Modeling Periodic Energy-Harvesting Computing Systems

Fatemeh Ghasemi and Magnus Jahre

Abstract—The Internet of Things (IoT) requires Ultra-Low Power (ULP) systems that communicate wirelessly and solely rely on harvested energy to scalably interact with the environment. This is challenging for IoT developers because (i) energy and performance are fundamentally intertwined — since capturing sensor samples and communicating more frequently increases energy consumption — and (ii) the performance versus energy trade-off typically needs to evaluated before ULP-platform selection — as the developer needs to be sure that a sufficiently performant system can be built before incurring the (substantial) effort of implementing the application on the ULP-platform. In this paper, we present the Periodic Energy-Harvesting Systems (PES) model which enables such trade-offs by faithfully modeling the energy impact of changing sampling and communication rates. Across our IoT applications, PES has an average energy prediction error of only 0.5%. In contrast, the average error of the state-of-the-art EH-model is 77.0%.

Index Terms—Energy-harvesting, Internet of Things, ultra-low power (ULP) platforms, energy modeling

1 Introduction

THE Internet of Things (IoT) requires highly efficient Ultra-Low Power (ULP) systems to enable interacting with the environment at scale. These ULP systems must communicate wirelessly and rely solely on energy harvesting — as manually retrieving 100s or 1000s of systems to capture data or change batteries is costly and impractical. Moreover, relying on wired power or wired communication would significantly constrain usability and increase costs. Ideally, such systems should be energy-neutral [1], i.e., that all of the energy supplied by the energy harvester is used to maximize application performance.

IoT applications are (typically) sensor-focused and periodically retrieve sample(s) from one or more sensor(s), process the sample (s), and then communicate the sample(s) to a mains-powered backend system. The key performance metrics of IoT applications are hence the *sampling frequency* — as it determines the temporal resolution of the captured data — and the *communication frequency* — which determines how often samples are sent to the back-end system. Sampling and communicating more frequently increases energy consumption and meeting this demand requires physically larger and more costly energy harvesters. IoT developers hence need to balance the sampling and communication frequencies (to satisfy performance constraints) against the capabilities of the energy harvester (to satisfy cost and physical size constraints). Unfortunately, the ULP-platform needs to be chosen early in the development process because a significant fraction of the development effort depends on the particularities of the selected platform.

Accurate energy models can address this challenge by enabling early-stage analysis of performance versus energy consumption trade-offs. Ideally, the ULP-system is energy-neutral, i.e., it is positioned at a design point where the system's average power consumption is equal to the average power output of the energy harvester. The benefit of energy-neutral design points is that they maximize sampling and communication frequencies subject to the energy provided by the harvester. Moreover, the platform needs to include sufficient energy storage to account for variability in energy supply and demand over short and long time horizons. (We focus on systems with rechargeable batteries in this work.) An energy-neutral system will hence sustain the desired sampling and communication rates whereas a system that over-consumes energy will be intermittent, i.e., sample and communicate at the desired rates in bursts before running out of energy and becoming inactive.

Fig. 1 illustrates the ideal energy-neutral design point. Initially, the system has a certain amount of stored energy (see ①). It then communicates (labeled C) which incurs a relatively high power consumption since both the processor and the communication subsystem are active. The amount of stored energy hence decreases rapidly and reaches a low point at ②. The system is then idle and enters a low-power sleep mode which results in the output of the energy harvester exceeding the system's power consumption and the system accumulating energy (see ③). At ④, the system retrieves a sample from the sensor, processes it, and stores it in platform-local memory (labeled S). The power consumption during sampling also exceeds the output of the harvester, and the amount of stored energy decreases while sampling. The system is then idle before capturing another sample. At this point, the system contains as much energy as it did before the previous communication event (i.e., the energy-level at ① and ⑤ are equal).

Our Periodic Energy-Harvesting Systems (PES) model enables IoT developers to select sustainable near-energy-neutral design points by faithfully modeling the energy consumption of computation and communication. In contrast, the state-of-the-art EH-model [2], [3] only models computation and backup; backups have a high energy cost in communicating systems because the connection to the back-end system needs to be re-established when power returns. Across our IoT applications, PES predicts energy consumption with an average error of 0.5% (maximally 1.4%) relative to our real-hardware baseline and simulator for energy-feasible and intermittent design points, respectively. This is a substantial improvement over the EH-model's 77.0% average error.

2 The PES Model

IoT Application Model. IoT applications are typically sensor-focused and either periodic or reactive. In a periodic system, sampling and communication occurs at predefined intervals whereas, in reactive systems, sampling is regular but communication occurs when the sampled value meets some criteria (e.g., temperature above a predefined threshold). As prior work, we model reactive applications by using the minimum expected arrival rate as the period [4]. IoT applications are hence fundamentally characterized by how often they collect samples whereas communication happens in response to one or more samples being collected. The communication rate hence depends on the sampling frequency, and we use the parameter *communication incidence* to denote the (typical) number of samples collected between each communication event.

We now describe our application model more formally. The application is characterized by (i) the sampling frequency f_s; (ii) the communication incidence n_s; (iii) the active time t_a which is the average processing time per sample; and (iv) the sample size d_s which is the number of bytes of sample data that will be communicated on average for each sample event. The sample size is not necessarily equal to the number of bytes in each sample as data may be aggregated or filtered before transmission. These parameters

• *The authors are with the Department of Computer Science, Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway. E-mail: {fatemeh.ghasemi, magnus.jahre}@ntnu.no.*

Manuscript received 30 Aug. 2021; accepted 26 Sept. 2021. Date of publication 1 Oct. 2021; date of current version 21 Oct. 2021.
This work was supported in part by the BAMPAM project which is funded by the Research Council of Norway under Grant 286596.
(Corresponding author: Fatemeh Ghasemi.)
Digital Object Identifier no. 10.1109/LCA.2021.3117031

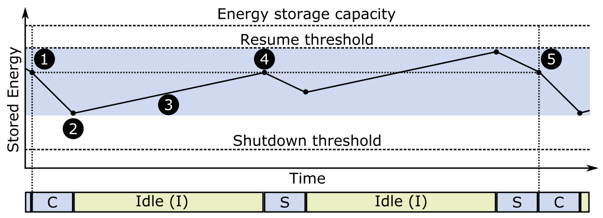

Fig. 1. IoT application example. *The energy-neutral design point is ideal as it achieves the maximum sustainable sampling and communication rates subject to the energy provided by the energy harvester.*

define the frequency and duration of the sampling and communication events, and the system is idle between events. Without loss of generality, we assume that the system immediately proceeds with communicating the sampled data when the last sample is captured.

PES Overview. PES enumerates the states the ULP-system can be in and aggregates the per-state energy consumption to predict overall energy consumption. We use E to denote aggregate energy and a subscript to relate energy consumption to system states. For instance, E_i is the energy consumed while the ULP-system is idle. PES takes a collection of system-specific energy and power consumption constants as input. For instance, e_t captures the energy cost of transmitting a data packet. The aggregate energy consumption E_x of type x is (i) the product of the basic energy cost of the operation e_x and the expected number of operations n_x, or (ii) the product of the average power consumption of the ULP-platform while performing the operation P_x and the time it takes to perform the operation t_x.

IoT systems typically operate indefinitely, and we hence need to consider a specific amount of work when exploring system configurations. More specifically, we predict the energy consumption over a fixed number of samples, i.e., the Region of Interest (RoI). Without loss of generality, we choose a number of samples that are a multiple of the communication incidence to simplify the model. Equation (1) explains how PES predicts E_{RoI}, the overall energy consumption across the RoI:

$$E_{\text{RoI}} = n_c \times [E_t + E_k + n_s \times E_s] + e_c + E_i + E_b. \tag{1}$$

More informally, the application collects sensor data at regular intervals at an energy cost E_s per sample. The communication incidence n_s specifies the number of samples per communication event, and n_c is the number of communication events in the RoI. Hence, we analyze the ULP-system over $n_c \times n_s$ samples. The energy cost of transmitting data in a single communication event is E_t. Since establishing a connection costs non-negligible energy, the application connects when it boots (at an energy cost e_c) and then maintains the connection across the RoI which incurs an energy cost E_k per communication event. The energy spent while idle and in a low-power sleep mode is E_i. If the system is forced to power down due to insufficient energy, it incurs an energy overhead E_b which is due to disconnecting and backing up application state in a non-volatile memory before powering down and then reconnecting and restoring application state when sufficient energy is available.

The minimum energy supplied to the ULP-system $E_{\text{h-min}}$ is the average power output of the energy harvester P_h times the minimum time the ULP-system can use to complete the RoI t_{RoI} (i.e., the sampling period times the number of sampling events):

$$E_{\text{h-min}} = P_h \times t_{\text{RoI}} = P_h \times (1/f_s) \times n_s \times n_c. \tag{2}$$

Similarly, $E_{\text{RoI-min}}$ is the minimum amount of energy the ULP-platform consumes across the RoI, i.e., the output of Equation (1) when no energy is used on backup and restore (i.e., E_b equals 0). The ULP-system is energy-neutral when the harvested energy equals the consumed energy (i.e., $E_{\text{h-min}}$ equals $E_{\text{RoI-min}}$). If the harvested energy is greater than or equal to the consumed energy (i.e., $E_{\text{h-min}} \geq E_{\text{RoI-min}}$), the system is energy-feasible. Conversely, the system is intermittent if it requires more energy than the harvester supplies (i.e., $E_{\text{h-min}} < E_{\text{RoI-min}}$).

Compute. PES predicts the energy cost of capturing a single sample E_s by multiplying the average power consumption of the ULP-system while computing each system or application task by the average time it uses to process the task:

$$E_s = P_{\text{sys}} \times t_{\text{sys}} + \sum_{i=1}^{T} \mathbb{P}(i) \times P_s^i \times t_s^i. \tag{3}$$

While many IoT applications treat all samples similarly and hence have a single task type, it is also common to preform pre-processing to filter out clearly non-interesting samples. For such applications, we identify T task types that are well-characterized by their average runtime and collect the average runtime and average power consumption of each task type i (i.e., t_s^i and P_s^i, respectively) as well as the probability of each task type being executed for a sample within the RoI (i.e., $\mathbb{P}(i)$). The ULP-platform also executes system tasks for t_{sys} seconds at an average power consumption P_{sys} in each sampling period. We opt for measuring the power consumption of each task as the variability can be significant (i.e., from 6.9 mW to 22.9 mW).

Communication. In a periodic IoT application, the amount of data to be transferred in a communication event is a function of the average number of bytes captured in each sampling event (i.e., d_s). Moreover, the protocol typically specifies the number of bytes that can be transferred in a single packet, i.e., the payload d_p. Hence, the energy cost of transferring data is:

$$E_{\text{t}} = n_p \times e_t = \lceil (n_s \times d_s)/d_p \rceil \times e_t. \tag{4}$$

Equation (4) multiplies the number of packets n_p required to transfer the n_s samples captured within a communication event by the average energy cost of transferring a packet e_t. The amount of data that needs to be transferred is hence n_s times the amount of data communicated for each processing event d_s. Since packet payload is fixed, every packet contains d_p bytes. The last packet is hence not full unless the transmitted number of bytes is a multiple of the payload size.

The ULP-system needs to transmit packets at regular intervals to keep the connection alive. More specifically, it needs to transmit at least one packet within each connection interval t_{ci}. The ULP-system hence needs to send empty packets, which we call keep-alive packets, during connection intervals where the application does not have data packets to transmit. To predict the energy cost of the keep-alive packets, we compute the number of keep-alive packets and multiply them by their energy cost e_k:

$$E_k = \left(\frac{n_s}{f_s \times t_{\text{ci}}} - \left\lceil \frac{n_p}{n_{\text{pci}}} \right\rceil \right) \times e_k. \tag{5}$$

We first compute the duration of a communication period by multiplying the inverse of the sampling frequency (i.e., $1/f_s$) by the communication incidence n_s. Then, we divide by the connection interval t_{ci} to compute the maximum number of keep-alive packets. We then subtract the number of connection intervals where we will send data packets, which is the total number of data packets n_p divided by the average number of data packets transmitted within a connection interval n_{pci}. Finally, we multiply the predicted number of keep-alive packets by their average energy cost e_k to predict the energy used to keep the connection alive across a communication period E_k. We empirically determined that the average energy cost of a keep-alive packet e_k is lower than the energy cost of a data packet e_{t} (i.e., 14.7 μJ versus 21.8 μJ per packet), presumably because keep-alive packets do not carry a payload.

Backup and Restore. Although we argue that periodic IoT applications should strive to be energy-feasible, the variability of energy harvester output means that it can be costly to avoid running out of energy altogether. We hence proceed to model the energy cost of backup and restore.

We predict the expected number of backups n_b by first computing the minimum average power consumption across the RoI $P_{\text{RoI-min}}$ which is the minimum energy required to execute the RoI divided by the minimum time the ULP-system uses to execute the RoI (i.e., $P_{\text{RoI-min}} = E_{\text{RoI-min}}/t_{\text{RoI}}$). As mentioned before, t_{RoI} is the product of the number of samples in the RoI and the sampling period (i.e., $t_{\text{RoI}} = 1/f_s \times n_s \times n_c$). The ULP-system restarts application execution with e_{resume} Joule. It then over-consumes energy at a rate of $P_h - P_{\text{RoI-min}}$ Watt on average until it reaches the shutdown energy threshold e_{shutdown}. The expected number of backups n_b is hence:

$$n_b = \frac{t_{\text{RoI}}}{t_o} = \frac{t_{\text{RoI}} \times (P_h - P_{\text{RoI-min}})}{e_{\text{shutdown}} - e_{\text{resume}}} \quad \text{if} \quad P_h < P_{\text{RoI-min}}. \tag{6}$$

In other words, the system will be operational for t_o seconds on average before running out of energy which in turn means that it will run out of energy t_{RoI}/t_o times while executing the RoI. If $P_h \geq P_{\text{RoI-min}}$, the system is energy-feasible and n_b equals zero.

The energy cost of each backup is the sum of the energy spent on disconnecting (i.e., e_d) and reconnecting (i.e., e_c) as well as the energy overhead of preparing the non-volatile memory for reading and writing (i.e., e_{mem}), writing the checkpoint, and reading it back again:

$$E_b = n_b \times (e_c + e_d + e_{\text{mem}} + d_b \times e_b). \tag{7}$$

The energy cost of a checkpoint is the product of the average number of bytes it contains d_b and the energy cost of writing and reading a byte of data e_b. As prior work [2], [3], we find that backup and restore costs are nearly perfectly linear in the number of bytes that needs to be read or written.

Idle: ULP-systems use sleep modes, which are low-energy states where selected components are either power-gated or clock-gated, to save energy while idle. ULP-platforms commonly support different sleep modes which differ in how aggressively they turn off components. The benefit of clock gating is that the components can retain state and resume execution faster at the cost of primarily reducing dynamic energy whereas power gating saves both dynamic and static energy but does not retain volatile state and resuming execution takes longer. For these reasons, different applications tend to prefer different sleep modes. PES hence takes the average power consumption P_i during the application's preferred sleep mode as input.

The ULP-platform is idle when it is powered and inactive, and we predict idle time by first computing the time it takes the system to process and communicate the samples of the RoI (i.e., t_{RoI}) and then subtract the time the system is actively processing and communicating. More specifically, the system is active for t_a seconds for each of the $n_s \times n_c$ samples (i.e., $t_a = t_{\text{sys}} + \sum_{i=1}^{T} \mathbb{P}(i) \times t_s^i$) and active for t_c seconds in each of the n_c communication events. Idle energy E_i is hence:

$$E_i = P_i \times [t_{\text{RoI}} - n_c \times (n_s \times t_a + t_c)]. \tag{8}$$

Equation (8) accounts time as active (i.e., t_a) if the ULP-processor is processing samples or performing system tasks, and communicating (i.e., t_c) if it controls the radio or performs other protocol-related processing. If necessary, the application can exploit idle periods during communication to collect samples.

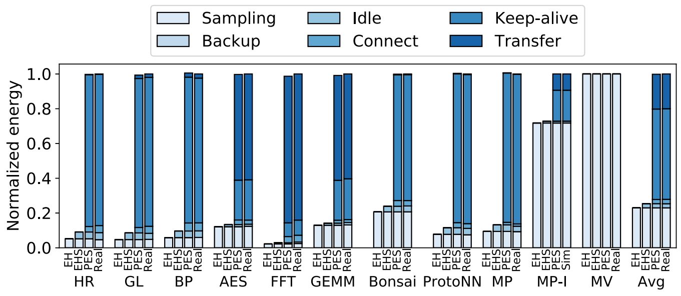

Fig. 2. PES model validation.

3 Experimental Setup

We use the Nordic Semiconductor's nRF52832 SoC [5] as our experimental platform and measure the instantaneous power consumption of the SoC at a frequency of 10 kHz using Lynsyn [6]. The SoC contains an ARM Cortex M4 processor running at 64 MHz and an integrated Bluetooth Low Energy (BLE) radio subsystem. We use Nordic Semiconductor's S132 BLE protocol stack and an Android mobile phone as the back-end system. To validate PES for intermittent design points, we use the solar panel energy trace from Kraemer *et al.* [7] to drive an in-house ULP-system simulator. Our error metric is absolute relative error (i.e., $\text{Error} = |E_{\text{measured}} - E_{\text{model}}|/E_{\text{measured}}$).

We select a range of IoT benchmarks that occupy different design points with respect to compute and communication intensity. More specifically, we run Heart Rate (HR), Glucose (GL), and Blood Pressure (BP) from the Nordic Semiconductor SDK [8], AES and GEMM from MachSuite [9], ProtoNN and Bonsai from SeeDot [10], FFT [11], and the Mean and Variance (MV) benchmark used to validate the EH-model [2]. To evaluate PES with time-variable applications, we create MP which first runs MV to compute the variance of the captured image and then ProtoNN if the variance is larger than 1%. We further include MP – I, which is MP configured to run at an intermittent design point, to validate PES' backup model.

4 Results

We now evaluate PES and compare to the EH-model [2], [3]. While EH only models compute, it is straightforward to extend it to model periodic systems that sleep (see Section 2). We refer to this model as EHS (EH with Sleep). We compare predicted energy consumption to actual consumption on our hardware platform (*Real*) and simulator (*Sim*) for energy-feasible and intermittent design points, respectively.

Fig. 2 reports predicted energy consumption (normalized to Real or Sim) and breaks down the energy consumption into the components of Equation (1). Overall, PES is very accurate with an average error across all benchmarks of 0.5% (maximally 1.4%); the residual error is mostly due to communication not being fully deterministic. PES' prediction error with MP is only 0.6% which demonstrates that PES can accurately model a time-variable application. PES is also accurate when MP is configured to run at an intermittent design point, see MP – I in Fig. 2. In contrast, EH is inaccurate (average error 77.0%), primarily because it does not model communication. However, we confirm that EH is accurate when applications compute continuously (i.e., MV). EHS improves accuracy compared to the EH (i.e., reduces average error from 77.0% to 74.5%), but it is still inaccurate compared to PES.

Fig. 3 shows the energy-feasible region for ProtoNN in a system with a solar panel energy harvester on three representative days throughout the year. We use the solar panel power output trace by Kraemer *et al.* [7] which is collected over a two-year period in Trondheim, Norway. Since Trondheim is close to the Arctic Circle,

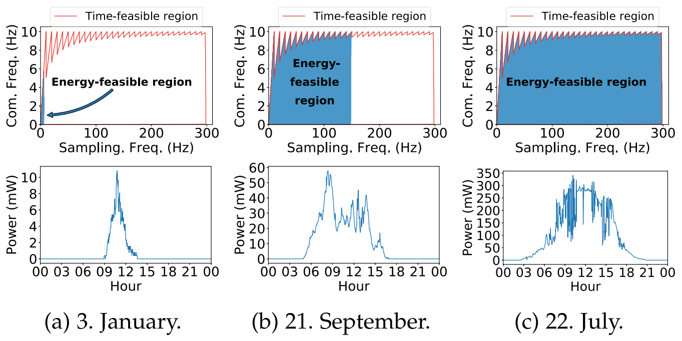

Fig. 3. Energy-feasible regions for ProtoNN with a solar panel harvester on three representative dates throughout the year.

there is very limited sunlight in winter which result in a daily average power output of merely 0.65 mW on 3. January. In summer, the sun hardly sets which results in orders of magnitude more power on average 22. July (i.e., 95.0 mW). 21. September is an intermediate case with an average power output of 11.5 mW. All power traces exhibit significant variability due to clouds. Fig. 3 also shows the time-feasible and energy-feasible regions of our hardware platform for ProtoNN on the respective dates; the time-feasible region indicates the compute capacity of the ULP-system. The x-axis shows sampling frequency and the y-axis effective communication frequency. We report effective communication frequency rather than communication incidence to make power consumption increase along both axes.

The time-feasible region is the same in Figs. 3a, 3b, and 3c because it is independent of the energy supply. When sampling frequency is increased from zero, the maximal communication frequency increases proportionally since every sample is immediately communicated (i.e., communication incidence equals one). Eventually, the ULP-processor saturates, i.e., it is always either sampling or communicating the sample. Increasing the sampling rate further requires increasing the communication incidence and thereby enable interleaving sampling and communication. This first reduces the effective communication frequency as two samples now have to be captured before they can be communicated. Then, sampling frequency proportionally increases the effective communication frequency until the ULP-processor again saturates. In general, increasing communication incidence frees processor resources and enables (almost) regaining the maximum effective communication frequency at a higher sampling rate; hence the triangular pattern.

The energy-feasible region is the area within the time-feasible region where the average power output of the energy harvester is greater than or equal to the average power consumption of the ULP-system. In Fig. 3a, ProtoNN maximally communicates at a rate of 5.8 Hz in which case the sampling rate is the same (i.e., communication incidence equals 1); the system requires a battery capacity of 53.3 J. In Fig. 3c, the maximal communication frequency only increases to 10 Hz because the ULP-system hits its communication latency limit, i.e., the sample needs to be fully communicated before the ULP-system has to start communicating the next sample. However, the sampling rate can be much higher than in January, i.e., maximally 298.1 Hz at a communication frequency of 34.8 mHz. Sustaining this performance-level requires a battery capacity of 8.2 kJ; supplying this amount of energy storage increases system cost and physical size and may hence be inappropriate.

5 Related Work

The EH-model [2], [3] enables design-time exploration of energy harvesting ULP-system design spaces, but it is inaccurate (see Section 4). Gobieski *et al.* [12] use an analytical model to motivate for performing inference within ULP-systems, but the model only considers energy and hence cannot be used to explore energy versus performance trade-offs. CatNap [4] guarantees that time-critical code will run successfully when sufficient energy is available; otherwise the application degrades gracefully. CatNap is hence orthogonal to PES as it focuses on achieving energy-feasible operation at runtime whereas PES helps developers match sampling and communication rates to energy harvester capabilities at design time.

6 Conclusion

We have now presented our Periodic Energy-Harvesting Systems (PES) model which enables IoT developers to explore performance versus energy trade-offs early in the design process. Most notably, PES can be used to identify sustainable and near-energy-feasible design points. Unlike the state-of-the-art EH-model [2], [3], PES faithfully models communication which leads to significantly lower energy prediction error.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable feedback.

References

[1] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Trans. Embedded Comput. Syst.*, vol. 6, no. 4, 2007, Art. no. 32.
[2] J. S. Miguel, K. Ganesan, M. Badr, and N. E. Jerger, "The EH model: Analytical exploration of energy-harvesting architectures," *IEEE Comput. Archit. Lett.*, vol. 17, no. 1, Jan.-Jun. 2018.
[3] J. S. Miguel *et al.*, "The EH model: Early design space exploration of intermittent processor architectures," in *Proc. Int. Symp. Microarchit.*, 2018, pp. 600–612.
[4] K. Maeng and B. Lucia, "Adaptive low-overhead scheduling for periodic and reactive intermittent execution," in *Proc. Conf. Program. Lang. Des. Implementation*, 2020, pp. 1005–1021.
[5] Nordic Semiconductor, "nRF52832," 2000. [Online]. Available: https://www.nordicsemi.com/Software-and-tools/Development-Kits/nRF52-DK
[6] A. Djupdal, B. Gottschall, F. Ghasemi, and M. Jahre, "Lynsyn and Lynsyn-Lite: The STHEM power measurement units," in *Towards Ubiquitous Low-power Image Processing Platforms*, M. Jahre, D. Göhringer, and P. Millet, Eds. Berlin, Germany: Springer, 2020.
[7] F. A. Kraemer, D. Palma, A. E. Braten, and D. Ammar, "Operationalizing solar energy predictions for sustainable, autonomous IoT device management," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11803–11814, Dec. 2020.
[8] Nordic Semiconductor, "BLE software development kit," 2020. [Online]. Available: https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.sdk5.v15.3.0/examples_ble_peripheral.html
[9] B. Reagen, R. Adolf, Y. S. Shao, G.-Y. Wei, and D. Brooks, "MachSuite: Benchmarks for accelerator design and customized architectures," in *Proc. Int. Symp. Workload Characterization*, 2014, pp. 110–119.
[10] S. Gopinath, N. Ghanathe, V. Seshadri, and R. Sharma, "Compiling KB-sized machine learning models to tiny IoT devices," in *Proc. Conf. Program. Lang. Des. Implementation*, 2019, pp. 79–95.
[11] Tomwi, "Fixed-point FFT," 2020. [Online]. Available: https://gist.github.com/Tomwi
[12] G. Gobieski, B. Lucia, and N. Beckmann, "Intelligence Beyond the Edge: Inference on intermittent embedded systems," in *Proc. Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2019, pp. 199–213.