Documentazione Progetto EV3

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

1.4 Analisi del dominio 3

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 7

2.1 Design dell’architettura del sistema 7

2.2 Design delle interfacce 8

2.3 Design procedurale 9

3 Implementazione 10

3.1 Pagina di benvenuto 10

3.2 Pagina di registrazione 11

3.2.1 Form inserimento dati e controlli 11

3.2.2 Tabella verifica dati 14

3.3 Pagina di salvataggio 16

3.3.1 Ricezione e controllo dati 16

3.3.2 Uso file csv 17

4 Test 19

4.1 Protocollo di test 19

4.2 Risultati test 22

4.3 Mancanze/limitazioni conosciute 22

5 Consuntivo 23

6 Conclusioni 24

6.1 Sviluppi futuri 24

6.2 Considerazioni personali 24

7 Bibliografia 25

7.1 Sitografia 25

8 Allegati 25

# Introduzione

## Informazioni sul progetto

Allievi coinvolti: Jari Näser, Paolo Gübeli

Classe: Informatica 3AA Presso la Scuola Arti e Mestieri Trevano

Docenti responsabili: Francesco Mussi, Adiano Barchi, Luca Muggiasca

Data inizio: 7-11-18

Data fine: 8-02-19

## Abstract

Today a lot of schools use mindstorms to introduce younglings into computer engineering, the scope of the project was to realize some libraries to make it easier for the schools to educate the students. Those libraries will simplify some operations that are too complicated for amateurs. To achieve this goal, we used the programming language Java.

There will be also a guide to help the teachers and the students to install the firmware and how to use those libraries.

## Scopo

Lo scopo di questo progetto è di creare delle librerie che permettono di semplificare delle operazione con il prodotto Mindstorm NXT. Queste librerie verrano in seguito usate per semplificare la programmazione dei Mindstorm togliendo passaggi ripetitivi. Questo per aiutare docenti che cercano di insegnare la programmazione a ragazzi alle prime armi.

**Analisi**

## Analisi del dominio

Attualmente si usa il vecchio sistema a blocchetti semplificato della lego che non permette di avere funzionalità avanzate ed è pensato per regazzi giovani alle prime armi con l’informatica.

## Analisi e specifica dei requisiti

Inizialmente bisogna installare il firmware per java leJOS che permette al mindtorm di leggere i file java.. In seguito bisogna creare per ogni blocchetto esistente nel editor grafico di NXT un metodo che lo sostituisca. Ogni libreria dovrà permettere di usare un metodo che gestisce un’azione con degli input e degli output.

In seguito con queste librerie bisogna crare un programma che muove il robot in giro e si gira quando tocca o vede un oggetto comunemente chiamato Explorer.

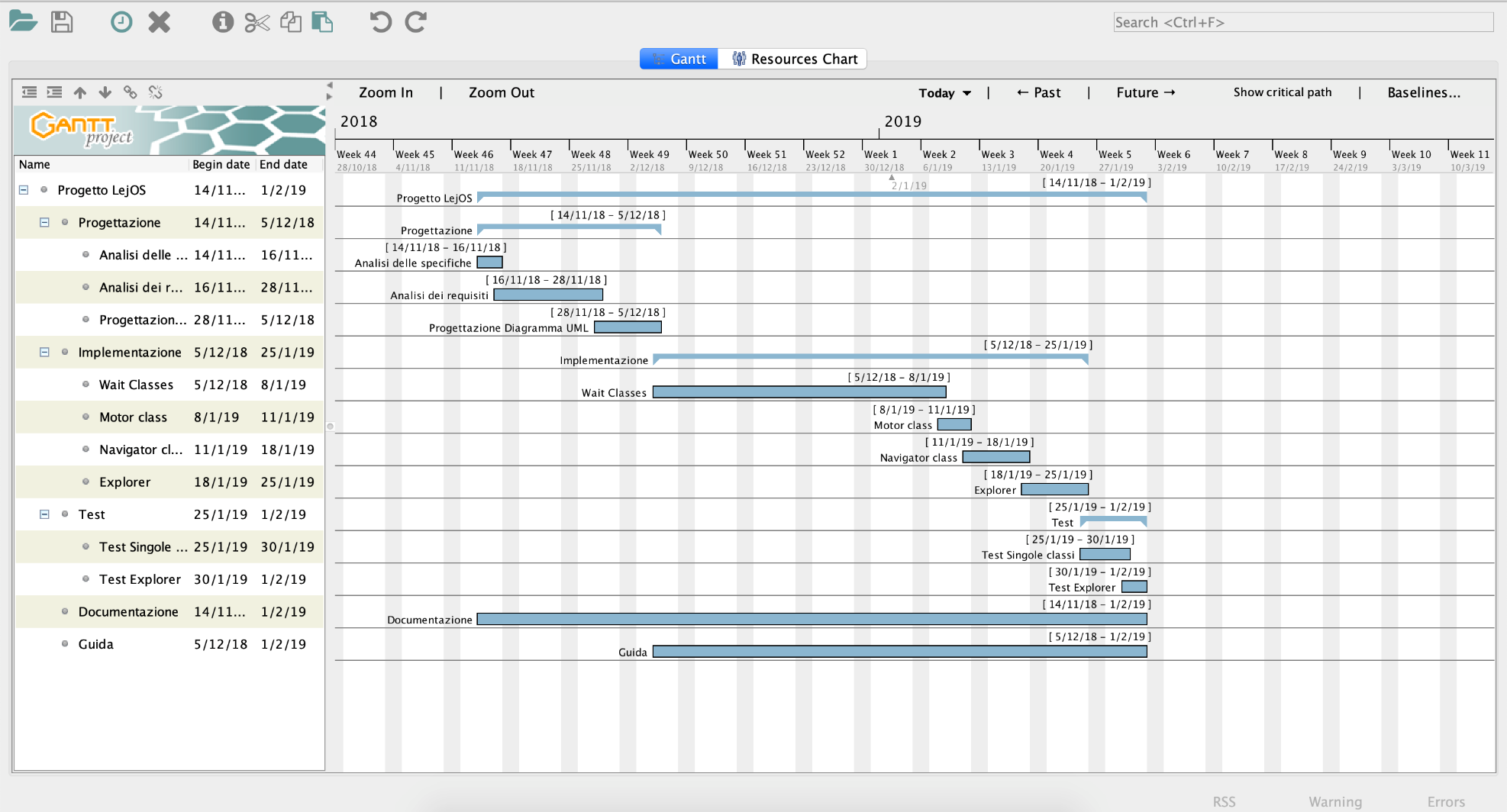
|  |  |
| --- | --- |
| **ID: REQ-01** | |
| **Nome** | Firmware |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Firmware che permettono di usare java |
| **Sotto requisiti** | |
| **01** | Si necessita di un mindstorm funzionante |

|  |  |
| --- | --- |
| **ID: REQ-02** | |
| **Nome** | Creazione librerie |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Librerie che permettono di fare azioni complicate in modo semplice così da semplificare l’uso del mindstorm |
| **Sotto requisiti** | |
| **01** | Si necessita dei sensori |
| **02** | Si necessita dei attuatori |
| **03** | Si necessita dei firmware installati |

|  |  |
| --- | --- |
| **ID: REQ-03** | |
| **Nome** | Guida per gli utenti |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Guida che permetta di usare le librerie create |
| **Sotto requisiti** | |
| **01** | Installazione dei firmware completa |
| **02** | Librerie terminate e complete |

|  |  |
| --- | --- |
| **ID: REQ-04** | |
| **Nome** | Explorer che fa us delle librerie |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Explorer che usa le classi delle librerie per funzionare |
| **Sotto requisiti** | |
| **01** | Mindsrorm funzionante con LeJos installato |
| **02** | Libreria creata e funzionante |

## Pianificazione



## Analisi dei mezzi

### Software

Per la realizzazione di questo progetto abbiamo usato come software:

* LeJOS 0.9.1: Firmware e compilatore che ci permette di usare Java su mindstorm.
* Sublime Text 3.1.1: ci aiuta a scrivere i codici java.
* Notepad++ 7.5.1: aiuta a scrivere ogni sorta di codice.
* Word 2016: ci ha permesso di scrivere la Documentazione del progetto e la guida.
* GanttProject 2.8.5: ci ha permesso di fare il Gantt iniziale e il Gantt consuntivo

### Hardware

Per questo progetto non ho necessitato di materiale particolare, abbiamo usato i nostri portatile MacBook Pro 2015 con il sistema operativo OS X Mojave e Hp OMEN 17” con sistema operativo Windows 10.

# Progettazione

# Implementazione

## Sensori

Abbiamo implementato le classi per aspettare che dei sensori ritornino un valore.

Per usare le classi bisogna prima implementarle nel proprio programma e passare il sensore usato.

Tutte le classi hanno due metodi base:

myWait() che interrompe il programma finché isFinished non ritorna true.

isFinished() che restituisce true se la condizione è verificata.

Questo perché magari un utente vuole eseguire del codice mentre aspetta la condizione, quindi lasciamo la possibilità di creare un proprio while e mettere come condizione isFinished().

Queste classi sono state suddivise in tre gruppi:

-Pulsanti

-Colore

-Sensori Analogici (Ultrasuoni, suono, luminosità)

### WaitTouchSensor

Per iniziare abbiamo implementato la classe WaitTouchSensor che è quella più semplice, abbiamo suddiviso gli input in due, pressed, released con dei valori int 0,1 che verrano passati al myWait() o al isFinished().

i*mport lejos.nxt.\*;*

*public class WaitTouchSensor{*

*private TouchSensor touch;*

*private boolean pressed = false;*

*public WaitTouchSensor(TouchSensor touch){*

*this.touch = touch;*

*}*

*public void myWait(int action){*

*while(isFinished(action)){*

*}*

*}*

*public boolean isFinished(int action){*

*boolean finished = false;*

*if(action == 0){*

*finished = touch.isPressed();*

*}else{*

*if(pressed){*

*finished = !touch.isPressed();*

*}*

*pressed = touch.isPressed();*

*}*

*return finished;*

*}*

*}*

### WaitColorSensor

WaitColorSensor è l‘unica classe che usa un range di valori quindi abbiamo dovuto cercare un buon range intorno ai valori passati, questo perché con questi sensori non uscira mai il valore esatto quindi bisogna essere larghi con i valori noi abbiamo optato per un 8% di margine d’errore. Quando si usa la classe bisogna passare il valore di RGB suddivisi in tre int.

*import lejos.nxt.\*;*

*import lejos.robotics.\*;*

*public class WaitColorSensor{*

*private ColorSensor cs;*

*public WaitColorSensor(ColorSensor cs){*

*this.cs = cs;*

*}*

*public void wait(int red, int green, int blue){*

*while(isFinished(red, green, blue)){*

*}*

*}*

*public boolean isFinished(int red, int blue, int green){*

*Color c = cs.getColor();*

*if(c.getRed() > red-10 && c.getRed() < red+10){*

*if(c.getGreen() > green-10 && c.getGreen() < green+10){*

*if(c.getBlue() > blue-10 && c.getBlue() < blue+10){*

*return true;*

*}*

*}*

*}*

*return false;*

*}*

*}*

### WaitLightSensor

WaitLightSensor è una classe che gestisce il sensore di luce e riceve un valoreint e un boolean per sapere se il valore cercato dev’essere maggiore o minore del valore passato.

import lejos.nxt.\*;

*public class WaitLightSensor{*

*private LightSensor light;*

*public WaitLightSensor(LightSensor light){*

*this.light = light;*

*}*

*public void myWait(boolean bigger, int value){*

*while(isFinished(bigger, value)){*

*}*

*}*

*public boolean isFinished(boolean bigger, int value){*

*if(bigger){*

*if(light.getLightValue() > value){*

*return true;*

*}*

*}else{*

*if(light.getLightValue() < value){*

*return true;*

*}*

*}*

*return false;*

*}*

*}*

### WaitUltrasonicSensor

WaitUltrasonicSensor è una classe che gestisce il sensore ad ultrasuoni e riceve un valore int e un boolean per sapere se il valore cercato dev’essere maggiore o minore del valore passato.

import lejos.nxt.\*;

public class WaitUltrasonicSensor{

private UltrasonicSensor sonic;

public WaitUltrasonicSensor(UltrasonicSensor sonic){

this.sonic = sonic;

}

public void wait(boolean bigger, int value){

while(isFinished(bigger, value)){

}

}

public boolean isFinished(boolean bigger, int value){

if(bigger){

if(sonic.getDistance() > value){

return true;

}

}else{

if(sonic.getDistance() < value){

return true;

}

}

return false;

}

}

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Installazione Firmware funzionante |
| **Descrizione:** | Prova del funzionamento corretto del firmware | | |
| **Prerequisiti:** | Robot Mindstorm funzionante | | |
| **Procedura:** | 1. Accendere il mindstorm tramite il pulsante centrale 2. Caricare tramite cavo un programma funzionante con un output | | |
| **Risultati attesi:** | Il robot dovrebbe avviare il programma e ritornare l’output | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-002 | **Nome:** | Controllo funzionamento caricamento librerie |
| **Descrizione:** | Prova se è possibile caricare le librerei su robot mindstorm | | |
| **Prerequisiti:** | Mindstorm funzionante  Firmware funzionante | | |
| **Procedura:** | 1. Scaricare le librerie 2. Caricare tramite cavo le librerie scaricate | | |
| **Risultati attesi:** | Sotto la sezione files dovrebbero esserci i file delle librerie | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-003 | **Nome:** | Guida corretta e funzionante |
| **Descrizione:** | Provare il funzionamento del codice d’esempio nella guida. | | |
| **Prerequisiti:** | Mindstorm funzionante  Firmware funzionante.  Librerie funzionanti. | | |
| **Procedura:** | 1. Aprire la guida 2. Copiare il codice illustrato nella guida 3. Inserire il codice copiato in un programma 4. Caricare il programma sul robot 5. Far partire il programma. | | |
| **Risultati attesi:** | Il programma dovrebbe funzionare correttamente senza interruzioni. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-004 | **Nome:** | Funzionamento programma explorer. |
| **Descrizione:** | Test del corretto funzionamento del programma explorer.class. | | |
| **Prerequisiti:** | Mindstorm funzionante  Firmware funzionante.  Librerie funzionanti. | | |
| **Procedura:** | 1. Scaricare il file aggiuntivo di prova della libreria explorer. 2. Caricare il programma explorer sul mindstorm tramite cavo. 3. Appogiare il robot su una superficie piana, con un terreno agibile (pavimento, tavolo) 4. Avviare il programma. | | |
| **Risultati attesi:** | Tutti gli input immessi nei Form vengono controllati e se sono accettati vengono mandati alla tabella di conferma | | |

## Risultati test

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Numero Passaggio** | **Risultato** |
| TC-001 | 1-2 | Il robot ci ritorna l’output corretto |
| TC-002 | 2 | Exception 134  Quest’eccezione non esiste nella documentazione ufficiale di LeJOS e da nessun’altra pagina |
| TC-003 | 4 | Exception 134  Quest’eccezione non esiste nella documentazione ufficiale di LeJOS e da nessun’altra pagina |
| TC-004 | 2 | Exception 134  Quest’eccezione non esiste nella documentazione ufficiale di LeJOS e da nessun’altra pagina |

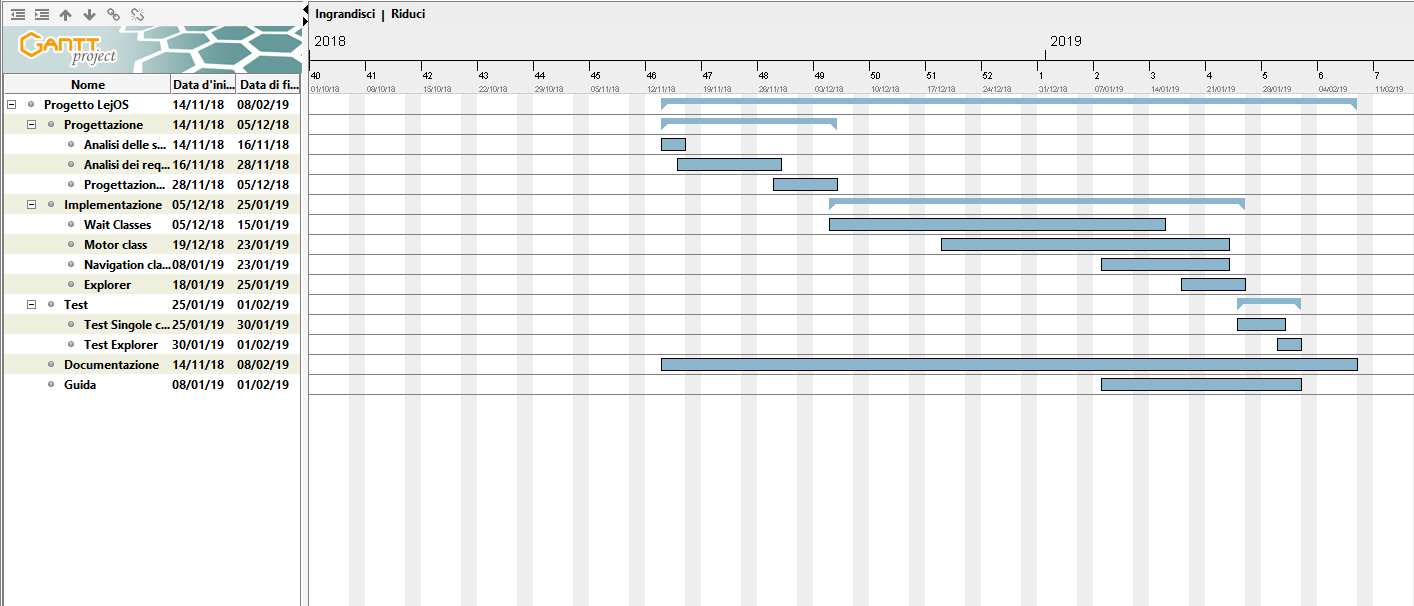
## Mancanze/limitazioni conosciute

Il tipo di Form “date” che al click apre un calendario che su mobile per un qualche motivo non funziona, non lascia immettere alcun testo nel campo o selezionare una data.

Di conseguenza non l’ho potuto implementare e ho tenuto il campo di tipo testo.

# Consuntivo

Siamo riusciti a mantenere i tempi della progettazione e dei test ma l’implementazione per colpa di vari problemi technici ci ha preso un po’ più tempo del previsto:



# Conclusioni

Il nostro prodotto permette di semplificare molte azioni

## Sviluppi futuri

Sicuramente ci sono molti metodi per aiutare l’utente che si possono aggiungere anche se dobbiamo dire che i metodi di base di LeJOS sono già molto buoni e permettono di fare tutto, con l’aggiunta delle nostre librerie l’uso di LeJOS diventa facile da usare ma bisogna avere comunque una conoscenza di base del linguaggio di programmazione java.

## Considerazioni personali

Abbiamo imparato a collaborare ad un progetto, e questo ci ha fatto capire l’importanza della puntualità e della costanza nei commit e nei push. Grazie a questo progetto abbiamo capito l’importanza della progettazione che ha reso facile la suddivisione dei lavori e la gestione dei tempi di consegna. A differenza del progetto fatto da soli si dipendeva dal compagno e viceversa quindi una buona collaborazione è essenziale per riuscire nel lavoro.

# Bibliografia

## Sitografia

* http://stackoverflow.com/, *Stack OverFlow*, dal 19.12.2018 al 25.01.2019
* <http://www.lejos.org/,dal> 19.12.2018 al 18.01.2019
* http://www.lejos.org/nxt/pc/api/index.html

# Allegati

* Diari di lavoro
* Codici sorgente
* Guida per l’utente
* Quaderno dei compiti
* Prodotto