ISSN:1226-7244 (Print)
ISSN:2288-243X (Online)
논문번호 17-03-07
54
j.inst.Korean.electr.electron.eng.Vol.21,No.3,227~233,September 2017
http://dx.doi.org/10.7471/ikeee.2017.21.3.227

다수의 아두이노를 파이썬과 I2C로 제어하기 위한 무오류 통신 프로토콜 구현

Implementation of errorless protocol for controlling multiple Arduinos using python via I2C communication

박 장 현 *, 김 성 환*, 박 태 식*★

Jang-Hyun Park*, Seong-Hwan Kim*, Tae-Sik Park*★

Abstract

Python language is widely used because of its ease of learning and its wide application range. Arduino, on the other hand, is also widely utilized hardware for physical computing and internet of things(IoT). However, Arduino is controlled by C++ language, which makes it difficult for non-experts to enter swiftly. This paper proposes an errorless protocol that can simultaneously control multiple Arduino devices in a master device using python language with I2C communication. Using the protocol proposed in this paper, we can take advantage of the python language to control multiple Arduinos.

요 약

근래에 전 세계적으로 비전공자나 중등학생들에게도 코딩 교육이 폭넓게 이루어지고 있으며 파이썬(python)은 비전문가의 교육용으로도 널리 채택되고 있다. 그리고 아두이노(arduino)는 피지컬 컴퓨팅(physical computing)과 사물인터넷 용도로 사용되는 대중적인 하드웨어이지만 C++ 언어로 제어되므로 비전문가가 초기에 진입하기에 어려움이 있다. 본 논문은 파이썬 언어와 I2C 통신으로 마스터(master) 기기에서 다수의 아두이노 기기들을 동시에 제어할 수 있는 무오류 프로토콜을 제안한다. 본 논문에서 구현된 프로토콜을 이용하면 다수의 아두이노를 파이썬으로 오류 없이 제어할 수 있으므로 아두이노를 제어하는데 파이썬 프로그램의 장점을 활용할 수 있다.

Key words : python, arduino, raspberry pi, i2c, protocol

Ⅰ. 서론

사물인터넷(Internet of Things, IoT)은 다수의 센서와 물리적 객체를 인터넷에 연결하는 중재자로 구성된 포괄적인 환경을 말한다.[1]-[3] 사물인터넷 기술이 주목받으면서 글로벌 기업들과 표준을 제정하는 국제기구 등에서 표준화와 관련 기술 개발이 활발히 이루어지고 있으며, 이러한 경향과 별개로 비전문가들을 포함한 개인들도 저가로 손쉽게 구현할 수 있는 다양한 하드웨어가 출시되었다. 그 중에서 가장 널리 사용되는 플랫폼으로 아두이노 (arduino)와 라즈베리 파이 (raspberry pi)가 있다. 아두이노는 AVR 8비트 마이크로컨트롤러 기반의 제어 보드이고 그림 1

* Dept. of Electrical and Control Engineering, Mokpo National University
★ Corresponding author
tspark@mokpo.ac.kr, 061-450-2465
※ Acknowledgment
This Research was supported by Research Funds of Mokpo National University in 2016.
Manuscript received Jul. 20, 2017; revised Sep. 7, 2017 ; accepted Sep. 20, 2017

에 예를 든 것과 같이 다양한 종류가 있다. 라즈베리파이는 저가의 소형 리눅스 PC 이며 그림 2와 같은 외형이다. 이들 기기와 센서 및 작동기를 조합하여 다양한 전자 제품을 직접 개인이 만들어 보는 소위 메이커 운동(maker movement)이 전세계적으로 일어나고 있다.[4]

아두이노[5]는 AVR 마이크로컨트롤러를 채용한 제어 보드로서 2007년에 처음 출시된 이후 피지털 컴퓨팅(physical computing) 도구로 널리 사용되고 있다. 저렴한 가격과 사용하기 쉬운 라이브러리 및 다양한 커뮤니티의 지원에 힘입어 대표적인 마이컴 보드로 인식되고 있다. 아두이노의 펌웨어는 C++ 언어를 이용하여 개발하는데 관련 라이브러리가 사용하기 용이하게 제작되어 무료로 배포되고 있어서 예술가 등 비전문가들도 널리 사용하고 있다. 단, C++언어의 특성상 초기의 학습 난이도가 높은 편이며 특히 아두이노를 이용하여 IoT 제품을 구현하기 위해서는 상대적으로 값비싼 하드웨어와 다소 복잡한 개발 과정을 거쳐야 한다. 이러한 단점에도 불구하고 다양한 아두이노 제어 프로그램이 오픈소스(open source)로 공개되어 있으며 이러한 인프라를 적절히 이용하는 것은 아두이노를 활용한 제품 개발에 있어서 무척 중요하다.

파이썬(python) 언어는 C/C++ 계열의 언어들보다 배우기 쉽고 활용 영역이 매우 광범위하여 널리 사용되고 있다.[6] 근래에는 전 세계적으로 비전공자나 중등학생들에게도 코딩 교육이 폭넓게 이루어지고 있으며 스크래치(scratch)와 같은 GUI(graphical user interface)형 프로그래밍 언어와 함께 파이썬은 비전문가의 교육용으로도 널리 채택되고 있다. 또한, 아두이노가 프로그래밍 교육에 활용되는 경우도 늘고 있으며[7] 라즈베리파이 같은 저가의 컴퓨터에서도 리눅스를 운영체제로 사용하여 파이썬을 사용할 수 있으며 이를 통해서 손쉽게 수치해석 알고리듬이나 네트워크 프로그래밍을 구현할 수 있다 [8][9].

본 논문은 파이썬 명령어로 I2C 통신을 이용하여 마스터(master) 기기에서 슬레이브(slave) 아두이노 기기를 제어할 수 있는 무오류 통신 프로토콜을 제안한다. 시리얼 통신이나 SPI통신과 달리 I2C 통신은 그 특성상 다수의 슬레이브를 동시에 연결하여 제어할 수 있다. 본 논문에서 제안하는 프로토콜을 이용하면 다수의 아두이노를 파이썬으로 제어할 수 있으므로 아두이노와 파이썬 프로그램의 장점을 동시에 가질 수 있다. 제안된 알고리듬은 파이썬 패키지와 아두이노 라이브러리로 구현하여 마스터 기기로 라즈베리파이3 (raspberry pi 3) 보드를 이용하여 실험을 진행하였으며, 구현된 모든 소스를 github.com에 공개하여[10] 누구나 자유롭게 사용할 수 있도록 하였다. 본 논문에서 제시된 알고리듬을 구현하여 공개된 라이브러리을 이용하면 다수의 센서 값을 동시에 모니터링하거나 읽어 들인 센서 값을 반영하여 모터나 릴레이 같은 다수의 작동기(actuator)를 동시에 파이썬 프로그램으로 제어하는 등의 응용이 가능하다.

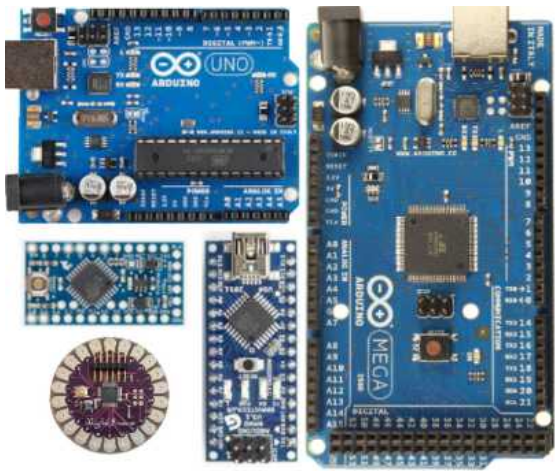

Fig. 1. arduino boards.
그림 1. 아두이노(arduino) 보드들

Fig. 2 Raspberry pi 3 model B.
그림 2 라즈베리파이3 모델 B

II. 프로토콜 설계

본 장에서는 파이썬으로 아두이노의 C++ 함수를 호출하기 위해서 무오류 통신 프로토콜 알고리듬을 제안한다. 마스터는 아두이노를 파이썬으로 제어하는 기기를 의미하며 본 논문에서는 라즈베리파이3를 사용하였으나 I2C 통신을 지원하는 리눅스 임베디드 보드라면 어느 것이나 사용할 수 있다. 마스터에서는 파이썬의 smbus 패키지를 이용하여 I2C 통신을 수행한다. 슬레이브는 마스터와 연결되어 마스터의 명령에 반응하여 적절한 동작을 수행하는 기기이다. 본 논문에서는 아두이노를 슬레이브로 사용하며 Wire.h 라이브러리를 활용하여 프로그램이 제작되었다. 프로토콜을 설계하는 목적은 아두이노 내부에서 C++ 로 작성된 함수(function)를 마스터 기기에서 파이썬으로 호출하고 그 결과 값을 오류 없이 읽어오는 기능을 구현하는데 사용되는 통신 알고리즘에 적용하기 위함이다. 마스터가 요구하는 작업들은 표 1에 기술된 바와 같이 분류된다.

Table 1. Tasks between master and slave devices.
표 1. 마스터/슬레이브 간 수행 작업들

no.	request of master	response of slave	error check
1	basic infomation	basic information	checksum
2	status value	status value	
3	function argument registration	register received argument	proposed protocol
4	function call	execute function and store result	
5	return value of the function	send stored results	
6	new I2C address	change I2C address	

이 중 3번, 4번, 5번과 6번의 경우는 마스터가 전송한 데이터를 오류 없이 정확하게 수신해야 슬레이브에서 치명적인 오동작이 발생하지 않는다.

1. 최초 접속 시 전송되는 데이터

마스터에서 슬레이브로 I2C방식으로 최초로 접속하면 슬레이브에서 초기 정보를 읽어야 하는데 그 데이터 구조는 표 2와 같다.

Table2. data packet for device information.
표 2. 디바이스 정보 데이터 패킷

information	data type	byte
device id	uint32_t	4
max. number of function arguments	byte	1
number of functions	byte	1
ardpy version	uint16_t	2
firmware version	uint16_t	2
checksum	byte	1

초기 정보는 슬레이브 기기의 식별 정보와 함수 호출에 사용되는 인자의 최대 개수, 전체 함수의 개수, 사용된 라이브러리의 버전 번호, 펌웨어 버전의 번호 그리고 정보의 오류를 검출할 체크썸(checksum)이다. 이러한 정보들은 이후에 통신을 수행하는 시점에서 사용된다.

2. 슬레이브에서 함수 실행

마스터에서 슬레이브로 전송하려는 데이터는 파이썬 측에서나 아두이노 내부에서나 모두 바이트(byte) 데이터열로 직렬화(serialization) 시킨 후 차례로 전송하게 된다. 통신으로 전송되는 데이터열이 오류가 없다고 판단하는 고전적인 방법으로 전송 데이터 끝에 계산된 체크썸(checksum)을 추가로 전송하는 것이다. 즉, 동일한 계산식으로 얻은 그것과 전송 받은 것이 같을 경우 오류 없는 데이터라고 판단하는 방식이 널리 사용된다. 하지만 이 방식을 이용해도 실제 통신 환경에서는 드물게 데이터와 체크썸이 동시에 오염된 경우 치명적인 오류를 피하기 어렵다.

본 논문에서 제안하는 핵심적인 알고리듬은 다음과 같다. 만약 마스터에서 일련의 데이터를 슬레이브로 보낸 후 이 데이터를 바로 처리하지 않고 받은 데이터를 그대로 다시 마스터에게 돌려보낸다. 그러면 마스터는 보낸 원본 데이터와 재전송 받은 데이터 전체를 비교하여 동일하다면 오류가 없다는 신호를 슬레이브에게 보낸다. 슬레이브에서는 이 신호를 받은 이후에 데이터를 처리하여 지정된 작업을 수행하게 된다. 이 동작을 시간적인 순서도로 도시하면 그림 3과 같다.

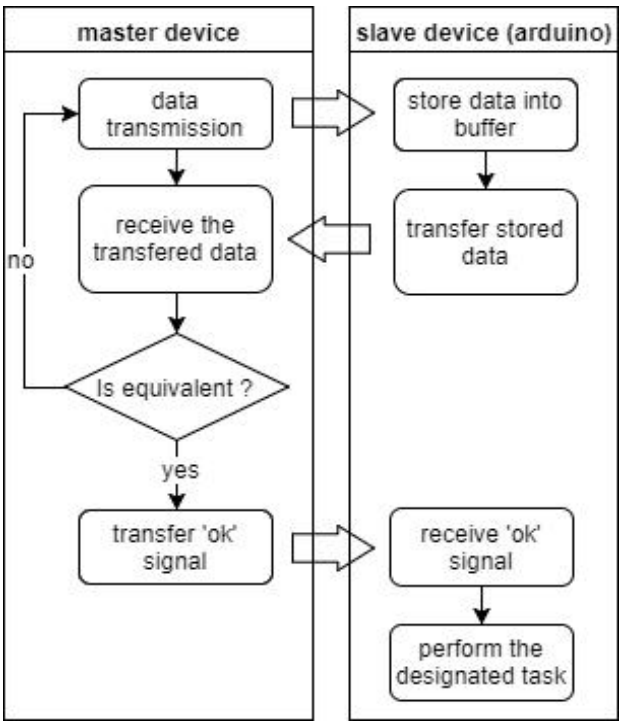

Fig. 3. Flowchart of sending data from master to slave.
그림 3. 마스터에서 슬레이브로 데이터를 전송하는 경우의 순서도

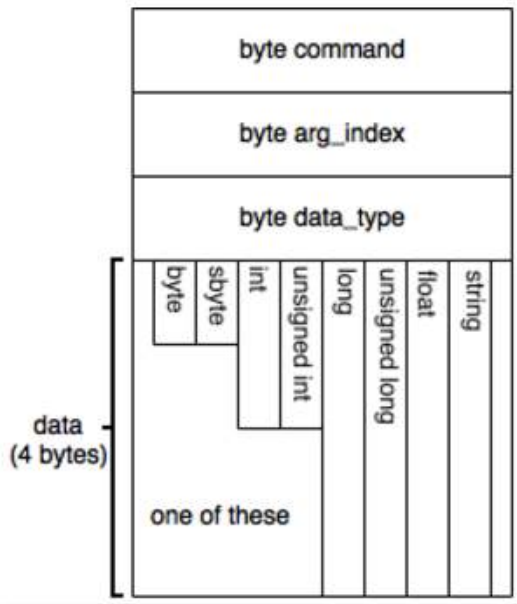

Fig. 4. Data packet for function argument.
그림 4. 함수 인자 데이터 패킷

아두이노에 함수의 인자 데이터를 전송할 경우 그림 4와 같은 데이터 패킷(packet)이 전송된다. 만약 실행하고자 하는 함수에 인자가 세 개가 요구된다면 서로 다른 인덱스(그림 4의 arg_index)를 가진 세 개의 패킷이 세 번 전송된다.

슬레이브의 함수를 실행하려면 표 3의 1항과 같은 정보를 갖는 패킷이 슬레이브로 전송된다.

Table 3. Various command/data packet.
표 3. 명령/데이터 패킷의 구조

no.	packet name	packet structure	byte
1	function execution	command	1
		function id	1
2	status data	status	1
		checksum	1
3	address change	command	1
		new address	1

그리고 함수 실행 명령을 전송한 후 처리가 완료되었는지를 검사하기 위해 상태 정보를 읽어오게 되는데 그 패킷의 구조는 표 3의 2항과 같다.

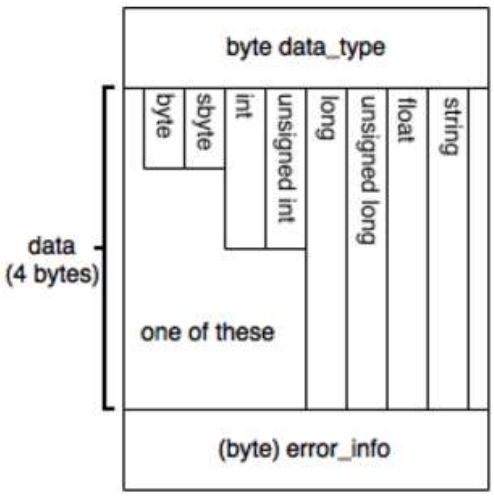

Fig. 5. Data packet for resulting data.
그림 5. 결과 데이터 패킷

함수의 수행이 끝나면 생성된 결과값을 마스터에서 요구하게 된다. 결과값의 데이터 패킷 구조는 그림 5에 도시된 바와 같고, 만약 실행 도중 오류가 발생했다면 부가적인 데이터에 오류의 종류를 기록하도록 하여 마스터에서 오류를 검출할 수 있도록 한다.

마스터에서 슬레이브의 함수를 호출하는 동작을 기술한 전체적인 순서도는 그림 6에 도시되어 있다.

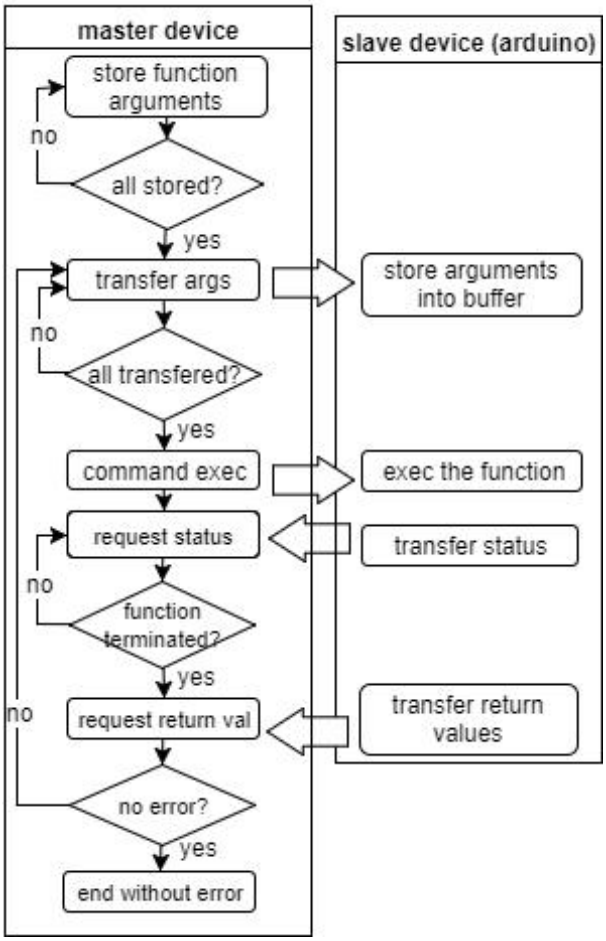

Fig. 6. Flow chart for calling C++ function in the slave.
그림 6. 슬레이브의 C++함수를 실행하는 순서도

3. 슬레이브의 I2C 주소 변경

마스터 기기에서 슬레이브의 I2C 주소를 파이썬 명령어로 변경할 수 있도록 하기 위해서 표 2의 3항과 같은 데이터 패킷을 전송한다. 주소 값을 전송하는데 있어서 오류가 있으면 안 되므로 이 패킷을 전송할 경우에도 제안된 무오류 통신 알고리듬을 사용한다. 아두이노에서 이 데이터를 받으면 내부 EEPROM에 주소를 저장하며 그 이후에 리셋되면 새로운 주소로 I2C 통신을 개시한다.

III 실험 결과

제안된 알고리듬을 실증하기 위해서 다음과 같은 테스트 환경에서 실험을 진행하였다. 마스터 보드로 라즈베리파이3를 이용하였다. 이 보드는 데비안(Debian) 리눅스 기반의 라즈비언(Raspbian)이라는 독자적인 운영체제로 구동되며 따라서 파이썬(버전 3)을 구동하는데 아무런 제약이 없다. 또한 입출력(GPIO) 핀들이 있고 여기에서 I2C 두 채널을 제공하며 각각 100kbps의 속도로 통신을 수행할 수 있다. 라즈베리파이 3 보드의 사양은 표 4에 기술하였다.

Table 4. Specificationa of Rapsberry pi 3.
표 4. 라즈베리파이3의 사양

item	specification
CPU	Boradcom BCM2837 64bit
clock speed	1.2MHz
RAM	1GB
GPIO	40pin
storage	microSD
network	10/100 Eternet / WiFi / BT
power	5V@2.4A

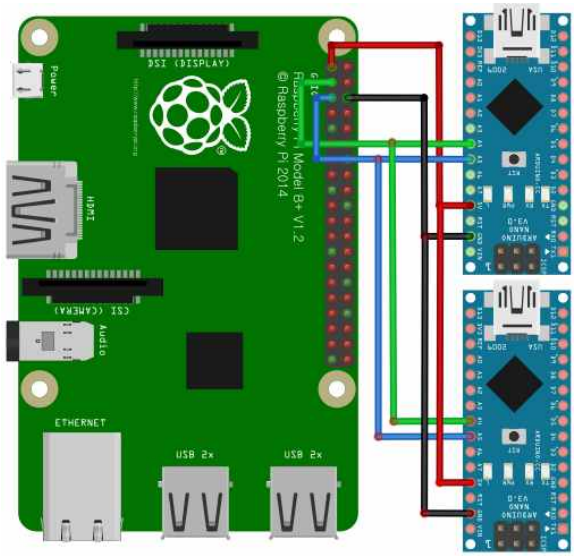

Fig. 7. interface circuit between Raspberry Pi and two Arduino nanos.
그림 7. 라즈베리파이와 두 개의 아두이노(나노) 간 I2C 인터페이스 회로

그림 7에 두 개의 아두이노 나노(3.3V 동작)와 라즈베리파이 3의 결선도를 도시하였다. I2C 통신 방식의 특성상 다수의 아두이노를 하나의 채널에 연결할 수 있으며 각각의 아두이노는 고유의 주소를 가진다. 따라서 이 주소를 이용하여 마스터에서는 개별 슬레이브 기기들을 구별하여 독립적으로 동시에 제어할 수 있다. 실험을 위하여 라즈베리파이에 장착할 수 있는 보드를 제작하였으며 결과물이 그림 8에 제시되어 있다. 이 애드온

(add-on) 보드에는 아두이노 나노가 포함되어 있으며 라즈베리파이 3의 GPIO에 장착하면 I2C 연결이 되도록 PCB가 설계되어 있다. 또한 다른 슬레이브 보드에 I2C 통신선을 연결할 수 있는 4핀 소켓을 포함하고 있다. 이 애드온 보드를 거쳐 연결 소켓을 통해 I2C로 연결할 수 있는 별도의 LCD 모듈을 제작하였으며 전체 실험도를 그림 9에 도시하였다.

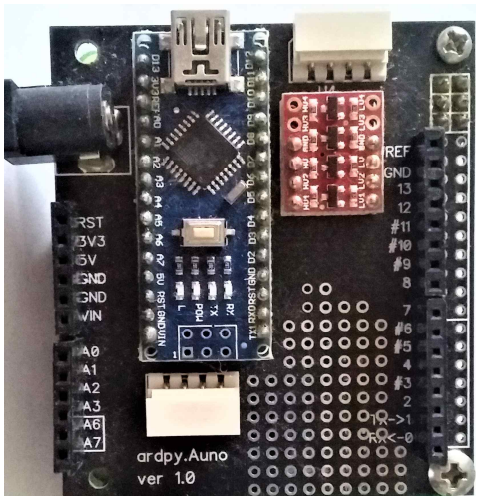

Fig. 8. Add-on board for Raspberry pi.
그림 8. 제작된 라즈베리파이 애드-온 보드

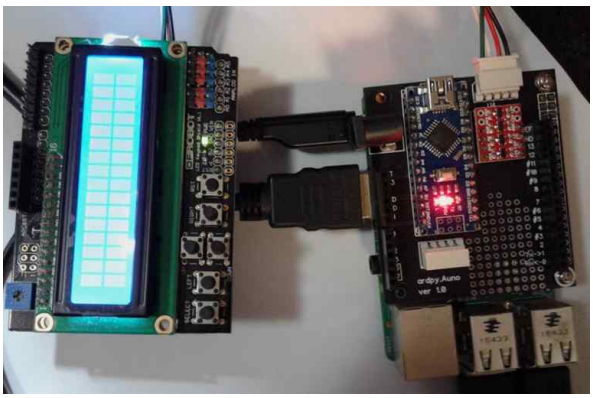

Fig. 9. Implement for controlling two Arduinos.
그림 9. 두 개의 아두이노를 동시에 제어하는 실험

그림 9에서 보면 두 개의 아두이노 나노 보드가 라즈베리파이 3에 동시에 연결되어 있으며 하나는 애드온 보드에 있고(편의상 1번 슬레이브 보드라 칭한다.) 다른 하나는 LCD보드에(2번 슬레이브 보드) 있다. 성능의 검증을 위해서 1번 아두이노에서 millis() 함수의 반환값인 경과 시간을 읽어 들이면서 동시에 그 값을 2번 아두이노에 연결된 LCD에 표시하는 실험을 실시하였다. 또한 1번 슬레이브 보드에 광량 센서와 온도 센서를 연결한 후 읽어 들인 센서 값을 2번 아두이노에 연결된 LCD에 표시하는 실험도 진행하였다. 이 실험들에서 두 개의 아두이노를 장시간 동시에 제어하는데 아무런 통신 오류가 발생하지 않음을 확인하였다. 따라서 본 논문에서 제시한 프로토콜을 이용하여 라즈베리파이3 보드에서 파이썬을 이용하여 다수의 아두이노를 성공적으로 제어할 수 있음을 확인하였다.

Ⅳ 결론

본 논문은 파이썬 명령어로 I2C 통신을 이용하여 마스터 기기에서 다수의 슬레이브 아두이노 기기들을 동시에 제어할 수 있는 무오류 프로토콜을 제안했다. 본 논문에서 제안하는 프로토콜을 이용하면 다수의 아두이노를 파이썬으로 통신 오류 없이 제어할 수 있으므로 아두이노와 파이썬 프로그램의 장점을 동시에 가질 수 있다. 또한 아두이노로 다양한 센서와 작동기를 연결하여 데이터를 취득하거나 제어를 할 수 있으므로 이를 작성하기 용이한 파이썬 프로그램으로 수행할 수 있는 방법을 구현하였다는데 의미가 있다. 제안된 알고리듬은 파이썬 패키지와 아두이노 라이브러리로 구현하여 마스터 기기로 라즈베리파이3 보드를 이용하여 실험을 진행하여 그 성능을 검증하였다. 본 논문에서 제안된 프로토콜과 이를 구현한 파이썬 패키지와 아두이노 라이브러리는 소스 코드를 github.com [10]에 등록하여 모든 사람들이 쉽게 이용할 수 있도록 하였다. 향후에는 이를 이용하여 제어 시스템을 제작하거나 사물인터넷 등의 응용에 활용될 수 있으리라 기대된다.

References

[1] G. Gardasevic et. al., “The IoT Architectural Framework, Design Issues and Application Domains,” *Wireless Personal Communications*, vol.92, no. 1, pp. 127-148, Jan. 2017.
DOI : https://doi.org/10.1007/s11277-016-3842-3
[2] C. G. Garcia, et. al., “A review about Smart Objects, Sensors, and Actuators,”

International Journal of Interactive Multimedia and Artificial Intelligence, vol. 4, no. 3, pp. 7-10, 2017.
DOI : 10.9781/ijimai.2017.431
[3] S. N. Srirama, "Mobile Web and cloud services enabling Internet of Things," *CSI trans. on ICT*, vol. 5, no. 1, pp. 109-117, 2016.
DOI : https://doi.org/10.1007/s40012-016-0139-3
[4] S. Papavlasopoulou, M.N. Giannakos, L. Jaccheri, "Empirical Studies on the Maker Movement, a Promising Approach to Learning: A Literature Review," *Entertainment Computing*, vol. 18, pp. 57-78, Jan. 2017.
DOI : https://doi.org/10.1016/j.entcom.2016.09.002
[5] Arudino organization, "ARDUINO," https:// www.arduino.cc/
[6] MASON, Raina, et al. "Introductory Programming Courses in Australasia in 2016," in *Proc. of the Nineteenth Australasian Computing Education Conference. ACM*, 2017, pp. 81-89.
DOI : 10.1145/3013499.3013512
[7] J.-H. Park, S.-H. Kim, "Case Study on Utilizing Arduino in Programming Education of Engineering," *J. IKEEE*, vol. 19, no. 2, pp. 276-281, 2016.
DOI : 10.7471/ikeee.2015.19.2.276
[8] RaspberryPi Foundation, "RASPBERRY PI," https://www.raspberrypi.org
[9] N. Petrov, et. al., "Example of Raspberry Pi usage in Internet of Things," in Proc. of *Int. Conf. Applied Internet and Information*, 2016, pp. 112-119.
DOI : 10.20544/AIIT2016.15
[10] J.-H. Park, "Ardpy", https:// github.com/ salesiopark/Ardpy

BIOGRAPHY

Jang-Hyun Park(Member)

1995 : BS degree in Electrical Engineering, Korea University.
1997 : MS degree in Electrical Engineering, Korea University.
2002 : PhD degree in Electrical Engineering, Korea University.
2003~ : Professor, Dept. Electrical and Control Engineering, Mokpo National University

Seong-Hwan Kim(Member)

1991 : BS degree in Electrical Engineering, Korea University.
1995 : MS degree in Electrical Engineering, Korea University.
1998 : PhD degree in Electrical Engineering, Korea University.
1999~ : Professor, Dept. Electrical and Control Engineering, Mokpo National University

Tae-Sik Park(Member)

1994 : BS degree in Electrical Engineering, Korea University.
1996 : MS degree in Electrical Engineering, Korea University.
2000 : PhD degree in Electrical Engineering, Korea University.
2013~ : Professor, Dept. Electrical and Control Engineering, Mokpo National University