SAAD AHMED | **RESEARCH STATEMENT**

Batteries are not a sustainable way to power the trillions of devices in the future Internet of Things (IoT). Only 5% of batteries in small electronics are recycled. The remaining ones are discarded and often end up in landfills, releasing toxic fumes and leaching chemicals into the soil as they break down. Mining for raw materials in batteries, like Lithium ("white gold"), has caused water insecurity for Indigenous communities as well as irreparable ecological damage[1]. This collection of concerns has motivated me to explore how to build computing systems with lower ecological impact by leaving batteries behind and running off purely renewable energy harvesting sources.

I build **system support** to enable the **Internet of Batteryless Things**; a vision that liberates the next trillion devices from batteries by powering via ambient energy sources such as solar, RF, and motion. However, such energy sources are erratic, causing frequent interruptions during application execution and requires an application to compute *intermittently*. Existing system support for IoT devices assumes continuous execution and is unable to support intermittent device operation. Therefore, a new system stack needs to be designed to support intermittent device operation while adhering to the following parameters:

Low Overhead– Intermittent computing requires software support, such as checkpointing, to ensure correct resumption after each reboot. This software support must not incur high computational or memory resources to allow more time and memory for program execution.

Small Energy Buffers– IoT devices have tiny form factors due to their ubiquitous nature and can only support small buffers to store energy. Therefore, Intermittent computing support must ensure energy-efficient execution of applications to make the best use of available energy.

Adaptability– With varying energy conditions, applications must be able to adapt program execution to meet user requirements.

Ease of Programming– Any new software support must provide programming abstractions that mitigate the cognitive burden of Intermittent computing (such as unreliable clocks, concurrency, and I/O), thus ensuring ease of programming. The software support must be open-source to enable widespread adoption of sustainable computing practices.

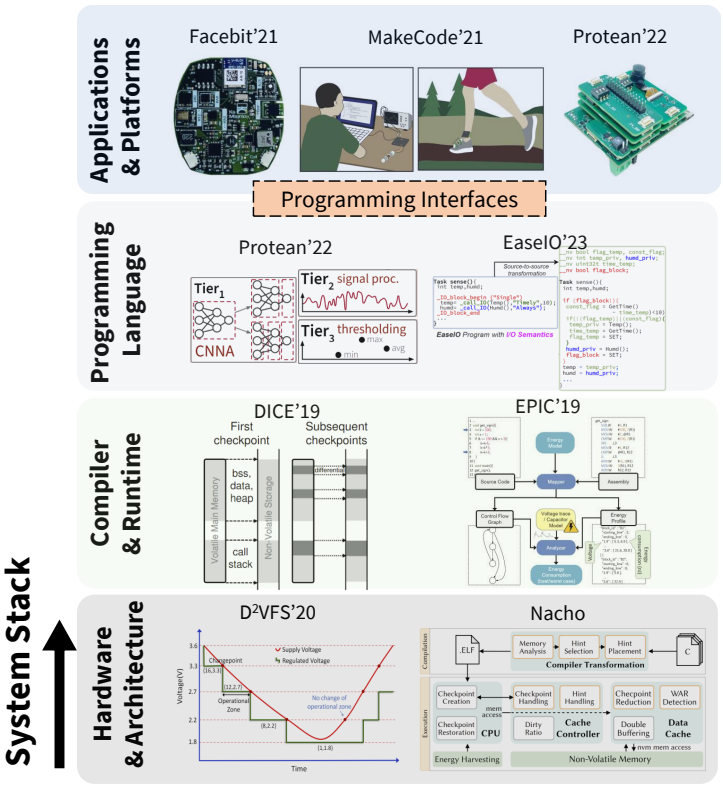

Figure 1: I build **energy-efficient, reliable, and adaptive** batteryless systems and demonstrate them in real-world applications.

During my Ph.D. and postdoctoral research, I designed **programming languages, compiler, runtimes, and hardware** support considering these design parameters. I released these systems as open-source to enable other researchers and practitioners to build innovative applications. I have also collaborated with healthcare workers and a K-8 teacher from a Native Hawaiian School to inform and validate the design of batteryless system support for real-world applications that a battery would not aid. My work has appeared in top computing journals and conferences, including **ACM IMWUT, ACM SenSys, ACM IPSN, ACM SIGOPS EuroSys, ACM TECS** and **ACM TCPS**, many others. As a postdoctoral researcher, I have also solo advised/mentored multiple Ph.D. students who published in these venues. My work was nominated for the best paper award (EWSN), invited to the Communications of ACM, GetMobile magazine for the SIGMOBILE research highlight, and covered by top media outlets such ACM Tech News, Forbes, Tech Crunch, Gizmodo, and many more. Based on my work, I was selected for the prestigious Alexandar von Humboldt Postdoctoral Fellowship.

Enabling The Internet of Batteryless Things

To power the next trillion IoT devices, removing batteries is essential to reduce, if not eliminate, the environmental and societal impact of IoT growth. Additionally, it will enable IoT deployments in far-to-reach areas

[1] https://www.washingtonpost.com/graphics/business/batteries/tossed-aside-in-the-lithium-rush/

without requiring any maintenance, thus unlocking unique applications. Ambient energy, however, has spatiotemporal variability and can interrupt program execution at any point thus giving rise to various challenges.

First, an IoT device loses its computational state after each power failure. Therefore, a device needs to **checkpoint** its **state** onto non-volatile memory to maintain forward progress across power failures. Second, IoT devices are equipped with peripherals to collect data, interact with the environment, or communicate the result to the user. Ensuring **intermittence-safe peripheral operations** requires dedicated system support as IoT devices have different re-execution semantics than the computing unit. Third, the time distance between two energy cycles can be seconds, minutes, or even hours, thus affecting the responsiveness of the application. Therefore, it is important to **adapt** program execution according to changes in the incoming energy to meet user requirements.

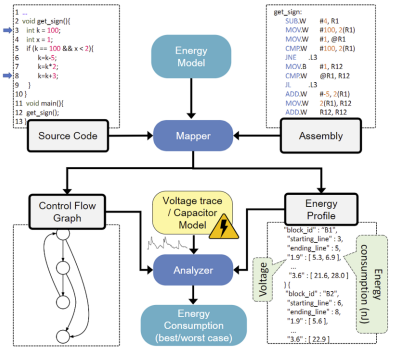

Figure 2: EPIC [3] estimates energy consumption for all program paths in an application .

With highly variable incoming energy, it is hard for the programmer to write appropriate code to address all of these challenges at the time of application development. Concomitantly, it can cause code bloating that can lead to energy waste, which is a scarce resource. My work addressed these challenges by **1)** designing energy-efficient compiler and runtime support to checkpoint application state **2)** enabling programming interfaces to capture re-execution semantics for peripheral operations and **3)** providing a multi-tier programming model and hardware-software co-design to adapt application execution with changing energy conditions.

Energy-efficient State Retention

Deciding *when* and *what* to checkpoint plays a crucial role in determining the energy efficiency of a system. However, it is entirely dependent on the program point where the interruption occurs, which is difficult for the programmer to predict at the time of development.

To decide *when* to checkpoint, EPIC [3] performs energy estimation of all program paths in an application at **compile-time** to predict program points where a power failure can occur (see Figure 2). It inserts a checkpoint call at the program point where the energy consumption of program path exceeds the available budget. Unlike existing works, EPIC considers each basic block in its estimation as well as accounts for the variation in clock frequency and current consumption of the MCU. Results show that the existing work reported eight warnings of non-termination bugs in an activity recognition application, for which EPIC reported none yet gave the correct output.

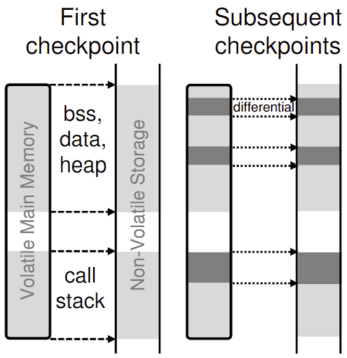

Figure 3: DICE [2] finds checkpoint differentials by tracking changes in the memory.

DICE [2] decides *what* to checkpoint by employing a **compiler** to track changes in the application state at **runtime**. In this way, only those memory addresses are copied that are modified from the previous checkpoint onto the non-volatile memory (NVM), as shown in Figure 3, thus bringing a significant reduction in the number of NVM writes. To lower the memory and computational overhead, DICE maps the entire memory onto a bitmap where each bit represents a byte in the memory. Results show that, with DICE, a device now requires a one-eighth smaller energy buffer to complete the same workload. An application requires 97% fewer checkpoints and one order of magnitude shorter completion time for a given workload with DICE, increasing the system's responsiveness.

Intermittence-safe Peripheral Operations

Peripherals enable IoT devices to sense, actuate, and send data while operating in the wild. However, integrating such peripherals with the computing unit and operating them on ambient energy requires deeper understanding of intermittent computing challenges and hampers rapid prototyping and application development. Protean [6] provides a plug-and-play **hardware platform**, SuperSensor (see Figure 5), that supports multiple harvesters, sensors, and MCUs with the help of novel interconnects. Equipped with a library to interact with peripherals, Protean makes intermittent computing challenges oblivious to the programmer, thus allowing them to focus only on application development. However, special care needs to be taken when developing

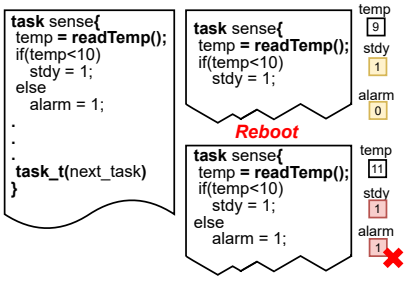

Figure 4: Unsafe program execution due to intermittent peripheral operation

intermittence-safe libraries for peripherals. Peripheral operations have different re-execution semantics than the computing unit and their state needs to be restored to make them consistent with the computational state, otherwise it can result in unsafe program execution, as shown in Figure 4. EaseIO [5] is a compiler front-end that utilizes the LLVM and Clang LibTooling frameworks to provide **programming interfaces** that allow the programmer to express such semantics, thus enabling the system to avoid redundant re-execution of such peripheral operations [7]. Results show that by allowing the programmer to express such semantics, we can reduce wasted work by up to 68% and total execution time by up to 44% by avoiding 76% redundant I/O operations when compared with existing state-of-the-art systems.

Adpative and Fast Application Execution

As energy is a scarce resource, it is important to make the best use of available energy by executing the application faster and adapting to the changing energy conditions to ensure the responsiveness of the application. D^2VFS [4] is a **runtime** technique that employs **hardware-software** co-design to maximize computations in a given energy cycle by dynamically reconfiguring the most energy-efficient voltage and frequency value. Selection of clock speed and undervolting of MCU in D^2VFS increases the number of clock cycles available in an energy cycle by 40-900% compared to different static frequency configurations. D^2VFS reduces the peak energy demand, enabling a reduction of up to one-sixth in the size of the energy buffers necessary for completing a given workload, cutting charging times and enabling smaller device footprints. However, it is not always possible to complete application execution within the available energy budget. Chameleon [6] solves the problem by providing a multi-tier **programming model** for intermittent computing applications, with each tier designed to fulfill user requirements based on the incoming energy. It uses SuperSensor's onboard energy measurement and adaptive reconfigurable energy storage unit to dynamically switch between tiers to match user requirements.

Figure 5: Protean [6] is capable of supporting heterogeneous peripheral devices and supports adaptive multi-tier execution of batteryless applications.

Real-world Deployments

The ultimate goal for IoT devices is to be deployed in real-world scenarios. I have worked with domain experts in mobile health and education to deploy batteryless devices in the real world in an effort to materialize my vision of the Internet of Batteryless Things.

Mobile Health: Frequent replacement of batteries in personal protective equipment (PPE) such as face masks can be burdensome for the healthcare professionals. I worked with clinicians to understand their typical masking behavior and to know more about the features they would find helpful in a facemask. Using the information, we designed a smart facemask platform [7] (see Figure 6); a platform that can power itself from the breath of the wearer and can be attached to the mask to keep a check on the mask fit as well as other body vitals such as heart rate, blood pressure, and respiration rate. The platforms ensure protection of health professionals for hours without requiring any intervention.

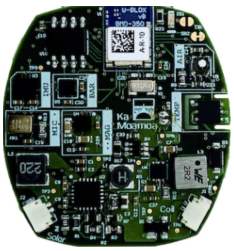

Figure 6: Smart Facemasks Platform [7]

Education: Sustainability is part of Hawaii's culture, and coding will be part of the curricula by 2025. I collaborated with a STEM Teacher from a Native Hawaiian public school responsible for the STEM curricula of K-8 students. The collaboration guided the design of our battery-free plugin [8] for Microsoft MakeCode which helped students learn coding skills while being able to deploy their projects in the wild. Students were able to deploy their projects in a culturally important ancient fishpond. In this way, the plugin educated potential future technologists about sustainable and green computing practices by connecting with their cultural values.

Future Agenda

My Ph.D. and postdoctoral research work has mainly focused on energy-efficient and intermittence-safe operation of a single batteryless device and explored novel applications enabled by it. With the advancements in machine learning techniques, my long-term goal is to enhance the system support capabilities to **enable distributed computing for batteryless IoT deployment**. Allowing multiple devices to collaborate increases computational and energy resources for an IoT deployment, thus enabling them to perform complex tasks. I intend to achieve

the above-mentioned goal by working on the following thrusts in the future.

Thrust 1: Programming & Memory Models I plan to explore new programming models to ease the development of distributed applications on batteryless devices. I have proposed programming interfaces for intermittence-safe peripheral operations [5]. I want to continue the work for energy-efficiency and ease of programming by developing appropriate compiler and runtime tools. Additionally, I would like to explore caches with non-volatile main memories to reduce the checkpointing burden while ensuring faster execution [9].

Thrust 2: Energy-efficient On-device Inference Currently, an IoT device relies on nearby edge nodes to perform machine learning inference, increasing the response time of a device and hampering its performance in a distributed setting. The workload of a single device is inevitably going to increase with modern machine learning applications. Therefore, I would like to explore novel computing architectures and hardware designs to support on-device inference. I have started exploring this avenue with my work, Protean [6], which employs an accelerator to speedup computation for machine learning applications. However, more investigation is required to reduce energy-consumption as well as carbon footprint of such architectures.

Thrust 3: Intermittence-safe Network Layer Reliable communication is essential in enabling collaboration among a distributed set of nodes. I will explore protocols and techniques to ensure reliable communication under an intermittent energy supply to reduce the burden of resynchronization. Once materialized, the output of this thrust would enable federated machine-learning applications on batteryless devices.

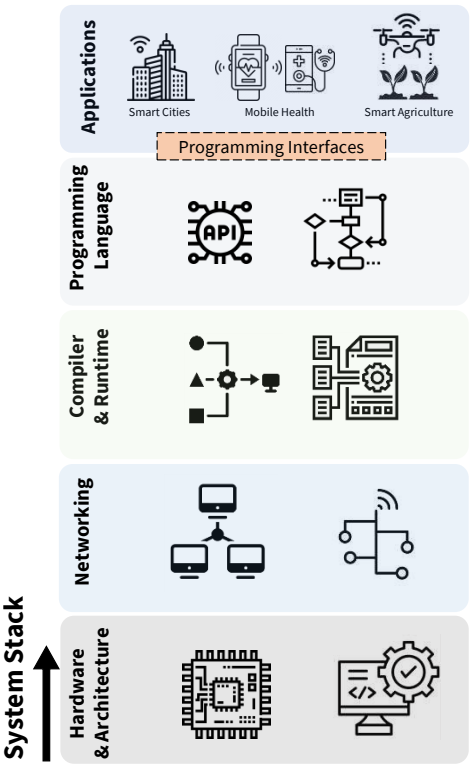

Figure 7: Building distributed batteryless systems requires exploring new programming languages, compiler techniques, reliable networking, and novel architecture to materialize unique applications.

Thrust 4: Applications System support developed in the thrusts mentioned above will unlock a wide range of applications in areas such as mobile health, smart cities, and smart agriculture, inspiring multidisciplinary collaboration. Similar to the real-world application in mobile health [7] and education [8], I intend to work with domain experts from different fields, e.g., medicine and agriculture to materialize novel applications of distributed batteryless systems.

Sustainable Computing Funding My future agenda aligns with the National Science Foundation's (NSF) vision of "Design for Sustainability in Computing," as mentioned in their dear colleague letter (DCL). The letter encourages proposals in the Computer and Information Science and Engineering (CISE) core program on the topics of sustainability-aware system support, such as programming languages, hardware-software optimizations, novel computer architecture, and intermittent computing in general. Furthermore, NSF cyber-physical systems (CPS) and National Institutes for Health (NIH) are other potential funding sources for my future projects as I intend to collaborate with environmental scientists and healthcare professionals to apply developed system support to enable real-world batteryless applications.

References

[1] **S. Ahmed** et al., The Internet of Batteryless Things, (To appear in **Communications of ACM**)
[2] **S. Ahmed** et al., Efficient intermittent computing with differential checkpointing, **(LCTES'19)**
[3] **S. Ahmed** et al., The betrayal of constant power× time..., **(LCTES'19)**
[4] **S. Ahmed** et al., Intermittent Computing with Dynamic Voltage and Frequency Scaling, **(EWSN'20)** Best Paper Nominee
[5] Eren Yildiz, **Saad Ahmed** et al., Efficient and Safe I/O Operations for Intermittent Systems **(EuroSys'23)**
[6] A. Bakar, R. Goel, J.D. Winkel, J. Huang, **S. Ahmed** et al., Protean: Adaptive Battery-free Computing Platform, **(SenSys'22)** ACM SIGMOBILE Research highlight
[7] A. Curtiss, B. Rothrock, A. Bakar, N. Arora, J. Huang, Z. Englehardt, A. Empedrado, C. Wang, **S. Ahmed** et al., FaceBit: Smart Facemask Platform, **(UbiComp'22/PACM IMWUT)** Fast Company's Innovation by Design Award Finalist
[8] C. Kraemer, A. Guo, **S. Ahmed**, J. Hester, Battery-free MakeCode: Accessible Programming for Intermittent Computing, **(UbiComp'22/PACM IMWUT)**
[9] V. Kortbeek, S. Mohapatra, **S. Ahmed**, P. Pawełczak, Data Cache for Intermittent Computing Systems with Non-Volatile Main Memory, In progress