International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 4, August 2020, pp. 3623~3634
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i4.pp3623-3634

A data estimation for failing nodes using fuzzy logic with integrated microcontroller in wireless sensor networks

Saad Al-Azzam[1], Ahmad Sharieh[2], Saleh Al-Sharaeh[3], Nedaa Azzam[4]
[1,2,3]Computer Science Department, The University of Jordan, Jordan
[4]Department of Computer Science, Imam Adbdulrahman Bin Faisal University, Saudi Arabia

Article Info

Article history:

Received Apr 18, 2019
Revised Dec 29, 2019
Accepted Jan 11, 2020

Keywords:

Failing sensor
Fuzzy logic
Microcontroller
Node replacement
Wireless sensor networks

ABSTRACT

Continuous data transmission in wireless sensor networks (WSNs) is one of the most important characteristics which makes sensors prone to failure. A backup strategy needs to co-exist with the infrastructure of the network to assure that no data is missing. The proposed system relies on a backup strategy of building a history file that stores all collected data from these nodes. This file is used later on by fuzzy logic to estimate missing data in case of failure. An easily programmable microcontroller unit is equipped with a data storage mechanism used as cost worthy storage media for these data. An error in estimation is calculated constantly and used for updating a reference "optimal table" that is used in the estimation of missing data. The error values also assure that the system doesn't go into an incremental error state. This paper presents a system integrated of optimal data table, microcontroller, and fuzzy logic to estimate missing data of failing sensors. The adapted approach is guided by the minimum error calculated from previously collected data. Experimental findings show that the system has great potentials of continuing to function with a failing node, with very low processing capabilities and storage requirements.

Corresponding Author:

Saad Al-Azzam,
Computer Science Department,
The University of Jordan,
Queen Rania Street, Amman, Jordan.
Email: sa3d_al3zam@hotmail.com

1. INTRODUCTION

Wireless sensor networks (WSNs) have become an essential part of daily life, as a source of data and a way of controlling disperse connected smart devices [1]. Some applications that are based on WSNs could not tolerate malfunction in its nodes. Dealing with missing data that result from such failure is of great importance, and an effective data restoration strategy must be integrated with the WSN's structure to ensure that the system keeps working with required efficiency, while keeping cost to its minimum.

Nodes in a WSN work in harmony and cooperate to deliver the system with needed data, and submit them as required to a base station (or another node in the same network or another network) [2]. Due to certain issues, nodes might become unable to collect or send data. Failing nodes could cause some critical data to get lost. For example, as in systems employ sets of servers that host directly attached storage and execute user application tasks in Hadoop Distributed File System [3]. A more cost effective solution would be to activate a strategy to estimate the data it collects so that the performance of the network is not affected. In some industrial systems that rely on sensors to collect data and initiate process, having a failing node could lead to shutting down part of the system (or sometimes the entire system) until the failing nodes are fixed or replaced. So, employing an appropriate data estimation or replacement technique could help these systems avoid losing critical data, or having corrupted products.

Different data prediction and replacement strategies have been proposed in recent literature. Some depend on using the history of collected data from the failing node, like those in [4], or moving a neighboring node to do the work of this failing node [5], or changing the route transmitted messages using a 'life time aware routing protocols' [6]. Other techniques involve assigning a backup node for those nodes that are under the threat of failing [7]. This procedure, although very efficient, would be very expensive. Some of the proposed node replacement strategies in literature could lead to having the network's energy consumption higher than it is supposed to be [8], especially in dangerous situations, where replacement is not easily doable. Node replacement might not always be the most applicable solutions to these failing nodes. These nodes could be expensive types of sensors, placed in a hard-to-reach area, take too much time to reach and replace, or require a certain level of experience from workers who might not be available by the time of the failure.

The most important feature in using historical statistics of nodes' readings is that estimation errors can be minimized along the time readings submitted by these nodes. In case of a node (or connectivity) failure, estimation of that node's data can be used using a weighted prediction mechanism from the history of previously registered data. The use of a microcontroller to govern the storage and manage historical data presents a more reliable and cost effective technique for such data management. It also provides an efficient facility to calculate errors between successive history files' entries. These can be easily accessed and adopted in an artificial intelligence module to help in estimating missing data in a WSN.

Some systems with nodes have a range of values that they collect data always fall within, like steam sterilizer systems. These types of system facilitate use an intelligent technique, like fuzzy logic, efficient and worthy from different aspects: financial, operational, and energy saving. Fuzzy logic techniques are becoming widely used in decision making application especially those related to WSN based systems. They provide the ability of dealing with real-world uncertainty problems that are not based on statistical measures [9]. this technique is considered less extensive and more reliable than automatic control of wireless data management and communication. Fuzzy logic deals with missing input or unclear input in a logical manner that resembles how a human makes decisions. The obvious advantage of using fuzzy logic is the speed of processing of fuzzy input and producing decisions. Fuzzy logic relies on having learning rules [10], according to which an inference engine makes decisions regarding the problem at hand. The input to these fuzzy sets is in natural language (not discrete values). The output can either be crisp or in natural language [11], along with confidence levels of how "appropriate" the produced output is to the system's application.

The proposed system in this study works over two main stages. The first stage is building a historical data archive that stores every node's readings over several time steps (TS) and the time stamp of that reading. The sterilizer works on treatment of material by activating the treatment device when treatment is required on timely basis. Each of these treatment phases is called a unit's treatment run. The second stage involves prediction of lost data when a node fails. It estimates lost data based on the output of the fuzzy logic inference engine that is fed with the historical data and the fuzzy rules to build the estimation. A dedicated microcontroller will be in charge of monitoring the nodes and detecting failing nodes, and then activates the data restoration by prediction technique. The microcontroller is also responsible for archiving and labeling history data files. It would facilitate fetching the correct file along with calculating estimation error to help the system use the appropriate rule base for the fuzzy logic system to generate the estimation. This would make the system more efficient in terms of processing and estimation speed, functionality, and adaptability with dealing with various failing nodes scenarios, and of course having a cost effective technique for nodes replacement and data recovery.

The rest of the paper will present a brief review of previous studies related to restoring data from failing nodes in WSNs, in Section 2. In Section 3, an explanation of the proposed algorithm for restoring missing data from a failing from historical data using fuzzy logic is presented. Section 4 illustrates different run scenarios with various nodes' readings, and shows an analysis of the results to prove the efficiency and applicability of the proposed technique, especially in steam sterilizers. Section 5 presents conclusions and future aspects of the system.

2. PREVIOUS WORK

Historical data are used to find a pattern of collected data that can be fed to a machine learning system to provide a prediction to future data, which is very helpful in cases of node failure. Complex patterns are recognized and fed into a machine learning classifier in the work of [12] to sensor data of phone-based accelerometer for activity recognition of mobile phone users. A predictive model is built to recognize activities of users and builds a predictive model of these activities afterwards.

Using probabilistic computation that depends on historical data of nodes was adopted in the work of [13]. This used Markov Decision Processes (MDP) to develop a probabilistic approach to specify the nodes that are in need to be replaced, and collect statistical data based on past behavior of the nodes in the network. Then, these data are used to determine a suitable policy for node replacement by finding an estimate of the long run cost of replacing the node (or the energy source). This method could be inefficient to reach node, and the computations are done repeatedly which could waste computational resources. In the work of [14], intrusion detection model in a WSN has been developed based on Markovian Intrusion Detection System (M-IDS) that predicts what nodes are prone to attacks, and predict anomaly and misuse behavior of nodes by integrating game theory and MDP. This will help in deciding what the best defense strategy to apply. Anode failure avoidance strategy requires constant monitoring, and could mean large amount of stored data to be kept for very long periods of time.

Fuzzy logic was used to predict which node is the most suitable node to be a cluster head in WSN [15, 16]. Historical data of all nodes including rate of recurrent communication, residual power, degree of neighboring nodes, and distance to base station are tracked and input into the fuzzy logic decision system. The decision system calculates the probability of a node to be a cluster head, based on the history of communication between the node and the base station. Yet the data that is collected from these nodes (cluster heads and other nodes) are not put into focus. The data itself could be lost during the cluster head assigning process; and no backup of the data itself is presented.The use of fuzzy logic in estimation of missing data based on historical data has been tackled and proven to be efficient [17, 18]. The historical data that was collected when the system was working with perfect operation work is the fuzzy input to the inference engine. These data are weighed and processed using the rules in the rule base to produce multiple estimations, with confidence levels of how efficient these data will be to make the system work.

Very limited number of recent research was found that employs Artificial Intelligence (AI) in estimation of data in failing node. However; predicting factors and levels of some natural elements, like the work of [19] was implemented. In their research, the history of weather data and remote sensing images was fed into an ensemble estimation system to predict the occurrence of forest fires. They did not mention a recovery strategy for failing data collection methods, which is very likely to happen in case of forest fires. Energy is a concern in systems that employ WSN within its structure. This made preserving nodes' energy and minimizing its consumption in developing data estimation strategies the subject of many researches [20]. Most of the aforementioned studies focus on one aspect of data estimation in failing nodes. Sometimes the data itself was not of importance to some studies. The historical files are not presented with an easy to retrieve and re-use structure in most of the researches. Energy and cost effective procedures were only taken from the failing node's aspect only, not the entire network's performance.

Most of the researches mentioned above made attempts to solve the node failure problem from one aspect; either by predicting data from lost node (like the work in [6]) or finding "the most suitable node" to replace the failing node (as in the work in [21]). The work of [22] focused on handling nodes in a WSN that is opted for weather forecast. They proposed a greedy algorithm to construct a barrier in heterogeneous WSN where mobile nodes are used to construct a barrier when the weather changes. In this work, both solutions are consideredto make the algorithm adabtable by almost all applications of WSNs.This is true for real-time data harvesting and monitoring (as in traffic control) and for data based WSNs (as in applications of monitoring climate changes in some area).

3. ESTIMATING MISSING DATA OF FAILED SENSORS

The technique presented in this research is based on getting readings from sets of sensors that are distributed in a steam sterilizing unit. The collected data are sent wirelessly to the base station (the control station), managed and stored through a dedicated microcontroller. A historical data file is built and labelled each time the unit runs for a treatment process. The data are sent over several time steps per run, with a t minutes' period, such as 5 minutes, for each time step. A fuzzy logic module is activated in case one (or more) sensor node failed to estimate the missing data. A representation of system's structure and components are shown in Figure 1.

In Figure 1, there are four main types of sensors that read data around a sterilizing unit over several time steps. These are read temperature, humidity, luminous (LUX), and voltage values. These values are sent to the base station (control room) where they are monitored and tuned as needed by the microcontroller. When a node fails, the fuzzy logic module is activated. It reads the latest optimal table's entries stored in the memory unit of the microcontroller to begin the estimation process, and continue working with the estimated values.

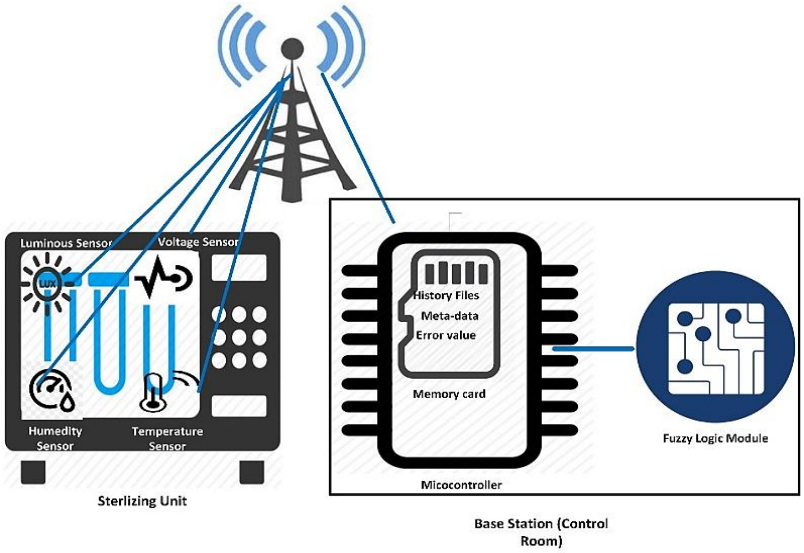

Figure 1. Historical-fuzzy data estimation system's structure with main components

Voltage sensors are used to monitor system power consumption to calculate treatment cost in terms of energy and computed locally in the base station using (1).

power_consumption = voltage * current (1)

Once the sterilizing unit starts functioning in the first run, a primary file with the first run's reading is initiated. The microcontroller registers these data and "meta data" in a folder labeled with the runs' number of that reading. For each of the sensors, this table is called the "optimal table" since there are no previous data to be compared to. This process is summarized in the text box in Figure 2.

```
Input: Sensors Readings
Output: Optimal Table with Estimated Values
Begin
       1. store sensors' data in history file
       2. build meta-data file and add run's readings
       3. build optimal table using first run's data
```

Figure 2. Algorithm for the first treatment run

On the following treatment runs, the received data from the nodes are accumulated in the tables. These values are compared with those in the optimal table. The data error between the data in the optimal table and the new table is calculated according to the equation: $W_d = {}^{1}/_{R}$, where W_d is the data weight of the values in the optimal table and R is the run's number. The data in the optimal table are updated after each run to get new weights. The new data N_d is calculated using (2).

$$N_d = \big(1 + (p \times s)\big) \times ((D_r \times W_d) + (D_o \times W_o)) \quad (2)$$

Where p is a parameter set by user, ranging [0.0 - 1.0], and s is a signal parameter that has a binary value of 1 or -1, depending on user preference.$D_r$is the actual data read from the sensor at run r. D_o is the old data and $W_o$is the weight read from the optimal table. The value of s is set to +1 if the acquired value is greater than the one in the optimal table and set to -1 if it less. This parameter ensures that the estimation system doesn't keep incrementing constantly. The value of p is set by the human controller to ensure smoothness of the alteration of estimated values. In other words, it indicates how fast the transition does from one value to the other changes. The value of p could be altered later on to use an AI mechanism for

automatic setting. The p value is balanced based on the submitted value (greater than or equal to 0.5 or less than 0.5).

The choice of the p value provides an indication of how much change does the user want in the estimated data. If p value is high (like 0.8), the system will respond to changes faster than if p value is small (like 0.2). Choosing p value depends on application change. If the read signal is a stable but noisy, p value must be small, or otherwise; the system will be unstable as it will respond to noise.

The data in the optimal table is updated after every run according to (2). This implies that the estimation error will be reduced when more data is acquired from the system. The first run probably has the highest error since there is no previous data to compare it to. This error value is stored separately, and added to the meta-data file. Yet, it is very unlikely for a node to fail on the first run of the system (this means the system is very unreliable!). Figure 3 shows designated steps of the algorithm to compute the averaged errors after the first run.

```
Input: Sensors Readings, Optimal Table
Output: Updated Optimal Table, Averaged Error
Condition: r> 1
   1. add sensors' data of run R to history file
   2. update meta-data file and add all sensor's readings
   3. update optimal table using (3)
   4. Compute Error value for every sensor    between R̊ and Rr
   5. average errors from all sensors' data in run r
   6. store averaged error in meta-data file
```

Figure 3. Algorithm for computing avereged errors and updating optimal table

A segment of the optimal table is shown in Table 1. It presents two sensors: one for temperature (Temp_sensor_1) and another for humidity (Hum_sensor_1), each with the collected data over 3 time steps in one of the treatment runs. Once the table is built and accumulation of data is used to update it after every run, it is stored on the microcontroller's memory card to be used for estimation in case of node failure. These data are retrieved from the memory when a node fails. The appropriate data file is fetched from the memory by scanning the meta-data file and get directly to the folder where the sensor's data and runs' data are stored. This makes the fetching process faster and more efficient, and cut-out the time needed by the fuzzy logic module in predicting missing data.

Table 1. Part of registered data from the optimal table constructed for a sterilizing run over different time steps (TS)

Time Steps/Sensors	TS0	TS1	TS2	TS3
Temp_sensor_1 (in Celsius Degrees)	50 °C	50.2°C	50.3°C	50.6°C
Hum_sensor_1 (in percentage)	42.2%	42.1%	42.0%	41.6%

The designated data are unified through a membership function. This is to define how each of the members' values in the input can be mapped to a degree of membership. The membership puts these values in the interval [0,1] in the fuzzification process before using them by the inference engine. The easiest and most direct normalization process is to assume that the lowest value in the optimal table (per sensor) is 0, and the highest is 1, and map each of the remaining values to the scale between them. Equation (3) is used in the normalization process.

$$N = (x - min) \times \frac{1}{(\max - \min)} \quad (3)$$

Where N is the new normalized value and x is the value to be normalized, max and min are the maximum and minimum values in the table, respectively. The resulting values are placed in a new temporary table. These values are then fed into the inference engine to produce predicted data for the failing node. The data in the optimal table and the data from the last run (or any run determined by the human controller, since some runs have similar conditions that other treatment runs don't) are used to produce the estimation. The data estimation process of failing nodesis summarized in the Figure 4.

```
Input: optimal table, average of errors
Output: error, new-error
1. Generate an estimation using Fuzzy Logic
  1.1. input optimal table & average of errors
     1.2. normalize each data in the optimal table, add error as a weight
   for each value
  1.3. compute estimated value for failing node.
2. compute error between optimal table and estimated value:
3.If (error <error_threshold)
   Continue treatment with estimated value
  Else
       Compute new error between estimated and previous runs of the failing
    node, get least error value.
3.1. if (new_error<error_threshold)
     Continue treatment with data from history file
    Else
     Shutdown system and discard treated material
```

Figure 4. Data estimation process of failing nodes

It is worth mentioning again that the estimation error was reduced in the optimal table's building process. This error is further reduced through the rules used by the rule base used by the inference engine. The highest error will exist if the node fails at the first run (which is, as mentioned before, very unlikely), since the data in the optimal table is the same as the data in the current treatment run. Figure 5 shows the error values in the first run for one of the temperature sensors, and Figure 6 shows the error after four runs for the same sensor.

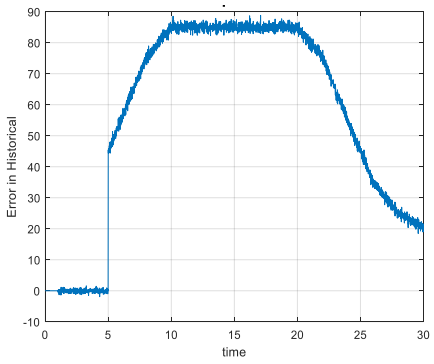

Figure 5. Error values in temperature in 1strun

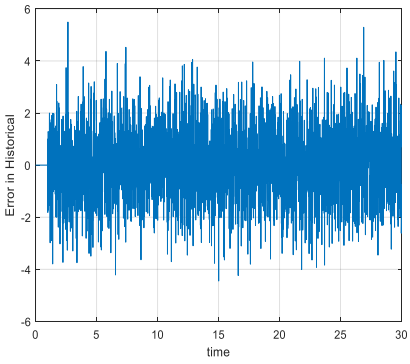

Figure 6. Error values in temperature in 4thrun

Figure 5 shows that the estimated error is 0 before minute 5 (before the failure), because there is no difference between read data and data in the optimal table. The optimal table is being filled while the system is running for the first time in this treatment. When failure happened at minute 5, the error value spiked to around 45°C, since the read value in last reading from that sensor was 45°C and there's no new data to update the optimal table from. The error continues to rise as the system continues running without new data coming from that node to update the optimal table properly. The error dropped again after minute 25, since the temperature value dropped within this treatment.

The data represented in Figure 6 shows that the error values are almost steady. Since the system has stored data about previous treatment runs, the historical estimation produces an error value of 0 for the first minute of treatment. Since data was collected from previous runs, the calculated estimated error ranges between -4°C and +5°C, which is a very good range. Failure happened at minute 5 of the run, yet the error values remained stable, and the system's performance was not affected.

After estimated errors are calculated, they are fed to the inference engine in the fuzzy logic module to be used as weight to the optimal table' values. An estimate of the actual value that could have been collected from that failed sensor is produced. The rule evaluation procedure in the fuzzy logic module

(for both rules' antecedents and consequent) is used by the fuzzy logic's inference engine to produce the output. The output represents the normalized values (not actual) of the sensor's expected readings. These normalized values are mainly used to test the results of the system before the de-fuzzification process. A simplification of the general structure of the fuzzy logic technique is shown in Figure 7.

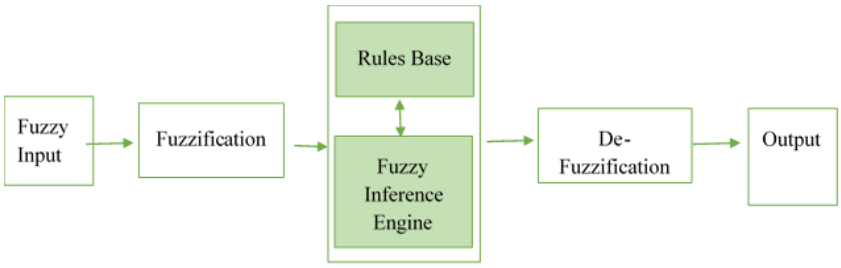

Figure 7. Fuzzy logic module basic operations

The fuzzification process begins with taking the numeric values from the optimal table, where each node sends its data item x that is in the universe of discourse X designated for that node (temperature values for example should be in the range [10°C – 90°C]). A sample of input received from some of these nodes is illustrated in Table 2.

Table 2. Sample real-time data read by the sensors for nine times

Temp_1	Temp_2	Temp_3	Hum_1	Hum_2	Hum_3	Lux_1	Lux_2	Lux_3
21.165	21.165	21.165	0.212	0.212	0.212	2.647	2.647	2.647
20.657	20.657	20.657	0.206	0.206	0.206	2.686	2.686	2.686
20.135	20.135	20.135	0.201	0.201	0.201	2.706	2.706	2.706
20.442	20.442	20.442	0.204	0.204	0.204	3.897	3.897	3.897
19.423	19.423	19.423	0.193	0.193	0.193	3.215	3.215	3.215
21.846	21.846	21.846	0.217	0.217	0.217	7.399	7.399	7.399
20.239	20.239	20.239	0.201	0.201	0.201	6.084	6.084	6.084
22.007	22.007	22.007	0.219	0.219	0.219	9.541	9.541	9.541
20.504	20.504	20.504	0.203	0.203	0.203	8.373	8.373	8.373

The rule base in this fuzzy logic functions is based on "Sugeno" method [23]. A single spike of data, called a "singleton", is used as the membership function of the rule consequent. Opposed to this method is "Mamdani" method which integrates across a continuously varying function to find a centroid of a two-dimensional shape [24, 25]. For example, for fuzzy variables x, y and z that are parts of fuzzy sets A and B that belong to the universe of discourse X and Y respectively, a rule evaluation comes in the form presented in formula (4) [23]. The $f(x.y)$ is a mathematical function. In a zero-order Sugeno fuzzy model, the function is evaluated to a constant k whose value determines the value of the singleton resulting from the inference.

$$if\ x\ is\ A\ AND\ y\ is\ B\ \rightarrow z\ is\ f(x.y) \quad (4)$$

For three sensors in Figure 1, for example, there are nine rules used by the fuzzy logic module, one for each sensor (Except Voltage sensors). These rules have been evaluated to confidence levels based on the error calculated when building the optimal table. A sample of the rules with associated confidence levels are illustrated in Table 3. Inference of output using the rule base adopts the "Sugeno" inference method [23]. The rule evaluation process is illustrated in Figure 8.

Table 3. Fuzzy rules mapped to historical data retrieved from optimal table, where L for Low, M for Medium, and H for High

Rule#	1	2	3	4	5	6	7	8	9
Historical Confidence	L	M	H	L	M	H	L	M	H

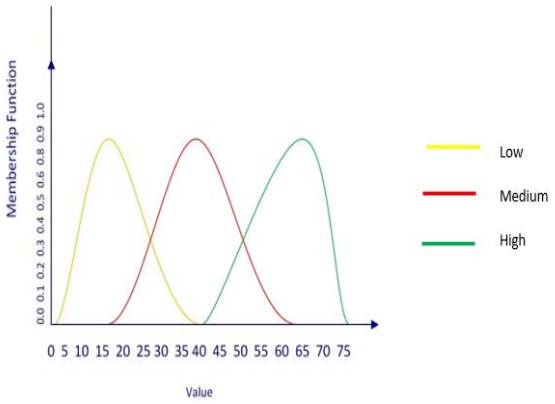

Figure 8. Sugeno rules evaluation process

One of the mostly used de-fuzzification functions used with Sugeno method is the "center of gravity for singleton" as shown in (5).

$$U = \frac{\sum_{i=1}^{P}[u_i \mu_i]}{\sum_{i=1}^{P}[\mu_i]} \quad (4)$$

Where U is the crisp output, u is the output variable, μ is the membership function (inverse of the one used for fuzzification) used after accumulation of values. The de-fuzzification process is done in a similar manner shown in Figure 9, that is adapted from the work in [10], for temperature values used in the node's data prediction for example. The resulting crisp output from the de-fuzzification process is sent to the controllers to help them make a decision regarding the treatment unit's functionality and the material being treated validity.

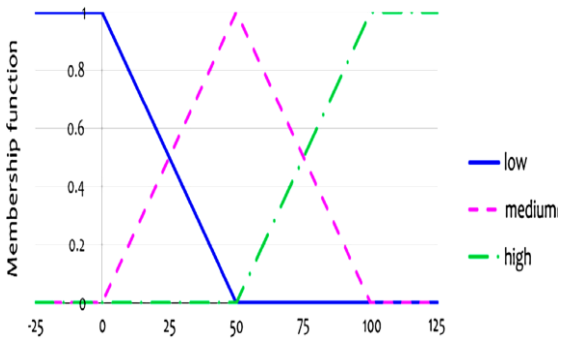

Figure 9. De-fuzzification process (Adapted from the work in [10])

4. IMPLEMENTATION AND TESTING

The system was tested in a steam sterilizer in a factory. The sensor nodes are put in a rough work environment, where steam and high and fluctuating temperature has a direct effect on these sensors' lifetime. This makes most of the working nodes prone to failing, and replacing them might not be an easy task since stopping the sterilizer in the middle of a treatment process would mean either losing the material being treated or losing time. In both cases, it means money for the factory and business owner.

Evaluation of the validity and applicability of the proposed procedure was tested in a simulation of a steam sterilization unit similar to those used in food processing establishments. All sensors in the treatment unit are connected to a base station where a human user monitors the entire system constantly, and makes decisions regarding the unit (or the treated material) if anything went wrong. Each group of sensors was connected wirelessly to a dedicated microcontroller. The microcontroller stores received data from each sensor over pre-determined time steps. It creates and updates the optimal table to be used in case one of the nodes fails and stops sending data.

The use of microcontrollers in this kind of systems has several benefits such as microcontrollers provide high performance and low processing time. They provide a good solution with low cost. On average, a microcontroller costs 1.3$ on Ebay and sensors cost around 8$. They also have the ability to connect the system to the internet and provide reliable real time readings. In addition to the aforementioned benefits, microcontrollers have the ability to transfer data between multiple controllers on the integrated memory card. This means that a system can learn from another system, assuming another sterilizing unit has better performance than the one at hand. Then, this more efficient unit's data can be easily copied from its memory card to the memory card of the unit at hand. Failing at the first run has the highest error in estimating missing data. So, this scenario was taken into consideration when testing the system. Another scenario involves measuring the error in estimating and comparing this estimated value with the actual value read by the sensor if it hasn't failed. Using Matlab package allows "peaking" at actual data that are hidden from the fuzzy logic estimation module. Figure 10 shows how the error was high in the first runs of the sterilizing unit and how it dropped to almost zero by the end of the 25th run, when a node failed.

The first run of the system is considered the main building block in establishing the optimal table. According to this run, the errors are updated and estimations are produced. The node fails at the first run has serious indications that the system in general is at high risk of failing soon. Risk avoidance procedures should take this possibility into consideration. The estimation should take such a case into consideration to help make the system running even if a node fails at first run. The optimal table is built from available readings of other sensors, or readings from pervious data under similar conditions – determined by the human controller. The calculated error for that failed node is expected to be high; yet, it can be corrected through upcoming system runs. Figures 11 and 12 show error values computed for two different sensors that failed after 5 minutes in the first run. The errors were managed as time steps went on.

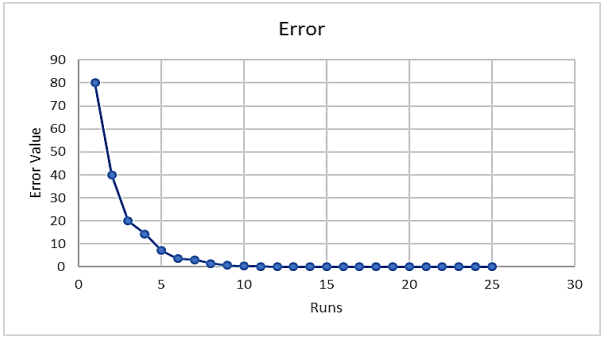

Figure 10. Error values computed by the proposed technique versus the number of runs before failure

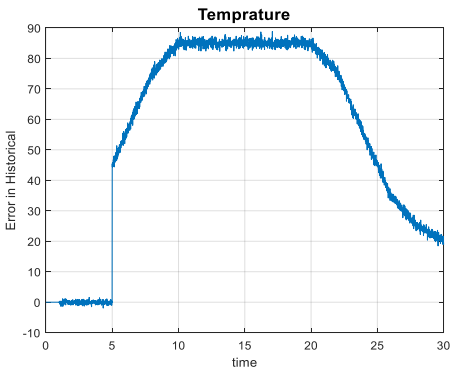

Figure 11. Error in temperature estimation after node failure in 1st run

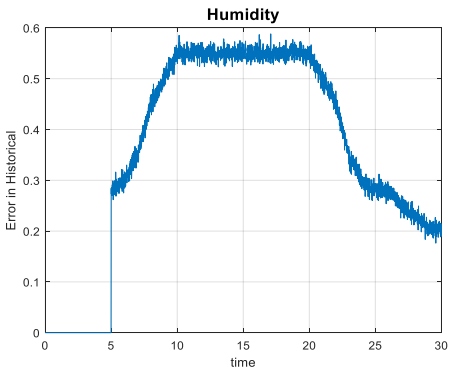

Figure 12. Error in humedity estimation after node failure in 1st run

The error (confidence level) begins at 0; the system is in the process of building the history file. When failure happened after 5 minutes of the run, the error spiked because there's no previous data to compare to and calculate the error. The error values noted in both figures are close to 0, because the actual data that would have been collected in case the node didn't fail is very close to the estimated data.

For example, the actual temperature value is almost 40° C and the actual humidity value is around 28%. The actual values were used just to show how the error is produced in case another measure (like predicting relationships between connected nodes, which is really the case in this system). The microcontroller activated the fuzzy logic module to produce estimates, which were very close to the real values. The error values are close to zero since the estimated values are very close to the real values; since

Error = Actual value –Estimated value.

In this example, the error would be: 40 – 45 = -5° C.

After several runs, the system stabilized, and continued operating with the failing nodes, but using estimated values. The error, in case of temperature by the end of the treatment time dropped to -25 (since the unit starts lowering the temperature values after 25 minutes of the treatment). Also, the humidity error dropped to 20% (since the temperature dropped). It's worth mentioning that there are some established relationships between the nodes in the sterilizer. For example, as the temperature increases, the humidity also increases and the energy consumption increases. This piece of information can be used in building the optimal table if a node fails at the first run, and can help the fuzzy logic estimation module predict values that are close to the real values.

Runs that follow a first "successful" run will have more data to work with and calculate a smaller error value. This would ultimately produce closer estimated values to real data. At the fourth run of the system, for example, the optimal table will have real data that the microcontroller will use in finding the error margin before using the fuzzy logic estimation module. These data will be closer to real collected data if the node did not fail. The system will have lower risk of overall failure, and the need for a unit shutdown can be avoided altogether and node replacement could be postponed without critical danger on the system. Figures 13, 14, and 15 show the estimated values produced by the system against the real values that were actually read by the sensors. MATLAB allows developers "peak" at real-data, and hide them from the fuzzy logic's module. The data presented in the graphs are for the same sensors at different time steps in the fourth run of the sterilizing unit.

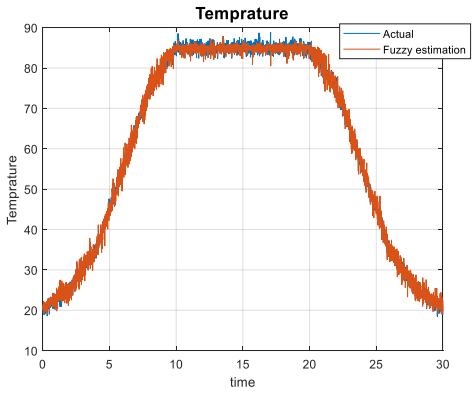

Figure 13. Actual temperature (blue) vs. estimated temperature (red) in 4thrun

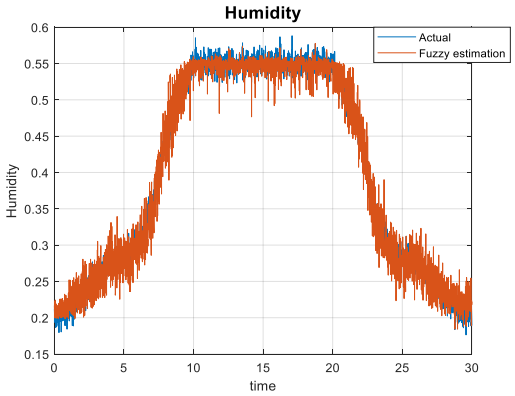

Figure 14. Actual humidity (blue) vs. estimated humidity(red) in 4thrun

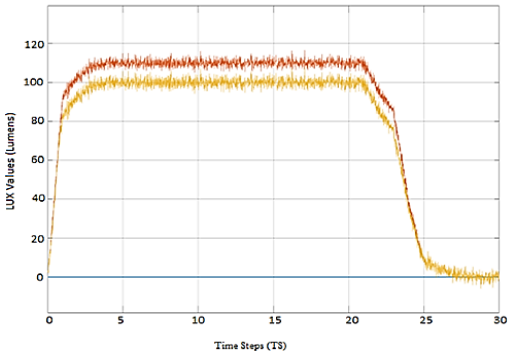

Figure 15. Actual lux (red) vs. estimated lux (yellow) in 4thrun

The produced errors are very low as shown by the charts and are compared to those for the first run. It produces better estimations to real data. Figures 16 and 17 show the errors calculated for temperature and humidity sensors in the fourth run over 30 time steps of the sterilizing unit. Note that failure happened after 5 minutes from starting the sterilization process.

The error rose to high values after the node failed to send data after 5 minutes in both the temperature and the humidity sensors. Nevertheless; these error values are considered very small when compared to those produced at the first run. The errors in temperature ranged between -4 and +4 °C, and the errors in humidity were below0.07 %.Adding to that, as the system's reading are stable, the errors in temperature dropped to almost zero, but the humidity peaked since the values of humidity keep rising in the system, and the historical records stored in the optimal table won't be suitable. This is when the fuzzy logic estimation is used to produce better estimations.

The experimental results show that the use of historical files in estimating data of failing nodes produces accurate predictions through the employment of fuzzy logic. When the errors are calculated every time, new history file makes update of the optimal tables that is used in prediction tuned and updated to produce even more accurate predictions. The microcontroller unit governs and manages data stored in its embedded memory card. The data received from the nodes provides a cheap, efficient, and reliable solution to manage and control a WSN with failing nodes.To the extent of our knowledge, there has been no research presented that implemented a microcontroller for storage and management of data received from sensors, and connected to a fuzzy logic module to predict missing data in case of a sensor's failure. The retrieved results are promising, and provide an affordable and reliable solution to WSN based systems.

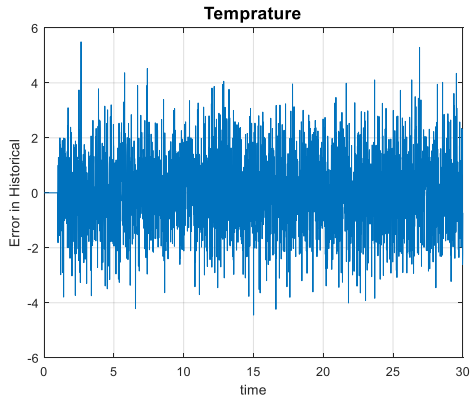

Figure 16. Error values of temperature sensor in the 4thrun

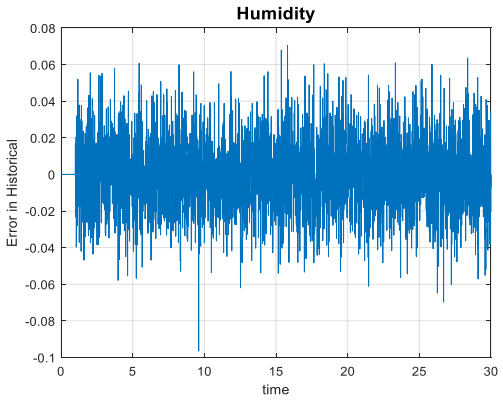

Figure 17. Error values of humidity sensor in the 4thrun

5. CONCLUSIONS

The world is moving towards adopting WSNs in countless applications, due to their efficiency and low cost. Having a failure in one of the nodes in wireless sensor network (WSN) might lead to failure in significant part of it, and thus lower the efficiency and not reaching the desired purpose of the network. Several techniques were implemented and studied to come over the problem of node failure and loss of data in WSN based applications. In this research, a technique using fuzzy logic for estimating missing data of failed nodes is introduced. It was tested and applied to sterilizing units that has sets of different sensors all connected through a WSN, where a data estimation strategy was used in case of a failing node.

By depending on the data estimation technique that integrated a microcontroller and a fuzzy logic estimation module, the dilemma of increment in error is avoided. The errors between the estimated values and the values of the actual collected data were computed constantly, especially that the sterilizer is a closed environment where sensors have an effect on each other. For example, if humidity increases, both temperature and lux are affected, and thus the calculated error depends on active sensors always reflects the error in the failed sensor. This method of averaging errors provides confidence levels towards the estimated values. This confidence indicator can be used by decision maker (the human controller) of either to continue working with the current treatment, find a new table of historical data to compare results, or to shut down the system. Yet, testing the system shows high confidence values. For future research, another artificial intelligence mechanism could be employed to "learn" from the behavior of the system. The learnt pattern of system's behaviors can be used for automatic tuning of parameters and other settings determined by the human controller.

REFERENCES

[1] Yu, R., *et al.,* "Cognitive Radio Based Hierarchical Communications Infrastructure for Smart Grid," *IEEE Network*, vol. 25, no. 5, 2011.

[2] Zhu, C., Chunlin Z., Lei S., and Guangjie H., "A Survey on Coverage and Connectivity Issues in Wireless Sensor Networks," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 619-632, 2012.

[3] Shvachko, K., Hairong, K., Sanjay, R., and Robert, C., "The Hadoop Distributed File System," *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies*, pp. 1-10, 2010.

[4] Younis M, Lee. S, and Abbasi A.A., "A Localized Algorithm for Restoring Internode Connectivity in Networks of Moveable Sensors," *IEEE Transactions on Computers*, vol. 59, no. 12, pp. 1669-1682, 2010.

[5] Du. J., *et al.,* "Application-Oriented Fault Detection and Recovery Algorithm for Wireless Sensor and Actor Networks," *International Journal of Distributed Sensor Networks*, vol. 8, no. 10, pp. 1-9, 2012.

[6] Nair, P.K., And Nath S.S., "Replacement of Deactivated Sensor Nodes in Wireless Sensor Networks," *International Journal of Advanced Computational Engineering and Networking*, vol. 2, no. 5, 2014.

[7] Liu, H., Nayak, A., and Stojmenović, I., "Fault-Tolerant Algorithms/Protocols in Wireless Sensor Networks," *Guide to Wireless Sensor Networks*, pp. 261-291, 2009.

[8] Younis M., and Kemal A., "Strategies and Techniques for Node Placement in Wireless Sensor Networks: A Survey," *Ad Hoc Networks*, vol. 6, no. 4, pp. 621-655, 2008.

[9] Bellman, R. E., and Zadeh, L. A., "Decision-Making in a Fuzzy Environment," *Management Science*, vol. 17, no. 4, 1970.

[10] Maksimović M., Vujović V., and Milošević V., "Fuzzy Logic and Wireless Sensor Networks-A Survey," *Journal of Intelligent & Fuzzy Systems*, vol. 27, no. 2, pp. 877-890, 2014.

[11] Kapitanova K., Son S.H., and Kang K.D., "Using Fuzzy Logic for Robust Event Detection in Wireless Sensor Networks," *Ad Hoc Networks*, vol. 10, no. 4, pp. 709-722, 2012.

[12] Kwapisz J.R., Weiss G.M., and Moore S.A., "Activity Recognition using Cell Phone Accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74-82, 2011.

[13] Misra S., Mohan S. R., and Choudhuri R., "A Probabilistic Approach to Minimize the Conjunctive Costs of Node Replacement and Performance Loss in the Management of Wireless Sensor Networks," *IEEE Transactions on Network and Service Management*, vol. 7, no. 2, pp. 107-117, 2010.

[14] Huang J. Y., *et al.,* "Shielding Wireless Sensor Network using Markovian Intrusion Detection System with Attack Pattern Mining," *Information Sciences*, vol. 231, pp. 32-44, 2013.

[15] Natarajan H., and Selvaraj S., "A Fuzzy based Predictive Cluster Head Selection Scheme for Wireless Sensor Networks," *International Conference on Sensing Technology*, pp. 560-566, 2014.

[16] Mao S., *et al.,* "An Improved Fuzzy Unequal Clustering Algorithm for Wireless Sensor Network," *Mobile Networks and Applications*, vol. 18, no. 2, pp. 206-214, 2013.

[17] Stojanova D., *et al.,* "Estimating the Risk of Fire Outbreaks in the Natural Environment," *Data Mining and Knowledge Discovery*, vol. 24, no. 2, 2012.

[18] Zhang Y., and Ye Z., "Short-Term Traffic Flow Forecasting using Fuzzy Logic System Methods," *Journal of Intelligent Transportation Systems*, vol. 12, no. 3, pp. 102-112, 2008.

[19] Ramadan A.B., *et al.,* "New Environmental Prediction Model using Fuzzy Logic and Neural Networks," *International Journal of Computer Science*, vol. 9, no. 2, pp. 309-313, 2012.

[20] Salarian H., Chin K. W., and Naghdy F., "An Energy-Efficient Mobile-Sink Path Selection Strategy for Wireless Sensor Networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 5, pp. 2407-2419, 2014.

[21] Gayathri S. and Divya R. "Heuristic Approach for Discovery and Recovery of Fault Nodes in Wireless Sensor Networks," *International Journal of Engineering Research & Technology*, vol. 4, no. 3, 2015.

[22] Tian J, Liang X, and Wang G., "Deployment and Reallocation in Mobile Survivability-Heterogeneous Wireless Sensor Networks for Barrier Coverage," *Ad Hoc Networks*, vol. 36, pp. 321-31, 2016.

[23] Sugeno M., and Kang G.T., "Structure Identification of Fuzzy Model," *Fuzzy Sets and Systems*, vol. 28, no. 1, pp. 15-33, 1988.

[24] Kaur A., and Kaur A., "Comparison of Mamdani-Type and Sugeno-Type Fuzzy Inference Systems for Air Conditioning System," *International journal of soft computing and engineering*, vol. 2, no. 2, pp. 323-325, 2012.

[25] Shokouhifar, M., and Jalali A., "Optimized Sugeno Fuzzy Clustering Algorithm for Wireless Sensor Networks," *Engineering Applications of Artificial Intelligence*, vol. 60, pp. 16-25, 2017.