

Consiglio Nazionale delle Ricerche
Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
(ISTI)
Laboratorio di Domotica

VESTA: un approccio innovativo per la configurazione e il controllo di sistemi domotici eterogenei

Nadia Pio, Dario Russo, Vittorio Miori

Indice

1	Introduzione	4
1.1	domoNet	5
2	Inquadramento del lavoro	8
2.1	Obiettivi	8
2.2	Motivazioni	9
2.3	Lo stato dell'arte del software per la domotica	12
2.3.1	ETS	12
2.3.2	Software sviluppato da ASG	13
2.3.3	CyberGarage	14
2.3.4	MisterHouse	14
2.3.5	Gli strumenti per i sistemi By-me e MyHome	15
2.4	La soluzione	16
3	VESTA: la semplicità al servizio della domotica	17
3.1	Concetti di base	17
3.2	L'interfaccia utente di Vesta	20
3.2.1	La barra dei menu	22
3.2.2	La barra degli strumenti	23
3.2.3	L'area di disegno o di visualizzazione	23
3.2.4	La barra di stato	24
3.2.5	L'esecuzione dei comandi	25
3.3	La gestione dei progetti	25
3.3.1	Creare un nuovo progetto	26
3.3.2	Aprire un progetto esistente	26
3.3.3	Salvare un progetto	27
3.3.4	Chiudere un progetto	27
3.4	La visualizzazione delle finestre	28
3.4.1	La visualizzazione a scheda	28
3.4.2	La visualizzazione a finestra	29
3.4.3	Attivare una finestra	30
3.5	La visualizzazione dei progetti	30
3.5.1	Lo scorrimento della vista	31
3.5.2	Attivare/disattivare la griglia	31
3.5.3	Stringere/allargare l'inquadratura	31
3.5.4	L'inquadratura automatica	31
3.5.5	L'inquadratura panoramica	32
3.6	I comandi per il disegno	32
3.6.1	La linea	33
3.6.2	Il rettangolo	33
3.6.3	Selezionare le entità	34
3.6.4	Spostare le entità	35
3.6.5	Modificare le entità	36
3.6.6	Cancellare le entità	36
3.7	La gestione dei blocchi	37
3.7.1	Creare un nuovo blocco	37
3.7.2	Importare un blocco esistente	37
3.7.3	Selezionare, spostare e cancellare un blocco	38
3.7.4	Modificare un blocco	39

3.8	La configurazione del sistema domotico	39
3.8.1	Configurare i dispositivi	39
3.8.2	Aggiungere un dispositivo	43
3.8.3	Configurare le relazioni tra i dispositivi	44
3.9	L'interazione con il sistema domotico	46
3.9.1	Inviare un comando a un dispositivo	46
3.9.2	Visualizzare la cronologia degli eventi	47
3.9.3	Visualizzare le informazioni sullo stato del dispositivo	47
4	Il prototipo	49
4.1	Strumenti e tecnologie usate	49
4.1.1	C++ e Qt	49
4.1.2	dxflib	51
4.1.3	XML	52
4.1.4	Apache Axis e Web Services	52
4.2	Architettura	52
4.3	L'interazione con domoNet	53
4.3.1	domoML	53
4.3.1.1	domoDevice	54
4.3.1.2	domoMessage	59
4.3.2	L'implementazione dell'interazione	61
4.4	L'implementazione della GUI	64
4.4.1	La finestra principale	64
4.4.2	L'importazione e il salvataggio di file DXF	66
4.4.3	Il salvataggio dei progetti	66
4.4.4	L'area di disegno	68
4.4.4.1	La scena	68
4.4.4.2	La vista	69
4.4.4.3	Gli oggetti	70
4.4.5	La configurazione	71
4.4.5.1	Il drag and drop	72
4.4.5.2	La configurazione del dispositivo	72
4.4.5.3	La configurazione delle relazioni tra dispositivi	73
4.4.6	L'interazione	74
4.4.6.1	L'invio di un comando ad un dispositivo	74
4.4.6.2	La visualizzazione della cronologia degli eventi	74
4.4.6.3	La visualizzazione dello stato di un dispositivo	75
4.5	Test	75
5	Conclusioni	76
5.1	Realizzazioni sperimentali e valutazioni	76
5.2	Sviluppi futuri	77
	Bibliografia	78

1 Introduzione

Il sistema domotico è l'insieme delle *reti domotiche* e delle *applicazioni domotiche*.

La rete domotica è l'insieme degli elementi del sistema che processano, gestiscono, trasportano e conservano le informazioni, permettendo la connessione e l'integrazione di dispositivi di calcolo, controllo, monitoraggio e comunicazione presenti nella casa.

Una rete domotica può essere costituita da più sottoreti interconnesse e viene usata per realizzare le applicazioni.

Le applicazioni domotiche si distinguono in due categorie:

- applicazioni per la gestione tecnica della casa;
- applicazioni per la gestione tecnica delle attività degli occupanti.

Gli elementi che costituiscono un sistema domotico si distinguono, rispetto alle loro funzionalità, in:

- *sensori*: dispositivi che rilevano i dati ambientali e che li trasmettono al controllore;
- *controllori hardware o software*: dispositivi o applicazioni che sono in grado di processare le informazioni provenienti dai sensori e, a fronte di queste, trasmettere dei comandi agli attuatori;

- attuatori: dispositivi che ricevono i comandi dal sistema ed eseguono i comandi;

I sistemi domotici si classificano, in base a dove risiede l'intelligenza, in tre diverse architetture:

- *architettura centralizzata*: un controllore centralizzato, in genere un processore “*special purpose*”, riceve le informazioni dai sensori, e una volta processate, genera gli opportuni comandi da indirizzare agli attuatori;
- *architettura distribuita*: tutta l'intelligenza è distribuita tra i vari sensori; tipicamente questi sistemi sono i cosiddetti sistemi a bus;
- *architettura mista*: il sistema è distribuito nel senso che è suddiviso in tanti sotto sistemi centralizzati, capaci di processare le informazioni provenienti dai sensori che controllano e trasmetterli ai loro attuatori o agli altri controllori distribuiti per la casa.

1.1 domoNet

*domoNet*¹ è un *framework open source* per Internet, che risolve il problema della cooperazione tra dispositivi domotici eterogenei dal punto di vista tecnologico. Esso fornisce una infrastruttura logica di collegamento tra le varie sottoreti domotiche basate sui diversi standard e mette a punto un meccanismo di comunicazione universale veicolato da questa infrastruttura; permette inoltre il controllo remoto dei dispositivi domotici, indipendentemente dalla loro locazione.

L'infrastruttura di *domoNet* è composta da un insieme di server rappresentati dai *web service* che espongono un insieme di servizi e/o operazioni, accessibili in un ambiente distribuito attraverso un protocollo basato su messaggi XML standardizzati. Nello specifico, i servizi offerti racchiudono le funzionalità messe a disposizione da ognuno dei dispositivi domotici, indipendentemente dalla tecnologia alla quale appartengono.

¹ <http://sourceforge.net/projects/domonet>

Ogni *web service* è connesso fisicamente ai dispositivi domotici ed è in grado di interagire con essi utilizzando dei moduli specializzati, chiamati *tech manager*, che fungono da *gateway* per la specifica tecnologia, pertanto ne serve uno per ogni sottorete relativa ad un determinato standard domotico. Ogni *tech manager* possiede quindi da un lato un'interfaccia verso il particolare sottosistema domotico cui si riferisce, e dall'altro un'interfaccia verso l'infrastruttura di *domoNet*.

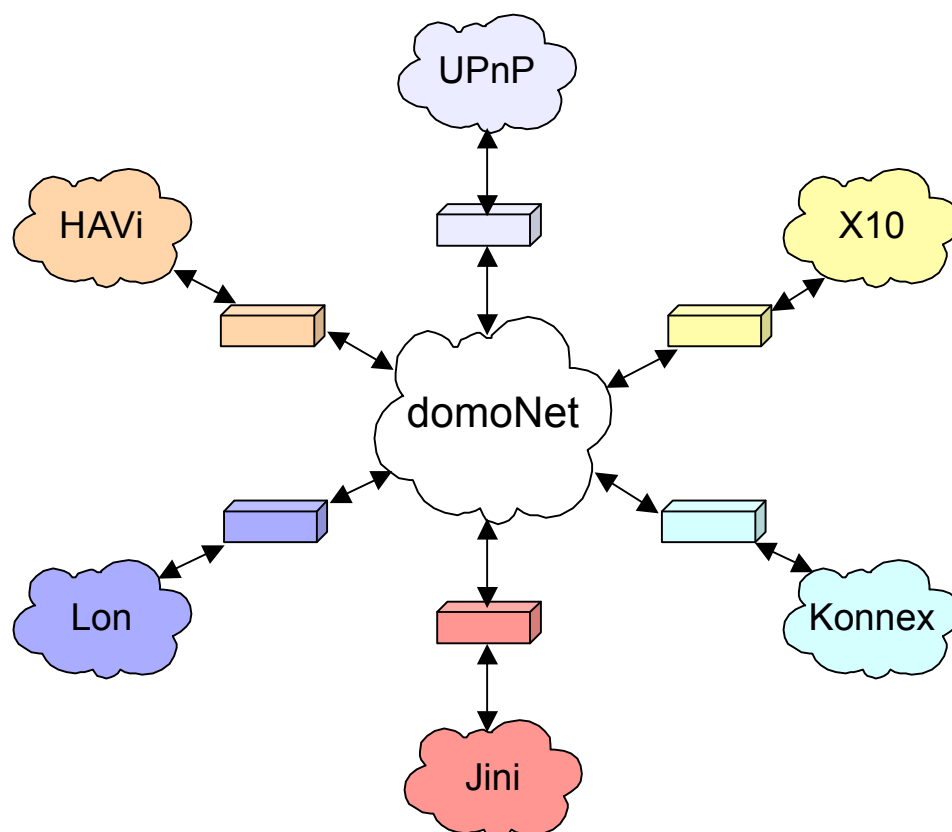


Figura 1.1: L'architettura di *domoNet*

Per permettere la comunicazione tra i vari sottosistemi domotici, viene impiegato un linguaggio ad hoc basato su XML (ovvero un dialetto di XML) chiamato *domoML*. Attraverso di esso è possibile astrarre in una maniera standard, ad un livello superiore ai sottosistemi, sia i dispositivi (*domoDevice*) che i messaggi (*domoMessage*) che circolano sull'infrastruttura di *domoNet*; esso è pertanto un linguaggio intermedio da e verso il quale tradurre descrizioni e azioni.

Ogni *web service* è in grado di catturare un messaggio proveniente dal *tech*

manager di una tecnologia, convertirlo in *domoMessage* e poi indirizzarlo al *tech manager* di destinazione che provvederà a riconvertirlo in modo che possa essere eseguito.

Per implementare il controllo remoto, *domoNet* fornisce una lista di dispositivi domotici connessi ad uno o più *web service* e descritti col formalismo *domoDevice*. Al momento in cui deve essere eseguita un'azione remota, l'applicazione cliente che interagisce con *domoNet* genera il corrispondente *domoMessage* e lo invia al *web service* che lo inoltra al *tech manager* dove viene eseguito. L'esito o l'eventuale risposta derivata dall'esecuzione viene ritrasmessa al mittente con il medesimo formalismo *domoMessage*.

Grazie a questa architettura che integra più protocolli diversi è possibile interagire con il sistema domotico attraverso un protocollo unico di comunicazione, permettendo quindi di progettare un'interfaccia “universale” per il sistema.

Con un'applicazione di questo tipo si evita l'uso dei tanti telecomandi normalmente presenti in una casa e non occorre cambiare interfaccia ogni volta che si effettua una modifica al sistema, avendo a disposizione un'interfaccia *unica* (per dispositivi di qualsiasi tecnologia), e *multifunzionale* perché integra tutte le operazioni da effettuare sul sistema domotico, dalla configurazione al controllo remoto dei dispositivi.

2 Inquadramento del lavoro

2.1 Obiettivi

Il *framework domoNet*, con la sua architettura basata sui *web service* e *tech manager* e con l'astrazione fornita dal linguaggio *domoML*, ha permesso di *integrare* e far *comunicare* tra loro sistemi domotici basati su diversi standard e tecnologie, rimuovendo un grosso ostacolo alla affermazione della domotica, e realizzando parte degli obiettivi proposti dal gruppo di ricerca.

L'obiettivo di questo lavoro è quello di aggiungere un altro tassello nel quadro sopra descritto, in modo che l'uso di un sistema domotico eterogeneo integrato attraverso *domoNet* avvenga nella maniera più semplice ed intuitiva possibile. La *semplicità* d'uso è infatti una prerogativa importante che può favorire la diffusione della domotica.

In questa ottica, lo strumento da realizzare deve rendere accessibili le funzionalità di *domoNet* e mettere a disposizione dell'utente una serie di *strumenti grafici* per la rappresentazione della pianta di un'abitazione, la disposizione dei dispositivi domotici, la loro configurazione e l'interazione con essi.

In particolare deve permettere:

- la *rappresentazione* grafica della pianta dell'abitazione in cui è installato il sistema domotico, attraverso strumenti simili a quelli messi a disposizione da applicazioni CAD (*Computer Aided Design*);

- l'eventuale *importazione* di disegni realizzati con altre applicazioni CAD per agevolare l'operazione di disegno;
- la *disposizione* all'interno della pianta, dei sistemi domotici installati nel sistema;
- la *configurazione* dei dispositivi e del sistema (cioè delle relazioni tra i vari dispositivi);
- l'*accesso remoto* al sistema domotico per impartire comandi ai dispositivi;
- il *monitoraggio* in tempo reale del sistema, visualizzando la cronologia degli eventi che si sono verificati e lo stato dei ciascun dispositivo;
- la *notifica* di situazioni anomale di funzionamento dei dispositivi e/o situazioni di pericolo captate dai sensori.

2.2 Motivazioni

L'architettura di *domoNet* è stata studiata per venire in contro all'esigenza di avere il *controllo remoto* dei dispositivi, pertanto un'applicazione cliente che voglia interagire con *domoNet*, non è vincolata al perimetro dell'abitazione, ma può essere eseguita su qualsiasi dispositivo remoto (telecomando specifico del sistema, calcolatore, palmare, cellulare, ecc.) che abbia accesso alla rete internet, in modo da potersi connettere ai server di *domoNet* (*web service*).

L'utilità di questa funzionalità è scontata: l'utente può avere in questa maniera informazioni in tempo reale sullo stato del suo sistema domotico, cosa che può essere molto utile quando è lontano dall'abitazione; non solo, egli può avere accesso, qualora ne abbia il permesso, anche al sistema di altri utenti, come ad esempio a quello dei genitori anziani che vivono da soli, beneficiando della tranquillità che qualunque situazione anomala o pericolosa accada presso la loro abitazione, venga tempestivamente segnalata.

L'interazione con i dispositivi, indipendentemente dallo standard cui sono conformi, è possibile solo effettuando a *domoNet* delle richieste che rispettino il

formalismo *domoMessage*, un linguaggio ad hoc basato su XML. Se si vuole garantire semplicità d'uso all'utente, sembra chiaro che questi non può formulare richieste o inviare comandi a *domoNet* in questa maniera. Serve perciò un modulo interfaccia, che da un lato comunichi in maniera molto intuitiva e semplice con l'utente, e dall'altro traduca la richiesta da inviare a *domoNet* nel formalismo corretto.

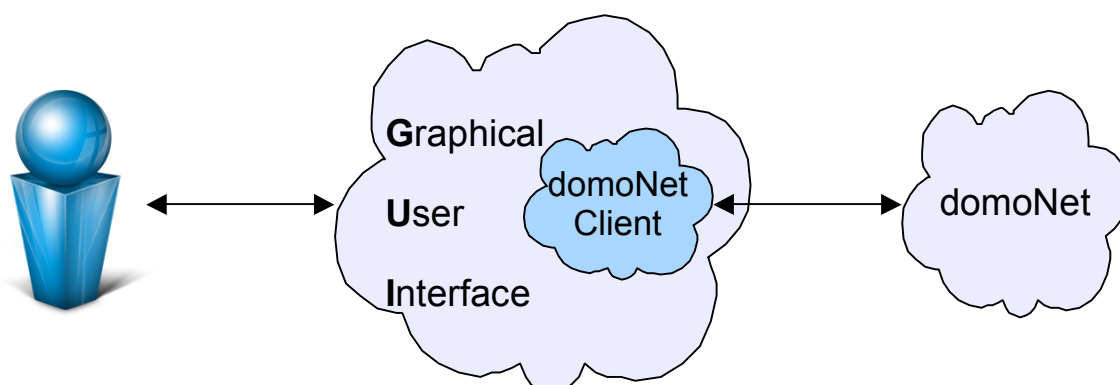


Figura 2.1: L'interazione dell'utente con *domoNet*

Con l'introduzione della domotica nelle abitazioni, e l'aumento dei dispositivi da collegare, si è reso necessario l'utilizzo di sistemi di interconnessione nuovi, che permettano di ridurre i limiti dei collegamenti punto-punto, tipici degli impianti tradizionali. In questi sistemi infatti, le funzionalità sono realizzate mediante collegamenti fisici, stabiliti in fase di progetto dell'impianto, e ogni loro modifica, estensione o sostituzione richiede un rifacimento più o meno impegnativo del cablaggio.

L'utilizzo del bus permette non solo di abbattere il cablaggio necessario per collegare i dispositivi, ma di implementare le funzionalità del sistema attraverso *connessioni logiche*, che possono essere facilmente modificate in qualsiasi momento, sia in corso d'opera che ad installazione ultimata, introducendo notevoli vantaggi in termini di flessibilità e costi da sostenere per la manutenzione.

La procedura mediante la quale si stabiliscono queste connessioni logiche tra i vari dispositivi si chiama *configurazione*. Tutti i dispositivi sono modulari e configurabili; molti di essi sono plurifunzionali e, in base alla loro configurazione,

non al cablaggio, possono svolgere funzioni che negli impianti tradizionali sono assicurate da molteplici componenti diversi.

Questa procedura viene effettuata in genere dal personale tecnico specializzato, pertanto se l'utente vuole cambiare la configurazione è obbligato a contattare l'azienda che ha installato l'impianto.

Ecco che ritorna evidente la necessità di avere a disposizione uno strumento che permetta in una maniera estremamente intuitiva di poter configurare i dispositivi domotici, dando all'utente, nei limiti delle sue capacità, maggiore libertà, flessibilità e cognizione del suo impianto.

Attualmente sono disponibili diverse soluzioni software che consentono il controllo remoto di sistemi domotici e della relativa configurazione. Questo perché ogni azienda o associazione che ha stabilito uno standard o messo a punto una tecnologia domotica, ha dovuto fornire agli utenti gli strumenti adatti a interagire con il sistema che è stato installato, cioè in grado di comunicare seguendo il protocollo stabilito.

Nel caso di *domoNet*, nessuno di questi software può funzionare, ma serve una *soluzione ad hoc*, per due motivi fondamentali.

Il primo è che, non esiste ancora una soluzione che abbia integrato in un'unica applicazione grafica sia la configurazione che il controllo remoto del sistema domotico.

Secondariamente, visto che *domoNet* non è conforme ad uno standard particolare, ma è una rete che mette in comunicazione sistemi eterogenei, serve un'applicazione in grado di comunicare con *domoNet*, utilizzando i formalismi *domoDevice* e *domoMessage* per permettere rispettivamente, la configurazione dei dispositivi e il controllo remoto. Saranno poi i vari *tech manager* a implementare l'interazione diretta con i sottosistemi rispettando i relativi protocolli.

Un'applicazione di questo tipo, non ha i limiti di quelle esistenti, ma sarebbe un'interfaccia “*universale*”, che non deve essere modificata se viene fatto qualche

cambiamento all'interno del sistema domotico, sia in termini di dispositivi sia di tecnologie utilizzate.

2.3 Lo stato dell'arte del software per la domotica

Tutte le maggiori aziende del settore dell'automazione domestica che hanno messo a punto tecnologie e protocolli per la domotica, forniscono ai loro clienti le applicazioni per poter gestire facilmente il sistema installato. Esistono anche molte applicazioni *open source* che offrono questa funzionalità, ma tutte hanno un limite: gestiscono solo i dispositivi basati su una particolare tecnologia domotica.

Pertanto, quando si dispone di un sistema eterogeneo, con dispositivi conformi a standard diversi, è necessario utilizzare più applicazioni per gestire i vari sottosistemi. Tutto ciò ovviamente, a discapito della semplificazione che dovrebbe portare la domotica nelle nostre abitazioni.

Per chiarire la necessità di una soluzione *ad hoc* per la configurazione ed il controllo remoto di *domoNet*, viene fatta una breve panoramica delle applicazioni software per la domotica più importanti attualmente esistenti, mettendo in risalto i limiti di queste.

2.3.1 ETS

*ETS*¹ è un software per la configurazione dei dispositivi basati sullo standard *Konnex (KNX)*.

Esso permette di definire la struttura gerarchica dell'edificio, la disposizione dei componenti e la relativa associazione logica per lo svolgimento delle funzioni richieste, il tutto attraverso un'unica interfaccia. Tutti i dispositivi certificati *Konnex* sono in grado di comunicare in modo standard all'interno dell'impianto, a patto che prima della configurazione di un dispositivo, venga caricata in *ETS* la relativa libreria fornita dal costruttore.

¹ <http://www.konnex.it/it/konnex/ETS.asp>

Attualmente *ETS* è l'unica possibilità reale per configurare un sistema domotico, ma purtroppo permette la configurazione dei soli dispositivi certificati *Konnex*. Inoltre è un software proprietario che necessita di licenza a pagamento, basato su librerie non libere ed è possibile eseguirlo esclusivamente su sistemi Windows.

2.3.2 Software sviluppato da ASG

Il gruppo di ricerca Automation Systems Group² dell'Istituto di Automazione dell'Università Tecnica di Vienna, ha realizzato nel corso degli ultimi anni diverse applicazioni open source per la gestione del bus e dei dispositivi *Konnex*.

- *BCU SDK*

È un kit di sviluppo software di alto livello che permette di creare propri applicativi per dispositivi *EIB/KNX*, usando un approccio orientato agli oggetti e senza dover usare codice assembler.

Esso comprende il GNU toolchain: un progetto per adattare il GNU Binutil3 e GCC4 ai microcontrollori M68HC05 e per creare altri strumenti di supporto per la programmazione di *BCU EIB*.

- *Calimero*

È una libreria Java che permette di gestire in maniera molto semplice l'accesso al bus *Konnex* (lettura e scrittura), attraverso un adattatore IP (*KNXnet/IP Tunneling Server*). Per l'utilizzo della libreria non è richiesta la conoscenza di questo protocollo.

- *KNXLive!*

Distribuzione Linux basata su *Knoppix 5.0.1* che non richiede l'installazione o configurazione di ulteriori software per connettersi ad un'installazione *EIB/KNX*. Il pacchetto di installazione contiene gran parte del software prodotto da *ASG*, pre-installato e pronto all'uso.

² <https://www.auto.tuwien.ac.at/a-lab/software.html>

³ GNU Binary Utilities: insieme di strumenti di programmazione per la manipolazione del codice oggetto.

⁴ GCC (GNU Compiler Collection): insieme di compilatori per sistemi GNU/Linux.

•BASys 2003

Fornisce un nuovo approccio per pianificare e configurare bus domotici. Attualmente è in versione beta e supporta l'accesso a installazioni multiple di sistemi bus EIB/KNX.

2.3.3 CyberGarage

Satoshi Konno è un programmatore giapponese che ha creato *CyberGarage*⁵, una raccolta di librerie e applicazioni scritte in Java e C++ per sistemi di realtà virtuale e grafica 3D, liberamente distribuite. Ha inoltre sviluppato un interessante *framework* per la configurazione e gestione di *UPnP*.

CyberLink è un pacchetto di sviluppo che permette di creare e gestire facilmente dispositivi domotici *UPnP* mettendo a disposizione il protocollo di base per la comunicazione. Esistono diverse implementazioni per C, C++, Java (con supporto IPv6) e Perl.

2.3.4 MisterHouse

*MisterHouse*⁶ è una tra le applicazioni *open source* più diffuse e utilizzate, che permette di gestire i dispositivi domotici basati su *X10*, tramite un'interfaccia web altamente configurabile e decisamente accattivante.

È scritta in Perl e può essere utilizzata su piattaforma Windows, Linux e MacOS. Permette di creare script Perl per estendere le funzionalità *X10* e programmare il comportamento del sistema in base agli eventi in corso.

Può eseguire comandi arrivati input dalla voce dell'utente finale, oppure secondo parametri di tempo a seconda dell'ora del giorno; legge e scrive i dati da qualsiasi apparato seriale collegato.

5 <http://www.cybergarage.org/>

6 <http://misterhouse.sourceforge.net/>

2.3.5 Gli strumenti per i sistemi By-me e MyHome

EasyTool e *Mediacenter* sono due applicazioni fornite da *Vimar*, un'azienda italiana del settore elettrico, che ha spostato i suoi interessi nella domotica. Esse permettono la gestione del sistema domotico *By-me*⁷; la prima offre funzionalità di backup e ripristino dei dati dell'impianto, e di configurazione degli scenari; la seconda consente di comandare e monitorare il sistema *By-me* mediante un'interfaccia grafica che funziona nell'ambiente *Media Center* di *Microsoft Vista*, gestendo tutte le apparecchiature di intrattenimento presenti nella casa.

Il sistema *MyHome*⁸ di *BTicino* invece utilizza strumenti diversi per la progettazione, la configurazione e il controllo.

Sul sito dell'azienda è possibile scaricare un software CAD che facilita il progettista nell'inserimento dei componenti a listino nel grafico. La configurazione avviene essenzialmente spostando manualmente alcuni spinotti che si trovano in tutti i loro dispositivi. In questo modo anche elettricisti generici che non sanno usare un computer, dopo un breve corso sono in grado di fare un'installazione⁹ e configurarla.

La gestione invece può venire fatta attraverso un telecomando locale, un telefono o attraverso la rete internet. Per questo sono stati predisposti alcuni componenti che gestiscono un linguaggio telefonico¹⁰ e un linguaggio aperto, chiamato *MyOpen*¹¹, che si può utilizzare per creare nuove interfacce web. *BTicino* gestisce anche un deposito di applicazioni fatte dall'utente con questo linguaggio. Alternativamente l'utente può abbonarsi al centro gestione *BTicino* e in questo caso può collegarsi alla propria abitazione attraverso un normale browser al sito *BTicino* e gestire in modo completo e sicuro tutta la propria abitazione (telecamere comprese).

7 <http://www.vimar.eu/>

8 <http://www.myhome-bticino.it>

9 L'installatore si baserà sul progetto predisposto dal progettista

10 Il linguaggio telefonico ha una sintassi che utilizza solo i caratteri numerici e i simboli "*" e "#" che si trovano in qualsiasi telefono moderno

11 *MyOpen* è descritto al sito <http://www.myopen-bticino.it/>

2.4 La soluzione

Data l'architettura del *framework domoNet* (basata sui *web service*), la soluzione software che risulta più adatta ad interfacciare *domoNet*, è un'applicazione *client* costituita da un modulo in grado di inviare richieste al server per mezzo della rete internet, e da una *GUI* (*Graphical User Interface*), ovvero un'interfaccia grafica (figura 3.1).

Essa permette all'utente di interagire con estrema semplicità e naturalezza, operando in un ambiente a lui familiare, basato sul sistema a finestre e molto simile anche nell'aspetto (*look and feel*) al sistema operativo che ne permette l'esecuzione.

Un'applicazione di questo tipo può essere eseguita sia da un calcolatore convenzionale che su un dispositivo mobile, come un palmare o un cellulare, purché abbia accesso alla rete internet, e potrà essere visualizzata su schermi convenzionali o sui nuovissimi *touch screen* che saranno la vera rivoluzione dei prossimi anni mandando in soffitta tanti anni di onorato lavoro dei mouse.

Naturalmente l'applicazione grafica deve essere adatta al tipo di dispositivo su cui verrà eseguita, cioè deve considerarne i limiti della capacità di calcolo, della capienza della memoria e del consumo di energia.

La soluzione scelta per questo lavoro è un'applicazione grafica *open source*, multiplatforma e multifunzionale per calcolatori convenzionali.

3 VESTA: la semplicità al servizio della domotica

*Vesta*¹ è un progetto software *open source*, che nasce dall'esigenza di semplificare l'uso dei sistemi domotici, dalla loro configurazione fino all'interazione con i dispositivi, locale o remota, da parte del personale tecnico e non.

In questo paragrafo si vuole fornire al lettore una panoramica di tutte le funzionalità e caratteristiche messe a disposizione da *Vesta*, pertanto può essere considerato il manuale di riferimento per l'utente.

3.1 Concetti di base

Il documento principale su cui si basa l'applicazione, è il *progetto (project)*, un file in formato DXF² (*Drawing Exchange Format*) contenente il disegno della pianta dell'abitazione in cui è installato il sistema domotico con cui si vuole interagire. La scelta di questo formato è stata fatta per rendere compatibili con applicazioni CAD i disegni creati con *Vesta*, e viceversa.

Il progetto si può trovare in due stati, che corrispondono allo stato del sistema domotico, ovvero se questo è configurato o meno.

- *Non configurato*: il progetto visualizza solo la pianta dell'abitazione e non contiene alcun dispositivo; il sistema domotico dunque non è configurato.
- *Configurato*: tutti i dispositivi domotici, identificati da un'icona, sono stati

¹ *Vesta* è il nome della dea romana protettrice del focolare domestico e della casa.

² <http://usa.autodesk.com/>

3. VESTA: la semplicità al servizio della domotica

disposti sulla pianta dell'abitazione rispecchiando le loro posizioni reali, e sono stati configurati attraverso gli strumenti grafici messi a disposizione da *Vesta*; se il progetto è configurato anche il sistema domotico lo è.

L'interazione col sistema è permessa solo se questo è stato configurato, e quindi anche il progetto che lo rappresenta.

In base allo stato del progetto, vengono previste diverse fasi operative, che contraddistinguono due modalità dell'esecuzione di *Vesta*, la modalità di configurazione e la modalità di controllo.

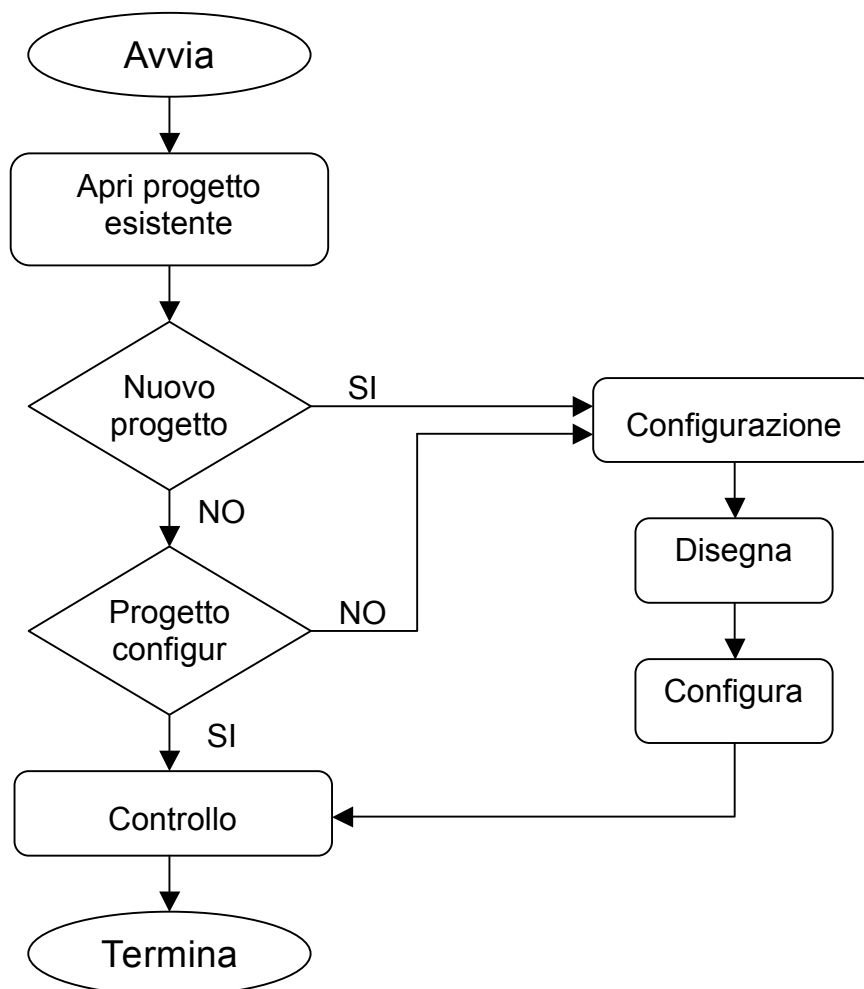


Figura 3.1: Diagramma di flusso dell'esecuzione di *Vesta*

All'avvio di *Vesta*, l'utente deve scegliere se creare un progetto nuovo o se aprirne

uno esistente.

Nel primo caso, il progetto non è configurato e l'applicazione inizia l'esecuzione in *modalità di configurazione* (fig. 3.2a), che consta di due fasi:

- 1 nella prima fase si deve disegnare la pianta dell'abitazione con gli strumenti da disegno messi a disposizione da Vesta; il disegno serve da base grafica per la fase di configurazione vera e propria;
- 2 nella seconda fase, Vesta si connette al sistema domotico³ che si intende gestire e fornisce all'utente la lista dei dispositivi presenti nel sistema. Essi devono essere disposti sulla pianta e configurati uno per volta. Al termine di questa procedura l'esecuzione di Vesta passa in modalità di controllo.

Se viene scelto di aprire un progetto esistente, possono verificarsi due casi: se questo non è configurato si entra in modalità di configurazione e si procede come su descritto; se è configurato, esso viene caricato e si entra direttamente in *modalità di controllo* (fig. 3.2b), in cui l'utente può interagire subito con il sistema domotico, inviando comandi a determinati dispositivi e ricevendo la cronologia degli eventi che si verificano nel sistema.

Come accennato all'inizio del paragrafo, la fase di disegno segue le procedure e il formato utilizzato dalle applicazioni CAD. A tal proposito è opportuno chiarire brevemente i concetti CAD di base che verranno utilizzati dall'applicazione.

Entità

Le entità sono oggetti grafici in un sistema CAD. Le tipiche entità esistenti nella maggior parte dei sistemi CAD sono: punti, linee, archi di cerchio e di ellisse. Poiché lo scopo di questa applicazione non è quello di disegnare progetti, l'unica entità disponibile è la linea, e per comodità il rettangolo, visto che la maggior parte delle stanze di un'abitazione sono rettangolari. Il rettangolo, non essendo una primitiva grafica dei sistemi CAD, nel salvataggio del progetto viene convertito in quattro linee.

³ Vesta si connette al web service di *domoNet*, che rende iteroperabile il sistema domotico.

Blocchi

Un blocco è un gruppo di entità che si comporta come se fosse un'unica entità. Il vantaggio del loro uso, sta nel fatto che i blocchi possono essere inseriti nello stesso disegno più volte, senza dover ridisegnare le singole entità che lo compongono. Inoltre è possibile inserirli con attributi differenti, vale a dire che si possono scegliere diversi fattori di scala e angoli di rotazione.

Il sistema di riferimento

Il sistema di riferimento utilizzato in *Vesta* e in tutte le applicazioni CAD è quello cartesiano. Esso è costituito da due assi perpendicolari, l'asse X delle ascisse e l'asse Y delle ordinate che si intersecano in un punto chiamato origine e indicato nell'applicazione da una croce rossa.

La posizione di un punto è determinata all'interno di questo sistema attraverso due numeri che rappresentano rispettivamente la distanza dall'asse Y e la distanza dall'asse X.

Visualizzazione in CAD

Diversamente dal disegno manuale, nel disegno CAD non vi è alcuna necessità di determinare in anticipo la dimensione del foglio e la scala del disegno. Nel disegno al calcolatore non esiste alcuna scala di disegno: tutti i valori e le distanze vengono specificati utilizzando i valori numerici reali. L'unità di misura può essere assunta dall'utente.

Sullo schermo, l'utente può regolare l'area visibile del disegno stringendo o allargando l'inquadratura, per individuare più dettagli o per avere una vista d'insieme del disegno.

3.2 L'interfaccia utente di Vesta

Quando si avvia *Vesta*, si apre la *finestra principale* dell'applicazione, contenente una serie di componenti grafiche, alcune delle quali sono visibili durante tutta l'esecuzione dell'applicazione, mentre altre sono peculiari solo di alcune fasi. Le

3. VESTA: la semplicità al servizio della domotica

prime verranno descritte in questa sezione, mentre le altre verranno trattate man mano che si presenta la necessità.

Quando *Vesta* viene avviato in *modalità di configurazione*, la finestra principale ha questo aspetto:

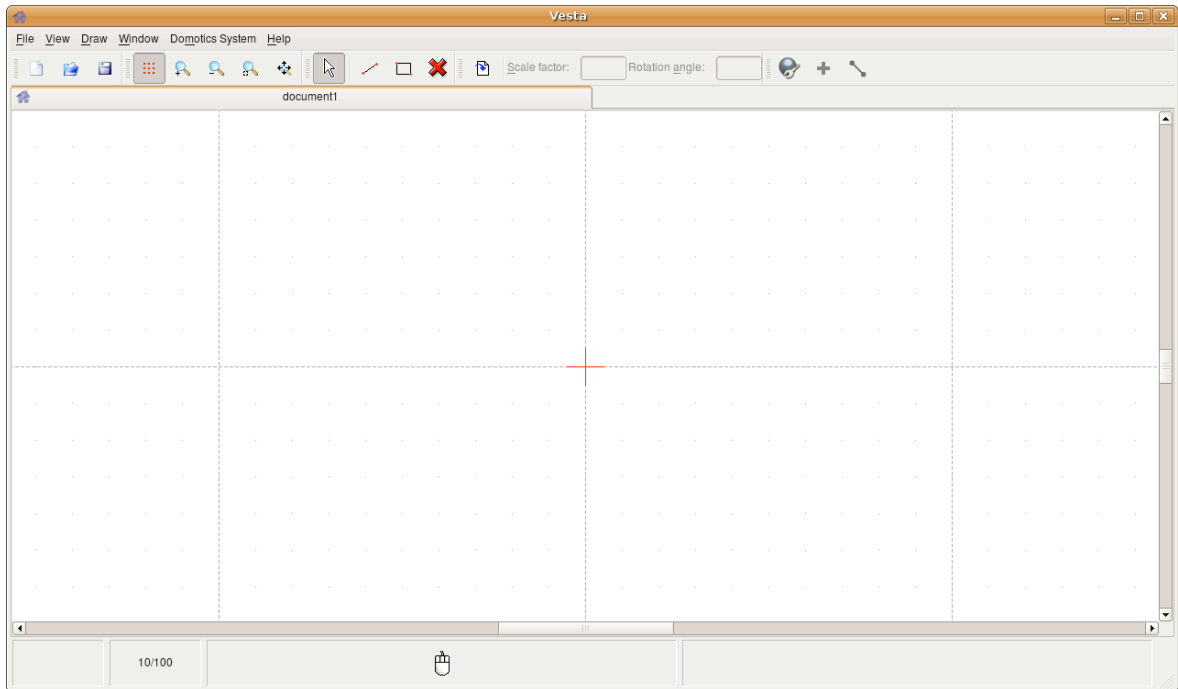


Figura 3.2a: *Vesta* in modalità di configurazione

mentre se viene caricato un progetto già configurato, la finestra principale sarà in *modalità di controllo*, come illustrato nella pagina successiva (fig. 3.2b).

3. VESTA: la semplicità al servizio della domotica

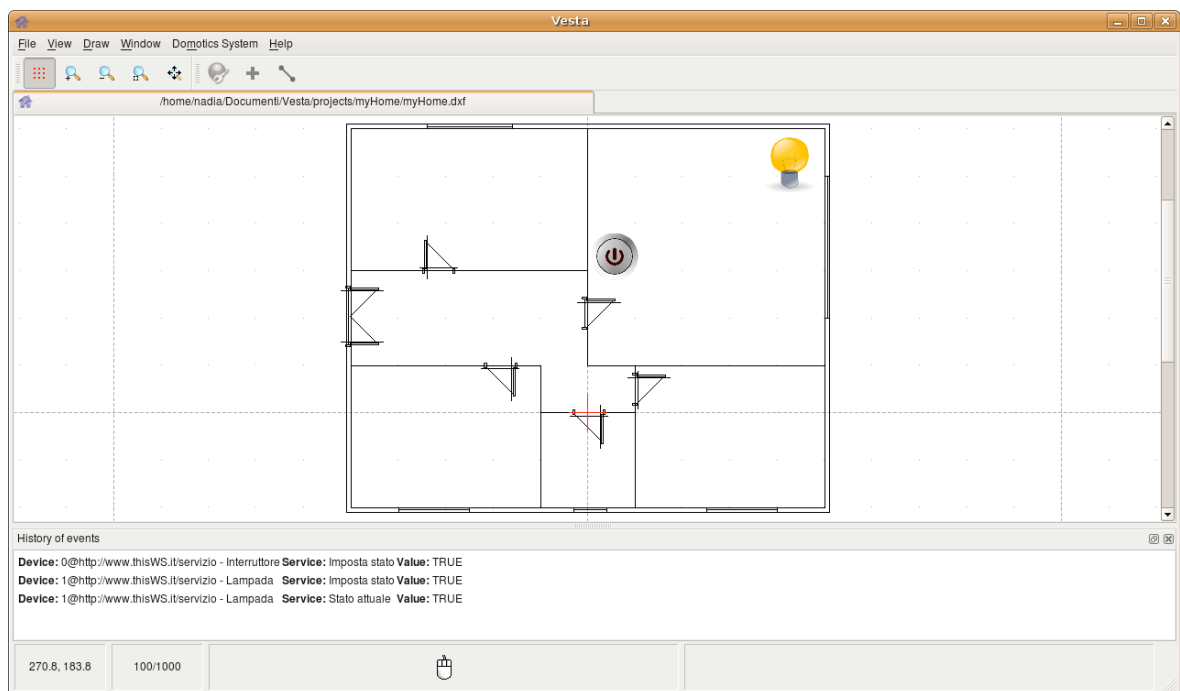


Figura 3.2b: *Vesta* in modalità di controllo

3.2.1 La barra dei menu

La barra dei menu raggruppa all'interno dei menu a tendina le funzionalità a seconda dei loro scopi, permettendo di avere una visione completa di tutti i comandi disponibili. L'uso di questa barra è raccomandato all'utente che non ha ancora dimestichezza con l'applicazione.

Per velocizzare l'esecuzione dei comandi, si può utilizzare la tastiera, digitando una combinazione di tasti:

- se si vuole aprire il menu per scegliere una funzione, bisogna premere il tasto Alt seguito dalla lettera sottolineata del menu che si vuole aprire, e poi dalla lettera sottolineata del comando da eseguire; ad esempio la combinazione Alt+F+N apre un nuovo progetto;
- se invece si vuole eseguire direttamente il comando, si può digitare la relativa combinazione di tasti (*scorciatoia da tastiera*), che si trova accanto ai comandi nei menu; per aprire un nuovo documento ad esempio basta

digitare Ctrl+N.

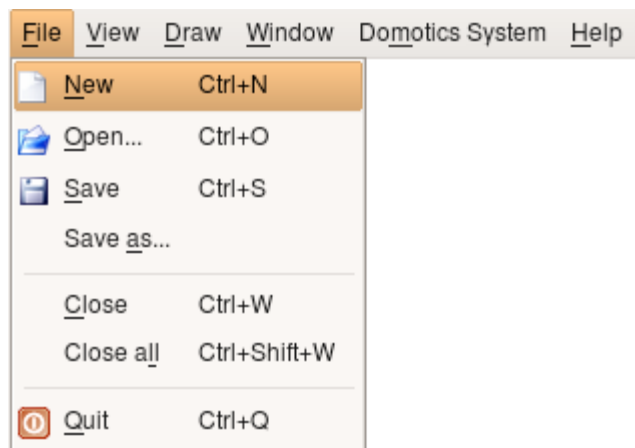


Figura 3.3: La barra dei menu

3.2.2 La barra degli strumenti

La barra degli strumenti permette di velocizzare l'esecuzione dei comandi utilizzando il mouse, perché attraverso un unico click sul pulsante relativo, questo viene eseguito. Oltre ai pulsanti, questa barra può contenere anche altre componenti grafiche, nel nostro caso si tratta di caselle di testo che permettono l'inserimento di alcuni parametri necessari all'esecuzione di alcuni comandi.



Figura 3.4: Le barre degli strumenti

3.2.3 L'area di disegno o di visualizzazione

La parte centrale della finestra principale è occupata dall'*area di disegno* (fig. 3.2a) o dall'*area di visualizzazione* (fig. 3.2b).

La prima permette il disegno della pianta dell'abitazione quando *Vesta* è in modalità di configurazione, mentre la seconda visualizza la pianta e dello stato del sistema domotico, quando *Vesta* è in modalità di controllo.

3. VESTA: la semplicità al servizio della domotica

Quest'area rappresenta un piano, teoricamente illimitato, al cui centro vi è l'origine degli assi del sistema di riferimento cartesiano e nello sfondo è rappresentata una griglia per agevolare la procedura di disegno e dare un'idea delle distanze.

3.2.4 La barra di stato

La barra di stato nella parte inferiore della finestra principale, mostra le informazioni relative allo stato corrente di *Vesta*, nonché delle indicazioni di aiuto per l'utente (fig. 3.5a).

- Il *pannello delle coordinate* visualizza la posizione corrente del cursore del mouse in coordinate cartesiane, quando questo si trova nell'area di disegno o visualizzazione. Il primo numero indica l'ascissa, mentre il secondo indica l'ordinata.
- Il *pannello del fattore di scala*, indica che al livello di ingrandimento corrente ogni quadrato rappresentato nella griglia ha un lato lungo 100 unità, e i punti in esso contenuti sono distanziati di 10 unità.
- Il *pannello del mouse* fornisce indicazioni sulle azioni che vengono eseguite se si cliccano il tasto destro e sinistro del mouse.
- Il *pannello della selezione* visualizza informazioni relative alle entità che vengono selezionate all'interno dell'area di disegno.



Figura 3.5a: La barra di stato con i pannelli

Quando si tiene il puntatore del mouse sopra un pulsante senza premerlo, la barra

3. VESTA: la semplicità al servizio della domotica

di stato viene sovrascritta da un commento (*tool tip*) che fornisce indicazioni sull'uso del pulsante. Quando si allontana il puntatore, ricompare la barra di stato con i pannelli sopra descritti.

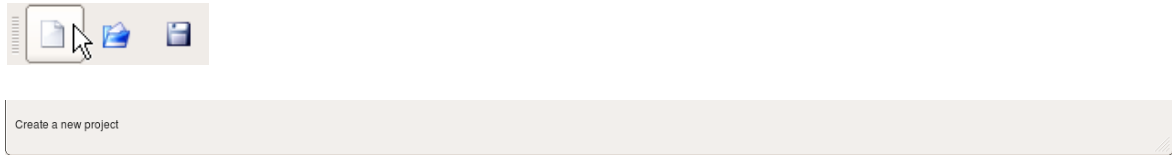


Figura 3.5b: La barra di stato con *tool tip*

3.2.5 L'esecuzione dei comandi

Per eseguire i comandi in *Vesta* ci sono tre possibilità:

- selezionare un pulsante da una barra degli strumenti;
- scegliere una voce da un menu a tendina;
- usare una scorciatoia da tastiera.

Altre azioni più complesse richiedono ulteriori operazioni all'utente, che possono essere:

- l'utilizzo di finestre di dialogo, come ad esempio⁴ nel caso dell'apertura di un progetto o la configurazione di un dispositivo domotico;
- l'inserimento di alcuni parametri nelle caselle della barra degli strumenti, come nell'importazione dei blocchi;
- l'uso diretto del mouse per operazioni come la modifica delle entità o il trascinamento dei dispositivi domotici in fase di configurazione.

3.3 La gestione dei progetti

Vesta può importare e visualizzare disegni creati con tutte le applicazioni CAD in

⁴ Gli esempi riportati in questa sezione verranno descritti in maniera approfondita nel corso della trattazione.

3. VESTA: la semplicità al servizio della domotica

grado di esportare disegni in formato DXF. Questi possono essere utilizzati come base per la configurazione del sistema domotico. Per salvare i progetti, *Vesta* si attiene al formato DXF 2000.

3.3.1 Creare un nuovo progetto

Barra degli strumenti:



Menu:

File → New

Scorciatoie da tastiera:

Alt+F+N | Ctrl+N

Abilitazione del comando:

Fase di disegno della modalità di configurazione

Con questo comando si possono creare nuovi progetti. Viene aperta una nuova finestra con l'area di disegno completamente vuota (non contiene né entità né blocchi); il progetto appena creato non è ancora configurato.

3.3.2 Aprire un progetto esistente

Barra degli strumenti:



Menu:

File → Open...

Scorciatoie da tastiera:

Alt+F+O | Ctrl+O

Abilitazione del comando:

Fase di disegno della modalità di configurazione

Questo comando permette di aprire un progetto esistente, scegliendolo da una finestra di dialogo in cui si può sfogliare le cartelle del *file system*.

Se il progetto che si vuole aprire non è ancora configurato, il disegno viene caricato nell'area di disegno di una nuova finestra e può essere utilizzato come rappresentazione grafica di base per la configurazione del sistema domotico, saltando o abbreviando la fase di disegno.

Si tenga presente che solo le entità supportate da *Vesta* presenti nei disegni saranno caricate, mentre tutte le altre saranno ignorate. Se si importata un documento DXF creato da un altro programma, è consigliabile salvare quel file con un nuovo nome prima di lavorarci su. Se questo non viene fatto, in fase di salvataggio andranno perse dal disegno tutte le entità non supportate.

3. VESTA: la semplicità al servizio della domotica

Se il progetto è già stato configurato, esso viene caricato nell'area di visualizzazione con il relativo disegno e l'ubicazione dei dispositivi domotici precedentemente configurati (figg. 3.2a e 3.2b).

In questo caso, *Vesta* richiede che tutti i file aperti vengano salvati e chiusi, perché nella modalità di controllo *Vesta* può gestire un progetto alla volta. Qualora si volesse aprire un altro progetto bisogna prima chiudere quello corrente e poi aprire quello desiderato.

3.3.3 Salvare un progetto

Barra degli strumenti:



Menu:

File → Save
File → Save as...

Scorciatoie da tastiera:

Alt+F+S | Ctrl+S
Alt+F+A

Abilitazione del comando:

Fase di disegno della modalità di configurazione

Questo comando salva il progetto corrente in un file.

Per salvare il lavoro nello stesso file da cui è stato aperto il disegno, si utilizza il pulsante sulla barra degli strumenti o il menu *File* → *Save*.

Se invece si vuole salvare un nuovo progetto o un progetto con un diverso nome, si usa il menu *File* → *Save as*. Verrà chiesto prima di salvare, di indicare il nome da dare al nuovo documento e la posizione all'interno del *file system*.

Un progetto si può salvare esplicitamente solo quando *Vesta* è in fase di disegno della modalità di configurazione. Oltre questa fase, l'unico progetto che rimane aperto viene salvato implicitamente al termine della configurazione, prima di passare in modalità di controllo.

3.3.4 Chiudere un progetto

Menu:

File → Close
File → Close all
File → Quit

3. VESTA: la semplicità al servizio della domotica

Scorciatoie da tastiera:	Alt+F+C Ctrl+W Alt+F+A Ctrl+Shift+W Alt+F+Q Ctrl+Q
Abilitazione del comando:	Sempre abilitato (eccetto il comando <code>Close all</code> che è abilitato solo in fase di disegno della modalità di configurazione)

Il comando *Close* chiude il progetto corrente.

Se *Vesta* è in fase di disegno e il progetto contiene cambiamenti non salvati, appare una finestra con un messaggio che chiede se bisogna salvare il progetto prima di chiuderlo; si può decidere di salvare le modifiche o di ignorarle.

Se si è in fase di configurazione o in modalità di controllo non si possono fare modifiche al progetto e quindi verrà solamente chiuso. Questo comando serve quando si vuole chiudere il progetto senza terminare l'applicazione, per poterne aprire un altro.

Il comando *Close all* chiude tutti i progetti aperti. È abilitato solo nella fase di disegno; oltre questa fase non ha più senso perché vi è un solo progetto aperto.

Se si vuole chiudere un progetto e tutta l'applicazione bisogna usare il comando *Quit*.

3.4 La visualizzazione delle finestre

Quando *Vesta* è nella fase di disegno della modalità di configurazione, nella finestra principale ci possono essere più sotto finestre aperte. Esse possono essere visualizzate come schede o come finestre classiche e vengono gestite dal menu *Window*.

3.4.1 La visualizzazione a scheda

Menu:	Window → Tab
Scorciatoie da tastiera:	Alt+W+T Ctrl+T
Abilitazione del comando:	Fase di disegno della modalità di configurazione

Si utilizza questo comando per visualizzare le finestre dei vari progetti in una

3. VESTA: la semplicità al servizio della domotica

barra di schede.

Questa modalità è attiva di default, e si può disattivare eseguendo il medesimo comando.

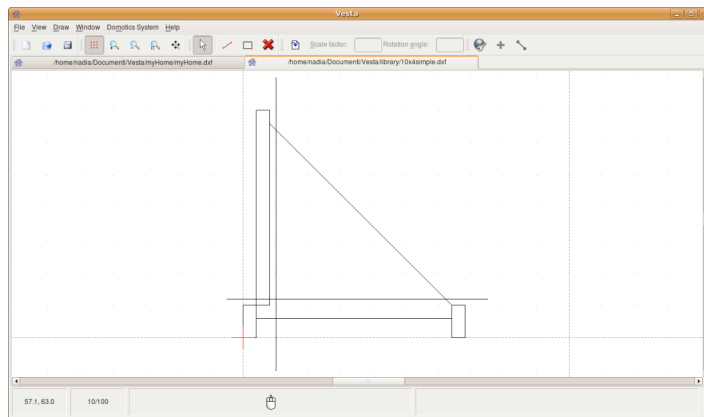


Figura 3.6a: La visualizzazione a schede

3.4.2 La visualizzazione a finestra

Menu:

Window → Tile

Window → Cascade

Scorciatoie da tastiera:

Alt+W+L | Shift+T

Alt+W+C | Shift+C

Abilitazione del comando:

Fase di disegno della modalità di configurazione

Si usano questi comandi se si vuole visualizzare i progetti nelle finestre classiche. Questo è possibile solo se il comando *Tab* è disattivato.

Il comando *Tile* suddivide l'area centrale della finestra principale e dispone le finestre in modo che possano essere visualizzate tutte contemporaneamente.

Il comando *Cascade* invece, le impila una sopra l'altra, pertanto sarà visibile solo quella che si trova al livello superiore.

3. VESTA: la semplicità al servizio della domotica

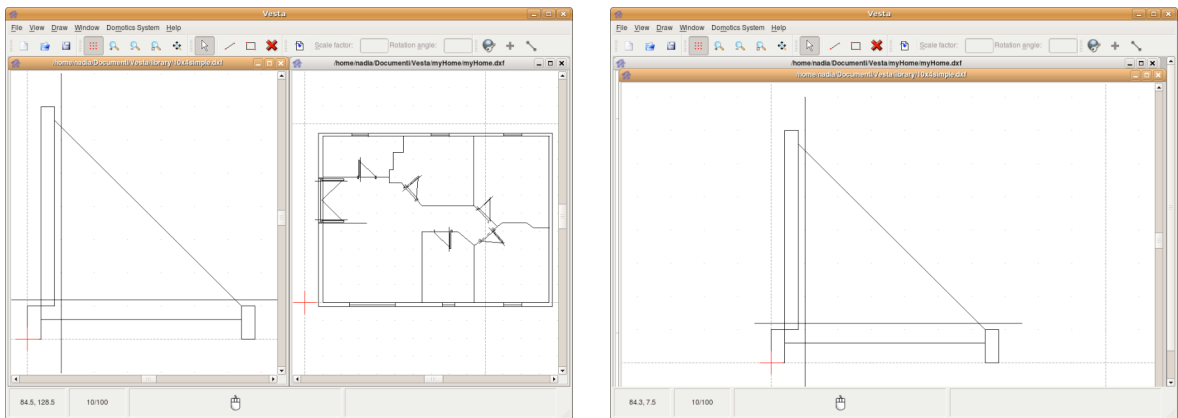


Figure 3.6b e 3.6c: La visualizzazione a finestra separata e a cascata

3.4.3 Attivare una finestra

Menu:

Window → *<number><document_name>*

Scorciatoie da tastiera:

Alt+W+*<number>* | Ctrl+Tab

Abilitazione del comando:

Fase di disegno della modalità di configurazione

Quando è aperta più di una finestra e non sono tutte visibili contemporaneamente, si può attivare una finestra non visibile utilizzando i suddetti comandi. A seconda dell'ordine della creazione delle finestre, viene attribuito un numero *<number>*, che viene visualizzato all'interno del menu *Window* accanto al nome *<document_name>* del file aperto nella finestra. Tale numero può essere utilizzato per identificare una finestra nella scorciatoia da tastiera.

Se invece di attivare direttamente una finestra si vuole sfogliare nell'ordine di creazione tutte quelle attive, si può usare la combinazione di tasti *Ctrl+Tab*.

3.5 La visualizzazione dei progetti

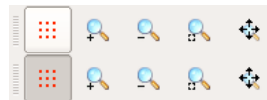
Il menu di visualizzazione e la barra degli strumenti offrono alcune funzioni per regolare la vista corrente del progetto. Tutti questi strumenti non hanno effetti sulla geometria delle entità del disegno ma modificano soltanto il fattore di ingrandimento e l'area inquadrata.

3.5.1 Lo scorrimento della vista

Per spostare la vista del progetto si possono usare le barre di scorrimento verticale e orizzontale poste rispettivamente a destra e in basso all'area di disegno. Se il mouse è dotato di rotella, questa può essere utilizzata per far scorrere il disegno in alto e in basso.

3.5.2 Attivare/disattivare la griglia

Barra degli strumenti:



Menu:

View → Grid | Alt+V+G

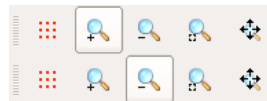
Abilitazione del comando:

Sempre abilitato

Mostra o nasconde la griglia del progetto corrente.

3.5.3 Stringere/allargare l'inquadratura

Barra degli strumenti:



Menu:

View → Zoom in
View → Zoom out

Scorciatoie da tastiera:

Alt+V+I | +
Alt+V+O | -

Abilitazione del comando:

Sempre abilitato

Questa funzione aumenta o diminuisce il fattore di vista corrente di 1,2. Lo stesso effetto si può ottenere tenendo premuto il tasto `Ctrl` e ruotando la rotella del mouse avanti e indietro.

3.5.4 L'inquadratura automatica

Barra degli strumenti:



Menu:

View → Auto zoom

Scorciatoie da tastiera:

Alt+V+A | az

Abilitazione del comando:

Sempre abilitato

3. VESTA: la semplicità al servizio della domotica

Scala la vista del progetto in modo che tutte le entità del disegno vengano visualizzate all'interno dell'area di disegno o visualizzazione.

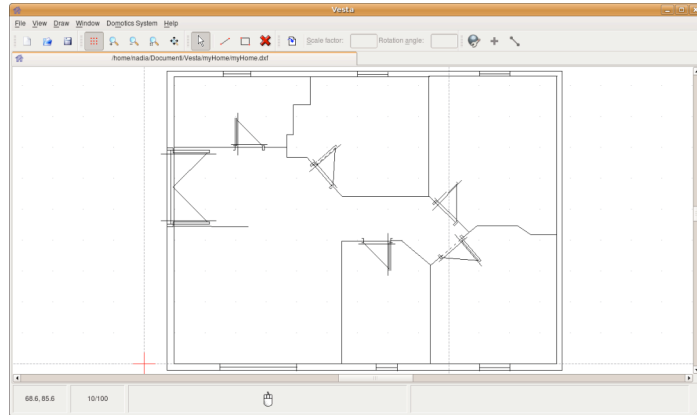
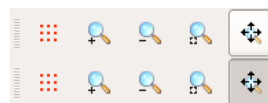


Figura 3.7: L'inquadratura automatica

3.5.5 L'inquadratura panoramica

Barra degli strumenti:



Menu:

View → Pan zoom

Scorciatoie da tastiera:

Alt+V+P | pz

Abilitazione del comando:

Sempre abilitato

Aspetto del cursore del mouse:



Inquadrare in modo panoramico significa muovere l'inquadratura sopra il disegno. Una volta attivato il comando, il cursore del mouse cambia aspetto; premere il tasto sinistro del mouse e spostare l'inquadratura tenendolo premuto. Al termine dell'operazione disattivare il comando eseguendolo nuovamente.

3.6 I comandi per il disegno

In questa sezione verranno descritti gli strumenti messi a disposizione da *Vesta* per disegnare gli oggetti grafici, ovvero le entità. Una volta disegnate, queste possono essere modificate o cancellate.

3.6.1 La linea

Barra degli strumenti:



Menu:

Draw → Line

Scorciatoie da tastiera:

Alt+D+L | Ctrl+L

Abilitazione del comando:

Fase di disegno della modalità di configurazione

Aspetto del cursore del mouse:



Quando viene attivato il comando *Line*, il cursore assume l'aspetto sopra mostrato ed è possibile disegnare una o più linee passante per due punti, seguendo la seguente procedura:

- specificare il primo punto della linea utilizzando il tasto sinistro del mouse;
- spostare il cursore fin dove si desidera e specificare il secondo punto del segmento di linea;
- specificare i punti finali di eventuali altri segmenti;
- terminare la linea premendo il tasto destro del mouse;
- ripetere la procedura da 1 a 4 per eventuali altre linee;
- disattivare il comando *Line* premendo il tasto destro del mouse o eseguendo il comando *Select*.

3.6.2 Il rettangolo

Barra degli strumenti:



Menu:

Draw → Rectangle

Scorciatoie da tastiera:

Alt+D+R | Ctrl+R

Abilitazione del comando:

Fase di disegno della modalità di configurazione

Aspetto del cursore del mouse:



Quando viene attivato il comando *Rectangle*, il cursore assume l'aspetto sopra mostrato ed è possibile disegnare uno o più rettangoli, seguendo la seguente procedura.

3. VESTA: la semplicità al servizio della domotica

- 1 Specificare il primo angolo del rettangolo;
- 2 spostare il cursore nella direzione dell'angolo opposto e specificare il secondo angolo;
- 3 ripetere la procedura da 1 a 2 per tutti i rettangoli che si desidera disegnare;
- 4 disattivare il comando Rectangle premendo il tasto destro del mouse o eseguendo il comando Select.

3.6.3 Selezionare le entità

Barra degli strumenti:



Menu:

Draw → Select

Scorciatoie da tastiera:

Alt+D+S

Abilitazione del comando:

Fase di disegno della modalità di configurazione

Aspetto del cursore del mouse:



Il comando *Select* è attivo di default in fase di disegno e indica che si è in modalità di selezione, ovvero che non si stanno disegnando né linee, né di rettangoli ed è abilitata la selezione delle entità. Il cursore ha l'aspetto standard.

Select, *Line* e *Rectangle*, sono mutuamente esclusivi, pertanto per attivare questo comando si possono utilizzare le indicazioni utilizzate in questo paragrafo oppure disattivare il comando di disegno linea o rettangolo correntemente attivo.

La selezione delle entità serve per poter identificare un sottoinsieme di esse che dovranno rispondere a uno o più comandi successivi; nella grafica esse vengono evidenziate con il colore blu, una linea tratteggiata e dei quadratini blu alle estremità della linea o nei quattro angoli del rettangolo (maniglie per la modifica).

Per poter selezionare una entità occorre premere il tasto sinistro del mouse sulla sua area. Nel pannello della selezione della barra di stato viene visualizzato quale entità è stata selezionata, la lunghezza nel caso di una linea e le dimensioni (larghezza x altezza) nel caso di un rettangolo.

3. VESTA: la semplicità al servizio della domotica

Le operazioni abilitate su un'entità selezionata sono lo spostamento, la modifica delle dimensioni e la cancellazione.

Per selezionare più di una entità occorre disegnare, tenendo premuto il tasto sinistro del mouse, un rettangolo ideale che le contenga graficamente (*rubber band*). Queste potranno essere spostate contemporaneamente o cancellate. Nel pannello della selezione non compare alcuna informazione.

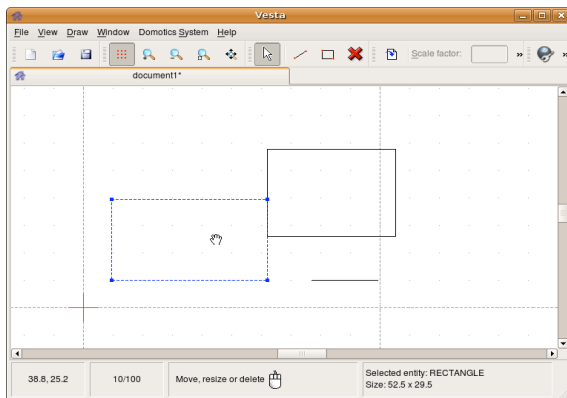


Figura 3.8a: La selezione di una entità

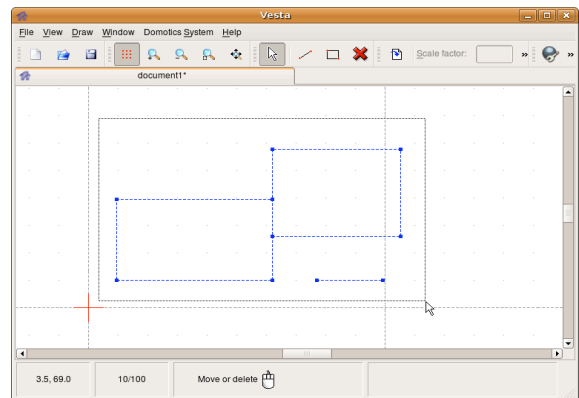


Figura 3.8b: La selezione con *rubber band*

Per deselezionare una o più entità è sufficiente premere il tasto sinistro del mouse quando il puntatore è fuori dall'area di ogni entità.

3.6.4 Spostare le entità

Abilitazione del comando:

Fase di disegno della modalità di configurazione

Aspetto del cursore del mouse:



Lo spostamento delle entità viene eseguito esclusivamente attraverso l'uso del mouse.

Dopo aver selezionato una o più entità è possibile spostarle all'interno dell'area di disegno tenendo premuto il tasto sinistro del mouse sopra una di esse e spostando il puntatore fino alla posizione desiderata.

Quando il puntatore del mouse passa sull'area di una entità selezionata, il cursore assume l'aspetto di una mano aperta (fig 3.8a); quando si trascinano le entità la

3. VESTA: la semplicità al servizio della domotica

mano si chiude, come illustrato sopra; altrove il cursore è quello standard.

3.6.5 Modificare le entità

Abilitazione del comando:

Fase di disegno della modalità di configurazione

Aspetto del cursore del mouse:



Anche la modifica delle entità, come lo spostamento, può essere effettuata solo attraverso l'uso del mouse.

Quando si seleziona una entità e si passa il puntatore del mouse su una delle maniglie (quadratinetti blu) poste alle estremità della figura, il mouse assume l'aspetto sopra indicato.

Premendo sopra di essa con il tasto sinistro del mouse e spostando il puntatore tenendo premuto, si può ridimensionare l'entità. Le modifiche della lunghezza della linea e della larghezza e altezza del rettangolo, verranno visualizzate nel pannello della selezione della barra di stato.

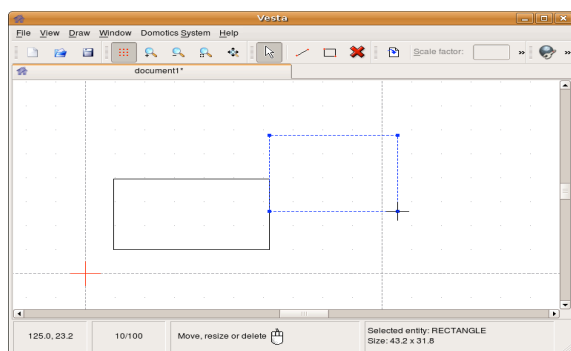


Figura 3.9a: Entità prima della modifica

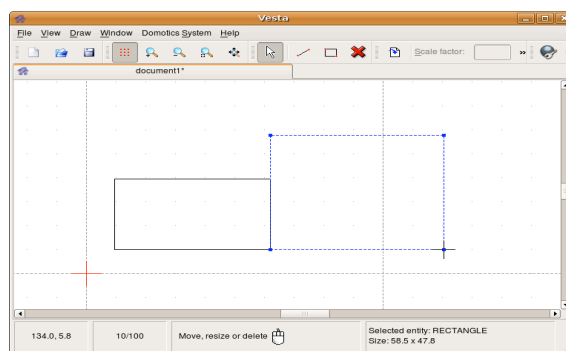


Figura 3.9b: Entità dopo la modifica

3.6.6 Cancellare le entità

Barra degli strumenti:



Menu:

Draw → Delete selected entities

Scorciatoie da tastiera:

Alt+D+D | Canc

Abilitazione del comando:

Fase di disegno della modalità di configurazione

Cancella le entità selezionate.

3.7 La gestione dei blocchi

I blocchi sono insiemi di entità che vengono considerati come se fossero un'unica entità. Vengono utilizzati per non dover ridisegnare parti del disegno che si ripetono all'interno del progetto.

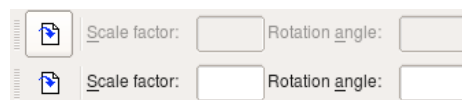
3.7.1 Creare un nuovo blocco

Per creare un nuovo blocco si procede come segue:

- 1 aprire un nuovo progetto;
- 2 disegnare le entità che dovranno far parte del blocco;
- 3 salvare il progetto nella cartella *library* della cartella in cui è installata l'applicazione;
- 4 chiudere il progetto.

3.7.2 Importare un blocco esistente

Barra degli strumenti:



Menu:

Draw → Import existing block...

Scorciatoie da tastiera:

Alt+D+B | Ctrl+B

Alt+S | Alt+A

Abilitazione del comando:

Fase di disegno della modalità di configurazione

Aspetto del cursore del mouse:



Quando si ha l'esigenza di dover disegnare un gruppo di entità in più punti del progetto, è possibile creare un blocco come spiegato nel paragrafo precedente e importarlo attraverso questo comando.

L'importazione del blocco consiste di più operazioni che verranno specificate di seguito.

3. VESTA: la semplicità al servizio della domotica

- 5 Attraverso la finestra di dialogo che compare, scegliere quale blocco della cartella *library* si vuole importare;
- 6 indicare nelle caselle di testo che si sono abilitate nella barra degli strumenti *Block*, le opzioni di inserimento relative al fattore di scala e all'angolo di rotazione con cui deve essere disegnato il blocco; se non si indica niente verranno usati il fattore di scala e l'angolo originali del blocco;
- 7 specificare col tasto sinistro del mouse il punto dell'area di disegno in cui si vuole inserire il blocco;
- 8 ripetere i punti 2 e 3 per tutte le copie del blocco che si desidera inserire;
- 9 premere il tasto destro del mouse per terminare l'inserimento dei blocchi.

Si tenga presente che per inserire i valori nelle suddette caselle di testo, occorre attivare al loro interno il cursore di inserimento di testo; per far ciò, si può sia premere il tasto sinistro del mouse nella casella desiderata, che usare una scorciatoia da tastiera indicata sopra.

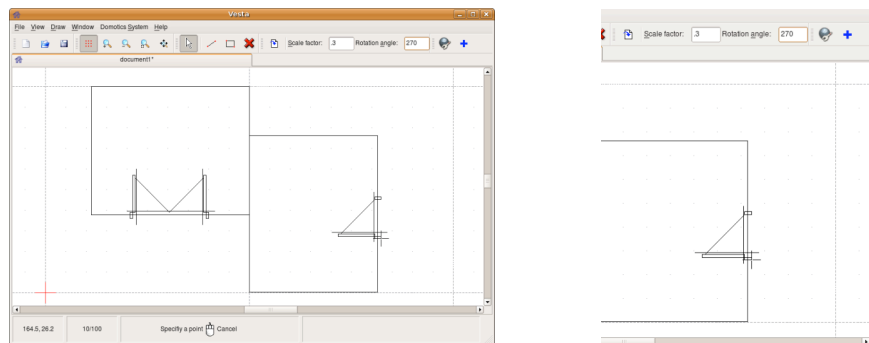


Figura 3.10: L'importazione di un blocco esistente con dettaglio

3.7.3 Selezionare, spostare e cancellare un blocco

Il blocco è considerata un'entità a tutti gli effetti, pertanto si può selezionare, spostare e cancellare.

Quando viene selezionato un blocco, nel pannello della selezione della barra di

3. VESTA: la semplicità al servizio della domotica

stato compaiono le informazioni che lo riguardano: le dimensioni del rettangolo più piccolo che contiene tutte le entità del blocco, il fattore di scala e l'angolo di rotazione utilizzati in fase di inserimento del blocco.

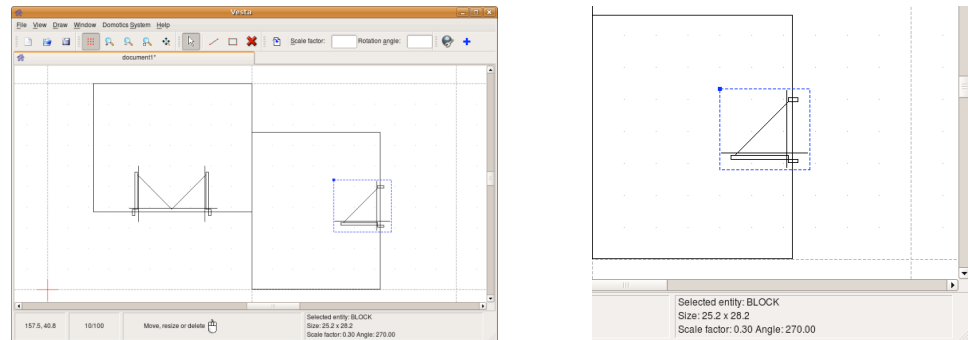


Figura 3.11: La selezione di un blocco con dettaglio

3.7.4 Modificare un blocco

Quando si seleziona un blocco si possono fare due tipi di modifiche:

- ridimensionare il rettangolo che contiene il blocco, con l'unica differenza che vi è disponibile una sola maniglia; tutte le entità del blocco verranno scalate in modo da rientrare nelle nuove dimensioni del rettangolo, che verranno visualizzate nel pannello della selezione della barra di stato;
- cambiare il fattore di scala e l'angolo di rotazione come viene fatto nell'importazione del blocco.

Se si vuole modificare le entità che costituiscono il blocco, si deve cancellare il blocco dal progetto, creare un nuovo blocco, e importarlo nuovamente.

3.8 La configurazione del sistema domotico

Quando si termina il disegno della pianta dell'abitazione, si passa alla fase di configurazione dei dispositivi e delle relazioni che vi sono fra essi.

3.8.1 Configurare i dispositivi

Barra degli strumenti:



3. VESTA: la semplicità al servizio della domotica

Menu:	Domotics System → Configure
Scorciatoie da tastiera:	Alt+M+C
Abilitazione del comando:	Modalità di configurazione

Quando si esegue questo comando si inizia la fase di configurazione, che è una procedura complessa, costituita da una serie di operazioni descritte di seguito.

- 1 Se vi sono altri progetti (non configurati) aperti, l'applicazione richiede di salvarli e chiuderli, perché in questa fase può gestire solo un progetto per volta.
- 2 A questo punto *Vesta* si connette al sistema domotico⁵ che si vuole configurare, recupera una lista contenente tutti i dispositivi che si possono rilevare nel sistema e li visualizza in una *dock window*⁶ sotto forma di lista consumabile. Ogni dispositivo è descritto da un'icona, un'identificatore univoco, il tipo di dispositivo e la sua descrizione (fig. 3.12).

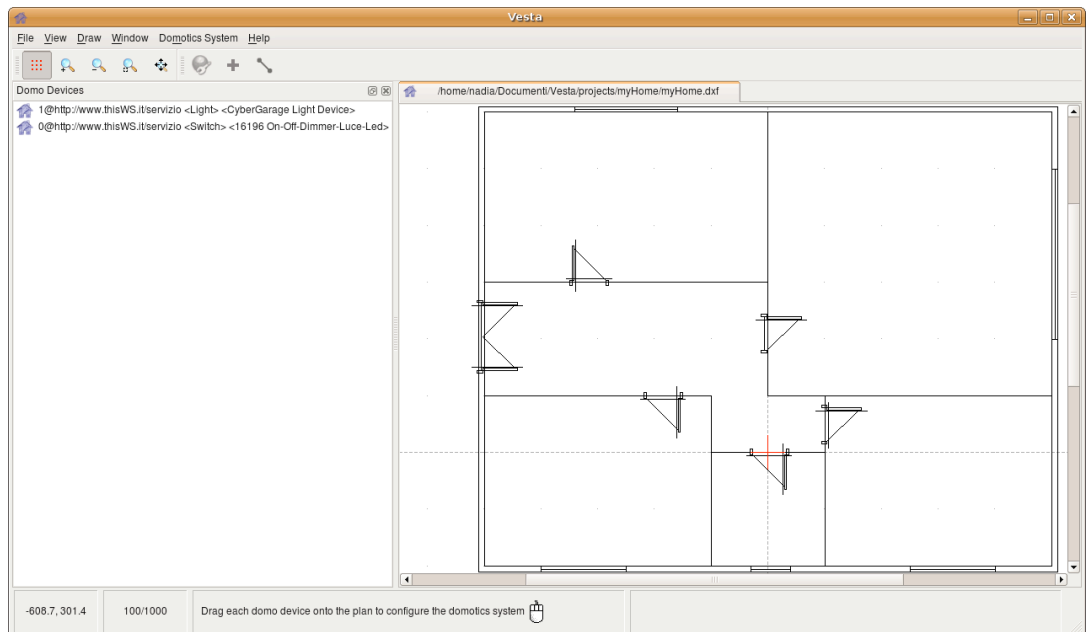


Figura 3.12: *Dock window* contenente la lista dei dispositivi domotici

3 Disporre sulla pianta della casa un dispositivo scelto dalla lista, utilizzando

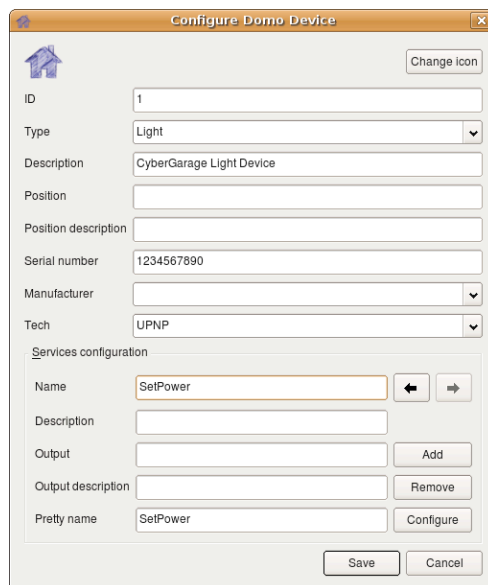
- 5 *Vesta* si connette a *domoNet*, che permette l'interoperabilità del sistema domotico che si vuole configurare.
- 6 Finestra di utilità integrata nella finestra principale dell'applicazione, che può anche essere staccata o spostata nelle aree consentite attorno all'area centrale della finestra principale.

3. VESTA: la semplicità al servizio della domotica

l'operazione di *drag and drop*: premere il tasto sinistro del mouse su un dispositivo della lista e trascinarlo, tenendo premuto, nella posizione desiderata sulla pianta. Nel progetto, il dispositivo viene identificato da un'icona che può essere spostata.

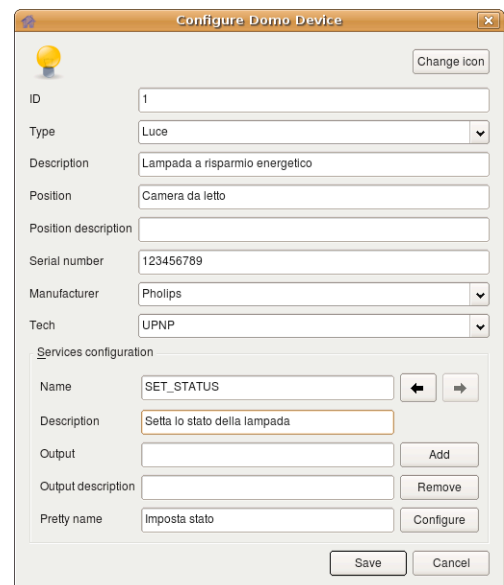
- 4 Procedere alla configurazione del dispositivo utilizzando la finestra di dialogo *Configure Domo Device* che compare quando si termina il trascinamento (figg. 3.13a e 3.13b).

Nella prima sezione della finestra di dialogo, ci sono le informazioni del dispositivo che sono state rilevate dal sistema e l'icona attribuita di default. Tutte le caselle possono essere modificate e attraverso il pulsante *Change icon* è possibile assegnare una nuova icona al dispositivo.



The screenshot shows the 'Configure Domo Device' dialog box with a house icon. The fields are: ID: 1, Type: Light, Description: CyberGarage Light Device, Position: (empty), Position description: (empty), Serial number: 1234567890, Manufacturer: (empty), Tech: UPNP. The 'Services configuration' section has: Name: SetPower, Description: (empty), Output: (empty), Output description: (empty), Pretty name: SetPower. Buttons include 'Change icon', 'Save', 'Cancel', and 'Configure'.

Figura 4.13a: Finestra di dialogo per configurare un dispositivo, con i dati rilevati dal sistema



The screenshot shows the 'Configure Domo Device' dialog box with a lightbulb icon. The fields are: ID: 1, Type: Luce, Description: Lampada a risparmio energetico, Position: Camera da letto, Position description: (empty), Serial number: 123456789, Manufacturer: Philips, Tech: UPNP. The 'Services configuration' section has: Name: SET_STATUS, Description: Setta lo stato della lampada, Output: (empty), Output description: (empty), Pretty name: Imposta stato. Buttons include 'Change icon', 'Save', 'Cancel', and 'Configure'.

Figura 4.13b: Finestra di dialogo per configurare un dispositivo, con i dati modificati dall'utente

Nella seconda sezione si possono configurare i servizi offerti dal dispositivo. Il pulsante *Add* aggiunge un servizio; i pulsanti che rappresentano le frecce, permettono di scorrere i vari servizi, e per ognuno di questi è possibile modificarne le informazioni o rimuoverli. Il pulsante *Configure* invece apre

3. VESTA: la semplicità al servizio della domotica

la finestra di dialogo *Configure Service* che permette di configurare gli input del servizio con i relativi valori ammessi.

- 5 Se si è premuto il pulsante *Configure* del dialogo *Configure Domo Device*, procedere alla configurazione del servizio utilizzando la finestra *Configure Service* (figg. 3.14a e 3.14b). In questo dialogo, nella sezione *Inputs configuration* è possibile aggiungere (pulsante *Add*), rimuovere (pulsante *Remove*) input dal servizio, modificarne le informazioni e per ognuno di essi è possibile aggiungere (pulsante *Add*) e rimuovere (pulsante *Remove*) i valori ammessi nella sezione *Allowed values configuration*.

Premere il pulsante *Save* per salvare la configurazione del servizio o *Cancel* per annullare.

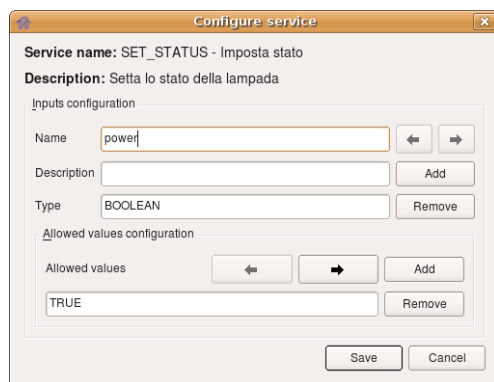


Figura 3.14a: Finestra di dialogo per configurare un servizio, con i dati rilevati dal sistema

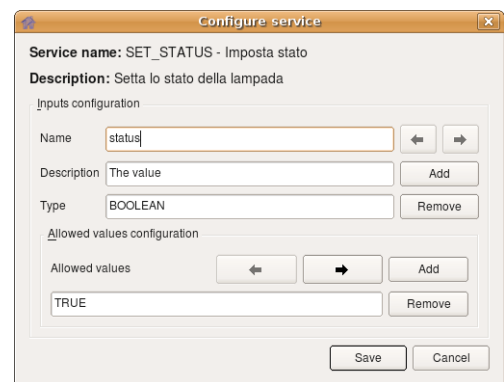


Figura 3.14b: Finestra di dialogo per configurare un servizio, con i dati modificati dall'utente

- 6 Ripetere la procedura del punto 5 per ogni servizio del dispositivo che si vuole configurare.
- 7 Salvare la configurazione del dispositivo utilizzando il pulsante *Save* del dialogo *Configure Domo Device*.
- 8 Ripetere le operazioni dal punto 3 al punto 7 per tutti i dispositivi presenti nella lista.

3.8.2 Aggiungere un dispositivo

Barra degli strumenti:



Menu:

Domotics System → Add domo device...

Scorciatoie da tastiera:

Alt+M+A

Abilitazione del comando:

Modalità di configurazione: in seguito alla configurazione di tutti i dispositivi

Può succedere che qualche dispositivo installato, a causa della tecnologia su cui si basa, non venga rilevato dal sistema. In questo caso si può utilizzare questo comando per aggiungerlo.

La finestra di dialogo che compare (*Add Domo Device*), è molto simile a quella utilizzata per la configurazione dei dispositivi, a parte che tutti i campi sono vuoti perché il sistema non ne ha cognizione. Seguire la procedura descritta nel paragrafo precedente dal punto 4 al punto 7. Una volta inserite tutte le informazioni, premere il pulsante *Add*, e dopo che la finestra si è chiusa, specificare con il puntatore del mouse in quale punto della pianta posizionare il nuovo dispositivo.

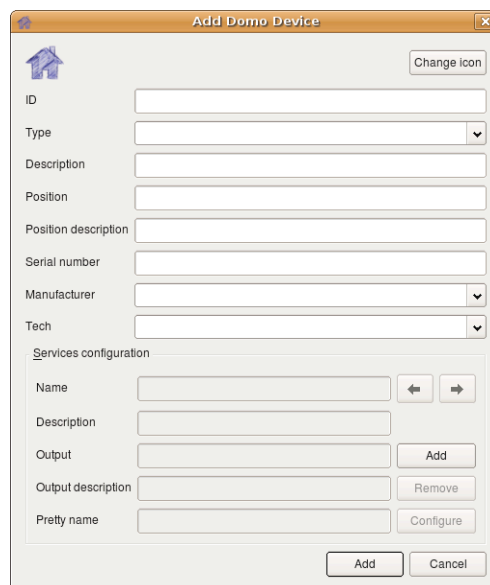


Figura 3.15: Finestra di dialogo per aggiungere un dispositivo

3.8.3 Configurare le relazioni tra i dispositivi

Barra degli strumenti:



Menu:

Domotics System → Configure linked services...

Scorciatoie da tastiera:

Alt+M+L

Abilitazione del comando:

Modalità di configurazione: in seguito alla configurazione di tutti i dispositivi

Quando si termina la configurazione di tutti i dispositivi domotici, è possibile stabilire delle relazioni tra i servizi di due dispositivi, in modo che quando viene eseguito il servizio del primo, viene eseguito anche quello del dispositivo collegato.

Per accedere a questa fase, bisogna invocare questo comando, e confermare di voler proseguire; dopo la conferma non è più possibile spostare le icone dei dispositivi sul progetto. La configurazione procede nella maniera seguente:

- 1 Utilizzando il tasto sinistro del mouse, premere sull'icona di un dispositivo e spostare il puntatore tenendo premuto, nella direzione del dispositivo da collegare. La relazione viene indicata da una linea temporanea, che scompare al termine della configurazione.
- 2 Appena si rilascia il tasto del mouse sul dispositivo da collegare, si apre la finestra di dialogo *Configure Linked Services*.

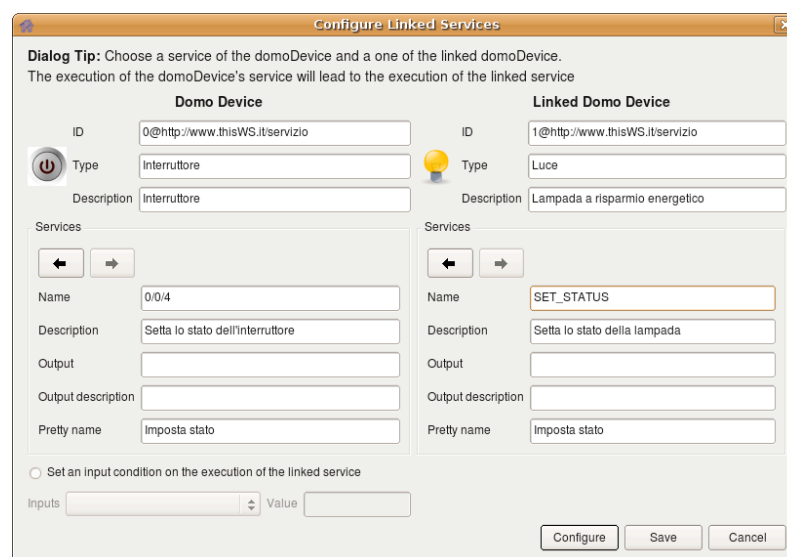


Figura 3.16: Finestra di dialogo per configurare il collegamento dei servizi di due dispositivi

3. VESTA: la semplicità al servizio della domotica

La finestra è suddivisa in due parti, il lato intitolato *Domo Device* è dedicato al dispositivo da cui parte la relazione, mentre quello intitolato *Linked Domo Device*, riguarda il dispositivo da collegare; entrambi contengono le informazioni più importanti per identificarli. Nelle rispettive sezioni *Services*, si possono scorrere i servizi dei due dispositivi. Una volta scelto un servizio del *Domo Device* e uno del *Linked Domo Device*, bisogna premere il pulsante *Configure* per procedere alla configurazione della relazione.

Se è necessario, è possibile imporre una condizione sull'esecuzione del servizio collegato, in modo che questo venga eseguito solo se un input del primo servizio assume un determinato valore indicato dall'utente. Per fare ciò, bisogna spuntare la richiesta di condizione, scegliere un input dalla casella di riepilogo e assegnare un valore nella casella di testo relativa.

- 3 Quando si apre la finestra di dialogo *Configure Linked Inputs*, per ogni input del servizio da collegare, bisogna indicare un valore. Questo può essere il valore di un input del servizio da cui parte la relazione, oppure un valore fisso indicato dall'utente. Per salvare l'associazione, va premuto il pulsante *Bind*.

Configure Linked Inputs

Dialog Tip: For each input of the linked service, choose an input of the service or set a value, then click the Bind button.

Linked Service	Service
Name: SET_STATUS - Imposta stato	Name: 0/0/4 - Imposta stato
Description: Setta lo stato della lampada	Description: Setta lo stato dell'interruttore

Linked inputs

← →

Name: status

Type: BOOLEAN

Description: The value

Inputs

← →

Name: status

Type: BOOLEAN

Description:

Set a value for the linked input

Value:

Save Cancel Bind

Figura 3.17: Finestra di dialogo per configurare gli input di due servizi collegati

3. VESTA: la semplicità al servizio della domotica

- 4 Ripetere la procedura del punto 3 finché il servizio da collegare non ha più input da configurare. Al termine premere il pulsante *Save* che salva e chiude il dialogo.
- 5 Ripetere le operazioni del punto 2 per tutti i servizi che si desidera associare. Al termine premere il pulsante *Save*.
- 6 Ripetere le operazioni dal punto 1 al punto 5 per tutti i dispositivi che si desidera collegare.

3.9 L'interazione con il sistema domotico

Al termine della configurazione delle relazioni tra i dispositivi, si esce dalla modalità di configurazione premendo il tasto destro del mouse. Se si conferma questa volontà, e non è stata una pressione involontaria, *Vesta* entra in modalità di controllo. In questa fase il progetto non è più editabile o modificabile.

3.9.1 Inviare un comando a un dispositivo

Per inviare un comando, premere due volte il tasto sinistro del mouse sull'icona del dispositivo che deve eseguire il comando e si apre la finestra di dialogo *Send Command*.

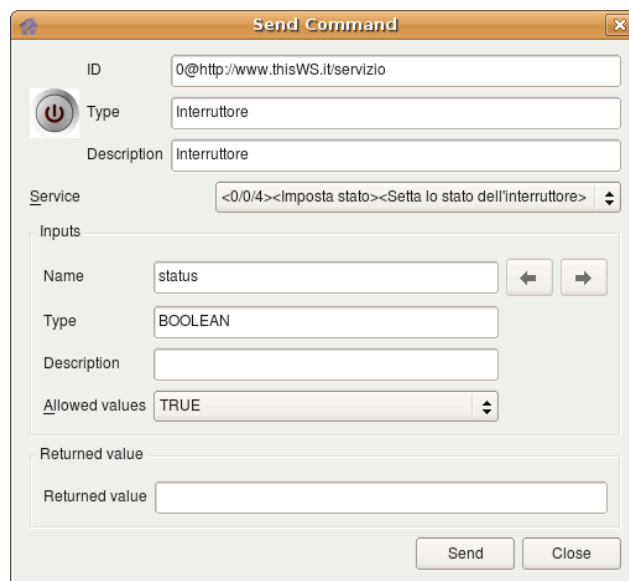


Figura 3.18: Finestra di dialogo per inviare un comando ad un dispositivo

3. VESTA: la semplicità al servizio della domotica

Scegliere il servizio da eseguire e l'input necessario, indicando un valore tra quelli ammessi per l'input scelto. Premere il pulsante *Send* per inviare il comando. Quando arriva l'esito dell'esecuzione e il valore (opzionale) di ritorno, si può chiudere il dialogo premendo il pulsante *Close*.

3.9.2 Visualizzare la cronologia degli eventi

Appena si entra in modalità di controllo, viene visualizzata una *dock window* che contiene gli aggiornamenti di stato dei vari dispositivi. Tale finestra può rimanere ancorata alla finestra principale o può essere staccata per essere una finestra indipendente.

Attraverso il menu *View*, o i pulsantini che ornano la finestra è possibile nasconderla o renderla visibile.

Non appena *Vesta* riceve un aggiornamento di stato dal sistema domotico, nella dock window vengono visualizzate le seguenti informazioni:

- il dispositivo che ha cambiato il suo stato;
- il servizio di tale dispositivo che è stato eseguito;
- il nuovo valore che dell'input associato al servizio.

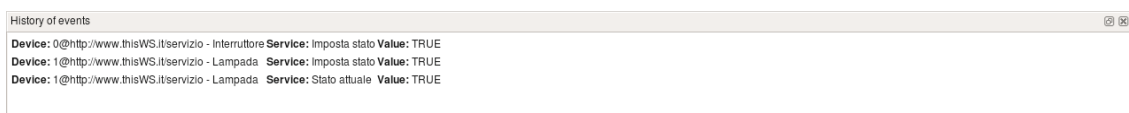


Figura 3.19: *Dock window* che visualizza la cronologia degli eventi del sistema domotico

3.9.3 Visualizzare le informazioni sullo stato del dispositivo

Per visualizzare lo stato di un determinato dispositivo domotico, è sufficiente tenere fermo il puntatore del mouse sull'icona relativa per 2 secondi, e si apre una finestra che ne visualizza le seguenti informazioni:

- identificatore, tipo e descrizione del dispositivo domotico;
- nome e descrizione dei servizi che forniscono un valore di output se eseguiti

3. VESTA: la semplicità al servizio della domotica

(che quindi offrono informazioni sullo stato del dispositivo);

- valore corrente degli input relativi.

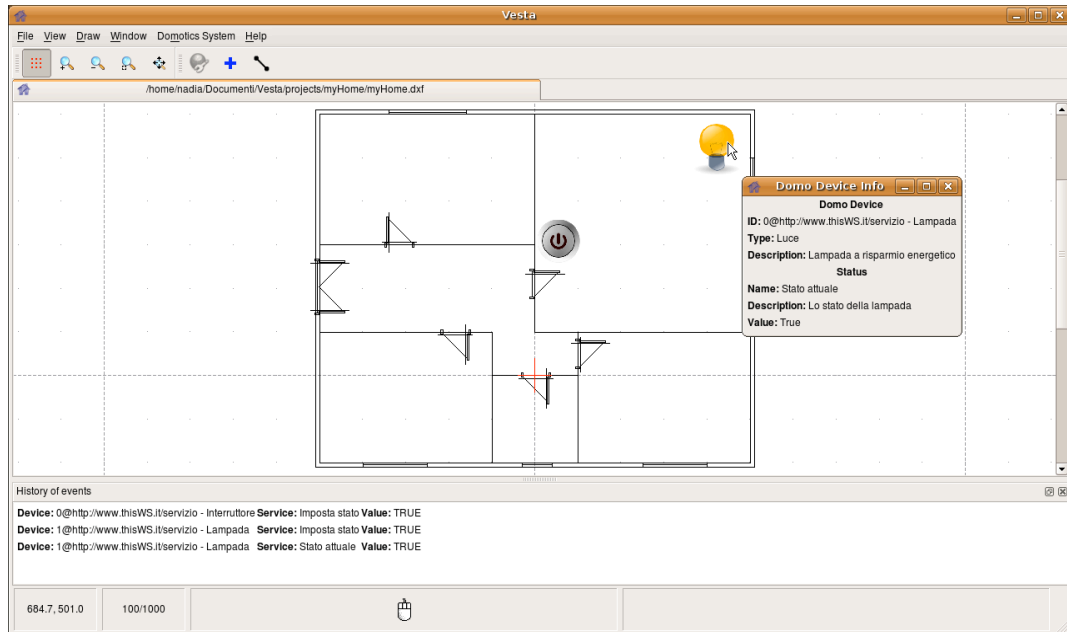


Figura 3.20: Visualizzazione dello stato del dispositivo puntato

4 Il prototipo

Scopo di questo paragrafo è dare al lettore un'idea degli aspetti più significativi e rilevanti dell'implementazione di *Vesta*, mettendo in risalto le scelte implementative fondamentali, rimandando al codice per i dettagli. Pertanto in questo paragrafo si vuole spiegare come si è realizzato il prototipo.

4.1 Strumenti e tecnologie usate

Il prototipo è stato realizzato utilizzando diversi strumenti e tecnologie esclusivamente *open source*. Di seguito vengono descritti brevemente.

4.1.1 C++ e Qt

Il software applicativo *Vesta* è stato implementato utilizzando il linguaggio di programmazione C++ e la libreria grafica Qt (versione 4.4), sviluppata dall'azienda norvegese *Qt Software*¹, originariamente nota come *Trolltech*, dal giugno 2008 di proprietà della *Nokia*.

C++ è un linguaggio di programmazione orientato agli oggetti, estremamente potente ed efficiente; è stato preferito ad altri linguaggi per le sue caratteristiche di efficienza, un aspetto cruciale nella programmazione delle interfacce grafiche, che richiede notevoli risorse di calcolo e di memoria.

QT è una libreria *open source* per lo sviluppo di applicazioni multipiattaforma, maggiormente utilizzata per implementare interfacce utente (*GUI*, *Graphical User*

¹ <http://www.qtsoftware.com/>

Interface), ma anche usata per sviluppare applicazioni *console* e *server*. Qt mette a disposizione, oltre agli elementi grafici necessari a costruire un'interfaccia (*widget*), anche funzioni per l'accesso ai *database*, il *parsing* di documenti *XML*, la gestione dei *file* e del *filesystem*, dei *socket*, e molto altro, tutto indipendentemente dalla piattaforma.

Le librerie Qt sono disponibili per le seguenti piattaforme:

- *Qt/X11* – Qt per X Window System;
- *Qt/Mac* – Qt per Mac OS X Apple;
- *Qt/Windows* – Qt per Microsoft Windows;
- *Qt/Embedded* – Qt per sistemi embedded (palmari e simili);
- *Qt/Jambi* – Qt per Java
- *Qt Extended*² – Qt per dispositivi mobili basati su *Embedded Linux* (Linux per sistemi dedicati).

Qt è scritto in C++, ma supporta anche altri linguaggi, come C, Java, Python, Perl e PHP ecc; è orientato agli oggetti e si basa sul paradigma di programmazione ad eventi, ma la peculiarità di Qt, è il meccanismo di segnalazione *signal/slot*.

Spesso nella programmazione di interfacce grafiche, è necessario notificare ad un oggetto il cambiamento di stato di un altro. La maggior parte delle librerie grafiche usa per questo tipo di comunicazione, le *callback*, ovvero puntatori a funzione; se si vuole che una funzione notifichi il suo stato, bisogna che questa invochi al momento opportuno la *callback* che gli è stata passata come parametro. Questa soluzione ha due limiti: non garantisce che la *callback* verrà richiamata con i parametri giusti, quindi non è *type safe*, e prevede che la funzione richiamante conosca la *callback* da invocare.

Qt invece utilizza un sistema alternativo alle *callback*, i segnali e gli slot. Un segnale viene emesso da un oggetto quando occorre un determinato evento che ne fa

² <http://qtopia.net>

cambiare lo stato in maniera interessante per il resto dell'applicazione; uno slot è una funzione che viene invocata in risposta al segnale. Questo sistema è *type safe*, perché la firma del segnale deve corrispondere a quella dello slot, e quindi il compilatore può segnalare eventuali errori. Inoltre, la classe che emette un segnale non conosce, né si deve occupare dello slot che riceverà il segnale.

Il meccanismo di segnalazione di Qt garantisce che una volta collegato il segnale allo slot, questo verrà invocato con i parametri del segnale al momento giusto.

Tale meccanismo è implementato attraverso un'estensione non-standard del C++ e il *moc* (*meta object compiler*), un programma che viene eseguito prima della compilazione, legge i vari file di intestazione e genera codice C++ standard che contiene il cosiddetto *meta object* per queste classi. Tra le informazioni contenute nel *meta object* vi sono i nomi dei membri dei segnali e gli *slot*, i puntatori a queste funzioni e altre funzioni accessorie come il nome della classe.

4.1.2 dxflib

*dxflib*³ è una libreria *open source* scritta in C++, che permette di leggere e scrivere file DXF senza doverne conoscere il formato.

DXF (*Drawing Interchange Format*, o *Drawing Exchange Format*) è un formato per i file di tipo CAD, sviluppato da *Autodesk*⁴ come soluzione per scambiare dati tra il programma *AutoCAD* e altri programmi CAD.

La libreria è stata sviluppata da *RibbonSoft*⁵, una società svizzera che sviluppa e distribuisce *QCAD*, un'applicazione *open source* per il disegno CAD, che utilizza anche essa la libreria *dxflib*.

dxflib permette di leggere e scrivere le sezioni *header*, *tables*, *blocks*, e *entities* di un file DXF. Le entità principali supportate sono: l'arco di circonferenza, la circonferenza, l'ellisse, la linea, il punto, la polilinea, e il testo.

3 <http://www.ribbonsoft.com/dxflib.html>

4 <http://usa.autodesk.com>

5 <http://www.ribbonsoft.com>

4.1.3 XML

XML (*Extensible Markup Language*) è un linguaggio di marcatura estensibile perché permette all'utente di definire nuovi elementi di marcatura. Lo scopo dell'XML è quello di facilitare lo scambio di dati strutturati, specialmente attraverso Internet, tra applicazioni differenti.

4.1.4 Apache Axis e Web Services

Axis⁶ è un'implementazione *open source* del *Simple Object Access Protocol* (SOAP), sviluppata da *Apache Software Foundation*⁷, sia in Java che in C++.

SOAP è un protocollo per lo scambio di informazioni strutturate (in formato XML) in ambienti distribuiti, utilizzato per l'implementazione dei *Web Service*.

Il *Web Service* definisce un'interfaccia che descrive un insieme di *Services* (servizi/operazioni) accessibili in rete mediante il protocollo SOAP. Il documento che contiene la descrizione dei servizi, il protocollo utilizzato e l'URI del *web service*, è scritto con il linguaggio *WSDL* (*Web Services Description Language*), utilizzando un formato automaticamente elaborabile, cioè l'XML standard.

Attraverso questi strumenti è possibile garantire l'interoperabilità tra applicazioni diverse, in maniera indipendente dalla piattaforma e dai linguaggi di programmazione utilizzati.

4.2 Architettura

L'architettura di *Vesta* è costituita da due moduli.

- Il modulo *GUI*, implementa l'interfaccia grafica, che si occupa di interagire con l'utente in maniera molto semplice e intuitiva. Essa mette a disposizione le funzionalità di *Vesta* attraverso degli elementi grafici che sono familiari all'utente, quali le *finestre*, i *pulsanti*, i *menu* e i *dialoghi*, che sono gli stessi, anche nell'aspetto grafico (*look and feel*), a quelli del sistema operativo

6 <http://ws.apache.org/axis/>

7 <http://www.apache.org/>

utilizzato dall'utente.

- Il modulo *domoNetClient* invece, implementa la logica di interazione con *domoNet* per sfruttarne le funzionalità. Si occupa di tradurre i comandi impartiti dall'utente in maniera grafica, in un linguaggio comprensibile a *domoNet*, e di comunicare con esso, seguendo un protocollo di comunicazione.

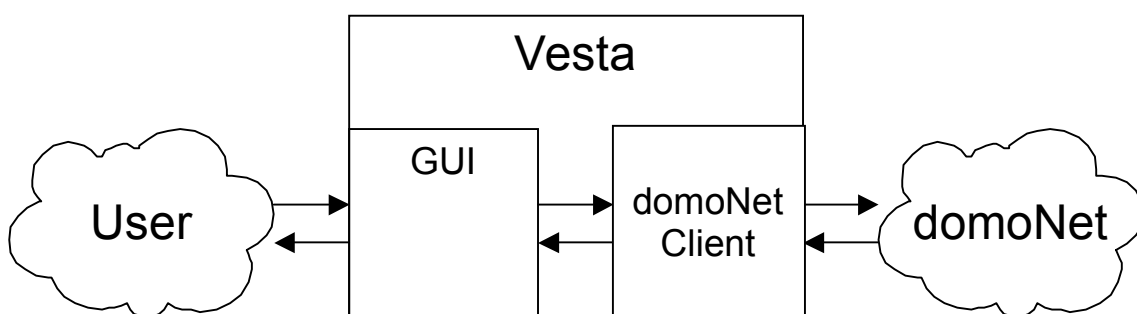


Figura 4.1: L'architettura di *Vesta*

4.3 L'interazione con *domoNet*

Il compito principale di *Vesta* è quello di permettere la configurazione dei dispositivi di un sistema domotico eterogeneo, e il controllo remoto di questi.

L'interoperabilità tra i vari sottosistemi domotici è garantita da *domoNet* che, attraverso il linguaggio *domoML*, permette di descrivere indipendentemente dalla tecnologia su cui si basano, i dispositivi domotici (*domoDevice*) e i messaggi (*domoMessage*) che vengono scambiati fra essi.

Pertanto il *domoNetClient* di *Vesta*, per implementare le suddette funzionalità, necessita di comunicare con *domoNet*, scambiando messaggi che utilizzano il formalismo *domoML*.

4.3.1 *domoML*

domoML si occupa di descrivere i dispositivi gestiti da *domoNet* e le interazioni che vi sono fra essi, attraverso le grammatiche basate sull'XML *domoDevice* e

domoMessage.

Parallelamente a questi due formalismi, il *domoDevice* viene rappresentato in *Vesta* da un oggetto di tipo `DomoDevice`, e il *domoMessage* da un oggetto di tipo `DomoMessage`.

4.3.1.1 **domoDevice**

Un *domoDevice* può essere rappresentato attraverso un albero largo e poco profondo (la grammatica è altamente attributata e con pochi figli). L'albero ha al massimo 4 livelli:

1. `<device>`:

tag che descrive il *domoDevice* utilizzando i seguenti attributi:

- `description`: una descrizione in linguaggio naturale del *domoDevice*;
- `id`: identifica il *domoDevice* all'interno di un *web service*;
- `manufacturer`: il produttore del dispositivo;
- `position`: una descrizione in linguaggio naturale della locazione del dispositivo;
- `positionDescription`: descrizione dettagliata della posizione del dispositivo;
- `serialNumber`: il numero seriale (può essere composto anche da lettere) del dispositivo;
- `tech`: la tecnologia originale del *domoDevice* rappresentato;
- `type`: la tipologia del dispositivo;
- `url`: indirizzo del *web service* che gestisce un dispositivo.

Tutti i campi sono opzionali tranne `id`, `url` e `tech`; gli attributi `id` e `url` insieme identificano univocamente un dispositivo.

2. `<service>`:

descrive un singolo servizio offerto da un dispositivo; per ogni *domoDevice* ci possono essere più servizi. Gli attributi di questo *tag* sono:

- `description`: una descrizione in linguaggio naturale del servizio;
- `name`: identificatore del servizio di un *domoDevice*;
- `output`: tipo del valore che viene restituito in seguito all'esecuzione di un comando su questo servizio; è opzionale;
- `outputDescription`: descrizione in linguaggio naturale dell'attributo `output`; è opzionale;
- `prettyName`: etichetta in linguaggio naturale del servizio.

3. Due possibili *tag* per questo livello:

- `<input>`:

indica che il servizio ha bisogno di un valore di input per poter essere eseguito. Questo *tag* è opzionale e ogni servizio può avere più input. Gli attributi di questo *tag* sono:

- `name`: identificatore dell'input;
- `description`: descrizione dell'input;
- `type`: tipo dell'input atteso.

- `<linkedService>`:

indica il servizio da associare quando questo servizio viene eseguito. Può essere invocato un servizio dello stesso o di qualsiasi altro *domoDevice*. L'associazione di servizi non ha vincoli semantici, ovvero è possibile combinare un servizio con qualsiasi altro, a patto di riuscire a dare dei valori ragionevoli ai `linkedInput` (discusso successivamente). Anche questo *tag* è opzionale e i suoi attributi sono:

- `url`: l'indirizzo del *web service* che gestisce il *domoDevice* da collegare al servizio;
- `id`: l'identificatore del *domoDevice* (all'interno del *web service*) da collegare al servizio;
- `service`: il nome del servizio del *domoDevice* da collegare, individuato dai precedenti attributi;
- `ifInput`: indica l'input del questo servizio (rappresentato dal *tag* di secondo livello), che si vuole sottoporre a condizione;
- `hasValue`: indica il valore da testare sull'input dell'attributo `ifInput`; il servizio collegato viene eseguito solo se la condizione di uguaglianza tra il valore dell'input specificato da `ifInput` e il valore dell'attributo `hasValue` è verificata;

4. In base al *tag* precedente, a questo livello è possibile avere:

- `<allowed>`:
 - se il *tag* precedente è `input`. Rappresenta un possibile valore che l'input può assumere; questo *tag* è opzionale e un input può avere diversi valori ammessi `allowed`. L'unico attributo è:
 - `value`: il valore ammesso in formato stringa.
- `<linkedInput>`:
 - se il *tag* precedente è `linkedService`. Rappresenta l'associazione di un input di questo servizio (rappresentato dal *tag* di secondo livello), con un input del servizio da richiamare. Ogni input del servizio collegato deve avere un'associazione; in questo modo il valore dell'input di questo servizio viene passato come input del servizio collegato. Gli attributi di questo *tag* sono:
 - `from`: il nome dell'input dal quale prendere il valore;

- `to`: il nome dell'input che deve usare il valore preso;
- `value`: indica un valore predefinito da utilizzare se l'attributo `from` è vuoto.

Un esempio di *domoDevice* per una lampadina è il seguente:

```
<device description="lampada a risparmio energetico"
  id="0" manufacturer="philips"
  position="camera da letto"
  positionDescription="comodino accanto al letto"
  serialNumber="xxxxxxxx" tech="KNX" type="lampada"
  url="http://www.questowebsevice.it/servizio">
  <service description="Get the status"
    output="BOOLEAN" outputDescription="The value"
    name="GET_STATUS" prettyName="Get status" />
  <service description="Set the status"
    name="SET_STATUS" prettyName="Set status">
    <input description="The value" name="status"
      type="BOOLEAN">
      <allowed value="TRUE" />
      <allowed value="FALSE" />
    </input>
    <linkedService id="3" service="setPower"
      url="http://www.altrowebsevice.it/servizio">
      <linkedInput from="status" to="power" />
    </linkedService>
  </service>
</device>
```

In questo esempio è possibile vedere che il *domoDevice* descritto dagli attributi di `<device>`, offre due servizi: prendere il suo stato corrente in formato booleano e settare il suo stato attraverso un input, sempre booleano, che può assumere i valori TRUE O FALSE.

Quando viene invocato quest'ultimo servizio, ne viene chiamato anche un altro di nome `setPower` appartenente al *domoDevice* gestito dal *web service* `http://www.altrowebservice.it/servizio` con `id=3`. Il valore dell'input assegnato al primo servizio (`status`) viene assegnato anche al secondo (`power`).

Se viene invocato quindi il servizio `setStatus` del *domoDevice* con `url="http://www.questowebservice.it/servizio"` e con `id=0` passando come valore di input TRUE, viene invocato anche il servizio `setPower` del *domoDevice* `url="http://www.altrowebservice.it/servizio"`, con `id=3` e con valore di input TRUE.

La classe `DomoDevice` che rappresenta un dispositivo domotico, ha un campo per ogni attributo del *tag* `<device>` più un campo che punta ad una lista di puntatori a oggetti di tipo `Service`.

La classe `Service`, alla stessa maniera, possiede un campo per ogni attributo del *tag* `<service>` più due campi puntatore, il primo ad una lista di puntatori ad oggetti di tipo `ServiceInput` e il secondo ad una lista di puntatori ad oggetti di tipo `LinkedException`.

La classe `ServiceInput` ha come campi gli attributi del *tag* `<input>` più un puntatore ad una lista di puntatori ad oggetti di tipo `Allowed`, che è una classe con il solo attributo del *tag* `<allowed>`.

La classe `LinkedException` ha come campi gli attributi del *tag* `<linkedException>` più un puntatore ad una lista di puntatori ad oggetti di tipo `LinkedExceptionInput`, che come `Allowed` possiede solo gli attributi del *tag* `<linkedExceptionInput>`.

Tutte i campi delle suddette classi sono campi di tipo stringa privati, e per

ciascuno di essi vi è un metodo per impostarne il valore e uno per reperirlo.

4.3.1.2 **domoMessage**

Anche il *domoMessage* può essere rappresentato attraverso un albero largo e poco profondo di massimo due livelli:

1. <message>:

è il *tag* che descrive il *domoMessage* e contiene i seguenti attributi:

- `message`: il corpo del messaggio, tipicamente il nome del servizio da invocare;
- `messageType`: il tipo di messaggio. Può essere:
 - `COMMAND`: se si tratta di servizio da eseguire;
 - `SUCCESS`: se è richiesto l'esito successivo al `COMMAND`. In questo caso il campo `message` può contenere il valore di risposta dell'esecuzione del servizio;
 - `FAILURE`: se è richiesto l'esito successivo al `COMMAND`. In questo caso il campo `message` può contenere il codice di errore del fallimento;
 - `UPDATE`: se si tratta di un aggiornamento di stato di un *domoDevice*;
- `receiverId`: l'identificatore del *domoDevice* ricevente del messaggio;
- `receiverURL`: l'url del *web service* che contiene il *domoDevice* richiesto;
- `senderId`: l'identificatore del *domoDevice* mittente del messaggio;
- `senderURL`: l'url del *web service* che contiene il *domoDevice* che invia il messaggio.

2. <input>:

i valori di input da associare al `message`. Questo *tag* è opzionale e ogni *tag* `message` può avere più *tag* `input`. Gli attributi per questo *tag* sono:

- `name`: il nome dell'input;
- `type`: il tipo dell'input;
- `value`: il valore dell'input in formato stringa.

Un esempio di *domoMessage* per accendere una lampadina è:

```
<message message="SET_STATUS" messageType="COMMAND"
  receiverId="1"
  receiverURL="http://www.altrowebservice.it/servizio"
  senderId="0"
  senderURL="http://www.questowebsevice.it/servizio">
  <input name="status" type="BOOLEAN" value="TRUE" />
</message>
```

In questo esempio viene richiesto di eseguire il servizio `SET_STATUS` sul *domoDevice* con web service `http://www.questowebsevice.it/servizio` e `id=1` con input di tipo booleano dal valore `TRUE`.

Eseguito il servizio, è possibile ricevere il seguente messaggio di conferma:

```
<message message="TRUE" messageType="SUCCESS"
  receiverURL="http://www.questowebsevice.it/servizio"
  receiverId="0" senderId="1"
  senderURL="http://www.altrowebservice.it/servizio" />
```

Questo messaggio dà la conferma al mittente che il precedente messaggio è stato eseguito con successo e che la lampada ora è accesa.

La classe `DomoMessage` che rappresenta un messaggio, ha un campo per ogni attributo del *tag* `<message>` più un campo che punta ad una lista di puntatori a oggetti di tipo `MessageInput`.

La classe `MessageInput`, possiede un campo per ogni attributo del `tag <input>`.

Tutte i campi delle suddette classi sono campi di tipo stringa privati, e per ciascuno di essi vi è un metodo per impostarne il valore e uno per reperirlo.

4.3.2 L'implementazione dell'interazione

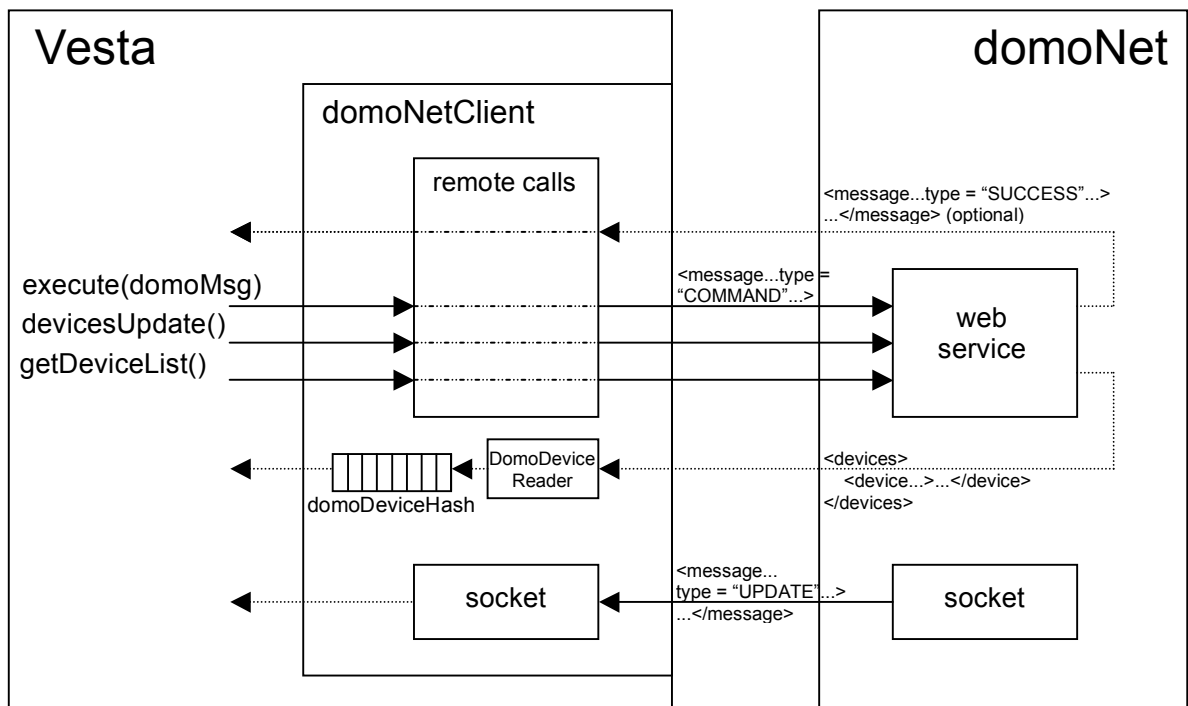


Figura 4.2: L'interazione tra *Vesta* e *domoNet*

L'interazione tra il modulo `domoNetClient` di `Vesta` e `domoNet`, serve per scambiare le seguenti informazioni (fig. 4.2):

- 1 reperire la lista dei dispositivi installati nel sistema domotico che si intende configurare;
- 2 aggiornare le strutture dati di `domoNet` con la nuova descrizione dei dispositivi, generata con la configurazione effettuata dall'utente;
- 3 inviare un comando remoto a un determinato dispositivo;
- 4 ricevere gli aggiornamenti sul cambiamento di stato dei dispositivi gestiti da `domoNet` per visualizzare una cronologia di eventi del sistema.

Le scelte implementative riguardo agli strumenti e le tecnologie da utilizzare per l'interazione su descritta, sono state fatte tenendo presente come viene implementata la comunicazione da *domoNet*, pertanto sono state scelte obbligate.

La comunicazione dei primi tre punti avviene mediante l'invocazione di metodi remoti, cioè dei servizi esposti dal *web service* di *domoNet*; questa tecnologia rende possibile lo scambio di informazioni tra *Vesta* e *domoNet* astruendo dal fatto che le due applicazioni si trovano su due macchine differenti e sono state scritte utilizzando linguaggi di programmazione diversi (*domoNet* è scritta in Java).

Per poter generare l'infrastruttura necessaria al protocollo SOAP sul lato *client* (*stub*), è necessario che sul calcolatore che ospita *Vesta* sia installato *Axis* per C++. Il file *domoNetWs.wsdl* che descrive l'interfaccia del *web service* viene dato in input a uno strumento fornito da *Axis*, *WSDL2Ws*, il quale genera le classi dello *stub* che contengono la definizione locale dei metodi remoti: `DomoNetWS.hpp` e `DomoNetWS.cpp`. A questo punto i metodi/servizi esposti da *domoNet* sono invocabili come se facessero parte integrante di *Vesta*.

1. Durante l'esecuzione di *Vesta* in modalità di configurazione, quando l'utente decide di terminare la fase di disegno e vuol configurare il sistema domotico, il *domoNetClient* deve reperire, per conto della *GUI*, la lista dei dispositivi domotici da configurare. Per fare ciò la *GUI* invoca il metodo `getDeviceList` della classe `DomoNetWS` generata da *Axis*, che restituisce una stringa contenente la concatenazione dei vari *domoDevice* racchiusi dai tag “<devices>...</devices>”.

Tale stringa viene analizzata sintatticamente attraverso il metodo `read(QString domoDeviceList)` della classe `DomoDeviceReader`⁸ e per ogni *domoDevice* viene creato un oggetto `DomoDevice`; il puntatore a questo oggetto viene inserito in una tabella *hash* indicizzata da una stringa che

⁸ Classe derivata da `QXmlStreamReader`, fornita da Qt; è una implementazione del parser XML simile a SAX, che differisce da questa per come i token xml vengono notificati al programmatore. Per dettagli si consulti la documentazione di Qt 4.4.3.

identifica il *domoDevice*, composta dal campo `DomoDevice::id`, il carattere `@` e il campo `DomoDevice::url`.

2. Al termine della configurazione, prima di passare in modalità di controllo, è necessario aggiornare le strutture dati di *domoNet* che salvano i vari *domoDevice*. Per fare ciò, la *GUI* chiede al *domoNetClient* di iterare la sua `domoDeviceHash`, che contiene i vari *domoDevice* configurati, formattare una stringa come quella ricevuta con la richiesta del punto 1, e invocare il metodo remoto `devicesUpdate` passandogli come parametro tale stringa.
3. Un volta entrati in modalità di controllo, l'utente può decidere di inviare un comando ad un determinato dispositivo. Per questa funzionalità, la *GUI*, utilizzando le informazioni immesse dall'utente, prepara una stringa *domoMessage* che rappresenta un messaggio di tipo `COMMAND`. Il *domoNetClient* provvede poi ad invocare il metodo remoto `execute` passandogli come parametro la stringa ricevuta dalla *GUI*. Ottenuta la risposta, questa viene ridata alla *GUI* che la visualizza all'utente.

La comunicazione del punto 4, a differenza delle altre, che usando il protocollo SOAP avvengono al livello di applicazione, avviene tramite l'uso di un *socket*, quindi a un livello più basso dello *stack* protocollare TCP/IP, più precisamente a livello di trasporto.

Un *socket* è un'astrazione software che permette lo scambio di dati tra processi che vengono eseguiti su calcolatori diversi all'interno di una rete. A seconda del protocollo di trasporto utilizzato si distinguono in *socket TCP* e *socket UDP*.

- 4 *domoNet* possiede un cosiddetto *server socket*, ovvero *socket* disposto ad accettare connessioni entranti da processi remoti. Esso è identificato da tre parametri: il protocollo di trasporto utilizzato, l'indirizzo IP del calcolatore su cui viene eseguita *domoNet* e il numero di porta su cui si mette in ascolto.

Quando *Vesta* passa in modalità di controllo, la *GUI* incarica il *domoNetClient* di creare un nuovo *thread* (`Listener`), che si occupa di fare le seguenti cose:

- creare un *socket TCP* e richiedere una connessione al *server socket* di *domoNet* attraverso il metodo `connectToHost` che ha bisogno di due parametri: l'indirizzo IP del calcolatore su cui viene eseguito *domoNet* e il numero di porta; queste informazioni vengono indicate dall'utente per mezzo della *GUI*;
- una volta accettata la connessione, il *thread* viene sospeso finché il *socket* non emette un segnale che notifica la presenza di dati nel *socket* (*lettura bloccante*). In tal caso i dati vengono letti ed elaborati e viene rifatta una lettura bloccante, fino alla prossima notifica.

Quando *domoNet* rileva un cambiamento di stato di un dispositivo, formatta una stringa *domoMessage* di tipo `UPDATE`, e la scrive nel suo *socket*. Il *thread* del *domoNetClient* viene risvegliato, legge la stringa dal *socket* ed effettua un'analisi sintattica per creare un oggetto di tipo *domoMessage* ed aggiornare lo stato del *domoDevice* nella `domoDeviceHash`. Il *domoMessage* viene poi passato alla *GUI* che visualizzerà un aggiornamento della cronologia degli eventi del sistema domotico.

4.4 L'implementazione della GUI

4.4.1 La finestra principale

La finestra principale è il cuore di tutta l'applicazione. Appena l'utente avvia *Vesta*, la classe che contiene il metodo `main` crea un oggetto di tipo `MainWindow` e lo visualizza. Poi, come in tutte le applicazioni grafiche, viene avviato il cosiddetto *event loop*, una sequenza di istruzioni che viene iterata per tutta l'esecuzione, che ha il compito di catturare gli eventi che si verificano e notificarli agli oggetti che devono gestire tali eventi. Qt nasconde la gestione dell'*event loop* nell'invocazione del metodo `exec` della classe `QApplication`.

La finestra principale è costituita da un *titolo* (il nome dell'applicazione), *un'icona* e da tanti oggetti grafici (*widgets*), che vanno costruiti attraverso i seguenti metodi

della classe `MainWindow`:

- `createActions`: le azioni, rappresentate dalla classe `QAction`, sono un'astrazione dei comandi che l'utente può dare all'applicazione. Poiché molti di questi possono essere invocati via menu, barra degli strumenti o scorciatoia da tastiera, per garantire all'utente la stessa esecuzione indipendentemente dal mezzo usato e per tenerli sincronizzati, si definisce dapprima un'azione, che poi può essere aggiunta al menu, o alla barra degli strumenti. Ad un'azione può essere associata un'icona, un testo, una scorciatoia da tastiera e un messaggio di aiuto da visualizzare nella barra di stato.
- `createToolBars`: costruisce le varie barre degli strumenti e aggiunge le rispettive azioni.
- `createStatusBar`: costruisce la barra di stato con i vari pannelli.
- `setCentralWidget (mdiArea)`: assegna l'oggetto grafico passato come parametro (discusso in seguito) all'area centrale della finestra principale.

Per ogni azione viene stabilito un collegamento tra il segnale che viene emesso quando il comando relativo viene invocato, e uno *slot*, cioè una funzione privata che lo gestisce. Così ad esempio il segnale di esecuzione dell'azione *new* viene collegata allo slot `new`, che si occupa di creare una nuova finestra che visualizza un nuovo documento all'interno della finestra principale.

Come per le azioni questo viene fatto per tutti i *widget* che devono segnalare un evento a un altro oggetto.

L'area centrale della finestra principale è un oggetto `QMdiArea`, cioè un gestore di sottofinestre. Un'applicazione come *Vesta* è definita *MDI (Multiple Document Interface)*, perché riesce a gestire più documenti contemporaneamente che condividono la stessa finestra principale, i menu, le barre degli strumenti e la barra di stato.

4.4.2 L'importazione e il salvataggio di file DXF

Per poter leggere e visualizzare un file DXF generato dalle applicazioni CAD, e successivamente salvarlo, viene utilizzata la libreria *dxLib*.

Per importare un file DXF, è stata implementata la classe `DxfLoader` che deriva dalla classe `DL_CreationAdapter` della libreria *dxLib*, e definisce i metodi per leggere le linee e i blocchi: `addLine` e `addBlock`. In questi metodi i dati che caratterizzano queste entità vengono utilizzati per visualizzarli nell'area di disegno.

Quindi per leggere un file DXF è sufficiente creare un oggetto `DL_Dxf` (sempre della libreria *dxLib*) e invocare su di esso il metodo `in` passandogli come parametri un riferimento al file da leggere e ad un oggetto di tipo `DxfLoader`.

Per salvare un disegno in formato DXF 2000/2002 invece, occorre invocare il metodo `out` su un oggetto `DL_Dxf`, passando come parametro il nome del file su cui scrivere e il codice della versione del formato DXF.

L'oggetto `DL_Dxf` possiede un metodo per scrivere ciascuna sezione del formato DXF, e le varie entità, incapsulate nella classe corrispondente messa a disposizione da *dxLib*.

Vesta supporta solo le linee e i blocchi, pertanto i rettangoli vengono salvati come quattro linee.

4.4.3 Il salvataggio dei progetti

Precedentemente si è fatta la distinzione tra progetto *Non Configurato* e *Configurato*.

Un progetto non configurato è un file DXF che rappresenta la pianta di un'abitazione, mentre il progetto configurato visualizza la pianta e l'ubicazione dei dispositivi domotici identificati da un'icona. Queste ulteriori informazioni non possono essere salvate nel file DXF, perché il formato non è predisposto in tal senso.

Le informazioni di un progetto configurato vanno allora salvate nella maniera seguente.

Quando l'utente crea e salva un nuovo progetto (quindi non è configurato), deve indicare una cartella del *file system* in cui conservare il suo lavoro. Supponiamo che scelga la cartella *projects* situata all'interno della cartella principale dell'applicazione, e che attribuisca il nome *myHome* al progetto. Al momento del salvataggio del file DXF viene creata in *projects* una nuova cartella chiamata *myHome* che conterrà a sua volta il file *myHome.dxf*. Questo non avviene quando si salva un blocco nella cartella *library*.

Se l'utente decide di configurare un sistema domotico e di usare il file *myHome.dxf* come supporto grafico per la configurazione, al termine di questa è necessario salvare ulteriori informazioni quali:

- l'indirizzo IP e la porta utilizzati da *domoNet* per la sua esecuzione;
- il percorso locale in cui sono salvate le icone che rappresentano sulla pianta i dispositivi;
- le coordinate della pianta in cui l'utente ha disposto le icone.

Esse vengono salvate nel file *myHome.xml* che è strutturato secondo la seguente semplice grammatica:

```
<project ip = "indirizzoIPdomoNet" port = "numeroPorta">
  <icon path = "percorsoIcona"
    xCoord = "coordinataX" yCoord = "coordinataY"/>
  <!-- Elencazione di tutte le icone presenti nel progetto -->
</project>
```

Tutte le informazioni che riguardano la configurazione e lo stato dei dispositivi domotici vengono salvate da *domoNet*.

Alla luce di quanto detto, a livello tecnico-implementativo, un progetto configurato viene distinto da uno non configurato dalla presenza di un file XML

nella cartella del progetto, con lo stesso nome di quest'ultima.

4.4.4 L'area di disegno

Vesta è in grado di gestire più documenti contemporaneamente nelle sottofinestre di quella principale. Ogni documento possiede un'area di disegno, in fase di configurazione, o di visualizzazione in modalità di controllo.

A livello implementativo questa differenza non esiste, nel senso che il *widget* che costituisce quest'area è sempre lo stesso, ma quando deve visualizzare non risponde agli input dell'utente.

Per poter implementare l'area di disegno, Qt mette a disposizione il *Graphics View Framework*.

Esso fornisce una superficie, la cosiddetta *scena*, per gestire e interagire con un gran numero di *oggetti* grafici 2D, e un *widget* chiamato *vista*, per la visualizzazione degli oggetti e il supporto alle loro trasformazioni.

Inoltre include un'architettura per la propagazione degli eventi che permette un'elevata precisione nella capacità di interazione degli oggetti sulla scena. Essi possono intercettare eventi della tastiera, della pressione e del rilascio dei tasti del mouse, e ne possono tracciare i movimenti.

4.4.4.1 La scena

La classe fornita da Qt per rappresentare la scena è `QGraphicsScene`. Essa ha le seguenti responsabilità:

- fungere da contenitore per gli oggetti, rappresentati dalla classe `QGraphicsItem`;
- fornire un'interfaccia efficiente per la gestione di un gran numero di oggetti;
- propagare gli eventi a ciascun oggetto;

- gestire lo stato di un oggetto, come la selezione e il *focus*⁹.

Per impostare secondo le esigenze di *Vesta* il comportamento della scena in risposta agli eventi, `QGraphicsScene` è stata ereditata dalla classe `DrawingScene`.

Essa permette di disegnare linee e rettangoli, importare blocchi, selezionare, spostare, ridimensionare e cancellare le entità, il tutto ridefinendo i metodi che gestiscono gli eventi del mouse e della tastiera: `mousePressEvent`, `mouseMoveEvent`, `mouseReleaseEvent` e `keyPressEvent`.

I metodi `dragEnterEvent`, `dragMoveEvent` e `dropEvent` servono invece a gestire gli eventi del mouse quando si sposta un oggetto da un *widget* sulla scena (*drag and drop*), e verranno descritti più avanti.

4.4.4.2 La vista

La classe `QGraphicsView` implementa il *widget* che fornisce la vista per la visualizzazione del contenuto della scena; è un'area dotata di barre di scorrimento che permettono di navigare grandi scene. Anch'essa riceve gli eventi del mouse e della tastiera e li trasforma in eventi da inviare alla scena.

La vista ha associata una matrice di trasformazione che permette di allargare o restringere l'inquadratura, quindi scalare la vista, o ruotarla.

La classe che in *Vesta* deriva da `QGraphicsView` è `DrawingView`. Ogni sottofinestra relativa ad un documento aperto, possiede questo *widget* nell'area centrale.

`DrawingView` si occupa di implementare nelle varie sottofinestre i comandi invocati dalla finestra principale, in modo che il relativo stato dell'area di disegno venga conservato anche quando si passa da una sottofinestra all'altra.

Così ad esempio, se vi sono due disegni in corso, e in una sottofinestra è stato impartito il comando che toglie la visualizzazione della griglia, nell'altra sottofinestra essa rimane visibile.

⁹ Indica lo stato di un *widget* in grado di ricevere gli eventi della tastiera.

4.4.4.3 Gli oggetti

`QGraphicsItem` è la classe base per gli oggetti grafici¹⁰ della scena.

Il *Graphics View Framework* mette a disposizione diversi oggetti predefiniti per le tipiche figure geometriche quali i rettangoli (`QGraphicsRectItem`), gli ellissi (`QGraphicsEllipseItem`), le linee (`QGraphicsLineItem`), e per il testo (`QGraphicsTextItem`). Ma le caratteristiche più potenti sono sicuramente disponibili negli oggetti definiti dal programmatore.

Tra le altre cose, il `QGraphicsItem` supporta le seguenti funzionalità:

- la gestione degli eventi del mouse, dall'uso dei tasti alla rotella, dal movimento al semplice passaggio del puntatore sopra un oggetto;
- la gestione degli eventi della tastiera;
- il *drag and drop*;
- la possibilità di raggruppare degli oggetti, sia attraverso la relazione padre-figlio che utilizzando la classe specifica `QGraphicsItemGroup`;
- la gestione delle collisioni.

Per realizzare al meglio tutte queste caratteristiche, in *Vesta* sono state implementate tre sottoclassi di `QGraphicsItem`:

- `LineEntity`, che estende direttamente `QGraphicsLineItem` e rappresenta una linea;
- `RectEntity`, che estende direttamente `QGraphicsRectItem` e rappresenta un rettangolo;
- `Block`, che estende direttamente `QGraphicsItemGroup` e rappresenta un blocco;
- `ResizeHandle`, una classe accessoria che estende `QGraphicsRectItem`, che

¹⁰ In questo caso la parola oggetto non indica un *widget*, ma una figura geometrica o gruppi di queste, ma anche del testo.

rappresenta una maniglia per il ridimensionamento dei primi tre oggetti. L'oggetto `LineEntity` ha come figli due `ResizeHandle`, poste alla sua estremità; il `RectEntity` ne ha quattro e il `Block` solo uno.

Per impostare il corretto comportamento delle suddette entità, per ciascuna di esse sono stati ridefiniti i metodi `mousePressEvent`, `mouseMoveEvent`, `mouseReleaseEvent`, per rispondere agli eventi di pressione dei tasti del mouse; `hoverEnterEvent`, `hoverMoveEvent`, `hoverLeaveEvent`, per reagire all'evento del mouse che passa col suo puntatore sopra un'entità; il metodo `paint` per indicare precisamente come l'oggetto deve essere disegnato quando è selezionato e non.

4.4.5 La configurazione

Quando l'utente invoca il comando *Configure domotics system*, viene eseguito lo slot `configure` della classe `MainWindow`.

Da questo momento in poi, *Vesta* si deve limitare a gestire un solo progetto, cioè quello rappresenta l'abitazione che contiene il sistema domotico da configurare. Un messaggio (`QMessageBox`) avverte l'utente di salvare e chiudere tutti i progetti che non servono e poi viene invocato il metodo remoto `getDeviceList` per reperire da *domoNet* la lista dei dispositivi domotici.

Nella finestra principale viene creata una dock window (`QDockWidget`), che contiene una lista consumabile implementata dalla classe `DomoDevicesList`, un'estensione di `QListWidget`. Ogni oggetto di questa lista (`QListWidgetItem`) rappresenta un dispositivo domotico (`DomoDevice`) contenuto nella `domoDeviceHash`, descritto da un'icona, l'identificatore (`id@url`), il tipo di dispositivo (dal quale dipende l'icona predefinita assegnata) e la sua descrizione.

Quando la *dock window* è visibile all'utente, si può iniziare la procedura di configurazione, che passa attraverso tre fasi descritte nei paragrafi seguenti.

4.4.5.1 Il drag and drop

Con l'operazione di *drag and drop*, l'utente può disporre un dispositivo per volta sulla scena.

Il *widget* da cui parte il trascinamento è la `DomoDevicesList`, pertanto in questa classe bisogna ridefinire il metodo `startDrag`, in cui si salvano le informazioni da spostare dalla lista alla scena.

L'identificatore del dispositivo e l'icona, vengono incapsulati all'interno di un oggetto `QMimeData`, che associa queste informazioni al corrispondente tipo *MIME*¹¹ per assicurare che queste vengano trasferite correttamente attraverso i due *widget*, ma potenzialmente anche attraverso applicazioni diverse.

Il *widget* che accoglie il trascinamento è la scena (`DrawingScene`), quindi in questa classe va ridefinito il metodo `dropEvent` che si occupa di aprire il pacchetto MIME e utilizzare le informazioni in esso contenute per visualizzare il dispositivo sulla scena.

La scena condivide con la lista un puntatore alla `domoDeviceHash`, che viene acceduta attraverso l'identificatore del dispositivo per reperire il `DomoDevice` associato.

Per poter essere visualizzato sulla scena, la classe che implementa il dispositivo (`DomoDevice`) estende `QGraphicsPixmapItem`, una sottoclasse di `QGraphicsItem` che rappresenta delle immagini; in questa maniera l'icona del dispositivo viene inserita nella scena e viene trattata come tutti gli oggetti che essa può contenere.

4.4.5.2 La configurazione del dispositivo

Al termine del *drag and drop*, appena l'utente rilascia il tasto del mouse, l'icona del dispositivo viene visualizzata sulla scena nel punto in cui si trova il mouse, e si apre una finestra di dialogo che permette la configurazione del dispositivo. Tale finestra è implementata dalla classe `DomoDeviceDialog`, che contiene le caselle di

¹¹ *Multipurpose Internet Mail Extensions* è un protocollo di Internet che estende il formato delle email per supportare il trasferimento di dati diversi da caratteri ASCII.

testo (`QLineEdit`) necessarie a inserire gli attributi del dispositivo e vari pulsanti (`QPushButton`) per il salvataggio dei dati immessi, la configurazione dei servizi e la cancellazione della procedura (si veda figg. 3.13a e 3.13b).

Per ogni servizio offerto dal dispositivo, è possibile impostarne gli input e i valori ammessi utilizzando il dialogo `ServiceDialog` che si apre quando si preme il pulsante *Configure* relativo al servizio (figg 3.14a e 3.14b). Tutte le informazioni riguardo al dispositivo, vengono aggiornate nel relativo oggetto `DomoDevice` della `domoDeviceHash`.

4.4.5.3 La configurazione delle relazioni tra dispositivi

Quando l'utente termina la configurazione dell'ultimo dispositivo visualizzato nella lista, la *dock window* che la contiene viene chiusa e a questo punto si possono stabilire le relazioni che ci devono essere tra l'esecuzione di un servizio di un dispositivo e un altro servizio.

Per identificare i due dispositivi coinvolti, l'utente deve tracciare una linea che parte dal dispositivo da cui parte la relazione e arriva a quello da collegare. La linea è implementata nella scena utilizzando un `QGraphicsLineItem` e intercettando gli eventi `mousePressEvent`, `mouseMoveEvent` e `mouseReleaseEvent` della scena (`DrawingScene`). Appena l'utente rilascia il tasto del mouse sul dispositivo da collegare, si apre una finestra di dialogo implementata dalla classe `LinkedDevicesDialog` che visualizza i servizi dei due dispositivi (fig. 3.16). I due `DomoDevice` vengono reperiti grazie al metodo `itemAt` di `QGraphicsScene`, che restituisce l'oggetto grafico che contiene il punto passato come parametro al metodo.

Con le informazioni immesse dall'utente nelle caselle di testo, viene costruito un oggetto `LinkedService` e passato nel costruttore della classe `LinkedServicesDialog` che implementa la finestra di dialogo *Configure Linked Inputs* (fig. 3.17). Per ogni associazione tra input indicata in questo dialogo, viene aggiunto un oggetto `LinkedInput` alla lista `linkedInputList` dell'oggetto

`LinkedService`. Se la procedura di configurazione non viene abortita dall'utente, il salvataggio inserisce il `LinkedService` al `DomoDevice` da quale è partita la relazione.

Per uscire dalla modalità di configurazione, l'utente deve premere il tasto destro del mouse e dopo la conferma, si passa in modalità di controllo.

Gli aggiornamenti dei vari `DomoDevice` coinvolti nella configurazione, vengono salvati anche nella `domoDeviceHash`, che viene poi utilizzata per formattare la stringa da inviare a *domoNet* invocando il metodo remoto `devicesUpdate`.

4.4.6 L'interazione

4.4.6.1 L'invio di un comando ad un dispositivo

Quando *Vesta* è in modalità di controllo, l'utente può inviare un comando remoto ad un dispositivo. Premendo due volte il tasto sinistro del mouse sul dispositivo scelto, si apre una finestra di dialogo (fig. 3.18) implementata nella classe `CommandDialog`, nella quale va scelto il servizio di tale dispositivo che si vuole eseguire e l'input relativo. Premendo il pulsante *Send command*, con i valori inseriti dall'utente, viene formattata una stringa *domoMessage* di tipo `COMMAND` ed eseguito il metodo remoto `execute`.

Il valore di ritorno del metodo remoto viene visualizzato nella finestra e viene abilitato il pulsante di chiusura del dialogo.

4.4.6.2 La visualizzazione della cronologia degli eventi

Appena si entra in modalità di controllo, viene creato il *thread* (`Listener`) e il *socket* (`QTcpSocket`) per ricevere da *domoNet* gli aggiornamenti dello stato dei dispositivi e una *dock window* (`QDockWidget`) che visualizza la cronologia degli eventi del sistema domotico all'interno di un oggetto `QTextEdit`.

Quando arriva un aggiornamento nel *socket*, viene inviata una notifica del `DomoMessage` ricevuto, alla finestra principale; lo *slot* `showDomoMessage` della

classe `MainWindow` si occupa poi di visualizzare nella *dock window* il contenuto del messaggio (fig. 3.19).

4.4.6.3 La visualizzazione dello stato di un dispositivo

In modalità di controllo, quando si sofferma il puntatore del mouse su un'icona di un dispositivo domotico per 2 secondi, si apre una finestra che contiene le informazioni sul suo stato; si richiude automaticamente quando si allontana il puntatore. Questo meccanismo è implementato intercettando il passaggio del mouse su un `DomoDevice`, attraverso i metodi `hoverEnterEvent` e `hoverLeaveEvent`. Quando si entra nell'area di un `DomoDevice` viene creata una `QLabel deviceInfo`, contenente la descrizione del dispositivo e lo stato dei servizi con i relativi valori di input correnti (fig. 4.20).

4.5 Test

I test dell'applicazione sono stati effettuati utilizzando il sistema domotico installato presso il Laboratorio di Domotica dell'ISTI (Istituto di Scienza e Tecnologie dell'Informazione) del CNR di Pisa.

Il laboratorio è dotato di una vasta gamma di dispositivi domotici aventi tutti caratteristiche diverse ed appartenenti a standard domotici diversi, resi interoperabili attraverso *domoNet*, che espone un *web service* sull'host `http://ha-mediacycenter.isti.cnr.it:8080/axis/services/DomoNetWS`.

Nei test sono stati invocati i metodi remoti di questo *web service* per reperire la lista dei dispositivi, aggiornarla in seguito alla configurazione, inviare comandi remoti e registrare presso *domoNet* il *socket* per la ricezione degli aggiornamenti di stato.

Vesta riceve correttamente gli aggiornamenti di stato, sia in seguito all'invio di un comando remoto, sia in seguito all'uso diretto del dispositivo nel laboratorio.

5 Conclusioni

5.1 Realizzazioni sperimentali e valutazioni

L'applicazione realizzata permette disegnare attraverso dei semplici strumenti grafici la pianta dell'abitazione in cui è installato il sistema domotico che si vuole utilizzare, di disporre i dispositivi al suo interno, simboleggiati da icone, di configurarli e di interagire con essi inviando dei comandi remoti e ricevendo gli aggiornamenti di stato. Il tutto nel rispetto della più importante prerogativa richiesta a questa applicazione: la *semplicità*.

La sua interfaccia amichevole infatti, agevola l'utente finale nell'interazione con il sistema, e l'installatore più o meno esperto, nella fase di configurazione. *Vesta* risulta essere così il giusto compromesso tra strumenti per la configurazione complicatissimi utilizzabili solo da personale estremamente qualificato (come il software ETS), e semplici meccanismi hardware basati su spinotti particolari, comunemente usati dagli elettricisti.

Oltre alla semplicità, *Vesta* racchiude in sé altre caratteristiche importanti.

- È un'applicazione *open source*, rilasciata sotto licenza GNU e pubblicata nella pagina del progetto *domoNet* su <http://sourceforge.net/projects/domonet>; è stata realizzata con soli strumenti *open source* e quindi può essere distribuita liberamente senza alcun vincolo.
- È un'applicazione *multifunzionale*: le operazioni di configurazione e

interazione sono per la prima volta integrate in una sola applicazione.

- È *universale*: sfruttando le funzionalità di interoperabilità messe a disposizione da *domoNet*, riesce a configurare e interagire con qualsiasi dispositivo domotico, indipendentemente dalla tecnologia su cui si basa, evitando di dover adattare l'applicazione quando si effettua qualche modifica al sistema, sia in termini di tipo che numero di tecnologie utilizzate.

5.2 Sviluppi futuri

Le caratteristiche di *Vesta* che sono state realizzate, pur essendo tutte completamente funzionanti, implementano le funzionalità di base e rappresentano solo una piccola parte di tutto ciò che un'applicazione di questo tipo potrebbe offrire.

In versioni successive di *Vesta*, si potrebbero potenziare e aumentare gli strumenti per il disegno della pianta di un'abitazione, in modo che non debba essere stilizzata e ridotta a sole linee e rettangoli, ma che possa rappresentare anche circonferenze e archi; si potrebbero implementare funzionalità per ancorare il disegno alla griglia, o alle estremità di entità già disegnate; funzionalità per impostare il colore e lo spessore delle linee che compongono entità e blocchi; si potrebbe fare in modo che la modifica di un blocco venga aggiornata in tutte le sue repliche nel progetto.

Nella fase di configurazione, si potrebbe implementare una funzionalità che configura automaticamente la posizione di un dispositivo, quando viene collocato all'interno della pianta, che quindi deve indicare chiaramente il nome delle diverse stanze o aree dell'abitazione.

Molto utile risulterebbe l'impostazione di determinati scenari, in modo che una serie di operazioni ripetitive possa essere eseguita attraverso un unico comando, impartito direttamente dall'utente o programmato ad orari prestabiliti.

La modalità di interazione infine potrebbe in futuro supportare l'utilizzo del sistema domotico attraverso schermi tattili (*touch screen*), o i più sofisticati sistemi

di riconoscimento vocale, per svincolare l'utente dal calcolatore.

Per supportare invece il controllo remoto attraverso dispositivi mobili (palmari o cellulari), occorrerà implementare una versione di *Vesta* più leggera in modo che possa essere eseguita con ridotte risorse di calcolo e di memoria.

Bibliografia

- Miori, V. Tarrini, L. Manca, M. Tolomei, G. "An open standard solution for domotic interoperability". In Consumer Electronics, IEEE Transaction (Feb 2006), Volume: 52, Issue: 1, page(s): 97- 103.
- Jammes F., Mensch A., Smit H., "Service oriented device communications using the devices profile for web services", Proc. 3rd international workshop on Middleware for pervasive and ad-hoc computing, Grenoble, France, Nov. 2005
- Papazoglou, M.P. and Georgakopolous, D. "Service Oriented Computing". In Communications of the ACM 46, 10. (2003), 42-47.
- F. Furfari, C. Soria, V. Pirelli, O. Signore, R. Bandinelli, "NICHE: Natural Interaction in Home Environment", Ercim News no. 58, pp. 66-67, July 2004
- Miori V., Russo D., Aliberti M. An informatics research contribution to the domotic take-off. In: Ercim News, vol. 72 pp. 54 - 55. ERCIM EEIG, 2008
- Eckel B., *Thinking in C++, Second Edition, Volume one: Introduction to standard C++*, Prentice Hall, 2000
- Blanchette J., Summerfied M., *C++ GUI Programming with Qt 4, Second*

Edition, Prentice Hall, 2008

- *Qt Reference Documentation*, <http://doc.trolltech.com/4.4/index.html>
- *QCAD User Reference Manual*, http://www.ribbonsoft.com/qcad_doc.html
- *AutoCAD 2000 DXF Reference*,
- <http://www.autodesk.com/techpubs/autocad/acad2000/dxf/index.htm>
- *dxflib Programmes's Guide*, http://www.ribbonsoft.com/dxflib_doc.html
- *dxflib Class Reference*,
- http://www.ribbonsoft.com/qcadlib_dxflib_classref.html
- *Axis C++ Documentation*, <http://ws.apache.org/axis/cpp/documentation.html>