Wireless Radio Communication with RCX

Pavel Petrovic, ppetrovic@acm.org, Richard Balogh[1], balogh@elf.stuba.sk
Department of Computer and Information Science
Norwegian University of Science and Technology (IDI)
Trondheim, Norway
IDI Technical report 1/2006.

Introduction

We use the low-cost hardware LEGO Mindstorms RCX robotics platform for simple prototyping in research and student projects. The communication capabilities of the RCX brick are limited to the IR-communication, which requires direct or at least reflected visibility, it is limited to short distances, and it has limited reliability mainly due to the irregularities in IR-light reflections. In addition, multiple modules communicating at the same time result in collisions, and the communication is only half-duplex (it is bi-directional, but sending in one direction at a time).

In order to overcome at least some of these limitations, we were investigating options for wireless radio-based communication. There are four basic options for setting up such a radio communication link known to us:

- Analog radio: using one of the variety of analog transmission systems, while building a specialized circuitry to translate the digital signals to analog and vice versa. This type of transmission (in addition to need for extra AD/DA circuitry for translating the signal) suffers from noise. The user would have to implement handshaking and error-correcting protocols to achieve a reliable communication.
- Radio modem: using one of the available radio modems that translate the serial port communication to radio signal and back. An example of such a module would be Parallax 433 MHz RF Transceiver Package, which is a high-cost solution requiring two separate modules on both sides, and allows connecting for only a single pair of communicating devices. In addition, it is vulnerable to the interference with other devices that use the same frequencies.
- Wifi: wireless Ethernet, (IEEE 802.11*) devices connect either peer-to-peer, or using a shared access point. An example of a robotic platform capable of WIFI communication was the SONY Aibo robotic platform. The advantages are the standardized and well-supported communication protocol, possibility for multiple units coexisting in the same space, while all the communication conflicts are handled by the lower levels of the communication protocols, transparent to the user. Another advantage is the possibility to connect the local WIFI network to the Internet, and thus expose the robots directly on the public network. Finally, WIFI networks have high throughput and can be used even in such applications as video transmission.
- Finally, Bluetooth communication protocol provides somewhat lower-cost alternative while keeping most of the advantages of WIFI: standardized protocol with automatic prevention of collisions and error correction, multiple communicating peers, secure authentication if needed. The communication speed is typically lower, but there is an advantage of performing as a direct serial communication link once a connection between the two communicating peers is established, and virtual serial port is opened.

[1] Department of Automation and Control, Slovak Technical University Bratislava

Bluetooth platform is typical for many embedded devices, which can perform in slave-mode and can be connected by master (for example a PC). In that way, the PC can take the role of a hub and connects to multiple hosts at the same time, possibly coordinating the communication. An example of a Bluetooth modem device designed for embedded devices is the SparkFun BlueSmiRF module, which will be discussed in the sections below.

The following sections give a very brief introduction to RCX platform, and describe two systems we implemented and used in projects at the Department of Computer and Information Science, NTNU that involved radio communication with the RCX.

Communication using Wireless camera

The student experiment consisted of an autonomous rescue mobile robot moving in an arena. Its task was to discover victims as detected by a color camera mounted on the robot. The camera is a network-type, i.e. it is running an on-board web server, and makes the video available to be viewed in a web browser, where it is shown by a proprietary ActiveX plug-in. The frame-rate of the update is up to 30 frames per second; however, the only way to obtain the frames, according to the producer, is to grab the frames from the web browser screen. We have therefore developed a C++ application (and indeed also Java API using JNI), which initializes the communication (i.e. starts the web-browser automatically), and grabs the images from the browser window (even if it is running in the background). This is achieved using the PrintWindow() Windows API, which prints the contents of the window into a buffer provided., The grabbed image is processed further using the Intel's OpenCV image processing library. We detect the colored victims using our proprietary algorithm for simple color segmentation of the image. To consider an alternative solution: the camera software provides also an option of static image, which can be downloaded from the camera using ftp-protocol. Unfortunately, the update rate of that image is only 1 frame per second. The program that is utilizing the image grabbing from browser window with the source code is available from http://webcvs.robotika.sk/cgi-bin/cvsweb/robotika/detect/releases/detectv3.zip.

More importantly for this article, the camera is designed to be used as a security camera and thus it has a single-bit digital output and a single-bit digital input, for connecting the alarm siren, and motion detection sensor respectively. These bits can be triggered and determined by logging to the camera using telnet protocol and issuing service commands from the console. Connecting both of these contacts to a single RCX sensor port yields a setup, where the RCX can trigger the digital input of the camera by changing the sensor port configuration between active and passive sensor mode (i.e. whether it is powered or not), and triggering of the camera's digital output can be detected by reading the value from the RCX sensor port. In this way, we have connected the program running on the RCX to the same WIFI network where the network of the robot communicates. See figure 1.

There is, however, one displeasing limitation of this setup: the digital input and output are electrically separated from the wireless camera, and the switching on and off is performed using a slow relay, which cannot switch more than a very few times a second.

Figure 1. Robot built by Jean Paul Franky Friquin as part of his diploma project. The image is transmitted to the PC workstation, and the victims are detected. In case a victim is found a signal is sent to the RCX to stop, beep, and signalize the presence of the victim.

The data link layer of our communication protocol allows sending three types of packets:

1. *Nullits* - a single unary piece of information, or event trigger. A communicating peer can trigger a signal by sending a nullit. It is analogous to the concept of hardware interrupts of CPU.
2. *Digits* - a four-bit number (one hexadecimal digit) for sending a quick control information that can indicate one of the possible actions to take.
3. *Bytes* - a whole 8-bit number, full byte.
4. *Sequences of bytes* - a packet consisting of several bytes (sent as a sequence of byte-packets).

The physical layer of the protocol allows changing the state on the port between 0 and 1. However, keeping the signal high is not recommended for longer time, particularly because RCX sensor input detects the changes of the camera's digital output only while the signal is low. Therefore our communication protocol consists of sending full impulses (changing the line from low to high and to low back). Logical zero is sent as "no-impulse", while logical one is sent as "impulse", which must occur within a designated time interval. The nullit is sent as a single "1" - single impulse that is not immediately followed by another impulse. A digit is

sent as "11XXXX" where XXXX are the four binary bits of the digit, and byte is sent as "111XXXXXXXX", where XXXXXXXX are the bits of the byte (little-endian).

The Java API for both PC-side (utilizing the implementation of Telnet protocol from Apache Commons Net), and RCX-side (in LejOS) is available with source code from http://webcvs.robotika.sk/cgi-bin/cvsweb/robotika/lego/d_link_camera/releases/WifiComm_v2.zip.

Communication using BlueSmiRF module

A higher bandwidth can be achieved using a virtual serial port Bluetooth device - BlueSmiRF. The device comes in two versions: simple and advanced differing only in whether the handshaking bits RTS and CTS are available. For our purposes, the Tx and Rx pins conveying the serial protocol data signals on TTL level are sufficient. They could be connected to the RCX motor and sensor ports, but since RCX already has a serial port, it is more natural to utilize it (and save the precious input and output ports of RCX). However, the RCX's serial port is permanently connected to the IR transceiver device. As discovered by Dave Baum already in 1998, the RCX's [default] serial protocol for sending messages is operating at 2400 baud, with 1 start, 1 stop, and 1 parity bit, and 8 data bits. A '0' is coded as a 417us of 38kHz IR, while the bit '1' is 417us of no IR signal. We have decided to develop a circuit that transforms the data received from the BlueSmiRF into IR signal transmitted further to the RCX IR serial port. At the same time, the device should transform the IR signal emitted by the RCX's serial port into a TTL level input further into the BlueSmiRF module, which transmits it further over the Bluetooth radio protocol. The device is performing a very similar function as the standard serial IR tower, with the difference that the PC UART operates on the RS232 levels, whereas the BlueSmiRF required the TTL level signals. The schematic is shown on the figure 2. The left part is responsible for modulating the 38 kHz signal that represents the active data signal. The right part transforms the battery power to 5V required by the IR detector. The circuit has been first tested on a testing board (see figure 3), before we proceeded with a prototype product shown in figure 4.

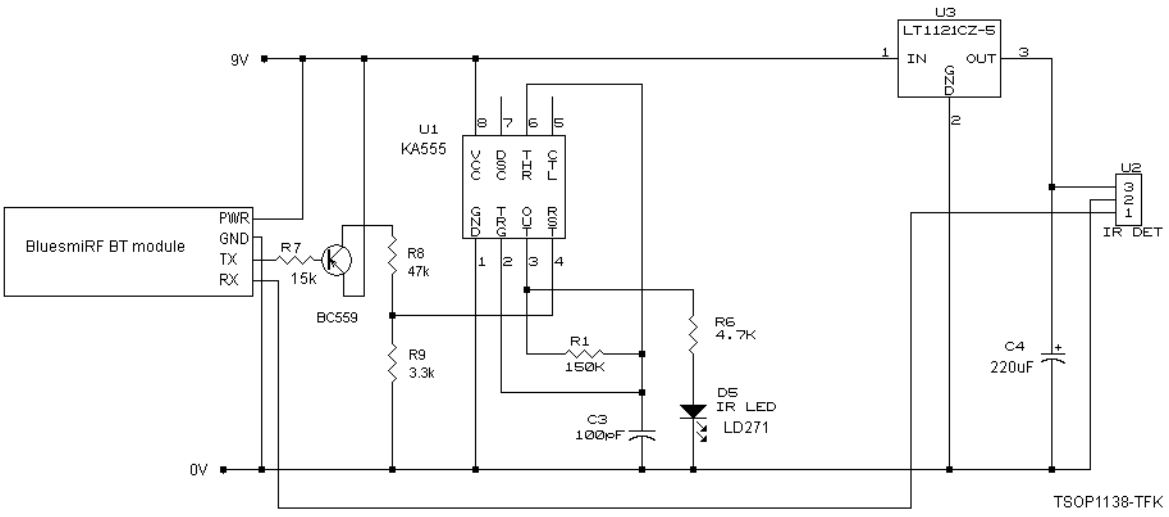

Figure 2. Schematic for the bidirectional IR to serial converter. Designed in cooperation with Ing. Richard Balogh, Faculty of Electrical Engineering, Slovak Technical University, Bratislava. The value of R1 should be tuned with the oscilloscope so that the frequency on pin 3 of U1 will be 38kHz (our value is 128kOhm). In our realization, we replaced the low drop-out voltage regulator LT1121CZ-5 with an equivalent LM2931AZ-5.0. According to the documentation, C4 needs to be at least 100uF.

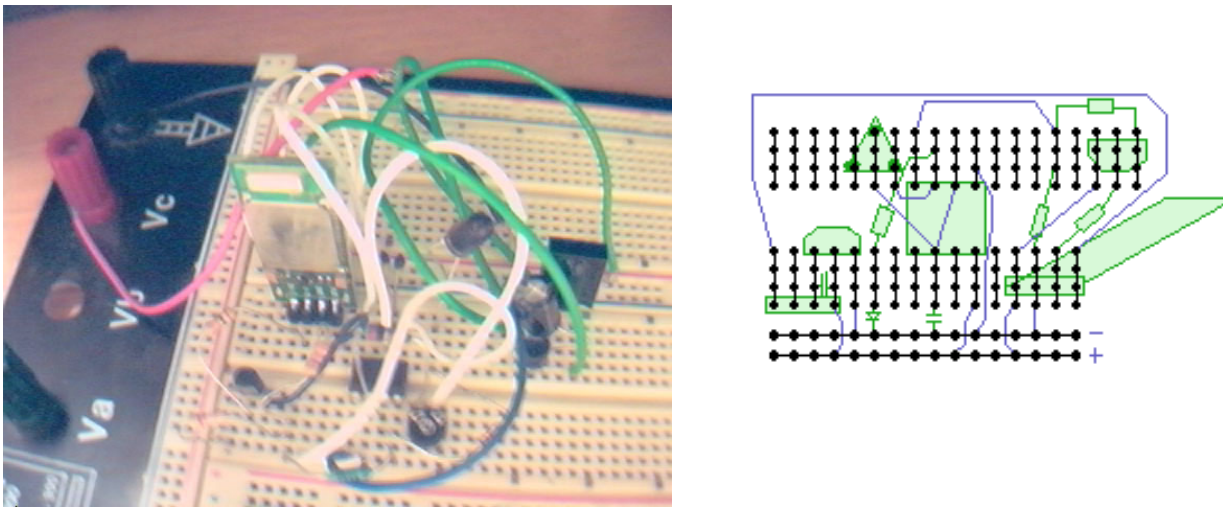

Figure 3. Experimental prototype on the testing board.

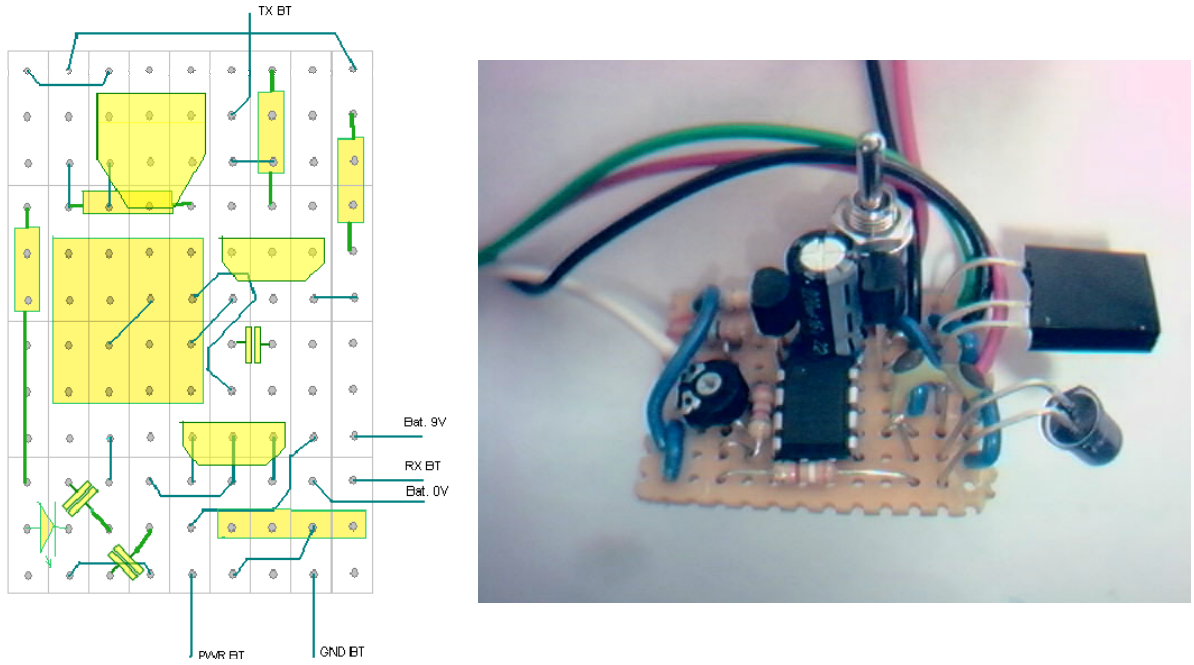

Figure 4. Serial to IR bi-directional converter. The board on the left is from the point of view of the parts. The switch is installed between the two pads, where the 9V battery cable connects. This design utilizes two parallel 47pF capacitors instead of the single 100pF shown on the schematic above.

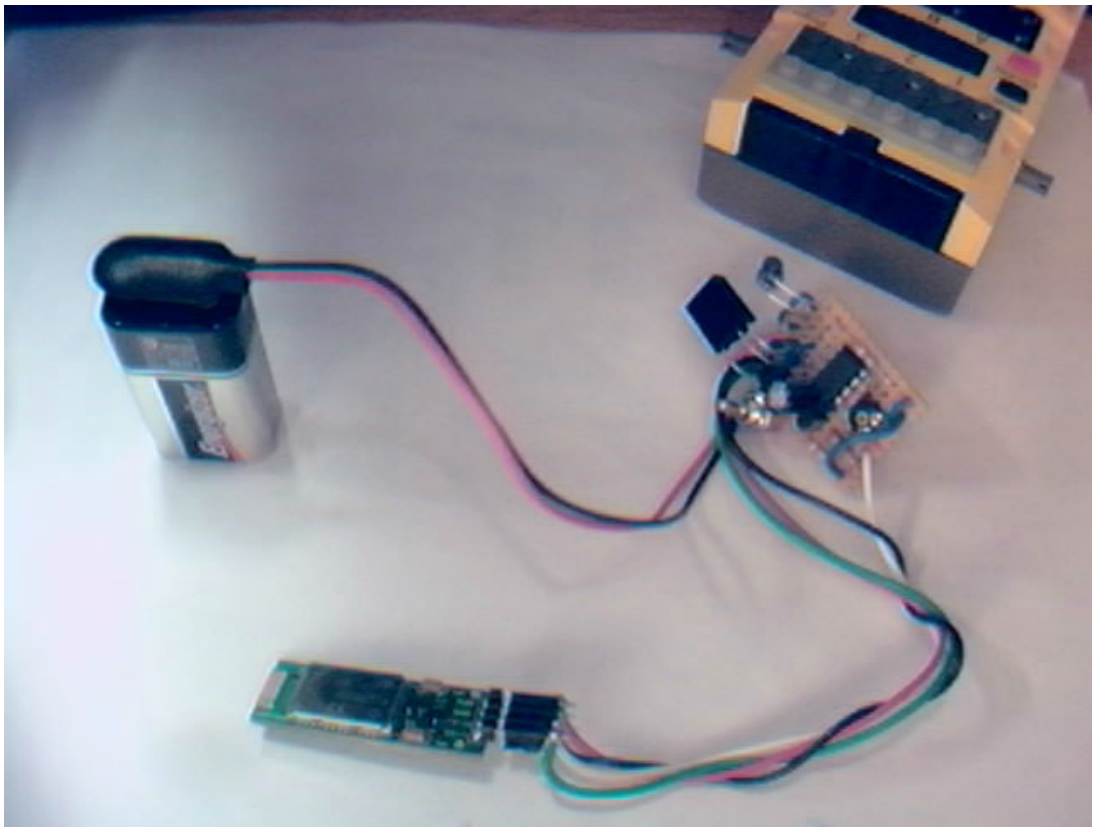

Figure 5. The device is powered from 9V battery. The BlueSmiRF module can be unplugged and used in other designs when needed. It can also be places comfortably at a separate location - such as on top of the robot for best RF transmission.

Configuration

Before the communication can begin, the BlueSmiRF module needs to be properly configured. Normally, the circuit comes configured as 9600-N-8-1, with no security, and various default settings. The configuration is performed when the module is in a command mode - that is either it is not yet connected, or the data mode is turned off when connected. Even though the manual claims the module can be switched from data to command mode by sending a particular sequence of escape characters, we found out that this sequence must be sent from the side of the embedded device: sending the escape sequence over the radio did not switch the module into command mode, unless already turned to command mode previously from the local link side.

Therefore, we have to have the capability to send the data to the Rx pin of the module from the side of the embedded device (local link). We could either connect it to a programmable microcontroller, or - we can connect it to the parallel port (LPT) of a PC. For this purpose, we designed a specialized utility that allows sending character strings using serial protocol over the pins of the parallel port. The utility (lptserial) is available from http://webcvs.robotika.sk/cgi-bin/cvsweb/robotika/bluesmirf/lptserial/.

The actual configuration protocol is described in the BlueRadios Bluetooth Intelligent Serial Module AT Command Set. In particular, the following configuration commands are recommended:

Change scan intervals thus decreasing the power consumption from 48mA to 2mA before the module is connected	ATSW21,2560,11,2560,11
Change the name of the module	ATSN,Usario
Change service name (virtual serial port service)	ATSSN,UsarioSerial
Prevent communication before connected - i.e. after this command, you can reconfigure BlueSmiRF only when it is connected, and you get out of the data mode using the escape sequence. This is important, because the randomly received and translated IR bytes will be ignored, and will not accidentally reconfigure the module.	ATSW25,0,1,1,0
Turn off verbal responses. This is important, because otherwise the BlueSmiRF will be talking to the RCX each time you open or close the port (CONNECT, NO CARRIER strings). After this command, the BlueSmiRF will output only data.	ATSW24,2,1,0,0
Set PIN	ATSP,newpin,default
Change communication parameters to RCX default: 2400-odd-1 and store them to flash.	ATSW20,10,1,0,1

A few notes:

- The lptserial utility is unidirectional: it doesn't show the response from BlueSmiRF. When connecting the parallel port pins to BlueSmiRF (use any pin D0-D7, and any of the ground signal pins (i.e. 25)), remove the BlueSmiRF from any other connections, except of the battery power.
- Be very careful when configuring BlueSmiRF. It is possible to reach a state when the module will be locked in a useless mode, which will be impossible to change.
- Some of the settings get accepted only after you turn off and on both BlueSmiRF AND the Bluetooth device on the PC side (such as the BT USB dongle).
- The configuration string has to be transmitted using the current communication parameters. It is therefore important to remember how you have configured the BlueSmiRF module. If the module doesn't respond, try first resetting all devices (including PC), and changing the battery. If the communication doesn't recover, our best advice is to try to connect the device from the PC (most of the BT software will detect and use the settings stored in BlueSmiRF automatically regardless of your manual setting on the side of PC). Then try to use an oscilloscope to measure the current operating bauds and bit length.
- The above settings are stored in the flash memory of BlueSmiRF. Once done with the configuration, you do not need to perform this configuration for the second time (regardless if the module is disconnected from the power).

Software

Once you locate the BlueSmiRF Bluetooth device, and successfully pair and connect it, you can start communicating with your RCX. We realized that some software has serious problem talking to the BlueSmiRF using our USB Bluetooth dongle from Trend Net, because the Windows driver automatically detects the connection settings. When the application attempts

to change the settings, the driver for the virtual serial port hangs. This behavior will probably differ at other system configurations, but we found out that simply opening the virtual serial port without attempting to change any of its settings works perfectly fine.

Our goal was to make the PC communicate with Lejos application running on the side of RCX, and since Lejos is particularly weak with its support for IR communication (it supports only the default ROM routines with a very limited buffer size), we first tested the device with the BrickOS system. A C++ API and an example application for communication with BrickOS are here: http://webcvs.robotika.sk/cgi-bin/cvsweb/robotika/lego/ir/bt2rcx/. The API is designed for simple use:

```
//---------------------------interface on the PC side
//                            (use standard LNP communication on RCX side)

//open virtual serial port
HANDLE fd = open_port("COM4");

//send lnp-message
lnp_send(fd, "Hello", 6);

//receive lnp-message if available
if (lnp_receive(fd)) printf("Message: %s\n", packet);

//close port
close_port(fd);
```

During our tests, the communication worked reliably even with messages of length 128 bytes. Once satisfied with testing, a Java API for communication with Lejos system has been developed as well, and it is available from: http://webcvs.robotika.sk/cgi-bin/cvsweb/robotika/lego/ir/bt2rcx/java/. Again, the API is designed to be simple to use and understand:

```
//---------------------the interface is symmetric between PC and RCX

//create communication object (and open port)
SimpleCommPC comm = new SimpleCommPC("COM4", (byte)0, false);

//send packet with 10 random bytes to RCX
byte[] packet = new byte[10];
for (i = 0; i < 10; j++) packet[i] = (byte)((int)(Math.random() * 256));
comm.send(packet, 10);

//receive packet from RCX, if available
byte[] answer = new byte[10];
len = comm.receive(answer);

//close port
comm.close();
```

Note that the implementation of the IR-communication in Lejos is poor. Therefore, the packets are sent (particularly in the direction PC->RCX) with extra overhead. In addition, it is unlikely to succeed with sending packets of longer length than 5 bytes (direction RCX->PC, the size is computed after doubling and encoding special characters – normally, the packet allows about 10 bytes) due to the limitation of RCX's internal buffer size. The current

implementation contains an error-correcting protocol with confirmation message. We introduced these features after the communication in the direction RCX->PC was not 100% reliable. For details, please see the source code, and its documenation. In summary - each message RCX->PC is confirmed by a short confirmation message. In addition, all bytes are sent twice (as N, N+1), with filtering out all possible situations that would prevent error recovery - it is thus sufficient that only 3 out of 4 consecutive bytes are received correctly, where the error can either consist of changing the value of single byte, or omitting a byte (this occurs often due to parity checking). The error detection is performed using two finite-state machines: separately for reading packet header, and packet data. The quality of line decreases with decreasing battery power level.

Conclusions

We have demonstrated implementations of two various extensions to the RCX programmable platform that allow the wireless radio communication of RCX with PC or other devices. The communication through the digital ports of D-Link wireless camera is very-low bandwidth and uses WIFI networking, whereas the solution using BlueSmiRF module works on the default serial communication bandwidth of RCX serial port, and provides reliable communication channel, which is not vulnerable to standard optical difficulties, sunlight, etc. inherent with IR-communication.

The LEGO company is planning to introduce their new robotics platform in the second half of this year. The new programmable LEGO brick will communicate using the Bluetooth protocol. This work not only is a precursor of the planned advancements, but even more - due to the generality of the BlueSmiRF module, it is a gateway for efficient communication between the current RCX platform and the NEXT platform, best of its kind known to us per today.

The work on the serial to IR converter can be further utilized for possibly controlling other devices from the PC, such as the WowWee robots, or home appliances.

References

- robotika.sk - more projects of IR-control with LEGO serial tower.
- hitechnic.com - plans to release NXT to RCX bridge, which is not wireless in several months.
- Infrared-radio-repeater for RCX using Conrad Electronics transceiver module TRX 433, http://www.restena.lu/convict/Jeunes/Mars_Mission_B/Mars_Mission_B_Main.htm.
- Parallax 433 MHz RF Transceiver Package, http://www.parallax.com/detail.asp?product_id=28180
- SparkFun BlueSmiRF module, http://www.sparkfun.com
- Lejos, http://lejos.sourceforge.net/
- BrickOS, http://brickos.sourceforge.net/
- BlueRadios Bluetooth Intelligent Serial Module AT Command Set, BlueRadios 2005, http://www.blueradios.com