

TINYML'S HIDDEN FOOTPRINT: LIFECYCLE ENERGY USE, EMBODIED EMISSIONS, AND DISPOSAL PATHWAYS OF ULTRA-LOW-POWER ON-DEVICE AI — A COMPREHENSIVE REVIEW

Emmanuel Uzochukwu Mordi¹; David Chinonso Anih^{2*}; Boluwatife Gabriel Jesutoye³, Khalid D. Muhammed⁴ & Kabir Edota⁵

¹Department of Computer Engineering, Faculty of Engineering, University of Benin, Edo, Nigeria; ²Federal University Wukari, Taraba, Nigeria

³Information Security Department, INHAF Ltd, Greater Manchester, United Kingdom; ⁴Federal University of Technology, Minna, Niger State, Nigeria.; ⁵University of Aberdeen, United Kingdom

*Corresponding author: anih.david@fuwukari.edu.ng

Abstract—Tiny Machine Learning (TinyML) is increasingly deployed on constrained devices to provide local intelligence with low latency and reduced communication demand. However, the environmental sustainability of these systems cannot be determined from inference energy alone because significant burdens may arise during model development, hardware production, maintenance, and end of life. This review examines the hidden environmental footprint of TinyML from a cradle to grave perspective, integrating computational, hardware, communication, lifetime, and end of life dimensions. The review prioritizes empirical evidence and examines energy use across model training, optimization, deployment, sensing, preprocessing, inference, standby operation, wireless communication, firmware updates, battery replacement, recycling, and disposal. Attention is given to embodied energy and carbon from semiconductors, neural accelerators, sensors, printed circuit boards, power management components, batteries, enclosures, and transportation. The evidence indicates that TinyML can reduce operational energy by enabling local inference and minimizing transmission of raw data, while model compression, quantization, pruning, efficient memory use, event driven computation, and hardware acceleration can also reduce computational demand. Nevertheless, these gains do not necessarily translate into lower lifecycle environmental impact. Manufacturing burdens may become relatively dominant in highly efficient, intermittently operated, short lived, or low carbon powered systems, whereas operational and communication energy can become more important under intensive sensing, frequent transmission, extended duty cycles, and carbon intensive electricity supplies. Device lifetime is a critical determinant because longer service can distribute embodied impacts across more useful service, while premature replacement, sensor degradation, battery renewal, and repeated hardware turnover increase resource use and waste generation. Fleet scale further amplifies cumulative impacts through device proliferation, maintenance, updates, and disposal. The review identifies inconsistent system boundaries, functional units, measurement practices, and limited TinyML specific lifecycle inventories as major barriers to comparison. It highlights the need for harmonized lifecycle reporting, field based energy measurements, material inventories, regional assessment, and experimentally validated circular hardware strategies. Overall, sustainable TinyML requires coordinated hardware, software, energy, and environmental design in which efficiency, longevity, repairability, modularity, reuse, refurbishment, and material recovery are considered alongside inference performance. Therefore, the review reframes TinyML sustainability from minimizing inference energy to minimizing total environmental impact across the lifecycle.

Index Terms—Tiny Machine Learning, lifecycle sustainability, embodied carbon, operational energy, embedded artificial intelligence, electronic waste, circular electronics, lifecycle assessment, sustainable edge computing

I. INTRODUCTION

Tiny Machine Learning (TinyML) has emerged from the broader convergence of machine learning, embedded computing, Internet of Things (IoT) systems, and edge intelligence, with its central idea being that useful machine learning inference can be performed directly on highly resource constrained devices rather than being dependent on remote cloud infrastructure. Unlike conventional artificial intelligence systems that commonly rely on powerful processors, large memories, and continuous network connectivity, TinyML is designed for microcontroller class platforms with tightly constrained memory, computational capacity, and energy budgets. This transition has opened opportunities for intelligent sensing in wearables, environmental monitoring, smart agriculture, industrial condition monitoring, smart buildings, transportation, healthcare devices, and remote sensing, particularly where low

latency, privacy, intermittent connectivity, and long battery life are important. The evolution of TinyML has therefore shifted part of the intelligence pipeline closer to the physical environment, allowing sensor nodes to acquire, preprocess, interpret, and respond to data locally rather than continuously transmitting raw observations to distant computing infrastructure [1].

Recent advances demonstrate that the energy envelope of such systems is becoming progressively smaller through model compression, quantization, pruning, distillation, hardware aware optimization, and dynamic runtime management. For example, a 2026 experimental study on an ARM Cortex M7 platform reported an energy requirement of 387 μ J per inference after jointly optimizing model precision, pruning, memory scheduling, and processor voltage and frequency, with a reported 58.1% reduction in inference energy relative to its comparison baseline. Such results illustrate the remarkable progress in operational efficiency, but

they also reveal why inference energy alone cannot be used as a proxy for environmental sustainability. TinyML devices do not consist of a microcontroller operating in isolation; they normally include sensors, memory, power management circuitry, printed circuit boards, communication interfaces, passive components, connectors, batteries or energy harvesting systems, protective materials, and packaging [2].

The distinction between low operational power and low environmental footprint is therefore fundamental to understanding the sustainability of TinyML. A device may consume only a small amount of electricity during each inference while carrying an environmental burden that was accumulated before deployment through raw material extraction, semiconductor processing, component fabrication, assembly, transportation, packaging, and battery production. This is particularly important because a substantial fraction of environmental impacts associated with electronic systems can occur outside the use phase. Semiconductor fabrication, for example, involves energy intensive processing and significant requirements for water, materials, chemicals, and specialized manufacturing infrastructure. Life cycle research on nanoscale semiconductor production has shown that manufacturing integrated circuit dies has environmental implications extending beyond climate change to water use, particulate matter, and mineral and metal resource use. Consequently, improving the efficiency of the computation performed by a microcontroller does not automatically eliminate the impacts embedded in producing the hardware that performs that computation [3].

A meaningful environmental analysis must consequently distinguish between operational energy and lifecycle energy. At the model development stage, energy may be consumed during dataset preparation, experimentation, training, hyperparameter optimization, pruning, quantization, and other compression procedures undertaken on comparatively powerful computing systems. Following development, additional computational resources may be required for conversion, compilation, testing, benchmarking, and deployment to the target embedded platform. During operation, the energy attributable to TinyML extends beyond the arithmetic operations involved in inference to include sensing, signal conditioning, feature extraction, memory access, processor activity, communication, security functions, idle and standby states, and battery charging or replacement. The relationship between on device processing and communication is especially important because moving intelligence onto an edge device can reduce the need to transmit large quantities of data. Experimental work using a Cortex M4 based platform for human activity recognition found that an edge AI arrangement reduced energy consumption by up to 21% compared with a remote server based approach, demonstrating that communication avoidance can generate meaningful system level energy savings. However, this benefit should not be interpreted as evidence that every edge deployment is automatically environmentally superior, because the result depends on workload, network characteristics, device lifetime, power source, and hardware composition [4].

The hidden footprint becomes even more apparent when TinyML hardware is examined as a collection of material and manufacturing processes rather than simply as an energy efficient processor. The microcontroller itself depends on semiconductor fabrication, packaging and testing, while sensors may require additional semiconductor, ceramic, polymeric, metallic, or piezoelectric materials. Printed circuit boards introduce substrates, copper layers, resins, glass fibres, solder, surface finishes, and other materials, while batteries introduce their own extraction, processing, manufacturing, transport, replacement, and recycling requirements. Environmental assessment of battery systems based on primary data has demonstrated that peripheral hardware can represent a substantial contribution to manufacturing impacts and that end of life treatment can alter overall global warming results. This observation is highly relevant to TinyML because many apparently minimal sensing nodes contain more physical components than their low computational load suggests, and repeated battery replacement can add further material and transport burdens over a long operational lifetime [5].

Scale further changes the sustainability equation. TinyML is attractive partly because its very low resource requirements make deployment possible across large numbers of distributed sensing nodes. At the individual device level, the environmental burden associated with a small microcontroller, a short inference event, or a modest amount of wireless communication may appear negligible. At fleet scale, however, the cumulative burden is determined by the number of devices, operating hours, sensing frequency, communication activity, maintenance interventions, battery replacements, firmware updates, manufacturing cycles, and eventual disposal. The environmental significance of TinyML should therefore be considered in terms of useful intelligence delivered over the entire service life of a device rather than energy consumed by one isolated inference. This scaling effect also creates an important methodological challenge: a design that appears optimal per device may not remain optimal when deployed across thousands, millions, or much larger populations of nodes. Research on AI energy use similarly emphasizes that even where energy efficiency improves, the overall consequence can increase as computational systems become more widely deployed and more deeply integrated into everyday activities [6].

End of life is another component of the TinyML footprint that deserves much greater attention. When a sensing node is discarded, its environmental relevance does not end with the cessation of inference. Printed circuit boards and electronic assemblies contain recoverable metals as well as materials that can create environmental burdens when poorly managed. The growing body of life cycle research on waste printed circuit boards indicates that recycling pathways differ substantially in their energy requirements, chemical consumption, material recovery performance, and associated environmental impacts. Recent process based assessment has demonstrated that alternative hydrometallurgical recycling routes can recover substantial quantities of copper, tin, zinc, and nickel while producing markedly different global warming impacts depending on the chemicals and processes employed. For

TinyML, this means that design decisions made during development can influence not only manufacturing and use phase impacts but also the feasibility and environmental performance of subsequent material recovery. Repairability, modularity, component identification, separability, material selection, reuse, refurbishment, and design for recycling should therefore be treated as integral characteristics of sustainable TinyML rather than as considerations added after a device becomes waste [7].

Taken together, these considerations establish the need for a genuinely lifecycle based understanding of TinyML sustainability. The central question is no longer simply how little energy a model consumes during inference, but where environmental burdens arise across model development, hardware fabrication, sensing, preprocessing, computation, communication, standby operation, battery provision, maintenance, transportation, replacement, refurbishment, recycling, and final disposal. A comprehensive perspective should consequently connect direct energy measurements with embodied energy, embodied carbon, material use, water consumption, resource depletion, device lifetime, deployment scale, energy source, and end of life pathways. Such an approach can reveal whether improvements in model efficiency merely reduce use phase electricity or actually decrease the environmental burden of the complete system. The aim of this review is therefore to critically examine the hidden environmental footprint of TinyML across its lifecycle, identify the principal energy, carbon, material, and waste hotspots, compare alternative architectures and deployment strategies, and evaluate how longer device lifetimes, efficient communication, responsible hardware selection, circular design, reuse, refurbishment, and improved recycling pathways can move TinyML from being merely ultra low power toward being genuinely sustainable.

II. METHODS

A. Scope and Evidence Base of the Review

1) Conceptual scope of the review

This review adopts a cradle to grave perspective for evaluating the environmental footprint of Tiny Machine Learning, or TinyML, systems. The central premise is that the environmental performance of a TinyML device cannot be inferred from its very low inference power alone. A complete assessment must account for the material and energy flows associated with obtaining raw materials, manufacturing semiconductors and sensors, assembling printed circuit boards, developing and training machine learning models, compressing and compiling models, deploying them to embedded hardware, operating the resulting device, communicating with other systems, maintaining hardware and software, replacing batteries or complete devices, and finally reusing, refurbishing, recycling, or disposing of the equipment. TinyML was originally promoted largely because machine learning could be moved from comparatively power intensive computing platforms to resource constrained devices. This shift can reduce operational computation and communication requirements, but it does not eliminate the environmental

burdens embedded in the physical hardware or in the upstream computing used to create the models [8].

The review therefore treats TinyML as a physical digital system rather than as an algorithm alone. The object being assessed may contain a microcontroller or accelerator, memory, sensors, analog interfaces, a printed circuit board, wireless communication components, a battery or energy harvesting subsystem, connectors, a protective enclosure, firmware, and supporting software. Each component contributes differently to the environmental profile of the complete system. The importance of this system view is reinforced by life cycle research on information and communication technologies, which shows that functional performance and environmental burden must be related to the service delivered by a product rather than to a single component or isolated operating event [9].

The environmental accounting boundary begins with extraction and processing of materials and continues through semiconductor fabrication, packaging, component manufacture, board assembly and device integration. It then follows the device through model preparation and deployment, active sensing, preprocessing, inference, local storage, communication, standby operation, maintenance, software updates, repair, battery replacement, hardware replacement, reuse, refurbishment, recycling, and disposal. This boundary is intentionally wider than the boundary commonly used in TinyML performance studies, where the main measurement may be current drawn by the processor during one inference. Such a processor level measurement is valuable for understanding computational efficiency but does not by itself quantify the environmental burden of the complete product [10].

The distinction between operational efficiency and life cycle sustainability is especially important because a very low energy inference can be repeated millions or billions of times across a large deployment. At the same time, the manufacturing burden of each device is incurred before the first inference occurs. Thus, an architecture that uses slightly more energy per inference may sometimes have a lower overall environmental footprint if it substantially extends device life, reduces battery replacement, supports repair, or uses less manufacturing intensive hardware. Conversely, extremely efficient inference can still be associated with a high total environmental burden when devices are frequently discarded or when the system requires energy intensive communication, maintenance, or cloud processing. A functionality based life cycle approach is therefore particularly appropriate because it encourages environmental impacts to be interpreted in relation to the function actually delivered by the technology [11].

Another important feature of the conceptual scope is that training is not excluded simply because inference takes place locally. The environmental cost of creating a TinyML model may involve dataset acquisition, storage, preprocessing, experimentation, hyperparameter optimization, architecture search, repeated training, validation, quantization, pruning, compilation, benchmarking, and deployment testing. Those activities can occur on workstations, institutional servers,

cloud infrastructure, or edge development platforms. The fact that the final model occupies only a few kilobytes or megabytes does not mean that its development was environmentally negligible [12].

Similarly, the scope recognizes that sensing and communication can account for a meaningful share of a device's energy demand. A processor may consume little energy while carrying out inference, yet the complete sensor node may draw much more energy because of sensing circuits, memory accesses, regulators, displays, indicators, wireless radios, or periodic synchronization. The same distinction applies to environmental impact: moving inference locally may reduce network traffic and cloud computation, but a complete assessment must determine whether those savings are large enough to compensate for the environmental burden of manufacturing and deploying the additional edge hardware [13].

2) Primary evidence base

The evidence base prioritizes empirical studies rather than relying primarily on conceptual discussions. For operational TinyML performance, preferred evidence includes direct measurements of voltage and current, energy per inference, average and peak power, inference latency, duty cycle, memory occupancy, processor utilization, communication energy, and battery discharge behavior. For physical environmental impacts, the preferred evidence includes component mass, material composition, hardware bill of materials, manufacturing process inventories, semiconductor process information, printed circuit board composition, battery chemistry and capacity, energy consumption during manufacturing, recycling energy requirements, material recovery yields, and measured or process based life cycle inventory data [10].

This emphasis on primary measurements is necessary because TinyML publications often report a model as low power without using a common measurement boundary. One study may report processor current during the neural network kernel, another may report board level power, and another may calculate energy from a nominal current value without separately measuring the sensor, regulator, memory, or radio. These values cannot be compared directly without identifying what was actually measured [11].

TinyML research has also developed rapidly since the technology became a distinct research area. Earlier work established the concept of deploying machine learning on highly constrained embedded devices, while subsequent studies have expanded the range of hardware, applications, model structures, and optimization methods. A rigorous environmental review therefore needs evidence that covers both established TinyML architectures and newer systems incorporating specialized accelerators, energy harvesting, dynamic model selection, and continuous learning [12].

For manufacturing, the evidence priority shifts from inference measurements to life cycle inventories. Bottom up assessment of IoT devices has shown that environmental

footprints can vary by more than two orders of magnitude depending on hardware complexity and configuration. That observation is directly relevant to TinyML because two devices may provide broadly similar sensing and inference functions while having very different numbers of components, semiconductor complexity, memory capacity, communication hardware, battery requirements, or enclosure materials [13].

Primary evidence is also sought for end of life processes. Recycling should not be assumed to be environmentally beneficial simply because valuable materials are recovered. Recycling itself requires collection, transportation, dismantling, separation, chemical treatment, mechanical processing, thermal treatment, water, and electricity. The appropriate question is therefore whether the recovered materials and avoided primary production provide sufficient environmental benefit to compensate for the burdens of recycling [14].

3) Evidence domains integrated by the review

The evidence was organized into five connected domains: computational footprint, hardware footprint, communication footprint, lifetime footprint, and end of life footprint. The computational footprint covers training, compilation, inference, standby processing, memory access, and model optimization. The hardware footprint covers semiconductor devices, sensors, printed circuit boards, batteries, regulators, connectors, packaging, and enclosures. The communication footprint covers wireless transmissions and the infrastructure required to transport or synchronize data. The lifetime footprint addresses durability, maintenance, updates, battery replacement, repairs, and replacement frequency. The end of life footprint covers collection, dismantling, reuse, refurbishment, recycling, recovery of materials, landfill, incineration, and other disposal pathways. These domains are not independent. Reducing model size can lower memory requirements and inference time, which can reduce operating energy. A smaller model may also permit the use of a less capable microcontroller, potentially reducing hardware complexity. Local inference can reduce communication demand but may require greater local processing or more capable sensors. Increasing battery capacity can lengthen operational life but adds material burden. A device designed for low energy operation may require more sophisticated power management circuitry. Consequently, environmental optimization requires system level analysis rather than optimization of one variable in isolation.

The evidence domains and their preferred measurements are summarized in Table I. The table is intended to establish a common vocabulary for the remainder of the review. In particular, it separates inference energy from hardware mass and manufacturing burden, while also identifying communication, battery, and recycling variables that are frequently omitted from processor centric TinyML evaluations. The use of these categories is consistent with the life cycle perspective recommended for ICT products and IoT devices [9], [10].

TABLE I
EVIDENCE DOMAINS AND PREFERRED MEASUREMENTS FOR TINYML LIFECYCLE SUSTAINABILITY

Evidence domain	Primary measurement or data	Typical units	Hardware or system studied	Lifecycle stage	Environmental relevance	Citation(s)
Inference	Energy per inference	μJ , mJ, J	MCU, DSP, NPU or accelerator	Use	Operational energy demand	[8]
Hardware	Material and mass inventory	g, mg	MCU, PCB, sensors, memory and enclosure	Manufacturing	Embodied energy and emissions	[9]
Battery	Capacity, chemistry and lifetime	mAh, Wh, cycles	Coin cell, Li ion and other storage	Use and end of life	Replacement burden and resource demand	[9]
Communication	Energy per transmission	μJ per packet, mJ per KB	BLE, Wi Fi, LoRa and related radios	Use	Network overhead and battery depletion	[10]
Manufacturing	Process and electricity inventory	kWh, MJ, kg CO _{2e}	Semiconductor, PCB and assembly	Manufacturing	Embodied footprint	[10]
Recycling	Process energy and recovery yield	kWh/t, %, kg recovered	PCB, battery and electronic components	End of life	Avoided primary material production	[9]
Lifetime	Service duration and replacement frequency	years, cycles, replacements/year	Complete TinyML system	Use and maintenance	Allocation of embodied impacts	[10]
Disposal	Treatment route and residual burden	kg waste, kg CO _{2e}	Complete device	End of life	Waste generation and pollution	[9]

MCU = microcontroller unit; DSP = digital signal processor; NPU = neural processing unit; PCB = printed circuit board; BLE = Bluetooth Low Energy; CO_{2e} = carbon dioxide equivalent. Citation note: Units include μJ = microjoule, mJ = millijoule, J = joule, mAh = milliampere hour, Wh = watt hour, kWh = kilowatt hour, and MJ = megajoule.

4) Measurement harmonization

A second methodological challenge is harmonization. TinyML studies vary in hardware, workload, input rate, model precision, operating voltage, clock frequency, compiler, framework, measurement instrument, and test conditions. Accordingly, raw numbers cannot always be compared directly. This review prefers energy per useful inference or energy per defined workload, accompanied by latency and measurement boundary [11].

The same principle applies to power. Average power without information about duty cycle can be misleading. A system that consumes 10 mW during an event lasting one

millisecond can have a lower energy contribution than a system consuming 2 mW continuously. For this reason, energy is treated as the primary operational quantity, with power and execution time retained as explanatory variables [12].

Carbon accounting also requires an explicit functional unit. Reporting kilograms of carbon dioxide equivalent per device is not sufficient if two devices have different operating lifetimes or perform different workloads. More meaningful units include kilograms of carbon dioxide equivalent per device year, per sensing event, per thousand inferences, per useful decision, or another functional unit appropriate to the application [13].

The preferred reporting basis is summarized in Table II. Each variable is paired with the information needed to make it interpretable. The table is especially important for inference energy because energy per inference is meaningful only when the model, input dimensions, sampling interval, precision, execution boundary, and workload are specified [14].

TABLE II
PREFERRED REPORTING BASIS FOR TINYML ENVIRONMENTAL AND ENERGY METRICS

Variable	Preferred reporting basis	Environmental interpretation	Citation(s)
Inference energy	Energy per inference in μJ , mJ or J plus workload definition	Direct computational energy	[8]
Power	Average and peak power plus voltage and current measurement boundary	Indicates instantaneous electrical demand	[8]
Latency	ms per inference or s per inference	Determines energy through execution duration	[9]
Duty cycle	Percentage active time and sleep time	Determines cumulative operational energy	[10]
Battery	Capacity, chemistry, cycle life and replacement interval	Determines lifetime and material burden	[10]
Carbon	kg CO _{2e} per device, device year or defined functional unit	Enables life cycle comparison	[9]
Materials	Mass and composition by component	Supports embodied impact assessment	[10]
Water and resources	Litres, kg Sb equivalent or method specific indicators	Captures impacts beyond carbon	[9]

Variable	Preferred reporting basis	Environmental interpretation	Citation(s)
Communication	Energy per packet, byte or reporting event	Quantifies radio and network overhead	[10]
End of life	Collection rate, recovery yield and process energy	Quantifies recycling and disposal consequences	[9]

CO₂e = carbon dioxide equivalent; Sb = antimony; the table also uses LCA = life cycle assessment where applicable to lifecycle measurement terminology. Citation note: ms = millisecond; s = second; kg = kilogram; kg Sb equivalent = kilograms of antimony equivalent.

Within the conceptual scope established in Sections II-A1 (Conceptual scope of the review) and II-A3 (Evidence domains integrated by the review), Fig. 1 presents an integrated cradle-to-grave lifecycle framework for TinyML, tracing the environmental pathway from raw material

extraction and processing, semiconductor and component manufacturing, and model development and deployment through sensing, preprocessing, inference, communication, standby operation, maintenance, replacement, reuse, refurbishment, recycling, and final disposal. The framework links these lifecycle stages to the energy and material inputs and emissions and waste outputs considered within the review boundary, while also showing feedback processes such as hardware upgrades and over-the-air model updates.

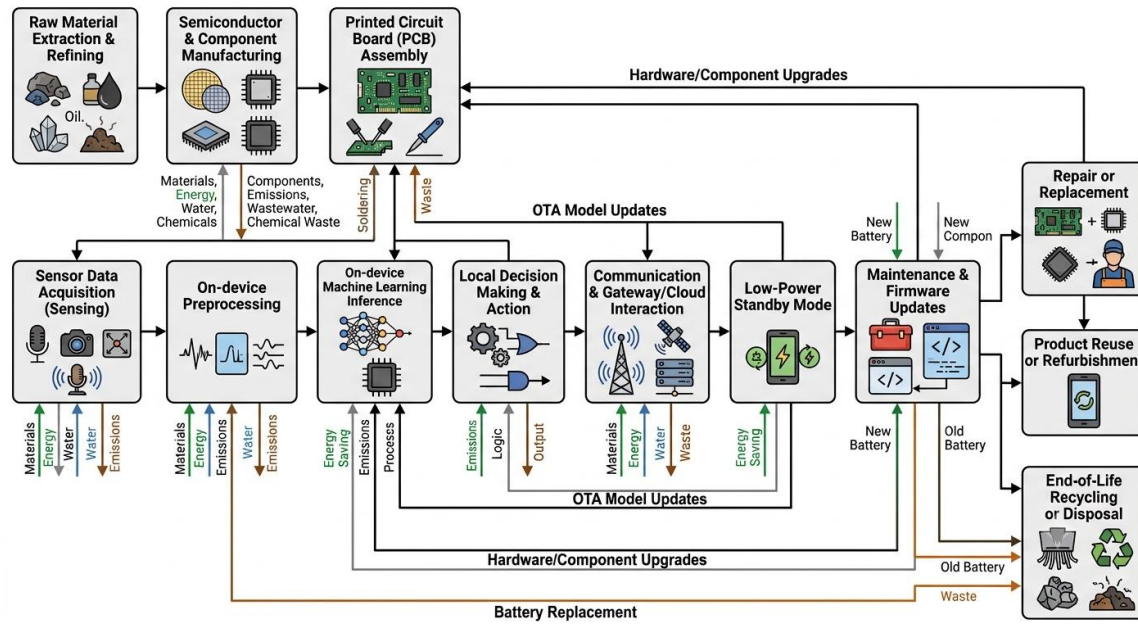


Fig. 1. A Comprehensive Lifecycle Analysis Diagram of the TinyML Environmental Footprint [8]–[10].

Grey arrows represent main process flow; black arrows indicate feedback loops including hardware upgrades and OTA model updates. Colored blocks detail specific lifecycle phases with environmental factors listed below; Abbreviations: OTA stands for Over-The-Air, PCB stands for Printed Circuit Board.

B. TinyML Architecture, Hardware Stack and Environmental Relevant System Components

1) Computational architecture

TinyML commonly operates on microcontroller class processors that combine modest computational capacity with small amounts of SRAM and non-volatile memory. These devices are designed for low energy embedded operation and can execute compact machine learning models without relying continuously on a remote cloud service. More recent architectures expand this model by incorporating digital signal processing units, vector extensions, dedicated neural accelerators, or heterogeneous combinations of general purpose processors and specialized compute engines. The basic computational stack can be represented as a hierarchy.

At the lowest level is the sensor or sensing interface. Above it lies signal conditioning and analog to digital conversion. The microcontroller then performs preprocessing and feature extraction before executing the inference model. SRAM stores activation values and temporary buffers, while flash or other non-volatile memory stores model parameters, firmware, and configuration data. Some systems also include external memory, digital signal processors, or dedicated neural processing units [12].

2) Sensor to inference data path

A TinyML device usually begins with physical sensing rather than directly with computation. A simplified data pathway is physical phenomenon → sensor → analog front end → analog to digital converter → preprocessing → feature extraction → inference → decision → communication or actuator response. This pipeline is environmentally significant because every stage can consume energy. Sensors may need continuous excitation or bias current. Analog circuits may remain active even when the neural network is asleep. ADCs can consume energy at every sampling event. Preprocessing

may require filtering, windowing, transforms, normalization, segmentation, or feature generation before a model can run [13].

3) Hardware resource constraints and environmental implications

Clock frequency, SRAM capacity, flash capacity, operating voltage, memory bandwidth, and accelerator utilization determine which models can be executed and how efficiently they can run. Resource constrained systems commonly use lower clock frequencies and aggressive sleep modes to reduce energy. However, reducing clock frequency can increase execution time and potentially offset some of the power savings. This relationship illustrates why power should not be optimized without considering time. Dynamic power generally rises with voltage and switching activity, while total energy depends on how long the processor remains active [14].

4) Hardware architectures and their energy trade offs

Cortex M class MCUs remain important because they offer low energy operation, integrated peripherals, and broad software support. RISC V based platforms add architectural flexibility and can incorporate vector or specialized instructions. Digital signal processor enhanced MCUs can accelerate common operations such as filtering and transforms,

while neural accelerators can improve the efficiency of convolutional or matrix based workloads. A heterogeneous architecture can deliver substantial energy benefits when the accelerator is well matched to the workload. However, the advantage depends on utilization [15].

5) System level versus processor level energy

Processor level measurements can substantially underestimate the real electrical demand of a TinyML node. A processor may draw very little current during inference while a camera, microphone, inertial sensor, illumination system, memory chip, voltage regulator, radio module, or display remains active. This is particularly important for sensing systems. Image based TinyML systems may spend significant energy acquiring frames, illuminating scenes, converting signals, and moving image data even when the neural network itself is efficient. Similarly, an environmental sensor may consume energy continuously to maintain its sensing chemistry or heating element [16].

The architecture and system components relevant to these environmental tradeoffs are summarized in Table III, which links common hardware configurations to their principal environmental considerations. The table is intended to guide interpretation of operational measurements in later sections.

TABLE III
TINYML HARDWARE ARCHITECTURES AND COMPONENTS: ENVIRONMENTAL ADVANTAGES, TRADE-OFFS AND MEASUREMENT PRIORITIES

Architecture or component	Primary role	Typical environmental advantage	Potential environmental trade off	Measurement priority	Citation(s)
Cortex M MCU	General purpose embedded inference	Low operating power and mature integration	Limited memory and compute can increase optimization effort	Board and processor energy	[12]
RISC V MCU	Flexible embedded compute	Custom instructions and vector support	Software and hardware ecosystem variation	Energy and execution time	[12]
DSP assisted MCU	Signal processing	Efficient filtering and feature extraction	Additional silicon and memory	Sensor to feature energy	[13]
NPU enabled MCU	Neural inference acceleration	Lower latency and potentially lower inference energy	Additional silicon complexity and embodied burden	Accelerator utilization and system energy	[13]
SRAM	Activation and working memory	Fast local access	Leakage and area burden	Memory access energy	[12]
Flash	Model and firmware storage	Nonvolatile local storage	Program memory access energy and manufacturing burden	Read energy and capacity	[12]
Sensor subsystem	Data acquisition	Enables local intelligent sensing	Can dominate standby and active energy	Sensor plus AFE energy	[14]
Radio	Communication	Enables remote reporting	Transmission can exceed local inference energy	Energy per packet	[14]
Power management	Voltage regulation and charging	Efficient supply conversion	Additional components and conversion losses	Input to load efficiency	[13]
External memory	Larger model and data storage	Permits more complex workloads	Higher access energy and additional components	Transfer energy	[12]

MCU = microcontroller unit; RISC V = reduced instruction set computer V; DSP = digital signal processor; NPU = neural processing unit; SRAM = static random access memory; AFE = analog front end. Citation note: The table also refers to Flash

memory for nonvolatile program and model storage; Flash is a memory technology designation rather than an abbreviation.

C. Operational Energy Consumption of TinyML Inference

1) Energy per inference across TinyML workloads

Operational inference energy varies considerably with workload, architecture, model complexity, input size, precision, execution frequency, and hardware implementation. Keyword spotting generally involves relatively small input representations and compact models, whereas image classification or object detection can require substantially more memory movement and arithmetic. Vibration analysis, environmental classification, anomaly detection, and time series prediction occupy intermediate positions depending on sampling rate and model architecture. Recent empirical work provides a useful picture of this diversity. In a 2025 Scientific Reports study of TinyML based visual wake word detection, 8 bit post training quantization reduced storage requirements by approximately three to four times, producing model footprints in the reported range of 286 to 536 KB on platforms constrained to about 1 MB of flash and 256 KB of SRAM. Inference latency ranged from 3.47 to 14.98 ms per frame across the evaluated configurations, while the study reported measured inference energy between 10.6 and 22.1 J per frame under its measurement boundary. These values should be interpreted with care because the reported measurement boundary and workload are specific to the experimental setup and should not be generalized to all TinyML applications [9], [16].

2) Power, latency and energy performance trade offs

The fundamental relationship among power, execution time, and energy can be expressed as $E = P \times t$, where E is energy, P is average power during the measured interval, and t is execution time. This simple relationship has important implications for TinyML. Lower power does not automatically imply lower energy per inference. A processor that operates at very low power but requires substantially longer completing a model can consume similar or greater total energy than a faster configuration. Conversely, a high performance accelerator may have a relatively high instantaneous power draw but complete the inference so quickly that its total energy is lower. The tradeoff becomes more complex when duty cycle is considered [17].

3) Memory access energy versus arithmetic energy

A common mistake in TinyML energy analysis is to focus heavily on multiply accumulates operations while giving insufficient attention to data movement. A neural network may contain a relatively small number of arithmetic operations but require repeated movement of weights and activations

between flash, SRAM, caches, buffers, and accelerators. Local memory reuse can therefore reduce energy even when the mathematical model remains unchanged. Operator fusion can reduce intermediate storage and memory transfers. Better scheduling can reduce repeated loading of weights. Quantization can reduce the amount of data moved through memory and buses [18].

4) Duty cycle and always on sensing

Always on sensing is one of the defining challenges of TinyML environmental efficiency. A device may need to observe its environment continuously even when the target event is rare. The most effective strategy is often to keep the sensing chain and a lightweight detection stage active while placing the more demanding processor or accelerator into deep sleep until a trigger occurs. Event driven sensing can therefore reduce cumulative energy dramatically. Energy harvesting makes this approach even more important because available energy can vary over time [19].

5) Effects of quantization, pruning and architecture compression

Quantization reduces the numerical precision used to represent model weights and activations. Pruning removes parameters or connections that contribute relatively little to model performance. Distillation transfers behavior from a larger teacher network to a smaller student model. Architecture search can identify model structures that achieve a desired accuracy within a hardware budget. The energy implications are often positive, but they are not guaranteed to scale linearly with model compression. Hardware mapping can also affect the result [20].

6) Measurement uncertainty and benchmarking problems

TinyML energy studies are difficult to compare because measurement boundaries and instruments differ. A current probe may measure the entire development board, while another experiment may measure only the processor rail. Some studies calculate energy from average current and execution time, whereas others use high resolution power traces. Peak current also matters. A device with a low average current can have high short duration peaks that affect regulator efficiency and battery behavior. Battery powered systems may therefore behave differently from laboratory power supplies [21].

The operational evidence is consolidated in Table IV. The table emphasizes measurable performance rather than theoretical operation counts and highlights the need to retain the measurement boundary when interpreting reported energy.

TABLE IV
OPERATIONAL ENERGY AND INFERENCE PERFORMANCE ACROSS TINYML WORKLOADS

Study	Task	MCU or platform	Model or approach	Precision	Input or workload	Reported latency	Power or energy result	Measurement interpretation	Citation(s)
TinyML object detection study	Visual wake word and object detection	Multiple MCU class platforms	Quantized MobileNet variants	8 bit	Visual Wake Words frames	3.47 to 14.98 ms/frame	10.6 to 22.1 J/inference reported under the study boundary	Complete experimental inference measurement	[16]

Study	Task	MCU or platform	Model or approach	Precision	Input or workload	Reported latency	Power or energy result	Measurement interpretation	Citation(s)
Model compression study	EIT hand gesture recognition	Arduino Nano BLE and other edge platforms	1D CNN with combined compression	Compressed integer model	EIT gesture signals	Large reductions reported	About 52% average power reduction on Arduino Nano BLE; 34.05% on Raspberry Pi 4	Platform level comparison	[17]
Energy aware TinyML	Variable sensing workloads	Zero energy devices	Multiple selectable models	Workload dependent	Intermittent energy supply	Energy availability dependent	Model selection optimized for available energy	System level energy aware execution	[18]
On device training study	Continuous learning and pruning	ESP32 and STM32	Pruned adaptive networks	Resource constrained	TON IoT based workloads	Training dependent	Energy savings targeted through pruning and local adaptation	MCU level experimental assessment	[19]
Reduced precision study	On device training	RISC V RV32 MCU	ResNet8 and DS CNN primitives	FP16 and FP32	Image classification and keyword spotting	17.1 ms and 6.4 ms per reported training step	Reduced precision improved execution efficiency	Processor and kernel level	[20]

MCU = microcontroller unit; EIT = electrical impedance tomography; CNN = convolutional neural network; BLE = Bluetooth Low Energy; FP16 = 16 bit floating point; FP32 = 32 bit floating point. Citation note: 1D CNN = one dimensional convolutional neural network; DS CNN = depthwise separable convolutional neural network; ms = millisecond; J = joule.

A further implication is that a TinyML environmental benchmark should report both energy intensity and service intensity. For example, joules per inference should be accompanied by the number of inferences required per day, because a 1 μ J inference executed once per hour has a very different annual energy contribution from the same inference executed continuously. Similarly, communication and standby energy must be included for always on devices. A neural network may represent only a small portion of total annual electricity use when the device spends most of its life sensing, waiting, transmitting, or periodically synchronizing.

D. Embodied Energy and Embodied Carbon of TinyML Hardware

1) Semiconductor manufacturing as an upstream footprint

The environmental footprint of TinyML begins before a model is loaded onto the device. Semiconductor manufacturing requires raw materials, wafer processing, clean room operation, ultrapure water, process chemicals, specialty gases, electricity, packaging, and testing. The resulting embodied impact becomes part of the device's environmental debt before it performs its first inference. This upstream burden is particularly important because advanced semiconductor production can involve highly controlled environments and multiple process steps. The environmental burden is not determined simply by chip mass. Process complexity, die area, packaging choice, materials, electricity mix, water scarcity, manufacturing location, and testing all contribute [22].

2) MCU and NPU embodied footprint

A conventional MCU can integrate processor logic, memory, timers, communication interfaces, analog peripherals, and power management support. Specialized NPU equipped devices introduce additional neural acceleration hardware, which can improve energy performance during inference. The environmental tradeoff between MCU and NPU designs is not straightforward. If the accelerator substantially reduces energy throughout a long operating life, its additional embodied burden can be amortized over many inference cycles. If the device performs only occasional inference, however, the additional hardware may provide little operational benefit [22].

3) Sensors as environmental hotspots

The sensor is often overlooked in TinyML environmental accounting because machine learning discussions naturally focus on the processor. Yet sensors can contain materials, packaging, signal conditioning components, and manufacturing processes that contribute significantly to environmental impact. Cameras are particularly relevant because image sensing can require relatively complex semiconductor arrays, optical assemblies, illumination, image processing, and memory. MEMS microphones and inertial sensors are generally smaller but may still require specialized fabrication and packaging. Gas sensors and biosensors can introduce additional materials and chemical processing requirements [21].

4) PCB and interconnect footprint

Printed circuit boards provide the mechanical and electrical foundation for TinyML devices. Their footprint includes laminates, resins, glass fibers, copper layers, solder, surface finishes, connectors, and the energy associated with fabrication and assembly. PCB complexity tends to increase with the number of components and signal requirements. Additional memory devices, sensors, communication modules, power management components, connectors, and displays

therefore affect not only the bill of materials but also the embodied environmental footprint [23].

5) Power management hardware

Power management components include voltage regulators, charging circuits, power management integrated circuits, protection devices, switches, energy harvesting interfaces, and sometimes dedicated battery management units. These components are often small by mass but can influence system efficiency continuously. A regulator with poor conversion efficiency increases operational losses and therefore increases the amount of energy that must be supplied by the battery or energy harvester [22].

6) Batteries and energy storage components

Battery choice is a particularly important environmental variable because the battery is both a material product and a consumable component when the device is designed for long term operation. Primary batteries can simplify deployment but

have finite capacity and must be replaced when depleted. Rechargeable batteries can support longer device life but introduce additional manufacturing, charging, degradation, and eventual recycling burdens [24].

7) Packaging, enclosure and transportation

TinyML products may be housed in plastics, aluminium, composites, silicone materials, glass, or combinations of these. Enclosures can contain a larger mass than the semiconductor itself and may contribute meaningfully to embodied environmental impact. Transportation adds another burden. Devices can move through several stages from component fabrication to board assembly, final integration, distribution, installation, maintenance, and end of life processing [20].

The importance of the hardware stage is summarized in Table V, which organizes the major components of a representative TinyML device according to mass, material, manufacturing process, embodied burden, and dominant environmental hotspot.

TABLE V
EMBODIED ENVIRONMENTAL FOOTPRINT OF MAJOR TINYML HARDWARE COMPONENTS

Component	Typical physical contribution	Major materials	Manufacturing process	Main embodied energy or GHG driver	Water or resource implication	Potential hotspot	Citation(s)
MCU	Low mass semiconductor package	Silicon, metals, polymers and packaging materials	Wafer fabrication and packaging	Semiconductor process electricity and materials	Ultrapure water and process chemicals	Chip fabrication	[20], [22]
NPU or accelerator	Additional silicon area	Silicon, metals and packaging materials	Advanced or specialized semiconductor processing	Die area, fabrication and packaging	Water and process chemicals	Added silicon and packaging	[22]
Sensor	Small to moderate mass depending on type	Silicon, ceramics, metals, polymers and specialized materials	MEMS, CMOS, chemical or optical fabrication	Specialized manufacturing	Water, chemicals and material processing	Camera or specialized sensor	[21]
PCB	Moderate physical mass	FR 4, copper, solder, resin and glass fiber	Lamination, etching, drilling and assembly	Electricity and copper processing	Chemicals and material recovery	Board fabrication and assembly	[23]
PMIC and regulators	Low mass but continuous electrical role	Semiconductor and passive component materials	Semiconductor and electronic assembly	Manufacturing plus conversion losses	Material and process burden	Operational losses	[22]
Battery	Variable mass and capacity	Lithium, nickel, cobalt, manganese, graphite or chemistry specific materials	Cell production and assembly	Material extraction and cell manufacturing	Mining and processing impacts	Replacement and material supply	[24]
Enclosure	Often larger mass than active electronics	Plastics, aluminium, glass or composites	Injection molding, machining or forming	Material processing and manufacturing	Material extraction and production	Material mass	[20]
Connectors and wiring	Low to moderate mass	Copper, plastics and plating metals	Metal processing and assembly	Material and manufacturing energy	Metal extraction	Copper and plating materials	[23]
Packaging	Variable	Paperboard, polymers, foams and protective materials	Conversion and printing	Material processing	Forestry or petrochemical inputs	Single use packaging	[20]

MCU = microcontroller unit; NPU = neural processing unit; MEMS = microelectromechanical systems; CMOS = complementary metal oxide semiconductor; PCB = printed circuit board; PMIC = power management integrated circuit; GHG = greenhouse gas. Citation note: FR 4 denotes FR 4

glass fiber reinforced laminate used in printed circuit boards; CO_{2e} denotes carbon dioxide equivalent where the environmental burden is expressed in carbon terms.

E. Model Development, Compression, Deployment, Communication and Lifecycle Energy

1) Training energy for TinyML models

Although TinyML inference occurs on constrained hardware, model development frequently happens on much larger computing platforms. Training can therefore produce a significant upstream energy requirement, particularly when models are repeatedly retrained during experimentation. The training footprint includes dataset preparation, preprocessing, model construction, repeated experiments, hyperparameter tuning, validation, storage, and model selection. A model that ultimately runs for years on a tiny MCU may have been produced through hundreds or thousands of experimental training runs [25].

2) Dataset preparation and preprocessing energy

Dataset preparation is often ignored in environmental accounting because it occurs before model training. In practice, data may need to be collected from physical sensors, transferred to storage, cleaned, labeled, augmented, filtered, transformed, and repeatedly processed. For time series applications, feature generation may include Fourier transforms, statistical windows, filtering, normalization, segmentation, and labeling. For image applications, resizing, augmentation, annotation, and storage can introduce substantial computational and storage demands [25].

3) Hyperparameter search and model iteration

Model iteration can be more energy intensive than the final training run because developers commonly test multiple model architectures, learning rates, feature sets, precisions, pruning thresholds, and training schedules. Neural architecture search is a particularly important example. Research has shown that conventional architecture search can consume substantial computational resources because candidate models must be sampled, trained, evaluated, and compared repeatedly. A hardware aware latency prediction framework can reduce the computational burden of the search process [26].

4) Compression and deployment energy

Model compression includes quantization, pruning, knowledge distillation, architecture simplification, compilation, kernel optimization, and hardware specific conversion. Compression has at least three distinct environmental effects. First, the compression process itself consumes computational energy. Second, the resulting model may reduce the energy required during deployment. Third, the smaller model may permit the use of less capable hardware [17].

5) Communication energy across wireless protocols

Communication can become one of the most important energy components of an IoT system because radios must activate, establish or maintain links, transmit packets, receive acknowledgements, and sometimes remain in listening states. BLE is generally designed for low energy short range communication. LoRa and LoRaWAN support long range low data rate applications. Wi Fi offers higher throughput but can require greater energy. Cellular connectivity can support broad

geographic coverage but generally introduces a different energy and infrastructure profile [26].

6) Edge cloud dependency

Three deployment configurations are particularly relevant. In the first, inference occurs entirely on the device and only final decisions are transmitted. In the second, inference occurs locally but the device periodically synchronizes models, metadata, diagnostics, or summary information with a server. In the third, the device performs preliminary processing locally but transfers complex cases to a gateway or cloud service. The environmentally preferable configuration depends on workload and operating conditions [27].

7) Maintenance, firmware updates and replacement

A TinyML environmental assessment should extend beyond the initial deployment. Firmware updates may involve local computation, wireless communication, cloud services, developer infrastructure, testing, and device restarts. Frequent updates can shorten battery life and increase communication energy. However, updates may extend useful device life by fixing vulnerabilities, correcting algorithms, or improving energy efficiency. Consequently, the environmental effect of updates may be negative or positive depending on whether they extend the lifetime of the hardware [27].

8) End of life, reuse and recycling

End of life is often treated as an afterthought in digital technology discussions, but TinyML devices can become a significant stream of small electronic waste if deployed in very large numbers. The environmental outcome depends strongly on whether the device is reused, refurbished, dismantled for components, recycled for materials, incinerated, or landfilled. Reuse generally has the potential to avoid the environmental burden associated with manufacturing a replacement product, although refurbishment itself consumes resources [24].

9) System level lifecycle energy model

The review uses the following conceptual accounting relationship: $E_{total} = E_{training} + E_{deployment} + E_{hardware} + E_{sensing} + E_{inference} + E_{communication} + E_{maintenance} + E_{EOL}$. $E_{training}$ represents model development and repeated training; $E_{deployment}$ represents model conversion, compilation, programming, testing, and installation; $E_{hardware}$ represents manufacturing related energy allocated to the device; $E_{sensing}$ represents sensor and signal acquisition energy; $E_{inference}$ represents computational energy during prediction; $E_{communication}$ represents radio and network related energy; $E_{maintenance}$ represents firmware updates, diagnostics, repairs, and replacement activities; and E_{EOL} represents collection, transport, dismantling, recycling, recovery, disposal, and associated processing [25].

The lifecycle accounting framework is summarized in Table VI. This table should be used throughout subsequent sections as the reference framework for discussing whether an environmental intervention affects development energy, manufacturing burden, operational energy, communication demand, lifetime, or end of life.

TABLE VI
LIFECYCLE ACCOUNTING FRAMEWORK FOR ASSESSING THE ENVIRONMENTAL FOOTPRINT OF TINYML

Lifecycle stage	Primary energy source or burden	Suggested measurement	Functional unit	Principal environmental question	Citation(s)
Development	Training, validation and model search	kWh per trained model or experiment	Per model or model family	How much energy was required to create the deployed model?	[25]
Dataset preparation	Data collection, preprocessing and labeling	kWh or computational hours	Per dataset or deployed model	How much upstream computation supports the model?	[25]
Manufacturing	Semiconductor, PCB, sensor and battery production	Process or LCA inventory	Per device	What environmental burden exists before use?	[20], [22]
Deployment	Programming, compilation, configuration and installation	Measured electricity or computational energy	Per device	What is required to make the device operational?	[25]
Sensing	Sensors, analog front end and ADC	mJ per sensing interval or Wh/year	Per device year	How much energy is consumed before inference?	[26]
Inference	MCU, DSP or NPU	μ J, mJ or J per inference	Per device year or workload	What is the computational operating burden?	[16], [17]
Communication	BLE, Wi Fi, LoRa, cellular or other radio	μ J per packet or mJ per KB	Per device year	How much energy is associated with data transfer?	[26], [27]
Standby	Sleep and background circuitry	mW or Wh per year	Per device year	Does idle operation dominate the duty cycle?	[26]
Maintenance	Firmware updates, diagnostics and replacement	Wh per update and replacements per year	Per device year	Does maintenance shorten or extend useful life?	[27]
End of life	Collection, dismantling and recycling	kWh/kg, recovery yield and material credits	Per kg or device	What burdens and avoided impacts result from disposal?	[24]
Complete lifecycle	All stages combined	LCA with common functional unit	Device year or useful service	What is the total environmental footprint?	[25]

LCA = life cycle assessment; MCU = microcontroller unit; DSP = digital signal processor; NPU = neural processing unit; ADC = analog to digital converter; BLE = Bluetooth Low Energy. Citation note: Wi Fi refers to wireless networking technology; LoRa = Long Range; EoL = end of life; Wh = watt hour; kWh = kilowatt hour.

This system level perspective is supported by the broader green AI literature, which argues that both computing hardware and software behavior influence environmental performance. It is further supported by IoT environmental research showing that production impacts can vary substantially among hardware configurations and that complete lifecycle accounting is required when large numbers of devices are deployed. Accordingly, the remaining sections of this review treat TinyML as a lifecycle system in which computational optimization, hardware selection, sensing, communication, lifetime extension, and end of life strategy are interconnected. This approach is necessary to avoid a narrow interpretation in which ultra-low power is equated automatically with low environmental impact [25]–[27].

III. RESULTS AND DISCUSSION

A. Where the TinyML Lifecycle Footprint Actually Occurs

1) Operational versus embodied footprint

The distinction between operational and embodied footprint is central to understanding TinyML sustainability. Operational footprint refers primarily to electricity or other energy consumed while the device is sensing, preprocessing, performing inference, transmitting information, receiving updates and remaining in standby. Embodied footprint refers to environmental burdens associated with extraction of raw

materials, semiconductor fabrication, component manufacturing, assembly, transport and preparation of the device for use.

For conventional computing platforms, prolonged operation can produce a large cumulative footprint because processors, displays, storage devices and communication systems consume meaningful amounts of electricity over long periods. TinyML changes this relationship because the computational component is deliberately designed to operate within severe energy constraints. Consequently, the environmental contribution of one inference can become extremely small. However, the manufacturing footprint does not decrease at exactly the same rate. A low energy microcontroller remains a semiconductor product requiring wafer fabrication, packaging, testing and supporting materials.

This asymmetry means that increasing operational efficiency indefinitely does not necessarily eliminate the environmental significance of manufacturing. A device that consumes only a small amount of electricity each day may continue carrying essentially the same embodied burden whether it operates for one year or ten years. The longer lifetime simply allows that initial burden to be allocated across a larger amount of delivered service. This is one of the most important reasons lifecycle thinking is necessary when assessing ultra-low power AI.

The problem becomes more evident when TinyML nodes are deployed as distributed sensing networks. The manufacturing burden is multiplied by every physical device, even if each individual node has a very small energy demand. Research on IoT edge devices has shown that production carbon footprints can vary by more than two orders of magnitude between different hardware profiles. This variation

means that two devices delivering apparently similar TinyML functions may have very different environmental footprints before the first inference is even performed [32].

The distinction between embodied and operational impacts should therefore be treated as a dynamic balance rather than as two isolated quantities. Manufacturing tends to become comparatively dominant when the device is highly energy efficient, operates intermittently, uses low carbon electricity or has a short useful life. Operational energy becomes progressively more important when the device has a high duty cycle, performs frequent sensing and communication, requires regular charging, uses carbon intensive electricity or remains active for many years. The environmental optimum is consequently influenced by both hardware efficiency and service conditions.

2) Processor, sensor, PCB and battery contributions

The processor or MCU is an obvious focus of TinyML analysis because it executes the neural network. Nevertheless, it should not automatically be assumed to dominate the total environmental footprint. A TinyML node can contain several components whose environmental burdens may be substantial relative to the processor itself. These include MEMS sensors, image sensors, microphones, accelerometers, gyroscopes, power management devices, memory, printed circuit boards, connectors, antennas, batteries and protective structures.

The processor becomes particularly important when the system uses comparatively sophisticated semiconductor technologies, additional embedded memory or dedicated acceleration logic. Manufacturing complexity can increase resource demand even when the resulting processor offers improved energy efficiency during inference. At the same time, sensor requirements can strongly influence the environmental profile. A simple temperature sensor and a camera sensor should not be treated as equivalent components. Cameras generally require substantially more supporting electronics and may encourage higher processing and communication loads because the information volume generated by imaging is much greater.

The printed circuit board is another potentially overlooked contributor. The PCB provides mechanical support and electrical connections for the processor, sensors, power management circuitry and communications components. It contains copper, fiberglass or other reinforcement materials, polymer resins, solder, surface finishes and, depending on design, valuable and potentially hazardous elements. Although its mass may appear small, manufacturing and end of life processing can contribute significantly to the footprint of electronic equipment.

Batteries require separate treatment because their importance is determined by both manufacturing and replacement. A TinyML system designed for five or ten years of unattended operation may need one high quality battery, whereas a smaller battery selected for low upfront cost may require several replacements. The environmental burden associated with mining, refining, processing and manufacturing battery materials therefore becomes linked to device lifetime and duty cycle rather than simply battery capacity.

A useful interpretation is that TinyML hardware should be analysed as a component hierarchy. At the top level, the node provides a sensing and decision service. At the next level, that service depends on computation, sensing, power and communication. Each of those functions depends on physical components with distinct material and manufacturing profiles. Consequently, lifecycle hotspot analysis should identify both component mass and component complexity rather than attributing the complete footprint to the microcontroller.

3) Carbon hotspot identification

Carbon hotspot identification is especially important because a TinyML system may appear environmentally benign when assessed only through its measured inference current. The principal hotspot may instead occur before deployment. Research on IoT edge devices demonstrates that the production carbon footprint of hardware can vary dramatically depending on component selection and hardware complexity [31]. This finding is directly relevant to TinyML because the architecture of an ultra-low power node can vary from a basic MCU and low complexity sensor to a more sophisticated platform containing high resolution sensors, specialized accelerators, nonvolatile memory and wireless interfaces.

The semiconductor stage can become a major hotspot because semiconductor fabrication requires energy intensive processing, ultrapure water, controlled environments, process chemicals and multiple fabrication stages. The environmental significance of semiconductor production therefore extends well beyond the silicon itself. Advances in process technology may improve the computational efficiency of a chip while simultaneously requiring more complex manufacturing procedures.

Sensing can also become a material hotspot when sensors are numerous or sophisticated. Always on microphones, imaging systems and multisensor platforms can increase both manufacturing and operating requirements. A sensor that consumes little power does not necessarily have a negligible embodied footprint. Its environmental significance depends on how it is manufactured, how much material it contains and how often it must be replaced.

Communication represents another hotspot because a radio link can cause the node to consume substantially more energy than local inference. This is particularly important when the underlying application sends raw or lightly processed sensor data rather than compact inference results. TinyML reduces this burden when local inference permits transmission of only events, classifications or anomalies instead of continuous raw measurements. However, communication infrastructure should still be included within the broader system boundary whenever it is necessary for service delivery.

Long term operation can become the principal hotspot when the system operates continuously, particularly in applications involving frequent sensing, inference and wireless communication. This is why lifecycle assessment should distinguish between always on and event driven operation rather than reporting a single generic value for "TinyML energy consumption."

4) Why “low power” does not automatically mean “low carbon”

The phrase “low power” describes an engineering property, whereas “low carbon” describes an environmental outcome. The two concepts are related but not interchangeable. A device can consume only a few milliwatts during operation while requiring a comparatively resource intensive manufacturing process. Conversely, a device with somewhat higher operational energy could potentially have a lower total lifecycle impact if it has a longer lifetime, avoids frequent replacement, uses recyclable materials and operates on low carbon electricity.

The electricity supply is particularly important. One kilowatt hour does not have the same environmental significance in every location. A TinyML node powered by a low carbon grid will accumulate a smaller use stage carbon footprint than an electrically identical node operating in a carbon intensive electricity system. Consequently, global TinyML deployments should not use a single universal grid factor without accounting for geographic differences.

Device lifetime creates another important relationship. If a TinyML node is discarded after two years because its sensor fails, its model becomes incompatible with the application, its battery is not replaceable or a minor hardware fault cannot be repaired, the embodied burden must be allocated across only two years of service. Extending that device to five or ten years can distribute the initial manufacturing burden across substantially more delivered sensing service.

This interpretation is supported by wider ICT lifecycle research showing that manufacturing and material efficiency can dominate the environmental profile of electronics and that extending the replacement interval can produce substantial reductions in annualized carbon footprint [32]. For TinyML, the implication is clear: energy efficient inference should be combined with durable hardware, maintainable firmware, replaceable energy storage and long service intervals.

5) Carbon versus broader environmental impacts

A carbon focused analysis is useful but incomplete. TinyML systems may contribute to several other environmental impact categories that are poorly represented by a single CO₂ equivalent value. Semiconductor fabrication, for example, requires substantial quantities of water and can generate wastewater and process residues. The global semiconductor industry has been reported to have significant water and energy requirements, illustrating the importance of water footprint alongside carbon footprint when evaluating electronic hardware [30].

Water consumption can be particularly important in regions where semiconductor production is located in water stressed basins. Resource depletion is similarly relevant because processors, connectors, sensors, PCBs and batteries contain materials that require mining and refining. Even very small quantities become significant when multiplied across millions or billions of devices.

Additional impact categories include acidification, eutrophication, fossil resource use, mineral resource scarcity, particulate matter formation, human toxicity and ecotoxicity. These impacts may arise from mining, chemical processing, electricity generation, semiconductor fabrication and waste treatment. A design that performs well in terms of carbon emissions could therefore perform less favourably in another category.

The correct objective is consequently not simply to minimize CO₂e per inference. Instead, TinyML environmental optimization should seek to minimize lifecycle impacts across a relevant set of indicators while maintaining functional performance. Carbon should remain an important metric, but not the only one.

Table VII compares the principal lifecycle hotspots associated with representative TinyML system configurations and distinguishes manufacturing, sensing, operation, communication and end of life contributions. The comparison illustrates why the dominant burden can shift with hardware complexity, duty cycle, electricity mix and service lifetime.

TABLE VII
LIFECYCLE HOTSPOT COMPARISON ACROSS TINYML ARCHITECTURES

Representative TinyML architecture	Manufacturing hotspot	Operational hotspot	Communication contribution	Battery / power contribution	Likely dominant lifecycle stage	Principal non carbon concern	Citation(s)
Simple MCU + temperature sensor	MCU, PCB and sensor fabrication	Very low intermittent inference	Low when event driven	Low to moderate	Manufacturing	Materials and resource use	[28]
MCU + motion sensor	MCU, sensor and PCB	Duty cycle dependent	Low to moderate	Moderate	Manufacturing or use	Battery resources	[29]
MCU + audio sensing	Sensor and processor complexity	Continuous sensing and inference	Moderate to high	Moderate to high	Manufacturing plus operation	Water and material demand	[30]
MCU + wireless anomaly detection	Electronics and radio subsystem	Inference plus communication	Potentially high	Moderate to high	Use phase in high duty cycles	Electricity carbon intensity	[31]
Multi sensor TinyML gateway	Processor, memory, sensors, radio and PCB	Continuous processing and communication	High	High	Manufacturing and operation	Resource depletion and e waste	[32]

MCU = microcontroller unit; PCB = printed circuit board; EoL = end of life; CO₂e = carbon dioxide equivalent. Citation

note: The table uses non carbon to distinguish environmental concerns beyond greenhouse gas emissions, including material

and resource demand.

The environmental trade-offs examined in Sections III-A1 to III-A5 are synthesized in Fig. 2, which shows how hardware complexity and component composition, device lifetime, duty cycle, energy source, and communication intensity can redistribute environmental burdens across manufacturing, model development, deployment, operation,

communication, maintenance, replacement, and end-of-life stages. The figure connects the distinction between operational and embodied impacts with component-level contributions and hotspot identification, while showing why low operational power does not necessarily correspond to low lifecycle carbon or to lower impacts across broader environmental categories.

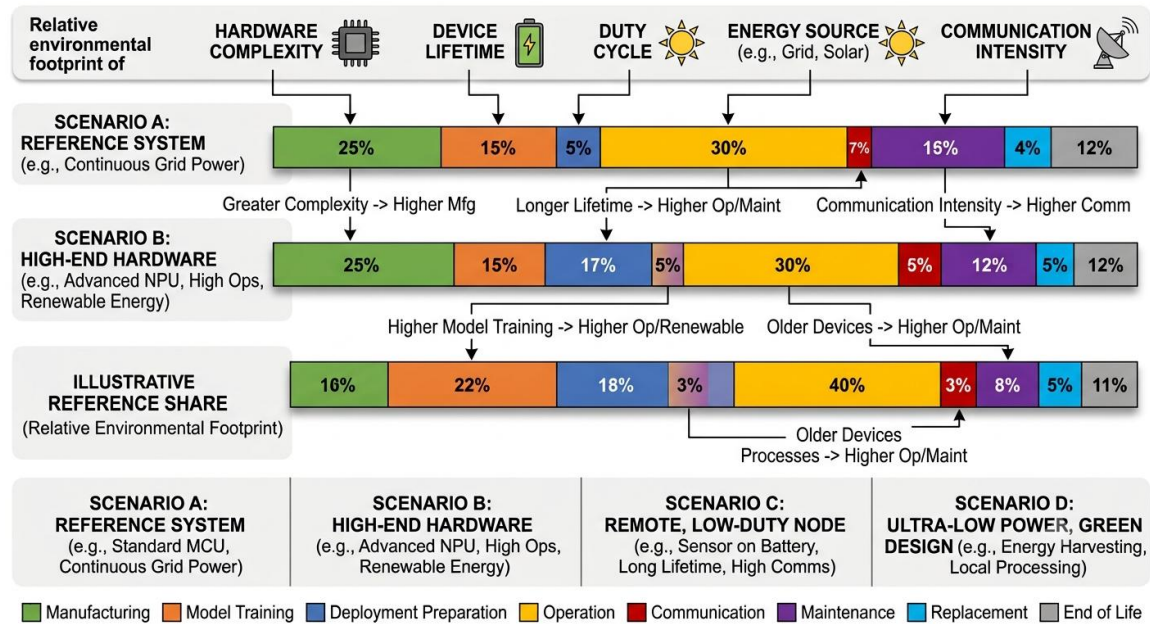


Fig. 2. Comparative analysis of input factor impacts on lifecycle environmental footprints across diverse TinyML usage scenarios [28]–[32].

Stacked bar segments are color-coded to denote relative environmental impact contributions from Manufacturing (green), Model Training (orange), Deployment Preparation (dark blue), Operation (yellow), Communication (red), Maintenance (purple), Replacement (light blue), and End of Life (grey). Abbreviations: Mfg (Manufacturing), Op/Ops (Operation), Renewable (Renewable Energy), Maint (Maintenance), Comm/Comms (Communication), MCU (Microcontroller Unit), and NPU (Neural Processing Unit).

B. Device Lifetime, Maintenance, Firmware Updates and Replacement

1) Device lifetime as a determinant of environmental performance

Device lifetime is one of the strongest variables linking technical design to environmental performance. A TinyML node carries its manufacturing burden from the moment it leaves the production system, regardless of how little electricity it subsequently consumes. A short lived device therefore tends to have a higher environmental burden per year of useful service than an otherwise identical device used for a longer period.

The importance of longevity does not mean that every device should be kept indefinitely. A replacement may be environmentally justified when a new device delivers a large reduction in energy consumption, eliminates a major hazardous material or provides a substantially longer service life. The relevant comparison is therefore between total

lifecycle impacts under competing service strategies rather than between old and new devices in isolation.

2) Battery replacement frequency

Battery replacement can become a hidden environmental driver. A small battery may support a sensor node for several months or years depending on duty cycle, radio usage and environmental conditions. Replacing the battery introduces not only the environmental burden of the battery itself but also transportation, technician time, packaging and potentially the disposal or recycling of the spent battery.

The situation changes for energy harvesting TinyML systems. Battery less architectures can reduce dependency on electrochemical storage and potentially improve long term maintainability. Energy aware TinyML research has demonstrated that local inference decisions can be coordinated with variable energy availability, including harvested energy, to maintain task progress under intermittently powered conditions [34]. This suggests that intelligent scheduling may extend system reliability and reduce unnecessary energy buffering.

Nevertheless, battery elimination is not automatically environmentally superior. Energy harvesting devices may require additional sensing, storage, power conditioning or physical infrastructure. Their overall environmental performance must therefore be evaluated at system level.

3) *Sensor degradation and calibration*

Sensors are often the practical lifetime limiting components of TinyML systems. A processor can remain functional for many years while a sensor experiences drift, contamination, mechanical wear or loss of calibration. This creates an important distinction between hardware lifetime and functional lifetime.

Sensor degradation can cause false detections, reduced model accuracy and increased maintenance frequency. In some systems, calibration itself may require additional energy, communications or physical access. A lifecycle model that counts only processor failure may therefore overestimate practical device longevity.

The environmental consequence is significant. If the whole device is discarded because one sensor becomes unreliable, a relatively minor component failure can trigger disposal of an otherwise usable MCU, memory, PCB and enclosure. Modular sensor architecture can reduce this problem by allowing the degraded component to be replaced without discarding the complete system.

4) *Firmware and model updates*

Firmware and model updates introduce an additional lifecycle stage that is often missing from simplified TinyML footprint calculations. OTA updates require communication energy, processor activity, nonvolatile memory writes and sometimes gateway or cloud processing. The burden is small for one device and one infrequent update, but it can become substantial when updates are repeated across large fleets.

A model update can also trigger a compatibility problem. A new model may require more memory, additional operators or a different accelerator. A device may therefore be technically functional yet unable to support future model versions. This creates a form of software driven obsolescence.

Model training itself may occur on conventional high performance infrastructure rather than on the TinyML node. The environmental burden of training should therefore be assigned to the model lifecycle and amortized across the number of devices and inference events that use the model. This is particularly important when a model is retrained frequently but deployed across a comparatively small fleet.

5) *Model obsolescence versus hardware longevity*

One of the most important lifecycle risks is technological obsolescence. A TinyML node may remain physically functional but become unsuitable because its model, firmware, communication stack, security requirements or data distribution has changed.

This is different from physical failure. A device that stops working has a direct technical reason for replacement. A device that still works but cannot accommodate a new model is replaced because of an incompatibility created by the software and system environment.

TinyML architectures should consequently separate model evolution from hardware replacement wherever possible. Techniques such as quantization, pruning, architecture search and efficient model design can help maintain compatibility

with constrained processors. Quantization and related compression approaches have been shown to reduce model size and computational requirements while supporting edge deployment under strict resource constraints [33].

A sustainability oriented update strategy should therefore ask whether a new model can be made compatible with existing hardware before recommending a new device. This approach transforms software optimization from a purely performance based activity into a lifecycle extension mechanism.

6) *Reliability and environmental trade offs*

Reliability engineering creates an apparent environmental contradiction. More robust hardware can require additional materials, protective packaging, larger enclosures and redundant components. These features may increase embodied impact. At the same time, increased reliability can substantially extend service life and avoid replacement.

The environmental optimum is therefore not necessarily the lightest or simplest device. Instead, the optimal design is the one that provides adequate durability with the smallest additional environmental burden. In practice, ruggedization should be targeted at the failure modes that actually threaten the service lifetime.

The same logic applies to redundancy. Redundant sensors or processors may increase manufacturing impacts but may prevent entire system replacement when one component fails. The environmental value of redundancy therefore depends on whether it extends service life sufficiently to compensate for its additional embodied footprint.

7) *Lifetime extension as a carbon reduction strategy*

Extending device life is often one of the most straightforward lifecycle interventions because it reduces the number of manufacturing events required to deliver a fixed amount of service. Research on electronic products has repeatedly shown that repair and reuse can outperform replacement in many scenarios, although the result depends on repair type, product characteristics and remaining useful life [35].

For TinyML, lifetime extension should involve several coordinated design decisions. Batteries should be replaceable where practical. Sensors should be modular. Firmware should support long term maintenance. Model interfaces should remain stable enough to permit software upgrades without hardware redesign. Components with high failure probability should be accessible for replacement.

Long lifetime, however, should be measured functionally. A ten year old device that produces unreliable sensor data is not equivalent to a ten year old device delivering valid information. The relevant metric is therefore years of reliable service rather than years since manufacture.

Table VIII illustrates how the same TinyML architecture can produce markedly different annualized environmental burdens under different service lifetimes and maintenance schedules. The comparison highlights the importance of battery replacement; firmware maintenance and model update frequency.

TABLE VIII
LIFETIME AND MAINTENANCE SCENARIOS FOR TINYML ENVIRONMENTAL PERFORMANCE

Scenario	Device lifetime	Battery replacement pattern	Model / firmware update frequency	Expected lifecycle interpretation	Sustainability implication	Citation(s)
A	1 year	Frequent	Frequent	Manufacturing burden concentrated over short service period	Poor unless rapid replacement delivers major efficiency gains	[33]
B	3 years	One replacement	Periodic	Embodied burden distributed over moderate service period	Better than short replacement cycle	[34]
C	5 years	One replacement	Infrequent	Manufacturing impact amortized across longer service	Stronger lifecycle performance	[35]
D	10 years	Replaceable battery as needed	Long support window	Hardware remains in service while software evolves	Potentially strong environmental advantage	[33]
E	10 years	Battery less	Energy harvesting and adaptive scheduling	Long service with minimized storage replacement	Potentially very low use stage burden	[34]

TinyML = Tiny Machine Learning; MCU = microcontroller unit. Citation note: 10 year battery less operation refers to operation supported through energy harvesting and adaptive scheduling rather than conventional battery replacement.

The role of service life and energy-storage strategy discussed in Sections III-B1, III-B2, III-B4 and III-B7 is illustrated in Fig. 3, which shows how device service life and

energy-storage strategy influence the cumulative lifecycle carbon footprint over a 10-year operating period. It compares frequent battery replacement, infrequent battery replacement, and batteryless energy harvesting, and includes an annualized footprint breakdown to show how manufacturing burden is distributed over time while use-phase impacts accumulate and maintenance and update requirements contribute to lifecycle performance.

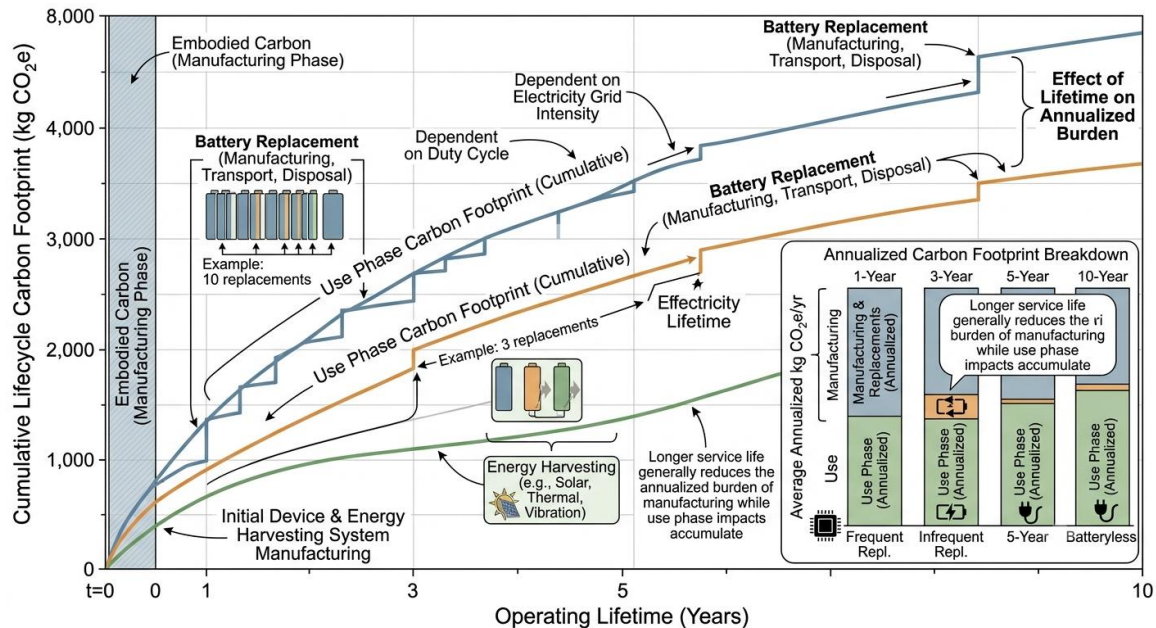


Fig. 3. Effect of Device Lifetime on Cumulative TinyML Carbon Footprint [33]–[35].

The primary graph plots cumulative carbon impact for frequent battery replacement (blue step-curve), infrequent replacement (orange step-curve), and energy harvesting (green curve), supplemented by a bar chart comparing their annualized burdens.

Abbreviations: kg CO₂e (kilograms of carbon dioxide equivalent); kg CO₂e/yr (kilograms of carbon dioxide equivalent per year); Repl. (Replacement); t (time).

C. Large Scale TinyML Deployment and the Cumulative Environmental Footprint

1) From per inference efficiency to fleet level impact

TinyML is attractive partly because the energy demand of an individual inference can be extremely small. However, a very low per device footprint should not be confused with a negligible global footprint. The environmental consequence of TinyML depends on both unit impact and deployment scale.

A useful conceptual relationship is:

Total lifecycle impact = impact per device $\times$ number of devices $\times$ replacement frequency $\times$ service period.

This simple relationship reveals why environmental burdens can remain significant even when individual nodes are exceptionally efficient. A device consuming tiny amounts of energy can still require substantial aggregate quantities of silicon, copper, polymers, batteries and other materials when deployed across millions or billions of locations.

2) Scaling by device count

The scale effect becomes clearer under hypothetical deployment scenarios. At 10^3 devices, manufacturing and operational impacts are generally manageable and can often be treated as a pilot project. At 10^6 devices, the same design represents an industrial scale resource demand. At 10^9 devices, the deployment becomes global materials and waste management issue rather than merely an engineering project.

This scaling principle is consistent with lifecycle analyses of IoT infrastructure, which emphasize the need to account for both the physical hardware and the infrastructure supporting the data flows generated by connected devices. A system cannot be considered environmentally negligible merely because individual endpoints are small [36].

3) Continuous versus event driven deployment

Duty cycle is one of the most influential operational variables. Continuous sensing maintains sensors and processing units in an active state and may require frequent inference. Event driven operation permits long periods of sleep and activates computation only when a relevant trigger occurs.

For many environmental applications, event driven TinyML offers a strong lifecycle advantage because it can reduce both computation and communication. Instead of continuously transmitting sensor measurements, the device can perform local classification and transmit only events of interest.

This advantage becomes particularly meaningful when communication is more energy intensive than local inference. Local decision making can allow the device to exchange significantly less information with gateways or cloud services, thereby shifting the architecture toward computation at the point of data generation.

4) Environmental implications of always on sensing

Always on operation creates a different lifecycle profile. Even if inference energy is minimal, the accumulated energy required for sensor operation, memory access, clocking, radio standby and periodic communication can become significant over several years.

Always on sensing can also increase battery replacement frequency. The environmental consequence is therefore cumulative: higher electricity use may combine with additional battery production and replacement transport.

A meaningful lifecycle study should therefore report at least two operating regimes: event driven and continuous. Reporting only the best case duty cycle risks understating real world impacts.

5) Geographical differences in electricity carbon intensity

Geographic variability is especially important for large scale deployments. A device operating in a region with relatively low carbon electricity will generate a smaller use stage carbon footprint than the same device deployed in a high carbon electricity system.

The same geographical principle applies to manufacturing. Semiconductor fabrication depends on local water availability, electricity generation and industrial infrastructure. Recent research examining semiconductor supply chains demonstrates that water stress can become a material risk because many semiconductor production facilities are located in areas facing current or projected water constraints [39].

This finding is particularly relevant when considering the apparent invisibility of manufacturing impacts in TinyML. The device may consume very little electricity after deployment while its production depends on water intensive semiconductor infrastructure in geographically sensitive regions.

6) Communication infrastructure at scale

Communication is one of the clearest examples of why lifecycle assessment should move beyond the device boundary. Each TinyML node may transmit only a small message, but a very large fleet can create persistent aggregate traffic. The environmental significance depends on message frequency, payload size, radio technology, gateway architecture, network infrastructure and backhaul requirements.

TinyML can potentially reduce this burden by performing local inference and transmitting only decisions or anomalous events. However, communication does not disappear. Firmware updates, model updates, health reports, synchronization, security exchanges and diagnostic information still require network activity.

At fleet scale, communication infrastructure therefore becomes a shared environmental cost. An analysis based only on energy measured at the MCU may underestimate the total environmental burden of delivering the sensing service.

7) Aggregate material demand

The scale of TinyML deployment has a direct materials dimension. Even a very small circuit board contains silicon based semiconductors, copper conductors, solder, polymer materials and other substances. Multiplying these quantities by 10^6 or 10^9 devices creates a meaningful material demand.

The same principle applies to aluminium housings, connectors, batteries and protective materials. The total quantity of materials can become more important environmentally than the tiny energy required by a single inference.

This is why materials efficiency must become a core TinyML design objective. Designers should minimize component count, avoid unnecessary material diversity, select materials that are recoverable and enable the separation of components at end of life.

8) *The paradox of TinyML efficiency*

The paradox of TinyML is that the technology can become environmentally attractive precisely because it is so efficient. Lower energy use, lower hardware cost and local intelligence reduce barriers to deployment. Those same advantages can encourage much larger numbers of connected devices.

The resulting environmental outcome depends on whether the efficiency gains are used to reduce the number of devices and the amount of computation, or whether they primarily enable an unprecedented expansion of sensing infrastructure.

This phenomenon can be understood as a rebound effect. If each device becomes cheaper and more energy efficient, organizations may deploy more devices than before. The unit burden decreases while the total burden increases.

The sustainability objective should consequently move from “minimum energy per inference” to “minimum environmental impact per delivered service.” The service may be a detected leak, a medical alert, an industrial fault prediction or a wildlife observation. The correct comparison is between the lifecycle impact of delivering that service through TinyML and the impact of competing architectures.

Table IX translates unit level TinyML efficiency into fleet level consequences under progressively larger deployment scenarios. The scenarios emphasize that environmental burden scales with device population, communication requirements, replacement rates and infrastructure dependence.

TABLE IX
SCENARIO BASED CUMULATIVE ENVIRONMENTAL FOOTPRINT OF LARGE SCALE TINYML DEPLOYMENT

Deployment scenario	Approximate device population	Dominant environmental drivers	Communication burden	Material and e waste implication	Main sustainability risk	Citation
Small pilot	10^3	Manufacturing and installation	Low	Limited	Poorly optimized hardware choices	[36]
Industrial deployment	10^6	Hardware production, batteries and use	Moderate	Significant	Fleet replacement and maintenance	[37]
Smart city deployment	10^9	Manufacturing, operation and communication infrastructure	High	Very high	Aggregate resource consumption	[38]
Global scale sensing network	$\geq 10^9$	Manufacturing, energy, communications and EoL	Very high	Potentially transformational	Rebound effect and e waste growth	[39]

EoL = end of life; e waste = electronic waste; TinyML = Tiny Machine Learning. Citation note: 10^3 , 10^6 , and 10^9 denote approximately one thousand, one million, and one billion devices, respectively.

Figure 4 extends the discussion of deployment scale developed in Sections III-C1 through III-C8, depicting the scaling relationship between declining carbon intensity per individual TinyML inference and increasing aggregate

environmental burden as device populations expand. The figure follows four deployment stages and highlights how manufacturing, communication, maintenance, end-of-life management, continuous or event-driven operation, electricity carbon intensity, aggregate material demand, and rebound effects can cause system-level environmental burdens to grow even as per-inference efficiency improves.

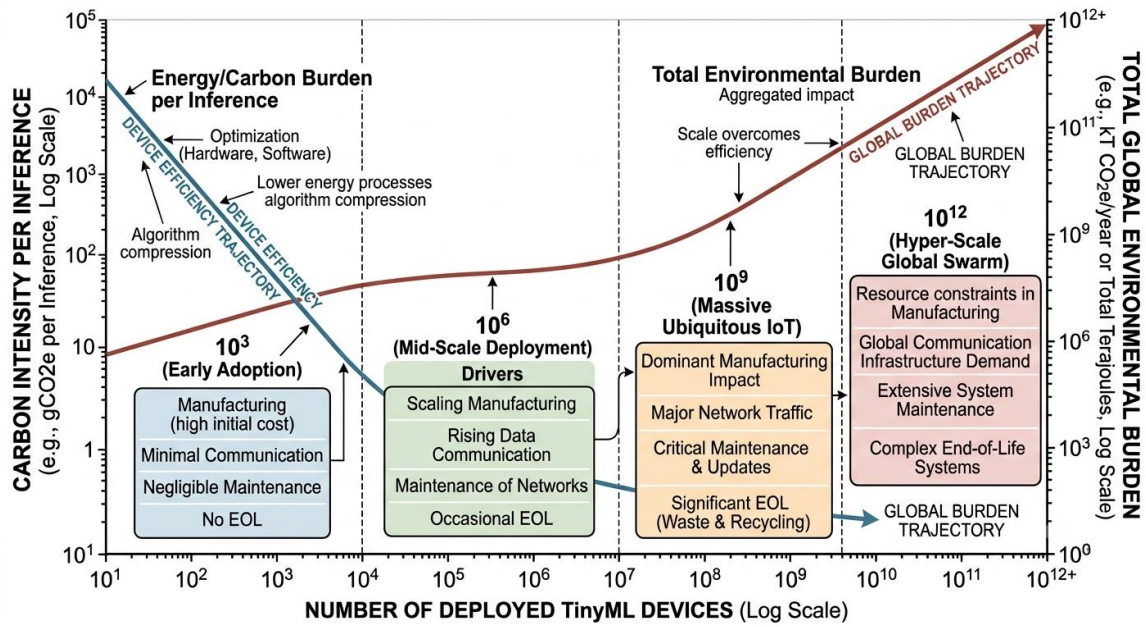


Fig. 4. Scaling Trajectories of TinyML Deployments: Contrasting Declining Inference Intensity Against Escalating Global Environmental Burden Across Four Era Stages [36]–[39].

The figure employs a dual logarithmic axis layout with text annotations to illustrate non-linear scaling dynamics between device-level optimization and global environmental impact across four chronological deployment eras (indicated by colored checklists and vertical delimiters). Blue lines and labels track carbon intensity per inference, while red lines and labels depict the total aggregated global environmental burden.

Abbreviations: TinyML = Tiny Machine Learning; gCO₂e = grams of carbon dioxide equivalent; kT CO₂e/year = kilotonnes of carbon dioxide equivalent per year; IoT = Internet of Things; EOL = End-of-Life.

D. E Waste, Resource Depletion and End of Life Pathways

1) TinyML contribution to WEEE streams

The environmental footprint of TinyML does not terminate when inference stops. End of life determines whether materials are recovered, downcycled, dispersed into the environment or permanently lost. TinyML devices contribute to wider waste electrical and electronic equipment streams through discarded MCUs, sensors, battery packs, PCB assemblies, connectors, cables and enclosures.

The problem is important because miniaturization can create a false perception that a device is too small to matter. A single discarded sensor may have negligible mass, but millions or billions of such devices create a substantial material flow.

Recent environmental assessments of e waste recycling demonstrate that end of life management can produce very different environmental outcomes depending on whether valuable materials are recovered, transported long distances, incinerated or landfilled. Life cycle assessment has therefore become increasingly important for comparing alternative recycling pathways [40].

2) Material composition of TinyML systems

TinyML systems contain materials that differ substantially in their recovery potential. The MCU contains silicon and metallic interconnects. Sensors may contain silicon, ceramics, polymers and metallic films. PCBs contain copper, fiberglass, epoxy resins, solder and surface finishes. Batteries contain electrochemically active materials and metal current collectors.

The diversity of materials complicates recycling because components often require different processing routes. Recovering copper from a PCB is not the same process as recovering battery metals. Likewise, extracting a reusable sensor requires a different pathway from shredding a mixed PCB stream.

This means circular design should begin at the architecture stage rather than being added after disposal. Designers should consider how individual components can be separated, identified and recovered.

3) Hazardous substances and environmental release

Electronic components can contain substances that require controlled management. The risk varies by component, material choice and manufacturing history. Batteries introduce additional safety concerns because damaged or improperly processed energy storage components can create fire or chemical hazards.

Informal treatment can increase exposure because crude dismantling, open burning, uncontrolled leaching and low protection conditions may release metals, particulate matter and hazardous chemical constituents. Recent reviews of e waste management highlight the environmental and human health consequences associated with poorly controlled disposal and recycling practices [40].

TinyML devices should therefore be designed to reduce hazardous material content where technically feasible and to facilitate safe component separation.

4) Informal versus formal recycling pathways

Formal recycling typically provides controlled collection, dismantling, material separation, pollution management and process monitoring. Informal recycling may recover valuable materials at lower cost but can involve unsafe dismantling and uncontrolled emissions.

This distinction is particularly important in regions where e waste collection systems are still developing. Rapid deployment of TinyML infrastructure without corresponding collection and recycling systems can produce a delayed environmental burden. The devices may operate successfully for several years and then accumulate as distributed electronic waste without a reliable recovery pathway.

5) Mechanical, pyro, hydro and bio metallurgical recovery

Mechanical recycling generally begins with dismantling, shredding, screening and physical separation. It can concentrate metallic fractions but may not recover all valuable materials at high purity.

Pyrometallurgical processing uses elevated temperatures to recover metals and separate materials through smelting and related reactions. Hydrometallurgical systems use chemical dissolution and subsequent precipitation, extraction or electrochemical recovery. Emerging biological approaches explore microorganisms or biological reagents for selective metal recovery.

No route is environmentally perfect. Mechanical recovery may leave complex fractions unrecovered. Pyrometallurgy can require significant energy and generate emissions. Hydrometallurgy may involve substantial chemical and wastewater burdens. The appropriate pathway therefore depends on material composition, local electricity mix, recovery rate and transportation distance.

6) Material recovery efficiency and environmental trade offs

Recent LCA studies demonstrate why recovery efficiency and process energy should be assessed simultaneously. In waste PCB recycling, electricity consumption can strongly influence environmental performance, while the recovery of valuable metals can generate environmental credits by avoiding primary metal production [41].

This has a direct TinyML implication. A recycling route should not be considered sustainable simply because it recovers material. The route should be evaluated on the basis of net lifecycle benefit, including collection, transportation, dismantling, electricity, chemicals, recovered material quality and avoided primary production.

7) Battery end of life

Batteries deserve a dedicated pathway. Their environmental performance depends on chemistry, capacity, replacement frequency, collection rate and recycling process. TinyML

products with non-replaceable batteries may become e waste even when the computational electronics remain functional.

Replaceable battery design allows the device housing, PCB, MCU and sensors to remain in service while only the storage component is replaced. This can substantially reduce unnecessary disposal.

8) Disassembly challenges in miniaturized electronics

Miniaturization can improve operational efficiency while making repair and recycling more difficult. Components may be permanently bonded, densely assembled or inaccessible without specialized equipment. Small connectors may be difficult to handle, and adhesive assemblies may discourage component recovery.

These constraints create a design tension. The same compactness that helps reduce material and energy use can interfere with reparability. Sustainable TinyML therefore requires deliberate design choices that preserve access to critical components without unnecessarily enlarging the product.

9) Recycling versus downcycling versus disposal

Recycling maintains materials in circulation, but not all recycling pathways preserve material quality. Some processes produce secondary materials with lower performance or purity. This is commonly described as downcycling.

For TinyML, the preferred pathway should therefore follow a hierarchy. Whole device reuse is usually preferable where the product remains functional. Refurbishment is appropriate when selected components can be repaired or replaced. Component harvesting can recover functioning MCUs, sensors or connectors. Material recycling is appropriate when components cannot be reused but valuable materials remain recoverable. Disposal should be treated as the least preferred option.

10) Regional disparities in end of life infrastructure

End of life outcomes vary greatly between regions. A TinyML design optimized for formal recycling in one country may not perform similarly where collection, sorting and treatment infrastructure is limited.

This is one reason regional LCA is important. Transportation distances, electricity mix, recycling technology, waste collection practices and labour conditions all influence the environmental result. Recent work on e waste recycling has shown that the preferred strategy can depend strongly on local infrastructure and the location of recycling activities [42].

The result is that sustainable TinyML should include geographical considerations during product design. A globally deployed device cannot assume that every unit will eventually enter an identical formal recycling process.

Table X compares preferred end of life pathways for the major physical components of TinyML systems. The table links reuse, refurbishment and recycling potential with material value, hazardous constituents and practical barriers.

TABLE X
END OF LIFE PATHWAYS FOR TINYML COMPONENTS

Component	Reuse potential	Refurbishment potential	Recyclability	Valuable materials	Hazardous constituents	Preferred pathway	Major barrier	Citation(s)
-----------	-----------------	-------------------------	---------------	--------------------	------------------------	-------------------	---------------	-------------

Component	Reuse potential	Refurbishment potential	Recyclability	Valuable materials	Hazardous constituents	Preferred pathway	Major barrier	Citation(s)
MCU / processor	High when functional	Moderate	Moderate to high	Silicon, copper, precious metals	Material dependent	Device or component reuse	Miniaturization and testing	[40]
Sensors	High	High for modular designs	Moderate	Silicon, metals, ceramics	Material dependent	Replacement and component harvesting	Calibration and proprietary packaging	[41]
PCB	Moderate	Moderate	High for metals	Copper, gold, silver, aluminium	Lead and chemical residues in some designs	Material recovery	Mixed materials and thermoset resin	[42]
Battery	Moderate	Limited	High for selected chemistries	Lithium, cobalt, nickel, copper	Electrolytes and reactive materials	Collection and controlled recycling	Collection and fire risk	[43]
Enclosure	High	High	Variable	Aluminium or engineering polymers	Additive dependent	Reuse or material recycling	Mixed polymers and composites	[44]
Connectors and small electronic parts	High	Moderate	High	Copper and precious metals	Material dependent	Component harvesting and metal recovery	Small size and labour cost	[40]

MCU = microcontroller unit; PCB = printed circuit board; material and component categories identify the principal recoverable or potentially hazardous constituents associated with each pathway. Citation note: The table uses component specific terms such as IC where applicable, meaning integrated circuit, and distinguishes recovery pathways according to component condition and material composition.

The major implication for TinyML is therefore that end of life must be treated as a design constraint. A device that is impossible to open, diagnose or separate may possess excellent operational efficiency but still perform poorly from a lifecycle perspective.

The circular economy and end-of-life pathways examined throughout Sections III-D1 to III-D10 are brought together in Fig. 5, which presents a circular economy framework for TinyML devices, linking collection and diagnosis with repair, refurbishment, disassembly, component harvesting, material recycling, reuse, and final disposal pathways. The framework connects the material composition and hazard considerations of TinyML hardware with formal and informal recycling routes, recovery technologies, battery end-of-life management, disassembly constraints, the reuse-versus-downcycling-versus-disposal hierarchy, and regional differences in recovery infrastructure.

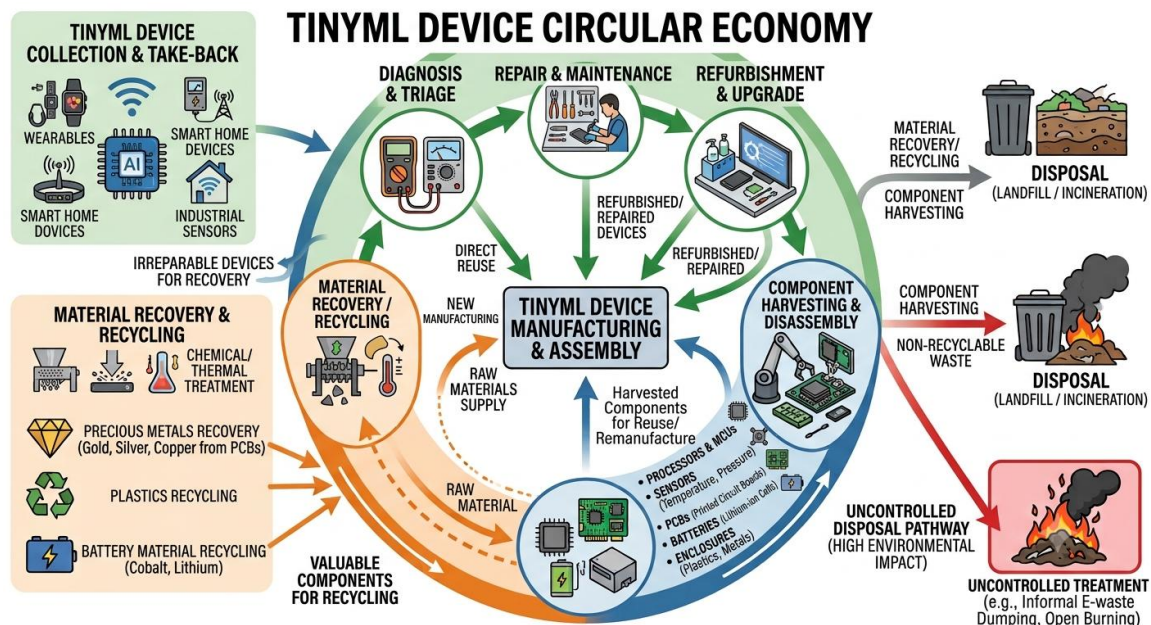


Fig. 5. TinyML Device Circular Economy[40]–[47].

The infographic employs color-coded circular arrows (green for repair/reuse loops, orange for component and material recycling paths, red for disposal streams) and detailed icons to visualize resource recovery options versus waste generation. Abbreviations: AI, Artificial Intelligence; MCUs, Microcontroller Units; PCBs, Printed Circuit Boards; Li-ion, Lithium-ion.

E. Circular TinyML, Design for Sustainability and Future Research Gaps

1) Design for energy efficiency

Energy efficiency remains a core TinyML objective, but it should be viewed as one element of lifecycle sustainability rather than the entire objective. Dynamic voltage and frequency scaling, event driven computation, sparse

processing, hardware acceleration, memory optimization and efficient data movement can reduce operational energy.

Model compression is especially valuable because memory movement can be an important contributor to embedded computing energy. Compact models reduce memory demand and can decrease computation and communication requirements. However, model compression should be evaluated across the entire lifecycle. A highly compressed model that significantly increases training cost or demands additional specialized hardware may not always produce the lowest environmental impact.

The appropriate objective is therefore minimum lifecycle energy rather than minimum inference energy alone.

2) Design for longevity

Longevity should become a first class design target. Durable enclosures, reliable sensors, replaceable batteries and long term firmware support can prevent premature device replacement.

This concept is strongly aligned with the emerging circular electronics literature. Repairable electronic product design is increasingly recognized as a mechanism for reducing obsolescence and environmental burdens because extending product use can avoid the manufacturing of replacement devices [43].

For TinyML, longevity can be achieved through modular architecture. The sensor subsystem should ideally be separable from the processing subsystem. Batteries should be replaceable. Communication modules should be upgradable where future network requirements are uncertain. Firmware should be maintained without requiring complete hardware replacement.

3) Design for repairability

Repairability is particularly important for systems deployed in remote or difficult to access environments. A device that can be diagnosed and repaired locally is less likely to be discarded because of a relatively minor component fault.

Standardized connectors can facilitate replacement. Accessible fasteners can simplify opening. Diagnostic interfaces can reduce troubleshooting time. Modular PCBs can allow a defective sensing module to be exchanged without replacing the entire electronics package.

The environmental advantage is cumulative because each avoided replacement also avoids a new manufacturing event, new transportation and new disposal.

4) Design for recyclability

Design for recyclability requires material separation to be considered before manufacturing. Reducing unnecessary material diversity can simplify sorting. Recoverable metals should be selected where appropriate. Adhesives should not prevent disassembly of critical components. Substrates and housings should be selected with end of life pathways in mind.

A major development is the emergence of recyclable PCB materials that demonstrate that circularity can be designed directly into the electronics substrate. Vitrimers based PCB research has demonstrated repairability together with high recovery of polymer and fibre fractions and measurable cradle to cradle benefits across multiple environmental categories

[45]. This is an important conceptual shift because the PCB has traditionally been treated primarily as a difficult end of life waste material rather than as a component that can be engineered for circularity.

5) Sustainable model design

Model design should balance accuracy, memory requirement, latency, energy, communication burden and embodied footprint. The most accurate model is not necessarily the most sustainable model if the accuracy improvement is small but the computational cost is large.

The same principle applies to model updates. A model that can remain stable for several years may be more environmentally efficient than one that demands frequent retraining and OTA replacement.

TinyML model design should therefore incorporate a lifecycle objective function in which environmental impact appears alongside predictive performance. This would represent a shift from conventional machine learning optimization toward sustainable machine intelligence.

6) Carbon aware TinyML

Carbon aware TinyML requires adaptation to the environmental conditions under which the device operates. Model training can be scheduled according to electricity carbon intensity. Large updates can be delayed when communication infrastructure is carbon intensive. Inference workloads can be adjusted when energy availability changes.

The same principle can be applied at fleet level. Devices located in different electricity regions may benefit from different update schedules or model configurations. Environmental context can therefore become another input to system optimization.

7) Hardware software environment co design

Hardware and software are traditionally optimized for performance, cost and reliability. Environmental constraints now need to become equally explicit.

A software developer may choose model architecture based on accuracy and memory. A hardware designer may select an MCU based on unit cost and computational capability. A sustainability engineer may focus on embodied carbon. These decisions should not occur independently.

A co design framework could optimize model size, processor architecture, sensor selection, memory, communication frequency, battery capacity, service lifetime and recyclability as a connected design space.

8) Environmental datasheets for TinyML systems

TinyML products should eventually have environmental datasheets alongside conventional technical specifications. Such documentation should report energy per inference, average and peak operational power, expected lifetime, embodied carbon, component composition, battery chemistry and replacement interval.

It should also report expected model update frequency, communication dependence and end of life options. A recyclability statement should identify whether the PCB, battery, sensors and processor can be separated and whether recovery pathways exist.

Environmental datasheets would make products easier to compare and would enable procurement decisions based on lifecycle performance rather than purchase price alone.

9) Standardization of lifecycle reporting

The lack of harmonized boundaries is one of the biggest obstacles to comparison. One study may report only device electricity consumption. Another may include PCB manufacturing. A third may include network communication and cloud infrastructure. These studies cannot be compared reliably unless their system boundaries and functional units are clearly documented.

A standardized TinyML environmental reporting framework should define:

- functional unit, such as one year of sensing service or one million valid classifications;
- system boundary from raw material extraction to end of life;
- operational duty cycle;
- electricity carbon intensity;
- communication assumptions;
- battery replacement assumptions;
- model training allocation;
- maintenance and update frequency;
- device lifetime;
- recycling assumptions;
- environmental impact categories.

Recent lifecycle research in circular electronics reinforces the importance of coherent system boundaries and component level data because environmental conclusions can change substantially depending on what stages and materials are included [46].

10) Major unresolved research gaps

The first major gap is the limited availability of primary lifecycle inventories for TinyML specific semiconductor hardware. Existing estimates often rely on broader electronics databases or older semiconductor assumptions. More current MCU, sensor and accelerator inventories are required.

The second gap is field based information about actual TinyML duty cycles. Laboratory measurements can

underestimate practical energy use when devices experience communication retries, environmental variation, firmware maintenance and sensor drift.

The third gap is battery replacement data. Long duration field studies should document actual battery lifetimes under representative workloads rather than relying exclusively on nominal manufacturer specifications.

The fourth gap is the absence of robust fleet scale e waste projections specifically for TinyML. Large deployment scenarios should be modeled using expected replacement rates and regional recycling capacity.

The fifth gap is the lack of experimentally validated circular TinyML hardware. Most sustainability discussions remain focused on energy efficiency while circular electronic materials, repairable assemblies and modular components remain comparatively underdeveloped.

Recent LCA work on waste PCBs demonstrates that recycling performance depends strongly on process energy, recovery efficiency and local conditions. In one recent analysis, electricity reduction was identified as a major opportunity for improving recycling performance, while different metallurgical pathways produced different environmental tradeoffs [46].

New semiconductor packaging research further illustrates the direction required for future TinyML systems. Recent LCA evidence shows that packaging material and geographical location can substantially influence carbon and water scarcity footprints, with copper based packaging offering environmental advantages over some gold based configurations and water scarcity varying markedly by location [47]. Such results demonstrate that environmental optimization must extend below the board level into semiconductor packaging decisions.

Taken together, these gaps show that sustainable TinyML requires a move from isolated energy measurements to experimentally validated lifecycle datasets.

Table XI synthesizes the principal research gaps identified across TinyML lifecycle sustainability and links each gap to the data required, appropriate methodology and expected sustainability outcome. The table emphasizes the need for primary measurements and common reporting boundaries.

TABLE XI
RESEARCH GAPS AND PRIORITY DIRECTIONS FOR SUSTAINABLE TINYML

Research gap	Current evidence	Evidence limitation	Required primary data	Recommended methodology	Expected sustainability outcome	Citation(s)
Repair and product lifetime	Repairability increasingly recognized as important	Limited TinyML specific field data	Failure modes, repair rates and service life	Longitudinal field studies and LCA	Lower replacement burden	[44]
PCB circularity	Recyclable PCB materials experimentally demonstrated	Limited commercial TinyML deployment evidence	Manufacturing, repair and recycling inventories	Cradle to cradle LCA	Improved repairability and material recovery	[45]
Component reuse	IC and PCB recovery increasingly studied	TinyML component sorting data remain limited	Recovery rate, testing yield and reuse quality	Material flow analysis and LCA	Higher component retention	[43]

Research gap	Current evidence	Evidence limitation	Required primary data	Recommended methodology	Expected sustainability outcome	Citation(s)
Battery and PCB recycling	Industrial recycling pathways increasingly assessed	Regional pathway variation remains high	Electricity, chemicals, recovery efficiencies and transport	Regional LCA and sensitivity analysis	Lower resource loss and disposal burden	[46]
Semiconductor environmental footprint	Packaging and semiconductor impacts increasingly quantified	MCU specific inventories remain scarce	Material, energy and water data by process node	Process based semiconductor LCA	Better component and packaging choices	[47]
Standard lifecycle reporting	LCA methodologies are established	Functional units and system boundaries vary	Harmonized inventory datasets	ISO aligned TinyML LCA framework	Comparable sustainability metrics	[46]
Fleet scale sustainability	Large scale IoT impacts recognized	TinyML specific projections remain limited	Device populations, failure rates and update schedules	Scenario analysis and material flow modeling	Avoidance of rebound effects	[47]

PCB = printed circuit board; IC = integrated circuit; LCA = life cycle assessment; MCU = microcontroller unit. Citation note: ISO = International Organization for Standardization; TinyML = Tiny Machine Learning; EoL = end of life.

The circular electronics literature demonstrates that extending product life can be environmentally beneficial when repair or reuse successfully displaces the manufacture of new products[44]. For TinyML, this suggests that design for replacement avoidance may sometimes deliver a larger lifecycle benefit than another incremental reduction in inference energy.

The same principle applies to PCB design. Recyclable vitrimer based PCBs demonstrate that repair and material recovery can be integrated into the electronics substrate itself rather than left entirely to downstream waste management [45]. Such technologies are particularly relevant to TinyML because sensor nodes are expected to be deployed in large numbers and may operate in environments where maintenance and recycling are difficult.

Component recovery is another emerging opportunity. Circular electronics LCA studies show that recovering ICs and other high value components can change the environmental profile of electronic products because reuse avoids some of the impacts associated with manufacturing replacement components [43]. TinyML could benefit from this approach by treating functional MCUs, sensors and communications modules as recoverable assets rather than disposable parts.

At the recycling stage, process selection remains critical. Recent PCB studies indicate that material recovery can generate environmental benefits relative to primary production, but those benefits depend on energy consumption, process route, transport and recovery efficiency [46]. Therefore, TinyML sustainability cannot be guaranteed by simply declaring a device “recyclable.” The actual recovery pathway must be feasible and environmentally preferable.

A further insight concerns semiconductor packaging. The environmental footprint of a processor is not determined solely by its silicon die. Packaging material, bonding methods, electricity consumption and geographical location influence both carbon and water scarcity impacts [47]. This suggests that future TinyML hardware procurement should consider semiconductor packaging as an explicit environmental design variable.

Overall, the evidence supports a transition from the conventional TinyML objective of maximizing inference efficiency toward a broader concept of lifecycle intelligent TinyML. Under this approach, the device is optimized not just to compute efficiently, but to remain useful for a long period, require minimal maintenance, communicate only what is necessary, tolerate model evolution, consume environmentally preferable materials, and enter a credible recovery pathway at end of life.

The most sustainable TinyML system will therefore not necessarily be the device with the smallest MCU or the lowest instantaneous power consumption. It will be the system that delivers the required sensing service with the lowest total environmental burden across manufacturing, training, operation, communication, maintenance, replacement and end of life.

The results also indicate that sustainability should be assessed on a service basis. A TinyML node that performs one million useful classifications over ten years should be compared with alternative technologies that deliver the same one million classifications, rather than compared solely on energy per inference. This service based perspective prevents misleading conclusions in which a highly efficient device is environmentally rewarded despite being frequently replaced or massively over deployed.

The same framework can be extended to different deployment scales. At the prototype level, the key questions concern component selection and measured power. At industrial scale, battery logistics, firmware updates and maintenance become increasingly important. At smart city or global scale, manufacturing capacity, mineral demand, communication infrastructure and e waste management become dominant considerations.

There is therefore a strong case for establishing a TinyML environmental reporting standard that combines engineering measurements with lifecycle assessment. Every published TinyML system should ideally report not only model accuracy, memory footprint and latency, but also energy per inference, standby energy, communication energy, expected lifetime, embodied carbon, component composition and end of life pathway.

Such reporting would also improve scientific reproducibility. Two TinyML models that achieve similar accuracy but have different hardware requirements could be

compared using a common lifecycle framework. Researchers could then identify whether a performance improvement genuinely reduces environmental impact or merely shifts the burden from inference to manufacturing.

Another important research direction is the integration of geographical information into TinyML lifecycle assessment. Semiconductor production, electricity generation, battery supply chains, transportation and recycling are geographically uneven. A model that produces acceptable environmental outcomes in one region may perform differently in another because the electricity mix, water scarcity, transport requirements and recycling infrastructure are different.

Future lifecycle studies should therefore combine device specific inventories with regional environmental data. This would be especially useful for large scale deployments in Africa, Asia and other rapidly expanding IoT markets where electronic waste management infrastructure may not grow at the same rate as device populations.

The cumulative evidence also supports the principle that TinyML should be designed for graceful degradation. Rather than treating every model update as a reason to discard hardware, systems should support smaller fallback models, reduced sensing frequency and local operation when communication becomes unavailable. Such adaptability can extend functional life and reduce replacement pressure.

The distinction between hardware obsolescence and service obsolescence should consequently become an explicit sustainability metric. A device may remain physically capable but become environmentally wasteful if it cannot support the evolving application. Conversely, a flexible hardware platform may remain useful even when the original model is replaced several times.

This leads to a final systems level observation. The environmental impact of TinyML is not located at one point in the architecture. It emerges from interactions among materials, semiconductor manufacturing, sensors, processing, communication, electricity, maintenance, software evolution, human decisions and waste management.

The lifecycle therefore behaves as a chain. Increasing efficiency at one stage can move the environmental burden elsewhere. Reducing communication can increase local computation. Increasing model complexity can reduce false alarms while increasing memory and energy demand. Adding protective materials can increase embodied footprint while extending service life. Designing for recyclability can require a material or assembly choice that slightly increases production energy but produces larger downstream benefits.

The appropriate goal is consequently not optimization of one variable, but identification of the combination of design decisions that minimizes total lifecycle impact while maintaining functional performance.

In this context, circular TinyML represents a natural next step in the evolution of ultra low power artificial intelligence. It combines low energy computation with long life hardware, modular components, repairable assemblies, adaptable models, efficient communication, responsible material selection and credible end of life recovery.

The evidence reviewed in this section supports several central conclusions. First, TinyML can substantially reduce operational energy associated with machine learning inference, particularly when computation is performed locally and communication is minimized. Second, very low inference energy does not eliminate embodied impacts associated with semiconductors, sensors, PCBs and batteries. Third, device lifetime strongly affects the environmental value of TinyML because manufacturing impacts are amortized over useful service. Fourth, fleet scale can transform negligible unit level impacts into substantial cumulative environmental burdens. Fifth, e waste and material recovery must be considered from the initial architecture stage rather than being treated exclusively as an end of life problem. Finally, sustainable TinyML requires hardware, software and environmental co design.

The most important future direction is therefore a shift from energy efficient TinyML to lifecycle efficient TinyML. The former asks how cheaply a model can run. The latter asks how much environmental resource is required to create, operate, maintain, update, reuse and ultimately recover the complete system that delivers the intelligence.

The broader implication is that the sustainability of TinyML will be determined not simply by how small the model becomes, but by how intelligently the entire physical and digital lifecycle is designed.

IV. CONCLUSION

TinyML research and development should adopt a lifecycle based sustainability framework that evaluates environmental impacts from semiconductor and component manufacturing through operation, maintenance, replacement, reuse, recycling, and final disposal. Future TinyML studies should report standardized environmental and engineering metrics, including energy per useful inference, standby energy, communication energy, expected functional lifetime, embodied carbon, material composition, battery replacement frequency, and end of life pathway. Designers should prioritize long lived, modular, repairable, upgradeable, and replaceable hardware so that failure of individual sensors, batteries, or other components does not require premature disposal of the complete device. Model and hardware optimization should consider total lifecycle burden rather than inference energy alone, particularly where increased computational efficiency requires more complex processors, sensors, memory, or manufacturing processes. Communication should be minimized through event driven sensing, local processing, adaptive transmission, and other strategies that reduce unnecessary data exchange while preserving the required system function. Battery selection and energy harvesting strategies should be evaluated using realistic service lifetimes, replacement requirements, environmental conditions, and supporting hardware rather than on nominal energy performance alone. Large scale deployments should incorporate fleet level assessments that account for device numbers, duty cycles, failure rates, maintenance, firmware and model updates, replacement cycles, material demand, and projected electronic waste. End of life management should be

incorporated at the architecture stage through component identification, ease of disassembly, recoverability, reuse, refurbishment, and access to credible recycling pathways rather than relying solely on disposal stage interventions. Future research should prioritize primary, field based lifecycle data for TinyML hardware, especially actual duty cycles, battery performance, sensor degradation, maintenance patterns, semiconductor inventories, and regional recycling conditions. Overall, TinyML should progress from the narrow objective of minimizing computational energy toward lifecycle efficient system design in which environmental performance, functional reliability, resource use, longevity, and material recovery are considered together.

ACKNOWLEDGEMENT

The authors acknowledge the contributions of researchers whose published findings and scientific evidence provided the foundation for this comprehensive review. The authors also appreciate the scholarly resources and literature that supported the integrated assessment of TinyML energy use, environmental impacts, and lifecycle sustainability.

FUNDING STATEMENT

This comprehensive review received no specific financial support, grant, or sponsorship from any funding agency, institution, or commercial organization. The work was conducted using the authors' available academic resources and did not receive external funding.

CONFLICTS OF INTEREST STATEMENT

The authors declare that they have no financial, professional, or personal conflicts of interest that could have influenced the content or conclusions of this review. All interpretations presented in the manuscript were developed independently and solely for scholarly and scientific purposes.

REFERENCES

- [1] P. P. Ray, "A review on TinyML: State-of-the-art and prospects," *J King Saud Univ Comput Inf Sci*, vol. 34, no. 4, pp. 1595–1623, 2022, doi: 10.1016/j.jksuci.2021.11.019.
- [2] A. G. Alanazi, H. A. Alanazi, and N. S. Albalawi, "Cross-Layer Resource Optimization for Ultra-Low-Power TinyML Inference on ARM Cortex-M Microcontrollers," *Electronics*, vol. 15, no. 13, Art. no. 2918, 2026, doi: 10.3390/electronics15132918.
- [3] T. C. Kuo, C. Y. Kuo, and L. W. Chen, "Assessing environmental impacts of nanoscale semi-conductor manufacturing from the life cycle assessment perspective," *Resour Conserv Recycl*, vol. 182, Art. no. 106289, 2022, doi: 10.1016/j.resconrec.2022.106289.
- [4] A. C. Muhoza, E. Bergeret, C. Brdys, and F. Gary, "Power consumption reduction for IoT devices thanks to Edge-AI: Application to human activity recognition," *Internet Things*, vol. 24, Art. no. 100930, 2023, doi: 10.1016/j.iot.2023.100930.
- [5] F. B. Jasper, J. Späthe, M. Baumann, J. F. Peters, J. Ruhland, and M. Weil, "Life cycle assessment (LCA) of a battery home storage system based on primary data," *J Clean Prod*, vol. 366, Art. no. 132899, 2022, doi: 10.1016/j.jclepro.2022.132899.
- [6] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Trends in AI inference energy consumption: Beyond the performance-vs-parameter laws of deep learning," *Sustain Comput Inform Syst*, vol. 38, Art. no. 100857, 2023, doi: 10.1016/j.suscom.2023.100857.
- [7] T. Nan, J. Yang, R. Aromaa-Stubb, Q. Zhu, and M. Lundström, "Process simulation and life cycle assessment of hydrometallurgical recycling routes of waste printed circuit boards," *J Clean Prod*, vol. 435, Art. no. 140458, 2024, doi: 10.1016/j.jclepro.2023.140458.
- [8] L. Dutta and S. Bharali, "TinyML meets IoT: A comprehensive survey," *Internet Things*, vol. 16, Art. no. 100461, 2021, doi: 10.1016/j.iot.2021.100461.
- [9] N. N. Alajlan and D. M. Ibrahim, "TinyML: Enabling of inference deep learning models on ultra-low-power IoT edge devices for AI applications," *Micromachines (Basel)*, vol. 13, no. 6, Art. no. 851, 2022, doi: 10.3390/mi13060851.
- [10] T. Pirson and D. Bol, "Assessing the embodied carbon footprint of IoT edge devices with a bottom-up life-cycle approach," *J Clean Prod*, vol. 322, Art. no. 128966, 2021, doi: 10.1016/j.jclepro.2021.128966.
- [11] L. Shi, K. J. Mach, S. Suh, and A. Brandt, "Functionality-based life cycle assessment framework: An information and communication technologies (ICT) product case study," *J Ind Ecol*, vol. 26, no. 3, pp. 782–800, 2022, doi: 10.1111/jiec.13240.
- [12] Y. Abadade, A. Temouden, H. Bamoumen, N. Benamar, Y. Chtouki, and A. Senhaji Hafid, "A comprehensive survey on TinyML," *IEEE Access*, vol. 11, pp. 96892–96922, 2023, doi: 10.1109/ACCESS.2023.3294111.
- [13] S. Bramante, F. Ferrandino, and A. Cilaro, "Understanding energy efficiency of AI deployments in IoT-driven smart cities," *IoT*, vol. 7, no. 1, Art. no. 27, 2026, doi: 10.3390/iot7010027.
- [14] C. M. Bhushan, P. Koppuravuri, N. Prasanthi, F. Gazi, M. M. Hussain, M. Abdussami, et al., "Deploying TinyML for energy-efficient object detection and communication in low-power edge AI systems," *Sci Rep*, vol. 15, Art. no. 44299, 2025, doi: 10.1038/s41598-025-27818-9.
- [15] T. Malche, S. Budhani, P. K. Soni, and G. M. Upadhyay, "Voice-activated home automation system for IoT edge devices using TinyML," *Discov Internet Things*, vol. 5, Art. no. 68, 2025, doi: 10.1007/s43926-025-00165-x.
- [16] V. Rajapakse, I. Karunanayake, and N. Ahmed, "Intelligence at the extreme edge: A survey on reconfigurable TinyML," *ACM Comput Surv*, vol. 55, no. 13s, Art. no. 282, pp. 1–30, 2023, doi: 10.1145/3583683.
- [17] M. Mnif, S. Sahnoun, Y. Ben Saad, A. Fakhfakh, and O. Kanoun, "Combinative model compression approach for enhancing 1D CNN efficiency for EIT-based hand gesture recognition on IoT edge devices," *Internet Things*, vol. 28, Art. no. 101403, 2024, doi: 10.1016/j.iot.2024.101403.
- [18] A. Sabovic, J. Fontaine, E. De Poorter, and J. Famaey, "Energy-aware tinyML model selection on zero energy devices," *Internet Things*, vol. 30, Art. no. 101488, 2025, doi: 10.1016/j.iot.2025.101488.
- [19] P. Fusco, G. P. Rimoli, A. Guerriero, F. Palmieri, and M. Ficco, "On-device training and pruning for energy saving and continuous learning in resource-constrained MCUs," *Future Gener Comput Syst*, vol. 176, Art. no. 108194, 2026, doi: 10.1016/j.future.2025.108194.
- [20] D. Nadalini, M. Rusci, L. Benini, and F. Conti, "Reduced precision floating-point optimization for deep neural network on-device learning on microcontrollers," *Future Gener Comput Syst*, vol. 149, pp. 212–226, 2023, doi: 10.1016/j.future.2023.07.020.
- [21] C. Ingrao, R. S. Evola, P. Cantore, P. De Bernardi, A. Del Borghi, E. Vesce, et al., "The contribution of sensor-based equipment to life cycle assessment through improvement of data collection in the industry," *Environ Impact Assess Rev*, vol. 88, Art. no. 106569, 2021, doi: 10.1016/j.eiar.2021.106569.
- [22] A. Ouattara, C. C. Lin, J. Y. Thum, C. H. Kuo, L. H. Huang, and A. H. Hu, "Towards cleaner semiconductor packaging: Life cycle assessment of material and location decisions on carbon and water scarcity footprints," *Cleaner Waste Syst*, vol. 13, Art. no. 100467, 2026, doi: 10.1016/j.clwas.2025.100467.
- [23] X. Lu, X. Gong, Z. Nie, W. Chen, Y. Liu, and X. Li, "Life cycle assessment of printed circuit boards recycling from waste mobile phones," *Int J Life Cycle Assess*, vol. 30, pp. 1860–1876, 2025, doi: 10.1007/s11367-025-02466-3.
- [24] F. Hanna, C. Somers, and A. Anctil, "Life cycle assessment of lithium-ion battery recycling: Evaluating the impact of recycling methods and location," *Environ Sci Technol*, vol. 59, no. 28, pp. 14432–14443, 2025, doi: 10.1021/acs.est.4c13838.

- [25] Y. Mehta, R. Xu, B. Lim, J. Wu, and J. Gao, "A review for green energy machine learning and AI services," *Energies*, vol. 16, no. 15, Art. no. 5718, 2023, doi: 10.3390/en16155718.
- [26] Y. Liu, J. R. Riba, and M. Moreno-Eguilaz, "Energy balance of wireless sensor nodes based on Bluetooth Low Energy and thermoelectric energy harvesting," *Sensors (Basel)*, vol. 23, no. 3, Art. no. 1480, 2023, doi: 10.3390/s23031480.
- [27] W. Villegas-Ch, R. Gutierrez, A. Maldonado Navarro, and A. Mera-Navarrete, "Optimizing federated learning on TinyML devices for privacy protection and energy efficiency in IoT networks," *IEEE Access*, vol. 12, pp. 174354–174370, 2024, doi: 10.1109/ACCESS.2024.3503516.
- [28] G. Pietroni, M. Marconi, C. Favi, P. M. Tamborrini, C. Marino, and E. Fiore, "Circular economy for electrical and electronic equipment: A life cycle design and management platform," *Int J Interact Des Manuf*, 2025, doi: 10.1007/s12008-025-02406-3.
- [29] M. Popowicz, N. J. Kater, M. Kettele, J. P. Schöggel, and R. J. Baumgartner, "Digital technologies for life cycle assessment: A review and integrated combination framework," *Int J Life Cycle Assess*, vol. 30, pp. 405–428, 2025, doi: 10.1007/s11367-024-02409-4.
- [30] R. J. Ferreira, C. Ranaweera, K. Lee, and J. G. Schneider, "Energy efficient resource management for real-time IoT applications," *Internet Things*, vol. 30, Art. no. 101515, 2025, doi: 10.1016/j.iot.2025.101515.
- [31] C. N. Coelho Jr., A. Kuusela, S. Li, H. Zhuang, J. Ngadiuba, T. Klæboe Aarrestad, et al., "Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors," *Nat Mach Intell*, vol. 3, pp. 675–686, 2021, doi: 10.1038/s42256-021-00356-5.
- [32] M. Cordella, F. Alfieri, and J. Sanfelix, "Reducing the carbon footprint of ICT products through material efficiency strategies: A life cycle analysis of smartphones," *J Ind Ecol*, vol. 25, pp. 448–464, 2021, doi: 10.1111/jiec.13119.
- [33] P. Nagapurkar and S. Das, "Economic and embodied energy analysis of integrated circuit manufacturing processes," *Sustain Comput Inform Syst*, vol. 35, Art. no. 100771, 2022, doi: 10.1016/j.suscom.2022.100771.
- [34] A. Sabovic, M. Aernouts, D. Subotic, J. Fontaine, E. De Poorter, and J. Famaey, "Towards energy-aware TinyML on battery-less IoT devices," *Internet Things*, vol. 22, Art. no. 100736, 2023, doi: 10.1016/j.iot.2023.100736.
- [35] S. Prakash, M. Stewart, C. R. Banbury, M. Mazumder, P. Warden, B. Plancher, et al., "Is TinyML sustainable? Assessing the environmental impacts of machine learning on microcontrollers," *Commun ACM*, vol. 66, no. 11, pp. 68–77, 2023, doi: 10.1145/3608473.
- [36] Q. Wang, N. Huang, Z. Chen, X. Chen, H. Cai, and Y. Wu, "Environmental data and facts in the semiconductor manufacturing industry: An unexpected high water and energy consumption situation," *Water Cycle*, vol. 4, pp. 47–54, 2023, doi: 10.1016/j.watcyc.2023.01.004.
- [37] M. T. Islam and U. Iyer-Raniga, "Life cycle assessment of e-waste management system in Australia: Case of waste printed circuit board (PCB)," *J Clean Prod*, vol. 418, Art. no. 138082, 2023, doi: 10.1016/j.jclepro.2023.138082.
- [38] M. Ruberti, "Environmental performance and trends of the world's semiconductor foundry industry," *J Ind Ecol*, vol. 28, pp. 1183–1197, 2024, doi: 10.1111/jiec.13529.
- [39] J. Lepawsky, "Climate change induced water stress and future semiconductor supply chain risk," *iScience*, vol. 27, no. 2, Art. no. 108791, 2024, doi: 10.1016/j.isci.2024.108791.
- [40] Y. He, M. Kiehbardroudezhad, H. Hosseinzadeh-Bandbafha, V. K. Gupta, W. Peng, S. S. Lam, et al., "Driving sustainable circular economy in electronics: A comprehensive review on environmental life cycle assessment of e-waste recycling," *Environ Pollut*, vol. 342, Art. no. 123081, 2024, doi: 10.1016/j.envpol.2023.123081.
- [41] Z. Zhang, A. K. Biswal, A. Nandi, K. Frost, J. A. Smith, B. H. Nguyen, et al., "Recyclable vitrimer-based printed circuit boards for sustainable electronics," *Nat Sustain*, vol. 7, pp. 616–627, 2024, doi: 10.1038/s41893-024-01333-7.
- [42] T. Zhang, A. Bainbridge, J. Harwell, S. Zhang, M. Wagih, and J. Kettle, "Life cycle assessment of circular consumer electronics based on IC recycling and emerging PCB assembly materials," *Sci Rep*, vol. 14, Art. no. 29183, 2024, doi: 10.1038/s41598-024-79732-1.
- [43] N. Roskladka, G. Bressanelli, N. Saccani, and G. Miragliotta, "Repairable electronic products for the circular economy: A review of design for repair features, practices and measures to contrast obsolescence," *Discov Sustain*, vol. 6, Art. no. 66, 2025, doi: 10.1007/s43621-024-00753-x.
- [44] M. D. Rao, R. B. Meshram, K. K. Singh, C. A. Morrison, and J. B. Love, "Life cycle analysis on sequential recovery of copper and gold from waste printed circuit boards," *Waste Manag*, vol. 171, pp. 621–627, 2023, doi: 10.1016/j.wasman.2023.10.008.
- [45] X. Wang, W. Huang, B. Yan, S. Zhou, X. Zhu, Z. Wang, et al., "E-waste recycling: Integrated life cycle assessment and techno-economic analysis unravels pyrometallurgy's edge and delivers an optimization framework for recovering waste printed circuit boards," *Waste Manag*, vol. 207, Art. no. 115135, 2025, doi: 10.1016/j.wasman.2025.115135.
- [46] Y. He, H. Hosseinzadeh-Bandbafha, M. Kiehbardroudezhad, W. Peng, M. Tabatabaei, and M. Aghbashlo, "Environmental footprint analysis of gold recycling from electronic waste: A comparative life cycle analysis," *J Clean Prod*, vol. 432, Art. no. 139675, 2023, doi: 10.1016/j.jclepro.2023.139675.
- [47] E. Mullen and M. A. Morris, "Green nanofabrication opportunities in the semiconductor industry: A life cycle perspective," *Nanomaterials (Basel)*, vol. 11, no. 5, Art. no. 1085, 2021, doi: 10.3390/nano11051085.

© 2026 IJFAST — International Journal of Vast Advancements in Science and Technology.

ISSN: 2839-5379 | DOI: 10.68296/IJFAST20260073 | Open Access — CC BY 4.0 License