ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

TESI DI LAUREA

in
Fondamenti di Informatica T2

AUTO-DETECT DEI DISPOSITIVI E CONFIGURAZIONE DINAMICA IN AMBIENTE HOME MANAGER

CANDIDATO
Mattia Fucili

RELATORE:
Chiar.mo Prof. Enrico Denti

CORRELATORE/CORRELATORI
Dott. Ing. Roberta Calegari

Anno Accademico 2015/16

Sessione II

Indice

Introduzione

La domotica[1] nasce dopo la terza rivoluzione industriale con i principali obiettivi di cercare di migliorare la qualità della vita di chi abita nella casa, migliorare la sicurezza e allo stesso tempo di ridurre i costi di gestione.

Il termine domotica è l'unione delle parole casa (in latino "domus") e robotica; ed è appunto la scienza che si occupa di rendere intelligenti apparecchiature, impianti e sistemi. Ad esempio una "casa intelligente" è una casa ben progettata che può essere controllata dall'utilizzatore attraverso opportune interfacce che realizzano il ponte di comunicazione tra umano e sistema intelligente. Oggi esistono vari esempi di apparecchiature intelligenti basti pensare ai nuovi televisori che possono essere collegati alla rete e anche "comandati" da remoto tramite app dal telefono, lavatrici, frigoriferi, climatizzatori, …

Home Manager è un primo prototipo di sistema per il controllo di una casa intelligente. È una applicazione che si basa su una struttura multi-agente, si appoggia sull'infrastruttura TuCSoN e Butlers.

I principali obiettivi dell'applicazione sono quelli di gestire in maniera "smart" ogni risorsa della casa, permettendo agli utenti di configurare le azioni che vengono fatte automaticamente dal sistema, e semplificare, dove possibile, la vita all'utilizzatore, come ad esempio inviando l'ordine della spesa al supermercato.

L'obiettivo che ci si pone in questa tesi è il seguente:

- aumentare l'intelligenza dell'applicazione implementando un sistema di autodetect dei dispositivi fisici, affiancando ad essi una versione simulata con la quale l'utente, in caso di assenza del dispositivo fisico, può interagire.

La tesi sarà quindi strutturata come segue:

- Nel primo capitolo verrà esposto a grandi linee il funzionamento generale del prototipo di Home Manager e delle sue caratteristiche principali.

- Nel secondo capitolo verrà analizzato il problema che pone la tesi e le varie possibili soluzioni adottabili.
- Nei tre capitoli centrali verrà fatta l'analisi della scelta implementativa adottata per poi passare alle scelte progettuali ed infine all'implementazione e collaudo.
- Chiudono la tesi i capitoli delle conclusioni e dei ringraziamenti.

Capitolo 1

Home Manager

In questo capitolo si parlerà in generale dell'applicativo Home Manager, delle sue funzionalità e di come è strutturato.

1.1 Home Manager

Home Manager[2] è un prototipo di software (sviluppato in Java) per gestire una "casa intelligente" il cui scopo è quello di supportare l'utente nella gestione della casa permettendogli di poter interagire con gli elettrodomestici attraverso comandi diretti. Esso è organizzato con agenti, ognuno dedicato ad uno specifico obiettivo, che gestiscono ogni apparecchiatura della casa e inoltre rendono il sistema in grado di anticipare i bisogni degli utenti in base alle conoscenze acquisite su di essi.

1.1.1 Il prototipo attuale

Nell'attuale prototipo si ipotizza di usare vari tipi di sensori: da quello di impronta digitale per l'autenticazione al sistema a sensori per il controllo della temperatura e sensori per determinare chi entra ed esce dalle stanze. Ad ogni utente è assegnato un ruolo che può essere abitante della casa o ospite e vari livelli di amministrazione, admin oppure semplice user. Ogni utente normale può interagire con le varie apparecchiature ed esprimere le proprie preferenze, mentre gli admin hanno il potere di configurare il sistema e di agire sui privilegi dei vari utenti, questo perché Home manager è altamente configurabile.

L'architettura Butlers[3] (maggiordomo) è un framework organizzato a 7 livelli ognuno dei quali rende "più intelligente" il sistema permettendo ad esempio di conoscere la posizione dell'utente attraverso strumenti di localizzazione, età, abitudini e preferenze anche mediante all'utilizzo dei social network. I vari Butlers interagiscono tra loro e osservano i comportamenti degli utenti e grazie a questa tecnologia Home Manager è in grado di anticipare le decisioni degli utenti sfruttando le informazioni ottenute in precedenza.

Home Manager simula una abitazione che comprende 4 locali: l'ingresso, la cucina con anche la sala da pranzo, il bagno e una camera da letto. In ognuna di questa stanze vi sono vari elettrodomestici e dispositivi quali ad esempio il forno e il frigorifero nella cucina, la televisione in camera da letto, la lavatrice nel bagno e altri dispositivi. Inoltre ci sono vari sensori per la misurazione della temperatura, per controllare la posizione dell'utente e per regolare l'impianto luminoso.

Alcuni dispositivi, per la precisione il forno e il frigorifero, hanno anche una versione fisica su Raspberry, il quale simula, come ad esempio parlando del frigorifero, l'interazione con un vero e proprio dispositivo fisico permettendo di inserire e togliere ingredienti dal suo interno tramite tag Rfid e display LCD per mostrare il tutto.

1.2 TuCSoN

In questo capitolo si farà una panoramica sulla piattaforma TuCSoN e le sue principali API.

1.2.1 Introduzione a TuCSoN

TuCSoN[4] (Tuple Centre Spread over the Network) è un'infrastruttura che fornisce servizi per la comunicazione e coordinazione di agenti, intesi come entità computazionali autonome e che interagiscono con altri agenti attraverso un sistema di comunicazione.

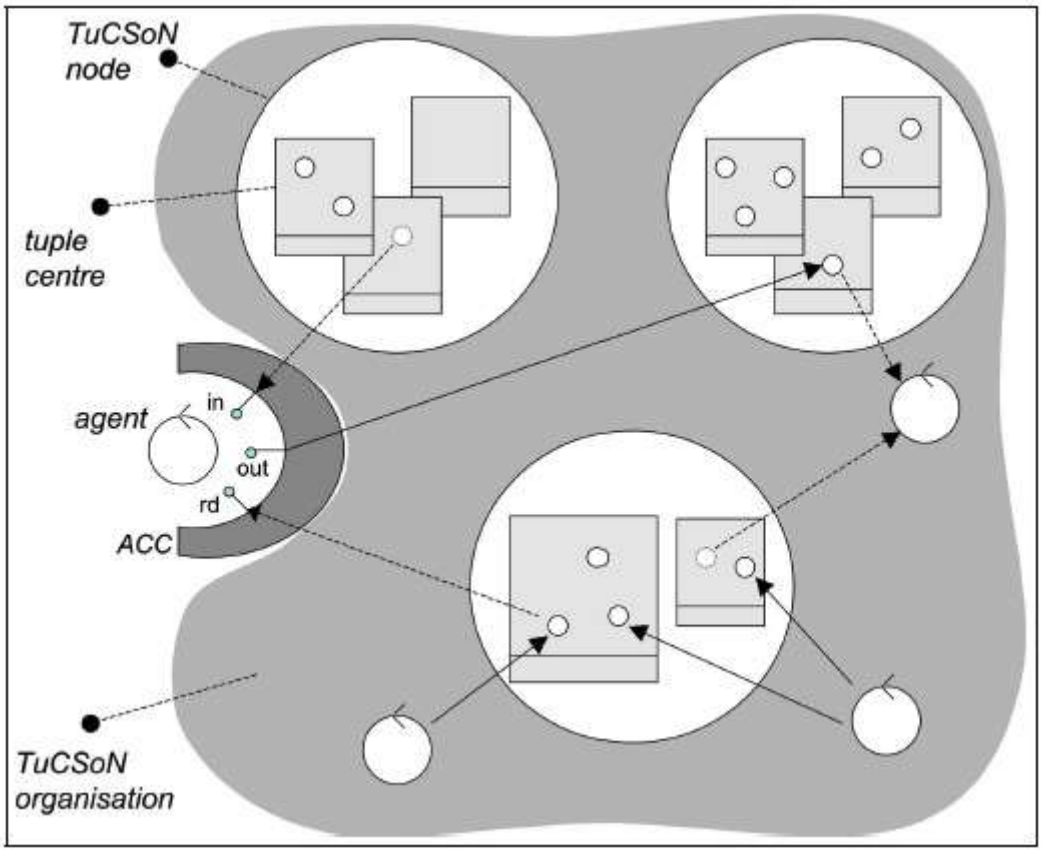
Gli agenti hanno il ruolo di entità attive, i quali possono avere un obiettivo da perseguire, sono intelligenti e sono le entità che sfruttano i centri di tuple.

La comunicazione avviene attraverso centri di tuple, ovvero spazi dove condividere informazioni comuni, i quali si trovano distribuiti nei nodi che compongono l'infrastruttura. I centri possono essere acceduti in due modi:

- come risorse globali, quindi specificando il nome del centro e l'indirizzo IP più la porta;
- oppure come risorse locali dove il centro è referenziato semplicemente con il nome.

I centri di tuple a loro volta possono essere programmati dinamicamente per essere reattivi all'inserimento, lettura o cancellazione di determinate tuple, ottenendo così un comportamento personalizzato del centro di tuple tutto grazie al linguaggio ReSpecT[5].

Per comunicare TuCSoN mette a disposizione un linguaggio, costituito da una serie di primitive, che permette di leggere, scrivere e cancellare tuple da un centro di tuple.



Come si vede dall'immagine ogni agente ha bisogno di un ACC, questa è un'interfaccia che permette all'agente di interagire con qualsiasi centro di tuple nel quale deve fare delle operazioni.

1.2.2 Operazioni possibili con TuCSoN

Ogni entità in TuCSoN è identificata in modo univoco con un id per essere indirizzata senza ambiguità in ogni momento. I centri di tuple sono identificati da 3 campi che devono essere univoci all'interno dello stesso nodo: la coppia indirizzo IP e porta e il nome del centro di tuple, gli agenti,

invece, sono identificati da un nome univoco e da un numero (Universally Unique IDentifier) che gli viene assegnato alla prima istanziazione dell'agente. Si ricorda che gli agenti vivono ovunque nella rete.

Per comunicare in TuCSoN vengono messe a disposizione delle primitive, le principali sono:

- `out`, scrive la tupla nel centro di tuple;
- `in`, elimina una tupla leggendola;
- `rd`, legge un tupla lasciandola però nel centro di tuple.

Inoltre vengono spesso utilizzate per comodità altre primitive:

- `inp`, elimina la tupla dal centro di tuple, se non c'è match lancia eccezione;
- `get`, restituisce tutte le tuple del centro di tuple specificato;
- `rdp`, legge la tupla specificata, se non c'è match lancia eccezione;
- `rd_all`, legge tutte le tuple dello specifico template passato;
- `in_all`, elimina tutte le tuple in un centro di tuple che fanno match con il template specificato;
- `out_all`, scrive tutte le tuple le tuple specificate;

1.3 RaspberryPI2

Home Manager ha anche una sua versione implementata su tecnologia Raspberry[6], più precisamente con la versione RaspberyyPi2.

Il Raspberry è un computer implementato su una singola scheda elettronica sviluppato nel Regno Unito dalla Raspberry Pi Foundation.

Il processore montato su questa versione è un ARM1176JZF-S a 700 MHz, una GPU VideoCore IV, e da 256 MB fino a 1GB di RAM. Inoltre nel progetto del Raspberry non vi è alcun hard disk come dispositivo di memoria non volatile ma, tutto viene affidato a una scheda SD. Inoltre ci sono varie porte USB che variano in numero da modello a modello, un ingresso per il cavo Ethernet, connettori per video camera, audio, uscita HDMI e una serie di pin utilizzati per connettere i vari sensori compatibili con il Raspberry ad esempio sfruttando GrovePi.

GrovePi è una scheda elettronica che fa da interfaccia tra i vari sensori che possono esservi installati ed il Raspberry, permettendo così di programmarli ed usarli a proprio piacimento.

I sistemi operativi che la scheda accetta sono quelli basati su kernel Linux o RISC OS, come ad esempio Raspbian, il sistema operativo ufficialmente supportato dalla Fondazione e quindi anche il più diffuso nel mondo.

Capitolo 2

Autodetect in Home Manager

Questo capitolo illustra il problema centrale della tesi e come viene pensato di risolverlo mediante due possibili strade.

2.1 Il problema

La versione attuale di Home Manager essendo un prototipo in fase di sviluppo manca ancora di alcune funzionalità utili. Una di queste, ad esempio, è quella dell'autodetect. Questa funzionalità permetterebbe al sistema di accorgersi se un nuovo dispositivo viene inserito all'interno dell'abitazione e, quindi, deve entrare a far parte di tutti quelle apparecchiature che il sistema deve monitorare o con le quali l'utente vuole o deve interagire.

2.2 Possibili approcci

Nella versione attuale ogni dispositivo che fa parte della casa simulata gestita da Home Manager è cablato nel codice stesso del programma. Questo è un problema perché per ogni nuovo dispositivo si dovranno cambiare molte righe di codice per cercare di adattare il tutto.

Qui di seguito verranno esposti due possibili approcci per la soluzione di questo problema, uno dei quali verrà scelto come linea guida per lo sviluppo della tesi e quindi sarà implementato in Home Manager.

2.2.1 Autodetect all'avvio del sistema

Una prima semplice soluzione pensata prevede la modifica del programma per fare in modo che all'avvio di Home Manager il sistema venga informato dei dispositivi al momento inseriti nella casa.

2.2.1.1 Problematiche di questa soluzione

Questa soluzione però si avvale ancora in parte della struttura attuale di Home Manager quindi il codice in alcune parti rimarrà lo stesso.

Il problema più grande però di questa soluzione ovviamente è che il sistema in questo modo sarà insensibile a nuovi dispositivi inseriti dopo l'avvio dell'applicazione, quindi si guadagna ben poco da questo tipo di approccio rispetto alla versione precedente.

Più intelligente sarà la soluzione che verrà presentata nel paragrafo sottostante.

2.2.2. Autodetect dinamico

Cosa logicamente più sensata è avere un sistema che si accorga di nuovi dispositivi inseriti al suo interno ogni volta che avviene un cambiamento. Questa soluzione dinamica di individuare nuovi dispositivi sarà la soluzione guida per lo sviluppo della tesi e verrà trattata nei capitoli che seguono.

2.2.2.1 Problematiche di questa soluzione

Avendo in Home Manager dispositivi rappresentati anche fisicamente tramite Raspberry un problema che sorge per l'autodetect è quello dell'assegnazione del nome del dispositivo che si collega al sistema.

Per ovviare a questo problema si possono intraprendere varie strade:

1. Per prima cosa sarebbe possibile assegnare un nome fisso al dispositivo alla sua prima installazione su Raspberry, ma questo comporta a dover assegnare ogni volta il nome per ogni nuovo dispositivo rendendolo un processo troppo manuale dal punto di vista dell'informatica. Il problema principale però è quello dell'assegnazione di un nome che potrebbe essere già appartenente ad un altro dispositivo fisico, perché nessuno lo vieta e quindi, sicuramente, si creeranno dei problemi di conflitto causa nomi identici.
2. Un'altra soluzione possibile sarebbe quella di utilizzare l'indirizzo MAC del Raspberry in quanto codice univoco e quindi può identificare in modo unico un dispositivo nella rete di Home Manager (escludendo il caso di MAC spoofing). Questa soluzione sembra molto ragionevole se non fosse che nello stesso Raspberry si possono simulare più dispositivi, quindi ad esempio posso avere sia

il forno che il frigorifero nello stesso Raspberry. In questo caso entrambi hanno lo stesso indirizzo MAC e quindi si hanno di nuovo problemi di conflitto.
Inoltre è possibile spostare il progetto che simula il dispositivo su un altro Raspberry e quindi il nome assegnato cambia con il nuovo MAC del Raspberry ospitante.

3. Ultima possibilità è quella di sfruttare i centri di tuple, in quanto Home Manager è basato su di essi. Quindi ad esempio si può adottare un protocollo che prevede l'inserimento di una tupla in un centro di tuple specializzato da parte del dispositivo che è stato appena aggiunto al sistema la quale verrà letta da chi di dovere per poi rispondere con il nome che verrà assegnato al nuovo dispositivo. Infine quest'ultimo deve leggere il suo nome e d'ora in avanti avrà sempre lo stesso nome.

I nomi ovviamente dovranno essere assegnati con un criterio che sarà poi spiegato in fase di progettazione.

Una volta assegnato il nome però si dovrà pensare ad una soluzione che lo renda persistente e soprattutto si deve pensare ad un modo per tenere traccia di tutti i dispositivi registrati nel sistema, per evitare in futuro di assegnare lo stesso nome ad un diverso dispositivo.

Capitolo 3

Autodetect dinamico: analisi

In questo capitolo verrà trattata la fase di analisi del problema della tesi

3.1 Obiettivi

L'obiettivo principale di questa tesi è quello di estendere il prototipo di Home Manager per renderlo sensibile all'inserimento di nuovi dispositivi nell'abitazione. Questo vuol dire che si dovrà gestire l'interazione tra Home Manager e il dispositivo stesso, indipendentemente dalla sua tipologia, assegnandogli un nome univoco all'interno del sistema. Inoltre occorrerà assegnargli un agente per il monitoraggio e la gestione, per permettere anche all'utente di interagire con esso e mostrare il suo stato.

3.2 Analisi del problema

Attualmente il sistema non ha questa funzionalità, infatti tutti gli agenti che monitorano e gestiscono i vari dispositivi presenti in Home Manager vengono creati in base ad informazioni note in quanto scritte a mano nel codice. Questo è un grosso problema ovviamente perché aggiungendo un nuovo dispositivo occorre modificare varie classi del progetto per renderlo allineato con la nuova soluzione, inoltre scambiando il semplice ordine all'interno del vettore contenente la lista di centri di tuple del sistema non funziona più niente.

Per ovviare a questi problemi sarà necessaria una modifica sostanziale del progetto per rendere Home Manager capace di interagire con qualsiasi dispositivo creando agenti indipendenti da quest'ultimo. Per di più verrà sfruttato maggiormente TuCSoN e i centri di tuple per avere una soluzione più intelligente e consistente.

Questa tesi verrà sviluppata seguendo la linea guida della soluzione del paragrafo Autodetect dinamico con assegnazione del nome attraverso lo scambio di tuple.

Capitolo 4

Autodetect dinamico: progettazione e implementazione

In questo capitolo verranno illustrate le scelte progettuali adottate in fase di sviluppo del codice e la sua implementazione.

4.1 Scelte progettuali

Per realizzare il progetto descritto precedentemente è necessario avere un centro di tuple specifico per questo compito e che sia quindi solo utilizzato per scambiare tuple richiedenti il nome del nuovo dispositivo che si è connesso e per il salvataggio di tutti i dispositivi connessi nella casa. Verrà inoltre creato anche un nuovo agente, `DeviceManagerAgent`, per gestire questa prima interazione tra Home Manager e i dispositivi appena collegati sfruttando proprio questo nuovo centro di tuple.
L'agente dopo essersi accorto dell'arrivo di una nuova richiesta dovrà leggerla ed in base al dispositivo che si è connesso dovrà creare il suo nome univoco all'interno del sistema e inviargli una risposta contenente quest'ultimo.

Una volta inviata la risposta l'agente dovrà salvarsi sul centro di tuple la presenza del nuovo dispositivo sia in versione fisica che simulata. La versione simulata sarà sempre presente perché, se per qualche motivo il dispositivo fisico viene a mancare, l'utente riceve la notifica dell'accaduto, ma può continuare a lavorare lo stesso con la versione simulata. Inoltre è utile avere, per tutti i tipo di dispositivi, una versione solamente simulata, ad esempio frigorifero_0, che in caso di nessun dispositivo fisico presente nell'abitazione permetta comunque all'utente di simularne il funzionamento.
Si deve inoltre salvare in maniera persistente anche tutta la lista di dispositivi conosciuti per poterla poi caricare all'avvio del sistema in caso di spegnimento. Questa lista verrà salvata in un file con lo stesso formato delle tuple; in questo modo sarà più veloce il caricamento durante l'inizializzazione del sistema. TuCSoN, infatti, permette l'inserimento di tuple anche partendo da una stringa ben formata.

Per la precisione nel file saranno presenti tante tuple del tipo: `device(name_device, info_device(0,null,-1), s);` oppure con ultimo campo a `p` in base a se il dispositivo è fisico o simulato. Lo stesso dispositivo fisico dovrà salvarsi il suo nome e anche qui si è scelto di salvarlo in un file per comodità.

Il DeviceManagerAgent inoltre, dopo aver aggiornato il suo stato ed aver inviato la risposta al dispositivo, è tenuto a notificare a tutti i componenti dell'applicazione, interessati all'arrivo di un nuovo dispositivo, tale avvenimento. Questo meccanismo lo si ottiene utilizzando una sorta di evento implementato grazie ad un'interfaccia, `DeviceListener`, che tutti i componenti interessati andranno ad implementare. L'interfaccia espone un metodo `OnNewDevice(Device device)` il quale sarà invocato dal DeviceManagerAgent ogni volta che arriva un nuovo dispositivo, il quale verrà passato come argomento. Gli ascoltatori di questo evento sono il pannello principale dell'applicazione e il pannello che contiene la lista di dispositivi presenti nell'abitazione.

La creazione degli agenti che monitorano i dispositivi è delegata, per il principio delle singole responsabilità, ad un altro componente chiamato `AgentManager`. Questa classe gestisce tutti gli altri agenti, tenendoli salvati in una lista interna, e permette, ai pannelli interessati dell'arrivo di nuovi dispositivi, per prima cosa di inizializzarsi ottenendo la lista di dispositivi e poi di aggiungere i nuovi agenti dei nuovi dispositivi. Questa creazione di agenti a sua volta si avvale di una Factory, chiamata `AgentsFactory`, che ricevuti determinati parametri di input permette di istanziare qualsiasi tipo di nuovo agente necessario, a patto che esista questo agente all'interno del sistema.

Tutte le classi riguardanti l'autodetect, in quanto tutte accomunate tra loro, sono state create all'interno di un nuovo package chiamato: `it.unibo.homemanager.detection`.

Un'ulteriore modifica apportata ad Home Manager prevede l'aggiunta di un'interfaccia di comunicazione per tutte le tuple presenti. Questa nuova classe verrà creata in un altro package: `it.unibo.homemanager.communication`. Nello specifico la classe

sarà una classe astratta contenente tante stringhe che rappresentano tutti i template possibili nelle tuple finora usate e tutti i metodi getter per accedervi. Il codice della classe è riportato qua sotto:

```
package it.unibo.homemanager.communication;

public abstract class AbstractCommunication {

    private static final String new_name = "new_name";
    private static final String info_device = "info_device";
    private static final String device = "device";

    public static String getNewName() {
        return new_name;
    }

    public static String getInfoDevice() {
        return info_device;
    }

    public static String getDevice() {
        return device;
    }
}
```

Questa classe verrà ampliata in futuro con tutte le nuove tuple che saranno inventate, in modo tale che, se per qualche motivo si desidera cambiare il nome di un template, non occorre aggiornare tutte le classi interessate ma semplicemente cambiare la stringa desiderata in questa classe.

4.2 La comunicazione tra i dispositivi e Home Manager

L'interazione tra i dispositivi e Home Manager all'inizio avverrà tramite il centro di tuple creato ad hoc per questo scopo: `device_manager_tc`. Per prima cosa ogni volta che un nuovo dispositivo si vuole connettere al sistema deve scrivere nel centro di tuple una tupla del tipo:

`new_name(unknown, myType, info_device(0,null,-1));`
dove:

- `unknown` serve per dire all'agente che il dispositivo non ha un nome ancora;
- `myType` si riferisce al tipo del dispositivo che fa la richiesta (ad esempio frigorifero);

- `info_device` non è utilizzato ma potrebbe essere necessario in futuro; i suoi valori sono stati scelti in questo modo perché per realizzare il progetto si sfrutta una classe (`Device`) già esistente, ma che viene modificata, che rappresenta un dispositivo. All'interno della classe tra i vari attributi vi è anche un campo relativo al consumo in watt del dispositivo, al quale è stato assegnato `0`, un attributo di tipo stringa per descrivere eventuali parametri, al quale è stato assegnato il valore `null` ed infine un campo per indicare il codice della stanza in cui si trova, al quale è stato assegnato `-1`.

Una volta letta la tupla il DeviceManagerAgent sa di che tipo è il dispositivo di cui si deve occupare e deve semplicemente creare il nome del nuovo dispositivo associando il suo tipo e un numero progressivo che viene aumentato di 1 ogni volta che arriva uno stesso dispositivo. Nel primo caso in assoluto, avendo per ogni tipologia di dispositivo una versione "0" simulata, il nome sarà, nell'esempio in cui arrivi un frigorifero, "frigorifero(1)". Dopo aver creato il nome l'agente invia una risposta, sempre sotto forma di tupla destinata allo specifico dispositivo. Questa tupla di risposta avrà la forma:

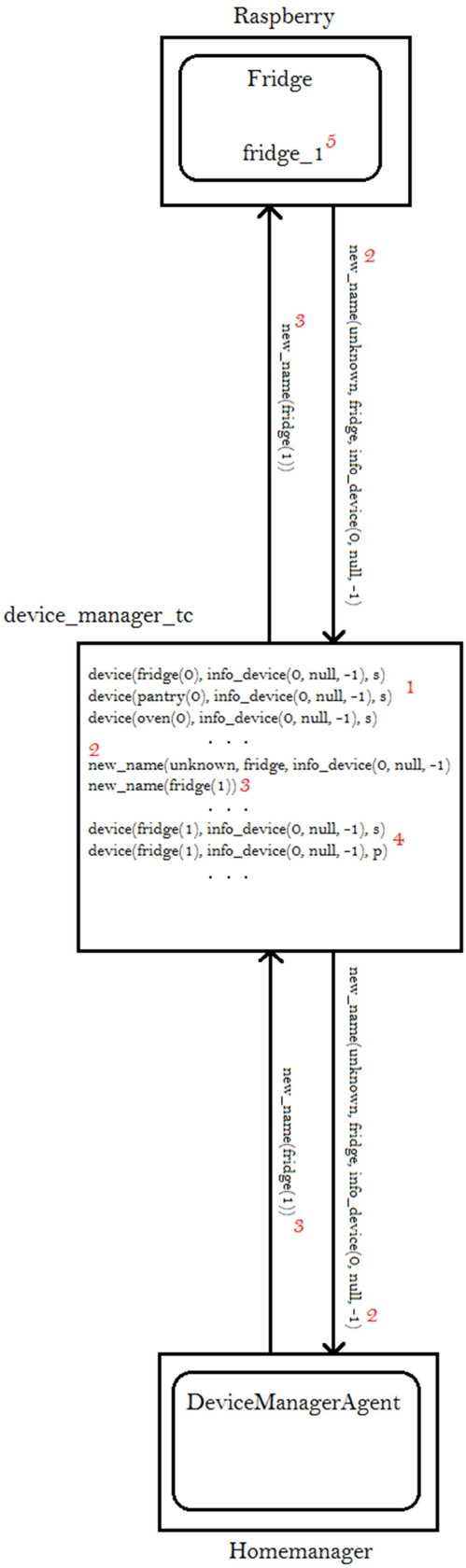
`new_name(name_dex\vice);` dove:

- `name_device` è il nome univoco che gli verrà assegnato, collegandoci all'esempio di prima sarà: frigorifero(1).

In ultimo il dispositivo andrà a leggere la tupla di risposta e si salverà in modo persistente il suo nome su file.

Questo protocollo viene adottato per qualsiasi nuovo dispositivo e prevede che il DeviceManagerAgent controlli la presenza di nuove richieste di "ingresso al sistema" ogni secondo dalla sua accensione.

4.3 Diagramma del protocollo



4.4 Diagramma delle classi

Diagramma della classe DeviceManagerAgent

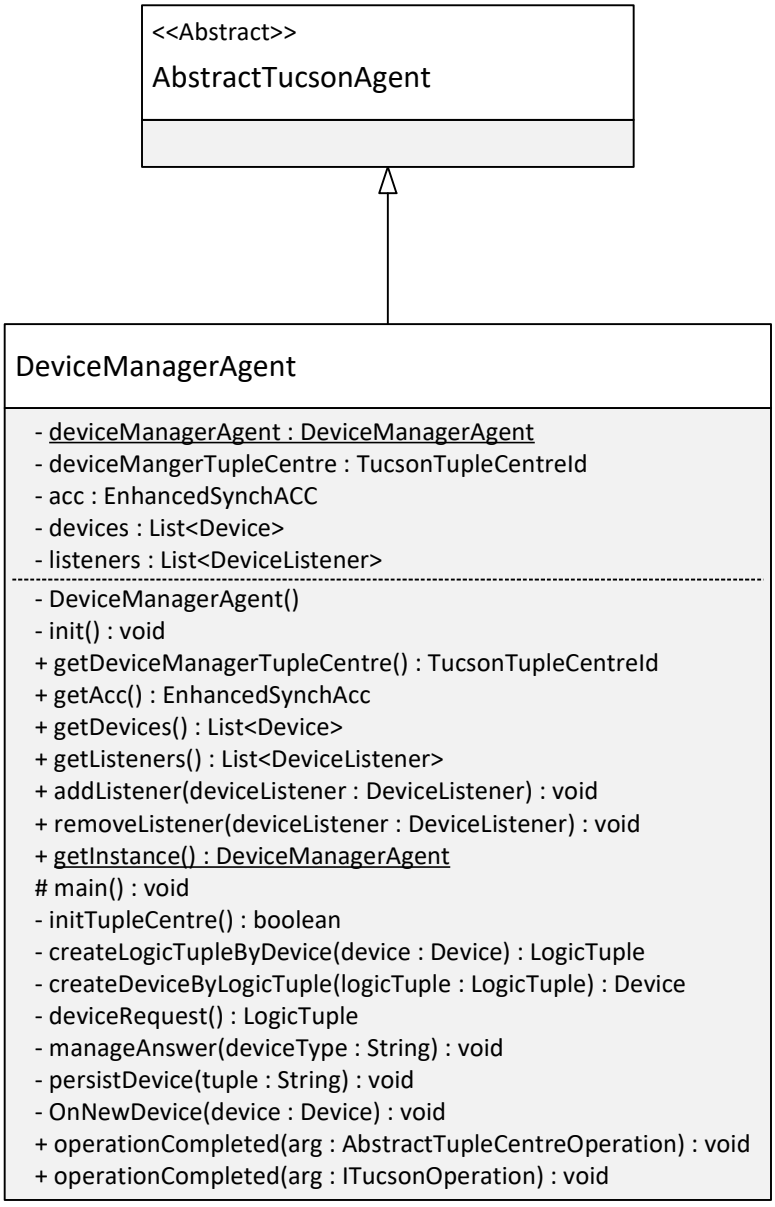
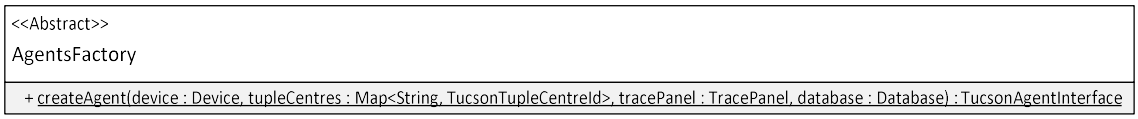


Diagramma della classe AgentsFactory



4.5 Gestore di nuovi dispositivi: DeviceManagerAgent

Il DeviceManagerAgent è implementato con il pattern singleton ed una volta fatto partire l'agente nel suo metodo `main()` compie principalmente due operazioni fondamentali:

- la prima, consiste nel chiamare un metodo per controllare se nel suo centro di tuple è stata depositata una nuova richiesta di ricevere un nome, invocando il metodo `deviceRequest()` riportato qua sotto

```
private LogicTuple deviceRequest() throws Exception {
    String tuple = AbstractCommunication.getNewName() + "(unknown,A,B)";
    LogicTuple logicTuple = LogicTuple.parse(tuple);
    ITucsonOperation operation = getAcc().inp(getDeviceManagerTupleCentre(), logicTuple, Long.MAX_VALUE);
    if(operation.isResultFailure())
        return null;
    return operation.getLogicTupleResult();
}
```

  dove si può vedere che si esegue una operazione di *in*, quindi leggiamo cancellando la tupla dal centro di tuple;

- la seconda operazione, invece, serve per gestire la risposta da inviare al dispositivo che ha fatto la richiesta. In questo caso il metodo invocato è `manageAnswer()` riportato qua sotto

```
private void manageAnswer(String deviceType) throws Exception {
    int index = 0;
    for(Device device : getDevices())
        if((device.getDeviceName().equals(deviceType)) && (device.getDeviceId() > index))
            index = device.getDeviceId();
    index = index + 1;

    String tuple = AbstractCommunication.getNewName() + "(" + deviceType + "(" + index + "))";
    LogicTuple logicTuple = LogicTuple.parse(tuple);
    ITucsonOperation operation = getAcc().out(getDeviceManagerTupleCentre(), logicTuple, Long.MAX_VALUE);
```

  il quale si occuperà di assegnare un nuovo nome, associando il tipo di device che ha fatto la richiesta ed un numero progressivo che tiene conto di tutti i dispositivi di quel tipo presenti nel sistema. Infine andrà a scrivere la tupla di presenza del dispositivo sia fisico che simulato e notificherà ai componenti interessati l'arrivo di un nuovo dispositivo, come si può vedere dal codice sottostante:

```
// device(deviceType(index), info_device(watt,null,-1),s)
tuple = AbstractCommunication.getDevice() + "(" + deviceType + "(" + index + ")," + AbstractCommunication.getInfoDevice() + "(" + watt + ",null,-1),s)";
logicTuple = LogicTuple.parse(tuple);
getAcc().out(getDeviceManagerTupleCentre(), logicTuple, Long.MAX_VALUE);
Device first = createDeviceByLogicTuple(logicTuple);
getDevices().add(first);
persistDevice(tuple);
// ---

// device(deviceType(index), info_device(watt,null,-1),p)
tuple = AbstractCommunication.getDevice() + "(" + deviceType + "(" + index + ")," + AbstractCommunication.getInfoDevice() + "(" + watt + ",null,-1),p)";
logicTuple = LogicTuple.parse(tuple);
getAcc().out(getDeviceManagerTupleCentre(), logicTuple, Long.MAX_VALUE);
Device second = createDeviceByLogicTuple(logicTuple);
getDevices().add(second);
persistDevice(tuple);
// ---

onNewDevice(first);
onNewDevice(second);
```

Questo agente esegue in background ed è sempre in esecuzione non appena viene fatto partire Home Manager cosicché se un dispositivo viene aggiunto al sistema in un qualsiasi momento questo agente, essendo sempre presente, si accorge e permette il suo ingresso al sistema.

4.6 Factory per la creazione di agenti: AgentsFactory

L'AgentsFactory è una classe astratta che ha solamente un metodo statico che permette appunto la creazione di qualsiasi agente per il monitoraggio di un dispositivo fornendo determinati parametri tra i quali troviamo, il device che deve monitorare, tutti i centri di tuple di cui hanno bisogno per comunicare e i pannelli utili per questioni di grafica. Il metodo è implementato in questo modo:

```
switch(device.getDeviceName()) {
    case "fridge":
        try {
            FridgeAgent agent = new FridgeAgent(device, tracePanel, tupleCentres.get("fridge_tc"), database);
            agent.setPanel(butlerPanel);
            return agent;
        } catch(TucsonInvalidAgentIdException e) {
            e.printStackTrace();
        }
    case "pantry":
        try {
            PantryAgent agent = new PantryAgent(device, tracePanel, tupleCentres.get("pantry_tc"), database);
            agent.setPanel(butlerPanel);
            return agent;
        } catch(TucsonInvalidAgentIdException e) {
            e.printStackTrace();
        }
    case "oven":
        try {
            OvenAgent agent = new OvenAgent(device, tracePanel, tupleCentres.get("oven_tc"), database);
            agent.setPanel(butlerPanel);
            return agent;
        } catch(TucsonInvalidAgentIdException e) {
            e.printStackTrace();
        }
    default:
        return null;
}
```

4.7 Implementazione dell'agente del mixer: MixerAgent

Dopo aver realizzato quello richiesto, per collaudare il tutto sarà necessario creare un dispositivo fisico. Essendoci già uno scenario che verrà implementato in futuro in Home Manager, dove sono presenti vari dispositivi per la gestione della cucina, tra i quali il mixer, si è optato di implementare quest'ultimo e il suo agente per interagire con esso. La sua implementazione è molto semplice in quanto verrà utilizzato solo per testare il funzionamento dell'autodetect ed ampliato in futuro. Verrà aggiunto un nuovo package contenente l'agente, dove poi andranno inseriti tutti gli altri agenti che verranno implementati, il nome sarà: `it.unibo.homemanager.agents`.

Verrà implementato direttamente su Raspberry, quindi per prima cosa occorre configurare il Raspberry per poterlo usare. Innanzitutto è necessario installare Raspbian, scaricabile dal sito ufficiale del Raspberry. Dopo le varie configurazioni richieste occorre installare Eclipse e l'ultima versione della JDK tramite il comando:

`sudo apt-get install oracle-java8-jdk`.

Terminata la configurazione è possibile implementare il codice per creare il mixer.

Per gestire la richiesta di un nuovo nome è stata creata una classe apposita chiamata `NameManager`, la quale espone un metodo `getName()` che gestisce tutta l'interazione con il DeviceManagerAgent di Home Manager. In questo metodo verrà creata quindi la tupla di richiesta del nome che sarà: `new_name(unknown,mixer,info_device(0,null,-1))`, verrà aspettata la risposta ed infine restituito il nome. Il codice del metodo è il seguente:

```
public String getName() throws Exception {
    File file = new File("name.txt");
    if (!file.exists()) {
        Configuration configuration = Configuration.getInstance();

        TucsonAgentId agentId = new TucsonAgentId("naming");
        EnhancedSynchACC acc = TucsonMetaACC.getContext(agentId);

        TucsonTupleCentreId managerTc = new TucsonTupleCentreId("device_manager_tc", configuration.getIp(),
                                                                String.valueOf(configuration.getPort()));

        String tuple = "new_name(unknown" + "," + configuration.getDeviceType() + "," + "info_device"
                       + "(" + configuration.getDeviceWatt() + ")" + ")";
        LogicTuple template = LogicTuple.parse(tuple);

        ITucsonOperation operation = acc.out(managerTc, template, Long.MAX_VALUE);

        if (!operation.isResultSuccess()) {
            return (null);
        }

        tuple = "new_name" + "(" + configuration.getDeviceType() + "(" + "A" + ")" + ")";
        template = LogicTuple.parse(tuple);

        Long timeout = new Long(5000);
        try {
            operation = acc.in(managerTc, template, timeout);
        } catch (OperationTimeOutException e) {
            return (null);
        }

        if (!operation.isResultSuccess()) {
            return (null);
        }

        LogicTuple result = operation.getLogicTupleResult();

        int first = 0;
        int index = result.getArg(first).getArg(first).intValue();

        String name = configuration.getDeviceType() + "_" + index;

        file.createNewFile();
        PrintWriter writer = new PrintWriter(file);
        writer.println(name);
        writer.close();
    }

    BufferedReader reader = new BufferedReader(new FileReader(file));
    String name = reader.readLine();
    reader.close();

    return (name);
}
```

Come si può notare per creare il centro di tuple e per prendere i dati relativi al dispositivo si interagisce con un'altra classe, `Configuration`, che, come dice la parola contiene tutte le informazioni di configurazione del dispositivo. La classe è riportata qua di seguito:

```
public class Configuration {
    private static Configuration configuration;

    private String ip;
    private int port;

    private String deviceType;
    private int deviceWatt;

    private Configuration() {
        ip = "localhost";
        port = 20504;

        deviceType = "mixer";
        deviceWatt = 0;
    }

    public static Configuration getInstance() {
        if (configuration == null) {
            configuration = new Configuration();
        }
        return (configuration);
    }

    public String getIp() {
        return (ip);
    }

    public int getPort() {
        return (port);
    }

    public String getDeviceType() {
        return (deviceType);
    }

    public int getDeviceWatt() {
        return (deviceWatt);
    }
}
```

Capitolo 5

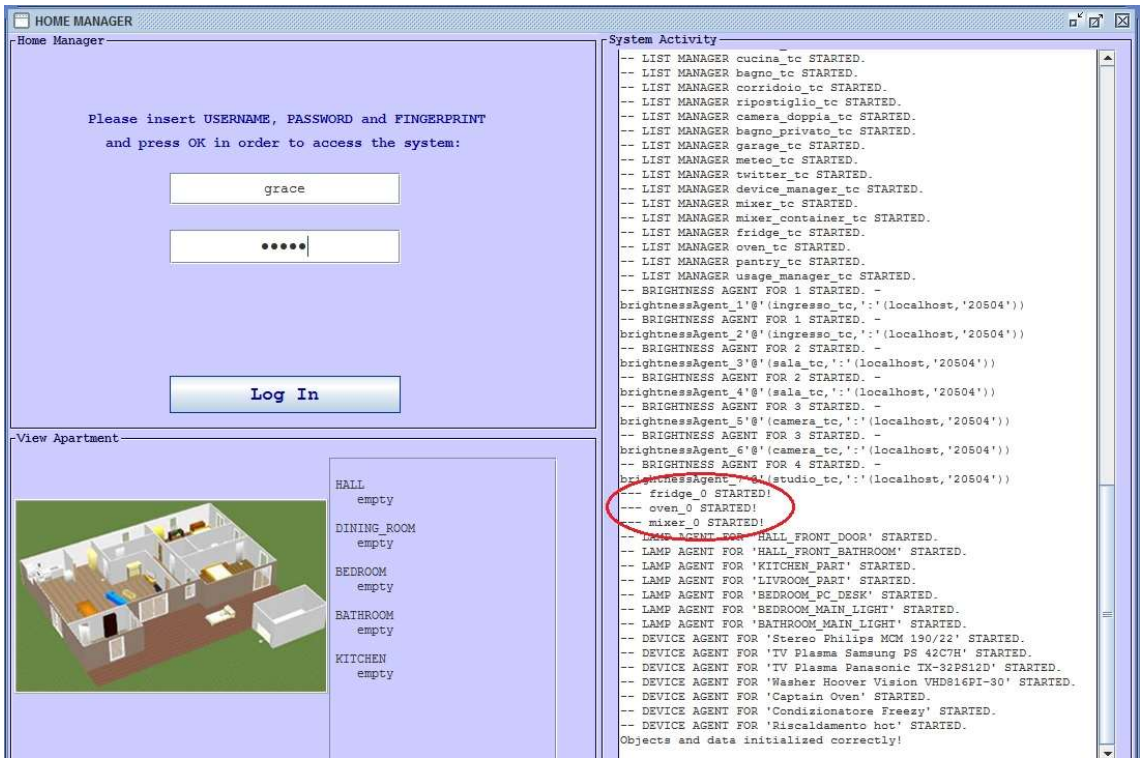
Autodetect dinamico: collaudo

Per collaudare il tutto occorre creare il .jar del progetto implementato nel Raspberry.

Una volta fatto partire TuCSoN è possibile testare il funzionamento eseguendo il .jar dal Raspberry e facendo partire Home Manager dal computer.

5.1 Inserimento di un nuovo dispositivo fisico

All'avvio di Home Manager l'applicativo avrà questo aspetto:

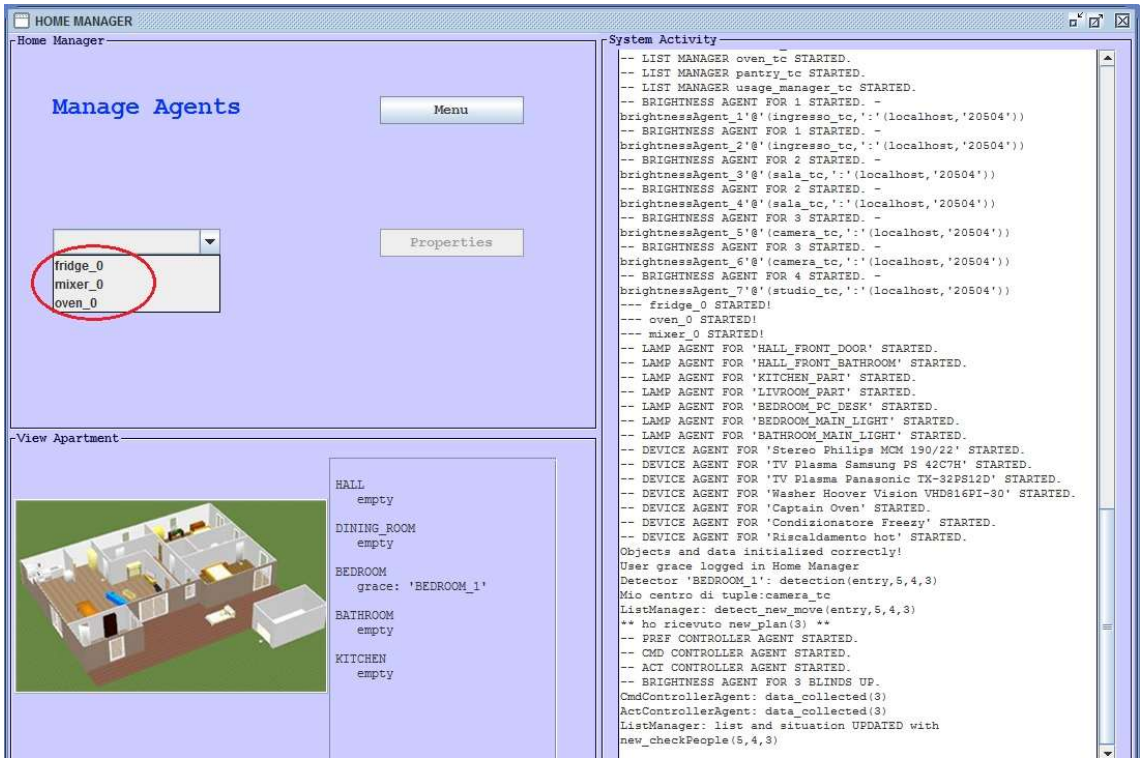


come si può vedere nel cerchio rosso l'agente DeviceManagerAgent è già partito ed ha scovato i primi dispositivi connessi alla casa. Questi sono i dispositivi di default (solo simulati) con i quali l'utente può iniziare ad interagire in attesa dei dispositivi fisici veri e propri. I dispositivi in questione sono quelli scritti nel file dove vengono salvati tutti i device della

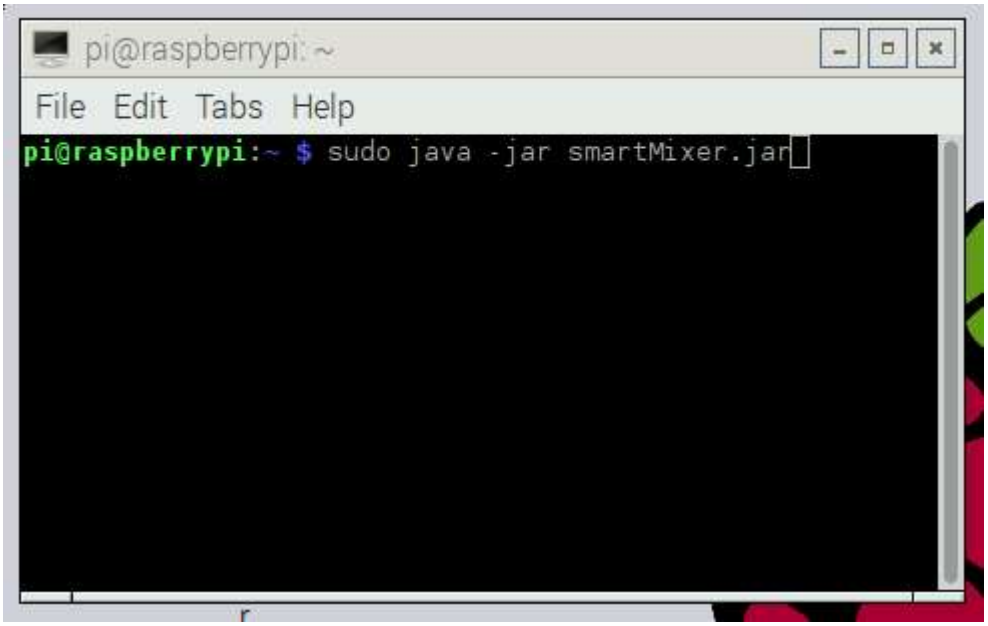
casa del quale si è parlato nelle scelte progettuali. Nello specifico il file in questione è composto dalle seguenti righe:

```
device(fridge(0),info_device(0,null,-1),s)
device(mixer(0),info_device(0,null,-1),s)
device(oven(0),info_device(0,null,-1),s)
```

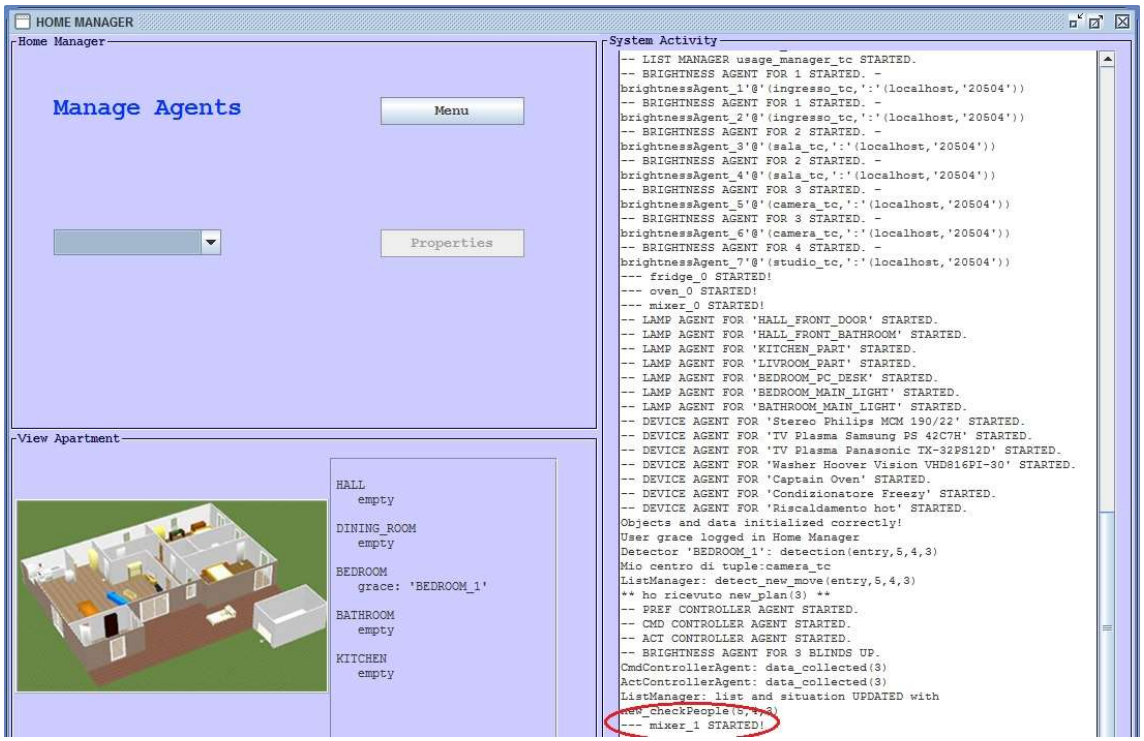
infatti se ora si va a controllare la lista degli agenti presenti si troveranno esattamente questi tre:



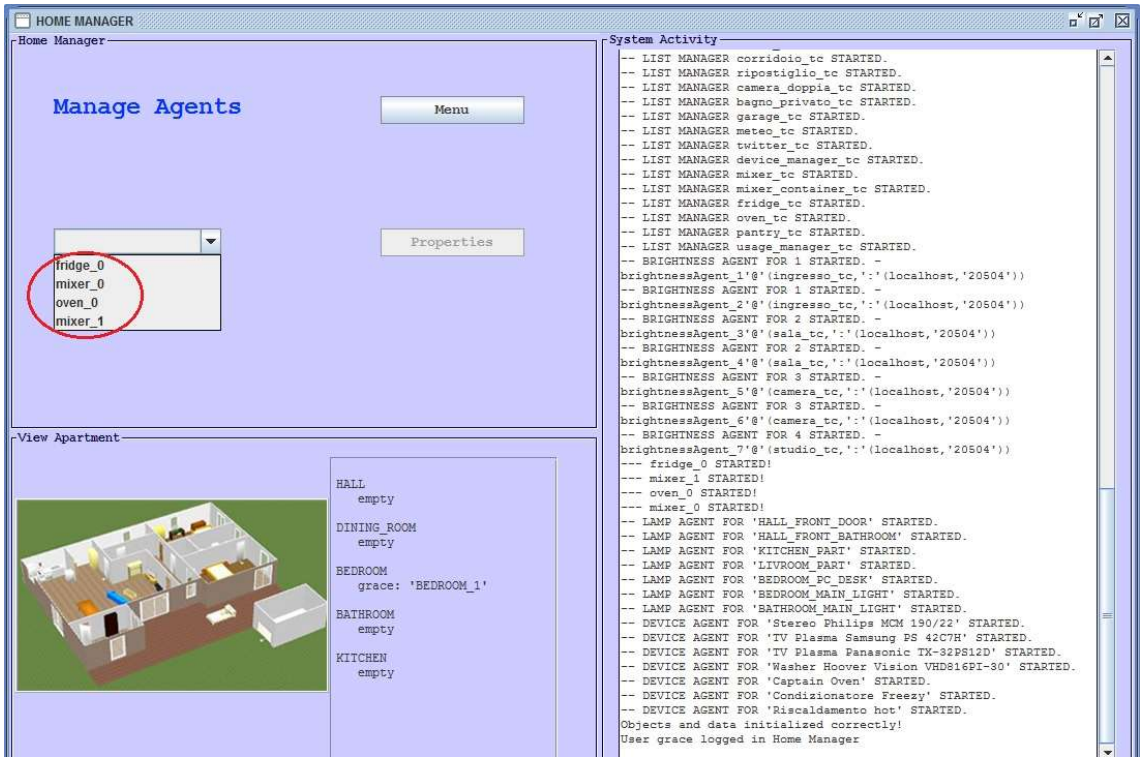
Ora si passa sul Raspberry e si esegue il progetto contenente il mixer, precedentemente estratto in un file .jar.



Come può notare ora su Home Manager è comparsa nel `System Activity` la scritta che è partito un nuovo agente chiamato, ovviamente, `mixer_1`:

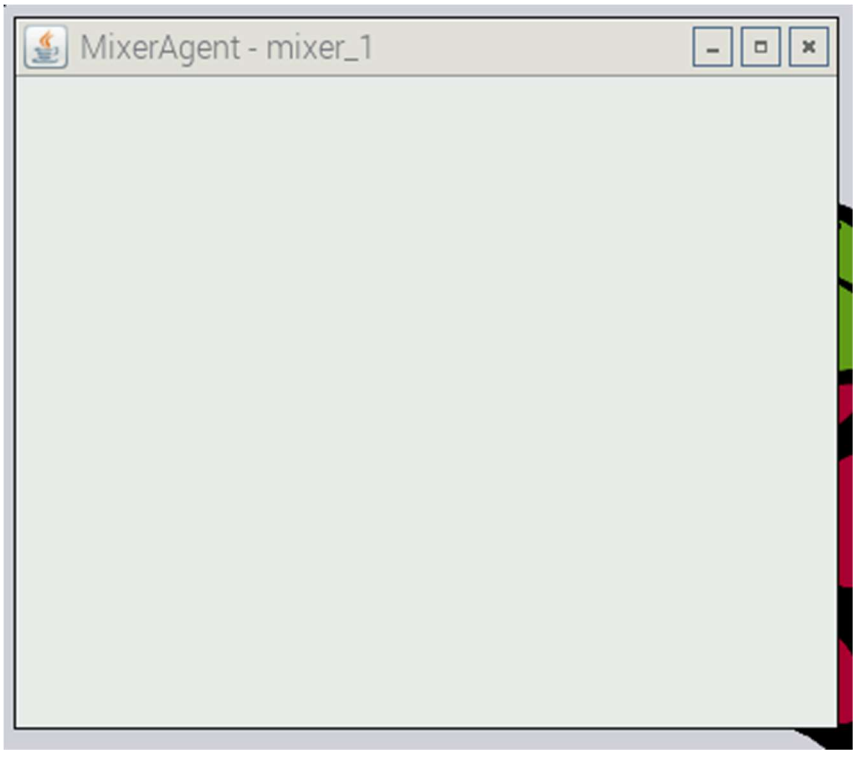


Ed infatti se si va a vedere la lista di agenti presenti si è aggiornata ed espone anche il nuovo agente mixer_1:



Questo vuole dire che il programma ha funzionato ed il dispositivo è stato aggiunto correttamente nel sistema.

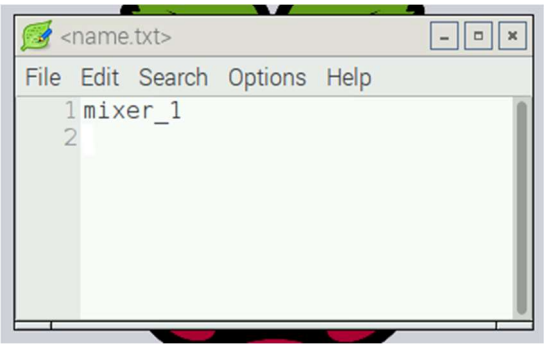
Come si può vedere nella finestra che è comparsa nello schermo del Raspberry il nome è lo stesso di quello che si ha su Home Manager:



Inoltre andando a vedere i file dove vengono salvati i dispositivi si può notare che è stato aggiunto un nuovo dispositivo, sia fisico che simulato:

```
device(fridge(0),info_device(0,null,-1),s)
device(mixer(0),info_device(0,null,-1),s)
device(oven(0),info_device(0,null,-1),s)
device(mixer(1),info_device(0,null,-1),s)
device(mixer(1),info_device(0,null,-1),p)
```

Anche nel progetto del Raspberry è stato creato il file contenente il nome che gli è stato assegnato:



Capitolo 6

Conclusioni

Questa tesi aveva come obiettivo quello di estendere il prototipo di Home Manager permettendo il riconoscimento di qualsiasi dispositivo venga collegato al sistema, introducendo un nuovo agente dedicato proprio a questo scopo.

La parte difficile della tesi consisteva nel progettare un protocollo di comunicazione standard e consistente tra i vari dispositivi ed Home Manager.

Lo studio fatto in partenza e le scelte progettuali effettuate hanno permesso di ottenere i risultati aspettati e quindi ora si può affermare che in Home Manager è presente un sistema, funzionante, di detect dei dispositivi, sia già presenti nel sistema, sia nuovi dispositivi appena inseriti.

Da questa tesi inoltre si mette in mostra l'efficacia della scelta dell'infrastruttura TuCSoN e dei centri di tuple, i quali permettono un solido sistema di comunicazione pur mantenendo i vari agenti indipendenti gli uni dagli altri.

Come possibile sviluppo futuro si potrebbe assegnare un nome diverso ai dispositivi, per evitare la semplice associazione "tipologia_numero", ad esempio si potrebbe permettere agli utenti di rinominarli una volta entrati nel sistema. Questo implica che anche il dispositivo fisico deve essere avvisato di questo cambiamento e che quindi, il protocollo necessiterà di una aggiunta di nuove tuple per scambiarsi queste informazioni.

Bibliografia

[1] Tratto da Wikipedia:

https://it.wikipedia.org/wiki/Domotica

[2] Prof. Enrico Denti e Ing. Roberta Calegari, Home Manager:

http://apice.unibo.it/xwiki/bin/view/Products/HomeManager

[3] Tratto da Enrico Denti:

"Novel pervasive scenarios for home management: the Butlers architecture"

[4] Tratto da:

http://apice.unibo.it/xwiki/bin/view/TuCSoN

"TuCSoN (Tuple Centre Spread over the Network)" Stefano Ficcadenti, Roberto Mindoli.

[5] Tratto da:

"TuCSoN (Tuple Centre Spread over the Network)" Stefano Ficcadenti, Roberto Mindoli.

[6] Tratto da Wikipedia:

https://it.wikipedia.org/wiki/Raspberry_Pi

Ringraziamenti

In questi ringraziamenti si conclude una parte del mio percorso universitario.

Ringrazio il Prof. Enrico Denti e l'Ing. Roberta Calegari per avermi dato la possibilità di lavorare allo sviluppo di una piccola parte di questo progetto e per essere stati sempre presenti, disponibili e tempestivi nel fornirmi indicazioni per la realizzazione di questa tesi.

Ringrazio la mia famiglia per avermi permesso di arrivare a questo traguardo sostenendomi moralmente, ma soprattutto economicamente.

Ringrazio i miei amici per tutte le serate passate in compagnia in questi 3 anni a Bologna.

Infine ringrazio i miei compagni Erika e Luca per l'ottimo lavoro fatto insieme nello sviluppo della macrotesi di cui la mia faceva parte.