On the Implementation of LMS-based Algorithm forIncreasing the Lifetime of IoT Networks

by

Anish Shastri, Vivek Jain, Rhishi Pratap Singh, Sachin Chaudhari, Shailesh Chouhan

in

IEEE International Conference on Advanced Networks and Telecommunications Systems

Radisson Blu, Indore, India

Report No: IIIT/TR/2019/-1

Centre for Communications
International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2019

On the Implementation of LMS-based Algorithm for Increasing the Lifetime of IoT Networks

Anish Shastri*, Vivek Jain*, Rhishi Pratap Singh*, Sachin Chaudhari*, Shailesh Singh Chouhan†
*Signal Processing and Communication Research Center
International Institute of Information Technology - Hyderabad, India
†Embedded Internet Systems Lab, Luleå University of Technology, Luleå, Sweden
Email: {anish.shastri, rhishi.pratap}@research.iiit.ac.in, jain.vivek@student.iiit.ac.in,
sachin.chaudhari@iiit.ac.in, shailesh.chouhan@ltu.se

***Abstract*—This paper focuses on the customized-wireless sensor node implementation of the classical least mean square (LMS) algorithm for the reduction in data-transmissions from the sensor nodes to the sink in internet of things (IoT) networks. This reduction, in turn, increases the battery life of the sensor node. The system was deployed in outdoor and indoor environments to read the ambient temperature and then perform the prediction of the sensed data in order to minimize the number of data-transmissions to the sink node. The utility of the proposed concept has been demonstrated using the measured data and the battery life is increased 2.64 and 2.53 times in indoor and outdoor environments, respectively.**

***Index Terms*—LMS, IoT, sensor node lifetime, data-transmission reduction, IEEE 802.15.4.**

I. Introduction

Internet of things (IoT) is enabling new objects to connect to the internet. These connected devices are generating data and collaboratively dispensing meaningful insights about the surrounding physical phenomenon. For example, with the help of versatile wireless sensors [1], smart buildings are making living spaces more pleasant and resource efficient by continuously monitoring the environment. By providing real-time information about pollution, traffic, garbage disposal and other amenities [2], internet-connected wireless sensor nodes help people in a smart city.

A typical wireless sensor node operates on limited battery power. It is well known that the communication cost is much higher than the computation cost [3]. With an increase in the number of IoT nodes, more data will be generated. This data is required to be sent to the sink for further processing. In a wireless network, the communication cost will increase with increased data, which in-turn reduces the battery life of the IoT node. It motivates us to save the battery by selectively transmitting the data, i.e., transmit only the required information across the wireless network. This advent the task of data-transmission reduction in IoT networks, which has been a very prominent area of research in the recent times.

Data prediction is one of the key techniques for reducing data-transmissions from sensor nodes to the sink node in a wireless sensor networks [3]. Data prediction consists of building an abstraction of a sensed phenomenon, i.e., a model describing data evolution [4]. In this method, there are prediction models running simultaneously on both source and sink, with sink predicting the source value within some error bound. This is known as dual prediction scheme. If the error is within the bound, no communication takes place between the source and sink nodes. In contrast, when the error exceeds the bound, the data is transmitted.

In WSN literature, data-transmission reduction using the data prediction strategy has been done in [5]–[9]. In [5], [6], the authors have compared prediction algorithms such as least mean square (LMS), fixed weight moving average, and autoregressive integrated moving average (ARIMA) by applying them on a standard dataset as well as on the dataset generated by their own testbed. The current consumed in different modes of operation of the wireless mote has been mentioned. However, current consumption over the duty-cycle and lifetime extension of the motes has not been compared. In [8], the algorithm has been implemented on an FPGA but the performance evaluation in terms of power consumption has not been reported. In addition, it is not feasible to deploy multiple FPGAs for monitoring networks in a real-time situation. In [7], a convex combination of two decoupled and different sized LMS filters is used for prediction of the temperature values. In [9], dual Kalman filter is used as prediction model which requires all the nodes to be fed with the same model to work coherently.

In this paper, we have employed the LMS algorithm [10] as it provides very low computational overhead and greater flexibility compared with other filter algorithms. LMS algorithm is an approximate steepest descent algorithm, in which the filter is only adapted based on the error at the current time in order to minimize the mean square error. Unlike standard gradient descent filter, LMS is a stochastic gradient filter. It does not require any *a priori* model information or need to maintain correlation matrices. This makes it more flexible to apply for a vivid range of phenomenon sensing. On the other hand, it is mandatory to have statistical features of the

978-1-5386-8134-3/18/$31.00

data in time-series prediction methods such as ARIMA. LMS filter does not require nodes to be assisted by a central entity for performing prediction. Since no global model parameters need to be defined, this makes it possible to apply for a variety of networks, including that of with data aggregation schemes.

In the previous works [5]–[7], LMS is tested and used only on standard datasets and very few hardware implementations or measurements were carried out. Thus, there is a dearth of low power implementation of LMS algorithm on a basic hardware platform and validating the gains obtained through LMS algorithm in theory. Specific contributions of this paper are:

- Low cost and low energy implementation of LMS algorithm for data-transmission reduction are employed.
- To achieve low cost, the wireless-sensor node was implemented using the commercially available IoT-enabled hardware such as ATmega328P as the microcontroller, XBee S2C as IEEE 802.15.4 radio and DS18B20 as the temperature sensor.
- To obtain low power consumption during operation, various microcontroller programming-based strategies were used. To further reduce the power consumption, the duty-cycle-controlled transmission strategy was adopted.
- To demonstrate the efficiency of the proposed approach, measurements have been carried out in various practical scenarios such as indoor and outdoor with and without LMS algorithm. The performance of the LMS-based algorithm is demonstrated by performance parameters such as savings in the number of transmissions, power consumption, and extension in the battery lifetime.

Rest of the paper is organized as follows. Section II briefs about the LMS algorithm. Section III describes the system setup and various other attributes of the implemented node. Measurement results are presented in Section IV. Finally, the paper is concluded in Section V.

II. LMS based Reduction in Data Transmissions

This section first describes the classical LMS based prediction algorithm while the later part of this section presents the scheme which reduces the number of data-transmissions using this LMS based prediction algorithm.

A. LMS based prediction algorithm

Fig. 1 shows the structure of LMS based prediction filter [3]. If we denote the actual reading at the n^{th} instant by $x[n]$, then the N-tap filter tries to predict the value of this reading using the previous N signal values by

$$\hat{x}[n] = \mathbf{w}_{n-1}^T \mathbf{x}_{n-1} \tag{1}$$

where $\mathbf{w}_{n-1} = [w[1] \dots w[N]]^T$ is the weight vector of the linear filter and $\mathbf{x}_{n-1} = [x[n-1] \dots x[n-N]]^T$. The error in the prediction of the actual value $x[n]$ using $\mathbf{x}_{n-1}$ is given by

$$e[n] = x[n] - \hat{x}[n] \tag{2}$$

If the $e[n]$ is within error threshold, the new observation vector is given as

$$\mathbf{x}_n = [\hat{x}[n]\ x[n-1] \dots x[n-(N-1)]]^T \tag{3}$$

else

$$\mathbf{x}_n = [x[n]\ x[n-1] \dots x[n-(N-1)]]^T \tag{4}$$

The filter coefficients are adapted using the rule

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu \cdot e[n] \cdot \mathbf{x}_n, \tag{5}$$

where μ represents the step size of the prediction filter. The weights are updated only when the predicted value is not within the error threshold. The parameter μ needs to satisfy the following constraint

$$0 \leq \mu \leq \frac{1}{E_x}, \tag{6}$$

where E_x is the mean power of the signal given by

$$E_x = \frac{1}{K} \sum_{n=1}^{K} |x[n]|^2. \tag{7}$$

Here K is the total number of temperature samples used to calculate the energy of the signal and then μ [3], [10]. The parameter μ tunes the speed of convergence of the algorithm. A larger value of μ would result in improving convergence speed but may result in the filter becoming unstable, whereas, smaller μ results in smaller steady-state error but decreases the convergence speed [11]. So μ should be chosen appropriately based on the application scenario.

B. Prediction based reduction in data transmissions

Let us consider a stream of sensor data $\{x[n]\}$ that needs to be sent to a sink node, which is a centralized node. The sink node has more resources than the end devices and is capable of gathering, processing and forwarding the data to the cloud. Instead of sending the sensor data each time, in the prediction based scheme, we run a prediction algorithm at both the sensor node and the sink node and define an error budget e_t so that the values predicted at the sink node are close approximation

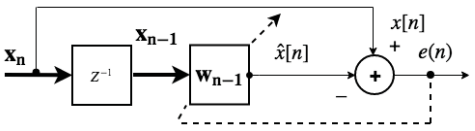

Fig. 1: LMS-based prediction filter structure

to the original sensed values $x[n]$ by a margin of $\pm e_t$. The error budget e_t is a user-defined parameter based on the selected application. The LMS based prediction described in the above subsection can then be used in the reduction of data-transmissions from an individual sensor to the sink node in the following way:

1) Based on the current observation vector $\mathbf{x}_{n-1}$, the sensor as well as the sink node predict the upcoming data $x[n]$ using (1) while the error $e[n]$ is computed using (2).
2) If the error $e[n]$ is greater than the given error threshold e_t, then the node reports the reading to the sink node, and the sink simply updates its filter weight coefficients for the upcoming prediction.
3) If the error is less than e_t, sensor node decides not to send the data to the sink and updates the reading using predicted value in place of the real sensor reading. This way, the communication between the sensor node and the sink is reduced.

III. Measurement Campaign

This section describes the sensor deployment, network topology, implementation of sensor hardware and process of data collection.

A. IoT Network Deployment

The measurement setup for this paper was deployed in our lab from 13^{th} to 16^{th} June 2018. Fig. 2 shows the deployment of four sensor nodes and a sink node to demonstrate the usefulness of the LMS based algorithm for reduction in data-transmissions from sensor nodes to the sink node. It mainly consists of two pairs of wireless sensor nodes, where each pair consists of an LMS-based node and a non-LMS-based node and a sink node. The purpose of this deployment is to sense the temperature in the indoor and outdoor environments. The temperature of the indoor environment is affected by occupancy of the room with no control over the heating, ventilation and air conditioning (HVAC). For the outdoor environment, the nodes were placed on the outer ledge of the window.

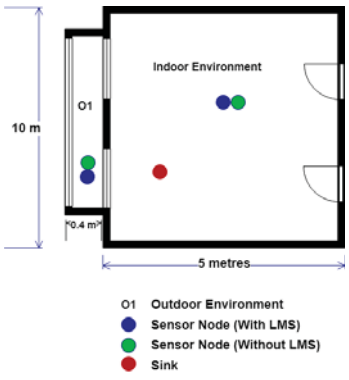

Fig. 2: Network deployment floor plan.

In this paper, IEEE 802.15.4 protocol is used to communicate to the sink node in a traditional star topology as shown in the Fig. 3. The sink node collects the data and pushes to the cloud. In our case, we are using Zigbee radio connected to desktop as a sink. Google drive services are used for the cloud storage. Currently, the stored information contains the source address, packet sequence number, temperature value and time-stamp.

B. Sensor node implementation

Fig. 4 shows the schematic of the sensor node while Fig. 5 shows the hardware design of the sensor node. The main components used for the sensor node implementation are ATmega328P microcontroller [12], DS18B20 temperature sensor [13] and XBee S2C Zigbee module [14]. The main reason for using an ATmega328P microcontroller is its easy-to-access low power consumption mode, low cost compared to other microcontrollers and high performance. DS18B20 digital thermometer temperature sensor has been used to collect the data as it provides a resolution of 0.0625°C. It is a widely used temperature sensor for HVAC applications and temperature monitoring in buildings, machinery etc. The communication module used in the sensor nodes and the sink node is the commercial RF Zigbee module (XBee S2C) manufactured by Digi International Inc. In addition to these, a standard 2800 mAh Li-ion battery has been used to power the sensor nodes.

The XBee radio module at the sensor nodes is configured as an *end device* whereas the one at the sink node is configured as the *coordinator*. The sink node is assumed to have a continuous and uninterrupted power.

C. Implementation of LMS based data reduction algorithm

The system is implemented as per the scheme discussed in the previous section. The system flow diagram is shown in Fig. 6. The sensor node initializes the LMS by calculating μ, where it uses K temperature values, and sends it to the sink. In this paper, $\mu = \frac{1}{50\,E_x}$ and

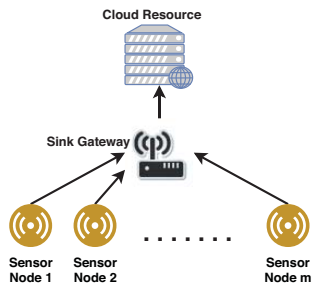

Fig. 3: Sensors use IEEE 802.15.4 communication protocol to send data to the sink node in a star topology. Sink node later pushes the data to the cloud.

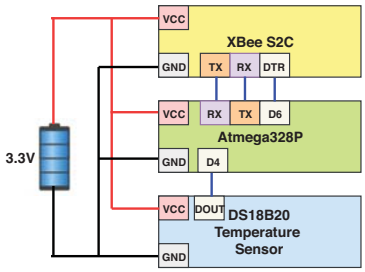

Fig. 4: Schematic of the sensor node.

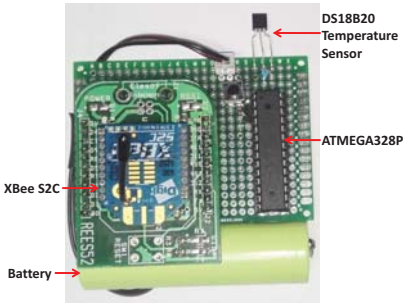

Fig. 5: Hardware design of the sensor node.

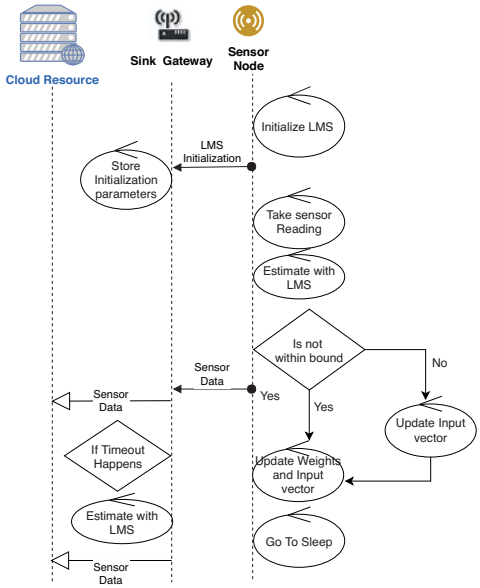

Fig. 6: System flow diagram.

$K = 50$ have been chosen based on several trial and error experiments while keeping in mind the constraints of real-time implementation. After initialization, the sensor node keeps sensing the temperature every T seconds and estimates the prediction. If it exceeds the user-defined error threshold e_t, the node sends it to the sink and updates its weights. The sink too updates its weights. If the estimate is within e_t, the node doesn't transmit. The sink waits for the data from node and if the timeout occurs, the sink estimates the value and stores it.

The sensor node becomes active for 170 ms to sense the temperature every T secs while for the rest of the time the node is programmed to be in sleep-state in low power mode to save energy. The sensing interval T can range from few secs to several minutes or hours based on the application scenario. In our paper, T is taken as 30 secs.

IV. Results

In this section, results are presented in four parts. The first part presents the results of the convergence of the LMS algorithm. The second part shows the reduction in the number of data-transmissions in various scenarios using the LMS-based algorithm. In the third part, current consumption analysis is presented for one transmission. The fourth part presents the estimated lifetime of the battery with and without LMS algorithm under various scenarios.

A. Convergence of LMS algorithm

Fig. 7 shows the number of iterations a sensor node with LMS takes to converge for different filter tap lengths 5, 9 and 13 as a function of the error threshold e_t. It can be seen that the convergence is faster for the filter with larger tap length. However, note that the computational complexity will increase with the increase in the tap length. It is also evident from the figure that, at higher values of error threshold e_t, the number of iterations to converge is less.

B. Data-transmission reduction

Fig. 8 shows the temperature values transmitted for the duration of around four days from LMS as well as non-LMS sensor nodes. For these figures, the temperature values were recorded every 30 seconds in two scenarios (indoor and outdoor) for a duration of about 4 days with 9-tap LMS filter. A total of 11,000 temperature sample point were taken for the comparison. From the figure, the first observation is that there is a significant reduction in

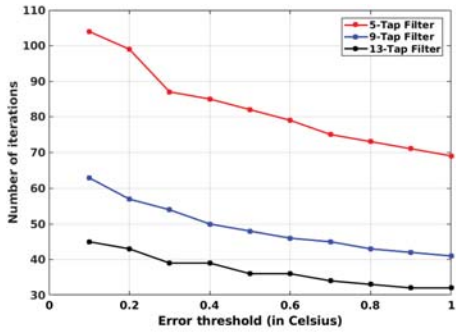

Fig. 7: Number of iterations required for convergence of LMS against varying error threshold in indoor environment. The step size $\mu = 2.768 \times 10^{-5}$.

data-transmissions with LMS based algorithm in both the scenarios as compared to the non-LMS case. The second observation is that the LMS is able to correctly predict the slowly-varying quantity as compared to a quantity changing dynamically. Table I shows the comparison of number of data packet transmissions in the two scenarios.

C. Current consumption analysis

The sensor node operates in four states: *Processing*, *Radio ON*, *Transmit* and *Sleep*. In *Processing* state, the temperature sensing and computation operations are considered. In the *Radio ON* state, we consider the current consumed when the radio is ON, just before the transmission. *Transmit* is when the transmission actually happens. In the *Sleep* state, the MCU goes to the power down mode and the radio module goes to sleep.

Table II summarizes the current consumption of the sensor node in different states of operation while Fig. 9 presents the measured current consumption profile with its different states of sensor node operation. The current has been measured using Agilent 34461A $6\frac{1}{2}$ digit, digital multimeter [15]. We have analyzed the approximate time duration using the BenchVue software by Keysight Technologies [16] to log readings of the digital multimeter with timestamps. In the first state of processing, which lasts for 130 ms, the sensor node first senses the temperature and runs the prediction algorithm. During this state, the radio module is OFF and therefore, the current consumption is only due to the microcontroller and the temperature sensor. This current is called I_p which is averaged out to 4.9 mA. The temperature sensor consumes very less current as compared to the microcontroller (0.7 mA) and is considered in the I_p itself. If the prediction values are not matched, the node goes into the second state of transmitting, where the radio module is turned ON and the temperature value is transmitted. Current consumption in this stage is 34.8 mA, as the XBee end device is configured to transmit at the highest power level (+8 dBm power gain). The radio coverage indoors is about 60 m [14] when transmitting at this power level. At this stage, the microcontroller, the sensor and radio module consume current, given as I_r, which is 34.8 mA (averaged) and lasts for approximately 30 ms. In the third state, the packet is transmitted with I_{t_x} of 45 mA where the duration is 10 ms. After this, in the last state, the radio module goes to sleep, and the microcontroller enters the power down mode. The current consumed at this stage, I_s is 6.1 μA and this stage lasts for 29.830 s.

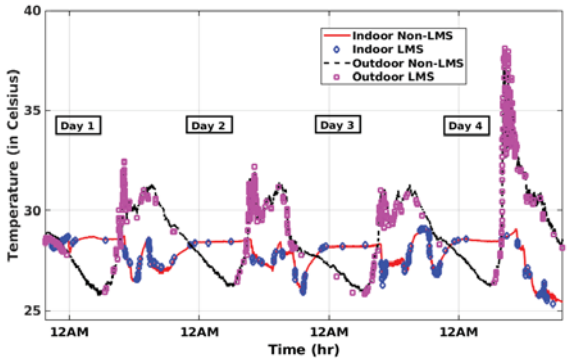

Fig. 8: Transmission instances of the four sensor nodes. The error threshold $e_t = 0.5\,^{\circ}$C, and number of taps $N = 9$ while step size $\mu = 2.768 \times 10^{-6}$ for indoor and $\mu = 2.77 \times 10^{-6}$ for outdoor.

D. Estimation of battery lifetime

The average current I_{avg} consumed by the node in an hour can be given as follows

$$I_{avg} = \frac{(I_p\, t_p + I_r\, t_r + I_{t_x}\, t_{t_x} + I_s\, t_s)\, L}{3600}, \qquad (8)$$

where L = Total number of transmissions events in 1 hr. Then, the expected lifetime of the battery in days (γ) [17] can be given as follows:

$$\gamma = \frac{Q}{24\, I_{avg}}, \qquad (9)$$

where Q is the capacity of the battery in mAh. The average current consumption and the expected lifetime of the node have been calculated as given in [17] and [18]. The I_{avg} is calculated for each case first when using and second, when not using the data-reduction algorithm. When a packet is transmitted, all the four current states are considered in I_{avg} calculation as in (8), whereas only I_p and I_s is considered when not transmitting.

TABLE I: Comparison of number of data-transmissions in the two scenarios of with and without LMS.

	Without LMS	With LMS	% savings
Indoor Scenario			
number of data-transmissions	11000	408	96.29
Outdoor Scenario			
number of data-transmissions	11000	682	93.80

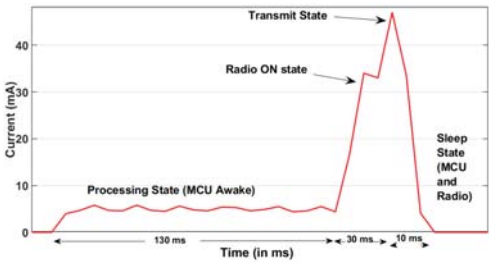

Fig. 9: Measured current consumption profile for one transmission event in interval of $T = 30$ s with payload of 11 bytes.

TABLE II: Average current consumption for different states for payload (11 bytes in our case).

State	Avg. Current	Duration (ms)
Processing	4.9 mA	130
Radio ON	34.8 mA	30
Transmit	45 mA	10
Sleep	6.1 μA	29830

TABLE III: Comparison of average current consumption and expected lifetime of a node in the two scenarios of with and without LMS.

	Without LMS (mA)	With LMS (mA)	% saving
Indoor Scenario			
Avg. Current	0.077	0.02901	62.3
Avg. Lifetime (days)	1513	4003	
Outdoor Scenario			
Avg. Current	0.077	0.0304	60
Avg. Lifetime (days)	1513	3839	

Once the I_{avg} is calculated, the expected lifetime of the node in both the cases can be calculated using (9). Table III summarizes the average current consumption in both the cases and also gives an estimate of the expected lifetime of the sensor node. Therefore, the battery backup time is approximately increased by 2.64 times in the indoor environment and 2.53 times in case of outdoor environment. Note that because of power consumption during different stages, savings in the number of transmissions (as shown in Table I) do not proportionately reflect in the extension of the battery lifetime (as shown in Table III).

V. CONCLUSIONS

In this paper, data-transmission reduction algorithm using LMS has been implemented on customized wireless-sensor node deployed in a star topology. The measurements have been carried out for different scenarios of ambient temperature. The proposed implementation gives excellent savings in the number of data-transmissions as well as significant improvement in the battery life of the sensor node. The measured data shows that the overall current consumption of the system has been reduced by 62.3% in the indoor scenario and 60% in the outdoor scenario, hence the battery lifetime has increased 2.64 times and 2.53 times respectively. These results propose the utility of the proposed implementation strategy in a wide spectrum of the IoT-enabled applications such as fire detection alarm systems, smart home monitoring, healthcare monitoring etc.

VI. ACKNOWLEDGEMENT

The authors are thankful to Center for International Mobility (CIMO grant no. Intia-1-2016-03) and Aalto University, Finland, for their financial support. The authors also thank Niranjan Reddy, Jagannath Dhar and Mahesh Murthy from Building Science Research Center, IIIT-Hyderabad for the technical discussions and access to lab instruments.

REFERENCES

[1] "Internet of Things: Wireless sensor networks," Nov 2014. [Online]. Available: http://www.iec.ch/whitepaper/pdf/iecWP-internetofthings-LR-en.pdf

[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[3] S. Santini and K. Romer, "An adaptive strategy for quality-based data reduction in wireless sensor networks," in *Proceedings of the 3rd Int. Conf. Networked Sensing Systems (INSS)*, 2006, pp. 29–36.

[4] G. Dias, B. Bellalta, and S. Oechsner, "Using data prediction techniques to reduce data transmissions in the IoT," in *IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016, pp. 331–335.

[5] F. Aderohunmu, G. Paci, D. Brunelli, J. Deng, and L. Benini, "Prolonging the lifetime of wireless sensor networks using light-weight forecasting algorithms," in *IEEE 8th Int. Conf. Intelligent Sensors, Sensor Networks and Information Processing*, 2013, pp. 461–466.

[6] F. Aderohunmu, G. Paci, D. Brunelli, J. Deng, L. Benini, and M. Purvis, "An application-specific forecasting algorithm for extending WSN lifetime," in *IEEE Int. Conf. Distributed Computing in Sensor Systems (DCOSS)*, 2013, pp. 374–381.

[7] Y. Fathy, P. Barnaghi, and R. Tafazolli, "An adaptive method for data reduction in the internet of things," in *Proceedings of IEEE 4th World Forum on Internet of Things*. IEEE, 2018.

[8] C. Debono and N. Borg, "The implementation of an adaptive data reduction technique for wireless sensor networks," in *IEEE Int. Symp. Signal Process. and Inform. Techn.*, Dec 2008, pp. 402–406.

[9] A. Jain, E. Chang, and Y. Wang, "Adaptive stream resource management using Kalman filters," in *Proceedings of the ACM SIGMOD Int. Conf. Management of Data*, 2004, pp. 11–22.

[10] S. Haykin, *Adaptive filter theory*. Pearson Education India, 2008.

[11] "Choosing a step size (adaptive filter toolkit)," Jun 2008. [Online]. Available: http://zone.ni.com/reference/en-XX/help/372357A-01/lvaftconcepts/aft_choose_stepsize/

[12] "ATmega328/P datasheet," Nov 2017. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf

[13] "DS18B20, programmable resolution 1-wire digital thermometer," 2015. [Online]. Available: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[14] "XBee/XBee-PRO S2C 802.15.4 radio frequency (RF) module," Feb 2018. [Online]. Available: https://www.digi.com/resources/documentation/digidocs/pdfs/90001500.pdf

[15] "Keysight Technologies Digital Multimeters," May 2018. [Online]. Available: https://literature.cdn.keysight.com/litweb/pdf/5991-1983EN.pdf

[16] "Keysight technologies Benchvue software 2017 (bv0000a)," March 2017. [Online]. Available: http://literature.cdn.keysight.com/litweb/pdf/5991-3850EN.pdf

[17] G. Bag, Z. Pang, M. Johansson, X. Min, and S. Zhu, "Engineering friendly tool to estimate battery life of a wireless sensor node," *Journal of Industrial Information Integration*, vol. 4, pp. 8–14, 2016.

[18] F. Wu, C. Tan, M. Sarvi, C. Rudiger, and M. Yuce, "Design and implementation of a low-power wireless sensor network platform based on XBee," in *IEEE 85th Vehicular Technol. Conf. (VTC Spring)*, 2017, pp. 1–5.