Consiglio Nazionale delle Ricerche

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" (ISTI) - Laboratorio di Domotica

Progetto DAGON
(Domotics, Automation and Orchestration Government Network)

Un unico telecomando per tutti i dispositivi domotici

Vittorio Miori, Gian Felice Meloni

Indice

1. Introduzione

Il lavoro ha previsto la progettazione e lo sviluppo di un software da installare su un telecomando programmabile di ultima generazione (dotato di monitor touch screen e wi-fi), al fine di pilotare e monitorare un ambiente domestico tramite l'interfacciamento con un middleware domotico preesistente in grado di comunicare e far interoperare tutti i dispositivi domotici presenti all'interno di un'abitazione.

Il prototipo creato è parte integrante delle attività svolte per il Progetto DAGON (Domotics, Automation and Orchestration Government Network), il cui obiettivo è lo sviluppo di una piattaforma integrata di servizi avanzati per il controllo di sistemi domotici e di navigazione nell'ambito della motoristica da diporto. Esso è stato testato all'interno della Casa Domotica realizzata dal Laboratorio di Domotica dell'Istituto ISTI del CNR di Pisa, all'interno del quale trovano posto diversi dispositivi intelligenti appartenti a sistemi domotici diversi; l'interoperabilità tra tali dispositivi è garantita dal middleware domoNet, che permette di vedere tutti i dispositivi come facenti parte di un unico sistema domotico.

La Casa Domotica del CNR, riproduce delle ambientazioni domestiche (ingresso, cucina, soggiorno, bagno, camera da letto, giardino) allo scopo di individuare una serie di soluzioni di automazione orientate al miglioramento della qualità della vita degli abitanti.

La scelta delle tecnologie domotiche, appartenenti a standard differenti, non si è basata sulla reciproca compatibilità, in quanto raramente tecnologie domotiche diverse sono interoperabili tra loro. La selezione è stata invece effettuata in base alle potenzialità, alla diffusione e ai costi.

L'integrazione di dispositivi eterogenei è stata resa possibile grazie all'introduzione dell'infrastruttura domoNet, la quale è in grado di astrarre logicamente le caratteristiche di ogni dispositivo in modo da permettere la condivisione di stati ed eventi all'interno dell'installazione domotica.

2. Sistemi domotici utilizzati

Le tecnologie e gli standard utilizzati, sono i seguenti:

- Konnex: (abbreviato con la sigla KNX - norme EN50090 ISO/IEC14543), è uno standard europeo aperto che prevede l'uso di un BUS per veicolare la comunicazione tra i dispositivi installati.
- My Home: tecnologia proprietaria legata al produttore italiano BTicino.
- UPnP: tecnologia aperta nata da un'iniziativa aziendale per connettere dinamicamente dispositivi intelligenti; la comunicazione è stata sviluppata attraverso reti TCP/IP cablate o in radiofrequenza (wireless).
- Altre tecnologie per la gestione delle interfacce e per la comunicazione in Internet (ad esempio Tomcat, Wifi, etc.).

2.1 Il Domotics Lab

Verranno di seguito descritti i vari ambienti simulati all'interno del domotics lab.

2.1.1 Ingresso

Nell'ingresso sono stati realizzati una serie di servizi presenti oramai in quasi tutte le abitazioni:

- apertura della porta d'ingresso - avviene a seguito del riconoscimento dell'impronta digitale dell'utente da parte di un lettore biometrico. Altre funzionalità collegate sono la disattivazione dell'allarme e l'accensione della luce dell'ingresso.
- luce notturna - può essere accesa normalmente tramite un interruttore, in maniera automatica al rientro in casa dell'abitante, oppure in remoto.
- controllo carichi - serve ad impedire che troppi elettrodomestici vengano attivati in contemporanea e provochino il distacco del limitatore per il superamento della soglia dei chilowatt di potenza impegnata imposto dal fornitore di energia

elettrica. Quando il controllo carichi è attivo il sistema osserva l'assorbimento di potenza sulla linea elettrica ed evita il superamento della soglia disattivando momentaneamente gli elettrodomestici meno critici (ad es. lavatrice, lavastoviglie, forno, asciugacapelli, etc.).

- postazione touch screen - Fornisce all'utente una comoda postazione di controllo delle automazioni della casa. Alcune funzioni disponibili sono: attivazione e modifica degli scenari, attivazione e visualizzazione accesso estranei.
- combinatore telefonico - gestisce l'invio di messaggi sms e vocali in caso di allarme rilevato.

2.1.2 Soggiorno

All'interno del soggiorno sono state realizzate funzionalità per soddisfare le esigenze di tutti gli abitanti della casa, anche quelli più anziani e che spesso non hanno familiarità con le tecnologie. Ne citiamo alcune:

- accensione automatica di un punto luce - mediante un rilevatore di presenza a raggi infrarossi viene attivato l'attuatore della luce della camera. Lo spegnimento può avvenire in modo temporizzato o può essere comandato manualmente da una pulsantiera. Quando l'impianto di allarme è inserito, il rilevatore di presenza funziona come rivelatore volumetrico antintrusione.
- controllo temperatura - realizzato tramite un termostato ed un attuatore per la variazione della portata d'acqua in un termosifone. Le funzionalità offerte da tali dispositivi sono la visualizzazione e la modifica (anche da remoto) sia della temperatura attuale, sia delle modalità preimpostate, come ad esempio la modalità confort, antigelo, etc.
- antintrusione - consiste in un contatto magnetico per la simulazione dell'apertura di una finestra. Se l'impianto di allarme è inserito, all'apertura di una finestra verrà segnalata al sistema un'intrusione. Quando invece l'allarme è disinserito, verrà comandato lo spegnimento del riscaldamento (risparmio energetico).
- sistema d'intrattenimento UPnP (Universal Plug and Play) e Mediacenter - i componenti principali del sistema di intrattenimento sono il Mediacenter e un router multifunzione dotato di hard disk. Entrambi i dispositivi utilizzano il

protocollo UPnP per condividere file multimediali tra le varie periferiche dell'abitazione.

- telecamera videosorveglianza - si tratta di una telecamera UPnP collegata alla rete locale che permette di visualizzare, anche da remoto, l'abitazione. Questa funzionalità è particolarmente utile nel caso si verifichi una qualsiasi situazione di allarme; il sistema infatti può rendere disponibile il link della webcam per visionare la scena.

2.1.3 Cucina

Nell'ambiente cucina sono stati simulati i principali elettrodomestici presenti nelle abitazioni moderne: forno elettrico, forno microonde, piastre elettriche per la cottura. Ad essi sono stati affiancati i seguenti servizi:

- impianto di rilevazione di fughe di gas e incendi - i dispositivi utilizzati per rilevare fughe di gas e incendi sono dei sensori KONNEX ai quali sono stati aggiunti interruttori a due posizioni per simulare una fuga di gas o un incendio. Al verificarsi di uno dei due eventi, il sistema produce un messaggio visivo sul touch screen e una segnalazione remota di allarme.

- sistema per la gestione sicura della zona cottura - è particolar- mente adatto a persone soggette a perdite di memoria o facili alle distrazioni. Per accendere un fornello, è necessario premere un pulsante che avvia l'erogazione del gas ed attiva un timer (di 30 minuti). In prossimità dello scadere del timeout, il sistema avvisa che sta per interrompere l'erogazione del gas; se il pulsante non viene premuto nuovamente l'erogazione verrà arrestata. Tale accorgimento riduce sensibilmente il rischio di fughe di gas conseguenti allo spegnimento accidentale in seguito a traboccamento dei liquidi.

2.1.4 Camera da letto

All'interno della camera da letto sono state realizzate le seguenti funzionalità:

- gestione luci - tramite un apposito pulsante è possibile regolare l'intensità

luminosa dell'abatjour posta sul comodino.

- percorso luminoso - mediante un pulsante è possibile attivare una scia luminosa che guida l'utente dalla camera da letto al bagno, senza bisogno di accendere le altre luci della casa.
- gestione scenari - grazie ad una pulsantiera posta vicino al letto si possono impostare due scenari preimpostati: lo scenario notte e lo scenario risveglio. L'impostazione del primo scenario comporta l'attivazione dell'allarme antintrusione, la disattivazione dell'erogazione del gas, e la modifica della temperatura per il comfort notturno. Il secondo scenario apre le tapparelle della casa e disabilita gli allarmi.

2.1.5 Bagno

Gli accorgimenti sviluppati nel bagno sono i seguenti:

- sistema per la rilevazione di perdite d'acqua - si tratta di un rilevatore di allagamento che, in caso di fuoriuscita d'acqua, produce un messaggio di allarme visivo sul touch screen e invia una segnalazione remota tramite il combinatore telefonico.
- sistema di chiamata soccorso - consiste in un pulsante di emergenza a tirante collocato in prossimità della doccia/vasca, attraverso il quale è possibile inviare al sistema una richiesta di soccorso.

2.1.6 Giardino

Nel giardino sono presenti tre dispositivi: la tapparella, la tenda ed un sensore meteo. Queste apparecchiature sono pilotabili sia manualmente, per mezzo di pulsanti, sia da remoto. Un tipico esempio del loro utilizzo è quello relativo agli scenari risveglio, notte, rientro in casa e esco di casa. In base allo scenario impostato avviene in automatico l'apertura/chiusura della tenda e della tapparella. Il sensore meteo, in base ai dati meteo scaricati, decide se aprire oppure chiudere la tenda.

Figura 1: Ambienti del domotics lab

3. Telecomando Pronto Philips TSU9400

Per la creazione dell'interfaccia user-friendly occorreva un dispositivo che sostanzialmente avesse tre caratteristiche: un'interfaccia wifi che consentisse l'utilizzo di socket per comunicare col framework, una dimensione il più possibile compatta e un ampio monitor che consentisse una facile visualizzazione dei menù e una rapida gestione dei dispositivi presenti nell'abitazione.

Per la scelta definitiva del prodotto, si sono confrontati vari apparecchi ognuno dei quali aveva caratteristiche hardware e design differenti. Alla fine la scelta è ricaduta sul telecomando Philips Pronto TSU9400, (figura 4.1), per diverse ragioni.

La prima motivazione è la dotazione hardware che rispetta i requisiti per lo sviluppo del progetto. Infatti, oltre agli infrarossi presenta una connettività wireless a 2.4 GHz, una memoria interna di 64MB, ed una batteria al litio da 1700 mAh.

La seconda motivazione è la tipologia del dispositivo. Il telecomando è un apparecchio compatto e quindi utilizzabile in maniera semplice da qualsiasi tipo di persona. Inoltre l'interfaccia utente risulta familiare in quanto presenta le stesse caratteristiche di un dispositivo presente in tutte le abitazioni.

Figura 2: Pronto Philips tsu9400

3.1 ProntoEdit: l'ambiente di sviluppo

Il ProntoEdit Professional è un editor grafico, disponibile su piattaforma Windows, usato per configurare il pannello di controllo del telecomando e per creare qualsiasi tipo di interfaccia utente. Il software mette anche a disposizione un emulatore per il test e il debug delle interfacce realizzate, ma non offre invece alcun supporto per i socket. In questo caso l'effettivo funzionamento di un programma deve essere verificato direttamente sul dispositivo.

La programmazione di un'interfaccia può avvenire sia per mezzo di strumenti grafici, col classico drag and drop di bottoni, sfondi e icone, oppure mediante un linguaggio di scripting col quale è possibile definire comportamenti e funzionalità specifiche. L'editor è mostrato in figura 3.

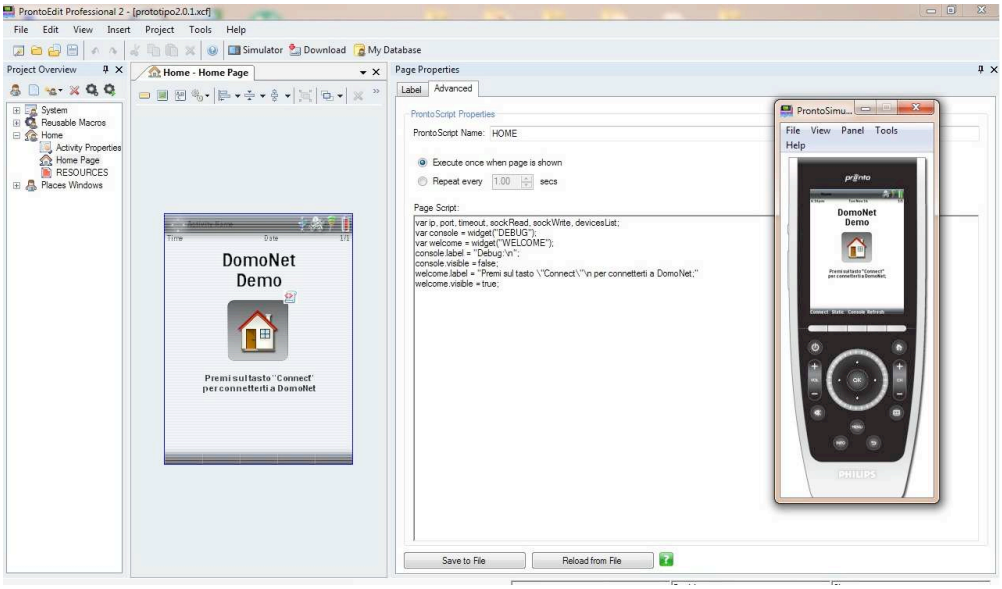

Figura 3: Pronto Edit Professional

3.2 ProntoScript: il linguaggio di scripting

Il core principale del telecomando deriva dal linguaggio javascript. Da esso infatti si ereditano le variabili, gli operatori, i blocchi di dichiarazioni, i controlli di flusso, le funzioni, le eccezioni, gli oggetti, e le cosiddette built-in classes (Array, Boolean, Math, etc.).

Il core iniziale è stato arricchito con ulteriori funzionalità che estendono le potenzialità del telecomando. Tra esse ricordiamo:

- Widgets - sono degli oggetti grafici che possono essere manipolati da uno script

per creare una user-interface dinamica.

I widget più ovvi sono i bottoni e i pannelli (visibili e manipolabili dal monitor touchscreen), e i cosiddetti hard buttons, ovvero i bottoni reali esistenti sul telecomando.

- Action List - per tutti i widgets può essere definita una lista di azioni (ad esempio si possono eseguire salti di pagine, esecuzione di suoni, ecc.).
- Timer - meccanismi utilizzati per far ritarare l'esecuzione di alcuni script.
- Level e Lifetime - Un file di configurazione consiste in una gerarchia di attività: ciascuna di esse è composta da un certo numero di pagine, che a loro volta presentano bottoni e pannelli. La gerarchia di livelli è: attività, pagine, bottoni. Il lifetime di uno script è il tempo in cui una funzione, una variabile o una classe rimangono definiti dopo la loro dichiarazione. Questo dipende dal livello in cui l'oggetto è dichiarato.
- Attività e Pagine - Le attività sono degli script utilizzati per definire oggetti, funzioni, variabili e tutti i parametri che verranno utilizzati all'interno delle pagine. Le pagine invece sono degli script eseguiti nell'istante prima che una pagina stia per essere visualizzata sullo schermo.
- Comunicazione di rete - Il telecomando è in grado di comunicare con altri dispositivi tramite wi-fi o ethernet. Può utilizzare sia connessioni TCP, sia connessioni UDP.

3.3 Le connessioni TCP

Nel linguaggio ProntoScript una connessione TCP può essere stabilita mediante la classe TCPSocket. Questa classe usa connessioni differenti dagli altri linguaggi di programmazione.

Il protocollo TCP è stato creato basandosi sulle connessioni seriali ed è per questo che i socket di rete possono essere usati sia in modo sincrono, sia in modo asincrono. Nella modalità sincrona lo script verrà bloccato per tutta la durata di una qualsiasi operazione sui socket. Nella modalità asincrona invece, sono messe a disposizione delle callback chiamate al completamento di un'operazione sui socket, e lo script continuerà normalmente la sua esecuzione. Una tipica connessione TCP sincrona può essere realizzata mediante lo script:

```
var socket = new TCPSocket(true);
socket.connect('google.com', 80, 3000);
```

```
socket.write("GET / HTTP/1.0\r\n\r\n");
label = socket.read(100, 3000);
socket.close();
```

Una connessione TCP asincrona viene realizzata nel seguente modo:

```
var socket, result;
socket = new TCPSocket(false);
result = "";
socket.onConnect = function()
  {
  write("GET / HTTP/1.0\r\n\r\n");
  };
socket.onData = function()
  {
  result += read();
  };
socket.onClose = function()
  {
  label = result;
  };
socket.connect('google.com', 80, 3000);
```

Come è visibile dai sorgenti sopra riportati, il telecomando non è in grado di funzionare da server, cioè non può rimanere in attesa di connessioni da parte di un'entità remota. Può soltanto svolgere il ruolo di client e quindi intraprendere connessioni per ricevere e inviare dati. Questa caratteristica dei socket è derivante dalla natura del dispositivo. Essendo esso un dispositivo mobile, e con alimentazione a batteria, l'intento principale è quello di scaricare dati e chiudere immediatamente le connessioni per preservarne l'autonomia.

Dato che il fulcro dell'intero progetto era proprio l'utilizzo delle connessioni, si è pensato di creare un programma client-server minimale per testare l'efficienza dei socket. Il lato client è stato implementato nel telecomando, mentre il lato server è stato scritto in linguaggio java. L'obiettivo era lo scambio di semplici stringhe.

Sono subito emerse diverse problematiche sulla comunicazione mediante i Socket TCP. Il primo malfunzionamento riscontrato si è manifestato sia con i socket sincroni che con quelli asincroni. Il client riesce con successo a stabilire una connessione col server, ma non riesce a scambiare dati. In particolare il server non riceve i dati inviati dal telecomando finché questo non chiude la connessione con la funzione close(). Lo

stesso si verifica quando è il server a scambiare dati col client.

Dopo vari tentativi e consulti con la casa produttrice, si è arrivati alla conclusione che connessioni di tipo full-duplex non potevano essere realizzate (cioè non è previsto lo scambio contemporaneo di dati tra le due entità).

L'unica soluzione trovata, in attesa che la Philips aggiorni l'attuale firmware, è stata quella di utilizzare i socket in modalità half-duplex : ciascuna entità utilizza i socket o per leggere un dato, o per scriverlo. Grazie a questo accorgimento è stato possibile lo scambio di dati tra telecomando e server.

In questo scenario una normale connessione TCP in cui un client scrive un comando sul server e legge il risultato deve necessariamente avvenire in due connessioni distinte. Nella prima connessione avverrà la scrittura del comando; nella seconda la lettura del risultato. In maniera duale anche il server dovrà comportarsi allo stesso modo: durante la prima connessione esso leggerà il comando; durante la seconda connessione esso scriverà sul socket il risultato. La figura 4 rappresenta lo scenario appena descritto.

Ovviamente affinché la soluzione scelta funzioni, nell'implementazione del lato server occorre specificare due stati: uno stato per eseguire la lettura di un comando, ed uno stato per scrivere il messaggio sul socket. L'idea di stato può essere realizzata con una variabile booleana che indica se ci si trova nella modalità scrittura oppure nella modalità lettura.

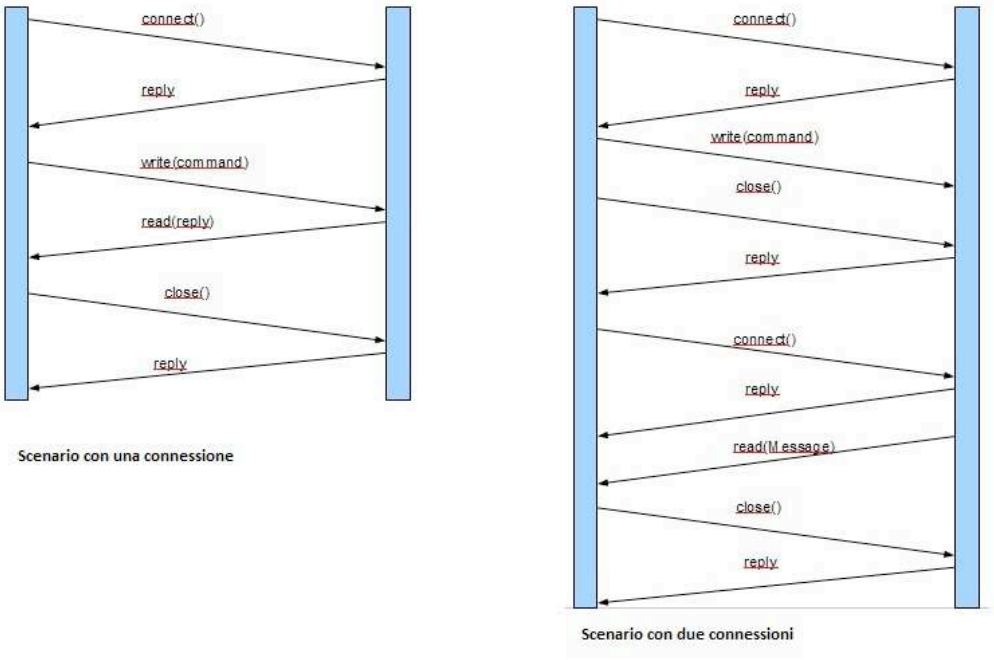

Figura 4: Connessione normale e connessione modificata

4. Analisi dei requisiti

L'obiettivo principale da raggiungere è quello di realizzare un software sul telecomando che permetta di controllare e visualizzare i dispositivi presenti nel domotics lab interfacciandosi col framework domoNet. Un obiettivo secondario, ma non di minor importanza, è quello di creare un'interfaccia grafica intuitiva che mostri i vari ambienti di un'ipotetica abitazione e offra una serie di funzionalità che siano utilizzabili da qualsiasi categoria di utente.

I punti cruciali su cui progettare l'architettura complessiva sono pertanto due:

- trovare un modo efficiente per far dialogare il telecomando con il framework domoNet. Poiché il dispositivo in dotazione è un dispositivo mobile e con limitate capacità harware, lo scambio di messaggi col sistema deve essere semplice e conciso.

- creare un'interfaccia grafica di facile comprensione e quanto più dinamica possibile; l'intento è quello di creare uno script sul telecomando che sia portabile, cioè che sia in grado di auto-configurarsi in qualsiasi ambiente dove sia presente l'architettura domoNet. Anche questo aspetto non risulta banale in quanto sia gli ambienti, sia i dispositivi presenti nelle abitazioni, sono generalmente eterogenei e quindi con caratteristiche interne profondamente differenti.

4.1 Ipotesi sulle possibili implementazioni

Dopo un'analisi accurata dell'intero sistema, si è immediatamente pensato a come il telecomando potesse interagire con esso. Per far questo ci si è soffermati sul modo in cui gli altri dispositivi dialogano col framework e su come cooperano tra loro.

Ricordiamo che ciascun dispositivo è connesso a domoNet tramite un gateway creato ad-hoc. Ogni gateway permette l'invio di messaggi, dal dispositivo al framework e viceversa, mediante un meccanismo di traduzione dal linguaggio del dispositivo al linguaggio DomoML.

La cooperazione tra i diversi dispositivi è implementata intrinsecamente all'interno del linguaggio domoML. Nel formalismo di un domoDevice è infatti presente il tag linkedService che permette di modificare lo stato di altri dispositivi. Possiamo affermare che la cooperazione è unidirezionale, in quanto un dispositivo può comunicare un comando ad altri dispositivi, ma l'esito di tale comando non gli verrà mai riscontrato. Il riscontro lo conoscerà soltanto domoNet che è l'unica entità in grado di avere una

visione globale dello stato di tutti i dispositivi connessi.

Da questo scenario emergono le prime analogie e differenze tra il telecomando e agli altri sistemi domotici presenti.

In analogia con essi, il telecomando condivide la caratteristica di essere un dispositivo mobile; nella fase di progettazione si dovrà tener conto delle sue limitazioni hardware.

E' invece il ruolo del telecomando ad essere differente: oltre a poter manovrare i dispositivi connessi, esso deve conoscere costantemente il loro stato. Essendo domoNet un'applicazione di rete, il controllo sui dispositivi può avvenire non solo tramite pulsantiere, ma anche da remoto mediante interfacce web, cellulari, ecc.

Per chiarire meglio il problema supponiamo che un utente si trovi nel soggiorno e, agendo sul telecomando, voglia spegnere la luce della camera da letto. Contemporaneamente un altro abitante della casa passa per la camera da letto e, accorgendosi che la luce è accesa, decide di spegnerla. Se al primo utente non viene comunicato che lo stato della lampadina è cambiato, egli azionerà sull'interfaccia il pulsante per spegnere la lampadina, ottenendo come risultato una nuova accensione della stessa.

Per evitare questo tipo di inconvenienti è necessario mantenere una connessione diretta con domoNet per ricevere eventuali aggiornamenti sullo stato dei dispositivi. La prima scelta implementativa è pertanto decidere in che modo collegare il telecomando col framework.

Inizialmente si è pensato di collegare il telecomando direttamente con domoNet usando l'interfaccia dei web service. Questo approccio avrebbe avuto come vantaggio il fatto che il nostro dispositivo e il framework avrebbero parlato la stessa lingua, ovvero avrebbero comunicato direttamente con il linguaggio domoML. Purtroppo tale soluzione è stata scartata per diverse ragioni.

La ragione più ovvia è che il telecomando, per controllare tutti i dispositivi presenti e visualizzarne lo stato sul display, dovrebbe conoscere i domoDevice di ciascun dispositivo e dovrebbe creare opportuni domoMessage per pilotarli. Questo comporterebbe un grande dispendio in termini di spazio e risorse computazionali. Occorrerebbe infatti una memoria abbastanza grande per contenere tutti i file di configurazione e un processore che consenta di elaborare e creare file XML rapidamente. Il telecomando, anche se presenta un parser per manipolare file di tipo XML, non consente il salvataggio di nessun tipo di file all'interno della memoria di massa; risulterebbe ovviamente impossibile tenere nella memoria principale tutti i file di configurazione dei dispositivi.

La seconda motivazione che ha fatto scartare il primo approccio è che domoNet e

tutte le funzionalità da esso offerte, risiedono all'interno dei webService. L'impossibilità per il telecomando di connettersi ad un servizio web risiede nel fatto che le connessioni offerte dal dispositivo sono quelle TCP e UDP standard; queste connessioni rappresentano il livello più basso dello stack protocollare dell'architettura dei web service. Per dialogare in maniera diretta con i web service si sarebbero dovuti implementare tutti i protocolli di rete dei livelli superiori, ovvero nell'ordine: HTTP, SOAP, XML-RPC, WSDL, UDDI, etc.

Nemmeno la casa produttrice si è preoccupata di creare librerie per creare connessioni http. Probabilmente tali librerie non sono state realizzate perché esulavano dagli scopi primari del telecomando. Solitamente dispositivi come questo controllano le altre apparecchiature mediante l'utilizzo dei raggi infrarossi. La funzionalità wireless è soltanto una caratteristica aggiuntiva.

L'approccio scelto è allora consistito nella realizzazione di un gateway posto tra il telecomando e domoNet. Questa scelta si è rivelata vincente per molte ragioni. Quella più ovvia è la più equa ripartizione del carico di lavoro del telecomando. Il suo ruolo si limita infatti alla visualizzazione e modifica dei dispositivi. La connessione al framework e la notifica degli update da parte dei dispositivi viene svolta dal gateway che possiede invece capacità di calcolo superiori.

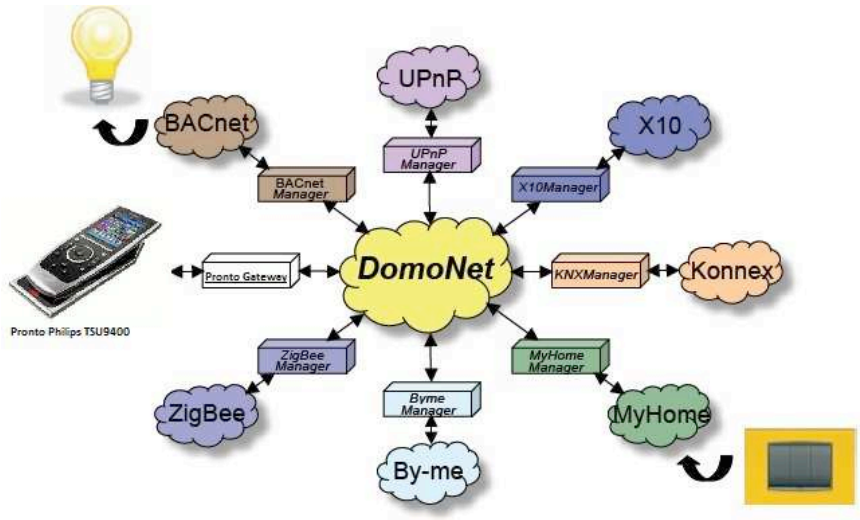

Figura 5: Architettura complessiva

4.2 Primo sguardo all'architettura

L'architettura complessiva è mostrata nella figura 5. Oltre a domoNet, le entità centrali sono il telecomando e il gateway. Tale struttura è in perfetta armonia col framework, nel senso che la maggior parte del carico di lavoro viene svolto dall'intermediario come per tutti gli altri dispositivi del laboratorio.

Il gateway, connesso sia al telecomando, sia a domoNet, avrà un duplice ruolo: servirà da server nei confronti del dispositivo, e servirà da client nei confronti del framework. In particolare esso dovrà adempiere ai seguenti compiti:

- scaricare da domoNet la lista dei dispositivi connessi e i loro file di configurazione (DomoDevice);
- comunicare al telecomando la lista dei dispositivi disponibili nel sistema;
- rimanere in attesa di comandi da parte del telecomando;
- tradurre i comandi ricevuti dal telecomando in formalismo DomoMessage e inoltrarli a domoNet ;
- rimanere in attesa di aggiornamenti, comunicati da domoNet, sul cambiamento di stato di qualche dispositivo;
- inoltrare gli aggiornamenti riscontrati al telecomando.

Il telecomando invece comunicherà soltanto col gateway ; l'interazione prevederà:

- lo scaricamento della lista dei dispositivi connessi, indispensabile per la creazione del menù principale dell'interfaccia grafica;
- lo scaricamento dei file di configurazione di ciascun dispositivo, per la creazione dei pulsanti per modificare lo stato del dispositivo;
- l'inoltro di comandi per manipolare lo stato dei dispositivi su richiesta dell'utente;
- la richiesta di aggiornamenti sul cambiamento di stato dei dispositivi, per aggiornare l'interfaccia grafica.

Il diagramma di sequenza della figura 6 mostra le interazioni che avverranno tra il telecomando, il gateway e domoNet.

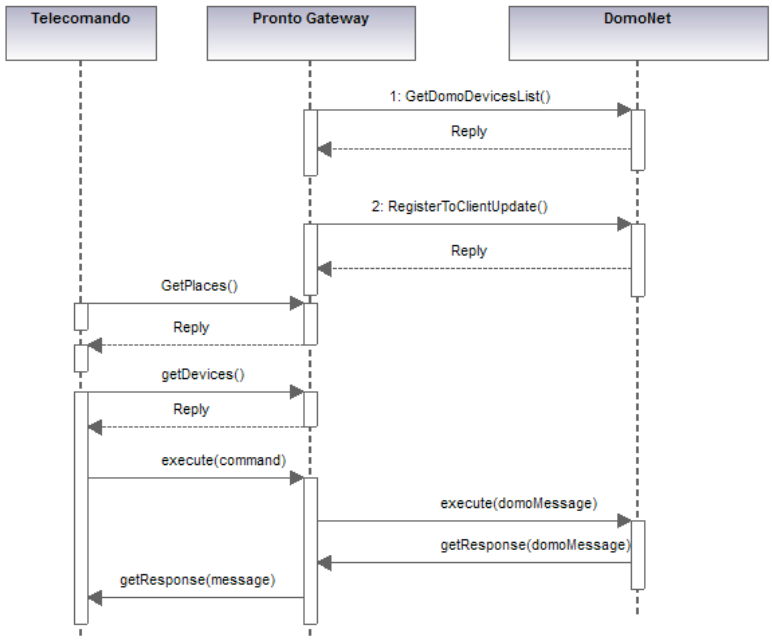

Figura 6: Diagramma sequenziale interazione telecomando-gateway-domoNet

5. Implementazione: il Gateway

5.1 Strumenti di lavoro

Il gateway è stato scritto interamente in Java e usa esclusivamente strumenti open source :

- le librerie web service axis per dialogare con i web service;
- la libreria per la gestione dell'XML Xerces ;

Il progetto è stato sviluppato nell'ambiente Gedit fornito dal sistema Debian ed è stato compilato e testato col build tool Ant, fornito da Apache.

5.2 Classi sviluppate

5.2.1 Device

La classe Device rappresenta un generico dispositivo domotico connesso a domoNet. Consiste in una coppia di oggetti: (DomoDevice, DomoDeviceId). Il primo contiene la descrizione astratta del dispositivo, ovvero la descrizione, il tipo di servizi offerti, i linked services, il tipo di input e output ammissibile, ecc. Il secondo serve a reperire le informazioni necessarie per inoltrare i messaggi al corretto web service: ad esempio l'Id univoco del dispositivo e la URL del servizio web coinvolto.

5.2.2 DevicesInPlace

La classe DevicesInPlace è un vettore che contiene tutti i dispositivi di un determinato ambiente, (ad esempio l'ambiente Cucina conterrà i dispositivi forno, forno elettrico, lavastoviglie, etc.).

5.2.3 WSAccessMethods

La classe WSAccessMethods contiene una serie di funzionalità necessarie per connettersi ai web services presenti in domoNet. Per la sua realizzazione si sono utilizzate le librerie fornite da apache axis per il linguaggio java. I metodi implementati sono essenzialmente quattro:

- `callWebServicesMethod(final String URL,final QName returnType, final String method, Object[] params)` - è un metodo che effettua una chiamata remota al web service identificato dalla stringa URL. Il tipo di valore di ritorno della chiamata remota, il metodo da eseguire e i parametri da passare al metodo, sono specificati nei parametri: `returnType, method, params`.

- `execute(DomoMessage domoMessage)` - è un metodo che esegue un comando specifico all'interno di un web service espresso secondo la sintassi DomoML di un domoMessage. Il valore di ritorno di tale chiamata è una stringa che può assumere soltanto due valori: `SUCCESS`, in caso di successo, `FAILURE`, in caso di fallimento.

- `registerToClientsUpdate()` - è un metodo che informa un web service che si ha a disposizione un socket TCP sul quale ricevere gli aggiornamenti dei dispositivi che hanno cambiato il proprio stato. La porta del socket creato non viene passata come parametro perché questa informazione risiede in un file di configurazione a parte.

- `getDomoDevicesList(final String URL)` - è un metodo che si connette al web service di indirizzo URL e richiede la lista di tutti i dispositivi connessi. La lista ricevuta sotto forma di stringa, viene convertita in linguaggio XML e il file risultante viene analizzato secondo la sintassi del domoML. Dal file verranno estratti i DomoMessage dei dispositivi che verranno inseriti in una Hash Table. Quest'ultima avrà come chiave l'identificativo univoco di ciascun dispositivo, e come valore l'oggetto Device discusso precedentemente. La scelta di utilizzare una tabella hash per memorizzare i dispositivi deriva dal tempo medio costante di ricerca e accesso.

5.2.4 ThreadUpdate

La classe ThreadUpdate implementa un thread che rimane in ascolto dei messaggi di aggiornamento da parte di domoNet. La sua principale funzionalità è quella di mantenere un meccanismo di caching dei DomoMessage provenienti dal framework. La necessità della memoria Cache deriva dal fatto che il telecomando non è sempre connesso al gateway.

Un esempio può essere la funzionalità standby : quando il telecomando non viene utilizzato, oltre all'oscuramento dello schermo, tutte le connessioni pendenti vengono terminate(questo accorgimento è utilizzato principalmente per aumentare l'autonomia del dispositivo). Sarà compito del telecomando intraprendere connessioni col gateway per avere aggiornamenti sullo stato dei dispositivi.

Il meccanismo di cache quindi, si occupa di tener traccia soltanto degli aggiornamenti più recenti, scartando invece quelli più vecchi. Gli update verranno inoltrati al telecomando uno per volta e solo su esclusiva richiesta del dispositivo. Una volta che il thread è sicuro che il messaggio è arrivato con successo al telecomando, lo rimuove definitivamente dalla cache.

5.2.5 ProntoUpdate

La classe ProntoUpdate implementa un thread in attesa di connessioni da parte del telecomando. Il suo unico ruolo è quello di inviare, uno per volta, gli update al dispositivo. Gli aggiornamenti verranno prelevati dalla cache condivisa, discussa nella sezione precedente.

I messaggi vengono spediti in base ad un identificativo specificato dal telecomando. Tale identificativo rappresenta uno degli ambienti della casa. Ad esempio, se il telecomando vuole gli aggiornamenti dei dispositivi presenti nella Cucina, che ha identificativo 2, invierà al thread un messaggio con scritto 2. Questa scelta si è resa necessaria per le limitazioni del linguaggio scripting del telecomando.

5.2.6 GatewayMain

La classe GatewayMain è la classe principale del gateway.

La fase di inizializzazione del sistema si preoccupa di controllare la raggiungibilità del web service. Successivamente richiede a domoNet la tabella hash contenente la lista dei dispositivi presenti utilizzando il metodo getDomoDevicesList(). Una volta ottenuti i DomoDevice dei dispositivi, li riordina per ambiente e mantiene tale ordinamento in

una struttura ausiliaria. Infine crea i thread ThreadUpdate e ProntoUpdate per gestire il meccanismo degli aggiornamenti.

Terminata la fase di inizializzazione, il thread rimane in attesa di istruzioni da parte del telecomando. I comandi disponibili sono:

- `getPlaces` - restituisce l'elenco degli ambienti disponibili all'in- terno dell'abitazione nel seguente formato: `place:id1,...,idN;`. Tale stringa viene utilizzata dal telecomando per generare dinamicamente i menù per visualizzare gli ambienti e i dispositivi di ciascun ambiente;

- `getDevice,idDevice` - restituisce il domoDevice, convertito opportunamente in stringa, del dispositivo avente come identificativo idDevice. Questa informazione verrà utilizzata dal telecomando per generare pulsanti adeguati ai servizi offerti dal dispositivo richiesto.

- `execute:id,service,output,value` - il risultato di tale istruzione consiste nell'invocazione del servizio service sul device id. Se il campo value è vuoto, il comando equivale alla richiesta di reperire l'attuale stato del dispositivo; se value non è vuoto, il valore contenuto verrà impostato come stato attuale del dispositivo. Il campo output rappresenta il formato in cui deve essere creato il messaggio di ritorno. L'output varia in base alla complessità del dispositivo. Ad esempio, se il dispositivo è una normale lampadina, l'output sarà un BOOLEAN: 0 oppure 1 (accesa/spenta). Se il dispositivo è invece una lampadina con intensità luminosa regolabile, il valore di output sarà un BYTE(0-255).

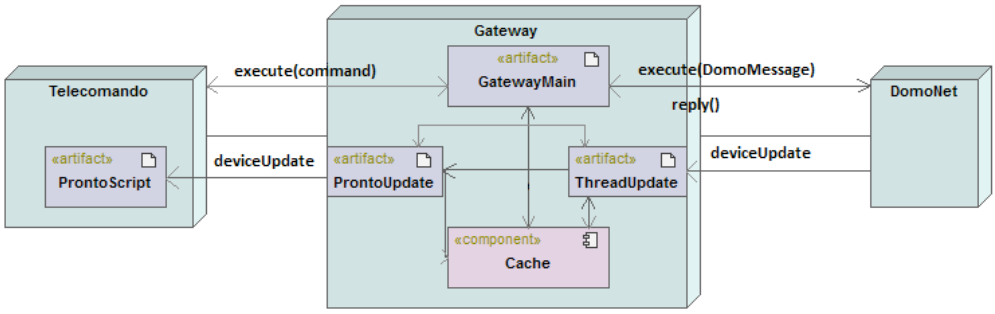

Figura 7: Vista di deployment dell'architettura realizzata

6. Implementazione: lo script del Telecomando

Per lo sviluppo dello script sul telecomando e per la realizzazione dell'interfaccia grafica si sono dovuti valutare diversi aspetti legati principalmente alle caratteristiche del dispositivo. La prima scelta fondamentale è stata la tipologia di connessioni da utilizzare per comunicare col gateway e il tipo di dati da scambiare. La seconda scelta ha riguardato la strutturazione dei menù e delle interfacce da fornire all'utente per monitorare i dispositivi domotici dell'abitazione.

Entrambe le problematiche verranno discusse nei paragrafi successivi.

6.1 Connessione al gateway

La scelta sul tipo di connessioni da utilizzare è stata dettata dal tipo di infrastrutture usate dal framework. Ricordiamo che domoNet funziona per mezzo dei web service che utilizzano connessioni TCP sulla porta 8080. Tale soluzione deriva dal principale vincolo che deve avere un sistema domotico: il trasporto affidabile dei dati. Non deve cioè essere possibile che, durante una comunicazione, alcuni messaggi vengano persi. Per capire l'importanza di questo aspetto basti pensare al verificarsi di una fuga di gas all'interno di un'abitazione. Se un messaggio di allarme andasse perso, tutti gli abitanti della casa correrebbero un grave rischio.

Chiarito il protocollo di comunicazione da impiegare rimane da capire quali funzioni utilizzare. Le API del ProntoScript forniscono due metodi per realizzare connessioni TCP: TCPSocket(TRUE), TCPSoc- ket(FALSE). Il primo metodo permette una comunicazione sincrona, il secondo una comunicazione asincrona. Entrambe le tipologie hanno dei vantaggi e degli svantaggi.

I socket sincroni hanno il vantaggio di essere più veloci ed efficienti, ma hanno lo svantaggio di interrompere lo script finché tutte le connessioni non sono terminate. Questo si ripercuote in un impatto visivo negativo nei confronti dell'utente che utilizza l'applicazione. Si pensi ad esempio ad un utente che preme un tasto sul display per accendere una lampadina. La pressione del tasto comporta l'instaurazione di una connessione al gateway per eseguire il comando di accensione. Utilizzando connessioni sincrone il display rimane bloccato dando l'impressione che il dispositivo abbia smesso di funzionare, mentre in realtà esso sta comunicando col sistema. Al termine della

connessione il display riprende ad aggiornarsi normalmente.

I socket asincroni, invece non interrompono lo script durante una connessione. Quindi, ripercorrendo l'esempio precedente, quando l'utente preme il tasto, lo schermo continua ad essere aggiornato, senza che si verifichi la sgradevole sensazione che il dispositivo si sia bloccato. Questo tipo di socket è però difficoltoso da gestire. Essendo asincrono, l'unico meccanismo possibile per verificare il corretto andamento di una comunicazione è quello delle callback. Sono infatti disponibili le proprietà ON connect, ON close, ON data, ON error, per gestire rispettivamente l'avvenuta connessione, la chiusura di una connessione, l'arrivo di dati e il verificarsi di un errore sulla trasmissione o ricezione di dati.

A causa delle precedenti considerazioni si è pensato di utilizzare le due tipologie di socket in due contesti differenti. In situazioni dove non interviene un'interazione con l'interfaccia grafica del dispositivo, si utilizzano i socket sincroni. In contesti di interazione con l'interfaccia, ad esempio tramite pressione di tasti, si utilizzeranno invece i socket asincroni.

6.2 Creazione dell'interfaccia grafica

Ricordiamo le caratteristiche principali di uno script creato mediante l'ambiente Pronto edit professional.

Uno script è composto da attività e pagine. All'interno di un'attività si trovano le dichiarazioni di tutte le variabili e degli eventuali oggetti di interesse per il funzionamento dello script. Dentro un'attività risiedono una o più pagine. Una pagina è un raccoglitore di Widget grafici che rappresentano ciò che l'utente visualizzerà sul display.

Come prima analisi il linguaggio offre un'ottima strutturazione dello script dividendolo in base alle funzionalità offerte da ogni attività e alle pagine in cui visualizzare la parte grafica. Purtroppo presenta una grave anomalia che riguarda la visibilità delle variabili: non è possibile infatti dichiarare variabili che siano visibili in più attività. E' possibile soltanto la dichiarazione di variabili all'interno di un'attività, le quali saranno visibili esclusivamente nelle pagine che ne fanno parte.

Un'altra funzionalità discutibile riguarda la creazione degli oggetti grafici. Un widget può essere creato in modo statico o dinamico. La creazione statica prevede una preallocazione dell'oggetto e una futura modifica dei parametri. Per la creazione di un oggetto dinamico invece, c'è sempre bisogno di un oggetto statico da cui reperire le funzionalità, come ad esempio il comportamento durante la pressione di un pulsante,

oppure semplicemente il tipo di immagini da attribuire all'oggetto. In sostanza anche l'oggetto dinamico è in realtà un oggetto statico.

Entrambe le peculiarità del linguaggio hanno fatto ridimensionare l'obiettivo di creare uno script che, connettendosi a domoNet, riuscisse a creare dinamicamente la grafica dei menù e dei bottoni dei dispositivi.

Si è invece deciso di creare oggetti grafici statici standard da modificare a posteriori, dopo essersi connessi al framework.

6.3 Interfaccia principale

Come visibile in figura 8 il progetto è strutturato in due attività: l'attività HOME e l'attività PLACES. Nella prima compare il menu di benvenuto, mentre nella seconda compaiono il menù delle stanze della casa e l'elenco dei dispositivi disponibili per ciascun ambiente.

Figura 8: Interfaccia grafica

Nell'attività HOME, tramite la pressione del tasto Connect, si intraprende una prima connessione a domoNet, mediante la quale il telecomando scarica la lista degli ambienti e dei dispositivi connessi. La stringa scaricata ha la seguente forma:

`place1:id1,..,idN;place2:di1,..,idN;..;placeN:id1,..,idN.`

Lo script prosegue effettuando il parsing della stringa ottenuta e generando dinamicamente un menù con gli ambienti dell'abitazione e altre N pagine, una per ambiente. Il menù degli ambienti è accedibile premendo il pulsante Home. Dal menù degli ambienti è infine possibile eseguire il jump al menù specifico di ciascun ambiente.

6.4 Interfaccia specifica di un ambiente

L'interfaccia specifica di un ambiente sostanzialmente è una pagina che raccoglie tutti i dispositivi dell'ambiente selezionato e tramite oggetti grafici, permette il loro controllo. La generazione della pagina prevede una fase di inizializzazione in cui lo script richiede il file di configurazione di tutti i dispositivi presenti nell'ambiente.

Entrando maggiormente nel dettaglio, nello script vengono eseguite le seguenti azioni:

- dalla lista dei dispositivi precedentemente scaricata, si estrapolano gli identificatori dei dispositivi dell'ambiente corrente;
- mediante connessioni sincrone vengono scaricati, uno per volta, i DomoDevice di ciascun dispositivo;
- dai DomoDevice ottenuti vengono estrapolate le informazioni necessarie per la creazione dei widget (ad esempio il tipo di servizi offerti dal dispositivo) le quali verranno salvate in opportune strutture dati;
- in base ai servizi di ciascun dispositivo vengono generati i bottoni per il controllo e la modifica dello stato.
- mediante connessioni sincrone viene scaricato lo stato iniziale di ciascun dispositivo presente;
- lo stato dei dispositivi viene visualizzato sul display.

6.5 Invio di un comando al Gateway e notifica di un Update

Nell'interfaccia sono presenti oggetti grafici che permettono di manipolare lo stato di un dispositivo. Ciascun dispositivo presenta uno o più bottoni, in base ai servizi offerti. Per semplificare la spiegazione dell'invio di un comando verrà utilizzato come esempio l'accensione di una luce.

Una lampadina normale offre due servizi: un servizio per l'accensione e lo spegnimento, e un servizio per conoscerne lo stato (On/Off). Gli oggetti grafici essenziali sono un bottone per il primo servizio e un label per il secondo servizio.

Alla pressione del bottone sul display, viene instaurata una connessione asincrona col gateway.

Durante la comunicazione viene inviato il comando:

```
execute,id,service,value;.
```

Il comando in questione informa il gateway che il servizio service (On/Off) del dispositivo di identificativo id, deve assumere il valore posto nella variabile value.

L'esito del comando appena inviato viene inoltrato dal gateway al telecomando sotto forma di stringa e può assumere soltanto due valori: `SUCCESS` oppure `FAILURE`.

E' da notare che si conosce l'esito del comando, ma non l'attuale valore da attribuire al dispositivo. Tale valore verrà invece notificato da un altro meccanismo: l'Update.

A prima vista questa scelta può sembrare discutibile, ma in realtà essa è un'implementazione coerente col sistema. Il responso del comando viene espresso in termini di successo o fallimento perché, a basso livello, l'invio di un comando non è altro che l'inoltro di un segnale su un bus. Il gateway e conseguentemente domoNet sono a conoscenza soltanto dell'inoltro del segnale, ma non del suo esito. Il meccanismo di update invece, è una funzionalità che informa il sistema del cambiamento di stato di un dispositivo ed è certamente più affidabile di un messaggio di tipo `SUCCESS`.

L'update offre anche la possibilità di conoscere lo stato di un dispositivo che è stato modificato da altri client remoti (interfacce web, cellulari, etc.). Senza tale informazione il telecomando potrebbe visualizzare stati errati.

La funzionalità dell'update è stata implementata sul telecomando mediante un socket asincrono che periodicamente richiede al gateway la notifica di aggiornamenti. Tali aggiornamenti verranno inseriti nel label del dispositivo coinvolto. Il socket, ogni volta che il telecomando entra in standby, viene inizializzato nuovamente per avere sempre tutti i dispositivi con lo stato corretto. Poiché i socket, come tutte le altre variabili, non sono visibili al di fuori della pagina in cui sono stati dichiarati, esisterà un socket per pagina che avrà il ruolo di scaricare soltanto gli aggiornamenti dell'ambiente corrente.

Negli Update fanno ovviamente eccezione i messaggi di tipo eccezionale che riguardano fughe di gas, incendi, perdite d'acqua, ecc.. Essi saranno visibili in qualsiasi pagina.

6.6 Debug e test

L'ambiente di sviluppo Pronto Edit Professional offre un simulatore che riproduce il telecomando e le sue funzionalità. Col simulatore si è testata principalmente l'interfaccia grafica dell'applicazione.

Il test sulle connessioni invece è stato fatto direttamente sul dispositivo in quanto il simulatore non possiede la capacità di gestire i socket. Per questo motivo, su ciascuna attività, è stata creata una pagina nascosta che ha assunto il ruolo di console di debug.

Il test complessivo non ha raggiunto completamente i risultati attesi a causa di anomalie che si sono riscontrate durante l'utilizzo dei socket. I metodi per la creazione di connessioni TCP, (TCPSocket, read(), write(), etc.) a volte lanciano eccezione. Questo comportamento è del tutto naturale in software distribuiti che utilizzano interfacce wifi per lo scambio di informazioni. Le librerie del ProntoScript forniscono infatti la possibilità di catturare l'evento eccezionale e associare una funzione che esegua una routine in caso di errore.

Nello script programmato, al verificarsi di un'eccezione, si intraprende nuovamente una connessione rinizializzando il socket e inviando l'ultima istruzione non pervenuta al gateway. Questa azione viene eseguita un numero di volte prefissato. Una volta superato il numero massimo di tentativi il sistema domotico viene considerato irraggiungibile. E' stato anche inserito un timeout di 5 minuti, allo scattare del quale viene eseguito un nuovo tentativo di connessione a domoNet.

Questi accorgimenti hanno limitato le anomalie, ma non le hanno eliminate del tutto. Probabilmente i malfunzionamenti continuano a presentarsi per un baco nel firmware del dispositivo.

Dal manuale risulta che il telecomando, quando entra in standby, chiude tutte le connessioni attive; al risveglio dallo standby è possibile definire la proprietà on wakeUP, tramite la quale inizializzare nuovamente le connessioni. Nella realtà, il funzionamento è differente: anche nella modalità standby le connessioni rimangono attive, mentre nella modalità normale le connessioni si chiudono inaspettatamente. Al fallire delle connessioni, non pervengono gli aggiornamenti dei dispositivi e conseguentemente si perde coerenza sullo stato complessivo del sistema domotico.

7. Conclusioni

Il software realizzato ha dimostrato la possibilità di integrare il telecomando Pronto Philis TSU9400 nell'ambiente domoNet. Il telecomando è perfettamente in grado di dialogare col framework e conseguentemente con tutti i dispositivi presenti nel laboratorio. Tramite l'interfaccia sviluppata, si possono infatti visionare tutti gli ambienti di un'abitazione e selezionare i dispositivi di qualsiasi stanza in modo semplice ed intuitivo. Con pochi gesti, è possibile configurare la climatizzazione della casa, l'impianto di illuminazione, il risparmio energetico e molto altro ancora.

La realizzazione del prototipo ha contribuito ad inserire un altro tassello per la gestione di domoNet. Prima dell'attuale lavoro, i possibili client per l'interazione col framework consistevano in un'interfaccia web, un'applicazione java su pc, ed un'applicazione per palmare. Era presente anche un telecomando del sistema Konnex in grado di interagire soltanto con dispositivi della stessa marca. Tra tutti i client presenti mancava un dispositivo che svolgesse il ruolo di telecomando unico e universale per la casa: questa funzione viene ora esercitata dal telecomando Pronto Philips TSU9400.

7.1 Sviluppi futuri

Lo script installato sul dispositivo è una demo creata per visualizzare e manipolare i dispositivi di un'abitazione, dimostrando l'efficienza di domoNet e il suo grado di interoperabilità. Essendo esso un prototipo, il software non è ancora completo, e alcune funzionalità verranno aggiunte successivamente.

La prima evoluzione riguarderà un miglioramento dell'interfaccia grafica, che per ora ha un design minimale e utilizza icone e pulsanti standard forniti dall'editor. Questi ultimi dovranno essere rimpiazzati da oggetti grafici creati ad hoc in base alle caratteristiche dei dispositivi.

Un secondo intervento prevede la totale risoluzione dei problemi sui socket. Verrà sollecitata la casa produttrice a rilasciare una versione del firmware più stabile dal punto di vista delle connessioni.

Il terzo importante passo sarà quello di portare il telecomando all'autoapprendimento della logistica della casa e dei dispositivi in essa presenti, in maniera tale che esso sia in grado di autoconfigurarsi e di scaricare le preferenze degli utenti direttamente da domoNet. Tutto ciò sarà possibile arricchendo il linguaggio DomoML che per ora offre

soltanto strumenti sufficienti ad identificare e pilotare un dispositivo all'interno di un web service.

8. Bibliografia

[1] Roberto Rocco, Editoriale Delfino, Domotica con KNX - Nuovi modi di abitare con un sistema domotico aperto, interoperabile e conforme alle norme

[2] Cesare Chiodelli, Giulio Destri, Introduzione alla Domotica, http://informatica.unipr.it/cgi-bin/files/linee - guida/dispense/chiodelli destri.pdf.

[3] Konnex Association, http://www.konnex.org

[4] http://www.myhome-bticino.it/ [5] http://www.domotica.ch/

[5] http://www.buildlab.com/article/51

[6] ProntoScript Developer's Guide Version 1.3

[7] http://www.pronto.philips.com/

[8] Cay S. Horstmann, Concetti di informatica e fondamenti di Java 2, seconda edizione

[9] Elliotte Rusty Harold, Java Network Programming, Third Edition

[10] http://java.sun.com/developer/technicalArticles/webServices/tutorial.html

[11] http://tomcat.apache.org/ [15] http://ws.apache.org/axis/

[12] Vittorio Miori, Luca Tarrini, Maurizio Manca, Gabriele Tolomei, domoNet: a framework and a prototype for interoperability of domotics middlewares based on XML and WebServices, ICCE – International Conference on Consumer Electronics, pp. 117 - 118, Suvisoft Oy Ltd, Hermiankatu 3A FIN-33720 Tampere FINLAND, IEEE, 2006.

[13] Vittorio Miori, Dario Russo, Massimo Aliberti, Domotic technologies incompatibility becomes user transparent. Communications of the Acm, vol. 53 (1), pp. 153 - 157, ACM, 2010.

[14] Vittorio Miori, Luca Tarrini, Maurizio Manca, Gabriele Tolomei, An open standard solution for domotic interoperability. IEEE Transactions on Consumer Electronics, vol. 52 (1), pp. 97 - 103, IEEE, 2006.

[15] http://ws.apache.org/xmlrpc/ [17] http://ant.apache.org/

[16] http://www.UPnP.org

[17] http://it.wikipedia.org/wiki/XML