

# Sistema didattico per NXT

Bryan Beffa e Matteo Forni

## 1. Introduzione

### 1. Informazioni sul progetto

### 2. Abstract

### 3. Scopo

## 2. Analisi

### 1. Analisi del dominio

### 2. Analisi e specifica dei requisiti

### 3. Pianificazione

### 4. Analisi dei mezzi

## 3. Progettazione

### 1. Design dell'architettura del sistema

### 2. Implementazione

### 3. Libreria wait

### 4. LineFollower

## 4. Test

### 1. Protocollo di test

### 2. Risultati test

### 3. Mancanze/limitazioni conosciute

## 5. Consuntivo

## 6. Conclusioni

### 1. Sviluppi futuri

### 2. Considerazioni personali

## 7. Bibliografia

### 1. Sitografia

## 8. Allegati



## Introduzione

### Informazioni sul progetto

- Allievi: Bryan Beffa e Matteo Forni
- Docenti: Adriano Barchi, Luca Muggiasca, Francesco Mussi, Massimo Sartori
- SAM Trevano, Informatica, I3AA
- 14.11.2018 - 08.02.2019

### Abstract

*The project requirement is to create a java and a robotC library. In these libraries, we have to put some useful classes for the guys of the second year. These classes will be used for the WRO, so they will have some methods for every actuator and sensor. These libraries should be universal and user-friendly, so the guys could use it in a simple way for their interests.*

### Scopo

Lo scopo di questo progetto è quello di creare due librerie, una in RobotC e l'altra in Java sia per bricks EV3 sia per NXT, per aiutare gli allievi del secondo anno di informatica che parteciperanno alla gara di robotica WRO. Esse permetteranno di eseguire le operazioni basiliari con i motori e con i seguenti sensori:

- Sensore tattile
- Sensore di luce/colore
- Giroscopio
- Sensore ad ultrasuoni
- Sensore di suono

Al termine della creazione delle librerie dovrà essere eseguito un test di performance per determinare quale tra le due librerie funziona meglio.

### Analisi

#### Analisi del dominio

La scuola necessita di una libreria per gli allievi di informatica del secondo anno, essa servirà a controllare i robot Ev3 e NXT durante la competizione di robotica WRO. La libreria dovrà contenere del codice per il controllo degli attuatori e dei sensori così da aiutare l'utente con l'uso dei Robot Mindstorm, essa dovrà essere scritta in RobotC e in Java e dovrà poi venire valutato quale dei due linguaggi è il migliore e più performante. La libreria dovrà essere di facile comprensione e modifica per gli allievi, dovrà quindi comprendere una guida all'utilizzo. Attualmente, per i ragazzi del secondo anno, non ci sono altri prodotti che svolgono questo compito. Le conoscenze che saranno necessarie per utilizzare il prodotto saranno quelle di base della programmazione.

#### Analisi e specifica dei requisiti

Il committente necessita di librerie utili per l'utilizzo dei robot LEGO Mindstorm. Le librerie devono contenere metodi utili per ogni sensore/attuatore. Queste librerie devono essere scritte in RobotC e java. I metodi devono essere di facile utilizzo (user-friendly) e dovranno essere compatibili con robot di diverse forme e dimensioni. Per ogni attuatore/sensore sono disponibili più metodi che effettuano azioni distinte. Queste librerie servono per aiutare gli allievi del secondo anno durante la WRO.

ID	REQ-01
<b>Nome</b>	Classe sensore ultrasuono Java
<b>Priorità</b>	1
<b>Versione</b>	2.0
<b>Note</b>	Il requisito è stato eliminato.

ID	REQ-02
<b>Nome</b>	Classe sensore di colore Java
<b>Priorità</b>	1
<b>Versione</b>	2.0
<b>Note</b>	Il requisito è stato eliminato.



ID	REQ-03
<b>Nome</b>	Classe sensore di tatto Java
<b>Priorità</b>	1
<b>Versione</b>	2.0
<b>Note</b>	Il requisito è stato eliminato.

ID	REQ-04
<b>Nome</b>	Classe sensore giroscopio Java
<b>Priorità</b>	1
<b>Versione</b>	2.0
<b>Note</b>	Il requisito è stato eliminato.

ID	REQ-05
<b>Nome</b>	Classe sensore di luce Java
<b>Priorità</b>	1
<b>Versione</b>	2.0
<b>Note</b>	Il requisito è stato eliminato.

ID	REQ-06
<b>Nome</b>	Classe motore principale Java
<b>Priorità</b>	1
<b>Versione</b>	2.0
<b>Note</b>	Il requisito è stato eliminato.

ID	REQ-07
<b>Nome</b>	Classe motore secondario Java
<b>Priorità</b>	1
<b>Versione</b>	2.0
<b>Note</b>	Il requisito è stato eliminato.

ID	REQ-08
<b>Nome</b>	Classe sensore di suono Java
<b>Priorità</b>	1
<b>Versione</b>	2.0
<b>Note</b>	Il requisito è stato eliminato.

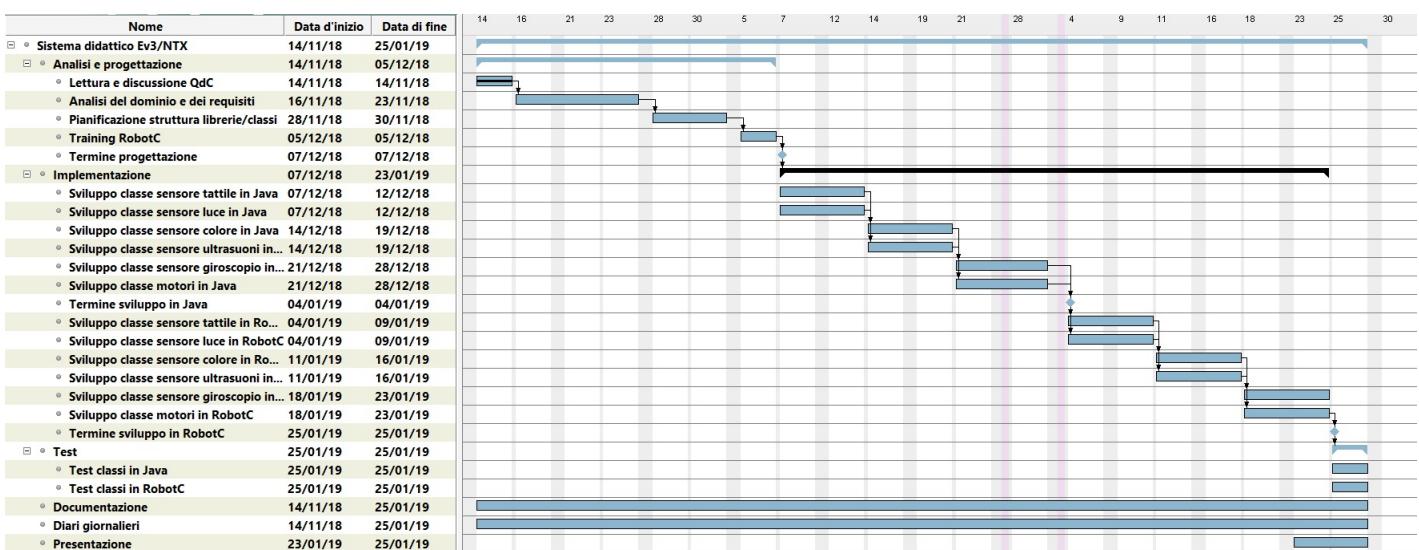


ID	REQ-09
Nome	Libreria wait
Priorità	1
Versione	1.0
Note	Libreria che riporta i metodi del blocchetto wait del linguaggio visivo di NXT.
	<b>Sotto requisiti</b>
001	Metodo di attesa del suono.
002	Metodo di attesa della distanza rilevata con l'ultrasuoni.
003	Metodo di attesa del tocco.
004	Metodo di attesa della luce.
005	Metodo di attesa di un lasso di tempo.
006	Metodo di attesa del movimento in rotazioni dei motori.
007	Metodo di attesa del movimento in gradi dei motori.

ID	REQ-10
Nome	Progetto di dimostrazione e test
Priorità	1
Versione	1.0
Note	Mini progetto che dimostra l'uso della libreria e ne testa il funzionamento.
	<b>Sotto requisiti</b>
001	Line follower percentuale.
002	Controllo della presenza di oggetti davanti a se.

## Pianificazione

Qui di seguito si può vedere il Gantt iniziale, nell'immagine si nota come era stato pensato di suddividere i metodi di entrambi i linguaggi fra i due componenti del gruppo e come questa parte fosse stata stimata come la più lunga del progetto. In questa progettazione non era presente nemmeno il programma dimostrativo.



## Analisi dei mezzi

### Software

I software che sono stati utilizzati sono i seguenti:

- LEGO Mindstorm NXT
- RobotC IDE
- Microsoft Word 2016
- Microsoft PowerPoint 2016
- Atom 1.34.0 x64

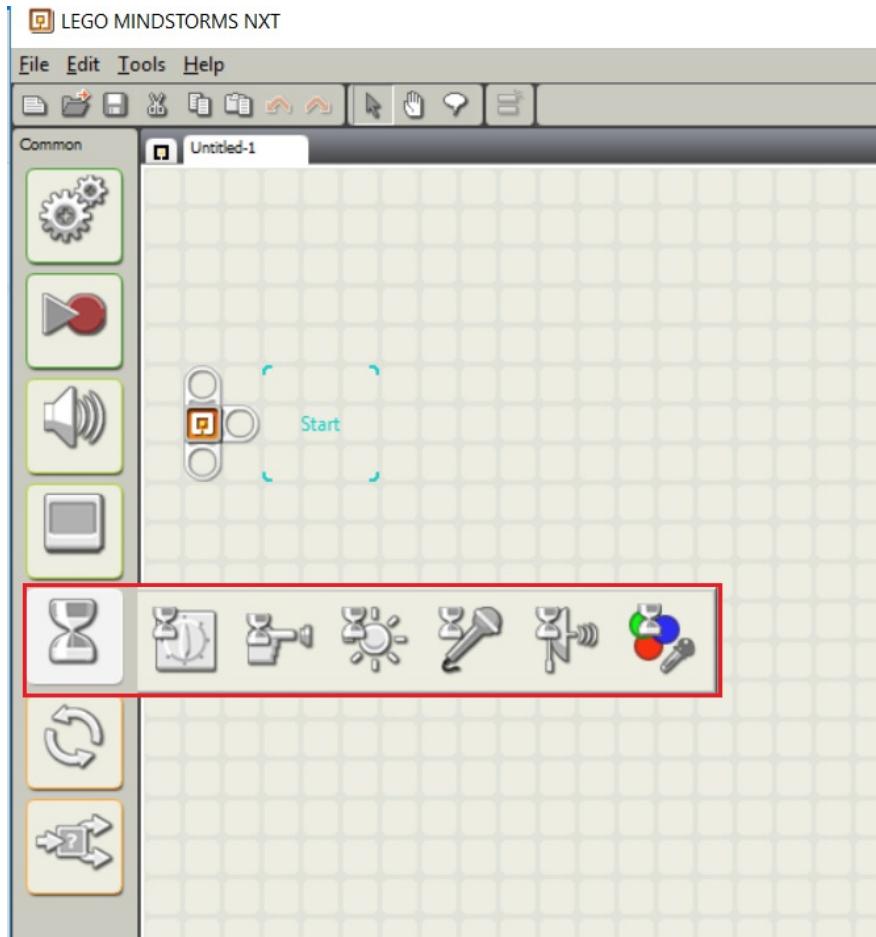
### Hardware

In questo progetto è stato utilizzato il seguente hardware:

- Portatili (specifiche tecniche nella media) con Windows 10
- Robot Lego NXT brick
- Sensore di luce
- Sensore ultrasuono
- Sensore di tatto
- Motore principale
- Motore secondario

## Progettazione

Durante i colloqui con i docenti l'idea di base del progetto è cambiata. Con i vari gruppi ci siamo suddivisi i compiti così che noi dobbiamo fare solamente una libreria scritta in RobotC. Questa libreria dovrà contenere i metodi di attesa del blocchetto del linguaggio visivo di NXT. Dovrà inoltre venire fatto un programma di esempio per dimostrare l'utilizzo dei metodi così come il loro funzionamento. La libreria da creare dovrà contenere sette metodi complementari a quelli del blocchetto wait del linguaggio visivo NXT, mostrati nella foto seguente, senza il metodo di riconoscimento del colore che non viene scritto dato che i sensori non possono distinguere i colori.

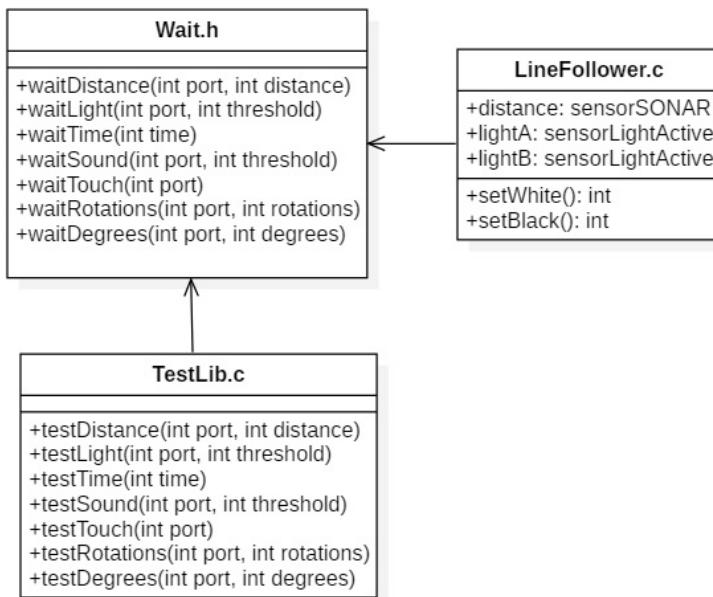


Oltre a ciò la libreria contiene tre metodi che rappresentano i movimenti del motore nelle sue tre opzioni pari a quelle del linguaggio visivo: per rotazioni, per secondi o per gradi.

Come seconda parte del progetto dovrà venire scritto un programma di test di alcuni dei metodi della libreria e del linguaggio RobotC in se così da verificare se esso porta dei vantaggi o meno. Il programma sarà un line follower proporzionale che si ferma ad una determinata distanza da un oggetto. Il robot avrà due sensori di luce, uno a destra e l'altro a sinistra della linea, ed uno di ultrasuoni che controllerà che non ci siano oggetti davanti a sé e se ne rileva fa muovere un terzo motore.

## Design dell'architettura del sistema

Qui di seguito si può vedere il diagramma delle classi, in esso vi sono tre classi: la libreria di wait, la classe di test della libreria e la classe che esegue il line follower. Entrambe le due classi utilizzano la libreria.



## Implementazione

### Libreria wait

La libreria wait, scritta nel linguaggio RobotC, definisce dei metodi d'attesa utili per LEGO Robot NXT.  
I metodi che abbiamo definito sono:

- waitTouch(int)
- waitLight(int, int, bool)
- waitDistance(int, int)
- waitSound(int, int)
- waitTime(long)
- waitRotations(int, int, int)
- waitDegrees(int, int, int)

#### waitTouch(int port)

Il metodo d'attesa waitTouch(int port) richiede come parametro la porta a cui è collegato il sensore. Questo metodo attende che il sensore di tatto venga premuto.

Esso è molto semplice. Viene dichiarata una variabile booleana che indica se il sensore di tatto è premuto, quando il sensore viene premuto il valore letto corrisponde "0" e alla variabile booleana viene assegnato il valore *true* in modo che il ciclo *while* termini.

```
/*
 * Metodo che attende che il sensore di tatto venga premuto
 *
 * @param port porta a cui è collegato il sensore di tatto
 */
void waitTouch(int port){
    bool isPressed = false;

    while(!isPressed){
        //controllo se il sensore è stato premuto
        if(SensorValue[port] > 0){
            isPressed = !isPressed;
        }
    }
}
```

Esempio di utilizzo:

Quando il sensore di tocco viene premuto il robot NXT emette un suono.



```
waitLight(int port, int threshold, bool higher)
```

Il metodo `waitLight(int port, int threshold, bool higher)` attende che il sensore di luce rilevi un valore superiore alla soglia `threshold` se al parametro `higher` è stato assegnato il valore `true` oppure inferiore alla soglia `threshold` se il parametro `higher` corrisponde a `false`.

Quando accade una delle due situazioni, la variabile booleana `flag`, utilizzata per ripetere il ciclo, viene settata a `false` così da terminare il ciclo.

```
/**  
 * Metodo che attende che il sensore di luce rilevi un valore sotto la soglia desiderata  
 *  
 * @param port porta a cui è collegato il sensore di luce  
 * @param threshold soglia di luce minima/massima  
 * @param higher valore booleano che determina se bisognerà attendere un valore superiore o inferiore alla soglia.  
 */  
bool waitLight(int port, int threshold, bool higher){  
    bool flag = true;  
  
    while(flag){  
        //controllo che il valore letto del sensore non sia sotto la soglia minima  
        if(higher){  
            if(SensorValue(port) > threshold){  
                flag = !flag;  
            }  
        }else{  
            if(SensorValue(port) < threshold){  
                flag = !flag;  
            }  
        }  
    }  
}
```

Esempio di utilizzo:

Il robot avanza fino al rilevamento del colore nero.

```
waitDistance(int port, int distance)
```

Il metodo `waitDistance(int port, int distance)` attende che venga rilevato, dal sensore di distanza (ultrasuoni), un valore inferiore a quello passato come parametro `distance`.

Questo metodo viene spesso utilizzato durante il movimento di un robot per farlo fermare ad una determinata distanza da un oggetto.

Il metodo, ritornando un valore booleano, deve essere chiamato all'interno di un ciclo così ad ogni ripetizione viene controllato se la distanza è inferiore o maggiore alla soglia determinata.

```
/**  
 * Metodo che attende che il sensore di distanza rilevi una distanza inferiore alla soglia  
 *  
 * @param port porta a cui è collegato il sensore infrarosso  
 * @param distance distanza minima da un oggetto  
 * @return se la distanza è inferiore alla soglia  
 */  
bool waitDistance(int port, int distance){  
    //controllo che il valore letto non sia inferiore alla soglia minima  
    if(SensorValue(port) <= distance){  
        return false;  
    }  
    return true;  
}
```

Esempio di utilizzo:

Il robot avanza fino ad arrivare alla distanza decisa dall'utente da un oggetto, si ferma, e torna indietro.

**waitSound(int port, int threshold)**

Questo metodo d'attesa ha lo stesso principio del metodo `waitDistance()` con la differenza che termina il ciclo quando il sensore di suono rileva un valore superiore alla soglia `threshold` passata come parametro.

```
/*
 * Metodo che attende che il sensore di suono rilevi un suono che superi la soglia desiderata
 *
 * @param port porta a cui è collegato il sensore di suono
 * @param threshold soglia del suono
 */
void waitSound(int port, int threshold){
    bool flag = true;
    while(flag){
        //controllo che il sensore di suono non rilevi un suono che superi la soglia
        if(SensorValue[port] >= threshold){
            flag = !flag;
        }
    }
}
```

Esempio di utilizzo:

Il robot avanza fino a quando rileva un valore superiore alla soglia si ferma.

**waitTime(long millis)**

Il metodo `waitTime` è molto semplice, attende il tempo in millesekondi passato come parametro `millis`.

Il codice è molto semplice ed è formato da un solo metodo al suo interno, abbiamo deciso comunque di inserirlo nella libreria per la sua completezza generale.

```
/*
 * Metodo che permette l'attesa di un certo numero di millisecondi
 *
 * @param millis millisecondi di attesa desiderati
 */
void waitTime(long millis){
    //aspetto i millesekondi desiderati
    wait1Msec(millis);
}
```

Esempio di utilizzo:

Il robot emette un suono per 2 secondi.

**waitRotations(int port, int times, int speed)**

Il metodo di attesa `waitRotations(int port, int times, int speed)` attende che il motore, collegato alla porta passata come parametro, effettui il numero di rotazioni specificate alla velocità desiderata.

```
/*
 * Metodo che attende che il motore passato come parametro svolga il numero di rotazioni desiderato
 *
 * @param port porta a cui è collegato il motore
 * @param times rotazioni che il motore deve svolgere
 * @param speed velocità desiderata del motore
 */
void waitRotations(int port, int times, int speed){
    //calcolo il numero di rotazioni in gradi
    int degree = 360*times;
    setMotorTarget(port, degree, speed);
    waitUntilMotorStop(port);
}
```

Esempio di utilizzo:

Il robot deve avanzare di 5 rotazioni del motore.

waitDegrees(int port, int degree, int speed)

Questo metodo attende che il motore, passato come parametro, effettui la rotazione dei gradi specificati come parametro *degree*.

```
/*
 * Metodo che permette la rotazione del motore del numero di gradi desiderato
 *
 * @param port porta a cui è collegato il motore
 * @param degree gradi di rotazione del motore desiderati
 * @param speed velocità del motore desiderata
 * @param millis millisecondi di attesa per permettere la rotazione dei gradi desiderati (scegliere un valore che permetta la rotazione completa)
 */
void waitDegrees(int port, int degree, int speed){
    setMotorTarget(port, degree, speed);
    waitUntilMotorStop(port);
}
```

Esempio di utilizzo:

Il motore del robot deve effettuare una rotazione di 90°.

## LineFollower

Abbiamo realizzato un line follower utilizzando alcuni dei metodi che abbiamo definito nella libreria wait.

Il programma esegue un LineFollower proporzionale con l'utilizzo di:

- due sensori di luce
- un sensore ultrasuono
- terzo motore (braccio meccanico)

### Descrizione

Il robot deve, dopo che sono stati configurati i valori per il colore bianco ed il colore nero, seguire la linea nera. Se il blocchetto trova un ostacolo ad una distanza, determinata dall'utente, si ferma e fa girare il braccio meccanico per poi terminare il programma.

Come prima cosa si setta il colore bianco ed il colore nero tramite dei metodi che abbiamo creato setWhite() and setBlack().

```
//variabili in cui viene salvato il valore del colore bianco e nero
int white;
int black;

//setto il bianco e il nero
white = setWhite();
black = setBlack();
```

I due metodi sono simili e chiedono all'utente di posizionare il robot sul rispettivo colore e premere il tasto descritto per settare il valore.

```
int setBlack()
{
    int black;
    while (true)
    {
        //stampo sul display
        nxtDisplayCenteredTextLine(1, "Premi la freccia");
        nxtDisplayCenteredTextLine(2, "sinistra per");
        nxtDisplayCenteredTextLine(3, "selezionare");
        nxtDisplayCenteredTextLine(4, "il colore nero");
        //premere il tasto sinistra per selezionare il colore nero
        if (nNxtButtonPressed == 2)
        {
            black = (SensorValue[lightA]+SensorValue[lightB])/2;
            return black;
        }
    }
}
```

Successivamente il robot inizia a svolgere il line follower.

```
while(waitDistance(distance, 20)){
    //margin consentito
    double margin = 1.5;

    //soglia di luce riflessa
    int threshold = 50;

    //calcolo l'errore
    int errorA = SensorValue[lightA] - (white+black)/2;
    int errorB = SensorValue[lightB] - (white+black)/2;

    //calcolo la velocità
    double speedA = errorA * margin;
    double speedB = errorB * margin;

    //setto la velocità al motore collegato alla porta A e alla porta B
    motor[motorB] = speedB+15;
    motor[motorA] = speedA+15;
    waitTime(10);
}
```

Come prima cosa viene definita la variabile *margin* che definisce la precisione del line follower. Più il valore è alto e più la variazione di velocità aumenta.

Viene definita una soglia, calcolando semplicemente la media tra il colore bianco e il colore nero, che serve a calcolare l'errore.

Per calcolare la velocità viene effettuato il calcolo *errorA* e rispettivamente *errorB* moltiplicato per il *margin*.

Infine viene assegnata ai due motori la velocità calcolata in precedenza e viene utilizzata l'attesa di 10 milisecondi tramite il metodo della libreria *wait*.

```
while(waitDistance(distance, 20)){
    ...
}
```

All'interno del while vi è il metodo *waitDistance* (vedi metodo *waitDistance(int port, int threshold)*) in cui ad ogni ciclo viene controllato se la distanza minima da un oggetto è rispettata o meno.

Infine se la distanza è inferiore alla soglia il robot fa ruotare di 160 gradi, a velocità di -20, il motore che fa da braccio.

```
while(waitDistance(distance, 20)){
    ...
    motor[motorB] = 0;
    motor[motorA] = 0;

    //muovo il motore secondario
    waitDegrees(motorC, 160, -20);
```

## Test

### Protocollo di test

Test Case	TC-001
Nome	Test di funzionamento della libreria di <i>wait</i>
Riferimento	REQ-09
Descrizione	Bisogna testare il corretto funzionamento della libreria di <i>wait</i> .
Prerequisiti	Bisogna avere il robot con il firmware di robotc installato, collegato al PC con i sensori e/o gli attuatori necessari collegati ad esso.
Procedura	Bisognerà aprire la classe di test <i>TestLibrary.c</i> e, per ogni metodo della libreria <i>Wait.h</i> , Bisognerà collegare al robot i sensori/attuatori giusti, decommentare il metodo di test ed eseguirlo. Se il robot attende la soglia o il valore atteso allora il metodo richiamato funzionerà correttamente
Risultati attesi	Con tutti i metodi richiamati il robot dovrebbe attendere il valore o la soglia e poi terminare il programma.



Test Case	TC-002
Nome	Test di funzionamento del line follower proporzionale
Riferimento	REQ-10
Descrizione	Bisogna testare se la classe LineFollower.c funziona correttamente.
Prerequisiti	Si necessita di avere un robot con il firmware di robotc installato e con collegati due sensori di luce e uno di ultrasuoni.
Procedura	Bisognerà compilare il codice dall'ambiente di sviluppo di robotc sul robot appositamente collegato al computer. Fatto ciò si eseguirà il programma, si seguiranno le istruzioni stampate a schermo ed infine si posizionerà il robot sopra una linea nera.
Risultati attesi	Il robot dovrebbe iniziare a seguire la linea nera.

## Risultati test

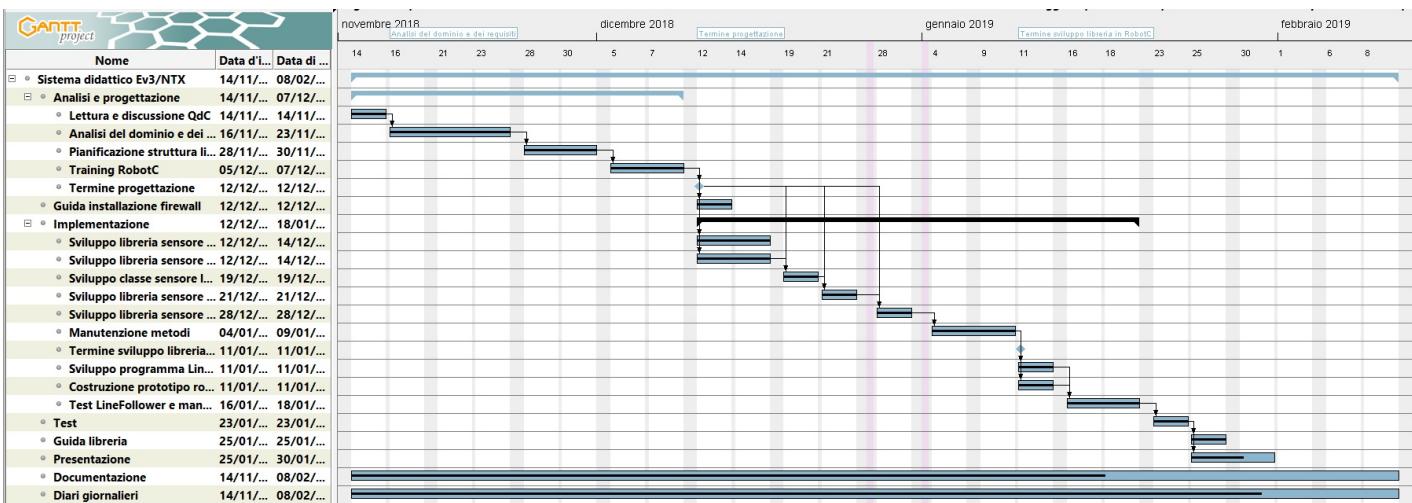
Risultati dei test			
Test case	Risultato	Note	
TC-001	successo	-	
TC-002	successo	-	

## Mancanze/limitazioni conosciute

Il progetto presenta alcune limitazioni che però sono date dall'hardware e non dalla libreria che abbiamo creato. La precisione dei sensori è approssimativa e può capitare che le letture dei vari sensori non corrispondano al valore reale. Abbiamo inoltre notato che il sensore ultrasuono, se non si lascia una distanza minima adeguata (5 - 6 centimetri), non riesce a misurare il percorso che intercalata tra esso e l'ostacolo.

## Consuntivo

Come si può notare le differenze con la prima progettazione sono molte dati i cambiamenti nei requisiti che ci sono stati con il passare del tempo. Durante l'implementazione abbiamo provato a dividerci i compiti ma per buona parte del tempo abbiamo lavorato insieme per evitare malfunzionamenti tra le soluzioni. La parte che ha comunque preso più tempo è stata la creazione della libreria a cui si è aggiunta la scrittura del codice per il programma line follower.



## Conclusioni

Questa soluzione certamente non cambierà il mondo ma superiamo che possa però essere d'aiuto ai ragazzi di seconda durante le loro competizioni. Personalmente lo riteniamo un successo perché, anche se l'obiettivo di questo progetto non era quello di diventare qualcosa di rivoluzionario ma qualcosa da utilizzare per facilitarsi la vita e velocizzare la scrittura del codice, esso fa esattamente quello per cui era stato pensato.

## Sviluppi futuri

Migliorie al prodotto si possono fare aggiungendo dei metodi che svolgono programmi più complessi magari aspettando delle combinazioni di sensori ma non sappiamo se questo potrebbe tornare utile e se esso fosse necessario può essere facilmente sviluppato dagli allievi interessati.

## Considerazioni personali

Da questo progetto abbiamo imparato che ci sono svariati linguaggi di programmazione marginali, ma che se conosciuti possono aiutare con cose che se sviluppate senza di essi risultano complicate. Noi personalmente, anche se siamo sempre stati scettici dell'utilità del prodotto finale, siamo soddisfatti dai risultati ottenuti.

## Bibliografia

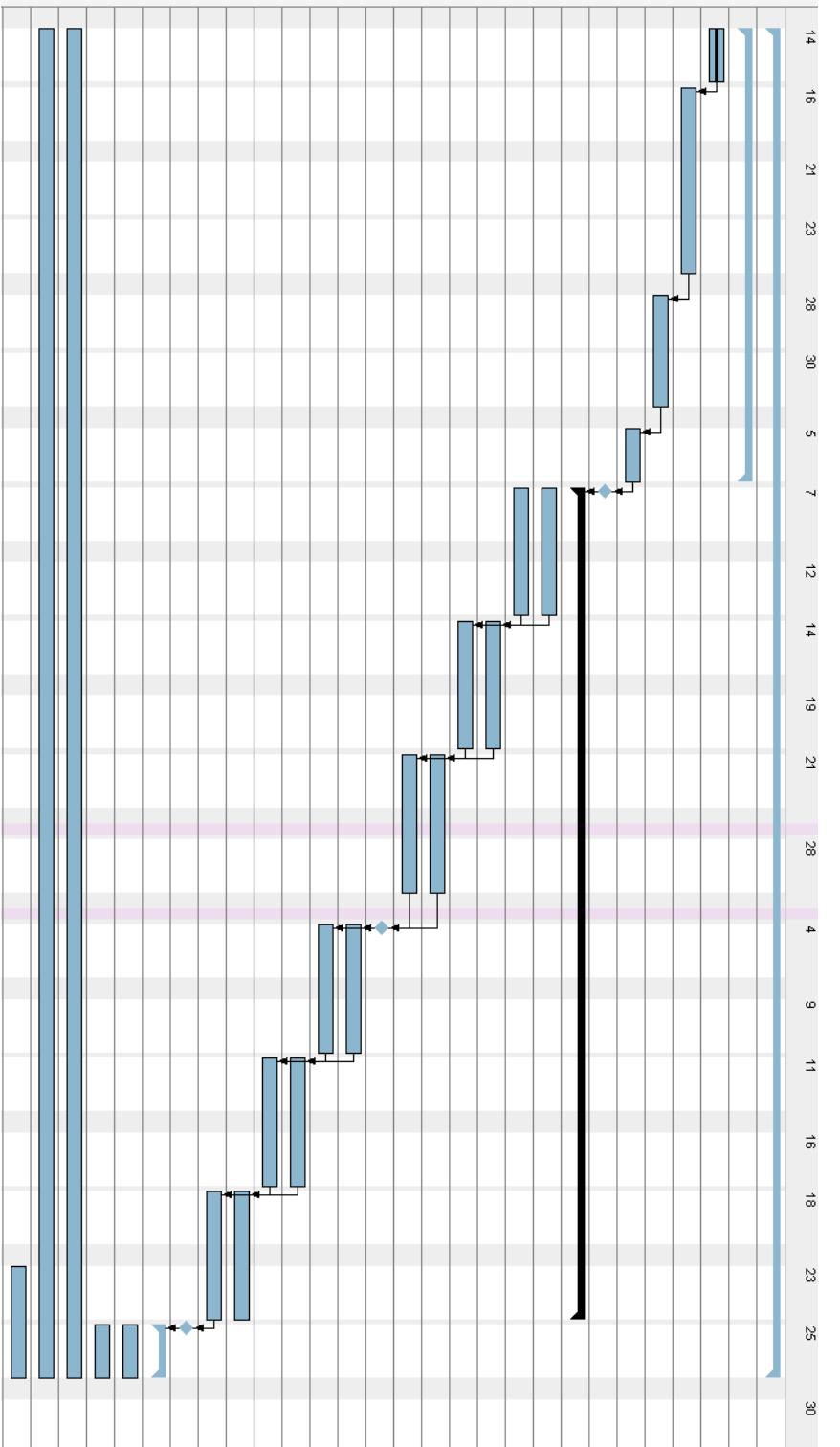
### Sitografia

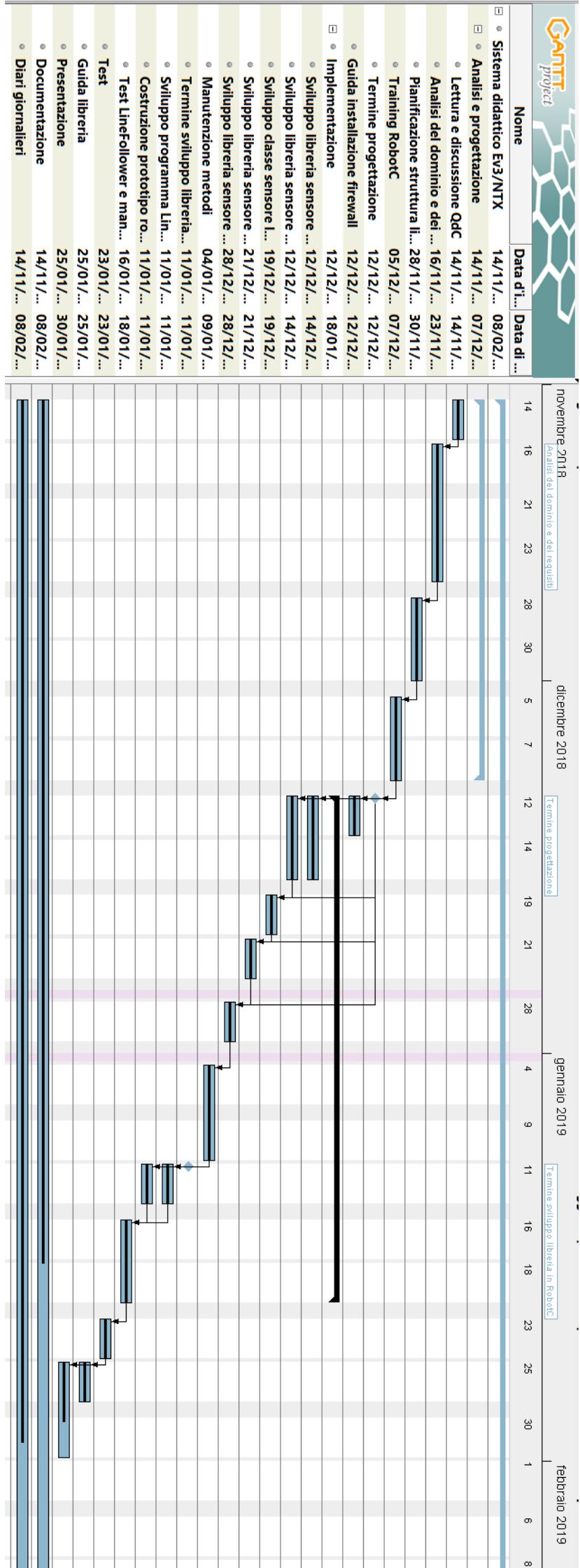
- <http://www.robotc.net/forums/viewtopic.php?f=1&t=19568>, *RobotC Forum*, 21.11.2018.
- <https://www.dexterindustries.com/howto/using-debugger-stream-in-robotc-for-lego/>, *RobotC debug stream*, 30.11.2018
- <http://www.robotc.net/forums/viewtopic.php?f=1&t=7556>, *RobotC buttons handling*, 12.12.2018.
- [http://www.robotc.net/wikiarchive/NXT\\_Functions\\_Display](http://www.robotc.net/wikiarchive/NXT_Functions_Display), *RobotC display functions Manual*, 17.12.2018.
- <http://www.robotc.net/forums/viewtopic.php?f=52&t=5916>, *RobotC pragma manual*, 19.12.2018.
- [http://help.robotc.net/WebHelpVEX/index.htm#Resources/topics/VEX\\_IQ/ROBOTC/Motor\\_Commands/setMotorTarget.htm](http://help.robotc.net/WebHelpVEX/index.htm#Resources/topics/VEX_IQ/ROBOTC/Motor_Commands/setMotorTarget.htm), *Move motor with degrees*, 09.01.2019.
- <https://gist.github.com/dhmmjoph/f63a002bde9409bbff97>, *Proportional line follower*, 16.01.2018.
- <http://ev3lessons.com/en/ProgrammingLessons/advanced/LineFollower.pdf>, *Proportional line follower*, 16.01.2018.
- <https://stackoverflow.com/questions/14675913/changing-image-size-in-markdown>, *Markdown images*, 23.01.2018.
- <https://stackoverflow.com/questions/22601053/pagebreak-in-markdown-while-creating-pdf>, *Markdown page break*, 30.01.2019.

## Allegati

- Diari di lavoro
- Istruzioni di installazione del firmware
- Istruzioni libreria wait.h
- Programma dimostrativo
- Immagine Gantt iniziale
- Immagine Gantt finale

Nome	Data d'inizio	Data di fine
• Sistema didattico EV3/NTX	14/11/18	25/01/19
• Analisi e progettazione	14/11/18	05/12/18
• Lettura e discussione QdC	14/11/18	14/11/18
• Analisi del dominio e dei requisiti	16/11/18	23/11/18
• Pianificazione struttura librerie/classi	28/11/18	30/11/18
• Training RobotC	05/12/18	05/12/18
• Termino progettazione	07/12/18	07/12/18
• Implementazione	07/12/18	23/01/19
◦ Sviluppo classe sensore tattile in Java	07/12/18	12/12/18
◦ Sviluppo classe sensore luce in Java	07/12/18	12/12/18
◦ Sviluppo classe sensore colore in Java	14/12/18	19/12/18
◦ Sviluppo classe sensore ultrasuoni in...	14/12/18	19/12/18
◦ Sviluppo classe sensore giroscopio in...	21/12/18	28/12/18
◦ Sviluppo classe motori in Java	21/12/18	28/12/18
◦ Termino sviluppo in Java	04/01/19	04/01/19
◦ Sviluppo classe sensore tattile in Ro...	04/01/19	09/01/19
◦ Sviluppo classe sensore luce in RobotC	04/01/19	09/01/19
◦ Sviluppo classe sensore colore in Ro...	11/01/19	16/01/19
◦ Sviluppo classe sensore ultrasuoni in...	11/01/19	16/01/19
◦ Sviluppo classe sensore giroscopio in...	18/01/19	23/01/19
◦ Sviluppo classe motori in RobotC	18/01/19	23/01/19
◦ Termino sviluppo in RobotC	25/01/19	25/01/19
• Test	25/01/19	25/01/19
◦ Test classi in Java	25/01/19	25/01/19
◦ Test classi in RobotC	25/01/19	25/01/19
◦ Documentazione	14/11/18	25/01/19
◦ Diari giornalieri	14/11/18	25/01/19
◦ Presentazione	23/01/19	25/01/19





# Installazione firmware su Lego Mindstorm NXT

## Sommario

RobotC .....	3
Installazione software .....	3
Installazione firmware .....	3
Risoluzione della samba mode/clicking syndrome.....	4

## RobotC

### Installazione software

Come primo passaggio per installare il firmware di RobotC bisognerà installare il software dello stesso, per fare ciò si dovrà andare sul sito ufficiale del linguaggio e scaricare la prova di 10 giorni o comprare la chiave per la versione illimitata.

### Installazione firmware

Per installare il firmware bisognerà collegare il robot NXT acceso al computer, aprire il software di RobotC, aprire il menu “Robot”, selezionare la voce “Download Firmware” e poi “Standard File (NXT\_1056.rfw).

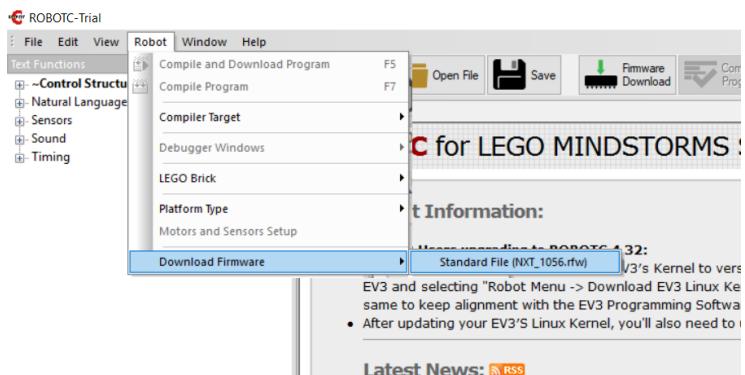


Figura 1 Selezione del firmware RobotC

Fatto ciò si aprirà una schermata in cui potremo selezionare il robot, rinominarlo se si vorrà e poi premere su “F/W Download” per scaricare il firmware

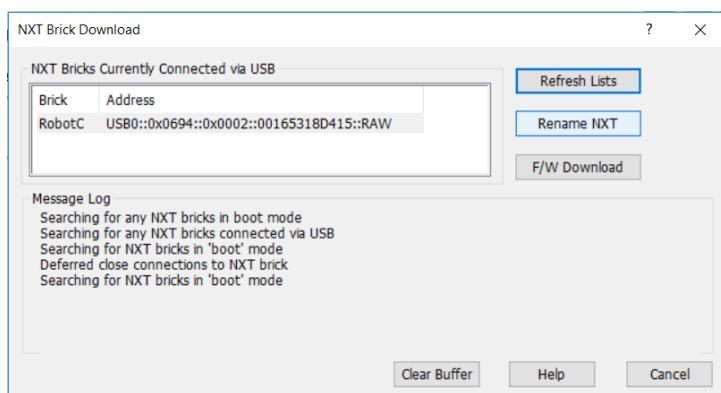


Figura 2 Schermata d'installazione del firmware

## Risoluzione della samba mode/clicking syndrome

Se durante l'installazione del firmware il robot si blocca con lo schermo nero ed emette un ticchettio costante vuol dire che è entrato nella cosiddetta "samba mode". Per risolvere questo problema bisognerà seguire la seguente procedura.

- Aprire la "Gestione dispositivi"
- Selezionare la voce "Porte (COM & LPT)"
- Scegliere la porta "Bossa Program Port"
- Premere sul menu "Driver"
- Selezionare "Aggiorna driver"
- Premere su "Cerca il software del driver nel computer"
- Selezionare "Scegli da un elenco di driver disponibili nel computer"
- Scegliere il driver "LEGO MINDSTORM NXT"
- Andare avanti fino al termine dell'installazione
- Tornare sul software di RobotC e ripetere l'operazione per installare il firmware.

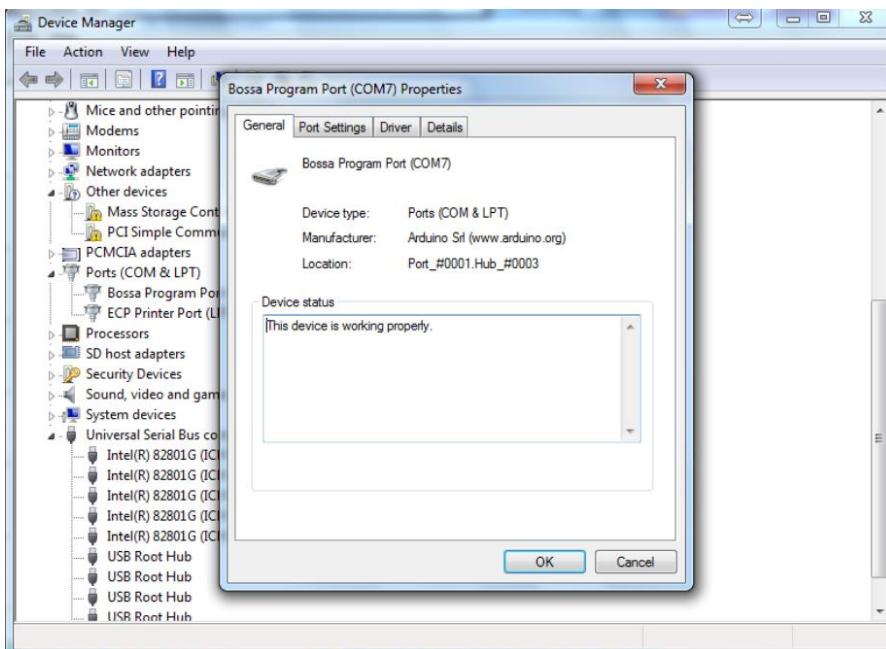


Figura 3 Bossa Program Port

# Guida all'utilizzo della libreria wait.h

## Sommario

Includere la libreria .....	3
Configurazione dei sensori .....	3
Utilizzo.....	5
Metodi.....	6
WaitTouch(int port).....	6
WaitLight(int port, int threshold, bool higher) .....	6
WaitDistance(int port, int distance) .....	7
WaitSound(int port, int threshold) .....	7
WaitTime(long millis).....	7
WaitRotations(int port, int times, int speed).....	7
WaitDegrees(int port, int degree, int speed).....	8

## Includere la libreria

Prima di tutto creare un nuovo file o aprirne uno già esistente. Successivamente bisogna la stringa di codice

```
#include "/path/wait.h" che permette di importare i metodi della libreria wait.
```

Tra i doppi apici deve essere specificato il percorso in cui si trova il file della libreria.



Figura 1 Includere la libreria wait.h

Nella figura 1 il file wait.h si trova nello stesso percorso del file appena creato quindi è abbastanza specificare il nome della libreria.

## Configurazione dei sensori

Per avere una lettura corretta dei valori da parte dei sensori è necessario configuarli. I passaggi sono semplici e veloci da effettuare:

Aprire la voce **Motors and Sensor Setup** nel menù **Robot**



Figura 2 Motors and Sensor Setup

Successivamente bisogna assegnare un nome (auto esplicativo) al sensore scelto tramite il menù a tendina sulla destra.

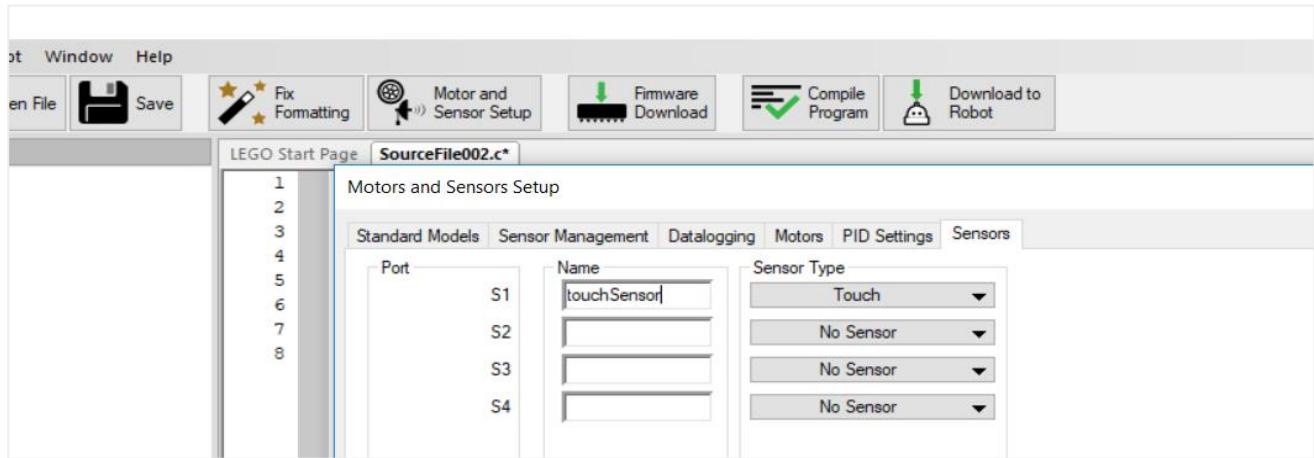


Figura 3 Configurazione sensore touch

Infine cliccare su applica e okay ed in automatico verrà generata la linea di codice contenente la configurazione.

```

1 #pragma config(Sensor, S1, touchSensor, sensorTouch)
2 //!!!Code automatically generated by 'ROBOTC' configuration wizard !!!
3
4 #include "wait.h";
5
6 task main()
7 {
8
9 }

```

Figura 4 Codice configurazione sensore touch

## Utilizzo

A questo punto non resta che utilizzare i metodi della libreria wait.

In questo esempio il motore continua a muoversi a velocità 50 fino a quando il sensore touch viene premuto.



The screenshot shows the LEGO MINDSTORMS software interface. The menu bar includes 'File', 'Window', and 'Help'. The toolbar contains icons for 'Save', 'Fix Formatting', 'Motor and Sensor Setup', 'Firmware Download', 'Compile Program', and 'Download to Robot'. The main window displays a code editor titled 'SourceFile002.c\*' with the following content:

```
1 #pragma config(Sensor, S1, touchSensor, sensorTouch)
2 /*!!Code automatically generated by 'ROBOTC' configuration wizard !*/
3
4 #include "wait.h";
5
6 task main()
7 {
8     motor[motorA] = 50;
9     waitTouch(touchSensor);
10}
```

Figura 5 Esempio di utilizzo

## Metodi

WaitTouch(int port)

### Parametri:

- **int port**: numero di tipo intero che indica la porta a cui il sensore è collegato al robot, in questo caso si possono passare i valori S1, S2, S3 e S4.

### Esempio:

```
waitTouch(S2);
```

WaitLight(int port, int threshold, bool higher)

### Parametri:

- **int port**: numero di tipo intero che indica la porta a cui il sensore è collegato al robot, in questo caso si possono passare i valori S1, S2, S3 e S4.
- **int threshold**: soglia di luce minima o massima (in base al valore di higher).
- **bool higher**: determina se il valore del sensore deve rilevare un valore superiore o inferiore alla soglia.

### Esempio:

```
waitLight(S2, 50, true);
```

### WaitDistance(int port, int distance)

Solitamente utilizzata come condizione all'interno di un ciclo.

#### Parametri:

- **int port:** numero di tipo intero che indica la porta a cui il sensore è collegato al robot, in questo caso si possono passare i valori S1, S2, S3 e S4.
- **int distance:** distanza minima da un oggetto in centimetri.

**Return:** se la distanza è inferiore alla soglia ritorna **true** altrimenti **false**.

#### Esempio:

```
waitDistance(S2, 30);
```

### WaitSound(int port, int threshold)

#### Parametri:

- **int port:** numero di tipo intero che indica la porta a cui il sensore è collegato al robot, in questo caso si possono passare i valori S1, S2, S3 e S4.
- **int threshold:** soglia del suono.

#### Esempio:

```
waitSound(S2, 70);
```

### WaitTime(long millis)

#### Parametri:

- **int millis:** tempo in millisecondi.

#### Esempio:

```
waitTime(S2, 70);
```

### WaitRotations(int port, int times, int speed)

#### Parametri:

- **int port:** numero di tipo intero che indica la porta a cui il motore è collegato al robot, in questo si possono passare i valori S1, S2, S3 e S4.
- **int times :** numero intero di rotazioni che il motore deve compiere.
- **int speed:** velocità con cui vengono svolte le rotazioni.

#### Esempio:

```
waitRotations(S2, 5, 100);
```

WaitDegrees(int port, int degree, int speed)

**Parametri:**

- **int port:** numero di tipo intero che indica la porta a cui il motore è collegato al robot, in questo si possono passare i valori S1, S2, S3 e S4.
- **int degree :** numero intero di gradi di cui deve girare il motore.
- **int speed:** velocità con cui vengono svolti i gradi di rotazione.

**Esempio:**

```
waitRotations(S2, 130, 100);
```

# Diario di lavoro

---

Luogo	SAM Trevano
Data	2018-11-14

## Lavori svolti

Oggi abbiamo ricevuto il nuovo quaderno dei compiti per il secondo progetto, ho iniziato a creare la repo su github e cercare eventuali punti non molto chiari così da poter porre le domande necessarie.

Ho poi cercato delle informazioni sul linguaggio robotC.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

## Programma di massima per la prossima giornata di lavoro

La prossima giornata vorrei discutere il QdC con il mio compagno dato che oggi era assente.

# Diario di lavoro

---

Luogo	SAM Trevano
Data	2018-11-16

**Lavori svolti**

Oggi abbiamo fatto il test del modulo.

**Problemi riscontrati e soluzioni adottate**

Non c'è stato il tempo di rispondere alle domande.

**Punto della situazione rispetto alla pianificazione****Programma di massima per la prossima giornata di lavoro**

La prossima giornata vorremmo ricevere risposta alle nostre domande sul progetto.

# Diario di lavoro

---

Luogo	SAM Trevano
Data	2018-11-21

**Lavori svolti**

Oggi abbiamo svolto le interviste in cui abbiamo posto le nostre domande. Abbiamo poi scritto i capitoli dell'analisi del dominio, l'abstract ed abbiamo iniziato l'analisi dei requisiti e il diagramma di gantt.

**Problemi riscontrati e soluzioni adottate****Punto della situazione rispetto alla pianificazione****Programma di massima per la prossima giornata di lavoro**

La prossima giornata vorremmo finire il diagramma di gantt e l'analisi dei requisiti.

# Diario di lavoro

Luogo	SAM Trevano
Data	2018-11-23

## Lavori svolti

Oggi abbiamo continuato e terminato i requisiti richiesti, che sono però ancora soggetti a modifiche. Abbiamo inserito lo scopo. Aggiunto i software che abbiamo utilizzato o siamo sicuri che utilizzeremo, in futuro verranno aggiunti i sw che useremo. Abbiamo terminato l'analisi dei mezzi. Abbiamo provato a scaricare il firmware per poter utilizzare robotC.

## Problemi riscontrati e soluzioni adottate

Il robot si è bloccato nel momento che abbiamo scaricato il firmware ed da quel momento non è più partito. In questo momento siamo bloccati e non abbiamo trovato una soluzione.

## Punto della situazione rispetto alla pianificazione

Al momento siamo rispettando la pianificazione effettuata.

## Programma di massima per la prossima giornata di lavoro

La prossima lezione dobbiamo trovare una soluzione per poter utilizzare il robot. Scaricheremo i software in modo da poter accedere all'attività di training.

# Diario di lavoro

Luogo	SAM Trevano
Data	28.11.2018

## Lavori svolti

Oggi abbiamo installato con successo il firmware di RobotC ed abbiamo scritto buona parte della documentazione che spiega come fare ciò.  
Abbiamo poi anche iniziato il diagramma delle classi.

## Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione  
Siamo in orario

## Programma di massima per la prossima giornata di lavoro

Vorremmo finire la guida all'installazione del firmware e fare dei test di funzionamento del linguaggio RobotC

# Diario di lavoro

Luogo	SAM Trevano
Data	30.11.2018

## Lavori svolti

Come prima cosa abbiamo installato il software per programmare in robotC e creato la guida per l'installazione del firmware. Abbiamo guardato un video per vedere come poter utilizzare il sensore di suono (<https://www.youtube.com/watch?v=YIjYQr4VoGA>) e completato in successione il capitolo requisiti. Abbiamo testato che i sensori che abbiamo ricevuto funzionassero. Ci siamo informati sul linguaggio Robot. In seguito abbiamo testato un po' di programmi per provare i sensori e capire il funzionamento di robotC.

## Problemi riscontrati e soluzioni adottate

## Punto della situazione rispetto alla pianificazione

Siamo in orario

## Programma di massima per la prossima giornata di lavoro

Matteo deve presentare il suo primo progetto.

Inizieremo a fare programmi utili per la libreria scritta in robotC per NXT.

# Diario di lavoro

---

Luogo	SAM Trevano
Data	05.12.2018

## Lavori svolti

Oggi abbiamo installato fatto delle prove con robotC e continuato dei programmi di prova lasciati incompiuti dalla volta scorsa.

## Problemi riscontrati e soluzioni adottate

RobotC da problemi random con i metodi di base che lui stesso importa dicendo che non esistono e non so come risolverlo.

## Punto della situazione rispetto alla pianificazione

Siamo in orario

## Programma di massima per la prossima giornata di lavoro

Vorremmo capire i problemi che da RobotC.

# Diario di lavoro

Luogo	SAM Trevano
Data	07.12.2018

## Lavori svolti

Oggi abbiamo stravolto in parte il progetto decidendo che dovremmo creare delle librerie basate sui blocchetti del software grafico di lego mindstorm ed inoltre un mini programma che simulerà un linefollower che terminerà con l'azionamento di un motore ad una determinata distanza da un oggetto.

Abbiamo inoltre ricevuto le chiavi per robotC:

Id: 63767870

Pass: 43583RQ3

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

Siamo in orario

## Programma di massima per la prossima giornata di lavoro

Dovremmo iniziare a progettare le parti del nuovo progetto.

# Diario di lavoro

Luogo	SAM Trevano
Data	12.12.2018

## Lavori svolti

Oggi abbiamo creato la libreria per il sensore tattile che contiene un metodo per impostare la porta ed uno per controllare se uno dei due sensori è premuto.  
Volevamo iniziare a lavorare con il robot Ev3 che avremmo dovuto ricevere ma non lo abbiamo ancora e quindi abbiamo fatto tutto per NXT.  
Abbiamo inoltre apportato delle modifiche al gantt consuntivo.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

Siamo in orario

## Programma di massima per la prossima giornata di lavoro

Dovremmo installare il firmware su Ev3 e modificare le classi per farlo funzionare su di esso.

# Diario di lavoro

Luogo	SAM Trevano
Data	14.12.2018

## Lavori svolti

Oggi abbiamo ridiscusso rapidamente il progetto ed abbiamo quindi eliminato le librerie create le scorse volte e ne abbiamo in compenso creata un'altra chiamata wait. Essa contiene i metodi che attendono fino alla lettura di un determinato dato da parte del sensore. Abbiamo anche creato la classe di test della libreria. Non abbiamo ricevuto il robot Ev3 quindi continueremo il progetto su NXT.

## Problemi riscontrati e soluzioni adottate

-

## Punto della situazione rispetto alla pianificazione

Siamo in orario

## Programma di massima per la prossima giornata di lavoro

Dovremmo finire la libreria e testarla.

# Diario di lavoro

---

Luogo	SAM Trevano
Data	19.12.2018

**Lavori svolti**

Oggi abbiamo terminato la prima versione della libreria wait completando i metodi riguardante il motore.

**Problemi riscontrati e soluzioni adottate**

-

**Punto della situazione rispetto alla pianificazione**

Siamo in orario

**Programma di massima per la prossima giornata di lavoro**

La prossima lezione ascolteremo le presentazioni dei compagni e mangeremo il panettone.

# Diario di lavoro

Luogo	SAM Trevano
Data	09.01.2019

## Lavori svolti

Oggi abbiamo riscontrato un po' di problemi con i metodi che abbiamo scritto le scorse lezioni.

## Problemi riscontrati e soluzioni adottate

Alcuni dei metodi creati e testati le scorsi lezioni hanno smesso di funzionare in modo corretto. Il metodo waitTouch legge il valore del sensore di tocco diverso da quello che veniva letto prima. Avevamo impostato a 500 il valore soglia per stabilire se il sensore di tocco fosse premuto o meno. Oggi dopo i dovuti test abbiamo notato che il valore del sensore da premuto è 100, e 0 se non è premuto. Abbiamo quindi dovuto cambiare lo statement in cui viene controllato se il bottone è premuto o meno.

```
//controllo se il sensore ♦ stato premuto
if(SensorValue[port] > 0){
    isPressed = !isPressed;
}
```

Questo è lo statement modificato.

## Punto della situazione rispetto alla pianificazione

Siamo in leggero anticipo rispetto al Gannt preventivo.

## Programma di massima per la prossima giornata di lavoro

Continuazione manutenzione dei metodi e test di quest'ultimi.

# Diario di lavoro

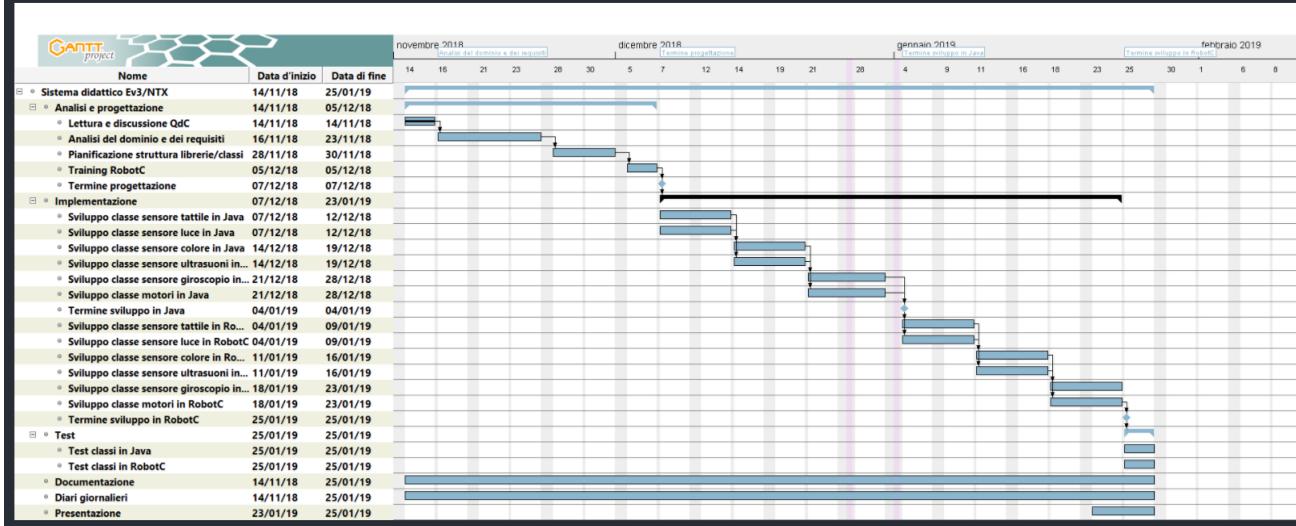
Luogo	SAM Trevano
Data	11.01.2019

## Lavori svolti

Oggi io e Matteo ci siamo divisi i compiti, lui si è occupato di modificare la documentazione, nello specifico la parte di pianificazione, in cui ha aggiunto l'immagine del Gantt preventivo e una descrizione.

### Pianificazione

Durante i colloqui con i docenti l'idea di base del progetto è cambiata. Con i vari gruppi ci siamo suddivisi i compiti così che noi dobbiamo fare solamente una libreria scritta in RobotC. Questa libreria dovrà contenere i metodi di attesa del blocco del linguaggio visivo di NXT. Dovrà inoltre venire fatto un programma di esempio per dimostrare l'utilizzo dei metodi così come il loro funzionamento. Qui di seguito si può vedere il Gantt iniziale pensato prima di sapere della modifica del progetto, nell'immagine si nota come era stato pensato di suddividere i metodi fra i due componenti del gruppo e come questa parte fosse stata pensata come la più lunga del progetto. In questa progettazione non era presente nemmeno il programma dimostrativo.



Ha rimosso 'java' dal capitolo software. Ha aggiunto due requisiti, il requisito sulla libreria 'Wait' e il programma di dimostrazione 'Line Follower'.

<b>ID</b>	<b>REQ-09</b>
<b>Nome</b>	Libreria wait
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Libreria che riporta i metodi del blocchetto wait del linguaggio visivo di NXT.
<b>Sotto requisiti</b>	
<b>001</b>	Metodo di attesa del suono.
<b>002</b>	Metodo di attesa della distanza rilevata con l'ultrasuoni.
<b>003</b>	Metodo di attesa del tocco.
<b>004</b>	Metodo di attesa della luce.
<b>005</b>	Metodo di attesa di un lasso di tempo.
<b>006</b>	Metodo di attesa del movimento in rotazioni dei motori.
<b>007</b>	Metodo di attesa del movimento in gradi dei motori.

<b>ID</b>	<b>REQ-10</b>
<b>Nome</b>	Progetto di dimostrazione e test
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Mini progetto che dimostra l'uso della libreria e ne testa il funzionamento.
<b>Sotto requisiti</b>	
<b>001</b>	Line follower percentuale.
<b>002</b>	Controllo della presenza di oggetti davanti a se.

Io invece ho controllato che i metodi che ho creato la scorsa lezione funzionassero in modo corretto e l'esito è stato positivo. Una volta fatto ciò abbiamo creato una bozza del programma ‘LineFollower’ proporzionale. Per effettuar e il calcolo della velocità dei motori mi ho visitato la seguente pagina:

<http://ev3lessons.com/en/ProgrammingLessons/advanced/LineFollower.pdf>.

```

#pragma config(Sensor, S1,      distance,      sensorSONAR)
#pragma config(Sensor, S2,      lightA,        sensorLightActive)
#pragma config(Sensor, S3,      lightB,        sensorLightActive)
//***Code automatically generated by 'ROBOTC' configuration wizard ****

#include "wait.h";

/**
 * Line follower proporzionale (con margine di errore) utilizzando i metodi della libreria wait.
 *
 * @author Bryan Beffa
 * @author Matteo Forni
 * @version 11.01.2019
 */
task main()
{
    while(waitDistance(distance, 20)) {
        //margine consentito
        double margin = 0.7;

        //soglia di luce riflessa
        int threshold = 50;

        //calcolo l'errore
        int errorA = threshold - SensorValue(lightA);
        int errorB = threshold - SensorValue(lightB);

        //calcolo la velocità
        double speedA = errorA * margin;
        double speedB = errorB * margin;

        //setto la velocità al motore collegato alla porta A e alla porta B
        motor[motorA] = speedA;
        motor[motorB] = speedB;
        waitTime(100);
    }
}

```

A questo punto abbiamo deciso di creare un prototipo di robot, il quale useremo per testare successivamente il programma. Il professor Peretti ci ha permesso di utilizzare i pezzi e dello spazio di lavoro nella sua aula. Abbiamo passato le seconde due ore (dalle 15:00 alle 16:30 ca.) a cercare i pezzi di lego necessari e costruire il robot. Infine ho aggiornato il Gantt consuntivo, aggiungendo l'attività di costruzione del robot, di sviluppo ‘LineFollower’.

### Problemi riscontrati e soluzioni adottate

Oggi non abbiamo riscontrato problemi di nessun tipo.

### Punto della situazione rispetto alla pianificazione

Punto della situazione rispetto alla pianificazione Rispetto alla pianificazione ci troviamo in orario.

### Programma di massima per la prossima giornata di lavoro

Programma di massima per la prossima giornata di lavoro

Test e manutenzione del programma LineFollower.

# Diario di lavoro

Luogo	SAM Trevano
Data	16.01.2019

## Lavori svolti

Oggi abbiamo provato a far funzionare il LineFollower proporzionale. Verso fine lezione siamo riusciti a far funzionare il LineFollower.

## Problemi riscontrati e soluzioni adottate

Abbiamo avuto più problemi, prima di tutto abbiamo perso 5 minuti a capire perché non venivano letti i valori dei sensori in modo corretto, dopodiché abbiamo notato che i sensori erano collegati in modo errato. Successivamente abbiamo dovuto smontare il robot per cambiare le batterie, abbiamo perso altri 5 minuti. La bozza del LineFollower che abbiamo creato la scorsa lezione si è rivelata inefficace e non svolgeva il suo compito. Prima di tutto abbiamo capito che dovevamo poter selezionare e salvare le soglie del colore bianco e nero. Il problema era che davamo delle velocità troppo piccole e negative.

## Punto della situazione rispetto alla pianificazione

Siamo in orario

## Programma di massima per la prossima giornata di lavoro

Migliorie del programma LineFollower

# Diario di lavoro

Luogo	SAM Trevano
Data	18.01.2019

## Lavori svolti

Oggi abbiamo continuato la documentazione e più precisamente i capitoli della progettazione e del design dell'architettura del sistema e abbiamo completato il diagramma delle classi. Abbiamo inoltre migliorato il codice del programma che esegue il line follower aggiungendo dei metodi che rendono il codice più pulito. Abbiamo infine ricostruito la classe di test della libreria così da renderla di facile utilizzo senza necessitare di scrivere altro codice.

Uno dei tre metodi aggiunti nella classe LineFollower.c, esso serve ad impostare il valore del bianco:

```
int setWhite()
{
    int white;
    while(true)
    {
        //stampo sul display
        nxtDisplayCenteredTextLine(1, "Premi la freccia");
        nxtDisplayCenteredTextLine(2, "destra per");
        nxtDisplayCenteredTextLine(3, "selezionare");
        nxtDisplayCenteredTextLine(4, "il colore bianco");

        //premere il tasto destro per selezionare il colore bianco
        if (nNxtButtonPressed == 1)
        {
            white = (SensorValue[lightA]+SensorValue[lightB])/2;
            writeDebugStreamLine("bianco: %d", white);
            return white;
        }
    }
}
```

## Problemi riscontrati e soluzioni adottate

Oggi non abbiamo riscontrato problemi.

## Punto della situazione rispetto alla pianificazione

Siamo in orario

## Programma di massima per la prossima giornata di lavoro

La prossima lezione vorremmo finire la parte di implementazione della documentazione e ricontrillare la parte di progettazione.

# Diario di lavoro

Luogo	SAM Trevano
Data	23.01.2019

## Lavori svolti

Oggi abbiamo, come da programma, scritto i capitoli dell'implementazione ed abbiamo scritto due test case, uno per la libreria ed uno per il line follower. Abbiamo inoltre rimosso un while dal metodo waitDistance che non serviva a nulla.

## Problemi riscontrati e soluzioni adottate

Oggi non abbiamo riscontrato problemi.

## Punto della situazione rispetto alla pianificazione

Siamo in orario

## Programma di massima per la prossima giornata di lavoro

La prossima lezione vorremmo finire la documentazione ed iniziare la presentazione. Vogliamo inoltre iniziare e finire una piccola guida all'utilizzo della libreria e dei suoi metodi.

# Diario di lavoro

Luogo	SAM Trevano
Data	25.01.2019

## Lavori svolti

Oggi abbiamo scritto la guida per l'utilizzo della libreria, abbiamo iniziato la presentazione del progetto, abbiamo modificato la guida per l'installazione del firmware aggiungendo il titolo e l'indicizzazione, abbiamo modificato il gantt consuntivo dato che la consegna è stata spostata e siamo andati avanti con la scrittura dell'implementazione del progetto del line follower.

## Problemi riscontrati e soluzioni adottate

Oggi non abbiamo riscontrato problemi.

## Punto della situazione rispetto alla pianificazione

Siamo in orario

## Programma di massima per la prossima giornata di lavoro

La prossima lezione vorremmo finire la documentazione ed iniziare la presentazione.

# Diario di lavoro

---

Luogo	SAM Trevano
Data	30.01.2019

**Lavori svolti**

Oggi abbiamo modificato la documentazione aggiungendo la tabella di resoconto dei test, il giustificato al testo e le interruzioni di pagina. Abbiamo inoltre controllato la correttezza grammaticale della documentazione scritta fino ad ora.

**Problemi riscontrati e soluzioni adottate**

Oggi non abbiamo riscontrato problemi.

**Punto della situazione rispetto alla pianificazione**

Siamo in orario

**Programma di massima per la prossima giornata di lavoro**

La prossima lezione vorremmo finire la documentazione.

# Diario di lavoro

Luogo	SAM Trevano
Data	01.02.2019

## Lavori svolti

Oggi abbiamo, io a scuola e Bryan da casa perché era malato, finito la documentazione in markdown. Abbiamo appunto controllato tutto il documento in cerca di errori ortografici e dove ne abbiamo trovati li abbiamo corretti, abbiamo inoltre completato i capitoli conclusivi della documentazione.

In secondo luogo abbiamo modificato la presentazione correggendo alcune imprecisioni.

## Problemi riscontrati e soluzioni adottate

Gli unici problemi riscontrati sono quelli con la nostra padronanza della lingua italiana.

## Punto della situazione rispetto alla pianificazione

Siamo in leggero anticipo.

## Programma di massima per la prossima giornata di lavoro

La prossima lezione vorremmo convertire la doc da md a pdf con un template di pandoc che ci darà Muggiasca.

# Diario di lavoro

Luogo	SAM Trevano
Data	06.02.2019

## Lavori svolti

Oggi abbiamo svolto una parte di revisione, Matteo ha aggiustato l'indice della documentazione tramite markdown. Bryan ha revisionato le slide della presentazione ed aggiunto le note e i commenti su alcune slide.  
Abbiamo inoltre dato una rilettura della documentazione e degli altri allegati (installazione firewall, guida di utilizzo della libreria).

## Problemi riscontrati e soluzioni adottate

Questa lezione non abbiamo riscontrato problemi.

## Punto della situazione rispetto alla pianificazione

Siamo in orario.

## Programma di massima per la prossima giornata di lavoro

Revisionare la documentazione e la presentazione, a fine lezione dovremo consegnare il progetto.

# Diario di lavoro

Luogo	SAM Trevano
Data	08.02.2019

## Lavori svolti

Abbiamo completato la revisione iniziata l'ultima lezione, abbiamo aggiunto i metodi nella guida della libreria e preparato tutto il materiale per la consegna del progetto. Abbiamo creato un unico documento che comprende, documentazione, guida della libreria, guida per l'installazione del firmware, diari e diagrammi di Gantt. Abbiamo inoltre ricontrollato il programma Line Follower ed il suo funzionamento a livello pratico.

## Problemi riscontrati e soluzioni adottate

Non abbiamo riscontrato alcun problema.

## Punto della situazione rispetto alla pianificazione

Abbiamo terminato il progetto, come pianificato in precedenza.

## Programma di massima per la prossima giornata di lavoro

-