting algorithms on both devices, as energy consumption and number of iterations satisfy a linear growth trajectory.

With these findings, we can calculate per sorting algorithm and device the average iterations which can be performed in an hour (n/h), the energy consumed per hour (W/h) and resulting in the energy consumed by 1000 iterations $(W/1000)$. The results per sorting algorithm can be found in table II for the Arduino UNO and in table III for the Arduino Nano ESP32 .

TABLE II
Results on Arduino UNO

	n/h	W/h	$W/1000$
Bubble Sort	10.568	$0,26Wh$	$24,60mWh$
Selection Sort	21.886	$0,26Wh$	$11,89mWh$
Insertion Sort	29.883	$0,27Wh$	$9,04mWh$
Quick Sort	65.527	$0,25Wh$	$3,82mWh$

Out of the four selected sorting algorithms, Selection Sort is supposed to perform worse than all other sorting algorithms given the complexity in table I is for all cases $\mathcal{O}(n^2)$. In contrast, Bubble Sort seems to be the worst performing sorting algorithm on UNO as well as Nano ESP32. On both devices, Quick Sort is the best performing sorting algorithm, which on average has a $\mathcal{O}(n^2)$ complexity and also is more run time

TABLE III
RESULTS ON ARDUINO NANO ESP32

	n/h	W/h	$W/1000$
Bubble Sort	476.948	$0,27Wh$	$0,57mWh$
Selection Sort	638.298	$0,27Wh$	$0,42mWh$
Insertion Sort	1.628.289	$0,27Wh$	$0,17mWh$
Quick Sort	5.301.336	$0,28Wh$	$0,05mWh$

efficient than the others listed. When looking at the energy consumption, both devices use the maximum of power which can be supplied. As there is no significant difference in energy consumed per hour, the number of iterations executed in an hour is the significant denominator.

This leads to the fact that Quick Sort consumes on average less energy per 1000 iterations than any of the other sorting algorithms in table I. The energy savings are quite significant, as Insertion Sort - which is the second ranking in iterations per hour - only achieves 46% of the iterations compared to Quick Sort on the Arduino UNO and only 31% on Arduino Nano ESP32.

IV. CONCLUSION

Comparing the results of the energy consumption of sorting algorithms on two different devices states that Quick Sort with an average runtime complexity of $\mathcal{O}(n \log n)$ consumes up to 54% less energy than any listed sorting algorithm with an average runtime complexity of $\mathcal{O}(n^2)$. While it seems true that the class of $\mathcal{O}(n \log n)$ sorting algorithms might be more runtime efficient and also energy efficient than the class of $\mathcal{O}(n^2)$ sorting algorithms, there is a noticeable and yet significant difference of energy consumption within the class of $\mathcal{O}(n^2)$. Therefore, the hypothesis (1) if energy efficiency and average runtime complexity relate, has shown a positive outcome when comparing the complexity classes of sorting algorithms, but not necessarily within the same complexity class.

Within the evaluation and the selected evaluation method, (2) energy consumption W and number of iterations n has been stable and linear over time. The temperature of the devices was not impacted significantly ($\pm 2K$) and the power supply has been monitored to be stable a maximum throughout the evaluations. We conducted another test to measure idle energy consumption of both devices. In the span 12 hours, the Arduino UNO consumed on average $0,26W$ which is the same consumption as performing the selected sorting algorithms. This seems to be indented, as the Arduino UNO is built for experimental purposes to control and monitor inputs and outputs via GPIO and more meant to be a development board where energy consumption was not focused at the time it was created. In contrast, the Arduino Nano ESP32 has consumed only $0,16W$ in the span of 12 hours when idle, so running at full performance consumes up to $0,12W$ more.

As expected, there has been a huge difference in performance, as the processing power of the Arduino Nano ESP32 is higher than the aged Arduino UNO. But, whilst the sorting algorithms achieve performance in the same order on both devices, there are noticeable variations in the performance of each sorting algorithms, when comparing the best performing. Insertion Sort on Arduino UNO performs at 46% of the Quick Sort level, while on Nano ESP32 only at 31%. Same for Selection Sort (33% and 12%) and Bubble Sort (16% and 9%). These differences might be observed because of the different CPU architecture, but also due to different architecture on memory and clock speeds.

Further research could be done on investigating other $\mathcal{O}(n \log n)$ class sorting algorithms and whether these consume more or less energy compared to Quick Sort to identify the best possible algorithm for a given environment [11]. Secondly, as we identified as well, the idle consumption of the selected devices was very high when compared to the energy consumed under load. In these scenarios and on these specific devices it is more relevant to reduce idle time rather than focusing on the consumption of a given algorithm. All code and data sets are publicly available on GitHub [10].

REFERENCES

[1] M. Rambabu, R. S. S Nuvvula, P. P. Kumar, K. Mounich, M. E. Loor-Cevallos, and M. Gupta, "Integrating renewable energy and computer science: Innovations and challenges in a sustainable future," in *2023 12th International Conference on Renewable Energy Research and Applications (ICRERA)*, 2023, pp. 472–479.

[2] V. Chauhan, A. Chauhan, S. Kapoor, S. Agrawal, and R. R. Singh, "Motivation for green computer, methods used in computer science program," in *2011 National Postgraduate Conference*, 2011, pp. 1–5.

[3] Y. Yang, P. Yu, and Y. Gan, "Experimental study on the five sort algorithms," in *2011 Second International Conference on Mechanic Automation and Control Engineering*, 2011, pp. 1314–1317.

[4] L. M. Feeney, R. Hartung, C. Rohner, U. Kulau, L. Wolf, and P. Gunningberg, "Towards realistic lifetime estimation in battery-powered iot devices," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3131672.3136985

[5] I. Y. Hegazy, A. Salah, and A. Kamel, "Enhancing energy consumption in iot," in *Proceedings of the 9th International Conference on Software and Information Engineering*, ser. ICSIE '20. New York, NY, USA: Association for Computing Machinery, 2021, p. 140–145. [Online]. Available: https://doi.org/10.1145/3436829.3436832

[6] A. S.r.l. (2023) Arduino uno r3 - product reference manual. [Online]. Available: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

[7] Arduino. (2024) Arduino nano esp32 - product reference manual. [Online]. Available: https://docs.arduino.cc/resources/datasheets/ABX00083-datasheet.pdf

[8] B. T. Park. (2021) Acesorting. [Online]. Available: https://github.com/bxparks/AceSorting

[9] Yojock. (2023) Energy consumption of common sorting algorithms. [Online]. Available: https://manuals.plus/yojock/j7-c-usb-c-tester-usb-power-meter-2-in-1-digital-multimeter-manual.pdf

[10] D. Mayr. (2024) Energy consumption of common sorting algorithms. [Online]. Available: https://github.com/mayrd/energy_sorting_algorithms

[11] M. Marcellino, D. W. Pratama, S. S. Suntiarko, and K. Margi, "Comparative of advanced sorting algorithms (quick sort, heap sort, merge sort, intro sort, radix sort) based on time and memory usage," in *2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI)*, vol. 1, 2021, pp. 154–160.