IoT based system for detection of gas leakage and house fire in smart kitchen environments

Conference Paper · November 2019

DOI: 10.1109/TELFOR48224.2019.8971021

2 authors:

Marjan Ralevski
Ss. Cyril and Methodius University in Skopje
2 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Biljana Risteska Stojkoska
Ss. Cyril and Methodius University in Skopje
75 PUBLICATIONS **1,541** CITATIONS
SEE PROFILE

IoT based system for detection of gas leakage and house fire in smart kitchen environments

Marjan Ralevski and Biljana Risteska Stojkoska

***Abstract* —The future holds a broader range of available technologies that will offer more innovative ways for solving everyday issues. By combining small processing units with artificial intelligence and machine learning, one can expand the horizon of new concepts and ideas to increase everyday safety. In this paper we have designed a cheap Internet of Things based system which enables the early detection of house fire and gas leaks. We are simulating a scenario where we detect the rising possibility of house fire in a kitchen environment, by measuring temperature and the gases concentration. To optimize the communication process and reduce the number of sent packets from the measuring node to the system gateway, we applied time series forecasting approach based on moving average prediction scheme.**

***Index terms* — Internet of Things, Wireless Network, Fire Detection, Smart Kitchen.**

I. INTRODUCTION

AS technology progresses towards simplicity, the already solved problems remain to be improved. Additionally, the problems are constantly evolving and growing, especially in environments where the human safety factor is included. One such environment is the industrial kitchen, where many people are working with a chaotic tempo. Thus, the possibility of the almost inevitable incidents is always there. The spectrum of these incidents includes house fire and gas leaks, which when combined can wreak havoc. From these misfortunes, the need for early detection and prevention of incidents is higher than ever [1]. Luckily, technology improvements provide us with new, faster and more sensitive sensors and more compact, simpler computing devices. The new paradigms of pervasive and ubiquity computing make Internet of Things (IoT) reality and create new exciting opportunities for designing the next-generation detection mechanisms [2].

In this paper, we design a system which enables early detection of house fire and gas leaks. Additionally, as the size of industrial kitchens increases, we cover the possibility for dangerous gas incidents, caused by gases such as liquefied petroleum gas (LPG) and carbon monoxide (CO). The system is implemented using cheap state-of-the-art off-the-shelf IoT components. This system logic relies on a few rules that define fire/hazard and the communication between the dedicated devices. Furthermore, we create a higher level of connectivity between the system and the end user, which will provide instantaneous notification to the user's mobile communication device. Nonetheless, the data gathered by the measuring devices provides exciting opportunities for statistical analysis of the environmental conditions and trends.

The rest of the paper is as follows. Section II describes the architecture of our system. Section III provides details about the system implementation. The paper is concluded in Section IV.

II. ARCHITECTURE

In this section we are explaining the system conceptually, as well as the hardware used for enabling the system's functionality (Fig. 1).

A. Concept

This project is targeting an industrial kitchen environment. Conceptually, the system tries to cope with the chaos that exists in the aforementioned environment. The main idea is the end user to be notified as soon as the system detects an intolerant presence of at least one of the three gases that are measured. The message that the user receives is a formatted highly informational one, which contains a simple status explanation for each of the three measured gases. This way, the user will become aware of the possibility for an incident or even worse, will become aware that some misfortune has occurred. This project was accomplished in a simulated environment where two Raspberry Pies equipped with sensors measure different parameters. For the sake of brevity, the Raspberry Pie that is being used to measure the temperature in degrees Celsius is going to be referred as piTemp, while the other one that is being used for measuring LPG, CO and smoke concentration in parts per million (ppm) is going to be referred as piGas (Fig. 2). The last component of the system is a laptop, to which we have given the role of a gateway and are going to refer to it in the future context as gateway. The system, when operating, is always in one of the following states:

1. *Initialization*
2. *Passive temperature measuring*
3. *Measuring of gases*
4. *Reasoning the gas presence status*

This project was financially supported by the Faculty of Computer Science and Engineering.

Marjan Ralevski is with the Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, Rudzer Boshkovikj 16, P.O. 393, 1000 Skopje, Republic of North Macedonia (e-mail: marjan.ralevski@students.finki.ukim.mk)

Biljana Risteska Stojkoska is with the Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, Rudzer Boshkovikj 16, P.O. 393, 1000 Skopje, Republic of North Macedonia (e-mail: biljana.stojkoska@finki.ukim.mk).

978-1-7281-4790-1/19/$31.00 ©2019 IEEE

5. *Notifying the end user*

In the first state - *initialization* piTemp takes three temperature measurements and sends them to the gateway. The gateway sets these values as initial and uses them in order to be able to predict the current temperature. This is achieved with the Moving Average 3 (MA3) algorithm. MA(3), with the same three values is running on the piTemp as well. From now on, the system is in the *passive temperature measuring* state. In this state, piTemp makes temperature measuring on a given interval. If the absolute difference in temperatures, meaning the predicted and the last measured temperature is under the threshold, then the system remains in this state. But if this difference is above the threshold, then piTemp sends the new measured temperature to the gateway.

From there on the system is in the *measuring of gases* state. In this state, the piGas listens to the gateway's request and starts to measure CO, LPG and smoke concentration in the working environment. Then it simply returns the measured values, expressed in parts per million - ppm. From this point, the system is in the *reasoning the gas presence status*, in which the gateway compares the measured gas concentrations with the standardized, regulative values. If the readings are above the limits, then the system goes in the *notifying the end user* state. In this state the gateway formats an informative JSON message which includes all of the measured gas values, as well as an descriptive explaining for each of them. This message is then sent to the end user's mobile device via HTTP PUSH requests.

The concept of data prediction based on time series forecasting is not new. It has its roots in wireless senor networks as a strategy to reduce transmissions [3] [4], but lately was reinvented in IoT perspective combined with fog computing approach [5].

In the literature, many algorithms can be used for time series forecasting, like moving average (MA) of different orders, autoregressive (AR), combination of AR and MA, like ARMA, integrated ARMA (ARIMA), least mean square with constant (LMS) or variable step size (LMS-VSS), etc. Still, implementing forecasting for data collected from embedded devices requires fundamentally different approach compared with traditional forecasting, since embedded devices are restricted by means of memory and computational power. The focus here should be more on the lightness of the forecasting method, rather than the quality of the prediction by means of prediction error. The algorithm should be simple and fast, to be able to operate in real-time under memory and computational constrains.

Many researches have been done to compare the algorithms by means of reduction of data transmission, as percentage of the total number of measurements. Generally, MA of small orders perform better than MA of greater order, but this conclusion cannot be generalized since it depends on many factors, like the nature of the data (indoor vs outdoor) and the frequency at which the data is being sampled. Choosing the right algorithm is usually application specific task and needs to be done separately for each data being collected.

In this research, we have chosen MA (3) - MA od third order, because in our previous research it has been proven to be good predictor for indoor temperature measurements.

A. Used hardware

In order to be consistent with the previous categorization of the hardware units, in this subsection we are going to present the used hardware in three groups. The first group or functional unit is the piGas subsystem. For this subsystem we used the following hardware components:

- Raspberry Pi 1 Model B
- MQ5 gas measuring sensor [6]
- Simple logic level converter
- MCP3008 analog-to-digital converter

We used the MCP3008 component in order to transform the analog signals that the MQ5 sensor [6] sends as measurements into digital signals that the Raspberry Pi can fully understand. Furthermore, we connected the MQ5 sensor [6] on the analog pin on the Raspberry Pi so we can improve the precision for measuring the gas concentration from the working environment. To satisfy the need of the gas sensor for alternating voltage when calibrating the resistance, the logic level converter was used to convert 3.3 V to 5 V and vice versa. This chip was used because the sensor needs variable voltage. Additionally, the MQ5 gas sensor [6] offers predefined curves that represent parameters used for calibrating. For example, we were given the curve for performing LPG measurements, but we calculated a curve for the CO and smoke measurements (described in Section III in more details).

The second functional unit is a gateway. We use laptop (Lenovo Y700, which was running on Windows 10 Professional operating system) but smartphone can also be applied for this purpose. The gateway generates data used for later analytical purposes.

The last functional unit is the piTemp subsystem, which is significantly simpler compared to piGas. For this subsystem we used the following hardware components:

- Raspberry Pi 3 Model B
- DS18B20 temperature measuring sensor
- 4 kΩ resistor

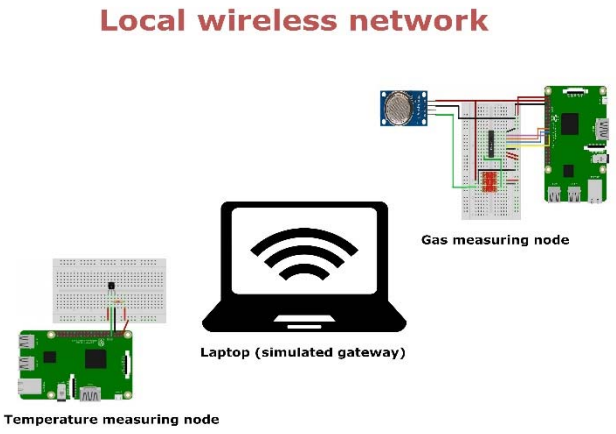

Fig. 1 Simplified representation of the whole system.

The used DS18B20 temperature measuring sensor is able to provide precise temperature measurements in the range [-55, -125] degrees Celsius, which is even more than what we needed, based on the fact that a modern industrial kitchen does not reach such extreme temperatures. It is a simple three-pinned sensor, with one pin for digital data signals. A resistor was used to reduce the sensitivity of the temperature sensor and hence, decrease the time needed for one temperature measurement. DS18B20 on the Raspberry Pi's GPIO 4 pin was connected using the 1-Wire protocol, which is used for transferring a small amount of data on longer distances, while using cheaper devices, meaning we could connect the sensor on a bigger distance from the Raspberry Pi itself.

Additionally, both Raspberry Pi were running on the Debian operating system.

II. Implementation

In this section we are going to explain the state transition process and each state in details.

In the first state - *initialization*, piTemp measures the temperature three times, on a 10-second interval and sends them as query parameters in a simple synchronous HTTP GET request to the gateway. After this, both the piTemp and the gateway run in the background the Moving Average 3 (MA3) (1).

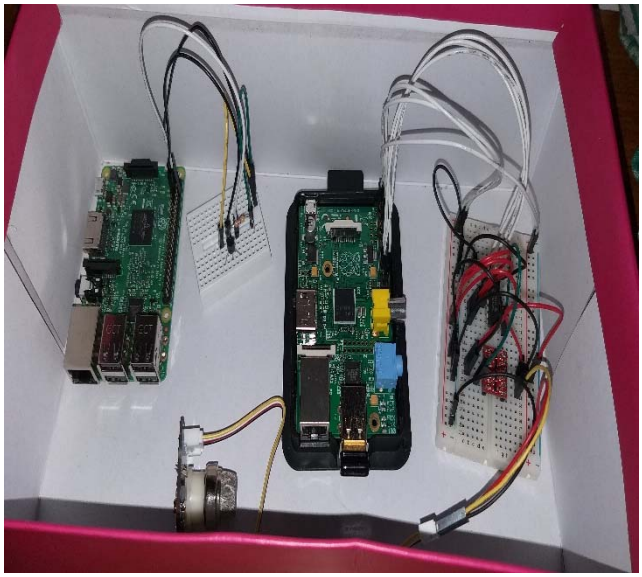

Fig. 2 Physical appearance of piTemp in the upper left corner and piGas on the right.

$$MCT = \frac{MT_{t-3}+MT_{t-2}+MT_{t-1}}{3} \quad (1)$$

The gateway can answer to a GET request with a prediction of the current temperature, based on the last three received measured temperatures. This enables the system to provide information about the current temperature without the need of new measurements. This prediction paradigm is well known technique in the literature as an efficient method for data reduction and energy saving in wireless networks [7].

Furthermore, all of the received measured temperatures from piTemp are written in a simple textual file, with additional suffix for the date and time of receiving them. This allows for future statistical analysis of the measured date, for example for analyzing and improving data prediction algorithms. In the second state - *passive temperature measuring* piTemp measures the temperature on the regular 10-second interval. We have chosen this interval in order to balance between the rate of sent packages from piTemp to the gateway and the security and reliability on our system. From the materials we read, we concluded that if we measure the temperature too often, we are simply wasting resources because a too sudden increase of temperature in an industrial kitchen environment would most probably mean an explosion, for which we did not create our system.

If piTemp measures a temperature whose absolute difference from the predicted temperature is more than 10 degrees Celsius, then the piTemp sends an HTTP GET request to the gateway. We set the threshold on 10 degrees Celsius while relying on the fact that in an industrial smart kitchen environment, the temperature varies more often, especially when preparing different types of meals. Moreover, the predicted temperature is based solely on the MA(3). Furthermore, the new request is in the same format as the initial HTTP GET request that sends the first three measured temperatures to the gateway. After this the system goes in the *measuring of gases* state, in which piGas measures CO, LPG and smoke concentration in the working environment. The first time piGas receives a measuring request, it goes in the initialization state in which it calibrates the resistance of the gas sensor in order to be most compatible with the surrounding working environment. This improves our ability and reliability to work in an industrial kitchen environment where the concentration of gases, such as LPG is higher from the normal. We also use a 3-second interval when measuring the surrounding gas concentration in order to cope with the worst-case scenario, meaning to catch the maximum value for each of the three gases. After completion of the measuring, the measurements are sent to the gateway, from which point the system is in the *reasoning the gas presence status* state. In this state, the gateway firstly writes the received values for each of the three measured gases in a textual file in order to provide further analytical opportunities. Then, it checks if the measured concentrations are above the standardized regulated values that the Occupational Safety and Health Administration (OSHA) [8] prescribes, and if they are, it enters in the *notifying the end user* state. In this state, the gateway formats a highly informative message that contains a quantitative and descriptive parts. This message is sent to the end user's mobile device via HTTP PUSH request and from here the system leaves the decision making about the current situation to the end user. If the measured gas concentrations are not above the regulated values, then the system goes back in the *passive temperature measuring* state. Furthermore, the message is in the following format:

LPG: x ppm. LPG STATUS: OK/BAD. CO: y ppm. CO STATUS: OK/BAD. SMOKE: z ppm.
SMOKE STATUS: OK/BAD.

where the x, y and z variables represent the measured values in ppm. With this format the end user does not need to be familiar with the allowed and standardized concentrations

of gases because we included the STATUS part. Moreover, for the HTTP PUSH requests we used the Pushover mobile application [9].

A. Calibration equation

The MQ5 gas sensor [6] comes with a few predefined curve equations that represent parameters used for calibrating the sensor's resistance when it is in the initialization state. Some of these equations cover the curve for calibrating the sensor for measuring hydrogen, LPG, methane and alcohol. Still, the provided curve equations do not cover all of the gasses that the sensor can measure. For this purpose, the MQ5's documentation comes with a graph [6] that provides visual aid so we can calculate new equations for curves that cover the rest of the available gases for measuring, as show on Fig. 3.

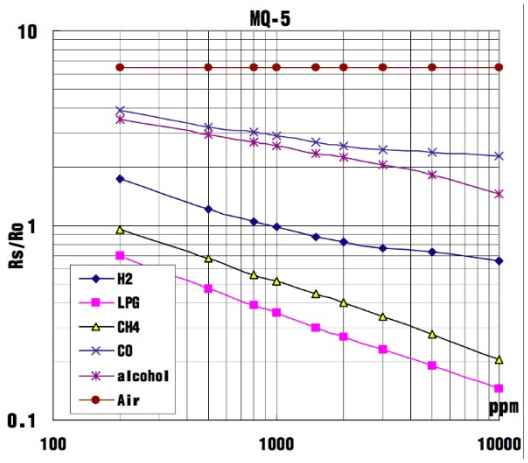

Fig. 3 MQ5 curve equation derivation graph.

According to this graph, we approximately take the leftmost point of the curve as a starting point and we take the rightmost point of the curve as an ending point. From where, based solely on the equation for finding a line equation with two given points (2), we calculate the slope of the line.

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1} x - x_1 \quad (2)$$

The calculated slope is then inserted into a logarithm with base 10 and the result is the actual slope of the gas curve that is being used when calibration the sensor's resistance, as shown in (3).

$$S = \frac{y_2 - y_1}{x_2 - x_1} \quad (3)$$

The curve is now described with the tuple x, y, s, where x and y are the coordinates of the starting or ending point and s represents the calculated slope. We have done this method for calculating the tuple for carbon monoxide, which resulted with the [2.3, 0.72, -0.34] tuple.

B. Rules

If the absolute difference of the predicted temperature and the latest measured temperature is higher than 10 degrees Celsius, then the piTemp notifies the gateway. The gateway compares each of the three received gas concentrations and compares them with the respective regulated values according to OSHA [8]. If at least one of the received gas concentrations is higher than the appropriate regulated value for it, then the gateway starts to format the message that is going to be send to the end user via HTTP PUSH request. The logic behind this decision is safety. We wanted to cover the scenario when at least one of the measured gas concentrations is higher then the safe value, the end user is notified about this. With this decision we cope with the risk that comes from possible gas poisoning or maybe even an explosion. For example, if we detect only a higher concentration of smoke, that could possible mean that a stove was forgotten to be turned off. Even worse, if we detect only a higher concentration of LPG, that could infer the fact that there is a gas leak in the kitchen, which may hold disastrous consequences.

We used the maximum value method when measuring the gases. This approach enables us to cover the worst-case scenario when measuring the concentrations of the gasses, which is consequentially providing a better risk prevention in combination with the previously explained method of comparing the received gas values with the prescribed regulations.

III. Conclusion

In this paper we described our system for early detection of gas leaks and possible risks for house fire. Moreover, we explained our decision for combining our system with the Moving Average 3 algorithm in order to provide better energy efficiency. This approach provides useful information when trying to develop a fully distributed system based on real models that can operate in the slightly extreme conditions and communicate between themselves with a low-energy consumption protocol. That is why we took into consideration the energy efficiency part and used the Moving Average 3 algorithm. We simulated the work of our system in an industrial smart kitchen environment.

References

[1] B. Risteska Stojkoska, K. Trivodaliev, and D. Davcev, "Internet of things framework for home care systems", *Wireless Communications and Mobile Computinig*, vol. 2017, 2017.

[2] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of internet of things for smart home: Challenges and Solutions", *Journal of Cleaner Production*, vol. 140, pp. 1454-1464, 2017.

[3] S. Santini, "An adaptive strategy for quality-based data reduction in wireless sensor networks".

[4] B. Risteska Stojkoska, A. Popovska Avramova, and P. Chatzimisios, "Application of wireless sensor networks for indoor temperature regulation", *International Journal of Distributed Sensor Networks,* vol. 10, no. 5, p. 502419, 2014.

[5] B. R. Stojkoska and Z. Nikolovski, "Data compression for energy efficient IOT solutions" in 2017 25*th Telecommunication Forum (TELFOR),* pp. 1-4, IEEE, 2017.

[6] "International E City seeed studio.", https://www.seeedstudio.com/Grove-Gas-Sensor-MQ5.html. Accessed: 2019-10-30.

[7] B. R. Stojkoska and K. Trivodaliev, "Enabling internet of things for smart homes through fog computing" in *2017 25th Telecommunication Forum (TELFOR)*, pp.1-4, IEEE, 2017.

[8] R. Nester, "Occupational safety & health administration" *Workplace Health & Safety,* vol. 44, no. 10, pp. 493-499, 1996.

[9] "Pushover pushover, LLC", https://play.google.com/store/apps/details?id=net.superblock.pushover. Accessed: 2019-10-30.