

Libreria LEGO | Diario di lavoro - 14.11.2018

Gabriele Alessi, Giulio Bosco

Canobbio, 14.11.2018

Lavori svolti

Durante questa giornata ci è stato consegnato il QdC del secondo progetto. Abbiamo dovuto scegliere tra due mandati (uno ogni 5 coppie) e noi abbiamo scelto: Sistema didattico per LEGO Mindstorms EV3 / NTX con libreria per attuatori. Mentre alcuni allievi svolgevano le presentazioni sul primo progetto appena concluso, gli altri hanno cominciato con l'analisi del QdC e scrivere eventuali domande per la lezione successiva.

Orario	Lavoro svolto
13:15 - 14:00	Introduzione al secondo progetto
14:00 - 14:45	Analisi e presentazioni

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Test Modulo 306, Analisi secondo progetto e presentazioni primo progetto.

Libreria LEGO | Diario di lavoro - 16.11.2018

Gabriele Alessi, Giulio Bosco

Canobbio, 16.11.2018

Lavori svolti

Oggi abbiamo svolto il secondo test del modulo 306 durante le prime due ore. Successivamente abbiamo potuto lavorare sul progetto ponendo delle domande al docente e eseguendo un po' di analisi e pianificazione. Durante le prossime occasioni definiremo la data di consegna e inizieremo a pensare più concretamente al metodo di lavoro da adottare durante l'implementazione.

Orario	Lavoro svolto
13:15 - 14:45	Test 2 Modulo 306
15:00 - 16:30	Analisi e pianificazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Analisi secondo progetto e presentazioni primo progetto.

Libreria LEGO | Diario di lavoro - 21.11.2018

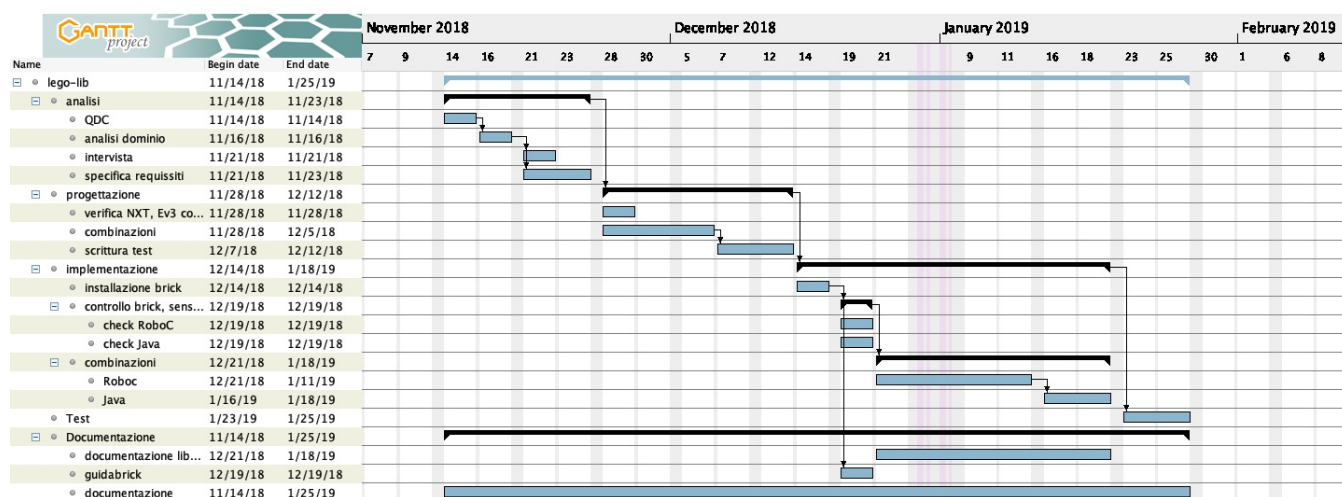
Gabriele Alessi, Giulio Bosco

Canobbio, 21.11.2018

Lavori svolti

Durante questa giornata abbiamo avuto l'occasione di ricevere le risposte del docente riguardo le domande tratte dopo aver analizzato il QdC. Successivamente abbiamo continuato con l'analisi del progetto iniziando la specifica dei requisiti e definendo la pianificazione. Qui di seguito c'è la prima versione del requisito 01 e il diagramma di Gantt.

ID	REQ-01
	Nome
	Priorità
	Versione
	Note



Orario	Lavoro svolto
13:15 - 14:45	Analisi e pianificazione

Problemi riscontrati e soluzioni adottate



Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Analisi secondo progetto.

Libreria LEGO | Diario di lavoro - 23.11.2018

Gabriele Alessi, Giulio Bosco

Canobbio, 23.11.2018

Lavori svolti

Oggi abbiamo continuato con il capitolo di analisi del progetto, che è praticamente concluso.

Nel frattempo, i docenti erano impegnati con le presentazioni del primo progetto, quindi eravamo liberi di lavorare come volevamo e su cosa volevamo.

Questa giornata è stata utile anche per effettuare un grande passo avanti con l'andamento del progetto, in quanto sono stati chiariti alcuni dubbi sulla progettazione durante qualche discussione con i compagni e domande ai docenti.

Di seguito si trova l'abstract del progetto, che verrà prossimamente tradotto in inglese.

*In questo documento è descritto come abbiamo sviluppato il prodotto "Libreria LEGO":
delle librerie utili per automatizzare le operazioni più comuni durante la programmazione di robot
LEGO®.*

*Ad esempio la lettura dei dati ricevuti da un sensore o le operazioni di movimento con degli
attuatori.*

*Queste librerie verranno utilizzate principalmente dagli informatici della classe seconda della
Scuola d'Arti e Mestieri di Trevano per sviluppare i programmi per la WRO (World Robot Olympiad)
e FLL (First LEGO League).*

*Questa raccolta di funzioni permetterà agli utenti di focalizzarsi sul problema principale da risolvere
avendo già le operazioni di base implementate e testate.*

Per concludere la giornata abbiamo fatto il diario e iniziato a pensare al capitolo di progettazione.

Orario	Lavoro svolto
13:15 - 16:30	Analisi

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro



Progettazione secondo progetto.

Libreria LEGO | Diario di lavoro - 28.11.2018

Gabriele Alessi, Giulio Bosco

Canobbio, 28.11.2018

Lavori svolti

Come pianificato, oggi abbiamo iniziato il capitolo di progettazione del secondo progetto.

Inizialmente abbiamo rivisto la documentazione per capire da dove cominciare e iniziare a scrivere alcune cose riguardo i componenti.

Poi abbiamo preso i brick NXT e provato a installare i firmware con cui avremo lavorato durante l'implementazione.

L'installazione di ROBOTC sembra essere andata a buon fine, ma con leJOS c'è stato qualche problema con l'installer e non siamo riusciti a finire.

A fine giornata ci sono stati dati i sensori e gli attuatori.

Orario	Lavoro svolto
13:15 - 14:45	Progettazione

Problemi riscontrati e soluzioni adottate

Problema con installer leJOS.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Progettazione secondo progetto e risoluzione problema leJOS.



Libreria LEGO | Diario di lavoro - 30.11.2018

Gabriele Alessi, Giulio Bosco

Canobbio, 30.11.2018

Lavori svolti

Oggi abbiamo continuato la progettazione del secondo progetto.

Ci è stato comunicato chi lavora su cosa (noi con leJOS), quindi abbiamo una situazione più chiara per quanto riguarda cosa fare e come muoversi.

Siamo riusciti a installare leJOS sul brick NXT, ma ci sono stati problemi nell'utilizzarlo con un IDE e eseguire del codice.

Orario	Lavoro svolto
13:15 - 16:30	Progettazione

Problemi riscontrati e soluzioni adottate

Problema con plugin IDE.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Progettazione secondo progetto e risoluzione problema leJOS.



Libreria LEGO | Diario di lavoro - 05.12.2018

Gabriele Alessi, Giulio Bosco

Canobbio, 05.12.2018

Lavori svolti

Durante questa giornata abbiamo risolto il problema con leJOS, riuscendo a compilare un programma (HelloWorld.java) e avviarlo nel nostro NXT. Per fare ciò è bastato scrivere il programma e eseguire i seguenti comandi:

```
nxjc HelloWorld.java
nxjlink -o HelloWorld.nxj HelloWorld
nxjupload -r HelloWorld.nxj
nxjupload -r HelloWorld.nxj
```

Poi abbiamo continuato progettando i metodi che implementeremo e inseriremo nella libreria finale.

Orario	Lavoro svolto
13:15 - 14:45	Progettazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Progettazione secondo progetto e risoluzione problema leJOS.

Libreria LEGO | Diario di lavoro - 07.12.2018

Gabriele Alessi, Giulio Bosco

Canobbio, 07.12.2018

Lavori svolti

Durante questa giornata abbiamo ancora lavorato sotto il capitolo di progettazione. Più precisamente stiamo svolgendo il diagramma e ideando come sarà composto il prodotto finale. Inoltre stiamo lavorando sulla guida (praticamente conclusa) per l'impostazione dell'ambiente di sviluppo in modo da saper utilizzare il prodotto ed eseguirne le installazioni necessarie.

Orario	Lavoro svolto
13:15 - 16:30	Progettazione e guida installazione leJOS

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Conclusione progettazione secondo progetto.

Libreria LEGO | Diario di lavoro - 12.12.2018

Gabriele Alessi, Giulio Bosco

Canobbio, 12.12.2018

Lavori svolti

Durante la scorsa settimana i docenti hanno ridefinito la consegna dei progetti: prima si doveva lavorare con entrambi i linguaggi (ROBOTC e Java), mentre ora il lavoro è più spartito tra le coppie in modo che ognuno faccia qualcosa di diverso (Java e NXT, Java e EV3, ROBOTC e NXT, ...).

Quindi è stato necessario adattare la documentazione alla nuova consegna: creare dei metodi di base con Java per NXT.

Nel frattempo abbiamo lavorato anche sulla progettazione iniziando a costruire i diagrammi e ideando i metodi.

Orario	Lavoro svolto
13:15 - 14:45	Progettazione e documentazione

Problemi riscontrati e soluzioni adottate

C'è stato un cambio di consegna quindi la documentazione è stata cambiata adattandola.

Punto della situazione rispetto alla pianificazione

Il cambio di consegna cambierà la pianificazione iniziale, quindi il cambiamento delle ore di lavoro sarà indicato nel consuntivo.

Programma di massima per la prossima giornata di lavoro

Conclusione progettazione secondo progetto.

Libreria LEGO | Diario di lavoro - 14.12.2018

Gabriele Alessi, Giulio Bosco

Canobbio, 14.12.2018

Lavori svolti

Oggi abbiamo concluso il capitolo di progettazione di questo progetto. Abbiamo concluso la guida per l'installazione dell'ambiente di sviluppo leJOS NXJ su Windows e il diagramma delle classi. È più o meno in chiaro la struttura delle classi per produrre il risultato finale, quindi durante le prossime giornate definiremo ciò e inizieremo con l'implementazione.

Orario	Lavoro svolto
13:15 - 16:30	Progettazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Introduzione implementazione.

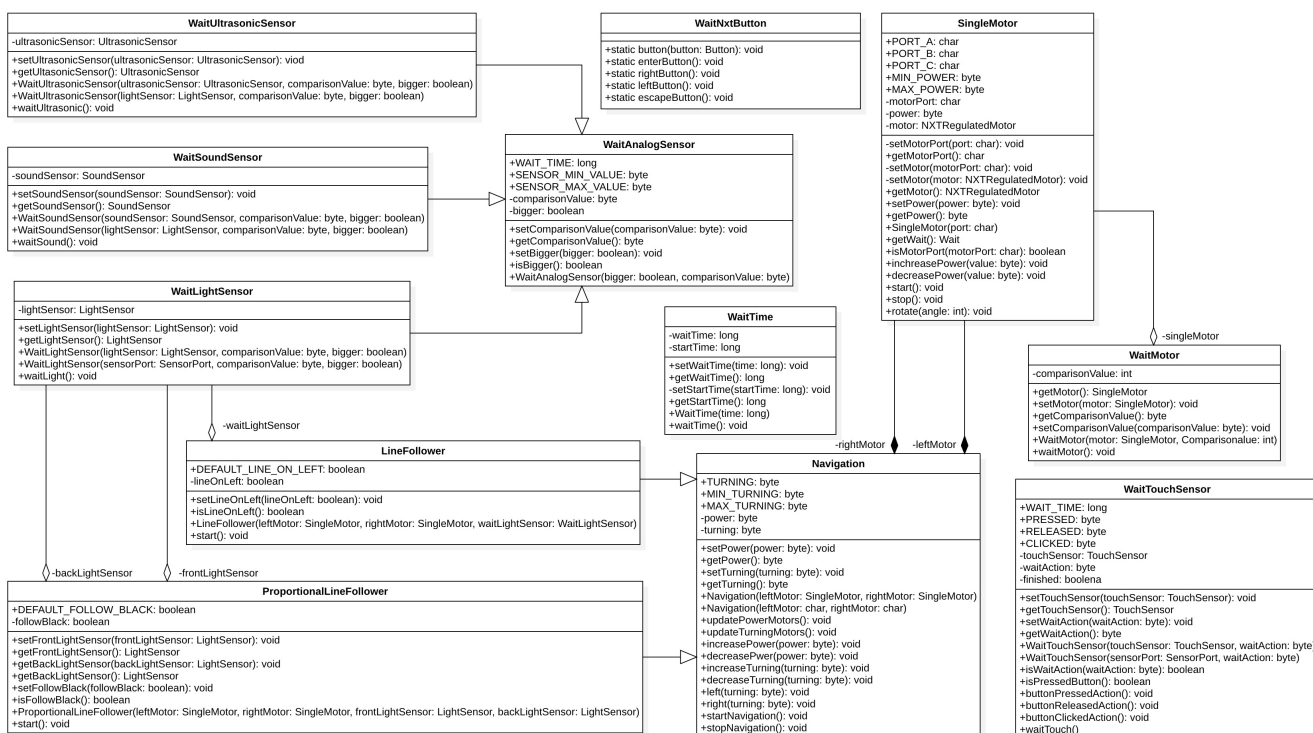
Libreria LEGO | Diario di lavoro - 19.12.2018

Gabriele Alessi, Giulio Bosco

Canobbio, 19.12.2018

Lavori svolti

Durante questa giornata abbiamo continuato con l'implementazione definendo le classi e i metodi della libreria e la documentazione.



Questo è il prototipo del diagramma delle classi e della struttura in generale, ma prossimamente verrà sicuramente aggiornato.

Orario	Lavoro svolto
13:15 - 14:45	Progettazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione



In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuazione implementazione metodi e doc.

Libreria LEGO | Diario di lavoro - 21.12.2018

Gabriele Alessi, Giulio Bosco

Canobbio, 21.12.2018

Lavori svolti

Oggi non abbiamo lavorato sul progetto perché abbiamo visto la teoria delle presentazioni per il modulo 306 e visto le presentazioni di alcuni compagni.

Orario	Lavoro svolto
13:15 - 14:45	Presentazioni modulo 306

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuazione implementazione metodi e doc.



Libreria LEGO | Diario di lavoro - 09.01.2019

Gabriele Alessi, Giulio Bosco

Canobbio, 09.01.2019

Lavori svolti

Durante la pausa invernale non abbiamo effettuato particolari progressi, quindi abbiamo innanzitutto fatto il punto della situazione per poi ricominciare a lavorare sull'implementazione.

Stiamo implementando in particolare le classi Wait.

Orario	Lavoro svolto
13:15 - 14:45	Implementazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuazione implementazione metodi e doc.

Libreria LEGO | Diario di lavoro - 11.01.2019

Gabriele Alessi, Giulio Bosco

Canobbio, 11.01.2019

Lavori svolti

Durante questa giornata abbiamo continuato con l'implementazione del progetto.

Ci siamo concentrati sulle classi `WaitSensor`, che sono composte fondamentalmente da un attributo che identifica il sensore e dei metodi generali che permettono di ricavarne il valore/stato. Inoltre vengono utilizzati metodi che vengono dal pacchetto `lejos.nxt`:

```
this.getTouchSensor().isPressed()
```

Il metodo `isPressed()` ritorna `true` o `false` se il sensore di pressione è premuto o meno.

Questo è un link che è stato utile per capire meglio il funzionamento dei sensori:

http://www.lejos.org/nxt/nxj/tutorial/LCD_Sensors/LCD_Sensors.htm.

Qui di seguito c'è il metodo di esecuzione principale del sensore di pressione:


```
public void run() {
    while (this.isFinished()) {
        try {
            if (this.getWaitAction() == PRESSED) {
                this.setFinished(this.getTouchSensor().isPressed());
            } else if (this.getWaitAction() == RELEASED) {
                if (this.getTouchSensor().isPressed()) {
                    while (this.getTouchSensor().isPressed()) {
                        Thread.sleep(WAIT_TIME);
                    }

                    this.setFinished(true);
                }
            } else if (this.getWaitAction() == CLICKED) {
                if (!this.getTouchSensor().isPressed()) {
                    while (this.getTouchSensor().isPressed()) {
                        Thread.sleep(WAIT_TIME);
                    }

                    this.setFinished(true);
                }
            }
            Thread.sleep(WAIT_TIME);
        } catch (InterruptedException ignored) {}
    }
}
```

Per quanto riguarda la documentazione siamo un po' fermi perché ci stiamo concentrando maggiormente sulla programmazione.

Bisogna dire che la struttura delle classi sta prendendo forma e alcune cose sono cambiate rispetto alla progettazione iniziale, quindi prossimamente verrà caricato il diagramma delle classi definitivo.

	SAMT - Sezione Informatica	Pagina: 2 / 2
	Lego Lib - Diario	

Orario	Lavoro svolto
13:15 - 16:30	Implementazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuazione implementazione metodi.

Libreria LEGO | Diario di lavoro - 16.01.2019

Gabriele Alessi, Giulio Bosco

Canobbio, 16.01.2019

Lavori svolti

Durante questa giornata abbiamo continuato con l'implementazione del progetto e anche documentato alcune classi.

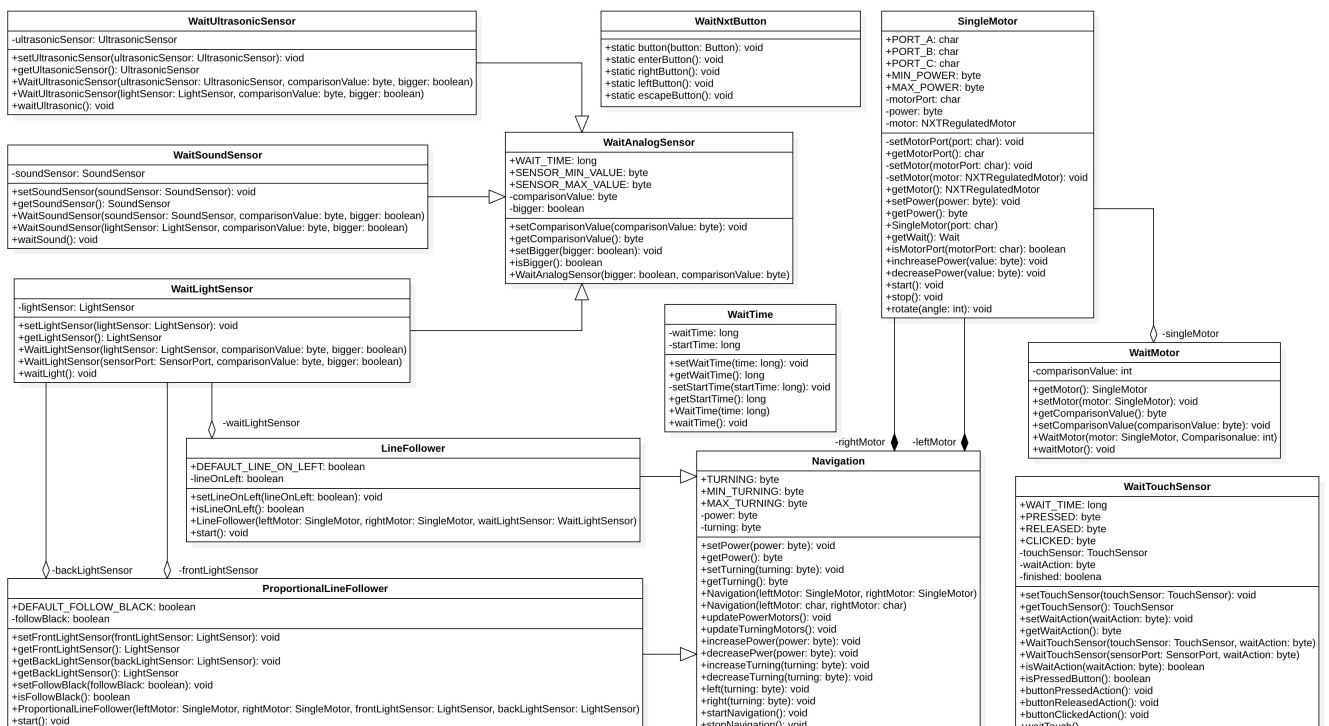
Ad esempio questa è la classe `WaitDigitalSensor`:

```

- PRESSED: È una costante che indica la pressione di un pulsante.
- RELEASED: È una costante che indica il rilascio di un pulsante.
- CLICKED: È una costante che indica il click di un pulsante.
- waitAction: Attributo che rappresenta l'azione eseguita sul pulsante (PRESSED, RELEASED, CLICKED).
- isWaitAction(): Metodo che indica se l'azione fatta sul pulsante è valida.
- getWaitAction(): Metodo che serve per ottenere l'azione sul pulsante.
- setWaitAction(): Metodo utile per impostare l'azione.
- Costruttore: Istanza un nuovo 'WaitDigitalSensor' impostando l'azione.
- isPressedButton(): Metodo che ritorna 'true' se il pulsante è premuto.
- buttonPressedAction(): Metodo che aspetta quando un pulsante è premuto.
- buttonReleasedAction(): Metodo che aspetta quando un pulsante è rilasciato.
- buttonClickedAction(): Metodo che aspetta quando un pulsante è cliccato (premuto e rilasciato).
- run(): È il metodo principale in cui si aspetta quando il pulsante viene cliccato.

```

È stato caricato il nuovo diagramma delle classi, in cui la struttura è la definizione delle classi sono più definiti e vicini al risultato finale.



E questa è l'implementazione del metodo principale della classe `WaitDigitalSensor`, dove si usano dei metodi di aiuto che identificano l'azione eseguita su un sensore digitale (pulsante) e aspetta in base a essa.

```
public void run() {
    while (this.isFinished()) {
        try {
            if (this.getWaitAction() == PRESSED) {
                this.buttonPressedAction();
            } else if (this.getWaitAction() == RELEASED) {
                this.buttonReleasedAction();
            } else if (this.getWaitAction() == CLICKED) {
                this.buttonClickedAction();
            }
            Thread.sleep(WAIT_TIME);
        } catch (InterruptedException ignored) {}
    }
}
```

Orario	Lavoro svolto
13:15 - 14:45	Implementazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Continuazione implementazione metodi e documentazione.

Libreria LEGO | Diario di lavoro - 18.01.2019

Gabriele Alessi, Giulio Bosco

Canobbio, 18.01.2019

Lavori svolti

Durante questa giornata abbiamo continuato con l'implementazione e la documentazione. Oggi era pianificato di terminare il capitolo, ma ci serviranno ancora alcuni giorni per perfezionare il tutto e procedere ai test. Nella documentazione stanno venendo descritti nel dettaglio tutte le classi con i relativi metodi.

Ad esempio oggi è stato implementato il LineFollower proporzionale e questa è la base del metodo principale:

```
public void run() {
    waitLightSensor.setValue((byte) 50);
    waitLightSensor.setBigger(this.isLineOnLeft());

    this.startNavigation();
    while (!isAlive()) {
        waitLightSensor.waiter();

        if (this.isLineOnLeft()) {
            this.right(TURNING);
        } else {
            this.left(TURNING);
        }

        waitLightSensor.setBigger(!waitLightSensor.isBigger());
        waitLightSensor.waiter();

        if (this.isLineOnLeft()) {
            this.left(TURNING);
        } else {
            this.right(TURNING);
        }
        waitLightSensor.setBigger(!waitLightSensor.isBigger());
    }
}
```

Dopo una rapida revisione generale, è stato creato un file in cui sono elencate le cose che dovranno venire perfezionate:

- Javadoc
- versions + authors
- use a default layout:
 - Constants
 - Fields
 - Getters
 - Setters
 - Constructors
 - Help Methods (private methods)
 - General Methods
- Rename some fields and methods

Questa è la documentazione della classe Wait, cioè la classe principale dei metodi di attesa rappresentati dal blocchetto arancione in Mindstorms:

Classe `Thread` usata per generalizzare tutte le classi `Wait`. Queste classi sono rappresentate tramite il blocco arancione in LEGO Mindstorms e servono per aspettare che succeda qualcosa prima di continuare la sequenza di azioni (ad esempio aspettare del tempo o un sensore che legga un certo valore o che cambi valore).

```

```

- `WAIT_TIME`: Costante che rappresenta il tempo (in millisecondi) da aspettare.
- `finished`: Attributo che indica se l'attesa è finita.
- `isFinished()`: Metodo utile per ottenere lo stato dell'attesa (finito/non finito).
- `setFinished()`: Metodo che serve per impostare lo stato dell'attesa.
- Costruttore: Istanza una nuova attesa.
- `beginWait()`: Imposta il valore dello stato dell'attesa a `true`.
- `waiter()`: Metodo principale utile per iniziare l'attesa.

Orario	Lavoro svolto
13:15 - 16:30	Implementazione e documentazione

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

Leggermente in ritardo con la pianificazione riguardo l'implementazione.

Programma di massima per la prossima giornata di lavoro

Messa a punto implementazione (Javadoc e revisione), documentazione e, se possibile, test.



Libreria LEGO | Diario di lavoro - 23.01.2019

Gabriele Alessi, Giulio Bosco

Canobbio, 23.01.2019

Lavori svolti

La scorsa settimana abbiamo finito di implementare le classi, quindi oggi cominceremo con i test modulari e proseguiremo con la documentazione. Nel frattempo guarderemo le classi che stiamo testando per perfezionare eventualmente qualcosa e mettere in ordine il codice e la Javadoc.

Il seguente è lo stile adottato per il codice e la Javadoc:

```
/*
 * The MIT License
 *
 * Copyright 2019 SAMT.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * ...
 */

package legolib.wait;

/**
 * Wait class, used to generalize all waiting classes.
 * In the LEGO Mindstorms environment is represented by the orange block "Wait".
 *
 * @author giuliobosco
 * @author gabrialessi
 * @version 1.1
 */
public class Wait extends Thread {

    // ----- Constants
    ...

    // ----- Fields
    ...

    // ----- Getters
    int getValue() { return ... }

    // ----- Setters
    void setValue(value) { ... }

    // ----- Constructors
    Wait(field1, field2) { ... }

    // ----- Help Methods
    ...

    // ----- General Methods
    ...

}
```

Poi è stato creato il package per i test delle classi, questa è la struttura delle classi di test:

```
public class WaitTouchSensorTest {  
    public static void main(String[] args) {  
        // Set a new touch sensor on port 1  
        TouchSensor touch = new TouchSensor(SensorPort.S1);  
  
        // Set the wait action to 2 (CLICKED)  
        byte action = 2;  
  
        // New wait for the touch sensor  
        WaitTouchSensor wait = new WaitTouchSensor(touch, action);  
  
        // Start the wait  
        wait.start();  
    }  
}
```

Nel frattempo stiamo anche lavorando nella creazione della guida per utilizzare il prodotto.

Orario	Lavoro svolto
13:15 - 14:45	Documentazione e test

Problemi riscontrati e soluzioni adottate

È stato riscontrato un problema riguardo l'import dei package nelle classi per fare i test. La prossima volta si troverà una soluzione e si definirà come eseguire i test e dove mettere i file generati dai test.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

Documentazione, revisione codice e test delle classi.

Libreria LEGO | Diario di lavoro - 25.01.2019

Gabriele Alessi, Giulio Bosco

Canobbio, 25.01.2019

Lavori svolti

Durante questa giornata ci è stato comunicato che la data di consegna è stata nuovamente posticipata di una settimana, quindi ciò cambierà il diagramma di Gantt consuntivo e avremo più tempo per fare i test e fare una documentazione migliore.

Stiamo lavorando su un metodo per convertire la documentazione da markdown a PDF che illustreremo prossimamente. Per il resto ci siamo concentrati sulla documentazione e la guida della del prodotto, visto che siamo ultimamente li abbiamo lasciati in secondo piano a causa dell'implementazione delle classi.

La seguente è la definizione messa nella documentazione della classe WaitLightSensor:

```
Classe figlia di `WaitAnalogSensor` utile per aspettare fino a quando si legge un certo valore con un sensore di luce. L'implementazione della classe non ha molte differenze rispetto alla sua classe superiore, infatti bisogna leggere un valore che va comparato con quello impostato dall'utente e si decide se deve essere maggiore o minore.
```

- lightSensor: Attributo che rappresenta il sensore di luce.
- getLightSensor(): Metodo utile per ottenere il sensore di luce.
- setLightSensor(): Metodo che serve per impostare il sensore di luce.
- WaitLightSensor(): Metodo costruttore, istanzia un nuovo `WaitLightSensor` impostando se il valore letto deve essere maggiore di quello inserito (`bigger`), il valore da comparare (`value`) e il sensore o la porta del brick in cui è inserito il sensore.
- run(): Metodo principale in cui si aspetta il giusto valore letto dal sensore di luce.

Inoltre sono state eliminate dal progetto i componenti riguardanti il sensore di colore (WaitAnalogSensorRange) visto che ci siamo resi conto che quest'ultimo non è disponibile.

Durante le prossime lezioni cercheremo di concludere definitivamente il capitolo di implementazione nella documentazione e iniziare a fare un po' di test.

Orario	Lavoro svolto
13:15 - 16:30	Documentazione


Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

In linea con la pianificazione.

Programma di massima per la prossima giornata di lavoro

	SAMT - Sezione Informatica	Pagina: 2 / 2
	Lego Lib - Diario	

Documentazione, revisione codice e test delle classi.

Libreria LEGO | Diario di lavoro - 30.01.2019

Gabriele Alessi, Giulio Bosco

Canobbio, 30.01.2019

Lavori svolti

Durante questa giornata abbiamo lavorato su documentazione, guida del prodotto e test modulari. Durante le prime due ore abbiamo capito come scrivere il codice di test e come effettivamente testarlo sul brick. Mentre nelle seconde ore è saltato fuori un problema riguardo la classe WaitTime, che non riesce a fare il confronto con i tempi per fare un concreto wait ed essere in grado di utilizzarli in modo sincrono e asincrono. Questo è il metodo della classe WaitTime su cui abbiamo lavorato di più e abbiamo modificato per vedere quale fosse il problema:

```
public void run() {
    this.setStartTime(System.currentTimeMillis());
    this.setFinished(false);
    System.out.println(this.getStartTime());
    while (!this.isFinished()) {
        try {
            System.out.print(System.currentTimeMillis()+" ");
            System.out.println(this.getStartTime() + this.getWaitTime());
            this.setFinished(
                this.getStartTime() + this.getWaitTime() >= System.currentTimeMillis());
            Thread.sleep(WAIT_TIME);
        } catch (InterruptedException ignored) {}
    }
}
```

Orario	Lavoro svolto
13:15 - 16:30	Documentazione e test

Problemi riscontrati e soluzioni adottate

Abbiamo riscontrato un problema riguardo i test delle classi, visto che la struttura del sistema è stata composta da vari package in modo da seguire gli standard di Java. Quindi abbiamo deciso di eliminare i package così da semplificare la procedura di test.

Abbiamo riscontrato anche dei problemi nelle comparazioni, tutti i tipi di comparazioni non lavorano correttamente

Punto della situazione rispetto alla pianificazione

Da recuperare il tempo perso con i test, è necessario accelerare su alcune cose, quindi probabilmente si lavorerà un po' al di fuori delle ore di lavoro.

<https://stackoverflow.com/questions/29899270/java-lejos-autonomous-nxj-robot-threads-causing-trouble> Su



questa pagina abbiamo scoperto che i NXT hanno problemi usando le threads.

Programma di massima per la prossima giornata di lavoro

Documentazione e test delle classi.

Provare ad eseguire del codice senza thread.

Libreria LEGO | Diario di lavoro - 01.02.2019

Gabriele Alessi, Giulio Bosco

Canobbio, 01.02.2019

Lavori svolti

La scorsa volta ci siamo lasciati con il problema delle classi Wait, che non funzionavano come avrebbero dovuto. Quindi a inizio giornata abbiamo provato la possibile soluzione al problema e abbiamo concluso che il brick non supporta correttamente le Threads. Dunque dobbiamo riscrivere tutte le classi senza usare le Thread e adattare tutta la documentazione.

Questo è il test che ci ha portato alla soluzione del problema:

```
System.out.println("inizio");
long s = System.currentTimeMillis();
sleep(5000);
if (System.currentTimeMillis() > s) System.out.println("prova");
System.out.println("finito");
```

Per effettuare i test ci affidiamo a un piccolo script che compila le classi e testa quella interessata caricandola sul NXT (in questo caso UseWaitTimeSynchron):

```
mkdir .\out\
nxjc *.java -d .\out\
cd out
nxjlink -o .\Test.nxj UseWaitTimeSynchron
nxjupload -r Test.nxj
```

La prima classe che abbiamo corretto è stata WaitTouchSensor e questa è la sua struttura:

- WAIT_TIME: Costante che definisce l'intervallo di tempo tra un controllo e un altro della fine dell'attesa.
- PRESSED: Costante che definisce la pressione del sensore.
- RELEASED: Costante che definisce il rilascio del sensore.
- CLICKED: Costante che definisce il click (pressione e rilascio) del sensore.
- touchSensor: Attributo che rappresenta il sensore di tocco.
- waitAction: Attributo che rappresenta l'azione da aspettare (premuto, rilasciato o cliccato).
- finished: Attributo interno che dice se l'attesa è finita.
- getTouchSensor(): Metodo che serve per ottenere il sensore di tocco.
- getWaitAction(): Metodo che serve per ottenere l'azione che si vuole aspettare.
- setTouchSensor(): Metodo utile per impostare il sensore di tocco.
- setWaitAction(): Metodo utile per impostare l'azione da aspettare.
- WaitTouchSensor(): Metodo costruttore, istanzia un nuovo `WaitTouchSensor` impostando l'azione (premuto, rilasciato, cliccato) e il sensore o la porta del brick in cui è inserito il sensore.
- isWaitAction(): Metodo utile per verificare che l'azione da aspettare imposta sia valida.
- isPressedButton(): Metodo che dice se il sensore è premuto.

- `buttonPressedAction()`: Metodo che aspetta la pressione del sensore.
- `buttonReleasedAction()`: Metodo che aspetta il rilascio del sensore.
- `buttonClickedAction()`: Metodo che aspetta il click (pressione e rilascio) del sensore.
- `waitTouchSensor()`: È il metodo principale che termina l'attesa in base all'azione impostata.

Invece questa è la descrizione della classe `WaitAnalogSensor`, che viene estesa dalle classi dei sensori di suono, di colore e ultrasuoni:

- `WAIT_TIME`: Costante che definisce l'intervallo di tempo tra un controllo e un altro della fine dell'attesa.
- `SENSOR_MIN_VALUE`: Costante che definisce il minimo valore che un sensore può leggere.
- `SENSOR_MAX_VALUE`: Costante che definisce il massimo valore che un sensore può leggere.
- `comparisonValue`: Attributo che rappresenta il valore da comparare con quello letto dal sensore.
- `bigger`: Attributo che indica se il valore letto deve essere maggiore o minore di quello di confronto.
- `getComparisonValue()`: Metodo che serve per ottenere il valore di confronto.
- `isBigger()`: Metodo utile per sapere il valore dell'attributo `bigger`.
- `setComparisonValue()`: Metodo utile per impostare il valore di confronto.
- `setBigger()`: Metodo utile per impostare il valore dell'attributo `bigger`.
- `WaitAnalogSensor()`: Metodo costruttore, istanzia un nuovo `WaitAnalogSensor`, definendo il campo `bigger` e il valore per comparare.

Bisogna dire che ora è molto più semplice usare le classi visto che praticamente non ci sono più dipendenze e ogni classe `Wait` funziona in base ai propri attributi e metodi.

A fine giornata siamo anche riusciti a finire di adattare tutte le classi e eliminare tutto quello che non è necessario, quindi se va tutto bene oggi termina l'implementazione.

Orario	Lavoro svolto
13:15 - 16:30	Documentazione e test

Problemi riscontrati e soluzioni adottate

Abbiamo scoperto il problema delle `Threads`, quindi lavoreremo per sistemare tutto cambiando la struttura delle classi.

Punto della situazione rispetto alla pianificazione

Ovviamente dopo ciò che è successo siamo in ritardo e faremo il possibile per far funzionare i moduli. Questo ci porterà sicuramente a lavorare al di fuori delle ore di lavoro.

Programma di massima per la prossima giornata di lavoro

Documentazione e test.

Libreria LEGO | Diario di lavoro - 06.02.2019

Gabriele Alessi, Giulio Bosco

Canobbio, 06.02.2019

Lavori svolti

Durante questa giornata ci siamo concentrati sulla documentazione, la guida e sui test modulari.

Il capitolo di implementazione è concluso e ora mancano principalmente i test. Questo è un esempio di classe documentata completamente:

WaitNxtButton

Classe utile per aspettare la pressione di uno dei pulsanti presenti sul brick NXT. L'implementazione è molto semplice poiché esiste la classe `Button` che contiene il necessario per far funzionare correttamente l'attesa.

WaitNxtButton
<pre>+static button(button: Button): void +static enterButton(): void +static rightButton(): void +static leftButton(): void +static escapeButton(): void</pre>

- `button()`: Metodo che aspetta la pressione del pulsante passato.

```
public static void button(Button button) {
    // waiting for pressing the button.
    button.waitForPress();
}
```

- `enterButton()`: Metodo che aspetta la pressione del pulsante centrale.

```
public static void enterButton() {
    button(Button.ENTER);
}
```

- `rightButton()`: Metodo che aspetta la pressione del pulsante destro.

```
public static void rightButton() {
    button(Button.RIGHT);
}
```

- `leftButton()`: Metodo che aspetta la pressione del pulsante sinistro.

```
public static void leftButton() {  
    button(Button.LEFT);  
}
```

- `escapeButton()`: Metodo che aspetta la pressione del pulsante in basso.

```
public static void escapeButton() {  
    button(Button.ESCAPE);  
}
```

Test WaitNxtButton

Per effettuare i test si usa `UseWaitNxtButton`, in cui semplicemente si usano i metodi della classe `WaitNxtButton`.

```
public static void main(String[] args) {  
    // Wait for the left button.  
    System.out.println("Press left button to continue");  
    WaitNxtButton.leftButton();  
    // Wait for the enter button.  
    System.out.println("Button pressed\n\nPress enter button to continue");  
    WaitNxtButton.enterButton();  
    // Wait for the right button.  
    System.out.println("Button pressed\n\nPress right button to continue");  
    WaitNxtButton.rightButton();  
    System.out.println("Button pressed\n\n");  
    // Wait for the escape button.  
    System.out.println("Button pressed\n\nPress escape button to continue");  
    WaitNxtButton.escapeButton();  
    System.out.println("Button pressed\n\n");  
    // Wait for another button to end the test.  
    System.out.println("Test over.");  
    Button.waitForAnyPress();  
}
```

I metodi funzionano correttamente perché il programma avanza al prossimo pulsante solo quando si preme quello richiesto.

Inoltre ci sarà da consegnare la presentazione, ma sarà fatta fuori dal posto perché ci servirà più tempo possibile per fare i test e tutto ciò che non possiamo fare a casa.

Orario	Lavoro svolto
13:15 - 16:30	Documentazione e test

Problemi riscontrati e soluzioni adottate

Ci sono stati alcuni problemi con i test ma sono stati risolti eseguendo un po' di prove.

Punto della situazione rispetto alla pianificazione



C'è qualcosa da recuperare con la documentazione e i test, ma si spera che per la prossima lezione sia tutto pronto se si lavora un po' in casa.

Programma di massima per la prossima giornata di lavoro

Conclusione e consegna progetto.