Sistemi Embedded

Sistemi (a microcontrollore) di elaborazione atti a svolgere una specifica funzione o compito (special-purpose), non ri-programmabili e incorporati in un sistema complesso.

Le loro caratteristiche principali:

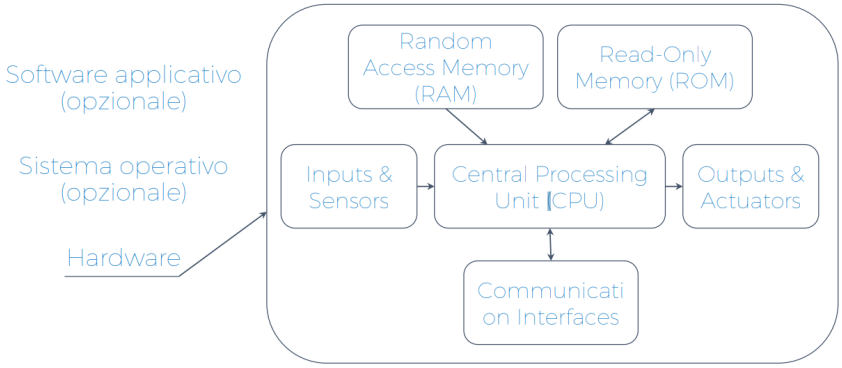
- Funzionalità specifica: eseguono sempre le stesse cose ripetutamente;
- Tightly constrained ed efficienza: memoria e CPU limitate;
- Uso di design metrics: dimensioni, performance e consumo di energie;
- Progettati per essere efficienti: efficienza energetica, di code-size, di run-time, di peso e di costo;
- Affidabilità;
- Reattivi e/o real time: permettono di avere un tempo di risposta basso o quasi nullo.

CPS - Cyber Physical System

Sono un'integrazione di diversa natura con lo scopo di controllare un processo fisico e il suo adattamento in tempo reale attraverso feedback.

Architettura sistema embedded

Non c'è ne una di riferimento, ma esistono molteplici piattaforme, una struttura di base potrebbe essere:



CPU

Possono essere 3:

- General-purpose:architettura e insieme di istruzioni predefinito, comportamento definito dal programma in esecuzione;
- Application-specific processor (ASIP): programmabili e ottimizzati per classi di applicazioni;

- Single-purpose: progettati per implementare una specifica funzionalità o programma.

Ma quale scegliere fra un processore GPP (general purpose) o ASP (application specific)?

I primi sono CPU di base che tramite l'utilizzo delle architetture di Von Neumann e Harvard permettono l'uso di software personalizzato per una specifica applicazione.

I secondi sono sistemi che danno soluzioni a specifiche applicazioni con un numero limitato di funzioni che non giustificano architetture più complesse.

MCU

Dispositivo integrato su un singolo circuito elettronico, evoluzione al microprocessore ed utilizzato nei sistemi embedded.

Composto da un processore, una memoria permanente e volatile, canali di I/O, gestore interrupt e blocchi specializzati.

Alcuni esempi possono essere gli Arduino dove si programma esternamente e si passa il file già buildato pronto all'esecuzione.

SOC e Single-Board PC

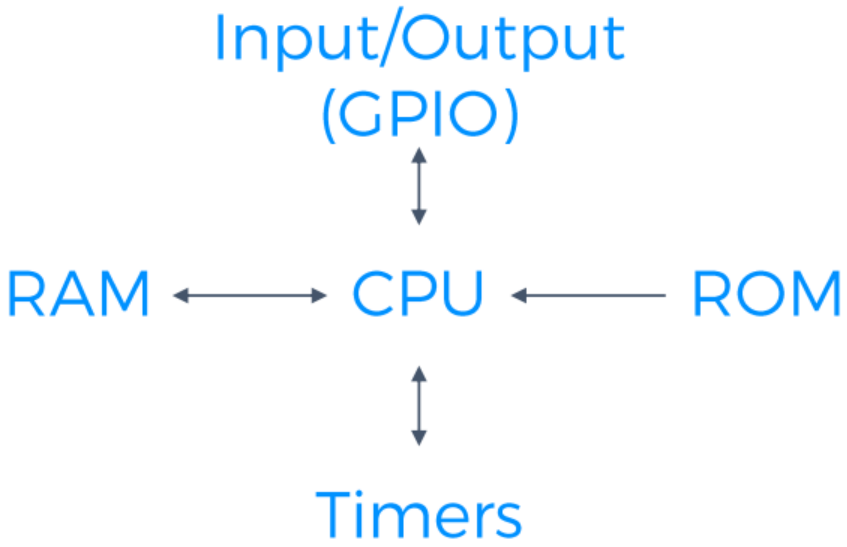
Sta per System on a chip, un unico chip incorpora un sistema completo.

Microcontrollori

Programmabile tramite sistema esterno, viene creato e buildato il file poi passato al microcontrollore con USB o seriale.

I microcontrollori non hanno S.O. quindi tutto ciò passato viene eseguito dal processore.

Elementi base



Esecuzione sul controllore

La CPU funziona tramite il ciclo fetch-decode-execute, nei microcontrollori si usa l'architettura Harvard che al contrario di quella Von Neumann memorizza istruzioni e dati in memorie separate (codice e dati letti/scritti insieme).

Il codice viene eseguito tramite super loop, che ha un inizializzazione e un ciclo infinito che esegue un task ripetutamente. Questo approccio semplice la rende perfetta per lo sviluppo di applicazioni ma non permette una gestione della temporizzazione molto accurata.

GPIO (General Purpose I/O)

I microcontrollori normalmente dispongono di una serie di pin, normalmente general purpose (programmabili per fare input o output).

I pin sono digitali (valore 0 o 1) o analogici (assumono qualsiasi valore).

PIN

Per i pin I/O occorre sempre prestare attenzione ai parametri di funzionamento, cioè tensione (volt) da applicare o emessa in uscita e la corrente (Ampere) che può ricevere o fornire.

Per evitare il floating (segnale input senza collegamento, cioè senza tensione che li porta a captare qualsiasi disturbo) è possibile equipaggiare i pin con circuiti pull-up.

I pin PWM emulano in uscita un segnale analogico a partire da uno digitale, possibile grazie alla modulazione del duty cycle (% di tempo che il segnale è ad 1 rispetto a quella in cui è 0) di un'onda quadra.

I pin IRQ ricevono segnali di interruzione permettendo al microcontrollore di reagire ad eventi e eseguire istruzioni che non sono direttamente nel loop ma in routine separate.

All'arrivo di una richiesta si sospende l'esecuzione delle istruzioni, si salva nello stack, restituisce il controllo alla routine di interrupt (chiamata a interrupt handler o interrupt service routine).

Finito l'interrupt si riprende dal punto di sospensione.

Si possono disabilitare le interruzioni, in questo modo siamo sicuri che certi parti di codice verranno eseguite senza intoppi, le interrupt request verranno eseguite quando le interruzioni saranno riabilitate (se ci dimentichiamo di riabilitarle il sistema potrebbe risentirne); inoltre le interrupt possono interrompersi fra loro (interrupt nesting).

Se occorre elaborare un segnale analogico bisogna trasformarlo in digitale tramite l'ADC che mappa il valore continuo in un valore discreto nel range 0-1023. Un segnale discreto può essere codificato.

Timers

Per realizzare un comportamento timer-oriented i timer diventano essenziali per misurare gli intervalli di tempo, gestire i PWM e realizzare timeout.

Alcuni timer sono:

- Watch dog timer: utilizzato per resettare il circuito in situazioni di blocco;
- Power management: permette di utilizzare il circuito con diversi livelli di risparmio energetico.

Protocolli di comunicazione e bus

Le interfacce di tipo seriale e parallelo servono per evitare lo scambio di dati mediante singoli pin, le parallele inviano dati parallelamente con l'utilizzo di diversi fili invece le seriali inviano dati sequenzialmente.

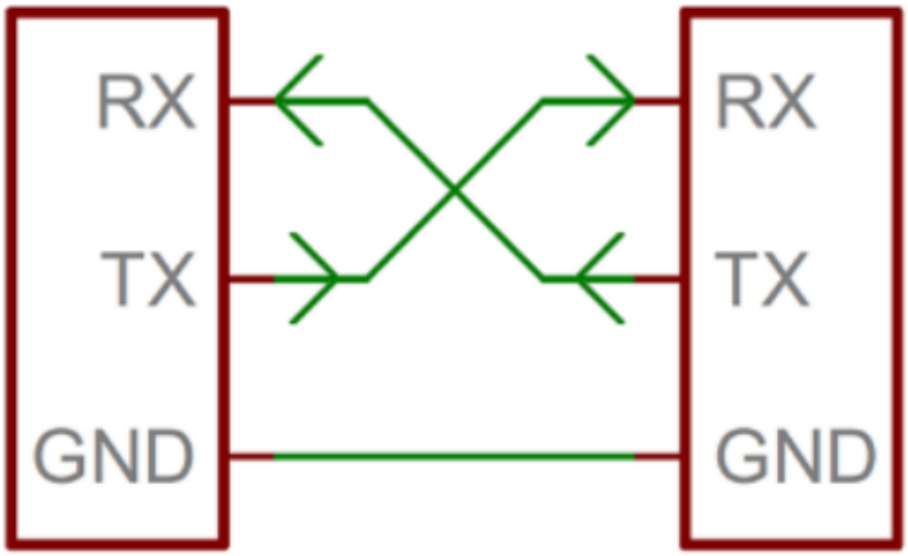
Seriali

Ne esistono di due tipologie:

- Sincroni: c'è un clock, aumentano velocità e complessità (I^2C e SPI);
- Asincroni: non c'è clock,2 linee (trasmissione e ricezione) (USB, RS-485, TTL); i parametri necessari per gestire la comunicazione basata su un protocollo prestabilito sono:

- baudRate: velocità della trasmissione (bps), più è alta più possibili errori di trasmissione;
- dataFrame: struttura del pacchetto;
- syncBit: identifica eventuali bit di start e stop del pacchetto dati;
- parityBit: serve per error checking (opzionale);

Normalmente i bus seriali sono cablati con due linee separate con un ricevitore (RX) e un trasmettitore (TX) collegati in maniera incrociata tra due device:



Se entrambi i dispositivi possono inviare e ricevere simultaneamente si ha una comunicazione FULL-DUPLEX, se comunicano a turno HALF-DUPLEX. Esiste anche la Serial Enable LCD, una comunicazione seriale basata su singolo cavo.

L'UART è il blocco circuitale in grado di convertire i dati da e verso l'interfaccia seriale mettendola in comunicazione con l'interfaccia parallela. È asincrono, si appoggia a dispositivi separati per gestire i flussi di dati e utilizza un bus seriale.

Seriali sincroni Abbinando un clock ad un'interfaccia seriale la si rende sincrona, portando ad avere trasmissioni più rapide (ES: protocolli SPI e I^2C).

I^2C è nato in Philips, veloce e robusto a due vie che utilizza l'indirizzamento a 7 o 10 bit e si può arrivare a 3.4Mbit/s. Ha un'architettura master-slave e più device condividono la SCL (Serial Clock Line) e SDA (Serial Data Line).

SPI è di Motorola ed è seriale full-duplex, basato su master-slave che utilizza due linee MOSI (Master Out Slave In) per trasmettere e MISO (Master In Slave Out) per ricevere; la linea Slave Select è riservata per la selezione dello slave.

Grazie al Segnale di clock condiviso (SCLK) è sincrono.

In sostanza:

- SPI: + veloce, + semplice, nessuna richiesta di pull-up;
- I^2C: solo 2 linee, indirizzamenti + semplici e aperti.

Sistemi SOC e RTOS

I SOC (System On a Chip) sono circuiti integrati in cui un unico chip contiene un intero sistema operativo (CPU, Chipset, COntroller, Sistema Video, I/O, Generatore di Clock e regolatori di tensione).

Nei SOC si installano S.O. detti RTOS (Real-Time Operating System) che permettono la configurazione di sistemi reattivi e completi.

Sistemi Operativi

Nei S.O. moderni hardware e software comunicano attraverso moduli che si occupano di diverse funzionalità (gestione di: processi, memoria, rete, ecc), fornendo interfacce che permettono lo scambio di dati e nascondendo dettagli implementativi tramite l'astrazione.

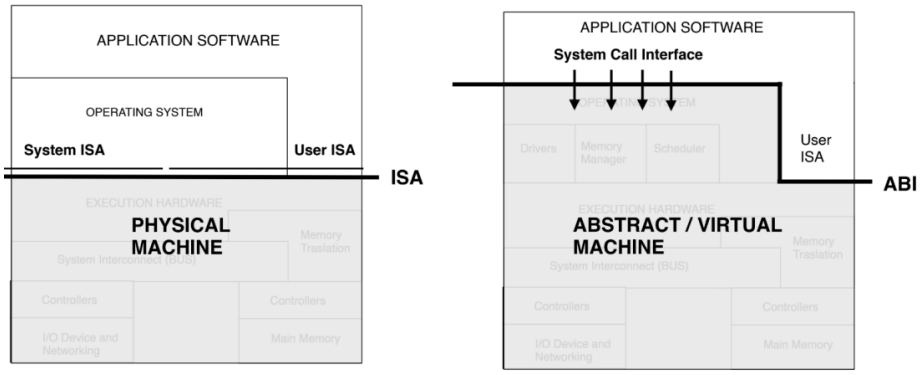
I S.O sono software di base con lo scopo di gestire hardware e software ponendosi come mediatore tramite l'utilizzo di API chiamate chiamate di sistema, eseguendo e gestendo i programmi e rendendo più facile ed efficace l'utilizzo delle risorse attuando algoritmi di scheduling che gestiscono la concorrenza dei programmi, garantendo sicurezza e protezione.

La vera struttura hardware viene nascosta al software che vede solo la macchina virtuale con una vista top-down che aumenta la portabilità dei programmi e semplificando la programmazione delle applicazioni che potranno essere eseguite su macchine fisiche differenti (se la macchina virtuale fornisce la stessa interfaccia).

Interfacce

Ne esistono due principali:

- ISA: Istruction Set Architecture, separa i livelli HW e SW, divisa in due parti: user ISA che si occupa delle funzionalità visibili ai programmi utente e system ISA che si occupa delle funzionalità del S.O;
- ABI: Application Binary Interface permette l'accesso alle risorse HW e ai servizi del sistema ai programmi tramite user ISA e System Call Interface fornite dal S.O.



Per l'esecuzione dei programmi il S.O. può adottare due costrutti:

- Multiprogrammazione quindi il caricamento in memoria centrale di più programmi e gestendo concorrentemente l'accesso all'utilizzo della CPU quando i programmi utilizzano risorse differenti.
- Multi-tasking permette di eseguire programmi concorrentemente senza che questi debbano accedere ad altre risorse grazie le strategie di schedulazione; se due task cercano di accedere alla stessa risorsa ho una dipendenza relativa ai dati.

RTOS

S.O. real time creati apposta per i sistemi embedded che offrono compattezza, estrema efficienza nelle gestione risorse e affidabilità.

Non offrono la multi-programmazione e l'esecuzione si limita ad una sola applicazione ma permette l'esecuzione multi-task.

Questi S.O. devono reagire e rispondere entro determinati range temporali prestabiliti e rispettare deadline, se le deadline sono sempre rispettare si dicono Hard real-time e si utilizzano nei sistemi safety-critical. Se sono ammessi casi in cui le deadline possono non essere rispettate si dicono Soft real-time e si utilizzano nell'elettronica di consumo.

Nei sistemi real time tutto deve essere predicibile, come il tempo impiegato da un task, il tempo max per un'azione e la costanza nella qualità dei cicli richiesti per un'operazione.

Le interruzioni sono comunque disponibili.

Responsiveness e Overhead

Al contrario dei microcontrollori, che prediligono un loop con polling continuo, per verifiche e esecuzioni di funzioni nei RTOS si sfruttano i context switch che sono più predicibili e non aggiungono overhead al dispositivo con eventuale calo di prestazione.

Gestione risorse

Per arbitrare e gestire l'accesso alle risorse vengono messi a disposizione meccanismi centralizzati, come:

- Allocazione/deallocazione della memoria;
- Semafori e mutex;
- Meccanismi per gestire le sezioni critiche ei suoi problemi.

Semplificazione

Gli RTOS offrono servizi di astrazione per l'HW e l'utilizzo di framework e piattaforme per gestire prodotti di terze parti; questo approccio migliora il lavoro di team e un'agevolazione del debug con la modulazione del software garantendo manutenibilità ed estensibilità

I RTOS non sono congeniali per applicazioni semplici perchè basterebbe un'architettura a looping o polling.

Sensoristica

I sensori sono dispositivi trasduttori che misurano un certo fenomeno fisico o concentrazione chimica fornendo scala o intervallo; sono sia digitali che analogici.

In sostanza misurano tutto ciò che "viene da fuori".

Il suo scopo principale infatti è quello di misurare, cioè confrontare due quantità omogenee, stabilendo in che rapporto una quantità incognita stia rispetto ad un'altra che opera come riferimento.

Gli attuatori producono un effetto sull'ambiente (un led per esempio) e sono sia analogici che digitali.

Entrambi forniscono segnali analogici quindi continui ad risoluzione infinita che rappresentano la grandezza fisica continua oppure codificati in bit o logici (booleani) che rappresentano la grandezza discreta.

Il segnale analogico per essere elaborato da un calcolatore deve essere trasformato in digitale tramite un convertitore ADC, che campiona il segnale (vengono presi dei campioni ad intervalli regolari di tempo) con la quantizzazione invece si approssimizza il valore campionato al più vicino digitale, può produrre un errore nella misurazione.

- Strumento di misura: dispositivo impiegato per eseguire misurazioni, solo o con strumenti di supporto
- Sistema di misura: insieme di uno o più strumenti di misura ed eventualmente altri dispositivi
- Trasduttore: dispositivo, impiegato in una misurazione, che fornisce un valore in uscita avente una relazione specificata con quello in ingresso
- Sensore: elemento di un sistema di misura che è direttamente influenzato dal fenomeno
- Catena di misura: elementi che costituiscono un percorso univoco per il segnale dal sensore all'elemento di uscita

Misure e incertezze

Ogni misura ha errori e/o incertezze, dove l'incertezza esprime la dispersione dei valori che verificatasi in misurazioni ripetute si trasforma in un errore.

In una misura il risultato è un numero ed una incertezza preceduta dal simbolo $\pm$ e con un'unità di misura.

Un errore può essere:

- Sistematico: sempre stessa influenza;
- Casuale:influenza che cambia in grandezza e segno;
- Grossolani: riguardano lo strumento o l'operatore.

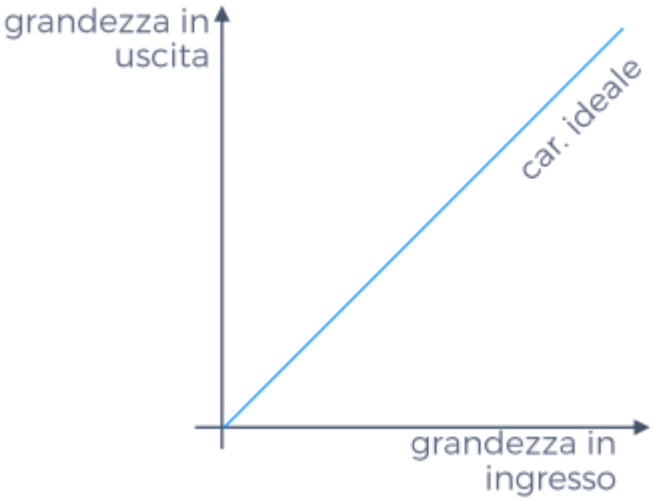
Se non vengono monitorate le condizioni ambientali possono sorgere errori casuali, diversamente sono sistematici.

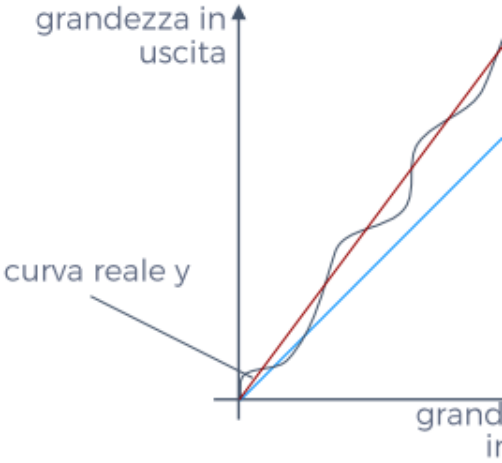
Caratteristiche dei sensori

I sistemi di misura possono essere descritti tramite diverse caratteristiche divisibili in statiche e dinamiche:

- Statiche: il tempo non influisce sullo strumento, definite dalla funzione $Y = F(x)$, con X segnale in ingresso e Y in uscita.

Sono valutate in una situazione di normale funzionamento, dove la caratteristica ideale è una retta con pendenza unitaria, ma nella realtà il comportamento non è ideale (per via di imperfezioni costruttive) e produce una deviazione di uscita rispetto al "vero", questa differenza è l'errore del sensore.

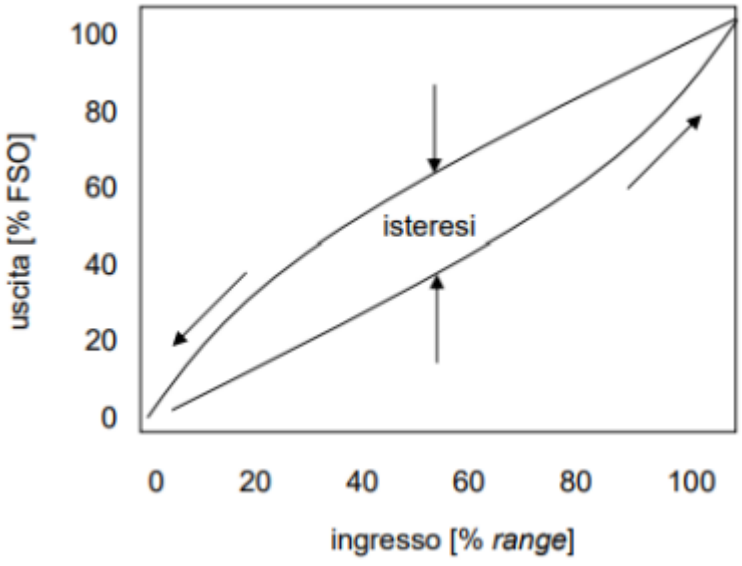




Per considerare l'errore bisogna definire la fascia di incertezza che rappresenta la massima deviazione della sua retta di riferimento.

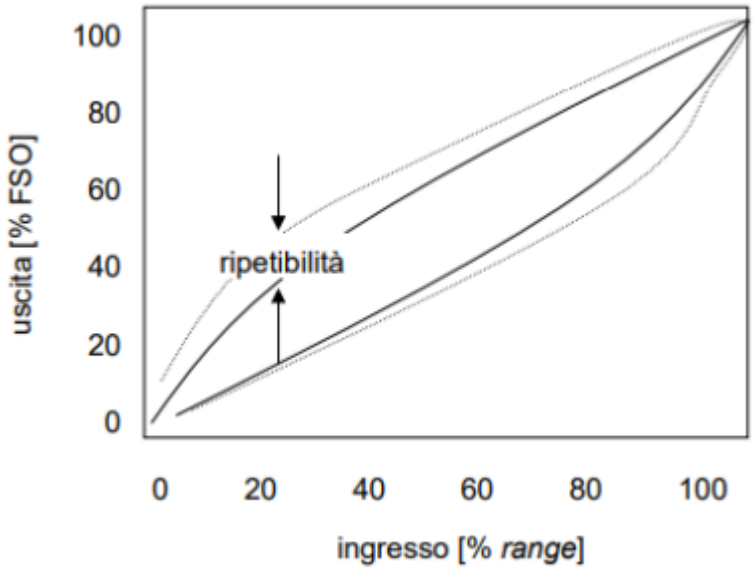
La scomposizione dell'errore nelle sue componenti serve per correggere a posteriori la misura, altrimenti con la calibrazione o taratura andiamo a mitigare l'errore sul nascere.

Isteresi cioè la differenza massima tra i valori di uscita corrispondente ad uno stesso ingresso, ottenuto prima per valori crescenti, poi decrescenti.

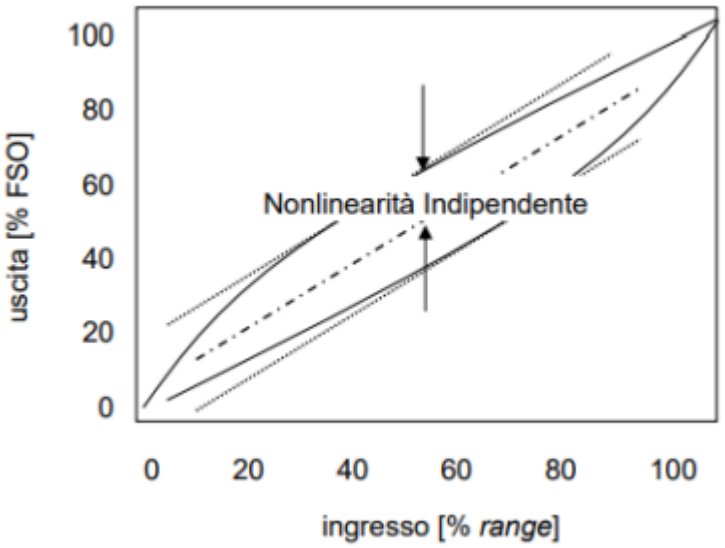


Ripetibilità è la capacità di riprodurre la stessa uscita con lo stesso ingresso

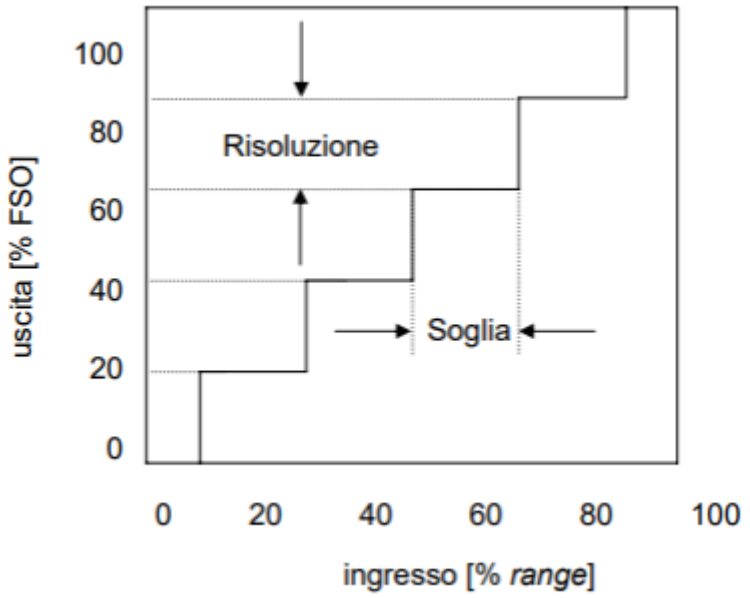
consecutivamente e nelle stesse condizioni.



Linearità, lo scostamento della curva di taratura della retta di riferimento, misurata agli estremi è detta terminale, a metà si dice indipendente e ai minimi quadrati.

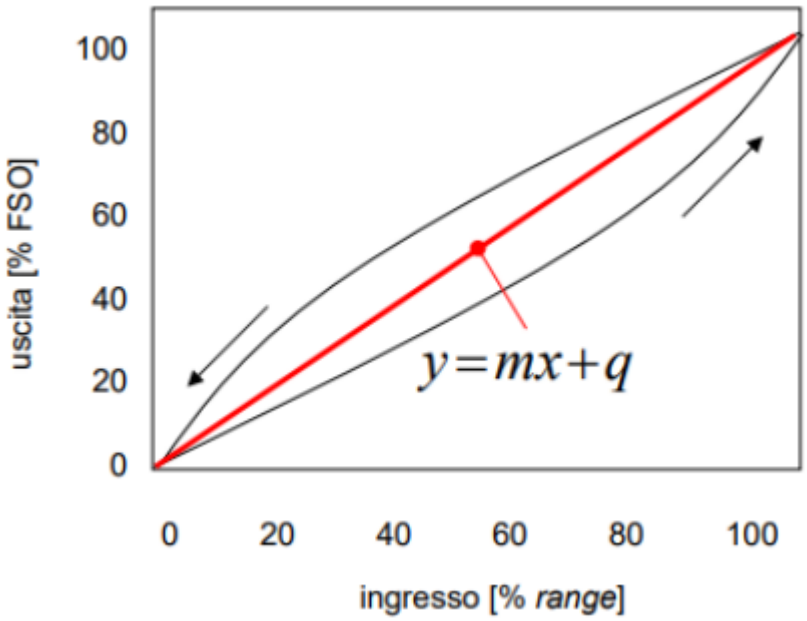


Risoluzione: l'ampiezza del passo delle uscite (distanza fra due uscite consecutive) al variare dell'ingresso in tutto il suo range.

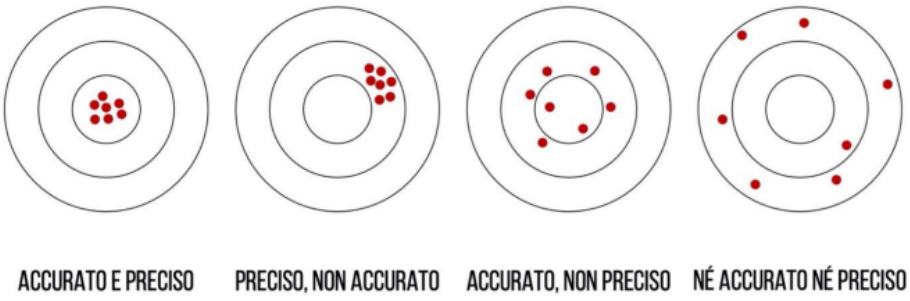


Sensibilità o guadagno è la proporzionalità tra valore di ingresso e valore di uscita, cioè la sensibilità in uscita di un sensore, rappresenta la pendenza m della retta.

Offset rappresenta il segnale in uscita anche in assenza del segnale di ingresso, è il termine noto Q della retta.



Accuratezza è la misura di quanto il valore letto dal sensore si discordi dal valore corretto, la Precisione descrive quanto un sensore sia soggetto o meno ad errori accidentali (legata alla ripetibilità).



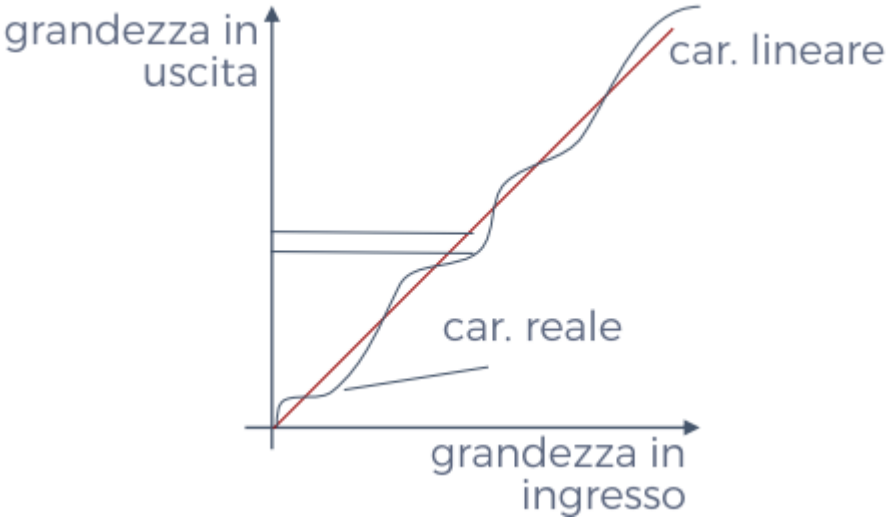
- Dinamiche: il tempo influisce ed è un fattore cruciale, alcune caratteristiche sono: tempo di risposta cioè il tempo necessario per l'uscita di raggiungere una percentuale specifica del valore finale, tempo di salita necessario affinché l'uscita vada da prefissato valore ad uno maggiore e costante di tempo necessaria affinché l'uscita raggiunga il 63% del valore finale.

Il guadagno (K) del trasduttore è la costante di proporzionalità fra i valori in ingresso e quelli di uscita prendendo in esame la caratteristica statica che idealmente deve essere lineare.

I trasduttori commerciali hanno una caratteristica statica reale che è leggermente diversa da quella ideale, questo a causa di imperfezioni costruttive. La qualità del sensore si misura in base a quanto la caratteristica reale si discosta da quella ideale.

Per i trasduttori lineari la relazione tra la grandezza fisica misurata e il segnale in uscita è una relazione lineare del tipo: $Y = KX$ (dove K è il guadagno).

L'errore di linearità è la massima deviazione dell'uscita del trasduttore rispetto alla caratteristica lineare che approssima al meglio quella reale. La caratteristica lineare viene normalmente ottenuta secondo il metodo dei minimi quadrati.



Errori di...

Offset o errore di fuori zero è il valore che assume l'uscita del trasduttore quando la grandezza da misurare è nulla.

Soglia corrisponde al più basso livello di segnale rilevabile dal sensore. Non è necessariamente coincidente con la grandezza da misurare.

Guadagno descrizione del comportamento dello strumento, fatta in relazione al tempo, ovvero si descrive il modo di funzionamento nel caso in cui l'ingresso sia tempovariante (tempo di risposta).

Tipi di sensori

Prossimità

Rileva la presenza di oggetti nelle vicinanze della parte sensibile del sensore (induttiva, magnetica, ottica, ecc) nel raggio della sua portata nominale (distanza entro la quale il sensore rileva oggetti).

//La parte di sensori leggila nella slides 4 dalla diapositiva 22 a 46

Attuatori

Dispositivo che converte dell'energia da una forma ad un'altra, l'interfacciamento avviene o con la corrente del GPIO o con corrente maggiore.

TIpi

LED: dispositivi optoelettronici che sfruttano alcuni materiali semiconduttori in grado di produrre fotoni attraverso un fenomeno di emissione spontanea; questo

li rende altamente efficienti e particolarmente affidabili.

Display LCD: pannelli formati da due strati di vetro tra i quali è racchiuso un liquido e numerosi contatti elettrici in grado di applicare un campo elettrico al liquido, comandando una piccola porzione di liquido che identifica un pixel.

Motori elettrici: all'interno uno statore ed un rotore che, producendo un campo magnetico, generano il movimento del rotore. Vi sono motori di diverso tipo, i più comuni nei sistemi embedded sono:

- Corrente continua: rotazione continua ad elevata velocità o coppia, controllo della velocità di rotazione. La corrente attraversa un avvolgimento nel rotore che genera il campo magnetico;
- Passo-Passo (Stepper): motore brushless che suddivide la rotazione in molti step, la posizione può quindi essere controllata accuratamente ed hanno un'escursione di 360 gradi;
- Servo motori: come per il precedente è possibile pilotare l'angolo del rotore ma la posizione è assoluta e non relativa alla posizione corrente, escursione di 180 gradi.

Pilotaggio di circuiti esterni

Il Relè permette di aprire/chiudere il secondo circuito mediante l'azione di un elettro-magnete pilotato dal primo circuito.

Possono essere:

- NC (normalmente chiusi);
- NO (normalmente aperti) se senza tensioni in ingresso sulla bobina il contatto in uscita risulta "spento",

Pilotati con diverse tensioni in ingresso e possono gestire tensioni in uscita.

I fotoaccoppiatori invece si usano principalmente per trasferire un segnale da un apparato ad un altro tenendoli isolati elettricamente. Un led interno si attiva, pilotato da un circuito a bassa tensione, e attiva una fotocellula, chiudendo l'interruttore del secondo circuito.

Partitore di tensione

Permette di produrre in uscita una tensione voluta a partire da una tensione in ingresso più alta

Viene usato per evitare problemi sui livelli di tensione differenti che sensori e circuiti hanno, in modo da evitare danni.

Resistenze di pull-up/down

Sono molto comuni per sopperire all'alta impedenza dei pin dei device che li può portare ad uno stato di «Floating ».

Si inserisce quindi una piccola resistenza o tra il pin e Vdd (resistenza di Pull-Up) o tra il pin e Vss (GND o V0) (resistenza di Pull-Down).

Architetture software

La progettazione di un sistema embedded è differente da quella dei sistemi general purpose, infatti vi sono tre aspetti fondamentali fortemente connessi fra loro:

- Architettura;
- Applicazione [requisiti funzionali e non];
- Metodologie di sviluppo.

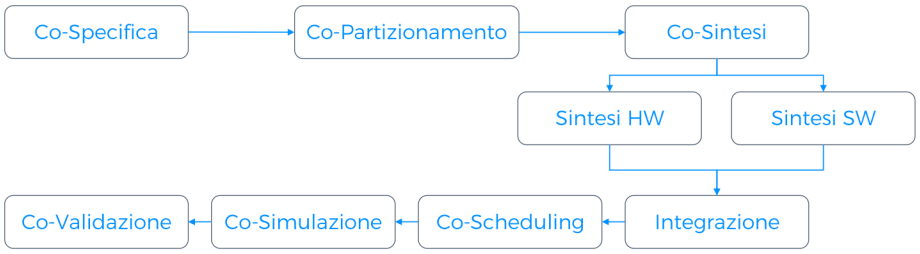
Questo tipo di sviluppo è molto complesso e per raggiungere un buon prodotto occorrono analisi, simulazioni alternate con l'uso di prototipi dove la simulazione non fornisce valori accettabili e strumenti avanzati per il testing dell'hardware.

Inoltre si consiglia l'utilizzo di una metodologia ibrida, in grado di integrare metodologie di sviluppo software e hardware.

Metodologie

Co-design

Utilizzata quando sono presenti componenti hardware e software insieme, mira a ottimizzare il processo di progetto andando ad aumentare la produttività e accorciando i tempi di sviluppo con il riuso dei componenti.



Co-Specifica: si analizza e si fa una specifica insieme;

Co-Partizionamento: divisione componenti tra hardware e software, un buon compromesso mitiga costi e ottimizza le prestazioni;

Co-Sintesi: sviluppo componenti, diviso in 4 fasi:

- comunicazione: interfacciamento tra HW e SW;
- personalizzazione delle specifiche: metodi di comunicazione per permettere l'interfacciamento;
- specifiche HW;
- specifiche SW;

Co-Scheduling: assegnamento timing di esecuzione per ridurre la latenza al minimo;

Co-Simulazione: verifica funzionalità hardware e software;

Co-validation: verifica del sistema e diversi livelli di astrazione.

Modello a cascata

Modello classico e vecchio stile, dove ogni fase genera un output necessario per quella successiva. Gestione semplice per progetti piccoli ma pericoloso per progetti più complicati e lunghi rendendo impossibile effettuare modifiche ai livelli superiori.

Modello a spirale

In realtà è un metamodello creato da Barry Boehm e rappresenta diversi cicli di vita, si basa su quattro fasi: pianificazione, analisi, sviluppo e verifica.

Il raggio delle fasi è il costo accumulato (+ è alto + il costo aumenta) e la dimensione angolare il progresso, favorisce uno sviluppo costante e definito.

Metodologia agili [agiail]

Usata dalla Vem, al centro c'è il cliente che si incontra ogni due settimane per mostrare i progressi fatti e verificare che siano in linea con quello che il cliente vuole.

Basato sul metodo "Scram", ma con regole meno ferree, dove si divide il progetto in sprint (massimo 3 settimane) dove si lavora senza sottoteam e ogni membro del progetto sceglie un task da sviluppare (scelto nello sprint planning); ogni task è creato dal capo progetto e sono molto base.

A fine sprint si mostra al cliente le nuove funzionalità nello sprint review; successivamente si ricomincia con lo sprint planning dove creo e distribuisco i task nuovi e decido quali task vecchi non completati portare allo sprint successivo.

Modellazione

Uno dei momenti più importanti durante lo sviluppo di un software poiché fondamentale per rappresentare il sistema definendo soluzioni più astratte offrendo migliore comprensione, estensibilità, portabilità e modularità.

Attraverso dei modelli ben definiti e basati su paradigmi di programmazione è possibile concettualizzare il sistema rappresentandone gli aspetti salienti ed astraendo i meno significativi.

La modellazione su MCU avviene tramite un controllore che contiene la logica di controllo e gestisce le risorse, è la parte normalmente attiva con un modello a loop di controllo.

Gli elementi controllati invece modellano le risorse gestite dal controllore per svolgere i compiti, contengono le funzionalità utili al controllore e sono tipicamente passivi, solo la parte riusabile.

Paradigma OO

Paradigma ad oggetti offre un buon livello di astrazione con proprietà come modularità, incapsulamento e meccanismi per il riuso e l'estensibilità.

Inoltre permette di avere un supporto naturale alla modellazione software di oggetti reali.

Il modello a loop ci sono alcuni limiti, infatti i loop continuano ad essere eseguiti anche quando non vi sono cambiamenti di input (efficienza) e se il ciclo è piuttosto complesso a livello computazionale si possono introdurre dei ritardi, perdendo eventuali variazioni che vengono compiute durante l'elaborazione (reattività).

Il controller del modello oo non è una classe ma un agente, ovvero:

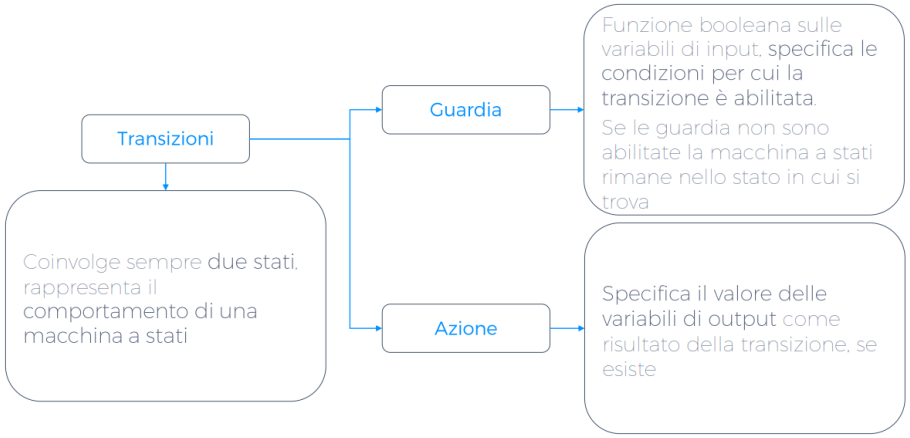
Entità attiva dotata di un flusso di controllo logico autonomo. progettato per svolgere uno o + task che richiedono di elaborare informazioni provenienti da input dei sensori e di agire su attuatori in output, comunicando con altri agenti.

Macchine a stati finiti

Modello di sistema a dinamica discreta, dove ogni input viene mappato in un output, a seconda del suo stato corrente ad ogni reazione rilevata, con un numero finito di stati possibili.

Stato di sistema: condizione in cui si trova in un certo istante temporale, ovvero l'insieme di tutte le attività passate, che determinano la reazione del sistema agli input futuri.

Reazione: step che il sistema effettua, istantanee e scatenate dall'ambiente o da eventi.



Le macchine a stati finiti possono essere:

- Asincrone: cioè event triggered, dove la reazione (o transizione) avviene a fronte di un evento in input;
- Sincrone: cioè time triggered, dove le reazioni avvengono ad intervalli regolari di tempo, per questo motivo questo tipo di macchine sono adatte alla gestione di sistemi time-oriented.

La lettura di un input con un certo periodo di tempo è detto campionamento, se si sceglie bene il periodo di campionamento è possibile non perdere eventi e mantenere la reattività al costo di risorse vista la rapidità del campionamento; il MEST (Minimum Event Separation TIme) è l'intervallo più piccolo che può esserci fra due eventi di input quindi scegliendo un periodo più piccolo del MEST rileveremo tutti gli eventi.

La latenza è il tempo che intercorre tra l'evento di input e la generazione dell'output.

Task

Organizzazione task concorrenti

L'architettura a task concorrenti nasce dalla necessità di modellare e progettare applicativi articolati dove è fondamentale decomporre e modularizzare le funzionalità.

Ogni task ha un compito, un'unità di lavoro e il suo comportamento è descritto da una macchina a stati; dove il comportamento di tutti i task è l'insieme delle varie macchine a stati.

I task possono essere suddivisi in sotto-task in maniera ricorsiva aumentando la modularità portando ad un sistema chiaro, manutenibile, estendibile e riusabile.

Bisogna ricordarsi che i task sono fra loro concorrenti e le loro dipendenze vanno gestite mediante forme di coordinazione cooperative o competitive.

La metodologia cooperativa usa scheduling di tipo round-robin che è più semplice ed è privo di corse critiche, a scapito di un loop infinito in uno dei task che può portare gli altri in una situazione di starvation.

Dipendenze task

Non riuscendo a garantire un'indipendenza completa esistono forme di dipendenza:

- Temporale: per far partire un altro task bisogna aspettare che l'altro finisca;
- Producer/Consumer: un task deve aspettare l'output di un altro;
- Relative ai dati: c'è una risorsa in comune, bisogna aspettare che l'altro task la liberi.

Nelle macchine a stati l'interleaving delle azioni non avviene.

Comunicazione task

Il caso più semplice è utilizzare una variabile globale, così si ha una comunicazione sincrona. C'è la possibilità di gestire i messaggi in maniera asincrona utilizzando per ogni task una coda.

Scheduling e prestazioni

Per ogni task è necessario valutare la loro durata per evitare malfunzionamenti, si possono definire eccezioni di overrun quando il tempo di esecuzione delle azioni oltrepassa il periodo pre stabilito andando a "rubare" tempo di esecuzione al task successivo (timer overrun nel caso di scheduler interrupt-driven).

È possibile definire se si verificherà un overrun analizzando il codice per stimare il Worst-Case Execution Time (con task che usano periodi diversi bisogna considerare il WCET con degli iper-periodi).

$$U = (\textit{tempo utilizzo per task} / \textit{tempo totale}) * 100\%$$

Ottenendo un valore >100% si verificherà un overrun.

Jitter: ritardo che intercorre dal momento che un task è pronto per essere eseguito e il momento in cui viene effettivamente eseguito, dando priorità ai task con periodo piccolo minimizziamo il jitter medio.

Deadline: intervallo di tempo entro il quale un task deve essere eseguito dopo essere diventato ready.

Priorità

Per priorità si intende l'ordinamento con cui lo scheduler esegue i task. Normalmente vi possono essere due tipologie di scheduler:

- Priorità statica: ogni task ha un livello di priorità che non cambia durante l'esecuzione, si usa la policy shortest-deadline first a meno che la deadline non sia uguale altrimenti si passa alla shortest-period first;
- Priorità dinamica: la priorità è determinata in esecuzione e la si da a chi ha una deadline più vicina alla fine.

Preemptive e cooperativi

Sono due tipi di scheduler, il primo può togliere il processo ad un task in esecuzione prima che abbia completato; il secondo aspetta finché il task non sia stato portato a termine ed è utilizzato per realizzare le macchine a stati finiti.

Architetture su eventi

A basso livello le interruzioni di input rappresentano eventi che gestiscono meglio le risorse ed evitano polling.

Si possono utilizzare gli eventi anche per realizzare architetture ad eventi di alto livello come pattern-observer o macchine a stati finiti asincrone.

Pattern-observer

Si sfruttano le interruzioni per far si che all'occorrenza di un'interruzione si chiamino i listener.

Macchine a stati finiti asincrone

Sfruttano le interruzioni per realizzarle, la valutazione delle reazioni avviene a seguito di un evento, non esiste il periodo anche se l'evento può essere di tipo temporale.

IOT

Coniato da Kevin Ashton nel '99, automatizzare l'inserimento dei dati inerenti al mondo fisico in Internet con sensori con l'ipotesi di un futuro interconnesso.

Definizione standard:

"Things having identities and virtuale personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environmental, and user contexts"

resto nelle slides

Comunicazione

In ottica IoT la connessione alla rete può essere:

- Diretta: si ha un modulo connessione;
- Indiretta: usando un gateway, usando un sistema terzo.

Wireless

Da non confondere con il WiFi perchè il wireless comprende il WiFi e tutto ciò che non usa fili per la comunicazione, in sostanza la radiocomunicazione.

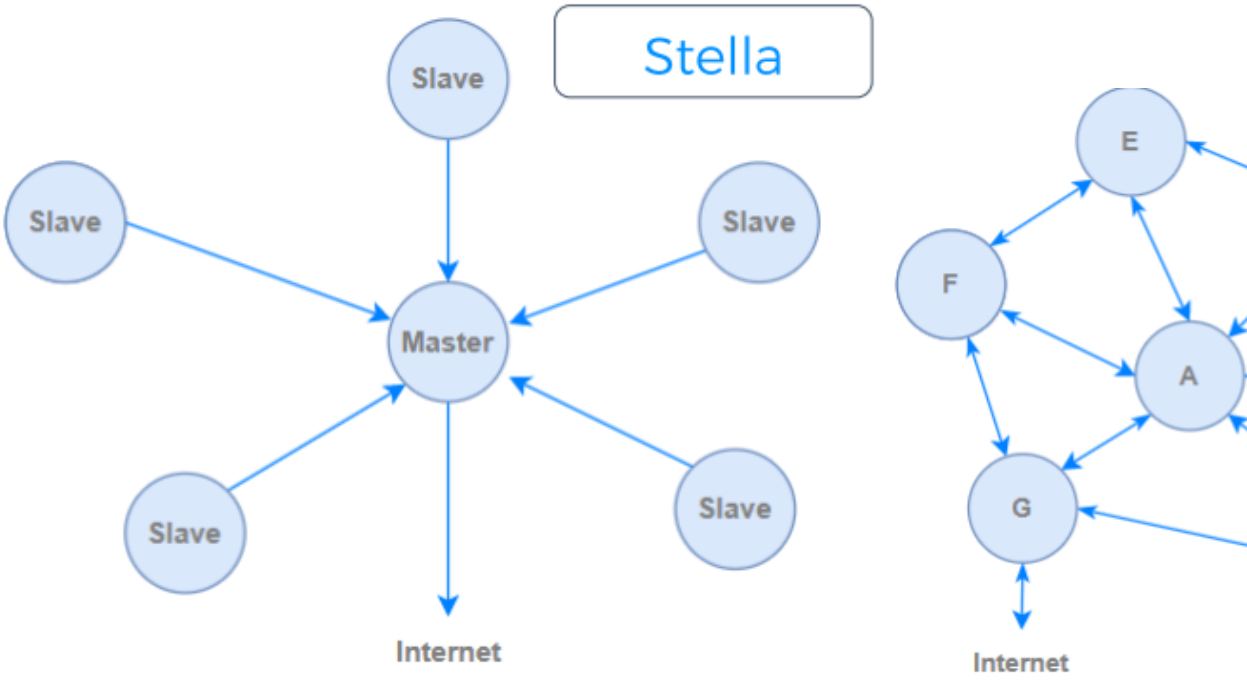
Sfruttando canali radio possiamo trasmettere dati attraverso segnali elettromagnetici che appartengono alla frequenza radio dello spettro elettromagnetico; tali comunicazioni sono satellitari o terrestri.

Sistema radiocomunicazione

Composto da:

- Canale trasmissivo: tutto ciò che serve per trasmettere;
- Modem: codifica da bit-segnale a elettrico e viceversa;
- Codec: codifica e decodifica digitalmente un segnale.

La rete può essere:



WiFi

Standard IEEE 802,11x, supportato da IP, TCP, UDP.

Data rate elevato con un range di 150m, frequenza 2,4 o 5 GHz e con prestazioni non favorevoli per i sistemi embedded.

Reti cellulari

Standard evoluti dal GPRS a LTE, data rate che può essere alto o basso in base allo standard, consumi abbastanza alti e range dell'ordine dei chilometri.

Successivamente è nato il Narrow Band IoT che utilizza le reti cellulari ma con copertura altissima e consumi ridotti a scapito del data rate.

Sistemi RF

Interfacce via radio con data rate a 1 Mbps, range elevati, frequenza di 2,4 GHz e senza supporto di protocolli ad alto livello.

Bluetooth

Protocollo 802.15.1, sfrutta la scoperta dinamica di dispositivi entro 10 metri di raggio, costi di produzione e energia bassi, frequenze libere e con il frequency hopping commuta la banda e permette di ottimizzare l'utilizzo della frequenza.

La rete del BT è detta piconet basata sull'architettura master-slave dove ogni dispositivo comunica con 7 slave e questi slave comunicano in maniera sincronizzata (grazie al clock master) uno per volta con il master. Più piconet connesse insieme sono dette scatternet, dove gli slave possono appartenere a più reti ma il master può essere slave solo di una sola piconet.

ZigBee

Il protocollo IEEE 802.15.4, si basa su uno standard di comunicazione e ne vuole definire uno complementare al WiFi (quindi personale e a basso costo/velocità), il ZigBee usa questa specifica con comunicazioni entro 10 metri, transfer rate di 250 kbits e utilizzo frequenze 868/915/2450 MHz.

La rete è composta da FFD (coordinatore che parla con tutti) e RFD (parla solo con il FED).

Il Zigbee è ideato per l'IoT con basso consumo e basso costo, i dispositivi si dividono in:

- ZigBee Coordinator (ZC): è un ponte fra reti, solo uno per rete e tiene le chiavi di sicurezza [FFD];
- ZigBee Router (ZR): trasmettono dati fra dispositivi [FFD];
- ZigBee End Device (ZED): parla solo con coordinatore e router [RFD]

LoraWAN

Poca potenza ma grossa area di copertura, costo energetico basso e utilizzo frequenze radio.

Struttura con topologia a stella avente un gateway per connettersi ad internet che comunica con un network server (di solito in cloud) che gestisce i downlink e

gli end device possono connettersi a più gateway.

Usando bande libere il protocollo ha dei duty-cycle, cioè un periodo di tempo per usare la banda, in sostanza il data rate è limitato.

Se aumenta il data rate ci sarà più velocità di trasmissione a scapito dello spreading factor (portata delle trasmissioni).

Con spreading factor + alti occorrono pacchetti più piccoli e consumi più alti.

La comunicazione LoraWAN è bidirezionale con classi di device diverse:

- Classe A: il device di campo inizia sempre a comunicare e solo dopo si può rispondere entro una finestra temporale;
- Classe B: come nella classe precedente ma si aprono anche finestre di ricezione schedulate;
- Classe C: finestra sempre aperta a meno che non ci sia una trasmissione in corso.

Bisogna sempre ricevere un uplink prima di inviare un downlink.

RFID

Tracking a distanza tramite tag che hanno un id univoco, i tag possono essere: passivi (non alimentato), attivo (ha batterie e comunica in broadcast), BAP (invia informazioni solo se sollecitato).

NFC

Fornisce connettività radio bidirezionale a corto raggio, quando si accostano i'initiator e il target viene creata una rete peer-to-peer.

Beacon

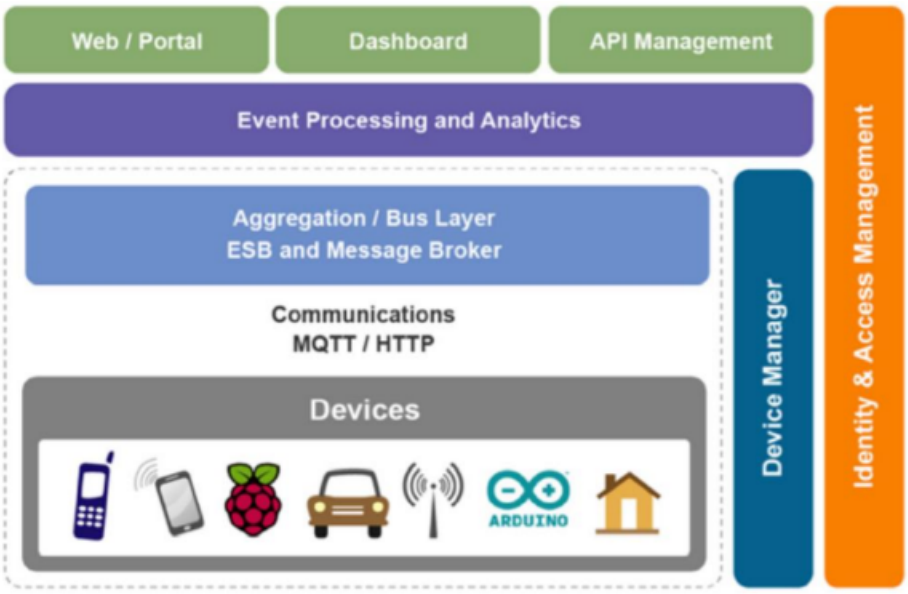
Basata sul BLE utile per localizzazione indoor, si usa un device detto beacon a basso consumo che trasmette di continuo il suo ID.

Middleware

Struttura di riferimento per la standardizzazione della connettività via Internet dei device IoT.

Il loro scopo è fornire interoperabilità (capacità di un sistema o di un prodotto informatico di cooperare e di scambiare informazioni) tramite supporto a protocolli di alto livello, discovery dinamico, aggiornamenti, disconnettere dispositivi problematici, fornire supporto per la memorizzazione, scalabilità e sicurezza.

Si hanno diversi livelli:



Dove:

- Event processing: rileva gli eventi e li processa;
- Aggregation: aggrega e gestisce le comunicazioni e fa da ponte fra i vari protocolli;
- Comunication: diversi protocolli possono essere utilizzati, http, MQTT, CoAP, XMPP;
- Devices: Dispositivi di vario tipo con o senza connessione diretta, identificati a livello HW.

MQTT

Protocollo a scambio di messaggi publish-subscribe, basato sul modello broker.

Vantaggioso per il suo basso overhead, possibilità di funzionare su diversi protocolli (TCP, ZigBee) e con anche la presenza di firewall e garantisce la bontà dei messaggi tramite QOS.

Il QOS ha tre livelli:

1. At most once: viene inviato il messaggio, non importa chi e se qualcuno lo legge, per questo detto fire and forget;
2. At least once: c'è la sicurezza che arrivi a destinazione almeno una volta;
3. Exactly once: il messaggio arriva una sola volta.

CoAP

Basato su HTTP con approccio client-server utilizzando UDP.

XMPP

Basato su scambio di messaggi con struttura message-brokers. Supporta anche altri metodi come il point-to-point, il request-response e l'asyncronous message, costruendo federazioni di server XMPP possiamo ottimizzare la scalabilità.

Modelli di comunicazione

A scambio di messaggi

Comunicazione tramite invio e ricezione di messaggi utilizzando send o receive.

Possono essere:

- Diretta: comunicazione diretta fra processi utilizzando un identificativo univoco;

o

- Indiretta: sfruttando canali di comunicazione.
- Sincrona: la send va a buon fine quando il messaggio è stato recepito tramite receive;

o

- Asincrona: la send va a buon fine quando il messaggio viene inviato.
- Descrizione dei messaggi.

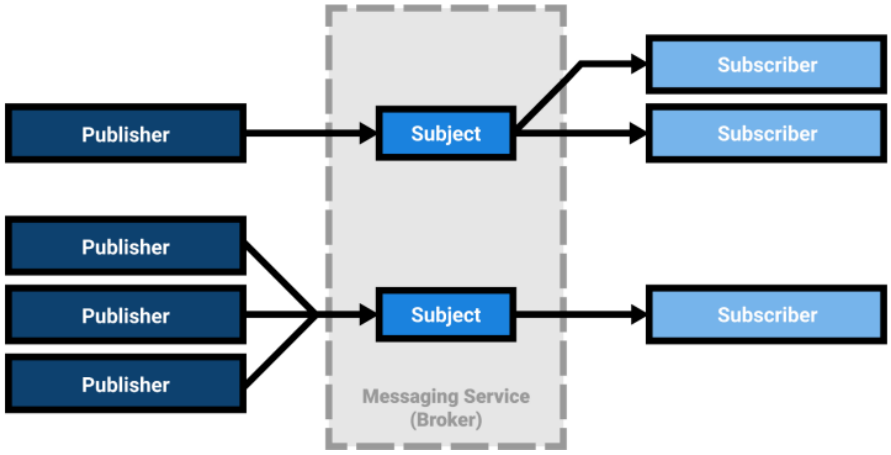
Non bisogna sottovalutare la bufferizzazione dei messaggi, che nel caso asincrono è fondamentale. Nel caso il buffer si riempia bisogna gestire il comportamento

della send scegliendo fra i seguenti meccanismi:

- La send fallisce con scarto del messaggio;
- La send si blocca finchè non c'è disponibilità nel buffer;
- La send ha successo e si riscrive il messaggio nel buffer più nuovo o vecchio.

Per la rappresentazione del messaggio si usa un linguaggio condiviso da tutti come Json o XML e adottare ontologie (sintassi e strutture comuni).

Publish-subscribe



Utilizzato nelle strutture ad eventi e composto da:

- Topics: canali in cui è possibile inviare dei messaggi e che è possibile «osservare»;
- Publisher: possono inviare messaggi sui topics;
- Subscriber: si registrano ai topics per ricevere tutti i messaggi pubblicati su questi.

Domande esame

13 Nel protocollo LoraWAN, lo spreading factor
[ ] è inversamente proporzionale al consumo energetico
[ ] è direttamente proporzionale alla dimensione dei pacchetti
[X] determina anche la portata delle trasmissioni
[ ] determina il solo tempo impiegato per le trasmissioni

14 I pin general purpose
[X] hanno diverse modalità di funzionamento
[ ] sono utilizzati per le comunicazioni su bus I2C
[ ] funzionano solo in modalità Input
[ ] sono solo pin analogici

15 I sistemi operativi (indica l'affermazione non vera)
[X] gestiscono la comunicazione con i sensori
[ ] gestiscono l'accesso alle risorse
[ ] sono software di base
[ ] gestiscono l'esecuzione dei programmi

16 L'accesso ad una memoria EEPROM, spesso utilizzata nei device IoT, come viene eseguito?
[ ] utilizzando una serie di indirizzi di memoria predefiniti
[ ] nessuna delle risposte è corretta
[X] in maniera sequenziale
[ ] in maniera posizionale

17 Il frequency hopping
[X] permette di ottimizzare l'utilizzo della frequenza

18 Il protocollo zigBee
[X] si basa su uno standard di com
[ ] permette comunicazioni anche
[ ] è una tecnologia LPWAN
[ ] utilizzano una comunicazione w

19 Quale delle seguenti affermazioni
[ ] permettono di eseguire istruzion
separate
[ ] permettono di arrestare il codice
[ ] permettono ai microcontrollori d
[X] permettono di sospendere l'ese
routine di interrupt corrispondente, ri

Nella domanda 30 la risposta giusta è: serve un server NTP (e non PTN) per questo sono tutte sbagliate !!!

39 Le metodologie agile
[X] portano al centro il dialogo con il
[ ] sono adatte per gestire la parte H
[ ] ogni finestra di sviluppo è legata
[ ] lo sviluppo viene fatto in una unic

40 I pin PWM permettono
[ ] di alimentare dispositivi esterni
[X] di emulare in uscita un segnale a
[ ] di aumentare la tensione in uscita
[ ] di impostare il duty cycle in un dis

41 Nel caso in cui, in un sistema multi
stessa risorsa, ho una dipendenza

31 La caratteristica statica reale è leggermente diversa da quella ideale a causa di
[ ] errori nelle misure
[ ] variazioni di tensione
[ ] fattori esterni
[X] imperfezioni costruttive

32 Le comunicazioni wireless si suddividono in
[X] terrestri e satellitari
[ ] dirette e indirette
[ ] LoraWAN e ZigBee
[ ] wifi e GSM

33 Nel metodo a spirale, il raggio della spirale rappresenta
[ ] il progresso
[X] il costo accumulato
[ ] l'obiettivo
[ ] il tempo

[ ] deadlock
[X] relativa ai dati
[ ] temporale
[ ] producer/consumer

Domande sbagliate per orale

1. Cosa viene eseguito al riavvio di un device Arduino? Un bootloader.
2. Nelle organizzazioni di task concorrenti... ogni task è descritto tramite una singola macchina a stati finiti.
3. Le porte che permettono la gestione dei pin I/O... sono gestiti da uno o più registri special purpose.
4. Nei dispositivi IoT non aventi un clock interno ma connessi a internet, cosa posso utilizzare per ottenere l'orario corretto? Si usa un server NTP che però è esterno e non dentro il router, se fosse dentro il router anche senza internet avremmo avuto l'orario corretto.
5. Quale fra queste NON è un'attività dei S.O? Gestione della comunicazione tra programmi.
6. Nel protocollo LoraWAN un messaggio di downlink può sempre essere inviato in quale di queste casistiche? In nessuna, i messaggi downlink vengono inviati solo come risposta di uno uplink non importa la classe.
7. Il duty cicle nel protocollo LoraWan... NON è inversamente proporzionale alla dimensione dei pacchetti, ma è il massimo tempo di trasmissione per un determinato periodo di tempo.
8. Quando una variabile condivisa da task deve essere in sezione critica per garantire il corretto funzionamento? Solo se i task sono in scrittura, se è in lettura invece non c'è bisogno.
9. Se ho un sensore di temperatura collegato alla corrente elettrica, in grado di comunicare ad intervalli di 30 minuti con un requisito funzionale di una temperatura media ogni 5 minuti (ogni 5 minuti voglio il valore medio di temperatura di quel periodo) campionamento ogni 500ms quale tempo utilizzo per ogni step della macchina a stati? Un secondo.
10. Quale tra le seguenti caratteristiche è legate a errori accidentali? La precisione.
11. Per gestire la comunicazione in modo asincrono... si possono utilizzare code di messaggi.
12. Quali delle seguenti affermazioni su UART è falsa? L'UART utilizza una comunicazione sincrona. Infatti è asincrono, si appoggia a dispositivi separati per gestire i flussi di dati e utilizza un bus seriale.
13. I gateway LoraWAN... sono l'unico accesso alla rete internet.