Department of
Information Engineering
and Computer Science
DISI

UNIVERSITY
OF TRENTO - Italy

DISI - Via Sommarive, 5 - 38123 POVO, Trento - Italy
http://disi.unitn.it

SENSEH: FROM SIMULATION TO DEPLOYMENT OF ENERGY HARVESTING WIRELESS SENSOR NETWORKS

Riccardo DallOra, Usman Raza, Davide Brunelli, and Gian Pietro Picco

April 2014

Technical Report # DISI-14-005

SensEH: From Simulation to Deployment of Energy Harvesting Wireless Sensor Networks

Riccardo Dall'Ora*, Usman Raza*†, Davide Brunelli* and Gian Pietro Picco*
*University of Trento, Italy — E-mail: {riccardo.dallora, davide.brunelli, gianpietro.picco}@unitn.it
†Bruno Kessler Foundation, Trento, Italy — E-mail: raza@fbk.eu

***Abstract*—Energy autonomy and system lifetime are critical concerns in wireless sensor networks (WSNs), for which energy harvesting (EH) is emerging as a promising solution. Nevertheless, the tools supporting the design of EH-WSN are limited to a few simulators that require developers to re-implement the application with programming languages different from WSN ones. Further, simulators notoriously provide only a rough approximation of the reality of low-power wireless communication.**

In this paper we present SENSEH, a software framework that allows developers to move back and forth between the power and speed of a simulated approach and the reality and accuracy of in-field experiments. SENSEH relies on COOJA for emulating the actual, deployment-ready code, and provides two modes of operation that allow the reuse of exactly the same code in real-world WSN deployments. We describe the toolchain and software architecture of SENSEH, and demonstrate its practical use and benefits in the context of a case study where we investigate how the lifetime of a WSN used for adaptive lighting in road tunnels can be extended using harvesters based on photovoltaic panels.

I. INTRODUCTION

Wireless sensor networks (WSNs), whose market value is growing steadily, are delivering their promise of enabling new, low-cost, and ubiquitous sensing applications. However, one of the main limiting factors of current applications comes from energy autonomy, which typically keeps the system lifetime (or the mean-time-before-maintenance) very far from the tens of years expected by industry stakeholders. It is therefore not surprising that the idea of equipping WSN nodes with an energy harvesting (EH) subsystem has gained quite some momentum in recent years. Scavenging energy directly from the environment around WSN nodes appears to be an ideal solution to alleviate, and possibly solve, the energy problem. Indeed, in many cases the energy density—whether solar, wind, vibrational or thermal in nature—available in the target environment is often compatible with the energy demands of the low-power devices participating in the WSN application.

While energy harvesting appears to be a very promising technology for extending WSN lifetime, new tools are required for accelerating time-to-market for these systems. Indeed, to ease the industrial development of Energy Harvesting WSNs (EH-WSNs), pre-prototyping tools must provide fast and reliable results at low cost, to inform appropriate hardware design choices or validate them. Furthermore, prototyping an application by deploying complete EH-WSN nodes, including their sensing, harvesting, and communication subsystems, is both a lengthy and expensive process, possibility inhibiting the wide-scale development of EH-WSNs.

For these reasons, a simulation approach is often preferred over hardware prototyping to limit costs and also to enable a precise analysis of each component's internal states. This approach also allows to investigate a greater system scale, the evaluation of design tradeoffs for system components (hardware and/or software) that may even not be physically implemented yet, the ability to replay specific scenarios or study the influence of given environmental parameters, and finally the possibility of observing the WSN behavior over extended periods of time. These benefits motivate the recent emergence of simulation tools for EH-WSNs we concisely survey in Section II. Simulation has its drawbacks, however, most notably the fact that all existing tools require that the simulated system is implemented in a programming framework (e.g., C++ or Matlab) different from the one of the final WSN implementation (e.g., TinyOS/nesC or Contiki/C). This makes the execution of simulations fast and efficient, but significantly widens the gap between the behavior of the EH-WSN that is simulated and the one that is actually deployed. The gap may be so significant that the considerations about energy sustainability derived in the former may actually become invalid once confronted with the reality of the latter. A typical example is provided by the simulation of wireless communication, which is known to provide only a very rough approximation of the real conditions, which are strongly affected by the deployment environment.

In contrast, SENSEH improves over the current state by providing a software framework that allows developers to move back and forth between the power and speed of a simulated approach and the reality and accuracy of in-field experiments, as shown in Table I. Further, this is achieved through two modes of operation, depending whether a real harvester is actually available or not. For simplicity, we call these modes MEMORY and PINS, for reasons that will become evident next.

In MEMORY mode, the harvester is only *simulated*: a real harvester does not necessarily exist, and in any case its

harvester	WSN nodes	environment	interface
simulated	simulated	simulated	MEMORY
simulated	real	simulated or real	MEMORY
simulated	simulated	simulated	PINS
real	real	real	PINS

TABLE I
SENSEH: FROM SIMULATION TO THE REAL-WORLD—AND BACK.

integration at code level is not a concern. SENSEH is designed in such a way that the *same* code using (and including) the simulated harvester can be used both in a simulator *and* in a real deployment. This allows developers to test the system first through the more efficient and scalable means offered by simulation, and then validate the results in the more realistic environment provided by an actual deployment—something that normally requires real harvesters, which is both expensive and risky in an experimentation phase where the "right" harvester is to be selected. For instance, this is very useful to verify whether the actual connectivity affects the estimates derived in simulation, e.g., due to interference or other causes generating communication overhead. The harvester operation depends on environmental parameters. Moreover, we use of serial communication to feed the harvester with traces from sensors (e.g., light for solar harvesters, temperature for thermal ones) to replicate real-world environmental trends affecting energy density. This is particularly useful during intermediate development steps where the WSN is tested in a testbed, which often provides out-of-band communication via USB cables. In the case of an in-field deployment, we also provide the option to acquire environmental parameters directly from on-board sensors, specified at configuration time.

The PINS mode, instead, is conceived to support the same ability to switch from simulation to the real-world, but in the case where a real harvester is available. In this case, we support through different mechanisms w.r.t. the first mode, the simulation of a WSN application where the MCU of nodes access directly the harvester via pin-level communication.

The toolchain and software architecture of SENSEH are described in Section III. From an implementation standpoint, our ability to directly reuse the code simulated into real-world is enabled by our reliance on COOJA [1] and specifically MSPSim, a hardware emulator for the MSP430 MCU. However, our first mode of operation assumes that some components simulating the behavior of the harvester are actually part of the binary deployed on the real WSN nodes. We achieve this goal by supporting the Contiki operating system, although it is straightforward to port our code to TinyOS or other operating systems. For the second mode of operation, instead, we directly extend COOJA, and therefore inherit its ability to simulate directly at the binary level applications written for either TinyOS or Contiki.

We illustrate concretely the use of our system through a case study, described in Section IV, concerned with a WSN application installed in a road tunnel [2], [3]. In this context, we analyze the effect of equipping the WSN nodes with photovoltaic panels. As light is non-uniform during the day and along the tunnel, this affects the energy density available to harvesting, and in turn the sustainability of our network. This case study is also the opportunity to reassert quantitatively the difference between findings from simulated and real-world experiments. Section V ends the paper with brief concluding remarks.

	Harvester	**Energy Storage**	**Power Consumption**	**Emulation**
Wu et al. [4]		✓	✓	
Sanchez et al. [5]	✓	✓	✓	
HarvWSNet [6]	✓	✓	✓	
GreenCastalia [7]	✓	✓		
WSNsim [8]	✓	✓	✓	
Jeong & Culler [9]	✓	✓	✓	
SIVEH [10]	✓	✓		
PASES [11]	✓	✓	✓	
TOSSIM [12]				✓
COOJA/MSPSim [1]				✓
SENSEH	✓	✓	✓	✓

TABLE II
SENSEH VS. STATE OF THE ART.

II. RELATED WORK

The recent interest in EH-WSN has determined a surge of approaches providing simulation support for these systems, typically through extension of available simulators.

Generic network simulators such as ns-3, OMNeT++, or OPNET, are not well-suited for energy-aware simulations of WSNs. The support for models of the energy harvester and storage as well as power consumption, all crucial to EH-WSNs, is limited or even absent. Several extensions of these generic simulators are proposed to incorporate energy models. Wu et al. [4] extend ns-3 with models for the node power consumption and its consequent effects on the battery; Sanchez et al. [5] additionally provides also a model of a solar harvester based on empirically collected radiation data.

Other approaches are instead based on existing WSN simulators, or develop dedicated ones. HarvWSNet [6] is an extension of WSNet [13]. It implements models for harvester, battery, and power consumption in Matlab, and interfaces them with the network stack of WSNet via TCP sockets. GreenCastalia [7], is an extension of Castalia [14], itself an extension of OMNeT++ providing realistic channel models. GreenCastalia provides modeling for multi-source, multi-storage EH architectures. It models both an ideal and empirical energy storage. WSNsim [8] is a standalone simulator, whose flexible software structure allows integration of diverse models for EH-WSNs. However, it consists only of a prototype implementation not currently available to research community.

Finally, PASES [11] is a flexible design space exploration framework which features also accurate power consumption analysis of WSNs. It does not target specific platforms and needs detailed power models of the digital architectures on board (i.e., MCU, memory, radio, sensors) to provide statistics about the power consumption and suggest the optimal hardware configuration. Its cycle-accurate because it is based on the SystemC framework, which significantly increases the simulation time. In a similar vein, system-level simulators [9], [10] trade simulation speed for more accurate modeling of low-level details of the harvesting system. Complexity of these models is a major obstacle in adopting them for the assessment

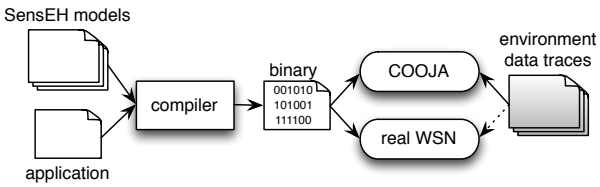


Fig. 1. SENSEH toolchain for MEMORY mode.

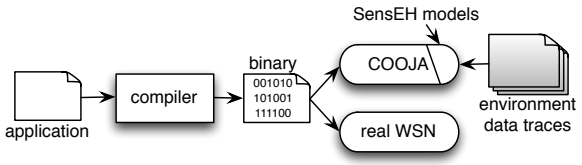


Fig. 2. SENSEH toolchain for PINS mode.

of EH systems.

Interestingly, some well-known WSN simulators such as TOSSIM [12] and COOJA are designed to use directly (through emulation) binary, deployment-ready code, and therefore provide the ability to move between simulated and real experiments. TOSSIM is designed for running only TinyOS based applications, while COOJA coupled with MSPSim, a hardware emulator for MSP430-based motes, allows simulation of both Contiki and TinyOS applications. However, neither include support for modeling power consumption or battery discharge, let apart energy harvesting.

Table II summarizes the related work we just surveyed, and compares against SENSEH. The table visually reasserts how SENSEH combines the emulation capabilities of COOJA with the models for power consumption, battery, and harvester similar to other simulation approaches. The combination of the two provides developers with a flexible tool enabling one to move back and forth between simulated and in-field experiments, as described next. To the best of our knowledge, SENSEH is the first software framework to provide such capabilities.

III. SENSEH

In this section we describe in more detail our software framework, SENSEH. We begin by describing in Section III-A the toolchain or, better, the toolchains that enable the two modes of operation described in Section I. An integral element of these toolchains is the actual modeling of the harvester, and the associated profile for power consumption, described in Section III-B. Finally, in Section III-C we describe the software architecture, supporting both modes of operation.

A. Toolchains

As we anticipated in Section I, SENSEH supports two distinct modes of operation. In MEMORY mode, the harvester is only simulated, even in a real deployment, and is fed real traces of environmental data to compute the harvested power. In PINS mode, the application accesses the harvester directly via pin-level communication, and is therefore deployment-ready. These two modes of operation result in different toolchains. Both revolve around the following elements, which are however positioned in different places in each toolchain:

- *Harvester model.* Describes how the harvester (e.g., a solar panel) collects energy from the environment, based on some input parameters (e.g., light intensity).
- *Battery model.* Describes how the battery behaves when connected to the harvester, taking into account the discharging due to the node activity, but also the recharging determined by the harvester.
- *Power consumption model.* Describes the power consumption of the WSN node, based on information about the time it spends in the various states (e.g., radio active vs. idle).

The toolchain for MEMORY mode is shown in Figure 1. All of the aforementioned models are input, as Contiki/C source code, to the standard platform compiler, along with the application. Indeed, in MEMORY mode the behavior of the harvester is simulated not only when the compiled code is input in COOJA, but also when such code is deployed in a real WSN. The models must therefore be part of the binary deployed on the WSN nodes. The first two models are provided as part of SENSEH; the power consumption module, instead, reuses the Energest [15] part of the Contiki distribution. Input traces are fed to the harvester module via the serial port.

The toolchain for PINS mode is shown in Figure 2, and is essentially identical to the standard toolchain one would use to compile applications and then either simulate them with COOJA or directly deploy them in-field. The only difference is that, in this case, we provide a modified version of COOJA that provides support for the aforementioned models, and to process traces of environmental parameters affecting the harvester.

B. Modeling the Energy Harvester

An energy harvester normally consists of three main components: the micro transducer which converts environmental energy into electrical energy, the voltage shifter which regulates the generated voltage, and an accumulator. The micro generator and the conversion circuit are usually considered together as the real energy harvesting system, while batteries or other kind of energy storage need a separate model because they exhibit performance variations (e.g. state of charge, aging, leakage, number of charging/discharging cycles) during the lifetime of the system. In this paper we are considering the State of Charge (SoC); other features are future work.

Harvester. Models of energy harvester circuits often treat the micro-generator as a voltage/current source and the efficiency of the whole system depends mainly on the input power delivered by the transducer and on the required output voltage. When an accumulator is connected directly to the harvester, the output voltage of the latter is correlated with the state of charge of the former, which therefore has a runtime impact on the conversion efficiency of the harvester.

Moreover, another important feature, which must be considered in modeling energy harvesters, is the capability of the

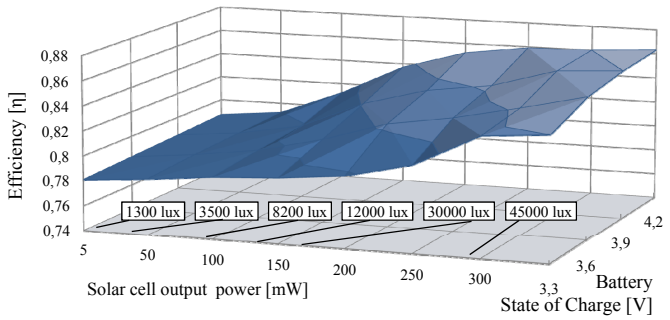


Fig. 3. Efficiency of the photovoltaic harvester as a function of the power intake and of the SoC of the battery.

scavenger to achieve the maximum conversion efficiency, under any environmental conditions and circuit states. Particular circuit implementations, called Maximum Power Point Trackers (MPPT), are responsible for maximizing the performance runtime.

In this paper, as an example, we provide the model of a photovoltaic (PV) harvester to SENSEH. Usually, when a PV module is directly connected to a load, the operating point is rarely at the maximum conversion efficiency. The principle of MPPT is to guarantee an impedance matching between the load and the PV module, and to regulate the DC/DC parameters so that the maximum available power is extracted and used to recharge the battery.

Hence, implementing a MPPT circuit boosts the average efficiency and the energy converted from the environment. On the other hand, it adds complexity to the harvester and causes additional losses due to more electronic components. For this reason, power losses must be kept remarkably lower than the incremental energy collected thanks to the MPPT. This is a critical condition which must be considered at design time and must be available in the models during simulations.

In conclusion, knowing the kind of harvester and its available features, is quite easy to extract a model, whose efficiency is dependent on the power input and on the State of Charge of the battery. A lookup table (LUT) can easily provide such a model. For example, Figure 3 shows the efficiency curve of the harvester used in SENSEH, and specifically in the case study of Section IV. For the sake of clarity, we indicated also the luminous emittance (lux) necessary to achieve specific input power values from the solar cell.

Batteries. A fundamental piece of information, needed when batteries are modeled, is the state of charge (SoC) of the accumulator. Many existing network simulations assume very simple battery models such as considering ideal energy storage devices. In this case, batteries are modeled as containers of finite capacity, containing a certain amount of energy units spent during the execution of application tasks, and added when the balance between the energy intake and the energy consumption is positive.

Real batteries, however, do not operate in such a simple way: they have a certain amount of non-ideal properties, that a simple energy model is not able to capture. These non-idealities strongly affect the behavior, the delivered capacity, and the lifetime of a real battery. For instance, all batteries suffer from self-discharge: a cell that simply sits on the shelf, without any connection between the electrodes, experiences a reduction in its stored charge due to internal chemical reactions, at a rate depending on the cell chemistry and the temperature. Batteries also have charge and discharge efficiency strictly less than one, i.e., some energy is lost when charging and discharging the battery. Additionally, batteries have some non-linear properties [16], such as temperature effects and recovery effects.

These properties should be taken into account when dimensioning and simulating energy harvesting systems, because they can easily lead to errors in the battery lifetime estimates. For example, if the harvester uses a rechargeable battery to store the energy from the environment, it is important to consider that the battery capacity reduction, at each recharge cycle, affects and reduces also the lifetime. To estimate the SoC of the batteries, on-line techniques, normally used in real operating devices, can be efficiently exploited as simulation models. They save complex computation, at the cost of less accuracy of the simulation results. In fact, these methods usually do not take into account aging and the change of parameter after several cycles.

SoC estimation techniques can be classified roughly into two main categories: *i)* direct voltage lookup tables (LUT) [17] derived from Peukert's equation; and *ii)* coulomb counting techniques [18]. LUT is a popular and simple method especially for WSNs [19], because it replaces the runtime computation with a simpler association $(V_{battery}, SoC)$ in an array. Since the LUT is usually pre-computed and static, this technique does not produce accurate results because it does not take into account the effect of aging and the variations of the component characteristics after several charging/discharges phases. Coulomb counting may lead to more accurate simulation results, but in this case the charge flowing in or out of the battery is counted by integrating the current over time. Therefore the simulator must take into account the current profile of each activity of the system (current profile of the load) and, concerning systems with energy harvesting, also the current intake from the harvester during the time of operation, with remarkable runtime computation. Our current implementation relies on the LUT technique.

C. Software Architecture

We designed the software architecture of SENSEH to be easily extendable to encompass various energy sources, storage, and harvesters. Both toolchains rely on the same high-level conceptual view of the relevant components, shown by the class diagram in Figure 4. The only difference between two toolchains, apart from the implementation language (C for MEMORY, Java for some of PINS modules) is how these components are used by the rest of the system, be it the application or the simulator.

1) Conceptual view: The architecture can be broadly divided into three parts, described next.

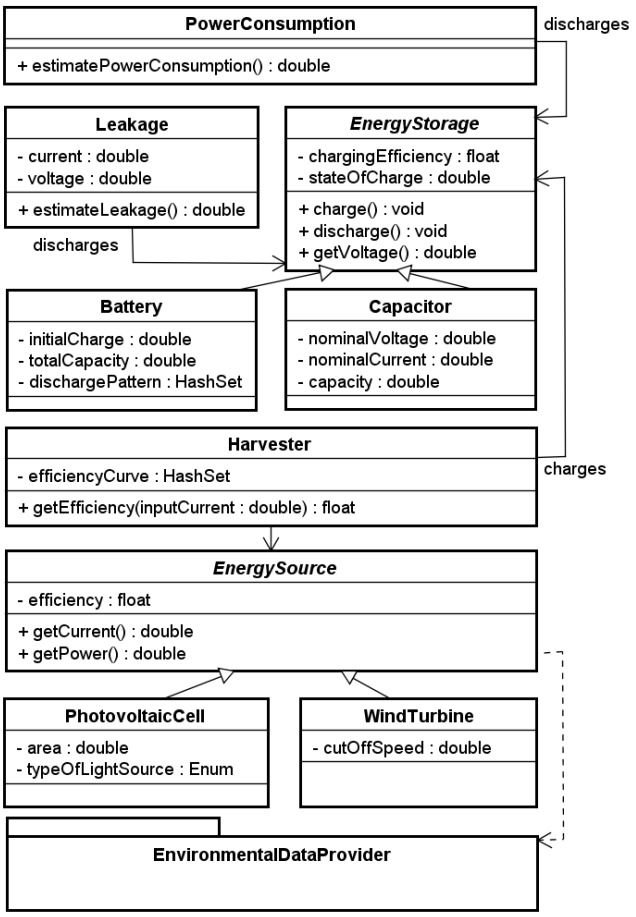


Fig. 4. Class Diagram of SENSEH.

Harvester. The behavior of the harvester is intimately connected with the one of the environment in which it is immersed, and from which energy is scavenged. These aspects are reified into two classes, `EnergySource` and `Harvester`.

The former is an abstract class, which can therefore accommodate different types of energy sources, and the related models. Of the examples shown in the picture, SENSEH currently provides support for harvesters based on photovoltaic cells, modeled using the LUT method described in Section III-B. The producer datasheet of the solar panels usually only provides output power for few reference light levels, therefore we apply a piecewise linear approximation to estimate the power at the intermediate points. The behavior of `EnergySource`, in turn, depends on the actual environmental parameters. These are fed to `EnergySource` via the package `EnvironmentalDataProvider`, which can be configured to feed data coming offline from pre-recorded experimental traces or online from on-board sensor readings.

Depending on the amount of current generated from `EnergySource`, the efficiency of the modeled harvester is different, as discussed in Section III-B. The `Harvester` class estimates the output power of the harvester considering an empirical efficiency curve, which is a non linear function of input source power. In our current implementation of the photovoltaic harvester, we rely on the harvester model described in [20]. This output power is then fed to the `EnergyStorage`, via its `charge` method.

Energy Storage. As in the case of `EnergySource`, `EnergyStorage` is an abstract class that can in principle accommodate multiple types of devices. In the current implementation we provide a model for a rechargeable `Battery`, but in principle (super)capacitors could also be included. `EnergyStorage` keeps track of the state of charge (SoC), which is affected by the two methods `charge` and `discharge`. The former is called from the `Harvester`, as discussed earlier. The second, instead, can be called from the `Leakage` class, which models the leakage current characteristic of the energy storage device, or by the `PowerConsumption` class, which models the discharge due to the actual operation of the WSN device. The former contribution to discharge is typically negligible w.r.t. the contribution of the latter, described next.

Power Consumption. An accurate profiling of the node power consumption is key to estimate whether the EH-WSN is energy-neutral. This functionality is provided by the `PowerConsumption` class, which effectively measures the time spent by each mote component in each of its operation modes (e.g., radio on for transmission or reception, MCU on or in low-power mode). By multiplying these time intervals by the power consumed by the respective mode we get an estimate of the amount of energy consumed by the nodes over time, which is therefore discharged from `EnergyStorage`.

2) Implementation details: Although they share the same conceptual architecture, the software implementations are different for each toolchain.

MEMORY . The code of the MEMORY version is written in C, to enable its direct integration in Contiki. The `PowerConsumption` module reuses the Energest [15] module already provided inside Contiki, which provides the power profiling required.

`EnvironmentalDataProvider` relies directly on the on-board functionality of the WSN node. In the case of offline input via recorded traces, these are fed through the serial port; if real sensed data are to be acquired instead, this is obtained via direct access to the sensor specified at configuration time. To retain the ability to use unmodified code, the same solution is adopted also when using COOJA.

PINS . In this variant of SENSEH, the implementations of `EnergyStorage`, `EnergySource`, and `Harvester` are directly provided as extensions to COOJA, implemented in Java. Therefore, they can be used to emulate both operating systems supported by COOJA, i.e., Contiki but also TinyOS.

Our implementation interfaces with MSPSim, an instruction-level emulator for the MSP430 MCU that offers also emulation of other on-board components for popular platforms such as TMote Sky. We access MSPSim to *i)* extract the information about power consumption of MCU

and radio; *ii)* access analog-to-digital converter pins to assign realistic battery voltage and harvester power to ADC pins.

The first functionality is necessary because, unlike Contiki, COOJA does not provide a power profiling functionality akin to Energest. Therefore, we essentially reimplemented the same logic of the latter inside COOJA, accessing MSPSim to derive the necessary power profiles during simulation.

The second functionality, instead, is necessary because COOJA does not emulate the battery; reading the emulated battery voltage returns a random value. Therefore, once we have estimated the battery SoC and consequently the variation in voltage, we force these values back to MSPSim, so that applications can read them directly through the (emulated) pins. Accessing directly MSPSim in this fashion allows us to offer accurate emulation of the harvester and its effect on the battery SoC to any operating system supported by COOJA.

IV. SENSEH IN ACTION

To illustrate concretely the use of SENSEH we resort to the application case study that motivated the research described in this paper, constituted by a WSN deployed in a road tunnel to acquire light readings [2]. These are relayed in multi-hop to a gateway, and from there to a Programmable Logic Controller (PLC) that closes the control loop by setting the intensity of the lamps inside the tunnel. In contrast with the state of the art in tunnels, where light intensity is pre-set based on the current date and time, or at best determined by the external conditions, this closed-loop adaptive lighting system maintains optimal light levels by considering the actual conditions inside the tunnel. This increases safety, and enables considerable energy savings. However, the lifetime of the WSN is determined by batteries. Although we showed [3] that this lifetime can be tripled by using data prediction techniques, we want to explore the potential of EH-WSN to approximate energy-neutrality and reduce the maintenance costs to replace depleted batteries.

A. Experimental Setup

Figure 5 shows the placement of WSN nodes inside our 260 m-long, two-way, two-lane tunnel. Overall, 40 nodes are split evenly between the tunnel walls and placed at a height of 1.70 m, compatible with legal regulations. Their data reports are collected by a gateway, installed 2 m from the entrance. Each node is functionally equivalent to a TMote Sky mote, augmented with a sensor board equipped with 4 ISL29004 digital light (illuminance) sensors. Every 30 s each node reports an aggregate of the light readings gathered in the period. Further details are provided in [2].

Here, we want to put SENSEH to play to answer a very simple question: to what extent the WSN can be made energy-neutral by equipping nodes with energy harvesters?

To answer this question we use a slightly different network stack w.r.t. the one in [2], and resort to the popular CTP [21] protocol, `collect` in the Contiki distribution. We rely on ContikiMAC with a sleep time of 125 ms.

Further, we assume that a photovoltaic panel is available as an energy harvester. Indeed, nodes very close to the tunnel entrance have direct exposure to sunlight during the day and can harvest a considerable amount of energy. However, starting few meters deep into the tunnel, nodes have only exposure to the artificial light, which provides much less energy. Nevertheless, with photovoltaic cells optimized for indoor environments and able to achieve high efficiency at low illuminance, even these nodes can harvest some energy. For this study we assume nodes are provided with the Panasonic AM-1816 [22], which fits this requirements. As we conduct our experiments with the MEMORY toolchain, however, this harvester is only simulated in both our simulation and in-field experiments.

To estimate the amount of energy harvested from our panels, we use the same dataset in [3], containing traces of light values collected during winter 2010 from the 40-nodes WSN in Figure 5. This implies that we are using the MEMORY mode of SENSEH, to feed traces to both the simulated and real-world WSN. The light is assumed to be fluorescent, the one our panels are optimized for. We assume a harvesting efficiency of 79% [20], derived from the light conditions in our dataset and the curve in Figure 3, a 100% charging efficiency, and no battery leakage.

Finally, we assume that nodes are equipped with 2 NiMH rechargeable batteries, each providing 2,100 mAh.

B. Results

To evaluate the performance of the WSN vs. the EH-WSN, we first compute the average consumption of the nodes of the network and see how many of them can operate in an energy-neutral manner. Then, we analyze what is the expected improvement in the lifetime of our tunnel WSN.

Consumed vs. Harvested Energy. We analyze the expected amount of energy consumed by the nodes on an average day. This has been computed as an average among a high number of COOJA simulations, trying to reproduce as close as possible the radio propagation conditions of the WSN in the tunnel. From these experiments we derived that a node needs, on average, 72 J/day to be self-sufficient; the harvester should therefore provide an average power of 0.834 mW during the 24 hours.

However, our light dataset shows that it is not possible to achieve a large amount of energy in this environment, as expected. By estimating the energy production achieved per day by an AM-1816 solar panel, we saw that only 8 nodes out of 40 can become self-sufficient with a solar cell array made of less than 10 solar panels, while the others cannot gain enough energy to compensate the one consumed. We therefore provided all the nodes which cannot self-sustain with 10 solar panels each, which already corresponds to a rather large array dimension of 20×30 cm.

Peering at lifetime. To analyze the expected improvement in node lifetime, we compare the initial battery charge with the remaining one when the WSN operates with and without the (simulated) energy harvester. Moreover, we compare the results achieved from COOJA simulations with those yielded by experiments in an indoor testbed. Both experiments have a

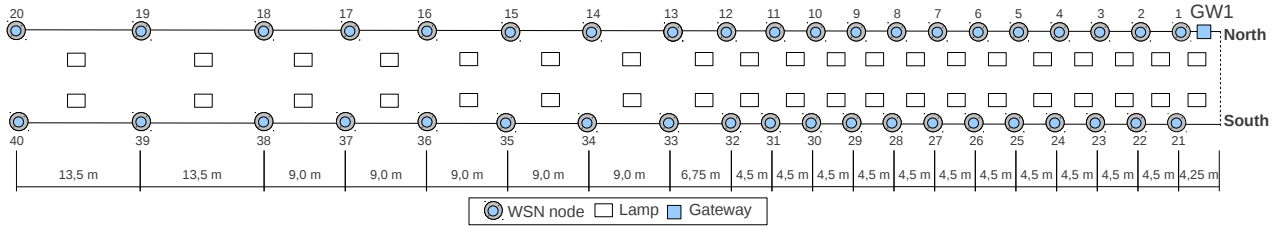
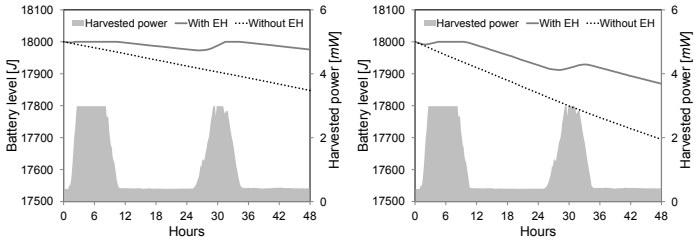
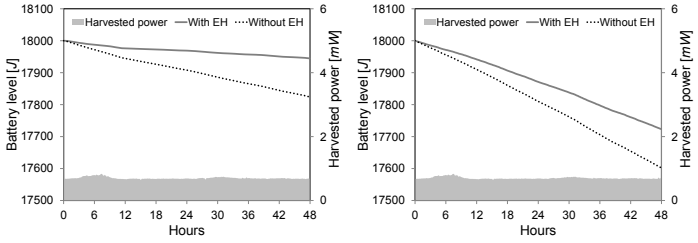


Fig. 5. Physical placement of WSN nodes in the tunnel.



(a) Simulation (b) Testbed

Fig. 6. Battery level and energy harvested for a node close to the entrance.



(a) Simulation (b) Testbed

Fig. 7. Battery level and energy harvested for a node in the transition area.

duration of 48 hours, so that the daily cycle is respected, and consider the exact same period of the light dataset. As a feature of SENSEH, the *same* binary code has been used both inside COOJA *and* on the real nodes; in both cases, light values from traces are injected into the nodes through the serial interface.

Figure 6(a) shows the battery depletion of node 1, the one closest to the tunnel entrance, for the COOJA simulation. We can see that when the node runs without the energy harvester its energy consumption is about 76 J/day, while when energy harvesting is applied the node achieves energy-neutrality: the energy consumed during the night is recovered during the day. This is not the case for node 8, shown in Figure 7(a), which is placed in the transition area between the entrance and the center of the tunnel. The chart shows how nodes in the transition area cannot reach energy-neutrality. As we already discussed, this is because nodes few meters inside the tunnel are never hit by direct sunlight, so their solar panel can only rely on artificial light or reflected sunlight. Still, the harvester reduces energy consumption from 88 J/day to 28 J/day, increasing the expected lifetime by 70%. Nodes placed deep inside the tunnel, with no contribution from sunlight, are exposed only to a very low light intensity coming from artificial illumination. This makes their energy production very scarce, with no appreciable difference with and without our energy harvester, and are therefore not commented further.

An interesting question is whether the same tradeoffs hold when the WSN is deployed in a real setting, in our case an indoor testbed in one of our institutions. Although the latter does not replicate the target tunnel environment, and in particular its topology, it is nonetheless representative of the leap between simulation and the real world, and therefore instructive to investigate, also as a concrete application of our MEMORY toolchain. Indeed, Figure 6(b) shows the battery depletion for the same node 1, but in the experiment run in the testbed. The node is fed *exactly* the same light traces as in the simulated experiment. In the testbed case the node shows a much higher battery discharge, and it cannot operate in an energy-neutral manner. The higher energy consumption of the node is possibly a function of the different topology, but also of radio propagation phenomena like interference and signal reflection, which the simulator grossly approximates. In the testbed experiment the energy depletion of the node without energy harvester is of about 152 J/day, more than twice than in simulation. Nevertheless, when energy harvesting is simulated in the testbed, the battery depletion of node 1 is of 65 J/day, still yielding an expected lifetime improvement of 60%. Similar considerations hold for node 8 in Figure 7(b).

Estimating lifetime through battery voltage decrease. Among the advantages offered by our SENSEH toolchain is the capability of estimating, in simulation, the voltage decrease of the energy source over time, a feature not provided by the original COOJA simulator. This is useful because, when a simulated harvester is not present as in our MEMORY toolchain, applications often use the direct access to the voltage pin as a coarse means to estimate lifetime. This is directly mirrored in the PINS mode, which therefore provides a means to utilize this feature both in simulation and in the final deployed application.

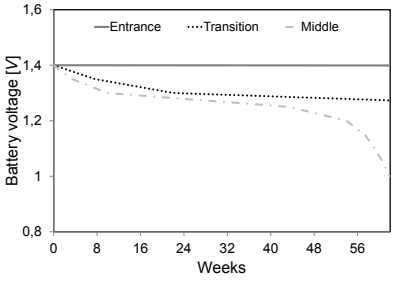


Fig. 8. Voltage depletion of NiMH batteries over time for different positions inside the tunnel.

As an example of what can be inferred from this information, from the 2-day simulations used to achieve the previous results we can derive the trends of voltage decrease of the NiMH batteries and provide an approximation of lifetime. Figure 8 shows the results, for nodes placed in different positions inside the tunnel. The battery of a node close to the tunnel entrance, which can be recharged over the day for the same amount of energy consumed during the night, has an almost constant (and close to full) voltage. The battery voltage for nodes placed in the transition area decreases over time, since the amount of energy consumed is higher than the one harvested. The same holds for nodes placed in the middle of the tunnel, with the very low amount of energy achievable from the environment. These nodes reach 1 V, the battery death point, after 62 weeks of operation, about the same obtained without a harvester.

V. Conclusions

Energy harvesting is becoming increasingly popular as a solution to achieve energy-neutral WSN, and therefore drastically increase their lifetime. However, the tools supporting the design of EH-WSN applications are limited to a few simulators. Developers must implement their application for these simulators using non-WSN programming languages, and then re-implement it for the final WSN deployment. This makes simulation efficient, but jeopardizes its accuracy w.r.t. real-world deployments.

SensEH fills the gap between simulation and deployment by providing a unified software framework that allows one to move back and forth between the power and speed of a simulated approach and the reality and accuracy of in-field experiments. In MEMORY mode, a simulated harvester and associated models are linked with the WSN application, which can therefore be either simulated or tested in-field without having several physical harvesters. In PINS mode, the same can be done for the real code using pin-level communication.

We exemplified the use of SensEH, specifically the MEMORY mode, in a real case study where we investigated to what extent photovoltaic panels can achieve energy-neutrality in a WSN used for adaptive lighting in road tunnels.

Future work will involve the development of models for other types of harvesters, and the study of their advantages by using both MEMORY and PINS modes in the actual tunnel environment.

Acknowledgments

The work described in this paper is partially supported by the European Institute of Innovation and Technology (EIT) ICT Labs, under activity n. 14280 "Intelligent Integrated critical Infrastructures for smarter future Cities" (I^3C).

References

[1] J. Eriksson, F. Österlind, N. Finne, N. Tsiftes, A. Dunkels, T. Voigt, R. Sauter, and P. J. Marrón, "COOJA/MSPSim: Interoperability testing for wireless sensor networks," in *Proc. of the 2^{nd} Int. Conf. on Simulation Tools and Techniques (SIMUTools)*, 2009.

[2] M. Ceriotti, M. Corrà, L. D'Orazio, R. Doriguzzi, D. Facchin, S. Guna, G. P. Jesi, R. L. Cigno, L. Mottola, A. L. Murphy, M. Pescalli, G. P. Picco, D. Pregnolato, and C. Torghele, "Is There Light at the Ends of the Tunnel? Wireless Sensor Networks for Adaptive Lighting in Road Tunnels," in *Proc. of the 10^{th} ACM/IEEE Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2011.

[3] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco, "What Does Model-driven Data Acquisition Really Achieve in Wireless Sensor Networks?" in *Proc. of the 10^{th} IEEE Int. Conf. on Pervasive Computing and Communications (PerCom)*, 2012.

[4] H. Wu, S. Nabar, and R. Poovendran, "An energy framework for the network simulator 3 (ns-3)," in *Proc. of the 4^{th} ICST Int. Conf. on Simulation Tools and Techniques (SIMUTools)*, 2011.

[5] A. Sánchez, S. Climent, S. Blanc, J. V. Capella, and I. Piqueras, "WSN with Energy-harvesting: Modeling and Simulation Based on a Practical Architecture Using Real Radiation Levels," in *Proc. of the 6^{th} ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks*, 2011.

[6] A. Didioui, C. Bernier, D. Morche, and O. Sentieys, "HarvWSNet: A co-simulation framework for energy harvesting wireless sensor networks," in *Proc. of the Int. Conf. on Computing, Networking and Communications (ICNC)*, Jan 2013.

[7] D. Benedetti, C. Petrioli, and D. Spenza, "GreenCastalia: An Energy-harvesting-enabled Framework for the Castalia Simulator," in *Proc. of the 1^{st} Int. Workshop on Energy Neutral Sensing Systems (ENSSys)*, 2013.

[8] G. V. Merrett, N. M. White, N. R. Harris, and B. M. Al-Hashimi, "Energy-aware simulation for wireless sensor networks," in *Proc. of the 6^{th} IEEE Int. Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2009.

[9] J. Jeong and D. Culler, "A practical theory of micro-solar power sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 1, Nov. 2012.

[10] A. Sànchez, S. Blanc, S. Climent, P. Yuste, and R. Ors, "SIVEH: Numerical Computing Simulation of Wireless Energy-Harvesting Sensor Nodes," *MDPI Sensors*, vol. 13, no. 9, pp. 11 750–11 771, 2013.

[11] I. Minakov and R. Passerone, "Pases: An energy-aware design space exploration framework for wireless sensor networks," *J. Syst. Archit.*, vol. 59, no. 8, pp. 626–642, Sep. 2013.

[12] P. Levis, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in *Proc. of the 1^{st} Int. Conf. on Embedded Networked Sensor Systems (SenSys)*, 2003.

[13] "WSNet/Worldsens simulator," http://wsnet.gforge.inria.fr/.

[14] A. Boulis, "Castalia: Revealing pitfalls in designing distributed algorithms in WSN," in *Proc. of the 5^{th} Int. Conf. on Embedded Networked Sensor Systems (SenSys)*, 2007.

[15] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based on-line energy estimation for sensor nodes," in *Proc. of the 4^{th} Workshop on Embedded networked sensors*. ACM, 2007.

[16] R. Rao, S. Vrudhula, and D. Rakhmatov, "Battery modeling for energy aware system design," *IEEE Computer*, vol. 36, no. 12, pp. 77–87, Dec 2003.

[17] D. Doerffel and S. A. Sharkh, "A critical review of using the Peukert equation for determining the remaining capacity of lead-acid and lithium-ion batteries," *Journal of Power Sources*, vol. 155, no. 2, pp. 395 – 400, 2006.

[18] K.-S. Ng, Y.-F. Huang, C.-S. Moo, and Y.-C. Hsieh, "An enhanced coulomb counting method for estimating state-of-charge and state-of-health of lead-acid batteries," in *Proc. of the 31^{st} Int. Telecommunications Energy Conf.*, Oct 2009, pp. 1–5.

[19] B. Buchli, D. Aschwanden, and J. Beutel, "Battery state-of-charge approximation for energy harvesting embedded systems," in *Proc. of the 10^{th} European Conf, on Wireless Sensor Networks (EWSN)*, 2013.

[20] D. Porcarelli, D. Brunelli, M. Magno, and L. Benini, "A multi-harvester architecture with hybrid storage devices and smart capabilities for low power systems," in *Proc. of the IEEE Int. Symp. on Power electronics, electrical drives, automation and motion (SPEEDAM)*, 2012.

[21] O. Gnawali *et al.*, "Collection tree protocol," in *Proc. of the 7^{th} Int. Conf. on Embedded Networked Sensor Systems (SenSys)*, 2009.

[22] Panasonic Eco Solutions Amorton Co., Ltd., "Am-1816," http://panasonic.net/energy/amorton/en/products/spec/AM-1816.html.