UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

Analisi e sviluppo della comunicazione in ambito IOT con dispositivi di campo di varia natura (PLC, microcontrollori, SCADA industriali).

Analysis and development of IOT communication with various field devices (PLC, microcontrollers, industrial SCADA).

Relatore:

Chiar.mo Prof. GIUSEPPE CONTE

Tesi di Laurea di:

LUCA CROCI

Anno Accademico 2018/2019

Sommario

La tesi proposta tratta l'ingegnerizzazione del progetto TOM (Technology Overlay Method) che l'azienda Bridger s.r.l. ha sviluppato per il bando imposto dalla Regione Marche.
Verranno presentate tutte le caratteristiche di TOM, con un focus sulle parti sviluppate dal sottoscritto durante il periodo di tirocinio in azienda.
Il lavoro svolto riguarda principalmente lo sviluppo di un driver che permettesse la comunicazione tra un dispositivo detto "Gateway" e i dispositivi presenti nel campo di lavorazione (come ad esempio PLC e sensori), che comunicano attraverso il protocollo BACnet.
La restante parte della collaborazione con l'azienda è stata impiegata per sviluppare un controllore PID (con relativo tuning dei suoi coefficienti attraverso il metodo di Ziegler-Nichols) atto a controllare la temperatura della CPU del nostro Gateway attraverso una ventola posizionata al di sopra di essa.
Come ultima parte dello sviluppo del progetto TOM sarà presentata la parte relativa al Cloud, ossia la parte che permette di visualizzare gli andamenti delle variabili fondamentali del sistema di controllo da remoto (accedendo semplicemente ad un indirizzo http), sviluppata in collaborazione con un altro tirocinante (Ing. Fabi Giorgio), con cui ho avuto il piacere di lavorare.

Indice

1) Introduzione.....7
1.1) IoT.....7
1.1.1) Definizione IoT.....7
1.1.2) Come nasce l'IoT.....7
1.1.3) Architettura IoT.....8
1.1.4) Ambiti applicativi.....9
1.1.5) Il successo dell'IoT.....13
1.2) Progetto TOM.....15
1.2.1) Definizione del progetto TOM.....15
1.2.2) Architettura del progetto TOM.....16
1.3) Obiettivo del lavoro di tesi.....18
1.3.1) Architettura del prototipo proposta.....18

2) Protocolli di comunicazione.....25
2.1) Applicazioni delle reti.....25
2.2) Aspetti hardware delle reti.....26
2.3) Tecnologia trasmissiva.....26
2.4) Scala dimensionale.....28
2.5) Tipologie di reti.....28
2.6) Interconnessione di reti.....32
2.7) Protocolli e aspetti software delle reti.....33
2.7.1) Gerarchie di protocollo.....33
2.7.2) Architettura di rete.....34
2.8) Suite di protocolli Internet.....35

3) Protocollo BACnet.....44
3.1) Introduzione al protocollo.....44
3.2) Driver BACnet.....63
3.3) Server BACnet.....75
3.4) Funzionamento del driver BACnet.....79
3.4.1) Salvataggio dei dati.....80
3.4.2) Risultati ottenuti.....82

4) PID con tuning adattivo dei parametri Kp, Ki e Kd.....86
4.1) Introduzione.....86
4.2) Controllore PID.....87
4.3) Azione Proporzionale.....89
4.4) Azione Integrale.....95
4.4.1) Ricalcolo del termine integrale.....102
4.4.2) Integrazione condizionata.....103
4.5) Azione Derivativa.....105
4.6) Metodo di taratura o tuning del PID.....112
4.6.1) Metodo della risposta al gradino (I metodo di Ziegler – Nicols).....112
4.6.2) Metodo della risposta in frequenza (II metodo di Ziegler -Nicols).....115

5) Applicazione per tuning dei parametri del PID.....120
5.1) Definizione dell'applicazione sviluppata.....120
5.2) Hardware utilizzato.....121
5.3) Software Node-RED.....127
5.4) Taratura ad anello chiuso: metodo di Ziegler-Nichols.....132
5.5) Salvataggio dei dati.....133
5.6) Dashboard locale.....135

6) Applicazione in Cloud.....137

7) Analisi dell'architettura sviluppata.....139
7.1) Analisi di performance.....139
7.2) Analisi economica.....140

8) Conclusioni.....142

9) Appendice A.....146
9.1) Codici del driver BACnet.....146

10) Bibliografia.....151

11) Ringraziamenti.....153

1) Introduzione

1.1) IoT (Internet of Things)

1.1.1) Definizione

Internet of Things – letteralmente "Internet delle cose" – è l'espressione utilizzata per definire la rete delle apparecchiature e dei dispositivi, connessi a Internet, equipaggiati con un software che permetta di scambiare dati con altri oggetti connessi. Obiettivi di questi dispositivi sono raccogliere dati e informazioni, dare comandi e controllare particolari attività.
Le due caratteristiche principali di un 'oggetto connesso' sono:

- avere un indirizzo IP che ne consenta l'identificazione univoca sulla Rete;
- avere la capacità di scambiare dati attraverso la rete stessa senza bisogno dell'intervento umano.

Alcuni esempi potrebbero essere i flaconi delle medicine che ci segnalano quando ci si dimentica di prendere un farmaco, occhialini da nuoto che contano vasche, velocità e accelerazione di chi li indossa, etichette che contengono in forma digitale tutte le informazioni sul prodotto e sistemi di riscaldamento intelligente. Non si tratta di episodi sperimentali, ma di un'innovazione che negli ultimi anni ha accelerato il ritmo dello sviluppo: si pensi allo Smart Metering che, in ambito domestico, sta portando le Utilities a sostituire i tradizionali contatori con apparati sensorizzati e controllati da remoto che ci dicono quanto stiamo consumando, in modo da permetterci di razionalizzare i consumi. Monumenti e opere d'arte iniziano a raccontare sé stessi attraverso tag RFId o Qr code, i quali, applicati all'interno dei libri, sui cartelli stradali, sui muri o a fianco dei poster, permettono di accedere a contenuti ulteriori, fruiti attraverso smartphone e altri dispositivi mobili.

1.1.2) Come nasce l'IoT

Nel 1998 i ricercatori dell'Auto-ID Center si misero alla ricerca di un nuovo modo per localizzare e identificare lo spostamento degli oggetti da un luogo fisico all'altro. L'Auto-ID Center ossia "Centro Di Identificazione Automatica" è un'organizzazione di ricerca globale, indipendente e no profit, con sede presso il MIT (Massachusetts Institute of Technology), a cui si deve la nascita della tecnologia RFId.
Proprio i ricercatori di questa organizzazione capirono che l'identificazione automatica, ovvero l'Auto-ID, consentisse di elevare la gestione dei prodotti ad un livello più avanzato, grazie all'introduzione di un sistema di identificazione che permettesse di rintracciare non solo i dati relativi a singoli container, ma persino i dati per singolo articolo. Di Internet of Things si sente parlare per la prima volta nel 1999 durante una presentazione presso Procter & Gamble.

Ad utilizzare questo termine è Kevin Ashton, cofondatore e direttore esecutivo di Auto-ID Center. Da allora si è assistito al grande sviluppo di Internet e alla connessione in rete di miliardi di persone in tutto il mondo, dapprima attraverso l'uso dei computer e più recentemente attraverso device mobili (come smartphone e tablet). Questa rivoluzione della comunicazione si sta ora evolvendo dalle persone alle cose. Nel 2008-2009 nasce ufficialmente l'Internet of Things, non solo come ideologia ma come reale interconnessione tra Internet e gli oggetti, e di conseguenza, tra le persone. Nel 2011 viene lasciato pubblicamente il nuovo protocollo Internet Ipv6, in successione all'Ipv4, che amplia i servizi offerti dalla rete e semplifica la gestione degli indirizzi IP. Nel 2014 Google acquista la start up Nest, che sviluppa il termostato Intelligente. Nel 2015 molte delle grandi aziende che offrono servizi Cloud mettono a disposizione piattaforme per sviluppare l'IoT.

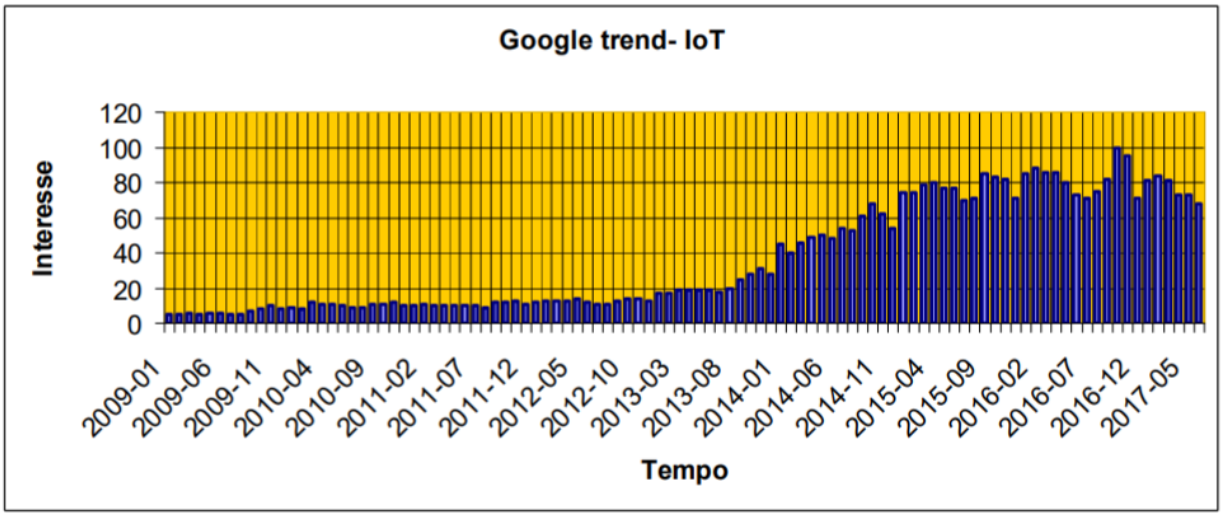


Figura 1.1: Interesse nel tempo dell'IoT, Google Trend.

Ad oggi, secondo Gartner, gli oggetti connessi sono circa 5 miliardi e diventeranno 25 miliardi entro il 2020. Altre fonti stimano che i dispositivi connessi nel globo siano ormai tra gli 8 e i 10 miliardi.

1.1.3) Architetture IoT

I dispositivi, anche detti microcontroller, sempre più connessi, ma dotati di risorse limitate, hanno continuamente bisogno di dialogare fra di loro in maniera sincronizzata, secondo protocolli ben precisi. Raccolgono dati mediante i sensori, li memorizzano in strutture dati, generalmente nel Cloud, dove vengono analizzati per ricavarne informazioni, che vengono reimmesse nel mondo reale ad utilizzo degli umani, secondo gli obiettivi del progettista. Diventano messaggi push su mobile app, grafici su browser, Qr code. L'intero ecosistema di oggetti Hardware/Software che fa parte di un progetto IoT è basato su un'architettura composta da elementi assemblati fra di loro con maestria ed esperienza. Non può esistere una sola architettura nel mondo Internet of Things.

Macroscopicamente bisognerà tener conto di:

- oggetti hardware;
- protocolli di comunicazione;
- servizi cloud;
- applicazioni software.

Gli oggetti hardware possono essere:

- **a corto raggio:** comunicano in rete attraverso Bluetooth o via cavo e quindi necessitano di gateway/hub per la connessione in Internet;
- **a lungo raggio:** si collegano in Internet in modo autonomo, perché possiedono SIM.

Di recente Vodafone ha sviluppato un nuovo protocollo NB-IoT (NarrowBand per l'IoT), studiato per offrire un ampio raggio di copertura, ma imponendo bassi consumi ai componenti radio dei dispositivi collegati alla rete. La rete di Vodafone prevede trasmissioni alla frequenza di 800 MHz, con la possibilità di raggiungere oggetti smart collocati anche all'interno di edifici e nel sottosuolo. Le prime applicazioni indicate riguardano soprattutto l'ambito Smart City, ad esempio il collegamento con i contatori delle utility o i parchimetri. NarrowBand IoT (o NB-IoT) è infatti una tecnologia di connessione a banda stretta che opera in bande soggette a licenza e può essere implementata nelle reti mobili esistenti, garantendo in questo modo un avvio più veloce delle reti rispetto ad altre soluzioni.
Il cloud computing, attraverso le sue caratteristiche più tipiche quali elasticità, scalabilità, potenza ed economicità, sta sostanzialmente abilitando l'IoT. Uno dei motivi del collegamento stretto tra il cloud computing e l'IoT dipende dal fatto che la tecnologia dei dispositivi intelligenti è ancora nella sua fase pionieristica e largamente sviluppata da imprese innovative. L'aspetto principale che rende così interessante il connubio cloud-IoT è sicuramente la quantità di dati che l'IoT è in grado di generare. Questi dati che possono essere analizzati al fine di trarre conclusioni rilevanti sugli utenti, sulle tendenze del mercato e così via. Ed è qui che l'IoT viene naturalmente sorretto dalla potenza e dalla flessibilità del cloud.
Tra gli oggetti fisici e gli utenti finali è presente una Piattaforma IoT, che ha il ruolo di comunicare con gli oggetti e con i software informativi aziendali, gestire le modalità di invio dei dati e di memorizzazione degli stessi, utilizzare algoritmi per estrarre conoscenza dai dati.

1.1.4) Ambiti applicativi

Gli ambiti applicativi si possono suddividere in 3 gradi di maturità:

- applicazioni consolidate,
- applicazioni sperimentali,
- applicazioni embrionali.

Le applicazioni consolidate coincidono con le più semplici, le applicazioni sperimentali, attualmente in fase di verifica, sono quelle che più si avvicinano al paradigma dell'Internet of Things, infine, le applicazioni embrionali sono i progetti per il futuro.

Applicazioni consolidate

Le applicazioni più diffuse e riuscite in Italia sono quelle legate alle soluzioni più semplici e di immediata realizzazione. Si pensi, per esempio, alla videosorveglianza e alla sicurezza nelle smart home (finalizzata al controllo e all'antintrusione o alla gestione delle flotte aziendali), alla tracciabilità degli oggetti di valore, così come al monitoraggio del traffico cittadino in ambito smart city. All'interno di questi ambiti l'Internet of Things italiano cresce sempre più, grazie all'applicazione immediata, di facile utilizzo e di rapida gestione. Tuttavia, queste applicazioni così semplici, limitano fortemente le potenzialità di apertura e raggiungibilità tipiche dell'Internet of Things.
Tuttavia, iniziano a diffondersi lentamente soluzioni più vicine al paradigma Internet of Things. Tra queste, troviamo i contatori intelligenti (Smart Metering) per misurare i consumi, le soluzioni domotiche, la sicurezza delle persone, i servizi di infomobilità e la registrazione dei parametri di guida. Il mercato di queste soluzioni applicative nel nostro Paese procede lentamente, ma, lavorando e ragionando sul valore reale che producono a lungo termine, sarà possibile raggiungere la loro diffusione in breve tempo. Perché questo accada, è necessario che le aziende ridefiniscano le strategie di comunicazione con i potenziali utenti.

Applicazioni sperimentali ed embrionali

Nella seconda fascia rientrano le soluzioni che rispecchiano il concetto di Internet of Things ma che ancora stentano a decollare, soprattutto in Italia. Un esempio sono le soluzioni basate su tecnologie RFId per la supply chain, che sono alla base dell'Internet delle cose. La stessa lentezza si riscontra sulle tecnologie nell'ambito eHealth (IoT per salute e medicina), in cui il tele-monitoraggio dei pazienti potrebbe ridurre drasticamente i costi ospedalieri. Occorrerebbe un forte impegno del soggetto pubblico, che dovrebbe pensare ai vantaggi a lungo termine e agire di conseguenza, finanziando i progetti. Vi sono infine diversi ambiti in cui l'Internet of Things è stato soltanto immaginato. Questo avviene in sperimentazioni di piccola scala e tra queste le più avanzate si trovano in ambito energetico con le Smart Grid, rete elettrica dotata di sensori intelligenti che raccolgono informazioni in tempo reale per ottimizzare la distribuzione di energia. I settori più interessati da applicazioni di IoT sono la Smart Home, lo Smart Building, la Smart City e la Smart Mobility, ma anche, e da molto tempo, lo Smart Manufacturing. Nell'ambito dell'energia è molto diffuso lo Smart Metering (telelettura dei contatori di energia elettrica, acqua e gas) mentre nel mondo della mobilità nuove opportunità sono in arrivo nell'ambito delle Smart Car. L'IoT porta "intelligenza" nei sistemi di elaborazione dell'informazione. Attraverso l'Internet of Things le cose possono essere comandate a distanza (controllo remoto delle cose) e sono capaci di trasmettere dati, dai quali si possono estrarre informazioni utili sul funzionamento di tali oggetti e sull'interazione tra questi oggetti e chi li utilizza (il consumatore). Da qui le critiche relative alla privacy in relazione con l'IoT, e alla trasparenza nel trattamento dei dati personali, oltre che alla sicurezza.

Applicazioni dell'Internet delle cose possono essere individuate nella:

- Domotica (ovvero la tecnologia applicata alle case, per gestire ad esempio frigoriferi, lavatrici, il telefono, etc.);

- Robotica (ovvero ingegneria e tecnologia che permettono ai robot di “prendere vita”, cioè di far fare ai robot compiti oggi svolti dagli esseri umani);
- Avionica (ovvero la tecnologia applicata agli aeromobili ed al pilotaggio, come, ad esempio, sistemi di comunicazione sugli aerei, autopilota, etc.);
- Industria automobilistica (che studia nuove applicazioni per le auto, come, ad esempio. tergicristalli intelligenti che si attivano da soli quando inizia a piovere, fino ad arrivare ad automobili “intelligenti”, le smart car, capaci di guidare da sole ed assistere il guidatore, come sta tentando di fare Apple e Google;
- Industria Biomedicale (ovvero l’IoT applicato alla medicina, come la gestione remota dei pazienti, fino ad arrivare ad interventi chirurgici fatti a distanza);
- Telemetria (che si occupa di sviluppare la trasmissione di dati ed informazioni).

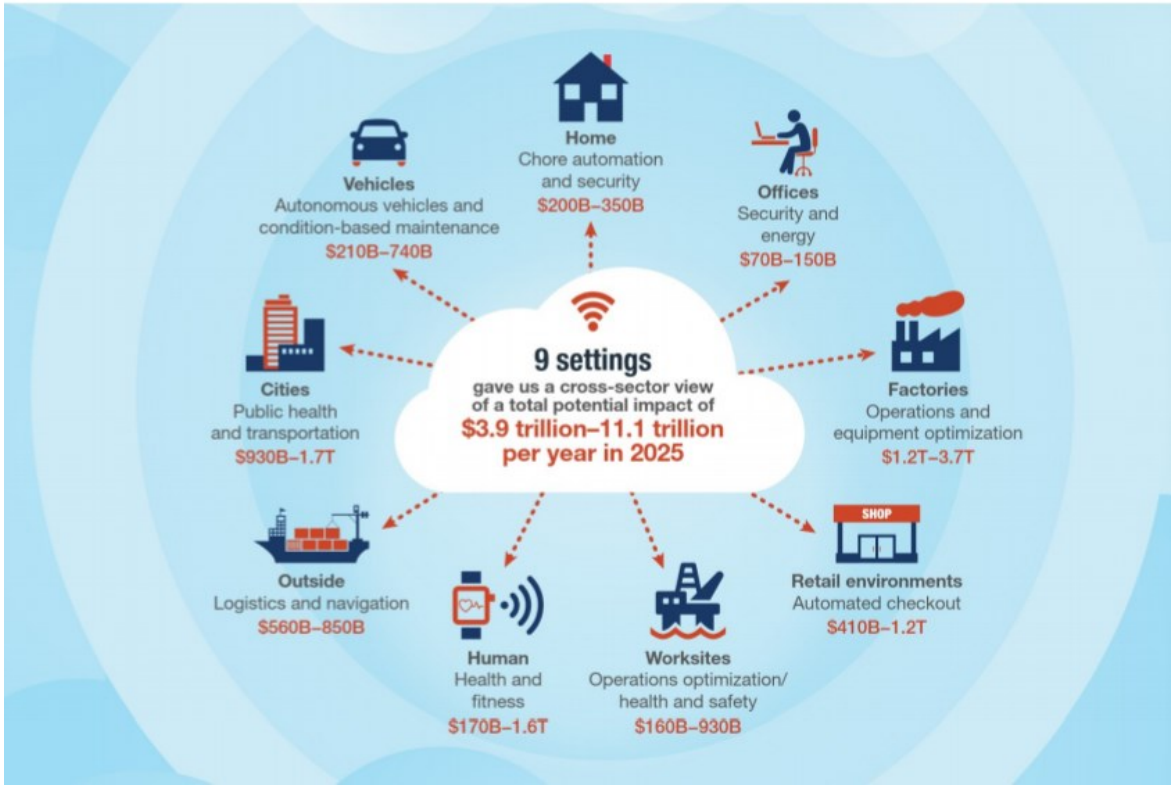


Figura 1.2: ambiti IoT, McKinsey Global Insitute.

A livello aziendale le applicazioni più interessanti sono:

- **Smart Manufacturing o Industria 4.0**
  Lo Smart Manufacturing è stato certamente uno dei precursori del mondo IoT. Applicazioni IoT sono attive da tanti anni, da ben prima che si iniziasse a parlare di Internet delle cose. Oggi questo settore è uno dei più maturi e unisce tematiche legate all’automazione con tematiche legate al mondo della robotica. Lo Smart Manufacturing si sovrappone con il mondo dell’Industry 4.0, vale a dire con una vera e propria politica di sviluppo per estendere l’introduzione del digitale nel mondo dell’industria. Essa è nata in Germania con il fenomeno Industrie 4.0 e ha trovato un suo corrispettivo negli Stati Uniti con il fenomeno della fabbrica digitale. L’Industry 4.0 o Industria 4.0 è una vera e propria realtà, tanto che nel nostro Paese rappresenta un business pari a un miliardo e

200 milioni di Euro nel corso dell'anno 2015. Questi dati sono scaturiti dalla ricerca dell'Osservatorio Smart Manufacturing della School of Management del Politecnico di Milano, la quale evidenzia, anche, come questo mercato stia crescendo ad un ritmo del 20% e rappresenti una spinta concreta nei confronti del Made in Italy.

- **Smart Logistics**
  Sono soluzioni progettate per la tracciabilità di filiera, per la protezione del brand e il monitoraggio della catena del freddo, per la sicurezza in poli logistici complessi e per la gestione delle flotte (tracciabilità del mezzo e delle sue condizioni).

- **Smart Asset Management**
  Consiste nella gestione in remoto di asset di valore (ad esempio dispositivi elettrobiomedicali, vending machine), al fine di rilevare guasti e manomissioni, come pure la localizzazione, la tracciabilità e la gestione inventariale.

- **Smart Factory**
  Implementazione di nuove logiche di gestione della produzione, grazie all'uso di macchine sensibili al contesto in cui operano, in grado di rilevare informazioni in tempo reale, poter comunicare tra loro e prendere decisioni.

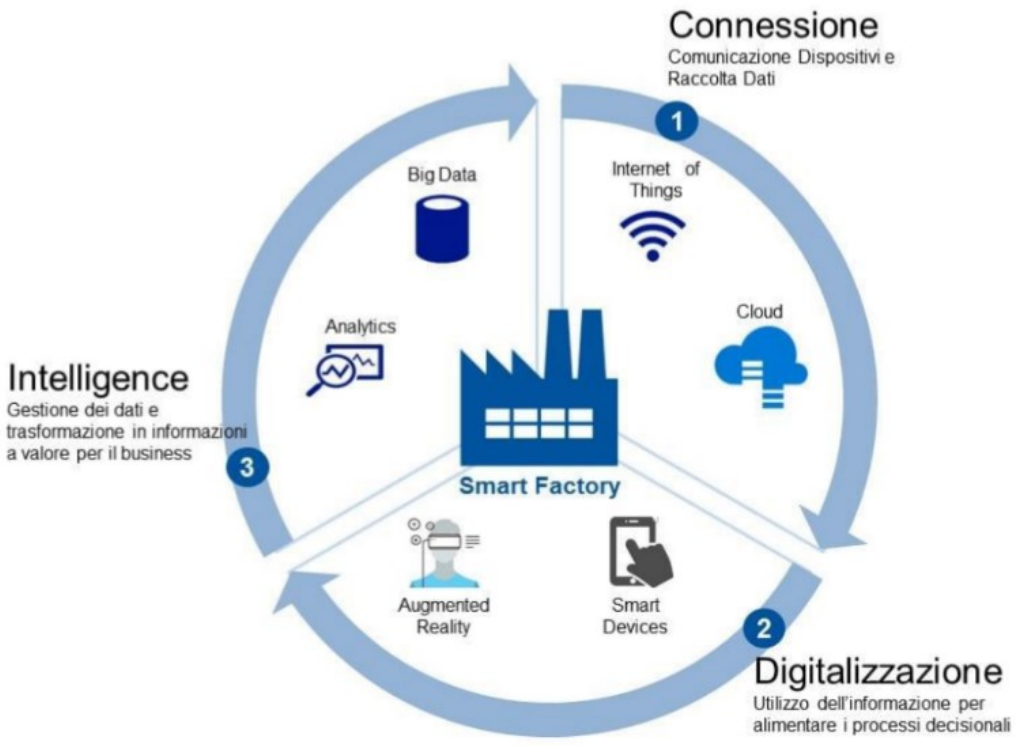


Figura 1.3: La smart factor.

1.1.5) Il successo dell'IoT

Le tecnologie più avanzate (quelle che entrano nelle nostre case, guidano le nostre auto e monitorano il nostro corpo) tendono a conversare con noi attraverso interfacce naturali. Non è sufficiente che la tecnologia funzioni, perché questa deve rientrare nelle abitudini culturali delle persone, idealmente finendo per diventare parte della quotidianità.
La differenza tra l'essere performante e l'essere adottata dagli utenti è determinante per il fallimento o il successo di una tecnologia e questa differenza è lo spazio in cui il design può dare il suo contributo migliore.
Per districarsi tra le tecnologie e portare innovazione nella propria azienda e verso il mercato, la prima regola da seguire è quella di effettuare una traslazione di linguaggio. Con quest' ultimo termine si vuol intendere che le tecnologie non debbono essere comparate o descritte in termini di standard di trasmissione, gigabyte di memoria o teraflops. In tempi recenti, questa operazione di traslazione del linguaggio è stata realizzata in modo mirabile da Steve Jobs il giorno del lancio del primo Ipod.
Apple non ha inventato lo standard di compressione della musica Mp3 né è stata la prima a introdurre sul mercato un lettore musicale digitale tascabile, ma ha introdotto il clickwheel, rotellina touch che permette di scorrere rapidamente le liste testuali su un piccolo display.
Ipod è stato concepito per generare un mercato di acquisto legale della musica, il quale parla in termini di brani e non di memoria, scevro da ogni linguaggio tecnico.

:

Figura 1.6: Primo Ipod.

Un’altra azienda il cui è successo è dovuto alla funzionalità del servizio e alla capacità di semplificare la vita delle persone è Slack, una piattaforma di discussione pensata per il business e cresciuta grazie a un’adozione generata dal basso. Mentre i grandi player provavano a realizzare super piattaforme integrate, una chat ben organizzata ha saputo ingaggiare, divertire, servire, soddisfare gli utenti di mezzo pianeta. Persino Microsoft ha dovuto riprogettare tutta da zero per tentare di adeguarsi a Slack. Ecco quindi cosa determina il successo di queste tecnologie: semplicità, rapidità, praticità e utilità.

Figura 1.7: Brand Identify di Slack.

1.2) Progetto TOM (Technology Overlay Method)

1.2.1) Definizione del progetto TOM

Questo lavoro di tesi nasce dalla collaborazione tra l'Università Politecnica delle Marche e l'impresa Bridger s.r.l., azienda con sede ad Ascoli Piceno, startup innovativa nell'ambito dell'Industria 4.0 e dell'IoT.

TOM avrà moduli che consentiranno:

- elevata capacità di connessione impiantistica. Sarà pertanto necessario sviluppare un numero elevato di driver di comunicazione, per abbracciare un'ampia gamma di dispositivi e controlli;
- raccolta dati da nuove fonti e DB esistenti;
- analisi dei dati, con output di supporto ad ingegneria di manutenzione;
- algoritmi d'intelligenza artificiale, per la prevenzione e la riduzione dei fermi;
- qualifica del servizio di manutenzione;
- regia di controllo centralizzata in ambiente virtuale.

La Nostra attività prevede l'introduzione di processi e tecnologie che vanno a sommarsi agli impianti, agli apparati, e, più in generale, ai processi esistenti, individuando azioni correttive o integrative attraverso supporti hardware, software, analisi e correlazione di eventi esistenti o monitoraggio e introduzioni di nuove misure volte ad aumentare la controllabilità e l'affidabilità dei sistemi.

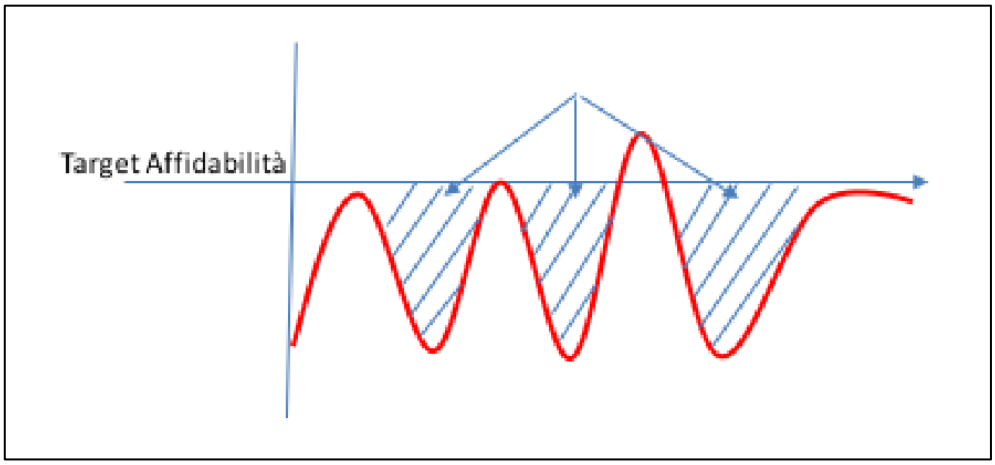


Figura 1.8: Aree di interesse delineate dal Target di Affidabilità.

La nuova piattaforma software, che da qui in avanti chiameremo Technology Overlay Method (TOM), affiancherà i sistemi esistenti, prelevando informazioni da fonti eterogenee, sincronizzandole e storicizzandole, infine, rendendole poi fruibili attraverso qualsiasi DBMS o ERP.
Ciò è stato possibile attraverso una piattaforma grafica che sfrutta la modellazione 3D e la geolocalizzazione, al fine di semplificare e di velocizzare i processi desiderati.

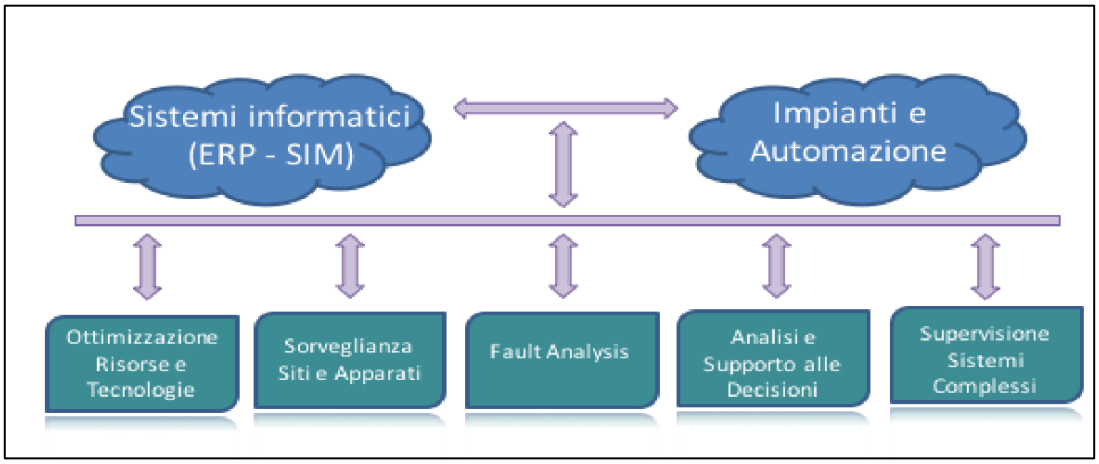


Figura 1.9: Schema a blocchi che costituiscono TOM.

1.2.2) Architettura del progetto TOM

La soluzione si dovrà pertanto integrare con le applicazioni preesistenti e sarà costituita da due componenti software (bordato in rosso) e dalle necessarie customizzazioni specifiche (bordate in blu).

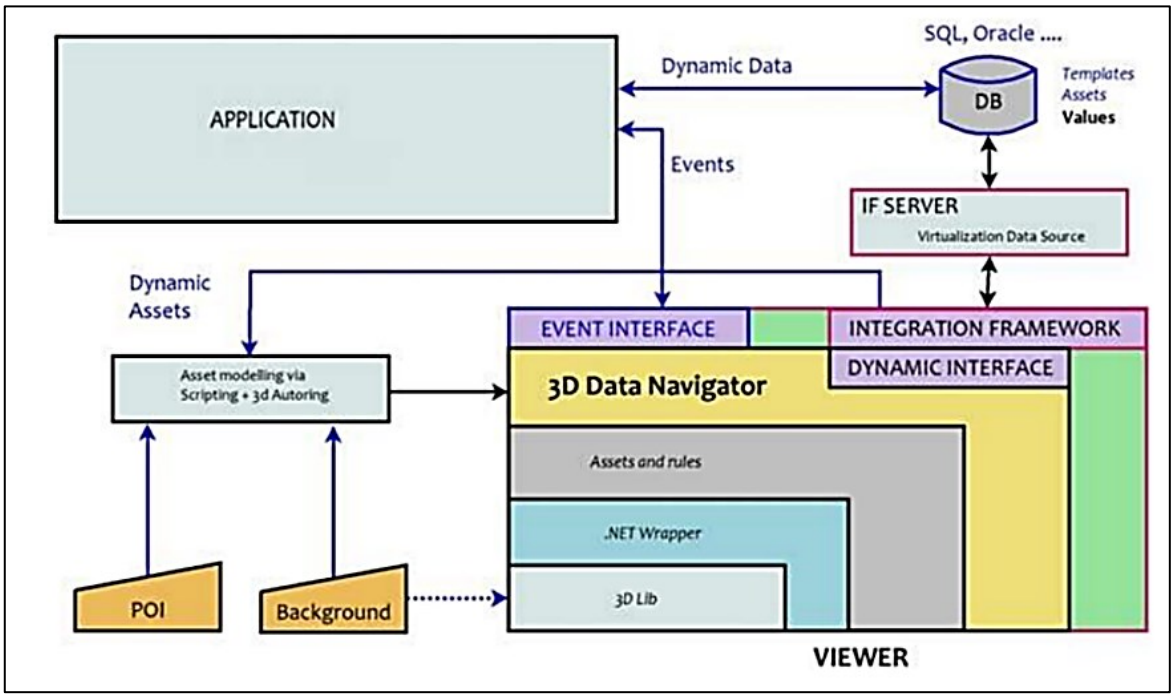


Figura 1.10: Schema integrazione applicazioni preesistenti e custom.

L'architettura TOM presenta una suddivisione geografica delle aree di controllo e un organo decisionale per la supervisione centralizzata, come risulta dalla figura seguente.

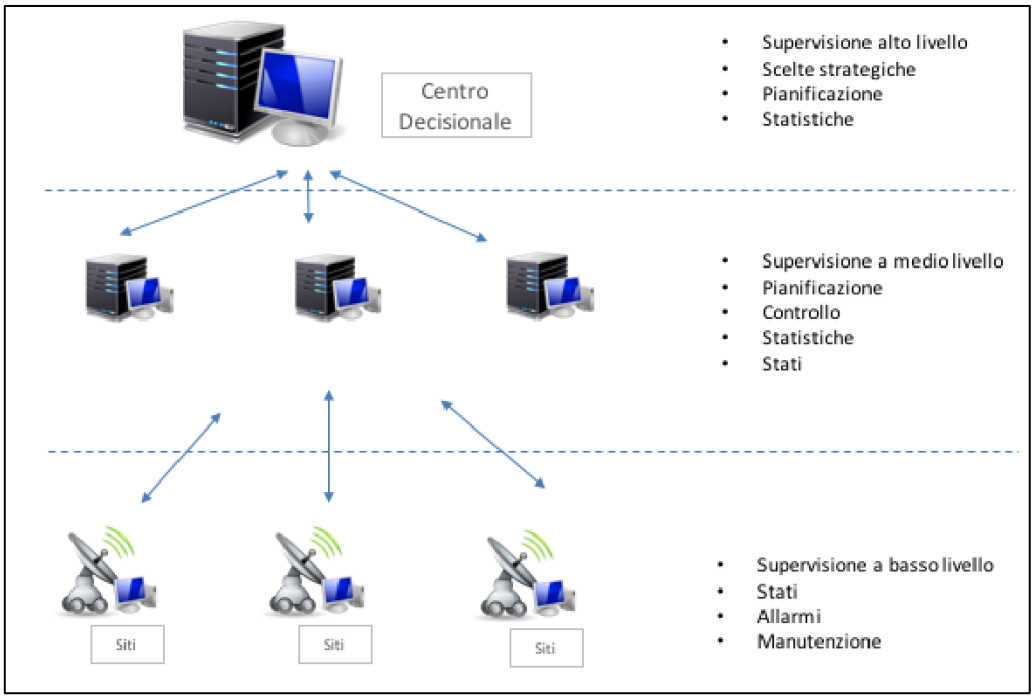


Figura 1.11: Schema suddivisione delle varie aree per la supervisione centralizzata.

Per la realizzazione degli impianti, dei sistemi informatici e di controllo TOM utilizza le più moderne tecnologie del settore. Le informazioni più rilevanti sono rese disponibili attraverso database SQL e/o datalogger.

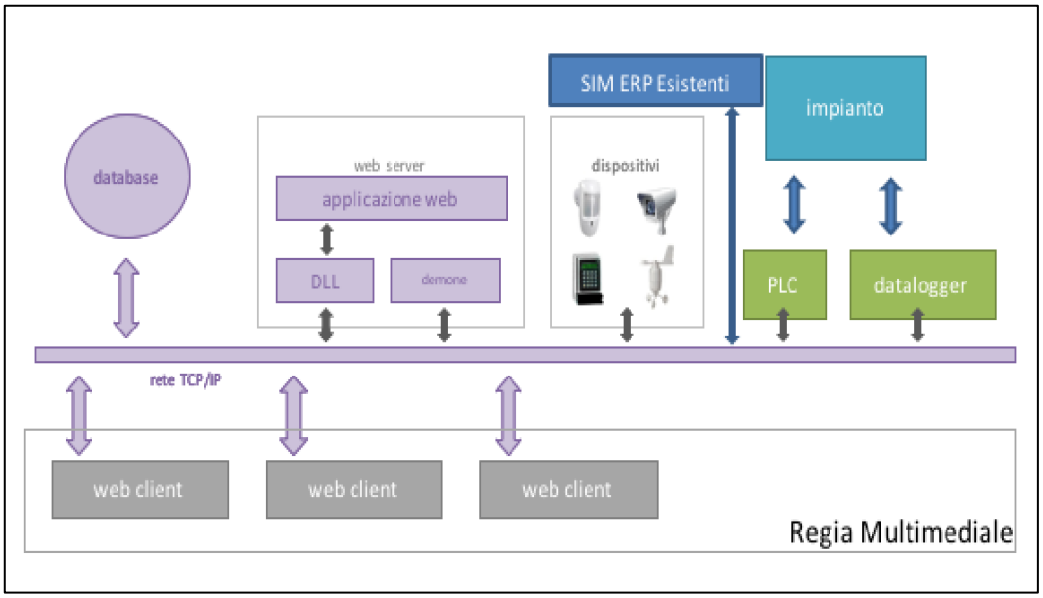


Figura 1.12: Schema rappresentativo delle dipendenze dei dati.

1.3) Obiettivo del lavoro di tesi

1.3.1) Architettura proposta

Il prototipo sviluppato si basa sulla realizzazione di una piattaforma software in grado di prelevare informazioni da fonti eterogenee presenti sul campo. Tali dati saranno in seguito analizzati e resi disponibili su una qualsiasi DBSM oppure ERP.
L'obiettivo è quello di sfruttare tale procedimento, al fine di ottenere un supporto che semplifichi e velocizzi i processi decisionali, riducendo o prevenendo potenziali blocchi di un impianto.

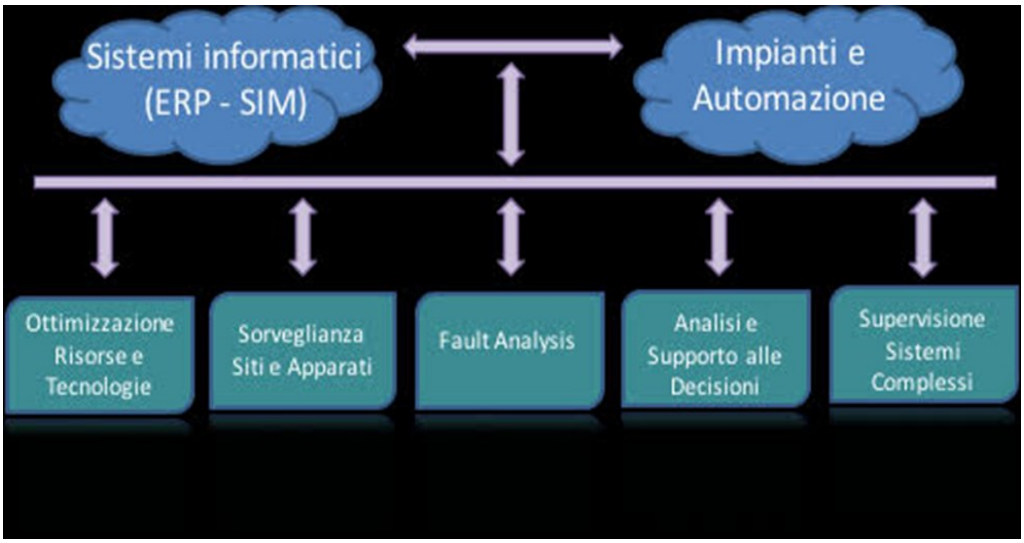


Figura 1.13: Schema rappresentativo del progetto TOM.

Lo schema di tale piattaforma prevede che ci sia alla base un sistema in grado di interfacciarsi con le risorse presenti sul campo, il quale sia utile per l'acquisizione delle informazioni da sottoporre ad analisi. Al termine o durante il processo di acquisizione, si provvede alla sincronizzazione e storicizzazione dei dati, passando poi ad un'analisi degli stessi, per rendere l'informazione leggibile ad alto livello.

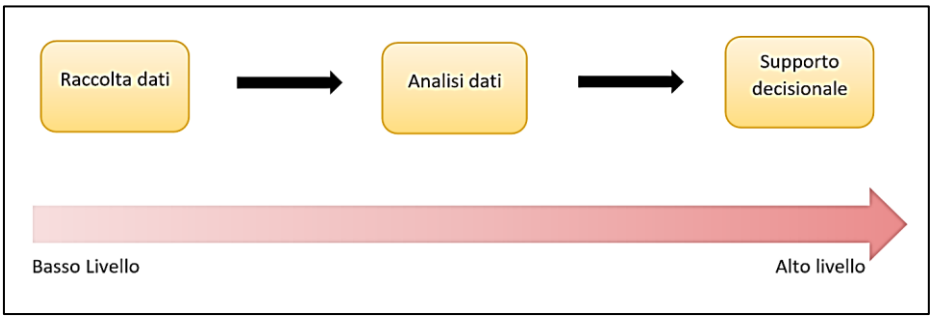


Figura 1.14: Schema a blocchi del progetto TOM.

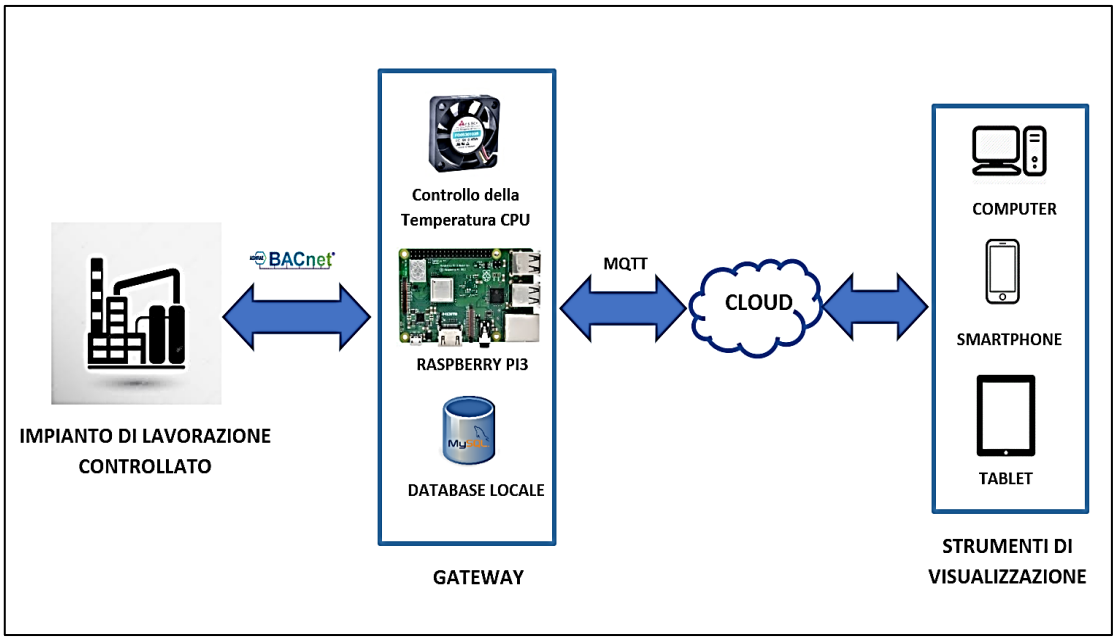


Figura 1.15: Schema del prototipo TOM sviluppato.

In Figura 1.15 si può valutare il prototipo sviluppato dal sottoscritto in fase di tirocinio, costituito essenzialmente da un Gateway, da un protocollo di comunicazione (BACnet), da un Cloud, da un impianto di lavorazione e da strumenti di visualizzazione.
Di seguito andremo a spiegare gli elementi fondamentali del nostro prototipo in modo più specifico:

Gateway:

Funge da "accentratore di dati", cioè ha il compito di acquisire dati dai dispositivi di campo che comunicano tramite protocollo BACnet, per immagazzinarli in un database locale MySQL.
All'interno del Gateway, che nel nostro caso applicativo è rappresentato da una Raspberry PI3 Model B+, è stato implementato un controllore PID tramite il software Node-RED.

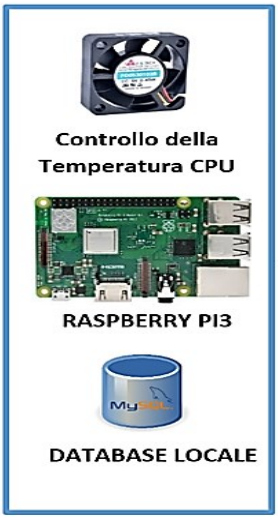


Figura 1.16: Elementi che costituiscono il Gateway.

Il PID, il cui tuning dei parametri avviene tramite la teoria del II metodo di Ziegler-Nicols, ha il compito di regolare la temperatura della CPU della nostra Raspberry mantenendo tale valore il più possibile vicino al valore di riferimento (set-point).

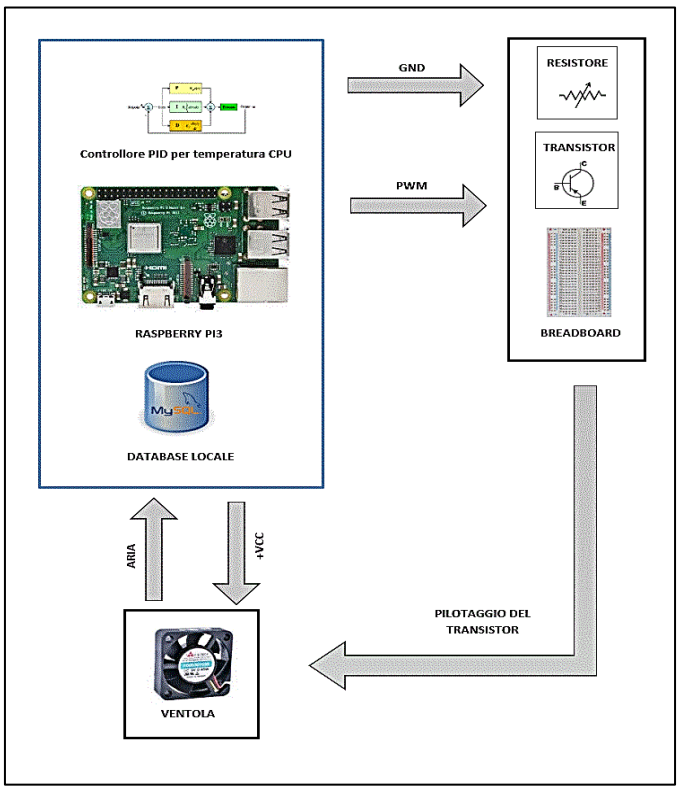


Figura 1.17: Schema dell'applicazione del controllore PID.

Nel Capitolo 5 verranno spiegati in dettaglio gli elementi hardware e software presenti negli schemi raffigurati in figura 1.16 e 1.17.

La raccolta dei dati viene gestita dal Gateway (Raspberry PI3), con l'obiettivo di salvare sul database (locale al Gateway) i valori significativi degli oggetti del processo controllato.
Il salvataggio dei dati nel database MySQL avviene in due tabelle differenti, a seconda degli intervalli di tempo d'inserimento:

1- la prima tabella sarà popolata con un intervallo di tempo pari al secondo, questo per permettere il salvataggio di tutti i parametri fondamentali in un periodo di tempo pari alle 8-12 ore (periodo di storicizzazione), con la possibilità di poter valutare eventuali variazioni significative dei parametri ogni secondo.
   Inoltre, i dati di questa tabella saranno inviati al gestore dei dati presente nel Cloud, utilizzati per la visualizzazione tramite dashboard;
2- la seconda tabella, invece, avrà un periodo d'inserimento più ampio rispetto alla prima, pari a 10 minuti.
   L'utilità di questa seconda tabella risulta essere quella di avere un set di dati meno preciso, dato che vengono storicizzati ogni 10 minuti, ma con il vantaggio di coprire un periodo di storicizzazione maggiore, pari a circa 2-3 settimane.

In fase di ingegnerizzazione si è passati da un Gateway Siemens IoT2040 ad una Raspberry PI3 (con sistema operativo Raspbian). Avendo quest'ultima un sistema operativo più aperto, permette test più avanzati e particolari rispetto alla scheda della Siemens, che vincola l'utente ai soli dispositivi Siemens.
Un altro motivo di particolare interesse risulta essere la differenza del costo dei due Gateway. Infatti, la Raspberry PI3 Model B+ ha un costo minore (circa 1/4) rispetto alla scheda Siemens IoT2040.

Protocollo di comunicazione:

La Raspberry comunica con il processo controllato, costituito da dispositivi che permettono la comunicazione tramite protocollo BACnet, utilizzando il driver sviluppato in fase di tirocinio e spiegato dettagliatamente nel Capitolo 3.
La fase di sviluppo del driver è stata effettuata utilizzando in un primo tempo un simulatore di device BACnet, con lo scopo di non compromettere l'integrità di dispositivi più costosi. Successivamente, sono stati svolti dei test su uno dei dispositivi di campo maggiormente utilizzato in applicazioni automatiche, ossia il PLC della Beckhoff CX5130, come risulta visibile in figura 1.18.

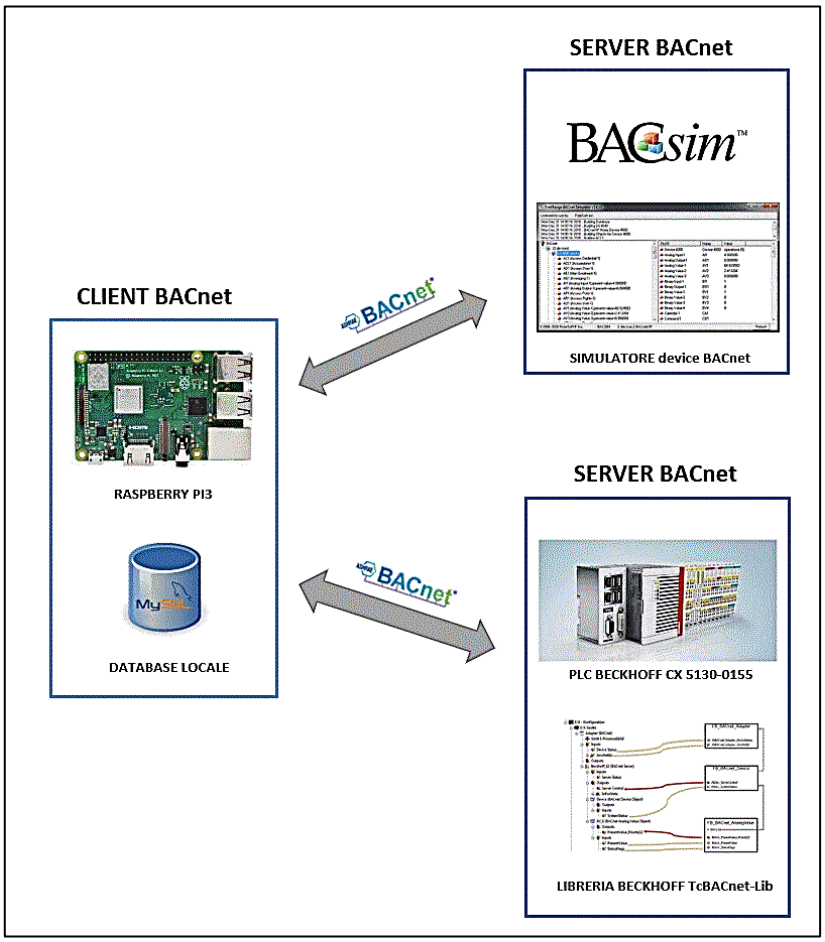


Figura 1.18: Schema dei dispositivi utilizzati come device BACnet.

Cloud:

Il Cloud rappresenta un dispositivo di mediazione tra il Gateway e l'utente che ha interesse nel valutare gli andamenti dei parametri caratteristici del sistema di lavorazione, accedendo tramite qualsiasi dispositivo (PC, Smartphone, Tablet) all'indirizzo in cui è configurato il Cloud.
Il nostro Cloud è supportato dai servizi Amazon ed è stato sviluppato in collaborazione con il tirocinante Ing. Fabi Giorgio.

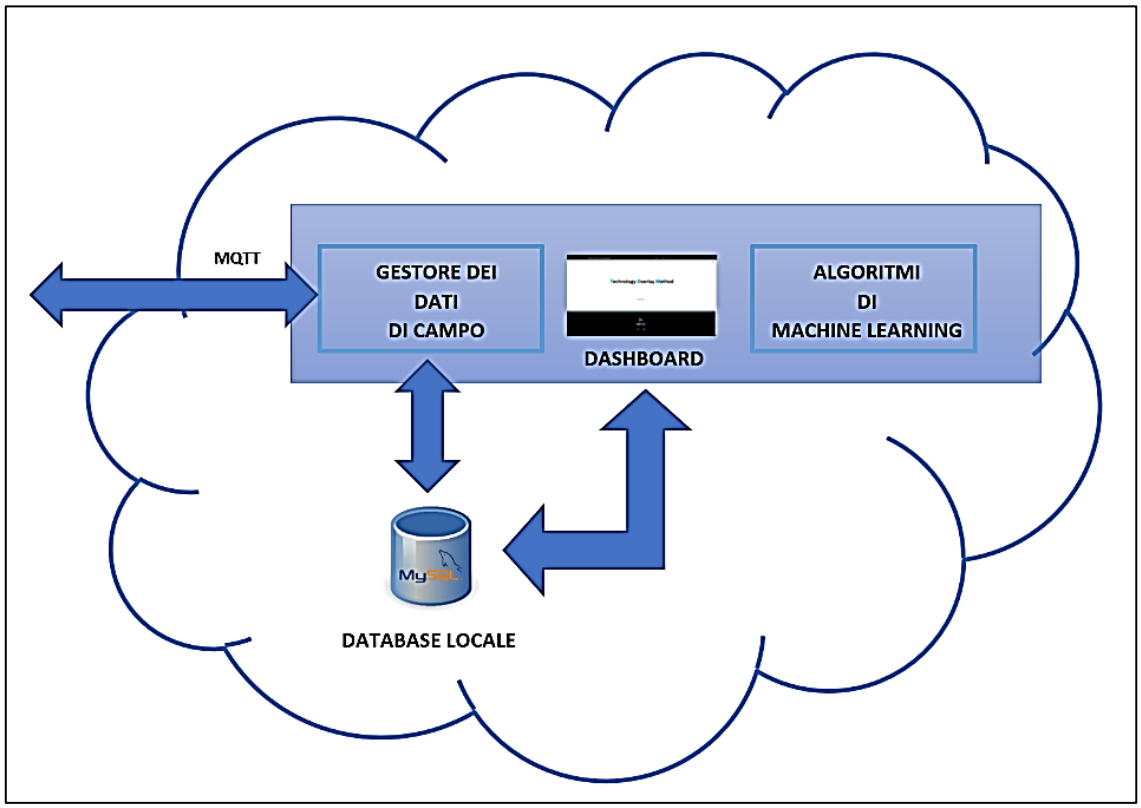


Figura 1.19: Schema degli elementi che costituiscono il Cloud.

Il Cloud, visibile in figura 1.19, è costituito da un "gestore dei dati di campo", che ha il compito di comunicare con il Gateway tramite protocollo MQTT e di aggiornare i dati presenti nel suo database locale (che risulta essere più capiente).
La dashboard permette di visualizzare gli andamenti delle variabili fondamentali di lavorazione e permette di scegliere il periodo di tempo in cui l'utente vuole valutare tali parametri.
Infine, vi sono algoritmi di Machine Learning, ancora in fase di sviluppo, con l'intento di ottenere una predizione su eventuali guasti, valutando tali set di dati prelevati dal Gateway.

Nel Capitolo 3 verrà definito lo sviluppo del protocollo di comunicazione BACnet e dopo un'introduzione sulla teoria del controllore PID (Capitolo 4), sarà presentata, nel Capitolo 5, l'applicazione utilizzata per il tuning dei parametri del PID.
Infine, nel Capitolo 6, verrà spiegata in breve la struttura del Cloud, e nel Capitolo 7, andremo ad effettuare delle valutazioni sui costi dell'architettura proposta.

2) Protocolli di comunicazione

Il ‘900 è stato, dal punto di vista della strumentazione tecnologica, un susseguirsi di successi. Principalmente nell’ ultimo cinquantennio, l’attenzione degli studiosi si è rivolta a nuove metodologie di informazione, che possono essere sintetizzate in tre punti:

- raccolta e memorizzazione;
- elaborazione;
- distribuzione.

Nel nostro secolo si sono via via diffusi:

- il sistema telefonico, a livello mondiale;
- la radio e la televisione;
- il computer;
- i satelliti per telecomunicazioni.

Queste tecnologie stanno rapidamente convergendo. In particolare, la combinazione di elaboratori e sistemi di telecomunicazione ha avuto una profonda influenza sull’organizzazione dei sistemi di calcolo. Infatti, si è passati dal vecchio modello mainframe – terminali, in cui la potenza di calcolo è concentrata in un unico grande elaboratore, a cui si accede per mezzo di un certo numero di terminali, a quello attuale, in cui vi è un grande numero di elaboratori autonomi, interconnessi fra loro. Per elaboratori autonomi si vuole intendere che non deve esserci fra loro stessi una relazione tipo master/slave (ad es., l’uno non può forzare lo spegnimento dell’altro), bensì interconnessione, cioè capacità di scambiarsi informazioni (sfruttando un opportuno mezzo fisico).
Un sistema di calcolo siffatto è detto rete di elaboratori o computer network. Rete di elaboratori non è sinonimo di sistema distribuito. Infatti, in un sistema distribuito l’esistenza di più elaboratori è invisibile all’utente, che ha l’impressione di avere a che fare con un unico sistema di calcolo. In una rete di elaboratori, l’utente è conscio dell’esistenza di molteplici elaboratori, che devono essere esplicitamente riferiti.

2.1) Applicazioni delle reti

Moltissimi sono gli usi delle reti di elaboratori, sia per le organizzazioni che per i singoli individui. Per le aziende e le associazioni, si possono sintetizzare in questi piccoli punti:

- **condivisione di risorse:** si possono rendere disponibili a qualsiasi programma le informazioni anche distanti migliaia di km;
- **affidabilità:** si ottiene mettendo in rete sorgenti alternative delle risorse (ad es. duplicando le applicazioni e i dati su più computer). È importante in sistemi che devono funzionare a tutti i costi (traffico aereo, centrali nucleari, sistemi militari, ecc.);
- **diminuzione dei costi:** una rete di personal computer costa molto meno di un mainframe. A volte, alcuni elaboratori sono più potenti ed offrono agli altri dei servizi (modello Client-Server, vedi figura 2.1);

- **scalabilità:** si possono aumentare le prestazioni del sistema aumentando il numero di elaboratori (entro certi limiti);
- **comunicazione fra persone:** è possibile inviarsi messaggi, scambiarsi file, ecc..
  Per i singoli individui (che, di solito, si connettono dalla propria casa tramite "fornitori di accesso") essa viene usata per l'accesso ad informazioni remote, (ad es.: accesso a servizi bancari, acquisti da casa; navigazione sul World Wide Web), oppure per la comunicazione fra persone (posta elettronica);

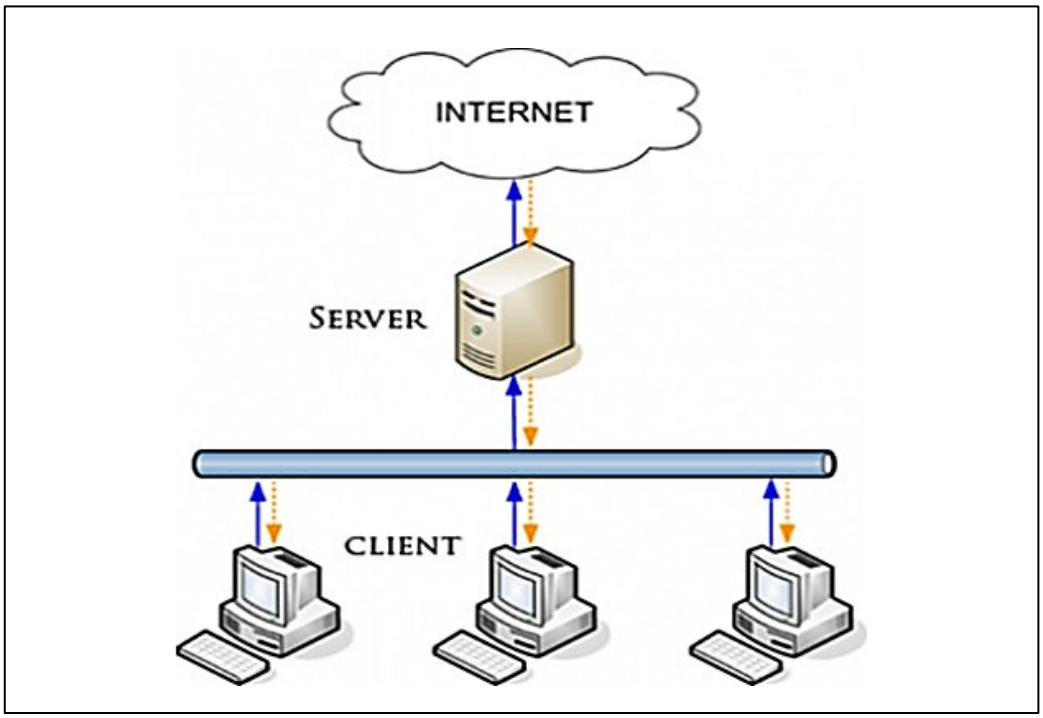


Figura 2.1: Modello Client-Server.

- **divertimento:** video on demand (selezione e ricezione via rete di un qualunque spettacolo tratto da un catalogo); giochi interattivi (contro macchine o avversari umani).

2.2) Aspetti hardware delle reti

I due parametri utili per definire le caratteristiche di una rete, anche se non vi è una tassonomia universalmente accettata, sono: la **tecnologia trasmissiva** e la **scala dimensionale**.

2.3) Tecnologia trasmissiva

Esistono due tipologie per quanto riguarda la tecnologia trasmissiva: le reti broadcast e le reti punto a punto. Una rete broadcast è caratterizzata da processi di comunicazione punto-multipunto, in cui i singoli nodi sono connessi tramite lo stesso supporto trasmissivo. Se sono presenti supporti trasmissivi di tipo diverso, occorre che alcuni nodi svolgano la funzione di gateway (repeater o bridge), rigenerando l'informazione in modo da essere propagata sul nuovo supporto.

In alcuni casi è possibile che si voglia impedire tale funzionalità e quindi il gateway sarà sostituito da un router configurato per filtrare il traffico (firewall).
Se gli elementi della rete utilizzano supporti trasmissivi condivisi, il principale problema da affrontare è la gestione dei conflitti nell'uso della rete, per cui esistono diverse tecniche di gestione e di risoluzione (aloha, csma, csma/cd).

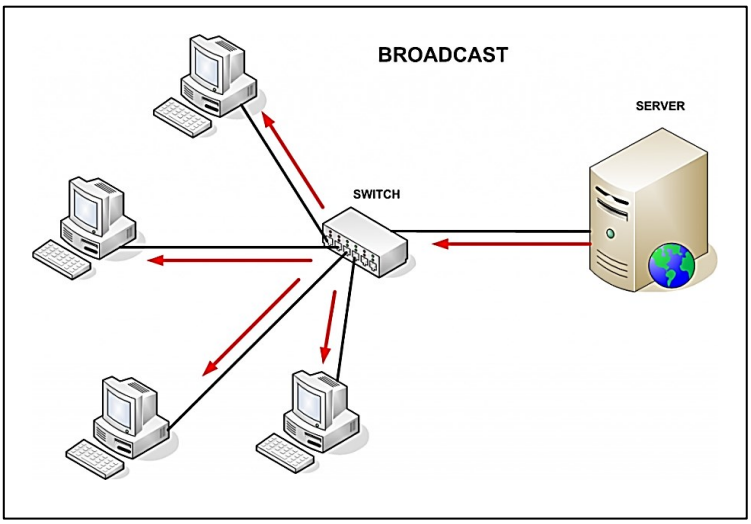


Figura 2.2: Rete broadcast.

Le reti broadcast sono dotate di un unico "canale" di comunicazione, che è condiviso da tutti gli elaboratori. Brevi messaggi, spesso chiamati pacchetti, inviati da un elaboratore, sono ricevuti da tutti gli altri elaboratori. Un indirizzo all'interno del pacchetto specifica il destinatario.
Quando un elaboratore riceve un pacchetto, esamina l'indirizzo di destinazione. Se questo coincide col proprio indirizzo, il pacchetto viene elaborato, altrimenti viene ignorato. Le reti broadcast, in genere, consentono anche di inviare un pacchetto a tutti gli altri elaboratori, usando un opportuno indirizzo (broadcasting). Tutti prendono in considerazione il pacchetto.
Un'altra possibilità è inviare il pacchetto ad un sottoinsieme degli elaboratori (multicasting). In tal caso solo gli elaboratori di tale sottoinsieme lo prendono in considerazione, mentre gli altri lo ignorano.
Le reti punto a punto consistono invece in un insieme di connessioni fra coppie di elaboratori.

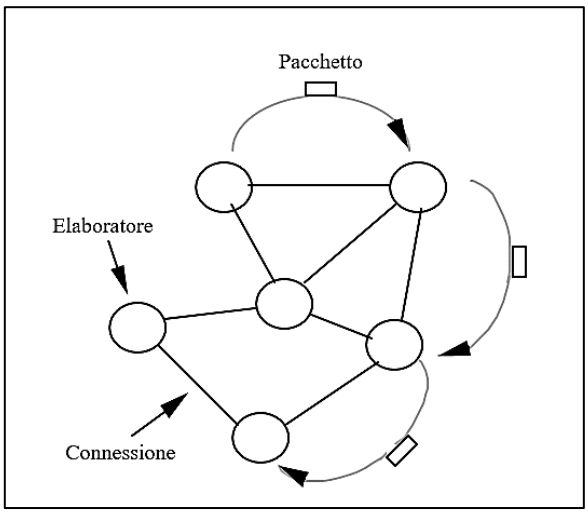


Figura 2.3: Rete punto a punto.

Per arrivare dalla sorgente alla destinazione, un pacchetto può attraversare uno o più elaboratori o nodi intermedi.
Spesso esistono più cammini alternativi, per cui gli algoritmi di instradamento (routing) hanno un ruolo molto importante. In generale, le reti geograficamente localizzate tendono ad essere broadcast, mentre le reti geograficamente molto estese tendono ad essere punto a punto. Esistono però alcune eccezioni, come, per esempio, la rete geografica realizzata via satellite (e quindi broadcast) ed infine la rete locale basata su ATM (e quindi punto a punto).

2.4) Scala dimensionale

Un criterio alternativo di classificazione è la scala dimensionale delle reti. In questo contesto si distinguono: reti locali, reti metropolitane e reti geografiche.

Distanza fra processori	**Ambito**	**Tipo di rete**
10 m.	Stanza	Rete locale
100 m.	Edificio	Rete locale
1 km.	Campus	Rete locale
10 km.	Città	Rete metropolitana
100 km.	Nazione	Rete geografica
1000 km.	Continente	Rete geografica
10.000 km.	Pianeta	Internet (Rete geografica)

La distanza è un fattore molto importante, poiché a differenti scale dimensionali corrispondono differenti tecniche di utilizzo.

2.5) Tipologie di reti

Reti locali

Le reti locali (**L**ocal **A**rea **N**etwork, LAN) sono possedute, in genere, da una organizzazione (reti private), hanno un'estensione che arriva fino a qualche km, si distendono nell'ambito di un singolo edificio o campus (non si possono, di norma, posare cavi sul suolo pubblico), sono utilizzate prevalentemente per connettere PC o workstation. Esse si distinguono dagli altri tipi di rete per tre caratteristiche diverse:

- la prima è la dimensione, che non può andare oltre un certo limite, per cui è noto a priori il tempo e quindi il ritardo di trasmissione. Questa conoscenza permette di utilizzare delle tecniche particolari per la gestione del canale di comunicazione;

- la seconda è la tecnologia trasmissiva, perché le LAN sono in generale reti broadcast. Velocità di trasmissione tipiche sono da 10,100,1000 Mbps con basso ritardo di propagazione del segnale da un capo all'altro del canale e basso tasso di errore;
- la terza, ma non meno importante, è la topologia. Sono possibili diverse topologie: le più diffuse, alcuni anni fa, erano il bus ed il ring. Nell'ambito della topologia bus può trasmettere in ogni istante solo un elaboratore, mentre gli altri devono astenersi. Inoltre, quando due o più elaboratori vogliono trasmettere contemporaneamente, è necessario, per risolvere i conflitti, utilizzare un meccanismo di arbitraggio, che può essere centralizzato o distribuito. Lo standard IEEE 802.3 (chiamato impropriamente Ethernet) è adottato per una rete broadcast, la quale è basata su un bus, con arbitraggio distribuito, operante a 10,100 oppure 1000 Mbps. Nella topologia ring ogni bit circumnaviga l'anello in un tempo tipicamente inferiore a quello di trasmissione di un pacchetto. Anche qui è necessario un meccanismo di arbitraggio, spesso basato sul possesso di un gettone (token) che abilita alla trasmissione. Lo standard IEEE 802.5 (derivante dalla rete IBM Token Ring) è una rete broadcast basata su ring, con arbitraggio distribuito, operante a 4 o 16 Mbps.

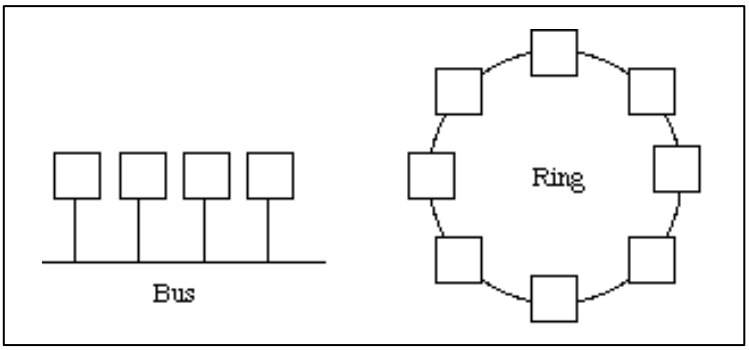


Figura 2.4: Tipologie Bus e Ring.

Infine, le reti broadcast possono essere classificate, a loro volta, in due sottogruppi, a seconda del meccanismo scelto per l'arbitraggio. Il primo meccanismo è denominato allocazione statica: le regole per decidere chi sarà il prossimo a trasmettere sono fissate a priori, ad esempio assegnando un time slot ad ogni elaboratore con un algoritmo round-robin. Lo svantaggio è rappresentato dallo spreco dei time slot assegnati a stazioni che non devono trasmettere. Il secondo meccanismo è denominato allocazione dinamica: in tal caso, si decide di volta in volta chi sarà il prossimo a trasmettere. È necessario un meccanismo di arbitraggio delle contese, che può essere un arbitraggio centralizzato, con un'apposita apparecchiatura che accetta richieste di trasmissione e decide chi abilitare, oppure un arbitraggio distribuito, in cui ognuno decide per conto proprio.

Reti metropolitane

Le reti metropolitane (**M**etropolitan **A**rea **N**etwork, MAN) hanno un'estensione tipicamente urbana (quindi anche molto superiore a quella di una LAN) e sono generalmente pubbliche (cioè un'azienda, ad es. Telecom Italia, Infostrada mettono la rete a disposizione di chiunque desideri, previo pagamento di una relativa tariffa). Fino a qualche anno fa, erano basate essenzialmente sulle tecnologie delle reti geografiche, utilizzate su scala urbana. Recentemente,

però, è stato definito un apposito standard, lo IEEE 802.6 o DQDB (Distributed Queue Dual Bus), che è effettivamente impiegato in varie realizzazioni, molto più vicino alla tecnologia LAN che WAN. Esiste un mezzo trasmissivo di tipo broadcast (due bus in 802.6), a cui tutti i computer sono attaccati.

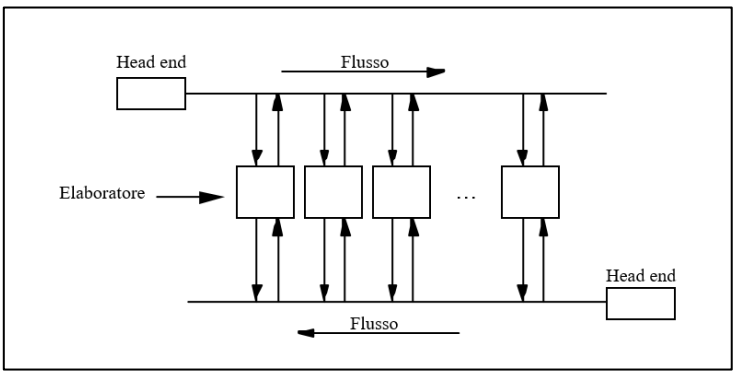


Figura 2.5: Doppio bus della coda distribuita.

Reti geografiche

Le reti geografiche (**W**ide **A**rea **N**etwork, WAN) si estendono a livello di una nazione, di un continente o dell'intero pianeta. Una WAN è tipicamente costituita da due componenti distinte:

- un insieme di elaboratori (host oppure end system), sui quali girano i programmi usati dagli utenti
- una comunication subnet (o subnet), che connette gli end system fra loro.

Il compito della WAN è quello di trasportare messaggi da un end system all'altro, così come il sistema telefonico trasporta parole da chi parla a chi ascolta. Di norma la componente subnet è costituita, a sua volta, da due elementi: linee di trasmissione (dette anche circuiti, canali, trunk) e elementi di commutazione (switching-element). Gli elementi di commutazione sono elaboratori specializzati, che vengono utilizzati per connettere fra loro due o più linee di trasmissione. Quando arrivano dati su una linea, l'elemento di commutazione deve scegliere una linea in uscita sul quale instradarli. Non esiste una terminologia standard per identificare gli elementi di commutazione. I termini usati sono sistemi intermedi, nodi di commutazione, pacchetti o router. Una tipica WAN è utilizzata per connettere più LAN fra loro, come si evidenzia nella figura successiva;

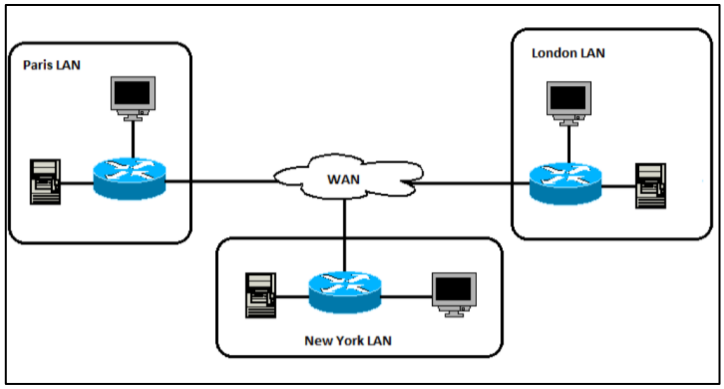


Figura 2.6: Struttura tipica della WAN.

In generale una WAN contiene numerose linee (spesso telefoniche), che congiungono coppie di router. Ogni router, in generale, deve: ricevere un pacchetto da una linea in ingresso, memorizzarlo per intero in un buffer interno e, infine, appena la necessaria linea in uscita è libera, instradare il pacchetto su essa. Una subnet basata su questo principio può essere chiamata: punto a punto, store and forward o a commutazione di pacchetto (packet switched).

Molte topologie di interconnessione possono essere impiegate fra i router:

- a stella (ridondanza zero);
- ad anello (ridondanza zero);
- a stella estesa o ad albero (ridondanza zero);
- a stella estesa (ridondanza media);
- completamente connessa o a maglia (ridondanza massima).

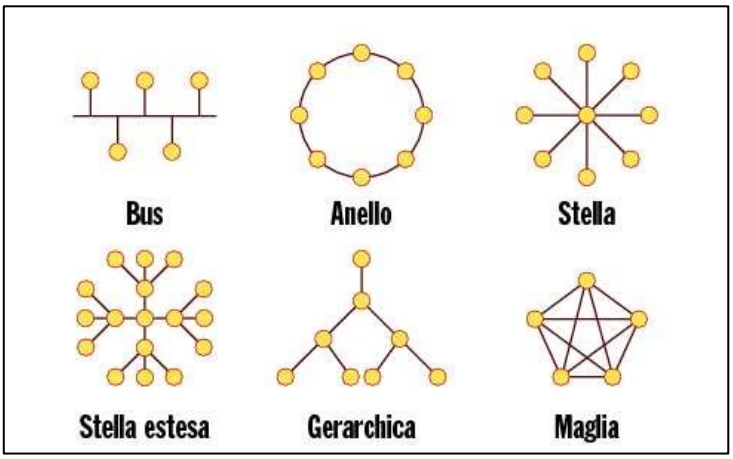


Figura 2.7: tipologie di interconnessione

Un'altra possibilità è una WAN basata su satellite oppure radio al suolo. Utilizzando il satellite, ogni router sente l'output del satellite e invia a sua volta dati al satellite. Dunque, in generale si ha un broadcast downlink (cioè dal satellite a terra) e un broadcast uplink (cioè da terra al satellite).

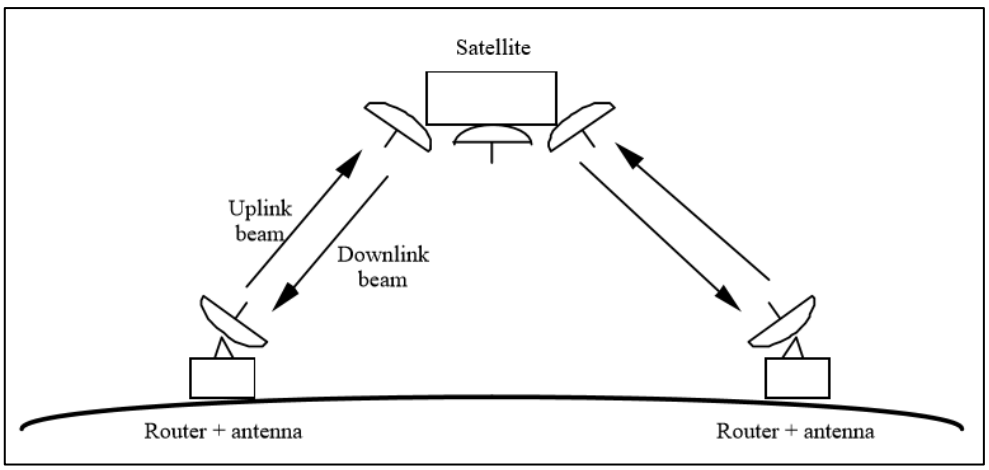


Figura 2.8: Interconnessione di router via satellite.

Una WAN può essere anche realizzata in maniera mista: in parte cablata, in parte basata su radio o satellite.

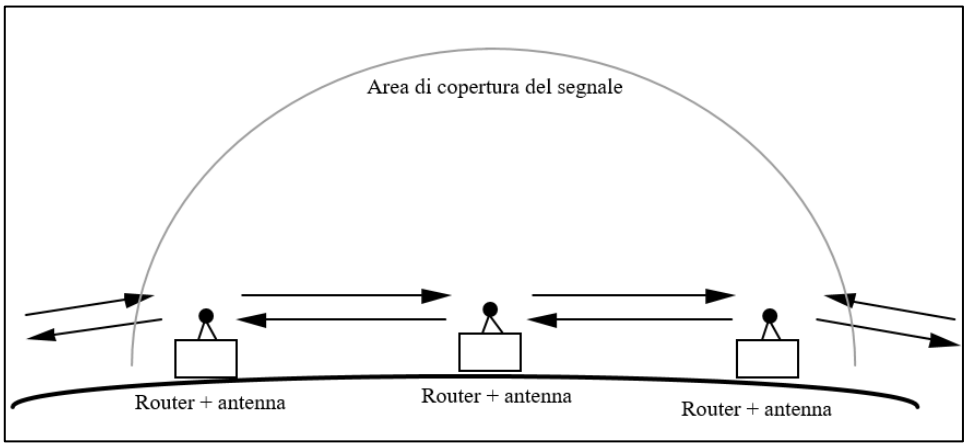


Figura 2.9: Interconnessione di router via radio al suolo.

2.6) Interconnessione di reti

Una interconnessione di reti, talvolta anche chiamata internetwork, è formata quando reti diverse (sia LAN che MAN o WAN) sono collegate fra loro. A prima vista, almeno in alcuni casi, la cosa è apparentemente uguale alla definizione di WAN vista precedentemente (se al posto di subnet si scrive WAN, abbiamo una internetwork costituita da una WAN e quattro LAN). Alcuni problemi, però, sorgono quando si vogliono connettere fra di loro reti progettualmente diverse (spesso incompatibili fra loro). In questo caso, si deve ricorrere a speciali attrezzature, dette gateway (o router multiprotocollo), che, oltre ad instradare i pacchetti da una rete all'altra, effettuano le operazioni necessarie per rendere possibili tali trasferimenti.

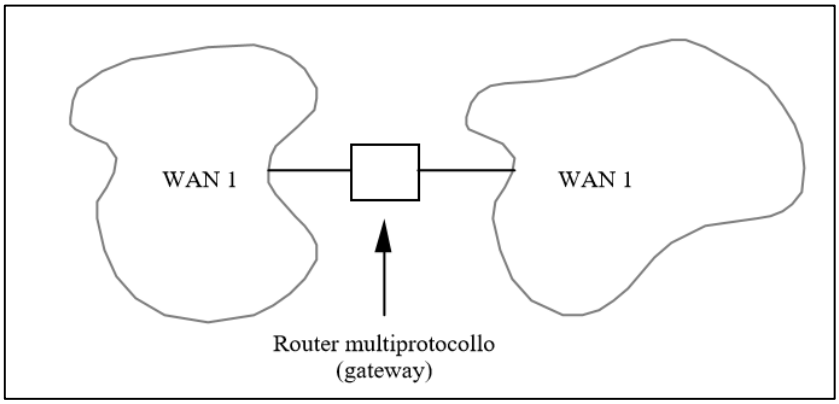


Figura 2.10: Interconnessione di rete.

In questo contesto, e concretamente, è utilizzata questa terminologia: internet come sinonimo di internetwork, cioè la interconnessione di più reti generiche; ed Internet (con la I maiuscola) per riferirsi alla specifica internetwork, basata su TCP/IP, che ormai tutti conoscono.
L' uso della terminologia non è sempre preciso ed univoco.

Si indica:

- con sottorete (subnet): l'insieme dei router e delle linee di trasmissione;
- con rete (network): una subnet più tutti gli host collegati;
- con internetwork: una collezione di più network, anche non omogenee, collegate per mezzo di gateway.

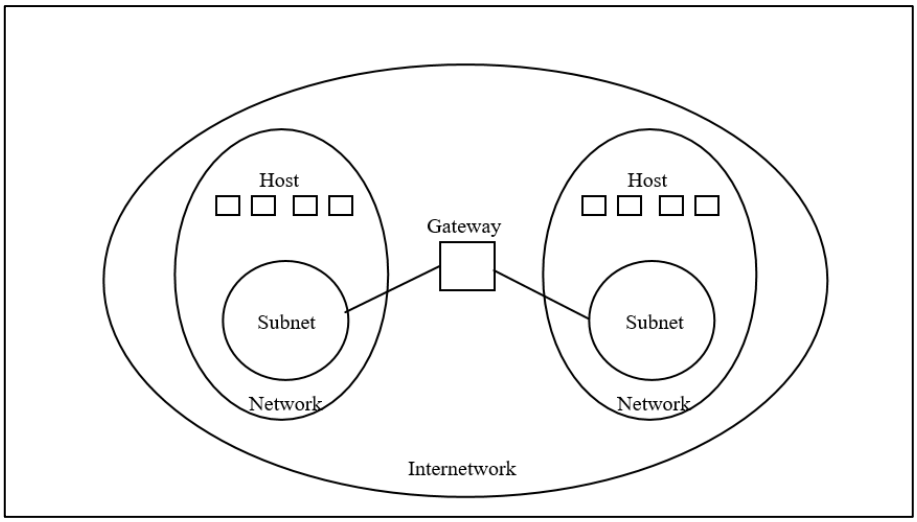


Figura 2.11: Relazione tra subnet, network e internetwork.

2.7) Protocolli e aspetti software delle reti

Le prime reti furono progettate cominciando dall'hardware e sviluppando il software solo successivamente, quasi come se esso fosse un'appendice accessoria dell'hardware. Questo approccio non può più funzionare. Il SW di rete è oggi altamente strutturato. Vogliamo, ora, esaminare, a grandi linee, tale strutturazione, che servirà come base per introdurre il lavoro svolto.

2.7.1) Gerarchie di protocollo

Per ridurre la complessità di progetto, le reti sono in generale organizzate a livelli, ciascuno costruito sopra il precedente. Fra un tipo di rete ed un'altra possono essere diversi il numero, i nomi, il contenuto e le funzioni dei livelli. Comunque, è sempre rispettato il principio generale secondo il quale lo scopo di un livello è offrire certi servizi ai livelli più alti, nascondendo i dettagli sul come tali servizi siano implementati.
La figura che segue definisce questi concetti, in particolare:

- il livello n su un host porta avanti una conversazione col livello n su di un altro host;
- le regole e le convenzioni che governano la comunicazione sono collettivamente indicate col termine di protocollo;
- le entità (processi) che effettuano tale conversazione si chiamano peer entity (entità di pari livello).

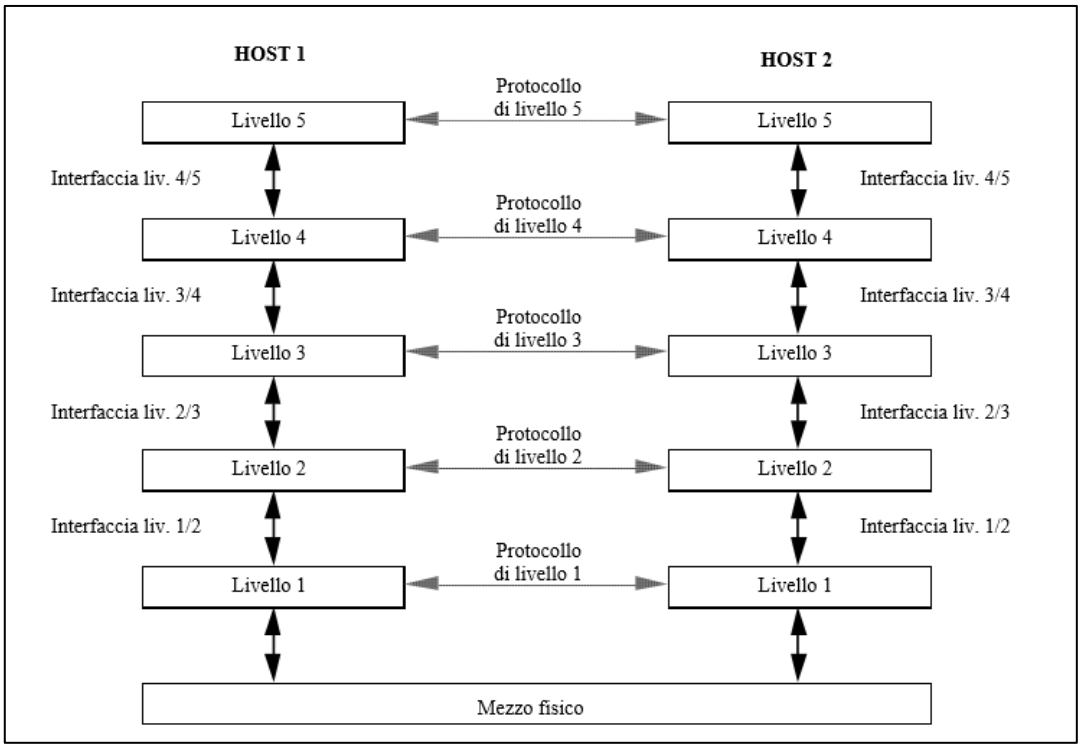


Figura 2.12: Dialogo fra peer entity.

In realtà non c'è un trasferimento diretto dal livello n di host 1 al livello n di host 2. Ogni livello di host 1 passa i dati, assieme a delle informazioni di controllo, al livello sottostante. Al di sotto del livello 1 c'è il mezzo fisico, attraverso il quale i dati vengono trasferiti da host 1 ad host 2. Quando arrivano a host 2, i dati vengono passati da ogni livello (a partire dal livello 1) a quello superiore, fino a raggiungere il livello n. Fra ogni coppia di livelli adiacenti è definita una interfaccia, che caratterizza sia le operazioni primitive (che possono essere richieste al livello sottostante) sia i servizi (che possono essere offerti dal livello sotto di esso). I vantaggi di una buona progettazione delle interfacce sono la minimizzazione delle informazioni da trasferire e la possibilità di modificare l'implementazione del livello (ad es., ove le linee telefoniche venissero sostituite da canali satellitari), con una più attuale che offra gli stessi servizi.

2.7.2) Architettura di rete

L'insieme dei livelli e dei relativi protocolli è detto architettura di rete. La specifica dell'architettura deve essere abbastanza dettagliata da consentire la realizzazione di SW e/o HW che, per ogni livello, rispetti il relativo protocollo. Viceversa, i dettagli implementativi di ogni livello e le interfacce fra livelli non sono parte dell'architettura, in quanto sono nascosti all'interno di un singolo host.
È possibile, quindi, che sui vari host della rete ci siano implementazioni che differiscono fra di loro anche in termini di interfacce fra livelli, purché ogni host implementi correttamente i protocolli previsti dall'architettura. In questo caso, possono dialogare fra loro anche host aventi caratteristiche diverse (processore, sistema operativo, costruttore).
Dunque, nell'ambito di una specifica architettura di rete, tutti gli host devono contenere implementazioni conformi in termini di livelli e di protocolli. Un'architettura di rete può essere: proprietaria, standard de facto, standard de iure. Un'architettura proprietaria è basata su scelte

indipendenti ed arbitrarie del costruttore ed è generalmente incompatibile con architetture diverse. Nel senso più stretto del termine, è un'architettura della quale il costruttore non rende pubbliche le specifiche, per cui nessun altro può produrre apparati compatibili. Alcuni esempi sono IBM SNA (System Network Architecture), Digital Decnet Phase IV; Novell IPX e Appletalk. Un'architettura standard de facto è un'architettura basata su specifiche di pubblico dominio (per cui diversi costruttori possono proporre la propria implementazione), che ha conosciuto una larghissima diffusione. L'esempio lampante di questa architettura è l'Internet Protocol Suite (detta anche architettura TCP/IP). Un'architettura standard de iure è un'architettura basata su specifiche (ovviamente di pubblico dominio) approvate da Enti internazionali che si occupano di standardizzazione. Anche in questo caso ogni costruttore può proporne una propria implementazione. Nello specifico, possono essere considerate standard de iure standard IEEE 802 per le reti locali, architettura OSI (Open Systems Interconnection) e Decnet Phase V (conforme allo standard OSI). L'insieme dei protocolli utilizzati su un host e relativi ad una specifica architettura di rete viene denominato pila di protocolli (protocol stack). Si noti che un host può avere contemporaneamente attive più pile di protocolli per eventualmente comunicare con host differenti.

2.8) Suite di protocolli Internet

In informatica e in telecomunicazioni, la **suite di protocolli Internet** indica una famiglia di protocolli di rete legati da dipendenze d'uso su cui si basa il funzionamento logico della rete Internet. A volte, per sineddoche, è chiamata suite di protocolli TCP/IP, in funzione dei due più importanti protocolli in essa definiti: il Transmission Control Protocol (TCP) e l'Internet Protocol (IP).
Il rispettivo modello di architettura di rete a strati rappresenta lo standard *de facto* nell'ambito delle reti, in contrapposizione allo standard *de iure,* rappresentato, invece, dal modello ISO/OSI.

Modello TCP/IP

Nei primi anni settanta del XX secolo, la Defense Advanced Research Project Agency (DARPA) finanziò l'Università di Stanford e la BBN (Bolt, Beranek and Newman) per lo sviluppo di un insieme di protocolli di comunicazione, da utilizzarsi per lo sviluppo di reti a commutazione di pacchetto e per l'interconnessione di calcolatori eterogenei.
Fu così che nacque l'Internet Protocol Suite i cui due protocolli più noti sono il TCP (**T**ransmission **C**ontrol **P**rotocol) e l'IP (**I**nternet **P**rotocol).
I creatori di tali protocolli di trasmissione, tuttora utilizzati nel web, sono nello specifico Robert Kahn e Vinton Cerf, a cui l'ex Presidente degli Stati Uniti George W. Bush ha consegnato il 9 novembre 2005.la Presidential Medal of Freedom, ovvero la più alta tra le onorificenze civili a stelle e strisce.
Cerf e Kahn hanno sviluppato lo standard per la trasmissione di pacchetti via rete nel 1973 mentre lavoravano ad un progetto di sviluppo dei sistemi di comunicazione voluto dalla DARPA (Defense Advanced Research Projects Agency). Attualmente, Vinton Cerf collabora con Google alla creazione degli standard per le future applicazioni e nel frattempo si

dedica allo sviluppo di nuovi protocolli di comunicazione interplanetaria per il Jet Propulsion Lab della NASA. Robert Kahn, invece, dopo 13 anni di servizio presso la DARPA è diventato presidente della Corporation for National Research Initiatives(CNRI).
Questi protocolli, utilizzabili gratuitamente da tutti perché di pubblico dominio fin dall'inizio, ottennero un elevato successo (utilizzati da un gruppo di ricercatori per ARPAnet).
Questo genera alcune ambiguità, dovute al fatto che il nome più corretto sarebbe Internet Protocol Suite. Per esempio, succede di sentir parlare di servizi basati su TCP/IP anche quando, in realtà, invece di TCP viene usato un protocollo alternativo, UDP, anch'esso appartenente all'Internet Protocol Suite. In genere, il TCP viene utilizzato per quelle applicazioni che richiedono un servizio orientato alla connessione, come, ad esempio, la posta elettronica e il file sharing, mentre l'UDP prende sempre più piede per le applicazioni in tempo reale, come l'on-line gaming o lo streaming audio e video. La differenza fra i due protocolli risiede nella maggiore affidabilità nel trasporto dei dati di TCP, che offre una serie di servizi appositamente pensati (gestione del flusso, della congestione, ecc.), mentre UDP punta molto sulla velocità di trasmissione, a scapito della affidabilità. Si tenga quindi sempre presente che la sigla TCP/IP è di utilizzo talmente comune da essere utilizzata, talvolta, anche quando esistono termini alternativi più corretti.
L'architettura adottata dalla rete Internet è il TCP/IP, che, quindi, si è imposto come standard de facto proprio grazie alla diffusione e ai massicci investimenti legati a Internet.

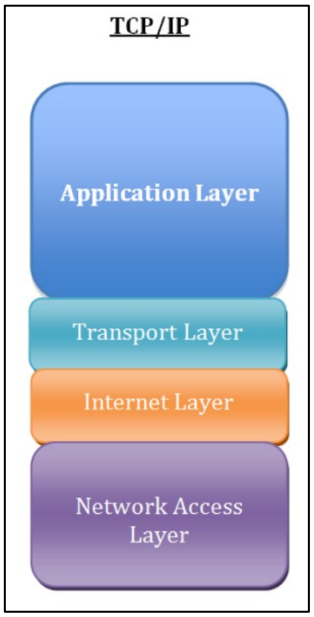


Figura 2.13: Layers del modello TCP/IP.

I livelli che costituiscono il modello TCP/IP sono, partendo dall'alto:

- livello di Applicazione (Application Layer);
- livello di Trasporto (Transport Layer);
- livello di Rete (Internet Layer);
- livello di Accesso alla Rete (Networ Access Layer).

Essendo importante la distinzione fra livello di collegamento e livello fisico, è possibile utilizzare un modello ibrido, che distingue i due strati e porta il numero di livelli a cinque, come di seguito indicato.

- **Application Layer:** il primo livello è quello dell'applicazione. Esso rappresenta l'interfaccia con l'utente ed abilita, ad esempio, la consultazione di pagine web, stabilendo e gestendo le sessioni di lavoro dei processi client tra il nostro browser ed il server web.
  Il protocollo di trasporto TCP mette in coda i messaggi generati da client e server e li trasmette sotto forma di pacchetti su una connessione full-duplex. Il buon fine della spedizione è attestato da una ricevuta di ritorno o riscontro. Anche questo è un collegamento virtuale tra le due applicazioni, i cui dettagli sono demandati al successivo livello, detto di trasporto.

- **Transport Layer:** il livello di trasporto offre un servizio al livello delle applicazioni, avvalendosi dei servizi del sottostante livello di rete (ed in particolare dell'Internet Protocol).
  Per gestire molteplici processi attivi nel trasferimento dati sul medesimo nodo (o computer), cioè più sessioni di navigazione attive, il livello di trasporto (TCP o UDP) utilizza più numeri di porta.
  TCP nell'invio dei pacchetti usa il meccanismo dello Sliding Window (o finestra scorrevole). Una serie di pacchetti viene inviata da TCP seguendo delle regole ben precise:
  - ad ogni finestra di pacchetti spedita il trasmettitore fa partire un timeout;
  - il ricevitore invia per ogni pacchetto ricevuto un ACK, indicando il successivo pacchetto atteso;
  - il trasmettitore considera quindi spediti tutti i pacchetti precedenti;
  - se il timeout scade oppure sono stati ricevuti 3 ACK duplicati, TCP presume che si sia verificata la perdita di uno o più pacchetti e provvede ad implementare opportune strategie di ritrasmissione dei dati e di controllo della congestione.

  Questa è una tecnicamolto importante perché fornisce un canale di comunicazione affidabile. Inoltre, TCP contiene meccanismi per gestire la congestione ed il controllo di flusso.

- **Internet Layer:** Internet Protocol (IP) è il protocollo di InterNetworking del modello DOD/DARPA (secondo il modello OSI è classificato nel livello rete). Esso si occupa di gestire l'indirizzamento dei nodi e l'instradamento. A ciascun nodo viene infatti assegnato un indirizzo IP che lo identificherà in modo non ambiguo in rete. Le funzionalità di instradamento, invece, consentono di selezionare il percorso migliore per veicolare un messaggio verso un dato nodo destinatario, il cui indirizzo IP sia noto.

- **Network Access Layer:** sotto il livello Internet, il modello di riferimento TCP/IP specifica solo che ci deve essere un livello di accesso alla rete in grado di spedire i pacchetti IP. Nel modello ISO/OSI questo strato corrisponde ai primi due livelli: il livello di collegamento e il livello fisico.
  Al livello di collegamento si decide come fare il trasferimento del messaggio per ogni singolo tratto del percorso: dal computer del web browser al primo router, dal primo router al secondo, dal secondo al terzo e dal terzo al computer del server. Questo è un collegamento virtuale tra due computer (o router) adiacenti. Anche in questo caso le interfacce di comunicazione dei nodi adiacenti saranno individuate per mezzo di un indirizzo univoco, usualmente denominato indirizzo MAC.

Il livello fisico trasmette il messaggio sul canale di comunicazione usualmente sotto forma di segnali elettrici o elettromagnetici, sebbene sia anche possibile utilizzare onde acustiche (come ad esempio nelle reti di sensori sottomarine).

Modello ISO/OSI

Il modello OSI (acronimo di **O**pen **S**ystems **I**nterconnection, conosciuto anche come modello ISO/OSI), in telecomunicazioni e informatica, è uno standard per le funzioni di comunicazione delle telecomunicazioni o di sistemi informatici stabilito nel 1984 dall'International Organization for Standardization (ISO), il principale ente di standardizzazione internazionale. Esso stabilisce per l'architettura logica di rete una struttura a strati composta da una pila di protocolli di comunicazione di rete suddivisa in 7 livelli, i quali insieme eseguono tutte le funzionalità della rete, seguendo un modello logico-gerarchico.
L'organizzazione sentì la necessità di produrre una serie standard per le reti di calcolatori ed avviò il progetto OSI, un modello standard di riferimento, per l'interconnessione di sistemi di computer. Il documento che illustra tale attività è il *Basic Reference Model* di OSI, noto come standard ISO 7498. A livello implementativo, lo standard *de facto* affermatosi per architetture di rete a livelli è il TCP/IP, che riprende in parte il modello OSI.
Il modello ISO/OSI, concepito per reti di telecomunicazioni a commutazione di pacchetto, è costituito da una pila (o stack) di protocolli attraverso i quali viene ridotta la complessità implementativa di un sistema di comunicazione per il networking.

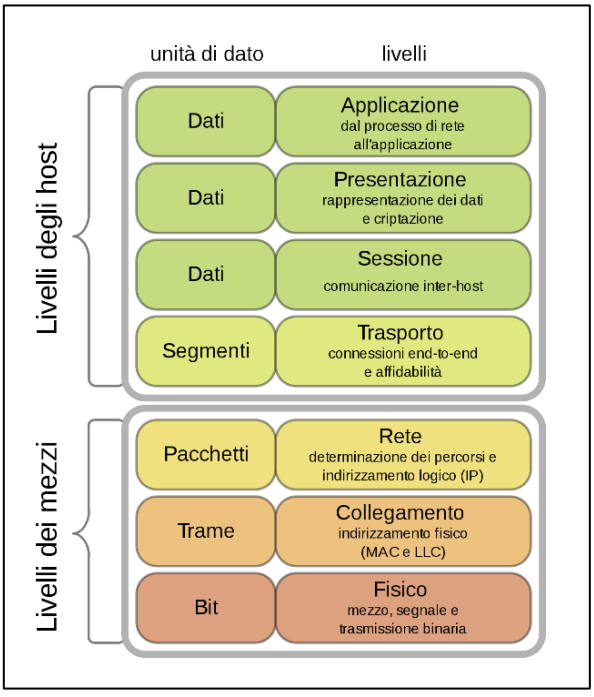


Figura 2.14: Modello di riferimento ISO/OSI

In particolare ISO/OSI è costituito da strati (o livelli), i cosiddetti layer, che definiscono e racchiudono in sé a livello logico uno o più aspetti fra loro correlati della comunicazione fra due nodi di una rete.
I layer sono in totale 7 e vanno dal livello fisico (quello del mezzo fisico, ossia del cavo, delle onde radio o di qualsiasi altro sistema trasmissivo, tra cui la fibra ottica) fino al livello delle applicazioni, attraverso cui si realizza la comunicazione di alto livello.

Ogni layer individua un protocollo di comunicazione del livello medesimo. ISO/OSI realizza una comunicazione per livelli. Cioè, dati due nodi A e B, il livello n del nodo A può scambiare informazioni col livello n del nodo B, ma non con gli altri. Ogni livello in trasmissione realizza la comunicazione col livello corrispondente sui nodi di transito o sui nodi destinatari, usando il SAP (service access point) del livello immediatamente sottostante. Sicché ISO/OSI incapsula i messaggi di livello n in messaggi del livello n-1.
Così se A deve inviare, ad esempio, una e-mail a B, l'applicazione (liv. 7) di A propagherà il messaggio usando il layer sottostante (liv. 6) che a sua volta userà il SAP del layer inferiore, fino ad arrivare alla comunicazione ovvero alla trasmissione sul canale o mezzo fisico trasmissivo. In tal modo, si realizza una comunicazione multilivello che consente, ad esempio, di implementare algoritmi diversi per l'instradamento in rete, pur disponendo di protocolli di trasporto connessi, e di scegliere ed adattare protocolli e relativi algoritmi alla particolare rete di telecomunicazioni da realizzare. Tutto ciò conferisce modularità al sistema con maggiore semplicità di progettazione e gestione della rete, ovvero possibilità di migliorare, sviluppare e dunque eventualmente sostituire i protocolli dei vari strati, cioè implementare e reimplementare con dei protocolli ciascuna delle rispettive funzionalità logiche, lasciando inalterate le funzioni altrui.
Al fine di ottenere un miglioramento dell'attuale modello, può risultare interessante la possibilità di trarre vantaggio da uno scambio di informazioni nell'ambito della stessa pila protocollare, tra gli strati non adiacenti. Questa possibilità prende il nome di Cross-Layer (CL). È interessante soprattutto nelle comunicazioni radio, per la stessa natura variabile del mezzo trasmissivo, e sta riscuotendo un notevole interesse negli ultimi anni soprattutto per la riduzione dei consumi di energia nelle comunicazioni Wireless. L'idea fondamentale del concetto Cross-layer è di introdurre, nei vari protocolli di comunicazione, la capacità di scambiarsi informazioni per adattarsi agli specifici stati del collegamento della rete. A differenza del modello OSI classico, non si vogliono più prendere solo delle contromisure preventive o dei meccanismi di controllo successivi ad un determinato evento verificatosi nella rete, bensì si cerca di decidere in modo interattivo con essa, sfruttando le informazioni comuni a tutti gli strati.
A differenza del modello TCP/IP, il modello ISO/OSI è costituito da 7 livelli, che vengono di seguito spiegati, partendo dall'alto.

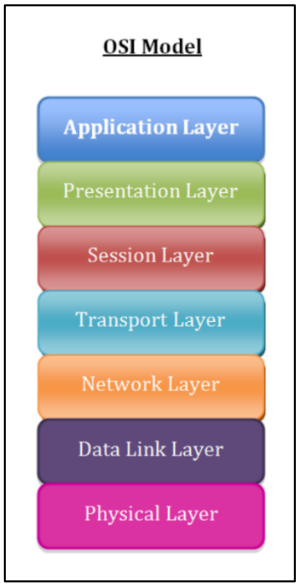


Figura 2.15: Layers del modello ISO/OSI.

- **Application Layer:** fornisce un insieme di protocolli che operano a stretto contatto con le applicazioni. Potrebbe essere errato identificare un'applicazione utente come parte del livello applicazione.
  I protocolli delle applicazioni tipiche di questo livello realizzano operazioni come ad esempio:

  - Trasferimento file;
  - Terminale virtuale;
  - Posta elettronica.

- **Presentation Layer:** permette di trasformare i dati forniti dalle applicazioni in un formato standardizzato, offrendo servizi di comunicazione comuni, come la crittografia, la compressione del testo e la riformattazione.
  Esso consente di gestire la sintassi dell'informazione da trasferire.
  Sono previste tre diverse sintassi:

  - astratta (definizione formale dei dati che gli applicativi si scambiano);
  - concreta locale (indicante come i dati sono rappresentati localmente);
  - di trasferimento (indicante come i dati sono codificati durante il trasferimento).

- **Session Layer:** controlla la comunicazione tra applicazioni. Essa Instaura, mantiene ed abbatte le connessioni (sessioni) tra applicazioni cooperanti. Si occupa anche della sincronia di invio/ricezione messaggi.
  Esso consente di aggiungere ai servizi forniti dal livello di trasporto servizi più avanzati, quali la gestione del dialogo (mono o bidirezionale), la gestione del token (per effettuare mutua esclusione) o la sincronizzazione (inserendo dei checkpoint in modo da ridurre la quantità di dati da ritrasmettere in caso di gravi malfunzionamenti).
  Si occupa anche di inserire dei punti di controllo nel flusso dati: in caso di errori nell'invio dei pacchetti, la comunicazione riprende dall'ultimo punto di controllo andato a buon fine.

- **Transport Layer:** permette un trasferimento trasparente e affidabile di dati (implementando anche un controllo degli errori e delle perdite) tra due host. È il primo livello realmente end-to-end, cioè da host sorgente a destinatario.
  Si occupa di:
  - stabilire, mantenere e terminare una connessione, garantendo il corretto e ottimale funzionamento della sottorete di comunicazione;
  - controllare la congestione, cioè evitare che troppi pacchetti dati arrivino allo stesso router contemporaneamente, con effetto di perdita di pacchetti stessi.

  Si occupa anche di effettuare la frammentazione dei dati provenienti dal livello superiore in pacchetti, detti "segmenti", e di trasmetterli in modo efficiente ed affidabile, usando il livello rete ed isolando da questo i livelli superiori. Infine, si preoccupa di ottimizzare l'uso delle risorse di rete e di prevenire la congestione.
  La sua unità dati fondamentale è il messaggio.

- **Network Layer:** Permette di rendere i livelli superiori indipendenti dai meccanismi e dalle tecnologie di trasmissione usate per la connessione e di prendersi carico della consegna a destinazione dei pacchetti.

È responsabile di:

- routing: scelta ottimale del percorso di rete da utilizzare per garantire la consegna delle informazioni dal mittente al destinatario. La scelta è svolta dal router attraverso dei particolari algoritmi di Routing ed attraverso tabelle di routing;
- conversione dei dati nel passaggio fra una rete ed un'altra con diverse caratteristiche, come il protocollo di rete utilizzato (internet-working).

Deve, quindi:

- tradurre gli indirizzi di rete;
- valutare la necessità di frammentare i pacchetti dati se la nuova rete ha una diversa Maximum Transmission Unit (MTU);
- valutare la necessità di gestire diversi protocolli attraverso l'impiego di gateway.

La sua unità dati fondamentale è il pacchetto.

- **Data Link Layer:** permette il trasferimento affidabile di dati attraverso il livello fisico. Invia frame di dati con la necessaria sincronizzazione ed effettua un controllo degli errori e delle perdite di segnale. Tutto ciò consente di far apparire, al livello superiore, il mezzo fisico come una linea di trasmissione esente da errori di trasmissione.
  Questo livello si occupa in primis di formare i dati da inviare attraverso il livello fisico, incapsulando il pacchetto proveniente dallo strato superiore in un nuovo pacchetto provvisto di un nuovo *header* (intestazione) e *tail* (coda), usati anche per sequenze di controllo. Questa frammentazione dei dati in specifici pacchetti è detta *framing* e i singoli pacchetti sono i *frame*.
  Come controllo di errore, per ogni pacchetto ricevuto, il destinatario invia al mittente un pacchetto ACK (*acknowledgement*, conferma), contenente lo stato della trasmissione. Il mittente deve ripetere l'invio dei pacchetti mal trasmessi e di quelli che non hanno ricevuto riscontro/risposta. Per ottimizzare l'invio degli ACK, si usa una tecnica detta Piggybacking, che consiste nell'accodare ai messaggi in uscita gli ACK relativi ad una connessione in entrata, per ottimizzare l'uso del livello fisico.
  I pacchetti ACK possono anche essere raggruppati e mandati in blocchi.
  Questo livello si occupa anche di controllare il flusso di dati (controllo di flusso). In caso di sbilanciamento della velocità di trasmissione tra mittente e destinatario, si occupa di rallentare l'operazione della macchina più veloce, accordandola all'altra e minimizzando così le perdite dovute a sovraccarico sul destinatario.
  La sua unità dati fondamentale è il frame.

- **Physical Layer:** trasmette un flusso di dati non strutturati attraverso un collegamento fisico, occupandosi della forma e dei livelli di tensione del segnale. Ha a che fare con le procedure meccaniche ed elettroniche necessarie a stabilire, mantenere e disattivare un collegamento fisico.
  Semplicemente, si occupa di controllare la rete, gli hardware che la compongono e i dispositivi che permettono la connessione.

  In questo livello si decidono:

  - il numero di bit in un collegamento (PROB);
  - le tensioni scelte per rappresentare i valori logici dei bit trasmessi;
  - la durata in microsecondi del segnale che identifica un bit;
  - la modulazione e la codifica utilizzata;

- l'eventuale trasmissione simultanea in due direzioni (Half-Duplex o Full-Duplex);
- la forma e la meccanica dei connettori usati per collegare l'hardware al mezzo trasmissivo.

Modello UDP:

Lo **U**ser **D**atagram **P**rotocol (UDP) è uno dei principali protocolli di rete della suite di protocolli Internet. È un protocollo di livello di trasporto a pacchetto, usato di solito in combinazione con il protocollo di livello di rete IP.
A differenza del TCP, l'UDP è un protocollo di tipo connectionless. Inoltre, non gestisce il riordinamento dei pacchetti né la ritrasmissione di quelli persi, per cui è generalmente considerato di minore affidabilità. In compenso è molto rapido (non c'è latenza per riordino e ritrasmissione) ed è efficiente per le applicazioni "leggere" o time-sensitive. Ad esempio, è usato spesso per la trasmissione di informazioni audio-videoreal-time come nel caso delle trasmissioni Voip (una tecnologia che rende possibile effettuare una conversazione tramite rete telefonica, sfruttando una connessione Internet o una qualsiasi altra rete di telecomunicazioni dedicata a commutazione di pacchetto). Infatti, visto che le applicazioni in tempo reale richiedono spesso un bit-rate minimo di trasmissione, esse non vogliono ritardare eccessivamente la trasmissione dei pacchetti e possono tollerare qualche perdita di dati, per cui il modello di servizio TCP può non essere particolarmente adatto alle loro caratteristiche.
L'UDP fornisce soltanto i servizi basilari del livello di trasporto, ovvero:

- multiplazione delle connessioni, ottenuta attraverso il meccanismo di assegnazione delle porte;
- verifica degli errori (integrità dei dati) mediante una checksum, inserita in un campo dell'intestazione (header) del pacchetto, Invece, TCP garantisce anche il trasferimento affidabile dei dati, il controllo di flusso e il controllo della congestione.

L'UDP è un protocollo stateless, ovvero non tiene nota dello stato della connessione. Rispetto al TCP, ha meno informazioni da memorizzare. Un server dedicato ad una particolare applicazione che scelga UDP come protocollo di trasporto può supportare, quindi, molti più client attivi.
L'utilizzo di UDP anziché TCP si giustifica col fatto che una comunicazione real-time non può tollerare ritardi e latenze aggiuntive o troppo elevate dovute alla ritrasmissione dei pacchetti persi e agli ACK.

3) Protocollo BACnet

3.1) Introduzione a BACnet

BACnet è un protocollo di comunicazione dati costituito da un insieme di regole che governano lo scambio di dati su una rete di computer coprendo tutto, dal tipo di cavo da utilizzare a come formare una particolare richiesta o dare un comando in modo standard. Rende speciale BACnet il fatto che le regole si riferiscono specificamente alle esigenze delle apparecchiature per l'automazione e il controllo degli edifici (BAC), ovvero riguardano aspetti come chiedere il valore di una temperatura, definire un programma operativo del ventilatore o inviare lo stato di una pompa allarme.
BACnet è stato sviluppato da un comitato formato dall'**A**merican **S**ociety of **H**eating, **R**efrigerating and **A**ir-Conditioning **E**ngineers (ASHRAE). L'obiettivo principale del comitato era quello di creare un protocollo che consentisse l'interazione tra sistemi costruttivi di diversi produttori, ovvero di lavorare insieme in modo armonioso. Prima dell'avvento di BACnet non c'era semplicemente alcun modo pratico per raggiungere questo obiettivo.
Il protocollo BACnet oggi è:

- uno standard nazionale in America;
- uno standard Europeo;
- uno standard adottato in oltre 30 paesi;
- uno standard ISO 2 globale.

Per ottenere l'interoperabilità su un ampio spettro di apparecchiature, la specifica BACnet è composta da tre parti principali. La prima parte descrive un metodo per rappresentare qualsiasi tipo di apparecchiatura per l'automazione degli edifici in modo standard. La seconda parte definisce i messaggi che possono essere inviati attraverso una rete di computer per monitorare e controllare tali apparecchiature. Infine, la terza parte definisce un insieme di LAN accettabili che possono essere utilizzate per trasmettere comunicazioni BACnet.
Diamo un'occhiata a ciascuno di questi componenti delle specifiche BACnet in modo un po' più dettagliato.

oggetti BACnet

Per realizzare questo trucco apparentemente impossibile, BACnet fornisce un modello standard nel rappresentare le funzioni di qualsiasi dispositivo, come ingressi e uscite analogici e binari, programmi, circuiti di controllo ed allarmi. A tal scopo, seleziona le raccolte di informazioni correlate, chiamate "oggetti", ciascuno dei quali ha una serie di "proprietà" che lo caratterizzano ulteriormente. Ogni ingresso analogico, ad esempio, è rappresentato da un "oggetto di ingresso analogico" BACnet, che ha una serie di proprietà standard, come valore attuale, tipo di sensore, posizione, limiti di allarme e così via. Alcune di queste proprietà sono richieste, mentre altre sono opzionali.
Una delle proprietà più importanti dell'oggetto è il suo identificatore, un nome numerico che consente a BACnet di accedervi in modo inequivocabile. Una volta che i dispositivi hanno "apparenze" comuni sulla rete in termini di oggetti e proprietà, è, quindi, possibile definire messaggi in grado di manipolare queste informazioni in modo standard.
Uno degli oggetti standard BACnet è l'oggetto di ingresso analogico, che rappresenta un ingresso di sensore analogico come un termistore. La figura 3.1 mostra un diagramma di un oggetto di ingresso analogico che può essere visto sulla rete attraverso cinque delle sue

proprietà. Alcune delle proprietà, come Descrizione, Tipo_unità e Unità, vengono impostate durante l'installazione.
Altre, tra cui Present_Value e Out_Of_Service, forniscono lo stato sull'ingresso del sensore, rappresentato dall'oggetto Analog Input. Infine, altre proprietà (un oggetto di ingresso analogico può avere fino a 25 proprietà) possono essere impostati dal produttore dell'apparecchiatura.
Tutto può essere letto; in questo esempio, una query sulla proprietà Present_Value di questo oggetto di input analogico otterrà la risposta "68.0".

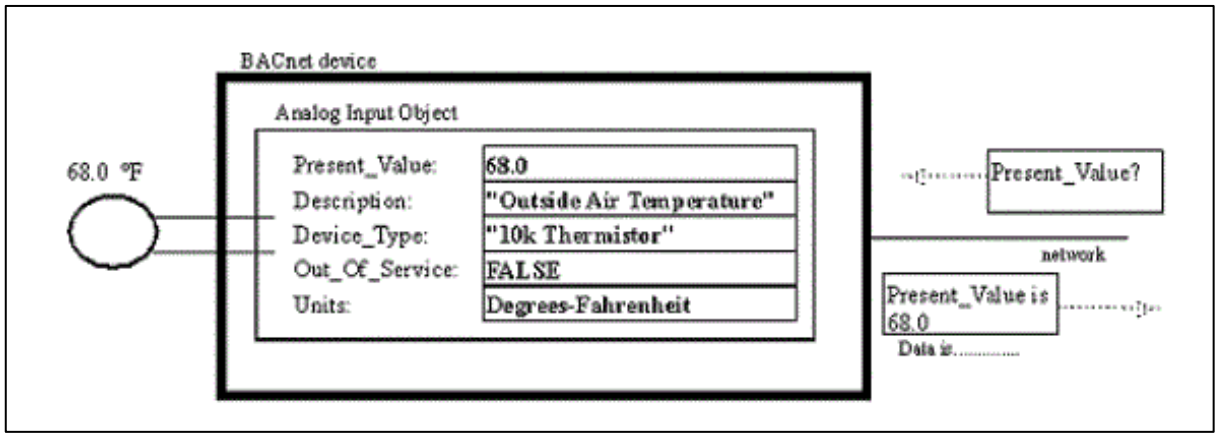


Figura 3.1: Un oggetto di tipo ingresso analogico.

BACnet definisce 18 tipi standard di oggetti, elencati nella tabella in figura 3.2. L'elenco deve essere completo. Ogni elemento di un sistema di controllo completo dell'edificio è rappresentato da uno o più oggetti, sia che si tratti di un ingresso analogico per un sensore o di una classe di notifica per la distribuzione degli allarmi.
La scelta di quali oggetti siano presenti in un dispositivo BACnet è determinata dalla funzione e dalle capacità del dispositivo. Lo standard BACnet non richiede tutti gli oggetti in tutti i dispositivi BACnet. Un dispositivo che controlla una scatola VAV avrà probabilmente diversi oggetti di ingresso analogico e di uscita analogica, mentre una workstation Windows non ha né ingressi sensore né uscite di controllo.
Ogni dispositivo BACnet deve avere un oggetto dispositivo, le cui proprietà descrivono completamente il dispositivo BACnet alla rete. La proprietà Object_List dell'oggetto dispositivo, ad esempio, fornisce un elenco di tutti gli oggetti contenuti nel dispositivo BACnet.
Le proprietà Vendor_Name, Vendor_Identifier e Model_Name forniscono il nome del produttore e il modello del dispositivo.
Inoltre, BACnet consente ai produttori di fornire oggetti proprietari, che non saranno necessariamente accessibili o compresi in apparecchiature di altri produttori. Tuttavia, non interferiranno con gli oggetti BACnet standard.

OBJECT	EXAMPLE OF USE
Analog Input	Sensor input
Analog Output	Control output
Analog Value	Setpoint or other analog control system parameter
Binary Input	Switch input
Binary Output	Relay output
Binary Value	Binary (digital) control system parameter
Calendar	Defines a list of dates, such as holidays or special events, for scheduling.
Command	Writes multiple values to multiple objects in multiple devices to accomplish a specific purpose, such as day-mode to night-mode, or emergency mode.
Device	Properties tell what objects and services the device supports, and other device-specific information such as vendor, firmware revision, etc.
Event Enrollment	Describes an event that might be an error condition (e.g., "Input out of range") or an alarm that other devices to know about. It can directly tell one device or use a Notification Class object to tell multiple devices.
File	Allows read and write access to data files supported by the device.
Group	Provides access to multiple properties of multiple objects in a read single operation.
Loop	Provides standardized access to a "control loop."
Multi-state Input	Represents the status of a multiple-state process, such as a refrigerator's On, Off, and Defrost cycles.
Multi-state Output	Represents the desired state of a multiple-state process (such as It's Time to Cool, It's Cold Enough and it's Time to Defrost).
Notification Class	Contains a list of devices to be informed if an Event Enrollment object determines that a warning or alarm message needs to be sent.
Program	Allows a program running in the device to be started, stopped, loaded and unloaded, and reports the present status of the program.
Schedule	Defines a weekly schedule of operations (performed by writing to specified list of objects with exceptions such as holidays. Can use a Calendar object for the exceptions.

Figura 3.2: Oggetti BACnet standard.

Proprietà

Lo standard BACnet identifica 123 diverse proprietà degli oggetti. Un sottoinsieme diverso di queste proprietà è specificato per ogni tipo di Oggetto. La specifica BACnet richiede che siano presenti determinate proprietà per ciascun Oggetto. Le altre proprietà specificate sono opzionali. In entrambi i casi, le proprietà implementate hanno comportamenti specifici definiti dalla specifica, in particolare quelli coinvolti nelle notifiche di allarmi o eventi e quelli che hanno un effetto sui valori o sugli stati di controllo.
Alcune proprietà standard sono richieste dalla specifica BACnet per essere scrivibili, mentre altre possono essere scrivibili a discrezione del produttore. Tutto può essere letto sulla rete.
BACnet consente ai fornitori di aggiungere proprietà proprie ma, come per gli oggetti propri, queste potrebbero non essere comprese o accessibili da apparecchiature di altri produttori.
L'oggetto di input analogico è rappresentativo degli oggetti coinvolti direttamente con gli elementi di controllo e molte delle sue proprietà lo riflettono. La tabella in figura 3.3 elenca le Proprietà definite dell'oggetto Input analogico, insieme ai valori tipici o di esempio per ciascuna proprietà. Ad esempio, Status_Flags, Event_State, Affidabilità, Out_Of_Service, Min_Pres_Value, Max_Pres_Value, Notification_Class, High_Limit, Low_Limit, Limit_Enable, Event_Enable, Acked_Transitions e Notify_Type Properties si occupano del rilevamento di anomalie, generando il sensore allarmi in risposta.

PROPRIETÀ	BACnet	ESEMPIO
Object_Identifier	necessario	**Ingresso analogico n. 1**
OBJECT_NAME	necessario	" **AI 01** "
Tipo di oggetto	necessario	**Ingresso analogico**
Valore attuale	necessario	**68.0**
Descrizione	Opzionale	"**Temperatura dell'aria esterna** "
Tipo di dispositivo	Opzionale	" **Termistore 10k** "
Status_Flags	necessario	**In_Alarm** , **Guasto** , **sovrascritto** , **Out_Of_Service** bandiere
Event_State	necessario	**Normale** (oltre a vari stati di segnalazione dei problemi)
Affidabilità	Opzionale	**No_Fault_Detected** (oltre a varie condizioni di errore)
Fuori servizio	necessario	**falso**
Intervallo di aggiornamento	Opzionale	**1,00** (secondi)
unità	necessario	**Gradi Fahrenheit**
Min_Pres_Value	Opzionale	**-100,0** , valore di lettura minimo affidabile
Max_Pres_Value	Opzionale	**+300,0** , massimo valore di lettura affidabile
Risoluzione	Opzionale	**0.1**
COV_Increment	Opzionale	Notifica se Present_Value cambia in base all'incremento: **0,5**
Ritardo	Opzionale	Secondi da attendere prima di rilevare fuori portata: **5**
Notification_Class	Opzionale	Invia notifica COV all'oggetto classe di notifica: **2**
High_Limit	Opzionale	**+215,0** , intervallo normale superiore
LOW_LIMIT	Opzionale	**-45,0** , intervallo normale inferiore
Banda morta	Opzionale	**0.1**
Limit_Enable	Opzionale	Abilita **report con limiti elevati** , **report con limiti bassi** .
Event_Enable	Opzionale	Abilita **segnalazione modifiche To_Offnormal** , **To_Fault** , **To_Normal** .
Acked_Transitions	Opzionale	Bandiere che indicano i riconoscimenti ricevuti per le modifiche precedenti.
Notify_Type	Opzionale	**Eventi** o allarmi

Figura 3.3: Proprietà dell'oggetto ingresso analogico.

Le prime tre proprietà elencate, Object_Identifier, Object_Name e Object_Type, devono essere presenti in ogni oggetto di un dispositivo BACnet.
Object_Identifier è un codice a 32 bit che identifica il tipo di oggetto (identificato anche dalla proprietà Object_Type) e il suo numero " *Instance* ", che insieme identificano in modo univoco l'oggetto all'interno del suo dispositivo BACnet. Teoricamente, un dispositivo BACnet potrebbe avere oltre quattro milioni di oggetti di un tipo particolare. Object_Name è una stringa di testo che ha una capacità unica. I dispositivi BACnet possono trasmettere query per dispositivi che contengono oggetti con un nome oggetto specifico. Ciò può semplificare notevolmente la configurazione del progetto.
BACnet richiede che un *oggetto dispositivo* sia presente in ogni dispositivo BACnet. L'oggetto dispositivo rende le informazioni sul dispositivo e le sue capacità disponibili ad altri dispositivi sulle reti. Prima che un dispositivo BACnet inizi a controllare le comunicazioni correlate con un altro, deve ottenere alcune delle informazioni presentate dall'oggetto dispositivo dell'altro dispositivo.

La figura 3.4 fornisce un esempio di un oggetto dispositivo per un controller VAV a doppio condotto BACnet. Sebbene l'elenco delle proprietà sia imponente, la maggior parte sono riparate dal produttore e sono lette solo da altri dispositivi BACnet.
A differenza di altri oggetti, il numero di istanza dell'oggetto dispositivo deve essere univoco nell'intera rete BACnet, perché viene utilizzato per identificare in modo univoco i dispositivi BACnet. Inoltre, esso può essere utilizzato per identificare comodamente il dispositivo BACnet da altri dispositivi durante l'installazione.

PROPRIETÀ	BACnet	ESEMPIO
Object_Identifier	necessario	**Dispositivo n. 1076**
OBJECT_NAME	necessario	" **Controllo DD Office 36** "
Tipo di oggetto	necessario	**Dispositivo**
Stato del sistema	necessario	**Operativo** (più altri)
Nome del fornitore	necessario	" **Alerton Technologies, Inc.** "
Vendor_Identifier	necessario	**Alerton**
Nome del modello	necessario	" **Controller VAV-DD** "
Firmware_Revision	necessario	" **1.0** "
Application_Software_Version	necessario	" **DDC a doppio condotto** "
Posizione	Opzionale	" **Office 36, Floor 3** "
Descrizione	Opzionale	" **(sulla rete 5)** "
protocol_version	necessario	**1** (versione del protocollo BACnet)
Protocol_Conformance_Class	necessario	**2**
Protocol_Services_Supported	necessario	**readProperty, writeProperty, atomicWriteFile, ...**
Protocol_Object_Types_Supported	necessario	**Ingresso analogico, Uscita analogica, ...**
Object_List	necessario	**Ingresso analogico n. 1, Ingresso analogico n. 2, ...**
Max_APDU_Length_Supported	necessario	**50** (byte o caratteri)
Segmentation_Supported	necessario	**No**
VT_Classes_Supported	Opzionale	**n / A**
Active_VT_Sessions	Opzionale	**n / A**
Ora locale	Opzionale	**12: 30: 15.22**
Local_Date	Opzionale	**Martedì 12 marzo 1996**
UTC_Offset	Opzionale	**+480** (minuti da GMT / UTM)
Daylight_Savings_Status	Opzionale	**Falso** (non in vigore)
APDU_Segment_Timeout	Opzionale	**n / A**
APDU_Timeout	necessario	**3000** millisecondi
Number_Of_APDU_Retries	necessario	**0**
List_Of_Session_Keys	Opzionale	**n / A**
Time_Synchronization_Recipients	Opzionale	**n / A**
Max_Master	Opzionale	**n / A**
Max_Info_Frames	Opzionale	**n / A**
Device_Address_Binding	necessario	**Nessuna**

Figura 3.4: Esempio dell'oggetto dispositivo Controller VAV-DD.

Servizi BACnet

BACnet definisce 32 servizi e li classifica in cinque categorie. Queste categorie di servizi sono Allarme ed Evento, Accesso ai file, Accesso agli oggetti, Gestione dispositivi remoti e Servizi terminali virtuali. Per ciascuno dei servizi "Confermati" (è prevista una risposta, in genere con dati), etichettato " **C** " nelle tabelle seguenti, il dispositivo BACnet può avere la possibilità di avviare la richiesta di Servizio o avere la capacità di elaborare e rispondere a una richiesta ricevuta di quel tipo, oppure comprendere entrambi. Per ciascuno dei servizi "Non confermato" (nessuna risposta prevista), etichettato " **U** " nelle tabelle seguenti, il dispositivo BACnet può avere la capacità di avviare la richiesta di servizio o avere la capacità di elaborare una richiesta ricevuta di quel tipo, oppure comprendere tutti e due.

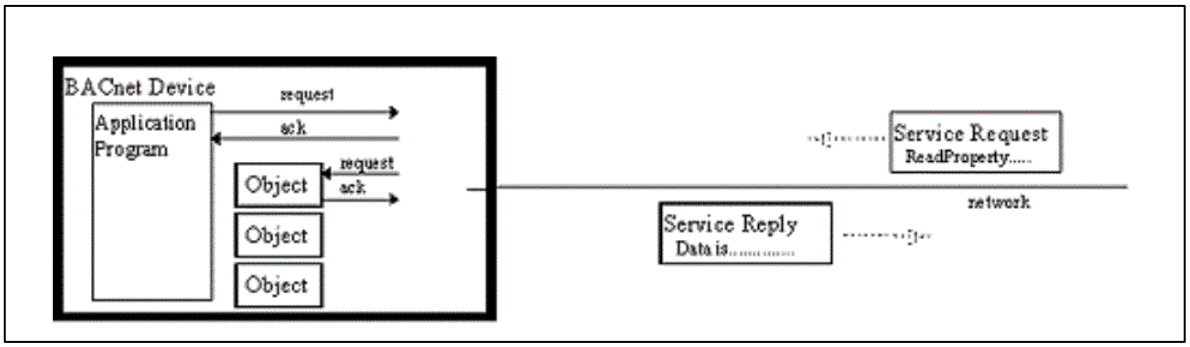


Figura 3.5: Richieste e risposte di servizi.

I dispositivi BACnet non sono tenuti ad implementare ogni singolo servizio. Solo un servizio, ReadProperty, deve essere elaborato da tutti i dispositivi BACnet. A seconda della funzione e della complessità del dispositivo, è possibile avviare o eseguire servizi aggiuntivi.
I servizi di allarme e di eventi gestiscono i cambiamenti nelle condizioni rilevate da un dispositivo BACnet, inclusi gli allarmi. Se si presentano modifiche di allarmi e di eventi che potrebbero indicare problemi o condizioni di errore, esso prevede il ripristino del normale funzionamento o la modifica di una lettura (riguardante qualche incremento rispetto al rapporto precedente), denominata "Change of Value" o COV.
Il reporting COV è un'alternativa utile al polling ripetuto di un oggetto per un valore monitorato. Un dispositivo contenente un oggetto monitorato potrebbe essere soggetto a congestione del traffico di rete, se molti altri dispositivi lo monitorassero. Il reporting COV consente di inviare avvisi quando si verifica una modifica.
Le notifiche di allarme, evento e COV sono generate dagli oggetti di monitoraggio del programma applicativo direttamente correlati alle operazioni di controllo: in particolare, i vari oggetti Input, Output e Value, nonché l'oggetto Loop.

SERVIZIO	BACnet	DESCRIZIONE
RiconosciSegnalazione	C	Utilizzato per dire al mittente dell'allarme che un essere umano ha visto l'allarme.
ConfirmedCOVNotification	C	Indica i dispositivi di sottoscrizione del COV verificatisi in una proprietà.
ConfirmedEventNotification	C	Utilizzato per comunicare al mittente una possibile condizione di errore.
GetAlarmSummary	C	Richiede da un dispositivo un elenco di "allarmi attivi", se presenti.
GenEnrollmentSummary	C	Richiede un elenco di "eventi" (possibile errore) che generano oggetti.
SubscribeCOV	C	Inviato da un dispositivo per richiedere la comunicazione dei COV in un oggetto.
UnConfirmedCOVNotification	U	Indica ai dispositivi in abbonamento che è avvenuta una modifica a una o più proprietà di un particolare oggetto.

Figura 3.6: Servizi di allarme ed eventi.

I servizi di accesso ai file in BACnet vengono utilizzati per leggere e manipolare i file nei dispositivi BACnet. In BACnet, i file rappresentano data-byte di gruppo di lunghezza e significato arbitrari; non si riferiscono necessariamente a nessun tipo di dispositivo di archiviazione di massa. A ogni file accessibile da BACnet è associato un oggetto file.
La parola "Atomic" nel nome del servizio significa semplicemente che può aver luogo una sola operazione alla volta di lettura o scrittura. Altri tentativi di accedere al file durante un accesso falliranno o verranno bloccati.
Al momento non ci sono file standard definiti da BACnet. Per definizione, tutti gli accessi ai file sono di proprietà del fornitore, sebbene BACnet abbia riservato i tipi di file EVENTLOG e VALUELOG per le strutture di file da definire in un secondo momento.

SERVIZIO	BACnet	DESCRIZIONE
AtomicReadFile	C	Richiede parte o tutto il file di un oggetto File.
AtomicWriteFile	C	Scrive in parte o in tutto il file di un oggetto File.

Figura 3.7: Servizi di accesso ai file.

I servizi di accesso agli oggetti forniscono i mezzi per leggere, modificare e scrivere proprietà e per aggiungere ed eliminare oggetti. Sono forniti più servizi per la lettura e la scrittura di proprietà. Lo scopo dei servizi più complessi (ReadPropertyMultiple e WritePropertyMultiple) è quello di combinare quante più letture o scritture su Proprietà degli oggetti all'interno di un dispositivo BACnet in un singolo messaggio, riducendo così l'overhead di rete. Il servizio ReadPropertyConditional fa un ulteriore passo avanti. Il dispositivo che elabora la richiesta verifica ogni Proprietà referenziata secondo i criteri inclusi nella richiesta e restituisce il valore solo se i criteri sono soddisfatti.
Sebbene i servizi CreateObject e DeleteObject siano definiti, si applicheranno a un set limitato di oggetti. Gli oggetti collegati a un dispositivo fisico, ad esempio, non sono suscettibili di essere né creabili né eliminabili. Gli oggetti Registrazione gruppo ed eventi, e possibilmente l'oggetto File, sono probabilmente i soliti soggetti dei servizi CreateObject e DeleteObject.

SERVIZIO	BACnet	DESCRIZIONE
AddListElement	C	Aggiunge uno o più elementi a una proprietà che è un elenco.
RemoveListElement	C	Rimuove uno o più elementi da una proprietà che è un elenco.
CreateObject	C	Utilizzato per creare una nuova istanza di un oggetto nel dispositivo di servizio.
DeleteObject	C	Utilizzato per eliminare un oggetto particolare nel dispositivo di servizio.
ReadProperty	C	Restituisce un valore di una proprietà di un oggetto.
ReadPropertyConditional	C	Restituisce i valori di più proprietà in più oggetti.
ReadPropertyMultiple	C	Restituisce i valori di più proprietà di più oggetti.
WriteProperty	C	Scrive un valore su una proprietà di un oggetto.
WritePropertyMultiple	C	Scrive valori su più proprietà di più oggetti.

Figura 3.8: Servizi di accesso agli oggetti.

I servizi di gestione dei dispositivi remoti, elencati nella Tabella in figura 3.8, forniscono una serie di funzioni disparate, tra cui il controllo da parte dell'operatore, il trasferimento di messaggi specializzati e le funzioni di indirizzamento / autoconfigurazione.
I servizi DeviceCommunicationControl e ReinitializeDevice hanno lo scopo di fornire a un operatore umano strumenti diagnostici che possono essere richiamati da remoto.

Con essi è possibile ordinare a un dispositivo BACnet di ignorare tutti i messaggi BACnet (ad eccezione dei servizi DeviceCommunicationControl e ReinitializeDevice) o avviare a caldo oppure a freddo.
I servizi ConfirmedPrivateTransfer e UnconfirmedPrivateTransfer vengono utilizzati per trasmettere messaggi al di fuori dello standard BACnet, invocando efficacemente servizi non standard. Questi Servizi riportano sia il Codice ID fornitore che il Codice servizio in forma BACnet standard. Il resto del contenuto è interamente a carico del venditore e non è probabile che sia interpretabile dai dispositivi di altri produttori.
I servizi ConfirmedTextMessage e UnconfirmedTextMessage trasportano messaggi di testo su altri dispositivi, come le stampanti.
Il servizio TimeSynchronization viene trasmesso da dispositivi designati, che dispongono di orologi, ad altri dispositivi, oppure vengono trasmessi in modo che i dispositivi possano essere sincronizzati.
I servizi Who-Is e I-Am vengono utilizzati per ottenere gli indirizzi di rete dei dispositivi BACnet su una rete BACnet. Possono semplificare la vita dell'installatore riducendo o eliminando la necessità di programmare gli indirizzi di rete di altri dispositivi in ciascun dispositivo BACnet. Al contrario, un dispositivo BACnet che deve conoscere l'indirizzo di uno o più dispositivi può trasmettere una richiesta Who-Is Service (messaggio) sulla rete interna, specificando un Numero istanza oggetto dispositivo o un intervallo di numeri istanza.
Le risposte non tornano come risposta. Invece, i dispositivi che hanno gli oggetti dispositivo specificati trasmettono una richiesta di servizio I-Am sulla rete locale, su una rete remota o sull'intera rete interna, in modo che il dispositivo richiedente vedrà la risposta, che porta con sé l'indirizzo informazioni del rispondente.
I servizi Who-Has e I-Have sono simili a Who-Is ed a I-Am, ma Who-Has aggiunge un identificatore oggetto (tipo di oggetto più il numero di istanza, univoco all'interno di ciascun dispositivo) o un nome oggetto (una proprietà di ogni oggetto, unico all'interno di ciascun dispositivo). I dispositivi di ricezione che contengono un oggetto corrispondente alla richiesta trasmettono la richiesta I-Have Service.

SERVIZIO	BACnet	DESCRIZIONE
DeviceCommunicationControl	C	Indica a un dispositivo di interrompere (e iniziare!) L'accettazione dei messaggi di rete.
ConfirmedPrivateTransfer	C	Invia un messaggio di proprietà del fornitore a un dispositivo.
UnconfirmedPrivateTransfer	U	Invia un messaggio di proprietà del fornitore a uno o più dispositivi.
ReinitializeDevice	C	Ordina al dispositivo ricevente di eseguire l'avvio a freddo o a caldo.
ConfirmedTextMessage	C	Trasmette un messaggio di testo a un altro dispositivo.
UnconfirmedTextMessage	U	Invia un messaggio di testo a uno o più dispositivi.
Timesynchronization	U	Invia l'ora corrente a uno o più dispositivi.
Che ha	U	Chiede quali dispositivi BACnet contengono un oggetto particolare.
Io ho	U	Risposta affermativa a Who-Has, trasmessa.
Chi è	U	Chiede la presenza di dispositivi BACnet specificati.
Sono	U	Risposta affermativa a Who-Is, trasmessa.

Figura 3.9: Servizi di gestione dei dispositivi remoti.

I servizi di terminale virtuale (VT) possono essere utilizzati da un operatore per stabilire una connessione di testo bidirezionale con un programma applicativo in esecuzione in un dispositivo remoto.
In effetti, per la durata di una sessione VT stabilita con il dispositivo remoto, il dispositivo dell'operatore appare come un terminale collegato al programma di applicazione remoto.

SERVIZIO	BACnet	DESCRIZIONE
VT-Open	C	Stabilisce una sessione di terminale virtuale con un dispositivo BACnet remoto.
VT-Chiudi	C	Chiude una sessione di terminale virtuale stabilita.
VT-Data	C	Invia il testo da un dispositivo all'altro partecipando a una sessione.

Figura 3.10: Servizi terminal virtuali.

Classi di conformità, gruppi di funzioni e PICS

La valutazione delle capacità di un dispositivo BACnet è potenzialmente un compito formidabile, data la grande scelta di oggetti, proprietà e servizi che possono essere implementati. Non è necessario che ogni dispositivo BACnet disponga di un'implementazione BACnet completa per svolgere il suo compito.
Il comitato BACnet di ASHRAE ha riconosciuto questo problema e ha risposto con aiuti alla valutazione sotto forma di "Classi di conformità", "Gruppi di funzioni" e "Dichiarazione di conformità di implementazione del protocollo" (PICS).
Il protocollo BACnet definisce sei livelli di classi di conformità, ognuna delle quali specifica il sottoinsieme minimo di servizi implementati sul dispositivo.
Il livello più basso (Classe di conformità 1) richiede solo che il dispositivo BACnet contenga un Oggetto dispositivo e che sia in grado di eseguire (rispondere a) una richiesta di servizio ReadProperty. Ogni livello successivo della Classe di conformità aggiunge le Richieste di servizio che devono essere eseguibili dal dispositivo, nonché le Richieste di servizio che deve essere in grado di avviare. La Classe di conformità 6 richiede l'implementazione di 21 tipi di richieste di servizio (su 32 in totale), di cui 20 devono essere avviabili e 17 eseguibili.
La Classe di conformità fornisce, quindi, una misura della capacità del dispositivo di comunicare.I gruppi di funzioni specificano una combinazione di oggetti e servizi necessari per eseguire determinate funzioni di automazione dell'edificio.
Essi sono definiti indipendentemente dalla Classe di conformità, sebbene l'implementazione di alcuni dei Gruppi di funzioni conferisca automaticamente una Classe di Conformità superiore a 1.
I Gruppi di funzioni specificati da BACnet sono elencati nella Figura seguente, con brevi descrizioni.

GRUPPO DI FUNZIONI	DESCRIZIONE
Orologio	Funzionalità generalmente associate all'avere un orologio.
Workstation portatile	Funzionalità di workstation portatili o portatili.
Workstation per personal computer	Funzionalità della workstation operatore principale.
Iniziazione all'evento	Rileva e genera allarmi ed eventi.
Risposta all'evento	Capacità di rispondere ad allarmi ed eventi.
Avvio evento COV	Capacità di avviare notifiche di cambio di valore.
Risposta evento COV	Iscriviti e ricevi notifiche di cambio di valore.
File	Capacità di leggere, scrivere, caricare e scaricare file.
Reinitialize	Possibilità di essere reinizializzato dal dispositivo remoto.
Operatore virtuale	In grado di fornire lato operatore della sessione di terminale virtuale.
Terminale virtuale	In grado di fornire lato server della sessione di terminale virtuale.
Comunicazioni del dispositivo	Esecuzione della richiesta di servizio DeviceCommunicationControl supportata.
Time Master	In grado di avviare la richiesta del servizio TimeSynchronization.

Figura 3.11: Gruppi di funzioni.

Al fine di facilitare al meglio le valutazioni dei dispositivi BACnet, la specifica BACnet fornisce anche una Dichiarazione di conformità all'implementazione del protocollo standard (PICS), creata dal produttore, che chiarisce in dettaglio le opzioni implementate in quel particolare dispositivo. La stessa specifica identifica il fornitore, descrive brevemente il dispositivo e precisa i dettagli dell'implementazione. Sono inclusi nella PICS:

- la classe di conformità del dispositivo,
- i gruppi di funzioni supportati,
- i servizi standard e proprietari eseguiti e avviati,
- gli oggetti standard e proprietari con proprietà opzionali, scrivibili, creabili o eliminabili

Inoltre, sono inclusi altri aspetti del dispositivo relativi alla sua implementazione, secondo lo standard BACnet. Lo standard fornisce una PICS pro-forma per semplificare lo sforzo delle specifiche.
Per la realizzazione di BACnet sono occorsi circa nove anni, al cui sviluppo hanno collaborato produttori, enti universitari, agenzie governative e società di consulenza. Ciò nel tentativo di produrre un protocollo veramente aperto, in base al quale le apparecchiature di diversi produttori possano interagire in un sistema di controllo dell'automazione dell'edificio completo ed integrato. Il risultato è stato uno standard che definisce tutti gli elementi di comunicazione tra dispositivi, dal linguaggio astratto di Oggetti e Servizi fino alle LAN fisiche. Con la sua adozione come standard ANSI e per l'interesse dimostrato sia da GSA che da altri, è corretto affermare che BACnet indica la strada verso il futuro dei controlli di automazione degli edifici.

LAN BACnet

LAN BACnet	Standard	Velocità dati	Dimensione pacchetto [1]	Costo
Ethernet	ISO / IEC 8802-3	Da 10 a 100 Mbps	1515 byte	alto
ARCNET	ATA / ANSI 878.1	Da 0,156 a 10 Mbps	501	medio
MS / TP	ANSI / ASHRAE 135-1995	Da 9,6 a 78,4 kbps	501	Basso
LonTalk [2]	n / A	Da 4,8 a 1250 kbps	228	vario

Figura 3.12: LAN BACnet

Ethernet è una LAN ad alta velocità che è stata ampiamente utilizzata per molti anni. In virtù della sua popolarità, le spese della sua interfaccia sono in calo, sebbene rimangano comunque elevate rispetto a molte altre LAN. Ethernet offre una serie di alternative multimediali, come cavi a doppini intrecciati, coassiali e in fibra ottica. Le interfacce standard per le postazioni di personal computer sono prontamente disponibili ed economiche, sebbene la necessità di hub e ripetitori possa aumentare il costo del sistema.
ARCNET è popolare nel settore del controllo di processo. È una LAN a basso costo rispetto a Ethernet, ma richiede circuiti integrati di comunicazione dedicati (IC), che mantengono il suo costo superiore rispetto ad alcune LAN BACnet.
La LAN MS / TP (Master-Slave / Token-Passing) è stata progettata per consentire ai produttori BAS di costruire dispositivi BACnet a basso costo, necessario per il successo di BACnet nella competizione con LAN proprietarie. Grazie alla sua semplice interfaccia e alle sue velocità di comunicazione, MS / TP può essere implementato su molti microcontrollori standard, senza il costo aggiuntivo di circuiti integrati di comunicazione dedicati. La LAN MS / TP utilizza la segnalazione EIA-485, tramite cablaggio a doppino intrecciato.
I dispositivi MS / TP sono disponibili in due varietà: Slaves e Masters. I dispositivi Slave sono particolarmente adatti per le implementazioni più economiche, ma mancano della capacità di avviare richieste. Essi possono solo rispondere ai messaggi di altri dispositivi. I dispositivi Master sono in grado di avviare richieste, ma devono anche essere in grado di negoziare un intervallo di tempo in cui effettuare le richieste. Ciò impone di aggiungere agli stessi alcuni requisiti di elaborazione e di memoria, che può comportare costi più elevati rispetto allo Slave.
Il dispositivo LonTalk è stato originariamente sviluppato come LAN proprietaria. Esso utilizzava uno speciale dispositivo di comunicazione che incorporava tre microprocessori, per gestire le spese generali. Recentemente è stato rilasciato come protocollo aperto. Lo standard BACnet sfrutta questa capacità di LonTalk per il trasporto di frame "estranei". LonTalk offre il maggior numero di opzioni nei mezzi di segnalazione, inclusi RF, infrarossi, doppini intrecciati, cavo coassiale e fibra ottica.

Per Internet

Spesso è necessario disporre di più reti in un'unica installazione BAS. È possibile che vi siano troppi dispositivi da collegare a una singola LAN oppure i requisiti dell'installazione potrebbero imporre l'utilizzo per funzioni diverse di vari tipi di LAN.

Quando due o più reti sono impostate per comunicare tra loro, il risultato è chiamato "internetwork". L'internetwork più noto è Internet.Esso, composto da molte reti più piccole in tutto il mondo. Internetworks può essere costituito da reti simili collegate tra loro, oppure, come in BACnet, possono contenere reti diverse con caratteristiche differenti.
Il collegamento in rete di un sistema di automazione degli edifici consente all'ingegnere di controllo di poter contenere i costi del sistema, con un paio di stratagemmi.
In primo luogo, i piccoli dispositivi come i controller unitari, che costituiscono la maggior parte di un BAS, danno un contributo importante al costo totale del sistema, in base al loro numero assoluto. Poiché i loro requisiti di comunicazione sono piccoli, in genere si connettono a una LAN progettata per avere un'interfaccia economica. Poiché tali LAN hanno un throughput inferiore, esiste un limite superiore pari al numero di dispositivi che possono essere collegati a una qualsiasi di tali LAN. Un edificio o campus più grande può avere diverse LAN di questo tipo.
In secondo luogo, un BAS può disporre di dispositivi, come workstation operatore o file server, che devono trasferire volumi di dati più grandi rispetto a quelli che possono essere gestiti dalla LAN a basso costo. Esistono pochi dispositivi di questo tipo in un BAS specifico e il costo di un'interfaccia LAN ad alte prestazioni è ridotto rispetto al prezzo del dispositivo. Utilizzando entrambi i tipi di LAN e selezionando ciascuna di esse in base alle caratteristiche che meglio soddisfano i requisiti dell'edificio, è possibile ottenere il sistema più economico senza penalizzare le prestazioni.
Questo concetto di miscelazione di varietà di rete non è nuovo. Molti sistemi di automazione degli edifici proprietari esistenti sono in realtà internetworks, che comprendono diversi tipi di LAN.
Il BAS, progettato per il grandissimo edificio federale Phillip Burton a San Francisco, dimostra l'uso efficace di queste strategie. LAN MS / TP a basso costo vengono utilizzate per comunicare con i controller unitari; ogni LAN MS / TP controlla uno o due piani. Una rete Ethernet ad alte prestazioni viene utilizzata come" backbone ", fornendo le comunicazioni tra le stazioni di lavoro dell'operatore, i file server e i server di stampa, e le reti MS / TP, come mostrato nella Figura 3.13.

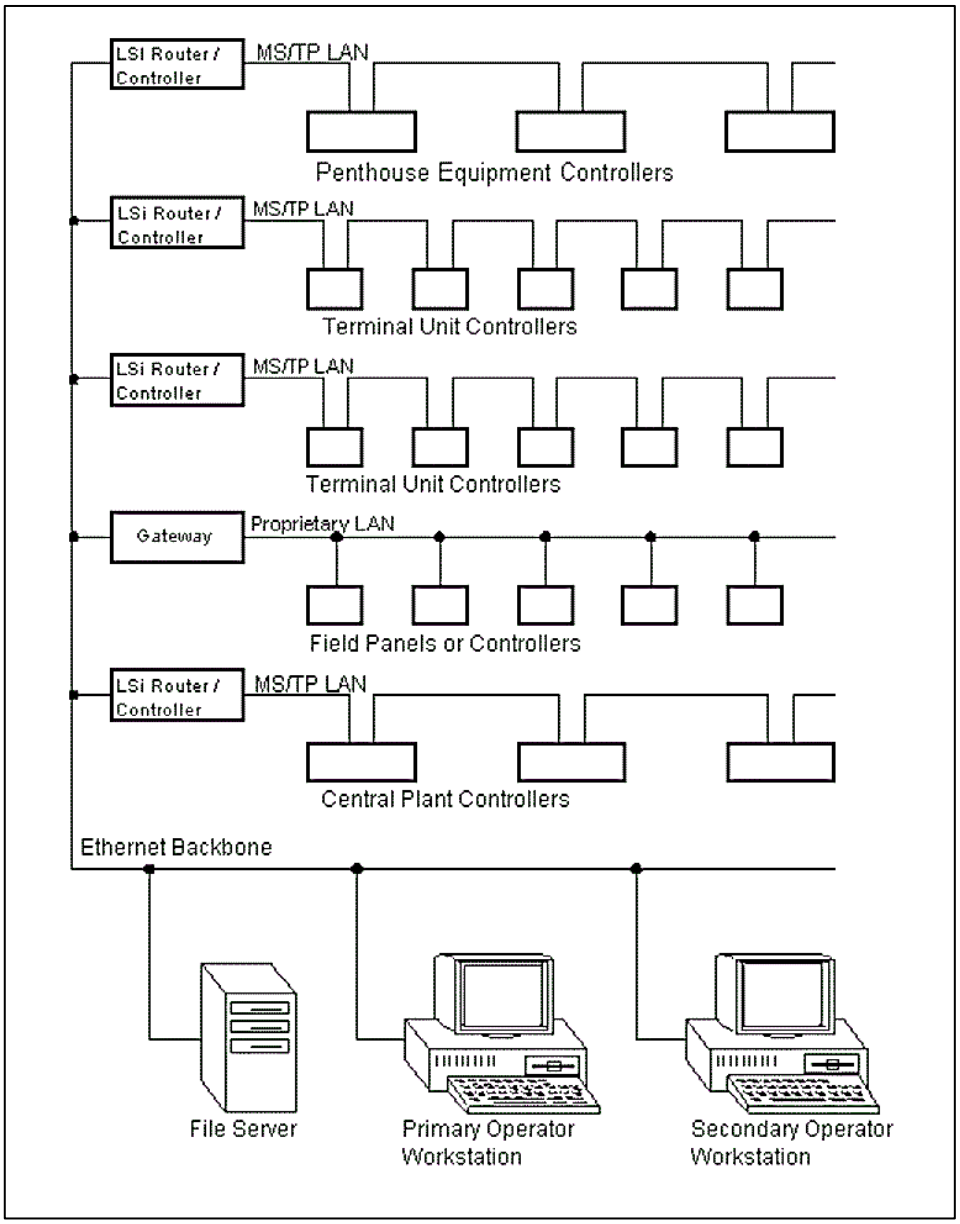


Figura 3.13: Esempio di una grande rete interna BAS.

Tali implementazioni non devono escludere sistemi proprietari. Come ulteriore dimostrazione di interoperabilità, verrà installata una rete proprietaria nell'edificio Phillip Burton per controllare l'HVAC su tre piani. Si collegherà al backbone Ethernet attraverso un "gateway", un dispositivo che fornisce un accesso BACnet alle operazioni della rete proprietaria.

Indirizzi Mac e di Rete

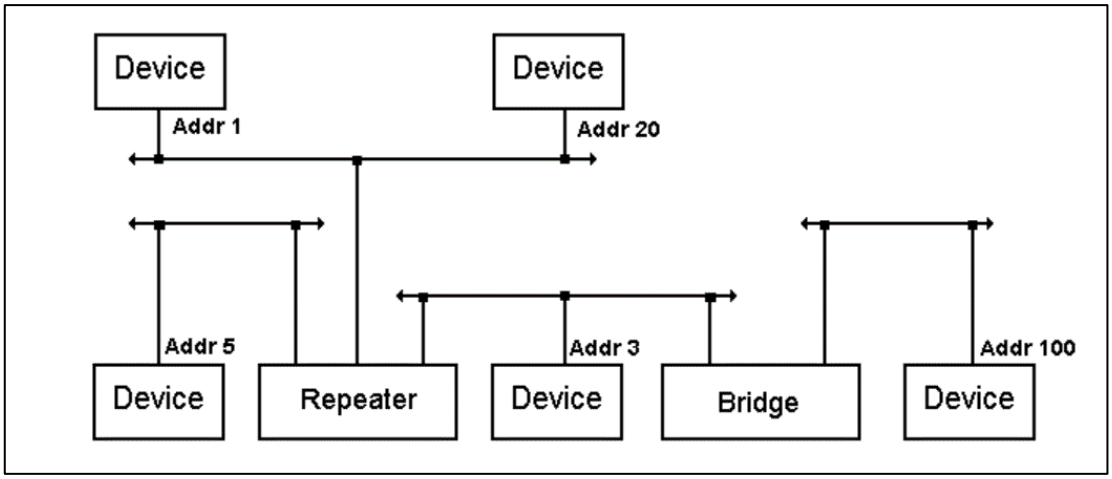


Figura 3.14: Trasferimento di messaggi in una singola rete.

L'esempio di figura 3.14 illustra un trasferimento di messaggi all'interno di una singola rete.
Il trasferimento di messaggi in una rete BACnet è un po' più complesso e si basa sulla connessione di ciascun dispositivo a una rete con un indirizzo univoco all'interno della rete intera. Questo indirizzo è composto da una coppia di numeri.
Il primo numero è indicato come indirizzo MAC (Medium Access Control), che deve essere univoco sulla rete a cui è collegato. Lo stesso indirizzo MAC può essere utilizzato su un dispositivo collegato su un'altra rete, allo stesso modo per cui i numeri civici possono essere gli stessi nelle case ubicate su strade diverse.
Gli indirizzi dei dispositivi mostrati in figura sono indirizzi MAC. In genere un indirizzo MAC è riservato per l'uso come indirizzo "broadcast". Ogni dispositivo sulla rete riceve messaggi inviati all'indirizzo di trasmissione e al proprio indirizzo MAC. L'indirizzo MAC è fisicamente programmato nel dispositivo in qualche modo, o dal produttore o dal fornitore del sistema.
La seconda parte dell'indirizzo di ogni dispositivo è il numero assegnato dal provider di sistema alla rete a cui viene stabilita la connessione. Questo numero deve essere univoco in una rete BACnet. È questo numero che distinguerà tra dispositivi con lo stesso numero MAC su reti diverse, al fine di garantire che i pacchetti di messaggi che attraversano la rete interna arrivino al dispositivo corretto.

Router:

I "router" sono i dispositivi che interconnettono le reti dei componenti in una rete interna. A differenza della maggior parte dei dispositivi che contengono una singola interfaccia di rete, i router si connettono a due o più LAN, allo scopo di trasmettere messaggi provenienti da un dispositivo su una rete a un dispositivo su un'altra rete.
I router trasmettono i messaggi solo alle LAN che devono essere incrociate per portare il messaggio alla sua destinazione finale. Questa capacità di inoltrare selettivamente il messaggio

è fondamentale; senza di essa l'intera rete interna sarebbe presto sovraccaricata, in quanto il traffico locale di ciascuna LAN verrebbe trasmesso inutilmente su ogni altra LAN.
I router si comportano come una struttura di smistamento della posta. Ogni pacchetto ricevuto da un router porta un indirizzo di destinazione. Il router esamina la parte del numero di rete dell'indirizzo e determina attraverso quale delle sue connessioni di rete (o "porte") il pacchetto deve essere inoltrato per raggiungere il dispositivo di destinazione.
Se l'indirizzo di destinazione è un dispositivo sulla LAN collegato a quella porta, il messaggio viene inviato direttamente a quel dispositivo. Se il dispositivo di destinazione non risiede su una LAN a cui è collegato il router, il pacchetto di messaggi deve essere inoltrato a un altro router, in grado di consegnare il pacchetto al dispositivo di destinazione.
Per mantenere basso il traffico di rete, è necessario che il router determini quale router adiacente può spostare il pacchetto verso la sua destinazione. Esso deve sapere su quale LAN si trova quel router (attraverso quale porta deve inviare il pacchetto) e quale indirizzo MAC di quel router è su quella LAN.
Ciò è possibile grazie all'uso di "tabelle di routing" presenti in ciascun router. Le tabelle di routing contengono un elenco di numeri di rete nella rete interna, con informazioni su come raggiungere tale rete, incluso se essa è direttamente connessa o meno a una delle porte di questo router. Se la porta non è nota, l'accesso attraverso il quale è possibile raggiungere il router successivo in quella direzione viene elencato insieme all'indirizzo MAC di quel router.
I router sono gli unici dispositivi BACnet che conoscono o hanno persino bisogno di sapere i numeri delle reti a cui sono collegati. Ciò semplifica il lavoro di configurazione di una rete interna BACnet.
La figura 3.15 illustra l'uso delle tabelle di routing nell'inoltro dei messaggi. Il router 1 riceve il messaggio destinato alla rete 4, indirizzo MAC 8. Cerca la rete 4 nella sua tabella e trova che deve inoltrare il messaggio alla rete 3, indirizzo MAC 5, che è il router 2.
Il router 2, alla ricezione del pacchetto, valuta la sua tabella e trova che la rete 4 è direttamente connessa ad esso. Quindi, invia il pacchetto all'indirizzo MAC 8 su quella rete. Si noti che il dispositivo sulla rete 3 con indirizzo MAC 8 non riceve il pacchetto.

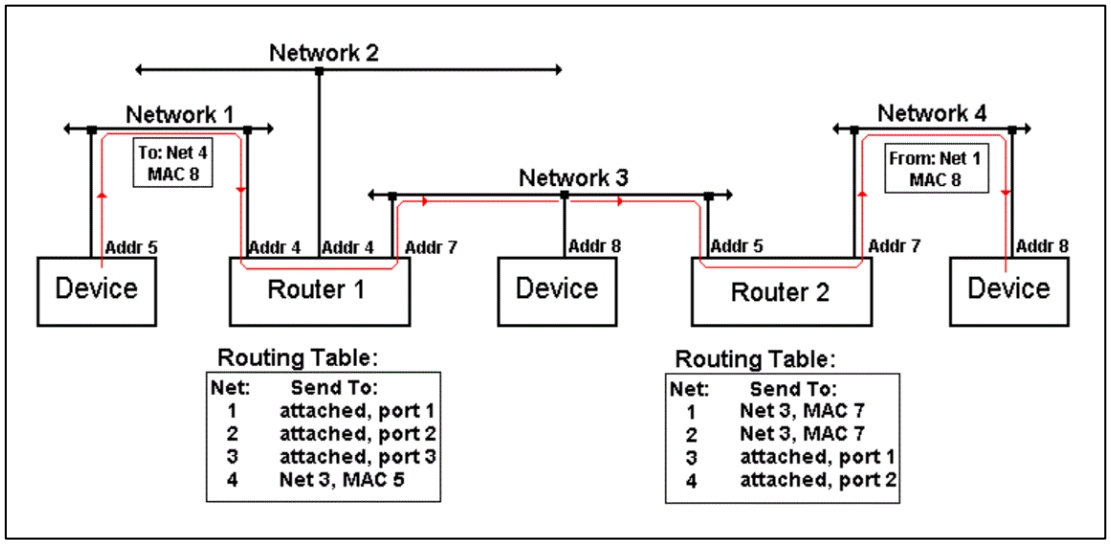


Figura 3.15: Inoltro dei messaggi tramite router.

I router sono disponibili in due varietà: "configurato" e "apprendimento". Un router configurato ha la sua intera tabella di routing impostata in anticipo, in genere con uno strumento di messa

in servizio. Questo è importante nell'impostare una rete di grandi dimensioni o nel predisporre un'attività ancora più grande per mantenerla, perché ogni cambiamento nella configurazione della rete richiede che molti router, se non tutti, debbano essere riprogrammati, prima che la rete possa funzionare correttamente.
BACnet specifica che tutti i router devono essere "in apprendimento".
Un router di apprendimento deve essere configurato solo con i numeri di rete e con le porte per le reti a cui è collegato. Lo standard BACnet specifica "messaggi di rete", mediante i quali i router possono scambiare informazioni e compilare automaticamente il resto delle loro tabelle di routing, anche regolando le tabelle quando le reti vengono aggiunte o rimosse dalla rete.
È importante notare che lo standard BACnet pone una limitazione sulla topologia di una rete BACnet: può esistere un solo percorso tra due dispositivi qualsiasi su una rete BACnet. Questo semplifica notevolmente il compito dei router, perché non devono eseguire test per determinare il routing ottimale dei messaggi verso altre reti, argomento di dibattito in corso per Internet stesso.
I router BACnet non si limitano a eseguire operazioni di routing. Si può avere un uso efficiente dell'hardware quando il dispositivo router esegue anche funzioni BAS di livello superiore. Una di queste applicazioni è un dispositivo di controllo globale che monitora i dispositivi BAS su LAN collegate per fornire la raccolta di dati e il coordinamento del sistema. Oltre alle proprie comunicazioni con questi dispositivi, può anche trasmettere messaggi dalla LAN backbone su cui comunica con i file server e le workstation.

Determinazione dell'indirizzo del dispositivo remoto

I dispositivi che stanno per inviare un pacchetto sulla propria LAN locale devono sapere a quale indirizzo MAC il pacchetto deve essere inviato. Ci sono un paio di mezzi con cui questo può essere determinato automaticamente.
Se il pacchetto di messaggi viene inviato come risposta a una query da un altro dispositivo, il pacchetto viene semplicemente restituito all'indirizzo MAC da cui proviene. Se la query proveniva da un altro dispositivo sulla stessa LAN, quell'indirizzo MAC è l'indirizzo di destinazione finale.
Se la query proviene da un dispositivo su un'altra rete, l'indirizzo MAC appartiene al router che ha inoltrato la richiesta a questa LAN, che può restituire la risposta al mittente. L'indirizzo MAC del router, insieme all'indirizzo di rete completo del mittente (che è presente se il messaggio proviene da un'altra rete), è un'informazione sufficiente per il pacchetto da consegnare al dispositivo remoto appropriato.
Per inviare una query a un dispositivo su un'altra rete, un dispositivo BACnet deve sapere dove si trova il dispositivo remoto, nonché l'indirizzo MAC sulla rete locale del router che è in grado di passare il messaggio nella direzione dell'altra rete. Queste informazioni potrebbero essere programmate manualmente nel dispositivo, ma normalmente i dispositivi in grado di avviare tali query supporteranno i servizi Who-Has e Who-Is.
Questi servizi forniscono i mezzi per localizzare un dispositivo remoto trasmettendo una richiesta per la sua identità. La risposta dal dispositivo contiene tutte le informazioni necessarie per inviargli un pacchetto. La richiesta deve solo identificare il numero del dispositivo remoto o il nome del dispositivo. I numeri di dispositivo o i nomi, ovviamente, devono essere inseriti come parte della sua configurazione nel dispositivo richiedente, ma la loro presenza elimina la necessità di inserire una considerevole quantità di informazioni di routing.

Il compito di creare una rete BACnet (campus) di grandi dimensioni può essere semplificato se all'inizio viene istituito un sistema di numerazione o denominazione uniforme per i dispositivi e le reti BACnet. Come esempio di tale codifica numerica, si potrebbero assegnare cifre del Numero dispositivo per codificare il tipo di dispositivo, l'edificio, il piano e il numero della stanza. In questo sistema, il numero di dispositivo 1203412 potrebbe rappresentare un controller unitario ("1") nell'edificio 203, piano 4, stanza 12. I numeri di dispositivo possono trovarsi ovunque nell'intervallo compreso tra 0 e 4.194.303, il che offre una certa flessibilità ai sistemi di numerazione. I numeri di rete devono essere assegnati in modo coerente ai precedenti, per evitare la duplicazione accidentale degli stessi e per facilitare la manutenzione del sistema.

Connessione al Modem e mezzi Router

Non tutte le comunicazioni tra dispositivi avvengono tramite LAN. Il tipico caso specifico è la connessione modem dial-up utilizzata per operazioni di supervisione o per il collegamento di siti remoti.
Lo standard BACnet supporta questa funzionalità con il protocollo Point-To-Point (PTP), che fornisce i mezzi con cui due dispositivi possono comunicare direttamente tra loro, senza l'overhead solitamente associato alle connessioni LAN. Questo protocollo consente ai due dispositivi di stabilire una connessione, scambiare pacchetti di messaggi (full duplex - in entrambe le direzioni contemporaneamente, per la massima efficienza) e, se lo si desidera, di terminare la connessione.
Lo standard BACnet esclude esplicitamente qualsiasi definizione del supporto fisico su cui viene utilizzato il protocollo PTP, ma supporta prontamente EIA-232 o connessioni modem dial-up. Con quest'ultimo in mente, supporta anche l'implementazione della protezione con password per la connessione. Il protocollo PTP è stato progettato anche per le velocità di comunicazione più lente (da 9.6 a 56 kbps), tipiche delle connessioni EIA-232 o modem.
Il protocollo PTP è esplicitamente progettato e utilizzato per il passaggio di pacchetti di messaggi da una rete all'altra, proprio come fa un router. Poiché ciascuno dei due dispositivi di comunicazione sembra un router per la rete o le reti a cui è collegato, ciascuno dei dispositivi è chiamato "mezzo router".
Nella figura 3.16 è mostrata una connessione PTP tra due half-router. Il router 1 è un router completo con funzionalità half-router (PTP); si collega a due reti e una porta PTP. Il router 2 è semplicemente un mezzo router; ha solo una connessione di rete.

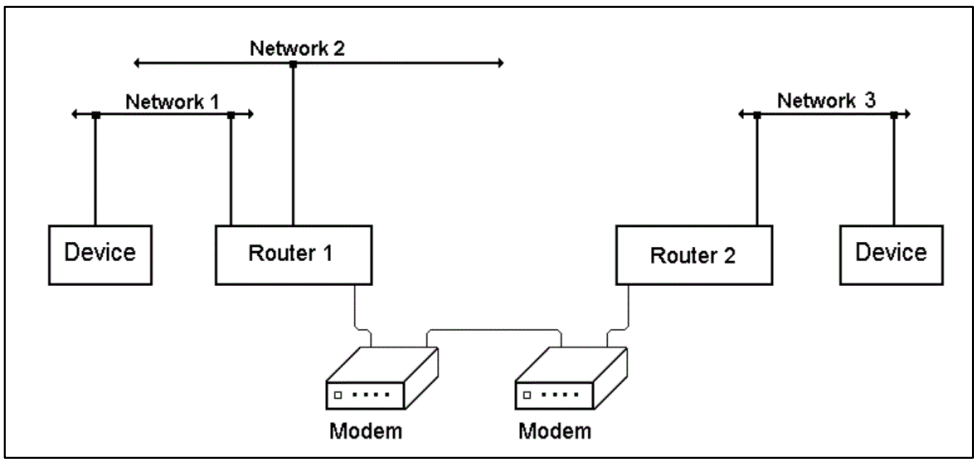


Figura 3.16: Connessione PTP tra due semi-router mediante modem.

Poiché è probabile che la connessione PTP venga stabilita tramite una connessione a pedaggio come la rete telefonica, di solito non sarà continuamente attiva. Lo standard BACnet definisce i messaggi con cui il mezzo router può segnalare lo stato della sua connessione PTP ai router sulle sue reti "locali" (quelli sullo stesso lato della connessione PTP del mezzo router). Definisce, inoltre, i messaggi mediante i quali è possibile indicare al mezzo router di stabilire o terminare la connessione.
Viene stabilita una connessione PTP quando un dispositivo BACnet deve comunicare con un dispositivo su una rete nell'altro lato della connessione. Il dispositivo di avvio invia una richiesta di informazioni su come raggiungere quella rete e il mezzo router risponde con un messaggio indicante che la rete è l'altra faccia di una connessione PTP. Il dispositivo di avvio indica quindi al mezzo router di effettuare la connessione.
Quando viene stabilita la connessione, il mezzo router trasmette queste informazioni. Poi, trasferisce i pacchetti di messaggi che riceve, destinati alle reti, sull'altro lato della connessione PTP. La connessione termina al verificarsi di una delle tre condizioni: la connessione viene esplicitamente terminata, la connessione scade o il protocollo rileva l'esistenza di più connessioni alla rete remota.

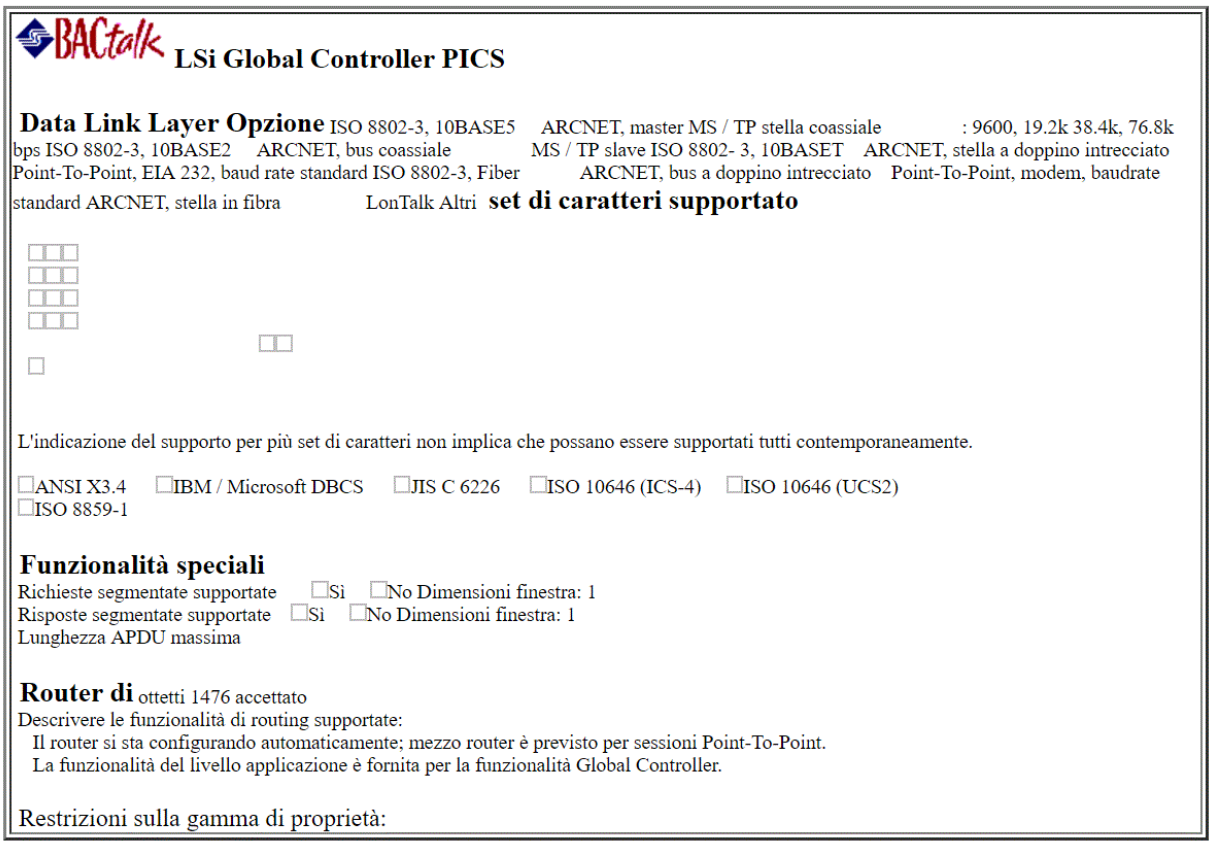
Grandi messaggi e segmentazione

Alcuni messaggi BACnet possono superare i limiti di dimensione dei pacchetti di una qualsiasi delle sue reti. Lo standard BACnet consente di inviare un messaggio di grandi dimensioni "segmentando", cioè suddividendo il messaggio in blocchi o "segmenti" per la trasmissione e, poi, ricostruendolo dai segmenti man mano che vengono ricevuti.
La segmentazione è una funzione opzionale per i dispositivi BACnet. Di solito è implementata in dispositivi più complessi, come pannelli di campo o stazioni di lavoro. Tuttavia, potrebbe non essere presente nei controller unitari più piccoli, che hanno meno probabilità di dover inviare o ricevere messaggi di grandi dimensioni.
Quando viene ricevuto un segmento di messaggio, il ricevitore verifica, prima di accettarlo, che il segmento sia intatto e, successivamente, invia un riconoscimento al trasmettitore. Il ricevitore può, in alternativa, richiedere la ritrasmissione della sequenza di segmenti, iniziando dal segmento errato. Ciò evita la necessità di ritrasmettere i segmenti buoni precedenti quando ne viene rilevato uno cattivo, riducendo, in tal modo, la quantità di traffico sulla rete interna, nel caso si verificano errori.
Per ottenere alcune prestazioni nel trasferimento di segmenti, BACnet consente l'invio e la ricezione di più segmenti prima della restituzione della risposta. Il numero massimo di segmenti che possono essere inviati prima di ricevere una risposta è chiamato "dimensione della finestra". Il grado di miglioramento delle prestazioni dipende in qualche modo dalla "qualità" delle reti. Se le reti fossero perfette e se i pacchetti non fossero mai persi o consegnati nella sequenza sbagliata, la dimensione della finestra potrebbe essere infinita. Nel mondo reale, in cui i pacchetti vengono talvolta persi durante il trasporto, la dimensione della finestra deve essere inferiore. Se i tempi di transito all'interno della rete sono ridotti, anche la dimensione della finestra può essere ridotta.
Prima che un messaggio BACnet sia trasmesso in segmenti, i dispositivi di invio e ricezione negoziano la meccanica della trasmissione. Essi determinano se entrambe le parti supportano la segmentazione e quale sarà la dimensione della finestra (ovvero, il numero più piccolo supportato da entrambi i dispositivi). Devono anche concordare la dimensione del pacchetto; sarà gestita da entrambi i dispositivi il pacchetto con le dimensioni più grandi Se queste

negoziazioni falliscono, viene generato un messaggio di errore, perché sarà richiesto l'intervento dell'operatore e alcuni elementi della rete interna BAS dovranno essere riconfigurati.
È molto importante notare che lo standard BACnet presuppone una gerarchia "normale" di LAN. Qualsiasi connessione LAN o PTP che formi un collegamento tra altre LAN deve essere in grado di gestire i messaggi almeno della dimensione degli altri due, perché il processo di negoziazione non tiene conto della dimensione del messaggio nelle reti intervenienti.

Specifiche e PICS

Il compito di valutare le capacità di rete di un dispositivo è stato semplificato dalla "Dichiarazione di conformità del protocollo" (PICS) specificata nello standard BACnet. Il PICS, fornito dal produttore di un dispositivo BACnet, identifica il produttore, descrive il dispositivo e fornisce dettagli sull'implementazione, inclusa la capacità di rete del dispositivo. La figura 3.17 mostra un esempio della parte di PICS che descrive le capacità di un router BACnet.


BACtalk **LSi Global Controller PICS**

Data Link Layer Opzione ISO 8802-3, 10BASE5 ARCNET, master MS / TP stella coassiale : 9600, 19.2k 38.4k, 76.8k bps ISO 8802-3, 10BASE2 ARCNET, bus coassiale MS / TP slave ISO 8802- 3, 10BASET ARCNET, stella a doppino intrecciato Point-To-Point, EIA 232, baud rate standard ISO 8802-3, Fiber ARCNET, bus a doppino intrecciato Point-To-Point, modem, baudrate standard ARCNET, stella in fibra LonTalk Altri **set di caratteri supportato**

L'indicazione del supporto per più set di caratteri non implica che possano essere supportati tutti contemporaneamente.

☐ANSI X3.4 ☐IBM / Microsoft DBCS ☐JIS C 6226 ☐ISO 10646 (ICS-4) ☐ISO 10646 (UCS2)
☐ISO 8859-1

Funzionalità speciali
Richieste segmentate supportate ☐Sì ☐No Dimensioni finestra: 1
Risposte segmentate supportate ☐Sì ☐No Dimensioni finestra: 1
Lunghezza APDU massima

Router di ottetti 1476 accettato
Descrivere le funzionalità di routing supportate:
Il router si sta configurando automaticamente; mezzo router è previsto per sessioni Point-To-Point.
La funzionalità del livello applicazione è fornita per la funzionalità Global Controller.

Restrizioni sulla gamma di proprietà:

Figura 3.17: PICS di rete di un router.

3.2) Driver BACnet

Il driver BACnet è stato sviluppato utilizzando Nodejs (cioè un runtime di JavaScript open source multipiattaforma orientato agli eventi per l'esecuzione di codice JavaScript) e la libreria del protocollo BACnet "backstack" open source, costituita al suo interno dalle funzioni che possono essere utilizzate per creare il proprio driver BACnet.
Il driver BACnet ha il compito di riconoscere, salvare e modificare le varie proprietà dei device BACnet collegati in rete TCP/IP.
Per riconoscimento s'intende la risposta (I-Am) che il Server BACnet inoltra al Client BACnet (che precedentemente aveva inoltrato una scansione della rete con il Who-Is). Nel nostro caso, la Raspberry PI3 definisce la comunicazione tra i dispositivi tramite protocollo BACnet.
Il driver Bacnet è costituito da una cartella contente tutte le librerie utilizzate (node_modules) e due file d'esecuzione, che rappresentano il corpo del driver (dbBacnet.js e mainBacnet.js), ognuno dei quali verrà spiegato in dettaglio nei paragrafi successivi.

Node_modules:

Il driver BACnet è costituito da una cartella node_modules in cui risiedono tutte le librerie, sia di default sia quelle installate successivamente, in modo tale da poter richiamare particolari funzioni, che, attraverso queste librerie, possono essere utilizzate liberamente all'interno del proprio codice Nodejs.

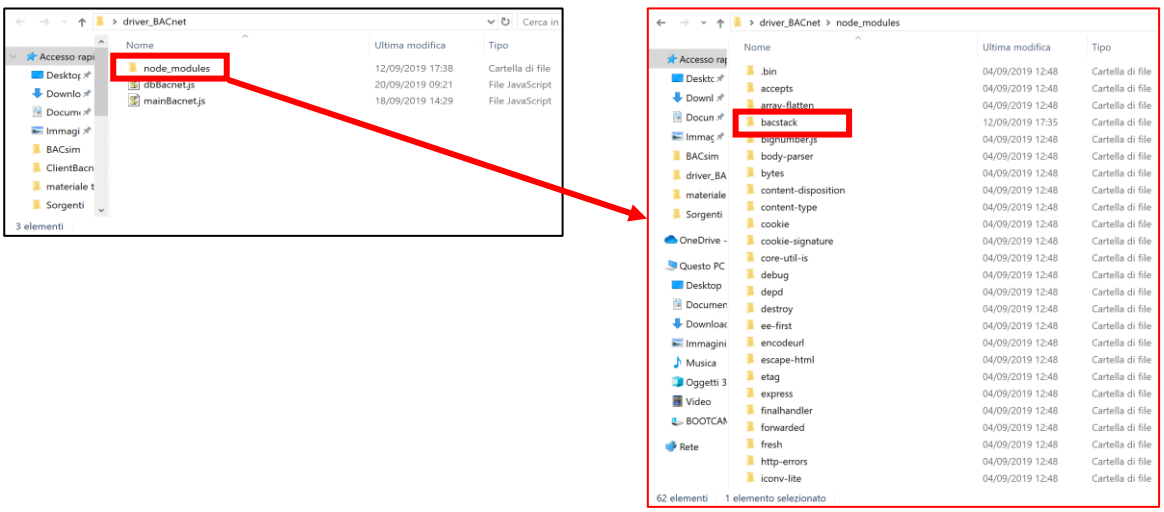


Figura 3.18: Caratterizzazione della cartella node_modules.

Un esempio risulta essere la libreria "backstack", scaricata gratuitamente da Github, in cui risiedono tutte le funzioni che permettono di sviluppare un driver BACnet corretto e funzionale. Come si spiegherà in seguito, per il codice sviluppato, ho avuto bisogno principalmente delle funzioni risiedenti all'interno del codice client.js, presente all'interno della libreria "backstack".

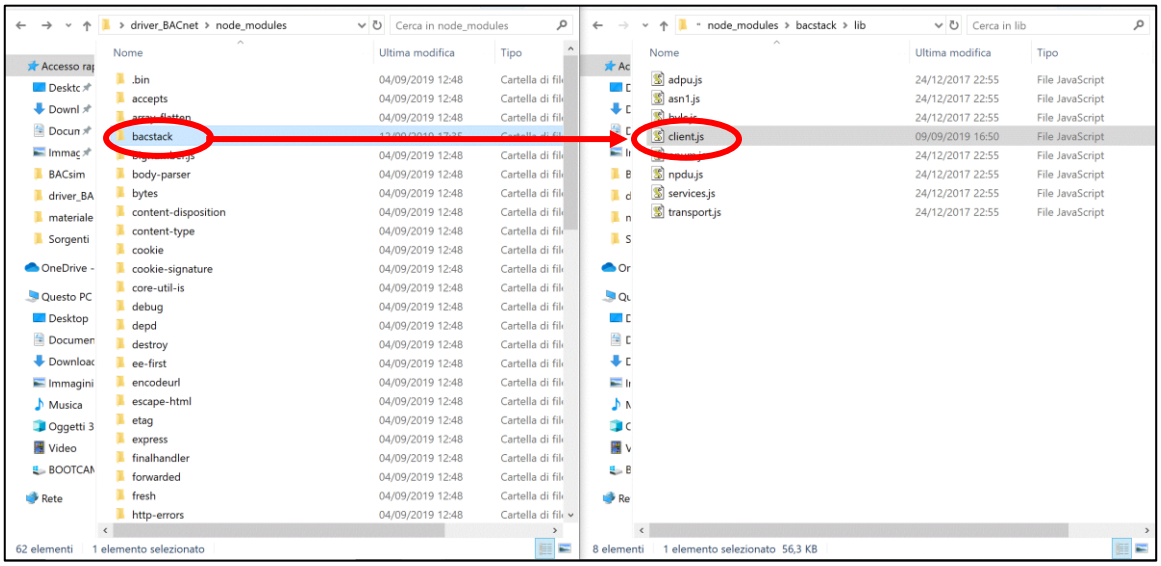


Figura 3.19: Caratterizzazione della libreria “backstack”.

In client.js sono presenti tutte le funzioni riguardanti la gestione dei device BACnet, la lettura e la scrittura di proprietà. Le interpretazioni delle risposte ricevute dai vari device BACnet possono essere esportate ed utilizzate da chiunque inserisse la libreria “backstack” all’interno del proprio codice Nodejs.

dbBacnet.js

Come accennato in precedenza, nella cartella driver_BACnet, abbiamo i due codici sviluppati in Nodejs. Il primo codice, chiamato dbBacnet.js, ha il compito di gestire al suo interno la connessione con il database e tutte le funzioni di manipolazione dei dati all’interno dello stesso. Nel suo corpo risiedono anche le funzioni di Polling, ossia l’aggiornamento nel database e la lettura da device BACnet di variabili.
Di seguito andremo a spiegare le parti fondamentali del codice dbBacnet.js, tralasciando le parti di meno interesse, ma comunque valutabili in Appendice A.

Nella figura 3.20 viene mostrata la parte iniziale del codice dbBacnet.js in cui vi è la definizione delle variabili che saranno utilizzate all’interno del codice dbBacnet.js e la definizione dei campi per la connessione con il database.
Particolare attenzione va fatta alla variabile “mainBacnet”, che permette di utilizzare all’interno del codice le funzioni esportate nel mainBacnet.js.

```
// Gestione della connessione , delle funzioni di aggiunta e richiesta del database:

// Variabili
var mainBacnet = require('./mainBacnet'); // connessione al mainBacnet.js ( permette l'utilizzo delle funzioni esportate da esso)
var mysql = require('mysql');             // connessione alla libreria mysql

var unixTime = require('unix-time');      // connessione alla
var time = unixTime(new Date());          // libreria unix-time

var deviceList = [];       // Array su cui vengono salvati i dati di rilevanza del device
var values =[];
var numDev = 0;

// Creazione della connessione espilicitando i campi fondamentali del nostro database MySQL:
var db = mysql.createConnection(
{
    host: '127.0.0.1',
    user: 'root',
    password: 'password',
    port: 3306
}
);
```

Figura 3.20: Definizione delle variabili in dbBacnet.js.

Nel codice in figura 3.21 si mostra, invece, la connessione vera e propria con il database precedentemente definito nella variabile “db” , come mostrato in figura 3.20.
In caso in cui vi sia una corretta connessione con il database, si provvederà ad inserire un messaggio del tipo “ +++ Database BACnet Connected +++ “ nel terminale di visualizzazione.

```
// Permette la connessione al database, se la connessione avviene con successo avvisa tramite un messaggio sulla console:
db.connect((err) => {
    if (err) {
        throw err;
    }
    console.log(' +++  Database BACnet Connected  +++ ');
});

```

Figura 3.21: Connessione con il database in dbBacnet.js.

La funzione “addDevice” ha il compito di aggiungere nel database i campi time, id_device, address, deviceId, maxApdu, segmentation e vendorId relativi al device BACnet, utilizzando la query presente in figura 3.22. Inoltre, andremo a salvare nell’array “deviceList” ognuno di questi device, per ottenere un salvataggio locale delle informazioni ed avere una disponibilità delle informazioni che possono essere richieste successivamente all’interno del codice, senza prelevarli ogni volta dal database.
La funzione “addPropertyIO” ha lo stesso compito della precedente, ma fa riferimento al salvataggio dei campi di interesse relativi ad ogni proprietà dell’oggetto di un particolare device BACnet, come ad esempio il Binary Input.
Anche in questo caso, verrà aggiornato l’array deviceList con le proprietà di quel particolare device BACnet.

```
// Funzione che permette l'aggiunta di un dispositivo Bacnet nel databse,la condizione di controllo che permette di non aggiungere 2 device uguali è
// definita dal valore dell'id_device (primari key):
function addDevice(device)
{
    var currDev;
    var deviceArray =[];
    deviceArray = [[time,
                    device.currentIdDevice,
                    device.device.address,
                    device.device.deviceId,
                    device.device.maxApdu,
                    device.device.segmentation,
                    device.device.vendorId]];

    currDev = {address: device.device.address, deviceId: device.device.deviceId, numProp: 0, object_prop:[]};
    deviceList.push(currDev);
    numDev++;

    db.query("INSERT INTO database_bacnet.devices_bacnet (time_stamp, id_device,address,deviceID,maxApdu,segmentation, vendorId) VALUES ?", [deviceArray]);
}

// Funzione che aggiunge tutte le proprietà dell'oggetto del device nel database:
function addPropertyIO(object)
{
    var idProp;
    var currProp;
    var record = [[time, object.object.id_bacnet_io, object.object.id_device, object.object.object_Identifier, object.object.object_Name, object.object.object_Type,
        object.object.present_Value, object.object.event_State, object.object.out_of_Service, object.object.update_Interval, object.object.units,
        object.object.min_Pres_Value, object.object.max_Pres_Value, object.object.resolution, object.object.cov_Increment, object.object.high_Limit,
        object.object.low_Limit]];

        db.query("INSERT INTO database_bacnet.bacnet_io (time_stamp, id_bacnet_io, id_device, object_Identifier, object_Name, object_Type, present_Value," +
        " event_State, out_of_Service, update_Interval, units, min_Pres_Value, max_Pres_Value, resolution, cov_Increment, hight_Limit, low_Limit)  VALUES ? ",
        [record]);

        idProp = deviceList[object.object.id_device -1].numProp;

        /* Aggiungi nuovo elemento nell'array */
        currProp = {type: object.object.object_Type,
                    presentValue: object.object.present_Value,
                    instance: object.object.object_Identifier,
                    id_io:object.object.id_bacnet_io
                    };

        deviceList[object.object.id_device - 1].object_prop.push(currProp);
        deviceList[object.object.id_device - 1].numProp++;
}

```

Figura 3.22: Funzioni di inserimento nel database in dbBacnet.js.

In dbBacnet.js, oltre alle funzioni di inserimento precedentemente spiegate, vi sono anche 2 funzioni che, ogni volta che vengono richiamate, hanno il compito di richiedere dal device (funzione selectOutputValue) e scrivere il Present_Value (cioè valore attuale) nel database (funzione updateInputValue)

```
// Funzione che aggiorna il Present Value nel database:
function updateInputValue(value, idxDev, idxProp, idxIdIo)
{
    deviceList[idxDev].object_prop[idxProp].presentValue = value;

    db.query("UPDATE database_bacnet.bacnet_io SET present_Value = "+value.toString()+" WHERE id_bacnet_io = "+idxIdIo.toString());
}

// Funzione che richiede il Present Value nel database:
function selectOutputValue(idxIdIo,type, instance, address, callback)
{
    db.query("SELECT present_Value FROM database_bacnet.bacnet_io WHERE id_bacnet_io = "+idxIdIo.toString(),
        function(err, value, fields)
        {
            if (callback != 0)
            {
                callback(value[0].present_Value,type, instance, address);

            }
        });
}
```

Figura 3.23: Funzioni di aggiornamento e lettura del Present Value del database in dbBacnet.js.

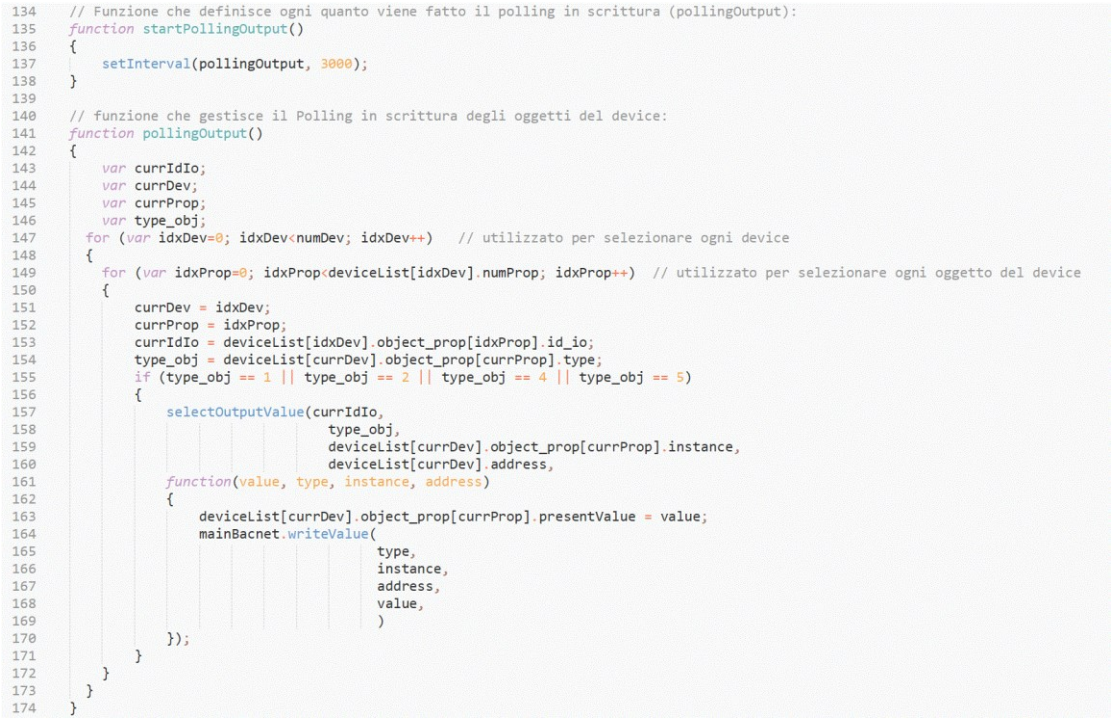
La funzione “pollingInput” presente in figura 3.24, gestisce l’aggiornamento di tutti i valori degli oggetti dei device BACnet presenti nel database. La funzione è definita da due cicli for: il primo per scorrere i device e il secondo, che risulta essere nel campo del primo ciclo for, per poter selezionare tutti gli oggetti del device BACnet i-esimo.
Per ogni oggetto del device viene richiamata la funzione “readValue”, presente in mainBacnet.js con l’obiettivo di andare a leggere il valore di quell’oggetto specifico. Una volta che questo valore è disponibile, tramite una callback, viene richiamata la funzione “updateInputValue”, che ha il compito di aggiornare il Present_Value nel database.

La funzione “startPollingInput” viene utilizzata per richiamare ogni 3 secondi la funzione di “pollingInput”.

```
102  // Funzione che definisce ogni quanto viene fatto il polling in lettura (pollingInput):
103  function startPollingInput()
104  {
105      setInterval(pollingInput, 3000);
106  }
107
108  // Funzione che gestisce il Polling in lettura degli oggetti del device:
109  function pollingInput ()
110  {
111      var currIdIo;
112      var currDev;
113      var currProp;
114      for (var idxDev=0; idxDev<numDev; idxDev++)  // utilizzato per selezionare ogni device
115      {
116          for (var idxProp=0; idxProp<deviceList[idxDev].numProp; idxProp++)  // utilizzato per selezionare ogni oggetto del device
117          {
118              currDev = idxDev;
119              currProp = idxProp;
120              currIdIo = deviceList[idxDev].object_prop[idxProp].id_io;
121
122              mainBacnet.readValue(
123                                      deviceList[idxDev].object_prop[idxProp].type,
124                                      deviceList[idxDev].object_prop[idxProp].instance,
125                                      deviceList[idxDev].address,
126                                      function(value)
127                                      {
128                                          updateInputValue(value, currDev, currProp, currIdIo);
129                                      });
130          }
131      }
132  }
```

Figura 3.24: Funzione di Polling in lettura in dbBacnet.js.

La funzione “pollingOutput”, presente in figura 3.25, gestisce l’aggiornamento dei valori del device BACnet. Infatti, il corpo della funzione è definito da due cicli for, come nella funzione precedentemente spiegata per la lettura, in modo da poter considerare tutti gli oggetti di ogni i-esimo device.
Per ogni oggetto BACnet, andremo a leggere il valore da aggiornare nel database tramite la funzione “selectOutputValue” e, una volta ottenuto tale valore, andremo a richiamare tramite una callback (funzione presente nella linea di codice numero 161 della Figura 3.25) la funzione del mainBacnet.js “writeValue”, che permette di andare a scrivere il nuovo Present_Value all’interno dell’oggetto del Device BACnet.
Inoltre, la funzione “startPollingOutput” viene utilizzata per richiamare ogni 3 secondi la funzione di “pollingOutput”.



```
// Funzione che definisce ogni quanto viene fatto il polling in scrittura (pollingOutput):
function startPollingOutput()
{
    setInterval(pollingOutput, 3000);
}

// funzione che gestisce il Polling in scrittura degli oggetti del device:
function pollingOutput()
{
    var currIdIo;
    var currDev;
    var currProp;
    var type_obj;
  for (var idxDev=0; idxDev<numDev; idxDev++)   // utilizzato per selezionare ogni device
  {
    for (var idxProp=0; idxProp<deviceList[idxDev].numProp; idxProp++)  // utilizzato per selezionare ogni oggetto del device
    {
        currDev = idxDev;
        currProp = idxProp;
        currIdIo = deviceList[idxDev].object_prop[idxProp].id_io;
        type_obj = deviceList[currDev].object_prop[currProp].type;
        if (type_obj == 1 || type_obj == 2 || type_obj == 4 || type_obj == 5)
        {
            selectOutputValue(currIdIo,
                              type_obj,
                              deviceList[currDev].object_prop[currProp].instance,
                              deviceList[currDev].address,
            function(value, type, instance, address)
            {
                deviceList[currDev].object_prop[currProp].presentValue = value;
                mainBacnet.writeValue(
                                      type,
                                      instance,
                                      address,
                                      value,
                                      )
            });
        }
    }
  }
}
```

Figura 3.25: Funzione di Polling in scrittura in dbBacnet.js.

La parte conclusiva del codice dbBacnet.js in figura 3.26 riguarda i comandi per esportare le funzioni precedentemente valutate, in modo tale da poter essere utilizzate da tutti gli altri moduli che importino il codice dbBacnet.js.
Nel nostro caso il modulo che utilizza le funzioni del dbBacnet.js risulta essere solo il mainBacnet.js.

```
// Funzione del dbBacnet.js esportate:

exports.startPollingOutput = startPollingOutput;   // Esporto la funzione di attivazione del Polling in scrittura
exports.startPollingInput = startPollingInput;     // Esporto la funzione di attivazione del Polling in lettura
exports.addDevice = addDevice;                     // Esporto la funzione di aggiunta del device BACnet
exports.addPropertyIO = addPropertyIO;             // Esporto la funzione di aggiunta dell'oggetto del device BACnet

```

Figura 3.26: Funzioni del dbBacnet.js esportate nel modulo mainBacnet.js.

mainBacnet.js

Il secondo codice presente nella cartella driver_BACnet risulta essere il mainBacnet.js, sviluppato sempre in Nodejs, in cui è implementata la logica effettiva del driver BACnet.
In questo modulo, andremo a definire una logica che permette, una volta riconosciuto, di richiedere e modificare i valori relativi agli oggetti che appartengono a quello specifico device BACnet, utilizzando le funzioni di default che sono presenti nel modulo client.js della libreria “backstack”.
Come nel dbBacnet.js, anche in questo caso la parte iniziale del codice riguarda la definizione delle variabili che serviranno all’interno del codice, oltre all’importazione di librerie e di moduli Nodejs.

```
// Valutazione delle risposte del Server Bacnet e indirizzamento alle varie funzioni di scrittura nel Database
// e lettura e scrittura dei valori del device:

//Costanti
const bacnet = require('bacstack');                 // connessione alla libreria BACnet
const client = new bacnet({apduTimeout: 6000 });

// Variabili
var dbBacnet = require('./dbBacnet.js');          //  // connessione al dbBacnet.js ( permette l'utilizzo delle funzioni esportate da esso)
var  value;
var whoIsTimer = setInterval(myTimer, 6000);       // Intervallo di tempo con cui viene richiamata la funzione di Who-Is
var read = [];                                     // Array su cui vengono salvati i dati di ritorno della funzione ReadPropertyMultiple
var id_device = 1;                                 // valore univoco assegnato ad ogni device
var id_bacnet_io = 1;                              // valore univoco assegnato ad ogni oggetto del device
```

Figura 3.27: Costanti e Variabili definite nel mainBacnet.js.

In figura 3.28, vi è l'attivazione delle funzioni di Polling presenti nel modulo dbBacnet.js spiegate precedentemente, che hanno la funzione di leggere dal device BACnet e di aggiornare nel database ogni set-point le proprietà degli oggetti appartenenti ai device presenti in rete.

```
dbBacnet.startPollingInput();    // avvio la funzione di aggiornamento del present_value di ogni object nel database;
dbBacnet.startPollingOutput();   // avvio la funzione di aggiornamento del present_value di ogni object nel device;
```

Figura 3.28: Attivazione delle funzioni di Polling nel mainBacnet.js.

All'avvio del mainBacnet.js si effettua la scansione della rete per identificare i dispositivi BACnet tramite la funzione Who-Is, presente nella libreria di default "backstack".
In figura 3.29 si può notare la gestione dopo aver ricevuto la risposta I-Am da parte del Server BACnet, dove, una volta visualizzato le proprietà del device nella console in cui viene lanciato lo script, andremo a stoppare la scansione del Who-Is.
Successivamente, provvederemo a salvare le proprietà del device BACnet sia nel database, tramite la funzione "addDevice" presente nel dbBacnet.js, sia nella struttura locale, denominata deviceAddressArray.
Infine, andremo a chiedere al device BACnet la proprietà che ci permetterà di conoscere la lista degli oggetti presenti in quel device. Una volta disponibile questa, attiveremo tramite una callback la funzione "readObjectList", che spiegheremo in dettaglio successivamente.

```
client.on('iAm', function(device)
{
  var currentIdDevice;
  var deviceAddressArray = [];
  currentIdDevice = id_device;
  var struct_device= {device, currentIdDevice};
  currentIdDevice = id_device;
  console.log(' +++  Device Identifier  +++ ');
  console.log(' +++  Trovato device numero:', id_device,
              'Address:', device.address,
              'DeviceId:', device.deviceId,
              'MaxApdu:', device.maxApdu,
              'Segmentation:', device.segmentation,
              'VendorID:', device.vendorId,'  +++ ');
  clearInterval(whoIsTimer);
  dbBacnet.addDevice(struct_device);
  deviceAddressArray.push({ address: device.address, deviceId: device.deviceId });
  client.readProperty(device.address, {type:29, instance:0}, 211, {}, function(err, value)
  {
    readObjList(err, value, device.address, currentIdDevice);     // leggo la lista degli oggetti presenti nel device
  });
  id_device++;
});
```

Figura 3.29: Gestione della risposta del device BACnet nel mainBacnet.js.

In figura 3.30 si può notare la funzione che permette di eseguire la scansione della rete TCP/IP, richiamando la funzione predefinita client.js, presente nella libreria "backstack".

La funzione “myTimer” permette di lanciare tale scansione ogni 6 secondi, come è possibile notare in linea 12 della figura 3.27.

```
49  // Funzione che permette di eseguire la scansione della rete ogni secondo
50  function myTimer()
51  {
52    console.log(' +++  Who-Is sent  +++ ');
53    client.whoIs();
54  }
```

Figura 3.30: Avvio della scansione della rete con Who-Is nel mainBacnet.js.

La funzione “fillDbIoProp”, richiamata dopo aver ottenuto i valori delle proprietà degli oggetti BACnet, ha il compito di differenziare l’associazione delle proprietà dell’oggetto del device BACnet in relazione al fatto che essi siano binari o analogici. Infatti, nel primo caso non avremo bisogno di salvare nel database proprietà come, ad esempio, l’unità di misura (units), il valore minimo che la grandezza può assumere (min_Pres_Value) o il valore massimo che la grandezza può assumere (max_pres_Value), dato che risultano inappropriate per quel determinato oggetto.

```
56  // Funzione che mi carica i campi del database inerente agli oggetti differenziandoli tra binary e analog
57  function fillDbIoProp (err,value, id_dev)
58  {
59    var object = {};
60    var struct_object = {};
61    object.id_bacnet_io = id_bacnet_io;
62    object.id_device = id_dev;
63    object.object_Identifier =  value.values[0].values[0].value[0].value.instance;
64    object.object_Name = value.values[0].values[1].value[0].value;
65    object.object_Type = value.values[0].values[2].value[0].value;
66
67    // vado a valutare il tipo, se è binario vado a settare i campi non interessati dal
68    //tipo con 0 e undefined ( per l'unità di misura);
69    if (object.object_Type == 3 || object.object_Type == 4 || object.object_Type ==5)
70    {
71      object.present_Value = value.values[0].values[3].value[0].value;
72      object.event_State = value.values[0].values[4].value[0].value;
73      object.out_of_Service = value.values[0].values[5].value[0].value;
74      object.update_Interval = 0;
75      object.units = 'not defined';
76      object.min_Pres_Value = 0;
77      object.max_Pres_Value = 0;
78      object.resolution = 0;
79      object.cov_Increment = 0;
80      object.high_Limit = 0;
81      object.low_Limit = 0;
82    }
83    else  // altrimenti vado a settare i seguenti campi:
84    {
85      object.present_Value = value.values[0].values[3].value[0].value;
86      object.event_State = value.values[0].values[4].value[0].value;
87      object.out_of_Service = value.values[0].values[5].value[0].value;
88      object.update_Interval = value.values[0].values[6].value[0].value;
89      object.units = value.values[0].values[7].value[0].value;
90      object.min_Pres_Value = value.values[0].values[8].value[0].value;
91      object.max_Pres_Value = value.values[0].values[9].value[0].value;
92      object.resolution = value.values[0].values[10].value[0].value;
93      object.cov_Increment = value.values[0].values[11].value[0].value;
94      object.high_Limit = value.values[0].values[12].value[0].value;
95      object.low_Limit = value.values[0].values[13].value[0].value;
96    }
97    console.log(' +++  Aggiunto oggetto ', object.object_Name,
98                ' del device numero ', id_dev,'  +++ ');
99
100   struct_object ={object, id_dev};
101   dbBacnet.addPropertyIO(struct_object);
102   id_bacnet_io++;
103 }
```

Figura 3.31: Funzione che definisce i campi nel database.

La particolarizzazione viene gestita dalla condizione all’interno dell’if. Cioè, se il valore associato a type (che definisce il tipo dell’oggetto) è uguale a binaryInput, binaryOutput o BinaryValue, allora i campi non utilizzati verranno settati a 0, altrimenti andranno associati a quelli letti nel device BACnet.

I numeri associati al tipo di oggetto BACnet sono sempre definiti dalla libreria “backstack”, in particolare dal modulo enum.js, come mostrato in figura 3.32.

```
674   module.exports.ObjectTypes = {
675     OBJECT_ANALOG_INPUT: 0,
676     OBJECT_ANALOG_OUTPUT: 1,
677     OBJECT_ANALOG_VALUE: 2,
678     OBJECT_BINARY_INPUT: 3,
679     OBJECT_BINARY_OUTPUT: 4,
680     OBJECT_BINARY_VALUE: 5,
681     OBJECT_CALENDAR: 6,
682     OBJECT_COMMAND: 7,
683     OBJECT_DEVICE: 8,
684     OBJECT_EVENT_ENROLLMENT: 9,
685     OBJECT_FILE: 10,
686     OBJECT_GROUP: 11,
687     OBJECT_LOOP: 12,
688     OBJECT_MULTI_STATE_INPUT: 13,
689     OBJECT_MULTI_STATE_OUTPUT: 14,
690     OBJECT_NOTIFICATION_CLASS: 15,
```

Figura 3.32: Definizione dei valori per definire il Tipo in enum.js.

La funzione in figura 3.33 ha il compito di richiedere ad ogni oggetto presente nella lista object_List del device BACnet le proprietà d’interesse, tramite la funzione “readPropertyMultipleBacnet”.
La funzione “readPropertyMultipleBacnet” richiede, in uno dei parametri, la lista delle proprietà, definite da un id (numeri in giallo). Questi valori sono stati scelti nel modulo enum.js della libreria “backstack”.

Una volta ottenuta la lista con i valori delle proprietà richieste, sarà richiamata la funzione “fillDbIoProp”, spiegata in dettaglio precedentemente.

```
105   // Callback che permette di aggiungere tutti gli oggetti del device una volta ottenuta la object_List:
106   function readObjList(err, objList, device_address, id_dev)
107   {
108     for (var i=0; i<objList.values.length; i++)
109     {  // crea una ricorsione con n cicli quanti sono object_list del device       [{id: 8},{id:9},{id:10},{id:11},{id:12}]
110       readPropertyMultipleBacnet(device_address,
111                                  objList.values[i].value.value.type,
112                                  objList.values[i].value.value.instance,
113                                  [{id:75}, {id:77}, {id:79}, {id:85}, {id:36}, {id:81}, {id:118}, {id:117}, {id:69}, {id:65}, {id:106},
114                                   {id:22}, {id:45}, {id:59}],
115                                  function(err,value)
116                                  {
117                                    fillDbIoProp(err, value, id_dev);
118                                  });
119     }
120   }
121
```

Figura 3.33: Funzione utilizzata per la richiesta dell’object_List in mainBacnet.js.

La funzione “readPropertyBacnet”, una volta richiamata, permette di leggere la proprietà dell’oggetto appartenente al device BACnet, all’interno di essa. Come nelle altre 3 funzioni, si richiama la funzione presente in client.js della libreria “backstack”, passando come parametri:

- l’indirizzo (address) del device;
- il tipo (type) e l’istanza (instance) dell’oggetto appartenente al device BACnet;
- la proprietà che ci interessa leggere, come ad esempio il Present_Value;
- la funzione che vogliamo richiamare una volta che il dato richiesto è disponibile (callback).

Le 4 funzioni presenti in figura 3.34 sono simili, due sono di lettura (read) e due sono di scrittura (write). Inoltre, una seconda particolarizzazione è definita per la lettura/scrittura di una singola proprietà o per la lettura/scrittura di più proprietà (multiple).

```
// Funzione che legge una proprietà del oggetto del device BACnet
function readPropertyBacnet(address, typeid, instance, property, callback)
{
    value = client.readProperty(address, {type:typeid, instance:instance}, property, {}, callback);
}

// funzione che legge più proprietà di un oggetto del device BACnet
function readPropertyMultipleBacnet(address, typeid, instance, list_properties, next)
{
  var requestArray = [
      {objectId: {type: typeid, instance: instance}, properties: list_properties}
      ];
  read = client.readPropertyMultiple( address, requestArray, next);
  return read;
}

// Funzione che permette di modificare una proprietà dell'oggetto del device BACnet
function writePropertyBacnet(address, typeid, instance, property, value, callback)
{
    client.writeProperty(address, {type:typeid, instance:instance}, property, [{type: 9, value: value}], {}, callback);
}

// Funzione che permette di modificare più proprietà dell'oggetto del device BACnet
function writePropertyMultipleBacnet(address, objectIdtype, objectIdinstance, propertyid, propertyindex, valuetype, value, priority)
{
    client.writePropertyMultiple(address,[{objectId: {objectIdtype, objectIdinstance},
        values: [{property: {propertyid, propertyindex }, value: [{valuetype, value}],
        priority}]}],
        function(err, value){
  });
}
```

(1) (2) (3) (4)

Figura 3.34: Funzioni di lettura/scrittura delle proprietà dell'oggetto BACnet.

Di seguito sono riportate le funzioni definite in client.js nella libreria "backstack", che vengono richiamate all'interno del corpo delle funzioni presenti in figura 3.34.

1) ReadProperty (lettura di una singola proprietà dell'oggetto del device BACnet):

```
self.readProperty = function(address, objectId, propertyId, options, next) {
  next = next || options;
  var settings = {
    maxSegments: options.maxSegments || baEnum.MaxSegments.MAX_SEG65,
    maxApdu: options.maxApdu || baEnum.MaxApdu.MAX_APDU1476,
    invokeId: options.invokeId || getInvokeId(),
    arrayIndex: options.arrayIndex || baAsn1.BACNET_ARRAY_ALL
  };
  var buffer = getBuffer();
  baNpdu.encode(buffer,
                baEnum.NpduControls.PRIORITY_NORMAL_MESSAGE | baEnum.NpduControls.EXPECTING_REPLY,
                address,
                null,
                DEFAULT_HOP_COUNT,
                baEnum.NetworkMessageTypes.NETWORK_MESSAGE_WHO_IS_ROUTER_TO_NETWORK,
                0);
  var type = baEnum.PduTypes.PDU_TYPE_CONFIRMED_SERVICE_REQUEST | (settings.maxSegments !== baEnum.MaxSegments.MAX_SEG0 ? baEnum.PduTypes.SEGMENTED_RESPONSE_ACCEPTED : 0);
  baAdpu.encodeConfirmedServiceRequest(buffer, type, baEnum.ConfirmedServices.SERVICE_CONFIRMED_READ_PROPERTY, settings.maxSegments, settings.maxApdu, settings.invokeId, 0, 0);
  baServices.encodeReadProperty(buffer, objectId.type, objectId.instance, propertyId, settings.arrayIndex);
  baBvlc.encode(buffer.buffer, baEnum.BvlcFunctions.BVLC_ORIGINAL_UNICAST_NPDU, buffer.offset);
  transport.send(buffer.buffer, buffer.offset, address);
  addCallback(settings.invokeId, function(err, data) {
    if (err) return next(err);
    var result = baServices.decodeReadPropertyAcknowledge(data.buffer, data.offset, data.length);
    if (!result) return next(new Error('INVALID_DECODING'));
    next(null, result);
  });
};
```

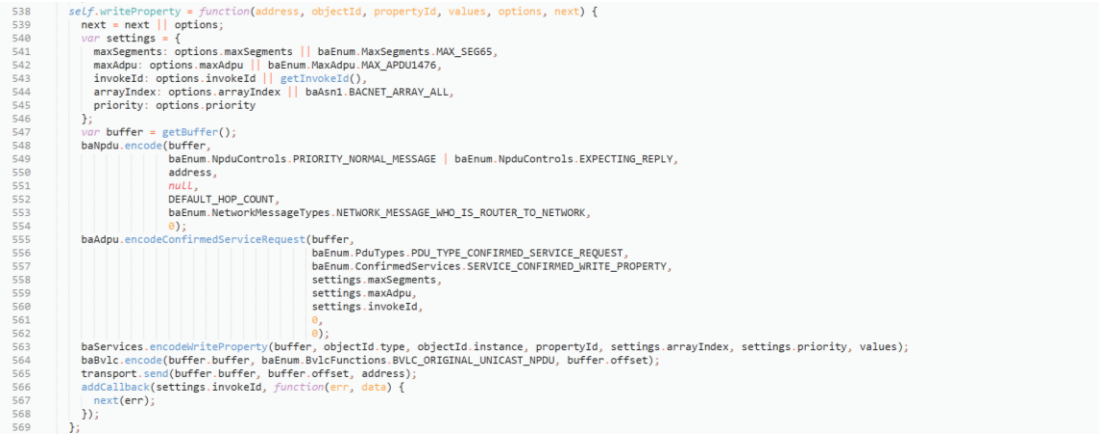
Figura 3.35: Funzione ReadProperty presente in client.js.

2) ReadPropertyMultiple (lettura di una lista di proprietà dell'oggetto del device BACnet):

```
self.readPropertyMultiple = function(address, propertiesArray, options, next) {
  next = next || options;
  var settings = {
    maxSegments: options.maxSegments || baEnum.MaxSegments.MAX_SEG65,
    maxApdu: options.maxApdu || baEnum.MaxApdu.MAX_APDU1476,
    invokeId: options.invokeId || getInvokeId()
  };
  var buffer = getBuffer();
  baNpdu.encode(buffer,
                baEnum.NpduControls.PRIORITY_NORMAL_MESSAGE | baEnum.NpduControls.EXPECTING_REPLY,
                address,
                null,
                DEFAULT_HOP_COUNT,
                baEnum.NetworkMessageTypes.NETWORK_MESSAGE_WHO_IS_ROUTER_TO_NETWORK,
                0);
  var type = baEnum.PduTypes.PDU_TYPE_CONFIRMED_SERVICE_REQUEST | (baEnum.maxSegments !== baEnum.MaxSegments.MAX_SEG0 ? baEnum.PduTypes.SEGMENTED_RESPONSE_ACCEPTED : 0);
  baAdpu.encodeConfirmedServiceRequest(buffer, type, baEnum.ConfirmedServices.SERVICE_CONFIRMED_READ_PROP_MULTIPLE, settings.maxSegments, settings.maxApdu, settings.invokeId, 0, 0);
  baServices.encodeReadPropertyMultiple(buffer, propertiesArray);
  baBvlc.encode(buffer.buffer, baEnum.BvlcFunctions.BVLC_ORIGINAL_UNICAST_NPDU, buffer.offset);
  transport.send(buffer.buffer, buffer.offset, address);
  addCallback(settings.invokeId, function(err, data) {
    if (err) return next(err);
    var result = baServices.decodeReadPropertyMultipleAcknowledge(data.buffer, data.offset, data.length);
    if (!result) return next(new Error('INVALID_DECODING'));
    next(null, result);
  });
};
```

Figura 3.36: Funzione ReadPropertyMultiple presente in client.js.

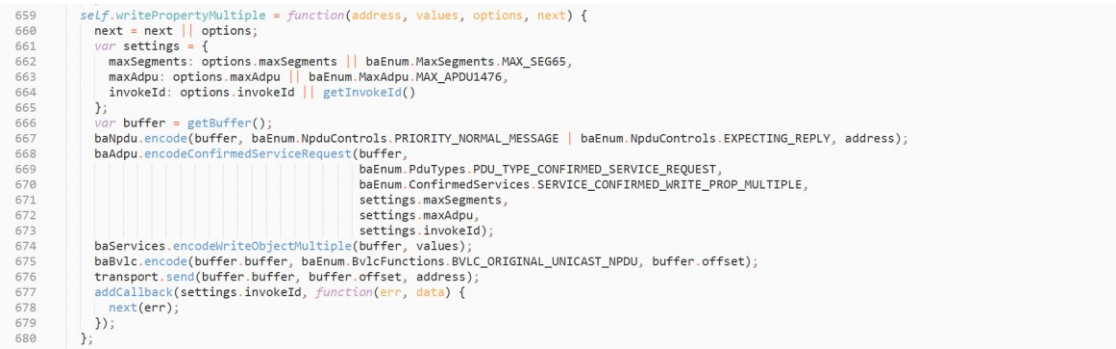
3) WriteProperty (scrittura di una singola proprietà dell'oggetto del device BACnet):



```
self.writeProperty = function(address, objectId, propertyId, values, options, next) {
  next = next || options;
  var settings = {
    maxSegments: options.maxSegments || baEnum.MaxSegments.MAX_SEG65,
    maxApdu: options.maxApdu || baEnum.MaxApdu.MAX_APDU1476,
    invokeId: options.invokeId || getInvokeId(),
    arrayIndex: options.arrayIndex || baAsn1.BACNET_ARRAY_ALL,
    priority: options.priority
  };
  var buffer = getBuffer();
  baNpdu.encode(buffer,
                baEnum.NpduControls.PRIORITY_NORMAL_MESSAGE | baEnum.NpduControls.EXPECTING_REPLY,
                address,
                null,
                DEFAULT_HOP_COUNT,
                baEnum.NetworkMessageTypes.NETWORK_MESSAGE_WHO_IS_ROUTER_TO_NETWORK,
                0);
  baAdpu.encodeConfirmedServiceRequest(buffer,
                                       baEnum.PduTypes.PDU_TYPE_CONFIRMED_SERVICE_REQUEST,
                                       baEnum.ConfirmedServices.SERVICE_CONFIRMED_WRITE_PROPERTY,
                                       settings.maxSegments,
                                       settings.maxApdu,
                                       settings.invokeId,
                                       0,
                                       0);
  baServices.encodeWriteProperty(buffer, objectId.type, objectId.instance, propertyId, settings.arrayIndex, settings.priority, values);
  baBvlc.encode(buffer.buffer, baEnum.BvlcFunctions.BVLC_ORIGINAL_UNICAST_NPDU, buffer.offset);
  transport.send(buffer.buffer, buffer.offset, address);
  addCallback(settings.invokeId, function(err, data) {
    next(err);
  });
};
```

Figura 3.37: Funzione WriteP roperty presente in client.js.

4) WritePropertyMultiple (scrittura di una lista di proprietà dell'oggetto del device BACnet):



```
self.writePropertyMultiple = function(address, values, options, next) {
  next = next || options;
  var settings = {
    maxSegments: options.maxSegments || baEnum.MaxSegments.MAX_SEG65,
    maxApdu: options.maxApdu || baEnum.MaxApdu.MAX_APDU1476,
    invokeId: options.invokeId || getInvokeId()
  };
  var buffer = getBuffer();
  baNpdu.encode(buffer, baEnum.NpduControls.PRIORITY_NORMAL_MESSAGE | baEnum.NpduControls.EXPECTING_REPLY, address);
  baAdpu.encodeConfirmedServiceRequest(buffer,
                                       baEnum.PduTypes.PDU_TYPE_CONFIRMED_SERVICE_REQUEST,
                                       baEnum.ConfirmedServices.SERVICE_CONFIRMED_WRITE_PROP_MULTIPLE,
                                       settings.maxSegments,
                                       settings.maxApdu,
                                       settings.invokeId);
  baServices.encodeWriteObjectMultiple(buffer, values);
  baBvlc.encode(buffer.buffer, baEnum.BvlcFunctions.BVLC_ORIGINAL_UNICAST_NPDU, buffer.offset);
  transport.send(buffer.buffer, buffer.offset, address);
  addCallback(settings.invokeId, function(err, data) {
    next(err);
  });
};
```

Figura 3.38: Funzione WritePropertyMultiple presente in client.js.

La funzione "readValue" in figura 3.39, una volta richiamata, ha il compito di leggere la proprietà del Present_Value dell'oggetto appartentente al device BACnet.

```
154   // Funzione che mi permette di leggere il present_Value di un oggetto:
155   function readValue(type, propInstance, address, callback)
156   {
157       // Lettura del Present Value dall'oggetto
158       readPropertyBacnet(address, type, propInstance, 85, function(err, value){
159         if (callback != 0)
160         {
161             callback(value.values[0].value);
162         }
163       });
164   }
```

Figura 3.39: Funzione che legge il Present_Value di un oggetto BACnet.

La funzione “writeValue” in figura 3.40 ha l’obbiettivo di scrivere il Present_Value nell’oggetto del device BACnet desiderato.

```
166   // funzione che mi permette di scrivere il present_Value di un oggetto:
167   function writeValue(type, propInstance, address, value)
168   {
169       // Scrittura del Present Value dell'oggetto
170       writePropertyBacnet(address, type, propInstance, 85, value, function(err,valueWrite){
171       });
172   }
```

Figura 3.40: Funzione che scrive il Present_Value di un oggetto BACnet.

Per terminare la spiegazione del modulo mainBacnet.js, occorre notare le funzioni esportate dal modulo stesso nel dbBacnet.js, in modo tale che cost’ultimo possa utilizzare le funzioni “readValue” e “writeValue” definite nel mainBacnet.js.

```
174   // Funzioni del mainBacnet.js esportate:
175
176   exports.readValue = readValue;    // esporto la funzione di lettura del present Value
177   exports.writeValue = writeValue;  // esporto la funzione di scrittuta del present Value
178
```

Figura 3.41: Funzione che vengono esportate dal mainBacnet.js.

3.3) Server BACnet

Il Server rappresenta il device BACnet, cioè qualsiasi dispositivo che permette la comunicazione secondo il protocollo BACnet.
La sua mansione è quella di rispondere alle richieste del Client BACnet, ossia al driver spiegato nel capitolo precedente.
Nel mio caso, il Server è servito per effettuare le prove di valutazione sull'effettivo funzionamento del driver BACnet. Queste prove servono per verificare la risposta effettiva del device alla scansione della rete (risposta I-Am da parte del Server) e, successivamente, per leggere e scrivere i valori delle proprietà degli oggetti appartenenti al device BACnet.
Per fare ciò, sono stati utilizzati due tipi di Server:

1. **BACsim** (Simulatore di device BACnet):

   BACsim è il primo simulatore di dispositivi BACnet, di facile utilizzo per Windows. BACsim funziona con Windows 7/8/10 e Server 2012 e offre un solido set di funzionalità di base per i simulatori costruiti su misura per le esigenze dei progettisti e dei tester BACnet.
   Esso può simulare oggetti BACnet standard come ingressi analogici, uscite analogiche, ingressi binari ed uscite binarie ecc.. Sono supportati tutti i tipi di oggetto nello standard 135-2012.

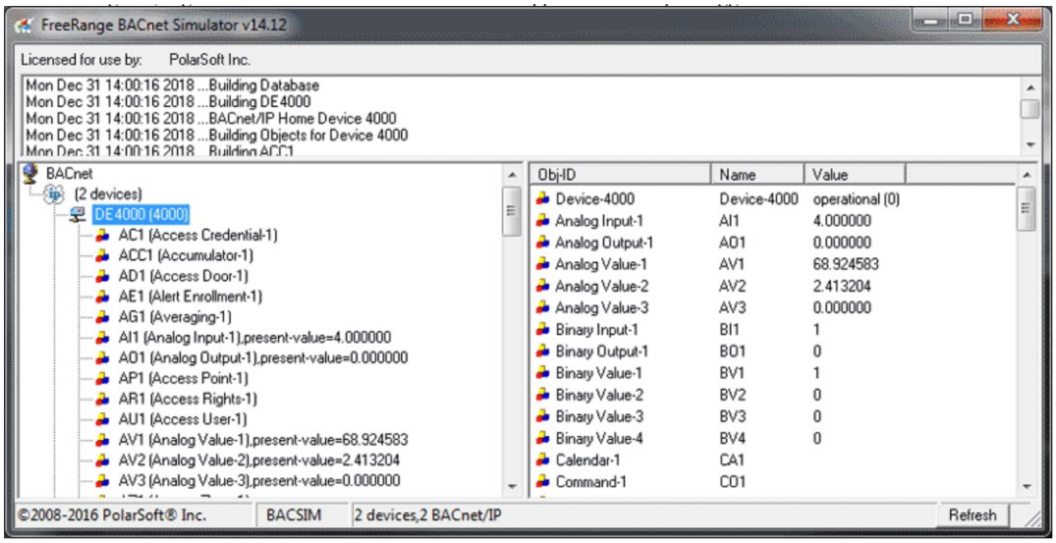


Figura 3.42: Interfaccia utente di BACsim.

BACsim fornisce una semplice interfaccia utente, che mostra gli oggetti e le loro proprietà in una gerarchia ad albero simile a Esplora risorse.

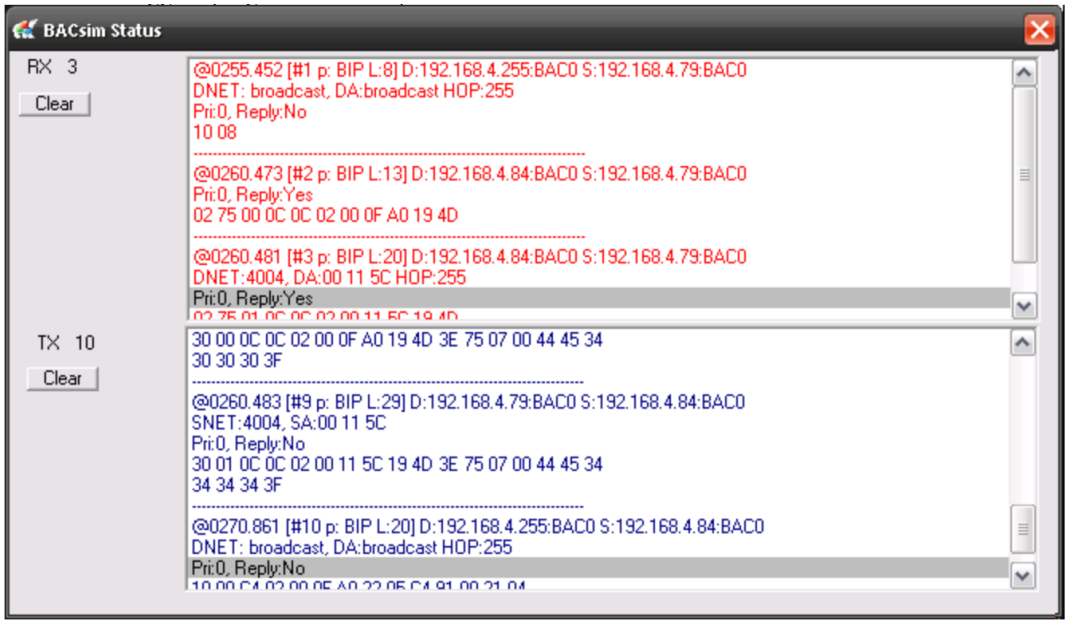


Figura 3.43: Interfaccia visualizzazione traffico di dati di BACsim.

Si può, anche, visualizzare il traffico di messaggi in entrata e in uscita con una funzione di traccia integrata che visualizza i messaggi in forma esadecimale, in modo da poter vedere la codifica effettiva.

2. **PLC Beckhoff CX5130-0155** (con libreria **Beckhoff TcBACnet-Lib**):

Il CX5130 ha un processore multi-core Intel Atom con una frequenza di clock di 1,75 GHz. Ciò rende possibile un'autentica tecnologia multi-core nel segmento PC incorporato. Le interfacce hardware di questa serie sono orientate e implementate in modo identico a quelle della serie CX5000 esistente.

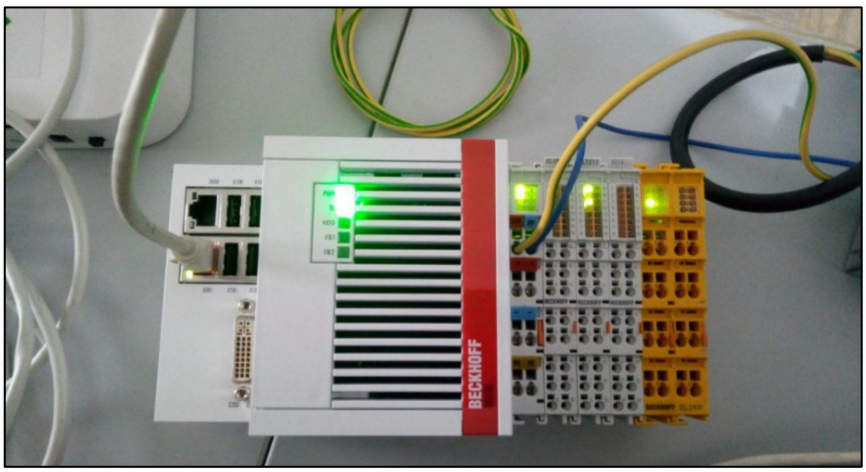


Figura 3.44: Immagine frontale del PLC CX5130.

Sono disponibili due interfacce Ethernet indipendenti, compatibili con Gigabit, quattro USB 2.0 e un'interfaccia DVI-I. Una moltitudine di ulteriori opzioni di connessione e funzioni

gateway viene creata da un'interfaccia opzionale. Il CX5130 può essere pre-equipaggiata di fabbrica o dal livello I/O e può facoltativamente essere costituito da terminali E-bus e K-bus. Esso è caratterizzato da un basso consumo energetico e un design senza ventole.
A seconda dell'ambiente di run-time TwinCAT installato, CX5130 può essere utilizzato per l'implementazione di progetti PLC o PLC/Motion Control, con o senza visualizzazione.
È, anche, possibile eseguire applicazioni Motion Control con movimenti di assi interpolati. L'ampio intervallo di temperatura di funzionamento da -25 a +60 °C consente l'utilizzo in ambienti climaticamente difficili. Come la CX5000, la serie CX5100 ha un design compatto e un dispositivo modulare, con moduli di estensione che nella serie CX2000 non è disponibile.

Technical data	CX5130
Processor	Intel Atom® E3827, 1.75 GHz
Number of cores	2
Flash memory	slot for CFast card and microSD card, cards not included
Main memory	4 GB DDR3 RAM (not expandable)
Persistent memory	integrated 1-second UPS (1 MB on CFast card)
Interfaces	2 x RJ45 10/100/1000 Mbit/s, 1 x DVI-I, 4 x USB 2.0, 1 x optional interface
Diagnostics LED	1 x power, 1 x TC status, 1 x flash access, 2 x bus status
Clock	internal battery-backed clock for time and date (battery exchangeable)
Operating system	Microsoft Windows Embedded Compact 7 (supports only one CPU core), Microsoft Windows Embedded Standard 7 P, Microsoft Windows 10 IoT Enterprise LTSB
Control software	TwinCAT 2 runtime TwinCAT 3 runtime (XAR)
I/O connection	E-bus or K-bus, automatic recognition
Power supply	24 V DC (-15 %/+20 %)
Current supply E-bus/K-bus	2 A
Max. power loss	11 W (including the system interfaces)
Dimensions (W x H x D)	142 mm x 100 mm x 92 mm
Weight	approx. 960 g
Operating/storage temperature	-25…+60 °C/-40…+85 °C
Relative humidity	95 %, no condensation
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4
Protection class	IP 20
Approvals/markings	CE, UL, ATEX, IECEx
Ex-Marking	II 3 G Ex nA IIC T4 Gc II 3 D Ex tc IIIC T135 °C Dc Ex nA IIC T4 Gc Ex tc IIIC T135 °C Dc
TC3 performance class	Performance (40); please see here for an overview of all the TwinCAT 3 performance classes

Figura 3.45: Dati tecnici del PLC CX5130.

Il PLC CX5130 è stato predisposto insieme alla libreria TcBACnet-Lib, con l'obbiettivo di conferirgli la capacità di poter comunicare tramite protocollo BACnet con un Client BACnet. La libreria offre un comodo accesso agli oggetti di una configurazione BACnet da programma PLC:

- i blocchi funzione o le variabili auto-dichiarate sono collegati con l'ausilio della mappatura "sincrona" dei dati di processo (collegamento del PLC con la configurazione hardware nel System Manager).
  A tale scopo, i blocchi funzioni del PLC comprendono l'allocazione come dati di input o output ("bValue AT% I *: BOOL" o "bValue AT% Q *: BOOL").
  Il vantaggio è che i dati vengono aggiornati ciclicamente e tutti i valori vengono scritti o letti in modo sincrono con il ciclo del PLC;
- i dati degli oggetti vengono letti o scritti in modo asincrono con l'aiuto di ADS. I dati da scrivere/leggere vengono inviati all'indirizzo corrispondente con l'aiuto di blocchi ADS (NetId, porta, gruppo di indice e offset -> tipo di oggetto, istanze e ID proprietà), scritti da esso (FB_BACnet_WriteProp) o letti (FB_BACnet_ReadProp). Questo metodo ha il vantaggio di poter scrivere dati solo se si è verificato un cambiamento effettivo. Tuttavia, nei casi con frequenti cambiamenti, possono verificarsi ritardi non prevedibili. Il ritardo massimo è di 5 secondi ("tBACnet_ADSTimeOut: TIME:= t #5s"). Trascorso questo tempo il blocco restituisce un codice errato.

La libreria TcBACnet-Lib è una raccolta di blocchi funzionali per la programmazione di un controller BACnet.
Per ogni oggetto BACnet sono disponibili blocchi funzione, che vengono letti e scritti tramite le proprietà selezionate. In generale, viene fatta una distinzione tra blocchi funzioni per oggetti Server e Client. I blocchi funzione per gli oggetti Client hanno il permesso "Remoto", poiché consentono l'accesso agli oggetti BACnet remoti su altri dispositivi. I blocchi funzione oggetto BACnet verificano se i dati di processo sono validi e se il controller BACNet è pronto per il funzionamento(bReady).
Il blocco funzionale FB_BACnet_Adapter rappresenta un dispositivo BACnet, cioè l'accesso alla rete BACnet tramite scheda di rete. Questo blocco consiste, ad esempio, di verificare l'esistenza di un collegamento. Al fine di rendere il programma PLC più trasparente, l'adattatore BACnet viene creato con variabili globali all'interno di TcBACnet-Lib, poiché nella maggior parte dei progetti viene utilizzato un singolo adattatore BACnet.
A seconda della funzionalità client e server, i blocchi funzione FB_BACnet_Device o FB_BACnet_RemoteDevice possono essere utilizzati per collegare le informazioni sullo stato e il controllo riguardante il Client e il Server BACnet.
La disponibilità operativa degli oggetti BACnet è determinata dallo stato del Client, del Server e della proprietà System_Status per il dispositivo remoto.
Ogni blocco funzione oggetto BACnet deve quindi trasferire un riferimento all'istanza corrispondente FB_BACnet-Device o FB_BACnet_RemoteDevice, al fine di abilitare la valutazione dello stato(bReady).

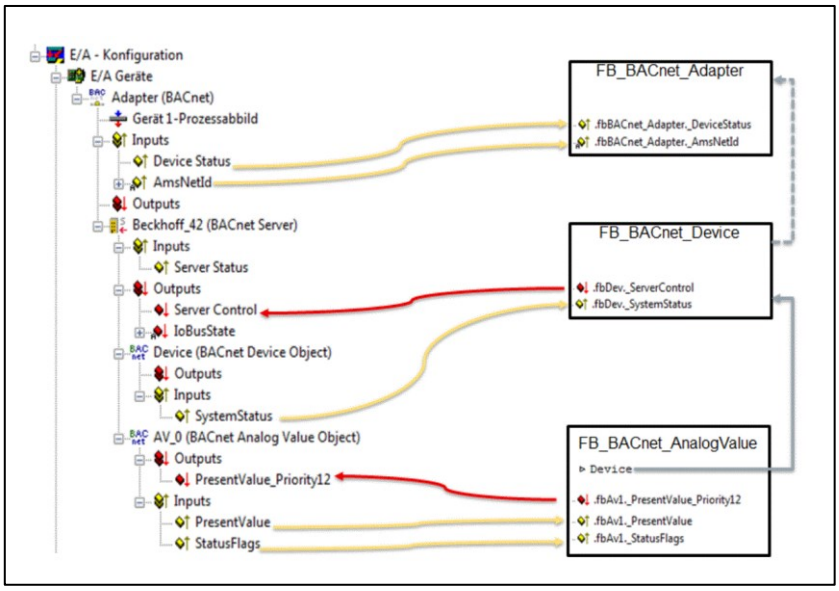


Figura 3.46: Blocchi funzione della libreria TcBACnet-Lib.

La figura 3.46 mostra una panoramica dei blocchi funzionali di TcBACnet-Lib e dei collegamenti corrispondenti con i moduli BACnet.
Il collegamento tra un programma PLC basato su TcBACnet-Lib e i moduli BACnet di una configurazione può essere automatizzato tramite la funzione Twincat di "automapping".

3.4) Funzionamento del driver BACnet

Il driver BACnet, come detto in precedenza, ha l'obbiettivo di riconoscere tutti gli oggetti connessi in modalità TCP/IP di tipo BACnet, quindi di comunicare con loro e successivamente andando a salvare sul database MySQL le caratteristiche dei device BACnet.
Per attivare il driver BACnet occorre aprire il terminale del proprio sistema (PC, Raspberry, ecc.) e una volta entrati nella directory in cui si trova la cartella "driver_BACnet" che contiene i file Nodejs del driver BACnet, digitare sul terminale 'node mainBacnet.js', come mostrato in figura 3.47:

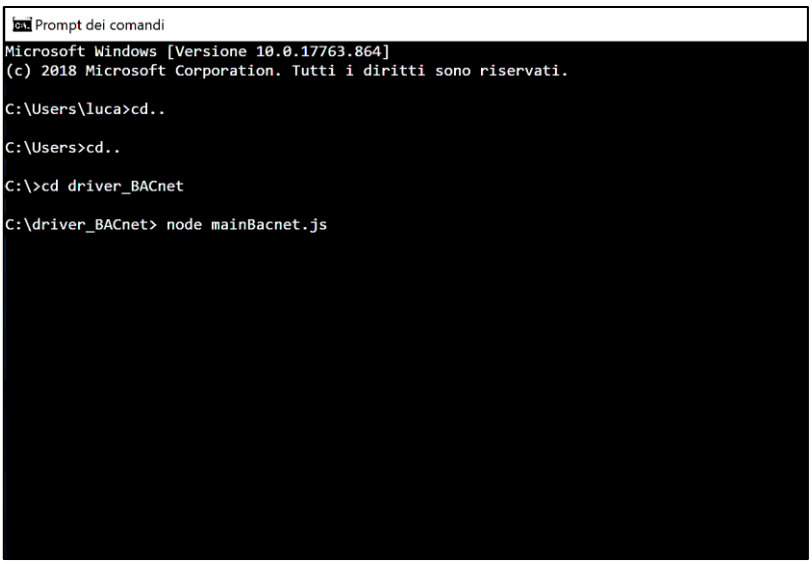


Figura 3.47: Attivazione driver BACnet.

Essendo il mainBacnet.js dipendente dall'altro script dbBacnet.js, verranno attivati entrambi, restituendo nel terminale la seguente nomenclatura mostrata in figura 3.48:

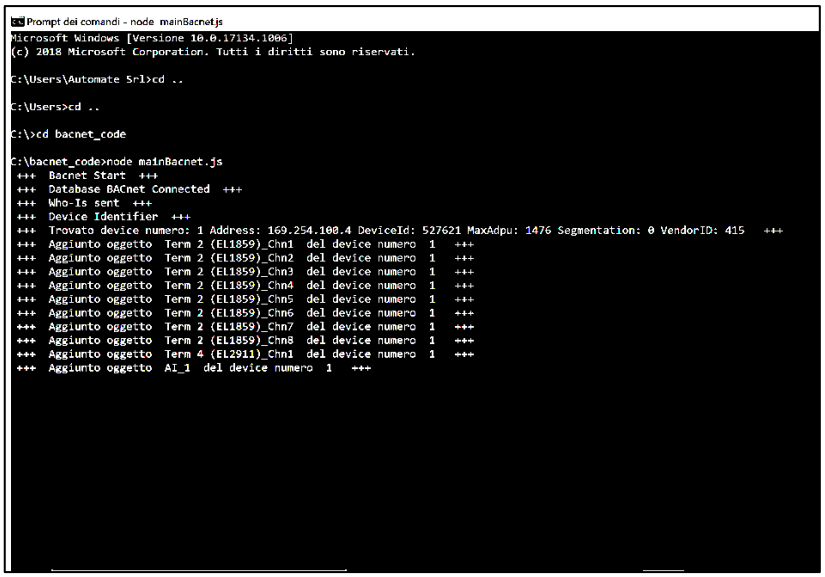


Figura 3.48: Risultato ottenuto sul terminale dopo l'avvio del driver BACnet.

3.4.1) Salvataggio dei dati

Per il salvataggio dei dati nel nostro Gateway è stato utilizzato il database MySQL Workbench, utilizzato per creare le tabelle che andranno ad accogliere i valori dei dispositivi BACnet.

MySQL Workbench

MySQL Workbench è uno strumento visuale di progettazione per database, che integra sviluppo SQL, gestione, modellazione dati, creazione e manutenzione di database MySQL all'interno di un unico ambiente sinergico. Inoltre, permette l'accesso e la gestione a database remoti attraverso vari protocolli, tra i quali anche SSH. È il successore di DBDesigner 4 prodotto da fabFORCE.net e sostituisce il precedente pacchetto software MySQL Administrator Bundle.

La struttura scelta per le tabelle del database BACnet è visibile in figura 3.49, in cui la prima tabella "devices_bacnet" sarà definita dalle proprietà che identificano il device BACnet in fase di scansione della rete, le quali permetteranno successivamente di richiedere valori e/o modifiche al device.
La seconda tabella, "bacnet_io", andrà ad accogliere tutte le proprietà di ogni singolo input/output presente nel device BACnet. Questi campi della seconda tabella saranno settati in maniera differente dal driver BACnet in relazione al tipo di sensore di input/output. Cioè, se il sensore è analogico, tutti i campi saranno riempiti con il relativo valore del I/O BACnet, altrimenti, se il sensore è digitale, allora alcuni campi verranno settati direttamente a "null", in quanto non identificati dal I/O BACnet.
Di particolare importanza è l'id device, che risulta avere un valore univoco diverso per ogni device assegnato dal driver BACnet, con il vantaggio di poter selezionare tutti gli I/O dell'n-esimo device effettuando una query sulla seconda tabella.

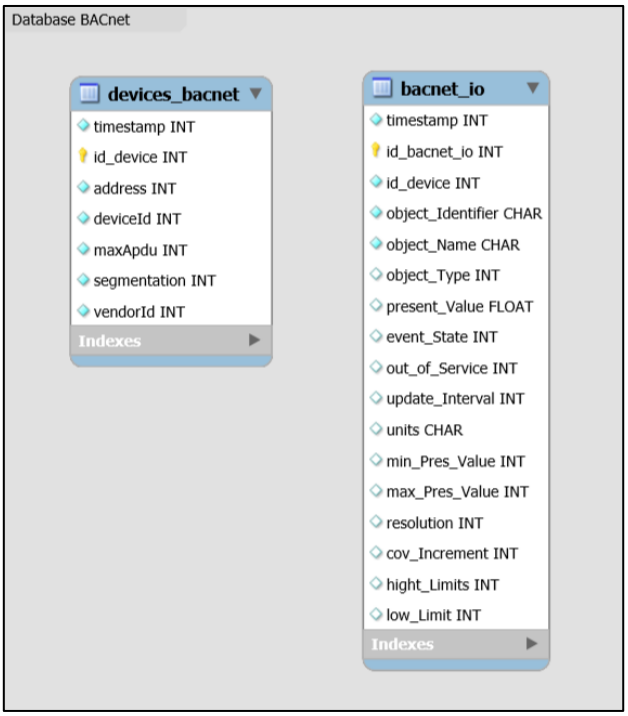


Figura 3.49: Struttura del database BACnet.

Da notare che nelle tabelle del database MySQL è presente una colonna, denominata "timestamp", che identifica quando è stato salvato quel particolare device e/o oggetto.
La serie di numeri identifica il tempo che viene rappresentato come offset in secondi rispetto alla mezzanotte (UTC) del 1º gennaio 1970. Questo tipo di rappresentazione, oltre che a essere compatta, è indipendente dai fusi orari, ed è quindi direttamente confrontabile anche tra calcolatori situati a grandi distanze geografiche tra loro. Ciò evita di dover effettuare aggiustamenti nel caso, per esempio, di dati trasmessi da un fuso orario all'altro.
Lo svantaggio è che, per averne una rappresentazione sotto forma di data e ora locali, occorre effettuare una conversione. Il sistema operativo offre, allo scopo, delle funzioni che consentono di effettuarle in entrambe le direzioni.

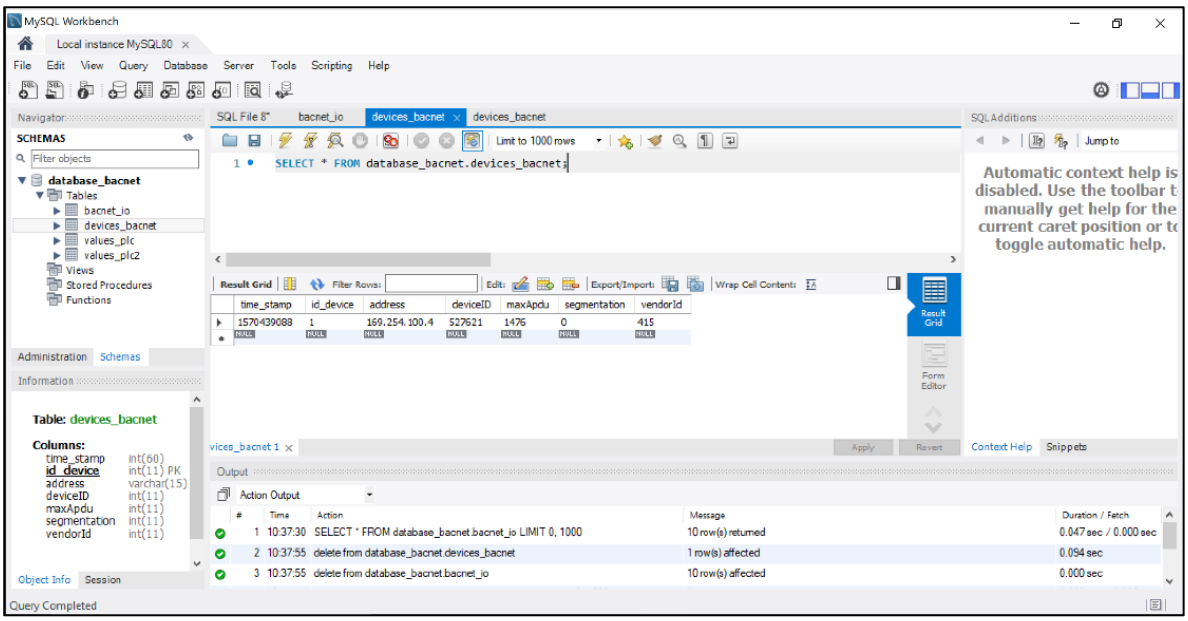


Figura 3.50: Salvataggio dei device BACnet nel database MySQL.

Una volta salvato il dispositivo nella tabella relativa ai device BACnet, andiamo a richiedere la lista degli oggetti del device BACnet e ad aggiungerli nella tabella “bacnet_io” insieme a tutte le sue proprietà, come mostrato in figura 3.51:

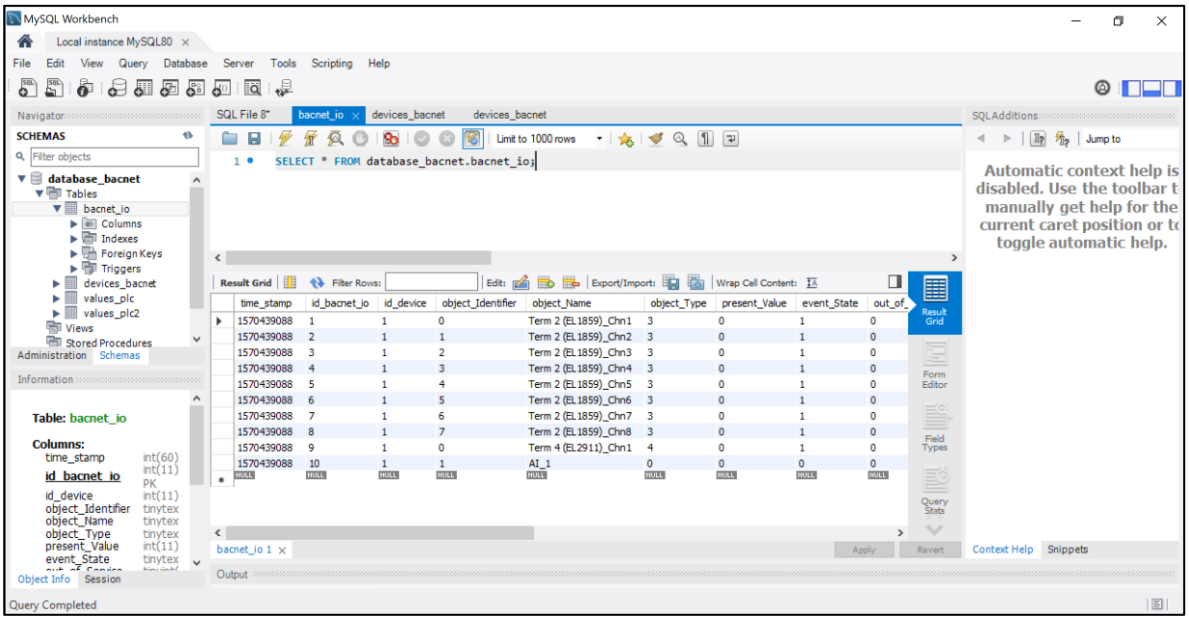


Figura 3.51: Salvataggio delle proprietà degli I/O del device BACnet nel database MySQL.

3.4.2) Risultati ottenuti

I test sul corretto funzionamento sono stati effettuati prima sul simulatore BACsim, in modo tale da non compromettere il funzionamento di alcun device fisico, e, successivamente, collegandoci tramite cavo ethernet (TCP/IP) ad un PLC Beckhoff, predisposto ad accogliere la comunicazione del protocollo BACnet tramite l’installazione al suo interno della libreria, come visto nel Capitolo 3.3.
In entrambi i casi, sono stati riscontrati risultati ottimi, sia in fase di robustezza del codice Nodejs sviluppato (quindi privo di malfunzionamenti o errori) sia sulla mancata perdita d’informazione derivante dall’ utilizzo del driver BACnet.

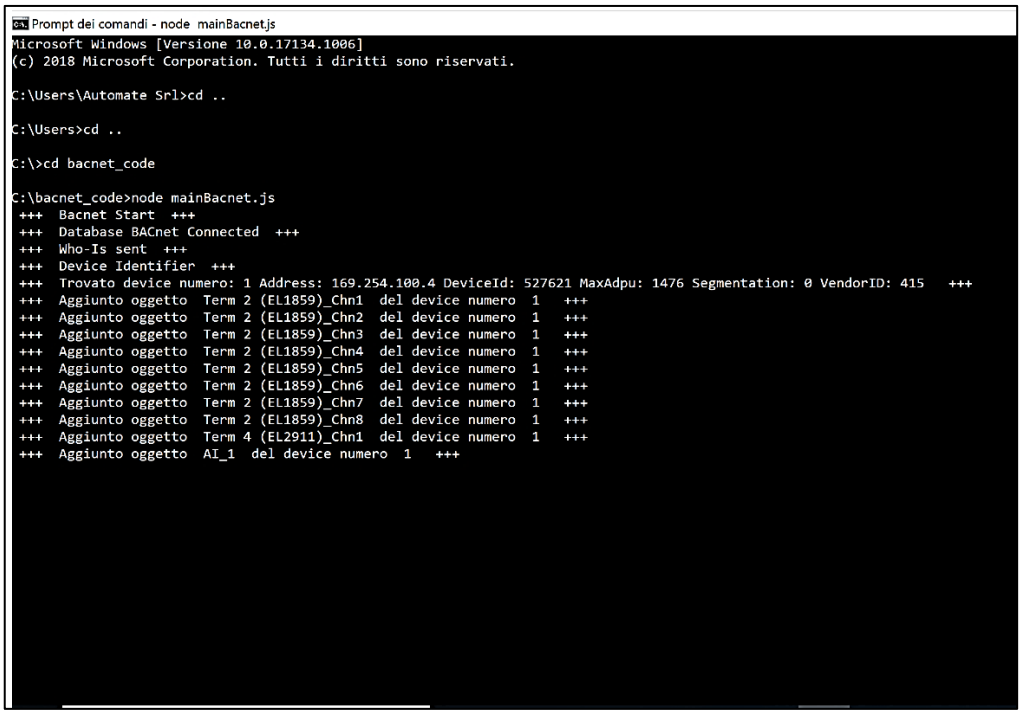


Figura 3.52: Risultato ottenuto sul terminale dopo l'avvio del driver BACnet.

La prova è stata effettuata cambiando i valori dal PLC Beckhoff con l'ausilio del software TwinCat3 ed andando a valutare l'effettivo cambiamento del valore nel database MySQL.
Come si può notare dall'immagine precedente, le prime informazioni riguardano l'effettiva attivazione del driver BACnet (+++ Bacnet Start +++) e la corretta connessione con il database MySQL (+++ Database BACnet Connected +++).
La fase successiva è quella di inviare la richiesta di Who-Is nella rete, per aspettare poi una risposta da parte di eventuali device BACnet collegati, verificato con la stampa di +++ Who-Is Send +++. Una volta che il device BACnet (Server) ci risponde, andiamo a visualizzare le sue caratteristiche primarie, come l'address, deviceId, MaxApdu, Segmentation e VendorID.
In ultimo, andremo a stampare gli oggetti che appartengono a quel device, esplicitando il nome e il tipo (analogico, digitale, ecc.).
L'effettivo funzionamento del driver BACnet è dato, come prima cosa, dal salvataggio di tutti i valori nel database MySQL, e poi nelle 2 tabelle "devices_bacnet" e "bacnet_io", come mostrato nelle figure 3.50 e 3.51.
Infine, come si può notare dalla figura 3.53, nel sottolivello evidenziato in blu vi sono tutti gli oggetti del Device BACnet (in questo caso del PLC Beckhoff CX5130), i quali risultano essere gli stessi che visualizziamo sul terminale dopo aver avviato il driver BACnet, come mostrato in figura 3.54.

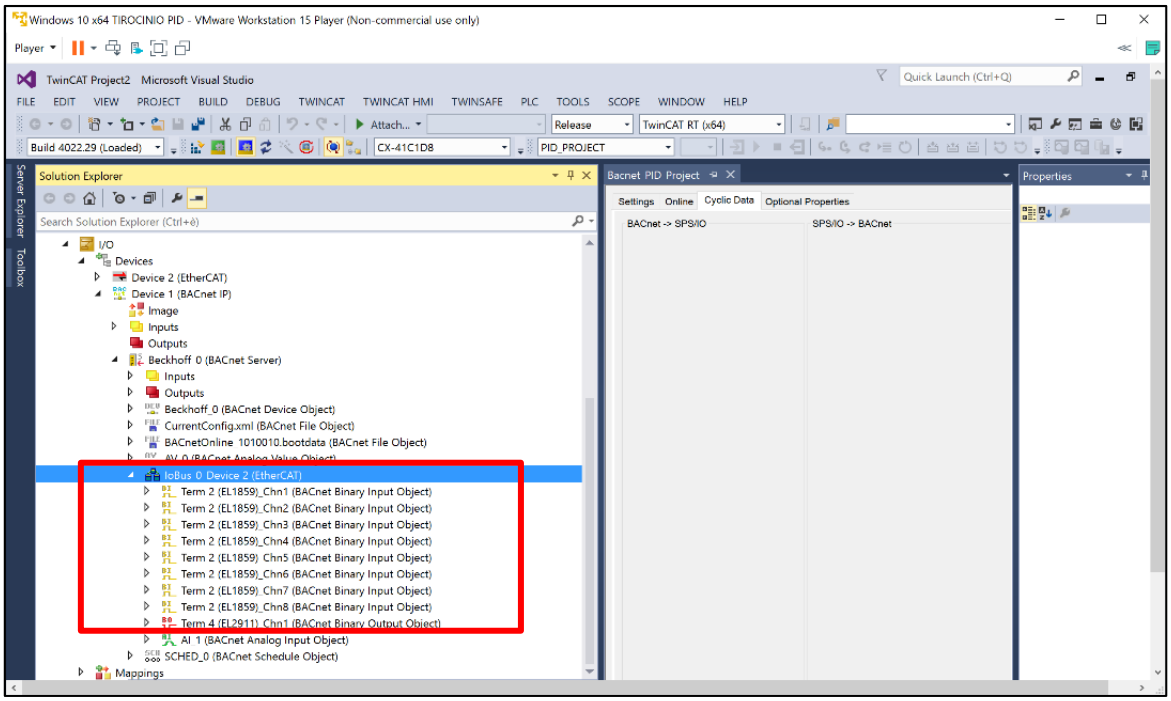


Figura 3.53: Oggetti del device BACnet su Twincat3.

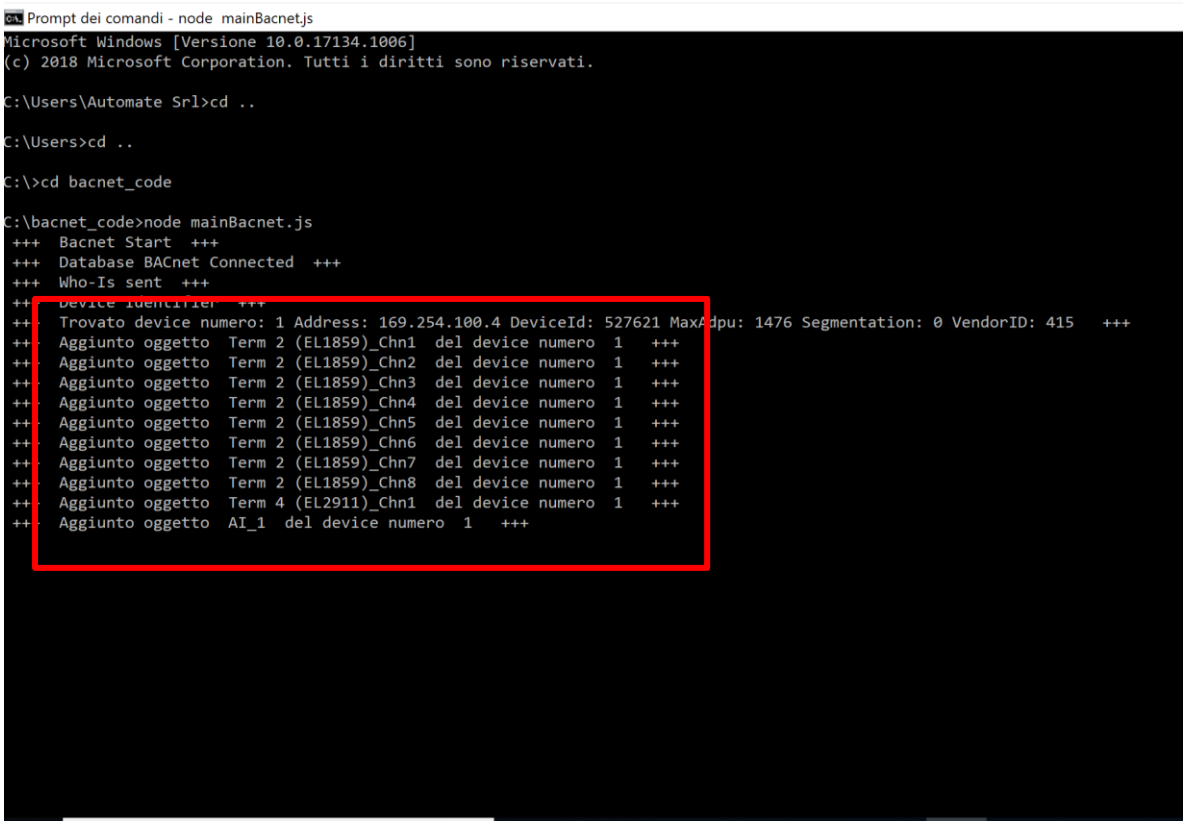


Figura 3.54: Risultato ottenuto sul terminale dopo l'avvio del driver BACnet.

4) PID e Tuning adattivo dei parametri Kp, Ki e Kd.

4.1) Introduzione

L'argomento trattato all'interno di questa sezione riguarda il mondo dei sistemi industriali (catene di produzione, controlli automatici, reti di sensori, ecc.), ognuno dei quali può essere schematizzato come un blocco generico che richiede, in ingresso, una sollecitazione di un certo tipo, al fine di poter produrre, secondo il suo utilizzo, un'uscita adeguata. Per fare in modo che il processo lavorativo di questi sistemi si attenga alle specifiche desiderate, è sempre necessario che ci sia un controllo su quello che arriva al sistema e su quello che esso produce come risultato (anche se stiamo parlando di sistemi automatici).
Per questa ragione ci si affida a dei sistemi di controllo, i quali scambiano informazioni con il/i processo/i fisico/i (attraverso sensori e attuatori) o con un operatore umano o con altri sistemi automatici.
Tali sistemi di controllo realizzano, in maniera automatica, gli algoritmi necessari affinché il comportamento del processo fisico sia quello desiderato.
Un dispositivo di controllo è un particolare sistema per l'elaborazione dell'informazione, destinato al controllo dei processi fisici, il quale deve potersi interfacciare con l'ambiente esterno. Generalmente, le sue funzionalità devono essere quelle di controllo a ciclo chiuso classico del sistema (regolazione o asservimento), di calcolo dei valori di riferimento (set-point), di gestione di eventuali allarmi ed anomalie ed, infine, di realizzazione dell'interfaccia di comunicazione con operatore o altri dispositivi.
Al giorno d'oggi esistono molti tipi di sistemi di controllo, più o meno sofisticati, realizzati con diverse tecniche (alcuni attraverso dispositivi informatici ma altri ancora secondo tecnologie idrauliche e pneumatiche), ma qui ci occuperemo di una particolare tipologia chiamata PID.
Questi regolatori standard tempo invarianti molto diffusi utilizzano tre tipi di leggi di controllo diverse: azione Proporzionale, azione Integrale, azione Derivativa. Da qui l'acronimo PID, che identifica questi particolari sistemi di controllo.
Benché utilizzino una tecnologia realizzativa antiquata per gli standard tecnologici attuali (possono infatti essere di tipo meccanico, idraulico o anche elettronico e digitale), questi controllori vengono ancora diffusamente utilizzati. soprattutto per la relativa semplicità di taratura dei loro parametri di lavoro. È, infatti, ragionevole ritenere che il controllore PID, nelle sue innumerevoli varianti, gestisca attualmente circa il 95% degli anelli di regolazione presenti negli impianti.
Il successo dei regolatori PID è legato a diversi fattori, alcuni tipicamente industriali:

- notevole efficacia nella regolazione di un'ampia gamma di processi industriali, anche in relazione alle specifiche di prestazione, non sempre stringenti;
- relativa semplicità di taratura (rapportata anche alle capacità degli utenti e alla difficoltà del compito);
- importanza e convenienza economica della standardizzazione (per l'utilizzo, la robustezza e l'affidabilità, il costo di progetto, produzione e manutenzione, la gestione delle scorte);
- prestazioni scadenti di molti sistemi di controllo, dovute a problemi nei sensori e negli attuatori. Rumore e filtri anti-aliasing inadeguati, errori di calibrazione dei sensori, eccessivo filtraggio nei sensori "smart", cattivo dimensionamento, isteresi e attriti statici nelle valvole di regolazione sono le cause più citate. In presenza di problemi di tale

natura, il ruolo della legge di controllo può diventare modesto, per cui viene meno la motivazione di cercare leggi di controllo più sofisticate di quelle PID;

- uso frequente come elementi di schemi di controllo più complessi e articolati, quale il controllo in cascata, che, sfruttando la conoscenza di specifiche proprietà dinamiche del processo, possono portare notevolissimi miglioramenti delle prestazioni, a fronte di costi e sforzi realizzativi relativamente modesti;
- sfruttamento completo degli algoritmi di controllo più complessi. É necessaria una conoscenza approfondita del processo da controllare, il che richiede investimenti non trascurabili e con ritorni non quantificabili a priori, sia per lo sviluppo, sia per la messa a punto, sia per la validazione di modelli matematici accurati.

In sintesi, si può forse affermare che i PID hanno successo perché rappresentano una soluzione non facilmente superabile, in generale, nel rapporto efficacia/costo (globale).
Nella trattazione seguente sui PID si analizzerà la loro struttura, descrivendo nel dettaglio il funzionamento di ognuna delle parti che li compongono e da cui prendono il nome. Si vedranno pregi e difetti di ognuna di esse ed anche i problemi tecnici che sorgono al momento del loro utilizzo pratico nelle diverse situazioni di lavoro. In esse possiamo trovare l'effettiva realizzabilità del dispositivo e l'instabilità creata da ingressi indesiderati e da aumenti esponenziali, dovuti al fenomeno del windup.
Per ognuno di questi problemi verranno esaminate anche le possibili soluzioni, studiandone l'efficacia e il comportamento.

4.2) Controllori PID

Nel passato i regolatori e le trasmissioni dei segnali venivano realizzati con tecnologie meccaniche ed elettroniche (con l'ausilio di circuiti analogici contenenti resistenze, condensatori o amplificatori operazionali), ma dall'inizio degli anni '80 assistiamo all'avvento dei primi controllori a microprocessore che, rispetto ai loro predecessori, presentano maggiori possibilità di elaborazione e una maggiore precisione e stabilità nel tempo.
La funzione che contraddistingue questi regolatori industriali è quella di assicurare un controllo accurato della grandezza che si desidera manipolare, supposta misurabile. Gli stessi regolatori assumono anche altre funzioni, come la possibilità di fungere da allarmi o da interfacce tra processo e operatore.
Il termine "controllo" definisce l'azione svolta per portare, e mantenere, ad un valore prefissato, un parametro fisico di un impianto o di un processo (ad esempio, la temperatura di un forno, il livello di un fluido, la posizione del braccio di un robot, la velocità di rotazione di un motore, l'automatismo che regola il servomotore, ecc.).
Nella categoria dei regolatori standard tempo invarianti sono molto diffusi dei particolari dispositivi che utilizzano tre tipi di leggi di controllo diverse. Essi vengono chiamati PID, acronimo che deriva dalle tre azioni svolte da questi dispositivi, che sono rispettivamente: azione Proporzionale (Kp), azione Integrale (Ki) e azione Derivativa (Kd).

I regolatori PID sono dei regolatori standard che, per mezzo di dispositivi di correzione (manopole o software di ausilio), permettono di regolare i parametri del sistema di controllo entro ampi limiti, così da poter essere adattati al particolare sistema di regolazione in cui vengono inseriti.

Da quanto detto, si capisce quindi che questi dispositivi possono essere utilizzati sia per controllare una portata, sia per controllare una temperatura (la quale ha una risposta molto più lenta della prima).
La funzione di controllo esercitata dai PID risulta essere molto grossolana. Ciononostante, essi rivestono un ruolo di fondamentale importanza (per esempio negli impianti chimici e petrolchimici), in primo luogo perché risulta economicamente conveniente unificare gli apparati di controllo, ed inoltre perché permettono di controllare processi per i quali non si riesce a trovare un modello.
Diamo una semplice e schematica rappresentazione di un regolatore PID, facendo riferimento alla figura 4.1, dove compaiono i seguenti segnali:

- r(t): segnale di riferimento in ingresso;
- y(t): segnale di uscita del sistema di controllo in retroazione;
- e(t): errore dovuto alla differenza algebrica tra il segnale di riferimento in ingresso r(t) e l'uscita y(t);
- u(t): ingresso di controllo.

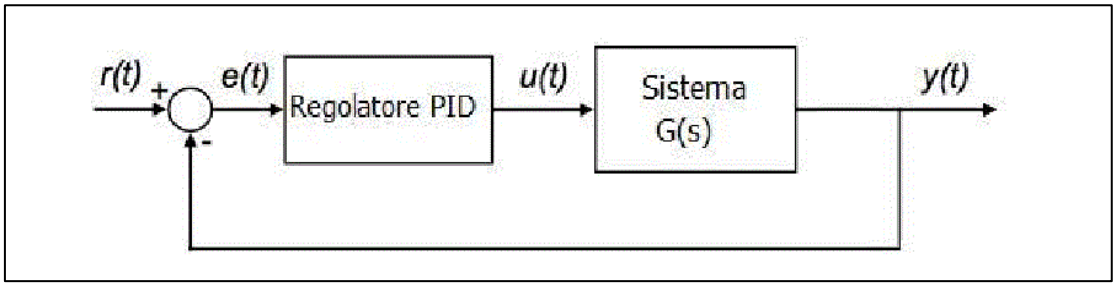


Figura 4.1: Schema a blocchi di un regolatore PID.

Nella figura 4.1 possiamo notare come il controllore venga posto in serie al sistema da controllare (ovviamente prima di esso, perché il controllo abbia effetto sulla grandezza in ingresso al sistema) e come il circuito di retroazione riporti in ingresso il termine y(t), il quale, attraverso un nodo sommatore (che in questo caso assume il ruolo di sottrattore), va a modificare il segnale vero e proprio d'ingresso del controllore. Inoltre, completiamo lo schema di figura 4.1 con la legge di controllo, cioè il legame tra e(t) e u(t), del nostro regolatore

$$u(t) = K_P e(t) + K_I \int_{t_0}^{t} e(\tau)d\tau + K_D \frac{de(t)}{dt}$$

dove i parametri Kp, Ki e Kd rappresentano, rispettivamente, i guadagni delle azioni proporzionale, derivativa e integrale. In realtà è più comunemente usata la seguente formula

$$u(t) = K_P \left(e(t) + \frac{1}{T_I} \cdot \int_{0}^{t} e(\tau)d\tau + T_D \cdot \frac{de(t)}{dt} \right)$$

in cui, a fianco al parametro Kp, si evidenziano le costanti di tempo delle azioni derivativa e integrale, rispettivamente Td e Ti. Chiaramente, i parametri Kd e Ki sono legati ai parametri Td e Ti dalle relazioni:

$$C_D = K_D s \qquad C_I = K_I / s.$$

È facile verificare che i PID, almeno nella loro forma ideale, sono sistemi dinamici SISO, lineari, stazionari, a tempo continuo e impropri. Applicando la trasformata di Laplace alla forma ideale sopra enunciata con $t_0 = 0$, si deduce immediatamente che essi sono descritti dalla funzione di trasferimento

$$C_{PID} = K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s}.$$

I parametri che identificano il PID dunque sono Kp, Ti (o Ki) e Td (o Kd) ed essi sono chiamati anche gradi di libertà del controllore.

Vediamo ora più nel dettaglio una per una le azioni descritte precedentemente.

4.3) Azione Proporzionale

Quando parliamo di azione proporzionale significa che l'ingresso e(t) e l'uscita u(t) sono legati algebricamente da un coefficiente K_P, detto anche coefficiente dell'azione proporzionale, o semplicemente guadagno proporzionale.

$$u(t) = K_P \cdot e(t)$$

Lo schema di riferimento è il seguente:

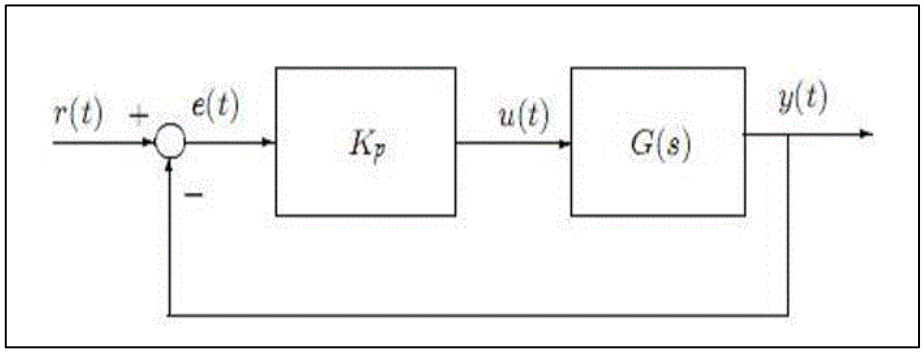


Figura 4.2: Schema a blocchi dell'azione proporzionale.

Il significato dell'azione proporzionale sta nel fatto che maggiore sarà l'errore e(t) all'ingresso del controllore e maggiore sarà l'azione di controllo svolta dallo stesso regolatore.
Guardando la formula generale del PID espressa in precedenza, possiamo capire quale significato assume il guadagno proporzionale K_P quando lavora in condizioni di regolatore puramente proporzionale.

$$u(t) = K_P \left(e(t) + \frac{1}{T_I} \cdot \int_0^t e(\tau) d\tau + T_D \cdot \frac{de(t)}{dt} \right)$$

In queste condizioni ($K_I = K_D = 0$ oppure Ti $\rightarrow \infty$ e $T_D = 0$) il regolatore provoca l'attuazione di una correzione proporzionale all'errore e(t) in funzione di K_P, aumenta la velocità di risposta del sistema e, con guadagno elevato, diminuisce la stabilità, aumentando le oscillazioni.
Il parametro caratteristico dell'azione proporzionale, cioè il guadagno proporzionale KP, ha una definizione prettamente teorica. Infatti, nella realtà industriale, al posto di quest'ultimo, si fa riferimento ad un'altra grandezza, chiamata Banda Proporzionale e indicata con il simbolo BP. Essa è definita come la minima variazione dell'ingresso e(t) (espressa in percentuale) che porta l'uscita u(t) dal valore minimo al suo valore di fondo scala.
In figura 4.3 sono evidenziati due esempi nei quali si ha un campo di valori uguale per ingresso ed uscita del PID. Possiamo vedere, infatti, come nel diagramma di sinistra la percentuale di variazione dell'ingresso e(t), che provoca il raggiungimento da parte dell'uscita u(t) del suo valore di fondo scala, sia del 50%, mentre nel diagramma di destra sia del 25%. Avviene la medesima variazione nei due casi ma con valori di banda proporzionale diversi.

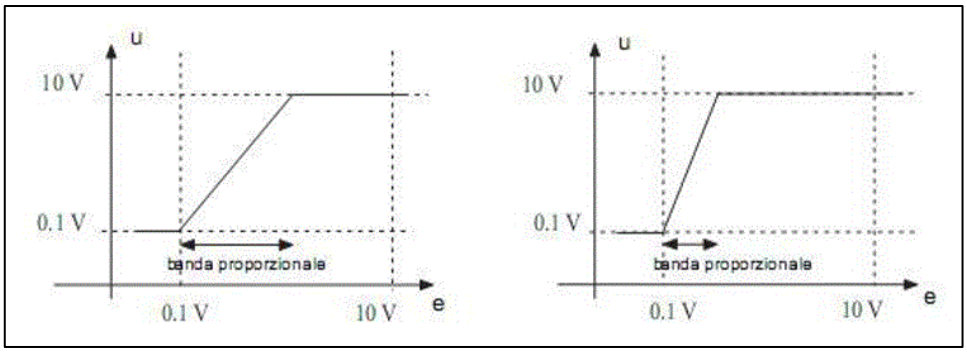


Figura 4.3: Esempi di ingressi e uscite del PID.

Quando K_P è espresso come rapporto tra i segnali normalizzati ai rispettivi valori di fondo scala, cioè definito nel seguente modo;

$$K_P = \frac{u(t)/u_{fondoscala}}{e(t)/e_{fondoscala}}$$

allora si può usare la seguente relazione tra B_P e K_P: $K_P = \frac{100}{B_P}$,

dalla quale si deduce che, riducendo la banda proporzionale, si ottiene una maggiore precisione dello strumento, in quanto è sufficiente una minore variazione dell'errore in ingresso perché questa porti ad un significativo cambiamento in uscita.

Dunque, riducendo la banda proporzionale, il guadagno aumenta e la precisione migliora (ecco che aumenta la velocità di risposta del sistema), ma non è tutto. Infatti, ad un aumento del guadagno, corrisponde anche un aumento del valore raggiunto all'uscita del blocco P (proporzionale). Tale valore si discosterà sempre di meno dal valore originale del segnale r(t), senza però raggiungerlo, a causa dell'errore a regime. Il contributo dovuto a K_P è appunto proporzionale all'errore e diminuisce man mano che l'errore si avvicina a zero.
All'aumento del guadagno c'è, però, un limite, dovuto al fatto che oltre un certo valore il sistema entra in uno stato di auto oscillazione, cioè di instabilità. Infatti, l'azione proporzionale viene utilizzata nei sistemi nei quali sono consentiti scostamenti tra il valore effettivo della grandezza controllata e quello desiderato, come, ad esempio, negli impianti di riscaldamento degli edifici.
Passiamo ora alla dimostrazione, con l'aiuto di formule matematiche, delle asserzioni fatte in precedenza riguardanti il guadagno e l'errore a regime.
A titolo d'esempio, associamo allo schema di figura 4.2 le due funzioni R(s) e G(s) con la seguente espressione:

$$R(s) = \frac{1}{s} \qquad G(s) = \frac{A_0}{1+s\tau}$$

e la funzione dell'azione proporzionale $C(s) = K_P$.

La funzione di trasferimento del sistema ad anello chiuso risulta essere:

$$W(s) = \frac{Y(s)}{R(s)} = \frac{\frac{K_P \cdot A_0}{1+\tau \cdot s}}{1+\frac{K_P \cdot A_0}{1+\tau \cdot s}} = \frac{\frac{K_P \cdot A_0}{\tau}}{s+\frac{1+K_P \cdot A_0}{\tau}}$$

Applicando ad essa l'ingresso a gradino unitario, si ottiene il seguente segnale d'uscita:

$$Y(s) = R(s) \cdot W(s) = \frac{1}{s} \cdot \frac{\frac{K_P \cdot A_0}{\tau}}{s+\frac{1+K_P \cdot A_0}{\tau}}$$

La funzione d'errore vale:

$$E(s) = R(s) - Y(s) = R(s) - E(s)C(s)G(s) \rightarrow E(s) = \frac{1}{s} \cdot \frac{1+\tau s}{1+\tau s+K_P A_0}$$

Applicando poi il teorema dei residui e antitrasformando con Laplace, si arriva alla risposta nel tempo del sistema:

$$y(t) = \left(\frac{K_P \cdot A_0}{1 + K_P \cdot A_0}\right) \cdot \left[1 - e^{-\frac{1+K_P \cdot A_0}{\tau} \cdot t}\right] \cdot \delta_{-1}(t)$$

Da quest'ultima formula si capisce come, all'aumento del guadagno proporzionale K_P, corrisponde una crescita del guadagno in continua del sistema (avvicinandosi circa all'unità) e, nel contempo, la costante di tempo dell'esponenziale viene ridotta.
In altre parole, aumentando K_P, il valore asintotico dell'uscita sarà sempre più vicino al valore costante richiesto (in questo caso si avvicinerà al valore 1, visto che il nostro segnale d'ingresso corrisponde a un gradino di ampiezza unitaria) e si osserva una contemporanea riduzione del tempo necessario per arrivare a tale livello asintotico. Notiamo, tuttavia, che l'uscita non arriverà mai esattamente al valore richiesto, a meno di non far tendere K_P all'infinito. Il valore asintotico dell'errore di regime permanente viene anche detto offset.
L'azione proporzionale del controllore ha l'effetto di ridurre il tempo di salita tr (parametro che caratterizza la prontezza del sistema), incrementare le sovraelongazioni e ridurre, ma non eliminare, l'errore a regime permanente. Quest'ultimo viene chiamato in causa quando si parla di tipo di un sistema, infatti un sistema si dice di tipo k se insegue con errore di regime permanente costante ma non nullo il segnale canonico $\delta_{-(k+1)}(t) = \frac{t^k}{k!}\,\delta_{-1}(t)$.
Se ipotizziamo che tale sistema sia SISO, lineare, tempo-invariante, causale e BIBO stabile, a tempo continuo, e la sua funzione di trasferimento W(s) sia strettamente propria e soddisfi $W(0) \neq 0$, allora, se indichiamo con $w_{-(k+1)}(t)$ la risposta (forzata) del sistema al segnale di ingresso $u(t) = \delta_{-(k+1)}(t)$, il sistema è di tipo k se:

$$e_{rp}^{(k+1)} := \lim_{t \to +\infty} [\, \delta_{-(k+1)}(t) - w_{-(k+1)}(t)]$$

esiste finito ed è un numero (reale) non nullo.

Nel caso in esame il nostro sistema presenta una $W(0) = \frac{K_P A_0}{1+K_P A_0}$ quindi il sistema è di tipo 0 perché la $W(0) \neq 1$.
Si noti che, poiché la relazione ingresso/uscita tiene conto anche della costante KP del controllore proporzionale, l'errore a regime, ottenuto mediante il teorema del valor finale, è espresso da:

$$\lim_{t \to +\infty} e(t) = \lim_{s \to 0} s \cdot E(s) = \frac{1}{1 + K_P \cdot A_0}$$

Allo stesso risultato si perviene se consideriamo che il sistema è di tipo 0 e che l'errore a regime permanente è pari a $e_{rp}^{(1)} = 1\text{-}W(0)$.
Da ciò si evince che per eliminare definitivamente l'errore a regime bisogna che il termine K_p tenda all'infinito, operazione idealmente realizzabile ma nella realtà di difficile implementazione.

Consideriamo ora un secondo esempio nel quale il processo sia caratterizzato da una funzione di trasferimento del secondo ordine, priva di zeri.

$$G(s) = \frac{A_0}{1 + 2\xi\left(\frac{s}{\omega_n}\right) + \left(\frac{s}{\omega_n}\right)^2}$$

dove ξ e ω_n sono due costanti positive.

Con un procedimento analogo a quello sviluppato per il polo del primo ordine si ricava:

$$W(s) = \frac{K_P \cdot A_0}{1 + K_P \cdot A_0 + 2\xi\left(\frac{s}{\omega_n}\right) + \left(\frac{s}{\omega_n}\right)^2}$$

L'errore di regime permanente al gradino E(s) lo esprimiamo nella stessa forma in cui abbiamo lavorato in precedenza, ottenendo:

$$E(s) = \frac{1}{s} \cdot \frac{1 + 2\xi\left(\frac{s}{\omega_n}\right) + \left(\frac{s}{\omega_n}\right)^2}{1 + K_P \cdot A_0 + 2\xi\left(\frac{s}{\omega_n}\right) + \left(\frac{s}{\omega_n}\right)^2}$$

dove si è tenuto conto del segnale R(s) a gradino di ampiezza unitaria e della funzione di trasferimento G(s) del secondo ordine.

Applicando il teorema del valor finale all'errore ottengo la seguente espressione:

$$\lim_{t \to +\infty} e(t) = \lim_{s \to 0} s \cdot E(s) = \frac{1}{1 + K_P \cdot A_0}$$

Analogamente a quanto visto per un processo con un polo del primo ordine, anche in questo caso si osserva la presenza di un offset che diminuisce all'aumentare di K_p.
Tuttavia, se andiamo ad analizzare l'espressione della funzione di trasferimento del sistema ad anello chiuso W(s), notiamo che, all'aumentare di K_P, il transitorio che segue rapide variazioni di r(t) è caratterizzato da oscillazioni di frequenza sempre più elevata e meno smorzate. Questo è evidenziato a titolo d'esempio in figura 4.4.

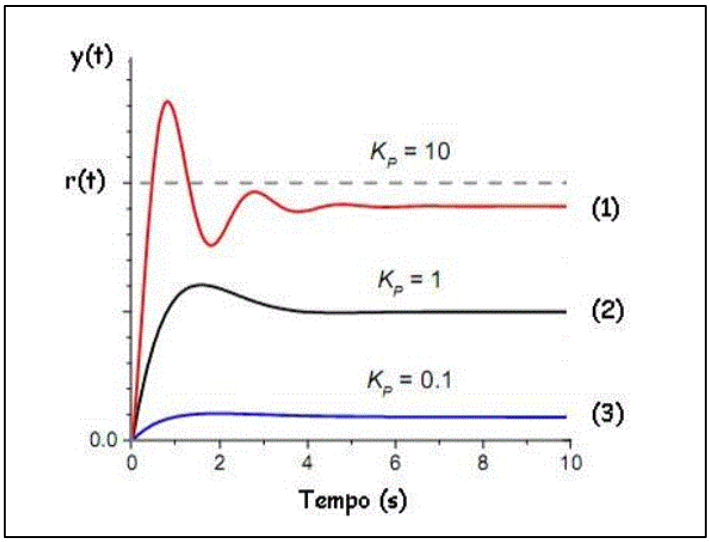


Figura 4.4: Andamento per vari valori di K_P.

In figura 4.4 sono riportati dei grafici che raffigurano gli effetti di una banda proporzionale eccessivamente piccola (1), adeguata (2) ed eccessivamente grande (3). Si noti che per $K_P = 10$ si riduce l'effetto dell'offset, ma la risposta al transitorio è caratterizzata da forti oscillazioni.
In conclusione, possiamo dire che utilizzando un controllore esclusivamente proporzionale per un sistema privo di poli nell'origine si produce una differenza (offset) tra il valore costante richiesto a regime e quello effettivamente ottenuto. Tale differenza può essere ridotta aumentando il guadagno proporzionale KP del controllore. Tuttavia, se il processo da controllare possiede coppie di poli complessi coniugati, l'aumento del guadagno proporzionale è accompagnato da un corrispondente aumento delle oscillazioni generate a seguito di rapidi transitori.
Dallo studio e dall'analisi operati in precedenza, traiamo dunque importanti conclusioni che riguardano l'utilità del blocco ad azione proporzionale P.
Vengono elencate di seguito:

- per avere un errore a regime molto piccolo bisogna che A_0K_P sia elevato (ed essendo fissato A_0 tale discorso vale per K_P), ma questo aumento può compromettere la stabilità del sistema rendendolo molto suscettibile ai rumori di misura (cioè agli errori che avvengono sull'uscita y(t) e che si ripercuotono sul canale di retroazione influendo sul segnale che genera e(t)).
- l'impiego di un regolatore proporzionale è normalmente limitato al controllo di processi asintoticamente stabili, o semplicemente stabili, e nel caso in cui le prestazioni statiche richieste al sistema di controllo non rendano necessario l'inserimento di un'azione integrale;
- siccome l'azione proporzionale non garantisce l'annullamento dell'errore a regime nel caso di segnali di riferimento costanti, si può pensare di annullare l'errore sommando alla variabile di controllo u(t) un valore appropriato U, facilmente calcolabile, eventualmente aggiustato dall'operatore: $u(t) = K_P\, e(t) + U$. Il valore U, nella letteratura inglese, è chiamato reset;
- il blocco proporzionale P non introduce ritardo di fase ed aumenta la velocità di risposta del sistema rendendolo così capace di inseguire più velocemente il segnale di riferimento (setpoint). Nel contempo però diminuiscono i margini di stabilità del

sistema, margini che possono essere facilmente determinati tramite il criterio di Routh, il quale permette di valutare il massimo valore di K_p che garantisce la stabilità del sistema ad anello chiuso. In alternativa al criterio di Routh si può utilizzare il diagramma di Nyquist.

Da quanto detto finora, si può avere un quadro riassuntivo delle proprietà generali che il blocco proporzionale possiede, guardando la tabella sottostante.

Simbolo	Funzione di regolazione	Scopo principale	Stabilità	Tempo di risposta
K_p	Uno scostamento sull'ingresso (Errore) produce una variazione dell'uscita proporzionale all'ampiezza dello scostamento	Fa variare la grandezza regolante in funzione della grandezza regolata		
Aumenta K_p	-	-	Migliora	Rallenta
Diminuisce K_p	-	-	Peggiora	Accelera

Figura 4.5: Tabella riassuntiva del termine proporzionale.

4.4) Azione Integrale

Veniamo ora ad analizzare la seconda azione introdotta dal PID, ovvero l'azione integrale. Il contributo di questa azione è proporzionale all'integrale dell'errore e(t) (e quindi proporzionale al suo valor medio) e il coefficiente dell'azione integrale K_I definisce la costante di tempo integrale T_I (chiamata anche tempo di reset):

$$u(t) = K_I \cdot \int_0^t e(\tau)d\tau \qquad\qquad T_I = \frac{K_P}{K_I}$$

Ciò deriva dalla formula generale, assumendo K_D e K_P nell'espressione

$$u(t) = K_p e(t) + K_I \int_{t_0}^{t} e(\tau)d\tau + K_D \frac{de(t)}{dt}$$

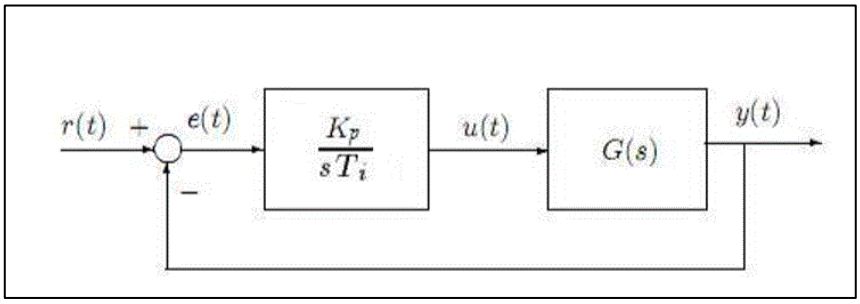


Figura 4.6: Schema a blocchi dell'azione integrale.

L'azione integrale è particolarmente importante nelle applicazioni, perché assicura un errore nullo a regime per variazioni a gradino del riferimento r(t). L'errore rimane nullo anche in presenza di variazioni del guadagno del processo, purché sia preservata la stabilità del sistema in anello chiuso.
In generale l'azione integrale è associata all'azione proporzionale in modo tale da realizzare i cosiddetti controllori PI:

$$u(t) = K_P\left(e(t) + \frac{1}{T_I}\cdot\int_0^t e(\tau)d\tau\right) \qquad U(s) = K_P\left(1 + \frac{1}{T_I\cdot s}\right)\cdot E(s)$$

i quali permettono di ottenere una maggiore precisione (incremento del tipo del sistema retroazionato) senza peggiorare il grado di stabilità del sistema (sempre che le costanti siano opportunamente scelte), unita ad una maggiore velocità di risposta.
I controllori PI vengono utilizzati quando è richiesto un errore a regime di modesta entità unito ad una buona velocità di risposta alle variazioni della sollecitazione; pertanto vengono inseriti soprattutto nei sistemi in cui le variazioni di carico avvengono lentamente.
La taratura della costante di integrazione è legata alle sovraelongazioni ed alle oscillazioni che si possono innescare. Infatti, cambiamenti improvvisi di carico possono portare il sistema verso l'instabilità, quando il coefficiente K_I dell'azione integrale non è scelto in modo opportuno. In ambito industriale oltre il 90% dei controllori é di questo tipo.
Nel caso di un segnale di riferimento r(t) a gradino, per esempio, l'integrale crescerà sempre più, come evidenziato in figura 4.7.

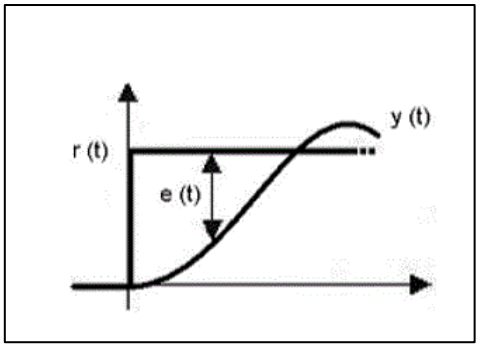


Figura 4.7: Esempio dell'andamento del termine integrale.

Quando aggiungiamo l'azione integrale al blocco proporzionale eliminiamo a priori la presenza dell'offset.
È evidente che in presenza di un offset costante il contributo integrale è destinato a crescere indefinitamente nel tempo e questo ci permette di attivare una efficace azione correttiva.
Inoltre, con l'azione integrale, un piccolo errore positivo e(t) condurrà sempre ad un incremento del segnale di controllo u(t), e un errore negativo darà un decremento del segnale di controllo a prescindere dal fatto di quanto sia piccolo l'errore.
Per il blocco in considerazione, la variabile da tarare in base al sistema con cui si ha a che fare è la costante di tempo integrale T_I; da essa infatti dipende l'effetto di integrazione che è tanto più importante quanto più T_I è piccola. A questo riguardo è necessario notare come riducendo il tempo di reset, l'integrale dell'errore salirà più velocemente verso il valore r(t) in ingresso al blocco pagando però questa velocità di salita con delle forti oscillazioni che richiedono tempo

prima di stabilizzarsi. Viceversa, aumentando T_I e facendolo tendere ad infinito, si ha come conseguenza l'eliminazione dell'effetto dell'integrale.
Analizziamo ora un esempio, tramite formule matematiche, della combinazione dei due termini, proporzionale e integrale. Supponiamo di inviare in entrata un gradino di ampiezza unitaria R(t)=1/s e assumiamo come funzioni di trasferimento che caratterizzano il sistema di figura 4.6,

$$C(s) = K_P\left(1 + \frac{1}{T_I s}\right) \qquad G(s) = \frac{A_0}{1+\tau s}$$

Da queste espressioni ricaviamo la funzione di trasferimento ad anello chiuso:

$$W(s) = \frac{1}{1+\frac{sT_I(1+\tau s)}{A_0 K_P(1+T_I s)}}$$

Insieme all'errore $E(s) = \frac{1}{s} \cdot \frac{sT_I(1+\tau s)}{sT_I(1+\tau s+A_0K_P)+A_0K_P}$

Come fatto in precedenza nell'analisi dell'azione proporzionale, anche qui determiniamo, tramite il teorema del valor finale, l'espressione dell'errore asintotico:

$$\lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s) = 0$$

Notiamo che facendo tendere T_I all'infinito, ovvero eliminando il termine integrale, le espressioni soprastanti si riducono a quelle trovate precedentemente per l'analogo esempio sviluppato considerando un controllore esclusivamente proporzionale.
L'effetto della scelta di T_I è mostrato in figura 4.8. Si nota che l'offset presente quando si opera solo con il controllore proporzionale ($T_I \to \infty$) sparisce attivando il termine integrale. Riducendo T_I il sistema risponde più velocemente al transitorio, ma si osservano anche delle forti oscillazioni.

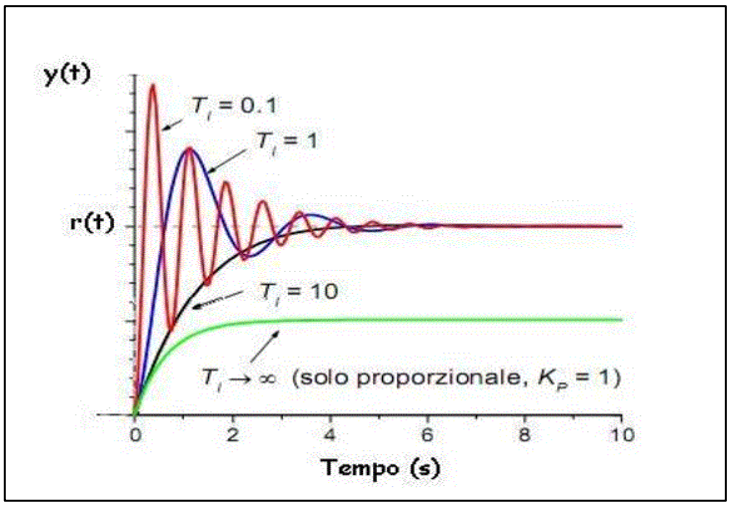


Figura 4.8: Andamento per vari valori di T_I.

Il termine integrale però, a differenza di quello proporzionale, introduce uno sfasamento di 90° in ritardo (dunque aggiunge un polo in termini di trasformata di Laplace) che porta ad un peggioramento dei margini di stabilità del sistema ad anello chiuso, limitando il valore del guadagno integrale (e quindi la banda del sistema).
Questa tabella ci fornisce una visione generale del blocco integrale:

Simbolo	Funzione di regolazione	Scopo principale	Stabilità	Tempo di risposta del loop
T_I	Appena si ha uno scostamento sull'ingresso (Errore), si produce una variazione dell'uscita con velocità proporzionale allo scostamento	Fissa il punto di regolazione (Elimina l'offset dato dall'azione proporzionale)		
Aumenta T_I	-	-	Migliora	Rallenta
Diminuisce T_I	-	-	Peggiora	Accelera

Figura 4.9: Tabella riassuntiva del termine integrale.

In particolare, gli attuatori posti nei sistemi industriali (e cioè quei meccanismi o dispositivi che agiscono su un ambiente a partire da una sollecitazione in ingresso), hanno un intervallo di lavoro che di solito ha dei limiti molto superiori al normale range di valori assunti dalla variabile che li controlla. In qualche caso, tuttavia, a causa di rapidi transitori provocati da disturbi sul carico, tali limiti possono essere anche raggiunti e superati.
Quando questo accade il circuito di retroazione è rotto e il sistema funziona come un anello aperto perché l'attuatore resterà al suo limite indipendentemente dal processo di uscita. L'attuatore entra in saturazione e il valore che si ottiene in uscita non rispecchia più quello in ingresso.

Ciò è evidenziato nella figura 4.10:

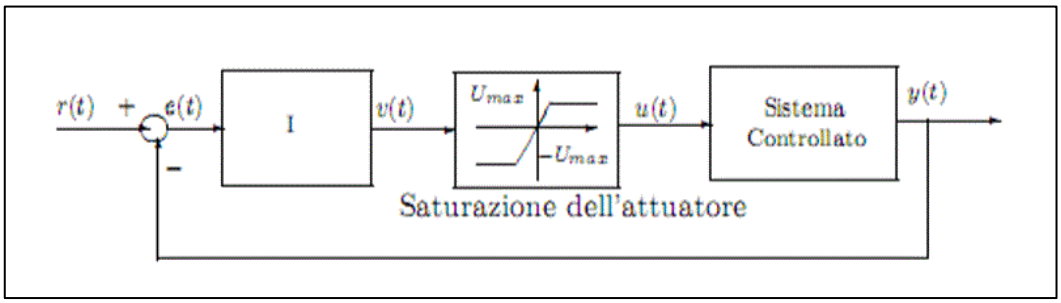


Figura 4.10: Schema a blocchi dell'azione integrale con saturazione.

Matematicamente la saturazione dell'attuatore è definita dalla seguente formula:

$$u(t) = \begin{cases} U_{max} & \rightarrow if\ v(t) \geq U_{max} \\ v(t) & \rightarrow if\ U_{min} < v(t) < U_{max} \\ U_{min} & \rightarrow if\ v(t) \leq U_{min} \end{cases}$$

dove U_{max} e U_{min} sono i limiti massimo e minimo rispettivamente dell'uscita. Quando si presenta questa situazione si ha in uscita dall'attuatore un valore inferiore, in modulo, al valore che lo stesso attuatore presenta in ingresso.
Dopo essere passato per il sistema controllato, il segnale correttivo che arriva con la retroazione all'ingresso è minore del previsto, dunque incide meno sulla correzione di r(t) e perciò fa aumentare il tempo nel quale l'errore viene inseguito e raggiunto.
Questo processo continuerà fino a quando l'errore non diventerà negativo (cioè fino a quando il segnale y(t) non supererà r(t)), ma in quell'istante il valore dell'integrale avrà raggiunto un livello tale che anche se inizierà a diminuire, passerà dell'altro tempo prima che questo valore arrivi a zero, causando una considerevole sovraelongazione dell'uscita y(t).
Con l'aiuto dei prossimi due grafici riusciremo a capire la differenza tra quanto avviene in presenza di un attuatore ideale (rappresentato da una linea tratteggiata) e in presenza di un attuatore con limiti di saturazione (rappresentato da una linea continua) quando viene utilizzato un gradino come segnale r(t).

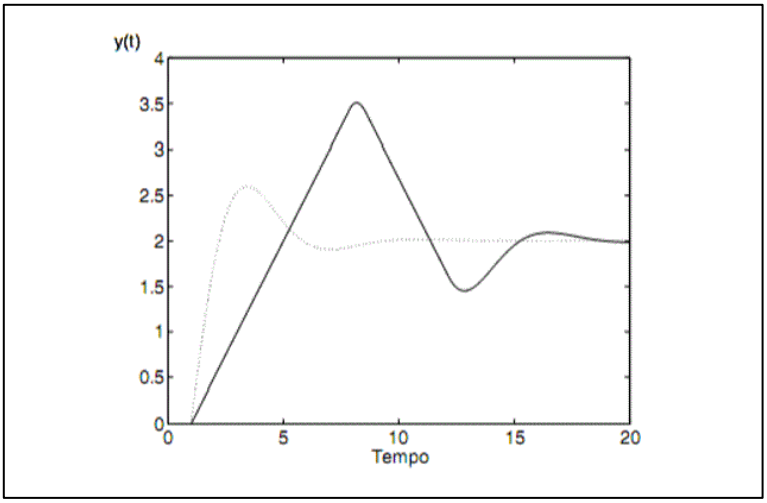


Figura 4.11: Rappresentazione dell'uscita del sistema.

Dal grafico si nota che, con l'attuatore ideale, l'uscita del sistema ha una forma d'onda che segue l'andamento dell'integrale, come visto in precedenza (vedi linea tratteggiata), mentre, se l'attuatore ha dei limiti di saturazione, si ottiene l'indesiderata sovraelongazione. Lo vediamo dall'andamento della linea continua di figura.
Nella figura 4.12, invece, viene rappresentato l'errore di inseguimento, che, all'inizio, cresce istantaneamente (nel momento di applicazione del gradino in ingresso), ma, successivamente, assume due diversi andamenti.

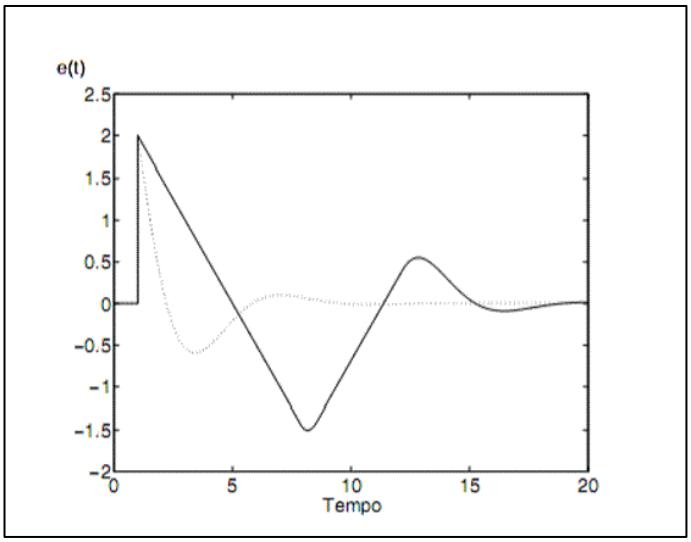


Figura 4.12: Rappresentazione dell'errore d'inseguimento.

Anche qui, la linea tratteggiata identifica il comportamento in presenza di attuatore ideale, con il quale l'errore di inseguimento viene presto eliminato, e si torna ad una situazione di normalità. Con la linea continua, invece, si mette in evidenza la presenza di un attuatore reale, con il quale si aumenta il ritardo di correzione (causato dal minore potere correttivo della retroazione) e, inoltre, si va a finire su valori negativi per la lunga durata della sovraelongazione. Dunque, il processo evolverà ricevendo un ingresso u(t) costante, come se fossimo in anello aperto e, mancando la retroazione, anche il controllore sarà costretto ad evolvere in quel modo. Questo porterà il blocco integratore ad allontanarsi anche di molto dai valori di controllo che gli competono, essendo esso un sistema dinamico non BIBO stabile, a causa del polo che possiede nell'origine. Inoltre, durante questo periodo di saturazione dell'attuatore, il sistema non avrà nessun tipo di reazione alla presenza di eventuali disturbi.
Possiamo avere un'idea di quanto appena detto osservando i grafici sottostanti.

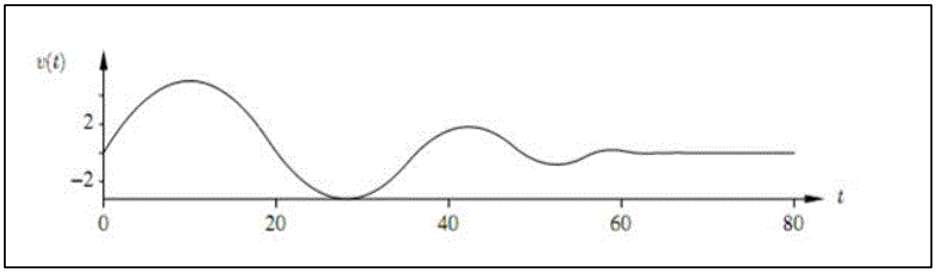


Figura 4.13: Andamento del termine integrale.

Dalla figura 4.13 possiamo evincere come il termine integrale cresca inizialmente perché l'errore è positivo e raggiunge il suo valore massimo al tempo t = 10, quando l'errore passa attraverso lo zero.

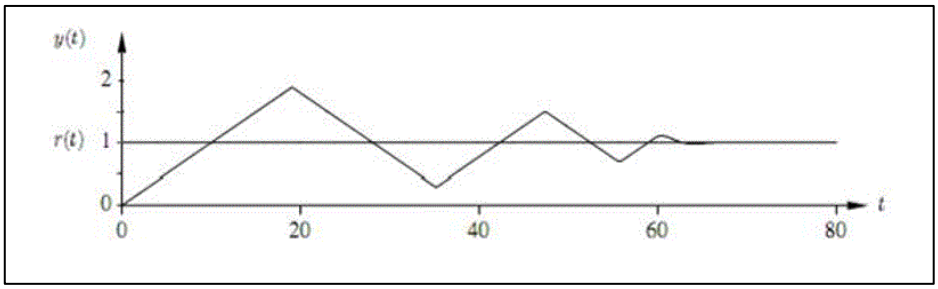


Figura 4.14: Andamento dell'uscita.

In figura 4.14 notiamo, invece, come l'uscita rimane satura in quel punto, a causa del grande valore del termine integrale. La stessa uscita non lascia il limite di saturazione, fino a quando l'errore è diventato negativo per un tempo sufficientemente lungo, tale da consentire alla parte integrante di scendere a un livello basso.

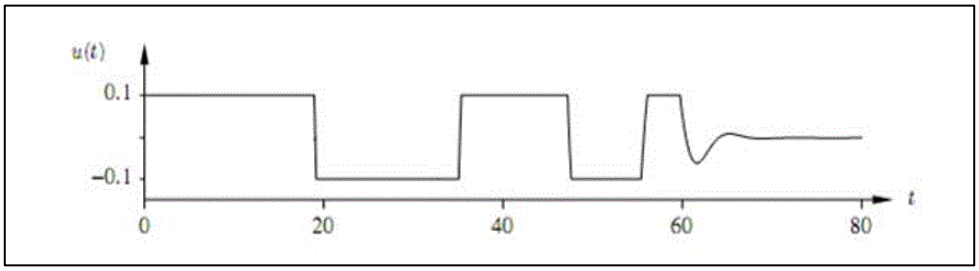


Figura 4.15: Andamento del segnale di controllo.

Si noti come in figura 4.15 il segnale di controllo rimbalzi tra i suoi limiti più volte. L'effetto risultante è una grande sovraelongazione ed una oscillazione smorzata sull'uscita y(t), in cui il segnale di controllo u(t) varia da un estremo all'altro, come se si trattasse dell'oscillazione conseguente all'apertura e chiusura di un relè. L'uscita finalmente arriva così vicino al valore di riferimento, che l'attuatore non satura.
Se si usa un controllore con azione integrale, l'errore continuerà a essere integrato. Questo significa che il termine integrale può diventare molto grande. Da qui il fenomeno conosciuto come windup. È, quindi, necessario che l'errore sia di segno opposto per un lungo periodo, prima che le cose tornino alla normalità. La conseguenza è che qualsiasi controllore con azione integrale può dare transitori di elevata durata, quando l'attuatore satura.
Supponendo di applicare al sistema di riferimento un disturbo a gradino r(t), l'integratore inizierà ad accrescere il suo valore in uscita per via dell'errore non nullo. Quando il valore in uscita al controllore è tale da saturare il comando di attuazione, si ha che l'uscita dell'integratore continuerà a crescere, fino a quando l'errore non diventerà nullo, mentre il comando di attuazione rimarrà costante.
Se il sistema è BIBO stabile, dopo un dato intervallo di tempo l'errore diventerà nullo e l'integratore inizierà a scaricarsi. Fino a quando il valore di uscita non sarà inferiore alla saturazione (relativa all'attuatore), il segnale di controllo rimarrà costante. Questo fenomeno fa si che il sistema, dopo aver raggiunto la condizione di errore nullo, si allontani in direzione opposta, creando un effetto di sovraelongazione dalle caratteristiche non lineari.

Una situazione del genere è indesiderata e richiede un certo periodo di tempo prima che la situazione normale si ristabilisca, ma, fortunatamente, ci sono vari modi per poter ovviare a questo problema.
Uno di questi metodi può essere l'immissione di limitatori sulle variazioni del valore di riferimento, in modo che l'uscita del regolatore non raggiunga mai i limiti dell'attuatore.
Sempre parlando di metodi correttivi al fenomeno del windup, possiamo anche avere il cosiddetto metodo degli algoritmi incrementali, nel quale si calcola, prima, il tasso di variazione del segnale di controllo. che viene, poi, inviato ad un integratore. In alcuni casi tale integratore è un motore direttamente connesso all'attuatore. Così facendo, è facile evitare il fenomeno del windup mediante integrazione, inibendo l'uscita ogni volta che satura.
Altri metodi di compensazione del fenomeno di windup verranno descritti nel seguito capitolo.

4.4.1) Ricalcolo del termine integrale

Questa tecnica di correzione introduce un segnale di compensazione che viene riportato all'ingresso del blocco integrale e sottratto al valore dell'errore e(t).
Questo segnale viene generato prendendo il valore della grandezza di interesse misurata all'uscita dell'attuatore e confrontandolo con il valore che la stessa grandezza presenta all'ingresso. Si tratta di $e_S(t) = v(t) - u(t)$.
Così facendo, si otterrà un segnale di compensazione nullo (che quindi non influirà in alcun modo nel controllo) nel caso in cui l'attuatore stia lavorando entro i limiti di saturazione, mentre, quando questi limiti verranno superati, si genererà un valore che aiuterà a smorzare la salita del termine integrale.
Nella figura 4.16 è illustrato uno schema di principio per la costruzione della variabile di attuazione.

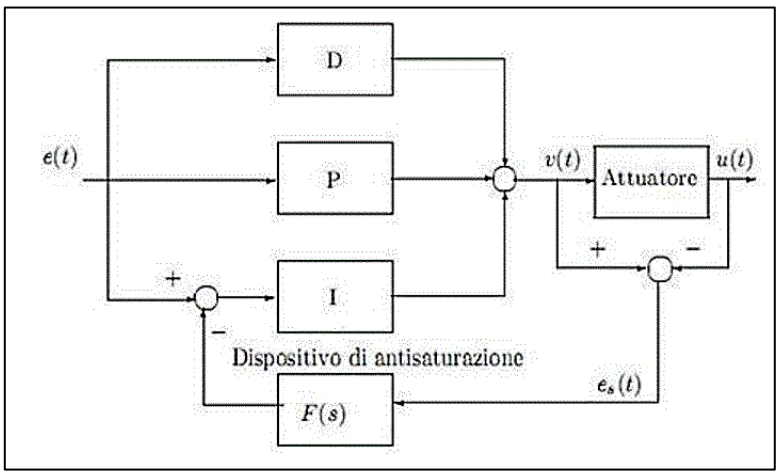


Figura 4.16: Schema per la costruzione delle variabili di attuazione.

Dallo schema si nota, inoltre, come sia possibile inserire, prima del ritorno del segnale di compensazione al blocco integratore, un altro blocco, chiamato dispositivo di antisaturazione. La sua funzione di trasferimento F(s) è solitamente pari a $1/T_t$, il cui valore controlla la velocità dell'azione di desaturazione.
Infatti, in presenza di valori ridotti della costante di tempo T_t, si avrà un'azione di desaturazione più rapida, ma molto più facilmente soggetta a disturbi. In caso contrario, se il valore assegnato

alla costante di tempo risulterà elevato, l'azione di desaturazione sarà meno influenzata dai disturbi, ma anche più lenta.
Alternativamente allo schema visto in figura 4.16, possiamo utilizzare un altro schema molto simile (vedi figura 4.17), nel quale si aggiunge un modello dell'attuatore nel circuito, ai cui capi verrà prelevato il termine correttivo $e_s(t)$.

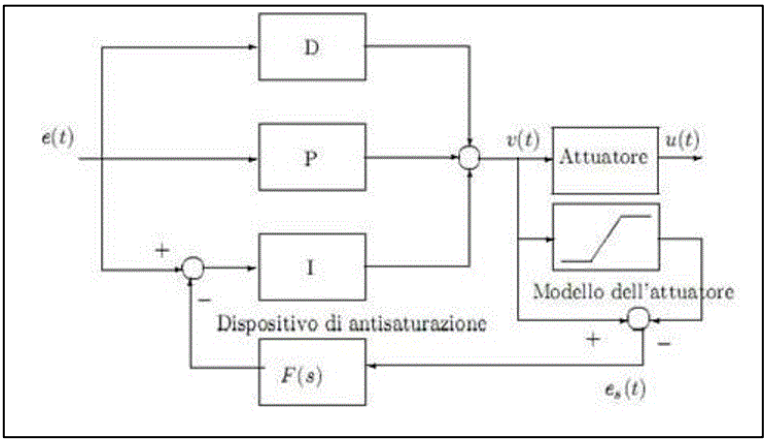


Figura 4.17: Schema precedente con il modello di attuazione nel circuito.

In questo caso, invece che di una misura della variabile di attuazione, siamo in presenza di una stima della variabile di attuazione. Questo sistema viene usato nel caso in cui il segnale u(t) in uscita dall'attuatore non sia misurabile. Perciò, si ricorre a un modello matematico che simuli le proprietà dell'attuatore in questione, ma che dia una risposta adatta ad essere trattata per ricavarne il segnale di compensazione.

4.4.2) Integrazione condizionata

Questa soluzione presenta uno schema di principio assai simile a quello visto in precedenza, ma si discosta da esso per la sostituzione del dispositivo di antisaturazione con un comparatore, la cui presenza serve per verificare quando il valore del termine di compensazione $e_s(t)$ assume valori non nulli.

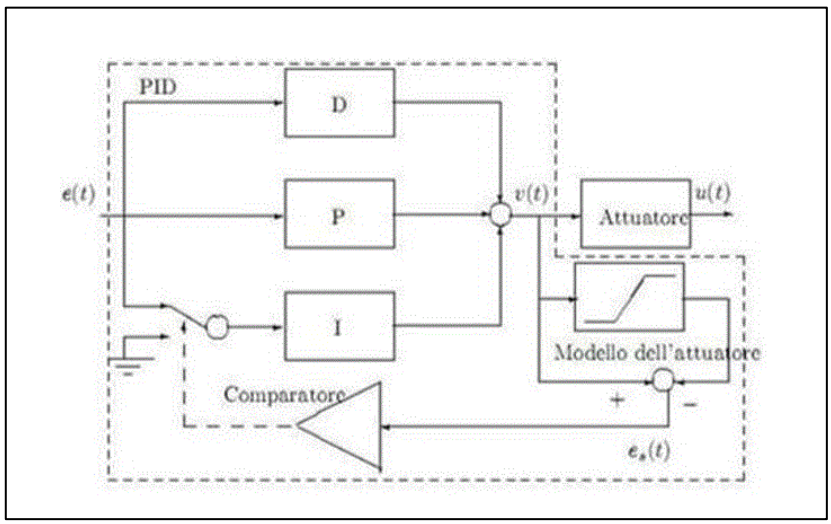


Figura 4.18: Schema di compensazione.

Lo schema di compensazione viene utilizzato solo in certe condizioni, altrimenti si lascia che l'azione integrale lavori normalmente, cioè senza provocare saturazione. Infatti, è proprio il fenomeno della saturazione dell'attuatore che provoca il fenomeno del windup.
Esistono alcune strategie di commutazione (switching conditions) tali da garantire la totale eliminazione del fenomeno windup. Un approccio semplice è quello di spegnere l'integrazione quando l'errore di controllo è di grandi dimensioni. Un altro approccio è quello di spegnere l'integrazione nel corso della saturazione. Entrambi i metodi hanno lo svantaggio che il controllore possa rimanere bloccato in presenza di un errore di controllo non nullo, se il termine integrale ha un valore grande al momento dello spegnimento dell'integrazione.
Lo svantaggio non si presenta nel momento in cui l'integrazione viene spenta, ma quando il controllore è saturo. Supponiamo, per esempio, che il controllore diventi saturo in condizioni di saturazione superiore. L'integrazione, quindi, è spenta se l'errore di controllo è positivo, ma non se è negativo.
In questo caso, l'ingresso del blocco integrale, invece di mantenere in ingresso l'errore e(t) che lo farebbe aumentare ulteriormente, viene posto a zero, garantendo una veloce scarica, grazie alla presenza del comparatore che confronta i segnali $e_s(t)$ ed e(t).

IMC (Internal Model Control)

Altra tecnica per compensare il fenomeno del windup, che verrà illustrata per sistemi a tempo continuo, ma che può essere applicata anche a sistemi a tempo discreto, è la cosiddetta tecnica IMC, il cui acronimo significa:

- Internal: modello del sistema interno al controllore;
- Model: modello del processo da controllare;
- Control: modello per migliorare le prestazioni.

Questa tecnica si affida all'IMP (Internal Model Principle), secondo il quale il controllo può essere raggiunto solo se il sistema di controllo incapsula, implicitamente o esplicitamente. una qualche rappresentazione del processo da controllare. In particolare, se lo schema di controllo è stato sviluppato sulla base di un modello esatto del processo, allora si avrà un controllo perfetto teoricamente possibile. Nello schema illustrato in figura 4.19 notiamo come questa particolare tecnica opera per risolvere il problema del windup.

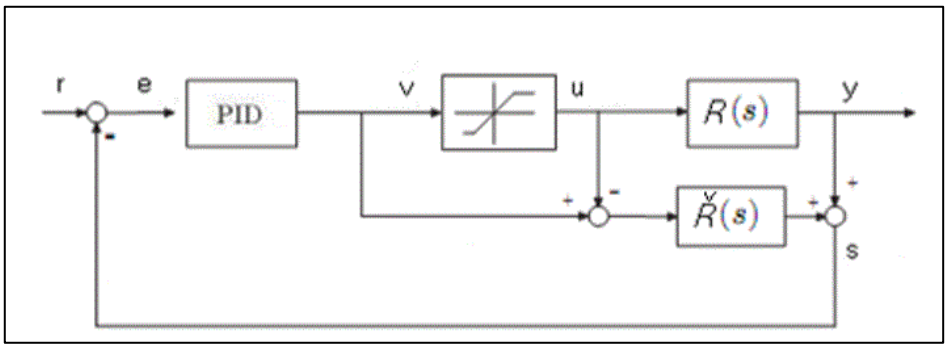


Figura 4.19: Schema del controllo per risolvere il windup.

In tale Figura si è indicata con R(s) la funzione di trasferimento del sistema controllato e con Ř(s) la funzione di trasferimento copia del sistema.

Con alcuni accorgimenti matematici, possiamo evidenziare il principio di funzionamento di questa specifica tecnica di compensazione.
Calcoliamo l'errore:

$$e = r - s \quad \rightarrow \quad e = r - \left[y + \check{R} \cdot (v - u)\right] \rightarrow e = r - Ru - \check{R}v + \check{R}u.$$

Assumendo che R = Ř, allora la formula dell'errore si riduce a $e = r - Rv.$

Da notare come la retroazione non sia necessaria se il processo è conosciuto nei minimi dettagli (cioè come incapsulato nel modello di processo), mentre la retroazione è valida quando il processo è inaccurato o incompleto.

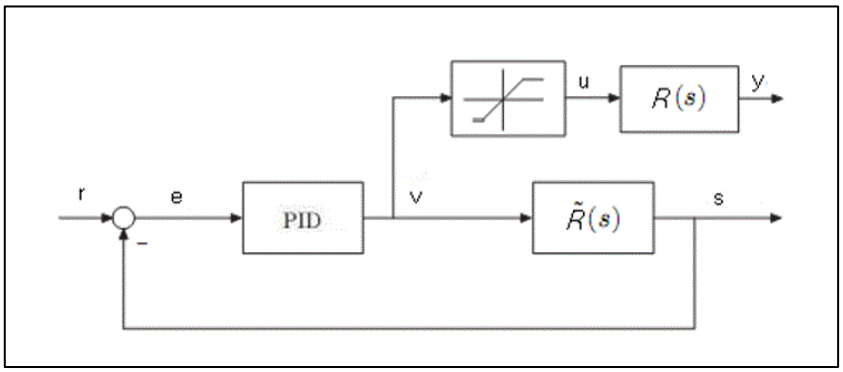


Figura 4.20: Schema alternativo del controllo per risolvere il windup.

La soluzione fornita dalla tecnica dell'IMC ha le seguenti proprietà:

- fornisce una compensazione al tempo di ritardo della risposta;
- il filtro passa - basso può essere utilizzato per modellare sia il monitoraggio del valore di riferimento, sia le risposte di reiezione al disturbo;
- il regolatore darà risposte prive di offset.

4.5) Azione derivativa

Come ultima caratteristica di un regolatore, ci rimane da analizzare l'azione derivativa che fornisce in uscita la derivata rispetto al tempo dell'errore e(t). In questo caso, abbiamo la presenza del coefficiente dell'azione derivativa K_D che definisce la costante di tempo derivativa T_D.

$$u(t) = K_D \cdot \frac{de(t)}{dt} \qquad\qquad T_D = \frac{K_D}{K_P}$$

Anche qui, queste considerazioni derivano dall'osservazione della formula generale dei PID in cui è stato assunto $K_P = 0$ e $K_I = 0$.

$$u(t) = K_P e(t) + K_I \int_{t_0}^{t} e(\tau) d\tau + K_D \frac{de(t)}{dt}$$

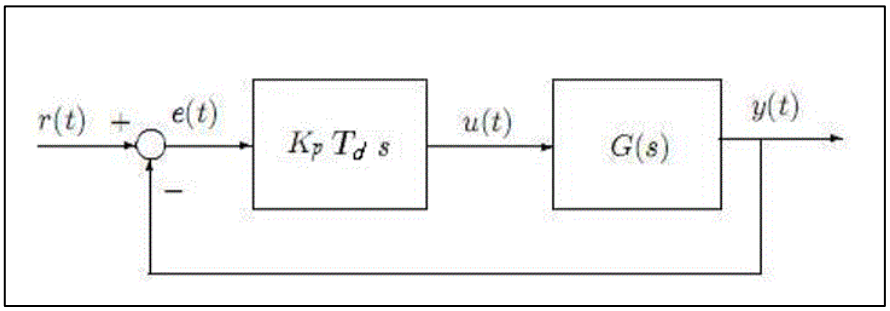


Figura 4.21: Schema a blocchi dell'azione derivativa.

Un controllore derivativo viene chiamato anche controllore di velocità, oppure anticipatore. Il suo comportamento è marcatamente diverso da quello dei controllori chiamati proporzionale e integrale. L'uscita di un controllore derivativo non dipende dall'errore presente o passato, ma dalla velocità con cui varia l'errore. Inoltre, l'azione derivativa è complementare all'azione integrale, perché fornisce un anticipo di fase di 90° (in quanto porta all'introduzione di uno zero nell'origine).
Per comprendere qualitativamente l'importanza dell'anticipo di fase, consideriamo un sistema in cui vi sia un forte ritardo di fase nell'azione in catena aperta.
In tal caso, una variazione della variabile di errore determina, attraverso il meccanismo di controllo in retroazione, una variazione della variabile controllata con un certo ritardo. Quindi, intuitivamente, è come se il controllore agisse "in ritardo" rispetto alle variazioni dell'errore.
Il parametro che governa questo blocco è la costante di tempo dell'azione derivativa T_D, il cui valore determina la velocità di salita del segnale di controllo. In questo caso, rispetto ai due blocchi visti precedentemente, la stabilità peggiora sia aumentando sia diminuendo il valore di T_D.
Per capire meglio l'effetto indotto dal contributo derivativo, oltre che qualitativamente anche per mezzo di formule matematiche, studiamo il comportamento di un controllore PD, la cui funzione di trasferimento risulta essere:

$$C(s) = \frac{U(s)}{E(s)} = K_P \cdot (1 + sT_D)$$

Consideriamo, come già visto in precedenza nella trattazione degli altri due blocchi, il caso in cui si applichi in ingresso una funzione gradino di ampiezza unitaria e supponiamo che la G(s) sia pari a:

$$G(s) = A_0 \cdot \frac{\omega_0^2}{s^2 + 2\xi\omega_0 s + \omega_0^2}$$

per cui l'uscita del sistema diventi:

$$Y(s) = \frac{A_0}{s} \cdot \frac{\omega_0^2 K_P (1 + sT_D)}{s^2 + s(2\xi\omega_0 + A_0\omega_0^2 K_P T_D) + \omega_0^2 (1 + A_0 K_P)}$$

Si noti che la presenza del termine derivativo introduce uno zero ed aumenta il coefficiente di s e di 1 nel polinomio di secondo grado posto al denominatore. Ambedue questi effetti producono una riduzione delle oscillazioni che si verificano in occasione dei transitori, contribuendo a stabilizzare il sistema.
In figura 4.22 è presentato il grafico dell'esempio appena analizzato, dal quale si può apprezzare l'azione anticipatrice del blocco derivatore e la sua correzione sulle oscillazioni attorno al valore desiderato.
L'offset è lo stesso che si ha con il solo controllo proporzionale e lo si può eliminare utilizzando un opportuno contributo integrale. Si conferma, dunque, la non incidenza di questo controllo sull'offset esistente tra y(t) e r(t).

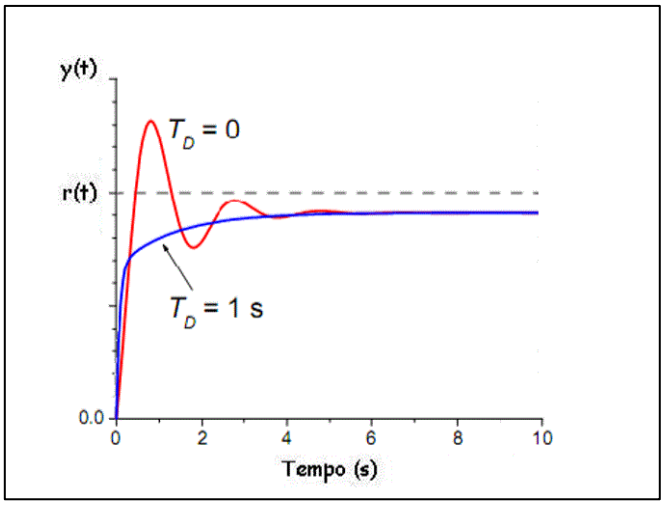


Figura 4.22: Andamento per vari valori di T_D.

Nella tabella sottostante possiamo dare una breve visione generale delle azioni che un blocco derivatore introduce.

Simbolo	Funzione di regolazione	Scopo principale	Stabilità	Tempo di risposta del loop
T_D	Uno scostamento sull'ingresso (Errore), produce una variazione dell'uscita proporzionale alla velocità di variazione dello scostamento	Dimunuisce il tempo di risposta per il ritorno al punto di regolazione		
Aumenta T_D	-	-	Peggiora	Accelera
Diminuisce T_D	-	-	Peggiora	Rallenta

Figura 4.23: Tabella riassuntiva del termine proporzionale.

Il termine derivativo, purtroppo, introduce l'inconveniente di amplificare i segnali con contenuto armonico a frequenze elevate. Siccome, in genere, questo tipo di segnali corrisponde al rumore elettromagnetico sovrapposto al segnale utile, l'utilizzo del termine derivativo deve essere valutato con cautela. Nella realizzazione pratica dei regolatori, infatti, si adotta l'accorgimento di limitare l'azione derivativa in modo tale che, in corrispondenza a transizioni a gradino, l'uscita del blocco derivatore non abbia andamento impulsivo.
Pertanto, la presenza di un rumore sui segnali di misura ne condiziona l'uso e richiede di associarla a un filtro passa - basso che ne limiti il guadagno ad alta frequenza.
Ad esempio, un rumore di misura sinusoidale, n(t) = A sin(ωt), in ingresso al blocco derivatore, dà luogo a un contributo sulla variabile di controllo pari a:

$$u_n(t) = T_D \frac{dn(t)}{dt} = AT_D\omega \cos \omega t,$$

di ampiezza crescente con ω. Quindi, i segnali di rumore aventi un contributo sensibile alle alte frequenze, e con una ω molto alta, vengono amplificati notevolmente dal termine derivativo del PID.
Passiamo ad un esempio numerico per chiarire meglio il concetto, usando formule e grafici. Supponiamo di avere come segnale utile e(t) = sin(ωt):

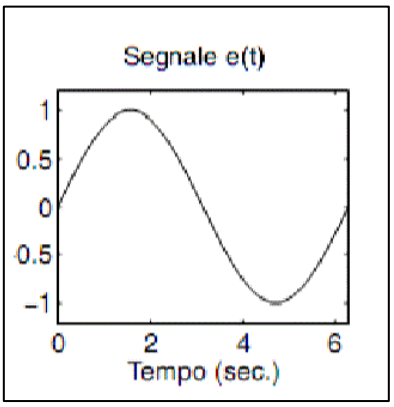


Figura 4.24: Andamento del segnale e(t) = sin(ωt).

mentre, come segnale di rumore n(t) = 0,1 sin(20πt). Sommando n(t) al segnale e(t) precedente, otteniamo la forma d'onda mostrata nel grafico di figura 4.25.

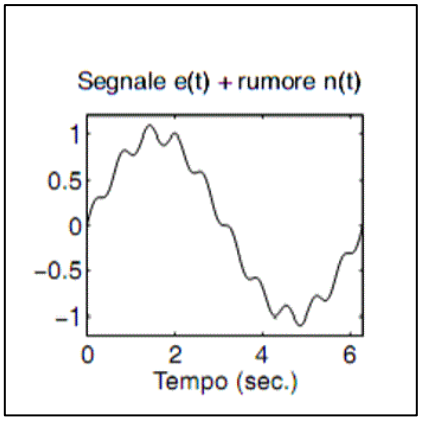


Figura 4.25: Somma dei segnali e(t) = sin(ωt) e n(t) = 0,1 sin(20πt).

A questo punto entra in gioco la derivazione dei componenti della forma d'onda di figura 4.25. Perciò, prima, la derivata del singolo segnale utile sarebbe stata quella di figura 4.26.

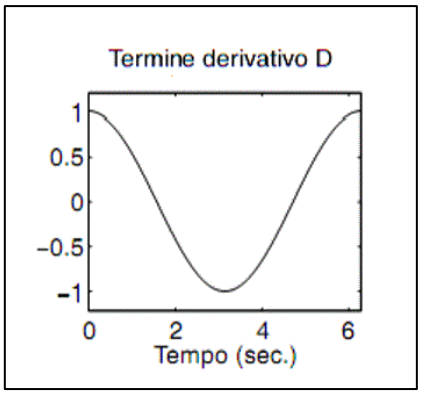


Figura 4.26: Andamento del termine derivativo iniziale.

Ora, invece, a causa dell'alta frequenza del rumore, la cui derivata all'uscita del blocco derivatore diviene $u_n = 2\pi \sin(20\pi t)$, assistiamo ad una degenerazione sostanziale del segnale, che lo rende irriconoscibile (vedi figura 4.27).

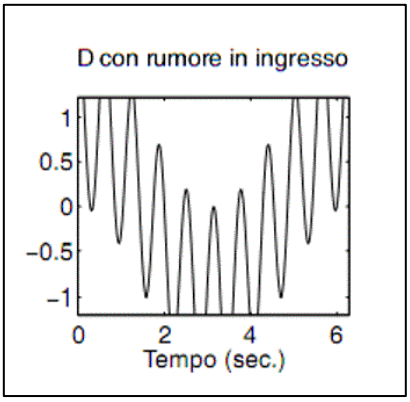


Figura 4.27: Andamento del termine derivativo con rumore in ingresso.

L'azione derivativa non è utilizzabile da sola, perché farebbe perdere al sistema di controllo la fondamentale proprietà di essere passa - basso. Spesso è inutilizzata (disinserita), anche in presenza delle altre azioni, sia a causa dei rumori di misura sia per le difficoltà di taratura da parte degli utenti. In alcuni casi, però, le caratteristiche dinamiche del processo possono non richiedere, o, addirittura, non consentire l'impiego di esso.
Da notare, inoltre, come i controllori D o ID non vengano quasi mai menzionati nei testi specifici, in quanto mai utilizzati. Il regolatore D perché non è in grado di controllare l'errore a regime, e il regolatore ID perché presenta un'antirisonanza ad una determinata pulsazione.
Se la banda a ciclo chiuso fosse minore di questa pulsazione, il regolatore ID si comporterebbe esattamente come farebbe un controllore I, mentre nel caso in cui la banda a ciclo chiuso fosse superiore alla pulsazione di anti-risonanza, essa taglierebbe all'interno della banda di interesse. Tuttavia, siccome l'azione derivativa è necessaria fino ad una certa frequenza, oltre la quale è conveniente che vi sia attenuazione (per filtrare il rumore fuori dalla banda di regolazione), si

limita il livello di amplificazione del rumore, introducendo un filtro del primo ordine, di funzione di trasferimento:

$$F(s) = \frac{1}{1 + \frac{sT_D}{N}}$$

con N che, orientativamente, assume valori compresi tra 5 e 10. Dunque, la nuova funzione di trasferimento del blocco derivatore assume la seguente forma:

$$C(s) = \frac{K_P sT_D}{1 + \frac{sT_D}{N}}$$

Ora rimane da capire come scegliere il termine N per fare in modo che il polo – N/ T_D sia fuori dalla banda di controllo.
Per ottenere questo risultato, N dovrà assumere valori bassi, perché più alto risulta essere N e più la nuova funzione di trasferimento combacerà con quella senza filtraggio, tendendo a riproporre il suo stesso comportamento alle alte frequenze.
In figura 4.28 vediamo come l'andamento della risposta in frequenza di un PID migliori in maniera sostanziale se passiamo da una versione senza filtraggio del termine derivativo (linea continua) a una con il filtro (linea tratteggiata).

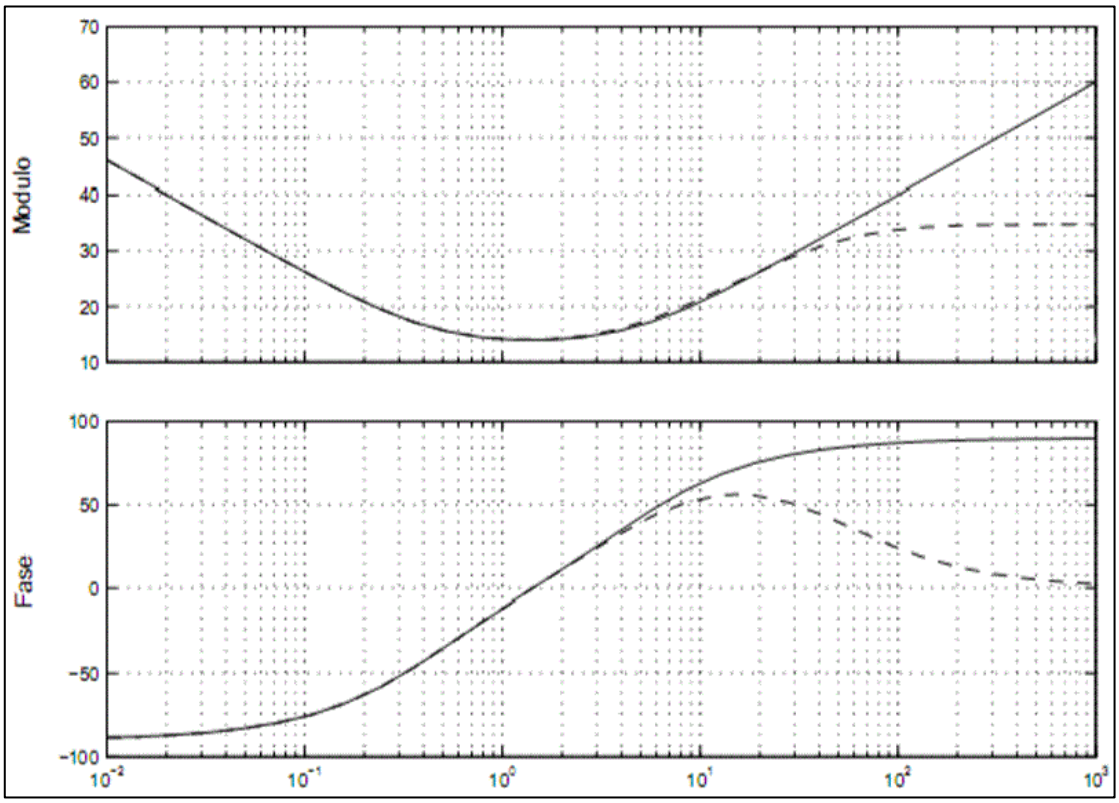


Figura 4.28: Andamenti delle risposte in frequenza del PID.

Nel caso in cui all'ingresso del controllore si presenti un segnale r(t) a gradino, potrebbero sorgere dei problemi se la disposizione dei blocchi, all'interno del controllore stesso, seguisse una disposizione classica in parallelo (vedi figura 4.29).

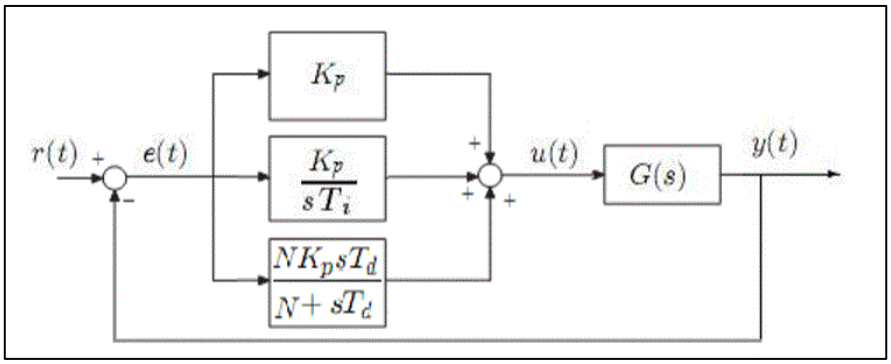


Figura 4.29: Schema di controllo classico in parallelo.

Questo perché, in presenza di un gradino come segnale di riferimento, il termine derivativo, e di conseguenza la variabile di controllo u(t), assumerebbero un andamento di tipo impulsivo. Questa brusca variazione è in contrasto con il requisito di moderazione del controllo e può provocare la saturazione dell'attuatore e l'allontanamento del sistema dalla condizione di linearità, con riferimento alla quale normalmente si progetta il regolatore.
Per queste ragioni, frequentemente l'azione derivativa è esercitata sulla sola variabile di uscita y(t).
Poiché essa è l'uscita di un sistema che usualmente ha le caratteristiche di un filtro passa -basso, le sue variazioni istantanee (e quindi la sua derivata) sono in genere contenute e la presenza dell'azione derivativa non provoca il suddetto andamento impulsivo di u(t).
Dunque, lo schema viene modificato, come illustrato nella seguente configurazione:

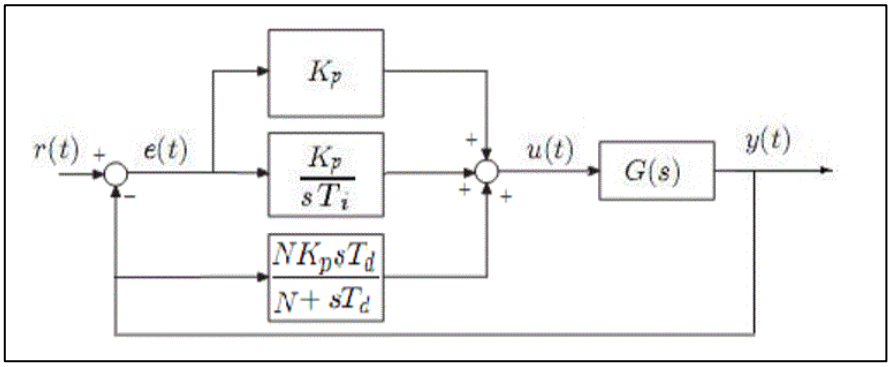


Figura 4.30: Schema di controllo classico in parallelo.

Nelle applicazioni industriali più diffuse, anche l'azione derivativa viene associata all'azione proporzionale, così da creare i cosiddetti controllori PD i quali possono essere utilizzati nei sistemi in cui si hanno improvvise variazioni di carico. Alcune tipologie di tali impianti possono essere sistemi di controllo per servomotori, oppure sistemi che non presentano problemi di stabilità e di prestazioni statiche ma che, invece, richiedono una buona velocità di risposta.

4.6) Metodi di taratura o tuning del PID

Dopo aver fatto una attenta e profonda analisi delle equazioni che governano i controllori PID e trovati quali sono i parametri da modificare per regolarne il comportamento, occorrono delle tecniche per poter settare questi parametri, in modo che un controllore possa funzionare correttamente se connesso a sistemi industriali generici, quindi diversi tra loro.
In molte applicazioni industriali, infatti, la costruzione di un buon modello dell'impianto può essere piuttosto onerosa, soprattutto a fronte di esigenze di controllo non particolarmente spinte. Per questi casi sono disponibili delle tecniche di taratura dei parametri del PID (K_P, T_I e T_D), che fanno riferimento a poche e semplici prove da eseguirsi sull'impianto. Occorre, in altre parole, trovare dei metodi di tuning per i parametri che siano indipendenti dal sistema da controllare.
Andremo ora a presentare due metodi di taratura dei regolatori PID, entrambi basati su considerazioni empiriche.

4.6.1) Metodo della risposta al gradino (I metodo di Ziegler-Nichols)

Questo metodo è basato sull'ipotesi di poter descrivere il processo da sottoporre a controllo attraverso una funzione di trasferimento di primo grado, BIBO stabile (e quindi che il polo della sua funzione di trasferimento sia reale negativo) e di guadagno statico, A, positivo.
I processi a cui si applica solitamente questo metodo sono quelli che hanno risposta al gradino monotona, la quale è tipica dei processi termici, nei quali il carattere distribuito del fenomeno del trasferimento del calore dà luogo a un ritardo iniziale e poi ad una salita progressiva e, generalmente, priva di oscillazioni. Pertanto, risulta un modello del tipo:

$$G(s) = \frac{A \cdot e^{-\tau s}}{1 + Tts}$$

cioè dotato di una funzione di trasferimento che abbia un guadagno statico A, un tempo di ritardo apparente τ ed una costante di tempo T.
Ora si cerca una stima dei parametri della funzione G(s) in modo sperimentale, procedendo ad applicare in ingresso al processo (staccato dalla catena di controllo e quindi in anello aperto) un segnale a gradino di ampiezza unitaria e misurando la risposta temporale del processo.
I parametri che caratterizzano la funzione di trasferimento del processo vengono successivamente determinati dai dati sperimentali.
Questi dati vengono ottenuti prendendo il grafico relativo alla risposta a gradino del sistema e tracciando su di essa, in corrispondenza del punto di flesso della curva, una retta tangente alla curva, facendola intersecare con l'asse dei tempi e con l'asse che rappresenta il guadagno statico A.

La procedura è illustrata in figura 4.31:

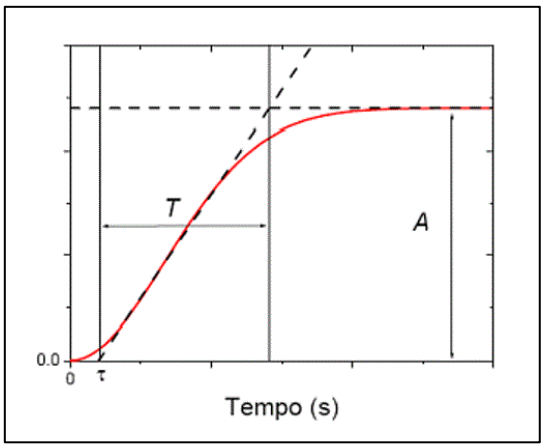


Figura 4.31: Procedura per la determinazione dei valori A, T e τ.

I dati sperimentali (curva rossa continua) sono approssimati nella zona del flesso con la linea tangente (curva nera tratteggiata). L'intercetta della curva tratteggiata con l'asse dei tempi fornisce la stima del ritardo τ, mentre la costante di tempo T viene fornita dal tempo di salita della curva, fino a quando l'approssimazione tratteggiata raggiunge il guadagno in continua A, guadagno che è considerato uguale al valore asintotico (ricordiamo che il segnale di ingresso ha ampiezza unitaria).
Per applicare il metodo, però, il gradino va applicato in condizioni di stazionarietà, cosa non facile da riconoscere automaticamente senza disporre di informazioni sulla dinamica del processo.
Inoltre, il gradino va scelto sufficientemente ampio, perché i parametri della risposta siano ben riconoscibili anche in presenza di rumore di misura, ma non troppo, per non coinvolgere le non linearità e/o per non creare pericolo per l'impianto.
Una volta noti A, τ e T è possibile calcolare i valori che ottimizzano la resa del controllore, trovando il miglior compromesso tra stabilità e velocità di risposta.
Le relazioni che forniscono i valori dei parametri del controllore in funzione dei dati ottenuti sperimentalmente sono riportate nella seguente tabella, dove per R si è considerato il valore dato da τ/T.

	P	PI	PD	PID
K_P	$\frac{1+R/3}{A\,R}$	$\frac{0.9+R/12}{A\,R}$	$\frac{1.25+R/6}{A\,R}$	$\frac{1.33+R/4}{A\,R}$
$1/T_I$		$\tau\cdot\frac{30+3R}{9+20R}$		$\tau\cdot\frac{32+6R}{13+8R}$
T_D			$\tau\cdot\frac{6-2R}{22+3R}$	$\tau\cdot\frac{4}{11+2R}$

Figura 4.32: Tabella per la determinazione dei parametri del PID.

La determinazione dei parametri τ e T della curva risente però dell'approssimazione della costruzione grafica e della difficoltà di individuazione univoca del punto di flesso della curva e della relativa tangente.

I parametri del PID, calcolati con questo metodo, forniscono buoni risultati nella reiezione dei disturbi sul carico, consentendo di abbattere i disturbi di un fattore 4. Per spiegare il significato di questo fattore, si pensi al transitorio imposto dai disturbi sul carico. Generalmente esso è costituito da oscillazioni smorzate. Il rapporto tra il primo e il secondo picco di queste oscillazioni è tipicamente 4 (il secondo picco è attenuato di un fattore 4 rispetto al primo) e, in altre parole, corrisponde ad un rapporto di decadimento pari a 0,25.
Un altro modo grafico di procedere con il calcolo dei parametri attraverso il primo metodo di Ziegler-Nichols (presente in alcuni testi come metodo alternativo a questo, anche se meno preciso) è quello di prolungare la retta tangente del grafico proposto in figura 4.33 fino a farla intersecare con l'asse delle ordinate. Operando in questo modo si ottiene un valore indicato con il termine a.

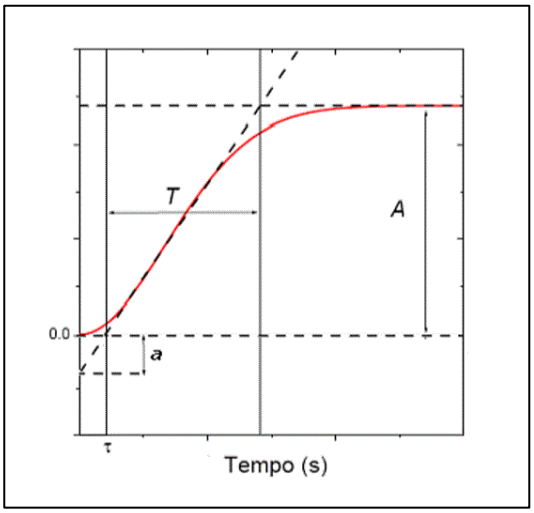


Figura 4.33: Procedura alternativa per la determinazione dei valori A, T e τ.

La procedura di calcolo dei valori dei parametri del PID viene modificata e la tabella viene variata come segue:

	P	PI	PD	PID
K_P	$\frac{1}{a}$	$\frac{0.9}{a}$	$\frac{1.25}{a}$	$\frac{1.33}{a}$
$1/T_I$		$\frac{1}{3\tau}$		$\frac{1}{2\tau}$
T_D			$\frac{\tau}{3}$	$\frac{\tau}{2}$

Figura 4.34: Tabella per la determinazione dei parametri del PID.

Volendo dare chiarezza alla tabella del primo metodo, possiamo ricorrere ad un semplice esempio numerico.

Supponiamo che, in risposta ad un gradino di ampiezza unitaria, il sistema risponda con un grafico come quello seguente:

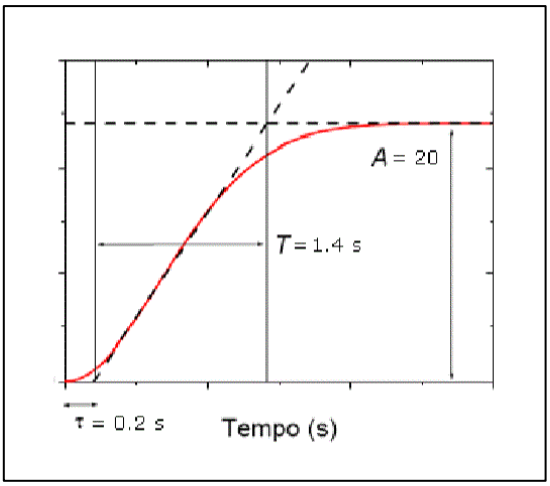


Figura 4.35: Esempio con valori numerici di A, T e τ.

Con i dati dell'esempio scritti in figura 4.35 ricaviamo una R pari a 0,143 circa. La tabella dei valori dei parametri caratterizzanti del controllore dunque sarà la seguente

	P	PI	PD	PID
K_P	0.366	0.319	0.445	0.478
$1/T_I$		0.513		0.465
T_D			0.051	0.071

Figura 4.36: Tabella con valori numerici dei parametri del PID.

4.6.2) Metodo della risposta in frequenza (II metodo di Ziegler-Nichols)

Questo metodo alternativo viene utilizzato con più frequenza rispetto al precedente e, a differenza di esso, opera con il sistema in retroazione unitaria, cioè si considera il sistema in catena chiusa, in cui è già presente il controllore PID.
L'idea con cui il metodo viene applicato è basata sulla conoscenza del punto sulla curva di Nyquist della funzione di trasferimento del processo G(s), in cui la stessa curva di Nyquist interseca l'asse reale negativo. Ricordiamo, inoltre, che il sistema retroazionato è BIBO stabile ad anello chiuso solo se il guadagno in continua ad anello aperto è mantenuto ad un livello sufficientemente piccolo, variabile da sistema a sistema. Infatti, di norma, aumentando il guadagno in continua oltre un certo valore, il sistema diventa instabile.

Tra la zona di stabilità (poli della funzione di trasferimento ad anello chiuso con parte reale negativa) e la zona di instabilità (poli della funzione di trasferimento con parte reale positiva) esiste un limite (poli a parte reale nulla), in cui il sistema non diverge, ma oscilla con ampiezza di oscillazione costante. Quindi, se aumentando il guadagno in continua si arriva ad una situazione oscillatoria (che significa avere un'oscillazione costante nel tempo), possiamo assumere che un valore di guadagno in continua nettamente inferiore a tale limite possa corrispondere ad una condizione di lavoro ragionevole.
Per poter applicare questo metodo si procede mediante una sequenza di passaggi che vengono illustrati nel seguente elenco:

- per prima cosa bisogna annullare, o quantomeno ridurre al minimo livello possibile, le azioni di integrazione e derivazione, facendo lavorare il controllore in modo puramente proporzionale. Per fare ciò si settano le corrispondenti variabili di controllo T_I e T_D al valore nullo;
- successivamente si aumenta in maniera progressiva (e prudentemente) il valore della variabile di controllo Kp, fino a quando si osserva che l'uscita y(t) del sistema oscilla in modo permanente (limite di stabilità). Indichiamo con K0 il valore limite di Kp, detto anche guadagno critico, e con T0 il periodo di oscillazione, misurato quando viene raggiunto questo valore limite di Kp. Se il sistema non entra mai in oscillazione, il metodo non è applicabile. In figura 4.37 è riportata l'uscita del sistema in oscillazione permanente;

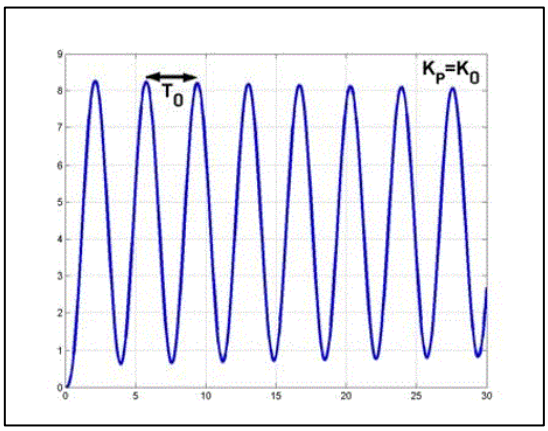


Figura 4.37: Andamento dell'uscita del sistema in oscillazione permanente.

- a questo punto si ricorre ad una tabella di conversione (riportata di seguito), per settare correttamente i parametri di controllo, così da avere una buona reiezione dei disturbi sul carico:

	P	PI	PD	PID
K_P	$0.5K_0$	$0.45K_0$	$0.5K_0$	$0.6K_0$
T_I		$0.85T_0$		$0.5T_0$
T_D			$0.2T_0$	$0.12T_0$

Figura 4.38: Tabella per la determinazione dei parametri del PID.

Si osservi che i valori riportati nella tabella non sono poi così strani: in presenza di controllore PI è necessario ridurre l'azione proporzionale per evitare oscillazioni che portano all'instabilità, dato che la presenza dell'azione integrale può introdurle di default. Aggiungendo anche l'azione derivativa, è possibile aumentare sia quella proporzionale che quella integrale, in quanto il derivatore funge da smorzatore.
In generale, il metodo in anello chiuso fornisce risultati migliori rispetto al precedente metodo ad anello aperto, ma presenta l'inconveniente di dover portare il sistema ai limiti della stabilità (si consideri che sono pochi i casi in cui il proprietario dell'impianto permette di farlo, in quanto il sistema, fortemente sollecitato, potrebbe facilmente guastarsi).
Anche in questo caso, per chiarire maggiormente l'analisi del metodo, ricorriamo ad un esempio numerico nel quale l'uscita del sistema, sollecitata fino all'oscillazione permanente e controllata da un blocco esclusivamente proporzionale, è presentata nel grafico di figura 4.39.

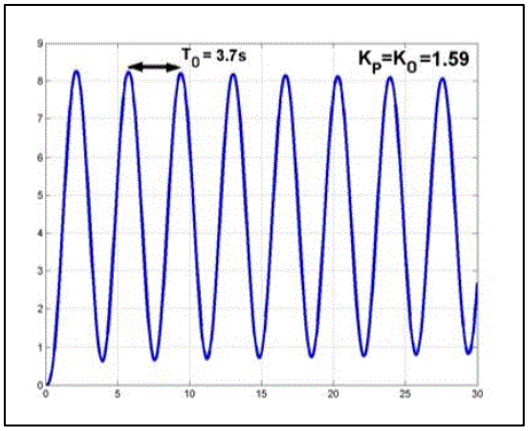


Figura 4.39: Esempio con valori numerici K_0 e T_0.

Di conseguenza, la relativa tabella assumerà i valori riportati di seguito:

	P	PI	PD	PID
K_P	0.795	0.716	0.795	0.954
T_I		3.145		1.85
T_D			0.74	0.44

Figura 4.40: Tabella con valori numerici dei parametri del PID.

Per concludere, ricordiamo alcuni limiti pratici dei metodi di tuning utilizzati dai controllori appena descritti.
Noi abbiamo sempre assunto che i processi da controllare siano lineari e stazionari, purtroppo, però, tali approssimazioni non sono sempre valide. In particolare, uno degli errori più comuni è quello di ottimizzare i parametri del controllore, quando il processo da controllare si trova in condizioni molto diverse da quelle operative.
In tal caso, eventuali non-linearità possono giocare un ruolo importante e l'ottimizzazione iniziale può perdere di significato.

In altri casi, le oscillazioni che devono essere innescate per tarare il controllore secondo il metodo della risposta in frequenza, possono provocare danni al processo stesso, come detto in precedenza. Quindi, nella pratica, bisogna operare con prudenza, ricordando che i metodi semi-empirici appena descritti forniscono una ottimizzazione solo parziale del controllore. Il risultato finale dipende strettamente dal comportamento dinamico del processo sotto controllo.
Per applicazioni critiche, si può ricorrere ad una ottimizzazione sperimentale dei parametri, utilizzando come punto di partenza quelli ottenuti con metodi semiempirici, oppure è necessario ricorrere a metodi di progetto più sofisticati, che però qui non verranno trattati.

5) Applicazione per tuning del PID

5.1) Definizione dell'applicazione sviluppata

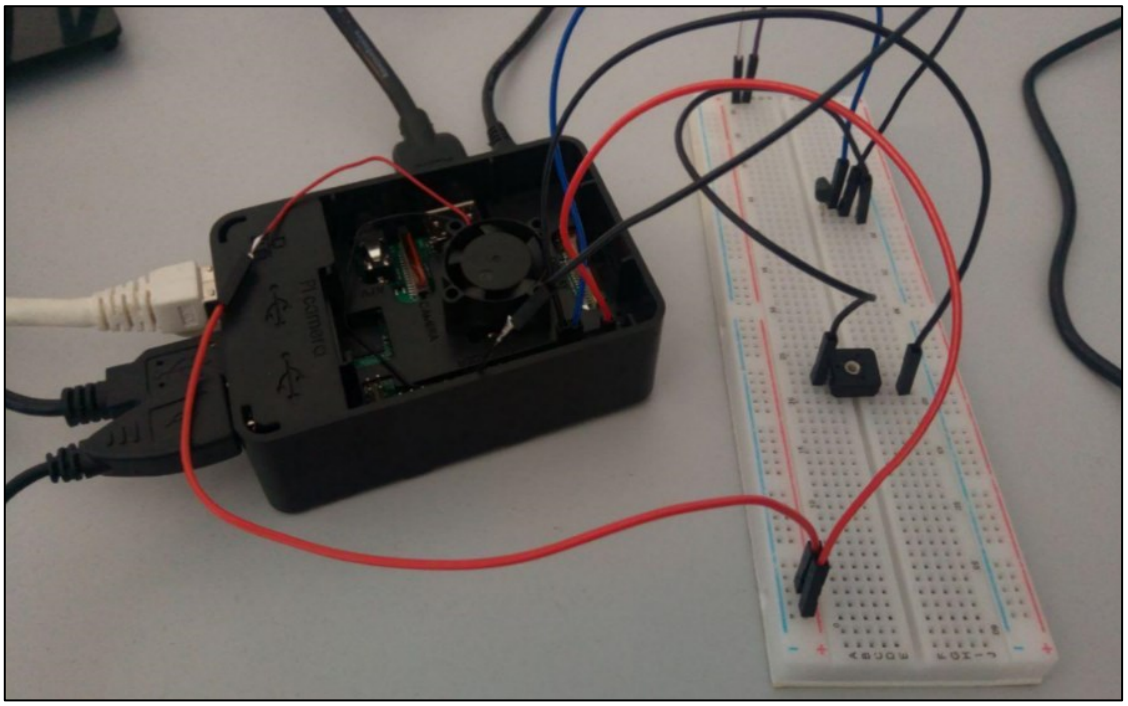


Figura 5.1: Immagine dell'applicazione sviluppata.

Dato l'utilizzo della Raspberry PI3 come Gateway, si è pensato di sviluppare un controllo di temperatura della CPU della stessa, implementando un controllore PID al suo interno ed effettuando il tuning dei parametri, secondo il metodo di Ziegler-Nichols.
L'applicazione è composta da un controllore sviluppato nell'ambiente di programmazione Node-RED .Node-RED è uno strumento di sviluppo basato sul flusso per la programmazione visiva, elaborato originariamente da IBM per il cablaggio di dispositivi hardware, API e servizi online come parte di Internet of Things. Esso è adibito al controllo della temperatura della CPU della Raspberry PI3 (attiva, cioè, una ventola posta al di sopra della stessa, per mantenere la temperatura pari al setpoint).
Sempre in Node-RED è stata implementata una sezione relativa al salvataggio dei valori d'interesse all'interno del database MySQL, situato localmente alla Raspberry PI3, ed una dashboard, in cui vi è la possibilità di visionare gli andamenti della temperatura e la modifica dei coefficienti del PID.
La Raspberry PI3 è stata affiancata da un circuito elettronico cablato su breadboard, in cui è presente anche la ventola controllata dal controllore PID.
Di seguito andremo ad elencare l'hardware utilizzato per la configurazione dell'applicazione, per poi passare alla spiegazione del software sviluppato su Node-RED.

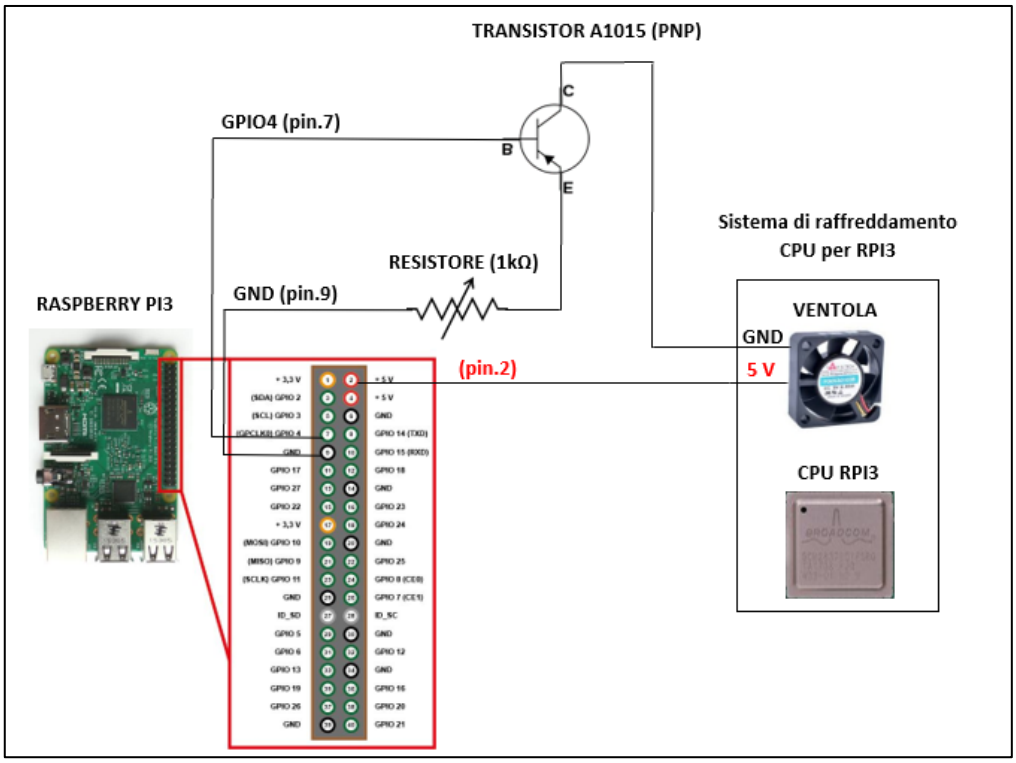


Figura 5.2: Rappresentazione del circuito sviluppato.

5.2) Hardware utilizzato

1) Raspberry PI3 model b+

Figura 5.3: Raspberry PI3 model b+.

Raspberry Pi 3 Model B + è l'ultimo prodotto della gamma Raspberry Pi 3, con processore quad core a 64 bit a 1,4 GHz, 2,4 GHz dual-band e funzionalità wireless LAN 5GHz, Bluetooth 4.2 / BLE, Ethernet più veloce e PoE tramite un CAPPELLO PoE separato.
La LAN wireless a doppia banda è dotata di certificazione di conformità modulare, consentendo alla scheda di essere progettata in prodotti finali, con una riduzione significativa del test di conformità LAN wireless, migliorando sia i costi che il time to market.
Raspberry Pi 3 Model B + mantiene la stessa impronta meccanica di entrambe le Raspberry Pi 2 modello B e Raspberry Pi 3 modello B.

Specifiche

- SoC: Broadcom BCM2837B0 quad-core A53 (ARMv8) a 64 bit a 1,4 GHz;
- GPU: Broadcom Videocore-IV;
- RAM: 1 GB LPDDR2 SDRAM;
- Rete: Gigabit Ethernet (tramite canale USB), 2,4 GHz e 5 GHz 802.11b / g / n / ac
- Wi-Fi;
- Bluetooth: Bluetooth 4.2, Bluetooth Low Energy (BLE);
- Conservazione: Micro-SD;
- GPIO: intestazione GPIO a 40 pin, popolata;
- Porte: HDMI, jack audio-video analogico da 3,5 mm, 4x USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI);
- Dimensioni: 82mm x 56mm x 19.5mm, 50g.

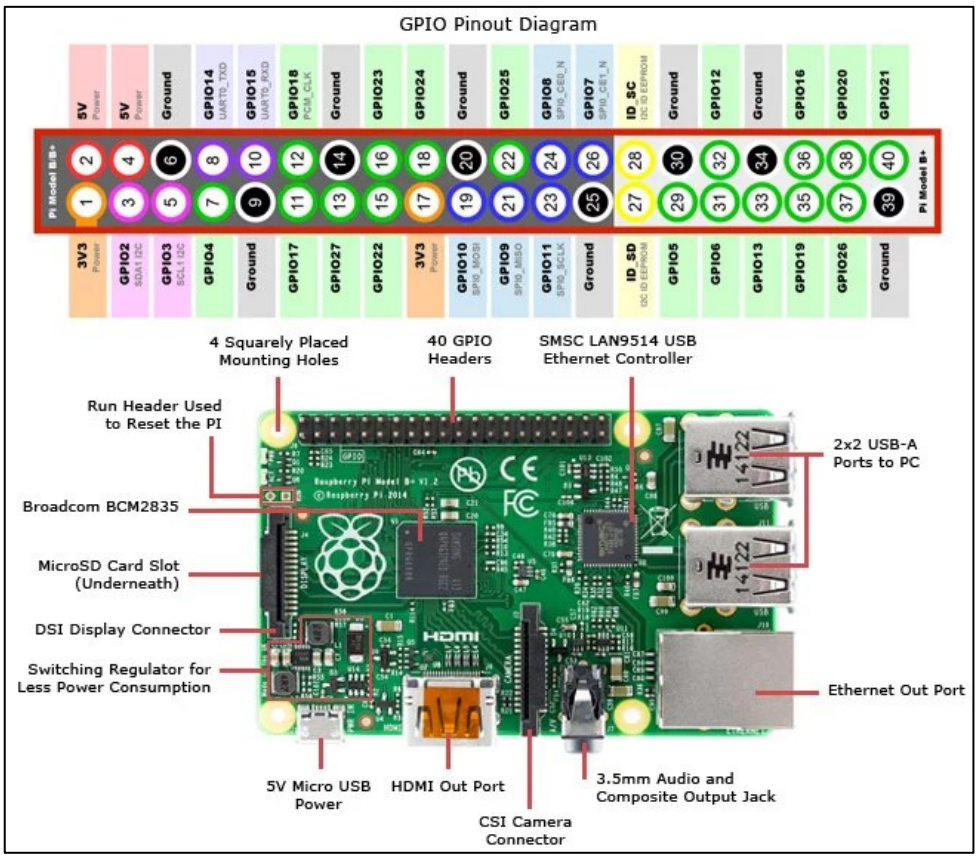


Figura 5.4: Rappresentazione GPIO della Raspberry PI3 model B+.

Gestione energetica

Un intelligente circuito integrato di gestione dell'alimentazione (PMIC) sostituisce i componenti discreti e fornisce un'alimentazione più fluida.

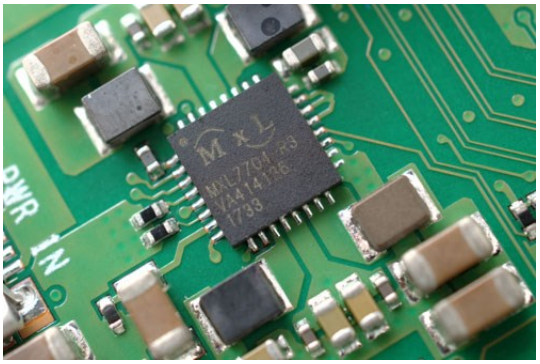


Figura 5.5: Modulo integrato per la gestione dell'alimentazione.

Sistema Operativo:

Raspbian è un sistema operativo basato sui rilasci ufficiali di Debian per l'Architettura ARM, adattato per l'utilizzo su Raspberry Pi.
Rispetto a Debian, tra le note più importanti si trova la modifica dell'interfaccia grafica da GDM3 a LightDM, che consente di alleggerire il carico della CPU nell'esecuzione del sistema. Oltre a questo, si trova l'aggiunta in APT di alcuni repository, ufficiali e non, contenenti quasi tutti i pacchetti ricompilati nella corretta architettura.
Viene, inoltre, aggiunto il logo del progetto, un lampone, sia come sfondo del desktop che come contorno della shell durante il caricamento del sistema.
In Raspbian è integrato raspi-config, uno strumento ad interfaccia a riga di comando per gestire in modo accessibile agli utenti varie configurazioni, quali il cambio della password, della lingua, della configurazione della tastiera e del metodo di avvio.

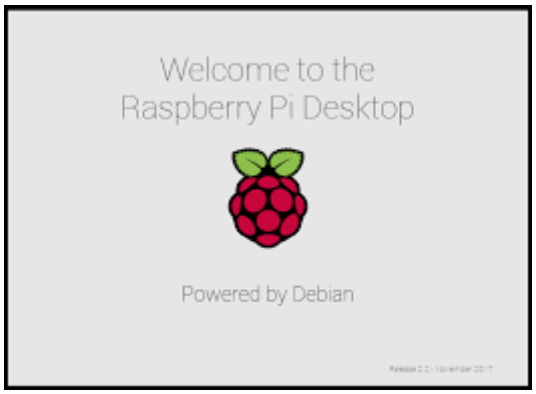


Figura 5.6: Interfaccia iniziale della Raspberry PI3 con Raspbian.

2) Breadboard

La breadboard (o anche detta basetta sperimentale) è uno strumento utilizzato per creare prototipi di circuiti elettrici. A differenza della basetta millefori, che è un circuito stampato su basetta ramata, sopra il quale vengono saldati i componenti e i collegamenti che formano il prototipo (difficilmente riusabile), la breadboard non richiede saldature ed è completamente riutilizzabile (per cui viene utilizzata soprattutto per circuiti temporanei). Sebbene essa venga usata normalmente per la prototipazione di circuiti semplici, può essere usata anche per testare interi calcolatori.

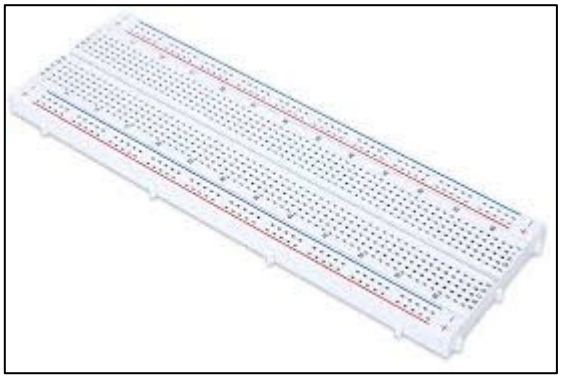


Figura 5.7: Immagine breadboard tradizionale.

3) Resistore 102 (1KΩ)

Il resistore (chiamato impropriamente anche resistenza per metonimia), è un tipo di componente elettrico destinato a opporre una specifica resistenza elettrica al passaggio della corrente elettrica.
I resistori hanno innumerevoli applicazioni, sia in apparecchiature elettriche che elettroniche. Sono a volte utilizzati per convertire energia elettrica in energia termica.

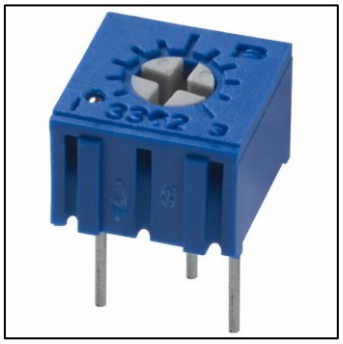


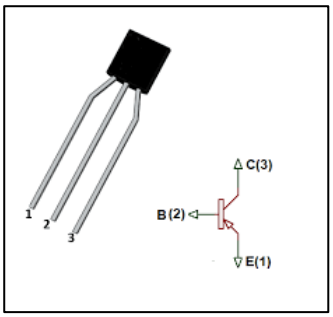
Figura 5.8: Resistore tipo 102 da 1KΩ.

TYPE	DESCRIPTION
Categories	Potentiometers, Variable Resistors Trimmer Potentiometers
Manufacturer	Bourns Inc.
Series	Trimpot® 3296 - Sealed
Packaging	Tube
Part Status	Active
Resistance	1 kOhms
Power (Watts)	0.5W, 1/2W
Tolerance	±10%
Temperature Coefficient	±100ppm/°C
Number of Turns	25
Adjustment Type	Top Adjustment
Resistive Material	Cermet
Mounting Type	Through Hole
Termination Style	PC Pins
Size / Dimension	Rectangular - 0.375" x 0.190" Face x 0.395" H (9.53mm x 4.83mm x 10.03mm)
Base Part Number	3296W

Figura 5.9: Datasheet resistore 102.

4) Transistor PNP A1015

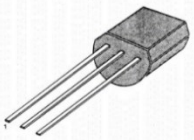
Il transistor a giunzione bipolare (abbreviazione comunemente utilizzata: BJT, dall'inglese Bipolar Junction Transistor) è una tipologia di transistor largamente usata nel campo dell'elettronica analogica, principalmente come amplificatore di corrente e come interruttore.
Esso è composto da tre strati di materiale semiconduttore drogato, solitamente silicio, in cui lo strato centrale ha drogaggio opposto agli altri due, in modo da formare una doppia giunzione p-n. Ad ogni strato è associato un terminale: quello centrale prende il nome di base, quelli esterni sono detti collettore ed emettitore. Il principio di funzionamento si fonda sulla possibilità di controllare la conduttività elettrica del dispositivo, e quindi la corrente elettrica che lo attraversa, mediante l'applicazione di una tensione tra i suoi terminali. Tale dispositivo coinvolge sia gli elettroni, sia le lacune (portatori di carica maggioritari), per cui questo tipo di transistor è detto bipolare.
Costituisce la famiglia più diffusa in elettronica insieme al transistor ad effetto di campo, rispetto al quale è in grado di offrire una maggiore corrente in uscita, con lo svantaggio, tuttavia, di non avere il terminale di controllo isolato (gate).
Il transistor a giunzione bipolare può essere usato classicamente in tre configurazioni diverse dette a base comune, a collettore comune o ad emettitore comune. Questi termini si riferiscono al terminale privo di segnale (di solito perché collegato al potenziale di riferimento, direttamente o tramite un condensatore di bypass).



LOW FREQUENCY AMPLIFIER

- Collection Dissipation : P_C(max) = 400mW
- Collector-Emitter Voltage : V_{CEO} = -50V

TO - 92



1. Emitter 2. Collector 3. Base

Absolute Maximum Ratings (TA=25°C)

Characteristic	Symbol	Rating	Unit
Collector-Base Voltage	V_{CBO}	-50	V
Collector-Emitter Voltage	V_{CEO}	-50	V
Collector Current	I_C	-150	mA
Collector Dissipation	P_C	400	mW
Junction Temperature	T_J	150	°C
Storage Temperature	T_{STG}	-55~+150	°C

Electrical Characteristics (TA=25°C)

Characteristic	Symbol	Test Conditions	Min	Max	Unit
Collector-Base Breakdown Voltage	BV_{CBO}	I_C= -100μA, I_E=0	-50		V
Collector-Emitter Breakdown Voltage	BV_{CEO}	I_C= -0.1mA, I_B=0	-50		V
Collector Cut-off Current	I_{CBO}	V_{CB}= -50V, I_E=0		-0.1	μA
Emitter Cut-off Current	I_{EBO}	V_{EB}= -5V, I_C=0		-0.1	μA
DC Current Gain	h_{FE}	V_{CE}= -6V, I_C= -2mA	70	400	
Collector-Emitter Saturation Voltage	$V_{CE(sat)}$	I_C= -100mA, I_B= -10mA		-0.3	V
Base-Emitter Saturation Voltage	$V_{BE(sat)}$	I_C= -100mA, I_B= -10mA		-1.1	V
Base-Emitter Voltage	V_{BE}	I_E= -310mA		-1.45	V
Transition Frequency	f_T	V_{CE}= -10V, I_C= -1mA			
		f=30MHz	80		MHz

h_{FE} CLASSIFICATION

Classification	O	Y	GR
h_{FE}	70-140	120-240	200-400

Figura 5.10: Transistor PNP tipo A1015 e datasheet.

5.3) Software Node-RED

Passiamo ora alla valutazione del software utilizzato per l'applicazione, sviluppato, come detto in precedenza, in Node-Red.
Per accedere alla schermata di sviluppo di Node-RED occorre aprire il menù a tendina presente sul desktop della Raspberry PI3, andare su "Programmazione" e cliccare sull'icona "Node-RED", come mostrato in figura 5.11.

Figura 5.11: Desktop della Raspberry PI3 con sistema operativo Raspbian.

Si aprirà una schermata terminale sul desktop inerente a Node-RED, in cui saranno visualizzate le relative informazioni e l'indirizzo internet su cui trovare l'ambiente di sviluppo Node-RED. Successivamente, utilizzando il terminale della Raspberry, andremo ad attivare il Server MySQL (che permette la lettura e scrittura su database MySQL), tramite i seguenti comando Linux:

1- **sudo su** (permette il riconoscimento come amministratore e quindi attiva funzionalità aggiuntive, come permessi di lettura e scrittura di cartelle e attivazione di software specifici);
2- **mysql -u root -p** (permette l'accesso al Server MySQL, con richiesta di eventuale password impostata durante l'installazione);
3- password **abc,** impostata dal sottoscritto durante la configurazione iniziale;
4- **quit,** per terminare l'utilizzo del database e chiudere la sessione del Server MySQL.

Prima di entrare nel vivo della programmazione, occorre attivare anche il broker Mosquitto. Esso è un broker di messaggi open source (licenza EPL/EDL), che implementa il protocollo MQTT versioni 5.0, 3.1.1 e 3.1.
Mosquitto è leggero ed è adatto per l'uso su tutti i dispositivi, dai computer a scheda singola (a bassa potenza) ai server completi. Il protocollo MQTT fornisce un metodo funzionale per eseguire la messaggistica, utilizzando un modello di pubblicazione / sottoscrizione. Ciò lo rende adatto alla messaggistica Internet of Things, ad esempio con sensori a bassa potenza o dispositivi mobili come telefoni, computer incorporati o microcontrollori.
Per attivare Mosquitto, occorre aprire una nuova sessione del terminale su Raspberry ed eseguire i seguenti comandi:

1- **sudo su** (per accedere, come detto in precedenza, come amministratore);
2- **mosquitto** (per attivare il broker mosquitto atto alla comunicazione tramite protocollo MQTT).

Il risultato ottenuto dai passi precedenti è visibile in figura 5.12, dove, nella sessione del terminale a sinistra, vi è l'attivazione del broker Mosquitto, mentre nella sessione a destra si nota l'attivazione del Server MySQL.

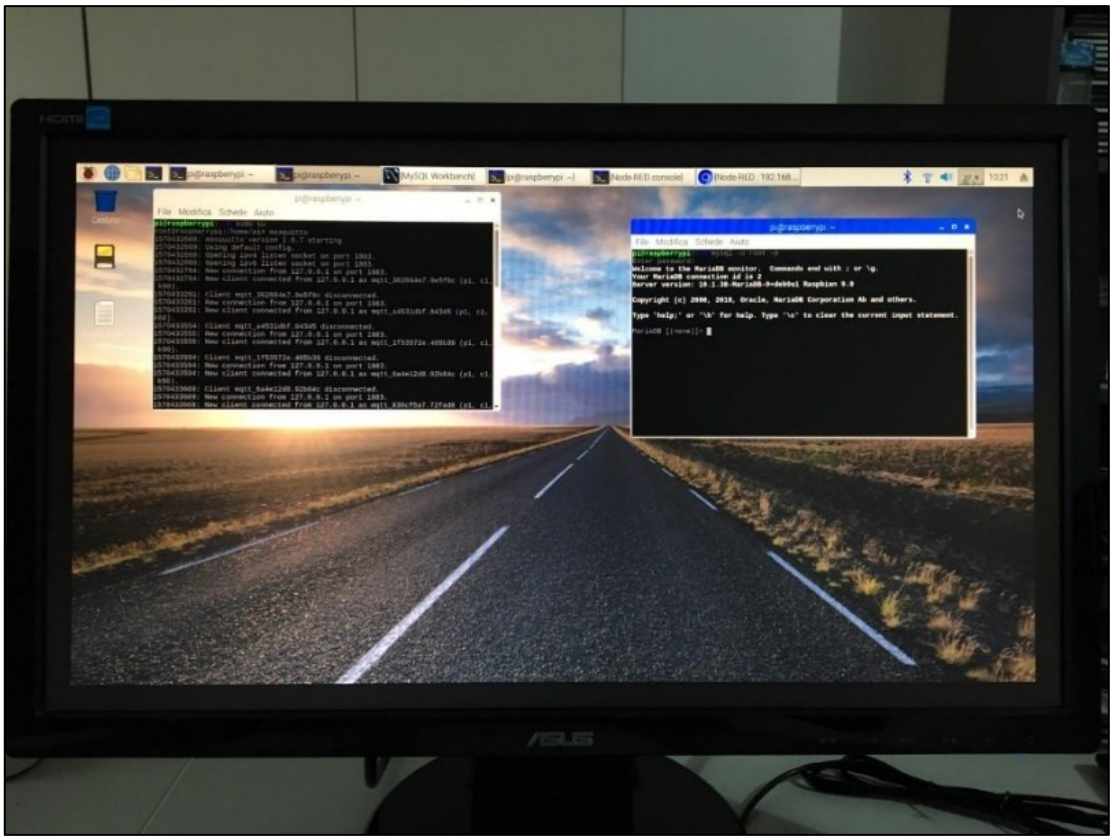


Figura 5.12: Attivazione di Mosquitto e Server MySQL.

Una volta eseguito la configurazione preliminare, andiamo nel browser di ricerca e digitiamo l'indirizzo presente nella schermata del terminale relativa a Node-RED, mostrato nel riquadro in rosso in figura 5.13.

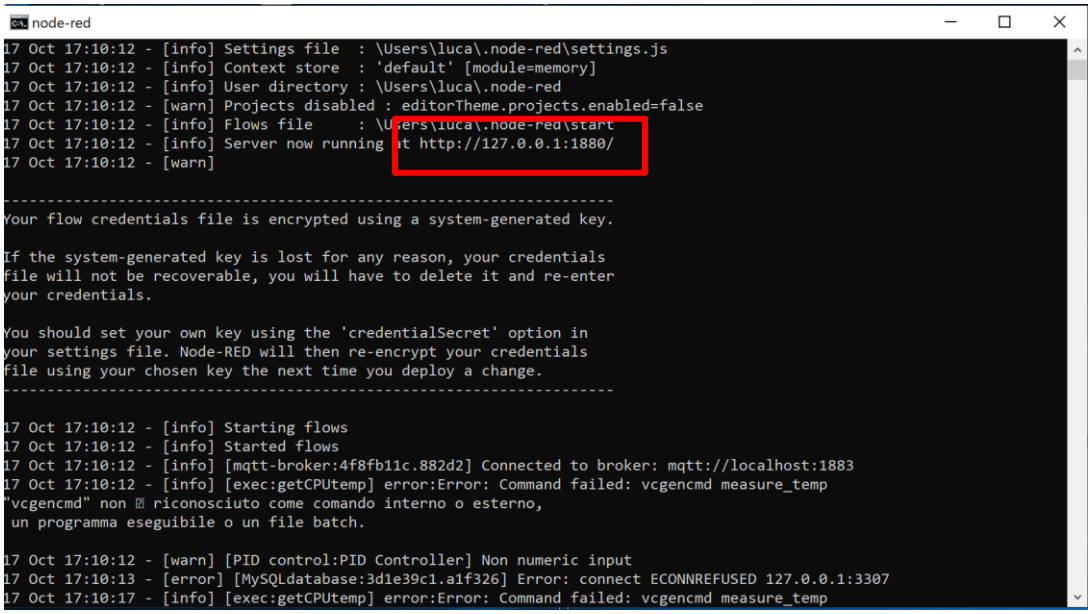


Figura 5.13: Terminale dopo attivazione Node-RED con relativo indirizzo.

Il programma sviluppato in Node-RED è mostrato in figura 5.14, in cui viene implementato il controllore PID, la lettura/scrittura dei parametri fondamentali (come i coefficienti del PID, il valore della temperatura misurata e il timestamp) e la dashboard sulla Raspberry PI3.
Il blocco del controllore PID (blocco in giallo in figura 5.14) ha come input il set-point (valore di riferimento della temperatura in °C) ed i parametri del PID (Kp,Ki e Kd), che vengono settati utilizzando la schermata utente (dashboard), sempre sviluppata in Node-RED.
Tutti i blocchi della dashboard sono riconoscibili in figura dal colore azzurro. Essi sono sia di input (utilizzati per inserire dei valori) che di output (utilizzati per la visualizzazione degli andamenti di particolari valori nel tempo).

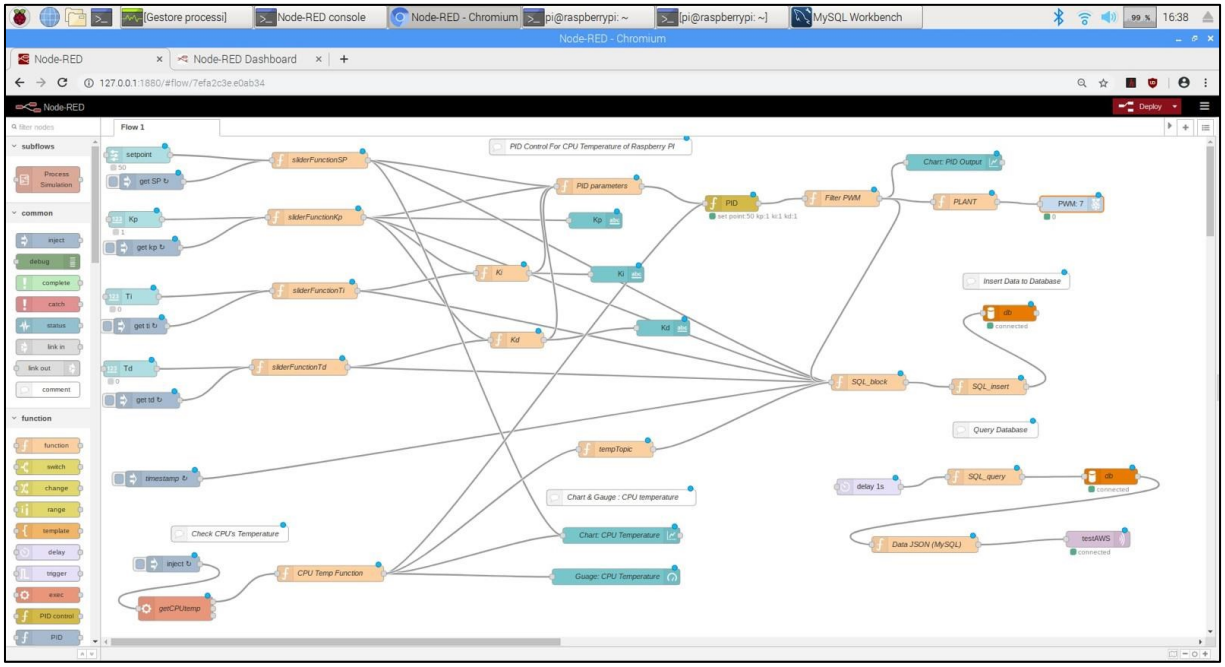


Figura 5.14: Programma sviluppato in Node-RED.

Di notevole importanza sono i blocchi denominati Ki e Kd, che hanno la funzione di determinare i valori dei coefficienti Ki e Kd del PID, partendo dai valori di Kp, Ti e Td.
In figura 5.15 è possibile vedere la logica implementativa per ottenere il valore di Ki, partendo dai valori di Kp (coefficiente proporzionale) e Ti (costante di tempo integrativa).
Dalla teoria dei PID, abbiamo la seguente formula matematica:

$$Ki = \frac{Kp}{Ti}$$

Valutando la linea di codice numero 15 della figura 5.15 è possibile notare l'implementazione della formula precedente.

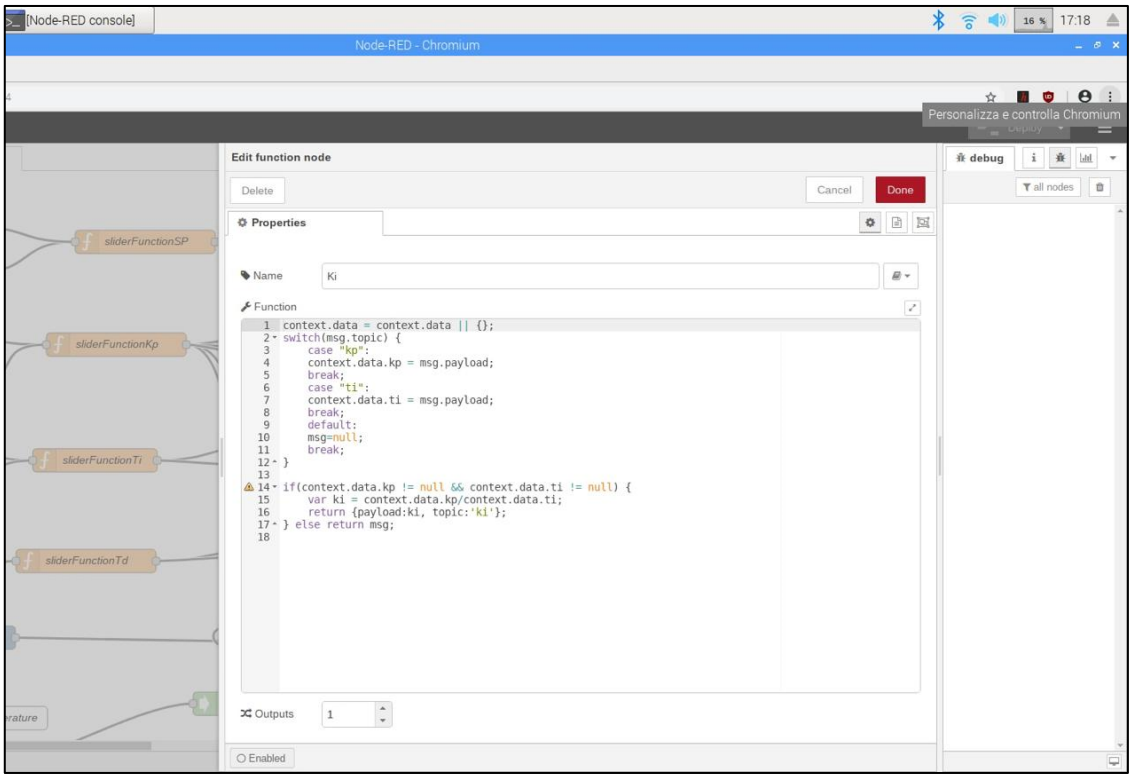


Figura 5.15: Blocco funzione Node-RED Ki.

Analogamente a Ki, per Kd è stato seguito lo stesso principio, andando però ad utilizzare la formula matematica seguente, derivata sempre dalla teoria dei PID, dove utilizzando Kp (coefficiente proporzionale) e Td (costante di tempo derivativa), riusciamo a ricavare il valore del coefficiente derivativo Kd:

$$Kd = Kp * Td$$

Valutando la linea di codice numero 15 della figura 5.16 è possibile notare l'implementazione della formula precedente.

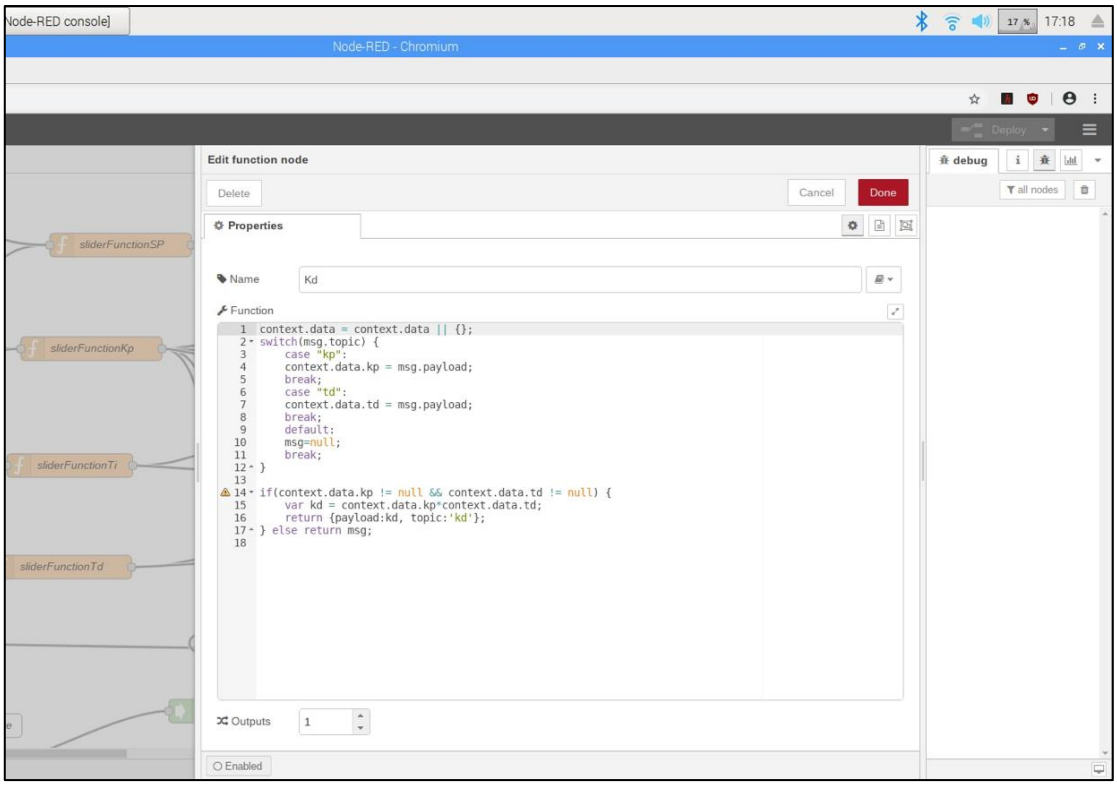


Figura 5.16: Blocco funzione Node-RED Kd.

Utilizzando il secondo metodo di Ziegler-Nichols, e considerando la dinamica lenta del sistema (in cui la grandezza controllata risulta essere la temperatura), sono stati scelti i valori di: Kp pari a 1.8, Ti pari a 4.5 e Td pari a 1.12, con i relativi valori dei coefficienti del PID pari a: Kp = 1.8, Ki = 0.4 e Kd = 2.016.

5.4) Taratura ad anello chiuso: metodo di Ziegler-Nichols

Per effettuare la taratura del regolatore PID, dopo un'opportuna analisi sui potenziali metodi, è stato scelto quello di Ziegler-Nichols. Tale procedimento si suddivide in 5 passi:

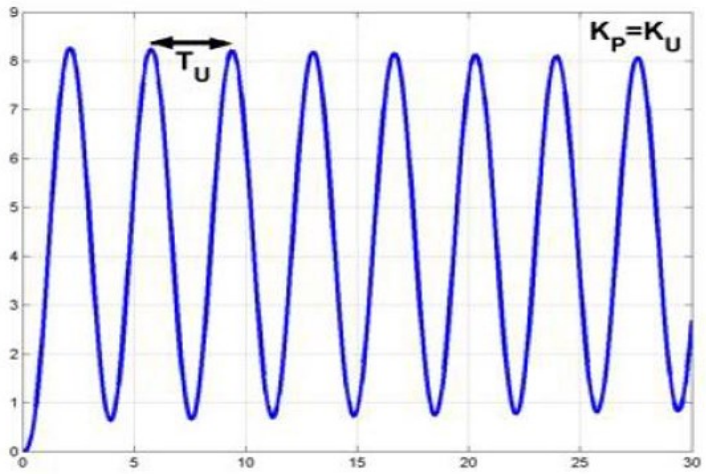


Figura 5.17: Grafico per la determinazione dei parametri di Ziegler-Nicols.

1. si chiude l'anello, inserendo un controllore puramente proporzionale con un basso guadagno Kp;

2. si aumenta progressivamente il guadagno Kp, fino a che si innesca una oscillazione regolare permanente;

3. si definisce K_u come il valore del guadagno Kp, in modo tale che, in corrispondenza della risposta del sistema al gradino unitario, si abbiano delle oscillazioni ad ampiezza costante;

4. si misura il periodo dell'oscillazione TU;

5. si calcolano i guadagni secondo le formule presenti in figura 5.18:

	K_I	T_I	T_D
P	$0{,}5Ku$		
PI	$0{,}45Ku$	$Tu/_{1,2}$	
PID	$0{,}6Ku$	$Tu/_{2}$	$Tu/_{8}$

Figura 5.18: Tabella per il calcolo dei coefficienti del PID.

Seguendo fedelmente i seguenti passi sono stati definiti i parametri del PID, che permettono di effettuare una buona regolazione:

- Kp = 1,8
- Ti = 4,5
- Td = 1,2

A causa di una quantizzazione poco puntuale del valore segnalato dal sensore relativo alla temperatura del processore, l'azione del regolatore risulta essere non ottimale, come visibile in figura 5.19.

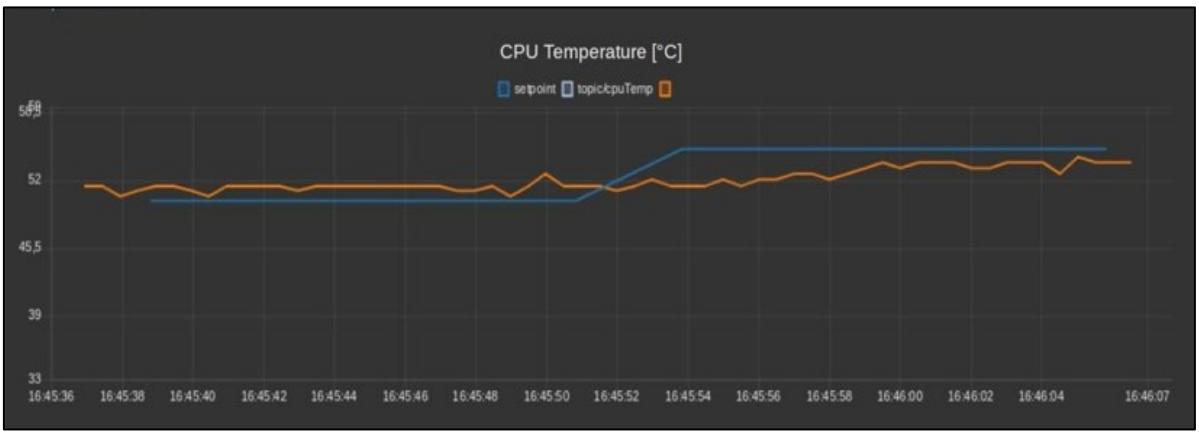


Figura 5.19: Andamenti delle temperature (riferimento e misurata) con l'azione del controllore PID.

5.5) Salvataggio dei dati

Per quanto riguarda il salvataggio dei dati sul database, è stato utilizzato il software MySQL Workbench installato direttamente sulla Raspberry PI3, creando due tabelle, "value_pid" (periodo d'inserimento dell'ordine del secondo) e "value_pid2" (periodo d'inserimento dell'ordine del minuto), in cui le colonne identificano i valori d'interesse, che successivamente verranno inviati al Cloud tramite protocollo MQTT.
La scelta di salvare i dati con periodi diversi su due tabelle differenti ha l'obiettivo di generare uno storico accurato con la tabella "value_pid", utilizzata per il caricamento dei dati in Cloud e per la visualizzazione degli andamenti delle variabili sulla dashboard locale.
Avendo "value_pid" un periodo di inserimento breve, il vantaggio è quello di permettere una visualizzazione più precisa dei dati, ma con una maggiore occupazione di memoria.
La seconda tabella "value_pid2", determina uno storico meno preciso ma con una minore occupazione di rete. Questo permette, a parità di memoria disponibile, di poter avere a disposizione informazioni più "vecchie" rispetto a quelle presenti in "value_pid".
L'utilità risiede nel fatto di poter andare a valutare situazioni passate (dell'ordine di settimane), che nella tabella con periodo di aggiunta più frequente non sono più disponibili, perché già sovrascritte.

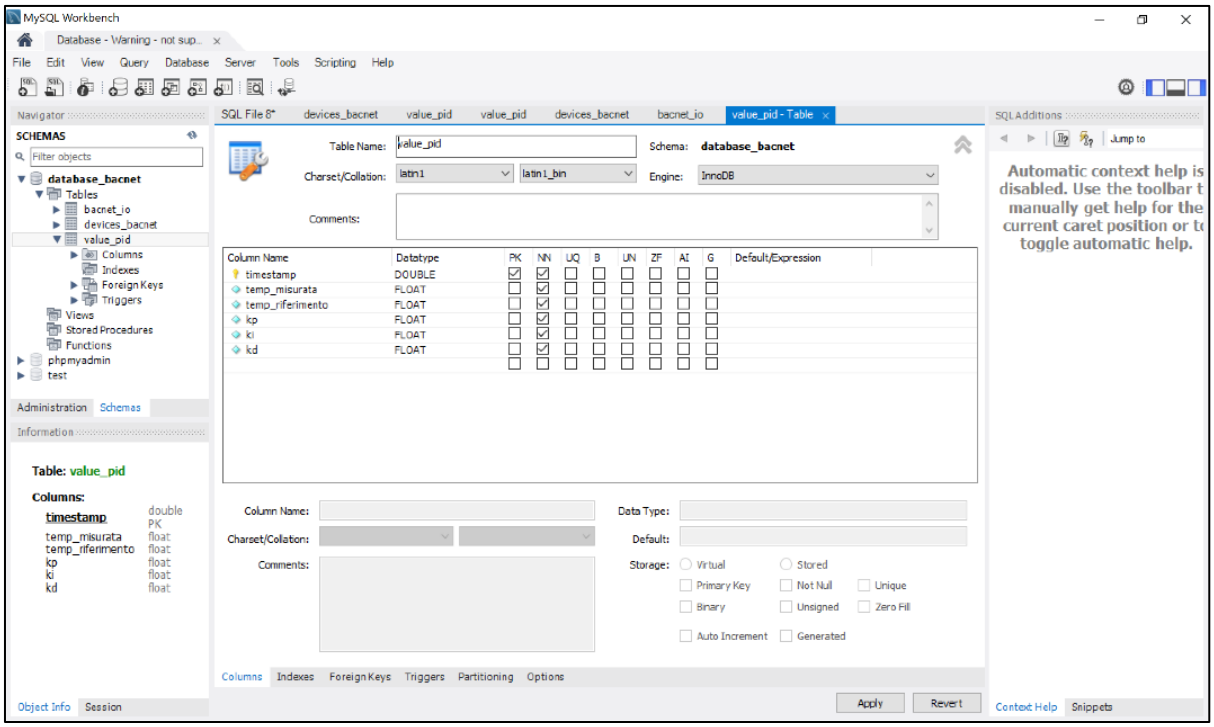


Figura 5.20: Definizione dei parametri da salvare sul database MySQL locale.

In figura 5.20, vediamo la costruzione delle colonne delle tabelle “value_pid” e “value_pid2”, entrambe costituite dal tempo in cui la riga viene inserita (timestamp), dalla temperatura misurata (temp_misurata), dalla temperatura di riferimento (temp_riferimento) e dai coefficienti del PID (Kp, Ki e Kd).
In figura 5.21 è possibile valutare il risultato ottenuto dall’attivazione del codice mostrato in figura 5.14, dove ogni riga identifica il salvataggio del set di dati nell’ordine del secondo.

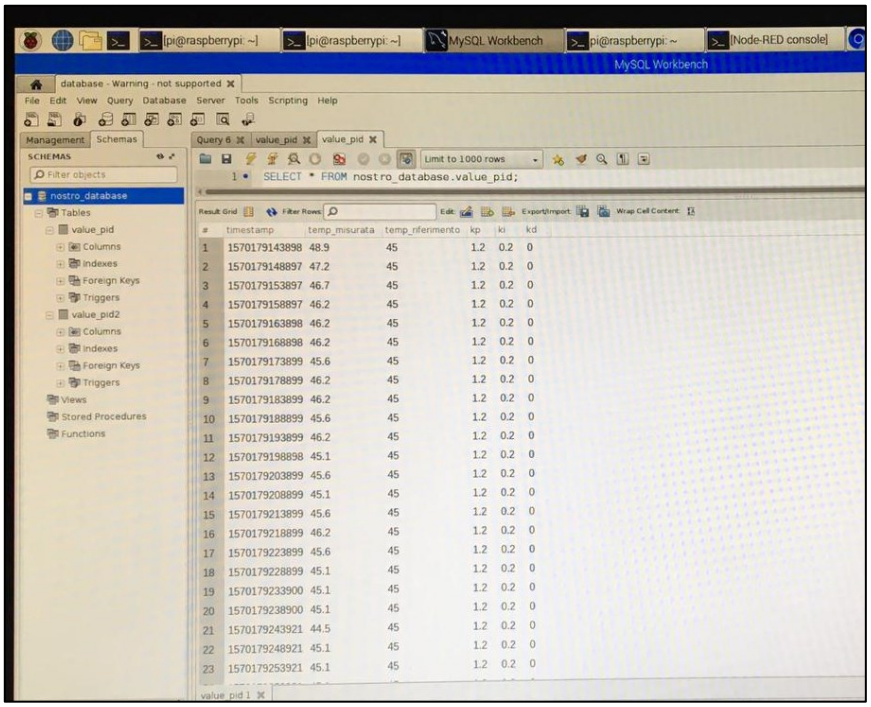


Figura 5.21: Salvataggio dei dati su database MySQL locale alla Raspberry PI3.

5.6) Dashboard locale

Infine, andiamo a valutare la funzionalità “dashboard” che mette a disposizione Node-RED e che consente di raggruppare tutte le informazioni più importanti relative al sistema di controllo.

In figura 5.22 è visibile la dashboard implementata per quest’applicazione. Al suo interno sono stati inseriti elementi per la visualizzazione, come gli andamenti nel tempo della temperatura misurata e di riferimento (visibili nel grafico in alto a sinistra), il valore di uscita del PID (in basso a sinistra), la temperatura attuale della CPU della Raspberry PI3 (in alto a destra), espressa in °C.
Per la modifica è stata predisposta una piccola sezione di setting del valore di riferimento della temperatura (in °C) e dei parametri del PID, la quale è situata in basso a destra nella dashboard.

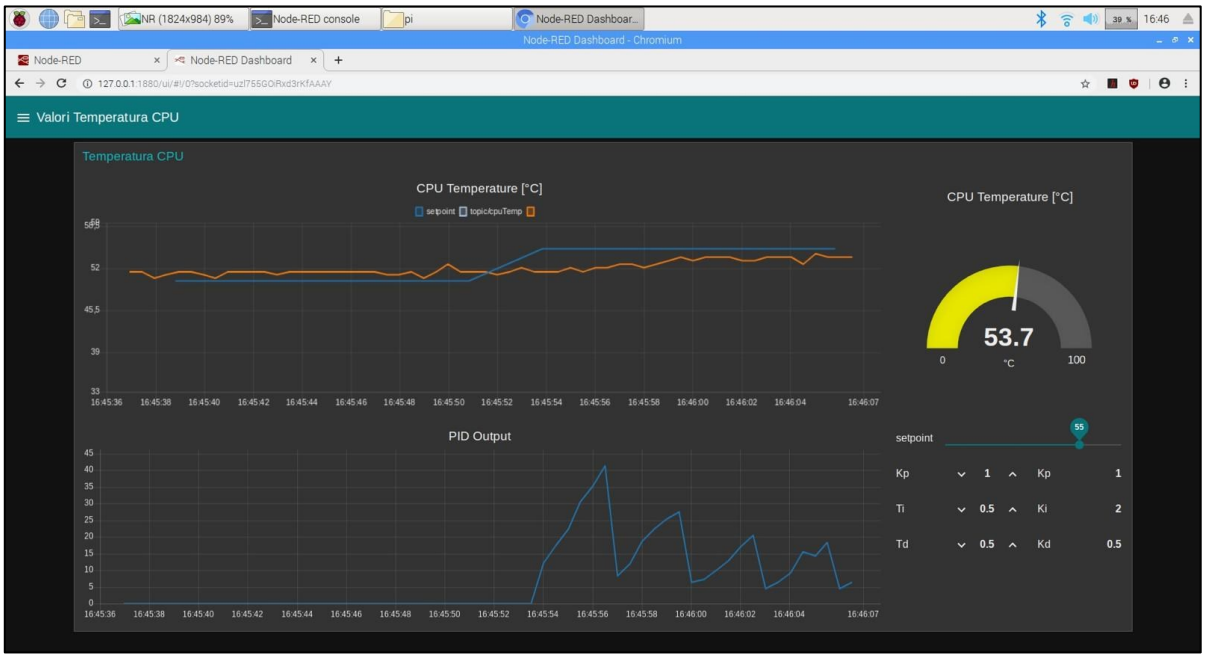


Figura 5.22: Dashboard per la visualizzazione dei parametri del sistema controllato.

Il valore della temperatura, rilevato tramite un apposito comando, costituisce l’input del regolatore PID, definito tramite Node-RED insieme ai parametri che lo caratterizzano (setpoint, Kp, Ki, Kd). L’output del PID sarà necessario per determinare l’azione della PWM, ovvero un tipo di modulazione digitale che permette di ottenere una tensione media variabile dipendente dal rapporto tra la durata dell'impulso positivo e di quello negativo (duty cycle). L’indicazione percentuale della tensione è compresa in un range tra 0 e 100: il valore 0 % indica la massima velocità raggiunta dalla ventola. Di conseguenza, al crescere del fattore percentuale si ha una relativa decrescita dell’azione della ventola.
Di notevole importanza è la funzionalità che permette di valutare come cambiano i valori dei coefficienti Ki e Kd in relazione alle modifiche sui valori Kp, Ti e Td. Ciò permette di aiutare l’operatore nella visualizzazione immediata dei valori che si ottengono relativamente ad una modifica dei valori di input.

6) Applicazione in Cloud

Oltre alla parte di salvataggio dei dati e del controllo, che cambiano in base alla natura del sistema e che sono oggetto del mio tirocinio, il prototipo TOM ha anche il compito di caricare i dati in Cloud, in cui si ha uno spazio di memoria molto maggiore rispetto a quello del gateway. Proprio per questo motivo, parliamo brevemente del lavoro svolto dal tirocinante Ing.Fabi Giorgio, che aggiunto al mio lavoro, permette di avere un prototipo del progetto TOM a tutti gli effetti funzionante.
L'Ing.Fabi ha implementato un caricamento dei dati salvati nel Gateway sul Cloud tramite protocollo MQTT, con una visualizzazione degli stessi parametri tramite una dashboard che, però, è raggiungibile da qualsiasi dispositivo che permette la navigazione sui browser utilizzati quotidianamente.
Il risultato è visibile in Figura 6.1, dove nel grafico a sinistra vi sono gli andamenti del PWM che la Raspberry PI3 genera sul gate del transistor tramite il pinout GPIO4, la temperatura di riferimento (setpoint) e la temperatura effettiva della CPU della Raspberry PI3.
Sulla sezione di destra, invece, è possibile andare a modificare i parametri della temperatura, quelli del PID ed i settaggi per la visualizzazione dei parametri sul grafico di sinistra. Infatti, è possibile scegliere l'intervallo della scala relativa all'asse dell'ordinata (che identifica l'intervallo della temperatura in °C) e dell'ascissa (che identifica il periodo di tempo che vogliamo andare a visionare).

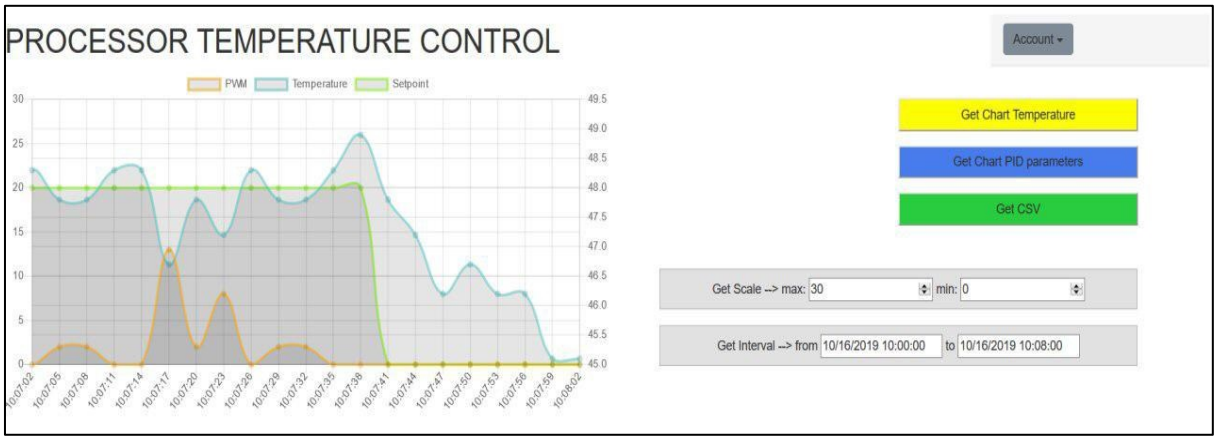


Figura 6.1: Dashboard in Cloud.

Una futura ingegnerizzazione del progetto TOM è quella di definire un algoritmo di prevenzione dei guasti, tramite l'utilizzo del Machine Learning.
Sempre con il Machine Learning si può cercare di rendere il sistema autonomo. Se consideriamo un' applicazione in cui, al variare delle caratteristiche fisiche del materiale utilizzato nel sistema (ad esempio la densità del materiale), si determina un diverso setting dei parametri del PID, è possibile predisporre un algoritmo tale da adattare autonomamente i parametri, in base al materiale che andrà lavorato. In questo modo non si avrà bisogno di un operatore che dovrà modificare ogni volta i parametri del PID, per ottenere una risposta soddisfacente.

7) Analisi dell'architettura sviluppata

7.1) Analisi di performance

Dopo aver effettuato i test sull'architettura implementata, si è voluto effettuare uno studio più approfondito riguardo l'ambiente di sviluppo utilizzato per l'acquisizione dei dati. Nello specifico, si è cercato di individuare, da una potenziale periferica di campo, quale potesse essere il limite del software in relazione alla frequenza di acquisizione dati massima. Per questo studio è stato preso in considerazione un PLC Beckhoff CX5130-0155, creando un canale di comunicazione con la Raspberry attraverso il protocollo BACnet.

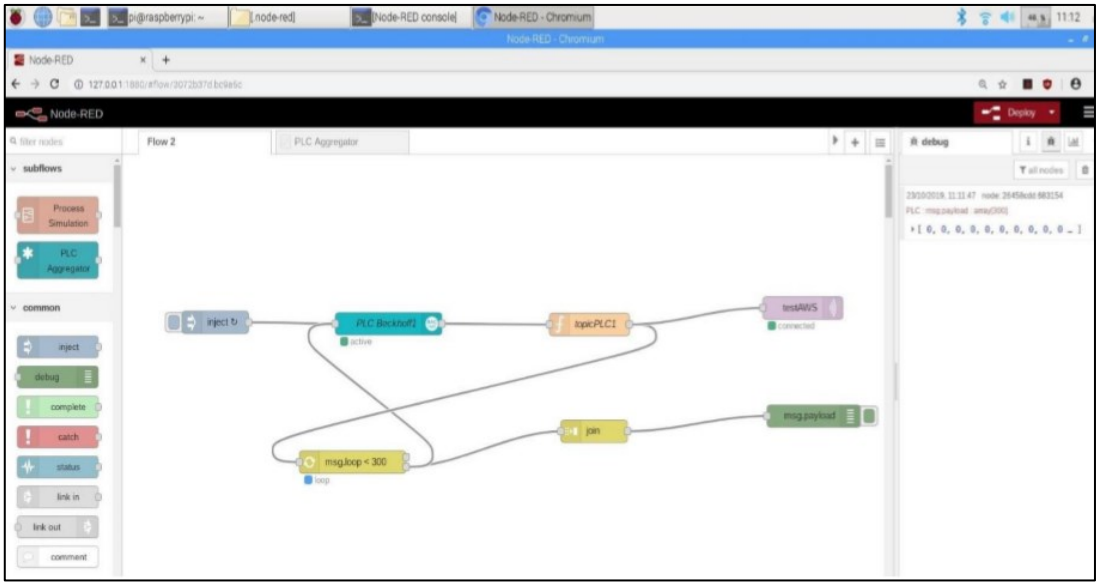
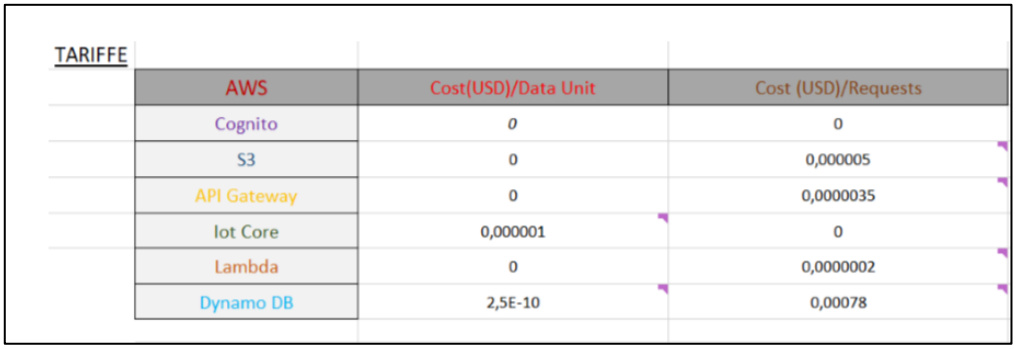


Figura 7.1: Implementazione in Node-RED per valutazione performance.

Dopo aver configurato il blocco BACnet con l'indirizzo del PLC, è stato valutato come 40 ms, il tempo di acquisizione del dato in seguito ad una singola richiesta. Per cercare di ottimizzare i tempi di risposta del device, sono state effettuate in parallelo una serie di richieste al PLC per valutare se, tramite questo protocollo, si potesse avere la possibilità di creare un buffer di richieste, senza dover necessariamente attendere l'esito della risposta. Tale opzione ha permesso, dunque, di ridurre drasticamente i tempi di accesso alle aree di memoria del PLC, ottenendo un controllo continuo delle variabili di interesse. In futuro, sarà necessario estendere tale studio a tutte le componenti o a tutti i protocolli industriali con i quali il software presente sul gateway dovrà interfacciarsi. Qualora sarà verificata la possibilità di implementare tale opzione su diverse periferiche, si potrà rafforzare il ruolo di Node-RED all'interno del progetto TOM. In caso contrario, sarà necessario pensare di intervenire attraverso l'utilizzo di strumenti differenti come, ad esempio, linguaggi di basso livello, che possano ridurre l'accesso alle aree di memoria.

7.2) Analisi economica

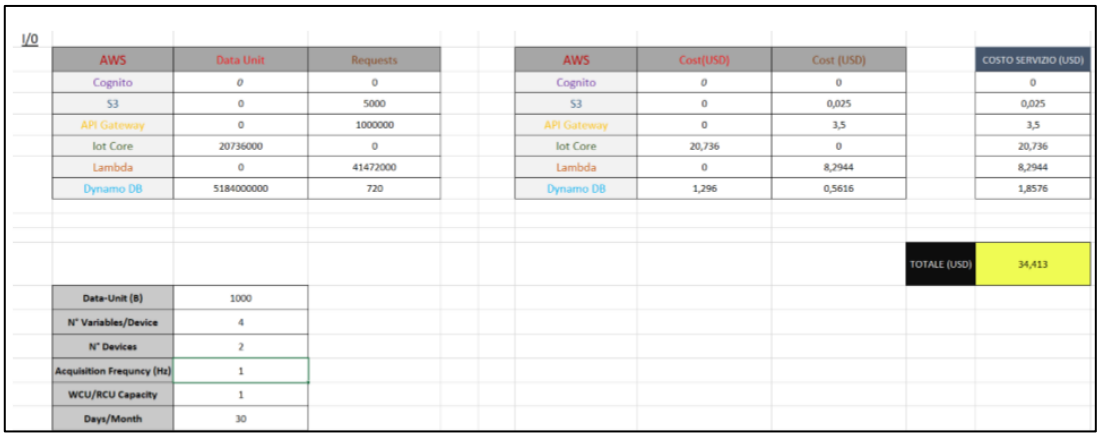
Uno degli obiettivi principali del progetto TOM è quello di creare un'infrastruttura solida e scalabile ma che, allo stesso tempo, fosse in grado di minimizzare i costi di implementazione. Per quanto riguarda la versione TOM sviluppata, il principale ostacolo dal punto di vista economico è rappresentato dall'utilizzo della piattaforma Cloud AWS. In rapporto a ciò, è stato creato un file di calcolo, tramite Microsoft Excel, in grado di effettuare una valutazione dei costi istantanea, sulla base degli input relativi alle periferiche di campo ed alle tariffe dei servizi Amazon che compongono la struttura software.



TARIFFE

AWS	Cost(USD)/Data Unit	Cost (USD)/Requests
Cognito	0	0
S3	0	0,000005
API Gateway	0	0,0000035
Iot Core	0,000001	0
Lambda	0	0,0000002
Dynamo DB	2,5E-10	0,00078

Figura 7.2: Tariffe AWS.



I/O

AWS	Data Unit	Requests
Cognito	0	0
S3	0	5000
API Gateway	0	1000000
Iot Core	20736000	0
Lambda	0	41472000
Dynamo DB	5184000000	720

AWS	Cost(USD)	Cost (USD)
Cognito	0	0
S3	0	0,025
API Gateway	0	3,5
Iot Core	20,736	0
Lambda	0	8,2944
Dynamo DB	1,296	0,5616

	COSTO SERVIZIO (USD)
	0
	0,025
	3,5
	20,736
	8,2944
	1,8576
TOTALE (USD)	34,413

Data-Unit (B)	1000
N° Variables/Device	4
N° Devices	2
Acquisition Frequncy (Hz)	1
WCU/RCU Capacity	1
Days/Month	30

Figura 7.3: Input/Output AWS.

Dall'immagine sopra è possibile capire, sebbene siano presenti dati di prova, in che modo ognuno dei servizi utilizzati incide sul costo complessivo relativo alla piattaforma Cloud. Focalizzando l'attenzione su tale esempio, si può notare che i servizi IoT Core e Lambda incidano maggiormente sul costo mensile, ottenendo così un'informazione importante che permette di effettuare un'ottimizzazione del flusso dati, al fine di minimizzare la spesa. Tale strumento potrà risultare utile all'azienda nella fase di offerta, qualora un cliente richieda il supporto del sistema TOM.

8) Conclusioni

Il progetto TOM (Tecnology Overlay Method) rappresenta un progetto focalizzato sulla predisposizione di un'azienda nell'ambito IoT.
In questo elaborato è stato presentato un percorso di tirocinio atto a sviluppare un prototipo del progetto TOM, con l'obbiettivo di fornire una base solida su cui effettuare studi e modifiche nel breve futuro, visibile in figura 8.1.

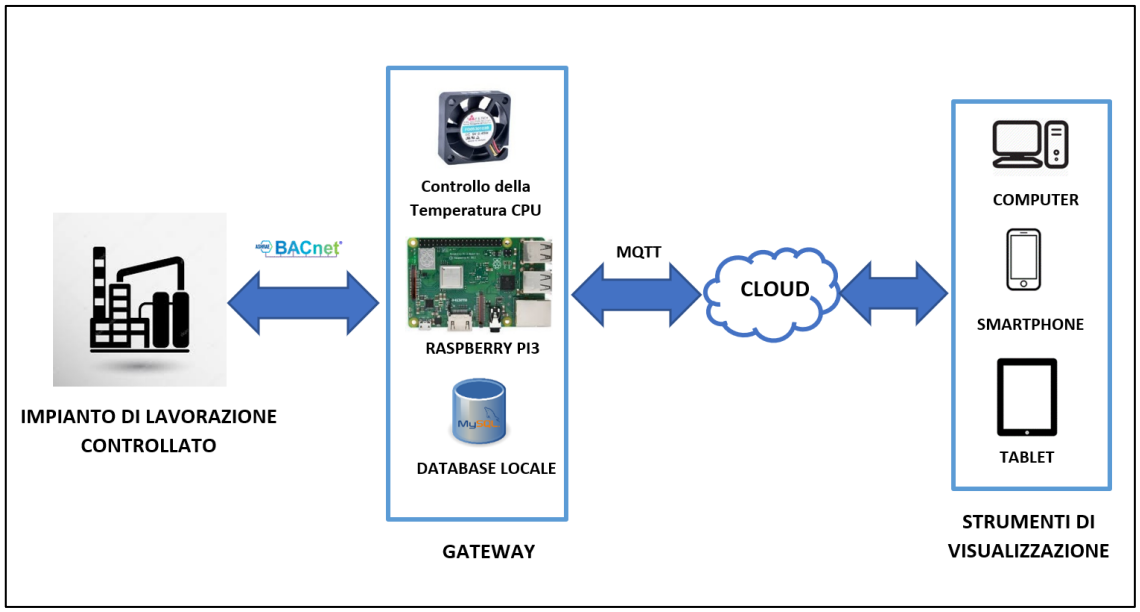


Figura 8.1: Schema del prototipo TOM sviluppato.

Lo sviluppo è stato principalmente quello di creare un driver presente in un Gateway, che permettesse la comunicazione con sistemi automatici (come sensori, PLC, ecc..), atti al controllo delle fasi lavorative presenti nelle aziende.

Figura 8.2: Marchio BACnet.

Il driver effettua la scansione di tutti i dispositivi presenti nella sede lavorativa dell'azienda, e riconoscerà tutti i dispositivi predisposti alla comunicazione tramite protocollo BACnet. In tal modo, esso fornisce, in un database locale al Gateway, il mezzo di salvataggio di tutti i parametri dei dispositivi, con la possibilità di variare quest'ultimi per eventuali modifiche all'impianto di lavorazione.

Il driver è stato sviluppato dal sottoscritto in 2 fasi, nella prima utilizzando un simulatore di device BACnet (BACsim), con l'obiettivo di effettuare prove, a volte anche azzardate, senza compromettere l'integrità di dispositivi costosi.
Una volta appurato il corretto funzionamento del driver, si è passati a testare quest'ultimo con un dispositivo reale, quale il PLC Beckhoff CX5130, uno dei dispositivi maggiormente utilizzati in applicazioni reali.
Come mostrato nel Capitolo 3.4.2, i risultati riguardanti il driver BACnet, in entrambi i casi, in relazione alle richieste affrontate dall'azienda ospitante, risultano ottimi. In particolare, si è ottenuta un'eccellente affidabilità a livello di codice e non si è avuta alcuna perdita d'informazione nel salvataggio di tutti i parametri che definiscono l'impianto di lavorazione nel database.
La parte automatica comprende lo sviluppo di un controllore PID e relativo tuning dei parametri tramite metodo di Ziegler-Nicols, con l'obiettivo di controllare la temperatura della CPU del nostro Gateway (Raspberry PI3).
Inizialmente, si era pensato di sviluppare il PID direttamente nel PLC Beckhoff, con il compito di controllare un servomotore. Questa idea, però, è stata abbandonata, in quanto il software utilizzato (Beckhoff Twincat3) non permette l'esportazione dei coefficienti del PID e, quindi, non vi è la possibilità di associare tali valori ad I/O di tipo BACnet.

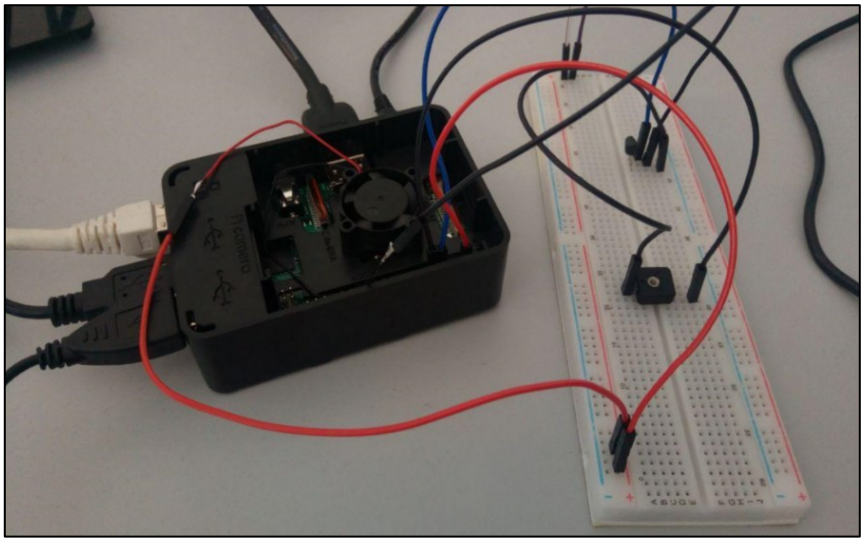


Figura 8.3: Immagine dell'applicazione del controllore PID.

Per lo sviluppo del controllore PID si è usato Node-RED, software di sviluppo a blocchi risultante molto semplice e intuitivo, in cui è stata predisposta una dashboard con l'obiettivo di fornire supporto all'operatore che, idealmente, avrebbe il compito di controllare il sistema di lavorazione.
Una volta ottenuto i parametri del PID tramite la teoria di Ziegler-Nicols, sono bastate piccole modifiche dei coefficienti, considerando la dinamica lenta del sistema, per ottenere un controllore capace di regolare l'azione della ventola presente nel nostro Gateway.
Lo sviluppo di questa semplice applicazione è frutto dell'idea futura dell'azienda, che vorrebbe costruire un Box, cioè il Gateway, contenente la Raspberry PI3 e il sistema di controllo dell'autoventilazione della CPU, con il vantaggio di fornire un valore aggiunto al proprio prodotto.

Infine, abbiamo creato il collegamento tra il Gateway ed il Cloud con la possibilità, accedendo all'indirizzo http://tom2040.s3-website.us-east-2.amazonaws.com/, di poter visualizzare, grazie ad un'altra dashboard predisposta nel Cloud, tutti gli andamenti delle variabili che rappresentano le fasi di lavorazione in qualsiasi momento e su qualsiasi dispositivo a disposizione dell'utente.
L'architettura sviluppata ha risposto positivamente al test effettuato. Gli elementi che compongono la struttura non hanno palesato alcuna criticità durante le operazioni di acquisizione, archiviazione e visualizzazione dei campioni.
Inoltre, a seguito di ulteriori verifiche effettuate al termine della raccolta dati, non vi è stata alcuna perdita di informazione. Tale aspetto è una conseguenza delle considerazioni avvenute in precedenza: mantenendo un tempo di campionamento sui 3 secondi, infatti, è stato possibile eseguire tutte le operazioni correttamente.

9) Appendice A

9.1) Codici del driver BACnet;

dbBacnet.js

```
// Gestione della connessione , delle funzioni di aggiunta e richiesta del database:

// Variabili
var mainBacnet = require('./mainBacnet'); // connessione al mainBacnet.js ( permette l'utilizzo delle funzioni esportate da esso)
var mysql = require('mysql');             // connessione alla libreria mysql

var unixTime = require('unix-time');      // connessione alla
var time = unixTime(new Date());          // libreria unix-time

var deviceList = [];       // Array su cui vengono salvati i dati di rilevanza del device
var values =[];
var numDev = 0;

// Creazione della connessione espilicitando i campi fondamentali del nostro database MySQL:
var db = mysql.createConnection(
{
    host: '127.0.0.1',
    user: 'root',
    password: 'password',
    port: 3306
}
);

// Permette la connessione al database, se la connessione avviene con successo avvisa tramite un messaggio sulla console:
db.connect((err) => {
    if (err) {
        throw err;
    }
    console.log(' +++  Database BACnet Connected  +++ ');
});

// Funzione che permette l'aggiunta di un dispositivo Bacnet nel databse,la condizione di controllo che permette di non aggiungere 2 device uguali è
// definita dal valore dell'id_device (primari key):
function addDevice(device)
{
    var currDev;
    var deviceArray =[];
    deviceArray = [[time,
                    device.currentIdDevice,
                    device.device.address,
                    device.device.deviceId,
                    device.device.maxApdu,
                    device.device.segmentation,
                    device.device.vendorId]];

    currDev = {address: device.device.address, deviceId: device.device.deviceId, numProp: 0, object_prop:[]};
    deviceList.push(currDev);
    numDev++;

    db.query("INSERT INTO database_bacnet.devices_bacnet (time_stamp, id_device,address,deviceID,maxApdu,segmentation, vendorId) VALUES ?", [deviceArray]);
}

// Funzione che aggiunge tutte le proprietà dell'oggetto del device nel database:
function addPropertyIO(object)
{
    var idProp;
    var currProp;
    var record = [[time, object.object.id_bacnet_io, object.object.id_device, object.object.object_Identifier, object.object.object_Name, object.object.object_Type,
        object.object.present_Value, object.object.event_State, object.object.out_of_Service, object.object.update_Interval, object.object.units,
        object.object.min_Pres_Value, object.object.max_Pres_Value, object.object.resolution, object.object.cov_Increment, object.object.high_Limit,
        object.object.low_Limit]];

        db.query("INSERT INTO database_bacnet.bacnet_io (time_stamp, id_bacnet_io, id_device, object_Identifier, object_Name, object_Type, present_Value," +
        " event_State, out_of_Service, update_Interval, units, min_Pres_Value, max_Pres_Value, resolution, cov_Increment, hight_Limit, low_Limit)  VALUES ? ",
        [record]);

        idProp = deviceList[object.object.id_device -1].numProp;

        /* Aggiungi nuovo elemento nell'array */
        currProp = {type: object.object.object_Type,
                    presentValue: object.object.present_Value,
                    instance: object.object.object_Identifier,
                    id_io:object.object.id_bacnet_io
                    };

        deviceList[object.object.id_device - 1].object_prop.push(currProp);
        deviceList[object.object.id_device - 1].numProp++;
}
```

```
79
80  // Funzione che aggiorna il Present Value nel database:
81  function updateInputValue(value, idxDev, idxProp, idxIdIo)
82  {
83      deviceList[idxDev].object_prop[idxProp].presentValue = value;
84
85      db.query("UPDATE database_bacnet.bacnet_io SET present_Value = "+value.toString()+" WHERE id_bacnet_io = "+idxIdIo.toString());
86  }
87
88  // Funzione che richiede il Present Value nel database:
89  function selectOutputValue(idxIdIo,type, instance, address, callback)
90  {
91      db.query("SELECT present_Value FROM database_bacnet.bacnet_io WHERE id_bacnet_io = "+idxIdIo.toString(),
92          function(err, value, fields)
93          {
94              if (callback != 0)
95              {
96                  callback(value[0].present_Value,type, instance, address);
97
98              }
99          });
100 }
101
102 // Funzione che definisce ogni quanto viene fatto il polling in lettura (pollingInput):
103 function startPollingInput()
104 {
105     setInterval(pollingInput, 3000);
106 }
107
108 // Funzione che gestisce il Polling in lettura degli oggetti del device:
109 function pollingInput ()
110 {
111     var currIdIo;
112     var currDev;
113     var currProp;
114     for (var idxDev=0; idxDev<numDev; idxDev++)  // utilizzato per selezionare ogni device
115     {
116         for (var idxProp=0; idxProp<deviceList[idxDev].numProp; idxProp++)  // utilizzato per selezionare ogni oggetto del device
117         {
118             currDev = idxDev;
119             currProp = idxProp;
120             currIdIo = deviceList[idxDev].object_prop[idxProp].id_io;
121
122             mainBacnet.readValue(
123                                  deviceList[idxDev].object_prop[idxProp].type,
124                                  deviceList[idxDev].object_prop[idxProp].instance,
125                                  deviceList[idxDev].address,
126                                  function(value)
127                                  {
128                                     updateInputValue(value, currDev, currProp, currIdIo);
129                                  });
130         }
131     }
132 }
133
134 // Funzione che definisce ogni quanto viene fatto il polling in scrittura (pollingOutput):
135 function startPollingOutput()
136 {
137     setInterval(pollingOutput, 3000);
138 }
139
140 // funzione che gestisce il Polling in scrittura degli oggetti del device:
141 function pollingOutput()
142 {
143     var currIdIo;
144     var currDev;
145     var currProp;
146     var type_obj;
147   for (var idxDev=0; idxDev<numDev; idxDev++)   // utilizzato per selezionare ogni device
148   {
149     for (var idxProp=0; idxProp<deviceList[idxDev].numProp; idxProp++)  // utilizzato per selezionare ogni oggetto del device
150     {
151         currDev = idxDev;
152         currProp = idxProp;
153         currIdIo = deviceList[idxDev].object_prop[idxProp].id_io;
154         type_obj = deviceList[currDev].object_prop[currProp].type;
155         if (type_obj == 1 || type_obj == 2 || type_obj == 4 || type_obj == 5)
156         {
157             selectOutputValue(currIdIo,
158                               type_obj,
159                               deviceList[currDev].object_prop[currProp].instance,
160                               deviceList[currDev].address,
161             function(value, type, instance, address)
162             {
163                 deviceList[currDev].object_prop[currProp].presentValue = value;
164                 mainBacnet.writeValue(
165                                       type,
166                                       instance,
167                                       address,
168                                       value,
169                                       )
170             });
171         }
172     }
173   }
174 }
175
176 // Funzione del dbBacnet.js esportate:
177
178 exports.startPollingOutput = startPollingOutput;   // Esporto la funzione di attivazione del Polling in scrittura
179 exports.startPollingInput = startPollingInput;     // Esporto la funzione di attivazione del Polling in lettura
180 exports.addDevice = addDevice;                     // Esporto la funzione di aggiunta del device BACnet
181 exports.addPropertyIO = addPropertyIO;             // Esporto la funzione di aggiunta dell'oggetto del device BACnet
182
```

mainBacnet.js

```
// Valutazione delle risposte del Server Bacnet e indirizzamento alle varie funzioni di scrittura nel Database
// e lettura e scrittura dei valori del device:

//Costanti
const bacnet = require('bacstack');                 // connessione alla libreria BACnet
const client = new bacnet({apduTimeout: 6000 });

// Variabili
var dbBacnet = require('./dbBacnet.js');          //  // connessione al dbBacnet.js ( permette l'utilizzo delle funzioni esportate da esso)
var  value;
var whoIsTimer = setInterval(myTimer, 6000);       // Intervallo di tempo con cui viene richiamata la funzione di Who-Is
var read = [];                                      // Array su cui vengono salvati i dati di ritorno della funzione ReadPropertyMultiple
var id_device = 1;                                  // valore univoco assegnato ad ogni device
var id_bacnet_io = 1;                               // valore univoco assegnato ad ogni oggetto del device

console.Log(' +++  Bacnet Start  +++ ');

dbBacnet.startPollingInput();    // avvio la funzione di aggiornamento del present_value di ogni object nel database;
dbBacnet.startPollingOutput();   // avvio la funzione di aggiornamento del present_value di ogni object nel device;

// Ogni volta che ricevo I-Am da un device nella rete, vado a salvare le proprietà del device e dei suoi oggetti
// nel database, mi sottoscrivo al "change of value", e infine mi rimetto in ascolto con un nuovo Who-is;

client.on('iAm', function(device)
{
  var currentIdDevice;
  var deviceAddressArray = [];
  currentIdDevice = id_device;
  var struct_device= {device, currentIdDevice};
  currentIdDevice = id_device;
  console.log(' +++  Device Identifier  +++ ');
  console.log(' +++  Trovato device numero:', id_device,
              'Address:', device.address,
              'DeviceId:', device.deviceId,
              'MaxAdpu:', device.maxApdu,
              'Segmentation:', device.segmentation,
              'VendorID:', device.vendorId,'  +++ ');
  clearInterval(whoIsTimer);
  dbBacnet.addDevice(struct_device);
  deviceAddressArray.push({ address: device.address, deviceId: device.deviceId });
  client.readProperty(device.address, {type:29, instance:0}, 211, {}, function(err, value)
  {
    readObjList(err, value, device.address, currentIdDevice);     // leggo la lista degli oggetti presenti nel device
  });
  id_device++;
});

// Funzione che permette di eseguire la scansione della rete ogni secondo
function myTimer()
{
  console.log(' +++  Who-Is sent  +++ ');
  client.whoIs();
}

// Funzione che mi carica i campi del database inerente agli oggetti differenziandoli tra binary e analog
function fillDbIoProp (err,value, id_dev)
{
  var object = {};
  var struct_object = {};
  object.id_bacnet_io = id_bacnet_io;
  object.id_device = id_dev;
  object.object_Identifier =  value.values[0].values[0].value[0].value.instance;
  object.object_Name = value.values[0].values[1].value[0].value;
  object.object_Type = value.values[0].values[2].value[0].value;

  // vado a valutare il tipo, se è binario vado a settare i campi non interessati dal
  //tipo con 0 e undefined ( per l'unità di misura);
  if (object.object_Type == 3 || object.object_Type == 4 || object.object_Type ==5)
  {
    object.present_Value = value.values[0].values[3].value[0].value;
    object.event_State = value.values[0].values[4].value[0].value;
    object.out_of_Service = value.values[0].values[5].value[0].value;
    object.update_Interval = 0;
    object.units = 'not defined';
    object.min_Pres_Value = 0;
    object.max_Pres_Value = 0;
    object.resolution = 0;
    object.cov_Increment = 0;
    object.high_Limit = 0;
    object.low_Limit = 0;
  }
  else  // altrimenti vado a settare i seguenti campi:
  {
    object.present_Value = value.values[0].values[3].value[0].value;
    object.event_State = value.values[0].values[4].value[0].value;
    object.out_of_Service = value.values[0].values[5].value[0].value;
    object.update_Interval = value.values[0].values[6].value[0].value;
    object.units = value.values[0].values[7].value[0].value;
    object.min_Pres_Value = value.values[0].values[8].value[0].value;
    object.max_Pres_Value = value.values[0].values[9].value[0].value;
    object.resolution = value.values[0].values[10].value[0].value;
    object.cov_Increment = value.values[0].values[11].value[0].value;
    object.high_Limit = value.values[0].values[12].value[0].value;
    object.low_Limit = value.values[0].values[13].value[0].value;
  }
  console.log(' +++  Aggiunto oggetto ', object.object_Name,
              ' del device numero ', id_dev,'  +++ ');

  struct_object ={object, id_dev};
  dbBacnet.addPropertyIO(struct_object);
  id_bacnet_io++;
}
```

```
104
105 // Callback che permette di aggiungere tutti gli oggetti del device una volta ottenuta la object_List:
106 function readObjList(err, objList, device_address, id_dev)
107 {
108   for (var i=0; i<objList.values.length; i++)
109    {  // crea una ricorsione con n cicli quanti sono object_list del device        [{id: 8},{id:9},{id:10},{id:11},{id:12}]
110     readPropertyMultipleBacnet(device_address,
111                                objList.values[i].value.value.type,
112                                objList.values[i].value.value.instance,
113                                [{id:75}, {id:77}, {id:79}, {id:85}, {id:36}, {id:81}, {id:118}, {id:117}, {id:69}, {id:65}, {id:106},
114                                 {id:22}, {id:45}, {id:59}],
115                                function(err,value)
116                                {
117                                  fillDbIoProp(err, value, id_dev);
118                                });
119   }
120 }
121
122 // Funzione che legge una proprietà del oggetto del device BACnet
123 function readPropertyBacnet(address, typeid, instance, property, callback)
124 {
125     value = client.readProperty(address, {type:typeid, instance:instance}, property, {}, callback);
126 }
127
128 // funzione che legge più proprietà di un oggetto del device BACnet
129 function readPropertyMultipleBacnet(address, typeid, instance, list_properties, next)
130 {
131   var requestArray = [
132       {objectId: {type: typeid, instance: instance}, properties: list_properties}
133       ];
134   read = client.readPropertyMultiple( address, requestArray, next);
135   return read;
136 }
137
138 // Funzione che permette di modificare una proprietà dell'oggetto del device BACnet
139 function writePropertyBacnet(address, typeid, instance, property, value, callback)
140 {
141   client.writeProperty(address, {type:typeid, instance:instance}, property, [{type: 9, value: value}], {}, callback);
142 }
143
144 // Funzione che permette di modificare più proprietà dell'oggetto del device BACnet
145 function writePropertyMultipleBacnet(address, objectIdtype, objectIdinstance, propertyid, propertyindex, valuetype, value, priority)
146 {
147   client.writePropertyMultiple(address,[{objectId: {objectIdtype, objectIdinstance},
148       values: [{property: {propertyid, propertyindex }, value: [{valuetype, value}],
149       priority}]}],
150       function(err, value){
151   });
152 }
153
154 // Funzione che mi permette di leggere il present_Value di un oggetto:
155 function readValue(type, propInstance, address, callback)
156 {
157     // Lettura del Present Value dall'oggetto
158     readPropertyBacnet(address, type, propInstance, 85, function(err, value){
159       if (callback != 0)
160       {
161         callback(value.values[0].value);
162       }
163     });
164 }
165
166 // funzione che mi permette di scrivere il present_Value di un oggetto:
167 function writeValue(type, propInstance, address, value)
168 {
169   // Scrittura del Present Value dell'oggetto
170   writePropertyBacnet(address, type, propInstance, 85, value, function(err,valueWrite){
171   });
172 }
173
174 // Funzioni del mainBacnet.js esportate:
175
176 exports.readValue = readValue;    // esporto la funzione di lettura del present Value
177 exports.writeValue = writeValue;  // esporto la funzione di scrittuta del present Value
178
```

10) Bibliografia

[1] Arzen K - E. *A simple event-based PID controller*. In: Procedings of the 14th World Congress of the International Federation of Automatic Control (IFAC), Beijing, P.R. China, 1999.

[2] Åström K J, Hägglund T. PID *Controllers: Theory, Design, and Tuning*, Second edition. Instrument Society of America, Research Triangle Park, NC, 1995.

[3] Bolzern P, Scattolini R, Schiavoni N. *Fondamenti di controlli automatici*, IV edizione. Mc Graw-Hill Education, Milano, 2015.

[4] Cipollone E, Cipollone F. *Progettazione e conduzione di reti di computer-Fondamenti di switching*. EPC Editore, Roma, 2012.

[5] Douglas E C. *Internetworking con TCP/IP-Principi, protocolli e architetture*, Volume I-Quinta edizione. Pearson Education Italia, Milano, 2006.

[6] Upton E Halfacree G. *Raspberry Pi- La guida Ufficiale*. Hoepli Informatica, Milano, 2018.

[7] Veronesi M. *Regolazione PID-Tecniche di taratura, schemi di controllo, valutazione delle prestazioni*. Franco Angeli Edizioni, Milano, 2011.

11) Ringraziamenti

Giunti alla fine di questo lavoro desidero ringraziare ed esprimere la mia riconoscenza nei confronti di tutte le persone che, in modi diversi, sono state con me durante questo percorso.
Ringrazio chi mi ha incoraggiato e motivato nei momenti più difficili nel corso dei miei studi, specialmente nell'ultimo periodo, rendendo piacevoli e divertenti questi ultimi due anni universitari.
Le mie gratitudini vanno al Relatore, Prof. Giuseppe Conte, per i suoi preziosi consigli e per avermi fornito tutti gli strumenti di cui avevo bisogno per intraprendere la strada giusta e portare a compimento la mia tesi.
Un grazie all'Ing. Marco De Vecchis, e all'azienda Bridger s.r.l., che mi hanno permesso di realizzare questo importante lavoro su una tecnologia altamente innovativa. La disponibilità ricevuta nel poter accedere al materiale tecnico necessario è stata essenziale al fine di poter portare a compimento questo lavoro.
Un sentito ringraziamento va a mia madre Giuliana e ai miei 2 fratelli, Sante ed Ettore, che sono il mio punto di riferimento e che mi hanno sostenuto sia economicamente che emotivamente e che mi hanno permesso di percorrere e concludere questo cammino.
Inoltre, vorrei ringraziare mio zio Clemente, per avermi aiutato nella correzione di questo elaborato, e mio cugino Felice, per avermi accompagnato durante questi anni universitari con i suoi consigli.
Infine, le mie gratitudini vanno, agli amici di "vecchia data" sui quali posso contare ogni volta, agli amici conosciuti lungo questo percorso con cui ho passato delle intense e belle giornate, ai miei compagni di corso che mi hanno spesso rallegrato ed insegnato che molte volte è proprio l'unione a fare la forza, ed infine ai miei coinquilini con i quali ho condiviso moltissime esperienze che porterò sempre con me.
Un pensiero particolare va a mio Padre, ai miei Nonni e ai miei Zii che oggi non possono essere qui, ma che porto sempre nel mio cuore.

Con affetto e ringraziamento,
Luca.