Usare lego lib

indice

introduzione

Lego lib è una libreria per controllare il brick Lego Mindstorm NXT con più facilià. Principalmente questa libreria è composta dai blocchi arancioni e da quelli verdi del'ambiente sviluppo Mindstorm NXT.

- Bloccchi arancioni: Servono per aspettare che un determinato sensore legga un determinato valore
- Blocchi verdi: servono per la navigazione del robot

Installazione

Per poter utilizzare legolib c'è bisogno dell'ambiente di sviluppo di lejos, siccome esso è sviluppato in java vi è bisogno di installare per prima cosa il suo ambiente di sviluppo.

Java

Java ha la potenza di poter essere eseguito su tutte le architetture di sistema operativo, che esso sia Windows, UNIX/Linux, Mac OS, sia a 32bit che a 64bit. Per avere questa potenza gli sviluppatori della Oracle (azienda che produce Java), hanno sviluppato una virtual machine che esegue il codice java. Quindi va installata, e siccome dobbiamo sviluppare serve anche il kid di sviluppo di java.

Java Runtime Environment

Prima di installare la JRE bisogna provare a controllare se java non è già installato sul computer. Quindi aprere il `Prompt dei comandi`, premere il tasto Windows che si trova sulla tastiera e contemporaneamente il tasto `R`. Quindi ciò farà aprire una piccola finestra in basso a sinistra, in cui bisognerà digitare `cmd` e premere invio. Comparirà una finestra nera, sulla quale scrivere il seguente comando:

```
java –version
```

E premere invio, se questo ritora una stringa a simile:

```
java version "1.8.0_191"
Java(TM) SE Runtime Environment (build 1.8.0_191-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)
```

se compare una stringa contentente qualcosa di simile a `comando sconosciuto` proseguire con la guida, altrimenti passare direttamente al prossimo capitolo (Java Development Kit).

Per installare java scaricare il pacchetto di installazione dal sito della Oracle, al seguente link: https://www.java.com/en/download/, dopo averlo scaricato aprire il file e seguire la procedura guidata.

Java Development Kit

Prima di installare la JDK bisogna provare a controllare se essa è gi&arave; presente sulla macchina. Quindi digitare sul `Prompt dei comandi` aperto nel precedente capitolo, il seguente e digitare il segutente comando:

```
javac –version
```

se il programma ritorna una stringa come la seguente:

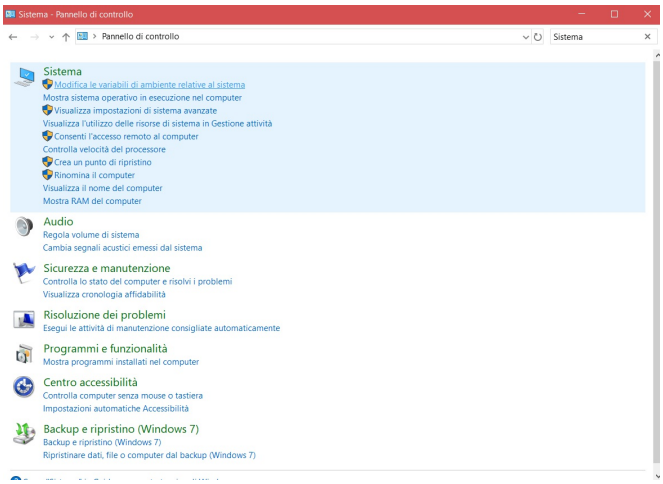
```
javac 1.8.0_191
```

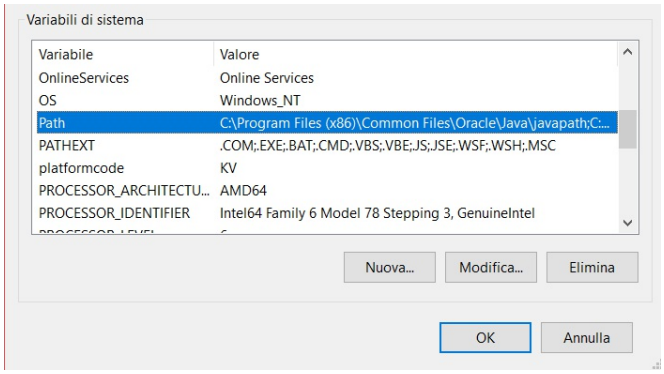
passare direttamente al capitolo "Driver USB", altrimenti proseguire con la guida.

Per insallare la JDK scaricare sempre dal sito ufficiale della Oracle al seguente link: https://www.java.com/en/download/ il pacchetto di installazione del Java Development Kit, successivamente installarlo e seguire la procedura guidata.

Variabili d'ambiente

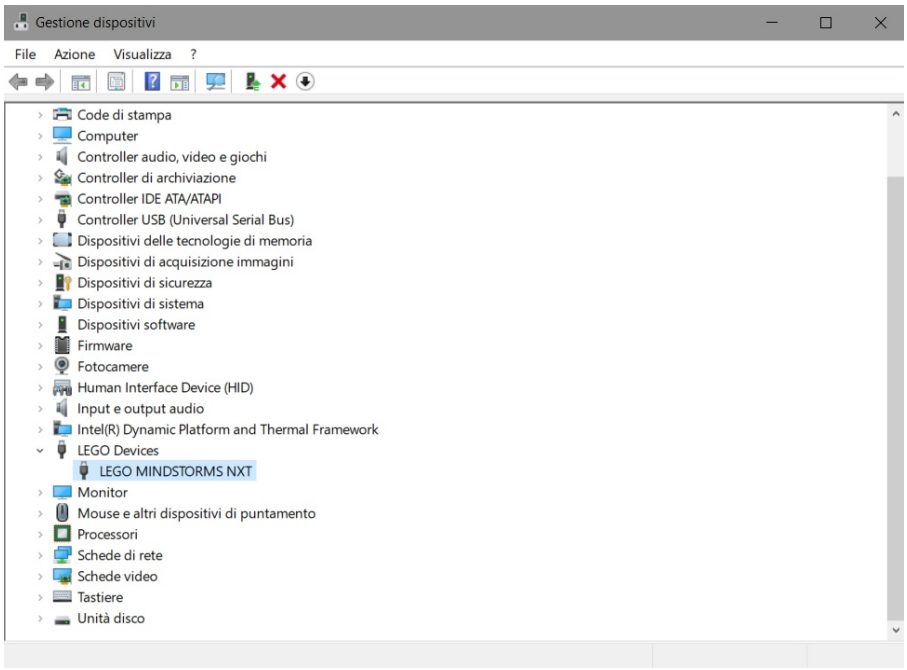
Prima di poter utilizzare Java bisogna impostare le sue `variabili d'ambiente`. Quindi andare nella cartella dove si ha installato Java (probabilmente `C:\Program Files (x86)\Java\<versione di Java>\bin`), copiare i percorsi delle due cartelle fino a `bin` e aprire il `Pannello di controllo`, recarsi nella sezione `Sistema`, poi su `Variabili d'ambiente` che si trovano sotto le impostazioni `Avanzate` e infine aggiungere il percorso nella variabile di sistema `Path`.





Driver USB

Per fare in modo che il computer individui il dispositivo LEGO®, è necessario disporre di un driver USB. Esso si scarica dal sito ufficiale Mindstorms (https://www.lego.com/r/www/r/mindstorms/-/media/franchises/mindstorms%202014/downloads/firmware%20and%20software/nxt%20software/nxt%20fantom%20drivers%20v120.zip?l.r2=-964392510) e bisognerà avviare il file `setup.exe` sotto la cartella `Windows` che si trova nella cartella compressa appena scaricata.
Quindi verificare che il brick NXT sia riconosciuto dal vostro PC aprendo `Gestione dispositivi` dal `Pannello di controllo` e il dispositivo verrà identificato in questo modo:



leJOS

Installazione

Procedere scaricando l'applicativo dell'ultima versione (0.9.1) dal sito (https://sourceforge.net/projects/nxt.lejos.p/files/latest/download). Quindi avviare il file appena scaricato e eseguire la procedura guidata dell'installer.

Configurazione

Assicurarsi che il brick sia acceso collegato correttamente via USB, successivamente finire la configurazione avviando la finestra finale dove è attivata l'opzione `Launch NXJ Flash utility`. L'applicazione dovrebbe identificare il brick, quindi cliccare `Start program` e procedere con l'attivazione del firmware.
Ora la configurazione è giunta al termine e sul NXT dovrebbe apparire la schermata di leJOS.

Utilizzo

In questo capitolo viene spiegato come iniziare a programmare con leJOS NXJ tramite un classico `HelloWorld`.
È solamente necessario disporre di un editore di testo per scrivere il codice.

HelloWorld

Iniziare creando il file `HelloWorld.java` e scrivendo la classe `HelloWorld` tramite il pacchetto predefinito di Java.

```
public class HelloWorld {

}
```

Proseguire implementando il metodo `main` (che viene di solito usato come il metodo che genera un output).

```
public class HelloWorld {
    public static void main (String[] args) {

    }
}
```

Ora scrivere la classica funzione che genera un output sotto forma di testo nello schermo LCD del brick.

```
public class HelloWorld {
    public static void main (String[] args) {
        System.out.println("Hello World");
    }
}
```

Se si avvia il programma in questo modo, verrà mostrata la scritta "HelloWorld" e si tornerà immediatamente nella schermata principale. Per limitare questo comportamento, si può inserire l' opzione che aspetta la pressione di un pulsante. Per fare ciò basta importare la libreria `Button` e inserire un semplice metodo.

```
import lejos.nxt.Button;

public class HelloWorld {
    public static void main (String[] args) {
        System.out.println("Hello World");
        Button.waitForAnyPress();
    }
}
```

Adesso il codice è pronto per essere compilato nel NXT e avviato.

Compilazione e avvio

Per verificare il funzionamento del codice appena scritto, bisogna aprire un'istanza di `Windows PowerShell` nella cartella dove si trova il file `HelloWorld.java`, cliccando sul menu `File` in alto a sinistra e scegliendo `Apri Windows PowerShell`.
Si dovrebbe aprire una schermata blu, in cui occorre scrivere i seguenti comandi uno dopo l'altro:

```
nxjc HelloWorld.java
```

Compilazione del file.

```
nxjlink -o HelloWorld.nxj HelloWorld
```

Caricamento della classe in un file compatibile con NXT.

```
nxjupload -r HelloWorld.nxj
```

Caricamento del file nel brick.

```
nxj -r -o HelloWorld.nxj HelloWorld
```

Avviamento del programma.

LegoLib & IDE

Gli IDE (Integrated Development Environment) sono degli applicativi studiati per facilitare il compito agli sviluppatori, per poter utilizzare legolib in un IDE bisogna importare in esso il file `.jar` che contiene tutte le classi di legolib. Per ogni IDE vi è una diffrente procedura.

Com'è strutturato

Lego lib, principalmente è compreso di una libreria che rappresentano i blocchi arancioni e quelli verdi dell'ambiente di sviluppo Mindstorm NXT.

Blocchi arancioni

Servono per aspettare che un determinato sensore legga un determinato valore oppure rappresentano le strutture di controllo di programmazione sequenziale.
Per esempio aspettare che il sensore di luce riflessa legga un valore più alto del 50%.

I blocchi arancioni rappresentano i blcochi blocchi Wait, che in lego lib sono rappresentati dalle classi contenute nel package `legolib` che cominciano il loro nome con `Wait`.

Nei blocchi arancioni sono compresi anche le selezioni e i cicli, i quali sono implementati dalle strutture di controllo `if (...) { ... }` e `while` o `do { ... } while` oppure `for (...) { ... }`.

Wait

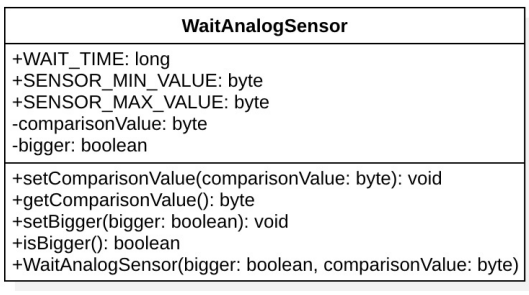
Tutte le classi hanno in comune hanno un costruttore che permette di inizializzare ogni wait con tutte le configurazioni possibili ed un metodo che fa eseguire lo wait.

Ogni waiter ha almeno un costruttore, nel quale vi sono tutti i parametri con il quale lo si configura ed un metodo che inizia con `wait` e finisce con il nome del sencore che deve aspettare, questo metodo &egrave quello da utilizzare per eseguire lo waiter.

Sensori analogici

Gli waiter analogici si basano su una variabile contentente il valore di riferimento, questo valore viene confrontato con quello letto dai sensori. Poi vi è una variabile boleana, la quale viene utilizzata per sapere se il valore letto dal sensore deve essere maggiore o minore rispetto a quello memorizzato nella variabile di riferimento. Tutto questo è nella classe `WaitAnalogSensor`, la quale viene estesa dalle classi dei sensori analogici.

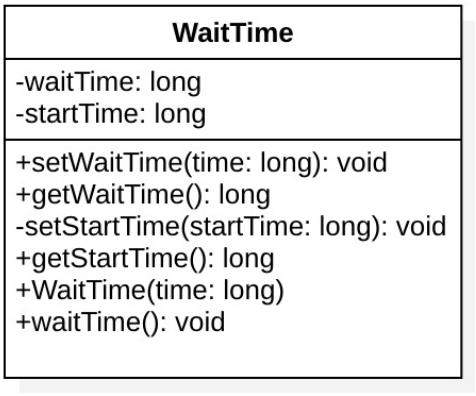
Diagramma UML della classe `WaitAnalogSensor`:



WaitTime

Il blocco wait time dell'ambiente di sviluppo Mindstorm NXT è rappresentato dalla classe `WaitTime`, la quale permette di aspettare del tempo, in millisecondi.

Il diagramma UML della classe:



La classe WaitTime è stata fatta per mantenere la coerenza con le altre classi, ma può essere facilmente sostituita da un `Thread.sleep(millis);`.

Esempio di utilizzo della classe in maniera asincrona:

```
import lejos.nxt.Button;

/**
 * Using WaitTime to test it.
 * Aspetta del tempo.
 *
 * @author gabrialessi
 * @author giuliobosco
 * @version 1.q (2019-02-05)
 */
public class UseWaitTime {

     /**
     * Metodo main della classe, avvia il programma di test della classe
     * WaitTime.
     *
     * @param args Argomenti da linea di comando.
     */
    public static void main(String[] args) {
        // creo lo wait time, con una attesa di 5000 millisecondi,
        // 5 secondi.
        WaitTime wt = new WaitTime(5000);

        // stamo il messaggio iniziale, "aspettando..."
        System.out.println("Aspettando...");

        // aspetto i 5000 millisecondi
        wt.waitTime();

        // stampo ilmessaggio finale
        System.out.println("Attesa terminata.");

        // aspetto che venga premuto un bottone sul brick per terminare
        // il programma
        Button.waitForAnyPress();
    }
}
```

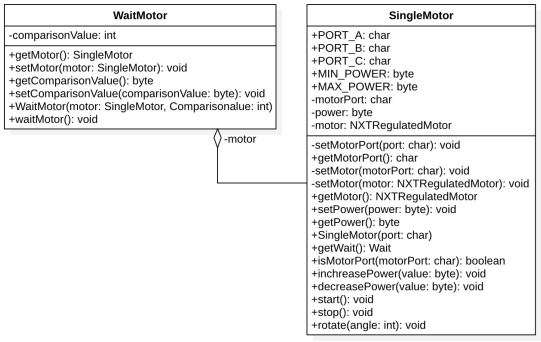
Oppure al posto della classe WaitTime, come detto in precedenza si può utilizzare il metodo `Thread.sleep(5000);`, che è un metodo interno alle libreire di java.

```
import lejos.nxt.Button;

/**
 * Using Thread.sleep(millis) for wait time.
 * Una alternativa alla classe WaitTime.
 *
 * @author giuliobosco
 * @version 1.0 (2019-02-05)
 */
public class UseThreadSleep {

     /**
     * Metodo main della classe, mostra come utilizzare il metodo
     * Thread.sleep(millis), che &egrave; una alternativa alla classe
     * WaitTime.
     *
     * @param args Argomenti da linea di comando.
     */
    public static void main(String[] args) {
        // per poter utilizzare il metodo Thread.sleep(millis) bisogna
        // utilizzare la struttura try {...} catch (Exception e) {...}
        // questo perch&eacute; la thread potrebbe venir interrotta e
        // provocherebbe un'eccezione.
        try {
            // stampo il messaggio iniziale, "aspettando..."
            System.out.println("Aspettando...");

            // aspetto i 5000 millisecondi
            Thread.sleep(5000);
        } catch (InterruptedException ie) {
            ie.printStackTrace();
        }

        // stampo ilmessaggio finale
        System.out.println("Attesa terminata.");

        // aspetto che venga premuto un bottone sul brick per terminare
        // il programma
        Button.waitForAnyPress();
    }
}
```

Wait Motor

Il blocco wait motor dell'ambiente di sviluppo Mindstorm NXT è rappresentato dalla classe `WaitMotor`, la quale permette di aspettare del che il motore abbia effettuato un determinato numbero di rotazioni.

Il diagramma UML della classe:



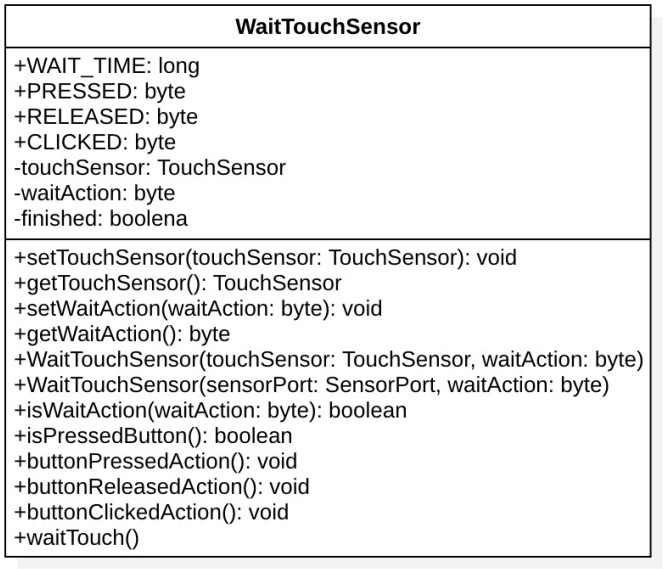
Esempio di utlizzo della classe:

```
import lejos.nxt.Button;

/**
 * Wait motor example class.
 * Aspetta che il motore effettui 3 rotazioni.
 *
 * @author giuliobosco
 * @version 1.0 (2019-02-01)
 */
public class UseWaitMotor {

    /**
     * Metodo main della classe, avvia il programma di test della classe
     * WaitMotor.
     *
     * @param args Argomenti da linea di comando.
     */
    public static void main(String[] args) {
        // creo il gestore del motore
        SingleMotor m = new SingleMotor('A');
        // creo lo waiter del motore
        WaitMotor wm = new WaitMotor(m, 3);

        // stampo il messaggio iniziale
        System.out.println("Avvio motore");
        // setto la velocita del motore a 10
        m.setPower((byte)10);
        // avvio il motore
        m.start();

        // aspetto le 3 rotazioni
        wm.waitMotor();

        // stampo il messaggio finale
        System.out.println("Fermo motore");
        // fermo il motore
        m.stop();

        // aspetto che venga premuto un bottone sul brick per terminare
        // il programma
        Button.waitForAnyPress();
    }
}
```

Wait Touch Sensor

Il blocco wait touch dell'ambiente di sviluppo Mindstorm NXT è rappresentato dalla classe `WaitTouchSensor`, la quale permette di aspettare del che un sensore di touch, su una delle porte venga premuto, o rilasciato oppure cliccato, cioè cliccato e rilasciato.

Il diagramma UML della classe:



Per sceglirere quale delle 3 azioni, (Premuto, rilasciato o cliccato) aspettare bisosgna cambiare il valore `CLICKED` nel costruttore dello waiter con:

- `PRESSED` per premuto
- `RELEASED` per rilasciato

Esempio di utlizzo della classe:

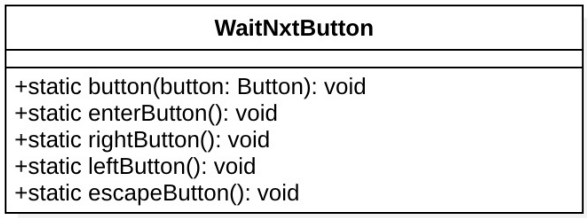
```
import lejos.nxt.Button;
import lejos.nxt.SensorPort;

/**
 * Wait touch sensor example class.
 * Aspetta che il touch sensor sulla porta 1 venga premuto.
 *
 * @author giuliobosco
 * @version 1.1 (01.02.2019)
 */
public class UseWaitTouchSensor {

    /**
     * Metodo main della classe, avvia il programma di test della classe
     * WaitTouchSensor.
     *
     * @param args Argomenti da linea di comando.
     */
    public static void main(String[] args) {
        // creo lo waiter per il touch sensor sulla porta uno, che aspetta
        // un click del sensore.
        WaitTouchSensor wtc = new WaitTouchSensor(SensorPort.S1,
                WaitTouchSensor.CLICKED);
```

```
        // stampo messaggio iniziale
        System.out.println("cliccare il touch sensor sulla porta 1");

        // aspetto che venga cliccato il touch sensor
        wtc.waitTouch();

        // stampo messagigo finale
        System.out.println("touch sensor cliccato");

        // aspetto che venga premuto un bottone sul brick per terminare
        // il programma
        Button.waitForAnyPress();
    }
}
```

Wait NXT Button

Il blocco wait nxt button dell'ambiente di sviluppo Mindstorm NXT è rappresentato dalla classe `WaitNxtButton`, la quale permette di aspettare del che venga premuto uno dei bottoni sul brick NXT.

Il diagramma UML della classe:



Per ogni bottone vi è un metodo statico, per cui per far aspettare il click di un bottone basterà richiamare il metodo.

- bottone sinistro: `WaitNxtButton.leftButton()`
- bottone di invio: `WaitNxtButton.enterButton()`
- bottone destro: `WaitNxtButton.rightButton()`
- bottone indietro: `WaitNxtButton.backButton()`

Esempio di utlizzo della classe:

```
import lejos.nxt.Button;

/**
 * Wait nxt button example class.
 * Testa la funzionalita dell'aspettare la premuta dei bottoni sul brick
 * nxt. Prima richiede di premere il tasto sinistro, poi quello di enter,
 * ed infine il tasto destro.
 *
 * @author giuliobosco
 * @version 1.1 (2019-02-01)
 */
public class UseWaitNxtButton {

    /**
     * Metodo main della classe, avvia il programma di test della classe
     * WaitNxtButton.
     *
     * @param args Argomenti da linea di comando.
     */
    public static void main(String[] args) {
        // aspetto che il tasto sinistro venga premuto
        System.out.println("Press left button to continue");
        WaitNxtButton.leftButton();

        // aspetto che il tasto enter venga premuto
        System.out.println(
                "Button pressed\n\nPress enter button to continue");
        WaitNxtButton.enterButton();

        // aspetto che il tasto destro venga premuto
        System.out.println(
                "Button pressed\n\nPress right button to continue");
        WaitNxtButton.rightButton();
        System.out.println("Button pressed\n\n");
```

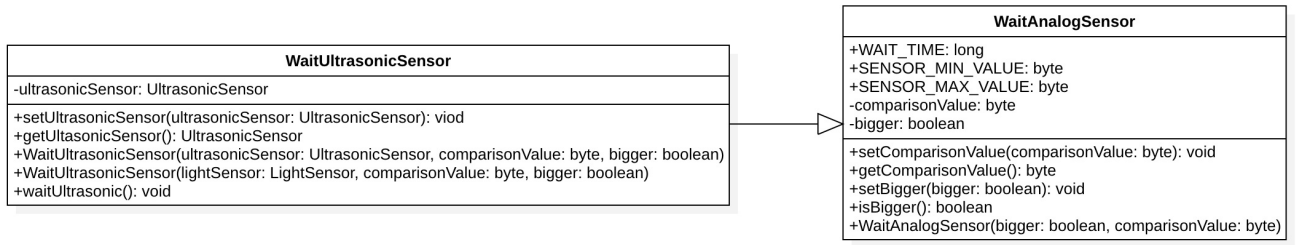
```
        // aspetto che venga premuto un bottone sul brick per terminare
        // il programma
        Button.waitForAnyPress();
    }
}
```

Wait Ultrasonic Sensor

Il blocco wait ultrasonic dell'ambiente di sviluppo Mindstorm NXT è rappresentato dalla classe `WaitUltrasonicSensor`, la quale permette di aspettare del che un sensore di ultrasuoni percepisca un valore più alto o piu basso di un certo valore.

Come descritto nel capitolo `Wait > Sensori Analogici` i valori di riferimento sono gestiti nella classe `WaitAnalogSensor`.

Il diagramma UML della classe:



Esempio di utlizzo della classe:

```
import lejos.nxt.Button;
import lejos.nxt.SensorPort;

/**
 * Wait ultrasonic sensor example class.
 * Aspetta che il sensore ad ultrasuoni (distanza) sulla porta 1
 * legga un valore maggiore di 50cm, poi aspetta che venga premuto
 * un qualunque tasto sul brick, poi aspetta di leggere un valore
 * minore di 50cm sul sensore.
 *
 * @author giuliobosco
 * @version 1.0 (2019-02-01)
 */
public class UseWaitUltrasonicSensor {

    /**
     * Metodo main della classe, avvia il programma di test della classe
     * WaitUltrasonicSensor.
     *
     * @param args Argomenti da linea di comando.
     */
    public static void main(String[] args) {
        // creo lo waiter per il ultrasonic sensor sulla porta uno.
        WaitUltrasonicSensor wus = new WaitUltrasonicSensor(
                SensorPort.S1, (byte) 50, true);

        // stampo messaggio iniziale, aspetto che il sensore ad ultrasuoni
        // legga un valore maggiore di 50cm
        System.out.println(
                "mettere il sensore piu lontano di 50cm dal sensore");
        wus.waitUltrasonic();

        // aspetto che venga premuto un qualunque bottone sul brick
        Button.waitForAnyPress();

        // stampo messaggio intermedio, aspetto che il sensore ad
        // ultrasuoni legga un valore miniore di 50cm
        wus.setBigger(false);
        System.out.println(
                "mettere il sensore piu vicino di 50cm dal sensore");
        wus.waitUltrasonic();
```

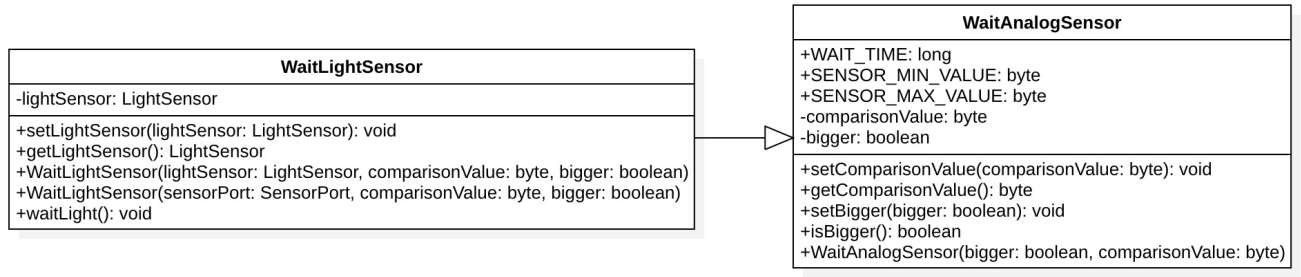
```
        // stampo messaggio finale
        System.out.println("fine del test");

        // aspetto che venga premuto un bottone sul brick per terminare
        // il programma
        Button.waitForAnyPress();
    }
}
```

Wait Light Sensor

Il blocco wait light dell'ambiente di sviluppo Mindstorm NXT è rappresentato dalla classe `WaitLightSensor`, la quale permette di aspettare del che un sensore di di suoni percepisca un valore più alto o piu basso di un certo valore.

Come descritto nel capitolo `Wait > Sensori Analogici` i valori di riferimento sono gestiti nella classe `WaitAnalogSensor`.

Il diagramma UML della classe:



Esempio di utlizzo della classe:

```
import lejos.nxt.Button;
import lejos.nxt.SensorPort;

/**
 * Wait light sensor example class.
 * Aspetta che il sensore di intensita di luce riflessa collegato
 * alla porta 1 vegna messo su una superfice chiara, e poi scura.
 *
 * @author giuliobosco
 * @version 1.0 (2019-02-01)
 */
public class UseWaitLightSensor {

    /**
     * Metodo main della classe, avvia il programma di test della classe
     * WaitLightSensor.
     *
     * @param args Argomenti da linea di comando.
     */
    public static void main(String[] args) {
        // creo lo waiter per il light sensor, sulla porta uno
        WaitLightSensor wls = new WaitLightSensor(
                SensorPort.S1, (byte)50, true);

        // stampo il messaggio iniziale e aspetto che il sensore
        // legga un valore alto piu alto di 50.
        System.out.println("Mettere su superficie chiara");
        wls.waitLight();

        Button.waitForAnyPress();

        // stampo il messaggio intermedio e aspetto che il sensore
        // legga un valore piu basso di 50.
        wls.setBigger(false);
        System.out.println("Mettere su superfice scura");
        wls.waitLight();

        // stampo messaggio finale
        System.out.println("Fine del test");
```

```
        // aspetto che venga premuto un bottone sul brick per terminare
        // il programma
        Button.waitForAnyPress();
    }
}
```

Calibrazione sensori

Per poter utilizzare in maniera ottimale i sensori, bisogna calibrarli con la luce attuale del' ambiente circostante. Per calibrare i sensori bisogna settare la luce massima e la luce minima che può leggere il sensore. La luce massima che un sensore può leggere solitamente è intesa come il bianco, che riflette molta luce; mentre la luce minima che il sensore può leggere è il nero, che riflette pochissima luce.

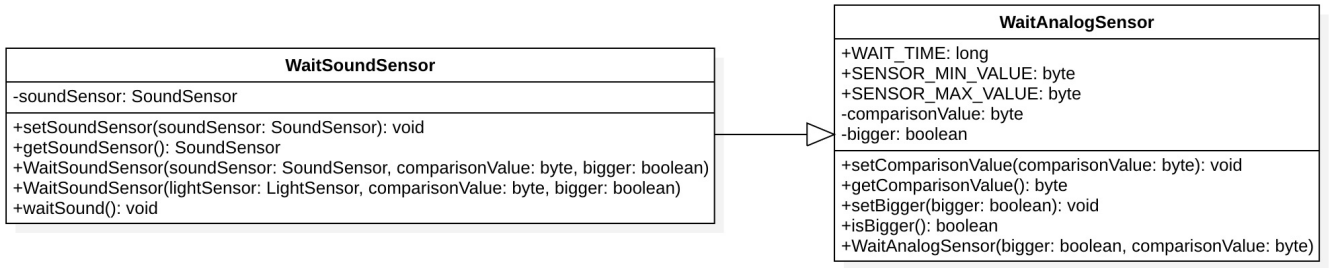
```
import lejos.nxt.LightSensor;
import lejos.nxt.SensorPort;

/**
 * Calibrate the light sensor.
 * Calibra il sensore di luce, con la luce che presente nel luogo
 * dov'&egrave; il sensore.
 *
 * @author giuliobosco
 * @version 1.0 (2019-02-06)
 */
public class LightSensorCalibrator {

    /**
     * Metodo main della classe, permette di calibrare il sensore
     * di luce.
     *
     * @param args Comm
     */
    public static void main(String[] args) {
        // setto il sensore di luce su cui eseguire la calibrazione
        LightSensor ls = new LightSensor(SensorPort.S1);

        // scrivo il messaggio per avvertire l'utente di mettere il
        // sensore di luce su una superfice bianca (o chiara)
        System.out.println(
                "Posizionare il sensore sul bianco. " +
                        "Poi premere Enter");

        // aspetto che venga premuto il bottone enter
        WaitNxtButton.enterButton();

        // calibro il massimo di luce letta sul senore
        ls.calibrateHigh();

        // pulisco il sensore del brick NXT
        System.out.println("\n\n\n\n\n\n\n\n\n\n");

        // scrivo il messaggio per avvertire l'utente di mettere il
        // sensore di luce su una superfice scura (o nera)
        System.out.println("Posizionare il sensore sul nero. Poi premere Enter.");

        // aspetto che venga premuto il bottone enter
        WaitNxtButton.enterButton();

        // calibro il minimo di luce letta sul sensore
        ls.calibrateLow();
    }

}
```

Wait Sound Sensor

Il blocco wait sound dell'ambiente di sviluppo Mindstorm NXT è rappresentato dalla classe `WaitSoundSensor`, la quale permette di aspettare del che un sensore di di suoni percepisca un valore più alto o piu basso di un certo valore.

Come descritto nel capitolo `Wait > Sensori Analogici` i valori di riferimento sono gestiti nella classe `WaitAnalogSensor`.

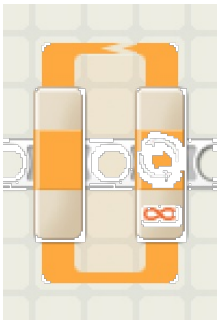
Il diagramma UML della classe:



Esempio di utlizzo della classe:

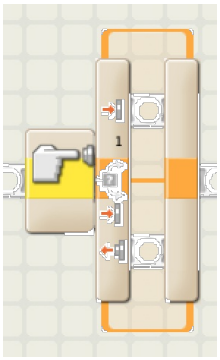
```
import lejos.nxt.Button;
import lejos.nxt.SensorPort;

/**
 * Wait sound sensor example class.
 * Aspetta che venga recepito un suono forte dal microfono
 * sulla porta 1.
 *
 * @author giuliobosco
 * @version 1.0 (2019-02-01)
 */
public class UseWaitSoundSensor {

    /**
     * Metodo main della classe, avvia il programma di test della classe
     * WaitSoundSensor.
     *
     * @param args Argomenti da linea di comando.
     */
    public static void main(String[] args) {
        // creo lo waiter per il sound sensor sulla porta uno, che aspetta
        // un suono forte.
        WaitSoundSensor wss = new WaitSoundSensor(
                SensorPort.S1, (byte)50, true);

        // stampo messaggio iniziale
        System.out.println("parlare davanti al microfono");

        // aspetto che venga recepito un suono forte
        wss.waitSound();

        // stampo messaggio finale
        System.out.println(
                "Valore alto del microfono recepito");

        // aspetto che venga premuto un bottone sul brick per terminare
        // il programma
        Button.waitForAnyPress();
    }
}
```

Strutture di controllo



La struttura di controllo del ciclo può essere rappresentata in diverse maniere:

- `while (...) { ... }`
- `do { ... } while (...)`
- `for (...) { ... }`



La struttura di controllo del switch o selezione è in programmazione è rappresentata dalla struttura di controllo `if (...) { ... }`

Blocchi verdi

I blocchi verdi dell'ambiente di sviluppo Lego® Mindstorm NXT sono quelli relativi agli attuatori.
Gli attuatori possono essere:

- motori
- schermi
- display
- led
- buzzer
- altoparlanti

Quindi sono tutti quegli elementi che collegati ad un cirtcuito di controllo, a dipendenza della loro natura e dell'istruzione o segnale che gli viene invato, fanno dei movimenti, o modificano il loro stato.

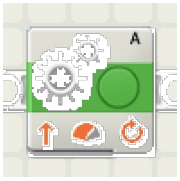
In questa libreria vi sono principalmente 2 classi di attuatori, perchè le altre gia sono state implementate dalla libreria interna di lejos.

- Motore singolo
- Navigazione

La gestione dei motori è stata suddivisa in due classi differenti, una per il motore singolo mentre l'altra per la navigazione a due motori (per navigazione si intende il movimento del robot con due motori che lavorano in sincronia).

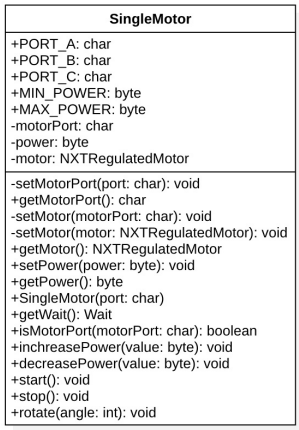
La gestione del display è gia implementata dalla classe `System.in`, mentre le funzioni audio sono implementate nella classe `lejos.nxt.Sound`.

Motore singolo

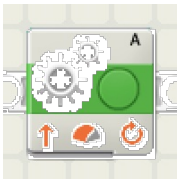


Il blocco motor (per il motore singolo) dell'ambiente di sviluppo Mindstorm NXT è rappresentato dalla classe `SingleMotor`, la quale permette di gestire facilmente un motore.

Il diagramma UML della classe:

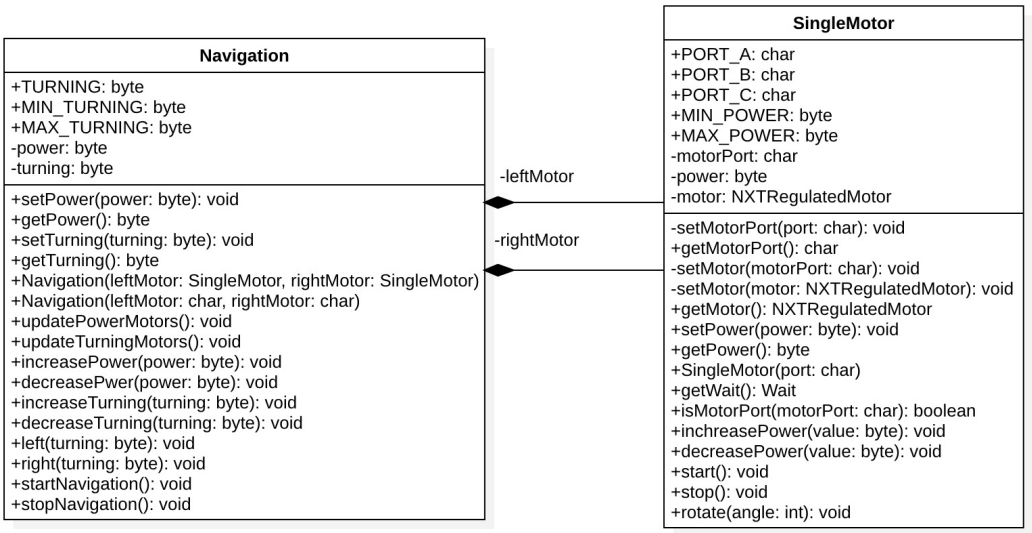


Navigazione



Il blocco motor (per la navigazione) dell'ambiente di sviluppo Mindstorm NXT è rappresentato dalla classe `Navigation`, la quale permette di manovrare con facilità i motori.

Il diagramma UML della classe:



Per poter manovrare i motori bisogna settare la velocità (con il metodo `setPower(power)`), mentre per manovrare la direzione bisogna usare il metodo `setTurning(turning)`. Poi bisogna avviare la navigazione con il metodo `startNaviagation()`, la quale può essere fermata con il metodo `stopNavigation()`.