

Smart Door

Premessa

Per la quarta consegna, ho deciso di strutturare il progetto nel seguente modo:

- Arduino → n° 3 Task
- Raspberry → n° 2 Agenti (Thread)
- Android → n° 3 Activity

Analizziamo ora le varie componenti, motivando le scelte prese.

Arduino

BluetoothCommTask

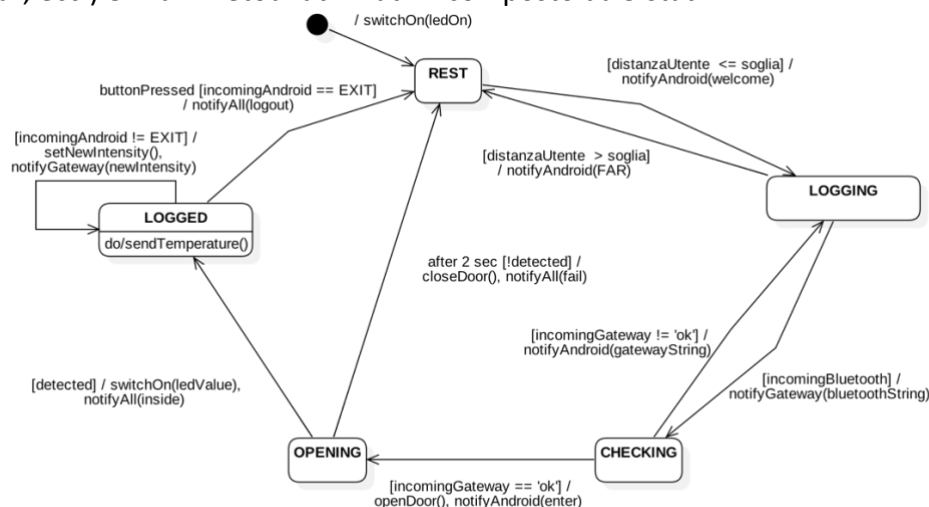
È il task responsabile della lettura dei messaggi in arrivo attraverso il canale Bluetooth riprodotto dalla SoftwareSerial. L'unico compito di tale task è la lettura dei messaggi in arrivo: nel caso vi siano dati in ingresso, il task esegue la lettura fino all'ultimo carattere e setta la variabile globale *incomingBluetooth* al messaggio letto; nel caso non vi fosse nessun messaggio in arrivo, il task resetta la stringa. Questa operazione è possibile e sicura in quanto non vi saranno mai corse critiche, dato che lo scheduler è di tipo Cooperative e i task non possono essere interrotti, ergo quando arriva un messaggio viene letto e il task Smart Door farà in tempo a leggerlo prima che esso venga resettato.

SerialCommTask

È il task responsabile della lettura dei messaggi in arrivo attraverso il canale Seriale di default di Arduino. L'unico compito di tale task è la lettura dei messaggi in arrivo: nel caso vi siano dati in ingresso, il task esegue la lettura fino all'ultimo carattere e setta la variabile globale *incomingSerial* al messaggio ricevuto; nel caso non vi siano messaggi, la variabile globale viene resettata. Questa operazione è possibile per le stesse motivazioni sopracitate nel paragrafo BluetoothCommTask.

SmartDoorTask

È il task responsabile della gestione del sistema Smart Door. In esso vi sono tutte le componenti utili (Pir, Sonar, ecc.) e i vari metodi utilizzati. È composto da 5 stati:



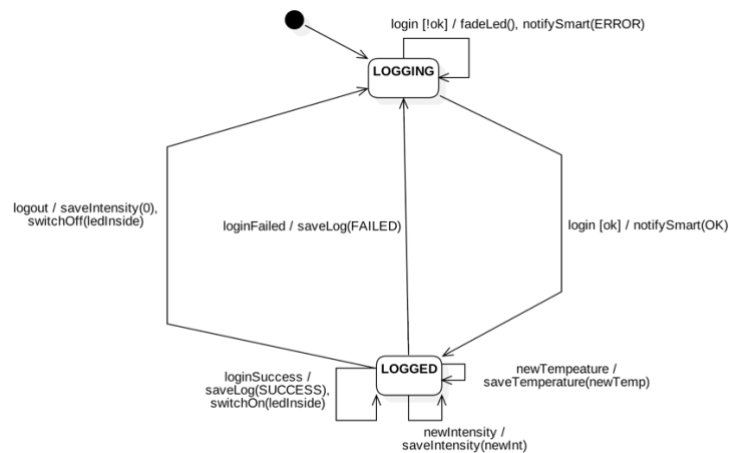
- *REST* → stato di riposo, il sistema è in attesa che l'utente si avvicini alla porta
- *LOGGING* → il sistema è in attesa che l'utente inserisca le credenziali (da Android)
- *CHECKING* → il sistema è in attesa che il gateway gli notifichi il risultato del tentativo
- *OPENING* → l'utente è loggato, il sistema è in attesa che esso entri nella stanza
- *LOGGED* → L'utente è all'interno della stanza, vengono inviati periodicamente i dati relativi alla temperatura e all'intensità (ogniqualevolta cambi dal sistema Android).

Raspberry

InputMsgReceiver

E' l'agente responsabile della ricezione dei messaggi attraverso il canale Seriale. Sostanzialmente sfrutta le tecnologie presentate a lezione (es. *BlockingQueue*, *serialEvent*, ecc.): tramite il metodo *waitForMsg()* l'agente resta in attesa di ricevere un messaggio; quando viene ricevuto, vengono effettuati determinati parsing della stringa letta per notificare il Gateway con i relativi eventi.

Gateway




È l'agente responsabile della logica di tutto il programma. Al suo interno contiene tutte le componenti necessarie per il funzionamento del sistema Gateway (ledInside e ledValue).

All'agente viene passato tramite il costruttore l'istanza dell'oggetto Seriale, in quanto ho deciso che fosse esso stesso il responsabile dell'invio dei messaggi (Si poteva esternare questa proprietà creando un'altra entità responsabile).


Il Gateway possiede il metodo *processEvent* all'interno del quale vi è implementata una macchina a stati finiti (2): lo stato *LOGGING* e lo stato *LOGGED*. Quando il sistema si trova nel primo stato, l'unico evento a cui è interessato è l'evento di Login, in seguito al quale vengono effettuati i controlli di esistenza dell'utente, di conferma dell'account e di correttezza password (interamente criptata sia lato client e lato server). In base all'esito viene notificato il sistema Arduino e, in caso positivo, il Gateway passa nello stato *LOGGED* il quale non è da interpretare come se l'utente fosse già dentro, ma si deve considerare che è loggato in quanto le credenziali e i permessi erano corretti. In questo stato vengono trattati 5 tipi di eventi differenti:

- *Logout* → il sistema deve tornare nello stato iniziale
- *LoginFailed* → l'utente si è loggato correttamente ma non è entrato in tempo nella stanza
- *LoginSuccess* → l'utente si è loggato correttamente ed è entrato nella stanza
- *NewIntensity* → Il sistema deve salvare nel DB il nuovo valore dell'intensità del led
- *NewTemperature* → il sistema deve salvare nel DB il nuovo valore della temperatura.

Il Gateway si interfaccia con DB di tipo *MySQL* grazie al quale è possibile tenere traccia dei log e degli utenti. Ho implementato un *WebServer* sicuro contro vari dei più noti attacchi, dal quale l'utente si può registrare, loggare, cambiare i propri dati e controllare i parametri della stanza.

Embedded_IoTMySmart_Home

Non hai effettuato il login? [Entra](#)



Login Registrati


Email

simonemagnani.96@gmail.com


Password

☐ Ricorda i dati


[Entra](#)



Chi sono
Sono un ragazzo frequentante il 3° Anno di
Ingegneria e Scienze Informatiche a Cesena




Contatti
simone.magnani3@studio.unibo.it
348 6652876




Dove trovarmi
Via Longone 573
Gambettola 47035
Forlì-Cesena, Italia

Grazie per utilizzare il mio software

Embedded_IoTMySmart_Home

Non hai effettuato il login? [Entra](#)



Login Registrati

Email

root

Password

Nome

Mario

Cognome

Rossi

Data di Nascita

gg/mm/aaaa


Città

Roma

Telefono

3332211321

☐ Non sono un robot



[Registra](#)

Impostazioni profilo

Nome: Simone Cognome: Magnani

Email: simonemagnani.96@gmail.com

Data di Nascita: 1996-09-25

Città: Gambettola

Password: [*****]

Modifica password

Password attuale:

Nuova password:

Conferma nuova:


[Modifica](#)

Numero di telefono: 3333333333


Modifica numero

Nuovo numero:


[Modifica](#)



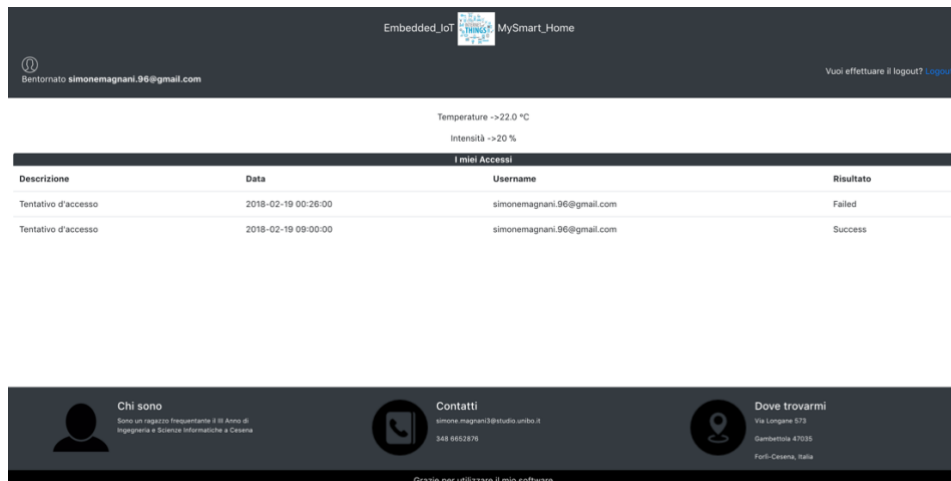
Chi sono
Sono un ragazzo frequentante il 3° Anno di



Contatti
simone.magnani3@studio.unibo.it



Dove trovarmi
Via Longone 573



Android

MainActivity

È l'activity iniziale che si presenta una volta avviata l'applicazione. È responsabile dell'avvio delle operazioni iniziali del sistema, quali la scoperta del dispositivo Bluetooth connesso e la relativa connessione. Una volta scoperto il dispositivo, viene avviato un task asincrono per la relativa connessione con esso, e in seguito viene creato il thread relativo alla connessione: tale thread ha come compito la ricezione e l'invio dei messaggi e la possibilità di notificare le relative activity dell'arrivo di messaggi.

LoggingActivity

È l'activity responsabile del login. In essa vi sono due campi da compilare con le credenziali dell'utente che verranno inviate in seguito al click del pulsante di login al sistema Arduino, il quale a sua volta le inoltra al Gateway per la verifica di tale credenziali. In caso di esito negativo, l'activity mostra all'utente un Toast (avviso) grazie al quale l'utente riesce a capire che tipo di errore è stato riscontrato (password errata, utente non confermato, utente non esistente, ecc.); nel caso le credenziali fossero corrette, viene lanciata attraverso un opportuno Intent l'activity SystemActivity.

SystemActivity

È l'activity rappresentante il sistema una volta che l'utente è loggato ed è entrato all'interno della stanza. In essa sono visualizzati i dati relativi alla temperatura (°C) e all'intensità. Quest'ultima è rappresentata da una SeekBar con il valore impostabile da 0 a 100; in seguito al cambiamento del valore da parte dell'utente, viene inviato un messaggio al sistema Arduino per notificare l'avvenuto cambiamento dell'intensità. Infine vi è un tasto di uscita per effettuare il logout dal sistema con eventuale chiusura dell'applicazione e ritorno allo stato di base delle altre componenti (Arduino e Raspi). Il sistema gestisce in maniera efficiente il caso di chiusura forzata dell'applicazione o uscita inaspettata.