



## What is LineLeader-v2

LineLeader-v2 is an array of 8 sensors to detect color equivalent gray scales in front of the sensor window. It has a built in light source for each of its sensors and it can detect dark lines on light backgrounds (or vice versa).



**This sensor is fully compatible with former NXTLineLeader software.**

## Connections and Placement

LineLeader-v2 can be connected to any of the four sensor ports of NXT or EV3 by using standard cables from NXT set, or FlexiCable from [mindensors.com](http://mindensors.com).

While attaching sensor to your robot, face the long clear sensor window towards the mat, placing it approx 5 millimeters above the mat surface (approx  $\frac{1}{4}$  inch).

Place the sensor in front of the driving wheels - (in the direction the robot will be moving).

### Mounting LineLeader-v2 on your contraption

The holes on the LineLeader-v2 enclosure are designed for tight fit of Technic pins (or axles) with '+' cross section. The holes however are not designed for repeated insertions/removals of these pins.



To mount LineLeader-v2 on your contraption we suggest that you use two dark gray 'Technic Axle 3 with Stud' as shown.

Insert axles from the top of the LineLeader-v2 and secure with a bushing on the back or mount it on your contraption directly.

Alternately, you may use blue 'Technic Axle Pin with Friction', as shown.



While disassembling contraption, leave the pins on LineLeader-v2.

## Sensor Calibrations



### NOTE

**For best results, LineLeader-v2 should be calibrated for the mat it will be used on.** To calibrate the sensor, keep the sensor on white area, and calibrate white. Then keep the sensor window on black area and calibrate black. While placing on the white or black area, ensure that all sensors are over the line.

Calibration programs can be found in the sample programs of the respective programming environment. Download sample programs from the Programming Techniques section below.

## Sensor Operations

You can use the raw sensor readings of the sensor and compute yourself whatever you need from it. The raw values are returned as byte per sensor with 0 indicating black and 100 indicating white.



### NOTE

In order to write a good line follower program, you will need to know and understand the '**PID Control Theory**'. <http://www.google.com/search?q=PID+control+theory>  
[http://en.wikipedia.org/wiki/PID\\_controller](http://en.wikipedia.org/wiki/PID_controller)  
[PID without a PhD](#)

LineLeader-v2 also implements a reference PID algorithm.

This algorithm provides a steering value which you can add to your left wheel power and subtract from right wheel power to get the line tracking on your robot. You can tune this algorithm by changing PID parameters of the sensor to match your robot's characteristics, and get better performance, however, for best performance, write your own program.

## Programming Techniques

### EV3:

To use capabilities of the sensor, please download EV3 blocks available at following URL:

[http://www.mindsensors.com/index.php?controller=attachment&id\\_attachment=189](http://www.mindsensors.com/index.php?controller=attachment&id_attachment=189)



Installation instructions for EV3 block are available at:

<http://www.mindsensors.com/content/13-how-to-install-blocks-in-ev3>

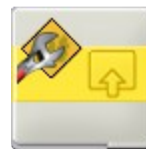
Also download sample programs from following location, and modify to suit your needs.

[http://www.mindsensors.com/index.php?controller=attachment&id\\_attachment=190](http://www.mindsensors.com/index.php?controller=attachment&id_attachment=190)

### NXT-G:

Download the NXT-G blocks available in the NXT-G Blocks Repository at Mindsensor's website, at following location:

[http://www.mindsensors.com/index.php?controller=attachment&id\\_attachment=57](http://www.mindsensors.com/index.php?controller=attachment&id_attachment=57)



Installation instructions for NXT block are available at:  
<http://www.mindsensors.com/content/21-nxt-g-blocks-how-to-install-blocks>

Also download sample programs from following location, and modify to suit your needs.

[http://www.mindsensors.com/index.php?controller=attachment&id\\_attachment=58](http://www.mindsensors.com/index.php?controller=attachment&id_attachment=58)

### RobotC:

Download library file and sample program from following location:

<https://github.com/botbench/robotcdriversuite>

### NXC:

Download the sample programs and library file available at following location, and include the library file it in your program by #include directive as:

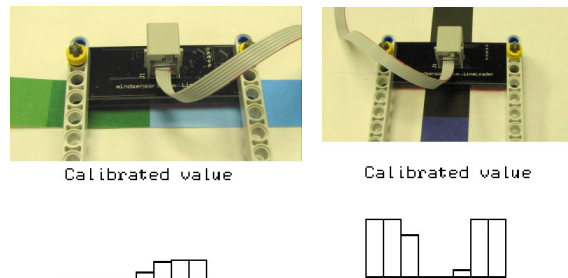
`#include "LL-lib.nxc"`

[http://www.mindsensors.com/index.php?controller=attachment&id\\_attachment=54](http://www.mindsensors.com/index.php?controller=attachment&id_attachment=54)

Alternately, you can modify the sample programs to suite your needs.

## Sensor readings on black line or color:

Adjacent pictures illustrate the readings you would get when the sensor is on a line (or color). The top part of the picture shows the sensor placement, where as bottom part of picture is a Screen capture of NXT running a program graphing sensor values.



The High bars indicate brighter surface under the sensors, whereas low bars indicate dark line or color under the sensor.

## I2C Registers:

The LineLeader-v2 appears as a set of registers as follows:

Register	Read	Write
----------	------	-------

0x00-0x07	Firmware version - Vxxxx	-
0x08-0x0f	Vendor Id - mndsnsrs	-
0x10-0x17	Device ID - LineLdr	-
0x41	-	Command
Register	Read	Write
0x49 - 0x50	Calibrated Sensor reading - Reading measured for each of the sensor in the array. Higher value indicates brighter object. (one byte for each sensor - 8 bytes)	-
0x51 - 0x58	White Reading Limit - Calibration value for White color for each sensor. (one byte for each sensor - 8 bytes)	-
0x59 - 0x60	Black Reading Limit - Calibration value for Black color for each sensor (one byte for each sensor - 8 bytes).	-
0x64-0x6B	White Calibration data	-
0x6C-0x73	Black Calibration data	-
0x74-0x83	Uncalibrated sensor voltage (Integer values, two bytes for each sensor) * (supported in LL firmware 1.15 or higher, if you need to upgrade firmware, please refer to section about Upgrading Firmware at the bottom of this document).	-

<b>Internal reference PID implementation related registers:</b>		
0x42	Steering - this is the power value returned by the sensor to correct your course. Add this value to your left motor and subtract from right motor.	-
0x43	Average - This is the weighted average of the sensor reading (explained below).	-
0x44	Result - this is the byte showing 8 sensors' readings (explained below)	-
0x45	Set Point	Set Point - you can set a non-standard value for this and the sensor will provide steering to match this value.
0x46	Kp	This value divided by PID Factor for Kp is the Proportional value for the PID control. ** Suggested value is 25 with a divisor factor of 32 (which is also a factory default), start with this value, and tune it to meet your needs. Value ranges between 0 and 255.
0x47	Ki	This value divided by PID Factor for Ki is the Integral value for the PID control. ** Suggested value is 0 with a divisor factor of 32 (which is also a factory default), start with this value, and tune it to meet your needs. Value ranges between 0 and 255.
0x48	Kd	This value divided by PID Factor for Kd is the Derivative value for the PID control. ** Suggested value is 8 with a divisor factor of 32, (which is also a factory default), start with this value, and tune it to

		meet your needs. Value ranges between 0 and 255.
0x61	The Kp divisor factor. The Kp value is divided by this number.	Value ranges between 1 and 255, (Default 32). Change this value if you need more granularities in Kp value.
0x62	The Ki divisor factor. The Ki value is divided by this number.	Value ranges between 1 and 255, (Default 32). Change this value if you need more granularities in Ki value.
0x63	The Kd divisor factor. The Kd value is divided by this number.	Value ranges between 1 and 255, (Default 32). Change this value if you need more granularities in Kd value.

\*\* In other words:

Kp value = (value in register 0x46) / (value in register 0x61)

Ki value = (value in register 0x47) / (value in register 0x62)

Kd value = (value in register 0x48) / (value in register 0x63)

### Supported I2C Commands:

CMD	Description
W	Calibrate White
B	Calibrate Black
D	Put Sensor to sleep
P	Wake up the sensor
I	Color inversion (White line on a black background)
R	Reset Color inversion (black line on a white background, this is also the default).
S	Take a snapshot, this command looks at the line under the sensor and stores the width and position of the line in sensor's memory. Subsequently, sensor will use these characteristics of line to track it. This command inverts the colors if it sees a white line on black background. (PID parameters are not affected).
A	Configure Sensor for US region (and regions with 60 Hz electrical frequency).
E	Configure sensor for European region (and regions with 50 Hz electrical frequency)
U	Configure sensor for universal frequency (in this mode the sensor adjusts for any frequency, this is also the default mode).

### Reading Frequency

The array of sensors is reading at 7 milli-seconds interval.

Note however, your reading speed from device is throttled by the I2C speed of your firmware.

## Physical Specs

Weight: 0.53 oz (14.8 grams)  
Foot-print: 24.5mm x 72.6 mm  
Height: 20.75 mm

## Current Consumption

To conserve power, the sensor goes to sleep after 1 minute of inactivity. The sensor will wake up on its own when any activity begins. You can also wake it up (or put to sleep) via command. Average measured current profile is as follows:

Current Consumption	Duration
5.7 mA (max)	While awake Depending on your mat colors ratio, this will vary between 3.5 mA to 5.7 mA
3.0 mA	While sleeping

## I2C Bus address

**Factory Default Address: 0x02**

### Changing the I2C Bus Address:

The I2C bus address of LineLeader-v2 can be changed. To set an address different from default address, send sequence of following commands on the command register:

0xA0, 0xAA, 0xA5, <new I2C address>

**Note:** Send these commands with no break/read operation in between. This new address is effective immediately. Please note down your address carefully for future reference.

**Instructions for changing the I2C address can be found at:**

NXT and EV3

<http://www.mindsensors.com/blog/how-to/change-i2c-device-address>.

PiStorms

<http://www.mindsensors.com/blog/how-to/change-i2c-device-address-with-pistorms>

## Sensor Values Explained

**Steering:**

In the simplest form, the sensor will try to maintain line at the center of the sensor. In order to do so, your robot may have to apply different power to left or right motors. This value is provided as the correction needed. You can add this value to your left motor and subtract from right motor. Alternately, you can perform other operations with this value, based on your robot's and course's needs.

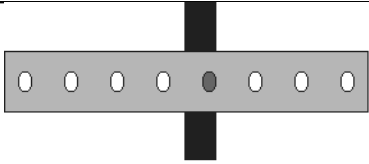
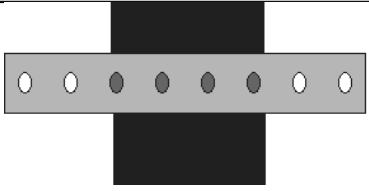
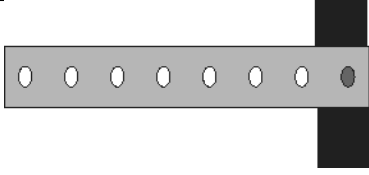
**Result:**

Result is a byte value of the sensor reading. Each bit corresponding to the sensor where the line is seen is set to 1, or else the bit is zero.

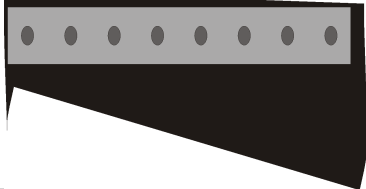
**Average:**

The average is a weighted average of the bits set to 1 based on the position. i.e. left most bit has weight of 10, second bit has weight of 20.

Table below explains the line scenario and the Result and Average returned.

Line Scenario	Values returned
	<p>Suppose your line is narrow and only under the sensor 5. I.e. all the 8 sensors except the 5<sup>th</sup> sensor will return zero value. And the 5<sup>th</sup> bit will be 1.</p> <p>The <b>'Result'</b> returned in this case will be binary: 0b00001000 (or decimal: 8 or hex: 0x08).</p> <p>The <b>'Average'</b> returned in this case will be 50, calculated as:  <math>(0+0+0+0+50+0+0+0)/1</math></p>
	<p>Suppose, your line is wider, and 4 sensors are over the line.</p> <p>The <b>Result</b> returned in this case will be binary: 0b00111100 (or decimal: 60 or hex: 0x3C)</p> <p>The <b>'Average'</b> returned in this case will be 45, calculated as:  <math>(0+0+30+40+50+60+0+0)/4</math></p>
	<p>The <b>'Result'</b> returned in this case will be binary: 0b00000001 (or decimal: 1 or hex: 0x01).</p> <p>The <b>'Average'</b> returned in this case will be 80, calculated as:  <math>(0+0+0+0+0+0+0+80)/1</math></p>



Line Scenario	Values returned
	<b>Exception:</b> When the sensor is fully on black surface (i.e. Result is 0b11111111) The ' <b>Average</b> ' returned will be 0.

#### Set Point:

The Set Point is a value you can ask sensor to maintain the average to. The default value is 45, whereby the line is maintained in center of the sensor. If you need to maintain line towards left of the sensor, set the Set Point to a lower value (minimum: 10). If you need it to be towards on the right of the sensor, set it to higher value (maximum: 80). Set point is also useful while tracking an edge of dark and light areas.

### Doing your own PID Control

If you wish to write your own PID control, you should use 'Result', 'Average' and/or Sensor Raw values in your program.

### Reference Links

Below is an excellent reference describing PID control:

[http://en.wikipedia.org/wiki/PID\\_controller](http://en.wikipedia.org/wiki/PID_controller)

A Google search on PID Control theory yields several informative artifacts -

<http://www.google.com/search?q=PID+control+theory>

An article from Embedded Systems: [PID without a PhD](#)