

## **Sistema didattico per Lego EV3/NTX con libreria e documentazione**

## Sommario

1	Introduzione .....	3
1.1	Informazioni sul progetto .....	3
1.2	Abstract .....	3
1.3	Scopo .....	3
	Analisi .....	4
1.4	Analisi del dominio .....	4
1.5	Analisi e specifica dei requisiti .....	4
1.6	Pianificazione .....	6
1.7	Analisi dei mezzi .....	7
1.7.1	Software .....	7
1.7.2	Hardware .....	7
2	Progettazione .....	8
2.1	Design dell'architettura del sistema .....	8
3	Implementazione .....	9
3.1	Classi Wait .....	9
3.1.1	WaitMotor .....	9
3.1.2	WaitTime .....	9
3.1.3	WaitLightSensor .....	10
3.1.4	WaitSoundSensor .....	11
3.1.5	WaitUltrasonicSensor .....	12
3.1.6	WaitTouchSensor .....	12
3.2	Classi di test .....	13
3.2.1	SimpleMotor .....	13
4	Test .....	15
4.1	Protocollo di test .....	15
4.2	Risultati test .....	15
4.3	Mancanze/limitazioni conosciute .....	16
5	Consuntivo .....	17
6	Conclusioni .....	18
6.1	Sviluppi futuri .....	18
6.2	Considerazioni personali .....	18
7	Bibliografia .....	18
7.1	Sitografia .....	18
8	Allegati .....	18

## **1 Introduzione**

---

### **1.1 Informazioni sul progetto**

Progetto: Sistema didattico per Lego EV3/NTX con libreria e documentazione

Docente responsabile: Francesco Mussi, Luca Muggiasca, Adriano Barchi, Massimo Sartori

Componenti del gruppo: Andrea Rauso, Peter Catania

Luogo di lavoro: Aula 417 Scuola arti e mestieri Trevano

Classe: I3AC

Materia: Modulo 306

Data di Inizio: 14/11/2018

Data di fine: 08/02/2019

### **1.2 Abstract**

*If you want to develop a program for a Lego NXT robot, mostly beginner's programmers use the graphic Mindstorms tool. However, as the complexity of the programs increase, is observable that becomes less intuitive how arrange the graphics programming blocks. A possible solution of this problem is the embrace of a programming language, like Java or a more specific language for robots like RobotC. Introduce a programming language to beginner's is a step too high because of this we are offering a simpler way to embrace it. With our library, we offer a more understandable method of programming, having made available simple constructs integrated with the Java programming language.*

### **1.3 Scopo**

Lo scopo di questo progetto è di creare una semplice libreria per i robot Lego NXT, utilizzando il linguaggio di programmazione Java. In pratica offriamo dei costrutti simili ai blocchetti grafici presenti su Mindstorms per rendere l'esperienza di sviluppo più intuitiva e semplificata.

Questa libreria viene messa a disposizione degli allievi del secondo anno di informatica che nel corso del secondo semestre parteciperanno alla WRO.

## Analisi

### 1.4 Analisi del dominio

Oggigiorno molte persone che vogliono avvicinarsi alla programmazione utilizzando i Robot Lego NXT, utilizzano lo strumento grafico creato dalla Lego.

Con questo metodo la maggior parte dei programmi possono essere sviluppati con quasi nessuna difficoltà, ma se si vuole fare qualcosa di molto più complesso diventa molto meno intuitivo.

I principali utenti che si avvicinano alla programmazione con i robot della Lego sono principalmente i ragazzi che hanno iniziato i loro studi nell'ambito informatico.

### 1.5 Analisi e specifica dei requisiti

Il committente necessita di una libreria scritta con il linguaggio Java per il robot Lego NXT, per utilizzare java sul robot è necessario installare il firmware LeJOS che permette di lavorare con java.

La libreria deve implementare dei metodi di attesa per ogni sensore e attuatore facente parte del NXT, i metodi devono essere semplici per permettere un utilizzo facilitato agli allievi del secondo anno di Informatica.

Devono essere create 2 guide, la prima è una guida passo per passo per l'installazione del firmware LeJOS sul blocchetto NXT mentre la seconda è una guida dettagliata alle classi contenute nella libreria.

Come obiettivo supplementare il committente necessita di sapere quali sistemi operativi siano compatibili con il blocchetto Lego EV3

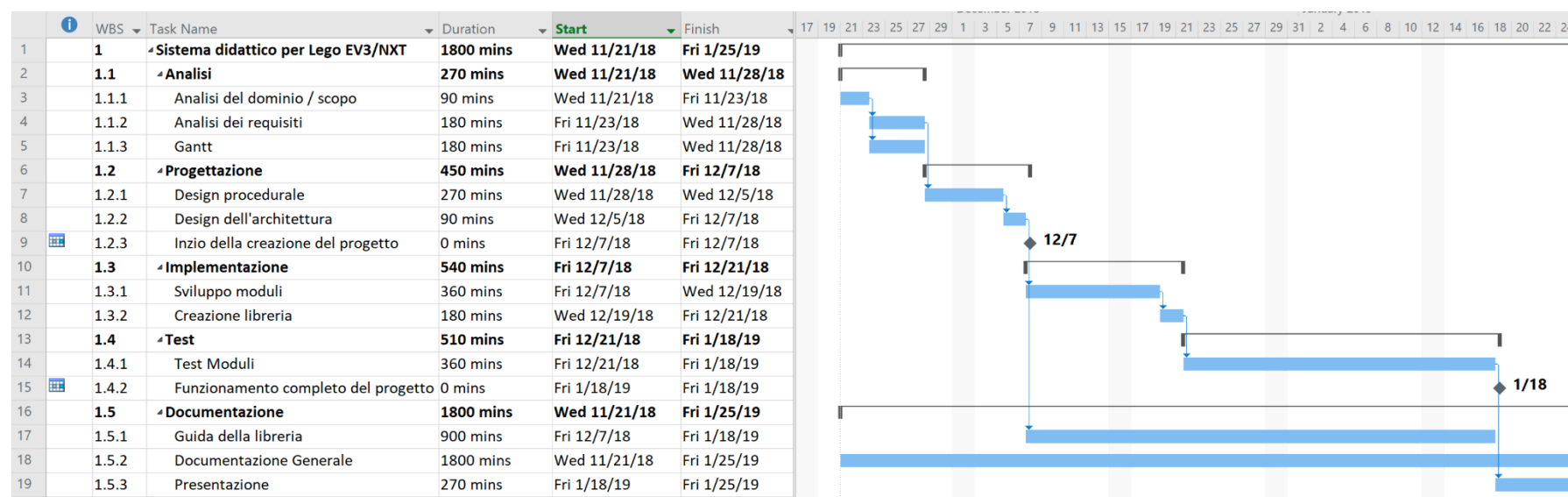
ID: REQ-001	
<b>Nome</b>	Creazione Libreria
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	-
Sotto requisiti	
<b>001</b>	Creazione di classi che replicano al funzionalità del blocchetto wait di Lego Mindstorm
<b>002</b>	Creazione della classe del Motore

ID: REQ-002	
<b>Nome</b>	Creazione Guide
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
Sotto requisiti	
<b>001</b>	Deve essere creata una guida per l'installazione di LeJOS
<b>002</b>	Deve essere creata una guida per l'utilizzo della libreria

ID: REQ-003	
<b>Nome</b>	Compatibilità del sistema EV3
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Compito supplementare che va a sostituire un progetto da creare con la libreria prodotta
Sotto requisiti	
<b>001</b>	Testare la compatibilità del Robot EV3 su diversi sistemi operativi

## 1.6 Pianificazione

Durante il progetto si fa come sempre una analisi, così da produrre anche i requisiti completi del progetto. Dopo facciamo la progettazione dove definiamo l'architettura delle classi, andando nel dettaglio come è composto il tutto e ogni singola parte. Poi passiamo all'implementazione dove creiamo la libreria partendo dai singoli moduli. Infine sulle librerie facciamo dei test che tutto funziona correttamente e fra di sé. Durante tutto il Progetto si terrà conto di tutto quello che abbiamo fatto, sia nei diari che nella documentazione, e bisognerà anche sviluppare delle guide, una per la libreria e una per l'installazione dei vari componenti.



## **1.7 Analisi dei mezzi**

### **1.7.1 Software**

I software che sono stati usati sono i seguenti:

- Eclipse (eclipse-committers-2018-09-win32-x86\_64)
- Plugin LeJOS per Eclipse
- StarUML 3.0.2
- LeJOS
- Microsoft Project
- Microsoft Word

### **1.7.2 Hardware**

L'hardware su cui è stato sviluppato questo sono i seguenti:

Portatile ASUS Vivobook Pro 15

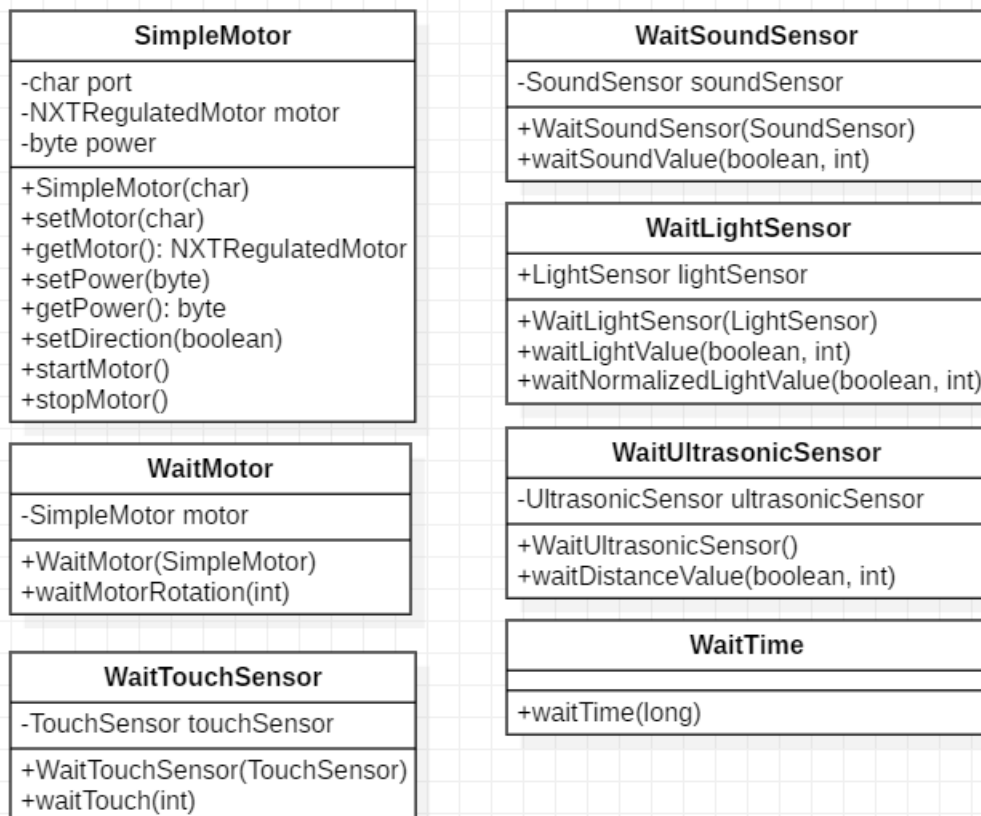
Portatile Apple MacBook Pro 15

Oltre al Robot Lego NXT sono stati utilizzati i seguenti sensori e attuatori: Sensore di tatto, Sensore di suono, Sensore Ultrasuoni, sensore di luce e il motore principale.

Questo progetto può girare su qualsiasi macchina che abbia almeno una porta USB per poter collegare il Robot Lego NXT, deve essere possibile installare il Driver Fantom che permette di interfacciarsi con il robot, facoltativamente deve anche essere possibile installare l'IDE Eclipse grazie al quale, tramite un plugin dedicato, è possibile creare direttamente i programmi per il Robot.

## 2 Progettazione

### 2.1 Design dell'architettura del sistema





### 3 Implementazione

#### 3.1 Classi Wait

All'interno di questa sezione sono elencati tutte le classi di attesa, una per ogni sensore e attuatore più il tempo, ogni classe comprende sempre il metodo di attesa e eventualmente il costruttore.

##### 3.1.1 WaitMotor

La classe WaitMotor è una classe che permette all'utente di aspettare che il robot compia determinate azioni con i motori, all'interno della classe è contenuto il costruttore che riceve un oggetto di tipo SimpleMotor, contiene i metodi waitRotation, waitDegrees e waitTime che rispettivamente aspettano le rotazioni, i gradi o il tempo del motore.

Implementazione:

```
public class WaitMotor {

    private SimpleMotor motor;

    public WaitMotor(SimpleMotor motor) {
        this.motor = motor;
    }

    public void waitMotorRotation(int rotation) {
        boolean finish = false;
        int previousRotation = this.motor.getMotor().getTachoCount();
        while(!finish) {
            try {
                if(previousRotation + rotation >
this.motor.getMotor().getTachoCount()) {
                    Thread.sleep(500);
                    finish = false;
                }else {
                    finish = true;
                }
            }catch(InterruptedException ie) {
            }
        }
    }
}
```

##### 3.1.2 WaitTime

La classe WaitTime aspetta un determinato lasso di tempo, all'interno della classe è contenuto il metodo waitTime che riceve un valore long di attesa e ricontrolla ogni 10 millisecondi il tempo passato.

Implementazione:

```
public class WaitTime {

    public void waitTime(long milliseconds) {
        long now = System.currentTimeMillis();
        while(now + milliseconds > System.currentTimeMillis()) {
            try {
                Thread.sleep(10);
            } catch (InterruptedException ie) {
                //Sleep interrupted
            }
        }
    }
}
```

### 3.1.3 WaitLightSensor

La classe WaitLightSensor è una classe che permette di aspettare un valore dal sensore di luce, all'interno della classe è contenuto un costruttore, il metodo getLightValue che riceve un valore int da trovare e un booleano per indicare se il valore deve essere maggiore o minore, contiene anche il metodo getNormalizedLightValue che rispetto al primo aspetta un valore normalizzato ( un valore compreso tra 0 e 1023).

Implementazione:

```
import lejos.nxt.*;

public class WaitLightSensor {

    private LightSensor lightSensor;

    public WaitLightSensor(LightSensor lightSensor) {
        this.lightSensor = lightSensor;
    }

    public void waitLightValue (boolean sign, int value) {
        boolean finish = false;
        while(!finish) {
            if(sign) {
                if(lightSensor.getLightValue() > value) {
                    finish = true;
                } else {
                    finish = false;
                }
            } else {
                if(lightSensor.getLightValue() < value) {
                    finish = true;
                } else {
                    finish = false;
                }
            }
        }
    }
}
```

```

public void waitNormalizedLightValue(boolean sign, int value) {
    boolean finish = false;
    while(!finish) {
        if(sign) {
            if(lightSensor.getNormalizedLightValue() > value) {
                finish = true;
            }else {
                finish = false;
            }
        }else{
            if(lightSensor.getNormalizedLightValue() < value) {
                finish = true;
            }else {
                finish = false;
            }
        }
    }
}
}
}

```

### 3.1.4 WaitSoundSensor

La classe WaitSoundSensor è una classe che permette di aspettare un valore dal sensore di suono, all'interno della classe è contenuto un costruttore, il metodo getSoundValue che riceve un valore int da trovare e un booleano per indicare se il valore deve essere maggiore o minore.

Implementazione:

```
import lejos.nxt.*;
```

```

public class WaitSoundSensor {

    private SoundSensor soundSensor;

    public WaitSoundSensor(SoundSensor soundSensor) {
        this.soundSensor = soundSensor;
    }

    public void waitSoundValue (boolean sign, int value) {
        boolean finish = false;
        while(!finish) {
            if(sign) {
                if(soundSensor.readValue() > value) {
                    finish = true;
                }else {
                    finish = false;
                }
            }else{
                if(soundSensor.readValue() < value) {
                    finish = true;
                }else {
                    finish = false;
                }
            }
        }
    }
}

```

### 3.1.5 WaitUltrasonicSensor

La classe WaitUltrasonicSensor è una classe che permette di aspettare una distanza dal sensore a ultrasuoni, all'interno della classe è contenuto un costruttore, il metodo getDistance che riceve un valore int da trovare e un booleano per indicare se il valore deve essere maggiore o minore.

Implementazione:

```
import lejos.nxt.*;

public class WaitUltrasonicSensor {

    private UltrasonicSensor ultrasonicSensor;

    public WaitUltrasonicSensor(UltrasonicSensor ultrasonicSensor) {
        this.ultrasonicSensor = ultrasonicSensor;
    }

    public void waitDistanceValue (boolean sign, int value) {
        boolean finish = false;
        while(!finish) {
            if(sign) {
                if(ultrasonicSensor.getDistance() > value) {
                    finish = true;
                }else {
                    finish= false;
                }
            }else{
                if(ultrasonicSensor.getDistance() < value) {
                    finish = true;
                }else {
                    finish = false;
                }
            }
        }
    }
}
```

### 3.1.6 WaitTouchSensor

La classe WaitTouchSensor è una classe che permette di aspettare un azione del sensore di tatto, all'interno della classe è contenuto il costruttore e il metodo waitTouch che riceve un int per indicare l'azione che deve aspettare (0 → premuto, 1 → rilasciato).

Implementazione:

```
import lejos.nxt.*;

public class WaitTouchSensor {

    private TouchSensor touchSensor;

    public WaitTouchSensor(TouchSensor touchSensor) {
        this.touchSensor = touchSensor;
    }
}
```

```

public void waitTouch(int mode) {
    boolean finish = false;
    boolean pressed = false;
    while(!finish) {
        if(mode == 0) {
            while(!pressed) {
                pressed = touchSensor.isPressed();
            }
            finish = true;
        } else {
            if(touchSensor.isPressed()) {
                pressed = true;
                while(pressed) {
                    pressed = touchSensor.isPressed();
                }
                finish = true;
            }
        }
    }
}

```

### 3.2 Classi di test

#### 3.2.1 SimpleMotor

La classe SimpleMotor ricrea la classe del motore contenuta nella libreria di LeJOS tenendo una versione semplificata. Contiene il costruttore a cui viene passata la porta a cui è collegato il motore e i metodi che permettono una gestione semplificata del motore.

Implementazione:

```

import lejos.nxt.*;

public class SimpleMotor {

    private char port;

    private NXTRegulatedMotor motor;

    private byte power;

    public SimpleMotor(char port) {
        if(port == 'A' || port == 'B' || port == 'C') {
            this.port = port;
            setMotor(this.port);
            this.power = 0;
        }
    }

    public void setMotor(char port) {
        if(port == 'A') {
            this.motor = Motor.A;
        } else if(port == 'B') {

```

```

        this.motor = Motor.B;
    }else {
        this.motor = Motor.C;
    }
}

public NXTRegulatedMotor getMotor() {
    return this.motor;
}

public void setPower(byte power){
    if(power >= 0 && power <= 255){
        this.power = power;
    }
}

public byte getPower(){
    return this.power;
}

public void setDirection(boolean direction) {
    if(direction) {
        this.getMotor().forward();
    }else {
        this.getMotor().backward();
    }
}

public void startMotor() {
    this.getMotor().setSpeed(this.getPower());
}

public void stopMotor() {
    this.getMotor().stop();
}
}

```

## 4 Test

### 4.1 Protocollo di test

<b>Test Case:</b>	TC-001	<b>Nome:</b>	Test delle classi della libreria
<b>Riferimento:</b>	REQ-001		
<b>Descrizione:</b>	Testare il buon funzionamento delle classi		
<b>Prerequisiti:</b>	Deve essere installato il firmware LeJOS sul robot, deve essere collegati i sensori necessari		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Implementare all'interno del progetto le classi della libreria</li> <li>2. Scrivere un codice che testi l'attesa dei valori da parte delle classi di Wait</li> </ol>		
<b>Risultati attesi:</b>	Le classi devono aspettare i valori corretti per poter proseguire		

<b>Test Case:</b>	TC-002	<b>Nome:</b>	Test comprensione e lettura guide
<b>Riferimento:</b>	REQ-003		
<b>Descrizione:</b>	Testare la comprensione e la leggibilità della guida		
<b>Prerequisiti:</b>	Devono essere state scritte le guide di installazione di LeJOS		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Lettura completa (dall'inizio alla fine) delle guide</li> <li>2. Vedere se i testi e le immagini siano leggibili e comprensibili</li> </ol>		
<b>Risultati attesi:</b>	Le guide devono essere leggibili e comprensibili.		

<b>Test Case:</b>	TC-003	<b>Nome:</b>	Test Compatibilità del robot MindStorms EV3
<b>Riferimento:</b>	REQ-003		
<b>Descrizione:</b>	Testare la compatibilità del blocchetto EV3		
<b>Prerequisiti:</b>	Deve essere installato il firmware LeJOS sul robot, deve essere collegato al computer tramite cavo USB		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Usare un sistema operativo diverso da Windows 10</li> <li>2. Installare il driver RNDIS sulla macchina</li> <li>3. Testare tramite ping, telnet o ssh tramite l'indirizzo 10.0.1.1 (l'indirizzo del robot) la connessione con il Robot</li> </ol>		
<b>Risultati attesi:</b>	Deve essere possibile collegarsi al robot o ricevere una sua risposta (tramite ping)		

### 4.2 Risultati test

Risultati:

TC-001	OK
TC-002	OK
TC-003	Fallito Sistemi operativi testati: Windows 7, Windows 8, MacOS 10.14 Mojave

#### **4.3 Mancanze/limitazioni conosciute**

Riguardo alla compatibilità del Robot Lego EV3, non siamo riusciti a trovare un sistema operativo che riuscisse a identificare il Robot. Crediamo che il problema risieda nel driver RNDIS, che permette la comunicazione tramite USB, ma che non è stato possibile installarlo per incompatibilità.

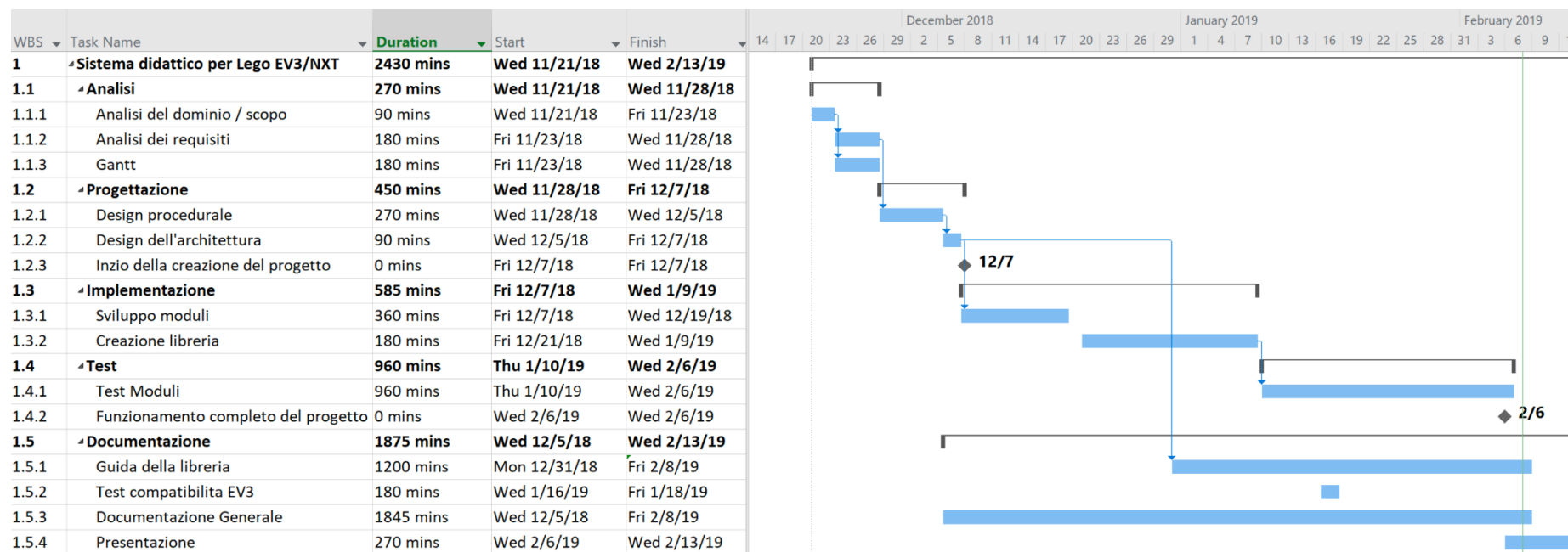


## 5 Consuntivo

Durante il progetto abbiamo avuto problemi di compatibilità con il robot Lego EV3, visto che il sistema operativo non lo riconosceva come dispositivo, così come ogni altro sistema operativo che abbiamo testato.

Riscontrato questo problema abbiamo dovuto concentrarci non più a sviluppare le librerie per EV3, ma quelle per NXT utilizzando comunque il linguaggio java.

La pianificazione non è stata rispettata soprattutto per il problema riscontrato, ma anche dal fatto che abbiamo avuto più tempo per svolgere il progetto.



## 6 Conclusioni

---

Grazie al nostro prodotto lo sviluppo di programmi per il Robot Lego NXT viene semplificato così da aiutare i giovani del secondo anno, che andranno a partecipare alla competizione WRO.

Con la nostra libreria gli studenti potranno utilizzarla anche per la creazione di programmi molto complessi, come anche i soliti che si fanno a lezione.

### 6.1 Sviluppi futuri

Come sviluppo futuro del nostro prodotto si possono aggiungere nuove classi, come ad esempio una classe che si occupi di gestire il giroscopio, oppure implementarne una che possa consentire di creare qualcosa di più complesso come un Line Follower o un Roomba.

### 6.2 Considerazioni personali

Da questo progetto abbiamo imparato cosa vuole dire collaborare in un progetto e quanto sia importante che ogni componente del gruppo sia puntuale nello svolgere le proprie mansioni.

Secondo noi la programmazione a oggetti è molto più comprensibile, è più facilmente utilizzabile per la creazione di programmi sia semplici ma soprattutto quelli più grandi, visto che serve una certa organizzazione, necessaria in questo caso.

## 7 Bibliografia

---

### 7.1 Sitografia

- <http://www.lejos.org/> LeJOS, Java for Lego MindStorms
- <http://www.lejos.org/ev3/docs/> Overview (LeJOS EV3 API Documentation)
- <https://lejos.sourceforge.io/forum/> LeJOS – Forum Index

## 8 Allegati

---

- A. Diari di Progetto
- B. Quaderno dei compiti
- C. Guida installazione LeJOS
- D. Guida Libreria
- E. Prodotto
  - a. Github: <https://github.com/andrearauso/Progetto-2>

Peter Catania

**Cannobio, 14.11.2018**

## Lavori svolti

Orario	Lavoro svolto
15:00 - 16:30	Ricevuto il diario dei compiti del progetto “Sistema didattico per Lego EV3/NTX con libreria e documentazione”.

## Problemi riscontrati e soluzioni adottate

--	--

## Punto della situazione rispetto alla pianificazione

--	--

## Programma di massima per la prossima giornata di lavoro

--	--

Peter Catania

**Cannobio, 16.11.2018**

## Lavori svolti

Orario	Lavoro svolto
15:00 - 16:30	Chiarimenti sui requisiti del progetto Lego. Istallato Lego Mindstorm

## Problemi riscontrati e soluzioni adottate

--	--

## Punto della situazione rispetto alla pianificazione

--	--

## Programma di massima per la prossima giornata di lavoro

--	--

Peter Catania

**Cannobio, 21.11.2018**

## Lavori svolti

Orario	Lavoro svolto
15:00 - 16:30	Abbiamo discusso sui requisiti del progetto, esposti sul quaderno dei compiti. Abbiamo cominciato l'analisi del progetto.

## Problemi riscontrati e soluzioni adottate

--	--

## Punto della situazione rispetto alla pianificazione

--	--

## Programma di massima per la prossima giornata di lavoro

--	--

Peter Catania

**Cannobio, 23.11.2018**

## Lavori svolti

Orario	Lavoro svolto
13:15 - 16:30	Abbiamo finito l'analisi dei requisiti. Abbiamo finito il gantt preventivo del progetto.

## Problemi riscontrati e soluzioni adottate

--	--

## Punto della situazione rispetto alla pianificazione

--	--

## Programma di massima per la prossima giornata di lavoro

	Cominciare la progettazione del progetto
--	--

Peter Catania

**Cannobio, 28.11.2018**

## Lavori svolti

Orario	Lavoro svolto
13:15 - 14:45	Siamo andati avanti con la progettazione, e abbiamo sistemato gli ambienti di sviluppo.

## Problemi riscontrati e soluzioni adottate

--	--

## Punto della situazione rispetto alla pianificazione

--	--

## Programma di massima per la prossima giornata di lavoro

	Finire la progettazione
--	-------------------------

Peter Catania

**Cannobio, 30.11.2018**

## Lavori svolti

Orario	Lavoro svolto
13:15 - 16:30	Abbiamo installato il software per programmare in java. Abbiamo testato se funzionavano i componenti NXT.

### Problemi riscontrati e soluzioni adottate

--	--

### Punto della soluzione rispetto alla pianificazione

	In orario
--	-----------

### Programma di massima per la prossima giornata di lavoro

	Cominciare a sviluppare il codice per i componenti
--	--



Peter Catania

**Cannobio, 5.12.2018**

## Lavori svolti

Orario	Lavoro svolto
13:15 - 16:30	Siamo andati avanti con la progettazione e abbiamo studiato un po come si farà l'implementazione.

### Problemi riscontrati e soluzioni adottate

--	--

### Punto della soluzione rispetto alla pianificazione

	In orario
--	-----------

### Programma di massima per la prossima giornata di lavoro

	Andare avanti con la progettazione
--	------------------------------------

Peter Catania

**Cannobio, 7.12.2018**

## Lavori svolti

Orario	Lavoro svolto
13:15 - 16:30	Siamo andati avanti con la progettazione.

### Problemi riscontrati e soluzioni adottate

--	--

### Punto della soluzione rispetto alla pianificazione

	In orario
--	-----------

### Programma di massima per la prossima giornata di lavoro

	Andare avanti con la progettazione
--	------------------------------------

Peter Catania

**Cannobio, 12.12.2018**

## Lavori svolti

Orario	Lavoro svolto
13:15 - 14:45	Installato i tool per lavorare con EV3, plugin di eclipse, installato leJOS per Mindstorm EV3

### Problemi riscontrati e soluzioni adottate

	Abbiamo avuto problemi nell'installazione del firmware LeJOS per l'EV3 a causa di una scheda microSD impostata in sola lettura, é stato risolto grazie ad un'altra schedaSD sostitutiva.
--	--

### Punto della soluzione rispetto alla pianificazione

	In orario
--	-----------

### Programma di massima per la prossima giornata di lavoro

	Cominciare la riscrittura del codice fatto per NXT in EV3 e continuare lo sviluppo. Modificare la documentazione in relazione al cambio di robot.
--	--

Peter Catania

Cannobio, 14.12.2018

## Lavori svolti

Orario	Lavoro svolto
13:15 - 14:45	Eliminato i tool vecchi per lavorare con NXT, Immeso firmware all'EV3.
15:00 - 16:30	Testato l'istallazione di lejos con Java 7 e 8 versione Embedded.

## Problemi riscontrati e soluzioni adottate

### **Problema:**

Non siamo riusciti ad installare il firmware lejos per EV3 a causa di un blocco continuo dell'installazione

/!\ Con alcune versioni della versione 8 di java non è possibile effettuare l'istallazione.

### **Soluzione:**

Abbiamo creato una partizione da 4 GB su una microSD da 32 GB in FAT32, grazie a questo siamo riusciti a installare il firmware sulla partizione creata.

## Punto della soluzione rispetto alla pianificazione

In orario

## Programma di massima per la prossima giornata di lavoro

Cominciare la riscrittura del codice fatto per NXT in EV3 e continuare lo sviluppo.  
Modificare la documentazione in relazione al cambio di robot.  
Cominciare la guida di installazione del firmware per EV3.



Peter Catania

**Canobbio, 14.12.2018**

## Lavori svolti

Orario	Lavoro svolto
13:15 - 14:45	Abbiamo iniziato a produrre la guida dell'installazione di LeJOS e NXT e quella riguardante la libreria sviluppata per NXT con java e LeJOS.

### Problemi riscontrati e soluzioni adottate

	Essendo che abbiamo cambiato il blocchetto EV3 con NXT, non abbiamo ancora ricevuto la nuova consegna dell'esercizio di esempio, da fare sfruttando la nostra libreria.
--	---

### Punto della soluzione rispetto alla pianificazione

	In ritardo
--	------------

### Programma di massima per la prossima giornata di lavoro

	Iniziare il programma d'esempio, che spiega come utilizzare la nostra libreria. Finire la guida dell'installazione di LeJOS e NXT
--	--

Peter Catania

**Canobbio, 18.01.2019**

## Lavori svolti

Orario	Lavoro svolto
13:15 - 14:45	Abbiamo testato il funzionamento su diversi sistemi operativi dell'applicativo per EV3 come parte del progetto. Abbiamo continuato le guide dell'installazione del firmware per NXT e della libreria.

### Problemi riscontrati e soluzioni adottate

Invece dell'esercizio di prova, ci è stato assegnato il compito di fare dei test per quanto riguarda il funzionamento e il collegamento del blocchetto EV3 su Sistemi operativi precedenti a Windows 10 e su Sistema operativo linux.  
Attualmente non abbiamo ancora trovato delle compatibilità con questi sistemi operativi.

### Punto della soluzione rispetto alla pianificazione

In orario

### Programma di massima per la prossima giornata di lavoro

Finire i test su i diversi sistemi operativi.  
Finire la guida dell'installazione di LeJOS su NXT.

Peter Catania

**Canobbio, 23.01.2019**

## Lavori svolti

Orario	Lavoro svolto
13:15 - 14:45	Abbiamo finito le guide dell'installazione del firmware per NXT e della libreria. Abbiamo finito i test su i diversi sistemi operativi.

### Problemi riscontrati e soluzioni adottate

	Attualmente non abbiamo ancora trovato delle compatibilità con questi sistemi operativi.
--	--

### Punto della soluzione rispetto alla pianificazione

	In orario
--	-----------

### Programma di massima per la prossima giornata di lavoro

	Cominciare la presentazione del progetto e continuare la documentazione di esso.
--	--



Peter Catania

**Canobbio, 25.01.2019**

## Lavori svolti

Orario	Lavoro svolto
13:15 - 16:30	Abbiamo continuato la documentazione e la presentazione del progetto.

### Problemi riscontrati e soluzioni adottate

--	--

### Punto della soluzione rispetto alla pianificazione

	In orario
--	-----------

### Programma di massima per la prossima giornata di lavoro

	Continuare la documentazione e la presentazione.
--	--

Peter Catania

**Canobbio, 30.01.2019**

## Lavori svolti

Orario	Lavoro svolto
13:15 - 16:30	Abbiamo continuato la documentazione.

### Problemi riscontrati e soluzioni adottate

-
---

### Punto della soluzione rispetto alla pianificazione

In orario
-----------

### Programma di massima per la prossima giornata di lavoro

Finire la documentazione
--------------------------

Peter Catania

**Canobbio, 01.02.2019**

## Lavori svolti

Orario	Lavoro svolto
13:15 - 16:30	Abbiamo continuato la documentazione.

### Problemi riscontrati e soluzioni adottate

-
---

### Punto della soluzione rispetto alla pianificazione

In orario
-----------

### Programma di massima per la prossima giornata di lavoro

Finire la documentazione
--------------------------

Peter Catania

**Canobbio, 06.02.2019**

## Lavori svolti

Orario	Lavoro svolto
13:15 - 16:30	Abbiamo continuato la documentazione. Abbiamo iniziato la presentazione.

### Problemi riscontrati e soluzioni adottate

-
---

### Punto della soluzione rispetto alla pianificazione

In orario
-----------

### Programma di massima per la prossima giornata di lavoro

Finire la documentazione e la presentazione. Consegnare la documentazione e i diari in un singolo PDF, con il link alla Repository del progetto
--

Peter Catania

**Canobbio, 08.02.2019**

## Lavori svolti

Orario	Lavoro svolto
13:15 - 16:30	Abbiamo finito la documentazione e la presentazione. Abbiamo sistemato le guide. Abbiamo Creato il file di consegna dei diari e della documentazione. E infine abbiamo consegnato il progetto tramite email.

### Problemi riscontrati e soluzioni adottate

--	--

### Punto della soluzione rispetto alla pianificazione

	In orario
--	-----------

### Programma di massima per la prossima giornata di lavoro

--	--

# Guida all'installazione di LeJOS

## Sommario

Requisiti .....	3
Installazione LeJOS .....	3
Installazione Fantom Driver.....	5
Comandi per il caricamento dei file.....	5
Installazione plugin eclipse.....	6

## Requisiti

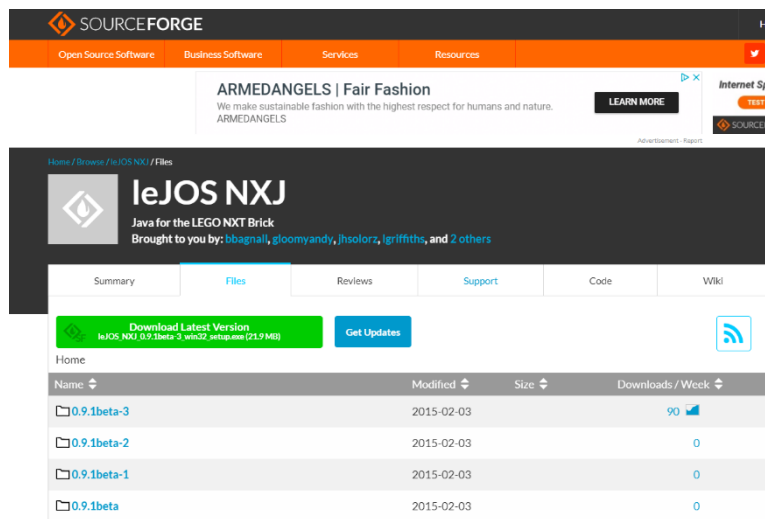
Per l'installazione di LeJOS è necessario

- Robot Lego NXT
- Software di installazione di LeJOS
- JDK Java 32-bit
- Driver Fantom
- Cavo USB fornito con il robot NXT

## Installazione LeJOS

L'installazione del firmware LeJOS sul blocchetto NXT è molto semplice.

Prima di tutto è necessario scaricare l'ultima versione del pacchetto LeJOS (in questo caso la versione 0.9.1) da questo indirizzo: <https://sourceforge.net/projects/nxt.lejos.p/files/>



SourceForge

Open Source Software Business Software Services Resources

ARMEDANGELS | Fair Fashion  
We make sustainable fashion with the highest respect for humans and nature.  
ARMEDANGELS

LEARN MORE

Internet Spe

TEST ME

Advertisement - Report

Home / Browse / leJOS NXJ / Files

**leJOS NXJ**  
Java for the LEGO NXT Brick  
Brought to you by: bbagnall, gloomyandy, hsolorz, griffiths, and 2 others

Summary Files Reviews Support Code Wiki

Download Latest Version  
leJOS NXJ 0.9.1beta-3, win32, setup.exe (21.5 MB)

Get Updates

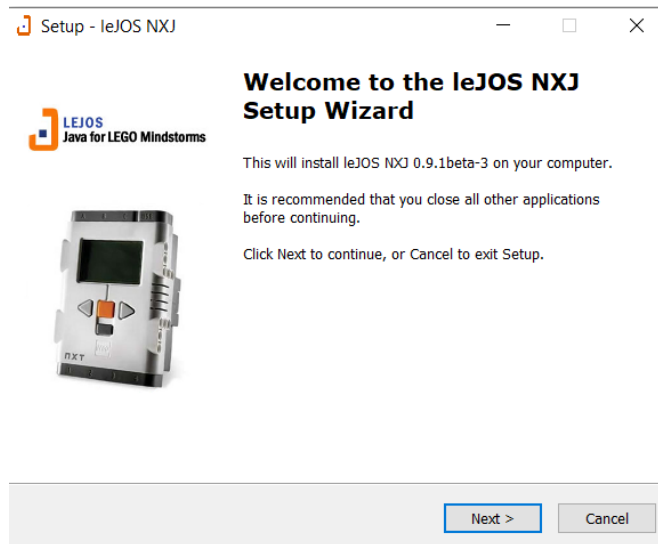
Home

Name	Modified	Size	Downloads / Week
0.9.1beta-3	2015-02-03		90
0.9.1beta-2	2015-02-03		0
0.9.1beta-1	2015-02-03		0
0.9.1beta	2015-02-03		0

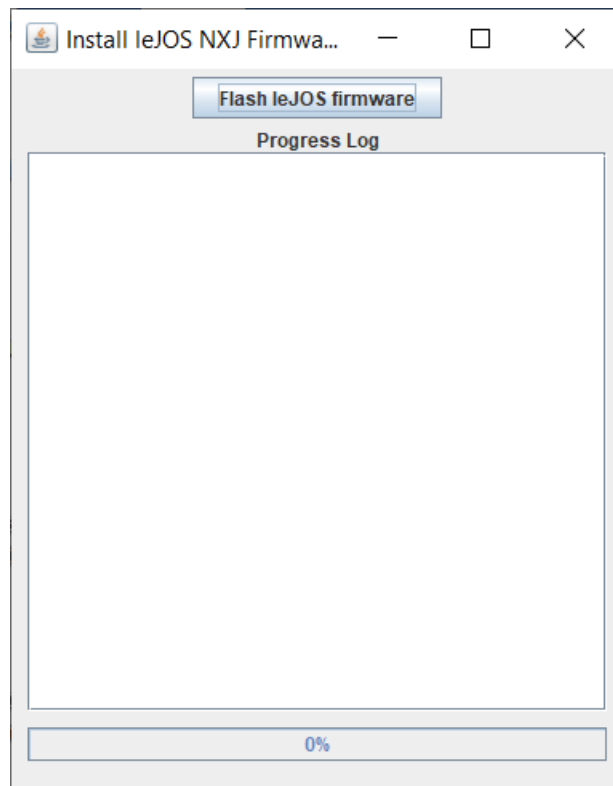
Scaricare anche la versione più aggiornata di Java nella versione a 32 bit, è consigliata la versione 1.7 ma è possibile utilizzare anche la versione 1.8.



Dopo aver scaricato il pacchetto avviate l'installazione di LeJOS sul vostro computer e accettate tutto quello che viene richiesto.



Appena viene completata l'installazione di LeJOS sul computer è possibile collegare tramite cavo USB il blocchetto NXT. Aprire l'applicativo NXJ Flash per immettere il firmware nel blocchetto, cliccare poi su "Flash LeJOS firmware" e attendere la fine dell'installazione.



## Installazione Fantom Driver

Per poter usare il robot Lego NXT è necessario installare il driver Fantom (usato per la comunicazione tra il blocchetto e il computer), che normalmente viene installato con il software grafico Lego Mindstorms.

Il link per scaricare il Driver è il seguente: <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>

Verrà scaricato un archivio zip contenente gli installer per Windows e MacOS.

## Comandi per il caricamento dei file

Per caricare i programmi sul blocchetto NXJ é necessario che questi vengano prima compilati con java e poi dal compilatore di LeJOS essendo che sfrutta librerie proprie per lo sviluppo del robot.

I comandi necessari sono:

- `Nxjc <java-files>` → compila il programma con il compilatore di Java
- `nxjlink -v -o HelloWorld.nxj HelloWorld` → viene creato un file nxj che fa il link a una determinata classe di Java. In questo viene fatto il link di una ipotetica classe HelloWorld
- `nxjupload <filename>.nxj` → permette l'upload del file creato sull'blocchetto
- `nxj <Nome classe java>` → permette di fare il link e l'upload di una classe di Java, corrisponde a fare nxjlink e nxjupload assieme

## Installazione plugin eclipse

Gli sviluppatori di LeJOS hanno creato un plugin per l'IDE Eclipse per permettere la progettazione e lo sviluppo dei programmi per NXT.

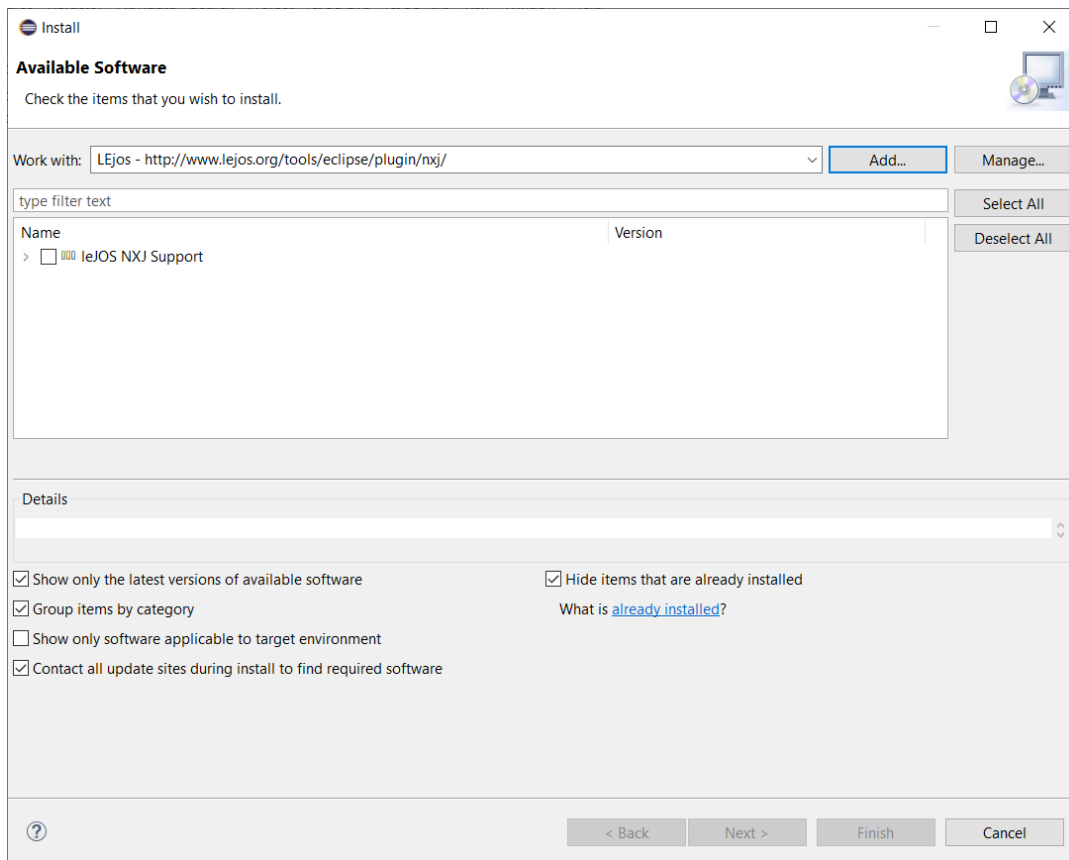
Per installarlo aprire l'IDE Eclipse, cliccare su "Help" e poi su Install New Software, si aprirà una finestra in cui è possibile installare dei software per Eclipse.

Per installare il plugin cliccare su "Add" e inserire il nome e l'URL del plugin.

Il link del plugin di LeJOS NXT è il seguente: <http://www.lejos.org/tools/eclipse/plugin/nxj/>

Il plugin ora appare nella lista, selezionatelo e poi cliccate "Next" per installarlo.

Ora è possibile, quando si crea un nuovo progetto, di includere e usare direttamente la libreria di LeJOS NXT.



## **Guida libreria LeJOS NXT**

**Titolo del progetto:** Guida libreria LeJOS NXT  
**Alunno/a:** Andrea Rauso, Peter Catania  
**Classe:** SAM I3AC  
**Anno scolastico:** 2018/2019  
**Docente responsabile:** Francesco Mussi, Luca Muggiasca, Adriano Barchi, Massimo Sartori

## Sommario

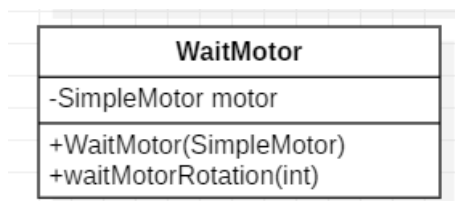
1	Classi di Wait.....	3
1.1	WaitMotor .....	3
1.2	WaitTime .....	3
1.3	WaitTouchSensor.....	3
1.4	WaitUltrasonicSensor.....	4
1.5	WaitSoundSensor .....	4
1.6	WaitLightSensor .....	5
2	Classi di Test.....	6
2.1	SimpleMotor .....	6

## 1 Classi di Wait

### 1.1 WaitMotor

La classe WaitMotor permette di aspettare un determinato numero di rotazioni del motore prima di eseguire altre operazioni.

La classe è strutturata nel seguente modo:



Attributi:

- SimpleMotor motor
  - Contiene il riferimento all'oggetto di tipo SimpleMotor

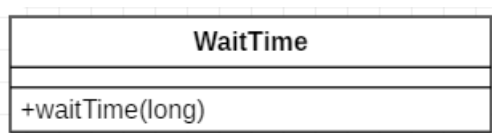
Metodi:

- public WaitMotor (SimpleMotor motor)
  - Costruttore, permette di istanziare un oggetto WaitMotor, per istanziarlo è necessario un oggetto SimpleMotor.
- Public void waitMotorRotation(int rotation)
  - Questo metodo permette di aspettare finché non sono stati effettuati un determinato numero di rotazioni del motore.
  - Il metodo ricontrolla ogni 500 millisecondi lo stato delle rotazioni

### 1.2 WaitTime

La classe WaitTime permette di aspettare un determinato lasso di tempo prima di eseguire altre operazioni.

La classe è strutturata nel seguente modo:



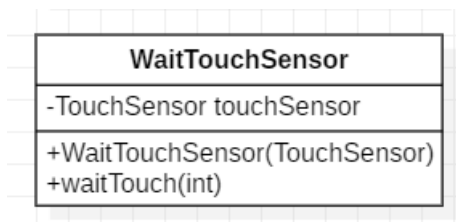
Metodi:

- waitTime (long milliseconds)
  - Questo metodo permette di far aspettare al programma un determinato lasso di tempo.

### 1.3 WaitTouchSensor

La classe WaitTouchSensor permette di aspettare che il sensore di tatto sia stato premuto o rilasciato.

La classe è strutturata nel seguente modo:



**Attributi:**

- TouchSensor touchSensor
  - Contiene il riferimento a un oggetto di tipo TouchSensor

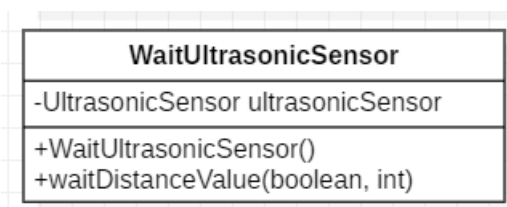
**Metodi:**

- WaitTouchSensor (TouchSensor touchSensor)
  - Costruttore, permette di istanziare un oggetto WaitTouchSensor, per istanziarlo è necessario un oggetto TouchSensor.
- waitTouch (int mode)
  - Questo metodo permette di aspettare finche il sensore di tatto non venga premuto o rilasciato, l'azione viene scelta dal parametro mode (0 → Premuto, 1→ Rilasciato)

## 1.4 WaitUltrasonicSensor

La classe WaitUltrasonicSensor permette di aspettare che il sensore a ultrasuoni vede una certa distanza prima di effettuare altre operazioni.

La classe è strutturata nel seguente modo:



**Attributi:**

- UltrasonicSensor ultrasonicSensor
  - Contiene il riferimento a un oggetto di tipo UltrasonicSensor

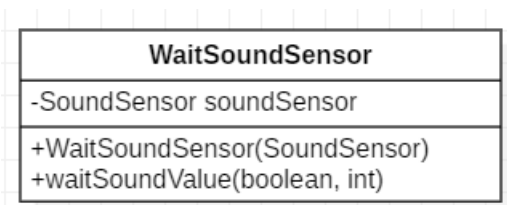
**Metodi:**

- WaitUltrasonicSensor (UltrasonicSensor ultrasonicSensor)
  - Costruttore, permette di istanziare un oggetto WaitUltrasonicSensor, per istanziarlo è necessario un oggetto UltrasonicSensor.
- waitDistanceValue (Boolean sign, int value)
  - Questo metodo permette di aspettare una determinata distanza prima di effettuare altre operazioni, il parametro sign determina se la soglia deve essere maggiore o minore rispetto al valore value.

## 1.5 WaitSoundSensor

La classe WaitSoundSensor permette di aspettare che il sensore di suono riceva una certa soglia di rumore prima di effettuare altre operazioni.

La classe è strutturata nel seguente modo:



**Attributi:**

- SoundSensor soundSensor
  - Contiene il riferimento a un oggetto di tipo SoundSensor

**Metodi:**

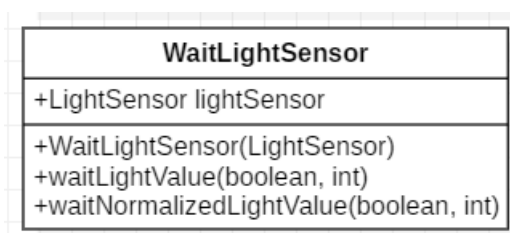
- WaitSoundSensor (SoundSensor soundSensor)

- Costruttore, permette di istanziare un oggetto WaitSoundSensor, per istanziarlo è necessario un oggetto SoundSensor.
- waitSoundValue (Boolean sign, int value)
  - Questo metodo permette di aspettare un determinato valore del sensore di suono prima di effettuare altre operazioni, il parametro sign determina se la soglia deve essere maggiore o minore rispetto al valore value.

## 1.6 WaitLightSensor

La classe WaitLightSensor permette di aspettare che il sensore di luce vede una certa soglia di luce prima di effettuare altre operazioni.

La classe è strutturata nel seguente modo:



Attributi:

- LightSensor lightSensor
  - Contiene il riferimento a un oggetto di tipo LightSensor.

Metodi:

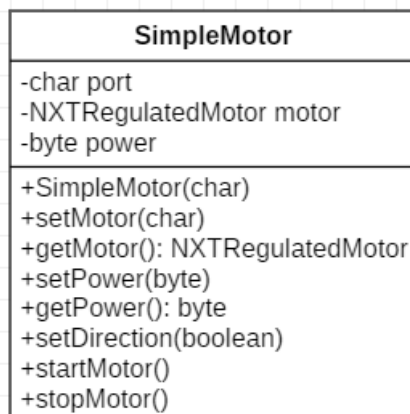
- WaitLightSensor (LightSensor lightSensor)
  - Costruttore, permette di istanziare un oggetto WaitLightSensor, per istanziarlo è necessario un oggetto LightSensor.
- waitLightValue(boolean sign, int value)
  - Questo metodo permette di aspettare un determinato valore del sensore di luce prima di effettuare altre operazioni, il parametro sign determina se la soglia deve essere maggiore o minore rispetto al valore value.
- waitNormalizedLightValue(boolean sign, int value)
  - Questo metodo permette di aspettare un determinato valore del sensore di luce, rispetto al metodo precedente aspetta il valore normalizzato (un valore da 0 a 1023), il parametro sign determina se la soglia deve essere maggiore o minore rispetto al valore value.



## 2 Classi di Test

### 2.1 SimpleMotor

La classe SimpleMotor è una classe che permette di gestire un singolo motore.  
La classe è strutturata nel modo seguente:



Attributi:

- Char port
  - La porta a cui è stato attaccato il motore
- NXTRegulatedMotor motor
  - Riferimento al Motore della classe NXTRegulatedMotor
- Byte power
  - La potenza del motore

Metodi:

- SimpleMotor (char port)
  - Costruttore, permette di istanziare un oggetto di tipo SimpleMotor, per istanziarlo è necessario indicare la porta a cui è collegato
- setMotor(char port)
  - Setta il motore in base alla porta indicata
- getMotor()
  - Ritorna il motore della classe NXTRegulatedMotor
- setPower(byte power)
  - Setta il valore della potenza del motore
- getPower()
  - Ritorna la potenza del motore
- setDirection(boolean direction)
  - Setta la direzione (forward o backward, avanti o indietro) in base al valore passato
- startMotor()
  - Permette di far partire il motore.
- stopMotor()
  - Permette di fermare il motore.