TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

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

# ARDUINO LEONARDO PROGRAMMING UNDER WINDOWS, IN JAVA, FROM JDEVELOPER USING ARDULINK

**Tiberiu Alexandru ANTAL**

***Abstract***: *The paper gives a step by step description for configuring Windows and JDeveloper IDE in order to interact with the Arduiono Leonardo microcontroller using the Java programming language based on the Ardulink Java Open Source project.*
***Key words:*** *Java, JDeveloper, Arduino, Microcontroller, Ardulink.*

## 1. INTRODUCTION

A microcontroller is a computer designed for embedded applications, which means that it will be part of some more complex device. We use it whenever limited computing functions are needed in order to control electronic devices attached to some processes. It contains the microprocessor, running at a low frequency, ROM, RAM and some I/O functions depending on the implemented architecture. Microcontroller programs are stored in the ROM, written is some C flavor and dedicated to perform a single task.

Microcontroller boards are printed circuit boards designed to facilitate the work with the microcontroller and include typical components as: power circuit, programming interface, basic input (usually buttons and LEDs) and I/O pins.

JDeveleoper is an Oracle IDE for Java programmers that can be used among others to write Java GUIs [1] or the program CAD applications [2] under certain limitations. The following paper describes the steps to program in Java, using the JDeveloper IDE, the Arduino Leonardo based on the Ardulink open source java solution for the control and coordination of Arduino boards (http://www.ardulink.org/). The following software must be downloaded and installed in order to succeed in Java programming of the board:

- libusb-win32-bin-1.2.6.0.zip from https://sourceforge.net/projects/libusb-win32/files/libusb-win32-releases/1.2.6.0/ in order to connect from Java over USB to the Arduino Leonardo board; the libusb is a C library that provides generic access to USB devices intended to be used by developers to facilitate the production of applications that communicate with USB hardware;
- the Arduino environment (IDE) from https://www.arduino.cc/en/Main/Software; the environment is necessary as the ArdulinkProtocol4LeonardoAndMicro.ino sketch must be uploaded to the Leonardo board in order achieve Java communication over USB;
- the ardulink java solution from http://www.ardulink.org/download/ in order to interact from Java with the board.

## 2. INSTALLING LIBUSB-WIN32-BIN-1.2.6.0.ZIP UNDE WINDOWS OS

The following figures are showing step-by-step what actions are needed in order to install the USB drivers for Arduino Leonardo board on Windows 10 of 64 bits. When unzipped the contents of the LIBUSB-WIN32-BIN-1.2.6.0 file is shown in Figure 1.

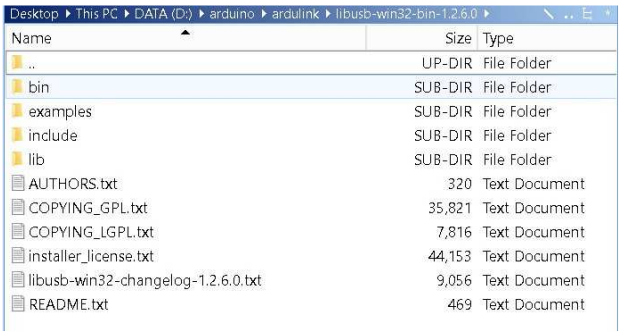


**Fig. 1.** – Contents of the LIBUSB-WIN32-BIN-1.2.6.0 zip file.

In order to start the installation from the *[bin]* directory (Fig. 1) we start the *inf-wizard.exe* application.

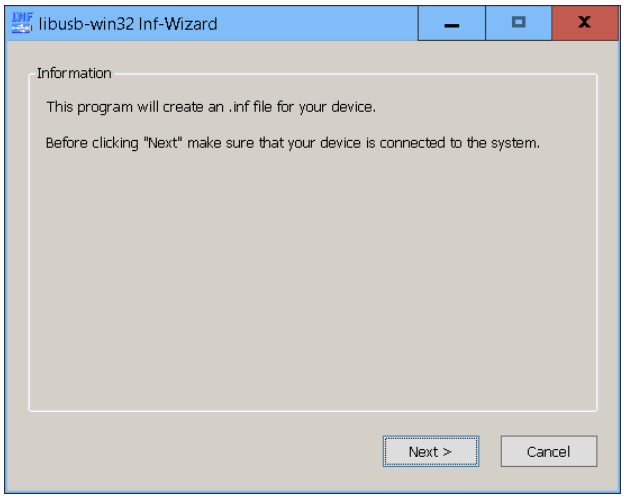


**Fig. 2.** – Start screen of the LIBUSB-WIN32-BIN-1.2.6.0 driver installation.

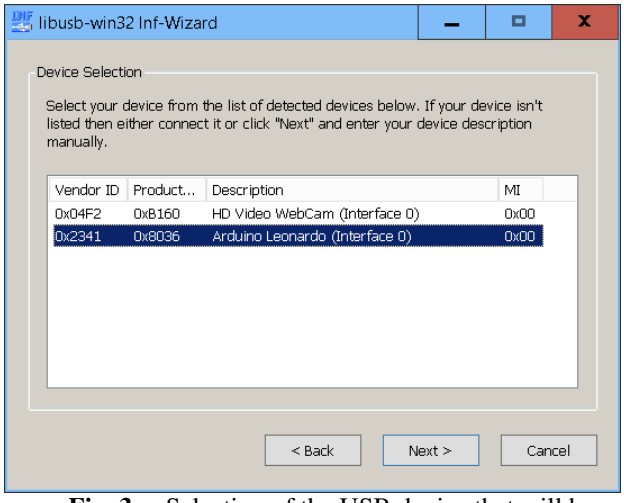


**Fig. 3.** – Selection of the USB device that will be used by the driver.

To get to the screen from Fig. 3 the Arduion Leonardo board must be connected to the computer over USB. Figure 4 is showing the configuration of the device

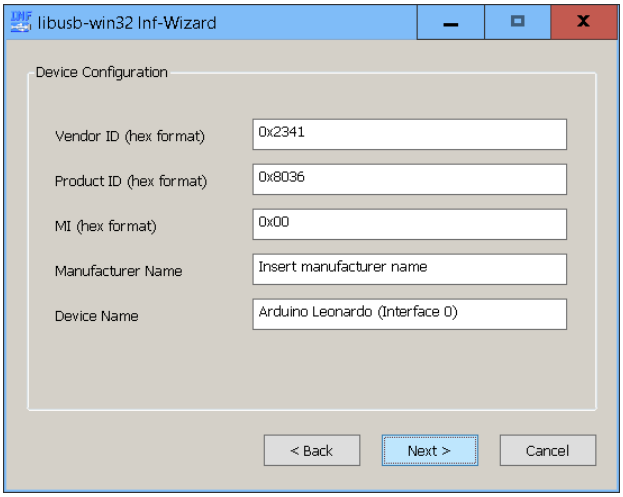


**Fig. 4.** – Arduino Leonardo device USB configuration.

After the *Next* button is clicked the Arduion_Leonardo_(Interface_0).inf file generated by the wizard must be saved on the disk as shown in Figure 5.

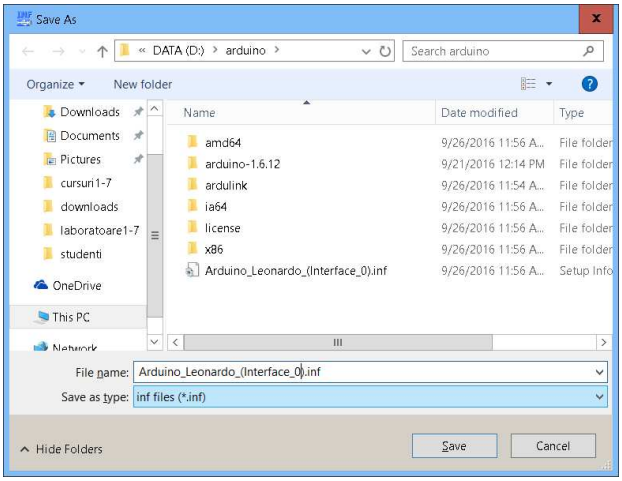


**Fig. 5.** – Selection of the place to store the Arduion_Leonardo_(Interface_0).inf.

The wizard can proceed with the installation of de device driver as shown in Fig. 6. Installation will succeed if Windows is configured to accept device drivers with no digital signature information. This means this security policy must be changed before running the wizard. If this has not been done then after clicking the *Install Now …* button from Fig. 6 Windows 10 will issue the error from Figure 7 .

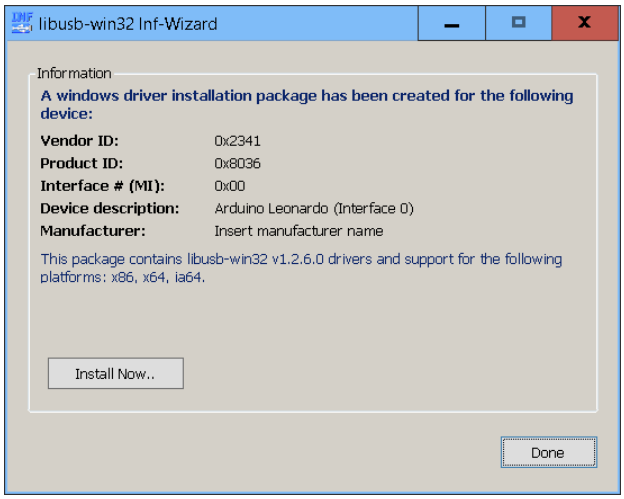


**Fig. 6.** – Selection of the place to store the Arduion_Leonardo_(Interface_0).inf.

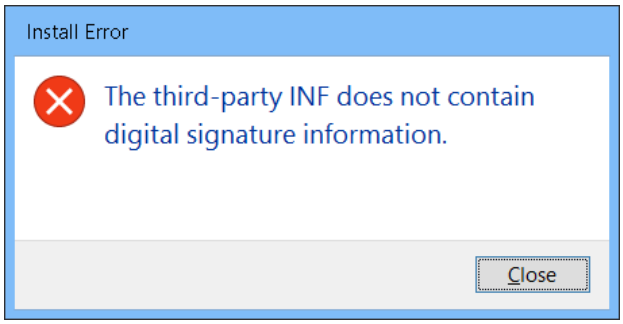


**Fig. 7.** – Windows 10 error message when trying to install a device driver with no digital signature.

If the error from above is displayed Windows 10 must be rebooted while holding down the <Shift> key. This is a way to install drivers without digital signature information. The Arduino_Leonardo_(Interface_0).inf file must be installed manually and the screen from Fig. 8 shows what to choose to successfully install the driver.

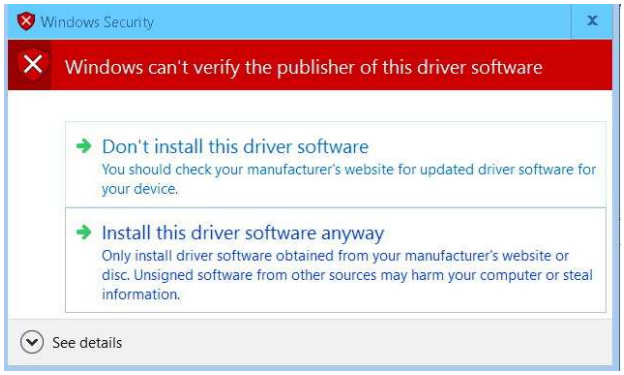


**Fig. 8.** – Selection screen for installing a device driver with no digital signature information under Windows 10.

This unsigned driver may not be shown in the Device Manager but be installed correctly to operate under Java over USB.

## 3. INSTALLING THE ARDIONO IDE

The steps necessary to work with the IDE are well described on the Arduino web site. Fig. 10 is only showing how a correctly configured IDE should look for the Lenardo board and that the ArdulinkProtocol4LeonardoAndMicro.ino sketch must be compiled and loaded to the board in order to proceed with the Java interaction over the USB port. The sketch is necessary because the Arduino Leonardo lacks the serial connection feature on USB so the code will 'supply' the missing feature to the board.

## 4. INSTALLING THE ARDULINK JAVA OPEN SOURCE SOLUTION

The contents of the unzipped ardulink-V0.6.1-20151223-2236.zip file is shown in Fig. 9. ArdulinkProtocol4LeonardoAndMicro.ino sketch from the *[sketches]* directory must be loaded to the board in order to use Java communication over the USB port.

| Name | Size |
|---|---|
| .. | UP-DIR |
| [3rds] | SUB-DIR |
| [bin] | SUB-DIR |
| [conf] | SUB-DIR |
| [examples] | SUB-DIR |
| [lib] | SUB-DIR |
| [sketches] | SUB-DIR |
| [winDLLs] | SUB-DIR |
| Features.txt | 5,257 |
| Notes.pdf | 183,731 |
| Notes.odt | 98,849 |
| License.txt | 604 |

**Fig. 9.** – Contents of the ardulink-V0.6.1-20151223-2236.zip file.

## 5. CONFIGURING JDEVELOPER TO RUN THE JAVA CODE BASED ON ARDULINK

The *[lib]* directory (Fig. 9) must be added to the *Project Properties …* in JDeveloper [3]- [9] with the help of the dialog from Fig. 11. Also,

the contents of the *[winDLLs]* directory (Fig. 9) must be copied to the project directory of the Java application (the one with the *src* and *classes* directories and holding the *.jpr* file [3], [4] – see Fig. 12) depending on the 32 or 64 bit Windows used on the machine.

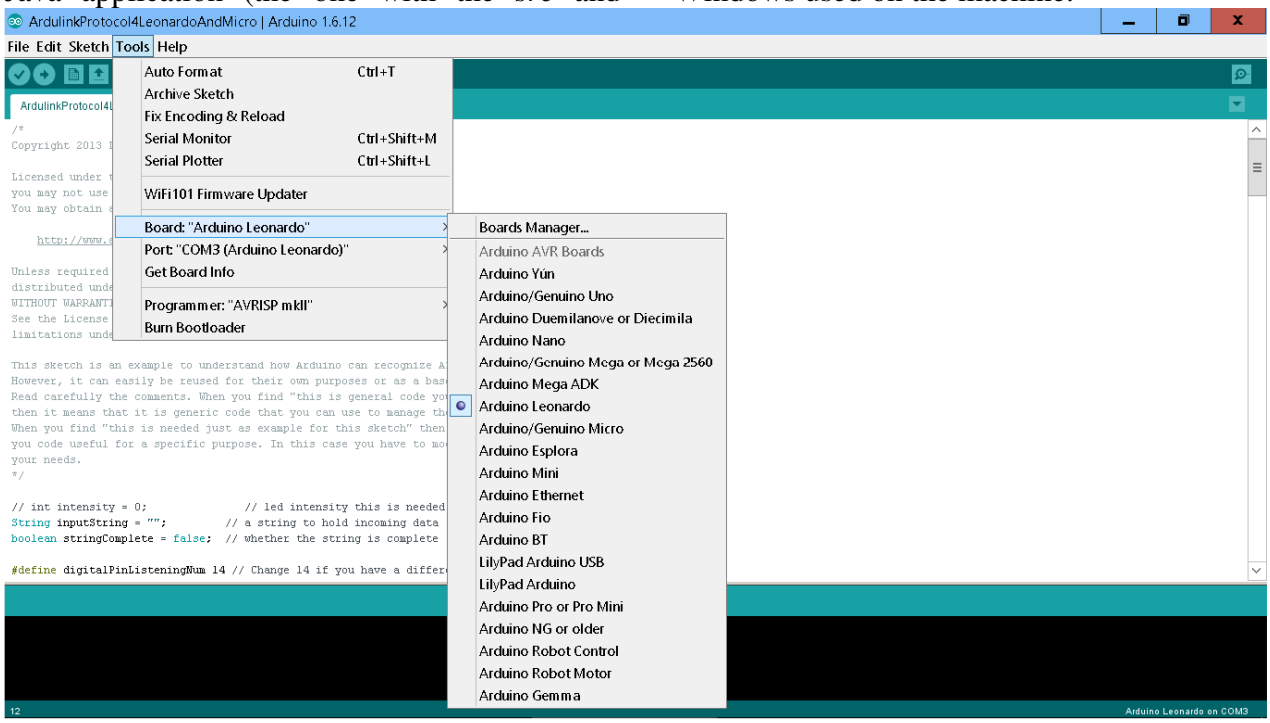


**Fig. 10.** – Arduion IDE used to connect to the de Leonardo board.

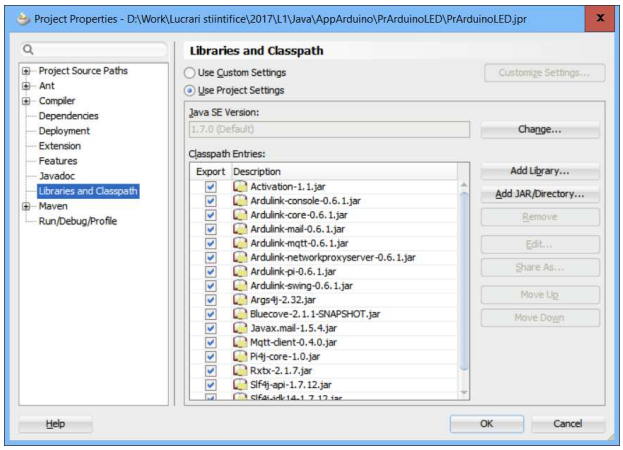


**Fig. 11.** – Project Properties, Libraries and Classpath tab in JDeveloper.

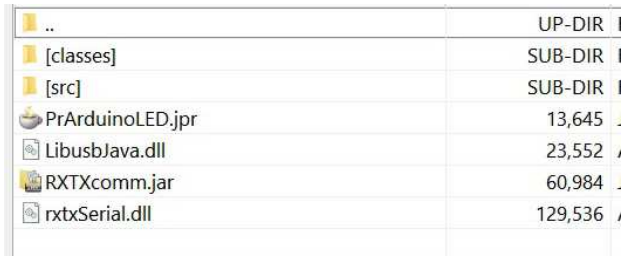


**Fig. 12.** – Content of the *[winDLLs]* copied to the JAVA project directory.

The original code for LED blinking can be found at http://www.ardulink.org/how-to-blink-a-led/. The following version is modified to run only on Arduino Leonardo USB connection using ALProtocol. The `Link.getDefaultInstance()` gets the default `Link` class instance that is using the ALProtocol. Each link can define a specific protocol or use the protocol Ardulink called ALProtocol. ALProtocol (Ardulink Protocol) is a legacy protocol that refer to an interface, called IProtocol, that ALProtocol implements. If the communication over USB is not working properly some investigations can be made with the help of the Ardulink Console desktop application that can be run by typing in the *Command Prompt Window* `java -jar ardulink-console-0.6.1.jar`. If all installations went well the *Log Window* from JDeveloper should look like the following log ending with *connection on COM3 established Connected: true*.

*Native lib Version = RXTX-2.2-20081207 Cloudhopper Build rxtx.cloudhopper.net*
*Java lib Version = RXTX-2.1-7*
*WARNING: RXTX Version mismatch*
*Jar version = RXTX-2.1-7*

*native lib Version = RXTX-2.2-20081207 Cloudhopper Build rxtx.cloudhopper.net*
*Nov 30, 2016 9:36:23 PM org.zu.ardulink.ConnectionContact writeLog*
*INFO: found the following ports:*
*Nov 30, 2016 9:36:23 PM org.zu.ardulink.ConnectionContact writeLog*
*INFO: COM3*
*Connecting on port: COM3*
*Nov 30, 2016 9:36:23 PM org.zu.ardulink.ConnectionContact writeLog*
*INFO: connection on COM3 established*
*Connected:true*
*Send power:1*
*Send power:0*
*…*
*Send power:0*

until the Java applications is stopped. Depending on the stop moment the LED might stay on of off. In order to understand how the following code was build the following documentation link must be consulted http://www.ardulink.org/javadoc/Arduino-1/ardulink-core/index.html

If the board works with the Java code then it should write to the *Log Window* lines like:

```
import java.util.List;
import org.zu.ardulink.Link;
import org.zu.ardulink.protocol.IProtocol;

public class BlinkLED {
    public static void main(String[] args) {
        try {
            Link link = Link.getDefaultInstance();
            List<String> portList = link.getPortList();
            if (portList != null && portList.size() > 0) {
                String port = portList.get(0);
                System.out.println("Connecting on port: " + port);
                boolean connected = link.connect(port);
                System.out.println("Connected:" + connected);
                Thread.sleep(1000);
                int power = IProtocol.HIGH;
                while (true) {
                    System.out.println("Send power:" + power);
                    link.sendPowerPinSwitch(13, power);
                    if (power == IProtocol.HIGH) {
                        power = IProtocol.LOW;
                        Thread.sleep(500);
                    } else {
                        power = IProtocol.HIGH;
                        Thread.sleep(500);
                    }
                    //Thread.sleep(500);
                }
            } else {
                System.out.println("No port found!");
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
```

The `Link` class represents the serial connection between the computer and the Arduino board over the USB. It must be used by any java class to communicate with Arduino. The steps necessary to establish the connection using the class and the methods for that are given at http://www.ardulink.org/how-to-blink-a-led/. The `sendPowerPinSwitch()` method request Arduino to perform a `digitalWrite()` function call to turn ON or OFF the LED. The first argument is the pin number and the second o constant with the `IProtocol.HIGH` or `IProtocol.LOW` values. The second class used to interact with the LED from the board is called `IProtocol`. This is an interface that defines all the messages that can be sent to the Arduino and provides a method to analyze all messages from Arduino. The Java code is based on an infinite while loop. Inside the loop the `Thread.sleep(long millis)` method causes the currently executing thread to sleep for the specified number of milliseconds. For each sleep period the LED is set ON by assigning to the variable `power` the `IProtocol.HIGH` value or OFF by assigning to the variable `power` the `IProtocol.LOW` value. So for each time the body of the loop is executed the LED will stay a while ON and then OFF. The process is repeated until the application is terminated.

## 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 AURDUINO LEONARDO SUB WINDOWS, ÎN JAVA, UTILIZÂND MEDIUL JEDEVELOPER ŞI BIBLIOTECA ARDULINK**


**Rezumat:** Lucrarea îşi propune să dea o descriere clară şi detaliată a modului în care se poate conecta mediul de programare JDeveloper cu placa Arduino Leonardo în vederea utilizării limbajului de programare Java pentru a interacţiona cu placa prin intermediul bibliotecii Ardulink.



**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.