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 d'informatica T2

SPERIMENTAZIONE DI TECNOLOGIE RASPBERRY IN CONTESTI DI HOME INTELLIGENCE

CANDIDATO
Matteo Carano

RELATORE:
Chiar.mo Prof. Enrico Denti

CORRELATORE/CORRELATORI
Dott. Ing. Roberta Calegari

Anno Accademico 2014/15

Sessione II

Indice

Introduzione

Il rapido sviluppo della tecnologia sta portando l'uomo ad impiegarla in tutto ciò che lo circonda, per semplificargli la vita di ogni giorno. Uno degli ambiti di maggiore interesse è la domotica[1], termine che deriva dall'unione delle parole domus (che in latino significa "casa") e robotica ed è appunto la scienza interdisciplinare che si occupa dello studio di tecnologie atte a migliorare la qualità della vita nella casa e più in generale negli ambienti antropizzati.

Home Manager[2] è un prototipo di applicazione per il controllo di una casa intelligente, progettato come un sistema multi-agente e implementato in cima dell'infrastruttura di coordinazione TuCSoN. Concetto principale è la totale integrazione tra le diverse funzioni presenti (illuminazione, controllo temperatura, automazioni, antintrusione, etc.) realizzate da dispositivi indipendenti, ma tutti dotati di un agente che faccia parte di una società ad agenti. Home manager è inoltre altamente configurabile attraverso policies e gestioni di piani.

La popolarità della domotica è aumentata notevolmente negli ultimi anni a causa di una maggiore semplicità e molto più ampia accessibilità di internet. Infatti con la rapida espansione del World Wide Web, si sono create opportunità per il controllo e il monitoraggio di tali tecnologie mediante l'utilizzo della rete.

Più precisamente si è giunti al concetto di Internet degli oggetti o IoT[3], acronimo dell'inglese Internet of Things, neologismo riferito all'estensione di Internet al mondo degli oggetti e dei luoghi concreti. Gli oggetti si rendono riconoscibili e acquisiscono intelligenza grazie al fatto di poter comunicare dati su sé stessi e accedere ad informazioni aggregate da parte di altri. Tutti gli oggetti possono acquisire un ruolo attivo grazie al collegamento alla Rete.

L'obiettivo dell'internet delle cose è di far sì che il mondo elettronico tracci una mappa di quello reale, dando un'identità elettronica alle cose e ai luoghi dell'ambiente fisico.

I campi di applicabilità sono molteplici: dalle applicazioni industriali (processi produttivi), alla logistica, fino all'efficienza energetica, all'assistenza remota e alla tutela ambientale.

Tuttavia le nuove ed eccitanti possibilità per aumentare la connettività, attraverso internet, dei dispositivi all'interno della casa, sono ancora da esplorare. Infatti, le soluzioni presentate finora offrono poco in termini di interoperabilità.

Recenti innovazioni hanno portato alla messa sul mercato di calcolatori single-board a basso costo, concepiti per stimolare l'insegnamento di base dell'informatica e della programmazione. Un classico esempio è il Raspberry, una tecnologia interattiva, efficiente e flessibile che è in grado di risponde a tutte le esigenze dei consumatori.

L'obiettivo che ci si propone di conseguire in questa tesi è duplice:

- da un lato, implementare l'intero sistema Home Manager sulla tecnologia Raspberry, creando così la possibilità di interagire con esso;
- dall'altro, sfruttare le potenzialità di tale tecnologia, in particolare la disponibilità di configurare sensori per simulare realmente il comportamento di un dispositivo intelligente, per passare così, da un puro sistema prototipale virtuale a uno parzialmente reale.

La tesi è quindi strutturata come segue. Nel primo capitolo si andrà ad analizzare il sistema Home Manager capendo i servizi che esso offre all'utente. Nel secondo capitolo si presenterà la tecnologia Raspberry e GrovePi, coi relativi concetti di base. Nel terzo e quarto capitolo si analizzeranno il problema e i requisiti necessari, si passerà poi all'analisi delle scelte progettuali ed infine si illustrerà l'esportazione del sistema sul Raspberry con la relativa configurazione dei sensori a disposizione, che rappresenta lo scopo di questa tesi, con il collaudo ed i risultati ottenuti. Completano la tesi, nel quinto ed ultimo capitolo, le conclusioni ed una breve panoramica di alcuni possibili sviluppi futuri.

Capitolo 1

Home Manager

In questo capitolo si esaminerà il prototipo attuale di Home Manager cercando di offrire una panoramica generale del programma.

1.1 Home Manager

HomeManager[2] è un applicativo multi agente per la gestione di una casa intelligente, che rappresenta un valido esempio di come possa essere applicata una metodologia di progettazione agent-oriented (SODA[4]). Di recente il prototipo, sviluppato inizialmente nel 2009, è stato esteso verso l'architettura Butlers[5], ovvero verso la realizzazione di un sistema che sia in grado di anticipare i bisogni utente ed agire autonomamente in base a conoscenza acquisita. Fino ad ora, il prototipo ha rappresentato esclusivamente una simulazione virtuale di casa intelligente, il tutto viene infatti eseguito su un normale computer.

1.1.1 Il prototipo attuale

HomeManager gestisce il controllo di una tipica abitazione famigliare. Si suppone che i locali siano provvisti di sensori che provvedono ad identificare i soggetti che entrano ed escono attraverso apposite tecniche di riconoscimento, è prevista anche la presenza di appositi terminali per l'identificazione esplicita. In questo modo il sistema conosce in qualsiasi istante la posizione degli utenti e può assecondare le loro preferenze. Gli obiettivi che il sistema mira a raggiungere per supportare l'utente sono:

- soddisfare le sue richieste e cercare di anticipare i suoi bisogni programmando per tempo i dispositivi secondo le sue preferenze;
- permettere l'invio di comandi direttamente o da remoto o tramite un'applicazione Android per smartphone;
- minimizzare il consumo energetico dell'abitazione.

Gli utenti sono suddivisi in ruoli differenti: possono essere abitanti della casa oppure semplici visitatori. I primi sono ulteriormente suddivisi in amministratori e utenti ordinari. Agli amministratori è concesso il pieno controllo: possono specificare tutte le politiche di gestione del sistema e quelle dei privilegi associati a tutti gli altri utenti. Gli ordinari, al contrario, possono solo esprimere le proprie preferenze e impartire comandi da remoto, non possono agire in nessun altro modo sugli altri utenti, né sul sistema. I visitatori non hanno alcun privilegio, ma a loro è comunque garantita un'assistenza di base.

HomeManager è realizzato in Java ed è basato sull'infrastruttura TuCSoN. Quest'ultimo si fonda sul concetto di centro di tuple, ovvero un particolare spazio di tuple in cui è possibile specificare il comportamento in termini di leggi di coordinazione.

L'abitazione simulata nel prototipo comprende quattro locali: un ingresso, una sala da pranzo, un bagno ed una camera da letto e in ciascuno di essi sono collocati alcuni dispositivi: una televisione, un frigorifero e uno stereo in cucina, una lavatrice nel bagno e un ulteriore televisione nella stanza da letto, oltre che attuatori per regolare l'impianto luminoso e la climatizzazione e sensori per la rilevazione dell'identità e della posizione degli utenti.

1.1.2 Architettura Butlers

L’architettura Butlers[5] definisce una cornice generale di riferimento per sistemi dotati di caratteristiche avanzate.

Butler, dall’inglese maggiordomo, è un componente, specializzato in una certa attività, ad alto livello di intelligenza con la possibilità di imparare le esigenze e le preferenze di un utente autonomamente, osservando il comportamento di quest’ultimo o interagendo con altri Butler. In questo modo il Butler, una volta istruito, è in grado di anticipare i bisogni dell’utente. È fondamentale, per un funzionamento ottimale, disporre di tutte le informazioni rilevanti, sia sui dispositivi presenti in casa, sia, se possibile, sull'utente e sulle sue preferenze e necessità.

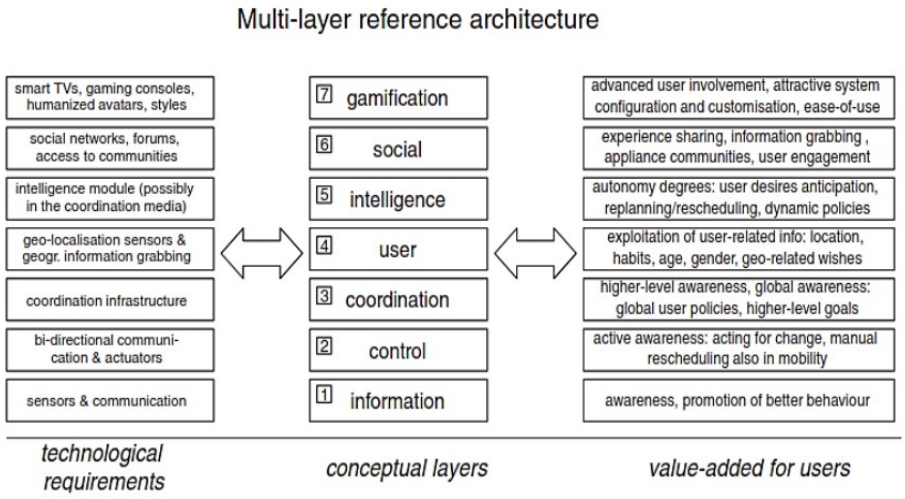

Figura 1.1: Livelli concettuali dell'architettura di riferimento con i requisiti tecnologici (sinistra) e i servizi offerti all'utente (destra)

Come descritto dalla Figura 1.1 il livello più basso, quello di information, è quello che utilizza le informazioni di tipo fisico, permettendo di ottenere informazioni, come ad esempio, i consumi energetici attuali della casa.

Il secondo livello, control, aggiunge una forma di controllo remoto permettendo di introdurre un'idea di casa automatizzata. Il terzo livello, coordination, permette di introdurre politiche di coordinazione e comunicazione tra i componenti che l'abitazione.

Il Butler user-aware è successivo al livello di coordination. Esso introduce, rispetto al livello sottostante, la conoscenza dell'utente, come la sua posizione (sfruttando strumenti di geolocalizzazione), l'età, le preferenze, le abitudini ed utilizza tutte queste informazioni per decidere e coordinare le attività su misura per l'utente in questione. In questo livello non c'è "intelligenza autonoma", intesa come la capacità di anticipare le possibili richieste dell'utente.

Le decisioni sono prese solo seguendo le regole incorporate al sistema di coordinazione e apprese durante il periodo di configurazione.

Il livello successivo, intelligence, introduce il Butler intelligente: ha tutte le caratteristiche dei Butler descritti in precedenza, alle quali aggiunge la capacità di anticipare le decisioni e i desideri degli utenti tramite l'esplorazione all'interno del sistema di tutte le risorse e informazioni utili per conseguire lo scopo. Esso è in grado di risolvere eventuali problematiche autonomamente e propone soluzioni. Successivamente è presente il livello social che introduce la possibilità di integrare i

Butlers con i social network, sia per accedere ad ulteriori informazioni sull'utente (foto vacanze, posizioni, attività preferite, etc....), sia per migliorare le proprie prestazioni scambiando informazioni con altri Butlers di altre abitazioni.

Questo livello, insieme a quello di gamification rendono a tutti gli effetti umanizzato il nostro collaboratore domestico virtuale. Si tratterà di un avere un componente in grado di coinvolgere gli utenti, prendere decisioni complesse e conoscere gli abitanti della casa. Inoltre, dovrà essere personalizzabile dall'utente stesso, permettergli di interagire tramite voce con l'utente ed essere visualizzabile in 3D da quest'ultimo. Per raggiungere un assistente domestico a 360 gradi come quello appena descritto, si passa attraverso una serie di altri livelli.

Il prototipo attuale di Home Manager opera sostanzialmente poco sopra il livello 3: la sua conoscenza dell'utente è infatti limitata al nome e ai ruoli associati, ma non sono possibili comportamenti tarati sulla specifica posizione o sulle sue abitudini. Proprio a questo livello, è possibile avere coordinazione tra i componenti che popolano il sistema e l'utente che può specificare le sue preferenze e le sue priorità, sia staticamente che dinamicamente.

1.2 TuCSoN

TuCSoN (Tuple Centres Spread over the Network) [6] è un modello e infrastruttura per la coordinazione dei processi distribuiti, così come agenti autonomi, intelligenti e mobili.

Si tratta di un'infrastruttura che supporta la comunicazione e la coordinazione degli agenti attraverso il concetto di centri di tuple, ovvero spazi di informazioni reattivi, condivisi e distribuiti sui nodi che compongono l'infrastruttura, dove gli agenti depositano, prelevano e leggono informazioni sotto forma di tuple, collezioni ordinate di porzioni di informazioni eterogenee.

Esso viene implementato come un middleware distribuito Java-based, distribuito Open Source sotto la licenza LGPL.

1.2.1 Introduzione a TuCSoN

L'infrastruttura TuCSoN può essere sintetizzata in tre entità principali:

- Agenti TuCSoN, i quali sono le entità base da coordinare all'interno del sistema distribuito.
- Centri di Tuple ReSpecT, che sono il mezzo sul quale avviene la coordinazione.
- Nodi TuCSoN, che rappresentano l'astrazione topologica di base, sul quale sono ospitati i centri di tuple.

Il sistema TuCSoN è un insieme di centri di tuple i quali lavorano insieme, in maniera coordinata, in un insieme di nodi distribuiti. I protagonisti di interazione e comunicazione del modello esaminato, sono gli agenti e i centri di tuple.

Gli agenti ricoprono il ruolo di entità attive, quindi hanno la possibilità di sfruttare i centri di tuple per comunicare. Questa comunicazione avviene per gli agenti tramite il linguaggio di coordinazione TuCSoN. Tale linguaggio è costituito da un insieme di primitive, le quali consentono lettura e scrittura di tuple all'interno di un qualsiasi centro di tuple.

I centri di tuple, invece, sono caratterizzati da un comportamento reattivo. Essi sono sfruttati dagli agenti e forniscono a quest'ultimi la possibilità di avere uno spazio condiviso sul quale gli agenti stessi possono comunicare tra loro attraverso le tuple. Sebbene i centri di tuple non abbiano come scopo la comunicazione con altri agenti, essi, tuttavia, possono essere programmati dinamicamente per reagire a determinate operazioni sulle tuple al loro interno, permettendo di ottenere un comportamento personalizzato dei centri stessi.

La topologia di un sistema TuCSoN adotta un modello client-server, dove i client sono gli agenti TuCSoN e i server sono i nodi TuCSoN, sui quali risiedono i centri di tuple. TuCSoN oltre essere un middleware Java-Based è anche Prolog-based: si appoggia a tuProlog[7], interprete Prolog leggero e Java-based, per applicazioni e infrastrutture distribuite per vari scopi, quali ad esempio la definizione delle tuple e dei loro template, le operazioni di parsing di primitive e identificatori, o per la definizione del linguaggio ReSpecT[8].

TuCSoN offre una serie di API che consentono un facile accesso alle primitive di coordinazione dal linguaggio Java. Oltre alle API, viene messo a disposizione dalla

libreria per l'utilizzo di TuCSoN in Java, alice.tucson.api.TucsonAgent che permette di creare facilmente agenti TuCSoN in Java.

I centri di tuple in TuCSoN possono essere strutturati come organizzazioni. Nasce così in TuCSoN il concetto di ACC. ACC è l'acronimo di Agent Coordination Context e consiste in un'interfaccia assegnata ad un agente che gli consente di effettuare operazioni su determinati centri di tuple facenti parte di una specifica organizzazione, differente a seconda dell'ACC assegnatogli. Questa interfaccia, fornita dall'infrastruttura, ha il duplice scopo di fornire un mezzo tramite il quale comunicare con il sistema, sia di limitare la comunicazione stessa alle organizzazioni specifiche.

1.2.2 Coordinazione in TuCSoN

Una volta attribuiti gli identificatori dei protagonisti del sistema è possibile avviare la comunicazione. Le comunicazioni in un sistema TuCSoN avvengono mediante l'utilizzo di alcune primitive definite dal linguaggio di coordinazione TuCSoN, le quali consentono agli agenti di leggere, aggiungere o eliminare tuple all'interno di un centro di tuple.

Lo scopo delle operazioni è quello di consentire agli agenti di comunicare tra loro e di sincronizzarsi. Una qualsiasi operazione avviene per mezzo di un agente verso un centro di tuple, che ha il compito di eseguirla.

È importante sottolineare, ai fini di questa tesi, che i centri di tuple appartengono ai nodi mentre gli agenti vivono ovunque sulla rete e possono interagire con il centro di tuple semplicemente conoscendo l'identificativo univoco del centro di tuple e purché il nodo su cui risiede quest'ultimo sia attivo.

Le operazioni richieste sono costituite generalmente da due fasi:

- Invocation, richiesta inoltrata dall'agente verso il centro di tuple contenente tutte le informazioni necessarie all'esecuzione dell'operazione richiesta.
- Completion, la quale riporta il risultato ottenuto dall'operazione nel centro di tuple e alcune informazioni relative all'esecuzione.

Le primitive offerte per la comunicazione principali sono:

- `out`, scrive una tupla specificata nel centro di tuple.

- `in`, elimina una tupla specifica nel centro di tuple.
- `rd`, legge una tupla specificata nel centro di tuple.

Nella versione attuale di TuCSoN sono state aggiunte altre primitive molto comode, non presenti nelle versioni precedenti: `out_all`, `rd_all`, `in_all`. Queste bulk sono le corrispondenti di quelle sopra elencate con la particolarità di effettuare operazioni su più di una tupla che fanno match con il template specificato.

Capitolo 2

La tecnologia Raspberry Pi2

In questo capitolo si esaminerà la tecnologia Raspberry cercando di offrire una panoramica generale del calcolatore e dei sensori che si sono utilizzati in questa tesi.

2.1 Raspberry Pi2

Il Raspberry Pi2[9] è un single-board computer (un calcolatore implementato su una sola scheda elettronica) sviluppato nel Regno Unito dalla Raspberry Pi Foundation.
L'idea di base è la realizzazione di un dispositivo economico, concepito per stimolare l'insegnamento di base dell'informatica e della programmazione nelle scuole.
Il progetto ruota attorno a un System-on-a-chip (SoC) Broadcom BCM2835, che incorpora un processore ARM1176JZF-S a 700 MHz, una GPU VideoCore IV, e 256 o 512 Megabyte di memoria.
Il progetto non prevede né hard disk né una unità a stato solido, affidandosi invece a una scheda SD per il boot e per la memoria non volatile. La scheda è stata progettata per ospitare sistemi operativi basati su un kernel Linux o RISC OS, come ad esempio Raspbian il sistema operativo ufficialmente supportato dalla Fondazione.

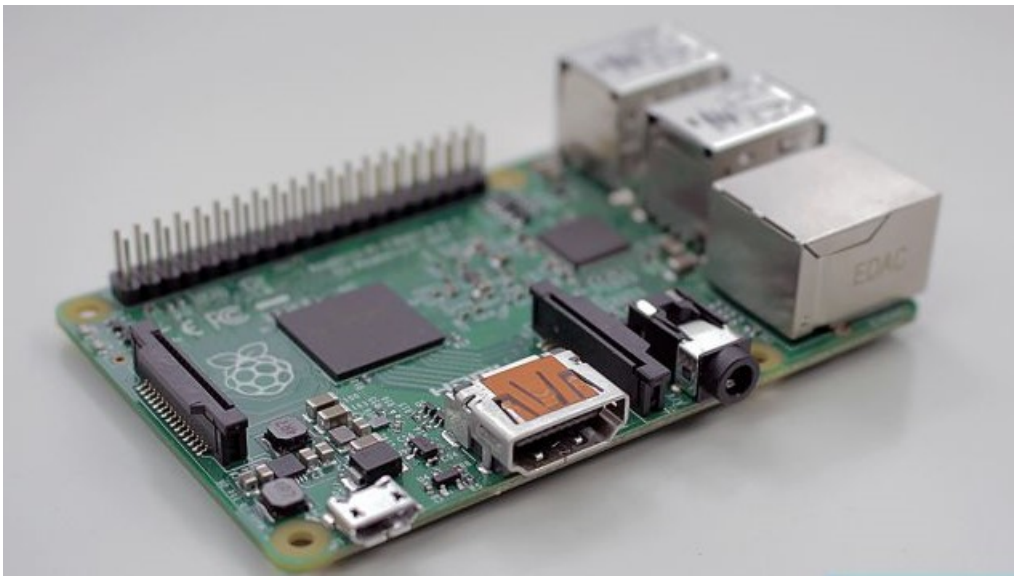

Figura 2.1: Raspberry Pi2 Model B

Come si vede dalla Figura 2.1 il Raspberry Pi2 Model B è dotato dei seguenti componenti:

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core

2.2 GrovePi

Il GrovePi [10] è una scheda elettronica che si collega a un dispositivo come il Raspberry, permettendo così di interfacciare, in modo semplificato, sensori scelti tra una centinaia di differenti moduli disponibili, che possono essere programmati per monitorare, controllare e automatizzare i dispositivi di proprio interesse.

Figura 2.2: GrovePi+

Figura 2.3: GrovePi+ Starter Kit

Come si vede dalla Figura 2.3, nello Starter Kit troviamo differenti sensori e dispostivi che possiamo collegare alla scheda. Di particolare interesse per questa tesi sono stati:

- Un led rosso;
- Un sensore di temperatura;
- Un segnalatore acustico;
- Un display lcd.

Il GrovePi è impilato sopra al Raspberry Pi, senza la necessità di altre connessioni e la comunicazione tra i due avviene tramite l'interfaccia I2C.

Tutti i moduli Grove si collegano ai connettori universali sulla piattaforma GrovePi tramite il cavo connettore universale a 4 pin .

I moduli Grove, che lavorano su segnali analogici e digitali, si collegano direttamente al microcontrollore ATMEGA328 presente sul GrovePi. Questo microcontrollore contiene un convertitore analogico-digitale (A/D) a sei canali (risoluzione a 10 bit) e funge da interprete tra il Raspberry Pi e i sensori Grove, inviando, ricevendo ed eseguendo i comandi inviati dal Raspberry.

I pin analogici sono di solito utilizzati per la lettura dei sensori analogici, ma possono essere utilizzati anche per un uso generale dell'I/O, stessa cosa per i pin digitali 0-13. Motivo per cui se c'è la necessità di avere più digital sockets, è possibile riutilizzare quelle analogiche.

Inoltre, il Raspberry Pi2 ha un bus I2C e un bus seriale, attraverso i quali ci si può connettere direttamente ai sensori collegati alle porte I2C e USART del GrovePi.

2.3 Lettore Rfid

Il modulo SL030 RFID HF[11] è un mini lettore/scrittore RFID 13.56 MHz con interfaccia IIC. Questo modulo integrato permette un rilevamento automatico di tag Rfid in tempo reale, che si muove avvicinandosi o allontanandosi dall'area di rilevamento ed emette un rapporto tramite un pin logico di output. Inoltre, integra tutti i componenti necessari e l'antenna in un PCB (Printed Circuit Board).

SL030 supporta i seguenti tag Rfid: Mifare 1k, Mifare 4k, Mifare UltraLight e presenta le seguenti caratteristiche:

- Rilevamento automatico del tag;
- Antenna incorporata;
- Comunicazione sul bus I2C;
- Distanza di funzionamento: fino a 50mm, dipende dal tipo di tag;
- Dimensioni 38 × 38 × 3 mm.

Figura 2.4: SL030 *Figura 2.5: Tag Rfid MIFARE™ Classic 1K*

In particolare lo schema dei contatti è descritto nella figura seguente:

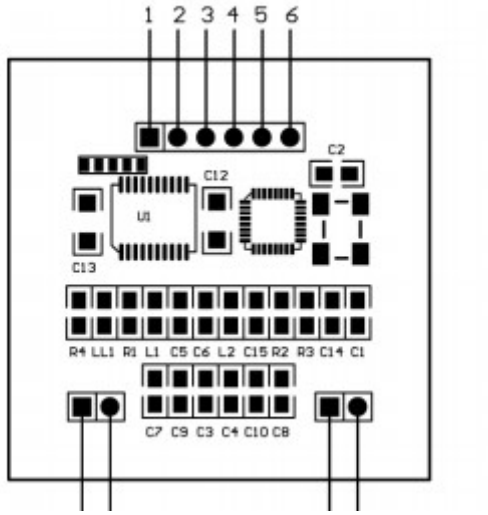

Figura 2.6: SL030 nel dettaglio

PIN	SIMBOLO	TIPOLOGIA	DESCRIZIONE
1	VDD	PWR	Power supply, 2.5V to 3.6VDC
2	IN	Input	Il fronte di discesa risveglia SL030 dalla modalità power down
3	SDA	Input/Output	Serial Data Line
4	SLC	Input	Serial Clock Line
5	Out	Output	Segnale che indica la presenza di un Tag: low level indica che il tag è presente high level indica che il tag non è presente
6	GND	PWR	Ground

Capitolo 3

Verso HomeManagerPi : Analisi e Progettazione

In questo capitolo si spiegheranno le fasi di analisi e progettazione, cercando di offrire una panoramica generale di ciò che è stato effettuato nelle prime fasi dello svolgimento di questa tesi.

3.1 Obiettivi

L'obiettivo di questa tesi è inizialmente estendere il prototipo attuale del sistema Home Manager muovendo il primo passo verso l'implementazione fisica dell'intero sistema, in particolare portando il tutto sulla tecnologia Raspberry; successivamente inserire diversi sensori e componenti per andare a realizzare concretamente il dispositivo virtuale frigo intelligente già presente su Home Manager. Si introdurranno, quindi, un sensore di temperatura, uno schermo lcd, un emettitore di segnale acustico, un led e un lettore rfid, che verranno programmati per registrare qualunque spostamento da e verso il frigo notificandolo sullo schermo lcd in modo da rendere l'utente sempre conscio dei prodotti contenuti nel frigorifero.

3.2 Analisi dei requisiti

Per garantire la portabilità del sistema, occorre che sul Raspberry sia presente una JVM (JavaVirtualMachine), quindi che sia installata una versione di Java.
Poiché il Raspberry supporta distribuzione Unix based, ed esse sono compatibili con Java, gli unici due requisiti che occorre soddisfare in linea di principio sono quelli di montare un sistema operativo basato appunto su un kernel Linux, e successivamente di installare una versione aggiornata di Java, la quale permetterà anche di eseguire TuCSoN.

In questo modo siamo in grado di soddisfare il primo obiettivo, ovvero far funzionare l'applicativo Home Manager sul Raspberry in modo del tutto indipendente da un normale computer come si è sempre dovuto fare finora.
Per quanto riguarda il secondo obiettivo, esso richiede sicuramente un po' più di analisi e lavoro. La realizzazione del frigorifero intelligente prevede appunto l'utilizzo di sensori, che dovranno essere configurati per funzionare con Java, e successivamente andranno adeguatamente programmati per fare in modo che all'avvicinamento di un tag Rfid, venga emesso un segnale acustico, acceso un led e scritto nel display lcd le informazioni relative al prodotto avvicinato; riportando successivamente sullo stesso display la temperatura del frigorifero, ottenuta mediante l'apposito sensore.

I sensori/dispositivi sono classificati in due categorie: quelli che fanno parte dello StarterKit di GrovePi e il lettore Rfid. Per i primi il settaggio e l'utilizzo dovrebbe risultare più semplice, in quanto sul sito internet del prodotto è presente anche una libreria per l'utilizzo attraverso Java. Per il secondo invece si dovranno cercare delle librerie e consultare il datasheet per poterlo rendere operativo.

3.3 Analisi del problema

Sulla carta i presupposti per rendere funzionante Home Manager sul Raspberry ci sono tutti, in quanto appunto una volta installata una distribuzione che supporta Java, non dovrebbe occorre nient'altro per poterlo eseguire. Questo deriva dalla potenzialità di Java di essere stato progettato per essere il più possibile indipendente dalla piattaforma di esecuzione.
La fase che potrà risultare più problematica, sarà quella della configurazione dei sensori e dell'utilizzo attraverso il linguaggio Java, poiché solitamente per questo genere di operazioni vengono utilizzati linguaggi di più basso livello o che comunque permettono un accesso più diretto all'hardware del Raspberry come il C, Python o direttamente script in Bash.
La ricerca di librerie per svolgere questo compito sarà assolutamente necessaria, tenendo a mente che la configurazione presenterà più difficoltà per il lettore Rfid che per gli altri moduli, in quanto essi sono stati progettati proprio per l'interazione con il Raspberry.

Per il lettore Rfid occorrerà infatti andare a capire come avviene la comunicazione e i meccanismi di interazione nelle operazioni di lettura dei tag; sarà perciò indispensabile consultare attentamente il datasheet del sensore.

3.4 Progettazione

Per raggiungere il primo obiettivo si dovrà quindi:

- Installare l'immagine Dexter Industries Raspbian for Robots sul Raspberry, che è una distribuzione basata su Linux, appositamente fatta per semplificare l'utilizzo della piattaforma GrovePi e di tutti i sensori annessi;
- Installare l'ultima release di Java sul Raspberry;
- Copiare sul Raspberry l'ultima versione di Home Manager con anche la versione aggiornata di TuCSoN.

Per il secondo si dovrà invece:

- Scrivere un agente TuCSoN, che utilizzando le librerie di GrovePi e quelle di Pi4J, legga il tag Rfid quando esso viene avvicinato all'apposito sensore, e notifichi questa informazione in modo da aggiornare l'elenco dei prodotti contenuti nel frigorifero.

Nel concreto si realizzerà un agente TuCSoN rappresentato da una classe che sfruttando la libreria Pi4J, legge fino a quando non trova un tag Rfid valido. Trovato un tag valido lo legge e preleva da un centro di tuple che funziona da "dizionario" il corrispondente nome del prodotto; dopo di che controlla nel centro di tuple che mantiene in memoria l'elenco dei prodotti contenuti nel frigo, se quello appena letto è uno di quelli già presente nel frigorifero o se invece è un nuovo prodotto. Nel caso in cui sia un nuovo prodotto, l'agente provvederà ad aggiungere una tupla nel centro di tuple che rappresenta lo stato attuale del frigo e notificherà, utilizzando le librerie di GrovePi, l'aggiunta sul display lcd con un'opportuna scritta e infine, mostrerà la temperatura del frigo, prelevata dall'apposito sensore.
Nel caso in cui invece sia un prodotto già presente, l'agente provvederà a rimuovere la tupla corrispondente dal centro di tuple, anche in questo caso notificando il tutto in modo appropriato.

3.5 NetBeans IDE 8.0.2

NetBeans[12] è un ambiente di sviluppo integrato (IDE) multi-linguaggio, scritto interamente in Java, scelto dalla Oracle Corporation come IDE ufficiale da contrapporre al più diffuso Eclipse.

Si è scelto di utilizzare NetBeans perché permette di semplificare il testing del codice, e il passaggio dello stesso sul Raspberry.

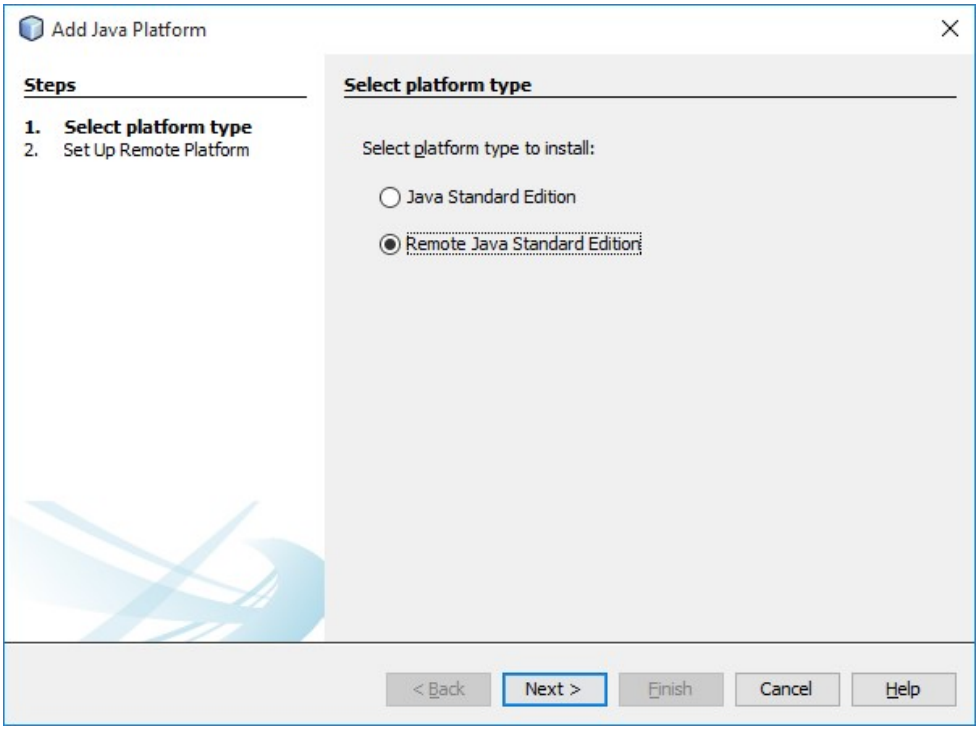

Figura 3.1: Aggiungi Java Platform

Come possiamo notare dalla Figura 3.1, NetBeans permette di aggiungere una nuova piattaforma Java che può essere remota. Scegliendo proprio quest'ultima arriviamo alla seguente schermata:

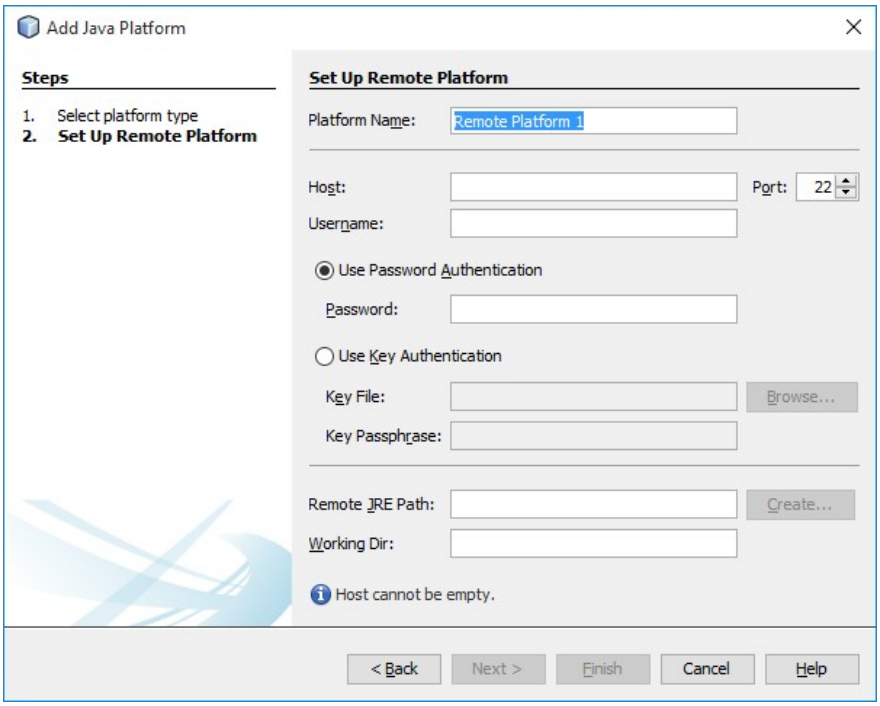

Figura 3.2: Aggiungi Java Platform (Dettagli)

Una volta compilati i seguenti campi abbiamo configurato la nostra piattaforma remota, che ci permetterà di testare il codice sul Raspberry direttamente dal pc, poiché il passaggio del progetto verrà automaticamente fatto attraverso la rete da NetBeans.

Una volta configurata la piattaforma dovremo avere una situazione analoga:

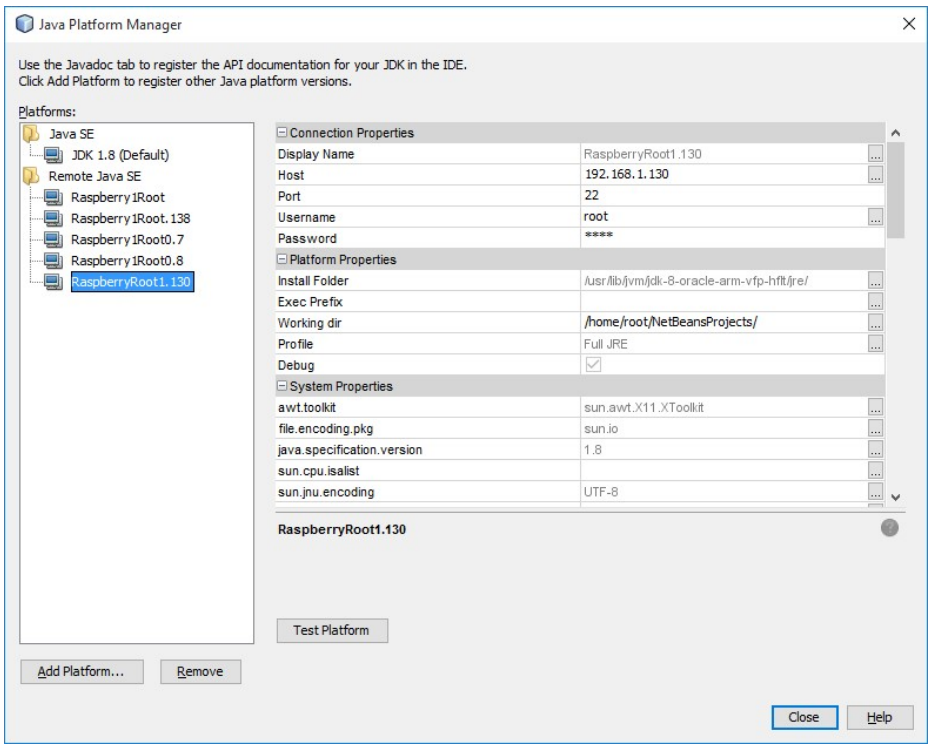

Figura 3.3: Esempio Remote Java Platform

3.6 Librerie Pi4J

Questo progetto[13] nasce con lo scopo di fornire una vasta offerta di friendly object-oriented I/O API e implementare librerie per programmatori Java per accedere a tutte le funzionalità di I/O della piattaforma Raspberry Pi. Questo progetto astrae il basso livello di integrazione nativa e interrupt monitoring per consentire ai programmatori Java di concentrarsi sull'attuazione della logica di business della loro applicazione.
Le funzionalità e i vantaggi che queste librerie offrono sono svariati, tra i più importanti abbiamo:

- Configurazione pin di GPIO
- Controllo (scrittura e lettura) stato dei pin di GPIO
- Invio e ricezione dati tramite comunicazione seriale RS232
- Comunicazione I2C
- Classi wrapper per l'accesso diretto a librerie WiringPi da Java

Per capire l'importanza di queste librerie e la semplicità con cui esse possono essere utilizzate, si passa ora a un esempio[14] concreto di impiego.

Si vuole accendere un led collegato ai pin del Raspberry nel seguente modo.

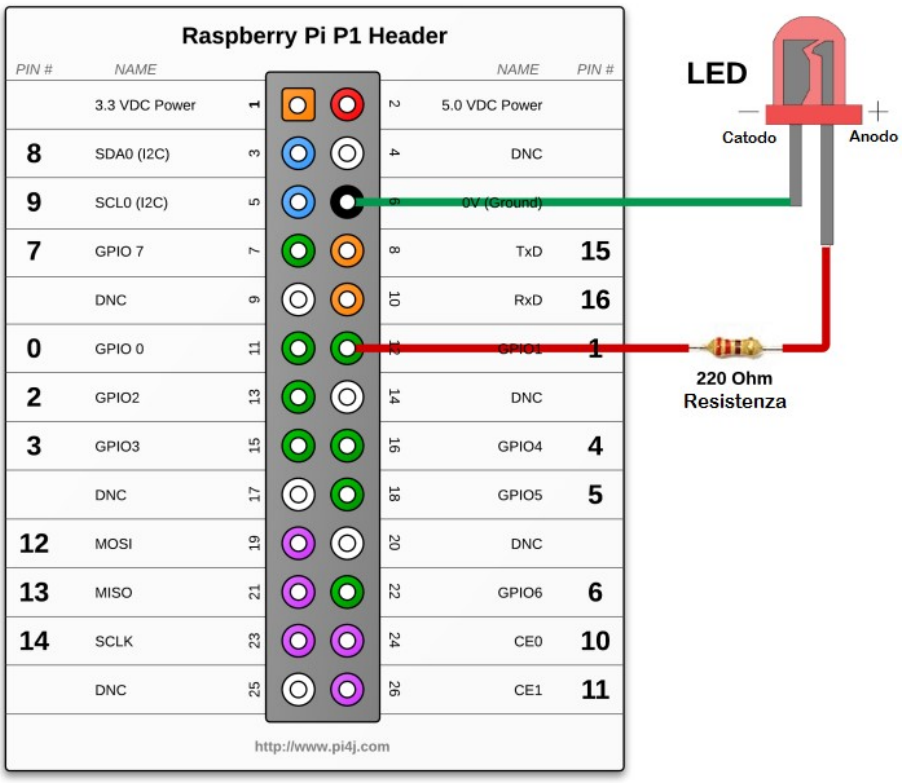

Figura 3.4: Circuito del caso di esempio

Il codice sarà impostato nel seguente modo:

```
import com.pi4j.io.gpio.GpioController;
import com.pi4j.io.gpio.GpioFactory;
import com.pi4j.io.gpio.GpioPinDigitalOutput;
import com.pi4j.io.gpio.PinState;
import com.pi4j.io.gpio.RaspiPin;

/**
 * This example code demonstrates how to perform simple state
 * control of a GPIO pin on the Raspberry Pi.
 * @author Robert Savage
 */
public class ControlGpioExample {

    public static void main(String[] args) throws InterruptedException {

      System.out.println("<--Pi4J--> GPIO Control Example ... started.");

// create gpio controller
      final GpioController gpio = GpioFactory.getInstance();

// provision gpio pin #01 as an output pin and turn on
      final GpioPinDigitalOutput pin =
gpio.provisionDigitalOutputPin(RaspiPin.GPIO_01, "MyLED", PinState.HIGH);

// set shutdown state for this pin
      pin.setShutdownOptions(true, PinState.LOW);

      System.out.println("--> GPIO state should be: ON");

      Thread.sleep(5000);

// turn off gpio pin #01
      pin.low();
      System.out.println("--> GPIO state should be: OFF");

      Thread.sleep(5000);

// toggle the current state of gpio pin #01 (should turn on)
      pin.toggle();
      System.out.println("--> GPIO state should be: ON");

      Thread.sleep(5000);

// toggle the current state of gpio pin #01  (should turn off)
      pin.toggle();
      System.out.println("--> GPIO state should be: OFF");

      Thread.sleep(5000);

// turn on gpio pin #01 for 1 second and then off
     System.out.println("--> GPIO state should be: ON for only 1 second");
     pin.pulse(1000, true); // set second argument to 'true' use a blocking call

// stop all GPIO activity/threads by shutting down the GPIO controller
// (this method will forcefully shutdown all GPIO monitoring threads and scheduled tasks)

     gpio.shutdown();
    }
}
```

3.7 Librerie GrovePi

IoTDevices[15] (Internet of things devices libraries) è una libreria basata sulle librerie di Pi4J che semplifica notevolmente l'interfacciamento e quindi l'utilizzo dei sensori specifici di GrovePi. Il suo funzionamento è molto analogo a quello di Pi4J, offre in più l'interfacciamento automatico tra i pin del Raspberry e quelli del GrovePi.

Capitolo 4

Verso HomeManagerPi : Implementazione e Collaudo

In questo capitolo si spiegheranno le fasi di implementazione e collaudo, cercando di offrire una panoramica generale sulle scelte implementative adottate e sui vari sperimenti effettuati prima di arrivare alla meta prestabilita.

4.1 Scelte implementative

Si è deciso di utilizzare l'ultima versione di Java (Java8). Per installarla sul Raspberry si può utilizzare il comando: *sudo apt-get install oracle-java8-jdk* stando attenti a controllare che sia stata selezionata la corretta versione di Java. Per scegliere la versione che si vuole, si può impiegare il comando *sudo update-alternatives --config java*.

Una volta installato Java e portato tutto il materiale necessario, ovvero una versione recente di TuCSoN e il .jar di Home Manager, possiamo già verificare che esso funzioni sul Raspberry. Le operazioni che bisognerà eseguire sono:

- Avviare il nodo TuCSoN;
- Eseguire il file .jar.

A questo punto abbiamo il sistema funzionante sul Raspberry e il primo obiettivo è stato raggiunto.

Per quanto riguarda il secondo, andiamo a predisporre i vari sensori e li andiamo ad attaccare nel seguente modo:

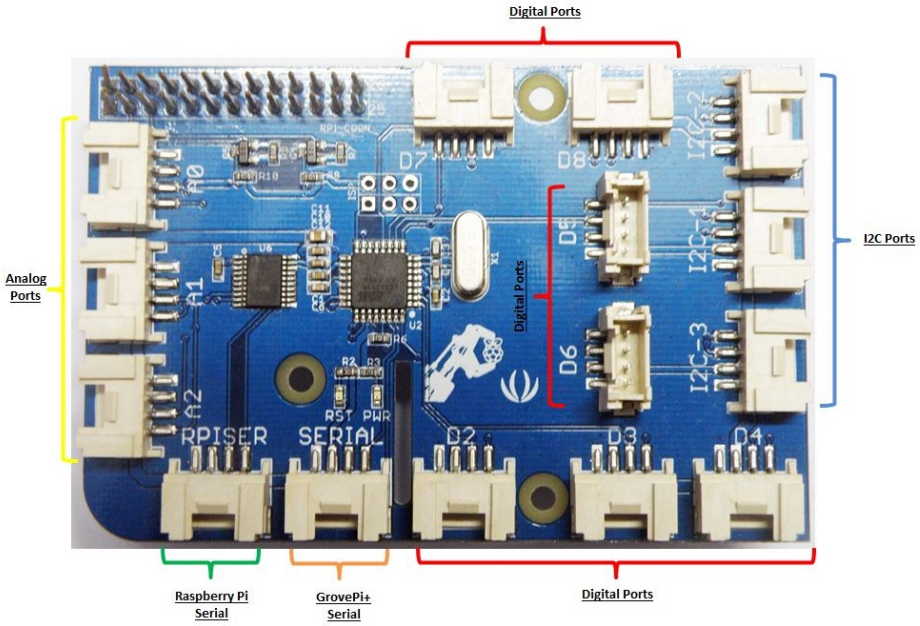

Figura 4.1: Porte della GrovePi

il display lcd alla porta I2C-1, il sensore di temperatura alla porta D2, il led alla porta D3 e l'emettitore di segnale acustico alla porta D4. Per quanto riguarda il lettore Rfid invece lo colleghiamo con il cavo in dotazione ai seguenti pin: 1,3,5,7.

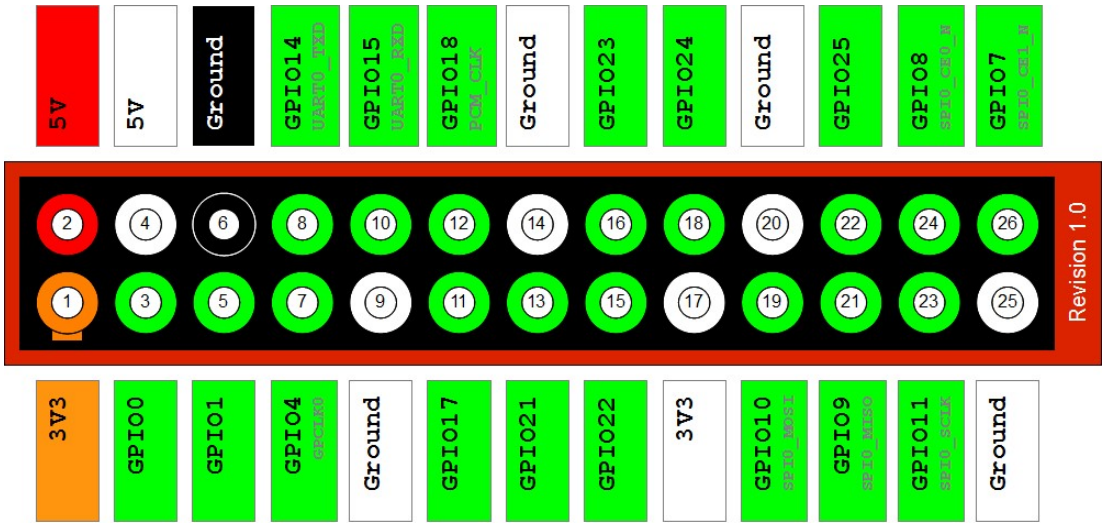

Figura 4.2: Porte della GrovePi

La programmazione dei moduli GrovePi è analoga a quella dell'esempio proposto precedentemente:

```
public void Display(String product,int mode){
    GrovePi grovePi;
    System.out.println("Inizio ");
    try {
    grovePi = new GrovePi4J();
    GroveDigitalOut led = grovePi.getDigitalOut(3);
    GroveDigitalOut bip = grovePi.getDigitalOut(4);
    GroveRgbLcd lcd = grovePi.getLCD();
    GroveTemperatureAndHumiditySensor termometro = new
GroveTemperatureAndHumiditySensor(grovePi, 2,
GroveTemperatureAndHumiditySensor.Type.DHT11);
    int[] color = new int[]{0, 128, 255};
    String text;
    if (mode==0) text= "PRODUCT TO BE\nREMOVED: "+product.toUpperCase();
    else text= "PRODUCT TO BE\nENTERED: "+product.toUpperCase();
    led.set(true);
    bip.set(true);
    lcd.setRGB(color[0], color[1], color[2]);
    Thread.sleep(100);
    led.set(false);
    bip.set(false);
    lcd.setText(text);
    Thread.sleep(2000);
    System.out.println(""+termometro.get().toString());
    double temperatura= termometro.get().getTemperature();
//  String temp=""+temperatura;
    lcd.setText("FRIDGE \nTEMP: "+temperatura+"C");
    } catch (IOException | InterruptedException e) {}
  }
```

Questa è la funzione che realizza tutta la sequenza delle operazioni che devono essere eseguite una volta letto il tag Rfid; essa prende in ingresso una stringa che rappresenta il nome del prodotto e un intero che può assumere i valori 0 o altro nel caso in cui sia un prodotto in uscita o in ingresso.

Letto l'Rfid viene invocata questa funzione che si occuperà di lampeggiare il led, emettere un breve segnale acustico e mostrare nel display lcd il prodotto passato alla funzione nella modalità indicata.

La funzione che invece è incaricata della lettura dell'Rfid, è leggermente più complessa, più precisamente, è più complesso il ragionamento necessario a capire come è stata impostata.

Per poter leggere un tag è stato necessario il datasheet[16] del sensore, dal quale si ottengono diverse importanti informazioni:

4. COMMAND DESCRIPTION

4-1. FORMAT

Host Write Command to SL030:

Address	Len	Command	Data

Address: 1 byte, 0xA0
Len: 1 byte indicating the number of bytes from Command to the end of Data
Command: 1 byte Command code, see Table 3
Data: Variable length depends on the command type

Host Read The Result:

Address	Len	Command	Status	Data

Address: 1 byte, 0xA1
Len: 1 byte indicating the number of bytes from Command to the end of Data
Command: 1 byte Command code, see Table 3
Status: 1 byte Command status, see Table 4
Data: Variable length depends on the command type.

Figura 4.3: Formato delle istruzioni

4-2. Command Overview

Table 3

Command	Description
0x01	Select Mifare card
0x02	Login to a sector
0x03	Read a data block
0x04	Write a data block
0x05	Read a value block
0x06	Initialize a value block
0x07	Write master key (key A)
0x08	Increment value
0x09	Decrement value
0x0A	Copy value
0x10	Read a data page (Ultralight & NATG203)
0x11	Write a data page (Ultralight & NTAG203)
0x12	Download Key
0x13	Login sector via stored Key
0x50	Go to Power Down mode
0xF0	Get firmware version

Figura 4.4: Lista dei comandi disponibili

4-3-1. Select Mifare card

Host Write:

Len	0x01

Host Read:

Len	0x01	Status	UID	Type

Status: 0x00: Operation succeed
0x01: No tag

UID: The uniquely serial number of Mifare card

Type: 0x01: Mifare 1k, 4 byte UID
0x02: Mifare 1k, 7 byte UID [1]
0x03: Mifare UltraLight or NATG203[2], 7 byte UID
0x04: Mifare 4k, 4 byte UID
0x05: Mifare 4k, 7 byte UID [1]
0x06: Mifare DesFire, 7 byte UID
0x0A: Other

Figura 4.5: Istruzione di lettura Tag Rfid

La figura 4.3 serve per capire come devono essere scritte le istruzioni e notiamo che bisogna indicare un comando. Dalla figura 4.4 possiamo dedurre che il comando che a noi interessa è quello di Select Mifare Card (0x01) dal quale possiamo ottenere l'id univoco del tag.

Infine dalla figura 4.5 si desume quale sarà l'output del comando desiderato; da tutte queste importanti informazioni si può scrivere la funzione di lettura del tag Rfid.

```
import com.pi4j.io.i2c.*;
import java.io.IOException;
import javax.xml.bind.DatatypeConverter;
import org.iot.raspberry.*;

public class Sensor4 {

//Raspberry Pi's I2C bus
    private static final int i2cBus = 1;
// RFID addresses
    private static final int DEVICE_ADDRESS = 0x50;
// Read RFID command
    private static final byte getRfidCmd = (byte) 0x01;
    private static final byte firmwareRfidCmd = (byte) 0xF0;
//I2C bus
    I2CBus bus;
// Device object
    private I2CDevice sl030;

    public Sensor4() {

   try {
     bus = I2CFactory.getInstance(I2CBus.BUS_1);
     System.out.println("Connected to bus OK!!!");
     sl030 = bus.getDevice(DEVICE_ADDRESS);
     System.out.println("Connected to device OK!!!");
        } catch (IOException e) {
         System.out.println("Exception: " + e.getMessage());
        } }
```

```
public String readRfid() throws IOException {

   byte[] writeBuffer= new byte[2];
   String result="NO-TAG";
   try {
// SEND COMMAND TO DEVICE TO REQUEST TAG RFID
        writeBuffer[0]=(byte)0x01;           // LENDTH (CMD+DATA)
        writeBuffer[1]=getRfidCmd;           // COMMAND
        sl030.write(writeBuffer, 0, 2);
        Thread.sleep(1000);

// READ SELECT-TAG RESPONSE
// first just read 1 byte that represents the com+stat+data+payload
        int length = sl030.read();

        System.out.println("TOTAL BYTES AVAILABLE: " + length);

// if there are no remaining bytes, then we can exit the function
        if(length <= 0) {
        System.out.format("Error: %n bytes read for LENGTH/n", length);
        return "NO-TAG";
        }

// if there is any length of remaining bytes, then lets read them now
        byte[] readBuffer = new byte[length+1];
        int readTotal = sl030.read(readBuffer, 0, length);
        System.out.println("Letti: "+readTotal);

// validate to ensure we got back at least the command and status bytes
        if (readTotal <= 3) {
         System.out.format("Error: %n bytes read/n", readTotal);
         return "NO-TAG";
        }

        byte leng=readBuffer[0];
        byte command = readBuffer[1];  // COMMAND BYTE
        byte status  = readBuffer[2];  // STATUS BYTES
        byte type= readBuffer[readBuffer.length-1];
        byte[] uidByte=new byte[readBuffer.length-4];
        int k=3;
        for(int i=0; i<uidByte.length; i++)
        {
          uidByte[i]=readBuffer[i+k];
        }
// now we need to get the payload data (if there is any?)
        String uid="?";

        if(readTotal > 3){
               uid=DatatypeConverter.printHexBinary(uidByte);
            }
// what did we get?
//         System.out.println("-- LENGTH BYTE READ  : " + leng);
//         System.out.println("-- COMMAND BYTE READ : " + command);
//         System.out.println("-- STATUS BYTE READ  : " + status);
//         System.out.println("-- TYPE BYTE READ  : " + type);
//         System.out.println("-- DATA BYTES READ   : " + uid);

           result=uid;
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        } catch (InterruptedException e) {
           System.out.println("Interrupted Exception: " + e.getMessage());
        }
    }
```

Passando ora alla parte di interfacciamento con Home Manager, è stata realizzata una classe che rappresenta l'agente TuCSoN e un main che serve per l'avvio di tutto il programma.

Il main del programma sarà così strutturato:

```
public static void main(String[] args) {
try{

 System.out.println("Initializing Connection to Home Manager!");
 TucsonTupleCentreId fridge_tc=new TucsonTupleCentreId("fridge", "localhost", "20504");
 TucsonTupleCentreId dictionary_tc=new TucsonTupleCentreId("dictionary", "localhost", "20504");

//  Versione Client/Server Remoto
// TucsonTupleCentreId fridge_tc=new TucsonTupleCentreId("fridge", "192.168.1.132", "20504");
// TucsonTupleCentreId dictionary_tc=new TucsonTupleCentreId("dictionary", "192.168.1.132",
  "20504");

   FrigoAgent fAgent= new FrigoAgent("5", fridge_tc,dictionary_tc);
   fAgent.main();

 } catch (TucsonInvalidTupleCentreIdException | TucsonInvalidAgentIdException ex) {
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
 }
}
```

Come possiamo vedere andiamo a richiamare due centri duple presenti in locale e li passiamo all'agente TuCSoN vero e proprio che sarà così impostato:

```
public FrigoAgent(String id, TucsonTupleCentreId tuple,TucsonTupleCentreId tuple2) throws
  TucsonInvalidAgentIdException {
        super(id);
        this.fridge_tc=tuple;
        this.dictionary_tc=tuple2;
    }

@Override
    protected void main() {
  System.out.println("-- FRIGO AGENT STARTED.");
        TucsonAgentId agentPref;
        try{
        agentPref = new TucsonAgentId("5");
        final NegotiationACC negAcc = TucsonMetaACC.getNegotiationContext(agentPref);
        this.acc = negAcc.playDefaultRole();
        this.dictionary=this.getDictionary();
        this.sensor=new Sensor4();
            while(true) {
//            Thread.sleep(1000);
// Waiting to read a valid Rfid
             String tagrfid=this.sensor.readRfid();
// IF VALID
//I check the id-nameproduct
             if (!tagrfid.equalsIgnoreCase("NO-TAG"))
             {
                String nameProduct= this.dictionary.get(tagrfid);
// I check if it is already in the fridge
                ArrayList<String> presenti=this.getRealList();
                boolean trovato=false;
                for(String s: presenti)
```

```
                {
                   if (s.contains(tagrfid))
                   {
                       trovato=true;
                       break;
                   }
                }
                User adminFrigo;
adminFrigo=new User(123, "AdminFrigo", "AdminFrigo", "admin", "admin", "AdminFrigo", 25, 22,"AdminFrigo");
               if (trovato==true)
               {
                  System.out.println("TROVATO");
                  this.sensor.Display(nameProduct,0);
// if is in fridge I delete one piece
                  this.deleteR(tagrfid,nameProduct);
                  this.delete(nameProduct, 1, adminFrigo);
               }
// On the other case, I put the product in the fridge
               else
               {
                   System.out.println("NON TROVATO");
                   this.sensor.Display(nameProduct,1);
                   this.insertR(tagrfid,nameProduct);
                   this.insert(nameProduct, 1, adminFrigo);
               }
               System.out.println("-- FRIGO AGENT UPDATED.");
               Thread.sleep(2000);
//               ArrayList<String> now=this.getList();
//               now.stream().forEach((s) -> {System.out.println(s);});
//               ArrayList<String> now2=this.getRealList();
//               now2.stream().forEach((s) -> {System.out.println(s);});
               }
             }
           }catch(Exception ex){
               System.out.println("FrigoAgent : error in activity.");
           }
 }
```

Questo agente continua a cercare di leggere un tag Rfid valido, e quando ne trova uno, attraverso il centro di tuple che è adibito alla funzione di dizionario, ottiene il nome del prodotto corrispondente all'id del tag. Dopo di che controlla se l'id di quel prodotto è già nel frigorifero e in caso positivo, notifica nel display, attraverso la funzione presentata in precedenza, l'evento di "prodotto in uscita" dal frigo; andando ad eliminare la tupla dal centro di tuple che memorizza il contenuto del frigorifero. In caso negativo procede a notificare l'evento di "prodotto in ingresso" e a inserire la tupla nei centri di tuple opportuni.

4.2 Collaudo

Per quanto riguarda il collaudo, esso può essere fatto sia attraverso NetBeans sia utilizzando esclusivamente il Raspberry. Poiché il primo obiettivo di questa tesi è di rendere Home Manager indipendente da un personal computer procedere con il seguente ordine:

- Preparare il Raspberry collegando tutti i sensori come spiegato in precedenza, collegando anche il cavo Hdmi dello schermo all'apposito ingresso;

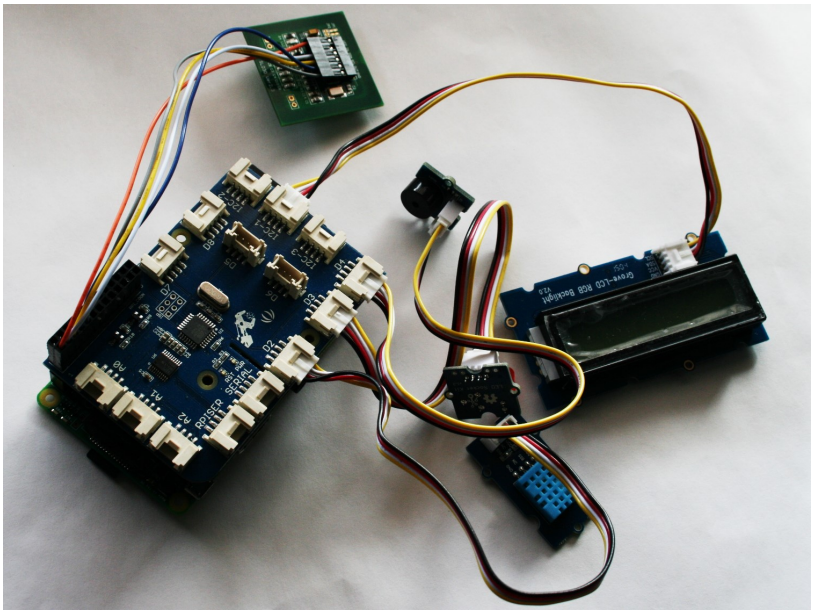

Figura 4.6: Preparazione Raspberry

- Accendere il Raspberry semplicemente collegando il cavo di alimentazione;

Figura 4.7: Avvio del Raspberry

- Una volta caricato il sistema si dovrebbe avere una situazione tipo questa;

Figura 4.8: Desktop Raspberry Pi2 (Dexter Industries Raspbian for Robots)

- Andare quindi ad avviare il nodo TuCSoN sul Raspberry. Posizionarsi nella cartella dove sono contenute le librerie di TuCSoN e scrivere in una nuova finestra del terminale il seguente comando:

  java -cp tucson.jar:2p.jar alice.tucson.service.TucsonNodeService

```
[TuCSoN Node Service]: ------------------------------------------------------------
[TuCSoN Node Service]:  _________              ______   ______          ____  ____
[TuCSoN Node Service]: |  _   _  |          .' ___  |.' ____ \        |_  _||_  _|
[TuCSoN Node Service]: |_/ | | \_|__   _  / .'   \_|| (___ \_|  .--.   |  \ | |
[TuCSoN Node Service]:     | |   [  | | | | |        _.____`. / .'\ \ | |\ \| |
[TuCSoN Node Service]:    _| |_   | \_/ |,\ `.___.'\| \____) || \__. |_| |_\  |_
[TuCSoN Node Service]:   |_____|  '.__.'_/ `.____ .' \_______.' '.__.'|_____|\____|
[TuCSoN Node Service]: ------------------------------------------------------------
[TuCSoN Node Service]: Welcome to the TuCSoN infrastructure :)
[TuCSoN Node Service]:   Version TuCSoN-1.12.0.0301
[TuCSoN Node Service]: ------------------------------------------------------------
[TuCSoN Node Service]: Fri Sep 25 10:04:02 CEST 2015
[TuCSoN Node Service]: Beginning TuCSoN Node Service installation...
[TuCSoN Node Service]: Configuring TuCSoN Node Service...
[TuCSoN Node Service]: Setting up Observation Service...
```

Figura 4.9: Avvio di TuCSoN

```
5-ea75-47f6-8dcd-c00e7a2f1556')), trg: '@'('$ORG',':'(localhost,'20504')), tc: '
@'('$ORG',':'(localhost,'20504')) ] ] ]
..[WelcomeAgent]: Listening to incoming connections...
..[WelcomeAgent]: ------------------------------------------------------------
------------------
....[RespectVMContext ('$ORG'@localhost:20504)]: guard = request
....[RespectVMContext ('$ORG'@localhost:20504)]: evaluation = false
..[ACCProxyNodeSide (20504, 0, '$EnvAgent')]: Listening to incoming TuCSoN agent
s/nodes requests...
....[OperationHandler ('$EnvAgent')]: requesting op 2, cmd(Type), '@'('$ENV',':'
(localhost,'20504'))
....[RespectVMContext ('$ORG'@localhost:20504)]: guard = request
....[RespectVMContext ('$ORG'@localhost:20504)]: evaluation = false
....[RespectVMContext ('$ORG'@localhost:20504)]: guard = request
....[RespectVMContext ('$ORG'@localhost:20504)]: evaluation = false
....[RespectVMContext ('$ORG'@localhost:20504)]: guard = request
....[RespectVMContext ('$ORG'@localhost:20504)]: evaluation = false
..[ACCProxyNodeSide (20504, 0, '$EnvAgent')]: Serving TucsonOperation request <
id=14, type=2, tuple=cmd(Type) >...
..[ACCProxyNodeSide (20504, 0, '$EnvAgent')]: Listening to incoming TuCSoN agent
s/nodes requests...
....[RespectVMContext ('$ENV'@localhost:20504)]: INVOCATION phase: [ src: '$EnvA
gent', op: in(cmd(Type)), trg: '@'('$ENV',':'(localhost,'20504')), tc: '@'('$ENV
',':'(localhost,'20504')) ]
```

Figura 4.10: Caricamento completato di TuCSoN

- Terminato il caricamento si è pronti ad eseguire Home Manager. Per effettuare questa operazione digitare nella cartella dove è presente il file .jar di Home Manager, il seguente comando e attendere il completo caricamento:

  java –jar HomeManager.jar

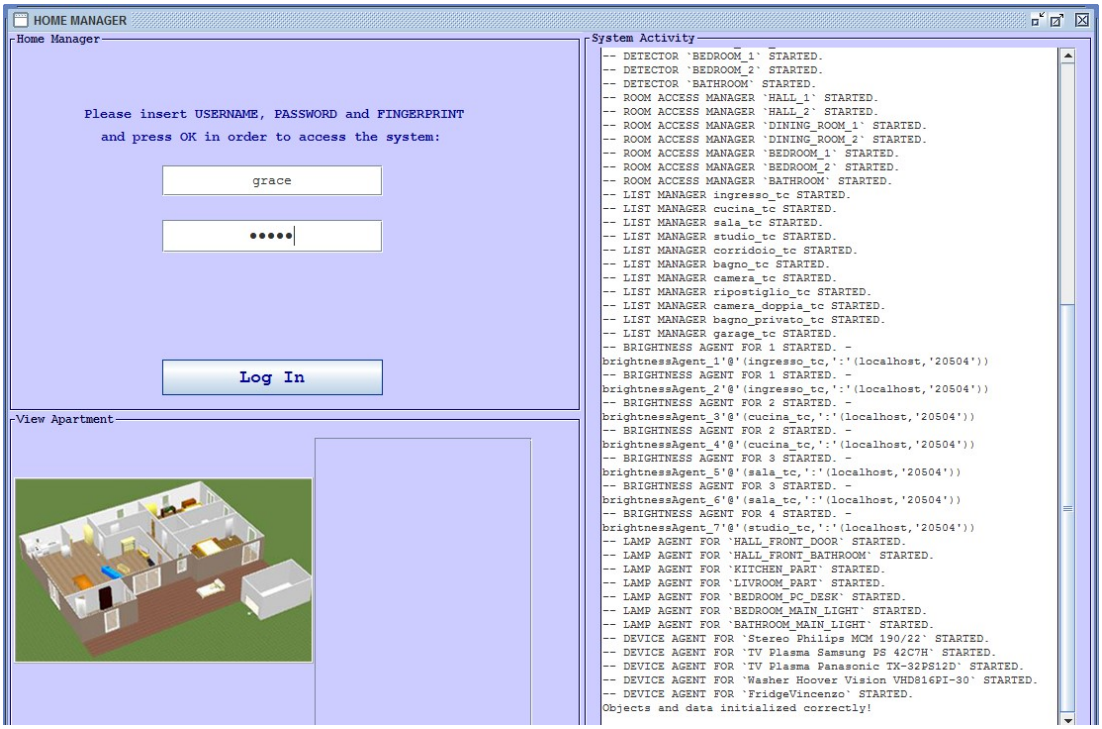

Figura 4.11: Caricamento completato di Home Manager, tutto pronto per il login

- Procedere quindi a fare il login, e ad entrare nella sezione Devices, dove si sceglierà dal menu a tendina il dispositivo Fridge;

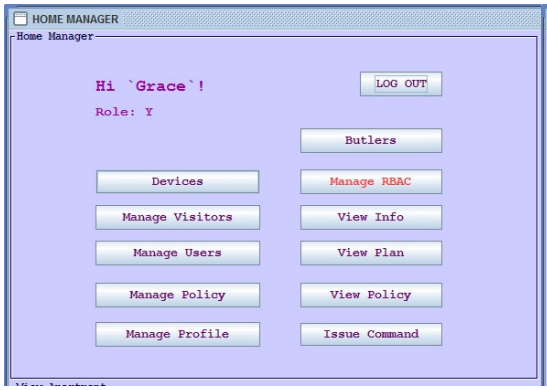

Figura 4.12: Schermata principale HM

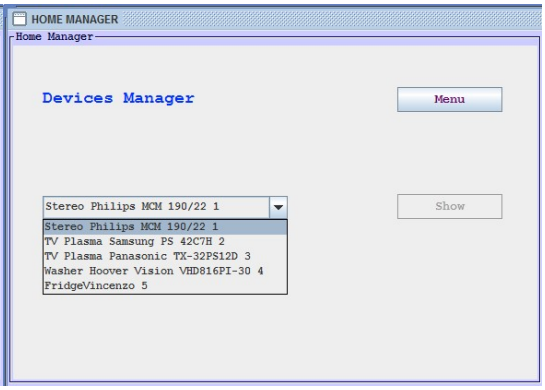

Figura 4.13: Schermata scelta dei dispositivi

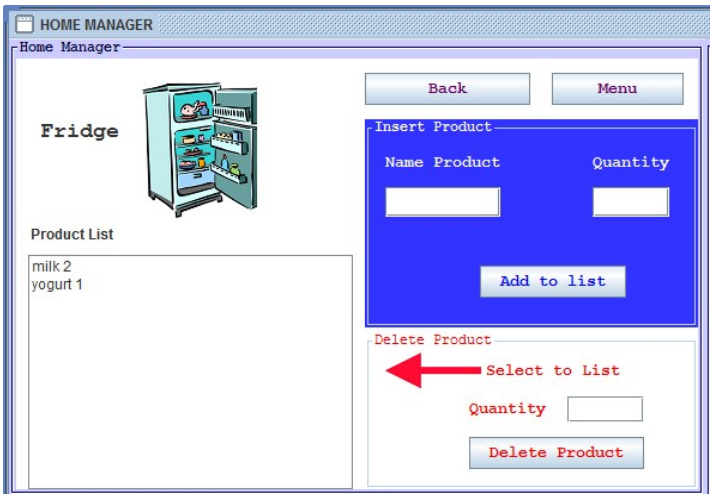

Figura 4.14: Schermata Fridge di Home Manager

- Dopo di che, se si è utilizzato NetBeans con JVM Remota, si ritroverà nella cartella *home/root/NetBeansProjects/NomeProgetto/dist* tutto il programma, che si eseguirà, dopo essersi posizionati nella cartella sopra citata, con il seguente comando:

  sudo java –jar NomeProgetto.jar

- Avvicinando un Tag Rfid di quelli in nostro possesso si noterà che avviene tutto quello che ci si era prefissati di ottenere, ovvero le opportune notifiche dei vari sensori e dispositivi e l'aggiornamento del contenuto del frigorifero nella schermata di Home Manager:

Figura 4.15: Display prodotto in ingresso

Figura 4.16: Display prodotto in uscita

Figura 4.17: Display temperatura frigo

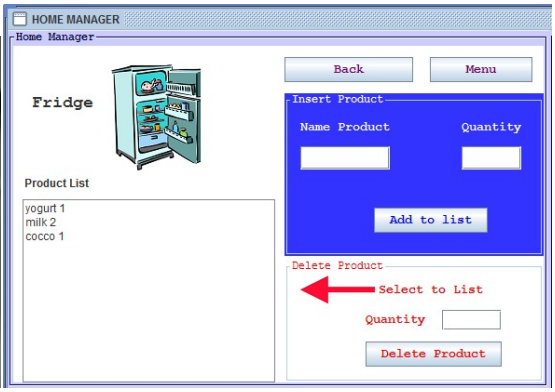

Figura 4.18: Schermata contenuto frigo

Capitolo 5

Conclusioni

Questa tesi aveva come scopo quello di portare il prototipo attuale di Home Manager sulla tecnologia Raspberry, introducendo sensori e moduli per la realizzazione del dispositivo frigo intelligente già presente nella precedente versione.

Lo studio condotto ha permesso di ottenere risultati che dimostrano la fattibilità del progetto realizzato ed offrono spunti per eventuali sviluppi futuri.

In questa tesi si è realizzato un primo esempio di "dispositivo intelligente" in grado di sapere in ogni momento tutti i prodotti in esso contenuti e di tracciare tutti gli spostamenti dei prodotti con cui interagiva.

Lo sviluppo, seppur ancora prototipale, mostra l'efficacia della scelta di un'infrastruttura come TuCSoN, che consente una facile interoperabilità tra i componenti ed una separazione tra l'intelligenza del sistema e quella dei singoli componenti (agenti). Anche la scelta della tecnologia Raspberry si è rivelata pienamente soddisfacente, essa ha conferito autonomia all'esecuzione del prototipo, permettendone il pieno utilizzo senza la necessita di un normale computer.

Diversi miglioramenti e sviluppi futuri possono essere considerati: primo fra tutti una gestione non a polling del lettore Rfid e una gestione non a polling dell'aggiornamento della grafica, possibilmente notificando al sistema quando è stato aggiunta, rimossa o modificata una tupla nel centro di tuple; in modo che il sistema capisca che bisogna effettuare un refresh della grafica.

Inoltre, un altro importante miglioramento, che è già stato impostato in questa tesi, è la separazione Client-Server, ovvero, utilizzare un Raspberry su cui girerà Home Manager che rappresenterà il lato Server e utilizzare un Raspberry con tutti i sensori sul quale girerà solo l'agente TuCSoN scritto qui, che rappresenterà il lato Client. In questo modo, si potrà avere un solo server e tanti client, che potranno rappresentare tanti dispositivi diversi.

Per questo fine, l'agente presente in questa tesi è stato scritto in modo che sia completamente indipendente dal nodo su cui viene avviato Home Manager.

Bibliografia

[1] Tratto da Wikipedia:
https://it.wikipedia.org/wiki/Domotica

[2] Prof. Enrico Denti, Ing. Roberta Calegari, “Home Manager”:
http://apice.unibo.it/xwiki/bin/view/Products/HomeManager

[3] Tratto da Wikipedia:
https://it.wikipedia.org/wiki/Internet_delle_cose

[4] Tratto da Documentazione ufficiale SODA:
http://apice.unibo.it/xwiki/bin/view/SODA/

[5] Tratto da Denti Enrico:
"Novel pervasive scenarios for home management: the Butlers architecture".

[6] Tratto da Documentazione ufficiale TuCSoN:
http://apice.unibo.it/xwiki/bin/view/TuCSoN/

[7] Tratto da Documentazione ufficiale tuProlog:
http://apice.unibo.it/xwiki/bin/view/Tuprolog/

[8] Tratto da Documentazione ufficiale ReSpecT:
http://apice.unibo.it/xwiki/bin/view/ReSpecT/

[9] Tratto da Wikipedia:
https://it.wikipedia.org/wiki/Raspberry_Pi

[10] Tratto da documenti on-line:
http://www.dexterindustries.com/GrovePi/

[11] Tratto da documenti on-line:
http://www.stronglink-rfid.com/it/rfid-modules/sl030.html

[12] Tratto da Wikipedia:
https://it.wikipedia.org/wiki/NetBeans

[13] Tratto da documentazione ufficiale pi4J:
http://pi4j.com/

[14] Tratto da documentazione ufficiale pi4J:
http://pi4j.com/example/control.html

[15] Tratto da documenti on-line:
https://github.com/emoranchel/IoTDevices

[16] Tratto da documenti on-line:
http://www.stronglink-rfid.com/download/SL030-User-Manual.pdf

Ringraziamenti

Eccomi arrivato alla parte più difficile. Sono tante, infatti, le persone che mi sono state accanto in questo percorso e che tengo a ringraziare. Perdonatemi, dunque, se per sbaglio lascerò indietro qualche nome.

Prima di tutto vorrei ringraziare il prof. Enrico Denti, che mi ha dato la sua piena fiducia, permettendomi di lavorare sui suoi preziosi Raspberry e che insieme all'Ing. Roberta Calegari sono stati presenti e disponibili ad aiutarmi in qualsiasi momento.

Vorrei ringraziare i miei genitori che mi hanno permesso di intraprendere questo percorso, soprattutto economicamente, ma anche sostenendo sempre questa mia scelta. Vi ringrazio per l'amore che mi avete dato e per i sacrifici fatti per consentirmi di essere qui in questo momento. Spero di avervi ripagato anche minimamente dei vostri sforzi. Insieme a loro ringrazio mio fratello Francesco e mia sorella Arianna che, anche se per vie secondarie, mi hanno sempre fatto sentire il loro sostegno.

Ringrazio con il cuore la mia ragazza Elena, per l'immenso supporto che mi ha mostrato in questo periodo di ansie e di preoccupazioni, per esserci sempre e comunque nei miei momenti di bisogno e per credere costantemente in me più di quanto non faccia io stesso. Ti ringrazio per aver scelto di camminare al mio fianco.

Ringrazio mia nonna Anna, che con amore si è sempre presa cura del suo "nipotino" e che è sempre stata pronta a tifare per me in qualsiasi circostanza.

Ringrazio il mio amico Federico, con cui ho passato le gioie e i dolori di praticamente tutti gli esami e che inconsciamente è stato stimolo continuo per un mio miglioramento personale.

Ringrazio Marco che mi è sempre stato accanto, disponibile per qualunque cosa avessi bisogno e per il bene che mi vuole.

Ringrazio Alessandro, Davide, Davide, Lorenzo e Michael senza i quali tutto questo percorso non sarebbe stato così divertente. Ringrazio per tutti i momenti che ho

passato con loro; i miei compagni che riuscivano sempre a rendere le lezioni divertenti anche le più impossibili.

Ringrazio anche Alberto, Andrea e Marco che sono stati un po' i miei “google” in ogni necessità e che nonostante tutto, riuscivano a rispondermi con molta gentilezza anche alle mie domande più stupide.

Non ultimi ringrazio Valeria, Sergio e Silvia che hanno sempre creduto in me e che mi hanno allietato le domeniche di studio con pranzi e cene di tutto rispetto. Grazie.

A special thanks goes to Robert Savage, who has devoted his time helping me in this project, although he didn't even know who I was.

Infine ringrazio tutti i miei amici che credono in me e mi supportano e sopportano nella vita di tutti i giorni.

Grazie.