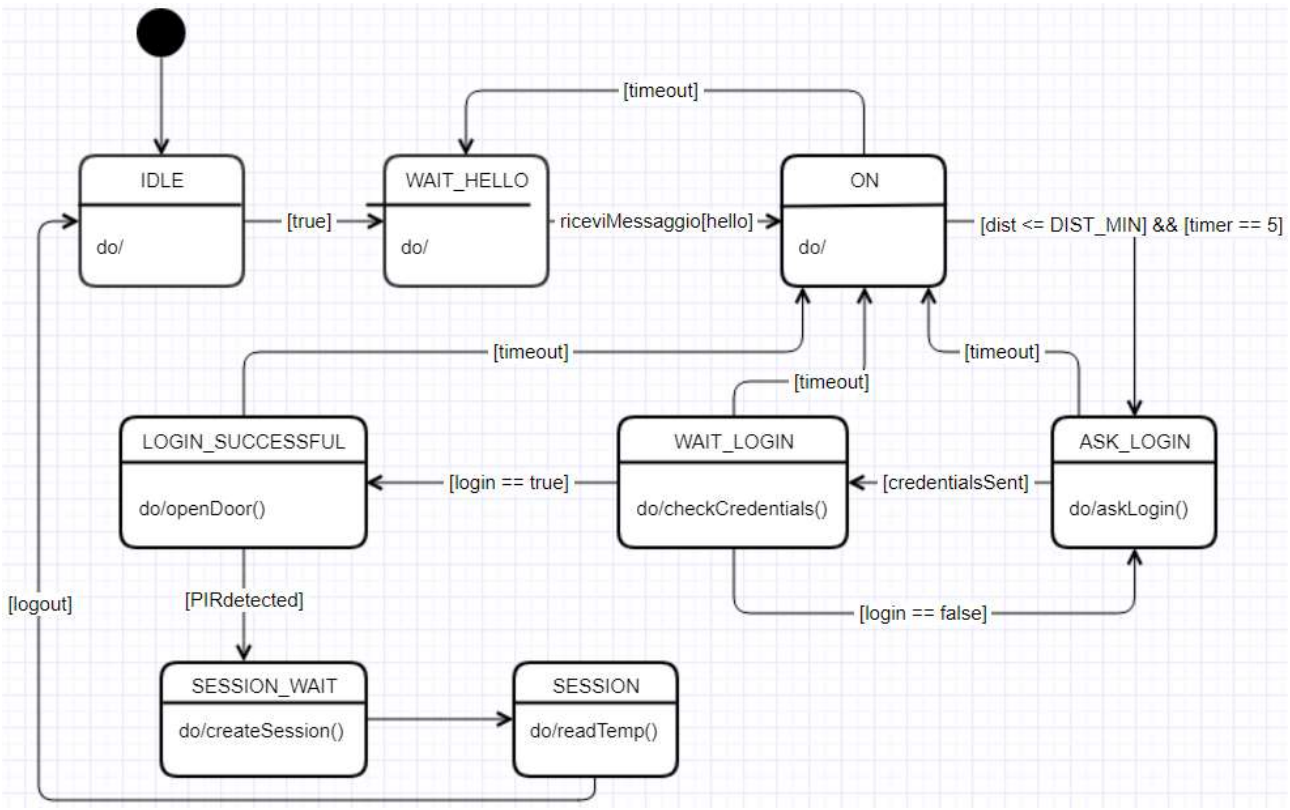
RELAZIONE CONSEGNA N° 4
CORSO SISTEMI EMBEDDED E IOT

Riccardo Marchi, Marco Modigliani, Simone Venturi

Si vuole realizzare un sistema embedded che funge da *varco intelligente*, controllando l'accesso esclusivo ad un certo ambiente mediante una porta. Il sistema è composto a sua volta da tre sottosistemi in comunicazione tra loro: un Arduino ospita la parte DOOR, un Raspberry Pi la parte GW e un dispositivo Android la parte MOBILE. Il fulcro della logica del sistema è la DOOR, la quale comunica e viene gestita via Bluetooth dal MOBILE, mentre fa uso del GW via seriale per gestire e tracciare gli accessi. Su quest'ultimo deve essere in funzione un servizio internet che permetta di ispezionare gli accessi e di leggere i valori rilevati dai sensori dell'Arduino.

ANALISI COMPORTAMENTO ARDUINO



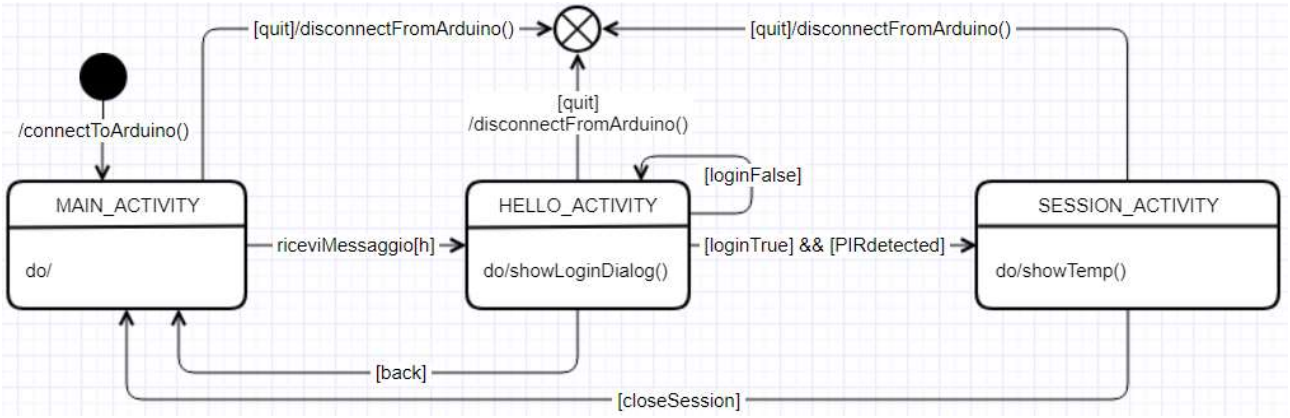
L'aspetto chiave su cui si vuole porre l'attenzione è la gestione dei timeout, i quali sono utilizzati quasi in ogni stato del sottosistema per evitare che, in caso di disconnessioni improvvise da parte del sistema mobile o interruzioni di comunicazione di qualsiasi genere, il sistema si blocchi in un certo punto richiedendo il reset dell'Arduino.

Infatti è fondamentale che il microcontrollore operi in totale sintonia con la parte mobile ed il gateway, essendo nella posizione centrale del sistema, gestendo al meglio la comunicazioni via Bluetooth con Android e quella via seriale con Raspberry Pi. Diversi messaggi identificano differenti richieste da e per la DOOR, la quale mantiene integrità e robustezza grazie alla curata successione degli eventi.
Il lavoro più importante ed oneroso dell'Arduino è rappresentato dal login, in quanto deve trasmettere informazioni da MOBILE al GW e viceversa nel minor tempo possibile senza andare in crash a causa di malfunzionamenti dovuti alla comunicazione.
Di particolare rilevanza è l'aggiunta dello stato di SESSION_WAIT: il suo scopo è quello di rappresentare un punto di sincronizzazione tra Arduino e Android, notando che la comunicazione si corrompeva per un ritardo nella creazione dell'activity relativa alla sessione da parte dell'applicazione mobile.
Una volta nello stato SESSION, l'applicazione termina non appena viene invocato il logout.

ANALISI COMPORTAMENTO RASPBERRY PI

Il ruolo del GW è rappresentato da un Raspberry Pi, il quale ha due funzioni chiave: il primo è quello di ospitare il database MySQL nel quale vengono memorizzati i dati di autenticazione e i log relativi agli utenti; il secondo è quello di mettere a disposizione un servizio internet tramite il quale ispezionare i log e accedere ai dati del sensore di temperatura e al valore di luminosità del led collegati all'Arduino.
Il tutto è stato pensato come processo node.js, basando quindi l'applicazione su un event-loop nel quale vengono interpretati i messaggi in entrata ed eseguite le corrispondenti azioni. Per questa motivazione non è stata fornita alcuna macchina a stati.
Numerose librerie sono state utilizzate a supporto della programmazione, per facilitare l'utilizzo e la gestione di elementi come i GPIO e le comunicazioni server-frontend.
In caso di malfunzionamento, la tipologia di errore viene visualizzata sullo schermo tramite l'utilizzo di una notifica toast, la quale informa subito l'utente del tentativo d'accesso fallito.

ANALISI COMPORTAMENTO ANDROID



Il sottosistema MOBILE è costituito da un dispositivo Android in grado di comunicare via Bluetooth con l'Arduino. Il suo scopo principale è quello di interagire con la DOOR, controllando la temperatura e impostando il livello di luminosità del LED collegato al microcontrollore.
L'aspetto cruciale del sottosistema è rappresentato dalla gestione della comunicazione, la quale deve essere mantenuta tramite l'utilizzo di un AsyncTask che avvia un Thread in background in grado di inviare e ricevere messaggi con il microcontrollore che sta dall'altro lato della connessione. In quanto il sensore Bluetooth dell'Arduino è una porta seriale, l'UUID da utilizzare è quello predefinito: 00001101-0000-1000-8000-00805F9B34FB.
Il problema principale è stato l'effettuare modifiche alla View dell'applicazione da Thread non principali (in questo caso il ConnectionManager): l'utilizzo di Handler è stato la soluzione a questa complicazione, permettendo così di svolgere qualsiasi azione necessaria a seconda dei diversi messaggi ricevuto dall'Arduino.