BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

Sustainable and Green Robotics: Energy- Aware Software Architecture for Ambient- Powered Autonomous Robotic System

[1]Nishant Singhal, [2]Prof. Ankita Pokale, [3]Prof. Arvind R. Gautam

[1]Student (T.E. Undergraute), [2]Seminar Mentor, [3]Coordinator
[1]Department of Artificial Intelligence and Data Science
[1]Ajeenkya DY Patil School of Engineering, Pune, India

Abstract

Most mobile and remote field robots, as well as sensor nodes, continue to operate on conventional batteries. Batteries offer flexibility, but they must eventually be replaced, which creates an e-waste problem. One approach discussed in my seminar readings is to remove batteries entirely: the robot harvests tiny amounts of energy from its surroundings, captures vibrations using triboelectric nanogenerators (TENGs), stores the electricity briefly in a capacitor, and operates in bursts. Materials for this process are developing quickly, but the software design lags behind. Embedded applications assume that power is always available and spend whatever charge has been collected in an indefinite loop that polls for charge. Here I combine research on biodegradable materials, TENGs, interrupt-driven firmware, voltage scaling, power-loss checkpointing, energy-aware scheduling, and small TinyML models that predict energy inflow. This work reports no new experiments. My aim is to put these separate ideas into one clear picture and show where they still do not fit together, chiefly the cost of the prediction model itself and the absence of proper outdoor testing. I have not built a prototype.

Keywords: Sustainable Robotics, Energy Harvesting, Triboelectric Nanogenerators, Batteryless Systems, Intermittent Computing, TinyML, Low-Power Firmware.

I. Introduction

Replacing a battery sounds like a small job; in a laboratory, it is. On a farm, in a forest, or in a disaster-hit area where hundreds of nodes may be spread over a large space, it is not small at all. Some nodes will never be recovered, and if each one carries a lithium or alkaline cell, that cell stays in the soil long after the mission ends. That was the starting point of my seminar topic, and honestly it is what made the topic interesting to me.

Green robotics tries to tackle this from two sides. On the hardware side, researchers are making parts that do less harm when left behind; Wang et al. [1] describe TENGs made with polymers obtained from biological sources, and other groups are working on circuits and sensors that break down in moisture after their work is done. On the energy side, the idea is to collect what is already nearby: vibration, movement and sunlight.

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

The complication starts soon after. A steady supply is presumed, and most microcontroller programs are written for exactly that case. A TENG on a moving platform may pulse strongly for a moment and produce almost nothing the next moment, simply because the vibration changed; harvested energy does not behave like a battery. A typical habit is busy-wait polling, where the processor checks a sensor register again and again in a tight loop. With a battery this is wasteful but tolerable. With a small capacitor as the only buffer, the same loop can empty the buffer before any useful work is done, and the board resets.

Batteryless research has therefore moved in another direction. A second line of work gives the node a little foresight: TinyML models small enough to run on a microcontroller have been employed to estimate how much energy may arrive next [3], and the node sleeps deeply, wakes only on an interrupt, does a short piece of work, saves its state, and is ready to shut down and restart cleanly [2]. Another way of fighting power loss is for the software to expect it; Sabović et al. [4] change the model depending on the charge available. I also read work on schedulers that refuse to start a task when charge is too low [5], on forwarding packets through nodes that keep losing power [6], and on a solar aerial robot that flies with no battery at all [7].

Let me be clear about what this paper is: a review, not a laboratory study. I have tried to read these scattered papers together and ask a simple question. I did not build a robot, train a model on a new dataset, or run a benchmark. Why are batteryless robots still hard to run reliably outdoors, if the materials are getting greener? My answer after this reading is that the runtime software has to be redesigned for energy that comes and goes; greener hardware alone will not be enough.

II. Questions That Guided This Review

I started with four questions and kept returning to them while reading. First, what decides how much power a node actually gets from a biodegradable harvester or another small ambient source? Second, how do interrupt-based firmware, dynamic voltage and frequency scaling (DVFS) and intermittent runtimes decrease waste, and do they survive a sudden voltage drop? Third, when does a TinyML predictor actually help? It is valuable only if it spends less energy than it saves, so this question took me the longest to think through. Fourth, where do the hardware and software papers fail to meet each other?

The rest of the paper follows these questions. Section III sets up the energy problem. Section IV explains how I chose the papers. I then group the work into four layers, compare them, join them into one closed-loop design, and end with gaps and what should be tested next.

III. Why Harvested Power Changes the Design

A. From a Battery to a Capacitor Buffer

A battery gives a fairly flat voltage for a long time and then turns down slowly. A batteryless node lives differently: it receives small, irregular currents, stores them in a ceramic capacitor or supercapacitor, and runs only after enough charge has built up. It then works concisely and shuts down again.

TENGs show this pattern well. They turn contact and vibration into charge, so output follows movement. Energy can therefore no longer be treated as a background detail for the programmer; it has to be scheduled, much like CPU time and memory. Weak movement gives almost nothing; strong movement gives more charge.

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

B. A Simple Way to Write the Energy Limit

A small calculation helped me more than any paragraph did. Say a capacitor of capacitance C wakes the node at V_{wake} and the node must stop at V_{floor}. The usable energy in one active window is

$$E_{cap} = 0.5 \cdot C \cdot (V_{wake}^2 - V_{floor}^2) \quad (1)$$

Inside that window the node may run a predictor, do its sensing or movement work, and save its state. All three must finish before the voltage collapses:

$$E_{cap} \geq E_{est} + E_{work} + E_{save} \quad (2)$$

Here E_{est} is the cost of the predictor, E_{work} is the cost of the task, and E_{save} is the cost of writing state to non-volatile memory. Equations (1) and (2) are not taken from any one paper; they are a straightforward teaching model. What we learn is uncomfortable for TinyML work: E_{est} is paid in every cycle, right or wrong, so the predictor must be very cheap when checked against E_{cap}; otherwise the node is spending its scarce energy thinking about energy. Fig. 1 shows the resulting charge, wake, work and sleep pattern.

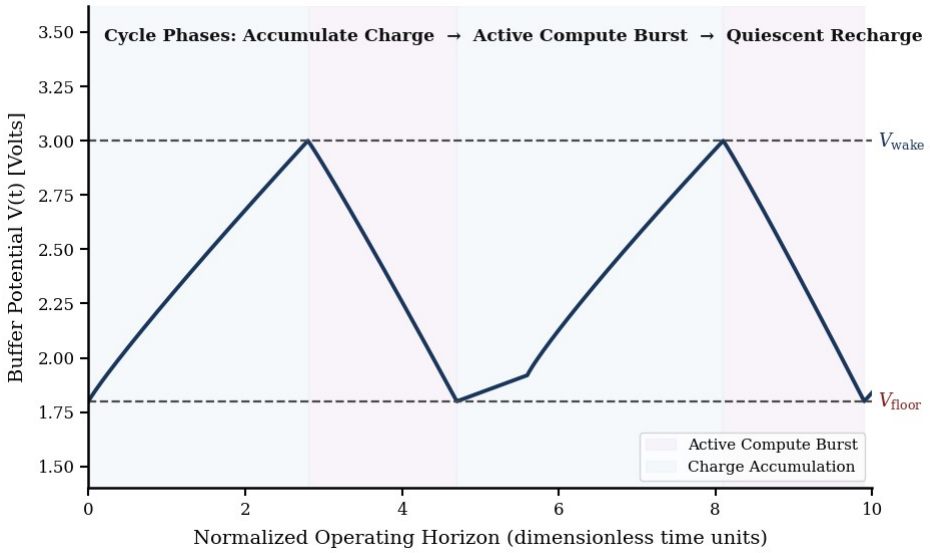


Fig. 1. Capacitor voltage in a batteryless node. Charge builds to V_{wake}, the node works for a short burst, and stops before V_{floor}. Schematic only, not measured data.

C. Why Polling Loops Fail So Quickly

In polling, the processor reads a sensor register again and again to see if anything has changed. Active and deep-sleep currents differ by orders of magnitude, so polling empties a microwatt-scale buffer fast, often before any actuation happens. Most of the time nothing has changed, but active current is still burned on every check.

Interrupt-driven firmware reverses this. Standard real-time operating systems suppose stable power and have no natural way to handle a task that loses power in the middle and returns seconds later. With interrupts, a pin change, timer or comparator wakes the processor only when there is something to do, and the core stays off the rest of the time.

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

IV. How I Selected and Read the Papers

The review is based on material collected for my technical seminar: seven primary seminar studies read together with seven peer-reviewed foundational papers [1]–[7]. Together they cover TENG materials, biodegradable sensors, low-power firmware, DVFS, intermittent execution, scheduling, routing on unreliable power, TinyML on capacitor-backed nodes, and one batteryless flying robot.

One early decision is worth explaining. Each finding has been kept in context, and the papers are compared by the obstacle they resolve and what they leave open. The studies use distinct boards, capacitors, energy traces and workloads, so an accuracy or uptime figure makes sense only in its own setup. Fig. 2 shows the five stages I followed. I have not merged numbers from diverse papers into a combined score; it would look neat, but it would not be honest.

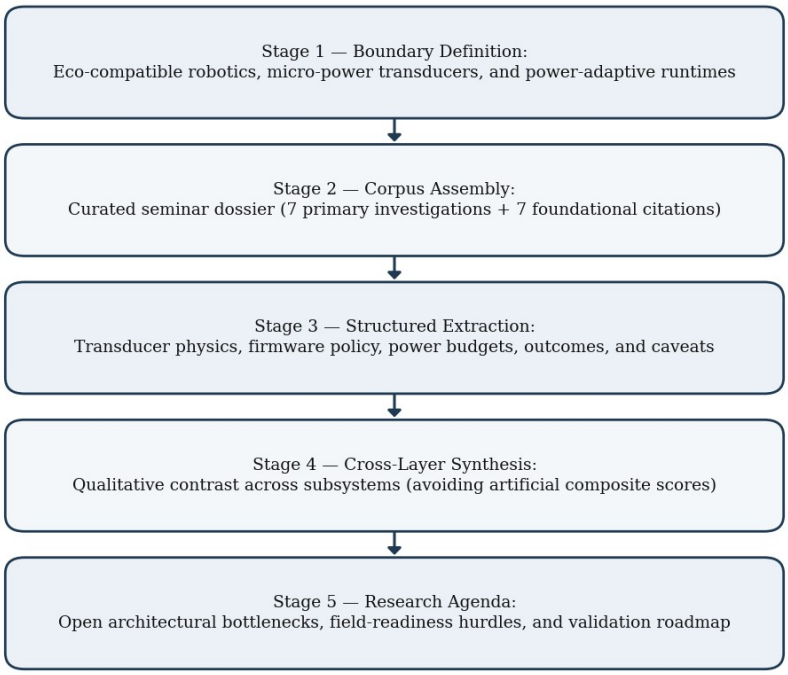


Fig. 2. Five steps followed in this review, from fixing the scope to listing open problems.

V. A Four-Layer View of the Literature

I found it easier to think of a sustainable robot as four stacked layers (Fig. 3). At the bottom is the physical layer: TENGs, bio-based films, moisture-dissolving sensors and temporary circuits. Above it is microcontroller control, such as deep sleep, wake-on-interrupt and DVFS. The third layer handles failure and choice: it saves state in non-volatile memory and decides which task runs now, runs in a cheaper form, or waits. On top is prediction, where a small TinyML model estimates near-term energy and, in some designs, changes its own size with the charge level.

These layers are not alternatives, a point I missed at first. They are parts of one machine. The predictor looks a little ahead, and charge is brought in by the harvester. Firmware also determines whether the

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

processor is awake at all, and the scheduler assigns jobs that can be completed without risk. Therefore, when one level is weak, it is likely that the ones above it are also affected.

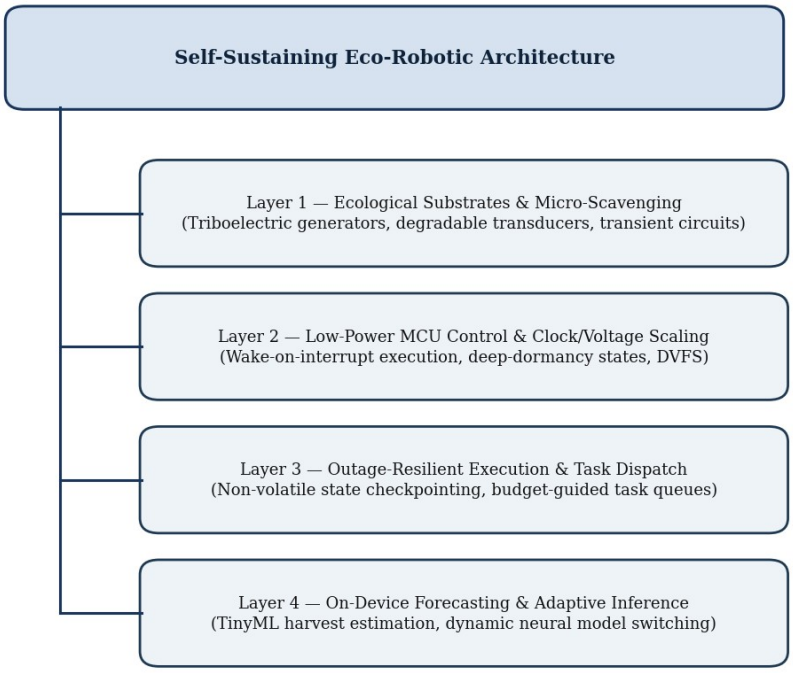


Fig. 3. Four layers used here to organise the hardware and software work.

VI. The Physical Layer: Harvesters and Degradable Materials

A. Triboelectric Nanogenerators

A TENG generates charge from contact electrification and movement. For a mobile robot this is attractive, because the platform is already vibrating. Wang et al. [1] tackled the material side, employing bio-derived polymers so that the harvester itself does not become long-lasting waste. In the seminar set, Sun et al. (2024) showed kinetic harvesting with TENGs but were clear about how uneven the output is.

A learned predictor has little to learn if the input has no learnable pattern. This link between a material property and a software choice is uncommonly stated directly, but it runs through the whole topic. For software, uneven output is not a minor detail: it decides how often the node can wake and whether prediction is worth trying at all.

B. Sensors and Circuits That Break Down After Use

Short-lived electronics and biodegradable sensors decompose safely in a natural environment, so nobody has to recover every node. Li et al. (2024) reported safe decomposition, but also found that sensor response changes with humidity.

That second point matters more than it first appears. A sensor that is slowly dissolving does not keep one calibration for its whole life; its readings drift. Most edge-intelligence work quietly postulates that the sensor never changes. Here that assumption breaks. I think this is one of the most overlooked roadblocks

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

in the set, partly because the people making the material and the people writing the perception code are typically different groups.

VII. The Software Layers: Staying Alive on Unreliable Power

A. Sleeping Properly and Waking on Interrupts

The simplest idea is also the most effective. Srivastava and Chen (2023) reported large savings from removing idle polling this way, and I find this easy to believe because the reason is basic physics. Keep the microcontroller in deep sleep, at sub-microampere current in the best cases, and wake it only when an interrupt says something happened.

Their shortcoming is distinct. The code was tied to the registers of one microcontroller family, so moving to another vendor means rewriting much of it. The open issue is therefore not whether interrupt-based sleep works; it does. The snag is the lack of a clean, portable layer for sleep modes and wake sources.

B. Slowing the Clock Down: DVFS

DVFS lowers clock speed and voltage when work is not urgent; Acar et al. (2023) showed clear savings in active cycles. As dynamic energy falls roughly with the square of voltage, even a small reduction helps.

The trouble is that their policy presumed a battery holding the rail up. On a harvester the input can simply disappear, and a slower clock does not protect against that. DVFS is therefore an inner control knob: it makes running work cheaper, but it still needs an outer process that understands blackouts.

C. Expecting Power Loss: Intermittent Execution

In intermittent computation, power outage is considered a norm rather than an anomaly; the program is therefore divided into small chunks. Some critical state information is backed up to non-volatile memory such as FRAM prior to voltage collapse; hence, when power returns, execution resumes from the last checkpoint rather than from the start. This point is explained very clearly by Hester and Sorber [2], and this paper changed my perception of the rest of the papers significantly. In addition, saving the state consumes energy, E_{save} in (2).

The problem is one of balance. Saving state too frequently consumes the budget for it, while saving too rarely means that work is lost in every power outage, which can in turn lead to inconsistency between volatile and non-volatile variables. Depending on the workload, the checkpoint location may change.

D. Deciding What to Run: Energy-Aware Scheduling

Something must choose what to run now, what to run in reduced form, and what should wait if a single node cannot do it all. Li et al. [5] proposed scheduling algorithms that try to avoid killing a job halfway due to lack of energy. According to Kumar and Rossi (2026), ordering jobs by available energy extended the availability of energy-scavenging nodes in practice, although not in the context of biodegradable devices.

Liu et al. [6] handled the problem in its network incarnation: forwarding packets while nodes get switched off at different times. This is not only about robot control, of course; any deployment involving a number of nodes will face the same problem.

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

E. Predicting the Next Few Seconds Using TinyML

This section drew my initial attention from my AI and data science background, and it is where I began to be more vigilant as I went through the text. The point is simple. A tiny regression algorithm on a microcontroller evaluates recent voltage and harvester readings and approximates the upcoming intake of energy; the scheduler can then start a larger task during fertile periods and simply wait during lean periods. Zhang et al. (2025) report precision exceeding 90% in their setting, and I would like to underline "in their setting", because the figures do not apply to all harvesters; the authors themselves admit that the inference uses the same small buffer.

Singh and Patel (2025) estimate the trade-off between model quality and model complexity, even without the sensor-ageing factor: use the large model when charge is abundant, and a small one or a simple heuristic rule when it is lacking. Sabović et al. [4] introduce a useful technique, namely maintaining several models and switching between them. Moreover, Kim et al. [3] demonstrated the running of very small models on capacitor-powered nodes consuming less than one milliwatt.

From these papers I conclude that the predictor should be flexible, meaning that it should get smaller or switch off fully during lean periods. A full-size model run in every cycle can become a net loss, and (2) is the test it has to pass every time.

F. Complete Robots Are Still Rare

Most studies test one layer alone. A failure during motion can also be a safety issue, not only a computing issue. Full robots matter because motors draw high peak currents and bodies have inertia, so until more whole-system tests exist, component findings should be read with care. Chen et al. [7] are an exception, with a batteryless solar aerial vehicle under energy-aware control.

VIII. Checking the Layers Side by Side

Table 1 lists what each subsystem does and where it struggles most; in plain words, Table 2 compares the software strategies. I have not scored the papers in these tables, because their test setups are too distinct for scoring to mean anything.

Table 1: What Each Subsystem Does and Where It Struggles

Subsystem	What it does	Primary obstacle
TENG harvester	Turns vibration into charge for the capacitor	Output arrives in irregular bursts
Biodegradable sensor	Measures the environment, then breaks down after use	Readings drift with humidity and decay
Wake-on-interrupt firmware	Keeps the MCU asleep until a real event emerges	Ordinarily written for one vendor's registers
DVFS controller	Lowers clock and voltage during light work	Assumes the supply rail stays up

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

Subsystem	What it does	Primary obstacle
Intermittent runtime	Saves state so work survives blackouts	Saving costs energy; consistency is typically tricky
Energy-aware scheduler	Runs, streamlines or postpones tasks by charge level	Hardly ever tested with degradable hardware
TinyML predictor	Estimates energy for the next short interval	Its own inference is paid from the same buffer

Table 2: Software Approaches in Simple Terms

Approach	Main benefit	What it still needs
Wake-on-interrupt	Removes polling waste through deep sleep	Needs portable sleep and wake-up abstraction
DVFS	Cuts active energy when full speed is not needed	Needs blackout protection above it
Intermittent execution	Repeated power failure is survived by progress	Checkpoint frequency is a hard balance
Energy-aware scheduling	Matches work size and timing to charge in hand	Depends on a forecast that may be wrong
TinyML forecasting	Gives local foresight with no cloud link	Model must shrink when energy is often scarce

IX. Putting the Pieces Together

Fig. 4 draws the framework as one loop; this is a reference design put together from the papers, not hardware that I built.

The capacitor is fed by the harvester. The node measures capacitor voltage and inflow rate and passes them to the predictor. The processor wakes only through interrupts. The scheduler then makes a simple choice: run the task fully, run a cheaper version, or wait. Sensing or actuation happens inside a safe margin; before voltage falls too low, the runtime saves state. Actual energy spent is fed back to correct the next prediction.

Every step remains controlled by (2). If prediction, work and especially saving do not fit in the stored energy, the right decision is to wait for more charge. Written down, that sounds obvious, yet many papers raise their own layer without verifying the combined budget, which helps explain why integration is weak.

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

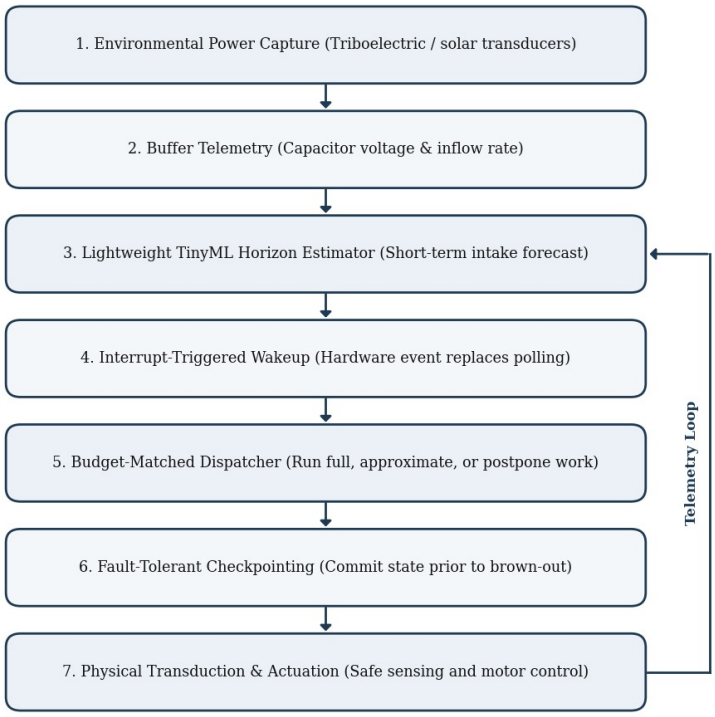


Fig. 4. Closed-loop reference design joining harvesting, prediction, interrupt wake-up, scheduling, checkpointing and actuation. Drawn from the surveyed papers; not a built prototype.

X. Summary of the Papers Read

Table 3 covers the seven seminar studies and Table 4 the seven foundational references [1]–[7]; outcomes are reported as their authors stated them, without conversion to a common scale.

Table 3: Seven Primary Seminar Studies

Study	Focus	Primary outcome (as reported)	Shortcoming gathered
Kumar and Rossi (2026)	Charge-based task ordering in harvester-powered networks	Nodes stayed available longer	No biodegradable hardware in the study
Zhang et al. (2025)	On-device TinyML regression for short-term intake prediction	Precision above 90 percent reported in their setup	Inference drains the buffer each cycle
Singh and Patel (2025)	Edge model quality versus harvested-power limits	Quantified the quality-versus-cost tension	Sensor dissolution not modelled

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

Study	Focus	Primary outcome (as reported)	Shortcoming gathered
Sun et al. (2024)	Vibration conversion with TENGs	Kinetic micro-power capture demonstrated	Output current is highly unpredictable
Li et al. (2024)	Dissolving circuits and biodegradable sensors	Safe decomposition verified	Response differs with humidity
Acar et al. (2023)	DVFS on microcontrollers	Active cycles and energy lowered	Steady battery supply assumed
Srivastava and Chen (2023)	Interrupt-driven dormancy management	Idle polling waste cut sharply	Locked to one MCU family

Table 4: Seven Foundational References

Reference	Insight to this review	Scope limit
[1] Bio-based polymers for TENGs	Harvester material itself can be eco-friendly	Materials only; no runtime software
[2] Intermittent execution for batteryless devices	Move from stable-power assumptions to failure-tolerant execution	Programming models, not robots specifically
[3] TinyML on capacitor-backed nodes	Neural inference below a milliwatt with no battery	Tested on static motes, not actuators
[4] Switching neural models by charge level	Model size follows available charge	Switching cost and per-platform tuning remain
[5] Scheduling that avoids mid-task blackouts	Formal scheduling under an energy bound	Algorithmic; no degradable hardware
[6] Packet forwarding on intermittent nodes	Frequent shutdowns are survived by communication	Routing problem, not local robot control
[7] Batteryless solar aerial robot	Energy-aware control on a real mobile platform	Particular to solar input and daylight

XI. My Critical Reading of the Trade-offs

One pattern stood out more than any single finding. Materials groups are boosting harvesters and dissolving sensors and leave software to someone else. Intermittent-computing and scheduling papers face unreliable power directly, which I found refreshing. TinyML papers add foresight, always at a price. Firmware groups

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

cut sleep current or tune DVFS. The field is split, and some papers still postulate a battery or one precise chip.

Two tensions kept returning. The first is the cost of being clever: a bigger predictor gives a better estimate, yet burns more charge on itself. Beyond a certain point it simply breaks (2), and an elementary voltage threshold would have been more appropriate. The second is that the evidence is good for components and weak for complete robots. The schedulers exclude dissolving harvesters, studies in edge intelligence do not account for sensor drift, and biodegradable hardware and energy-efficient software are rarely tested together. Model switching [4] looked interesting to me because it allows for correct sizing of the model.

Once I attempted making a table of persons with all the figures about accuracy and uptime. However, I abandoned the idea, as it would only be misleading. There are differences in energy traces, sizes of capacitors, boards and baselines. What is required is boring and fundamental: common replayed energy traces, common tasks for robots, the same board, and common measures such as energy per completed task, fraction of activity and blackout recovery time. Otherwise, each paper wins on its own bench.

XII. Issues That Will Appear in Real Deployment

A. Unreliable Input

The scheduler needs some buffer and also an alternative cheaper mode, rather than a brittle one that fails when one forecast goes awry. Harvesting usually produces a bursty stream, so all forecasts have variance.

B. Inference Has to Earn Its Keep

The model draws on the same buffer as the motors and sensors. Heavy quantisation and pruning are usually basic requirements. Further, where the harvest is mostly poor, the model should degrade gracefully.

C. Firmware Portability

Portability can add more here than an increase in forecasting accuracy. Sleep routines, pin-outs and checkpointing procedures are coded against the vendor register maps.

D. Degrading Sensors

The software needs to distinguish true environmental changes from degradation caused by the sensor itself, and discard the unreliable channel. A biodegradable sensor will dissolve and become less sensitive.

E. Safe Failures of Robots

An inactive mote simply turns silent. Actuators require default safe states, apart from checkpointed states. A robot may otherwise be left with its gripper closed or a misadjusted rotor.

F. Benchmarking the Code

A new stack needs to outperform a generic RTOS on the same replayed trace, judged on task completion, blackout resistance and adaptation rate. In sustainable robotics, a suitable benchmark does not yet exist.

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

XIII. Gaps I Could Spot

Four apparent gaps exist. First, each component is found often in the literature, but the integrated system is shown seldom: no study in this set field-tests one robot combining a degradable harvester, interrupt firmware, checkpointing, energy-aware scheduling and an elastic predictor. Second, runtimes built specifically for unstable microwatt budgets are still rare; many predictors spend the same fixed energy on every run, even when intake has fallen near zero, so a saving trait becomes a loss. Third, mainstream RTOS kernels suppose continuous power and offer nothing native for checkpointing or approximating a task when current vanishes. Fourth, hardware and software are advancing apart.

XIV. Where Such Robots Make Sense

Habitat monitoring, precision farming and disaster-area search all spread devices where spent batteries cannot realistically be collected. Factories and off-grid sites are a second case, where harvesting nodes decrease wiring and remove battery changes from maintenance.

For builders, an applied order is proposed by the papers: remove polling, sleep deeply, wake on interrupts and watch the capacitor voltage with a straightforward comparator, and then add checkpointing and charge-aware queues. Add a TinyML predictor last, only after real traces prove it saves more than it spends. Future experimental papers should also publish raw traces, capacitor values, sleep currents, checkpoint costs and the exact baseline kernel setup, so findings can be verified by others.

XV. Directions for Future Work

The next step should be experimental: implement Fig. 4 on a TENG-fed board and compare it directly with a FreeRTOS baseline on task completion, blackout recovery and energy per task. Longer-term needs include board-independent intermittent middleware, predictors that adjust depth to capacitor voltage, joint monitoring of harvesting and sensor decay, fail-safe actuation, and outdoor trials over months that also record how materials decompose in practice.

XVI. Shortcomings of This Review

This is a narrative review from a seminar dossier, not a PRISMA systematic review or meta-analysis, so figures such as the above-90-percent accuracy are quoted as their authors stated them. For some seminar entries I worked from structured abstracts without re-checking raw laboratory logs. Seeing as the studies cover diverse layers and conditions, no new measurements and no combined ranking are added.

XVII. Conclusion

Green robotics is often told as a materials story: remove the battery, use a better polymer, harvest some vibration. My reading says that is only half the story. A robot living on tiny, unreliable bursts of energy also needs software designed for that life: sleeping until interrupted, slowing down when speed is not needed, saving state before the unavoidable blackout, choosing work that fits the charge in hand, and forecasting just a little ahead.

None of these technologies is out of reach now. What is missing is a combined system proven outdoors over a long period: a portable intermittent runtime that can judge whether its own prediction is worth the energy, a perception layer that is aware of its own sensor dissolving, and safe actuation when a crash comes. I think that integration, not any single new material or model, is the real holdup now.

BODHIVRUKSHA JOURNAL OF DIVERSE DISCIPLINE
E-ISSN : 3139-1486
An Open Access, Peer-Reviewed, Multidisciplinary Scholarly Journal
Bi-monthly | Multilingual (English, Hindi, Marathi) | Academic Research
@ Volume 2, Issue 5 (September-October 2026) (Special Issue 1) ISSN: 3139-1486 (Online)

Acknowledgment

I thank Prof. Ankita Pokale, Department of Artificial Intelligence and Data Science, Ajeenkya DY Patil School of Engineering, Pune, for guidance during this technical seminar and for comments on an earlier draft.

References

[1] J. Wang *et al.*, "Bio-derived polymeric substrates for eco-compatible triboelectric nanogenerator power scavenging," *Adv. Funct. Mater.*, 2023.

[2] J. Hester and J. Sorber, "Outage-resilient intermittent execution models for batteryless embedded computing," *Commun. ACM*, 2022.

[3] Kim *et al.*, "Power-conscious TinyML rollout on capacitor-backed batteryless nodes," *Internet of Things (Elsevier)*, vol. 22, Art. no. 100736, 2023.

[4] A. Sabović *et al.*, "Dynamic neural network switching under harvested power constraints," *IEEE Internet Things J.*, 2025.

[5] D. Li *et al.*, "Algorithmic statement of charge-aware task dispatch for batteryless embedded nodes," *IEEE Trans. Mobile Comput.*, 2024.

[6] Z. Liu *et al.*, "Continuous packet forwarding across intermittently powered batteryless sensor networks," *IEEE/ACM Trans. Netw.*, 2024.

[7] Y. Chen *et al.*, "Design and validation of a solar-scavenging batteryless autonomous aerial vehicle," *IEEE Robot. Autom. Lett.*, 2022.