Progetto "Casa intelligente per una longevità attiva ed indipendente dell'anziano"
DGR 1464, 7/11/2011

Ambient-Aware LIfeStyle tutor, Aiming at a BETter Health

(Tutoraggio dello stile di vita basato sulla intelligenza ambientale, per una salute migliore)

Risultato D2.2

Specifica funzionale del sistema AALISABETH

Rev. 1.0, 7 Ottobre 2013

REGIONE MARCHE

INRCA Istituto di Ricovero e Cura a Carattere Scientifico
HEALTH AND SCIENCE ON AGING

La visione funzionale del sistema AALISABETH è riassunta dall'immagine in Figura 1: l'utente primario è definito dalla proposta come una (o più) persone in età anziana (65+), non soggetta a malattie croniche gravi o a disabilità importanti, ma sofferenti (o a rischio) di patologie metaboliche o circolatorie (per esempio, l'ipertensione o il diabete) o di deficit cognitivi lievi. Poiché il progresso di tali patologie è notoriamente favorito da stili di vita non correttamente regolati, caratterizzati da scarsa attenzione alla corretta alimentazione, all'esercizio fisico e alla rigorosa attinenza alle terapie mediche, il sistema AALISABETH dedica particolare attenzione al riconoscimento e alla valutazione di indicatori dello stile di vita, e allo sviluppo di logiche di interazione (che coinvolgano anche il medico curante e la rete di assistenza formale e informale) volte a favorire pratiche e stili di vita salutari. Il sistema AALISABETH intende quindi agire prevalentemente su aspetti di prevenzione, senza tuttavia tralasciare le componenti di monitoraggio, sicurezza, efficienza energetica che la natura tecnologica degli obiettivi naturalmente comprendono.

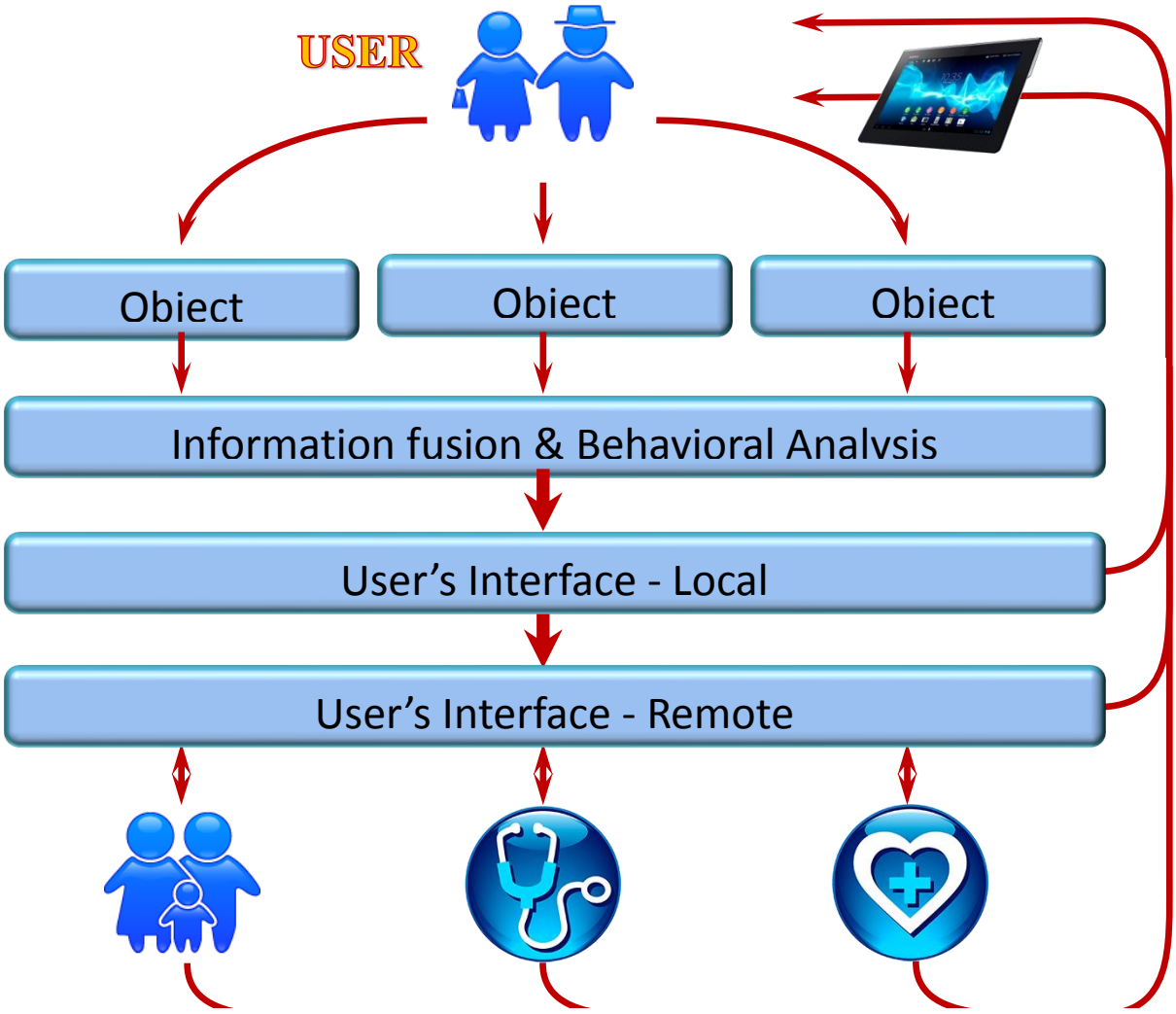


Figura 1: *visione funzionale del sistema AALISABETH*

Smart objects

La strategia identificata dal consorzio si basa sulla visione "stratificata" di figura: l'utente interagisce con il sistema prevalentemente attraverso gli **oggetti intelligenti** distribuiti nell'ambiente domestico. È prevista una ampia varietà di oggetti, con differenti funzioni primarie:

- **Sensori clinici,** utilizzati per la misurazione periodica delle grandezze fisiologiche di interesse. È previsto l'impiego di sensori di pressione sanguigna, glucometri, cardiofrequenzimetri, bilance (peso corporeo). Altri dispositivi (per esempio, pulsossimetri) potranno essere integrati in funzione delle specifico quadro clinico. I sensori utilizzati a questo scopo saranno sensori caratterizzati dalla massima semplicità d'uso, compatibili con l'impiego diretto da parte dell'utente o dei familiari. Il sistema fornirà assistenza e guida all'utente nell'utilizzo dei sensori clinici e gestirà in maniera automatica e trasparente all'utente il processo di memorizzazione e registrazione storica dei dati di misura.
- **sensori ambientali**, destinati al rilievo dei principali parametri ambientali, sia allo scopo di integrare le previste funzioni di gestione energetica, sia a quello di ottenere informazioni indirette e dettagliate sullo stile di vita. Un sottosistema "domotico" completo gestirà le principali funzioni di controllo ambientale (illuminazione, climatizzazione,…) e sicurezza. Dal punto di vista funzionale, anche in questo caso è essenziale la semplicità e accessibilità dei meccanismi di interazione, e lo sviluppo, per quanto possibile, di politiche di gestione automatizzate e coordinate. Come evidenziato più sotto, anche i sensori più tradizionalmente domotici, al di là della loro funzione primaria, contribuiranno al quadro generale di intelligenza ambientale e comportamentale previsto da AALISABETH. Per questo motivo, la tecnologia domotica di base (ITC, MAC) verrà integrata nel quadro infrastrutturale generale, completandola con le necessarie funzioni di comunicazione e registrazione degli eventi. Il sistema AALISABETH prevede inoltre sensori ambientali più specificamente orientati alla valutazione comportamentale, facenti uso della medesima tecnologia domotica, ma con collocazioni ed impieghi differenti: per esempio, sensori di apertura delle porte (comunemente previsti per applicazioni anti-intrusione perimetrale) saranno utilizzati sui varchi interni o su sportelli o cassetti per ricavare indicazioni comportamentali. Analogamente, i sensori di presenza (in tecnologia passiva a infrarossi – PIR, comunemente adottati per controlli volumetrici anti-intrusione) forniranno informazioni comportamentali in maniera economica e non invasiva. Tale impiego di sensori ambientali, tuttavia, richiede al sistema la capacità di attribuire l'informazione rilevata ad una specifica persona (nel caso l'ambiente sia abitato o frequentato da più di una persona), richiamando i temi di identificazione e localizzazione discussi nel seguito.
- **sensori personali**: è previsto il ricorso a sensori indossabili, dotati di una varietà di sensori per il rilievo simultaneo di più grandezze. Anche in questo caso, AALISABETH prevede un doppio livello di funzionalità: un primo livello in cui il sensore provvede autonomamente ad alcune funzioni primarie (per esempio, la rilevazione automatica delle cadute o di anomalie posturali, oppure la richiesta di assistenza tramite un pulsante) si affianca ad un secondo livello nel quale il sensore contribuisce alla conoscenza comportamentale del sistema attraverso dati più dettagliati relativi all'attività del portatore. Si farà riferimento a tecnologie già disponibili, impiegando il sensore wireless MuSA (UniPR, [1]) ed estendendone le funzionalità pregresse in diverse direzioni. MuSA consente il monitoraggio continuo delle cadute e, se necessario, di frequenza cardiaca e respiratoria. Ai fini propri di AALISABETH, sono previsti sviluppi sia di tipo tecnologico (riduzione di consumi e dimensioni) che funzionale. In particolare, è in corso lo studio di funzioni più specificamente orientate al supporto dell'analisi comportamentale: per esempio, l'identificazione di caratteristiche dinamiche e posturali permette il riconoscimento qualitativo di alcune azioni elementari (sedersi, alzarsi, camminare, ecc.) che a loro volta permettono un primo livello di intelligenza comportamentale autonoma, elaborata "a bordo" del dispositivo indossabile: la disponibilità di tale funzione permetterà, da una parte, di migliorare le prestazioni intrinseche del dispositivo (aumentandone l'affidabilità, per

esempio, nel discriminare le condizioni di rischio o caduta, e riducendone i consumi dovuti alla trasmissione, non più necessaria, di grandi quantità di data "grezzi" verso una unità di elaborazione centrale) e, dall'altra, di fornire al livello superiore di analisi una serie di dati più ricchi ed accurati. Inoltre, i sensori indossabili forniscono intrinsecamente un importante supporto alle necessarie funzioni di identificazione e localizzazione.

- **sensori specifici**, volti alla valutazione diretta di parametri connessi allo stile di vita. Sono state identificate diverse soluzioni, anche in funzione delle specifiche competenze presenti nella partnership, con riferimento alle tre aree sopra ricordate: attività fisica, alimentazione, terapie.

  Attività fisica: informazioni in merito verranno in larga parte acquisite dai sensori ambientali e personali sopra indicati. Tuttavia alcuni elementi specifici sono stati identificati: verranno realizzati (UniPR, MAC, Valdichienti) arredi (letti, sedie, poltrone) sensorizzati, utili per il tracciamento delle abitudini quotidiane e delle loro variazioni significative. Si utilizzeranno, a questo scopo, tecnologie diverse: una prima soluzione, basata su dispositivi più semplici e commercialmente disponibili (*sensitive pads*), è adatta anche alla "sensorizzazione" di mobili non esplicitamente progettati allo scopo, richiedendo interventi solo nella fase di finitura o allestimento. Una seconda soluzione studiata prevede l'impiego di celle di carico multiple, consentendo non solo la classificazione "binaria" dell'occupazione del letto/poltrona, ma anche una analisi della distribuzione dinamica dei pesi, utile per esempio alla valutazione della "qualità" del sonno.

  Un'altra applicazione prevede la stima della velocità della camminata e delle sue variazioni nel tempo, clinicamente riconosciuta come un parametro particolarmente espressivo dello stato di salute. Anche in questo caso, sono allo studio soluzioni diverse: per esempio, la misura della velocità potrebbe essere ottenuta misurando il tempo di attraversamento di una base di lunghezza nota lungo un cammino obbligato (un corridoio), attraverso sensori "di traguardo". Alternativamente, l'impiego di tag RFID, opportunamente inseriti nelle calzature (DIENPI) permetterebbe una funzionalità analoga, contribuendo inoltre al tema della localizzazione ed identificazione.

  Alimentazione: l'acquisizione di informazioni sul regime alimentare comporta notevoli difficoltà, sia sul piano puramente tecnologico che su quello, non meno rilevante, della non-intrusività e accessibilità delle tecnologie utilizzate. Tuttavia, va notato che una valutazione analitica e quantitativa del regime alimentare non è strettamente necessaria agli obiettivi progettuali: lo scopo non è infatti, in questo caso, un rigoroso controllo nutrizionale. Sono invece importanti la valutazione delle abitudini, la verifica di una sufficiente varietà nella tipologie degli alimenti, la regolarità, ecc. È quindi possibile limitarsi ad una valutazione "semi-quantitativa", con tolleranze sufficientemente ampie da evitare il ricorso a metodiche di misura eccessivamente invasive. Il progetto prevede una serie di informazioni indirette, acquisite attraverso il monitoraggio di alcune attività nella preparazione del cibo: anche in questo caso alcuni cassetti/sportelli della dispensa saranno monitorati; gli accessi al frigorifero saranno monitorati attraverso un sensore autonomo, in grado di fornire informazioni sull'apertura della porta e sulla conservazione degli alimenti (temperatura, umidità e, in prospettiva, qualità dell'aria, UNIPR). L'impiego del piano cottura (indicatore della frequenza e regolarità delle attività di preparazione del cibo) verrà monitorato attraverso sensori appositi (ERS). Altri elettrodomestici (forno, per esempio) potranno essere monitorati indirettamente attraverso il controllo dei consumi elettrici.

  Informazioni dirette sul consumo di alimenti verranno invece da dispositivi progettati ad hoc: sono in corso di definizione le specifiche di dispositivi per l'identificazione e la quantificazione degli alimenti, basati su una strategia ibrida che prevede sia l'impiego di sensori che l'interazione semplificata con l'utente. In particolare, un "tovaglietta" sensorizzata e interattiva sarà utilizzata a questo scopo (MAC), sviluppata anche in una versione fissa, da includersi nella cucina (MAC, UNIPR, EUSEBI). È pianificato l'impiego di contenitori e stoviglie destinati a tale interazione (Ralò). Verranno sperimentate anche

tecniche di identificazione e quantificazione semiautomatiche, basate su visione artificiale (UNIPR). Il problema della stima volumetrica rappresenta, in questo caso, il tema di maggiore impegno tecnologico, e sono allo studio tecniche di analisi basate su una molteplicità di immagini, adatte sia alla implementazione su stazione fissa (arredo cucina) che mobile (smartphone). Quest'ultima permetterà l'estensione di alcuni servizi previsti da AALISABETH anche al di fuori dei confini domestici (per esempio inserendo nelle valutazioni comportamentali dati rilevati nei pasti consumati fuori di casa).

Aderenza alle prescrizioni: è in fase di sviluppo il progetto di un erogatore intelligente di medicinali. Il dispositivo, integrato nella infrastruttura di comunicazione generale di AALISABETH ma, al pari degli altri componenti, dotato di intelligenza e capacità di decisione autonoma, sarà in grado di erogare, al tempo opportuno, i diversi medicinali previsti dalla terapia (è frequente che l'utente anziano sia soggetto all'assunzione quotidiana di medicinali di diverso genere). Le caratteristiche specifiche e maggiormente innovative del dispositivo consisteranno nella possibilità di gestire i medicinali direttamente all'interno della confezione (blister), senza quindi i rischi igienici associati alla manipolazione di medicinali sfusi, e nel riconoscimento del medicinale in via di erogazione, attraverso procedure di controllo incrociato in grado di prevenire errori. In questo contesto, verranno utilizzate anche tecniche di visione artificiale, con sensori di immagine a bordo dell'erogatore: il medicinale verrà seguito dalla estrazione dalla confezione fino al prelievo da parte dell'utente, mantenendo quindi traccia dell'erogazione e dell'effettivo prelievo di ciascun medicinale prescritto, consentendo l'implementazione di procedure per il richiamo dell'attenzione dell'utente nel caso di mancato prelievo e successivamente, se la condizione persiste, di allertare familiari e rete di assistenza. Nel progetto del dispositivo di erogazione (SassoMeccanica, METEDA, Western, UniPR) particolare attenzione verrà dedicata alla "scalabilità" dell'approccio, in modo da identificare soluzioni praticabili nell'ambiente domestico, in grado tuttavia di crescere in maniera modulare fino a rispondere alle esigenze di strutture di ricovero e cura.

Infrastruttura

Le informazioni provenienti dagli oggetti intelligenti distribuiti nell'ambiente, oltre ad essere oggetto di elaborazione locale (come descritto nel paragrafo precedente) vengono raccolte dallo strato gerarchicamente superiore, ove vengono opportunamente astratte dalla propria specifica origine fisica e registrate su una base di dati comune all'intero sistema. Tali dati vengono utilizzati per una varietà di operazioni: a questo livello viene implementata la supervisione del sistema domotico, le funzionalità di comunicazione a livello locale e remoto condivise dai vari oggetti e, soprattutto, le funzionalità maggiormente innovative di fusione dell'informazione e analisi comportamentale.

A questo scopo, è in fase di costruzione una specifica ontologia di sistema (OntoAALISABETH, UniCAM), necessaria a rappresentare formalmente i concetti e le relazioni rilevanti per il dominio.

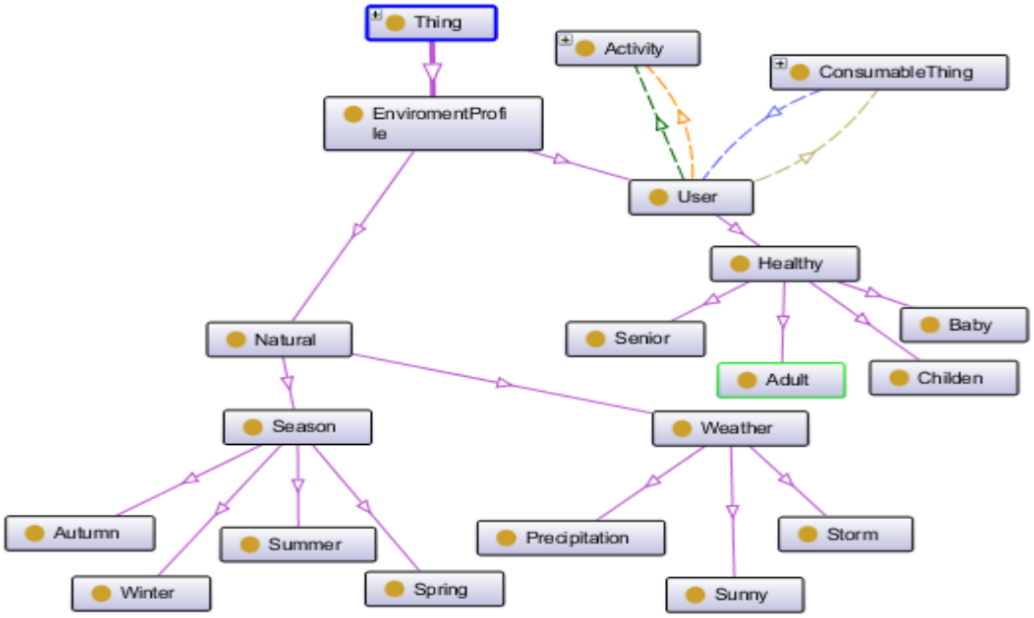


Figura 2: *descrizione ontologica di un profilo ambientale*

La disponibilità di tale ontologia facilita l'interoperabilità semantica e le costruzione della inferenze e abilita l'uso di strumenti (reasoner) che consentono di esplicitare la conoscenza implicita. Per la costruzione di OntoAALISABETH, ci si è basati sulla disponibilità di una ontologia sviluppata per la descrizione di *smart homes* (DogOnt, [2]), che è stata completata introducendo elementi relativi alla descrizione dei parametri clinici personali, alle abitudini alimentari e alle abitudini personali. Formalmente, quindi, l'ontologia ha richiesto l'introduzione e la caratterizzazione di nuove classi destinate alla rappresentazione di tali informazioni. A titolo di esempio, la Figura 2 mostra la struttura gerarchica ipotizzata per la descrizione di un profilo ambientale.

A livello di astrazione inferiore, è già stata definita e concordata fra i partner una struttura dati sufficientemente aperta e versatile per includere le informazioni eterogenee provenienti dal sistema stesso, sulla quale sarà necessario "mappare" l'ontologia una volta completata. In allegato viene riportato il documento di riferimento elaborato per la descrizione del database, mentre la Figura 3 ne rappresenta una visione sinottica.

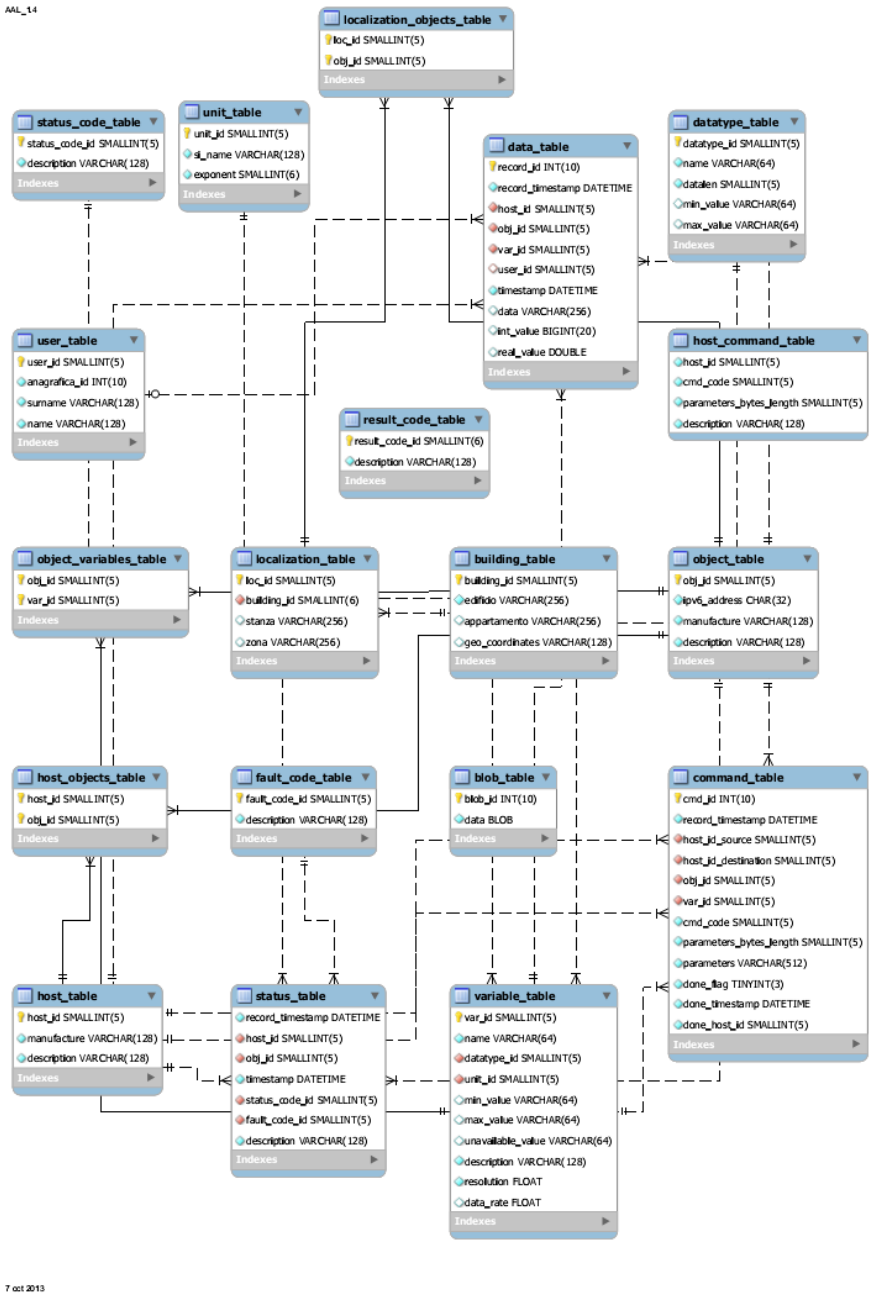


Figura 3: *rappresentazione sinottica della base di dati del sistema AALISABETH*

In accordo con la visione della "Internet of things", ogni elemento del sistema verrà virtualmente mappato su una rete IPV6; gli elementi più semplici, privi di connettività IP nativa, verranno affacciati alla rete attraverso gateway appositi. Particolare attenzione è stata rivolta alla interoperabilità: la struttura infatti non prevede l'adozione "obbligatoria" di specifici protocolli di comunicazione, ma definisce uno spazio di informazione aperto a sistemi eterogenei. In pratica, il cuore del sistema consiste in una base di dati MySQL, alla quale ciascun sottosistema può accedere secondo le modalità di comunicazione ed i protocolli più convenienti. Anche in questo caso, i dispositivi più semplici sono assistiti dalle funzioni di moduli con funzioni di gateway, indipendenti fra loro. La Figura 4 sintetizza l'architettura di sistema prevista:

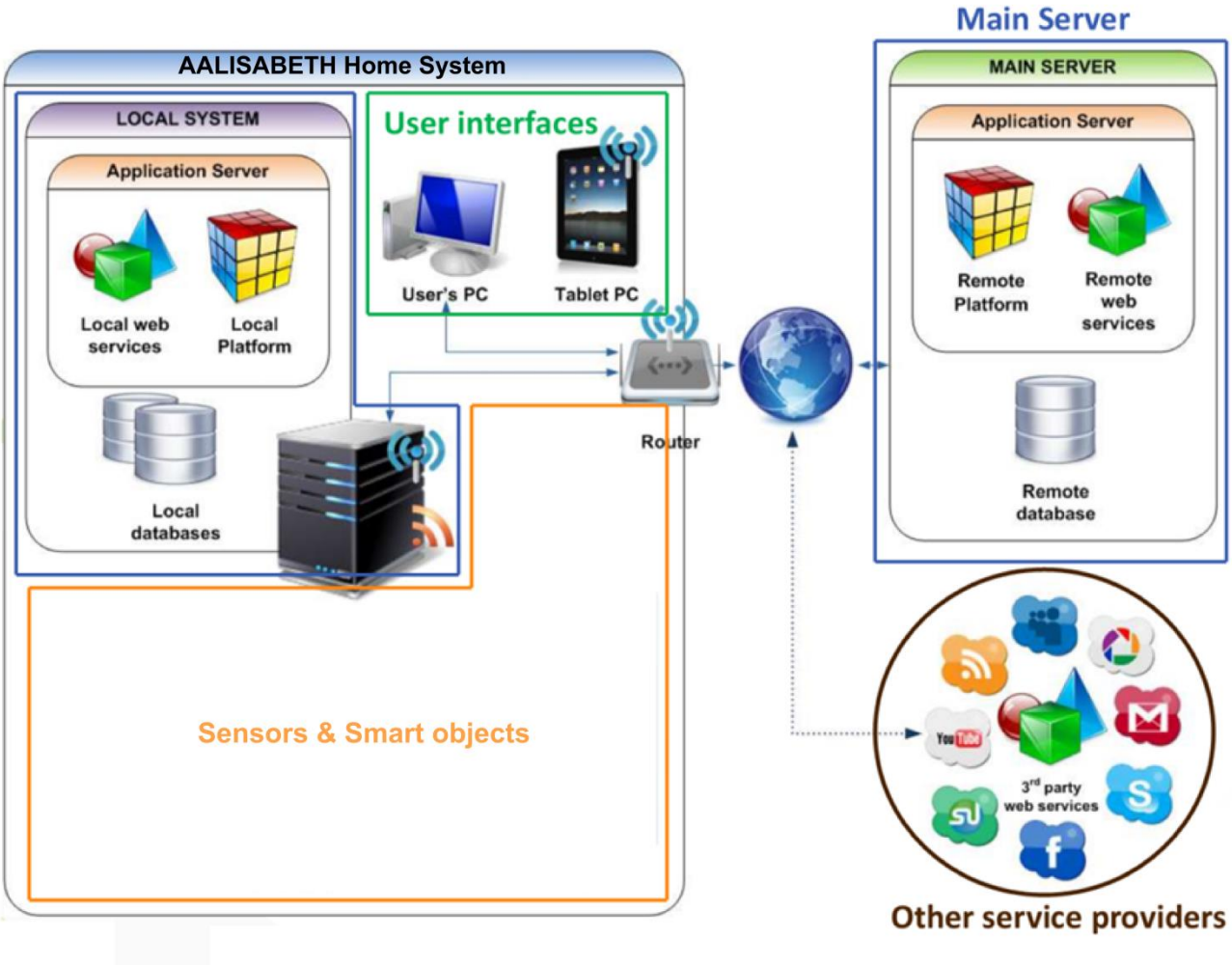


Figura 4: *rappresentazione schematica della infrastruttura del sistema AALISABETH*

In particolare, in figura viene evidenziata l'apertura a servizi remoti, attraverso i quali comunicare con familiari, medici, assistenti sanitari e interagire con altri sistemi (per esempio nel contesto di progetti paralleli nel programma *"Casa intelligente per una longevità attiva ed indipendente dell'anziano"*, o per l'integrazione di servizi AALISABETH entro sistemi di gestione sanitaria).

Lo sviluppo delle interfacce utente sarà oggetto di una relazione specifica (D2.3, Linee guida per lo sviluppo delle interfacce utente) e non viene quindi dettagliato in questo contesto.

Identificazione e localizzazione

Un tema di particolare rilievo emerso nei primi mesi dell'attività di progetto e al quale sono state dedicate attività di analisi e sperimentazione è il tema relativo alle funzionalità di identificazione dell'utente: la filosofia di AALISABETH prevede infatti di sfruttare informazioni di natura ambientale (quindi non associate univocamente all'utente) allo scopo di inferire profili comportamentali. Nel caso di ambienti abitati da più persone, si pone quindi il problema dell'attribuzione della specifica azione rilevata ad uno specifico attore. Un aiuto significativo può venire in questo caso dai sensori indossabili, univocamente associati alla persona. Tuttavia, alcune situazioni possibili non possono essere risolte solo su questa base; un semplice esempio può aiutare a chiarire la problematica: se il sensore del frigorifero segnala l'apertura della porta, e la rete wireless rileva la presenza di due persone diverse (ovvero due sensori indossati) nello stesso ambiente, occorre "localizzare" le persone con precisione sufficiente ad attribuire l'azione univocamente ad una delle due. Il tema della localizzazione in ambienti chiusi è un tema di rilevante importanza per numerosi ambiti applicativi e numerose soluzioni sono proposte in letteratura: tuttavia, lo specifico contesto applicativo, ed i relativi vincoli di costo e di non-invasività

precludono l'adozione di alcune delle soluzioni più efficaci e note. D'altra parte, occorre rilevare che il sistema, essendo orientato al monitoraggio di lungo periodo e non ad applicazioni "*life-critical*", permette intrinsecamente una certa tolleranza in tali funzioni: è infatti perfettamente accettabile che una particolare azione non venga attribuita, se l'identificazione è incerta; semplicemente questa non contribuirà alla costruzione dei profili comportamentali. È quindi rilevante, ai fini della qualità di tali profili, ottenere una elevata percentuale di riconoscimento e localizzazione, ma non è indispensabile il ricorso a tecnologie (più impegnative e costose) caratteristiche dei sistemi di sicurezza.

Le attività di progetto hanno quindi dedicato particolare attenzione a questo tema: in particolare, le risoluzioni permesse (in ambienti chiusi e di ridotte dimensioni) dalle tecniche classiche di radiolocalizzazione, basate sulla triangolazione e sull'analisi delle intensità/qualità del segnale radio oppure sulla stima del "tempo di volo", non sono in genere sufficienti allo scopo. A causa dei cammini multipli nella propagazione del segnale, la discriminazione richiederebbe infatti una eccessiva densità nella distribuzione di antenne fisse e si è ritenuto tale approccio eccessivamente invasivo. Tecniche di "navigazione inerziale", basate sulla ricostruzione dello spostamento dell'utente sulla base dei tracciati di accelerazione rilevati dal sensore indossato, sfruttano dispositivi già disponibili, ma soffrono tuttavia notoriamente di errori di "deriva" associati alla integrazione delle informazioni accelerometriche. Altre informazioni di localizzazione possono venire dalla interazione con oggetti intelligenti: per esempio, il passaggio in un varco, oppure la seduta su una poltrona sensorizzata.

Si intende quindi fare ricorso, anche in questo caso, a soluzioni basate sulla fusione dell'informazione, combinando tecniche diverse e fra di loro complementari allo scopo di raggiungere l'affidabilità richiesta.

Il modello generale prevede di affidare parte del compito di localizzazione al sensore indossabile, parte alla rete di dispositivi ambientali e parte al sistema di fusione dell'informazione: il sensore indossabile ricostruisce le traiettorie attraverso i dati accelerometrici, e riceve dalla rete ambientale informazioni di riposizionamento puntuale, inferite in funzione della interazione con i diversi dispositivi, che premettono di rigenerare frequentemente l'informazione di localizzazione inerziale, azzerandone l'errore associato alla deriva.

Sono state analizzate e sperimentate altre tecnologie per la costruzione di "fari di puntualizzazione": in particolare tecnologie basate su transponder ultrasonici indossati (analizzate da MAC) hanno dimostrato adeguate caratteristiche di portata e selettività, soffrendo tuttavia di significativi problemi di attenuazione dovuti agli indumenti indossati. Dettagli su tale sperimentazione sono riportati nel rapporto tecnico allegato. Uno studio è in corso per valutare l'implementazione di funzionalità analoghe sfruttando direttamente i segnali a radiofrequenza, opportunamente modulati in portata e direzionalità. Una possibilità ulteriore in corso di valutazione è rappresentata dalla fusione (a basso livello, ossia a bordo del sensore indossabile) di dati inerziali e di tecniche di radio-triangolazione convenzionali.

[1] Francesco Montalto, Valentina Bianchi, Ilaria De Munari and Paolo Ciampolini, "A Wearable Assistive Device for AAL Applications", in *Assistive Technology: From Research to Practice*, P. Encarnação et al. (Eds.), IOS Press, 2013, pp. 101-106

[2] http://www.cad.polito.it/pap/exact/iswc08.html

ALLEGATO 1 – Rapporto tecnico MAC

AALISABETH Utilizzo sensore ad Ultrasuoni per il riconoscimento utente	26.09.2013
Stesura : ing Andrea Pieretti	

Nel progetto AALISABETH, uno dei requisiti fondamentali richiesti dal bando è di monitorare il comportamento dell'utente soggetto di studio. Di particolare importanza risulta la necessità di associare determinate azioni ad un determinato soggetto: e.g. l'apertura di particolari sportelli, il passaggio ripetuto in determinate zone della casa, etc.
Risulta quindi fondamentale trovare uno strumento che permetta di capire la posizione dell'utente all'interno dell'abitazione, in modo da permettere le associazioni "azione/utente".
La sperimentazione fatta riguarda l'utilizzo di sensori ad ultrasuoni per determinare il riconoscimento dell'utente.
Il sensore ad ultrasuoni utilizzato ha una frequenza di trasmissione di 40kHz ed ha un funzionamento da ricetrasmettitore:

- Se eccitato con una tensione (non superiore a 20V peak to peak) genera un'onda di pressione sonora a 40kHz proporzionale alla tensione.
- Se riceve un'onda di pressione sonora a 40kHz, genera una tensione sinusoidale di ampiezza proporzionale all'intensità dell'onda sonora ricevuta.

Per ottenere delle tensioni con una dinamica sufficiente ad essere rilevata, il ricevitore deve essere seguito da uno stadio amplificatore in cascata, possibilmente con uno slew-rate elevato in modo da poter ottenere la massima dinamica per il segnale a 40kHz.

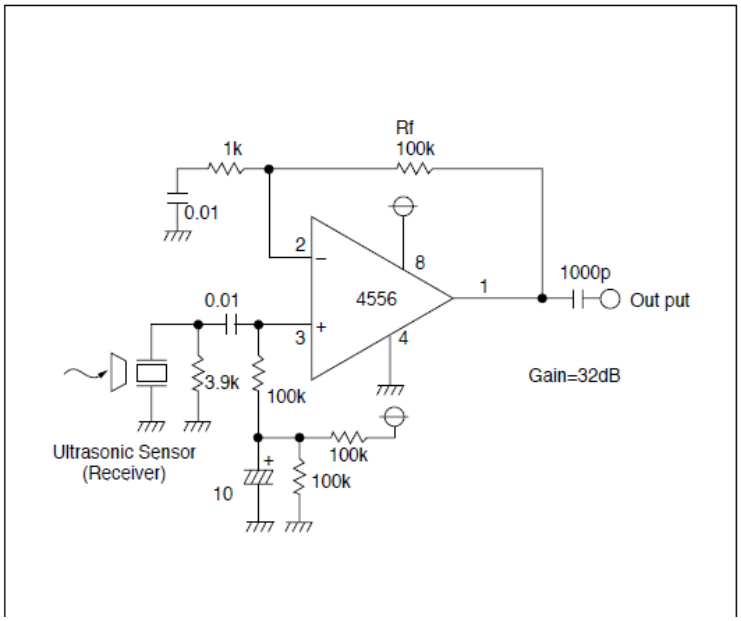


Figura 1: Es. schema circuitale stadio ricevitore

L'idea è quella di posizionare dei trasmettitori in diverse posizioni strategiche all'interno dell'abitazione, e di posizionare un ricevitore a bordo del badge Zig-Bee indossato dall'utente: tale ricevitore dovrà discriminare i vari segnali emessi dai vari trasmettitori e potrà decidere quale trasmettitore si trova in sua prossimità.
Per evitare la sovrapposizione dei vari segnali emessi dai fari ultrasonici, questi ultimi potrebbero essere associati ad una rete gestita in maniera sincronizzata: associando un time-slot ad ogni trasmettitore si facilita così la discriminazione del segnale da parte del ricevitore, in quanto le eventuali interferenze saranno dovute solamente alle onde sonore riflesse (inevitabili in ambiente domestico). Per la sincronizzazione si potrebbe utilizzare un rete IEEE 802.15.4 *beacon-enabled* in modo da associare un *time-slot* di ogni *super-frame* ad ogni trasmettitore.
Ogni trasmettitore inoltre trasmetterà un segnale *burst* caratterizzato da un pattern specifico: al badge ricevitore basterà quindi riconoscere il pattern per determinare a quale faro ultrasonico corrisponde il segnale ricevuto.
Oltre all'ampiezza del segnale ricevuto, l'utilizzo dei sensori ad ultrasuoni permette anche di avere a disposizione il parametro "tempo di volo" per determinare la distanza del trasmettitore dal badge ricevitore. La velocità finita delle onde sonore permette di determinare facilmente lo sfasamento tra il segnale ricevuto ed il segnale trasmesso: tale sfasamento permette di estrapolare la distanza tx-rx (supponendo che la velocità nel mezzo trasmissivo sia costante, mentre in realtà è influenzata da fattori come umidità, temperatura, etc).
Per misurare tale sfasamento si possono seguire due approcci:

1. Il badge ricevitore appartiene alla stessa rete sincronizzata dei vari trasmettitori ultrasonici: in questo modo il ricevitore viene informato direttamente dalla rete sull'istante di inizio di trasmissione del burst ultrasonico e ne può facilmente individuare lo sfasamento.

2. Il badge ricevitore non appartiene alla rete sincronizzata dei trasmettitori, ma una volta ricevuto il segnale ne trasmette a sua volta uno in cui è codificata l'ampiezza del segnale ricevuto. Il faro che ha trasmesso, riceverà tale segnale contenente l'informazione dell'ampiezza ricevuta dal trasmettitore (oppure analizzerà l'ampiezza del segnale stesso). Inoltre, impostando un tempo fisso Δt per l'elaborazione del segnale da parte del badge, si potrà risalire al tempo di volo. Tale soluzione risulta più complessa della precedente in quanto richiede l'implementazione sia dello stadio trasmettitore che ricevitore in ogni dispositivo (sia faro ultrasonico che badge utente). Inoltre la trasmissione del segnale da parte del badge utente richiederà una maggiore tensione (max 20Vpp) e quindi maggiori ingombri e un maggior consumo di batterie.

Le prove effettuate hanno dimostrato che i sensori hanno un'apertura di cattura di circa 50° e che si ottengono ottimi risultati entro 1.5m di range.
Il problema fondamentale riscontrato è l'attenuazione causata dalla presenza di tessuti davanti al ricevitore: in particolar modo diversi strati di tessuto pesante oppure di jeans attenuano il segnale fino a renderlo inutilizzabile allo scopo. Tale limitazione risulta troppo elevata e comporta quindi l'impossibilità di utilizzare i sensori ad ultrasuoni per l'applicazione in esame.
Riteniamo tuttavia che il lavoro svolto possa essere la base per ulteriori sperimentazioni e di interesse per eventuali applicazioni future.

Versioning			
Codice Revisione	Data	Approvato da	Commenti
1.0	26.09.2013	Pieretti - Mandolini	

ALLEGATO 2 – Descrizione Database

Descrizione
Database interazione
AALISABETH
(DRAFT)

Rev. 0.3 ref. database: AAL_1.2.sql	Aggiunta BLOB TABLE Aggiunta USER TABLE Revisionata DATA TABLE Aggiunta tabella BUILDING_TABLE e revisionata LOCATION_TABLE Revisionata OBJECT TABLE
Rev. 0.4 ref. database: AAL_1.3.sql	Resisionata DATA TABLE (eliminati field datatype_id/datalen impliciti in var_id reference) Script database: esplicitate le foreign_keys

Generalità

Al fine di consentire un'interazione fra i diversi elementi di campo forniti dai vari partner del progetto (sensori ed attuatori) con il sistema complesso di supervisione ed analisi comportamentale è utilizzato un database organizzato in modo da astrarre quanto più possibile l'informazione che deve essere scambiata ed immagazzinata rispetto a quelle che sono le caratteristiche peculiari del sensore o attuatore che la genera o utilizza.

Per consentire una semplice ed efficiente estrazione ed elaborazione dei dati da parte del sistema di analisi comportamentale, si è scelto un approccio di interazione / memorizzazione quanto più semplificato nelle modalità di estrazione e comprensione.

Per semplificare al massimo una interazione con il mondo esterno al sistema in modo uniforme all'approccio "internet of things" si è scelto di utilizzare come identificativo dei singoli dispositivi e/o elementi virtuali un identificativo IPv6 che possa essere, se necessario, esposto verso l'esterno mediante opportuni gateway, pur non richiedendo all'interno del sistema composito l'utilizzo di altri protocolli, se non quelli necessari all'interazione con il database di interscambio stesso.

Il database scelto per realizzare l'interscambio e la memorizzazione dello storico è MySQL della MySQL, ora acquisito da Oracle, ma sempre disponibile anche in modalità open-source gratuita. La scelta ricade su questo tipo di database per la portabilità su più piattaforme, Linux e Windows in particolare, la gratuità, la vasta disponibilità di API (application program interface) e librerie per l'interazione con lo stesso in svariati linguaggi ed ambienti operativi, la gratuità.

Organizzazione dei dati

Ciascun oggetto fisico (e virtuale) nel sistema ha un corrispondente identificativo univoco nella forma di indirizzo IPv6 (intero a 128 bit).

Ciascun oggetto può poi avere un numero arbitrario, compreso fra 1 e 2^{32}(16 bit sono sufficienti: 65535 variabili per dispositivo), di variabili, ciascuna univocamente identificata mediante un identificativo numerico a 32 bit. Ciascuna variabile rappresenta una specifica proprietà, dell'oggetto stesso.

Il tipo di ciascuna variabile può essere:

- intero con segno a 64 bit
- numero in virgola mobile in precisione singola (float)
- stringa di testo di lunghezza arbitraria

Questa scelta consente un'ampia libertà di descrizione dei dati, una facile estrazione delle informazioni semantiche con minime o nulle conversioni di formato, un buon compromesso in termini di spazio di memorizzazione richiesto, oltre a consentire in generale una lettura intellegibile diretta delle informazioni memorizzate nel database a fini di debug.

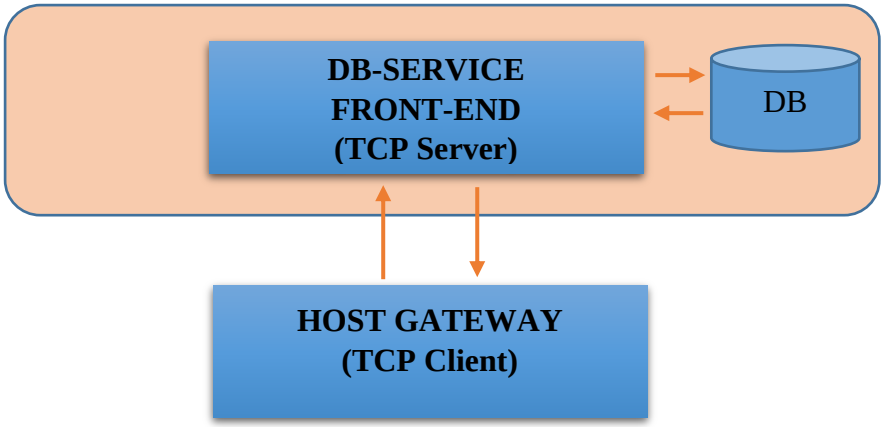
Data l'eterogeneità dei dispositivi coinvolti nel sistema è altresì necessario che sia presente una descrizione dettagliata di ciascun dispositivo inserito nel sistema. Verranno quindi definite di seguito anche delle caratteristiche delle singole tipologie di dispositivi, che verranno utilizzate per inserire, in ogni installazione, un file di configurazione che associ ad ogni singolo identificativo IPv6 una descrizione formale, necessaria ad interpretare correttamente le informazioni memorizzate.

Per inviare comandi ai singoli dispositivi verranno utilizzate apposite variabili e non scrivendo direttamente sulla proprietà che si intenda eventualmente modificare. In questo modo una variazione del valore delle proprietà si verificherà solamente in conseguenza di un'effettiva variazione della stessa segnalata dal dispositivo, e non in conseguenza di una mera scrittura da parte di un elemento che intenda inviare un comando (che può al limite anche essere ignorato o rifiutato).

DB-SERVICE AALISABETH

I vari gateway possono avvalersi dell'accesso al database per le operazioni di scrittura e lettura tramite l'apposito front-end (**DB-Service**).

I comandi utilizzati per la comunicazione con il front-end del database dovranno garantire la massima flessibilità e la massima astrazione dei dati, in modo che ogni gateway sia in grado di interagire con il database efficientemente.



Il compito del DB-Service è di rendere agnostici i gateway dalla natura del database utilizzato (il DB-Service è locale mentre il database può essere anche remoto).

Gli applicativi di *Data-Crunch/Data-Mining/Data-Statistics* non necessiteranno l'accesso al database tramite il DB-Service, ma potranno accedervi direttamente tramite delle query appropriate.

Le varie operazioni sul database vengono fornite ai gateway tramite le API fornite dal DB-Service. Per ogni comando inviato dal gateway, il DB-Service fornisce sempre un messaggio di risposta.

A basso livello le API rappresentano dei pacchetti nella seguente forma:

AAL_PREAMBLE	AAL_MSG_CODE	PAYLOAD

AAL_PREAMBLE: Trailer indicante l'inizio di un messaggio AALISABETH:

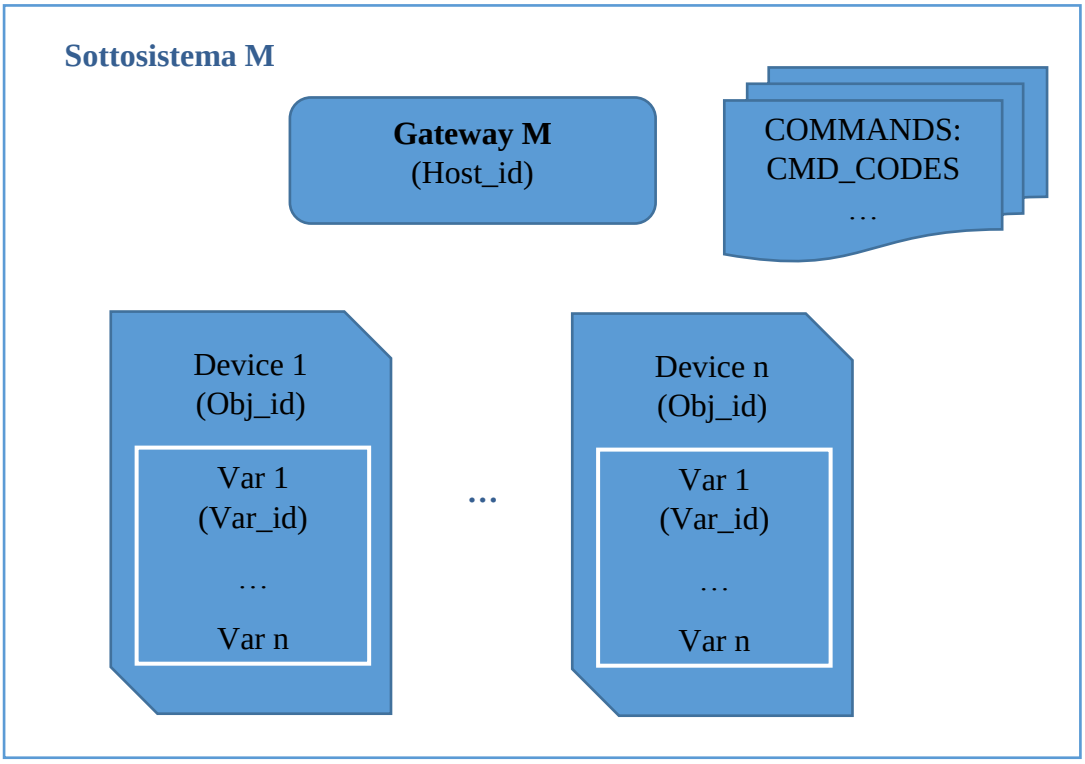
- **AAh AAh**: preamble AALISABETH request (usually from the client: host gateway).
- **AAh BBh**: preamble AALISABETH response (usually from the server: DB-Service).

AAL_MSG_CODE:

- **00h:** AAL_WRITE_DATA
- **01h:** AAL_WRITE_STATUS
- **02h:** AAL_WRITE_CMD

- **03h:** AAL_GET_NEXT_UNDONE_CMD
- **04h:** AAL_SET_CMD_DONE
- **….**

INDIRIZZAMENTO



Ogni sottosistema è caratterizzato da un gateway (definito univocamente dal *host_id*), al quale sono interfacciati diversi dispositivi (objs) caratterizzati da un codice univoco *obj_id*. Ogni elemento è caratterizzato da un codice id.
Ogni obj espone un insieme di grandezze/attributi/proprietà/configurazioni definite "variabili" e contraddistinte da un *var_id* univoco. Ogni sottosistema è dotato di un proprio (peculiare) set di comandi.

Host_id: Identificativo univoco del gateway. [2 bytes]

Obj_id: Identificativo univoco del dispositivo sotteso[2 bytes] espresso anche attraverso un ipv6_address

Var_id: Identificativo della variabile (grandezza, attributo, proprietà, misura) del dispositivo obj_id. [2 bytes]

TABELLE DEL DATABASE

Il database sarà dotato di alcune tabelle **compilate manualmente durante la realizzazione del progetto**. Il loro compito è di descrivere in modo univoco gli elementi che compongono il sistema, e di creare delle associazione tra gli elementi stessi.

Ogni ID di ogni tabella deve iniziare dall'indice 1. Il valore 0 è riservato per indicare il valore NULL.

HOST_TABLE (Gateways)[1]

Tabella contenente gli identificativi e le descrizioni dei vari gateway della rete. Ogni riga corrisponde ad un gateway.

HOST_ID	MANUFACTURE	DESCRIPTION
smallint unsigned	varchar[128]	varchar[128]

Host_id: Identificativo univoco del gateway [2 bytes]

Manufacture: Campo ASCII indicante il produttore del gateway (Free Text). [128 bytes] (e.g. MAC Srl)

Description: Campo ASCII con la descrizione del gateway (Free Text). [128 bytes]

OBJECT_TABLE[2]

Tabella contenente gli identificativi e le descrizioni dei vari dispositivi.

OBJ_ID	IPv6_ADDRESS	MANUFACTURE	DESCRIPTION
Smallint unsigned	Ipv6 char[32] (nibble ASCII)	varchar [128]	varchar [128]

Obj_id: Identificativo univoco nel db

IPv6_address: worlwide unique id espresso in nibble ASCII[32 bytes]

Manufacture: Campo ASCII indicante il produttore del dispositivo (Free Text). [128 bytes] (e.g. MAC Srl)

Description: Campo ASCII con la descrizione del dispositivo (Free Text). [128 bytes] (e.g. Codice modello / Product Number)

[1] Compilata manualmente durante la fase di realizzazione del progetto.
[2] Compilata manualmente durante la fase di realizzazione del progetto.

VARIABLE_TABLE[3]

Tabella contenente gli identificativi e le descrizioni delle varie variabili associate a dispositivi.

VAR_ID	NAME	DATATYPE_ID	UNIT_ID	MIN VALUE	MAX VALUE	UNVAILABLE VALUE	DESCRIPTION
smallint unsigned	varchar[64]	tinyint unsigned	tinyint unsigned	varchar[64] (nibble ASCII)	varchar[64] (nibble ASCII)	varchar[64] (nibble ASCII)	varchar [128]

Var_id: Identificativo della variale del dispositivo da modificare nel database. [2 bytes]

Datatype_id: Tipo di dato. [1 byte]

Unitcode_id: Codice indicante la grandezza fisica associata alla variabile e l'eventuale fattore moltiplicativo rappresentante le cifre decimali disponibili (evitando l'utilizzo dei float). [1 byte]

Min_value: Limite minimo range di validità della variabile (big-endian). [32 bytes]
Il numero di byte significativo è dato dal campo *Datatype.Length* ottenibile tramite il *Datatype_id* tramite la *DataType Table*.
NULL se privo di significato.

Max_value: Limite massimo range di validità della variabile (big-endian). [32 bytes]
Il numero di byte significativo è dato dal campo *Datatype.Length* ottenibile tramite il *Datatype_id* tramite la *DataType Table*.
NULL se privo di significato.

Unv Value: Valore non valido (big-endian). [32 bytes]
Il numero di byte significativo è dato dal campo *Datatype.Length* ottenibile tramite il *Datatype_id* tramite la *DataType Table*.
NULL se privo di significato.
(e.g. Valore privo di significato, Grandezza non ancora disponibile)

Description: Campo ASCII con la descrizione della variabile. [128 bytes].
(e.g. umidità relativa)

[3] Compilata manualmente durante la fase di realizzazione del progetto.

UNIT_TABLE[4]

Tabella che codifica le unità di misura utilizzate.

UNIT_ID	SI_NAME	EXPONENT
smallint unsigned	varchar [128]	smallInt

Unit_id: Codice della unità di misura (key univoca generata dal database). [1 bytes]

SI_name: Unità di misura espressa col nome della grandezza nel sistema SI. [128 bytes] (e.g. Ampere)

Exponent: Fattore moltiplicativo 10^X. [int] (e.g. -2 : unità di misura in centesimi)

e.g.

[1, Volt, 3]: unità di misura in kV

[2, Ampere, -3]: unità di misura in mA

HOST_OBJECTS_TABLE[5]

Tabella utilizzata per mettere in relazione un gateway con i relativi dispositivi.

HOST_ID	OBJ_ID

Tale corrispondenza è 1 a molti, in quanto ad un singolo gateway possono corrispondere più dispositivi.

[4] Compilata manualmente durante la fase di realizzazione del progetto.
[5] Compilata manualmente durante la fase di realizzazione del progetto.

OBJECT_VARIABLES_TABLE[6]

Tabella utilizzata per mettere in relazione un dispositivo con le relative variabili.

OBJ_ID	VAR_ID

Tale corrispondenza è 1 a molti, in quanto ad un singolo dispositivo possono corrispondere più variabili.

e.g.

Ad un multi-sensore (obj_id = 0000Ah) sono associate sia una variabile temperatura (var_id = 0007h) che una variabile pressione (var_id = 0008h)

000Ah 0007h

000Ah 0008h

BUILDING_TABLE[7]

Tabella che contiene l'indicazioni degli edifici usati nella sperimentazione AALISABETH.

BUILDING_ID	EDIFICIO	APPARTAMENTO	GEO_COORDINATES
smallint unsigned	varchar[256]	varchar[256]	Varchar [128]

Building_id: Identificativo univoco. [2 bytes]

Edificio: Campo ASCII con la descrizione dell'edificio. [256 bytes]

Appartamento: Campo ASCII con la descrizione dell'appartamento. [256 bytes] NULL se non applicabile.

GeoCoordinates: Campo ASCII con i parametri di localizzazione: longitudine/latitudine dell'edificio. NULL se non disponibile.

LOCALIZATION_TABLE[8]

Tabella che associa un indirizzo univoco ad ogni possibile localizzazione fisica dello stesso all'interno dell'appartamento dell'utente AALISABETH.

LOC_ID	BUILDING_ID	STANZA	ZONA	USER_ID
Usmallint	Usmallint	varchar[256]	varchar[256]	Usmallint

Loc_id: Identificativo della localizzazione. [2 bytes]

[6] Compilata manualmente durante la fase di realizzazione del progetto.
[7] Compilata manualmente durante la fase di realizzazione del progetto.
[8] Compilata manualmente durante la fase di realizzazione del progetto.

Building_id: Identificativo edificio [2 bytes]

Stanza: Campo ASCII con la descrizione della stanza.
NULL se si ha tratta di un dispositivo mobile. [256 bytes]

Zona: Campo ASCII con la descrizione della zona.
NULL se si ha tratta di un dispositivo mobile. [256 bytes]

User_id: Identificativo persona se dispositivo indossabile ad esclusivo utilizzo
NULL se non applicabile

OBJECTS_LOCALIZATION_TABLE[9]

Tabella utilizzata per mettere in relazione un dispositivo con le relative variabili.

LOC_ID	OBJ_ID

Tale corrispondenza è 1 a molti, in quanto ad una singola locazione possono corrispondere più dispositivi.

DATATYPE_TABLE[10]

Tabella utilizzata per la descrizione dei tipi di dati nativi utilizzati

DATA_CODE	NAME	DATALEN	MIN_VALUE	MAX_VALUE
tinyint unsigned	varchar[64]	smallint unsigned	varchar[64] (nibble ASCII)	varchar[64] (nibble ASCII)

e.g.

01	BOOLEAN	1	FALSE	TRUE
02	UCHAR	1	0	255
03	CHAR	1	-128	127
04	UINT2	2	0	65,535
05	INT2	2	–32,768	32,767
06	UINT4	4	0	4,294,967,295
07	INT4	4	–2,147,483,648	2,147,483,647
08	UINT8	8	0	18,446,744,073,709,551,615
09	INT8	8	–9,223,372,036,854,775,808	9,223,372,036,854,775,807
10	FLOAT	4	3.4E - 38	3.4E + 38
11	DOUBLE	8	1.7E - 308	1.7E + 308
12	TIMESTAMP	6	NULL	NULL
13	STRING	255	NULL	NULL
14	BLOB	8	NULL	NULL

[9] Compilata manualmente durante la fase di realizzazione del progetto.
[10] Compilata manualmente durante la fase di realizzazione del progetto.

TIMESTAMP DATATYPE

Il timestamp nelle API viene rappresentato come una struttura di 6 bytes. Nel database verrà utilizzato il tipo natio DATETIME.

```
struct timestamp_t
{
      uint8_t  year;        // from 2000
      uint8_t  month;       // [1 - 12]
      uint8_t  day;         // [1 - 31]
      uint8_t  hours;       // [1 - 23]
      uint8_t  minutes;     // [0 - 59]
      uint8_t  seconds;     // [0 - 59]
};
```

RESULT_CODE_TABLE[11]

Tabella contenente i result_code che vengono restituiti dal DB-Service.

RESULT_CODE	DESCRIPTION
tinyint unsigned	varchar[128]

e.g.

00	NO_ERROR
01	DB_ERROR
02	…

STATUS_CODE_TABLE[12]

Tabella contenente i status_code dei vari dispositivi (utilizzati nella Status Table).

STATUS_CODE	DESCRIPTION
tinyint unsigned	varchar[128]

e.g.

00	OPERATIVE_ONLINE
01	OFFLINE
02	UNCONFIGURED
03	OUT_OF_SERVICE
04	DEBUG_INFO
05	…

[11] Compilata manualmente durante la fase di realizzazione del progetto.
[12] Compilata manualmente durante la fase di realizzazione del progetto.

FAULT_CODE_TABLE[13]

Tabella contenente i fault_code dei vari dispositivi (utilizzati nella Status Table).

FAULT_CODE_ID	DESCRIPTION
tinyint unsigned	varchar[128]

e.g.

00	NONE
01	MEASURE_WRONG_READING
02	…

HOST_CMD_TABLE[14]

Ogni gateway fornirà una propria tabella di codici comandi e relativa descrizione.

HOST_ID	CMD_CODE	PARAMETER_BYTES_LENGTH	DESCRIPTION
smallint unsigned	tinyint unsigned	smallint unsigned	varchar[128]

[13] Compilata manualmente durante la fase di realizzazione del progetto.
[14] Compilata manualmente durante la fase di realizzazione del progetto.

PROTOTIPO API

Viene proposta una API che permette la comunicazione tra un host (gateway) ed il DB-Service. Il sistema si basa sulla scrittura di 3 tabelle fondamentali, le quali conterranno tutte le informazioni acquisite dal campo della rete AALISABETH.

DATA_TABLE

Si tratta della **tabella principale**. Periodicamente tutti i gateway scriveranno su tale tabella le informazioni riguardanti i loro dispositivi (e le variabili/attributi dei dispositivi stessi).

Quando un gateway invia un comando di scrittura di dati al DB-Service, quest'ultimo scriverà un nuovo record (nuova riga) sulla data table.

record_id	record_timestamp	Host_id	Obj_id	Var_id	user_id	timestamp	data	Int_value	Real_value
Uint	datetime	smallint unsigned	smallint unsigned	smallint unsigned	smallint unsigned	datetime	varchar [256] (nibble ASCII)	bigint	double

record_id Int, progressivo univoco

record_Timestamp: Timestamp inserito dal DB-Service quando il record viene scritto sulla tabella. [8 bytes]

host_id: Identificativo univoco del gateway [2 bytes]

obj_id: Identificativo univoco del dispositivo.[2 bytes]

var_id: Identificativo della variale del sensore obj_id che si vuole modificare nel database [2 bytes]

user_id: Identificativo della persona a cui e' associato il dato. NULL se non applicabile [2 bytes]

timestamp: Timestamp relativo al dato. Rappresenta il timestamp che viene fornito dal gateway (se disponibile) e non il timestamp che viene aggiunto dal DB-Service. NULL se non disponibile. [8 bytes]

data: Payload del messaggio (big-endian). Rappresentazione in nibble ASCII fino a 128 bytes (2 caratteri per byte). [varchar(256)]. In caso di data_type STRING la rappresentazione ASCII e' diretta e consente stringhe sino a 256 caratteri

int_value Bigint, codifica intera del payload (se applicabile)

real_value Double, codifica double del payload(se applicabile)

user_id Int, riferimento alla chiave della tabella PERSONAL_DATA

USER_TABLE

Individua la persona che partecipa alla sperimentazione

USER_ID	ANAGRAFICA_ID	SURNAME	NAME
datetime	smallint unsigned	Varchar(128)	Varchar(128)

User_id Int, progressivo univoco

Anagrafica_Id Int, contiene il rifermento esterno all'anagrafica

Surname varchar(128),contiene il nome dell'anagrafica

Name varchar(128), contiene il cognome dell'anagrafica

BLOB_TABLE[15]

Contiene insieme dati generici di grosse dimensioni (>128 bytes o stringhe > 256 caratteri)

BLOB_ID	DATA
smallint unsigned	Blob[65535 bytes]

Usata per insieme di dati non rappresentabili in 'data_table'. In tale situazione la 'data_table' contiene un 'datatype_id' di tipo 'blob' e come 'int_value' il blog_id a questa tabella

[15] Compilata manualmente durante la fase di realizzazione del progetto.

AAL_WRITE DATA (AAL Message Code 00h)

`AalWriteData(host_id, obj_id, var_id, user_id, timestamp, datalen, data)`[16]

Host_id: Identificativo univoco del gateway [2 bytes]

Obj_id: Identificativo univoco del dispositivo [2 bytes]

Var_id: Identificativo della variale del sensore obj_id che si vuole modificare nel database [2 bytes]

User_id: Identificativo della persona a cui e' associato il dato. 0x0000 se non applicabile [2 bytes]

Timestamp: Timestamp relativo al dato. Rappresenta il timestamp che viene fornito dal gateway (se disponibile) e non il timestamp che viene aggiunto dal DB-Service. [8 bytes]

Datalen: Lunghezza in byte del payload.

Data: Payload del messaggio (big-endian). [Datalen bytes]

`AalWriteDataResult(host_id, obj_id, result_code)`[17]

Host_id: Identificativo univoco del gateway che ha inviato il messaggio. [2 bytes]

Obj_id: Identificativo univoco del dispositivo che ha inviato il messaggio [2 bytes]

Result_code: Codice di errore. [1 byte]

Una volta ricevuto il comando, il DB-Service memorizzerà nella **DATA TABLE** del database l'intero comando ricevuto inserendo in testa un timestamp [8 bytes] relativo all'istante di inserimento del dato nel database.

RECORD_TIMESTAMP	HOST_ID	…	DATA
[8]	[2]		varchar[256] (nibble ASCII)

[16] Ipotetica API sul client (gateway)
[17] Ipotetica API sul server (DB-Service)

STATUS_TABLE

Periodicamente tutti i gateway scriveranno su tale tabella le informazioni riguardanti lo stato dei loro dispositivi.

Quando un gateway invia un comando di scrittura dello stato al DB-Service, quest'ultimo scriverà un nuovo record (nuova riga) sulla status table.

RECORD TIMESTAMP	HOST_ID	OBJ_ID	STATUS TIMESTAMP	STATUS_CODE	FAULT_CODE	FREETEXT
datetime	[2]	Usmallint	datetime	tinyint unsigned	tinyint unsigned	varchar[128]

Record_Timestamp: Timestamp inserito dal DB-Service quando il record viene scritto sulla tabella. [8 bytes]

Host_id: Identificativo univoco del gateway [2 bytes]

Obj_id: Identificativo univoco del dispositivo [2 bytes]

Status_ts: Timestamp dello status. [8 bytes]

Status_code: Codice dello stato [1 byte]

Fault_code: Codice del fault [1 byte]

Freetext: Campo ASCII descrittivo. [varchar(128)]

AAL_WRITE_STATUS (AAL Message 01h)

```
AalWriteStatus(host_id, obj_id, status_ts, status_code, fault_code,
               freetext)
```
[18]

Host_id: Identificativo univoco del gateway [2 bytes]

Obj_id: Identificativo univoco del dispositivo [2 bytes]

Status_ts: Timestamp dello status. [8 bytes]

Status_code: Codice dello stato [1 byte]

Fault_code: Codice del fault [1 byte]

Freetext: Campo ASCII descrittivo. [128 bytes]

```
AalWriteStatusResult(host_id, obj_id, result_code)
```
[19]

Host_id: Identificativo univoco del gateway [2 bytes]

Obj_id: Identificativo univoco del dispositivo [2 bytes]

Result_code: Risultato dell'operazione. [1 byte]

[18] Ipotetica API sul client (gateway)
[19] Ipotetica API sul server (DB-Service)

Una volta ricevuto il comando, il DB-Service memorizzerà in una apposita **STATUS TABLE** nel database l'intero comando ricevuto inserendo in testa un timestamp [8 bytes] relativo all'istante di inserimento del dato nel database.

RECORD_TIMESTAMP [8]	HOST_ID [2]	…	FREETEXT [N]

CMD_TABLE

La tabella CMD_TABLE permette ai scambiare dei comandi tra i gateway oppure ad un gateway stesso di registrare i propri comandi eseguiti. Tali comandi dovranno essere implementati e forniti dagli implementatori dei singoli gateway. Ad ogni comando registrato verrà assegnato automaticamente un *cmd_id* (auto-incrementale) univoco, oltre al timestamp di registrazione.

Quando un gateway invia un comando di scrittura dello stato al DB-Service, quest'ultimo scriverà un nuovo record (nuova riga) sulla cmd_table.

CMD ID	RECORD TS	HOST_ID SOURCE	HOST_ID DEST	OBJ ID	VAR ID	CMD CODE	PARAMS LEN	PARAMS	DONE FLAG	DONE TS	DONE HOST_ID
int	datetime	smallint unsigned	smallint unsigned	smallint unsigned	smallint unsigned	tinyint unsigned	smallint unsigned	varchar[512] nibble ASCII	tinyint unsigned	datetime	smallint unsigned

Record_Timestamp: Timestamp inserito dal DB-Service quando il record viene scritto sulla tabella. [8 bytes]

Host_id_source: Identificativo univoco del gateway sorgente. [2 bytes]

Host_id_dest: Identificativo univoco del gateway destinazione. [2 bytes]

Obj_id: Identificativo univoco del dispositivo [2 bytes]

Var_id: Identificativo della variale/attributo del dispositivo che si è letta dal dispositivo obj_id. NULL se il comando è a livello host o a livello oggetto. [2 bytes]

Cmd_code: Codice dello comando [1 byte]

Parameters_len: Lunghezza in byte della struttura parametri [2 bytes]

Parameters: Struttura parametri da passare al comando. I dati all'interno della struttura devono essere convertiti in big-endian. [varchar(512)]

Done_flag: Valore indicante se il comando è stato eseguito. [1 byte]

Done_Ts: Timestamp relativo all'esecuzione del comando. [8 bytes]

Done_host_id: Codice del gateway che ha eseguito il comando. [2 bytes]

AAL_WRITE_CMD (AAL Message 02h)

API utilizzata per trasmettere un comando tra 2 gateway oppure per loggare i comandi eseguiti da un singolo gateway (logging).

```
AalWriteCommand(host_id_source, host_id_dest, obj_id, var_id, cmd_code,
                parameters_len, parameters, doneflag)
```
[20]

Host_id_source: Identificativo univoco del gateway sorgente. [2 bytes]

Host_id_dest: Identificativo univoco del gateway destinazione. [2 bytes]

Obj_id: Identificativo univoco del dispositivo . 0x0000 se il comando è a livello host. [2 bytes]

Var_id: Identificativo della variale/attributo del dispositivo che si è letta dal dispositivo obj_id. 0x0000 se il comando è a livello host o a livello oggetto. [2 bytes]

Cmd_code: Codice dello comando [1 byte]

Parameters_len: Lunghezza in byte della struttura parametri [2 bytes]

Parameters: Struttura parametri da passare al comando. I dati all'interno della struttura devono essere convertiti in big-endian. [n bytes]

Done_flag: Valore da impostare a True se si vuole registrare l'esecuzione di un comando già eseguito da parte dello stesso host (logging: Host_id_source = Host_dest_source). [1 byte]

Conferma scrittura sul database.

```
AalWriteCommandResult(host_id, obj_id, result_code)
```
[21]

Host_id: Identificativo univoco del gateway [2 bytes]

Obj_id: Identificativo univoco del dispositivo [2 bytes]

Result_code: Risultato dell'operazione. [1 byte]

[20] Ipotetica API sul client (gateway)
[21] Ipotetica API sul server (DB-Service)

AAL_GET_NEXT_CMD_UNDONE (AAL Message 03h)

Permette di ottenere il prossimo comando da eseguire per il gateway *host_id*. Restituisce *cmd_id = 0* se non ci sono comandi da eseguire.

`AalGetNextCmdUndone(host_id_dest)`[22]

Host_id_dest: Identificativo univoco del gateway destinazione. [2 bytes]

`AalGetNextCmdUndoneResult(cmd_id, host_id_dest, obj_id, var_id, cmd_code, parameters_len, parameters)`[23]

Cmd_id: Identificativo univoco del comando che deve essere eseguito del gateway destinazione. [4 bytes]

Host_id_dest: Identificativo univoco del gateway destinazione del comando. [2 bytes]

Obj_id: Identificativo univoco del dispositivo destinazione del comando [2 bytes]

Var_id: Identificativo della variale che del dispositivo obj_id. [2 bytes]

Cmd_code: Codice dello comando [1 byte]

Parameters_len: Lunghezza in byte della struttura parametri [2 bytes]

Parameters: Struttura parametri da passare al comando. I dati all'interno della struttura devono essere convertiti in big-endian. [n bytes]

AAL_SET_DONE_CMD (AAL Message 04h)

Permette di impostare sul database l'avvenuta esecuzione di un comando. Il DB-Service imposterà a True il campo DONE_FLAG relativo.

`AalSetDoneCmd(host_id_dest, cmd_id)`[24]

Host_id_dest: Identificativo univoco del gateway destinazione. [2 bytes]

Cmd_id: Identificativo univoco del comando che è stato eseguito [4 bytes].

`AalSetDoneCmdResult(host_id_dest, cmd_id, result_code)`

Host_id_dest: Identificativo univoco del gateway destinazione. [2 bytes]

Cmd_id: Identificativo univoco del comando che è stato eseguito. [4 bytes]

Result_code: Risultato dell'operazione. [1 byte]

[22] Ipotetica API sul client (gateway)
[23] Ipotetica API sul server (DB-Service)
[24] Ipotetica API sul client (gateway)

REGISTRAZIONE COMANDI E SCAMBIO COMANDI TRA GATEWAY

L'invio di comandi tra due dispositivi che si trovano in due sottosistemi differenti (quindi con gateway differenti) avviene tramite il DB-Service.

- Quando un gateway deve inviare un comando ad dispositivo appartenente ad un altro gateway o per loggare i comandi eseguiti, viene generata una richiesta al DB-Service tramite il comando AalWriteCmd(…).
  Il DB-Service scrive sul database un nuovo record nella tabella **CMD_TABLE.**
  Ogni record è strutturato del seguente modo:

CMD_ID	RECORD_TS	…	DONE_FLAG	DONE_TS	DONE_HOST_ID
[4]	[8]	…	[1]	[8]	[2]

Cmd_id: Identificativo univoco del comando che è stato eseguito. [4 bytes]

Record_ts: Timestamp di inserimento del record. [8 bytes]

Done_flag: Flag indicante l'esecuzione del comando. [1 byte]

Done_ts: Timestamp di esecuzione del comando. [8 bytes]

Done_host_id: Identificativo univoco del gateway esecutore del comando. [2 bytes]

- Ogni gateway interroga periodicamente il DB-Service tramite il comando AalGetNextCmdUndone(…) in modo da verificare la presenza di un comando a lui indirizzato.
- Ottenuto il comando, il gateway lo esegue e comunica l'esecuzione del comando al DB-Service tramite la funzione AalSetDoneCmd(…).
  Il DB-Service imposta le flag nel relativo record della CMD_TABLE.

ESEMPIO

Il gateway 0001h deve registrare sul database la misura 00A154h della variabile 0034h del dispositivo FE80F0D01002005100000000000000004h(obj_id:0x0003).

Il tipo di dato della variabile è un uint4(datatype_id 06h).

Il timestamp dell'acquisizione è 2013/07/16 14:30:02 => 0D07100E1E02h

Dunque il comando AAL_WRITE_DATA (00h) a livello socket sarà:

```
AAAA00000100030034 0D07100E1E02 04 00A154
```

In caso di risposta positiva (result_code 00h), questa a livello socket sarà:

```
AABB000001000300
```

Descrizione variabili oggetti

Ferma restando la necessità di una descrizione formale dettagliata dei singoli oggetti, di seguito sono riportate le informazioni relative alla rappresentazione mediante variabili degli oggetti da utilizzarsi all'interno del sistema.

Tabella 1 – Rilevatore di Fumo e Monossido di carbonio (CO)

Variable ID	**Campo**	**Descrizione**	**Valori consentiti**
0	Int_value (tinyint)	Dispositivo OK	0 – OK (è lo *status* del dispositivo e non una sua variabile: dovrebbe essere collocato nella STATUS_CODE_TABLE) 1 – Guasto
1000	int_value smallint unsigned	Stato di allarme	0x0020 – Nessun Allarme 0x0021 – Allarme CO 0x0022 – Allarme Fumo 0x0023 – Allarme Fumo e CO
1001	int_value (Boolean)	Stato Batteria	0 – OK 1 – Batteria scarica

Descriviamo il rilevatore di Fumo e monossido di carbonio secondo la nuova struttura del database/DB-Service proposta.

Ipotiziamo che l'oggetto rilevatore di fumo sia caratterizzato da un obj.ipv6_address: FE80F0D01002005100000000000000001 (con corrispondente obj_id 0x0003), e quindi da una riga nella tabella OBJECT_TABLE.

e.g.

3	FE80F0D01002005100000000000000001	MAC Srl	Modulo K2263V1 Rilevatore di Fumo e Monossido di carbonio (CO)
Obj_id	Ipv6_address nibble ASCII	Manufacture	Description

Le variabile “Allarme” viene definita nella VARIABLE_TABLE (e potrebbe essere condivisa anche con un altro dispositivo analogo).

e.g.

1000	Allarme Gas	04	01	0020h	0023h	FFFFh	Stato di allarme 0x0020 – Nessun Allarme 0x0021 – Allarme CO 0x0022 – Allarme Fumo 0x0023 – Allarme Fumo e CO
VAR_ID	NAME	DATATYPE ID smallint unsigned	UNIT_ID (adimensional)	MIN VALUE	MAX VALUE	UNV VALUE	DESCRIPTION

La variabile “Stato Batteria” verrà definita in maniera analoga nella VARIABLE_TABLE con la seguente riga:

e.g.

1001	Stato Batteria	01	01	0000h	0001h	FFFFh	Stato Batteria 0 – OK 1 – Batteria scarica
VAR_ID	NAME	DATATYPE ID (Boolean)	UNIT_ID (adimensional)	MIN_ VALUE	MAX_ VALUE	UNV VALUE	DESCRIPTION

Le variabili verranno assegnate al dispositivo nella OBJECT_VARIABLES_TABLE

e.g.

3	1000
3	1001
obj_id	var_id

Lo stato sarà invece codificato nella STATUS_CODE_TABLE.

e.g.

Ipotizzando che attualmente il dispositivo sia operativo, il suo status sarà rappresentato dallo StatusCode OPERATIVE_ONLINE (00h) nella STATUS_TABLE (aggiornata dai gateway ai cambiamenti di stato dei suoi dispositivi controllati).

In maniera analoga potranno essere modellati gli altri dispositivi.

Tabella 2 – Rilevatore allagamento

Variable ID	**Campo**	**Descrizione**	**Valori consentiti**
0	Int_value	Dispositivo OK	0 – OK 1 – Guasto

1000	int_value	Stato di allarme	0x0020 – Nessun Allarme 0x0021 – Allagamento
1001	int_value	Stato Batteria	0 – OK 1 – Batteria scarica

Tabella 3 – Sensore di Movimento/Presenza

Variable ID	**Campo**	**Descrizione**	**Valori consentiti**
0	Int_value	Dispositivo OK	0 – OK 1 – Guasto
1000	int_value	Stato occupazione	0x0000 – Non Occupato 0x0001 – Occupato
1001	int_value	Stato Batteria	0 – OK 1 – Batteria scarica

Tabella 4 – Contatto magnetico

Variable ID	**Campo**	**Descrizione**	**Valori consentiti**
0	Int_value	Dispositivo OK	0 – OK 1 – Guasto
1000	int_value	Stato contatto	0x0020 – Chiuso 0x0021 – Aperto
1001	int_value	Stato Batteria	0 – OK 1 – Batteria scarica

Tabella 5 – Presa di corrente controllata

Variable ID	**Campo**	**Descrizione**	**Valori consentiti**
0	Int_value	Dispositivo OK	0 – OK 1 – Guasto
1000	int_value	Stato On-Off	0 – Off 1 – On
1001	int_value	Accumulatore potenza erogata	Espresso in kWh

1002	float_value	Richiesta istantanea di potenza	Espresso in kW
10000	int_value	COMANDO: Modifica stato On/Off	0 – Off 1 – On 2 – Commuta

Tabella 6 – Sensore Frigo (Fridge Box)

Variable ID	**Campo**	**Descrizione**	**Valori consentiti**
0	Int_value	Dispositivo OK	0 – OK 1 – Guasto
1000	float_value	Misura temperatura	Espressa in °C
1001	int_value	Misura umidità relativa	Espressa in %
1002	int_value	Stato Batteria	0 – OK 1 – Batteria scarica
1003	int_value	Stato porta	0 = Chiusa 1 = Aperta

Tabella 7 – Combinatore telefonico

Variable ID	Campo	Descrizione	Valori consentiti
1000	int_value	Status	0 – Nessuna chiamata in corso 1 – Chiamata canale 1 in corso 2 – Chiamata canale 2 in corso 4 – Chiamata canale 3 in corso 8 – Chiamata canale 4 in corso
10000	int_value	COMANDO: Modifica lo stato	0 – Interrompere chiamata in corso 1 – Avviare chiamata canale 1 2 – Avviare chiamata canale 2 4 – Avviare chiamata canale 3 8 – Avviare chiamata canale 4

Tabella 8 – Rilevatore di caduta e pulsante di emergenza

Variable ID	Campo	Descrizione	Valori consentiti
0	Int_value	Dispositivo OK	0 – OK 1 – Guasto
1000	int_value	Stato di allarme	0 – Nessun Allarme 1 – Allarme Caduta 2 – Chiamata 3 – Allarme di caduta e chiamata
1001	int_value	Stato Batteria	0 – OK 1 – Batteria scarica