User Manual

Stable Version 1.1

Introduction

Embedding Arduino components along with Java programs is a procedure which can unfold situations in which it is necessary to establish the serial communication between these different channels. In order to achieve this task both hardware and software are supposed to share a common communication protocol, which can be defined as a rule for sending blocks of data from a computer system to another. In this context, there are some libraries such as RxTx and IO (both for Java) which are develop to fill up this gap by employing a serial port to deal with one-sided messages. However, those libraries only provide errors' identification for one side of the platform, leaving the other side to the programmer. In addition they can also be considered a limited resource in the sense that they can easily either lost data or suffer physical interferences.

What is Javino and how does it work?

Javino can be understood as a double-sided library for exchanging messages between an Arduino and a Java program by using a serial port. It is composed by the Javino for Arduino and the Javino for Java, which work together in order to increase the level of correctness in this operating context. Based on this communication protocol the information sent is compound by three fields: *preamble*, *field size* and *message content* (figure 1). The first one is formed by four hexadecimal characters which are used to identify the beginning of the message. The second one is structured by two hexadecimal characters that are employed to calculate the data extension. Finally, the last one is structured by the written content, which can be up to 255 bytes. During this process, both *preamble* and *field size* are used together in order to avoid loss of information and Javino, for the sake of practice, automatically mounts the final message.

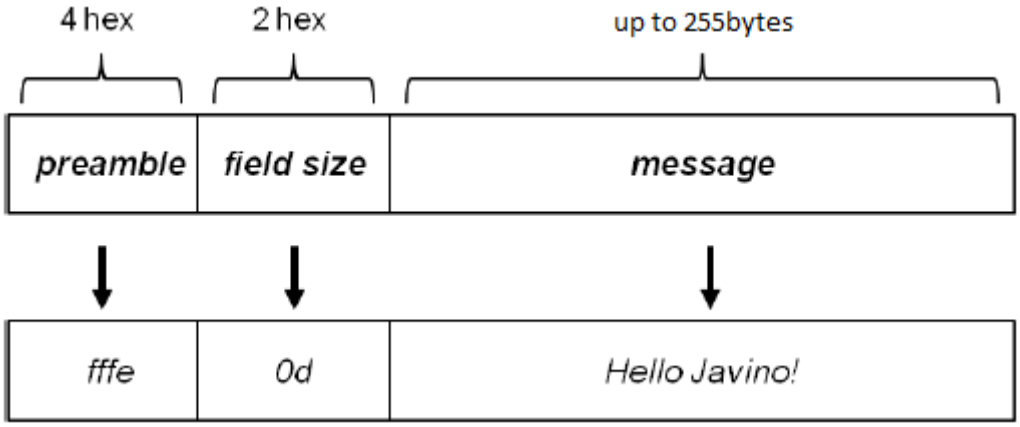

Figure 1: The message format

When a message is sent the Java library starts listening to the serial port (awaiting char by char messages) and if there is any information arriving it stores the character, analyzing if it is part of the expected *preamble*. Thus, this process is repeated until the message has been completely received and Javino discards all information while a valid *preamble* is not confirmed. Otherwise, it verifies the *field size* value in order to identify the message length. Because of this, it is possible to either avoid computational errors or define where a message starts and where it ends. Moreover, this double-sided library also has a default time-out to every character received: it ends the current process if the time-out has already been reached and starts a new message listening. After finishing this entire process, the string data is mounted and returned to either Arduino or Java. In the first case, a hardware function can be activated or sensors can be asked about information. Differently, in the second case, the data can be simply printed in the software console.

Java is a strictly programming language which is compiled to machine code for a simulated CPU before being emulated on the current operating system. On the other hand, Python is known for being an object-oriented programming language. In the context of Javino functioning, Python and pySerial are used to establish a lower-level communication for the serial port because it is also a purely interpreted system.

How to program the Java side by using Javino?

Javino can be programmed in different ways because the projects developed can vary from one to another and in order to start programming the serial communication it is necessary to create a new Javino instance. So this double-sided library can be used by instancing two sorts of methods based on a *Constructor Summary* (table 1), which is a set of programmable structures that create new objects in Java. Differently, the *Method Summary* (table 2) only presents syntaxes which guide how the messages exchanged are supposed to be employed.

I. *pathBash*: A string attribute, standardized at different values according to the operating system employed, which sets the path of Python and can be understood as an environment's component which Javino depends on. Two examples which expose the use of this programmable tool can be seen below:

- The Bash path in an edition of the Windows 64-bit:
  - `String pathBash= " C:\\Python27";`
- The Bash path in the Linux operating system:
  - `String pathBash = "/usr/bin";`

Constructor Summary	
`Javino j = new Javino ();`	Instancing the Javino by considering that Python is already installed on the computer and by employing the default Bash path related to the operating system used.
`Javino j = new Javino (pathBash);`	Instancing the attributes called *port* and *pathBash* in Javino.

Table 1: Javino's constructor summary

Method summary	
`j.sendCommand(port, msg);`	Sending a string message, which can be up to 256 characters, to Arduino. This method returns a boolean value which gives a feedback about the communication with the microcontroller. In some cases a command cannot be received by Arduino because the serial port is locked, which means that it is currently dealing with another sort of data.
`j.requestData(port, msg);`	Returning a boolean value which aims at checking if there is any answer to be sent by Arduino. The user is supposed to send a request in order to inform Arduino that information was returned to Java.
`j.listenArduino(port);`	Returning a boolean value which aims at checking if there is any answer to be sent by Arduino.
`j.getData();`	Returning the message, which can be up to 256 characters, from Arduino.

Table 2: Java's method summary

How to download both Javino for Java and Python?

Javino requires both Python 2.7 and pySerial, which enables access for Python running on any operating system but Linux, which can installed those dependencies by using the following command: *# apt-get update && apt-get install python python-serial*. Python 2.7 can be found at https://www.python.org/ and the pySerial can be acquired at http://pyserial.sf.net/. In addition, Javino for Java can be downloaded at the link: http://javino.sf.net.

How to install Javino for Java?

The Javino for Java can be installed by following a sequence of steps which can be easily performed. Additionally, this process is based on the idea of configuring the Build Path, which is used for building an application and contains both source files and libraries which are useful to compile a Java project.

I. Configuring the Build Path:

- Click with the right button of the mouse over an icon called *JRE System Library [JavaSE-1.6]*.Then look for an operation named *Build Path* and choose the following option: *Configure Build Path*. This operation can be seen in figure 2.

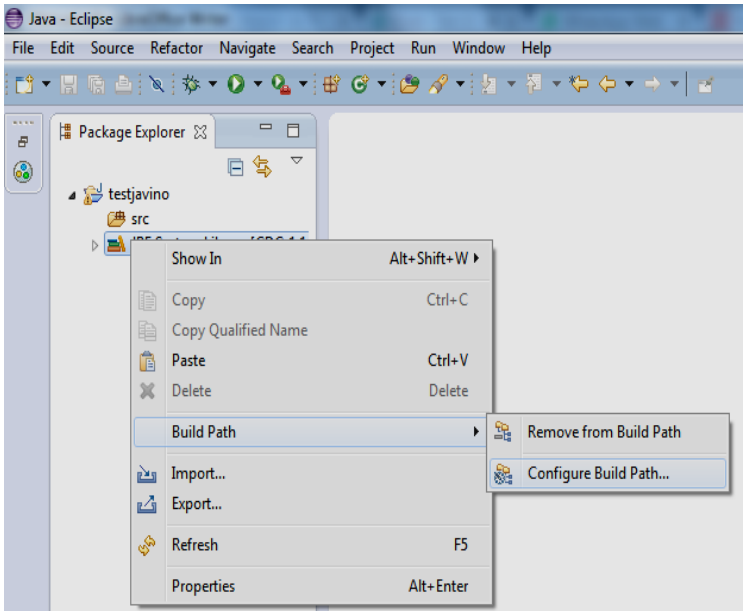

Figure 2: Configuring the Build Path

- Pick out the option called *Add External JARs* in the *Libraries* field and choose the javino.stable1.0 as the file which is supposed to be included in the set of Java libraries available. This operation can be seen in figure 3.

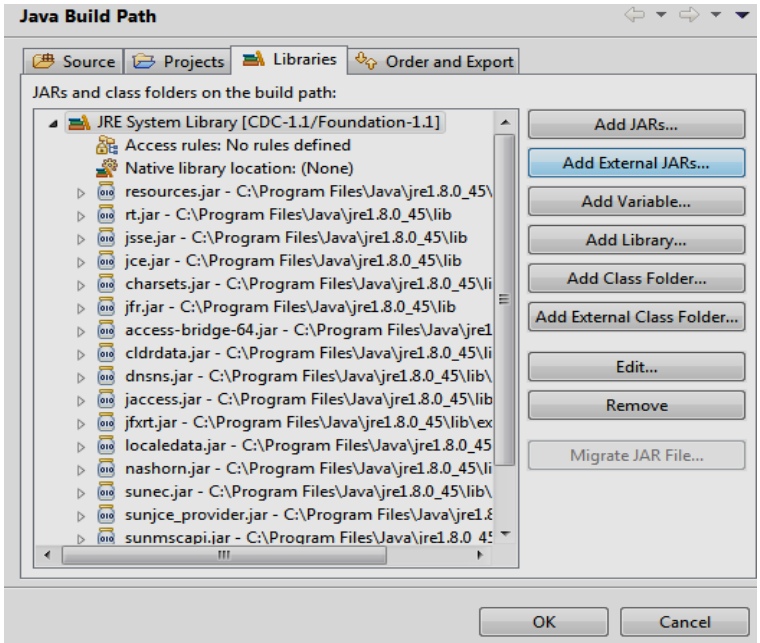

Figure 3: Adding External JARs

II. Importing a new library:

- After configuring the Build Path and importing the javino.jar file, it is possible to have this library referenced in the field named *Referenced Libraries*. This operation can be seen in figure 4.

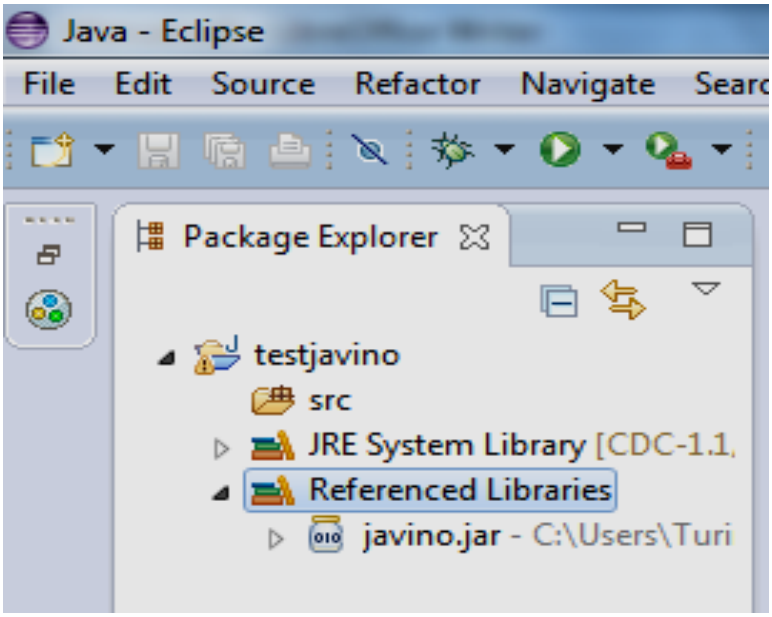

Figure 4: After importing the javino.jar library

Is there a practical example available?

I. A Java source code which presents a practical example of serial communication by using Javino:

```
import br.pro.turing.javino.*;

public class Main {
    public static void main(String args[]) {
            Javino j = new Javino();
            String port = "COM3";

            String ask = "hi";
            if (j.requestData(port, ask)) {
                    System.out.println(j.getData());
            }

            ask = "who";
            if (j.requestData(port, ask)) {
                    System.out.println(j.getData());
            }

            ask = "status";
            if (j.requestData(port, ask)) {
                    System.out.println(j.getData());
            }
    }
}
```

II. The Javino messages which are supposed to appear in the Java console:

```
[JAVINO] Using version stable 1.0 CEFET/RJ, Brazil
Hi
I'm Arduino!
I am alive!
```

How to program the Arduino side by using Javino?

The Javino for Arduino is also supposed to be programmed in order to establish a serial communication along with the Javino for Java. Firstly, it is necessary to import another library called *Javino.h* and instance it into an Arduino sketch (figure 5).Secondly, it is required to know a *Method Summary*, which also presents syntaxes that guide how the messages exchanged are supposed to be used in the communication cycle (table 3).

```
#include <Javino.h>
Javino j;

void setup(){
  Serial.begin(9600);
}
```

Method Summary	
`j.sendmsg(msg);`	Sending a string message, which can be up to 256 characters, to Java.
`j.availablemsg();`	Returning a boolean value which aims at checking if there is any answer to be sent by Java.
`j.getmsg();`	Returning the message, which can be up to 256 characters, from Java.

Table 3 : Arduino's method summary

How to download Javino for Arduino?

The Javino for Arduino library can be downloaded at the following link: http://javino.sf.net. In this case, there is no difference between employing this double-sided library in a computer in which the OS is either Windows or Linux.

How to install Javino for Arduino?

In order to install the Javino for Arduino library, *Javino.h*, in the Arduino IDE it is required to import the *Javino.zip* file as a new library, according to figure 6. After this procedure it is possible to start programming the serial communication by using the Arduino's functions along with the pre-defined Java objects.

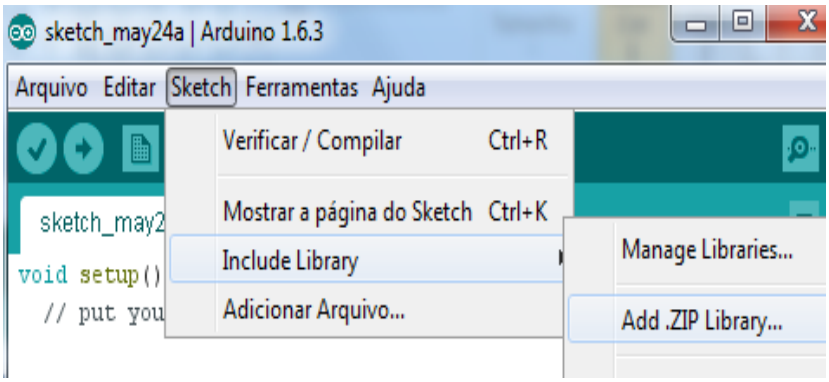

Figure 5: Importing the Arduino library

Is there a practical example available?

I. An Arduino source code which presents a practical example of serial communication by using Javino:

```
#include <Javino.h>

Javino j;

void setup() {Serial.begin(9600);}
void loop() {
  if(j.availablemsg()){answer(j.getmsg());}}
  void answer(String ask){

    if(ask=="status"){j.sendmsg("I am alive!");}
    else if (ask =="hi") { j.sendmsg("Hi");}
    else if (ask =="who"){j.sendmsg("I'm Arduino!");}
    else{ j.sendmsg("Sorry. I don`t understand you.");}
  }
```

II. Based on the serial communication established between the Java source code and the Arduino one, it is possible to visualize the following messages in the Java console:

```
[JAVINO] Using version stable 1.0 CEFET/RJ, Brazil
Hi
I'm Arduino!
I am alive!
```

III. For a better serial communication disable the autoreset on serial connection. See more: http://playground.arduino.cc/Main/DisablingAutoResetOnSerialConnection.

Conclusion

Javino is a double-sided library protocol for exchanging messages between Arduino and Java by using the serial port. It was developed by Lazarin, N.M and Pantoja, C.E on January 29 2015. Besides, more information about this project can be found by accessing either the authors or the official web site. In the first case, the developers can be contacted at either nilson.lazarin@cefet-rj.br or carlos.pantoja@cefet-rj.br. In the second one, the following link is also available for those who would like to know the Javino project: http://www.turing.pro.br/javino.

Who to cite this paper

LAZARIN, N M, e PANTOJA C E. 2015. “A Robotic-agent Platform For Embedding Software Agents using Raspberry Pi and Arduino Boards”. *Anais do IX Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações – IX WESAAC.*