UNIVERSITÀ DEGLI STUDI DI TORINO

Facoltà di Scienze Matematiche Fisiche e Naturali

Corso di laurea in Scienze dell'Informazione

TESI DI LAUREA

Sistema di controllo per "Casa Intelligente" aperto, configurabile e programmabile con particolare attenzione ad utenti anziani, disabili o lungodegenti

Candidato

Edoardo Frola

Relatore

Prof. **Albert Werbrouck**

Relatore Esterno

Dott. **Marco Mercinelli**

Anno Accademico 1994-95

Questo lavoro è stato svolto nel periodo 1 *Luglio 1994 - 30 Giugno 1995* presso i laboratori dello *CSELT*, Centro Studi E Laboratori Telecomunicazioni, di Torino, nell'unità di *Telemedicina e Telematica Sociale* diretta dal Dott. Marco Mercinelli.

È mio desiderio ringraziare il Prof. **Albert WERBROUK** e il Dott. **Marco MERCINELLI**, e nelle loro persone il Dipartimento di Scienze dell'Informazione e lo CSELT, per avermi offerto l'opportunità di realizzare questa Tesi e per la costante attenzione con cui ne hanno seguito l'evolversi.

Ringrazio anche il P.I. Renzo Bortignon, l'Ing. Enzo Contini, il Dott. Gianpaolo Costantino, l'Ing. Stefano Rizzetto e l'Ing. Giuseppe Scarpi, ovvero tutta l'unità di Telemedicina e Telematica Sociale, per la sopportazione che hanno dimostrato nei miei confronti in questi lunghi mesi.

Un ringraziamento particolare va poi a tutti coloro, primi fra tutti i miei genitori, che hanno atteso pazientemente per molto tempo che io arrivassi a questo ambito traguardo.

Indice del documento

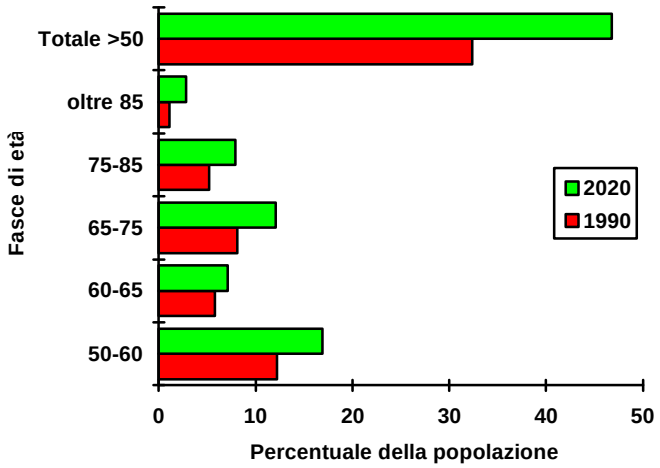
Indice delle tabelle

Indice delle figure

1. Introduzione

Nei paesi dell'Unione Europea è evidente la tendenza ad un incremento sempre più marcato della percentuale di persone anziane. Come si può osservare nella Tabella 1.1, tale tendenza porterà per la fine del secolo ad avere circa il 20% della popolazione europea con una età di oltre 65 anni. Si stima che questo gruppo di popolazione potrà raggiungere una percentuale di circa il 25% per il 2025.

Tabella 1.1: Aumento dell'età media della popolazione (studio CEE)



Anno	Percentuale della popolazione nella fascia di età indicata					
	50-60	60-65	65-75	75-85	oltre 85	Totale > 50
1990	12.2	5.8	8.1	5.2	1.1	32.4
2020	16.9	7.1	12.1	7.9	2.8	46.8

Tale "esplosione demografica" della popolazione anziana richiede nuove idee e soluzioni per garantire un adeguato livello di assistenza, dato che le attuali modalità per la fornitura di servizi sociali rivolti agli anziani rischiano di rivelarsi impraticabili, sia dal punto di vista organizzativo che da quello dei costi di esercizio. Un orientamento di tipo generale è verso la realizzazione di soluzioni integrate, basate sulle tecnologie attualmente disponibili, che forniscano all'anziano un supporto nelle attività quotidiane e che mettano a sua disposizione dei servizi di utilità generale a cui potere accedere anche a distanza. L'obiettivo è quello di consentire una maggiore indipendenza dell'anziano e di evitare, per quanto possibile, il ricovero in strutture assistenziali. L'idea della "Casa Intelligente" si situa in questo contesto, fornendo delle soluzioni abitative in grado di supportare le persone anziane nelle loro attività quotidiane.

Il termine "Casa Intelligente" è divenuto recentemente di uso comune per indicare un ambiente domestico opportunamente progettato e tecnologicamente attrezzato al fine di *rendere più agevoli le attività all'interno dell'abitazione* (quali apertura porte e finestre, gestione riscaldamento, accensione luci, attivazione elettrodomestici, ecc.), di *aumentare la sicurezza degli abitanti* (controllo fughe di gas, incendi, allagamenti, anti-intrusione, ecc.) e di *consentire la connessione a distanza con servizi di assistenza* (quali tele-soccorso, tele-monitoraggio, tele-medicina, ecc.) *e servizi di utilità generale* (tele-shopping, home-banking, servizi informativi,

ecc.). Il concetto di "Casa Intelligente" risulta utile per rispondere a differenti esigenze di una vasta fascia della popolazione, ma diventa particolarmente interessante per rispondere alle particolari esigenze degli anziani, dei disabili e dei lungo degenti.

Occorre anche osservare che il problema che si pone per l'assistenza di un numero sempre crescente di persone anziane si ripropone con maggior forza per quanto riguarda le persone disabili e lungo degenti. Anche queste categorie necessitano di un'assistenza spesso continua, che per gli stessi motivi osservati in precedenza, tende a diventare sempre meno fornibile con le modalità e la qualità richiesta. Quindi, anche per essi, un ambiente domestico capace di supportarli nelle operazioni della vita quotidiana che sarebbero loro rese difficoltose o negate da limitazioni fisiche o cognitive, può essere di innegabile aiuto per ripristinare, almeno in parte, la loro indipendenza.

L'invecchiamento della popolazione comporta un aumento delle patologie relative all'età avanzata, come la ridotta visibilità, la debolezza d'udito, le difficoltà motorie e così via. Si viene quindi a creare una sovrapposizione fra i disabili anziani e quelli appartenenti ad altre fasce d'età. Poiché dal punto di vista delle necessità e dell'assistenza non vi è motivo di distinguere gli uni dagli altri, faremo riferimento ai vari tipi di handicap senza specificare l'età del soggetto. È però necessario tenere in considerazione che l'aumento dell'età media della popolazione provoca un conseguente aumento degli handicap ad essa correlati ed è caratterizzata, in genere, da situazioni di handicap multipli.

In Tabella 1.2 sono riportati i principali handicap con il numero di disabili in Italia[1] ed i possibili supporti fornibili in ambiente di "Casa Intelligente". Per ogni ausilio è indicato il grado d'utilità (non rilevante, utile, molto utile, indispensabile) nei confronti del disabile. Per comodità ed essendo la categoria maggiormente interessata alla "Casa Intelligente", nel seguito del documento si utilizzerà il termine *persone anziane* per riferirsi anche ai disabili ed ai lungo degenti.

Sebbene risulti molto promettente, l'applicazione del concetto di "Casa Intelligente" alle esigenze degli anziani e dei disabili presenta anche delle difficoltà e dei potenziali rischi. Attualmente le varie componenti tecnologiche necessarie per la progettazione di una "Casa Intelligente" non sono concepite per un utilizzo semplice ed immediato da parte di "tutti" gli utenti, e quindi gli anziani potrebbero trovare delle difficoltà nell'utilizzo di questo tipo di tecnologia. Se si tiene conto che un'alta percentuale di anziani presenta anche varie forme di disabilità, i problemi pratici di utilizzo della tecnologia diventano ancora più evidenti. Non sono neanche trascurabili gli impatti e le difficoltà di ordine psicologico e culturale che una persona anziana può riscontrare nell'adattarsi ad un ambiente automatizzato. Devono inoltre essere assolutamente evitati, con opportuni interventi normativi, i rischi legati ad applicazioni del concetto di "Casa Intelligente" che possano portare ad un maggiore isolamento sociale degli anziani, oppure ad una riduzione della qualità dei servizi di assistenza. Occorre anche evitare che la "Casa Intelligente" possa essere concepita o percepita come un mezzo per un controllo indebito sulla vita privata dell'anziano.

Alcuni dei requisiti fondamentali di una "Casa Intelligente" per gli anziani sono quelli relativi alla necessità di una estrema modularità, flessibilità, adattabilità e semplicità delle soluzioni tecnologiche utilizzate. Infatti, le esigenze di automazione e gestione dell'ambiente domestico possono essere molto diverse per differenti individui, così come possono essere molto diverse le modalità con cui l'utente anziano può interagire e colloquiare con il sistema. Le esigenze possono anche variare in modo considerevole nel corso del tempo, con l'avanzare dell'età, l'insorgere di particolari patologie o in conseguenza di eventi traumatici. Ad esempio, le esigenze di una persona con difficoltà di mobilità sono molto diverse da quelle di un'altra che sia ipo-vedente, e saranno ancora diverse dalle esigenze di una persona con problemi di tipo cognitivo.

[1] *I dati relativi al numero di disabili in Italia per le varie tipologie di handicap fanno riferimento a statistiche CEE - COST 219.*

Tabella 1.2: Numero dei disabili in Italia e grado d'utilità dei vari ausili

	N° disabili in Italia	Telecomando ad infrarossi	Telecomando a radiofreq.	Comando vocale	Comando Eyegaze	Sintesi vocale	Ausili visivi	Ausili tattili
Non vedente	228.000	☺☺	☺☺☺	☺☺		☺☺☺		
Ipovedente	2.050.000	☺☺☺	☺☺☺	☺☺		☺☺☺		
Difficoltà di parola	285.000					☺☺☺		
Non udente	114.000						☺☺☺☺	☺☺☺☺
Debole d'udito	5.700.000						☺☺	☺☺
Senza l'uso delle gambe	390.000	☺☺☺	☺☺☺	☺☺				
Difficoltà di deambulazione	3.500.000	☺☺	☺☺	☺☺				
Senza l'uso d'entrambi le mani	130.000	☺	☺	☺☺☺				
Difficoltà d'uso delle mani	1.700.000	☺	☺	☺☺☺				
Tetraplegici	N. D.	☺	☺	☺☺☺	☺☺☺			
Carenza intellettiva	1.480.000					☺	☺	

☺☺☺☺ = Indispensabile ☺☺☺ = Molto utile ☺☺ = Utile ☺ = Utile con Adattatore (comando a soffio, joystick, ecc.)

Esistono al momento differenti soluzioni e dispositivi, disponibili sul mercato italiano, per l'automazione e la sicurezza domestica, il tele-soccorso e la tele-assistenza. Sono stati inoltre realizzati a livello europeo vari progetti di ricerca per la realizzazione di "Case Intelligenti" con funzionalità avanzate. In particolare il programma europeo TIDE[2] ha consentito di realizzare delle interessanti esperienze pilota di "Case Intelligenti" esplicitamente concepite in base alle particolari esigenze di utenti anziani e/o disabili[3].

Tuttavia, i sistemi scaturiti dai progetti di ricerca spesso si basano su soluzioni prototipali e su componenti difficilmente reperibili sul mercato italiano, oltre che di costo elevato. Viceversa, le soluzioni commerciali esistenti non sono in grado di adattarsi in modo sufficientemente flessibile alle differenti e variabili esigenze degli utenti anziani. Inoltre, soluzioni commerciali di diversi costruttori sono spesso "chiuse" e non sono in grado di integrarsi ed interagire fra di loro.

Già molto si può fare comunque per un gran numero di utenti anziani a partire dalla tecnologia esistente, se opportunamente scelta ed adattata, anche se le soluzioni possono non essere sempre ottimali. Un sistema in grado di rispondere alle esigenze degli anziani e integrabile con i componenti attualmente disponibile sul mercato è l'oggetto di questa tesi, sviluppata presso lo CSELT, il centro ricerche del gruppo STET, nell'ambito di una commessa TELECOM ITALIA.

Deve essere anche sottolineato come la realtà italiana abbia delle sue caratteristiche particolari che possono incidere sulla attivazione e diffusione di esperienze di "Casa

---

[2] *Technology Initiative for Disabled and Elderly*

[3] *Fra i progetti TIDE di maggiore interesse relativi alla "Casa Intelligente" per anziani e/o disabili è da citare in particolare il progetto ASHoRED (Adaptable Smarter Homes for Residents who are Elderly or Disabled) che ha posto in evidenza le esigenze di adattabilità e flessibilità delle soluzioni tecnologiche adottate, ed ha realizzato delle esperienze pilota in diversi Paesi europei. Al progetto hanno partecipato anche lo CSELT (Centri Studi e Laboratori Telecomunicazioni) del gruppo STET e la TELECOM ITALIA.*

Intelligente" per gli anziani. Ad esempio, la difficoltà di installare alcune nuove soluzioni tecnologiche in strutture abitative di non recente costruzione (ad esempio nei centri storici), la scarsa flessibilità delle strutture sociali per la promozione, la diffusione ed il coordinamento di soluzioni assistenziali avanzate, la carenza di una cultura tecnologica di base ed infine la carenza di centri pubblici multi-funzionali in grado di gestire i necessari servizi di consulenza, addestramento, assistenza, supervisione e manutenzione per gli utenti delle "Case Intelligenti".

Se si vogliono fornire a livello italiano delle soluzioni di "Casa Intelligente" per gli anziani adeguate e consistenti occorre quindi un impegno preciso ed un intervento programmato a vari livelli che veda il coinvolgimento delle strutture di assistenza sociale, degli enti di ricerca e delle industrie.

2. Il concetto di "Casa Intelligente" e le sue funzionalità

Con "Casa intelligente" si intende l'integrazione di vari *dispositivi per il controllo automatico di apparati domestici* (es. attuatori per il movimento di porte e finestre, attivazione di elettrodomestici, controllo dell'illuminazione, gestione del riscaldamento, ecc.), di *sensori dello stato dell'ambiente* (es. rilevatori di fumo, di gas, di perdite idriche, sensori di calore, sensori di movimento, ecc.), di *funzioni intelligenti di supporto* (es. agenda domestica, guida per la preparazione dei cibi, avviso per gli orari di assunzione medicine, ecc.) e di *sistemi di telecomunicazione per la connessione con centri servizi* per l'assistenza a distanza (es. tele-soccorso, tele-assistenza, tele-monitoraggio, ecc.).

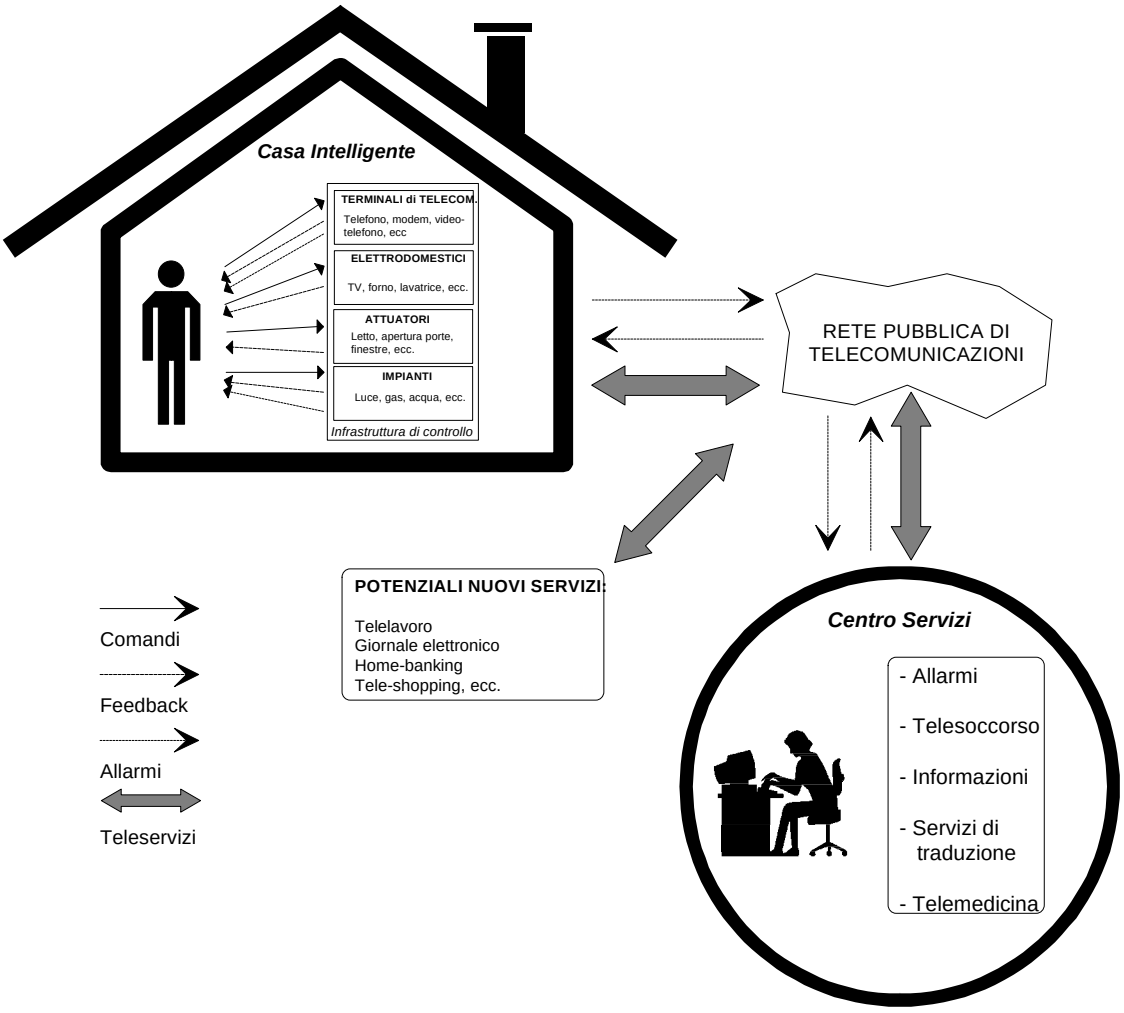


Figura 1: L'ambiente della Casa Intelligente

L'ambiente domestico così attrezzato ed integrato può essere gestito e controllato dall'utente tramite una opportuna *interfaccia utente* (es. telecomando, video tastiera, interfaccia vocale, ecc.), che realizza il colloquio (invio comandi e ricezione messaggi) con un *sistema intelligente di controllo*, basato in genere su di un microcomputer. I vari componenti della "Casa Intelligente" sono connessi con il sistema di controllo tramite un *sistema di interconnessione* di vari tipo (es. rete locale domestica, onde convogliate su rete elettrica, onde radio, ecc.).

Il sistema di controllo provvede ad interpretare ed eseguire i comandi dell'utente (es. apertura porta), a monitorare continuamente lo stato dell'ambiente (es. presenza di gas), a gestire automaticamente delle operazioni di controllo (es. regolazione temperatura) e ad attivare gli eventuali messaggi nei confronti dell'utente e dei servizi di assistenza remoti (es. allarme incendio). Alla attivazione di ogni comando, l'infrastruttura di controllo della "Casa intelligente" può fare giungere all'utente un segnale di avviso/conferma dell'operazione effettuata. Tale segnale può essere utile per fare comprendere all'utente l'attuale stato dei dispositivi domestici, in quanto egli potrebbe non averne la percezione diretta (ad esempio, un utente non vedente, attivata l'accensione delle luci non avrebbe la certezza di avere attivato il comando giusto e neanche che le luci funzionino).

2.1 I tele-servizi integrabili con la "Casa Intelligente"

L'integrazione di un ambiente di "Casa Intelligente" con i sistemi di telecomunicazioni consente la realizzazione di funzionalità avanzate di tele-soccorso, tele-assistenza e tele-monitoraggio.

Il *tele-soccorso* è genericamente un servizio che può essere attivato direttamente dall'utente (premendo pulsanti situati in posizioni fisse o tramite un trasmettitore portatile) per segnalare l'insorgere di un malore o la necessità di un intervento di assistenza[4]. La chiamata di tele-soccorso viene inoltrata direttamente dal sistema a una sequenza di numeri telefonici pre-impostati (es. parenti, Centro Servizi, 113). Appena un numero risponde può essere previsto l'invio di un messaggio vocale preregistrato e/o di un identificativo elettronico dell'utente riconoscibile automaticamente dal Centro Servizi. Occorre che il telefono dell'utente sia commutato direttamente in viva-voce dopo la chiamata, per consentire al chiamato di verificare direttamente e immediatamente la situazione del chiamante.

L'integrazione del tele-soccorso con le funzionalità della "Casa Intelligente" può consentire di ampliare notevolmente le potenzialità del tele-soccorso. Infatti, possono essere inviate in modo automatico dal sistema delle chiamate di emergenza anche in caso di allarmi rilevati dai sensori all'interno della casa (es. sensori di fumo, anti incendio, ecc.). Il sistema sarà anche in grado di chiamare automaticamente gli appropriati servizi di soccorso (es. pompieri, polizia, ecc.).

Il servizio di *tele-assistenza* consente genericamente di contattare l'utente periodicamente per verificarne lo stato di salute e fornire un supporto di tipo sociale e/o psicologico.

L'integrazione della tele-assistenza con le funzionalità della "Casa Intelligente" può consentire ad un Centro Servizi di controllare direttamente lo stato della Casa e di effettuare remotamente delle operazioni di assistenza (es. attivazione elettrodomestici, regolazione temperatura, preparazione di messaggi temporizzati per ricordare all'utente di eseguire particolari attività, ecc.).

Ovviamente occorrerà tenere conto accuratamente delle implicazioni etiche e legali poste dalla possibilità di controllo remoto della casa date dalla tele-assistenza: le funzioni di tele-assistenza dovranno potere essere inibite dall'utente e non dovranno potere essere attivate senza il suo consenso; potranno essere consentite in modo automatico solo a fronte di un allarme o sempre abilitate solo nel caso di particolari utenti a rischio. Dovrà inoltre essere garantito un adeguato livello di sicurezza affinché le funzioni di controllo remoto non possano essere attivate per errore da persone non autorizzate o in modo doloso da malintenzionati.

[4] *Occorre rilevare come attualmente non esistano delle soluzioni tecnologiche completamente affidabili in grado di rilevare automaticamente cadute o perdita di sensi dell'utente. Si tratta di un'area di ricerca estremamente importante, in cui l'individuazione di soluzioni adeguate potrebbe essere di estremo interesse per aumentare il livello di sicurezza in casa delle persone anziane o con particolari patologie.*

Il servizio di *tele-monitoraggio* consente genericamente l'invio di messaggi diagnostici sullo stato di funzionalità degli apparati della casa, consentendo di attivare prontamente le operazioni di manutenzione necessarie.

L'integrazione del tele-monitoraggio con le funzionalità della "Casa Intelligente" può consentire ad un Centro Servizi di effettuare, per quanto possibile, delle operazioni remote di test e controllo degli apparati, riconfigurare opportunamente il sistema oppure aggiornare i programmi del sistema di controllo. Anche in questo caso è essenziale che siano garantite delle funzioni di sicurezza affinché non sia possibile operare sulla Casa senza il consenso dell'utente e sia garantita una protezione contro l'accesso al sistema di persone non autorizzate.

La diffusione della "Casa Intelligente" potrebbe anche fornire nuove possibilità per i servizi di *Protezione Civile* che potrebbero inviare messaggi di allarme e istruzione agli utenti tramite i Centri Servizi. Le attuali procedure di allarme possono infatti essere inadeguate per alcune persone anziane; si pensi ad esempio agli anziani con problemi di udito che potrebbero essere informati di un pericolo tramite gli avvisatori luminosi e/o tattili (es. cuscino vibratore, orologio vibratore) della "Casa Intelligente".

L'ambiente di "Casa Intelligente" può favorire la realizzazione e la diffusione di altri tele-servizi di tipo generale, che possono usufruire della stessa infrastruttura telematica (in particolare se questa è basata su rete ISDN). Alcuni di questi tele-servizi potrebbero essere di particolare interesse per gli utenti anziani.

Il "*tele-shopping*" può consentire la gestione remota di ordinazioni di prodotti, consentendo l'eventuale consultazione di cataloghi elettronici o utilizzando cataloghi dotati di codici a barre per facilitare la scelta dei prodotti. L'integrazione di funzionalità di tele-shopping con l'ambiente della "Casa Intelligente" può consentire di mantenere un controllo sulla disponibilità di prodotti essenziali e fornire la possibilità di effettuare gli ordini automaticamente.

La diffusione di servizi di "*home-banking*" potrà consentire l'effettuazione remota di procedure bancarie.

Risulta anche possibile realizzare dei servizi di *accesso remoto agli uffici pubblici*, per la richiesta di documenti o l'espletamento di pratiche. Le tecnologie di lavoro cooperativo attualmente disponibili possono anche consentire la realizzazione di servizi pubblici per l'aiuto interattivo agli anziani nella compilazione di moduli e pratiche.

Il sistema di comunicazione della "Casa Intelligente" può inoltre essere integrato con *servizi informativi e banche dati* di vario tipo.

La diffusione e *consultazione di giornali e libri in forma elettronica* è facilmente realizzabile, con l'attuale tecnologia, e può rendere disponibili nuove modalità di informazione per gli anziani, quali ad esempio la lettura automatica dei testi tramite dispositivi di sintesi vocale.

I servizi di *tele-medicina* possono consentire un monitoraggio a distanza di alcuni parametri fisici, consentendo ad esempio il controllo remoto di un elettrocardiogramma, della pressione sanguinea o del livello di glucosio nei pazienti diabetici.

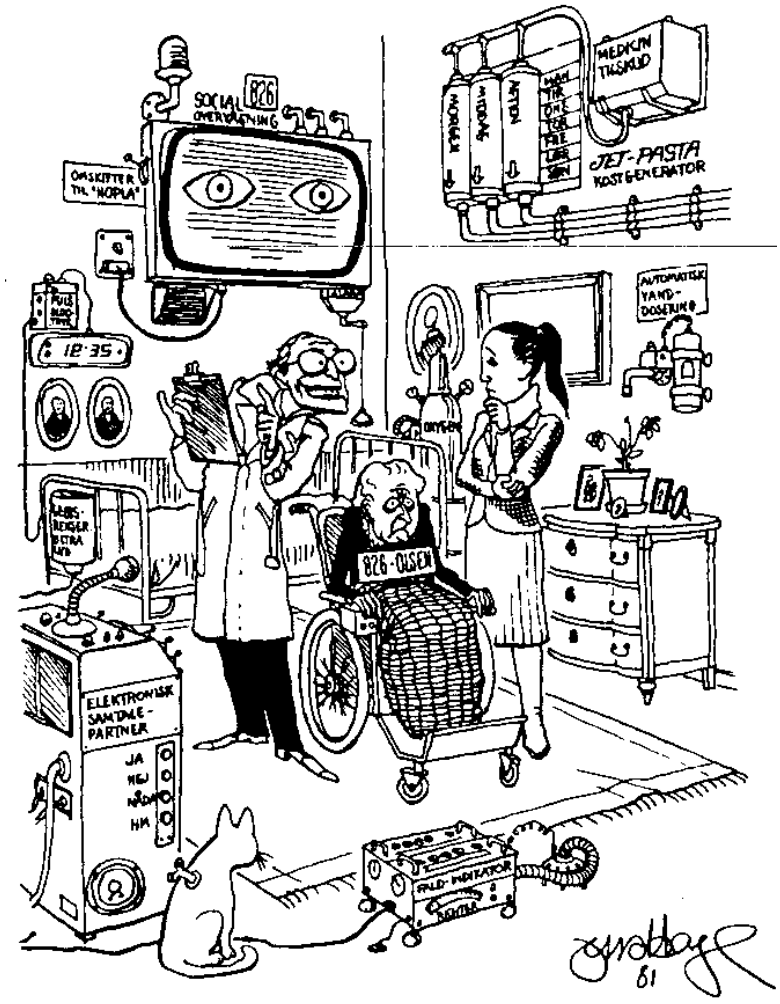
Infine la realizzazione di servizi di "*tele-training*" (insegnamento a distanza), può aprire nuove possibilità per l'aggiornamento continuo degli utenti e per l'esecuzione di programmi di riabilitazione.

L'integrazione nella "Casa Intelligente" di alcuni nuovi servizi di telecomunicazione può aprire ulteriori possibilità. Ad esempio la disponibilità di un *videotelefono* (installabile già da oggi sulla rete pubblica ISDN) potrà consentire agli utenti una migliore interazione con i servizi sociali e con gli altri utenti, contribuendo a rendere più "personali" le comunicazioni e a ridurre

l'isolamento. Il videotelefono può consentire inoltre una maggiore efficacia dei servizi sociali di assistenza, che potranno garantire un colloquio più frequente con gli anziani ed un controllo visivo delle loro condizioni. L'introduzione dei servizi di "*video-on-demand*" e la *televisione interattiva* potranno aggiungere delle nuove dimensioni anche per quanto riguarda le attività di intrattenimento.

3. I requisiti della Casa Intelligente per gli anziani

Nel definire i requisiti di una "Casa Intelligente" per gli anziani occorre anzitutto porre in evidenza il fatto che le esigenze appaiono molto più differenziate e variegate rispetto a quelle della popolazione più giovane. Infatti, sia le capacità fisiche che quelle cognitive necessarie per interagire con l'ambiente possono essere molto diverse per differenti individui e subire notevoli variazioni nel corso del tempo. Queste differenze si riflettono sulle funzionalità che si vogliono rendere disponibili nella "Casa Intelligente" e soprattutto sulle modalità con cui l'anziano può interagire con il sistema di controllo. Non sono inoltre da trascurare i fattori psicologici che possono generare un rifiuto o una scarsa confidenza nella tecnologia, rendendo la "Casa Intelligente" un fattore di frustrazione e di isolamento anziché l'occasione per una maggiore indipendenza.



"Grazie alla Casa Intelligente sarà necessario farle visita solo una volta all'anno, per regolare l'orologio..."

Figura 2: Quello che la Casa Intelligente per gli anziani NON deve essere

Fra i fattori fisici dell'utente che possono influenzare la configurazione della "Casa Intelligente" si possono citare le capacità motorie e di deambulazione, la ridotta coordinazione

dei movimenti e la presenza di eventuali tremori, la presenza di disabilità sensoriali quali ad esempio la riduzione della vista o dell'udito. Fra i fattori cognitivi, di particolare rilevanza possono risultare la riduzione delle capacità di memoria a breve termine, la difficoltà ad eseguire sequenze complesse di operazioni, la possibilità di insorgere di stati confusionali.

Nel seguito sono riportati dei requisiti di tipo generale per la realizzazione di una "Casa Intelligente" in grado di rispondere in modo ottimale alle esigenze degli utenti anziani[5]. I requisiti riportati sono il frutto della valutazione di varie esperienze pilota di "Casa Intelligente" per utenti anziani e/o disabili realizzati in vari Paesi Europei.

Sarà necessario che l'ambiente di "Casa Intelligente" per gli anziani risulti *facilmente e rapidamente modificabile, espandibile e riconfigurabile* per prevedere l'integrazione di nuovi moduli di controllo man mano che si manifestano delle nuove esigenze. L'architettura del sistema dovrà essere modulare e composta da componenti facilmente inseribili ed intercambiabili. Se, ad esempio, un anziano vede ridotta la sua mobilità a causa di un incidente o di una malattia, potrà essere necessario prevedere la possibilità di utilizzare un telecomando per gestire l'illuminazione e l'apertura di tutte le porte e finestre, oltre alla gestione remota del citofono. Se invece l'anziano soffre di crisi di confusione potrà essere necessario installare dei sensori di movimento che avvisino un Centro Servizi se l'utente esce di casa ad ore insolite.

L'interfaccia del sistema con l'utente dovrà prevedere diverse modalità di interazione per l'acquisizione dei comandi e la presentazione di messaggi, al fine di consentire l'utilizzo anche in presenza di una riduzione delle capacità di manipolazione, delle capacità sensoriali (vista e udito) e delle capacità cognitive. Se, ad esempio, si verifica una riduzione della vista sarà necessario utilizzare una interfaccia utente che si basi sull'invio di messaggi acustici (es. sintetizzatore vocale, cicalino d'allarme, ecc.). Viceversa se l'anziano presenta una diminuzione dell'udito i messaggi del sistema dovranno essere inviati tramite dispositivi di tipo visivo o tattile (es. lampeggiatori, cuscino vibratore, ecc.). Se l'anziano ha difficoltà nell'uso delle mani (es. problemi di coordinamento dei movimenti, paralisi degli arti superiori) l'interfaccia utente non potrà essere basata sull'uso di pulsanti ma dovrà utilizzare altre modalità di interazione (es. riconoscitore vocale, selezionatori a soffio, ecc.).

La tecnologia della "Casa Intelligente" deve risultare ergonomica e semplice da usare al fine di evitare confusione, timore e rifiuto da parte degli utenti. Occorre che l'interfaccia utente della "Casa Intelligente" per utenti anziani riduca al minimo il numero e la complessità dei comandi, rendendoli quanto più possibile simili al modo di operare comune e ricorrendo per quanto possibile a "metafore" o analogie con l'utilizzo di dispositivi noti. Occorre inoltre che il modo di comandare i diversi dispositivi sia il più possibile omogeneo. L'interfaccia deve inoltre "guidare" per quanto possibile l'utente nel suo utilizzo.

In base alle caratteristiche dell'utente si dovrebbe potere scegliere se presentare i messaggi all'utente in forma testuale, grafica, vocale, sonora o tattile, oppure con combinazioni di queste modalità. Devono potere essere scelte la dimensione, il colore e la luminosità dei simboli e del testo usati, in particolare se gli utenti sono ipovedenti o hanno problemi cognitivi (es. dyslexia). Il volume e il tono dei messaggi sonori possono influire sull'intelligibilità per i deboli di udito e possono anche avere un impatto psicologico notevole. Deve anche essere curata la posizione, la dimensione, il colore e l'eventuale marcatura tattile dei tasti e dei pulsanti in base alle esigenze di utenti ipovedenti, non vedenti o con problemi di coordinazione dei movimenti. In casi particolari devono potere essere considerati specifici adattamenti, ausili o metodi alternativi per accettare i comandi dagli utenti (es. riconoscitori vocali, switch attivati da alcune parti del corpo, selezionatori a soffio, ecc.).

Anche il tipo e *la complessità del livello di dialogo con l'interfaccia e l'aiuto fornito da questa dovrebbero essere configurabili*. Ad esempio, una persona che abbia problemi di

[5] *Il rispetto di tali requisiti garantirebbe comunque la disponibilità di sistemi di "Casa Intelligente" più usabili e di migliore qualità anche per utenti di tipo generale.*

coordinamento o movimento degli arti superiori può avere bisogno di una interfaccia che renda minime le necessità di movimento. Invece, una persona con problemi cognitivi può avere bisogno di una interfaccia che non richieda di ricordarsi azioni precedenti, non richieda di effettuare scelte complesse e renda sempre evidente le conseguenze delle azioni effettuate.

Occorre comunque tenere conto che la complessità del colloquio con l'utente deve essere sempre adeguata al suo livello cognitivo (possono ad esempio essere previsti dei livelli di utilizzo differenziati). Infatti, anche l'utilizzo di una interfaccia troppo semplificata e guidata può risultare eccessivamente pesante e frustrante per utenti con una adeguata preparazione e buone capacità cognitive.

L'interfaccia utente della "Casa Intelligente" deve risultare, per quanto possibile, sicura ed "a prova di errore". Occorre infatti che il sistema sia in grado di filtrare i comandi dell'utente e controllarne l'esecuzione in modo da limitare il rischio di errori. Il sistema dovrà essere configurato, a seconda delle capacità cognitive dell'utente, in modo da chiedere conferma a fronte di richieste non "ortodosse" (es. apertura di tutte le finestre in inverno).

La percezione che il sistema è in grado di "perdonare" gli errori accidentali può essere di grande aiuto per superare la paura, tipica dell'utente anziano (ma non solo), di fare un uso sbagliato o pericoloso della tecnologia, paura che può portare a percepire la "Casa Intelligente" come un fattore di inibizione anziché di indipendenza.

Il sistema deve prevedere delle funzioni automatiche di controllo per sopperire a eventuali dimenticanze dell'utente (es. provvedere ad avvertire l'utente se ha dimenticato aperta la porta di ingresso o lo sportello del frigorifero).

Il sistema dovrà attuare delle azioni automatiche di controllo per evitare il rischi di possibili incidenti (es. chiusura di un rubinetto al riempimento di una vasca, spegnimento del fornello della cucina al superamento di una soglia di calore o di un tempo massimo di cottura, blocco dell'erogazione dell'acqua a fronte di un corto circuito).

I componenti utilizzati nella realizzazione della "Casa Intelligente" devono prevedere delle funzioni intrinseche di sicurezza. Tutti gli automatismi della casa dovranno essere dotati di sensori in grado di evitare rischi di qualunque tipo (es. sensori di pressione anti-schiacciamento sugli automatismi di porte e finestre, avvisatori visivi e acustici dell'attivazione di un automatismo). Per alcuni soggetti potrà essere anche utile installare particolari dispositivi domestici di sicurezza, quali ad esempio i fornelli a induzione che riducono i rischi di bruciature.

Tutte le funzioni di controllo e di sicurezza devono sempre essere ridondanti e disponibili anche a fronte di malfunzionamenti del sistema o mancanze di corrente. Un errore del sistema o una mancanza di corrente non dovranno mai rendere impossibile l'effettuazione di una qualunque funzione di controllo (es. aprire porta) o inibire un sotto-sistema di allarme (es. antincendio). Tutti i dispositivi dovranno essere sempre attivabili anche manualmente o tramite controlli locali alternativi. I sensori ed i dispositivi di allarme dovrebbero potere continuare a funzionare autonomamente in qualunque condizione. La disponibilità di una unità di alimentazione di emergenza può essere prevista, ad esempio, per residenze comprendenti più "Case Intelligenti".

Particolari situazioni di utilizzo non devono potere inibire le funzioni di controllo e di sicurezza. Se, ad esempio, la "Casa Intelligente" prevede l'invio di allarmi ad un Centro servizi tramite la linea telefonica, tale funzionalità non dovrà essere inibita dalle telefonate personali dell'utente o da un telefono lasciato sconnesso; il sistema dovrà invece essere in grado di acquisire il controllo della linea oppure dovrà essere prevista una linea telefonica aggiuntiva per le funzioni di tele-soccorso e tele-allarme.

Il sistema dovrebbe essere in grado di monitorare in modo continuo la funzionalità dei suoi componenti, avvertendo automaticamente un Centro servizi in caso di malfunzionamenti.

Deve essere previsto un servizio di controllo e manutenzione periodica del sistema. Risulta evidente l'inutilità di funzioni sofisticate di controllo e di sicurezza se queste non sono affidabili e se non ci si può rendere conto della loro effettiva funzionalità. L'utente anziano deve inoltre sviluppare e mantenere una sufficiente confidenza nella affidabilità del sistema al fine di poterlo accettare e sentirsi a proprio agio nel suo utilizzo.

L'attivazione delle funzioni della "Casa Intelligente" deve potere essere effettuata solo dall'utente oppure da personale autorizzato, garantendone la sicurezza. Sarà necessario prevedere delle funzioni di sicurezza contro un uso non autorizzato delle funzioni della "Casa Intelligente" (es. l'apertura automatica di una porta o finestra dall'esterno da parte di un malintenzionato). Occorre inoltre garantire che diversi sistemi di controllo di "Case Intelligenti" vicine non interferiscano fra di loro. A tale riguardo i sistemi di connessione basati su onde radio (sconsigliabili) o raggi infrarossi dovranno essere opportunamente crittografati. Il sistema di controllo dovrà inoltre essere opportunamente schermato ed immune da interferenze elettromagnetiche.

Le funzioni di riconfigurazione della "Casa Intelligente" dovranno essere consentiti solo al personale di supporto o a utenti particolarmente esperti. Potranno essere previsti dei livelli di accesso al sistema protetti da opportuni accorgimenti di sicurezza, quali ad esempio parole chiave.

Deve essere previsto un adeguato supporto per la configurazione della "Casa Intelligente", in base alle esigenze dell'utente anziano ed un adeguato addestramento dell'utente all'uso delle sue funzionalità. In particolare gli utenti più anziani dovranno essere accompagnati e motivati nell'imparare ad usare la loro "Casa Intelligente". L'estrema configurabilità richiesta al sistema ed alla sua interfaccia impone che sia disponibile un servizio di assistenza continuo per una adeguata progettazione e installazione della Casa e delle sue funzionalità. Non è escluso che in alcuni casi si debba procedere per tentativi al fine di identificare l'insieme delle funzionalità e le modalità di interazione con il sistema che più si adattano all'utente. La disponibilità di manuali chiari, leggibili e con differenti livelli di dettaglio è anche indispensabile.

Da tutto quanto detto precedentemente risulta evidente che la progettazione di una "Casa Intelligente" per anziani richiede una accurata valutazione di vari fattori medici, psicologici, ergonomici, tecnici e socioculturali. Tale progettazione non può essere una attività che viene effettuata una volta per tutte, ma deve seguire passo-passo l'evoluzione della storia dell'anziano. *Risulta quindi necessario che la "Casa Intelligente" e le esigenze dei suoi abitanti siano visti in continua relazione e seguiti da un opportuno servizio tecnico-sociale (pubblico o privato), con del personale opportunamente addestrato per potere affrontare tutti gli aspetti coinvolti.*

4. Le soluzioni tecnologiche

4.1 Le architetture funzionali di riferimento per il controllo della Casa

Le due principali architetture funzionali che possono essere utilizzate per automatizzare e controllare i dispositivi domestici sono analizzate in questo capitolo. Il primo caso corrisponde a molte delle soluzioni attualmente disponibili sul mercato, ma non consente la realizzazione di funzioni di controllo complesse e realmente "intelligenti". Il secondo caso integra anche le soluzioni presenti nel primo ma presenta maggiori possibilità di controllo e consente di gestire in modo coordinato i vari dispositivi realizzando un reale ambiente "intelligente" all'interno della casa.

4.1.1 Casa assistita con infrastruttura di controllo distribuita

La possibilità di comandare e/o ricevere informazioni circa lo stato dei dispositivi domestici è il punto di partenza per consentire all'utente il controllo dell'ambiente domestico circostante.

In questo caso i dispositivi domestici (es. attuatori porte e finestre, accensione luci, elettrodomestici, ecc.) vengono comandati in modo indipendente uno dall'altro, o direttamente oppure mediante un telecomando (es. infrarosso, radiofrequenza, ecc.). Gruppi di dispositivi possono essere controllati da un'unica semplice centralina per realizzare sottosistemi dedicati a singole funzionalità (es. controllo anti-intrusione).

La caratteristica di questa configurazione è data dall'assenza di un sistema di controllo centralizzato. I vari dispositivi ed i sottosistemi operano quindi in modo scorrelato, ciascuno secondo una sua particolare logica interna. L'infrastruttura di controllo è quindi "distribuita" fra i vari sottosistemi. L'utente comanda separatamente i diversi dispositivi e sottosistemi e spesso anche le modalità di comando dei diversi sottosistemi possono risultare differenti o poco coerenti tra loro.

Questo tipo di sistemi di controllo è spesso caratterizzato dall'assenza di memoria sullo stato dei dispositivi e dall'assenza di "feedback" verso l'infrastruttura di controllo. Sarà quindi compito dell'utente verificare la corretta attuazione di un comando, tramite verifica personale oppure mediante un opportuno feedback "sensoriale" generato direttamente dal dispositivo (es. segnalazione luminosa, acustica, ecc.).

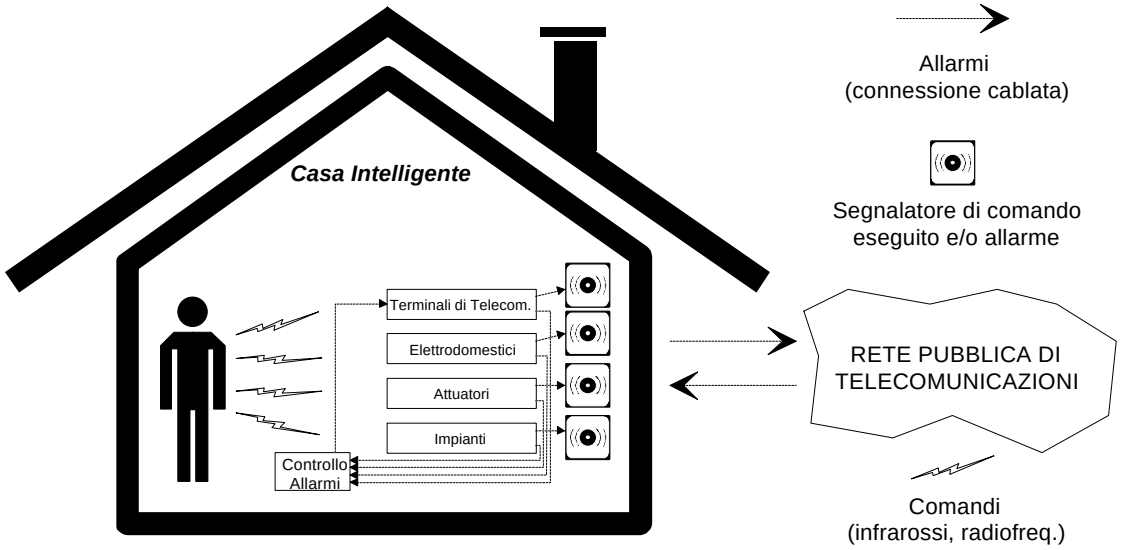


Figura 3: Struttura di Casa Assistita con controllo distribuito

4.1.2 Controllo integrato - Home Bus.

Questa configurazione è quella tipica di una vera "Casa Intelligente", ed è caratterizzata dall'interconnessione dei vari dispositivi domestici con una Unità di Controllo centralizzata (tipicamente un microcomputer), mediante un sistema di comunicazione che permetta l'interscambio di dati. Questa rete di comunicazione ha in genere una configurazione di tipo "bus", e quindi questo tipo di soluzione viene comunemente denominata "Home Bus".

Il sistema opera in modo integrato ed è in grado di coordinare il controllo di più dispositivi per potere realizzare delle funzionalità complesse. È inoltre in grado di supervisionare tutta l'operatività dell'ambiente domestico.

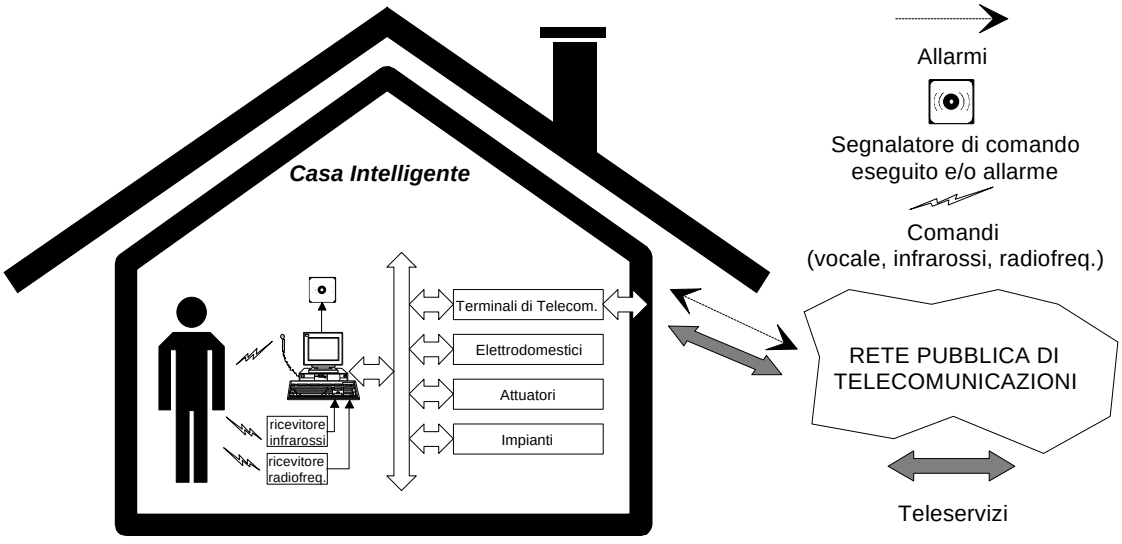


Figura 4: Struttura di Casa Intelligente con Sistema di Controllo integrato e Home Bus

Nel seguito faremo riferimento alla "Casa Intelligente" utilizzando quest'ultima concezione, che risulta essere la più completa ed include, tra l'altro, tutte le diverse componenti tecnologiche viste nella soluzione precedente.

4.2 La rete di interconnessione (Home Bus)

Alla rete di interconnessione sono connessi i vari dispositivi domestici tramite una opportuna interfaccia. Specifici protocolli di trasmissione sono utilizzati per condividere un canale fisico per la trasmissione di dati che è unico per tutti i dispositivi, ottenendo così una serie di canali logici che pongono in collegamento diretto l'unità di controllo con i singoli dispositivi.

Ogni dispositivo connesso alla rete domestica è identificato con un particolare codice univoco e la sua interfaccia è in grado di riconoscere tale codice e di rispondere a particolari comandi (es. accendi/spegni). Ad esempio, in una architettura di questo tipo l'accensione di una luce viene realizzata tramite l'invio di un particolare comando codificato all'indirizzo della lampada interessata. L'invio del comando di accensione può essere effettuato dall'Unità di Controllo centralizzata oppure da un "interruttore intelligente" che conosca il codice della lampada a cui è associato. Si noti che questo tipo di architettura consente una estrema flessibilità in quanto, ad esempio, per cambiare la lampada associata ad un interruttore è sufficiente cambiare il codice con il quale è programmato l'interruttore.

La rete domestica che interconnette tra loro i vari dispositivi e l'unità di controllo può essere fisicamente realizzata tramite diverse modalità. Queste possono essere il cavo coassiale, la trasmissione mediante onde convogliate o la connessione radio, sia essa analogica o digitale.

4.2.1 Cavo coassiale

Il cavo coassiale è unico e deve essere fisicamente fatto passare in tutti i locali costituenti l'ambiente domestico. A questo singolo cavo sono direttamente collegati, tramite opportune interfacce, tutti i dispositivi presenti nell'ambiente.

Nel caso di posa in opera in case di nuova costruzione è la soluzione meno costosa e di più semplice installazione, minimizzando il numero di cavi da installare (ad esempio non è necessario tirare direttamente dei fili fra gli interruttori e le lampade). Può invece richiedere una spesa di installazione elevata nel caso si voglia installare la rete in cavo coassiale in abitazioni esistenti, soprattutto se di vecchia costruzione. Ha il vantaggio di consentire velocità di trasmissione molto elevate.

4.2.2 Trasmissione in onde convogliate sulla rete elettrica

Si utilizza come mezzo di trasmissione la rete di distribuzione dell'energia elettrica già installata nella casa, che fornisce logicamente un unico "canale" comune in rame a partire dall'interruttore generale della Casa. I messaggi sono codificati e inviati su tale rete elettrica.

Alla rete possono essere connessi dei componenti di interfaccia, identificati univocamente da un codice, che sono in grado di ricevere e trasmettere dati codificati sui cavi della rete elettrica. Questi componenti di interfaccia possono comunicare fra loro e con il Sistema di Controllo, ricevere comandi, trasmettere allarmi o messaggi e possono anche essere integrati in dispositivi "driver" in grado di realizzare la gestione di apparecchiature domestiche complesse (es. televisore, videoregistratore, lavatrice, ecc.) . Nel caso più semplice l'interfaccia può essere interposta fra un dispositivo (es. lampada) e la rete elettrica ed abilitare il passaggio di corrente solo quando riceve l'indicazione del proprio codice.

La comunicazione mediante onde convogliate è la soluzione ideale e globalmente meno costosa per l'installazione in edifici esistenti. Richiede tuttavia un buon isolamento della rete elettrica e l'installazione di un dispositivo di filtraggio sull'attacco di rete della casa (per disaccoppiare fra loro case distinte). Può essere soggetta a disturbi da parte di alcuni dispositivi domestici (es. dimmer per luci alogene, motori). Ha velocità di trasmissione medie (fino a 14.4k bps).

4.2.3 Connessione radio (analogica o digitale)

Non richiede grandi lavori a livello della infrastruttura della casa. Le interfacce di controllo dei dispositivi sono però generalmente costose. Può essere soggetta a disturbi, soprattutto nel caso di trasmissioni analogiche.

La trasmissione con onde radio viene in genere utilizzata per connettere diversi spezzoni di rete domestica realizzati con altre tecnologie, quando non vi sia la possibilità di installare dei collegamenti diretti fra le diverse parti di una stessa residenza.

Le connessioni radio possono presentare dei rischi dal punto di vista della sicurezza e devono essere realizzate con degli accorgimenti (es. crittografia ed utilizzo di chiavi) per evitare interferenze accidentali o volute da parte di persone non autorizzate.

4.3 L'interfaccia utente

L'utente della "Casa Intelligente" dovrà poter attivare direttamente i vari dispositivi (in modo manuale oppure tramite un apposito dispositivo di comando remoto, quale per esempio un telecomando), oppure potrà gestirli in modo centralizzato, colloquiando con il sistema di controllo, quindi il microcomputer, tramite un'opportuna interfaccia utente.

Le funzionalità e le modalità di colloquio dell'interfaccia utente sono, come abbiamo visto in precedenza, un punto molto delicato per quanto riguarda la reale usabilità della "Casa Intelligente da parte di utenti anziani, in particolare se affetti da qualche tipo di disabilità. Il livello di complessità dell'interfaccia, il modo in cui si presenta all'utente e le modalità di colloquio possono determinare l'usabilità, l'accettazione o il rifiuto della tecnologia da parte dell'utente. L'estrema variabilità dei requisiti dei singoli utenti richiedono inoltre che l'interfaccia utente consenta il massimo della configurabilità rispetto al modo in cui vengono presentate le informazioni e con cui si accettano i comandi.

Purtroppo i sistemi commerciali attuali sono alquanto rigidi e non consentono il livello di configurabilità che è richiesto per consentirne l'usabilità da parte di alcune persone anziane. A volte anche solo la dimensione dei tasti, il colore dei messaggi o la necessità di eseguire sequenze di operazioni troppo lunghe possono risultare degli ostacoli insormontabili per l'utente.

In alcuni casi è possibile prevedere degli adattamenti o l'utilizzo di dispositivi di ausilio, quali possono essere, per esempio, tasti ingranditi, aggiunta di cicalini e lampeggiatori, integrazione di switch attivabili in modo particolare, integrazione di sintetizzatori e riconoscitori vocali. Questi adattamenti non sono tuttavia sempre fattibili, possono risultare costosi, e spesso richiedono una analisi ed una progettazione specifica per ogni singolo caso.

Alcuni sistemi prototipali, sviluppati in progetti di ricerca europei, presentano delle soluzioni di adattabilità e configurabilità molto interessanti. Tuttavia questi non hanno dato origine, per il momento, a prodotti di facile reperibilità sul mercato.

4.4 Connessione dell'interfaccia utente con il Sistema di Controllo

L'interfaccia utente può essere connessa direttamente all'unità di controllo, e in questo caso si possono prevedere più punti di accesso, posti nei diversi locali, collegati direttamente all'Home Bus. Un'altra soluzione può essere il rendere l'interfaccia trasportabile e connessa tramite dispositivi a raggi infrarossi oppure onde radio all'Home Bus.

Le connessioni dirette sono in genere le meno costose, ma sono adatte solo per persone che non abbiano eccessivi problemi di mobilità, perché per le altre può risultare difficoltoso avvicinarsi ai punti di accesso fissi.

Le connessioni a raggi infrarossi sono in genere affidabili e sicure ma presentano altri problemi, quali il limitato raggio d'azione, la necessità di avere un ricevitore a portata visiva e di dovere approssimativamente puntare il dispositivo verso di esso, l'impossibilità di uso al di fuori della casa stessa (ad esempio nel giardino). Queste limitazioni possono creare delle difficoltà di utilizzo a persone con gravi problemi di mobilità o con problemi di tipo cognitivo.

Le connessioni radio sono attualmente disponibili, anche se a costi maggiori rispetto a quelle con dispositivi all'infrarosso, e consentono di risolvere i problemi elencati in precedenza. Anche in questo caso occorre tenere conto degli aspetti legati alla necessità di sicurezza e di immunità dai disturbi. Un grosso passo avanti può essere fatto con l'utilizzo del nuovo standard DECT[6], che consente di gestire le comunicazioni fra un terminale d'utente ed una stazione base integrando la trasmissione di voce e dati e garantendo un trasmissione sicura ed efficiente. È da segnalare il fatto che esista già un prototipo di un dispositivo, prodotto dal progetto europeo TIDE ASHoRED, che può funzionare sia da telefono senza fili, sia da terminale di controllo.

4.5 L'integrazione con il mondo delle telecomunicazioni

Un aspetto di rilevante importanza è dato dalle possibilità dell'integrazione della "Casa Intelligente" con il mondo delle telecomunicazioni, consentendo la realizzazione dei tele-servizi.

In base al tipo di interconnessione telematica si hanno diversi livelli di prestazioni fornibili all'utente. Le connessioni basate sulla normale rete telefonica (PSTN) presentano il vantaggio di un basso costo e di una capillare diffusione sul territorio, ma i possibili tele-servizi implementabili sono limitati dalla bassa velocità di trasferimento dati. Le connessioni basate sulla nuova rete ISDN presentano numerosi vantaggi, data la maggiore velocità di trasferimento dati ed il maggior numero di servizi di base offerti (es. videotelefonia, connessioni contemporanee su canali multipli, ecc.). In prospettiva la rete ISDN si presenta come la più promettente per la realizzazione di tele-servizi basati sulle nuove tecnologie multimediali.

4.6 Lo scenario tecnologico e di mercato attuale

La situazione attuale relativa alle soluzioni per realizzare delle "Case Intelligenti" è caratterizzata principalmente da due approcci tecnico-commerciali di tipo contrapposto.

4.6.1 L'approccio sistematico "top down"

Questo primo tipo di approccio è di tipo rigoroso e si basa sull'analisi preventiva dei requisiti degli utenti, sulla definizione dell'insieme completo dei componenti di una "Casa Intelligente" e delle sue modalità di interazione con gli utenti. A partire da questa analisi viene definita una struttura di controllo gerarchica in grado di rispondere a tutti i requisiti identificati ed in grado di ospitare tutte le componenti necessarie.

Questo approccio porta alla definizione di precisi standard da sottoporre alle industrie per la produzione di dispositivi compatibili e facilmente integrabili. È quello che è stato adottato da molti dei progetti di ricerca finanziati dalla Comunità Europea, ed ha portato alla definizione di una sofisticata specifica, denominata HS[7] nell'ambito del programma di ricerca ESPRIT. La specifica HS garantisce l'interoperabilità fra tutti i componenti conformi alla specifica, e vi sono vari progetti di ricerca attualmente attivi nello sviluppo di componenti ad essa conformi.

---

[6] *Digital European Cordless Telephone*

[7] *Home Systems*

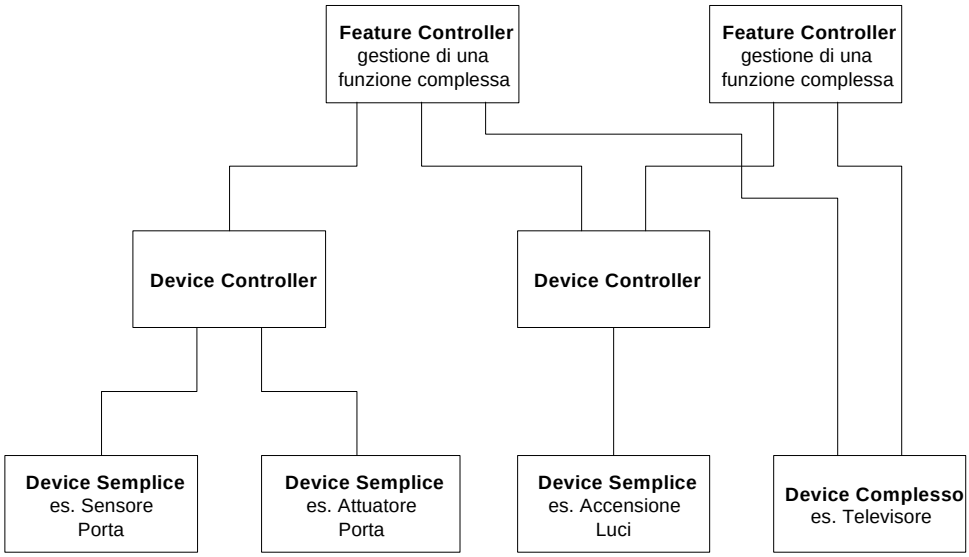


Figura 5: Esempio di Struttura Gerarchica di Comunicazione secondo le specifiche HS

La specifica HS è sponsorizzata da alcuni grandi gruppi industriali fra cui Philips, Marconi, Daimler, Zanussi e Thompson CSF. Il proprietario della specifica è il consorzio EHSA[8], che dovrebbe essere la fonte primaria di informazioni sul sistema e sui componenti. Il sistema per lo sviluppo di componenti aderenti alla specifica viene venduto dalla Philips Consumer Electronics (UK) al prezzo di 11.000 sterline inglesi.

I sistemi di "Casa Intelligente" costruiti in base alle specifiche HS hanno una struttura modulare e con una architettura gerarchica basata su di un sistema di comunicazione condiviso di tipo "bus".

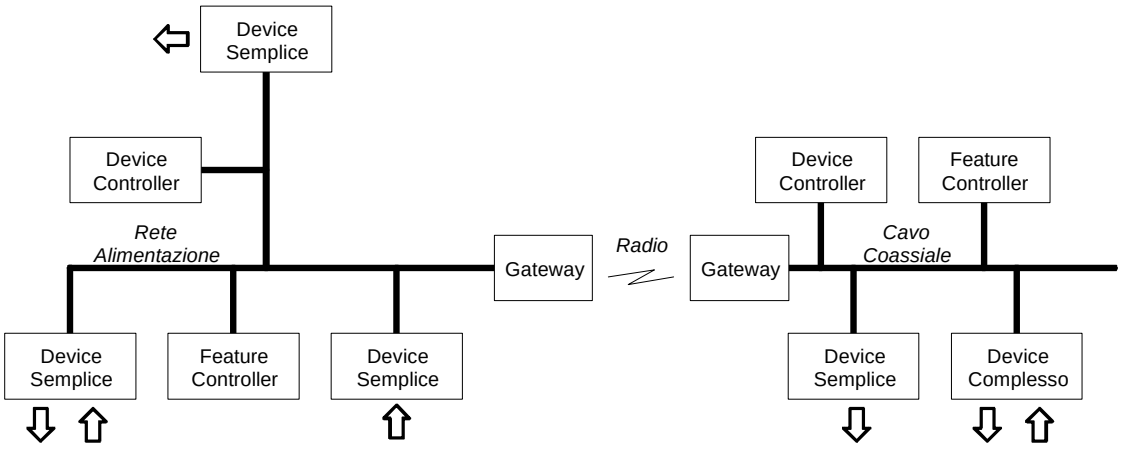


Figura 6: Esempio di Struttura Fisica di Interconnessione secondo le specifiche HS

In pratica risulta attualmente estremamente arduo reperire informazioni a riguardo dei componenti effettivamente disponibili e sugli sviluppi in corso. I sistemi disponibili sono ancora ad uno stadio non commerciale, i componenti non sono facilmente reperibili in Italia ed hanno un costo relativamente elevato.

I principali vantaggi dell'approccio HS sono dati dal fatto che la sua architettura e le sue specifiche sofisticate lo rendono ideale per realizzare dei sistemi completi e modulari, dal

[8] *European Home Systems Association*

supporto da parte di grandi aziende europee e da parte della Comunità Europea (che ne promuoverà la standardizzazione).

Gli svantaggi principali sono dati dal fatto che vi è una generale mancanza di informazioni a livello pubblico, dal fatto che la sofisticazione del sistema può scoraggiare molte piccole e medie imprese a realizzare componenti e soprattutto dal fatto che lo sviluppo del sistema è ancora ad uno stadio prototipale. Realizzazioni commerciali potranno essere disponibili con un ritardo di qualche anno rispetto ad altre soluzioni, che potrebbero affermarsi nel frattempo come standard "de facto".

4.6.2 L'approccio pragmatico "bottom up"

Il secondo approccio parte dalla automazione di dispositivi e apparecchiature domestiche di vario tipo e dalla realizzazione di sistemi di controllo di tipo semplice. Questi sistemi vengono man mano fatti evolvere per rispondere alle richieste di soluzioni nuove e più complesse da parte del mercato. La rispondenza ai requisiti degli utenti dovrebbe in questo caso essere guidata dalle richieste del mercato.

Attualmente vi è una sempre maggiore diffusione di soluzioni commerciali parziali, che consentono un certo livello di "controllo ambientale", orientate però alla soluzione di generiche esigenze di una generica utenza. Sono anche disponibili, prodotti da piccole aziende ed a costi elevati, dei sistemi speciali progettati ed adattati per le esigenze specifiche di alcune categorie particolari di disabili.

Anche sul mercato italiano sono reperibili dispositivi per effettuare l'automazione dell'apertura di porte e finestre, l'accensione delle luci pilotata da sensori di movimento, la regolazione automatica della temperatura ambientale, il controllo di situazioni di emergenza tramite sensori (le situazioni più comuni, e quindi collegate ad un maggior numero di automatismi presenti sul mercato, sono le fughe di gas, la presenza di fumo, gli allagamenti, il controllo anti-intrusione), il controllo remoto di dispositivi tramite telecomandi di tipo "universale"[9] o attuatori ad onde convogliate su rete elettrica, ed altri ancora. Sono anche in funzione dei servizi di tipo generico per il tele-soccorso, la tele-assistenza e la telemedicina. Per gli utenti portatori di particolari disabilità motorie o sensoriali sono invece disponibili delle soluzioni per rendere possibile o più agevole l'utilizzo di dispositivi di uso comune. Sono disponibili sul mercato telefoni modificati con tasti di grandi dimensioni, telefoni a selezione vocale, attuatori ed ausili di vario tipo, sintetizzatori vocali, ecc.

Tutte queste soluzioni presentano però il grave difetto di essere spesso incompatibili fra loro e difficilmente integrabili. Questo perché tutti questi dispositivi vengono progettati dai vari costruttori senza avere uno standard di riferimento comune. Non sono inoltre disponibili delle unità di controllo sufficientemente "aperte", in grado cioè di poter integrare i differenti dispositivi disponibili sul mercato, e che soprattutto presentino quelle caratteristiche di modularità e configurabilità che abbiamo visto essere requisiti essenziali per la realizzazione ottimale di Case Intelligenti per gli utenti anziani.

Iniziano attualmente ad apparire sul mercato dei sistemi completi di automatizzazione domestica, realizzati da alcuni costruttori attorno ad una struttura di connessione di tipo "bus" proprietaria (e quindi non standardizzata). Esempi di sistemi di questo tipo sono dati dal sistema LonWorks della Echelon americana e dal sistema di interconnessione tramite onde convogliate della SGS Thompson. In alcuni casi viene fornita la sola infrastruttura di connessione ed i componenti necessari per sviluppare dei sistemi intelligenti completi. Non viene data invece alcuna garanzia sull'interoperabilità di dispositivi e sistemi sviluppati da

[9] *Questi telecomandi sono dotati di un numero elevato di tasti, ad ognuno dei quali può essere associato un codice ben specifico. Questi telecomandi possono inoltre essere programmati per generare il codice riconosciuto dagli elettrodomestici già dotati di controllo remoto, permettendo così all'utente di utilizzare un solo telecomando per interagire con l'ambiente domestico.*

diversi costruttori. Il sistema di controllo deve quindi essere quasi sempre sviluppato ad hoc sulle specifiche richieste.

Questo secondo approccio, sebbene sicuramente meno "raffinato" dal punto di vista implementativo e non garantisca attualmente il rispetto di tutti i requisiti finora esposti per la realizzazione di "Case Intelligenti" per utenti anziani, potrebbe però risultare alla vincente sul lungo periodo. È infatti possibile che si realizzi un'ampia diffusione di soluzioni proprietarie e per nulla standardizzate, alcune delle quali potrebbero affermarsi su specifici mercati come "standard de facto" difficilmente sostituibili. Il rischio è che questo tipo di sistemi tenga conto solo delle esigenze del pubblico più vasto, sorvolando su alcune delle esigenze che sono specifiche delle persone anziane e disabili in genere.

4.6.3 Esiste quindi una terza via ?

Si tratta di progettare un sistema di controllo per "Casa Intelligente" specifico, che tenga conto non solo dei requisiti esposti in precedenza, ma anche che utilizzi la componentistica ed i dispositivi attualmente disponibili sul mercato. Tale sistema deve essere in grado di adattarsi in modo flessibile alle esigenze di utenti diversi ed al mutare di tali esigenze nel tempo. Deve inoltre risultare facilmente configurabile e "programmabile" dall'utente stesso o da personale di assistenza non specializzato nel settore tecnico informatico. Deve essere altresì in grado di interfacciarsi con dispositivi periferici per l'automazione domestica, il controllo ambientale e la comunicazione con centri servizi di tipi e costruttori differenti. Questo tipo di apertura verso dispositivi diversi può essere realizzata disaccoppiando la logica di controllo dei servizi forniti dalla "Casa Intelligente" dalle modalità fisiche con cui questi sono attivati. Questa struttura di controllo "aperta" e "flessibile" può rispondere in modo ottimale alle mutevoli esigenze degli utenti anziani, adattandosi ad esse e permettendo di integrare delle soluzioni presenti sul mercato e quindi facilmente disponibili ed a costi contenuti. La struttura di un sistema di controllo avente le caratteristiche appena dette è riassunta in Figura 7.

Sulla base di queste considerazioni e dopo un'attenta analisi delle soluzioni esistenti, si sono individuati i requisiti chiave di cui deve essere dotato il sistema, per renderlo *un passo più avanti* rispetto agli altri. Queste sono:

- Possibilità di collegamento con i dispositivi commerciali per l'automazione domestica, il controllo ambientale e la comunicazione con centri servizi, indipendentemente dal tipo e dal costruttore.
- Capacità di adattarsi in modo flessibile alle diverse esigenze degli utenti ed al mutare delle loro necessità nel tempo.
- Disponibilità di un'interfaccia utente configurabile ed intercambiabile, per consentirne anche l'uso da parte di utenti che possono avere dei problemi di interazione a livello sensoriale o cognitivo, oppure per adattare il sistema a diverse esigenze ambientali.
- Integrabilità con applicazioni di servizi di supporto (tele-soccorso, tele-monitoraggio, tele-medicina, ecc.), di utilità (tele-shopping, home-banking, servizi informativi, ecc.) e con programmi esterni.

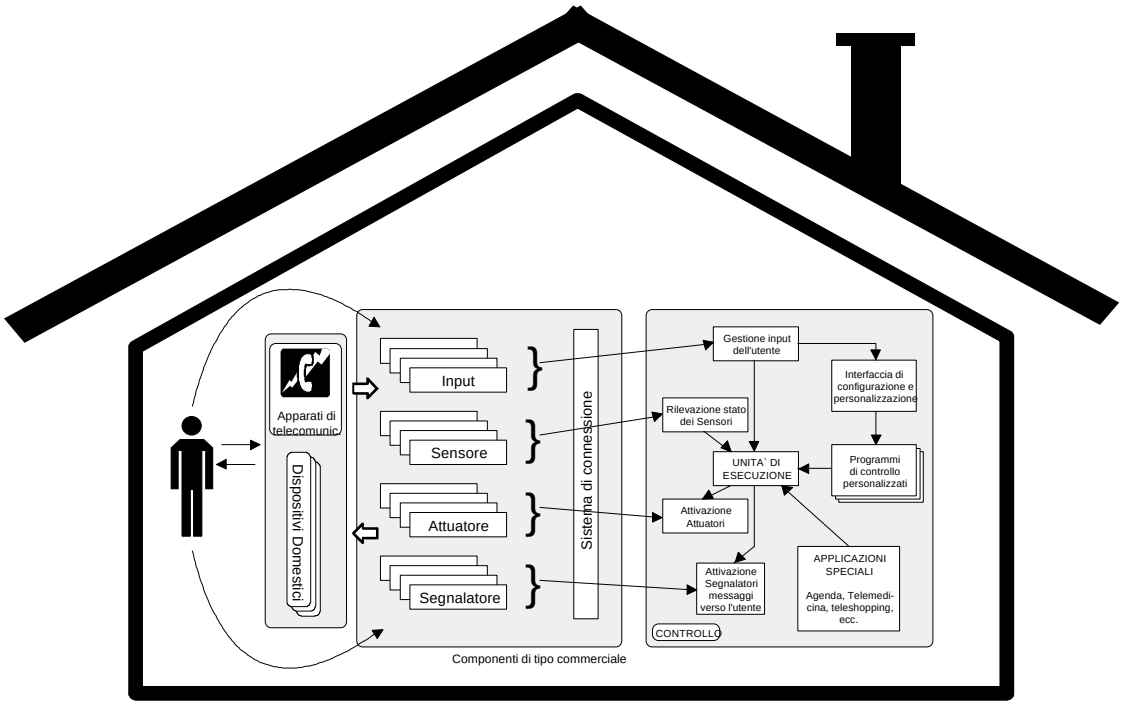


Figura 7: Sistema di Controllo programmabile e aperto per Casa Intelligente (CSELT)

L'approccio utilizzato per affrontare il primo requisito è stato la definizione di un'architettura di sistema nella quale *viene disaccoppiata la logica di controllo dei dispositivi che compongono la Casa Intelligente dalle modalità fisiche con qui questi sono attivati*. Questo disaccoppiamento viene realizzato con la creazione di un'interfaccia logica fra il sistema ed i dispositivi. Tale interfaccia è realizzata tramite dei *drivers* che gestiscono diversi dispositivi e che colloquiano con il Sistema di Controllo tramite delle modalità ben definite. All'apparire sul mercato di nuovi dispositivi, occorrerà scrivere il "driver" apposito per permetterne il colloquio con il sistema, senza dover effettuare lunghe e costose modifiche al sistema stesso.

Per soddisfare il secondo requisito si è deciso di realizzare un sistema *completamente programmabile*, in modo tale da permettere rapide riconfigurazioni e personalizzazioni, realizzabili anche da persone assolutamente non specializzate, quali possono essere l'utente stesso, l'assistente sociale o l'elettricista che installa il sistema. Chiaramente non a tutti gli utenti deve essere permesso "fare di tutto" nel sistema, quindi sono stati previsti diversi *permessi di accesso* e diversi livelli di complessità nell'interfaccia utente. La programmazione viene realizzata, ad alto livello, tramite un'interfaccia utente estremamente semplice basata su finestre. Ai livelli più sofisticati viene invece fornito l'accesso ai comandi di basso livello utilizzati dal Sistema di Controllo.

Il terzo requisito viene risolto dotando il sistema di una libreria di informazioni riguardanti come deve essere realizzata una segnalazione in base alle caratteristiche dell'utente. Il sistema sarà inoltre a conoscenza di che problemi sensoriali o cognitivi ha l'utente. Questo dato sarà modificabile esclusivamente dal personale competente, per evitare possibili (e probabili) problemi. Per ogni segnalazione che il sistema deve inviare all'utente verrà letta dalla libreria la metodologia con la quale è corretto realizzare la segnalazione. Lo stesso procedimento verrà utilizzato per la personalizzazione dell'input da parte dell'utente. Potranno inoltre essere disponibili diversi tipi di interfacce utente, costruite sopra il Sistema di Controllo, che possono colloquiare con l'utente in modi differenti, rendendo quindi disponibili diverse "viste" sull'ambiente controllato dal Sistema di Controllo.

Per realizzare l'integrabilità indicata nel quarto punto, il sistema è dotato di un apposito modulo di comunicazione, che permette ad applicativi esterni di colloquiare con il sistema.

Questi applicativi potranno quindi interagire con l'ambiente domestico tramite la mediazione del Sistema di Controllo, arricchendolo di nuove funzionalità.

5. Architettura del Sistema di Controllo

Verranno ora indicate nel dettaglio le scelte architetturali derivanti dalle considerazioni viste nel capitolo precedente. Queste scelte sono relative ad alcune componenti specifiche del sistema.

5.1 Eventi e Programmi di Controllo

Si è visto precedentemente che il Sistema di Controllo è gestito tramite dei *programmi* realizzabili anche dall'utente stesso. Questi programmi saranno attivati in base ad *eventi*, che possono essere generati dai sensori collegati al sistema o da particolari strutture di controllo, interne al Sistema di Controllo stesso, che simulano l'esistenza di altri sensori.

Dopo essere stati attivati, i programmi potranno acquisire dati dai sensori, attendere il verificarsi di altri eventi, prendere decisioni e comandare degli attuatori. Ad esempio, il pulsante del campanello della porta d'ingresso provocherà l'esecuzione del programma di controllo che deve gestire la segnalazione all'utente che c'è qualcuno alla porta, nel modo appropriato alla condizione psicofisica dell'utente stesso.

Con il termine *evento* si intende il verificarsi di una particolare condizione rilevata dai sensori e segnalata al Sistema di Controllo. Gli eventi potranno essere *semplici* o *complessi*: i primi saranno semplicemente generati dal cambiamento di stato di un singolo sensore, mentre i secondi saranno costituiti dal verificarsi di una combinazione logica di eventi semplici, composta da AND e OR, di diversi sensori.

Gli eventi saranno sempre associati a *livelli di priorità*, in modo da permettere una gestione degli arrivi degli eventi stessi che rispecchi le naturali esigenze di servizio urgente di particolari programmi. Per esempio, il segnale proveniente dal telecomando utilizzato per il teleallarme dovrà essere trattato dal sistema con una priorità più alta di quello proveniente dal termostato della temperatura ambiente. Dato il particolare contesto di utilizzo del programma, si è scelto di utilizzare esclusivamente due livelli di priorità, *normale* e *urgente*. Essi dovranno essere assegnati agli eventi in fase di configurazione. Quando occorrerà scegliere tra più eventi da servire, verranno scelti per primi quelli con priorità urgente.

Gli eventi potranno anche avere dei dati associati o *stati*, che verranno inseriti in un'apposita tabella, dalla quale i comandi di lettura li potranno prelevare. Questi stati saranno ad esempio i valori assunti dai sensori, oppure dei messaggi di errore (per esempio un messaggio di avvenuto timeout).

L'esecuzione dei programmi avverrà in modo *sincrono* e non interrompibile se non su richiesta di un'esplicita istruzione (e quindi non sarà preemptive). I programmi collegati agli eventi sono normalmente brevi e di rapida esecuzione, perciò non monopolizzano il sistema per un tempo eccessivo. Quando un programma di controllo richiede ad un sensore un dato non immediatamente disponibile, il programma si pone in attesa e quindi viene sospeso. Perciò nell'attesa dell'evento collegato si può passare a gestirne un altro. Ovviamente quando l'evento atteso si verifica, il programma di controllo che era stato sospeso verrà ripreso dal punto in cui era stato interrotto, e potrà continuare la sua esecuzione.

Un'altra caratteristica dei programmi di controllo è *l'assenza di variabili*. Sono rese inutili dalla voluta semplicità dei programmi e dalla presenza di istruzioni capaci di controllare dei dati associati agli eventi e di eseguire appropriati comandi. Sono però a disposizione delle istruzioni per la gestione di contatori, utilizzabili per controllare il numero di occorrenze di un particolare evento in un intervallo di tempo.

È possibile inoltre associare lo stesso programma di controllo ad eventi diversi (*programmi rientranti*), in modo da evitare il proliferare di procedure uguali. Per lo stesso motivo si possono richiamare altri programmi dall'interno di un programma di controllo tramite la generazione di *eventi*. Si simulerà cioè l'avvenuto cambiamento di stato di uno o più sensori, provocando di conseguenza un evento e la successiva attivazione del programma ad esso collegato. Naturalmente si possono anche attivare dei programmi tramite un'apposita istruzione, senza dover necessariamente generare un evento. Questa attivazione non è però gestita in modo analogo ad un richiamo di subroutines, bensì come lancio di un nuovo programma che si sovrappone al chiamante. Si è scelto di non fornire un meccanismo di richiamo di subroutines a causa della semplicità dei programmi di controllo. Ciò non toglie che, se venisse reputato necessario, esso non possa essere inserito nel sistema.

5.2 Dispositivi esterni

Al Sistema di Controllo saranno collegati un insieme di *dispositivi*, suddivisi in quattro categorie:

- *Input*: si occupano di ricevere comandi dall'utente.
- *Sensori*: ricevono informazioni dai dispositivi domestici e dalle altre apparecchiature collegate.
- *Attuatori*: comandano i dispositivi domestici e le altre apparecchiature collegate.
- *Segnalatori*: inviano comunicazioni all'utente.

Ogni categoria di dispositivi verrà vista in modo omogeneo dai programmi di controllo, permettendo di avere delle istruzioni generiche per l'interazione con l'ambiente. Verranno cioè messe a disposizione per la programmazione del sistema delle istruzioni uniche per la gestione delle diverse categorie di dispositivi, in modo tale da permettere a chi scrive i programmi di controllo di non doversi preoccupare di come è costituito il particolare dispositivo che si vuole gestire.

5.3 Sensori ed attuatori

Il Sistema di Controllo riceverà informazioni per la gestione dello stato del sistema tramite i *sensori*. Questi possono essere *dispositivi reali*, ovvero componenti collegati fisicamente al sistema, oppure *simulati*, costituiti cioè da elementi facenti parte del sistema stesso. I sensori simulati forniscono degli input aggiuntivi al sistema, utilizzabili per particolari funzioni, quali la gestione dei timeout sulla lettura di dati dai sensori e l'esecuzione di compiti a scadenza periodica, o ad un particolare orario.

Le operazioni di lettura dei dati dai sensori potranno aver associato un intervallo di tempo entro il quale si vuole che il sensore risponda. Questa funzione permette di accorgersi dei malfunzionamenti di alcune componenti del sistema. Per esempio, se viene inviato il comando di chiusura porta e il sensore di porta chiusa non ne comunica l'avvenuta chiusura entro un tempo prestabilito, si può affermare che sussiste un problema sulla porta, ovvero guasti nell'attuatore o nel sensore, oppure un ostacolo che ne impedisce la chiusura.

In molti casi può essere utile fare in modo che il Sistema di Controllo ignori particolari eventi, per esempio per evitare l'esecuzione di comandi che nell'attuale configurazione del sistema sarebbero errati. Questa funzionalità viene realizzata dal *Filtro Eventi*, che si occupa della gestione degli eventi complessi. L'attivazione e la disattivazione di particolari eventi complessi viene anche utilizzata durante l'esecuzione dei programmi di controllo, per impedire che un programma associato ad un particolare evento complesso venga richiamato nuovamente prima che ne sia terminata la sua esecuzione corrente.

A causa della natura stessa dei dispositivi collegati al sistema, esiste la possibilità che qualche sensore o qualche attuatore si blocchi per un guasto durante l'utilizzo del sistema. Per impedire che questo malfunzionamento provochi funzionamenti erronei nell'intero Sistema di Controllo, occorrerà avvertire il sistema di ignorare il dispositivo guasto (ed automaticamente i relativi programmi associati). Una volta ripristinata la funzionalità del componente, si potranno riattivare le parti del sistema precedentemente disabilitate. Questo verrà effettuato senza far ripartire il Sistema di Controllo, grazie alla possibilità di inizializzare "a richiesta" i drivers che gestiscono la comunicazione con i sensori e gli attuatori.

Per come è strutturata la gestione degli eventi, è possibile che variazioni di stato transitorie di brevissima durata passino inosservate. Ciò avviene quando la segnalazione di avvenuto evento viene rimpiazzata da quella di evento terminato senza che il Gestore Driver Sensori abbia tempo di accorgersi del primo cambiamento. La suddetta limitazione non è un inconveniente, bensì permette di diminuire drasticamente la complessità della parte del sistema che si occupa della gestione degli eventi. Inoltre, gli eventi che passeranno inosservati saranno esclusivamente quelli aventi durata minore di *50* millisecondi, tempo più che sufficiente per il controllo di un ambiente domestico. Se questi eventi dovranno per forza essere rilevati, occorrerà agire sul driver del dispositivo o sul dispositivo stesso.

Si è scelto di non dotare il sistema di code, ma di utilizzare al loro posto semplicemente delle *tabelle*. Questa soluzione permette di ottenere una minor occupazione del Sistema di Controllo nella gestione delle sue strutture interne, con un conseguente aumento della velocità di gestione dei programmi di controllo. Le tabelle contenenti i dati delle varie componenti del sistema saranno accessibili da un solo modulo del Sistema di Controllo alla volta, per evitare problemi di non coerenza nei dati.

5.4 Programmazione del Sistema di Controllo

I programmi di controllo utilizzati dal Sistema di Controllo saranno realizzabili tramite due diverse modalità, una più "naturale" ed una più "informatica". Questo consente l'accesso alla personalizzazione del sistema da parte di un'ampia categoria di persone, generalmente non a conoscenza delle problematiche insite nella programmazione vera e propria.

La programmazione del sistema da parte dell'utente avverrà tramite un'*interfaccia a tabelle*, nelle quali dovranno essere inserite le condizioni che provocano l'esecuzione di una certa azione e le azioni da eseguire. Questa parte del sistema andrà accuratamente studiata con l'ausilio di esperti in "fattori umani". Il programma di controllo così realizzato verrà successivamente trasformato nel formato più a basso livello, per renderlo comprensibile al Sistema di Controllo.

Dall'interno del sistema e dai programmi stessi saranno attivabili dei *livelli di diagnostica*, utilizzabili per effettuare la messa a punto dei programmi di controllo e la ricerca di guasti nei sensori e negli attuatori. Saranno inoltre disponibili delle istruzioni per la segnalazione di condizioni di errore, che utilizzeranno i suddetti livelli diagnostici per sapere come effettuare la registrazione e la segnalazione delle anomalie.

5.5 Componenti del Sistema di Controllo

È possibile suddividere il Sistema di Controllo nei seguenti blocchi funzionali:

- Esecutore Programmi
- Filtro Eventi
- Gestore Timer
- Gestore Timeout
- Gestore Driver Sensori

- Gestore Driver Attuatori
- Drivers Sensori
- Drivers Attuatori
- Modulo di comunicazione con servizi di supporto

I suddetti blocchi comunicano tra di loro per poter eseguire una serie di operazioni elementari. Una sequenza di queste operazioni elementari è raggruppata per formare un *programma di controllo*. Questi programmi utente sono attivati da diversi eventi e dalle loro combinazioni.

Nei successivi capitoli verranno descritte in dettaglio le parti componenti il sistema, con particolare attenzione alle operazioni compiute dalle stesse durante l'esecuzione dei programmi di controllo collegati ai vari eventi che si possono verificare all'interno dell'ambiente domestico. Verranno inoltre mostrate le istruzioni disponibili al tecnico per la scrittura di queste procedure e alcuni esempi di risoluzione delle problematiche connesse alla programmazione di una "Casa Intelligente".

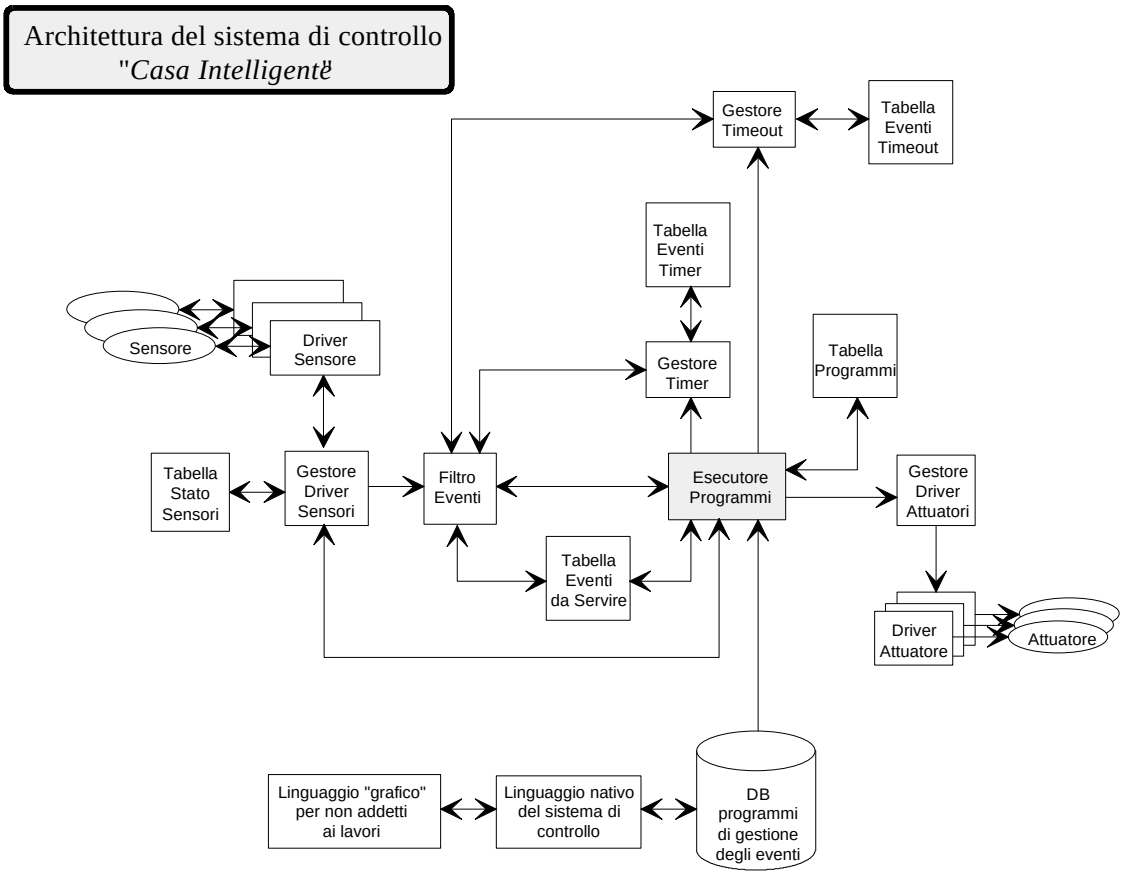


Figura 8: Architettura del Sistema di Controllo

6. Interfacciamento con l'utente

L'utente colloquia con il Sistema di Controllo tramite degli appropriati dispositivi di input/output, quali possono essere il telecomando, il riconoscitore vocale, i segnalatori, ecc. Tutti questi dispositivi verranno associati alle operazioni che dovrà eseguire il Sistema di Controllo tramite la fase di configurazione, permettendo così di ottenere un'interfaccia utente complessa, senza dover intervenire nella struttura del sistema.

Occorre osservare l'estrema importanza che ha la possibilità di avere una completa *programmabilità dell'interfaccia utente*, come anche dei dispositivi usati per comandarla. Sarà possibile quindi interagire con il sistema tramite dispositivi appositamente realizzati per ovviare a carenze sensoriali o cognitive dell'utente. Se le condizioni dell'utente variano, è possibile sostituire questi dispositivi con altri più adatti, semplicemente agendo sulla configurazione del Sistema di Controllo.

Il Sistema di Controllo è un nucleo intorno al quale si può costruire un ambiente telematico anche molto complesso e sofisticato, tramite il quale un utente anziano e/o disabile può essere aiutato nell'espletamento delle attività domestiche quotidiane in modo indipendente. È quindi fondamentale che il Sistema di Controllo possa essere complementato tramite una serie di *programmi esterni*, che ne sfruttino le sue capacità di interazione con l'ambiente per fornire servizi telematici complementari, quali, per esempio, il tele-soccorso, il tele-monitoraggio, applicazioni di tele-medicina, home-banking, tele-shopping, ecc. Tramite questi programmi verranno inoltre realizzate ulteriori funzionalità, quali possono essere la configurazione e la programmazione del sistema, la visualizzazione del suo funzionamento e le funzionalità di debug del sistema e dei programmi. I programmi esterni colloquiano con il sistema tramite delle modalità che verranno viste successivamente.

Questi programmi aggiuntivi potranno utilizzare per il loro colloquio con l'utente gli stessi dispositivi utilizzati come interfaccia utente dal Sistema di Controllo, sfruttandone la struttura logica e svincolandosi perciò dai dispositivi fisici realmente presenti nell'ambiente.

Saranno inoltre intercambiabili, perché sfruttano una comune interfaccia di comunicazione con il Sistema di Controllo per utilizzarne le caratteristiche. Sarà quindi possibile in ogni momento personalizzare il sistema in base alle mutate esigenze dell'utente.

Oltre alle segnalazioni all'utente realizzabili mediante i normali dispositivi di segnalazione inseribili in un ambiente domestico (lampeggiatore, campanello, vibratore, ecc.), può essere necessario fornire le informazioni riguardanti lo stato dell'ambiente circostante in un diverso formato visivo (per esempio, la pianta dell'appartamento con l'indicazione delle porte aperte). Tramite la realizzazione di specifici programmi aggiuntivi sarà possibile fornire all'utente le informazioni a lui necessarie tramite un'appropriata interfaccia visuale, realizzata appositamente per le necessità dell'utente stesso.

Sono descritti nel seguito alcuni programmi esterni al nucleo di controllo che risultano necessari per un suo utilizzo a livello commerciale.

6.1 Programma di configurazione del sistema

Si è visto nei capitolo precedenti che a causa dei problemi di tipo cognitivo o sensoriale, è necessario adattare l'interazione tra il Sistema di Controllo e l'utente affinché essa avvenga nel modo migliore possibile. Questo viene realizzato tramite delle apposite tabelle, che contengono i dispositivi che il sistema deve utilizzare per effettuare il colloquio dell'ambiente con l'utente. Si avrà una tabella contenente i dispositivi da utilizzare per effettuare le segnalazioni all'utente, e un'altra tramite la quale associare delle azioni intraprese dall'utente alle risposte attese dal sistema.

La scelta di questi dispositivi e la loro combinazione per effettuare le segnalazioni complesse è un compito estremamente delicato, che deve quindi essere riservato al personale competente. Se venisse alterata questa configurazione, si rischierebbe di non riuscire più a comunicare con l'utente, provocando perciò gravi inconvenienti e vanificando lo scopo per cui è stata ideata la "Casa Intelligente". Occorre perciò impedire che l'utente possa inavvertitamente o consapevolmente alterare queste scelte.

La configurazione del Sistema di Controllo sarà quindi realizzata tramite un programma esterno, reso disponibile solo al personale tecnico incaricato. Si realizzerà in tal modo una separazione netta tra utente e tecnico, impedendo perciò al primo di alterare il funzionamento corretto del sistema.

In fase di configurazione del sistema occorrerà inoltre associare agli eventi generabili all'interno dell'ambiente degli appositi programmi di controllo, che si occuperanno di implementare i passi necessari alla corretta gestione degli eventi stessi.

6.2 Programmazione del sistema

Il comportamento del Sistema di Controllo è regolato dall'esecuzione di programmi appositamente realizzati in base agli eventi che possono essere generati dai dispositivi presenti nell'ambiente domestico e da quelli usati dall'utente per interagire con il sistema.

Per la realizzazione di tali programmi viene reso disponibile al personale tecnico un semplice *linguaggio di programmazione.* Questo linguaggio è fortemente orientato alla risoluzione dei problemi insiti nella preparazione delle procedure di controllo di una "Casa Intelligente". Ciò permette al tecnico di realizzare in modo rapido e semplice un insieme di procedure collegate alle caratteristiche dell'ambiente e alle problematiche dell'utente. Queste procedure potranno quindi essere di tipo generico, quali quelle di accensione delle luci o di chiusura delle porte, o mirate alla compensazione delle carenze dell'utente e all'aumento della sua sicurezza, quali quelle di tele-soccorso o di automatizzazione di compiti manuali.

Anche per quel che riguarda la gestione dei programmi di controllo vi è lo stesso problema di sicurezza visto poc'anzi per quel che riguarda la configurazione del Sistema di Controllo. Quindi anche la programmazione del sistema sarà realizzata tramite un programma esterno, disponibile solo al personale tecnico incaricato.

Occorre però poter permettere ad un utente in grado di farlo, di personalizzare ulteriormente l'ambiente nel quale vive. Bisogna quindi fornire all'utente uno strumento software più vicino al suo modo di ragionare, rispetto al modo di pensare di un tecnico abituato ad interagire con un linguaggio di programmazione. Questo strumento dovrà inoltre essere dotato di tutti gli accorgimenti necessari ad impedire che venga alterata la corretta modalità di gestione dell'ambiente.

Questo *generatore di programmi* dovrà quindi essere capace di interpretare una serie di comandi dati dall'utente e di conseguenza generare un programma di controllo. Un tale tipo di programmazione non consentirà di generare dei programmi particolarmente sofisticati, ma permette comunque all'utente di ottenere delle procedure sufficienti alle sue necessità. Questo generatore di programmi può essere realizzato tramite un'interfaccia a tabelle, tramite le quali l'utente sceglie che operazioni vuole associare ad un determinato evento complesso rilevato dal Sistema di Controllo.

Un aspetto importante del quale occorre tenere conto durante lo studio del Sistema di Controllo per una "Casa Intelligente" è il problema dell'individuazione dei guasti e di come vi si può porre rimedio provocando il minor disagio possibile nell'utente.

Durante il funzionamento quotidiano del Sistema di Controllo, è certamente prevedibile che possano accadere guasti o malfunzionamenti ai dispositivi collegati al sistema. Sarà compito del sistema e soprattutto dei programmi di controllo controllare se questo accade.

Se un guasto o un malfunzionamento viene rilevato, verrà inserita un'annotazione in un apposito *file di log*, consultabile esclusivamente dal personale tecnico mediante un apposito programma esterno, tramite il quale si potrà risalire alle cause del malfunzionamento. In base al tipo di guasto, e quindi alla sua gravità, occorrerà decidere se effettuare una segnalazione immediata agli addetti alla manutenzione, oppure limitarsi ad escludere i componenti non funzionanti del sistema.

6.3 Visualizzazione dello stato del sistema

Come si è visto all'inizio di questo capitolo, il Sistema di Controllo comunica con l'utente tramite i dispositivi di segnalazione, quindi senza generare output verso l'eventuale monitor collegato al calcolatore. Durante la fase di configurazione e programmazione può essere utile avere a disposizione una visualizzazione istantanea su video dello stato interno del sistema, per permettere al tecnico di verificare il corretto funzionamento dei programmi di controllo da lui scritti.

Tramite un apposito programma esterno è possibile realizzare questa funzionalità, permettendo così la creazione di un completo sistema di sviluppo per la creazione e la verifica funzionale dei programmi di controllo. Sarà inoltre possibile generare artificialmente gli eventi, permettendo così di verificare immediatamente se la risposta del Sistema di Controllo è corretta.

La visualizzazione tramite il monitor del calcolatore dello stato dell'ambiente può inoltre essere molto utile all'utente, permettendogli di vedere immediatamente quale è la condizione corrente dei dispositivi presenti, per esempio vedere se lo scaldabagno è acceso senza doversi alzare o per controllare che tutte le finestre siano chiuse se si mette a piovere. Il monitor diventerà quindi per l'utente un'estensione dei suoi sensi, permettendogli di controllare completamente lo stato dell'ambiente in cui vive.

6.4 Esempi di ulteriori programmi esterni opzionali

Oltre ai programmi esterni già visti in precedenza, quali il tele-soccorso, tele-monitoraggio, tele-shopping, home-banking, ecc., è possibile interfacciare con il Sistema di Controllo ulteriori programmi. Tutti questi pacchetti software aggiuntivi devono naturalmente essere realizzati con lo scopo di aiutare l'utente nella sua routine quotidiana.

Vediamo ora alcuni esempi di funzioni aggiuntive che possono essere fornite all'utente tramite questi programmi aggiuntivi.

- *Agenda domestica*: aiuta l'utente a ricordarsi gli appuntamenti della giornata, quali possono essere la visita medica, il parrucchiere, ecc.
- *Contabilità domestica*: tramite questa funzione l'utente può facilmente controllare i movimenti finanziari, quali possono essere la pensione, le bollette da pagare, ecc.
- *Promemoria somministrazione medicinali e terapie*: permette di assicurare il corretto progredire delle terapie mediche dell'utente, ricordando quando, come e quali medicinali deve prendere ed eventualmente che esercizi fisici deve fare.
- *Guida culinaria* assiste l'utente nella preparazione dei pasti.
- *Scorta alimentare*: tiene traccia del consumo dei prodotti alimentari e avverte l'utente o il personale addetto se il quantitativo presente nella dispensa scende sotto una soglia prefissata.

7. Esecutore Programmi

L'Esecutore Programmi è il nucleo del Sistema di Controllo. Si occupa di eseguire i singoli programmi di controllo degli eventi, trasferendo il controllo da uno all'altro in base agli eventi stessi e alle attese sulla lettura dello stato dei sensori.

Un altro compito dell'Esecutore Programmi è quello della gestione della personalizzazione delle segnalazioni utente da parte del sistema, in base alle condizioni psicofisiche dello stesso.

I dati necessari al funzionamento dell'Esecutore Programmi sono contenuti nelle tabelle Programmi, Eventi da Servire e Segnalazioni Utente.

7.1 Tabella Programmi

Nella tabella *Programmi* l'Esecutore Programmi ha memorizzate le informazioni riguardanti i programmi di controllo del sistema. Oltre all'associazione con gli eventi complessi responsabili dell'attivazione iniziale dei programmi di controllo, in questa tabella verranno inseriti anche gli eventi complessi sui quali i programmi sono in attesa, e perciò responsabili della loro riattivazione. I programmi di controllo che dovranno essere attivati ad un orario prestabilito saranno identificati dagli ultimi due campi della tabella. Essa avrà la struttura seguente:

Tabella 7.1: Programmi

Evento Attivante	Identificatore Programma	Istruzione Iniziale	PID	OFlag	Orario

- *Evento Attivante*: il nome dell'evento che realizza l'associazione tra la variazione di una caratteristica ambientale ed il relativo programma di controllo. Si tratta di un evento complesso.
- *Identificatore Programma* nome del programma di controllo.
- *Istruzione Iniziale*: numero della riga di codice dalla quale deve avvenire l'esecuzione del programma di controllo. Sarà pari ad 1 se il programma deve iniziare dall'inizio, altrimenti avrà un altro valore se il programma è stato sospeso e deve continuare la sua esecuzione da una certa posizione in poi.
- *PID*: identificatore del programma di controllo, generato automaticamente ed in modo univoco dal sistema all'atto dell'avvio del programma. Viene utilizzato per identificare univocamente la copia del programma in fase di esecuzione, dato che posso avere più copie dello stesso programma associate a diversi eventi.
- *OFlag*: indica che il programma deve essere eseguito all'orario specificato dal campo seguente.
- *Orario*: specifica l'orario in cui deve essere attivato questo programma di controllo. L'orario è nel formato *hhmm*, con $0 \leq hh \leq 23$ e $0 \leq mm \leq 59$.

7.2 Tabella Eventi da Servire

La tabella *Eventi da Servire* contiene l'elenco degli eventi attualmente verificati, dei quali occorre quindi eseguire i programmi di controllo associati. La struttura della tabella è la seguente:

Tabella 7.2: Eventi da Servire

Nome Evento	Priorità

- *Nome Evento*: il nome dell'evento complesso da servire.
- *Priorità*: la priorità caratteristica dell'evento (normale o urgente).

7.3 Tabella Segnalazioni Utente

La tabella *Segnalazioni Utente* viene utilizzata dall'Esecutore Programmi nell'esecuzione di segnalazioni all'utente, per sapere che tipo di segnalatori utilizzare per informare l'utente nel modo più corretto possibile. I dati presenti in questa tabella verranno caricati in fase di configurazione e programmazione del sistema da parte del personale addetto, e non dovranno essere accessibili all'utente, per evitare malfunzionamenti e segnalazioni ambigue. La tabella è strutturata nel modo seguente:

Tabella 7.3: Segnalazioni Utente

	Dispositivo 1	Dispositivo 2	. . .	Dispositivo m
Segnalazione 1				
Segnalazione 2				
. . .				
Segnalazione n				

- *Segnalazione*: tipo di segnalazione da effettuare (per esempio campanello porta, telefono, allarme porta aperta, ecc.)
- *Dispositivo*: elenco dei dispositivi di segnalazione presenti nel sistema.

Se deve essere effettuata una certa segnalazione, viene analizzata la linea corrispondente e vengono attivati i dispositivi che hanno un valore su questa riga con la modalità indicata sulla riga stessa. Per esempio, è possibile avere un diverso numero di squilli del campanello o di lampeggi del lampeggiatore in base al tipo di segnalazione da effettuare all'utente.

7.4 Tabella Risposte Utente

La tabella *Risposte Utente* viene utilizzata dall'Esecutore Programmi quando vi è la necessità, all'interno di un programma di controllo, di attendere una risposta proveniente dall'utente. Tramite questa tabella, l'Esecutore Programmi realizza l'associazione tra il tipo di risposta da attendere dall'utente e i sensori utilizzabili per fornirla. I dati presenti in questa tabella verranno caricati in fase di configurazione e programmazione del sistema da parte del personale addetto, e non dovranno essere accessibili all'utente, per evitare malfunzionamenti e comportamenti ambigui. La tabella è strutturata nel modo seguente:

Tabella 7.4: Risposte Utente

	Dispositivo 1	Dispositivo 2	. . .	Dispositivo m
Risposta 1				
Risposta 2				
. . .				
Risposta n				

- *Risposta*: tipo di risposta proveniente dall'utente da attendere (per esempio l'annullamento del teleallarme, la conferma di un comando, ecc.)
- *Dispositivo*: elenco dei dispositivi di input presenti nel sistema.

7.5 Funzionamento dell'Esecutore Programmi

Quando nella tabella Eventi da servire è presente un evento, l'Esecutore Programmi lo preleva (utilizzando una politica ben definita per le priorità), lo elimina ed pone a zero il flag "Evento Attualmente in Coda" nella tabella Eventi Complessi. Occorrerà anche porre ad uno il flag "Mask" per impedire che vengano attivate più copie dello stesso programma di controllo. Questo flag andrà azzerato solo quando il programma di controllo raggiungerà il termine.

Dopo di che ricerca nella tabella Programmi l'identificativo del programma di controllo associato all'evento prelevato dalla tabella Eventi Da Servire e lo avvia a partire dall'istruzione specificata nel campo "Istruzione Iniziale". Se questo programma di controllo è uno di quelli che era stato sospeso perché in attesa di un valore da un sensore, l'istruzione da cui farlo ripartire sarà diversa dalla prima. All'atto dell'inserimento di un nuovo evento nella tabella programmi, verrà associato alla copia del programma di controllo ad esso associata il campo "PID", per identificarla univocamente.

Se invece si ha a che fare con un programma di controllo lanciato dall'interno di un altro programma, non si dovrà disabilitare nessun evento all'interno della tabella Eventi Complessi.

Le istruzioni contenute nel programma di controllo saranno analizzate ed eseguite sequenzialmente, una per volta, finché non si troverà un'istruzione di lettura dello stato di un sensore (o dell'attesa di una risposta dall'utente, che verrà trattata in modo analogo). Se questa lettura richiede un'attesa perché il valore non è già disponibile al Gestore Driver Sensori, il programma di controllo verrà sospeso e verrà aggiornata la tabella Eventi Complessi. Questo aggiornamento consisterà nell'inserire una linea nella tabella contenente l'evento complesso che deve risvegliare il programma di controllo, il flag "Wait" indicante che questo evento deve solo riattivare un programma sospeso (quindi questa riga dovrà essere rimossa appena l'evento si verifica e viene attivato) ed un livello di priorità Normale. Sospeso il programma di controllo, l'Esecutore Programmi preleva dalla tabella Eventi Da Servire un nuovo evento.

Quando il programma di controllo arriva al suo termine naturale, il flag "Mask" dell'evento complesso responsabile della sua attivazione viene azzerato, in modo da permetterne l'ulteriore esecuzione. Questo evento complesso viene identificato univocamente tramite il campo "PID" associato alla copia del programma in fase di esecuzione.

Lo stato restituito dalle operazioni di lettura dei valori dai sensori verrà mantenuto in un buffer interno all'Esecutore Programmi e verrà utilizzato per le operazioni di confronto.

I passi elementari che l'Esecutore Programmi deve eseguire in base alle istruzioni presenti nei programmi sono indicate in dettaglio insieme alla descrizione delle stesse.

8. Filtro Eventi

Il Filtro Eventi si occupa del rilevamento degli eventi complessi che si verificano e del loro inserimento nella tabella Eventi da Servire. Permette inoltre di effettuare una selezione sugli eventi complessi che occorre tenere sotto controllo, sensibilizzando il sistema ad attivarsi solo su alcuni di essi. Gli eventi disabilitati non verranno rilevati e quindi il loro occorrere verrà ignorato.

8.1 Tabella Eventi Complessi

I dati necessari al funzionamento del Filtro Eventi sono contenuti nella tabella *Eventi Complessi*, che ha la seguente struttura:

Tabella 8.1: Eventi Complessi

Evento Complesso	Flag Mask	Flag EAC	Flag Wait	Flag Timeout	Flag Timer	Priorità

- *Evento Complesso*: espressione logica tra più eventi semplici (per es. "A and B or C").
- *Flag Mask*: se vale 1 indica che l'evento complesso deve essere ignorato.
- *Flag EAC*: se vale 1 indica che l'evento complesso si è verificato ed è stato posto nella tabella Eventi da Servire, ma non è ancora stato servito dall'Esecutore Programmi.
- *Flag Wait*: se vale 1 indica che l'evento complesso ha come scopo quello di riattivare un programma di controllo sospeso, quindi questa riga dovrà essere eliminata appena il suddetto evento si verifica.
- *Flag Timeout*: se vale 1 indica che l'evento complesso è collegato ad un timeout.
- *Flag Timer*: se vale 1 indica che l'evento complesso è collegato ad un timer, con la funzione di interromperlo.
- *Priorità*: Normale o Urgente (rispettivamente i valori 0 e 1).

8.2 Funzionamento del Filtro Eventi

Il Filtro Eventi si attiva quando avviene una modificazione nello stato dei sensori, veri o simulati che siano, ovvero quando si verifica un evento semplice. Questa situazione viene segnalata dal Gestore Driver Sensori, dal Gestore Timeout e dal Gestore Timer.

Quando viene attivato, il Filtro Eventi inizia a scorrere la tabella Eventi Complessi per vedere se uno o più eventi complessi si sono verificati. Si inizia ad analizzare quelli aventi priorità Urgente e si continua con quelli aventi priorità Normale. Gli eventi complessi vengono analizzati solo se almeno uno degli eventi semplici che li compongono è variato rispetto alla scansione precedente della tabella Eventi Complessi. Tutti quelli che vengono trovati attivi sono posti, con la loro priorità, nella tabella Eventi Da Servire per essere serviti dall'Esecutore Programmi, nell'ordine con il quale sono stati trovati nella tabella Eventi Complessi. A questi eventi viene posto a 1 il flag "EAC".

Agli eventi complessi che vengono attivati verrà posto a 1 il flag "Mask", per impedire che i programmi di controllo ad essi collegati vengano attivati più volte sullo stesso evento complesso. Naturalmente si potranno avere più copie dello stesso programma di controllo, ma collegate ad eventi complessi differenti.

Terminata la scansione della tabella Eventi Complessi verrà azzerata la colonna della tabella Stato Sensori contenente il flag "Variazione".

Tra gli eventi complessi che verranno attivati ci potranno essere anche quelli che si occupano di riattivare i programmi momentaneamente sospesi perché in attesa di dati dai sensori. Questi eventi complessi sono individuati tramite il flag "Wait" posto a 1. La riga che li riguarda andrà rimossa dalla tabella Eventi Complessi, perché devono essere attivati una ed una sola volta, ed inoltre non devono poter essere rimossi dalla tabella Eventi da Servire se l'evento complesso che li ha attivati viene a cessare prima che siano stati riattivati i programmi ad essi collegati.

Un altro tipo di eventi complessi presenti nella tabella sono quelli generati dal *Gestore Timer* e dal *Gestore Timeout*.

Gli eventi generati dal Gestore Timer si verificheranno quando un timer scade e dovranno essere inseriti nella tabella Eventi da Servire con l'indicazione che l'evento deve riattivare un programma di controllo sospeso, quindi avente il flag "Wait" posto a 1, e con una priorità da stabilire. Nella tabella Eventi Complessi il Gestore Timer inserirà degli eventi complessi capaci di interrompere un timer attualmente in esecuzione, prima che questo raggiunga la sua scadenza naturale. Essi avranno il flag "Timer" posto a 1 per indicare che sono utilizzati per annullare un timer e quindi il Filtro Eventi, quando uno di questi eventi si verifica, dovrà segnalare questo fatto al Gestore Timer.

Il Gestore Timeout inserirà nella tabella Eventi Complessi un evento complesso con il flag "Timeout" posto a 1 indicante che l'evento è collegato ad un timeout. Se l'evento si verifica, esso verrà trattato allo stesso modo di un evento normale, con l'unica differenza che verrà effettuata una segnalazione al Gestore Timeout per far si che elimini il timer di timeout dalla sua tabella. Se invece il timeout scade, il Gestore Timeout eliminerà dalla tabella Eventi Complessi l'evento complesso che avrebbe dovuto sospenderlo e inserisce nella tabella Eventi da Servire l'evento complesso che deve riattivare il programma di controllo, ma inserendo come valore associato al sensore su cui si era in attesa, nella tabella Stato Sensori, la segnalazione di avvenuto timeout.

Il controllo per l'attivazione dei programmi che devono essere eseguiti in un particolare istante della giornata viene anch'esso realizzato scandendo ad intervalli regolari la tabella Programmi, per verificare se occorre attivarne qualcuno. Se ciò si verifica, viene posto nella tabella Eventi da Servire l'evento che identifica univocamente il programma all'interno della tabella Programmi. Questi eventi univoci vengono generati all'atto del caricamento in memoria di questi particolari programmi di controllo. Saranno riconoscibili perché composti dall'identificativo di un sensore che si è deciso non poter esistere all'interno del Sistema di Controllo, cioè il sensore numero 0.

Verrà effettuata anche una scansione per verificare se nella tabella Eventi Complessi si hanno degli eventi complessi la cui condizione è ora non più verificata ma che devono ancora essere serviti, cioè sono ancora in attesa nella tabella Eventi Da Servire. Questi andranno rimossi dalla tabella Eventi Da Servire ed ad essi andrà anche azzerato il flag "EAC".

L'Esecutore Programmi può richiedere l'abilitazione o la disabilitazione di particolari eventi. Per fare ciò viene ricercata nella tabella Eventi Complessi la riga corrispondente all'evento complesso richiesto, e ne viene modificato il flag "Mask".

Vediamo alcune tipologie di eventi con i possibili valori associati come stati:

Tabella 8.2: Eventi e Stati

Sensore	**Evento Semplice**	**Stato Sensore**
Interruttore	Acceso / Spento	Nessuno
Pulsante	Premuto / Non premuto	Nessuno
Termostato	Superamento / Non superamento della soglia impostata	Nessuno
Termocoppia	Misurazione effettuata	Temperatura misurata

Gli eventi semplici potranno essere o indicazioni di una variazione avvenuta nello stato dei sensori oppure la segnalazione dello stato di un sensore avente come stati associati solo "ON" e "OFF". I valori misurati dai sensori del primo tipo verranno inseriti nella tabella Stato Sensori, e da essa prelevati dalle apposite istruzioni.

9. Gestore Timer

Nel caso in cui si abbia necessità di inserire delle pause nel programma di controllo, oppure operazioni da eseguire a intervalli regolari o ad un orario prestabilito, si può ricorrere al Gestore Timer. Questo è una parte del Sistema di Controllo che simula il verificarsi di eventi complessi in base a scadenze temporali prestabilite, si potranno quindi associare ad essi degli appropriati programmi di controllo.

9.1 Tabella Timer

Le informazioni necessarie al Gestore Timer sono contenute nella tabella*Timer*, così strutturata:

Tabella 9.1: Timer

Scadenza Timer	Timer Singolo / Ciclico	Evento Riattivante

- *Scadenza Timer*: indicazione temporale per l'attivazione del timer. Può essere assoluta (es. 15:30) o relativa (es. 10 min.).
- *Timer Singolo / Ciclico*: flag indicante se il timer deve essere eseguito una tantum (0) o ciclicamente finché non viene interrotto (1).
- *Evento Riattivante*: nome dell'evento complesso da generare per riattivare il programma di controllo. Questo nome viene generato univocamente dal Gestore Timer e associato al programma di controllo che richiede l'uso del timer.

9.2 Funzionamento del Gestore Timer

Se viene richiesta la creazione di un timer singolo, l'Esecutore Programmi invierà al Gestore Timer la scadenza temporale del timer alla quale riattivare il programma di controllo e il Gestore Timer restituirà l'evento da generare per segnalare al Filtro Eventi che l'intervallo è trascorso. Il Gestore Timer potrà inoltre ricevere un evento complesso che permetta eventualmente di interrompere il timer e riprendere la normale esecuzione del programma di controllo. Occorre anche specificare al Gestore Timer che si tratta di un timer singolo, cioè se deve essere attivato una volta soltanto, ponendo a 0 il flag "Timer Singolo / Ciclico".

L'Esecutore Programmi provvederà ad inserire i dati riguardanti il programma di controllo da risvegliare nella tabella Programmi. Questi dati consisteranno nell'identificativo del programma di controllo corrente, della posizione dalla quale farlo ripartire, cioè l'istruzione successiva a quella che ha richiesto l'uso del timer, e l'evento capace di riattivare il programma (è il nome evento restituito dal Gestore Timer). Questo eventuale evento complesso utilizzabile per interrompere il timer verrà anche inviato al Filtro Eventi che lo inserirà nella tabella Eventi Complessi, con l'indicazione che è un evento utilizzato per interrompere un timer non ciclico, cioè avente il flag "Timer" uguale a 1.

Nel caso in cui si tratti di un timer ciclico, verrà inserito nella tabella Timer il nome dell'evento complesso da generare per attivare il programma di controllo che si vuol fare eseguire ad intervalli prestabiliti, la scadenza temporale ed il valore 1 nel flag "Timer Singolo / Ciclico".

Ciclicamente verrà scandita la tabella Timer. Se un timer raggiunge la scadenza senza essere interrotto, il Gestore Timer deve verificare se è un timer singolo o se è ciclico, ed eseguire le appropriate operazioni.

Se si tratta di un timer singolo, andranno rimossi i dati del timer dalla tabella Timer e i dati dell'evento capace di interrompere il timer dalla tabella Eventi Complessi. Dopo di che verrà inserito nella tabella Eventi Da Servire il codice dell'evento complesso collegato alla riattivazione del programma di controllo ed una appropriata priorità. Se invece è un timer ciclico, verrà inserito nella tabella Eventi Da Servire l'evento complesso necessario ad attivare il programma di controllo specificato.

Se un timer non ciclico viene interrotto, cioè si verifica l'evento complesso collegato all'annullamento del timer, il Gestore Timer riceverà la segnalazione dal Filtro Eventi. Occorrerà inserire nella tabella Eventi da Servire l'evento riattivante (con priorità Normale), per risvegliare il programma di controllo che era in attesa sul timer, e dovrà eliminare dalla tabella Timer la riga corrispondente al timer annullato. Dovrà anche essere eliminata dalla tabella Eventi Complessi la riga corrispondente all'evento che ha annullato il timer.

I timer ciclici potranno venire annullati in qualsiasi momento da un qualsiasi programma di controllo, anche diverso da quelli che li ha attivati, utilizzando un'apposita istruzione. Questa capacità dei timer ciclici di essere annullati è utile per cambiare la risposta del sistema agli eventi se vengono a mutare le condizioni che avevano reso necessari i succitati timer.

Quando un timer ciclico viene annullato, il Gestore Timer dovrà ricercare nella tabella Programmi l'evento complesso corrispondente al programma di controllo del quale si vuole eliminare il richiamo ciclico, cioè l'evento attivante, ed eliminare la riga corrispondente. Dovrà anche venire eliminata dalla tabella Timer la riga corrispondente al timer ciclico stesso.

10. Gestore Timeout

La capacità di ricevere informazioni sullo stato dell'ambiente è fondamentale per il funzionamento del Sistema di Controllo. Le operazioni di acquisizione dati dai sensori possono essere collegate ad un tempo massimo entro il quale ottenere una risposta, cioè un timeout. La gestione di questi tempi di attesa è affidata al Gestore Timeout. Quando un programma di controllo vorrà attendere il verificarsi di un evento su di un sensore, dovrà perciò stabilire anche entro quanto tempo questo si dovrà verificare, scaduto il quale il programma dovrà riattivarsi e accorgersi di questa condizione anomala.

L'Esecutore Programmi passerà al Gestore Timeout il nome dell'evento complesso tramite il quale si è in attesa di un valore e il tempo massimo entro il quale si vuole ottenere questa risposta. Questo evento complesso sarà a tutti gli effetti composto solo dal sensore dal quale si è in attesa di una risposta.

10.1 Tabella Timeout

Le suddette informazioni verranno inserite nella tabella *Timeout*, così strutturata:

Tabella 10.1: Timeout

Evento sul quale si è in attesa	Tempo massimo di attesa

- *Evento sul quale si è in attesa*: nome dell'evento complesso, presente nella tabella Eventi Complessi, al quale si vuole associare un tempo massimo di attesa della risposta.
- *Tempo massimo di attesa*: intervallo di tempo (espresso in secondi) entro il quale deve giungere l'attivazione dell'evento complesso, ovvero la risposta dai sensori.

10.2 Funzionamento del Gestore Timeout

Al Filtro Eventi verrà inviato l'evento complesso da attendere e l'indicazione che quell'evento è collegato ad un timeout, cioè verrà posto a 1 il flag "Timeout". Nella tabella Programmi sarà inserita una riga contenente l'evento complesso da attendere, l'identificativo del programma di controllo e l'istruzione da cui riattivare l'esecuzione, perciò quella seguente rispetto a quella nella quale era presente la richiesta dell'attesa.

La tabella verrà scandita ciclicamente, e se uno di questi timeout si verifica, occorrerà innanzitutto eliminare dalla tabella Eventi Complessi l'evento complesso al quale è associato il timeout. Bisognerà inoltre inserire nella tabella Eventi Da Servire l'evento complesso collegato al timeout e l'indicazione di priorità Urgente. Nella tabella Stato Sensori si inserirà nel campo "Valore" del sensore interessato dall'attesa un codice di timeout, per indicare che non sono stati ricevuti in tempo i dati dal sensore.

Se invece l'evento si verifica, il Filtro Eventi dovrà segnalare al Gestore Timeout di eliminare dalla tabella Timeout le informazioni relative all'evento occorso, e inserire nella tabella Eventi Da Servire il codice dell'evento.

Un'operazione di attesa con timeout può anche essere annullata da un programma di controllo diverso da quello che l'ha richiesta, mediante un'apposita istruzione.

In questo caso il Gestore Timeout dovrà eliminare i dati riguardanti l'attesa dalla tabella Timeout. Anche il Filtro Eventi dovrà eliminare dalla tabella Eventi Complessi la riga

corrispondente all'evento sul quale vi era l'attesa. Dovrà inoltre porre a 0 il flag "Mask" dell'evento complesso responsabile dell'attivazione del programma di controllo che era in attesa, per permettere che venga richiamato nuovamente. Nella tabella Programmi, l'Esecutore Programmi porrà ad 1 il campo "Istruzione Iniziale" della riga corrispondente al programma di controllo che era in attesa, per permettere al programma stesso di essere riavviato correttamente al richiamo successivo.

Il programma di controllo che era in attesa della risposta da un sensore verrà quindi abortito. Verrà quindi posto a 0 il flag "Mask" dell'evento complesso responsabile dell'attivazione del suddetto programma di controllo.

11. Gestore Driver sensori e attuatori

Tra i sensori e il Sistema di Controllo esisteranno dei driver legati alla tipologia dei sensori e degli attuatori, che si occuperanno del pilotaggio diretto degli stessi, e dei Gestori Driver, che si occuperanno di passare i comandi e i dati da e per i driver al resto del Sistema di Controllo.

Sono presenti nel sistema due Gestori Driver, il *Gestore Driver Sensori* e *il Gestore Driver Attuatori*. Il primo si occupa di rilevare i dati provenienti dai driver dei sensori e di inviare ad essi le informazioni necessarie alla loro configurazione. Il secondo ha il compito di inviare ai driver degli attuatori i comandi necessari a variarne lo stato.

11.1 Tabella Stato Sensori

La tabella *Stato Sensori* avrà una riga per ogni sensore, e sarà strutturata nel modo seguente:

Tabella 11.1: Stato Sensori

Nome Sensore	Stato (Valore assunto)	Flag Variazione	Contatore

- *Nome Sensore*: identificatore univoco di un sensore del sistema.
- *Stato*: ultimo valore conosciuto assunto dal sensore.
- *Variazione*: flag indicante se lo stato di questo sensore è variato dall'ultima scansione della tabella da parte del Filtro Eventi.
- *Contatore*: numero di volte che questo sensore ha cambiato di stato dall'ultima volta che è stato azzerato, tramite l'apposito comando.

I comandi di lettura dello stato dei sensori faranno direttamente riferimento ai valori contenuti in questa tabella, mentre i comandi che vogliono acquisire lo stato aggiornato dei sensori richiederanno al Gestore Driver Sensori il prelievo dei valori dai sensori stessi e, quando verrà loro notificato che sono disponibili, li preleveranno dalla tabella.

11.2 Funzionamento del Gestore Driver

Il Gestore Driver Sensori riceverà i comandi dall'Esecutore Programmi. Questi comandi potranno essere di acquisizione dello stato di un sensore o di modifica delle sue caratteristiche operative. I dati provenienti da questi sensori verranno sempre posti in un'apposita tabella, la tabella Stato Sensori, e la segnalazione che vi è stata una variazione dello stato del sistema viene inviata al Filtro Eventi.

I dati richiesti dall'Esecutore Programmi saranno prelevati direttamente dalla tabella, se sono già disponibili, o richiesti al driver del sensore e poi posti anch'essi nella tabella, pronti per essere prelevati, se non sono ancora disponibili o se si desidera ottenerne una versione aggiornata. I comandi per variare la configurazione dei sensori verranno semplicemente passati ai driver relativi, che si occuperanno di gestirli in modo opportuno.

Quando un sensore comunica al sistema il suo stato, questi viene ricevuto dal driver. A sua volta il driver lo invia al Gestore Driver Sensori, il quale lo inserisce nella tabella Valori

Sensori. Dopo di che il Gestore Driver Sensori invia al Filtro Eventi la notifica che si è verificato l'evento collegato al sensore, cioè che c'è stata una variazione nella configurazione dei sensori. Per indicare che c'è stata questa variazione viene posto a 1 il flag "Variazione", mentre i sensori che non hanno avuto variazioni avranno il flag a 0. Il Filtro Eventi controllerà se questa variazione ha provocato il verificarsi di qualche evento complesso, e agirà di conseguenza. Se viene richiesto da un programma di controllo lo stato di un sensore, sarà l'Esecutore Programmi a richiederlo al Gestore Driver Sensori, che lo preleverà dalla suddetta tabella.

12. Driver Sensori e Attuatori

I driver sono la parte del Sistema di Controllo che si occupa di svincolare i dispositivi periferici dal sistema stesso. Rendono cioè possibile il collegamento al Sistema di Controllo di qualsiasi dispositivo utilizzabile come sensore o come attuatore, indipendentemente dalle caratteristiche elettriche e di protocollo di comunicazione del dispositivo stesso.

Anche gli input ed i sensori vengono visti a basso livello come gli altri dispositivi collegati al sistema, quindi anch'essi verranno pilotati da degli opportuni driver.

I *Driver Sensori* si occupano del colloquio tra i sensori ed il Gestore Driver Sensori, mentre i *Driver Attuatori* del colloquio con il Gestore Driver Attuatori.

I driver colloquiano con gli appositi gestori, tramite un insieme elementare di funzioni. Queste sono differenti per i Driver Sensori e i Driver Attuatori.

I Driver Sensori comunicheranno con il Gestore Driver Sensori tramite i seguenti comandi:

- *OPEN*: viene utilizzata dal Gestore Driver Sensori per inizializzare il driver e quindi fare in modo che il suo stato interno si allinei con lo stato dell'ambiente. Se il Sistema di Controllo si accorge di un funzionamento erroneo in un sensore, può richiedere al Gestore Driver Sensori di ripetere l'inizializzazione del driver tramite questo comando.
- *READ*: permette al Gestore Driver Sensori di leggere il valore di una caratteristica variabile di un dispositivo. Dà sempre origine ad una risposta da parte del dispositivo interrogato, che risponde con il suo stato attuale (cioè il valore attuale della caratteristica richiesta).
- *WRITE*: usato per scopi di controllo e configurazione, permette di modificare il valore di un parametro di stato di un dispositivo remoto, cambiando quindi la sua risposta alle sollecitazioni ambientali da parte del dispositivo stesso.
- *INFO*: usato dal dispositivo controllato per informare il Gestore Driver Sensori di un suo qualsiasi cambiamento di stato.

I Driver Attuatori dovranno essere in grado di accettare il seguente comando:

- *WRITE*: permette al Gestore Driver Attuatori di modificare lo stato di un attuatore.

Il concetto di attuatore e di sensore è esteso a qualsiasi dispositivo collegabile al sistema. Verranno quindi considerati attuatori anche sintetizzatori vocali, display CRT, ecc. Similmente verranno considerati sensori la tastiera, il telecomando, il riconoscitore vocale, ecc.

I dispositivi di segnalazione verso l'utente non verranno comandati dai programmi come gli altri attuatori, ma saranno pilotati tramite delle apposite istruzioni. Come però specificato poc'anzi il Sistema di Controllo li comanderà come gli altri attuatori.

13. Dispositivi complementari al sistema

A causa dell'estrema delicatezza dei compiti svolti dal Sistema di Controllo, occorre fare alcune considerazioni sui dispositivi complementari al sistema stesso.

Occorre prevedere un dispositivo di supervisione che si occupi di controllare che il sistema non si blocchi. Se questo dovesse avvenire, il suddetto dispositivo dovrà riavviare il sistema e riportarlo in una configurazione stabile. La procedura di riattivazione del sistema deve ovviamente essere la più rapida possibile.

Un altro aspetto del quale occorre tenere conto è la disponibilità di energia elettrica. A causa della struttura stessa del Sistema di Controllo, una interruzione nella fornitura dell'energia elettrica porterebbe alla paralisi completa del sistema, con la conseguente "immobilizzazione" dell'utente da esso dipendente. Occorre quindi che l'ambiente nel quale è installato il Sistema di Controllo sia dotato di un gruppo di continuità (UPS).

Se il sistema informativo che gestisce la Casa Intelligente viene utilizzato anche per altre funzioni, quali tele-lavoro, tele-shopping, home-banking, ecc., occorrerà inoltre dotarlo di periferiche appropriate al tipo di problemi di interazione che può avere l'utente.

14. Modulo di comunicazione con servizi di supporto

In un Sistema di Controllo per "Casa Intelligente" è importante permettere l'interazione tra programmi applicativi esterni al sistema e l'ambiente, tramite il sistema stesso.

Per permettere a questi programmi di influenzare il funzionamento del sistema, e di esserne a loro volta influenzati, viene reso disponibile un procedimento di scambio delle informazioni contenute nelle strutture dati proprie del Sistema di Controllo.

Tramite un'apposita serie di comandi è possibile per un programma esterno accedere in lettura ed in scrittura alle tabelle dati del Sistema di Controllo. Questo accesso avverrà in modo tale da non compromettere la coerenza dei dati in esse contenuti, evitando cioè di portare il Sistema di Controllo in una situazione instabile e potenzialmente pericolosa per l'utente. Tramite questi comandi si avvertirà anche il sistema che si sono effettuate delle modifiche al suo stato interno, permettendogli cosi di sincronizzarsi e di procedere in modo coerente nella gestione dell'ambiente.

I comandi per effettuare il prelievo e la modifica dei dati del sistema da parte dei programmi esterni sono caratterizzati dalle seguenti informazioni:

- *Struttura*: nome della struttura dati alla quale si vuole accedere.
- *Elemento*: elemento della struttura su cui operare
- *Operazione*: operazione da effettuare (lettura, scrittura).

L'esatto meccanismo con cui avviene questo colloquio tra il Sistema di Controllo ed i programmi esterni è strettamente legato alla piattaforma hardware/software scelta per l'implementazione. Per una descrizione dettagliata si faccia riferimento ai capitoli che descrivono le scelte implementative.

15. Istruzioni suddivise per categoria

La seguente tabella contiene l'elenco delle istruzioni disponibili per la creazione dei programmi di controllo per il Sistema di Controllo, suddivise per categoria.

Tabella 15.1: Istruzioni suddivise per categoria

	Eventi	Timer	Timeout	Programmi	Debug	Interazione con l'utente	Interazione con l'ambiente	Configura-zione
Abilita_Evento	✓						✓	
Annulla_Attesa_Evento	✓			✓			✓	
Annulla_Timer_Ciclico		✓		✓				
Aspetta		✓		✓				
Associa_Evento	✓			✓				✓
Attendi_Evento	✓		✓				✓	
Attiva_Ciclicamente		✓		✓				✓
Azzera_Contatore					✓		✓	
Comanda_Attuatore							✓	
Configura_Sensore							✓	✓
Controllo_Stato_Sistema					✓		✓	
Disabilita_Evento	✓			✓			✓	✓
Errore					✓			
Fine				✓				
Genera_Evento	✓			✓			✓	
Lancia				✓				
Leggi_Contatore					✓		✓	
Leggi_Ora		✓					✓	✓
Leggi_Stato							✓	
Risposta_Utente						✓		
Segnala_Utente						✓		
Se_Uguale				✓				
Se_Diverso				✓				
Se_Maggiore				✓				
Se_Minore				✓				
Teleallarme						✓	✓	

16. Istruzioni in ordine alfabetico

Questo capitolo contiene una descrizione dettagliata delle istruzioni utilizzabili per la scrittura dei programmi di controllo. Per ogni istruzione sono specificate le seguenti informazioni:

- *Descrizione*: riassume che cosa fa l'istruzione.
- *Sintassi*: fornisce la sintassi per richiamare correttamente l'istruzione.
- *Osservazioni*: questa sezione descrive che cosa fa l'istruzione, i parametri che richiede, e ogni dettaglio necessario per usare l'istruzione corrente ed eventualmente quelle a lei complementari.
- *Vedere anche*: le istruzioni collegate a quella corrente sono indicate qui.

Abilita_Evento

Descrizione Abilita il sistema alla ricezione di eventi da parte di un certo sensore, e quindi dei relativi valori di stato.

Sintassi Abilita_Evento (nome evento)

Osservazioni L'argomento *nome evento* è l'evento complesso da abilitare. Questa operazione viene anche eseguita automaticamente quando un programma di controllo termina la sua esecuzione.

Vedere anche Associa_Evento, Disabilita_Evento

Annulla_Attesa_Evento

Descrizione Annulla un'attesa con timeout su di un evento complesso.

Sintassi Annulla_Attesa_Evento (nome evento)

Osservazioni L'argomento *nome evento* è l'evento complesso sul quale vi è in attesa un programma di controllo. Utilizzato per interrompere l'attesa di un evento da parte di un altro programma. Può essere utilizzato per situazioni nelle quali occorre annullare delle operazioni attivate da un altro programma di controllo, che nella nuova configurazione ambientale sono diventate obsolete. La procedura che era in attesa sull'evento specificato dovrà essere abortita.

Vedere anche Attendi_Evento, Genera_Evento

Annulla_Timer_Ciclico

Descrizione Disabilita il timer ciclico responsabile dell'esecuzione ripetitiva di un certo programma di controllo.

Sintassi Annulla_Timer_Ciclico (nome programma)

Osservazioni L'argomento *nome programma* è il programma di controllo del quale si vuole terminare l'esecuzione ciclica. La possibilità di annullare i timer ciclici permette di cambiare la risposta del sistema agli eventi, se vengono a mutare le condizioni che avevano reso necessari i suddetti timer.

Vedere anche Attiva_Ciclicamente

Aspetta

Descrizione Attiva un timer, eventualmente interrompibile da un evento complesso.

Sintassi Aspetta (intervallo [, evento complesso])

Osservazioni L'argomento *intervallo* è l'intervallo di tempo da attendere prima di procedere con l'esecuzione del programma di controllo; il secondo

argomento, *evento complesso*, è opzionale ed è il nome di un evento complesso capace di interrompere l'attesa. Il nome dell'evento che verrà generato per riattivare la procedura sarà generato e gestito internamente dal sistema, quindi non sarà accessibile ai programmi di gestione.

Vedere anche Genera_Evento

Associa_Evento

Descrizione Associa ad un evento complesso un programma di controllo.

Sintassi Associa_Evento (nome evento , programma di controllo)

Osservazioni Permette di associare ad un evento complesso un particolare programma di controllo, permettendo così di poter variare in modo dinamico la configurazione del sistema in fase di esecuzione di un programma. L'argomento *nome evento* è il nome dell'evento complesso al quale associare l'esecuzione del *programma di controllo*. Utilizzabile sia in fase di configurazione sia in fase di esecuzione del programma di controllo.

Vedere anche Abilita_Evento, Disabilita_Evento, Genera_Evento

Attendi_Evento

Descrizione Pone il programma di controllo in attesa dell'arrivo dei dati da un sensore.

Sintassi Attendi_Evento (evento complesso , timeout)

Osservazioni L'argomento *evento complesso* è l'evento che si verifica quando arrivano i dati dal sensore oggetto dell'attesa. Il *timeout* associato rappresenta il tempo massimo che si vuole attendere per ottenere la risposta. Dopo questa istruzione ne andrà inserita, se necessaria, una di confronto dello stato assunto dal sensore interessato da questa attesa con un valore di riferimento. Più eventi complessi potranno essere generati quando avviene una modifica nello stato di un sensore. Se si verifica il timeout, il valore da utilizzare per effettuare il confronto sarà rappresentato dalla costante TIMEOUT.

Vedere anche Abilita_Evento, Annulla_Attesa_Evento, Disabilita_Evento, Genera_Evento

Attiva_Ciclicamente

Descrizione Imposta l'esecuzione ciclica di un programma di controllo.

Sintassi Attiva_Ciclicamente (nome programma , intervallo di tempo | orario)

Osservazioni Occorre specificare il *nome programma* che deve essere eseguito ogni *intervallo di tempo* o all'*orario* prestabilito. Tramite questa istruzione è possibile abilitare una procedura ad essere eseguita

ciclicamente. La scadenza ciclica alla quale la procedura deve essere eseguita può essere specificata indicando un intervallo di tempo o un orario prestabilito. L'Esecutore Programmi si occuperà di assegnare un evento timer a questa procedura, per poterla risvegliare ad intervalli regolari. Il timer che si occupa del risveglio ciclico di questa procedura non verrà mai rimosso dalla tabella del Gestore Timer, tranne nel caso in cui venga richiamata l'apposita istruzione Annulla_Timer_Ciclico. Il programma di controllo del quale si richiede l'attivazione ciclica deve avere già presente nella tabella Eventi Complessi l'evento necessario alla sua attivazione. Se così non fosse, questa istruzione deve essere preceduta da un appropriato richiamo dell'istruzione Associa_Evento.

Vedere anche Annulla_Timer_Ciclico, Associa_Evento

Azzera_Contatore

Descrizione Azzera il valore del contatore associato ad un sensore.

Sintassi Azzera_Contatore (nome sensore)

Osservazioni L'argomento *nome sensore* è il nome del sensore del quale si vuole azzerare il valore del campo contatore.

Vedere anche Leggi_Contatore

Comanda_Attuatore

Descrizione Permette di interagire con l'ambiente, modificando la posizione degli attuatori.

Sintassi Comanda_Attuatore (nome attuatore , stato)

Osservazioni Con questo comando si invia all'attuatore *nome attuatore* lo *stato* che deve assumere. Questo stato può essere acceso / spento ma non solo, perché si possono avere degli attuatori che accettano come comando un indicazione della posizione da assumere.

Vedere anche

Configura_Sensore

Descrizione Modifica le caratteristiche di funzionamento del sensore.

Sintassi Configura_Sensore (nome sensore , parametro , valore)

Osservazioni L'argomento *nome sensore* specifica quale sensore deve essere configurato. La configurazione consisterà nel porre nel *parametro* il nuovo *valore* che esso deve assumere. Questo comando serve per impostare i parametri caratteristici di funzionamento del sensore, in modo da poterne cambiare il comportamento e le funzionalità

Vedere anche

Controllo_Stato_Sistema

Descrizione Confronta lo stato corrente di alcuni sensori del sistema con una configurazione di riferimento. Restituisce una segnalazione di errore se vi sono differenze.

Sintassi Controllo_Stato_Sistema (configurazione di riferimento)

Osservazioni L'argomento *configurazione di riferimento* sarà un file contenente una serie di coppie di valori, nelle quali sono rappresentati i sensori ed i valori che si vuole confrontare con gli attuali. Questa istruzione è utile per analisi diagnostiche del sistema, per fare in modo che nel caso si presenti una situazione anomala nello stato dei sensori possa essere interpellato il Centro Servizi per un'analisi più approfondita del problema. Per esempio, si potrebbe avere una configurazione standard diurna ed una notturna, da confrontare automaticamente con quella corrente ad intervalli regolari; se queste differiscono si può attivare un'apposita procedura di controllo dell'ambiente. Queste configurazioni di riferimento potranno non avere un valore di riferimento per tutti i sensori presenti nel sistema.

Vedere anche Leggi_Stato

Disabilita_Evento

Descrizione Permette di fare in modo che il sistema non risponda all'occorrere di un particolare evento.

Sintassi Disabilita_Evento (nome evento)

Osservazioni L'argomento *nome evento* è l'evento complesso del quale si vuol far ignorare il verificarsi al Sistema di Controllo. Viene richiamata internamente dal sistema ogni volta che viene attivato un programma di controllo.

Vedere anche Abilita_Evento, Genera_Evento

Errore

Descrizione Segnalazione di avvenuto malfunzionamento nel sistema, ad uso del personale tecnico.

Sintassi Errore (messaggio)

Osservazioni L'argomento *messaggio* è una stringa utilizzabile per la descrizione dettagliata della situazione interna del programma di controllo, all'interno del quale si è rilevata la condizione di funzionamento anomalo. In base al livello di diagnostica attivato, questa istruzione provvederà ad inserire in un file di storico le informazioni necessarie per la diagnostica del Sistema di Controllo. Oltre al messaggio inserito dal programma di controllo che ha rilevato una condizione di errore, vi verrà scritto il nome del programma attualmente in esecuzione, la data e l'ora e una copia dello stato attuale del sistema.

Potrà altresì esserci una segnalazione immediata della condizione di errore verificatasi al Centro Servizi. Il Centro Servizi e il Tecnico potranno inoltre ottenere dal sistema in qualunque momento un file contenente lo stato globale del sistema, cioè i valori attuali di tutti i sensori. Questa funzione è utilizzabile per effettuare un controllo remoto "rapido" del sistema da parte di un Centro Servizi, che può così avere una fotografia della situazione corrente all'interno dell'abitazione. Può inoltre venire utilizzato a scopo diagnostico da parte del Tecnico incaricato della manutenzione del sistema. Queste informazioni diagnostiche saranno esclusivamente ad uso del personale qualificato.

Vedere anche Controllo_Stato_Sistema

Fine

Descrizione Termina l'esecuzione del programma di controllo corrente.

Sintassi Fine

Osservazioni Può essere inserita in qualunque punto del programma di controllo. All'interno di un programma di controllo questa istruzione può essere presente più volte. Non è necessario utilizzarla come ultima istruzione nei programmi.

Vedere anche Esegui

Genera_Evento

Descrizione Simula il verificarsi di un evento.

Sintassi Genera_Evento (nome evento)

Osservazioni L'argomento *nome evento* è l'identificatore di un evento complesso presente nel sistema. Permette di simulare dall'interno di una procedura il verificarsi di un evento complesso atto a scatenare un certo programma di controllo. Non provoca una sospensione del programma attualmente in fase di esecuzione.

Vedere anche Attendi_Evento, Associa_Evento

Lancia

Descrizione Esegue un programma di controllo.

Sintassi Lancia (programma di controllo)

Osservazioni Permette di richiamare dall'interno di un programma di controllo un altro programma. L'argomento *programma di controllo* è il nome del programma di controllo da attivare. Il programma che esegue questa istruzione termina dopo averla eseguita.

Vedere anche Associa_Evento, Genera_Evento, Fine

Leggi_Contatore

Descrizione Legge il valore assunto dal contatore associato ad un sensore

Sintassi Leggi_Contatore (nome sensore)

Osservazioni L'argomento *nome sensore* è l'identificatore del sensore del quale si vuole leggere il valore del contatore ad esso associato. L'ultimo valore letto da questa istruzione viene mantenuto in un buffer interno all'Esecutore Programmi e viene utilizzato dalle istruzioni di confronto.

Vedere anche Azzera_Contatore

Leggi_Ora

Descrizione Legge l'ora attuale dal calcolatore.

Sintassi Leggi_Ora

Osservazioni Il valore letto da questa istruzione viene inserito nel buffer interno all'Esecutore Programmi per essere utilizzato dalle istruzioni di confronto. Il valore dell'istante attuale è composto dall'indicazione dell'ora e dei minuti, ed è confrontabile con costanti aventi formato *hhmm*, con $0 \leq hh \leq 23$ e $0 \leq mm \leq 59$.

Vedere anche Se_Uguale, Se_Diverso, Se_Maggiore, Se_Minore

Leggi_Stato

Descrizione Legge lo stato assunto da un sensore.

Sintassi Leggi_Stato (nome sensore)

Osservazioni L'argomento *nome sensore* è l'identificatore del sensore del quale si vuole leggere lo stato corrente. Serve per la lettura del valore assunto da un sensore, senza che il programma di controllo si debba porre in uno stato di attesa. L'ultimo valore letto da questa istruzione viene mantenuto in un buffer interno all'Esecutore Programmi e viene utilizzato dalle istruzioni di confronto.

Vedere anche Se_Uguale, Se_Diverso, Se_Maggiore, Se_Minore

Risposta_Utente

Descrizione Attende una risposta dall'utente tramite una modalità specificata.

Sintassi Risposta_Utente (codice risposta)

Osservazioni L'argomento *codice risposta* è l'identificatore del tipo di risposta che si vuole attendere dall'utente, e avrà una riga corrispondente nella

tabella Risposte Utente. Tramite questa tabella verranno associati alla suddetta risposta una serie di sensori dai quali attendere una risposta dall'utente. Il valore restituito da questa istruzione viene mantenuto in un buffer interno all'Esecutore Programmi e viene utilizzato dalle istruzioni di confronto.

Vedere anche Segnala_Utente

Segnala_Utente

Descrizione Effettua una comunicazione all'utente utilizzando i dispositivi appropriati.

Sintassi Segnala_Utente (codice messaggio)

Osservazioni L'argomento *codice messaggio* sarà la segnalazione che si vuole inviare all'utente ed avrà una riga corrispondente nella tabella Segnalazioni Utente, tramite la quale verranno associati al suddetto codice messaggio una serie di dispositivi da utilizzare per la segnalazione all'utente.

Vedere anche Risposta_Utente

Se_Uguale

Descrizione Confronta l'ultimo valore restituito da un'istruzione Leggi_Stato con un valore di riferimento per vedere se sono uguali.

Sintassi Se_Uguale (valore , linea | FINE)

Osservazioni L'argomento *valore* viene confrontato con quello contenente il risultato dell'ultima lettura dello stato di un sensore. Se il confronto ha esito positivo salta all'istruzione contenuta nella *linea* specificata o termina il programma di controllo (*FINE*), mentre se ha esito negativo continua l'esecuzione con l'istruzione successiva.

Vedere anche Leggi_Stato, Se_Diverso, Se_Maggiore, Se_Minore

Se_Diverso

Descrizione Confronta l'ultimo valore restituito da un'istruzione Leggi_Stato con un valore di riferimento per vedere se sono diversi.

Sintassi Se_Diverso (valore , linea | FINE)

Osservazioni L'argomento *valore* viene confrontato con quello contenente il risultato dell'ultima lettura dello stato di un sensore. Se il confronto ha esito positivo salta all'istruzione contenuta nella *linea* specificata o termina il programma di controllo (*FINE*), mentre se ha esito negativo continua l'esecuzione con l'istruzione successiva.

Vedere anche Leggi_Stato, Se_Uguale, Se_Maggiore, Se_Minore

Se_Maggiore

Descrizione Confronta l'ultimo valore restituito da un'istruzione Leggi_Stato con un valore di riferimento per vedere se il valore letto è maggiore.

Sintassi Se_Maggiore (valore, linea | FINE)

Osservazioni L'argomento *valore* viene confrontato con quello contenente il risultato dell'ultima lettura dello stato di un sensore. Se il confronto ha esito positivo salta all'istruzione contenuta nella *linea* specificata o termina il programma di controllo (*FINE*), mentre se ha esito negativo continua l'esecuzione con l'istruzione successiva.

Vedere anche Leggi_Stato, Se_Uguale, Se_Diverso, Se_Minore

Se_Minore

Descrizione Confronta l'ultimo valore restituito da un'istruzione Leggi_Stato con un valore di riferimento per vedere se il valore letto è minore.

Sintassi Se_Minore (valore, linea | FINE)

Osservazioni L'argomento *valore* viene confrontato con quello contenente il risultato dell'ultima lettura dello stato di un sensore. Se il confronto ha esito positivo salta all'istruzione contenuta nella *linea* specificata o termina il programma di controllo (*FINE*), mentre se ha esito negativo continua l'esecuzione con l'istruzione successiva.

Vedere anche Leggi_Stato, Se_Uguale, Se_Diverso, Se_Maggiore

Teleallarme

Descrizione Provoca l'invio di una richiesta di teleallarme specificando che tipo di allarme si è verificato e quale sensore lo ha provocato.

Sintassi Teleallarme (tipo allarme, nome sensore)

Osservazioni L'argomento *tipo allarme* deve essere uno dei seguenti: gas, acqua, intrusione, incendio, elettrico, teleallarme. Il *nome sensore* servirà eventualmente al Centro Servizi per sapere in che modo occorre intervenire, in modo tale da poter graduare l'intervento in base alla gravità dell'accaduto.

Vedere anche Risposta_Utente, Segnala_Utente

17. Specifiche dettagliate del comportamento del sistema

Verranno ora descritte le operazioni elementari eseguite dal Sistema di Controllo per eseguire le istruzioni contenute nei programmi di controllo e per svolgere i compiti base necessari al corretto funzionamento del sistema stesso.

17.1 Modifica nello stato di un sensore

Sensore:

Il sensore varia di stato.

Driver:

Il driver ne rileva la variazione e la segnala al Gestore Driver Sensori.

Gestore Driver Sensori:

Il Gestore Driver Sensori riceve dal driver la notifica della variazione dello stato di un sensore (tramite il comando INFO) e richiede al driver stesso lo stato assunto dal sensore (tramite il comando READ). Notifica al Filtro Eventi che è avvenuto l'evento associato al sensore e inserisce nella tabella Stato Sensore lo stato restituito dal sensore stesso, indicando quale sensore è variato tramite il flag "Variazione".

Filtro Eventi:

Inizia a scorrere la tabella Eventi Complessi per vedere se uno o più eventi complessi si sono verificati. Vengono ignorati gli Eventi Complessi aventi il flag "Mask" uguale a 1. Si inizia ad analizzare quelli aventi priorità Urgente e si continua con quelli aventi priorità Normale. Gli eventi complessi vengono analizzati solo se almeno uno degli eventi semplici che li compongono è variato rispetto alla scansione precedente della tabella Eventi Complessi. Tutti quelli che vengono trovati attivi sono posti, con la loro priorità, nella tabella Eventi Da Servire per essere serviti dall'Esecutore Programmi, nell'ordine con il quale sono stati trovati nella tabella Eventi Complessi. A questi eventi viene posto a 1 il flag "EAC".

Agli eventi complessi che vengono attivati verrà posto a 1 il flag "Mask", per impedire che i programmi ad essi collegati vengano attivati più volte sullo stesso evento complesso finché non sono terminati. Naturalmente si potranno avere più copie dello stesso programma di controllo, ma collegate ad eventi complessi differenti.

Terminata la scansione della tabella Eventi Complessi verrà azzerata la colonna della tabella Stato Sensori contenente il flag "Variazione".

Tra gli eventi complessi che verranno attivati ci potranno essere anche quelli che si occupano di riattivare i programmi momentaneamente sospesi perché in attesa di dati dai sensori. Questi eventi complessi sono individuati tramite il flag "Wait" posto a 1. La riga che li riguarda andrà rimossa dalla tabella Eventi Complessi, perché devono essere attivati una ed una sola volta, ed inoltre non devono poter essere rimossi dalla tabella Eventi da Servire se l'evento complesso che li ha attivati viene a cessare prima che siano stati riattivati i programmi di controllo ad essi collegati.

Un altro tipo di eventi complessi presenti nella tabella sono quelli collegati al Gestore Timer ed al Gestore Timeout.

Gli eventi generati dal Gestore Timer si verificheranno quando un timer scade. Essi dovranno essere inseriti nella tabella Eventi Complessi con l'indicazione che l'evento deve

riattivare un programma di controllo sospeso, quindi con il flag "Wait" posto a 1, e con una priorità Normale. Nella tabella Eventi Complessi il Gestore Timer inserirà anche degli eventi complessi capaci di interrompere un timer attualmente in esecuzione, prima che questo raggiunga la sua scadenza naturale. Essi avranno il flag "Timer" posto a 1 per indicare che sono utilizzati per annullare un timer e quindi il Filtro Eventi, quando uno di questi eventi si verifica, dovrà segnalare questo fatto al Gestore Timer.

Il Gestore Timeout inserirà nella tabella Eventi Complessi un evento complesso con il flag "Timeout" posto a 1 indicante che l'evento è collegato ad un timeout. Se l'evento si verifica, esso verrà trattato allo stesso modo di un evento normale, con l'unica differenza che verrà effettuata una segnalazione al Gestore Timeout per fare in modo che elimini il timer di timeout dalla sua tabella.

Se invece il timeout scade, il Gestore Timeout eliminerà dalla tabella Eventi Complessi l'evento complesso che avrebbe dovuto sospenderlo e inserisce nella tabella Eventi da Servire l'evento complesso che deve riattivare il programma di controllo, ma inserendo come valore associato al sensore su cui si era in attesa, nella tabella Stato Sensori, la segnalazione di avvenuto timeout.

Verrà effettuata anche una scansione per verificare se nella tabella Eventi Complessi si hanno degli eventi complessi la cui condizione è ora non più verificata ma che devono ancora essere serviti, cioè sono ancora in attesa nella tabella Eventi Da Servire. Questi andranno rimossi dalla tabella Eventi Da Servire ed ad essi andrà anche azzerato il flag "EAC".

Esecutore Programmi:

L'Esecutore Programmi preleva il primo evento presente nella tabella Eventi Da Servire (utilizzando una politica ben definita per le priorità), lo elimina e azzera il flag "Evento Attualmente in Coda" nella tabella Eventi Complessi. Occorrerà anche porre ad uno il flag "Mask" per impedire che vengano attivate più copie dello stesso programma di controllo. Questo flag andrà azzerato solo quando il programma raggiungerà il termine.

Dopo di che ricerca nella tabella Programmi l'identificativo del programma di controllo associato all'evento prelevato dalla tabella Eventi Da Servire e lo avvia a partire dall'istruzione specificata nel campo "Istruzione Iniziale". Se questo programma di controllo è uno di quelli che era stato sospeso perché in attesa di un valore da un sensore, l'istruzione da cui farlo ripartire sarà diversa dalla prima.

Le istruzioni contenute nel programma di controllo saranno analizzate ed eseguite sequenzialmente, finché non si troverà un'istruzione di attesa su di un evento. Il programma di controllo verrà sospeso e verrà aggiornata la tabella Eventi Complessi. Questo aggiornamento consisterà nell'inserire una linea nella tabella contenente l'evento complesso che deve risvegliare il programma di controllo, il flag "Wait" indicante che questo evento deve solo riattivare un programma sospeso (quindi questa riga dovrà essere rimossa appena l'evento si verifica e viene attivato) ed il livello di priorità Normale. Sospeso il programma di controllo, l'Esecutore Programmi preleva dalla tabella Eventi Da Servire un nuovo evento.

Quando il programma di controllo arriva al suo termine naturale, il flag "Mask" viene azzerato, in modo da permetterne l'ulteriore esecuzione.

17.2 Attivazione di un timer non ciclico

Esecutore programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione, viene trovata un'istruzione di attesa (Aspetta). Questa istruzione avrà associato un intervallo di tempo da attendere, ed eventualmente un evento complesso capace di interrompere questa attesa.

Questi dati verranno passati al Gestore Timer che restituirà un codice evento particolare, tramite il quale si segnalerà al Filtro Eventi che l'intervallo di tempo è trascorso e che il programma di controllo dovrà essere riattivato.

L'eventuale evento complesso capace di interrompere il timer verrà inviato al Filtro Eventi che lo inserirà nella tabella Eventi Complessi, con l'indicazione che è un evento utilizzato per interrompere un timer, tramite il flag "Timer" uguale a 1.

Verranno poi inseriti i dati riguardanti il programma da risvegliare nella tabella Programmi. Questi dati consisteranno nell'identificativo del programma di controllo corrente, nella posizione dalla quale farlo ripartire, cioè l'istruzione successiva a quella che ha richiesto l'uso del timer, e l'evento capace di riattivare il programma stesso (è il nome evento restituito dal Gestore Timer).

Dopo di che il programma di controllo che ha richiesto l'uso del timer verrà sospeso.

Gestore Timer:

Verrà creata una nuova riga nella tabella Timer, contenente l'evento da generare per riattivare il programma di controllo (generato casualmente ma univoco nel sistema), la scadenza temporale del timer e l'indicazione che è un timer singolo. Verrà anche inserito nella tabella Eventi Complessi un evento complesso capace di interrompere anticipatamente il timer attivato, insieme con l'indicazione che è un evento collegato ad un timer.

Ciclicamente verrà scandita la tabella Timer. Se un timer non ciclico è scaduto, esso verrà rimosso dalla tabella Timer e verrà anche eliminato dalla tabella Eventi Complessi l'evento capace di interromperlo, perché ormai non è più necessario. Verrà quindi generato l'evento necessario a riattivare il programma di controllo.

Se invece si verifica l'evento complesso collegato all'annullamento del timer, il Gestore Timer riceverà la segnalazione dal Filtro Eventi, e dovrà eliminare dalla tabella Timer il timer annullato, eliminare dalla tabella Eventi l'evento collegato all'annullamento timer e generare l'evento necessario a risvegliare il programma che era in attesa sul timer.

17.3 Attivazione di un timer ciclico

Esecutore programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione di attivazione ciclica di un programma (Attiva_Ciclicamente). All'istruzione sarà associato l'identificativo del programma di controllo da attivare ciclicamente ed un intervallo di tempo tra un'attivazione e la successiva, oppure un orario prestabilito al quale attivare il programma. L'esecuzione ciclica potrà essere annullata tramite l'istruzione Annulla_Timer_Ciclico. Questi dati verranno passati al Gestore Timer e il programma di controllo continuerà la sua esecuzione.

Gestore Timer:

Verrà creata una nuova riga nella tabella Timer, contenente l'evento da generare per attivare il programma di controllo (prelevato dalla tabella Programmi), l'intervallo di tempo o l'orario prestabilito per la sua attivazione e l'indicazione che è un timer ciclico, quindi che non deve essere eliminato dalla tabella la prima volta che viene attivato.

Nella tabella Programmi dovrà essere presente l'evento complesso capace di attivare il programma di controllo collegato a questo timer ciclico e l'identificativo del programma stesso. Se non è già presente, occorre inserirlo da programma con il richiamo dell'istruzione Associa_Evento.

Ciclicamente verrà scandita la tabella Timer. Se un timer ciclico deve essere attivato, verrà posto nella tabella Eventi da Servire l'evento complesso necessario ad attivare il programma di controllo a lui collegato.

17.4 Eliminazione di un timer ciclico

Esecutore programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione di eliminazione dell'attivazione ciclica di un programma (Annulla_Timer_Ciclico). Questa istruzione avrà associato l'identificativo del programma da eliminare dalla tabella Timer. Questi dati verranno passati al Gestore Timer e il programma continuerà la sua esecuzione.

Gestore Timer:

Verrà ricercato nella tabella Programmi l'evento complesso corrispondente al programma di controllo del quale si vuole eliminare il richiamo ciclico, presente nel campo "Evento Attivante", e si eliminerà la riga corrispondente. Dovrà anche venire eliminata dalla tabella Timer la riga corrispondente al timer ciclico stesso, identificata tramite l'evento utilizzato per l'attivazione del programma.

17.5 Attesa di un evento

Esecutore programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione di attesa di un evento (Attendi_Evento). Questa istruzione avrà associato l'identificativo dell'evento complesso da attendere e un intervallo di tempo massimo entro il quale si vuole ottenere la risposta, ovvero un "timeout".

Questi dati verranno passati al Gestore Timeout e verrà inviato al Filtro Eventi l'evento da attendere con l'indicazione che si tratta di un evento collegato ad un timeout. Nella tabella Programmi sarà inserita una riga contenente l'evento complesso da attendere, l'identificativo del programma di controllo e l'istruzione da cui riattivare l'esecuzione, perciò quella seguente rispetto a quella che ha richiesto l'attesa.

Dopo aver fatto questi aggiornamenti nelle strutture del sistema, il programma di controllo che ha richiesto l'attesa verrà sospeso.

Gestore Timeout:

Verrà creata una nuova riga nella tabella Timeout, contenente l'evento sul quale vi è l'attesa e l'intervallo di tempo massimo da attendere.

Ciclicamente verrà scandita la tabella Timeout. Se un timeout si è verificato, verrà innanzitutto rimossa la riga corrispondente dalla tabella Timeout. Dopo di che si dovrà cercare nella tabella Eventi Complessi l'evento corrispondente e attivarlo, associandogli però come stato l'indicazione che non sono stati ricevuti in tempo i dati dal sensore, ma si è verificato il timeout. Andrà inoltre eliminata la riga corrispondente all'evento atteso dalla tabella Eventi Complessi.

Filtro Eventi:

Se si verifica l'evento prima che avvenga il timeout, le operazioni da effettuare sono a carico del Filtro Eventi. Esso dovrà eliminare dalla tabella Timeout la riga corrispondente

all'evento che si è verificato, inserire nella tabella Eventi Da Servire l'evento con una appropriata priorità, ed eliminare dalla tabella Eventi Complessi la riga corrispondente.

17.6 Annullamento dell'attesa di un evento

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione di annullamento dell'attesa di un evento da parte di un altro programma (Annulla_Attesa_Evento). A questa istruzione sarà associato l'evento complesso sul quale è in attesa il programma di controllo da annullare. Questo dato verrà passato al Gestore Timeout ed al Filtro Eventi.

L'Esecutore Programmi dovrà ricercare nella tabella Programmi la riga contenente l'identificativo del programma di controllo collegato all'evento ed eliminarlo dalla tabella Programmi. Tramite il campo "PID" si avrà l'associazione univoca con la copia del programma di controllo utilizzata, e perciò con l'evento complesso responsabile della sua attivazione. Questo evento complesso verrà passato al Filtro Eventi.

Gestore Timeout:

Verrà eliminata dalla tabella Timeout la riga corrispondente all'evento indicato nell'istruzione di annullamento dell'attesa, ricevuto dall'Esecutore Programmi.

Filtro Eventi:

Occorrerà eliminare dalla tabella Eventi Complessi la riga corrispondente all'evento sul quale vi è l'attesa, e modificare il flag "Mask" dell'evento responsabile dell'attivazione del programma di controllo che era sospeso sull'evento, per permettere che possa venire nuovamente attivato quando richiesto. Quest'ultimo evento viene ricevuto dall'Esecutore Programmi.

17.7 Abilitazione di un evento

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione di richiesta di abilitazione di un evento complesso (Abilita_Evento). Questa istruzione avrà associato l'identificativo dell'evento complesso da abilitare, che verrà passato al Filtro Eventi.

Filtro Eventi:

Viene ricercata nella tabella Eventi Complessi la riga corrispondente all'evento richiesto, e viene posto a 0 il flag "Mask". Questa operazione viene anche effettuata automaticamente dal sistema quando un programma di controllo termina la sua esecuzione.

17.8 Disabilitazione di un evento

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione di richiesta di disabilitazione di un evento complesso (Disabilita_Evento). Questa istruzione avrà associato l'identificativo dell'evento complesso da disabilitare, che verrà passato al Filtro Eventi.

Filtro Eventi:

Viene ricercata nella tabella Eventi Complessi la riga corrispondente all'evento richiesto, e viene posto a 1 il flag "Mask". Questa operazione viene anche effettuata automaticamente dal sistema quando un programma di controllo viene attivato.

17.9 Comando di un attuatore

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione di richiesta di modifica dello stato di un attuatore (Comanda_Attuatore). Questa istruzione avrà associato l'identificativo dell'attuatore e lo stato che esso dovrà assumere. Questi dati verranno inviati al Gestore Driver Attuatori.

Gestore Driver Attuatori:

Invia al driver dell'attuatore interessato i dati riguardanti lo stato che deve assumere l'attuatore stesso. Per effettuare questa operazione viene usato il comando WRITE.

17.10 Configurazione di un sensore

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione di richiesta di modifica della configurazione di un sensore (Configura_Sensore). Questa istruzione avrà associato l'identificativo del sensore, il nome del parametro da variare ed il valore che deve assumere il parametro stesso. Questi dati verranno inviati al Gestore Driver Sensori.

Gestore Driver Sensori:

Invia al driver del sensore interessato i dati riguardanti il parametro da modificare ed il valore che esso deve assumere. Per effettuare questa operazione viene utilizzato il comando WRITE.

17.11 Confronto dello stato del sistema con una configurazione di riferimento

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione per il confronto dello stato attuale del Sistema di Controllo con una configurazione di riferimento (Controllo_Stato_Sistema). Associato a questa istruzione vi sarà il nome di un file contenente una serie di coppie di valori consistenti nel nome sensore e del valore che deve avere. Non è obbligatorio che esista una coppia di valori per ogni sensore presente nel sistema.

L'Esecutore Programmi scandirà queste coppie e confronterà i valori di riferimento con quelli restituiti dal Gestore Driver Sensori. Alla prima discrepanza si terminerà questa comparazione e verrà inserita la segnalazione di errore nella variabile (nel buffer) utilizzata dall'istruzione Se_*.

17.12 Associazione di un evento ad un programma di controllo

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione per associare ad un evento complesso un programma di controllo (Associa_Evento). Associato a questa istruzione vi sarà il nome dell'evento e quello del programma di controllo da associare ad esso.

Questa istruzione sarà utilizzabile solo all'interno di un programma di controllo, ma le sue funzioni saranno realizzate anche da un'apposita opzione della procedura di configurazione del sistema.

L'Esecutore Programmi inserirà questi dati nella tabella Programmi, generando in modo univoco il campo "PID". Nella tabella Eventi Complessi verrà inserita una nuova riga contenente l'evento complesso necessario ad attivare il programma di controllo. L'evento complesso verrà abilitato tramite il flag "Mask".

17.13 Segnalazione di problemi nel sistema

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione per la segnalazione diagnostica di un malfunzionamento nel sistema (Errore). Associato a questa istruzione vi sarà un messaggio personalizzabile dal creatore del programma, contente un indicazione dettagliata riguardante quale parte del programma di controllo ha rilevato l'errore.

In base al livello di diagnostica che è stato attivato (presente in un'apposita variabile di sistema), verranno inserite una serie di informazioni in un file di storico. Queste informazioni verranno utilizzate per analizzare e risolvere i malfunzionamenti del sistema. Oltre al messaggio inserito dal programma di controllo che ha rilevato una condizione di errore, vi verrà scritto il nome del programma attualmente in esecuzione, la data e l'ora e una copia dello stato attuale del sistema. Potrà altresì esserci una segnalazione immediata al Centro Servizi della condizione di errore verificatasi.

17.14 Termine di un programma di controllo

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione provocante la fine del programma di controllo stesso (Fine).

Verrà riabilitato l'evento complesso collegato all'attivazione del programma, ponendo a 0 il flag "Mask" presente nella tabella Eventi Complessi. Occorrerà anche porre ad 1 il campo della Tabella Programmi contenente l'indicazione dell'istruzione dalla quale attivare il programma. L'evento complesso responsabile dell'attivazione del programma di controllo verrà identificato nella tabella Programmi tramite il campo "PID". Il programma verrà concluso e l'Esecutore Programmi passerà a gestire un altro evento complesso in attesa di essere servito nella tabella Eventi Da Servire.

17.15 Generazione simulata di un evento

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione per la generazione simulata di un evento complesso (Genera_Evento). Associato a questa istruzione vi sarà il nome dell'evento complesso da generare.

L'evento complesso verrà analizzato e scomposto negli eventi semplici. Per questi eventi semplici verrà attivato il flag "Variazione" presente nella tabella Eventi Semplici. Dopo di che verranno effettuate le stesse operazioni di ricerca degli eventi complessi attivati e loro inserimento nella tabella Eventi Da Servire. Un'altra soluzione è non scomporre l'evento complesso negli eventi semplici che lo compongono, e limitarsi ad inserirlo nella tabella Eventi Da Servire con priorità Normale, configurando in modo opportuno i flag dell'evento stesso. Lo svantaggio fondamentale della prima tecnica è la possibile attivazione di altri eventi complessi, grazie agli eventi semplici indicati come variati. È quindi conveniente usare il secondo approccio per conservare lo scopo per il quale è stata studiata questa istruzione, cioè la semplice attivazione di altri programmi di controllo.

17.16 Lettura dello stato di un sensore

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione per la lettura dello stato di un sensore (Leggi_Stato). Come parametro di questa istruzione vi sarà il nome del sensore del quale si vuole sapere lo stato assunto.

L'Esecutore Programmi richiederà al Gestore Driver Sensori di restituire il valore assunto dal sensore specificato e porrà questo valore nella sua variabile interna usata dalle istruzioni di confronto.

Gestore Driver Sensori:

Il Gestore Driver Sensori andrà a ricercare nella tabella Stato Sensori il valore associato al sensore specificato e lo restituirà all'Esecutore Programmi. Questi valori sono immediatamente disponibili perché letti precedentemente, quindi non si avranno attese.

17.17 Segnalazione all'utente

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione per la segnalazione all'utente di un messaggio (Segnala_Utente). Associato come parametro di questa istruzione vi sarà l'identificativo del messaggio da inviare all'utente.

Tramite la tabella Segnalazioni Utente, l'Esecutore Programmi ricerca quali sono i dispositivi da attivare per eseguire correttamente la segnalazione e li attiva in sequenza.

17.18 Risposta dall'utente

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione per l'attesa di una risposta dall'utente (Risposta_Utente). Associato come parametro di questa istruzione vi sarà l'identificativo della risposta da attendere dall'utente.

Tramite la tabella Risposte Utente, l'Esecutore Programmi ricerca quali sono i sensori dai quali attendere la risposta e inserisce nella tabella Eventi Complessi un apposito evento complesso costituito dall'OR di tutti i sensori presenti nella suddetta tabella. Il Programma verrà poi posto in attesa di una risposta, con le stesse modalità dell'attesa di un valore da un sensore.

17.19 Salto condizionato

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata una delle istruzioni responsabili dei salti condizionati (Se_*). Questa istruzione avrà come parametri il valore da confrontare e il numero della linea del programma di controllo a cui saltare se il confronto è verificato. Al posto di un numero di linea è possibile avere l'indicazione che se il confronto è verificato, occorre terminare il programma di controllo.

Il valore da confrontare viene comparato con il valore di riferimento contenuto nella variabile interna all'Esecutore Programmi. Se il test ha risultato positivo, viene effettuato un salto alla riga indicata o viene conclusa l'esecuzione del programma di controllo.

Nel primo caso viene sostituita la linea di programma di controllo attualmente in esecuzione con quella presente alla riga indicata nell'istruzione di confronto, mentre nel secondo caso vengono effettuate le stesse operazioni caratterizzanti l'esecuzione dell'istruzione Fine.

17.20 Azzeramento di un contatore

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione utilizzata per l'azzeramento del contatore associato ad un particolare sensore (Azzera_Contatore). Questa istruzione avrà come parametro il nome del sensore del quale occorre azzerare il valore del contatore associato.

L'Esecutore Programmi andrà a modificare all'interno della tabella Stato Sensori il campo Contatore corrispondente al sensore specificato, ponendolo a zero.

17.21 Lettura di un contatore

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione utilizzata per la lettura del valore assunto dal contatore associato ad un particolare sensore (Leggi_Contatore). Questa istruzione avrà come parametro il nome del sensore dal quale si vuole leggere il valore del contatore.

L'Esecutore Programmi andrà a prelevare dalla tabella Stato Sensori il campo Contatore corrispondente al sensore specificato, e porrà questo valore nella sua variabile interna usata dalle istruzioni di confronto.

17.22 Richiamo di un programma di controllo dall'interno di un altro.

Esecutore Programmi:

Durante l'analisi del programma di controllo attualmente in fase di esecuzione viene trovata un'istruzione utilizzata per l'attivazione di un programma di controllo (Lancia). Questa istruzione avrà come parametro il nome del programma di controllo da richiamare.

Il programma attualmente in esecuzione dovrà terminare. Per far questo occorrerà riabilitare l'evento complesso collegato all'attivazione del programma, ponendo a 0 il flag "Mask" presente nella tabella Eventi Complessi. L'evento complesso responsabile dell'attivazione del programma di controllo verrà identificato nella tabella Programmi tramite il campo "PID". Il programma verrà concluso e l'Esecutore Programmi passerà a gestire il nuovo programma di controllo (quello specificato dall'istruzione Lancia).

18. Requisiti Hardware e Software

Le considerazioni effettuate per operare la scelta della piattaforma hardware e software sulla quale sviluppare il Sistema di Controllo sono state le seguenti:

- Contenere i costi dell'unità elaborativa a cui delegare il controllo ambientale.
- Permettere l'eventuale uso del calcolatore per altri programmi utilizzabili dall'utente.
- Disponibilità di interfacce per il collegamento con i dispositivi presenti nell'ambiente domestico.
- Predisposizione per il collegamento telematico ad un Centro Servizi.
- Disponibilità di strumenti di sviluppo software object-oriented, per modularità, portabilità e riusabilità dei componenti.
- Ambiente di lavoro standardizzato per ridurre il tempo di apprendimento dell'uso dello stesso da parte del personale tecnico.

Queste considerazioni hanno portato a scegliere come ambiente software all'interno del quale sviluppare il Sistema di Controllo il sistema *Microsoft Windows* ™, e come hardware un personal computer *MS-DOS* ™ compatibile, con una configurazione adatta a svolgere nel modo corretto il compito a lui affidato. A causa del fatto che Windows non è un ambiente portato alla risoluzione di problematiche in real-time, si è dovuto ovviare a questo problema in fase di progettazione del Sistema di Controllo, realizzandolo in modo tale da supplire a questa mancanza.

Per lo sviluppo software del Sistema di Controllo si è scelto il linguaggio C++ per la sua naturale predisposizione alla creazione di programmi con una metodologia orientata all'oggetto. Per la specifica piattaforma software scelta, si è deciso di utilizzare il compilatore *Microsoft Visual C++* ™, attualmente uno dei più avanzati e flessibili ambienti di sviluppo presenti sul mercato.

Per quanto riguarda i requisiti hardware, occorre suddividere le caratteristiche richieste dal calcolatore utilizzato per il Sistema di Controllo da quelle del calcolatore utilizzato dal personale tecnico per sviluppare i programmi di controllo ambientale. È chiaro che il calcolatore utilizzato per lo sviluppo dovrà necessariamente avere una diversa e più completa configurazione rispetto a quello utilizzato come supporto del Sistema di Controllo.

Le caratteristiche richieste per il calcolatore sul quale viene fatto funzionare il Sistema di Controllo sono essenzialmente legate alla complessità dell'ambiente da gestire, intesa come quantità di dispositivi collegati. Più questo numero di dispositivi è elevato, maggiore dovrà essere la potenza elaborativa e di I/O fornita dal calcolatore. Si può osservare nella Tabella 18.1 quali possono essere le configurazioni minima e consigliata per il calcolatore tramite il quale gestire l'ambiente. Si è inoltre deciso di non dotare il calcolatore utilizzato per il Sistema di Controllo di un lettore di floppy disk, mentre è presente un hard disk, perché necessario per il funzionamento di Windows. Una soluzione alternativa potrebbe essere l'utilizzo di una delle versioni di Windows in ROM, ma tale possibilità deve essere ancora analizzata in dettaglio.

Per quanto riguarda il calcolatore messo a disposizione del personale tecnico, le considerazioni da effettuare sono differenti. Dalla Tabella 18.2 si può osservare come le caratteristiche proposte siano le stesse di un sistema utilizzato per lo sviluppo software in ambiente Windows. Non sarà quindi necessario avere una configurazione esclusivamente dedicata allo sviluppo ed alla manutenzione delle procedure di controllo per l'ambiente. È inoltre preferibile che il tecnico sia dotato di un calcolatore portatile, in modo tale che possa effettuare il collegamento con il calcolatore che gestisce il Sistema di Controllo tramite una connessione via cavo. In tal modo si elimina la necessità di avere un lettore di floppy disk collegato al calcolatore utente.

Tabella 18.1: Configurazione hardware utente

	Configurazione minima	**Configurazione consigliata**
CPU	386sx-33	i486sx
Architettura	ISA	ISA + Vesa Local Bus
Memoria RAM (Mb)	2	4
Floppy disk (Mb)	Assente	Assente
Capacità Hard Disk (Mb)	40	120
Scheda video (r x c x colori)	Vga (640x480x16)	Svga (800x600x256)
Altro		Eventuale modem per telemanutenzione

Tabella 18.2: Configurazione hardware tecnico

	Configurazione minima	**Configurazione consigliata**
CPU	i486DX-33	i486DX-66
Architettura	ISA + Vesa Local Bus	ISA + Vesa Local Bus
Memoria Ram (Mb)	4	8
Floppy disk (Mb)	1.44	1.44
Capacità Hard Disk (Mb)	120	250
Scheda video (r x c x colori)	Svga (800x600x256)	Svga (1024x768x256)
Altro	stampante	modem per telemanutenzione stampante

19. Implementazione

A causa della complessità del progetto e della sua necessità di evolvere nel tempo si è deciso di utilizzare un approccio *object-oriented*. Si è quindi scelto il linguaggio *C++* perché è un linguaggio ibrido che unisce le funzionalità di un linguaggio orientato agli oggetti e le caratteristiche di un linguaggio strutturato tradizionale ed efficiente, il C. Il C++ offre al programmatore le capacità orientate agli oggetti senza però compromettere l'efficienza nelle fasi di esecuzione e di gestione della memoria del sistema. Questo aspetto è molto importante a causa delle caratteristiche di funzionamento real-time richieste per questo progetto.

Oltre all'approccio orientato agli oggetti utilizzato per la scrittura dei programmi costituenti il Sistema di Controllo ed i programmi accessori, si sono utilizzare delle classi di base rese disponibili dal compilatore *Microsoft Visual C++*, chiamate *Microsoft Foundation Class*. Queste permettono di utilizzare un approccio object-oriented anche per quanto riguarda l'interazione con l'ambiente *Windows*, e forniscono inoltre una serie di classi utili per la gestione delle strutture e dei tipi di dati normalmente utilizzati nei programmi.

19.1 Composizione del sistema

Come risulta chiaramente dalle specifiche progettuali, l'intero sistema è composto dalle seguenti componenti distinte:

- Il *Sistema di Controllo*, che è la parte che si occupa di controllare l'ambiente domestico e di eseguire i diversi compiti per i quali viene predisposto attraverso i programmi di controllo.
- L'*Ambiente di Configurazione e Programmazione della Casa*, che viene utilizzato dal personale qualificato per la realizzazione dei programmi di controllo e per la personalizzazione del comportamento del Sistema di Controllo, al fine di adattarlo alle specifiche esigenze dell'utente.
- Il *Modulo di Comunicazione*, che si occupa del collegamento con il Centro Servizi per la segnalazione di emergenze e allarmi. Questo programma svolge una funzione accessoria al Sistema di Controllo, anche se essa risulta di fondamentale importanza per ovvi motivi. Questo modulo comunica con il Sistema di Controllo tramite i meccanismi generali di comunicazione definiti nel progetto. In particolare questi sono realizzati tramite il meccanismo standard di Windows per la comunicazione tra processi.

Una precisazione è doverosa. Le immagini riportate in questo capitolo sono state "catturate" utilizzando l'ultima versione dell'ambiente Microsoft, ovvero *Windows 95*, ma le applicazioni sono state scritte utilizzando esclusivamente le risorse messe a disposizione dalla versione 3.1. Questo perché il compilatore disponibile non riconosce ancora questo nuovo sistema operativo e la versione di Win 95 utilizzata non è ancora la versione definitiva, bensì la *Beta Final*, ed è stata fornita dalla Microsoft quando il progetto era praticamente concluso. Si è potuto però verificare l'assoluta compatibilità tra l'applicazione sviluppata ed il nuovo ambiente.

19.2 Sistema di Controllo

Il *Sistema di Controllo* è la componente del sistema che si occupa di supervisionare l'ambiente domestico e di fornire all'utente particolari servizi atti a semplificargli la vita di tutti i giorni. Occorre però notare che l'utente non avrà mai interazioni dirette con esso, cioè non si accorgerà neppure che le funzioni basilari all'interno della sua abitazione sono gestite da un calcolatore. Quindi non è necessario che il programma possegga una particolare interfaccia utente, perché esso comunicherà con l'utente tramite gli appositi dispositivi presenti nell'ambiente. Si è comunque dotato il programma di una scarna interfaccia utente a scopo di

dimostrazione, per far vedere che esso è in funzione, e per poterlo attivare e disattivare facilmente in fase di dimostrazione e di test. Questa semplice interfaccia utente è visibile nella Figura 9.

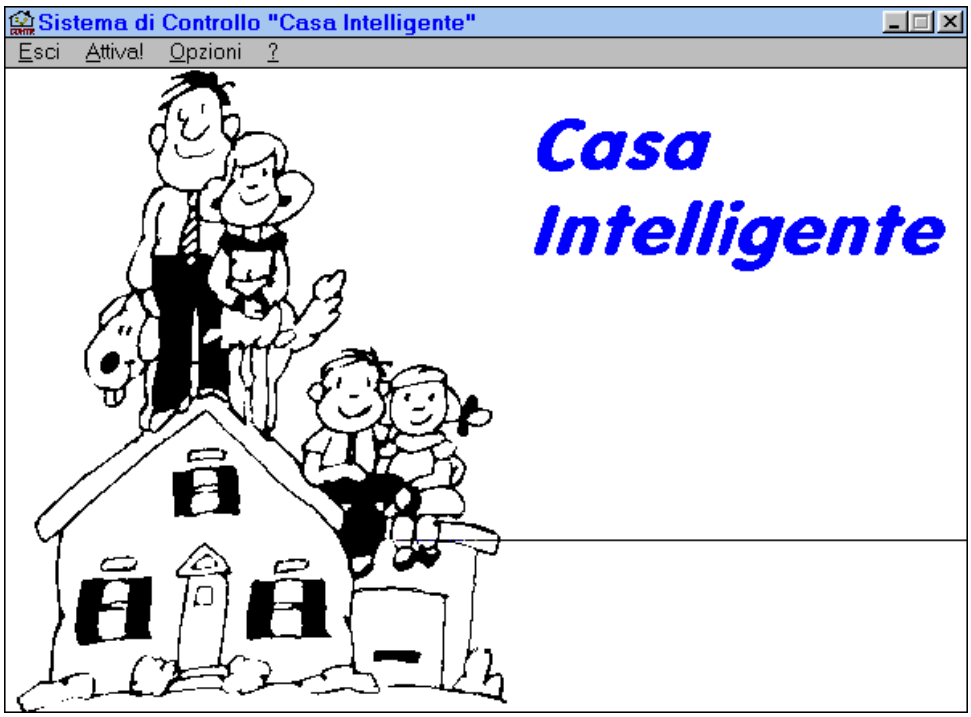


Figura 9: Interfaccia utente del Sistema di Controllo

Sono presenti i comandi *Esci*, utilizzato per chiudere l'applicazione, *Attiva!*, necessario per attivare e disattivare il Sistema di Controllo (quando il sistema è in funzione, questa voce di menu diventa *Disattiva!*) e *Opzioni*, che permette di visualizzare una linea di stato in fondo alla finestra e una toolbar sotto la linea occupata dal menu. Questa linea di stato viene utilizzata durante il funzionamento del Sistema di Controllo per visualizzare informazioni riguardanti i vari componenti attivi e indicare la risposta del sistema agli stimoli esterni. Il comando indicato con il *?* (punto interrogativo) visualizza le informazioni riguardanti la versione del programma attualmente in uso e alcuni dati dell'ambiente Windows, come illustrato in Figura 10.

Figura 10: Informazioni sul "Sistema di Controllo"

Quando l'applicazione viene minimizzata, tramite l'apposito comando Windows, il Sistema di Controllo continua naturalmente a funzionare, ma non appariranno più segnalazioni sullo schermo. In tal modo si ha la reale modalità di funzionamento trasparente all'utente descritta precedentemente.

19.2.1 Architettura interna

Per la realizzazione del Sistema di Controllo si è utilizzato come base la struttura standard fornita dal compilatore Visual C++. Questa struttura è costituita da una finestra base associata all'applicazione, alla quale vengono associate delle ulteriori sottofinestre per realizzare le altre componenti del programma necessarie ad ottenere la classica struttura windows-menu-dialogs tipica di un ambiente grafico.

Si è però visto poc'anzi che il Sistema di Controllo non necessità di interazione con l'utente, quindi questa struttura viene utilizzata esclusivamente come "contenitore" dell'oggetto che racchiude in se le funzionalità del controllo ambientale. Questo oggetto è definito essere un'istanza della classe *kernel*.

La classe kernel contiene al suo interno tutte le funzionalità riguardanti il funzionamento del Sistema di Controllo, tranne quello che riguarda la gestione dei dispositivi di input/output con l'ambiente. Questi ultimi sono gestiti da appositi oggetti, che si occupano di trasformare i dati provenienti e destinati all'ambiente in una forma comprensibile dal sistema.

La struttura interna della classe kernel ricalca fedelmente quanto è stato riportato nel progetto teorico, e per mantenere il più possibile questa struttura si è scelto di separare esplicitamente le parti nelle quali è stato necessario ricorrere alle funzionalità tipiche dell'ambiente Windows per risolvere particolari problemi. Queste parti riguardano esclusivamente la gestione delle scadenze temporali e dei timeout. Nell'implementazione dei moduli descritti nel progetto teorico si è fatto largo uso delle classi predefinite dal compilatore, ottenendo così una elevata chiarezza e comprensibilità, difficilmente ottenibile con un programma C "tradizionale"

Per quanto riguarda la gestione dei timer e dei timeout, si è fatto uso di un meccanismo basato sull'invio di messaggi a scadenze di tempo ben determinate. Windows fornisce la possibilità alle applicazioni di richiedere l'attivazione di timer ciclici, aventi la risoluzione minima di 1 millisecondo. La notifica della scadenza dei timer avviene tramite l'invio, alla finestra dell'applicazione che ne ha richiesto l'attivazione, di un messaggio WM_TIMER. Per questo motivo è stato necessario creare una finestra "invisibile" e associarla alla classe kernel.

Sono stati utilizzati due timer: il primo, identificato da IDT_CICLICO, viene attivato una volta al secondo e serve per tutto quanto concerne il controllo dei timer e dei timeout; il secondo, chiamato IDT_MINUTO, viene attivato una volta ogni minuto e viene utilizzato per controllare se occorre far partire qualche programma la cui esecuzione deve avvenire ad un particolare orario.

Un altro componente del Sistema di Controllo la cui struttura è fortemente collegata al funzionamento dell'ambiente Windows è la parte che si occupa del collegamento con il Centro Servizi. Questo collegamento non viene realizzato da una componente software interna al Sistema di Controllo, ma è ottenuto tramite il colloquio tra un oggetto apposito dichiarato all'interno della classe kernel ed un programma che viene eseguito contemporaneamente all'interno del sistema Windows. La comunicazione con questo programma viene realizzata mediante un meccanismo, chiamato *Dynamic Data Exchange (DDE)*, che permette di effettuare una comunicazione client-server tra programmi diversi in esecuzione concorrente all'interno del calcolatore. Una trattazione dettagliata di questo meccanismo viene riportata nel capitolo 23. L'aver separato queste componenti permette di sostituire in maniera del tutto trasparente i programmi che si occupano di stabilire la connessione con il Centro Servizi, rendendo possibile l'uso indifferente di qualsiasi tecnologia trasmissiva. È quindi possibili passare da una comunicazione tramite linea telefonica commutata ad una realizzata tramite ISDN senza dover minimamente alterare il Sistema di Controllo.

Per poter realizzare quanto detto, quando occorre contattare il Centro Servizi il Sistema di Controllo effettua una richiesta per un server capace di offrire un servizio caratterizzato da *App* = *TLC* e *Topic* = *ALLARME*. Questo server dovrà essere inoltre in grado di comprendere

gli *Item* definiti in fase di configurazione del sistema. Essi avranno associato l'identificativo del sensore che ha provocato la richiesta di collegamento ed il valore assunto dal sensore stesso.

Tutta la comunicazione DDE viene realizzata da un'apposita classe chiamata *ClientDDE*, i cui metodi permettono al programma di stabilire la connessione, inviare i dati e terminare il colloquio.

Si è visto precedentemente che la gestione delle periferiche addette all'interazione con l'utente e l'ambiente domestico sono gestite da un apposito modulo. Questo è costituito da una classe i cui metodi devono essere in grado di aprire e chiudere un canale di comunicazione, leggere e scrivere dei dati su di esso e, cosa molto importante, segnalare la presenza di nuovi dati al gestore driver del Sistema di Controllo. Questa classe ha a disposizione tutti gli strumenti software disponibili all'interno dell'ambiente Windows, quindi anche i timer per eventuali periferiche che richiedano un polling ad intervalli regolari.

19.2.2 Metodi della classe kernel

Qui di seguito viene riportata una breve descrizione dei metodi pubblici e privati della classe kernel, insieme con i parametri da essi richiesti.

Metodi pubblici:

- *int InitSystem();* - Si occupa di inizializzare tutte le strutture dati del Sistema di Controllo.
- *void InitTimerSistema();* - Inizializza la struttura dei messaggi di Windows per avere un evento timer al secondo ed uno al minuto.
- *void KillTimerSistema();*- Elimina i timer Windows utilizzati dal Sistema di Controllo.
- *void Driver(int msg, UINT sens = 0, int val = 0);*- Gestore dei drivers delle periferiche.

Metodi privati:

- *void FiltroEventi();* - Richiamato quando avviene una modifica nello stato di un sensore, dopo che è stata aggiornata la tabella stato sensori.
- *void Esecutore();* - È il metodo che si occupa dell'esecuzione dei programmi. Viene richiamato quando ci sono eventi da servire in coda.
- *void EliminaEvds(UINT sensore, int valore);* - Elimina dalla tabella eventi da servire l'entry identificata dal valore e dal sensore specificati.
- *int VSens(UINT sensore);*- Restituisce lo stato del sensore dato.
- *BOOL ZSens(UINT sensore);*- Restituisce lo stato del flag variazione del sensore dato.
- *void EliminaTimeout(TEV ev);* - Elimina dalla tabella timeout l'entry corrispondente all'evento specificato.
- *void Attuatore(BYTE dispositivo, int stato);*- Invia un comando all'attuatore specificato.
- void Segnala_Utente(int messaggio); - Invia all'utente il messaggio specificato, utilizzando la tabella messaggi per stabilire come deve essere effettuato.
- *void EliminaTimer(TEV ev);* - Usato quando un evento collegato ad un timer si verifica. Occorre quindi annullare il timer. Usato anche per eliminare un timer ciclico.
- *void NuovoTimer(TEV ev, time_t scadenza, BOOL ciclico, time_t intervallo = 0);* - Si occupa dell'inserimento nelle apposite strutture di nuovi timer ciclici e non. L'ultimo argomento e' opzionale e riguarda esclusivamente i timer ciclici.
- *BOOL Test_Timeout();* - Controlla se qualche timeout è scaduto.
- *void EliminaEC(TEV ev);*- Elimina un evento complesso dalla tabella omonima.
- *BOOL Test_Timer();* - Controlla se qualche timer è scaduto

- *int PosEC(TEV ev);* - Fornisce la posizione di un evento all'interno della tabella eventi complessi. Restituisce -1 se non lo trova.
- *int PosProg(TEV ev);* - Fornisce la posizione all'interno della tabella programmi dei dati relativi al programma collegato all'evento specificato. Restituisce -1 se non lo trova.
- *void EliminaPgm(TEV ev);* - Elimina un programma dalla tabella omonima, ma non elimina fisicamente il codice dalla tabella memoria.
- *void CnfSensore(int sensore, int parametro, int valore);* Configura un sensore.
- *void InCoda(TEV ev, int pri);*- Mette in coda gli eventi da servire.
- *void Pid2Evento(UINT pid, TEV &ev);* - Restituisce l'evento responsabile dell'attivazione iniziale del programma.
- BOOL CtrlStato(int cnf); - Confronta la configurazione attuale del sistema con quella di riferimento specificata. Restituisce una segnalazione di errore se viene riscontrata una differenza.
- *void Errore(int mess);* - Inserisce nel file di log la segnalazione dell'avvenuto errore nel funzionamento del Sistema di Controllo.
- *void Status(CString testo);*- Scrive il messaggio specificato nella status bar.
- *void Teleallarme(int tipo, int sensore);* - Invia il messaggio DDE al client di telecomunicazione.

19.3 Ambiente di Configurazione e Programmazione della Casa

L'*Ambiente di Configurazione e Programmazione della Casa* è la componente del sistema mediante la quale il personale tecnico può configurare la risposta del Sistema di Controllo alle necessità dell'utente. Questo verrà ottenuto preparando degli appositi programmi e collegandoli ad eventi che si possono verificare all'interno dell'ambiente domestico. L'Ambiente di Configurazione e Programmazione della Casa permette inoltre di compilare tutte le strutture dati necessarie al corretto funzionamento del Sistema di Controllo, quali le tabelle contenenti l'associazione tra programmi ed eventi, le scadenze orarie alle quali occorre attivare particolari procedure, le modalità con le quali l'utente interagisce con il sistema e viceversa.

Dato che L'Ambiente di Configurazione e Programmazione della Casa deve essere utilizzato per le funzioni appena viste, esso ha un'elevata interazione con il personale tecnico che lo deve utilizzare, e quindi deve fornire un'interfaccia utente ed un comportamento conforme a quanto consigliato dalla Microsoft per le applicazioni scritte in ambiente Windows[10].

Essendo essenziale la fase di scrittura dei programmi di controllo, viene messo a disposizione del personale tecnico un editor integrato all'interno dell'Ambiente di Configurazione e Programmazione della Casa, come si può osservare in Figura 11.

[10] *Le specifiche sono riportate nel volume "The Windows™ Interface - An Application Design Guide. For the Microsoft® Windows™ Operating System" - Microsoft Corporation 1992*

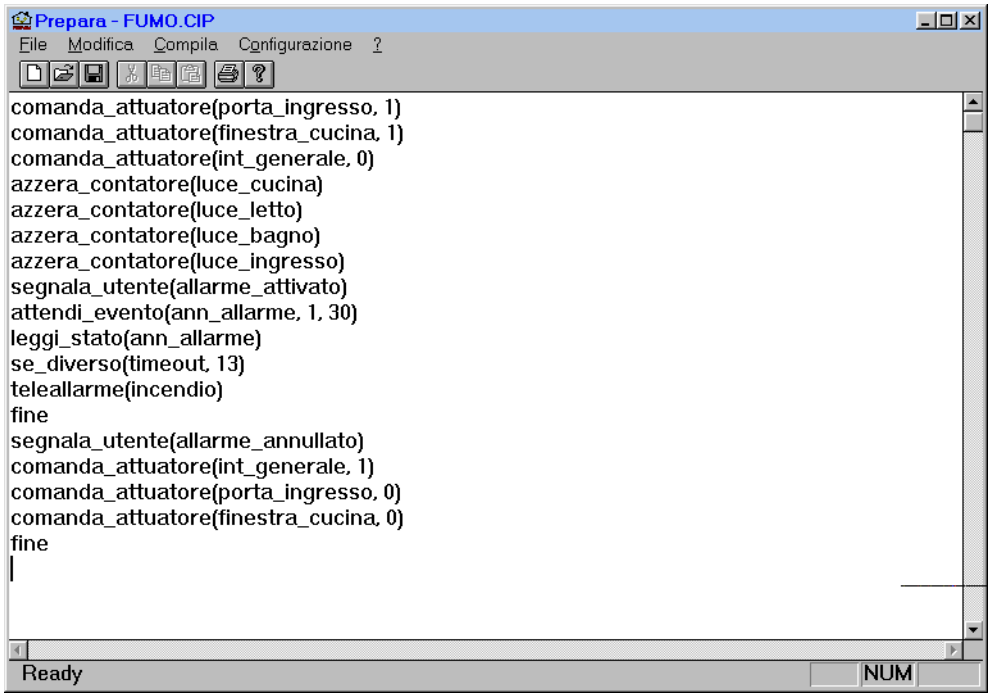


Figura 11: Interfaccia utente dell'Ambiente di Configurazione e Programmazione della Casa

Come si può osservare sono presenti alcune voci di menu, una toolbar che velocizza l'uso dei comandi più frequenti ed una status bar che riporta segnalazioni sul funzionamento del programma e lo stato di alcune funzioni della tastiera.

Verranno ora analizzate nel dettaglio le diverse voci di menu e le funzioni ad esse associate.

19.3.1 Menu File

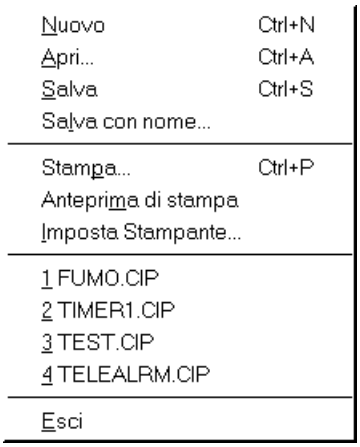


Figura 12: Ambiente di Configurazione e Programmazione della Casa - menu File

Il menu *File* è una caratteristica standardizzata delle applicazioni Windows. Esso riporta i seguenti comandi:

- *Nuovo* - Cancella il programma attualmente caricato e si predispone alla creazione di uno nuovo. Se il programma attualmente presente non è stato salvato, viene richiesto se lo si vuole salvare.

- *Apri...* - Visualizza una dialog box standard per effettuare il caricamento di un programma precedentemente memorizzato sul disco.
- *Salva* - Salva su disco il programma attualmente caricato. Se esso non ha ancora un nome associato, viene richiesto.
- *Salva con nome...* - Permette di salvare il programma attualmente caricato specificando un nome diverso da quello attuale.
- *Stampa...* - Produce una copia su stampante del listato del programma attualmente in memoria.
- *Anteprima di stampa*- Visualizza come avverrà la stampa del listato del programma.
- *Imposta Stampante...* - Permette di selezionare la stampante da utilizzare e di impostarne i parametri necessari al suo corretto funzionamento
- *Esci* - Esce dall'Ambiente di Configurazione e Programmazione della Casa. Se il programma attualmente caricato non è ancora stato salvato, viene richiesto se lo si vuole salvare.

Sono anche presenti i nomi degli ultimi file che sono stati utilizzati all'interno del programma, per poterli richiamare semplicemente selezionandoli.

19.3.2 Menu Modifica

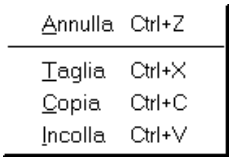


Figura 13: Ambiente di Configurazione e Programmazione della Casa - menu Modifica

Anche il menu *File* è una caratteristica standardizzata delle applicazioni Windows. Esso riporta i seguenti comandi:

- *Annulla* - Permette di annullare l'ultimo comando dato.
- *Taglia* - Elimina la porzione di testo attualmente selezionata e la copia nella Clipboard[11].
- *Copia* - Copia la porzione di testo attualmente selezionata nella Clipboard.
- *Incolla* - Inserisce alla posizione corrente del cursore i dati presenti nella Clipboard.

Questi comandi servono esclusivamente in fase di creazione o di modifica dei programmi di controllo.

19.3.3 Menu Compila

L'unica voce di questo menu si occupa di attivare una fase molto importante della scrittura dei programmi per il Sistema di Controllo: la loro verifica sintattica e la successiva trasformazione nel formato oggetto, che è l'unico comprensibile dal sistema. In questa fase vengono anche convertiti gli identificatori simbolici dei sensori e degli attuatori negli equivalenti numerici assegnati ai dispositivi presenti nell'ambiente.

[11] *La Clipboard è una struttura di Windows che permette l'interscambio di dati tra diverse applicazioni, utilizzando il meccanismo "Taglia e Incolla".*

19.3.4 Menu Configurazione

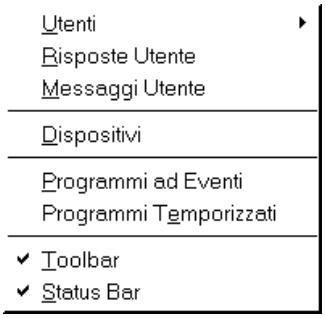


Figura 14: Ambiente di Configurazione e Programmazione della Casa - menu Configurazione

Tramite questo menu è possibile associare ai dispositivi presenti nell'ambiente un identificatore simbolico, decidere come e quando devono essere attivati i programmi di controllo, in che forma l'utente ed il sistema devono interagire fra loro e quali sono i permessi di accesso degli utilizzatori dell'Ambiente di Configurazione e Programmazione della Casa.

I Comandi riportati nel menu *Configurazione*sono i seguenti:

- *Utenti* - Questa voce è in realtà il titolo del seguente sottomenu:

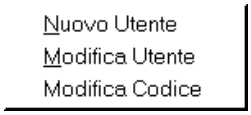


Figura 15: Ambiente di Configurazione e Programmazione della Casa - menu Configurazione/Utenti

Le voci presenti in questo sottomenu hanno un significato ovvio.

- *Risposte Utente* - Permette di stabilire tramite quali modalità l'utente può dare dei comandi al Sistema di Controllo.
- *Messaggi Utente* - Definisce l'associazione tra un tipo di messaggio da inviare all'utente e i dispositivi da utilizzare per effettuare la segnalazione.
- *Dispositivi* - Realizza l'associazione tra gli identificativi numerici dei sensori e degli attuatori e i nomi simbolici utilizzati all'interno dei programmi di controllo.
- *Programmi ad Eventi* - Tramite questa opzione si stabilisce quale evento è responsabile dell'attivazione di uno specifico programma di controllo.
- *Programmi Temporizzati* - Definisce l'orario al quale devono essere attivati dei programmi di controllo prestabiliti.
- *Toolbar* - Attiva o disattiva la toolbar.
- *Status Bar* - Attiva o disattiva la status bar.

19.3.5 Menu ?

Come si è già visto per il Sistema di Controllo, la voce di menu indicata con iP (punto interrogativo) visualizza le informazioni riguardanti la versione del programma attualmente in uso e alcuni dati dell'ambiente Windows, come illustrato in Figura 16.

Figura 16: Informazioni su "Ambiente di Configurazione e Programmazione della Casa "

19.3.6 Dialog "Definizione Identificativi Dispositivi"

Per poter realizzare l'associazione tra i codici numerici che identificano i sensori e gli attuatori e degli identificatori simbolici da utilizzare nei programmi e nelle altre fasi di configurazione, si utilizza la seguente dialog box:

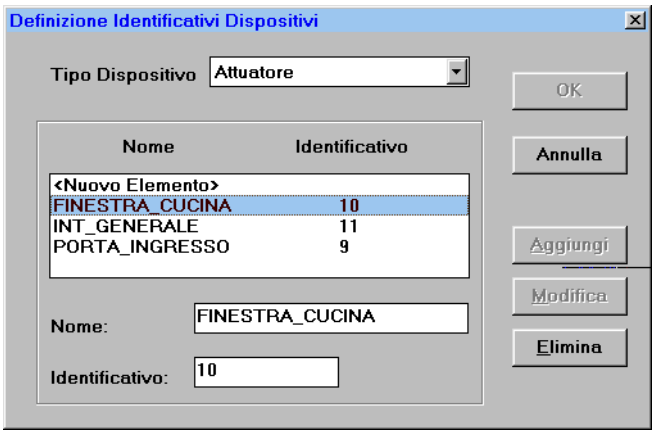


Figura 17: Dialog box "Definizione Identificativi Dispositivi - Attuatori"

Questa stessa dialog box viene utilizzata anche per i sensori, scegliendo l'apposita voce nella combobox:

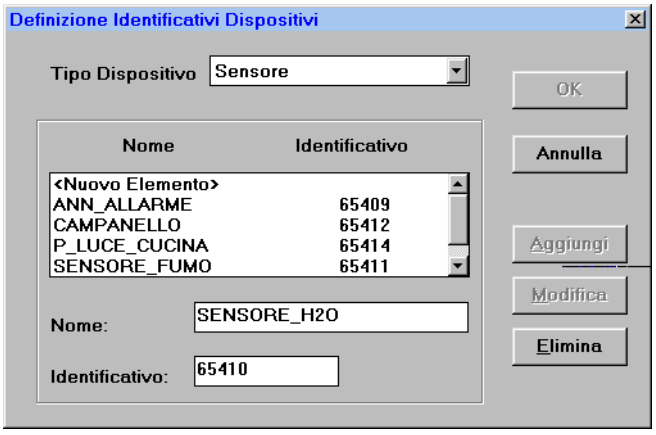


Figura 18: Dialog box "Definizione Identificativi Dispositivi - Sensori"

Selezionando la voce <*Nuovo Elemento*> è possibile inserire una nuova coppia di dati Nome/Identificativo, mentre scegliendo un dato già presente si ha la possibilità di modificarlo.

19.3.7 Dialog "Configurazione Avvio Programmi"

La struttura del Sistema di Controllo è imperniata sul concetto dell'attivazione dei programmi tramite eventi. Questa dialog permette di preparare la struttura dati necessaria per realizzare correttamente questa attivazione.

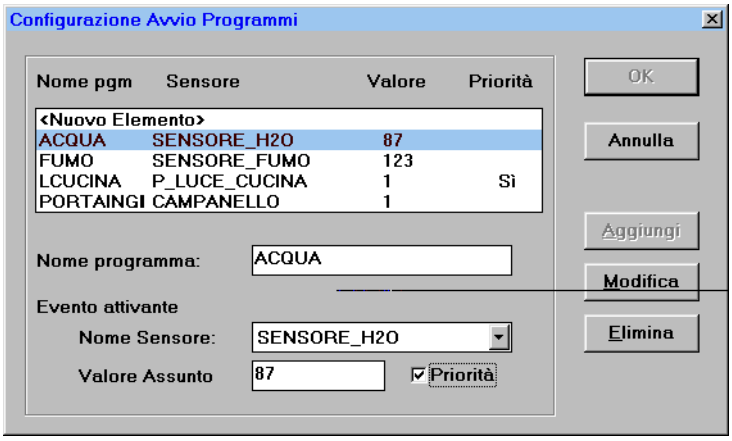


Figura 19: Dialog box "Configurazione Avvio Programmi"

Il nome di un programma di controllo avrà associato un evento attivante, costituito dal nome di un sensore, il valore che esso deve assumere ed eventualmente un indicazione per premettere che il Sistema di Controllo esegua questo programma prima di altri meno urgenti.

Anche in questa dialog è possibile inserire nuovi elementi e modificare quelli preesistenti, con le stesse modalità viste precedentemente.

19.3.8 Dialog "Configurazione avvio programmi su scadenza temporale"

È spesso necessario attivare dei particolari programmi di controllo ad un orario specifico. La creazione della struttura dati utilizzata dal Sistema di Controllo viene effettuata dalla seguente dialog:

Figura 20: Dialog box "Configurazione avvio programmi su scadenza temporale"

Al nome di un programma di controllo sarà associato un orario di attivazione, suddiviso in ore e minuti, al verificarsi del quale il programma sarà attivato.

Come nelle dialog precedenti, anche qui è possibile inserire nuovi elementi e modificare quelli preesistenti, con le stesse modalità viste prima.

19.3.9 Permessi di accesso all'Ambiente di Configurazione e Programmazione della Casa

A causa della delicatezza di alcune delle operazioni riguardanti la configurazione e la programmazione del Sistema di Controllo dall'interno dell'Ambiente di Configurazione e Programmazione della Casa, è permesso l'accesso alle diverse fasi di configurazione solo alle persone dotate di un apposito sistema di riconoscimento, cioè l'insieme di un *identificativo utente* e di una *password*.

Questi permessi possono essere creati e modificati tramite le dialog riportate qui di seguito.

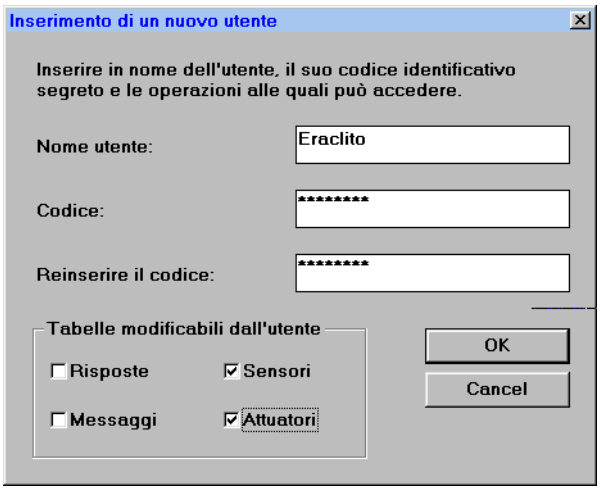


Figura 21: Dialog Box "Inserimento di un nuovo utente"

Questa dialog si occupa dell'inserimento di un nuovo utente, del suo codice riservato e dell'indicazione delle strutture dati alle quali può accedere.

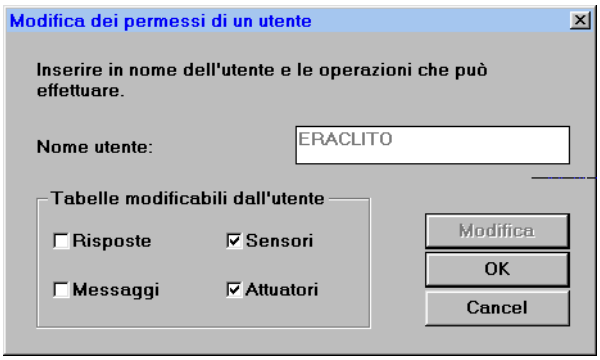


Figura 22: Dialog Box "Modifica dei permessi di un utente"

Tramite questa dialog si possono modificare i permessi di accesso di un utente già esistente.

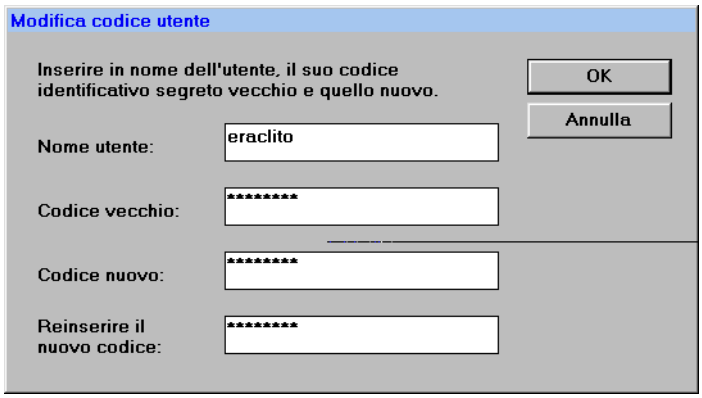


Figura 23: Dialog Box "Modifica codice utente"

Questa dialog permette di sostituire il codice riservato dell'utente con uno nuovo.

19.3.10 Risposte e Segnalazioni utente

Per poter personalizzare le risposte che si attende il Sistema di Controllo dall'utente e scegliere come devono essere inviate le segnalazioni all'utente, si fa ricorso alle dialog box riportate in Figura 24 e Figura 25, che permettono di modificare le strutture dati utilizzate dal sistema.

Il tipo di risposta o di segnalazione sarà identificata tramite un identificatore simbolico utilizzabile in fase di scrittura dei programmi, e da un insieme di coppie formate da un identificatore di dispositivo e da un codice che deve essere o inviato o restituito dal dispositivo stesso affinché venga effettuata la segnalazione o venga letta la risposta.

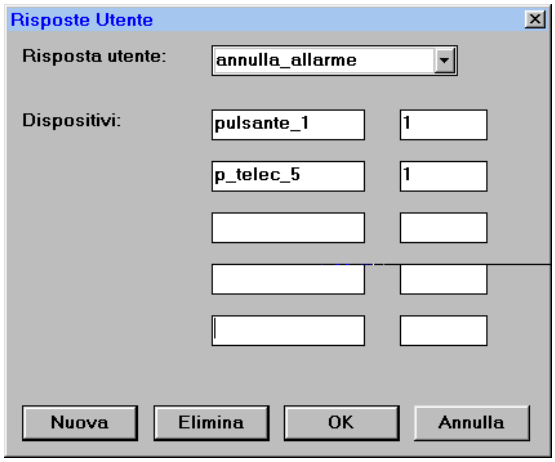


Figura 24: Dialog box "Risposte Utente"

Figura 25: Dialog box "Segnalazioni Utente"

19.3.11 Architettura interna

L'intero Ambiente di Configurazione e Programmazione della Casa è costruito intorno ad un oggetto derivato dalla classe *CEditView*, che mette a disposizione le caratteristiche degli edit control di Windows fornendo inoltre le funzionalità addizionali di stampa e di taglia-incolla. In questo modo si ottiene un semplice ambiente di editing, più che sufficiente per scrivere e modificare i programmi di controllo.

Tutte le dialog box viste in precedenza sono oggetti derivati dalla classe *CDialog*, che è la classe base dalla quale devono essere derivate le dialog, siano esse modali o non modali. A questi oggetti occorre poi aggiungere delle variabili interne per poter memorizzare i dati inseriti nella dialog dall'utente o per poter visualizzare delle informazioni. Per ognuna di queste variabili occorre poi stabilire il tipo ed l'intervallo dei valori ammessi. Per poter poi far si che il comportamento dei vari oggetti contenuti all'interno della dialog sia quello voluto, bisogna intercettare i messaggi generati dalle diverse modificazioni che avvengono in questi oggetti, e configurarne di conseguenza le proprietà. Per esempio, sarà necessario controllare che il pulsante Modifica non diventi attivo finché non sono stati effettivamente modificati dei dati.

I metodi sugli oggetti costituenti le dialog si occuperanno inoltre di generare gli appositi file di configurazione, necessari per il funzionamento del Sistema di Controllo. La struttura di questi file viene descritta in dettaglio nel capitolo 22.

L'Ambiente di Configurazione e Programmazione della Casa ha la sua parte più importante nella fase di compilazione dei programmi di controllo, al fine di generare un programma oggetto comprensibile ed eseguibile dal Sistema di Controllo. Questa funzione viene svolta dai metodi della classe *CCompila*, che si occupano appunto di analizzare linea per linea il programma sorgente presente all'interno dell'oggetto di tipo CEditView ed di generare un file avente estensione *.CIO*.

Qui di seguito vengono descritti i metodi definiti sulla classe CCompila, insieme con i loro parametri.

Metodi pubblici::

- *BOOL Parser();* - È il metodo che si occupa della compilazione vera e propria. Dal suo interno vengono richiamati tutti gli altri metodi privati, che hanno compiti più specifici.

Metodi privati:

- *int FindInstr(CString instr);* - Verifica se il token passato come parametro è tra quelli riconosciuti. Se non lo trova restituisce -1 come segnalazione di errore.

- *CString GetTok(int &p, CString linea);* - Estrae un token alla volta dalla stringa e lo restituisce, modificando il puntatore alla posizione corrente della stringa stessa.
- *int ParScad(CString s);* - Analizza un parametro di tipo scadenza temporale e ne restituisce il valore sotto forma di un int.
- *int ParSS(CString s);* - Analizza un parametro di tipo stato sensore e ne restituisce il valore sotto forma di int.
- *void Errore(int codice, CString mess);* - Segnala che si è verificato un particolare errore con eventualmente associata una stringa contenente una descrizione dettagliata dell'errore verificatosi.
- *int ParSE(CString s);* - Analizza il parametro contenente l'istruzione di destinazione delle istruzioni di confronto SE_*.
- *int TA2Par(CString Allarme);* - Converte il tipo di allarme da inviare al Centro Servizi in un parametro numerico di tipo int.
- *int Att2Par(CString Attuatore);* - Converte il nome simbolico dell'attuatore nell'equivalente parametro numerico.
- *int Sen2Par(CString Sensore);* - Converte il nome simbolico del sensore nell'equivalente parametro numerico.

A causa della voluta semplicità del linguaggio con il quale vengono scritti i programmi di controllo, anche la fase di compilazione risulta molto semplificata ed è possibile effettuarla in un unico passaggio.

Dalla descrizione dei metodi privati della classe CCompila si può anche notare come l'associazione tra i nomi simbolici dei sensori e degli attuatori ed i codici numerici, quelli effettivamente riconosciuti dai dispositivi presenti nell'ambiente, venga effettuata in fase di compilazione.

19.4 Modulo di comunicazione con il Centro Servizi

Come si è già ampiamente visto nei capitoli precedenti, per ottenere la completa integrazione del Sistema di Controllo con le infrastrutture sociali occorre avere la possibilità di un collegamento telematico tra l'ambiente, e per estensione l'utente, e un Centro Servizi.

Questo collegamento si è visto che non viene realizzato da una componente software interna al Sistema di Controllo, ma è ottenuto tramite il colloquio con un programma che viene eseguito contemporaneamente ad esso all'interno del sistema Windows, mediante l'apposito meccanismo di Dynamic Data Exchange.

L'aver separato queste componenti permette di sostituire in maniera del tutto trasparente i programmi che si occupano di stabilire la connessione con il Centro Servizi, rendendo possibile l'utilizzo in maniera trasparente di qualsiasi tecnologia trasmissiva. È quindi possibili passare, per esempio, da una comunicazione tramite linea telefonica commutata ad una realizzata tramite ISDN senza dover minimamente alterare il funzionamento del Sistema di Controllo, addirittura senza doverlo prima fermare e poi farlo ripartire.

È stata realizzata un'apposita classe, chiamata*ServerDDE*, per gestire tutta la parte di comunicazione fra il Sistema di Controllo e il Modulo di Comunicazione, svincolandola completamente dalla parte riguardante le modalità di trasferimento dei dati tramite la tecnologia trasmissiva scelta. Una tale classe può eventualmente essere resa disponibile ai produttori di particolari apparecchiature per trasmissione di dati, al fine di poter ottenere la scelta più ampia possibile di soluzioni per le diverse esigenze di collegamento tra il Sistema di Controllo ed il Centro Servizi.

20. Conclusioni

Questa tesi ha voluto dimostrare come sia possibile realizzare un sistema di controllo configurabile e in grado di utilizzare le componenti esistenti sul mercato, senza dover sacrificare funzionalità, ma anzi fornendo una maggiore flessibilità. L'innovativo approccio utilizzato ha portato a progettare una struttura completamente svincolata dall'hardware disponibile e di conseguenza completamente adattabile alle specifiche esigenze dell'utente, contrariamente a quanto succede spesso nei sistemi commerciali, dove sono le esigenze dell'utente che si devono adattare alle funzionalità messe a disposizione dal sistema.

La "Casa Intelligente" può creare delle nuove opportunità per molte persone anziane, disabili o lungodegenti, consentendo loro una maggiore indipendenza e sicurezza all'interno della propria abitazione, e può contribuire a migliorare la qualità e l'efficienza dei servizi sociali di assistenza. Tuttavia, la "Casa Intelligente" deve essere accettata come fattore di stimolo, indipendenza e aiuto quotidiano e non deve diventare un fattore di inibizione, frustrazione o isolamento. A tal fine occorre rispettare alcuni requisiti di base, ampiamente esposti nei capitoli precedenti, per tenere conto delle differenti e variabili esigenze (fisiche e psicologiche) degli utenti. Occorre inoltre che siano pianificati ed attivati tutti quei servizi di supporto che possono realmente rendere possibile una vita indipendente e sicura della persona, sia essa anziana, disabile o lungodegente, quali il tele-soccorso, la tele-assistenza, il tele-allarme, ecc. Gli utenti devono in ogni caso essere continuamente seguiti nell'evolversi della loro patologia da personale esperto, che sia in grado di consigliare le necessarie estensioni, modifiche e adattamenti alla struttura della "Casa Intelligente" e ne garantisca la continua funzionalità.

Esistono al momento differenti soluzioni e dispositivi, disponibili sul mercato italiano ed estero, per l'automazione domestica, il tele-soccorso e la tele-assistenza. Sono stati inoltre realizzati a livello europeo vari progetti di ricerca per la realizzazione di "Case Intelligenti" con funzionalità avanzate. Tuttavia, questi ultimi spesso si basano su soluzioni prototipali e su componenti difficilmente reperibili sul mercato italiano e di costo elevato. Viceversa, le soluzioni commerciali esistenti non sono in grado di adattarsi in modo flessibile alle differenti esigenze degli utenti. Inoltre, soluzioni commerciali di diversi costruttori sono spesso "chiuse" e non sono in grado di integrarsi ed interagire fra di loro.

La scarsa diffusione delle esperienze pilota di "Casa Intelligente", soprattutto per quanto riguarda gli anziani, è spesso dovuta alla carenza di servizi di supporto adeguati (progettazione, addestramento, installazione, manutenzione e assistenza continua, gestione dei tele-servizi, ecc.). Le esperienze di "Casa Intelligente" devono essere accuratamente pianificate tenendo conto di tutti gli aspetti coinvolti e di tutte le infrastrutture di supporto necessarie. La "Casa Intelligente" è certamente uno strumento per rendere più efficiente la spesa sociale a favore delle categorie sopracitate, ma non è certo un mezzo per eliminare la necessità dei servizi sociali. La tecnologia può solo essere un supporto e deve comunque favorire quei contatti personali ed umani che sono essenziali e motivanti per qualunque persona, a maggior ragione per gli anziani ed i disabili.

Il Sistema di Controllo, nelle varie fasi del suo sviluppo, è stato presentato in alcuni congressi e manifestazioni specializzate, suscitando un notevole interesse da parte degli addetti ai lavori, a causa nella sua struttura innovativa e completamente programmabile. Durante la sua realizzazione vi è anche stato un continuo scambio di idee con psicologi ed operatori nel settore dell'assistenza, per poter comprendere a fondo le vere esigenze delle categorie di utenti per i quali questo sistema è stato concepito.

Nei paragrafi successivi sono riportate le occasioni nelle quali il Sistema di Controllo è stato presentato, insieme con una breve descrizione del dimostratore utilizzato.

20.1 Le telecomunicazioni per la qualità della vita

Si è trattato del primo workshop internazionale promosso da Telecom Italia sul ruolo delle telecomunicazioni telefoniche connesse ai teleservizi, inclusi quelli telematici, per promuovere la qualità della vita della gente, partendo anzitutto dai portatori di handicap. Si è svolto a Roma il 30 e 31 marzo 1995.

Per questa dimostrazione si è scelto di realizzare un "simulatore di ambiente domestico", raffigurato nella Figura 26, ovvero di utilizzare un personal computer come fosse l'ambiente controllato dal Sistema di Controllo. Questo a causa dell'impossibilità di realizzare per l'occasione un vero ambiente il cui controllo fosse realizzato dal sistema.

Si sono quindi utilizzati due notebook collegati via RS232, dei quali quello contenete il simulatore è posto a completa disposizione del pubblico. Tramite questo simulatore è possibile verificare il comportamento del sistema a seguito di particolari eventi che si verificano nell'ambiente. A scopo dimostrativo sono stati inseriti pulsanti per accendere e spegnere le luci, il campanello della porta, il pulsante per richiedere aiuto (tele-soccorso) e la possibilità di simulare gli allarmi di presenza di acqua sui pavimenti e di fumo nell'ambiente. In base a questi eventi vengono prese le opportune contromisure tramite attuatori (simulati) presenti nell'ambiente. Questi permettono di aprire e chiudere porte e finestre, di interrompere le forniture di luce e acqua e di chiamare il Centro Servizi. È inoltre presente la gestione completa del sistema di riscaldamento e climatizzazione, che entra automaticamente il funzione se la temperatura ambientale scende sotto o sale sopra due valori di soglia. Tramite un semplice cursore è possibile effettuare una variazione della temperatura rilevata dai sensori e osservare come il sistema provveda a riportarla al più presto possibile al valore ottimale, previsto in 20°. Per aumentare il coinvolgimento dello sperimentatore, sono state utilizzate semplici capacità multimediali, quali suoni e animazioni.

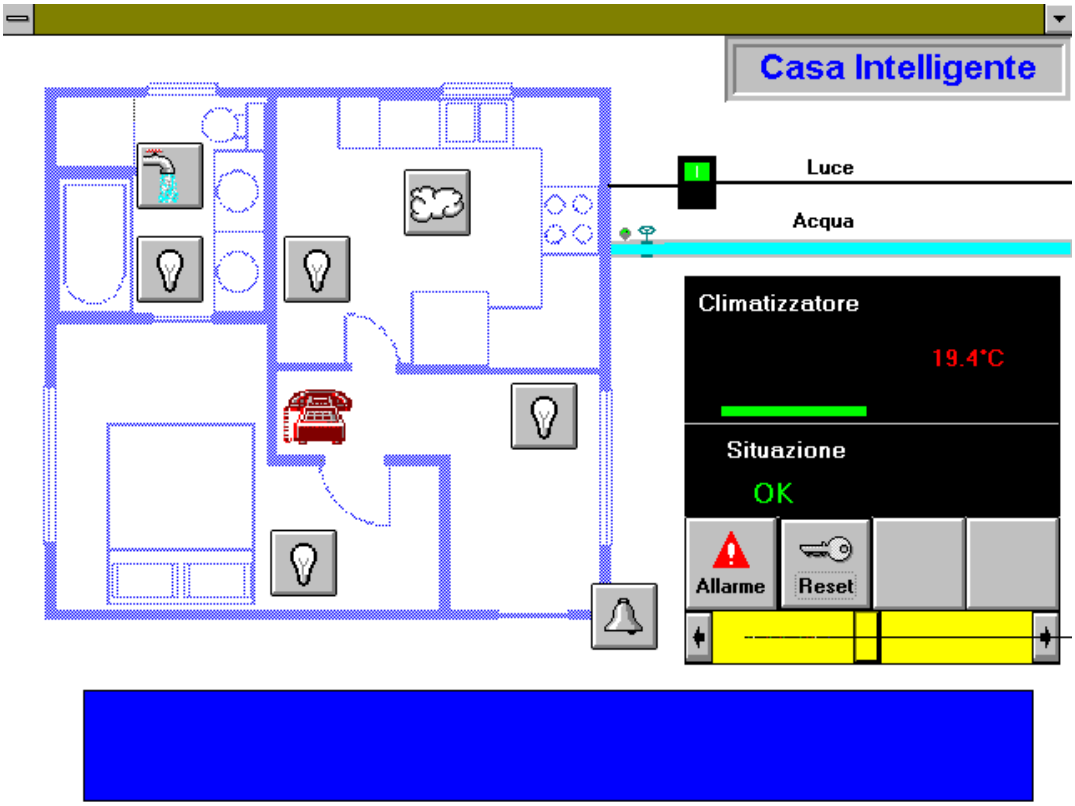


Figura 26: Dimostratore workshop "Le telecomunicazioni per la qualità della vita"

20.2 Gerontechnology

Si è trattato di un convegno riguardante gli ausili tecnologici per la terza età. All'interno di questo convegno erano presenti alcune sessioni parallele, una delle quali riguardante le tecnologie per "Case Intelligenti". Si è svolto a Firenze dal 30 Aprile al 3 Maggio 1995.

Questo convegno non prevedeva la possibilità di realizzare una dimostrazione pratica, quindi è stata effettuata una relazione descrittiva sul Sistema di Controllo realizzata dall'Ing. Scarpi, quale rappresentante dell'unità di Telemedicina e Telematica Sociale dello CSELT.

20.3 ABIL EXPO '95

Si tratta di una manifestazione che intende essere un'antologia di ausili, ricerche ed idee per vivere l'handicap. Si è svolta alla Fiera di Verona, dal 9 al 12 Giugno 1995.

Questa è sicuramente la dimostrazione più impegnativa realizzata, perché prevede la realizzazione di un vero appartamento, completamente controllato dal Sistema di Controllo, e di un Centro Servizi, dialogante con l'ambiente tramite un normale collegamento telefonico.

Per questa occasione è stato realizzato appositamente un dispositivo hardware non dotato di intelligenza propria, per potersi interfacciare con l'ambiente. Tramite questo dispositivo sono a disposizione 14 input per i sensori e 7 output di potenza per gli attuatori, sufficienti per il controllo dell'ambiente messo a disposizione.

Si è deciso di realizzare il controllo di un alloggio per una persona sola con problemi motori e/o di età avanzata. L'alloggio si compone di un ingresso che accede direttamente al soggiorno attrezzato con angolo cucina, una camera da letto ed il bagno.

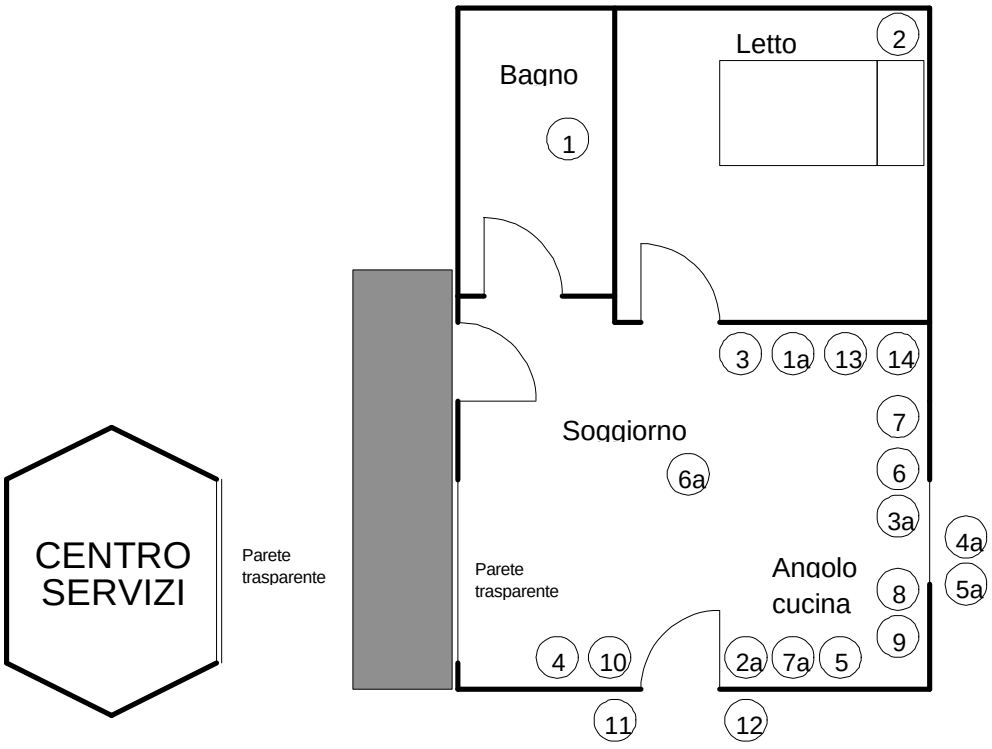


Figura 27: Dimostratore Abil Expo '95

Nel bagno e nella camera da letto sono disponibili pulsanti, a muro o sul comodino, per la richiesta di soccorso, con l'ulteriore possibilità di azionamento tramite un telecomando.

Nel soggiorno è presente un rilevatore ad infrarossi (IR) per rilevare la presenza delle persone nell'ambiente (siano esse il proprietario o eventuali intrusi).

Alcuni serramenti sono motorizzati (porta d'ingresso, finestra, tapparella) con comando a muro o tramite telecomando.

Una delle funzioni di "Casa Intelligente" è il controllo ambientale (allarmi di fuga gas, allarmi incendio, ecc.). Per simulare eventi anomali e la conseguente generazione di allarmi sono stati installati accanto ai sensori un tasto di per provocare un allarme forzato. Premendo tale tasto il sistema avrà la stessa reazione provocata dall'allarme vero (chiamata di soccorso al Centro Servizi e azioni volte a garantire la sicurezza dell'occupante dell'alloggio).

Il sistema di controllo ambientale si compone di un personal computer portatile posto in una posizione ben visibile dell'alloggio e da un box elettrico di controllo, connesso al p.c. mediante collegamento RS232, situato in una posizione non visibile dal visitatore.

Nelle vicinanze del p.c. è posta una presa telefonica per la connessione con il Centro Servizi, che ha anch'esso una linea telefonica e ospiterà un p.c. ed il modem di ricezione delle chiamate di soccorso.

Per meglio far comprendere quanto accade all'interno dell'ambiente domestico e nel Centro Servizi in caso di chiamata di allarme, sono presenti pareti trasparenti sia nell'alloggio che nel Centro Servizi stesso, che consentono di vedere contemporaneamente da un locale all'altro.

20.4 Telecom 95

È stata richiesta dalla STET la presentazione del prototipo di "Casa Intelligente" nel proprio stand a Telecom 95, che si terrà a Ginevra nel periodo 3-11 ottobre 1995. Si tratta di una delle maggiori manifestazioni mondiali dedicate al mondo della telematica e delle telecomunicazioni, e vede la partecipazione della quasi totalità dei produttori appartenenti a questo settore. Al momento non si conoscono ancora le modalità con le quali avverrà la dimostrazione.

21. Appendice A: Esempi di situazioni ambientali da gestire e relative procedure

Verranno ora descritti alcuni esempi di risoluzione di problemi che occorre gestire in un ambiente domestico. Per ognuno di essi ne verrà data una descrizione sommaria; verranno inoltre indicati gli eventi che si occupano di attivare le singole procedure, i passi elementari che il sistema deve effettuare per una corretta gestione del problema, il programma di controllo vero e proprio e come quest'ultimo debba venire inizializzato. Quest'ultima operazione è parte integrante della configurazione del sistema, ma a scopo esemplificativo è stata indicata esplicitamente.

21.1 Teleallarme

Descrizione:

L'utente preme il pulsante sul telecomando o un apposito pulsante sul telefono per richiedere aiuto. Verrà attivata una segnalazione (acustica, luminosa, ecc.) per indicare all'utente che la sua richiesta di allarme è stata ricevuta dal Sistema di Controllo. Se entro 10 secondi da questa segnalazione viene premuto l'apposito tasto di cancellazione, la procedura di teleallarme viene annullata, altrimenti procede con i passi necessari (inoltro della chiamata al Centro Servizi, ecc.).

Evento scatenante:

- Pulsante telecomando
- Pulsante telefono

Azioni elementari:

- Segnalazione all'utente che il teleallarme è stato attivato.
- Attesa dell'evento Teleallarme Annullato.
- Se l'utente ha annullato il teleallarme termina la procedura.
- Se sono trascorsi 10 secondi si procede con il teleallarme.

Programma di controllo:

1. Segnala_Utente ("Teleallarme inserito")
2. Attendi_Evento ("Pulsante annullamento teleallarme" , 10 secondi)
3. Leggi_Stato ("Pulsante annullamento teleallarme")
4. Se_Diverso ("Timeout" , fine)
5. "Procedi con la procedura di Teleallarme"

Inizializzazione procedura:

- Associa_Evento ("Pulsante telecomando teleallarme" , teleallarme.prog)
- Associa_Evento ("Pulsante telefono teleallarme" , teleallarme.prog)

21.2 Campanello porta

Descrizione:

Un visitatore preme il pulsante del campanello della porta d'ingresso. L'utente viene avvisato tramite un'appropriata segnalazione che qualcuno è alla porta.

Evento scatenante:

- Pulsante campanello

Azioni elementari:

- Segnalazione all'utente di visitatore alla porta

Programma di controllo:

1. Segnala_Utente ("Visitatore alla porta")

Inizializzazione procedura:

- Associa_Evento ("Pulsante campanello" , campanello.prog)

21.3 Apertura porta

Descrizione:

L'utente preme il pulsante dedicato all'apertura della porta. Se l'apertura non avviene, l'utente viene avvisato che c'è un problema e viene inserita una annotazione nel file di storico del sistema, in modo che il tecnico incaricato della manutenzione possa capire cosa è successo.

Evento scatenante:

- Pulsante apriporta

Azioni elementari:

- Controllo se la porta è già aperta.
- Se lo è, fine della procedura.
- Invio comando di apertura porta all'attuatore.
- Attesa per 30 secondi che la porta si apra.
- Se non si è aperta segnalazione utente e scrittura storico.

Programma di controllo:

1. Leggi_Stato ("Sensore porta")
2. Se_Uguale ("Aperta" , fine)
3. Comanda_Attuatore ("Attuatore porta" , "Apri")
4. Attendi_Evento ("Sensore porta" , 30 secondi)
5. Leggi_Stato ("Sensore porta")
6. Se_Diverso ("Timeout" , fine)
7. Segnala_Utente ("Attenzione! La porta non si è aperta!")
8. Errore ("La porta non si è aperta!")

Inizializzazione procedura:

- Associa_Evento ("Pulsante apriporta" , apriporta.prog)

21.4 Sensore fumo

Descrizione:

Il sensore di presenza di fumo rileva una situazione anomala nell'ambiente. L'utente viene avvisato e vengono messe in atto le appropriate contromisure. Il comportamento che deve seguire il sistema dipende molto dal grado di affidabilità dell'utente, ovvero da che grado di libertà gli si può lasciare nella gestione dell'ambiente domestico. Per questo esempio supponiamo che l'utente sia affidabile, perché questo comporta un maggior numero di interazioni con il Sistema di Controllo.

Evento scatenante:

- Sensore di presenza fumo

Azioni elementari:

- Segnalazione all'utente della presenza di fumo nell'ambiente.
- Attesa che l'utente annulli la procedura di allarme.
- Se la procedura viene annullata occorre attendere che il fumo nell'ambiente si diradi e scenda sotto il livello necessario ad attivare il sensore. Dopo di che la procedura termina.
- Se sono trascorsi 30 secondi senza che l'utente sia intervenuto, si procede con la procedura di allarme.
- Accensione delle luci ambiente se è notte.
- Apertura porta di ingresso per permettere all'utente di uscire.
- Chiamata del Centro Servizi o dei Vigili del Fuoco per comunicare l'emergenza.
- Scrittura sullo storico dei dati relativi all'allarme.

Programma di controllo:

1. Segnala_Utente ("Fumo nell'ambiente")
2. Attendi_Evento ("Comando annullamento allarme fumo" , 30 secondi)
3. Leggi_Stato ("Comando annullamento allarme fumo")
4. Se_Uguale ("Timeout" , 7)
5. Aspetta (2 minuti)
6. Fine
7. Comanda_Attuatore ("Luci di tutta la casa" , "Acceso")
8. Genera_Evento ("Pulsante apriporta")
9. "Chiama chi di dovere"
10. Errore ("Allarme sensore di fumo")

Inizializzazione procedura:

- Associa_Evento ("Sensore fumo" , fumo.prog)
- Associa_Evento ("Pulsante apriporta" , apriporta.prog)

21.5 Sensore acqua pavimento

Descrizione:

Il sensore di umidità rileva la presenza di acqua sul pavimento. L'utente viene avvisato e vengono messe in atto le appropriate contromisure. Il comportamento che deve seguire il sistema dipende molto dal grado di affidabilità dell'utente, ovvero da che grado di libertà gli si può lasciare nella gestione dell'ambiente domestico. Per questo esempio supponiamo che l'utente sia affidabile, perché questo comporta un maggior numero di interazioni con il Sistema di Controllo.

Evento scatenante:

- Sensore di umidità

Azioni elementari:

- Interruzione della fornitura di energia elettrica agli utilizzatori 'escludibili' del locale nel quale si è osservata la presenza di acqua sul pavimento. Per utilizzatori 'escludibili' si intendono gli elettrodomestici o le apparecchiature di supporto domestico la cui esclusione non provoca una diminuzione del livello di sicurezza all'interno dell'abitazione.

- Interruzione della fornitura d'acqua tramite un'appropriata elettrovalvola.
- Segnalazione all'utente della presenza di acqua sul pavimento di una certa stanza.
- Attesa che l'utente annulli la procedura di allarme.
- Se la procedura viene annullata occorre attendere che l'umidità provocata dalla presenza dell'acqua diminuisca e scenda sotto il livello necessario ad attivare il sensore. Viene poi richiesto all'utente se occorre ripristinare la fornitura d'acqua oppure no (per esempio l'utente può aver versato una bacinella d'acqua per terra, e quindi sarebbe inutile tenere bloccata l'acqua dell'impianto idrico). Dopo di che la procedura termina.
- Se sono trascorsi 30 secondi senza che l'utente sia intervenuto, si procede con la procedura di allarme.
- Chiamata del Centro Servizi per comunicare l'emergenza.
- Scrittura sullo storico dei dati relativi all'allarme.

Programma di controllo:

1. Comanda_Attuatore ("Interruttore elettricità stanza" , Aperto)
2. Comanda_Attuatore ("Elettrovalvola impianto idraulico" , Aperto)
3. Segnala_Utente ("Acqua sul pavimento")
4. Attendi_Evento ("Comando annullamento allarme acqua" , 30 secondi)
5. Leggi_Stato ("Comando annullamento allarme acqua")
6. Se_Uguale ("Timeout" , 19)
7. Segnala_Utente ("Ripristino fornitura acqua?")
8. Attendi_Evento ("Comando ripristino fornitura acqua" , no Timeout)
9. Leggi_Stato ("Comando ripristino fornitura acqua")
10. Se_Uguale ("No" , 12)
11. Comanda_Attuatore ("Elettrovalvola impianto idraulico" , Chiuso)
12. Segnala_Utente ("Ripristino fornitura energia elettrica?")
13. Attendi_Evento ("Comando ripristino fornitura energia elettrica" , no Timeout)
14. Leggi_Stato ("Comando ripristino fornitura energia elettrica")
15. Se_Uguale ("No" , 17)
16. Comanda_Attuatore ("Interruttore elettricità stanza" , Chiuso)
17. Aspetta (5 minuti)
18. Fine
19. "Chiama chi di dovere"
20. Errore ("Allarme sensore di fumo")

Inizializzazione procedura:

- Associa_Evento ("Sensore acqua pavimento" , fumo.prog)

21.6 Alzatapparelle

Descrizione:

L'utente preme il pulsante per alzare o abbassare la tapparella. Quando lo rilascia la tapparella si ferma. Per questo esempio consideriamo il programma di controllo relativo al pulsante utilizzato per sollevare la tapparella; il problema relativo all'altro pulsante è analogo. Si suppone che l'alzatapparella sia dotato di un motore e di un sensore che provveda a segnalare quando la tapparella è tutta sollevata. Si suppone inoltre che il motore sia dotato di

una frizione che gli impedisca di danneggiarsi quando la tapparella arriva in cima e il pulsante non viene rilasciato. Notare che questa procedura usa come evento scatenante lo stesso evento usato poi all'interno della stessa come controllo del comando utente. Questo è possibile perché la prima operazione effettuata dall'Esecutore Programmi quando viene richiamata una procedura è la disattivazione dell'evento responsabile della sua attivazione.

Evento scatenante:

- Pulsante sollevamento tapparella

Azioni elementari:

- Controlla che la tapparella non sia già tutta sollevata.
- Se lo è esci dalla procedura.
- Attiva il motore dell'alzatapparella per fare in modo che si sollevi.
- Attesa dell'evento Pulsante sollevamento tapparella rilasciato per 1 minuto.
- Se non si è alzata (quindi possibile guasto di: motore, pulsante o sensore) segnalazione utente, scrittura storico e uscita dalla procedura.
- Disattiva il motore dell'alzatapparella.

Programma di controllo:

1. Leggi_Stato ("Sensore alzatapparella")
2. Se_Uguale ("Sollevata" , fine)
3. Comanda_Attuatore ("Alzatapparella" , "Solleva")
4. Attendi_Evento ("Pulsante alzatapparella" , 1 minuto)
5. Leggi_Stato ("Pulsante alzatapparella")
6. Se_Diverso ("Timeout" , 11)
7. Segnala_Utente ("Attenzione! Alzatapparella non funzionante!")
8. Errore ("Alzatapparella non funzionante!")
9. Fine
10. Comanda_Attuatore ("Alzatapparella" , "Stop")

Inizializzazione procedura:

- Associa_Evento ("Pulsante alzatapparella" , tapparella_su.prog)

21.7 Predisposizione diurna / notturna abitazione

Descrizione:

In base all'orario predisposto, la situazione ambientale (luci, finestre, porte, riscaldamento) viene mutata, per adattarla alle esigenze dell'utente. In questo esempio verrà presentata la procedura per l'attivazione della configurazione notturna. Si suppone esistano una serie di programmi di configurazione dell'ambiente, chiamati situazione?.prog, che contengono le istruzioni per configurare i sensori e gli attuatori in un certo modo. Si è scelto di separare la procedura di configurazione ambientale da quella richiamata dal timer orario non solo per maggior chiarezza espositiva, ma anche per separare nettamente le due fasi. Per vedere come questi programmi di configurazione sono strutturati, riferirsi all'apposito esempio.

Evento scatenante:

- Timer orario

Azioni elementari:

- Richiama l'appropriata procedura di configurazione.

Programma di controllo:

1. Genera_Evento ("Configurazione1")

Inizializzazione procedura:

- Attiva_Ciclicamente (pred_notte.prog , 21.00)
- Associa_Evento ("Configurazione1" , situazione1.prog)

21.8 Configurazione notturna abitazione

Descrizione:

Procedura che si occupa di configurare l'ambiente, predisponendolo per la notte. È composta da una serie di passi elementari di controllo e predisposizione. Per ovvi motivi di esemplificazione, questa procedura non è sicuramente completa ed esaustiva.

Evento scatenante:

- Nessuno. Viene richiamata da un'altra procedura.

Azioni elementari:

- Interruzione della fornitura di gas da cucina all'appartamento.
- Configura il termostato della caldaia per una temperatura adatta alla notte (per esempio 18°).
- Comanda la chiusura delle finestre.
- Controlla la chiusura della porta d'ingresso.
- Se non è chiusa, chiudila.
- Accendi le spie luminose per l'orientamento al buio.

Programma di controllo:

1. Comanda_Attuatore ("Elettrovalvola gas cucina" , Chiuso)
2. Configura_Sensore ("Termostato caldaia" , "Soglia temperatura" , 18°)
3. Genera_Evento ("Pulsante chiusura finestra 1")
4. Genera_Evento ("Pulsante chiusura finestra 2")
5. ...
6. Leggi_Stato ("Sensore porta")
7. Se_Uguale ("Chiusa" , 15)
8. Comanda_Attuatore ("Attuatore porta" , "Chiudi")
9. Attendi_Evento ("Sensore porta" , 30 secondi)
10. Leggi_Stato ("Sensore porta")
11. Se_Diverso ("Timeout" , 15)
12. Segnala_Utente ("Attenzione! La porta non si è chiusa!")
13. Errore ("La porta non si è chiusa!")
14. Comanda_Attuatore ("Spie luminose" , Acceso)

Inizializzazione procedura:

- Associa_Evento ("Configurazione1" , situazione1.prog)

- Associa_Evento ("Pulsante chiusura finestra 1" , chiudi_f_1.prog)
- Associa_Evento ("Pulsante chiusura finestra 2" , chiudi_f_2.prog)
- ...

21.9 Controllo temperatura ambiente

Descrizione:

Procedura che si occupa di controllare la temperatura dell'ambiente, facendo in modo da mantenerla ad un valore prestabilito. Il programma di controllo considera un intervallo di ±2° intorno al valore prestabilito.

Evento scatenante:

- Viene attivata ciclicamente ogni 3 minuti.

Azioni elementari:

- Lettura del valore della temperatura.
- Confronto con il valore prestabilito, per esempio 20°.
- Se è inferiore o superiore prendere le appropriate contromisure.

Programma di controllo:

1. Leggi_Stato ("Temperatura ambiente")
2. Se_Maggiore (22° , 7)
3. Se_Minore (18° , 10)
4. Comanda_Attuatore ("Riscaldamento" , Spento)
5. Comanda_Attuatore ("Condizionatore" , Spento)
6. Fine
7. Comanda_Attuatore ("Riscaldamento" , Spento)
8. Comanda_Attuatore ("Condizionatore" , Acceso)
9. Fine
10. Comanda_Attuatore ("Riscaldamento" , Acceso)
11. Comanda_Attuatore ("Condizionatore" , Spento)

Inizializzazione procedura:

- Attiva_Ciclicamente (ctrltemp.prog , ogni 3.00 minuti)

22. Appendice B: Struttura dei files utilizzati

22.1 Files Oggetto

I files oggetto sono la struttura dati contenente la sequenza di istruzioni base comprensibili dal Sistema di Controllo, cioè le istruzioni costituenti i programmi di controllo dell'ambiente. Questi files vengono generati dalla compilazione dei programmi scritti mediante l'Ambiente di Configurazione e Programmazione della Casa reso disponibile al personale tecnico, e vengono caricati dal Sistema di Controllo in fase di attivazione. Per ogni programma che viene scritto, saranno presenti su disco due diversi files: il primo (**.CIP*) conterrà il programma in formato sorgente, cioè quello scritto o modificato dal programmatore, mentre il secondo (**.CIO*) sarà il frutto della compilazione e quindi l'unico in grado di essere compreso dal Sistema di Controllo.

I files oggetto sono costituiti da un numero variabile di elementi lunghi 7 byte contenenti le seguenti informazioni:

- *Codice Operativo*: identificazione univoca dell'istruzione da eseguire. (1 byte)
- *Parametro 1*: primo parametro dell'istruzione. (2 byte)
- *Parametro 2*: secondo parametro dell'istruzione. (2 byte)
- *Parametro 3*: terzo parametro dell'istruzione. (2 byte)

Nel linguaggio C++ ciò può essere scritto come:

```
struct riga {
        BYTE opcode;
        int par1;
        int par2;
        int par3;
}
```

Il campo *Codice Operativo* verrà utilizzato dal Sistema di Controllo per identificare univocamente l'istruzione che occorre eseguire, associandogli inoltre in modo corretto i parametri.

Nella Tabella 22.1 sono riportate tutte le istruzioni riconosciute dal sistema, il numero di parametri ad esse associate ed il codice operativo che le identifica. Viene inoltre indicato il procedimento utilizzato per ottenere l'univocità del codice operativo.

Tabella 22.1: Codici operativi delle istruzioni riconosciute dal Sistema di Controllo

Nome istruzione	Numero parametri	Codice operativo (esadecimale)	Struttura del Codice operativo	
			N° parametri	Identificatore
Abilita_Evento	1	0x21	001	00001
Annulla_Attesa_Evento	1	0x22	001	00010
Annulla_Timer_Ciclico	1	0x23	001	00011
Aspetta	2	0x41	010	00001
Associa_Evento	2	0x42	010	00010
Attendi_Evento	2	0x43	010	00011
Attiva_Ciclicamente	2	0x44	010	00100
Azzera_Contatore	1	0x24	001	00100
Comanda_Attuatore	2	0x45	010	00101
Configura_Sensore	3	0x61	011	00001
Controllo_Stato_Sistema	1	0x25	001	00101
Disabilita_Evento	1	0x26	001	00110
Errore	1	0x27	001	00111
Fine	0	0x01	000	00001
Genera_Evento	1	0x29	001	01001
Lancia	1	0x28	001	01000
Leggi_Contatore	1	0x2a	001	01010
Leggi_Ora	0	0x02	000	00010
Leggi_Stato	1	0x2b	001	01011
Risposta_Utente	1	0x2c	001	01100
Segnala_Utente	1	0x2d	001	01101
Se_Uguale	2	0x46	010	00110
Se_Diverso	2	0x47	010	00111
Se_Maggiore	2	0x48	010	01000
Se_Minore	2	0x49	010	01001
Teleallarme	2	0x4a	010	01010

22.2 Files di configurazione del Sistema di Controllo

Si è visto nella descrizione del funzionamento del Sistema di Controllo che questo è governato da un insieme di programmi collegati sia a degli eventi sia a delle scadenze temporali. Questa associazione tra programmi di controllo e gli eventi viene realizzata tramite due diversi file. Il primo, *SISTEMA.CFG*, contiene l'insieme dei programmi che governano il Sistema di Controllo e gli eventi che li attivano, mentre il secondo,*TIMER.CFG*, contiene un insieme di indicazioni temporali con indicati i nomi dei programmi che devono essere attivati in particolari orari della giornata.

22.2.1 Configurazione programmi

La struttura dati che contiene le informazioni necessarie a realizzare la corretta associazione tra gli eventi ed i corrispondenti programmi è composta da un vettore di elementi lunghi 13 byte strutturati nel modo seguente:

- *Programma*: nome del programma di controllo. (8 byte)
- *Sensore*: identificatore del sensore responsabile dell'attivazione del programma. (2 byte)
- *Valore*: valore che deve assumere il sensore specificato per provocare l'attivazione del programma. (2 byte)
- *Priorità*: caratterizza l'urgenza con la quale il sistema deve rispondere a questo particolare evento. (1 byte)

Nel linguaggio C++ ciò può essere scritto come:

```
struct riga {
        char programma[8];
        UINT sensore;
        int valore;
        BYTE pri;
}
```

Il codice di identificazione del sensore viene associato al nome "mnemonico" in fase di configurazione, all'interno del Sistema di Sviluppo.

22.2.2 Configurazione programmi da attivare ad un orario prestabilito

È importante poter impostare l'esecuzione di particolari compiti ad orari prestabiliti. Questo viene realizzato tramite la struttura dati, organizzata in un vettore avente elementi lunghi 10 byte, che è descritta qui di seguito:

- *Programma*: nome del programma di controllo. (8 byte)
- *Orario*: scadenza temporale alla quale deve essere attivato il programma. (2 byte)

Nel linguaggio C++ ciò può essere scritto come:

```
struct riga {
        char programma[8];
        UINT orario;
}
```

Il campo *Orario* è memorizzato mediante il formato *hhmm* all'interno dei 2 byte costituenti il secondo campo della struttura. Per ovvie ragioni, questo campo conterrà esclusivamente valori appartenenti all'intervallo $0 \leq hh \leq 23$ e $0 \leq mm \leq 59$.

22.3 Files di configurazione del Sistema di Sviluppo

All'interno del Sistema di Sviluppo vengono utilizzati una serie di files per memorizzare i dati riguardanti le caratteristiche ambientali alle quali si fa riferimento all'interno dei programmi di controllo. Queste informazioni riguardano gli identificatori dei sensori e degli attuatori, rispettivamente in *SENSORI.DAT* e *ATTUATORI.DAT*, il modo in cui vanno effettuate le segnalazioni all'utente, in *MESSAGGI.DAT*, e come possono essere formulate le risposte da parte dell'utente, nel file *RISPOSTE.DAT*. Sono inoltre presenti i dati riguardanti

l'identificazione ed i permessi di accesso al Sistema di Sviluppo da parte degli utenti, nel file *PERMESSI.PW.*

22.3.1 Configurazione sensori ed attuatori

L'associazione tra gli identificatori simbolici dei sensori e degli attuatori ed i loro codici viene memorizzata in due diverse strutture dati sostanzialmente identiche. Queste strutture sono composte da un vettore di elementi lunghi 22 byte così organizzati:

- *Nome*: nome del sensore o dell'attuatore. (20 byte)
- *Identificatore*: codice numerico che identifica in modo univoco all'interno del sistema il dispositivo. Dovrà essere compreso nell'intervallo $0 \leq id \leq 65535$. (2 byte)

Nel linguaggio C++ ciò può essere scritto come:

```
struct riga {
        char nome[20];
        UINT identif;
}
```

Questi dati verranno utilizzati sia in fase di compilazione dei singoli programmi di controllo, sia durante la fase di associazione dei programmi agli eventi che li devono attivare.

22.3.2 Configurazione segnalazioni e messaggi

Si è visto che uno dei requisiti di questo progetto è la possibilità di inviare e ricevere messaggi all'utente in maniera tale da non dover modificare i programmi di controllo se vi è necessità di cambiare queste modalità di interazione. Questo viene realizzato tramite le strutture dati Messaggi e Risposte, che realizzano la corrispondenza tra un tipo di messaggio o di risposta ed i dispositivi che si devono occupare di inoltrare il messaggio o di ottenere una risposta.

È stato ritenuto sufficiente avere per ogni messaggio o risposta cinque dispositivi massimi. Le due strutture dati sono simili tra loro, e sono composte da un vettore i cui elementi hanno ognuno dimensione 11 byte.

La struttura dati per i messaggi è la seguente:

- *Messaggio*: identificatore del messaggio che si vuole inviare. (1 byte)
- *Dispositivo[ ]*: identificatore di dispositivo. (1 byte * 5)
- *Dati[ ]*: i dati che occorre inviare al dispositivo specificato per generare il tipo di messaggio richiesto. (1 byte * 5)

Nel linguaggio C++ ciò può essere scritto come:

```
struct riga {
        BYTE messaggio;
        struct{
          BYTE nome;
          BYTE dato;
        } dispositivi[5];
}
```

La struttura dati per le risposte è la stessa di quella utilizzata per i messaggi, ma viene riportata interamente per chiarezza:

- *Risposta*: identificatore della risposta che si vuole attendere. (1 byte)
- *Dispositivo[ ]*: identificatore di dispositivo. (1 byte * 5)
- *Dati[ ]*: i dati che occorre ricevere dal dispositivo specificato per ottenere il tipo di risposta richiesta. (1 byte * 5)

Nel linguaggio C++ ciò può essere scritto come:

```
struct riga {
        BYTE risposta;
        struct{
          BYTE nome;
          BYTE dato;
        } dispositivi[5];
}
```

22.3.3 Configurazione permessi d'accesso

A causa della delicatezza di alcune delle operazioni riguardanti la configurazione e la programmazione del Sistema di Controllo, è permesso l'accesso alle diverse fasi di configurazione solo alle persone dotate di un apposito sistema di riconoscimento, cioè l'insieme di un *identificativo utente* e di una *password*.

Questi dati sono memorizzati all'interno di una struttura dati, i cui elementi sono composti da 31 byte ed hanno la seguente struttura:

- *Nome utente*: Identificativo dell'utente. (15 byte)
- *Password*: codice segreto associato all'utente. (15 byte)
- *Permessi*: operazioni che sono permesse all'utente. (1 byte)

Nel linguaggio C++ ciò può essere scritto come:

```
struct riga {
        char nome[15];
        char password[15];
        BYTE perm;
}
```

Le operazioni permesse all'utente sono codificate tramite i bit del campo *permessi*, come riportato nella Tabella 22.2. Se un'operazione ha il bit associato ad 1, allora l'utente può effettuare modifiche ai dati relativi, mentre se il bit vale 0 non è possibile accedere a questi dati.

Tabella 22.2: Bit di protezione

				Attuatori	Sensori	Messaggi	Risposte

23. Appendice C: Interfacciamento del Sistema di Controllo con altri applicativi

La possibilità di far interagire l'utente con qualsiasi tipo di programma applicativo, utilizzando il Sistema di Controllo come intermediario, può semplificare enormemente l'utilizzo di sistemi informatici da parte di un utenza assolutamente non tecnica. Inoltre, rendendo disponibili i dispositivi di controllo e di segnalazione presenti nell'ambiente domestico ad applicativi diversi, rende questo sistema enormemente flessibile ed espandibile.

Occorre quindi poter effettuare un mutuo scambio di informazioni tra il Sistema di Controllo e gli altri programmi in esecuzione all'interno dell'ambiente Windows. Per risolvere questo tipo di esigenza ci si avvale di uno strumento basato sul passaggio di dati tramite l'uso di appositi messaggi; questo sottoinsieme dell'API[12] prende il nome di protocollo di comunicazione dinamica dei dati (DDE o Dynamic Data Exchange). Questo è un meccanismo di scambio dei dati regolato direttamente dalle applicazioni (application-driven).

23.1 Dynamic Data Exchange

Il protocollo di comunicazione *Dynamic Data Exchange* è lo strumento offerto da Windows per collegare due applicazioni (due moduli EXE) permettendo lo scambio di informazioni. Nella sua prima implementazione consiste nella generazione di una serie di messaggi da parte di un'applicazione, chiamata *client*, indirizzati verso un'applicazione ricevente, chiamata *server*.

Ciò significa che generalmente un *client* inizia ad inviare una serie di messaggi DDE per rispondere ad una propria esigenza di funzionamento e solo dopo qualche tempo (magari solo qualche centesimo di secondo o molto più) un'altra applicazione incomincia a rispondere fornendo le informazioni richieste; questo tipo di rapporto prende il nome di *conversazione*, mentre le interrogazioni del *client* verso il *server* e le risposte di quest'ultimo vengono definite *transazioni*. I dati sono l'oggetto delle transazioni bidirezionali *server-client* e sono costituiti da gruppi di elementi omogenei per formare degli argomenti (*topic*). In realtà tramite la generazione di un messaggio DDE non avviene il trasferimento di oggetti presenti nella memoria, ma solo dei puntatori ad essi associati.

Un argomento (*topic*) può essere quindi costituito da una serie di transazioni a loro volta formate da diversi *data item* (elementi di dati). Il passaggio di informazioni tra *client* e *server* avviene a tre distinti livelli:

- applicazione (*application*)
- argomento (*topic*)
- dato (*item*)

I primi due parametri riguardano tipicamente l'invio di un messaggio di impostazione di una conversazione e vengono interpretati ed analizzati dal *server* all'interno della propria window procedure. Il dato entra in gioco a conversazione instaurata.

La distinzione tra *client* e *server* non è però netta; un'applicazione può essere allo stesso tempo *client* e *server* oppure *server* o *client* nei confronti di altre applicazioni. Per questa seconda eventualità è indispensabile che il programma disponga di più finestre esistendo la limitazione di fondo che impone una sola conversazione DDE per finestra. Questa regola non

[12] *Application Program Interface. Sono così chiamate le funzioni di sistema dell'ambiente Windows richiamabili dall'interno dei programmi.*

è assolutamente vincolante, ma deriva dalla pratica e dalla volontà di scrivere del codice facile da leggere e da manutenere.

23.2 I messaggi DDE

Il protocollo dei messaggi DDE comprende nove messaggi caratterizzati dal prefisso WM_DDE_ e contenuti nel file DDE.H fornito con l'SDK[13] di Windows. La Tabella 23.1 riassume i nove messaggi DDE.

Tabella 23.1: I messaggi del protocollo DDE

Messaggio	**Descrizione**
WM_DDE_INITIATE	Instaurazione di una conversazione *client/server*; questo messaggio viene sempre inviato dall'applicazione che prenderà il nome di *client*.
WM_DDE_TERMINATE	Termine di una conversazione *client-server*, inviato sia dal *client* che dal *server*.
WM_DDE_ADVISE	Richiesta al *server* di fornire un aggiornamento dei dati quando risultano modificati.
WM_DDE_UNADVISE	Richiesta al *server* di interrompere l'aggiornamento relativo al tipo di dati specificato nel messaggio.
WM_DDE_ACK	Inviato in risposta a qualsiasi messaggio, in particolare WM_DDE_INITIATE e WM_DDE_EXECUTE quale conferma della ricezione delle informazioni.
WM_DDE_DATA	Notifica ad un'applicazione *client* la disponibilità di un certo pacchetto di dati.
WM_DDE_REQUEST	Richiesta al *server* di fornire un certo tipo di dati.
WM_DDE_POKE	Messaggio utilizzato per il trasferimento di dati in risposta ad una richiesta formulata come WM_DDE_REQUEST.
WM_DDE_EXECUTE	Invio di una stringa di testo ad un *server* che la interpreta come una serie di comandi che occorre eseguire.

Ancora più interessante è analizzare la direzione di spedizione di ciascuno dei nove messaggi DDE. La Tabella 23.2 riassume il tutto. È interessante notare come il flusso sia fortemente indirizzato verso il server e come in alcuni casi si verifichi la ricezione dello stesso messaggio da entrambi i lati di una conversazione DDE.

[13] *Software Development Kit. Insieme degli strumenti software messi a disposizione dalla Microsoft per la realizzazione di applicazioni all'interno dell'ambiente Windows.*

Tabella 23.2: Direzione del flusso dei messaggi una volta instaurata una conversazione DDE

Client	Messaggio	Server
	WM_DDE_INITIATE	→
←	WM_DDE_ACK	→
	WM_DDE_REQUEST	→
	WM_DDE_POKE	→
	WM_DDE_ADVISE	→
	WM_DDE_UNADVISE	→
←	WM_DDE_DATA	
	WM_DDE_EXECUTE	→
←	WM_DDE_TERMINATE	→

23.3 L'invio dei messaggi

Il passaggio dei messaggi DDE tra client e server è regolato dall'uso delle apposite funzioni dell'API di Windows: *SendMessage()* e *PostMessage()*.

Nel primo caso si procede alla generazione di un messaggio di tipo sincrono indirizzato direttamente alla window procedure della finestra il cui handle compare come primo parametro della funzione (si parla di messaggi non accodati). Con *PostMessage()*, invece, il messaggio viene imbucato direttamente nella coda dei messaggi dell'applicazione dando origine ad un messaggio asincrono o accodato. La regola che permette di distinguere tra l'uso di *SendMessage()* rispetto a *PostMessage()* è molto semplice; la prima funzione viene utilizzata solo nel caso di invio di WM_DDE_INITIATE e della conseguente risposta del server di partecipazione alla conversazione con un WM_DDE_ACK. In tutte le altre circostanze si ricorrerà a *PostMessage()*.

La logica di questa scelta è evidente. Il controllo dell'esecuzione del programma non ritorna da una chiamata a *SendMessage()* fino a quando il messaggio non è stato elaborato, corrispondendo in questo caso all'individuazione in un'applicazione che si vuole configurare come un server nei confronti del client. Questo comportamento permette di interrompere l'esecuzione del client che potrà procedere nell'elaborazione del messaggio che ha dato origine alla ricerca del server soltanto quando verrà a conoscenza della presenza o meno di applicazioni server. Sempre per lo stesso motivo, quando un'applicazione accetta di operare in qualità di server di un'altra le notifica la propria disponibilità mediante l'invio di WM_DDE_ACK veicolato con *SendMessage()*.

È utile per l'analisi del meccanismo della conversazione DDE vedere brevemente la sintassi delle istruzioni *SendMessage()* e *PostMessage()* adattata alle nostre specifiche esigenze:

SendMessage(hwnd, msg, wParam, lParam)

- *hwnd*: handle della finestra ricevente
- *msg*: messaggio DDE
- *wParam*: hwnd della finestra mittente
- *lParam*: oggetto a 32 bit contenente diversi valori

L'handle *hwnd* individua la finestra ricevente ed è seguito dal messaggio vero e proprio (*msg*). Il *wParam* è specificato l'handle della finestra che ha provveduto alla spedizione del messaggio, mentre in *lParam* sono contenute informazioni specifiche dei diversi messaggi.

23.4 Iniziare una conversazione

Per stabilire un collegamento (*link*) tra due applicazioni è necessario che una delle due, quella che andrà a configurarsi come client, proceda all'invio di un messaggio WM_DDE_INITIATE a tutte le applicazioni attive nell'ambiente Windows. La precedente definizione è concettualmente corretta: da un'applicazione (il potenziale client) si procede alla ricerca di un server, sicuramente un'altra applicazione. Più correttamente, quanto avviene in Windows consiste nell'invio del messaggio WM_DDE_INITIATE verso tutte le finestre di tipo overlapped e popup presenti nell'ambiente. Rimarranno quindi escluse da questa ricerca le finestre create con il flag WS_CHILD. Un tipico esempio di *SendMessage()* per iniziare una conversazione è:

```
SendMessage(HWND_BROADCAST, WM_DDE_INITIATE, (WPARAM) hwnd, ...)
```

Ovviamente il potenziale client non sa nulla del server che sta ricercando; potrebbe anche non esistere alcuna applicazione (finestra) attiva in Windows in grado di soddisfare le esigenze del client. Ciò che occorre fare è piuttosto un'azione di *broadcasting* verso tutte le finestre top-level presenti in Windows, nella speranza di trovarne una che soddisfi le nostre esigenze di potenziale client. Il valore dell'handle della finestra ricevente è stato posto uguale a HWND_BROADCAST (-1) per indicare che indiscriminatamente tutte le finestre possono divenire le fornitrici delle informazioni richieste dall'applicazione client. In *wParam* compare invece l'*hwnd* della finestra mittente. Per il momento è stato trascurato *lParam*. Eseguita la *SendMessage()* con WM_DDE_INITIATE faremo ritorno all'istruzione immediatamente successiva soltanto dopo che il messaggio in questione è stato recapitato a tutte le finestre top-level.

Se una delle finestre che riceve il messaggio è un potenziale server, occorre rendere partecipe immediatamente il potenziale client (la finestra che ha inviato il messaggio WM_DDE_INITIATE) della nostra presenza. L'*hwnd* del mittente è contenuto in *wParam*: non resta altro che spedire il messaggio WM_DDE_ACK per instaurare di fatto la conversazione tra le due applicazioni (finestre). Anche in questo caso il trasportatore è *SendMessage()*: vogliamo essere sicuri che il client venga a conoscenza di questo potenziale server immediatamente.

23.5 Il contenuto di *lParam*

Quando un client inizia la ricerca di un server, può essere estremamente selettivo o assolutamente generico. Può cioè richiedere espressamente di instaurare una conversazione DDE con una particolare applicazione o, all'estremo opposto, essere disposto a "chiacchierare" con chiunque. Si tratta del livello applicazione (*aApp*), il primo dei tre di una conversazione DDE a messaggi. Lo stesso atteggiamento vale anche per l'argomento della conversazione, *aTopic*: può trattarsi di un elemento specifico o di qualcosa di estremamente generico. Comunque sia, è indispensabile che queste informazioni vengano specificate direttamente al momento della ricerca di un potenziale server. L'unico parametro a nostra disposizione è rappresentato da *lParam*, insufficientemente grande per contenere due stringhe di testo per individuare i due elementi. Si ricorre quindi agli atomi.

Un atomo è un dato di tipo intero il cui valore corrisponde ad una stringa di testo; per semplificare si può immaginare un atomo come l'identificatore di un dato presente in una tabella. La differenza è che questo valore viene ritornato dall'API di Windows e non viene impostato dal programmatore. La scelta degli atomi come veicolo per il passaggio di informazioni tra client e server è quantomai appropriata perché consente di impacchettare in 32 bit informazioni molto più voluminose.

I due atomi *aApp* e *aTopic* individuano rispettivamente il nome del server ricercato e l'argomento oggetto di future conversazioni. Specificando il valore NULL per entrambi i parametri si intende che la richiesta di inizio di una conversazione debba essere la più generale possibile: indirizzata a tutti e su tutto. L'inizio della conversazione DDE diventerà ora:

```
SendMessage(HWND_BROADCAST,WM_DDE_INITIATE,(WPARAM)hwnd,MAKELPARAM(aApp,
aTopic));
```

L'applicazione che riceve il messaggio WM_DDE_INITIATE provvede ad eseguire una serie di test per verificare le seguenti condizioni:

- il server specificato in *aApp* è identificabile con l'applicazione;
- *aApp* non specifica alcun server in particolare.

Terminato di valutare il contenuto di *aApp*, l'applicazione che ha ricevuto il messaggio di inizializzazione di un collegamento DDE passa all'esame di quanto richiesto dal client in termini di argomenti della possibile conversazione. I test che riguardano aTopic consistono in:

- individuare l'argomento proposto;
- determinare se il server può fornire quanto richiesto dall'applicazione client;
- notificare al client la disponibilità a partecipare alla conversazione o fargli pervenire un elenco dei soggetti per una possibile conversazione qualora non fosse stato specificato un argomento preciso.

La risposta del server ad un messaggio WM_DDE_INITIATE assume il seguente aspetto:

```
SendMessage(wParam, WM_DDE_ACK, (WPARAM) hwnd, MAKELPARAM(aApp, aTopic));
```

Qualora il server dovesse notificare la disponibilità ad instaurare delle conversazioni su più temi (*topics*) l'invio di WM_DDE_ACK deve essere ripetuto in un numero pari ai soggetti trattabili.

Una conversazione DDE si intende instaurata quando il client non esegue un'esplicita azione di terminazione in conseguenza della ricezione di un messaggio WM_DDE_ACK da parte del server. Inoltre, una conversazione può vertere su un unico argomento (*topic*). Nel caso in cui il server provveda a notificare al client differenti argomenti è opportuno che quest'ultimo proceda all'uso di WM_DDE_TERMINATE per terminare le conversazioni alle quali non è interessato. Con WM_DDE_INITIATE e WM_DDE_ACK terminano tutti i casi nei quali si fa ricorso a *SendMessage()*.

23.6 Gestire gli atomi

In Windows esistono due tabelle di atomi: quella locale e quella globale. La prima opera all'interno del segmento dati di un'applicazione ed è ad uso esclusivo di un programma. La seconda (quella che ci interessa nell'implementazione di una conversazione DDE) è invece accessibile da qualsiasi applicazione attiva nel sistema.

Per generare un nuovo atomo si usa la funzione *AddAtom()* che richiede come argomento un puntatore ad una stringa di testo ASCIIZ. Una stringa può comparire nella tabella degli atomi una sola volta; qualora si tentasse di aggiungerne una seconda copia la funzione *AddAtom()* ritornerebbe l'atomo che già la identifica incrementando anche il contatore di riferimento al numero di accessi fatti a quell'atomo. L'operazione contraria è svolta da *DeleteAtom()*; l'atomo scompare definitivamente dalla tabella, cioè la stringa di testo non è più accessibile, soltanto quando il contatore di riferimento si annulla. Per poter eseguire questa funzione dell'API è indispensabile disporre di un atomo di riferimento. Con *FindAtom()* si ottiene l'atomo che identifica la stringa passata a questa funzione come suo unico argomento.

Disponendo sempre di un atomo è possibile recuperare la stringa di testo associata e memorizzarla in un apposito buffer mediante *GetAtomName()*.

Tutte le funzioni finora descritte si riferiscono alla gestione degli atomi locali, cioè residenti all'interno del *data segment* dell'applicazione e quindi di uso esclusivo. L'equivalente globale è rappresentato dalle stesse funzioni facendole precedere dal prefisso *Global*, mantenendo però inalterata la sintassi ed il numero di parametri coinvolti. La Tabella 23.3 riassume tutti gli strumenti messi a disposizione dell'API di Windows correlati all'uso degli atomi.

Tabella 23.3: Le funzioni, la macro e il tipo di dato associato all'uso degli atomi

Gli atomi in Windows
typedef WORD ATOM;
BOOL FAR PASCAL InitAtomTable(int);
ATOM FAR PASCAL AddAtom(LPSTR);
ATOM FAR PASCAL DeleteAtom(ATOM);
ATOM FAR PASCAL FindAtom(LPSTR);
WORD FAR PASCAL GetAtomName(ATOM, LPSTR, int);
ATOM FAR PASCAL GlobalAddAtom(LPSTR);
ATOM FAR PASCAL GlobalDeleteAtom(ATOM);
ATOM FAR PASCAL GlobalFindAtom(LPSTR);
WORD FAR PASCAL GlobalGetAtomName(ATOM, LPSTR, int);
HANDLE FAR PASCAL GetAtomHandle(ATOM);
MATEINTATOM(i)

È importante notare che il primo dei due elementi a 16 bit costituisce la word inferiore e il secondo la superiore; questa distinzione va ricordata quando si specificano i parametri dei messaggi DDE. Il significato della high-word di *lParam*, come d'altronde la low-word, varia in funzione del messaggio inviato. La Tabella 23.4 descrive gli elementi che possono essere inseriti in *lParam*.

Tabella 23.4: Il contenuto di lParam in funzione del messaggio inviato

Messaggio	**Low word**	**High Word**
WM_DDE_INITIATE	aApp	aTopic
WM_DDE_TERMINATE	riservato	riservato
in risposta a WM_DDE_INITIATE	aApplication	aTopic
in risposta a WM_DDE_EXECUTE	wStatus	hCommands
in risposta a tutti gli altri messaggi	wStatus	aItem
WM_DDE_REQUEST	cfFormat	aItem
WM_DDE_DATA	hData	aItem
WM_DDE_ADVISE	hOptions	aItem
WM_DDE_UNADVISE	riservato	aItem
WM_DDE_POKE	hData	aItem
WM_DDE_EXECUTE	riservato	hCommands

I campi indicati nella tabella hanno i seguenti significati:

- *aApp*: nome dell'applicazione
- *aTopic*: argomento
- *aItem*: elemento di dato oggetto del messaggio
- *cfFormat* :formato della Clipboard
- *hData*: handle ad un oggetto nella memoria globale
- *hOptions*: handle ad un oggetto nella memoria globale
- *hCommands*: handle ad un oggetto nella memoria globale
- *wStatus*: parametro lungo una word

In generale la sintassi relativa all'invio di un messaggio DDE a conversazione instaurata assume il seguente aspetto:

```
PostMessage(hwndDest, WM_DDE_xxx, (WPARAM)hwndSender, MAKELPARAM(low-word, high-word));
```

Lo spazio a disposizione per impostare il contenuto del messaggio DDE è limitato a *lParam*, una quantità estremamente ridotta, ma più che sufficiente anche per trasferire grandi oggetti di memoria mediante un handle.

23.7 Le conversazioni DDE

Le conversazioni DDE sono sempre dei collegamenti secondo una direzione precisa tra due applicazioni chiamate client-server in funzione del ruolo assunto (chi interroga e chi risponde). La prima operazione da compiere consiste nell'iniziare una conversazione. Superata questa prima fase il client è in condizione di richiedere delle informazioni al server. Queste richieste possono essere di tipo estemporaneo o permanente.

23.8 Collegamenti estemporanei

La realizzazione di collegamenti di tipo estemporaneo è la conseguenza dell'uso dei messaggi WM_DDE_REQUEST e WM_DDE_POKE. Il primo viene impiegato per richiedere al

server un determinato dato, mentre con il secondo si procede all'invio di informazioni *una tantum* dal client al server.

La richiesta generata da WM_DDE_REQUEST prevede che nella *low word* di *lParam* venga specificato un formato di dati compatibile con quelli della Clipboard, cioè uno di quelli originali o un formato appositamente registrato mediante la funzione *RegisterClipboardFormat()*. Nella *high word* è precisato l'oggetto della trattazione in corso attraverso l'atomo corrispondente.

```
PostMessage(hwndServer,WM_DDE_REQUEST, (WPARAM)hwndClient, MAKELPARAM(cfFormat, altem));
```

Se il server è in grado di soddisfare la richiesta risponde inviando al client:

```
PostMessage(hwndClient, WM_DDE_DATA, (WPARAM) hwndServer, MAKELPARAM(hData, altem));
```

In caso negativo il server notifica al proprio client l'impossibilità di esaudire quanto richiesto. La logica del protocollo DDE prevede che per ogni invio di un messaggio WM_DDE_DATA da parte del *server* debba corrispondere un messaggio WM_DDE_ACK dal client per confermare l'avvenuta ricezione.

L'invio di un messaggio WM_DDE_ACK da parte del server come risposta ad un precedente WM_DDE_REQUEST può indurre il client a ripetere l'operazione di richiesta specificando però un formato differente. Questo genere di interazione prosegue fino a quando le applicazioni interessate non individuano un formato di dati accessibile ad entrambe. Generalmente un client richiede dapprima i dati nel formato più complesso che è in grado di supportare, accontentandosi via via di livelli inferiori in relazione alla capacità del server. Il formato CF_TEXT è il più semplice e risiede alla base dell'ipotetico elenco dei formati.

Il secondo meccanismo di conversazione su base estemporanea è gestito dal messaggio WM_DDE_POKE. L'applicazione client desidera inviare dei dati al server fornendo un handle a un oggetto della memoria globale e specificando indirettamente il formato dei dati.

```
PostMessage(hwndServer, WM_DDE_POKE, (WPARAM) hwndClient, MAKELPARAM(hData, altem));
```

L'*handle* hData identifica l'oggetto della memoria globale passato al server; la porzione iniziale di quest'area di memoria viene comunque impiegata dal protocollo DDE per definire altri attributi indispensabili per un corretto rapporto tra client e server. In DDE.H è presente la struttura DDEPOKE che assolve allo scopo:

```
typedef struct {
        unsigned unused:13,
                 fRelease:1,
                 fReserved:2;
        int cfFormat;
        BYTE Value[1];
} DDEPOKE;
```

Quando vengono passati degli oggetti tra il server e il client o viceversa, come in questo caso, uno dei problemi da affrontare consiste nello stabilire quale delle due applicazioni deve procedere alla distruzione dell'area di memoria comune. Il bit *fRelease* posto uguale a TRUE stabilisce che deve essere il server a distruggere l'oggetto in memoria dopo averlo utilizzato. Tutti i restanti bit sono riservati. Completano la struttura un intero per indicare il formato con il quale sono descritti i dati e l'inizio vero e proprio dell'oggetto.

Il server deve analizzare molto attentamente il contenuto dei primi byte dell'oggetto trasferito dal client perché parte del suo comportamento ne risulta direttamente influenzata.

23.9 Collegamento permanente

La logica a monte di questo genere di rapporto tra client e server è molto semplice. Il client dapprima notifica al server che vuole essere costantemente aggiornato sul valore di un determinato dato; quest'azione viene condotta con il messaggio WM_DDE_ADVISE. Quando il server riceve questo messaggio deve implementare un proprio comportamento interno che prevede l'invio automatico dei dati al client veicolandoli tramite il messaggio WM_DDE_DATA. Per interrompere un legame permanente il client fa pervenire al server il messaggio WM_DDE_UNADVISE.

Il messaggio WM_DDE_ADVISE contiene in lParam l'atomo dell'item richiesto e un handle a un blocco di memoria al cui interno sono state impacchettate delle informazioni che definiscono nei dettagli le caratteristiche del collegamento. In DDE.H è presente la struttura DDEADVISE appositamente pensata allo scopo:

```
typedef struct {
        unsigned reserved:14,
                 fDeferUpd:1,
                 fAckReq:1;
        int cfFormat;
} DDEADVISE;
```

I due soli campi accessibili di questa struttura sono *fDeferUpd* e *fAckReq* adibiti rispettivamente a notificare al server il comportamento da adottare. con *fAckReq* il client stabilisce se il server deve ritornare subito un messaggio WM_DDE_ACK per far capire al client il livello di accettazione del legame permanente richiesto. Con *fDeferUpd* il client ha la facoltà di attivare due differenti tipi di legame permanente. Assegnando il valore TRUE il client richiede un legame permanente che non obbliga il server a inviare fisicamente i dati quando essi cambiano, mentre con il valore FALSE il server deve inviare i dati oggetto del legame mediante il messaggio WM_DDE_DATA. L'ultimo membro della struttura specifica il formato richiesto dal client per la trattazione dei dati.

Il client invia il messaggio WM_DDE_ADVISE appoggiandosi alla funzione *PostMessage()* dopo aver opportunamente definito la configurazione della struttura DDEADVISE.

```
PostMessage(hwndServer, WM_DDE_ADVISE, (WPARAM) hwndClient, MAKELPARAM(hOptions, aItem));
```

Quando si verificano dei cambiamenti nei dati specificati mediante un messaggio WM_DDE_ADVISE, il server risponde con

```
PostMessage(hwndClient, WM_DDE_DATA, (WPARAM) hwndServer, MAKELPARAM(hData, aItem));
```

aspettandosi a sua volta quasi sempre un messaggio di conferma dell'avvenuta ricezione in funzione di quanto specificato al momento dell'instaurazione di un legame permanente.

Per interrompere un collegamento permanente il client invia un messaggio WM_DDE_UNADVISE al server; questi verifica l'esistenza di un precedente rapporto tra i due inviando un messaggio WM_DDE_ACK positivo qualora esistesse un precedente legame dato dal messaggio WM_DDE_ADVISE.

23.10 Terminare una conversazione

Per porre fine ad una conversazione DDE sia il *server* che il *client* possono inviare un messaggio WM_DDE_TERMINATE:

```
PostMessage(hwndDest, WM_DDE_TERMINATE, (WPARAM) hwndSender, 0L);
```

L'applicazione destinataria deve inviare a sua volta un messaggio WM_DDE_TERMINATE per porre fine alla conversazione.

23.11 La gestione degli oggetti globali

Uno dei problemi principali riguardanti la generazione dei messaggi DDE riguarda la gestione degli oggetti della memoria globale. Nelle transazioni gli oggetti presenti nella memoria condivisa sono utilizzati per individuare fisicamente i dati da trasferire usando WM_DDE_DATA e WM_DDE_POKE, per contenere delle opzioni con WM_DDE_ADVISE o per specificare una stringa di comandi con WM_DDE_EXECUTE e il corrispondente WM_DDE_ACK generato in risposta.

L'allocazione con *GlobalAlloc()* permette di ritornare un handle a una porzione di memoria, trasformato poi in un puntatore tramite *GlobalLock()*. Disponendo del puntatore vengono trasferite nel segmento di memoria le informazioni richieste dalla transazione, provvedendo all'uso di *GlobalUnlock()* immediatamente al termine dell'operazione.

L'applicazione che invia i dati generalmente non prevede alla rimozione dell'oggetto nella memoria, lasciando il compito al ricevente. Il destinatario opera esattamente come il mittente per quanto riguarda *GlobalLock()* e *GlobalUnlock()*, differenziando il proprio comportamento in relazione a quanto specificato nel messaggio ricevuto.

Quindi, quando serve passare delle informazioni dal client al server o viceversa, occorre allocare un blocco di memoria prelevandolo dall'heap globale. Questo oggetto deve essere assolutamente marcato con il flag GMEM_DDESHARE. L'handle ritornato dall'operazione di allocazione è quanto verrà passato al server. Il secondo parametro di *GlobalAlloc()* corrisponde alla dimensione in byte da allocare: deve essere una quantità che comprende anche l'ingombro rappresentato dalla struttura DDEDATA così definita in DDE.H:

```
typedef struct {
        unsigned unused:12,
                 fResponse:1,
                 fRelease:1,
                 reserved:1,
                 fAckReq:1;
        int cfFormat;
        BYTE Value[1];
} DDEDATA;
```

Immaginando di voler inviare una stringa di testo lunga 10 byte, l'allocazione nel server assume il seguente aspetto:

```
hmem = GlobalAlloc(GMEM_DDESHARE , sizeof(DDEDATA) + 10);
```

Nella dimensione della struttura DDEDATA è compreso un byte, il membro *Value[1]*, che dovrebbe corrispondere con l'inizio dell'oggetto vero e proprio da trasferire dal server verso il client. Rispettando questo schema i 10 byte successivi più quello in DDEDATA sono sufficienti per descrivere la stringa di testo e il suo carattere nullo di terminazione.

Il membro *fResponse* impostato a TRUE deve essere interpretato dal client come un obbligo all'invio di un WM_DDE_ACK per confermare l'avvenuta ricezione. Con *fRelease* posto uguale a TRUE il server stabilisce che debba essere il client a distruggere l'oggetto in memoria dopo averlo utilizzato convenientemente. Il membro *fAckReq* consente invece di

risalire alla fonte del messaggio WM_DDE_DATA: il valore TRUE sottintende trattarsi di una risposta ad un WM_DDE_REQUEST, FALSE a WM_DDE_ADVISE. L'intero *cfFormat* definisce il formato dei dati.

23.12 Il messaggio WM_DDE_ACK

Le possibilità di impiego di questo messaggio sono molteplici: in risposta a un WM_DDE_INITIATE, a un WM_DDE_EXECUTE, a un WM_DDE_DATA o a un WM_DDE_ADVISE. Inoltre, il destinatario può essere sia il client che il server in relazione alla situazione.

Lo scenario più complesso è comunque dalla parte del client. Questi deve discriminare la fonte del WM_DDE_ACK ricevuto basandosi esclusivamente sul contenuto di *lParam*. Nel caso di un WM_DDE_ACK proveniente in risposta a un WM_DDE_INITIATE in *lParam* sono presenti due atomi relativi all'applicazione e all'argomento della conversazione.

Un simile test

```
if (lParam == MAKELPARAM(aApp, aTopic))
```

consente di verificare se si tratta della condizione di instaurazione di una conversazione DDE. Più complicate, invece, le altre occasioni. In risposta al messaggio WM_DDE_EXECUTE in *lParam* compare un riferimento ad una struttura di tipo DDEACK e l'handle ad un oggetto in memoria. La struttura DDEACK è così descritta in DDE.H:

```
typedef struct {
        unsigned bAppReturnCode:8,
                 reserved:6,
                 fBusy:1,
                 fAck:1;
} DDEACK;
```

il bit *fAck* viene posto a TRUE quando la richiesta è stata accettata: in questo caso *fBusy* è sempre FALSE. Per convenzione, quando invece *fAck* vale FALSE *fBusy* vale TRUE a indicare l'impossibilità da parte del server di portare a termine il lavoro richiesto dal client in quel particolare momento.

Dato l'elevato grado di variabilità, il membro *bAppReturnCode* è completamente a disposizione del programmatore per contenere dei valori significativi per la propria applicazione. Per comprendere se il messaggio WM_DDE_ACK proviene in conseguenza di un WM_DDE_REQUEST o un WM_DDE_ADVISE è sufficiente verificare che nella high-word di *lParam* sia presente l'atomo relativo all'item trattato:

```
if (HIWORD(lParam) == altem)
```

In tutti i casi ad eccezione di WM_DDE_ACK in risposta a WM_DDE_INITIATE è consigliabile far precedere ai test sulla low-word e la high-word di *lParam* un'analisi del membro *bAppReturnCode* gestito dall'applicazione.

23.13 Il messaggio WM_DDE_EXECUTE

Con questo messaggio il client passa al server una stringa di testo contenente una serie di istruzioni che inducono alcune azioni nel ricevente. La sintassi di questo messaggio prevede nella high-word di lParam un handle ad un blocco di memoria al cui interno l'applicazione client ha inserito la serie di comandi che intende far eseguire al server.