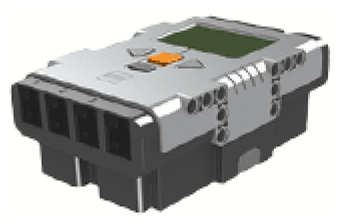
LEGO NXT: Features & Limitations

NXT Brick

The main component in the kit is a brick-shaped computer called the NXT brick. It can take input from up to four sensors and control up to three motors, via RJ12 cables, very similar to but incompatible with RJ11 phone cords. The brick has a 100x64 pixel monochrome LCD display and four buttons that can be used to navigate a user interface using hierarchical menus. It also has a speaker and can play sound files at sampling rates up to 8 kHz. Power is supplied by 6 AA (1.5 V each) batteries in the consumer version of the kit and by a Li-Ion rechargeable battery and charger in the educational version.



Technical Specifications

- 32-bit AT91SAM7S256 main microprocessor (256 KB flash memory, 64 KB RAM)
- 8-bit ATmega48 microcontroller @ 4 MHz (4 KB flash memory, 512 Bytes RAM)
- 100x64 pixel LCD matrix display
- Can be programmed using Windows or Mac OS (NBC/NXC supports Linux as well)
- A single USB 2.0 port
- Bluetooth (Class II) wireless connectivity, to transfer programs to the NXT wirelessly or offer ways to control robots remotely (through mobile phones and possibly by PDA's)
- 4 input ports, 6-wire cable digital platform (One port includes a IEC 61158 Fieldbus Type 4/EN 50 170 compliant expansion port for future use)
- 3 output ports, 6-wire cable digital platform
- Digital Wire Interface, allowing for third-party development of external devices
- Other software can also be used.

Firmware and Developer Kits

Lego has released the firmware for the NXT Intelligent Brick as Open Source.
Several developer kits are available that contain documentation for the NXT:

- Software Developer Kit (SDK), includes information on host USB drivers, executable file format, and byte code reference
- Hardware Developer Kit (HDK), includes documentation and schematics for the NXT brick and sensors
- Bluetooth Developer Kit (BDK), documents the protocols used for Bluetooth communications

Memory

The NXT uses a 32-bit ARM7 microprocessor with 256 Kbytes of Flash and 64 Kbytes of Ram. Flash memory can be read like RAM (access is a bit slower) but can only be written in 256-byte pages (maximum of 767 pages) by specific hardware instructions. Flash memory cannot be read while a page is being written. The first two pages hold the file table (directory) and the rest of the pages hold user files. Files are held as a contiguous set of bytes – i.e. they use a single range of page numbers with no gaps. This allows a file to be addressed as a region of memory.

The leJOS NXJ firmware is written in a combination of C and ARM assembler code. It consists of the initialization code, the Java VM and device drivers for all the hardware subsystems. The leJOS firmware is a complete firmware replacement and has no reliance on the standard LEGO firmware. The first 32kb of flash memory is allocated to the leJOS NXJ firmware. Most code is executed from flash memory, but a small amount (e.g. the code that writes pages of flash memory) is copied to RAM. Read-only data is held in flash memory but read/write data is copied to RAM. The firmware uses a fixed size stack and interrupt stack.

The leJOS NXJ Java VM executes one Java program at a time. This can either be a user program or the leJOS start-up menu. One Java program can execute another. When this is done the first Java program is removed from memory, and the second one is then executed. This is how the start-up menu executes user programs.

The start-up menu occupies up to 48kb of memory that starts at address 32k (i.e. after the end of the firmware). The last word of the 48kb allocated to the start-up menu holds the size of the start-up menu). Java programs execute from flash memory. Static read-only data is held in flash memory. Static read-write data is copied to RAM. Objects are created in a heap that starts at the top of the RAM and grows downwards. The Java stack starts at the bottom of free RAM memory and grows up. A garbage collector frees memory used by un-referenced objects when the heap becomes full.

The maximum filename length is 20 characters. If the average filename length is 15 characters, only 10 files can be supported. This limitation is not checked, and will cause an exception when the file table becomes full. Also, only one file can be open at a time.

<u>Power Supply</u>

Power is supplied to the NXT using 6 AA/LR6 batteries or a rechargeable battery pack with a power adapter plug (120VAC 60Hz). The rechargeable battery pack takes up more physical space than 6 AA batteries. The NXT comes with a cover that sits flush with the NXT unit while the rechargeable battery pack protrudes almost a centimetre out from the NXT brick. When one builds a robot from existing build instructions, this will have to be taken into account.

When charging the NXT, the green indicator light turns on. When the battery is recharging, the red light is on. A fully charged battery requires approximately four hours. The NXT can be used when the battery is recharging; however, recharging then requires more time. The Li-Ion Polymer battery can be recharged up to 500 times.

There is a difference in performance (power and time) when it comes to using 6 AA's vs. the rechargeable battery pack. 6 AA's will provide more power to the NXT motors than the Li-Ion battery pack will. In fact, in competitions, the preferred choice is Energizer Lithium batteries over conventional alkaline batteries, when power is of the utmost importance. Although the Li-Ion battery pack provides less power than 6 AA's, it delivers more steady power over a longer period of time.

It is difficult to quantify the time it takes to fully discharge the alkaline batteries or the Li-Ion battery pack. This depends on what types of tasks the NXT is performing (all motors turning at full speed, or just data calculations). One thing is for certain, as the batteries begin to deplete, the power of the motors begin to deplete as well.

The NXT has a sleep option to conserve power. These settings can be changed in the menu system under 'Settings/Sleep/Never'. You can change the Settings to wait before going to sleep from 2, 5, 10, 30 or 60 minutes.

Usable Programming Languages

Very simple programs can be created using the menu on the NXT Intelligent Brick. More complicated programs and sound files can be downloaded using a USB port or wirelessly using Bluetooth. Files can also be copied between two NXT bricks wirelessly, and some mobile phones can be used as a remote control. Up to three NXT bricks can communicate simultaneously (via Bluetooth) when user created programs are run.

The retail version of the kit includes software for writing programs that run on PC and Macintosh personal computers. The software is based on National Instruments LabVIEW and provides a visual programming language for writing simple programs and downloading them to the NXT Brick.

NXT-G

NXT-G v1.0 is the programming software that comes bundled with the NXT. There are two different programming interfaces. One is included with the retail and educational kits and the other can be purchased separately. This software is adequate for basic programming, such as driving motors, incorporating sensor inputs, doing calculations, and learning simplified programming structures and flow control. Here are some advantages and disadvantages of using version 1.0 of this software:

Pros:

- NXT-G is easy to install on Windows XP, Windows Vista and Windows 7 machines, and Mac OS X is also supported.
- NXT-G can transfer data via Bluetooth or included USB cable.
- NXT-G provides an easy to use, drag and drop, graphical environment.
- The graphics include data wires that show data flow from block to block.

Cons:

- NXT programs can be much larger than identical programs developed with a third party programming language (e.g. 12 kiB versus 2 kiB).
- Programs take substantially longer to load than third party programs.
- When creating large programs, NXT-G tends to crash and lose unsaved data.
- NXT-G software usually runs sluggishly, even on powerful PCs.

Most of these issues have been addressed in NXT-G v1.1 version of the software.

Unofficial Programming Languages

While NXT-G provides a colourful and user friendly approach to programming NXT robots, more advanced robot builders and programmers may prefer programming with a text-based language, which involves writing lines of code rather than dragging and dropping code blocks. There is quite a large variety of unofficial text-based languages for the NXT freely available on the Internet. Generally speaking, these programming languages offer greater control but can be more difficult to learn. An extensive (but not exhaustive) list follows:

Name	Language type(s)	Notes
Actor-Lab	Custom flowchart-like language	
Ada	Ada	Requires nxtOSEK
brickOS	C/C++	
Ch	C/C++ Interpreter	Control Lego Mindstorm in C/C++ interactively without compilation
FLL NXT Navigation	Uses NXT-G and .txt files	
GCC	C/C++, Objective C, Fortran, Java, Ada among others	
jaraco.nxt	Python	Python modules providing low-level interfaces for controlling a Lego NXT brick via Bluetooth. Also includes code for controlling motors with an Xbox 360 controller using pyglet.
LabVIEW	National Instruments LabVIEW Visual programming language (G code)	Core language used to develop Mindstorms NXT software. Can use available add-on kit to create and download programs to NXT, create original NXT blocks or control robot directly via USB or Bluetooth using NXT fantom.dll
Lego.NET	Anything that can compile to .NET, works best with C#	Does not come with a compiler, converts bytecode to machine code
Lego::NXT	Perl	Set of Perl modules providing real-time low-level control of a Lego NXT brick over Bluetooth.
LegoNXTRemote	Objective C	Remote control program for remotely operating and programming a Lego NXT Brick. Supports NXT 2.0 and 1.0, sensors, all 3 motors, automatic "steering" control, and running preloaded programs.
leJOS	Java	A java based system for advanced programmers can handle most sensors and things like GPS, speech recognition and mapping technology. Can be interfaced with the Eclipse IDE or run from the command line
NXTGCC	Assembler, C, makefiles, Eclipse, etc.	The first GCC toolchain for programming the Lego Mindstorms NXT firmware.
nxtOSEK	C	
librcx	C/C++	A library for GCC
MicroWorlds EX Robotics Edition		This is a program in the MicroWorlds series that allows students to control the NXT.
NQC	NQC, a C-like language	This is the most widely used unofficial language

Name	Language type(s)	Notes
NXT++	C++	Allows you to control the NXT directly from any C++ program, in Visual Studio, Windows.
NXT_Python	Python	NXT_Python is a package for controlling a LEGO NXT robot using the Python programming language. It can communicate using either USB or Bluetooth.
NXT-Python	Python	NXT-Python is a newer version of NXT_Python, and has some extra capabilities. The svn repository is at Google code. Anyone can contribute, just ask.
Lestat	C++	Allows you to control the NXT directly from any C++ program in Linux.
OCaml Mindstorm	OCaml	Module to control LEGO NXT robots using OCaml through the Bluetooth and USB interfaces.
Mindstorms SDK	Visual Basic, Visual C++, MindScript, LASM	You do not need VB to use the VB features as MS Office comes with a cut down version of VB for making macros
PBrickDev	PBrickDev, a flowchart based language.	Has more functionality than the RIS language, such as datalogs and subroutines/multithreading.
PRO-BOT	A kind of Visual Basic/spirit.ocx-based language	Designed for robots which are in contact with the workstation at all times
QuiteC	C	A library for use with GCC and comes with GCC for Windows.
RCX Code	RCX Code, a custom flowchart-based language	Included in the Mindstorms consumer version sold at toy store
ROBOLAB	A flowchart language based on LabVIEW	This is the programming environment offered to schools who use MindStorms, supports the Lego Cam
RoboRealm	A multi-platform language that works with IRobot Roomba, NXT, RCX, VEX, and many other popular robotic sets.	This language is also capable for video processing using a webcam; this gives your robot excellent vision since it can filter out certain colors, lock-on to a certain area of color, display variables from the robot or computer, and much more. The software works with keyboard, joystick, and mouse. This software is freeware.
ROBOTC	A multi-platform C programming language designed for the programmer in need of powerful debugging tools for the NXT, RCX, VEX, and soon-to-be FIRST Controller (for FRC).	ROBOTC gives the ability to use a text-based language based on the C programming language. It includes built-in debugger tools, as well as (but not limited to) code templates, Math/Trig operations (sin, cos, tan, asin, acos... etc), user-friendly auto-complete function built into the interface, built-in sample programs

Name	**Language type(s)**	**Notes**
ruby-nxt	Ruby	Provides low-level access to the NXT via Bluetooth as well as some preliminary high-level functionality.
RWTH – Mindstorms NXT Toolbox	MATLAB	Interface to control the NXT from MATLAB via Bluetooth or USB (open-source).
Gostai URBI for Lego Mindstorms NXT	URBI, C++, Java, Matlab	Easy to use parallel and event-driven script language with a component architecture and open source interface to many programming languages. It also offers voice/speech recognition/synthesis, face recognition/detection, Simultaneous localization and mapping, etc.
Vision Command	RCX Code	The official programming language for use with the Lego Cam. It allows you to control your robot with color, motion, and flashes of light.
LegoLog	Prolog	Uses an NQC program to interpret commands send from the PC running the Prolog code
Microsoft Visual Programming Language (VPL)	Graphical flowchart, based on .NET	With the Microsoft Robotics Studio, it uses a native NXT program *msrs* to send and receive messages to and from a controlling program on a computer via Bluetooth
DialogOS	Graphical Flowchart for voice controlled robots	DialogOS combines speech recognition and speech synthesis with robotics, enabling you to build talking robots that react to your voice commands.
Processing	Java (Simplified / programmed C-style)	Processing (programming language) is an open source programming language and environment for people who want to program images, animation, and interactions. It is used by students, artists, designers, researchers, and hobbyists for learning, prototyping, and production. To control the NXT with Processing you can use the NXTComm Processing library developed by Jorge Cardoso.
Interactive C	C-style language.	Language developed for the Botball robotics competition
pbLua	API for the Lua programming language for the Mindstorms NXT, text-based	pBLua: ... is written in portable C, with minimal runtime requirements; can be compiled on the fly on NXT; is a small, easy to read, and easy to write language; has extensive documentation available online and in dead-tree format, and a very friendly newsgroup

Sensors

One of the most important components of a robot is the sensor. Much of the NXT set's potential lies in its excellent selection of sensors. The primary purpose of a sensor is to allow the robot to interact with its environment and perform actions based on feedback from its surroundings.

This process is called *autonomy,* which by definition means freedom from external law. An autonomous robot is a self-governing device that takes input from its sensors and makes decisions based on this input.

Passive Sensors
The NXT set includes three passive sensors: the touch sensor, the light sensor, and the sound sensor. Passive sensors either do not require power from the NXT in order to function or do not require the NXT to employ a process that rapidly switches between supplying power to and reading the value of a sensor.

Touch Sensor
The touch sensor is useful for a wide variety of applications. The orange missile-shaped tip at the front end, known as the push button, provides the touch-sensing abilities. When pressed, the push button

moves backward into the sensor and completes an electrical circuit, resulting in a flow of electricity detectable by the NXT. When released, the push button springs forward, the circuit is broken, and the electrical flow stops. Hence, this simple configuration allows for only two possible conditions: pressed or released.

Just as the touch sensor can have two different conditions, it can produce two different values or readings. If the push button is pressed, the NXT reads the touch sensor as having a value of 1; if it's not pressed the NXT reads a value of 0. You can use these values in several ways, however. For example, you can use any of the following conditions to trigger a reaction in a robot:

- When the touch sensor is pressed
- When the touch sensor is released
- When the touch sensor is pressed and released (i.e., “bumped”)

If you look closely at the push button, you’ll notice a shaft shaped hole. This hole accommodates LEGO axles, allowing you to customize the push button. You can use the touch Sensor to make your robot pick up things: a robotic arm equipped with a Touch Sensor lets the robot know whether or not there is something in its arm to grab. Or you can use a Touch Sensor to make your robot act on a command. For example, by pressing the Touch Sensor you can make your robot walk, talk, close a door, or turn on your TV.

Light Sensor

Capable of measuring light intensity, the NXT light sensor can determine the brightness or darkness of its surrounding area as well as the light intensity of surfaces, enabling it to (indirectly) distinguish between surfaces of different colors. The NXT generates the light sensor’s readings as a percentage. The highest possible reading is 100 percent, which you can easily achieve by holding the sensor up to a light bulb. The lowest possible reading is 0 percent, which you could achieve in a very dark closet.

If you look at the front of the light sensor, you’ll notice two small bulbs poking out. The one on top is a *phototransistor*, which measures the light; the one on the bottom is a *light-emitting diode* (LED), which shines a bright red light. When the light sensor is positioned closely to a surface, the LED increases the amount of reflected light that the sensor reads; likewise increasing the sensor’s sensitivity to different colors (different colors reflect light differently). You can turn off the LED in a program, however, if you want the light sensor to detect only the surrounding or ambient light.

You can use the light sensor to create line following robots that use the light sensor to follow a line. Or perhaps you could create a robot that uses the light sensor to measure a room’s overall light level and leaves the room when you turn off the lights.

Sound Sensor

The NXT sound sensor can not only detect the volume of sound (amplitude) but also sound patterns. While it cannot distinguish between types of sounds, such as between the sound of a bird and the sound of a cat, it still allows from many creative applications.

Although the NXT reads the sound sensor’s value in decibels (dB, includes sounds inaudible to humans) or adjusted decibels (dBA, only sounds audible to humans), it reports the value as a percentage, with 100 percent being the highest and 0 percent being the lowest. For example, a silent room will return a value of 4 to 5 percent, distant talking will return a value of 5 to 10 percent, and a regular conversation or music playing at a moderate volume will return a value of 10 to 30 percent, and yelling or loud music will return a value of 30 to 100 percent. The maximum sound level that the sound sensor can determine is 90 dB (about the level of a lawnmower). It can hear samples between 20 and 30 Hz (not fast enough to recognize human speech).

The sound sensor definitely provides exciting possibilities for NXT robots. One of the simplest applications is a sound-activated robot that begins to operate and/or stops operating upon hearing a loud sound (such as a verbal command). Another possibility is a robot that attempts to escape from sound by finding a more peaceful spot.

Digital Sensors

The NXT set includes three types of Digital Sensors: the ultrasonic sensor, the colour sensor and rotation sensors. An NXT digital sensor has two important characteristics. It has its own microcontroller, which enables the sensor to take readings of its environment itself (as opposed to the NXT doing it), and it sends its data to the NXT using I^2C communication, which allows the sensor to operate independently and transmit only its readings to the NXT.

Ultrasonic Sensor

The Ultrasonic Sensor is one of the three sensors that give your robot vision [The Light Sensor and Color Sensor are the others]. The Ultrasonic Sensor enables your robot to see and detect objects. You can also use it to make your robot avoid obstacles, sense and measure distance, and detect movement.

The Ultrasonic Sensor measures distance in centimetres and in inches. It is able to measure distances from 0 to 255 centimetres with a precision of +/- 3 cm.

The Ultrasonic Sensor uses the same scientific principle as bats: it measures distance by calculating the time it takes for a sound wave to hit an object and return – just like an echo. Large sized objects with hard surfaces return the best readings. Objects made of soft fabrics or those that are curved [like a ball] or are very thin or small can be difficult for the sensor to detect.

* Note that two or more Ultrasonic Sensors operating in the same room may interrupt each other's readings.

The Ultrasonic Sensor operates in two modes *Continuous* (default) and *Ping*. When in *Continuous* mode the sensor sends out pings as often as it can and the most recently obtained result is available to the NXT through a function call (to getDistance() when using LeJos). The function called depends on the programming language being used.

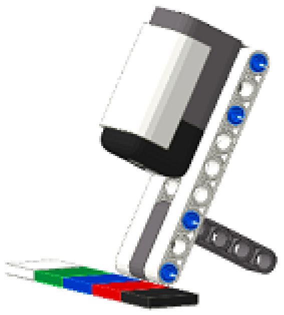
When in Ping mode, a ping is sent only when a function call is made (to ping() when using LeJos). Invoking the ping() method switches the sensor into ping mode and sends a single ping. From this single ping, up to 8 echoes are captured. The return value of a ping is in centimetres. If no echo was detected, the returned value is 255.

These echoes may be read by making another function call (to int readDistances(int [] distances) when using LeJos). You provide an integer array of length 8 that contains the data after the method returns. A delay of approximately 20ms is required between the call to ping() and getDistances(). This delay is not included in the method. Calls to getDistances() before this period may result in an error or no data being returned. The normal getDistance() call may also be used with ping, returning information for the first echo.

Colour Sensor

The Colour Sensor is one of the three sensors that give your robot vision [The Light Sensor and Ultrasonic Sensor are the others]. The newest Colour Sensor (from HiTechnic) operates by using a single white LED (light emitting diode) to illuminate the target and analyses the colour components of the light reflected by the target's surface and calculates a *Colour Number* that is returned to the NXT program.

The Colour Sensor works best when it is positioned so that it is not too close to the surface being tested. The picture on the right should be used as a guide for proper distance and angle to the surface. The angle prevents the direct reflection of the light from the LED from coming back into the sensor element, which can prevent proper colour determination.



The Colour Sensor connects to an NXT sensor port using a standard NXT wire and digital I2C communications protocol. The *Colour Number* calculated by the sensor is refreshed approximately 100 times per second. It can detect an extended range of more than 15 target colours and the NXT robot can be programmed to react to each colour.

The HiTechnic Colour Sensor is "tuned" to standard LEGO colours. When positioned over a surface, the Colour Sensor will return a numeric value identifying the target colour. For more in depth information visit the HiTechnic Colour Sensor page.

The Servo Motor Encoder (Rotation Sensor)

NXT robots can move in many different ways. They can grasp, race, walk, swivel and do much more. These capabilities come from using the NXT servo motors. Servo motors are different from other common Lego motors. They are interactive, meaning that they include a built-in Rotation Sensor or tachometer. This lets you control your robot's movements precisely.

The Rotation Sensor measures motor rotations in degrees or full rotations [accuracy of +/- one degree]. One rotation is equal to 360 degrees, so if you set a motor to turn 180 degrees, its output shaft will make half a turn. Of course, you can also simply instruct the motor to run indefinitely or for a specified amount of time.

The built-in Rotation Sensor in each motor also lets you set different speeds for your motors, and there are definitely times when you will want to run the motors at less than full power. This is easily done by setting different power parameters in the software (from -100 to 100).

By connecting Lego pieces to the shaft heads, you can transfer power from the motor to your creation. The motor itself also has several places for attaching Lego pieces. These are most often used to secure the motor in place.

For more in depth information on the Servo motor including characteristics, efficiency, charts (power, speed, torque, etc.) and more you can visit this resource titled: NXT Motor Internals.

Sources:

http://lejos.sourceforge.net/nxt/nxj/tutorial/AdvancedTopics/UnderstandingFilesLCPMemTools.htm
http://cache.lego.com/downloads/education/9797_LME_UserGuide_US_low.pdf
http://student.seas.gwu.edu/~darbyt/cs1/lejos.html
http://www.comp.dit.ie/jkelleher/rtf/classmaterial/week5/NXT-Sensors.pdf
http://nxtguide.davidjperdue.com/
http://www.legoengineering.com/nxt-sensors-2.html
http://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=NCO1038
http://www.philohome.com/nxtmotor/nxtmotor.htm
http://www.enotes.com/topic/Lego_Mindstorms#Programming_languages_2