Progetto di Domotica

Sistemi Operativi A.A. 2018/2019

Matteo Franzil
Ruben Bettoni
Paolo Baiguera

26 maggio 2019

1 Informazioni generali

Il seguente progetto è stato realizzato da:

- Matteo Franzil (192198)
- Paolo Baiguera (194008)
- Ruben Bettoni (192685)

Il progetto segue le specifiche della *variante con processi*, ed è stato testato sulle versioni Ubuntu 18.04, Ubuntu 19.04, Ubuntu 16.04 LTS.

2 Il sistema

2.1 Avvio del progetto

Per iniziare, entrare nella cartella del progetto e digitare `make build` e poi `./run`. Quest'ultimo è un link simbolico all'eseguibile `launcher`, situato in `/bin/`.

Per rimuovere i file temporanei (la cui posizione è specificata in fondo al documento) e gli eseguibili, digitare `make clear`.

2.2 Logica del progetto

L'idea dietro l'implementazione è che si accede a un "ambiente d'interazione" una volta eseguito `./run`. All'interno di quest'ambiente è possibile eseguire le "interazioni manuali" con i dispositivi, nonché spegnere e riaccedendere la centralina con il comando `user turn shell on/off`.

Quest'ultimo comando aprirà una seconda finestra di terminale, distinguibile per via del colore della `pwd`. Questo terminale permette di controllare la centralina e quindi eseguire tutti i comandi richiesti dall'implementazione.

La stragrande maggioranza dell'inter-process communication è stato implementato tramite message queue, i cui descrittori vengono salvati per comodità nella cartella dei file temporanei del sistema. Tutte le pipe, nonché quasi tutte le stringhe usate nel sistema, hanno come dimensione simbolica `MAX_BUF_SIZE = 1024` che è facilmente modificabile.

Per implementare la comunicazione tra dispositivi si utilizzano prevalentemente i segnali `SIGUSR1` (per la richiesta di informazioni) e `SIGUSR2` (per l'interazione con il dispositivo), mentre altri segnali di sistema vengono usati per la cancellazione come `SIGINT`.

Il protocollo usato per comunicare le informazioni dei dispositivi è descritto in dettaglio nella sezione successiva.

2.3 Protocollo di comunicazione

All'arrivo di un `SIGUSR1`, un dispositivo invia sulla message queue connessa al padre una stringa separata da pipe `|`: i parametri più importanti sono:

- identificativo del dispositivo
  - 0 → centralina
  - 1 → lampadina
  - 2 → frigo

  - 3 → finestra
  - 4 → hub
  - 5 → timer
- PID del dispositivo
- Indice del dispositivo
- Stato (generalmente acceso/spento, con informazioni sull'override per i dispositivi di controllo)

Seguono dei parametri, specificati nella lista dei dispositivi, propri dei registri assegnati al singolo dispositivo.

Per i dispositivi di controllo (hub e timer) la stringa continua con le informazioni sui figli (il numero di figli è specificato nell'ultimo parametro) con la seguente sintassi:

- `<!` - indica l'inizio della lista dei figli; il parametro successivo sarà l'identificativo del primo figlio
- `!` - indica la fine di un figlio
- `!>` - indica la fine della lista dei figli

Questa sintassi può essere nestata a più livelli, in modo da avere una quantità arbitraria di dispositivi di controllo uniti tra di loro.

2.4 Tipi di dispositivi

Per i dispositivi, a fianco è segnalata una lista di registri che vengono inseriti nel protocollo descritto in precedenza.

- Launcher
  - Può accendere/spegnere la centralina (`user turn shell on/off`)
  - Può chiedere le informazioni ai device (`info <id>`)
  - Simula lo switch manuale dei dispositivi (`switch <id> nome_interruttore> <posizione>`)
- Centralina - conosce solo i pid dei dispositivi collegati direttamente ad essa, così come gli hub e i timer. Se la centralina viene spenta dal Launcher, si rifiuterà di ricevere comandi ma manterrà lo stato dei dispositivi, a meno che non venga esplicitamente uccisa con `exit`
  - Può creare i dispositivi (`add <tipo_device>`)
  - Può distruggere i dispositivi (`del <id>`) - per problemi implementativi, tutti i dispositivi figli del dispositivo distrutto vengono distrutti a loro volta
  - Può mostrare la struttura ad albero del programma (`list`)
  - Può chiedere le informazioni ai device (`info <id>`)
  - Simula lo switch manuale dei dispositivi (`switch <id> nome_interruttore> <posizione>`); il secondo dispositivo deve essere obbligatoriamente un dispositivo di controllo (hub, timer)

  - Può collegare due dispositivi tra loro (`link <id> to <id>`)

- Lampadina - `(tipo|pid|indice|stato|tempo_accensione)`; mantiene lo stato e il tempo di accensione

- Frigo - `(tipo|pid|indice|stato|tempo_accensione|delay|temperatura|riempimento)`; mantiene lo stato, il tempo di apertura, la temperatura e a percentuale di riempimento. Dopo un tempo di delay (modificabile) il frigorifero si chiuderà in automatico, anche se comandato da un timer o da un hub.

- Finestra - `(tipo|pid|indice|stato|tempo_apertura)`; mantiene lo stato, il tempo di apertura. Ha due interruttori diversi per aprire/chiudere la finestra, che sono soltanto di tipo attivo.

- Hub - `(tipo|pid|indice|stato|numero_figli_collegati<! ... ! ... ! ... !>)`; può supportare dispositivi di qualsiasi tipo. Se viene modificato il suo stato, questo modificherà lo stato di tutti i suoi figli. Se vengono chieste informazioni ad esso, verrà mostrato tutto il suo sottoalbero e se è in override.

- Timer - `(tipo|pid|indice|stato|orario_apertura|orario_chiusura)`; può essere settato un orario di accensione e di spegnimento per un qualsiasi dispositivo, questo verrà rimandato ogni giorno ciclicamente. Se viene settato un range nella quale cade l'orario corrente, il timer si accenderà in automatico

2.5 File utilizzati

All'interno del progetto vengono usate le cartelle `src/` per i sorgenti, `bin/` per gli eseguibili. Fuori dalla cartella principale viene fatto uso della sola cartella `/tmp/ipc`, dove vengono create le FIFO, e `tmp/ipc/mqueues`, dove vengono create le message queues.

All'interno della cartella `src/` si trovano `launcher.c` e `shell.c` che contengono gli eseguibili del launcher e della shell della centralina; `util.c` che accomunato al suo header viene incluso in tutti gli altri file, e contiene tutte le primitive di basso livello per lavorare con le informazioni e le stringhe, oltre a tutte le macro; `actions.c` che contiene le funzioni usate da tutti i dispositivi di controllo (hub, timer e centralina) per eseguire le loro funzioni. All'interno della sottocartella `src/devices/` si trovano gli eseguibili di tutti i dispositivi.