LabSO 2018-2019
Laboratorio Sistemi Operativi 1

PROGETTO LabSO 2018-2019 - Revisione I

SISTEMA DI DOMOTICA

Implementare l'emulazione di un sistema di domotica in modo che ogni dispositivo sia rappresentato da un'unica entità (cartella con uno o più file oppure processo con eventuali sottoprocessi: vedere "varianti" in coda). I dispositivi possono essere di due generi e vari tipi e hanno tutti uno "stato", degli "interruttori" a due posizioni (on/off) e un eventuale registro di parametri (se ne possono aggiungere quanti ritenuti utili oltre a quelli indicati di seguito):

- **Dispositivi di controllo**
  (in "scrittura" setta il corrispondente campo in tutti i dispositivi controllati e in lettura ne mostra il valore che deve essere uguale per tutti salvo "override" manuali o inizializzazione - ad esempio se si collegano delle lampadine già attive in stati diversi ad un hub - nel qual caso deve essere indicato che vi è stato un qualche intervento di tale genere, comportamento indicato come "mirroring" di seguito. Stato iniziale dei dispositivi e posizione iniziale degli interruttori possono essere predefiniti o si può fare un metodo di impostazione. Le informazioni specifiche sui dispositivi controllati - ad esempio lo stato - NON possono essere salvate dentro quello di controllo: devono essere recuperate "interrogandoli" singolarmente)

  - **controller "centralina"**: gestisce l'intero sistema (è unico e non può essere rimosso)

    stato: acceso/spento
    interruttori: generale (on/off per accendere spegnere l'intero sistema)
    registro: num = numero dispositivi direttamente connessi

    Deve aprire una "shell" interattiva in cui si possono eseguire almeno i comandi:

`list`	elenca tutti i dispositivi (quelli disponibili con un nome, quelli attivi anche con un "id" univoco per ciascuno e inoltre ne riepiloga le caratteristiche)
`add <device>`	aggiunge un <device> al sistema (es. "`add bulb`") e ne mostra i dettagli
`del <id>`	rimuove il dispositivo <id>: se è di controllo rimuove anche i dispositivi sottostanti
`link <id> to <id>`	collega i due dispositivi tra loro (almeno uno dei due dev'essere di controllo: *controller*, *hub* o *timer*)
`switch <id> <label> <pos>`	del dispositivo <id> modifica l'interruttore <label> in posizione <pos>, ad esempio:

`switch 3 open on`	imposta per il dispositivo #3 l'interruttore "open" su "on" (ad esempio apre una finestra)
`info <id>`	mostra i dettagli del dispositivo

- **hub**: permette di collegare più dispositivi dello stesso tipo in parallelo tra loro

  stato: "mirroring" dei dispositivi collegati
  interruttori: "mirroring" dei dispositivi collegati
  registro: "mirroring" dei dispositivi collegati

- **timer**: permette di definire una schedulazione (almeno una fascia oraria più eventuali maggiori dettagli come giorni del mese e/o della settimana e/o altro) per comandare un dispositivo collegato

  stato: "mirroring" del dispositivo collegato
  interruttori: "mirroring" del dispositivo collegato
  registro: begin = ora di attivazione, end = ora di disattivazione + eventuali extra

  Se un dispositivo (di interazione) controllato da un altro (hub o timer) è azionato manualmente questo deve funzionare come "override manuale".
  Esempi particolari:
  - se un timer accende una lampadina l'interruttore di quest'ultima si setta su "on", ma se uno manualmente lo sposta su "off" si deve spegnere (e viceversa).
  - se un hub controlla quattro lampadine e lo si accende, queste si attivano tutte: se manualmente se ne spegne una le altre restano accese; interrogando l'hub questo deve indicare di essere "acceso" ma con "override" (cosa che deve essere dedotta dal fatto che c'è incongruenza nei dispositivi controllati)

- **Dispositivi di interazione**

  deve essere possibile azionare gli interruttori e impostare eventuali parametri specifici (ad esempio i tempi di attivazione di un timer) al di fuori del controller (ad esempio con un comando ad-hoc direttamente da terminale) per simulare un'azione completamente esterna all'intero sistema (che non passa dalla centralina "controller")

  - **bulb (lampadina)**: può essere accesa/spenta e ha un singolo interruttore di comando per il cambio di stato

    stato: accesa/spenta
    interruttori: accensione (on/off per accendere/spegnere)
    registro: time = tempo di utilizzo (accesa)

- **window (finestra)**: può essere aperta/chiusa e ha un pulsante di apertura ed uno di chiusura

  stato: aperta/chiusa
  interruttori: apertura e chiusura (on/off per aprire/chiudere: tornano subito in "off" dopo essere stati azionati)
  registro: time = tempo di utilizzo (aperta)

- **fridge (frigorifero)**: può essere aperto/chiuso con un pulsante singolo per apertura/chiusura. Si richiude automaticamente dopo un tempo che può essere variato. Una "percentuale di riempimento" indica quanto contenuto c'è all'interno (0%-100%): è possibile togliere o aggiungere contenuto solo "manualmente". Un termostato/termometro permette di gestire e impostare la temperatura interna.

  stato: aperto/chiuso
  interruttori: apertura/chiusura (on/off per accendere/spegnere), *termostato* (con valore)
  registro: time = tempo di utilizzo (di apertura)
  delay = tempo dopo cui si richiude automaticamente
  perc = percentuale di riempimento
  temp = temperatura interna

Varianti
Sono possibili due implementazioni differenti, una basata principalmente sul file-system, l'altra su processi. In entrambi i casi ci deve essere una gerarchia chiara dei dispositivi.

Variante "file-system", implementazione su file-system: i singoli dispositivi sono rappresentati da "sottocartelle" dentro una cartella principale, ciascuna con uno o più file di testo per le informazioni specifiche (ad esempio un file per lo stato, uno per il registro, etc. oppure uno unico con tutti i dati). I dispositivi controllati devono essere rappresentati con sottocartelle dentro la cartella di quello di controllo in modo che l'albero di tutte le cartelle rappresenti la gerarchia complessiva.
VOTAZIONE: max. 26

Variante "processi", implementazione con processi: i singoli dispositivi sono rappresentati da processi (con eventuali sottoprocessi di supporto) figli di un processo principale, ciascuno contenente le informazioni specifiche . I dispositivi controllati devono essere rappresentati come figli del processo di quello di controllo in modo che l'albero di tutti i processi rappresenti la gerarchia complessiva.
VOTAZIONE: max. 30

In caso di modifica della gerarchia (ad esempio perché un dispositivo controllato cambia il suo controllore) si deve ricostruire correttamente l'albero (eventualmente eliminando cartelle/file o processi e ricostruendoli opportunamente): è quindi necessario recuperare le informazioni, eliminare la rappresentazione (cartella+files o processi) e crearne un "duplicato" con i dati corretti.
Le interazioni "esterne" (che simulano ad esempio un soggetto che accende manualmente una lampadina) devono avvenire con una modalità dedicata (sempre da "terminale bash"): per esempio si può realizzare un eseguibile ad-hoc che riceve come parametri le informazioni necessarie all'interazione (che può utilizzare canali e strumenti differenti da quelli usati all'interno del sistema principale).

L'implementazione minima deve comprendere almeno la "centralina" (con la shell interattiva), un dispositivo di controllo e uno di interazione (ad esempio: centralina, timer, lampadina per due possibili collegamenti: centralina->lampadina o centralina->timer->lampadina). È possibile inoltre implementare ulteriori dispositivi oltre a quelli indicati purchè rispecchino le indicazioni generali fornite e siano ben definiti e descritti.

Implementazione

Creare una cartella principale con denominazione tipo "*LabSO2018-2019__matricola1__matricola2__....*" (con i numeri di matricola dei componenti del gruppo) e all'interno inserire:

- un file README con i nomi completi e i numeri di matricola dei componenti del gruppo oltre ad una sintesi del progetto realizzato con le eventuali peculiarità
- una sottocartella "project" con dentro:
    - un makefile "Makefile" con <u>almeno</u> tre ricette:
        - "help" (default) che mostra brevi info testuali
        - "build" che compila il progetto, la compilazione DEVE avvenire con il tool "`gcc`" e flag "`-std=gnu90`" e nessun altro
        - "clean" che rimuove eventuali file temporanei e riporta tutto allo stato iniziale
    - una sotto-sottocartella "src" con tutti i sorgenti COMMENTATI DIFFUSAMENTE

Il progetto DEVE partire se lanciato da "terminale bash" su s.o. Ubuntu 18.x (specificare comunque se funziona anche su altri ambienti, come Ubuntu 14.x o Ubuntu 16.x, o altri testati) entrando nella sottocartella "project" e digitando "make build" e poi eseguendo i file opportuni. Si può interagire unicamente attraverso il "terminale bash", ma è possibile aprirne più di uno (ad esempio per mostrare interazioni combinate).
Si terrà conto anche di eventuali avvisi/errori o artifici sia in fase di compilazione che di esecuzione, della rispondenza ai requisiti, della soluzione adottata in generale e anche della resa d'utilizzo (ad esempio se ci sono dei feedback chiari a terminale).
Si deve usare il linguaggio C "gnu90" e si possono adottare solo le librerie "standard".

Bisogna "iscriversi" (si indicheranno i dettagli dei componenti del gruppo e la variante scelta) ed effettuare la "consegna" utilizzando dei moduli online che saranno comunicati più avanti tramite la bacheca Esse3 e/o via mail.

Salvo indicazioni successive differenti ad oggi (10/04/2019) si prevede che i moduli saranno pubblicati entro il 18/04/2019 e le scadenze sono fissate in:

iscrizione: entro il 25/04/2019
consegna: entro il 19/05/2019

VERIFICARE EVENTUALI NUOVE REVISIONI PER VARIAZIONI O CORREZIONI DI REFUSI
CONTROLLARE SEMPRE GLI AVVISI!!!