ASTES

Advances in Science, Technology and Engineering Systems Journal Vol. 5, No. 5, 295-301 (2020)

www.astesj.com

ASTESJ

ISSN: 2415-6698

Arduino-Compatible Modular Kit Design and Implementation for Programming Education

Gyeongyong Heo*

Department of Electronic Engineering, Dong-eui University, Busan, 47340, Korea

A R T I C L E I N F O

Article history:
Received: 22 July, 2020
Accepted: 11 August, 2020
Online: 17 September, 2020

Keywords:
Arduino
Programming Education
Modular Design
FRUTO Kit

A B S T R A C T

To cultivate creative talent, ways to learn creative problem-solving skills is needed, and one of them is programming. Arduino is a well-known tool used for programming education and the usefulness has been demonstrated in various case studies. However, there are several problems in existing Arduino-compatible kits as education tools, including the need for understanding hardware and the difficulty of expanding the kits with third-party hardware. In this paper, the design of an Arduino-compatible modular kit, called as FRUTO, is proposed that can be easily connected and conveniently programmed to overcome the problems. The structure and features of the FRUTO kit that implements the proposed design are also shown. The FRUTO kit consists of the FRUTO module that uses a unified connector for easy and intuitive connection and the FRUTO library that abstracts hardware-dependent code for easy programming. The FRUTO kit is easier to use and more scalable than existing kits. Even more, it can be used in various ways depending on the students' familiarity with hardware and programming. These strengths will make the kit to be an appropriate tool for various microcontroller-related education as well as programming education.

1. Introduction

In modern society, programming goes beyond creating programs that run on computers to include interacting with everyday environments through computers. This concept of programming is more important than ever in the context of the fourth industrial revolution and the internet of things. The importance of programming education is also evident in the inclusion of programming in curriculum, beginning in the United Kingdom in 2014. In Korea, programming education has been implemented gradually since 2018 following the curriculum revision in 2015 [1]. Accordingly, information course was designated as compulsory and physical computing was included as part of information course. Physical computing, designed to teach interactive design at Interactive Telecommunications Program at New York University, means building interactive physical systems by the use of software and hardware that can sense and respond to analog world [2], which can be extended to a creative framework for understanding human beings' relationship to digital world. The introduction of physical computing, not just computer-based programming, in information course reflects the assessment that programming education helps to develop problem-solving skills and improve logical thinking. The findings that hardware education has a positive impact on students' interest, motivation and learning attitudes also influenced this decision [3-4]. Physical computing can be beneficial for students who are not necessarily interested in pursuing computer programming but would like to gain a better understanding of technology and how it is shaping our world[5].

Arduino is one of the representative tools used for programming education[6-8]. Arduino began as an open-source microcontroller project for artists and has become one of the leading microcontroller projects for its ease of use. Arduino has been widely adopted as an educational tool in elementary, middle and high schools as well as universities, and various educational effects using Arduino have been reported [9-10]. In addition, various studies on the development of Arduino-compatible educational tools are also in progress [11-13]. However, one of the problems in using Arduino as an educational tool is that it requires knowledge of hardware. Besides, existing Arduino-compatible kits are designed without consideration of various applications and are limited in their use in ways that are not considered in design phase. These problems are from the fact that existing kits were designed for Arduino itself. The kit for programming education should be considered as a tool.

*Corresponding Author: Gyeongyong Heo, Department of Electronic Engineering, Dong-eui University, Busan, 47340 Korea, +82-51-890-1673, hgycap@deu.ac.kr

https://dx.doi.org/10.25046/aj050537

In this paper, the design of a modular Arduino-compatible kit, called as FRUTO, is proposed to overcome the problems mentioned, and its structure and features are shown that implement the proposed design. On the hardware side, the connection between modules is made easy and intuitive with a unified connector, while on the software side, the code for controlling modules uses common abstract functions to facilitate easy programming, which results in improved usability. The proposed design also improved scalability by allowing anyone to produce hardware and software modules using the proposed design. As a result, improved usability and scalability make the FRUTO kit a learning tool for students of various levels. The proposed design was recognized for its differentiation through patent registration [14-15], and the FRUTO kit that implements the design is undergoing a pre-launch test.

In the next section, Arduino-compatible kits used in programming education and their limitations are presented. Section 3 is devoted to propose a new kit design that solves the problems of existing kits, and section 4 describes the FRUTO kit that meets the proposed design. Conclusions and directions for further improvements are discussed in section 5.

2. Arduino as a Tool for Programming Education

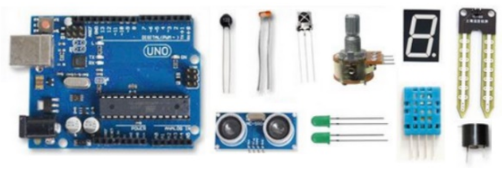
2.1. Programming Education using Arduino

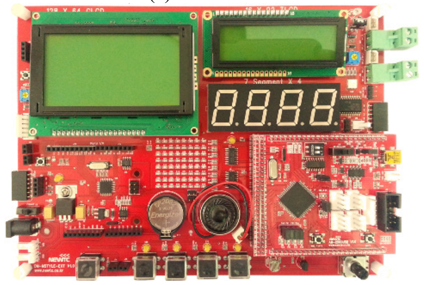
As a leading physical computing platform, Arduino has been widely used for programming education as well as physical computing. In particular, Arduino is drawing attention as an alternative to overcome the limitations of the traditional programming education, as it places importance on the process of ‘thinking by hand’ through the prototyping process [16]. The use of Arduino for physical computing and programming education comes from the fact that it is highly accessible and usable as it is designed as a platform for artists and designers. The fact that Arduino is suitable for project-based learning (PBL) is another reason why Arduino is used as a tool for programming education. PBL has advantages such as getting structured and integrated knowledge for problem-solving beyond fragmentary knowledge and fostering active attitudes and confidence by searching for the knowledge required in problem-solving. The usefulness of PBL using Arduino in engineering subjects as well as programming education has been demonstrated in various case studies [17-19].

2.2. Problems with Existing Kits in Hardware

There are several Arduino-compatible kits used for programming education, which can be divided into three groups: (1) Arduino kit using the original Arduino board, (2) Arduino-compatible all-in-one board with built-in I/O devices, (3) Arduino-compatible kit with modular I/O devices and a unified connector.

The fundamental problem in using the original Arduino board as an educational tool is that students need to understand hardware. Since most I/O devices use different numbers of I/O pins and different data exchange methods, it is necessary to understand the hardware specification of each I/O device to obtain the desired operation. One way to solve the hardware dependency is to use an all-in-one board that contains all the necessary I/O devices as shown in Fig. 1-(b). All-in-one boards do not require a deep understanding of hardware, which can be seen as an advantage. On the other hand, it is difficult to use in ways that are not considered in the design, that is, it is poor in scalability. Moreover, programming education using an all-in-one board is almost the same with the conventional computer-only education.


(a) Arduino kit


(b) Arduino-compatible all-in-one board with built-in I/O devices

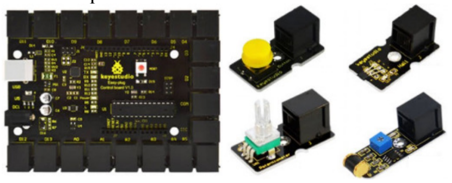

(c) Arduino-compatible kit with modular I/O devices

Figure 1: Arduino-compatible kits for programming education

The modular kit in Fig. 1-(c) uses a unified connector to make the connection easy. However, only simple I/O devices can be made in modules, and the position where the module can be connected to the main module is fixed. In addition, the control method varies from module to module in spite of the unified connector.

Table 1: Evaluation of existing Arduino-compatible kits

Type	Diversity	Connectivity	Independency
Arduino kit	○	×	×
All-in-one board	×	△	○
Modular kit	△	△	△

Table 1 summarizes the evaluation of existing kits based on the number of available I/O devices (Diversity), the degree of easy connection (Connectivity), and the degree of hardware-related knowledge required (Independency). Of the three types, modular kits can be used reasonably in all respects. However, modularity in software is not considered and hardware modularity is limited to some simple I/O devices. Therefore, there is a need for a new design that all modules can be connected easily and intuitively using a unified connector.

2.3. Problems with Existing Kits in Software

Two programming tools are mostly used to make programs for existing kits: Arduino and Scratch. Arduino uses C/C++ language to write a program, while scratch uses drag-and-drop blocks to build a program visually. It is not easy, however, to make complex code with the blocks given and the assembled blocks should be converted to C/C++ code before compilation. Therefore, only

C/C++ language under Arduino environment is considered in this paper.

Writing a program using C/C++ language under Arduino environment also has a similar problem with that on the hardware side. Since different I/O devices have different I/O control methods, it is also necessary to understand hardware to write the code for a specific I/O device. Most existing kits are based on Arduino environment. The Arduino library already provides an abstraction of low-level functions for AVR microcontrollers in Arduino boards, which makes it possible for different AVR microcontrollers to control a specific I/O device in the same way. That is, Arduino provides a microcontroller-independent library. However, as I/O devices are still controlled in different ways, the Arduino library is still hardware-dependent. Therefore, it is necessary to ensure hardware independence through a library that can control I/O devices with common abstract functions. This library should be based on the Arduino library for compatibility with Arduino.

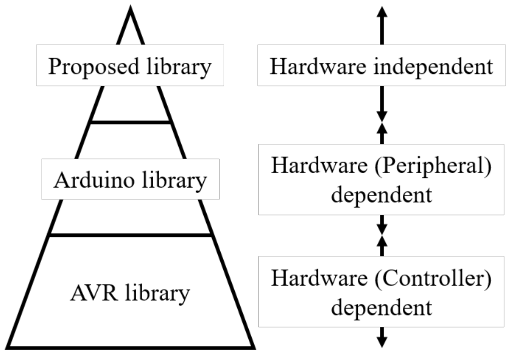

Proposed library
Arduino library
AVR library
Hardware independent
Hardware (Peripheral) dependent
Hardware (Controller) dependent

Figure 2: Hierarchical library structure

3. Proposed Kit Design

3.1. Requirements

The most important problem to be solved in a new design is the dependence on hardware. Most of the existing kits are for learning Arduino, not for learning programming or physical computing. As a kit for programming education should reduce hardware dependency as much as possible, the kit design proposed here has the following goals:

- a modular kit with easy and intuitive connection using a unified connector
- hardware-independent abstract functions to exchange data between modules
- scalability by adding third party or DIY(Do It Yourself) modules and corresponding libraries
- applicability in various microcontroller-related education

To achieve these goals, the followings are applied to the design.

- There are two kinds of modules: main module and expansion module with an I/O device.
- The Modules form a cascading connection with I2C(Inter-Integrated Circuit).
- The main module is compatible with Arduino and has one dedicated I2C connector for connecting an expansion module.
- Each expansion module includes a microcontroller that controls the I/O device and communicates with the main module.
- The microcontroller in each expansion module is one of the microcontrollers used in Arduino boards for compatibility.
- The expansion module has two dedicated I2C connectors to support cascading connection.
- The program for the main module consists of the code for I2C communication and system logic for overall system control.
- The program for the expansion module consists of the code for I2C communication and the code for I/O device control.
- Hardware-dependent code for I2C communication and I/O device control is provided through a dedicated library.
- I2C communication code in the main module uses common abstract functions in the library across all expansion modules.
- The dedicated library is based on the Arduino library to maintain the compatibility with Arduino.
- The module design is based on the published Arduino design and the proposed design is also open source.
- The dedicated library is also open source.

The core of this design and implementation is in the modularization of hardware and software. Hardware modularization enables easy and intuitive connection and software modularization allows students to focus on hardware-independent system logic.

All modules are compatible with Arduino, so the kit can be used in a variety of configurations, such as main and expansion modules together, main module only, and expansion module only. These configurations can be selected according to the student's prior knowledge of hardware and software, and used in various microcontroller-related education.

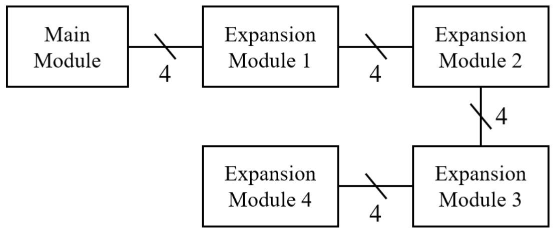


Figure 3: Cascading module connection

3.2. Hardware Design

The key in the proposed design is to enable cascading connection using a unified connector. In order to enable cascading connection, a communication method that can share a serial connection line and a dedicated controller for communication are required. There are many ways to share the serial connection line, and one of them is I2C communication[20], which is supported by Arduino by default. I2C communication allows cascading connection with four lines. In I2C communication, the main module acts as a master and the expansion modules act as slaves.

Fig. 3 shows an example of module cascading using I2C connection.

The controller in the expansion module is responsible for controlling I/O devices as well as I2C communication. The proposed design adopts ATmega328 as the microcontroller for all modules to maintain compatibility with Arduino UNO. Fig. 4 shows the basic configuration of the main and expansion module in the proposed design. The main module has one I2C connector as a starting point, and the expansion module has two I2C connectors to support cascading connection.

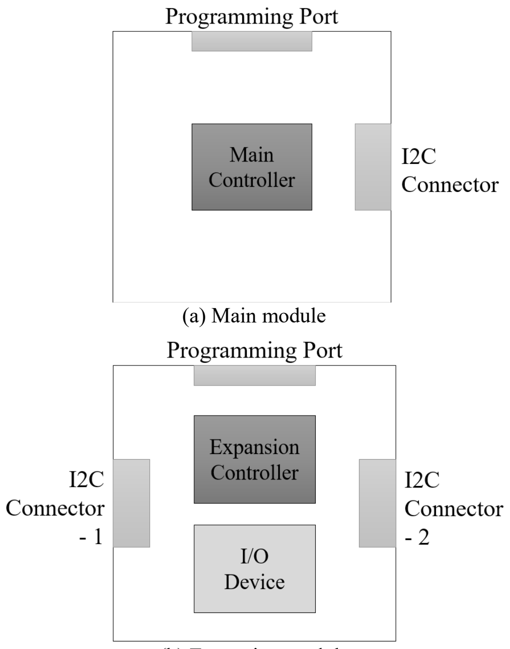


Figure 4: Module design

3.3. Software Design

In the proposed design, the library is also modular. The Arduino library is an abstraction of low-level microcontroller functions. However, the Arduino library is hardware dependent as it requires the understanding of I/O devices connected to an Arduino board. Therefore, the proposed design reduces hardware dependency by separating hardware-dependent and hardware-independent code.

Consider a system that uses a main module, a button module, and an LED module to represent button states to LEDs. In this configuration, three ATmega328s are used, and three programs for each ATmega328 are required. The button module requires a program to read the status of the buttons and send it to the main module through I2C communication, while the LED module requires a program to represent the data received from the main module to LEDs. That is, the program for the expansion module includes hardware-dependent code for controlling I/O devices and I2C communication. On the other hand, the program for the main module contains the code for I2C communication with the expansion modules and the hardware-independent code for controlling the overall system, called system logic. The emphasis in programming education should be on hardware-independent system logic.

In the proposed design, the library consists of a main (or master) library and expansion (or slave) libraries. The expansion library has I2C communication and I/O device control functions, and is pre-installed in the expansion module. The main library, on the other hand, provides common abstract functions for exchanging data through I2C communication with expansion modules. The main library is compiled with system logic and installed in the main module.

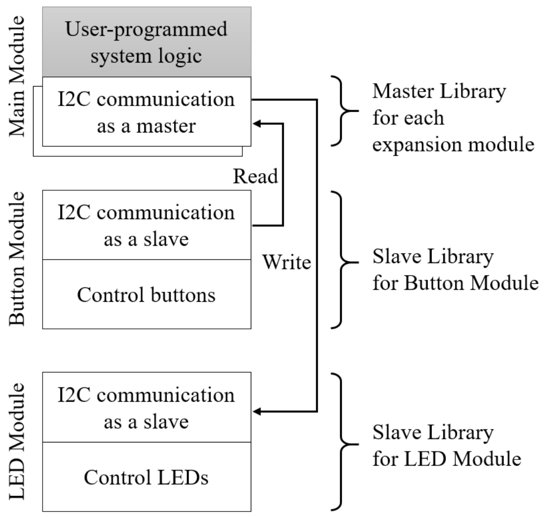

Main Module
User-programmed system logic
I2C communication as a master
Master Library for each expansion module
Read
Button Module
I2C communication as a slave
Write
Control buttons
Slave Library for Button Module
LED Module
I2C communication as a slave
Control LEDs
Slave Library for LED Module

Figure 5: Program structure

In Fig. 5, each library contains hardware-dependent code, and system logic is hardware-independent code that is the only one students must write. Hardware-independent system logic allows students to develop logical thinking and problem-solving skills that are the goal of programming education.

4. FRUTO Kit

The FRUTO kit is a kit that satisfies the guide and consists of the FRUTO modules corresponding to hardware and the FRUTO library corresponding to software.

4.1. FRUTO Module

The FRUTO module can be a main module or an expansion module, and ATmega328 is used as the controller for each module. The main module follows the published design of Arduino UNO and has an additional I2C connector for cascading connection. On the other hand, an expansion module is composed of an expansion controller, I/O devices, and I2C connectors.

Fig. 6 shows the circuit diagram of an LED module. Unlike the main module, the expansion controller uses minimal circuit in the Arduino UNO design. Therefore the expansion module cannot fully operate as an Arduino UNO compatible board. However, in most cases, the expansion module uses the pre-installed program, and a custom program can be installed with an external programmer.

The modules are connected directly to each other by default, but a dedicated cable and a distribution module is also included in the FRUTO kit to enable flexible connection. Fig. 7 shows an example of connecting 4 modules using a distribution module and a dedicated cable.

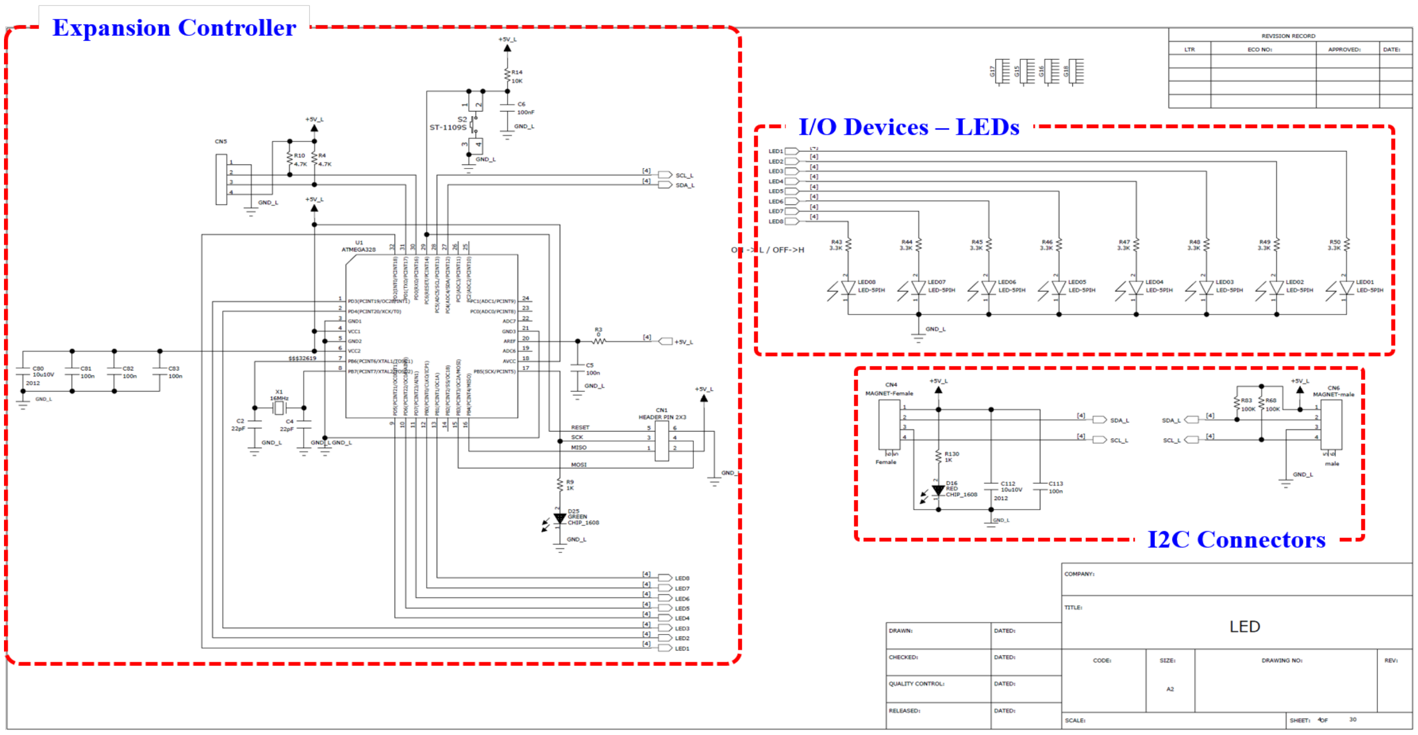


Figure 6: LED module circuit diagram

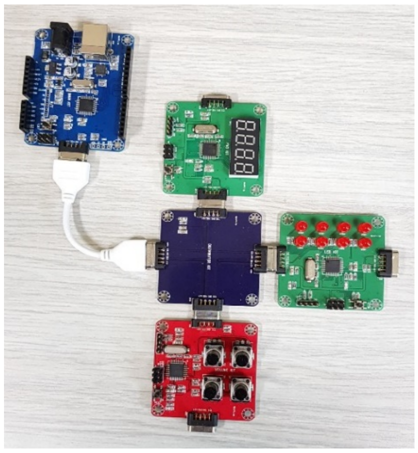


Figure 7: FRUTO module connection

4.2. FRUTO Library

As shown in Fig. 5 the programs needed for the FRUTO kit are a library for a master, libraries for slaves, and a main Arduino sketch. The master and slave libraries are provided as part of the FRUTO library, and students only need to make a master sketch for a main module. Fig. 8 shows a blink sketch that blinks the LEDs on the LED module at 1-second intervals.

```
#include <FRUTO.h>
void setup() {
    FRUTO.begin();
}
void loop() {
    Module_LED.on();                    // turn on 8 LEDs
    delay(1000);
    Module_LED.off();                   // turn off 8 LEDs
    delay(1000);
}
```

Figure 8: Master sketch

Fig. 8 does not look very different from Arduino's blink sketch. However, the FRUTO library is characterized by the fact that hardware-independent ‘on’ and ‘off’ are used instead of hardware-dependent ‘digitalWrite’, and that the ‘on’ and ‘off’ are based on the common abstract function ‘write’.

```
#include "Module_LED.h"
void _Module_LED::on(void){             // turn on 8 LEDs
    write(0xFF);
}
void _Module_LED::off(void){            // turn off 8 LEDs
    write(0x00);
}
void _Module_LED::write(byte value){
    Wire.beginTransmission(MODULE_LED);
    Wire.write(value);
    Wire.endTransmission();
}
```

Figure 9: Master library for an LED module – Module_LED.cpp

```
#ifndef _MODULE_LED_
#define _MODULE_LED_
#include "Arduino.h"
#include <Wire.h>
#define MODULE_LED 11                   // I2C address
class _Module_LED {
    public:
        void on(void);                  // turn on 8 LEDs
        void off(void);                 // turn off 8 LEDs
        void write(byte value);         // send LED data
};
extern _Module_LED Module_LED;          // LED module instance

#endif
```

Figure 10: Master library for an LED module – Module_LED.h

The master library in Fig. 9 and 10 is a library defined differently according to each expansion module. Whereas the operations common to all modules are defined in the system library. Typical common operations included are to initiate I2C communication and to initialize each expansion module.

```
#include "FRUTO.h"

void _FRUTO::begin(void){
    Wire.begin();                         // start I2C as a master
}
```

Figure 11: FRUTO system library – FRUTO.cpp

```
#ifndef _FRUTO_
#define _FRUTO_

#include "Arduino.h"
#include <Module_LED.h>                   // initialize a module

class _FRUTO {
    public:
        void begin(void);
};

extern _FRUTO FRUTO;
#endif
```

Figure 12: FRUTO system library – FRUTO.h

Data sent to the expansion module using the master library is received and processed by the LED module. Fig. 13 and 14 show the slave library for an LED module, which receives one byte data and controls 8 LEDs accordingly. Since the slave library is the same as the sketch for Arduino, it is also possible to learn Arduino through the slave library. This can also be seen from the fact that the slave library has an INO extension.

```
#include <Wire.h>
#define MODULE_LED 11                     // I2C address
int LED_pins[] = { 2, 3, 4, 5, 6, 7, 8, 9 };
byte LED_status = 0;                      // LED state

void setup() {
    for (int i = 0; i < 8; i++) {
        pinMode(LED_pins[i], OUTPUT);
    }
    Wire.begin(MODULE_LED);               // start I2C as a slave
    // register data receive handler
    Wire.onReceive(receiveFromMaster);
    LED_control();
}
void loop() { }
// receive data from master
void receiveFromMaster(int bytes) {
    LED_status = Wire.read();
    LED_control();
}
void LED_control(void) {
    for (int i = 0; i < 8; i++){
        digitalWrite(LED_pins[i], ((LED_status >> i) & 0x01));
    }
}
```

Figure 13: Slave library for an LED module – Module_LED_slave.ino

Although the types of code presented seem diverse and complex, students can start by writing a main Arduino sketch for programming education. Table 2 summarizes the functions and features of each code.

4.3. FRUTO Kit

The FRUTO kit consists of modular hardware compatible with Arduino UNO and the FRUTO library for programming support. The biggest advantage of the FRUTO kit in hardware is that it is easy to add I/O devices. The hardware configuration is completed by simply connecting the module having the I/O devices required through a unified 4-pin connector. Observations of the course using the FRUTO kit showed that the time required for hardware configuration was less than one third compared to the course using other Arduino-compatible kit. The rest of the time could be spent on conceptual descriptions of the hardware and how it works.

The biggest advantage of the FRUTO kit in software is that the hardware-dependent code is provided as a library. Low-level hardware control tasks are handled in each expansion module, which allows students to build and test a system by writing only hardware-independent code for a main module. The hardware-independent code can be made in the same manner with the original sketch except that there is no hardware-controlling code. The length of the code required when using the FRUTO library is about 1/2 when controlling simple I/O device such as an LED module, and about 1/4 when controlling a complex I/O device such as an LED matrix module compared to the code using the Arduino library. Overall, it was less than one third. The rest of the time can be used to develop the logical procedures of the system.

In addition to simplifying hardware connection and programming, one of the advantages of the FRUTO kit is that it can be used as an educational tool for students of various levels. Table 3 shows an example of how to use the FRUTO kit depending on the students' level. For beginners, the biggest advantage is that it allows them to easily configure a system and write a program for it with minimal hardware knowledge. Intermediates can use the expansion modules and Arduino environment to learn Arduino and/or microcontrollers. Module DIY allows advanced students to design and implement an Arduino-compatible kit.

5. Conclusion

As the importance of programming education is being emphasized more than ever, the demand for an appropriate learning tool is also increasing. In addition, research findings that using Arduino can improve problem-solving and team-level collaboration, have led to various attempts to use Arduino for programming education. However, existing Arduino-compatible tools are limited in their use because of their hardware dependency. In this paper, the design of a modular Arduino-compatible kit, termed as FRUTO, that minimizes hardware dependency has been proposed, and its configuration and features are examined that implemented the proposed design. The FRUTO kit is easy to connect and program, and expandable in many directions, which makes it a versatile learning tool.

Table 2: Codes for the FRUTO kit

Name		Code Sample	Uploaded to	Function	Student Level	Remarks
Master Sketch		Fig. 8	Main Module	Control expansion modules with the master library	Beginner	
FRUTO Library	Master Library	Fig. 9, 10	(Main Module)	Exchange data with expansion modules	Expert	Used in conjunction with a master sketch
	System Library	Fig. 11, 12	(Main Module)	Initialize the system	Expert	
	Slave Library	Fig. 13	(Expansion Module)	Exchange data with a main module and control I/O devices accordingly	Intermediate	Pre-installed in each expansion module

Table 3: FRUTO kit utilization by level

Level	Hardware		Software		Remarks
	Used hardware	Hardware dependency	Programming area	Used library	
Beginner	FRUTO kit	LOW	Master sketch	FRUTO	Useful for rapid prototyping
Intermediate	Main or expansion module alone	MID	Slave library, Module test sketch	Arduino	Similar to using existing Arduino-compatible kits
Expert	Module DIY	HIGH	FRUTO library	Arduino	

Currently, the FRUTO kit has 10 expansion modules and is undergoing a pre-launch test in programming and microcontroller-related education. Although the FRUTO kit satisfies the design guide, developing a Scratch-like block programming tool is expected to increase the number of users. Securing compatibility with existing learning tools like Lego is also expected to contribute to user growth. The feedback collected during the test including the ones mentioned above might be incorporated into future modifications and revisions.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] J. Choi, S. An, and Y. Lee, "Computing education in Korea-current issues and endeavors", ACM Transactions on Computing Education, **15**(2), 1-8, 2015. http://doi.org/10.1145/2716311

[2] D. O'Sullivan and T. Igoe, Physical Computing, Thomson, 2004.

[3] A. Khanlari, "Effects of educational robots on learning STEM and on students' attitude toward STEM," in 2013 IEEE 5th Conference on Engineering Education, Kuala Lumpur, Malaysia, 2013. http://doi.org/10.1109/ICEED.2013.6908304

[4] H. Aoki, J. M. Kim, Y. Idosaka, T. Kamada, S. Kanemune, and W. G. Lee, "Development of state-based squeak and an examination of its effect on robot programming education," KSII Transactions on Internet and Information Systems, **6**(11), 2880-2900, 2012. http://doi.org/10.3837/tiis.2012.11.008

[5] H. Bort, M. Czarnik, and D. Brylow, "Introducing computing concepts to non-majors: a case study in gothic novels," in the 46th ACM Technical Symposium on Computer Science Education, Kansas City, Missouri, USA, 2015. http://doi.org/10.1145/2676723.2677308

[6] M. Banzi and M. Shiloh, Getting Started with Arduino: The Open Source Electronics Prototyping Platform, Make Media, 2014.

[7] L. M. Herger and M. Bodarky, "Engaging students with open source technologies and Arduino," in 2015 IEEE Integrated STEM Education Conference, Princeton, NJ, USA, 2015. http://doi.org/10.1109/ISECon.2015.7119938

[8] Y. Jang, W. Lee, and J. Kim, "Assessing the usefulness of object-based programming education using Arduino," Indian Journal of Science and Technology, **8**(S1), 89-96, 2015. http://doi.org/10.17485/ijst/2015/v8iS1/57701

[9] J. H. Park and S. H. Kim, "Case study on utilizing Arduino in programming education of engineering," The Journal of Institute of Korean Electrical and Electronics Engineers, **19**(2), 276-281, 2015. http://doi.org/10.7471/ikeee.2015.19.2.276

[10] P. Mellodge and I. Russel, "Using the Arduino platform to enhance student learning experiences," in the 18th ACM Conference on Innovation and Technology in Computer Science Education, Canterbury, England, UK, 2013. http://doi.org/10.1145/2462476.2466530

[11] J. Sarik and I. Kymissis, "Lab kits using the Arduino prototyping platform," in 2010 IEEE Frontiers in Education Conference, Washington, DC, USA, 2010. http://doi.org/10.1109/FIE.2010.5673417

[12] K. Eom, Y. Jang, J. Kim, and W. Lee, "Development of a Board for Physical Computing Education in Secondary Schools Informatics Education," The Journal of Korean Association of Computer Education, **19**(2), 41-50, 2016. http://doi.org/10.32431/kace.2016.19.2.005

[13] A. Garrigos, D. Marroqui, J. M. Blanes, R. Gutierrez, I. Blanquer, and M. Canto, "Designing Arduino electronic shields: Experiences from secondary and university courses," in 2017 IEEE Global Engineering Education Conference, Athens, Greece, 2017. http://doi.org/10.1109/EDUCON.2017.7942960

[14] Tn1, Microcontroller Kit, Korean Patent 1017353010000 to Korean Intellectual Property Office, 2017.

[15] G. Heo and J. Jung, "Arduino Compatible Modular Kit Design for Educational Purpose", Journal of the Korea Institute of Information and Communication Engineering, **22**(10), 1371-1378, 2018. https://doi.org/10.6109/jkiice.2018.22.10.1371

[16] M. Rpzybylla and R. Romeike, "Physical computing and its scope – Towards a constructionist Computer science curriculum with physical computing," Informatics in Education, **13**(2), 241-254, 2014. http://doi.org/10.15388/infedu.2014.05

[17] P. Plaza, E. Sancristobal, G. Fernandez, M. Castro, and C. Perez, "Collaborative robotic educational tool based on programmable logic and Arduino," in 2016 Technologies Applied to Electronics Teaching, Seville, Spain, 2016. http://doi.org/10.1109/TAEE.2016.7528380

[18] P. Martin-Ramos, M. J. Lopes, M. M. L. da Silva, P. E. B. Gomes, P. S. P. da Silva, J. P. P. Domingues, and M. R. Silva, "First exposure to Arduino through peer-coaching: Impact on students' attitudes towards programming," Computers in Human Behavior, **76**, 51-58, 2017. http://doi.org/10.1016/j.chb.2017.07.007.

[19] S. J. Kim, "Project-based embedded system education using Arduino," The Journal of Korean Institute of Information Technology, **15**(12), 173-180, 2017. http://doi.org/10.14801/jkiit.2017.15.12.173

[20] X. Righetti and D. Thalmann, "Proposition of a modular I2C-based wearable architecture," in 2010 15th IEEE Mediterranean Electrotechnical Conference, Valletta, Malta, 2010. http://doi.org/10.1109/MELCON.2010.5475965