Simple-API (Application Programming Interface) for an Arduino-based Wireless Sensor Mote

Thiti Sittivangkul*, Weerasak Cheunta**, Nitthita Chirdchoo**, and Lunchakorn Wuttisittikulkij*
Department of Electrical Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand
**Wireless Sensor Network and Embedded System Research Unit, Nakhon Pathom Rajabhat University, Nakhon Pathom, Thailand
thiti_sit@me.com, {weerasak, nitthita}@webmail.npru.ac.th and lunchakorn.ww@chula.ac.th

Abstract

In this paper, Simple-API for an Arduino-based platform sensor mote is proposed. It is designed especially to reduce the complexity and the difficulty in dealing with MCU hardware register, interface driver, hardware register and device driver, with which the wireless sensor network (WSN) application developers need to involve during software development. Simple-API allows developers to focus more on algorithm design and coding only at the application level, thus providing a more convenient and faster means in developing an application for WSNs.

When comparing to the commercially popular Waspmote platform, it is obvious that coding using Simple-API requires less effort in coding by minimizing the complexity in dealing with hardware level coding.

Keywords: Wireless sensor network, sensor node, software development, API, microcontroller, coding

1. Introduction

Recently, wireless sensor networks (WSNs) have found their share in many real-life applications such as smart farming, environment monitoring, smart home automation and many more. To be able to form a WSN for any application, a number of small devices equipped with a microcontroller, communication modules and multiple sensors, must be deployed [1]. These small devices, or a sensor mote, are capable of sensing physical phenomenal, processing and wirelessly transmitting/receiving data to/from a remote location. A sensor mote is usually designed to be small-size and lightweight, causing it to be resource-constraint, in terms of available battery power, processing capability and memory capacity [2]. With these limitations, it is highly challenging when designing and deploying these sensor motes into a WSN for real-life applications.

Although there are many different types of sensor mote currently available in the market today, these motes normally share similar hardware composition and functions. As shown in Fig. 1, mote's composition can generally be grouped, according to the function it performs, into four main sub-systems [3]. These include (1) sensing, (2) processing, (3) empowering and (4) communicating, sub-systems. Each of which are responsible for data acquisition, data processing, empowering the motes and communication, respectively. These functions are also applied when dealing with software development for a sensor mote.

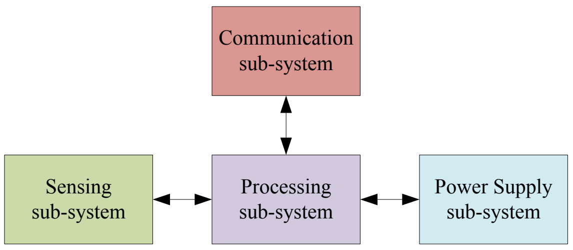

Figure 1: Software category

In software development, the function of a mote will be programed by a software developer or a programmer, typically using C or C++ programming language. During this process (shown in Fig. 2), the developer requires to have not only programming skill but also a good background and understanding in a hardware level coding, in order to deal with microcontroller register, interface driver, hardware register and device driver. Unfortunately, most application developers are often not familiar with hardware architecture of a microcontroller, making it

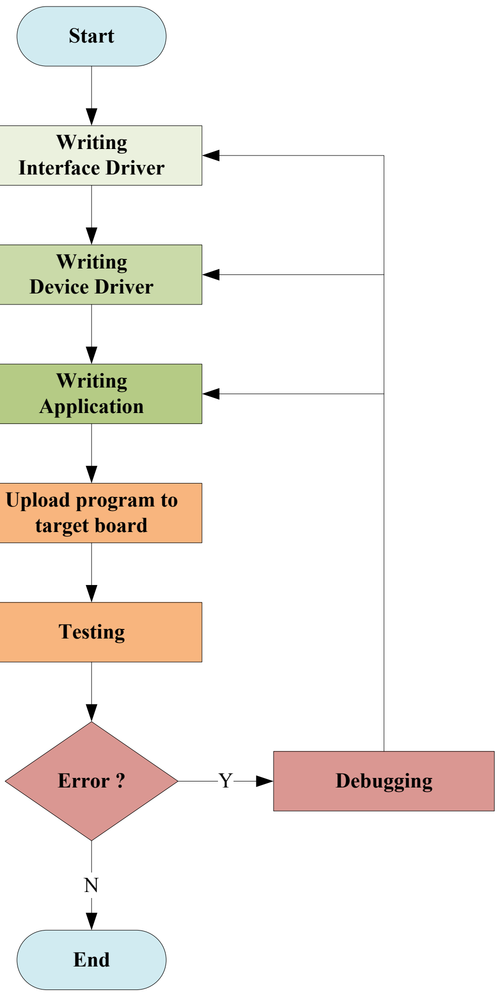

Figure 2: Processes when dealing with an application development for a wireless sensor network.

highly difficult in both coding and debugging process. This consequently slows down the application development process.

To alleviate the need of manually hardware-level coding by an application developer, Arduino platform [4] is introduced. It is an open-source platform that includes both hardware and software architectures. Moreover, it is designed especially for those users who has minimal or no experience in dealing with microcontrollers [5]. Since the introduction of Arduino platform, it has continually gained attentions from many WSN application developers, due to its capability to seal away the difficulty in coding at the hardware level.

In detail, Arduino offers a more convenient way in software development because it provides common device drivers and interfaces drivers. As a result, users do not need to learn or have any experience about microcontroller registers and how to write source code for common interfaces such as digital I/O, analog to digital converter, serial interface, SPI interface and I^2C interface. However, when creating an application with Arduino, developers are still required to have a good understanding on how to interface with external devices (sensors, radio modules etc.) and also a step-by-step of how each operation must be processed.

A recent advancement in terms of simplicity in application development for a WSN can be found in Waspmote [6] platform, which is commercially launched by Libelium. Specifically, Waspmote is also an Arduino-based sensor mote which integrates microcontroller, wireless communication modules and sensors onto a single circuit board. It also provides an open-source software environment that offers common device drivers and interface drivers, similarly to those found in Arduino platform. The main differences between the two are that Waspmote not only support the interfaces and the device drivers required by the microcontroller, but also provides the interfaces and the device drivers for communicating with external devices such as sensors and radio modules. With these add-ons, Waspmote offers a more user-friendliness and convenient means in WSN application development.

Although Waspmote provides device drivers for both the microcontroller and the external devices, developers are still required to understand the device manual and datasheet, in order to perform accurate order and suitable initialization and configuration processes, in order to create a WSN application. Moreover, function calls in Waspmote are non-trivial, requiring developers to specifically study all the function names and call sequences before being able to utilize the devices.

In this paper, Simple-API for an Arduino-based platform sensor mote is proposed. It is designed especially to reduce the complexity and the difficulty in dealing with MCU hardware register, interface driver, hardware register and device driver, with which WSN application developers need to involve during software development. Simple-API allows developers to focus more on algorithm design and coding only at the application level, thus providing a more convenient and faster means in developing an application for WSNs.

The rest of this paper is organized as follows. In Section 2, we provide a discussion of the design of the proposed Simple-API. Next, in Section 3, the gain in terms of a number of lines in coding when developing an application in both Simple-API and Waspmote are illustrated. We finally conclude our work in Section 4.

2. Simple-API

In order to help developers to accelerate the time required to develop an application on an Arduino-based sensor mote, we develop a set of APIs, namely, “Simple-API”. Specifically, as shown in Fig. 3, simple-API provides a convenient way for developers by sealing away the need of understanding the microcontroller architecture and other hardware-level issues, including the MCU hardware register, the interface driver, the hardware register and the device driver from the software developers' perspective. By doing so, Simple-API allows an application developer to be able to focus more on algorithm design and coding at the application level (e.g., when a packet should be sent, which operating mode should the sensor mote be switched to, in order to minimize battery consumption, etc.), resulting in a more convenient and a faster WSN application development.

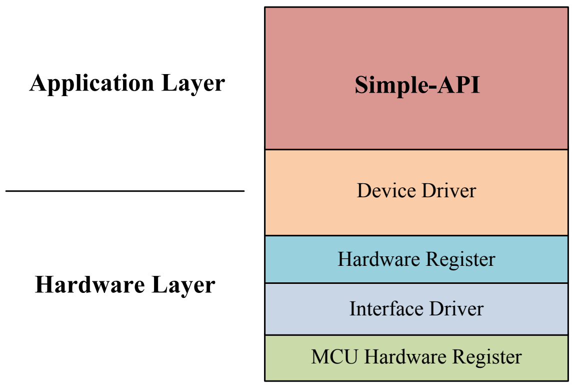

Figure 3: Simple-API seals away of the difficulty in coding relating to MCU hardware register, interface driver, hardware register and device driver.

SensorDevice

```
init(SensorName, Interface);
setParam(SensorName, ParamName, Value);
getParam(SensorName, ParamName);
read(SensorName, ValueName);
```

CommunicationDevice

```
init(ModuleName, Interface);
setParam(ModuleName, ParamName, Value);
getParam(ModuleName, ParamName);
sendData(ModuleName, Destination, Msg);
readData(ModuleName, Msg, Msg_Type);
```

Processing

```
init();
setTime(Time_Value);
getTime(*Time_Value);
setAlarm(Time_Value, Alarm_Num);
isAlarm(Alarm_Num);
setOperationMode(Mode);
```

PowerSupply

```
getBatteryVoltage();
getBattery();
setPower(Slot_Num, Mode);
```

Figure 4: Example of functions, listed according to the mote architecture sub-systems.

In detail, Simple-API includes 4 sets of function calls, corresponding to the 4 sub-systems (sensing, communicating, processing and empowering) of the sensor mote hardware architecture. Fig. 4 illustrates a list of functions, provided by Simple-API, which can be called to support each sub-system. For example, let us look at the functions available for sensing sub-system (see SensorDevice in Fig. 4). When the developer calls function named init() to initialize the certain type of sensor (e.g., temperature, pH or dissolved Oxygen) that is currently of interest via certain interface (e.g., I^2C or SPI interface), Simple-API is responsible for interfacing in MCU hardware register level and then starts to communicate with the sensor, in order to provide initial setting and configuration. Moreover, when a function in Simple-API is invoked, it will create an instance similarly to the instance in found in typical C++ programming, making it highly recognizable and user-friendly for most programmers.

3. Result

Fig. 5 shows an example of how Simple-API provides a more convenient means in developing an application in WSN. According to the example, to broadcast a packet in Waspmote environment via XBEE module, it is important to ensure that XBEE module must be initialized and configured correctly before specifically indicating the destination node ID, data packet to be sent and MAC protocol.

```
 // Set params to send
paq_sent=(packetXBee*) calloc(1,sizeof(packetXBee));
paq_sent->mode=BROADCAST;
paq_sent->MY_known=0;
paq_sent->opt=0;
xbeeDM.hops=0;
xbeeDM.setOriginParams(paq_sent, "5678", MY_TYPE);
xbeeDM.setDestinationParams(paq_sent,
"0013A200404873F1", data, MAC_TYPE,
DATA_ABSOLUTE);
// Send command
xbeeDM.sendXBee(paq_sent);
```

(a) Waspmote environment.

```
 // Set params to send
CommSystem.setParam(nRF24, MAC_TYPE, ALOHA);
CommSystem.setParam(nRF24, MSG_TYPE,
BROADCAST);
dest = "0013A200404873F1";
// Send command
CommSystem.sendData(nRF24, dest, data);
```

(b) Simple-API environment

Figure 5: Comparison in the total of number of function calls needed to broadcast a single packet between Waspmote and Simple-API environment.

These processes are coded by a software developer. For Simple-API environment, to perform the same task, it requires smaller number of function calls. Actually, the function calls needed in Simple-API is considered minimal since it only requires the information related to the application level as needed, such as communication module, MAC protocol, type of transmission, destination node ID. The rest of the processes are all handled by Simple-API automatically.

4. Conclusion

In this paper, we proposed Simple-API for an Arduino-based platform sensor mote. It is designed especially to reduce the complexity and difficulty in dealing with MCU hardware register, interface driver, hardware register and device driver, with which the WSN application developers need to involve during software development. This allows an application developer to be able to focus more on algorithm design and coding only at the application level, thus providing a more convenient means in developing an application for WSNs. Simple-API provides four sets of functions. Each of which is designed to support the task of sensing, processing, communicating and empowering of a sensor mote.

When comparing to Waspmote environment, it is obvious that coding with Simple-API requires less coding and lower complexity in dealing with hardware level coding.

References

[1] I. F. Akyildiz, T. Melodia and K. R. Chowdhury, “A survey on wireless multimedia sensor networks,” IEEE on Wireless Communications, vol. 14, no. 6, pp. 32-39, Dec. 2007.
[2] M. A. Razzaque, C. Bleakley and S. Dobson, “Compression in wireless sensor networks: A survey and comparative evaluation,” ACM Transactions on Sensor Network (TOSN), vol. 10, no. 1, pp. 537-568, Nov. 2013.
[3] T. Sittivangkul, W. Cheunta, N. Chirdchoo, M. Saadi and L. Wuttisittikulkij, “Design and development of a wireless sensor mote prototype for laboratory usage”, ITC-CSCC 2014, pp. 678-681, Jul., 2014.
[4] Arduino getting start [Online]. Available: http://ardruino.cc/en/Guide/HomePage.
[5] M. Margolis, “Chapter 1 Getting start”, Arduino cookbook, 2nd Ed., O’Reilly, pp. 1-3, 2012.
[6] Waspmote [Online]. Available: http://www.libelium.com/development/waspmote/documentation/waspmote-datasheet/