Report - Progetto Sistemi Operativi 2018/19

De Rebotti Lorenzo - 1745942

October 11, 2019

1 What

Questo progetto consiste nella realizzazione di un **remote-robot-controller** che consente di guidare il robot *MARRtino*.
Il computer che si trova sul robot è l'*host* e sarà indicato con **H**, il computer a cui è connessa la periferica di controllo è il *client* e sarà indicato con **C**.

2 How

I punti chiave del progetto sono:

- lettura di un input da una periferica, nel nostro caso un joypad;
- acquisizione di un'immagine da una periferica video, nel nostro caso la webcam PSEye.

Riguardo alla lettura del joypad ho utilizzato la system call *open()* che prende come argomento il percorso del file, in particolare */dev/input/js0*, i parametri di lettura/scrittura e ritorna il file descriptor associato. Successivamente il file descriptor viene ciclicamente letto dalla funzione *read()* ogni 1 ms e i valori appena letti sono conservati nella struttura dati *jsevent*. Questi valori vengono poi copiati nella struttura dati JoyPacket che contiene *axis*, *value* e *type* pronti per essere inviati a H.

Per acquisire l'immagine dalla webcam, connessa tramite USB, sono stati necessari più funzioni. Innanzitutto, siccome la camera è considerata come un file dal sistema operativo, viene richiesto il file descriptor associato ad essa tramite la funzione *open()*. Quindi viene inizializzata la struttura dati *camera_t* associando a questa, oltre al file descriptor, la risoluzione e i buffer associati per il frame.

L'inizializzazione vera e propria viene effettuata nella funzione *camera_init*:

1. tramite *xioctl* vediamo le capacità della camera;
2. ulteriore controllo per testare se il dispositivo supporta il cropping;
3. i buffer di memoria sono allocati, per le operazioni DMA;
4. ogni singolo buffer viene mappato in memoria tramite *mmap*.

Lo streaming vero e proprio viene avviato dalla funzione *camera_start()*, che acquisisce i frame e li conserva nei buffer. Tramite *camera_frame()* invece i frame vengono salvati in un buffer disponibile per l'utente.
Infine questo frame viene salvato in formato *.jpeg* e, dopo aver subito una compressione, questo file viene inviato al **C**.

Per connettere **C** e **H** è stata sfruttata una connessione Websocket, utilizzando la libreria *libwebsockets*.
Il programma *rrc_host* in esecuzione su **H** inizializza, in un thread, un server sulla porta 9000 che ha il compito di ricevere i comandi da **C**, mentre invia a quest'ultimo i frame della camera: in un altro thread invece legge i comandi ricevuti e li invia al microcontrollore, nel mio caso è un *ATMega 2560*, che monta il firmware *Orazio*; in un terzo thread viene prima effettuata la procedura per acquisire i frame descritta in precedenza e li salva in una coda circolare a cui accede anche il thread che gestisce le callback. Ho optato per questa soluzione perché il all'interno delle funzioni invocate dal server sarebbe opportuno non effettuare operazioni I/O bloccanti
Il programma *rrc_client* in esecuzione su **C**, dopo essersi connesso al server, legge i comandi del joypad in un thread, mentre nell'altro gestisce l'invio di questi comandi e la ricezione dei frame che vengono mostrati a schermo tramite la libreria *OpenCV*.

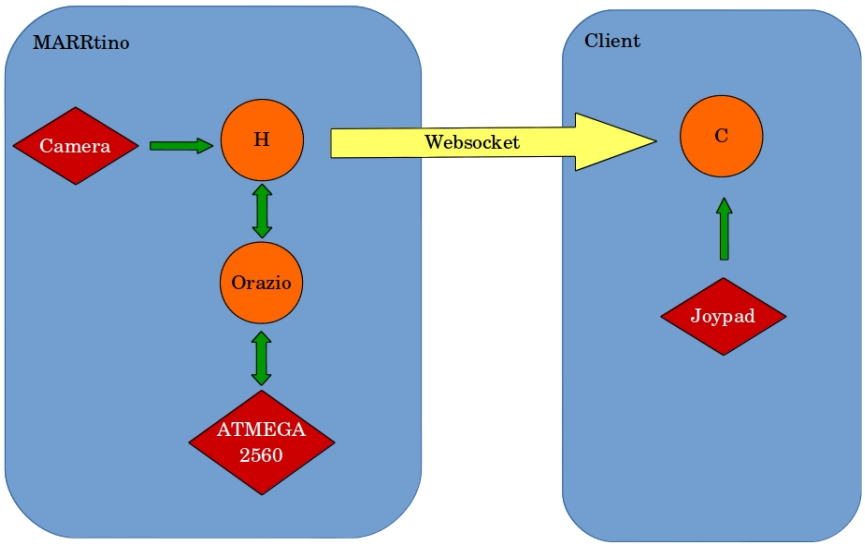

Figure 1: Schema della divisione logica dei componenti

3 How-to-run

Per compilare il programma sono necessarie le librerie aggiuntive *libwebsockets* 3.1.99 e *OpenCV* 2.4.9.1 su entrambi i computer. Successivamente per compilare occore eseguire il comando *make* all'interno della cartella *build*, sia su **C** che **H**.

Per l'esecuzione bisogna eseguire su **H** il comando:
rrc_host [SERIAL-DEVICE] [CAMERA]
Come valori predefiniti abbiamo */dev/tty/ACM0* come serial device e come camera */dev/video0*

Invece su **C** bisogna eseguire il comando:
rrc_client [INPUT-DEVICE] [ADDRESS]
Come input device sarà impostato */dev/input/js0*, mentre l'indirizzo a cui connettersi di default è *localhost*.