Camaroptera: A Long-range Image Sensor with Local Inference for Remote Sensing Applications

HARSH DESAI, Carnegie Mellon University, USA
MATTEO NARDELLO and DAVIDE BRUNELLI, Univeristy of Trento, Italy
BRANDON LUCIA, Carnegie Mellon University, USA

Batteryless image sensors present an opportunity for long-life, long-range sensor deployments that require zero maintenance, and have low cost. Such deployments are critical for enabling remote sensing applications, e.g., instrumenting national highways, where individual devices are deployed far (kms away) from supporting infrastructure. In this work, we develop and characterize Camaroptera, the first batteryless image-sensing platform to combine energy-harvesting with active, long-range (LoRa) communication. We also equip Camaroptera with a Machine Learning-based processing pipeline to mitigate costly, long-distance communication of image data. This processing pipeline filters out uninteresting images and only transmits the images interesting to the application. We show that compared to running a traditional *Sense-and-Send* workload, Camaroptera's *Local Inference* pipeline captures and sends upto 12× more images of interest to an application. By performing *Local Inference*, Camaroptera also sends upto 6.5× fewer uninteresting images, instead using that energy to capture upto 14.7× more new images, increasing its sensing effectiveness and availability. We fully prototype the Camaroptera hardware platform in a compact, 2 cm × 3 cm × 5 cm volume. Our evaluation demonstrates the viability of a batteryless, remote, visual-sensing platform in a small package that collects and usefully processes acquired data and transmits it over long distances (kms), while being deployed for multiple decades with zero maintenance.

CCS Concepts: • **Hardware** → **Sensor devices and platforms**; **Sensor applications and deployments**; • **Computer systems organization** → *Sensor networks;*

Additional Key Words and Phrases: Energy-harvesting sensor, batteryless computing, edge computing, long-range sensing, long-lifetime sensing

ACM Reference format:
Harsh Desai, Matteo Nardello, Davide Brunelli, and Brandon Lucia. 2022. Camaroptera: A Long-range Image Sensor with Local Inference for Remote Sensing Applications. *ACM Trans. Embedd. Comput. Syst.* 21, 3, Article 32 (May 2022), 25 pages.
https://doi.org/10.1145/3510850

1 INTRODUCTION

Sustained by the recent advances in low-power sensing and computing technology, **Internet of Things (IoT)** devices are increasingly capable of interacting with the physical world. Systems that sense, compute, and communicate are now frequently deployed into human environments to sense

Authors' addresses: H. Desai and B. Lucia, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213 USA; emails: {harshd, blucia}@andrew.cmu.edu; M. Nardello and D. Brunelli, University of Trento, Via Sommarive 9 38123 Trento (TN) Italy; emails: {matteo.nardello, davide.brunelli}@unitn.it.

1539-9087/2022/05-ART32 $15.00
https://doi.org/10.1145/3510850

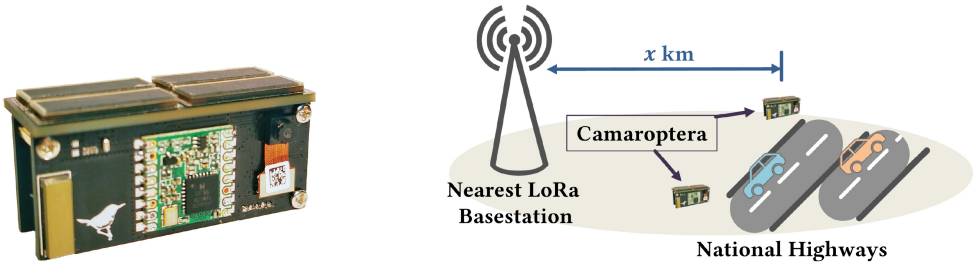


Fig. 1. (a) Camaroptera prototype. (b) Example remote sensing application (national highways), where each device is located far (kms) away from the nearest supporting infrastructure (base station).

and process important signals for a breadth of applications, including security, environmental science, urban planning [1] and optimization, precision agriculture, and even space exploration [11, 15, 78].

A challenging class of sensing applications are remote sensing applications, where the sensing devices are deployed far away from supporting infrastructure. Examples of remote sensing applications include deploying sensors deep into a rainforest or instrumenting national highways that are kilometers away from the closest towns [12] (one such scenario is shown in Figure 1(b)). Each sensor node deployed for such applications must be able to transmit data over long distances, potentially over multiple kilometers. The radios commonly used in sensor nodes, such as **Bluetooth Low-Energy (BLE)** or WiFi, cannot service these long-range requirements. Recently, chirp spread-spectrum technology has enabled radios like LoRa to achieve kilometer-scale data transmission [40] by sacrificing data-rate (vs. WiFi) and power-consumption (vs. BLE). LoRa presents a promising way to design sensor nodes for such remote deployments.

Remote sensing applications also require sensor deployments to have long lifetimes with zero maintenance. Such deployments are geographically distributed over large distances, making regular device maintenance an expensive operation in terms of cost as well as human effort. Over the past few years, improvements in energy-harvesting systems have led to the emergence of wireless IoT systems that are entirely *energy neutral*. These systems extract (e.g., Radio waves, solar) energy from their environment, buffering the energy in a battery [34] or capacitor [11]. After collecting sufficient energy, the system activates and performs some sensing, computing, or communication for its application. While energy-harvesting extends the lifetime of remote sensing systems, batteries still need to be periodically replaced.

Swapping these batteries for small capacitors or supercapacitors (batteryless operation) allows sensor deployments to achieve long lifetimes with zero maintenance. Such batteryless devices present several key benefits and have attracted growing interest in recent years [11, 15, 26]. Along with enabling maintenance-free, long lifetimes, batteryless devices avoid creating battery waste, allow the design of more compact and cheaper devices, and therefore, enable the development of the "next trillion" IoT devices [60], especially in remote sensing applications.

Future remote sensing systems must also support gathering *visual* sensor data to directly, rather than indirectly, observe complex environmental phenomena. However, high-data-rate visual data requires larger on-device storage and more energy for long-distance transmission than sensors with low-data-rates such as accelerometers or temperature sensors. Kilometer-scale transmission of visual data is even more challenging on batteryless systems, as energy-expensive transmission keeps the batteryless device busy with recharging energy, preventing it from sensing new, potentially important events. Consequently, deploying visual sensors in long-life, maintenance-free,

remote sensing applications demands solutions that mitigate this high-energy cost of long-range radio transmissions.

Our Contributions: In this article, we enable remote sensing applications with long lifetimes by developing the Camaroptera batteryless remote visual sensing system. Camaroptera senses visual spectrum data using an ultra-low-power image sensor. Camaroptera is also equipped with a LoRa [65] radio, enabling it to communicate over long distances, even in the presence of urban signal occlusion [16]. Camaroptera is batteryless and harvests its operating energy using small solar cells, storing energy in a small supercapacitor. Finally, Camaroptera employs on-device processing of data to reduce the use of its costly radio, using the radio only when it identifies data to be interesting to the application. Our main contributions with Camaroptera are:

- Camaroptera presents the first batteryless, energy-harvesting platform to support active, **long-range communication (LoRa)**, enabling long-life, maintenance-free deployments of sensor nodes in remote sensing applications.
- Camaroptera mitigates the high energy cost of kilometer-scale communication, with a *Local Inference*-based processing pipeline that identifies and transmits only the images that are interesting to the application.
- We develop a fully functional prototype of Camaroptera, with a compact 3cm × 5cm × 2cm footprint. The prototype works within these tight volume constraints, which limit solar cell output (to a few mW) and energy storage volume (e.g., 33mF at 3V).

Our evaluation shows that *Local Inference* allows Camaroptera to reduce the transmission of uninteresting images by upto 6.5× when compared against *Sense-and-Send*, a popular design for sensor nodes. Camaroptera uses the extra energy to capture upto 14.7× more new images, reporting upto 12× more interesting events than *Sense-and-Send*.

2 BACKGROUND

Kilometer-range, batteryless image-sensors are critical to enabling future remote sensing applications. Unmodified, existing long-range wireless technologies (2.1) and batteryless systems (2.2) are key enablers, although this work is distinct in its goals and mechanism from existing batteryless image sensors. Camaroptera brings these ideas together, addressing the unique challenges of long-range communication for remote image sensing by using local computation.

2.1 Long-range Wireless Communication

Remote sensing devices are increasingly able to include a long-range radio, such as a LoRa chirp spread-spectrum radio [16, 18, 40]. LoRa **integrated circuits (ICs)** are commercially available, inexpensive, and offer long range (i.e., kilometers) at relatively low power (i.e., hundreds of mW). Extensible receiver infrastructure, like OpenChirp [16], affords simple, publish/subscribe data management (e.g., MQTT) with simple endpoints. The ability to communicate kilometers at low power creates the opportunity for more devices to be deployed in remote environments than is possible using other radio technologies. 4G/LTE incurs per-byte subscription costs [52], Bluetooth has limited range [8, 20], and WiFi requires many access points for wide-area coverage. Backscatter [37, 38, 69, 73, 76] is appealing, although limited by the need for large, powered transmitter infrastructure that provides wireless power and a communication carrier signal. LoRa provides a critical balance between long range and low-power operation, making it suitable for remote IoT deployments. While LoRa consumes relatively low power (hundreds of mW), this power draw can still be expensive for energy-constrained remote sensors, who must judiciously use their radio link or risk exceeding their constrained energy budget.

2.2 Batteryless Systems

Batteryless, energy-harvesting devices are emerging to support sensing, communication, and computation with no batteries for energy storage [11, 25, 27, 63, 66]. These devices harvest energy from light [11, 34], **radio waves (RF)** [63], user interaction [36], or other sources, storing the energy in a capacitor or super-capacitor and using it when sufficient energy accumulates to do useful work. Some batteryless devices operate only intermittently, as energy accumulates [10, 28, 44, 45, 48, 62, 77]. A key advantage to batteryless operation is eliminating the need for per-device maintenance to replace batteries, which effectively extends deployed device lifetimes to the lifetime of the ICs on the board. Eliminating batteries also has secondary benefits: reducing size, weight, and environmental footprint (battery waste). In contrast, battery-powered systems have a number of disadvantages. Batteries require maintenance, because they need to be replaced over a long lifetime: Fixed batteries deplete, and rechargable batteries have a limited number of recharge cycles. Some recent work combines rechargable and fixed battery storage [34] using LTO batteries. These batteries bring more recharge cycles, but smaller capacity (40 mAh) than Lithium-Ion batteries, impeding their use in multi-decade deployments with energy-hungry radios.

Batteryless devices are becoming useful in image sensing platforms [57, 58]. Existing devices are limited, however, in their reliance on an instrumented environment with installed wireless power and a backscatter communication medium precludes their application in wide-spread, long-range (i.e., kilometer-scale) deployments. Avoiding complex wireless power infrastructure, which increases cost, is a key problem that we address in this work.

Communication is energy-hungry. Communicating large image data over multiple kilometers has a high energy cost, making it challenging to deploy on batteryless devices typically designed for ultra-low-power operation. Camaroptera's main goal is to use local computation to address this challenge. Computing to process a QQVGA image using an image-classifying neural network consumes around $65mJ$ and transmitting an image consumes $288mJ$ (Section 5 describes our platform in detail, and Section 8 describes these data in more detail). High energy costs translate to high time costs in a batteryless system, because a batteryless system must spend its time collecting the energy that it uses to operate. Figure 2 shows the collection time at different power levels for the quantity of energy required to transmit an image: At 10–20 kLux (a typical cloudy day outdoors), recharging takes 45 seconds to two minutes. A batteryless device is unavailable to sense new data during this recharging period, and a long recharging latency could cause it to miss important events.

Many existing sensor designs commonly employ a *Sense-and-Send* approach, transmitting data as they are collected [57, 58]. *Sense-and-Send* is, however, a poor match for batteryless, remote-sensing, because transmitting all data leads to a high aggregate recharge time, which may miss important events. Moreover, all data are not equally interesting and sending these data at high cost is not useful to an application.

Camaroptera reduces unnecessary communication (and recharging) by computing locally on sensed data to find interesting data that should be transmitted, rather than indiscriminantly transmitting data in the *Sense-and-Send* model. While the computational capability of low-power microcontrollers is limited, prior work shows promise for sophisticated sensor data processing on batteryless, energy-harvesting devices [23]. With architectural support for yet more efficient computing in ultra-low-power MCUs [14, 24], the computational capabilities available to batteryless, energy-harvesting sensor nodes will further increase. The increase in computational capability of batteryless systems motivates Camaroptera, as it allows the use of local computing for reducing costly long-range communication, consequently enabling maintenance-free, multi-decade deployments of batteryless, long-range sensing systems.

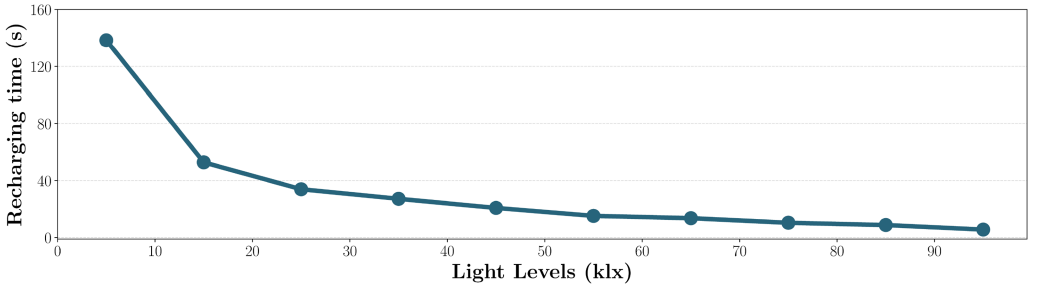


Fig. 2. Energy collection times at different light levels for sending a JPEG-compressed image over the LoRa radio on Camaroptera. An energy-harvesting system cannot capture new data during these energy collection periods, which can go upto two minutes at lower light levels, causing it to miss potentially important events. Camaroptera uses on-device processing to avoid using the radio for transmitting uninteresting events, saving the associated time and energy for capturing and processing more new events.

3 CAMAROPTERA DESIGN REQUIREMENTS

Camaroptera's design is motivated by the unique challenges of deploying a batteryless device that is capable of high-data-rate image sensing and expensive, long-range communication. We identify four key requirements for designing an effective remote sensing system and present how existing works fail to meet them. These requirements are **(R1)** kilometer-range, **(R2)** multi-decade lifetime, **(R3)** minimal environmental impact, and **(R4)** low cost.

(R1) Kilometer-range. A remotely deployed, image-sensing system must cover a large geographic area while respecting cost and maintenance requirements. Communication cannot rely on large numbers of high-cost base stations that require continuous power (i.e., imposing an infrastructure cost) or large batteries (i.e., imposing a maintenance cost). Each device must, therefore, support transmission over a multiple kilometers independently to enable large-scale, geo-distributed remote sensing applications.

(R2) Multi-decade Lifetime. A remotely deployed, image-sensing system must operate for a long period of time (decades) with zero maintenance, given the high maintenance costs of geo-distributed systems. These systems should require no component or battery replacements over its lifetime, and should operate without costly centralized power or communications infrastructure; fully autonomous and wireless operation is ideal.

(R3) Minimal Environmental Impact. A remotely deployed image-sensor must have minimal negative impact on its environment. It must minimize its short-term environmental impact by being small, unobtrusive, and by not interfering with existing (e.g., radio) infrastructure. It must minimize its long-term environmental impact by minimizing the amount of hazardous chemical waste due to batteries and other toxic components.

(R4) Low Cost. A remotely deployed image-sensing system must have a low cost. Devices must be manufactured at large scale (i.e., millions) requiring each device's cost to be low to minimize total cost. These devices must also avoid relying on costly centralized power or communications infrastructure. Minimizing cost minimizes operator liability, as deployed devices are at risk of damage or loss due to vandalism, animal interactions [43], and weather.

3.1 Existing Systems Fall Short

Existing systems meet some, but not all, of the requirements for designing effective remote-sensing systems, as we show in Table 1. Prior attempts at batteryless image sensing [57, 58] rely on short-range, RF-energy-transmission infrastructures, violating requirements **R1, R3,** and **R4**. While prior work on batteryless communication often relies on short-range RF-backscatter [19, 37, 38, 76],

Table 1. Comparing Camaroptera with Prior Work

Requirement	Batteryless Cams [57, 58]	Backscatter [19, 37, 38, 76]	LoRa Backscatter [69]	Magno et al. [22]	Permamote [34]	Camaroptera
R1: km-range	✗	✗	✗	✔	✗	✔
R2: Multi-decade	✔	✔	✔	✗	✗	✔
R3: Low Env. Impact	✗	✗	✔	✗	✔	✔
R4: Low Cost	✗	✗	✔	✗	✔	✔

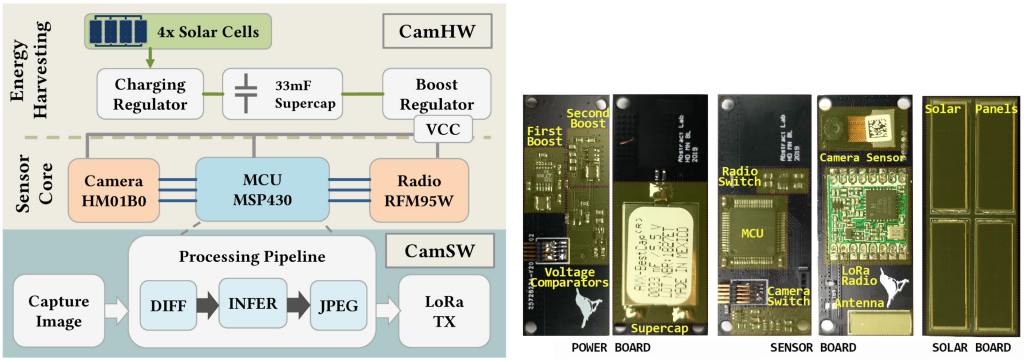
a prior work has proposed using LoRa signals for energy-harvesting and backscatter communication [69], eliminating costly RF-energy-transmission infrastructures. However, LoRa backscatter supports a maximum separation of 475 m between the source and receiver, failing **R1** and precluding its use for remote-sensing applications. In contrast, we equip Camaroptera with an active, long-range LoRa radio, enabling kilometer-range communication (**R1**) and a decrease in environmental impact and infrastructure requirements (low-power receivers only—**R3**).

A recent system (Magno et al. [22]) presents a batteryless image-sensing platform, designed for face recognition. While it employs a LoRa radio to achieve a long deployment range, it uses an ARM Cortex-M4 core with FLASH memory, whose limited write endurance precludes long-lifetimes and fails (**R2**). In contrast, Camaroptera uses an MSP430 microcontroller with embedded **Ferroelectric RAM (FRAM)** for long-lifetime non-volatile storage. Magno et al. also is designed for indoor operation, resulting in a power system with ∼ 6× larger solar panels and a larger cost and environmental impact (**R3, R4**).

Permamote [34] presents a battery-powered solution for long-term, low-cost sensing. However, it achieves a long lifetime by restricting operation to low-data-rate sensors (accelerometers and temperature sensors) combined with short-range, low-power communication, making it unsuitable for image-sensing applications with high-power, kilometer-range radios. Camaroptera achieves a maintenance-free, multi-decade lifetime (**R2**) by relying on batteryless operation, which also lowers per-device cost (**R4**) compared to equivalent, battery-powered systems.

4 CAMAROPTERA DESIGN OVERVIEW

Camaroptera is a batteryless sensing, computing, and communication system composed of a custom hardware platform, application-level software components, and software control components. As outlined in Section 3, we have designed Camaroptera according to the design requirements for multi-decade deployments in remote sensing applications. The hardware and software components of Camaroptera are outlined in Figure 3(a). Each Camaroptera device is built on CamHW, a custom hardware platform. The hardware includes a small, low-power image sensor to collect images, a microcontroller with embedded memory to process images, a long-range radio chip for communication, and a solar energy-harvesting power system for collecting and storing energy from the environment. CamHW is composed of several **printed circuit boards (PCBs)** that assemble into a three-dimensional device. Section 5 describes CamHW in detail and Figure 1(a) shows the assembled device hardware. The software of each Camaroptera device is built around CamSW, a simple operating system and device driver layer. CamSW manages sensor data collection and operates an *at-sensor processing pipeline* to process collected images. The at-sensor processing pipeline can support arbitrary application-specific operations (e.g., CNN/DNN-based image classifiers [23]) and built-in operations, which include image-difference detection and image compression. Our implementation performs DNN-based image classification, followed by JPEG compression for the images to be transmitted. CamSW also controls the radio hardware and segments and packetizes data for transmission.



(a) (b)

Fig. 3. (a) System overview of Camaroptera. (b) Camaroptera prototype PCBs.

Meeting Design Requirements. Together, Camaroptera's hardware and software meet the design requirements for a pervasive, long-term image sensing device. CamHW includes a LoRa IC, enabling communication over kilometer ranges (**R1**). Kilometer-scale communication is a key requirement for widespread deployment without excessive base station infrastructure costs. Camaroptera's batteryless energy-harvesting power system is simple to deploy and requires no maintenance once deployed (**R2**). Deployment requires simply installing a device with its solar panels exposed to sunlight, not requiring any direct connection to infrastructure. Batteryless operation requires no battery replacements and produces no battery waste (**R3**). Camaroptera's design minimizes its cost (**R4**). CamHW is a low-cost two-layer custom PCB populated entirely with COTS components and ICs. Each fully assembled device costs around USD$50 in low volume; high-volume pricing will further reduce the device cost. Low cost enables large-scale deployments.

CamSW is designed for immediate, at-sensor processing of collected sensor data. Application-specific at-sensor processing allows Camaroptera to identify uninteresting data and to discard them immediately and avoid consuming the energy, time, and bandwidth required to send them to a base station. Camaroptera then uses this saved energy and time to capture and process new images, improving its sensing effectiveness and availability. By identifying the images of interest to an application and sending only these images, Camaroptera efficiently uses the scarce bandwidth available to remotely deployed devices.

5 HARDWARE DESIGN

CamHW is a hardware platform designed for batteryless sensing and computing and long-range communication. CamHW is composed entirely of COTS components, limiting the per-device cost. The platform mounts these COTS components on three small two-layer PCBs mounted to one another in a three-dimensional package. CamHW's three boards are the *sensor board*, the *power board*, and the *solar board*. The sensor board includes sensing, computing, and communication components. The power system board includes energy storage and power conditioning components. The solar board includes the device's solar cells and provides structural support for the manufactured device. Figure 3(b) shows a photograph of the populated sides of the three boards.

5.1 Sensor Board

The sensor board incorporates the main active components of Camaroptera's remote sensor system, including a **microcontroller (MCU)**, a low-power image sensor, and a LoRa transceiver with

a ceramic patch antenna. The sensor board hardware is agnostic to its power system and can be powered by Camaroptera's power board or by a standard 3V supply.

MCU. Camaroptera's MCU is a Texas Instruments Ultra Low-Power MCU MSP430FR5994 [71] running at 16 MHz with 8 kB of SRAM and 256 kB of embedded non-volatile **Ferroelectric RAM (FRAM)**. We select this MCU specifically for the embedded FRAM memory, as its non-volatility allows Camaroptera to save image data across power failures, and the higher write endurance of FRAM (compared to FLASH) allows Camaroptera to be deployed for multiple years without memory corruption. The limited memory, low clock frequency, and simple architecture of the MSP430FR5994 are key challenges to supporting sophisticated computations, as observed in prior work [23], and require device-specific software optimizations (described in Section 6.2).

Image Sensor. Camaroptera uses a Himax HM01B0 [30] image sensor, which is an ultra-low-power CMOS image sensor. Its sensor has an active area of 320 × 320 pixels each of which with a side dimension of 3.6μm. Camaroptera configures the camera to operate in QQVGA (160 × 120) mode, capturing 8-bit grayscale images. We use this image sensor for its ultra-low-power operation ($\leq 1.1mA$ for a QQVGA image).

Transceiver. Camaroptera uses a Semtech RFM95W LoRa chirp spread spectrum [64] modulation transceiver IC [31]. The chip incorporates an ultra-low-power 20 dBm power amplifier with a sensitivity over −148 dBm. Camaroptera connects this LoRa IC to a ceramic chip antenna with a maximum gain of 3.42 dBi. Using a LoRa transceiver allows Camaroptera to communicate the captured images to remote base stations located multiple kilometers away.

5.2 Power Board

The power board implements Camaroptera's energy harvesting power system. The power system uses a two-stage voltage boosting circuit with hardware voltage comparators to keep system voltage in the most efficient operating range for the boosters. The power board also houses Camaroptera's supercapacitor-based energy storage.

Boosting. The first voltage boosting stage connects the solar cells to the supercapacitor using an LTC3105 [42]. The booster is a high-efficiency step-up DC/DC converter that operates down to 225 mV input and supports **maximum power point control (MPPC)**. Both of these features are important for operating on variable solar energy.

The second boosting stage connects the energy storage capacitor to the sensor board using a TPS61070 [70] synchronous voltage boost converter. This boost IC provides efficiency over 85% with input as low as 1.2V for a regulated output of 3V. Camaroptera controls the booster operation, keeping it powered off when the supercapacitor voltage is outside its maximum efficiency region.

Voltage Thresholding. Camaroptera uses two MIC841 [53] voltage comparators with an externally adjustable hysteresis to drive the enable input of Camaroptera's second booster and the reset line of the MCU. The first comparator ensures that the second boosting stage is enabled only when the energy storage capacitor is charged between a lower threshold of 1.24 V and its rated maximum of 3 V. Camaroptera's second comparator holds the MCU in reset while the V_{CC} output of the second boosting stage stabilizes. We empirically determined that the system stabilizes at 2.2 V, which is the the minimum voltage to operate the LoRa modem. The second comparator is set to a low threshold of 2.2 V and a high threshold of 3 V.

Energy Storage. Camaroptera stores energy in a high-density supercapacitor. Camaroptera's supercapacitor must store sufficient energy to ensure that its longest energy-atomic task completes without exhausting the device's stored energy. Camaroptera's largest energy-atomic task is sending a single 255-byte LoRa packet. Under this constraint, we equipped Camaroptera with a BestCap [4] 33 mF high-density supercapacitor. This supercapacitor also has a low **Equivalent Series Resistance (ESR)** enabling it to provide the high radio current with a smaller drop in voltage.

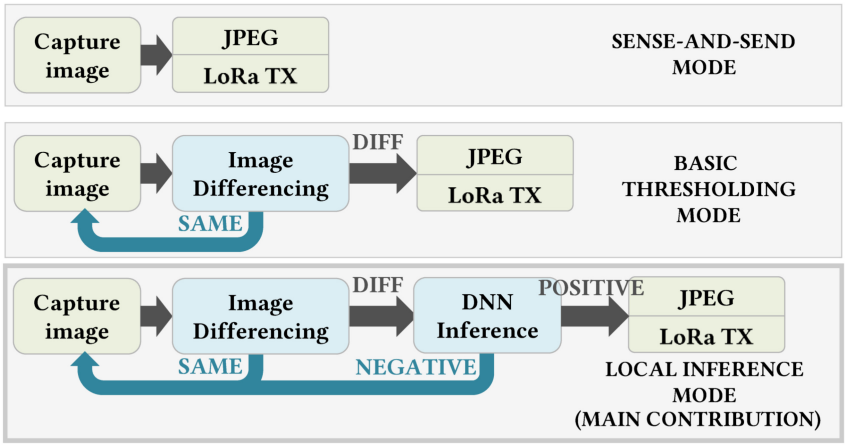


Fig. 4. Camaroptera software flow chart.

5.3 Solar Board

Camaroptera's solar board is covered by four solar cells connected in parallel and mounts perpendicularly to the sensor and power boards. The solar cells are an array of IXYS [33] high-efficiency monocrystalline cells, measuring 22 mm × 7 mm each. The solar board provides structure and power for the assembled device with mechanical and electrical connections to the other boards.

6 SOFTWARE DESIGN

Camaroptera's software subsystem, CamSW, is centered around locally processing captured images to avoid transmitting images that are not interesting to an application.

6.1 Local Processing

CamSW locally processes every image after its capture. The goal of processing is to find images of interest to the application that should be transmitted, otherwise discarding uninteresting images. Avoiding transmission of uninteresting images enables Camaroptera to judiciously use its time and energy. CamSW supports arbitrary image processing pipelines that can vary by application. A pipeline may use general filtering operations, such as differencing, or application-specific ones, such as a deep neural network trained for a specific task.

We built a representative, prototype processing pipeline designed to identify and transmit images containing people. This representative person-detection pipeline has both general and application-specific stages. The person-detection pipeline includes four stages. Figure 4 shows the software flow of this person-detection pipeline.

Image Differencing. After capturing an image, Camaroptera compares it to the most recently captured image to determine if the new image differs. If the image differs, then it may be of interest to the application and should continue through the pipeline. If not, then Camaroptera can safely discard the image. After comparison to a previous frame, Camaroptera takes the new frame and saves it as a reference for future differences. Difference computation is generally applicable.

DNN Inference. If a new image differed from a previous one, then Camaroptera runs an application-specific inference routine on that image to identify whether it is interesting to the application. For our person-detection application, Camaroptera runs a deep neural network trained to detect images containing people. Inference using statistical methods or learned inference models are inherently application-specific and each application requires its own model. For person detection, we run a single model, but Camaroptera supports cascading models, as MCU resources permit.

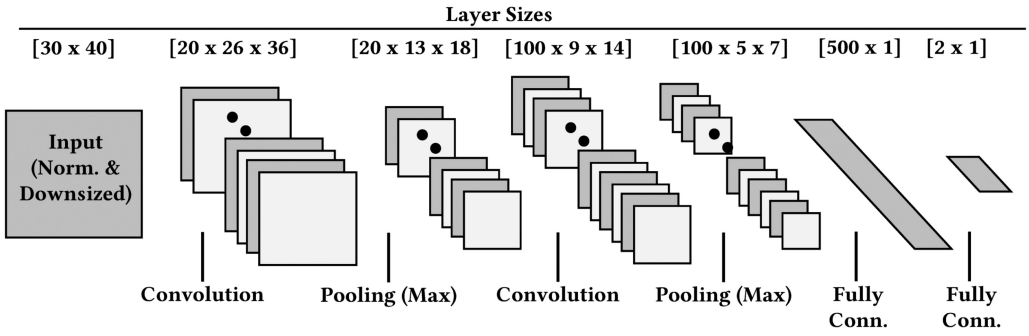


Fig. 5. The DNN architecture used by Camaroptera to perform person detection.

Pre-transmission Transformation. If an image is deemed interesting by inference, then Camaroptera transforms the image to prepare it for transmission. This transformation could include a variety of encoding and encryption, depending on application requirements. In our person-detection prototype, we compress each interesting image before transmission.

Transmission. Once transformed for transmission, Camaroptera packetizes the image and transmits each of an image's packets in sequence using its LoRa radio.

6.2 Software Implementation Details

We implemented Camaroptera's processing pipeline, including differencing, inference, and compression.

Differencing. The goal of this processing stage is to act as a fast filter for images of static or slowly changing scenes. We implemented a simple image differencing algorithm that explicitly compares corresponding pixel values between images. We chose this approach for its simple and fast operation, where a more sophisticated approach (e.g., SSIM-based) would be considerably slower on the MSP430 MCU, which lacks a floating-point unit and must rely on software emulation of floating-point operations.

In our simple approach, we deem images different from one another if the number of *different* pixels exceeds a heuristically defined threshold. We set the threshold empirically to 400 pixels by observing that human figures in our images tend to be around 20×20 pixels in size.

Inference. Our Camaroptera prototype uses a **Deep Neural Network (DNN)** for image classification. The DNN's structure is derived from the LeNet [39] digit classification convolutional neural network. The architecture of the network we used is shown in Figure 5. We trained the LeNet-structured DNN using a set of images collected using our Camaroptera prototype. We collected 4,000 images around our university campus in five different locations in a wide variety of lighting conditions. We used Amazon Mechanical Turk [2] workers to label them as containing a person, not containing a person, or not being a valid image. The dataset included 60% negative images and 40% positive images. After labeling the dataset, we trained the network using a subset of 3,600 of the images, holding 10% aside for testing and validation.

We optimized the network to decrease its size because the model, as trained, does not fit in the 250 kB of available memory on our MSP430 MCU. First, we stored the network weights in a sparse format, using 16-bit fixed-point integers to save space and increase compute speed. The MSP430 MCU does not natively support floating point operations and software emulation is extremely slow. All computations for the network were therefore performed in fixed-point arithmetic, with kernels written for sparse matrices. Second, we reduced the size of the network by passing the 160×120 input image through a 4×4 average pooling layer, before the network's first convolutional

(a) (b) (c)

Fig. 6. Comparing an uncompressed and JPEG compressed images captured by Camaroptera. (a) Original image. (b) Floating point JPEG. (c) Fixed point JPEG.

layer. The pooling layer effectively reduces the input to 30 × 40 pixels. We then use the Genesis [23] network minimization tool to perform hyperparameter (i.e., structural) optimization on our trained, modified network. Genesis applies aggressive near-zero pruning and layer separation to reduce a network's weight storage requirements, the size of its intermediate activiations, and the expected inference latency and energy. Camaroptera encodes the optimized network from Genesis as **Compressed Sparse Row (CSR)** for space efficiency. After optimization, the network's weights consume 20 kB of memory, which is a substantial reduction compared to its initial 3.6 MB of network weights. The final network additionally requires around 80 kB for intermediate activations and yields 78% accuracy on a test set, with 40% **False Positives (FP)** and 1% **False Negatives (FN)**. Section 8 evaluates Camaroptera with different false positive and false negative ratios.

Compression. Camaroptera implements JPEG compression, based on an existing implementation [56]. As with inference, we modified the JPEG implementation to use fixed point arithmetic instead of floating point, given the lack of floating point support in the MSP430 MCU. Shifting to fixed point reduced the latency to compress a 160 × 120 image from 25 seconds to 7 seconds. Fixed point JPEG degrades image quality, but not excessively. Figure 6 compares an uncompressed image and ones compressed using floating and fixed point.

We additionally optimized the transmission of JPEG headers. The first 500 bytes of the compressed bit stream is always the same, using the same quality factor and frame resolution. We store the header on the receiver and avoid sending the 500 bytes of header data, which amounts to the transmission of two LoRa packets and seconds or minutes of device operation.

Transmission. We operate the LoRa radio with the following parameters: Frequency = 915 MHz, Bandwidth = 500 kHz, Spreading Factor = 7, Coding Rate = $\frac{4}{5}$, Preamble Length = 8 symbols, Output TX Power = 17 dBm, Packet Size = 255 bytes.

Energy Management. We manage the energy usage of different tasks on Camaroptera in software by charging the capacitor to 3 V before the start of each task. A task can be either image capture, any single stage of the processing pipeline, or transmission of a single LoRa packet. Once the execution of a task is complete, if the capacitor voltage is below 3 V, Camaroptera goes into deep-sleep and charges the capacitor before resuming operation from the next task. We have sized the capacitor to atomically support the largest energy task (transmission of single LoRa packet), avoiding mid-task power failures.

7 EVALUATION METHODOLOGY

We evaluated Camaroptera using a fully built hardware and software prototype (Figure 1(a)) to show that local inference effectively enables Camaroptera to find and transmit interesting images, to avoid transmitting uninteresting images and to capture and process more images overall compared to several baselines. We also characterize Camaroptera across a range of software configurations and environments. Running experiments on a real Camaroptera device in real lighting

conditions allowed capturing real system timing behaviors, including energy-harvesting inefficiencies, processing latency, and energy variability.

7.1 Application

Our prototype implements a representative person-detection application using a deep neural network and sends images containing people to the base station, which we refer to as operating in the *Local Inference* mode. We demonstrate the effectiveness of this *Local Inference* mode by comparing it with two alternate modes: *Sense-and-Send* and *Basic Thresholding*. The software flowcharts for these three modes are shown in Figure 4. *Sense-and-Send* indiscriminately transmits all the images it captures, only running JPEG compression for each image. *Basic Thresholding* sends JPEG compressed images that signficiantly differ from the device's previous image, with 2% or greater number of pixels differing by 15% or greater. *Local Inference* uses local DNN inference to detect images of people. *Local Inference* first runs *Basic Thresholding* and for significantly different images, invokes the trained DNN person detector from Section 6. *Local Inference* sends images classified as containing people only and discards other images.

7.2 Methodology

We ran three multi-week, in-lab experiments, with Camaroptera handling images in one of *Sense-and-Send*, *Basic Thresholding*, and *Local Inference* modes (Section 7.1) for the respective experiment. We evaluated how Camaroptera operates at three different, realistic outdoor light conditions ranging from overcast to bright sun (*15, 30, and 45 kLux*), setting these light levels using a dimmable incandescent bulb and a URCERI MT-912 light meter. Each trial had 5,000 image events emulated via an MSP430 experimental control board connected to Camaroptera. We emulated image delivery to ensure repeatable experiments. To emulate an *event* (i.e., a significantly different image), the control board raises the *event* GPIO pin, connected to Camaroptera. To emulate an *interesting event* (i.e., an image containing a person), the control board raises the *interesting* GPIO pin, connected to Camaroptera. The control board generates *events* and *interesting events* with Gaussian ($\mu = 3s$, $\sigma = 1s$) durations and Poisson ($\lambda = 10s$) inter-arrival times, based on our informal profiling of real, pre-COVID19 human activity on a college campus during daytime. We use the same event sequences across trials with different system configurations to ensure fair comparison.

As outlined in Section 7.1, we studied Camaroptera in three operating modes. In the *Sense-and-Send* mode, Camaroptera captures and transmits images continuously. In the *Basic Thresholding* and *Local Inference* modes, Camaroptera first captures an image. If Camaroptera captures an image with the *event* GPIO raised, then the image is treated as an event. *Basic Thresholding* transmits an image if it is an event, assuming perfect discrimination of events using differencing. For the *Local Inference* mode, an image captured with both the *event* and *interesting event* GPIO pins raised indicates the presence of a person, and not otherwise. The number of *event* images that are *interesting* is dictated by the **True Positive (TP)** rate: *TP%* of all *event* images will be marked as *interesting*. For every *event* image, Camaroptera runs its classifier and makes a classification judgment uniformly randomly based on its **false positive (FP)** and **false negative (FN)** ratio: An uninteresting image has *FP%* chance of being transmitted, and an interesting image has *FN%* chance of begin discarded. We ran the multi-week experiment with a FP:FN = 10:10 classifier that emulated a reasonable operating point and a 1% TP rate.

8 EVALUATION RESULTS

Our evaluation shows that compared to the *Sense-and-Send* and *Basic Thresholding* modes, Camaroptera's *Local Inference* mode enables it to capture and transmit more interesting images, transmit fewer uninteresting images, and capture more total images. We evaluated Camaroptera's

sensitivity to variation in several evaluation parameters, including the rate of interesting events in the environment (True Positive Rate) and the FP:FN rate of the classifier.

8.1 Local Inference Yields Better Data

We present the results of the multi-week experiment described in Section 7.2, showing that the *Local Inference* mode yields better data than *Sense-and-Send* and *Basic Thresholding*, providing more interesting images, fewer uninteresting images, and processing more images overall.

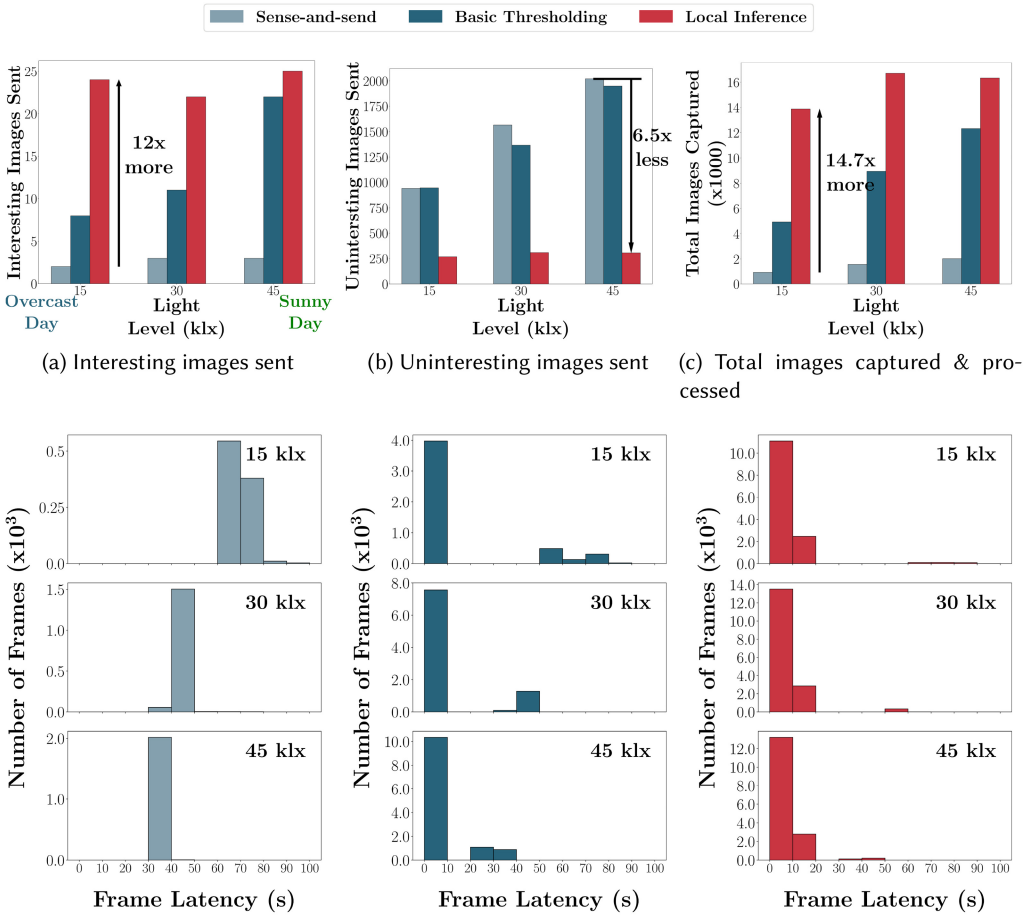
Local Inference sends *more interesting* images: Figure 7(a) shows the number of interesting images—images containing people—that Camaroptera captures in the *Local Inference* mode, compared to the *Sense-and-Send* and *Basic Thresholding* modes. Across all three light levels, Camaroptera with *Local Inference* captures and sends a larger number of interesting images. This difference is greatest at low input power (15klx), with the *Local Inference* mode enabling Camaroptera to send around 12× more interesting images than the commonly used *Sense-and-Send* mode, and around 3× more than the *Basic Thresholding* mode. The *Local Inference* mode outperforms the other two modes as it uses energy judiciously, avoiding the costly transmission of uninteresting images.

The data show that in *Local Inference* mode, Camaroptera transmits as many as 50% of all interesting images, which is 12*x* more interesting images than *Sense-and-Send* transmits. There are two main reasons that Camaroptera does not capture 100% of interesting images in *Local Inference* mode. First, Camaroptera spends time processing each image, creating a risk of not capturing an interesting image while processing an uninteresting one. Section 9 discusses strategies to further reduce this risk and capture more interesting events. Second, even in *Local Inference* mode, Camaroptera spends time recharging energy spent transmitting interesting events, which blocks capturing new data.

Local Inference sends *fewer uninteresting* images: Camaroptera's local inference avoids transmitting uninteresting images more effectively than other modes. Figure 7(b) shows how the number of uninteresting images sent varies with input power across the three operating modes. *Local Inference* mode sends up to 6.5× fewer uninteresting images than *Sense-and-Send* mode. Additionally, while *Local Inference* mode transmits a roughly constant number of uninteresting images across input power levels, *Sense-and-Send* and *Basic Thresholding* send more uninteresting images as input power increases. *Local Inference* avoids the problem of eager transmission faced by *Sense-and-Send* and *Basic Thresholding*: As power increases, recharging becomes faster, enabling sending *more* images. However, without the ability to discriminate interesting from uninteresting, *Sense-and-Send* and *Basic Thresholding* more quickly send more uninteresting images.

Local Inference captures *more total* images: Operating in the *Local Inference* mode allows Camaroptera to avoid costly transmission of uninteresting images and use that energy to capture and process newer images. Figure 7(c) shows the total number of images that Camaroptera processes in all the operating modes across the whole trial. *Local Inference* mode enables Camaroptera to capture upto 14.7× more total images than the *Sense-and-Send* mode, and upto 2.8× more than the *Basic Thresholding* mode. Processing more raw images by using local inference decreases the chance that Camaroptera misses a critical, interesting event.

Transmitting images is energy-expensive, and collecting that energy takes significant time. Avoiding unnecessary image transmission allows the *Local Inference* mode to significantly reduce the energy collection time, resulting in less time spent on each image than the other two modes. Figures 7(d) to 7(f) show the distribution of total time spent on an image frame, across three light levels for each mode. The total time includes the time to capture, process and, if applicable, transmit the frame, as well as the time to collect the energy required for these tasks. As radio transmissions are energy-expensive, the frames that are transmitted incur a large latency, dominated by energy



(a) Interesting images sent (b) Uninteresting images sent (c) Total images captured & processed

(d) Latency distribution for *Sense-and-Send* (e) Latency distribution for *Basic Thresholding* (f) Latency distribution for *Local Inference*

Fig. 7. (a) to (c) show the results of a multi-week experiment demonstrating that the *Local Inference* mode outperforms the *Sense-and-Send* and *Basic Thresholding* modes on Camaroptera, in terms of Interesting images sent, Uninteresting images sent, and Total images captured, respectively. (d) to (f) show the per-frame latency distribution across the three operating modes, for three different light levels.

collection. This can be observed in Figure 7(d), which shows the *Sense-and-Send* mode transmitting every image, and thus incurring large frame latencies (60*s* to 90*s* at 15 klx) on all the frames it captures. Further, the frame latencies are higher when input power is low (≥60*s* at 15 klx), due to slower energy collection, and lower at higher input power (30*s* to 40*s* at 45 klx), where energy collection is faster. The *Basic Thresholding* mode (Figure 7(e)) avoids transmitting unchanged images, which is reflected in the large number of frames with latency ≤10*s*, as these frames avoid large energy collection delays. However, the uninteresting images that the *Basic Thresholding* mode transmits still incurs this large delay, as represented by the frames having large total times (between 50*s* and 90*s* at 15 klx). In contrast, the *Local Inference* mode (Figure 7(f)) minimizes the total per-frame latency by using the high-energy radio only for transmitting the images it classifies as interesting for the application. This results in very few frames incurring the high energy collection cost of radio transmissions (frames with latency between 60*s* to 90*s* at 15 klx). Majority of the frames in the *Local*

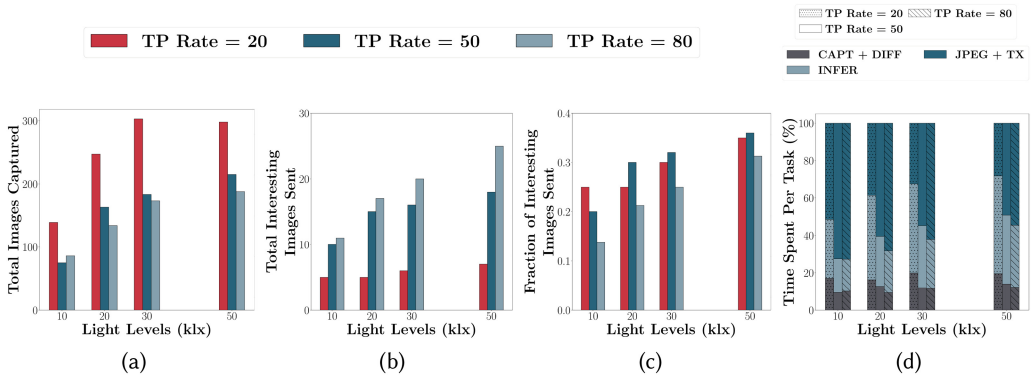


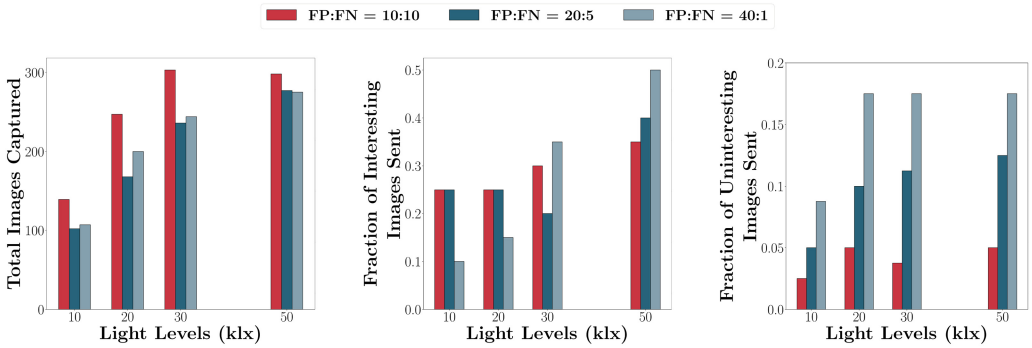
Fig. 8. Figures (a) to (d) show the performance of Camaroptera under different True Positive (TP) rates at different light levels. Here, Camaroptera operates in the *Local Inference* mode with a classifer that has a 10:10 False Positive:False Negative ratio. Changing the TP rate represents different amount of interesting events in the environment.

Inference mode either take $\leq$10*s* when eliminated by Image Differencing or take 10*s* to 20*s* when eliminated by the Inference module. Faster frame latencies allow the *Local Inference* mode to capture more total images, increasing the effectiveness of Camaroptera in detecting interesting events.

8.2 Effect of Varying Event Composition

We studied Camaroptera's sensitivity to varying the **True Positive (TP)** rate of images encountered across Figures 8(a) and 8(d). We restrict this study to Camaroptera's *Local Inference* mode with a DNN having a ratio of False Positives to False Negatives of 10:10. The trials for this sensitivity study had 100 image events, and the results were averaged across three repetitions of each trial. We varied the TP rate from 20% (few people) to 80% (crowded area), representing different amounts of interesting event traffic. The data shows that higher interesting event traffic (TP rate) degrades Camaroptera's ability to detect interesting events *only when input power is low*. We expected that a higher TP rate would be detrimental in the *Local Inference* mode, since it would require more energy-expensive image transmissions. This expectation holds true for total images captured, as seen in Figure 8(a), where the most images are captured for the lowest TP rate (20%). Figure 8(d) shows that as the TP rate increases, Camaroptera spends more time on image transmission and less on image capturing, resulting in fewer total images.

Figures 8(b) and 8(c) provide additional insights into how Camaroptera operates at different TP rates. As we expect, Camaroptera reports more total interesting events when there is a higher amount of interesting event traffic; however, the *fraction* of interesting events it reports varies. At 10 klx, energy efficiency matters most, and a low TP rate helps Camaroptera avoid activating the expensive radio. Camaroptera reports the largest fraction of interesting events for the 20% TP rate at 10 klx. At higher input power, a TP rate of 50% presents a higher fraction of interesting events. Camaroptera reporting a larger fraction of interesting events at 50% TP than 80% TP is expected; more interesting events use the radio more frequently, missing the capture of newer interesting events. When comparing against a 20% TP rate, Camaroptera misses consecutive interesting events due to its long processing latency. While Camaroptera misses consecutive event for all TP rates, this degrades performance the most at 20% TP rate (since there are few interesting events to begin with). Figure 8 shows that Camaroptera's ability to report interesting events is robust across a reasonable amount of event traffic (especially at 50%) and can be deployed across a variety of environments.



(a) Total images captured & processed (b) Interesting images sent (c) Uninteresting images sent

Fig. 9. Figures (a) to (c) show the performance of Camaroptera with different classifiers at different light levels, operating in the *Local Inference* mode. The different classifiers are represented by their False Positive:False Negative (FP:FN) ratio. Here, the True Positive rate for the environment is set to be 20%. Different classifier FP:FN ratios represent different design points for Camaroptera.

8.3 Effect of Varying DNN Parameters

We measured Camaroptera's sensitivity to variations in the False Positive and False Negative rate of its DNN classifier, evaluating its effectiveness with different learned inference models (Figures 9(a) to 9(c)). We evaluted three classifiers, representing differently tuned versions of the DNN in Section 6 with the same memory footprint. We identify a classifier by the ratio of its False Positives to its False Negatives, e.g., FP:FN = 10:10. Similar to Section 8.2, this sensitivity study was also conducted with trials comprising 100 image events, averaged across three repetitions.

Figures 9(a) and 9(b) show that a 10:10 classifier captures the most total images. However, the classifier that reports the largest fraction of interesting events depends on the input power. At low input power ($10 - 20klx$), a low FP rate (10%, 20%) leads to fewer uninteresting images transmitted (Figure 9(c)) than a 40% FP rate, preserving the limited available energy. Camaroptera uses this energy to capture and report a higher fraction of interesting events. At higher input power ($\geq 30klx$), energy is more abundant, and lowering the FN rate takes precedence; a 40:1 classifier reports the largest fraction of interesting events, even when its high FP rate leads to transmitting the most uninteresting events. Figure 9 shows that designers using Camaroptera must tune the classifier to match their application requirements (e.g., suffering higher FP for achieving lower FN), and also the deployment environment (e.g., prioritizing a lower FP when input power is low).

8.4 Device Characterization

We characterized and compared the lifetime of our energy-harvesting, batteryless Camaroptera system with different battery-powered Camaroptera systems, as well as the Permamote [34] system, in Table 2. Non-rechargeable batteries are a poor choice for a long-range, visual-sensing system like Camaroptera given their limited lifetimes of a few weeks. Rechargeable batteries allow Camaroptera to achieve longer lifetimes, but still require expensive battery replacements every 4–5 years. Permamote combines a rechargeable LTO battery with a non-rechargeable backup CR2477 battery for powering operation during periods of no input power (e.g., night time). While this does extend the operation to night time, we argue that this fits our Camaroptera system poorly for two reasons. First, the 5-year lifetime even with a rechargeable LTO battery requires expensive battery replacements, failing our requirement for a multi-decade lifetime. Second, a visual-sensing-based system like Camaroptera can capture useful images only when the scene is well-lit,

Table 2. Lifetime of Camaroptera with Different Power Systems

System	Capacity	Lifetime	Transmitted Frame Latency
CR2477	1,000 mAh	17.6 days	20.676 s
Non-rechargeable AA only	3,000 mAh	52.7 days	20.676 s
Rechargeable AA only	2,650 mAh	4–5 years[a]	20.676 s
Rechargeable LTO only	10 mAh	4.8 years[b]	20.676 s
Permamote [34] (CR2477 + LTO)	–	4.8 years + 17.6 days	20.676 s
CamHW	–	∞[c]	36–114 s (45–15 klx)

[a] Shelf-life limitations.
[b] Assuming 10,000 recharge cycles at half-depth discharge.
[c] Theoretically infinite, practical device lifetimes dictated by IC degradation.

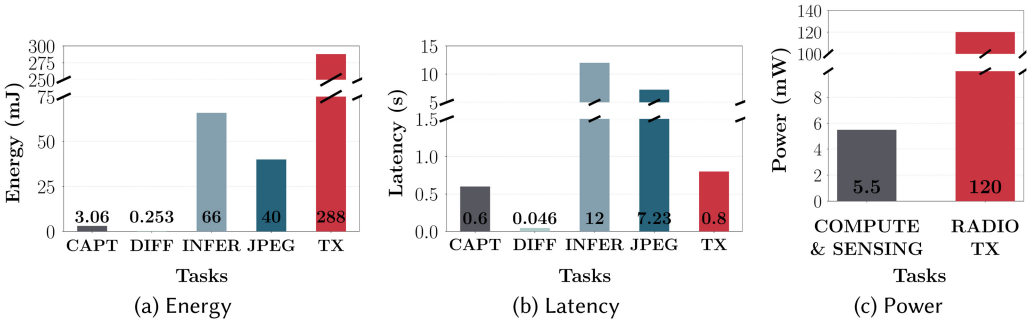


Fig. 10. Cost breakdown for performing different tasks on Camaroptera.

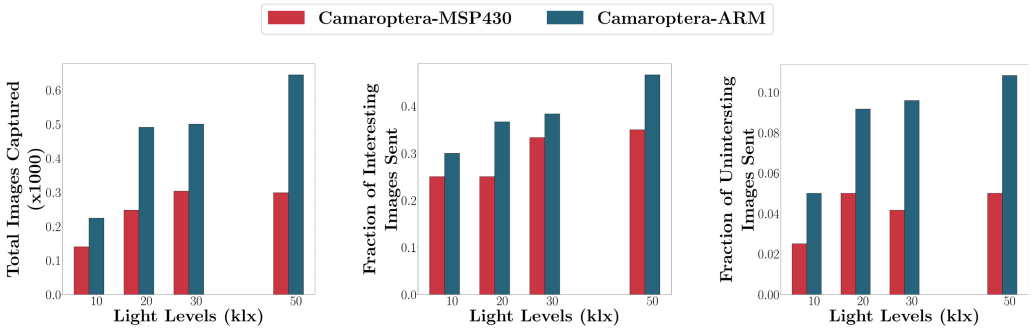
rendering night-time operation unnecessary, especially in remote areas where artificial lighting will be rare.

We also characterized Camaroptera's latency and energy for its main operations, with data in Figure 10. The data show that while inference takes the most time to complete, transmitting using LoRa consumes the most energy, because the radio system has much higher power consumption. With low input power, the high energy cost of transmitting requires a long recharging latency (Figure 2) despite the low transmission latency. With high input power, recharge times drop and the time spent computing exceeds the time spent transmitting *and* recharging.

9 FUTURE WORK

The less time Camaroptera spends between two subsequent image captures, the more images Camaroptera can capture and process, resulting in higher sensing effectiveness. We discuss a few ways to run Camaroptera processing pipeline faster, so that it captures and processes images more frequently.

More compute: Camaroptera performs computations on each image it captures, and running these computations faster will enable Camaroptera to capture a new image sooner. The most latency-intensive computations on Camaroptera are Inference and JPEG Compression, as shown in Figure 10(b). Optimizing the speed of these two computations will in turn reduce the time between two subsequent image captures. This can be achieved by employing special architectures, from DNN accelerators to dedicated DSP co-processors.



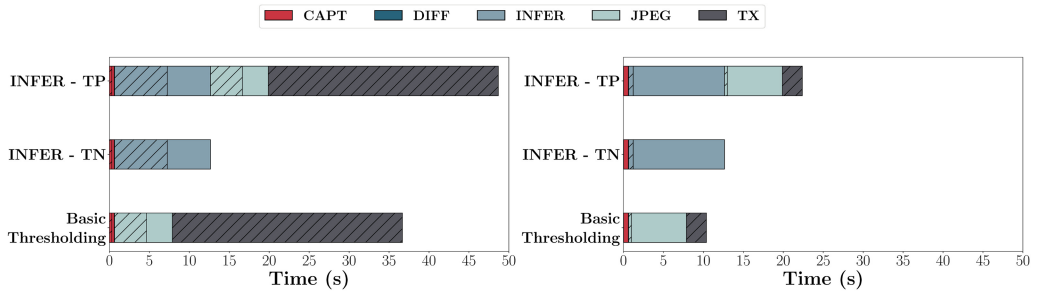
(a) Total images captured & processed (b) Interesting images sent (c) Uninteresting images sent

Fig. 11. Figures (a) to (c) show the performance of Camaroptera with our MSP-based design and using an ARM Cortex-M4 board to run the DNN. While using an ARM core to replace the MSP430 enables Camaroptera to perform better, the lack of FRAM support in commercial ARM cores justifies our choice of building Camaroptera around an MSP430 MCU.

We quantitatively show the benefits of faster compute by running the DNN in Camaroptera on an ARM Cortex-M4 core (NUCLEO-G474RE [67]). We ran experiments with the FP:FN = 10:10 classifier, with event emulation using the 20% True Positive rate, at four different light levels. For each experiment, Camaroptera ran everything but the DNN on the MSP430, and invoked the ARM core for running the DNN. Figure 11 shows that using the ARM core to accelerate the DNN computations enables Camaroptera to capture more interesting and total images. The ARM core provides higher energy efficiency than the MSP430, consuming $16.5mJ$ ($\sim$ $93mW$ for $178ms$) for running the DNN; the ARM core outperforms the MSP430 in terms of both energy and latency. However, commercial ARM cores lack support for byte-addressible, non-volatile memory like FRAM, justifying our MSP430-centric design of Camaroptera. We envision future revisions of Camaroptera to have new computational accelerators specific to the nature of deployed applications, which will enable Camaroptera to become a more effective sensor.

More power: Camaroptera has to spend time recharging the energy used for performing tasks, and improving energy-harvesting efficiency will reduce this recharging delay. The energy-harvesting efficiency depends on the size and efficiency of the solar cells and the efficiency of Camaroptera's power board. As solar cell technology matures and more efficient boosters become available, future revisions of Camaroptera can employ them to reduce their energy recharging latency, processing each image faster.

Adaptation: At high input power levels, Camaroptera processes each image faster in the *Basic Thresholding* mode, as compared to the *Local Inference* mode. Figures 12(a) and 12(b) show a breakdown of the per-image latency for Camaroptera operating in the *Basic Thresholding* and *Local Inference* modes, harvesting 30 klx and the maximum rated power of Camaroptera's solar cells, respectively. Each bar shows the time spent performing a different task for an image (from capture to transmission), including the time to recharge the energy required by the task; the shaded region shows the recharging latency. The breakdown for transmitting an interesting image (INFER-TP) and for discarding an uninteresting image (INFER-TN) in the *Local Inference* mode are shown separately. At 30 klx, it is faster to discard an uninteresting image using machine inference (INFER-TN) than transmitting it indiscriminately in the *Basic Thresholding* mode, since it is slow to recharge the energy required for image transmission. While INFER-TP (transmitting an interesting image) takes the longest total time, it is infrequent enough that Camaroptera operates faster in the *Local Inference* mode as compared to the *Basic Thresholding* mode (as shown by results in Section 8).



(a) 30 klx (b) Maximum rated power

Fig. 12. Latency breakdown for processing *events* in the *Local Inference* mode for an interesting event (INFER-TP), for an uninteresting event (INFER-TN), and in the *Basic Thresholding* mode, at (a) 30 klx and (b) the maximum rated power of Camaroptera's solar cells. The shaded regions show the portion of time spent recharging energy; a task is shaded entirely if recharging takes longer than execution. At common levels of input power (30 klx), the *Local Inference* mode outperforms the *Basic Thresholding* mode, as processing uninteresting images (INFER-TN) is faster in the *Local Inference* mode. In contrast, at the maximum rated input power, sending an image after *Basic Thresholding* is faster than running *Local Inference*, even for uninteresting images. However, this increased device speed comes at the cost of flooding the network with uninteresting images, representing an important design tradeoff.

Conversely, when harvesting the maximum rated power, operating in the *Local Inference* is always slower than the *Basic Thresholding* mode. At this high input power, recharging is quick and transmitting an entire image is faster than running inference. This presents another way to improve the overall speed of Camaroptera—by switching to the *Basic Thresholding* mode at very high input power levels.

However, this faster operation comes at the cost of a sharp increase in network traffic. At this input power level, with a 20% TP rate, the *Basic Thresholding* mode sends an image every ~13 seconds, while the *Local Inference* mode sends one every ~74 seconds. The *Local Inference* mode discards a majority of the uninteresting images, transmitting less frequently. Assuming an airtime of 0.33 ms per JPEG-compressed image, a time-multiplexed base station should be able to receive packets from close to 40 devices operating in *Basic Thresholding*. However, the same base station will be able to service more than 200 devices operating in the *Local Inference* mode. This translates to a 5× increase in the number of required base stations. Every base station will have to further expend resources to filter out the uninteresting images sent by the *Basic Thresholding* mode. Input-power-based switching presents a design tradeoff in terms of local device speed vs. network congestion, and the decision whether to employ this will depend on the specific deployment scenario.

10 RELATED WORK

There are several categories of work related to Camaroptera: batteryless remote sensing and communication systems, intermittent computing systems, and work on edge and near-sensor computing.

10.1 Batteryless Sensing and Communication

Prior work developed batteryless devices to sense their environment and transmit acquired data. The most related of these devices are batteryless image sensing systems [57, 58] described in Section 3.1. These systems rely on backscatter communication, requiring heavy RF infrastructure and are not appropriate for pervasive, wide-spread deployment, which is the motivation of Camaroptera. A recent system [22] presents a batteryless image sensing system for face

recognition. While using a similar camera and radio to Camaroptera, its architecture and deployment targets are entirely different. It targets indoor face detection, resulting in a different power system design with significantly (roughly 6×) larger solar panels and uses a time-of-flight sensor to trigger image captures. More importantly, its computing subsystem is designed around an ARM Cortex-M4 core, which does not support byte-addressable non-volatile memory, making this system unsuitable for long-lifetime, energy-harvesting applications. Other batteryless systems [9, 11, 25, 27, 36, 63, 79, 80] are designed to collect low-data-rate time series data and do not support image collection. Additionally, by relying on backscatter, **Bluetooth Low Energy (BLE)**, or other short range communciation media, these devices do not communicate over long distances, making them inapplicable to Camaroptera.

Other prior work on batteryless communication relates to Camaroptera. Some work uses active radios [8, 11, 20] focusing primarily on BLE. We are unaware at the time of writing of any batteryless device supporting active LoRa communication, making Camaroptera the first device with this capability. NeoFOG [46] relies on high-powered radios and focuses on optimizing sensor-to-sensor communication, not long-range backhaul. Other work uses passive communication, primarily based on RF backscattering. Some work aims at short-range backscattering of directed or ambient RF energy [37, 38, 63, 73, 76], which are inapplicable to the demands of wide-spread sensor deployment. Longer-range passive systems [59, 69, 75] extend the range of passive networking, but to insufficient range (tens of meters) [75] and with dependence on large, powered RF transmitters. Camaroptera has the full range of LoRa (hundreds of meters to kilometers) and requires only inexpensive receivers, not complex, high-power RF power transmitters.

10.2 Intermittent Computing and DNNs

Recent work improved the computational capability of batteryless devices with software and hardware models for intermittent computing. Most relevant is recent work on bringing deep learning to energy-harvesting devices [23]; Camaroptera directly leverages this work for its DNN hyper-parameter optimization. Binary networks [13, 41] and bit-serial [17] approaches are an appealing alternative option for DNNs on intermittent devices. Other work on intermittent computing focused on correctness using software support for tasks [10, 28, 48, 62, 77] and checkpoints [5, 6, 35, 49, 50, 61, 72], some with support for approximation [21, 47]. Other systems provide hardware support for designing intermittent architectures [29, 54], circuits [55], and platforms [11, 35]. Some work on intermittent computing targets the safety of I/O operations [3, 7] and concurrency [62, 77].

Camaroptera is largely orthogonal to this prior work on intermittent computing because, while batteryless, Camaroptera avoids intermittent operation by provisioning its capacitive energy storage for communication, which is an order of magnitude larger than what computing requires. Consequently, Camaroptera avoids unpredictable intermittent operation, instead Camaroptera is designed to have sufficient stored energy before attempting any computation.

10.3 Edge Computing

Prior work on edge computing has studied early image discard for constrained image sensing systems, similar to Camaroptera's processing pipelines. Systems like WULoRa [51] use a secondary wake-up receiver to trigger a sensing task on the LoRa-based sensing device, whereas Camaroptera triggers its LoRa radio upon detecting interesting events in its environment. Edge systems process image and video data [32, 68, 74] efficiently on inelastic deployed resources. Camaroptera differs in scale from most prior edge systems (with at-sensor computing being "beyond the edge," according to prior work [23]). Edge computing on larger, yet inelastic systems represens an important future research topic. We assume Camaroptera's base stations are cheap

receivers, but in the future Camaroptera base stations could include sophisticated edge computing resources. Managing the division of labor between Camaroptera-scale sensor nodes, larger, base-station-scale edge computing resources, and clouds is an open problem.

11 CONCLUSION

We presented Camaroptera, the first batteryless image sensor with the ability to communicate over extremely long distances using an active LoRa radio. Camaroptera is designed to reduce the high cost of long-range communication by processing data locally, using at-sensor processing pipelines. Our fully built hardware and software prototype supports all of Camaroptera's capabilities in a compact form factor. Our evaluation showed that employing *Local Inference* on Camaroptera allows it to send upto 12× more interesting images, while reducing the uninteresting images sent by upto 6.5×, as compared to a traditional *Sense-and-Send* approach. This also allows Camaroptera to capture upto 14.7× more total images, increasing Camaroptera's effectiveness as a sensor. Future Camaroptera iterations can employ novel architectures to reduce computational costs, further improving its availability in real-world deployments.

ACKNOWLEDGMENTS

This work was supported in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

REFERENCES

[1] Joshua Adkins, Branden Ghena, Neal Jackson, Pat Pannuto, Samuel Rohrer, Bradford Campbell, and Prabal Dutta. 2018. The signpost platform for city-scale sensing. In *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'18)*. IEEE Press, 188–199. DOI:https://doi.org/10.1109/IPSN.2018.00047

[2] Amazon. Amazon Mechanical Turk. Retrieved from https://www.mturk.com/.

[3] A. Arreola, D. Balsamo, G. Merrett, and A. Weddell. 2018. RESTOP: Retaining external peripheral state in intermittently-powered sensor systems. *Sensors (Basel)* 18, 1 (Jan. 2018), 172. DOI:https://doi.org/10.3390/s18010172

[4] AVX. 2019. BestCap®Ultra-low ESR High Power Pulse Supercapacitors. Retrieved from http://catalogs.avx.com/BestCap.pdf.

[5] D. Balsamo, A. S. Weddell, A. Das, A. R. Arreola, D. Brunelli, B. M. Al-Hashimi, G. V. Merrett, and L. Benini. 2016. Hibernus++: A self-calibrating and adaptive system for transiently-powered embedded devices. *IEEE Trans. Comput.-Aid. Des. Integ. Circ. Syst.* 35, 12 (2016), 1968–1980. DOI:https://doi.org/10.1109/TCAD.2016.2547919

[6] D. Balsamo, A. S. Weddell, G. V. Merrett, B. M. Al-Hashimi, D. Brunelli, and L. Benini. 2015. Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems. *IEEE Embed. Syst. Lett.* 7, 1 (Mar. 2015), 15–18. DOI:https://doi.org/10.1109/LES.2014.2371494

[7] G. Berthou, T. Delizy, K. Marquet, T. Risset, and G. Salagnac. 2019. Sytare: A lightweight kernel for NVRAM-Based transiently-powered systems. *IEEE Trans. Comput.* 68, 9 (Sep. 2019), 1390–1403. DOI:https://doi.org/10.1109/TC.2018.2889080

[8] Bradford Campbell, Joshua Adkins, and Prabal Dutta. 2016. Cinamin: A perpetual and nearly invisible BLE beacon. In *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN'16)*. Junction Publishing, USA, 331–332. http://dl.acm.org/citation.cfm?id=2893711.2893793

[9] B. Campbell, M. Clark, S. DeBruin, B. Ghena, N. Jackson, Y. Kuo, and P. Dutta. 2016. Perpetual sensing for the built environment. *IEEE Pervas. Comput.* 15, 4 (Oct. 2016), 45–55. DOI:https://doi.org/10.1109/MPRV.2016.66

[10] Alexei Colin and Brandon Lucia. 2016. Chain: Tasks and channels for reliable intermittent programs. In *Proceedings of the ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'16)*. ACM, New York, NY, 514–530. DOI:https://doi.org/10.1145/2983990.2983995

[11] Alexei Colin, Emily Ruppel, and Brandon Lucia. 2018. A reconfigurable energy storage architecture for energy-harvesting devices. In *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'18)*. ACM, New York, NY, 767–781. DOI:https://doi.org/10.1145/3173162.3173210

[12] Council of Economic Advisors to President Barack Obama. Srengthening rural infrastructure. 2010. Retrieved from https://obamawhitehouse.archives.gov/administration/eop/cea/factsheets-reports/strengthening-the-rural-economy/strengthening-rural-infrastructure.

[13] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. BinaryConnect: Training deep neural networks with binary weights during propagations. *CoRR* abs/1511.00363 (2015).

[14] W. J. Dally, J. Balfour, D. Black-Shaffer, J. Chen, R. C. Harting, V. Parikh, J. Park, and D. Sheffield. 2008. Efficient embedded computing. *Computer* 41, 7 (July 2008), 27–32. DOI:https://doi.org/10.1109/MC.2008.224

[15] Bradley Denby and Brandon Lucia. 2020. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'20)*. Association for Computing Machinery, New York, NY, 939–954. DOI:https://doi.org/10.1145/3373376.3378473

[16] A. Dongare, C. Hesling, K. Bhatia, A. Balanuta, R. L. Pereira, B. Iannucci, and A. Rowe. 2017. OpenChirp: A low-power wide-area networking architecture. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops'17)*. 569–574. DOI:https://doi.org/10.1109/PERCOMW.2017.7917625

[17] Charles Eckert, Xiaowei Wang, Jingcheng Wang, Arun Subramaniyan, Ravi Iyer, Dennis Sylvester, David Blaauw, and Reetuparna Das. 2018. Neural cache: Bit-serial in-cache acceleration of deep neural networks. In *Proceedings of the 45th Annual International Symposium on Computer Architecture (ISCA'18)*. IEEE Press, Piscataway, NJ, 383–396. DOI:https://doi.org/10.1109/ISCA.2018.00040

[18] Rashad Eletreby, Diana Zhang, Swarun Kumar, and Osman Yağan. 2017. Empowering low-power wide area networks in urban settings. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'17)*. Association for Computing Machinery, New York, NY, 309–321. DOI:https://doi.org/10.1145/3098822.3098845

[19] J. F. Ensworth and M. S. Reynolds. 2017. BLE-Backscatter: Ultralow-power IoT nodes compatible with Bluetooth 4.0 low energy (BLE) smartphones and tablets. *IEEE Trans. Microw. Theor. Techniq.* 65, 9 (Sep. 2017), 3360–3368. DOI:https://doi.org/10.1109/TMTT.2017.2687866

[20] Francesco Fraternali, Bharathan Balaji, Yuvraj Agarwal, Luca Benini, and Rajesh Gupta. 2018. Pible: Battery-free mote for perpetual indoor BLE applications. In *Proceedings of the 5th Conference on Systems for Built Environments (BuildSys'18)*. ACM, New York, NY, 168–171. DOI:https://doi.org/10.1145/3276774.3282822

[21] K. Ganesan, J. San Miguel, and N. Enright Jerger. 2019. The what's next intermittent computing architecture. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA'19)*. 211–223. DOI:https://doi.org/10.1109/HPCA.2019.00039

[22] Marco Giordano, Philipp Mayer, and Michele Magno. 2020. A battery-free long-range wireless smart camera for face detection. In *Proceedings of the 8th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems (ENSsys'20)*. Association for Computing Machinery, New York, NY, 29–35. DOI:https://doi.org/10.1145/3417308.3430273

[23] Graham Gobieski, Brandon Lucia, and Nathan Beckmann. 2019. Intelligence beyond the edge: Inference on intermittent embedded systems. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'19)*. ACM, New York, NY, 199–213. DOI:https://doi.org/10.1145/3297858.3304011

[24] Graham Gobieski, Amolak Nagi, Nathan Serafin, Mehmet Meric Isgenc, Nathan Beckmann, and Brandon Lucia. 2019. MANIC: An energy-efficient, parallel architecture for ultra-low-power embedded systems. In *Proceedings of the 52nd IEEE/ACM International Symposium on Microarchitecture (MICRO'19)*.

[25] Josiah Hester, Travis Peters, Tianlong Yun, Ronald Peterson, Joseph Skinner, Bhargav Golla, Kevin Storer, Steven Hearndon, Kevin Freeman, Sarah Lord, Ryan Halter, David Kotz, and Jacob Sorber. 2016. Amulet: An energy-efficient, multi-application wearable platform. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM (SenSys'16)*. ACM, New York, NY, 216–229. DOI:https://doi.org/10.1145/2994551.2994554

[26] Josiah Hester, Lanny Sitanayah, and Jacob Sorber. 2015. Tragedy of the coulombs: Federating energy storage for tiny, intermittently-powered sensors. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys'15)*. Association for Computing Machinery, New York, NY, 5–16. DOI:https://doi.org/10.1145/2809695.2809707

[27] Josiah Hester and Jacob Sorber. 2017. Flicker: Rapid prototyping for the batteryless internet-of-things. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems (SenSys'17)*. ACM, New York, NY. DOI:https://doi.org/10.1145/3131672.3131674

[28] Josiah Hester, Kevin Storer, and Jacob Sorber. 2017. Timely execution on intermittently powered batteryless sensors. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems (SenSys'17)*. ACM, New York, NY. DOI:https://doi.org/10.1145/3131672.3131673

[29] M. Hicks. 2017. Clank: Architectural support for intermittent computation. In *Proceedings of the ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA'17)*. 228–240. DOI:https://doi.org/10.1145/3079856.3080238

[30] Himax Technologies, Inc. 2018. HM01B0 Ultra Low Power camera sensor. Retrieved from https://github.com/cjosephson/backcam/blob/master/hardware/datasheets/HM01B0_DS_preliminary_v06.pdf.

[31] HopeRF. 2019. RFM95W LoRa Module. Retrieved from https://www.hoperf.com/data/upload/portal/20190801/RFM95W-V2.0.pdf.

[32] C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, and M. Philipose. 2018. VideoEdge: Processing camera streams using hierarchical clusters. In *Proceedings of the IEEE/ACM Symposium on Edge Computing (SEC'18)*. 115–131. DOI:https://doi.org/10.1109/SEC.2018.00016

[33] IXYS. 2016. IXOLAR High Efficiency SolarBIT. Retrieved from http://ixapps.ixys.com/DataSheet/KXOB22-01X8F_Nov16.pdf.

[34] Neal Jackson, Joshua Adkins, and Prabal Dutta. 2019. Capacity over capacitance for reliable energy harvesting sensors. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks (IPSN'19)*. ACM, New York, NY, 193–204. DOI:https://doi.org/10.1145/3302506.3310400

[35] H. Jayakumar, A. Raha, and V. Raghunathan. 2014. QUICKRECALL: A low overhead HW/SW approach for enabling computations across power cycles in transiently powered computers. In *Proceedings of the 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems*. 330–335. DOI:https://doi.org/10.1109/VLSID.2014.63

[36] Mustafa Emre Karagozler, Ivan Poupyrev, Gary K. Fedder, and Yuri Suzuki. 2013. Paper generators: Harvesting energy from touching, rubbing and sliding. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST'13)*. ACM, New York, NY, 23–30. DOI:https://doi.org/10.1145/2501988.2502054

[37] Bryce Kellogg, Aaron Parks, Shyamnath Gollakota, Joshua R. Smith, and David Wetherall. 2014. Wi-fi backscatter: Internet connectivity for RF-powered devices. In *Proceedings of the ACM Conference on SIGCOMM (SIGCOMM'14)*. ACM, New York, NY, 607–618. DOI:https://doi.org/10.1145/2619239.2626319

[38] Bryce Kellogg, Vamsi Talla, Shyamnath Gollakota, and Joshua R. Smith. 2016. Passive Wi-Fi: Bringing low power to Wi-Fi transmissions. In *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI'16)*. USENIX Association, 151–164. Retrieved from https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/kellogg.

[39] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-Based learning applied to document recognition. *Proc. IEEE* 86, 11 (Nov. 1998), 2278–2324.

[40] Jansen C. Liando, Amalinda Gamage, Agustinus W. Tengourtius, and Mo Li. 2019. Known and unknown facts of LoRa: Experiences from a large-scale measurement study. *ACM Trans. Sen. Netw.* 15, 2 (Feb. 2019). DOI:https://doi.org/10.1145/3293534

[41] Jeng-Hau Lin, Yunfan Yang, Rajesh K. Gupta, and Zhuowen Tu. 2018. Local binary pattern networks. *CoRR* abs/1803.07125 (2018).

[42] Linear Technology. 2010. LTC3105 DC/DC converter. Retrieved from https://www.analog.com/media/en/technical-documentation/data-sheets/3105fb.pdf.

[43] Ting Liu, Christopher M. Sadler, Pei Zhang, and Margaret Martonosi. 2004. Implementing software on resource-constrained mobile sensors: Experiences with Impala and ZebraNet. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys'04)*. ACM, New York, NY, 256–269. DOI:https://doi.org/10.1145/990064.990095

[44] Brandon Lucia, Vignesh Balaji, Alexei Colin, Kiwan Maeng, and Emily Ruppel. 2017. Intermittent computing: Challenges and opportunities. In *Proceedings of the 2nd Summit on Advances in Programming Languages (SNAPL'17) (Leibniz International Proceedings in Informatics (LIPIcs))*, Benjamin S. Lerner, Rastislav Bodík, and Shriram Krishnamurthi (Eds.), Vol. 71. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 8:1–8:14. DOI:https://doi.org/10.4230/LIPIcs.SNAPL.2017.8

[45] Brandon Lucia and Benjamin Ransford. 2015. A simpler, safer programming and execution model for intermittent systems. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'15)*. ACM, New York, NY, 575–585. DOI:https://doi.org/10.1145/2737924.2737978

[46] Kaisheng Ma, Xueqing Li, Mahmut Taylan Kandemir, Jack Sampson, Vijaykrishnan Narayanan, Jinyang Li, Tongda Wu, Zhibo Wang, Yongpan Liu, and Yuan Xie. 2018. NEOFog: Nonvolatility-exploiting optimizations for fog computing. *SIGPLAN Not.* 53, 2 (Mar. 2018), 782–796. DOI:https://doi.org/10.1145/3296957.3177154

[47] Kaisheng Ma, Xueqing Li, Jinyang Li, Yongpan Liu, Yuan Xie, Jack Sampson, Mahmut Taylan Kandemir, and Vijaykrishnan Narayanan. 2017. Incidental computing on IoT nonvolatile processors. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'17)*. ACM, New York, NY, 204–218. DOI:https://doi.org/10.1145/3123939.3124533

[48] Kiwan Maeng, Alexei Colin, and Brandon Lucia. 2017. Alpaca: Intermittent execution without checkpoints. *Proc. ACM Program. Lang.* 1, OOPSLA (Oct. 2017). DOI:https://doi.org/10.1145/3133920

[49] Kiwan Maeng and Brandon Lucia. 2018. Adaptive dynamic checkpointing for safe efficient intermittent computing. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI'18)*. USENIX Association, 129–144. Retrieved from https://www.usenix.org/conference/osdi18/presentation/maeng.

[50] Kiwan Maeng and Brandon Lucia. 2019. Supporting peripherals in intermittent systems with just-in-time checkpoints. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'19)*. ACM, New York, NY, 1101–1116. DOI:https://doi.org/10.1145/3314221.3314613

[51] Michele Magno, Faycal Ait Aoudia, Matthieu Gautier, Olivier Berder, and Luca Benini. 2017. WULoRa: An energy efficient IoT end-node for energy harvesting and heterogeneous communication. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE'17)*. 1528–1533. DOI:https://doi.org/10.23919/DATE.2017.7927233

[52] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. 2019. A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express* 5, 1 (2019), 1–7. DOI:https://doi.org/10.1016/j.icte.2017.12.005

[53] Microchip. 2017. MIC841 Comparator with Reference and Adjustable Hysteresis. Retrieved from http://ww1.microchip.com/downloads/en/devicedoc/20005758a.pdf.

[54] Joshua San Miguel, Karthik Ganesan, Mario Badr, Chunqiu Xia, Rose Li, Hsuan Hsiao, and Natalie Enright Jerger. 2018. The EH model: Early design space exploration of intermittent processor architectures. In *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'18)*. IEEE Press, Piscataway, NJ, 600–612. DOI:https://doi.org/10.1109/MICRO.2018.00055

[55] A. Mirhoseini, E. M. Songhori, and F. Koushanfar. 2013. Idetic: A high-level synthesis approach for enabling long computations on transiently-powered ASICs. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom'13)*. 216–224. DOI:https://doi.org/10.1109/PerCom.2013.6526735

[56] Moodstocks. 2016. jpec - a JPEG encoder in C. Retrieved from https://github.com/Moodstocks/jpec.

[57] Saman Naderiparizi, Mehrdad Hessar, Vamsi Talla, Shyamnath Gollakota, and Joshua R. Smith. 2018. Towards battery-free HD video streaming. In *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI'18)*. USENIX Association, 233–247. Retrieved from https://www.usenix.org/conference/nsdi18/presentation/naderiparizi.

[58] S. Naderiparizi, A. N. Parks, Z. Kapetanovic, B. Ransford, and J. R. Smith. 2015. WISPCam: A battery-free RFID camera. In *Proceedings of the IEEE International Conference on RFID (RFID'15)*. 166–173. DOI:https://doi.org/10.1109/RFID.2015.7113088

[59] Yao Peng, Longfei Shangguan, Yue Hu, Yujie Qian, Xianshang Lin, Xiaojiang Chen, Dingyi Fang, and Kyle Jamieson. 2018. PLoRa: A passive long-range data network from ambient LoRa transmissions. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'18)*. ACM, New York, NY, 147–160. DOI:https://doi.org/10.1145/3230543.3230567

[60] Phillip Sparks. 2017. The route to a trillion devices: The outlook to IoT investment to 2035. Retrieved from https://learn.arm.com/rs/714-XIJ-402/images/Arm-The-route-to-trillion-devices_2018.pdf.

[61] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 2011. Mementos: System support for long-running computation on RFID-scale devices. In *Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'11)*. ACM, New York, NY, 159–170. DOI:https://doi.org/10.1145/1950365.1950386

[62] Emily Ruppel and Brandon Lucia. 2019. Transactional concurrency control for intermittent, energy-harvesting computing systems. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'19)*. ACM, New York, NY, 1085–1100. DOI:https://doi.org/10.1145/3314221.3314583

[63] Alanson P. Sample, Daniel J. Yeager, Pauline S. Powledge, Alexander V. Mamishev, and Joshua R. Smith. 2008. Design of an RFID-based battery-free programmable sensing platform. *IEEE Trans. Instrum. Meas.* 57, 11 (2008), 2608–2615.

[64] Semtech. 2018. LoRa™Modulation Basics. Retrieved from https://www.semtech.com/uploads/documents/an1200.22.pdf.

[65] Semtech. 2019. SX127x transceivers Datasheet. Retrieved from https://www.semtech.com/uploads/documents/DS_SX1276-7-8-9_W_APP_V6.pdf.

[66] P. Stanley-Marbell and M. Rinard. 2020. Warp: A hardware platform for efficient multimodal sensing with adaptive approximation. *IEEE Micro* 40, 1 (2020), 57–66. DOI:https://doi.org/10.1109/MM.2019.2951004

[67] STMicroelectronics. 2014. NUCLEO-G474RE. Retrieved from https://www.st.com/resource/en/data_brief/nucleo-g474re.pdf.

[68] Christopher Streiffer, Animesh Srivastava, Victor Orlikowski, Yesenia Velasco, Vincentius Martin, Nisarg Raval, Ashwin Machanavajjhala, and Landon P. Cox. 2017. ePrivateeye: To the edge and beyond! In *Proceedings of the 2nd ACM/IEEE Symposium on Edge Computing (SEC'17)*. ACM, New York, NY. DOI:https://doi.org/10.1145/3132211.3134457

[69] Vamsi Talla, Mehrdad Hessar, Bryce Kellogg, Ali Najafi, Joshua R. Smith, and Shyamnath Gollakota. 2017. LoRa backscatter: Enabling the vision of ubiquitous connectivity. *Proc. ACM Interact. Mob. Wear. Ubiq. Technol.* 1, 3 (Sept. 2017). DOI:https://doi.org/10.1145/3130970

[70] Texas Instruments. 2015. TPS61070 s Boost Converter. Retrieved from http://www.ti.com/lit/ds/symlink/tps61070.pdf.

[71] Texas Instruments. 2018. MSP430FR599 Mixed-Signal Microcontrollers datasheet. Retrieved from http://www.ti.com/lit/ds/symlink/msp430fr5994.pdf.

[72] Joel Van Der Woude and Matthew Hicks. 2016. Intermittent computation without hardware support or programmer intervention. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI'16)*. USENIX Association, Berkeley, CA, 17–32. Retrieved from http://dl.acm.org/citation.cfm?id=3026877.3026880.

[73] Anran Wang, Vikram Iyer, Vamsi Talla, Joshua R. Smith, and Shyamnath Gollakota. 2017. FM backscatter: Enabling connected cities and smart fabrics. In *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI'17)*. USENIX Association, Boston, MA, 243–258. Retrieved from https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/wang-anran.

[74] J. Wang, Z. Feng, Z. Chen, S. George, M. Bala, P. Pillai, S. Yang, and M. Satyanarayanan. 2018. Bandwidth-efficient live video analytics for drones via edge computing. In *Proceedings of the IEEE/ACM Symposium on Edge Computing (SEC'18)*. 159–173. DOI:https://doi.org/10.1109/SEC.2018.00019

[75] Jingxian Wang, Junbo Zhang, Rajarshi Saha, Haojian Jin, and Swarun Kumar. 2019. Pushing the range limits of commercial passive RFIDs. In *Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI'19)*. USENIX Association, Boston, MA, 301–316. Retrieved from https://www.usenix.org/conference/nsdi19/presentation/wangjingxian.

[76] C. Yang, J. Gummeson, and A. Sample. 2017. Riding the airways: Ultra-wideband ambient backscatter via commercial broadcast systems. In *Proceedings of the IEEE Conference on Computer Communications*. 1–9. DOI:https://doi.org/10.1109/INFOCOM.2017.8057162

[77] Kasım Sinan Yıldırım, Amjad Yousef Majid, Dimitris Patoukas, Koen Schaper, Przemyslaw Pawelczak, and Josiah Hester. 2018. Ink: Reactive kernel for tiny batteryless sensors. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*. ACM, 41–53.

[78] Zac Manchester. 2015. KickSat. Retrieved from http://zacinaction.github.io/kicksat/.

[79] Hong Zhang, Jeremy Gummeson, Benjamin Ransford, and Kevin Fu. 2011. *Moo: A Batteryless Computational RFID and Sensing Platform.* Technical Report. Department of Computer Science, University of Massachusetts Amherst.

[80] Yi Zhao, Joshua R. Smith, and Alanson Sample. 2015. NFC-WISP: An open source software defined near field RFID sensing platform. In *Adjunct Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the ACM International Symposium on Wearable Computers (UbiComp/ISWC'15 Adjunct)*. ACM, New York, NY, 369–372. DOI:https://doi.org/10.1145/2800835.2800912

Received January 2021; revised December 2021; accepted January 2022