Sensors & Transducers, Vol. 218, Issue 12, December 2017, pp. 19-26

www.sensorsportal.com

Sensors & Transducers
Published by IFSA Publishing, S. L., 2017
http://www.sensorsportal.com

A Modular Wireless Sensor Platform with a Sensor Identification Scheme

Yi-Jie Hsieh, Chih-Chyau Yang, Yi-Jun Liu, Wei-Lin Lai, Chien-Ming Wu, and Chun-Ming Huang
National Chip Implementation Center
National Applied Research Laboratories,
No. 26, Prosperity Rd. 1, Hsinchu City, Taiwan
Tel.: (+886)3-5773693
E-mail: yjhsieh@cic.narl.org.tw, ccyang@cic.narl.org.tw, yjliu@cic.narl.org.tw, wllai@cic.narl.org.tw, wucm@cic.narl.org.tw, cmhuang@cic.narl.org.tw

Received: 10 November 2017 /Accepted: 10 December 2017 /Published: 29 December 2017

Abstract: In this paper, a modular wireless sensor platform with a sensor identification scheme is presented. Our presented modular wireless sensor platform consists of sensor modules, and each sensor module is a part of the sensor system and in charge of one job in the system, such as computation, communication, output or sensing. Users can stack multiple modules together to build a unique sensor system. Since users are able to easily replace one module with others, our platform is highly extendable and reusable. A low-cost sensor identification scheme is proposed in this paper to detect which sensor is mounted on the platform automatically. This scheme utilizes a unique I^{2}C address to identify the sensor type. A low-cost Electrically-Erasable Programmable Read-Only Memory (EEPROM) only needs to be setup in the non-I^{2}C sensing modules. Furthermore, a firmware initialization process is also adopted to achieve the sensor identification mechanism. To demonstrate the proposed platform, we show an ambient temperature detection application and a carbon dioxide detection application in the paper. The results show that the proposed platform is suitable for academic researches and industrial prototype verification.

Keywords: Flexible sensing platform, Sensor system.

1. Introduction

The Internet of Things (IoT) development has progressed rapidly in the past few years. This concept was widely used not only for the industry and research purposes, but also in commercial products in our daily life [1-5]. The idea of IoT is to group “things” together with internet, allow “things” to communicate or interact with each other, and even, to gather their information and utilize it. Numerous IoT devices have used wireless sensors to recognize environments due to the continuously increasing availability of wireless sensors. Besides, wireless sensors are also widely used in many research fields, such as sensor networks and sensor fusion.

There are several ways to build a sensor platform. The first one is to manually compose different sensor units according to requirements. In [6], Spanbauer, *et al.* proposed a sensor cube called MICA. Each MICA node contains multiple sensors inside. Different kinds of MICA nodes can be applied in various applications. To build up these kinds of sensor platforms, users must have enough hardware knowledge and resources. Besides, these self-made sensors are designed for specific purposes, so they have limited extendibility and reusability. The second way is to use existing sensor platforms. However, most ordinary sensor platforms are designed for sensing only one feature, so multiple sensors have to be used for multi-sensing applications. This will increase cost, lose accuracy, and cause synchronization problems. There are also products that divide a sensor node into wireless module and sensor boards, such as MicaZ [7]. For different purposes, users can stack different sensor boards on the wireless module, so the reusability is further increased. However, the architecture usually allows only one sensor board connecting with the wireless module. Although some sensor boards have multiple sensor modules, the flexibility is still confined.

To support various researches and product developments, a broad range of wireless sensors is required. In this paper, we present a flexible wireless sensor platform which enables users to arbitrarily combine different modules with few constraints, so that they can create a unique sensor system according to their requirements. For the purposes of extensibility and reusability, we divide the sensor system into six units: output, sensing, communication, processing, power, and debug unit. Each unit is in charge of one specific function in the system. To build a sensor platform, users can select required sensor modules and stack them one by one, just like building bricks. This feature makes the proposed platform highly flexible and reusable.

Since our presented platform supports a variety of sensor types and sensor interfaces, how to make the sensor system easy to use has become an important task. Normally, every time we stack a different sensing module on the processing module, we have to download the corresponding firmware code to the Micro Control Unit (MCU) in order to drive this module. This work not only increases complexity for application developers, but also brings inconvenience to common users. A sensor identification scheme is therefore inevitable to identify which sensor is mounted on the platform automatically. In [8-9], R. Morello, *et al.* adopted the Transducer Electronic Data Sheet (TEDS) based on IEEE P1451 standard to store sensor information. This method allows microprocessors to access data through a standardized protocol and to realize the sensor self-identification. For the only purpose of the sensor identification, the hardware cost of this method is high since the TEDS memory needs to be setup in each sensing module. K. Mikhaylov, *et al.* [10-11] proposed a sensor identification mechanism by using the Intelligent Modular Periphery Interface (IMPI). The IMPI is implemented as a daisy chain interface based on the Serial Peripheral Interface (SPI) bus. However, the bus routing complexity is high while the number of interconnections becomes large. In this paper, a sensor identification scheme is proposed to identify the sensors on the platform automatically. This scheme utilizes a unique I^2C address to identify the sensor type. A low-cost EEPROM only needs to be setup in the non-I^2C sensing module. Furthermore, a firmware initialization process is also adopted to enable the sensor identification mechanism.

The rest of this paper is organized as follows. In Section II, we present the idea of modular sensor platform. Then, the hardware implementation and the proposed sensor identification scheme are described in Section III. Next, we use two examples for demonstration in Section IV. Finally, the conclusions are given in Section V.

2. Flexible Sensor Platform

Our presented sensor system is divided into six units for the purposes of extensibility and reusability. Each unit is in charge of one function in the system:

1) Power unit which provides power to all other units is the key influence factor of the sensor life time and sensor size. A power unit can be, for example, Li-Po battery, button cell battery, or car charger.

2) Processing unit has to drive other units and execute firmware commands. A processing unit can be a MCU, a Field-Programmable Gate Array (FPGA), or just a controller.

3) Sensing unit is one of the main components in the sensor system to recognize surrounding environments, such as acceleration, color, image and so on.

4) Communication unit is used for communication. It can receive commands, transmit results, and relay messages. Here we focus on only wireless transmission, such as Bluetooth Smart (BLE), ZigBee, or Wi-Fi.

5) Output unit shows computational results and reminds users by screen, sound, or vibration.

6) Debug unit provides debug functions. When a sensor module is stacked on the debug unit, the designer can check signals of each pin and download images from PC.

Each unit has I/O pins to receive commands and transmit data. These pins can be standardized I/O pins such as Inter-Integrated Circuit (I^2C), Serial Peripheral Interface Bus (SPI), Inter-IC Sound (I^2S) and Universal Asynchronous Receiver/Transmitter (UART), or they may be General-Purpose Input/Output (GPIO) pins defined for the specific usage. For communication between different units, we define a universal bus that connects all units together and is implemented by connectors, as shown in Fig. 1. The universal bus contains three kinds of pins, standard I/O pins, GPIO pins, and power lines. All signals from the processing unit are physically connected to the universal bus, so the processing unit can control other units through the universal bus. As for the power lines, they deliver electric power from the power unit to others.

Under the definition of each unit, we further classify a unit into modules. Each unit can have multiple modules. Each module is one of the implementations of the unit. For example, the transceiver unit may have BLE module and Wi-Fi module, the sensing unit may have compass module and thermometer module, and the power unit may have Li-Po battery module and button cell battery module. The reason we classify a sensor platform into units and modules is to increase flexibility. For each unit in a highly flexible platform, users should have more choices to replace one module with others. Fig. 1 shows the schematic view of a sensor module in the proposed platform. The green and black areas are PCB substrate and electronic components, respectively. Each module is implemented on an equal-size PCB substrate which has connectors on both front side and back side. Thanks to the universal connector, different modules can be combined concurrently, as illustrated in Fig. 2. Here, two connectors are used in order to increase the stability of the architecture. The area between two connectors on the PCB substrate is for placement of electronic components.

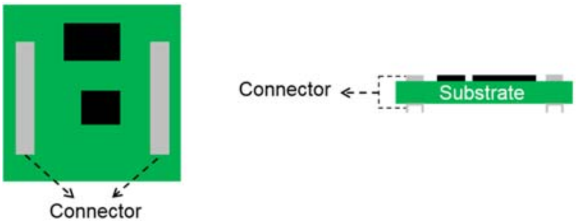

Fig. 1. The top schematic view and side schematic view of a sensor module.

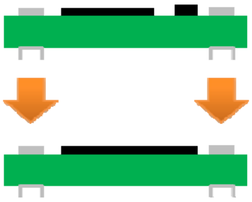

Fig. 2. Combination of two modules.

3. Implementation

In this section, we introduce the implementation of the presented sensor platform and the proposed auto sensor identification scheme.

3.1. Hardware Implementation

Our presented flexible wireless sensor platform primarily consists of

1) Power module;
2) Processing module;
3) Sensing module;
4) Communication module;
5) Output module;
6) Debug module.

The power module includes a Li-Po battery, a power management unit, a Near-Field-Communication (NFC) control, a wireless charging unit and the coils. In the current implementation, the processing module includes not only an MCU unit but also a 9-axis motion sensor and a BLE unit. The sensing module, communication module, output module or debug module can be integrated in the platform through the I^2C, SPI, UART, I^2S, and analog interfaces. Fig. 3 shows the appearance of our power module, processing module, and sensing module. Each module size is 35 mm × 35 mm. The processing module is a 32-bit ARM Cortex-M3 processor, supporting I^2S, I^2C, SPI, UART, and analog interfaces. The BLE communicates with the MCU by SPI interface. There are four universal connectors on each module, two on the top side and the other two on the bottom side. As mentioned in the previous section, all peripheral signals and power lines are delivered through these connectors.

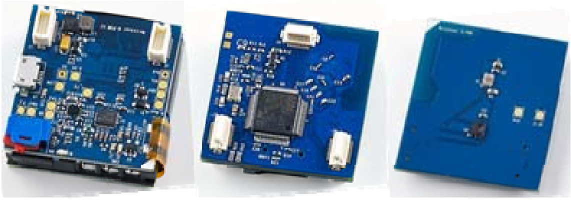

Fig. 3. The pictures of power module, and processing module and an temperature sensing module.

Another important feature of the proposed sensor platform is the mountable ability. For some applications, such as altitude detection, the sensor has to be tightly mounted on the object. For this purpose, we design packages for sensor boards. Fig. 4 shows packaged sensor bricks with different colors. Each sensor unit is given a unique color.

Fig. 4. Packages for attachable and wearable applications.

The mapping table of sensor units and their corresponding colors are described in Table 1. Besides the five colors mapping to five units, the purple ones are sensor mounts. Currently, there are mounts for bats, wrists, tripods, belts, flat surface, and magnetic surface.

Table 1. Package colors and sensor units.

Color	Units	Color	Units
Blue	Analog sensing unit	**Orange**	Processing unit
Green	Digital sensing unit	**Red**	Power unit
Yellow	Communication unit	**Purple**	Sensor mount

3.2. Auto Sensor Identification Scheme

In this paper, an auto sensor identification scheme is proposed to detect which sensor is mounted on the platform automatically. The users can therefore launch the corresponding user's application according to the identified sensor type. This scheme utilizes the unique I^{2}C address to identify the sensor type. A low-cost EEPROM only needs to be setup in the non-I^{2}C sensing module. Moreover, a firmware initialization process is also adopted to enable the sensor identification mechanism.

Currently, the transmission interfaces of the sensor modules to MCU can be analog voltage output, I^{2}C, I^{2}S, UART and SPI, etc. Among these interfaces, the I^{2}C device possesses a unique 7-bit address, which is used to appoint a certain I^{2}C device to perform the operations of data write or read. Normally, the unique I^{2}C device address is configured in advance, so that the address for each I^{2}C sensor device can be different. This characteristic can thus be used to perform the sensor identification. For those non- I^{2}C sensor devices, we can just add an I^{2}C-interfaced EEPROM on the sensing module and use it to perform the sensor identification as the method used in I^{2}C sensors.

Fig. 5 shows the hardware architecture of our proposed sensor identification scheme. All the sensor signals are connected directly to the MCU. Each sensor with I^{2}C interface owns its unique I^{2}C address (Addr.), and for the rest of non-I^{2}C sensing modules, we add an I^{2}C EEPROM and configure it with a unique I^{2}C address. Before the MCU starts to read sensor data, it scans the I^{2}C address of the sensor first.

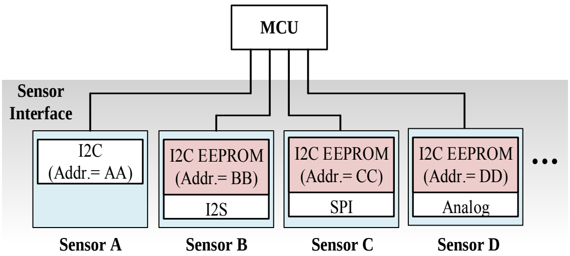

Fig. 5. The hardware architecture of the sensor identification scheme.

Since the I^{2}C address of each sensor is set to be unique, it can be used to identify the sensor type. For example, if the result of I^{2}C address scan is AA, the MCU identifies that the sensor is "Sensor A", and then starts to read the sensor data with the I^{2}C protocol; if the result of I^{2}C address scan is BB, the MCU determines that the sensor is "Sensor B". Since the interface of Sensor B is I^{2}S interface, the MCU starts to read the sensor data with the I^{2}S protocol. The same flow can be applied to the other sensors. With this kind of hardware design, we can identify the sensor type automatically by using the unique I^{2}C address.

To achieve the auto-sensor-identification function in our presented platform, the MCU needs to perform a firmware initial process, as shown in Fig. 6. First, we need to turn on the power of the flexible platform. The MCU then starts to perform the hardware initialization setting (INITIAL_HW) which initializes required peripheral controllers. After the hardware initialization completes, MCU starts to scan the number of sensors plugged-on and the sensor IDs (SCAN_SENSOR). The sensor IDs are defined based on the I^{2}C address of the sensing module. When the scan process completes, MCU begins to initialize those sensors connected on it (INIT_SENSOR) and put the initial sensor data in the built-in table. In the following step, MCU reads the Media Access Control (MAC) address of the BLE device and writes it in the Near-field communication (NFC) Tag (READ_NFC_TAG). Then, the MCU enters the waiting status, and waits for the BLE connection and communication (WAIT COMM LINK).

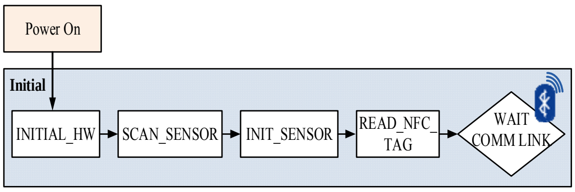

Fig. 6. Firmware initialization process.

After the presented flexible platform finishes the firmware initialization process, it is ready to connect the smart phone App with the BLE. For now, there are two ways to establish BLE connection. Firstly, users can open the App in their smart phone and see all available proposed platforms around with different MAC addresses. Afterwards, users manually select the target device to establish the BLE connection. Secondly, users can use the NFC functions of both the smart phone and the platform to set up the connection. Once you move the NFC sensing area of the platform towards that of a smart phone, the smart phone detects the MAC address of the flexible platform through NFC tag thus knows which device to connect with. The BLE connection can thus be established automatically.

In the case of the BLE transmission between the smart phone and the platform, the smart phone acts as a master and the platform acts as a slave. After the BLE connection between the two devices is established, the smart phone starts to send commands, while the platform starts to receive and decode the commands, then responds to the smart phone. Fig. 7 shows the handshaking flow for realizing auto-sensor-identification. The smart phone sends a command to be aware what sensors are plugged on the platform (Retrieve Sensor List), while the platform decodes the command and responds with the sensor IDs and quantity (Receive Sensor ID & No.). In this way, the smart phone can identify what sensors are present on the platform, and open the corresponding App page. The smart phone then sends a command to retrieve the measurement results of the sensors (Retrieve Sensor Data). The platform executes the command and transfers the sensor data to the smart phone (Receive Sensor Data). Consequently, the smart phone shows the received data on the App. Normally, the smart phone will send the Retrieve Sensor Data command continuously to get the latest sensor data.

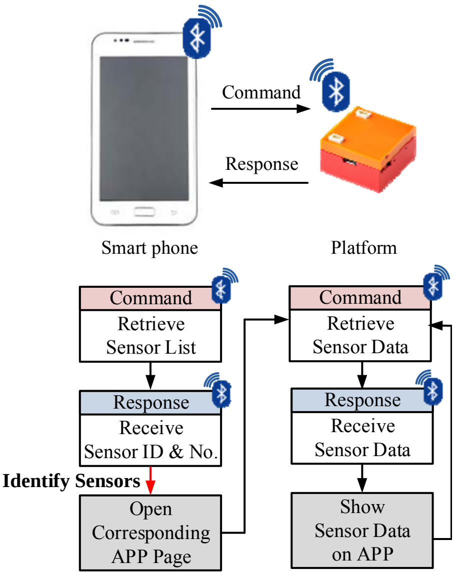

Fig. 7. Communication flow path between smart phone and platform.

4. Applications and Results

For demonstration of the proposed platform, an ambient temperature detection application and a CO_2 concentration sensor system are given as examples in this section. Users can develop more applications by combining different modules together. In the first example application, we use an I^2C semiconductor-based temperature sensing module, a processing module, and a power module, as shown in Fig. 8.

The main difficulty of this application is the sensor calibration. Since each temperature sensor in the sensing module has slightly inaccuracy, we have to individually calibrate each sensor module. Fig. 9 shows the instrument of temperature forcing chamber [12] used to perform the sensor calibration. The temperature sensor system is first setup in this temperature chamber. The corresponding sensor temperature can be recorded according to each target temperature set in the temperature chamber. In this way, we can therefore obtain the mapping table between temperature values of sensing module and the chamber.

Fig. 8. A temperature sensor system and its smart phone App.

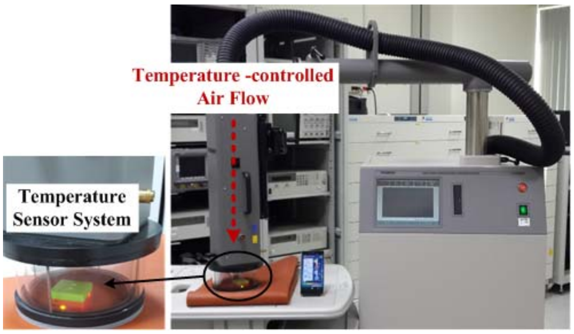

Fig. 9. Experimental environment of a temperature sensor system.

The relationship between the temperature in the sensing unit and the temperature in the chamber reading is shown in Fig. 10. The vertical axis indicates the target temperature set in the chamber, while the horizontal axis represents the corresponding temperature obtained from the sensing module in our platform. The temperature of each temperature sensor system can be modeled as a function of the temperature obtained in the chamber. The transfer function can be expressed by the following equation:

$$y = -0.0011x^2 + 1.1915x - 5.6884, \quad (1)$$

where y indicates the chamber readings (°C), and x represents the sensor readings (°C). The App can thus use the equation to quickly obtain a calibrated and accurate temperature. Fig. 11 shows the temperature errors before and after the calibration in different target temperature values. Before the calibration, large errors mostly fall on the low and high ends of the temperature, i.e., 0 ºC and 80 ºC, and the error can be up to 6 ºC. After the calibration, the temperature error is balanced in different target temperature values and the temperature error can be thus lower than 1.5 ºC.

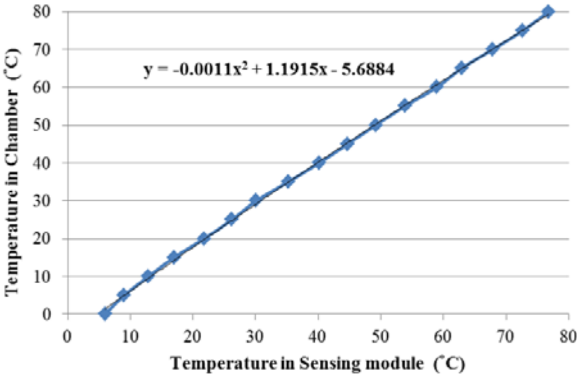

Fig. 10. Relationship between sensor readings and chamber readings with transfer equation.

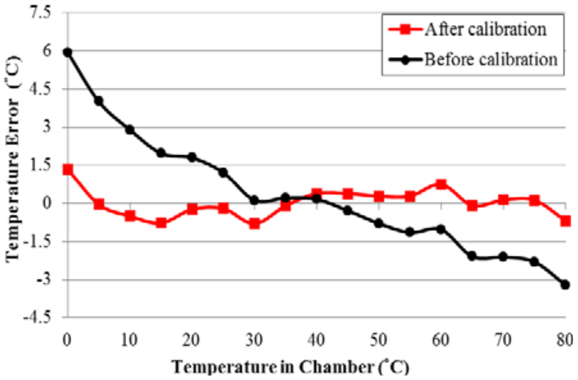

Fig. 11. Temperature errors in different target temperature values.

Most of the commercial sensors, like the temperature sensor just mentioned, have built-in readout circuits. In addition, they provide users with digital interfaces to easily configure the sensor or to retrieve the sensor data with an MCU. Furthermore, some even have automatic calibration mechanism that guarantees different sensor chips have similar output values. However, there are many analog interface sensors in the market, whose sensor signals are primitive and vary little in the form of current. We have adopted some sensors of this kind. To successfully retrieve the sensor value, we append a readout circuit that amplifies the output signal and transform it into the form of voltage for MCU's ADC controller to read. Besides, since the signals are primitive and not calibrated, the output will vary from chip to chip. We have to map the output voltage to a meaningful value for different chips.

We take the carbon dioxide (CO_2) sensor module for example to demonstrate how we calibrate this kind of sensors. To obtain the corresponding relationship between the sensor output voltage and its actual meaning, we build a simple experiment environment as shown in Fig. 12. We use sealed plastic chamber and CO_2 gas cylinder to create a test environment, where the concentration of CO_2 is controllable. We place both the CO_2 sensor system and a commercial CO_2 monitor [13] in this airtight plastic chamber. In addition, we attach the CO_2 gas cylinder to the plastic chamber with a plastic soft tube. To increase the concentration of the CO_2 in the chamber, we turn on the air valve on the cylinder to allow CO_2 gas flow into the chamber. From the voltage output of the CO_2 sensor and the reading of the CO_2 monitor, we can therefore get the relationship between the CO_2 sensor readings and the CO_2 concentration.

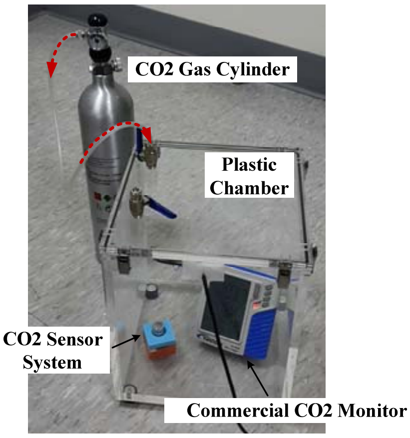

Fig. 12. Experiment environment of a CO_2 sensor system.

Each sensor needs to be calibrated individually due to the differences in their electrical characteristics. In Fig. 13, the vertical axis indicates the reading of the commercial CO_2 monitor, and the horizontal axis shows the corresponding voltage output of our CO_2 sensor module, green round nodes, red square nodes and blue triangle nodes are readings from three different sensors.

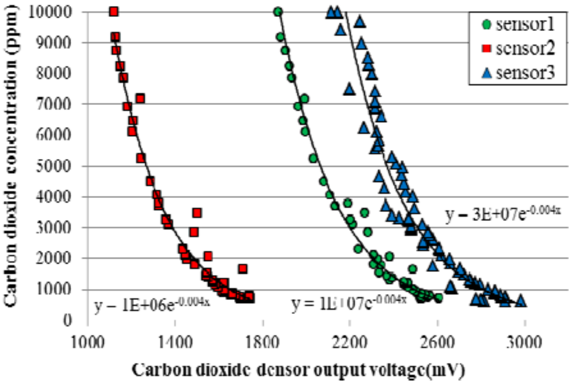

Fig. 13. Relationship between sensor readings and commercial CO_2 monitor.

It can be observed that the trends of three different sensors are similar, but there is a significant voltage bias. Each trend can be approximated with an equation, for example, the transfer equation of sensor1 is as follow:

$$y = 10^7 \cdot e^{-0.004x}, \qquad (2)$$

where y represents the CO_2 concentration (ppm) in the chamber, and x represents the voltage output (V) of the sensor. User can therefore derive the CO_2 concentration from the voltage output with this equation easily.

To avoid the tedious work that individually measures each sensor and finds out the transfer function for each one, we build a simple calibration methodology which makes use of the characteristic of the CO_2 sensor, as shown in Fig. 13. Obviously, different initial voltages lead to different transfer equations, where the initial voltage is the sensor output voltage under 1000 ppm CO_2 concentration. We try to collect data from as many sensors as possible, and find that we can classify those sensors into different categories based on their initial output voltage. Each category has one specific transfer function which can be built in the smart phone App. Once you know the initial voltage of a certain CO_2 sensor, you know which equation it fits. The App page is as Fig. 14 shows. For now, the atmospheric CO_2 concentration is about 410 ppm. It is recommended to turn on the CO_2 sensor system at a ventilated space, which is likely to have the CO_2 concentration under 1000 ppm, and click the switch icon on our App to read the current CO_2 output voltage to know which category this CO_2 sensor belongs to.

Fig. 14. A CO_2 sensor system and its smart phone App.

In this section, an ambient temperature detection application is adopted to show this sensor platform with a sensor identification scheme can work correctly. The calibration method for this temperature sensor system is also presented. After that, we take our CO2 sensor system to demonstrate how to calibrate an analog-type sensor. The firmware/hardware flexibility can also be observed from Fig. 8 and Fig. 14 that once the sensor brick is changed, the corresponding App page will pop-up automatically.

Until now, we have developed several kinds of sensor systems [14] and can rapidly integrate more kinds of sensors with this modular firmware/hardware framework. For each developed sensor system, the sensor calibration is performed and the corresponding transfer function according to the calibration result is embedded in the delivered Apps.

In the following, we compare 3 kinds of sensor systems in terms of their extensibilities, flexibility, identification methods, and cost. As shown in Table 2, our presented sensor platform owns better extensibility and flexibility benefited from the modular design. Besides, our platform also has lower cost feature while implementing a sensor identification scheme since a low-cost EEPROM only needs to be setup in the non-I^2C sensing modules.

Table 2. Comparisons of 3 kinds of sensor systems.

	[8]	[10]	This Work
Extensibility	Fair	Fair	Good
Flexibility	Fair	Fair	Good
Identification Method	TEDS	IMPI	EEPROM
Cost	Low	Fair	Low

5. Conclusion and Future Work

In this paper, we present a novel architecture of modular sensor platform. The most significant features of the proposed architectures are high flexibility and reusability. By stacking different modules together, users can create their own sensor system. In addition, a low-cost sensor identification scheme is also proposed to detect which sensor is mounted on the platform automatically. A temperature sensor system and a CO2 sensor system are demonstrated to show that our presented platform works correctly. The comparisons of 3 kinds of sensor systems are also given in this paper regarding cost, extensibility and flexibility. Currently, this platform has been licensed to an industrial company and has become a commercial product. Users' feedback shows that the platform is very useful especially for the purposes of academic researches and industrial prototype verification. In the future, we will focus on providing more modules per users' requirement.

References

[1]. C. Savaglio, G. Fortino, Mengchu Zhou, Towards interoperable, cognitive and autonomic IoT systems:

An agent-based approach, in *Proceedings of the IEEE Conference World Forum on Internet of Things (WF-IoT)*, Dec. 2016, pp. 58-63.

[2]. N. Matthys, *et al.*, µPnP-Mesh: The plug-and-play mesh network for the Internet of Things, in *Proceedings of the IEEE Conference 2nd World Forum on Internet of Things (WF-IoT)*, December 2015, pp. 311-315.

[3]. A. Al-Fuqaha, M. Guizani, M. Mohammadi, Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications, in *IEEE Communications Surveys & Tutorials*, Vol. 17, No. 4, June 2015, pp. 2347-2376.

[4]. J.-A. Stankovic, Research Directions for the Internet of Things, in *IEEE Internet of Things Journal*, Vol. 1, No. 1, March 2014, pp. 3-9.

[5]. C. Perera, A. Zaslavsky, P. Christen, Context Aware Computing for The Internet of Things: A Survey, in *IEEE Communications Surveys & Tutorials*, Vol. 16, No. 1, May 2013, pp. 414-454.

[6]. A. Spanbauer, A. Wehab, B. Hemond, I. Hunter, L. Jones, Measurement, instrumentation, control and analysis (MICA): A modular system of wireless sensor, in *Proceedings of the IEEE International Conference on Body Sensor Networks*, 2013, pp. 1-5.

[7]. Y.-M. Yusof, A.-M Islam, S. Baharum, An experimental study of WSN transmission power optimisation using MICAz motes, in *Proceedings of the IEEE International Conference on Advances in Electrical Engineering (ICAEE)*, 2015, pp. 182-185.

[8]. R. Morello, Use of TEDS to Improve Performances of Smart Biomedical Sensors and Instrumentation, *Sensors Journal IEEE*, Vol. 15, Issue 5, 2015, pp. 2497-2504.

[9]. A. Fatecha, J. Guevara, E. Vargas, Reconfigurable architecture for smart sensor node based on IEEE 1451 standard, *IEEE Sensors*, 2013, pp. 1-4.

[10]. K. Mikhaylov, M. Huttunen, Modular wireless sensor and Actuator Network Nodes with Plug-and-Play module connection, *IEEE Sensors*, Nov. 2014, pp. 470-473.

[11]. K. Mikhaylov, A. Paatelma, Enabling modular plug&play wireless sensor and actuator network nodes: Software architecture, *IEEE Sensors*, 2015, pp. 1-4.

[12]. The temperatue forcing system: Thermonics T-2500SE datasheet (https://www.atecorp.com/ATECorp/media/pdfs/data-sheets/Thermonics-T-2500SE_Datasheet.pdf)

[13]. The CO_2 monitor: Twintex TE-702D.

[14]. Y.-J. Hsieh, C.-C. Yang, Y.-J. Liu, W.-L. Lai, C.-M. Wu, C.-M. Huang, A Flexible Wireless Sensor Platform with an Auto Sensor Identification Scheme, in *Proceedings of the 11th International Conference on Sensor Technologies and Applications (SENSORCOMM' 2017)*, 10-14 September 2017, Rome, Italy, pp. 46-50.