AENEAS: An Energy-Aware Simulator of Automatic Weather Stations

Daniel Cesarini, Luca Cassano, Marijan Kuri,
Vedran Bilas, *Senior Member, IEEE*, and Marco Avvenuti, *Member, IEEE*

I. Introduction and Motivations

AUTOMATIC Weather Stations (AWSs) are employed for meteorological sensing in extreme environments, such as desert, Antarctica [1] and glaciers [2]. An AWS is typically composed of a processing unit (CPU, volatile and non-volatile memory) connected to a number of meteorological sensors, e.g., humidity, wind speed, solar radiation, snow height. Wireless communication can be provided through WiFi links, radio bridge, GSM, or satellite links [3]. Since AWSs are usually far from mains power sources, such systems are battery-powered and rely on energy harvesting, typically from solar radiation or wind power [4]. AWS deployments are generally long-term, in the order of years and, meaningfully, after-deployment visits to these installations are generally difficult, expensive, and sometimes even impossible.

For the reasons above, the design of an AWS is a challenging task where guaranteeing long-term and continuous operation in any working condition is of paramount importance [5]. To fulfil this requirement, designers use to over-provision the power supply subsystems, thus increasing the cost and size of the AWS. In addition, sensing and communication frequencies are preferably set to lower values in order to reduce the energy consumption for collecting and communicating data. Unfortunately, such a conservative approach prevents the AWS from taking advantage of energy harvesting surpluses, when available, to offer adaptive sensing and communication.

To allow practitioners to properly configure an AWS, it is necessary to provide them with tools for assessing the impact of hardware choices (e.g., battery and solar panel size) and software parameters (e.g., sensing and communication duty cycles) on the energy behaviour of the system, as early as possible during the design process [6]. Moreover, enabling the AWS to adapt its behaviour based on context information is a desirable solution to maximize the exploitation of the available energy, and thus the amount of sampled and transmitted data, while guaranteeing the survival of the system [7].

A number of techniques and tools for the energy analysis of wireless systems have been proposed in the literature. Nevertheless, although *wireless sensor networks (WSN)* and AWSs have many common features, we believe that the two technologies are subtly but deeply different, not only in size (AWSs are generally bigger than WSN nodes), but also for the following reasons:

- The ratio between energy stored in the battery and energy spent during operation (acquisition, storage and communication) between WSN nodes and AWSs can be very different (up to two orders of magnitude). Indeed, the maximum energy for a typical WSN node battery, AAA 1.5V type, is about 5.4 kJ, while for an AWS battery, 24Ah-12V type, it is 1036 kJ, thus the battery ratio is about 2 orders of magnitude. The transmission energy

Manuscript received March 31, 2014; revised June 10, 2014; accepted August 13, 2014. Date of publication September 12, 2014; date of current version September 26, 2014. This work was supported by the Italian MIUR Project PRIN 2010-11: the Project entitled Response of Morphoclimatic System Dynamics to Global Changes and Related Geomorphological Hazards. The associate editor coordinating the review of this paper and approving it for publication was Dr. Paul Mitchell.

D. Cesarini and M. Avvenuti are with the Department of Information Engineering, University of Pisa, Pisa 56122, Italy (e-mail: danielcesarini@gmail.com; m.avvenuti@iet.unipi.it).

L. Cassano is with the Department of Information Engineering, University of Pisa, Pisa 56122, Italy, and also with the Dipartimento di Elettronica, Informatica e Bioingegneria, Politecnico di Milano, Milan 20133, Italy (e-mail: luca.cassano@polimi.it).

M. Kuri and V. Bilas are with the Department of Electronic Systems and Information Processing, Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb 10000, Croatia (e-mail: marijan.kuri@fer.hr; vedran.bilas@fer.hr).

Color versions of one or more of the figures in this paper are available online.

cost for a typical WSN transceiver, IEEE 802.15.4 radio, is about 200 nJ/bit, while for a typical AWS transceiver, Low-Earth-Orbit satellite modem, it is 1 mJ/bit, thus the transmission energy ratio is about 4 orders of magnitude.
- WSN nodes are typically equipped with a reduced number of sensors, while a single AWS is typically equipped with a high number of sensors.
- Concerning WSNs, in the last 10–15 years there have been a broad number of proposed research platforms, but only few HW (telos, mica) and few SW (TinyOS, Contiki) platforms became used in research, while their industrial adoption was even less, probably for the lack of standards. On the other hand, the world of AWSs is characterized by an heterogeneous platform panorama [3], motivated by a broader range of different manufacturers, each one with it's own legacy hardware and software stack.

Given these points we argue that approaches focused on WSNs hardly apply to AWSs, and thus, that new tools specifically aimed at the analysis of AWSs are needed to successfully design and install such systems.

Analytical models of the energy balance of energy production plants based on solar cells and wind turbines are widely used [8]. Nevertheless, available analytical approaches focus only on single aspects of the system (e.g., assessing the charge of the battery or estimating the amount of energy produced by a solar panel) but fail in providing a holistic analysis of the AWS as a whole complex programmable system. Following a different approach, testbed-based power profiling techniques and software-based power measurement tools can be found in the literature [9], [10], as well as some simulation approaches for the analysis of the energy behaviour of AWSs [11]. They successfully analyse the energy behaviour at the level of hardware configuration or they accurately assess the energy efficiency of communication protocols and scheduling algorithms. However, they miss to consider the high-level software behaviour and its interaction with the complete system.

This paper presents *AENEAS*, an energy-aware AWS simulator that allows designers to perform evaluation of the energy feasibility of system parameters early in the development process. In particular, both the hardware components (batteries, wind turbines, solar panels, maximum power point trackers, sensors, receivers, transmitters, embedded computers, memories) and applications running on the AWS (data acquisition, storage and communication) are modelled following the modeling paradigm first proposed in [12]. Thanks to AENEAS designers can easily study the energy behaviour of the system in order to figure out the best trade-off between cost/size and data (produced and transmitted), while guaranteeing the survival of the system. In order to allow the analysis of adaptive sensing and communication behaviours, AENEAS implements a configurable model of applications running on the considered AWS. AENEAS is not meant to assess only the energy efficiency of communication protocols or scheduling algorithms, tasks for which accurate simulators already exist. The goal of AENEAS is allowing an early evaluation of the energy feasibility of an AWS installation taking into account configurable hardware models and a high-level implementation-independent model of the applications run by the system.

With respect to the state of the art AENEAS represents the first holistic approach to the analysis and simulation of the energy aspects of AWS systems. Indeed, AENEAS is the only simulator that enables designers to analyse the energy impact of both a large number of hardware components and of the application run by the AWS. A preliminary version of the simulator has been presented in [13].

The remainder of this paper is organized as follows: Section II discusses the related works; Section III presents the general structure of an AWS; Section IV shows the overall architecture of the proposed AENEAS tool, and the implemented energy models; Section V discusses how adaptive sensing and communication policies can be modelled; Section VI reports the results of a set of validation experiments; Section VII shows some possible usage of AENEAS on a case study AWS; Section VIII concludes the paper.

II. Related Work

Energy analysis of AWSs and sensor networks can be made by means of analytical approaches, testbeds and simulators.

Analytical approaches are generally used to estimate the amount of energy produced by the energy sources (e.g., solar panels [14]–[16] or wind turbines [17], [18]) and accumulated into batteries [19], [20] based on statistical weather models. Analytical approaches are very effective in characterizing the behaviour of a particular component of the system, while modelling the whole system can become a very difficult task. Moreover, as discussed in [12], analytical approaches generally model the load in a simplistic way and very hardly achieve a model detailing the behaviour caused by changes of data acquisition frequencies and transmission times, that deeply affect the energy demand of the system. We have found only one work presenting an analytical model of a complete energy harvesting system [21] but it targets wireless sensor networks and thus, for the reasons previously presented, it could not be effectively applied to AWSs.

Testbeds are very realistic since they assess the behaviour of the system under design exposing a prototype of it to particular conditions. Testbeds can be expensive and hard to set up, as they use the same (or similar) components and drive the system with the same (or similar) application as the final implementation. In particular we classify testbeds as *outdoor-uncontrolled* [22], [23], and *indoor-controlled* [24]. Outdoor-uncontrolled testbeds are exposed to outdoor weather conditions, similar to those in which the system will operate after deployment. As it has been discussed in [12], experiments on outdoor-uncontrolled testbed can be very long, e.g., if the designer wants to study the behaviour of the system in a given outdoor location for a period of three months, the experiment has to last for three months. On the other hand, indoor-controlled testbeds allow evaluating different setups, having full control over the environment (e.g., using climate-chambers or controllable light or wind sources). Moreover, both types of testbed-based approaches may require long time and high cost to be set up.

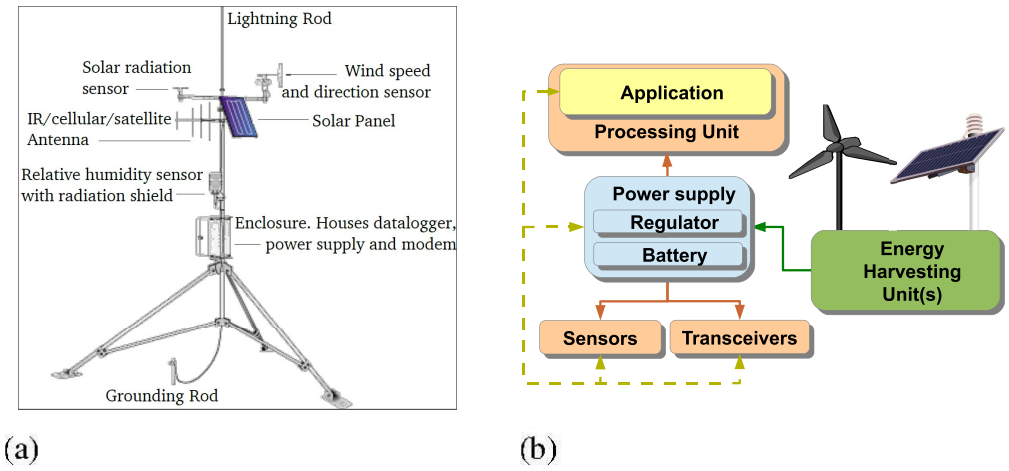

Fig. 1. Picture and structure of an AWS. (a) AWS Drawing - Copyright: Vaisala Inc. Used with permission. (b) General architecture of an AWS.

Simulators enable to study the behaviour of the system early in the design process, thus allowing to assess design issues, such as solar panel or battery size, before getting to the prototype phase. As discussed in [12] most available simulators focus much more on modelling the wireless channel than on energy-related aspects. Few effort has indeed been paid in modelling energy harvesting and even less in analysing energy awareness. Moreover, previously given considerations about the differences between WSNs and AWSs imply that simulators for WSNs cannot be effectively applied for the energy analysis of AWSs. To the best of our knowledge, the only work addressing simulation-based energy analysis that could be applied to AWSs are [11], [25]–[27]. All simulators described in the literature address the problem of a proper estimation of harvested and stored energy. In particular, C. Viehweger *et al.* in [25] presented a simulation-based method focused on the estimation of the energy production level, while F. Cabrera *et al.* in [26] focused on the energy storage capabilities of the system. The two simulators presented in [11], [27] take into account also the loads, but in a simplified way, since they model the loads as an hourly energy consumption value.

III. General Structure of an AWS

The AWS shown in Figure 1(a) is composed of a *Processing Unit*, several *Sensors*, and a *Battery*. *Communication Module(s)* enables radio telemetry and *Energy Harvesting Unit(s)* enables perpetual work. Figure 1(b) depicts the general architecture of an AWS. In the remainder of this section we introduce the basic concepts of AWSs, while for a comprehensive discussion of such systems we refer to [1], [28].

The *Processing units* (General Purpose Processors, FPGAs, ASICs) coordinates the functions of the AWS. It generally uses a persistent memory to store data. A number of analogue and digital Input/Output ports are used to connect sensors and transducers. The processing unit executes tasks, that determine the sampling frequency of sensors and the processing of sampled data. If telemetry is used, tasks also determine the starting time and duration of transmissions. The energy consumption of the processing unit is characterised by different working states, e.g., idle, sampling, transmitting. The energy consumption of both the processing unit and the persistent storage depends on the technology.

Sensors are under the control of the processing unit to sample environmental variables and return electric signals. Typical sensors for AWSs are thermo-hygrometers, thermistors, albedometers, anemometers and snow gauges. Sensors are energy consumers. When inactive, their energy consumption is zero or very low, depending on technology. While sampling, they usually consume a higher energy. Some sensors, such as photo sensors, depending on the conditioning circuitry, can consume a negligible amount of power. Apart from the energy consumed during sampling, some sensors may consume energy also during an activation period, the settling-time.

Batteries provide the necessary energy to the AWS. If harvesting is available, the battery has to be rechargeable, in order to store energy surplus. Rechargeable battery technologies are Lead Acid, Nickel-Metal Hydride (NiMH), Lithium-ion (Li-ion), etc. Rechargeable batteries generally need additional management circuitry in order to regulate charge cycles to maximize available energy and battery lifetime. For low power systems that do not use energy harvesting, batteries can be non-rechargeable, reducing costs and control circuitry.

Transceivers enable the AWS to send the sampled data. This is particularly useful, in some case necessary, for remote installations. Wireless communication can be implemented using WiFi, radio bridge, GSM/GPRS (it is a GSM modem with GPRS data communication) or satelite modems. AWSs usually employ long-range communication, e.g., satellite/GSM links. Such transceivers are eager energy consumers: when idle their consumption is *very low*; when active their consumption depends on the activity, e.g. receiving, connecting, transmitting. Depending on the technology, the energy consumption of transceivers may significantly vary, e.g. a WiFi transceiver consumes $0.1 - 0.5$W while a satellite modem consumes $5 - 15$W. They could represent one of the main energy consumers in AWSs.

On most existing AWSs the *Energy Harvesting* is usually implemented by a photovoltaic panel and/or a wind turbine. Photovoltaic Panels (PV), also referred to as solar panels,

transform solar radiation energy into electric energy. Panels are composed of PV cells. The output power of a PV cell is inversely proportional to the temperature and directly proportional to the solar radiation and the cell area. Wind Turbines transform the kinetic energy of wind into electric energy. The output power of a wind turbine is directly proportional to the area swept by the turbine rotor, air density, and the wind speed. PV panels and wind turbines power is strongly dependant upon the working conditions. Maximum Power Point Tracking (MPPT) is usually installed to optimize their operation.

IV. AENEAS Tool

AENEAS is a *discrete-time-driven* simulator for AWSs, where the energy consumptions of both the hardware components and the applicative tasks run by the station are taken into consideration. In this section we introduce the problem of designing AWSs, how AENEAS addresses it, the structure of AENEAS from a functional point of view, the general energy model and the component models implemented in AENEAS.

We do not strive to substitute analytic models of the various components of an AWS: indeed, in order to incrementally increase the overall accuracy of the proposed simulator, we wish to integrate accurate models into the system as soon as they become available and interested researchers provide evidence for their models.

A. Problem Statement

The goal of AENEAS is supporting AWS designers in assessing the energy feasibility of the overall HW/SW system configuration. As we discussed in the Introduction, AWS designers generally adopt an over-provisioning approach, based on a static analysis: batteries and harvesters are oversized with respect to the actual energy needs of sensing and communication activities. Moreover, both sampling and transmission frequencies are kept constant and set as low as possible in order to minimize the energy needs of the system.

AENEAS performs a dynamic analysis aimed at providing designers a deep insight into the energy flows among the components of the AWS, as determined by application software. In this way, designers can understand the effects on the energy balance of the system due to the characteristics of different hardware components and of application-related features.

AENEAS empowers designers in tailoring both hardware configuration and application features to the needs of the tasks, while guaranteeing the energy survival of the system. The tool computes the energy state of the AWS based on a *conservative estimation approach (CEA)*. Following this approach, the energy consumptions of components is overestimated, the harvested energy and the battery capacity are underestimated, while the power requirements of applications are accurately modelled. As a consequence of the CEA, if AENEAS simulates a positive outcome, i.e., the system can survive, the corresponding real system will very likely survive under the same working conditions. On the other hand, a negative outcome from AENEAS, i.e., the system can not survive, is not indicative of an energy outage of the real system

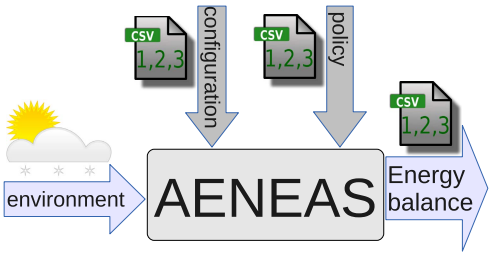

Fig. 2. High-level representation of AENEAS.

```
Time; irradiance; temperature; windspeed
2012/08/01-14:00:00; 125.20; 13.49; 3.4
2012/08/01-15:00:00; 70.77; 13.61; 2.5
2012/08/01-16:00:00; 231.20; 14.15; 4.5
```

Listing 1. An excerpt of an example environment file.

in the same conditions. In other words, AENEAS provides a sufficient but not a necessary condition to the energy feasibility of an AWS's configuration.

Although the difference in the final effect could appear minimal, our over- and under-estimations of consumptions and productions of energy is deeply conceptually and practically different from over-provisioning the AWS. Indeed, in the latter case the designer provides the system with batteries, solar panels etc. larger than what is actually needed because he is not able to (or he cannot) precisely determine the right dimensions. In the former case we followed *CEA* conform choices every time we had to introduce (inevitable) inaccuracies/approximations for the energy models of the components of the AWS. CEA allowed us on the one hand to claim with a high probability that if the simulations give positive results then the system will actually survive in its final installation and on the other hand to tailor the over-provisioning of the system which is actually inevitable when models have inaccuracies/uncertainties. Moreover, the CEA approach guarantees that the more accurate the energy models of the components, the more accurate the analysis performed by AENEAS, and thus, the more restrained the over-provisioning effect caused by the design.

B. General Structure of AENEAS

A high level representation of AENEAS is depicted in Figure 2. The inputs of the tool are *environment*, *configuration* and *policies*, the output is *energy balance*.

Environment contains a description of the environmental conditions under which the system to be studied will work. More in detail, for each simulation time interval, the file contains the values of the variables having impact on the energy state of the system, e.g., irradiance and temperature for the solar panel, wind speed for the wind turbine. An excerpt of an example environment file is shown in Listing 1.

Policy contains information related to the application running on the processing unit of the AWS. In particular, the policy file specifies how the processing unit manages sensors and transceivers. A more detailed discussion of policies is given in the following of this section.

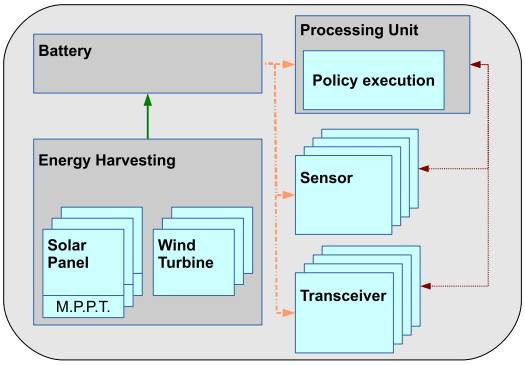

Fig. 3. Internal structure of AENEAS. Green continuous line is energy flow from harvesters to battery, orange dashed lines are energy flow from battery to devices, red dashed lines are control flow between processing unit and sensors/transceivers.

```
ThermhygrHMP45C; 10;
NivometerSR50A; 60;
```

Listing 2. Example of sensors policies.

```
Modem MiChroSat2403; 360;
```

Listing 3. Example of transceiver policy.

Configuration contains information related to the hardware configuration of the AWS. The file describes the number and type of sensors and transceivers, and the type of battery, including component-specific details such as the settling times and the energy a sensor consumes for sampling, the energy a transceiver consumes for transmitting/receiving, the capacity of the battery. More details on the content of the configuration file will be discussed in the next subsection.

The output of AENEAS is the *energy balance* file, containing a detailed report of: the *consumed* energy, i.e., the energy consumed by sensors, transceivers and by the processing unit, the *wasted* energy, i.e., the amount of energy that could have been harvested from the environment but that could not be gathered and stored because the battery was full, and the *stored* energy, i.e., the amount of energy stored in the battery.

The internal structure of AENEAS, depicted in Figure 3, reflects the typical structure of an AWS discussed in the previous section. The AWS is modelled as of a *processing unit*, a number of *sensors*, *transceivers* and *energy harvesters*, and a *battery*.

The model of the processing unit is meant to *execute* a number of policies. Let a *policy* be the set of rules specifying how the processing unit uses a sensor or a transceiver. The application run by the AWS can be viewed as a collection of policies, one for each sensor and transceiver the AWS is composed by. Such application describes the overall behaviour of the system at a high level of abstraction, independently from the actual HW/SW implementation. A application can be easily implemented by, for example, automatically translating the policies into a software program (if the processing unit is a microcontroller) or into a hardware description language specification (if the processing unit is an ASIC or FPGA).

Policies consider only those aspects/parameters that impact on the energy balance of the system. For example, a sensor policy specifies the sampling frequency, a transceiver policy specifies the transmission frequency. Listing 2 gives two example policies for the thermo-hygrometer (sampled once every ten minutes) and for the nivometer (sampled once every sixty minutes). Listing 3 gives an example policy specifying that a transmission must be performed once every six hours.

For each time slot, the execution flow of the simulator is the following:

1) read the input environmental data;
2) based on the given policies, compute the energy consumption of sensors and transceivers;
3) compute the energy consumption of the processing unit;
4) compute the energy produced by the energy harvesters;
5) compute the energy balance.

C. Energy Models

Although real electric and electronic devices have a non-linear characterisation, the level of abstraction and accuracy expected from the simulator, in consequence of the requirements discussed Section IV-A, allowed us to use linear models for the energy behaviour of devices [10]. Sensors, transducers, processing unit, batteries and harvesting units are approximated by linearisation and look-up table based models, and the overall energy model of the AWS is simulated as a composition of these approximations. Such a simplified approach considerably reduced the computational complexity of the simulator.

1) General Model: Let E_{bat}, E_{har}, and E_{cons} be:

- E_{bat}, the energy stored in the battery, calculated as:

$$E_{\text{bat}} = Q \times V \tag{1}$$

  where Q is the charge capacity and V is the voltage.
- E_{har_i}, the energy produced by harvester i, calculated as:

$$E_{\text{har}_i} = I_i \times V_i \times \Delta t = P_i \times \Delta t \tag{2}$$

  where I_i the produced current, V_i the operation voltage, and Δt_i the time interval. Total power P_i depends upon the particular harvester technology.
- E_{cons_j}, the energy consumption of load j, calculated as:

$$E_{\text{cons}_j} = I_j \times V_j \times \Delta t = P_j \times \Delta t \tag{3}$$

  where I_i the consumed current, V_i the operation voltage, and Δt the time interval. Total power P_i depends upon the particular load (transducer, sensor, or processing unit).

For every time interval Δt, the available energy balance E can be calculated by equation (4):

$$E = E_{\text{bat}} + \sum_{i=1}^{i=N_h} E_{\text{har}_i} - \sum_{j=1}^{j=N_c} E_{\text{cons}_j} \tag{4}$$

with N_h harvesters and N_c consumers (loads).

```
Name; Idle; Active; ActivationTime
CR1000; 0.5; 10; 5
```

Listing 4. Example of processing unit configuration.

```
Name; Idle; Active; SensingTime[s]; Bytes
ThermhygrHMP45C; 2; 40; 0.15; 32
NivometerSR50A; 1; 2500; 1.00; 12
```

Listing 5. Example of sensor configurations.

```
Name; MaxEnergy
Vaisala; 500000
```

Listing 6. Example of battery configuration.

2) Component Models: All *energy consumers* have been modelled as finite state machines (FSM), with states characterized by a given energy consumption. When transition times between states exhibit a *relevant energy impact*, also transition times are considered in the model. Consumptions in every state and relevant transition times can be configured through configuration files. The *battery* is modelled as an energy *bucket*, characterized by a configurable finite energy capacity. *Energy harvesters* are modelled as transfer functions, characterized by either analytic functions or look-up-tables filled with producer data-sheet values.

The *Processing Unit model* has two states *Idle* and *Active*. It is Active when the system performs a sensing or communication activity, Idle otherwise. We also considered an *Activation-Time*, needed by the unit to pass from Idle to Active. We gather power (P_j in (3)) and Activation-Time parameters from data-sheets of the processing unit, and provide these values through the corresponding *configuration* file, as shown in Listing 4.

Sensor models have two states: *Idle* and *Active*. The FSM associated with a sensor is active when the simulated sensor is sampling, Idle otherwise. The time spent by the FSM associated with each sensor in the active state encompasses the total time needed to take a sample (settling time and sampling time). Note that, the activation of a sensor causes not only an energy consumption but also data generation. Generated data is stored and, in case of telemetry, it is then sent. Through the *configuration* file we can configure which sensors are available to the AWS, and the characteristics of each sensor (information about sensors can be found in data-sheets). Listing 5 shows an example of sensor configuration file: *Name* is the identifier of the sensor; *Idle* and *Active* are respectively the power need when Idle and when Active; *SensingTime* is the total time needed to take a sample; *Bytes* is the number of bytes stored for each sampling.

The *Battery model* is characterized by an operation Voltage and a maximal charge capacity. According to (1), the battery is modelled as an *energy accumulator*, characterized by a *finite* capacity. Exceeding energy is lost, and its value written in output to *wasted* energy, in order to enable search for optimal design. The implemented battery model does not consider charge efficiency, temperature dependence, and battery ageing. These phenomena can however easily be incorporated in future into the simulator, considering them as additional characteristics of the battery model. Listing 6 shows an example configuration of the battery, where *Name* is the identifier, while *MaxEnergy* represents the maximum energy, Joule [J], that the battery can store, before it stops accepting new charge.

```
Name;Rest;Idle;Tx;Rate[bps];TxOverhead[s]
ModemMiChroSat2403;15;150;4500;2400;20
```

Listing 7. Example of configuration for the transceiver.

```
Model; Impp; Vmpp; Isc; Voc; Alpha; Beta
SP20; 1170; 17.1; 1270; 20.8; 1.2; -73
```

Listing 8. Example of configuration for the solar panel.

The *Transceiver model* has three states: *Rest*, *Idle* and *Transmission*. The FSM associated with the transceiver is in the *transmission* state when the simulated transceiver is sending/receiving data; it is in *idle* state when it is active but not sending/receiving; it is in *rest* state when it is deactivated by the simulated system. The time the FSM remains in the transmission state is determined by the amount of data sampled by the sensors, the transmission rate, and the transmission protocol. We configure which transceivers are available to the AWS, and the characteristics of each transceiver through the *configuration* file (Listing (7)): *Name* is the identifier; *Rest*, *Idle*, *Tx* are respectively the energy consumption when inactive, idle, and in transmission states; *Rate* is expressed in bits per second; and *TxOverhead* represents the technology dependant time-overhead for each transmission (expressed in seconds). The transmission protocol is defined in another configuration file containing the packet structure, through which the actual number of bits that have to be sent can be calculated (considering overhead and packet fragmentation).

The *Energy Harvesting model* receives environmental data and calculates the energy production for every simulation interval. To model harvesters we followed two different approaches, *analytic* and *look-up-table* based: *analytic* models are based on harvester's *equivalent* circuits; *look-up-table* based models use tables as a discrete approximation of the transfer function of the harvester, and are generally simpler to implement and to execute.

The *Photovoltaic (PV) panel model*: analytic approaches typically model a single cell and then replicate the model as many times as needed to obtain the panel. The higher the solar radiation is, the higher will the current/voltage characteristic be. A descending temperature rises the open circuit voltage (V_{oc}) and power of the panel. In particular, we model PV cells using a simplified equivalent circuit with Ohmic resistance losses [29], when no active Maximum Power Point Tracker (MPPT) devices are used on the AWS. The model can be characterized by the current at maximum power point (I_{mpp}),

```
Model;RatedSpeed;CutOffSpeed;Power;Volt
Rutland504; 9.77; 25; 12
```

Listing 9. Example of configuration for the wind turbine.

the voltage at maximum power point (V_{mpp}), the short circuit current (I_{sc}), the current temperature coefficient, and the voltage temperature coefficient. Configuration for the solar panel is provided as in Listing 8, where the parameters, that can be found in data-sheets are: *Model* the identifier; I_{mpp} computed in Standard Test Conditions (STC); V_{mpp} computed in STC; I_{sc} computed in STC; V_{oc} computed in STC; *Alpha* the current temperature coefficient; *Beta* the voltage temperature coefficient. On the other hand, for systems using active MPPT devices we use a look-up-table based model, containing <irradiance/temperature, output power> value set, because analytic models are typically not provided by vendors.

For the *wind turbine model* we used two different approaches, *parametric* and *look-up-table* based. Both approaches are used to approximate the analytic ideal model $P_{turbine} = \frac{1}{2} A \rho v^3 C_p$, characterized by the area swept by the turbine rotor (A), the density of air (ρ), the wind speed (v) and the aerodynamic efficiency of the rotor, called power coefficient (C_p). This analytic model is not used because it does not take into account technology dependant parameters of the wind turbine. The parametric model uses 4 parameters: *rated speed*, i.e., the minimum speed at which the turbine starts to produce its nominal power, *cut-off speed*, i.e., the speed at which the turbine stops to prevent mechanical failures, the constant output *power* of the turbine when the wind speed is between the rated speed and the cut-off-speed, and the operating *voltage* of the wind turbine. Parameters are provided in the configuration file (Listing 9). On the other hand, the look-up-table based model on the other hand is configured providing <wind-speed, output-power> value set, for the range of possible wind speeds.

V. Modelling Adaptive Sensing and Communication Policies

Energy-aware self adaptation of an AWS system is the ability of the system to adapt its working parameters to the environmental conditions [30]. An adaptive AWS system should be able to maximize the usage of the available energy in order to guarantee, on the one hand, its energy survival and, on the other hand, an amount of sampled and transmitted data satisfactory from the data users point of view. Accommodation of sampling rates and transmission frequencies to changing environmental conditions can be carried out using Adaptive Duty Cycling [31], or using Energy-aware Lazy Scheduling Algorithms [32]. While these works are focused on the analysis of the systems from a hardware point of view, the impact on the energy balance of the software running on the processing unit has not yet been extensively studied. Indeed, we believe that the problem should be faced with a comprehensive approach taking into account hardware, software and interactions between them and the environment.

```
Name;BMin;ThHarMin;ThHarMax;Fmin;Fmax;Var
ThermoCS215;400000;280;1080;10;1;Bat
NivometerSR50A;400000;280;1080;30;5;Bat
```

Listing 10. Example of adaptive sensing policy.

With respect to sensing, we define adaptability as the ability to modify the sampling frequency to the environmental conditions. We linearly reduce (increase) the sampling frequency (F_i) of a given sensor i when the amount of energy that can be harvested from the environment (E_h) decreases (increases). Nevertheless, this mechanism is disabled, and the sampling frequency is fixed at a minimum value (F_{min_i}), when the battery level is lower than a given threshold (E_b). In this way, during sunny and windy days the AWS will collect a large amount of data, while during cloudy days or during the night the AWS will save energy. The minimum value of sampling frequency (F_{min_i}) is defined in order to guarantee the minimum amount of sampled data required by the final users (this sampling frequency is used when E_h is lower than a given threshold $E^h_{min_i}$). Similarly, a maximum value of sampling frequency (F_{MAX_i}) is defined in order to avoid oversampling and wasting energy (this sampling frequency is used when E_h is higher than a given threshold $E^h_{MAX_i}$). In this way we force the AWS to exploit all the available energy while guaranteeing the survival of the system when the battery level is high and to recharge the battery while warranting the minimum amount of sampled data when the battery level is low.

With respect to communication, we define adaptability as the ability to dynamically decide whether to start or not a transmission. In particular, the processing unit will try to start a transmission every T_{min_i} (the minimum time between two consecutive transmissions). The transmission will start only if a minimum level of energy is stored in the battery ($E^b_{min_i}$) and a minimum amount of energy could be harvested from the environment according to the values read from the environmental sensors ($E^h_{min_i}$). In this way, we start a transmission only when its energy impact will affect less the energy survival of the system. Further details on modelling of energy-aware adaptive policies are in [33] and [34]. Listing 10 shows an example of adaptive policy.

The policies modelled above are just an example of how a designer could easily simulate his own scenario-dependent policies by means of the very easy-to-define policy configuration file of the simulator.

VI. Validation of AENEAS

Two sets of experiments on two real-world AWSs have been carried out in order to validate the proposed simulator. The first set of experiments, which allowed us to perform a *qualitative* validation, considered an AWS installed on the La Mare Alpine glacier in Italy, used for environmental monitoring. The second set of experiments, which allowed us to perform a *quantitative* validation, considered an urban AWS placed at the University of Zagreb, used primarily as a testbench.

According to the previously discussed *CEA* paradigm we define as *correct* the cases in which the simulator underes-

TABLE I
VALIDATION EXPERIMENT CRITERIA

ΔQ_{sim}	ΔV_{real}	Type of result	Meaning
+	+	True Positive (TP)	Correct
-	-	True Negative (TN)	Correct
+	-	False Positive (FP)	Critical Fail
-	+	False Negative (FN)	Conservative Fail

TABLE II
VALIDATION EXPERIMENTS ON THE LA MARE GLACIER AWS

Type of result	Number	Percentage
True Positive (TP)	611	17.4%
True Negative (TN)	2514	71.8%
False Positive (FP)	88	2.5%
False Negative (FN)	289	8.3%

timates the system lifetime. Moreover, the second validation experiment allowed us to analyse the accuracy of the proposed tool under a quantitative point of view, so we have also measured the error between the real and simulated system lifetimes.

A. The AWS on the La Mare Glacier

The validation experiment described in this section uses the same methodology and set of data of an analogous experiment we carried out on a preliminary version of the simulator. For the paper to be self-contained, we report here the description of the experiment already described in [13].

The AWS is operating since 2007 at 3000 m on La Mare Glacier, Ortles-Cevedale group in the Italian Alps [2]. The station was installed in the framework of a research project concerning the climate change effects on the hydrology and cryosphere of high-altitude catchments [35]. The processing subsystem is a Campbell Scientific CR1000 programmable datalogger based on a Renesas H8S 2322 16-bit CPU, running at 7.3 MHz. The station is powered by a rechargeable 12 V/24 Ah battery. A solar panel with a 20 W power peak output is connected to the battery through a charge regulator. The station samples the snow height sensor once per hour, all the other sensors every 15 min. Collected data are transmitted every three days, at 11.30 a.m.

Environmental data considered for validation purposes were samples of solar radiation and temperature, as they directly affect the efficiency of the solar panel. The available data regarding the energy status of the AWS was the hour-by-hour minimum voltage of the battery.

As the AWS is not supplied with an ammeter, we could not measure current drains, thus no information about the actual charge balance of the power supply was available. However, since batteries have a monotonic voltage/charge characteristic, we could assume that when the voltage of the battery increases the charge balance is positive, and vice-versa. Under this assumption, it is possible to define the validation criteria by comparing the sign of the voltage differential measured in the real system (ΔV_{real}), with the sign of the charge balance estimated by the simulator (ΔQ_{sim}). Table I summarizes the validation criteria used to rank the simulation results.

Considering that the proposed simulator is aimed at supporting designers in the evaluation of the energy impact of AWS's policies, we gave different meanings to the two failing responses of the simulator. In the case of false positive (FP), the simulator estimates an increasing charge while the actual battery voltage is decreasing: we define FP a *critical fail* because it could lead designers to adopt energy unsustainable hardware configurations or policies. In the case of a false negative (FN), the simulator estimates a decreasing charge while the actual battery voltage is increasing: we define FN a *conservative fail* because it could only lead designers to adopt sub-optimal yet energy sustainable hardware configurations and policies.

We carried out the validation experiment by configuring the simulator to reproduce the La Mare glacier AWS and feeding it with data collected by the AWS over a period of 3500 hours, from August to December 2012. The results of the validation, ranked with the criteria discussed above, are reported in Table II. The per-hour comparison between the actual voltage differential and the estimated charge balance has shown that the response of the simulator is *correct* for the 89.2%, a *conservative fail* for the 8.25% and a *critical fail* for only the 2.51% of the observed time period.

These validation experiments have shown that the simulator has a critically wrong behaviour within only 2.51% of the considered time, while in the remaining 97.49% the simulator either correctly estimates the charge balance of the system or underestimates it, thus not affecting the energy feasibility of a given policy. This experiment was used to compare the real battery voltage with the simulated battery energy level. However it was not aimed at estimating the error produced by the simulator, but rather at assessing whether the simulator was able to analyse or not the energy trends in the system.

B. The AWS at the University of Zagreb

The "UniZG AWS" is based on the configurable Libelium Waspmote sensing-node, supplied by a PV energy harvester designed at ZESOI[1] as a successor of the one used in [36]. It acquires environmental data and working parameters of its own power supply. The harvester consists of a solar panel, a MPPT, implemented according to the method presented in [37], a direct power supply, a Li-ion battery pack, a charger, a thermoregulated-heater, used in low battery temperature conditions in which Li-ion battery charging is not allowed, a balance controller and a power path controller. The energy from the output of the MPPT is managed by the balance controller that distributes it between the direct power supply and the charger or the thermoregulated-heater which use energy surpluses. Power path controller switches the load between the direct power supply and the battery. Irradiance, temperatures (panel and system), voltages (panel and battery), current consumption and statuses (charger, transmitter, thermoregulated-heater) are measured, stored and sent to a server using an Xbee 868 RF module. In particular the AWS acquires 100 Bytes of data every 10 seconds, and sends it in blocks every 30 minutes.

[1]Dept. of Electronic Systems and Information Processing, FER-UniZG.

TABLE III

EXPERIMENTS ON THE UNIZG AWS (2013). COMPARISON OF REAL/SIMULATED START/END TIMES. e_s IS SIMULATION ERROR

	Activated	Discharged	Time active [s]
Real	08:27:40, 15/03	05:18:40, 16/03	75064
Simulated	08:27:40, 15/03	05:09:32, 16/03	74512
			e_s = 0.24%
Real	08:32:07, 16/03	04:43:40, 17/03	72693
Simulated	08:32:07, 16/03	01:48:39, 17/03	62192
			e_s = 14.45%
Real	08:24:47, 17/03	13:49:08, 18/03	105861
Simulated	08:24:47, 17/03	09:06:35, 18/03	88908
			e_s = 16.02%
Real	07:45:20, 19/03	01:08:01, 21/03	148961
Simulated	07:45:20, 19/03	18:54:48, 20/03	128568
			e_s = 13.7%
		Average	e_s = 11.1%

The availability of the values of *temperature* and *irradiance*, *charging current* and *battery voltage*, allowed us to perform a quantitative validation of the proposed simulator. In particular, the environmental and electric data used for the validation were generated in four *high-consumption* experiments that ran on the AWS during March 2013. In each of these experiments the following conditions were verified:

- At the beginning of the experiment the battery of the AWS was completely discharged.
- As soon as the solar panel started to produce energy the AWS automatically turned on and started to collect data. The time at which this event happened represented the *starting time of the real experiment* (t_{rs}).
- When the battery level reached zero, the AWS turned off. The time at which this event happened represented the *ending time of the real experiment* (t_{re}).

Moreover, in order to require high currents from the battery and stress the system, two high consumption resistances (1.5 W) are turned on for an hour twice a day. This very high-consumption resistance was added to the system in order to emulate the presence of a high-consumption device, e.g., a modem, which was not available on the system at the moment of the experiments.

We configured AENEAS in order to simulate the AWS in Zagreb in each of the previously described real experiments and we fed it with the data collected by the radiation sensor of the AWS during these experiments. For each simulated experiment we calculated the *starting time of the simulated experiment* (t_{ss}) and the *ending time of the real experiment* (t_{se}). We defined the *error of the simulation* $e_s = 1 - \frac{t_{se}}{t_{re}}$.

Table III summarizes the results from the real and the simulated experiments. First, it can be noticed that, under a quantitative point of view, the proposed simulator quite correctly predicts the energy behaviour of the AWS under analysis. The average error is 11.1%, with an error never exceeding 17%. Moreover, under a qualitative point of view, the simulations show that AENEAS always under-estimates the AWS lifetime. Thus, again, the simulator can be considered effective in aiding the designer in analysing the energy feasibility of a given hardware and software configuration.

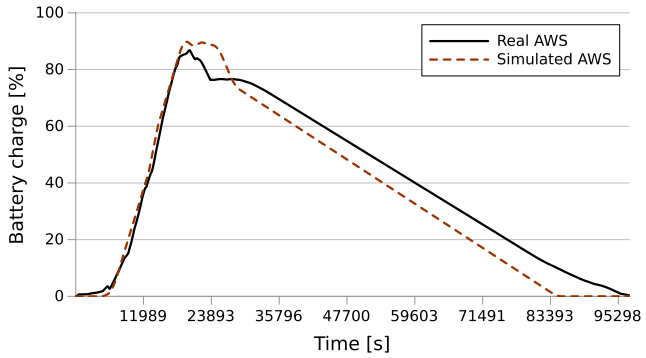

Fig. 4. Comparison between the charge balance of the simulated and the real AWS at the University of Zagreb for the worst of the considered days under the estimated lifetime point of view.

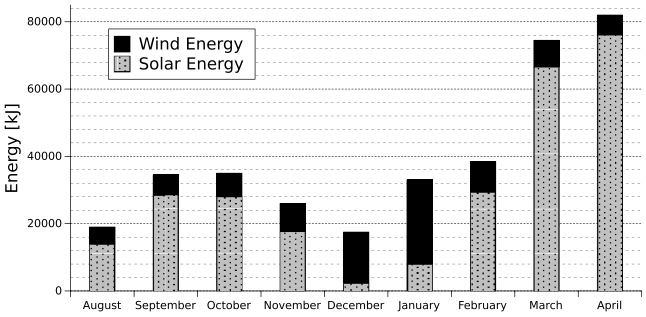

Fig. 5. Comparison between the energy harvested by the solar panel and a wind turbine on the La Mare glacier AWS.

Figure 4 shows the comparison between the energy balances of the simulated and the real AWSs: for the sake of space we only show this comparison for day March, 17. Although the chosen day is the worst one in terms of error in the lifetime estimation, the trend of the energy balance of the real AWS is quite accurately reproduced by the simulator.

VII. EXAMPLE OF USAGE OF THE SIMULATOR

After validating the simulator we ran a set of studies in order to show some example of usage. All these experiments were ran reproducing the previously described AWS on the La Mare glacier and feeding AENEAS with the historical irradiance, temperature and wind speed data collected by the AWS in the period $Aug1, 2012 - Apr5, 2013$.

A. Analysis of the Hardware Configuration

A first set of studies was ran in order to show how the simulator could be used to determine the best hardware configuration for the AWS under design. Thus, a set of simple static sensing and communication policies have been defined: *Sensing policy*: 1 $sample/10$ min, *Communication policy*: 1 $transmission/360$ min.

The first study was aimed at determining whether the installation of a wind turbine on the La Mare glacier AWS would have been beneficial or not (note that the real AWS has an anemometer but not a wind turbine). We first simulated the AWS in the considered period of time with its actual

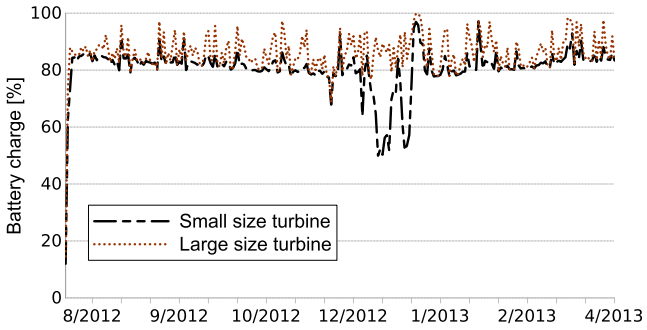

Fig. 6. Comparison between charge level of the battery for the small size and the big size wind turbine cases.

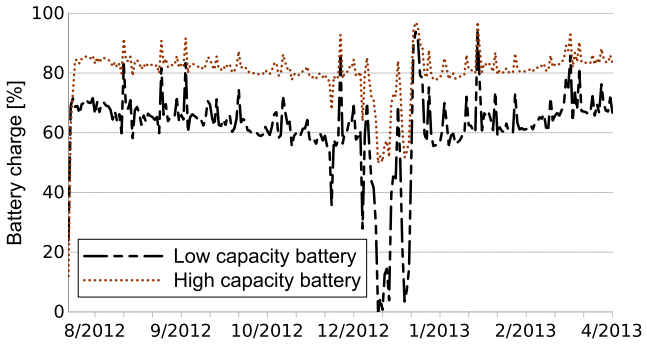

Fig. 7. Comparison between charge levels for the low and the high capacity battery cases.

hardware configuration and then adding the configuration of a Rutland 504 micro wind turbine that produces 25 W when the wind speed reaches the rated speed. The results, in terms of energy harvested from the external environment, are shown in Figure 5. It clearly appears that when the amount of energy harvested by the solar panel is low, a large amount of energy could be harvested by the wind turbine, if available. Thus, such a simulation study suggests the designer that, according to historical data, the AWS could benefit from the installation of a wind turbine if additional budget was available.

The second study was aimed at determining the best hardware configuration in terms of type and size of the wind turbine. We considered two different wind turbines: (i) The Marlec Rutland 504, $287 cost, 3.5 kg, 550 mm in diameter and 60 W as max output power; and (ii) the Ampair 100, $771, 12.5 kg, 928 mm in diameter and 110 W as max output power. We simulated the two configurations for the previously mentioned period of time. As shown by the battery levels curves reported in Figure 6, in both configurations the battery never discharges completely; indeed, both turbines are able to guarantee the energy survival of the AWS. Thus, the choice between the two hardware configurations should be based on other parameters, such as the cost of the components and on the installation difficulties, e.g., turbine weight and size.

The third study, similar to the second one, was aimed at determining the best choice in terms of battery size. We considered two 12 V lead acid batteries: (i) the Power-Sonic PSH-1255, 6 Ah; and (ii) the Yuasa NP12-12, 12 Ah. Again, we simulated the two configurations for the previously

TABLE IV
COMMUNICATION AND SENSING POLICIES USED IN THE ADAPTIVE POLICIES STUDY ON THE LA MARE AWS

	Static		Adaptive	
	Sensing	Communicat.	Sensing	Communicat.
F_{min} [min^{-1}]	1/15	–	1/15	–
F_{MAX} [min^{-1}]	–	–	1/2.5	–
E^h_{min} [mJ]	–	–	100	100
E^h_{MAX} [mJ]	–	–	600	–
E^b_{min} [%]	–	–	80	80
T_{min} [h]	–	3	–	3

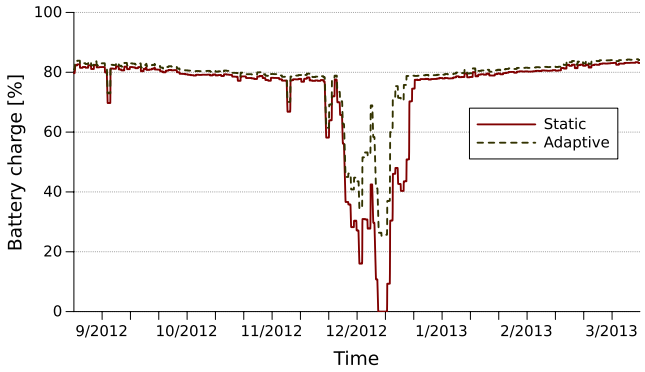

Fig. 8. Comparison of adaptive/static policies. Minimum battery levels.

mentioned period of time. Figure 7 shows the battery levels for the two hardware configurations. It can be noticed that using the Power-Sonic PSH-1255 the battery level sometimes reaches zero during the winter months. Thus, in order to guarantee the energy survival of the AWS, the designer should choose the larger (and most expensive) Yuasa NP12-12 battery.

B. Analysis of the Software Policies

As we previously stated, the proposed simulator is aimed at allowing designers to assess the impact of both hardware- and software-related choices on the energy behaviour of the AWS. In order to show how AENEAS could help designers in analysing the energy impact of software policies we ran a study in which we first simulated the La Mare glacier AWS with a static policy, and then with an adaptive one. The two considered policies are summarized in Table IV.

Figure 8 reports the day-by-day minimum charge levels of the battery, determined by the static and the adaptive policies. It can be observed that in 4 days (from Dec 31 to Jan 3), the static policy leads the AWS to the switch-off, while the adaptive policy achieves the goal of guaranteeing the energy survival of the system. In particular, the benefit is due to the adaptive communication policy that, by postponing transmissions as long as the battery energy level is low, avoids complete discharging the battery.

The simulation also reveals that, in the static case, the AWS is able to collect and transmit up to 2 MBytes data, while in the adaptive case the amount of sampled and transmitted data is 4.2 MBytes. This result shows that the adaptive policies achieve the goal of maximising the amount of data that the AWS is able to collect and transmit.

C. Performance Evaluation of AENEAS

From the computational point of view, we can argue that our simulator is very efficient: we simulated 3500 hours (almost five months) on a computer equipped with an Intel i5-2500k (quad core) CPU working at 3.3 GHz, with 320 KBytes L1 Cache, 1 MByte L2 Cache, 6 MBytes L3 Cache and 8 GBytes RAM, in about 20 seconds.

VIII. Conclusion and Future Work

Automatic weather stations are sensor systems working under harsh conditions. Whilst the traditional mission of AWSs is just collecting and storing data for off-line analysis, it is becoming increasingly desirable to have these systems delivering real-time data via wireless technologies. Wireless communication requirements together with the lack of tools providing an energy analysis of the AWS as a complex programmable system, make even more challenging to design an energy sustainable system. We tried to contribute in bridging this gap by developing AENEAS, a configurable, energy-aware simulator aimed at supporting designers in assessing the energy feasibility of a complex AWS system configuration. Furthermore, we have proposed an approach to describe and test adaptive sensing and communication policies, with the aim to optimise the amount of collected and transmitted data by the AWS. In fact, AENEAS enables simulations during the whole design process, from the hardware configuration phase to the software development phase. In this sense, the tool provides a co-design environment, useful to tailor the hardware components and their usage to satisfy the application requirements, while guaranteeing the energy survival of the system. obtaining an accuracy of about 90%. To validate the simulator, we compared the results of AENEAS simulations with the behaviour of two real AWSs. The first validation experiment allowed us a *qualitative* validation that has shown that in more than the 95% of the cases the simulator either correctly estimates the charge balance of the system or under-estimates it, thus fulfilling *Conservative Estimation Approach*. The second validation experiment allowed us a *quantitative* validation that has shown that the average error is 11.1%, with a maximum error of 17%. Moreover, this experiment showed that AENEAS always under-estimates the AWS lifetime, thus again fulfilling *Conservative Estimation Approach* we defined.

We plan to extend the definitions of sensing and communication policies to model more complex behaviours. Furthermore, we intend to implement an automatic translator from policies models to hardware description and programming languages, to provide a skeleton of the final application based on its model. Moreover, we want to introduce other evaluation metrics for policies, as monetary communication costs. Furthermore, we want to build statistical sound environment models, in order to provide more extensive tests for AWS designs, under different simulated environmental conditions. Finally, we plan to integrate a meteorological and environmental statistically model in the simulator.

Acknowledgment

The authors would like to thank A. Fagioli and S. Mandalà that, during their master theses, helped in developing the proposed simulator. They also would like to thank V. Jelicic for the helpful suggestions during the preparation of the manuscript.

References

[1] C. Reijmer, "Antarctic meteorology: A study with automatic weather stations," Ph.D. dissertation, Univ. Utrecht, Utrecht, The Netherlands, 2001.

[2] S. Abbate, M. Avvenuti, L. Carturan, and D. Cesarini, "Deploying a communicating automatic weather station on an alpine glacier," in *Proc. 4th Int. Conf. Ambient Syst., Netw., Technol. (ANT)*, vol. 19. Jun. 2013, pp. 1190–1195.

[3] D. Cesarini, V. Jelicic, V. Bilas, and M. Avvenuti, "From single point of measurement to distributed sensing in long-term glacier monitoring," *J. Phys., Conf. Ser.*, vol. 450, no. 1, p. 012050, 2013.

[4] S. Priya and D. Inman, *Energy Harvesting Technologies*. New York, NY, USA: Spinger-Verlag, 2009.

[5] J. E. Box, P. S. Anderson, and M. R. van den Broeke, "Lessons to be learned: Extended abstracts," in *Automatic Weather Stations on Glaciers*. Utrecht, The Nederlands: Univ. Utrecht, 2004.

[6] M. Sabharwal, A. Agrawal, and G. Metri, "Enabling green IT through energy-aware software," *IT Professional*, vol. 15, no. 1, pp. 19–27, Jan./Feb. 2013.

[7] S. K. S. Gupta, T. Mukherjee, G. Varsamopoulos, and A. Banerjee, "Research directions in energy-sustainable cyber–physical systems," *Sustain. Comput., Inform. Syst.*, vol. 1, no. 1, pp. 57–74, 2011.

[8] D. Rakhmatov and S. Vrudhula, "Energy management for battery-powered embedded systems," *ACM Trans. Embedded Comput. Syst.*, vol. 2, no. 3, pp. 277–324, 2003.

[9] S. Sundresh, W. Kim, and G. Agha, "Sens: A sensor, environment and network simulator," in *Proc. 37th Annu. Simul. Symp.*, Apr. 2004, pp. 221–228.

[10] S. Abbate, M. Avvenuti, D. Cesarini, and A. Vecchio, "Estimation of energy consumption for TinyOS 2.x-based applications," in *Proc. 3rd Int. Conf. Ambient Syst., Netw., Technol. (ANT)*, vol. 10. Aug. 2012, pp. 1166–1171.

[11] D. Pimentel, P. Musilek, and A. Knight, "Energy harvesting simulation for automatic arctic monitoring stations," in *Proc. IEEE Electr. Power Energy Conf. (EPEC)*, Aug. 2010, pp. 1–6.

[12] G. V. Merrett, N. M. White, N. R. Harris, and B. M. Al-Hashimi, "Energy-aware simulation for wireless sensor networks," in *Proc. 6th Annu. IEEE Commun. Soc. Conf. Sensor, Mesh, Ad Hoc Commun. Netw. (SECON)*, Jun. 2009, pp. 64–71. [Online]. Available: http://dl.acm.org/citation.cfm?id=1687299.1687307

[13] M. Avvenuti, L. Cassano, D. Cesarini, and S. Mandalá, "Simulation of automatic weather stations for the energy estimation of sensing and communication software policies," in *Proc. 5th ExtremeCom*, 2013, to be published.

[14] E. Ortiz-Rivera and F. Peng, "Analytical model for a photovoltaic module using the electrical characteristics provided by the manufacturer data sheet," in *Proc. IEEE 36th Power Electron. Specialists Conf. (PESC)*, Jun. 2005, pp. 2087–2091.

[15] D. Dondi, A. Bertacchini, D. Brunelli, L. Larcher, and L. Benini, "Modeling and optimization of a solar energy harvester system for self-powered wireless sensor networks," *IEEE Trans. Ind. Electron.*, vol. 55, no. 7, pp. 2759–2766, Jul. 2008.

[16] D. Sera, R. Teodorescu, and P. Rodriguez, "PV panel model based on datasheet values," in *Proc. IEEE Int. Symp. Ind. Electron. (ISIE)*, Jun. 2007, pp. 2392–2396.

[17] Y. Lei, A. Mullane, G. Lightbody, and R. Yacamini, "Modeling of the wind turbine with a doubly fed induction generator for grid integration studies," *IEEE Trans. Energy Convers.*, vol. 21, no. 1, pp. 257–264, Mar. 2006.

[18] C. Eisenhut, F. Krug, C. Schram, and B. Klockl, "Wind-turbine model for system simulations near cut-in wind speed," *IEEE Trans. Energy Convers.*, vol. 22, no. 2, pp. 414–420, Jun. 2007.

[19] S. Piller, M. Perrin, and A. Jossen, "Methods for state-of-charge determination and their applications," *J. Power Sour.*, vol. 96, no. 1, pp. 113–120, 2001.

[20] V. Pop, H. J. Bergveld, P. H. L. Notten, and P. P. L. Regtien, "State-of-the-art of battery state-of-charge determination," *Meas. Sci. Technol.*, vol. 16, no. 12, pp. R93–R110, 2005.

[21] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Trans. Embedded Comput. Syst.*, vol. 6, no. 4, Sep. 2007, Art. ID 32.

[22] J. A. R. Azevedo and F. E. S. Santos, "Energy harvesting from wind and water for autonomous wireless sensor nodes," *IET Circuits, Devices, Syst.*, vol. 6, no. 6, pp. 413–420, Nov. 2012.
[23] J. Taneja, J. Jeong, and D. Culler, "Design, modeling, and capacity planning for micro-solar power sensor networks," in *Proc. Int. Conf. Inf. Process. Sensor Netw.*, Apr. 2008, pp. 407–418.
[24] M. Gorlatova *et al.*, "Prototyping energy harvesting active networked tags (EnHANTs)," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 585–589.
[25] C. Viehweger, M. Baldauf, T. Keutel, and O. Kanoun, "Energy profile analysis by simulation for the design of energy harvesting systems," in *Proc. 9th Int. Mult-Conf. Syst., Signals, Devices*, Mar. 2012, pp. 1–3.
[26] F. Cabrera, V. Araña, L. Suárez, G. Gutiérrez, and C. M. Travieso, "Software simulator to model an energy autonomous system," in *Computer Aided Systems Theory—EUROCAST*. New York, NY, USA: Springer-Verlag, 2009, pp. 485–491.
[27] S. Bader, T. Schölzel, and B. Oelmann, "A method for dimensioning micro-scale solar energy harvesting systems based on energy level simulations," in *Proc. IEEE/IFIP 8th Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Dec. 2010, pp. 372–379.
[28] J. Oerlemans and E. J. Klok, "Energy balance of a glacier surface: Analysis of automatic weather station data from the Morteratschgletscher, Switzerland," *Arctic, Antarctic, Alpine Res.*, vol. 34, no. 4, pp. 477–485, 2002.
[29] D. Rekioua and E. Matagne, *Optimization of Photovoltaic Power Systems*. New York, NY, USA: Springer-Verlag, 2012.
[30] C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Lazy scheduling for energy harvesting sensor nodes," in *From Model-Driven Design to Resource Management for Distributed Embedded Systems*, vol. 225. New York, NY, USA: Springer-Verlag, 2006, pp. 125–134.
[31] J. Hsu, S. Zahedi, A. Kansal, M. Srivastava, and V. Raghunathan, "Adaptive duty cycling for energy harvesting systems," in *Proc. Int. Symp. Low Power Electron. Design*, Oct. 2006, pp. 180–185.
[32] C. M. Vigorito, D. Ganesan, and A. G. Barto, "Adaptive control of duty cycling in energy-harvesting wireless sensor networks," in *Proc. 4th Annu. IEEE Commun. Soc. Conf. SECON*, Jun. 2007, pp. 21–30.
[33] D. Cesarini, L. Cassano, A. Fagioli, and M. Avvenuti, "Modeling and simulation of energy-aware adaptive policies for automatic weather stations," in *Proc. Int. Workshop Eng. Simul. Cyber-Phys. Syst.*, 2014, p. 44.
[34] M. Gorlatova, A. Bernstein, and G. Zussman, "Performance evaluation of resource allocation policies for energy harvesting devices," in *Proc. Int. Symp. Model. Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, May 2011, pp. 189–196.
[35] L. Carturan, F. Cazorzi, and G. D. Fontana, "Distributed mass-balance modelling on two neighbouring glaciers in Ortles–Cevedale, Italy, from 2004 to 2009," *J. Glaciol.*, vol. 58, no. 209, pp. 467–486, 2012.
[36] V. Jelicic, T. Razov, D. Oletic, M. Kuri, and V. Bilas, "MasliNET: A wireless sensor network based environmental monitoring system," in *Proc. 34th Int. Conv. MIPRO*, May 2011, pp. 150–155.
[37] O. Lopez-Lapena, M. T. Penella, and M. Gasulla, "A new MPPT method for low-power solar energy harvesting," *IEEE Trans. Ind. Electron.*, vol. 57, no. 9, pp. 3129–3138, Sep. 2010.