Development of Teaching Materials for Computer Programming using a Robot Remotely Controlled by a PC through Wireless Communication

Toshihiro Shikama†

†Fukui University of Technology
shikama@fukui-ut.ac.jp

Abstract - We developed teaching materials for students to increase their interest in computer programming. We employed a robot specified by ET Robocon (Embedded Technology Software Design Robot Contest). Although the robot in ET Robocon is controlled by a program running in the robot itself, a program written by a student runs on a separate PC and also controls the robot through wireless communication via Bluetooth. As for the programming language that students learn, we selected Python because of its simplicity and similarity with object-oriented programming. A student can start programming simple sequential control of the robot and extend it to programming that realizes line tracing.

Keywords: Embedded Systems, ET Robocon, Python, Teaching Materials, Line Tracing

1 INTRODUCTION

This paper reports the development of teaching materials (sometimes shortened to "materials" in this paper) for computer programming. When a student is learning computer programming, the initial stage is important. The student generally takes a long time to become familiar with abstract programming concepts such as data types, structures, and classes of object-oriented programming. These concepts are separate from physical instances and difficult to learn. If we educate students under the false assumption that they will easily understand these abstract concepts, the students may abandon learning because of a loss of interest.

We participated in the ET Robocon (Embedded Technology Software Design Robot Contest) so that students could learn embedded systems [1]. In this contest, students analyze and design a computer program using UML (Unified Modeling Language) to control the robot, which contains strictly defined hardware with no modifications allowed. We observed that students who participated in this contest tended to become enthusiastic about computer programming. From this experience, we expect that more students will show an interest in computer programming if we incorporate the robot into the programming education. Based on this motivation, we developed teaching materials for computer programming using the robot defined by ET Robocon.[1]

2 OUTLINE OF ET ROBOCON AND OBJECTIVES OF THIS WORK

2.1 ET Robocon

The objective of ET Robocon is to improve the capability of software technology for embedded systems. This contest uses control software targeting a two-wheel self-balancing robot using LEGO® MINDSTORMS NXT [2]. Figure 1 shows the appearance of the robot and its components. The robot consists of an ultrasonic sensor, a gyro sensor, a light sensor, and three motors for the right wheel, left wheel, and tail. It is equipped with a 32-bit ARM7 microprocessor. Students develop control programs that enable the robot to autonomously trace a line around a specified course. Figure 2 shows a photo of the ET Robocon 2011 course.

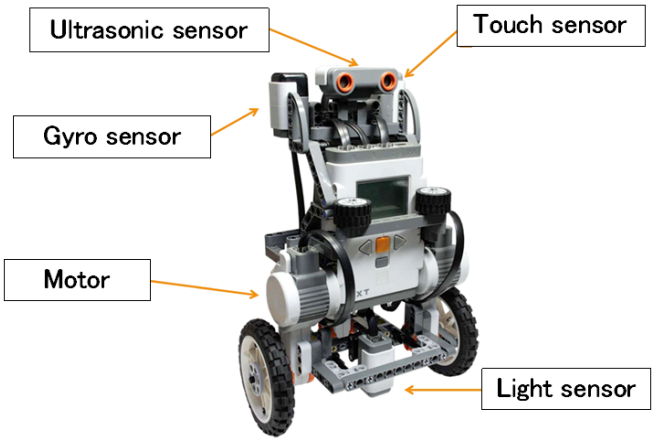

Figure 1: The robot and its components

Figure 2: ET Robocon 2011 course

[1] The work reported in the paper was supported by the Special Research Grant-in-Aid of Fukui University of Technology.

The robot runs along the black lines drawn over white areas, which are surrounded by green "ground". Students are required to develop a control program that makes the robot run along the black line at high speed. All teams in the contest use the same robot, which has a limited number of sensors (an ultrasonic sensor, a gyro sensor, and a light sensor). The ET Robocon consists of two parts: modeling and a time trial. The modeling part is a competition of the UML modeling skill used in developing the program, and the time trial part is a run-time competition of the robot. The total score is determined from the results of the two parts.

2.2 Objectives of this work

The intention of this work is to develop teaching materials for computer programming and to promote the enjoyment of computer programming for beginner students. We are aiming at the following goals:

1) The control of the robot is realized by a simple program (i.e., a small number of program steps).
2) A student can start programming without understanding abstract programming concepts.
3) The basics of programming skills, such as conditional branches, loops, functions, etc., can be studied through the developed materials.
4) The materials can be applied to the education of object-oriented programming and multi-thread programming.

3 THE BASIC ARCHITECTURE

3.1 Outline of the materials

In ET Robocon, a control program is written in the C or C++ language and the program is compiled to produce a binary file which is loaded into the robot through a USB interface. After the program is invoked, the robot is autonomously controlled by the program. Although this scheme can enable accurate and efficient control of the robot, debugging is limited since the robot has only a small display to show internal information and status. Another problem is the time and work required for students; each time a program is modified, the students must compile, link, and download the binary file through the USB interface. After considering these drawbacks, we apply the following scheme for a program running on a PC to control the robot.

1) A fixed control program is pre-loaded into the robot. This program executes basic commands from the PC. Students do not modify the program in the robot.
2) The basic commands are sent from the PC through wireless communication via Bluetooth.
3) Control of the robot is achieved by the program running in the PC. This program describes combinations of basic commands.
4) As the programming language, Python [3] is selected for the program in the PC.

3.2 Adoption of Python

As mentioned above, we adopted Python as the programming language for students to learn. Python is an object-oriented scripting language and has the following features:

- Since a program can be executed without compiling, a student can modify his program easily and test it quickly.
- Python is well defined and easy for beginners to learn.
- Python is used globally.
- A student can learn object-oriented programming easily by Python.
- Python is available at no cost and is supported by multiple platforms, including Windows and Linux.
- Because indents are mandatory in Python, a program can generally be read easily. In addition, differences in the programming style between students are small.

Adoption of Python has the following drawbacks:

- Python has a compatibility problem between versions 2 and 3.
- Performance of the program is slow because it is a scripting language.
- Indentation is employed for identifying program blocks; this programming style is different from other languages, such as C and C++.

For compatibility between Python versions, we use version 2. Our focus is on the educational aspect, and so we do not seek to realize high running speed of the robot. Concerning the indentation used in Python, we think that this is not a serious problem for students, who will study the C or C++ programming language after learning Python.

3.3 Configuration of the materials

Figure 3 shows the total configuration of the developed materials. The robot and the PC are connected through wireless communication via Bluetooth. The program running in the PC controls the robot remotely.

The Python program running in the PC sends a command through wireless communication; the robot moves forward or makes a left or a right turn by following the program commands. The running speed of the robot is also controlled by the program.

A fixed program running on the ARM 7 microprocessor inside the robot controls the movement of the robot; students do not modify this program, which is developed in the C++ language and runs on the Real-Time Operating System nxtOSEK [4]. This program performs the following functions:

- Controls the posture of the robot
- Receives commands through Bluetooth and interprets them
- Executes commands from the PC
- Sends log data to the PC every 40 ms

The Python module, which was developed for this setup, calculates data to get information about the robot, including

the X- and Y-coordinates of the robot, the angle of the robot, and the total running distance from the start point.

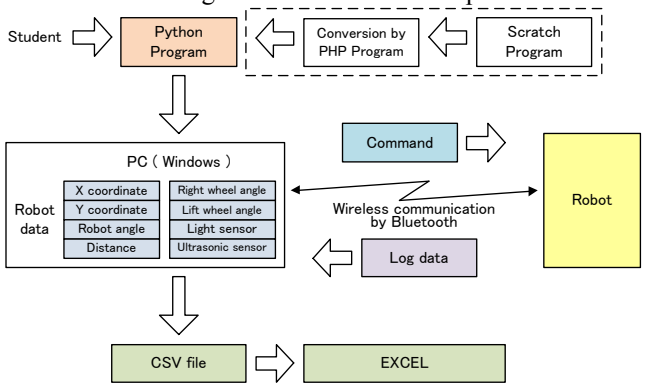

Figure 3: Configuration of the developed materials

When students write programs, they can use variables concerning these data by importing the Python module. The module also provides a log file, including all log data, in the CSV format. Using EXCEL, students can analyze the log file to obtain, for example, a trace of the robot.

Table 1 summarizes the basic functions and commands that students can use in their Python programs. For simplification, the specification commands are limited to one character, whereas extended commands consisting of multiple characters are also provided for future use.

The variable "bt" is the object to control the robot. At the head of a program, the object is generated from the defined class "nxt_bluetooth" as follows:

```
from nxt_bluetooth import nxt_bt
bt = nxt_bt("00:16:53:0c:82:39", 0).
```

The first line imports the class "nxt_bt" from the module "nxt_bluetooth". This mandatory class was developed for the materials used. The second line generates the object "bt", where the parameter "00:16:53:0c:82:39" is an example of the MAC address of the robot. Bluetooth employs a 48-bit MAC address, which is the same as that of LANs. As the "nxt_bt" class hides the Bluetooth communications and updates of the robot variables from students, the students can develop their programs without knowing the internal details.

Table 1: Basic functions and commands

Function	Command	Meaning
bt.send (character)	"f"	Move forward
	"r"	Make a right turn
	"l"	Make a left turn
	"b"	Move backward
	"0"–"9"	Set speed
bt.swait (seconds)	-	Wait specified seconds

Table 2 summarizes the variables of the robot status and sensors. The values of these variables are updated every 40 ms, and so students can use these variables to control the robot. For example, students can quantitatively control the robot, such as moving it forward 500 mm or making a 90-degree left turn.

Table 2: Variables of robot status and sensors

Variable	Meaning
bt.x	X-coordinate of the robot
bt.y	Y-coordinate of the robot
bt.angle	Angle of the robot
bt.distance	Total running distance from the start point
bt.diff_light	Value of the light sensor

4 INTERNAL REALIZATION SCHEME OF THE MODULE

Students import the module "nxt_bluetooth" at the head of their program. The class "nxt_bt", which has been developed in the materials, is included in this module. This class has the functions of communication via Bluetooth, processing log data, and synchronization between the robot and the program running in the PC. As the details are hidden, students have to be aware of only the MAC address of the robot.

When we consider the implementation of the Python class, it is natural to use two separate threads for sending and receiving functions. However, to simplify the implementation, we realize the functions by a single thread, since the robot sends log data every 40 ms. We can eliminate the complexity of the multi-threading and extend the materials to realize the control of multiple robots simply by multi-threading.

Inside the nxt_bt class, sending and receiving functions via Bluetooth are realized by importing the Bluetooth module. This module is provided by python-bluez [5], which is a wrapper function that enables Python to use BlueZ [6]. The robot employs the virtual serial port communication by the serial port profile (SPP) of Bluetooth. BlueZ supports this profile. Although python-bluez is for Linux, PyBluez is also available for the Windows environment. This means that the materials can be used on both platforms, if Python is installed.

Each communication via Bluetooth is initiated by generating a socket with the required parameters and connecting it as follows, where "bt_addr" holds the character string of the MAC address.

```
self.etrobo_address = bt_addr
self.port = 1
self.sock = BluetoothSocket( RFCOMM )
try:
   self.sock.connect((self.etrobo_address, self.port))
except IOError:
   print "Robot is not invoked."
   sys.exit()
print "connected address = ", self.etrobo_address
```

After the socket has been connected, the program enters a wait state, if it calls the receive() function. As we mentioned before, since a single thread performs both sending and receiving, the program has to call the receive() function to enter the receive wait state after its process has completed. This is actually done by calling the wait() or swait() functions. The wait() function specifies a number of 40 ms

units as the wait duration, while the swait() function specifies the wait time in seconds.

```
def wait(self, n):
    self.i = 0
    while self.i < n:
        self.receive()
        self.i = self.i + 1

def swait(self, time):
    self.n = time // 0.04
    self.wait(self.n)
```

In the wait() function, the program waits for the receiving data by the self.receive() function. Since the robot sends log data periodically, completion of the receive occurs within 40 ms. The initial part of the receive() function executes the following code.

```
def receive(self):
    self.starttime = time.time()
    self.data = self.sock.recv(34)
    if len(self.data) != 34:
        print "receive byte length =", len(self.data)
        sys.exit()
    self.udata = unpack('<2BI2bH3i4hi', self.data)
```

The second line records the receive time of the log data, and then the third line extracts the received data. As the length of the log data is fixed in units of 34 bytes, the log data is divided into pre-defined formats and stored, if the data length is normal (unpack process). The unpacked data is used to calculate X- and Y-coordinates and the angle of the robot. These calculated values are stored in the Python variables, which students can use in their programs.

As mentioned above, one of the features of the materials described in this paper is that the module and programs, including the one used inside the robot, are completely open (i.e., white box). We are able to customize the robot itself and the Python module for future requests from students as well as teachers.

5 EXAMPLES OF PROGRAMS USING THE DEVELOPED MATERIALS

To explain the use of the developed materials, it is appropriate to show some program samples. We will show examples of a simple sequence control, usage of loops, usage of functions, and a simple line trace in the following.

5.1 Example 1

The program shown in Fig.4 is a basic program that controls the robot sequentially. After the program is invoked, the robot moves forward for 2 seconds, turns right for 2 seconds, moves forward for 2 seconds, turns left for 2 seconds, and then stops. Each time a command is sent, the next command is issued after the time specified by the “swait()” function. Since the program is written in Python, the program file has the extension “py”. If the name of the program is “sample1.py”, the program is invoked by typing the following command in a terminal window.

```
python sample1.py
```

```
from nxt_bluetooth import nxt_bt
bt = nxt_bt("00:16:53:0c:48:1e", 0)

print "START"
bt.send( "3" )
bt.send( "f" )
bt.swait(2)
bt.send( "r" )
bt.swait(2)
bt.send( "f" )
bt.swait(2)
bt.send( "l" )
bt.swait(2)
bt.send( "0" )
print "END"
```

Figure 4: Example 1—sequential control

5.2 Example 2

The program shown in Fig.5 uses a “while” loop to check the value of a variable repeatedly. The execution leaves the loop if the variable takes a specific value. Here, the robot moves forward 500 mm (50 cm), then it makes a 180-degree left turn. After this it moves 50 cm forward again and then stops. By using the variable “bt.distance” that indicates the total distance from the start point and the “while” loop, the program can control the moving distance quantitatively. When the robot makes a turn, the angle of the robot can also be controlled in the same manner.

```
from nxt_bluetooth import nxt_bt
bt = nxt_bt("00:16:53:0c:48:1e", 0)

print "START"
bt.send( "f" )
bt.send( "3" )
while bt.distance < 500:
    bt.swait(0.04)
bt.send( "l" )
while bt.angle < 180:
    bt.swait(0.04)
bt.send( "f" )
target_dist = bt.distance + 500
while bt.distance < target_dist:
    bt.swait(0.04)
bt.send( "0" )
print "END"
```

Figure 5: Example 2—while loops

5.3 Example 3

The program shown in Fig.6 defines functions concerning an advance and a left turn. Each function takes a parameter:

a distance or an angle. This program makes the robot move forward 300 mm, make a 180-degree left turn, and then move forward 300 mm. The program repeats these actions four times by using the “for” loop.

```
from nxt_bluetooth import nxt_bt
bt = nxt_bt("00:16:53:0c:48:1e", 0)

def forward(distance):
    t_distance = bt.distance + distance
    bt.send( "f" )
    while bt.distance < t_distance:
        bt.swait(0.04)

def left_turn(angle):
    t_angle = bt.angle + angle
    bt.send( "l" )
    while bt.angle < t_angle:
        bt.swait(0.04)

print "START"
bt.send( "3" )
bt.swait(0.04)
for var in range(0, 4):
    forward(300)
    left_turn(180)

bt.send( "0" )
print "END"
```

Figure 6: Example 3—“for” loop

5.4 Value of the light sensor

Figure 7 shows changes in the light sensor value as the robot moves over the course illustrated in Fig.8. In this case, the robot crosses the black line on the course several times to measure the characteristics of its light sensor. The sharp dips observed in Fig.7 occur when the robot crosses the black line. While the robot moves over the white part of the course, the sensor value is approximately 900. When it crosses the black line, the sensor value decreases below 400. Students confirm the characteristics of the light sensor by themselves. Based on these results, black and while colors can be identified by using some threshold value, for example, 700.

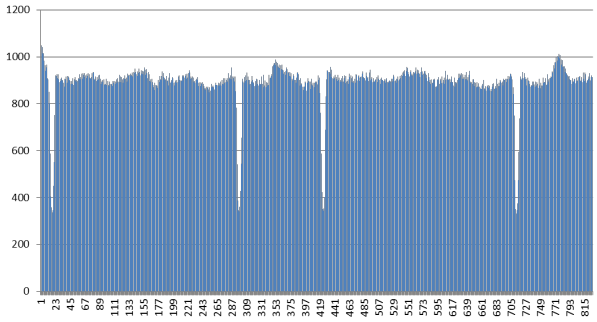

Figure 7: Change of the light sensor value, where horizontal axis is time

If the value of the light sensor is larger than 700, it seems that the robot is running over the white part; otherwise, the robot is running on the black line. Students can know that the robot movement is tracing the black line by using this threshold value.

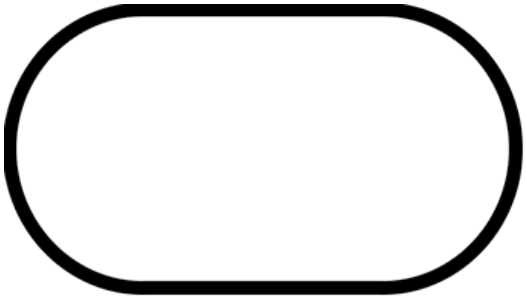

Figure 8: Course for line tracing

5.5 Example 4

Making use of the characteristic of the light sensor and the threshold value, the student can realize line tracing by the robot.

Figure 9 shows the simple program that realizes the line tracing. The variable “target” holds the threshold value. In the infinite “while”, the variable “diff_light” holds the value of the light sensor. If the value of the light sensor is less than the threshold value, the robot makes a right turn; otherwise it makes a left turn. The program repeats this process endlessly every 40 ms. This simple program is able to make the robot move along the black line. Students can learn conditional branching through this example.

```
from nxt_bluetooth import nxt_bt
bt = nxt_bt("00:16:53:0c:48:1e", 0)

print "START"
target = 700
bt.send( "2" )
bt.send( "f" )
while True:
    print bt.diff_light
    if bt.diff_light < target:
        bt.send( "r" )
    else:
        bt.send( "l" )
    bt.wait(0.04)

bt.send( "0" )
```

Figure 9: Example 4—Line tracing by simple control

5.6 The log file

Each time a program is executed, a log file is produced. This file includes a record concerning the details of the robot every 40 ms. Table 3 summarizes the items included in each record of the file. The log data is recorded in CVS format. Figure 10 shows an example of the log file opened in EXCEL. Students can analyze the log file and obtain a

trace of the robot by using some mathematical calculations. Figure 11 shows an example of a trace of the robot obtained from calculations. The trace is almost the same as the course depicted in Fig.8. We can observe zigzag lines in the trace, which is the effect of the simple "ON and OFF" control by the program listed in Fig.9.

Table 3: Items recorded in the log file

Item	Meaning
Time	Elapsed time (ms)
Dt1	PWM value for right motor
Dt2	PWM value for left motor
Batt	Voltage of battery
Mtr1	Rotate angle of tail motor
Mtr2	Rotate angle of right wheel motor
Mtr3	Rotate angle of left wheel motor
ADC s1	Gyro sensor value
ADC s2	Ultrasonic sensor value
ADC s3	Light sensor value
ADC s4	Touch sensor value
I2c	Distance measured by the ultrasonic sensor

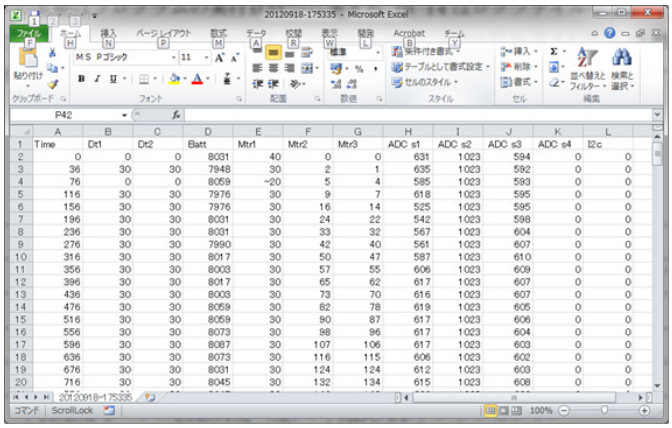

Figure 10: Example of a log file opened in EXCEL

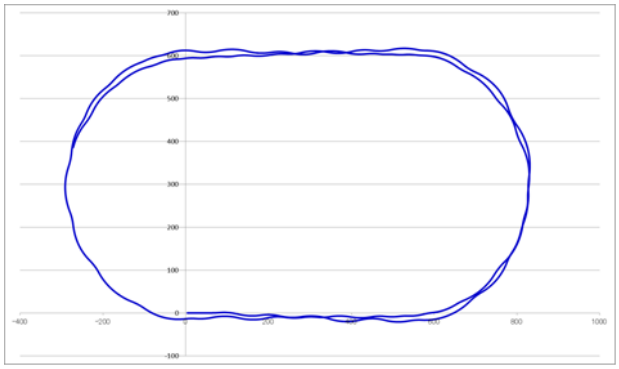

Figure 11: Example of a trace of the robot

6 EXTENSION OF THE PROGRAM FOR CONTROLLING MULTIPLE ROBOTS

The examples explained above concern basic programs that control a single robot. The materials can be applied to the advanced case where multiple robots are controlled by threads. Figure 12 shows the configuration of this case, where two robots are controlled by a single program. A program developed by a student generates two threads for two robots. Each thread executes the same program and controls one of the two robots. Figure 13 shows the sample program for this.

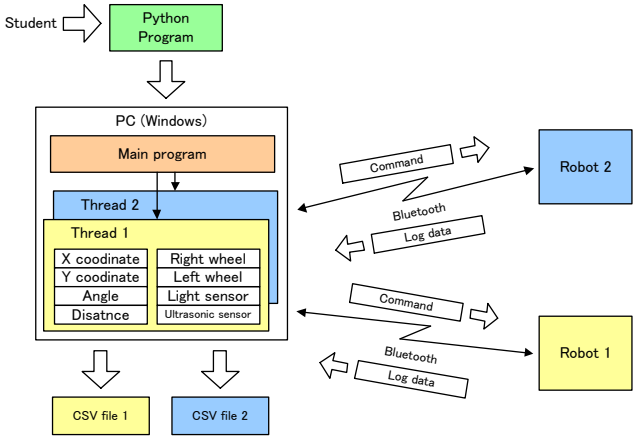

Figure 12: Configuration of the developed materials for controlling two robots

```
import threading    # thread model
import time
from nxt_bluetooth import nxt_bt

class test(threading.Thread):
    def __init__(self, s):
        threading.Thread.__init__(self)
        self.setDaemon(True)
        self.bt = nxt_bt(s, 0)

    def run(self):
        self.bt.start()
        self.bt.send( "3" )
        self.bt.send( "f" )
        self.bt.swait(5)
        self.bt.send( "r" )
        self.bt.swait(5)
        self.bt.send( "0" )

if __name__ == "__main__":
    t1 = test(""00:16:53:0c:48:1e")
    t2 = test("00:16:53:0c:82:39")
    print "Hit enter key, if you are ready."
    raw_input()
    t1.start()
    t2.start()
    time.sleep(20)
```

Figure 13: Program controlling two robots

The part of the program surrounded by the dashed line is the definition of the class that defines the movement of the robots. Two threads are generated from the same class in the main part; the movement of the two robots is the same in this case. Advanced students can learn thread mechanisms through this example.

7 APPLICATION OF THE MATERIALS TO AN ACTUAL CLASS

We applied part of the materials to sessions of an actual experimental class in the first semester of this year. We conducted 9 sessions. The total number of students who participated in the sessions was 45. Approximately half of the students had no experience in programming. The duration of each session was 3 hours and the number of students for one session was at most 7. We explained the material for the first 40 minutes. Two robots were employed to execute the programs. Then students were asked to write four simple programs, including one for line tracing. Although some students had difficulty in understanding Python, 87% of the students indicated a positive impression of the session and expressed their satisfaction when the robot moved correctly. Although students with no programming experience had strong concerns about Python programming, after the session, most of them stated that it was easier than they had thought it would be.

Table 4 shows the summary of their impressions of the materials. Students of Group A had not taken a class in the C programming language as university students, whereas Group B students had, although some students in Group A had learned the C programming language in high school. Students also pointed out aspects of the materials that could be improved.

Table 4: Summary of impressions by students

Impression	Excellent	Good	No comment
Group A	28	1	4
Group B	17	0	1

We also demonstrated the materials and explained a simple program to high school students. A large number of these students found the materials highly interesting.

Through the experience of the actual class, we could identify advantages of the developed materials, summarized as follows:

- The materials could attract more attention from students who had no programming experience.
- As students could edit and execute a Python script directly, program errors were modified quickly. Most of the students could complete the given exercises within the prescribed class hour.
- Students were strongly impressed when the robot performed as they intended.
- Students were also surprised when they got a trace of the robot from the log file.

However, we found drawbacks of the materials from the experience:

- Much effort was needed for preparing and guiding a session.
- Support by teaching assistants was needed for every three or four students to help when they encounter programming problems.
- The difficulty level of programs that students have to develop should be reconsidered. Natural steps from simple to advanced are required.
- We have to improve the assignments before the session to shorten the time needed to explain the materials.
- The number of robots is too small in the case of 7 students.
- Because the quality of components in the robot varied, the robot could not go straight accurately after it received the forward command. Compensation for the error caused this variation in the components is needed.

8 RELATED WORK

Application of robots to education has been prevailing. Target of students are widely spread from elementary and high school students to university students [8][9]. There is also a report on a class using a robot for teacher training [10]. In many cases robots are used for the education of embedded systems including programming [11]-[13]. Robots are also used for object-oriented programming education [14]. Robot contests targeting education have also been widely held [15][16].

In all of the above related work, programming is done for a micro-controller located in a robot itself, while, in our materials of this paper, programming is done for a PC that remotely controls a robot through wireless communication. This remote control architecture realizes easy debugging of a program and quick modification of program errors.

There is NXT Python which provides a Python driver and interface for LEGO Mindstorms NXT, where a NXT robot is remotely controlled by Python programming [17]. Although the control architecture is similar to our materials, there is no report concerning application of NXT Python to programming education. In this NXT Python, the default control program pre-installed in NXT is used and cannot be customized, while, in our materials, both the control program in NXT itself and the Python module in the PC are all developed by ourselves. This means the all functions in our materials are in a while box state easy to modify to cope with the future demand.

Another feature of our materials as compared with the related work is that we intended to cover introductory programming as well as object-oriented programming seamlessly. There is no existing work aiming at this aspect.

Most of the related work employs LEGO Mindstorm® for the target robot. Several products target the education of programming by using the robot of LEGO Mindstorms NXT [1]. The typical product is NI LabVIEW for LEGO MINDSTORMS software [18], which makes it possible for students to develop programs by combining predefined blocks graphically. The difference between our materials and this product is that our materials control the robot remotely by the scripting language Python. Students can learn programming through widely used high-level programming language. As a result, they become accustomed to the conventional programming paradigm.

9 CONCLUDING REMARKS

In this paper we reported the development of teaching materials for computer programming. Our objective is to give beginner students the satisfaction of creating programs that control a robot. Students can implement line tracing by

a simple program consisting of a small number of program steps.

One of the features of the teaching materials is that the module and programs are in a white box state. We are able to flexibly customize the robot itself and the Python module for future requests. Since the characteristics of the students vary depending on the number of students, their interests, their scholastic ability, and characteristics of the university or college, the capability of customizing the materials is considered to be important. We will improve the teaching materials based on the experiences of this semester and extend them for teaching the basics of object-oriented programming.

We are planning to extend the materials by integrating them with the programming language SCRATCH [7], which is intended for students in elementary or junior high school. The boxes surrounded by the dashed lines in Fig.3 show this extension. We will report on this development in the future.

REFERENCES

[1] ET Robot Contest, available from http://www.etrobo.jp/2013/, accessed 2014-04-09.

[2] LEGO Mindstorms education, available from http://www.afrel.co.jp/mindstorms/nxt/, accessed 2014-04-09.

[3] Python, available from http://www.python.org/, accessed 201404-09.

[4] nxtOSEK/JSP, available from http://lejos-osek.sourceforge.net/index.htm, accessed 2014-04-09.

[5] pybluez, available from http://code.google.com/p/pybluez/, accessed 2014-04-09.

[6] BlueZ, available from http://www.bluez.org/, accessed 2014-04-09.

[7] SCRATCH, available from http://scratch.mit.edu/, accessed 2014-04-09.

[8] S. Kato, and H. Tominaga, ``Applied Programming Exercises for Problem Solving Learning with LEGO Robot Control - Teaching Materials and Exercise Problems for Basic Control Practice by ROBOTC -,'' IPSJ SIG Technical Report, 2011-CE-108, No.3, pp.1-10 (2011).

[9] T. Takahashi, and H. Tominaga, ``Game Projects and Simulation Materials of LEGO Robot Control in Introductory Programming Exercises for High School Students,'' Proc. of EC2013, pp.301-304 (2013).

[10] T. Kamada, ``A Report on Measurement and Control Class Using a Robot for Teacher Training Course Students,'' IPSJ SIG Technical Report, 2009-CE-99, No.11, pp.1-6 (2009).

[11] E. Hayakawa, T. Takahashi, and K. Aoshima, ``Experiment on Embedded System Education using Robot in Computer Science Course,'' IPSJ SIG Technical Report, 2009-CE-98, No.15, pp.127-134 (2009).

[12] H. Nishigaya, H. Aoki, S. Inoue, K. Eguchi, and S. Kurebayashi, ``Lessons of Learning Measurement and Control with an Autonomous 3 Motor Control Robot,'' IPSJ SIG Technical Report, 2009-CE-98, No.15, pp.113-120 (2009).

[13] Y. Nishino, and E. Hayakawa, ``A Note on Robot Based Embedded System Study Environment in Technical High School,'' IPSJ Journal, Vol.51, No.12, pp.2261-2272 (2010).

[14] N. Chubachi, and K. Ito, ``A Trial of Experimental Task using LEGO® in Object Oriented Programming Education,'' IPSJ SIG Technical Report, 2014-CE-124, No.8, pp.1-6 (2014).

[15] H. Yamashita, ``A Robot Contest for Children and Comprehensive Science Education,'' IPSJ Magazine, Vol.48, No.5, pp.502-511 (2007).

[16] K. Hisazumi et al., ``A Distributed Project Based Learning Curriculum to Design Embedded Systems using Contest Challenge,'' IPSJ SIG Technical Report, 2014-EMB-32, No.34, pp.1-6 (2014).

[17] NXT_Python, available from http://home.comcast.net/~dplau/nxt_python/, accessed 2014-04-09.

[18] NI LabVIEW for LEGO® MINDSTORMS®, available from http://www.ni.com/academic/mindstorms/, accessed 2014-04-09.

(Received December 9, 2013)
(Revised April 10, 2014)

Toshihiro Shikama received his B.E and M.E. degrees from Tokyo Institute of Technology in 1974 and 1976, respectively. From 1984 to 1985, he stayed at University of Waterloo. He received his Ph.D. degree from Shizuoka University in 2006. He joined Mitsubishi Electric Corp. in 1976 and had engaged in developing a computer network using a satellite channel, high speed ring type LANs, time division multiplexers, ATM equipment, a high speed IP switch, and network security systems. Since April of 2007, he has been a professor of Department of Electrical, Electronic and Computer Engineering, Fukui University of Technology. He is a member of IPSJ, IEICE, and IEEE Communications Society.