Cloud Based WiFi Multi-Sensor Network

https://doi.org/10.3991/ijoe.v14i08.8536

Anekwong Yoddumnern, Roungsan Chaisricharoen, Thongchai Yooyativong
Mae Fah Luang University, Chiangrai, Thailand.
mycoreidea@gmail.com

Abstract—A WiFi technology was run on a smart device using three types of sensors: a temperature sensor, smoke-CO, and PIR sensor. This project sought to design a multi-sensor node. All of the sensors were connected to a microcontroller unit (MCU) with a general purpose input output (GPIO). After the connection, there were invalid multi-sensor data when the processor started up. First, the temperature sensor did not work. Second, the smoke-CO sensors read an invalid value higher than the actual. This situation was solved through sensor calibration methodology following calibration time and self-calibration with a Finite Impulse Response (FIR) filter. After the system ran for a long time, there were invalid data with values different than the truth. While the system was running, there was some noise and heat collected on the device, creating invalid values. These errors were solved by using the Full Scale Kalman Filter (FSKF) to estimate the right value. Next, the OFF-Mode was used to save power consumption by 37.24% before sending the sensor data to the Cloud. These methods helped the device run a long time and work a long life. Finally, there was a high-performance WiFi multi-sensor network.

Keywords—WiFi, Multi-Sensor, FIR Filter, Calibrate, Calibration Time, Self-Calibration, FSKF, Kalman Filter, OFF-Mode, Low Power, Cloud.

1 Introduction

In general, a single sensor node is accessible because it merely works at a more comfortable usage level [2]. However, a single sensor node does not cover much work time. The function of a multi-sensor node is to detect and report events in a multi-processing mode, especially in the pattern of a high-security system or as an accurate system. The multi-sensor node is a necessity in system projects – such as the Internet of Thing (IoT) [3] [5] [16]. The system can access and use the WiFi signals everywhere and at any time. The multi-sensor node can detect the analog signal and digital signal by co-operating to serve more and multiple projects.

This project was run with four sensors on an microcontroller unit (MCU). They are set in a case study of a Home Security System (HSS): temperature sensor, humidity sensor, smoke sensor, and PIR motion sensor. The MCUs were an Arduino-UNO, NodeMCU and LinkIt Smart 7688 Duo. After the circuit had run 12 times—following the number of the MCUs, there was a problem with the sensors detecting invalid values. There was a goal of low cost and creating the ideal device following the current

IT market with the trend of IoT [19]. Most of the sensors detected invalid low and high data values from the truth rather than smooth values. When the system ran for an extended time, the sensors detected invalid data values, especially the analog sensor. The multi-sensors detect all time and send the values to the Cloud storage via the WiFi module following each processing cycle. These are unnecessary and do not save power. Table 1 shows the experiment with some errors of data detection.

Table 1. The MCUs detected signal value.

Type of MCU	**Sensor Detection**					
	Temperature sensor		***Smoke detector***		***PIR movement detector***	
	Startup	***Loop-Detecting***	***Startup***	***Loop-Detecting***	***Startup***	***Loop-Detecting***
Arduino UNO#1	Invalid-high	Invalid at 463	Invalid-high	Invalid low-high	-	-
Arduino UNO#2	Fail	Invalid low-high	Invalid-high	-	Fail	-
Arduino UNO#3	Invalid-high	Invalid at 412,413	Invalid-high	-	-	-
Arduino UNO#4	Invalid-high	Invalid at 224	Invalid-high	Invalid at 119	-	-
*ArduinoUNO #5	Fail	Invalid at 211,278	-	Invalid at 423	-	-
NodeMCU#1	Fail	Invalid low-high	Invalid-high	Invalid at 342	-	-
NodeMCU#2	Invalid-high	Invalid at 74,266	Invalid-high	Invalid low-high	Fail	-
NodeMCU#3	Fail	Invalid at 320	Invalid-high	Invalid at 390	-	-
*NodeMCU#4	Fail	Invalid at 195	Invalid-high	Invalid at 287	Fail	-
*NodeMCU#5	Invalid-high	Invalid at 337	Invalid-high	Invalid at 124	-	-
LinkIt-Smart 7688 Duo	Invalid high	Invalid at 434	Invalid high	-	-	-
*LinkIt-Smart 7688 Duo	Invalid high	-	-	Invalid at 207	-	-
Percentage (Error Detection)	Invalid 58.33 %	Invalid 97.67 %	Invalid 83.33 %	Invalid 75.00 %	Fail 25.00 %	-
	Fail 41.67 %	No Error 2.33%	No Error 16.67 %	No Error 25.00 %	No Error 75.00 %	-

Note: (1) The startup was processed at first to 50 delta-time.
(2) The loop detecting was processed between 51 to 500 delta-time.
(3) These [*] used the old MCU twice.
(4) These [–] were mean valid processed.
(5) These [Invalid-high] were the highest data value of the truth.
(6) These [Invalid-low-high] were higher-lower data values from the truth.
(7) These [Fail] did not detect a data value

Table 1 confirms the MCU and multi-sensor showed error detection in all parts of the processing. Consequently, the startup of the temperature sensor was invalid

(58.33%) and fell to 41.67%, the smoke-CO sensor invalid value was 83.33%, and the PIR motion sensor fails were 25.00%. The loop detecting the invalid temperature was 97.67%, and invalid smoke-CO sensor was 75.00%.

The multi-sensing element had more than one sensor on the MCU. When the system startup experiences a running failure, this disrupts some sensors from scanning the situation, such as the temperature sensor not being able to detect an accurate value due to the system sending only a zero figure until some seconds had passed. Some sensors, such as the Smoke-CO sensor, will detect the situation with the wrong data, such as high-value data until some seconds passed and then it would read it as the right value. Consequently, calibrating each sensor should be completed. In this case, the calibration in time from the average of GPIO function assigned and the sensor duration can change to a different state. The use of the FIR filter takes the sensor work as being right for pre-processing [10]. In the loop portion process, it discovered that the invalid data from any sensor comes from an error device in which heat is collected in the circuit or from the outside noise signal. This process used the FSKF for filling and estimating the right data value [8] [13] [18].

Table 1 shows the problem of experimenting between a multi-sensing element and a single MCU-detected error with two main points independent from the MCU had processed with multi-sensors, but following the MCU changed 12 times, to parallel the analog and digital sensors. First, the startup step collapsed from an invalid high and fall of each sensor from the truth. Second, while they were running, the detection produced invalid low-high sensors for the PIR motion sensor. This case can apply sensor calibration and the Kalman filter to hold up and estimate the sensor data values.

A digital filter has many ways to process reduced noise and maintain a data signal. Yuan Zhuang and his teams (2016) proposed a two-filter integrated suitable for real-time application with smart devices, their limit of the proposed algorithm. Li Yuanyuan and his colleagues (2016) used MATLAB to set the method of a Max-hold frequency-domain. This study worked well for noise cancellation but had to adjust the center frequencies several times. Feng Guo (2015) examined noise cancellation methods for time series analysis. He improved the SNR (signal to noise ratio) and found the sensor value, validity and this method could not be used in direct processing. Rekha N and Fathima Jabeem (2015) proposed a stochastic approach for noise cancellation (SANC) in GSM signals. Their study performed cancellation of the noise by incorporating white Gaussian noise and stored it in a matrix format. Next, they used time series. The temporal data matrix of the contaminated signals used a digital low pass filter for filtering the amount of randomly distributed noise. The simple formula was designed with weight function for checking the cutoff point in the filtering process. The equation informs noisy signal equality to clean the signal by adding white Gaussian noise. The last pattern of the signal values was the matrix form. The SANC was capable of eliminating the noises [24].

A mathematic pattern set the Kalman filter and provided efficient recursive means to estimate the state of the object processing by minimizing the means in the squared error. The filter can be used for prediction of random signals. There should separation of random signals from random noise and to detect the signals in the presence of random noise [26]. Using a Kalman filter to remove noise from a signal and the measuring process to be described by a linear system, Jinhong Lim and his team (2015) proposed the Kalman filter based on Gaussian Process with dead-reckoning to estimate the position of a mobile robot and to control it. Their work set the model of the two-wheeled robot with the controlling algorithm under the wireless sensor networks. Jinhong Lim, Jae Hyun Yoo and H. Jin Kim (2015) had high performance with their proposed method. Shu-Li Sun (2004) presented the multi-sensor optimal information fusion algorithm weighted with the matrix form in the linear minimum variance in that his algorithm can handle as well as inform fusion with multi-sensors for the system. Olfati-Saber (2007) presented collaborative data problems in the wireless sensor network (WSN) and found that multi-sensor problems have a long history in Kalman filters. He modified the DKF algorithm from the node. He had a sensor with covariance data. The MCU was set at a high-pass filter with Kalman filter iterations. His model reduced disagreement in the estimates by different nodes. He confirmed his algorithm had the best overall performance.

All of the previous review shows that the Kalman filter is suitable for multi-sensors and WiFi processing to reduce noise. This work focuses on a digital signal filter with multi-sensor to reduce the noise's sensor data and achieve stability. The sensors were prepared and set with sensor calibration. The purpose was to design and implement a new model of a multi-sensor filter and to estimate the sensor value. The final goal was to find a method to save power consumption and work for an extended lifetime.

2 Methodology

When the system is running, a part of the multi-sensor can be a barrier for detecting data and sending it to the MCU because the detector is not ready to work. First, the pre-processing should set the calibration processing. This calibration time can be set from a duration time of GPIO and calculated by the average of the sensors. This step has a dynamic time. After this event, the detector impulse is completed by using self-calibration with the FIR filter. The main loop should filter and estimate the sensor value by the FSKF. Finally, the OFF-Mode should compare and check the sensor value before sending it to Cloud storage. This will support saving power consumption (see Fig 1).

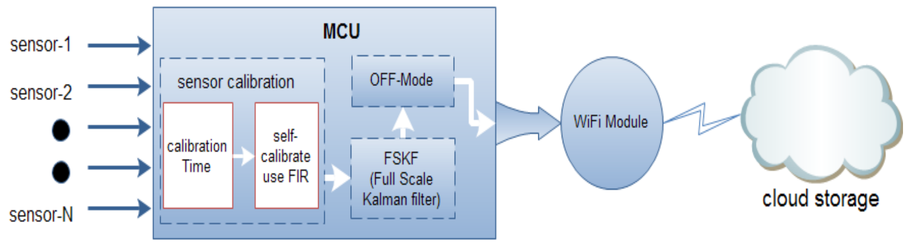

Fig. 1. Calibration and FSKF with OFF-Mode Framework.

This concept applies to any MCU for the IoT society that meets the minimum requirement of the microcontroller of not less than 8 bits with CPU speed 8 MHz, Ram not less than 2 Kbyte, program memory/EEPROM more than 1 Kbyte and at least a 12 Kbyte flash memory. These include LinkIt Smart 7688/Duo, Arduino UNO/ NANO/MKR1000, Intel Edison, Raspberry-Pi 2/3, NodeMCU, etc. with several sensors of the analog and digital output values.

The MCU works with all of the analog sensors and digital sensors. The sensors send the data value to the system in the setup portion. The sensors were adjusted using calibration time and self-calibration with the FIR filter. Sensors in the main-loop portion are filtered and estimated by Kalman filtering. Furthermore, the sensor value is taken to compare with the previous value and current value. If they are suddenly equal, they should be put into the OFF-Mode and another one should send data to the Cloud storage.

2.1 The multi-sensor calibration

Calibration is a methodology that includes the process of adjusting the output, where there is an indication of a measurement of the device to agree with the value of the applied standard within a specified accuracy [11]. In this case, we used a method following calibration time and in terms of self-calibration.

Calibration time. According to the datasheet, most of the sensors should show the time for calibration between 7 and 50 seconds. The general state sets the maximum time for the sensors. This should make the processing take more time. If the sensor's output pin goes to HIGH, it is present. However, if it is present, it goes to LOW from time to time. After setting the GPIO to run, it might give the impression that it is not present. This program deals with the issue by ignoring LOW phases shorter than a given time, assuming that it is continuous if it is present during these phases. Finally, the calibration time to be used as the dynamic time process follows up on the millisecond function with the multi-sensory. This step changes each sensor from HIGH to LOW, and the summary of this time to process for the system (equation 1) was 0.88, 2.70 or 7.44 seconds.

Let:

calbSensor1,2,.,N = duration of setup pinMode(sensor1,2,..N) into millisecond()

*calibrationTime = (calbSensor1+calbSensor2+..+calbSensor-N)/(N*1000)   (1)*

This calibration time (equation 1) set to generate the algorithm as:

Algorithm calibrationTime (N as the number of the sensor)

```
1. Input: duration of time process
2. Output: calibration time as the summary time for all of the sensors.
3. Initialization: set i as an integer
4. do
5.     delay ← calibrationTime
6.     i ← i + 1
7. until  i > N
```

Self-calibration using Finite Impulse Response (FIR) filter. When the sensors run and detect the environment, then they have a data signal value. Most of the signals are not smooth. They are taken to an Analog to Digital Converter (ADC), which takes the input, as x(n), to the digital low-pass filter and the output,y(n) after, into the Digital to Analog Converter(DAC) and then has the best data signal [10]. The data of digital filters can achieve virtually any filtering effect it can express as a mathematical function and an algorithm [1].

The discrete system transforms the data sequences of inputs as a signal into sequences of the output signal. This is assigned as the following: y[n] = x[n].

The operator equation and the difference equation correspond to this delay-adder-gain block diagram. This study characterized them by using a linear equation that shows the relationships between the consecutive values of the input and output signals (see equation 2).

$$y[n] = \sum_{k=0}^{n} x(k) \tag{2}$$

The WiFi signal is a digital filter in which an electronic filter is based on a small set of a simple linear shift-invariant operation in the sequence. So the Finite Impulse Response (FIR) filtering is the discrete-time between the input signal and the impulse response of the filling.

This approach requires four multiplications and accumulations for each output. The finite impulse response of discrete time filters should be processed by following equation 3.

$$y(n) = \sum_{k=0}^{N} h_k x(n-k) \tag{3}$$

Input x(n) and output y(n) signals are related via the convolution, h(k)—filter coefficients, and N is a feature-length (following the block diagram). All terms can be set to equation 4. For a 4-point FIR filter as

$$y(n) = \frac{1}{4} \sum_{k=0}^{n} x(n-k) \tag{4}$$

Thus [h] is the coefficients of the impulse response with x as input sensors, y(n) is the output sensors at time n, and big N is the length of the coefficients impulse response—following equation 3 and the short term for all filter coefficients backward with the input forward as in equation 4.

Each output requires one multiplication, one addition, and one subtraction to find the Fourier transformation of the output sensor signal to obtain the amplitude-frequency characteristic of the filter. The length of the self-calibration—the input all sensor signals—is 1000 points and the setting of the array is variable. After filtering the sensor signal, the results are stored in the y[i]; consider the algorithm of the FIRsensor function.

Algorithm FIRsensor (N-sensorNo as an integer of the sensor number)

```
1.  Input: x[i]as the data sensor input.
2.  Output: y[i]as the data sensor output.
3.  Initialization: x[1000] = {0}, x[0] = 1, y[1000] = {0}, yn =0, N=4,
                    h[N] = { 0.25, 0.25, 0.25, 0.25 }, x[N] = { 0,0,0,0 }, i=0
4.  do
5.        for k ← N-1 to 0
6.        begin
7.             x [k] ← x [k-1]
8.        end k
9.        x[0] ← x[i]
10.       Yn ← 0;
11.     for k ← 0 to N
12.       begin
13.            yn ← yn + h[k] * x[h];
14.       end k
15.       y [i] ← yn;
16.       i ← i+1
17. Until i > N
18. Return y[i]
```

2.2 Full-Scale Kalman Filter (FSKF)

R.E. Kalman has published studies describing the recursive solution of the discrete linear data filtering problem since 1960 [14]. Since that time, digital computing has been quite popular and provides more in-depth research on the concept. The Kalman filter is a mathematical equation that provides an efficient recursive for computing mean. This filter is very powerful in several aspects, as it supports estimations from the past to the present. The precise nature of the system model is unknown [13].

The mathematics of the discrete Kalman filter with the time update equations are presented in specific equations as

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{5}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{6}$$

After the time updates in equations 5 and 6, the state and covariance estimate forward from time step *k-1* to *k*. Variables A and B are updated, while Q is used to process the noise covariance with normal probability [18].

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{7}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{8}$$

$$P_k = (I - K_k H)P_k^- \tag{9}$$

During the measurement, the system updates the calculation of the Kalman Gain (KG), K_k , and then the equation is given as equation 7. The next step is to measure the process to obtain Z_k and then to generate a posteriori state estimate, the measurement takes on equation 8. The final step is to obtain a posteriori error covariance estimate – equation 9.

An algorithm of the Kalman filter uses a series of the measurements observed over time. Usually, these measurements will contain noise that collects errors of the next delta time. The process will try to estimate a new value based on the previous and current states. The system tends to be more precise than only taking the measurements [12], [13].

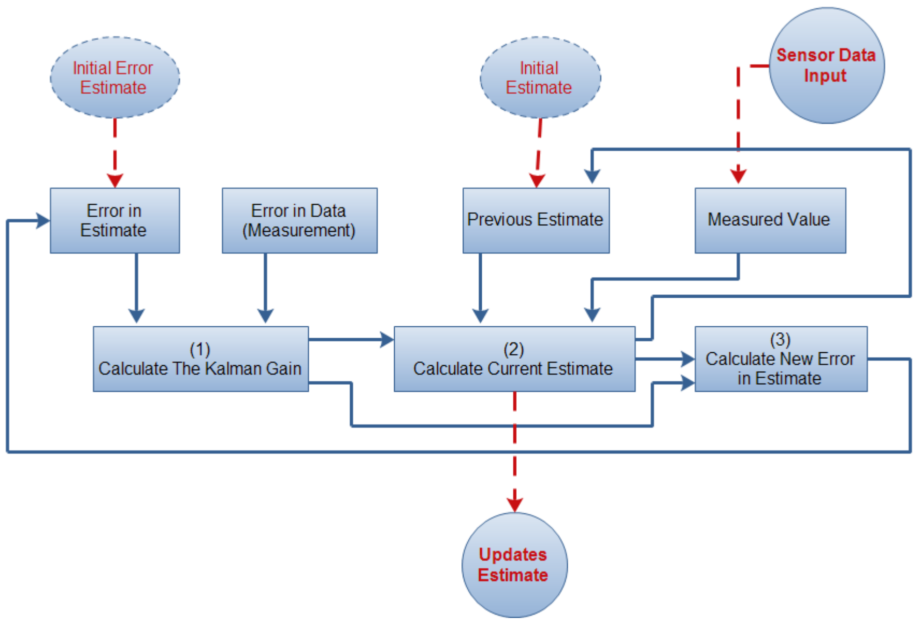
Initial Error Estimate
Initial Estimate
Sensor Data Input
Error in Estimate
Error in Data (Measurement)
Previous Estimate
Measured Value
(1) Calculate The Kalman Gain
(2) Calculate Current Estimate
(3) Calculate New Error in Estimate
Updates Estimate

Fig. 2. The FSKF diagram with sensors data input.

KGs between 0 to 1 are significant to the measurement, which is accurate/ inaccurate with the estimates that are unstable/ stable.

The KG is taken from the errors in the estimation to calculate the proportion of the error in the data with a sensor measurement that has a value from 0.00 to 1.00.

Let

$$KG = \frac{E_{est}}{E_{est} + E_{mea}} \quad ; \text{where } 0 \le KG \le 1. \tag{10}$$

The 3-block calculations.

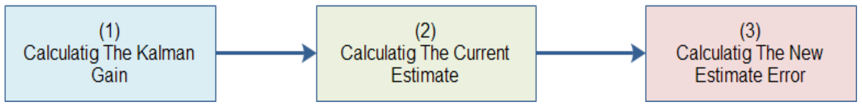
(1) Calculatig The Kalman Gain
(2) Calculatig The Current Estimate
(3) Calculatig The New Estimate Error

Fig. 3. The 3 state calculations.

Following the Kalman filter, processing should take the inside values to combine and calculate that based on the consequences of block 1—the Kalman gain estimate with the MCU is shown as block 2—the current estimate value. At last, data checking in the form of block 3, estimate errors, should identify the system has been remarkably stable.

Algorithm KalmanFilter::correctData (sensorData)

```
1. Input: sensorData
2. Output: KG, ESTstate, errEST
3. Initialization: x0 = (Meax)(ESTstate), p0 =(Mea)(Mea)(errEST) + $Q_{insideloop}$
4. KG ← (Emes )( p0)/((Emes) (p0) (Emes) + (errMea))
5. ESTstate ← x0 + Mea (sensorData – (Emes) x0)
6. errEST ← (1 – (Mea)(Emes) p0)
```

2.3 The OFF-Mode

When the system is running, these filters cannot turn off the sensor. However, if it wants to save power, there should be control of the multi-sensors with an OFF-mode and it should be able to enter it from any state by removing the power state. When the sensor is in the sleep mode under the microcontroller in these conditions, it becomes deficient, less than one milliAmp for some models [17].

The multi-sensory approach is used for reducing power consumption (Low-power sensing, 2015). There are one or lower power states, and the low-power operating mode of the main loop is made of two parts: the setup and loop processing. Hence, this type of sensor operation is immediately held by the command state.

The MCU with the sensor is entirely off (same VDD=0) in the OFF-mode. The detector can run on for a millisecond every minute, but basically there is no consummation of power.

Algorithm: The OFF-mode state of the sensor for turning down power consumption

```
1. Input: sensor (n)
2. Output: HIGH, LOW
3. Initialization: state = {LOW, HIGH, DELAY}
4. Delay ← N millisecond
5. If sensor[n] is HIGH then
6.    Send (value)
7.    Sensor[n] ← LOW
8. Else
9.    Delay ← N millisecond
10.   Sensor[n] ← HIGH
11. End if
```

Note: N is the value of a Cloud circle process (ThingSpeak.com has been 15000 milliseconds).

3 The Experiment and Result

3.1 The MCU and WiFi module with the multi-sensing element

The hardware is connected by any of the MCU to the multi-sensors (see Fig.5) with two main parts:

- The MCUs used Arduino UNO, NodeMCU and LinkIt Smart 7688 Duo with a WiFi module.
- Home security sensors monitored temperature, humidity, motion, and smoke.

The sensors of motion, smoke, humidity, and temperature were wired to the MCU (Arduino UNO/NodeMCU and LinkIt Smart 7688 Duo board) circuit. The WiFi module was connected to the MCU so it would be able to connect to the router and the circuit via wireless signals after the WiFi module sends a sensor data signal to the Cloud storage system.

A cost of the necessary components with the multi-sensor is $22.98. The cost of the WiFi module is $25.81, and the cost of GSM-module is $65.44. The average cost of the device is cheaper than 50% of internet market products such as www.indiegogo.com or www.aliExpress.com. This case suggests setting up a WiFi-module compact device. It uses a free WiFi signal from a free home internet and stands on 802.11 b/g/n.

Circuit design: Connecting GPIO between sensors, WiFi module and the MCU.

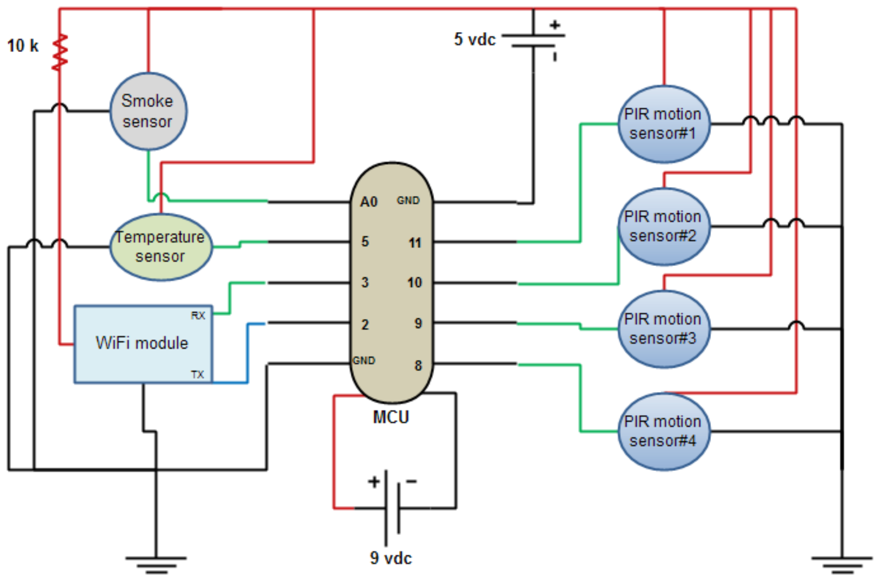

Fig. 4. The leading circuit design with the multi-sensing element.

Currently, the MCU has the same purpose as Raspberry-Pi, Arduino, Intel Edison, ESP32, NodeMCU or ESPresso lite. The GPIO should interface with the data analog signals, and the digital signal should follow the properties of the sensor. This system causes a potential difference at around 3.3 or 5.0, where the VDC should be working. The study used an Arduino and NodeMCU with four sensors: temperature sensor, humidity sensor, smoke detector, and PIR-motion detector. The Arduino used the WiFi module as NodeMCU-ESP01. The researchers needed to update the firmware that can move and reinforce the AT-command.

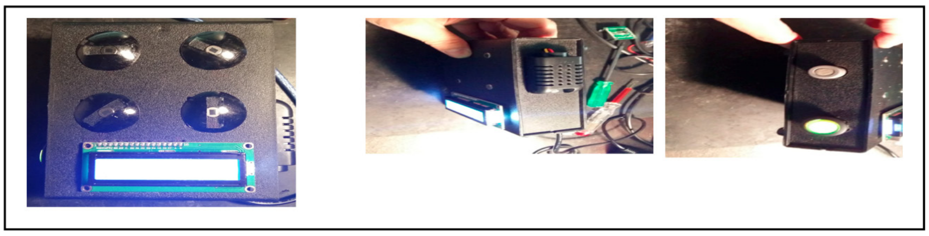

Fig. 5. The MCU and WiFi module—this CPU frequency set processing at 80 MHz upon a default value.

3.2 The calibration time processing

The calibration time was processed in dynamic time, even with the number of detectors that were set up in the system. This study found that the calibration time should help the sensor work well when starting up. There are suitable ways to define the first step of the device that should be the best for working (see Fig 6). Then the sensors come to the pre-processing check following the mill()-function from the pinMode()-function upon setting them. Subsequently, summary and looping were run following for-loop I >N: when I was the integer counter and N was the number of sensors.

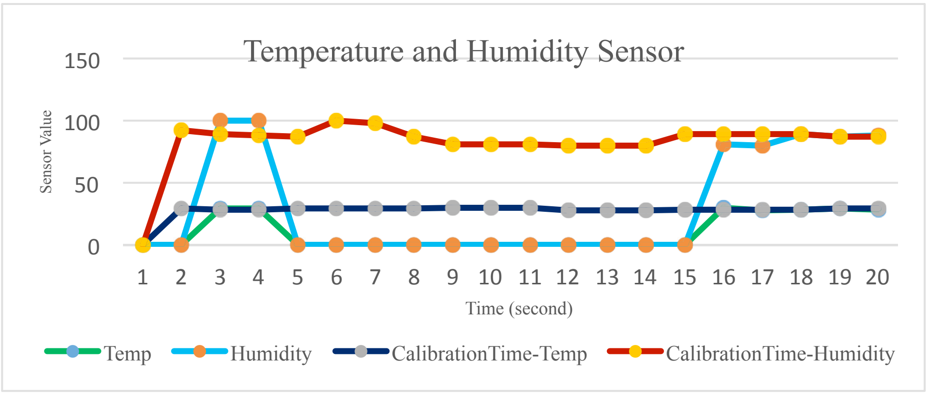

Fig. 6. The result of calibration time for temperature and humidity sensing element.

3.3 The self-calibration with FIR processing

For self-calibration with the FIR filter, when the startup could become the invalid of truth, it could also act as the impulse of the sensor. The MCU changed the value near the truth before the system processed it into a loop portion. If the smoke-CO sent value was 442, Let [n] was the coefficient data. Let n = 0, the coefficient = 105.50, n=1 the coefficient = 105.50, n=2 the coefficient = 211.00, and n=3 the coefficient = 316.50.

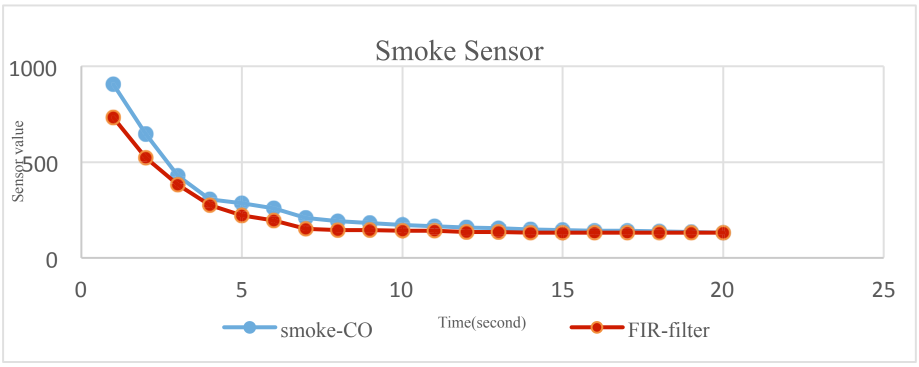

Fig. 7. Self-calibrate by FIR filter.

3.4 The FSKF processing

Solve: To set the smoke sensor with the FSKF when the scheme made the measurement smoke sensor data 144, 127, 130 and 119. This initial measurement set the first step of delta time with the smoke sensor measurement of 150 after setting Δt= t+1, t+2, and t+3 to calculate the same method with KG, EST_t, and E_{EST_t}, as demonstrated in Table 2.

Table 2. Summary of the appraisal in the smoke sensor with the FSKF.

Delta Time	MEA	E_{mes}	EST	$E_{EST_{t-1}}$	KG	E_{EST_t}
t-1			132.00	2		
t	144	4	135.96		0.33	1.34
t+1	127	17	135.31		0.07	1.24
t+2	130	3	133.75		0.29	0.88
t+3	119	11	132.66		0.07	0.81

Table 2 shows the error of approximation in the delta time and then reduces the KG when the system runs a long time. Finally, the system was highly stable.

The Kalman Gain inverted to a measurement that had a value equaling 1, which is highly accurate, but when the Kalman Gain stepped down to 0, it had low accuracy. It is different from the estimation that the Kalman Gain that equals 1 is unstable and stable when it is equal to 0.

The FSKF can help rapidly reduce error data value. All of this processing was filled and estimated to obtain accurate data for the truth (see Fig 8). While the system was working with multi-sensors and eliminating the noise, this filter could estimate the data value as being right and highly stable.

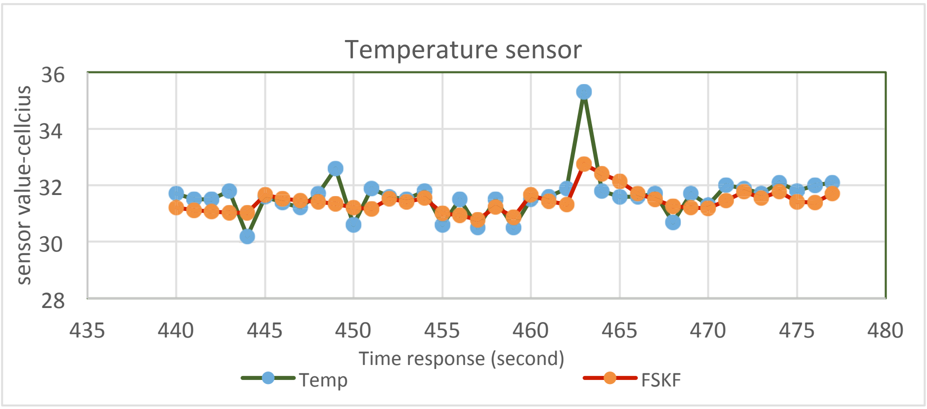

Fig. 8. The FSKF with a temperature detector.

3.5 The Multi-Sensor is running with OFF-Mode and Cloud processing

After, the multi-sensor was calibrated and estimated with the FSKF, the researchers took off the OFF-Mode processing and sent the value to the Cloud (thingspeak.com) via the WiFi module.

This experiment did not use the OFF-Mode; the system used the currency by 17400 mA for each sensor running with the Tx of the WiFi module by setting the OFF-Mode. The Smoke-CO saved about 20.34%, the Temperature saved about 58.96%, the Humidity saved about 32.41%. The three sensors were run in 120 delta times with a power to save 6479.56 mA from 17400 mA. The complete process saved 37.24%.

4 Discussion and Conclusion

When the MCU and sensors are running an inside loop portion, they can process for a more extended period. The sensors can detect an invalid value and values that are swaying. These have noise from the inside of the system where the heat is being gathered from the device or extraneous noise the system is detecting all of the time [15]. All of this will have an impact on the system because there is no way to change the procedure. Although this parallel processed between the FSKF and the OFF-Mode, it will reduce the noise and save power consumption (Fig. 9) before sending the data to the Cloud storage.

This study sustained the system working with a multi-sensor by setting more sensors to the processor on the MCU to adjust by calibrating the device. In the startup step, the sensors were set to be an immediate step. Subsequently, the system was getting noise and some of the values had errors or were invalid. The FSKF should reduce noise and estimate the measurement value, while the OFF-Mode should save power consumption.

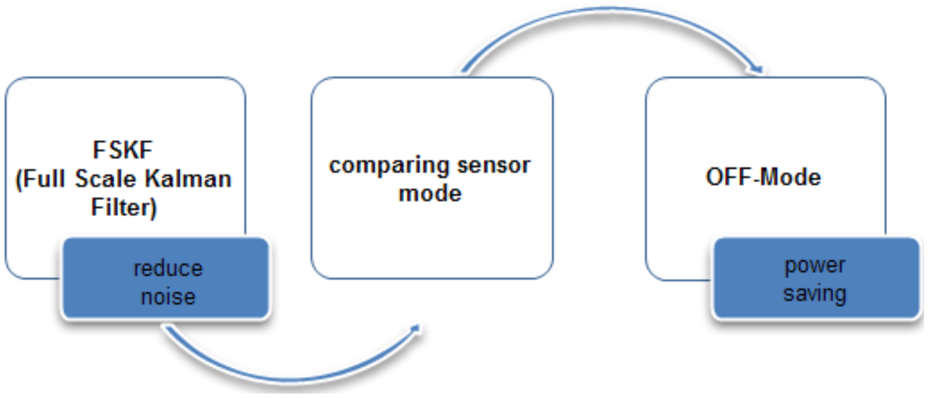

Fig. 9. Parallel processing between FSKF and OFF-mode

The multi-sensor node should run as well as the calibration time and FIR calibration in pre-processing. The FSKF should support the system and nearly estimate data the accuracy with the OFF-mode before sending it to the Cloud storage. Finally, there was high performance of the WiFi multi-sensor values.

After setting the sensors and the MCU was connected to a WiFi signal, the device was run continuously. The calibration helped to calibrate valid data. Following any sensor with its duty, the calibration time changed any sensor run to balance by itself. The FIR filter sent impulses to the sensor to run in a short time under the scope of pre-processing. It was suitable for running in the digital pin processing. The FSKF run followed the multi-sensor at the same time and did not run slowly. When the system was running a long time, this method was taken to be stable. The OFF-Mode helped to save power consumption by 37.24%. It supported the lifetime of the sensor. One caution is that any MCU has a watchdog function. It should reboot the device when the delay-function has run for a long time—more than 2 minutes or 5 minutes depending on the kind of MCU, especially if it used the same multi-sensor. These findings could be used to design an algorithm to check the watchdog processing.

5 Acknowledgements

I would like to thank my family, and especially my mother, for the massive support in doing this research. Finally, I would like to send my deepest appreciation to my advisor Asst. Prof. Gp. Capt. Thongchai Yooyativong, Ph.D. and Asst. Prof. Roungsan Chaisricharoen, Ph.D. from the School of Information Technology, as well as the scholar-ship support from the Mae Fah Luang University.

6 References

[1] David Dorran (2015), “Digital Signal Processing Foundations”, Dublin Institute of Technology, School of Electrical and Electronic Engineering at ARROW@DIT, 2015.

[2] Anindya Maiti and S. Sivanesan (2012), “Cloud Controlled Intrusion Detection and Burglary Prevention Stratagems in Home Automation Systems”, School of Computing Science & Engineering VIT University Vellore, India. 2nd Baltic Congress on Future Internet Communications, 2012. pp.183-186.

[3] A. Sinha and A. Chandrakasan (2000),“Energy Aware Software”, Proc. Thirteenth International Conference on VLSI Design, Jan. 2000, pp. 50-55. https://doi.org/10.1109/ICVD.2000.812584

[4] Hong-Son Vu and Kuan-Hung Chen (2016), "A Low-Power Broad-Bandwidth Noise Cancellation VLSI Circuit Design for In-Ear Headphones",IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 24, NO. 6, JUNE 2016, p 2013-2025.

[5] R. Caceres, A. Friday (2012), "Ubicomp systems at 20: progress, opportunities, and challenges" ,IEEE Pervasive Computing, 11, pp. 14–21. https://doi.org/10.1109/MPRV.2011.85

[6] K. Bult et al., “Low Power Systems for Wireless Microsystems”, Proc. ISLPED 1996, pp. 17-21.

[7] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica and Matei Zaharia (2010), “A view of cloud computing”, Communications of the ACM, Volume: 53, Issue: 4, pp.50–58. https://doi.org/10.1145/1721654.1721672

[8] Shengwen Chen, Chunghuang Yang, Chung-Huang Yang (2011), "Design and Implementation of Live SD Acquisition Tool in Android Smart Phone". 2011 Fifth International Conference on Genetic and Evolutionary Computing, pp. 157-162. https://doi.org/10.1109/ICGEC.2011.46

[9] Dexiang J. Xu and Michael L. Daley (1995),"Design of Optimal Digital Filter Using a Parallel Genetic Algorithm", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-11: ANALOG AND DIGITAL SIGNAL PROCESSING, VOL. 42, NO. IO. OCTOBER 1995.

[10] Emmanuel C. Ifeachor and Barrie W. Jervis (2008), “Digital Signal Processing: A Practical Approach”, 2nd Edition. Hardcover from English. - Moscow: Publishing House Williams, 2008. - 992 pp.

[11] ROBERT c. POWELL, AND ANNE c. MILLER (1987), "Determination ofthe Reflection Correction when Using a Symmetrical Two-Resistor Power Splitter to Calibrate a Power Sensor", IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, VOL. IM-36, NO.2, JUNE 1987 p.458-467.

[12] Sasikanth Avancha, Chintan Patel and Anupam Joshi (2011), “Ontology-drivenAdaptiveSensorNetworks”, Departmentof ComputerScience and ElectricalEngineering, 1000 HilltopCircle, University of Maryland Baltimore County,Baltimore,MD 21250. 2011.

[13] HSIN-I LIU, JONATHAN KOTKER, HOWARD LEI, AND BABAK AYAZIFAR (2012), "ECG SIGNAL FILTERING", Electrical Engineering 20N Department of Electrical Engineering and Computer Sciences University of California, Berkeley, 2012.

[14] R. Olfati-Saber (2007), "Distributed Kalman Filtering for Sensor Networks", 46th IEEE Conference on Decision and Control New Orleans, LA, USA, Dec. 12-14, 2007. https://doi.org/10.1109/CDC.2007.4434303

[15] JON A. WELLNER (2012), "Gaussian White Noise Models: Some Results for Monotone Functions", National Science Foundation grant DMS-95-32039 and NIAID grant 2R01 AI291968-04, UNIVERSITY OF WASHINGTON STATISTICS Box 354322 SEATTLE, WASHINGTON 98195-4322 U.S.A.

[16] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos (2014), "Context Aware Computing for The Internet of Things: A Survey", IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 16, NO. 1, FIRST QUARTER 2014.

[17] Fernando E. Valdes-Perez Ramon Pallas-Areny (2009), "Microcontrollers Fundamentals and Applications with PIC", CRC Press: Taylor&Francis Group, 2009.

[18] Greg Welch and Gary Bishop (2001), "An introduction to the Kalman Filter", University of North Carolina at Chapel Hill, Department of Computer Science, Chapel Hill, NC, 2001.

[19] George W. Musumba, Henry O. Nyongesa (2013), "Context awareness in mobile computing: A review", Department of Computer Science, Dedan Kimathi University of Technology Kenya and Department of Computer Science, University of the Western Cape, South Africa, The Authors.Licensee: AOSIS OpenJournals. Copyright: 2013.

[20] Yuan Zhuang, You Li, Longning Qi, Haiyu Lan, Jun Yang, and Naser El-Sheimy (2016), "A Two-Filter Integration of MEMS Sensors and WiFi Fingerprinting for Indoor Positioning", IEEE SENSORS JOURNAL, VOL. 16, NO. 13, JULY 1, 2016.

[21] Li Yuanyuan, Xie Shuguo, Zhang Tong, and Hao Xuchun (2016),"Max-hold Frequency-domain Measurement for Noise Cancellation", 7th Asia-Pacific International Symposium on Electromagnetic Compatibility, IEEE 2016.

[22] Feng GUO (2015),"Noise Cancellation Method of the Telemetry Signal Based on Time Series Analysis", 2015 International Conference on Computer Science and Mechanical Automation.

[23] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita (2005), "A Line in the Sand:A Wireless Sensor Network for Target Detection, Classification, and Tracking", Department of Computer Science and Engineering, The Ohio State University, IEEE Computer, Jan. 2005.

[24] Rekha N and Fathima Jabeem (2015), "SANC: Stochastic Approach for Noise Cancellation in GSM Signals", International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 978-1-4673-9223-5/15/$31.00c2015 IEEE, pp.701-707.

[25] Lan, K. T. (2010), "What's Next? Sensor+Cloud?," Proceeding of the 7th International Workshop on Data Management for Sensor Networks, ACM Digital Library, p. 978–971, 2010.

[26] Robert J. Schilling and Sandra L. Harris (2015),"Digital Signal Processing using Matlab 3th", Clarkson University Potsdam, NY, Printed in the United States of America, 2015.

[27] Jinhong Lim, Jae Hyun Yoo and H. Jin Kim (2015), "A Mobile Robot Tracking using Kalman Filter-based Gaussian Process in Wireless Sensor Networks", 15th International Conference on Control, Automation and Systems (ICCAS 2015) Oct. 13-16, 2015 in BEXCO, Busan, Korea. https://doi.org/10.1109/ICCAS.2015.7364990

[28] Shu-li Sun (2004), "Multi-sensor optimal information fusion Kalman filters with applications", Elsevier SAS.ScienceDirect, Aerospace Science, and Technology 8 (2004) 57–62. https://doi.org/10.1016/j.ast.2003.08.003

7 Authors

Anekwong Yoddumnern is a Ph.D. student of the School of Information Technology, Mae Fah Luang University, Chiangrai 5700, Thailand. He entrusts to the WiFi Multi-Sensor processing and reinforces of robotic. This study is a passion of his one project under the generation with MCU. His contacts with my-coreidea@gmail.com or general purpose to contact by anek.cr@gmail.com.

Thongchai Yooyativong is a professor of the School of Information Technology, Mae Fah Luang University, Chiangrai 5700, Thailand. His contacts with thongchai.mfu@gmail.com.

Roungsan Chaisricharoen is a professor of the School of Information Technology, Mae Fah Luang University, Chiangrai 5700, Thailand. His contacts with ch.roungsan@gmail.com.

Article submitted 5 March 2018. Resubmitted 07 April and 19 May 2018. Final acceptance 09 June 2018. Final version published as submitted by the authors.