applied sciences

MDPI

Article

Towards the Real-World Deployment of a Smart Home EMS: A DP Implementation on the Raspberry Pi

Giuseppe La Tona, Massimiliano Luna *,†, Annalisa Di Piazza and Maria Carmela Di Piazza

Consiglio Nazionale delle Ricerche (CNR), Istituto di Ingegneria del Mare (INM), 90146 Palermo, Italy; giuseppe.latona@cnr.it (G.L.T.); annalisadipiazza79@gmail.com (A.D.P.); mariacarmela.dipiazza@cnr.it (M.C.D.P.)

* Correspondence: massimiliano.luna@cnr.it; Tel.: +39-091-680-9111

† Current address: via Ugo La Malfa, 153, 90146 Palermo, Italy.

Received: 29 March 2019; Accepted: 18 May 2019; Published: 24 May 2019

Abstract: As the adoption of distributed generation and energy storage grows and the attention to energy efficiency rises, Energy Management is assuming a growing importance in smart homes. Energy Management Systems (EMSs) should be easily deployable on smart homes and seamlessly integrate with the Internet of Things (IoT) ecosystem, including generators and storage devices. This paper redesigns a previously presented EMS to reduce its computational complexity, implement it on a Raspberry Pi, and make it compatible with the IoT paradigm. The EMS manages the power flows between smart home loads, renewable generators, electrical storage, and power grid. It communicates with a network of wireless sensors for electrical appliances and with a cloud-based utility data aggregator. The EMS uses Artificial Intelligence and a Dynamic Programming algorithm to fulfill two objectives at the same time: lowering the end user's electricity bill and reducing the uncertainty on the power exchanged between the end user and the grid manager. The latter goal is obtained by an effective compensation of forecasting errors. A test bench emulating four smart homes was used to measure the effectiveness of the EMS and the efficiency of the proposed implementation. The results show an uncertainty of the aggregated exchanged power of only 2.88% and a reduction of the electrical bill for end-users of up to 3.23%. Furthermore, the EMS can complete its most onerous task in less than 9 min. The good performance of the proposed EMS makes it a candidate for fast adoption by the market.

Keywords: Energy Management System; embedded implementation; IoT; Raspberry Pi; optimization algorithm; energy efficiency; demand uncertainty

1. Introduction

The increasing use of distributed renewable energy sources (RES) is one of the primary steps towards the global transition to the smart grid paradigm [1,2]. On the other hand, uncertainties and intermittency of RES result in unpredictable electricity generation profiles having an impact on power system balance and stability [3]. There are many technological and economic issues associated with high rates of renewables penetration in the electrical grid; in particular, daily and seasonal fluctuations in energy production and load demand profiles, high cost of energy storage, and the delicate balance between grid reliability and flexibility are among the most troubling aspects [4,5].

To deal with such issues, suitable Energy Management Systems (EMSs) have been discussed in the technical literature [6–8]. The key functions of EMSs are monitoring and optimization of power flows. These systems, at the end-user level, allow customers to play an active role in the energy

market. On the other hand, they provide support to the grid operator in terms of optimized decisions, for example implementing Demand Side Management schemes [9,10].

Along with the industrial and commercial sectors, EMSs are also advantageous in the context of smart homes, where different kinds of system components (i.e., hardware elements, software algorithms, network connections, and sensors) cooperate with each other to provide various services supporting human lifestyle [11]. An EMS should exhibit distinct features to be engaging for end users. Besides providing a commodity and an advantage, it should seamlessly integrate with the ecosystem of home devices and appliances, it should require an effortless setup and a small amount of wiring, and it should not be bulky. Therefore, besides the research on the functional aspects of EMSs (e.g., how they are defined, what they optimize, and which strategy they use), for their real-world adoption, it is important to investigate how to perform their embedded implementation and to integrate them into a smart home more easily. To this aim, the Internet of Things (IoT) paradigm is currently the favored solution.

However, most works on EMSs disregard these aspects and focus solely on the implementation of support technologies (e.g., networking, smart metering, controllable devices and appliances), or architectural aspects. In fact, networking technologies such as Home Area Networks (HAN) or Personal Area Networks (PAN) are extensively studied to enable the implementation of EMSs [12,13]. Furthermore, some authors focus on the implementation of smart devices that support the interaction with an EMS. For example, the embedded implementation of network-enabled Battery Management System (BMS) is presented in [14]. Instead, Ozadowicz and Grela [15] proposed a smart meter that uses the IoT paradigm to provide a universal device interface. Other examples of IoT system prototypes have been proposed in the technical literature for home automation and energy monitoring inside the smart home/building [16–18].

Several works propose architectural approaches for the smart integration of appliances, devices, and energy management facilities. For example, Khamphanchai et al. [19] proposed a platform for energy management of buildings that is based on distributed agents. Furthermore, architectures based on the IoT paradigm are presented in [20,21]. A cloud-based architecture that permits data collection, control of appliances, and the definition of EMSs as cloud-based services is presented in [22]. Following a similar approach, Al Faruque and Vatanparvar [23] and Yaghmaee Moghaddam and Leon-Garcia [24] proposed architectures based on fog computing [25]. In particular, Al Faruque and Vatanparvar [23] proposed the concept of energy management as a service.

Few works have proposed embedded implementations of EMSs. The authors of [26] implemented an optimization algorithm on embedded devices but did not describe the details of the algorithm. Other works implemented just rule-based EMSs or short-term controllers. For example, some works implemented very simple on/off appliance switching functionalities based either on the hour of the day and the presence of users [27] or on a user-selected power consumption limit [28]. In other works, more complex EMSs are proposed. However, only the rule-based short-term layer of the EMS are implemented on microcontrollers, whereas the medium- and long-term layers run on a centralized outside-of-the-house controller [29] or a dedicated PC [30]. A more sophisticated approach is presented in [31,32], which implement EMSs on microcontrollers to perform load-shifting. These EMSs, however, do not use optimization techniques but a set of preprogrammed rules (fuzzy or deterministic). Although this choice simplifies the implementation, the obtained results are suboptimal in the medium/long term. To the authors' knowledge, the literature misses a detailed description of an embedded EMS implementation that optimizes power flows among renewable generators, loads, and storage systems.

To provide a further step towards the real-world deployment of EMSs, this paper presents an embedded implementation and an extended validation of the EMS proposed in [33]. This EMS is very promising because it provides advantages for both the end user and the utility. In fact, it applies optimization algorithms: (1) to obtain a reduction of the end user's electricity bill; and (2) to reduce the uncertainty, due to forecasting errors, on the daily profile of the power exchanged between the user

and the public utility. The EMS is designed to preserve user habits and to require little communication with the grid manager. These are some of the key requirements for the domestic adoption of an EMS. However, another important requirement is the easy connection to home appliances and to the grid manager. To this aim, it is advantageous to implement the EMS as an IoT device (rather than as a software module of the smart meter) that connects wirelessly to the appliances' sensors and actuators and exposes its services to the grid manager and the user [34].

For the above reasons, the EMS proposed in [33] is particularly suitable for the smart home context. Therefore, the main aim of the present paper is the embedded implementation of such an EMS in order to make it compatible with the IoT paradigm. To this aim, the EMS must be redesigned guaranteeing a feasible implementation and the optimality of the results. Specifically, Di Piazza et al. [33] used Mixed Integer Linear Programming (MILP) and exhibited software dependencies from third-party solvers, which are currently unavailable for embedded platforms; thus, MILP should be replaced with other techniques that do not impose such dependencies.

In general, other authors use heuristic algorithms such as Particle Swarm Optimization to implement EMSs [35,36]. These algorithms are quite simple and do not depend on third-party solvers, thus they satisfy the feasibility requirement. However, these authors only perform simulations and do not make an embedded implementation. In any case, heuristics-based algorithms can get stuck in local optima and do not always return the global optimum; therefore, they do not ensure the optimality of the results.

Based on such considerations, in this paper, the algorithm of Di Piazza et al. [33] is completely re-implemented using Dynamic Programming (DP) instead of MILP. The EMS is implemented and tested on a Raspberry Pi board [37], but can also run on any other Linux-based embedded platform. Its experimental validation was performed connecting it to a network of wireless sensors and actuators based on Wemos boards [38]. The differences between this paper and the work in [33] can be summarized as follows:

- DP is used instead of MILP for the previously explained reasons.
- Instead of considering only one dwelling, the EMS performance was evaluated considering a small grid of four smart homes; this aimed at measuring the increased advantage from the point of view of the utility when several users adopt the EMS.
- The EMS performance was also assessed experimentally by connecting it to a network of wireless sensors according to the IoT paradigm.

The goal of the investigation was twofold: (1) evaluating the theoretical and measured computational efficiency, showing the real-world feasibility of the implemented EMS; and (2) measuring the effectiveness of the EMS for both the utility and the users during a 30-day period.

The paper is organized as follows. The fundamentals of the chosen EMS are highlighted in Section 2, and the case study is described in Section 3. Section 4 is focused on the embedded implementation of the EMS. After describing the chosen embedded platform and the test bench, Section 5 presents and discusses the experimental results. Finally, some conclusions are drawn.

2. Fundamentals of the Chosen EMS

The chosen EMS has been presented in [33]; for the sake of brevity, only the main features and the optimization problems are recalled in the following.

2.1. Main Features

The flowchart of the chosen EMS is shown in Figure 1. As for many EMSs, the processing starts with the *Forecasting* stage, in which the 24-h ahead profiles of load demand and environmental variables tied to renewable generation are forecast on the basis of past data. To achieve good results, the *Forecasting* stage of the chosen EMS is based on the Nonlinear AutoRegressive network with eXogenous inputs (NARX). This Artificial Neural Network (ANN) has been successfully used in

time-series modeling thanks to its simple implementation and its adaptive learning process, even with small-scale data [39–41]. In the considered case, the exogenous input is the environmental temperature.

The subsequent optimization stage considers forecast load demand and plans to satisfy it while pursuing the chosen goals. This stage is split into two tasks, i.e., *Planning* and *Online Replanning*, aiming to provide advantages both to the end user and to the grid manager at the same time.

The *Planning* task optimizes the user's cash flow (CF) over the planned period (24 h) and produces a set of power reference values, one for each system component (i.e., electrical loads, storage systems, and renewable generators). In other EMSs, this set is directly sent to lower level devices that perform the instantaneous control of the hardware. This approach, therefore, neglects forecasting errors. Instead, in the chosen EMS, the output of the *Planning* task is only used to compute the optimal Grid-Exchanged Power Profile (GEPP) for the whole next day. The GEPP is a vector of power values across the time steps of a day. In particular, these values are positive when the grid supplies the smart house, or negative when the power from renewable generators is injected into the grid. The planned GEPP is transmitted over the Internet to the grid manager as an obligation, to which the user commits himself.

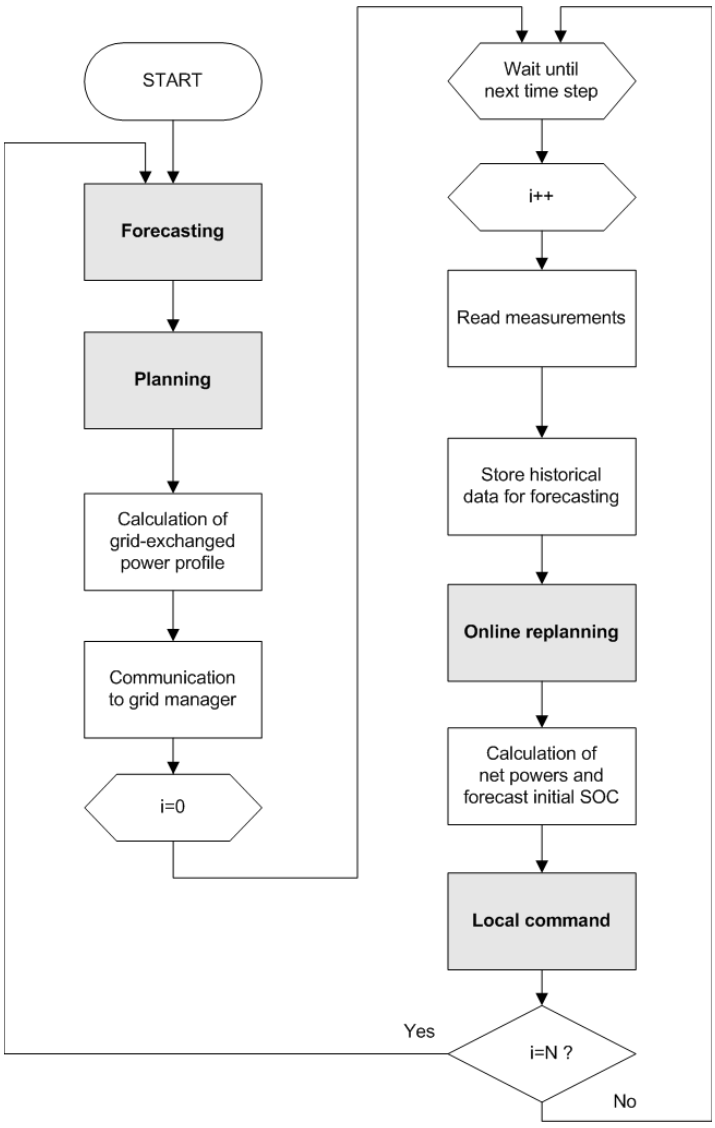

Figure 1. Flowchart of the proposed EMS.

Then, the *Online Replanning* task is repeated during each 24 h cycle. It aims at minimizing the deviation between the actual and the transmitted GEPPs and computes the actual set of power references to obtain this goal. These values are delivered to the different system components by the

Local Command stage via an internal HAN. Besides minimizing the user's cash flow, the proposed EMS reduces the uncertainty on the GEPP thanks to its *Online Replanning* stage. The public utility can exploit this last feature to optimize power generation/distribution and to improve its economic policy planning.

2.2. Optimization Problems

Both *Planning* and *Online Replanning* tasks are executed solving optimization problems. The considered variables are the battery state of charge (SOC) and the power flows between couples of system devices: loads, renewable generators, battery storage systems, and connection to the public grid.

In the optimization problem of the *Planning* stage, the objective function is the end user's cash flow (CF), which is expressed as:

$$f_{obj}(\boldsymbol{x}) = \sum_{t=1}^{N} CF_t \tag{1}$$

$$CF_t = \begin{cases} GEPP_t \Delta t \, C_{buy} & \text{if } GEPP_t \geq 0 \\ GEPP_t \Delta t \, C_{sell} & \text{otherwise} \end{cases} \tag{2}$$

where C_{sell} and C_{buy} are the prices of sold/purchased energy, respectively; Δt is the duration of each time step; t is a discrete variable representing the time step index; and N is the number of time steps in a day.

The objective function in Equation (1) must be minimized satisfying a set of physical and design constraints at each time step, namely:

1. non-negative value for each variable;
2. power balance at generation node;
3. power balance at load node;
4. maximum contractual grid-exchanged power;
5. maximum battery charging/discharging power;
6. minimum and maximum SOC values;
7. continuity of SOC between consecutive days;
8. evolution of SOC between time steps; and
9. cyclicity of SOC profile between consecutive days.

The optimization problem of the *Online Replanning* stage aims at finding the solution that minimizes the maximum deviation between the actual GEPP and the self-committed GEPP ($\check{\gamma}$). Such a deviation is due to forecasting errors. Hence, the objective function to minimize is:

$$f_{obj}(\boldsymbol{x}) = \max_{i \leq t \leq N} |\check{\gamma}_t - GEPP_t| \tag{3}$$

To avoid greedy minimization of instantaneous differences, at each execution, the algorithm decides the variable values for the current time step $t = i$ (using measurements) and also for the following steps $t \leq N$ (using forecast values). However, only the reference power values for the current time step ($t = i$) are sent to the hardware, whereas the other values (for $t > i$) are discarded.

The constraints for Equation (3) are readily adapted from those of the *Planning* stage by:

- changing the time intervals on which the constraints are defined; and
- splitting power balance constraints to account for measured values (at $t = i$) and for forecast values (at $t > i$).

3. The Case Study

The proposed technique was illustrated and validated referring to a small electrical grid that encompasses four networked dwellings. Each smart home was equipped with an EMS of the proposed

type, which generated in real time the references for the power flows among the system components: loads, renewable generators, and battery storage systems. Furthermore, a series of previously devised wireless sensors [38] was connected to the electric appliances to send measured data to the EMS and to forward the power references set by the EMS to the power electronic converters that interface system components. The wireless sensors communicated using Message Queue Telemetry Transport (MQTT), a lightweight messaging protocol for the IoT based on the publish-subscribe paradigm. A pictorial view of each smart home is shown in Figure 2.

The four users (A÷D) had different habits, hence different load profiles. The main parameters of the system under study are shown in Table 1, and the following working assumptions were made:

1. The aggregated load profile is considered as an input; if needed, a lower-level control system can shift or schedule each load, while respecting the aggregated load profile.
2. The maximum contractual power for Users A and C is 3 kW, whereas for Users B and D it is 4 kW.
3. Each user has only one renewable generator; in particular, Users A and B have PV generators, whereas Users C and D have micro Wind Energy Conversion Systems (μWECSs).
4. The renewable generators are always operated in the maximum power point for each environmental condition (wind speed, solar irradiance, and temperature) since they are usually equipped with maximum power point trackers (MPPT).
5. Transferring power from the battery to the grid is not allowed by the utility, according to the technical rule for grid-connection in force in some European countries at the time of writing.
6. The battery must be small and affordable for the end user, thus it is not suitable to sustain hours-long islanding; the considered capacity values for each user are given in Table 1.

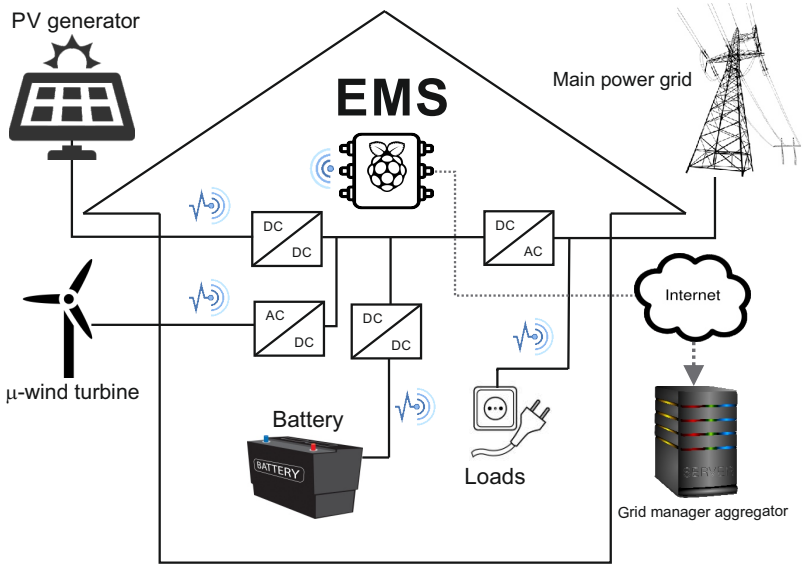

Figure 2. Pictorial view of the smart home under study.

Such working assumptions did not affect the general validity of the proposed EMS. With regard to Points 1 and 4, even if slight deviations occurred (e.g., the lower-level control system does not precisely respect the aggregated load profile or the renewable generator does not work exactly in its maximum power point), they were seen as forecasting errors. Hence, using the battery as an energy buffer, they were effectively corrected by the *Online Replanning* stage of the EMS, as happens for errors due to unexpected weather fluctuations [42]. As for Point 3, in the presence of more than one renewable generator, their power profiles could be aggregated following the same approach used for loads. However, this scenario is highly unusual in the smart home context because it would be very expensive. Finally, with regard to Point 6, the influence of battery size on the EMS performance has been already studied in [42], where a sensitivity analysis has been performed.

To send the self-committed GEPP once a day, each of the four smart homes was connected to the data aggregator of the grid manager through a secure Internet connection [13]. The data aggregator received and aggregated the user-committed GEPPs; the resulting profile could be exploited by the utility to improve its policy planning.

Table 1. System parameters.

User	Parameter	Value	Description
all	C_{buy}	0.0921 ($/kWh)	price of purchased energy
all	C_{sell}	0.0765 ($/kWh)	price of sold energy
all	V_n	200 (V)	battery nominal voltage
all	η	0.95	battery efficiency
all	SOC_{min}	20%	lower bound for battery SOC
all	SOC_{max}	100%	upper bound for battery SOC
all	SOC_{start}	80%	initial value of battery SOC
A, C	P_{xg}	3 (kW)	maximum grid contractual power
B, D	P_{xg}	4 (kW)	maximum grid contractual power
A, B	P_{xp}	3 (kW)	maximum power of the PV generator
C	P_{xw}	3 (kW)	maximum power of the μWECS
D	P_{xw}	1.5 (kW)	maximum power of the μWECS
A, C	C_b	4.6 (kWh)	battery capacity
B	C_b	3.8 (kWh)	battery capacity
D	C_b	2.3 (kWh)	battery capacity
A, C	P_{xb}	1.5 (kW)	maximum battery charge/discharge power
B	P_{xb}	1.24 (kW)	maximum battery charge/discharge power
D	P_{xb}	0.75 (kW)	maximum battery charge/discharge power

4. Embedded Implementation of the Chosen EMS

An effort was made to keep a low computational cost of the algorithm without sacrificing the optimality of the solution. The following subsections describe the details of the implementation. The algorithm was coded in C++ language. Since the chosen platform for the embedded implementation is a Raspberry Pi, as shown in Section 5, the code was compiled using Linaro GNU cross-toolchain armv8l-linux-gnueabihf version 7.2.

4.1. Dynamic Programming Algorithm

The *Planning* and *Online Replanning* problems of the chosen EMS were initially formulated and solved as MILP problems in [33]. However, in the present work, they have been reformulated using Dynamic Programming (DP) to remove software dependencies from third-party solvers and to simplify the overall implementation process. The algorithm for solving these problems has been designed following the approach presented in [43]. In particular, the state of the system corresponds to the battery SOC, which must be discretized using a suitable step size.

The *Planning* stage problem can be formulated as a DP problem if it is framed as the problem of choosing the optimal policy that reaches the best final state from an initial state in a limited time horizon. At each time step and state, the system can choose among a set of actions, i.e., the power reference values for the battery. This choice results in a state transition from a discretized SOC value to another. The *Planning* stage problem, therefore, consists in finding the best path from the initial state to a final state. From Equation (2) it is possible to obtain the cost of each action, i.e., the cost of a state transition. In fact, from a given power reference for the battery, the corresponding power exchanged with the grid (and thus the CF) can be obtained through power balance constraints. Furthermore, the optimization problem constraints define the admissibility of an action.

The problem, as just described, can be represented using a graph such as that in Figure 3. Graph vertices represent system states arranged in vertical layers according to the time step; edges account

for the cost of the action that corresponds to the state transition between the states they link. An edge exists only if the corresponding state transition satisfies the constraints of the problem. Such a graph is generated as described in Algorithm 1. With reference to this graph, the best policy corresponds to the choice of the shortest path from the initial state to one of the vertices associated with the last time step of the planning period. Considering that edges can have negative values (negative CF corresponds to power sold to the utility), the shortest path problem can be solved using Bellman–Ford algorithm [44]. This algorithm computes the shortest paths from a single source vertex (the initial SOC) to all the other vertices in the graph. Then, among the vertices corresponding to the last time step of the planning period, the system chooses the vertex with the shortest path as the target state. In addition, it is worth pointing out that the check for negative cycles in Bellman–Ford algorithm can be skipped since the graph never contains cycles. The pseudo-code of the *Planning* algorithm is shown in Algorithm 2.

The computed optimal path represents the sequence of actions that the system plans, i.e., the profile of power reference values for the battery. From this profile, the *Planning* stage calculates the planned GEPP and finally transmits it to the utility.

Algorithm 1 Pseudo-code of graph building function. It builds a graph such as the one in Figure 3.

function BUILDGRAPH($t_0, genForecast, loadForecast, soc, currGen, currDemand$)
  $G(V, E) \leftarrow \{\}$ ▷ Create empty graph
  $v_0 \leftarrow createVertex(soc, t_0)$ ▷ vertex for the initial SOC
  $addVertex(G, v_0)$
  for $t \leftarrow t_0 + 1$, $N + 1$ **do** ▷ repeats for N time steps starting from the second
    for $next\_soc\_val$ in $discrete\_soc\_values$ **do** ▷ repeats for all the discretized values of SOC
      $v \leftarrow createVertex(next\_soc\_val, t)$
      $addVertex(G, v)$
      for $prev\_v$ in $getTimeStepNodes(t-1)$ **do** ▷ repeats for all the vertices of $t-1$
        $\Delta_{soc} \leftarrow next\_soc\_val - getSoc(prev\_v)$ ▷ calculates difference of SOC
        $pBatt \leftarrow batteryModel(\Delta_{soc})$ ▷ and corresponding power
        if $t == t_0 + 1$ & $replanning$ **then**
          $pGrid \leftarrow pBatt + currDemand - currGen$
        else
          $pGrid \leftarrow pBatt + loadForecast[t-1] - genForecast[t-1]$
        if $checkConstraints(pBatt, pGrid)$ **then** ▷ if all constraints are satisfied
          ▷ calculates weight and adds edge
          if $planning$ **then**
            $w \leftarrow calcCashFlow(pGrid)$
          else
            $w \leftarrow abs(pGrid - plannedGepp[t-1])$
          $addEdge((prev\_v, v), w)$
  return $G(V, E), v_0$

Algorithm 2 Pseudo-code of the *Planning* algorithm.

$t_0 \leftarrow 0$
$(genForecast, loadForecast) \leftarrow getForecasts()$
$init\_soc \leftarrow getInitSoc()$ ▷ retrieves SOC measurement at t_0
$G(V, E), v_0 \leftarrow$BUILDGRAPH($t_0, genForecast, loadForecast, init\_soc, NULL, NULL$)
$(distances, predecessors) \leftarrow bellmanFordAlg(v_0, G)$ ▷ shortest paths from v_0 to all other vertices
$final\_states \leftarrow getTimeStepNodes(N+1)$ ▷ retrieves vertices of final time step
$target \leftarrow$ argmin $distances(x)$ for $x \in final\_states$
$path \leftarrow getPath(predecessors, target)$ ▷ reconstructs path from V_0 to $target$
$pBattProfile \leftarrow calcPBattProfile(path)$ ▷ calculates power corresponding to SOC sequence
$gepp \leftarrow pBattProfile + loadForecast - genForecast$
SENDGEPPTOUTILITY($gepp$) ▷ sends the GEPP to the data aggregator

Algorithm 3 Pseudo-code of the *Online Replanning* algorithm.

$t_0 \leftarrow currenttimestep$
$plannedGepp \leftarrow readPlannedGepp()$ ▷ retrieves committed GEPP
$(genForecast, loadForecast) \leftarrow getForecasts()$
$curr\_soc \leftarrow readCurrSoc()$
$currGeneration, currLoadDemand \leftarrow readMeasures()$
$G(V, E), v_0 \leftarrow$ BUILDGRAPH($t_0, genForecast, loadForecast, curr\_soc, currGeneration, currLoadDemand$)
$(distances, predecessors) \leftarrow findMinMaxPaths(v_0, G)$ ▷ minmax paths from v_0 to all other vertices
$final\_states \leftarrow getTimeStepNodes(N+1)$ ▷ retrieves vertices of final time step
$target \leftarrow \text{argmin}\, distances(x)$ for $x \in final\_states$
$path \leftarrow getPath(predecessors, target)$ ▷ reconstructs path from target to $init\_soc$
$pBattProfile \leftarrow calcPBattProfile(path)$ ▷ calculates power corresponding to SOC sequence
SENDPROFILETOLOCALCOMMAND() ▷ sends battery profile to local command

The same approach can be used to formulate the *Online Replanning* stage as a DP problem. Algorithm 3 reports the pseudo-code of the algorithm. Each time the *Online Replanning* task is executed, a new graph is built. However, the objective function in Equation (3) cannot be expressed as a sum of state transition costs. Thus, the problem cannot be formulated as a shortest path problem. Instead, the cost of each state transition, i.e., the weight of the edges of the graph, corresponds to the deviation of the planned GEPP from the GEPP that results from the state transition. Therefore, the cost of a path is the maximum weight of the edges composing it. The problem corresponds to the single source minmax path search problem. This is the dual of the single source bottleneck path problem [45], and it can be solved executing a modified version of Dijkstra's algorithm [46]. After this execution, the target node and the best path are chosen as in the *Planning* stage.

From the obtained optimal path, at each execution of the *Online Replanning* task, only the first edge is used to compute the power reference values to be sent to hardware devices.

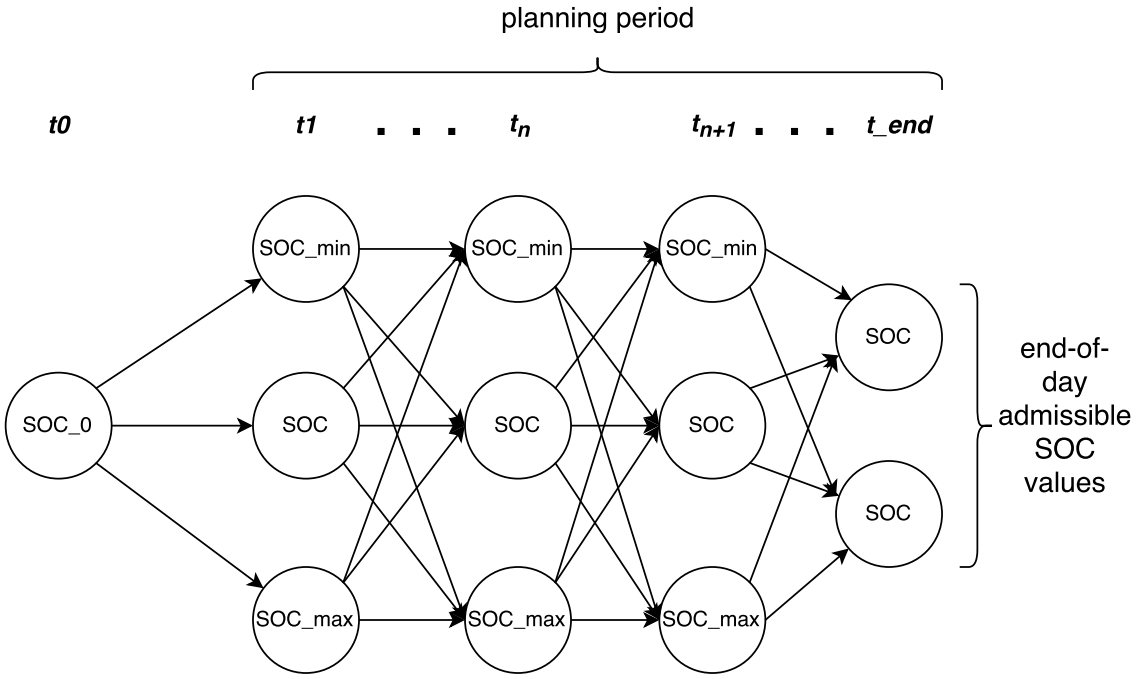

Figure 3. Graph of SOC values representing the actions and the resulting states that the system can choose to schedule its operations.

4.2. Computational Complexity Analysis

The computational complexity of the optimization stage depends on two variables. The first is the number of time steps in a day (N); and the second is the coarseness of the SOC variable discretization,

i.e., the number of possible discrete SOC values (M), which allows computing the SOC discretization step as $\Delta SOC = M^{-1} \cdot 100\%$. The worst-case complexity of the *Planning* algorithm is $O(N^2M^3)$. On the other hand, the worst-case complexity of the *Online Replanning* algorithm is $O(NM^2 + NMlogNM)$. As such, it is lower than the complexity of the *Planning* algorithm.

4.3. NARX ANNs

The NARX ANNs that forecast the 24-h ahead profiles of load demand and environmental variables were implemented using Matlab/Simulink R2014b with Neural Network Toolbox. Then, C/C++ code was generated taking advantage of Matlab Coder. Finally, the obtained code was further optimized and compiled with Linaro GNU cross-toolchain armv8l-linux-gnueabihf version 7.2.

5. Experimental Verification and Results

A dedicated test bench was set up to validate the embedded implementation of the chosen EMS under the hypothesis of controlling four smart homes of the type described in Section 3. Such a test bench is described in detail in the following subsection.

The input data came from publicly available datasets. Outdoor temperature, solar irradiance, and wind speed data came from the U.S. National Solar Radiation Database [47], whereas load demand data came from [48]. Prices for sold and purchased energy were retrieved from [49]. The data considered for the study referred to the city of Amarillo TX, USA. Given each dataset, a first part of the data was used to train and validate the related ANN. The remaining part was considered as the set of instantaneous measurements to be compared with the forecast profile.

The electrical behavior of the appliances of each smart home was emulated by the related wireless sensors, as explained in the following subsection. This approach was followed because the main aim of the work was to assess whether the EMS can run on the embedded platform, perform its computation within the required time frame, and exchange data with the wireless sensors.

5.1. Test Bench Description

Four EMSs of the proposed type were implemented on four Raspberry Pi 3 model B boards running the Linux operating system. This embedded platform was based on a quad-core Broadcom BCM2837 64-bit CPU clocked at 1.2 GHz and has 1 GB of RAM. Furthermore, it provides an onboard WiFi connection, a gigabit Ethernet port, and a Micro SD port to boot the Linux operating system [37]. As an alternative, the proposed EMS could be easily executed on other Linux-based embedded platforms such as the Zedboard [50] by just recompiling the same source code.

The wireless sensors used in the test bench have been previously devised in [38] and are based on Wemos D1 mini pro boards. These boards encompass an ESP8266 system-on-chip by Espressif Systems, which includes a low-cost IEEE 802.11b/g/n Wi-Fi chip with full TCP/IP support and a 32-bit RISC L106 microcontroller. For their use in the field, such devices are appropriately interfaced with physical sensors to acquire the variables of interest (e.g., voltage and current of each appliance, solar irradiance, wind speed, and outdoor temperature). At the same time, they send the appropriate power reference to each power electronic converter, and they read the instantaneous battery SOC. For the present work, instead, such wireless sensors were modified to bypass the physical sensors and to emulate the electrical equipment to which they were connected. In particular, the power profiles of renewable generators and electrical loads were computed on the basis of the considered public datasets. As for the storage system, instead, a simple static battery model was implemented; it is expressed by Equation (4), and it takes into account battery power reference ($P_{batt,ref}$) and charge/discharge efficiency (η) to compute SOC variation. On the other hand, the grid manager's data aggregator was

emulated using a desktop PC connected to the Internet and running a small server application written on purpose.

$$SOC_{t+1} - SOC_t = \begin{cases} \eta \dfrac{\Delta t}{C_b} P_{batt,ref} & \text{if } P_{batt,ref} \geq 0 \\ \dfrac{\Delta t}{\eta C_b} P_{batt,ref} & \text{otherwise} \end{cases} \tag{4}$$

The synopsis of the experimental setup is shown in Figure 4, and its operation is briefly explained in the following. At the beginning of each considered day, each EMS read the instantaneous measurements of load demand, solar irradiance, and wind speed sent from the wireless sensors. Using these data, it performed the forecasting of renewable generation and load profiles thanks to the NARX ANNs. It executed the *Planning* task to compute the self-committed GEPP and sent it to the emulated grid manager's data aggregator using the Internet connection of the Raspberry Pi. Then, it started performing the *Online Replanning* task obtaining power references to control the plant accordingly. In particular, at each time step, it read the current values from the wireless sensors, solved the optimization problem, and sent the power references for the electrical appliances. Given the working assumptions of Section 3, the battery was the only controllable device in each of the four plants. Hence, each EMS sent the battery power reference to the wireless sensor that emulated the battery power flow and updated the SOC. Then, these two quantities were plotted, as shown in Section 5.3. Concurrently, the emulated grid manager's data aggregator performed the following operations:

- It listened to the messages with which each EMS sent its self-committed GEPP 24 h ahead via a secure Internet connection.
- It retrieved the measured power exchanged by the grid and each user via the network of wireless sensors.
- It compared each user-committed GEPP with the corresponding actual GEPP.
- It computed the error metrics on each user's GEPP, which are shown in Section 5.3.
- It computed each user's cash flow starting from the GEPP.
- It compared the aggregated actual GEPP with the aggregated user-committed GEPPs. The two power profiles were then plotted, as shown in Section 5.3.

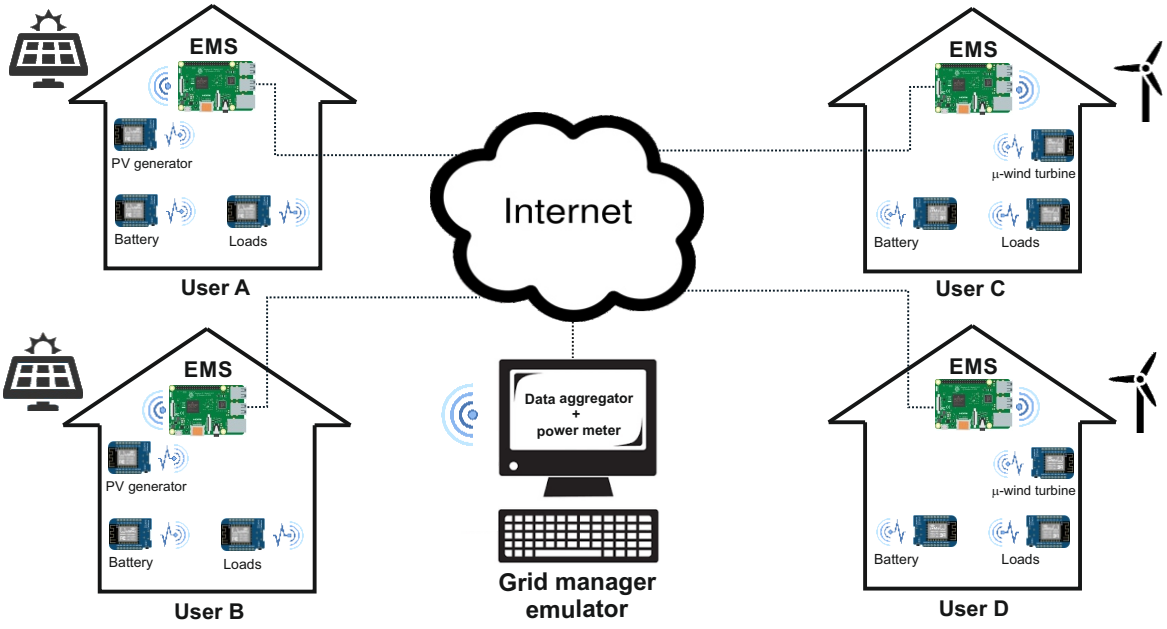

Figure 4. Synopsis of the experimental setup.

5.2. Assessment of EMS Execution Time

The execution time of the *Planning* and *Online Replanning* tasks on the chosen embedded platform was measured in four scenarios to consider different sets of EMS parameters. In particular, two values of the SOC discretization step (i.e., 0.2% and 0.5%) and two values of the time step duration (i.e., 15 and 30 min) were considered.

The experimental results of such an analysis are shown in Tables 2 and 3, and they confirm that *Planning* is the most computationally intensive operation, as demonstrated in Section 4.2. The EMS must start and complete the execution of such a task during the last time step of the day. Therefore, its execution time must be smaller than Δt (i.e., 900 or 1800 s). Table 2 shows that the execution time of the *Planning* task is well below the allowed time frame for the two scenarios with Δt = 30 min. On the other hand, a SOC discretization step of 0.2% cannot be chosen if the user also wants to set the time step to 15 min.

As for the *Online Replanning* task, it has a significantly lower computational complexity. Furthermore, the related execution time decreases during the day. In fact, the number of remaining time steps gets smaller at each execution, and the graph becomes progressively less complex. Therefore, the first time step has the longest execution time of the *Online Replanning* task. As Table 3 shows, the execution of such a task took no more than 2.5 s in all considered scenarios.

Table 2. Execution time of the *Planning* task.

		Δt	
		15 min	**30 min**
ΔSOC	0.2%	1177.3 s	512.9 s
	0.5%	98.6 s	40.3 s

Table 3. Execution time of the *Online Replanning* task (First Step).

		Δt	
		15 min	**30 min**
ΔSOC	0.2%	2249 ms	1703 ms
	0.5%	292 ms	211 ms

On the basis of the results shown in Tables 2 and 3, it is worth estimating whether older Raspberry Pi models are suitable platforms to run the proposed embedded EMS. The constraint to be satisfied is that both *Planning* and *Online Replanning* tasks should be executed during the last time step of the day. As an example, the scenario with ΔSOC = 0.2% and Δt = 30 min was considered. Therefore, the maximum execution time must not exceed 1800 s. The performance degradation index (PDI) due to the use of older Raspberry Pi platforms could be estimated on the basis of benchmark indices such as single-core Million Instructions Per Second (MIPS) and Mega FLoating point Operations Per Second (MFLOPS) values; such indices were retrieved from [51] and are summarized in Table 4. The performance of Raspberry Pi 3 model B was compared with that of the other models by computing the MIPS ratio and the MFLOPS ratio. For a more conservative estimation, the PDI was computed as the maximum of such ratios. Then, the execution time of the proposed embedded EMS on each Raspberry Pi platform was estimated by multiplying the PDI by the execution time measured on the Raspberry Pi 3 model B. As Table 4 shows, the proposed embedded EMS can be executed in all the considered platforms except for the Raspberry Pi 1 model B. Using the same approach, the comparison could be easily extended to the combination of other scenarios and other low-cost embedded platforms for IoT applications [52].

Table 4. Estimated performance comparison of different Raspberry Pi models.

Model	CPU	Clock freq.	MIPS	MFLOPS	PDI	exec. Time
Raspberry Pi 3 B	A53	1.2 GHz	2201	176	-	515 s (meas.)
Raspberry Pi 2 B	A7	900 MHz	1538	120	1.47	757 s (est.)
Raspberry Pi 1 B	ARM11	700 MHz	847	42	4.19	2158 s (est.)
Raspberry Pi Zero	ARM11	1 GHz	1226	68	2.59	1334 s (est.)

5.3. Assessment of the EMS Effectiveness

The above described experimental setup was used to assess the effectiveness of the implemented EMS in the scenario with Δt = 30 min and ΔSOC = 0.5%. In fact, slightly better results could be obtained in the other scenarios. At the functional level, the performed tests showed that the EMS correctly interacted with the wireless sensors and the grid manager's data aggregator. To assess the EMS effectiveness, the uncertainty of the output profile (i.e., the GEPP) must be compared with that of the input profiles (i.e., renewable generation and load demand), and the cash flow variation must be evaluated. The input data for emulating the electric appliances span thirty consecutive days and came from the aforementioned publicly available datasets. As an example, Figure 5a,b shows the comparison between the forecast profiles and the actual input data for the EMSs of Users B and C for the first three days. These plots are also representative of the other two user profiles.

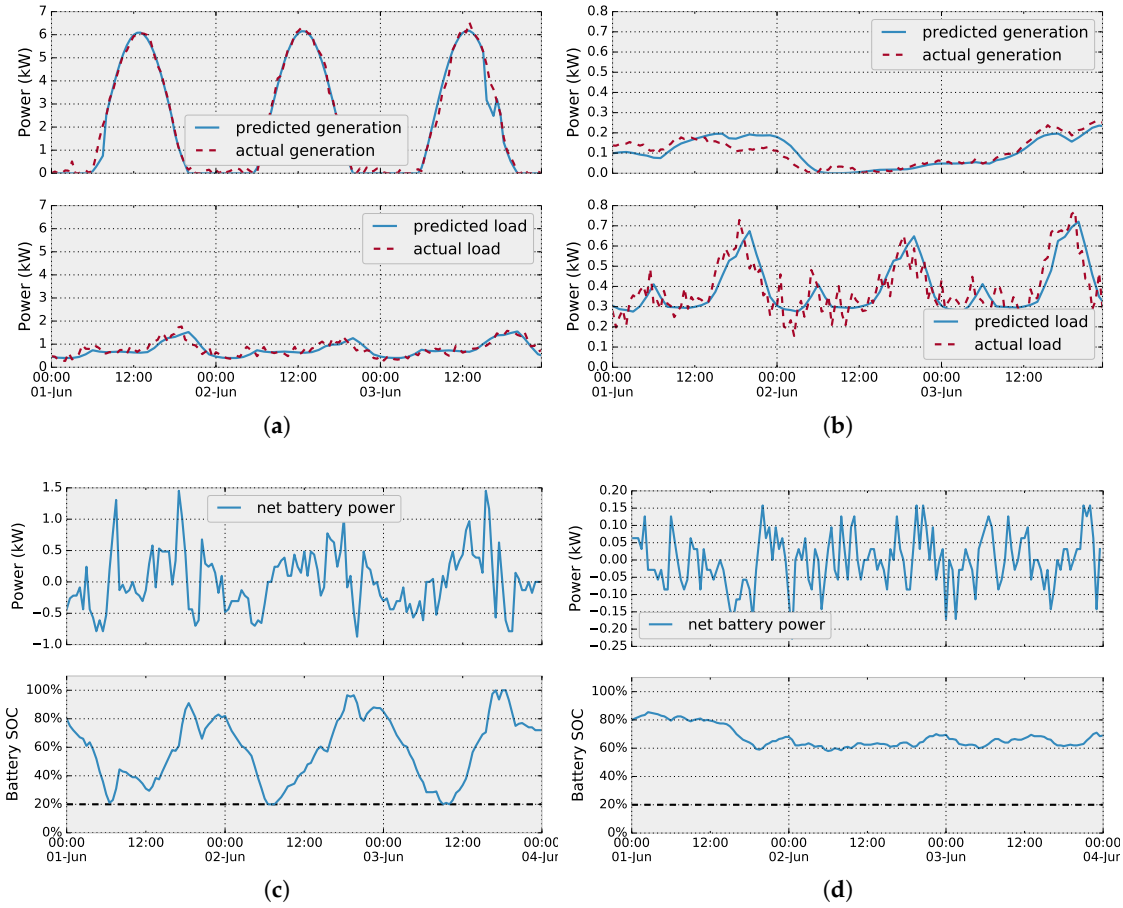

Figure 5. Input and output profiles for two sample users: (**a**) PV generation and load profiles for User B; (**b**) wind turbine generation and load profiles for User C; and battery power and SOC profiles for User B (**c**) and C (**d**).

Two metrics were computed for the forecasting errors, namely the Normalized Root Mean Square Error (NRMSE) and the Normalized Maximum Absolute Error (NMAE). It is worth noting that the error metrics of PV and wind generation were normalized with respect to the nominal power of each renewable generator. On the other hand, error metrics on load demand were normalized with respect to the maximum contractual power. The values of the forecasting error metrics for all users during the 30 days are shown in Table 5; the maximum NRMSE and NMAE values are about 8% and 50%, respectively. The same table also reports the cumulated errors obtained combining generation and load NRMSE values in quadrature.

As an example of the EMS outputs, the plots of battery power and SOC profiles for Users B and C in the first three days of the considered month are shown in Figure 5c,d. These plots are also representative of the profiles of the other two users. As expected, the maximum battery charging/discharging powers are not exceeded.

A plot of the aggregated user-committed GEPP versus the actual cumulative grid power in the same three days is shown in Figure 6. As the figure shows, the two curves overlap each other almost perfectly. Furthermore, Figure 7 shows the difference between planned and measured aggregated GEPP. In particular, the predictability of the power profile is clearly expressed by the error metrics that were computed on the GEPP, as shown in Table 6. The normalization factor for the metrics of each user was the maximum contractual power (i.e., 3 kW for Users A and C, and 4 kW for Users B and D). On the other hand, the normalization factor for the cumulative GEPP was the maximum cumulative contractual power, i.e., 14 kW.

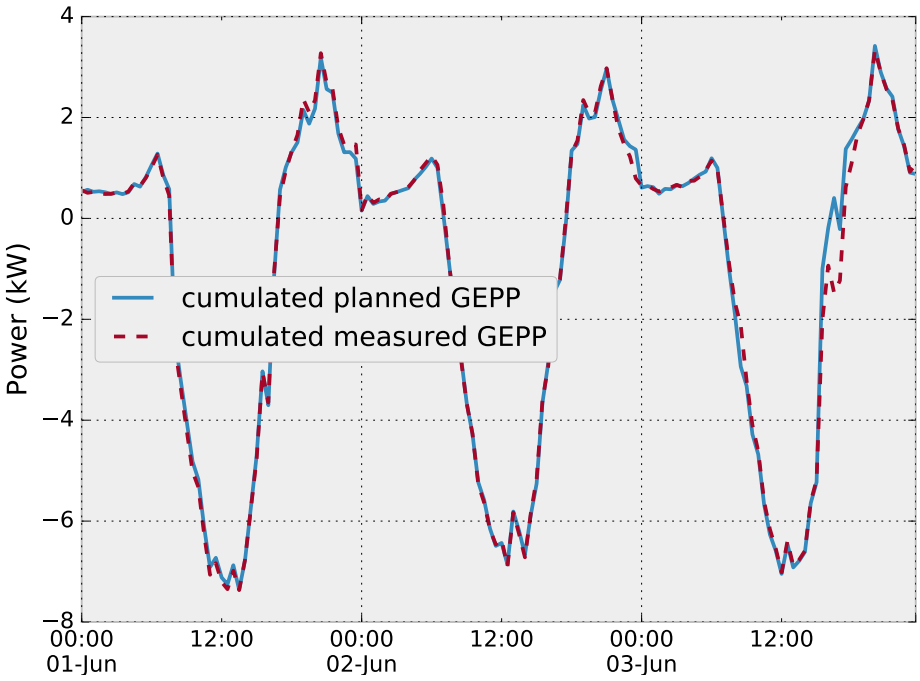

Figure 6. Aggregated user-committed GEPP vs. actual cumulative GEPP.

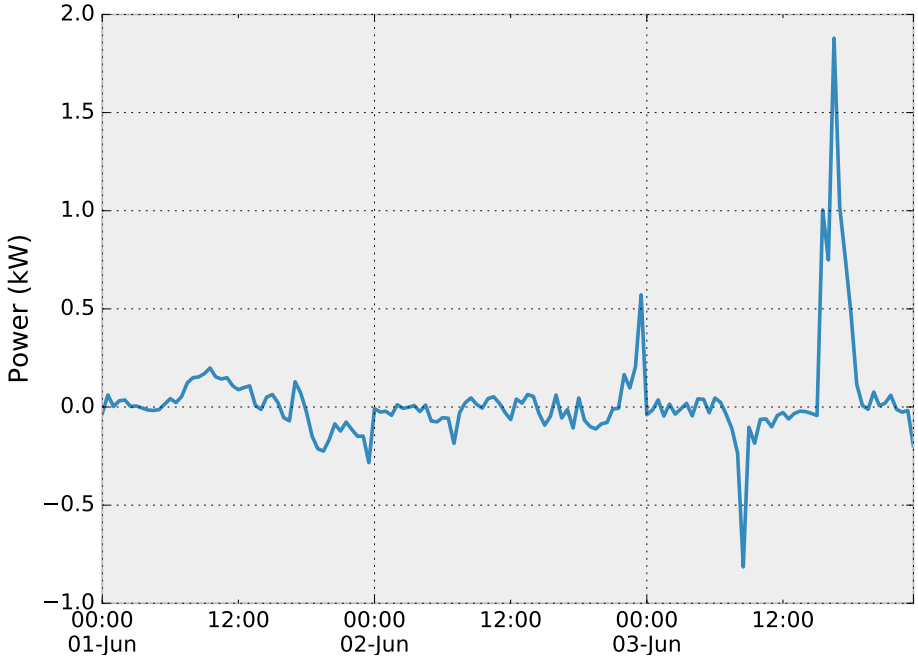

Figure 7. Absolute error on the aggregated GEPP.

It is worth noting that the maximum NRMSE and NMAE values in Table 6 are about 7% and 55%, respectively. Furthermore, the low NRMSE value on the aggregated profile (2.88%) shows that, from the utility perspective, the uncertainty can be even lower than that of the single households. This result is due to the statistical compensation of users' deviations, and it also indicates the effectiveness of the EMS for the utility operator.

Table 5. Error metrics on input profiles.

		User A	User B	User C	User D
generation	nrmse	0.0610	0.0620	0.0843	0.0855
	nmae	0.5014	0.4796	0.4163	0.4253
load	nrmse	0.0351	0.0576	0.0321	0.0648
	nmae	0.1616	0.2473	0.1590	0.2396
cumulated	nrmse	0.0704	0.0846	0.0902	0.1073

Table 6. Error metrics on GEPP.

	User A	User B	User C	User D	Aggregated
nrmse	0.0452	0.0767	0.0135	0.0450	0.0288
nmae	0.4667	0.5521	0.1163	0.3694	0.2202

As an example of the end user's cash flow, a sample day is discussed, i.e., day 4. Despite the forecasting errors and the effort to adhere to the committed GEPP, User B earns $2.46, i.e., 1.46% more than the income without any EMS; User C, instead, pays $0.70, i.e., 3.23% less than the expense without any EMS. The actual profit of both users could even be higher if economic incentives were proposed by the public utility to reward the uncertainty reduction on the GEPP.

6. Conclusions and Future Perspectives

An embedded and computationally efficient implementation of an EMS for smart homes has been proposed. The chosen EMS algorithm has been redesigned using DP instead of MILP to simplify the implementation process. As demonstrated by a computational complexity analysis, the obtained EMS

is suitable for the embedded implementation. Furthermore, the optimality of the solutions has not been compromised in the algorithm redesign.

A suitable test bench was set up to assess the EMS performance in a chosen case study (i.e., four networked dwellings including renewable generators) that was emulated. The experimental results show that the EMS can perform each of its tasks on the embedded platform within the allowed time frames; they also show that the embedded EMS correctly exchanges data with the wireless sensors and the grid manager's data aggregator. Furthermore, an extended validation of the EMS effectiveness was performed in terms of cash flow increase and uncertainty reduction.

In particular, the experiments showed that the EMS performs its computations in a few minutes for the *Planning* task and in less than 2.5 s for the *Online Replanning* task. Furthermore, the EMS performance is very good since the uncertainty on the daily profile of the power exchanged by the users with the utility is significantly mitigated (NMRSE = 2.88% and NMAE = 22%). This result is a first step towards the quick and affordable adoption of EMSs in smart homes.

As for the future development of this work, the short execution times obtained for the proposed embedded EMS suggest that there is sufficient margin to take into account other important aspects such as battery state of health in the EMS formulation. Furthermore, future activities will contemplate testing the embedded EMS on the field, i.e., within real-world smart homes.

Author Contributions: Conceptualization and methodology, G.L.T., M.L., M.C.D.P. and A.D.P.; software development, G.L.T.; validation, G.L.T., M.L. and M.C.D.P.; and writing, review and editing, G.L.T., M.L., M.C.D.P. and A.D.P.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yu, X.; Cecati, C.; Dillon, T.; Simões, M. The New Frontier of Smart Grids. *IEEE Ind. Electron. Mag.* **2011**, *5*, 49–63. [CrossRef]
2. Liserre, M.; Sauter, T.; Hung, J. Future Energy Systems: Integrating Renewable Energy Sources into the Smart Power Grid Through Industrial Electronics. *IEEE Ind. Electron. Mag.* **2010**, *4*, 18–37. [CrossRef]
3. Lenzi, V.; Ulbig, A.; Andersson, G. Impacts of forecast accuracy on grid integration of renewable energy sources. In Proceedings of the 2013 IEEE Powertech Grenoble Conference, Grenoble, France, 16–20 June 2013; pp. 1–6. [CrossRef]
4. Denholm, P.; Margolis, R.M. Evaluating the limits of solar photovoltaics (PV) in traditional electric power systems. *Energy Policy* **2007**, *35*, 2852–2861. [CrossRef]
5. Denholm, P.; Margolis, R.M. Evaluating the limits of solar photovoltaics (PV) in electric power systems utilizing energy storage and other enabling technologies. *Energy Policy* **2007**, *35*, 4424–4433. [CrossRef]
6. Soares, A.; Gomes, A.; Antunes, C.H.; Oliveira, C. A Customized Evolutionary Algorithm for Multiobjective Management of Residential Energy Resources. *IEEE Trans. Ind. Inform.* **2017**, *13*, 492–501. [CrossRef]
7. Paterakis, N.G.; Erdinç, O.; Bakirtzis, A.G.; Catalão, J.P.S. Optimal household appliances scheduling under day-ahead pricing and load-shaping demand response strategies. *IEEE Trans. Ind. Inform.* **2015**, *11*, 1509–1519. [CrossRef]
8. Nambi, S.N.A.U.; Pournaras, E.; Venkatesha Prasad, R. Temporal Self-Regulation of Energy Demand. *IEEE Trans. Ind. Inform.* **2016**, *12*, 1196–1205. [CrossRef]
9. Palensky, P.; Dietrich, D. Demand side management: Demand response, intelligent energy systems, and smart loads. *IEEE Trans. Ind. Inform.* **2011**, *7*, 381–388. [CrossRef]
10. Kumar Nunna, H.S.V.S.; Doolla, S. Responsive End-User based Demand Side Management in Multi-Microgrid Environment. *IEEE Trans. Ind. Inform.* **2014**, *3203*. [CrossRef]
11. Amer, M.; Naaman, A.; M'Sirdi, N.K.; El-Zonkoly, A.M. Smart home energy management systems survey. In Proceedings of the 2014 International Conference on Renewable Energies for Developing Countries, REDEC 2014, Beirut, Lebanon, 26–27 November 2014; pp. 167–173. [CrossRef]

12. Bian, D.; Kuzlu, M. Assessment of communication technologies for a home energy management system. In Proceedings of the Innovative Smart Grid Technologies Conference (ISGT) 2014, Washington, DC, USA, 19–22 February 2014; pp. 1–5. [CrossRef]
13. Kailas, A.; Cecchi, V.; Mukherjee, A. A Survey of Communications and Networking Technologies for Energy Management in Buildings and Home Automation. *J. Comput. Netw. Commun.* **2012**, *2012*, 1–12. [CrossRef]
14. Wei, T.; Kim, T.; Park, S.; Zhu, Q.; Tan, S.X.D.; Chang, N.; Ula, S.; Maasoumy, M. Battery Management and Application for Energy-Efficient Buildings. In Proceedings of the 51st Annual Design Automation Conference—DAC '14, San Francisco, CA, USA, 1–5 June 2014; pp. 1–6. [CrossRef]
15. Ozadowicz, A.; Grela, J. Control application for Internet of Things energy meter—A key part of integrated building energy management system. In Proceedings of the 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), Luxembourg, 8–11 September 2015; pp. 1–4. [CrossRef]
16. Al-Ali, A.; Zualkernan, I.A.; Rashid, M.; Gupta, R.; Alikarar, M. A smart home energy management system using IoT and big data analytics approach. *IEEE Trans. Consum. Electron.* **2017**, *63*, 426–434. [CrossRef]
17. Turkmen, A.; Peng, A.S.; Miller, M.; Dassow, B.; Bauer, D.; Mills, L.; Zimmer, M.; Batzler, R. Development of a Home Energy Monitoring System: A Capstone Project Experience. In Proceedings of the 2019 IEEE Power and Energy Conference at Illinois (PECI), Champaign, IL, USA, 28 February–1 March 2019; pp. 1–6. [CrossRef]
18. Pavithra, D.; Balakrishnan, R. IoT based monitoring and control system for home automation. In Proceedings of the 2015 Global Conference on Communication Technologies (GCCT), Thuckalay, India, 23–24 April 2015; pp. 169–173. [CrossRef]
19. Khamphanchai, W.; Saha, A.; Rathinavel, K.; Kuzlu, M.; Pipattanasomporn, M.; Rahman, S.; Akyol, B.; Haack, J. Conceptual architecture of building energy management open source software (BEMOSS). In Proceedings of the IEEE PES Innovative Smart Grid Technologies, Europe, Istanbul, Turkey, 12–15 October 2014; pp. 1–6. [CrossRef]
20. Han, D.M.; Lim, J.H. Smart home energy management system using IEEE 802.15.4 and ZigBee. *IEEE Trans. Consum. Electron.* **2010**, *56*, 1403–1410. [CrossRef]
21. Gibescu, M.; Wattjes, F.D.; Kling, W.L.; Gonzalez, R.M.; Vermeiden, W.; Slootweg, J.G. Applied Internet of Things Architecture to Unlock the Value of Smart Microgrids. *IEEE Internet Things J.* **2018**, *5*, 5326–5336. [CrossRef]
22. Li, W.T.; Yuen, C.; Hassan, N.U.; Tushar, W.; Wen, C.K.; Wood, K.L.; Hu, K.; Liu, X. Demand Response Management for Residential Smart Grid: From Theory to Practice. *IEEE Access* **2015**, *3*, 2431–2440. [CrossRef]
23. Al Faruque, M.A.; Vatanparvar, K. Energy Management-as-a-Service Over Fog Computing Platform. *IEEE Internet Things J.* **2016**, *3*, 161–169. [CrossRef]
24. Yaghmaee Moghaddam, M.H.; Leon-Garcia, A. A Fog-Based Internet of Energy Architecture for Transactive Energy Management Systems. *IEEE Internet Things J.* **2018**, *5*, 1055–1069. [CrossRef]
25. Aazam, M.; Huh, E.N. Fog Computing and Smart Gateway Based Communication for Cloud of Things. In Proceedings of the 2014 International Conference on Future Internet of Things and Cloud, Barcelona, Spain, 27–29 August 2014; pp. 464–470. [CrossRef]
26. Belcredi, G.; Modernell, P.; Sosa, N.; Steinfeld, L.; Silveira, F. An implementation of a home energy management platform for Smart Grid. In Proceedings of the 22015 IEEE PES Innovative Smart Grid Technologies Latin America (ISGT LATAM), Montevideo, Uruguay, 5–7 October 2015; pp. 270–274. [CrossRef]
27. Park, S.; Choi, M.I.; Kang, B.; Park, S. Design and Implementation of Smart Energy Management System for Reducing Power Consumption Using ZigBee Wireless Communication Module. *Procedia Comput. Sci.* **2013**, *19*, 662–668. [CrossRef]
28. Bhuvaneswari, S.; Satish, B.; Mahalaksmi, R. Wireless Home Energy Consumption Control based on prioritised load switching. In Proceedings of the 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), Chennai, India, 6–8 May 2015; pp. 548–553. [CrossRef]
29. Kanchev, H.; Lu, D.; Colas, F.; Lazarov, V.; Francois, B. Energy Management and Operational Planning of a Microgrid With a PV-Based Active Generator for Smart Grid Applications. *IEEE Trans. Ind. Electron.* **2011**, *58*, 4583–4592. [CrossRef]

30. Khoury, J.; Mbayed, R.; Salloum, G.; Monmasson, E. Design and implementation of a real time demand side management under intermittent primary energy source conditions with a PV-battery backup system. *Energy Build.* **2016**, *133*, 122–130. [CrossRef]
31. Ogwumike, C.; Short, M.; Abugchem, F. An embedded prototype of a residential smart appliance scheduling system. In Proceedings of the 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, Germany, 6–9 September 2016; pp. 1–5. [CrossRef]
32. Ciabattoni, L.; Grisostomi, M.; Ippoliti, G.; Proietti Pagnotta, D.; Foresi, G.; Longhi, S. Residential energy monitoring and management based on fuzzy logic. In Proceedings of the 2015 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 9–12 January 2015; pp. 536–539. [CrossRef]
33. Di Piazza, M.; La Tona, G.; Luna, M.; Di Piazza, A. A two-stage Energy Management System for smart buildings reducing the impact of demand uncertainty. *Energy Build.* **2017**, *139*, 1–9. [CrossRef]
34. Pan, Z.; Guo, Q.; Sun, H. Impacts of optimization interval on home energy scheduling for thermostatically controlled appliances. *CSEE J. Power Energy Syst.* **2015**, *1*, 90–100. [CrossRef]
35. Pedrasa, M.A.A.; Spooner, T.D.; MacGill, I.F. Coordinated scheduling of residential distributed energy resources to optimize smart home energy services. *IEEE Trans. Smart Grid* **2010**, *1*, 134–143. [CrossRef]
36. Zhuang Zhao.; Won Cheol Lee.; Yoan Shin.; Kyung-Bin Song. An Optimal Power Scheduling Method for Demand Response in Home Energy Management System. *IEEE Trans. Smart Grid* **2013**, *4*, 1391–1400. [CrossRef]
37. Raspberry Pi Foundation. Raspberry Pi. 2018. Available online: https://www.raspberrypi.org/ (accessed on 20 March 2019).
38. Luna, M.; La Tona, G.; Di Piazza, M.C.; Pucci, M.; Accetta, A.; Taibi, D.; Vetro, C.; La Grassa, R. A Prototype of Wireless Sensor for Data Acquisition in Energy Management Systems. In Proceedings of the 2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe), Palermo, Italy, 12–15 June 2018. [CrossRef]
39. Di Piazza, A.; Di Piazza, M.; Vitale, G. Estimation and Forecast of Wind Power Generation by FTDNN and NARX-net based models for Energy Management Purpose in Smart Grids. In Proceedings of the International Conference on Renewable Energies and Power Quality (ICREPQ'14), Cordoba, Spain, 8–10 April 2014.
40. Di Piazza, A.; Di Piazza, M.; Vitale, G. Solar Radiation Forecasting Based on Artificial Neural Networks Optimized by Genetic Algorithm for Energy Management in Smart Grids. In Proceedings of the European PV Solar Energy Conference and Exhibition (EU PVSEC 2014), Amsterdam, The Netherlands, 22–26 September 2014; pp. 2574–2579. [CrossRef]
41. Siegelmann, H.; Horne, B.; Giles, C. Computational capabilities of recurrent NARX neural networks. *IEEE Trans. Syst. Man Cybern. Part B* **1997**, *27*, 208–215. [CrossRef]
42. Di Piazza, M.; Luna, M.; Di Piazza, A.; La Tona, G. Energy Management Systems for Effective Gap Reduction Between Actual and Predicted Power in Smart Homes and Buildings. In Proceedings of the 42nd Annual Conference of the IEEE Industrial Electronics Society (IEEE IECON2016), Florence, Italy, 23–26 October 2016.
43. Riffonneau, Y.; Bacha, S.; Barruel, F.; Ploix, S. Optimal Power Flow Management for Grid Connected PV Systems With Batteries. *IEEE Trans. Sustain. Energy* **2011**, *2*, 309–320. [CrossRef]
44. Bang-Jensen, J.; Gutin, G.Z. *Digraphs: Theory, Algorithms and Applications*; Springer Science & Business Media: Berlin, Germany, 2008.
45. Pollack, M. Letter to the Editor—The Maximum Capacity Through a Network. *Oper. Res.* **1960**, *8*, 733–736. [CrossRef]
46. Knuth, D.E. A generalization of Dijkstra's algorithm. *Inf. Process. Lett.* **1977**, *6*, 1–5. [CrossRef]
47. Habte, A.; Sengupta, M.; Lopez, A. *Evaluation of the National Solar Radiation Database (NSRDB): 1998–2015*; Technical Report; NREL (National Renewable Energy Laboratory (NREL)): Golden, CO, USA, 2017.
48. Office of Energy Efficiency & Renewable Energy (EERE). Commercial and Residential Hourly Load Profiles for all TMY3 Locations in the United States. 2013. Available online: http://en.openei.org/datasets/dataset/commercial-and-residential-hourly-load-profiles-for-all-tmy3-locations-in-the-united-states (accessed on 20 March 2019).
49. Energy Information Administration. Electric Utility Rates. 2017. Available online: http://en.openei.org/wiki/Utility_Rate_Database (accessed on 20 March 2019).

50. Avnet. ZedBoard (Zynq Evaluation and Development) Hardware User's Guide (Version 2.2). 2018. Available online: http://zedboard.org/sites/default/files/documentations/ZedBoard_HW_UG_v2_2.pdf (accessed on 20 March 2019).
51. Longbottom, R. *Raspberry Pi 32 Bit and 64 Bit Benchmarks and Stress Tests*; Technical Report. 2017. Available online: http://rgdoi.net/10.13140/RG.2.2.14460.64641 (accessed on 20 March 2019).
52. Raza, A.; Ikram, A.A.; Amin, A.; Ikram, A.J. A review of low cost and power efficient development boards for IoT applications. In Proceedings of the 2016 Future Technologies Conference (FTC), San Francisco, CA, USA, 6–7 December 2016; pp. 786–790. [CrossRef]