

A Compute Budget Set by the Radio: Energy Analysis of On-Device Processing Against Raw Transmission in Battery-Powered IoT Nodes

Madhu Sahu

Assistant Professor, Department of Civil Engineering, Kalinga University, Raipur, Chhattisgarh.

Email: madhusahu95289@gmail.com

ABSTRACT

A battery-powered sensor node can transmit raw samples or process them on the microcontroller and transmit a small result. Processing costs energy and transmission costs more per byte, so the choice is an energy trade. We compute node lifetime for both architectures across five radio technologies and derive the quantity a designer actually needs: how much computation the energy saving will pay for. An analytical energy model was built for a node sampling at 100 Hz with a 10 s reporting window, producing 2,000 raw bytes or a 12-byte result, powered by a 2400 mAh cell at 3 V. Radio energy was modelled as a per-byte term plus a fixed per-packet overhead, using representative catalogue values for BLE 5, 802.15.4, LoRa at two spreading factors and NB-IoT. We swept processing cost, reporting interval, and the case where the node transmits both the result and the raw data. On-device processing extended lifetime from 1.02 to 5.06 years on BLE (5.0 times) and from 0.003 to 0.22 years on LoRa SF10 (74 times). The energy saving paid for 17,623 cycles per sample on BLE, 65,539 on 802.15.4 and 7,422,187 on LoRa SF10, respectively 42, 156 and 17,672 times the modelled workload. NB-IoT was the exception: despite a 167:1 payload reduction it gained only 4.6 times, because fixed per-packet overhead left 98.7 per cent of node energy in the radio even with a 12-byte payload. Transmitting the result alongside the raw data forfeited 80 to 99 per cent of the gain across the radios tested. The admissible processing cost is a property of the radio link, computable before any algorithm is written. Where per-packet overhead dominates, payload reduction achieves little and reporting interval is the effective lever.

Keywords: Internet of Things; edge computing; energy budget; low-power wireless; embedded systems; connected systems.

1. INTRODUCTION

Battery-powered sensor nodes are deployed in places where changing a cell is expensive, so lifetime is usually the binding design constraint [1,2]. Two architectures compete. The node can stream raw samples to a gateway and do the analysis there, or it can run the analysis on its own microcontroller and transmit only a result.

The second option is widely advocated under the heading of edge computing, on the reasoning that radios are expensive and microcontrollers are cheap [3-5,19-22]. That reasoning is broadly correct and is rarely quantified for a specific link, with the result that designers know they should process on device but not how much processing they can afford.

Two features of real radio links complicate the arithmetic. The first is that transmission energy is not proportional to payload: every packet carries a fixed cost for wake-up, preamble, synchronisation and, on cellular links, network attachment, and that cost can exceed the

payload-dependent term by orders of magnitude [6,7,16]. A large payload reduction therefore does not produce a proportional energy reduction.

The second is that the energy per byte differs by more than two orders of magnitude across the technologies a designer might choose, from short-range links at a few microjoules per byte to long-range low-rate links at close to a millijoule [10,13-15]. The answer to the architecture question is therefore not a property of the algorithm but of the link.

We set out to produce the number a designer can use at the specification stage: given a radio and a reporting requirement, how many processor cycles per sample may be spent before on-device processing stops saving energy.

1.1 Contributions

We give an analytical energy model including fixed per-packet overhead; node lifetimes for both architectures across five radio technologies; a closed-form compute budget in cycles per sample for each; the sensitivity of the benefit to processing cost; the energy consequence of transmitting the result alongside the raw data; and the interaction between reporting interval and detection latency.

2. SYSTEM MODEL

2.1 Node and Workload

The node samples a single channel at 100 Hz with 2 bytes per sample and reports once per 10-second window, giving 1,000 samples and 2,000 raw bytes per window. The on-device alternative produces a 12-byte result comprising a classification and a small feature vector, a payload reduction of 167:1.

The microcontroller runs at 48 MHz and the modelled signal-processing workload is 420 cycles per sample, representative of windowed feature extraction followed by a small classifier. Following established node-level power modelling practice [16,24], energy per window decomposes as

$$E = E_{\text{sense}} + E_{\text{compute}} + E_{\text{transmit}} + E_{\text{sleep}} \quad (1)$$

with $E_{\text{sense}} = 1.200$ mJ, $E_{\text{compute}} = 0.158$ mJ and $E_{\text{sleep}} = 0.0540$ mJ in the base configuration. Lifetime follows from a usable battery energy of 25,920 J.

2.2 Radio Energy

Transmission energy for a payload of n bytes is modelled as a fixed per-packet cost plus a per-byte cost, with fragmentation where the payload exceeds the maximum transmission unit:

$$E_{\text{transmit}}(n) = \lceil \frac{n}{MTU} \rceil \cdot E_{\text{packet}} + n \cdot E_{\text{byte}} \quad (2)$$

The ceiling term matters for the raw architecture, which fragments on every radio tested, and the fixed term matters most for the processed architecture, where the payload is small enough that the per-byte term nearly vanishes.

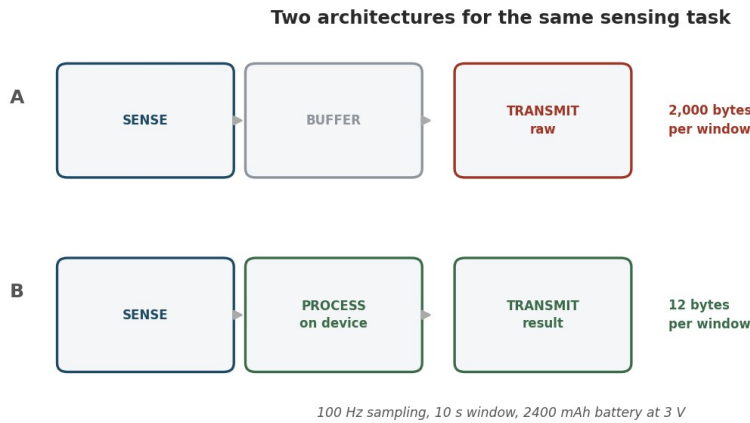


Fig.1: The two architectures and the payload each produces per reporting window.

Parameters for the five technologies are given in Table 1. They are representative catalogue and datasheet values for commonly used parts rather than measurements of a specific design, and a designer should substitute their own; the script accompanying this paper, implemented in Python using NumPy, SciPy and Matplotlib [28-30], takes them as inputs [13,23].

Table 1: Radio and node parameters.

Technology	Energy per byte (J)	Fixed per-packet energy (J)	Maximum payload (bytes)	Packets per raw window
BLE 5 (1 Mbps)	2.6e-6	1.8e-4	244	9
802.15.4 / Zigbee	9.4e-6	3.1e-4	104	20
LoRa SF7	1.2e-4	4.2e-3	222	10
LoRa SF10	8.9e-4	2.6e-2	51	40
NB-IoT	3.4e-5	1.1e-1	512	4

Node: 100 Hz sampling, 10 s window, 2,000 raw bytes or 12 processed bytes, 48 MHz microcontroller, 2400 mAh at 3 V.

3. RESULTS

3.1 Lifetime under the Two Architectures

Table 2 and Figure 2 give node lifetime. Streaming raw samples gave 1.02 years on BLE, 0.313 on 802.15.4, 0.030 on LoRa SF7 and 0.003 on LoRa SF10 — roughly one day on the slowest link.

Processing on device gave 5.06, 4.48, 1.17 and 0.22 years respectively, improvements of 5.0, 14.3, 39.1 and 74 times.

The gain rises with the cost of the radio, which is the expected ordering. What is worth noting is the absolute position: on the two long-range links, raw streaming is not a viable architecture at this sampling rate under any battery a small node can carry, so the choice is not between two designs but between one design and none.

NB-IoT breaks the pattern, gaining only 4.6 times, and Section 3.5 explains why.

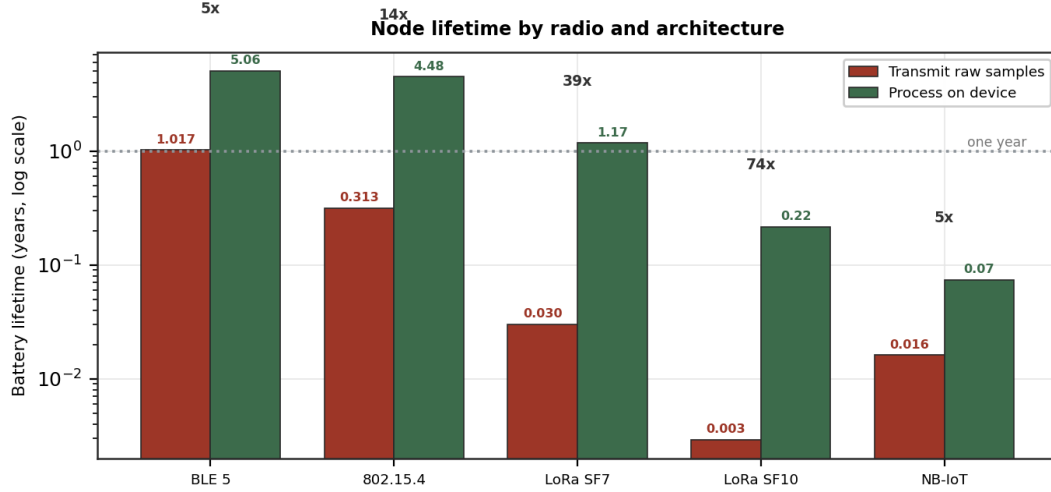


Fig.2: Node lifetime by radio technology and architecture.

Table 2: Node lifetime and energy composition.

Technology	Lifetime, raw (years)	Lifetime, processed (years)	Gain	Radio share, raw	Radio share, processed
BLE 5 (1 Mbps)	1.017	5.06	5.0×	84.5%	13.0%
802.15.4 / Zigbee	0.313	4.48	14.3×	95.2%	23.0%
LoRa SF7	0.030	1.17	39.1×	99.5%	79.8%
LoRa SF10	0.003	0.22	74.1×	100.0%	96.3%
NB-IoT	0.016	0.07	4.6×	99.8%	98.7%

3.2 Where the Energy Goes

Figure 3 shows the energy composition. Under raw streaming the radio accounted for 84.5 per cent of node energy on BLE and above 95 per cent on every other technology.

Under on-device processing the picture divides. On the short-range links the radio share fell to 13.0 per cent for BLE and 23.0 per cent for 802.15.4, so the node became sensing-dominated and further payload reduction would achieve little. On the long-range links it remained at 79.8 and 96.3 per cent, so the radio still dominates even with a 12-byte payload.

This split determines what a designer should optimise next. A sensing-dominated node is improved by a lower sampling rate, a more efficient front end or duty-cycling the sensor. A radio-dominated node is improved by reporting less often or by a more efficient link, and not by further compressing an already tiny payload, a limit also observed in studies of energy-aware data compression on constrained devices [17,18].

3.3 The Compute Budget

The quantity a designer needs is not whether to process on device but how much processing the saving affords. Setting the compute energy equal to the transmission energy saved and solving for cycles gives

$$cycles_{max} = \frac{E_{transmit}(raw) - E_{transmit}(result)}{n \cdot t_{cycle} \cdot I_{active} \cdot V} \quad (3)$$

$\text{saving} = E_{\text{tx}}(\text{RAW_BYTES}) - E_{\text{tx}}(\text{RESULT_BYTES})$
 $\text{per_cycle} = (1 / F_{\text{CPU}}) * I_{\text{ACTIVE}} * V_{\text{NOM}}$
 $\text{budget} = \text{saving} / \text{per_cycle} \quad \# \text{ cycles per sample}$

Table 3 and Figure 4 evaluate it. The budget was 17,623 cycles per sample on BLE, 65,539 on 802.15.4, 710,453 on LoRa SF7, 7,422,187 on LoRa SF10 and 1,060,245 on NB-IoT.

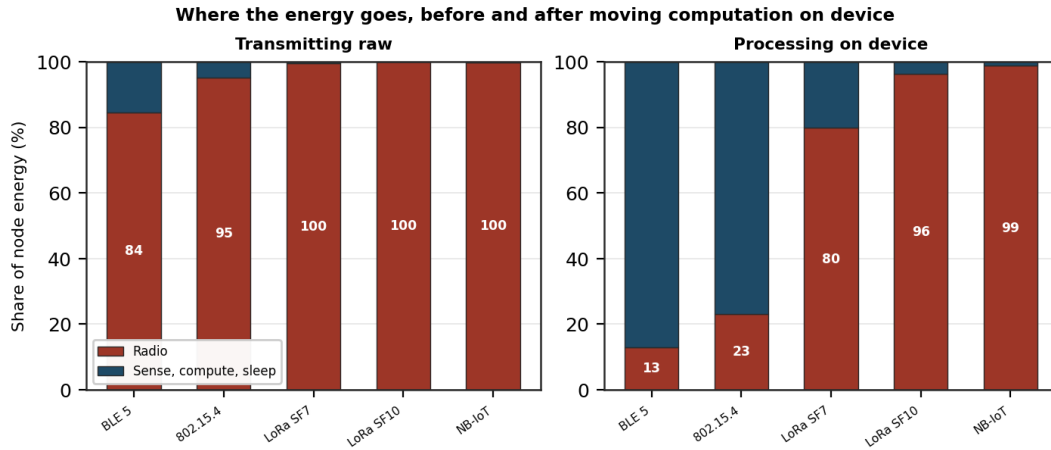


Fig.3: Share of node energy consumed by the radio under each architecture.

Expressed as MCU time, the BLE budget is 367 ms of active processing per 10-second window, which is a real constraint that a large neural network could exceed. The LoRa SF10 budget is 155 seconds of processing per 10-second window, which exceeds real time and therefore cannot be exhausted: on that link, any processing that reduces the payload pays.

This is the practical deliverable. Equation 3 needs only datasheet values and can be evaluated before an algorithm is selected, converting an open question about architecture into a bounded one about how large a model will fit.

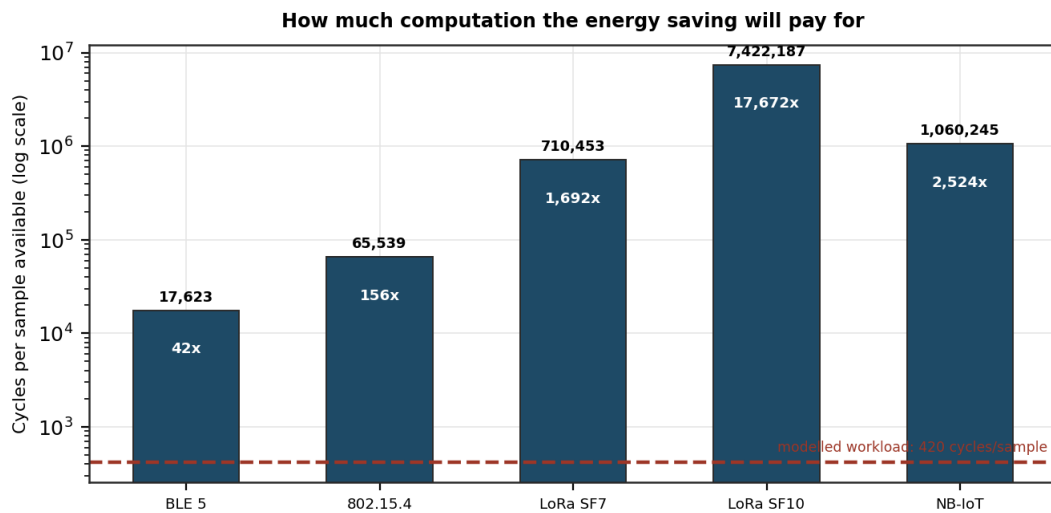


Fig.4: Cycles per sample the energy saving will pay for, by radio technology.

Table 3: Compute budget by radio technology.

Technology	Cycles per sample available	MCU time per window (ms)	Multiple of modelled workload	Exhaustible in real time?
BLE 5 (1 Mbps)	17,623	367.2	42×	Yes
802.15.4 / Zigbee	65,539	1365.4	156×	Yes
LoRa SF7	710,453	14801.1	1,692×	No
LoRa SF10	7,422,187	154628.9	17,672×	No
NB-IoT	1,060,245	22088.4	2,524×	No

3.4 How Heavy Can the Processing Be?

Table 4 and Figure 5 sweep processing cost from 50 to 12,000 cycles per sample. On BLE the gain fell from 5.4 to 1.4 times across that range, approaching the point where processing stops paying.

On LoRa SF10 the gain fell only from 74.3 to 66.5 times, confirming that the budget there is effectively unbounded for any workload an MCU of this class can run.

The engineering reading is that the architecture decision is insensitive to algorithm choice on expensive links and sensitive on cheap ones. A team targeting BLE should profile its processing; a team targeting LoRa need not, and should spend the effort on the reporting schedule instead.

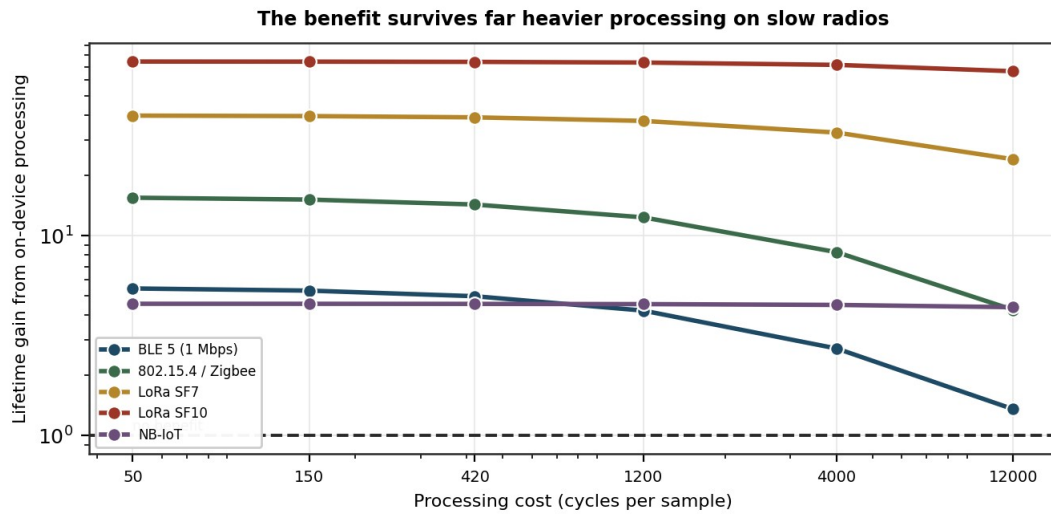


Fig.5: Lifetime gain from on-device processing against processing cost.

Table 4: Lifetime gain against processing cost (cycles per sample).

Technology	50	150	420	1,200	4,000	12,000
BLE 5	5.4×	5.3×	5.0×	4.2×	2.7×	1.4×
802.15.4 / Zigbee	15.5×	15.1×	14.3×	12.3×	8.3×	4.3×
LoRa SF7	39.9×	39.7×	39.1×	37.5×	32.8×	24.1×
LoRa SF10	74.3×	74.3×	74.1×	73.5×	71.5×	66.5×
NB-IoT	4.6×	4.6×	4.6×	4.5×	4.5×	4.4×

3.5 Why NB-IoT Behaves Differently

NB-IoT gained only 4.6 times from on-device processing despite the same 167:1 payload reduction that produced a 74-fold gain on LoRa SF10.

The cause is the fixed per-packet term. At $1.1\text{e-}1$ J per transmission, a single NB-IoT packet costs more energy than transmitting the entire 2,000-byte raw payload over BLE. Reducing the payload from 2,000 to 12 bytes therefore removes almost none of the transmission cost, and 98.7 per cent of node energy remains in the radio after processing.

The design implication is specific and useful. On a link whose energy is dominated by per-packet overhead, compressing the payload is close to futile and the only effective lever is transmitting fewer packets. A team that adopts edge processing on such a link to extend battery life, without also lengthening the reporting interval, will be disappointed by a margin it could have predicted from the datasheet.

More generally, the ratio of per-packet to per-byte energy is the parameter that determines whether payload reduction or schedule reduction is the productive optimisation, and it is rarely reported alongside the headline energy-per-bit figures used to compare technologies [15].

3.6 Transmitting the Result and the Raw Data

Table 5 and Figure 6 model a node that computes a result and transmits it together with the raw samples, a configuration that arises when a debug, audit or model-retraining path is left enabled in production.

Lifetime fell to 0.994 years on BLE against 5.06 for the processed architecture, a loss of 80.4 per cent of the gain. Across the five technologies the loss ranged from 78.1 to 98.7 per cent.

The node in this configuration is marginally worse than one that never processed at all — 0.994 against 1.017 years on BLE — because it pays the compute cost and the full transmission cost. The energy argument for the architecture is entirely forfeited.

We include this case because it is a configuration error rather than a design trade, and because it is easy to introduce and difficult to notice: the device works correctly, the analytics pipeline receives everything it expects, and the only symptom is that batteries fail on the schedule of the unprocessed design.

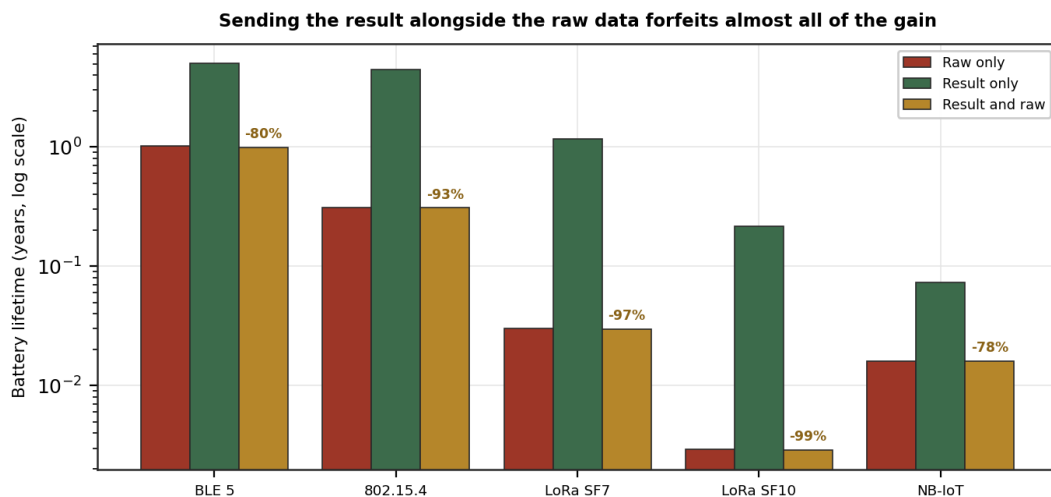


Fig.6: Lifetime when the node transmits the result, the raw data, or both.

Table 5: Energy consequence of transmitting both.

Technology	Raw only (years)	Result only (years)	Both (years)	Gain forfeited
BLE 5 (1 Mbps)	1.017	5.06	0.994	80.4%
802.15.4 / Zigbee	0.313	4.48	0.310	93.1%
LoRa SF7	0.030	1.17	0.030	97.5%
LoRa SF10	0.003	0.22	0.003	98.7%
NB-IoT	0.016	0.07	0.016	78.1%

3.7 Reporting Interval and Detection Latency

Table 6 gives lifetime against reporting interval for the processed architecture, with mean detection latency taken as half the interval.

On BLE, lengthening the interval from 1 to 300 seconds raised lifetime from 2.33 to 5.79 years, a factor of 2.5, and the curve saturates because the node becomes sensing-dominated.

On LoRa SF10 the same change raised lifetime from 0.02 to 3.12 years, a factor of 140, and the curve does not saturate because the radio still dominates at every interval tested.

So the value of accepting longer detection latency depends on which term dominates, and the same trade is worth a modest amount on a fast link and a large amount on a slow one. A latency requirement that is cheap to meet on BLE may be the single most expensive requirement in a LoRa design.

Table 6: Lifetime against reporting interval, processed architecture.

Reporting interval (s)	Mean detection latency (s)	Lifetime, BLE 5 (years)	Lifetime, LoRa SF10 (years)
1	0.5	2.33	0.02
5	2.5	4.48	0.11
10	5.0	5.06	0.22
30	15.0	5.54	0.60
60	30.0	5.68	1.09
300	150.0	5.79	3.12

4. DISCUSSION

4.1 Principal Findings

Four findings follow. On-device processing extended lifetime by 5.0 to 74 times across the radios tested, with raw streaming not viable at all on the long-range links. The compute budget implied by that saving ranged from 17,623 cycles per sample on BLE to 7,422,187 on LoRa SF10, the latter exceeding what the processor can execute in real time. NB-IoT gained only 4.6 times because fixed per-packet overhead dominates its energy. And transmitting the result alongside the raw data forfeited 78 to 99 per cent of the gain.

The organising observation is that the admissible processing cost is a property of the link rather than of the algorithm, and is available from datasheets before any software is written.

4.2 Relation to Existing Work

Energy models for low-power wireless nodes are long established, and the dominance of radio energy over computation is a standard result in wireless sensor network design [1,2,8]. Measurement studies of LPWAN and cellular IoT energy consumption have characterised per-packet and per-byte costs for the technologies modelled here [6,7,9,10]. The case for on-device inference and its energy rationale has been made extensively in the embedded machine learning literature [3-5,11], and more broadly in surveys of edge computing and edge intelligence for IoT [19-22]. Earlier comparative work has measured in-sensor processing against raw data transmission over ZigBee, BLE and Wi-Fi for data-intensive monitoring [25], and approximate computing has been proposed to reduce the processing side of the same trade [26].

What we add is the inversion: rather than asking whether on-device processing saves energy, which is usually yes, we compute how much processing the saving affords and express it in a unit a firmware engineer can check against a profiler. The NB-IoT result and the both-paths configuration error are, so far as we can find, not quantified elsewhere in this form.

4.3 Applying the Model

Table 7 gives a six-step checklist. Two steps deserve comment.

SIZE THE COMPUTE BUDGET FROM THE RADIO, NOT THE MCU
- On BLE the saving paid for 17,623 cycles per sample, 42 times the modelled workload - On LoRa the budget exceeded what the MCU can execute in real time, so processing always pays
WHERE PACKET OVERHEAD DOMINATES, PAYLOAD SIZE BARELY MATTERS
- NB-IoT gained only 4.6x despite a 166:1 payload reduction - 99% of node energy was still radio after processing; report less often instead
DO NOT SHIP THE RAW DATA ALONGSIDE THE RESULT
- Transmitting both forfeited 97% of the edge gain on LoRa SF7 - A debug or audit path left enabled in production removes the reason for the architecture
CHECK WHICH TERM DOMINATES BEFORE TUNING ANYTHING
- Lengthening the interval from 1 s to 300 s gained 2.5x on BLE and 140x on LoRa SF10 - Sensing dominates on fast radios, so reporting less often saturates quickly

Fig.7: Design guidance for the architecture decision.

Table 7: Design checklist.

Step	Question it answers	Inputs needed	Section
Evaluate Equation 3	How many cycles per sample are affordable	Radio datasheet, MCU current, clock	3.3
Compute the per-packet to per-byte energy ratio	Whether payload or schedule is the lever	Radio datasheet	3.5
Identify the dominant energy term after processing	What to optimise next	Equation 1 terms	3.2
Confirm the raw path is disabled in production	Whether the architecture delivers its saving	Firmware configuration	3.6

Step	Question it answers	Inputs needed	Section
Price the latency requirement in battery years	What the reporting interval costs	Equation 1 at each interval	3.7
Re-evaluate after any radio change	A budget that moves by orders of magnitude	Equation 3	3.3

Evaluate the compute budget before selecting an algorithm. Equation 3 requires three datasheet numbers and a division. It converts the question from whether a model is small enough, which cannot be answered in the abstract, into whether it fits in a stated number of cycles per sample, which a profiler answers directly.

Check the per-packet to per-byte ratio before assuming payload reduction will help. On the technologies modelled here that ratio spans four orders of magnitude, and it determines whether the productive optimisation is compressing the message or sending fewer of them. It is computable from the same datasheet and is not usually part of a technology comparison.

5. LIMITATIONS

The model is analytical and uses representative catalogue values rather than bench measurements. Real energy per packet depends on link margin, retransmissions, acknowledgement policy, network attachment state and duty-cycle regulation, all of which vary with deployment and none of which is represented [23]. The ordering across technologies is robust; the specific lifetimes are not.

Radio energy is modelled as transmission only. Receive windows, listening for downlink and clock synchronisation add energy that falls on both architectures but not equally, and on some protocols the receive window is comparable to the transmit cost [12].

Compute energy is taken as active current multiplied by execution time at a fixed clock. Real microcontrollers have voltage and frequency scaling, memory access costs that differ from core cycles, and accelerators whose energy per operation differs by an order of magnitude from the core. A design using a dedicated DSP or neural accelerator would have a substantially larger effective budget than Equation 3 gives [24,26].

We model a single sensing channel and a fixed reporting cadence. Event-driven reporting, in which the node transmits only when the classifier fires, changes the arithmetic substantially and generally favours on-device processing further, since the transmission term scales with event rate rather than with time.

Finally, the analysis is purely energetic. It does not price the engineering effort of implementing and validating on-device processing, the loss of the raw data for later model improvement and knowledge discovery [27], or the regulatory and assurance implications of a device that makes a classification decision locally, each of which bears on a real architecture choice.

6. CONCLUSION

For a node sampling at 100 Hz, processing on device rather than streaming raw samples extended battery lifetime from 1.02 to 5.06 years on BLE and from 0.003 to 0.22 years on LoRa SF10. On the long-range links raw streaming is not a viable architecture at any battery a small node can carry.

The saving buys a quantifiable amount of computation: 17,623 cycles per sample on BLE, equivalent to 367 ms of processing per window, rising to 7,422,187 on LoRa SF10, which exceeds real time and therefore cannot be spent. Equation 3 produces this figure from datasheet values alone.

Two results qualify the general recommendation. Where fixed per-packet energy dominates, as on NB-IoT, a 167:1 payload reduction yielded only a 4.6-fold gain and the effective lever is reporting interval. And a node that transmits the result alongside the raw data forfeits 78 to 99 per cent of the benefit while paying the compute cost, which is a configuration to check for rather than a trade to make.

DECLARATIONS

Ethics approval and consent to participate. Not applicable. This is an analytical modelling study. It did not involve human participants, animals or personal data.

Consent for publication. Not applicable.

Data availability statement. This is an analytical modelling study; no external, experimental or human-subject datasets were used. All results were generated by an analysis script developed by the author in Python using NumPy, SciPy and Matplotlib [28-30]. Every model parameter is reported in full in Table 1, and all figures and all values in Tables 2 to 6 can be reproduced from these parameters using Equations 1 to 3. The analysis script is available from the corresponding author upon reasonable request.

Conflict of interest. The author declares no competing interests, financial or non-financial, in relation to the subject matter of this manuscript. The author has no financial or advisory relationship with any manufacturer of radio transceivers, microcontrollers or IoT platform services.

Funding. This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

Author's contribution. Madhu Sahu is the sole author of this work. She conceived the study, derived the energy model, implemented the analysis, produced the figures, verified all reported values, developed the design checklist, and wrote, revised and approved the final manuscript. She is accountable for all aspects of the work.

Acknowledgements. None declared.

REFERENCES

1. Raghunathan V, Schurgers C, Park S, Srivastava MB. Energy-aware wireless microsensor networks. *IEEE Signal Process Mag.* 2002;19(2):40-50.
2. Anastasi G, Conti M, Di Francesco M, Passarella A. Energy conservation in wireless sensor networks: a survey. *Ad Hoc Netw.* 2009;7(3):537-68.
3. Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge computing: vision and challenges. *IEEE Internet Things J.* 2016;3(5):637-46.
4. Warden P, Situnayake D. *TinyML: machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers.* Sebastopol: O'Reilly; 2019.
5. Banbury CR, Reddi VJ, Lam M, Fu W, Fazel A, Holleman J, et al. Benchmarking TinyML systems: challenges and direction. *arXiv:2003.04821.* 2020.
6. Casals L, Mir B, Vidal R, Gomez C. Modeling the energy performance of LoRaWAN. *Sensors.* 2017;17(10):2364.
7. Lauridsen M, Krigslund R, Rohr M, Madueno G. An empirical NB-IoT power consumption model for battery lifetime estimation. *Proc IEEE Veh Technol Conf.* 2018;1-5.
8. Pottie GJ, Kaiser WJ. Wireless integrated network sensors. *Commun ACM.* 2000;43(5):51-58.
9. Bouguera T, Diouris JF, Chaillout JJ, Jaouadi R, Andrieux G. Energy consumption model for sensor nodes based on LoRa and LoRaWAN. *Sensors.* 2018;18(7):2104.

10. Sinha RS, Wei Y, Hwang SH. A survey on LPWA technology: LoRa and NB-IoT. *ICT Express*. 2017;3(1):14-21.
11. David R, Duke J, Jain A, Janapa Reddi V, Jeffries N, Li J, et al. TensorFlow Lite Micro: embedded machine learning for TinyML systems. *Proc Mach Learn Syst*. 2021;3:800-11.
12. Dutta P, Dawson-Haggerty S, Chen Y, Liang CJM, Terzis A. Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless. *Proc ACM SenSys*. 2010;1-14.
13. Gomez C, Oller J, Paradells J. Overview and evaluation of Bluetooth Low Energy: an emerging low-power wireless technology. *Sensors*. 2012;12(9):11734-53.
14. Adame T, Bel A, Bellalta B, Barcelo J, Oliver M. IEEE 802.11ah: the WiFi approach for M2M communications. *IEEE Wirel Commun*. 2014;21(6):144-52.
15. Vejlgaard B, Lauridsen M, Nguyen H, Kovacs IZ, Mogensen P, Sorensen M. Coverage and capacity analysis of Sigfox, LoRa, GPRS, and NB-IoT. *Proc IEEE Veh Technol Conf*. 2017;1-5.
16. Martinez B, Montón M, Vilajosana I, Prades JD. The power of models: modeling power consumption for IoT devices. *IEEE Sens J*. 2015;15(10):5777-89.
17. Sadler CM, Martonosi M. Data compression algorithms for energy-constrained devices in delay tolerant networks. *Proc ACM SenSys*. 2006;265-78.
18. Barr KC, Asanovic K. Energy-aware lossless data compression. *ACM Trans Comput Syst*. 2006;24(3):250-91.
19. Kumar N, Zeadally S, Rodrigues JJPC. Vehicular delay-tolerant networks for smart grid data management using mobile edge computing. *IEEE Commun Mag*. 2016;54(10):60-66.
20. Premsankar G, Di Francesco M, Taleb T. Edge computing for the Internet of Things: a case study. *IEEE Internet Things J*. 2018;5(2):1275-84.
21. Chen J, Ran X. Deep learning with edge computing: a review. *Proc IEEE*. 2019;107(8):1655-74.
22. Zhou Z, Chen X, Li E, Zeng L, Luo K, Zhang J. Edge intelligence: paving the last mile of artificial intelligence with edge computing. *Proc IEEE*. 2019;107(8):1738-62.
23. Sanchez-Iborra R, Sanchez-Gomez J, Ballesta-Viñas J, Cano MD, Skarmeta AF. Performance evaluation of LoRa considering scenario conditions. *Sensors*. 2018;18(3):772.
24. Maier A, Sharp A, Vagapov Y. Comparative analysis and practical implementation of the ESP32 microcontroller module for the Internet of Things. *Proc Internet Technol Appl*. 2017;143-48.
25. Shahzad K, Oelmann B. A comparative study of in-sensor processing vs. raw data transmission using ZigBee, BLE and Wi-Fi for data intensive monitoring applications. *Proc Int Symp Wirel Commun Syst*. 2014;519-24.
26. Ghosh A, Raha A, Mukhopadhyay S. Energy-efficient IoT-health monitoring system using approximate computing. *Internet Things*. 2020;9:100166.
27. Mishra N, Lin CC, Chang HT. A cognitive adopted framework for IoT big-data management and knowledge discovery prospective. *Int J Distrib Sens Netw*. 2015;11(10):718390.
28. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. *Nature*. 2020;585(7825):357-62.
29. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat Methods*. 2020;17(3):261-72.
30. Hunter JD. Matplotlib: a 2D graphics environment. *Comput Sci Eng*. 2007;9(3):90-95.