Sistema didattico per Lego EV3/NTX con libreria e documentazione

Sommario

[1 Introduzione 3](#_Toc554544)

[1.1 Informazioni sul progetto 3](#_Toc554545)

[1.2 Abstract 3](#_Toc554546)

[1.3 Scopo 3](#_Toc554547)

[Analisi 4](#_Toc554548)

[1.4 Analisi del dominio 4](#_Toc554549)

[1.5 Analisi e specifica dei requisiti 4](#_Toc554550)

[1.6 Pianificazione 6](#_Toc554551)

[1.7 Analisi dei mezzi 7](#_Toc554552)

[1.7.1 Software 7](#_Toc554553)

[1.7.2 Hardware 7](#_Toc554554)

[2 Progettazione 8](#_Toc554555)

[2.1 Design dell’architettura del sistema 8](#_Toc554556)

[3 Implementazione 9](#_Toc554557)

[3.1 Classi Wait 9](#_Toc554558)

[3.1.1 WaitMotor 9](#_Toc554559)

[3.1.2 WaitTime 9](#_Toc554560)

[3.1.3 WaitLightSensor 10](#_Toc554561)

[3.1.4 WaitSoundSensor 11](#_Toc554562)

[3.1.5 WaitUltrasonicSensor 12](#_Toc554563)

[3.1.6 WaitTouchSensor 12](#_Toc554564)

[3.2 Classi di test 13](#_Toc554565)

[3.2.1 SimpleMotor 13](#_Toc554566)

[4 Test 15](#_Toc554567)

[4.1 Protocollo di test 15](#_Toc554568)

[4.2 Risultati test 15](#_Toc554569)

[4.3 Mancanze/limitazioni conosciute 16](#_Toc554570)

[5 Consuntivo 17](#_Toc554571)

[6 Conclusioni 18](#_Toc554572)

[6.1 Sviluppi futuri 18](#_Toc554573)

[6.2 Considerazioni personali 18](#_Toc554574)

[7 Bibliografia 18](#_Toc554575)

[7.1 Sitografia 18](#_Toc554576)

[8 Allegati 18](#_Toc554577)

# Introduzione

## 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

## 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.*

## 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

## 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 voule 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.

## 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 obbiettivo supplementare il committente necessita di sapere quali sistemi operativi siano compatibili con il blocchetto Lego EV3

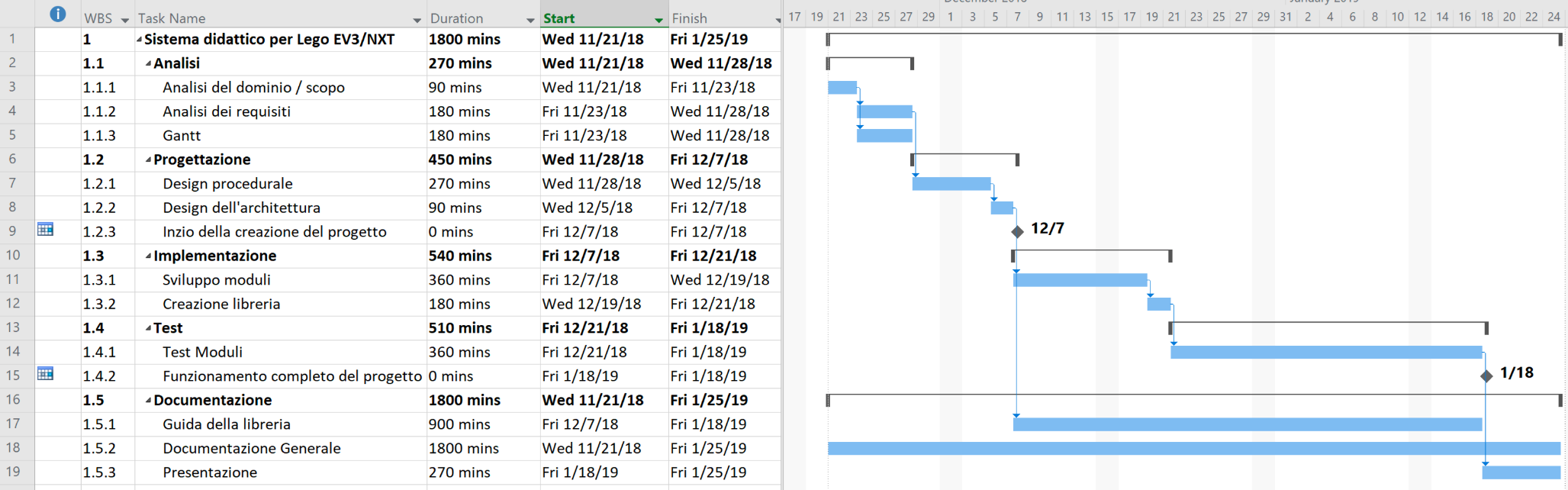
|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Creazione Libreria |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | Creazione di classi che replicano al funzionalità del blocchetto wait di Lego Mindstorm |
| **002** | Creazione della classe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

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Creazione Guide |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Deve essere creata una guida per l’installazione di LeJOS |
| **002** | Deve essere creata una guida per l’utilizzo della libreria |

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

## Pianificazione

Durante il progetto si fa come sempre una analisi, cosi 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 terra 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.



## Analisi dei mezzi

### 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

### Hardware

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

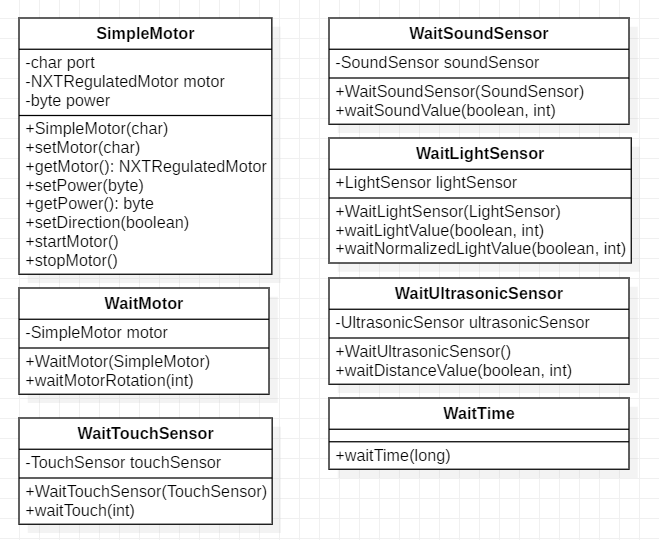
Portatile ASUS Vivobook Pro 15  
Portatile Apple MacBook Pro 15

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.

# Progettazione

## Design dell’architettura del sistema



# Implementazione

## 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.

### 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) {

}

}

}

}

### 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

}

}

}

}

### 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**;

}

}

}

}

}

### 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**;

}

}

}

}

}

### 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**;

}

}

}

}

}

### 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**;

}

}

}

}

}

## Classi di test

### 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();

}

}

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Test delle classi della libreria |
| **Descrizione:** | Testare il buon funzionamento delle classi | | |
| **Prerequisiti:** | Deve essere installato il firmware LeJOS sul robot, deve essere collegati i sensori necesssari | | |
| **Procedura:** | 1. Implementare all’interno del progetto le classi della libreria 2. Scrivere un codice che testi l’attesa dei valori da parte delle classi di Wait | | |
| **Risultati attesi:** | Le classi devono aspettare i valori corretti per poter proseguire | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-003 | **Nome:** | Test comprensione e lettura guide |
| **Descrizione:** | Testare la comprensione e la leggibilità della guida | | |
| **Prerequisiti:** | Devono essere state scritte le guide di installazione di LeJOS | | |
| **Procedura:** | 1. Lettura completa (dall’inizio alla fine) delle guide 2. Vedere se i testi e le immagini siano leggibili e comprensibili | | |
| **Risultati attesi:** | Le guide devono essere leggibili e comprensibili. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-003 | **Nome:** | Test Compatibilità del robot MindStorms EV3 |
| **Descrizione:** | Testare la compatibilità del blocchetto EV3 | | |
| **Prerequisiti:** | Deve essere installato il firmware LeJOS sul robot, deve essere collegato al computer tramite cavo USB | | |
| **Procedura:** | 1. Usare un sistema operativo diverso da Windows 10 2. Installare il driver RNDIS sulla macchina 3. Testare tramite ping, telnet o ssh tramite l’indirizzo 10.0.1.1 (l’indirizzo del robot) la connessione con il Robot | | |
| **Risultati attesi:** | Deve essere possibile collegarsi al robot o ricevere una sua risposta (tramite ping) | | |

## Risultati test

Risultati:

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

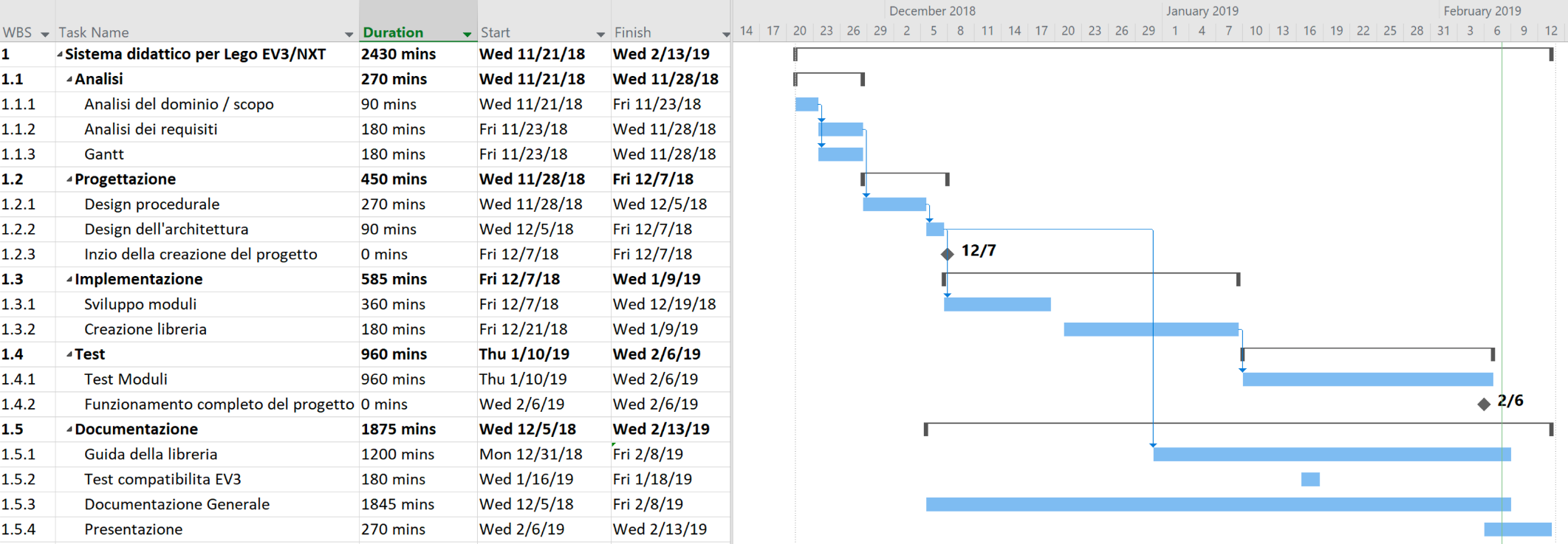
## Mancanze/limitazioni conosciute

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.

# 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 concentraci 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.

# 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.

## 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.

## 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 può essere 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.

# Bibliografia

## 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*

# Allegati

1. Diari di Progetto
2. Quaderno dei compiti
3. Guida installazione LeJOS
4. Guida Libreria
5. Prodotto
   1. Github: <https://github.com/andrearauso/Progetto-2>