Synchronous/Asynchronous Serial Port communication With JSSC

Asked 7 years, 7 months ago Modified 7 years, 5 months ago Viewed 2k times

0

I am trying to communicate between two computers using the Serial port and I am new to this area. I need to send requests from one computer(say A) to the other(say B) and receive responses for the requests sent. I am updating a Java Swing user interface with the responses.

I am using the jSSC library to do this. I have looked at the `SerialPortReader` examples and following is my understanding.

I will have to implement the `SerialPortEventListener` on both computers.A will use the `writeBytes` method to send the requests. B will listen to the commands sent using the `SerialPortEventListener` and will use the `writeBytes` method to send the response. A will listen to the data using its implemntation of `SerialPortEventListener` and when data is recieved, will update the UI. The following are my questions.

a) Is my above observation correct? Is there a different way to do this(for example is it possible that a `writeBytes` method that will return the response exists within the protocol?)

b) I read in several paces that serial port communication can be either synchronous or asynchronous. But from the examples, I can't understand if that code has implemented an asynchronous or synchronous communication. How would one go about implementing synchronous/asynchronous communication using jSSC? I am not asking for an implementation. Just some guidelines and what methods can be used.

c) It's possible that messages will be partly delivered. For example, if I send the command as a String "get variableThreeValue", it is possible that only the "get" or something like "get varia" will be received.(this can result in messages like "get get" etc..) How can I handle this kind of scenario? Again, I am not asking for an implementation. Just some guidelines and what methods can be used.

java serial-port jssc

Share Improve this question Follow

asked Feb 11, 2015 at 4:59

Can't Tell
12.1k 9 59 87

You don't want synchronous. Serial connections are typically asynchronous: start and stop bits are handled by the HW. If you don't trust the connection, you'll have to design a protocol: message format (length

@laune I have to support both wi-fi and Serail port communication. Radio transmitter/receiver will be plugged into the serial port and communication will happen wirelessly. Thanks for the advice. But I asked the questions to improve my understanding as well. Can you help me with the above questions as well? – Can't Tell Feb 11, 2015 at 5:27

Checking and writing an answer. – laune Feb 11, 2015 at 6:07

2 Answers

Sorted by:
Trending sort available
Highest score (default)

1

A classic "serial port" is something very "low level". Parameters like baud rate, start and stop bits and flow control must be set, and then byte sequences are read and written. The Java library uses a listener to receive events which are directly derived from what the serial driver senses (you find terms like "line" in the javadoc). How to react, depends on "the other side".

Flow control is what you use to avoid overrunning or overfilling the receiver. The lines of an RS-232 contain C(lear)T(o)S(end) and R(equest)TS, so "hardware handshake" is one option. Alternatively, US-ASCII defined control characters XON and XOFF which could be embedded in the data stream *if it is not binary data*. Flow control should not be an issue if the sending end doesn't send at full blast or computers differ significantly in speed.

As I understand, you'll connect some radio device to the port, and its documentation should specify all the parameters, and also the higher level protocol, i.e., how to transmit and receive data. The device may have special requirements, e.g., that you must pass on some setup data before you transmit actual data. (If you connect two computers with just a cable, then *everything* is up to you.)

Once you can basically send and receive, you'll have to think about safe transmission. You need to design a protocol:

- message format, e.g. containing a length, a sequence number, data bytes, a CRC.
- message sequencing, i.e., who may send what and when, e.g., a message A to B, an acknowledge from B to A, repeat. Or a NAK from B, and A must resend.
- Perhaps you need a "session protocol", i.e., a login (as in ftp) and a logout
- Timeout: what if either side doesn't receive another message within T?
- Do you need a heartbeat, i.e., a message sent when the channel is idle to learn that the other side is still "alive".

A WLAN connection should make most of this unnecessary. Real "radio" (some short wave?) I've never heard of, but I'm not a radio expert.

laune
30.8k 3 27 41

The device is learn.sparkfun.com/tutorials/exploring-xbees-and-xctu, just for your infromation – Can't Tell Feb 11, 2015 at 8:33

Thanks! Nice toy. - I've seen "flow control". Do you know what XON/XOFF vs. "hardware" means? – laune Feb 11, 2015 at 8:48

0

XON/XOFF is software flow control. Assume bi directional serial bytes between producer a and b. If either sends a XON byte it means whoa, stop sending me bytes until I send you a XOFF byte.

For hardware replace XON byte with CTS and XOFF byte with RTS.

Share Improve this answer Follow

answered Apr 22, 2015 at 19:45

Clyde W. Phillips Jr.
9 3