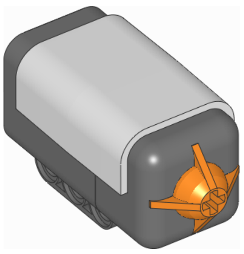
`nxtdevices` – NXT Devices

Use LEGO® MINDSTORMS® NXT motors and sensors with the EV3 brick.

NXT Motor

This motor works just like a LEGO MINDSTORMS EV3 Large Motor. You can use it in your programs using the `Motor` class.

NXT Touch Sensor



class **`TouchSensor`**(*port*)

LEGO® MINDSTORMS® NXT Touch Sensor.

Parameters: **port** (*Port*) – Port to which the sensor is connected.

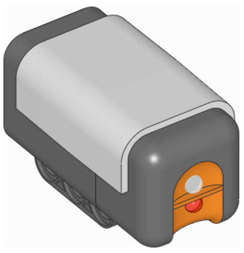
`pressed`()

Checks if the sensor is pressed.

Returns: `True` if the sensor is pressed, `False` if it is not pressed.

Return type: bool

NXT Light Sensor



class **`LightSensor`**(*port*)

LEGO® MINDSTORMS® NXT Color Sensor.

Parameters: **port** (*Port*) – Port to which the sensor is connected.

`ambient()`

Measures the ambient light intensity.

Returns: Ambient light intensity, ranging from 0 (dark) to 100 (bright).

Return type: percentage: %

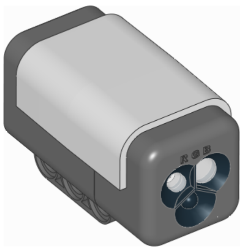
`reflection()`

Measures the reflection of a surface using a red light.

Returns: Reflection, ranging from 0 (no reflection) to 100 (high reflection).

Return type: percentage: %

NXT Color Sensor



class `ColorSensor`(*port*)

LEGO® MINDSTORMS® NXT Color Sensor.

Parameters: **port** (*Port*) – Port to which the sensor is connected.

`color()`

Measures the color of a surface.

Returns: `Color.BLACK`, `Color.BLUE`, `Color.GREEN`, `Color.YELLOW`, `Color.RED`, `Color.WHITE` or `None`.

Return type: `Color`, or `None` if no color is detected.

`ambient()`

Measures the ambient light intensity.

Returns: Ambient light intensity, ranging from 0 (dark) to 100 (bright).

Return type: percentage: %

`reflection()`

Measures the reflection of a surface.

Returns: Reflection, ranging from 0 (no reflection) to 100 (high reflection).

Return type: percentage: %

`rgb()`

Measures the reflection of a surface using a red, green, and then a blue light.

Returns: Tuple of reflections for red, green, and blue light, each ranging from 0.0 (no reflection) to 100.0 (high reflection).

Return type: (percentage: %, percentage: %, percentage: %)

Built-in light

This sensor has a built-in light. You can make it red, green, blue, or turn it off.

`light.on`(*color*)

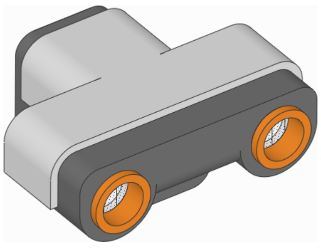
Turns on the light at the specified color.

Parameters: **color** (*Color*) – Color of the light. The light turns off if you choose `None` or a color that is not available.

`light.off()`

Turns off the light.

NXT Ultrasonic Sensor



class `UltrasonicSensor`(*port*)

LEGO® MINDSTORMS® NXT Ultrasonic Sensor.

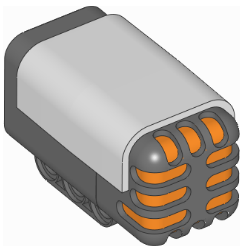
Parameters: **port** (*Port*) – Port to which the sensor is connected.

`distance()`

Measures the distance between the sensor and an object using ultrasonic sound waves.

Returns:	Distance.
Return type:	distance: mm

NXT Sound Sensor



class `SoundSensor`(*port*)

LEGO® MINDSTORMS® NXT Sound Sensor.

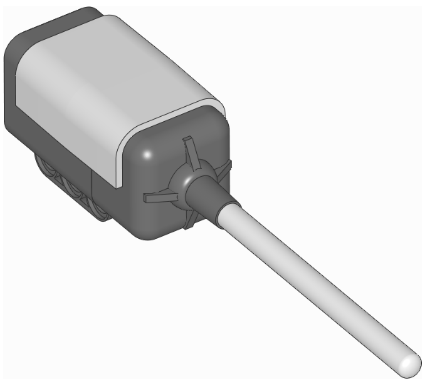
Parameters:	**port** (*Port*) – Port to which the sensor is connected.

`intensity`(*audible_only=True*)

Measures the ambient sound intensity (loudness).

Parameters:	**audible_only** (*bool*) – Detect only audible sounds. This tries to filter out frequencies that cannot be heard by the human ear.
Returns:	Sound intensity.
Return type:	percentage: %

NXT Temperature Sensor



class `TemperatureSensor`(*port*)

LEGO® MINDSTORMS® NXT Temperature Sensor.

Parameters: **port** (*Port*) – Port to which the sensor is connected.

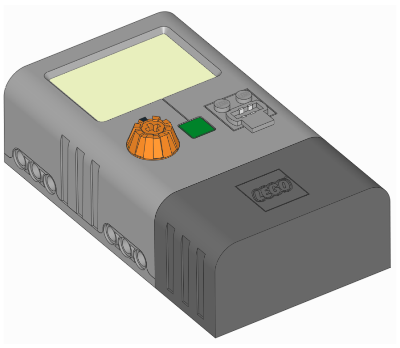
temperature()

Measures the temperature.

Returns: Measured temperature.

Return type: temperature: °C

NXT Energy Meter



class **EnergyMeter**(*port*)

LEGO® MINDSTORMS® Education NXT Energy Meter.

Parameters: **port** (*Port*) – Port to which the sensor is connected.

storage()

Gets the total available energy stored in the battery.

Returns: Remaining stored energy.

Return type: energy: J

input()

Measures the electrical signals at the input (bottom) side of the energy meter. It measures the voltage applied to it and the current passing through it. The product of these two values is power. This power value is the rate at which the stored energy increases. This power is supplied by an energy source such as the provided solar panel or an externally driven motor.

Returns: Voltage, current, and power measured at the input port.

Return type: (voltage: mV, current: mA, power: mW)

output()

Measures the electrical signals at the output (top) side of the energy meter. It measures the voltage applied to the external load and the current passing to it. The product of these two values is power. This power value is the rate at which the stored energy decreases. This power is consumed by the load, such as a light or a motor.

Returns: Voltage, current, and power measured at the output port.

Return type: (voltage: mV, current: mA, power: mW)

Vernier Adapter

class `VernierAdapter`(*port*, *conversion=None*)

LEGO® MINDSTORMS® Education NXT/EV3 Adapter for Vernier Sensors.

Parameters:

- **port** (*Port*) – Port to which the sensor is connected.
- **conversion** (*callable*) – Function of the format `conversion()`. This function is used to convert the raw analog voltage to the sensor-specific output value. Each Vernier Sensor has its own conversion function. The example given below demonstrates the conversion for the Surface Temperature Sensor.

`voltage`()

Measures the raw analog sensor voltage.

Returns: Analog voltage.

Return type: voltage: mV

`conversion`(*voltage*)

Converts the raw voltage (mV) to a sensor value.

If you did not provide a `conversion` function earlier, no conversion will be applied.

Parameters: **voltage** (voltage: mV) – Analog sensor voltage

Returns: Converted sensor value.

Return type: float

`value`()

Measures the sensor `voltage()` and then applies your `conversion()` to give you the sensor value.

Returns: Converted sensor value.

Return type: float

Show/hide example ▲

Example: Using the Surface Temperature Sensor.

```
#!/usr/bin/env pybricks-micropython
from pybricks.parameters import Port
from pybricks.nxtdevices import VernierAdapter

from math import log


# Conversion formula for Surface Temperature Sensor
def convert_raw_to_temperature(voltage):

    # Convert the raw voltage to the NTC resistance
    # according to the Vernier Adapter EV3 block.
    counts = voltage/5000*4096
    ntc = 15000*(counts)/(4130-counts)

    # Handle log(0) safely: make sure that ntc value is positive.
    if ntc <= 0:
        ntc = 1

    # Apply Steinhart-Hart equation as given in the sensor documentation.
    K0 = 1.02119e-3
    K1 = 2.22468e-4
    K2 = 1.33342e-7
    return 1/(K0 + K1*log(ntc) + K2*log(ntc)**3)


# Initialize the adapter on port 1
thermometer = VernierAdapter(Port.S1, convert_raw_to_temperature)

# Get the measured value and print it
temp = thermometer.value()
print(temp)
```