Corso di Sistemi Operativi – Anno accademico 2017/2018

Università degli Studi di Trento

Gestore I/O a 8 bit su Raspberry

Bonora Matteo

Fronza Massimiliano

Tagliaro Carlotta

Zanella Davide

Descrizione Generale

Nel progetto I/O a 8 bit su Raspberry è necessario leggere 8 valori in input (interpretati come singoli bit 1/0), eseguire delle operazioni su di essi e restituire altrettanti valori in output. Prima di fare questo abbiamo implementato un controllo per assicurarci di star eseguendo il programma su un Raspberry.

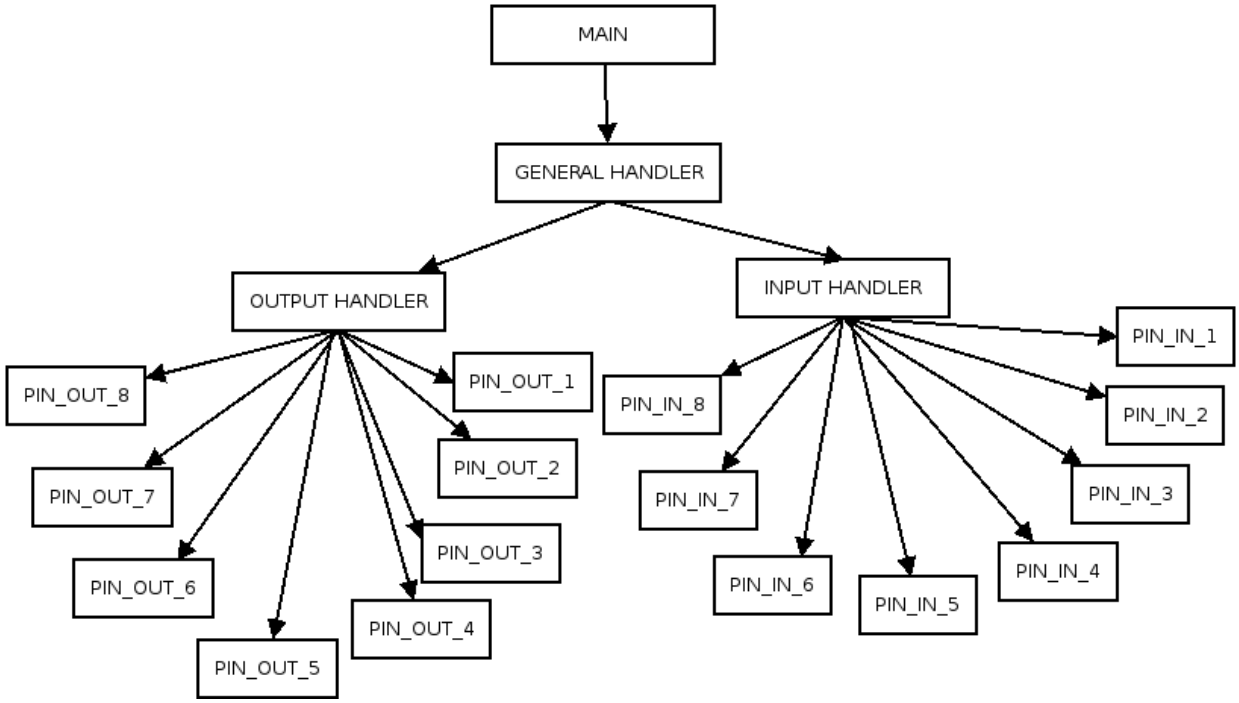
Per fare questo abbiamo creato 20 processi. Il processo main chiama il processo general handler, il quale esegue il vero e proprio programma. Questo va a creare altri due processi, i gestori di input e di output che sono incaricati di ricevere e inviare segnali ai pin del Raspberry.

Il programma principale, prima di avviare i vari handler, va a leggere dai parametri passati dall'utente i file di configurazione dei pin di input/output (se non specificati vengono usati i pin di default) e il numero di uova da gestire.

L'input handler, dal momento che deve leggere i segnali di input (interpretati come dei bit) di 8 diversi pin, va a creare 8 processi figli che serviranno a gestire singolarmente questi segnali.
Ad ognuno di questi 8 processi viene passato l'identificativo numerico del pin di input di cui deve andare a leggere il valore. Questi identificativi (insieme a quelli dei pin di output) vengono presi da un file di configurazione esterno.

Una volta ottenuti questi bit, il gestore di input va a salvarli in un array di 8 interi (in modo da salvare lo stato attuale). Questo array viene poi passato al general handler che andrà ad implementare la logica per trasformare questi 8 valori iniziali in un array risultante, secondo quanto stabilito dalle specifiche del progetto.

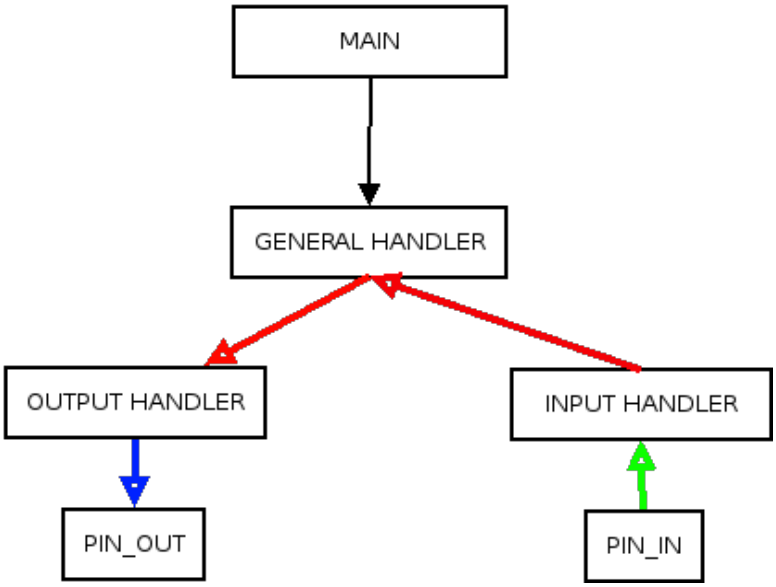
L'array modificato viene poi passato al gestore di output, il quale è incaricato di inviare i singoli valori dei bit ai rispettivi 8 pin di output. Per fare questo anch'esso sfrutta 8 processi differenti che andranno a scrivere il valore passato sul pin di output, così da accendere/spegnere il led a lui associato (ed avere un feedback esplicito di quanto si sta facendo).



Schema generale della struttura del programma

IPC: message queues

Per implementare la comunicazione tra processi abbiamo usato le message queues. Una di queste è stata usata per far comunicare tra loro i processi di input, che gestiscono i singoli pin fisici, con l'input handler. Il corpo del messaggio è composto dal valore del pin (un bit 1/0); inoltre ogni messaggio ha un identificativo uguale al numero del pin, in modo da rendere possibile la distinzione tra messaggi. Infatti, una volta che l'input handler riceve un messaggio, esso deve sapere a quale pin è relativo per scrivere il valore nella giusta posizione dell'array di 8 valori da passare all'handler.

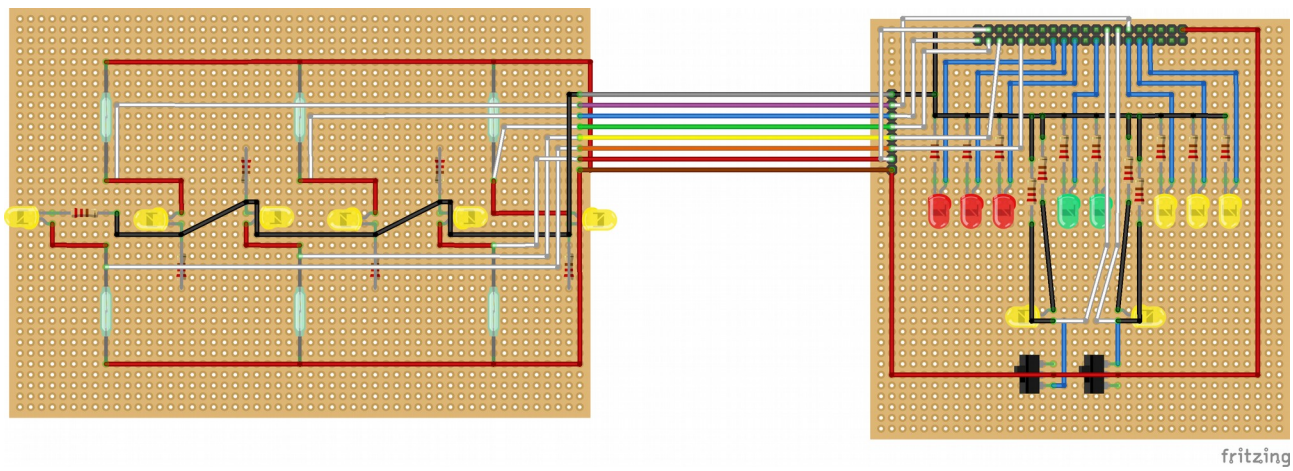


La seconda message queue implementata è stata usata per gestire il "passaggio" degli array di input e dell'array di output tra l'input handler, il general handler e l'output handler. Per distinguere i messaggi sono stati usati due valori: 1 se il messaggio proviene dall'input handler e deve essere letto dal gestore generale, 2 se il messaggio è inviato dal gestore generale verso l'output handler. In questo modo, avendo due tipi diversi, non si rischia che due processi vadano a leggere qualcosa che non spetti a loro (in questo caso il general handler potrà leggere solo messaggi di tipo 1 mentre l'output handler solo quelli di tipo 2).

Infine la terza coda è quella usata tra l'output handler e gli 8 singoli processi dei pin di output. Come con la prima coda il messaggio trasportato è il valore da scrivere nel pin (bit 1/0) e l'identificativo è il riferimento numerico al pin. A differenza della prima coda (in cui l'input handler legge tutti i valori dei messaggi e poi li posizione nella giusta posizione dell'array) i processi in ricezione vanno solo a leggere i messaggi con tipo uguale al pin su cui devono andare a scrivere.

Hardware

L'interfaccia di gestione è realizzata interamente in hardware, senza richiedere componenti software aggiuntive per gestire le funzioni del programma.



Scheda sensori (sinistra) e pannello di controllo (destra)

L’hardware è composto da due schede: Il pannello di controllo e la scheda sensori.

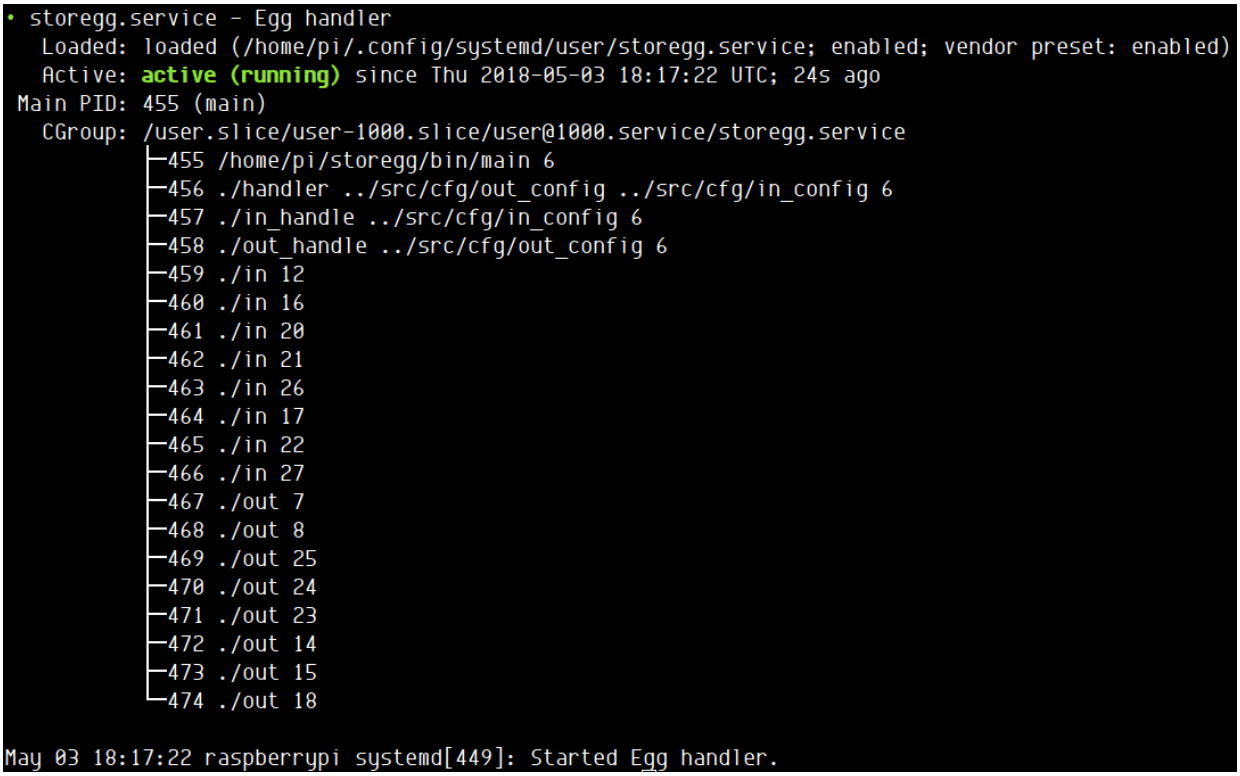
Il pannello di controllo è composto da 8 LED che mostrano la quantità di uova in magazzino, nel cartone e da ordinare, e da 2 interruttori che controllano quante uova sono presenti in magazzino. Per interfacciarsi con le altre schede esso presenta anche un header per la connessione alla scheda Raspberry Pi e uno per la connessione alla scheda sensori, in modo che le schede siano indipendenti tra loro.

La scheda sensori è composta da 6 sensori di tipo magnetico (reed) e 6 LED collegati direttamente ai sensori che servono a verificare che essi reagiscano alla presenza delle uova.

Autostart

Il progetto è stato pensato e implementato in modo che sia utilizzabile senza doversi connettere al raspberry via ssh, facendolo quindi funzionare in piena autonomia.

Una volta acceso il raspberry, esso eseguirà direttamente il programma main e di conseguenza tutti i relativi sotto-processi necessari come descritto nel primo capitolo di questa relazione. Per fare questo è stato creato un servizio linux che viene gestito e avviato in automatico da systemd, ossia il gestore dei servizi su sistemi operativi linux.



Output del comando status relativo al servizio in cui si possono vedere tutti i processi generati