A Plug&Play Scheme of GPIO Sensors/Actuators in Linux Platforms

Chi-Hwa Song
Department of Information Communication Engineering
Hannam University
Daejeon, Republic of Korea
sch@djtp.or.kr

Ji-Hoon Kyung
Department of Industrial & Management Engineering
Hannam University
Daejeon, Republic of Korea
kjh@hnu.kr

Seongbae Eun
Department of Information Communication Engineering
Hannam University
Daejeon, Republic of Korea
sbeun@hnu.kr

***Abstract*—Linux platforms such as Raspberry-Pi are widely used as development platforms for high-performance sensors/actuators in IoT application development. Application developers connect sensors or actuators to the GPIO port of Raspberry-Pi and write applications that access them. If the Plug&Play technique of GPIO sensors and actuators is provided, the difficulty of application development will be reduced. Existing Plug&Play techniques do not support Linux platforms, especially GPIO sensors/actuators. This paper proposes a method in which the device driver is automatically installed and played on Linux when the sensor/actuator is plugged into the GPIO port. By comparing and analyzing the method of this paper with the existing method by exemplifying the structure and device driver of the GPIO sensor, the method of this paper was shown to be more cost-effective.**

Keywords—Plug&Play, Raspberry-Pi, Linux, Sensor/Actuator, GPIO Port

I. INTRODUCTION

IoT technology has established itself as a core technology of the Fourth Industrial Revolution [1, 2], and various sensors/actuators are utilized. The development of IoT devices is to mount sensors or actuators on development platforms such as Arduino [3] and Raspberry-Pi [4] and write device drivers and applications for those sensors and actuators. At this time, it is difficult for software application developers to understand the hardware operation method of sensors and actuators.

In the case of Linux [5], when I/O devices are plugged, the driver of the device is mounted and played in the operating system, which can reduce the development burden on application developers. These Plug & Play technologies are also used in UPnP (Universal Plug & Play) [6] for office or home devices.

For example, the IEEE1451[7,8] standard is used as a plug&play technology for sensors/actuators. The technology is a method of plugging sensors or actuators into a standardized network and then playing immediately. ETRI and Hannam University's research team proposed a method [9-12] in which a pre-manufactured device driver is dynamically connected to the processor and played when an individual sensor or actuator is plugged into the AVR processor.

Based on the perception that existing Plug & Play techniques for sensors and actuators are inappropriate for application to the latest platforms like Arduino or Raspberry-Pi, we proposed a new approach [13,14]. This method is played by plugging the sensor/actuator into the USB of Raspberry-Pi. The USB method is easy to supply power and has a fast data transmission speed, but it has the disadvantage of being more complex and expensive than GPIO ports.

In this paper, we present a technique in which the device driver is automatically mounted and played on Linux over the Internet when the sensor and actuator are plugged into Raspberry-Pi's GPIO.

II. RELATED RESEARCH

A. Legacy Plug&Play Schemes

IEEE1451[7] is a standard led by NIST in the United States that focuses on overcoming the diversity of transducers. As a way to overcome the diversity of sensors and actuators, it stores a data structure called TEDS (Transducer Electronic Data Sheets) [8] in the memory of the NCAP (Network Capable Application Processor) device and operates based on it. This TEDS is an abstraction of the operation, restrictions, and calibration information of the sensor or actuator. The implication is that NCAP should be equipped with a standard stack that understands TEDS and performs protocols based on it.

The ETRI research team proposed the Sensor Plug & Play technique [9], which is a method of developing IoT applications on a standardized platform [10] based on ATmega128. Application developers can develop applications using standardized APIs [11] provided by the platform. Sensor providers can develop and supply device drivers according to standardized HW[12] and SW interfaces. In this way, since the features of the sensor/actuator are built into the driver as SW code, the abstraction data like IEEE1451 did not be requested.

B. USB Sensor/actuator Method

Figure 1 shows the overall configuration diagram of the USB-based Plug & Play system [13,14] which was proposed by us in the near past. The upper part of the figure is Raspberry-Pi and the lower part is the USB Sensor/Actuator Development Platform (USAP). USAP is connected to the USB port of Raspberry-Pi and performs Plug & Play functions.

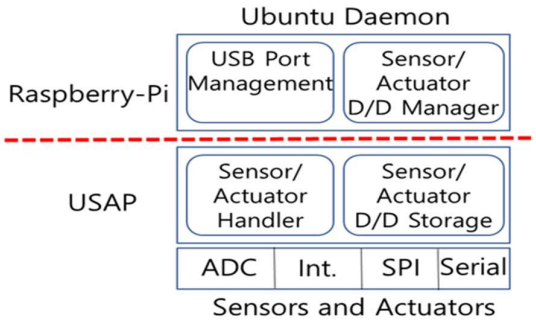

Fig. 1. Overall Architecture of USB Sensor/Actuator[14]

When a USAP-based sensor/actuator is plugged into the USB port of the Raspberry-Pi, the device driver stored inside the USAP are delivered to the Linux OS to play. This is accessed by Linux's standard I/O input/output API to produce applications.

The advantage of this method is that everything is don eautomatically tn the way of Plug&Play when the driver of the sensor/actuator is installed. However, the disadvantage is that the sensor/actuator requires an ATmega128-class MCU to store USB management chips and device drivers, which increases the price.

III. GPIO PLUG&PLAY

A. System Configuration

Figure 2 shows our Plug & Play system. In the figure, the USB sensor/actuator contains a USB management module, sensor D/D storage, etc. inside. On the other hand, sensors/actuators mounted on GPIO ports operate in a plug & play manner by mounting them on Linux without adding additional hardware.

At this time, it is different that the device driver is downloaded from the driver server through the Internet of the Linux platform. In addition, the GPIO sensor/actuator is a dummy sensor/actuator purchased in the off-the-shelf method, so it is a semi-automatic method because the user must enter the ID of the sensor to be installed.

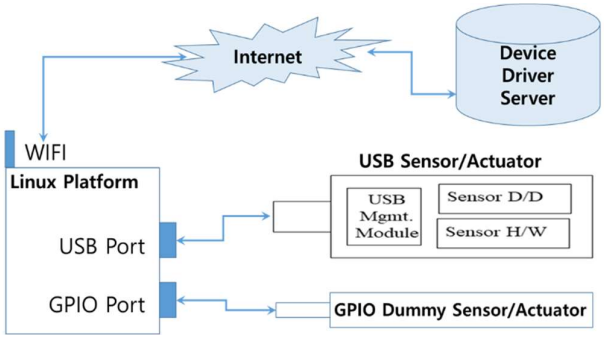

Fig. 2. Overall Architecture of Our Plug&Play Schemes

B. How the system works

A GPIO port is a digital signal line used as an input or output and its use is determined by the software. The GPIO method has the advantage of realizing low prices by attaching an off-the-shelf dummy sensor/actuator.

The problem is that dummy sensors/actuators do not contain MCUs or memory, so they cannot store device drivers or store the ID of the sensor/actuators.

In this paper, we propose the Semi Automatic method. Figure 3 shows the operation of the GPIO sensor/actuator-based plug & play method.

- step1. he user mounts the GPIO sensor/actuator to the Raspberry-Pi.
- step2. Plug&Play daemon receives an ID from the user and sends it to a P&P driver cloud server connected to the Internet.
- step3. The Plug&Play daemon interworks with the Plug&Play Driver Cloud Server to import the device drivers for its sensors/actuators.
- step4. Plug&Play Daemon inserts the driver to Linux.
- step5. The application uses standard input/output APIs such as Linux's open(), close(), read(), and write() to access its sensors/actuators.

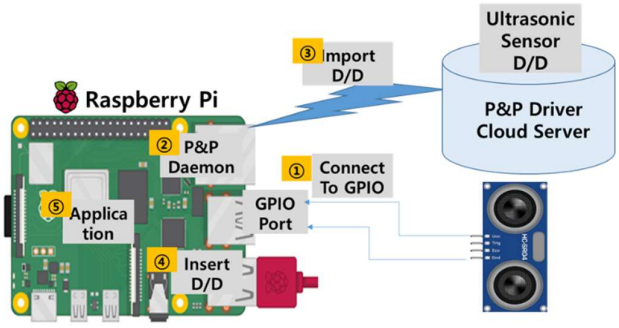

Fig. 3. Operating Procedure of the System

C. Implementation of the System

Figure 4 shows the configuration of the prototype. Left in Figure shows the connection between the Raspberry-Pi and the Ultra Sonic Sensor. Right) displays the performance results of the application. Using Linux's read() system call once a second, the distance value according to the change in the ultrasound value is read and output on the screen.

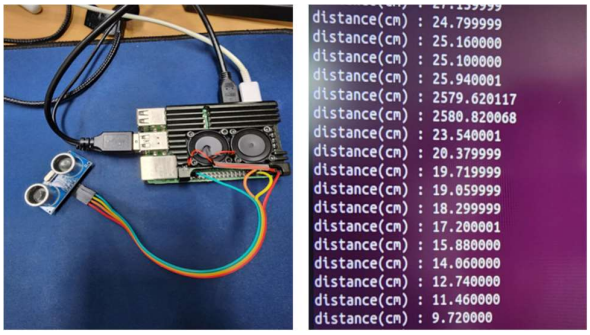

Fig.. 4 . Left) Ultra Sonic Sensor attached to Raspberry-Pi, Right) Monitor showing the result of Ultra Sonic Sensor

Left in Figure 5 shows the application. Applications can access sensors only with open().close(), read(). Right) shows the device drivers installed in the kernel. Basically, you can build it according to the device driver system of Linux.

```
dev = open("/dev/dht11_dev", O_RDWR);
  for(int i = 0; i < 100; i++) {
    read(dev, buff, 4);
    sleep(1);
  }
close(dev);
```

```
int dht11_read(int pin, dht11_data *_data) {
  . . . .
  return PARTY_ERR;
}
static int dht11_open(struct inode *inode,
struct file *filp) {
  if (gpio_direction_input(ECHO_PIN) != 0)
{  return -1;
      return 0;
}
```

Fig. 5 Left) Application Program, Right) Device Driver of Ultra Sonic Sensor

D. Comparative evaluation of USB and GPIO methods

Figure 6 shows the results of comparing, analyzing the costs of IEEE1451, ETRI method, and our USB method and GPIO method. In Figure, the horizontal line shows the size of the abstraction unit. The vertical line indicates the cost of the system cost. Stack cost refers to the stack cost according to the degree of standardization of each method. Application costs refer to development costs that vary from application to application. When the abstraction unit is 0, the stack development cost is 0, and it can be seen that the cost increases as the abstraction unit increases. Conversely, the cost of application decreases as the abstraction stage increases.

In the case of IEEE1451, which has a large abstraction unit, the application cost is small, but the stack cost is high, which increases the total development cost. Conversely, in the case of the ETRI method, the stack cost is small, but the application development cost increases, so the overall development cost increases as shown in the figure. Therefore, the USB style and GPIO style presented in this paper can be reduced as shown in the figure in terms of overall cost.

Cost analysis is currently in progress, and four parameters are derived from each method. 1) These include the number of lines of code in the stack, 2) the understanding of the stack, 3) the number of lines of code in the application, and 4) difficulties in application development. It is also being analyzed based on the software development cost table announced by the government.

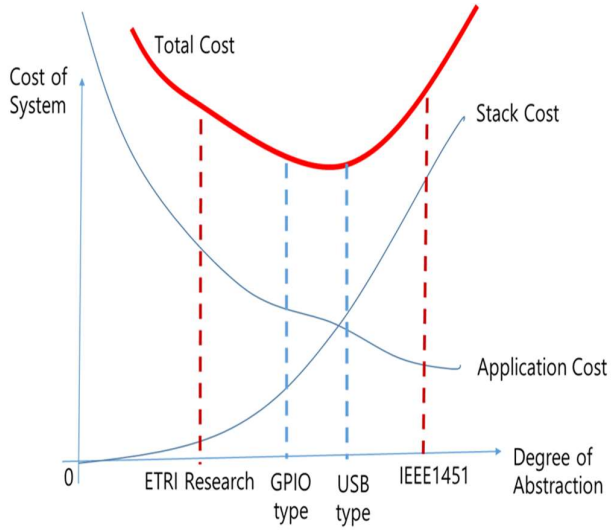

Fig. 6 Analysis of Plug&Play Schemes

IV. Conclusion and Future Research Direction

In this paper, a method of operating in a plug & play manner by installing a sensor/actuator in the GPIO port of Raspberry-Pi was presented. The sensor mounted on the GPIO port is a dummy sensor and cannot store device drivers or IDs, so the user enters it manually according to the request of the daemon. While the USB sensor proposed by our research team is Full Automatic, it is Semi Automatic, but it has the advantage of being inexpensive.

In addition, we analyzed the cost of application development based on the existing representative Plug&Play methods, the IEEE1451 method, and the ETRI method, and showed that our method could be mode cost-effective in terms of cost.

The direction of future research is to quantitatively analyze the cost of the four standardization methods.

References

[1] B. Jeong and M. Hong, "A Study on the Activation of IoT Industry Strategy and Policy in the 4th Industrial Revolution," International Commerce and Information Review, Vol. 21, No. 1, pp. 341–360, Mar. 2019.

[2] Y. Kim and D. Kim, "IoT Applications and Adaptation Examples," Journal of the KSME, Vol. 56, No. 2 , pp. 37–41, Feb. 2016.

[3] JeongBeom Song, Young Hwan Kim, Semin Kim," Development of Educational Contents for Making Traffic Lights Using Arduino," Proc. of The Korea Institute of Information and Communication Engineering, pp. 587-590, 2019.10.

[4] Hee-jun Kim, Hee-dae You, Jae-woo Chang,"Development of Realtime Pet Monitoring System by using Raspberry Pi," Proc. of KOREA INFORMATION SCIENCE SOCIETY, pp. 1543–1545, 2016.12.

[5] A. Rubini, Linux Device Drivers, O'Reilly & Associates, Inc., 1998.

[6] S.E. Lee, J.-W. Min, T. Kim, Y. Jun, and T. Yun, "Implementation of Home Network based on UPnP for Multimedia service," Proceedings of the Korea Information Processing Society Conference, May 2003, pp. 31–34.

[7] Institute of Electrical and Electronics Engineers, Inc., "IEEE Standard for Smart Transducer Interface for Sensors and Actuators – Network Capable Application Processor (NCAP) Information Model,"Mixed-Mobile Communication Working Group of the Technical Committee on Sensor Technology TC-9 of the IEEE Instrumentation and Measurement Society, June 1999.

[8] TC-9 Committee on Sensor Technology, "IEEE Standard for a Smart Transducer Interface for Sensors and Actuators-Transducer to microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats", 1997

[9] M. Yang, S. So, S. Eun, B. Kim, and J. Kim, "Sensos: A Sensor Node Operating System with a Device Management Scheme for Sensor Nodes," International Conference on Information Technology (ITNG'07), 2007, pp. 134–139.

[10] Plug&Play based USN Sensor Access Interface – Part1: Reference Model, Telecommunications Technology Association, 2012.

[11] Plug&Play based USN Sensor Access Interface – Part2: Physical Interface and Transmission Protocol between Sensor Node Platform and Sensor module, Telecommunications Technology Association, 2012.

[12] Plug&Play based USN Sensor Access Interface – Part3: HAL Library for Sensor Device Driver, Telecommunications Technology Association, 2012.

[13] S. Lee, S. Woo. S. Cha, G. Jeong, S Eun, and S. So,"The Plug&Play of Sensor/Actuator Modules in Raspberry-Pi," Proc. of ICGHIT2022. pp.218–220, Jan. 2022.

[14] C. Song, J. Park, S. So, and S. Eun,"Abstraction Granularity of Sensors/Actuators," Proc. of 51st Conference of KIICE, pp. 94–96, May 2022.