“MyRSD” Project

ITI Francesco Severi – Padova

(Technical-industrial institute)

“My Robot Sensor Development” is an extracurricular project which dealt with the complex theme of the robotics and automation, through a series of didactic experiences. It has been a cooperation between didactics and industrial fields, purposing to create a hardware and software platform for the “research and educational” area, which allows to elaborate advanced automation controls in a simple and instinctive way.

The said targets have been possible thanks to both the components and the development environment chosen: the first were the Lego™ Mindstorms mechatronic components, which offered high availability and low price; while, to program the embedded platform “NI MyRIO”, it has been used NI LabVIEW, the National Instruments’ graphic environment. This combination offers an instinctive, powerful and reliable way to do didactic automation.

Moreover, the Lego™ Mindstorms products offered large range of hardware components, like sensors, actuators and structural mechanic pieces, at a low price. Therefore it permitted to quickly assemble and overhaul different prototypes of robots or, generally, automation systems, thus eliminating the problems resulting from expensive and complex mechanical machining processes.

The project was launched during the 2013-14 AY by the technical institute ITI “F. Severi”, homed in Padua, and supported by I.R.S. S.r.l. Padova and National Instruments Italy Srl. It was presented for the first time during the “NIDays” in Milan on 12th March 2015.

Thanks to the project, the school won a funding of 20 000€ by the scholastic education and research ministry (MIUR). The amount has entirely been investedinto equipment for the automation and the mechatronics laboratories, thus enabling an ever-increasing number of students to take part in the project in future.

During the past academic year (2014-15 AY), the project has seen the active participation of five enterprising and deserving students, chosen from the last year (5th) and among three different courses of the secondary school: Giacomo Bernabei and Francesco Vidaich from Automation; Matteo Bernabei from Electronics and Luca Girardi and Biasio Federico from “Mechanics and Mechatronics”. Hence, special merit must be acknowledged to the students for having worked validly and with enthusiasm during extracurricular hours.

Special thanks are also due to all the people who made MyRSD project possible:

thanks to Eng. Massimo Rapini, from I.R.S. S.r.l. Padova (National Instruments partner), who had the intuition from where the project was born and who constantly cooperated with the project during the year;

in addition, it must be remembered that also National Instruments Italy S.r.l. provided the project with technical support, especially thanks toEng. Simone Suaria and his staff;

finally, it’s dutifully mentioned the work done by the people who worked at the project from inside the institute, together with the students: Prof. Alessandro Scroccaro, also technical director of the project, and the teachers Ugo Riso and Giorgio Tombola.

Progetto didattico “My Robot Sensor Development - myRSD”: "l’intelligenza NI entra nel mondo educational"

Introduzione

Il progetto "myRSD", nato da una collaborazione tra mondo della didattica e mondo dell'industria, si propone di offrire al settore "research & educational" una piattaforma hardware e software con cui sperimentare, in maniera semplice ed immediata, soluzioni di controllo avanzate per l'automazione e la robotica. Tali propositi sono stati ad oggi raggiunti finalmente con un prodotto commerciale, da un lato, grazie all'adozione di componentistica meccatronica a basso costo e di facile reperibilità, come quella offerta dai prodotti LEGO Mindstorm, e dall'altro, grazie all'impiego di un linguaggio di programmazione grafico, potente ed immediato, come NI LabVIEW di National Instruments, impiegato su di una piattaforma embedded myRIO, che è, come vedremo nel seguito, molto più di un comune PC embedded.

In particolare, myRSD offre un hardware d'interfacciamento tra myRIO e dispositivi Mindstorm, con cui sarà possibile realizzare velocemente prototipi, senza la necessità di ricorrere a saldatore e stagno, permettendo così di apportare modifiche "al volo", in qualunque ambiente (laboratorio o aula) ed in qualunque istituto (professionale, tecnico e liceo). Inoltre, le librerie offerte da myRSD permetteranno a tutti di sperimentare la potenza, la flessibilità e l'immediatezza di NI LabVIEW, applicati al mondo della robotica e dell'automazione, dando così "una marcia in più" a tutte quelle applicazioni oggi controllate da semplici microcontrollori, oppure dai più sofisticati, quanto complessi da gestire, PC embedded.

La scelta dei prodotti LEGO Mindstorm, in abbinamento all'architettura myRIO, è stata una scelta naturale, volendo offrire un hardware ricco di componenti come sensori, attuatori e particolari meccanici, facilmente reperibile ed a basso costo, che offrisse la possibilità di realizzare prototipi di robot e modelli di automazioni in genere, in tempi rapidi e senza problemi derivanti da lavorazioni meccaniche costose ed impegnative.

Il progetto è stato avviato dall’istituto ITIS “F. Severi” di Padova nel a. s. 2013/2014, grazie al supporto di IRS Srl di Padova e di National Instrumens Italy Srl, ed è stato presentato per la prima volta in occasione dell'NIDays di Milano il 12 Marzo 2015.

L'Istituto con questo progetto ha vinto il bando del MIUR “Istituzioni Scolastiche - Legge 113/91 D.D. 2216/Ric/01-07-2014 T2” ottenendo un finanziamento per un totale di 20.000 €, completamente investiti in attrezzature per il laboratorio didattico di automazione e meccatronica, così da poter aprire questa iniziativa ad un numero sempre maggiore di allievi.

Quest'attività ha visto protagonisti alcuni intraprendenti e meritevoli allievi delle classi V dell’Istituto F. Severi (a. s. 2014/2015), a cui va riconosciuto il merito di aver validamente lavorato con convinzione ed entusiasmo a questo progetto in orario extracurricolare. Un particolare riconoscimento va quindi agli allievi: Federico Biasio, Giacomo e Matteo Bernabei, Luca Girardi, Francesco Vidaich, degli indirizzi di elettronica-automazione e di meccanica-meccatronica. Un ulteriore ringraziamento va poi ai partner aziendali, senza i quali nulla di questo sarebbe stato possibile. Doveroso è inoltre ricordare come tutto sia nato da una felice intuizione dell'ing. Rapini di IRS Srl di Padova, a cui va la nostra gratitudine per aver offerto questa possibilità di collaborazione proprio all'ITIS F. Severi di Padova. Oltre alla collaborazione di IRS, il progetto è stato tecnicamente supportato da NI, pertanto un particolare ringraziamento va all’ing. Simone Suaria ed allo staff di Milano di NI Italia. In fine, concludiamo i ringraziamenti citando coloro che hanno lavorato all'interno dell'Istituto, assieme agli allievi: il prof. Alessandro Scroccaro (referente tecnico del progetto) ed i professori Ugo Riso e Giorgio Tombola.

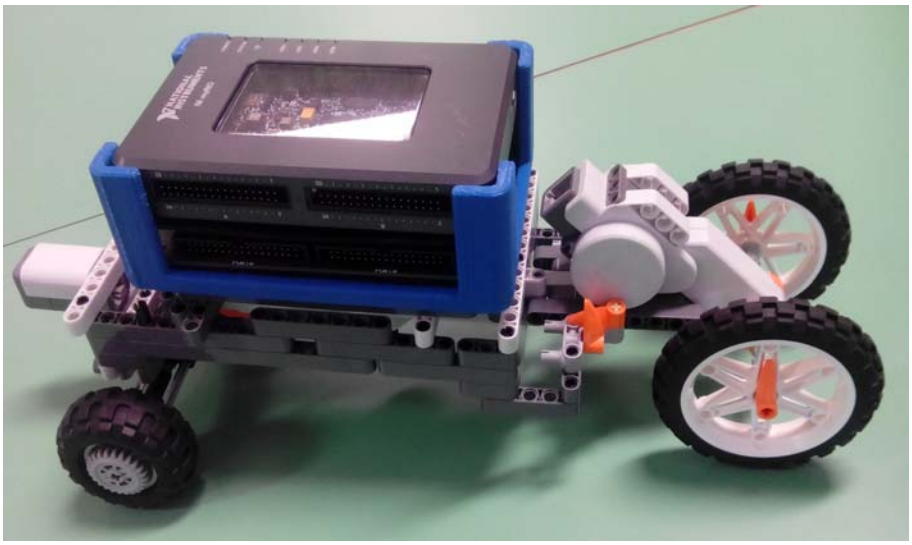
Descrizione di NI myRIO

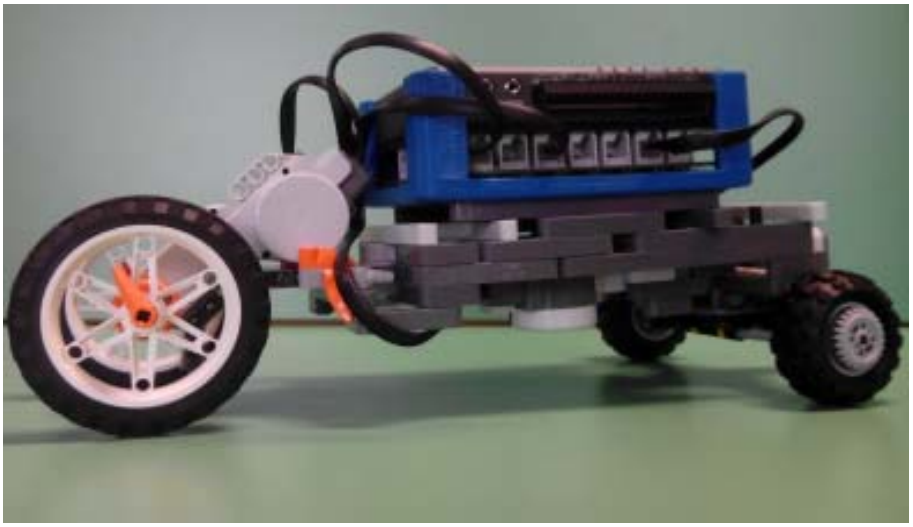
Vediamo ora, in dettaglio, le principali specifiche della scheda myRIO, ma prima diciamo subito che non si tratta del solito PC Embedded su cui gira un Linux, come siamo soliti vedere, ma bensì di un vero e proprio sistema di misura e controllo basato su un SO Real Time (Linux RT) che impiega la tecnologia "system on chip" SoC Zinq 7000 di Xilinx: costituita da un microprocessore "dual core" di tipo ARM Cortex A9 ed una potente FPGA a 40 MHz che si interfaccia con un front-end di tipo analogico e digitale. La presenza della FPGA ha messo a disposizione del progettista la potenza della riconfigurabilità delle risorse realizzate attraverso la FPGA stessa, permettendo così di superare i limiti talvolta imposti dalle architetture tradizionali di tipo non riconfigurabile. Si tratta di un prodotto che deriva dall'esperienza NI sui sistemi industriali di questo tipo, specificatamente progettato per il mondo educational, dal costo contenuto (in particolare per gli studenti), ma dalla prestazioni di tutto rispetto, come il front-end analogico e digitale di acquisizione e controllo dati con 10 Analog Input e 6 Analog Output, le 40 linee Digital Input / Output, suddivise tra la porta Mini System Port (MSP) per applicazioni didattiche e le due porte myRIO Expansion Port (MXP), permettendo così di sfruttare tutte le potenzialità del dispositivo. L'ADC è a 12 bit con una frequenza di campionamento 500kS/s, mentre la CPU dispone di 512MB di RAM per il SO ed i programmi utente in LabView, oltre naturalmente ad una memoria SSD di caricamento e dati. La disponibilità del "Wireless Link" e della connettività USB, LED e pulsanti integrati, un accelerometro tri-assiale, il tutto con un consumo che va dai 2.6W ai 14W, fanno del myRIO un prodotto veramente completo ed adatto a tutte le applicazioni. In conclusione possiamo realizzare, grazie all'ambiente di sviluppo NI LabVIEW, sia software eseguito in ambiente "real time" e residente sulla SSD integrata, che algoritmi eseguiti in hardware sulla FPGA; garantendo quindi il determinismo della nostra applicazione. Non vanno dimenticati poi i numerosi moduli e toolkit aggiuntivi all'ambiente NI LabVIEW, tra cui la Mathscript RT module, NI Vision, Control Design & Simulation e molti altri.

Esperienze sulla meccanica

La progettazione dei veicoli sperimentati, basati su LEGO Mindstorm, a livello strutturale ha richiesto non poche valutazioni e le modifiche al progetto in corso d'opera sono state molte. Tuttavia la flessibilità dei componenti lego ha reso possibile giungere alla struttura ottimale in breve tempo; inoltre il posizionamento dei trasduttori LEGO è stata semplice avendo realizzato le strutture interamente con blocchetti della medesima casa produttrice.

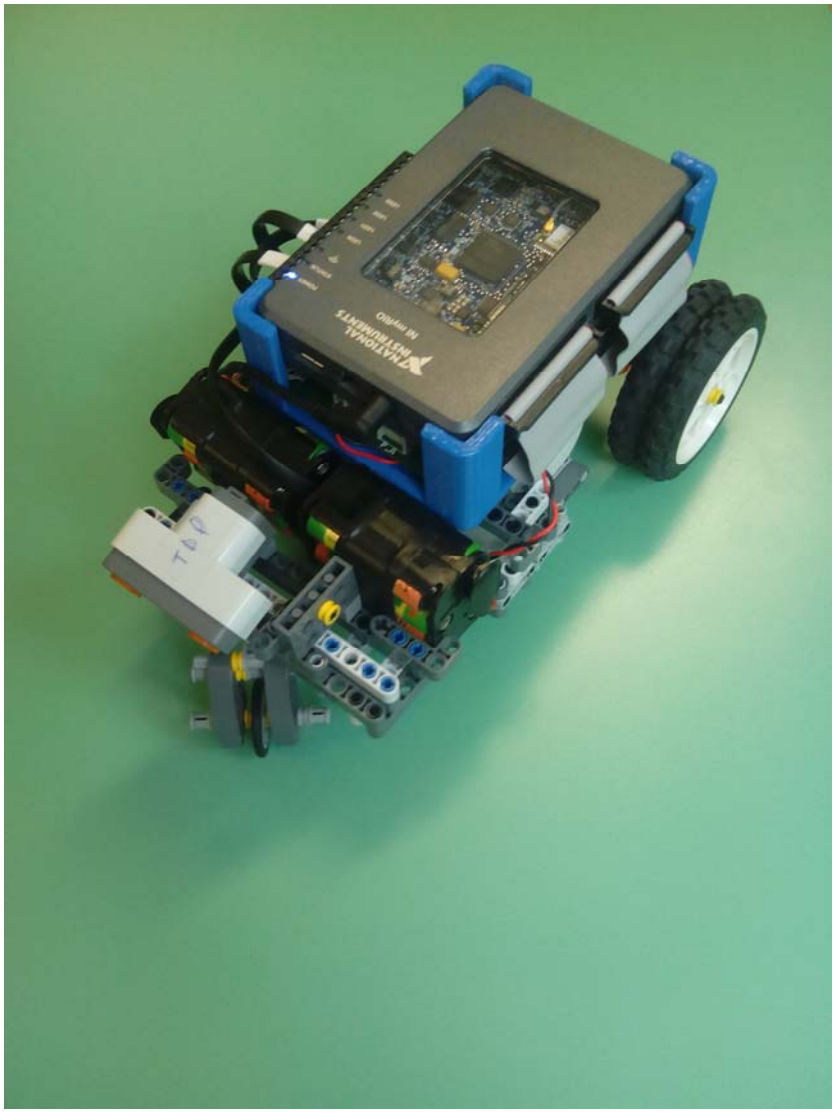
Nel caso del primo modello, visibile nelle immagini qui sotto riportate, dotato di un asse sterzante e uno di trazione indipendenti, la sfida maggiore è stata contenere la lunghezza del veicolo in rapporto alla larghezza e all'altezza dello stesso. Infatti, le considerevoli dimensioni e il peso complessivo di board myRIO, scheda di interfaccia e batterie aumentavano significativamente il rollio in curva. Il problema è stato risolto mediante il ribassamento dello chassis, l'impiego di assali più larghi e il posizionamento del motore di trazione in diagonale anziché orizzontale. La precisione della sterzata risentiva dell'assale maggiorato ed ancora la non trascurabile lunghezza di interasse rendeva il raggio di sterzata piuttosto elevato.

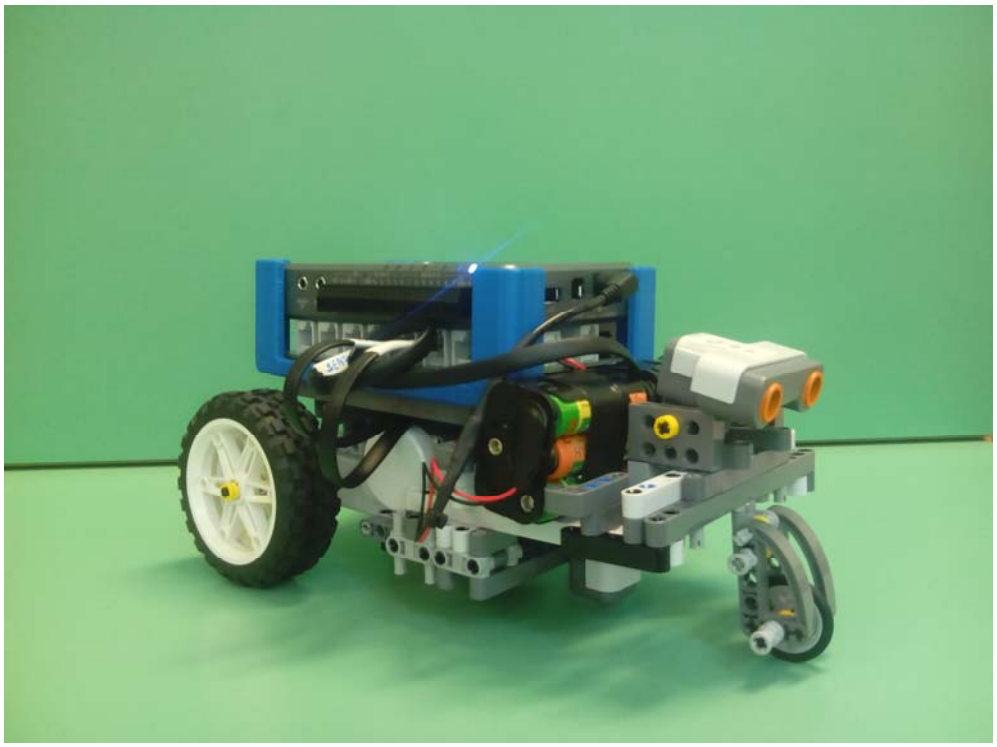


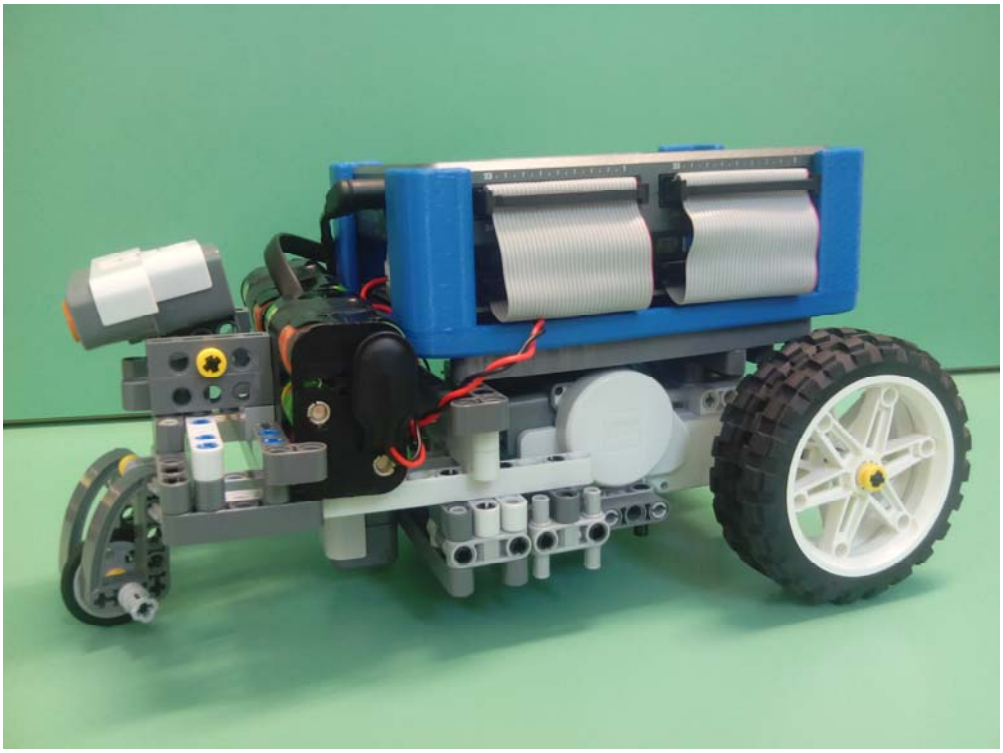


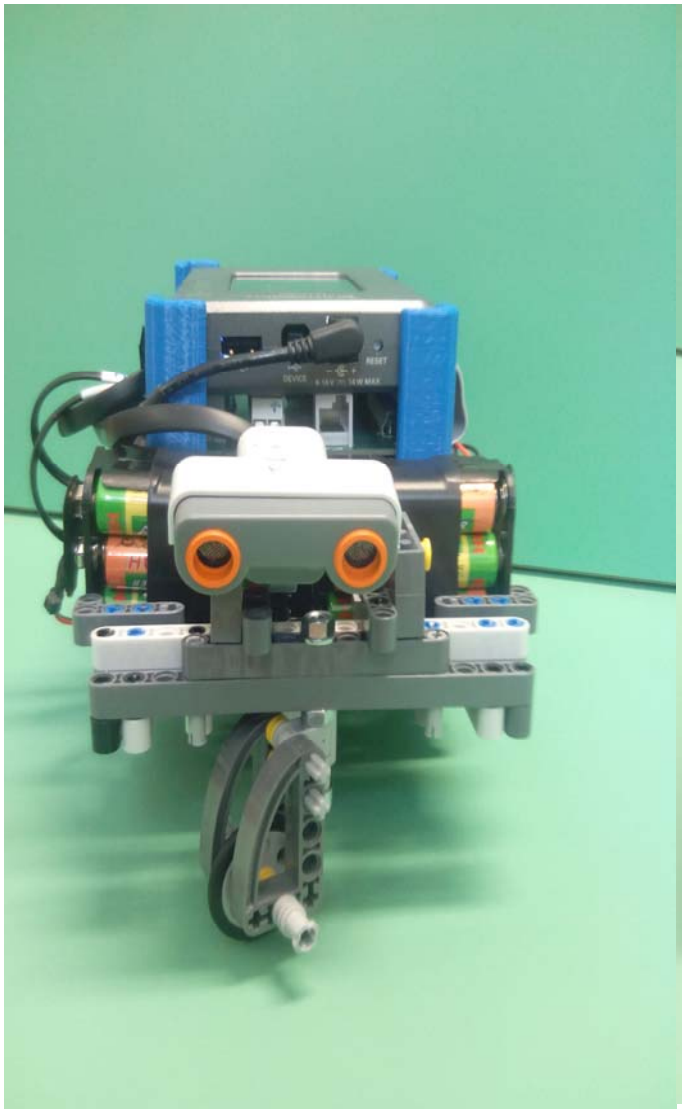
Si è quindi deciso di revisionare la struttura da capo, modificandone i principi base di funzionamento: ne è nato il secondo modello myRSD, visibile nelle immagini qui sotto riportate. In esso abbiamo accostato due motori per la trazione, ognuno dei quali trasmetteva potenza ad un semiasse. In tal modo siamo riusciti ad ottenere un veicolo in cui l'avanzamento è realizzato mediante l'azionamento concorde e sincrono di entrambi i motori, mentre la sterzata dal rallentamento o, più incisivamente, dall'inversione del motore interno alla curva. Il tutto è reso possibile dall'appoggio sulla ruota pivotante anteriore. In questo prototipo inoltre, la distanza assiale tra le ruote è risultata notevolmente maggiore, senza che la resistenza dei semiassi a flessione ne risultasse compromessa, permettendo di posizionare l'unità di controllo e alimentazione direttamente sopra l'asse ed i motori. Unico punto debole del robot di Fig. 2 è stata la ruota pivotante: inizialmente si era utilizzata una ruota con pneumatico classica dei veicoli telecomandati lego; in seguito, nel tentativo di ridurne al minimo l'attrito nel movimento di rotazione rispetto all'asse del perno, si è optato per la più sottile ruota gommata messa a disposizione dalla casa LEGO. L'intenzione di ridurre l'attrito ha avuto buon esito, ma ha trascurato un eccessiva flessione cui è sottoposto lo snello assieme ruota-perno. Tale inconveniente è stato comunque risolto aumentando il numero di staffe LEGO che componevano il meccanismo, irrobustendolo.

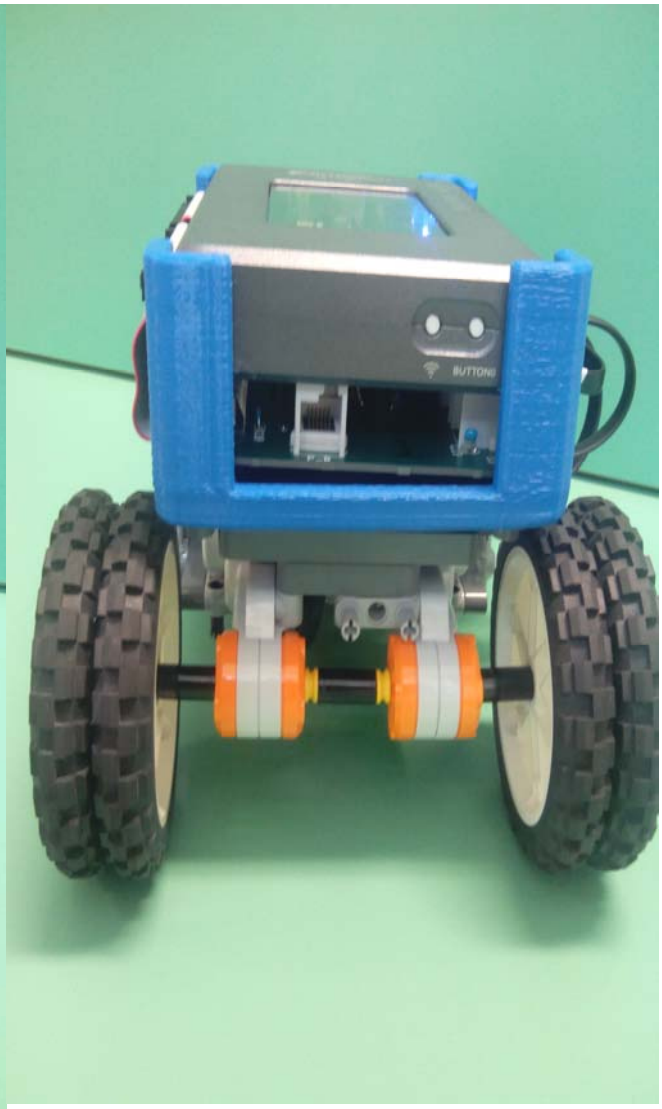
In conclusione ne è risultato un veicolo che gode della capacità di sterzare sul posto e che è decisamente più compatto e stabile del modello precedente.








RESET
DEVICE


BUTTON0

HARDWARE di INTERFACCIAMENTO

La scheda d'interfaccia è così costituita:

Nove connettori presa "NXT-style RJ12", compatibili con quelli presenti nei sensori utilizzati (LEGO).

Questi connettori, diversamente dai comuni RJ12 presentano la linguetta di fermo a sinistra anziché in posizione centrale come tipicamente avviene per i connettori RJxx.

Semplici da inserire e disinserire e molto resistenti, i connettori "NXT-style RJ12", sono facilmente reperibili in rete oppure possono essere creati artigianalmente modificando i normali RJ12.

Due connettori orizzontali maschi del tipo a vaschetta da 34 poli, per poter collegare il PCB alla myRIO, posizionati in modo da rendere diretto e facilmente estraibile il collegamento tramite piattina.

Un connettore per l'alimentazione dei driver (9V), dotato di due morsetti con vite, semplici da svitare e avvitare con un cacciavite a taglio.

Due doppi ponti ad H per il controllo ON/OFF di motori, del tipo LM298 della STMicroelectronics, nella versione montaggio superficiale (SMD).

Ogni Driver è in grado di comandare 2 motori brushed, oppure un motore stepper a due fasi, azionando naturalmente i motori sia in senso orario (FORWARD) che in senso antiorario (REVERSE).

La scelta dei driver L298 è dettata dalla necessità di rendere più semplice e veloce la connessione di un qualunque altro attuatore.

Qui di seguito sono riportate in breve le principali caratteristiche dell'integrato L298.

Symbol	Parameter	Value	Unit
VS	Power Supply	50	V
VSS	Logic Supply Voltage	7	V
VI,Ven	Input and Enable Voltage	-0.3 to 7	V
IO	Peak Output Current (each Channel) – Non Repetitive (t = 100□s) –Repetitive (80% on –20% off; ton = 10ms) –DC Operation	 3 2.5 2	 A A A
Vsens	Sensing Voltage	–1 to 2.3	V
Ptot	Total Power Dissipation (Tcase = 75°C)	25	W
Top	Junction Operating Temperature	–25 to 130	°C
Tstg, Tj	Storage and Junction Temperature	–40 to 150	°C

Per quanto riguarda i SENSORI sull'interfaccia myRSD possono essere connessi ben 5 sensori:

Due pulsanti (Button_1 e Button_2), un sensore analogico (Sound Senso / light sensor / rgb) due sensori con comunicazione seriale I2C (Ultrasonic Sensor / Compass / Gyro / Accelerometer).

SCHEMA ELETTRICO

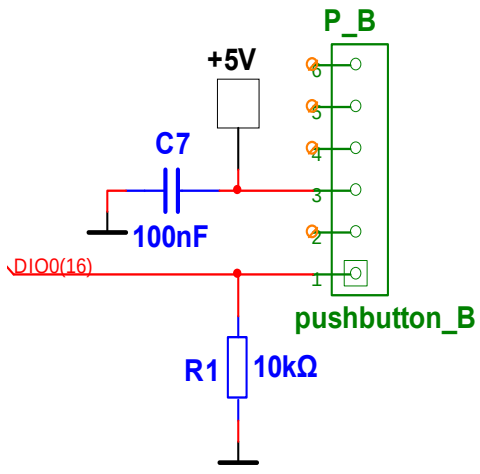
Nel PCB sono presenti numerosi condensatori ceramici da 10nF, impiegati come by-pass per le correnti a frequenza elevata, collegati tra l'alimentazione di 5 V e la massa e situati in prossimità dei circuiti integrati.

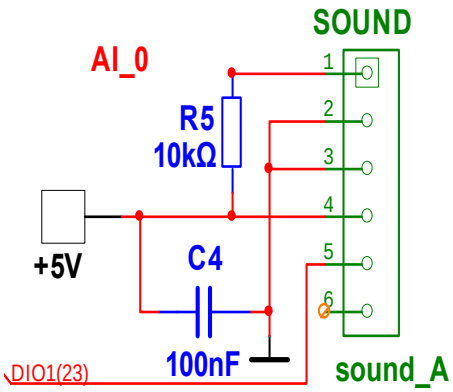
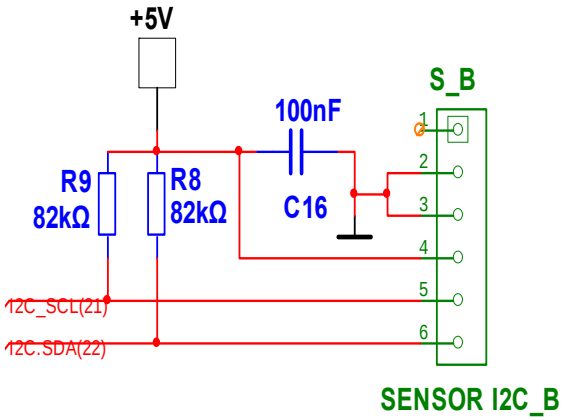
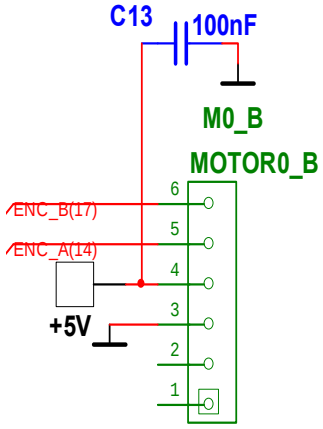
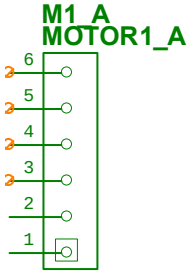
Il circuito integra anche quattro resistori di pull-up da 82kΩ, collegati ai due connettori che utilizzano la trasmissione dati seriale I2C, utilizzati per mantenere a livello logico alto i terminali di SCL e SDA quando sono inutilizzati. Mentre altri sei resistori di pull-down da 10kΩ sono impiegati per mantenere a livello basso (0V) gli ingressi dei sensori digitali (push-button) normalmente bassi; e per mantenere a livello basso i pin di enable dei driver motori, quando non sono presenti segnali di comando.

Nel circuito sono presenti due distinte alimentazioni: quella da 5 V, prelevata dal myRIO tramite alcuni pin dai due connettori a 34 poli, necessaria per alimentare il controllo dei driver ed i sensori e quella da 9 V che deve essere fornita tramite batteria o alimentatore esterno, utilizzata per alimentare i motori e particolari sensori, come quello a ultrasuoni, che per funzionare hanno bisogno di 2 tensioni.

FUNZIONAMENTO DEL CIRCUITO

Differentemente dal BRICK della LEGO, nel quale ad ogni connettore può essere collegato a qualsiasi sensore, nell'interfaccia myRSD, per semplicità si è deciso di configurare ogni connettore per un unico tipo di sensore (ad es. segnale analogico, digitale, I2C, ...).

	Nei connettori P_B e P_A possono essere collegati i sensori digitali (push button); In questi connettori sono utilizzati solamente 2 terminali : uno per l' alimentazione di 5 V e uno collegato all' ingresso digitale 0 del myRIO.

	Il connettore SOUND è in grado di gestire sensori analogici (light e sound); In questo connettore vengono utilizzati 5 terminali: 3 per l' alimentazione, uno per AI_0, il canale analogico utilizzato per acquisire il valore analogico in uscita dal sensore,e uno per DIO0, il canale digitale in uscita, che serve per abilitare per esempio l'accensione del led nel light sensor.
	I connettori S_A e S_B possono essere utilizzati per gestire i sensori che utilizzano il protocollo di trasmissione I2C (bussola, ultrasuoni, giroscopio); In questi connettori sono utilizzati 5 pin: 3 per l' alimentazione, uno per il segnale di clock (SCL) e uno per la trasmissione dei dati (SDA).
	I connettori M0_A e M0_B sono in grado di gestire degli attuatori come i motori DC LEGO e la lettura dell' encoder; In questi connettori vengono utilizzati tutti e 6 i pin: 2 per l' alimentazione, 2 per il collegamento con i driver motori , per la gestione del movimento e 2 per il collegamento con la myRIO per aggiornare l' encoder.
	I connettori M1_A e M1_B riescono a gestire degli attuatori come i motori DC LEGO; In questi connettori sono utilizzati solamente 2 terminali connessi ai driver motori, per la gestione del movimento. In questi connettori non può essere effettuata la lettura dell'encoder.

PCB: Grazie al piano di massa, tutte le piazzole collegate a massa sono collegate tra loro, quindi sono allo stesso potenziale elettrico, evitando la stampa di piste che avrebbero prodotto induttanze parassite che avrebbero potuto indurre dei mal funzionamenti.

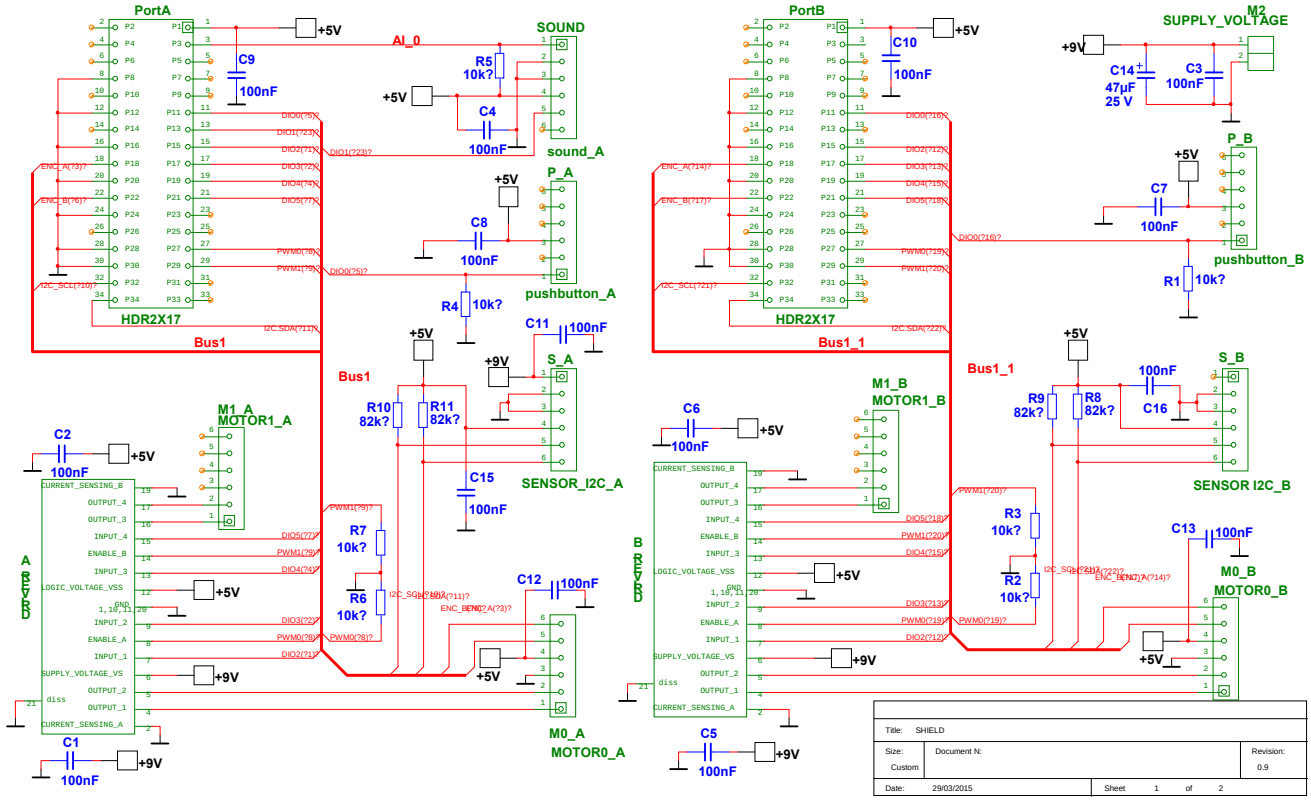
Inoltre, essendo i driver motori progettati per montaggio superficiale, sono dotati di un'aletta per la dissipazione del calore in silicio, che ricopre la maggior parte della superficie inferiore dell'integrato. Il PCB è stato progettato in modo da permettere una facile saldatura dei driver motori, predisponendo delle zone di rame non ricoperte dal solder e dotate di via per meglio disperdere il calore sulla faccia sottostante.

In fine, il PCB è dotato di serigrafie che permettono l'immediato riconoscimento dei nomi dei connettori e delle loro funzioni e di fori per fissaggio dello stesso a delle eventuali colonnine di supporto.

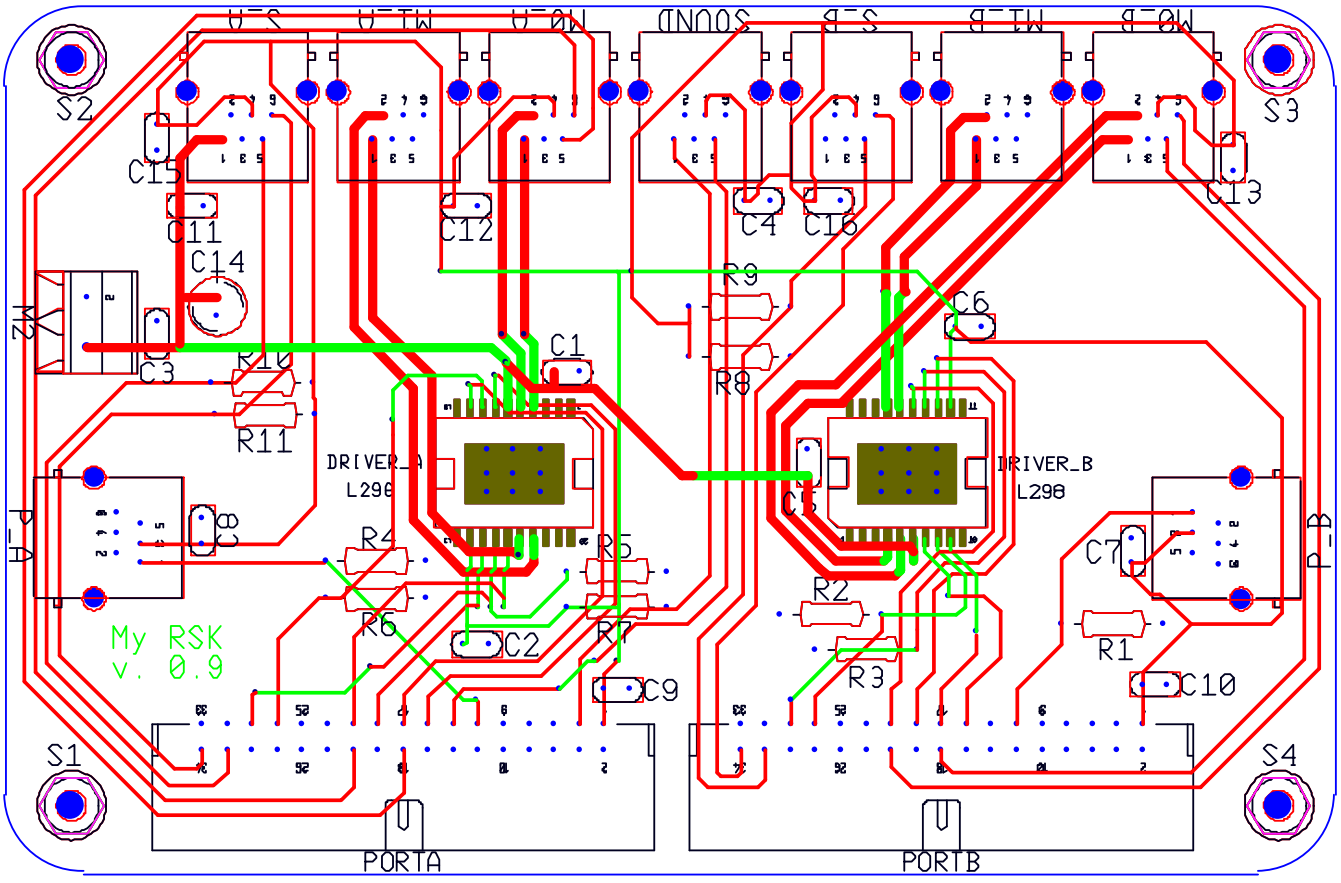
MIGLIORAMENTI

Nelle prossime versioni dello stampato, saranno inseriti alcuni miglioramenti: come un led per segnalare la presenza delle alimentazioni ed il corretto funzionamento del myRSD, alcuni test point per prelevare alcune tensioni utili in fase di collaudo, serigrafie più facilmente interpretabili ed ulteriori connettori per gestire altri sensori.

Qui di seguito sono riportati lo schema elettrico ed il PCB dell'interfaccia myRSD.



Schema elettrico



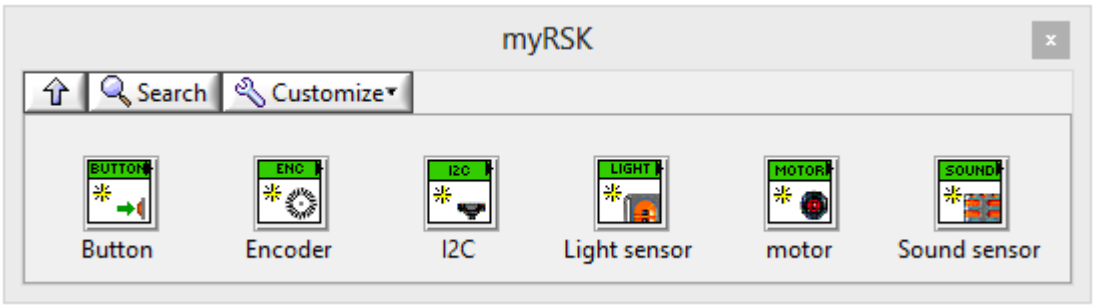
PCB vista superiore.

LIBRERIE MYRSD

Per l'interfacciamento software dell'ecosistema LEGO Mindstorm con il PC Embedded myRIO di NI sono state create delle apposite librerie composte da un VI polimorfico per ciascun tipo di sensori e attuatori LEGO.

In base al funzionamento dei vari sensori ed attuatori, sono stati utilizzati canali analogici, digitali e seriali configurati come ingressi ed uscite.

La realizzazione di programmi con l'utilizzo della libreria è immediata e molto simile a quella utilizzata nella filosofia di LEGO, grazie all'utilizzo dei VI polimorfici, in grado da svolgere tutte le diverse funzioni necessarie per l'utilizzo dei sensori ed attuatori.



Questa è la palette myRSD situata nel blocco funzioni.

Si possono distinguere 5 tipi di sensori e un attuatore:

- digitali: Button, Encoder

- seriali I2C: Compass

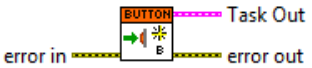
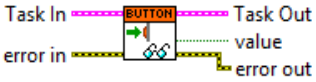
- analogici: Light sensor, Sound sensor

- attuatore: DC motor

N.B. L' utilizzo dell'encoder è legato esclusivamente ai connettori M_0 A e B. Nei connettori M_1 A e B non è stata implementata la lettura dell’encoder.

Segue una breve descrizione dei VI polimorfici.

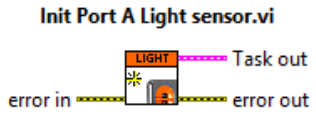
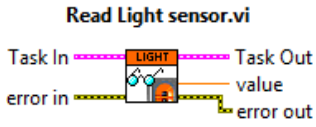
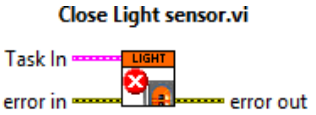
PUSH_BUTTON POLYMORPHIC VI

Init Port A Button.vi  Il VI crea l'istanza del pulsante.	Il pulsante è collegato al pin digitale **DIO0** del **portA** (pin 11) del myRIO. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.
Init Port B Button.vi  Il VI crea l'istanza del pulsante.	Il pulsante è collegato al pin digitale **DIO0** del **portB** (pin 11) del myRIO. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.
Read Button.vi  Il VI legge lo stato del pulsante.	**Task In:** riceve in ingresso l'istanza precedentemente creata. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma. **value:** corrisponde al valore dello stato del pulsante.
Released.vi  Il VI aspetta il rilascio pulsante .	**Task In:** riceve in ingresso l'istanza precedentemente creata. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma. **value:** corrisponde al valore dello stato del pulsante.
Stop Button.vi  Il VI chiude l'istanza del pulsante.	**Task In:** riceve in ingresso l'istanza precedentemente creata. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.

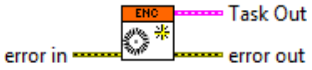
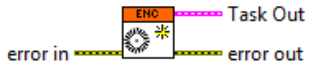
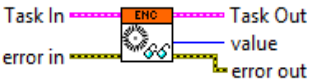
SOUND SENSOR POLYMORPHIC VI

Init Port A Sound Sensor.vi  Il VI crea l'istanza del Sound sensor.	Il Sound sensor è collegato al pin analogico**AI0** del **portA** (pin 3) del myRIO. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.
Read Sound Sensor.vi  Il VI legge il valore in uscita dal Sound sensor.	**Task In:** riceve in ingresso l'istanza precedentemente creata. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma. **value:** corrisponde al valore in uscita dal Sound sensor.
Stop Sound Sensor.vi  Il VI chiude l'istanza del Sound sensor.	**Task In:** riceve in ingresso l'istanza precedentemente creata. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.

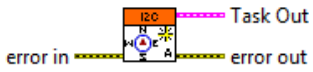
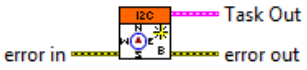
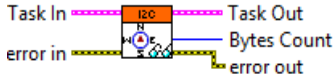
LIGHT SENSOR POLYMORPHIC VI

 Il VI crea l'istanza del Light sensor	Il Light sensor è collegato al pin analogico **AI0** del **portA** (pin 3) del myRIO e al pin digitale **DIO1** del **portA** (pin 12) del myRIO, per azionare l' accensione del led, per il funzionamento del sensore in modo attivo. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.
 Il VI legge il valore in uscita dal Light sensor.	**Task In:** riceve in ingresso l'istanza precedentemente creata. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma. **value:** corrisponde al valore in uscita dal Light sensor (luminosità).
 Il VI chiude l'istanza del Light sensor.	**Task In:** riceve in ingresso l'istanza precedentemente creata. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.

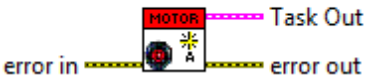
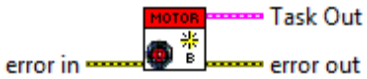
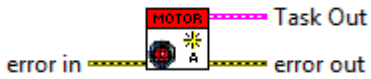
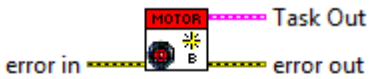
ENCODER POLYMORPHIC VI

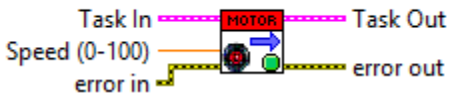
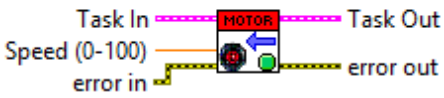
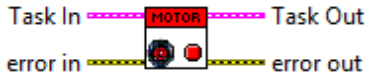
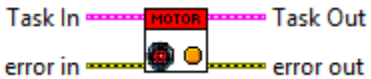
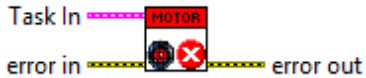
Init Port A Encoder.vi  Il VI crea l'istanza dell' Encoder.	L'Encoder è collegato ai pin digitali **ENCA** e **ENCB** del **portA** pin (18 e 22) del myRIO. La lettura dell' encoder può essere fatta solo con l'utilizzo del connettore M_0 (Motore 0). **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.
Init Port B Encoder.vi  Il VI crea l'istanza dell' Encoder.	L'Encoder è collegato ai pin digitali **ENCA** e **ENCB** del **portB** pin (18 e 22) del myRIO. La lettura dell' encoder può essere fatta solo con l'utilizzo del connettore M_0 (Motore 0). **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.
Read Encoder.vi  Il VI legge il valore in uscita dall' Encoder.	**Task In:** riceve in ingresso l'istanza precedentemente creata. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma. **value:** corrisponde al valore in uscita dall' Encoder.
Stop Encoder.vi  Il VI chiude l'istanza dell' Encoder.	**Task In:** riceve in ingresso l'istanza precedentemente creata. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.

I2C POLYMORPHIC VI

Init I2C Port A.vi  Il VI crea l'istanza del Compass sensor	L'Encoder è collegato ai pin digitali **SCL** e **SDA** del **portA** pin (33 e 34) del myRIO. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.
Init I2C Port B.vi  Il VI crea l'istanza del Compass sensor	L'Encoder è collegato ai pin digitali **SCL** e **SDA** del **portB**pin (33 e 34) del myRIO. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.
Read I2C.vi  Il VI legge il valore in uscita dal Compass sensor.	**Task In:** riceve in ingresso l'istanza precedentemente creata. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma. **value:** corrisponde al valore in uscita dal Compass sensor.
Close I2C.vi  Il VI chiude l'istanza del Compass sensor.	**Task In:** riceve in ingresso l'istanza precedentemente creata. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.

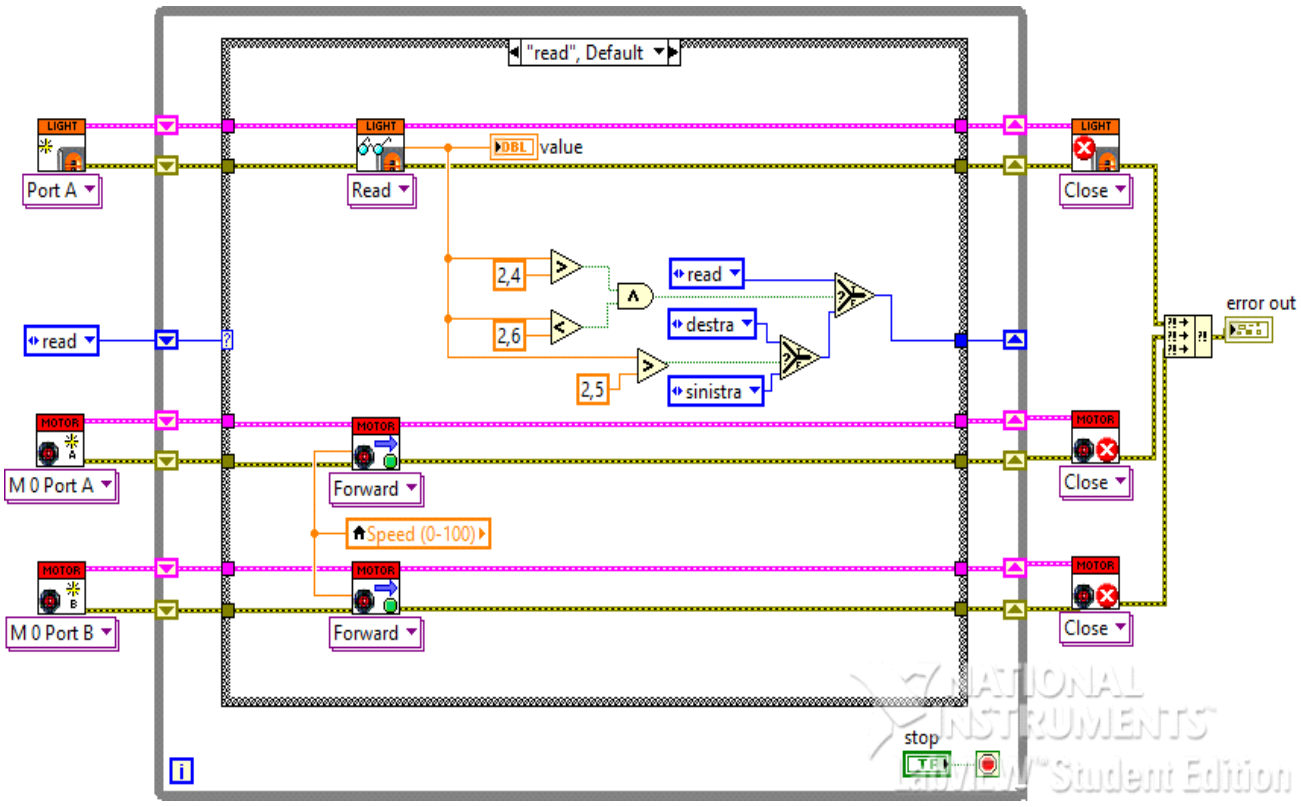
MOTOR POLYMORPHIC VI

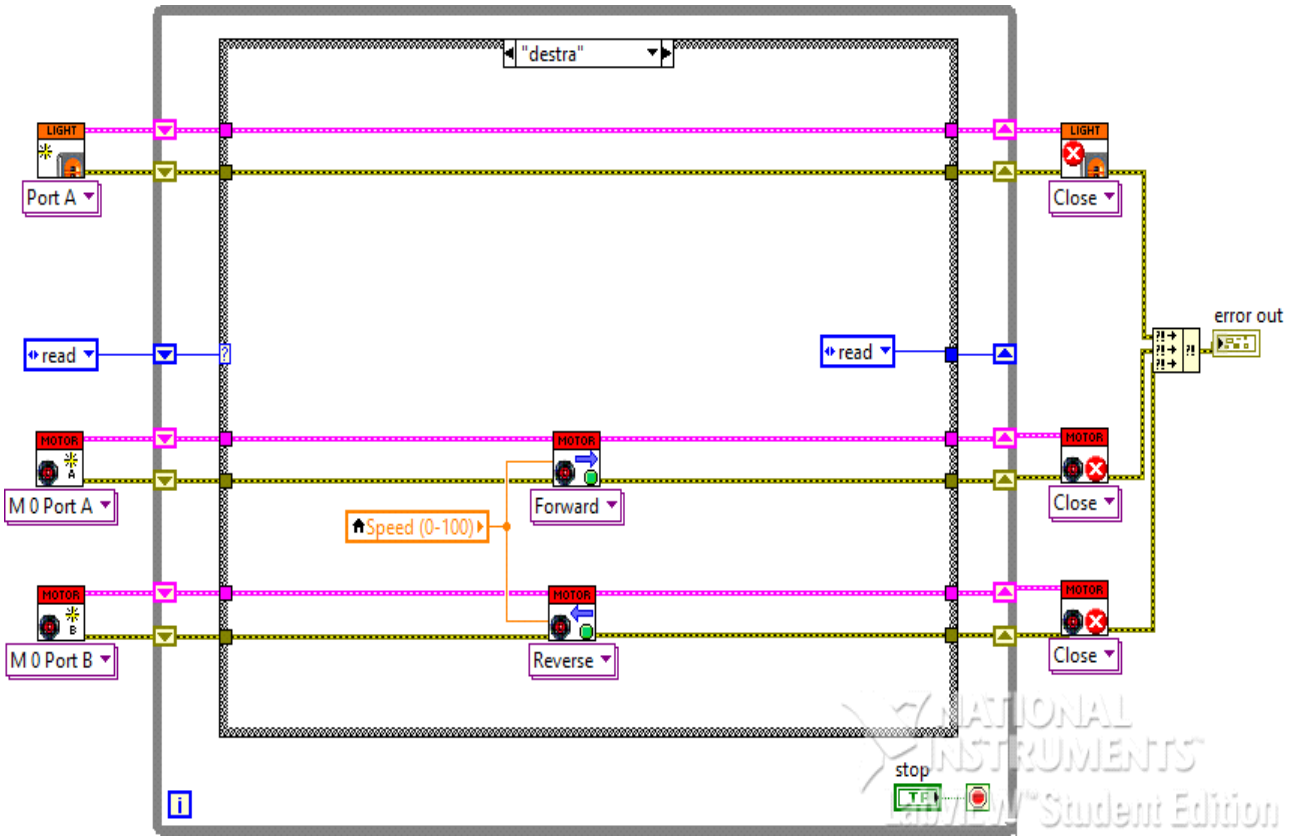
Init Port A Motor_0.vi  Il VI Crea l'istanza per il comando del motore M_0	Per azionare il motore viene utilizzato il driver **L298**. Sono utilizzati i canali digitali **DIO2 (pin 15)** e **DIO3 (pin 17)** del **PortA** per la direzione. I due canali devono essere rispettivamente collegati all' **input1** e **input2** del **driver**. Attraverso il canale **PWM0 (pin 27)** del **PortA** si andrà ad abilitare l'**EnableA** del **driver**, utilizzato per il controllo della velocità, grazie alla possibilità di variare il dutycicle del segnale d'ingresso. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.
Init Port B Motor_0.vi  Il VI Crea l'istanza per il comando del motore M_0	Per azionare il motore viene utilizzato il driver **L298**. Sono utilizzati i canali digitali **DIO2 (pin 15)** e **DIO3 (pin 17)** del **PortB** per la direzione. I due canali devono essere rispettivamente collegati all' **input1** e **input2** del **driver**. Attraverso il canale **PWM0 (pin 27)** del **PortB** si andrà ad abilitare l'**EnableA** del **driver**, utilizzato per il controllo della velocità, grazie alla possibilità di variare il dutycicle del segnale d'ingresso. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.
Init Port A Motor_1.vi  Il VI Crea l'istanza per il comando del motore M_1	Per azionare il motore viene utilizzato il driver **L298**. Sono utilizzati i canali digitali **DIO4 (pin 19)** e **DIO5 (pin 21)** del **PortA** per la direzione. I due canali devono essere rispettivamente collegati all' **input3** e **input4** del **driver**. Attraverso il canale **PWM1 (pin 29)** del **PortA** si andrà ad abilitare l'**EnableB** del **driver**, utilizzato per il controllo della velocità, grazie alla possibilità di variare il dutycicle del segnale d'ingresso. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.
Init Port B Motor_1.vi  Il VI Crea l'istanza per il comando del motore M_1	Per azionare il motore viene utilizzato il driver **L298**. Sono utilizzati i canali digitali **DIO4 (pin 19)** e **DIO5 (pin 21)** del **PortB** per la direzione. I due canali devono essere rispettivamente collegati all' **input3** e **input4** del **driver**. Attraverso il canale **PWM1 (pin 29)** del **PortB** si andrà ad abilitare l'**EnableB** del **driver**, utilizzato per il controllo della velocità, grazie alla possibilità di variare il dutycicle del segnale d'ingresso. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.

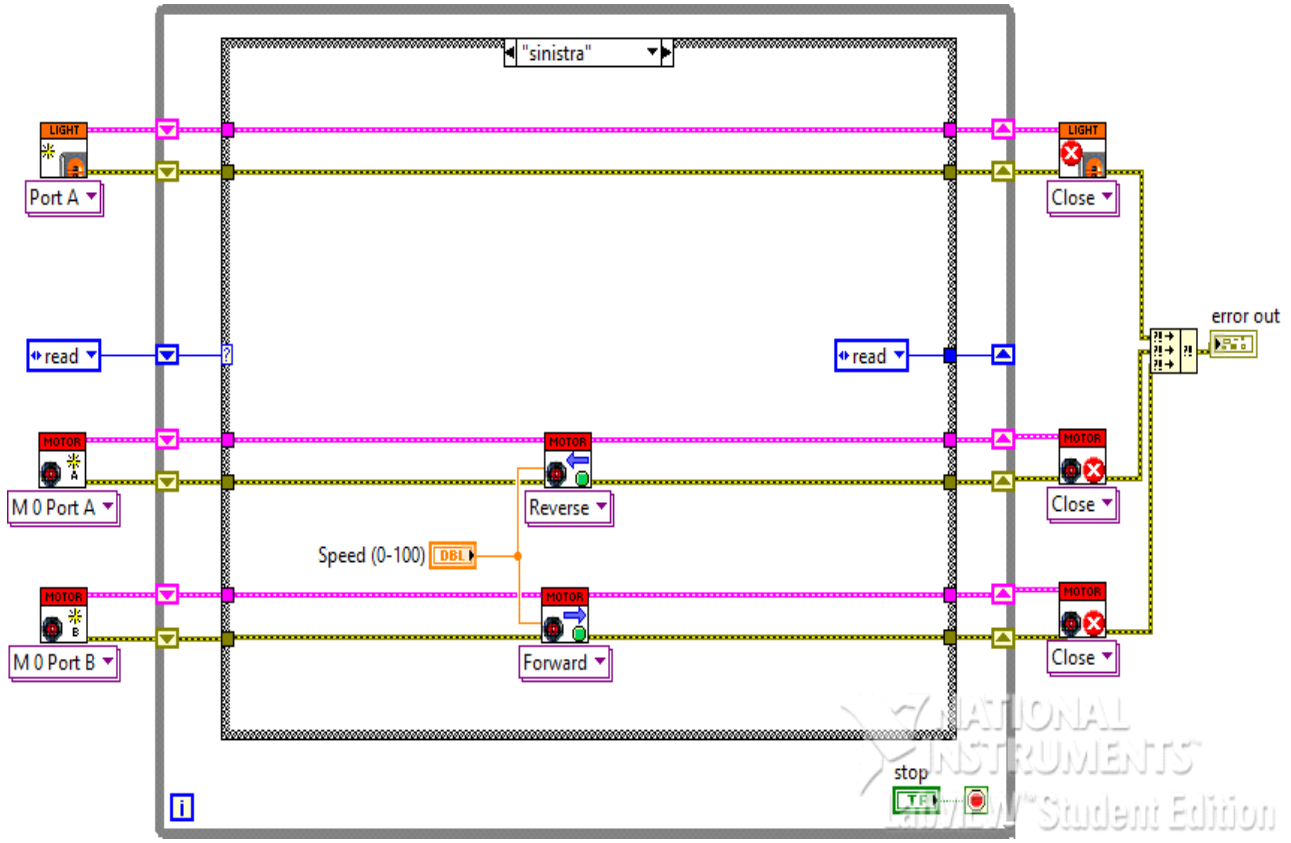
Motor Forward.vi  Il VI fa ruotare il motore in avanti.	**Task In:** riceve in ingresso l'istanza precedentemente creata. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma. **Speed:** varia la velocità del motore da 0 a 100.
Motor Reverse.vi  Il VI fa ruotare il motore all' indietro.	**Task In:** riceve in ingresso l'istanza precedentemente creata. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma. **Speed:** varia la velocità del motore da 0 a 100.
Motor Break.vi  Il VI Permette la frenata d'emergenza del motore.	**Task In:** riceve in ingresso l'istanza precedentemente creata. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.
Motor Slow Down .vi  Il VI Permette di rallentare il motore senza bloccarlo.	**Task In:** riceve in ingresso l'istanza precedentemente creata. **Task Out:** trasmette l'istanza in uscita. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.
Motor Close.vi  Il VI chiude l'istanza del motore.	**Task In:** riceve in ingresso l'istanza precedentemente creata. **error In:** il terminale è usato per connettere VI precedenti per stabilire il flusso del programma. **error Out:** il terminale è usato per connettere VI successivi per stabilire il flusso del programma.

Line follower

Il seguente programma è un semplicissimo esempio di applicazione del tipo "line follower", che ci permette di controllare un prototipo di robot in grado di seguire una linea di color nero su di uno sfondo bianco grazie all'utilizzo del sensore reflex o light sensor di LEGO, collegato all'ingresso analogico AI0 della porta A di myRIO (PORT_A), e di due motori per far ruotare le ruote posteriori indipendenti. Nel robot ognuno dei motori trasmette potenza ad un semiasse posteriore. In tal modo siamo riusciti ad ottenere un veicolo in cui l'avanzamento è realizzato mediante l'azionamento concorde e sincrono di entrambi i motori, mentre la sterzata dal rallentamento o, più incisivamente, dall'inversione del motore interno alla curva. Il tutto è reso possibile dall'appoggio sulla ruota pivotante anteriore.




"destra"
LIGHT
Port A
LIGHT
Close
read
read
error out
MOTOR
M 0 Port A
MOTOR
Forward
Speed (0-100)
MOTOR
Close
MOTOR
M 0 Port B
MOTOR
Reverse
MOTOR
Close
stop
TF
i


"sinistra"
LIGHT
Port A
LIGHT
Close
read
read
error out
MOTOR
M 0 Port A
MOTOR
Reverse
Speed (0-100)
DBL
MOTOR
Close
MOTOR
M 0 Port B
MOTOR
Forward
MOTOR
Close
stop
TF
i

Il programma è composto da tre parti principali, come d'altronde tutti i programmi realizzabili con l'utilizzo delle librerie myRSD.

La prima parte è l'inizializzazione, la seconda è la lettura del valore del sensore e il controllo dei motori, l'ultima è la chiusura dell'istanza.

Nell'inizializzazione, il VI di Init PortA, è in grado di creare l'istanza del Light Sensor (canale AI0 PortA), cioè ci permette di dichiarare che stiamo utilizzando il pin AI0 del PortA come pin Analogico in ingresso.

Vengono inoltre fatte le inizializzazioni dei due motori.

Tramite il filo dell'istanza (color viola) l'istanza viene trasportata per tutto il programma.

Il programma è basato su una struttura chiamata macchina a stati, composta da tre stati: read, destra e sinistra.

La lettura (read) consiste appunto nella lettura del valore del Sensore reflex. Il valore acquisito, viene poi comparato grazie all'utilizzo di un comparatore a finestra. Se il valore è corretto (compreso tra 2,4 e 2,6 ovvero il robot sta seguendo la linea), il programma resta nello stato di lettura.

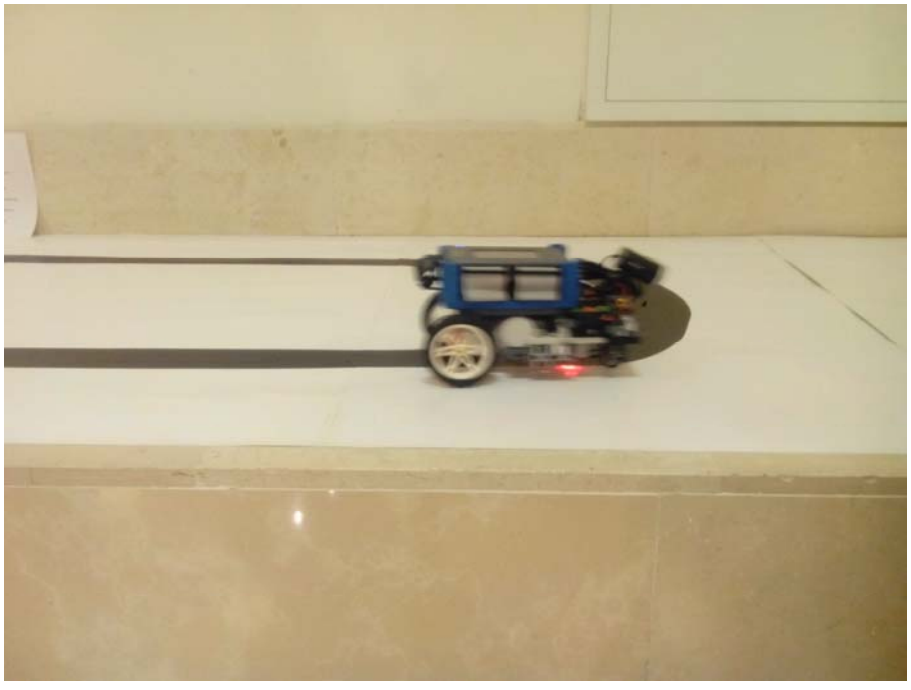
Invece se il valore è minore o maggiore delle due soglie, vengono prese delle decisioni.

Viene fatto ruotare il robot verso destra o sinistra, in base al valore letto, se "troppo bianco" o "troppo nero".

Nell'ultima fase, quella di chiusura dell'istanza, il blocco di close permette all'utente di chiudere i canali di comunicazione, (AI0 e dei motori) in modo da liberare le istanze per poterli poi riutilizzare.

Grazie al filo dell'errore (giallo), siamo in grado di dare una sequenzialità alle funzioni svolte dal nostro programma ed inoltre abbiamo la possibilità di debugger, in modo da rilevare eventuali errori durante lo svolgimento del programma.

Nella foto seguente è possibile vedere il nostro piccolo rover LEGO all'azione grazie al semplice algoritmo di controllo sopra esposto e completamente sviluppato dagli allievi coinvolti.



In conclusione, la piattaforma myRSD offre agli sperimentatori una soluzione completa (hardware e software) per il controllo di robot didattici realizzati con LEGO Mindstorm. La piattaforma è costituita da una scheda d'interfaccia d'interconnessione tra l'ecosistema LEGO Mindstorm ed il potente PC-Embedded myRIO e dalle necessarie librerie software, per la programmazione in LabView del myRIO stesso.

Per il futuro pensiamo di poter affinare e completare questa piattaforma con una seconda versione del PCB ed anche con ulteriori e più sofisticate applicazioni, come quelle che prevedano l'impiego di una Webcam ed il toolkit Vision di LabVIEW per il riconoscimento di oggetto. Questo sviluppi saranno naturalmente possibili a condizione di poter coinvolgere una "community" di Istituti desiderosi di cimentarsi assieme a noi su questo ambizioso progetto nello spirito della condivisione delle idee per lo sviluppo del mondo educational. Nell'immagine sotto riportata è visibile lo "stand" dell'Istituto Tecnico F. Severi di Padova in occasione dell'NIDays 2015, tenutosi a Milano il 12 Marzo di quest'anno, dove National Instruments Italia ci ha gentilmente ospitati.

Contatto per informazioni tecnico-scientifiche sul progetto e sulle possibilità di collaborazione con l'Istituto:

prof. ing. Alessandro Scroccaro, didattica@ingscroccaro.it ITIS Francesco Severi Padova, tel. 049 86 58 111

Contatto per informazioni tecnico-commerciali sul progetto e sulle possibilità di collaborazione:

dott. Massimo Rapini, rapini@irsweb.it IRS Padova, PADOVA - cel. 347 1293915 - tel: +39 049 8705156