-
-
-

-
-
-
-
-

-
-

!

- " #

$ %

-
- &
- ' (

) * +

, *

- +
- *

-

.

- +/' 0 1 2 1
- /" ' 30 ' 3 1 3'
- ' % + 3 3
- ! 4 - 54 4 4 -

> 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.

" 6 3 & * 7 8 + 9 : ; <=%3 +
> &? :
/

-
- (
- @
-
-

+ A

" + 3 + A : + <=%
> &? " + A B & ' 3 + A
& * 8 + + A + + 6 C " + A
+3 + A
3 3 + "

" : @?
' " +
(D + & *
E+ + F 5 2G
+ +
(C
D + +
> &?

	* 8 +
!	
"	4
	6

	#
	* 8 +
!	
"	4
	6

	$
	* 8 +
!	
"	4
	6

	%
	* 8 +
!	

"	4
	6

	&
	* 8 +
!	
"	4
	6

	'
	* 8 +
!	
"	4
	6

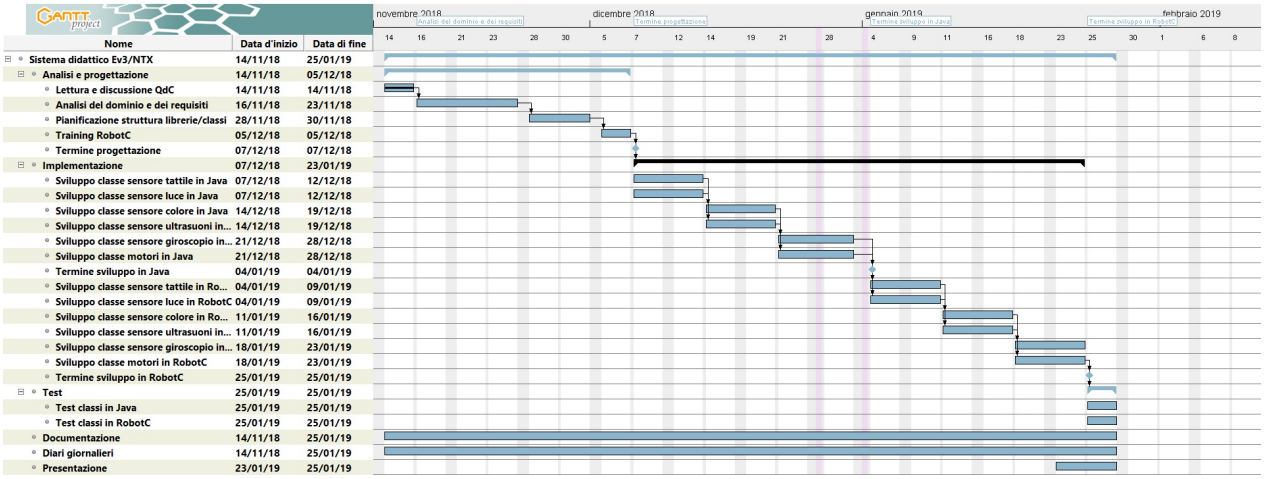
	(
	* 8 +
!	
"	4
	6

	)
	* 8 +
!	
"	4
	6

	*
	" #
!	
"	4
	" # + + <=%
	'
#	' + 7
$	'
%	'
&	'
'	' +
(	' +

!	
"	4
	' 7
	" #
#	* +

D H+ @ 3 7 +
C +



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
Termine 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
Termine 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
Termine 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

+

/

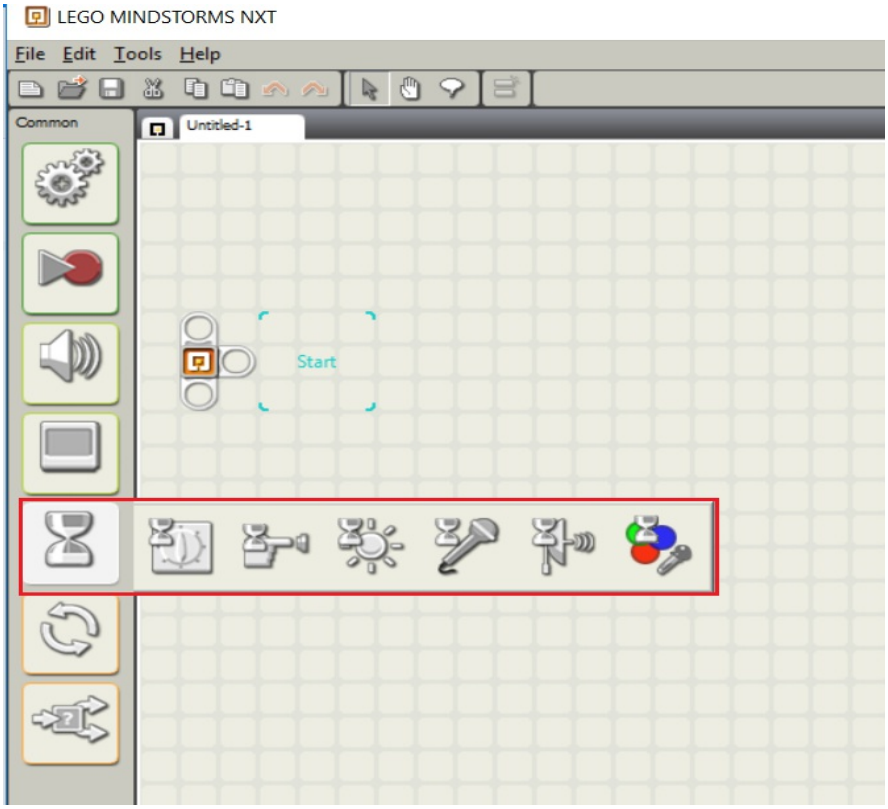
- ": @? ' <=%
- & * :

, +

6 # /

- F G > # 4
- & " <=% 9
-
-
-
- '
- '

7 6 * + + B
& * D + A + + <=% + A +
7 B
" + A # + + <=%
+

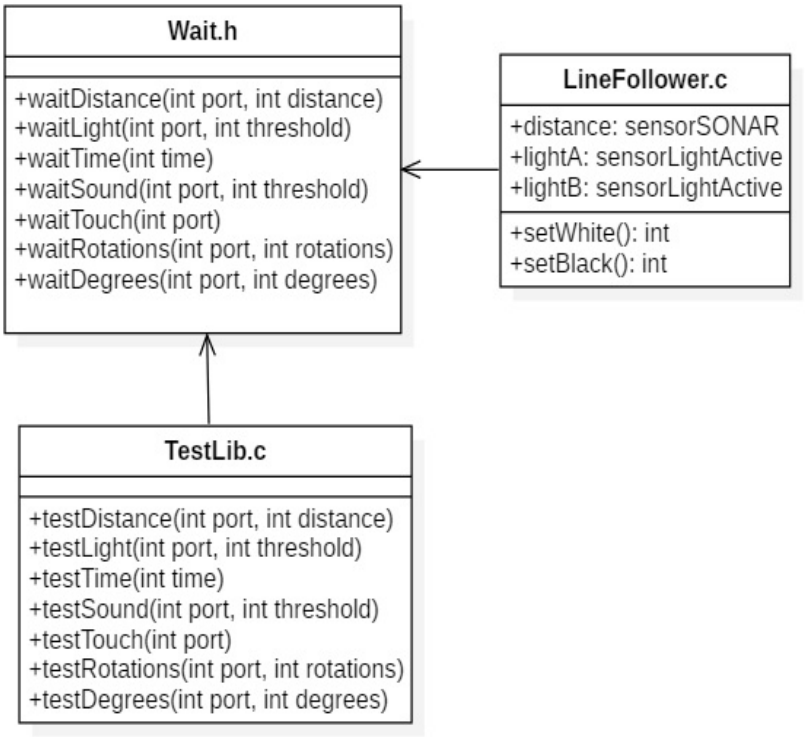


? H + + + / 3

* + A+ & * B +
+
A # + A 3 7
3 A + + +

- .

D H+ 3 + / # 3
:



/ +

" # 3 & *3 7 ": @? & <=%

- # % F G
- # " F 3 3 G
- # F 3 G
- # F 3 G
- # % F G
- # & F 3 3 G
- # F 3 3 G

+ 0 . 1 2

7 # % F G 6 D +

6 ; + 6
D + + I4I + + + true while

```
/**
 * 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;
        }
    }
}
```

: /

<=% +

+ / . 1 3 . . 3 . . 2

" F 3 3 G
+ +
threshold higher 6 + true
threshold higher
false

D 3 + flag 3
3+ false B

```
/**
 * 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;
            }
        }
    }
}
```

: /

+ +

+ 1 3 2

F 3 G + + 3
F G3 +
distance

D + +

> F 3 G3 + 3 +
7 B +
6

```
/**
 * 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;
}
```

: /

+ + 7
3 3

+ 1 3 . . 2

D 7 # FG

+

+ threshold

```
/**
 * Metodo che attende che il sensore di suono rilevi un suono che supero 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;
        }
    }
}
```

: /

+ 3 + +

+ 0 1 2

% 6 3
millis

6 6 3

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

: /

+ 1 3 3 2

& F 3 3 G
3 3
+ A

```
/**
 * Metodo che attende che il motore passato come parametro svolga il numero di rotazioni desiderato
 *
 * @param port porta a cui è collegato il motore
 * @paaram 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);
}
```

: /

+ + $

+ 1 3 3 2

D 3 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);
}
```

: /

 + . 4J

/ 4 +

 #
 #

 " 0 # 7 /

-
-
- F G

 + 3 +
 3
 + 3
 7 3

*
 > FG 1 9FG

```
//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();
```

 7
 + +

```
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;
    }
  }
}
```

 + + #

```
 while(waitDistance(distance, 20)){
    //margine 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);
  }
```

* + + margin
# C + 6 C
+ + A

; 3
3 + 7

+ A+ errorA
+ errorB margin

+ + A
+ 7 4
#

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

7 # + 6 # F+
waitDistance(int port, int threshold) G +
6

6)4
3 + A 5 43

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

motor[motorB] = 0;
motor[motorA] = 0;

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

0

+

A ? +

0 5	05
	% #
	&: D54.
	1 #
	1 + # 3 * (

	1 A % " 2 3 > 31 A
(3 + A	
	* + +

0 5	05 #
	% #
	&: D5 4
	1 " 0 #
	+ #
	1 A 7 + 0 H A 3 A
	+

% +
+ 3
+ A F
A G3 + A
+

6 7

+ +
3 < +

5 8

* + + +
F , GF @
+ G

5

D K * + AK
* A K L K L
6 + 6
K 3
K

8

' +

5

* K

9

9 8

* F G 3
3

% F + G3

% + F G3

!

$ 3

9

* F G 3
3

% F G3

+ < 3

! < 3

$ 3

) 1<

- /((# # # ((+ # K M N M . $) - 3RobotC Forum 3 4 -
- /((# # # O (# (5 5 5 5 5 5 (3RobotC debug stream 3 4 4 -
- /((# # # ((+ # K M N M, $$) 3RobotC buttons handling 3 4 -
- /((# # # (# 9 + (<=%P0 P 23RobotC display functions Manual 3 , 4 -
- /((# # # ((+ # K M$ N M$.) 3RobotC pragma manual 3 . 4 -
- /(((E ( ) 44 . ! 4. . , 3Proportional line follower 3) 4 4 -
- /((+ ((" (+ (" 0 # 3Proportional line follower 3) 4 4 -
- /((9 + # ((!) , $. (5 5 5 5 9 # 3Markdown images 3 4 4 -

: 3 /

- +
- #
- " #
- +