

Connecting Lego Mindstorms NXT and RP6v2 Robot

Author: Georg Icking-Konert (icking@onlinehome.de)

version: 1.1

date: 2015-02-01

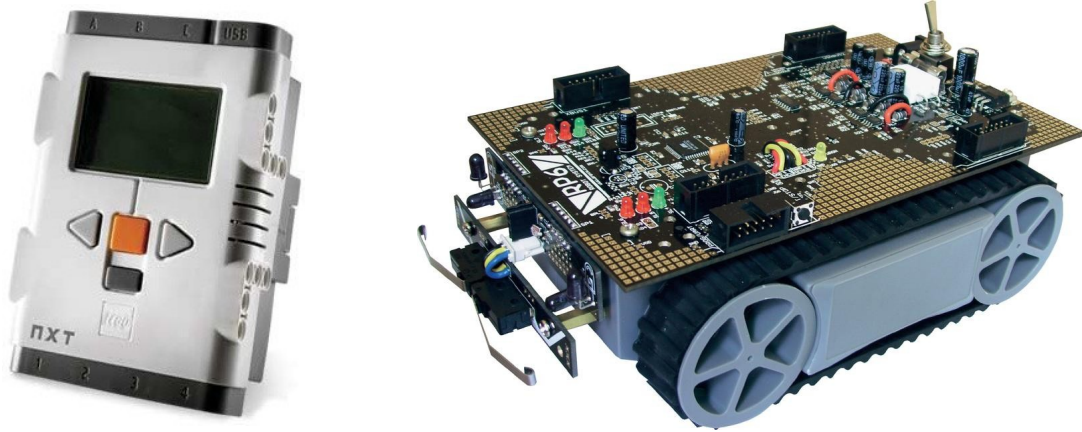


Table of Contents

Motivation.....	2
Requirements.....	2
Step 1: Connect NXT ↔ RP6v2.....	2
1a: Cable.....	2
1b: Pull-Up.....	4
Step 2: Program RP6v2.....	4
Step 3: Install NXT Blocks.....	5
Step 4: NXT Programming.....	6
Appendix: RP6v2 Commands.....	7
Read Registers.....	7
Send Command.....	7

Motivation

The famous Lego Mindstorms NXTⁱ brick controls its “smart” sensors, e.g. ultra-sound, via an I2C busⁱⁱ. After a small modification this bus can also be used for communicating with an Arexx RP6v2 robotⁱⁱⁱ (or other I2C gadgets). Below please find a collection of NXT blocks and corresponding RP6v2 software, which allow controlling a RP6v2 robot via a LEGO Mindstorm NXT.

In the future I hope to add more NXT+RP6v2 projects to the repository. For now I start with a simple crane. But the sky is the limit – let your imagination go wild!

Requirements

The following hardware and software are required for our NXT+RP6v2 creations:

- LEGO Mindstorms NTX (e.g. set 8527) with NXT-G programming language
- custom RP6v2 blocks “[NXT-RP6v2_Blocks.zip](#)” installed (see Step 3: Install NXT Blocks)
- Arexx RP6v2 robot with WinAVR toolchain (download from <http://www.arexx.com>). Robot programmed with I2C slave software (see Step 2: Program the RP6v2)
- these NXT-G blocks from <http://www.teamhassenplug.org/NXT/NXTGAdditions.html>. Import via “Block Import and Export Wizard...” from the NXT-G “Tools” menu.
 - “Bit Logic“
 - “Display Number“
 - “IIC Read“
 - “IIC Write“
 - “IIC Read – multi“
- I2C hardware modification (see 1b: Pull-Up)
- I2C cable to connect NXT to RP6v2 (see 1a: Cable)

Step 1: Connect NXT ↔ RP6v2

1a: Cable

The cable for NXT has a proprietary RJ12 plug, while the RP6v2 has a standard multi-pin connector. To connect the two you can either assemble a new cable as described in <http://www.philohome.com/nxtplug/nxtplug.htm>, or modify an original NXT cables to make a Y-type adapter (see figure 1).

I have chosen the second option to allow connecting a smart sensor and the RP6v2 to the same port (only 4 ports on NXT!). Note that RP6V2 and non-I2C sensors cannot share the same port, and that not all NXT sensors communicate via I2C (touch sensor is a switch). I am not sure about all sensors, but the RP6v2 and the NXT ultra-sound sensor apparently can share one port.

On the RP6v2 side of the cable, I used a 5-pin connector to fit pin-head “I2C”, which can be found on the top, near the rear-left corner of the RP6v2 (see figure 2). For the connector pinning, see the schematic shown in figure 3.

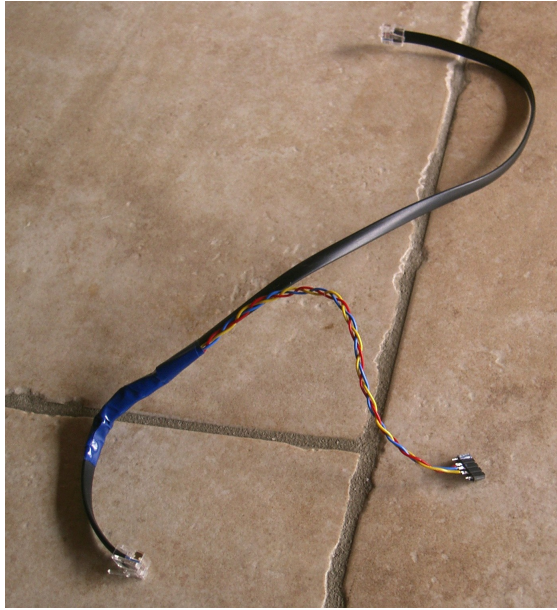


figure 1: Y-shape NXT adapter cable for RP6v2

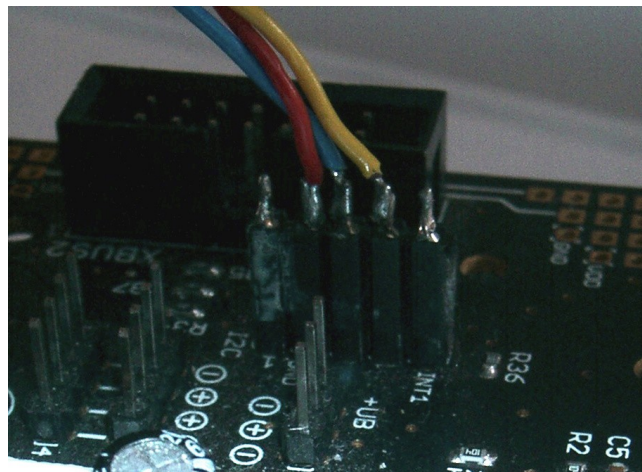


figure 2: I2C connector on RP6v2 side

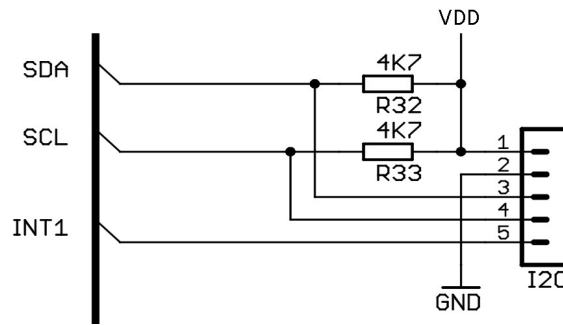


figure 3: schematics of the RP6v2 I2C connector

Number	NXT brick	RP6v2
1	Analog (white)	-
2	GND (black)	GND (red)
3	GND (red)	SDA (blue)
4	Vdd=4.3V (green)	SCL (yellow)
5	SCL (yellow)	-
6	SDA (blue)	

figure 4: NXT color coding and RP6v2 connector in figure 2

1b: Pull-Up

The I2C bus was developed by Philips Semiconductors (now NXP) in the 1980s for on-PCB communication between ICs, e.g. microcontroller and sensors. On an I2C bus, the I2C master communicates with one or several I2C slaves, which are addressed by unique identifiers.

Other than intended by NXP, the Mindstorms NXT uses the I2C bus between separate units (i.e. brick and sensors) – and in the most difficult environment imaginable, namely children's playrooms. To improve robustness against ESD^{iv}, Lego therefore deviated from the I2C standard. Specifically, they implemented a 4.7kΩ series resistor for the SDA and SCL lines^v, which are missing in standard systems. As a consequence, the standard I2C 4.7kΩ pull-up resistor between SDA/SCL and Vdd is too low-ohmic and has to be replaced with a much higher value. Lego specifies a value of 82kΩ, but seemingly 47kΩ also works.

To achieve this, resistors R32 and R33 (both 4.7kΩ) on the RP6v2 have to be replaced with 82kΩ SMD resistors. Alternatively, if you have no SMD parts available, you can also remove R32 and R33, and implement the pull-up resistors in the above described cable (which I did).

Step 2: Program the RP6v2

Since the NXT is configured as I2C master the RP6v2 has to act as an I2C slave. The provided firmware “[RP6_Firmware.zip](#)” is almost identical to the original “I2C slave” example program provided by Arexx. Specifically, only debug options have been added and unused commands

disabled. In the software the I2C identifier is set by macro `RP6BASE_I2C_SLAVE_ADR = 10` (line 37 in source file). If a different ID is needed the below NXT blocks have to be adapted correspondingly.

Reading a variable is simply by reading the respective I2C register on the RP6v2.

Sending a command to the RP6v6 requires the following steps:

1. if required, write parameters to 1B registers 1..6
2. write command identifier to register 0

For a full list of implemented I2C commands, please refer to Appendix: RP6v2 Commands. For NXT-G some read blocks (e.g. get bumper status) and command blocks (e.g. start motor) are provided as templates.

For support on how to compile and upload “[RP6_Firmware.zip](#)”, please refer to the Arexx homepageⁱⁱⁱ and RP6v2 forums on the internet.

Step 3: Install NXT Blocks

File “[NXT-RP6v2_Blocks.zip](#)” contains some blocks to control the RP6v2 via I2C and also some generic blocks. Copy all provided blocks to “~/Documents/LEGO Creations/MINDSTORMS Projects/Profiles/Standardvorgabe/Blocks/Eigene Blöcke” (on a German MacOSX system). The provided blocks for RP6v2 control are shown in figure 5.

Other functions can be added easily by modifying a read (r) or write (w) block. Section Appendix: RP6v2 Commands gives you a full list of I2C commands implemented on the RP6v2.

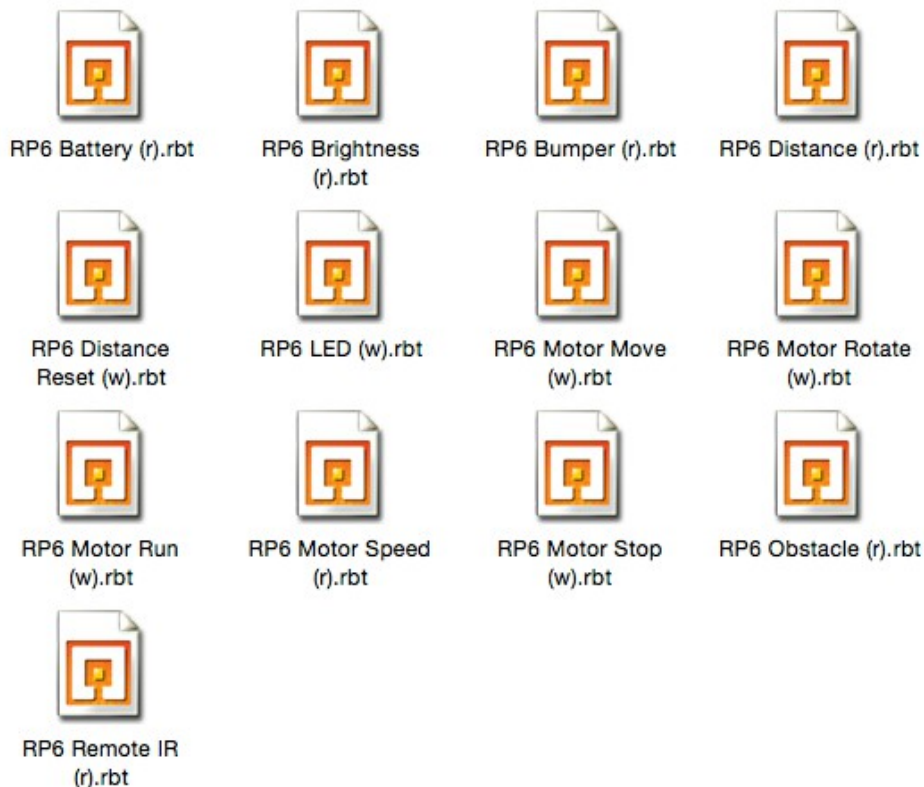


figure 5: provided NXT blocks for remote control of RP6v2 via I2C

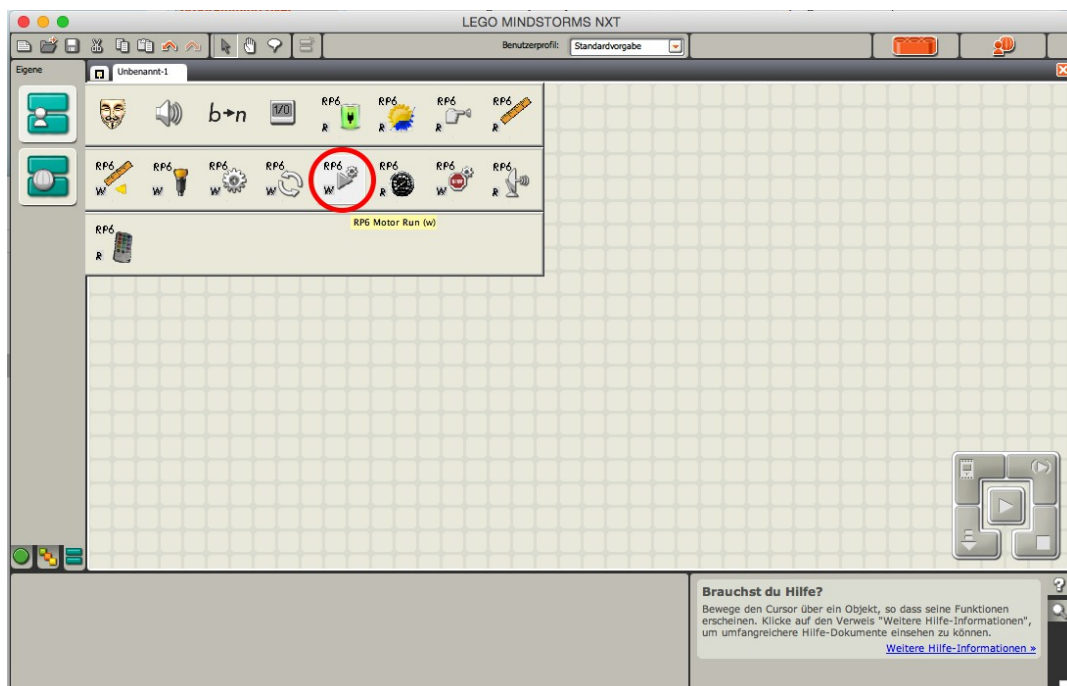
Step 4: Program the NXT

Below I give a step-by-step introduction for a small NXT/RP6v2 project. The target of the project is simply move the RP6v2 forward (\rightarrow control motors) and play a sound on the NXT when the RP6v2 hits an obstacle (\rightarrow read sensor). Before we begin please make sure that the RP6v2 is connected to NXT port 3 and is programmed as I2C slave (see Step 2: Program the RP6v2). No other motor or sensor is required on the NXT.

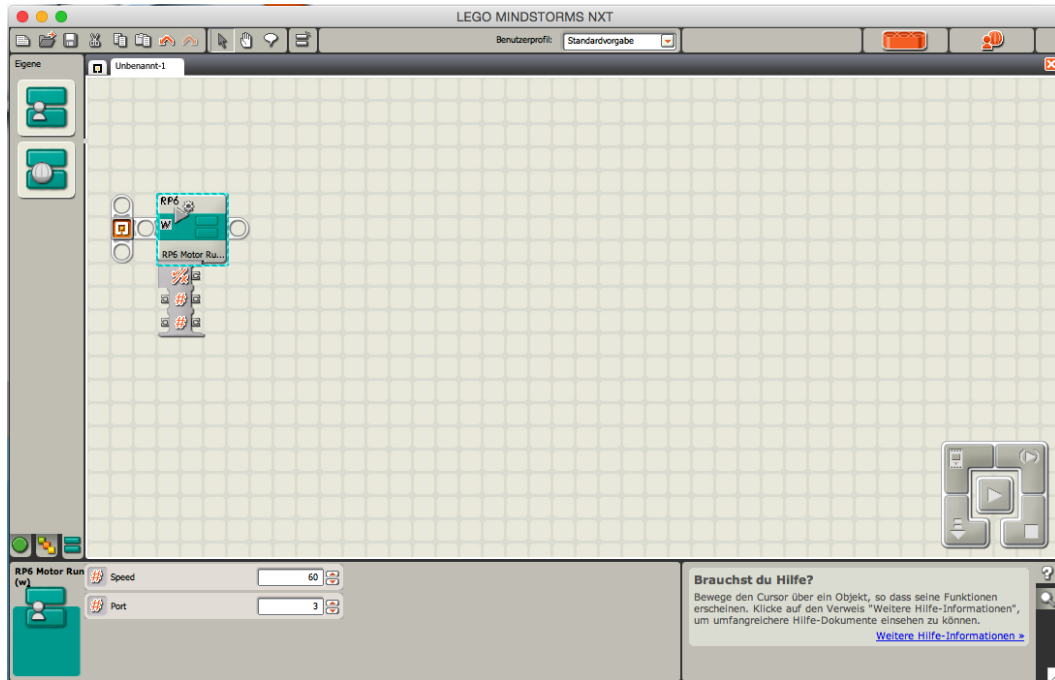
1. launch the NXT-G IDE and start a new project



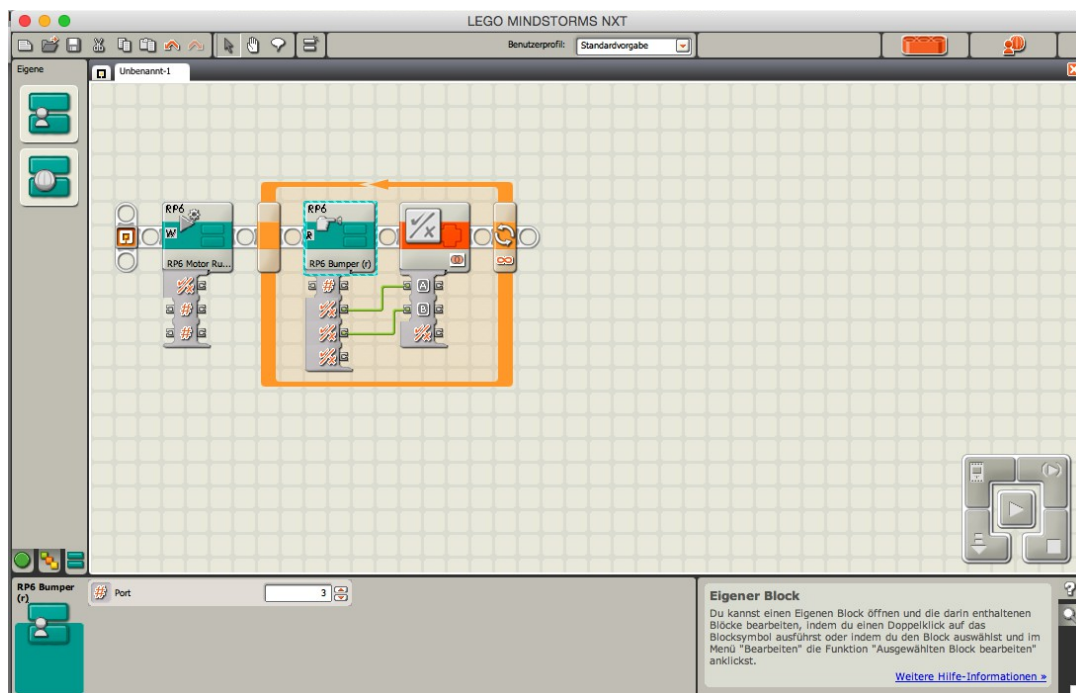
2. use “RP6v2 Motor Run (w)” from “Custom Palette” to start motors



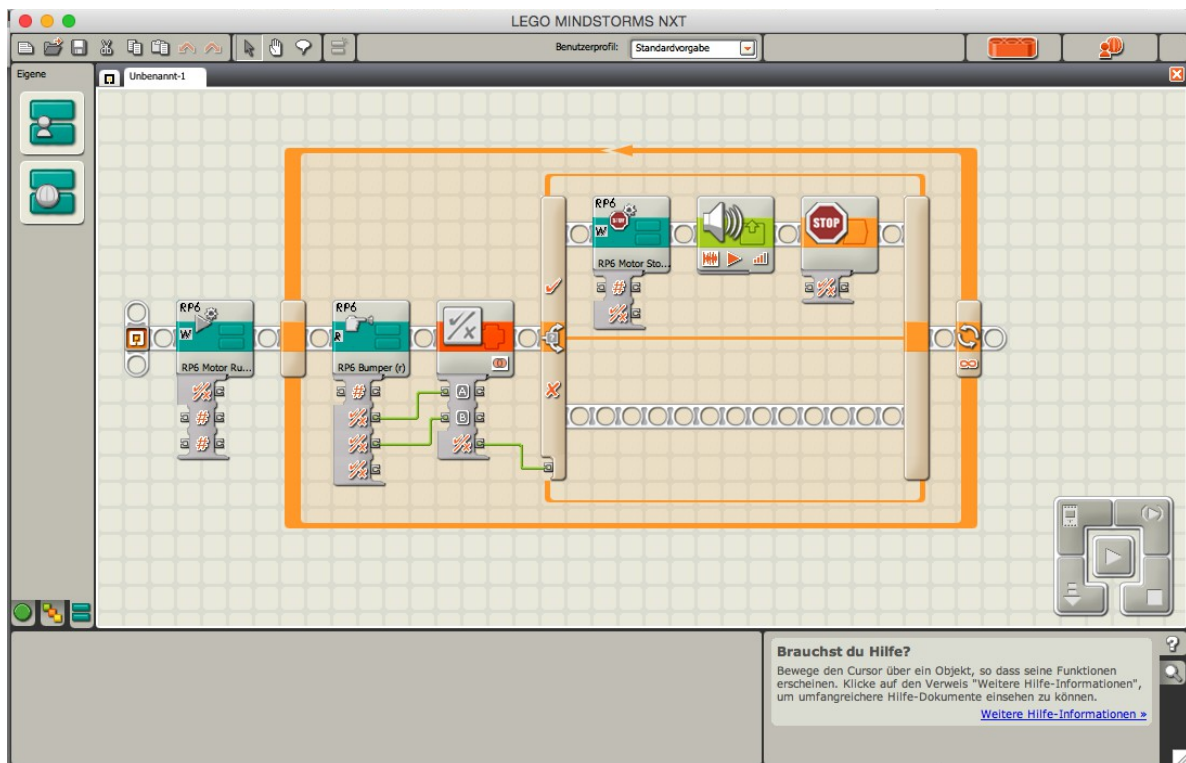
3. set port to 3 (connected to RP6v2) and motor speed e.g. to 60



4. within endless loop, use “RP6 bumper (r)” block with port=3 to check for collisions. OR the result → TRUE if any bumper is hit.



5. if bumper is hit, stop both RP6v2 motors using “RP6 stop motors (w)”, play a sound on the NXT and exit the program. The final project should look like this



6. upload above NXT-G project to the NXT and run. If you upload via USB disconnect cable before launching the program
7. there's no step 7. Now play and have fun!

Notes:

- in case of I2C communication errors, provided NXT blocks sound a warning
- RP6v2 needs to be in user mode to respond to I2C (green LEDs blink alternating every 1s)

Appendix: RP6v2 Commands

Read Registers

register	bits	description	type	NXT block
0	0	battery low warning	logic	I2C Multiple Read
	1	bumber left	logic	
	2	bumber right	logic	
	3	IR-RC5 received	logic	
	4	IR-RC5 ready to send	logic	
	5	ACS: obstacle left	logic	
	6	ACS: obstacle right	logic	
	7	motor state changed	logic	
1	0	power on	logic	I2C Multiple Read
	1	ACS active	logic	
	2	watchdog timer	logic	
	3	watchdog request	logic	
	4	watchdog active	logic	
2	0	motor activity completed	logic	I2C Multiple Read
	1	motor running	logic	
	2	motor over-current	logic	
	3..4	direction (FWD=0, BWD=1, LEFT=2, RIGHT=3)	number	
3	0..7	motor power left (0..210)	number	I2C Multiple Read
4	0..7	motor power right (0..210)	number	
5	0..7	motor speed left	number	RP6 Motor Speed
6	0..7	motor speed right	number	
7	0..7	target speed left	number	I2C Multiple Read
8	0..7	target speed right	number	
9+10	0..15	distance left (L+H)	number	RP6 Distance
11+12	0..15	distance right (L+H)	number	
13+14	0..15	brightness left (L+H)	number	RP6 Brightness
15+16	0..15	brightness right (L+H)	number	
17+18	0..15	motor current left (L+H)	number	I2C Multiple Read
19+20	0..15	motor current right (L+H)	number	
21+22	0..15	battery voltage level (L+H)	number	RP6 Battery
23+24	0..15	ADC0 (L+H) → 0..1023	number	I2C Multiple Read
25+26	0..15	ADC1 (L+H) → 0..1023	number	I2C Multiple Read
27	0..7	IR-RC5 address	number	RP6 Remote IR
28	0..7	IR-RC5 data	number	
29	0..7	state LEDs	number	I2C Multiple Read

Send Command

NXT block	description	command ID	parameter 1	parameter 2	parameter 3	parameter 4	parameter 5
RP6 LED	set LED status	3	bit mask LEDs (0x00=all off; 0x01=all on; >=bit 2 → mask)	-	-	-	-
RP6 Motor Stop	stop motor	4	-	-	-	-	-
RP6 Motor Run	start motor	5	speed left (0..255)	speed right (0..255)	-	-	-
RP6 Motor Move	drive fixed distance	7	speed (0..255)	direction (FWD 0, BWD 1, LEFT 2, RIGHT 3)	distance (HB) in encode pulses	distance (LB) in encode pulses	wait until finished?
RP6 Motor Rotate	rotate on spot	8	speed (0..255)	direction (FWD 0, BWD 1, LEFT 2, RIGHT 3)	angle (HB) in deg	angle (LB) in deg	wait until finished?
RP6 Distance Reset	reset distance measurement	13	-	-	-	-	-

- i Lego Mindstorms, see http://en.wikipedia.org/wiki/Lego_Mindstorms_NXT
- ii I2C bus, see <http://en.wikipedia.org/wiki/I%C2>
- iii RP6v2 robot, see <http://www.arexx.com/rp6/>
- iv Electrostatic discharges, see http://en.wikipedia.org/wiki/Electrostatic_discharge
- v Mindstorms NXT I2C, see <http://mbed.org/users/aworsley/notebook/lego-mindstorms-nxt-brick-interface/>