AsTAR: Sustainable Battery Free Energy Harvesting for Heterogeneous Platforms and Dynamic Environments

Fan Yang, Ashok Samraj Thangarajan, Wouter Joosen, Christophe Huygens and Danny Hughes
imec-DistriNet, KU Leuven, Leuven, Belgium
fan.yang@cs.kuleuven.be

Gowri Sankar Ramachandran and Bhaskar Krishnamachari
Department of Electrical Engineering, University of Southern California, Los Angeles, USA

Abstract

Today's commercial Internet of Things devices remain largely dependent upon batteries, which offer high capacity, stable energy storage at the expense of limited shelf-lives and toxic chemical compositions. Research on sustainable energy harvesting platforms is essential to realizing a new generation of long-lived and environmentally friendly IoT products. This paper contributes to this goal by introducing AsTAR, an energy-aware task scheduler and associated reference platform that aims to lower the burden of developing sustainable applications through self-adaptive task scheduling. We evaluate AsTAR based on its capability to deliver sustainable operation on heterogeneous platforms. Evaluation shows that: (i.) With zero modeling AsTAR rapidly identifies optimum task scheduling rates, while (ii.) reacting quickly to environmental change and (iii.) these features incur minimal performance overhead in terms of memory, computation and energy. Considered in sum, we believe that these features significantly simplify the process of creating sustainable energy harvesting IoT applications.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Microprocessor/microcomputer applications, Real-time and embedded systems

General Terms

Design, Experimentation, Management

Keywords

Energy harvesting, adaptive scheduling, Internet of Things

1 Introduction

The Internet of Things (IoT) is being deployed at massive scale in homes, smart cities and industrial applications. Analysts such as Gartner [14] and Forrester [16] predict that billions of IoT devices will be deployed by the year 2020. In this context, the typical battery life of IoT devices, which ranges from a few years to a decade, is problematic. Manually replacing billions of batteries is not economically feasible, and abandoning large quantities of toxic batteries in our environment is not sustainable, as batteries require many years to fully decompose. Energy harvesting provides the only feasible means to address this problem. By ensuring that IoT devices can be perpetually powered by environmental energy, batteries can be eliminated, conserving both human effort and the environment.

While the potential of energy harvesting in the IoT is clear, it is notoriously complex to develop reliable energy harvesting applications due to three factors: *(i.)* Environmental dynamism, which gives rise to an unpredictable power budget, *(ii.)* heterogeneity in platform and peripheral power consumption profiles and *(iii.)* the *tragedy of the coulombs* [8] wherein a single energy-hungry software module can starve all other modules of energy, rendering the device inoperable.

This paper addresses these three problems by introducing AsTAR (named for its Asymmetric Task Adaptation Rate scheduler), a hardware and software platform for building energy harvesting IoT applications. AsTAR addresses environmental dynamism by providing *self-adaptive task scheduling*. Sustainability is ensured by throttling task execution rates as required to acquire and sustain a developer-specified optimal level of charge. AsTAR is fully autonomous and requires no pre-configuration or a-priori modeling to deal with different energy harvesting sources or peripheral hardware, which may even be modified at runtime. Fine-grained control over task execution and peripheral usage prevent a single energy-hungry software module from rendering the system non-functional. The novel contributions of AsTAR are threefold: *(i.)* Simple, yet effective energy-aware task scheduling on IETF Class-1 devices [12], *(ii.)* Support for platform heterogeneity and dynamism with zero up-front modeling and *(iii.)* A reference platform for experimentation with sustainable energy harvesting applications.

We evaluate AsTAR in both laboratory tests and a realistic deployment scenario, using indoor solar energy harvesting. Evaluation shows that *(i.)* AsTAR achieves its goal of amassing an optimal charge level on heterogeneous hardware platforms, *(ii.)* AsTAR adapts quickly to dynamic levels of

International Conference on Embedded Wireless Systems and Networks (EWSN) 2019
25–27 February, Beijing, China

ISBN: 978-0-9949886-3-8

power production or consumption and *(iii.)* AsTAR has extremely low performance overhead even on IETF Class-1 devices. Considered in sum, we believe that AsTAR's features significantly simplify the development of sustainable energy harvesting IoT applications.

The remainder of this paper is structured as follows. Section 2 discusses related work and highlights the gap that AsTAR addresses in the research literature. Section 3 presents the AsTAR approach including adaptive task scheduling, software stack, and hardware platform. Section 4 describes our reference implementation. Section 5 evaluates the performance of AsTAR. Finally, Section 6 concludes and discusses directions for future work.

2 Related Work

We review three key streams of research that are related to energy harvesting. Section 2.1 reviews prior work on energy harvesting and battery-free platforms. Section 2.2 discusses OS support for energy harvesting. Section 2.3 discusses software development support for energy harvesting applications. Finally, in Section 2.4 we highlight the gap in prior research that is addressed by AsTAR. Due to space constraints, we are unable to provide a complete review of prior energy harvesting work. For a more in-depth view on the topic, we refer the reader to [2].

2.1 Energy Harvesting IoT Platforms

The first generation of energy harvesting platforms for the IoT augmented battery-powered systems with solar panels to extend the service-life of devices and to provide an expanded energy envelope for system operation. Hughes et al. developed GridStix [11], a wireless sensor network for flood monitoring and warning that was capable of sustainable multi-year operation using a combination of solar panels and rechargeable batteries. By adapting its network stack, GridStix provided reliable telemetry in different operational conditions [20]. Taneja et al. [22] and Buchli et al [3] developed systematic models of solar energy production in outdoor sensing scenarios. Hsu et al. [10] tackle the problem of optimally scheduling solar device operation under variable solar loads based upon a combination of in-situ benchmarking and modeling. While the majority of work in the area of predictive energy harvesting has focused on solar power, Gaglione et al. also demonstrated sustainable battery-free operation using harvested kinetic energy in a vibration based structural health monitoring application [7] using a combination of benchmarking and simulation.

Contemporary energy harvesting platforms enable a broader range of energy harvesting sources such as vibration, thermal and solar, and eliminates batteries as energy storage in order to prolong the operational life of IoT devices. While the lifetime of an appropriately selected supercapacitor is orders of magnitude longer than a standard battery, the charge density of capacitors is far lower than that of batteries. The result is that battery-free sensing platforms have a much smaller and more dynamic energy envelope than battery-powered systems. Hester et al., identify the *'Tragedy of the Coulombs'* in energy harvesting systems wherein a single energy-hungry hardware or software component can consume sufficient energy to render the entire system inoperable. United Federation of Peripherals (UFoP) [8] addresses this problem through a hardware-driven solution, wherein harvested energy is divided into multiple peripheral-specific capacitors, which provide independent power sources and therefore prevent a single sub-optimal peripheral from causing the Tragedy of the Coulombs. Flicker [9] builds on UFoP, extending the concept of isolated per-peripheral energy storage to support plug-and-play peripheral boards that may be flexibly connected at development time. Hester et al. demonstrate through representative user-trials that the isolation of peripheral power consumption significantly reduces the difficulty of realizing energy-harvesting applications.

The approaches to energy harvesting described above fall into two broad categories. The first category of research is to provide *sustainable* operation in the face of dynamic energy availability [11] [22]. The second approach embraces *intermittent* operation [8] [9]. AsTAR contributes to the sustainability stream of research, aiming to ensure sustainable system operation on a battery-free energy harvesting platform that can function as a drop-in replacement for long-life battery powered systems.

2.2 OS Support for Energy Management

IoT Operating Systems (OSs) play a key role in managing the complexity of a dynamic energy envelope through the adaptive scheduling of tasks and the reconfiguration of system-level functionality. Nano-RK [6] provides energy awareness on an embedded Real Time OS (RTOS). As with many Real Time Operating Systems, Nano-RK guarantees that task deadlines are met using priority-based preemptive multi-threading. Nano-RK mediates access to every system resource with a consistent API and is thereby capable of performing fine-grained energy accounting based upon an energy model, which is established prior to deployment. Nano-RK is deeply dependent upon its static energy model, which makes it difficult to apply on reconfigurable IoT platforms such as Flicker [9] and μPnP [24].

The Cinder OS [19] is built for energy sensitive mobile devices such as smart phones. Cinder supports energy management by providing developers with two dedicated abstractions, *reserves* and *taps*. A *reserve* permits an application to use a certain amount of energy, while a *tap* limits the maximum rate of energy that can be transferred or consumed. Cinder tasks may allocate energy from their *reserve* to subtasks and control the rate of energy consumption through the *tap*. These features together lower the burden on developers to create energy-aware applications. However, Cinder does not consider energy harvesting. Furthermore, it targets far more powerful mobile-phone class devices.

Cao et al. [4] provide support for energy management on multi-application IoT devices by introducing dedicated energy management abstractions in the form of *reserves*, which provide a virtual representation of the amount of energy available to an application. Isolation of energy reserves is enforced by the system at runtime to ensure fair access to energy resources on shared platforms. When an application exhausts its reserve, it is terminated. This approach is dependent upon a common virtual machine which mediates and can therefore control access to low-level, energy consuming resources.

Eon [21] is a programming language and supporting runtime that is designed to facilitate the development of perpetually powered systems. Eon allows programmers to define multiple paths or *flows* through a program. The Eon scheduler then selects flows and their rates of execution based upon the current energy levels in order to maintain sustainable operation. Eon makes energy supply predictions based upon an Exponentially Weighted Moving Average (EWMA) of observed energy generation, while energy consumption estimates are based upon benchmarks obtained from testing of each hardware and software platform.

2.3 Software Development for Energy Harvesting Applications

It is notoriously difficult to develop energy-aware software that is capable of sustainable operation in unpredictable energy harvesting conditions. Prior work has contributed a number of software engineering tools to reduce this complexity.

Eco [25] is a programming model for sustainable software. Programs written in Eco may adaptively adjust their behavior to stay within a given energy or temperature budget. Eco achieves a fine-grained, programmable, and declarative sustainability by the language runtime consistently matching supply and demand. This feature would significantly simplify the creation of sustainable energy harvesting applications. However, the Eco runtime targets only server-class devices with orders of magnitude more resources than typical IoT devices.

Ritter et al. [18] combine software instrumentation and super-capacitor charge storage to accurately estimate the lifetime of an IoT device. By replacing the battery of the device with an appropriately sized capacitor and measuring the voltage before and after the execution of a function, Ritter et al. demonstrate that it is possible to accurately measure energy consumption based upon the near linear relationship between the capacitor voltage and average current with a fixed capacitor. While the authors do not focus on energy-harvesting, the paper showcases a secondary benefit of using super-capacitors beyond their long lifespan; the ability to profile energy consumption in the field. This proposed methodology is dependent upon manual code instrumentation and therefore should be expected to incur significant development-time overhead.

Lachenmann et al. [13] append energy consumption data to software tasks and enable developers to assign tasks within energy *levels*, which may be deactivated in order to reduce energy consumption. The Levels task scheduler then selects the highest energy level, and thus group of associated tasks, which may be executed while still achieving a given lifetime goal. This approach ensures that battery lifetime goals are met, while allowing developers to prioritize tasks for execution. All energy estimates must be generated before deployment.

2.4 Gap Analysis and Requirements

We now reflect on the gap that AsTAR addresses with respect to the related work discussed in Section 2.1 to Section 2.3.

- *Reducing the burden of benchmarking:* many of the approaches described above [18] [13] [22] [3] demand the creation of detailed hardware models. Modeling requires significant effort and access to laboratory equipment, forming a barrier to entry for energy aware software development. We argue that energy management software should not require up-front benchmarking and instead gather all necessary information automatically.
- *Operation on heterogeneous platforms:* IoT applications increasingly run across multiple platforms and, for most energy management approaches, this requires extensive re-parameterization of an underlying energy model. The problem is even more acute for reconfigurable IoT platforms [9] [24], where the hardware base may change at runtime through the connection of new peripherals. Energy aware software should cope with platform heterogeneity without the need for per-platform developer intervention.
- *Mitigating dynamism:* To maximize applicability, energy management approaches for energy harvesting IoT platforms should ideally respect the inherently unpredictable nature of many environments, where available energy from light, heat or vibration may vary widely or even be lost without notice. In such unpredictable environments, long term modeling [22] [3] is likely to prove unproductive. In the case of reconfigurable IoT platforms such as Flicker [9] and μPnP [24], the modification of connected peripherals at runtime gives rise to dynamic power demands.
- *Low runtime overhead:* Finally, any energy management approach must naturally itself be efficient in terms of memory and computation if it is to be implemented on IETF Class-1 IoT devices, precluding heavyweight techniques such from mainstream computer systems [25] [19] [20] .

The following section discusses how the design of AsTAR achieves sustainable energy harvesting operation, while addressing the gaps identified in prior research.

3 AsTAR Approach

The primary design goal of AsTAR is to ensure *sustainable operation* in the widest possible range of energy harvesting scenarios. The AsTAR approach achieves this by optimizing task scheduling in order to first acquire and then sustain a developer-specified optimum level of stored energy. AsTAR aims to accomplish this without benchmarking or environmental modeling in order to minimize developer effort. Instead, AsTAR continually observes and reacts to system state at runtime. Maximizing task execution rates is useful for all sensing tasks that benefit from higher temporal resolution in order to reduce latency.

Scope and limitations: AsTAR's energy aware adaptation is limited in scope to adapting how often sensor tasks may execute. The more frequently a sensing task is executed, the higher the temporal resolution of sensed data. However, this higher resolution comes at the expense of greater energy consumption. AsTAR strives to balance these conflicting optimization goals by maximizing task execution to the greatest extent possible, while also ensuring that charge is smoothly accumulated and sustained at a developer-specified optimum. The AsTAR approach is limited in its applicability of tasks whose schedule can be independently controlled, such as sensing operations.

AsTAR is designed to deal with platform *heterogeneity* in terms of: device power supply, peripheral power demands and energy storage capacity. AsTAR also aims to operate in unpredictable energy harvesting environments where it mitigates *dynamism*. In the design of AsTAR, we have drawn inspiration from the Additive Increase Multiplicative Decrease (AIMD) technique that is used to implement congestion control in TCP. TCP senses congestion and responds by increasing or decreasing message rates. By using different behavior when increasing (additive) and decreasing (multiplicative) transmission rates, AIMD tends to avoid oscillation and spends most of its time near an optimum value. In 1989, Chiu et al. [5] showed that AIMD is near optimal in identifying an optimum messaging rate for a channel of unknown and dynamic capacity. From a non-functional perspective, AIMD has two important properties: *(i.)* it is extremely lightweight, storing minimal state and *(ii.)* it is robust, having been deployed and studied at massive scale for over 40 years.

Intuitively, the properties of AIMD are a good fit with the energy harvesting problem tackled by AsTAR, though we have modified the operation of the scheme to reflect that the goal of charge accumulation is to reach an optimum, rather than to maximize charge to the greatest extent possible. Section 3.1 describes the adaptive AsTAR task scheduling algorithm and how we have modified AIMD for our application case. Section 3.2 then describes the AsTAR software platform. Finally, Section 3.3 describes the AsTAR hardware platform.

3.1 Adaptive Task Scheduling

AsTAR tasks are self-contained and execute at a rate controlled by the AsTAR runtime. AsTAR observes capacitor charge levels by periodically measuring the capacitor voltage at a rate specified by the developer, and based upon the observed level of stored charge, task execution rates are adapted. AsTAR monitors and reacts to changes in stored charge, rather than energy supply as this inherently take into account the complex interactions that occur between supply current, supply voltage, capacitor voltage and capacitor leakage current, all of which would otherwise need to be explicitly modeled. Our initial prototype of AsTAR treats all tasks as equally important and does not allow for prioritization. We plan to add support for this in our future work.

Energy Aware Task Scheduling: AsTAR defines three charge levels which guide the operation of the task scheduling algorithm based upon the voltage of the supercapacitor. These levels may be configured by the developer prior to deployment.

- ***Shut-off Voltage state***: In this state, system voltage has fallen to a level below the user specified Shut-off Voltage (SV). The primary goal of the system is to maximize sustainability by reducing power consumption. All application tasks are therefore disabled until sufficient charge accumulates to run all system components.
- ***Low Voltage state:*** in this state, which lies between the Shut-off Voltage (SV) and Optimum Voltage (OV) thresholds, the goal of the system is to increase the frequency of tasks while ensuring that charge accumulates during each *adaptation-period*. The system applies AIMD on scheduling task rate in this state, where it gently probes the available energy supply by increasing task rates additively and, if charge ceases to accumulate in the capacitor, backs off aggressively using multiplicative decrease to protect the charging process.

```
// The adaptation algorithm
uint16_t get_adapted_rate(){
  read Capacitor_Voltage from ADC

  if  Capacitor_Voltage < Shut_Off_Threshold
    Shut_Off_state()
  else if Capacitor_Voltage < Optimum_Voltage_Threshold
    Low_Voltage_state()
  else if Capacitor_Voltage > Optimum_Voltage_Threshold
    High_Voltage_state()
  else
    Optimum_Voltage_state()
  return task_rate
} // End of function

// Low Voltage state function: apply AIMD
Low_Voltage_state(){
  if voltage is rising
    task_rate ++
  else
    task_rate = task_rate / 2
}

// High Voltage state: apply MIAD
High_Voltage_state(){
  if voltage is falling
    task_rate --
  else
    task_rate = task_rate * 2
}

// Optimum voltage state: stay the same rate
Optimum_Voltage_state(){
  task_rate = task_rate
}

// Shut off state: Task disabled
Shut_Off_state(){
  task_rate = 0
}
```

Listing 1. Pseudocode of the AsTAR adaptation algorithm

- ***Optimum Voltage state:*** in this state, charge levels are optimal and the goal of AsTAR is to remain at this charge level. Task rates will not be adapted unless charge falls into the LV state, or rises into the HV state.
- ***High Voltage state***: In this state, system voltage is above the Optimum Voltage (OV) threshold. The goal of the system is to make the most use of the abundant energy to avoid wasting energy above the OV threshold. AsTAR applies AIMD in reverse: if charge is not

decreasing, tasks rates are Multiplicatively Increased (MI). Conversely, when voltage level falls, AsTAR gently reduces task rates with Additive Decrease (AD) to prevent under-shooting the optimum.

The AsTAR scheduling algorithm is simple and compact, as is shown by the pseudo code provided in Listing 1. Furthermore, AsTAR requires very little state information. All decision making is based upon the developer-specified SV and OV thresholds (which implicitly defines the four states) along with current and previous voltage readings and per-task minimum and maximum execution rates. Together AsTAR requires a fixed total of 4 bytes of memory for system wide configuration and 8 bytes per task. This makes AsTAR suitable for even the most constrained embedded devices.

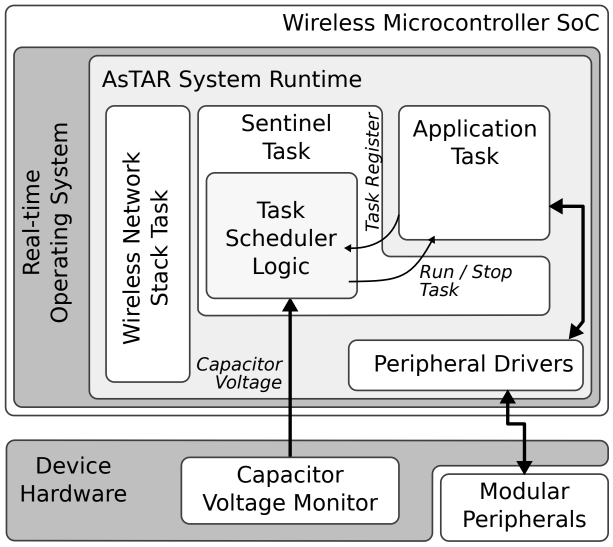

Figure 1. AsTAR software architecture extends Real Time Operating Systems.

3.2 AsTAR Software Stack

To achieve per-task energy accounting, AsTAR extends Real Time Operating Systems (RTOS) with support for energy awareness. Real-time task scheduling is necessary to ensure that tasks execute atomically and thereby isolate their individual power consumption by measuring capacitor voltage before and after every task execution.

The core of AsTAR is the *Sentinel* task, which controls and manages the execution of all energy aware *AsTAR tasks*. Each AsTAR task registers with the sentinel, passing its *maximum* and *minimum* execution rate. The sentinel will then schedule each application task according to the algorithm described in Section 3.1 within the minimum/maximum range. The in-situ power monitor task is controlled and configured by the Sentinel task.

The API to register tasks is as follows:

$$register(task, min\_rate, max\_rate)$$

Where *task* is a reference to the task to be scheduled, *min_rate* specifies the minimum rate at which the task must run, and *max_rate* specifies the maximum rate at which the task may run. All task rates are currently specified in messages per hour.

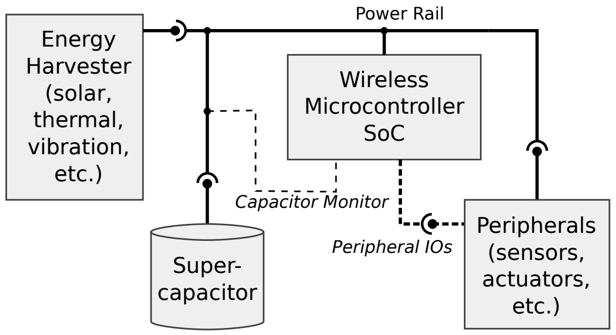
Power Rail
Energy Harvester (solar, thermal, vibration, etc.)
Wireless Microcontroller SoC
Capacitor Monitor
Peripheral IOs
Super-capacitor
Peripherals (sensors, actuators, etc.)

Figure 2. Architecture of the minimalist AsTAR hardware platform. AsTAR can operate with various energy harvesters, supercapacitors and peripherals.

AsTAR tasks may call any underlying system functionality, however, the tasks themselves are fully isolated and may therefore communicate only via messaging over the network stack. In terms of energy, the sentinel considers tasks to be black boxes and all energy accounting and control occurs at the level of individual task execution

Background power consumption below the level of the AsTAR software runtime is neither measured nor controlled, however *AsTAR tasks* will adapt to this changing load by modifying their own behavior. The greater the proportion of system functionality that is realized as *AsTAR tasks*, the finer the granularity with which energy can be managed.

3.3 AsTAR Hardware Platform

We have designed a minimalist battery-free Internet of Things hardware platform equipped with support for energy harvesting in order to evaluate AsTAR. This platform uses a super-capacitor for energy storage and may use various forms of harvested energy as its power supply. The platform can be configured to work with different application peripherals. The AsTAR software stack adapts to different sizes of a capacitor, a variety of peripherals, and harvesting sources.

The design of AsTAR hardware strives for simplicity, it consists of only the basic elements of an energy harvesting IoT platform, which ensures that it is easy to port AsTAR to other platforms. A key feature of the AsTAR hardware platform is its modular design, which allows the energy harvester, super-capacitor, and application peripherals to be replaced. In the case of peripherals, these may also be replaced at runtime. Figure 2 provides a high level overview of the AsTAR hardware platform, the major elements of which are explained below.

Super-capacitor Energy Storage: As can be seen in Figure 2, the AsTAR hardware platform uses a super-capacitor as its primary energy storage. The platform design makes no assumptions about the specifications of its super-capacitor. The adaptive software stack introduced in Section 3.1 will adapt system behavior to suit the characteristics of this super-capacitor.

Energy Harvester: The AsTAR hardware platform adapts to various energy harvesting mechanisms through a modular connector. The AsTAR platform accepts voltages in the native working range of the system (e.g., harvesting from solar panels). Energy harvesters that generates AC voltages (e.g. vibration energy harvesting) or DC voltages outside of working voltage range of system (e.g. Thermoelectric Generators (TEGs)) should be responsible for rectifying and converting the voltage to the working range. The AsTAR system makes no assumptions about the specifications of its energy harvester, which can be drop-in replaced without any software reconfiguration or modeling. Multiple energy harvesters can also be connected together to maximize useful exploitation of available energy.

Voltage Monitoring: The AsTAR hardware platform provides voltage measurement of the super-capacitor. Voltage measurement is critical to the system since it is required by the AsTAR scheduler as discussed in Section 3.1. Naturally, voltage measurement with a higher resolution provides finer grained of adaptation control. Our lab experiments suggest that a standard 10bit ADC is sufficient for this purpose.

Peripherals: The AsTAR platform supports modular application peripherals through a generic peripheral connector. Peripherals may include sensors, actuators, or even secondary networks. Modular peripheral connection provides flexibility to the platform so that the system can fit into different applications by adapting different peripherals. AsTAR system has no restrictions on peripheral energy demands, and the system adapts to changing energy demands and even changing types of connected peripherals.

Wireless Microcontroller: In principle, the AsTAR approach is compatible with any wireless micro controller unit that offers sufficient memory to execute the system software, and an analog channel to monitor capacitor voltage. Naturally, given the limited energy envelope offered by energy harvesting, low power operation is critical. We selected the LTC5800-IPM from Analog Devices International [1], an IETF Class-1 microcontroller, as our wireless microcontroller due to its industrially-proven low power operation [15] and high reliability [17].

4 Implementation

Section 4.1 describes key hardware implementation details for our prototype. Section 4.2 then describes implementation details of the AsTAR software stack.

4.1 Hardware Implementation

Figure 3 shows a finished prototype of the AsTAR device equipped with solar panel. We selected the LTC5800-IPM from ADI to implement our prototype AsTAR platform. The LTC5800-IPM is an IETF Class-1 device that is realized as a highly integrated System on Chip (SoC), it includes the lowest power commercially available IEEE 802.15.4e radio unit and a 7.37MHz ARM Cortex-M3 32-bit microprocessor with 12KB RAM and 32KB flash memory available for applications. The LTC5800-IPM consumes 800 nA in deep-sleep mode, 1.3 mA in active mode, 4.5 mA when receiving a packet and 9.7 mA when transmitting a packet. The LTC5800-IPM runs the AsTAR software stack, as defined in Section 3.2.

Our experience has shown that super-capacitors with a low Equivalent Series Resistance (ESR) provide the best per-

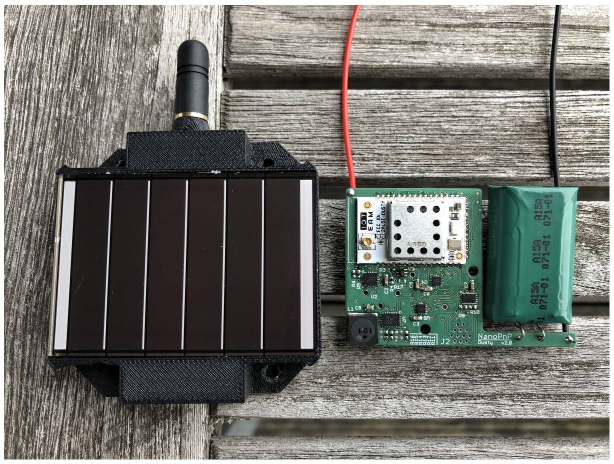

Figure 3. Finished AsTAR device prototype with solar panel as energy harvester, together with the bare circuit board.

formance as (i.) they can be charged easily from low current energy harvesting sources and (ii.) they can supply large currents without experiencing a significant voltage drop. This is necessary to support radios with high transmission power or high current sensors. Low leakage current is also essential for charge retention. We selected carbon aerogel capacitors for our prototype implementation in sizes from 1F to 5F.

We evaluated our prototype implementation of AsTAR with controlled power supplies and solar energy harvesting. Other energy harvesting techniques such as *thermal*, *RF* and *kinetic energy harvesting* can be integrated with minimal hardware changes and will be the subject of our future work. In the case of solar tests, we use AM-5608CAR solar panel from Panasonic-BSG, a polycrystalline panel that generates up to 36mA at 3.3V under ideal operating conditions.

4.2 Software Implementation

We selected uC-OS/II from Micrium [1] as the RTOS implementation for our prototype due to its integration with the LTC-5800-IPM and SmartMesh-IP. uC-OS/II also runs the SmartMesh IP network stack [23], a commercial implementation of Time Synchronized Channel Hopping (TSCH) and IPv6.

The SmartMesh IP stack is composed of IEEE 802.15.4e at the physical layer, TSCH at the link layer, and a standards-based IPv6-ready IoT *upper stack* (6LoWPAN, UDP). SmartMesh IP motes form a redundant low-power wireless multi-hop mesh network. The manager is responsible for building and maintaining the TSCH schedule of the network. By continuously monitoring the network and adapting the TSCH schedule to topological changes or different communication requirements, a SmartMesh IP network runs autonomously without human intervention. Matthys et al. [15] highlights the performance of SmartMesh IP, and demonstrates that it is possible to achieve battery lifetimes in excess of 6 years using a 2400mAh battery, even with plug-and-play peripheral support. The Key goal of AsTAR is to extend this lifespan.

[1]https://www.micrium.com/rtos/kernels/

The AsTAR runtime is compact and efficient, requiring just 8kB of flash memory and under 8 bytes of memory for each AsTAR peripheral being managed in order to hold the tasks minimum rate, maximum rate and the voltage that was observed during the last sample cycle.

5 Evaluation

In this section, we evaluate AsTAR against the requirements identified in Section 2.4. Demonstrating that without any prior benchmarking and modeling, AsTAR can achieve its sustainability goals in the face of (i.) platform heterogeneity and (ii.) runtime dynamism, while (iii.) incurring minimal memory and computation overhead. These features are evaluated firstly in controlled lab environment and then in a real-world solar energy harvesting scenario.

For all of our experiments, AsTAR was configured with the following parameters: Shut-off Voltage (SV) = 2.7V, Optimal Voltage (OV) = 3.7V. The adaptation-period was set to 1 minute. The minimum task execution rate was set to 1 execution per adaptation-period, while the maximum task execution rate was set to 255 executions per period. We created test peripherals with a representative range of current demands (1mA, 20mA and 100mA) and a fixed sensing cycle of 100ms. In the case of laboratory tests, we modified the supply current from 1mA to 4mA. In the case of real-world validation, the available power supply was determined by the harvesting capability of the AM-5608CAR solar panel.

5.1 Adapting to Platform Heterogeneity

We evaluate AsTAR with respect to three key dimensions of platform heterogeneity: (i.) available power supply, (ii.) peripheral power draw and (iii.) capacitor size. In all cases the goal of AsTAR is to ensure consistent and sustainable operation in the face of this heterogeneity and without the need for any platform specific modeling. Figures 4 to 6 follow a common timeline which begins at 0 minutes with motes at 3.3V, in the LV state. AsTAR then charges motes to the OV state, which it then aims to maintain until power is shut off at 200 minutes. At that point, AsTAR detects the lack of power and rapidly throttles task execution rates. Each graph shows the super-capacitor voltage, which is presented on a linear scale, and the task execution rate, which is presented on a log scale due to its large variance. It should be noted that our power supply hardware has an inherent inaccuracy of up to 0.05mA in all configurations, which causes a low level of background ‘noise’ in all of our experiments.

Heterogeneity in Power Supply: Figure 4 shows the impact that different power supplies have on the operation of AsTAR with a fixed 5F super-capacitor and 20mA peripheral load. Charging from 3.3V to 3.7V takes 50 minutes using a 1mA supply with an average execution rate of 9.5 executions per minute, 42 minutes at 2mA with an average execution rate of 31.9 executions per minute and 24 minutes at 4mA with an average execution rate of 74.5 executions per minute. It should be noted that the mapping between the input power level and the resulting task rate is indirect as the goal of AsTAR is only to guarantee that charge accumulates during each adaptation period, rather than attempting to guarantee the rate at which charge accumulates. The classic saw-tooth pattern of Additive Increase Multiplicative Decrease (AIMD) [5] can be seen most clearly in the case of the 1mA and 2mA supplies, while in the case of the 4mA supply, charging is so rapid that the multiplicative decrease state is not triggered before the Optimal Voltage (OV) is reached. For all power supply levels, it can be seen that AsTAR ensures a smooth accumulation of charge and is extremely stable in the OV state, demonstrating a maximum deviation of 0.03V or 0.81% from the optimum. When power is cut off at 200 minutes, AsTAR rapidly backs off in under 5 minutes to the minimum task execution rate of once per minute, where it remains until power is either restored or voltage levels fall into the Shut-off Voltage (SV) state at 2.7V. AsTAR clearly achieves its optimization goals of acquiring and sustaining an optimal charge in the face of 4x heterogeneity in power supply current.

Heterogeneity in Peripheral Power Demand: Figure 5 shows the impact that different peripheral power demands have on the operation of AsTAR with a fixed 5F super-capacitor and a 1mA power supply. Charging from 3.3V to 3.7V takes 49 minutes using a 5mA peripheral load with an average execution rate of 26.8 executions per minute, 54 minutes with a peripheral load of 20mA using an average execution rate of 8.7 executions per minute and 70 minutes with a peripheral load of 100mA using an average execution rate of 2.7 executions per minute. For all peripheral test loads, AsTAR ensures a smooth accumulation of charge and remains stable in the OV state, with a maximum deviation of 0.02V or 0.54%. When power is cut off, AsTAR backs off to the minimum task execution rate in a worst case of 8 minutes. These results show that AsTAR remains robust in the face of heterogeneous peripheral loads, such as those arising from sensor, actuator and radio peripherals. In the case of platforms which support the plug-and-play modification of peripherals at runtime such as Flicker [9] and μPnP [24], AsTAR will even adapt this behavior at runtime as described in Section 5.2.

Heterogeneity in Capacitor Size: Figure 6 shows the impact that different capacitor sizes have on the operation of AsTAR with a fixed 20mA peripheral load and a 1mA power supply. Charging from 3.3V to 3.7V takes 41 minutes using a 1F capacitor with an average execution rate of 16.1 executions per minute, 48 minutes with 2.5F capacitor and an average execution rate of 9.9 executions per minute and 55 minutes with a 5F capacitor and an average execution rate of 7.1 executions per minute. These values approximate the ratios of the capacitor sizes. For all capacitors, AsTAR ensures a smooth accumulation of charge, stable operation in the OV state (maximum deviation of 0.03V or 0.81%) and rapidly throttles task execution rates to their minimum rate within 5 minutes of power being cut off. As can be seen from Figure 6, the major trade-off in capacitor sizes is how quickly they discharge when power is removed, which varies in direct proportion to their capacitance.

Discussion: As can be seen from the evaluation presented above, AsTAR is capable of ensuring sustainable energy harvesting operation even in the face of heterogeneity of power supplies, peripherals and energy storage capacity. AsTAR ensures that devices rapidly accumulate and then maintain an optimum charge level, adjusting quickly when power is

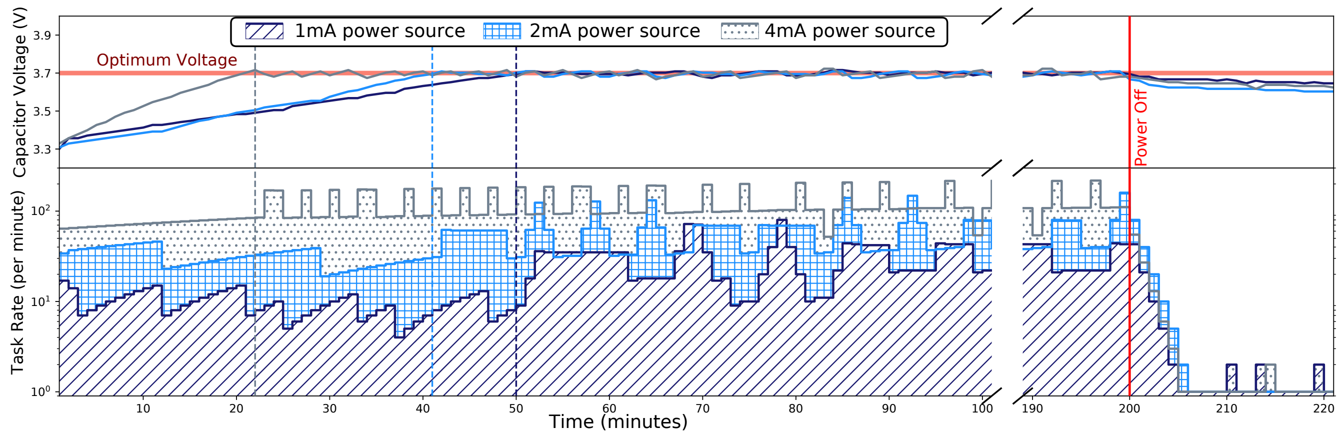

Figure 4. Heterogeneous power supply: 1mA, 2mA and 4mA with fixed 5F capacitor and 20mA peripheral load

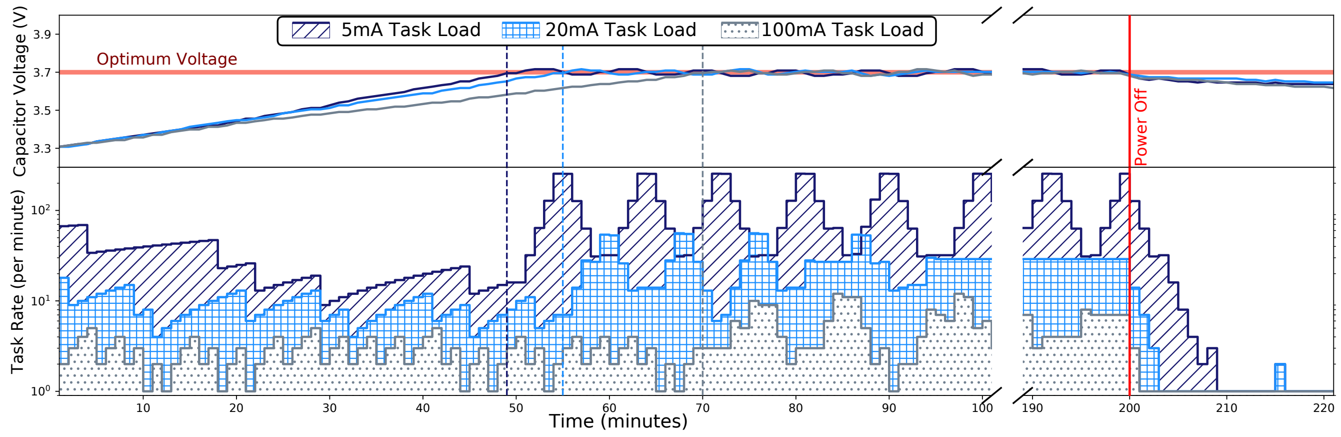

Figure 5. Heterogeneous peripheral demand: 5mA, 20mA and 100mA with fixed 5F capacitor and 1mA power supply

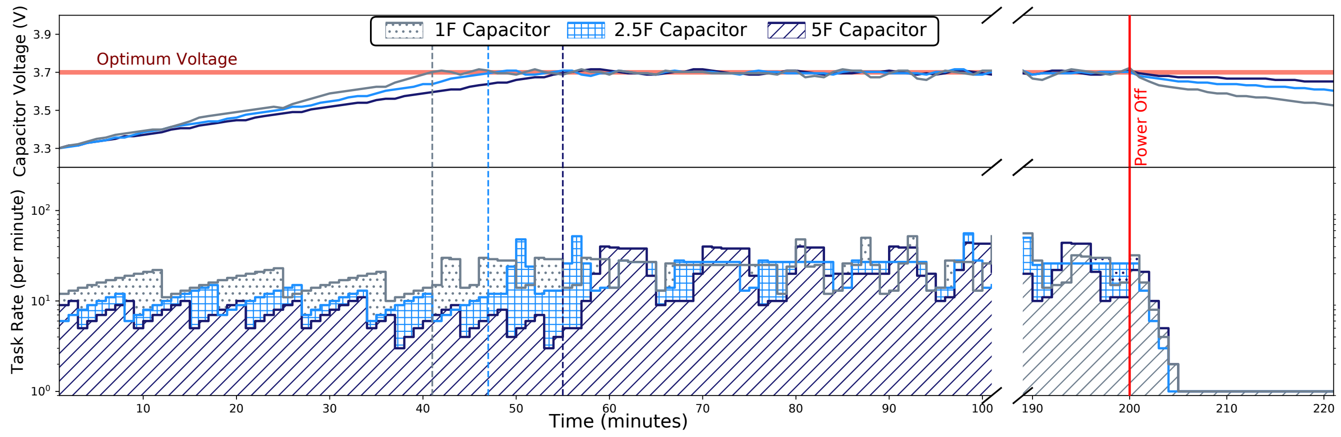

Figure 6. Heterogeneous capacitor sizes: 1F, 2.5F and 5F with fixed 20mA peripheral load and 1mA power supply

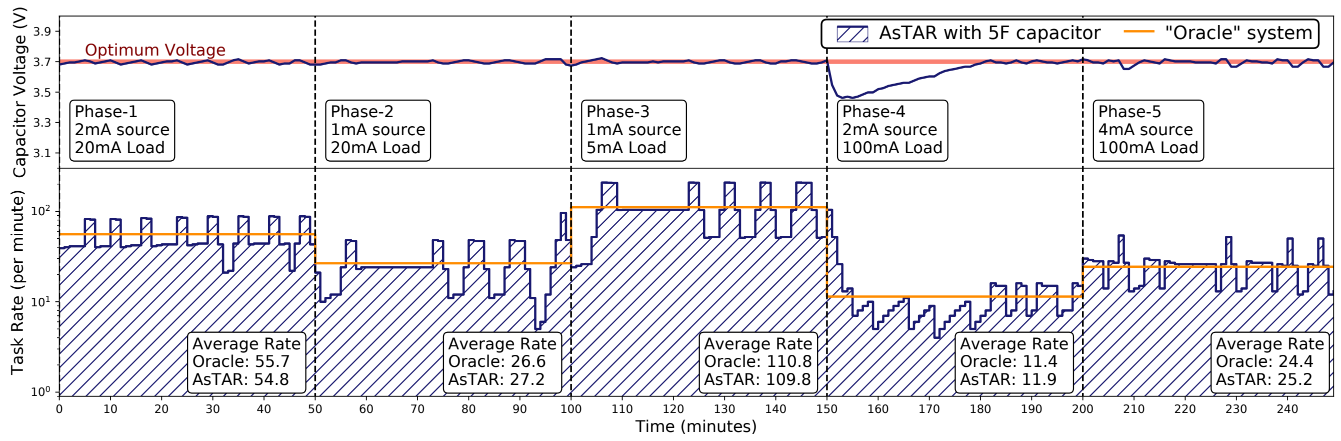

Figure 7. Robustness of AsTAR to runtime dynamism in power supply and demand, in comparison with the Oracle system

lost. Considering the experiments performed above, it can be seen that AsTAR allows developers considerable freedom in their energy harvesting designs, while providing consistent behavior. This enables developers to, for example, select larger capacitors for applications with longer typical power outages or smaller capacitors where rapid re-charging after power outages was considered more important. Considering peripherals, developers are relieved form peripheral-specific power consumption concerns. This is critical for plug-and-play platforms [9] [24], for which new peripherals may be developed even after platforms have been deployed in the field. AsTAR achieves its goals without the need for any platform-specific benchmarking or modeling, which we believe eliminates a significant barrier to entry for the building sustainable energy harvesting systems. In Section 5.2, we next evaluate the robustness of AsTAR in the face of runtime dynamism.

5.2 Adapting to Runtime Dynamism

To evaluate the effect of runtime dynamism on the behavior of AsTAR, we designed an evaluation scenario comparing AsTAR's scheduling performance against a theoretically ideal scheduler: the Oracle. The Oracle knows exactly when and how much energy is going to be available, as well as the amount of energy one task operation consumes, so it always makes perfect scheduling decisions about task operating rates, spending exactly the same amount of energy as is gathered from harvester. The Oracle is also free to schedule tasks with perfect precision, allowing for arbitrary sample periods. In this way the Oracle maintains an optimum charge level at all times with sufficient energy income.

In this evaluation, our platform is equipped with a fixed 5F capacitor, and the Oracle is executed after all experimental data is available with perfect knowledge. The evaluation starts with the system at optimum charge. We then unpredictably modified both power demand and available supply by connecting different peripheral loads and varying the power supply current. Afterwards we compare the scheduling performance of AsTAR against the Oracle.

Figure 7 shows the result in this evaluation, and illustrates the five phases of our experiment:

- ***Phase-1***: The experiment begins in the OV state with a 20mA peripheral load and a 2mA power supply. AsTAR configures task execution rate at an average of 54.8 executions per minute, which is 1.62% less than the Oracle.
- ***Phase-2***: At 50 minutes, the supply current is cut by 50% to 1mA. The Oracle immediately changes the task execution rate, AsTAR follows quickly and throttles task execution rates to 27.2 executions per minute on average, which is 2.26% more. The maximum deviation from the optimal voltage level during this supply-side reconfiguration was 0.02V or 0.54%. The sub-optimality caused by this reconfiguration is in fact too small to distinguish from the normal variance that arises from the 0.05mA inaccuracy of our power supply.
- ***Phase-3***: At 100 minutes, we replace the peripheral, reducing power demand by 75% from 20mA to 5mA. AsTAR reconfigures the task execution rate to 109.8 executions per minute on average, which is 0.9% less than the Oracle, and voltage levels remain within 0.023V or 0.62% of the optimum.
- ***Phase-4***: At 150 minutes, we apply an extreme runtime reconfiguration by increasing the power supply by 100% from 1mA to 2mA, while at the same time increasing peripheral power demand by 1900% to 100mA. The Oracle changes the task execution rate immediately, and keeps the charge level unchanged. On the other hand, without the knowledge of incoming energy change, AsTAR schedules tasks too frequently at the very beginning of this phase, and this leads to a voltage drop of 0.2V or 5.4% deviation from the optimum voltage level. In response to the voltage drop, AsTAR rapidly decreases the task execution rate to compensate for the charge loss, until the charge is fully restored after 30 minutes. Although task execution rates are temporarily higher at beginning, AsTAR quickly reschedules at a lower average execution rate during the voltage

drop period, and the average rate over the entire phase is 11.9 executions per minute, which is 4.39% more than the Oracle.

- ***Phase-5***: At 200 minutes, we increase the power supply by 100% from 2mA to 4mA, while keeping the 100mA load constant. AsTAR reconfigures the task execution rate to 25.2 executions per minute with minimal deviation from the optimal, which is 3.28% more than the Oracle. It should be noted that with a larger power supply and demand variance in voltage level is greater at 0.048V or 1.3%.

Discussion: Considered in sum, Figure 7 shows that AsTAR reacts rapidly to runtime dynamism, ensuring that charge levels remain optimum for the vast majority of the experiment timeline. Throughout the experiment, AsTAR is remarkably robust to changes in power supply and demand, recovering quickly even in the case of very large runtime reconfigurations that impact both the supply and demand side of the energy equation. Comparing to the Oracle which offers ideal performance, AsTAR may over-shoot or undershoot at the beginning of each phase, but it quickly reacts to the situation and restores the optimum charge, and remains close to the ideal task execution rate on average. As the task scheduling problem is to find an optimum task rate, AsTAR may under, or over-shoot the Oracle at a given time.

The results of our evaluation firstly validate our strategy of focusing upon changes in stored charge, rather than the supply or demand side and secondly eliminates the significant problem of dynamism for developers, even in cases where peripherals may change at runtime. Section 5.3 applies AsTAR in a real-world solar energy harvesting case study in order to validate that these properties hold outside of the lab.

5.3 Real World Validation

This section evaluates AsTAR in a real-world energy harvesting scenario. We deployed a solar powered AsTAR device running the software stack next to a window of our office building for a period of two days. This device is equipped with a 5F capacitor and a sensing peripheral that consumes 100mA for 100ms when sampling. In addition to monitoring capacitor voltage and task scheduling rates, we also logged light levels on a separate device using a standard light sensor that is deployed at the same location. It should be noted that this light sensor provides only an indication of available solar energy rather than a one-to-one mapping of the energy produced by the solar panel. As our office is surrounded by other buildings, direct sunlight is available for only a small part of the day, with reflected light for the remainder of the time. During the two days of the experiment the weather was predominantly cloudy with intermittent sunny periods.

Figure 8 shows the results of this experiment. Capacitor voltage is plotted at the top, followed by task scheduling rates in the middle and light levels at the bottom. Capacitor voltage is plotted on a linear scale. Task rates and light levels were plotted on a log scale due to their large variance. This figure again shows that AsTAR achieves its goals of acquiring and sustaining an optimal charge whenever sufficient energy is available, and even in the face of large fluctuations in available energy. We now highlight interesting elements from our two-day experiment.

- ***The sun rises*** at around 7AM on both days. After a night of darkness, the mote is in the SV mode in order to preserve operation and no application tasks are scheduled. As the sun rises, light levels quickly increase to around 1000 lux and the capacitor charges past the SV threshold.
- ***Indirect sunlight***: LV mode begins at around 10AM on both days as the mote voltage rises past the shut-off threshold of 2.7V. The AsTAR scheduler applies AIMD to reach an optimal task rate of approximately 2.5 executions per minute while the capacitor charges in somewhat consistent indirect sunlight.
- ***Direct sunlight*** falls upon the panel at around noon for a period of between one to two hours, depending on weather conditions. This is clearly visible in the light level graph, which increases by more than an order of magnitude from under 1000 lux to over 15000 lux. This causes the mote to rapidly accumulate charge and reach the optimum level of 3.7V. AsTAR handles this by rapidly increasing task rates to 41 executions per minute, which sustains the voltage level at the optimum value of 3.7V, with a variance during this period of 0.07V or 1.9%. The larger variance during this period arises due to clouds unpredictably obscuring the sun.
- ***Falling light levels:*** When the period of direct sunlight ends in the early afternoon, light levels again fall by more than an order of magnitude from around 15000 lux to 1000 lux, which briefly pulls the voltage below OV, causing AsTAR to quickly adapt the task rate to an average of around 2.5 executions per minute. This adaptation is sufficient to ensure a deviation of less than 0.04V or 1.1% from the optimum.
- ***Increased reflected light*** on day two occurs due to high white clouds from late in the afternoon. Light levels more than double from 1000 lux to 2400 lux. This causes the voltage to briefly enter the HV zone and AsTAR applies MIAD to increase task rates to 4.6 executions per minute. This compensates for the increased light levels, resulting in a deviation of less than 0.02V or 0.55% from the optimum.
- ***The sun sets*** around 7PM on both days, causing the light level to quickly drop towards 0 lux, which prevents the accumulation of charge. AsTAR reacts by rapidly cutting the task rate using *Multiplicative Decrease* to the developer-specified minimum of 1 task execution per minute.
- ***Night time power saving*** begins between midnight and 1AM as the capacitor voltage falls below 2.7V, causing AsTAR to enter the SV mode until the sun rises again and charges the capacitor past the LV threshold. As can be seen from the figure, disabling the task significantly slows the discharge rate of the capacitor, enabling the

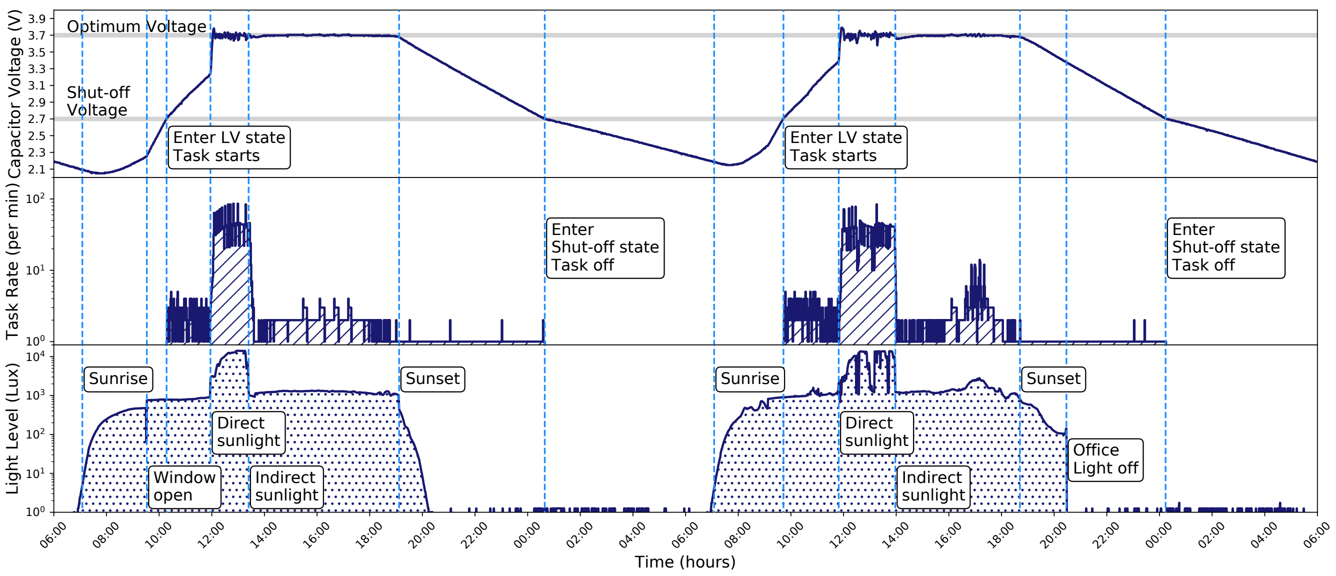

Figure 8. A section of 2 full days from the real-world deployment. The AsTAR device is equipped with a 5F capacitor, a peripheral that demands 100mA current, and it uses only solar energy harvested from indoor location. The three curves each shows the capacitor voltage, the application task rate, and light level recorded by light sensor.

device to survive 12 hours of darkness.

In all phases of the experiment, AsTAR behaved as expected. During this evaluation, it remained remarkably robust against the radical changes in available energy levels that arise due to specifics of the deployment location. This is accomplished without the need for any benchmarking and modeling. Furthermore, while solar modeling approaches [22] [3] provide an accurate prediction of solar energy at a given location and time-of-year, they cannot take into account deployment specifics such as the impact of buildings and local weather.

5.4 Overhead

This section isolates the overhead of AsTAR in relation to the underlying software stack in terms of: memory, computation and energy. As discussed in Section 3.1 and shown in Listing 1, the AsTAR scheduler is extremely lightweight. In each adaptation cycle, it only executes a small number of single-cycle instructions: *comparison*, *increment*, *decrement*, *multiply by 2 (i.e. left shift)*, and *divide by 2 (shift right)* and one analog read operation.

Table 1 shows the time and charge required to compute adaptations in each state. As the execution period was smaller than the temporal resolution of our laboratory equipment, we executed each adaptation 10 times and provide the average. The associated charge consumption is based upon the power numbers specified in the LTC5800 data sheet. The primary power cost arises from the active time of the processor, during which it consumers 1.3mA.

The AsTAR software runtime requires only 8kB of flash memory and 8 bytes of RAM for each task being managed. Considered in sum, AsTAR is extremely efficient and has very limited overhead in terms of computation, energy and memory.

Table 1. Time and charge required for processing AsTAR scheduler

	Time (μs)	Charge (μC)
Shut-off state	21.70	0.028
Low Voltage state	104.75	0.136
Optimum Voltage state	87.80	0.114
High Voltage state	121.70	0.158
Average	83.99	0.109

6 Conclusions and Future Work

This paper introduced AsTAR, a novel approach to building energy harvesting Internet of Things applications. AsTAR contributes a *(i.)* simple, yet effective energy-aware task scheduling scheme for on IETF Class-1 devices and *(ii.)* a reference platform for experimentation with sustainable energy harvesting applications.

Our evaluation shows that AsTAR is capable of reliable and sustainable operation under both laboratory tests and a real-world solar energy harvesting case-study. Our self-adaptive task scheduling approach is an elegant, simple and robust approach to optimizing task schedules in the face of heterogeneity and changing operational conditions.

It is important to note that AsTAR achieves these benefits in a fully autonomic manner with zero modeling prior to device deployment and extremely low runtime overhead. Considered as a whole, we believe that the features of AsTAR represent a significant reduction in the complexity of writing energy-aware IoT applications for energy harvesting platforms. We note however, that AsTAR's applicability is limited to those tasks that can be independently scheduled.

Our future work will follow three main scientific tracks: Firstly, by extending AsTAR's task management concept from the application layer, deeper into the OS and network stack, we aim to provide greater adaptability to changing environmental conditions and application demands. For example, by modifying network configurations and routes to maximize the role that is played by nodes with abundant energy. Secondly, we plan to extend AsTAR with support for more energy harvesting techniques, including thermal, RF and kinetic energy. Finally, we will develop techniques that allow developers to prioritize the execution of tasks.

Complementing these scientific tracks, we plan an engineering track, which will establish a large-scale AsTAR testbed that we will make available to the IoT research community in order to accelerate the development of energy harvesting algorithms, tools and applications.

7 References

[1] ANALOG DEVICES, INC. Ltc5800 datasheet. http://www.analog.com/media/en/technical-documentation/data-sheets/5800ipmfa.pdf, 2015. [Accessed: December 2018].

[2] N. A. Bhatti, M. H. Alizai, A. A. Syed, and L. Mottola. Energy harvesting and wireless transfer in sensor network applications: Concepts and experiences. *ACM Trans. Sen. Netw.*, 12(3):24:1–24:40, Aug. 2016.

[3] B. Buchli, F. Sutton, J. Beutel, and L. Thiele. Towards enabling uninterrupted long-term operation of solar energy harvesting embedded systems. In *Proceedings of the 11th European Conference on Wireless Sensor Networks - Volume 8354*, EWSN 2014, pages 66–83, New York, NY, USA, 2014. Springer-Verlag New York, Inc.

[4] Q. Cao, D. Fesehaye, N. Pham, Y. Sarwar, and T. Abdelzaher. Virtual battery: An energy reserve abstraction for embedded sensor networks. In *2008 Real-Time Systems Symposium*, pages 123–133, Nov 2008.

[5] D.-M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1):1 – 14, 1989.

[6] A. Eswaran, A. Rowe, and R. Rajkumar. Nano-rk: An energy-aware resource-centric rtos for sensor networks. In *Proceedings of the 26th IEEE International Real-Time Systems Symposium*, RTSS '05, pages 256–265, Washington, DC, USA, 2005. IEEE Computer Society.

[7] A. Gaglione, D. Rodenas-Herraiz, Y. Jia, S. Nawaz, E. Arroyo, C. Mascolo, K. Soga, and A. A. Seshia. Energy neutral operation of vibration energy-harvesting sensor networks for bridge applications. In *Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks*, EWSN &#8217;18, pages 1–12, USA, 2018. Junction Publishing.

[8] J. Hester, L. Sitanayah, and J. Sorber. 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, pages 5–16, New York, NY, USA, 2015. ACM.

[9] J. Hester and J. Sorber. Flicker: Rapid prototyping for the batteryless internet-of-things. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, SenSys '17, pages 19:1–19:13, New York, NY, USA, 2017. ACM.

[10] J. Hsu, S. Zahedi, A. Kansal, M. Srivastava, and V. Raghunathan. Adaptive duty cycling for energy harvesting systems. In *Proceedings of the 2006 International Symposium on Low Power Electronics and Design*, ISLPED '06, pages 180–185, New York, NY, USA, 2006. ACM.

[11] D. Hughes, P. Greenwood, C. G, and G. Blair. Gridstix: supporting flood prediction using embedded hardware and next generation grid middleware. In *2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks(WoWMoM'06)*, pages 6 pp.–626, 2006.

[12] IETF. Terminology for constrained-node networks. https://tools.ietf.org/html/rfc7228#section-3, 2014. [Accessed: December 2018].

[13] A. Lachenmann, P. J. Marrón, D. Minder, and K. Rothermel. Meeting lifetime goals with energy levels. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, SenSys '07, pages 131–144, New York, NY, USA, 2007. ACM.

[14] A. Matt, O. Pierce, B. Emma, and R. Margaret. Iot global forecast and analysis, 2015-2025. https://www.gartner.com/doc/3659018/iot-global-forecast-analysis-, 2017. [Accessed: December 2018].

[15] N. Matthys, F. Yang, W. Daniels, S. Michiels, W. Joosen, D. Hughes, and T. Watteyne. μpnp-mesh: The plug-and-play mesh network for the internet of things. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 311–315, Dec 2015.

[16] P. Michele, S. H. Jeffrey, D. Charlie, M. Paul, B. Jennifer, A. A. Julie, F. Nigel, E. G. Frank, H. Thomas, V. Merritt, Maxim with Christopher, G. Clare, and L. Diane. Predictions 2018: Iot moves from experimentation to business scale. https://www.forrester.com/report/Predictions+2018+IoT+Moves+From+Experimentation+To+Business+Scale/-/E-RES139752, 2017. [Accessed: December 2018].

[17] G. S. Ramachandran, N. Matthys, W. Daniels, W. Joosen, and D. Hughes. Building dynamic and dependable component-based internet-of-things applications with dawn. In *2016 19th International ACM SIGSOFT Symposium on Component-Based Software Engineering (CBSE)*, pages 97–106, April 2016.

[18] H. Ritter, J. Schiller, T. Voigt, A. Dunkels, and J. Alonso. Experimental evaluation of lifetime bounds for wireless sensor networks. In *Proceeedings of the Second European Workshop on Wireless Sensor Networks, 2005.*, pages 25–32, Jan 2005.

[19] A. Roy, S. M. Rumble, R. Stutsman, P. Levis, D. Mazières, and N. Zeldovich. Energy management in mobile devices with the cinder operating system. In *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, pages 139–152, New York, NY, USA, 2011. ACM.

[20] P. Sawyer, N. Bencomo, D. Hughes, P. Grace, H. J. Goldsby, and B. H. C. Cheng. Visualizing the analysis of dynamically adaptive systems using i* and dsls. In *Second International Workshop on Requirements Engineering Visualization (REV 2007)*, pages 3–3, Oct 2007.

[21] J. Sorber, A. Kostadinov, M. Garber, M. Brennan, M. D. Corner, and E. D. Berger. Eon: A language and runtime system for perpetual systems. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, SenSys '07, pages 161–174, New York, NY, USA, 2007. ACM.

[22] J. Taneja, J. Jeong, and D. Culler. Design, modeling, and capacity planning for micro-solar power sensor networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, IPSN '08, pages 407–418, Washington, DC, USA, 2008. IEEE Computer Society.

[23] T. Watteyne, L. Doherty, J. Simon, and K. Pister. Technical overview of smartmesh ip. In *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 547–551, July 2013.

[24] F. Yang, N. Matthys, R. Bachiller, S. Michiels, W. Joosen, and D. Hughes. μpnp: Plug and play peripherals for the internet of things. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15, pages 25:1–25:14, New York, NY, USA, 2015. ACM.

[25] H. S. Zhu, C. Lin, and Y. D. Liu. A programming model for sustainable software. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ICSE '15, pages 767–777, Piscataway, NJ, USA, 2015. IEEE Press.