지능정보연구 제15권 제3호 2009년 9월(pp.17~29)

Java based Platform for Educational Robots on AVR

Lee-Sub Lee
Kumoh Nationa. Institute of Technology
(eesub@kumoh.ac.kr)

Seong-Hoon Kim
Kyungpook National University
(shkim1454@knu.ac.kr)

C programming is a main programming for the Educational Robot Arm which is based on AVR ATmega128. The development environment is not integrated, so it is complex and difficult to study for middle or high school students who want to learn programming and control the educational robot arm. Furthermore, there is no debug and testing environment support. This paper presents a Java-based development platform for the educational robot arm. This platform includes: an up-to-date tiny Java Virtual Machine (NanoVM) for the educational robot arm; An Eclipse based Java integrated development environment as an Eclipse plug-in; a 3D simulator on the PCs to support testing and debugging programs without real robots. The Java programming environment makes development for educational robot arm easier for students.

Received : June 20, 2009 Revision : July 05, 2009 Accepted : July 30, 2009 Corresponding author : Seong-Hoon Kim

1. Introduction

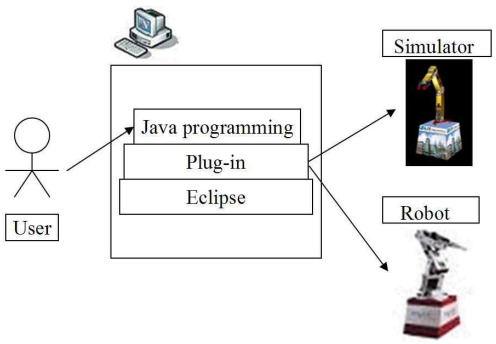
The education robot arm (Dajin Co, http://dajin.kr.ec21.com/.) is one of products of DAJIN ROBOT CO.LTD. This robot uses AVR ATmega128 (Amtel Co, http://ww w.datasheetcatalog.com/datasheets_pdf/A/T/M/E/ATMEGA128.shtml.) as it's Micro Control Unit (MCU). It supports serial/parallel programming and remote control. The education robot arm simulates a human's arm to grasp something and to move it somewhere. This robot has five motors to control five joints, so it can do many complicated actions similar to a real arm. Users can also control robot action with serial port or remote device according to a program. This robot is popular in the education field, many people use it to learn how to program and control a robot.

Conventional development language for educational robot arm is C language. But in practice, this C development environment is difficult to learn and requires time-consuming. First, it is

difficult to learn for young students, since C language is invented and used for system programming purpose. Users must manually execute several steps to compile and use the several tools to test even a simple program, because there is no integrated development environment (IDE). Furthermore, there is no debugging support in this environment. Because of these limitations, the middle or high school students feel many difficulties to learn programming robots.

This paper presents a Java-based development platform for education robot arm so it is easy to learn and provides an easy to use IDE. The platform includes two layers. Firstly, a tiny Java virtual machine (JVM) is ported to the education robot arm. The JVM is extended from NanoVM (Harbaum, 2005) which is an open source project and initially developed to run on the Atmel AVR ATmega8. The process of porting the JVM to the education robot arm is similar to that of C programming language for educational robot arm, because the JVM is developed by C language. Some modifications are performed to adapt to the target system, because the open source have some bugs. Secondly, we focused on Java programming. A Java IDE based on Eclipse is implemented with an Eclipse plug-in development technology (Holzner, 2004). This IDE allows creating a robot project and uploading the object code to the robot with one step, therefore users can only focus on the Java programming itself. Furthermore, a 3D simulator is designed to support testing programs without real robots. Based on Eclipse, this simulator also supports debugging the program without real robots. This Java development environment is showed in <Figure 1>.

Based on this Java-based development platform for education robot arm, the user can easily program the Java language in the popular Eclipse environment. The final program can be uploaded to the robot or run on the 3D simulator. All the process is supported by GUI even without the real robot.

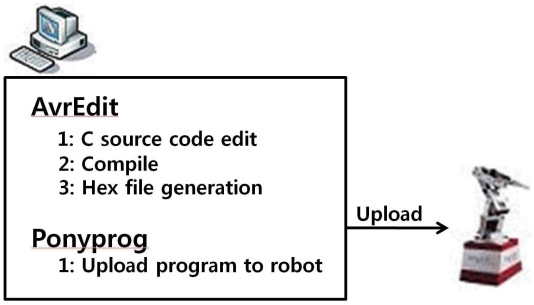


<Figure 1> The Java development environment for the robot arm

The rest of the paper is organized as follows. In section 2, some related works are given and background techniques are introduced in section 3. The Java-based development platform for education robot arm is presented in section 4. A sample Hanoi tower application for this IDE is introduced in section 5. Conclusions and future works are discussed in section 6.

2. Related Works

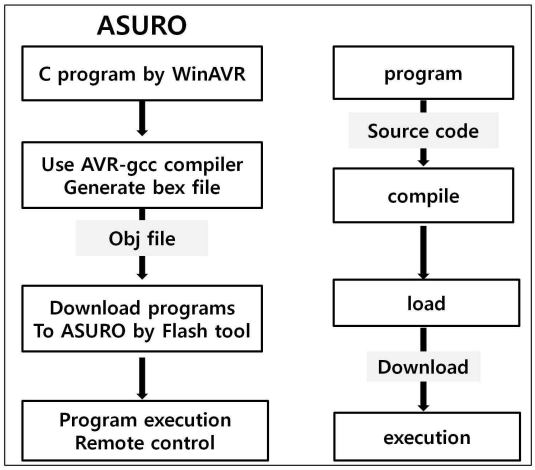
Conventional development environment for



<Figure 2> Conventional C development environment

educational robot arm is C programming by AvrEdit and uploading program by PonyProg (Lancos Co, htt p://www.lancos.com/e2p/ponyprog2000.html.). This process is showed in <Figure 2> In this development environment user must concern about microcontroller type on that target is based, even though a user wants to make a simple application to control the robot. In this environment there is no debugging and simulator support. These make several difficulties for middle or high school students.

ASURO is another popular robot toolkit for DLR School Lab (Tick, 2006). As shown in <Figure 3> it supports C language and also provides similar development environment for educational robot arm. This development environment is difficult for middle or high school students also because of C language, no IDE, no debugging function, and no simulator support. It supports NanoVM so ASURO can support Java. But there is also no debugging and simulator support for ASURO robot, while in this paper, a 3D simulator for educational robot arm is implemented to support debugging and simulation functions without the real robot.



<Figure 3> The C development procedure for ASURO robot

3. Background

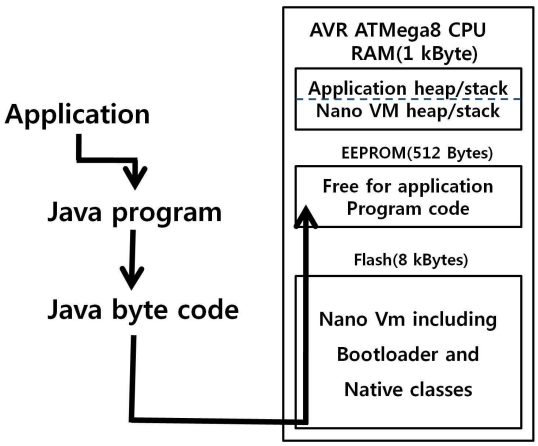
3.1 Tiny Java Virtual Machine (NanoVM)

JVM is a set of computer software and data structures which implements a specific virtual machine model. This model accepts a form of computer intermediate language, commonly referred to as Java bytecode, which conceptually represents the instruction set of a stack-oriented, capability architecture. JVMs using the “Java” trademark may be developed by other companies as long as they adhere to the JVM standard published by Sun and related contractual obligations.

There are many kinds of JVM implementations: some are proprietary implementations and the others are open source implementations. NanoVM is an open source implementation of JVM. It is not a full featured JVM, only limited to a small subset of the Java language and the standard Java libraries and a few application specific

methods. The NanoVM was initially developed to run on the Atmel AVR ATmega8 used in the Asuro Robot. But it can be ported to other AVRs. Since the educational robot arm is based on AVR ATmega128, we choose the NanoVM as the JVM for the educational robot arm.

The original NanoVM uses almost 8K bytes of code memory (entire flash in case of ATmega8) and 256 bytes of RAM. User's class file is processed by NanoVM's converter, sent through serial line to NanoVM's boot loader and stored in on-chip EEPROM. Finally, NanoVM runs the class file stored in EEPROM until finished. This process is showed in <Figure 4>



<Figure 4> The Java development environment for AVR ATMega8 CPU based on NanoVM

3.2 Eclipse

Eclipse (Eclipse Foundation, http://www.eclipse.org/.) is an open-source, platform-independent software framework, written primarily in Java, for delivering what the project calls "rich-client applications", as opposed to "thin client" browser-based applications. So far this framework has typically been used to develop IDEs, such as the Java IDE called JDT (Java Development Toolkit) (Eclipse Foundation, http://www.eclipse.org/jdt.) which has been the most popular Java IDE and compiler that comes as part of Eclipse.

Eclipse was created as a platform for plug-in tools (Clayberg, and Rubel, 2006) that extend the IDE's capabilities so that it can work with numerous programming languages and applications. Anyone can write plug-ins for Eclipse and have them work directly with any other plug-ins for the platform.

The Plug-in Development Environment (PDE) (Gallardo, Brunette, and McGovern, 2003) is freely distributed as a part of the Eclipse SDK. The PDE provides tools to create, develop, test, debug, build and deploy Eclipse plug-ins, fragments, features, update sites and RCP products. In this paper, an Eclipse plug-in is implemented for Java programming for the educational robot arm which consists of a code editor, a compiler, a program uploader and a debugger. The details will be given in section 3.2.

3.3 OpenGL with SWT

OpenGL (Wright, 2000) is a universal, sharing and open 3D graphical standard sparkplug to develop by many famous graphics and CAD companies such as SGI, which is based on 3D graphics library of GL. It has been a graphical standard in truth for its strong graphical functions and the ability of working on different platform.

As of Eclipse 3.2, OpenGL support has migrated into the org.eclipse.swt (Eclipse Foundation, http://www.eclipse.org/swt/opengl/). project. Standard Widget Toolkit (SWT) is a graphical widget toolkit for Java platform which is an alternative to AWT and Swing Java GUI toolkits. In this paper, a simulator of educational robot arm is developed with OpenGL in SWT which uses org.eclipse. opengl Java binding. This simulator can be easily integrated to Eclipse based Java development environment for educational robot arm, so it is easy to run the program on the 3D simulator to test and debug.

4. Java-based Development Platform for Educational Robot Arm

This Java-based development platform for educational robot arm includes two layers. On the first layer, a tiny JVM (NanoVM) is ported to the education robot arm. The second layer focuses on Java programming with a Java IDE based on Eclipse and a 3D simulator.

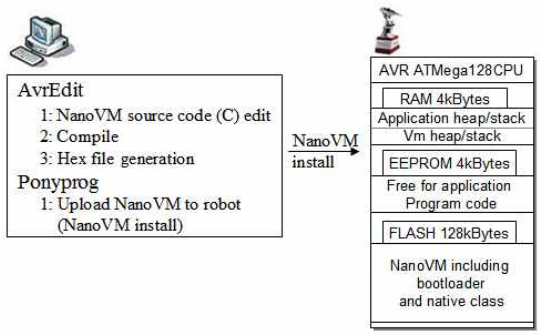
4.1 NanoVM Installation

NanoVM was initially developed to run on the Atmel AVR ATmega8 that is used in the Asuro robot while the educational robot arm is based on Atmel AVR ATmega128. So some modifications are made for the NanoVM for the educational robot arm. The important updated parts are collected as:

- Initializing the board of the educational robot arm.
- Updating the configuration for ATmega 128: Because the memory size of ATmega 128 is larger than ATmega8. Some configurations such as code size, heap size, and EEPROM size should be extended for running bigger Java application.
- Implementing native methods for the educational robot arm: NanoVM was initially developed for Asuro Robot, so native methods for the educational robot arm should be implemented. We create 14 native methods for educational robot arm. shown as follows;

```
{
#define NATIVE_METHOD_MOVE                1
#define NATIVE_METHOD_WAIT                2
#define NATIVE_METHOD_MOVEMOTOR1          3
#define NATIVE_METHOD_MOVEMOTOR2          4
#define NATIVE_METHOD_MOVEMOTOR3          5
#define NATIVE_METHOD_MOVEMOTOR4          6
#define NATIVE_METHOD_MOVEMOTOR5          7
#define NATIVE_METHOD_MOVEMOTORALL        8
#define NATIVE_METHOD_LED                 9
#define NATIVE_METHOD_BEEP               10
#define NATIVE_METHOD_SEGMENT7           11
#define NATIVE_METHOD_HANOI              12
#define NATIVE_METHOD_HOLDRING           13
#define NATIVE_METHOD_PUTRING            14
}
```

After updating the parts mentioned above by AvrEdit, source code should be compiled and a HEX file is generated. With Ponyprog, the updated NanoVM was uploaded to the educational robot arm by parallel cables. This process is shown as <Figure 5>.



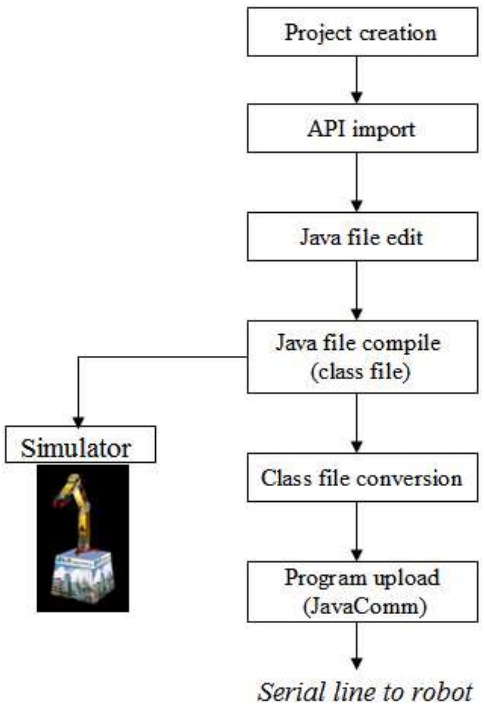
<Figure 5> The NanoVM installation procedure

4.2 Java IDE for Educational Robot Arm

The functions for the Eclipse plug-in for the educational robot arm are collected as follows:

1 : The educational robot arm project wizard.
2 : The educational robot arm API importing.
3 : Class files conversion and uploading.
4 : Java program executer on 3D simulator.
5 : Java program debugger based on 3D simulator

The procedure of using the Java IDE for the educational robot arm programming is shown in <Figure 6> Firstly, a robot project can be created interactively with a project wizard. The robot API is imported automatically to the project path. Because this IDE is based on Eclipse platform, a default Java IDE supports a Java editor, auto-compile and a debug environment. For uploading this converted class files to robot, Java communications API (JavaComm) (SunMicro Co, http://java.sun.com/products/Javacomm/.) is used. This API provides applications that access RS-232 hardware.



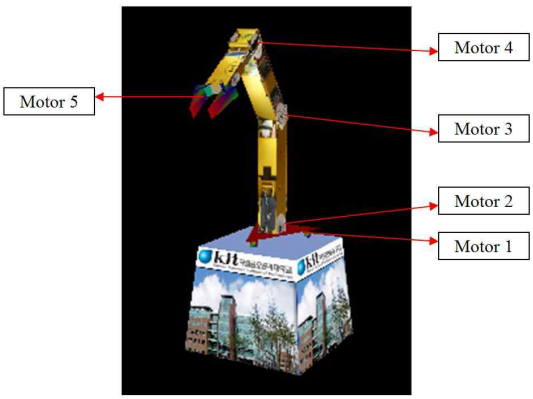
<Figure 6> The system architecture of the Java IDE

For mapping the native methods that is defined in JVM in the educational robot arm, a related Java API should be implemented and all the methods should be registered in configuration file as follows:

```
Java API :
 public class ArmAction {
.....
public static native void wait(int msec) ;
public static native void beep(int msec) ;
public static native void segment7(int number) ;
```

```
public static native void moveMotor1(int data) ;
.....
}
```

```
Configuration file :
class   ArmAction 28
method <init> : ()V                    0
method move : ()V                      1
method wait : (I)V                     2
method moveMotor1 : (I)V               3
method moveMotor2 : (I)V               4
method moveMotor3 : (I)V               5
method moveMotor4 : (I)V               6
method moveMotor5 : (I)V               7
method moveMotorAll : (IIIII           8
method led : (IIIIIIII)V               9
method beep : (I                       10
method segment7 : (I)V                 11
method hanoi : ()V                     12
method holdRing : (I)V                 13
method putRing : (I)V                  14
```



<Figure 7> A 3D simulator for the robot arm

As shown in <Figure 6> Java programs also can run on the 3D simulator. This simulator is developed with OpenGL in SWT which uses org.eclipse.opengl Java binding. The real robot arm has five motors, so the 3D simulator also simulates five motors as shown in <Figure 7>.

The structure of these five motors is developed like <Figure 8>. A base object “Motor” is created to draw the 3D simulator. With using OpenGL, each object, no matter how complicated it is, is drawn with primitive shapes. For example, to draw a Motor3 which is a quad, the following code is given:

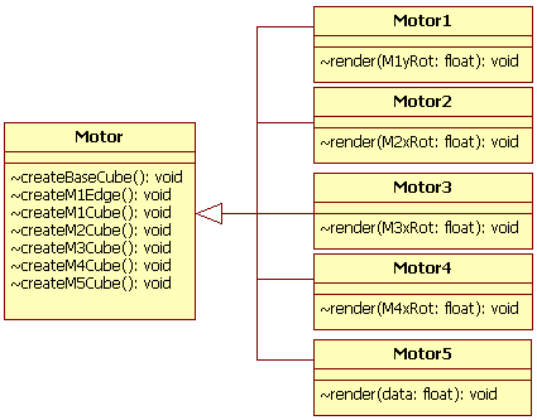
```
GL.glBegin(GL.GL_QUADS);
// front face
GL.glVertex3f(-0.2f, -1.2f, 0.2f) ; // bottom left
GL.glVertex3f(0.2f, -1.2f, 0.2f)  ; // bottom right
GL.glVertex3f(0.2f, 1.2f, 0.2f)   ; // top right
GL.glVertex3f(-0.2f, 1.2f, 0.2f)  ; // top left

// back,  left, right, top, bottom
.....
GL.glEnd() ;
```

For each motor, one object which extends the base object “Motor” is implemented. The main function of these child objects is to manage the transformation of each Motor when the location data is changed. Because all motors are associated, transformation of one motor should be considered the influence on other motors. A nested structure is implemented to satisfy this requirement. The following code for Motor3 shows this structure:

```
// Transform Motor3 which influences Motor4
// and Motor5
GL.glPushMatrix();
```

```
GL.glTranslatef(0f, 1f, 0f) ;
GL.glRotatef(Motor3.CurM3xRot, 1, 0.0f, 0.0f) ;
// Motor4 is nested in Motor3
// Transform Motor4 which influences Motor5
GL.glPushMatrix() ;
    .....
    // Motor5 is nested in Motor4
    // Transform Motor5
    GL.glPushMatrix() ;
        .....
    GL.glPopMatrix() ;
GL.glPopMatrix() ;
GL.glPopMatrix() ;
```



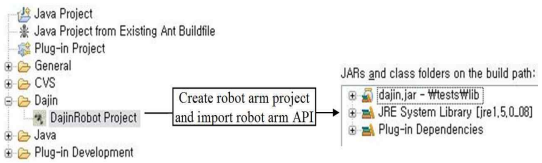
<Figure 8> The structure of five motors in the simulator

5. Application

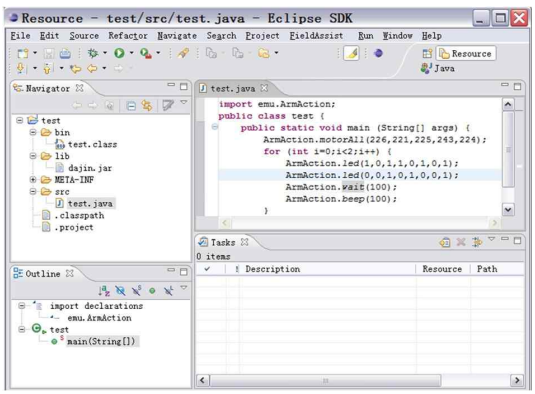
After porting NanoVM to the educational robot arm and installing the Eclipse plug-in to Eclipse, we can program Java language to run on the real robot or run on the 3D simulator.

Firstly, a robot project is created by wizard and import the API automatically as shown in <Figure 9>. After creating a project, Eclipse will show a tree style project view. Then we can program Java language for educational robot arm as shown in <Figure 10> while the compile is automatically supported.

In the project tree view, a right button menu of the Java file there is a menu “run (Dajin robot)”, With clicking this menu, the class file will be converted and then uploaded to the educational robot arm by serial port. The execution logs will be recorded in a log view. After uploading, robot will start JVM to run this Java program.



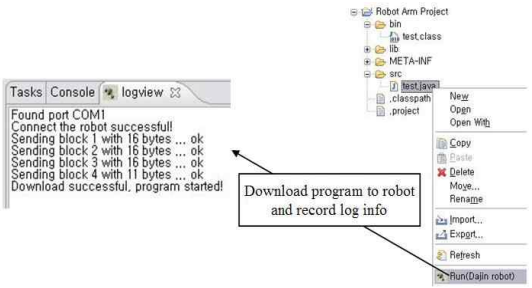
<Figure 9> The robot project wizard



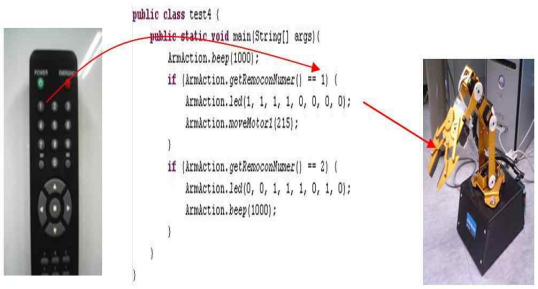
<Figure 10> A Java program for the robot arm

The real robot arm can be controlled by a remote device. That means the real robot can receive external events to run program interactively by C programming. As shown in <Figure 12>,

the robot pauses running when an external event function appears; here is waiting the remote device's signal. After received the external event (clicking key "1" of remote device), the robot will continue running according the program.



<Figure 11> A screenshot of the program uploading to the robot arm



<Figure 12> A screenshot of program running with remote device

Only by adding one line: import emu.Arm Action, we can run a program on the 3D simulator. The simulator will execute like <Figure 13>. This simulator supports the simulation of LCD, LED, 7 segment, beep, and robot arm. Almost all the real environments can be simulated. This will be easy to test the program and reduce cost since real robot is not necessary.

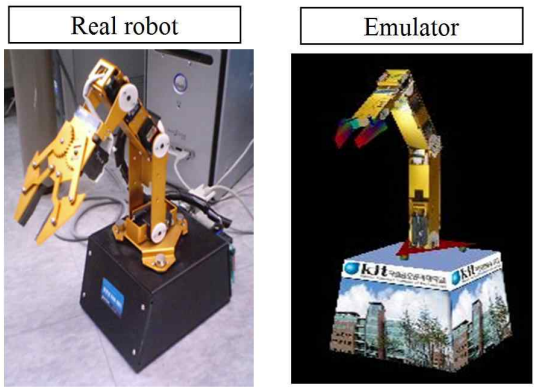
<Figure 13> A screenshot of a program running on the 3D simulator

Based on the same source code, we can compare the execution between real robot and 3D simulator as show in <Figure 14>. The locations of all 5 motors are almost same. That proves the 3D simulator can simulate the real robot on PC. Eclipse platform supports a debugging environment, so we can debug the program based on the 3D simulator like <Figure 15> without real robot. This will reduce the cost for development or study for young student.

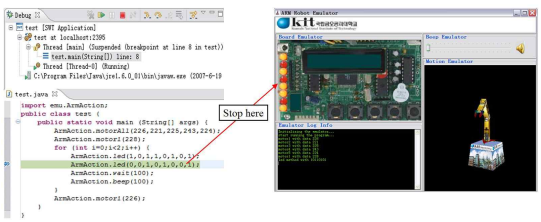
To show the advantage of this approach for educational field better, a Hanoi application is implemented by Java for robot arm. The execution is showed in <Figure 16>. The following Java source code shows the main implementation.

```
static void Hanoi(int n, int a, int b, int c) {
  if (n > 0) {
   if (n == 1) {
      ArmAction.moveMotor1(a);
      ArmAction.holdRing(a);
      ArmAction.moveMotor1(c);
      ArmAction.putRing(c);
   } else {
      Hanoi(n-1, a, c, b);
      ArmAction.moveMotor1(a);
      ArmAction.holdRing(a);
      ArmAction.moveMotor1(c);
```

```
            ArmAction.putRing(c);
            Hanoi(n - 1, b, a, c);
        }
    }
}
```

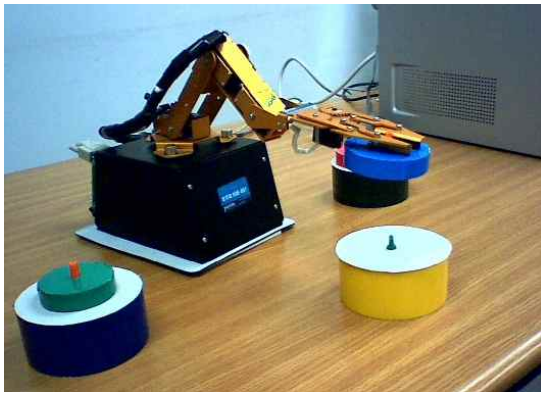


<Figure 14> An execution comparison between real robot and 3D simulator



<Figure 15> A screenshot of debugging based on 3D simulator

Comparing to the C language, this Java programming only has about 20 lines source code. The user can only focus on the Hanoi application itself to study Hanoi algorithm and learn how to program robot arm. A existing Hanoi tower C program includes more 500 lines source code and user should spend much time on hardware programming and that makes difficult for middle or high school students.



<Figure 16> A screenshot of Hanoi application for the robot arm

6. Conclusion and Future Work

Because the traditional C language based text programming environment is very difficult for the middle and high school to learn robot programming, this paper presents a Java-based development platform for the education robot arm.

The contributions of the system are follows. Firstly this provides a Tiny Java Virtual Machine on ATMega128 that has very poor capability. Therefore based on this platform, Java programming for educational robot arm becomes much easier than C programming for middle or high school students. Secondly Eclipse plug-in for robot arm supports IDE for developing Java applications for robot arm. This makes that user will not suffer from bothering complicated handing of editing, compiling, unloading, and execution of robot programs. Finally it provides a 3D simulator for the educational robot arm can test and debug the program without real robot. This will enhance efficiency, productivity, and convenience

of development.

This Java-based development platform now only can be used for a specific education robot arm. The future work will concern about a generic architecture for other robots.

References

Dajin Co. “Dajin five axis robot arms for education”, http://dajin.kr.ec21.com/.

Amtel Co. “ATMEGA128 datasheet”, http://www.datasheetcatalog.com/datasheets_pdf/A/T/M/E/ATMEGA128.shtml.

Harbaum, T., “The NanoVM-Java for the AVR”, http://www.harbaum.org/till/nanovm. (2005).

Holzner, S., *Eclipse Cookbook*. O’Reill., (2004).

Lancos Co. “PonyProg2000 documentation”, htt p://www.lancos.com/e2p/ponyprog2000.html.

Tick, J., “Modeling Control Systems by Autonomous Mini Robot”, *Proceedings of 3rd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence*, (2006), 283～290.

Eclipse Foundation, http://www.eclipse.org/.

Eclipse Foundation, “Java Development Tools (JDT)”, http://www.eclipse.org/jdt.

Clayberg, E. and D. Rubel, *Eclipse: Bu ilding Commercial-Quality Plug-ins (2nd ed.)*. Addison-Wesley Professional, (2006).

Gallardo, D., E. Brunette, and R. McGovern, *Eclipse in Action*. MANNING, (2003).

R. Wright, *OpenGL super bible*. Waite Group Press, (2000).

Eclipse Foundation, “Using OpenGL in SWT Applications”, http://www.eclipse.org/swt/opengl/.

SunMicro Co, “Core Java: Java communications”, http://java.sun.com/products/Javacomm/.

Abstract

교육용 AVR 로봇의 자바기반 플랫폼

이이섭* · 김성훈**

C 언어는 AVR사의 ATmega128을 기반으로 하는 교육용 로봇 팔의 주 프로그래밍 언어다. C 언어는 교육용 로봇 팔을 제어하고 프로그램 학습하기 원하는 중학교 또는 고등학생들에게는 매우 어렵고 복잡하였다. 더구나, 통합개발환경, 디버딩 및 테스팅 환경도 제공하지 못하였다. 본 연구에서는 이러한 문제를 해결하기 위하여 교육용 로봇을 위한 자바 기반의 통합 개발 플랫폼을 제안하였다. 이 플랫폼은 교육용 로봇 팔을 위한 최신의 초소형 자바 가상 머신(NanoVM), 이클립스 플러그인을 사용한 통합개발환경, 실제 로봇을 연결하지 않아도 테스트 및 디버깅을 할 수 있는 3차원 시뮬레이터를 포함하고 있다. 이러한 자바 개발환경은 어린 학생들이 매우 용이한 교육용 로봇 팔 학습환경을 제공하게 되었다.

Keywords : 교육용 로봇, 자바 가상 머신, 시뮬레이터, 디버거, 통합개발환경

* 금오공과대학교 컴퓨터공학부 조교수
** 경북대학교 컴퓨터정보공학부 조교수

저 자 소 개

이이섭
서강대학교에서 수학과 학사(1988) 및 전자계산학과 석사(1990)를 취득하였으며, 고려대학교 정보통신대학 컴퓨터과에서 박사학위(2004)를 취득하였다. 1990년부터 2004년까지 삼성 SDS 정보기술연구소에서 수석연구원으로 일했으며, 2004년 9월부터 현재까지 금오공과대학교 컴퓨터공학과에서 조교수로 재직 중에 있다. 2007년 12월부터 2009년 1월까지 미국 콜로라도주립대 방문교수로 역임하였다. 주요 관심분야로는 소프트웨어공학, 모바일 및 임베디드 컴퓨팅, 지식관리, 등이다.

김성훈
서강대학교 전자공학과 학사(1988)를 졸업하고 연세대학교 전자공학과에서 석사(1990) 및 공학박사(1996)를 취득하였다. 1996년부터 2006년까지 영동대학교 컴퓨터공학과에서 부교수를 역임하였고, 그 후 현재까지 경북대학교 컴퓨터정보학부에서 조교수로 재직 중에 있다. 주요 관심분야로는 로봇비전, 지식표현, 지능형 미디어 및 소프트웨어 등이다.