Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

Emulations on Energy Harvesting Embedded Systems

Performance and Conservativeness Interplay

Semester Thesis

Weikang Kong
wekong@ethz.ch

Computer Engineering and Networks Laboratory
Department of Information Technology and Electrical Engineering
ETH Zürich

Supervisors:
Naomi Stricker
Stefan Draskovic
Prof. Dr. Cong Liu
Prof. Dr. Lothar Thiele

3 July 2020

Acknowledgements

I would like to thank Naomi Stricker, Stefan Draskovic and Prof. Dr. Cong Liu for their supervision. They cleared my confusion and gave me insight into this topic with their professional knowledge, scientific reasoning and great patience. From them, I learnt more than just what is related to the content of this thesis, but also the qualities indispensable to crack scientific problems and go further.

Furthermore, I would like to thank Prof. Dr. Lothar Thiele for giving me the opportunity to work on this interesting semester project.

Abstract

For an energy harvesting embedded system, one possible implementation for long-term operation involves a predictor and a scheduler. Such a system's performance is affected by the accuracy of prediction, but not every system is equivalently affected, which means different systems are not equally robust against prediction inaccuracy. For systems that are not robust, their performances might drop drastically given less accurate prediction. To alleviate this issue, conservativeness is applied to the prediction and is shown to improve the performance of some systems. Furthermore, a typical wireless sensor node application is implemented to demonstrate how system performance and system behavior are related and how improved system performance is interpreted into better system behavior.

Contents

Chapter 1

Introduction

1.1 Motivation

Wireless sensor networks (WSN) are now broadly deployed in many areas. Some of them are battery-powered and deployed in remote locations that are hard to access for battery replacement [1, 2]. In this scenario, an energy harvesting embedded system is often used to support the long-term operation. For an application that requires a constant level of operation, designing the system with an energy transducer, a battery, and an energy sink would enable the simplest, and sometimes the most efficient implementation. However, for this system, to make sure the energy required by the application can always be supplied, the system should be over-dimensioned (e.g., for solar energy harvesting embedded system, the solar panel area and battery size should be larger than actually needed). Over-dimensioning the system yields extra wasted energy and a higher budget. Furthermore, not every system demands a constant operation level. For these systems, an energy predictor and a scheduler can be introduced to schedule the system's energy consumption in an adaptive manner. In this case, the system does not have to be over-dimensioned and can thus be deployed in a more economical way.

For an energy harvesting embedded system that involves a predictor and a scheduler, the dimensioning is based on the assumption of the environment and the expected system behavior. If the assumption reflects the reality, then the scheduled energy consumption will meet the design-time expectation. However, the assumption is never accurate, which means the performance of the system may not meet the design-time expectation. For some systems [3–5], the assumption and prediction are directly related, a more accurate assumption yields a more accurate prediction. Some systems' performances are not strongly affected by the prediction accuracy and are thus considered robust, while others vary a lot given predictions of different accuracies. In order to make the system performance more resilient to inaccurate predictions, the concept of conservativeness is introduced to tune the system robustness.

1.2 Organization

In chapter 2, we first briefly introduce solar energy harvesting embedded systems. Then, some prediction models and schedulers that are investigated in this report are discussed. Finally, it will be shown that different systems are not equally robust. In chapter 3, it is discussed how a scheduler's performance is affected by an inaccurate prediction. Furthermore, to alleviate the problem caused by inaccurate prediction, an intuitional solution, namely introducing conservativeness, is proposed. In chapter 4, emulation results show how conservativeness interplays with system performance. Lastly, in chapter 5, a real-world application demonstrates how system performance and system behavior are related and how improved system performance yields better system behavior.

Chapter 2

Solar Energy Harvesting Embedded System

Figure 2.1 shows the energy harvesting embedded system investigated in this report. It consists of five parts: harvested energy from the energy transducer (i.e., solar panel for this solar energy harvesting embedded system), a harvestable energy predictor (i.e., energy prediction model) which predicts the energy availability profile in future time slots, a scheduler which schedules the energy consumption profile, a battery which stores energy from energy surplus period so as to compensate for the energy consumption in energy deficit period, and an energy sink where energy is consumed.

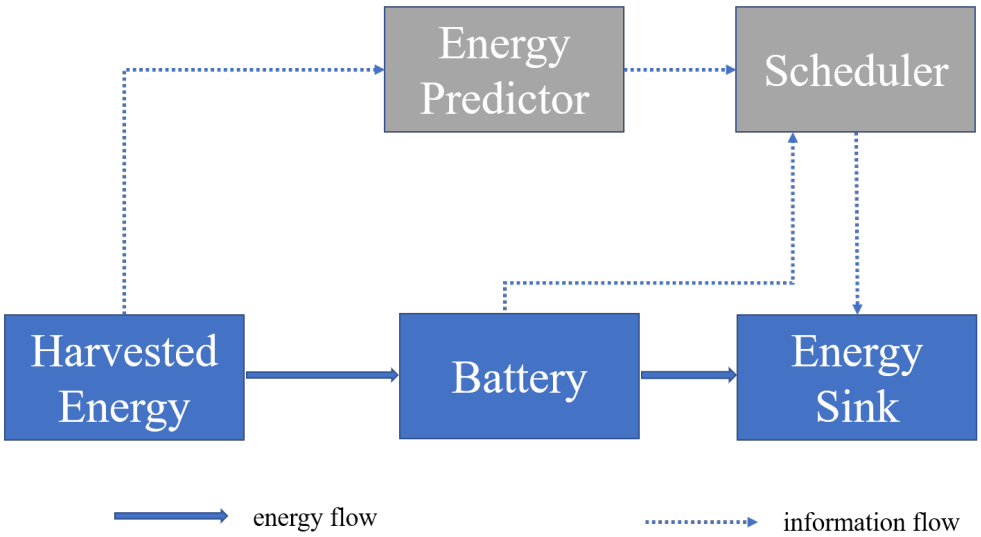

Figure 2.1: Investigated energy harvesting embedded system

This system is dimensioned based on the assumption on the environment and expected system behavior. At the start of each time slot, the scheduler

determines the energy consumption of that time slot. Then the energy sink consumes energy accordingly in that time slot. Along this process, the battery state of charge is updated based on the actual harvested energy, scheduled energy consumption, and battery state of charge at the previous time.

The actual energy consumption $E_{cons}(t)$ supported by the system in time slot t is bounded by three constraints. The first is energy consumption $E_{sch}(t)$ scheduled by the scheduler based on the prediction of harvestable energy $E_{pre}(t)$ in the time slot and some other information (e.g., battery state of charge). The second is the energy consumption when the system runs at full duty cycle E_{max}. The third is the maximum energy available in this time slot $E_{avai}(t)$, which is defined by the available energy in the battery at the start of this time slot $B(t)$ and the energy that could be harvested by the solar panel $E_{har}(t)$. Which means the actual energy consumption for time slot t is

$$E_{cons}(t) = Min\{E_{sch}(t), E_{max}, B(t) + E_{har}(t)\} \tag{2.1}$$

The energy consumption level in a time slot reflects the workload completed by the application (e.g., execution frequency, precision). A broad range of application scenarios benefit from a minimum supported performance level that can be sustained over time periods on the order of multiple years [3]. For these applications, it is crucial to maintain the energy consumption level over a certain threshold. Furthermore, it is desired that the system finishes more workload in total [6], so the total energy consumption is also something that should be noted.

To better understand a system's performance, it is indispensable to look into the prediction model and scheduler.

2.1 Prediction Model

Many prediction models have been proposed so far for solar energy prediction, such as an astronomical model, *exponentially weighted moving average* (EWMA) [4], *weather conditioned moving average* (WCMA) [7], and machine learning based predictors [8]. Two prediction models are investigated in the report.

2.1.1 Astronomical Model

This model computes the hourly solar energy incident by considering sky radiation and ground reflection based on the cloud-free atmospheric model. The original model, as firstly proposed in [9], requires some parameters that are difficult to quantify. To alleviate the parameter issue, the simplification model, as proposed in [10] is applied by adopting the parameters as shown in [11].

2.1.2 Exponentially Weighted Moving Average

This model assumes that the harvestable energy in a particular time slot should be similar to the harvestable energy in the same time slot on previous days. The prediction is generated based on the actual harvest energy in the previous day and the historical summary of the energy generation profile. This method could exploit the diurnal cycle in solar energy, and also adapts to seasonal variations [4].

2.2 Scheduler

2.2.1 Finite Horizon Control

As proven in [5], given perfect knowledge of the harvestable energy to be expected in the future (i.e., clairvoyant case), *finite horizon control* (FHC) is the optimal power management scheduler. Given perfect prediction, this scheduler never leads to failure state or wastes energy, besides the minimal used energy and the total utility are maximized. When available energy is increasing, battery is empty, since an empty battery marks the end of an energy deficit period. When available energy is decreasing, battery is full, since a full battery marks the end of a surplus period.

In order to achieve long-term minimum performance guarantees, it is reasonable to choose the horizon of control to be $T = 1year$. By assuming a periodic estimated harvested energy with $P = 1year$ and equal battery state of charge at the start and end of the period, offline periodic optimal control computes periodic battery state of charge and periodic optimal use function. Due to the inaccuracy of prediction, the battery state of charge may be different from what was expected in offline periodic optimal control, then online finite horizon control scheme will be performed to readjust the use function. If harvestable energy is underestimated, the battery state of charge at the end of a time slot is higher than expected, then the scheduler will arrange more energy consumption in the descendent slots to adjust the battery state of charge curve along time back to track. If harvested energy is overestimated, lower battery state of charge than expectation will be encountered, and thus leads to less energy consumption in the future, which sometimes leads to failure state.

2.2.2 Long-term Energy Neutral Operation

For *long-term energy neutral operation* (LT-ENO) [3], a target duty cycle is set, and all other parameters such as battery size and solar panel area are chosen based on the target duty cycle and the specific assumed environment represented with an astronomical model. If all parameters are chosen carefully,

the system will generate an energy consumption profile that fluctuates around the target duty cycle. The deviation of the performing duty cycle (i.e., the actual duty cycle executed) from the target duty cycle is determined by the prediction error in the past. If parameters are chosen carefully, when past data indicate overestimation, the algorithm will shrink the span of surplus period and extend the span of deficit region, trying to improve the performing duty cycle as close to the target duty cycle as possible; When past data indicate underestimation, the algorithm tries to lower down the performing duty cycle as close to the target duty cycle as possible.

2.2.3 Duty Cycle Adaptation

Duty cycle adaptation (DCA) [4] intends to use as much energy as could be harvested in the day and aims for energy neutrality by always trying to compensate for prediction errors in previous time slots. At the start of each day, initial duty cycles for each time slot in this day are assigned, and then the target duty cycles in future time slots will be adjusted based on prediction errors in previous time slots at run time.

2.3 Robustness of Energy Harvesting Embedded Systems

There is no available rigorous definition of the robustness of the performance of an energy harvesting embedded system in currently available publications. From an ongoing study on robustness analysis in the group [1], if a system's performance metric remains unchanged or only slightly changes given predictions with different accuracies, then the system is considered robust with respect to this metric. The drastic change in the system performance given predictions of different accuracies may not be wanted by some applications. For applications requiring minimum energy consumption guarantee, a system with poor robustness will bring too much uncertainty to its performance. In this case, a more robust system may be preferred.

Three different systems are evaluated, these systems adopt FHC and astronomical model, LT-ENO and astronomical model, DCA and EWMA as the scheduler and prediction model, respectively. When considering minimal daily energy consumption, LT-ENO is shown to be the most robust among these three, and DCA tends to be robust in the low-error region but less robust in the high-error region, while FHC is revealed to be the least robust system. When considering total utility, FHC and LT-ENO are both very robust, while DCA is

[1] By Naomi Stricker, https://tec.ee.ethz.ch/the-group/people/person-detail.MTgxMjAw.TGlzdC8yMjM0LC03MTAxNDI4MDI=.html

unrobust.

FHC is shown to have better minimal daily energy consumption and total utility than the other two [5], which gives designers preference of this scheduler over the other two at design time. However, if taking robustness into consideration, designers might make a different choice.

Chapter 3

Conservativeness Analysis

For systems that are not robust, their performances may change drastically towards unfavorable direction when prediction error is significant. Note that prediction can be wrong in two ways: overestimation and underestimation. In this chapter, it is going to be discussed how prediction error and the scheduler's performance are quantified. Furthermore, the issue caused by overestimation will be discussed, and one possible solution to the issue, namely introducing the concept of conservativeness, will be analyzed.

3.1 Analysis Metrics

3.1.1 Prediction Accuracy Metric

Many metrics can be used to quantify prediction accuracy, for example, average absolute error [12], mean absolute percentage of error [13] and median of error [14]. In this report, median of error is chosen to be the metric. The prediction error for each time slot is defined as

$$err(t) = \frac{E_{pre}(t) - E_{har}(t)}{E_{har}(t)} \tag{3.1}$$

where $E_{pre}(t)$ is the harvestable energy prediction for time slot t, and $E_{har}(t)$ is the actual energy harvested in time slot i.

The median of error is defined as

$$Err_{med} = median(\overrightarrow{err}) \tag{3.2}$$

where $\overrightarrow{err}$ is a vector that contains the prediction errors for all time slots.

If the median of error is larger than zero, it means over half of the prediction results are overestimations compared to reality. If the median of error is smaller than zero, then over half of the prediction results are underestimations.

3.1.2 Performance Metrics

As mentioned in chapter 2, a broad range of application scenarios benefit from a minimal supported performance level that can be sustained over time periods on the order of multiple years [3]. Hence an important metric for system performance is the minimal daily energy consumption.

At the same time, total energy consumption is also a metric of interest. However, directly define the metric as the sum of energy consumptions in all time slots is problematic, because in this way the system that uses as much energy as available would be optimal. Noting that the importance of additional energy in case of low energy consumption outweighs the one in case of high energy consumption [5], the metric that quantifies the total workloads finished is defined as

$$U(t_1, t_2) = \sum_{\tau=t_1}^{t_2-1} \mu(u(\tau)) \tag{3.3}$$

where $\mu(u(t)) = \sqrt{u(t)}$ and $u(t)$ denotes the energy consumption in time slot t. The concave function applied to energy consumption enables this metric to reflect the fact that additional energy in case of high energy consumption brings less benefit compared to the case of low energy consumption.

3.2 Theoretical Analysis

Both overestimation and underestimation can cause meager energy consumption. The meager energy consumption is sometimes related to the empty or almost-empty battery caused by overestimation.

An intuitional method to alleviate the meager energy consumption caused by overestimation is introducing conservativeness. Conservativeness can be applied in different manners. One is to apply it to scheduling result, which corresponds to generally use less energy than scheduled. Assume conservativeness is applied to one time slot, by using less energy than scheduled, the battery state of charge in this time slot is no less than when conservativeness is not applied. Another method is to apply conservativeness to prediction so that the prediction becomes more pessimistic. These two manners correspond to the direct reason for meager energy consumption: an empty or almost-empty battery, and the fundamental reason: overestimation, respectively.

In this report, to alleviate the problem caused by overestimation, conservativeness is applied in the following manner

$$\overrightarrow{E_{pre}}' = C * \overrightarrow{E_{pre}} \tag{3.4}$$

where $\overrightarrow{E_{pre}}$ is the vector that contains original prediction for all time slots, $\overrightarrow{E_{pre}}'$ is the scaled version of prediction, and $C \in (0, 1]$ is the scaling factor.

After applying conservativeness, underestimation will become stronger. As for overestimation, depending on the extent of overestimation, it might be rendered into underestimation or weaker overestimation.

For some systems under some environments, applying conservativeness should be able to improve minimal energy consumption. In this case, stronger conservativeness does not always yield better minimal energy consumption. When conservativeness is so strong that the prediction in some time slots become too underestimated, the scheduler will determine a low energy consumption due to the pessimistic prediction. When aiming for maximal minimum energy consumption, there is an optimal conservativeness factor C_{opt} that leads to maximal minimum energy consumption for each environment.

Chapter 4

Emulation Setup and Results

In this chapter, firstly, the emulation setup will be introduced, then it is going to be investigated how conservativeness interplays with a system's performance with emulations on real-world data.

4.1 Emulation Setup

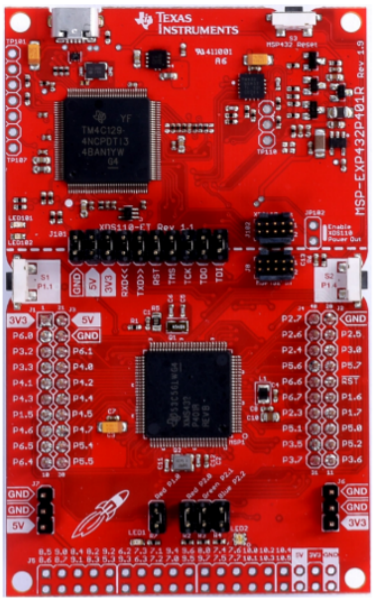

Figure 4.1: MSP432P401R

Due to the current special circumstance, conducting the entire experiment on solar testbed is unrealistic. Given what is available under work-from-home environment, emulations are conducted on a Texas Instrument MSP432P401R launch pad (figure 4.1), where the energy input is defined based on the dataset

and the battery state of charge is calculated instead of measured. The energy sink is the tri-LED on the MSP board. By using the EnergyTrace^{TM} technology in *code composer studio* (CCS), the maximum power dissipation of the LED is measured to be $81.16mW$.

4.1.1 Input Data

The input data is acquired from National Solar Radiation Database (NSRD)[1]. Hourly raw data from Ontario, Michigan and California are aggregated into daily and weekly frames to serve as the inputs of the three schedulers. Note that the first-year raw data is used to fit the parameters in the astronomical model, so these data cannot be the input for the schedulers. The actual input data have a time span of 10 years from Jan. 1999 to Dec. 2008.

4.1.2 Time Scaling

Since it is time inefficient to conduct the emulation in a real-time manner to observe a system's long-term performance, time can be compressed to speed up the emulation process. One week is scaled to $17.92s$. Accordingly, one day is scaled to $2.56s$. As for hourly data, however, considering that the computation overhead for the algorithm that uses hourly frame, namely DCA, is around 0.1s, it is safer to scale one hour to $0.16s$. Note than when applying different scaling factors, all related parameters should also be scaled proportionally to make sure the system is dimensioned in the same way as the original one.

The parameters are all scaled proportionally in a linear manner, and the schedulers all perform linear operations. By applying different scaling factors, the energy consumption profiles should have the same trend, and the profiles should be identical after being scaled back to real-world time. This has been shown by emulation results. However, even though the emulation involves the MSP board, it is still far from experiments on a solar testbed, not to mention actual deployments. For example, different scaling factors correspond to different battery sizes, yet batteries of different sizes may have different charging and discharging inefficiencies [15]. Different inefficiencies will lead to different energy consumption profiles. However, the difference in discharging inefficiencies can be approximated by scaling the values in energy consumption profile [11]. If this method could be applied to other inefficiencies, then the difference between the inefficiencies of time-compressed system and actual system does not significantly affect the trend of energy consumption profile.

[1] http://rredc.nrel.gov/solar/old˙data/nsrdb/

4.1.3 Choice of Parameters

Three different systems are evaluated, these systems adopt FHC and astronomical model, LT-ENO and astronomical model, DCA and EWMA as the scheduler and prediction model, respectively.

A.General Parameters For an energy harvesting embedded system, it is reasonable to assume that the hardware part remains unchanged if deployed at different locations, because it would be a competitive feature of the system to be deployed under different environments without changing hardware. For example, users might expect to deploy the system in an open outdoor environment or under the tree without changing the battery.

Three different systems are investigated under three different environments. In these nine different scenarios, the systems are dimensioned in the same way, including battery size.

In the following emulation, a solar panel with an area of $16.5cm^2$, conversion efficiency of 5.65% and maximum power output of $87.6mW$ is assumed as the transducer. The battery size for the time-compressed emulations is $3.0J$ for FHC and LT-ENO, whereas $4.5J$ for DCA due to its computation overhead issue. In an actual deployment, the battery size should be accordingly 33750 times larger, which is $28.125Wh$. Battery charging inefficiency is 90%. All other efficiencies are ignored.

B.Scheduler-specific Parameters

Finite Horizon Control: Since the scheduler is designed for long-term operation, the period is set as $P = 1year$, horizon of control is set as $T = P = 1year$ and stopping criterion is set as $\epsilon = 10^{-4}$ (see section 4.2.1).

LT-ENO: For LT-ENO, the battery size should satisfy

$$E_{deficit} < B < E_{surplus} \tag{4.1}$$

where $E_{deficit}$ and $E_{surplus}$ are determined by the target duty cycle and prediction model [3]. The battery size determined for LT-ENO is also applied to other two systems.

The concept of target duty cycle is directly used as a crucial parameter in LT-ENO. Since battery size remains unchanged for different datasets, it is crucial for this scheduler to choose a proper target duty cycle. If not chosen properly, the condition (4.1) cannot be satisfied, and the scheduler may never enter the energy surplus and deficit regions readjustment procedure, which makes it senseless to use this scheduler. The target duty cycle is chosen based on the mean of energy input prediction and tuned a bit so that when using different energy input traces the battery size could remain unchanged without violating the condition (4.1). The target duty cycles for the Ontario, Michigan and California datasets are

$C*15.0\%$, $C*16.4\%$ $C*20.7\%$ respectively, where C denotes the conservativeness factor.

The scheduler updates the prediction model at run time. The window size is chosen to be $N_w = 63$ as suggested in [3]. The first year's input data is not incorporated in the emulation so it can be used to initialize the prediction model scalar α.

Duty Cycle Adaptation: The paper [4] suggests performing the scheduling process every half hour. However, due to the time frame of available raw data, the scheduling can be performed at best in an hourly manner and thus set the window size $N_w = 24$. The maximum and minimum duty cycles are $D_{max} = 1.0$ and $D_{min} = 0.1$, respectively. Predictor is chosen to be EWMA and the weighting factor is 0.5.

4.1.4 Power Dissipation Controlling

Algorithm 1: Power dissipation controlling

Input: Target power dissipation PC, minimum power dissipation P_{min}
Output: The state of tri-color LED: designated power dissipation of each LED PC_r, PC_g and PC_b

```
if PC ≤ P_min then
    Failure state
else
    if P_min < PC ≤ P_min + P_r then
        PC_r = PC − P_min, PC_g = 0, PC_b = 0
    else
        if P_min + P_r < PC ≤ P_min + P_r + P_g then
            PC_r = PC − P_min − P_g, PC_g = P_g, PC_b = 0
        else
            if P_min + P_r + P_g < PC ≤ P_min + P_r + P_g + P_b then
                PC_r = PC − P_min − P_g − P_b, PC_g = P_g, PC_b = P_b
            else
                Infeasible
```

The MSP board has a minimum power dissipation $P_{min} = 1.866mW$. This is measured by turning off the energy sink(s) but keeping other peripherals (e.g., timer) working. The LED used as the energy sink is a tri-color Led. Red, green and blue LEDs can be lit independently. The maximum power dissipations for them are $P_r = 31.966mW$, $P_g = 25.429mW$ and $P_b = 21.895mW$ respectively. The red color LED is mapped as the output of pulse width modulation, which means the power dissipation of the red LED can be roughly controlled around any level smaller than $31.966mW$. The other two LEDs can only be turned on

at maximum illumination or turned off. The actual power consumption PC of the energy sink is defined as

$$PC = \frac{E_{cons}}{L} \tag{4.2}$$

where E_{cons} is the scheduled energy consumption at some time slot, L is the length of that time slot. When power dissipation is available, the state of each LED in the tri-color LED can be determined by applying algorithm 1.

4.2 Emulation Results

4.2.1 Choice of Stopping Criterion for FHC

In emulation, an important consideration is the time cost. FHC determines the explicit solutions to the optimal power management problem in an iterative manner. The stopping criterion ϵ controls when the iteration stops. A smaller stopping criterion yields better solution precision yet higher computation overhead. At design time, the trade-off between precision and computation overhead must be performed. By setting parameters in the way as shown in section 4.1.3 and emulating on CA dataset, the computation overheads for different stopping criteria ϵ and different data types are listed in table 4.1 and table 4.2.

ϵ	Time [s]
10^{-1}	0.01
10^{-2}	0.02
10^{-3}	1
10^{-4}	2
10^{-5}	4
10^{-6}	6
10^{-7}	7
10^{-8}	7
10^{-9}	7
10^{-10}	7
10^{-20}	7
10^{-30}	7
10^{-40}	7

Table 4.1: Computation overhead when data type is float

As can be observed from the tables, using double as the data type yields much higher computation overhead. Due to the higher precision of double type, the computation overhead stops increasing with a much smaller stopping criterion compared to using float as the data type. In real-world applications, the computation overhead is small compared to the time span of one week, so the

ϵ	Time [s]
10^{-1}	1.4
10^{-2}	7
10^{-3}	22
10^{-4}	39
10^{-5}	59
10^{-6}	76
10^{-7}	96
10^{-8}	116
10^{-9}	135
10^{-10}	155
10^{-11}	176
10^{-12}	195
10^{-13}	218
10^{-14}	237
10^{-20}	237
10^{-30}	237

Table 4.2: Computation overhead when data type is double

computation overhead is negligible. To make sure the computation overhead is still negligible in time-compressed version while maintaining an acceptable emulation time cost, using double as the data type is ruled out.

With float as the data type, if using a stopping criterion smaller than 10^{-4}, the performance metric does not change too much, yet computation overhead increases significantly. By performing the trade-off between precision and time cost, the stopping criterion is chosen to be 10^{-4} with float as the data type.

4.2.2 Performance and Conservativeness Interplay

Three different conservativeness factors are applied to the original prediction. A smaller conservativeness factor means stronger conservativeness. The scenarios where $C = 1.0$ are the comparison baselines. How minimum energy consumption, prediction error, and conservativeness factor interplay is shown in table 4.3. The percentages in the parentheses represent the changes in metrics in proportion to the baseline case.

Both overestimation and underestimation could contribute to the decrease in energy consumption. By analyzing the emulation results on ON, MI and CA datasets, it appears minimum energy consumption takes place in two different patterns: one case is that minimum energy consumption is still in the trend of the trace, as is shown in figure 4.2; the other is that minimum energy con-

Scheduler	Dataset	Median of Error	Minimum Daily Energy Consumption [J] (scaled back to actual time)		
			C = 1.00 (baseline)	C = 0.95	C = 0.90
FHC	ON	4.12%	606.04	689.19 (+13.72%)	957.51 (+57.99%)
	MI	10.00%	475.84	509.37 (+7.05%)	717.77 (+50.84%)
	CA	-3.61%	1033.35	1458.64 (+41.16%)	1424.03 (+37.81%)
DPS	ON	2.21%	194.00	194.00 (0.00%)	194.00 (0.00%)
	MI	3.13%	202.68	202.68 (0.00%)	202.68 (0.00%)
	CA	-8.64%	258.96	258.96 (0.00%)	258.96 (0.00%)
DCA	ON	0.00%	115.72	109.94 (-5.00%)	104.15 (-10.00)
	MI	0.00%	182.17	173.06 (-5.00%)	163.95 (-10.00%)
	CA	0.00%	271.33	257.76 (-5.00%)	244.20 (-10.00%)

Table 4.3: Minimum daily energy consumption for different scenarios

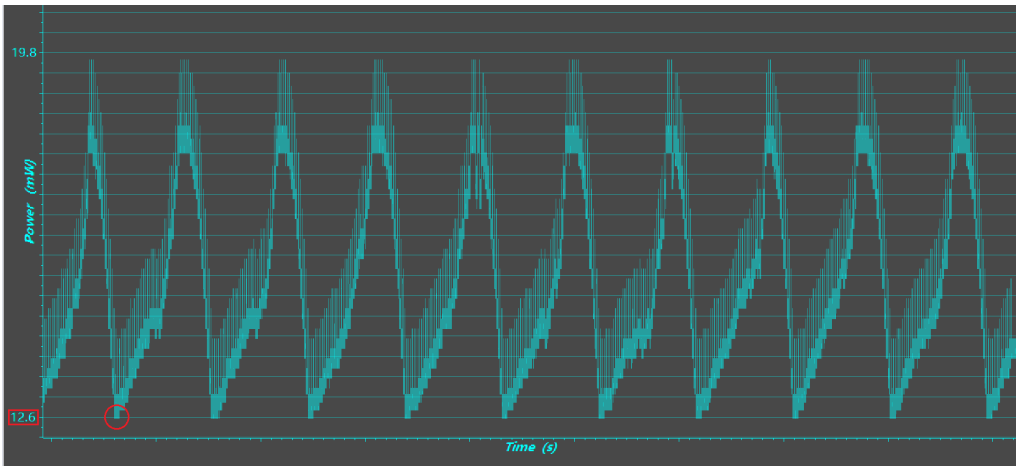

Figure 4.2: Minimum energy consumption still in the trend

sumption apparently deviates from the trend, as is shown in figure 4.3, which is often related to an unexpected empty or almost-empty battery state of charge caused by overestimation at the start of current time slot. The minimum energy consumption in the former case is not too low and can still support the basic operation of the system in this shown example. However, the latter case is likely to witness meager energy consumption that cannot even support the system's basic operation and, in turn, leads to the failure state.

For FHC, the medians of errors for ON and MI datasets are both positive, which means over half of their predictions are overestimated. For these two datasets, when a conservativeness factor of 0.95 is applied, their minimum energy consumption all increase, and when a stronger conservativeness factor 0.9 is applied, their minimum energy consumptions improve even more. For CA dataset, over half of the predictions are underestimated. Applying a conservativeness fac-

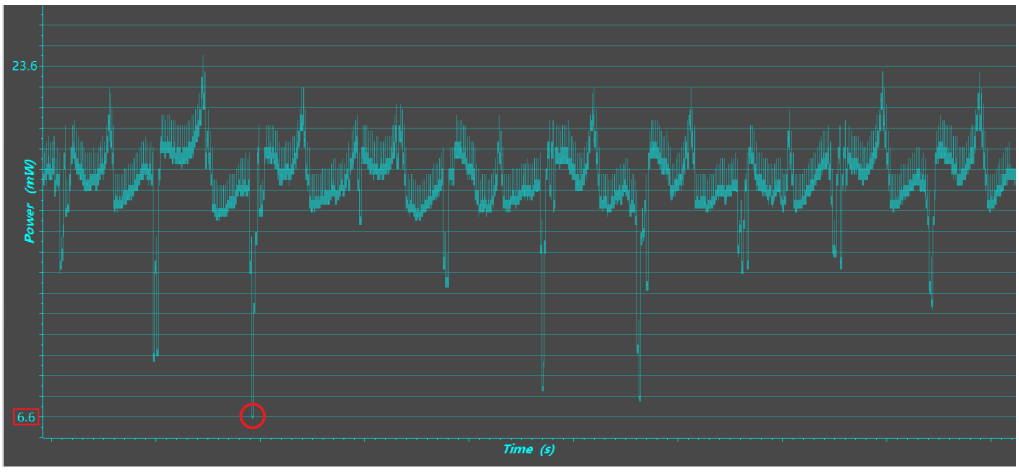

Figure 4.3: Minimum energy consumption not in the trend

tor of 0.95 helps improve its minimum energy consumption, however, applying a conservativeness factor of 0.9 yields a smaller minimum energy consumption than when 0.95 is applied, but still, it is better than baseline. From these data, it looks like there is an turning point (i.e., optimal conservativeness factor) after which the performance of the system stops having more benefits from stronger conservativeness. As mentioned in 3.2, this happens because if conservativeness is too strong, then the scheduler will schedule the energy consumption based on the over-pessimistic prediction and determine a rather low energy consumption. By conducting a simulation, it is found that the optimal conservativeness factors C_{opt} for ON, MI and CA datasets are 0.915, 0.892 and 0.977 respectively. Note that the prediction for MI dataset is the most overestimated and for CA the least overestimated with median of error as the metric. From the results, it appears that a more overestimated prediction requires a stronger conservativeness factor to achieve optimal performance.

For DCA, it looks surprising that the median of errors for all the three datasets are 0. Considering that this scheduler adopts hourly data as input, nearly half of the time slots have zero actual harvested energy, having zero as the median of prediction errors is reasonable. With conservativeness factor applied, the minimum energy consumption decreases, and stronger conservativeness yields smaller minimum energy consumption.

LT-ENO has a built-in prediction scaling mechanism. Due to the prediction scaling mechanism, conservativeness does not affect the scaled version of prediction, which leads to unchanged system performances when different conservativeness factors are applied. But if we skip this mechanism and directly apply conservativeness to the scaled version of prediction, we can still observe an improvement in minimum energy consumption.

Scheduler	Dataset	Median of Error	Total Utility		
			C = 1.00 (baseline)	C = 0.95	C = 0.90
FHC	ON	4.12%	44927.25	44721.49 (-0.46%)	44373.02 (-1.23%)
	MI	10.00%	46663.29	46646.05 (-0.04%)	46471.96 (-0.41%)
	CA	-3.61%	54394.21	54048.52 (-0.64)	53407.16 (-1.82)
DPS	ON	2.21%	44429.59	44429.59 (0.00%)	44429.59 (0.00%)
	MI	3.13%	46552.93	46552.93 (0.00%)	46552.93 (0.00%)
	CA	-8.64%	54419.40	54419.40 (0.00%)	54419.40 (0.00%)
DCA	ON	0.00%	42557.50	42523.29 (-0.01%)	42520.87 (-0.01%)
	MI	0.00%	45020.18	45006.14 (-0.00%)	45002.19 (-0.00%)
	CA	0.00%	53590.95	53589.52 (-0.00%)	53588.66 (-0.00%)

Table 4.4: Total utility for different scenarios

Dataset	Wasted Energy [J]		
	C = 1.0 (baseline)	C = 0.95	C = 0.90
ON	33416.5	67372.91	121607.42
MI	0.00	992.55	29078.85
CA	2552.90	69625.07	195063.05

Table 4.5: Wasted energy for FHC

As can be observed from table 4.4, total utility is barely affected by conservativeness.

There is an intuition explanation for the barely changed total utility. Total utility is possibly affected by conservativeness in two ways: redistribution of scheduled energy consumption and change in the amount of wasted energy. If the energy consumption in one time slot is already high, then additional energy does not significantly impact the utility of this time slot, if the energy consumption is low, then additional energy would drastically improve utility in that time slot. By applying conservativeness, the meager energy consumption in one time slot is alleviated, which brings benefit to total utility. However, by applying conservativeness, the battery state of charge is more likely to remain at a high level, which may waste more energy due to full battery, as is shown in table 4.5.

Chapter 5

Environmental Data Sensing Application

In this chapter, a real-world application will be implemented. The influence of conservativeness on a system's minimum energy consumption is translated into the influence on system behavior.

5.1 Application Description

In this application, a mote in a sensor network collects environmental information such as ambient temperature, ambient humidity, air pressure and illumination level. Data are stored in the flash after collection. At the end of a time slot, the data are sent out.

5.2 Implementation Detail

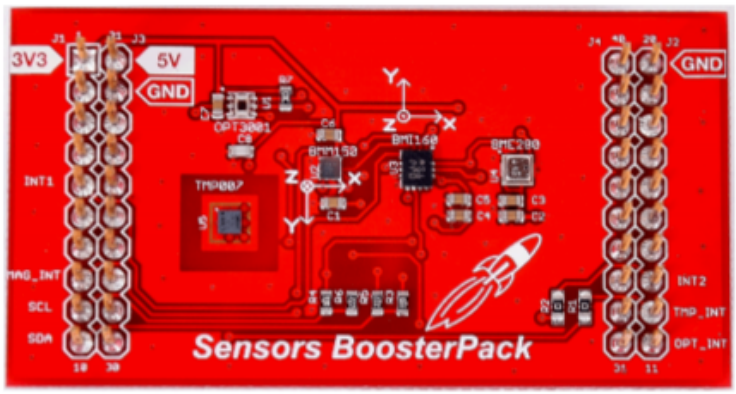

Figure 5.1: MSP432P401R

The sensing data are collected using the sensors on Texas Instrument BOOSTXL-SENSORS BoosterPack plug-in module (figure 5.1). Due to the inaccessibility of wireless transmission modules (e.g., bluetooth module), the data sending part is realized with UART. The energy consumptions on data storage and wired data sending are almost negligible, and in a real-world application the energy consumption on the computation overhead of scheduler is also neglected due to its low occurrence, so to roughly control the energy consumption in one time slot, one only has to roughly control the energy consumed by data measurement and storage. When the energy consumption of one execution of measurement and storage operation, the number of executions taken in one time slot can be determined, which can then be interpreted into the time interval between two consecutive executions. In this way, the energy consumption can be only controlled discretely, and a longer time slot makes more precise energy consumption controlling possible. In the following emulation, FHC is used as the scheduler, astronomical model as the predictor; one week is scaled to 60 seconds; the battery size for the time-compressed emulations is increased to $6.0J$, corresponding to a battery size of $16.8Wh$ in an actual deployment; the rest of the parameters are the same as the FHC part in the previous chapter.

5.3 Emulation Results

5.3.1 Energy Consumption for One Execution

To control the energy consumption in each time slot, the energy consumption for each execution of data measurement and storage operation should be determined. By using EnergyTrace^{TM} technology in CCS, it is measured when there is only one execution in the 60-second time slot, the energy consumption is $436.800mJ$, when there are 360 executions, the energy consumption is $880.140mJ$. The energy consumption for each execution is calculated as $1.235mJ$. The energy consumed by the system when there is no execution is $435.565mJ$.

5.3.2 System Behavior Influenced by Conservativeness

How conservativeness influences system behavior is shown by emulating the system on CA dataset. We can see three time slots in figure 5.2. There are two bars between time slots: the lower one represents the data sending process, and the higher one represents the computation overhead of FHC. Each pulse represents one measurement and storage operation. The empty time slot in the middle means there is no data collected. This happens because the available energy in this time slot is insufficient to support the basic functionality of the system. As mentioned in previous chapters, this energy insufficiency is caused

by overestimation.

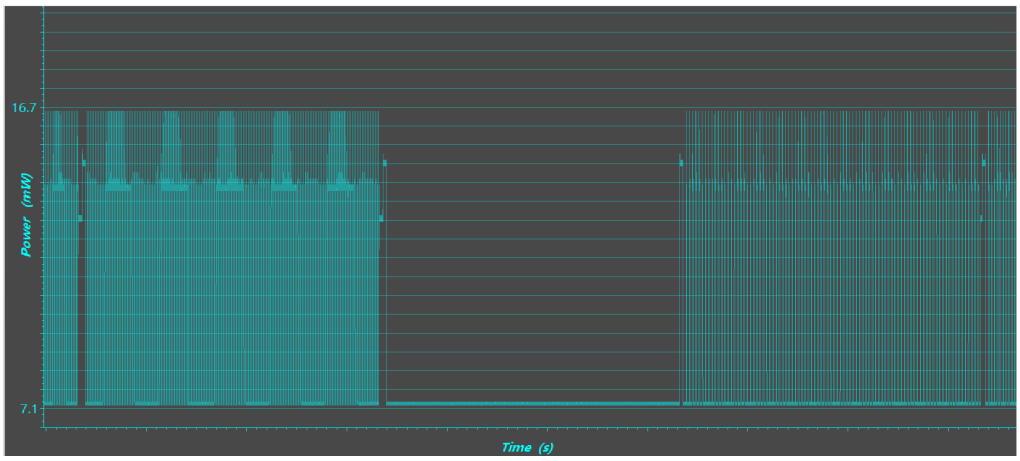

Figure 5.2: System behavior without conservativeness

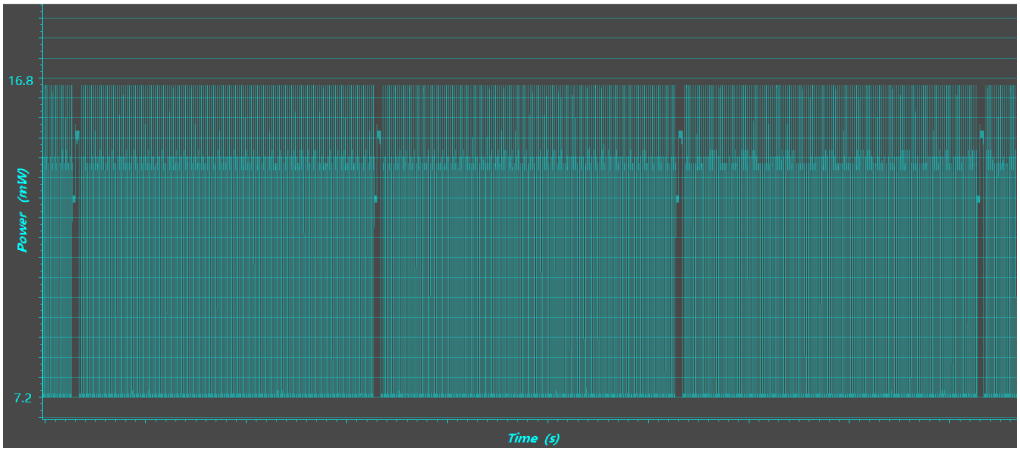

Figure 5.3: System behavior with conservativeness factor $C = 0.9$

Conservativeness can alleviate overestimation and thus improve minimum energy consumption. As can be seen in figure 5.3, after applying a conservativeness factor $C = 0.9$ to the prediction, there is no empty time slot.

Chapter 6

Conclusion

For some systems, applying conservativeness improves system's minimum energy consumption. But it does not mean stronger conservativeness guarantees better minimum energy consumption. For these systems, there is an optimal conservativeness factor for each environment. When conservativeness factor is larger than the optimal factor, smaller conservativeness factor improves minimum energy consumption. When conservativeness factor is smaller than this optimal factor, smaller conservativeness factor hurts minimum energy consumption.

Bibliography

[1] Henry A Sodano, Gyuhae Park, and DJ Inman. Estimation of electric charge output for piezoelectric energy harvesting. *Strain*, 40(2):49–58, 2004.

[2] Kamarul Zaman Panatik, Kamilia Kamardin, Sya Azmeela Shariff, Siti Sophiayati Yuhaniz, Noor Azurati Ahmad, Othman Mohd Yusop, and SaifulAdli Ismail. Energy harvesting in wireless sensor networks: A survey. In *2016 IEEE 3rd international symposium on Telecommunication Technologies (ISTT)*, pages 53–58. IEEE, 2016.

[3] Bernhard Buchli, Felix Sutton, Jan Beutel, and Lothar Thiele. Dynamic power management for long-term energy neutral operation of solar energy harvesting systems. In *Proceedings of the 12th ACM conference on embedded network sensor systems*, pages 31–45, 2014.

[4] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B Srivastava. Power management in energy harvesting sensor networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 6(4):32–es, 2007.

[5] Bernhard Buchli, Pratyush Kumar, and Lothar Thiele. Optimal power management with guaranteed minimum energy utilization for solar energy harvesting systems. In *2015 International Conference on Distributed Computing in Sensor Systems*, pages 147–158. IEEE, 2015.

[6] Strahinja Janković and Lazar Saranovac. Improving energy usage in energy harvesting wireless sensor nodes using weather forecast. In *2017 25th Telecommunication Forum (TELFOR)*, pages 1–4. IEEE, 2017.

[7] Joaquin Recas Piorno, Carlo Bergonzini, David Atienza, and Tajana Simunic Rosing. Prediction and management in energy harvested wireless sensor nodes. In *2009 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology*, pages 6–10. IEEE, 2009.

[8] Selahattin Kosunalp. A new energy prediction algorithm for energy-harvesting wireless sensor networks with q-learning. *IEEE Access*, 4:5755–5763, 2016.

[9] JV Dave, P Halpern, and HJ Myers. Computation of incident solar energy. *IBM Journal of Research and Development*, 19(6):539–549, 1975.

[10] Bernhard Buchli, Felix Sutton, Jan Beutel, and Lothar Thiele. Towards enabling uninterrupted long-term operation of solar energy harvesting embedded systems. In *European Conference on Wireless Sensor Networks*, pages 66–83. Springer, 2014.

[11] Rehan Ahmed, Bernhard Buchli, Stefan Draskovic, Lukas Sigrist, Pratyush Kumar, and Lothar Thiele. Optimal power management with guaranteed minimum energy utilization for solar energy harvesting systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(4):1–26, 2019.

[12] Muhammad Hassan and Amine Bermak. Solar harvested energy prediction algorithm for wireless sensors. In *2012 4th Asia Symposium on Quality Electronic Design (ASQED)*, pages 178–181. IEEE, 2012.

[13] Amit Kumar Yadav and SS Chandel. Solar radiation prediction using artificial neural network techniques: A review. *Renewable and sustainable energy reviews*, 33:772–781, 2014.

[14] Y Poissant, S Pelland, and D Turcotte. A comparison of energy rating methodologies using field test measurements. In *23rd European PV Solar Energy Conference and Exhibition, Valencia, Spain*, 2008.

[15] Donghwa Shin, Younghyun Kim, Jaeam Seo, Naehyuck Chang, Yanzhi Wang, and Massoud Pedram. Battery-supercapacitor hybrid system for high-rate pulsed load applications. In *2011 Design, Automation & Test in Europe*, pages 1–4. IEEE, 2011.