ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA DEI SISTEMI E DELLE TECNOLOGIE DELL'INFORMAZIONE

IoT e progettazione di Sistemi di Home Automation: un caso di studio reale basato su framework e standard open.

Tesi in
Sistemi Intelligenti Distribuiti LS

Relatore

Prof. Alessandro Ricci

Presentata da

Danilo Candiotti

Sessione III

Anno Accademico 2014-2015

Indice

Elenco delle figure

Elenco delle tabelle

Elenco degli algoritmi

1 Introduzione

La Home Automation è un ambito in cui la Internet of Things è in grado di fornire strumenti capaci di garantire nuove funzionalità che una visione ormai datata e economicamente svantaggiosa di questi sistemi ne sta limitando l'adozione presso un vasto pubblico di potenziali acquirenti. La chiusura di certi sistemi rispetto a nuovi protocolli sviluppati per favorire l'integrazione e a nuovi dispositivi recentemente comparsi sul mercato, ne relegano l'impiego esclusivamente agli ambiti per i quali sono stati pensati inizialmente e quindi difficilmente ampliabili in futuro.
In questo contesto hanno fatto comparsa, ormai da tempo, dispositivi dotati di discrete capacità di calcolo e di connettività in grado di fornire nuovi servizi e nuove possibilità di utilizzo che erano impensabili precedentemente. Nonostante queste nuove caratteristiche, le soluzioni proposte dalle aziende rappresentano applicazioni verticali che, sebbene utilizzino alcune delle tecnologie della IoT per aumentarne le funzionalità, risultano isolate le une rispetto alle altre limitandone nuovamente i possibili utilizzi solamente ai casi previsti in fase progettuale.
A questo proposito diverse aziende hanno realizzato propri hub in grado di integrare sotto un unico ecosistema svariati oggetti, dotati di connettività e di un sistema di controllo proprietario, consentendo di ampliarne le funzionalità e la possibilità di creare nuovi scenari non previsti inizialmente dal singolo costruttore.
Anche grossi nomi come Apple e Google hanno recentemente fatto il loro ingresso in questo mercato presentando architetture realizzate su una loro idea di IoT e spinti sicuramente anche dalle previsioni economiche molto favorevoli.

Il loro approccio, sebbene fornendo tutti gli strumenti di sviluppo necessari, è incentrato nel fornire sia un protocollo che servizi basati sulla presenza di un ecosistema cloud a cui gli altri produttori, interessati ad adottare la loro tecnologia, saranno costretti ad adeguarsi.
In questo contesto e contrariamente a questo modo di pensare, tecnologie open-source come Eclipse SmartHome, danno la possibilità di superare lo scoglio iniziale nella scelta di quale tecnologia adottare, quale eventuale sistema di automazione installare in una abitazione e, capacità di programmazione permettendo, anche di implementare la propria integrazione come è stato fatto nel caso di studio presentato in questo testo.
Il caso di studio, pensato e realizzato con lo scopo di poterlo realmente utilizzare in ambiente domestico, cerca di raccogliere i problemi e le soluzioni adottate per giungere a quella ambita integrazione, tra sistemi e dispositivi, che la IoT promuove tentando di fornire a un sistema di Home Automation tradizionale l'apertura di cui purtroppo è carente.
Lo scopo di questo progetto è stato quello di fornire uno strumento che potesse da un lato avvicinare alla Home Automation quelle persone che ancora sono piuttosto restie ad adottare una tecnologia per loro ritenuta ancora complessa e costosa e dall'altro far comprendere ai fornitori di sistemi domotici che, adottare le nuove tecnologie della IoT e l'apertura che ne può derivare, può portare un reale giovamento a un mercato che ancora manca di una ampia adozione da parte del pubblico.

Dopo questa breve introduzione vediamo come è stato suddiviso il testo per fornire un quadro generale che possa aiutare la lettura e la ricerca di un eventuale argomento.
La prima sezione riguarda Internet of Things dove verranno date le definizioni del termine e si prenderanno in esame gli standard e i protocolli che risultano di notevole importanza in questo contesto. Verranno inoltre forniti alcuni dati

interessanti per cercare di dare una idea delle dimensioni di questo fenomeno e come venga percepito dall'industria.
Nella seconda sezione, riguardante il settore della Home Automation, ne presenta il percorso nel tempo e i problemi che ne limitano l'adozione da parte dei proprietari di appartamenti. Vengono illustrati alcuni degli standard che vengono utilizzati in questo ambito e i principali sistemi che possono essere acquistati in Italia esponendone le principali caratteristiche. Per concludere vengono presentate alcune soluzioni che si sono affacciate recentemente su questo mercato che sfruttano le tecnologie proprie della IoT.
L'ultima sezione, la più corposa di tutte, è riservata alla progettazione di un sistema di Home Automation che riuscisse a sfruttare le tecnologie della IoT per ampliare le funzionalità di un reale sistema domotico in un reale contesto abitativo. Viene presentato in parte il percorso seguito per raggiungere gli obiettivi, superando alcune difficoltà incontrate, fino ad ottenere un sistema completamente funzionante e realmente utilizzato che presenti buone caratteristiche di apertura, flessibilità e performance. A conclusione vengono esposte alcune valutazione su quanto realizzato.

2 Internet of Things

Probabilmente la prima volta che comparve il termine Internet of Things fu in una presentazione fatta da Kevin Ashton[1] nel 1999. Kevin Ashton è stato il cofondatore e direttore esecutivo dell'Auto-ID Center, un gruppo del Massachusetts Institute of Technology (MIT), che fu fondato nell'intento di sviluppare un sistema, basato su tecnologia RFID, per sostituire l'ormai datato UPC Bar Code[2].

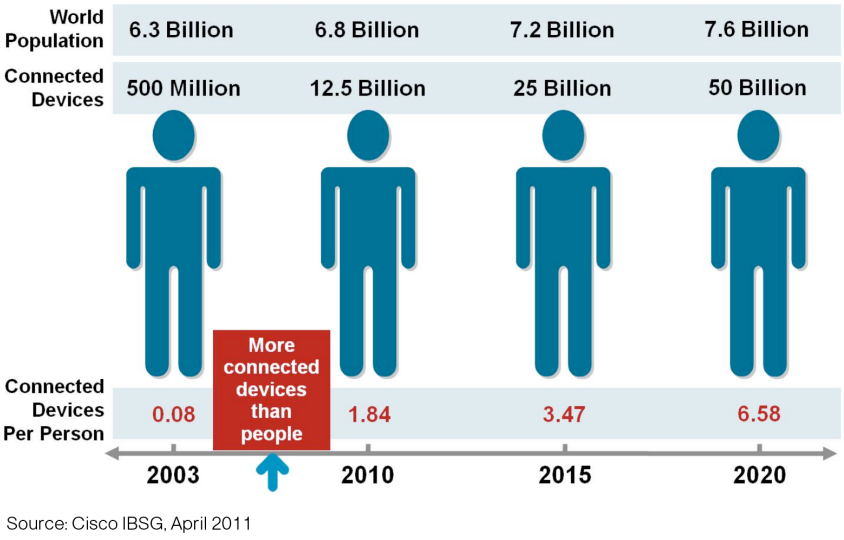


Figura 2.1: Nascita dell'internet delle cose

Secondo una stima di Cisco IBSG (Internet Business Solutions Group) la Internet of Things sarebbe "nata" tra il 2008 e il 2009 quando il numero di dispositivi connessi ad internet ha superato il numero della popolazione mondiale[11]. Nel

[1] http://www.rfidjournal.com/articles/view?4986
[2] https://en.wikipedia.org/wiki/Barcode

2010 il boom di smartphone e tablet ha portato il numero di dispositivi connessi a internet a 12,5 miliardi mentre la popolazione mondiale è salita a 6,8 miliardi quindi, il numero di dispositivi connessi per persona, ha raggiunto il valore 1,84[3]. Come si può vedere dalla figura 2.1 la stima di Cisco IBSG proseguiva ipotizzando che i dispositivi connessi ad internet sarebbero diventati 25 miliardi nel 2015 e avrebbero raggiunto i 50 miliardi entro il 2020.
Gartner, multinazionale leader nella consulenza strategica, ricerca e analisi nel campo dell'Information Technology, stila annualmente documenti e rapporti e, tra i numerosi dati, compare anche una rappresentazione grafica sviluppata e utilizzata per indicare la maturità, adozione e applicazione di specifiche tecnologie.

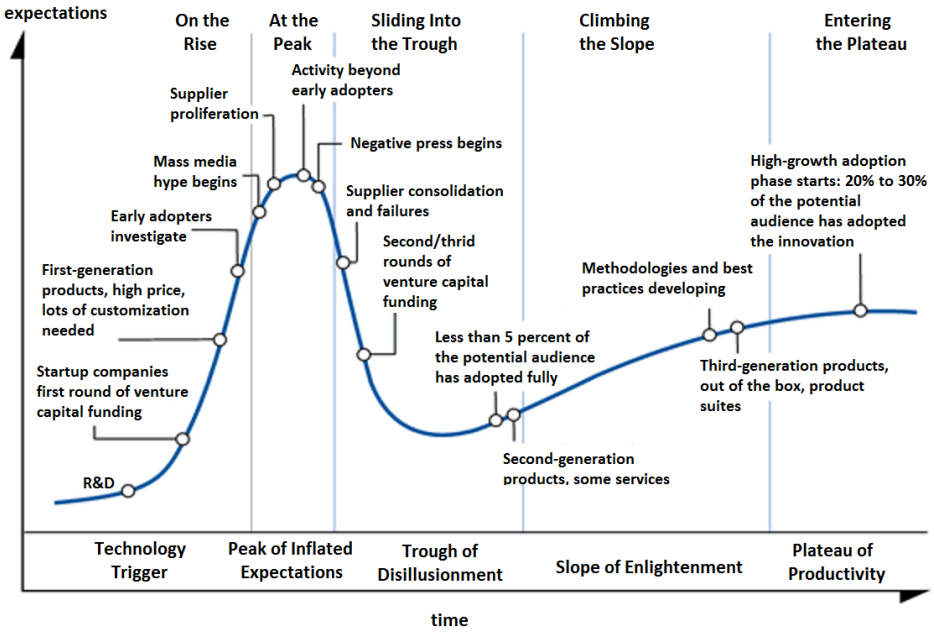


Figura 2.2: Gartner Hype Cycle

L'Hype Cycle (vedi figura 2.2) è suddiviso in cinque fasi chiave il cui insieme descrive il ciclo di vita a cui è destinata una tecnologia:

[3]Fonti: Cisco IBSG, 2010; U.S. Census Bureau, 2010.

Technology Trigger. Una tecnologia potenzialmente innovativa ha inizio. Primi tentativi di progetto e l'interesse dei media genera significativa divulgazione. Spesso non esistono prodotti utilizzabili e la fattibilità per la commercializzazione non è certificata.

Peak of Inflated Expectations. L'iniziale divulgazione produce un certo numero di casi di successo - spesso accompagnati da molteplici fallimenti. Alcune aziende vi prendono parte; molte evitano.

Trough of Disillusionment. L'interesse diminuisce così come gli esperimenti e le implementazioni non vengono realizzate. Fornitori della tecnologia si ridimensionano o falliscono. Gli investimenti continuano solo se i fornitori sopravvissuti migliorano i loro prodotti per la soddisfazione dei primi acquirenti.

Slope of Enlightenment. Maggiori esempi di come le aziende possono beneficiare della tecnologia cominciano a cristallizzarsi e iniziano ad essere capite maggiormente. Seconda e terza generazione di prodotti appaiono dai fornitori di tecnologia. Più aziende finanziano esperimenti; ditte più conservative rimangono caute.

Plateau of Productivity. L'adozione su larga scala inizia a decollare. Per i fornitori sono definiti più chiaramente i criteri per valutarne la fattibilità. Un ampio mercato per l'applicabilità della tecnologia sta chiaramente dando risultati.

Interpretando la curva di Gartner possiamo notare che quando "nasce" una nuova tecnologia molte aziende e start-up hanno forti aspettative e fanno investimenti nel settore. Quando si arriva al top delle aspettative però queste aziende non iniziano subito a raccoglierne i frutti ma cadono in una fase in cui molte start-up e progetti falliscono perchè non rievono ordini. Nel tempo però, le aziende rimaste, attraverso una ulteriore fase, raggiungono il plateau di produttività dove finalmente cominciano a fatturare e dove si crea il reale mercato.

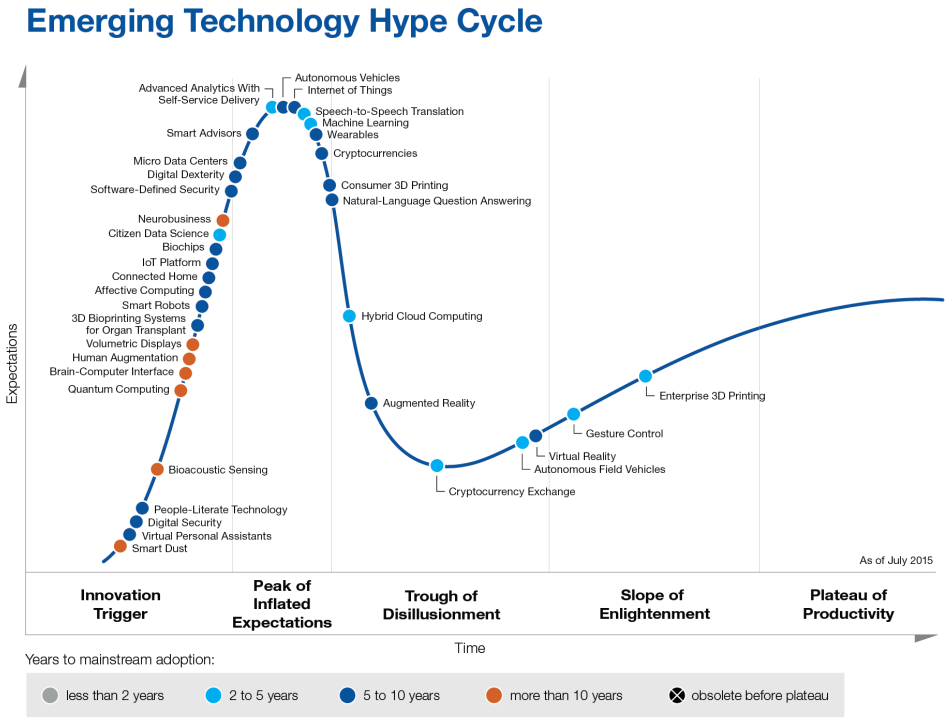


Figura 2.3: Gartner Hype Cycle of technology 2015

Dalla figura 2.3 si nota come l'internet delle cose attualmente sia nella fase in cui sta creando maggiori aspettative e nella legenda si può ricavare la stima del tempo che impiegherà questa tecnologia a raggiungere la fase in cui diventerà produttiva.
Nel presente capitolo si cercherà di dare una definizione di Internet of Things e di illustrare le principali tecnologie e standard adottati in questo contesto.

2.1 Definizione

Inizialmente, in letteratura, potevano essere ritrovate molteplici definizioni del termine Internet of Things e questo non era altro che il risultato del forte interesse che la comunità di ricerca rivestiva in questo nuovo concetto e dalla vivacità dei dibattiti su di esso [2]. La confusione era prodotta dal termine stesso e dalle parole da cui è composto.

Differenti visioni (vedi figura 2.4) dipesero da come enti di ricerca, standardizzazione e alleanze commerciali approcciarono il problema: solitamente in base ai loro interessi, finalità e background.

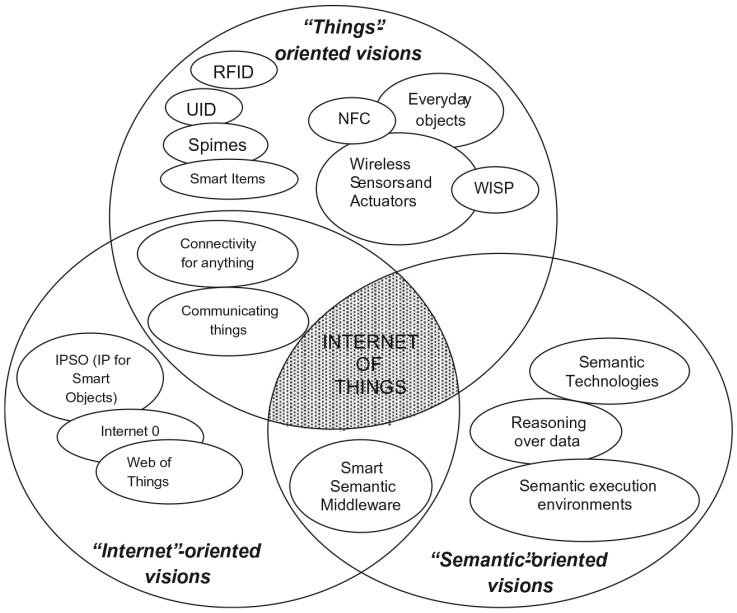


Figura 2.4: IoT come unione di più visioni

La prima definizione di IoT, come abbiamo visto all'inizio del capitolo, deriva da una prospettiva "Thing oriented" dovuta allo sviluppo di semplici oggetti: i tag Radio-Frequency IDentification (RFID). Il termine IoT però implica una visione più ampia rispetto alla semplice identificazione degli oggetti come, ad esempio, gli Smart Item, dotati di memoria, capacità di elaborazione e comunicazione wireless. L'abilità di connettere questi dispositivi, permetterne la comunicazione tra loro e internet, garantendo la possibilità di nuovi servizi per le persone, rappresentava la congiunzione tra la visione "Thing oriented" e la visione "Internet oriented" [2], che promuoveva Internet Protocol (IP) come tecnologia per connettere tutti gli Smart Object del mondo.
A lato, non meno importante delle prime due, c'era la visione "Semantic oriented" maturata dalla consapevolezza che in questa futura internet il problema

sarebbe stato l'enorme mole di informazione che avrebbe creato un numero così elevato di oggetti abilitati a comunicare.
Per comprendere meglio la portata di questa innovazione cerchiamo di elencare i principali elementi che sono in grado di generare le funzionalità della IoT [1]:

Identification: bisogna distinguere tra identificazione e indirizzamento. Sebbene ci siano diversi modi per identificare un oggetto, come ad esempio tramite Electronic Product Code (EPC), i metodi di identificazione possono non essere univoci globalmente. Quindi si rende necessario utilizzare anche un ulteriore metodo di indirizzamento per identificare con certezza un oggetto (ad esempio tramite IPv6).

Sensing: si intende ottenere informazione dagli oggetti di una rete per una successiva elaborazione. In questo caso si tratta sia di sensori che di attuatori ma anche di Single Board Computer (SBC) tipicamente utilizzati per realizzare prodotti della IoT.

Communication: l'insieme delle tecnologie per collegare tra loro gli oggetti in modo da fornire i servizi a loro richiesti.

Computation: comprende sia le piattaforme hardware che quelle software. Tra le piattaforme software vengono considerati anche i sistemi operativi inclusi nei vari device. Per finire si aggiungono le piattaforme cloud e i framework per realizzare servizi IoT.

Service: Al-Fuqaha [1] suddivide i servizi forniti dalla IoT in quattro classi.

Identity-Related si tratta di servizi la cui importanza è tale da risultare come base per tutti gli altri tipi di servizi. Qualsiasi applicazione ha bisogno di identificare gli oggetti con cui deve lavorare.

Information Aggregation il cui scopo è quello di collezionare i dati ricevuti dalle misure effettuate dai sensori.

Collaborative-Aware sfruttando il servizio Information Aggregation e i dati da esso forniti sono in grado di fornire supporto alle decisioni e compiere azioni opportune. Ricadono, ad esempio, in questa categoria le applicazioni per Smart Home & Building.

Ubiquitous consentono di fornire i servizi della classe Collaboratie-Aware a chiunque, in qualunque momento e dovunque. Un esempio di questo tipo di servizi è ciò che viene chiamato Smart City.

Semantic: modellare, riconoscere e analizzare le informazioni per fornire supporto alle decisioni. Rappresenta la mente della IoT.

Vediamo ora i principali standard e le tecnologie sviluppate a supporto della Internet of Things.

2.2 Principali standard

Il termine Internet of Things (IoT) si riferisce a una rete globale di reti che comprendono miliardi di dispositivi differenti chiamati "smart object" oppure "thing". Si tratta di dispositivi con risorse limitate, limitata capacità di calcolo e memoria, tipicamente alimentati a batteria, con interfaccia radio e forniti di sensori e attuatori. Le limitazioni non sono semplicemente hardware ma, per mantenere efficienza energetica i protocolli di comunicazione e i sistemi operativi devono essere progettati attentamente e implementati in modo da minimizzare il consumo di energia. Generalmente questi Smart Object operano in reti wireless a bassa potenza e con molte perdite di conseguenza fanno uso del protocollo IEEE 802.15.4 nato e progettato proprio per questo scenario.
Questa enorme quantità di dispositivi richiede la possibilità di integrarsi e interagire con la rete internet tradizionale e questo richiede politiche di indirizzamento efficaci che il protocollo IPv4, con i suoi indirizzi da 4 byte, non è in grado di fornire. Mediante l'introduzione di IPv6, grazie ai suoi 128 bit, si è in grado di definire fino a 10^{38} indirizzi il che dovrebbe essere sufficiente a identificare qualsiasi oggetto al quale fossimo interessati ad assegnare un indirizzo.

RFID e NFC.

Agli inizi del 2000 l'RFID era considerata la tecnologia più promettente a fornire un'accelerazione alla formazione della IoT. Fu realizzato anche un nuovo standard chiamato UHF RFID in grado di automatizzare la lettura dei tag a una distanza di 8-10 metri in modo da sostituire l'utilizzo dei Barcode ma, in pratica, problemi nella rilevazione dovuti a un errato posizionamento del tag/-prodotto limitarono l'adozione di questo nuovo standard da parte di grossisti del calibro di Walmart e Tesco.
La Near-Field Communication (NFC) è una forma di RFID che ha fornito a questo sistema un'ulteriore possibilità soprattutto legata al supporto dei pagamenti elettronici. Sebbene non tutti gli smartphone comprendono questa tecnologia grandi aziende come Apple (e la diffusione dei loro sistemi di pagamento come ApplePay) potrebbe fornire una ulteriore spinta all'adozione di questa tecnologia anche come abilitante per la IoT.

Optical tag e Quick Response Codes.

Il successo del Quick Response Code (QRCode) è legato direttamnete alla diffusione dell'applicazoine che fa utilizzo della fotocamera ad alta risoluzione che è possibile trovare oramai in tutti gli smartphone più recenti.
Sebbene si possano trovare esempi di QRcode in giornali, riviste, depliant, ecc. la sua adozione è rallentata da una tiepida risposta da parte degli utenti. Il motivo si può ricercare proprio nella necessità di pre-installare l'applicazione richiesta per leggere i QRcode e nella difficoltà di posizionamento della fotocamera per la messa a fuoco e la decodifica accurata dell'immagine.

6LoWPAN

Si tratta di un protocollo nato per permettere ai pacchetti IPv6 di viaggiare su reti wireless a bassa potenza (in particolare IEEE 802.15.4). La nascita è dovuta all'idea di voler applicare il protocollo internet anche ai più piccoli dei

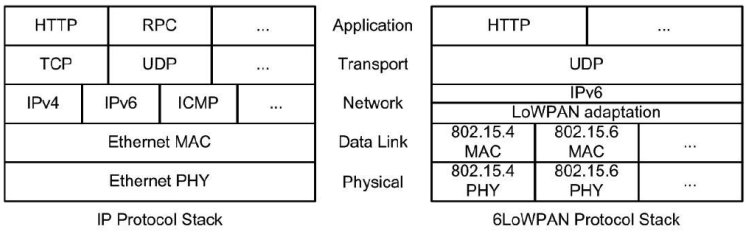


Figura 2.5: Stack IP e 6LoWPAN

dispositivi ma in grado di gestire i requisiti per IPv6, come il maggior spazio di indirizzamento e l'MTU di 1280 byte.
Il protocollo *IPv6 over Low power Wireless Personal Area Networks* (6LoWPAN) prevede l'incapsulamento e meccanismi di compressione degli header che permettono di in inviare e ricevere pacchetti IPv6 tra dispositivi con risorse limitate.
Dalla figura 2.5 si può notare la presenza di un livello di adattamento tra il livello data link e quello IP. Il livello di adattamento è il componente che consente di inserire pacchetti IPv6 all'interno del payload dei frame IEEE 802.15.4. Ad esempio l'header IPv6 è formato da 40 byte e senza compressione sarebbe impossibile trasmettere un qualsiasi payload con un livello data link come quello di IEEE 802.15.4 che ha una dimensione massima del pacchetto di solo 128 byte. Un altro caso è il Maximum Transmission Unit (MTU) che nel caso di IPv6 può raggiungere i 1280 byte mentre IEEE 802.15.4 supporta al massimo 128 byte [32].
Per connettere una Personal Area Network (PAN) a internet è possibile utilizzare *edge router* il cui ruolo è quello di instradare i pacchetti IP da e per l'esterno oltre che gestire i dispositivi interni tramite *discovery* e distribuzione del prefisso IPv6.

Ethernet o IEEE 802.3

Ethernet (anche noto come IEEE 802.3) è lo standard risultato vincitore per la realizzazione di LAN domestiche e aziendali, per cui può sembrare appropriato

un suo utilizzo anche per collegare periferiche in ambito domotico, in quanto lo scopo di Ethernet è quello di collegare tra loro in modo semplice ed efficiente dispositivi e risorse di rete. Il grande vantaggio di questo protocollo è la sua enorme diffusione, che ne ha permesso l'evoluzione negli anni. Il mezzo di comunicazione è il cavo, o coassiale (in disuso), o doppino di categoria 5 e 6 (il più utilizzato) o la fibra ottica e le possibili velocità di trasmissione variano di un ordine di grandezza da 1 Mbps a 1 Gbps. Il sistema di indirizzamento è statico, in quanto a ogni periferica viene assegnato un indirizzo (MAC address) a tempo di fabbricazione e la comunicazione avviene in broadcast. Il protocollo viene comunemente integrato nello stack TCP/IP. Per le sue caratteristiche, Ethernet può essere utilizzato per la realizzazione di alcune funzioni domotiche all'interno di un'abitazione, ma la sua diffusione massiccia è fortemente frenata dal fatto di essere legata strettamente alla cablatura, che ne scoraggia l'uso.

Wi-Fi o IEEE 802.11

La scelta più ovvia come tecnologia di networking per un dispositivo per l'IoT è il Wi-Fi perchè è molto diffuso. Purtroppo il Wi-Fi richiede una discreta quantità di potenza che non tutti i dispositivi possono permettersi, ad esempio, sensori posizionati in luoghi difficili da raggiungere dall'alimentazione di rete.

IEEE 802.15.4

Rilasciato nel 2003 lo standard radio IEEE 802.15.4 è una delle maggiori tecnologie abilitanti per l'IoT ed è stato concepito per regolamentare il livello livello fisico (PHY) per *Low-Rate Wireless Private Area Network* (LR-WPAN) ed il livello *Medium Access Control* (MAC) di reti in area personale (massimo 30 metri) e che lavorano con basse velocità di trasferimento dati.
Grazie alle sue caratteristiche come basso consumo di potenza, bassi data-rate, prezzo basso e alto throughput è utilizzato da IoT, M2M e *Wireless Sensor Network* (WSN). È in grado di fornire una comunicazione affidabile e può gestire un elevato numero di nodi (circa 65k). Garantisce inoltre un alto livello di

sicurezza, cifratura e servizi di autenticazione sebbene non fornisca garanzie di QoS [1].
A seconda del canale di frequenza usato il livello fisico trasmette/riceve i dati con tre data-rate diversi: 250 kbps @ 2.4 GHz, 40 kbps @ 915 MHz e 20 kbps a 868 MHz. Per ridurre le collisioni IEEE 802.15.4 MAC utilizza il protocollo CSMA/CA.

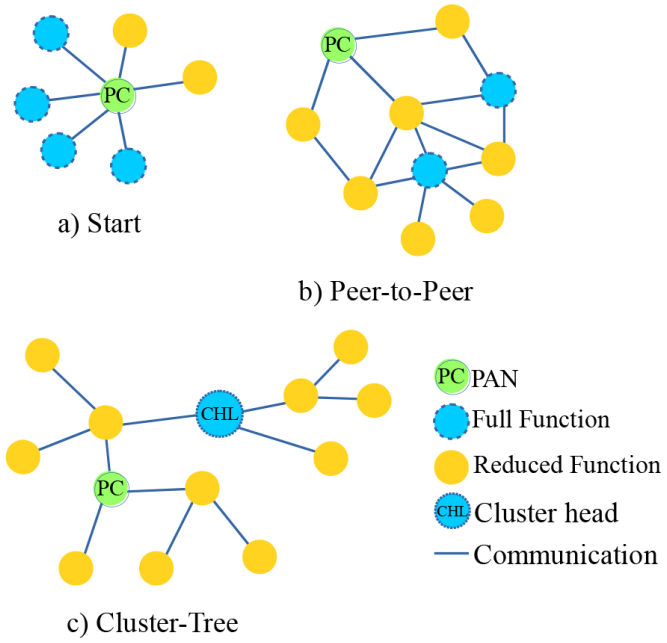


Figura 2.6: Topologie per 802.15.4

Supporta due tipi di nodi di rete: *Full* e *Reduced Function Device* (FFD e RFD). Un FFD può servire come coordinatore di una Personal Area Network (PAN) o anche come nodo normale. Un coordinatore è responsabile della creazione, controllo e gestione della rete. Gli RFD invece sono nodi molto semplici con risorse limitate.
Si ottiene una rete quando due o più device comunicano sullo stesso canale fisico. Per costruire una rete è necessario avere almeno un dispositivo FFD che agisca come PAN Coordinator. Il PAN Coordinator oltre a iniziare e terminare

la comunicazione (come un RFD) può anche instradare la comunicazione sulla rete. Generalmente il coordinatore è alimentato da rete e non a batterie.

ZigBee

Il protocollo ZigBee, basato sullo standard 802.15.4 e sviluppato dalla ZigBee Alliance, è stato definito nel 2003 allo scopo di supportare reti di oggetti a costi e consumi energetici molto minori rispetto ad altri più noti protocolli wireless; per questo è appropriato ad essere utilizzato per reti di sensori, ma le sue caratteristiche possono essere sfruttate da qualsiasi dispositivo. Il protocollo è wireless e lavora tra le bande di 2,4 GHz, 902-928 MHz e 868,3 MHz, mentre l'accesso al mezzo è gestito tramite CSMA/CA. I componenti di una rete ZigBee sono:

- ZigBee Coordinator: l'apparato con maggiore memoria e capacità di calcolo, in quanto deve avere una conoscenza completa della rete e deve gestirla. Ogni rete ne deve avere uno;
- ZigBee Routers: dispositivi che implementano tutte le funzionalità del protocollo 802.11.4 e che sono tipicamente usati per gestire un gruppo di ZigBee End Devices e instradare i messaggi allo ZigBee Coordinator. Reti semplici, ad esempio con tipologia a stella con il Coordinator in posizione centrale, possono non essere dotate di ZigBee Routers;
- ZigBee End Devices: dispositivi a ridotte funzionalità al fine di ridurne la complessità ed il consumo energetico; tipicamente sono i sensori alla periferia della rete.

La rete ZigBee è autoconfigurante e offre interessanti caratteristiche di sicurezza. Il throughput di 224 Kbps è sufficiente per le normali funzioni di automazione domestica escluse quelle per la gestione dei flussi audio/video. Essendo una rete conforme agli standard IEEE 802, ZigBee può essere collegata tramite bridge ad altre reti IEEE 802 come Ethernet. E' quindi possibile controllare una rete ZigBee da un personal computer, controllato a sua volta tramite Internet.

Z-Wave

Realizzato dalla Zensys e dalla Z-Wave alliance, Z-Wave è un protocollo per realizzare reti wireless di oggetti di semplici a basso consumo energetico. Molto simile a ZigBee per scopi e vincoli progettuali. Le sue differenze da ZigBee possono essere così riassunte:

- non lavora alla frequenza di 2,4 GHz;
- le reti Z-Wave sono tipicamente reti mesh senza un'entità centrale di controllo, la comunicazione tra dispositivi tra loro fuori portata avviene tramite la ripetizione del segnale dei nodi intermedi;
- ZigBee è stato realizzato dall'unione di grandi industrie (Philips, Motorola, Mitsubishi, Honeywell, ecc.) e ha ottenuto la standardizzazione dall'IEEE, Z-Wave è stato realizzato dalla Z-Wave alliance (Zensys, Leviton, Danfoss, ecc.).

Bluetooth

Lo standard Bluetooth è stato realizzato nel 1998 dai maggiori produttori di apparecchiature telefoniche. Inizialmente è stato pensato per realizzare PAN (Personal Area Network) e fondamentalmente per collegare senza bisogno di fili l'auricolare al telefono cellulare (lo standard prevede meccanismi particolari per la gestione di audio); successivamente lo standard è stato sfruttato per svariati tipi di uso e non si esclude un suo possibile utilizzo in domotica, nonostante le sue caratteristiche non si addicano pienamente a tale scopo. Lo standard è wireless ed è basato su *Frequency Hopping Spread Spectrum* (FHSS) con frequenza centrale attorno ai 2,4 GHz (e quindi sulle frequenze libere, le stesse usate per le WLAN 802.11b). I vantaggi nell'uso di Bluetooth nelle applicazioni domotiche derivano solamente dal fatto che è uno standard wireless (e quindi non necessita di interventi manuali sulle abitazioni esistenti) e che è uno standard conosciuto e abbastanza diffuso. Gli svantaggi tuttavia sono molto maggiori; infatti Bluetooth è stato concepito per reti a corto raggio, una decina di metri circa, (caratteristica migliorata con l'introduzione di Bluetooth 2) e per lo

scambio di grosse quantità di dati con frequenza molto sporadica (situazione perfetta per la comunicazione tra auricolare e telefono cellulare). Le applicazioni domotiche hanno difficoltà a rimanere nei raggi d'azione di Bluetooth (anche per la presenza di ostacoli come muri o mobili tra i dispositivi) e tipicamente necessitano di scambiarsi piccole quantità di dati o semplici comandi in maniera più frequente. Ogni singola rete Bluetooth, detta piconet, è dinamica e gestisce autonomamente l'ingresso e l'abbandono delle periferiche, ma ha una struttura master-slave che impone la presenza di un'entità centrale, il master, con il compito di gestire le altre periferiche. In una piconet non possono essere presenti più di sette slave attivi per ogni istante di tempo (esistono meccanismi per il cambio di stato degli slave da ready a parked). Se si vuole quindi creare una rete con più di otto dispositivi attivi contemporaneamente è necessario unire più piconet, creando una scatternet, ad esempio creando una gerarchia di master o facendo condividere a due o più masters uno stesso slave.

Bluetooth Low Energy.

Rispetto alla precedente versione sfrutta una comunicazione radio a corto raggio (massimo 100 metri) con una quantità minima di potenza in modo da poter lavorare per tempi più lunghi. BLE può lavorare con potenze di trasmissione tra i 0.01 mW e i 10 mW e risulta essere un buon candidato per le applicazioni IoT [1].
Confrontato a ZigBee è più efficiente in termini di energia consumata e nel rapporto tra energia trasmessa e bit. Permette ai dispositivi di lavorare come master o come slave in una topologia a stella. Dotato di meccanismo di discovery gli slave inviano annunci su uno o più canali dedicati a questo meccanismo che vengono scannerizzati dal dispositivo master. A parte quando avviene lo scambio dei dati rimangono in modalità sleep per il resto del tempo.

2.3 Protocolli a livello applicativo

Ci sono due paradigmi che possono essere utilizzati per la comunicazione tra applicazioni: request/response (polling) e publish/subscribe (push-based). Nel primo caso il client richiede esplicitamente alcune informazioni a un server che risponde con le informazioni richieste. Nel secondo un nodo pubblica del contenuto che è successivamente consegnato a uno o più subscriber attraverso un broker oppure direttamente. HTTP, ad esempio, implementa il modello a polling.

CoAP

Come progetto del gruppo IETF *Constrained RESTful Environments* (CoRE) il protocollo *Constrained Application Protocol* (CoAP) è stato progettato per portare il paradigma *REpresentational State Transfer* (REST), originariamente concepito per applicazioni basate su HTTP, verso applicazioni IoT notoriamente affette da molti vincoli di risorse.

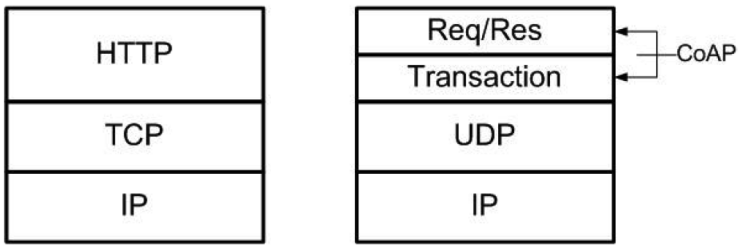


Figura 2.7: Stack HTTP e CoAP

CoAP è un protocollo binario che sfrutta il trasporto UDP, basato su un modello di comunicazione del tipo request/response simile al modello client/server di HTTP ed è in grado di effettuare comunicazioni multicast. Il protocollo HTTP, basato su trasporto TCP, ha problemi con ambienti dalle risorse limitate in particolare con comunicazioni wireless a bassa potenza mentre il nuovo protocollo realizzato da IETF può operare anche su reti IP tradizionali; ma il meglio di se lo da su reti vincolate da risorse limitate come reti wireless di sensori dove ci sono vincoli di potenza, memoria e di calcolo.

Per gestire il packet loss e successive ritrasmissioni, le applicazioni che lavorano in questi ambienti devono inviare la quantità di dati più piccola possibile. I protocolli per il livello di applicazione devono essere progettati in modo da tenere in considerazione il requisito di basso overhead richiesto dai livelli inferiori. CoAP è un protocollo binario molto leggero che può essere mappato facilmente in HTTP e quindi può integrarsi con il Web ma fornisce anche capacità ulteriori quali multicast, resource observing e service discovery [10].
CoAP implementa un modello request/response sopra UDP e l'utilizzo di UDP al posto di TCP ha le sue ragioni: è un protocollo di trasporto con un header di soli 8 byte, non richiede di impostare una connessione tra gli endpoint di comunicazione ed è compatibile con dispositivi che richiedono di "cadere" in sleep mode per lunghi periodi di tempo (e che quindi non potrebbero mantenere attiva una connessione).

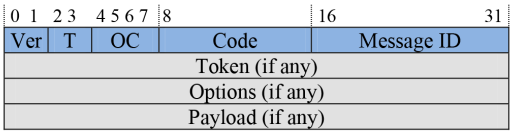


Figura 2.8: Formato dei messaggi CoAP

CoAP utilizza un formato molto semplice e contenuto per codificare i messaggi. Una prima parte fissa di 4 byte rappresenta l'header. Ci può essere anche un token la cui lunghezza può variare da zero a 8 byte e che aiuta a correlare le richieste e le risposte. Seguono campi opzionali per option e payload. Un tipico messaggio CoAP può variare tra 10 e i 20 byte e i campi presenti nella figura 2.8 hanno i seguenti significati: Ver è la versione di CoAP, T è il tipo di transazione, OC sta per Option count e Code rappresenta il metodo della richiesta (1-10) o il codice della risposta (40-255). Ad esempio 1, 2, 3 e 4 stanno rispettivamente per GET, POST, PUT e DELETE. Message ID è un identificativo per fare match con la risposta [1].
La natura di UDP però introduce alcune limitazioni (invece automaticamente gestite da TCP) che devono essere risolte al livello di applicazione come ad esempio le ritrasmissioni e il request/response matching. CoAP risolve il proble-

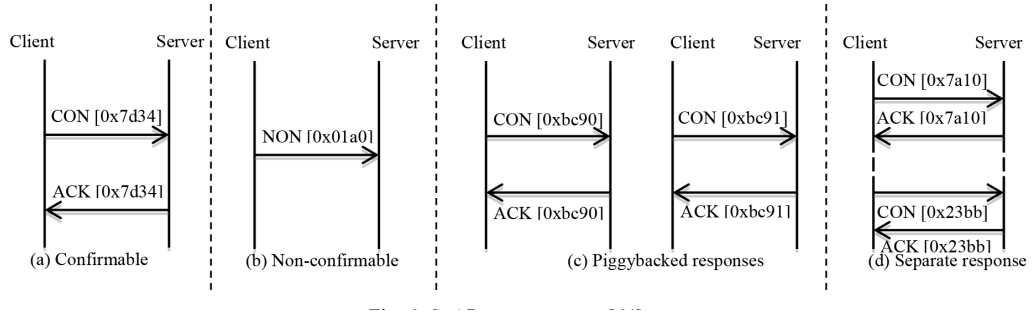


Figura 2.9: Tipi di messaggi CoAP

ma dell'affidabilità mediante l'utilizzo di messaggi Confirmable (CON). Quando una richiesta CON viene inviata il client si aspetta di ricevere un messaggio di Acknowledgement (ACK) dal server per confermare l'avvenuta consegna della richiesta. Se questo non avviene il client ritrasmette la richiesta al server. Allo stesso modo il server quando deve rispondere a una richiesta CON lo fa trasmettendo la risposta in un ulteriore messaggio CON attendendo in questo modo un messaggio di ACK a confermare la consegna con successo della risposta. Sia client che server utilizzano un timeout di ritrasmissione con exponential back-off.

CoAP è stato progettato per fornire una integrazione trasparente con il Web. Un "cross-proxy" è un elemento della rete che fornisce funzionalità di protocol translation per permettere l'integrazione tra endpoint HTTP e CoAP. I Cross-proxy garantiscono la massima interoperabilità tra internet e internet of thing rappresentando il punto di contatto tra di loro. Altri vantaggi sono la possibilità di nascondere dettagli della rete interna (fornendo anche meccanismi di caching), migliorare la sicurezza e ridurre il consumo di energia[10, 1].

Tra le opzioni c'è la modalità Observe con la quale viene introdotto un modello di comunicazione nuovo che differisce dal tradizionale request/response (polling). Quando un client vuole "osservare" una risorsa inoltra una richiesta GET e include la Observe option. Il server invia la risposta e registra il client come destinatario di notifiche qualora avvengano cambiamenti nello stato della risorsa. Quindi, una singola richiesta può risultare in più risposte, implementando

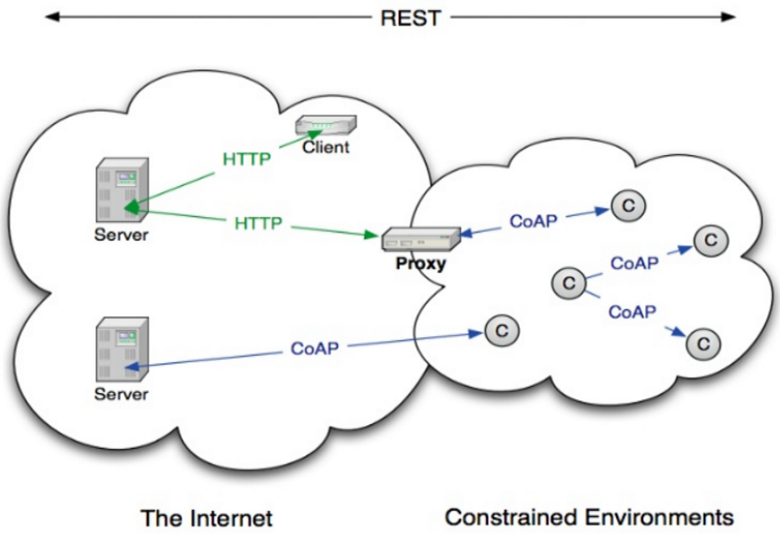


Figura 2.10: Cross-Proxy CoAP

un modello publish/subscribe [10, 32].

MQTT

MQTT è un protocollo per il trasporto di messaggi per client e server del tipo publish/subscribe. È leggero, aperto, semplice e progettato in modo da essere semplice da implementare. Queste caratteristiche lo rendono ideale per l'utilizzo in molti casi, incluso in contesti vincolati come per le comunicazioni Machine to Machine (M2M e IoT dove è richiesto un footprint ridotto del codice e/o la quantità di banda è limitata [4].
MQTT è stato inventato da Andy Stanford-Clark (IBM) e Arlen Nipper (Arcom, ora Eurotech) nel 1999 quando lo scopo era quello di creare un protocollo per ridotte capacità di banda e consumi di batteria per collegare tubazioni di petrolio su connessione satellitare.
MQTT non ha un significato ufficiale anche se per ragioni storiche si potrebbe usare l'acronimo di MQ Telemetry Transport dove per MQ potrebbe far rife-

[4]specifiche MQTT 3.1.1 - http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

rimento a un prodotto di IBM che supportava MQTT. Spesso viene chiamato Message Queue protocol ma non è corretto in quanto non esistono code come nelle soluzioni tradizionali di message queuing.
In alternativa al modello client/server dove un client comunica direttamente con un endpoint il pattern publish/subscribe disaccoppia un client (chiamato publisher), che invia un messaggio, da un altro client (chiamato subscriber), che lo riceve. C'è una terza entità chiamata broker che mette in comunicazione i due client, filtra i messaggi in arrivo e li distribuisce in modo opportuno.
Oltre ai principali aspetti del modello pub/sub il disaccoppiamento tra publisher e subscriber può essere riferito anche ad altre dimensioni:

- nello spazio - publisher e subscriber non devono conoscersi tra loro
- nel tempo - publisher e subscriber non devono essere presenti allo stesso istante
- comunicazione asincrona - le operazioni in entrambi i componenti non sono in attesa durante il trasferimento dei dati.

MQTT incarna completamente il modello appena esposto e disaccoppia lo spazio di publisher e subscriber ed è in grado di conservare i messaggi di client che non sono online. La maggior parte delle librerie client lavorano in modo asincrono e sono basate su callback permettendo al client di non bloccarsi nell'attesa di un messaggio o nel caso di invio.
Il Broker MQTT è in grado di ricevere tutti i messaggi, filtrarli e decidere tra tutti i client a lui registrati chi deve ricevere un particolare messaggio.

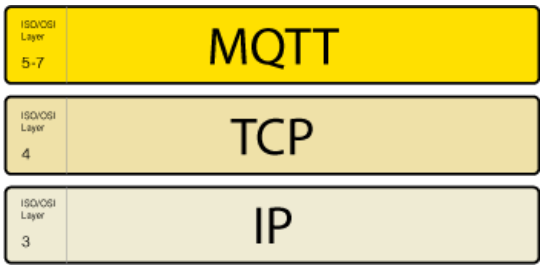


Figura 2.11: Stack MQTT

Il protocollo MQTT è costruito sopra TCP/IP e di conseguenza entrambi, tra client e broker, devono implementare uno stack TCP/IP. La connessione avviene sempre tra client e broker in quanto non è possibile trovare client connessi direttamente ad altri client. Una connessione inizia quando un client invia un messaggio di tipo CONNECT al broker. Il broker risponde con un CONNACK e uno staus code. Il broker mantiene aperta la connessione fino a quando il client non invia un comando di disconnessione o si perde la connessione. Dato che la connessione inizia sempre lato client non ci sono problemi nel caso di Network Address Translation (NAT) e il broker ha solitamente un indirizzo pubblico.

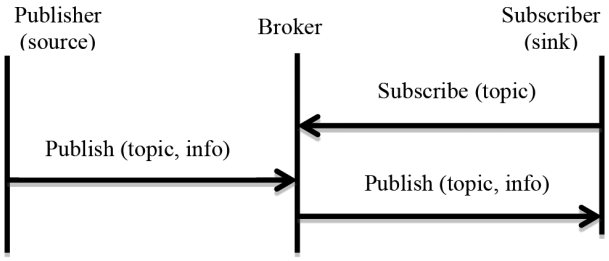


Figura 2.12: Trasferimento dati con MQTT

Dopo che un client si è connesso al broker può pubblicare i messaggi. Dato che il broker basa le sue regole di filtraggio su un soggetto ogni messaggio deve contenere un topic, che sarà utilizzato dal broker per inoltrare il messaggio ai client interessati, più il payload da trasmettere. MQTT è completamente indifferente al contenuto del payload che quindi dipende completamente dal publisher.
Per ricevere un messaggio è necessario inviare una messaggio di tipo SUBSCRIBE al broker. Questo tipo di messaggio è molto semplice e richiede solo la lista di topic a cui si è interessati.
Un topic non è altro che una stringa espressa in formato UTF-8 espressa in livelli separati da un carattere speciale (forward slash). Quando un client sottoscrive un topic per la ricezione di un messaggio può utilizzare la stringa esatta o può sottoscrivere più topic alla volta utilizzando le wildcards. Si tratta di simboli particolari inseriti a un certo livello del topic.

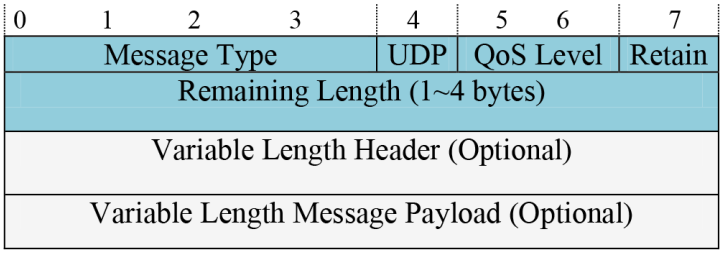


Figura 2.13: Formato messaggio MQTT

Una delle caratteristiche di MQTT è che è in grado di rendere più semplici le comunicazioni su reti inaffidabili perchè il protocollo gestisce le ritrasmissioni e garantisce la consegna dei messaggi. Inoltre permette ai client di scegliere il livello di QoS in base all'affidabilità della rete e alla logica di applicazione. QoS in MQTT è implementato a livelli:

- QoS 0 - at most once è il livello minimo che garantisce una consegna con minor sforzo. Il ricevitore non darà un ACK ed è chiamato spesso "fire and forget"
- QoS 1 - at least once garantisce che il messaggio sarà consegnato almeno una volta al ricevitore. Ma potrebbe essere consegnato anche più volte. Chi invia il messaggio lo memorizza fino a quando non riceve un ACK nella forma di PUBACK dal ricevitore. Il publisher reinvierà il messaggio (con flag DUP) se non riceverà un PUBACK in un tempo ragionevole.
- QoS 2 - garantisce che ogni messaggio sia ricevuto solo una volta. È il livello più sicuro ma anche il più lento.

2.4 IoT e WoT

Inizialmente col termine Internet of Things non ci si riferiva altro che a una interconnessione tra dispositivi [13]. Il termine non si riferiva a qualche tecnologia particolare o a una qualche struttura di rete ma semplicemente all'idea di connettere oggetti così come si potevano connettere i computer a internet.

Qualche dettaglio in più può essere ricavato da Atzori [2] che spiega come molti enti di standardizzazione e ricerca erano coinvolti nell'attività di sviluppo per realizzare una maggiore interoperabilità possibile tra dispositivi fornendogli un alto grado di adattamento e autonomia garantendo comunque sicurezza e privacy. Ad ogni modo era ancora difficile comprendere cosa si intendeva con il termine IoT e il problema nasceva proprio dalle parole da cui era composto; differenti visioni nacquero dal modo in cui enti di ricerca e alleanze commerciali, basandosi sui propri interessi, finalità e background approcciavano il termine.
Per chi utilizzava una prospettiva "Internet oriented", piuttosto che "Thing oriented", il web era diventato il principale mezzo di comunicazione in internet e sempre più servizi erano forniti tramite applicazioni web[32]. L'idea di accedere a dispositivi nelle immediate vicinanze tramite applicazioni web venne chiamata Web of Thing [13].
Per applicare i concetti della Web of Thing occorre riuscire a inserire dei web server in dispositivi con solo qualche centinaia di byte di RAM e pochi kbyte di EEPROM. Fortunatamente, e diversamente dai web server per internet, non è richiesto di gestire migliaia di richieste e connessioni simultanee, inoltre molti sforzi sono stati fatti per ridurre la dimensione dello stack TCP/IP riducendolo fino a richiedere pochi kbyte di RAM [13, 32].
Per integrare questi Smart Thing nel web si possono seguire due strade: Direct e Indirect Integration. Tramite Direct Integration occorre che i dispositivi siano indirizzabili e quindi che posseggano un indirizzo IP. Deve possedere un web server in modo da consentire che una *thing* possa accedere e comunicare con un'altra attraverso operazioni web stardard come, ad esempio GET e POST. Non tutti i dispositivi però sono in grado di integrare un web server per colpa delle ridotte risorse di cui dispone e, a dire il vero, non è necessario integrare direttamente tutte le smart thing nel web (ad esempio i sensori di una sensor network). In questo caso di Indirect Integration sono utilizzati degli *Intermediate Proxy* posizionati tra le smart thing e il web. Questi proxy sono solitamente chiamati *Smart Gateway* in quanto astraggono i protocolli proprietari o le API native offrendo un accesso uniforme tramite web API [32].

Dopo aver integrato differenti dispositivi, con varie capacità, nel web il passo successivo è quello di astrarre questi dispositivi in web services anzichè fornire semplicemente delle pagine statiche o dinamiche. I Web Service, definiti dal *World Wide Web Consortium* (W3C) come un sistema software progettato per supportare le comunicazioni *Machine to Machine* (M2M), seguono due maggiori paradigmi.

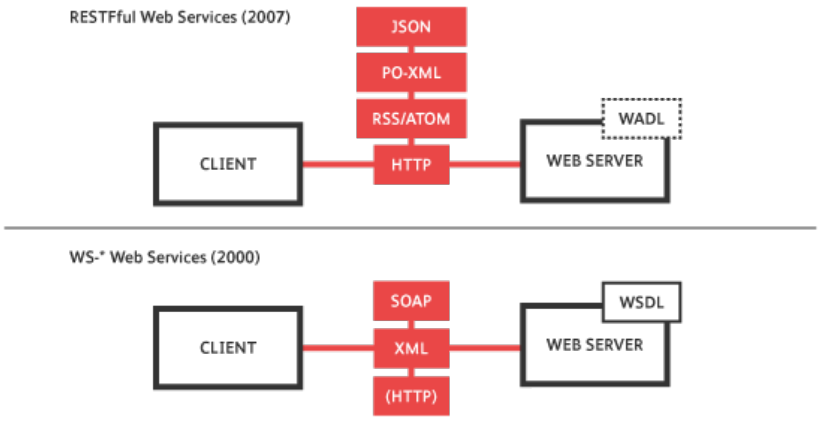


Figura 2.14: Web Service

Ci si riferisce ad architetture di tipo WS-* quando si parla di web service che utilizzano messaggi *Simple Object Access Protocol* (SOAP) con payload di tipo *Extensible Markup Language* (XML) e forniscono *Remote Procedure Call* (RPC) tra client e server tramite un protocollo di trasporto basato su HTTP. Il problema è che questo tipo di architettura è realmente molto pesante e non si adatta particolarmente bene al contesto della IoT.
Molto più snella è invece l'architettura *REpresentational State Transfer* (RESTful) considerata la reale architettura del Web. Il concetto di fondo di REST è che ad ogni risorsa è associato un *Universal Resource Identifier* (URI) in modo tale che ogni client possa identificare univocamente tale risorsa. HTTP non viene utilizzato come protocollo di trasporto come in WS-* ma come livello di applicazione quindi le risorse possono essere manipolate usando verbi di HTTP come PUT, GET, POST e DELETE. L'architettura RESTful risulta preferibile per la WoT principalmente per la bassa complessità e l'interazione stateless che

	WS-*	REST
HTTP	Transport protocol	Application protocol
Complexity	High	Low
Stateless	No	Yes
Mashup	No	Yes
Coupling	Tight	Loosely
Flexibility	Low	High
Security	Built-in	Self-defined

Figura 2.15: Confronto tra WS-* e REST

garantisce il loose-coupling tra gli oggetti [32]. Queste caratteristiche consentono di integrare l'architettura in dispositivi dalle risorse limitate e facilitano la composizione di web service (mashup).

3 Home Automation, Domotica e Smart Home

Il termine Domotica deriva dall'importazione del neologismo francese Domotique, a sua volta contrazione della parola greca Domos e di Informatique e rappresenta la disciplina che si occupa dell'integrazione dei dispositivi elettronici, degli elettrodomestici e dei sistemi di comunicazione e di controllo che si trovano nelle nostre abitazioni. Le origini risalgono intorno agli anni '70 quando si vennero a sviluppare i primi progetti concreti di automatismi legati alla gestione di impianti di allarme o altre funzionalità come l'accensione, lo spegnimento e la temporizzazione delle luci [25, 29].
Se vogliamo essere più precisi possiamo distinguere diversi termini: Home & Building Automation, Domotica e Smart Home. L'automazione di un edificio può consistere sia nell'automatizzare funzioni relative ad impianti elettrici e tecnologici sia nell'automazione di azioni normalmente eseguite dall'uomo: accensione delle luci comandate da sensori anzichè pulsanti/interruttori oppure regolazione automatica di elettrovalvole per il controllo di temperature ambiente. Nel caso residenziale si parla di Domotica o Home Automation ad indicare la misura più domestica nella gestione della casa mediante automatismi e tecnologie in grado, ad esempio, di migliorare il livello della qualità della vita oppure riuscendo ad avere un impatto sui consumi e di conseguenza sull'economia domestica. Il termine Smart Home invece sta ad indicare lo svecchiamento del termine ormai obsoleto di domotica per indicare la presenza di tecnologie (specialmente della IoT) in grado di interfacciarsi con la casa come involucro e come sistema impiantistico e la spinta a sviluppare ulteriori tecnologie in grado

di adeguare il comfort percepito basandosi sui bisogni, più o meno reali, di un abitante.

3.1 Passato, presente e futuro.

Nel ventesimo secolo si è assistito a una sensibile rivoluzione nella tecnologia domestica, una rivoluzione culminata verso la fine del secolo con il sorgere del concetto di Smart Home. La tecnologia domestica presente all'inizio del ventesimo secolo poteva essere tranquillamente riconosciuta e utilizzata da individui dei secoli precedenti mentre quella disponibile alla fine del secolo sarebbe risultata incomprensibile per chiunque [19].
Il primo impulso per questi cambiamenti è stato senz'altro l'introduzione nel primo quarto di secolo dell'elettricità nelle case che ha permesso la comparsa di nuovi elettrodomestici. Il secondo fu sicuramente l'introduzione delle tecnologie informatiche nell'ultimo quarto di secolo che ha permesso lo scambio di informazioni tra le persone, tra sistemi e dispositivi nelle abitazioni e oltre alle mura domestiche.
Nella sua analisi storica Frances K. Aldrich [19] raccoglie, tra avvenimenti e aspetti sociali, i motivi per l'adozione delle nuove tecnologie in ambito domestico. Soprattutto fa una distinzione tra prodotti che "fanno risparmiare tempo" e prodotti che "occupano il tempo". I prodotti che "fanno risparmiare tempo" sono quelli che lasciano più tempo a disposizione riducendo il tempo richiesto per completare un particolare compito - ad esempio una lavatrice. I prodotti che "occupano il tempo" sono quelli che riducono il tempo a disposizione ma ne aumentano la qualità percepita - ad esempio la televisione. Quello che viene fatto notare sono i tempi con cui si sono diffusi questi tipi di prodotti: lavatrice, aspirapolvere, frigorifero hanno impiegato decadi per essere adottati e la loro diffusione è dipesa dal reddito familiare mentre, ad esempio, televisione e radio hanno raggiunto pari livelli di diffusione in pochi anni e con un legame molto meno marcato al reddito familiare. Inoltre si può notare come il tempo risparmiato grazie agli elettrodomestici è stato prontamente occupato dalla te-

levisione e, in particolare verso la fine del secolo, attraverso la connessione di dispositivi e computer verso informazioni e servizi esterni rispetto all'abitazione. Nonostante il concetto di "Smart House" fosse ben presente e dalla metà del 1970 cominciavano a comparire i primi protocolli per la Home Automation solo una piccola parte di installazioni vennero costruite e vendute contrariamente alle previsioni favorevoli iniziali. Tra i motivi suggeriti da Gann [15] i principali ostacoli erano dovuti a:

- un grosso investimento iniziale richiesto al compratore limitando quindi il mercato a famiglie con reddito elevato. E i potenziali acquirenti dovevano essere prima convinti dei possibili benefici che potevano trarne
- specialmente in Europa, vista la diffusione di case già costruite, dovevano essere trovate soluzioni per modificare case già esistenti (che è più costoso che predisporla al momento della costruzione)
- mancanza di un protocollo unico e di conseguenza, sempre in Europa, l'industria tendeva a focalizzarsi su semplici sistemi di controllo remoto di accensione/spegnimento che non richiedevano installazioni di particolari reti
- i fornitori avevano adottato un approccio del tipo "technology push" senza prestare bene attenzione nell'ascoltare le necessità degli utenti. In particolar modo mancava di superare l'accettazione da parte delle donne alle quali ancora rimaneva gran parte dei compiti domestici
- il grado di formazione richiesto prima di poter utilizzare i sistemi proposti

Ci sono analogie tra l'adozione dei primi elettrodomestici e l'accettazione da parte degli utenti di nuovi sistemi di automazione. A parte le ovvie caratteristiche di semplicità di gestione, robustezza, flessibilità, aggiornamento, semplicità di installazione e soprattutto economicità, occorre superare le paure sulla sicurezza e i rischi a cui si va incontro da parte dei possibili compratori.
Anche l'ambito della ricerca non era particolarmente attratto dalla Home Automation e le ragioni andavano ricercate nella mancanza di motivazione ad

aumentare la produttività nel lavoro domestico. In effetti le decisioni sui prodotti da acquistare in ambito lavorativo sono determinate da argomentazioni riguardanti la produttività mentre i proprietari delle abitazioni sono interessati all'estetica, alla moda e alla volontà di apparire e in una famiglia le decisioni vengono prese in modo differente rispetto a una azienda.
Inoltre i progettisti ritenevano la tecnologia domestica non particolarmente attraente e continuavano a concentrare i loro sforzi sui singoli elettrodomestici dotandoli di maggiori funzionalità, magari aggiungendo connettività verso l'esterno, e spesso anche solo per ragioni di marketing.
In una ricerca effettuata da un punto di vista delle scienze sociali Berg nota come nonostante i lavori ripetitivi, che occupano più tempo in una abitazione, sono quelli non retribuiti ed effettuati dalle donne cucinare, lavare, pulire, riordinare sono molto spesso lavori trascurati da chi realizza prototipi per le Smart House. Questo studio, sebbene realizzato ormai molti anni fa (ma tutto sommato ancora attuale), ha portato alla conclusione come ancora una volta si ricade in un caso di "technology push" anzichè quello di "consumer pull" ed è dovuto principalmente ad ostinarsi al realizzare ciò che è tecnicamente possibile anzichè quello che è desiderabile.
Negli ultimi anni si è assistito a un cambiamento negli attori del mercato delle tecnologie per la Home Automation. Inizialmente erano i fornitori di prodotti e materiale elettrico che guidavano il mercato fornendo i loro sistemi proprietari, aperti o meno all'integrazione con altri sistemi, mentre più recentemente sempre più produttori di dispositivi elettronici fanno ingresso spingendo i loro elettrodomestici, dispositivi e sistemi affiancando al prodotto il termine "Smart" a indicare funzionalità evolute. Si parla di nomi del calibro di Samsung ed LG che offrono sistemi acquistabili direttamente sul mercato consumer.
Anche i fornitori di servizi, come per l'energia elettrica, si sono affacciati al mondo della domotica proponendo soluzioni in collaborazione con produttori di elettrodomestici o di sistemi di Home Automation. Ad esempio Enel con il sistema Energy@home, avviato nel 2009 in collaborazione con Electrolux, Indesit Company e Telecom Italia con l'obiettivo di sviluppare prodotti e servizi

basati sull'interoperabilità e la collaborazione degli elettrodomestici oppure Eni con prodotti e soluzioni di Acotel Net per la domotica e in particolare per il risparmio energetico.
Alle nuove tecnologie invece spetta il compito di risolvere alcuni dei problemi che inizialmente limitavano l'adozione di domotica da parte delle famiglie:

- tecnologie a radio frequenza come il Wi-Fi, Bluetooth, ZigBee e altre per risolvere il problema di non poter modificare impianti già preesistenti o difficoltà nell'eseguire opere murarie nelle abitazioni già costruite
- hub elettronici per superare l'ostacolo dovuto a un protocollo comune di interoperabilità tra apparecchiature di produttori differenti
- riduzione dei costi grazie a un mercato sempre più ricco di dispositivi elettronici a costi sempre minori e sistemi con un grado maggiore di modularità in modo da far crescere nel tempo il sistema anche successivamente all'acquisto iniziale

Possono rimanere problemi legati all'usabilità dei prodotti e una attenzione maggiore ai bisogni dell'utenza. Ad ogni modo la diffusione degli smartphone, tablet ma soprattutto di internet e la diffusione dei social network ha in qualche modo abituato gli utenti a scontrarsi con le nuove tecnologie e a diventare più smaliziati nell'approcciarsi ad esse e a "spendere tempo" nel tentativo di utilizzarle.
Ma quand'è che si può parlare di Smart Home?
Già Aldrich [19] più di dieci anni fa affermava che il termine era applicato in modo poco stringente e qualsiasi cosa da una casa con un sistema a circuito chiuso di videosorveglianza a installazioni molto complesse che comprendevano computer, elettrodomestici, sistemi di sicurezza, sensori di luminosità e di temperatura venivano definite Smart Home.
Questa affermazione potrebbe anche essere applicata tutt'ora in quando difficilmente potrebbe essere trovato un qualcosa che rivoluzioni radicalmente il modo di vivere.
Sempre Aldrich [19] propone una gerarchia di possibili Smart Home:

1. Homes which contain intelligent objects - che contengono cioè elettrodomestici, dispositivi e oggetti che sono in grado di lavorare in maniera intelligente

2. Homes which contain intelligent, communicating objects - che contengono elettrodomestici, dispositivi e oggetti che non solo lavorano autonomamente in maniera intelligente ma che scambiano informazioni gli uni con gli altri per incrementare le funzionalità

3. Connected Homes - che hanno reti interne ed esterne che permettono un controllo remoto e interattivo così come l'accesso a servizi all'interno e all'esterno dell'abitazione

4. Learning homes - in cui pattern di attività nella casa sono percepite, memorizzate e analizzate in modo da anticipare i bisogni dell'utente e in grado di controllare la tecnologia della casa.

5. Attentive homes - in cui l'attività e la posizione delle persone e gli oggetti all'interno della casa sono costantemente registrate e questa informazione è utilizzata per anticipare nuovamente i bisogni degli occupanti.

Sicuramente se un cambio di paradigma deve avvenire nel modo in cui viene vissuta la tecnologia domestica questo dovrà avvenire sicuramente considerando il livello assegnato alla Attentive Home che presenta potenziali qualità di flessiblità, proattività e reattività in modo da offrire un ambiente qualitativamente differente ai suoi abitanti.

3.2 Domotica e Home Automation

La differenza fondamentale tra un impianto elettrico tradizionale e un sistema di Home Automation deriva essenzialmente dal livello di integrazione. Nella progettazione tradizionale i vari servizi potrebbero essere assicurati da impianti diversi ed indipendenti che però non riuscirebbero a interagire fra loro; inoltre la

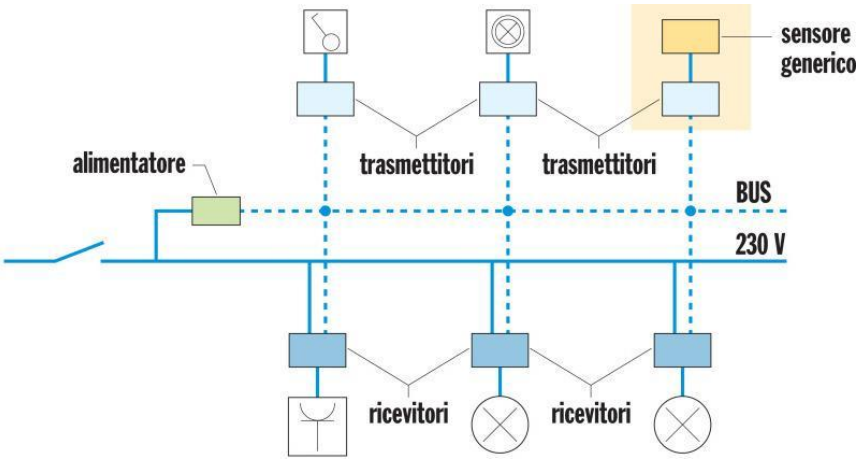


Figura 3.1: Cablaggio di tipico impianto domotico

loro coordinazione porterebbe a costi d'implementazione rilevanti e soprattutto ad una minor efficacia nel garantire possibili modifiche nel tempo [29].
Al contrario i sistemi d'automazione vengono progettati proprio per far integrare vari sottoinsiemi ognuno con la propria funzionalità. Facendo parte dello stesso sistema essi sono perfettamente in grado di colloquiare ed interagire con tutti gli altri. Ciò significa che, grazie a questo livello di integrazione, si riescono ad ottenere risultati che gli impianti tradizionali non sono in grado di offrire.
Ad esempio in un impianto elettrico tradizionale la linea di alimentazione collega il pulsante (o interruttore) direttamente al carico, ad esempio una lampada. Per gestire questo carico di conseguenza occorre collegare ad esso tutti i suoi dispositivi di comando quindi, ad ogni variazione o aggiunta di funzioni, occorre modificare il cablaggio. In un impianto domotico, invece, a seguito della pressione di un pulsante viene generato un evento e la presenza di un circuito logico permette di inviare l'informazione a uno o più attuatori che eseguiranno l'azione richiesta [29].
Questa complicazione per accendere semplicemente una luce potrebbe sembrare eccessiva ma se pensiamo a impianti particolarmente estesi sicuramente il numero di componenti e connessioni si ridurrebbe sensibilmente. Inoltre nell'evenienza di dover modificare o estendere alcune funzioni il più delle volte si riconduce a una riprogrammazione della logica di funzionamento riducendo

sicuramente i costi di manutenzione.

Ambiti funzionali

Dai vari manuali di domotica dei vari produttori di dispositivi o di elettronica per impianti elettrici è possibile ricavare molte informazioni su quelli che vengono ritenuti da questi fornitori gli ambiti di applicazione di automatismi appartenenti al contesto della home automation.

Illuminazione

L'illuminazione è la risultante dell'effetto dell'utilizzo dei flussi luminosi naturali (mediati da elementi architettonici) o emessi da sorgenti artificiali (apparecchiature generalmente elettriche) allo scopo di ottenere determinati livelli di luce (illuminamenti); la relativa tecnica si chiama illuminotecnica. Ulteriori scopi dell'illuminazione possono essere: creare effetti scenografici o di accento oppure fare arredo. Molto spesso il termine illuminazione è anche usato come semplificazione di "impianto di illuminazione".

Aperture motorizzate

Il sistema domotico permette di comandare le aperture motorizzate di qualsiasi genere (avvolgibili, tende da sole, oscuranti ecc. ecc.) sia singolarmente che in maniera coordinata con tutte le altre apparecchiature di casa. Un caso molto frequente nelle abitazioni di nuova costruzione, sono le tapparelle automatizzate con motori con alimentazione a 230 Vca. Il cosiddetto "attuatore" è un motoriduttore installato dentro o a lato del rullo sul quale si avvolge la tapparella e generalmente il comando è dato con un semplice deviatore che commuta direttamente la tensione di rete.

Riscaldamento/Raffrescamento

Con il termine "comfort ambientale" si definisce quella particolare condizione di benessere determinata, in funzione delle percezioni sensoriali di un indivi-

duo in un ambiente, da temperatura, umidità dell'aria e livello di rumorosità e luminosità rilevati all'interno dello stesso: è quindi possibile distinguere tra benessere termo-igrometrico, benessere acustico e benessere luminoso. Il comfort ambientale si identifica con il benessere psicofisico delle persone che vivono un ambiente (casa, ufficio) ed è una sensazione dipendente da determinate condizioni che sono in gran parte pianificabili e quindi rientranti nella responsabilità del progettista (ad esempio nelle fasi di progettazione, realizzazione e gestione di un "green building"). Il benessere termo-igrometrico (thermal comfort in inglese) è definito dall'American Society of Heating Ventilation and Air-conditioning Engineers (ASHRAE) come quel particolare stato della mente che esprime soddisfazione con l'ambiente circostante.
Le variabili ambientali da cui dipendono le condizioni climatiche esterne ed interne all'edificio e che influenzano il benessere termo-igrometrico sono:

- Temperatura dell'aria: si misura in °C o °F.
- Umidità relativa dell'aria interna: indica il rapporto tra la quantità di vapore contenuto da una massa d'aria e la quantità massima che ne può contenere quella massa d'aria nelle stesse condizioni di temperatura e pressione. Si misura, quindi, in percentuale (%).
- Temperatura media radiante: espressa in °C o °F, è la media delle temperature delle pareti interne all'ambiente, compresi soffitto e pavimento.
- Velocità dell'aria: espressa in m/s.

Temporizzazioni

In genere tutti i sistemi domotici consentono di attivare le funzioni previste attraverso cosiddette "interfacce uomo macchina" (Human Machine Interface in inglese) più o meno convenzionali quali pulsanti, interruttori, pannelli touch-screen, telecomandi ecc... che sottintendono l'intervento manuale dell'utilizzatore. Una delle caratteristiche fondamentali di un sistema "intelligente" è

consentire il controllo anche senza la presenza fisica dell'utilizzatore: ad esempio accendere il riscaldamento o disattivare l'impianto d'irrigazione senza essere in casa oppure eseguire un'azione in momenti della giornata in cui si è impossibilitati ad attivarla manualmente (ad es. mentre si sta dormendo). In tutti questi casi, una gestione "a tempo" delle funzioni di sistema, risulta utile ai fini di ottenere in modo semplice un vero "comfort".

Scenari

Un sistema domotico può essere usato per eseguire una singola azione, (ad esempio accendere una lampada o impostare una temperatura) o per eseguire automaticamente una sequenza ripetitiva di operazioni in diversi momenti della giornata o provvedere autonomamente ad attivare le funzioni conseguenti all'accadimento di un certo evento: questa "sequenza di operazioni" viene normalmente definita "scenario". Ad esempio si potrà programmare uno scenario "buongiorno" contenente tutte le azioni che normalmente vengono eseguite al risveglio mattutino (accensione luci, riscaldamento, disinserimento allarme, ecc.). Riguardo alle modalità di attivazione si possono distinguere:

- scenari in cui le operazioni sono programmate a tempo e devono essere eseguite in un determinato momento del giorno o di tutti i giorni;
- scenari in cui le operazioni vengono attivate dall'accadere di determinati eventi rilevati da sensori (di presenza, di temperatura ecc...);
- scenari in cui le operazioni vengono eseguite a seguito di un comando dell'utente sia "in locale" tramite pulsanti o terminali che "da remoto" tramite comunicatore GSM.

Un impianto domotico mette a disposizione una serie di funzioni innovative che nascono dall'integrazione di diversi sistemi; non si tratta dunque di migliorie applicate ad un singolo dispositivo, ma funzionalità disponibili solo ed esclusivamente con la gestione efficiente e contemporanea di più sistemi. Un esempio sono i cosiddetti "scenari" domotici, cioè la possibilità di attivare in modo coordinato luci, attuazioni motorizzate, temporizzazioni, impianto di anti-intrusione,

di termoregolazione, di irrigazione ecc... con un unico comando manuale. Oppure, in modo automatico, si può pensare di attivare uno scenario in base alle indicazioni date da un sensore di intensità luminosa, da un anemometro, da un igrometro ecc... o mediante la combinazione di comandi a pulsante oppure da un comando dato dal sistema anti-intrusione. Le possibili combinazione per gli "scenari" sono limitate solo dal numero e dal tipo di carichi comandati dall'impianto domotico, per cui è possibile personalizzare qualsiasi "scenario" domotico in base alle esigenze dell'utente finale. Gli scenari possono essere attivati anche da remoto per esempio attraverso un semplice SMS; allo stesso modo il sistema domotico può inviare le informazioni riguardanti lo stato dell'impianto in tempo reale, così da avere un controllo completo in qualsiasi situazione.

Videocitofonia

La citofonia, concepita in Italia nei primi anni '60, costituì sin da subito un reale miglioramento della qualità della vita nelle abitazioni. Fino a quel momento, le chiamate si effettuavano con semplici pulsantiere ed un campanello interno costringendo il proprietario ad aprire la porta per poter verificare l'identità del visitatore.
Con l'avvento dell'elettronica, sono stati inseriti all'interno delle pulsantiere i cosiddetti "gruppi fonici" (costituiti da microfono ed altoparlante) e sono stati creati i primi "derivati citofonici" con i quali poter effettuare, stando all'interno dell'abitazione, la conversazione con il posto esterno.
La successiva implementazione funzionale fu quella del comando per l'elettroserratura al fine di agevolare l'accesso soprattutto negli stabili condominiali. In ultimo, la richiesta di avere un derivato citofonico in più stanze dell'abitazione, consentì lo sviluppo della funzione di "intercomunicazione". Ancora oggi, a distanza di decenni, le funzioni "base" di un impianto citofonico sono rimaste sostanzialmente immutate.
Negli anni '80, l'avvento di telecamere e tubi catodici di dimensioni ridotte e di costo contenuto, ha portato alla conseguente aggiunta nel posto esterno della telecamera stessa, ed alla realizzazione di derivati interni videocitofonici

(inizialmente viva-voce o parla-ascolta) con ulteriore miglioramento della sicurezza grazie alla possibilità di identificare correttamente il chiamante anche in condizioni di audio disturbato.

Irrigazione

L'irrigazione è l'insieme delle conoscenze tecniche finalizzate all'incremento della produttività di un terreno e all'ottimizzazione dell'utilizzo dell'acqua ad esso necessaria. L'irrigazione può anche essere vista come fattore di incremento della qualità e tutela ambientale vista la rilevanza del "verde" nel paesaggio urbano. Il giardino o il parterre fiorito della piazza urbana risultano essenziali al livello di gradimento: tanto maggiore è il valore estetico della componente verde di un'area tanto maggiore sarà il suo valore economico. Per mantenere il prato, la siepe o l'aiuola fiorita al massimo del suo splendore è necessario irrigare tenendo conto che la scelta di una certa varietà botanica o tipologia colturale comporterà fabbisogni irrigui specifici: la piena consapevolezza di queste implicazioni è essenziale alla riuscita di un progetto. L'irrigazione può, essere vista anche come limitatore dei costi di reimpianto contribuendo ad evitare nuovi investimenti al mutare della stagionalità.

Controllo dei carichi e dei consumi elettrici

La funzione "controllo dei carichi" impedisce lo scatto dell'interruttore generale per superamento dell'assorbimento massimo mediante la disinserzione preventiva di uno o più carichi elettrici controllati. Questa funzione è particolarmente importante nelle abitazioni che hanno contratti con il fornitore di energia elettrica per potenze nominali molto inferiori al potenziale consumo di picco: ad esempio difficilmente un boiler elettrico può funzionare contemporaneamente a frigo, congelatore e lavatrice senza che l'interruttore generale intervenga. In generale il dimensionamento della potenza richiesta da un impianto elettrico si effettua sommando le potenze richieste dai carichi elettrici più importanti in termini di consumo e moltiplicando tale somma per il "fattore di contemporaneità" (che normalmente oscilla fra 30% e 70%) il quale dipende dall'utilizzo

previsto: il "controllo carichi" permette di abbassare notevolmente questo "fattore di contemporaneità" e quindi consente di "risparmiare" la potenza richiesta dall'impianto. In un mondo in cui i costi dell'energia sono in continua e costante crescita diventa essenziale avere un sistema che aiuti a evitare sprechi inutili senza inficiare il livello di benessere e comfort quotidiano. Al puro "controllo carichi" spesso vengono attribuite funzioni di "risparmio energetico" che non possono essere soddisfatte a pieno da una semplice lista di priorità di disinserzione. La funzione che garantisce il vero e proprio risparmio energetico è il "controllo dei consumi".

Sicurezza Antintrusione

La crescente diffusione della microcriminalità fa sì che la sicurezza antintrusione debba essere, inclusa nelle funzionalità di "base" di un impianto domotico al fine di sorvegliare i propri beni ma di proteggere le persone. Anche in questo campo, l'evoluzione tecnologica ha reso possibile la realizzazione di sistemi di allarme semplici nell'uso, adeguatamente affidabili e poco invasivi in termini "estetici"; si può facilmente prevedere che la disponibilità di microprocessori sempre più potenti che consentiranno l'adozione di software più raffinati e l'utilizzo di nuovi principi fisici nelle tecniche di rilevazione porteranno nei prossimi anni ad ulteriori significativi progressi. Ci sono molte scelte che si possono fare per assicurare la tranquillità e la sicurezza, una di queste è proteggere la propria casa dai possibili furti, installando un impianto antifurto innovativo, affidabile e di alta qualità. Data la natura intrinseca della funzione svolta, il "sotto-sistema" di antintrusione non deve in nessun caso essere soggetto a "fuori servizio" e il suo funzionamento non deve essere in alcun modo impedito o parzialmente limitato da guasti o malfunzionamenti che dovessero occasionalmente verificarsi nel resto dell'impianto (per esempio a seguito di un semplice black-out). Per questo motivo e per l'ottemperanza alle specifiche norme di prodotto italiane ed europee, il "sotto-sistema" di antintrusione deve essere separato da un punto di vista installativo: pertanto l'unità centrale del sotto-sistema svolge i suoi compiti specifici in modo autonomo e non influenzabile dal resto dell'im-

pianto domotico ed il "bus di comunicazione" è cablato separatamente con gli accorgimenti "anti-manomissione" del caso per garantire il massimo livello di sicurezza previsto dalle normative vigenti. Ma le precedenti ineludibili necessità installative non debbono avere alcun impatto nella semplicità di utilizzo del sotto-sistema di antintrusione da parte dell'utilizzatore finale che anzi molto spesso è restio all'adozione di simili sistemi di protezione proprio a causa degli ostici strumenti di interfaccia messi generalmente a disposizione.

Controllo da remoto

La maggior parte degli individui trascorre gran parte della giornata al di fuori delle mura domestiche sia per impegni di lavoro che per attività di "tempo libero". È quindi indispensabile che un sistema domotico sia dotato di strumenti tramite i quali l'utilizzatore possa intervenire anche non essendo fisicamente presente sull'impianto. La soluzione più semplice a questo tipo di esigenza, è dare la possibilità di interagire con la propria abitazione tramite i telefoni cellulari su rete GSM.

Gestione efficiente delle risorse energetiche

Risparmio energetico significa risparmio economico: basta semplicemente predisporre un impiantistica in grado di ottimizzare l'utilizzo dei carichi per ridurre notevolmente l'importo della bolletta sulla quale, in maniera non marginale, incidono gli sprechi dovuti a un utilizzo inefficiente degli impianti (luci dimenticate accese, riscaldamento continuo di stanze utilizzate raramente ecc...). La gestione efficiente delle risorse energetiche mediante la Domotica permette anche di evitare il superamento dei limiti contrattuali dell'erogazione dell'energia elettrica e il conseguente black out: è possibile infatti fare in modo che gli elettrodomestici funzionino consumando solo il livello di energia prestabilito per quella fascia oraria, eventualmente disattivando uno o più carichi secondo una priorità assegnabile in base al tipo del carico, al giorno della settimana e all'ora del giorno. Infine non va mai dimenticato che risparmiare vuol dire anche realizzare oggi lavori che, in futuro, si renderanno necessari per fruire di tecnologie

che presto potrebbero diventare alla portata di tutti. Predisporre oggi significa avere la consapevolezza che le necessità di chi usufruisci dell'abitazione variano nel tempo e che realizzare o adeguare gli impianti in futuro potrebbe essere impossibile o diventare complicato e soprattutto molto costoso.

Protocolli principali

X-10

Il protocollo X-10, nato nel 1976, è lo standard storico della domotica. X-10 è basato su una semplice architettura centralizzata, in cui un singolo dispositivo di controllo può controllare fino a 256 tipi di periferiche (è possibile assegnare uno stesso indirizzo a più periferiche per far eseguire a tutte gli stessi comandi). È inoltre possibile controllare i vari dispositivi sfruttando dei telecomandi a onde radio. Il dispositivo di controllo è gestibile anche mediante un personal computer. Per consentire la diffusione del protocollo sono state scelte come mezzo di comunicazione le power-line. Il protocollo è molto semplice e prevede che a ogni periferica venga assegnato un indirizzo statico di 8 bit (i primi quattro tipicamente rappresentati da una lettera da A a P e i secondi quattro da un numero da 1 a 16) a tempo di installazione. L'invio di un comando da parte del controllore o di un telecomando prevede l'invio in broadcast sul bus utilizzato di 12 bit, i primi 8 rappresentanti l'indirizzo della periferica che deve eseguire il comando, e i restanti 4 rappresentanti il comando stesso. È evidente quindi come tramite X-10 sia possibile l'automazione solamente di semplici funzionalità domestiche. L'estrema semplicità e il basso costo di installazione e realizzazione di prodotti compatibili hanno fatto sì che le aziende realizzassero molti prodotti idonei allo standard e quindi, ancora oggi, lo X-10 è molto diffuso.

BACnet

Lo sviluppo del *Building Automation and Control Networking Protocol* (BACnet) è iniziato nel 1987 quando l'ASHRAE (*American Society of Heating, Refrigerating and Air-Conditioning Engineers*) non riusciva a trovare un protocollo

adatto a soddisfare tutti i requisiti necessari per uno standard di comunicazione di building automation. Lo sviluppo terminò nel 1995 quando BACnet fu pubblicato come standard ANSI/ASHRAE e nel 2003 è stato adottato come sandard ISO e utilizzato in UE, Russia, Korea, Cina e Giappone [21].
Prima dello sviluppo di BACnet, tutti i dispositivi di controllo erano connessi tra loro attraverso protocolli proprietari. Ogni produttore aveva un suo protocollo di comunicazione. Era così necessaria la creazione di reti diverse, una per ogni produttore, con costi di integrazione particolarmente ingenti. Mettere in comunicazione dispositivi di produttori diversi diventava quindi un'operazione onerosa.
I messaggi BACnet possono essere veicolati su qualsiasi tipo di rete (Ethernet, ARCNET, Master-Salve/Token-Passing, LonTalk e Point-to-Point) ma il protocollo IP è stato ritenuto particolarmente utile e quindi è stato formalizzato il BACnet/IP. Quindi le reti IP sono supportate nativamente dallo strato di rete di BACnet che permette ai propri dispositivi di comunicare direttamente utilizzando IP anzichè sfruttare router per il tunneling.
L'elemento base della topologia di rete BACnet è il *segment*. I *segment* possono essere connessi tramite repeater e bridge per formare una rete. Le reti BACnet (solitamente composte da mezzi di trasmissione differenti) sono connesse da router per formare una internetwork ma solo un percorso può esistere tra due dispositivi qualsiasi di una internetwork. Un indirizzo consiste di 2-byte per il numero di rete e un indirizzo locale composto al massimo di 255 byte. L'indirizzo locale è specifico al metodo utilizzato per il collegamento, ad esempio, un indirizzo IP nel caso di BACnet/IP oppure un MAC address per una LAN.
Le modifiche più recenti descrivono la possibilità di utilizzare XML e Web Service per l'integrazione di BAC con altri sistemi di gestione a livello enterprise (BACnet/WS). La figura 3.2 mostra un esempio di una configurazione con BACnet/IP con l'utilizzo di un Web server per l'interfaccia grafica e Web service e una workstation che utilizza una applicazione client tradizionale.
Il protocollo BACnet definisce un insieme di servizi per il discovery di device e object. Sono definiti 25 tipi differenti di object. Ci sono semplici object come

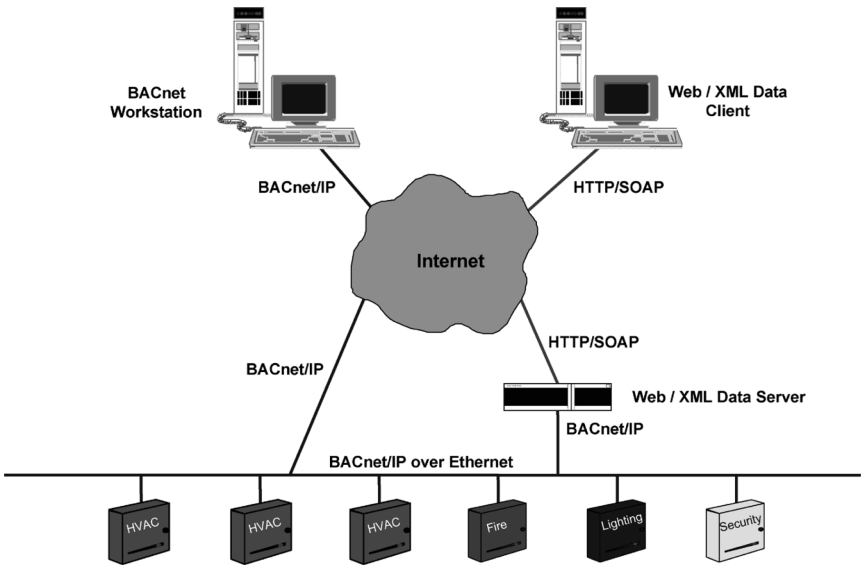


Figura 3.2: Esempio di rete BACnet [21]

Analog Input e Output, Binary Input e Output, Multi-State input ma anche diversi object più complessi e l'object device stesso. Ogni BACnet object è una collezione di *data element* che si riferistono a una particolare funzione. I singoli *data element* sono chiamate le *property* dell'object, ad esempio, un object Analog Input che fornisce la temperatura di una stanza avrà una *property* per il valore corrente e altre *property* per descriere il tipo di sensore, i valori minimo e massimo accettati, la risoluzione e l'unità con cui esprimere il valore. Ogni device può avere zero, uno o più tipi di object fatta eccezione del device object che deve essere presente in ogni device. Quindi ogni device a ogni livello dichiara le proprie grandezze e funzionalità ai dispositivi che ne fanno richiesta. Mentre gli object forniscono una rappresentazione del device nel sistema di automazione i BACnet service forniscono messaggi per accedere e manipolare queste informazioni. Le comunicazioni avvengono secondo un modello client/server e sono definiti 40 servizi raggruppati in cinque categorie tra cui allarmi ed eventi. BACnet rappresenta attualmente la più completa astrazione di comunicazione tra dispositivi per la building automation.

KNX/Konnex

KNX è nato in seguito alla fondazione dell'associazione Konnex, avvenuta nel maggio del 1999 da parte di EIBA (European Installation Bus Association), BCI (Batibus Club International) ed EHSA (European Home System Association) con lo scopo di realizzare e promuovere uno standard unico per applicazioni di Home e Building Automation. Konnex è l'evoluzione dello standard EIB (col quale rimane completamente compatibile) nato grazie all'aggiunta delle funzionalità e il supporto ai mezzi di comunicazione degli standard BatiBus e EHS. Il nuovo standard KNX cerca di combinare i loro aspetti migliori per creare un unico standard Europeo per sistemi di Home & Building Automation [29, 21].
Lo standard Konnex prevede l'utilizzo di alcuni mezzi trasmissivi che possono essere combinati tra loro:

- cavo twistato in rame tramite di tipo TP-0 e TP-1: il primo di derivazione BatiBUS che può raggiungere la bitrate di 4800 bit/s e il secondo derivato da EIB che può raggiungere le velocità di trasmissione di 9600 bit/s.
- onde radio nella banda degli 868 MHz, frequenza interna alla banda ISM (Industrial Scientific Medical) con velocità di 16,38 kbit/s. I dispositivi KNX RF dialogano in maniera peer to peer.
- Power Line utilizzando la tecnologia PL-110 di derivazione EIB con velocità di 1200 bit/s e PL-132 di derivazione EHS con velocità di 2400 bit/s riferite a linee di potenza con 230 Vac e 50 Hz.
- infrarosso
- KNX può utilizzare anche connessioni ethernet mediante tunneling su rete IP. Lo standard utilizza il tunneling su frame IP per la gestione remota di installazioni KNX e tramite tecniche di routing è possibile connettere tramite backbone diverse installazioni KNX.

La rete KNX è a logica distribuita e quindi non è presente un dispositivo centrale di decisione; ogni nodo possiede un indirizzo a 16 bit per cui è possibile

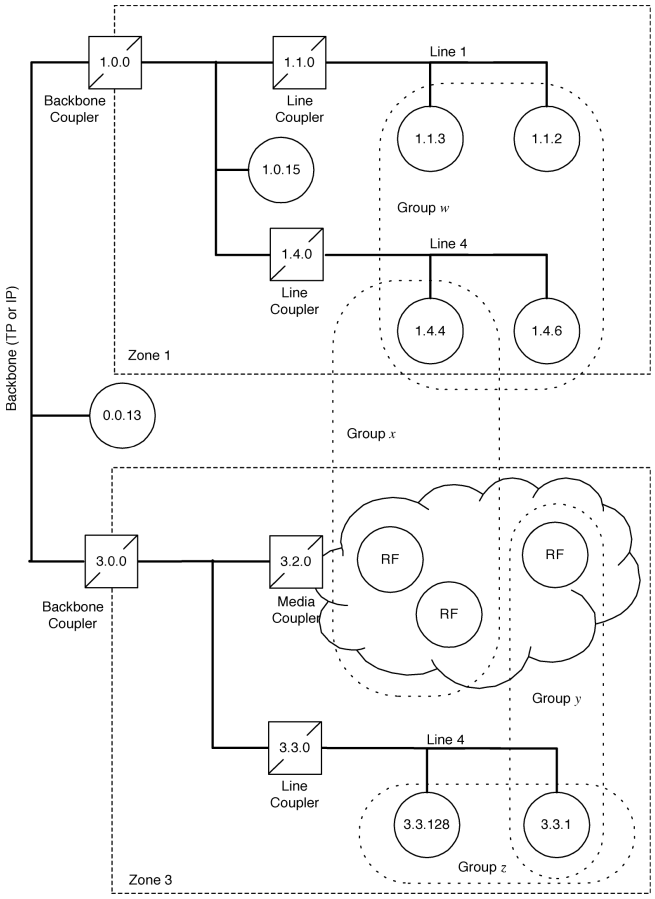


Figura 3.3: Topologia sistema KNX [21]

in teoria indirizzare fino a 65536 dispositivi. Sono ammesse tutte le possibili combinazioni di topologie tra quelle a stella, ad albero e a bus, ma non quella ad anello. La rete è strutturata in *line* contenenti al massimo 256 dispositivi e una singola *line* può essere composta di un massimo di 4 segmenti che possono raggiungere 1000 metri l'uno. Per concatenare i quattro segmenti possono essere utilizzati opportuni bridge (chiamati *line repeater*). L'insieme di 15 line può essere raggruppata in quello che viene definita *zone* e una linea particolare chiamata backbone line può raggruppare fino a 15 zone. Ne deriva una struttura gerarchica (vedi figura 3.3).
Ad ogni nodo di una rete KNX viene assegnato un indirizzo univoco che corrisponde alla sua posizione all'interno della struttura topologica della rete (zone/-line/device). Questo indirizzo viene utilizzato esclusivamente per comunicazioni di tipo unicast mentre viene utilizzato anche un indirizzamento di tipo multicast implementato al livello due dello standard OSI (livello data link) dove ad ogni nodo viene assegnato anche un ulteriore indirizzo di gruppo [29]. Gli indirizzi di gruppo seguono la seguente gerarchia a tre livelli:

1. gruppo principale o livello di sistema (es. illuminazione, termoregolazione, ecc.)

2. gruppo centrale per la funzione particolare del livello considerato (es. interruttore o dimmer per il livello di illuminazione, automazione tapparelle, ecc.)

3. sottogruppo per i dispositivi che eseguono la stessa funzione (es. luce salotto, finestra bagno, ecc.)

I tre campi sono separati dal carattere *slash* (/) e possono essere assegnati a piacimento (gruppo principale/gruppo centrale/sottogruppo).
Lo standard è completamente aperto e dispone di funzionalità Plug & Play e di un efficace metodo di correzione degli errori, allo scopo di assicurare un'alta affidabilità al sistema. Il sistema è inoltre in grado di autoconfigurarsi eliminando autonomamente apparecchi non funzionanti o inserendone di nuovi.

Lo standard KNX prevede diversi metodi di configurazione: i principali sono la Easy-Mode e la System-Mode. La scelta del tipo di configurazione dipende dalla complessità dell'impianto e delle funzioni che si vogliono implementare. I dispositivi compatibili Easy-Mode sono già programmati con un set limitato di funzioni e se questa scelta da un lato limita le possibilità dall'altro semplifica notevolmente la messa in servizio dell'impianto [29, 25].
Nella configurazione System-Mode tutti gli elementi della rete vengono configurati mediante l'ausilio di un unico tool software, denominato ETS (Engineering Tool Software), che permette di definire la struttura dell'impianto. Ogni costruttore di dispositivi compatibili Konnex deve rendere disponibili gratuitamente il software per i propri dispositivi che un utente deve mantenere aggiornati e inseriti all'interno di ETS. Una volta terminata la progettazione dell'impianto il software scarica la logica di funzionamento nei dispositivi stessi rendendo l'impianto funzionante in autonomia.

Lonworks

La tecnologia LonWorks, creata da Echelon Corporation, costituisce una piattaforma completa, aperta e indipendente dal tipo di media scelto per la gestione di dispositivi connessi in rete ed è stato progettato come sistema di controllo event-triggered [25, 21]. Il protocollo di comunicazione utilizzato è il LonTalk (sviluppato dalla stessa Echelon Corporation), un protocollo aperto che offre una vasta quantità di funzionalità di controllo per l'automazione di casa, edifici e fabbriche che è stato pubblicato come standard formale denominato ANSI/EIA-709.
L'implementazione del protocollo è stata realizzata su un singolo chip da Motorola e Toshiba, detto Neuron Chip. Per realizzare un sistema LonWorks minimale sono sufficienti due Neuron Chips e un mezzo di comunicazione; questo dimostra l'elevato interesse degli sviluppatori alla semplicità di installazione e manutenzione e al basso costo di installazione. Supporta una grande varietà di mezzi trasmissivi e diverse topologie di collegamento: sono inclusi i cavi twisted-pair, la power-line, fibra ottica e anche soluzioni a radio frequenza [21].

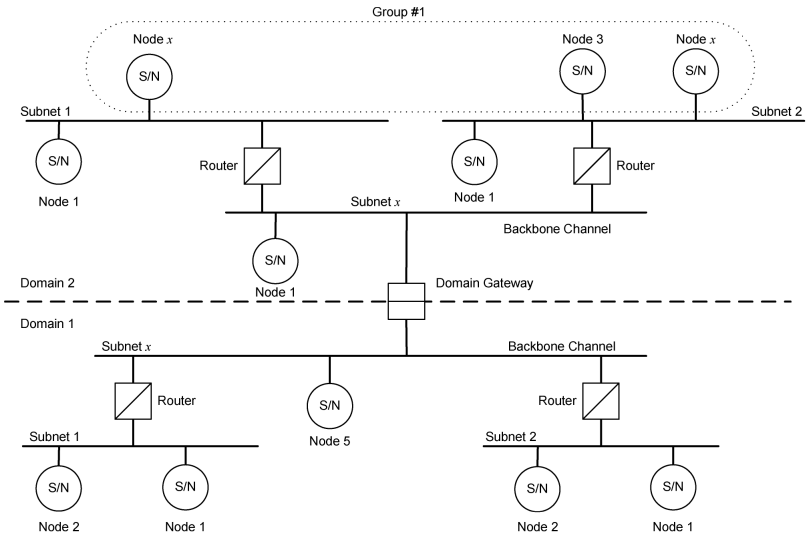


Figura 3.4: Schema di rete Lonworks [21]

Il protocollo di comunicazione LonTalk è indipendente dal mezzo trasmissivo e l'intero spazio indirizzabile è chiamato *domain*. I *domain* sono suddivisi fino a 255 subnet con un massimo di 127 nodi ognuna. Quindi è prevista la presenza di un massimo di 32.385 periferiche all'interno di un unico *domain*. Una subnet corrisponde a un singolo canale fisico sebbene la configurazione sia molto flessibile rendendo possibile collegare più canali fisici in una subnet tramite bridge o repeater ma anche permettere che più subnet possano coesistere nello stesso canale fisico. Il routing tra differenti subnet è reso possibile da opportuni nodi gateway (vedi figura 3.4). È possibile l'invio di messaggi in unicast, multicat o broadcast.

L'architettura LonWorks fornisce anche un sistema per il collegamento tra la rete LonTalk e la rete TCP/IP per la gestione delle periferiche da parte di personal computer. Ciò è garantito dal LNS (*LonWorks Network Services*), un sistema operativo che fornisce le API per l'installazione, la configurazione e il monitoraggio della rete domestica. LNS è basato su una parte server, realizzata su un apposito dispositivo in grado di implementare sia il protocollo LonTalk che il protocollo TCP/IP e che funge da gateway per l'accesso alla rete domotica dall'esterno, e da una parte client realizzata tramite un software installabile su

ogni piattaforma (PC, MAC, UNIX, embedded, ecc.).

Confronto tra i vari sistemi in Italia

bTicino MyHome

MyHome rappresenta forse il sistema più diffuso sul mercato italiano. Basato sul protocollo proprietario SCS il sistema ha dei componenti che non possono comunicare nativamente con prodotti non appartenenti al sistema MyHome. Per ovviare a questo limite nel tempo è stato introdotto il protocollo aperto OpenWebNet attraverso il quale il sistema bTicino può comunicare con altri dispositivi attraverso un gateway [28].

Figura 3.5: Logo OpenWebNet

Il catalogo di prodotti MyHome è uno dei più completi, il che rende possibile realizzare quasi tutte le funzionalità della home automation solo con prodotti MyHome. Inoltre grazie a OpenWebNet e l'attività di una community di sviluppatori indipendenti supportati da bTicino, il livello di integrabilità del sistema è costantemente aumentato nel tempo. Risulta difficile comunque superare limiti strutturali di un sistema nato inizialmente chiuso: rispetto ad altre soluzioni domotiche l'integrabilità riesta comunque bassa.

I prodotti MyHome non sono completamente programmabili dove, in realtà, la programmazione consiste nel selezionare uno tra i vari preset predefiniti di fabbrica. La programmazione reale viene effettuata a livello di supervisione con il server scenari e il webserver. L'introduzione di questi dispositivi "programmabili" ha migliorato notevolmente la flessibilità del sistema ma al prezzo di una forte centralizzazione dell'intelligenza del sistema domotico. Più le funzio-

ni richieste dall'impianto sono evolute e complesse più l'impianto dipenderà dal server scenari.

Konnex

Figura 3.6: Logo Konnex

Di filosofia completamente opposta rispetto a MyHome è il sistema Konnex [28]. È un sistema plurimarca (attualmente partecipano al consorzio più di 300 produttori). Diverse aziende adottano una lingua comune per consentire ai propri dispositivi di dialogare con quelli di altre aziende. Il protocollo di comunicazione è aperto (il che non significa che sia gratuito), per cui chiunque può sviluppare prodotti e software per il Konnex.
Il konnex è nativamente integrabile in qualsiasi altro sistema domotico. È oggi uno degli standard per la domotica più diffusi al mondo, e di sicuro è lo standard aperto più diffuso in Europa, ragion per cui esistono interfacce per consentire l'integrazione con qualsiasi altro prodotto/sistema.
L'affidabilità del Konnex è elevatissima. Anche qui parliamo di un sistema domotico che ha diversi anni di onorata carriera. La programmazione dei componenti Konnex, qualsiasi sia il produttore, viene eseguita con un tool unico che si chiama ETS. Questo consente al system integrator di programmare un impianto indipendentemente dai prodotti installati. La programmazione avviene sempre nell'ambito di parametri definiti dal produttori, ma la flessibilità è elevata, anche perché molti dispositivi integrano funzioni logiche evolute, fino a veri e propri mini-PLC.
L'espandibilità futura dell'impianto è assoluta. Un impianto può essere aumentato nelle funzioni e nelle dimensioni quasi senza limiti. Con il konnex si può realizzare un monolocale come un aeroporto. L'elevato numero di produttori e l'intercambiabilità dei prodotti tra un'azienda e l'altra, rende abbastanza sicuri della disponibilità di ricambi e di supporto in futuro.

Vimar e Gewiss

Aziende che si sono inserite nel mercato successivamente, come le italiane Vimar o Gewiss, hanno provato ad adottare una terza via, che potremo definire intermedia, tra le due: l'easy-konnex [28]. Il konnex infatti è nato in due versioni, lo standard mode, quello di cui abbiamo parlato sopra e che si programma con ETS, e l'easy mode, una versione semplificata in cui la programmazione dei dispositivi avviene attraverso una centralina dedicata. Questa soluzione dovrebbe unire i vantaggi del protocollo aperto konnex con il minor costo di un sistema proprietario come bTicino. Inoltre, non essendo necessaria la programmazione con ETS, non c'é il costo a parte del programmatore.
Entrambe le aziende citate, Gewiss e Vimar, hanno poi affiancato a questa linea di prodotti una più ristretta gamma di prodotti in Konnex "puro". L'idea è quella di poter realizzare un impianto domotico a minor costo, che possa essere poi "upgradato" in un sistema konnex completo. Sembrerebbe la quadratura del cerchio, ma l'upgrade non è del tutto scontato anche perchè al momento di realizzarlo sarà necessario riprogrammare in ETS anche i componenti inizialmente programmati in easy mode. Inoltre contemporaneamente i prezzi dei componenti Konnex standard scendono, e i sistemi easy sentono la concorrenza dei loro cugini evoluti.

Net Building Automation HomePLC.

I PLC sono stati utilizzati nei più svariati settori, tra cui quelli principali delle macchine, del terziario e nell'Industriale.
Sviluppato inizialmente per sostituire grandi quantità di relè e cablaggi, si è evoluto poi con varie tipologie di interfacce, riducendo progressivamente le dimensioni. Negli ultimi anni il concetto di PLC si è evoluto notevol-

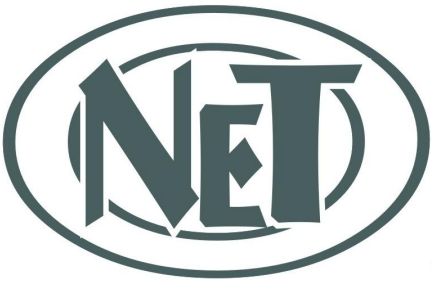


Figura 3.7: Logo NetBA

mente occupando la stragrande maggioranza di settori dell'automazione e la consueta forma che lo contraddistingueva si è in parte modificata secondo le necessità e i settori d'impiego.
L'unico settore di produzione che ancora non aveva recepito appieno l'uso di PLC era la Domotica e per questo si è pensato di sviluppare HomePLC ovvero il primo PLC Domotico con caratteristiche tecnico/costruttive specifiche per il settore Domotico.
Lo strumento utilizzato per definire la logica di funzionamento di HomePLC è LadderHOME che è un programma per lo sviluppo di Applicazioni Domotiche e non. LadderHOME è differrente dagli altri sistemi di programmazione per hardware domotici perchè utilizza nativamente linguaggi grafici conformi alla simbologia EN CEI/IEC-61131-3.
Questo significa che sarà possibile utilizzare LadderHOME con una minima esperienza di schemi elettrici in quanto utilizza la terminologia, icone e concetti familiari ad elettrotecnici e tecnici di settore: simboli grafici piuttosto che linguaggi testuali per descrivere azioni di programmazione.
I programmi sviluppati con LadderHOME sono denominati Ladder Diagram e il loro aspetto e funzioni simulano i circuiti elettrici. Tuttavia, sono analoghi alle funzioni presenti nei convenzionali linguaggi di programmazione testuali.
Ognuno dei possibili elementi della libreria di componenti rappresenta un modulo del programma. Quando l'utente seleziona ed inserisce un nuovo coponente nell'attuale layout del progetto, connettendolo per mezzo di linee al circuito, automaticamente effettua un link fra i moduli di programma correlati al progetto che sta sviluppando.
Durante la fase di compilazione il LD viene tradotto in un programma compatibile con HomePLC, adatto quindi ad essere inviato e memorizzato su questo PLC Domotico.
Nella logica a LADDER il programmatore accede direttamente a questa zona di memoria specificando, in formato IEC, l'indirizzo in cui è presente il dato da leggere o da modificare. In questo modo è lasciato completamente al programmatore verificare la correttezza di tale indirizzo che può essere desunto

da una opportuna tabella che mette in corrispondenza l'indirizzamento logico dei device con l'indirizzamento in standard IEC. Il programma, inoltre, viene eseguito in maniera sequenziale da sinistra verso destra, dall'alto verso il basso, rendendo necessaria un certa pratica per evitare comportamenti anomali nelle attuazioni in quanto potrei inizialmente attivare una uscita e, successivamente, disattivarla inavvertitamente.
L'utilizzo dei linguaggi dello standard IEC-EN 61131-3 rende sicuramente possibile l'approccio alla domotica da parte di professionisti che non hanno conoscenze relative allo sviluppo di software anche se purtroppo risulta carente rispetto al trend attuale di integrazione e interazione tra sistemi diversi e altamente connessi.
L'uscita del controllore HomePLC.Linux ha consentito al fornitore dei dispositivi di aprirsi a un pubblico diverso da quello normalmente servito. Ovviamente non stiamo parlando di utilizzatore finale del prodotto ma di una figura intermedia che ha intenzione di realizzare il proprio sistema domotico sia per ragioni personali sia nella speranza di realizzare un prodotto destinato alla commercializzazione.

3.3 IoT e Smart Home

Come abbiamo visto ci sono molti sistemi di Home Automation e standard affermati ma fino ad oggi hanno riguardato singole aziende che hanno prodotto tecnologie proprietarie e tra loro incompatibili. E questo ha inevitabilmente portato ad atteggiamenti spesso protettivi (riguardo alle proprie tecnologie) o di chiusura (rispetto ad altri sistemi) e prezzi piuttosto alti.
La nuova generazione di sistemi, spinti da una integrazione e una capacità di calcolo sempre più elevata e dalle possibilità create da una connettività sempre più estesa offre, punta sul connettere qualsiasi cosa, a "tutto il resto", nella maniera più semplice e più economica possibile.
Questi nuovi sistemi per la Smart Home spesso richiedono un dispositivo che lavori come hub sebbene alcuni siano esclusivamente software. Questi hub so-

litamente sono nella forma di piccole scatole al cui interno sono implementati diversi apparati radio che trasmettono a specifiche frequenze utilizzando vari protocolli oppure connettori per ulteriori standard in modo da connettere ulteriori dispositivi. Includono tra possibili nuovi standard: Zigbee, Z-wave, Bluetooth, Bluetooth Low Energy, WiFi e molti altri.
Gli approcci di questi nuovi sistemi si possono riassumere in due categorie. Da un lato chi realizza un tale sistema tenta di incorpare il più possibile dispositivi di altri produttori sfruttando i loro protocolli di connessione (se aperti) oppure utilizzando protocolli web ed eventuali gateway nel caso di protocolli proprietari. Dall'altro grandi nomi dell'ICT tentano di realizzare una propria infrastruttura e nuovi protocolli che altri produttori saranno obbligati ad utilizzare per integrare i loro dispositivi nella speranza di maggior visibilità garantita di grossi nomi come Apple e Google e dal fatto che già un grosso pubblico utilizza i loro prodotti.
Tra tutti i produttori Apple ha fissato i propri criteri per come la Smart Home Automation dovrebbe lavorare spesso vincolando chi desidera far parte del loro sistema. Apple HomeKit richiede che i dispositivi lavorino esclusivamente con Bluetooth o Wi-Fi tralasciando gli altri protocolli ma richiede anche forti requisiti di cifratura (per fornire un ambiente sicuro) e particolari requisiti hardware (specialmente per ottenere la sua certificazione).
HomeKit di Apple e Brillo di Google sono piattaforme nuove e l'idea di fondo è quella di acquistare un certo numero di dispositivi compatibili, aggiungerli alla propria rete e controllarli con il proprio smartphone.
Per entrambi sarà possibile anche includere dispositivi non supportati direttamente utilizzando dei bridge, che si occuperanno di tradurre tra HomeKit/Brillo e dispositivi non compatibili.
Gli Hub sono invece elementi essenziali per aziende come Wink o SmartThings di Samsung dato che il loro scopo è quello di incorpare dispositivi differenti che utilizzano standard differenti.

Apple HomeKit

Figura 3.8: Logo di Apple HomeKit

"HomeKit Securely control your home. Right from the palm of your hand." è lo slogan presente alla pagina di riferimento di Apple[1].
HomeKit è un framework che permette di mettere in comunicazione e controllare accessori per la Home Automation che supportano il protocollo HomeKit Accessory e i dispositivi iOS di Apple. Lo scopo è di promuovere un protocollo comune per i dispositivi della home automation e delle API pubbliche per configurare e comunicare con questi dispositivi.
Lo scopo di HomeKit è quello di mettere in comunicazione tutti i vari dispositivi tramite una unica interfaccia oppure per impostarli in modo che eseguano azioni assieme in certi orari o al verificarsi di particolari eventi. Senza HomeKit il controllo di questi dispositivi avverrebbe con app individuali e a lungo andare la cosa diventerebbe abbastanza scomoda.
I costruttori possono implementare HomeKit all'interno dei loro prodotti rendendoli facili da controllare (anche tramite Siri), garantendone lo scambio di dati in modo sicuro e in grado di lavorare con iPhone e iPad.
HomeKit permette ad app sviluppate da terze parti di compiere tre funzioni principali:

1. Rilevare accessori e aggiungerli a un db denominato *cross-device home configuration database*
2. Visualizzare, modificare e eseguire azioni sui dati presenti nel db
3. Comunicare con gli accessori e i servizi per eseguire comandi

L'*home configuration database* è utilizzato anche da Siri che permette di ricevere comandi vocali da parte degli utenti. Se gli utenti creano configurazioni rag-

[1] www.apple.com/ios/homekit

gruppando logicamente accessori, servizi e comandi Siri è in grado di eseguire sofisticate operazioni tramite controllo vocale.
Se non si vuole utilizzare continuamente i comandi vocali di Siri è possibile impostare gli *Accessory* per rispondere ad altri *trigger* come la propria posizione, il tempo o un altro *Accessory*. Così, nel caso della posizione, puoi impostare che quando viene aperta la porta del garage si accenda la luce all'interno ma solo per dati orari. Se si vuole accedere ad una interfaccia touch invece ci si deve ancora affidare alle app individuali di ogni singolo dispositivo e per accedere alle loro funzionalità complete.
HomeKit vede una casa come una collezione di accessori per la home automation e le informazioni vengono organizzate in modo gerarchico definendo Home, Room, Accessory, Service e Zone. Le Home possono essere opzionalmente partizionate tramite Room e Zone mentre gli Accessory vengono assegnati alle Room. Un singolo Accessory può permenttere il controllo di più Service da parte dell'utente; ad esempio il dispositivo per l'apertura della porta del garage può avere un servizio per aprire e chiudere la porta ma anche un servizio per gestire la luce del dispositivo di apertura.
Diversamente da altre piattaforme Apple non offre un dispositivo che svolge il ruolo di hub centrale ma basa il proprio ecosistema sul fatto che l'utente abbia già un dispositivo basato su iOS. Si potrebbe pensare che il proprio dispositivo iOS diventi l'hub centrale per HomeKit. Sono state create specifiche con le quali i dispositivi possono comunicare gli uni con gli altri e HomeKit è il filo conduttore che lega tutti questi dispositivi assieme[2].
Inizialmente era possibile accedere al proprio sistema di Home Automation da remoto solo se si aveva a disposizione una Apple TV (almeno di terza generazione) ma è possibile controllare gli Accessory anche con l'Apple Watch. Ora Apple offre anche un HomeKit "Cloud" e in questo caso i dispositivi Wi-Fi possono lavorare anche se il proprio smartphone non è presente. iCloud già salva oggetti come i contatti, documenti e altri dati e la stessa cosa può avvenire mantenendo un database localmente che conserva tutti i dati HomeKit. Cove questi davi

[2]http://www.pocket-lint.com/news/129922-apple-homekit-explained-is-it-available-yet-and-how-does-it-work

vengono modificati lo smartphone li sincronizza con iCloud così che, in ogni momento, tutti i propri dispositivi abbiano le stesse informazioni degli altri. Tutti i dispositivi iOS accoppiati allo stesso account iCloud vedranno gli stessi dati HomeKit e avranno la stessa abilità di controllare gli stessi dispositivi.
Riguardo la sicurezza HomeKit obbliga alla cifratura end-to-end tra gli Smart Accessory e i dispositivi iOS e ad ogni sessione di lavoro viene utilizzata una chiave diversa di cifratura. In questo modo risulta particolarmente difficile rubare informazioni, inserirsi nelle comunicazioni e prendere il controllo della propria home automation.
Realizzare un prodotto compatibile con HomeKit Apple richiede ai produttori di sottoscrivere l'MFI Program per poter accedere alle risorse dedicate a chi vuole integrare la tecnologia HomeKit per il proprio hardware. Assieme alle specifiche tecniche viene ricevuto il logo MFi e il diritto ad apporlo sulle proprie confezioni e un supporto tecnico dedicato all'hardware.
HomeKit è stato presentato inizialmente alla Worldwide Developers Conference (WWDC) nel 2014 ma con il rilascio degli aggiornamenti del sistema iOS vengono introdotte sempre nuove funzionalità che ne fanno quindi un sistema sempre in evoluzione.

Google Brillo e Weave

Nel 2014, al tempo in cui Apple presentò HomeKit, Google annunciò un programma di sviluppo per la divisione Nest. Era chiamato "Works with Nest" e forniva un set di API che i produttori potevano includere nei loro smart device in modo da poterli collegare e integrare con Nest e altri prodotti Google.
Come affermato da Todd Henderson[3] Brillo è un nuovo sistema operativo basato su Android utilizzato per lo sviluppo di sistemi embedded - in particolare per la IoT e per dispositivi low-power come anche dispositivi quali Raspberry Pi o Intel Edison.

[3] https://www.getstructure.io/blog/everything-i-learned-about-googles-brillo-and-weave-at-ubiquity-dev-summit

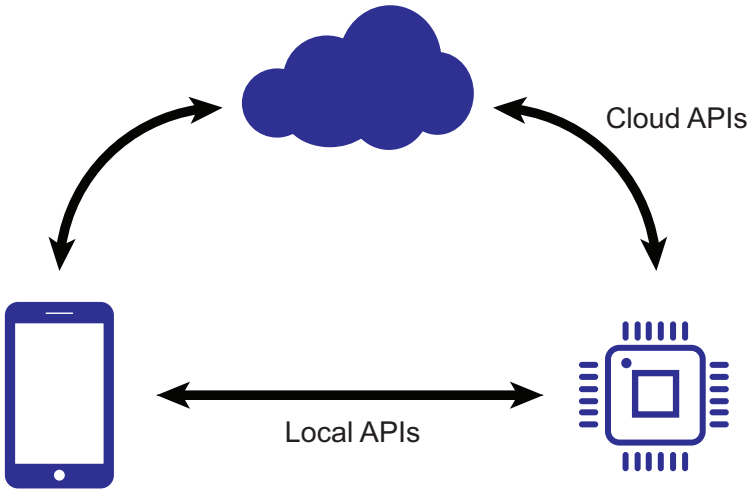


Figura 3.9: Weave API

Brillo attualmente utilizza esclusivamente c/c++ per lo sviluppo ma supporterà altri linguaggi in futuro e, anche per Google, la sicurezza viene riconosciuta come priorità per le applicazioni orientate alla IoT. Una particolarità interessante è la possibilità di ottenere aggiornamenti OTA senza che sia necessario un account Google.
Mentre Brillo è un sistema operativo Google Weave è una piattaforma di comunicazione tra dispositivi basata su JSON e comprende anche servizi cloud per provisioning, discovery, authentication, ecc.
E' da notare che il protocollo proprietario Weave di Nest è una cosa differente rispetto a quello di Google ma sicuramente questo inizialmente ha causato qualche confusione. Ad ogni modo le due teconologie sono compatibili e condividono gli stessi obiettivi: semplicemente lo fanno in modi differenti.
In un articolo su Forbes[4] viene dettagliato che Brillo è una piattaforma IoT formata da tre elementi: un Embedded OS basato su Android, una piattaforma per servizi di base e un kit di sviluppo. Brillo può essere compilato dai sorgenti per architetture ARM, Intel e MIPS. Il sistema operativo può avviarsi su dispositivi con almeno 64 MByte di storage e 32 MByte di RAM e Google sta

[4]http://www.forbes.com/sites/janakirammsv/2015/10/29/google-brillo-vs-apple-homekit-the-battleground-shifts-to-iot/

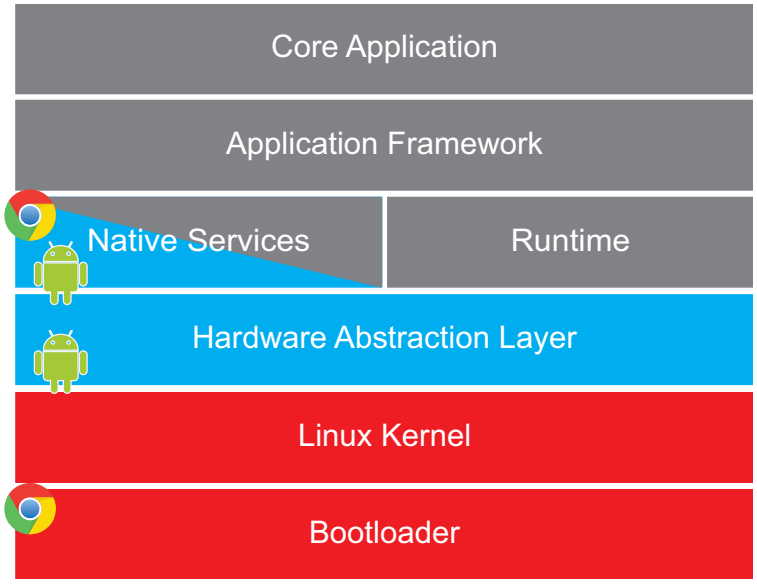


Figura 3.10: Brillo stack

lavorando con i partner per certificare le schede compatibili con Brillo: saranno schede verificate e testate per lavorare con le versioni correnti e future del sistema operativo.
I servizi principali della piattaforma includono Weave che permette ai dispositivi di connettersi in sicurezza alla rete. I dispositivi che utilizzano Weave possono scambiare dati tra loro. Il componente Metrics fa parte dei servizi di base e colleziona i dati di utilizzo del dispositivo in base ai permessi impostati dall'utente.
Ogni dispositivo che utilizza Weave è automaticamente connesso ai servizi cloud e i dispositivi che non appartengono alla stessa rete utilizzano il cloud per comunicare con tuti gli altri. Non è ancora chiaro se Weave sarà in grado di lavorare con gli standard già esistenti come BACNet, Bluetooth Low Energy, ZigBee, and Z-Wave.
Come esposto al DevFest MN 2016[5] da Dave Smith nello stack software di Brillo tutto ciò che era legato a Java nell'Android stack non c'è più. Niente più Application Framework e Core Applications e anche nei Native Services

[5]https://www.youtube.com/watch?v=ig9GKAFzDxQ

parte degli elementi sono stati rimossi: elementi necessari per supportare il Java Framework. Android non è l'unica piattaforma inclusa in Brillo: c'è anche parte del codice di Chrome OS all'interno del pacchetto Native Services in modo da fornire alcuni servizi per la connettività e altri elementi che le applicazioni possono utilizzare per lavorare in quanto non dipendono dal Java Runtime. Anche gran parte del Bootloader è stato preso da Chrome OS.
Weave è un protocollo di comunicazione tra device e non è specifico per Brillo ed è stato pensato per essere implementato su un vasto numero di device; Brillo comunque ha Weave al suo interno di default. Weave fornisce alcuni mobile sdk che permettono Android, iOS e Web Device di interagire con embedded device, fornisce apposite librerie per embedded device (di cui quelle incluse con Brillo sono una possibilità - ma non l'unica) e fornisce uno strato di servizi Cloud per connettere dispositivi remoti a dispositivi locali. Queste librerie consentono che tutto accada automaticamente e nella maniera più sicura e sono disposibili sia per le Cloud API che per le Local API.
Per quanto riguarda i dispositivi Weave è disponibile in due versioni: libweave disponibile per dispositivi MMU-Enabled, cioè processori o SoC (system on chip) in grado di supportare Linux (con presenza di Brillo o meno) e libuweave per dispositivi pensati direttamente per la Iot in cui Linux non può essere installato o per microcontroller tipo Cortex o altro. Queste librerie sono già disponibili e anche se non si vuole partecipare al programma ufficiale di supporto (ad invito) sono completamente opensource e liberamente scaricabili per fare i propri esperimenti[6].
Quando viene realizzato un dispositivo che utilizza Weave è necessario che venga definito uno schema di quello che lo stato del dispositivo rappresenta, dichiarare i comandi accettati e gli stati in cui può trovarsi un dispositivo: per i pulsanti lo stato può essere ON e OFF, oppure ON, OFF e Brightness per le lampadine. Lo schema di un dispositivo può essere utilizzato da altri dispositivi per scoprire le funzionalità del primo.
Weave automaticamente si preoccupa che lo stato di un dispositivo sia sincro-

[6]https://weave.googlesource.com/

nizzato e propagato a tuti gli altri dispositivi connessi e il concetto è che i vari dispositivi possono registrarsi gli uni con gli altri.
Brillo fornisce, oltre a Weave, altri servizi ereditati da Chrome OS come Metrics & Crash Report che fornisce informazioni sui dispositivi che sono stati posizionati in campo, quale versione del firmware hanno e quali comandi gli ha inviato l'utente. Inoltre restituisce informazioni sullo stato di salute dei dispositivi in modo da poterli riparare o controllare per primi se c'è qualcosa che non va. I dati possono essere visualizzati e analizzati nella console per capire i pattern di utilizzo. Possono anche essere analizzati i crash report per effettuare il debug dei dispositivi impiegati sul campo.
OTA è un altro servizio di Brillo che consente di preparare una nuova immagine, postarla sulla Google Developer Console e automaticamente di essere propagata a tutti i dispositivi connessi. Anche questa funzione è stata ereditata da Chrome OS e la procedura di update avviene in background senza arrestare il servizio del dispositivo. Per evitare che il dispositivo si blocchi nella fase di installazione e riavvio viene sfruttata la struttura del bootloader di Chrome OS che possiede due porzioni parallele del sistema in esecuzione nello stesso momento e mentre una sta eseguendo il suo compito la seconda è disponibile sulla quale si può applicare un aggiornamento. Quando l'update è pronto per essere mandato in esecuzione l'unica cosa che rimane da fare è il reboot del dispositivo e il tempo di disservizio del dispositivo è ridotto al minimo. Inoltre è l'utente che può scegliere quando è più opportuno applicare l'update.
Per quanto riguarda la sicurezza Google ha fatto notevoli sforzi per applicarla a tutti i livelli. Su Brillo sono presenti elementi come sandboxing e process isolation che gli sviluppatori già utilizzano quando sviluppano app per Android e quindi tutte le separazioni per i processi utilizzate per evitare che un processo possa causare problemi all'intero sistema. E' presente anche un meccanismo nel bootloader che assicura che le immagini scaricate siano firmate in modo opportuno e che non possano essere state manomesse; in questo modo se qualche problema viene rilevato il sistema semplicemente non lo carica in memoria ma riavvia la vecchia versione del firmware. E nella developer console è possibile

essere avvisati di quanto è accaduto per risolvere nel modo opportuno i vari problemi.
La sicurezza è presente anche ai livelli di comunicazione così che quando ho un dispositivo abilitato da Weave l'accesso a tale dispositivo è gestito da un account Google così che tutti i meccanismi di sicurezza forniti da Google siano presenti ed è fornito un maggior controllo su chi può vedere lo stato, chi può aggiornarlo, ecc. Inoltre tutto il traffico dati che viene inviato tra due device e tra device e cloud è criptato su TLS e anche i dati che sono presenti su un dispositivo o sul Cloud sono criptati.
Google ha presentato Brillo e Weave al Google I/O nel giugno del 2015.

Samsung SmartThings e gli altri

Mentre Apple e Google pensano di attirare produttori di oggetti per la IoT con un proprio protocollo di comunicazione e con servizi per abilitare e integrare dispositivi di terzi nel loro ecosistema Samsung e altri produttori hanno scelto una strada differente.
La scelta è stata quella di realizzare un hub centrale e connesso alla propria rete domestica in grado di sfruttare tanti protocolli di comunicazione differenti allo scopo di integrare quanti più dispositivi per la IoT possibili.
Samsung da un lato ha un grosso potenziale in quanto azienda costruttrice di molti dispositivi domestici: dalle lavatrici ai condizionatori, da prodotti per la cucina a prodotti per l'intrattenimento, ecc. SmartThings oltre a diversi dispositivi della stessa serie riesce ad integrare prodotti di altre marche quali Honeywell, Osram, D-Link, Yale e altri dedicati ai compiti più disparati.
Wink ha a catalogo un crescente numero di dispositivi di marche molto conosciute quali Philips per l'illuminazione, Nest per videosorveglianza e riscaldamento, Lutron come controller per l'illuminazione, Honeywell, Ecobee, Amazon e molti altri.
Mentre SmartThing richiede di connettere il proprio Hub con un cavo di rete al proprio router, Wink utilizza il WiFi. Wink per funzionare richiede che la connessione internet sia funzionante in quanto l'hub risulterebbe non raggiungibile

dallo smartphone: infatti entrambi si mettono in comunicazione passando tramite il Wink cloud service. Samsung invece, con la seconda versione del proprio hub, permette di effettuare alcune automazioni anche in assenza di connessione internet con il vantaggio di migliorare le performace e ridurre i tempi di latenza. Ovviamente in questo caso è precluso l'accesso da remoto.
Wink supporta Bluetooth LE, Wi-Fi, ZigBee, Z-Wave, Lutron ClearConnect e Kidde e, come indicato sul loro sito, a parte la comprensibile comodità dei primi, ClearConnect e Kidde vengono utilizzati in quanto partner del sistema Wink e anche perchè (sempre per quanto scritto sul sito) creano dei buoni prodotti. L'app per Android e iOS viene utilizzata per controllare tutti gli oggetti connessi all'hub quindi se la connessione ad internet viene interrotta l'unico modo per accedere ai dispositivi è manualmente oppure via l'app proprietaria (se presente).
Samsung supporta ZigBee e Z-Wave e in futuro potrebbe integrare anche il protocollo Bluetooth. Nel caso venga meno l'alimentazione l'hub v2 ha a disposizione anche 4 batterie del tipo AA che garantiscono un pò di autonomia (anche fino a 10 ore).

Eclipse SmartHome framework

Eclipse SmartHome non è un prodotto finito ma un framework su cui costruire soluzioni per l'utente finale. Ad ogni modo sono presenti già delle soluzioni già pronte basate su questo framework come openHAB (che ne è la release ufficiale) e altre.
Il motivo di esistere per SmartHome può essere ricercato nella necessità di una piattaforma che permetta l'integrazione di differenti sistemi, protocolli o standard e che fornisca una interfaccia uniforme per l'utente e servizi di alto livello. Il progetto è stato pensato per essere utilizzato su ogni tipo di sistema che sia in grado di eseguire uno stack OSGi: sia un server multi-core, un residential gateway o anche un Raspberry Pi.
Inizialmente fu creato openHAB (dopo una serie di tentativi) realizzato da Kai

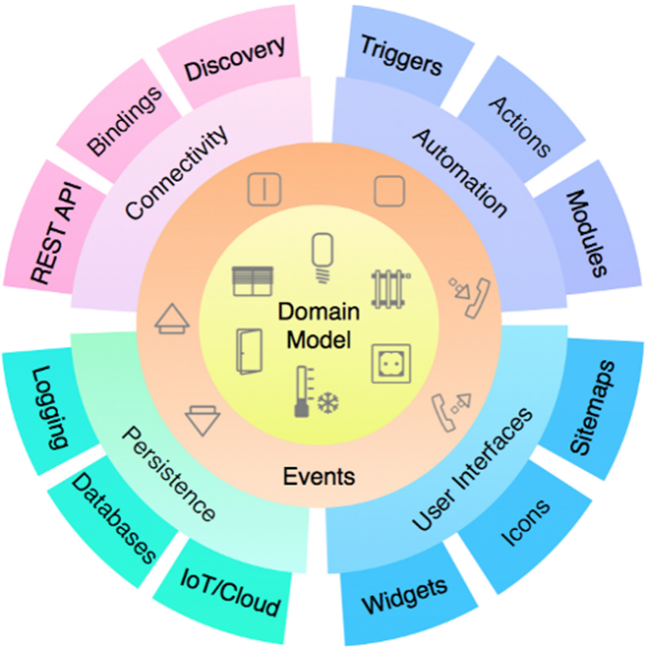


Figura 3.11: Architettura di Eclipse SmartHome

Kreuzer[7], un *software Architect and Home Automation Professional* così come indicato nel suo blog. Nel luglio del 2007 si innamora di OSGi e dopo qualche esperimento con MisterHouse (un software opensource focalizzato su tecnologia KNX) nel novembre del 2009 pensa di realizzare SmartKNX un sistema da lui descritto *"so modular and extensible that any other domostics standard can be supported as well"*. Nel febbraio del 2010 finalmente nasce openHAB. Il nome del progetto iniziale fu cambiato probabilmente per evitare violazioni sul copyright di altri software. Da allora il sistema è cresciuto e ampliato da una nutrita comunità di utenti e sviluppatori. Nel giugno del 2014 Kai decide di cedere il core Framework di openHAB alla Eclipse Foundation che darà vita al progetto Eclipse SmartHome. SmartHome è un progetto che fa parte di un ecosistema più ampio dedicato alla IoT che comprende framework e protocolli, tutti rigorosamente opensource. L'idea alla base di questo nuovo progetto è permettere a chi sia interessato di realizzare la propria soluzione domotica partendo da componenti software già disponibili e fornendo anche

[7]http://kaikreuzer.blogspot.it/

un quadro chiaro sugli aspetti legali che l'utilizzo di software di terze parti implica. Attualmente la prima versione di openHAB è arrivata alla release 1.8.x che si prevede sia l'ultima della sua serie mentre è in sviluppo la versione 2, attualmente giunta alla fase di beta release, che introduce alcune novità e migliorie.
La filosofia dietro al progetto openHAB/SmartHome nasce dalla constatazione che esistono già sul mercato molte soluzioni di home automation e dispositivi per l'Internet of Things e che tutte hanno sicuramente una loro utilità prese singolarmente. Tutti i sistemi di automazione e i dispositivi per la IoT hanno un proprio modo di installazione e configurazione e sono ben progettate per gli usi per i quali sono state pensate ma il problema è che spesso un utente vorrebbe utilizzare questi dispositivi in un modo totalmente nuovo sfruttando l'interazione con sistemi differenti che non era stato previsto inizialmente dal costruttore. Qui entra in gioco SmartHome che viene concepito come punto di integrazione permettendo ai vari sistemi di dialogare l'uno con l'altro superando i confini imposti da costruttori o protocolli.
Gli elementi principali su cui si basa il framework SmartHome[8] sono le Thing, gli Item e i Channel.
SmartHome definisce Thing come quelle entità che fisicamente aggiunte al sistema possono potenzialmente fornire nuove funzionalità. Le Thing non devono essere per forza dispositivi ma anche web service o qualsiasi altra sorgente di informazioni e funzionalità. Bridge è un tipo particolare di Thing che è necessaria quando è richiesto che l'entità permetta l'accesso ad altre Thing connesse ad essa. Un esempio tipico è un IP gateway per un sistema di automazione che internamente non è basato su IP ma su protocollo proprietario.
Diversi invece sono gli Item, entità dotate di stato che rappresentano funzionalità usate all'interno dell'applicazione (generalmente per popolare le interfacce e la logica di attuazione) e, dato il loro ruolo, possono ricevere update e generare eventi.
Tra le Thing e gli Item ci sono i Channel che rappresentano le funzionalità

[8] http://www.eclipse.org/smarthome/documentation/index.html

fornite dalle Thing e connesse agli Item. I Channel, che vengono percorsi dagli eventi del sistema di automazione, rappresentano il collante tra il livello fisico, rappresentato dalle Thing, e il livello virtuale dove risiedono gli Item.
Il cuore del framework SmartHome è formato da un event bus per la comunicazione tra i componenti. Gli eventi, che vengono veicolati all'interno dell'event bus, possono essere inviati e ricevuti in maniera asincrona e sono raggruppati in diverse categorie: Core Event, Item Event, Thing Event e altri secondo a quale entità si riferisce l'evento. Queste categorie di eventi sono tutte osservabili in modo da poter generare comportamenti e scenari anche molto complessi.

Considerazioni

Uno dei pericoli maggiori con queste nuove tecnologie è quello di rimanere legati e spesso abbracciare un particolare standard significa rimanere vincolati ad esso per sempre; è poco probabile che Apple investa nello sforzo di rendere compatibili standard di altre ditte con HomeKit - ad esempio lasciando che altri dispositivi si possano connettere tramite WiFi, citando possibili problemi di sicurezza. Questo sicuramente manterrà intatta la visione di Apple ma è probabile che manterrà alti i prezzi e limiterà le possibili scelte.
C'è anche da considerare la privacy. Apple ha un forti politiche orientate alla privacy dei propri utenti ma Google ha basato il proprio business sul data mining mentre su altri produttori non ci sono molte informazioni su questi aspetti. Inoltre molto spesso occorre necessariamente connettere il proprio ecosistema ad internet, aspetto che non libera questi nuovi prodotti da possibili attacchi, e più dispositivi sono connessi e più punti da difendere ci saranno.
Un vantaggio dei sistemi di Apple e di Google è quello riservato a chi ha intenzione di creare un nuovo dispositivo o un nuovo software per le loro piattaforme. Si tratta di documentazione e linee guida ben realizzate e molto spesso di parti intere di codice a disposizione da prendere come spunto e che consente di ridurre, e non di poco, il tempo necessario a rilasciare un nuovo prodotto.
Eclipse SmartHome è un sistema software e quindi non comprende la necessità di sottoscrivere contratti per la realizzazione di prodotti compatibili. Dato che

chiunque può realizzare un componente per integrare sistemi di altri produttori c'è sempre il rischio che l'utilizzo di parti di codice siano sotto vincoli di brevetto. Per ridurre questi rischi si è reso necessario avere una rigida gestione della proprietà intellettuale e una guida molto chiara per i chi volesse contribuire con i propri progetti. La documentazione è estesa e la generazione degli artefatti e degli scheletri per i nuovi binding è automatizzata. L'ambiente di sviluppo non poteva che essere quello di Eclipse.
Nel prossimo capitolo verrà mostrato come è stato possibile integrare più sistemi attraverso l'implementazione ufficiale di Eclipse SmartHome, chiamata openHAB, e il progetto di una libreria di integrazione per un sistema domotico commerciale. Verranno esposti i passi principali che sono stati necessari per arrivare a un prodotto utilizzato realmente in ambito domestico.

4 Caso di Studio

Come è stato scritto più volte l'Home Automation fa parte dell'ecosistema IoT e il caso di studio riguarda proprio la costruzione di un sistema di HA aperto e non proprietario. Con aperto si vuole intendere la possibilità nel futuro di poter espandere il sistema integrando nuove tecnologie e dispositivi in grado di aggiungere funzionalità, mentre con non proprietario si vuole evitare il più possibile di legarsi a protocolli o sistemi ai quali non si possa all'occorrenza apportare modifiche; proprio per non rimanere vincolati ad una particolare tecnologia.
L'utilizzo di framework e protocolli opensource ha permesso di ridurre il tempo di sviluppo di una soluzione software in grado di controllare l'intera abitazione ma soprattutto di risparmiare sui costi di acquisto in prodotti di aziende in grado di fornire le stesse funzionalità. Non tutti i sistemi domotici hanno a catalogo prodotti che lasciano all'utente la possibilità di intervenire sulla logica di controllo limitando di fatto a una semplice configurazione dei dispositivi l'operazione finale dell'installazione. Men che meno hanno la fornitura di un dispositivo programmabile con linguaggi di alto livello e in grado di personalizzare fino nel minimo dettaglio il controllo dell'intero sistema di automazione. Spesso viene fornito un dispositivo che posizionato a bordo del sistema ne espone in parte la programmabilità trasformando il protocollo proprietario di comunicazione interno in uno esterno di tipo aperto; si tratta generalmente di gateway che forniscono un accesso tramite rete ethernet e protocolli web al sistema domotico.
Come vedremo un dispositivo completamente programmabile ma privo di qualsiasi logica di funzionamento preinstallata richiede che uno sviluppatore debba

progettare e costruire tutti i meccanismi e i servizi di base in grado di rilevare il verificarsi di un particolare evento all'interno del sistema; una volta acquisito l'evento è possibile applicare la logica di controllo che poi porterà a opportune attuazioni nei dispositivi in campo.
Programmare una logica di controllo con linguaggi di alto livello, sebbene permetta una personalizzazione completa del comportamento del sistema, pone alcuni problemi nel caso si voglia modificare o aggiungere altre funzionalità che magari non si erano previste inizialmente; questi aspetti verranno affrontati successivamente integrando gli artefatti del sistema di test iniziale in un progetto più ampio che tenterà di risolverli mediante l'uso di framework opensource.
Per permettere questa integrazione vedremo come realizzare un particolare componente software che sfrutti gli elementi precedentemente realizzati e parte dei servizi resi disponibili dal framework al fine di garantire all'utente la possibilità di riprogrammare a runtime la logica di controllo e anche la possibilità di aggiungere o rimuovere nuovi dispositivi senza dover riavviare completamente il sistema.
Dato che la ristrutturazione ha riguardato l'intero appartamento si è pensato di predisporre in modo opportuno, fin dalla posa dei corrugati e delle scatole di derivazione, i vari sottosistemi: impianto elettrico, impianto multimediale, impianto di comunicazione. Si è deciso per una posa delle canalizzazioni per l'impianto elettrico in modo tale che, in caso di necessità, fosse stato possibile tornare ad una posa quasi tradizionale dei cavi tramite l'utilizzo di normalissimi relè e pulsanti.
Ad esempio un impianto di illuminazione tradizionale prevede la posa di un interruttore nella tratta del cavo che dal quadro elettrico porta al punto luce. Nel nostro caso invece il comando e il punto luce hanno tratte differenti raggiunte entrambe da una derivazione di zona o dal quadro generale. In alcuni casi sono state collegate tra loro più scatole dei pulsanti qualora avessimo passato il cavo bus tra l'una e l'altra.
Il tipo di posa delle canalizzazioni sebbene, come abbiamo visto, sufficientemente flessibile ha determinato in parte anche il tipo di hardware domotico su cui si

è scelto di orientarsi in quanto non tutti i sistemi sul mercato accettano una posa distribuita per tutto l'appartamento obbligando a centralizzare completamente la logica di controllo in un unico punto. Sebbene per chi deve intraprendere gli interventi di manutenzione possa essere visto come un vantaggio per il tecnico che deve effettuare la posa delle canalizzazioni non è sicuramente un compito semplice.

4.1 Sistema domotico HomePLC

Il sistema HomePLC fornisce due controller: uno programmabile con standard IEC-EN 61131-3 e un secondo programmabile con linguaggi di alto livello quali c/c++, php, python e Java.
Entrambi i controller condividono una parte di hardware chiamata processore domotico che fondamentalmente si comporta come un PLC classico.

HomePLC.Linux

L'HomePLC.Linux è un embedded formato da processore e scheda di espansione. Il modulo principale è realizzato dalla Engicam e si tratta di un modulo SODIMM basato sul processore i.MX53 della Freescale™ il cui core è un ARM Cortex™-A8 la cui frequenza raggiunge 1GHz.

Figura 4.1: i.Core M53

La scheda di espansione fornisce diversi connettori: alimentazione, 2 porte RS-485 per il collegamento con i device HomePLC, una coppia condivisa RS-485/RS-232 per la connessione a dispositivi esterni (ad es. Energy Meter con protocollo modbus rtu), una seriale RS-232, una presa console, una presa USB e una presa ethernet 10/100Mbit.
Questo dispositivo è corredato da diverse librerie: la libreria XComm nativa del Kernel Linux e le librerie di integrazione per i linguaggi c/c++, php, python e Java.

Lettura e scrittura degli I/O

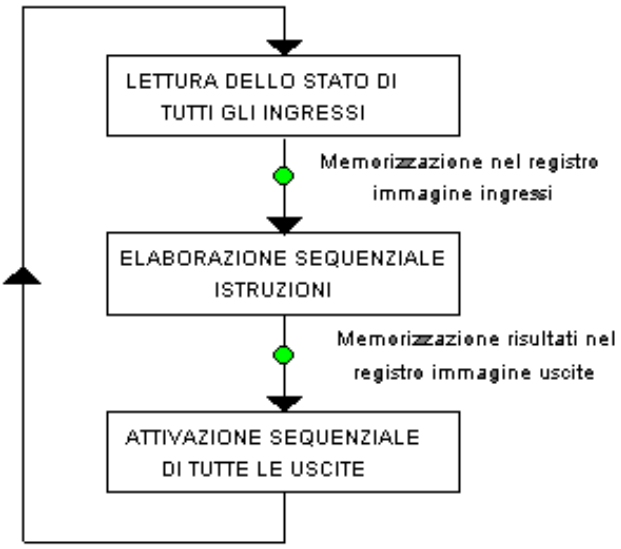


Figura 4.2: Ciclo di esecuzione del PLC

Non sono disponibili specifiche dettagliate di come il protocollo XComm++ lavori su bus RS485 però alla base del funzionamento del processore domotico c'è un ciclo continuo di operazioni così come avviene per qualunque altro dispositivo PLC classico.
Quello che cambia è semplicemente che HomePLC non ha accesso diretto ai vari I/O ma i tempi per l'accesso ai vari moduli sono regolati in base al trasferimento che avviene su bus e quindi con una logica diversa da quella industriale. Sebbene

il periodo del ciclo principale sia fissato a 100 msec i tempi con cui arrivano le informazioni al controller dipendono dal particolare device e su quale area sono mappate le sue risorse. Abbiamo quindi:

- un'area "real-time" per gli ingressi e le uscite digitali con periodo di aggiornamento di 150 msec
- le restanti area generalmente con periodo tra i 400 e i 600 msec

In questo contesto i tempi di lavoro, sebbene non adatti a un contesto di controllo numerico, sono da considerarsi normali per il settore specifico della domotica.

Caratteristiche

- Memoria Programma utente 15 KByte
- Memoria Utente 8 KByte
- Memoria Processore Domotico 8 KByte
- RTC Real Time Clock on Chip
- Capacità di elaborazione Ladder 10 MIPS
- Capacità di elaborazione Domotica 7,2 MIPS
- Ciclo di programma 100msec fisso

L'HomePLC è dotato di funzionalità Plug&Play che permette di configurare automaticamente ogni tipo di dispositivo, mappare il dispositivo nella corretta area IEC per la programmazione oltre che tabulare il modello di dispositivo secondo le sue funzioni caratteristiche per il corretto utilizzo da parte del sistema.
Il plug&play non esegue però l'indirizzamento automatico dei device per cui è richiesto che i vari dispositivi, prima di eseguire questa funzione, siano correttamente indirizzati sui vari livelli del sistema secondo la corretta mappatura di progetto, che viene normalmente stilato sulla tabella delle Risorse di Sistema.

L'indirizzamento dei device avviene su livelli:

Livello 1 Supervisione/Multimaster (indirizzi 1..999)

Livello 2 dedicato alle varie tipologie di master (indirizzi 1..46)

Livello 3 dedicato alle varie tipologie di dispositivi slave.
In base al master di livello 2 si può avere:

- caso Master I/O (indirizzi 1..46)
- caso Master DMX (indirizzi 1..512)
- caso Master DALI (indirizzi 1..128)

Ligica distribuita

Nonostante la logica di programma sia contenuta nell'HomePLC alcuni moduli prevedono la possibilità di svolgere una serie di funzioni sia in combinazione con la logica dell'HomePLC sia in autonomia: come nel caso il PLC sia in stop oppure sia caduta la comunicazione per qualche motivo.
Questa è una particolarità molto utile di questo sistema domotico in quanto ha permesso di installare il sistema e cominciare a utilizzarlo senza prima aver realizzato il sistema software di controllo.
Molte delle funzionalità di genstione luci, tapparelle, comprese temporizzazioni e altro sono già disposibili tramite configurazione dei moduli che quindi permettono di diventare produttivi molto velocemente e garantiscono uno scenario di utilizzo in sicurezza se dovesse accadere qualcosa di spiacevole al controller principale o alle connesioni bus.

4.2 Sistema domotico di test

Prima di affrontare l'integrazione del controller HomePLC nel sistema openHAB è stato necessario realizzare un componente software in grado di rilevare cambiamenti di stato che avvengono nella tabella dei registri HomePLC; queste variazioni, ad esempio, avvengono alla pressione di pulsanti, al variare delle

condizioni dei relè di uscita o quando un qualsiasi parametro, dei dispositivi connessi al sistema, subisce una variazione (il caso più semplice è una variazione di temperatura).
Come spunto per iniziare realizzare un primo sistema domotico di test ho considerato le logiche già presenti all'interno dei moduli HomePLC. Come abbiamo già avuto modo di illustrare alcuni moduli del sistema HomePLC sono dotati di micro programmazione consentendo ad essi di operare anche in caso di problemi sul bus o quando si interrompe la comunicazione con il modulo di livello superiore.
Ad esempio i moduli adibiti all'input/output digitale sono mappati nella parte iniziale della tabella dei registri di HomePLC di conseguenza alla variazione di un ingresso vedrò variare un bit in una delle word (da 16bit) nella zona che comprende i primi 400 registri. I moduli da guida din gestiscono internamente logiche passo-passo tra ingressi e uscite e almeno una logica tapparelle che si preoccupa di interfacciare due ingressi e due uscite gestendo anche il caso di interblocco fra le due uscite.

Descrizione del problema

Il problema in esame, sebbene di ridotte dimensioni, può essere ricondotto a un esempio tipico di automazione che agisce su l'impianto elettrico di una stanza di appartamento residenziale; dotata di luce elettrica (emessa da una lampadina), una finestra dotata di tapparella con avvolgibile elettrico e tre pulsanti dedicati al controllo dell'illuminazione e della movimentazione della tapparella.
L'impianto elettrico tradizionale è stato sostituito da un sistema domotico HomePLC basato su bus di campo e, per quanto riguarda lo specifico esempio, composto da:

- un modulo di controllo sul quale andrà programmato il sistema
- un modulo di espansione in grado di rilevare la pressione dei pulsanti e il controllo di relè di attuazione per il lampadario e per il motore della tapparella.

L'utilizzatore tipico di un tale sistema è il proprietario dell'abitazione che, entrato nella stanza, preme i pulsanti relativi all'accensione della luce della lampadina e all'azionamento del motore della tapparella.
L'azione di pressione del pulsante associato alla lampadina chiude e apre un contatto tramite il meccanismo basculante del pulsante e invia un segnale elettrico al contatto di ingresso del modulo di espansione che traduce in eventi logici la pressione e il rilascio del pulsante. Nell'istante in cui questi eventi vengono rilevati il modulo di espansione invia tali eventi al modulo di controllo tramite bus che memorizza questa successione di eventi in una tabella di memoria contenuta al suo interno.
La tabella di memoria del controller HomePLC è dotata di numerosi registri da 16bit. Una particolare area di questa tabella corrisponde a tutti i possibili dispositivi di espansione, dotati in ingressi e uscite digitali, che possono essere connessi al controller tramite bus. Questa area viene suddivisa ulteriormente nella parte riservata agli input e una parte riservata agli output, la distinzione è che la parte riservata agli input può essere solo letta mentre la parte adibita agli output può essere letta e scritta.
Quando l'utente premerà il pulsante a parete, tutto il meccanismo fin qui descritto, modificherà un bit della tabella di memoria impostandolo a 1 mentre quando l'utente rilascerà il pulsante lo stesso bit precedente verrà reimpostato a 0.
Il meccanismo di comando funziona in maniera opposta alla precedente descrizione modificando i valori impostati nella tabella del controller e propagando il valore logico impostato fino al modulo di espansione che agirà sul comando del relè contenuto al suo interno aprendo e chiudendo il contatto e di conseguenza abilitando o inibendo il flusso di corrente nel cavo ad esso collegato.
La ditta costruttrice del sistema domotico fornisce il controllore corredato di una libreria software di interfaccia verso la tabella dei registri dotata di istruzioni per l'inizializzazione, per la lettura e per la scrittura dei registri o parte di essi. Questa libreria, scritta in c, permette anche di utilizzare il sistema domotico tramite il linguaggio Java.

Quello che si vuole realizzare è la logica di controllo di accensione della lampadina e della movimentazione della tapparella.
Il lampadario può essere acceso o spento e la sequenza di accensione e spegnimento è descritta da una logica di tipo passo-passo (nome utilizzato anche dai relè tradizionali solitamente impiegati in questo caso) che in una situazione di riposo mantiene lo stato ma che cambia di stato ogni volta che viene premuto (e successivamente rilasciato) il relativo pulsante.
La tapparella ha due direzioni di movimentazione determinate da quale cavo del motore è percorso dalla corrente e la logica tapparella rileva gli stati attuali dei due relè connessi ai due cavi delle fasi del motore adibito alla movimentazione della tapparella. Se entrambi i relè sono disattivati la tapparella risulta ferma in quanto nei cavi delle due fasi non scorre corrente. Ognuno dei due pulsanti, e di conseguenza ognuno dei due relè, corrisponde a una delle due direzioni in cui la tapparella può essere movimentata. Quando l'utente preme il pulsante corrispondente al movimento di salita la logica tapparella attiva il relè corrispondente fino a quando non è trascorso un tempo T_{up} che disattiva automaticamente il relè di uscita. La stessa logica di controllo avviene per il secondo pulsante ma con il secondo relè, un tempo T_{down} e il movimento di discesa della tapparella. Qualora l'utente prema nuovamente un pulsante prima della scadenza dei tempi T_{up} o T_{down} la tapparella si deve arrestare.

Analisi dei requisiti

Use Case Model

In base a quanto esposto nella descrizione del problema si possono identificare due casi d'uso principali legati rispettivamente all'accensione e spegnimento di una lampadina e alla movimentazione di una tapparella motorizzata. Continuando a leggere la descrizione del problema si può notare come il sistema HomePLC consenta tramite i suoi dispositivi di percepire la pressione di un pulsante e comandare l'apertura e chiusura di un relè ma soprattutto come per-

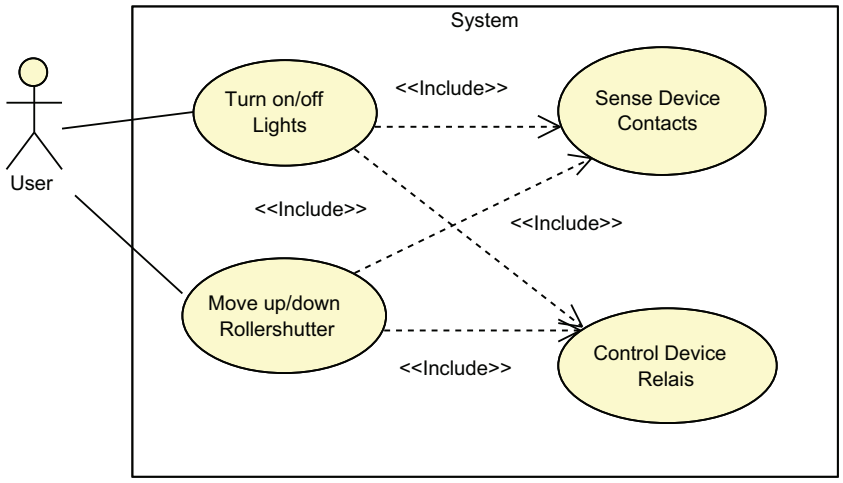


Figura 4.3: Use case model

metta di astrarre completamente da questi meccanismi e ridurre tutto a stati logici contenuti nella tabella dei registri del controller domotico.

Domain Object Model

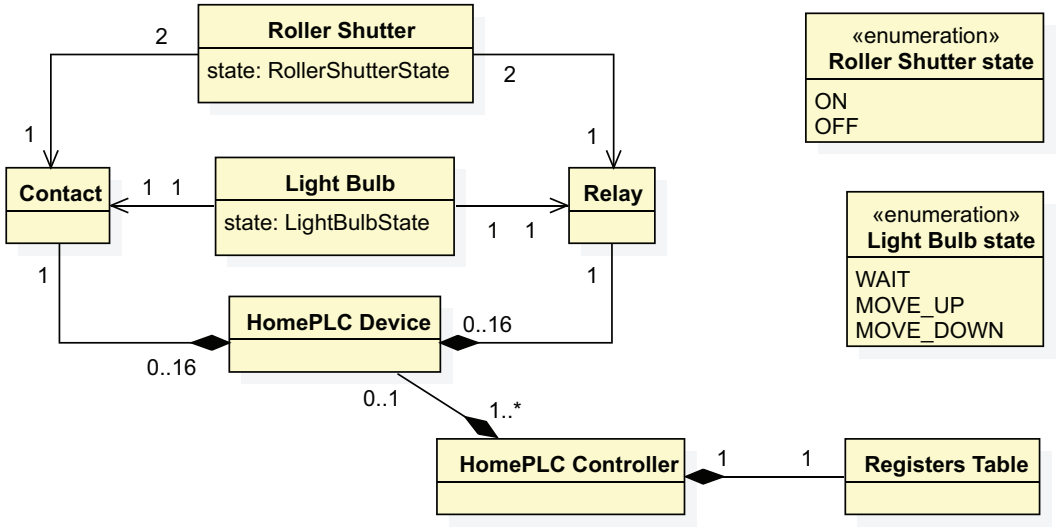


Figura 4.4: Domain Object Model

Come si può vedere dal modello il controller HomePLC può disporre di più

dispositivi esterni. Quelli che ci interessano in questo momento sono i dispositivi dedicati agli ingressi e alle uscite digitali. Questo tipo di dispositivi può essere dotato da un minimo di 4 a un massimo di 16 ingressi e uscite; sono disponibili diversi dispositivi tra cui gli 8in, gli 8out, i 4in+4out e i 16in+16out.
Dal modello risulta che per il comando di una lampadina occorre un solo pulsante mentre per la tapparella sono richiesti due pulsanti: uno per direzione di movimento. Analogamente sia la lampadina che la tapparella richiedono un numero di relè corrispondenti al numero di pulsanti appena considerati.
Nel modello sono stati indicati anche gli stati in cui possono trovarsi i due oggetti che dobbiamo comandare nel sistema.

Glossario

Tabella 4.1: Glossario

Nome	Descrizione
Light Bulb	Fonte luminosa artificiale che può essere basata su tecnologie molto diverse tra loro. Collegata all'impianto elettrico di un appartamento produce luce quando alimentata da corrente elettrica.
Roller Shutter	Tipo di porta o finestra formata da barre orizzontali unite tra loro. Nel nostro caso è dotato di motore elettrico che permette di movimentarla in una direzione o nell'altra aprendo o chiudendo il varco dove è montata.
Switch	Pulsante basculante montato in scatole a parete. Premuto chiude un contatto elettrico presente al suo interno e una volta rilasciato torna nel suo stato di riposo aprendo il contatto presente al suo interno.
Relay	Dispositivo elettrico comandabile che apre e chiude un contatto in grado di abilitare o inibire il flusso di corrente in un cavo a cui è connesso. Il comando avviene facendo scorrere corrente in una spirare che producendo un campo magnetico aziona il contatto principale.
Contact	Sensore di ingresso che rileva quando viene fatta scorrere corrente nel cavo ad esso collegato. Solitamente utilizzato congiuntamente con uno Switch.

Tabella 4.1: Glossario

Nome	Descrizione
Controller	Dispositivo del sistema HomePLC dotato di processore, RAM e flash memory in cui memorizzare il software di controllo del sistema di automazione. Abilitato a ricevere i segnali dei Device a lui collegati memorizza le informazioni in una tabella dei registri contenuta al suo interno. Tramite opportuni segnali inoltra comandi ai Device che dovranno preoccuparsi degli azionamenti.
Device	Dispositivo del sistema HomePLC connesso tramite bus al Controller e dotato di un certo numero di Contact e Relay. Funziona come sensore per i Contact e sia da sensore che attuatore con i Relay. Ogni volta che viene rilevata una variazione ai suoi ingressi o alle sue uscite inoltra opportuni segnali al Controller.
Register Table	Contenuta all'interno del controller HomePLC mantiene lo stato aggiornato di tutto il sistema domotico. La correttezza delle informazioni in essa contenute rispetto allo stato reale in campo è dovuto alla microprogrammazione contenuta all'interno dei singoli moduli che compongono il sistema HomePLC.

Scenari

Dettagliamo di seguito gli scenari associati agli use case precedenti.
Inizialmente vediamo il caso d'uso legato all'accensione e spegnimento della luce in una stanza. In questo caso si possono notare lo scenario principale e lo scenario alternativo nel caso in cui la lampadina fosse già accesa. Come si può notare sono stati inseriti i rami di inclusione verso altri use case. Si può notare come l'elevato numero di inclusioni indichi una forte dipendenza di questo use case dagli altri due.

Tabella 4.2: Turn on/off Light Bulb use case

Use Case: Turn on/off Light Bulb	
Attori	L'utente
Precondizioni	Il sistema è avviato e non ci sono errori di comunicazione tra dispositivo e controller . Il LightBult è in stato di OFF.
Scenario principale	

L'utente entra nella stanza e osserva la quantità di luce presente. Notando che la quantità non è sufficiente e la lampadina è spenta preme e rilascia il pulsante a parete. --> include(Sense Device Contacts) Il sistema, rilevato l'evento di pressione del pulsante, applica la logica passo-passo e invia i comandi per far chiudere i contatti del relè corrispondente al cavo di alimentazione della lampadina. --> include(Control Device Relais) Il dispositivo effettua le azioni corrispondenti ai comandi richiesti. Il dispositivo completata l'azione di comando aggiorna di conseguenza lo stato logico interno della lampadina.
Scenario alternativo
L'utente nota che la luce nella stanza è sufficiente e la lampadina è accesa. Preme e rilascia il pulsante a parete. --> include(Sense Device Contacts) Il sistema, rilevato l'evento di pressione del pulsante, applica la logica passo-passo inviando i comandi per far aprire i contatti del relè corrispondente al cavo di alimentazione della lampadina. --> include(Control Device Relais) Il dispositivo effettua le azioni corrispondenti ai comandi richiesti. Il dispositivo completata l'azione di comando aggiorna di conseguenza lo stato logico interno della lampadina.

Il secondo caso d'uso considerato è quello relativo alla movimentazione della tapparella. Non sono stati dettagliati tutti gli scenari alternativi per brevità ma semplicemente i due più rappresentativi di questo use case. Ciò che si è tralasciato è il caso opposto a quello considerato in cui venga alzata la tapparella e il caso in cui per arrestare la tapparella in movimento viene premuto il pulsante del verso di movimentazione opposto. In questo ultimo caso il comportamento è identico allo scenario alternativo indicato di seguito. Anche in questo caso c'è una forte dipendenza dai due use case restanti.

Tabella 4.3: Move up/down Blind use case

Use Case: Move up/down Roller Shutter	
Attori	L'utente
Precondizioni	Il sistema è avviato e non ci sono errori di comunicazione tra dispositivo e controlle. La RollerShutter è in stato di wait.
Scenario principale	

L'utente entra nella stanza e osserva la posizione della tapparella. Notando che la tapparella è alzata decide di abbassarla e preme e rilascia il pulsante a parete che comanda l'abbassamento. --> include(Sense Device Contacts) Il sistema, rilevato l'evento di pressione del pulsante di abbassamento e applicando la logica Blind invia i comandi per fare abbassare la tapparella. --> include(Control Device Relais) Il dispositivo effettua le azioni corrispondenti ai comandi richiesti. Il sistema completata l'azione di comando aggiorna di conseguenza lo stato logico della Tapparella. Subito dopo attiva un timer per il tempo Tdown impostato. Trascorso il tempo Tdown la logica Blind invia i comandi per togliere l'alimentazione al motore della tapparella. --> include(Control Device Relais) Il dispositivo effettua le azioni corrispondenti ai comandi richiesti. Il dispositivo completata l'azione di comando aggiorna di conseguenza lo stato logico della Tapparella.
Scenario alternativo
L'utente entra nella stanza e osserva la posizione della tapparella. Notando che la tapparella è alzata decide di abbassarla e preme e rilascia il pulsante a parete che comanda l'abbassamento. --> include(Sense Device Contacts) Il sistema, rilevato l'evento di pressione del pulsante di abbassamento e applicando la logica Blind invia i comandi per fare abbassare la tapparella. --> include(Control Device Relais) Il dispositivo effettua le azioni corrispondenti ai comandi richiesti. Il dispositivo completata l'azione di comando aggiorna di conseguenza lo stato logico della Tapparella. Subito dopo attiva un timer per il tempo Tdown impostato. L'utente non attende il completamento del tempo Tdown e preme e rilascia nuovamente il pulsante a parete. --> include(Sense Device Contacts) Il sistema, rilevato l'evento di pressione del pulsante di abbassamento applica la logica Blind e invia i comandi per far arrestare la tapparella. --> include(Control Device Relais) Il dispositivo azzera timer Tdown precedentemente impostato. Il dispositivo effettua le azioni corrispondenti ai comandi richiesti. Il dispositivo completata l'azione di comando aggiorna di conseguenza lo stato logico della Tapparella.

I due casi d'uso seguenti, sebbene non vengano avviati dall'interazione con l'utente, sono di fondamentale importanza per il corretto avanzamento degli scenari precedenti. Senza questi due casi d'uso le azioni dell'utente non potrebbero essere rilevate dal sistema e non sarebbe possibile effetturare gli azionamenti delle logiche appena descritte.
Come vedremo più avanti, anche se lo scenario indicato risulta molto breve, molte saranno le implicazioni che una corretta progettazione e implementazione avranno sull'intero sistema.

Tabella 4.4: Sense Device Contacts use case

Use case: Sense Device Contacts	
Attori	
Precondizioni	Il sistema è avviato e non ci sono errori di comunicazione tra dispositivo e controller.
Scenario principale	
Il sistema effettua una lettura completa della tabella dei registri utilizzando le funzioni rese disponibili dalla libreria nativa. Tutti i registri letti vengono salvati in una zona di memoria e confrontati con i valori letti al passo precedente. Nel caso venga rilevata una differenza tra il vecchio valore e il nuovo valore viene generato un evento e inviato internamente al sistema in modo che chiunque fosse interessato possa riceverlo e compiere le azioni necessarie.	

Seque anche l'ultimo caso d'uso che riguarda gli azionamenti effettuati dalla logica di controllo.

Tabella 4.5: Control Device Relais

Use case: Control Device Relais	
Attori	
Precondizioni	Il sistema è avviato e non ci sono errori di comunicazione tra dispositivo e controller.
Scenario principale	
Il sistema riceve i comandi dalle logiche di controllo e li elabora richiamando le funzioni rese disponibili dalla libreria nativa che andranno a modificare i valori nella tabella dei registri.	

Analisi del problema

Dalla descrizione degli scenari risultano molto importanti i casi d'uso relativi alla rilevazione dei segnali di ingresso e delle attuazioni in uscita. Visto l'aspetto fondamentale che rivestono si decide di analizzare per primi questi argomenti. Il sistema HomePLC fornisce una serie di strumenti software per consentire agli utilizzatori di realizzare il proprio sistema domotico utilizzando linguaggi di alto

livello. Questo si traduce nella possibilità di utilizzare librerie per l'accesso alle risorse della tabella dei registri HomePLC.
Anziché accedere mediante polling le singole risorse nella tabella dei registri risulta preferibile realizzare un componente dedicato in grado di rilevare ogni possibile variazione nelle risorse HomePLC e notificare in modo opportuno il sistema; questo per evitare che il sistema resti impegnato inutilmente nel caso nulla di importante accada nell'ambiente o nessuna richiesta venga effettuata da parte dell'utente.
Chiameremo Event Generator il componente che leggerà la tabella dei registri e in base alle variazioni rilevate confezionerà un oggetto contenete tutte le informazioni necessarie a risalire all'occorrenza che è avvenuta nel sistema.

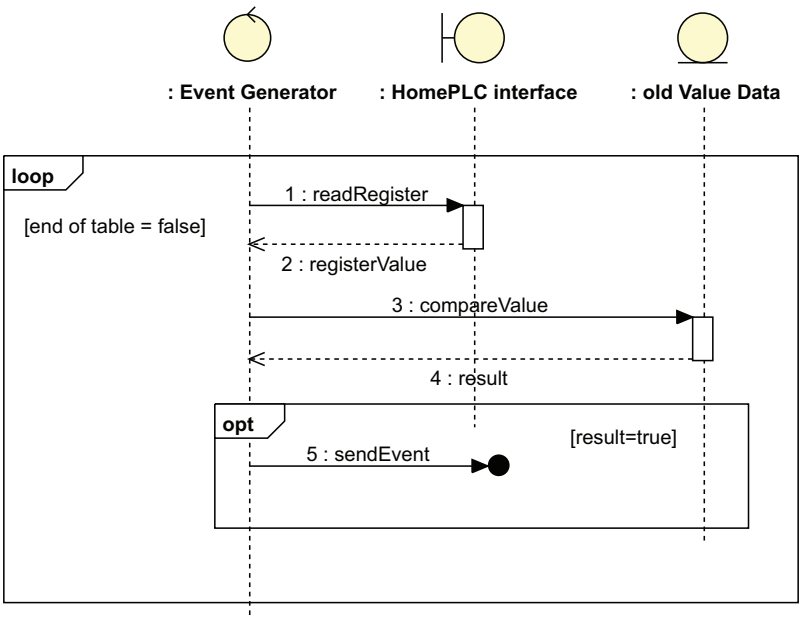


Figura 4.5: Comportameto dell'Event Generator

Dato che l'evento deve essere consegnato ad un arbitrario numero di osservatori è conveniente introdurre una ulteriore entità che permetterà di alleggerire i compiti dell'Event Generator. Introdurre un Event Dispatcher nel sistema consente all'Event Generator di concentrarsi esclusivamente sul rilevamento e la generazione degli eventi riducendo al minimo il tempo necessario a consegnare

l'evento al dispatcher. L'Event Dispatcher riceverà uno di seguito all'altro gli eventi che il generatore gli invierà accodandoli in una struttura al suo interno e gestirà in autonomia la consegna a tutti gli interessati di una copia del singolo evento prelevandolo, quando disponibile, dalla coda.

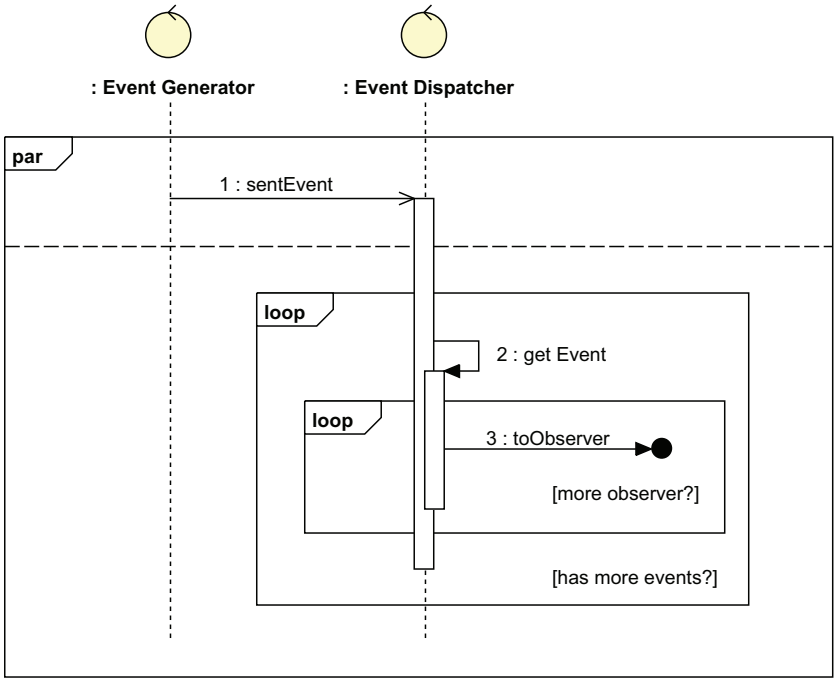


Figura 4.6: Aggiunta di Event Dispatcher

Vediamo di comprendere a chi vengono inviati questi eventi.
Abbiamo visto che, per il momento, siamo interessati esclusivamente a variazioni sul singolo ingresso e quindi stiamo parlando di un elemento che resterà in ascolto dell'evento per lui importante. Sarà un oggetto che dovrà registrarsi presso l'event dispatcher indicando la risorsa di suo interesse e una volta ricevuto l'evento potrà trattarlo nella maniera più opportuna.
Possiamo distinguere due casi. Il primo elemento, che possiamo chiamare DigitalInput, è legato ai segnali di ingresso: ricevuti, ad esempio, alla pressione dei pulsanti. In genere l'azionamento di un pulsate è un'operazione abbastanza rapida che prevede l'invio di due eventi: il primo alla pressione e il secondo al rilascio. Senza complicare troppo la gestione di un ingresso considereremo solo

il fronte di salita al momento della pressione trascurando il fronte di discesa al successivo rilascio. Nella realtà però non è detto che a un ingresso sia collegato un pulsante e soprattutto non è detto che sia interessato solo al fronte di salita del segnale; potrei voler rilevare solo il fronte di discesa oppure entrambi, senza considerare il caso in cui potrei avere un ingresso "normalmente chiuso" in cui i due fronti sono invertiti.
Il secondo elemento, chiamato DigitalOutput, è legato ai comandi in uscita che gli vengono impartiti dalle logiche di controllo. Nonostante il suo compito principale sia quello di propagare i comandi ricevuti verso la libreria nativa risulta comodo inserire al suo interno anche la logica di funzionamento del DigitalInput. In questo modo ogni volta che un relè di uscita cambia stato DigitalOutput viene notificato dall'Event Dispatcher. Nel caso delle uscite si vogliono rilevare entrambi i fronti del segnale di uscita in quanto generalmente le variazioni non sono così rapide come i segnali di ingresso e poi perchè l'utilizzo che se ne fa può essere legato alla conferma dell'avvenuto azionamento.

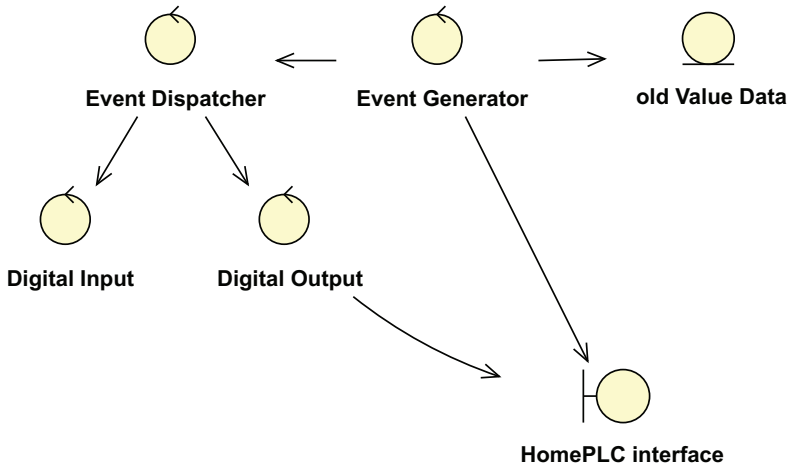


Figura 4.7: Architettura rilevamento eventi e azionamento

È giusto notare che anche in questo caso abbiamo considerato solo una piccola parte delle specifiche che può avere un DigitalOutput; l'azionamento di un DigitalOutput potrebbe essere ritardato mediante un opportuno T_{delay}, avere un tempo massimo T_{off} dopo il quale resettarsi e, in entrambi i casi, ricominciare il

conteggio se dovesse ricevere un segnale opportuno o annullarne l'azionamento. Analogamente al DigitalInput l'uscita potrebbe invertire gli stati se impostato come "normalmente chiuso".

Logica Passo-Passo

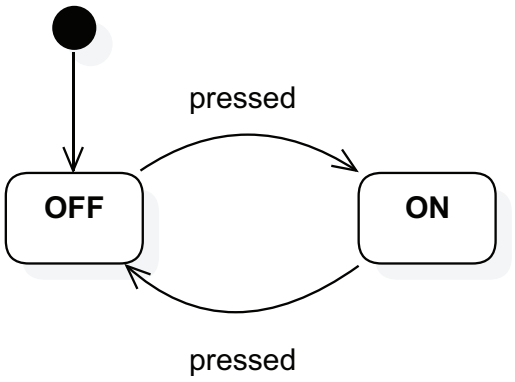


Figura 4.8: State Diagram logica passo-passo

La logica passo-passo è di per se molto semplice in quanto la macchina a stati finiti che la descrive ha due soli stati e il passaggio da uno stato all'altro avviene tramite l'unico segnale di ingresso accettato.
Rispetto alla logica Blind non sono presenti timer ma, come abbiamo visto per gli elementi DigitalInput e DigitalOutput, potrebbero esserci casi in cui venga richiesto di disabilitare l'uscita dopo un certo periodo (ad es. l'uscita temporizzata della luce scale) così come anche un ritardo di attivazione (ad es. l'attivazione di una sirena di allarme). In questo caso viene presentata solo l'architettura logica mentre per il diagramma di interazione si rimanda la quello della logica Blind che risulta un pò più complesso.

Logica Blind

Nella logica blind, leggermente più complessa della passo-passo, gli ingressi sono due e compaiono anche T_{up} e T_{down} che sono rispettivamente i tempi di salita e i tempi di discesa che indicano dopo quanto tempo va tolta l'alimentazione al

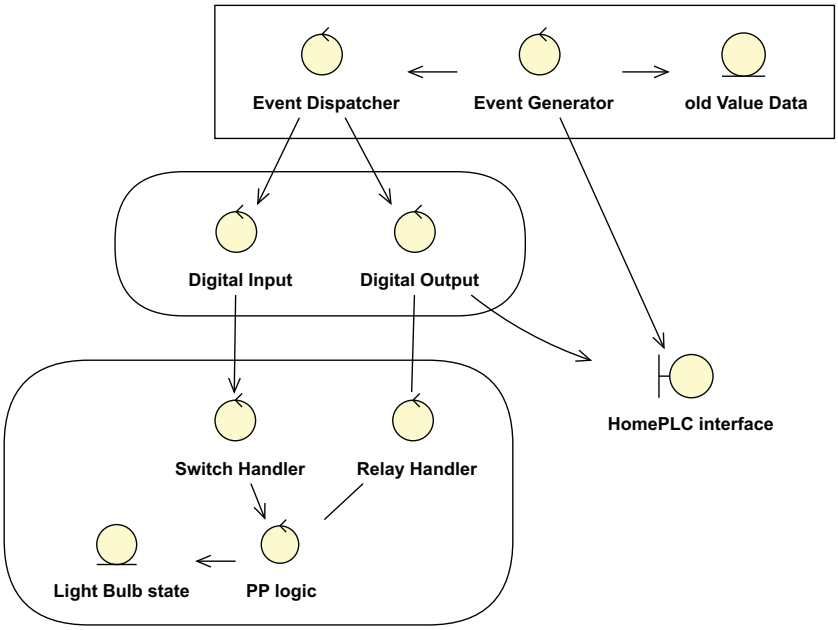


Figura 4.9: Light Bulb Analysis Model

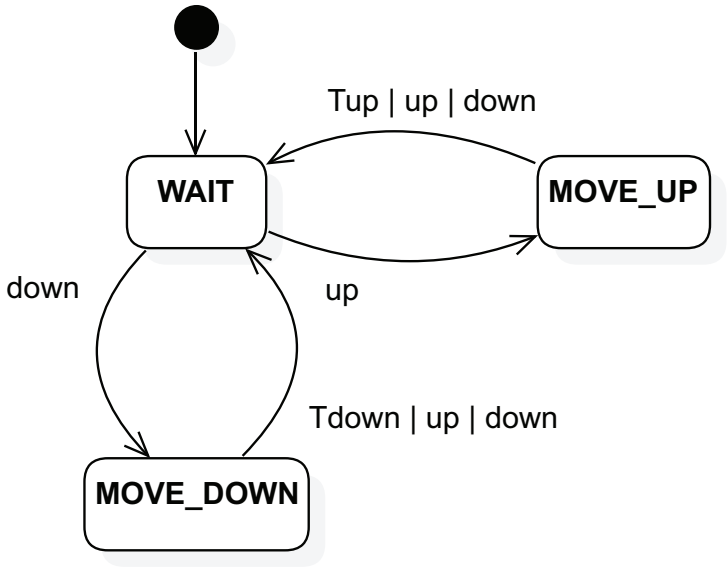


Figura 4.10: State Diagram logica Blind

motore di movimentazione. Va notato anche che qualsiasi pressione di pulsante provoca l'arresto della tapparella. Questo comportamento è necessario e viene

chiamato interblocco. Si rende necessario in quanto alimentare il motore per movimentare la tapparella in entrambe le direzioni potrebbe causare guasti al motore.
La logica Blind è una macchina a stati finiti che reagisce a una coppia di segnali in ingresso e genera una coppia di segnali in uscita. Quando un evento è presente alla coppia di ingresso relativa ai pulsanti elabora il comportamento da intraprendere e produce il segnale opportuno in uscita.
L'interazione con i DigitalInput e i DigitalOutput viene realizzata introducendo degli opportuni handler che hanno il compito di riceve i segnali ricevuti in input e inoltrare i comandi ritenuti necessari dalla logica Blind.

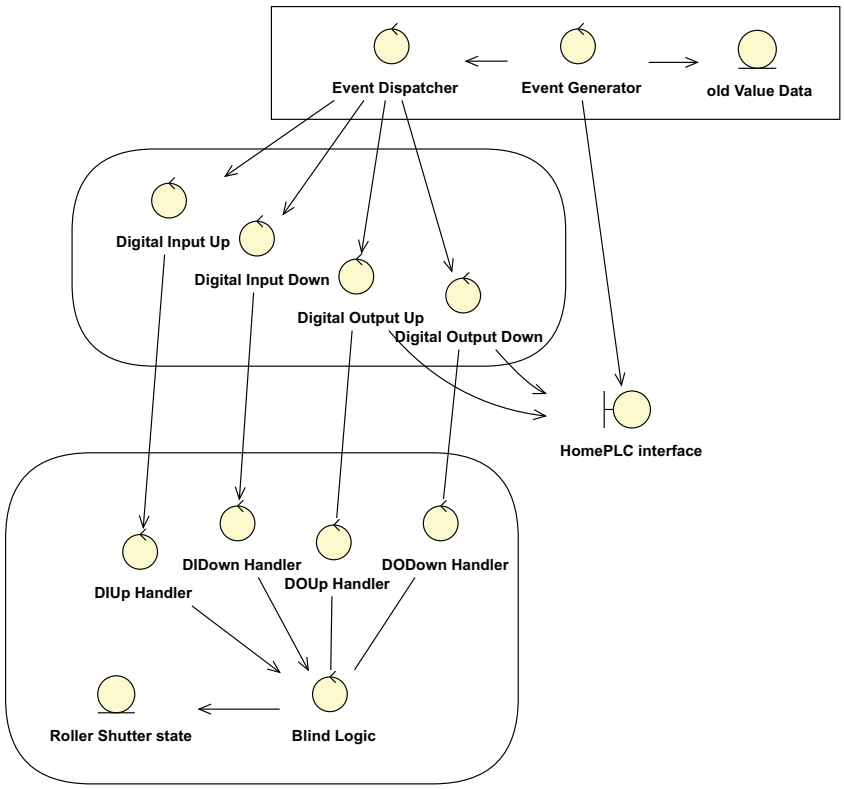


Figura 4.11: Roller Shutter Analysis Model

Vediamo di mostrare ora la dinamica dello scenario principale relativo alla movimentazione della tapparella verso il basso.

In questo scenario tra gli eventi che il sistema invia al DigitalInput viene scelto di inoltrare all'handler solo quello relativo alla pressione del pulsante trascurando il successivo rilascio. L'handler informa la logica Blind che è stato premuto il pulsante down. La logica Blind in base allo stato interno invia la richiesta all'handler relativo alla movimentazione di abilitare la chiusura del relè corrispondente per il motore della tapparella.
Quando DigitalOutput risponde alla logica di controllo tramite l'handler relativo, la logica Blind aggiorna lo stato interno in accordo al comando inviato. Nel caso l'evento sia quello relativo a una delle due movimentazioni viene attivato un timer, al termine del quale, la logica Blind (previo controllo dello stato interno) invia il segnale di aprire il relè corrispondente alla movimentazione in atto.
Questo scenario mostra come la logica Blind attenda un tempo T_{down} prima di togliere alimentazione al relè. Lo scenario alternativo, ma ce ne sarebbero altri, prevede che l'utente prema nuovamente il pulsante prima del tempo di arresto; in questo caso ci sarebbe un altro ramo parallelo al timer in cui, una volta che la logica Blind ha ricevuto l'informazione che è stato ripremuto il pulsante, viene cancellato il timer ma viene comunque inoltrata la richiesta di aprire il contatto del relè per arrestare la tapparella.

Analisi dei rischi

Analizzando l'architettura logica del sistema si può notare come la parte più critica risulta essere quella che si occupa della generazione degli eventi. Le criticità che emergono sono dovute all'alto numero di confronti che potrebbero essere richiesti se un numero elevato di variazioni avvenissero in un certo istante di tempo. Un alto numero di confronti potrebbero causare un notevole ritardo al tempo richiesto per completare la lettura di tutti i registri della tabella HomePLC obbligando a posticipare l'inizio del ciclo successivo oltre il limite del periodo che si è deciso di adottare.
Questo sicuramente si tradurrebbe nel percepire il sistema come non sufficientemente reattivo e il ritardo dalla pressione di un pulsante al corrispondente

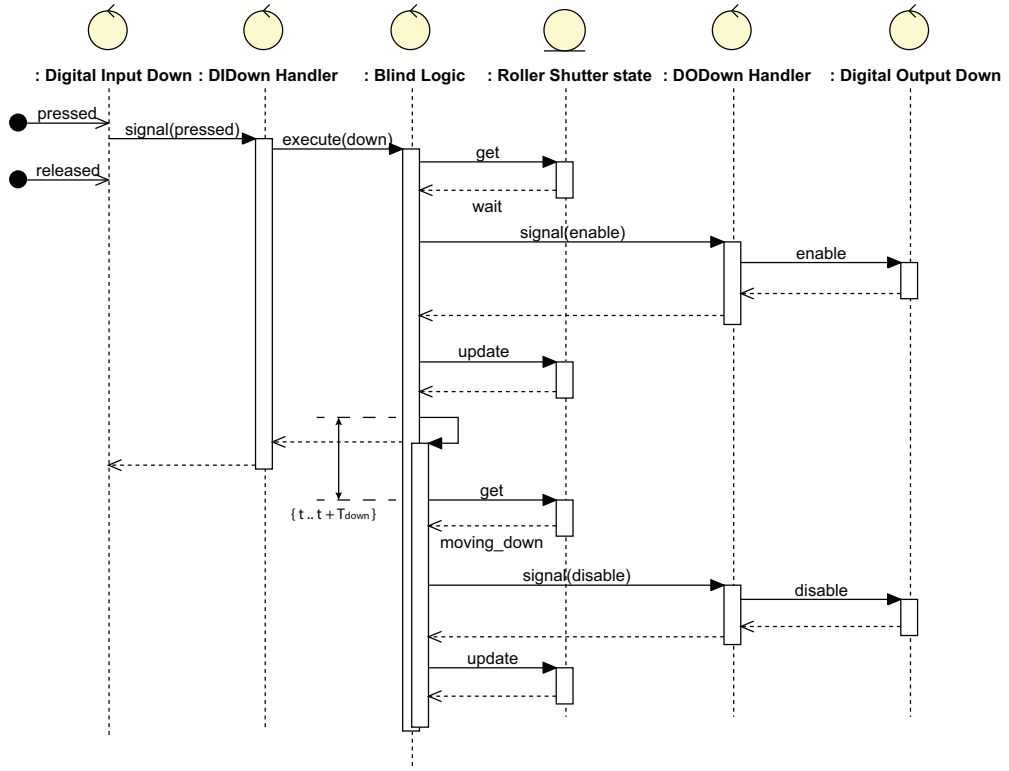


Figura 4.12: Interazione Roller Shutter

azionamento sarebbe non accettabile, risultando in una pessima sensazione da parte dell'utente.

Inoltre incrementando troppo il periodo per il ciclo di lettura, per permettere di gestire un numero molto elevato di eventi, si potrebbe generare la controindicazione di non rilevare affatto un evento nel sistema causando la perdita di comandi se non addirittura l'esecuzione di comandi completamente errati rispetto a quanto realmente richiesto dall'utente.

I DigitalInput e i DigitalOutput, come abbiamo visto, ricevono eventi dall'EventDispatcher per poi avviare le logiche di controllo alle quali sono collegati. Se la logica di controllo dovesse impiegare troppo per essere completata l'EventDispatcher rischierebbe di rimanere bloccato e non riuscire a gestire con rapidità la consegna degli eventi a tutti gli elementi che lo richiedono. Presumibilmente anche DigitalInput e DigitalOutput potrebbero richiedere un proprio flusso

di controllo e una coda di eventi dalla quale estrarre l'evento da gestire. Alla stessa maniera come è stato pensato per l'EventDispatcher e l'EventGenerator. Il problema che potrebbe sorgere in questo caso è legato all'alto numero di elementi digitali che corrisponderebbe a un alto numero di thread nel sistema. La gestione multi-thread da parte di un sistema potrebbe richiedere lo spreco di molte risorse e nuovamente subire rallentamenti inattesi. Se il dispositivo che ospita il sistema inoltre è dotato di un unico processore l'utilizzo di un elevato numero di thread contemporaneamente potrebbe far degradare le prestazioni.

Piano di collaudo

Per collaudare i componenti software che verranno realizzati contestualmente al progetto del sistema verrà sfruttato anche l'ambiente domestico in cui è installato il sistema domotico. Fortunatamente avendo già installato i componenti principali e collegati i carichi ai rispettivi relè di azionamento si potrà sfruttare l'osservazione diretta del comportamento del sistema per avere una idea dei tempi e della correttezza della logica di controllo. Un risultato più dettagliata si può ottenere sfruttando opportune istruzioni di debug che generino output verso la console del sistema HomePLC in moda da ricavare i tempi reali di azionamento e di esecuzione di alcune logiche.
La parte più importante da collaudare è indubbiamente il sistema di generazione degli eventi in quanto se non opportunamente ottimizzato può causare fastidiosi rallentamenti di tutto il sistema causando una fastidiosa sensazione di inefficienza.
Ci si aspetta che i tempi richiesti per la scansione di tutti i registri della tabella HomePLC non siano particolarmente rapidi di conseguenza sarà probabilmente necessario trovare un opportuno metodo di ottimizzazione se non addirittura prevedere, in caso di miglioramenti modesti, di impiegare parte del tempo a modificare la libreria nativa.

Piano di lavoro

Le fasi di progetto seguiranno la seguente sequenza dovuta alla priorità rilevate nella precedente analisi:

1. uso della Java Reflection per la soluzione al problema del default package. *L'introduzione di questo punto nel piano di lavoro è stato reso necessario a seguito di quanto emerso in fase progettuale e implementativa.*

2. realizzazione di un generatore di eventi basato su polling delle risorse della tabella di sistema

3. realizzazione dell'event dispatcher per la distribuzione degli eventi a più osservatori liberando il generatore di eventi dal peso della loro consegna

4. modifica della native library per rimuovere l'overhead dell'impiego della Java Reflection. *L'introduzione di questo punto nel piano di lavoro è stato reso necessario a seguito di quanto emerso in fase progettuale e implementativa.*

5. implementazione in codice nativo del generatore di eventi riducendo notevolmente il tempo impiegato per la generazione di eventi. *L'introduzione di questo punto nel piano di lavoro è stato reso necessario a seguito di quanto emerso in fase progettuale e implementativa.*

6. realizzazione componenti di gestione degli eventi e comando. *Questo punto non verrà trattato in quanto si passerà a illustrare altri argomenti per i quali sono sufficienti quanto realizzato per il generatore degli eventi e l'interfaccia verso la libreria nativa.*

Progetto del sistema

Dalla descrizione del problema risulta che il sistema HomePLC mantiene aggiornata una tabella di registri in cui sono mappati i dispositivi esterni. Per quello che riguarda i nostri casi di studio ci limiteremo a considerare gli ingressi e le uscite digitali. Per accedere a questa area di memoria viene utilizzata

una libreria scritta in linguaggio nativo ma della quale esistono diverse versioni: una per ogni linguaggio di programmazione che il progettista potrebbe utilizzare. In questa fase quindi è possibile fare i nostri ragionamenti mantenendoci sufficientemente distanti da aspetti implementativi.
Mediante la libreria nativa è possibile richiedere il valore di un particolare registro tramite una istruzione opportuna. Il valore restituito dipenderà dallo stato dei bit da cui e composto e quindi dallo stato degli ingressi a cui fanno riferimento quei bit. Un registro HomePLC è formato da 16 bit e quindi tramite un solo accesso sono in grado di leggere lo stato di 16 ingressi oppure 16 uscite.
A questo punto una possibilità è quella di leggere i registri della tabella e successivamente rilevare quali di questi registri sono variati rispetto al passo precedente.
Tra un ciclo di lettura e l'altro dobbiamo provvedere a valutare quali bit sono variati rispetto all'ultima lettura. Valutare bit per bit tutta la tabella potrebbe richiedere un tempo eccessivamente lungo quindi si decide di valutare registro per registro le possibili variazioni. Nel momento in cui avviene una variazione su un registro viene generato quello che nel sistema verrà chiamato evento.

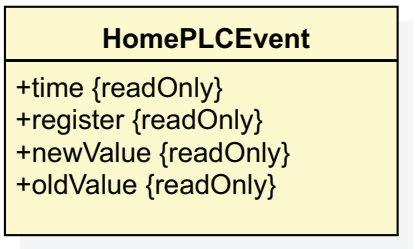


Figura 4.13: HomePLCEvent

Per il nostro scopo considereremo evento la creazione di un particolare elemento contenente diverse variabili di stato. Questo particolare oggetto è formato da quattro variabili:

- l'indicazione del tempo in cui è stata rilevata la variazione del registro
- l'indirizzo del registro

- il vecchio valore
- il nuovo valore

Inserire tale contenuto informativo in questo elemento porta il vantaggio di poter inoltrare direttamente all'osservatore i dati di cui ha bisogno per effettuare le sue operazioni.

Generatore di eventi

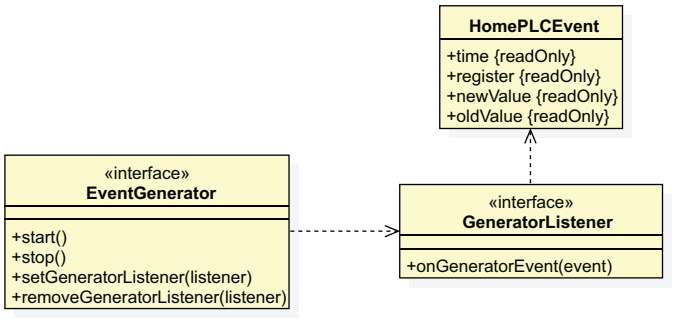


Figura 4.14: EventGenerator interfaces

Per ogni processore domotico sono definite diverse aree di memoria che rappresentano la mappatura delle risorse del sistema HomePLC: in particolare area I/O, area System Flag, area Extended Address, System area ed altre. Quando un utente preme un pulsante il modulo a cui è connesso fisicamente rileva la pressione e inoltra l'informazione, risalendo l'architettura HomePLC, fino al modulo principale che la registra come variazione in un bit di un registro corrispondente al modulo rilevatore. Quanti registri sono associati a ciascun modulo dipende dal suo numero di ingressi, dalle uscite, dai sensori connessi o dalle attuazioni per cui è predisposto; inoltre ogni modulo viene impostato con un indirizzo univoco che quindi determina l'indirizzo del registro all'interno della tabella.
Per realizzare il nostro generatore di eventi non dobbiamo far altro che rilevare i cambiamenti all'interno della tabella delle risorse in quanto siamo sicuri che rispecchia costantemente lo stato dei moduli in campo: questa certezza è data

dalla particolare architettura del sistema HomePLC e dal protocollo XComm++ utilizzato su bus RS485.

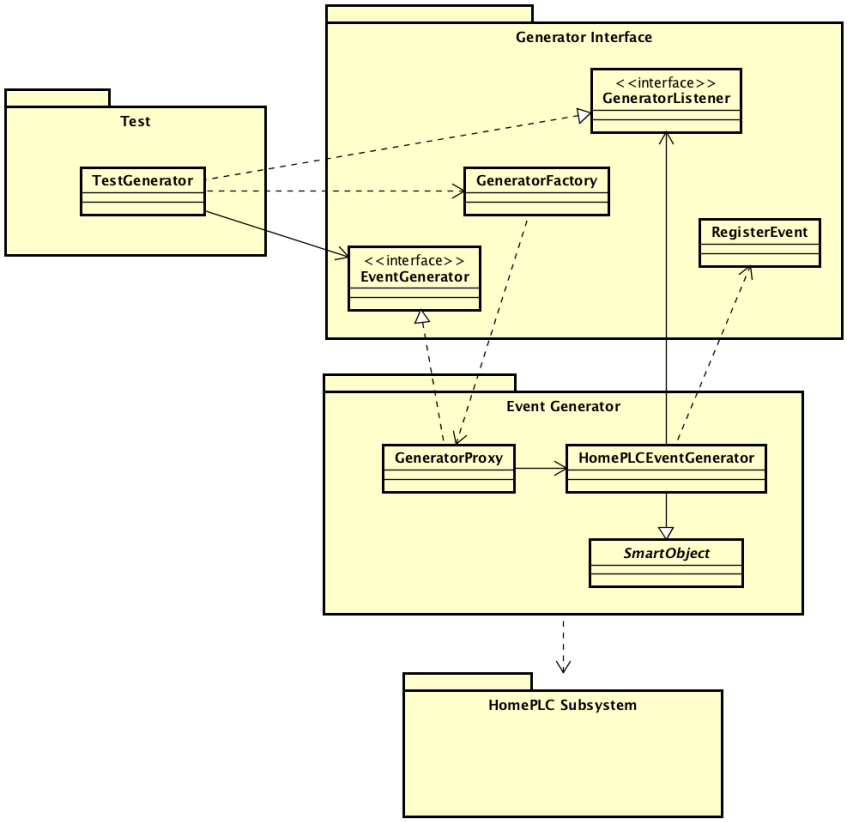


Figura 4.15: Struttura Event Generator

I registri delle zone real-time ed estesa della tabella di memoria del sistema domotico sono divisi in due sottoinsiemi: registri in ingresso e registri di uscita. I registri di ingresso possono solamente essere letti tramite le istruzioni della libreria nativa e ogni tentativo di scrittura non porta ad alcuna variazione del loro contenuto. I registri di uscita, oltre ad essere letti, possono anche essere modificati generando di conseguenza opportuni azionamenti in campo.
L'unico modo per poter accedere al contenuto della tabella dei registri è tramite le istruzioni della libreria nativa e di conseguenza occorre risolvere il problema di accesso alla libreria nativa da un package qualunque. *Si rimanda alla sezione implementativa per la soluzione a questo problema.*

Ingressi	Funzioni	Note
Registro su indirizzo di nodo	Misura di temperatura T1	Sonda connettore verde
Registro esteso 1	Misura di temperatura T2	Sonda connettore nero
Registro esteso 2	Misura di umidità RH%	(1/10)
Registro esteso 3 High Byte	Funzionalità evolute	Bit 15 = Valore byte low in negativo
		Bit 14 = Blocco per raggiungimento soglia punto di rugiada anticipato per offset temperatura (Temp. - offset)
		Bit 13 = Avviso prossimo raggiungimento blocco (Temp. - (offset + 1 grado))
		Bit 12 = Guasto sonda umidità
		Bit 11 = Guasto sonda Temp. 1 (usata per il blocco)
		Bit 10 = Guasto sonda Temp. 2
Registro esteso 3 Low Byte	Punto di rugiada	

Tabella 4.6: Registri Ragnetto Temperatura e Umidità

Una volta risolto il problema del default package possiamo finalmente progettare un generatore di eventi basato sul componente di interfaccia con la libreria nativa che è stato appena realizzato.
Utilizzando l'attuale componente di interfaccia possiamo leggere solamente un registro alla volta quindi l'unica possibilità è quella di realizzare un opportuno loop per leggere tutta una lista di registri e inoltrare l'evento qualora sia avvenuta una variazione nel registro corrente.
Resa da considerare come consegnare gli eventi generati e il Pattern Observer pare la scelta naturale dato che in questo momento non vengono fatte considerazioni legate ad una possibile natura distribuita del sistema e nemmeno rispetto alla possibilità che l'EventDispatcher possa entrare o uscire dal sistema dinamicamente.

Event Dispatcher

Come abbiamo visto, l'introduzione di un nuovo componente, si rende necessario per permettere al generatore di eventi di rimanere libero da ritardi dovuti alla consegna dell'evento a un numero molto elevato di osservatori.

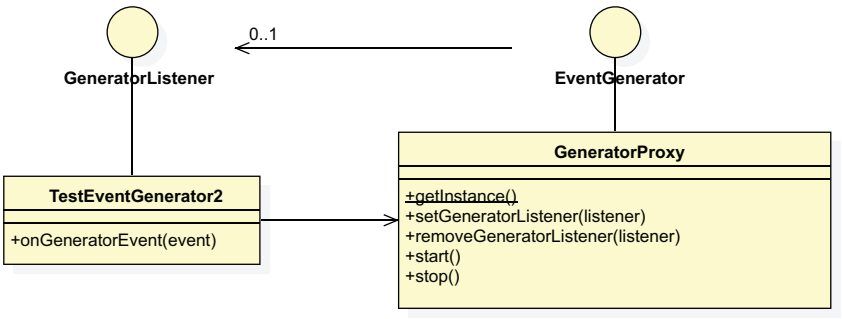


Figura 4.16: Event Generator Observer Pattern

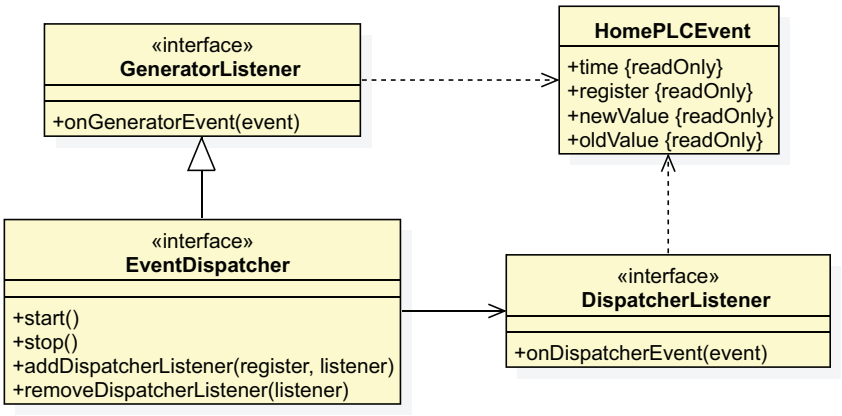


Figura 4.17: Dispatcher interface

Un passo verso una migliore razionalizzazione delle risorse è quella di aggiungere il componente Event Dispatcher che risulta essere observer rispetto al generatore di eventi ma source rispetto alla totalità degli observers: che si registreranno (o verranno registrati) presso il dispatcher specificando anche a quale registro sono interessati. L'indicazione del registro è utile al dispatcher per ridurre il traffico generato dalla consegna degli eventi; in questo modo anzichè realizzare una comunicazione broadcast dove tutti i ricevitori indistintamente riceveranno una copia dell'evento si preferirà realizzare una comunicazione multicast dove l'invio dell'evento avverrà a insiemi di observer dipendenti dal particolare registro in cui è avvenuto un evento.
L'event dispatcher è basato sul Producer-Consumer design pattern ed è in grado

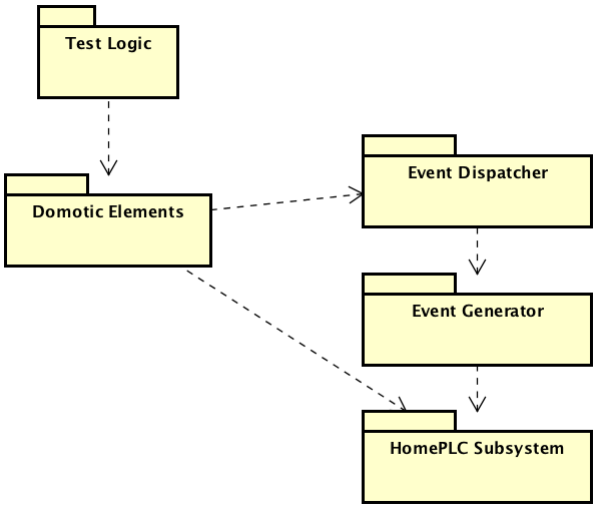


Figura 4.18: Vista dei packages

di semplificare lo sviluppo in quanto permette di rimuovere le dipendenze tra le classi di producer e consumer e inoltre disaccoppia le attività che producono o elaborano le informazioni con una rapidità variabile o differente.

Implementazione

In base a quanto analizzato risulta che la libreria di interfaccia fornita assieme al controller HomePLC può essere acceduta esclusivamente dal default package di Java e questo pone diverse limitazioni al suo utilizzo. Inoltre non presenta di suo un meccanismo in grado di generare eventi al variare dei valori dei registri della tabella di sistema.
Prima di poter realizzare le logiche di più alto livello associate agli use cases del nostro caso di studio siamo costretti a risolvere questi limiti implementativi; dapprima sfruttando il meccanismo Java della Reflection e successivamente modificando in maniera più consistente la libreria nativa di interfaccia aggiungendo parte del codice per la generazione degli eventi.
In particolare uso della Java Reflection per la soluzione al problema del default package è una delle priorità senza la quale il progetto non può proseguire e quindi richiede una variazione del piano di lavoro per includere questo argomento.

Il Default Package

Il dispositivo viene fornito con alcuni esempi di applicazione. Nel caso di linguaggio Java la struttura di un programma è la seguente:

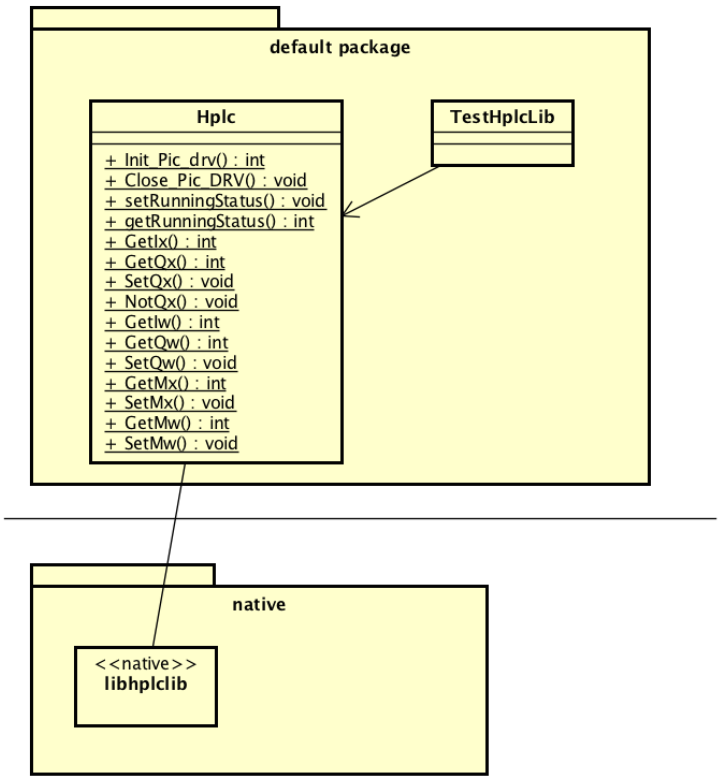


Figura 4.19: Esempio di programma Java e libreria nativa originale.

Come è possibile vedere la classe java di interfaccia con la libreria nativa è formata da tutti metodi statici e pubblici in modo da consentire di utilizzarne i metodi senza bisogno di istanziare la classe. Per caricare a runtime la libreria nativa è stato utilizzato uno static initializer:
Il difetto di questo approccio è che si viene obbligati a realizzare l'intero programma che andrà a gestire la logica di controllo nell'unico package utilizzabile: il default package. Il problema nasce perchè, sebbena la classe Hplc sia pubblica (e così anche i suoi metodi statici), tutte le ulteriori classi che si inseriranno a progetto dovranno necessariamente appartenere al default package. Pena la

Algoritmo 4.1 static initializer

```
static {
  System.out.println("Loading JNI lib");
  System.loadLibrary("homeplclib");
  System.out.println("Done...");
}
```

non visibilità della classe Hplc. Questo limite è dovuto direttamente alle Java language specifications che testualmente asseriscono «It is a compile time error to import a type from the unnamed package».
Il motivo di questo approccio probabilmente è da ricercarsi nella mentalità maturata nell'utilizzo del linguaggio grafico di programmazione chiamato Ladder Diagram (standard internazionale IEC 61131-3) impiegato da tutti gli installatori del sistema HomePLC. Il linguaggio Ladder, come abbiamo visto, utilizza un sistema grafico per definire un programma di controllo sequenziale e, sebbene siano presenti qualche forme di controllo di flusso, sicuramente non ha quella espressività che ha un linguaggio di programmazione di alto livello come Java. L'idea di chi ha rilasciato questi esempi di programmazione era sicuramente quella di definire un unico loop principale dove, una volta rilevate determinate condizioni, venivano intraprese azioni più o meno elaborate a seconda dello scopo che si voleva ottenere. L'approccio è molto simile quindi a quello, rule based, utilizzato anche dal linguaggio Ladder.
L'idea, fin da subito, è stata quella di risolvere questo vincolo iniziale per consentire di realizzare sistemi software dotati di una struttura più complessa. Gli approcci utilizzabili erano la Java Reflection e la riscrittura della libreria nativa. La riscrittura della libreria nativa inizialmente è stata scartata in quanto la difficoltà non era tanto nel tempo dedicato a ripassare un linguaggio quale il c/c++ ma la necessità di predisporre tutto l'ambiente di sviluppo adatto alla cross compilazione di software per processori ARM quali quelli installati nell'embedded in nostro possesso.

Utilizzo di Java Reflection

Le Classi appartenenti al Default Package non possono essere importate da un Package differente.
Sebbene prima della J2SE 1.4 si potevano importare tali classi grazie a un particolare import syntax ora non è più permesso: pena un compile-time error. Visto che non è possibile accedere direttamente alle classi del Default Package, se non essendo all'interno dello stesso, è stato possibile utilizzare un metodo alternativo sfruttando le Reflections API.

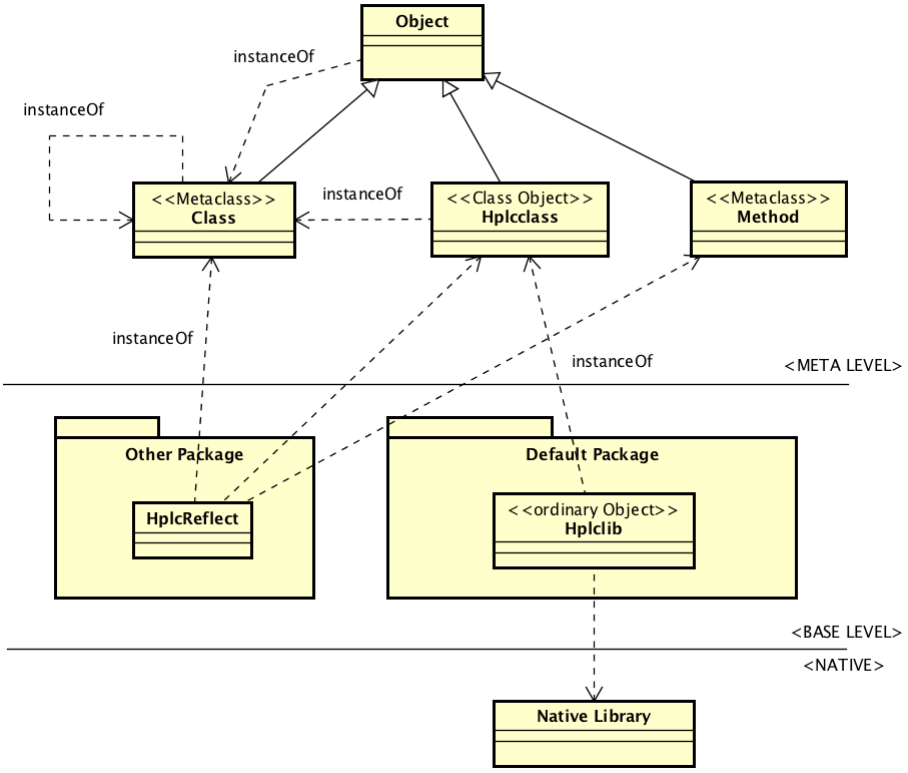


Figura 4.20: Diagramma con Meta Livello nel caso di Reflection

Il trucco, se così si può definire, è quello di caricare dinamicamente a runtime la Classe contenente il riferimento ai metodi nativi. Il metodo Class.forName(String className) permette di inizializzare la classe className quindi può essere utilizzato per caricare a runtime la classe Hplc purchè presente nel classpath. Una

volta ottenuto il Class Object i metodi resi disponibili permettono di estrarre Method Object, Field Object e altro.
Dall'immagine, nonostante la notazione UML non proprio standard, possiamo vedere come l'utilizzo di oggetti del meta livello (livello ottenuto grazie all'introspezione fornita dalla Java Reflection) riescano ad accedere alla Classe di interfaccia con il codice nativo.
Di conseguenza le Reflection API e la Dynaminc ClassLoading consentono di risolvere alcune limitazioni dovute alla realizzazione dell'interfaccia nativa per l'accesso alle risorse domotiche.

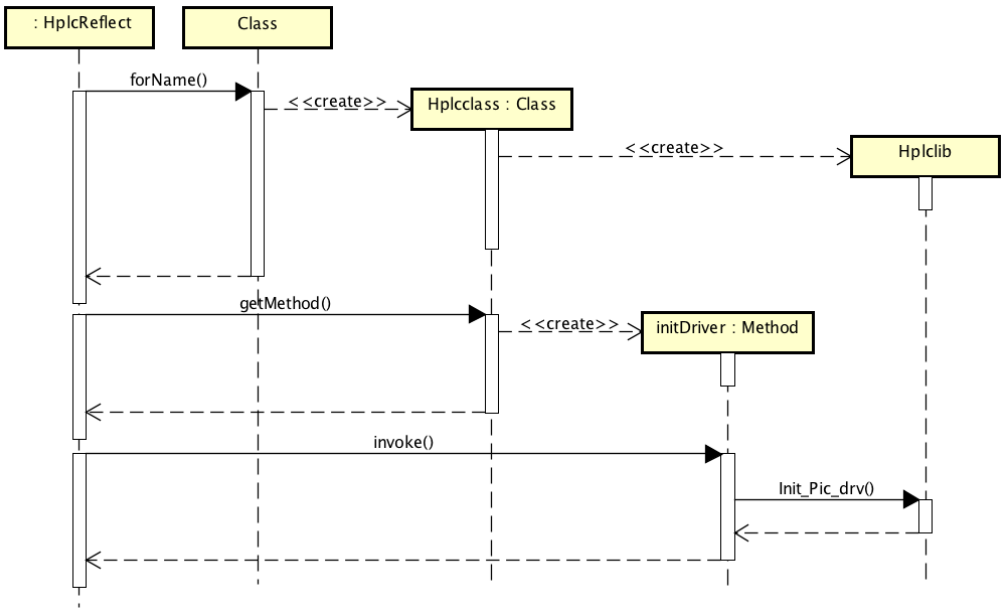


Figura 4.21: Utilizzo della Reflection.

Dalla figura si può vedere il meccanismo utilizzato per chiamare un metodo della classe Hplc: dopo una parte iniziale in cui vengono creati gli oggeti dal framework Reflection ogni successiva richiesta viene soddisfatta chiamando il metodo invoke() dell'oggetto rappresentativo associato al rispettivo metodo della classe Hplc.

Un componente di interfaccia verso la libreria nativa

Un Subsystem è una risorsa riutilizzabile che altri componenti all'interno del nostro sistema può utilizzare. Gli altri componenti, comunque, sono legati a una interfaccia ben precisa che il Subsystem espone in modo che sia possibile alterarne l'implementazione senza dover modificare tutti i componenti che ne dipendono.

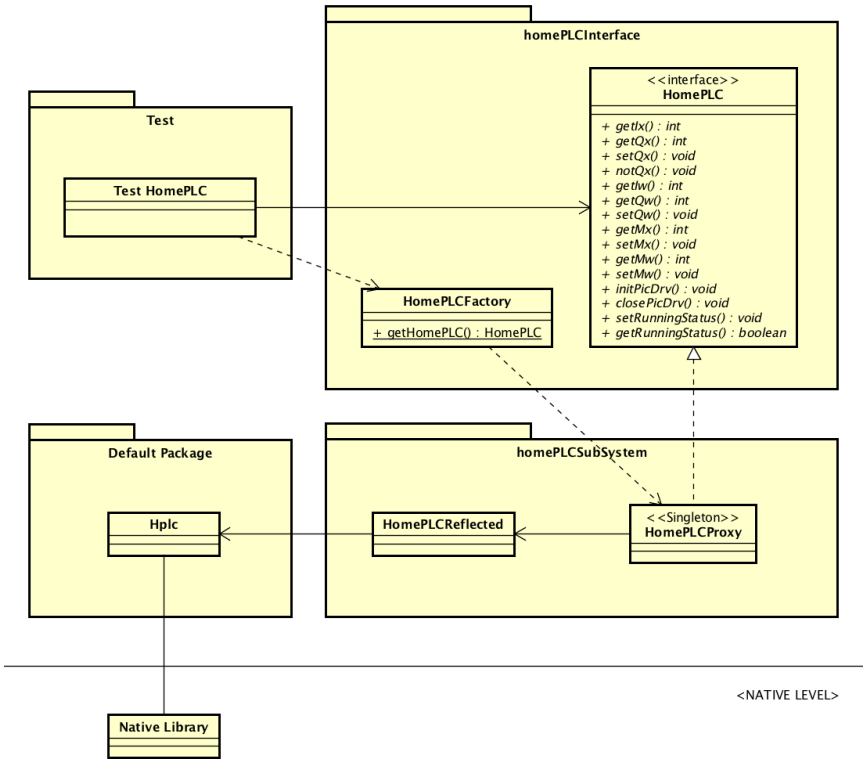


Figura 4.22: Relazioni tra packages e subsystem.

L'interfaccia del subsystem ha lo scopo di servire come contratto tra il subsystem e il mondo esterno. Qualsiasi interazione con il subsystem deve avvenire attraverso questa interfaccia. Ovviamente la Factory Class deve poter accedere ad almeno un elemento interno, che poi altro non è che la classe Proxy, implementazione dell'interfaccia.

Un tipo di subsystem così concepito ha delle forti implicazioni sulle dipendenze che si creano: mentre non ci sono particolari vincoli sulle dipendenze in uscita le restrizioni sulle dipendenze in entrata consentono di sostituire all'occorrenza una implementazione con un'altra e garantiscono un'alta riusabilità del codice.

Event Generator

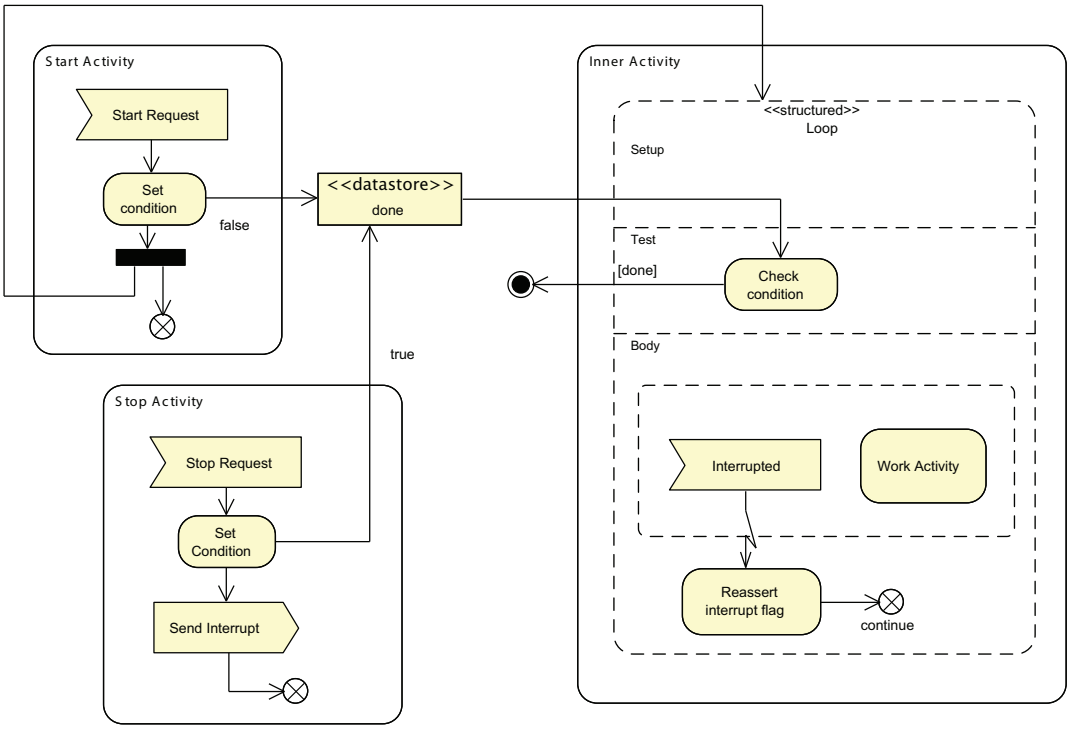


Figura 4.23: Smart Object Activity Diagram

L'elemento esteso dalla classe EventGenerator è uno SmartObject: un oggetto dotato di proprio flusso di controllo che può essere attivato e terminato a piacere. Dalla versione JDK1.2 il metodo stop() della classe Thread è stato deprecato in quanto terminava l'esecuzione del Thread in maniera molto secca e questo non dava tempo di eseguire procedure di clean-up causando possibili corruzioni di dati. Inoltre l'arresto repentino del Thread era seguito anche dal rilascio di tutti i lock eventualmente acquisiti lasciando eventuali dati in alcuni oggetti in uno stato inconsistente.

Algoritmo 4.2 SmartObject code

```
public abstract class SmartObject implements Runnable {
  private Thread internalThread;
  private volatile boolean stopRequested;

  protected SmartObject() {
    stopRequested = false;
  }

  public final void run() {
    while (!stopRequested) {
      try {
        runWork();
      } catch (InterruptedException x) {
        // Reassert interrupt so that remaining code
        // sees that an interrupt has been requested.
        Thread.currentThread().interrupt();
        // Reevaluate while condition --probably
        // false now.
        continue;
      }
    }
  }

  protected abstract void runWork() throws InterruptedException;

  public void startRequest() {
    stopRequested = false;
    if (internalThread == null) {
      internalThread = new Thread(this);
      internalThread.start();
    }
  }

  public void stopRequest() {
    stopRequested = true;
    if (internalThread != null) {
      internalThread.interrupt();
      internalThread = null;
    }
  }

  protected boolean isAlive() {
    return internalThread != null ? internalThread.isAlive() : false;
  }
}
```

Un metodo per arrestare un thread è quello di utilizzare una variable booleana come indicatore del fatto che il thread debba continuare il suo lavoro oppure se debba terminare. Quindi anzichè continuare all'infinito il metodo run() esamina lo stato della variabile ad ogni loop ed esce quando il flag viene impostato. Questo meccanismo è un semplice sistema di sincronizzazione tra due thread ed è necessario che il thread che deve essere arrestato veda un valore aggiornato

per la variabile flag: evenienza non certa dato che Java permette ai thread di mantenere i valori delle variabili in una memoria locale e inoltre può avvenere anche nel caso di macchine con più processori. Per essere sicuri di questo una possibile soluzione è di dichiarare volatile la variabile booleana assicurando al thread di leggere il dato sempre aggiornato.

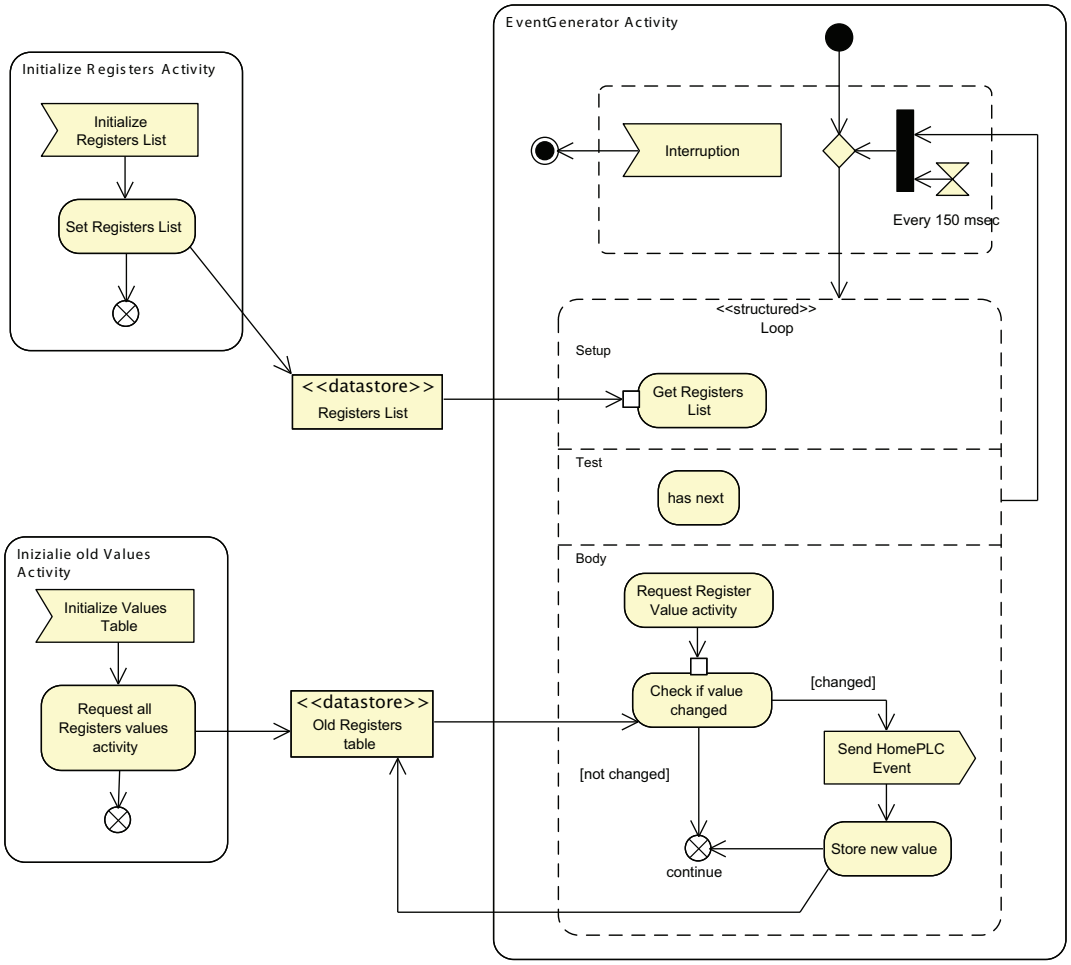


Figura 4.24: Event Generator Loop

Il precedente metodo ha il leggero difetto di poter subire ritardi prima che lo stato del flag venga nuovamente valutato e nel caso il task eseguito dal thread sia molto lungo può portare a tempi di attesa troppo lunghi. Per terminare rapidamente l'esecuzione del task quello che si può utilizzare è il metodo in-

terrupt() della classe Thread. Il metodo interrupt() ha due effetti: il primo è quello per cui i metodi bloccanti tipo sleep() e wait() generano una InterruptedException e il secondo è quello di impostare un flag all'interno dell'oggetto che indica che il thread è stato interrotto. Tale flag può essere interrogato con il metodo isInterrupted().
E' possibile quindi sostiture il controllo tramite variabile volatile con una chiamata al metodo interrupt() e un controllo tramite il metodo isInterrupted(). Ancora però potremmo subire ritardi in quanto se il medoto interrupt() viene richiamato subito dopo il controllo tramite isInterrupted() il medoto bloccante non si arresta e occorre attendere fino al termine del loop corrente e a una nuova verifica. In pratica un caso di race condition.
Si può però unire i due metodi precedenti in modo da combinare i privilegi derivati da entrambi: effettuo un controllo su una variabile booleana di tipo volatile per terminare il loop quando richiesto mentre controllo se viene generata una InterruptedException per rilevare il prima possibile l'interruzione e arrestare l'esecuzione del task. E' buona norma riasserire l'interrupt flag subito dopo aver rilevato l'eccezione.

Event Dispatcher

Lo strumento in grado di agevolare la realizzazione di questo pattern è sicuramente la Blocking Queue. Il produttore inserisce elementi nella coda appena risultano disponibili mentre il consumatore estrae elementi dalla coda quando è pronto ad elaborare i dati.
Come per il generatore di eventi la classe EventDispatcher estende la classe SmartObject quindi eredita i metodi startRequest() e stopRequest() che avviano e arrestano il dispatcher.

Refactoring della libreria nativa

Come abbiamo potuto vedere l'utilizzo di Java per realizzare il loop generatore di eventi e del framework Reflection richiede un tempo abbastanza elevato per elaborare tutto quello che è richiesto. Inoltre, inevitabilmente, all'aumentare del

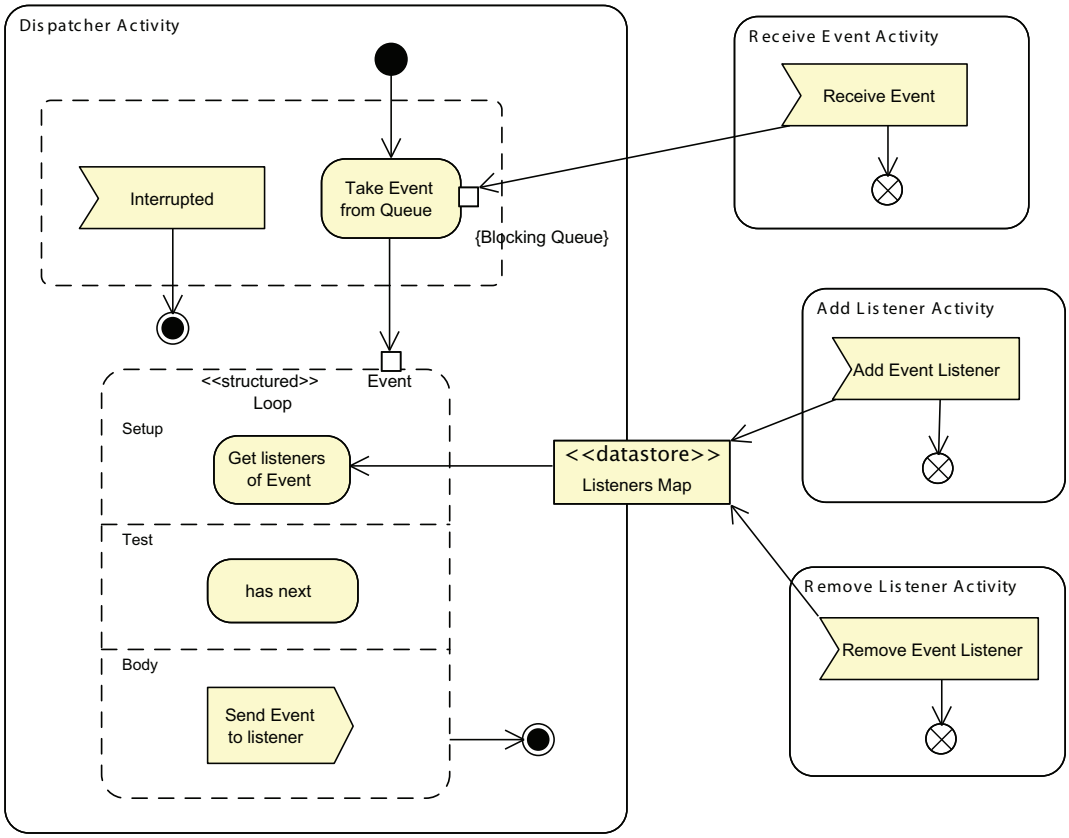


Figura 4.25: Event Dispatcher Activity Diagram

numero di eventi generati (caso in cui sia avvenuta una variazione nel contenuto dei registri) il ritardo dovuto al dispatch dell'evento potrebbe compromettere la reattività di tutto il sistema.

Per ottimizzare questa parte di codice abbastanza onerosa l'idea è quella di realizzare l'intero generatore di eventi in codice nativo inoltrando il risultato dei confronti verso opportuni metodi callback di una classe Java e successivamente eseguire il dispaching dell'evento come sempre fatto. Oltre ad aumentare le performances l'idea è anche quella di eliminare completamente la noia dovuta all'accesso al default package consentendo l'eliminazione della Java Reflection per accedere alle classi, e ai rispettivi metodi, contenute al suo interno.

Uno degli aspetti critici di questa operazione è la necessità di cross-compilare tutto il codice nativo: cosa che richiede di avere sulla propria macchina tutta una

serie di tools opportunamente configurati per generare eseguibili e librerie per il target system. Purtroppo non viene fornita indicazione di come realizzare una stazione predisposta a questo scopo e quindi c'è voluto molto tempo, composto da vari tentativi, per ottenere una configurazione valida e adatta allo scopo. Fortunatamente è possibile installare, tramite il gestore a pacchetti Debian, arm-linux-gnueabi-gcc.

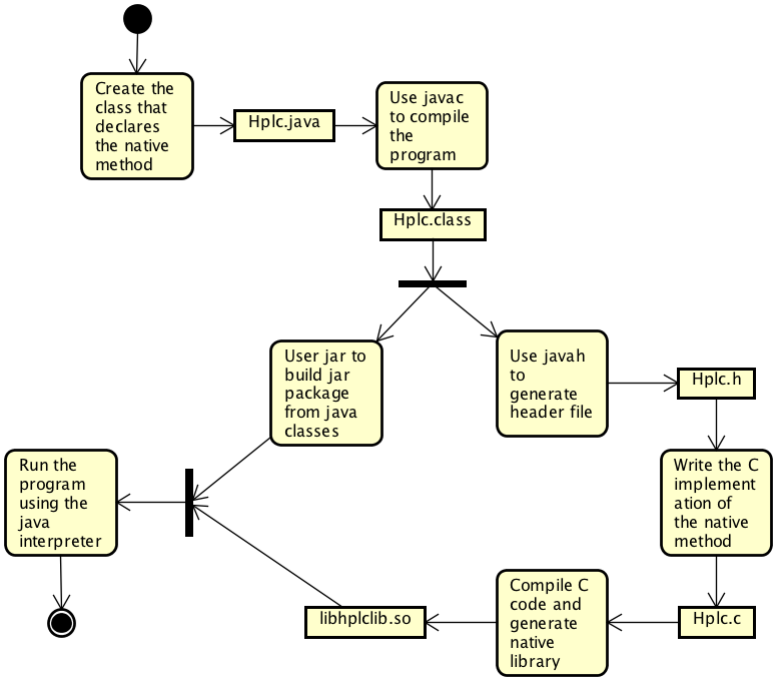


Figura 4.26: Procedimento per generare la native library

Come mostrato in figura Java permette di generare gli header file per le classi che contengono dichiarazioni di metodi nativi.
Viene mantenuto il file sorgente originale per garantire compatibilità con le vecchie implementazioni ma vengono aggiunti gli header file e i sorgenti per le nuove classi Java contenute nel subpackage privato al bundle.
La classe di test ottiene il riferimento all'implementazione dell'interfaccia HomePLC mentre la classe HomePLCProxy si preoccupa di caricare in memoria la classe Hplc con i metodi nativi e la shared library nativa con i file necessari.

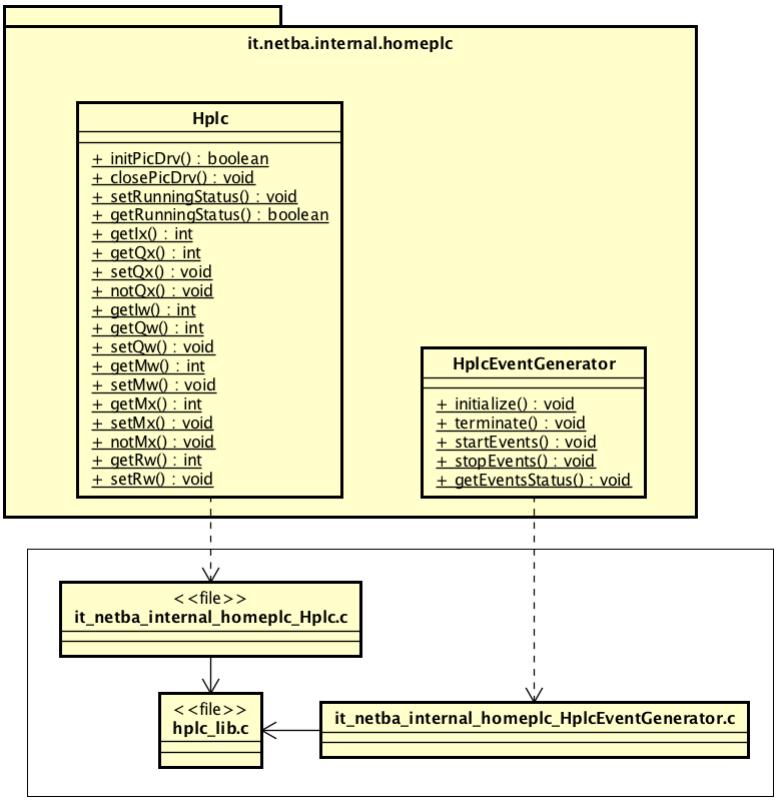


Figura 4.27: Nuova composizione della libreria nativa.

4.3 openHAB come sistema di automazione

"Welcome to openHAB - a vendor and technology agnostic open source automation software for your home. Build your smart home in no time!"
Questo è quello che si può leggere nella prima pagina del sito dedicato all'implementazione opensource di riferimento di quello che nel tempo è diventato il framework SmartHome: progetto per la Home Automation che fa parte dell'ecosistema dedicato alla IoT della Eclipse Foundation. Ma andiamo con ordine.
openHAB è un insieme di bundle installati su di un framework OSGi (Equinox). E' una soluzione realizzata interamente in Java e che di conseguenza richiede una JVM per essere avviato. Essendo basato su OSGi l'architettura è fortemente modulare e permette di aggiungere o rimuovere funzionalità mentre è in

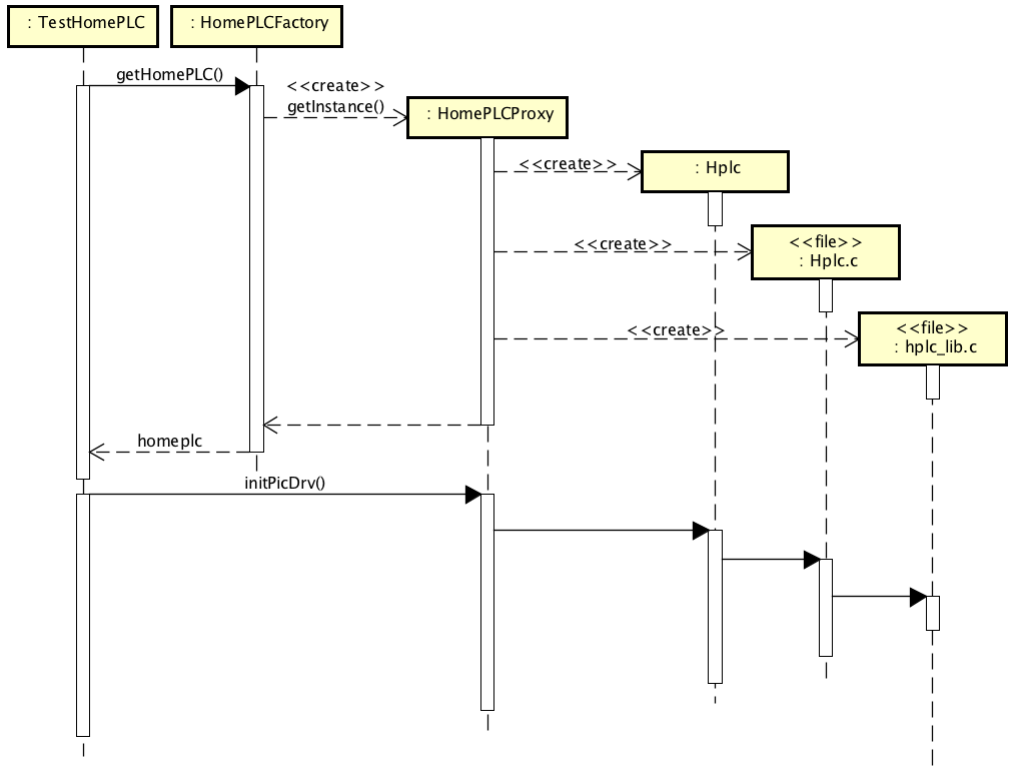


Figura 4.28: Esempio di chiamata a metodo nativo senza Java reflection

esecuzione senza arrestare i restanti servizi.
L'event bus è il servizio principale di openHAB e tutti i bundles possono utilizzarlo per informare altri mediante emissione di propri eventi o ricevere informazioni da altri bundle mediante ricezione di eventi esterni. Ci sono due tipi di evento: *command* che avviano una azione o un cambiamento di stato di qualche Item/Device e *status update* che modificano lo stato di qualche Item/-Device (spesso in risposta ad un *command*). Tutti i binding (che realizzano il collegamento con i dispositivi hardware) dovrebbero comunicare tramite l'event bus in modo da rendere l'accoppiamento tra di essi molto debole e di conseguenza assecondare la natura fortemente dinamica di openHAB. Tecnicamente per realizzare l'event bus viene utilizzato l'OSGi EventAdmin Service che altro non è che una implementazione di una architettura publish-subscribe che svolge perfettamente il suo compito.

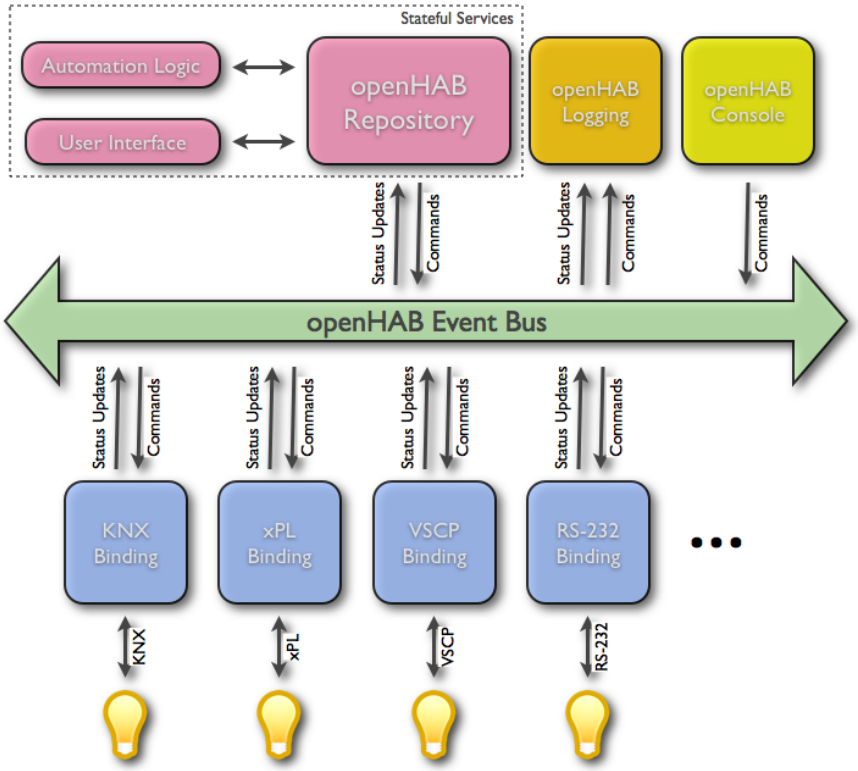


Figura 4.29: openHAB event bus

Spesso è necessario accedere allo stato corrente degli Item come nel caso della logica di automazione e dell'interfaccia grafica che deve mostrare all'utente sempre lo stato aggiornato degli Item. Il componente che svolge questo compitò è l'openHAB Repository che è anch'esso connesso all'event bus e tiene traccia di tutti i cambiamenti di stato degli Item. L'utilizzo di questo componente dedicato evita che ogni binding debba mantenere internamente una cache degli stati a cui è interessanto per un suo proprio utilizzo e automaticamente mantenendo sincronizzati e disponibili questi stati per tutti i restanti bundle.

Items

Uno dei concetti principali di openHAB è quello di Item. Un Item è un mattone importante alla base del sistema che basa la sua natura sul concetto di dato e non è legato a un particolare dispositivo fisico, una sorgente virtuale quale potrebbe

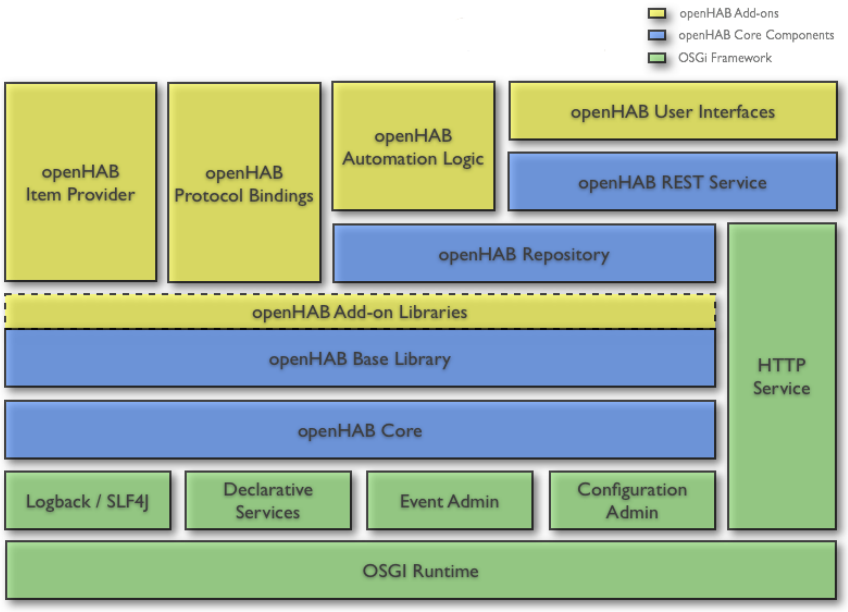


Figura 4.30: openHAB Architecture Overview

essere un web service o il risultato di qualche calcolo. Tutte le funzionalità di openHAB sono basate sull'astrazione di Item e, ad esempio, non si troverà mai nessun riferimento a informazioni specifiche tipo indirizzi IP, Id, etc. nelle Rules, nella definizione della UI e così via. Questo consente in qualunque momento di sostituire una tecnologia con un'altra senza modificare Rules o UI.
Un Item è definito dalla omonima interfaccia alla quale è associata una classe astratta chiamata GenericItem che ne realizza una prima implementazione. E' un oggetto dotato di stato interno e quando questo stato viene modificato invia notifiche a una lista di listeners che si sono registrati presso di lui.
L'oggetto di tipo StateChangeListener accetta due tipi di notifica:

- stateUpdated ogni volta che viene chiesto di aggiornare lo stato dell'Item tramite il metodo setState
- stateChanged nel caso il valore associato allo stato dell'Item è realmente cambiato

Da notare che se uno StateChangeListener riceve una notifica di stateChanged riceverà anche una notifica di tipo stateUpdated.

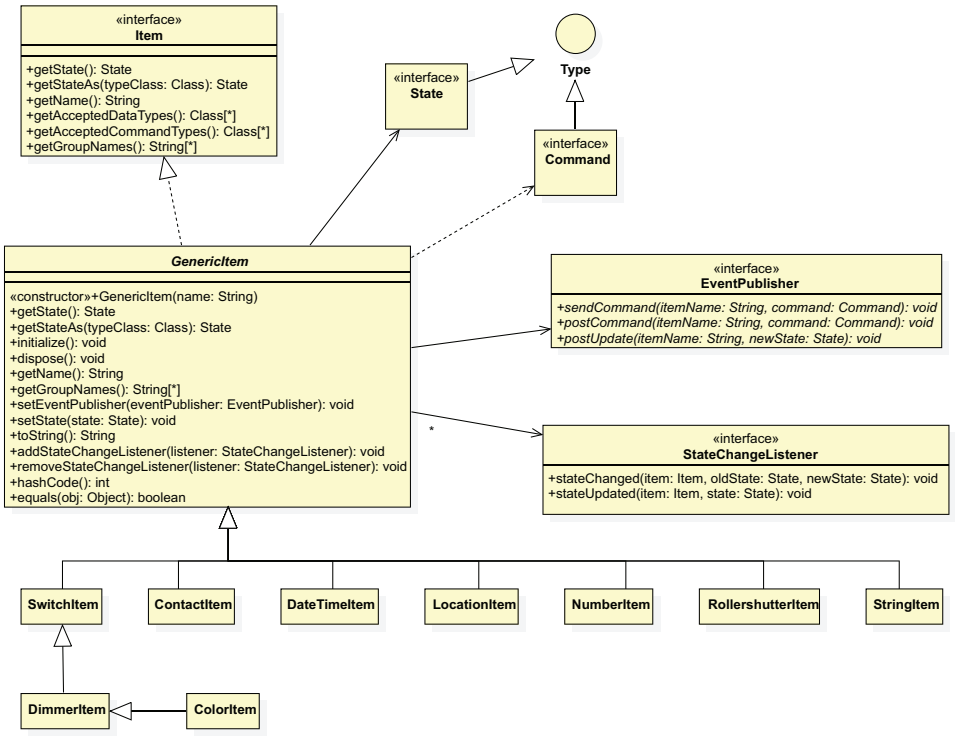


Figura 4.31: openHAB Items Model

L'Item può anche pubblicare comandi sull'event bus sempre che un oggetto di tipo EventPublisher venga registrato all'Item.
Stranamente il motodo internalSend() non è richiamato da tutte le implementazioni di Item. Solo SwitchItem, DimmerItem, ContactItem e ColorItem implementano il metodo send() che internamente richiama internalSend(). Il metodo send() non appartiene ad alcuna interfaccia implementata dagli Item così è probabile che sia una parte di codice che è stata dimenticata e mai revisionata.
Prima di dettagliare i tipi di Item definiti in openHAB è giusto introdurre l'entità Type e tutte le classi derivate in modo da comprendere come venga gestito lo stato interno di ogni Item.

Types

Il modello dei Types è composto sia da interfacce che fungono da markers sia da classi concrete.

Algoritmo 4.3 GenericItem implemetation

```
protected void internalSend(Command command) {
  // try to send the command to the bus
  if(eventPublisher!=null) {
    eventPublisher.sendCommand(this.getName(), command);
  }
}

public void setState(State state) {
  State oldState = this.state;
  this.state = state;
  notifyListeners(oldState, state);
}

private void notifyListeners(State oldState, State newState) {
  // if nothing has changed, we send update notifications
  Set<StateChangeListener> clonedListeners = null;
  clonedListeners = new CopyOnWriteArraySet<StateChangeListener>(listeners);

  for(StateChangeListener listener : clonedListeners) {
    listener.stateUpdated(this, newState);
  }

  if(!oldState.equals(newState)) {
    for(StateChangeListener listener : clonedListeners) {
      listener.stateChanged(this, oldState, newState);
    }
  }
}
```

Ci sono casi in cui lo stato degli Item non possiede un valore definito e può accadere perchè non sono ancora stati inizializzati (non hanno ancora ricevuto un update dello stato) oppure perchè è ambiguo (una luce dimmerabile trattata come Switch Item e impostata a un valore ad esempio del 50%) e quindi, per questi casi viene definito UnDefType che può avere due valori UNDEF e NULL (non ancora inizializzato).
La quasi interezza dei Type definiti appartengono alla categoria dei PrimitiveType tra i quali ci sono classi enum che possono accettare solo due valori (OnOffType, OpenClosetype, UpDownType e StopMoveType) e altre più complesse come DecimalType, derivata dalla classe Java Number e Comparable, che internamente utilizza un Java BigDecimal e quindi può essere utilizzata indistintamente per valori integer, long e di tipo floating point.
Direttamente da DecimalType deriva PercentType (utilizzata ad esempio per esprimere grandezze tipo il volume di uno stereo) che può assumere valori tra 0 e

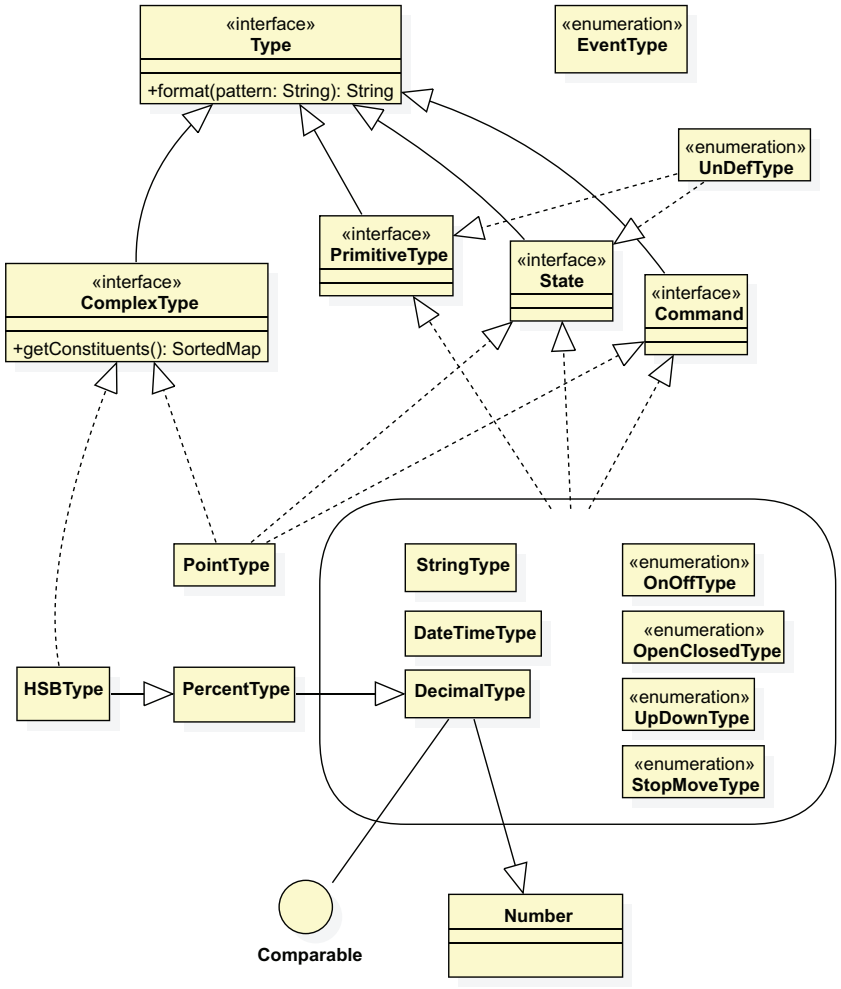


Figura 4.32: openHAB Types Model

100 e una ulteriore classe, di tipo ComplexType, chiamata HSBType utilizzata per esprimere i colori (Hue-Saturation-Brightness). Proprio perchè HSBType è una ComplexType la grandezza che esprime può essere scomposta in più valori e allora si capisce come questo tipo può essere espresso con un valore che va da 0 a 360 per Hue e da 0 a 100 per Saturation e Brightness.
A parte, rispetto a tutto questo discorso, si trova la classe EventType. Tale classe è stata introdotto in quanto alcuni Type possono essere indifferentemente sia State che Command e allora, quando un messaggio viene pubblicato sull'event bus, è necessario un modo per distinguere i significato del messaggio: quando

osservo "<item> ON" intendo che il suo stato è cambiato in ON oppure che deve essere azionato a ON? Per decidere si invia questa informazione aggiuntiva sull'event bus per ogni messaggio.

Gli Item, i Type e lo State

Tornando al discorso precedentemente interrotto sugli Item si nota come tutta una serie di classi derivi da una base comune astratta data da GenericItem. Tutti i tipi di Item contengono definita staticamente al loro interno la lista di State e Command che possono accettare ovvero quali tipi di informazioni possono essere salvate al loro interno e quali comandi possono essere inviati dal particolare Item.
Allora ad esempio uno SwitchItem accetta dei dati di tipo OnOffType e UnDefType mentre può inviare comandi del tipo OnOffType.

Algoritmo 4.4 SwitchItem static initialization

```
private static List<Class<? extends State>> acceptedDataTypes =
      new ArrayList<Class<? extends State>>();
private static List<Class<? extends Command>> acceptedCommandTypes =
      new ArrayList<Class<? extends Command>>();

static {
  acceptedDataTypes.add(OnOffType.class);
  acceptedDataTypes.add(UnDefType.class);
  acceptedCommandTypes.add(OnOffType.class);
}
```

Quindi, sebbene sembri quasi che ad ogni particolare Item sia associato un particolare Type, in realtà la situazione è un pò più articolata e si può notare come lo stato di un Item possa variare liberamente in un range di valori definito dal particolare Type associato al valore che si intende memorizzare.

EventPublisher

L'architettura di openHAB viene connessa all'Event Bus tramite due interfacce appartenenti al framework OSGi: EventAdmin e EventHandler. EventAdmin è un servizio col quale, una volta registrati, viene permesso di pubblicare eventi sia in modo sincrono che asincrono mentre EventHandler consente, agli oggetti

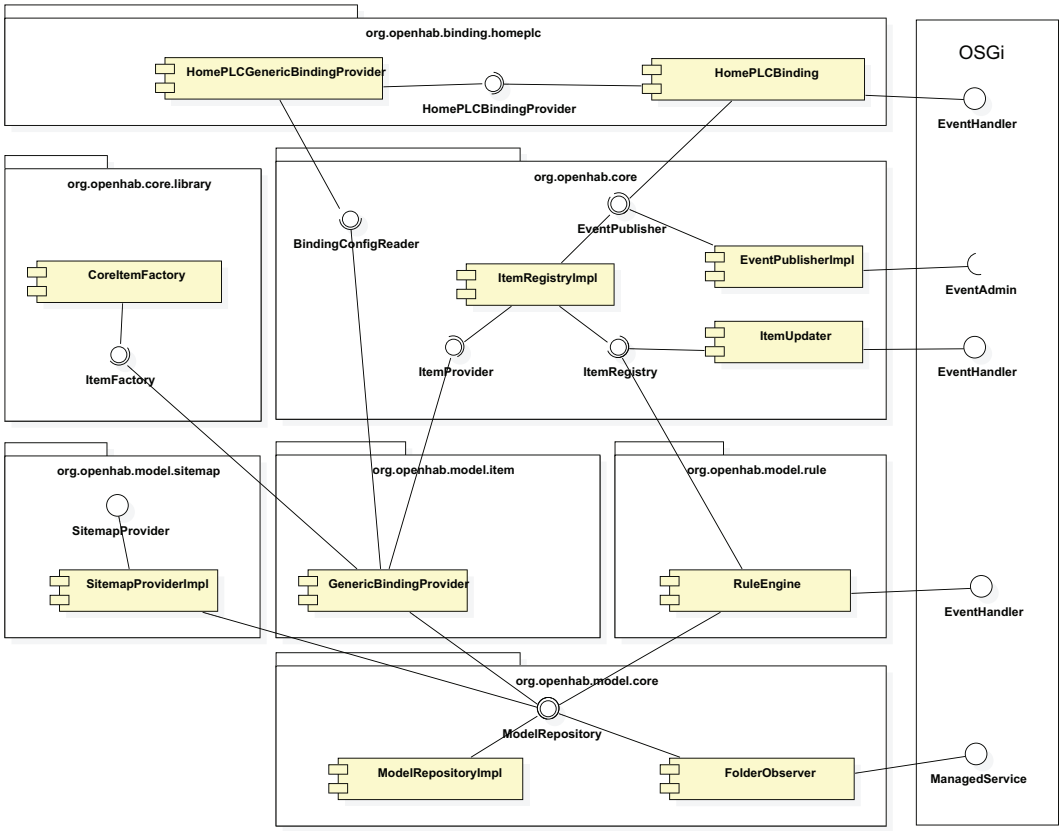


Figura 4.33: openHAB Component Diagram

che implementano tale interfaccia e si registrano al Framework service registry, di ricevere notifiche con un Event object quando tale evento è stato inviato.
openHAB estende queste interfacce includendo i concetti di State e Command che abbiamo visto in precedenza.
Come avevamo visto occorre distinguere il tipo di openHAB Event da inviare sul bus altrimenti non saprei come trattare un evento: come comando o come aggiornamento di stato.
Per risolvere questo problema prima di inviare un evento la classe EventPublisher prepara il tipo di evento e lo confeziona componendo il parametro Topic dell'Evento OSGi con opportune stringhe di testo. Ovviamente due sono i casi possibili:

- `openhab/command/<item-name>`

Algoritmo 4.5 openHAB EventPublisher internals

```
public interface EventConstants {
  public static final String TOPIC_PREFIX = "openhab";
  public static final String TOPIC_SEPERATOR = "/";
}

public class EventPublisherImpl implements EventPublisher {

  //...

  private Event createUpdateEvent(String itemName, State newState) {
    Dictionary<String, Object> properties = new Hashtable<String, Object>();
    properties.put("item", itemName);
    properties.put("state", newState);
    return new Event(createTopic(EventType.UPDATE, itemName), properties);
  }

  private Event createCommandEvent(String itemName, Command command) {
    Dictionary<String, Object> properties = new Hashtable<String, Object>();
    properties.put("item", itemName);
    properties.put("command", command);
    return new Event(createTopic(EventType.COMMAND, itemName) , properties);
  }

  private String createTopic(EventType type, String itemName) {
    return TOPIC_PREFIX + TOPIC_SEPERATOR + type + TOPIC_SEPERATOR + itemName;
  }
}
```

- `openhab/update/<item-name>`

Inserendo sia il tipo di evento che il nome dell'Item che ha generato il comando (o a cui è destinato l'aggiornamento di stato) è facile immaginare che, da qualche parte, ci sia del codice che applica opportune regole di filtraggio.

ItemRegistry

L'ItemRegistry è il luogo centrale in cui gli Item vengono memorizzati e il loro stato viene continuamente monitorato. Qualsiasi parte di codice che richiede lo stato corrente di un Item dovrebbe utilizzare questo servizio anzichè mantenere una propria copia locale.
I vari Item vengono registrati tramite gli ItemProvider, elemento che può generarli ricavandoli da una una opportuna sorgente (ad es. un file esterno) e inoltre possono aggiungerli e rimuoverli dinamicamente dall'ItemRegistry.

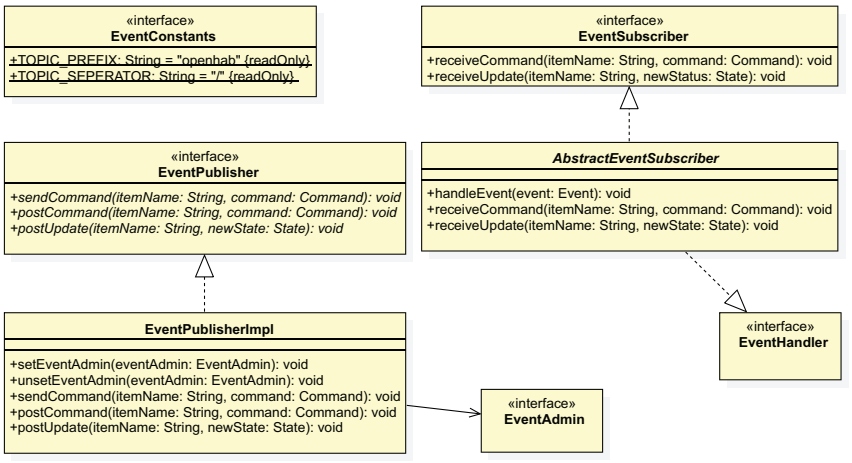


Figura 4.34: openHAB link with OSGi EventAdmin Service

L'ItemRegistry aggiunge il riferimento all'EventPublisher ad ogni Item che viene inserito al suo interno.

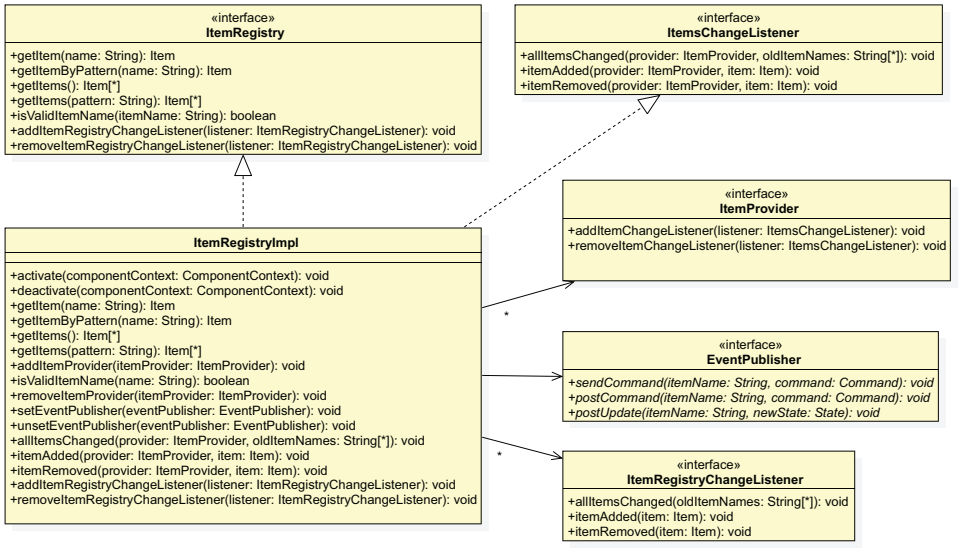


Figura 4.35: ItemRegistry structure Model

ItemProvider

openHAB permette di definire gli Item in file di testo contenuti all'interno di una particolare directory del runtime. In tali file sono contenute le definizioni degli Item e la stringa di configurazione del binding che si vuole integrare con openHAB. Queste definizioni devono rispettare le specifiche definite da delle grammatiche espresse in EMF.
La classe GenericItemProvider realizza l'interfaccia ItemProvider.
Quando tramite il servizio ConfigAdmin di OSGi viene rilevato che uno dei file *.items è stato modificato il contenuto del ModelRepository viene aggiornato e successivamente viene inviata una notifica alla classe GenericItemProvider per ogni modello che è stato modificato. Come modello si intende l'intero file *.items.
Ricevuto il nome del modello che è stato modificato GenericItemProvider rimuove le configurazioni contenute nel modello da tutti i BindingConfigReader che conosce e successivamente genera dal nuovo modello preso dal ModelRepository una nuova istanza di tutti gli Item.
Creati i nuovi Item, per ognuno di essi, viene chiesto ai BindingConfigReader corrispondenti di validare il tipo di Item e processarne la configurazione. Nel caso che venga rilevato qualche problema di incompatibilità o di configurazione l'Item viene scartato.
Per concludere vengono notificati tutti gli ItemChangeListener. Tra i listener ovviamente c'è ItemRegistry che provvede a rimuovere tutti gli Item vecchi inseriti dall'ItemProvider e li sostituisce con tutti quelli nuovi appena processati. Si conclude notificando tutti gli ItemRegistryChangeListener.

BindingConfigReader

BindingConfigReader è una interfaccia che viene implementata dai vari BindingProvider che possono essere realizzati da chiunque voglia integrare il proprio sistema in openHAB.

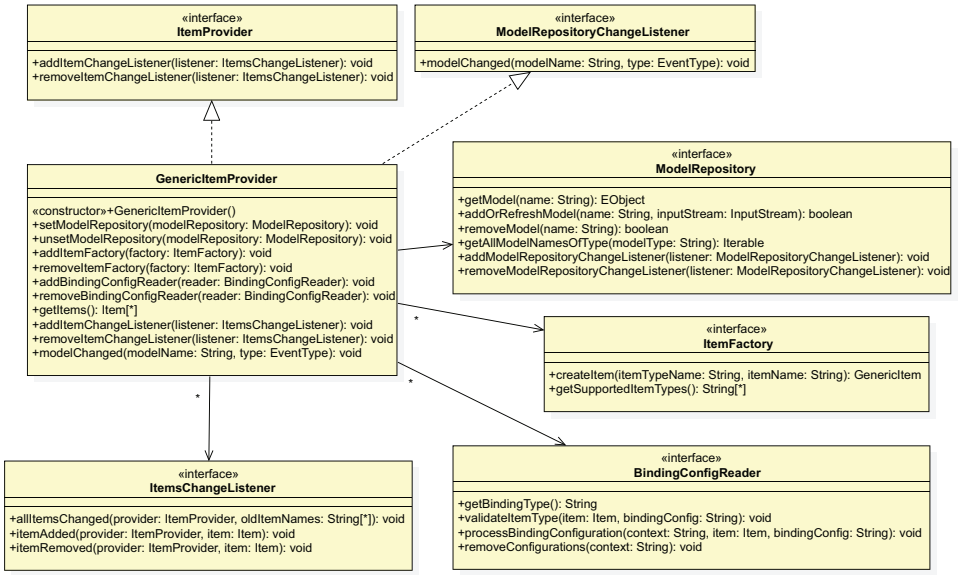


Figura 4.36: ItemProvider structure Model

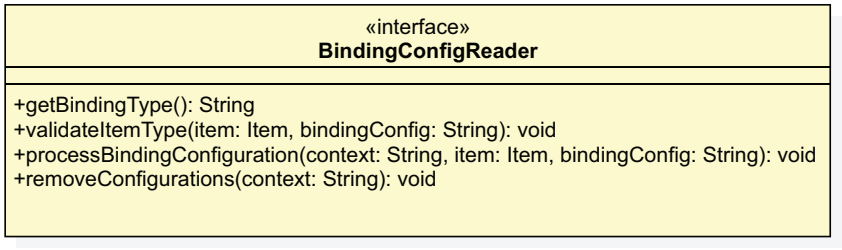


Figura 4.37: BindingConfigReader Model

Refactoring di EventGenerator in OSGi bundle

A questo punto abbiamo a disposizione una libreria in grado di interfacciare un ipotetico sistema software, scritto in linguaggio Java, con l'hardware domotico fornito dall'azienda NET Building Automation. Due sono le interfacce fornite: la prima, ereditata dal pacchetto originale e lasciata pressochè identica, in grado di abilitare il driver del dispositivo domotico e di leggere e scrivere i registri di HomePLC e la seconda, progettata ex novo, in grado di generare eventi ogni qualvolta un registro di HomePLC subisce una variazione.

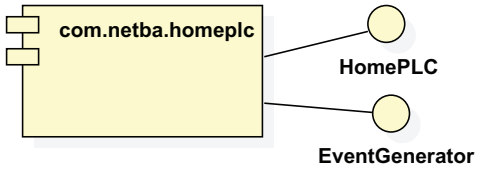


Figura 4.38: Componente OSGi

Nell'ottica di un futuro sviluppo per componenti si è voluto aggiungere una funzionalità in più alla libreria a nostra disposizione inserendo meccanismi in grado di abilitare il suo utilizzo anche con i framework OSGi.
Due sono i servizi forniti dal bundle e corrispondono alle interfacce Java presenti nei rispettivi packages. OSGi necessita anche delle classi che implementano i due servizi in quanto il bundle fa anche la parte di service provider.
Per permettere ad OSGi di caricare in memoria questi servizi è stato utilizzato il meccanismo dei Declarative Services: in particolare la definizione del componente per il generatore di eventi è stato impostato in modo tale da richiedere la presenza di un servizio HomePLC in modo tale da stabilire una priorità nella sequenza di caricamento. Entrambi i componenti vengono immediatamente attivati.

Struttura openHAB e HomePLCBinding

Col termine Binding si intende il componente di tutta l'architettura openHAB che viene realizzato per integrare all'interno del sistema un particolare dispositivo, un servizio internet o fino a un intero sistema di home automation. Nell'ottica di realizzare il binding per il sistema HomePLC occorre, come prima cosa, considerare quali tipi di informazioni vengono scambiate con openHAB: da questi elementi principali poi si sviluppa tutto il resto del componente.
Come abbiamo visto nella descrizione del sistema HomePLC tutti i dispositivi in campo vengono mappati in una tabella di registri contenuta all'interno del controller. I dispositivi per gli I/O digitali (contatti optoisolati e relè di co-

mando) occupano un bit all'interno dei registri ed ecco spiegato perchè si può arrivare ad avere Master o Slave I/O anche da 16 ingressi e 16 uscite.

Item	state	command
ContactItem	UnDefType OpenCloseType	
SwitchItem	UnDefType OnOffType	OnOffType
NumberItem	UnDefType DecimalType	DecimalType
DimmerItem	UnDefType OnOffType PercentType	OnOffType IncreaseDecreaseType PercentType
RollershutterItem	UnDefType UpDownType PercentType	UpDownType StopMoveType PercentType

Tabella 4.7: Item per HomePLCBinding

Master speciali e slave analogici possono partizionare le loro aree di input e di output in maniera più elaborata: posso avere grandezze esprimibili con valori booleani ma anche valori interi oppure valori appartenenti a grandezze reali e quindi dotate di valori decimali. Ogni dispositivo viene fornito completo di tabella di mappatura dalla quale è possibile ricavare con quanti bit esprimere un valore intero o uno reale opportunamente scalato per farlo rientrare in quello spazio di memoria.
Visti i tipi di dato che possono essere elaborati con HomePLC è stato scelto di utilizzare queste corrispondenze:

- Per i valori binari avremo sia i ContactItem che gli SwitchItem
- Per i valori numerici NumberItem e DimmerItem.

Il bundle org.openhab.binding.homeplc è composto da due componenti principali che consentono di interfacciare il sistema openHAB con il sistema di automazione HomePLC.

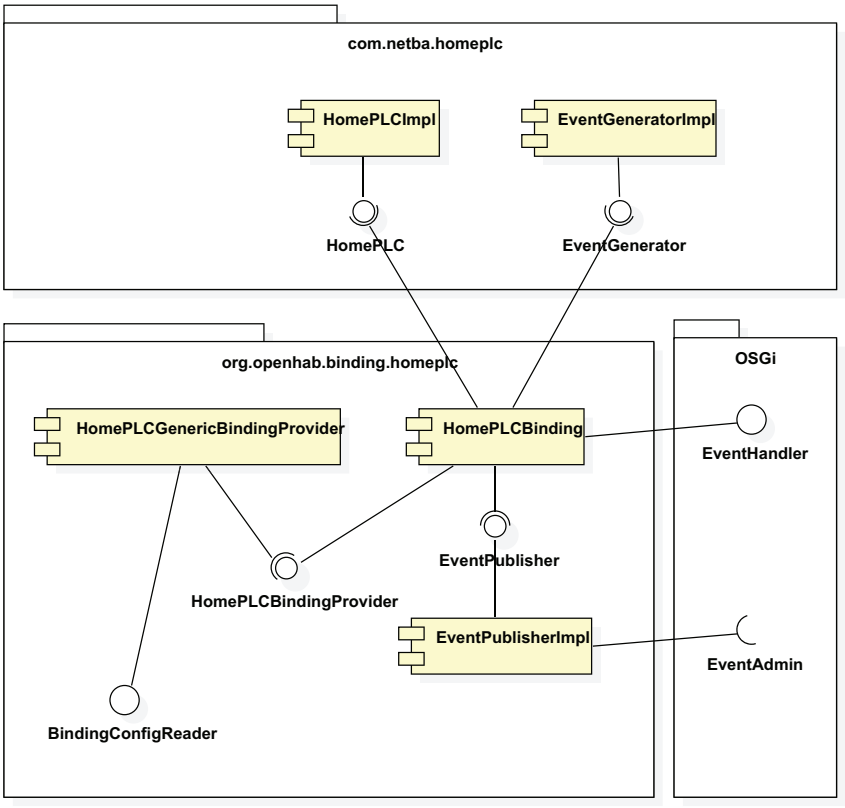
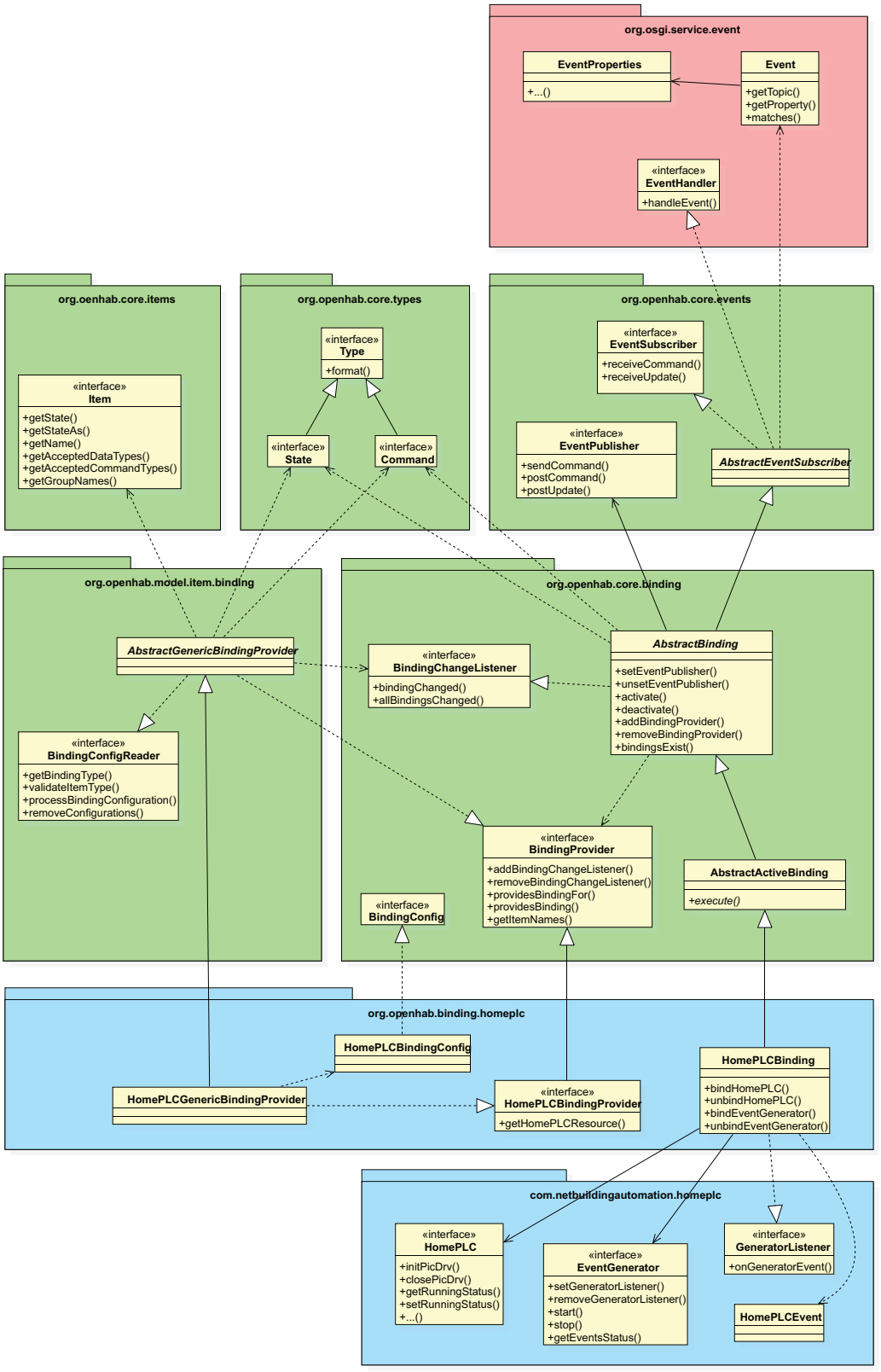


Figura 4.39: Relazioni tra EventGenerator e Binding HomePLC

Il componente HomePLCBinding è quello che mantiene il legame vero e proprio con il sistema di automazione e, tramite le interfacce che abbiamo costruito precedentemente, riesce a inoltrare i comandi che provengono dal sistema openHAB trasformandoli in richieste verso l'interfaccia di HomePLC. HomePLCBinding si comporta anche da observer per il generatore di eventi trasformando tutti gli eventi ricevuti in stateUpdate per gli Item; per inoltrare all'Item l'update viene sfruttato l'EventPublisherImpl e l'event bus che abbiamo visto essere il cuore di tutto openHAB.
La classe HomePLCGenericBindingProvider invece implementa l'interfaccia BindingConfigReader e come abbiamo visto in precedenza permette di gestire la configurazione dei singoli Item appartenenti al binding HomePLC. Una volta che openHAB rileva una modifica nei file *.items HomePLCGenericBindingPro-

vider cerca di validare tutti gli Item per il binding homeplc e successivamente memorizza la configurazione di ognuno. Questa configurazione è importantissima per il componente HomePLCBinding perchè determina con esattezza le trasformazioni che vengono applicate sui Command e sugli Update da e per il binding homeplc.
Quando verrà inviato un Command verso il sistema di automazione tra i parametri comparirà anche lo stato che vogliamo attivare per quel particolare Item. Il tipo di Item, il tipo del nuovo Stato richiesto e la configurazione ricavata dall'HomePLCBindingProvider determineranno le operazioni fatte sull'interfaccia HomePLC.
Analogamente quando viene ricevuto un nuovo evento gli Item dell'HomePLCBindingProvider interessati a quel particolare evento riceveranno uno stateUpdate con un valore determinato dalla particolare natura dell'Item stesso e dalle informazioni ricavate dalla configurazione ottenuta sempre da HomePLCBindingProvider.


org.osgi.service.event
EventProperties
+...()
Event
+getTopic()
+getProperty()
+matches()
«interface»
EventHandler
+handleEvent()
org.oenhab.core.items
«interface»
Item
+getState()
+getStateAs()
+getName()
+getAcceptedDataTypes()
+getAcceptedCommandTypes()
+getGroupNames()
org.openhab.core.types
«interface»
Type
+format()
«interface»
State
«interface»
Command
org.openhab.core.events
«interface»
EventSubscriber
+receiveCommand()
+receiveUpdate()
«interface»
EventPublisher
+sendCommand()
+postCommand()
+postUpdate()
AbstractEventSubscriber
org.openhab.model.item.binding
AbstractGenericBindingProvider
«interface»
BindingConfigReader
+getBindingType()
+validateItemType()
+processBindingConfiguration()
+removeConfigurations()
org.openhab.core.binding
«interface»
BindingChangeListener
+bindingChanged()
+allBindingsChanged()
AbstractBinding
+setEventPublisher()
+unsetEventPublisher()
+activate()
+deactivate()
+addBindingProvider()
+removeBindingProvider()
+bindingsExist()
«interface»
BindingProvider
+addBindingChangeListener()
+removeBindingChangeListener()
+providesBindingFor()
+providesBinding()
+getItemNames()
AbstractActiveBinding
+execute()
«interface»
BindingConfig
org.openhab.binding.homeplc
HomePLCBindingConfig
HomePLCGenericBindingProvider
«interface»
HomePLCBindingProvider
+getHomePLCResource()
HomePLCBinding
+bindHomePLC()
+unbindHomePLC()
+bindEventGenerator()
+unbindEventGenerator()
com.netbuildingautomation.homeplc
«interface»
HomePLC
+initPicDrv()
+closePicDrv()
+getRunningStatus()
+setRunningStatus()
+...()
«interface»
EventGenerator
+setGeneratorListener()
+removeGeneratorListener()
+start()
+stop()
+getEventsStatus()
«interface»
GeneratorListener
+onGeneratorEvent()
HomePLCEvent

4.4 Ambiente di prova e valutazioni

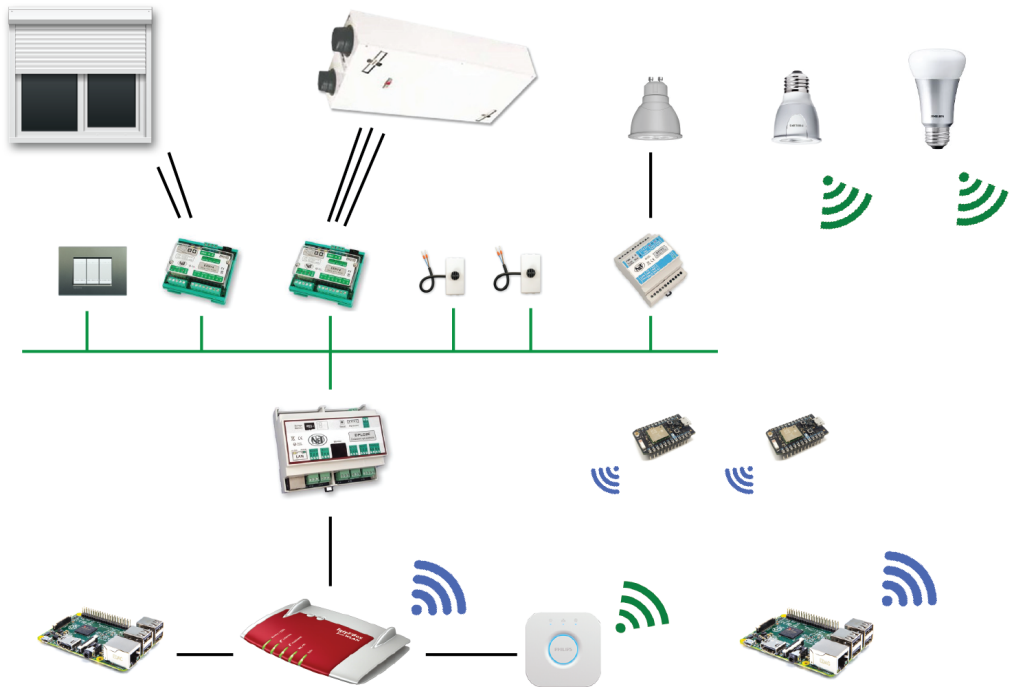


Figura 4.40: Sistema in uso

Per dare un idea più chiara del risultato ottenuto vediamo di illustrare una parte del sistema in uso come indicato nella figura.
Un router centrale all'abitazione fornisce connettività TCP/IP su cavo ethernet e Wi-Fi a diversi dispositivi sparsi per la casa. A questo router sono collegati con cavo ethernet l'HomePLC (per il sistema domotico), un Raspberry Pi e l'Hub Philips HUE. Sul Raspberry Pi è installato openHAB che controllerà tutto l'impianto di Home Automation mentre l'Hub Philips HUE fa da bridge tramite rete ZigBee per alcune luci sparse nelle varie stanze. Il controller HomePLC è connesso con bus proprietario a tutti i dispositivi attualmente acquistati che permettono di rilevare le pressioni dei pulsanti, il comando della tapparella, il comando della Ventilazione Meccanica Controllata (Wolf CWL-F-150 excellent), le sonde di temperatura e umidità, il dimmer per alcune luci.

Tramite rete wireless sono connessi un ulteriore Raspberry Pi con una installazione di Mosquitto MQTT Broker e alcuni dispositivi Particle Photon programmati per la rilevazione di temperatura e umidità tramite sonde DHT22. Sia l'impiando domotico su rete proprietaria HomePLC che il bridge Philips sono integrati in openHAB mediante i binding relativi di cui quello per HomePLC sviluppato in precedenza. Il broker Mosquitto serve per connettere tramite protocollo MQTT i due Photon che rilevano la temperature in alcuni punti ad openHAB su cui è installato un bundle apposito con client MQTT.
Inizialmente questa era la configurazione utilizzata in cui openHAB e l'HomePLC.Linux erano connessi tramite protocollo CoAP in quanto la memoria flash sul dispositivo era limitata e c'erano alcuni problemi ad eseguire openHAB sul dispositivo stesso.
Tutto l'impianto è comandabile tramite i pulsanti a parete oppure tramite browser web che con app per smartphone e tablet (android e iOS) connessi tramite Wi-Fi oppure da remoto su rete cellulare.

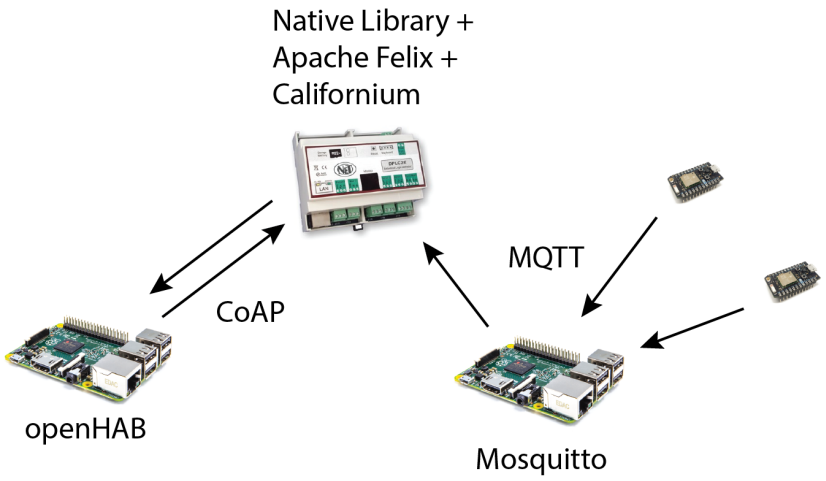


Figura 4.41: Protocolli IoT e openHAB

Inizialmente pensato per un utilizzo personale il progetto ha riscosso un certo interesse in alcuni forum italiani dedicati alla Home Automation favorendone l'adozione anche ad altri interessati.

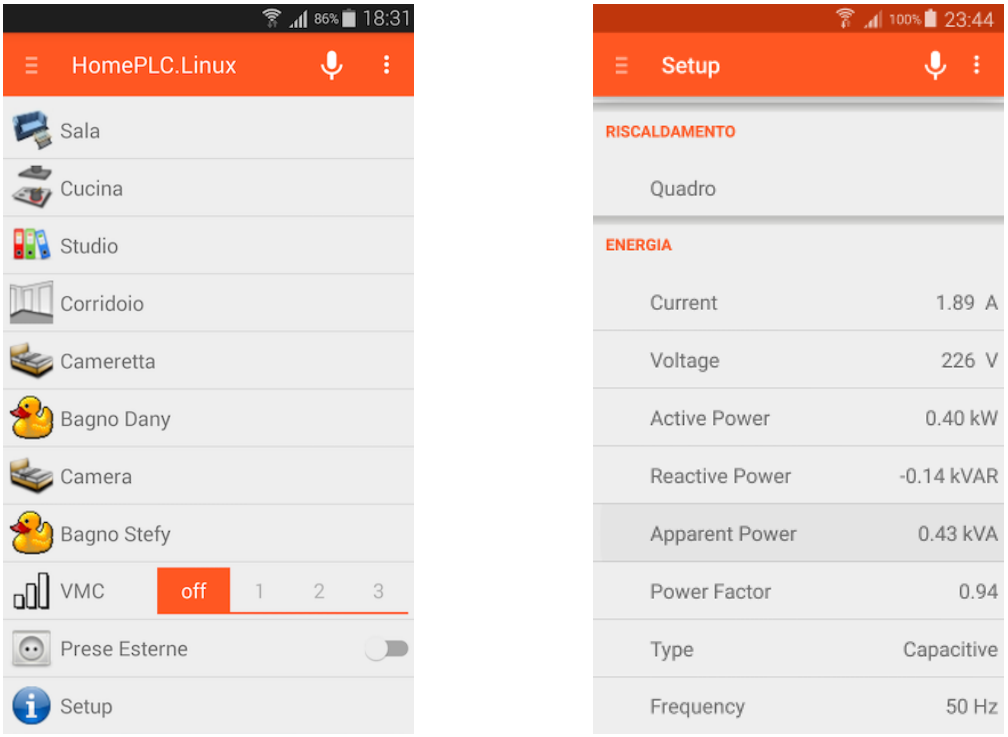


Figura 4.42: Alcune parti dell'interfaccia Android

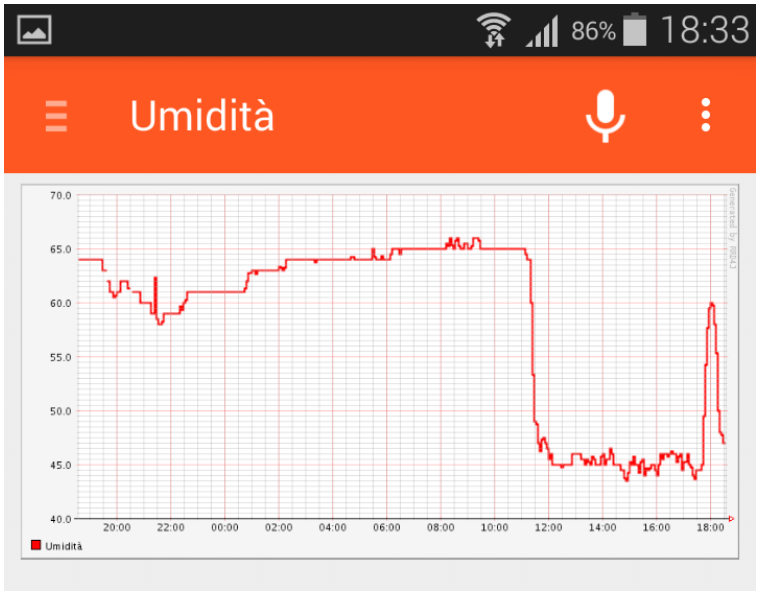


Figura 4.43: Esempio di grafico di umidità

Attualmente vengono mantenute e distribuite due versioni pensate rispettivamente per controller Linux e per Master Web dedicato a installazioni tradizionali per dispositivi programmabili in standard IEC 1131-3.

Valutazioni

Prima di scegliere il fornitore dell'impianto domotico è stata fatta una disamina dei sistemi di Home Automation presenti sul mercato e disponibili presso i fornitori di zona. La prima cosa che val la pena di notare è che quasi tutti gli elettricisti contattati hanno opposto una certa resistenza all'installazione di un impianto domotico. Questo purtroppo rappresenta una situazione ancora abbastanza diffusa in quanto gli artigiani, che normalmente realizzano impianti elettrici, difficilmente investono parte del loro poco, e prezioso, tempo a disposizione per questo tipo di aggiornamento. Il motivo è da ricercare anche nella ridotta domanda di questo genere di installazioni sintomo, come abbiamo visto, di una certa diffidenza da parte dei proprietari degli appartamenti e molto spesso dei costi generalmente molto elevati.
La scelta, alla fine ricaduta sul sistema HomePLC, è dovuta alle particolari caratteristiche che questo sistema domotico fornisce e, non da ultimo, per la realtà interamente italiana di questa azienda. Il sistema HomePLC garantisce, oltre ai dispositivi programmabili in standard IEC 1131-3, anche un controller embedded con sistema operativo Linux sufficientemente potente da far girare senza problemi una macchina virtuale Java.
Peculiarità del sistema HomePLC è la possibilità di molti moduli slave di lavorare autonomamente dovesse mai interrompersi la connessione su bus RS485 oppure dovesse bloccarsi il sistema sul controller Linux. Grazie al microcodice programmabile la logica locale del modulo entra in funzione appena rilevata l'interruzione e, se viene mantenuta una cerca corrispondenza fra la cablatura e la programmazione sul controller, permette di raggiungere una modalità di sicurezza tale da garantire quasi una funzionalità completa.

Sistema Operativo e hardware

Il dispositivo è un embedded con processore i.mx53 ARMv7 della Frescale su cui è installata una versione dedicata di Linux versione 2.6.35.3. Prevedendo che lo sviluppo di software Java avrebbe richiesto qualche risorsa in più è stato acquistata la versione con 1GByte di RAM; purtroppo è sfuggito il dettaglio della memoria flash per sistema operativo e programmi limitata a 256Mbyte.
L'idea iniziale era quella di installare il sistema openHAB direttamente sull'embedded ma essendo basato su una target release di Equinox abbastanza corposa non c'era sufficiente spazio per contenerlo.
Fortunatamente l'embedded è dotato di slot microSD di espansione e successivamente è stato possibile aggiungere una notevole quantità di memoria anche se più lenta della flash interna.

Installazione di openHAB

L'installazione di openHAB è molto semplice: è sufficiente scompattare il file di runtime in una directory. Dopo di che si procede con la programmazione e con l'aggiunta degli addon.
Il problema è stato che la versione originale di openHAB dava dei problemi in avvio e quindi si è reso necessario modificare parte dei pacchetti della target release per eliminare alcuni errori che impedivano al sistema di avviarsi. Come prima cosa occorre scompattare l'archivio con il comando...

```
tar xzf openhab-2.0.0.alpha2-runtime.tar.gz
```

successivamente si devono sostituire i bundles dei pacchetti `slf4j` e `logback` per evitare l'errore...

```
07:03:38,130 |-ERROR in ch.qos.logback.core.util.ContextUtil@83df40b - Failed
     to get local hostname java.net.UnknownHostException: sdpro: sdpro:
    nodename nor servname provided, or not known     at java.net.
    UnknownHostException:...
```

Per risolvere l'errore si devono utilizzare le versioni 1.0.13 di `logback` e 1.7.5 di `slf4j`.

Solo sostituendo i pacchetti con le versioni più aggiornate non è sufficiente, occorre far capire al framework OSGi equinox che sono stati sostituiti dei bundles. Per fare questo non resta che modificare i files

```
<openhab_dir>/runtime/server/artifacts.xml
<openhab_dir>/runtime/server/configuration/config.ini
```

cancellando le voci relative alle vecchie versioni di `logback` e di `slf4j` sostituendole con le nuove

```
<artifacts size='140'>
...
<artifact classifier='osgi.bundle' id='slf4j-api' version='1.7.5' />
<artifact classifier='osgi.bundle' id='logback-classic' version='1.0.13' />
<artifact classifier='osgi.bundle' id='logback-core' version='1.0.13' />
<artifact classifier='osgi.bundle' id='jul-to-slf4j' version='1.7.5' />
<artifact classifier='osgi.bundle' id='log4j-over-slf4j' version='1.7.5' />
<artifact classifier='osgi.bundle' id='jcl-over-slf4j' version='1.7.5' />
...
</artifact>
```

dove il parametro size deve essere modificato da 141 a 140 in quanto le librerie `logback` di versione 1.0.13 non comprendono più il bundle
`ch.qos.logback.slf4j_1.0.13` e

```
...
osgi.bundles=reference\:file\:logback-classic_1.0.13.jar@1\:start,reference\:
    file\:logback-core_1.0.13.jar@1\:start,...,reference\:file\:slf4j-api_1
    .7.5.jar@4,reference\:file\:jcl-over-slf4j_1.7.5.jar@4,reference\:file\:
    jul-to-slf4j_1.7.5.jar@4,reference\:file\:log4j-over-slf4j_1.7.5.jar.
    jar@4
...
```

dove i punti di sospensione indicano che potrebbero esserci altre righe di codice (trascurabili).

Problemi iniziali

La libreria nativa per l'interazione del linguaggio Java con il sistema domotico presentava una limitazione dovuta probabilmente alla poca dimestichezza con questo linguaggio in quanto l'unica possibilità di richiamare i metodi nativi della classe di interfaccia era inserendo tutte le classi nel default package.

Come è possibile notare dal progetto si è dovuto ricorrere allo strumento della Java Reflection per accedere alle classi del default package e creare una classe proxy per nascondere il problema al resto del sistema che finalmente poteva essere strutturato al meglio.

Generatore di eventi

La tabella delle risorse HomePLC contiene 8000 registri da 16 bit e l'unico metodo disponibile di default e quello che permette di leggere un registro alla volta.
Per generare quello che abbiamo chiamato eventi la libreria inizialmente, anche per testare i tempi richiesti, effettuava una lettura di tutti e 8000 i registri impiegando generalmente tra i 50 e i 60 msec (compresi i confronti registro per registro).
Bisogna ammettere che il tempo richiesto non è molto confortante soprattutto perchè in questi 50 msec il tempo impiegato non comprendeva che pochi eventi realmente emessi. Infatti i test riguardavano la pressione di un paio di pulsanti e, nonostante i pochi eventi, il carico della cpu raggiungeva tranquillamente il 50% di picco. Inoltre il periodo di lettura era stato impostato a 100 msec quindi rimaneva veramente poco tempo per l'inoltro degli eventuali eventi col rischio ritardi sul ciclo di lettura successivo.
Per ottimizzare le prestazioni del generatore si è ridotto la lettura degli 8000 registri alle due aree realmente necessarie: l'area real-time e l'area estesa. Visto che nel sistema HomePLC non erano presenti particolari dispositivi che richiedevano altre aree a disposizione l'ottimizzazione è risultata assolutamente idonea.
Così facendo il tempo di lettura di questi nuovi 1000 registri richiedeva un tempo che oscillava tra i 6 e i 15 msec e un carico del processore limitato al 2%. Tutti questi test inoltre sono stati effettuati con una cpu da 1 GHz quindi con processori meno potenti i risultati potrebbero essere ancora più sconfortanti.

Ottimizzazione con codice nativo

Occorreva quindi trovare una soluzione a questa situazione altrimenti tutto il progetto si sarebbe fermato. L'unica alternativa era ottimizzare il codice della generazione degli eventi realizzandolo in C successivamente richiamare metodi di callback in Java per la consegna degli eventi.
Il problema è stato predisporre l'intero ambiente di produzione con tutto il necessario per la cross-compilazione in quanto non fornito ufficialmente dalla ditta. C'è voluto molto tempo per impostare un progetto eclipse CDT con le giuste librerie e i parametri per il tipo di cpu ma alla fine, mediante una macchina Linux con ubuntu e i pacchetti gcc-arm-linux-gnueabi, si è riuscito a generare il primo eseguibile per l'embedded in questione.
Dai primi test effettuati la lettura dei 1000 registri e il relativo confronto impiegava ora 200usec, esattamente due ordini di grandezza in meno, mantenendo un carico della cpu al 1%.
Visto che è stato messo mano al codice nativo tanto valeva fare in modo da eliminare l'accesso tramite Java Reflection e quindi ottimizzare ancora di più l'accesso al codice nativo. Con questa ulteriore ottimizzazione la lettura e confronto dei 1000 registri impiegava in media 60usec con picchi sporadici sui 100usec. La chiamata al metodo di callback Java si attestava sui 500usec ognuna.

5 Conclusioni

Lo scopo di questo lavoro è stato quello di sfruttare concretamente le tecnologie che ha reso disponibili lo sviluppo della IoT calate in un contesto reale di Home Automation.
Uno dei problemi che hanno sempre mostrato i sistemi di Home Automation, e che ne hanno sempre limitato l'adozione, é sicuramente l'alto costo di acquisto e installazione unito ai successivi costi di manutenzione nel caso di problemi o di aggiornamenti.
La spinta che la IoT ha fornito al mercato dei dispositivi per il confort, la sicurezza e la gestione degli automatismi in ambito residenziale ha permesso la comparsa di numerosi dispositivi, dal prezzo più contenuto, dotati di connessione ad internet e di relativi servizi basati su cloud.
Queste soluzioni però creano applicazioni verticali che poco hanno a che vedere con gli scenari di integrazione orizzontale che la IoT promette di raggiungere.
Al fine di dotare la propria abitazione di un sistema domotico aperto, programmabile, espandibile e dai bassi costi di manutenzione si è pensato di realizzare un personale sistema di Home Automation basato interamente su software opensource e quindi limitando alle sole componenti hardware i costi sostenuti.
Dopo una iniziale realizzazione di un sistema domotico di test, per valutare la bontà dell'implementazione ottenuta modificando le librerie di base fornite dal sistema domotico proprietario, si è realizzato il vero componente di integrazione tra il sistema di Home Automation tradizionale e openHAB; sistema di integrazione indipendente dalla particolare piattaforma e protocollo di comunicazione.
L'utilizzo di openHAB, del framework OSGi e i sorgenti delle librerie che il

controller del sistema di automazione rende disponibili ha consentito di raggiungere un risultato di integrazione dotato di notevole flessibilità. Se da un lato è possibile controllare ogni piccolo dettaglio del sistema domotico proprietario dall'altro è possibile anche integrare uno qualunque dei sistemi presenti a catalogo del sistema openHAB. Il fatto notevole, che deriva direttamente dalle potenzialità di OSGi, è di avere a disposizione un sistema modulare e aperto in cui servizi, dispositivi e interi sistemi possono entrare e uscire liberamente mostrando chiaramente la flessibilità fornita da queste tecnologie.
Il sistema così realizzato è stato installato nella propria abitazione e da diversi mesi è in funzione regolarmente senza interruzione ricevendo comandi in locale e da remoto tramite le app per i principali sistemi mobile. Inoltre sono stati integrati più dispositivi e apparecchiature di fornitori differenti sfruttando il sistema openHAB e le tecnologie peculiari della IoT.
A riprova del risultato ottenuto e le particolarità di questa soluzione di Home Automation, il produttore del sistema domotico utilizzato come piattaforma hardware, ne ha espresso la sua valutazione favorevole e il suo interesse autorizzando a fornire una qualche forma di supporto, sui forum di assistenza ufficiali, a chiunque sia intenzionato a installare tale prodotto sui dispositivi dell'azienda.
Ciò che manca in questa trattazione, che rimane comunque di fondamentale importanza, è l'aspetto trasversale della sicurezza nella IoT e nelle relative applicazioni. Ad ogni modo la soluzione realizzata sfrutta gli strumenti forniti dal sistema openHAB e gli strumenti adottati dai dispositivi che forniscono connettività all'intera abitazione ma manca un'analisi dettagliata degli aspetti relativi alla sicurezza di questo sistema.
Inoltre sarebbe stato interessante mostrare i vantaggi che si possono ottenere adottando una soluzione di questo tipo analizzando e valutando scenari in cui, dati alla mano, risulti un reale vantaggio in termini di risparmio economico oppure di consumo.

Bibliografia

[1] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols and Applications. *IEEE Communications Surveys & Tutorials*, PP(99):1–1, 2015.

[2] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[3] N Bartlett. OSGi In Practice. *Bd January*, 11:229, 2009.

[4] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice Third Edition*. Addison-Wesley Professional, 2012.

[5] Simon Bennet, John Skelton, and Ken Lunn. *UML*. McGraw-Hill Education, 2001.

[6] Joshua Bloch. *Effective Java?* Addison-Wesley Professional, 2008.

[7] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. *Pattern-Oriented Software Architecture, Volume 4: A Pattern Language for Distributed Computing*. Number 1. John Wiley & Sons, 2014.

[8] F Bushmann, Regine Meunier, Hans Rohnert, Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-oriented software architecture: A system of patterns*, volume Vol. 1. John Wiley & Sons, 1996.

[9] Kirk Chen and Li Gong. *Programming Open Service Gateways with Java Embedded Server(TM) Technology*. Addison-Wesley Professional, 2001.

[10] S Cirani, M Picone, and L Veltri. mjCoAP: an open-source lightweight Java CoAP library for internet of things applications. *... and Open-Source Solutions for the Internet of ...*, 2015.

[11] Cisco and Dave Evans. Internet of Things L ' Internet delle cose Tutto cambierà con la prossima era di Internet. 2011.

[12] Ian Darwin. *Java Cookbook, 3rd Edition*. O'Reilly Media, Inc., 2014.

[13] Simon Duquennoy, Gilles Grimaud, Jean-jacques Vandewalle, and Jean-jacques Vandewalle The. The Web of Things : interconnecting devices with high usability and performance To cite this version : The Web of Things : interconnecting devices with high usability and performance. 2009.

[14] Ira R. Forman, Nate Forman, Dr John Vlissides Ibm, Ira R. Forman, and Nate Forman. *Java Reflection in Action*. Manning Pubns Co, 2004.

[15] David Gann, James Barlow, and Tim Venables. *Digital Futures: making homes smarter*. 1999.

[16] Brian Göetz and Addison Wesley Professional. *Java Concurrency In Practice*, volume 39. Addison-Wesley Professional, 2006.

[17] Software Guide. *Java NIO*, volume 8. O'Reilly Media, Inc., 2009.

[18] Dominique Guinard, Vlad Trifa, Friedemann Mattern, and Erik Wilde. 5 From the Internet of Things to the Web of Things : Resource Oriented Architecture and Best Practices. *Architecting the Internet of Things*, pages 97–129, 2011.

[19] R Harper. *Inside the Smart Home*, volume 5. 2003.

[20] I Jacobson, M Christerson, P Jonsson, and G Overgaard. Object-Oriented Software Engineering: A Use Case Driven Approach. In *HarlowEssex England Addison*, volume 2640, page 552. Addison-Wesley Pub (Sd), 1992.

[21] Wolfgang Kastner, Georg Neugschwandtner, Stefan Soucek, and H. Michael Newman. Communication systems for building automation and control. In *Proceedings of the IEEE*, volume 93, pages 1178–1203, 2005.

[22] Kirk Knoernschild. *Java Design: Objects, UML, and Process.* Addison-Wesley Professional, 2001.

[23] C Larman. *Applying {UML} and Patterns: An Introduction to Object-Oriented Analysis and Design.* Prentice Hall, 2001.

[24] Sheng Liang. *Java Native Interface: Programmer's Guide and Specification, The.* Addison-Wesley Professional, 1999.

[25] Francesco Luciani. Analisi delle tecnologie di supporto alla domotica e alla localizzazione in un contesto di utenti mobili, 2006.

[26] Jeff McAffer, Paul VanderLei, and Simon Archer. *OSGi And Equinox.* Addison-Wesley Professional, 2010.

[27] Antonio Natali. *Costruire sistemi software: dai modelli al codice.* Esculapio, 2012.

[28] Luca Ricci. Mettiamo i sistemi domotici a confronto.

[29] Pietro Antonio Scarpino. Domotica e Building Automation.

[30] Robert Simmons. *Hardcore Java.* Number 0. O'Reilly Media, Inc., 2004.

[31] Luca Vetti Tagliati. *UML.* Tecniche Nuove, 2003.

[32] Deze Zeng, Song Guo, and Zixue Cheng. The Web of Things: A Survey (Invited Paper). *Journal of Communications*, 6(6):424–438, 2011.