ROVIS - ROver for VIdeo Streaming

Corso di Fisica dei Sistemi Complessi - Prof. Sandro Rambaldi

Alessandro Cordella
Natale Vadalà
alessandro.cordella@studio.unibo.it
natale.vadala@studio.unibo.it

Novembre 2018

Indice

1 Introduzione

L'obiettivo del presente lavoro è stato quello di costruire un robot capace di muoversi su ruote e trasmettere un canale di *streaming video over HTTP*. Il dispositivo ottenuto è controllabile da un'interfaccia web ed è possibile visionare ciò che è posto di fronte al robot tramite una webcam.
Nelle seguenti sezioni verranno descritte le fasi di progettazione e realizzazione, i componenti hardware e le tecniche software utilizzate, e verranno presentati eventuali sviluppi futuri.

2 Progettazione

La fase di progettazione è stata condotta attraverso l'analisa della specifica e lo studio di progetti open-source precedenti.

2.1 Specifica

Progettare e realizzare un robot, controllabile attraverso un'interfaccia Web, dotato di una webcam in modo da poter trasmettere lo streaming video sulla stessa interfaccia.
Inoltre, si richiede l'uso microcontrollore o di un single-board computer, di componenti low-cost e software open-source.

2.2 Analisi dei requisiti

Hardware Ciò che serve per realizzare ROVIS:

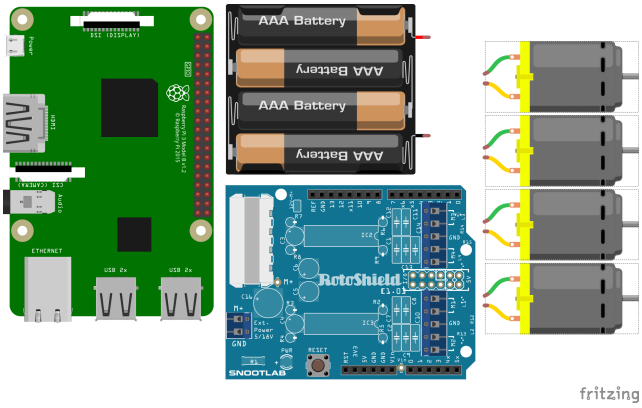
- Microcontrollore / Single-board computer
  Si è deciso di utilizzare un RaspberryPi 3 model B[1], che presenta nativamente una scheda di rete wireless, sebbene vada bene un qualsiasi Raspberry dotato di scheda di rete (interna o esterna) e porta USB per la webcam.

- Una "carrozzeria" con tutti gli alloggi necessari e 4 motori continui
  Per semplicità si è deciso di acquistare un *case*[2] che comprendesse già i motori fissati, evitando di incorrere in ulteriori future complicazioni con l'asseblaggio e messa in asse delle ruote, del moto, ecc...

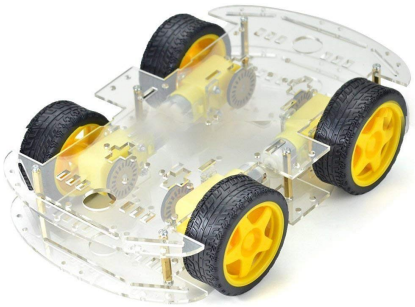
[1] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
[2] https://www.diymore.cc/collections/robot-chassis/products/diymore-4-wheel-robot-chassis-smart-car-with-speed-and-tacho-encoder-for-arduino-raspberry-pi-robot-diy-kits-65x26mm-tire

- Una scheda che faccia da driver per i motori
  Acquistata su Amazon.it[3], una shield con dei chip L293D[4] (driver per motori molto comuni), permette di collegare i motori e gestirne la direzione (moto orario/antiorario) e la velocità, ottebibili grazie allo sfruttamento di alcuni pin GPio d'appoggio per il controllo singolo di ogni motore ai e ponti H per permettere i cambi di direzione. Va alimentato con 5V.

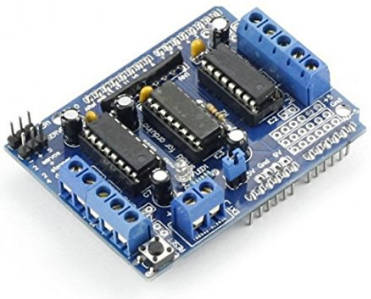
- Un Power-bank e un alloggiamentoper per shield e motori
  Si è deciso di utilizzare un power bank per il RaspberryPi per il semplice motivo che è più comodo, a nostro avviso, servire l'alimentazione da una porta micro USB invece che saldare un ulteriore alloggiamento per batterie sul RaspberryPi (Ma andrebbe comunque un'alimentazione uguale a quella per i motori e la shield, cioè 4* batterie AAA quindi 4.8-5V).



(a) RaspberryPi, case per batterie, 4 motori continui e L293D drive shield



(b) *Case* con motori e ruote

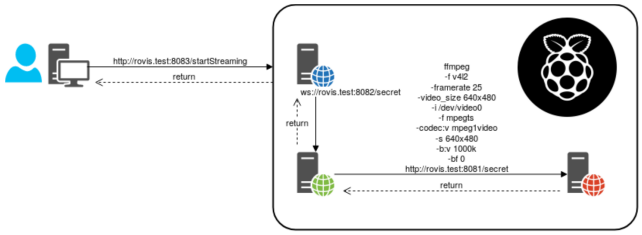


(c) L293D drive shield

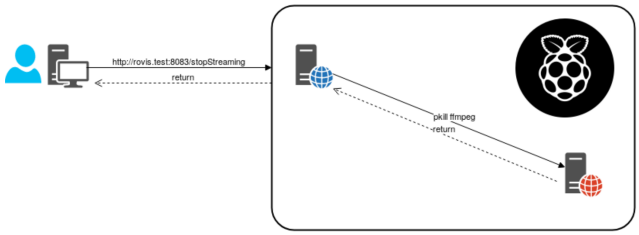
[3] http://amzn.eu/d/a9PD82e
[4] https://www.robot-italy.com/it/l293d-motor-driver.html

Software

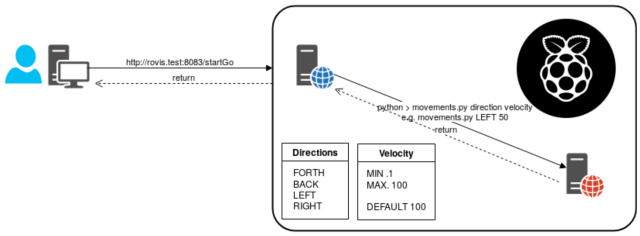
scrivere un po' di cose, domani sarò fatto



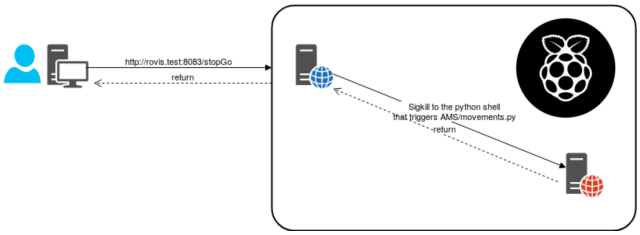
(a) Caso d'uso 1: ***stop streaming***



(b) Caso d'uso 2: ***stop streaming***



(c) Caso d'uso 3: ***start go***



(d) Caso d'uso 4: ***stop go***

3 Realizzazione

3.1 Hardware

3.2 Software

Per garantire il controllo del dispositivo tramite un'interfaccia web è stato necessario progettare un server al quale un client può collegarsi per ottenere il controllo del dispositivo. È stato utilizzato il framework javascript *NodeJs* per

gestire sia la parte client che server dell'applicazione. Nelle seguenti sezioni verrà descritta la struttura gerarchica delle cartelle e le funzionalità presenti nei file al loro interno.

Server I file presenti all'interno della cartella *server* contengono le funzioni per gestire la parte server del sistema, mentre all'interno delle sue sottocartelle sono presenti altri file che verranno descritti in seguito.

main.js Il file *main.js* è il file principale del server. Al suo interno sono presenti le funzioni di inizializzazione del sistema e per la gestione dello streaming video.
Per avviare il server è necessario digitare il seguente comando:

```
node main.js ciao 8081 8082 8083
```

Listing 1: Avvio server

- **node** nodejs;
- **main.js** file del server;
- **ciao** parola chiave;
- **8081 8082 8083** qualcosa

cambiare ciao

scrivere

Una volta lanciato il comando, sarà possibile accedere all'interfaccia web digitando l'indirizzo IP del raspberry seguito da *:8083*.

```
192.168.1.10:8083
```

Listing 2: Esempio indirizzo server

routes.js Questo file è quello che gestisce la comunicazione client server, associando ad ogni interazione del client sulla pagina web una specifica azione del server. Gestisce quindi le funzioni che regolano la webcam e il movimento. In particolare viene utilizzata una libreria di nodejs per eseguire la libreria Python *AMSpi* per la gestione dei motori.

AMSpi *AMSpi* è la libreria Python utilizzata per controllare i motori. Al suo interno è presente il file *movements.py*. È il file che usa le funzioni della libreria per controllare i motori.

Client I file presenti all'interno della cartella *public* sono quelli che gestiscono la parte client dell'applicazione.

ex3-1.html È il file contenente la struttura del sito web. immagine

css All'interno di questa cartella sono presenti i file che regolano lo stile della pagina.

js In questa cartella sono contenuti i file che regolano le interazioni dell'utente con la pagina e comunicano al server le azioni intraprese da esso.

4 Ulteriori sviluppi e Conclusioni