GUIDA

Guida per l’uso semplificato di lejOS con Lego NXT

Sommario

[Installazione dell’ambiente per l’uso della libreria 3](#_Toc356485)

[Installazione ambiente Java 3](#_Toc356486)

[Installazione Fantom Driver 6](#_Toc356487)

[Installazione lejOS 6](#_Toc356488)

[Classi 7](#_Toc356489)

[WaitMotor 7](#_Toc356490)

[Navigator 7](#_Toc356491)

[WaitTime 7](#_Toc356492)

[WaitTouchSensor 7](#_Toc356493)

[WaitColorSensor 8](#_Toc356494)

[WaitUltrasonicSensor 8](#_Toc356495)

[WaitSoundSensor 9](#_Toc356496)

[WaitLightSensor 9](#_Toc356497)

# Installazione dell’ambiente per l’uso della libreria

Per poter usare nel modo corretto questa libreria bisogna installare Java, il Fantom Driver e lejOS che ci permetteranno di far funzionare il tutto una volta messi assieme.

## Installazione ambiente Java

Java è l’ambiente su cui è basato lejOS, di conseguenza lo necessitiamo per fare funzionare i nostri programmi, l’installazione è relativamente semplice.

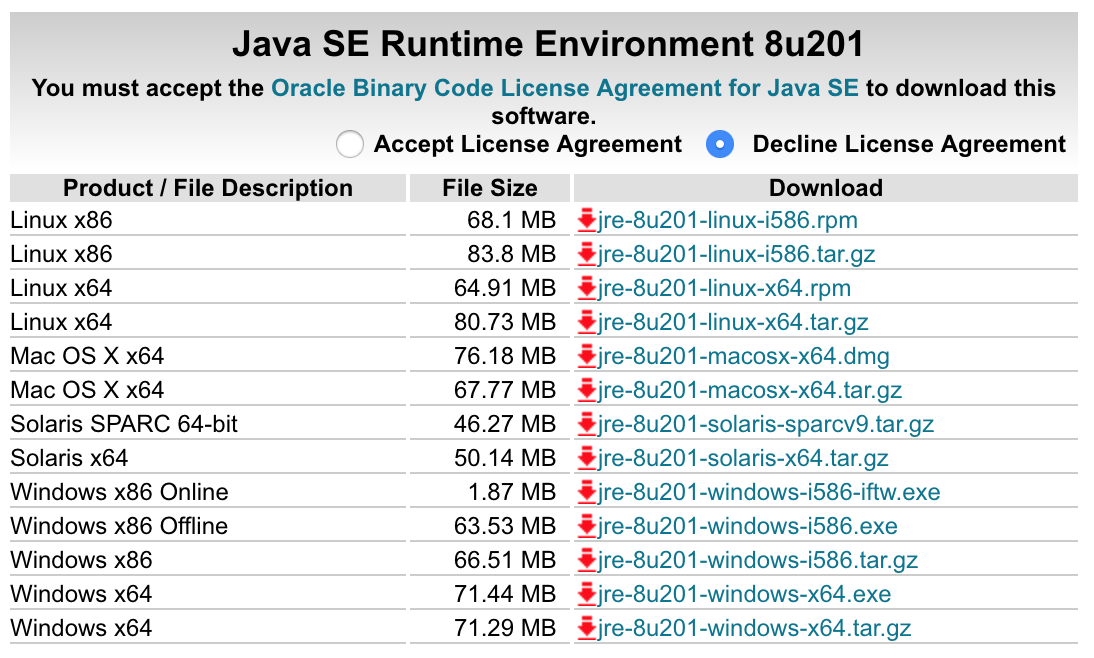
***Nota****: Bisogna scaricare Java solamente per pc Windows, su Mac è già installato di default.*

Bisogna scaricare la JDK e la JRE che sono i due componenti principali di Java che troviamo nei seguenti posti:

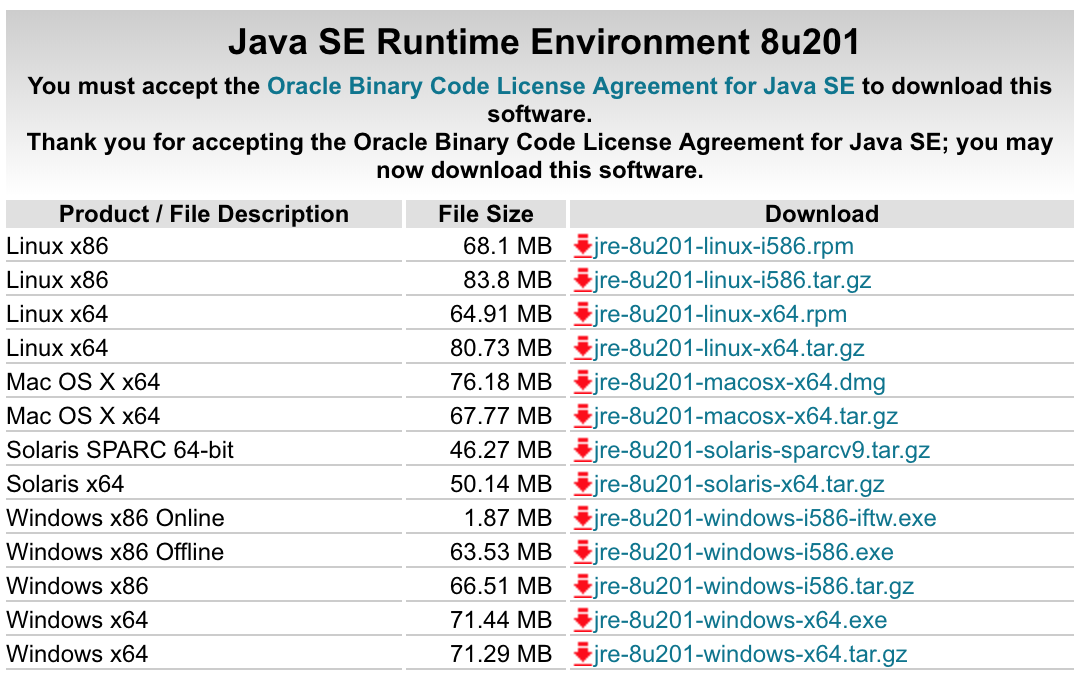
JDK: https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

JRE: https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

Per entrambi le procedure di installazione sono le stesse, bisogna andare sui due siti e cercare la versione corretta, nel nostro caso la “**Java SE Runtime Environment 8u201**”; Dovrebbe apparirci questa pagina:

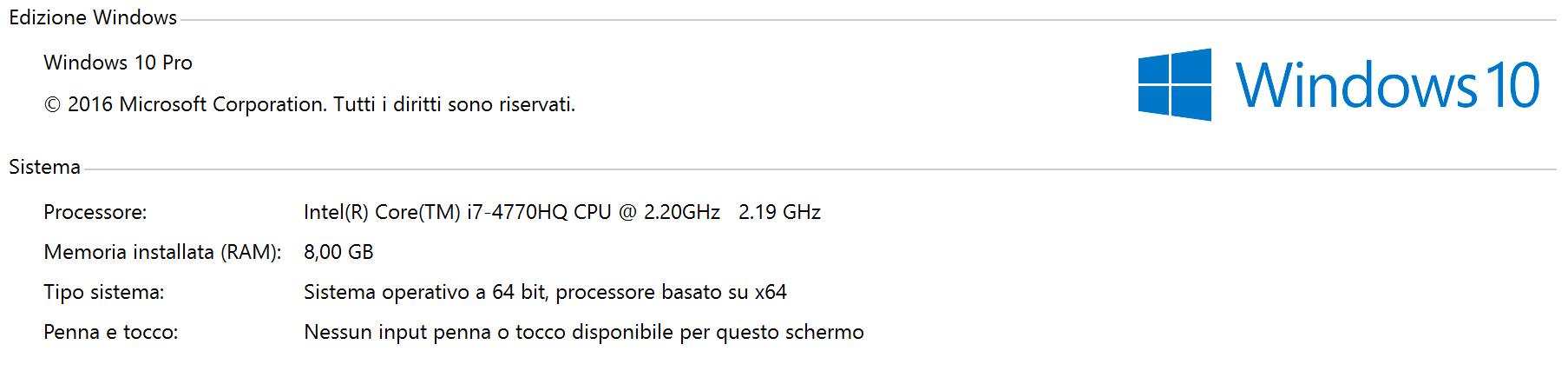


Successivamente per poter scaricare la versione dobbiamo accettare il contratto della licenza cliccando su “Accept License Agreement” in alto a sinistra. Successivamente la pagina dovrebbe essere la seguente:



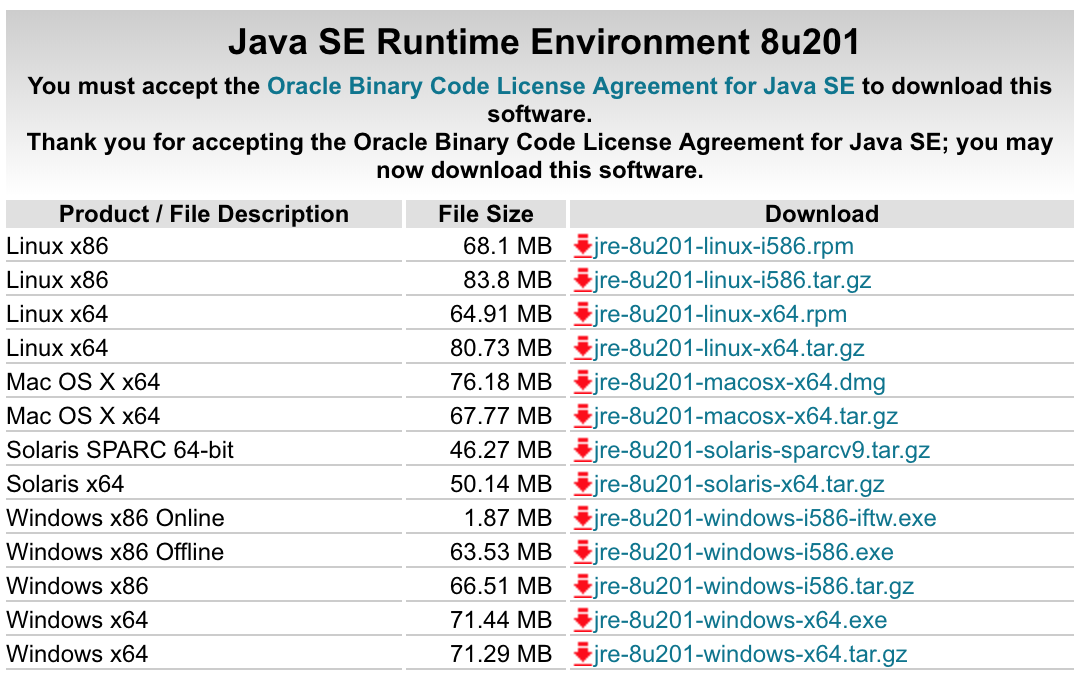
Come prossimo passo dobbiamo conoscere l’architettura del nostro sistema operativo, se il pc è un Windows basta andare nello start, scrivere “Questo PC” e cliccare il tasto destro andando su “Proprietà”.

Appare una finestra con svariati dettagli sul proprio pc fra di cui l’architettura del sistema:



Per procedere al download basta cliccare sui link blu che stanno a destra della dimensione della versione e, conoscendo la propria versione del sistema operativo scegliere quella corretta.

Esempio per la versione di Windows 10 64-bit:



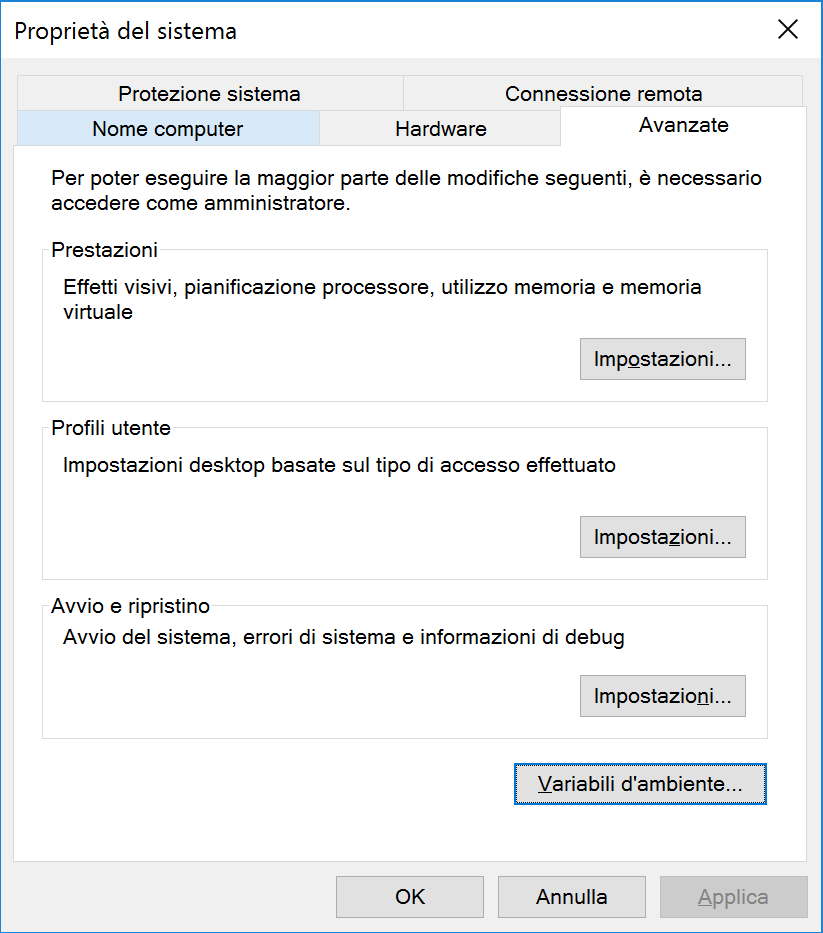
Fare questa procedura sia per la JDK che per la JRE.

Una volta scaricati entrambi, basta eseguirle i loro installer cliccando sempre su “Avanti”.

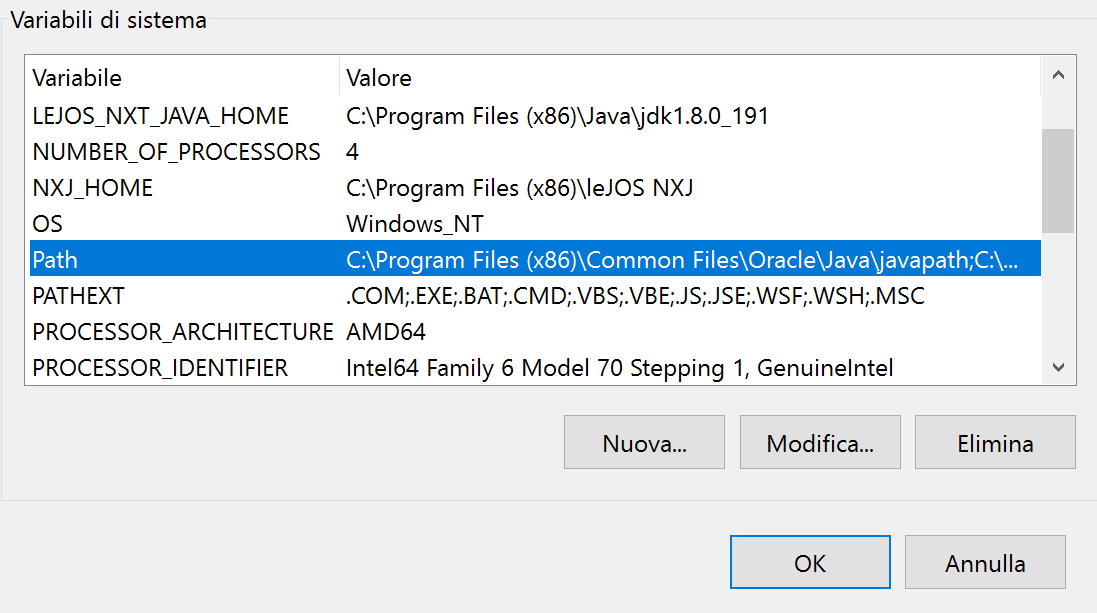


Come ultimo passo dobbiamo impostare la variabile d’ambiente del sistema facendo le seguenti operazioni:

Scriviamo “Modifica le variabili di ambiente del sistema” nello start e clicchiamo sull’icona che appare, successivamente bisogna andare su “**Variabili d’ambiente**”



Poi si deve andare sotto “**Path**” e cliccare “**Modifica**”



Infine basta cliccare il bottone “Nuovo” e immettere il percorso della cartella bin della JDK come ad esempio: **C:\Program Files\Java\jdk1.8.0\_191\bin**.

## Installazione Fantom Driver

Il Fantom Driver serve a…

Si scarica dove…

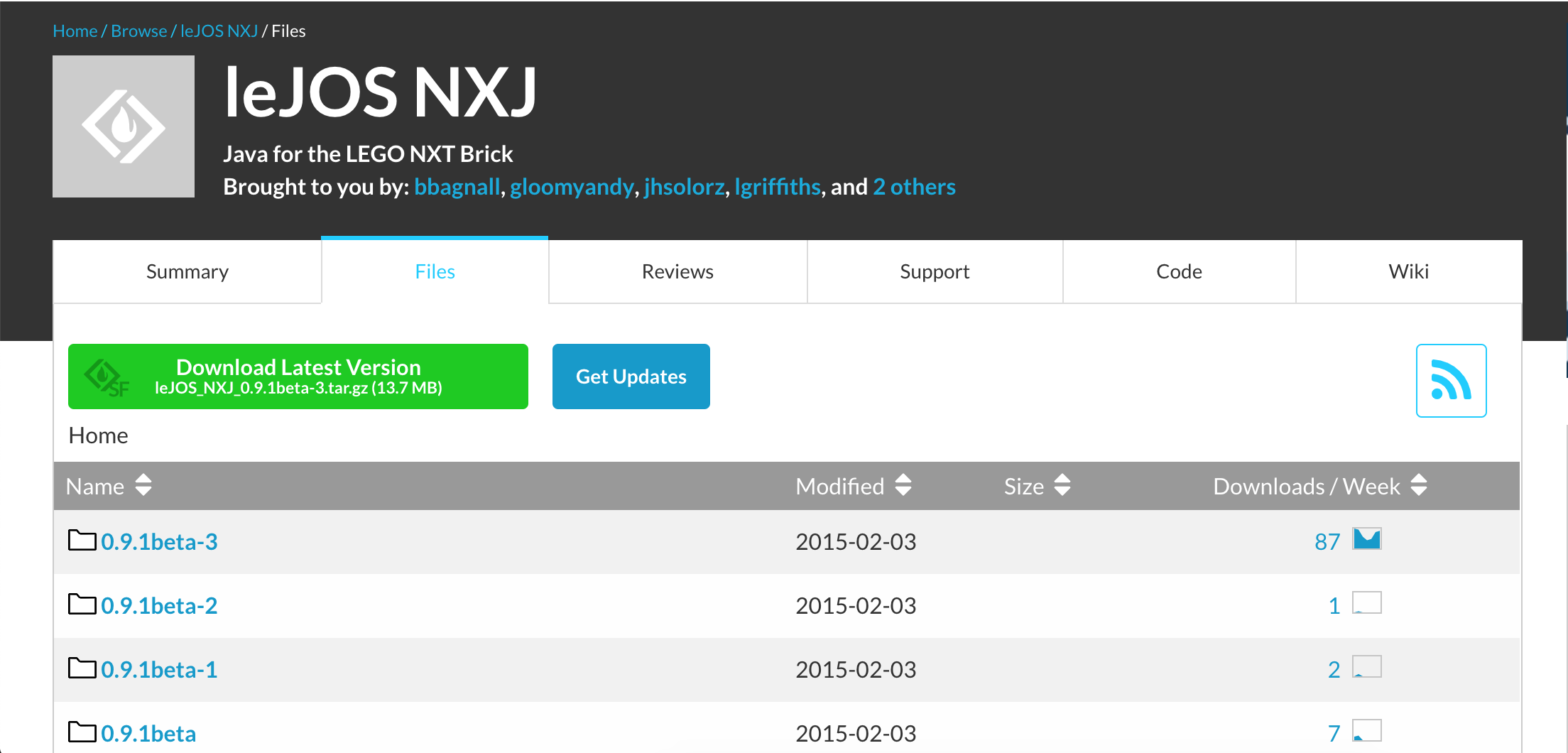
## Installazione lejOS

LejOS è un insieme di classi scritte in Java che ci permette di controllare i robot di tipo NXT ed EV3(NXT nel nostro caso).

Per scaricarlo basta andare sul suo sito ufficiale di download:

<https://sourceforge.net/projects/nxt.lejos.p/files/>

Successivamente dobbiamo scaricare la versione più recente, quindi la 0.9.1-3.



Ed infine scaricare la versione .exe se siamo su Windows oppure la .zip se si è su Mac.

Una volta scaricato iniziamo ad eseguire l’installer nel quale dovremo solamente cliccare continuamente “Avanti”.

# Classi

## Navigation

Navigation è la classe che si occupa di gestire l’uso dei motori con tutti i suoi rispettivi metodi.

Per implementare questa classe si usa:

*Navigation navigator = new Navigation(LEFT\_MOTOR\_PORT, RIGHT\_MOTOR\_PORT) ;*

I metodi che contiene sono i seguenti :

|  |  |
| --- | --- |
| Navigation(char Left\_Motor, char Right\_Motor) ; | Cosutruttore della classe che riceve come parametri la porta sinistra e quella destra dei due motori.  *Navigation(‘A’, ‘B’);* |
| char getLeftMotorPort() | Ritorna la porta del motore sinistro. |
| char getRightMotorPort() | Ritorna la porta del motore destro. |
| void setMotorLeftPort(char port) |  |
| void setMotorRightPort(char port) |  |
| int getMySpeed() |  |
| void setMySpeed(int speed) |  |
| int getSteering() |  |
| void setSteering(int steering) |  |
| char getDirection() |  |
| void setDirection(char direction) |  |
| void move() |  |
| void left(int howMuch) |  |
| void right(int howMuch) |  |
| void stop() |  |

## WaitTime

WaitTime è una classe che permette di aspettare del tempo in millisecondi.

Per implementare questa classe si usa:

*WaitTime wt = new WaitTime();*

Questa classe contiene tre metodi che possono essere usati dal utente:

|  |  |
| --- | --- |
| void myWait(int time); | Per usare questo metodo si usa:  *wt.myWait(2000);*  In questo caso il programma aspetta per 2 secondi e poi continua. |
| void setStartTime(); | Questo metodo serve per il metodo isFinished(int time). |
| boolean isFinished(int time); | Per usare questo metodo si usa:  *setStartTime();*  *while(wt.isFinished(2000)){*  *//codice da eseguire per 2 secondi*  *}*  Questo metodo esegue il codice nel while per 2 secondi. |

## WaitTouchSensor

WaitTouchSensor è una classe che permette di aspettare che un sensore di tatto venga premuto.

Per implementare questa classe si usa:

*WaitTouchSensor wts = new WaitTouchSensor(new TouchSensor(SensorPort.S3));*

In questa classe c’è int action che definisce l’azione per terminare l’attesa secondo questi criteri:

0 Premuto

1 Rilasciato

2 Cliccato(Premuto rilasciato)

Questa classe contiene due metodi che possono essere usati dal utente:

|  |  |
| --- | --- |
| void myWait(int action); | Per usare questo metodo si usa:  *wts.myWait(0);*  In questo caso il programma aspetta che il pulsante venga premuto e poi continua. |
| boolean isFinished(int action); | Per usare questo metodo si usa:  *while(wts.isFinished(0)){*  *//codice da eseguire finché non premuto*  *}*  Questo metodo esegue il codice nel while finché non viene preuto il pulsante. |

## WaitColorSensor

WaitColorSensor è una classe che permette di aspettare che un sensore di colore veda un certo colore.

Per implementare questa classe si usa:

*WaitColorSensor wcs = new WaitColorSensor(new ColorSensor(SensorPort.S3));*

Questa classe contiene due metodi che possono essere usati dall’utente:

|  |  |
| --- | --- |
| void myWait(int red, int green, int blue); | Per usare questo metodo si usa:  *wcs.myWait(255,0,0);*  In questo caso il programma aspetta che il sensore di colore vede rosso. |
| boolean isFinished(int red, int green, int blue); | Per usare questo metodo si usa:  *while(wcs.isFinished(255,0,0)){*  *//codice da eseguire finché non vede rosso*  *}*  Questo metodo esegue il codice nel while finché il sensore non vede rosso. |

## WaitUltrasonicSensor

WaitUltrasonicSensor è una classe che permette di aspettare che il sensore di ultrasuoni non veda una certa distanza.

Per implementare questa classe si usa:

*WaitUltrasonicSensor wus = new WaitUltrasonicSensor(new UltrasonicSensor(SensorPort.S3));*

Questa classe contiene due metodi che possono essere usati dall’utente:

|  |  |
| --- | --- |
| void myWait(Boolean bigger, int value); | Per usare questo metodo si usa:  *wus.myWait(false, 40);*  In questo caso il programma aspetta che il sensore di utrasuoni è a 40 cm di distanza da un oggetto poi continua. |
| boolean isFinished(Boolean bigger, int value); | Per usare questo metodo si usa:  *while(wus.isFinished(false, 40)){*  *//codice da eseguire finché non vede 40cm*  *}*  Questo metodo esegue il codice nel while finché il sensore di utrasuoni non è a 40 cm di distanza da un oggetto. |

## WaitSoundSensor

WaitSoundSensor è una classe che permette di aspettare che il sensore del suono non sente un rumore di una certa potenza/debolezza.

Per implementare questa classe si usa:

*WaitSoundSensor wus = new WaitSoundSensor(new SoundSensor(SensorPort.S3));*

Questa classe contiene due metodi che possono essere usati dall’utente:

|  |  |
| --- | --- |
| void myWait(Boolean bigger, int value); | Per usare questo metodo si usa:  *wss.myWait(false, 40);*  In questo caso il programma aspetta che il sensore del suono sente un suono di potenza 40 poi continua. |
| boolean isFinished(Boolean bigger, int value); | Per usare questo metodo si usa:  *while(wss.isFinished(false, 40)){*  *//codice da eseguire finché non sente meno di 40*  *}*  Questo metodo esegue il codice nel while finché il sensore del suono sente un suono di potenza 40. |

## WaitLightSensor

WaitLightSensor è una classe che permette di aspettare che il sensore di luce non vede una luminosità maggiore/minore di un valore.

Per implementare questa classe si usa:

*WaitLightSensor wls = new WaitLightSensor(new LightSensor(SensorPort.S3));*

Questa classe contiene due metodi che possono essere usati dall’utente:

|  |  |
| --- | --- |
| void myWait(Boolean bigger, int value); | Per usare questo metodo si usa:  *wls.myWait(false, 40);*  In questo caso il programma aspetta che il sensore di luce non vede luminosità di 40 poi continua. |
| boolean isFinished(Boolean bigger, int value); | Per usare questo metodo si usa:  *while(wls.isFinished(false, 40)){*  *//codice da eseguire finché non vede 40*  *}*  Questo metodo esegue il codice nel while finché il sensore di luce non vede una luminosità di 40. |