Domotic Room
Smart City
Process Report Template

Nicola Casadei, Marco Benedetti, and Enrico Benini

Alma Mater Studiorum – University of Bologna
via Venezia 52, 47023 Cesena, Italy
{nicola.casadei4, marco.benedetti7, enrico.benini5}@studio.unibo.it

Table of Contents

1 Introduzione

Questo è il template di progetto del corso di smart city dell'università di Bologna. Di seguito saranno consultabile tutto il processo di analisi del progetto: modelli, problemi riscontrati e soluzioni adottate, interazione con l'ambiente, sensori utilizzati e il loro collegamento ...

Per qualsiasi dubbio in merito fare riferimento agli autori.

2 Visione

La visione che guida questo progetto è quella di raggiungere rapidamente l'ideale di città intelligente, quindi con un ambiente aumentato, capace di prendere decisioni e agire tempestivamente per far fronte a casi specifici e capace di comunicare direttamente con chi si trova immerso in esso, per facilitarne la vita di tutti i giorni.

In particolare vogliamo prepararci, imparare a modellare e costruire sistemi che si integreranno in questo contesto, visto l'andamento stesso del mercato che sta sempre più rendendo disponibili risorse di elaborazione e sensoristica a minor prezzo.

3 Obbiettivi

Lo scopo del progetto è quello di implementare concretamente un'applicazione di domotica. Affrontando quindi tutte le problematiche ad essa annesse e fornire una possibile soluzione a queste. Ci auguriamo che questa possa essere di spunto per applicazioni simili e che possa quindi favorirne lo sviluppo.

Sfruttando questo progetto, vogliamo esplorare e apprendere la teoria e i concetti affrontati nel corso di smart city. Quindi tutti gli aspetti riguardanti la gestione di sensori e input provenienti dall'ambiente esterno. Uscendo dalla, tipica, zona di confort classica dei sistemi software.

4 Requisiti

Si vuole monitorare lo stato ambientale di una stanza. In particolare si vogliono monitorare lo stato di: luce, temperatura e movimento, mantenendo la possibilità di aggiungere altre tipologie di sensori.

Il sistema dovrà dare all'utente la possibilità di inserire attraverso un'interfaccia web, per ogni valore misurato, un'apposito range che indichi i valori ammessi all'intero della stanza in modo che, se uno dei valori misurati non è conforme alle specifiche, venga indicata una notifica di allarme sull'interfaccia stessa. Questo con l'idea di simulare la possibilità di eseguire delle azioni collegate all'allarme (ad esempio, accensione delle luci o del riscaldamento)

L'utente potrà inoltre visualizzare all'interno del sito i valori misurati in tempo reale e il valore dei vari sensori nel tempo, potendone quindi consultare la storia.

4.1 Requisiti Non Funzionali

Aggiungiamo nei requsiti non funzionali tutte le proprietá che il sistema deve avere e che non sono state ufficialmente formalizzati.

- Logging: Possibilitá di controllare il fluire delle informazioni e delle operazioni svolte attraverso una qualsivoglia forma di logging
- Separazione dei compiti e indipendenza tra le parti: garantire i principi SOLID dove é possibile ed evitare che un qualsiasi componente sia strettamente legato ad un'altro.

5 Acquisto Hardware

Sfortunatamente il primo probelma che si e' incontrato in un progetto come il seguente e' stato la necessita' di acquistare la parte hardware del sistema che si andra' a costruire. Di consequenza si e' messo in atto un processo di ricerca dei sensori, cavi e quant'altro per riuscire a soddisfare i requisiti

5.1 Dispositivi di Computazione

Prima di tutto necessitiamo di un dispositivo in grado di computare i dati emessi dai vari sensori e che sia interamente programmabile. Nel corso abbiamo visto due possibilita' che hanno avuto molto successo recentemente:

– Arduino
– Raspberry Pi

Noi abbiamo scelto la seconda opzione perche' abbiamo piu' familiarita' con il dispositivo e perche' risulta piu' facile il riutilizzo dello stesso una volta terminato questo progetto.

Costo del dispositivo: 44,50 €

5.2 Sensori

Un'altra cosa fondamentale riguarda i sensori necessari per catturare i parametri richiesti. Abbiamo Quindi scelto i sequenti sensori

Parametri Ambientali	**Sensori**	**Costo**
Temperatura		
Luce		
Movimento		

Table 1. Sensor Table

5.3 Hardware Aggiuntivo

Hardware	**Costo**
Breadboard	
Wires	
Resistors	

Table 2. Addictional Hardware

6 Analisi dei Requisiti

6.1 Casi D'Uso

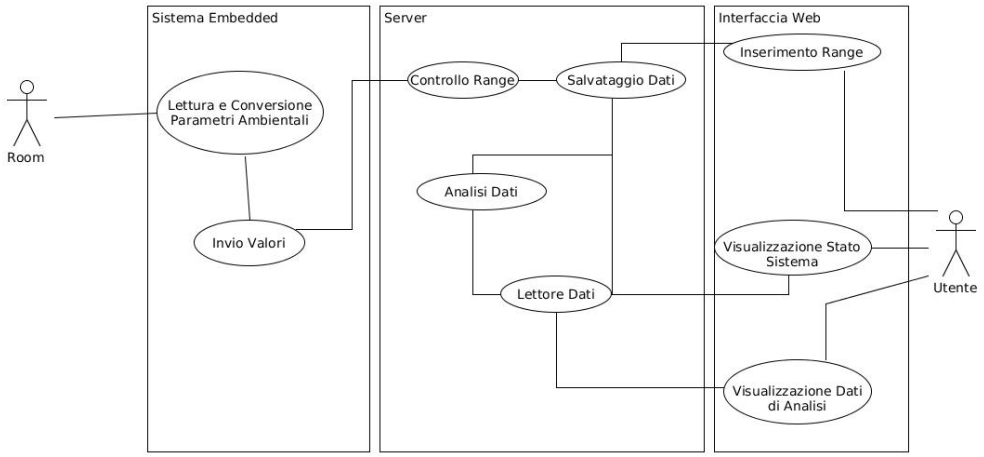

Fig. 1. Casi d'Uso

Nell'immagine sopra si possono vedere quali sono le macro operazioni principali effettuate dal sistema e le interazioni con l'esterno. In particolare gli attori che interagiscono con il sistema saranno:

- La stanza: con questo attore si intendono i vari parametri che si possono rilevare attraverso i sensori e che quindi saranno di input per il sistema.
- Utente: con questo attore rappresenta l'utente che puó interagire con il sistema.

Il sistema é stato volutamente suddiviso in tre parti distinte con l'idea di seguire un modello MVC dove peró la parte di modello non viene aggiornato solamente attraverso l'input inserito dall'utente, ma anche e soprattutto dall'input dei sensori. L'organizzazione hardware ha fortemente influito su questa suddivisione.

É inoltre possibile visualizzare le macro operazioni che devono essere effettuate e modellate dal sistema. Si vedano gli scenari di seguito per avere

una piú dettagliata visualizzazione dell'interazione tra le varie parti che lo schema sovrastante vuole rappresentare.

6.2 Scenario

In questa sezione verranno illustrati le principali modalitá di utilizzo del sistema. Gli scenari elencati di seguito riguardano l'utente e di conseguenza si prevede l'accesso da parte di questo all'interfaccia di input.

Si prevede che il sistema sia opportunamente configurato e settato a livello hardware, senza errori durante la fase di start up.

6.2.1 Inserimento Range

1. Attraverso un apposito menú l'utente é in grado di accedere alla funzionalitá di settaggio dei range associati ai parametri ambientali.
2. Il server sa giá i sensori che sono collegati al raspberry appena questi inviano qualche dato e quindi l'utente é in grado di visualizzare i controlli relativi ad ogni tipologia di sensore attualmente connesso. Conseguentemente l'utente é in grado di modificare tali intervalli.
3. Al termine della modifica degli intervalli l'utente dovrá confermare le modifiche attraverso un'apposito pulsante.
4. Il sistema mostra un messaggio di conferma o di errore.

6.2.2 Visualizzazione Stato Realtime

All'accesso del sistema l'utente visualizza lo stato realtime dei valori dei sensori ed eventuali notifiche:

- Se i valori vanno oltre gli intervalli correnti.
- Sullo stato dei sensori, se sono attivi al momento oppure no.

In questa modalitá l'utente non puó effettuare alcuna operazione.

6.2.3 Visualizzazione Dati

Attraverso un apposito menú l'utente é in grado di accedere alla visualizzazione dei dati storici dei vari sensori.

6.3 Modello del Dominio

In questa sezione vogliamo cercare di modellare le entitá inserite all'interno dei requisiti senza fare riferimento alla parte tecnologica e alla parte hardware. In questo modo siamo in grado di decidere noi le interfaccie con cui vogliamo lavorare e costruire la parte software aumentando il disaccoppiamento con la parte fisica e quindi aumentando anche la possibilitá di utilizzare volendo lo stesso software su diverse configurazioni. In questo progetto ad ogni modo si utilizzerá solamente la configurazione che trovate descritta in questo report.

Suddivisione del Sistema: visto che é emerso giá dai casi d'uso la separazione del sistema in varie parti, ci é sembrato giusto iniziare a modellare dividendolo fin da subito in modo da:

- semplificare il processo
- suddividere il lavoro di implementazione successivamente tra i membri del gruppo.

In particolare le parti individuate sono tre:

- Sistema Embedded: che nel nostro caso si occuperá di catturare, convertire e inviare i dati alle altre parti
- Server: Sará la parte che riceve i dati e si occupa di effettuare le varie elaborazioni come ad esempio, il salvataggio dei dati, il calcolo delle statistiche e il controllo dei range. Infine questa parte dovrá rendere disponibili i risultati alla parte successiva.
- Web site: parte che, prende i risultati dalle parti precedenti e li mostra all'utente rispettando i casi d'uso predecenti.

Chiaramente in questa fase tutto viene semplificato e ridotto a poche entitá. Questo peró non significa che, ogni entitá presente negli schemi sottostanti, non si riveli essere poi a sua volta un sottosistema piú complesso.

Lo scopo di questa fase é appunto quella ri riflettere, in un primo modello, i requisiti. Poi da questo modello iniziare a sviluppare il progetto cercando di mantenere coerenza nelle interfaccie principali che indicano le interazioni piú importanti.

6.3.1 Sistema Embedded

Per quanto riguarda il sistema embedded é necessario prima effettuare delle indagini su come interagire e comunicare con i sensori in modo da modellare adeguatamente il tutto e quindi assicurarsi che in seguito sará semplice riuscire a integrare il tutto con la tecnologia che avremo intenzione di utilizzare. Quindi questa é una piccola eccezione che é necessario fare a questo livello, anche se in particolare si fa riferimento a un paradigma piú che ad una tecnologia specifica.

Il modello di interrogazione dei sensori é a polling di consequenza il nostro modello dovrá riflettere questa modalitá di interazione. Sfortunatamente questo implica che a livello di modello giá **un'entitá attiva** che si occupa di reperire i valori visto, che in una modalitá a polling devo esplicitamente chiedere ai sensori i valori.

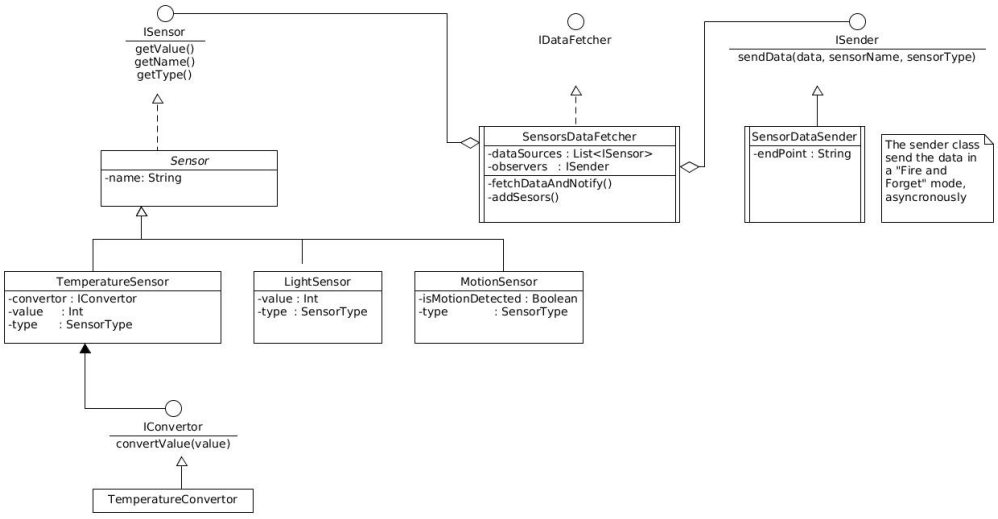

Fig. 2. Sistema Embedded, Struttura

Struttura Dalla struttura si possono subito individuare le entitá principali che sono presenti nei requisiti, in particolare la presenza di una specifica gerarchia per i sensori che condividono la stessa interfaccia che consente agilmente di ottenere il valore corrente del sensore. Sono stati inseriti sola-

mente i sensori citati nei requisiti, ma si puó facilmente intuire come qualsiasi tipo di sensore sia facilmente modellabile secondo questa struttura.

Si noti inoltre come viene anche inserita l'entitá IConvertor che si occoperá di convertire il valore di uno specifico sensore in un'unitá piú consona per la sua gestione. Chiaramente questo viene affrontato fin da questo livello perché si immagina l'operazione di conversione come un'operazione quasi istantanea e necessaria.

Infine sono presenti le entitá che si occupano, attivamente, di interrogare i sensori ogni intervallo di tempo predefinito e quindi inviarli altrove, In questo caso é stato tutto ridotto ad un singolo endpoint, anche se poi possono essere facilmente piú di uno. Come si puó visionare nel commento, l'invio dei dati avviene con una metodologia di tipo *fire and forget*, quindi non avvengono reinvii dei dati e vengono ignorati eventuali errori. Chiaramente tutto questo é dovuto all'idea che i cicli di invio siano abbastanza brevi da potersi permettere eventuali perdite.

Interazione Prima di iniziare si vuole evidenziare come viene riportato solamente un diagramma di interazione perché é presente solamente un'entitá attiva, ma nel futuro potrebbero esserci piú task e quindi saranno necessari piú diagrammi dell'interazione.

Nello schema di interazione vengono evidenziate le varie fasi del Sistema, in particolare la fase iniziale di setup dove, conoscendo quali sensori sono presenti, questi vengono aggiunti nella memoria dell'entitá principale in modo che poi in seguito siano facilmente interrogabili.

Terminata la fase di *Init* inizia il loop infinito che appunto aspetta inizialmente un piccolo lasso di tempo e poi va a interrodage iterativamente tutti i sensori che sono stati aggiunti precedentemente, raccoglie i dati e interroga asincronamente l'entitá di invio, per poi ricominciare il ciclo stesso.

Comportamento Nel diagramma del comportamento semplicemente viene riportato quanto é stato precedentemente discusso semplicemente attraverso una state machine.

6.3.2 Server

La parte server é quella piú importante di tutto il sistema in quanto é quella che concentra tutta la logica applicativa del sistema stesso.

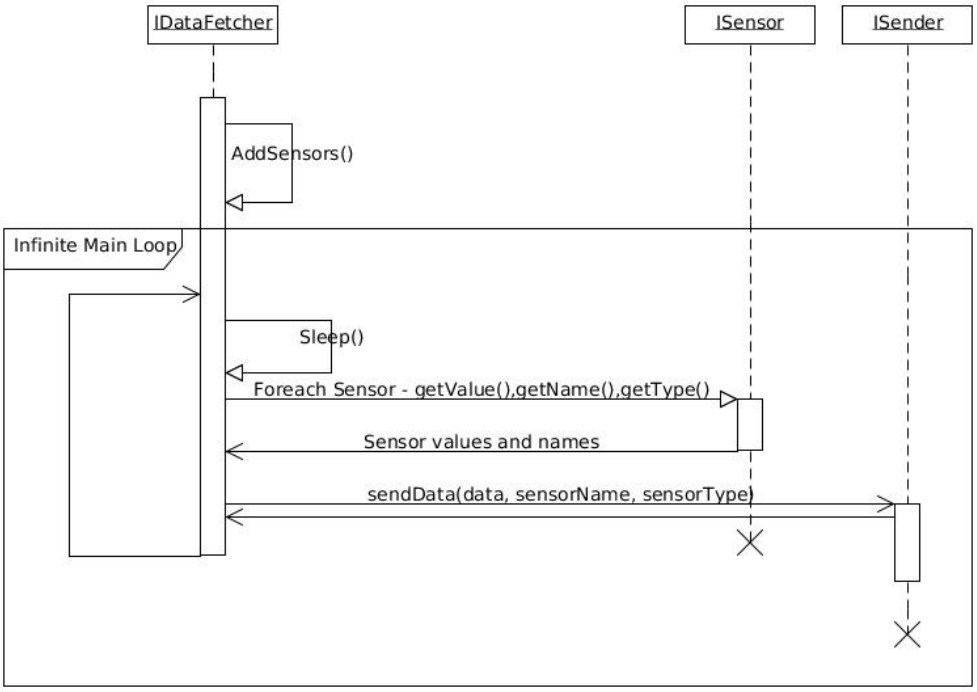

Fig. 3. Sistema Embedded, Interazione

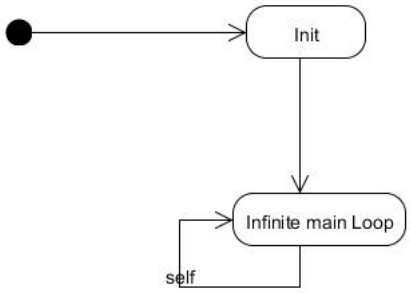

Fig. 4. Sistema Embedded, Comportamento

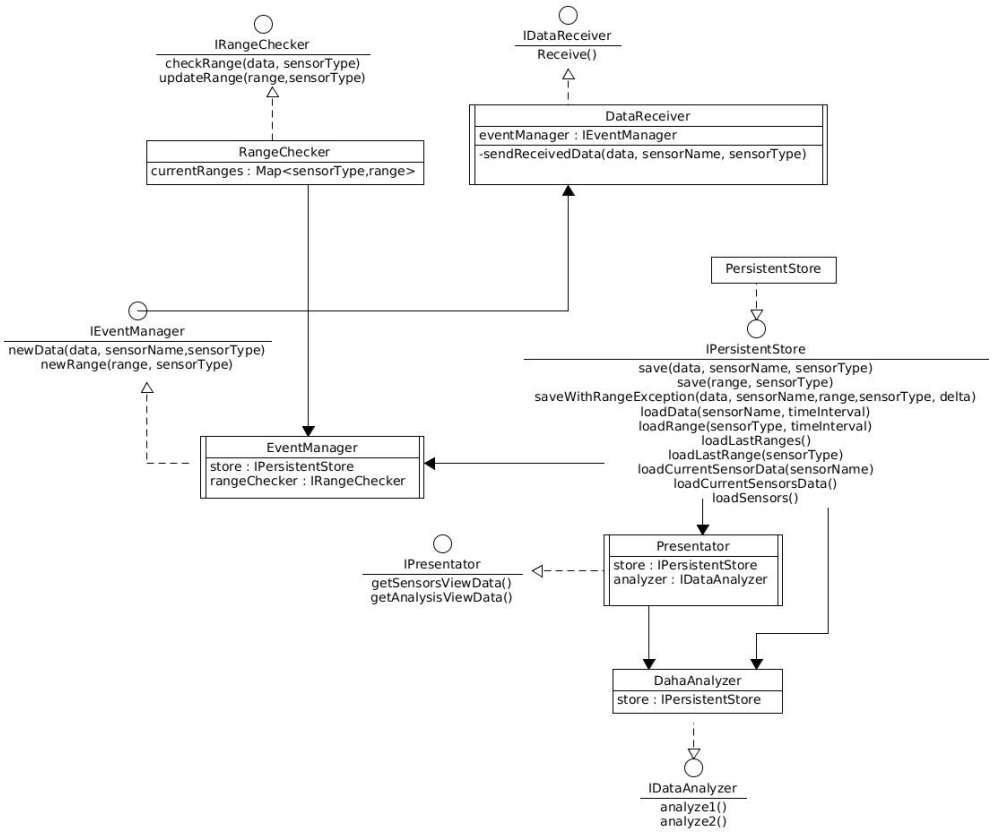

Fig. 5. Server, Struttura

Struttura Le entitá principali di questo schema sono:

- IPersistentStore, che si occuperá di salvare opportunamente i dati provenienti dai sensori e dall'utente
- IDataReceiver, che sará sempre in ascolto di ogni messaggio proveniente dal sistema embedded e quindi notificherá opportunamente il sistema ad ogni nuovo arrivo
- IPresentator che si occuperá di ottenere i dati necessari per le viste da mostrare all'utente quanto una nuova richiesta viene inoltrata.

Chiaramente ogniuna di queste entitá é in parte citata nei casi d'uso.

Si veda i diagrammi dell'interazione per i dettagli di come avviene la comunicazione di tutte queste entitá al fronte di garantire il funzionamento e il soddisfacimento dei requisiti.

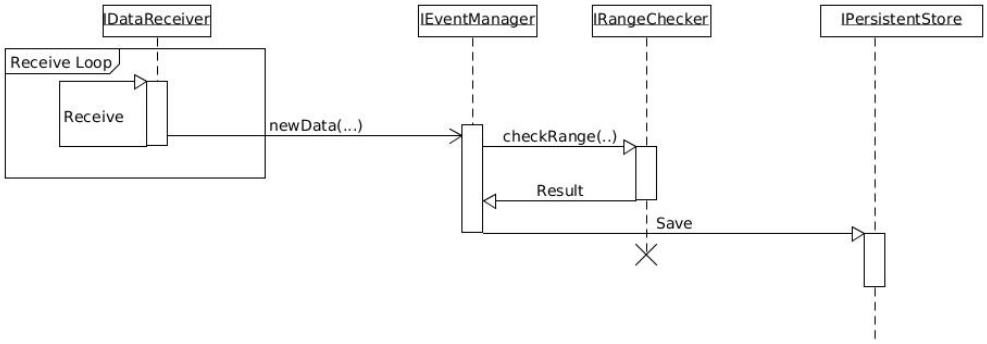

Fig. 6. Server, Interazione, Dati dai Sensori

Interazione Abbiamo deciso di omettere la parte di inizializzazione del sistema in questa fase anche perché sará meglio defita nell'analisi del problema dovendo aggiungere entitá che non sono direttamente correlate con i requisiti principali ma con requisiti non funzionali.

Nello schema sovrastante si puó osservare l'interazione nel caso i sensori emettano un nuovo valore. Si vuole porre particolare enfasi su come l'entitá che riceve i dati rimane sempre in ascolto di nuovi arrivi delegando, tramite

una chiamata asincrona alle altre entitá il compito di salvare i dati appena arrivati.

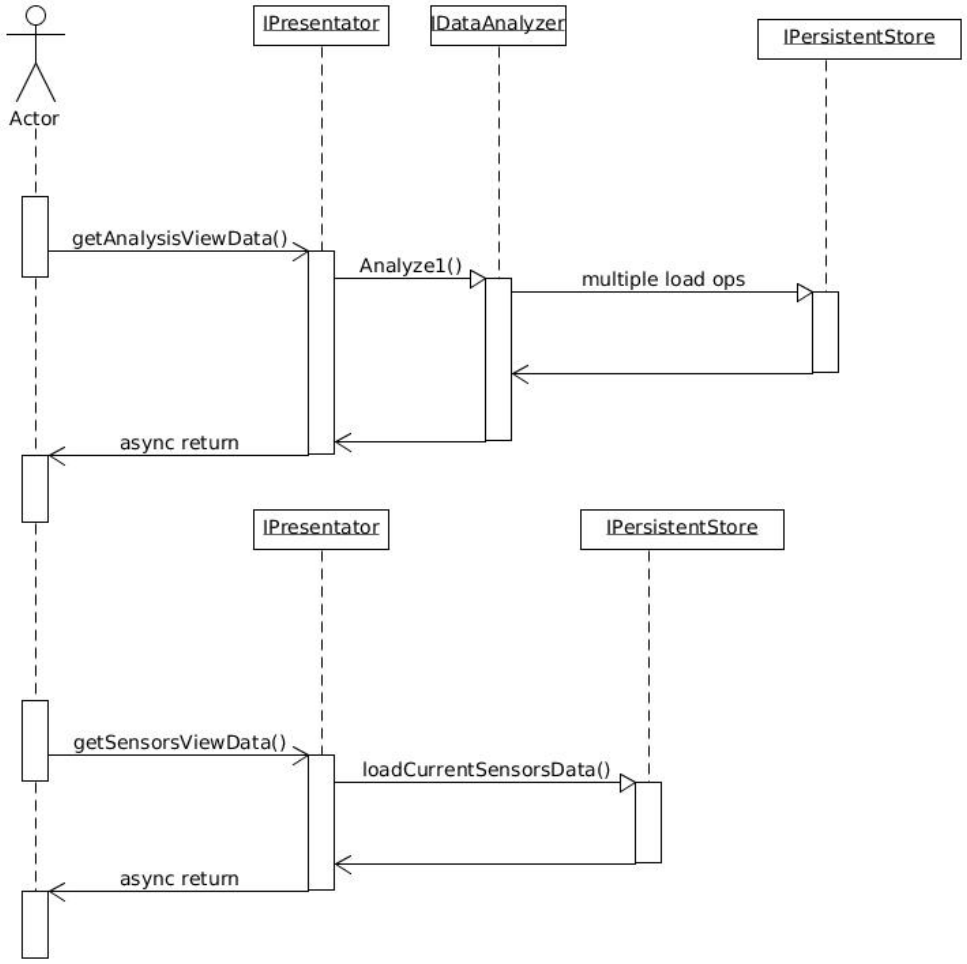

Fig. 7. Server, Interazione, Richiesta Visualizzazione Dati

In quest'altro schema dell'interazione si puó osservare che entitá vengono conivolte a fronte di una chiamata di visualizzazione dati da parte dell'utente. Anche in questo caso la chiamata iniziale é stata effettuata in

maniera asincrona in modo da lasciare libero il sistema di rispondere ad eventuali altre richieste e gestirle concorrentemente.

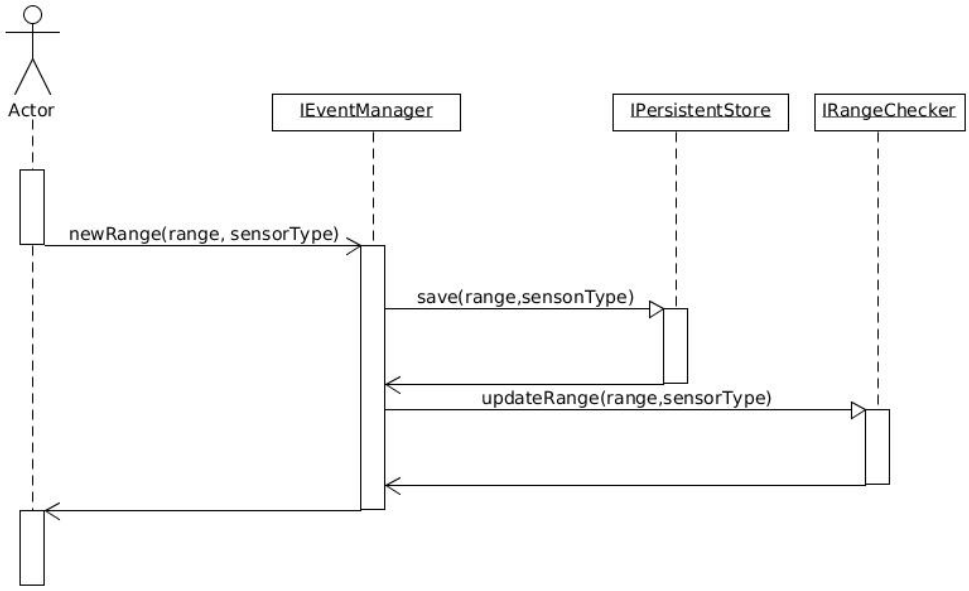

Fig. 8. Server, Interazione, Nuovo Range

In questo caso invece viene illustata l'interazione per quanto riguarda la richiesta da parte dell'utente dell'inserimento di un nuovo range. Si vede come la chiamata questa volta arriva all'entitá che gestisce gli eventi essendo questo effettivamente un caso incui avviene un cambiamento nei dati e quindi tutto va gestito appropriatamente per evitare i conflitti.

Comportamento In questa fase si mostrano le entitá che sono effettivamente attive e che quindi operano cambiamenti sul sistema. Si é deciso di omettere tutte le entitá che vengono solamente chiamate ed effettuano una unica operazione al fine di evitare di formalizzare troppe cose a questo livello.

Si vuole far notare inoltre come tutte queste entitá abbiano effettivamente uno stato di attesa di un qualche messaggio o evento. Questo aspetto puó essere molto importante in seguito nelle scelte per la progettazione.

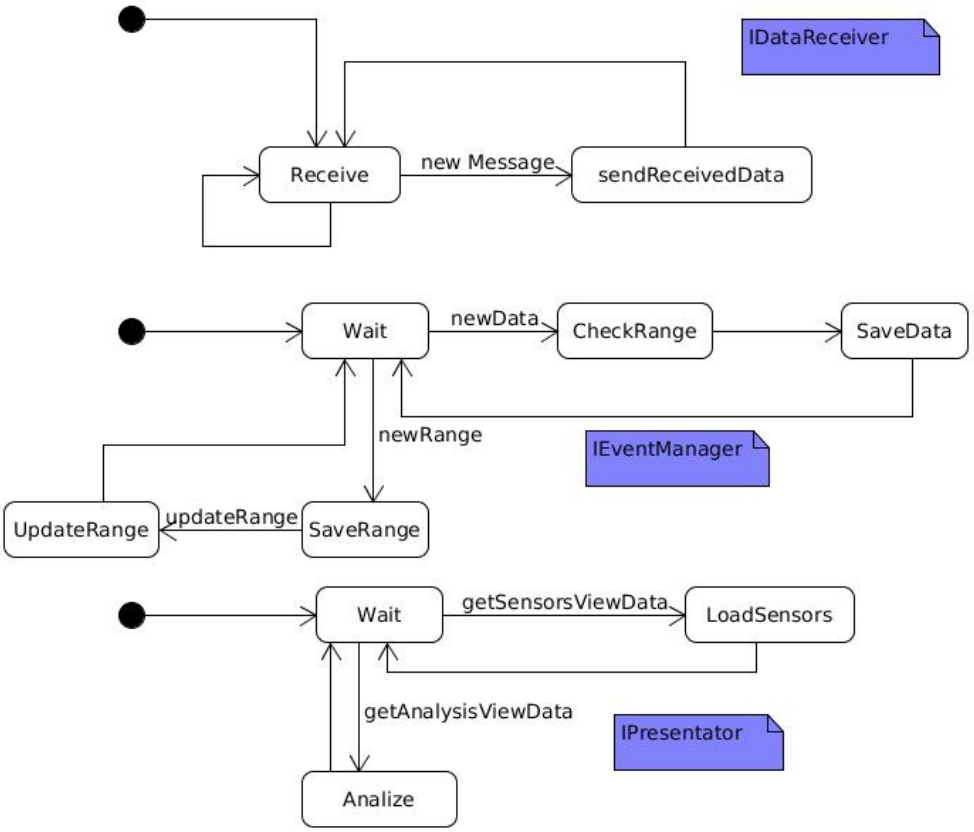

Fig. 9. Server, Comportamento

6.3.3 Web Site

Dopo un'attenta analisi abbiamo ritenuto superflua la necessitá di modellare e analizzare la parte di visualizzazione in quanto non é effettivamente presente nessuna business logic ma solamente la parte che effettua le richieste di dati alle parti precedenti e visualizza tutto su schermo.

La documentazione relativa a questa parte si limiterá solamente a come utilizzare poi il software attraverso l'interfaccia.

Possiamo peró pensare di definire i link a cui l'interfaccia dovrá rispondere appropriatamente attraverso una pagina web. Questo ci consente di

impostare comunque dei test che esulano dal contenuto della pagina, ma ci garantiscono che questa sia effettivamente online in maniera automatica.

gli URL scelti sono:

- *http://...../DomoticRoom/Status*
- *http://...../DomoticRoom/NewRange*
- *http://...../DomoticRoom/Analysis*

6.4 Piani di Test

In questa fase saranno presenti i piani di test che vengono costruiti sulla base dei modelli precedenti, ancora privi di implementazione. Ció é possibile perché a questo punto abbiamo giá individuato le interfaccie delle entitiá principali e le loro operazioni pubbliche.

Chiaramente a questo punto la maggior parte dei test non possono funzionare perché manca l'implementazione. Anche in questa fase si divide il tutto sulle 3 parti precedenti.

6.4.1 Sistema Embedded

6.4.2 Server

6.4.3 Web Site

7 Analisi del Problema

Assunzioni Prima di iniziare l'analisi del problema abbiamo ritenuto neccessario effettuare delle assuzioni riguardanti al paradigma che ci consentirebbe di effettuare un prodotto di qualitá migliore e con meno sforzi.

Il paradigma di programmazione di riferimento é il *reactive programming* perché la concezione di flusso di dati é proprio quello che ci interessa modellare in quanto anche nel nostro sistema sará presente un flusso di dati dal client al server. Per la comunicazione attraverso la rete questo ci consente di sfruttare chiamate asincrore aumentando il disaccoppiamento tra client e server.

Per questo abbiamo deciso di utilizzare per i dati un database NoSQL per via dell'estremo dinamismo, in quanto ci consente di aggiungere dei

campi anche in seguito e miglioramento di performance nell'accesso a dati che normalmente richiederebbero dei join.

Per ulteriori informazioni piú dettagliate sul paradigma si rimanda ad un'altra sede.

7.1 Architettura Logica

Per la costruzione dell'architettura logica ci si baserá ovviamente sulla precedente analisi dei requisiti in modo da mantenere il contatto con questi e non rischiare di uscire fuori specifica. Anche in questo caso si andrá separare l'analisi in due parti:

- Sistema Embedded
- Server e Website

La parte del website é stata incorpata direttamente nella parte server proprio per la sua assenza di business logic.

Modellazione Reactive Prima di iniziare a costruire la modellazione sulla base del paradigma appena citato é necessario capire come fare a modellare lo stream stesso all'interno della nostra architettura logica, conseguentemente si é deciso di utilizzare i *marable diagrams*. Questi sono una rappresentazione di come il flusso di dati nel tempo avviene e quali tipologie di trasformazioni consentono di effettuare.

Con questi schemi ci viene concesso ancora una volta di concentrarci prima sul "cosa" e non sul "come" che invece viene delegata alla parte progettuale, dove si cercherá effettivamnete di implementare il risultato finale dell'analisi

7.1.1 Sistema EmbeddedAlla luce del paradigma di programmazione

che si é deciso di adottare sono stati effettuati dei cambiamenti anche alla struttura che é stata esposta precedentemente proprio perché ci sono state fornite da questa assunzione nuovi livelli di astrazione, primo tra tutti il concetto di **Stream/Flusso**. Quindi in questa fase abbiamo ritenuto molto utile iniziare a visualizzare effettivamente questi flussi tramite un'apposito diagramma.

In particolare si puo' notare come in questo diagramma il compito di convertire i valori di un sensore sia stato disaccoppiato dal sensore stesso

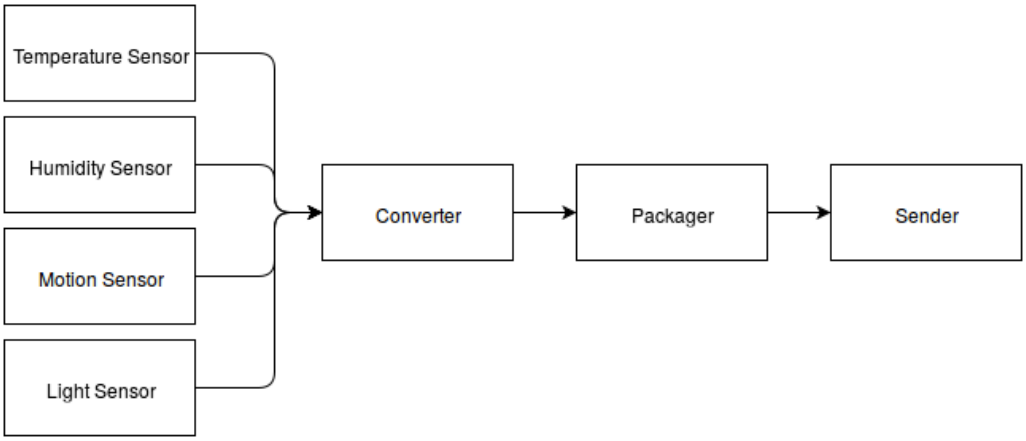

Fig. 10. Diagramma di flusso, per sensore, dei dati

che quindi si deve solamente occupare di invitare i dati all'interno del flusso. É stato inoltre aggiunto anche una nuova entitá *packager* che si occuperá di formattare i valori secondo il protocollo che deve essere definito tra client e server per l'apposita comunicazione. In particolare ogni dato proveniente da un sensore avrá un suo formato per via: della differenziazione dei dati, della tempistica di rilevazione che puó anche risultare diversa e dell'idea percui la comunicazione in ogni caso debba essere asincrona.

Sulla base di questa ultima frase si vuole sottolineare che ogni sensore quindi avrá un suo flusso asincrono, ma che in quest'immagine si é deciso di condensare in un'unica immagine per comoditá.

Di seguito si indica un primo marable diagram che mostra le varie trasformazioni che verranno applicate. Questo serve soprattutto per iniziare a capire poi come interpretare eventuali altri schemi come questo.

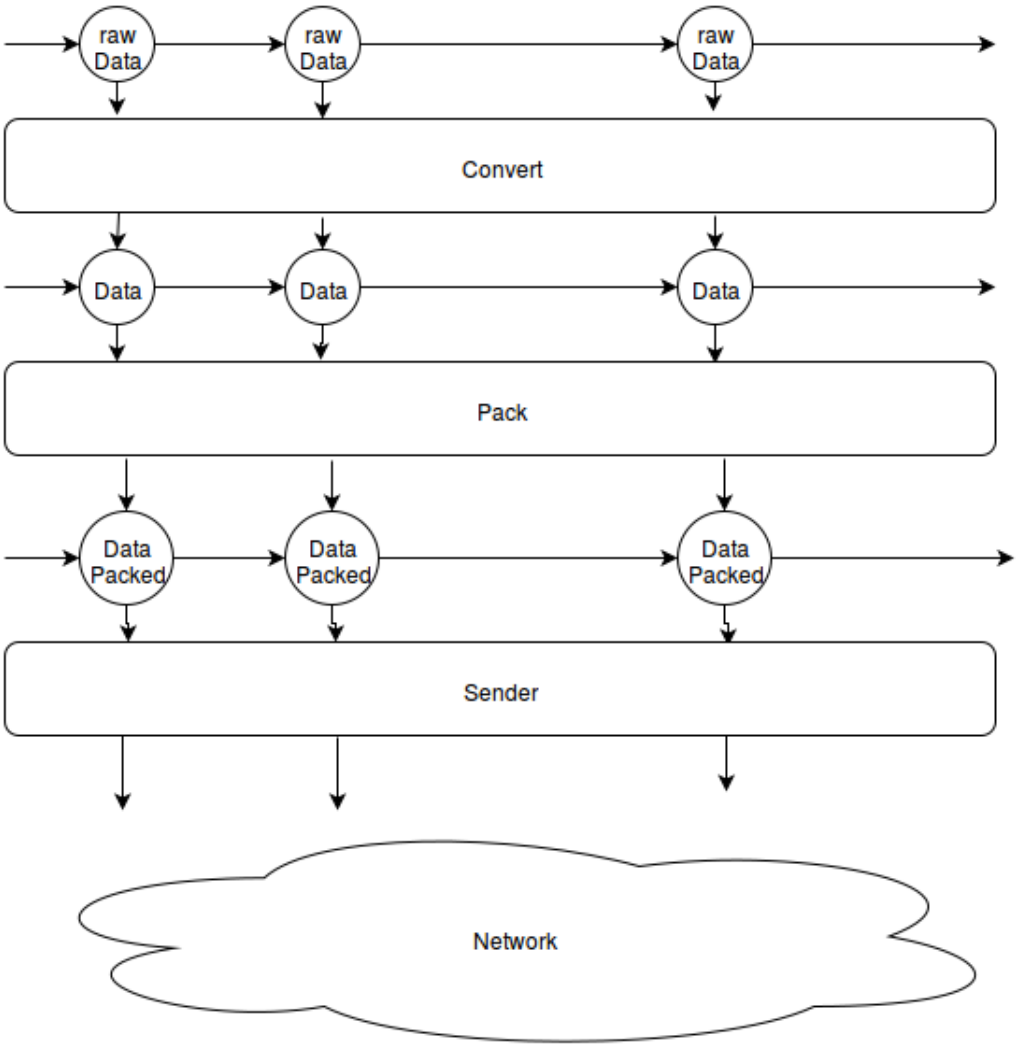

Fig. 11. Marable diagram dei singoli dati proveniente da una sorgente

Struttura Chiaramente la struttura, che si basa sul precedente step di analisi dei requisiti, risulta cambiata anche alla luce delle assunzioni effettuate e quindi si possono notare l'introduzione di alcune entitá che servono per la gestione del sistema embedded e per impostare i flussi di dati provenienti dai sensori

Interazione La parte di interazione viene drasticamente modificata proprio perché l'assunzione del paradigma reactive risulta principalmente improntato su questo, in particolare é possibile condensare in meno codice, e piú dichiarativo. Vengono in ogni caso mostrati le varie operazioni effettuate sullo stream per mantenere il contatto con quanto mostrato attraverso i marable diagrams. Particolare attenzione va posta poi sulla parte che converte da *procedure calls* a *reactive streams*.

Comportamento

7.2 Gap di Astrazione

In questa sezione aggiungeremo tutti i tipi di astrazioni che sono richieste per affrontare il progetto e che non sono direttamente fruibili attraverso la tecnologia di riferimento.

1. Web Server: Vista la necessitá di comunicare attraverso la rete é necessario che si utilizzi un paradigma a message-passing o attraverso chiamate asincrone, soprattutto per la comunicazione che avviene tra il raspberry e il server.
2. Continuous Integration, Testing and collaborative source control: Lavorando in gruppo sullo stesso repository é necessario impostare il lavoro affinché sia possibile effettuare le modifiche in maniera indipendente gli uni dagli altri e allo stesso modo sia possibile controllare automaticamente che i test predisposti e le modifiche effettuate siano coerenti con le specifiche e che il building del progetto sia in ogni caso garantito.
3. Paradigmi Eterogenei: si é deciso di utilizzare dei paradigmi diversi dal OOP classico e questo puó portare a problematiche di utilizzo, per via dell'inesperienza.Tutto questo puó causare dei ritardi durante la realizzazione.

7.3 Analisi dei Rischi

In questa sezione vanno elencati invece i rischi che si possono intraprendere in un progetto di questo tipo e quindi formalizzarli fin da subito, prima di eseguire l'effettiva realizzazione dello stesso in modo da poterli affontare e discutere preventivamente per essere pronti se questi si verificano.

- Aumento dei Costi: é necessario porre particolare attenzione alla struttura hardware del sistema e di come i singoli componenti vanno ad interconnettersi assieme per evitare che si debbano affrontare dei costi aggiuntivi, una volta che il progetto é giá avviato, causati da una qualche mancanza o per via di un'estensione del progetto in corso d'opera.
- Quantitativo della memoria: l'adozione di un database NoSQL porta con se un certo livello di ridondanza e quindi questo puó portare ad un aumento di utilizzo della memoria e di spazio. Il tutto va valutato accuratamente durante il progetto, magari attraverso una serie di prove in base a come é stutturato il dato inizialmente. Se il rischio quindi é reale e quanto é critico.
- Integrazione tra le tecnologie: un possibile rischio riguarda la difficoltá nell'integrare tutte le tecnologie che devono essere sfruttate nel progetto per riuscire a colmare l'abstraction gap e quindi riuscire a completare il progetto attraverso l'analisi appena completata.

8 Work Plan

Il piano di lavoro che abbiamo intenzione di attuare si divide in varie fasi:

1. Controllo del Funzionamento Hardware: la prima cosa che é necessario fare é quella di controllare che tutta la sensoristica sia effettivamente compatibile e che la parte sistemistica sia assemblabile e funzionante.
2. Ricerca e Analisi Problematiche Viste le problematiche sollevate precedentemente nell'abstraction gap é necessario soffermarsi prima di partire con il progetto per andare ad individuare i tools che possono coprire queste mancanze e quindi avere un'impatto significativo sulla realizzazione del progetto stesso. Queste possono cambiare drasticamente anche la realizzazione del progetto stesso che peró deve rimanere fedele all'analisi del problema.

3. Modello del Progetto: contruzione del modello del progetto alla luce delle considerazioni precedenti
4. Impostazione dell'Environment: setup di tutti i tools precedenti e controllo del loro corretto funzionamento
5. Suddivisione dei Compiti: quando tutto é formalizzato adeguatamente si possono suddividere i vari compiti e quindi parallelizzare il lavoro
6. Cicli di Feedback: é molto utile ai fini della realizzazione del progetto, organizzare dei meeting periodici al fine di effettuare un check sullo stato dei lavori e quindi controllare eventuali incongruenze, discutere i problemi, rivedere i modelli precedenti e altro
7. Integrazione: Integrare tutti i progetti assieme per costruire tutto il sistema assicurandosi che sia corretta l'interazione tra le parti
8. Testing: La parte di testing deve essere sviluppata assieme al codice stesso se possibile sulla base del modello in modo da arrivare alle fasi finali da poter automaticamente capire cosa funziona e cosa no.
9. Deploy: Per la parte di deploy non si porrá particolare attenzione in questa prima versione perché non é effettivamente richiesta una particolare complessitá nel deploy su piú macchine. Comunque si tratta di un'aspetto che puó essere affrontato anche a progetto finito e vedere di apportare le modifiche richieste per realizzare un deploy piú articolato.

8.1 Strumenti e Framework

In questa sottosezione vengono illustrati i tools utilizzati durante questo progetto e che servono per cercare di colmare l'abstraction gap e per il supporto alla gestione del progetto stesso.

- Umlet: Questo tool é stato utilizzato per creare gli schemi UML che realizzano i modelli di tutto il progetto.
- Play Framework and Akka: framework creato da Typesafe che consente di gestire un web server con REST API e di gestire autonomamente le chiamate in maniera asincrona. Akka é tutta l'infrastruttura sottostante che consente di gestire tutto questo. Per ulteriori informazioni basta cercare nel web.
- Activator Template: tool associato al framework precedente che consente di gestire le proprie applicazioni secondo degli standard affermati e delle configurazioni di defalut in modo da velocizzare lo startup di tutto il progetto.

- Scala Build Tool: anche se é stato pensato per progetti in scala questo strumento funziona correttamnete anche per java e consente di gestire tutte le dipendenze, eventuali librerie aggiuntive e configurazioni, molti dei progetti precedenti vengono inseriti all'interno del sistema attraverso questo.
- Bootstrap: ...
- New Relic: ...
- Quickcheck: ...
- rxJava: ...

9 Project

9.1 Structure

9.2 Interaction

9.3 Behavior

10 Implementation

11 Testing

12 Deployment

13 Maintenance

References