Empirical Power Analysis of Embedded Devices

FOSSketeers

Buğra Kağan ACAR, Furkan ŞİMŞEKLİ, İlker AVCI

Supervisor

Assoc. Prof. Harun ARTUNER

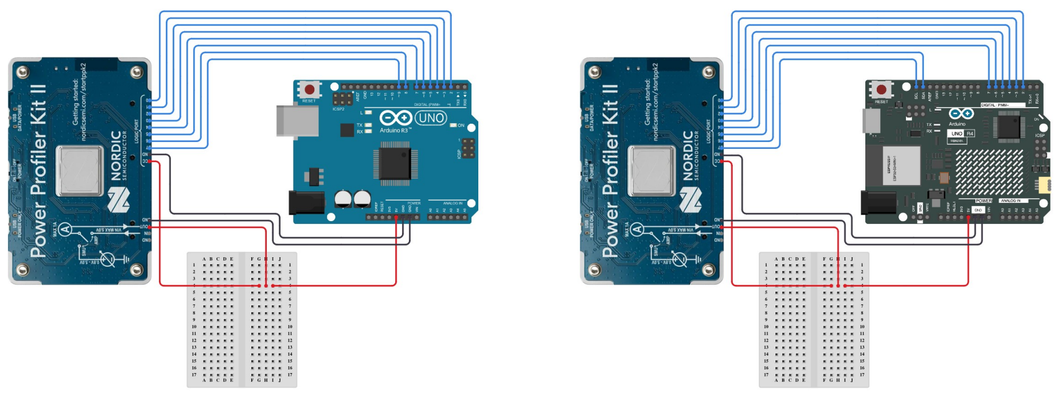
Department of Computer Engineering, Hacettepe University

Introduction

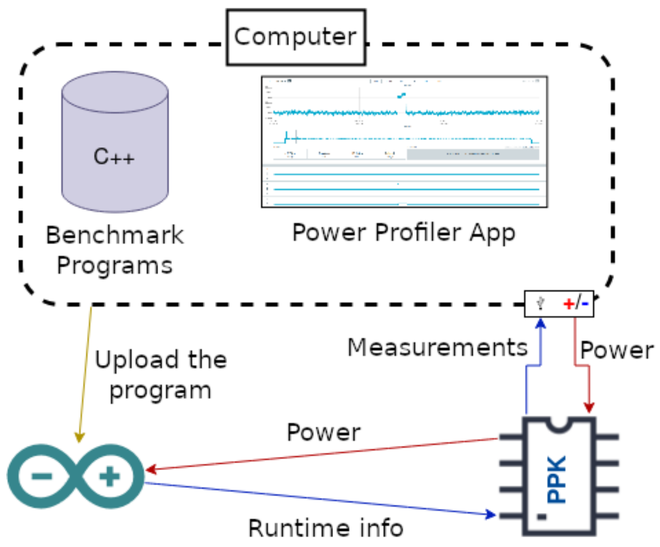
Energy consumption is a topic that never gets old in our lives. As programmers, we tend to choose algorithms based on their time and space complexity, but we rarely know how much energy these algorithms actually consume. In this project, we explored the world of energy consumption on embedded devices to see whether theory and reality align.

Solution Methodology

We used Arduino UNO R3 and R4 as test platforms, with the Nordic® Power Profiler Kit II (PPK) in sourcemeter mode to supply power and measure current draw. The PPK collected real-time energy consumption data during the experiment, as shown in the figure below.



We developed a robust software environment with PlatformIO to manage benchmarks across devices. This environment streamlined the process of compiling, deploying, and debugging, enabling consistent and efficient experimentation. Our modified nRF desktop application played a central role in synchronizing measurement sessions. A custom visualization tool processed the output data, classified algorithm types, and performed aggregate analysis.



To investigate the alignment between theoretical time complexity and real-world energy consumption, we focused primarily on 10 sorting algorithms tested with varying array sizes and data types. For statistical correlation, only Sort benchmark was used. However, our infrastructure supported further experimentation: cryptographic benchmarks (ChaCha, AES, AEADs), various general-purpose algorithms (search, GCD, matrix ops, Dijkstra etc.), and code generated by large language models (ChatGPT, Gemini, Claude) based on a single, consistent prompt. These broader tests, though not included in the main analysis, highlight the extensibility of our framework and demonstrate that our effort extended beyond minimal course requirements.

TABLE I: TEST DEVICES AND THEIR SPECIFICATIONS

Cihaz	Microprocessor	Architecture	Clock Speed	Memory	EEPROM	Flash Mem.
R3	ATmega328P	Atmel AVR®(8-bit)	16MHz	2 KB	2 KB	32 KB
R4	Renesas RA4M1	ARM®v7E-M (32-bit)	up to 48MHz	32 KB	8 KB	256 KB

Results and Discussion

Our analysis revealed several key findings. Hardware architecture significantly influenced algorithm performance; for example, **Radix Sort** outperformed other algorithms on the R4 but was the worst performer on R3 due to AVR's hardware incapability for the division instruction. **Quick Sort** was the fastest algorithm in both devices.

Among the AI models evaluated, ChatGPT produced the most energy-efficient code, maintaining a **3% efficiency** advantage over Gemini and Claude.

We also observed a **strong correlation** ([0.98, 1.00], $p < 0.0001$) between power consumption and time complexity, aligning with intuition, though this contradicts earlier claims in the literature (Bunse et al.). In terms of encryption, **Ascon128** proved optimal for the R3, surpassing *Acorn128*, *AES*, and *ChaCha*. On the R4, **ChaCha20-Poly1305** emerged as the best performer, outperforming *Acorn128*, *Ascon128*, and a non-hardware-accelerated *AES* implementation.

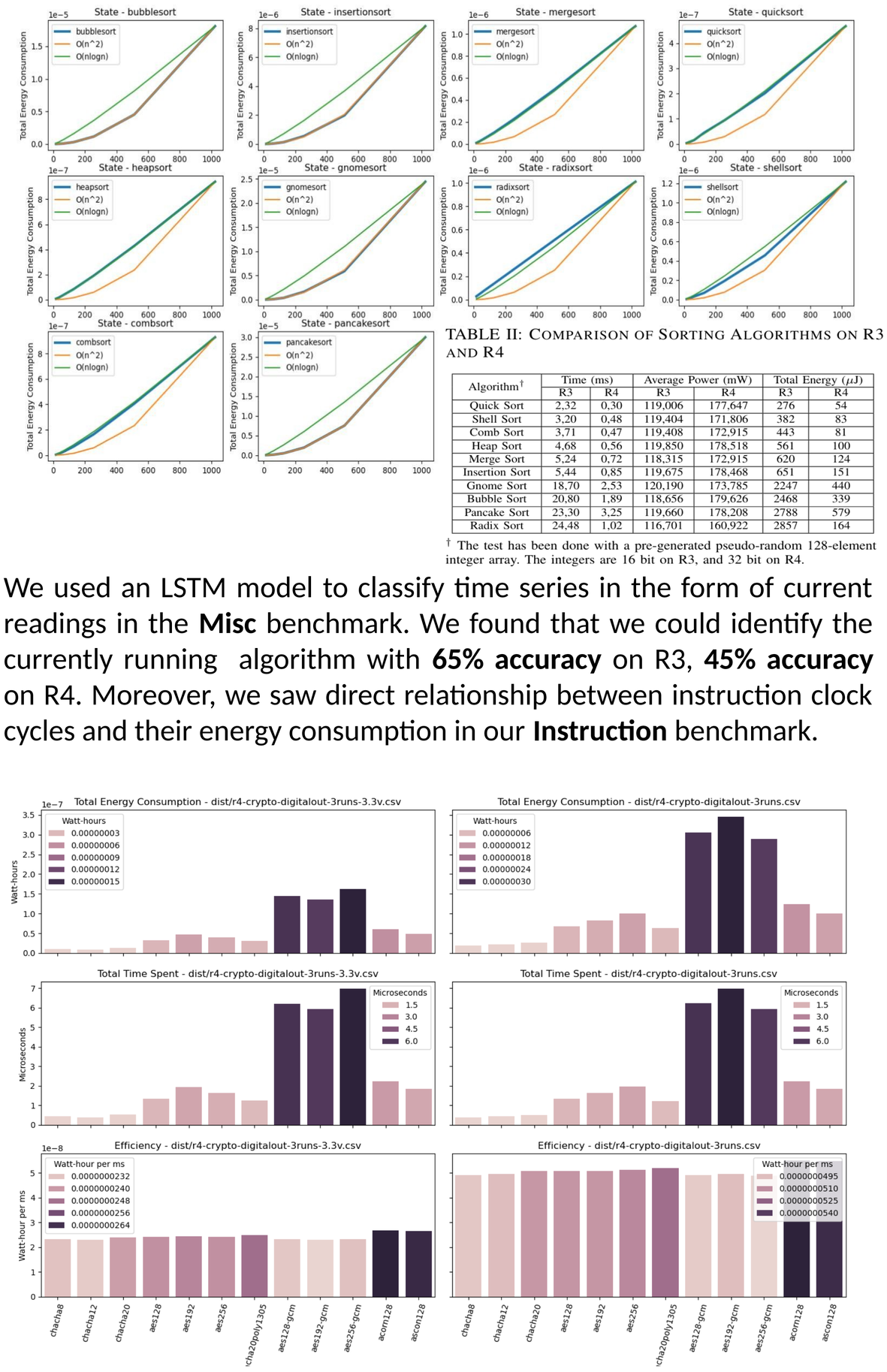


TABLE II: COMPARISON OF SORTING ALGORITHMS ON R3 AND R4

Algorithm†	Time (ms)		Average Power (mW)		Total Energy (μJ)	
	R3	R4	R3	R4	R3	R4
Quick Sort	2,32	0,30	119,006	177,647	276	54
Shell Sort	3,20	0,48	119,404	171,806	382	83
Comb Sort	3,71	0,47	119,408	172,915	443	81
Heap Sort	4,68	0,56	119,850	178,518	561	100
Merge Sort	5,24	0,72	118,315	172,915	620	124
Insertion Sort	5,44	0,85	119,675	178,468	651	151
Gnome Sort	18,70	2,53	120,190	173,785	2247	440
Bubble Sort	20,80	1,89	118,656	179,626	2468	339
Pancake Sort	23,30	3,25	119,660	178,208	2788	579
Radix Sort	24,48	1,02	116,701	160,922	2857	164

† The test has been done with a pre-generated pseudo-random 128-element integer array. The integers are 16 bit on R3, and 32 bit on R4.

We used an LSTM model to classify time series in the form of current readings in the **Misc** benchmark. We found that we could identify the currently running algorithm with **65% accuracy** on R3, **45% accuracy** on R4. Moreover, we saw direct relationship between instruction clock cycles and their energy consumption in our **Instruction** benchmark.

Acknowledgements

This project was completed within the context of BBM479-480 Design Project courses in Hacettepe University, Faculty of Engineering, Department of Computer Engineering.