Documentazione Progetto NXT

Titolo del progetto: Documentazione Progetto Registrazione
Alunni/e: Jari Näser, Paolo Gübeli
Classe: Informatica 3AA
Anno scolastico: 2018/2019
Docente responsabile: Adriano Barchi, Francesco Mussi e Luca Muggiasca

1 Introduzione

1.1 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

1.2 Abstract

Today a lot of schools use LEGO Mindstorm to introduce younglings into computer programming and engineering, the scope of the project was to realize some libraries to make it easier for schools to educate the students. These 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.

1.3 Scopo

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

Analisi

1.4 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.
Di conseguenza non è ancora possibile programmare i robot con del codice.

1.5 Analisi e specifica dei requisiti

Inizialmente bisogna installare il firmware per java leJOS che permette al mindstorm 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

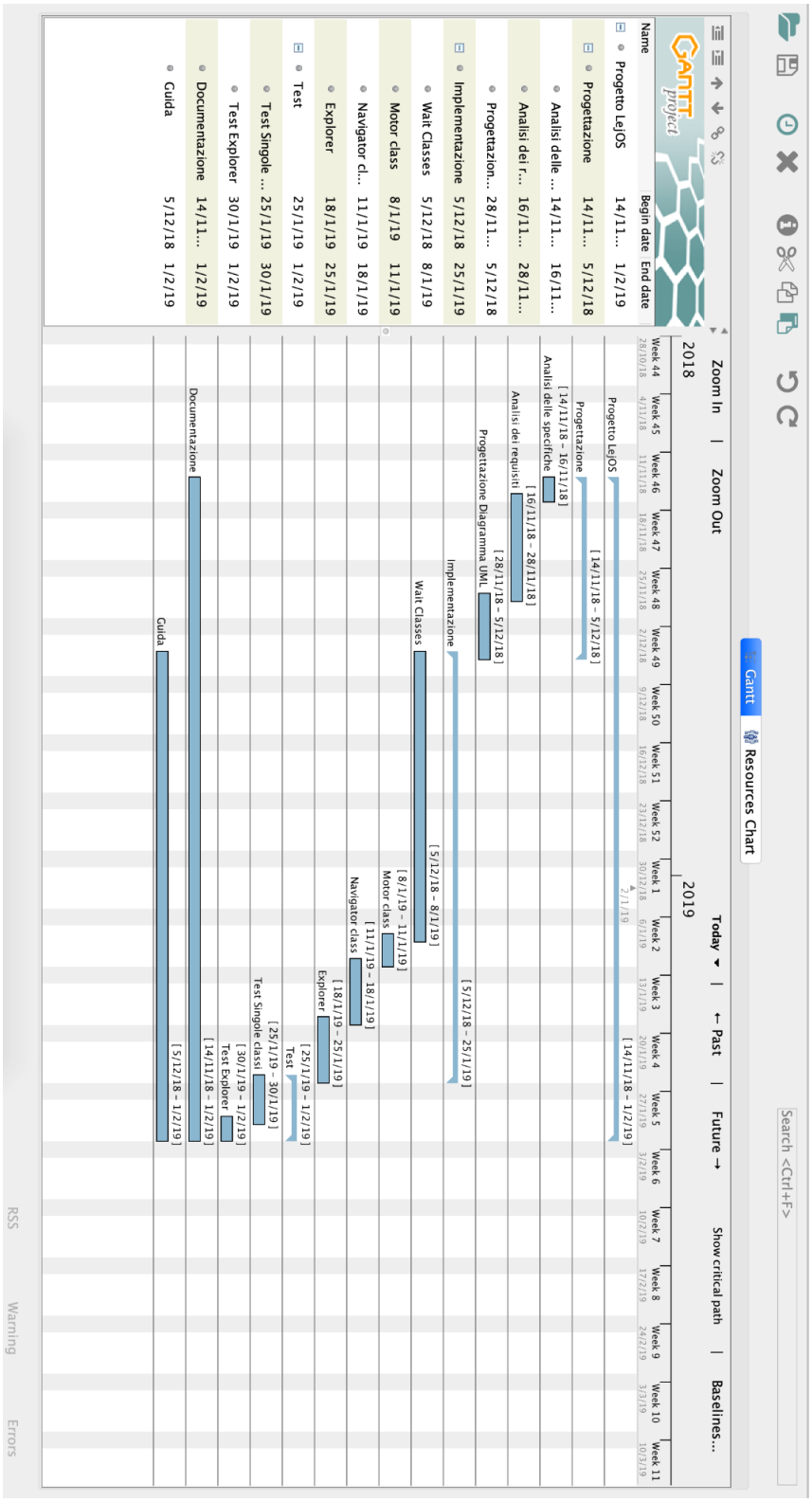
ID: REQ-03	
Nome	Guida per gli utenti
Priorità	1
Versione	1.0
Note	Guida che permetta di usare le librerie create
Sotto requisiti	

01	Installazione dei firmware completa
02	Librerie terminate e complete

ID: REQ-04	
Nome	Explorer che fa uso 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

1.6 Pianificazione

Questa è la nostra pianificazione del progetto e il modo in cui abbiamo suddiviso i tempi:



1.7 Analisi dei mezzi

1.7.1 Software

Per la realizzazione di questo progetto abbiamo usato come software:

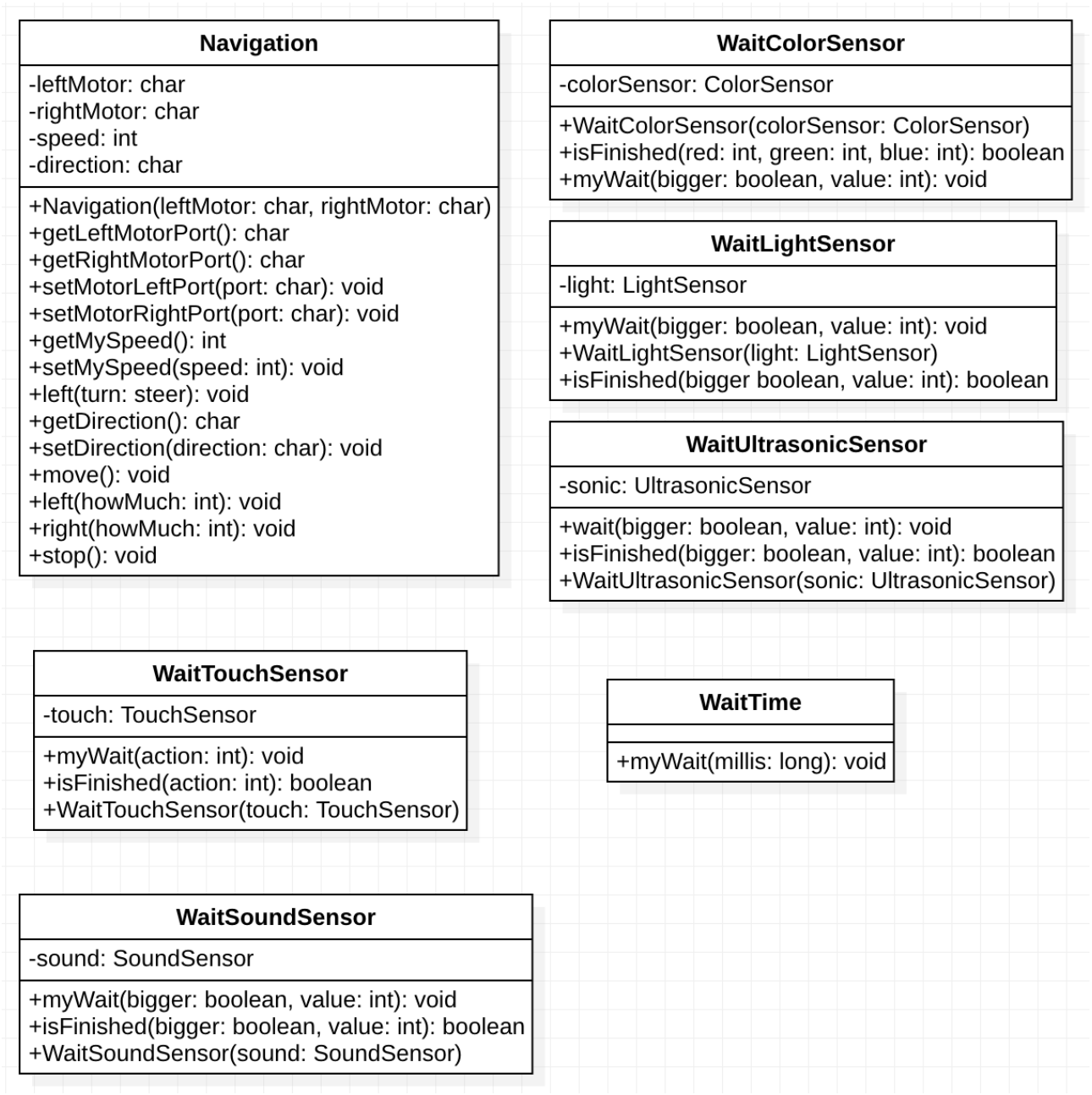
- LeJOS 0.9.1: Firmware e compilatore che ci permette di usare Java su mindstorm.
- Sublime Text 3.1.1: ci aiuta a scrivere i codici java.
- Notepad++ 7.5.1: aiuta a scrivere ogni sorta di codice.
- Word 2016: ci ha permesso di scrivere la Documentazione del progetto e la guida.
- GanttProject 2.8.5: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

1.7.2 Hardware

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

2 Progettazione

2.1 Design dell'architettura del sistema



3 Implementazione

3.1 Navigation

La classe Navigation ha come scopo principale quello di mettere assieme tutte le classi ed esportare dei metodi semplici e funzionanti per guidare il robot.
Infatti è composto dai 4 metodi principali:

- **Move**: Si occupa di far partire il robot.
- **Left**: Si occupa di far curvare il robot a sinistra.
- **Right**: Si occupa di far curvare il robot a destra.
- **Stop**: Si occupa di fermare il robot

E di tutti i suoi metodi ed attributi rimanenti.

```
import lejos.nxt.*;

public class Navigation{

	private char leftMotor = 'A';
	private char rightMotor = 'B';
	private int speed = 0;
	private char direction = 'F';

	public Navigation(char leftMotor, char rightMotor){
		setMotorLeftPort(leftMotor);
		setMotorRightPort(rightMotor);
	}

	public char getLeftMotorPort(){
		return this.leftMotor;
	}

	public char getRightMotorPort(){
		return this.rightMotor;
	}

	public void setMotorLeftPort(char port){
		String mp = port + "";
		port = mp.toUpperCase().charAt(0);
		if(port == 'A' || port == 'B' || port == 'C'){
			leftMotor = port;
		}
	}

	public void setMotorRightPort(char port){
		String mp = port + "";
		port = mp.toUpperCase().charAt(0);
		if(port == 'A' || port == 'B' || port == 'C'){
			leftMotor = port;
		}
	}

	public int getMySpeed(){
		return this.speed;
	}

	public void setMySpeed(int speed){
```

```
		if(speed >= 0){
			this.speed = speed;
		}
	}

	public char getDirection(){
		return this.direction;
	}

	public void setDirection(char direction){
		String s = direction + "";
		direction = s.toUpperCase().charAt(0);
		if(direction == 'F' || direction == 'B'){
			this.direction = direction;
		}
	}

	public void move(){
		switch(getLeftMotorPort()){
			case 'A':
				Motor.A.setSpeed((float)this.getMySpeed());
			case 'B':
				Motor.B.setSpeed((float)this.getMySpeed());
			case 'C':
				Motor.C.setSpeed((float)this.getMySpeed());
		}
		switch(getRightMotorPort()){
			case 'A':
				Motor.A.setSpeed((float)this.getMySpeed());
			case 'B':
				Motor.B.setSpeed((float)this.getMySpeed());
			case 'C':
				Motor.C.setSpeed((float)this.getMySpeed());
		}
		if(getDirection() == 'F'){
			switch(getLeftMotorPort()){
				case 'A':
					Motor.A.forward();
				case 'B':
					Motor.B.forward();
				case 'C':
					Motor.C.forward();
			}
			switch(getRightMotorPort()){
				case 'A':
					Motor.A.forward();
				case 'B':
					Motor.B.forward();
				case 'C':
					Motor.C.forward();
			}
		}else{
			switch(getLeftMotorPort()){
				case 'A':
					Motor.A.backward();
				case 'B':
					Motor.B.backward();
				case 'C':
```

```
                                Motor.C.backward();
                        }
                        switch(getRightMotorPort()){
                                case 'A':
                                        Motor.A.backward();
                                case 'B':
                                        Motor.B.backward();
                                case 'C':
                                        Motor.C.backward();
                        }
                }
        }

        public void left(int howMuch){
                switch(getLeftMotorPort()){
                        case 'A':
                                Motor.A.setSpeed((float)(this.getMySpeed() - this.getSteering()/2));
                        case 'B':
                                Motor.B.setSpeed((float)(this.getMySpeed() - this.getSteering()/2));
                        case 'C':
                                Motor.C.setSpeed((float)(this.getMySpeed() - this.getSteering()/2));
                }
                switch(getRightMotorPort()){
                        case 'A':
                                Motor.A.setSpeed((float)(this.getMySpeed() + this.getSteering()/2));
                        case 'B':
                                Motor.B.setSpeed((float)(this.getMySpeed() + this.getSteering()/2));
                        case 'C':
                                Motor.C.setSpeed((float)(this.getMySpeed() + this.getSteering()/2));
                }
        }

        public void right(int howMuch){
                switch(getLeftMotorPort()){
                        case 'A':
                                Motor.A.setSpeed((float)(this.getMySpeed() + this.getSteering()/2));
                        case 'B':
                                Motor.B.setSpeed((float)(this.getMySpeed() + this.getSteering()/2));
                        case 'C':
                                Motor.C.setSpeed((float)(this.getMySpeed() + this.getSteering()/2));
                }
                switch(getRightMotorPort()){
                        case 'A':
                                Motor.A.setSpeed((float)(this.getMySpeed() - this.getSteering()/2));
                        case 'B':
                                Motor.B.setSpeed((float)(this.getMySpeed() - this.getSteering()/2));
                        case 'C':
                                Motor.C.setSpeed((float)(this.getMySpeed() - this.getSteering()/2));
                }
        }

        public void stop(){
                switch(getLeftMotorPort()){
                        case 'A':
                                Motor.A.stop();
                        case 'B':
                                Motor.B.stop();
                        case 'C':
```

```
                    Motor.C.stop();
                }
                switch(getRightMotorPort()){
                    case 'A':
                        Motor.A.stop();
                    case 'B':
                        Motor.B.stop();
                    case 'C':
                        Motor.C.stop();
                }
        }
}
```

3.2 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)

3.2.1 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().

```
import 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;
        }
}
```

3.2.2 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;
        }
}
```

3.2.3 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;
	}
}
```

3.2.4 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;
	}
}
```

3.3 Explorer

La classe explorer si occupa di fare schivare tutti gli ostacoli presenti nel percorso da effettuare oppure in una zona rinchiusa al robot attraverso due sensori di tatto, uno ad ultrasuoni ed uno di colore, inoltre quando passa su una riga nera si ferma.

Il suo funzionamento è molto semplice: Quando sta per incontrare o ha incontrato un ostacolo si ferma, torna in dietro, gira a destra o sinistra, si ferma e riparte proseguendo il suo nuovo percorso.

```
import lejos.nxt.*;

public class Explorer{
	private static final char MOTOR_LEFT_PORT = 'A';
	private static final char MOTOR_RIGHT_PORT = 'B';
	private static final int DISTANCE = 30;
	private static Navigation navigator;
	private static UltrasonicSensor ultrasonicSensor = new UltrasonicSensor(SensorPort.S3);
	private static WaitUltrasonicSensor waitUltrasonicSensor = new
WaitUltrasonicSensor(ultrasonicSensor);
	private static TouchSensor touchSensorLeft = new TouchSensor(SensorPort.S2);
	private static TouchSensor touchSensorRight = new TouchSensor(SensorPort.S4);
	private static WaitTouchSensor waitTouchSensorLeft = new WaitTouchSensor(touchSensorLeft);
	private static WaitTouchSensor waitTouchSensorRight = new WaitTouchSensor(touchSensorRight);
	private static LightSensor lightSensor = new LightSensor(SensorPort.S1);
	private static WaitLightSensor waitLightSensor = new WaitLightSensor(lightSensor);
	private static WaitTime wait;

	public Explorer(){
		navigator = new Navigation(MOTOR_LEFT_PORT, MOTOR_RIGHT_PORT);
		navigator.setMySpeed(300);
		wait = new WaitTime();
	}

	public static void main(String[] args){
		navigator.move();
		while(true){
			if(waitUltrasonicSensor.isFinished(false, DISTANCE)){
				//Inserire cosa deve fare se è più vicino di 30 cm
				navigator.stop();
				navigator.setDirection('B');
				navigator.move();
```

```
                                wait.myWait(1500);
                                navigator.stop();
                                navigator.setDirection('F');
                                navigator.right(60);
                                wait.myWait(2000);
                                navigator.setMySpeed(300);
                        }
                        if(waitTouchSensorLeft.isFinished(0)){
                                //gira a sinistra indietro
                                navigator.stop();
                                navigator.setDirection('B');
                                navigator.move();
                                wait.myWait(1500);
                                navigator.stop();
                                navigator.setDirection('F');
                                navigator.left(60);
                                wait.myWait(2000);
                                navigator.setMySpeed(300);
                        }
                        if(waitTouchSensorRight.isFinished(0)){
                                //gira a destra indietro
                                navigator.stop();
                                navigator.setDirection('B');
                                navigator.move();
                                wait.myWait(1500);
                                navigator.stop();
                                navigator.setDirection('F');
                                navigator.right(60);
                                wait.myWait(2000);
                                navigator.setMySpeed(300);
                        }
                        if(waitLightSensor.isFinished(false, 50)){
                                //Si ferma
                                navigator.stop();
                        }
                }
        }
}
```

4 Test

4.1 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		

4.2 Risultati test

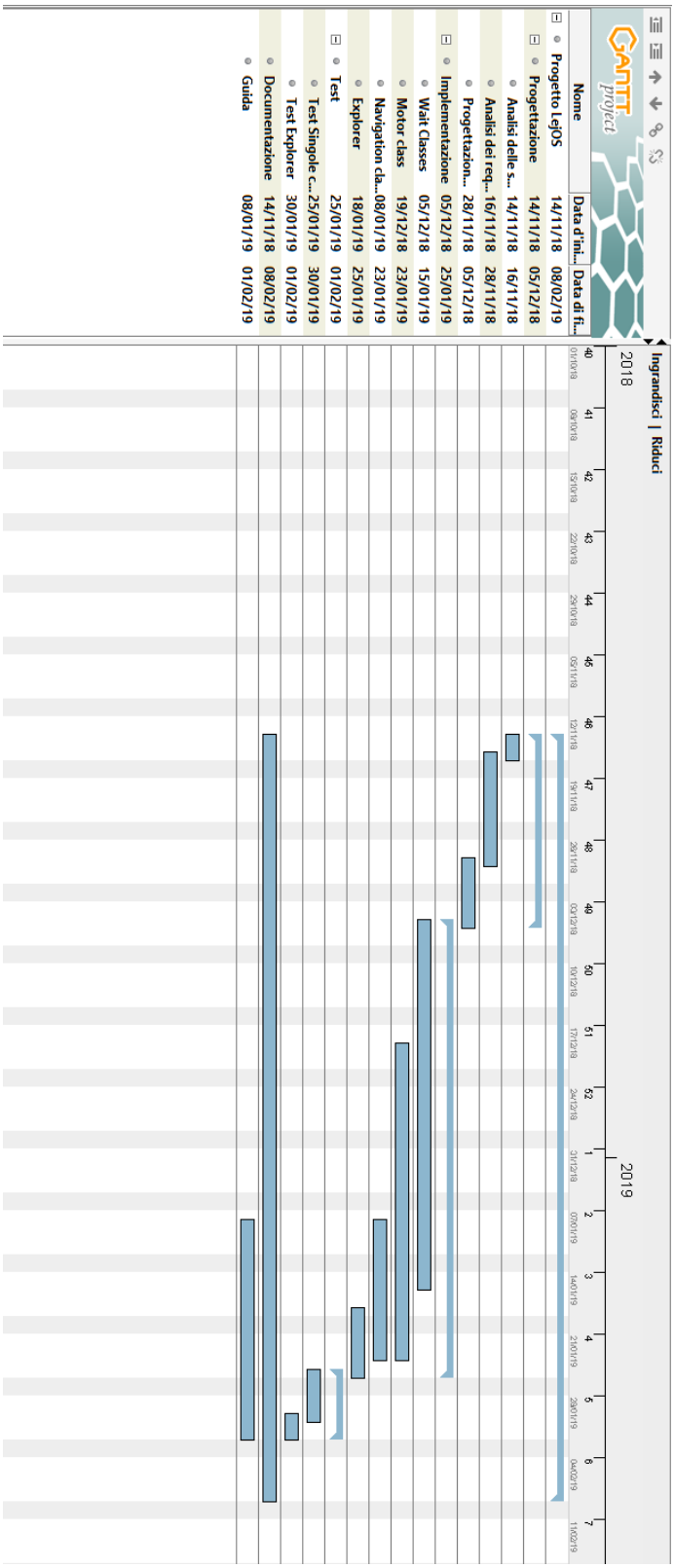
Test Case	Numero Passaggio	Risultato
TC-001	1-2	Il robot ci ritorna l'output corretto
TC-002	2	Exception 134 Quest'eccezione non esiste nella documentazione ufficiale di LeJOS e da nessun'altra pagina
TC-003	4	Exception 134 Quest'eccezione non esiste nella documentazione ufficiale di LeJOS e da nessun'altra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

4.3 Mancanze/limitazioni conosciute

La struttura di lejOS è limitante non potendo creare i motori come oggetti essendo già presenti, inoltre abbiamo riscontrato un'errore nel passaggio dei file da PC a Blocchetto NXT in quanto sollevava un'eccezione non documentata nella documentazione ufficiale di lejOS.

5 Consuntivo

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



6 Conclusioni

Il nostro prodotto offre un uso molto semplice del robot LEGO NXT attraverso del codice Java che può essere facilmente usato ed interpretato da chiunque abbia delle minime conoscenze nel mondo della programmazione.

6.1 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.

6.2 Considerazioni personali

Abbiamo imparato a collaborare in 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 e comunicazione è essenziale per la riuscita del lavoro.

7 Bibliografia

7.1 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
- http://www.free-powerpoint-templates-design.com

8 Allegati

- Diari di lavoro
- Codici sorgente
- Guida per l'utente
- Quaderno dei compiti
- Prodotto

Diario di lavoro

Luogo	SAM Trevano
Data	2018-11-14

Lavori svolti

Ci sono stati descritti i nuovi progetti e abbiamo scelto quello con i Lego Mindstorm. Abbiamo letto il quaderno dei compiti, creato la repository su github con la sua struttura e preparato un paio di domande da porre nella prossima lezione.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

-

Programma di massima per la prossima giornata di lavoro

Porre le domande e annotare le risposte

Diario di lavoro

Luogo	SAM Trevano
Data	2018-11-16

Lavori svolti
Oggi abbiamo fatto un test sul M.306 che ha occupato 2 ore lezione, in seguito il gruppo dei progetti arduino ha esposto le proprie domande sul QdC ai docenti. Mentre noi abbiamo guardato come funziona RobotC e scaricato i 10 giorni di prova.

Problemi riscontrati e soluzioni adottate
-

Punto della situazione rispetto alla pianificazione
-

Programma di massima per la prossima giornata di lavoro
Esporre le nostre domande inerenti al QdC
Iniziare a definire i requisiti del progetto

Diario di lavoro

Luogo	SAM Trevano
Data	2018-11-21

Lavori svolti
Oggi abbiamo posto le domande sul QdC al docente Barchi Adriano, in seguito abbiamo iniziato a installare leJOS che ci permette di caricare file java su NXT per poi essere interpretati dallo stesso. Per sapere come fare abbiamo consultato il sito http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/GettingStartedWindows.htm Per testare il corretto funzionamento dell'installazione avremmo bisogno delle batterie per l'NXT.

Problemi riscontrati e soluzioni adottate
-

Punto della situazione rispetto alla pianificazione
-

Programma di massima per la prossima giornata di lavoro
Iniziare a definire i requisiti del progetto
Iniziare a definire la progettazione

Diario di lavoro

Luogo	SAM Trevano
Data	2018-11-23

Lavori svolti
Oggi ho definito i requisiti e l'analisi del dominio in più ho iniziato a installare il firmware leJOS sul NXT. Ho avuto problemi con l'installazione e ho dovuto fare un reset del mindstorm. Il mio compagno ha creato il GANTT del progetto.

Problemi riscontrati e soluzioni adottate
Il mindstorm è resettato e non riesco a reinstallare un software

Punto della situazione rispetto alla pianificazione
-

Programma di massima per la prossima giornata di lavoro
Risolvere il problema con NXT
Imparare ad usare leJOS

Diario di lavoro

Luogo	SAM Trevano
Data	2018-11-28

Lavori svolti
Abbiamo resettato il firmware dell'NXT e installato quello di LeJOS cambiando le batterie e i driver della porta del computer (Windows). Inoltre abbiamo preparato tutti gli ambienti di sviluppo sui nostri portatili e abbiamo continuato a fare il diagramma di gantt.

Problemi riscontrati e soluzioni adottate
Problema nell'installazione del Phantom Driver su uno dei due PC.

Punto della situazione rispetto alla pianificazione
Siamo in orario rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro
Finire il diagramma di gantt.
Risolvere il problema con il Phantom driver.

Diario di lavoro

Luogo	SAM Trevano
Data	2018-11-30

Lavori svolti
Oggi abbiamo provato ogni sensore per essere sicuri che funzionano prima di implementare le librerie che li usano. In contemporanea abbiamo risolto il problema con lejOS della scorsa lezione. In seguito abbiamo implementato il primo metodo Line Follower che contiene tre parametri il flag che decide quando il programma si ferma, la velocità del motore passivo, e la velocità del motore attivo. La velocità dell'attivo dev'essere maggiore del passivo così che il robot curva verso la linea.

Problemi riscontrati e soluzioni adottate
Risolto il problema con l'intallazione di lejos. Cambiando la versione di java da 64 bit a 32.

Punto della situazione rispetto alla pianificazione
Siamo in tempo

Programma di massima per la prossima giornata di lavoro
Iniziare a implementare i primi metodi di base

Diario di lavoro

Luogo	SAM Trevano
Data	2018-12-05

Lavori svolti

Oggi abbiamo finito lo script che si occupa di compilare e passare i file .java da computer a robot.
Inoltre abbiamo modificato la struttura della repo aggiungendo la cartella guida e popolando la cartella src con i primi file .java di test come il **LineFollower**.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Siamo in orario rispetto alla pianificazione

Programma di massima per la prossima giornata di lavoro

Continuare lo sviluppo delle varie classi come il **LineAlligner**

Diario di lavoro

Luogo	SAM Trevano
Data	2018-12-07

Lavori svolti
Oggi abbiamo riformulato il progetto con il docente Mussi. Il progetto adesso consiste nel creare un metodo per ogni blocchetto verde o arancione presente nel editor grafico. In più usando questi metodo dobbiamo creare un explorer con due sensori di tatto e un sensore a ultrauoni frontali e un sensore di luce per ritrovare la casa.

Problemi riscontrati e soluzioni adottate
-

Punto della situazione rispetto alla pianificazione
Siamo in ritardo per il cambio di programma ma è stata aggiunta una settimana quindi dobbiamo ridefinire le tempistiche

Programma di massima per la prossima giornata di lavoro
Mettere a posto la pianificazione
Iniziare a fare i metodi di base

Diario di lavoro

Luogo	SAM Trevano
Data	2018-12-12

Lavori svolti

Oggi abbiamo inizialmente guardato cosa e che parametri richiedeva l'IDE "LEGO MINDSTORMS NXT" nella quale i programmi vengono creati attraverso l'insieme di vari blocchetti per farci un'idea chiara di cosa dovevamo implementare nelle nostre classi.
Successivamente abbiamo incominciato a sviluppare le classi base per ogni componente che ci faranno da base per la libreria.
Abbiamo iniziato con quella del motore , dei sensori come quello di tatto e del display.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Siamo in orario con il diagramma di gantt

Programma di massima per la prossima giornata di lavoro

Continuare a sviluppare le varie classi per tutti i componenti base.

Diario di lavoro

Luogo	SAM Trevano
Data	2018-12-14

Lavori svolti

Oggi abbiamo chiarito i nostri dubbi sulle specifiche del progetto. Abbiamo capito che nel progetto dobbiamo principalmente implementare una libreria per il wait che permette all'utente di aspettare del tempo o un valore. In seguito abbiamo discusso insieme al gruppo Bosco/Alessi sulla progettazione e l'orgaanizzazione delle classi. Verso la fine della lezione abbiamo iniziato ad implementare le prime classi

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Siamo in tempo

Programma di massima per la prossima giornata di lavoro

Iniziare a implementare le prime classi di base.

Diario di lavoro

Luogo	SAM Trevano
Data	2018-12-19

Lavori svolti

Oggi abbiamo continuato con l'implementazione delle varie classi del progetto basandoci sul nostro diagramma delle classi.
Le classi che abbiamo sviluppato o stiamo finendo di svilupparle sono: Motor, WaitMotor, Wait, WaitTime, WaitDigitalSensor e WaitTouchSensor.
Abbiamo anche corretto e modificato la sua struttura diagramma li dove ritenevamo necessario.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Siamo in orario rispetto al diagramma di gantt.

Programma di massima per la prossima giornata di lavoro

Continuare con l'implementazione delle varie classi.

Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-09

Lavori svolti
Oggi abbiamo continuato con lo sviluppo delle varie classi, soprattutto la Navigation.java nella quale siamo avanzati molto e dobbiamo ancora finire piccolezze. Inoltre abbiamo anche fatto svariati test sul robot NXT stesso per capire come funzionavano esattamente certi metodi della libreria di lejOS come il rotate(angle) e il readValue() del SoundSensor.

Problemi riscontrati e soluzioni adottate
-

Punto della situazione rispetto alla pianificazione
Siamo in orario rispetto al diagramma.

Programma di massima per la prossima giornata di lavoro
Finire la classe Navigator
Iniziare la classe Explorer

Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-11

Lavori svolti
Oggi abbiamo continuato l'implementazione aggiungendo le classi WaitLightSensor, WaitSoundSensor, WaitUltrasonicSensor e in più abbiamo iniziato la guida per l'uso della nostra libreria

Problemi riscontrati e soluzioni adottate
-

Punto della situazione rispetto alla pianificazione
Siamo in ritardo secondo il programma ma dovremmo riscire ad accelerare i test

Programma di massima per la prossima giornata di lavoro
Finire la guida
Mettere insieme le ultime classi e iniziare la fase di testing

Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-16

Lavori svolti

Oggi abbiamo finito la classe Navigator contenente tutti i metodi per la nagivazione del robot, inoltre abbiamo anche aggiornato il diagramma delle classi togliendo ciò che non abbiamo ritenuto necessario di implementare o quello che abbiamo modificato.
Infine abbiamo costruito il robot NXT con i pezzi LEGO che ci ha preso la maggior parte del tempo della lezione a disposozione.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Siamo in orario rispetto al diagramma

Programma di massima per la prossima giornata di lavoro

Iniziare a creare le classi Explorer e LineStop.

Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-18

Lavori svolti
Oggi abbiamo messo insieme le classi create nelle scorse lezioni e abbiamo creato la classe Explorer.java che fa muovere un robot in giro e se incontra un ostacolo lo evita usando i sensori di tatto e quello degli ultrasuoni. Quando passa su una linea nera si ferma il programma.

```
21
22      public static void main(String[] args){
23          try{
24              navigator.move();
25              while(true){
26                  if(waitUltrasonicSensor.isFinished(false, DISTANCE)){
27                      //Inserire cosa deve fare se è più vicino di 30 cm
28                      navigator.stop();
29                      navigator.setDirection('B');
30                      navigator.move();
31                      Thread.sleep(1500);
32                      navigator.stop();
33                      navigator.setDirection('F');
34                      navigator.right(20);
35                      Thread.sleep(2000);
36                      navigator.setPower(80);
37                  }
38                  if(waitTouchSensorLeft.isFinished(0)){
39                      //gira a sinistra indietro
40                      navigator.stop();
41                      navigator.setDirection('B');
42                      navigator.move();
43                      Thread.sleep(1500);
44                      navigator.stop();
45                      navigator.setDirection('F');
46                      navigator.left(20);
47                      Thread.sleep(2000);
48                      navigator.setPower(80);
49                  }
50                  if(waitTouchSensorRight.isFinished(0)){
51                      //gira a destra indietro
52                      navigator.stop();
53                      navigator.setDirection('B');
54                      navigator.move();
55                      Thread.sleep(1500);
56                      navigator.stop();
57                      navigator.setDirection('F');
58                      navigator.right(20);
59                      Thread.sleep(2000);
60                      navigator.setPower(80);
61                  }
62                  if(waitLightSensor.isFinished(false, 50)){
63                      //Si ferma
64                      navigator.stop();
65                  }
66              }
67          }catch(InterruptedException ie){
68              //Interrupted
69          }
70      }
```

Problemi riscontrati e soluzioni adottate
Con il sistema a gerarchia c'era un problema con il passaggio dei valori quindi abbiamo eliminato la gerarchia e lasciato tutte le classi asestanti.

Punto della situazione rispetto alla pianificazione
Siamo in ritardo per il cambio di programma

Programma di massima per la prossima giornata di lavoro
Continuare la guida
Testare l'explorer

Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-23

Lavori svolti
Oggi abbiamo cambiato la struttura delle classi rimuovendo la classe MyMotor. Abbiamo continuato con lo sviluppo della classe Explorer nella quale ci mancano ancora poche cose da mettere a posto come il controllo della porta dei motori. Infine abbiamo anche continuato a sviluppare la guida e la documentazione.

Problemi riscontrati e soluzioni adottate
Abbiamo avuto problemi con il metodo Character.toUpperCase(char); che abbiamo rimpiazzato con una stringa ed il suo rispettivo metodo String.toUpperCase(String).

Punto della situazione rispetto alla pianificazione
Siamo in ritardo per il cambio di programma.

Programma di massima per la prossima giornata di lavoro
Finire la classe Explorer
Assicurare che l'Explorer funzioni come dovuto
Continuare la guida e la documentazione

Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-25

Lavori svolti
Oggi abbiamo continuato a redarre la guida e abbiamo messo a posto un errore presente nella classe WaitMotor, vista la eliminazione della classe MyMotor per una ridefinizione della struttura dlle classi manca il riferimento e abbiamo spostato dei metodi direttamente nella classe WaitMotor per risolvere il problema in più abbiamo rimosso ogni riferimento a MyMotor. In seguito ci siamo resi conto che Spostare WaitMotor nella classe navigation rende più facile l'uso della libreria all'utente finale.

Problemi riscontrati e soluzioni adottate
Gestione della classe WaitMotor dopo l'eliminazione della classe MyMotor-Abbiamo spostato tutti i metodi sia di WaitMotor sia di MyMotor nella classe Navigation.

Punto della situazione rispetto alla pianificazione
Siamo in ritardo per il cambio di programma ma è stata aggiunta una settimana quindi dovremmo riuscire con una piccola accelerazione a rimanere nei tempi

Programma di massima per la prossima giornata di lavoro
Finire la guida
Finire la parte d'implementazione sulla doc

Diario di lavoro

Luogo	SAM Trevano
Data	2019-01-30

Lavori svolti
Oggi abbiamo risolto tutti i problemi che avevamo nel codice di tutte le classi, soprattutto in Navigator ed Explorer nelle quali abbiamo dovuto rimuovere il metodo setPower() non essendo più supportato dalle nuove versioni di lejOS con setSpeed(). Inoltre abbiamo incontrato un problema con il nostro blocchetto che al passaggio dei file da PC a blocco sollevava un'eccezione (134) che non esiste sulla documentazione della lejOS.

Problemi riscontrati e soluzioni adottate
Eccezione sollevata che non siamo ancora riusciti a risolvere.

Punto della situazione rispetto alla pianificazione
Siamo in ritardo per il cambio di programma ma è stata aggiunta una settimana quindi dovremmo riuscire con una piccola accelerazione a rimanere nei tempi

Programma di massima per la prossima giornata di lavoro
Finire la guida
Risolvere il problema con il blocchetto NXT

Diario di lavoro

Luogo	SAM Trevano
Data	2019-02-01

Lavori svolti
Oggi abbiamo continuato la redazione della guida e della documentazione, nella guida manca la parte d'installazione di LeJOS invece nella Documentazione manca l'implementazione di WaitMotor e Navigation e un paio di test. Quindi abbiamo scritto l'implementazione dei sensori e creato il gantt consuntivo. Nella guida invece abbiamo aggiunto la guida di WaitMotor e Navigation e la guida dettagliata all'installazione di Java.

Problemi riscontrati e soluzioni adottate
-

Punto della situazione rispetto alla pianificazione
Siamo in ritardo per il cambio di programma ma è stata aggiunta una settimana quindi dovremmo riuscire con una piccola accelerazione a rimanere nei tempi

Programma di massima per la prossima giornata di lavoro
Finire la guida
Finire la parte d'implementazione sulla doc e la parte di test

Diario di lavoro

Luogo	SAM Trevano
Data	2019-02-06

Lavori svolti
Oggi abbiamo finito la guida e in parte concluso la documentazione. Generalmente abbiamo concluso la maggior parte delle cose per la consegna.

Problemi riscontrati e soluzioni adottate
-

Punto della situazione rispetto alla pianificazione
Siamo leggermente in ritardo per la mancanza di Paolo in queste 4 ore

Programma di massima per la prossima giornata di lavoro
Finire la documentazione
Finire la presentazione

Diario di lavoro

Luogo	SAM Trevano
Data	2019-02-08

Lavori svolti

Oggi abbiamo concluso le ultime correzione alla documentazione e alla guida, abbiamo ricontrollato tutti i testi alla ricerca di eventuali errori ortografici o di battitura. In seguito abbiamo creato la presentazione del progetto e completata con successo.

Problemi riscontrati e soluzioni adottate

-

Punto della situazione rispetto alla pianificazione

Il tempo è finito e noi abbiamo concluso il progetto.

Programma di massima per la prossima giornata di lavoro

-

GUIDA

Guida per l'uso semplificato di lejOS con Lego NXT

Näser Gübeli

Sommario

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:

Java SE Runtime Environment 8u201

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ● Decline License Agreement

Product / File Description	File Size	Download
Linux x86	68.1 MB	jre-8u201-linux-i586.rpm
Linux x86	83.8 MB	jre-8u201-linux-i586.tar.gz
Linux x64	64.91 MB	jre-8u201-linux-x64.rpm
Linux x64	80.73 MB	jre-8u201-linux-x64.tar.gz
Mac OS X x64	76.18 MB	jre-8u201-macosx-x64.dmg
Mac OS X x64	67.77 MB	jre-8u201-macosx-x64.tar.gz
Solaris SPARC 64-bit	46.27 MB	jre-8u201-solaris-sparcv9.tar.gz
Solaris x64	50.14 MB	jre-8u201-solaris-x64.tar.gz
Windows x86 Online	1.87 MB	jre-8u201-windows-i586-iftw.exe
Windows x86 Offline	63.53 MB	jre-8u201-windows-i586.exe
Windows x86	66.51 MB	jre-8u201-windows-i586.tar.gz
Windows x64	71.44 MB	jre-8u201-windows-x64.exe
Windows x64	71.29 MB	jre-8u201-windows-x64.tar.gz

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:

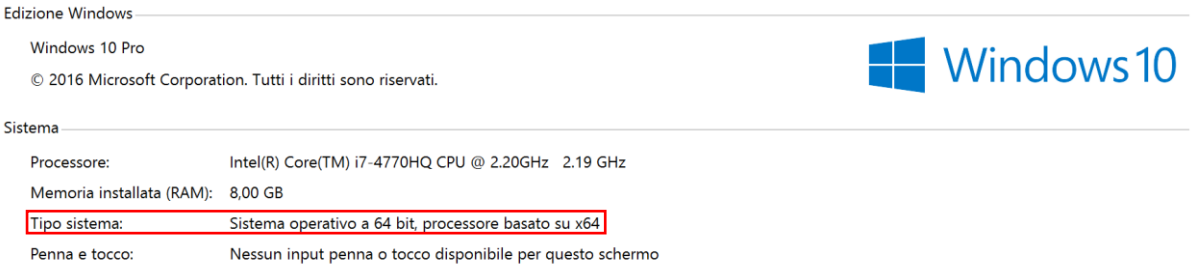
Java SE Runtime Environment 8u201

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux x86	68.1 MB	jre-8u201-linux-i586.rpm
Linux x86	83.8 MB	jre-8u201-linux-i586.tar.gz
Linux x64	64.91 MB	jre-8u201-linux-x64.rpm
Linux x64	80.73 MB	jre-8u201-linux-x64.tar.gz
Mac OS X x64	76.18 MB	jre-8u201-macosx-x64.dmg
Mac OS X x64	67.77 MB	jre-8u201-macosx-x64.tar.gz
Solaris SPARC 64-bit	46.27 MB	jre-8u201-solaris-sparcv9.tar.gz
Solaris x64	50.14 MB	jre-8u201-solaris-x64.tar.gz
Windows x86 Online	1.87 MB	jre-8u201-windows-i586-iftw.exe
Windows x86 Offline	63.53 MB	jre-8u201-windows-i586.exe
Windows x86	66.51 MB	jre-8u201-windows-i586.tar.gz
Windows x64	71.44 MB	jre-8u201-windows-x64.exe
Windows x64	71.29 MB	jre-8u201-windows-x64.tar.gz

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:

Java SE Runtime Environment 8u201

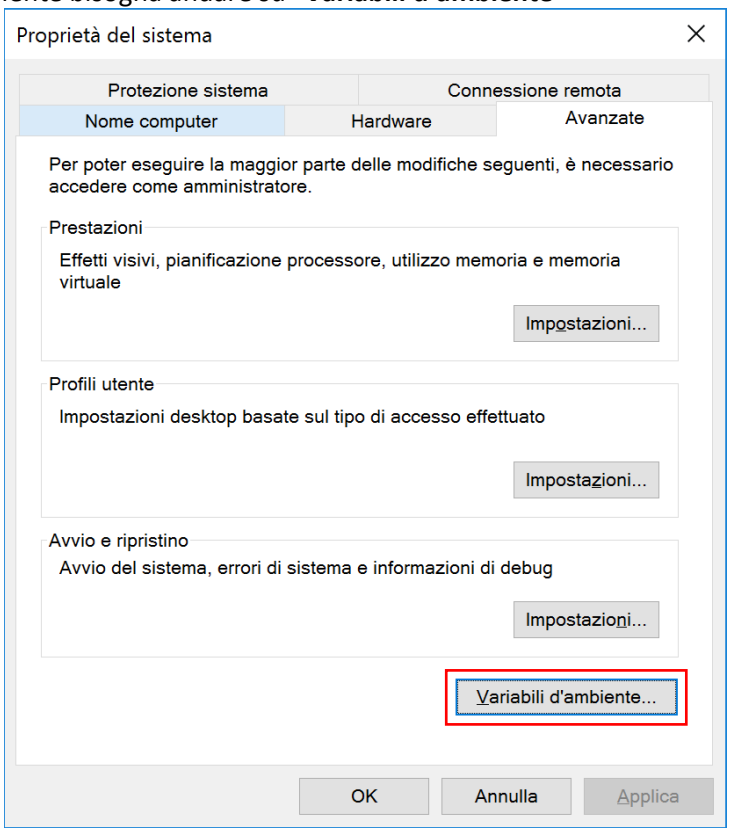
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux x86	68.1 MB	jre-8u201-linux-i586.rpm
Linux x86	83.8 MB	jre-8u201-linux-i586.tar.gz
Linux x64	64.91 MB	jre-8u201-linux-x64.rpm
Linux x64	80.73 MB	jre-8u201-linux-x64.tar.gz
Mac OS X x64	76.18 MB	jre-8u201-macosx-x64.dmg
Mac OS X x64	67.77 MB	jre-8u201-macosx-x64.tar.gz
Solaris SPARC 64-bit	46.27 MB	jre-8u201-solaris-sparcv9.tar.gz
Solaris x64	50.14 MB	jre-8u201-solaris-x64.tar.gz
Windows x86 Online	1.87 MB	jre-8u201-windows-i586-iftw.exe
Windows x86 Offline	63.53 MB	jre-8u201-windows-i586.exe
Windows x86	66.51 MB	jre-8u201-windows-i586.tar.gz
Windows x64	71.44 MB	jre-8u201-windows-x64.exe
Windows x64	71.29 MB	jre-8u201-windows-x64.tar.gz

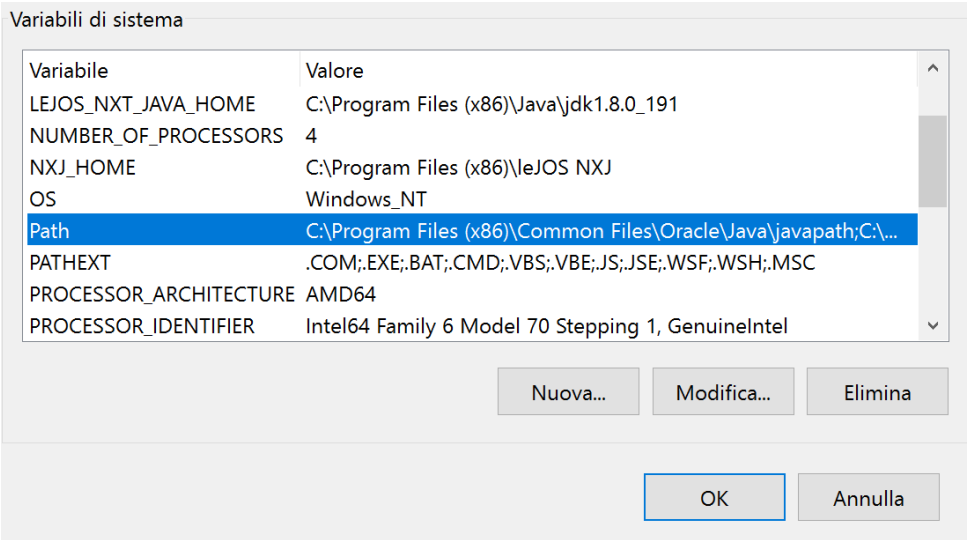
Fare questa procedura sia per la JDK che per la JRE.
Una volta scaricati entrambi, basta eseguirle i loro installer cliccando sempre su "Avanti".

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**"



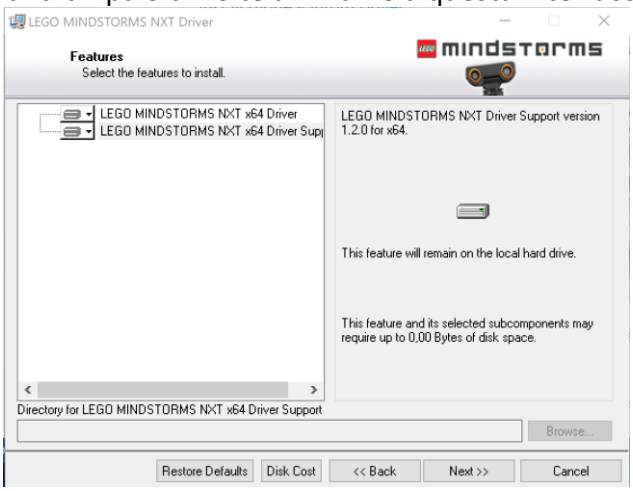
Infine basta cliccare il bottone "Nuovo" e immettere il percorso della cartella bin della JDK come ad esempio: **C:\Program Files\Java\jdk1.8.0_191\bin**.

Installazione Fantom Driver

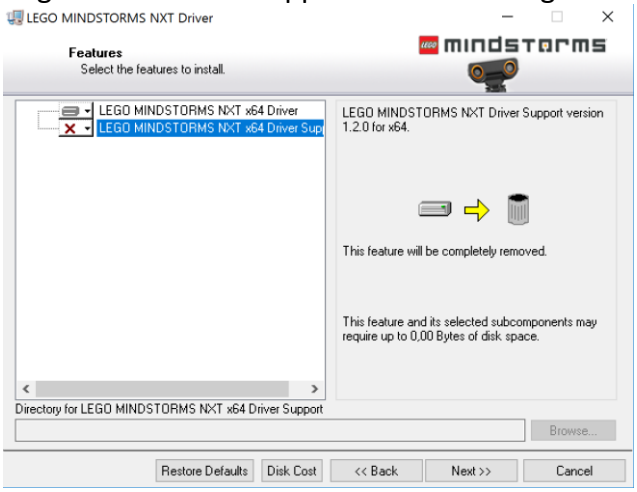
Un'ulteriore tool che ci serve per far funzionare lejOS ed NXT nel modo corretto è un piccolo programmino che si chiama Fantom driver, questo lo si scarica sul sito della lego: https://www.lego.com/en-us/mindstorms/downloads

Basta cliccare sulla scritta con "Download the NXT…" e il driver verrà scaricato.

Una volta scaricato bisogna estrarre il contenuto della cartella .zip ed eseguire il setup.
Dopo aver premuto avanti un paio di volte arriviamo a questa interfaccia:



Nella quale dobbiamo togliere il secondo supporto nel modo seguente:



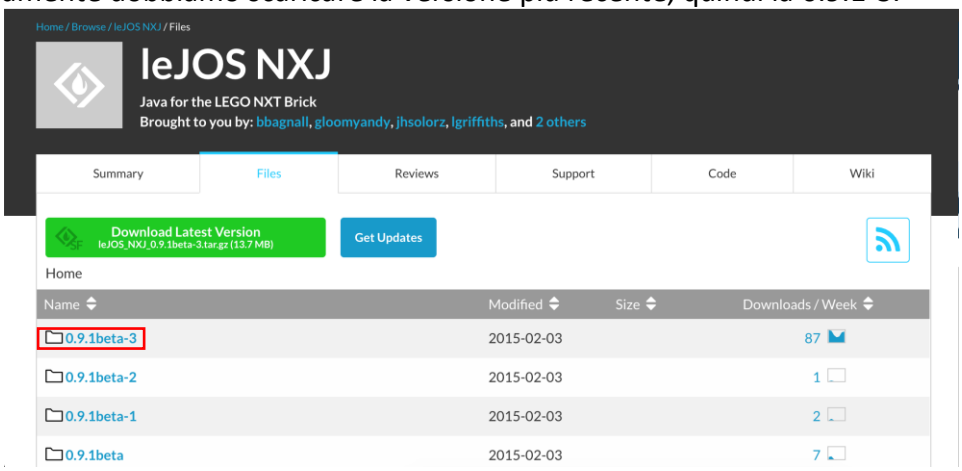
Come ultima cosa basta cliccare "Next" e "Finish".

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) ;	Costruttore 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)	Controlla e assegna il valore "port" alla variabile leftMotor. Valori accettati: 'A', 'B' o 'C'. *setMotorLeftPort('A');*
void setMotorRightPort(char port)	Controlla e assegna il valore "port" alla variabile rightMotor. Valori accettati: 'A', 'B' o 'C'. *setMotorRightPort('A');*
int getMySpeed()	Ritorna la velocità dei due motori.
void setMySpeed(int speed)	Controlla ed assegna il valore "speed" alla variabile speed dei due motori. Valori accettati: Consigliato da 0-100 *setMySpeed(75);*
char getDirection()	Ritorna la direzione nella quale va il robot, 'F' o 'B'.
void setDirection(char direction)	Controlla ed assegna la direzione in cui va il robot. Valori accettati: 'F' per avanti, 'B' per indietro. *setDirection('F');*
void move()	Metodo che fa partire i motori e quindi muovere il motore.
void left(int howMuch)	Fa curvare il robot a sinistra per i gradi definiti fa "howMuch". Valori accettati: Consigliato da 0-180 *left(10);*
void right(int howMuch)	Fa curvare il robot a destra per i gradi definiti fa "howMuch". Valori accettati: Consigliato da 0-180 *right(10);*
void stop()	Ferma i motori e quindi il robot.

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.