Embedded Sensing through Energy Harvesting at the Mithræum of Circus Maximus

Mikhail Afanasov*, Naveed Anwar Bhatti*, Koustabh Dolui*+ and Luca Mottola*†‡
*Politecnico di Milano (Italy), †RI.SE Sweden, ‡Uppsala University (Sweden), +KU Leuven (Belgium)

ABSTRACT

We present the design and evaluation of an embedded sensing deployment at the *Mithræum of Circus Maximus*, an underground archaeological site in Rome (Italy). Unique to our deployment is the use of energy harvesting through a combination of thermal and kinetic energy sources. The extreme scarcity of energy, however, poses great challenges in energy management, embedded hardware, and system software. We tackle them by testing existing solutions from areas such as energy harvesting, low-power hardware, and intermittent computing. We thus demonstrate the efficient performance of a hardware/software co-design providing accurate energy management and capturing the coupling between energy sources and sensed quantities. Installing a battery-operated system alongside also allows us to perform a comparative analysis of energy harvesting in such a demanding setting. Albeit energy harvesting reduces energy availability and thus lowers the *data yield* to about 22% of that provided by batteries, our hardware/software co-design allows the system to provide a comparable *level of insight* into environmental conditions and structural health of the site.

1 INTRODUCTION

Ambient energy harvesting is progressively enabling battery-less embedded sensing. A variety of harvesting techniques now exist that apply to, for example, light, vibrations, and thermal phenomena [11]. These technologies are naturally attractive wherever replacing batteries is unfeasible or impractical, and represent a foundation to achieve *zero-maintenance* embedded sensing [45].

Real-world deployments. Besides systems that use solar radiation as energy source, few examples exist of long-term deployments demonstrating energy-harvesting zero-maintenance systems [17, 18, 39], as we discuss in Sec. 2. The longest-running such deployment is reported to be operational for 3 months [17]. Further, very few of these deployments serve the needs of actual end users; rather, they are most often instrumental to demonstrate isolated software, hardware, or energy harvesting techniques. We argue that the limited span and scope of such real-world experiences is a sign that current technology is not ready for prime time, as a complete-system perspective is sorely missing.

This paper is about our first-hand experience of such state of affairs, specific to an embedded sensing deployment at the *Mithræum of Circus Maximus*, an archaeological site in Rome (Italy). Such an effort is motivated by the need to understand environmental and structural conditions of the site, as we illustrate in Sec. 3. The site, shown in Fig. 1, is generally closed to the public, completely underground, and only accessible through staircases and ladders.

Our work. Our deployment unfolds through two distinct phases. The first design iteration, illustrated in Sec. 4, is largely based on off-the-shelf components and operates with batteries. We use a commercial platform coupled with acceleration, inclination, temperature, and relative humidity sensors, along with a sub-GHz radio. Despite its satisfactory performance during operational times, its reliability is limited, mainly because of batteries. Due to the difficulties to access the site to replace them, this renders the system impractical. We thus eventually turn to energy harvesting. Besides making battery replacement a hurdle, however, the site characteristics rule out most of the energy-rich sources, including light.

(a) Mitra altar. (b) Concrete columns.

Figure 1: *Mithræum of Circus Maximus* in Rome, Italy. *The site is underground and only accessible through spiral staircases and provisional ladders.*

In the second design iteration, discussed in Sec. 5, we rely on thermal and kinetic sources, harvesting energy from temperature gradients and structural vibrations. We do not expect to achieve energy-neutral operation [5, 57], and design the system as an intermittently-executing one [35]. Intermittent executions interleave periods of active operation with periods of solely recharging of energy buffers. This design is eventually rooted in two key observations, namely *i)* a hardware/software co-design is required to efficiently manage the little available energy, and *ii)* in our deployment, a form of coupling exists between energy sources and sensed quantities [18, 56]. We make the former concrete through dedicated hardware designs that tightly integrate with program structure and execution model. As for the latter, we capitalize on structural vibrations representing both the energy source and the data we sense.

Outcomes. We report on site-specific insights from sensed data and on system performance in Sec. 6. We show, for example, that relative humidity levels easily cross 90% in a 21C°-25C° temperature range. We also analyze the performance trade-offs through the two design iterations and compare energy harvesting to battery-powered operation. We specifically show that in the same conditions, energy harvesting reduces energy availability and thus lowers the system's data yield to about 22% of that provided by batteries, but ways exist to retain the quality of collected data.

In Sec. 7, we discuss limitations of our experience and provide directions for further work in the area. Sec. 8 ends the paper with brief concluding remarks.

2 BACKGROUND AND RELATED WORK

Our work touches upon several different areas. In the following, we discuss the relation of our efforts to those works we deem closer in terms of goals, experiences, and outcomes.

2.1 Deployments

A rich body of literature exists on deploying battery-powered embedded sensing systems at different scales and in various environments [9, 14–16, 24, 28, 38, 44, 55, 58, 60, 72]. Common to these efforts are the many sources of unreliable operation and the hectic experience with frequent battery replacements. Lessons from these works help us swiftly set up a fully functional first design, but despite decades of research, limited and unpredictable battery lifetime remains the root cause of malfunction.

Various works demonstrate prolonged lifetime using rechargeable batteries backed by solar [1, 18, 21, 25, 40, 56, 59] or sometimes kinetic and thermal energy harvesting [18, 56]. The longest such deployment is understandably based on solar, and demonstrates a 2-year uninterrupted operation [21]. In contrast, deployments based on thermal [56] and kinetic [18] energy harvesting are limited in lifespan, extending up to four weeks [56]. Our deployment location is void of solar energy, mandating the use of lower-energy sources like thermal and kinetic.

Fewer examples exist replacing rechargeable batteries with environment-friendly super-capacitors [17, 30, 40, 47, 62] or regular capacitors [39, 65, 67–69] to buffer energy and smoothen harvesting fluctuations. Again, only a fraction of these works consider energy sources other than solar [47, 65, 67, 69], let alone real-world deployments [17, 39]. The longest such deployment uses microbial fuel cells to power nodes for water quality monitoring for three months [17]. Although these efforts communicate invaluable lessons on specific techniques, they provide no evidence of a complete system design. Similarly, only a few of them concretely fulfills the requirements of real end users [17, 39], unlike we do here.

2.2 System Support

Limited form factors impose restrictions on the physical dimensions of the harvesting unit, limiting power supply to tens of *mW* [17, 30, 40]. This often creates a demand-supply gap between harvested and required energy, which is typically tackled through specialized system support. Two approaches are possible: *energy-neutral system design* and *intermittent computing*.

Energy-neutral systems. The key idea is to aggressively tune system performance to achieve a demand-supply balance, thus enabling continuous operation [5, 6, 29, 61, 70, 71]. A broad range of hardware and software optimizations exist to improve energy generation or reduce its consumption, such as maximum power point tracking (MPPT) [6, 67], variable duty-cycling [61, 70, 71] and dynamic voltage and frequency scaling (DVFS) [5].

Techniques for energy neutrality, however, tend to cap the system performance. As a result, such an approach squeezes the set of feasible applications. Energy neutrality, moreover, may simply not be feasible whenever the average input power is generally lower than the minimum system requirement. This is precisely our setting, where the thermal and kinetic sources offer an insufficient energy content to even conceive continuous operation.

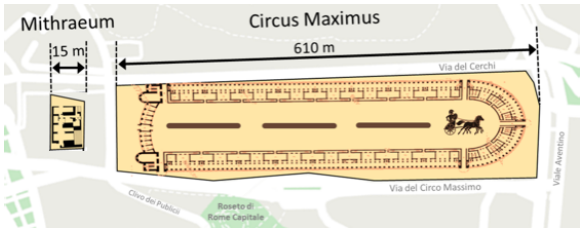


Figure 2: *Mithræum* location relative to the *Circus Maximus*.

Intermittent computing. As opposed to energy-neutral system design, intermittent computing allows energy to buffer for performing operations whose consumption may exceed the maximum power harvested at once. As a result, executions become intermittent [35], as periods of active operation are interspersed with periods only dedicated to recharging energy buffers, while the rest of the system is quiescent.

Intermittent systems typically employ techniques such as checkpointing [3, 7, 8, 12, 41, 52, 53, 64, 76] or task-based programming abstractions [19, 49, 51, 54, 66, 78] to recover from power failures. The former consist in replicating the application state on non-volatile memory, where it is retrieved back once the system resumes with sufficient energy. The latter target mixed-volatile platforms and offer abstractions that programmers use to define and manage persistent state, while taking care of data consistency in case of repeated executions of non-idempotent code [76].

Most existing solutions in intermittent computing, again, operate in isolation and lack integration into a complete system. Our work uses a hardware/software co-design to achieve higher efficiency in a complete system.

3 MOTIVATION

The *Mithræum of Circus Maximus* is an archaeological site in Rome (Italy) [73]. The *Mithræum* was accidentally discovered in 1931 while performing construction works to build a workshop for the local Opera Theater. Historians conjecture that the location was used to host horses and carriages (*carceres*) before entering the nearby *Circus Maximus* for chariot races. Fig. 2 shows the location of the *Mithræum* relative to the *Circus Maximus*. In the third century *d.C.*, a place of worship to god Mitra was created.

The site unfolds as a series of small communicating rooms, covered by barrel vaults whose remains are shown in Fig. 1. The workshop of the Opera Theater currently sits above the *Mithræum* and hosts large machinery and equipment for building theatrical backdrops and sceneries. Concrete columns support the ground level of the workshop, reaching into several of the rooms of *Mithræum* or standing on top of the barrel vaults, as shown in Fig. 1(b).

Goal and requirements. End users wish to gain a thorough understanding of the current conditions at the *Mithræum*. Two requirements are key:

[R1] The integrity of the plaster layers may be affected by specific patterns of *temperature* and *relative humidity*. Given a certain temperature, a threshold exists in humidity where hygroscopic salts start forming on the surfaces. The salts absorb water from vapor in the air, causing a corrosion process to happen on the surface.

[R2] *Vibrations* originating from surrounding vehicular traffic and from the activities at the workshop above may be detrimental to the integrity of the barrel vaults [33]. No evidence is currently available on this aspect at the *Mithræum*.

Collecting data to support a quantitative investigation on these aspects at the *Mithræum* must co-exist with specific constraints:

[C1] *Placing devices* to record vibrations is difficult, as it requires installing accelerometers on *several of the columns* supporting the ground level of the Opera Theater workshop. This necessitates climbing up the barrel vaults every time access to the device is needed. This kind of maintenance operations are to be reduced to a minimum.

[C2] *Form factors* must be reduced, because of the visual impact on historical and artistic pieces. This aspect limits the size of deployed batteries. Such a constraint is not unique to our experience and many embedded sensing deployments, especially in heritage buildings, share similar limitations [9].

[C3] *Commercial chemical batteries* may be considered dangerous. With average relative humidity values in excess of 90% at the *Mithræum*, as discussed in Sec. 6, the chances that batteries start leaking greatly increase [79]. This is, of course, not welcome in such a sensitive environment.

Lowering the *maintenance effort* is thus fundamental, as it determines how practical is the system and the benefit for end users.

4 BATTERY-POWERED OPERATION

We set off by using commercial off-the-shelf components. As such, our first design represents a baseline based on established solutions.

4.1 Design and Deployment

We describe next the hardware we use for our first design, the software we implement, and the initial deployment at *Mithræum*.

Hardware. We use Libelium Waspmotes [46] as the computation and communication core. The device features an ATmega1281 microcontroller unit (MCU) with 8Kb of SRAM. The device absorbs 17mA when computing. Current consumption drops to 7μA in hibernate mode, where computation is not possible and the device state is saved on the local EEPROM. We couple the computing core with an XBee 868LP sub-GHz radio for communication to a data sink. The base board is shown in Fig. 3

To read temperature and humidity, we use a Sensirion SHT85 digital sensor through I^{2}C because of the low-power operation and the ±0.1C° temperature accuracy. It also features a PTFE membrane for protecting the sensor from liquids and dust as per IP67 specifications, without affecting the response time. The nodes equipped with this sensor are termed T/H nodes.

Acceleration readings are obtained through an Analog Devices ADIS16210 combined inclinometer and accelerometer, connected through SPI on a subset of the deployed devices. High accuracy of acceleration sensing and availability of the on-board inclinometer

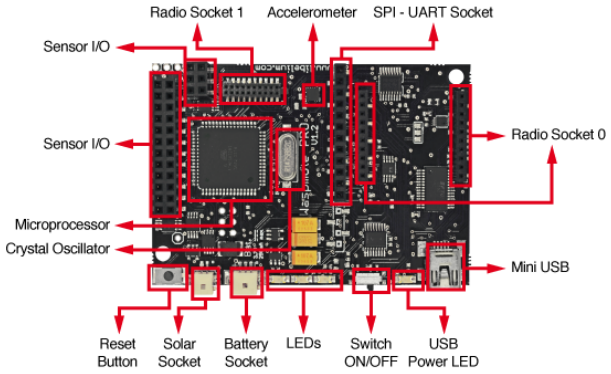
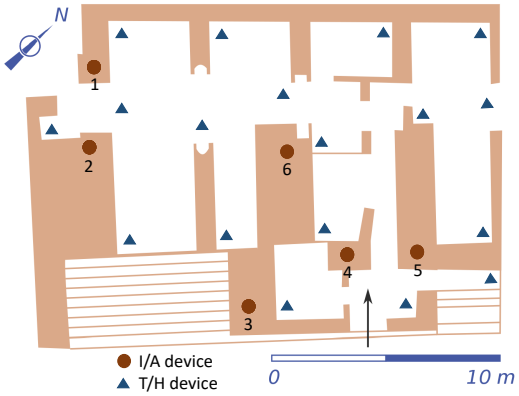


Figure 3: Libelium Waspmote base board.



(a) Deployment map.

(b) Paper authors installing I/A devices.

(c) I/A device in place.

Figure 4: Deployment at *Mithræum of Circus Maximus*. *We install a total of 24 devices; 18 devices are of type T/H with temperature and humidity sensors, 6 devices are of type I/A with temperature, humidity, inclination, and acceleration sensors.*

motivate this choice; the latter may be used to detect permanent changes in the structure [37]. We calibrate each sensor using a shake table and piezoelectric accelerometers for seismic vibrations as a reference [16]. The nodes equipped with this sensor *in addition* to the temperature/humidity one are termed I/A devices.

Software. We implement a periodic procedure to sense temperature and humidity every 20 minutes and to locally store the readings.

At every hour, average and standard deviation of these quantities are computed and reported via radio to the sink.

Onboard I/A nodes, every 20 minutes we additionally record a one minute burst of acceleration readings at 400Hz and sample the inclinometer, according to the guidelines of the structural engineers. At every hour, we process acceleration data by computing the Fast Fourier transform and determining the fundamental frequency as well as spectral density. These information are compressed and also reported to the sink. Such a form of periodic acceleration sensing is common to many deployments for structural analysis [37].

Upon reception, the sink timestamps the data along a global time reference. Between every sensing period, the radio is switched off and the system is placed in low-power mode.

Deployment. Fig. 4 illustrates the deployment. We install a total of 24 devices; 18 devices of type T/H, and 6 devices of type I/A, laid down as shown in Fig. 4(a). For the latter, we use industry-grade epoxy resins to attach the inclinometer/accelerometer sensor to the structure, as shown in Fig. 4(b) and Fig. 4(c) during and after installation. The devices are powered with six type-C batteries.

We deployed a data sink using a Raspberry Pi 3 computer, not shown in the picture, connected to the Internet via 4G. The sink is powered from the grid and, due to the availability of cellular connectivity, could only be installed in a different building at about 250 meters from the *Mithræum*. This motivates the choice of a sub-GHz radio over a 2.4GHz radio, as the signal from the sensing devices needs to penetrate two layers of concrete to reach the sink. Using the sub-GHz radio, no multi-hopping is necessary.

4.2 Lessons Learned

Sec. 6 provides a quantitative account of the performance of the first design iteration. We anticipate here the fundamental lessons learned, which are input to the following design iteration.

> **Lesson 1:** *Whenever there is sufficient energy, embedded sensing runs like a charm.*

Compared to the literature discussed in Sec. 2, the effort required to go from zero to a fully-working embedded sensing deployment drastically reduced. In our case, we quantify this effort from one to two person-months. Whenever the system is operational, it does provide substantial data yield.

We also experiment with more complex networking functionality, such as multi-hop networking and transmission power control, in an attempt to lower the impact of the radio chip. Despite the additional complexity and although none of these attempts really changes the energy figure, the system continues to work reliably *whenever energy was available.*

> **Lesson 2:** *Batteries are the one and only aspect that makes the system unreliable.*

The system experiences a number of failures. Batteries are ultimately accountable for all such occurrences, but for two cases where cellular failures cut the system out of the Internet. The latter represent, however, no significant problem as the sink locally caches sensor data while being disconnected. Our experience contrasts the existing literature discussed in Sec. 2, where earlier deployment experiences resulted in a number of failures due to a great variety of factors, including hardware failures, sensor inaccuracies, or software bugs [9, 38].

The peculiar conditions at the *Mithræum* makes predicting the system lifetime extremely difficult. High ambient humidity and temperature fluctuations cause the alkaline batteries we use to fail unpredictably. This complicates planning the maintenance visits and the associated logistics, causing the periods of down to prolong. Using different battery technology, such as industry-grade alkaline, pro-alkaline, or lithium make essentially no difference. The maintenance effort soon becomes a hampering factor regardless of the value of the data the system produces.

Faced with the limitations of batteries, we have two options to proceed. One possibility is to apply iterative improvements to lower the system energy consumption and extend the maintenance cycle. The unpredictability of failures would, however, remain. The other option is to attack the problem at the root, that is, to seek sources of energy other than batteries. We choose the latter.

5 ENERGY HARVESTING

We set off by swapping batteries for a suitable harvester.

5.1 Design and Deployment

The opportunities for energy harvesting at the *Mithræum* are minimal. As described in Sec. 3, the site is underground and is not illuminated besides when someone is there. Moreover, the nature of the site requires minimally-invasive solutions. T/H and I/A devices thus use different energy harvesting mechanisms because of their different deployment configuration; T/H being placed next to the ground, whereas I/A being attached to the structure.

Thermoelectric energy harvesting. The heat flux generated from the thermal transfer between air and soil at the *Mithræum* creates an opportunity to employ a thermoelectric energy generator (TEG).

A broad range of commercial TEGs exists. Based on the air temperature values collected during the first design iteration and the outdoor seasonal trends in Rome, we expect the thermal deltas between air and layer B to be of some K°. We thus choose a Thermalforce 254-150-36 TEG [74], offering a 30mm by 60mm harvesting surface, connected to layer B through a thermal guide.

Available harvesting management circuits often combine battery charge functionality and output voltage regulation. Solutions specific for TEG may either be passively controlled coupled inductor converters or actively controlled single inductor circuits [75]. The latter feature a dynamic conversion ratio and use maximum power point tracking (MPTT) [6], but require a comparably higher minimal input voltage. Despite these disadvantages, we use a BQ25570 due to its high efficiency for the range of input voltages that most likely correspond to the TEG output in our conditions.

Because the output voltage of the TEG depends on the direction of heat transfer, depending on time of the day, its output may be positive or negative. However, the BQ25570 does not support negative input voltages and hence the TEG output needs to be rectified before being input to the harvesting circuit. We built an ultra low-power rectifier using SiR404DP switches, based on the observation that the TEG output only switches twice a day [32].

Piezoelectric energy harvesting. Thermoelectric generation is not available for I/A devices, as they are too far from the ground. We absorb energy from structural vibrations to power them, taking advantage of the piezoelectric effect. The limited vibrations of the structure, however, require a careful dimensioning of the harvester and of the energy management circuitry, as we discuss next.

We employ a ReVibe modelD energy harvester [26]. The device can be customized by the manufacturer for highest efficiency at a given resonance frequency. We do this based on vibration data gathered with the first design. We choose this specific harvester over alternatives, for example, the modelQ [27] of the same manufacturer, because of the higher power output at the target frequencies. The harvester is attached to the columns of Fig. 1(b) using the same epoxy resins used for attaching the accelerometer/inclinometer.

Based on similar considerations as for T/H devices, we use a BQ25505 here as well. No rectifier circuit is needed.

Computing and communication. As discussed in Sec. 2.2, due to the limited energy availability, it is not conceivable to achieve energy-neutral operation [5, 57]. Therefore, we design the system to work in an intermittent fashion [35].

The Libelium Waspmote we use for the first design is not designed to work in such a setting. We opt to build our own computing and communication platform, using an MSP430FR5989 MCU coupled to a CC1101 transceiver. The choice of an MCU from the FR series is motivated by the need of non-volatile memory to manage persistent state. The radio chip retains the advantages of sub-GHz transmissions described in Sec. 4, with comparable energy consumption. The sensors we use are the same as in the first design.

We configure the output voltage of the BQ25505 buck converter to 2.2V, which represents the worst-case energy need including sensing, local processing, and data transmission. This means that the device is activated as soon as the capacitor voltage is at or above 2.2V. We also configure the BQ25505 to operate in pass-through mode whenever the capacitor voltage falls below this value, to prolong the execution for as long as possible.

We use a 20μF capacitor as energy buffer. We determine its size through a mixed analytical and experimental approach [75], striking a balance between charging times and minimum available energy to guarantee eventual progress of the application. A too large capacitor may take long to charge to a sufficient level, yielding large periods of no system operation when interesting environmental events might be missed. A too small capacitor may not suffice to supply enough energy to complete the most energy-intensive operations, such as transmitting data.

An external Abracon AB18X5 real-time clock (RTC) keeps track of the passing of time while the MCU is off, connected via I^2C. We choose this over remanence timekeepers [23, 36] because of the lower power consumption in the setting we consider. As we only require minute granularity, using the internal RC oscillator on the AB18X5 requires a mere 14nA current. Should the capacitor voltage fall below the RTC supply voltage, causing the latter to reset, we post-process the data at the sink to re-align the timestamps to the global time reference [77].

Programming. As described in Sec. 2.2, system supports exist to enable an intermittent computing pattern [35]. We use a static checkpoint approach [12, 64], which inlines calls to a checkpoint library to copy the complete system state onto FRAM. This allows the system to resume from a point close to the latest power failure. To place checkpoints, we profile the energy consumption of different parts of the code [2] and accordingly inline checkpoint calls. At every such call, a checkpoint takes place if the capacitor voltage drops below a threshold that barely guarantees the energy to dump the system state on FRAM.

We opt for static as opposed to dynamic checkpoints [7, 8, 41, 42], as we cannot afford additional hardware. Compared to task-based programming abstractions [19, 36, 49, 51] that require restructurin the program [43], we wish to leverage the earlier codebase.

Sensing. As the device activates depending on energy intake, the periodicity of sensing can no longer be guaranteed. Depending on harvesting performance, we may simply not have sufficient energy to activate the device every 20 minutes. As a result, we modify the local processing and data transmission functionality to execute only when the same amount of data as in the first design.

It may also happen that the required operation complete with some energy left. To avoid unnecessarily performing a checkpoint at this time, we borrow from Lukosevicius et al.[50] and enter a sleep state. This includes switching the radio off, putting the MCU in the lowest power mode, and setting a timer to trigger another round of sensing in 20 minutes. This also ensures that, at least in the cases where some consecutive rounds may be achieved, this happens with the same period as in the first design.

5.2 Lessons Learned

Similar to Sec. 4.2, we discuss here the main learned lessons and postpone the performance discussion to Sec. 6.

> **Lesson 3:** *When executions are intermittent, peripherals become markedly decisive.*

The workload at *Mithræum* is peripheral-bound. Peripherals execute asynchronously with respect to the computing unit. Their functioning is characterized by own states, which are frequently updated due to the execution of I/O instructions. Information on peripheral states is not automatically reflected in main memory, neither it may be simply queried nor restored [13]. System support for intermittent computing most often only provides support for the computing unit and expect developers to take care of peripherals [19, 49, 78]. Similarly, the few systems addressing the intermittent peripheral problem are not integrated with those for the computing unit [4, 10, 13].

To address this issue, we manually replicate the initialization procedures of all peripherals, including sensors and radio, at every point in the code where execution can possibly resume after a power failure. This is necessary as we cannot anticipate for how long an execution proceeds after resuming and thus what peripherals are used when. The profiling data we use to place checkpoint calls indicates, however, that re-initializing peripherals this way accounts for about 28% of the overall energy consumption, opening up avenues for energy savings with a better solution.

We also crucially realize how the use of radio and sensors vastly determines how far the computation can progress. We particularly observe that the first checkpoint call right after a packet transmission is systematically triggering a checkpoint, as radio operations

are sufficient to cause the capacitor voltage to fall below the checkpoint threshold. However, handling peripherals is not the only source of inefficiency.

> **Lesson 4:** *When energy is scarce, sleeping may not be a smart choice.*

The technique we use in case some energy is left after completing the required workload ultimately represents a waste of energy. Based on the logs we collect, after setting a timer to expire in 20 minutes, in about 89% of the cases the node dies before the timer fires. This means that the energy invested in keeping the system in sleep mode is wasted, as another round of sensing cannot happen in the majority of the cases. In Sec. 6, we further quantify the performance impact of this design choice.

To some extent, this is again an effect of how peripherals impact the energy figure. As every time the device activates at least one peripheral is used, the chances that some energy is left that could power the sleep state for another 20 minutes are slim. This problem aggravates if the radio is also used. If we only consider the cases where we set the 20-minute timer after a packet transmission, in 98% of the cases the node dies before the timer fires.

> **Lesson 5:** *Energy availability may not necessarily overlap with events of interest.*

We expect the data yield to be affected, due to the lower availability of energy. We initially find that energy harvesting can only provide about 22% of the net amount of data that the battery-powered design provides on a monthly basis. Worse is that the information gain obtained from energy harvesting is comparatively way below the reduction in data yield.

This observation particularly applies to I/A devices. Using batteries, the relative abundance of acceleration data can forgive that acceleration sensing is not necessarily synchronized with certain events of interest, such as activities at the Opera Theatre workshop or vehicular traffic. Using energy harvesting, I/A devices activate only depending on the capacitor voltage levels, which might cross 2.2V merely because of vibration noise of no specific interest [37].

The experience and insights we gain prompt us to merge the hardware and software design processes, co-designing the two in an attempt to improve the system performance.

5.3 Re-design and Deployment

We realize different designs for T/H and I/A devices. Their key elements are described next, whereas attached sensors and the timer subsystem remain the same as before.

Programmable activation threshold. Fig. 5 shows the block diagram of the 2nd generation T/H device. It offers two fundamental features: *i)* it allows the MCU to dynamically configure the amount of energy available at the next device activation, by means of controlling the voltage level where the device activates, and *ii)* it provides a software-controlled shutdown functionality, which the MCU uses once the required operations are completed.

To achieve these functionality, we place three voltage comparators in parallel; each corresponding to a different activation threshold. We select comparators from the BU49xx series corresponding to voltage levels matching the energy required for *i)* sensing (V_s^{th}), *ii)* sensing and local processing (V_{sp}^{th}), and *iii)* all application functionality, also including data transmission (V_{spt}^{th}), where $V_s^{th} < V_{sp}^{th} < V_{spt}^{th}$. Every threshold also accounts for the energy required to dump the state on FRAM once the necessary operations complete. An ADG704 digital multiplexer selects what comparator to use based on the input of a two-bit memory the MCU can program directly, typically right before shutting down, by manipulating two GPIO pins. The choice of components is dictated by both their low energy consumption and their matching with the voltage threshold we require, given the same capacitor size.

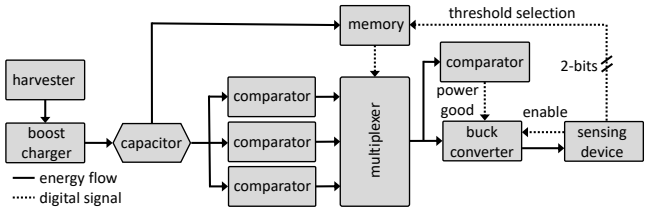


Figure 5: Second generation T/H device. *The design offers the ability for the MCU to programmatically configure the amount of energy available at the next device activation and provides a software-controlled shutdown functionality.*

We implement the two-bit memory using two SN74AUP1G74 flip-flops in a cascading configuration. These feature both an extremely limited quiescent current and a low reset voltage. Below 0.8V, however, they lose their state. This occurs very rarely in our deployment. If the flip-flops reset, their default configuration causes the multiplexer to select the lowest threshold V_s^{th}. This ensures that some progress is eventually achieved.

We deploy a TPS62736 buck converter, which is optimized for the range of currents we expect from the computing and communication core. A further voltage detector turns the "power good" signal up to make the buck converter activate the sensing device whenever the selected input comparator switches its output. As device activation is now separately controlled, we configure the output of the buck converter exactly to 2.1V, which represents an energy-efficient regime for both the MCU [2] and the radio [22] The converter still operates in pass-through mode whenever the capacitor voltage falls below this value. The TPS62736 also features an independent "enable" signal that can be used by the MCU to disconnect from the power sub-system, effectively implementing a software-controlled shutdown.

Our concept of programmable activation shares similarities with Capybara [20] and Dynamic Energy Burst Scaling (DEBS) [31], but we trade generality for a lower energy overhead. The whole power sub-system, in fact, only consumes 5.35μA of quiescent current.

Vibration-triggered activation. Fig. 6 depicts the block diagram of the 2nd generation I/A device. It features two key elements: *i)* it uses a secondary piezo element that operates as a trigger, activating the device only when vibrations above a certain frequency are detected, and *ii)* it provides a 2-bit input line that informs the MCU of the amount of energy available at activation time. Based on the latter, upon activation the application can determine what operations can be completed with the given energy budget.

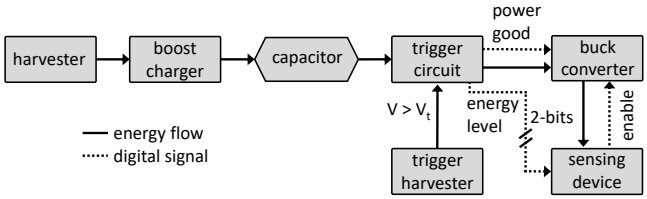


Figure 6: Second generation I/A device. *We use a secondary piezo element to trigger device activation. A 2-bit input line informs the MCU of the energy available at activation time, based on what voltage level the capacitor is currently crossing.*

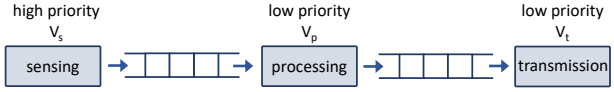


Figure 7: Task-based program structure. *Every task demands a different amount of energy, corresponding to a given voltage threshold. Tasks have different priorities, are loosely coupled, and connected through non-volatile data pipelines.*

We use a Piezo.com Q220-H4BR-2513YB piezoelectric bending transducer [63] as trigger, enclosed in a PPA-500x clamping base with a 13mg tip mass. A custom trigger circuit turns up the "power good" line of the buck converter whenever the trigger piezo generates an output voltage above a threshold V_t and the capacitor voltage is above a threshold V_s^{ia} sufficient for acceleration sensing. The buck converter then activates the device. We select both piezoelectric element and tip mass in a way that V_t can be accurately detected and corresponds to vibrations of interest [37].

The trigger circuit features a set of TLV369x comparators and SiR404DP switches to control the "power good" line of the buck converter and a 2-bit "energy level" input line connected via GPIO to the MCU. Upon activating the device, the latter informs the MCU of the amount of energy available at activation time, based on whether three additional voltage thresholds are crossed. These correspond to the energy for *i)* sensing and local processing (V_{sp}^{ia}), *ii)* sensing and data transmission (V_{st}^{ia}), and *iii)* all application functionality (V_{spt}^{ia}), where $V_s^{ia} < V_{sp}^{ia} < V_{st}^{ia} < V_{spt}^{ia}$. Depending on this input, the application schedules the operations it can perform given a certain energy budget.

The roles and connections of the remaining components are similar to the T/H devices. In this case, the power sub-system only consumes 4.98μA of quiescent current.

Programming. At a fundamental level, both hardware designs aim to exert a higher control on otherwise erratic energy patterns. The design of T/H devices achieves that by giving the MCU the ability to decide the quantity of energy available for the next iteration. I/A devices proactively provide the MCU with information on available energy at the time of activation. Both designs also give the MCU a means to shutdown the device as the current workload is completed.

Taking advantage of these features requires to co-design the software in ways to *i)* precisely isolate and decouple the different functionality corresponding to the different voltage thresholds, *ii)* abandon the strictly-sequential execution semantics, so that different functionality can execute independent of each other, depending on available energy. In doing so, we must come to terms with the effort required to refactor the codebase created earlier, which is unavoidable now.

We opt for a task-based structuring of the code, shown in Fig. 7. Similar to existing work [19, 49, 51, 54, 78], a task here is an atomic piece of functionality that executes in a transactional manner. If energy suffices and a task completes, its output are committed onto a non-volatile data pipeline. If a power failure happens before the task completes, the effects of a partial execution are lost and the task restarts from the beginning when the device is newly active.

Unlike existing solutions, our hardware/software co-design enables a form of energy-aware scheduling that simplifies system operation, while reducing overhead. Upon device activation, the sensing task is enabled and sensors are (re-)initialized. The power subsystem ensures that sufficient energy is available for this when activating the device, as V_s^{ia} is certainly crossed. We additionally enable any other task with input data and whose energy demands are fulfilled by the available energy budget and accordingly (re-)initialize (only) the necessary peripherals. Decoupling tasks so they can execute independent of each other allows the system to make use of the complete 2-bit information of the "energy level" input line, as tasks do not necessarily need to execute in order.

We set higher priority for the sensing task to make sure we do not miss any environment data. Among enabled tasks, we therefore run the sensing task first and commit its results on FRAM. We proceed to run the other enabled tasks and similarly commit the results on FRAM. For I/A devices, as long as sufficient energy is available not to starve the transmission task, no data buffers overflow. For T/H devices, we can proactively ensure this by configuring the activation threshold to provide the transmission task with sufficient energy to run when necessary.

In Sec. 7, we discuss the limitations of our work and cast our design rationale in the larger context of battery-less systems.

6 EVALUATION

In Sec. 6.1 we evaluate our deployment as a scientific instrument to fulfill the end user requirements described in Sec. 3. We compare the system performance of the three design iterations in Sec. 6.2.

6.1 Application

We separate the discussion of the environmental information we gather, as per requirement **R1** in Sec. 3, from the analysis of the structural health of the site, as per requirement **R2** in Sec. 3.

We find that relative humidity at the *Mithræum* is markedly higher than in a regular environment. This may be attributed to the nature of the soil, which tends to funnel humidity from the ground, and to the lack of external ventilation. Combined with our recording of ambient temperature in the 21C°-25C° range, the situation corresponds to roughly 15 grams of vapor per kilogram of air, with peaks of 18 grams in the summer months. This is well above the threshold indicating the creation of hygroscopic salts that possibly cause corrosion processes to occur on the surfaces [48], as explained in Sec. 3.

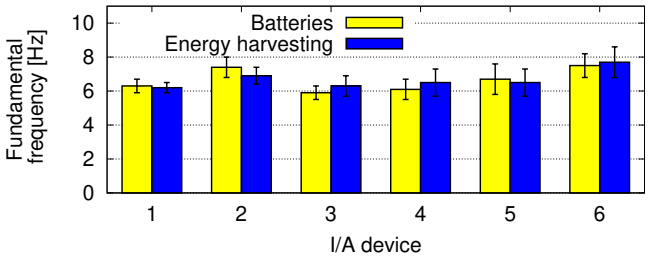


Figure 8: Fundamental frequencies detected when both systems are operational. *The structures at* Mithræum *have different fundamental frequencies than possible external excitating phenomena. This rules out resonance behaviors.*

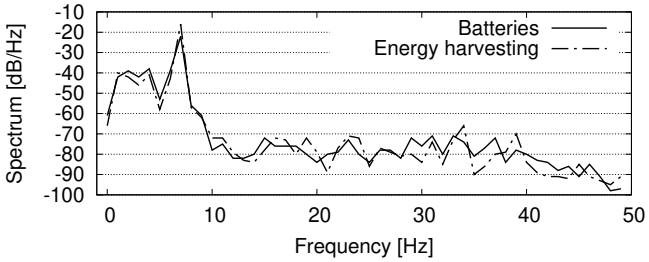


Figure 9: Spectral density at I/A device #6 when both systems are operational. *Only one dominant frequency exists and most of the energy is concentrated below 10Hz, supporting general validity of the structural analysis.*

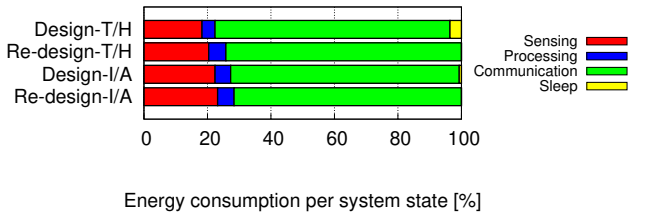


Figure 10: Breakdown of percentage energy consumption depending on functionality. *Peripherals bear a marked impact on the energy figure, as per Lesson 3 in Sec. 5.2. Accurate energy management makes better use of energy spent in sleep mode at T/H devices, as per Lesson 4 in Sec. 5.2.*

Structure. Fig. 8 shows a sample output of the analysis on acceleration data, plotting the average fundamental frequencies recorded throughout the deployment at every I/A device. This information is valuable in that, if the fundamental frequency of an external excitating phenomenon match those of the structure, then the motion of the structure is amplified, resulting in resonance behavior [37].

The external phenomena may be, in our case, activities at the Opera Theater workshop or vehicular traffic. The values in Fig. 8, however, indicate that the fundamental frequencies of the structures at the *Mithræum* are relatively far from those possibly characterizing the aforementioned phenomena, which are thought to lie above 10Hz [37]. Resonance behaviors may thus be safely ruled out.

This reasoning is confirmed by the information on spectral density, shown in Fig. 9 for node #6 as an example. Only one dominant fundamental frequency exists and most of the signal energy is concentrated below 10Hz. As every fundamental frequency follows a specific deflection shape, usually referred to as vibrational mode, we can argue only one such mode exists for the structure at the sampling points. The analysis on the dominant fundamental frequency thus bears general validity [37].

6.2 System

We take the battery-powered design as a baseline hereafter, as the sensing equipment is the same across the two design iterations and only the power source and the associated design choices differ.

Based on detailed logs we collect at a subset of the devices, Fig. 10 quantifies this aspect by showing the breakdown of energy consumption across the four main system states. The plot demonstrates the impact of the peripherals on the energy figure, supporting our claims in Lesson 3 in Sec. 5.2. For the earlier energy-harvesting design, it also shows the contribution of entering a sleep state when the required operations complete with energy left, as described in Sec. 5.1. Crucially, the latter accounts for almost the same fraction of energy consumption as local processing, thus providing a quantitative indication for Lesson 4 of Sec. 5.2. The re-design shifts this energy budget to other functionality, as our hardware design offers a way for the software to shutdown the device when the current workload completes.

Different from T/H devices, Fig. 10 shows that the impact of using low-power modes at I/A devices is minimal. Conversely, the ability of the re-design to activate when a relevant phenomenon occurs counterbalances the smaller amount of collected data. Fig. 8 and Fig. 9, for example, demonstrate that the structural analysis obtained using the battery-powered design or the energy-harvesting re-design is largely equivalent, as the outputs are quite similar in absolute value and variability. This performance is enabled by our design of I/A devices, including *i)* the use of a secondary piezo element to activate the device upon detecting vibrations of interest, and *ii)* the 2-bit "energy level" input line that enables energy-aware scheduling of tasks.

7 DISCUSSION AND OUTLOOK

We articulate how the insights we gain through our specific experience also enjoy more general validity and may seed new directions in the area. At the basis of this discussion is that, in situations of energy scarcity like ours, generality in concrete implementations is a luxury one cannot afford. Different than existing literature that seeks generality in *both* concepts and concrete implementations, our experience motivates developing *general concepts* supported by *application-* or even *deployment-specific implementations*.

Evidence of our reasoning is found on the hardware side, where existing works that focus on accurate energy management [20, 31, 34] largely trade generality for overhead. The generic implementation of the federated energy architecture concept in the Flicker platform [34], for example, costs 10.24μA in device quiescent current: almost twice the figure we have for T/H devices in the re-design. Similar observations apply to Capybara [20] and Dynamic Energy Burst Scaling (DEBS) [31], both proposing useful concepts coupled

with general-purpose implementations whose overhead, in settings akin to ours, are hardly tolerable.

Existing programming techniques largely seek independence from energy patterns and hardware platforms. Most task-based solutions, in particular, adopt a pure software approach [19, 49, 51, 54, 78]. In contrast, our work is based on a few key features:

1) the decision on what task to execute is taken not just based on the availability of input data [19, 49, 51], but also on whether sufficient energy is available.
2) available energy at the start of an active cycle matches the energy demands of a defined subset of tasks, while little to no energy is harvested during an active cycle; as a result, techniques such as two-phase commit for persisting task outputs [51], management of energy events at run-time [78], or task splitting [54] represent an unnecessary overhead: if we schedule a task to start, that task completes successfully.
3) tasks are decoupled and only connected by variable-sized data pipelines; therefore, there is no strict ordering of task executions to be guaranteed [49], neither there are relative timing constraints on their execution [36], as long as the transmission task does not starve, no buffer overflows occur.
4) partitioning the application in tasks explicates the relation between functionality and required peripherals; as a consequence, general solutions for intermittent peripheral operations become unnecessary [4, 10, 13, 20], as every task knows what peripherals it needs and only (re-)initializes those.

Our arguments do not entail that work in this area is necessarily destined to a narrow scope. One may argue, for example, that the our final design is ultimately enabled by a priori knowledge of energy demands. Albeit this is generally not an issue [2], as soon as the demand varies at run-time, our design is no longer applicable. In practice, the role of peripherals, as we learn from Lesson 3 in Sec. 5.2, makes the case of varying run-time energy demands a rare, and often remediable issue. Peripherals largely dominate the energy figure in our deployment and in many other settings [17, 18, 39, 75]. Should peripherals be used based on run-time information, our design is applicable by scaling down the granularity of individual functionality to match the single peripheral operation [31].

We advocate that our experience be an instrument to develop general concepts, backed by *(semi-)automatic methods* to synthesize application- or deployment-specific implementations. For example, the concept of energy buffering we use for T/H devices, while similar to Capybara [20] and DEBS [31] that only offer generic implementations, currently has no way to be instantiated with little effort for another application or deployment. Enabling such a form of (semi-)automatic generation of hardware/software designs may reap the best of both general concepts and efficient application- or deployment-specific implementations.

8 CONCLUSION

We presented the design and evaluation of an embedded sensing deployment at the *Mithræum of Circus Maximus* in Rome, Italy. Besides serving the concrete needs of the end users at hand, the effort was an opportunity to assess the state of maturity of battery-powered embedded sensing as opposed to energy harvesting and corresponding existing solutions.

In our first design, we find that the state of the art in battery-powered embedded sensing is no longer affected by many of the issues that plagued earlier experiences, but still suffers from the hectic performance of batteries. In our later designs, we realize that using energy-harvesting as a replacement for batteries is not as easy in a setting void of energy-rich sources, also due to the lack of integration into a complete system of existing solutions. In contrast, a dedicated hardware/software co-design achieves better utility for data, bringing it back to the level of a battery-powered system.

REFERENCES

[1] J. Adkins, B. Ghena, N. Jackson, P. Pannuto, S. Rohrer, B. Campbell, and P. 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)*.

[2] S. Ahmed, A. Bakar, N. A. Bhatti, M. H. Alizai, J. H. Siddiqui, and L. Mottola. 2019. The Betrayal of Constant Power × Time: Finding the Missing Joules of Transiently-powered Computers. In *Proceedings of the 20th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*.

[3] S. Ahmed, N. A. Bhatti, M. H. Alizai, J. H. Siddiqui, and L. Mottola. 2019. Efficient Intermittent Computing with Differential Checkpointing. In *Proceedings of the 20th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*.

[4] A. R. Arreola, D. Balsamo, G. V. Merrett, and A. S. Weddell. 2018. RESTOP: Retaining External Peripheral State in Intermittently-Powered Sensor Systems. *Sensors* (2018).

[5] D. Balsamo, A. Das, A. S. Weddell, D. Brunelli, B. M. Al-Hashimi, G. V. Merrett, and L. Benini. 2016. Graceful Performance Modulation for Power-Neutral Transient Computing Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2016).

[6] D. Balsamo, B. J. Fletcher, A. S. Weddell, G. Karatziolas, B. M. Al-Hashimi, and G. V. Merrett. 2019. Momentum: Power-Neutral Performance Scaling with Intrinsic MPPT for Energy Harvesting Computing Systems. *ACM Transactions on Embedded Computing Systems* (2019).

[7] 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 Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2016).

[8] 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 Embedded Systems Letters* (2015).

[9] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli. 2008. The Hitchhiker's Guide to Successful Wireless Sensor Network Deployments. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SENSYS)*.

[10] G. Berthou, T. Delizy, K. Marquet, T. Risset, and G. Salagnac. 2018. Sytare: a Lightweight Kernel for NVRAM-Based Transiently-Powered Systems. *IEEE Trans. Comput.* (2018).

[11] N. A. Bhatti, M. H. Alizai, A. A. Syed, and L. Mottola. 2016. Energy Harvesting and Wireless Transfer in Sensor Network Applications: Concepts and Experiences. *ACM Transactions on Sensor Networks* (2016).

[12] N. A. Bhatti and L. Mottola. 2017. HarvOS: Efficient Code Instrumentation for Transiently-powered Embedded Sensing. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*.

[13] A. Branco, L. Mottola, M. H. Alizai, and J. H. Siddiqui. 2019. Intermittent Asynchronous Peripheral Operations. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems (SENSYS)*.

[14] D. Carlson, J. Gupchup, R. Fatland, and A. Terzis. 2010. K2: A System for Campaign Deployments of Wireless Sensor Networks. (2010).

[15] M. Ceriotti, M. Corrà, L. D'Orazio, R. Doriguzzi, D. Facchin, G. P. Jesi, R. L. Cigno, L. Mottola, A. L. Murphy, M. Pescalli, et al. 2011. Is there light at the ends of the tunnel? Wireless sensor networks for adaptive lighting in road tunnels. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*.

[16] M. Ceriotti, L. Mottola, G. P. Picco, A. L. Murphy, S. Guna, M. Corrà, M. Pozzi, D. Zonta, and P. Zanon. 2009. Monitoring Heritage Buildings with Wireless Sensor Networks: The Torre Aquila Deployment. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*.

[17] Q. Chen, Y. Liu, G. Liu, Q. Yang, X. Shi, H. Gao, L. Su, and Q. Li. 2017. Harvest Energy from the Water: A Self-Sustained Wireless Water Quality Sensing System. *ACM Transactions on Embedded Computing Systems* (2017).

[18] H. Chiang, J. Hong, K. Kiningham, L. Riliskis, P. Levis, and M. Horowitz. 2018. Tethys: Collecting Sensor Data without Infrastracture or Trust. In *Proceedings of the 3rd IEEE/ACM International Conference on Internet-of-Things Design and*

Implementation (IoTDI).

[19] A. Colin and B. 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).*

[20] A. Colin, E. Ruppel, and B. Lucia. 2018. A Reconfigurable Energy Storage Architecture for Energy-Harvesting Devices. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS).*

[21] P. Corke, P. Valencia, P. Sikka, T. Wark, and L. Overs. 2007. Long-Duration Solar-Powered Wireless Sensor Networks. In *Proceedings of the 4th Workshop on Embedded Networked Sensors (EMNETS).*

[22] Datasheet. [n.d.]. ChipCon 1101. Retrieved July 10th, 2020 from https://www.ti.com/lit/ds/symlink/cc1101.pdf

[23] J. de Winkel, C. Delle Donne, K. S. Yildirim, P. Pawelczak, and J. Hester. 2020. Reliable Timekeeping for Intermittent Computing. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS).*

[24] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler. 2005. Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN).*

[25] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler. 2006. Trio: enabling sustainable and scalable outdoor wireless sensor network deployments. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN).*

[26] ReVibe Energy. [n.d.]. modelD Piezoelectric Energy Harvester. Retrieved July 8th, 2020 from https://revibeenergy.com/modeld/

[27] ReVibe Energy. [n.d.]. modelQ Piezoelectric Energy Harvester. Retrieved July 8th, 2020 from https://revibeenergy.com/modelq/

[28] V. L. Erickson, S. Achleitner, and A. E. Cerpa. 2013. POEM: Power-Efficient Occupancy-Based Energy Management System. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks (IPSN).*

[29] B. J. Fletcher, D. Balsamo, and G. V. Merrett. 2017. Power Neutral Performance Scaling for Energy Harvesting MP-SoCs. In *Proceedings of the Conference on Design, Automation & Test in Europe (DATE).*

[30] F. Fraternali, B. Balaji, Y. Agarwal, L. Benini, and R. Gupta. 2018. Pible: Battery-Free Mote for Perpetual Indoor BLE Applications. In *Proceedings of the 5th Conference on Systems for Built Environments (BUILDSYS).*

[31] A. Gomez, L. Sigrist, M. Magno, L. Benini, and L. Thiele. 2016. Dynamic Energy Burst Scaling for Transiently Powered Systems. In *Proceedings of the 2016 Conference on Design, Automation & Test in Europe (DATE).*

[32] A. Gomez, L. Sigrist, T. Schalch, L. Benini, and L. Thiele. 2017. Efficient, Long-Term Logging of Rich Data Sensors Using Transient Sensor Nodes. *ACM Transactions on Embeddded Computing Systems* (2017).

[33] M. Guarducci. 2015. Ricordo della Magia in un Graffito del Mitreo del Circo Massimo. In *Mysteria Mithrae.* In Italian.

[34] J. Hester and J. Sorber. 2017. Flicker: Rapid Prototyping for the Batteryless Internet-of-Things. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems (SENSYS).*

[35] J. Hester and J. Sorber. 2017. The Future of Sensing is Batteryless, Intermittent, and Awesome. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems (SENSYS).*

[36] J. Hester, K. Storer, and J. Sorber. 2017. Timely Execution on Intermittently Powered Batteryless Sensors. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems (SENSYS).*

[37] R. C. Hibbeler and T. Kiang. 2015. *Structural analysis.* Pearson Prentice Hall Upper Saddle River.

[38] T. W. Hnat, V. Srinivasan, J. Lu, T. I. Sookoor, R. Dawson, J. Stankovic, and K. Whitehouse. 2011. The Hitchhiker's Guide to Successful Residential Sensing Deployments. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SENSYS).*

[39] N. Ikeda, R. Shigeta, J. Shiomi, and Y. Kawahara. 2020. Soil-Monitoring Sensor Powered by Temperature Difference between Air and Shallow Underground Soil. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)* (2020).

[40] N. Jackson, J. Adkins, and P. Dutta. 2019. Capacity over Capacitance for Reliable Energy Harvesting Sensors. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks (IPSN).*

[41] H. Jayakumar, A. Raha, W. S. Lee, and V. Raghunathan. 2015. QuickRecall: A HW/SW Approach for Computing Across Power Cycles in Transiently Powered Computers. *ACM Journal on Emerging Technologies in Computing Systems* (2015).

[42] H. Jayakumar, A. Raha, J. R. Stevens, and V. Raghunathan. 2017. Energy-Aware Memory Mapping for Hybrid FRAM-SRAM MCUs in Intermittently-Powered IoT Devices. *ACM Transactions on Embedded Computing Systems* (2017).

[43] V. Kortbeek, K. S. Yildirim, A. Bakar, J. Sorber, J. Hester, and P. Pawelczak. 2020. Time-Sensitive Intermittent Computing Meets Legacy Software. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS).*

[44] T. T. Lai, W. Chen, K. Li, P. Huang, and H. Chu. 2012. TriopusNet: Automating wireless sensor network deployment and replacement in pipeline monitoring. In *Proceedings of the 11th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN).*

[45] E. A. Lee and S. A Seshia. 2016. *Introduction to embedded systems: A cyber-physical systems approach.* Mit Press.

[46] Libelium. [n.d.]. Waspmote. Retrieved July 10th, 2020 from http://www.libelium.com/products/waspmote/

[47] G. Loubet, A. Takacs, and D. Dragomirescu. 2019. Implementation of a Battery-Free Wireless Sensor for Cyber-Physical Systems Dedicated to Structural Health Monitoring Applications. *IEEE Access* (2019).

[48] B. Lubelli, R.P.J. Van Hees, and C.J.W.P. Groot. 2006. Sodium chloride crystallization in a salt-transporting restoration plaster. *Cement and concrete research* (2006).

[49] B. Lucia and B. 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).*

[50] G. Lukosevicius, A. R. Arreola, and A. S. Weddell. 2017. Using Sleep States to Maximize the Active Time of Transient Computing Systems. In *Proceedings of the ACM International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems (ENSSYS).*

[51] K. Maeng, A. Colin, and B. Lucia. 2017. Alpaca: Intermittent Execution Without Checkpoints. *Proceedings of the ACM Programming Languages* (2017).

[52] K. Maeng and B. Lucia. 2018. Adaptive dynamic checkpointing for safe efficient intermittent computing. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI).*

[53] K. Maeng and B. Lucia. 2019. Supporting Peripherals in Intermittent Systems with Just-in-Time Checkpoints. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI) (PLDI).*

[54] A. Y. Majid, C. Delle Donne, K. Maeng, A. Colin, K. S. Yildirim, B. Lucia, and P. Pawelczak. 2020. Dynamic Task-Based Intermittent Execution for Energy-Harvesting Devices. *ACM Transactions on Sensor Networks* (2020).

[55] R. Marfievici, P. Corbalán, D. Rojas, A. McGibney, S. Rea, and D. Pesch. 2017. Tales from the C130 Horror Room: A Wireless Sensor Network Story in a Data Center. In *Proceedings of the First ACM International Workshop on the Engineering of Reliable, Robust, and Secure Embedded Wireless Sensing Systems (FAILSAFE).*

[56] P. Martin, Z. Charbiwala, and M. Srivastava. 2012. DoubleDip: Leveraging Thermoelectric Harvesting for Low Power Monitoring of Sporadic Water Use. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems (SENSYS).*

[57] G. V. Merrett and B. M. Al-Hashimi. 2017. Energy-Driven Computing: Rethinking the Design of Energy Harvesting Systems. In *Proceedings of the Conference on Design, Automation & Test in Europe (DATE).*

[58] L. Mottola, G. P. Picco, M. Ceriotti, S. Guna, and A. L. Murphy. 2010. Not All Wireless Sensor Networks Are Created Equal: A Comparative Study on Tunnels. *ACM Transactions on Sensor Networks* (2010).

[59] F. E. Murphy, E. Popovici, P. Whelan, and M. Magno. 2015. Development of an heterogeneous wireless sensor network for instrumentation and analysis of beehives. In *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference (I2MTC).*

[60] M. Navarro, T. W. Davis, Y. Liang, and X. Liang. 2013. A study of long-term WSN deployment for environmental monitoring. In *Proceedings of the 24th IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC).*

[61] S. Peng and C. P. Low. 2012. Throughput optimal energy neutral management for energy harvesting wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC).*

[62] A. I. Petrariu, A. Lavric, and E. Coca. 2019. Renewable Energy Powered LoRa-based IoT Multi Sensor Node. In *Proceedings of the 25th IEEE International Symposium for Design and Technology in Electronic Packaging (SIITME).*

[63] Piezo.com. [n.d.]. Q220-H4BR-2513YB piezoelectric bending transducer. Retrieved July 8th, 2020 from https://piezo.com/products/piezoelectric-bending-transducer-q220-h4br-2513yb

[64] B. Ransford, J. Sorber, and K. Fu. 2011. Mementos: System Support for Long-running Computation on RFID-scale Devices. *ACM SIGARCH Computer Architecture News* (2011).

[65] A. Rodriguez, D. Balsamo, Z. Luo, S. P. Beeby, G. V. Merrett, and A. S. Weddell. 2017. Intermittently-powered energy harvesting step counter for fitness tracking. In *Proceedings of the IEEE Sensors Applications Symposium (SAS).*

[66] E. Ruppel and B. 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).*

[67] M. M. Sandhu, K. Geissdoerfer, S. Khalifa, R. Jurdak, M. Portmann, and B. Kusy. 2020. Towards Optimal Kinetic Energy Harvesting for the Batteryless IoT. *arXiv preprint arXiv:2002.08887* (2020).

[68] N. Saoda and B. Campbell. 2019. No Batteries Needed: Providing Physical Context with Energy-Harvesting Beacons. In *Proceedings of the 7th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems (ENSSYS).*

[69] U. Senkans, D. Balsamo, T. D. Verykios, and G. V. Merrett. 2017. Applications of Energy-Driven Computing: A Transiently-Powered Wireless Cycle Computer. In *Proceedings of the 5th ACM International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems (ENSSYS)*.

[70] V. Sharma, U. Mukherji, V. Joseph, and S. Gupta. 2010. Optimal energy management policies for energy harvesting sensor nodes. *IEEE Transactions on Wireless Communications* (2010).

[71] L. Spadaro, M. Magno, and L. Benini. 2016. Poster Abstract: KinetiSee - A Perpetual Wearable Camera Acquisition System with a Kinetic Harvester. In *Proceedings of the 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*.

[72] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. 2004. An Analysis of a Large Scale Habitat Monitoring Application. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SENSYS)*.

[73] C. Tavolieri and P. Ciafardoni. 2010. Mithra. Un viaggio dall'Oriente a Roma: l'esempio del Mitreo del Circo Massimo. *Archaeology Archives, BA* (2010). In Italian.

[74] Thermalforce. [n.d.]. 254-150-36 TEG. Retrieved July 10th, 2020 from https://www.dropbox.com/s/4xx1z2gwddntc42/TG254-150-36l.pdf?dl=0

[75] M. Thielen, L. Sigrist, M. Magno, C. Hierold, and L. Benini. 2017. Human body heat for powering wearable devices: From thermal energy to application. *Energy conversion and management* (2017).

[76] J. Van Der Woude and M. Hicks. 2016. Intermittent Computation Without Hardware Support or Programmer Intervention. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI)*.

[77] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. 2006. Fidelity and Yield in a Volcano Monitoring Sensor Network. In *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI)*.

[78] K. S. Yildirim, A. Y. Majid, D. Patoukas, K. Schaper, P. Pawelczak, and J. Hester. 2018. InK: Reactive Kernel for Tiny Batteryless Sensors. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems (SENSYS)*.

[79] J. Zhang, C. Chen, X. Zhang, and S. Liu. 2016. Study on the environmental risk assessment of batteries. *Procedia Environmental Sciences* (2016).