Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Telemonitoraggio di parametri ambientali critici in ambiente OSGi

Antonio Coronato
Email : antonio.coronato@na.icar.cnr.it

Giovanni Paragliola
Email: giovani.paragliola85@gmail.com

RT-ICAR-NA-2012-06 **06 2012**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, I-80131 Napoli, Tel: +39-0816139508, Fax: +39-0816139531, e-mail: napoli@icar.cnr.it, URL: *www.na.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Telemonitoraggio di parametri ambientali critici in ambiente OSGi

Antonio Coronato
Email : antonio.coronato@na.icar.cnr.it

Giovanni Paragliola
Email: giovani.paragliola85@gmail.com

Rapporto Tecnico N.: **RT-ICAR-NA-2012-05**

Data: **06 2012**

---

[1] note titolo
[2] affiliazione

L'evoluzione delle tecnologie per la società dell'informazione suggerisce uno scenario in cui gli utenti possono disporre di dispositivi elettronici ed informatici tanto piccoli e "mimetizzati" negli oggetti di uso comune (da cui il termine "pervasivi") in grado di scambiare informazioni attraverso reti non strutturate. I sistemi pervasivi devono soddisfare stringenti specifiche in termini di bassi consumi di potenza, caratteristiche elettromeccaniche, ergonomia, anche in vista della portabilità o addirittura vestibilità dei dispositivi.Nell'ambito di questo lavoro ci si concentrerà sulla sviluppo di una sistema di monitoraggio per parametri ambientali medianti l'ausilio di sensori basati sullo standard ZigBee. Utilizzando inoltre tecnologie emergenti, come ad esempio OSGi, si cercherà di fare in modo che almeno una parte del sistema sia introducibile all'interno di un ambiente di tipo Smart-Home o Smart-Office.

Introduzione

I sistemi software/hardware che rendono un ambiente "intelligente" possono essere caratterizzati da due attributi fondamentali:
1)Ubiquità : l'interazione con tali sistemi è disponibile dovunque l'utente ne abbia bisogno;
2)Trasparenza : tali sistemi sono non intrusivi e risultano essere integrati negli ambienti della vita quotidiana..
Con il nascere di tali ambienti, la potenza elaborativa si sposta, quindi, dal tradizionale desktop all'ambiente circostante. Questo introduce nuove difficoltà su tutti i macro livelli di un sistema tecnologico, dall'hardware fino al software. Tali difficoltà derivano soprattutto dal fatto che tali ambiente, e quindi i sistemi tecnologici di cui sono costituiti, differentemente dai sistemi tradizionali, sono soggetti ad un utilizzo più intensivo, meno prevedibile, non sempre fisicamente localizzabile e difficilmente programmabile.
Di seguito saranno descritti alcuni requisiti fortemente caratterizzanti e tipici di un ambiente "intelligente". Tra i requisiti fondamentali individuabili possiamo citare:
1)**Adattabile**. L'ambiente deve essere flessibile e autonomo in risposta ai cambiamenti delle richieste dell'utente e delle condizioni operative 2) **Disponibile e affidabile.** Le funzionalità offerte dall'ambiente devono essere sempre disponibili e quindi non sono ammissibili situazioni di arresto, spegnimento o riavvio. Inoltre l'ambiente deve garantire la persistenza dei dati e la tolleranza ai guasti 3)**Integrato.** L'ambiente deve essere inserito come parte integrante del nostro mondo offrendo sia capacità per "sentire" ciò che accade sia per "agire" adeguatamente in risposta.
4)***Intenzionale*.** Le persone devono essere in grado di formulare richieste esprimendo delle intenzioni e nominando degli oggetti o delle funzioni. 5)**Nomadico**. L'ambiente deve permettere agli utenti e a qualsiasi dispositivo di potersi muovere liberamente secondo le necessità
6)**Pervasivo**. L'ambiente deve essere ovunque, ossia i componenti tecnologici facenti parte dell'ambiente pervadono qualsiasi entità che può interagire con l'uomo, con l'ambiente stesso o con altre entità.
Un ambiente "intelligente" deve essere quindi "*context-aware*" (consapevole del contesto), ossia capace di "sentire" le variazioni di contesto e "agire" di conseguenza. Tuttavia, occorre fornire una definizione formale di ciò che si intende per contesto e consapevolezza del contesto riferiti a tali ambienti.
Il concetto di "*Context-awareness"* è stato per la prima volta introdotto e definito da Schilit e Theimer come "*la capacità delle applicazioni di un utente mobile di scoprire e reagire ai cambiamenti nell'ambiente in cui sono situati*". L'uso del termine contesto è riferito all'ubicazione, all'identità delle persone e cose nelle vicinanze di una locazione e ai cambiamenti subiti da tali oggetti.

Tecnologia Abilitanti

I. OSGi

L'Open Services Gateway Initiative (OSGi) è un consorzio nato nel 1999, raggruppando le più importanti società impegnate nel campo dell'*Home Automation*, con l'intento di definire le specifiche SOA per la diffusione e l'amministrazione dei servizi tipici di un residential gateway (RG).
L'architettura risultante ha preso il nome di **OSGi Service Platform** e, dalla domotica, si è ora diffusa in molteplici campi: dall'automazione industriale, alla telematica, passando per Internet. La tecnologia OSGi trova ora applicazione nei veicoli, nei cellulari di ultima generazione, oltre che nel software, sia desktop (ad es. Eclipse usa OSGi per risolvere il problema dell'aggiornamento automatico dei plug-in) che server (le ultime versioni dei più diffusi *application server* si basano su OSGi: JBoss, IBM WebSphere, BEA WebLogic).
In sostanza, OSGi è una piattaforma *Java-based* che, secondo un approccio *microkernel* a plug-in, fornisce le specifiche per sviluppare applicazioni che implementano servizi, permettendo di registrarne di nuovi, di aggiornare o rimuovere gli esistenti *on the fly*, senza compromettere cioè l'operatività della macchina, su cui la piattaforma sta girando. Un pregio della piattaforma sta nell'**interoperabilità**: il modello di cooperazione tra componenti prevede la possibilità di ricercare, individuare ed usare in maniera condivisa i servizi forniti da diverse applicazioni nell'ambito della stessa *virtual machine*, con conseguenti vantaggi in termini di prestazioni e consumo delle risorse.
Tecnicamente, le specifiche della piattaforma OSGi introducono il concetto di **servizio**, inteso come semplice interfaccia, e di **bundle** (componente), inteso come archivio (JAR) contenente l'implementazione dei servizi e le direttive di distribuzione ed installazione all'interno della piattaforma, oltre che le dipendenze da altri package e servizi. Il framework OSGi è di fatto l'ambiente di esecuzione dei *bundle.*

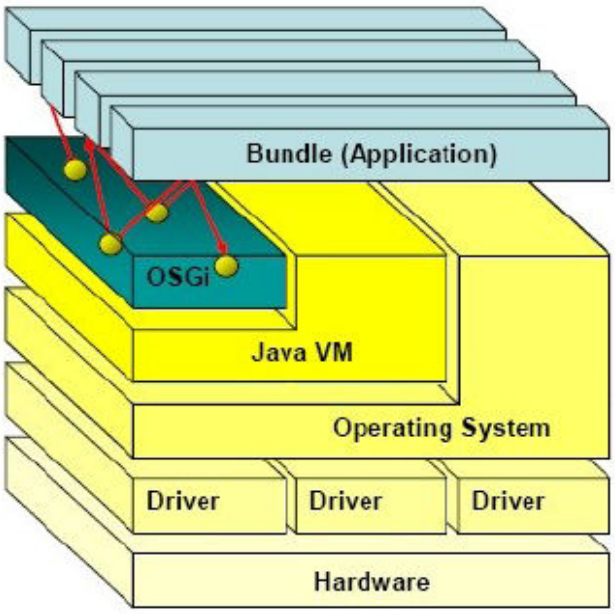

Figura 1. Modello Architetturale OSGi

Come già anticipato un bundle è un componente OSGi. Esso può essere installato, disinstallato , avviato o fermato. Una volta avviato esso può definire servizi OSGi o usare servizi OSGi definiti da altri bundle. Può inoltre esportare pacchetti java o importare i pacchetti esportati da altri bundles. Strutturalmente esso non è altro che un file JAR al cui interno ci sono:
1) **Un insieme di risorse** che servono per implementare i servizi messi a disposizione. Essenzialmente tali risorse sono **classi Java** o files di varia natura (immagini, HTML,…).
2)**Un file Manifest.mf** che è una lista di attributi associati al bundle.

II Modbus

II protocollo MODBUS definisce il formato e la modalità di comunicazione tra un "master" che gestisce il sistema e uno o più "slave" che rispondono alle interrogazioni del master.
In una transazione è sempre il master ad iniziare la comunicazione , lo slave può solo rispondere alle richieste , di conseguenza non può inviare messaggi se non sotto richiesta esplicite da parte del nodo master .
La comunicazione assume diverse molteplicità a secondo del tipo di dispositivi interessati nel caso del master la comunicazione può essere :
1)Unicast : il master indirizza un singolo slave. Dopo aver ricevuto e processato la richiesta lo slave restituisce al master un messaggio di risposta;
2)Broadcast :il master invia una richiesta a tutti i nodi slave. Gli slave, in questo caso, non gli restituiscono alcuna risposta. La richiesta broadcast è necessariamente un comandodi scrittura e tutti i dispositivi devono accettare questo tipo di richieste. L'indirizzo 0 è riservato alla loro identificazione.
Il protocollo definisce come il master e gli slave stabiliscono ed interrompono la comunicazione, come trasmettitore e ricevitore devono essere identificati, come i messaggi devono venire scambiati e come gli errori rilevati.
Solo il master può iniziare una transazione. Una transazione può avere il formato domanda/risposta diretta ad un singolo slave o broadcast in cui il messaggio viene inviato a tutti i dispositivi sulla linea che non danno risposta.
Una transazione è composta da una struttura singola domanda/singola risposta o una struttura singolo messaggio broadcast/nessuna risposta.
Modbus consente la comunicazione fra diversi dispositivi connessi alla stessa rete, per esempio un sistema che misura la temperatura e l'umidità e comunica il risultato a un computer, per tale peculiarità non c'è da stupirsi se in una qualsiasi rete modbus possiamo
trovare dispositivi anche molto diversi tra loro come PC , PLC , ambienti SCADA , HMI (Human Machine Interface)
Esistono due versioni del protocollo: una su porta seriale (RS232 di default, ma anche RS485) e Ethernet. Come si vede dalla figura queste due versioni possono coesistere nella stessa rete nel pieno rispetto delle funzionalità del protocollo ; nei capitoli successivi vedremo le peculiarità che distinguono le due versioni del protocollo.

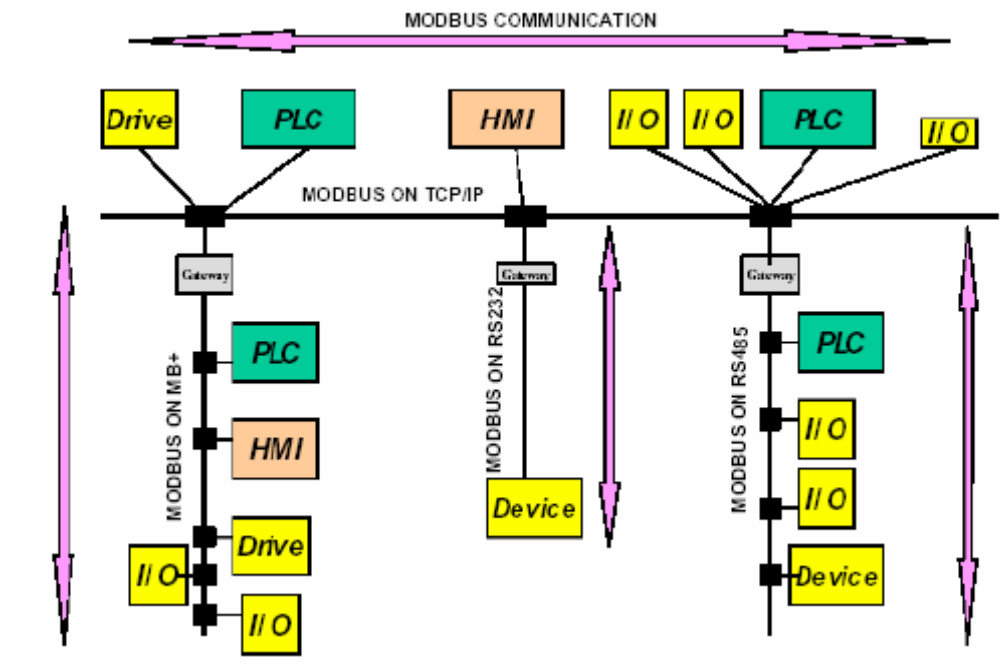

Figura 2. Modello di Comunicazione del protocollo Modbus

III ZigBee

Lo scopo del protocollo ZigBee è quello di fornire il supporto per reti *wireless* a basso consumo di energia, corto raggio e bassa latenza. Questa tecnologia consente il rimpiazzo di cavi di collegamento tra dispositivi realizzando un collegamento radio. Sensori, spie luminose, impianti di condizionamento, segnalatori di posizione a breve raggio, tutti questi dispositivi possono far parte di un unico sistema ZigBee. Oltre a sostituire i cavi, ZigBee può agire da collegamento verso reti preesistenti (ovvero agire da *bridge*) oppure può essere visto come un sistema per realizzare piccole reti ad hoc quando ci si trova lontano da altre infrastrutture di rete. ZigBee prevede la possibilità di realizzare reti multihop. .
I dispositivi ZigBee operano nelle bande 2,4 GHz in tutto il mondo e sulle bande 915 MHz e 868 MHz in America ed in Europa rispettivamente. I canali disponibili su queste bande sono 16 per la 2,4 GHz, 10 per la 915 MHz ed 1 per la 868 MHz. Queste bande non richiedono la licenza delle frequenze utilizzate ma, proprio per questo, i dispositivi ZigBee devono essere molto tolleranti rispetto alle interferenze radio , l'accesso al canale viene gestito mediante l'utilizzo del protocollo CSMA/CA

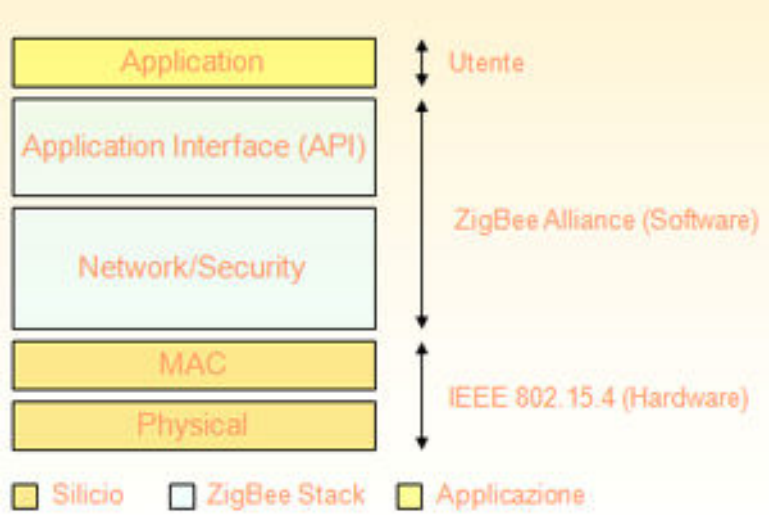

Figura 3. Protocollo ZibBree

Progettazione

I. Modello OSGI applicato al Protocollo Modbus

Prima di descrivere lo sviluppo del sistema software è opportuno illustrare brevemente la struttura di un generico componente software per ambiente OSGi.

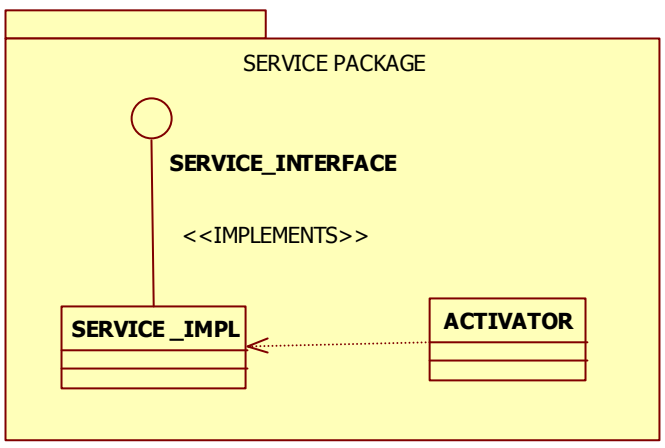

Figura 4. Struttura interna di un Bundle

Come illustra il diagramma , ogni bundle è articolato in tre file :
1)Il file **Activator** 2)Il file **Service_Interface** 3) Il file **Service_impl**
Il file activator che, implementando l'interfaccia *BundleActivator*, permette al framework di gestire il *lifecycle* del componente con i metodi *Start()* - invocato all'attivazione del *bundle* - e *Stop()* - richiamato in fase di chiusura.
L'interfaccia Service_Interface definisce le funzionalità che può offrire il componente, inoltre è all'interfaccia a cui ci si deve riferire per ottenere un riferimento ad un servizio registrato.
La classe Service_Impl , infine, non è altro che l'implementazione dell'interfaccia definita al punto precedente.
Alla fase di realizzazione dei file sopra citati , segue la fase di creazione e modifica del file manifest.mf che definisce le direttive di distribuzione e installazione del *bundle* all'interno della piattaforma, il suo *classpath*, le sue dipendenze da altri package e servizi, oltre che i servizi che espone.

Al fine di implementare il servizio andiamo ad analizzare le classi che costituiscono il componente ; nel caso in esame, la struttura delle classi è molto semplice : il servizio è formato da tre classi di cui una è la classe activator del bundle che implementa il servizio ,le altre due classi sono rispettivamente 1) l'interfaccia del componente 2)l'implementazione dell'interfaccia del componente.
Andiamo a definire il package MBComunicator in cui viene implementato il servizio Modbus Comunicator.
Descriviamo alcuni aspetti della classe ModbusComunicatorImpl, che implementa le funzionalità atte alla comunicazione su di una rete modbus.
Partendo dagli attributi possiamo osservare le variabili : **1)serialParameters :** questa variabile ha la responsabilità di gestire i parametri per l'apertura della comunicazione seriale con la rete modbus. 2)**mobusMaster :** questo oggetto ha la responsabilità di implementare alcune funzionalità per l'accesso in lettura /scrittura dei dispositivi hardware
Una nota va fatta per i metodi **OpenSerialConnection()** e **CloseSerialConnection()** , essi non appartengono alla semantica del protocollo ma sono indispensabili per avviare una connessione seriale su cui far viaggiare le frame modbus; tali metodi sfruttano le funzionalità offerte dalla libreria Java Comunicator .

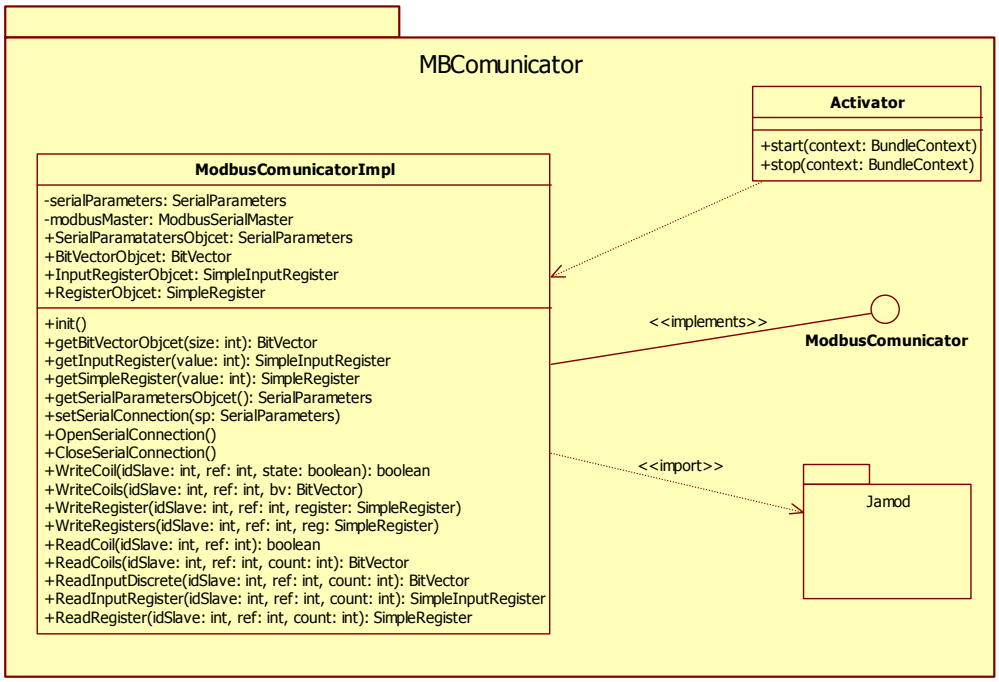

Figura 5. Modbus Comucator

Andiamo a osservare cosa devono fare i metodi descritti all'interno delle interfacce ModbusComunicator ai fini sia dell'applicazioni di monitoraggio sviluppata ma anche per terze applicazioni .
Si noti come un qualsiasi applicazioni che sfrutti il componente THL_Sensor possa ottenere informazioni dalla rete hardware , come la temperatura , pur ignorando il protocollo modbus , ciò perche nello sviluppo dei componenti si è cercato di sollevare l'utilizzatore finale dall'onere di conoscere i dettagli del protocollo di comunicazione della rete.

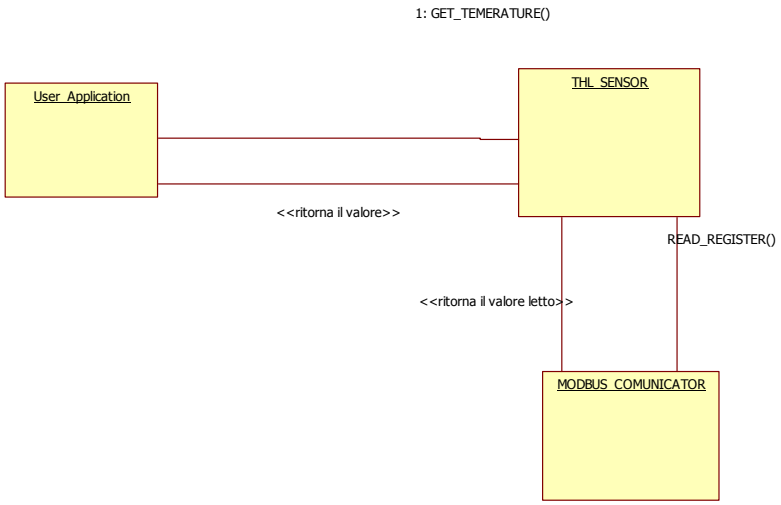

Figura 6. Iterazione del Modabus Comunicator con i sensori e all'applicazione utente

II. Progettazione Sistema Monitoraggio

Il sistema di monitoraggio dovrà provvedere a fornire all'utente utilizzatore tutte le funzionalità atte alla conoscenza dello stato della rete di sensori in termini di monitoraggio dei parametri ambientali come temperatura , luce , ecc e dovrà fornire informazioni di livello gestionale come livello della batteria dei dispositivi a batteria , numero di sensori installati ecc.

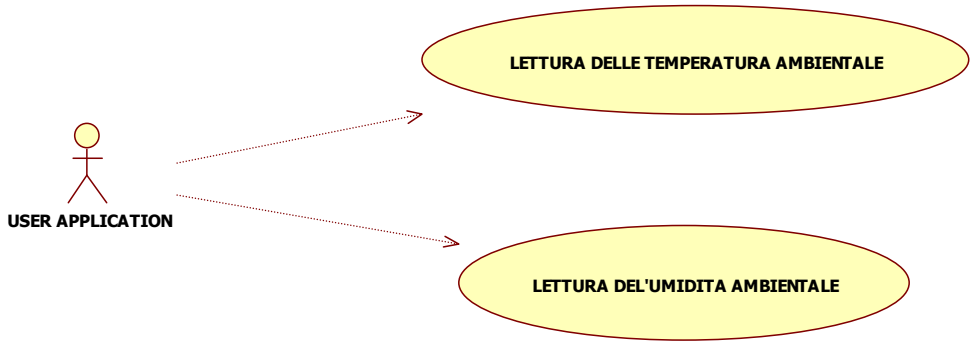

Figura 7. Use Case lettura parametri ambientali.

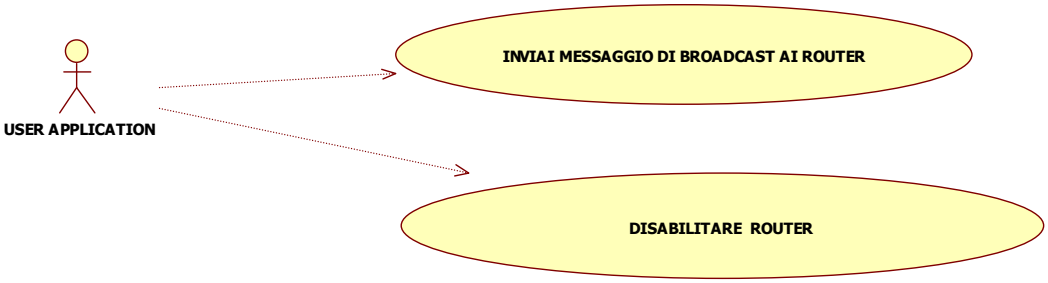

Figura 8. Use Case di comunicazione con il router.

Andiamo ad analizzare l'insieme delle classi che costituiscono il nostro sistema di monitoraggio. Per la progettazione si è scelto di virtualizzare ogni dispositivo facente parte della rete , di conseguenza possiamo classificare i vari componenti in modo molto semplice :
1)Un componente **Modbus Comunicator** per la comunicazione mediante protocollo modbus tra applicazione e dispositivi. 2) Un componente **WSN_Manger** per la gestione delle funzionalità amministrative sia del Gateway sia dei Sensori. 3)Un componente **Gateway** che implementa le funzionalità di monitoraggio che mette a disposizione il gateway , come numero di sensori attivi , ecc.. 4)Un componente **Router** che, oltre a fornire le funzionalità di routing, mette a disposizioni funzioni per il monitoraggio termico e per l'attivazione di attuatori 5) Un componente **Thl** che virtualizza i sensori usati per le misurazioni.
Mediante l'ausilio di un component diagram, andiamo a vedere le relazioni tra i vari componenti :
Come si può osservare dal grafico, tutti i componenti fanno riferimento al Modbus Comunicator ma ciò era prevedibile in quanto lo standard di comunicazione è il protocollo Modbus. Una considerazione va fatta sui componenti Gateway e THL infatti, come si vede, le totalità delle loro funzioni sono state divise in due componenti distinti : il WSN per le funzioni amministrative e i componenti Gateway e THL per quelle di monitoraggio; il motivo di questa scelta è per permettere il riuso delle funzioni di amministrazioni .
Il componente Router eredita l'interfaccia Sensor , ciò perché, come anticipato in precedenza , il router, oltre a fornire funzioni di routng, può essere integrato con sensori e attuatori da pilotare col l'ausilio di quest'ultima interfaccia.

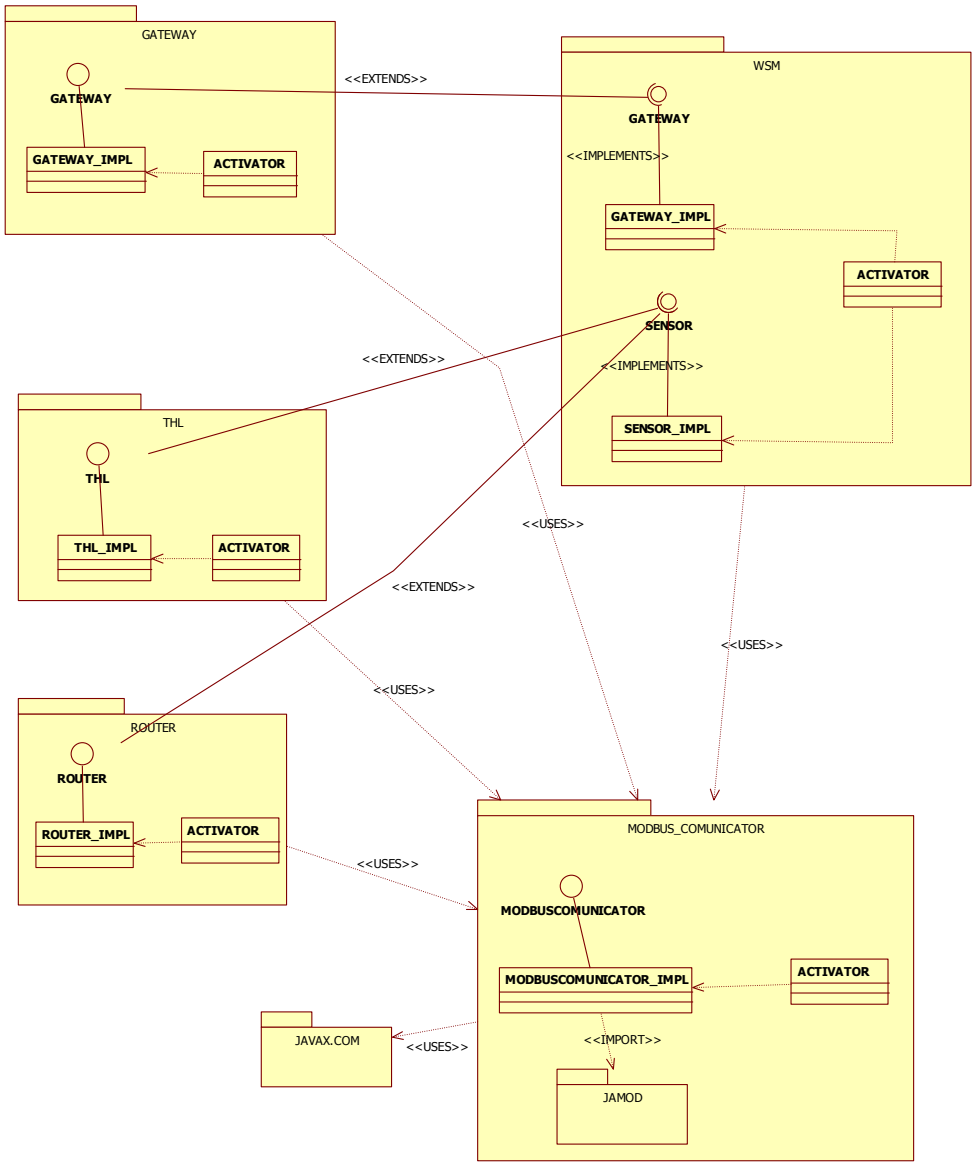

Figura 9. Relazione tra i componenti sviluppati

Nell'ultimo diagramma possiamo osservare il dispiegamento dei componenti sviluppati in relazione alle risorse esterne utilizzate per l'implementazione del progetto, possiamo osservare , come il servizio Modbus Comunicator debba essere allegato con le librerie Jamod e con la libreria Java Comunicator che fornisce i servizi per l'uso dell'interfaccia seriale; per quest'ultima vale il discorso fatto in precedenza per il Modbus Comunicator: è possibile disporre su due nodi distinti Comunicator e Libreria Java Comm

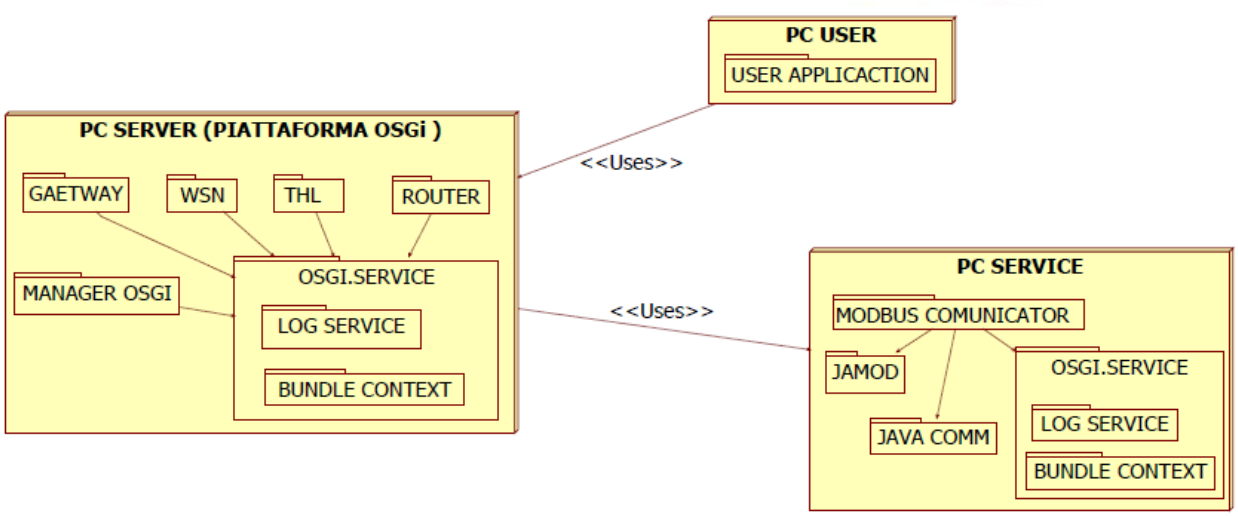

Figura 10. Dispiegamento dei componenti (OSGi)

Conclusioni

Il software prodotto è il risultato di conoscenze e tecnologie tutt'ora in fase di sviluppo , che si prestano e si presteranno a migliorare la qualità dei servizi tutt'ora esistenti.
Apparecchi come palmari e cellulari potranno , grazie a queste tecnologie, ampliare il loro insieme di servizi tutto a beneficio del consumatore.
Tecnologie come OSGi e ZigBee attualmente sono in via di sviluppo ma presto diventeranno protagonisti in molti ambiti industriali e sociali

Bibliografia

[1] The OSGi Alliance, *"OSGi Platform – Core Specification –Release 4"* (August 2005).[http://www.osgi.org]
[2] The OSGi Alliance, *"OSGi Platform – Service Compendium –Release 4"* (August 200)[http://www.osgi.org]
[3] The OSGi Alliance, *"About.the OSGi Service Platform – Technical WhitePaper"*, (Revision 4.1 – 7 June 2007). [http://www.osgi.org]
[4] Cesare Concordia, *"Lucidi del corso di Domotica"*.
[5] *"Pervasive Computing"* [http://en.wikipedia.org/wiki/Ubiquitous_computing]
[6] *"Pervasive Computing"* [http://laspinanelfianco.wordpress.com/2006/03/08/pervasive-computingubiquitus-computing/]
[7] *"Ambient Intelligence"* [http://en.wikipedia.org/wiki/Ambient_intelligence]
[8] *"Smart Home"* [http://it.wikipedia.org/wiki/Domotica]
[9]"Modbus" [*it.wikipedia.org/wiki/Modbus]*
[10] "Modbus IDA" [*www.modbus.org/*]
[11] Richard S. Hall , "OSGi R4 Service Platform:Java Modularity and Beyond", akquinet fws, BerlinMarch 21st, 2007
[12] "ZigBee" [*www.zigbee.org/*]

[13] ”Zig Bee” [*it.wikipedia.org/wiki/ZigBee*]
[14] “ IEEE 8002.15.4” [*en.wikipedia.org/wiki/IEEE_802.15.4*]