

SAMT - Sezione Informatica

Lego Lib - Diario

Pagina: 1 / 2

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:

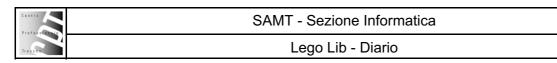
```
System.out.println("inizio");
long s = System.currentTimeMillis();
sleep(5000);
if (System.currentTimeMillis() > s) System.out.println("prova");
System.out.println("finito");
```

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:

Pagina: 2 / 2

- 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, defininendo 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.