RELAZIONE TECNICA LEGO Mindstorms NXT

ITTS VITO VOLTERRA a.s. 2015/2016

Niccolò Rorato

3H 31/01/2016

RELAZIONE TECNICA

1. Introduzione

In questa relazione vengono trattati dal punto di vista elettrico, meccanico e applicativo i principali componenti riferiti al sistema LEGO Mindstorms NXT.

Viene analizzato, attraverso un'approfondita analisi, l'aspetto di ogni componente e i limiti a cui è soggetto.

I dati ricavati dalle misurazioni vengono riportati su tabelle e/o grafici con opportune descrizioni e/o considerazioni.

In particolare la relazione è sviluppata relativamente a:

- Sensore di contatto NXT;
- Sensore di luminosità NXT;
- Sensore ad ultrasuoni NXT;
- Servomotori interattivi NXT.

2. Strumenti Utilizzati

La strumentazione utilizzata per la raccolta dei dati consiste in:

STRUMENTO	TIPO
PC	PC_110
KIT DI SVILUPPO	Kit LEGO Mindstorms NXT
AMBIENTE DI SVILUPPO	BricxCC

3. Ambiente di Sviluppo

Il software utilizzato per la programmazione è l'editor: BricxCC o Bricx Command Center (sito). Il software BricxCC permette la scrittura dei programmi, la loro compilazione, la ricerca degli errori ed il trasferimento del codice al robot.

Può gestire programmi scritti in vari linguaggi, ma quello utilizzato per questa relazione è il linguaggio di programmazione NXC (tutorial e manuale completo).

4. Unità di Controllo

Il kit LEGO Minsdstorms NXT fornisce diversi sensori e componenti tra cui il brick intelligente (essenziale per il controllo e la gestione di tutti i componenti).

Il brick è composto da 4 porte di **input** (1, 2, 3, 4) e 3 porte di **output** (A, B, C).

È inoltre provvisto di una porta USB 2.0 per scaricare e caricare informazioni dal computer al brick e viceversa (es. codice del programma, datalog files).

I connettori utilizzati, sia per la comunicazione che per l'alimentazione, sono simili ma **incompatibili** ai comuni **RJ11** (cosa voluta dalla LEGO per questioni di sicurezza)**;**

Le specifiche del brick NXT sono le seguenti:

- Processore principale:

Atmel® 32-bit ARM® processor, AT91SAM7S256
256 KB FLASH
64 KB RAM
48 MHz

- Coprocessore:

Atmel® 8-bit AVR processor, ATmega48
4 KB FLASH
512 Byte RAM
8 MHz

- Comunicazione bluetooth:

CSR BlueCore™ 4 v2.0 +EDR System
Supporting the Serial Port Profile (SPP)
Internal 47 KByte RAM
External 8 MBit FLASH
26 MHz

- Comunicazione USB 2.0:

Velocità massima della porta (12 Mbit/s)

- 4 porte di input:

6-wire interface supporting both digital and analog interface

1 high speed port, IEC 61158 Type 4/EN 50170 compliant
- 3 porte di output:

6-wire interface supporting input from encoders
- Display:

100 x 64 pixel LCD nero e bianco
View area: 26 X 40.6 mm
- Speaker:

Sound output channel with 8-bit resolution
Supporting a sample rate of 2-16 KHz
- Tastiera:

4 tasti in gomma
- Alimentazione:

Rechargeable Lithium-Ion battery 1400 mAH is available
- Connettori:

6-wire industry-standard connector, RJ12 Right side adjustment

5. Porte di INPUT

Come già detto in precedenza, vi sono 4 porte di input (1, 2, 3, 4) che permettono all'NXT di comunicare con i sensori.

L'interfaccia di ogni porta (input) è costituita da 6 fili, ognuno dei quali ha una funzione ben precisa. Questo consente di avere un'interfaccia analogica e una digitale all'interno di ogni porta.

É presente anche una resistenza (R49) per evitare cortocircuiti in caso di errato collegamento ad una porta di output dell'NXT.

La figura 1 mostra lo schema della porta 1 dell'NXT (uguale per le porte 2, 3, 4).

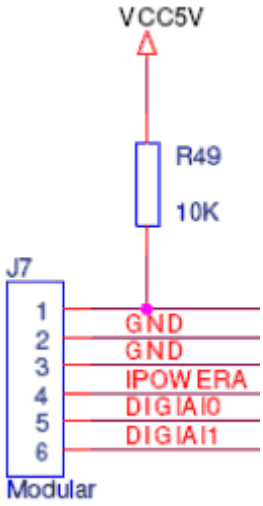


Figura 1: Schema Porta INPUT

PIN	Colore	Nome	Descrizione
1	Bianco	AN	Può essere utilizzato in due modi: fonte di alimentazione da 9 V o ingresso analogico (assume valori da 0 a 1023)
2	Nero	GND	Sono PIN di terra, vengono collegati insieme nell'NXT.
3	Rosso	GND	
4	Verde	4,3 V Power	É l'alimentazione principale per i sensori NXT, fornisce 4,3 V in uscita.
5	Giallo	DIGI0	Sono PIN digitali, collegati direttamente al microprocessore dell'NXT (ARM7), forniscono 3,3 V. Sono utilizzati per le comunicazioni grazie al protocollo i2c.
6	Blu	DIGI1	

La frequenza di campionamento usata per tutti i sensori analogici è di 333 Hz.

6. Porte di OUTPUT

Come già detto in precedenza vi sono 3 porte di OUTPUT (A, B, C) che permettono all'NXT di comunicare con i motori.

L'interfaccia di ogni porta (output) è costituita da 6 fili, ognuno dei quali ha una funzione ben precisa.

La figura 1 mostra lo schema della porta 1 dell'NXT.

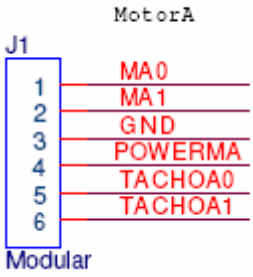


Figura 2: Schema Porta OUTPUT

PIN	Colore	Nome	Descrizione
1	Bianco	MA0	Segnale di uscita PWM
2	Nero	MA1	Segnale di uscita PWM
3	Rosso	GND	PIN di terra relativo all'alimentazione principale
4	Verde	POWERMA	É l'alimentazione principale, fornisce 4,3 V in uscita.
5	Giallo	TACHOA0	Sono PIN digitali, collegati direttamente al microprocessore dell'NXT (ARM7), forniscono 3,3 V. Sono utilizzati per le comunicazioni grazie al protocollo i2c.
6	Blu	TACHOA1	

7. Sensori

I sensori forniti con il kit appartengono a tre categorie:

a. Analogici attivi

Per garantire la compatibilità con i sensori sviluppati per il brick RCX; nel brick NXT è presente un generatore di corrente necessario a produrre la potenza e gli intervalli di misurazione corretti a questi sensori.

b. Analogici passivi

Tutti i sensori che non necessitano di speciali potenze e/o frequenze di campionamento, sopra citate, sono chiamati sensori passivi. Questi sensori vengono campionati ogni 3mS.

I seguenti sensori sono analogici passivi:

- Sensore di contatto;
- Sensore di luce;
- Sensore di sonoro;
- Sensore di temperatura.

c. Digitali

Tutti i sensori che usano la comunicazione I^2C sono chiamati sensori digitali, perché contengono un microcontrollore che gestisce il campionamento dell'ambiente circostante.

I seguente sensore è digitale:

- Sensore ad ultrasuoni.

SENSORE DI CONTATTO

1. Utilizzo

Questo sensore permette di rilevare la pressione e il rilascio del pulsante.

2. Specifiche

Elemento	Touch Sensor
Numero ITEM	9843
Lunghezza (cm)	4,3
Larghezza (cm)	2,3
Altezza (cm)	3,3
Peso (g)	12

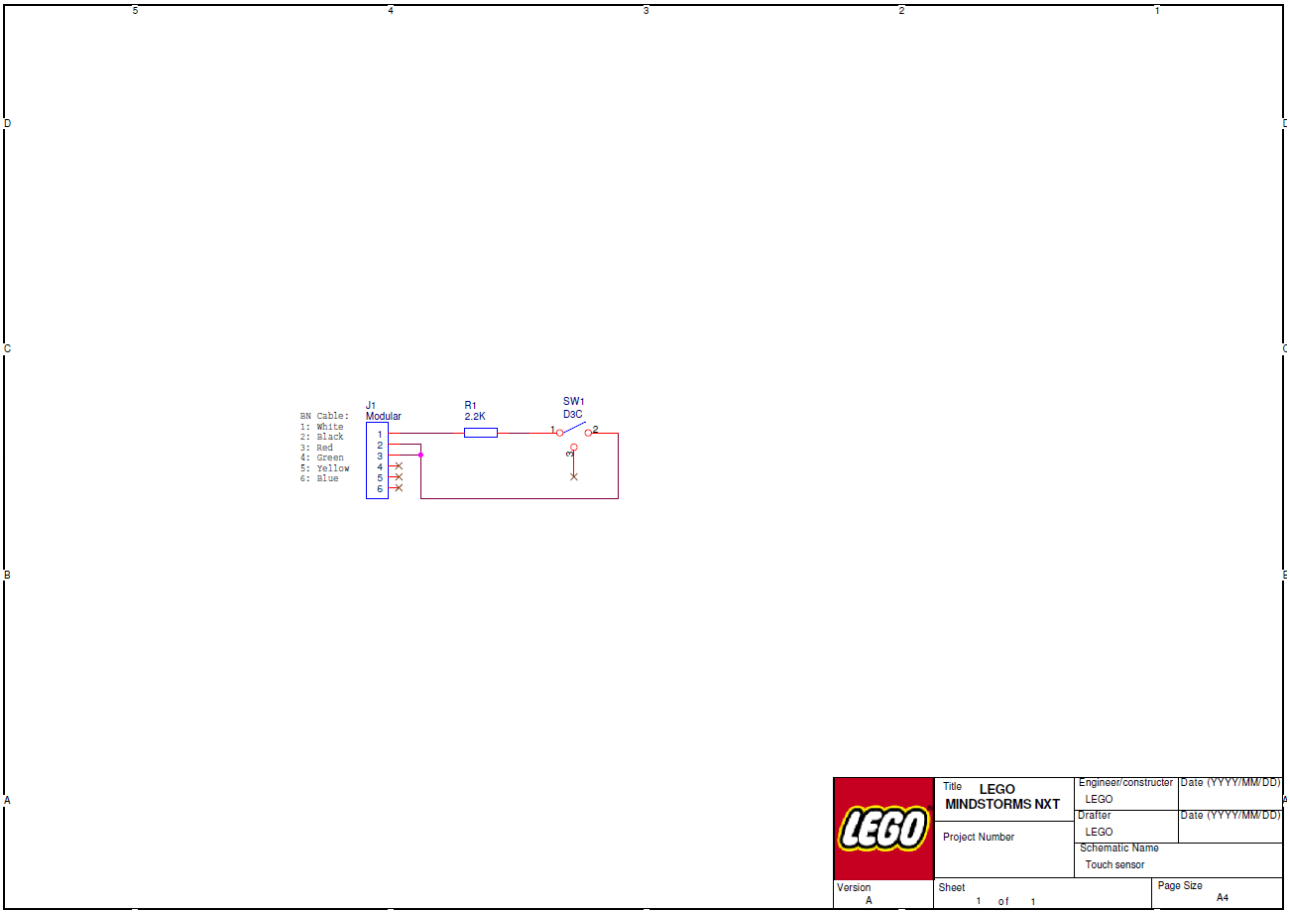
3. Hardware

Il sensore è di tipo passivo (vedi sensori) e viene collegato ad una delle 4 porte di input dell'NXT.

Il sensore è provvisto di una porta per l'alimentazione e la trasmissione dei dati. Questa viene collegata direttamente all'NXT.

PIN	Nome	Utilizzato
1	AN	SI
2	GND	SI
3	GND	SI
4	4,3 V Power	NO
5	DIGI0	NO
6	DIGI1	NO

Di seguito lo schema elettrico del sensore di contatto:



4. Funzionamento

Il sensore percepisce la pressione e il rilascio del pulsante fornendo il valore di corrente che viene poi elaborato dall'NXT, restituendo 1 o 0.

5. Software (SENSORE DI TOCCO)

a. Configurazione di un sensore

SetSensor(const byte & port, const unsigned int config);

PARAMETRI

• Port indica la porta. Vedi costanti delle porte di ingresso.

• Config indica la costante di configurazione contenente il tipo e la modalità. Vedi costanti di modo e tipo del sensore.

ESEMPI

SetSensor(S1, SENSOR_TOUCH);

Imposta nell'ingresso S1 che è presente un sensore di tocco.

b. Configurazione del sensore di tocco

SetSensorTouch (const byte & port);

PARAMETRI

• Port indica la porta. Vedi costanti delle porte di ingresso.

ESEMPI

SetSensorTouch(IN_1);

Imposta nell'ingresso IN_1 del NXT che è presente un sensore di tocco.

c. Impostare la modalità di un sensore

SetSensorMode (const byte & port, byte mode);

PARAMETRI

• Port indica la porta. Vedi costanti delle porte di ingresso.

• mode indica la modalità del sensore. Vedi costanti modalità del sensore.

ESEMPI

SetSensorMode(IN_1, SENSOR_MODE_PULSE);

Imposta la modalità SENSOR_MODE_PULSE nell'ingresso IN_1. A ogni pressione e conseguente rilascio si produrrà un cambiamento che viene conteggiato una sola volta.

d. Impostare la tipologia del sensore

SetSensorType (const byte & port, byte type)

PARAMETRI

• Port indica la porta. Vedi costanti delle porte di ingresso.

• mode indica la tipologia del sensore. Vedi costanti tipologie del sensore.

ESEMPI

SetSensorType(IN_1, SENSOR_TYPE_TOUCH);

Imposta la tipologia SENSOR_TYPE_TOUCH nell'ingresso IN_1.

e. Resettare il sensore

ResetSensor (const byte & port)

PARAMETRI

• Port indica la porta del sensore. Vedi costanti delle porte di ingresso.

ESEMPI

ResetSensor(IN_1);

Imposta il sensore nell'ingresso IN_1 con nuovi parametri. Questa funzione deve essere chiamata dopo che la modalità o la tipologia del sensore vengono cambiate.

f. Leggere valore del sensore

Sensor(const byte & port)

PARAMETRI

• Port indica la porta del sensore. Vedi costanti delle porte di ingresso.

ESEMPI

Sensor(S1);

Comunica al sensore di leggere il valore nell'ingresso S1.

g. Cancella il valore del sensore

ClearSensor (const byte & port)

PARAMETRI

• Port indica la porta del sensore. Vedi costanti delle porte di ingresso.

ESEMPI

ClearSensor(IN_1);

Cancella il valore del sensore nell'ingresso IN_1; ha effetto solo con i sensori che sono impostati per misurare un valore cumulativo, come una rotazione o un conteggio di impulsi.

6. Applicazioni

RILEVAZIONE TOCCO

Viene mostrato sul display dell'NXT il valore 1 se il pulsante viene premuto, 0 se il pulsante non viene premuto.

Codice utilizzato:

```
task main() {
    SetSensorTouch(IN_1); //Imposta nell'ingresso IN_1 che è presente un sensore di tocco
    while(TRUE) { //Ciclo che viene ripetuto per sempre
        NumOut(50,32,Sensor(IN_1)); //Scrive a schermo alle coordinate indicate se il pulsante viene premuto(1) o meno (0)
    }
}
```

RILEVAZIONE TOCCO v2

Viene mostrato sul display dell'NXT il messaggio "Tocco rilevato" 1 se il pulsante viene premuto, "Nessun tocco" se il pulsante non viene premuto.

Codice utilizzato:

```
task main() {
    SetSensorTouch(IN_1); //Imposta nell'ingresso IN_1 che è presente un sensore di tocco
    while (true) { //Ciclo che viene ripetuto per sempre
        if (SENSOR_1 == 1) { //Se il pulsante viene premuto scrive a schermo alle coordinate indicate il messaggio "Tocco rilevato"
            TextOut(0,32,"Tocco rilevato");
        }
        else { //Se non viene premuto il pulsante scrive a schermo alle coordinate indicate il messaggio "Nessun tocco"
            TextOut(0,32,"Nessun tocco");
        }
        Wait(100); //Attesa di 100ms prima di pulire lo schermo
        ClearScreen();
    }
}
```

RILEVAZIONE TOCCO v3

Viene mostrato sul display dell'NXT quante volte viene premuto il pulsante.

Codice utilizzato:

```
task main() {
    SetSensorTouch(IN_1);
    SetSensorMode(IN_1,SENSOR_MODE_PULSE); //Impostata la modalità PULSE
    while(TRUE){
        NumOut(50,32,Sensor(IN_1)); //Scrive a schermo alle coordinate indicate quante volte viene premuto il pulsante
    }
}
```

RILEVAZIONE TOCCO v4

Viene mostrato sul display dell'NXT quante volte viene premuto e rilasciato il pulsante

Codice utilizzato:

```
task main() {
    SetSensorTouch(IN_1);
    SetSensorMode(IN_1,SENSOR_MODE_EDGE);
    while(TRUE){
        NumOut(50,32,Sensor(IN_1)); //Scrive a schermo alle coordinate indicate quante volte viene premuto e rilasciato il pulsante
    }
}
```

SENSORE DI LUCE

1. Utilizzo

Questo sensore permette di rilevare la luce riflessa da una superficie e può essere regolato in modo da misurare la luminosità ambientale circostante.

2. Specifiche

Elemento	Light Sensor
Numero ITEM	9844
Lunghezza (cm)	4,3
Larghezza (cm)	2,3
Altezza (cm)	3,3
Peso (g)	15

3. Hardware

Il sensore è di tipo attivo (vedi sensori) e viene collegato a una delle 4 porte di input dell'NXT. Il segnale fornito è di tipo analogico.

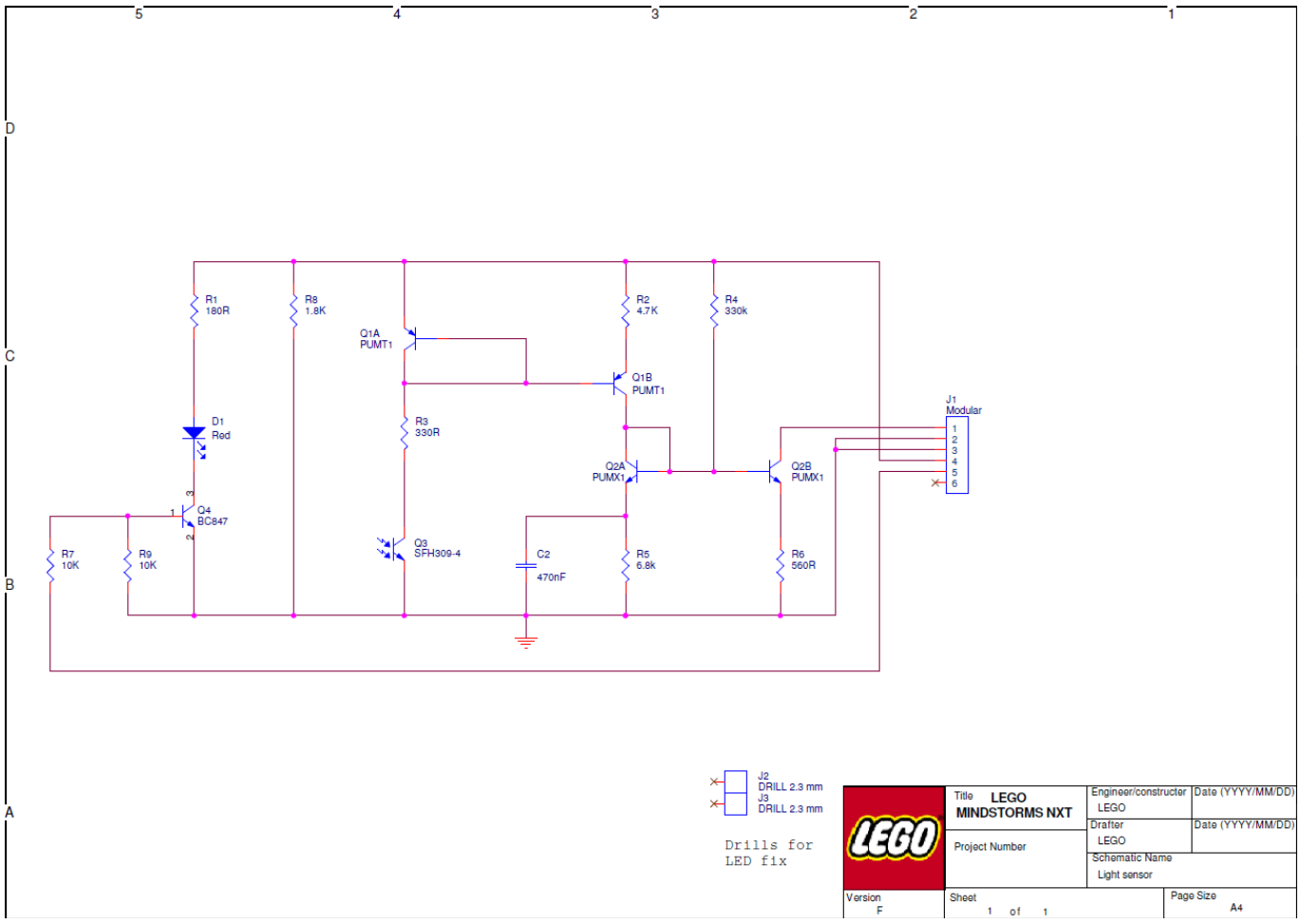
Il sensore è costituito da due componenti:

- 1 LED (diodo a emissione luminosa rossa);
- 1 fotoresistenza, questa è molto più sensibile ai colori infrarossi della luce, rispetto allo stretto spettro visibile che vediamo.

Il sensore è provvisto di una porta per l'alimentazione e la ricezione / trasmissione di dati. Questa viene collegata direttamente all'NXT. Il PIN DIGI0 viene utilizzato per controllare lo stato del diodo LED.

PIN	Nome	Utilizzato
1	AN	SI
2	GND	SI
3	GND	SI
4	4,3 V Power	SI
5	DIGI0	SI
6	DIGI1	NO

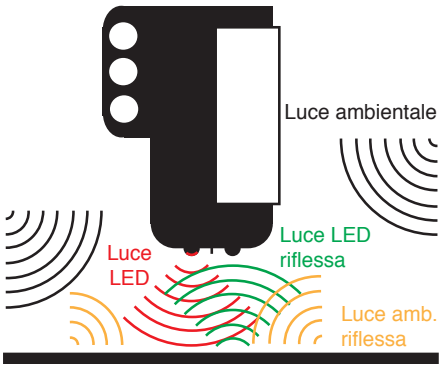
Di seguito lo schema elettrico del sensore di luce:



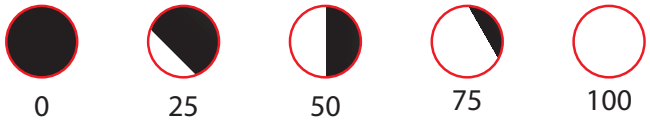
4. Funzionamento

Il sensore può lavorare in due modalità:

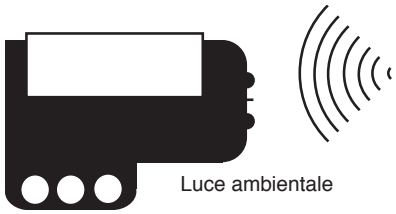
1. In **modalità di luce riflessa,** il LED è acceso (di default) ed emette una luce rossa che viene riflessa dalla superficie incontrata; in base alla quantità di luce riflessa ricevuta dalla fotoresistenza, viene restituito un valore da 0 a 100 (di default).



Cosa legge il sensore di luce in modalità di luce riflessa



2. In **modalità di luce ambientale**, il LED è spento. La fotoresistenza riceve solamente la luce ambientale. In base alla quantità di luce ambientale ricevuta dalla fotoresistenza, viene restituito un valore in base da 0 a 100 (di default).



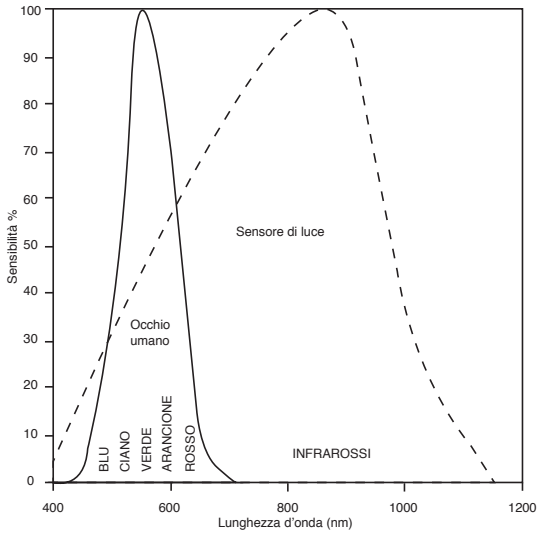
Cosa vede il sensore di luce in modalità di luce ambientale rispetto all'occhio umano

Vista dell'occhio umano

Vista tramite il sensore di luce

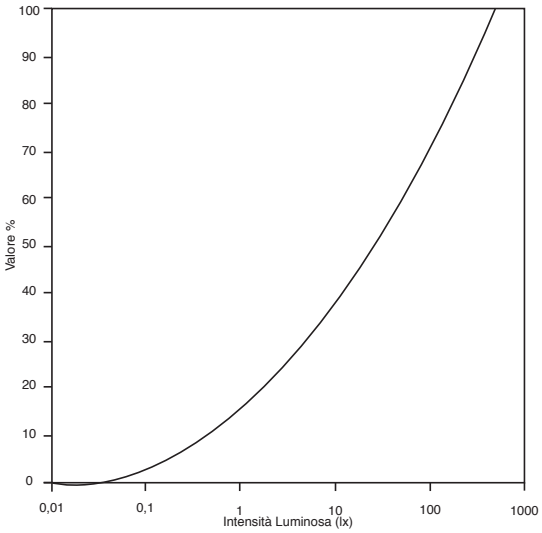
5. Curve delle caratteristiche del sensore

Di seguito viene rappresentato un grafico relativo alla sensibilità dell'occhio umano e del sensore di luce a differenti lunghezza d'onda:



Descrizione
Il grafico è composto da due curve: • quella rappresentata con una linea continua indica la luce percepita dall'occhio umano; • quella rappresentata con una linea tratteggiata rappresenta invece la luce percepita del sensore. Il grafico evidenzia le sensibilità dell'occhio umano e del sensore a determinate lunghezze d'onda. L'occhio umano è molto più sensibile alla luce visibile rispetto al sensore di luce, ma il sensore di luce risulta molto più sensibile alla luce infrarossa e per questo mostra un picco di sensibilità alle frequenze infrarosse.

Di seguito grafico relativo al valore restituito in base all'intensità luminosa:

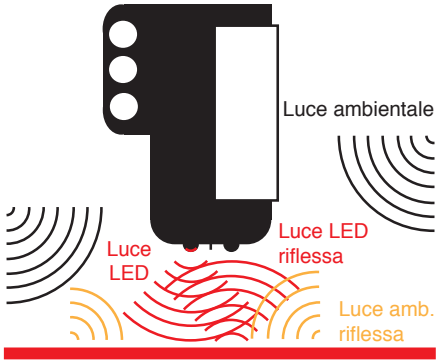


Descrizione
È possibile dedurre che il segnale restituito dal sensore è direttamente proporzionale all'intensità luminosa.

6. Considerazioni

In modalità di luce riflessa il LED rosso è attivo. La fotoresistenza non rileva solo il colore rosso, ma è in grado di rilevarne qualsiasi altro, pur non individuando quale esso sia.

Nel caso in cui la superficie che il sensore incontra sia rossa la fotoresistenza restituisce un valore più alto rispetto a quando il sensore incontra una superficie di qualunque altro colore diverso dal rosso (verde, viola etc.); ciò avviene perché la luce rossa emessa dal LED viene riflessa totalmente dalla superficie rossa rispetto a quando questa sia di un altro colore.

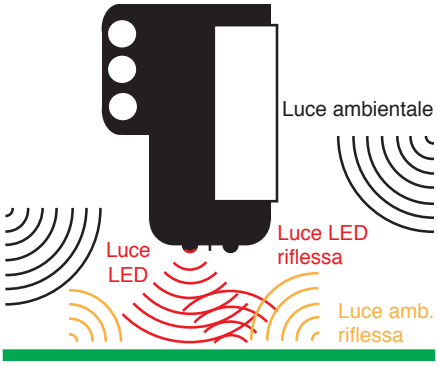


La luce rossa viene riflessa completamente dalla superficie anch'essa rossa.

La superficie assorbe quindi tutti i colori ad eccezione del rosso (vedi assorbimento della luce da parte di oggetti).

La luce rossa viene riflessa solo parzialmente dalla superficie.

La superficie assorbe tutti i colori ad eccezione del verde (vedi assorbimento della luce da parte di oggetti).



Di seguito sono riportati i dati ricavati:

Luce ambientale (%)	Luce riflessa (%)	Angolazione (°)	Altezza sensore (mm)	Superficie
24	64	90	8	Bianca Opaca
24	61	90	8	Rossa Opaca
24	47	90	8	Verde Opaca

Di seguito è riportato il codice (vedi software per i comandi) utilizzato:

Codice utilizzato:

```
task main() {
    SetSensorLight(IN_1); //Imposta nell'ingresso IN_1 che è presente un sensore di luce
    while(TRUE){ //Ciclo ripetuto all'infinito
        SetSensorType(IN_1,IN_TYPE_LIGHT_INACTIVE); //Imposta la tipologia col LED non attivo
        ResetSensor(IN_1); //Imposta il sensore nell'ingresso IN_1 con i nuovi parametri sopra indicati
        Wait(500); //Attesa di 500ms prima di scrivere a schermo alle coord. specificate il valore ricevuto dal sensore
        NumOut(50,42,Sensor(IN_1));
        SetSensorType(IN_1,IN_TYPE_LIGHT_ACTIVE); //Imposta la tipologia col LED non attivo
        ResetSensor(IN_1); //Imposta il sensore nell'ingresso IN_1 con i nuovi parametri sopra indicati
        Wait(500); //Attesa di 500ms prima di scrivere a schermo alle coord. specificate il valore ricevuto dal sensore
        NumOut(50,32,Sensor(IN_1));
    }
}
```

N.B. : Sfruttando questa caratteristica mediante l'utilizzo di tre LED specificatamente di colore rosso, blu e verde (RGB) e data un'opportuna sequenza, il sensore potrà riconoscere di quale colore è la superficie.

7. Limiti e problemi riscontrati

Il sensore di luce può essere influenzato in maniera più o meno rilevante da vari fattori qui di seguito individuati:

a. ANGOLAZIONE

Per avere una lettura ottimale durante la modalità di luce riflessa il sensore deve essere perpendicolare al terreno. L'inclinazione infatti, influenza la corretta lettura da parte del sensore.
Durante la progettazione di un eventuale robot è bene tener presente questo aspetto.

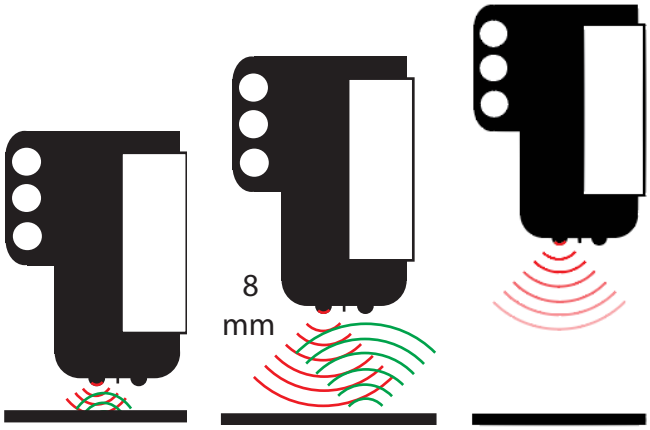
Di seguito sono riportati i valori ricavati:

Luce ambientale (%)	Luce riflessa (%)	Angolazione (°)	Altezza sensore (mm)	Superficie
24	64	90	8	Bianca Opaca
36	51	30	8	Bianca Opaca

Codice utilizzato: vedi codice a pag 14.

b. VICINANZA E O LONTANANZA DALLA SUPERFICIE

Per una migliore accuratezza di lettura, il sensore deve essere tenuto a 8 mm dalla superficie. Diversamente non sarebbe in grado di fornire dati attendibili sulla reale riflessione di una superficie.



Di seguito sono riportati i valori ricavati:

Luce ambientale (%)	Luce riflessa (%)	Angolazione (°)	Altezza sensore (mm)	Superficie
8	71	90	4	Bianca Opaca
24	64	90	8	Bianca Opaca
35	48	90	32	Bianca Opaca

Codice utilizzato: vedi codice a pag 14.

c. LUCE AMBIENTALE

Anche la luce ambientale in modalità di luce riflessa influenza la lettura del sensore, poiché altera in modo consistente l'effettivo valore della lettura. Soprattutto lunghezze d'onda superiori a 590 nm influenzano la lettura del sensore.

Possibile soluzione a tale problema:

- Applicare un materiale opaco attorno ai sensori.

d. LUCIDITÀ OD OPACITÀ DELLA SUPERFICIE

A seconda che la superficie si lucida od opaca il sensore restituisce un valore maggiore nel caso in cui essa sia lucida, minore nel caso sia opaca.

Di seguito sono riportati i valori ricavati:

Luce ambientale (%)	Luce riflessa (%)	Angolazione (°)	Altezza sensore (mm)	Superficie
4	34	90	8	Nera Opaca
4	37	90	8	Nera Lucida

Codice utilizzato: vedi codice a pag 14.

8. Software (SENSORE DI LUCE)

a. Configurazione di un sensore

SetSensor (const byte & port, const unsigned int config);

PARAMETRI

• Port indica la porta. Vedi costanti delle porte di ingresso.

• Config indica la costante di configurazione contenente il tipo e la modalità. Vedi costanti di modo e tipo del sensore.

ESEMPI

SetSensor(S1, SENSOR_LIGHT);

Imposta nell'ingresso S1 che è presente un sensore di luce.

Per default il LED è acceso. Il sensore ritorna un valore in percentuale (0 a 100).

b. Configurazione di un sensore di luce

SetSensorLight (const byte & port, bool bActive = true);

PARAMETRI

• Port indica la porta. Vedi costanti delle porte di ingresso.

• bActive indica attraverso un valore BOOL se il LED del sensore è attivo o inattivo. Di default è attivo.

ESEMPI

SetSensorLight(IN_1);

Imposta nell'ingresso IN_1 che è presente un sensore di luce.

Per default il LED è acceso. Il sensore ritorna un valore in percentuale (0 a 100).

c. Impostare la modalità di un sensore

SetSensorMode (const byte & port, byte mode);

PARAMETRI

• Port indica la porta. Vedi costanti delle porte di ingresso.

• mode indica la modalità del sensore. Vedi costanti modalità del sensore.

ESEMPI

SetSensorMode(IN_1, SENSOR_MODE_RAW);

Imposta la modalità SENSOR_MODE_RAW nell'ingresso IN_1, con la quale il sensore restituisce il valore.

d. Impostare la tipologia del sensore

SetSensorType (const byte & port, byte type)

PARAMETRI

• Port indica la porta. Vedi costanti delle porte di ingresso.

• type indica la tipologia del sensore. Vedi costanti tipologie del sensore.

ESEMPI

SetSensorType(IN_1, SENSOR_TYPE_LIGHT_INACTIVE);

Imposta la tipologia SENSOR_TYPE_LIGHT_INACTIVE nell'ingresso IN_1, con la quale viene controllato lo stato del LED.

e. Resettare il sensore

ResetSensor (const byte & port)

PARAMETRI

• Port indica la porta del sensore. Vedi costanti delle porte di ingresso.

ESEMPI

ResetSensor(IN_1);

Imposta il sensore con nuovi parametri. Questa funzione deve essere chiamata dopo che la modalità o la tipologia del sensore vengono cambiate.

f. Leggere valore del sensore

Sensor(const byte & port)

PARAMETRI

• Port indica la porta del sensore. Vedi costanti delle porte di ingresso.

ESEMPI

Sensor(S1);

Comunica al sensore di leggere il valore nell'ingresso S1.

9. Applicazioni

LUMINOSITA' AMBIENTALE

Viene mostrato sul display dell'NXT un valore da 0 (poca luce) a 100 (molta luce) che indica la luminosità ambientale.

Codice utilizzato:

```
task main() {
    SetSensorLight(IN_1); //Imposta nell'ingresso IN_1 che è presente un sensore di luce
    SetSensorType(IN_1,IN_TYPE_LIGHT_INACTIVE); //Imposta la tipologia col LED non attivo
    while (TRUE) { //Ciclo ripetuto all'infinito
        NumOut(50,32,Sensor(IN_1)); //Scrive a schermo alle coordinate indicate il valore letto dal sensore di luce
        Wait(100); //Attesa di 100ms prima di pulire lo schermo
        ClearScreen();
    }
}
```

LUMINOSITÀ RIFLESSA DA UNA SUPERFICIE

Viene mostrato sul display dell'NXT un valore da 0 (poca luce) a 100 (molta luce) che indica la luce riflessa da una superficie.

Codice utilizzato:

```
task main() {
    SetSensorLight(IN_1); //Imposta nell'ingresso IN_1 che è presente un sensore di luce
    SetSensorType(IN_1,IN_TYPE_LIGHT_ACTIVE); //Imposta la tipologia col LED attivo (possibile ometterlo)
    while (TRUE) {
        NumOut(50,32,Sensor(IN_1)); //Scrive a schermo alle coordinate indicate il valore letto dal sensore di luce
        Wait(100); //Attesa di 100ms prima di pulire lo schermo
        ClearScreen();
    }
}
```

RILEVAZIONE DI UNA SUPERFICIE SCURA

Il robot si muove in avanti finché non incontra una superficie scura.

Codice utilizzato:

```
task main() {
    SetSensorLight(IN_1); //Imposta nell'ingresso IN_1 che è presente un sensore di luce
    SetSensorType(IN_1,IN_TYPE_LIGHT_ACTIVE); //Imposta la tipologia col LED attivo (possibile ometterlo)
    OnFwd(OUT_AC, 75); //Muove in avanti i motori nelle uscite A e C
    if (Sensor(IN_1) < 64) {
        Off(OUT_AC); //Ferma i motori nelle uscite A e C
    }
}
```

SENSORE AD ULTRASUONI

1. Utilizzo

Questo sensore permette di rilevare la distanza degli oggetti circostanti.

2. Specifiche

Elemento	Ultrasonic Sensor
Numero ITEM	9846
Lunghezza (cm)	4,3
Larghezza (cm)	5,5
Altezza (cm)	3,3
Peso (g)	25

3. Hardware

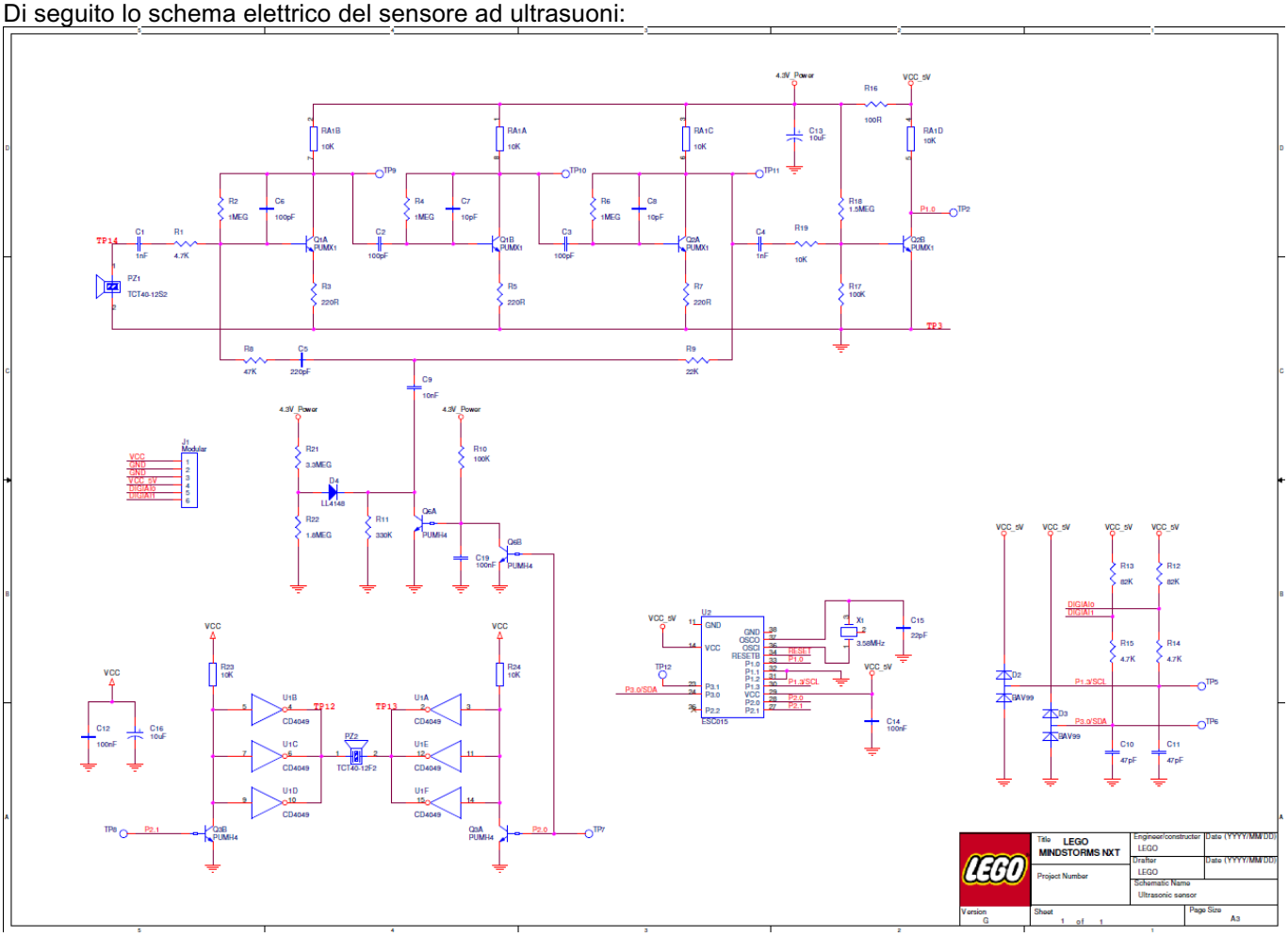
Il sensore è di tipo digitale (vedi sensori) e viene collegato a una delle 4 porte di input dell'NXT.
Il sensore ad ultrasuoni è composto da tre elementi essenziali per il suo funzionamento:

- Trasmettitore (emette un suono ad una frequenza definita, tipicamente intorno ai 40 - 250 KHz);
- Ricevitore (riceve il suono riflesso dagli ostacoli incontrati);
- Chip (utilizzato per il conteggio del tempo impiegato e quindi della distanza percorsa).

Il sensore è provvisto anche di una porta per l'alimentazione e la trasmissione di dati. Questa viene collegata direttamente all'NXT.

PIN	Nome	Utilizzato
1	AN	SI
2	GND	SI
3	GND	SI
4	4,3 V Power	SI
5	DIGI0	SI
6	DIGI1	SI

Di seguito lo schema elettrico del sensore ad ultrasuoni:



4. Funzionamento

Ogni sensore ad ultrasuoni è provvisto di un identificativo digitale. Non è possibile utilizzare più sensori ad ultrasuoni sulla stessa porta.

Il sensore ad ultrasuoni misura la distanza in centimetri e/o in pollici di un oggetto posto davanti ad esso. Può misurare distanze da 0 a 255 centimetri con una precisione di ± 3 cm.

Per il suo funzionamento utilizza lo stesso principio scientifico usato dai pipistrelli: misura la distanza calcolando il tempo impiegato da un'onda sonora a colpire un oggetto e a ritornare.

Il trasmettitore del sensore emette onde sonore (40 a 250 KHz non udibili dall'orecchio umano) e quando queste incontrano un oggetto, vengono riflesse e percepite dal ricevitore. Per permettere queste misurazioni (conteggio del tempo e quindi la distanza percorsa) il sensore ad ultrasuoni utilizza il chip posto al suo interno.

Il sensore è più sensibile a oggetti grandi e che presentano superfici dure, mentre lo è meno con oggetti in tessuto o curvi (come una palla) oppure molto sottili e piccoli.

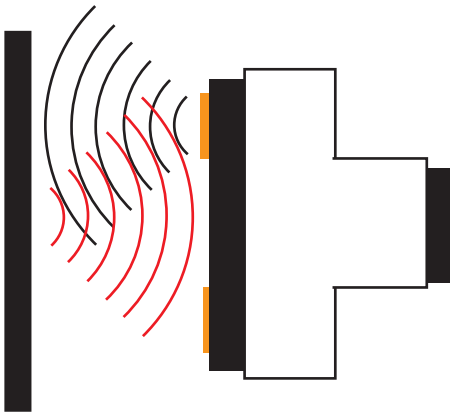
Il sensore possiede un campo visivo asimmetrico di circa 45°.

L'asimmetria è dovuta al fatto che il lato destro emette il fascio di ultrasuoni, mentre il lato sinistro lo riceve.

5. Considerazioni

Di seguito sono riportati i dati ricavati in una **prima misurazione**:

Distanza effettiva (cm)	Distanza restituita (cm)	Oggetto
attaccato al sensore	255	Rigido
<4	4 - 6	Rigido
4	4	Rigido
6	6	Rigido
14	14	Rigido
21	21	Rigido
30	29	Rigido
44	43	Rigido
50	49	Rigido
133	133	Rigido
227	227	Rigido
228	228	Rigido
>228	illeggibile	Rigido



Di seguito è riportato il codice utilizzato:

Codice utilizzato:

```
task main() {
    SetSensorLowspeed(IN_1); //Imposta nell'ingresso IN_1 che è presente un sensore che utilizza protocollo I2C
    while(TRUE){ //Ciclo ripetuto all'infinito
        NumOut(50,32,SensorUS(IN_1)); //Scrive a schermo alle coordinate indicate il valore letto dal sensore ad ultrasuoni
        Wait(50); //Attesa di 50 ms prima di pulire lo schermo
        ClearScreen();
    }
}
```

Il codice utilizzato ha un intervallo di 50ms tra l'emissione di un suono e il successivo, questo per evitare che vi siano interferenze di segnale che possono causare un'errata misura della distanza.

Nella raccolta dati è stato utilizzato come ostacolo un pannello rigido in plexiglass di 30 × 24 cm di grandezza e durante le misurazioni il pannello è sempre stato posto perfettamente perpendicolare al terreno e parallelo al sensore, in modo da evitare riflessi che potrebbero causare dei problemi sull'effettiva distanza.

Salvo casi particolari, il sensore ha sempre restituito dei valori molto precisi sulla reale distanza dell'oggetto.

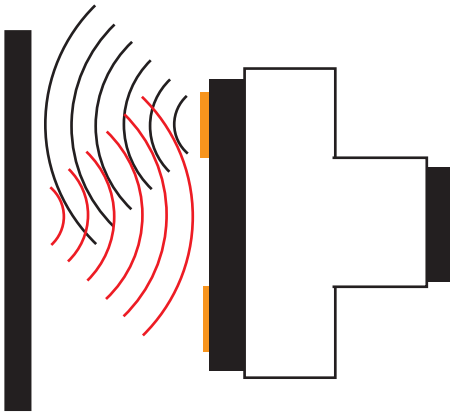
In un primo caso, posto il pannello attaccato al sensore, questo restituisce come valore: 255 non rilevando alcun riflesso.

Il un secondo caso, quando il pannello è posto ad una distanza minore di 4 cm il sensore restituisce come valori 4,5,6. Questo significa che in presenza di oggetti posti a distanze inferiori ai 4 cm il sensore diventa inaffidabile.

In un terzo caso posto l'ostacolo a distanze maggiori di 228 cm, il sensore restituisce valori illeggibili sul display dell'NXT. Questo perché vi è un continuo flusso di dati in elaborazione visualizzati sul display.

Di seguito sono riportati i dati ricavati in una **seconda misurazione**:

Distanza effettiva (cm)	Distanza restituita (cm)	Oggetto
attaccato al sensore	255	Rigido
<4	4 - 6	Rigido
4	4	Rigido
6	6	Rigido
14	14	Rigido
21	21	Rigido
30	29	Rigido
44	43	Rigido
50	49	Rigido
133	133	Rigido
227	227	Rigido
228	228	Rigido
254	254	Rigido
255	255	Rigido
>255	255	Rigido



Di seguito è riportato il codice utilizzato:

Codice utilizzato:

```
task main() {
    SetSensorLowspeed(IN_1); //Imposta nell'ingresso IN_1 che è presente un sensore che utilizza protocollo I2C
    while(TRUE){ //Ciclo ripetuto all'infinito
        NumOut(50,32,SensorUS(IN_1)); //Scrive a schermo alle coordinate indicate il valore letto dal sensore ad ultrasuoni
        Wait(50); //Attesa di 100ms prima di pulire lo schermo
        ClearScreen();
    }
}
```

Il codice utilizzato ha un intervallo di 100ms tra l'emissione di un suono e il successivo. Questo valore è stato aumentato a causa dell'elevata distanza da misurare poiché la precedente frequenza con cui venivano emesse le onde sonore provocava interferenze di segnale e un flusso di dati troppo elevato.

Nella raccolta dati è stato utilizzato come ostacolo un pannello rigido in plexiglass di 30 × 24 cm di grandezza e durante le misurazioni il pannello è sempre stato posto perfettamente perpendicolare al terreno e parallelo al sensore, in modo da evitare riflessi che potrebbero causare dei problemi sull'effettiva distanza.

Salvo casi particolari, il sensore ha sempre restituito dei valori molto precisi sulla reale distanza dell'oggetto.

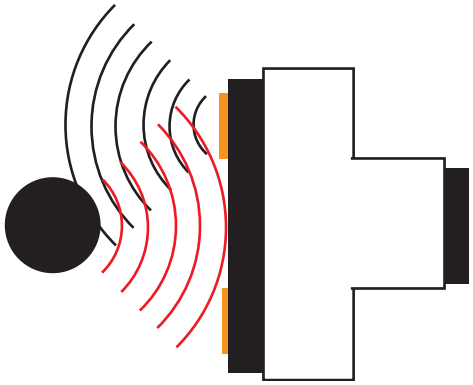
In un primo caso, posto il pannello attaccato al sensore, questo restituisce come valore: 255 non rilevando alcun riflesso.

Il un secondo caso, quando il pannello è posto ad una distanza minore di 4 cm il sensore restituisce come valori 4,5,6. Questo significa che in presenza di oggetti posti a distanze inferiori ai 4 cm il sensore diventa inaffidabile.

In un terzo caso, posto l'ostacolo a distanze maggiori di 255 cm, il sensore restituisce come valore 255 perché non rileva alcun eco.

Di seguito sono riportati i dati ricavati in una **terza misurazione**:

Distanza effettiva (cm)	Distanza restituita (cm)	Oggetto
attaccato al sensore	7	Cilindro
2	24	Cilindro
3	8	Cilindro
6	8	Cilindro
9	25	Cilindro
14	20	Cilindro
23	25	Cilindro
32	35	Cilindro
44	47	Cilindro
53	56	Cilindro
76	81	Cilindro
100	129	Cilindro
118	136	Cilindro
135	135	Cilindro
140	141	Cilindro
>150	255 e altri	Cilindro



Di seguito è riportato il codice utilizzato:

Codice utilizzato:

```
task main() {
    SetSensorLowspeed(IN_1); //Imposta nell'ingresso IN_1 che è presente un sensore che utilizza protocollo I2C
    while(TRUE){ //Ciclo ripetuto all'infinito
        NumOut(50,32,SensorUS(IN_1)); //Scrive a schermo alle coordinate indicate il valore letto dal sensore ad ultrasuoni
        Wait(50); //Attesa di 50ms prima di pulire lo schermo
        ClearScreen();
    }
}
```

Il codice utilizzato ha un intervallo di 100ms tra l'emissione di un suono e il successivo.

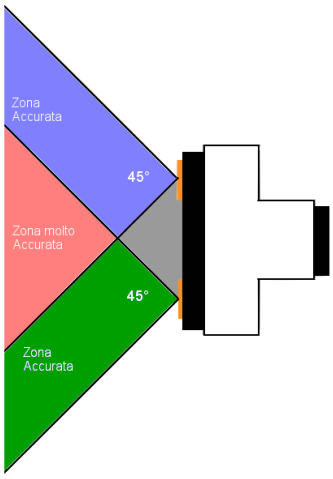
Nella raccolta dati è stato utilizzato come ostacolo un cilindro rigido di 6,4 cm di diametro e durante le misurazioni il cilindro è sempre stato posto perfettamente perpendicolare al terreno e parallelo al sensore, in modo da evitare riflessi che potrebbero causare dei problemi sull'effettiva distanza.

Il sensore ad ultrasuoni ha restituito valori non accurati sull'effettiva distanza del cilindro.

Per distanze maggiori ai 150 cm viene restituito molto spesso il valore 255, ciò significa che il suono non ritorna indietro ma viene riflesso altrove.

Di seguito sono riportati i dati di una **quarta misurazione**:

Distanza effettiva (cm)	Distanza restituita (cm)	Oggetto	Zona
14	14	Pannello	Rossa
14	24	Pannello	Blu
14	26	Pannello	Verde



Dai dati precedentemente ricavati possiamo dedurre che a seconda posizione che l'oggetto occupa rispetto al sensore ad ultrasuoni e quindi della zona (blu, rossa, verde), la distanza risulta più o meno accurata. Inoltre possiamo determinare che l'angolo di visione del sensore è di circa 45°.

6. Limiti e problemi riscontrati

Il sensore ad ultrasuoni può essere influenzato in maniera più o meno rilevante da vari fattori qui di seguito individuati:

a. DISTANZA

Il sensore ad ultrasuoni può rilevare distanze da 0 a 255 centimetri con una precisione di ± 3 cm.
Distanze maggiori di 255 cm ritornano sempre valore 255.

b. FORMA OGGETTI

Come mostra la figura 3 gli errori generati dalla forma degli oggetti devono essere presi in considerazione.

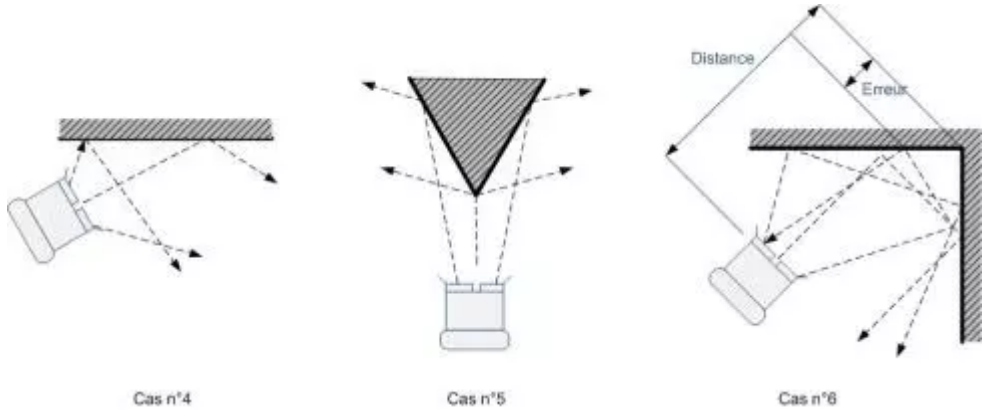


Figura 3: Forma Oggetti

c. CAMPO VISIVO

Il sensore possiede un campo visivo asimmetrico di circa 45°.
L'asimmetria è dovuta al fatto che il lato destro emette il fascio di ultrasuoni, mentre il lato sinistro lo riceve.
Ciò comporta che misurazioni effettuate a destra sono più accurate rispetto alle misurazioni effettuate a sinistra.

d. INTERFERNZE

Il sensore è sensibile a frequenze tipicamente intorno ai 40 - 250 KHz. Quindi è sconveniente avvicinare più sensori ad ultrasuoni.

e. TEMPERATURA

Il sensore ad ultrasuoni è sensibile alla temperatura (da verificare).

f. PRESSIONE

Il sensore ad ultrasuoni è sensibile alla pressione (da verificare).

7. Software (SENSORE AD ULTRASUONI)

a. Configurare sensore

SetSensorLowspeed (const byte & port, bool bIsPowered = true)

PARAMETRI

• Port indica la porta. Vedi costanti delle porte di ingresso.

• bIsPowered indica se per configurare la porta per i dispositivi I_2C è necessaria un'alimentazione. Il valore predefinito per questo parametro opzionale è TRUE.

ESEMPI

SetSensorLowspeed(S1);

Imposta nell'ingresso S1 che è presente un sensore digitale e utilizza il protocollo di comunicazione I_2C.

b. Leggere valore del sensore

SensorUS (const byte port)

PARAMETRI

• Port indica la porta. Vedi costanti delle porte di ingresso.

ESEMPI

SensorUS(IN_1);

Comunica all'NXT di leggere il valore nell'ingresso IN_1.

8. Applicazioni

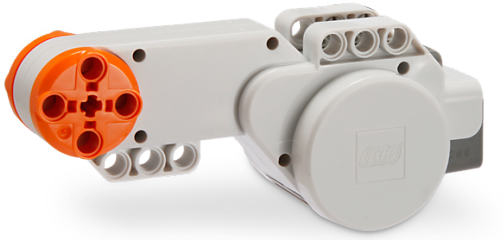
LEGGERE DISTANZA DI UN OGGETTO

Viene mostrato sul display dell'NXT un valore che indica in centimetri la distanza di un oggetto.

Codice utilizzato:

```
task main() {
    SetSensorLowspeed(IN_1); //Imposta nell'ingresso IN_1 che è presente un sensore che utilizza protocollo I2C
    while(TRUE){
        NumOut(50,32,SensorUS(IN_1)); //Scrive a schermo alle coordinate indicate il valore letto dal sensore ad ultrasuoni
        Wait(100); //Attesa di 100ms prima di pulire lo schermo
        ClearScreen();
    }
}
```

SERVOMOTORE INTERATTIVO



1. Utilizzo

Il servomotore permette al robot di svolgere diversi compiti, come muoversi e afferrare oggetti.

2. Specifiche

Elemento	Interactive Servo Motor
Numero ITEM	9842
Lunghezza (cm)	9,4
Larghezza (cm)	7,9
Altezza (cm)	2,5
Peso (g)	80

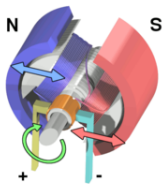
3. Hardware

Il servomotore interattivo è internamente costituito da:
• 1 Motore in corrente continua;
• 1 Encoder;
• Una serie di ingranaggi tra loro in rapporto di riduzione.

a. MOTORE IN CORRENTE CONTINUA

La classica macchina in corrente continua ha una parte che gira detta rotore o armatura, e una parte che genera un campo magnetico fisso (nell'esempio i due magneti colorati) detta statore; Un interruttore rotante, detto commutatore o collettore a spazzole, che inverte due volte ad ogni giro la direzione della corrente elettrica, che percorre i due avvolgimenti generando un campo magnetico, che entra ed esce dalle parti arrotondate dell'armatura. Si generano così, forze di attrazione e repulsione con i magneti permanenti fissi (indicati con N ed S nelle figure).

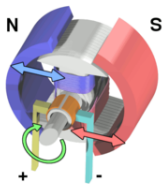
FASE 1



Quando la corrente scorre negli avvolgimenti, si genera un campo magnetico intorno al rotore. La parte sinistra del rotore è respinta dal magnete di sinistra ed attirata da quello di destra. Analogamente fa la parte in basso a destra.
Si genera la rotazione.

FASE 2
Il rotore continua a girare.

FASE 3



Quando le armature si allineano orizzontalmente, il commutatore inverte la direzione di corrente attraverso gli avvolgimenti, modificando anche il campo magnetico. Il processo ritorna quindi allo stato di partenza e il ciclo si ripete.

La velocità di rotazione del motore dipende da:
• Tensione applicata;
• Corrente assorbita dal rotore
• Carico applicato.

La coppia generata è proporzionale alla corrente ed il controllo più semplice agisce sulla tensione d'alimentazione. Talvolta è usato anche per il recupero dell'energia.

Il suo limite principale sta nel commutatore essendo costituito da spazzole in lega metallica bianca. Queste infatti, pongono un freno alla velocità massima di rotazione poiché maggiore è la velocità e più forte sarà la pressione che bisogna esercitare per mantenere un buon contatto.

Tra spazzole e collettore, nei momenti di commutazione, si hanno degli scintillii che possono comportare disturbi elettrici.

Il motore è equipaggiato con una protezione termica (fusibile termico o termistore) in modo da proteggerlo da sovracorrenti ed eventuali blocchi violenti (Raychem RXE065 o Bourns MF-R065).

b. ENCODER

L'encoder (trasduttore di posizione angolare o sensore di rotazione) è un dispositivo elettromeccanico che converte la posizione angolare del suo asse rotante in segnali elettrici numerici digitali e che consente il controllo del motore con una precisione di ±1°. L'encoder può essere utilizzato per diversi scopi (vedi applicazioni) e permette di sincronizzare automaticamente i motori.

Gli encoder si dividono in due grandi categorie:
• Assoluti
• Relativi

L'encoder è costituito da:
• 1 Ingranaggio che presenta 12 fessure;
• 2 LED;
• 2 fotoresistenze.

Il rapporto di riduzione tra l'ingranaggio del motore e l'ingranaggio dell'encoder è: 10:32 (ruota motrice / ruota condotta).

10:32 -> 1:3,2

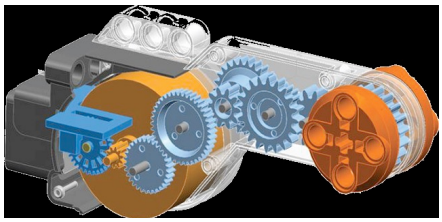
Ciò significa che la ruota motrice, per far compiere alla ruota condotta un giro completo, deve svolgere 3,2 giri.

c. INGRANAGGI

Dentro il servomotore vi sono complessivamente 6 ingranaggi. Questi trasferiscono il movimento del nucleo motore all'uscita dell'albero motore.

Gli ingranaggi sono tra loro in rapporto di trasmissione:
10 : 30 : 40 = 1 : 4
9 : 27 = 1 : 3
10 : 20 = 1: 2
10 : 13 : 20 = 1 : 2
Complessivamente -> 1 : 48

Ciò significa che 48 giri dell'albero motore principale corrispondono a 1 giro dell'albero in uscita.



4. Funzionamento

Il servomotore permette al robot di muoversi e per questo motivo è costituito da una serie di ingranaggi che lo rendono molto robusto.

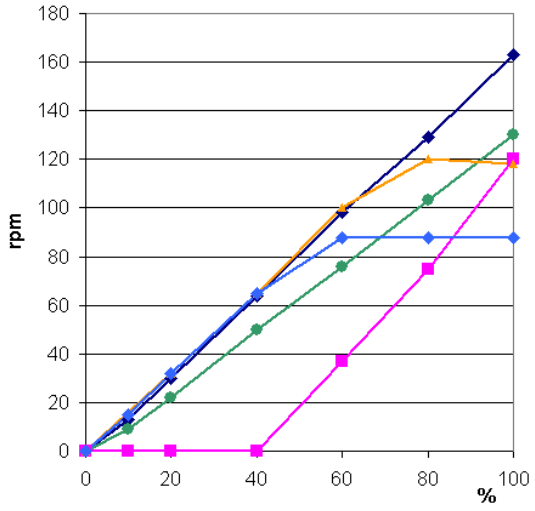
Come già detto in precedenza, il servomotore è provvisto di un encoder che possiede una precisione di ± 1 °.

5. Dati tecnici

Gamma funzionamento (V)	Nominale
0 -9 V	9 V

Tensione alimentazione 9 V (C.C.) SENZA CARICO						
Velocità massima (rpm)	Corrente (mA)	Coppia (N*m)	Efficienza (%)	Assorbimento (mA)	Potenza meccanica (W)	Potenza elettrica (W)
117	60	16,5	41	0,55	2,03	4,95

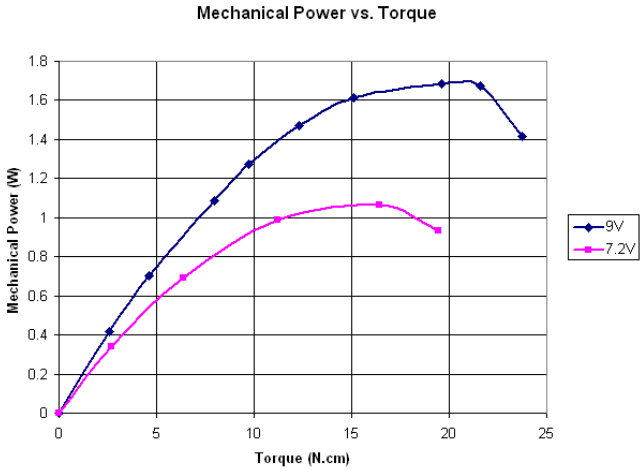
6. Curve delle caratteristiche del servomotore



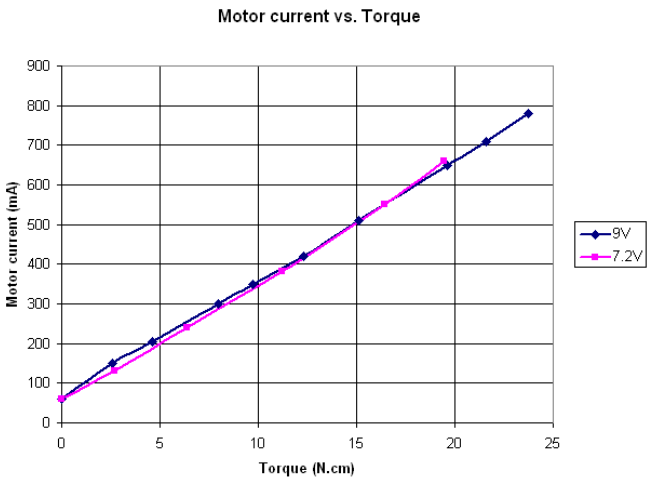
Descrizione
Rotazione del motore senza carichi (a vuoto) a 9V. E' evidente la relazione lineare tra il livello di potenza e velocità del motore.
Rotazione del motore senza carichi (a vuoto) a 7.2V. La velocità di rotazione è proporzionalmente inferiore rispetto alla rotazione del motore a 9 V.
Rotazione del motore con un carico 11.5 N×cm a 9V . Sotto il 40% della potenza del motore questo non si muove (regione orizzontale della curva). Una volta che viene applicata abbastanza potenza, la velocità aumenta proporzionalmente.
Rotazione del motore con un carico 11,5 N×cm con 9V. Questa curva mostra l'efficienza del Power Control: fino al 70% la velocità è la stessa di un motore senza carico. Dopo che la curva è piatta, il motore effettivamente funziona a piena potenza.
Rotazione del motore con un carico 11,5 N×cm con 7.2V. Questa curva mostra l'efficienza del Power Control: fino al 50% la velocità è la stessa come a vuoto, il motore viene alimentato a 9V. Poi data la minore corrente i motori non raggiungono le stesse apettative di quelli con 9V.

Le seguenti tabelle mostrano le caratteristiche del motore NXT con un carico applicato. Per le curve blu scuro, l'NXT è stato alimentato a 9V (tensione di batterie alcaline), per quelle magenta invece è stato alimentato a 7,2 V (tensione di batterie NiMH). Livello di potenza è 100% per tutti i grafici.

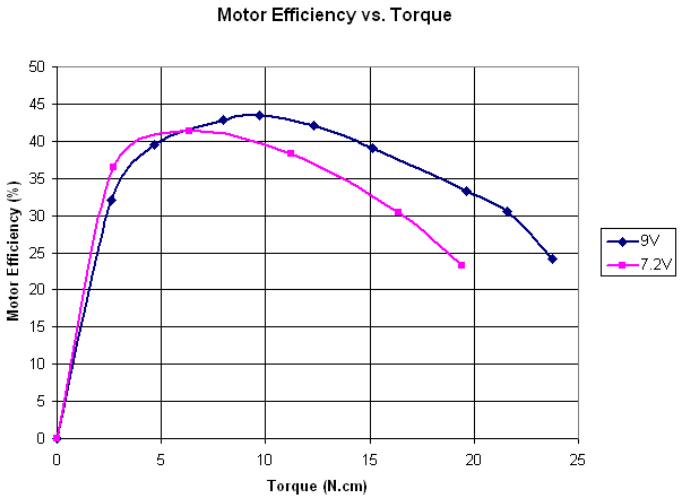
I grafici successivi sono tutti in funzione del carico applicato.



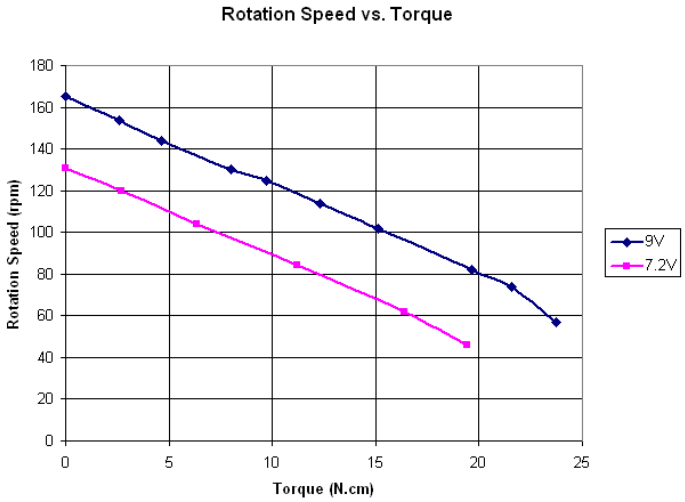
Descrizione
Grafico relativo alla potenza meccanica in funzione del carico applicato (N×cm);



Descrizione
Grafico relativo alla corrente assorbita dai motori in funzione del carico applicato (N×cm);



Descrizione
Grafico relativo all'efficenza dei motori in funzione del carico applicato (N×cm);



Descrizione
Grafico relativo alla rotazione dei motori in funzione del carico applicato (N×cm);

7. Limiti e problemi riscontrati

Il servomotore presenta alcuni limiti e o problemi risolvibili ponendo la giusta attenzione alla scrittura del codice.

a. BATTERIA

Prendiamo in considerazione il seguente programma:

Codice:
```
task main() {
    OnFwd(OUT_AC,100);
    while(TRUE){
    }
}
```

Il codice precedente fa muovere in avanti i motori delle uscite A e C al 100% della velocità.

Essendo però i servomotori alimentati direttamente dall'NXT attraverso le porte di uscita, quando la carica dell'NXT va poco a poco ad esaurirsi, anche i motori rallentano proporzionalmente.

b. SALITA

Quando il robot percorre una salita bisogna considerare 2 aspetti fondamentali: coppia e sincronizzazione dei motori. Questi problemi sono facilmente risolvibili utilizzando gli opportuni comandi (vedi software).

c. SCRITTURA DEL CODICE

ESEMPIO

Codice ESEMPIO 1:
```
task main() {
    while(TRUE){
        OnFwd(OUT_AC,75);
    }
}
```

Codice ESEMPIO 2:
```
task main() {
    OnFwd(OUT_AC,75);
    while(TRUE){
    //NESSUN CODICE INTERNO
    }
}
```

I due esempi pur facendo la stessa cosa cioè: far muovere in avanti i motori dell'uscita A e C al 75% della velocità, sono diversi sotto l'aspetto dell'utilizzo delle risorse.

Il codice ESEMPIO 1 ha un utilizzo delle risorse molto maggiore rispetto al codice ESEMPIO 2, perché il comando OnFwd(OUT_AC,75); viene sempre utilizzato ogni volta che il ciclo si ripete (in questo caso all'infinito).

8. Software (SERVOMOTORE INTERATTIVO)

a. Ruotare motore avanti

OnFwd (byte outputs, char pwr)

PARAMETRI

• outputs indica la porta. Vedi costanti delle porte di uscita.
• pwr indica la potenza in uscita (0 - 100). Può essere negativa per direzione inversa.

ESEMPI

OnFwd(OUT_AC,75);
Muove i due motori nell'uscita A e C in avanti al 75%.

b. Ruotare motore indietro

OnRev (byte outputs, char pwr)

PARAMETRI

• outputs indica la porta. Vedi costanti delle porte di uscita.
• pwr indica la potenza in uscita (0 - 100). Può essere negativa per direzione inversa.

ESEMPI

OnRev(OUT_AC,50);
Muove i due motori nell'uscita A e C all'indietro al 50%.

c. Ruotare motore avanti con una certa regolazione

OnFwdReg (byte outputs, char pwr, byte regmode)

PARAMETRI

• outputs indica la porta. Vedi costanti delle porte di uscita.
• pwr indica la potenza in uscita (0 - 100). Può essere negativa per direzione inversa.
• regmode indica la regolazione. Vedi costanti delle porte di regolazione.

ESEMPI

OnFwdReg(OUT_AC,75,OUT_REGMODE_IDLE);
Fa ruotare i due motori avanti collegati alle uscite A e C al 75% senza nessuna regolazione.

d. Ruotare motore indietro con una certa regolazione

OnRevReg (byte outputs, char pwr, byte regmode)

PARAMETRI

• outputs indica la porta. Vedi costanti delle porte di uscita.
• pwr indica la potenza in uscita (0 - 100). Può essere negativa per direzione inversa.
• regmode indica la regolazione. Vedi costanti delle porte di regolazione.

ESEMPI

OnRevReg(OUT_AC,75,OUT_REGMODE_SYNC);
Muove i due motori nell'uscita A e C all'indietro al 75%, regolati in modo da muoversi sincronizzati.

e. Ruotare motore con un grado di sincronia

OnFwdSync (byte outputs, char pwr, char turnpct)

PARAMETRI

• outputs indica la porta. Vedi costanti delle porte di uscita.
• pwr indica la potenza in uscita (0 a 100) Può essere negativa per direzione inversa.
• turnptc indica il rapporto di svolta (-100 a 100).

ESEMPI

OnFwdSync (OUT_AC,75,-100);
Muove i due motori nell'uscita A e C in avanti al 75%, regolati in modo da muoversi sincronizzati.
regolati in modo da muoversi sincronizzati utilizzando il rapporto di svolta specificato (varia da robot a robot).

f. Ruotare motore di n gradi

RotateMotor (byte outputs, char pwr, long angle)

PARAMETRI

• outputs indica la porta. Vedi costanti delle porte di uscita.
• pwr indica la potenza in uscita (0 a 100) Può essere negativa per direzione inversa.
• angle indica di quanti gradi far girare il motore.

ESEMPI

RotateMotor(OUT_A,75,45);
Muove i due motori nell'uscita A e C al 50% di 45 gradi.

g. Ruotare motore di n gradi con un rapporto di svolta in modo sincronizzato e stop

RotateMotorEx (byte outputs, char pwr, long angle, char turnpct, bool sync, bool stop)

PARAMETRI

• outputs indica la porta. Vedi costanti delle porte di uscita.
• pwr indica la potenza in uscita (0 a 100) Può essere negativa per direzione inversa;
• angle indica di quanti gradi far girare il motore;
• turnptc indica il rapporto di svolta (-100 a 100);
• sync indica se la sincronizzazione dei motori è attivata (TRUE) o disattivata(FALSE);
• stop indica di quanti gradi far girare il motore.

ESEMPI

RotateMotorEx(OUT_AC,75,45,0,TRUE,FALSE);
Fa ruotare il motore A e C in avanti al 75% della potenza di 45 gradi in modo sincronizzato.

h. Spegnimento dei motori in modo brusco

Off (byte outputs)

PARAMETRI

• outputs indica la porta. Vedi costanti delle porte di uscita.

ESEMPI

Off(OUT_AC);
Ferma in modo brusco i motori collegati alle porte A e C.

i. Spegnimento dei motori in modo dolce

Coast (byte outputs)

PARAMETRI

• outputs indica la porta. Vedi costanti delle porte di uscita.

ESEMPI

Coast(OUT_AC);
Ferma in modo dolce i motori collegati alle porte A e C.

j. Encoder

MotorTachoCount (byte output)

PARAMETRI

• outputs indica la porta. Vedi costanti delle porte di uscita.

ESEMPI

Ferma in modo dolce i motori collegati alle porte A e C.
x = MotorTachoCount(OUT_A);
Assegna a x il valore dell'encoder del motore nell'uscita A.

9. Applicazioni

GIRARE MOTORI

I motori collegati alle porte di uscita A e C vengono fatti muovere in avanti al 75% della velocità per 10 secondi, dopodiché vengono fermati bruscamente.

Codice utilizzato:

```
task main() {
    OnFwd(OUT_AC,75);
    Wait(10000); //Aspetta 10000ms
    Off(OUT_AC);
}
```

GIRARE MOTORI v2

I motori collegati alle porte di uscita A e C vengono fatti muovere in avanti al 75% della velocità per 10 secondi, dopodiché vengono fermati progressivamente.

Codice utilizzato:

```
task main() {
    OnFwd(OUT_AC,75);
    Wait(10000);
    Coast(OUT_AC);
}
```

GIRARE MOTORI v3

I motori collegati alle porte di uscita A e C vengono fatti muovere sincronizzati in avanti al 75% della velocità per 10 secondi, dopodiché vengono fermati progressivamente.

Codice utilizzato:

```
task main() {
    OnFwdSync(OUT_AC,50,360);
    Wait(10000);
    Coast(OUT_AC);
}
```

N.B. : Quando i motori devono essere fatti muovere sincronizzati ma uno dei due è scollegato dalla porta neanche l'altro si muove.

GIRARE MOTORI v4

I motori collegati alle porte di uscita A e C vengono fatti muovere sincronizzati in avanti al 75% della velocità per 360 gradi, dopodiché vengono fermati progressivamente.

Codice utilizzato:

```
task main() {
    RotateMotor(OUT_AC,50,360);
    Float(OUT_AC);
}
```

GIRARE MOTORI v5

I motori collegati alle porte di uscita A e C vengono fatti muovere sincronizzati in avanti al 50% della velocità per 360 gradi con una percentuale di sterzata del 100%, dopodiché vengono fermati progressivamente.

Codice utilizzato:

```
task main() {
    RotateMotorEx(OUT_AC, 50, 360, 100, true, false);
    Coast(OUT_AC);
}
```

GIRARE MOTORI v6

I motori collegati alle porte di uscita A e C vengono fatti muovere sincronizzati in avanti al 50% della velocità mantenendola per 5 secondi, dopodiché vengono fermati progressivamente.

Codice utilizzato:

```
task main() {
    OnFwdReg(OUT_AC,50,OUT_REGMODE_SPEED);
    Wait(5000);
    Coast(OUT_AC);
`
```

CONTEGGIO ENCODER

Vengono conteggiati i gradi che compie il motore nell'uscita A quando viene girato. Col bottone centrale il conteggio viene resettato. Questo risulta utile per calcolare la distanza percorsa dal robot.

Codice utilizzato:

```
task main(){
long conteggio;

while(TRUE) {
    if (ButtonPressed(BTNCENTER, false)){
        ResetTachoCount(OUT_A);
    }
    conteggio =  MotorTachoCount(OUT_A);
    ClearScreen();
    NumOut(0, LCD_LINE1, conteggio);
    Wait(200);
    }

}
```

GIRARE IL ROBOT DI N GRADI

Il robot viene fatto girare di n gradi su sè stesso.

Codice utilizzato:

```
float DiametroRuota = 56.0;
float Carreggiata = 113.0;

void gira(float angolo) {
    float DEG, SPACE;
    DEG = angolo*Carreggiata/DiametroRuota;
    NumOut(0, LCD_LINE1, DEG);
    if(DEG>0) {
        RotateMotorEx(OUT_AC, 20, DEG, 100, true, true);
    }
    else {
        RotateMotorEx(OUT_AC, 20, DEG, -100, true, true);
    }
}

task main() {
    gira(90.0);
    Wait(2000);
    gira(-90.0);
    Wait(2000);
    gira(360);
    Wait(2000);
}
```

AGGIRARE OSTACOLO

Il robot avanza finché non incontra un ostacolo (20 × 20 cm). Lo aggira e prosegue.

Codice utilizzato:

```
#define speed 50
#define angolo 90
#define distanza 20
float diametro_ruota = 56.0;
float carreggiata = 113.0;
float DEG, DIST;

void supero_stacolo() {

    DEG = angolo*carreggiata/diametro_ruota;
    DIST = 360/(diametro_ruota*3.1415)*(distanza*3);
    RotateMotorEx(OUT_AC,speed, DEG, 100, true, true);
    RotateMotorEx(OUT_AC,speed, DIST, 0, true, true);
    RotateMotorEx(OUT_AC,speed, DEG, -100, true, true);
    RotateMotorEx(OUT_AC,speed, DIST, 0, true, true);
    RotateMotorEx(OUT_AC,speed, DEG, -100, true, true);
    RotateMotorEx(OUT_AC,speed, DIST, 0, true, true);
    RotateMotorEx(OUT_AC,speed, DEG, 100, true, true);
    OnFwd(OUT_AC,75);
}

task main() {
    SetSensorLowspeed(IN_1);

    OnFwd(OUT_AC,50);
    while(SensorUS(IN_1)>20){
        Wait(50);
    }
    Off(OUT_AC);
    Wait(1000);
    supero_stacolo();
}
```

Costanti

a. Costanti delle porte di ingresso

- #define S1 0
- #define S2 1
- #define S3 2
- #define S4 3

b. Costanti delle porte in uscita

- #define OUT_A 0x00
- #define OUT_B 0x01
- #define OUT_C 0x02
- #define OUT_AB 0x03
- #define OUT_AC 0x04
- #define OUT_BC 0x05
- #define OUT_ABC 0x06

c. Costanti di modo e tipo del sensore

- #define _SENSOR_CFG(_type, _mode) (((_type)<<8)+(_mode))
- #define SENSOR_LIGHT _SENSOR_CFG(SENSOR_TYPE_LIGHT, SENSOR_MODE_PERCENT)
- #define SENSOR_NXTLIGHT_SENSOR_CFG(SENSOR_TYPE_LIGHT_ACTIVE,SENSOR_MODE_PERCENT)

d. Costanti modalità del sensore

- #define SENSOR_MODE_RAW IN_MODE_RAW
- #define SENSOR_MODE_BOOL IN_MODE_BOOLEAN
- #define SENSOR_MODE_EDGE IN_MODE_TRANSITIONCNT
- #define SENSOR_MODE_PULSE IN_MODE_PERIODCOUNTER
- #define SENSOR_MODE_PERCENT IN_MODE_PCTFULLSCALE
- #define SENSOR_MODE_CELSIUS IN_MODE_CELSIUS
- #define SENSOR_MODE_FAHRENHEIT IN_MODE_FAHRENHEIT
- #define SENSOR_MODE_ROTATION IN_MODE_ANGLESTEP

In modalità RAW vengono restituiti valori da 0 a 1023.
In modalità PERCENT vengono restituiti valori da 0 a 100.

e. Costanti tipologie del sensore

- #define SENSOR_TYPE_NONE IN_TYPE_NO_SENSOR
- #define SENSOR_TYPE_TOUCH IN_TYPE_SWITCH
- #define SENSOR_TYPE_TEMPERATURE IN_TYPE_TEMPERATURE
- #define SENSOR_TYPE_LIGHT IN_TYPE_REFLECTION
- #define SENSOR_TYPE_ROTATION IN_TYPE_ANGLE
- #define SENSOR_TYPE_LIGHT_ACTIVE IN_TYPE_LIGHT_ACTIVE
- #define SENSOR_TYPE_LIGHT_INACTIVE IN_TYPE_LIGHT_INACTIVE
- #define SENSOR_TYPE_SOUND_DB IN_TYPE_SOUND_DB
- #define SENSOR_TYPE_SOUND_DBA IN_TYPE_SOUND_DBA
- #define SENSOR_TYPE_CUSTOM IN_TYPE_CUSTOM
- #define SENSOR_TYPE_LOWSPEED IN_TYPE_LOWSPEED
- #define SENSOR_TYPE_LOWSPEED_9V IN_TYPE_LOWSPEED_9V
- #define SENSOR_TYPE_HIGHSPEED IN_TYPE_HISPEED
- #define SENSOR_TYPE_COLORFULL IN_TYPE_COLORFULL
- #define SENSOR_TYPE_COLORRED IN_TYPE_COLORRED
- #define SENSOR_TYPE_COLORGREEN IN_TYPE_COLORGREEN
- #define SENSOR_TYPE_COLORBLUE IN_TYPE_COLORBLUE
- #define SENSOR_TYPE_COLORNONE IN_TYPE_COLORNONE

f. Lettura analogica dei valori dei sensori

- #define SENSOR_1 Sensor(S1)
- #define SENSOR_2 Sensor(S2)
- #define SENSOR_3 Sensor(S3)
- #define SENSOR_4 Sensor(S4)

g. Costanti di regolazione delle porte di output

- #define OUT_REGMODE_IDLE 0
- #define OUT_REGMODE_SPEED 1
- #define OUT_REGMODE_SYNC 2
- #define OUT_REGMODE_POS 4

BIBLIOGRAFIA

http://www.legoengineering.com/nxt-sensors/

http://www.lego.com/en-us/mindstorms/downloads

http://nxt.cs.uwindsor.ca//499football/features_limitations.pdf

http://cache.lego.com/downloads/education/9797_LME_UserGuide_US_low.pdf

http://www.nxtprograms.com/NXT2/multi-bot/programs.html

https://sites.google.com/site/tuftsceeok12projects/labview-for-lego-support/getting-started/light-sensors

http://digilander.libero.it/Valter_NXT/kitnxt.html

http://www.philohome.com/nxtmotor/nxtmotor.htm

http://www.adrirobot.it/lego/mindstorms_nxt/mindstorms_ntx.htm

https://nxttime.wordpress.com/2012/09/12/the-ultrasonic-sensor/

http://nxt.cs.uwindsor.ca//499football/features_limitations.pdf

http://www.salatin.eu/wp-content/files/Laboratorio_Avanzato_di_Robotica.pdf

http://www.generationrobots.com/en/content/65-ultrasonic-sonar-sensors-for-robots