TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics, Mechanics, and Engineering
Vol. 60, Issue I, March, 2017

RASPEBRRY PI 3 PROGRAMMING, IN JAVA, USING BLUE J AND JDEVELOPER BASED ON Pi4J

Tiberiu Alexandru ANTAL

Abstract: *The paper gives a step by step guide in preparing the Rapsberry Pi 3 board for Java programming under Raspbian operating system using the BuleJ IDE for Raspbian and JDeveloper IDE for Windows using the Pi4J Java I/O library for Raspeberry Pi open source project.*
Key words: *BlueJ, Java, JDeveloper, Pi4J, Raspberry Pi 3, Raspbian.*

1. INTRODUCTION

The Raspberry Pi 3 is the last from a series of single board computers developed in the United Kingdom by the Raspberry Pi Foundation. Eben Upton's goal - the creator of Raspberry Pi - was to create a low-cost device that would improve programming skills and hardware understanding of computers at the pre-university level. The Raspberry Pi was designed for the Linux operating system, and some Linux distributions now have a version optimized for the Raspberry Pi. Today the most popular operating system is Raspbian, which is based on the Debian operating system. Multiple operating systems can be installed easily on the board with the help of NOOBS, which stands for New Out Of Box Software and can be downloaded from https://www.raspberrypi.org/downloads/noobs/. The NOOBS_v2_1_0.zip file contains a group operating systems, among them are Raspbian, Ubuntu MATE and Windows 10 IoT Core, that can be used with the board. Figure 1 shows the contents of the zip file that must be unzipped and copied on a microSD card. The microSD card must be formatted previously using SDFormatter tool that can be downloaded for Windows from the following link https://www.sdcard.org/downloads/formatter_4/index.html. In order to install the operating system to the board this must be booted with the formatted microSD containing exactly the files (not stored in some parent directory) from Figure1.

Fig. 1. – Contents of the NOOBS_v2_1_0.zip file.

Depending on the speed of the microSD card and the numbers of chosen operating systems

the installation process varies between minutes and hours. After the installation ends the contents of the microSD will be lost and the selected operating system(s) will occupy that space and use it further to operate with the board. It might be possible that the microSD will be able to store only some of the selected operating systems, in this case copy reinstall everything with a single operating system. For example, Raspbian and Windows 10 IoT Core will occupy about 12 Gb installed (unpacked) on the microSD, so they won't fit to an 8 Gb microSD though the full image of NOOBS is packaged under 2 Gb. Once the operating system is installed the Raspbian desktop will be shown as seen in Figure 2.

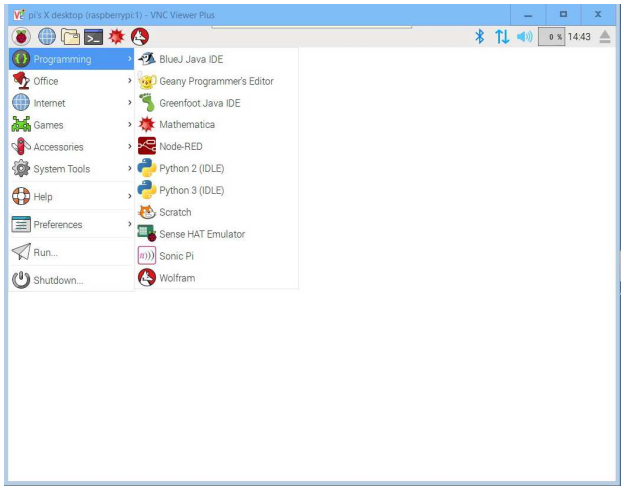

Fig. 2. – Raspbian operating system desktop.

The November 2016, release date: 2016-11-25, kernel version: 4.4 will contain the java virtual machine (JRE), the JDK, a Java programming tool called BlueJ Java IDE preinstalled.

2. INSTALLING PI4J UNDER RASPIAN ON RASPBERRY PI 3

Programming the Raspberry Pi in Java is based on the Pi4J project that can be found at http://pi4j.com/. The simplest way to install Pi4J on your RaspberryPi is to execute the following command directly on the RaspberryPi terminal windows:

```
curl -s get.pi4j.com | sudo bash.
```

Alternative methods to install and to manage the project can be found at http://pi4j.com/install.html. If the simple installation method succeeds the terminal screen will look like the one from Fig. 3 showing the "Pi4J INSTALLATION COMPLETE" message.

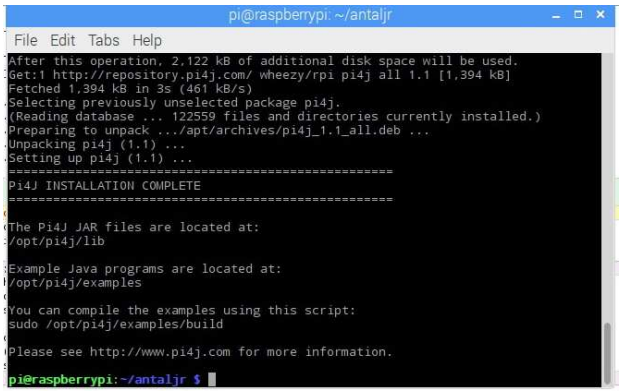

Fig. 3. – Pi4J installation final screen.

3. CREATING A JAVA CODE FOR BLINKING AND DIMMING A LED ON RASPEBRYY PI IN BLUEJ

Form version 3.14 the *BlueJ Java IDE* is installed on Raspbian (the Novemeber 2016 release) under the *Programming* menu as shown in Figure 2 designed for the teaching of java for beginners. BlueJ is a Java integrated development environment that allows development as well as program execution on the Raspberry Pi. At the following link http://www.bluej.org/raspberrypi/ some Pi applications are described in order to help the programmer to start working with the IDE.

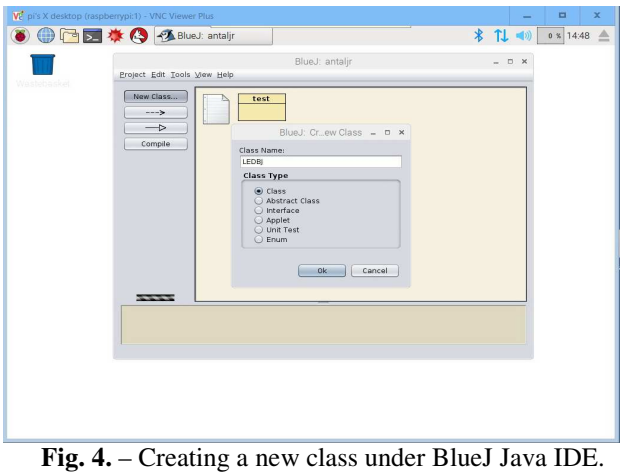

Fig. 4. – Creating a new class under BlueJ Java IDE.

We create a new class (Fig. 4) named *LEDBJ* under the *antaljr* directory. Double-clicking the *LEDBJ* rectangle from Fig. 5 will start the Java code editor from Fig.6. If the code compiles without errors then if can be executed from the Terminal window typing in the following line:

```
sudo java -classpath .:classes:/opt/pi4j/lib/'*' LEDB
```

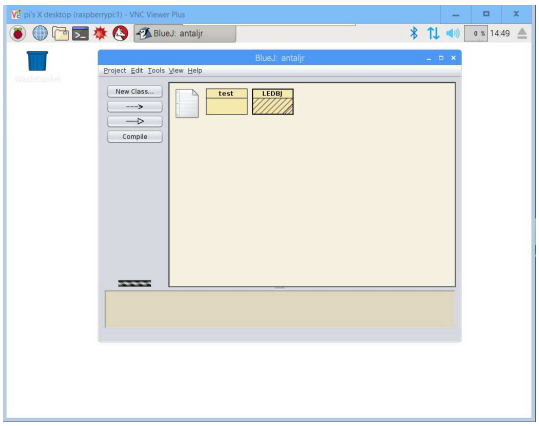

Fig. 5. – Launching the BlueJ code editor.

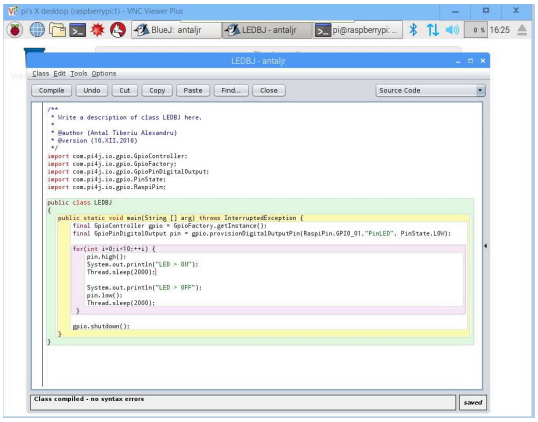

Fig. 6. – The BlueJ code editor.

The Java code that follows will light up and dim a LED connected to the Raspberry Pi 3 as shown in Figure 7.

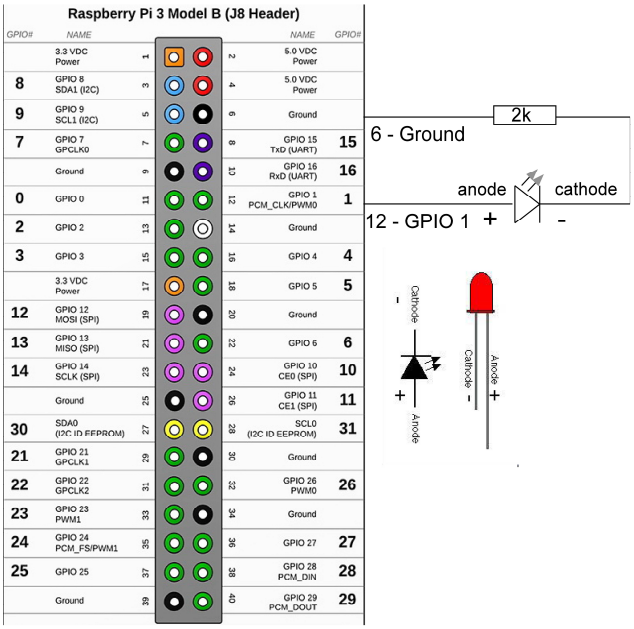

Fig. 7. – The wiring diagram to connect a LED to Pi.

```
/**
 * Write a description of class LEDBJ here.
 *
 * @author (Antal Tiberiu Alexandru)
 * @version (10.XII.2016)
 */

import com.pi4j.io.gpio.GpioController;
import com.pi4j.io.gpio.GpioFactory;
import com.pi4j.io.gpio.GpioPinDigitalOutput;
import com.pi4j.io.gpio.PinState;
import com.pi4j.io.gpio.RaspiPin;
import com.pi4j.wiringpi.Gpio;
import com.pi4j.wiringpi.SoftPwm;

public class LEDBJ
```

```
   {
      public static void main(String [] arg) throws InterruptedException {
          final GpioController gpio = GpioFactory.getInstance();
          final              GpioPinDigitalOutput              pin              =
gpio.provisionDigitalOutputPin(RaspiPin.GPIO_01,"PinLED", PinState.LOW);

          for(int i=0;i<3;++i) {
              pin.high();
              System.out.println("LED > ON");
              Thread.sleep(2000);
              System.out.println("LED > OFF");
              pin.low();
              Thread.sleep(2000);
           }

           for(int i =100;i<=1000;i+=100) {
               pin.pulse(i,true);
               Thread.sleep(100);
           }

           int pinno = pin.getPin().getAddress();
           SoftPwm.softPwmCreate(pinno,0,100);
           for(int i=0;i<=100;++i) {
               SoftPwm.softPwmWrite(pinno,i);
               Thread.sleep(100);
           }

           for(int i=100;i>0;--i) {
               SoftPwm.softPwmWrite(pinno,i);
               Thread.sleep(100);
           }
          gpio.shutdown();
      }
   }
```

4. CREATING A JAVA CODE FOR BLINKING A LED ON RASPEBRRY PI IN JDEVELOPER

JDeveloper IDE can be used to create many types of simple applications starting from GUIs to CAD programming [1] - [4]. However, JDeveloper, is a mature IDE from Oracle that can be used to create and deploy Java EE applications [5] - [9] too. The ORACLE Fusion Middleware version of the IDE is able to generate and deploy executable jar files. In order to create Java code for Raspberry Pi 3 the same Pi4J project must be used, however the Windows version zip file must be, this time, downloaded and unpacked. The code from the previous example used the Pi4J library installed under the Raspbian OS, however, in this example, due to the more advanced IDE, I will create an archive that includes all de parts from Pi4J necessary to run the application independently. In order to do that the unzipped *[pi4j-1.1]* directory must be copied as shown in Fig. 8. The application (Fig. 8) is called *AppRaspberryPi*, the project is called *Pi* and the result of the deployment will be stored in the *[deploy]* directory. The LEDPi class that follows will blink a LED attached to Raspberry Pi 3 based on Fig. 7.

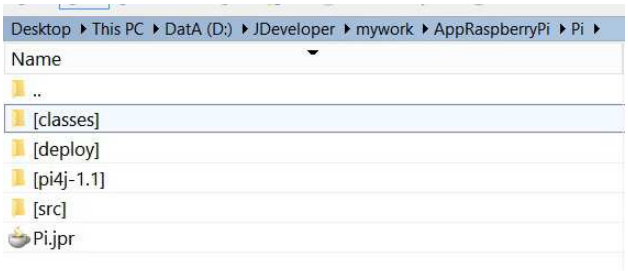

Fig. 8. – JDeveloper directory structure to create jar files for Raspberyy Pi 3.

```
import
   com.pi4j.io.gpio.GpioController;
import com.pi4j.io.gpio.GpioFactory;
import
   com.pi4j.io.gpio.GpioPinDigitalOutpu
   t;
import com.pi4j.io.gpio.PinState;
import com.pi4j.io.gpio.RaspiPin;
public class LEDPi {
    public static void main(String[]
   args) throws InterruptedException {
                final GpioController
   gpio = GpioFactory.getInstance();
         final GpioPinDigitalOutput pin
   =
   gpio.provisionDigitalOutputPin(Raspi
   Pin.GPIO_01, "PinLED",
   PinState.HIGH);
         System.out.println("Blink the
   led every 100 ms second for 15
   seconds");
         pin.blink(100, 15000);
         System.out.println("Program
   will end in 15 s.");
         Thread.sleep(15000);
         // release the GPIO controller
   resources
         gpio.shutdown();
    }
}
```

The JDeveloper has to be configured in order to deploy the application as an executable jar. From the *Project Properties* we select *Deployment* and from there we can create a new deployment profile or edit an existing one. In this example the deployment profile is called *pi4j* and the jar file is called *pi4j.jar* as seen in Fig. 9. The *Main* class must be selected to have an executable jar file. In *File Groups* > *Project Output* the *Contributors* section must be selected and using the *Add* button the external jar files necessary to run de project must be added there (see Fig. 10). Then, we must check that in the *Library Dependencies* the necessary jar files are visible and selected. The final step in the project configuration is given in Figure 11 where we must add to *Libraries and Classpath* section the external jar libraries used in the code. From this moment, in the main menu of the JDeveloped IDE, at *Build*, under the *Deploy* section we have a new deployment profile called pi4j.

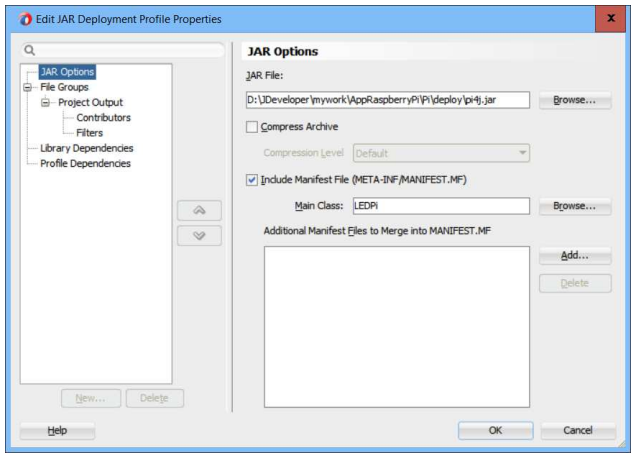

Fig. 9. – JDeveloper Options in deployment profile properties for executable jar creation on Raspberry Pi 3.

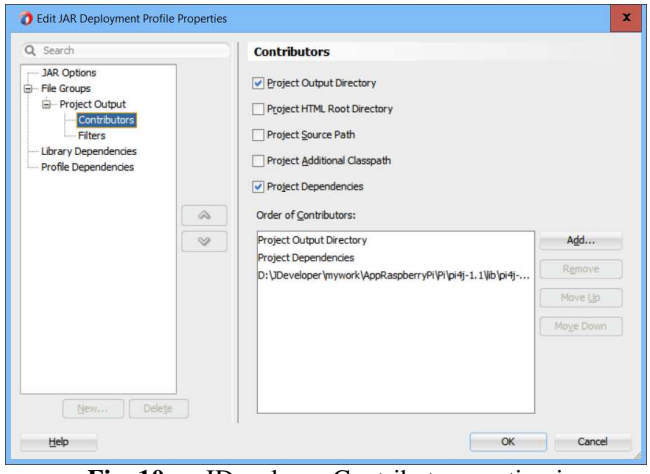

Fig. 10. – JDeveloper Contributors section in deployment profile properties for executable jar creation on Raspberry Pi 3.

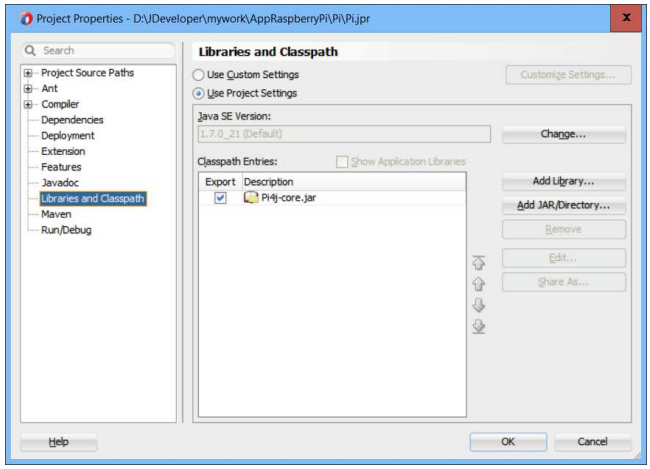

Fig. 11. – JDeveloper external jar files used in the creation of the executable jar on Raspberry Pi 3.

Building the deployment profile will output to the Log screen lines like:

[09:15:45 PM] ---- Deployment started. ----
[09:15:45 PM] Target platform is Standard Java EE.

```
[09:15:45 PM] Running dependency analysis...
[09:15:45 PM] Building...
[09:15:45 PM] Deploying profile...
[09:15:46 PM] Wrote Archive Module to
    D:\JDeveloper\mywork\AppRaspberryPi\Pi\deploy\pi
    4j.jar
[09:15:46 PM] Elapsed time for deployment:  1 second
[09:15:46 PM] ----  Deployment finished.  ----
```

Once the jar file is created under Windows it must be uploaded to the Raspberry Pi 3. To do that SSH must be activated on the Raspbian machine. Based on the IP, SSH user name and password I used SpeedCommander to transfer the file using a Secure FTP over SSH connection. The application is run as shown in Fig. 12 without any reference to the Pi4J library on the Raspbian.

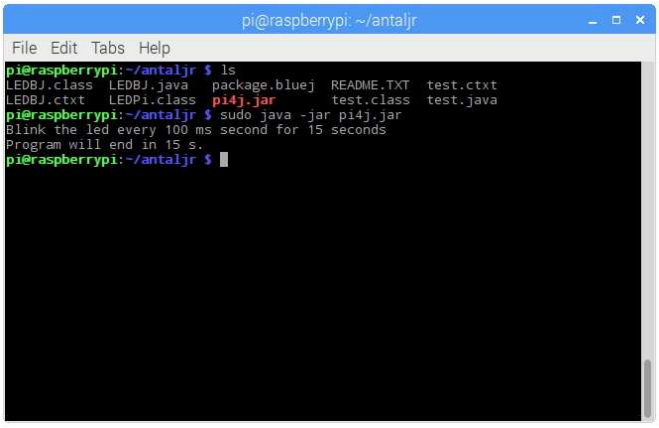

Fig. 12. – Running the jar file on Raspberry Pi 3.

REFERENCES

[1] ANTAL, T .A., *GUI's in JDeveloper*, Acta Technica Napocensis, Series: Applied Mathemathics and Mechanics, Nr. 52, Vol. IV, 2009, p.27-32, ISSN 1221-5872.

[2] ANTAL, T .A., *Programming AutoCAD using JAWIN from Java in JDeveloper*, Acta Technica Napocensis, Series: Applied Mathemathics and Mechanics, Nr. 53, Vol. III, 2010, p.481-486, ISSN 1221-5872.

[3] ANTAL, T. A., *Elemente de Java cu JDeveloper - îndrumător de laborator*, Editura UTPRES, 2013, p.150, ISBN: 978-973-662-827-6.

[4] ANTAL, T. A., *Java - Iniţiere - îndrumător de laborator*, Editura UTPRES, 2013, p. 246, ISBN: 978-973-662-832-0.

[5] Harshad Oak. Oracle JDeveloper 10g, APRESS, 2004.

[6] Avrom Roy-Faderman, Peter Koletzke, Paul Dorsey, *Oracle JDeveloper 10g Handbook*, McGraw-Hill/Osborne, 2004.

[7] Deitel, H. M., Deitel, P. J., *Java: How to program*, fifth editor, Prentice Hall, 2004, ISBN 0-13-120236-7.

[8] Herbert Schildt, *Java 2: The complete reference*, fifth edition, Osborne, 2001, ISBN 0-07-213084-9.

[9] Kathy Siera, Bert Bates, *Sun Certified Programmer for Java 6 Study Guide*, McGraw Hill, 2008, ISBN 978--0-07-159108-9.

PROGRAMAREA PLĂCII RASPBERRY Pi 3, ÎN JAVA, SUB MEDIILE BLUEJ ŞI JEDEVELOPER UTILIZÂND BIBLIOTECA Pi4J

Rezumat: Lucrarea îşi propune să dea o descriere clară şi detaliată a configurării sistemului de operare Raspbian rulat pe placa Rapberry Pi 3 pentru programarea în Java folosind mediile BuleJ şi JDeveloper utilizând biblioteca de intrare/ieşire Pi4J.

Tiberiu Alexandru ANTAL, Professor, dr. eng., Technical University of Cluj-Napoca, Department of Mechanical System Engineering, antaljr@bavaria.utcluj.ro, 0264-401667, B-dul Muncii, Nr. 103-105, Cluj-Napoca, ROMANIA.