BRICKS | TEMA

Interrupt, non polling: il diritto alla pausa insegnato con il linguaggio dei sistemi

a cura di:
Pierangelo Indolfi

Benessere digitale, Mindfulness, Informatica, Progettazione software, Didattica universitaria, Ritmo di lavoro, Disconnessione

Durante Agorà Tech, il percorso di progettazione di oggetti connessi organizzato dalla Provincia di Taranto, una delle scoperte più interessanti per i partecipanti non riguarda il codice che funziona, ma il codice che continua a chiedere senza sosta se qualcosa è successo. È il momento in cui, leggendo insieme uno *sketch* Arduino, ci si accorge che il sensore viene interrogato decine di volte al secondo dentro il *loop* principale, anche quando non c'è nessun evento da rilevare. Da quella scoperta tecnica nasce un'analogia che vale la pena estendere ben oltre il microcontrollore, fino al modo in cui anche le persone gestiscono la propria attenzione.

Contesto della pratica

Agorà Tech è un percorso tematico di sei incontri, ventiquattro ore complessive, promosso dalla Provincia di Taranto, rivolto a persone con un interesse specifico per l'elettronica e l'Internet delle cose, non necessariamente studenti in un percorso scolastico tradizionale: si lavora in coppia, ciascuna con un kit Arduino, verso la certificazione Arduino Fundamentals. L'obiettivo didattico centrale è la progettazione di oggetti connessi, dalla lettura dei sensori alla comunicazione via rete; le competenze coinvolte riguardano la programmazione *embedded*, la gestione degli *interrupt* hardware e, in modo trasversale, la capacità di leggere il proprio rapporto con le notifiche e i flussi informativi continui. È un contesto più libero di un corso universitario strutturato, e proprio per questo si presta bene a un'attività che parte da un concetto tecnico per arrivare a una riflessione personale. La stessa attività, pensata qui per un pubblico di appassionati e curiosi, è facilmente adattabile a corsi universitari più vincolati nei programmi, a percorsi ITS e a corsi di formazione per adulti come quelli dei progetti GOL, e in generale in tutti i contesti in cui si rischia il sovraccarico informativo.

Due strategie, un solo sistema

In un microcontrollore esistono due modi per sapere che un sensore ha rilevato qualcosa. Il *polling* è la tecnica più immediata, ed è spesso la prima che si scrive in uno *sketch* Arduino: dentro il *loop()* si controlla a ogni ciclo lo stato del pin, decine o centinaia di volte al secondo, anche quando non è successo nulla. L'*interrupt* è la tecnica più elegante, e tipicamente arriva dopo, quando ci si accorge dei limiti della prima: si collega il sensore a un pin abilitato agli *interrupt* hardware, il microcontrollore resta libero di fare altro o di entrare in *sleep*, e solo quando il segnale cambia stato viene richiamata la funzione che gestisce l'evento.

In un progetto IoT la differenza non è solo elegante, è vitale. Molti sensori vanno installati in luoghi impervi, un versante di collina, il fondo di una cisterna, un palo lontano dalla rete elettrica, dove non c'è presa di corrente e l'unica alimentazione è una batteria che nessuno vuole andare a sostituire ogni settimana. Un dispositivo che fa *polling* continuo consuma la batteria in pochi giorni. Lo stesso dispositivo, progettato per restare in *sleep* e svegliarsi solo all'*interrupt* del sensore, può funzionare per anni con la stessa pila. La differenza tra le due strategie, in campo, si misura in autonomia: mesi contro anni, senza bisogno di un Supradyn ricarica per il microcontrollore.

Controllare il telefono ogni cinque minuti, tenere trenta schede aperte per non perdere un aggiornamento, rispondere alle email tardi la sera per dimostrare di essere sempre operativi: sono tutte forme di *polling* umano. Sembra vigilanza, è solo spreco di risorse. Non riposa davvero nemmeno chi sembra fermo, esattamente come il microcontrollore che interroga il sensore in *loop* non risparmia energia neanche quando non c'è nulla da leggere.

C'è anche una spiegazione fisiologica precisa. Durante l'attività cognitiva intensa si accumula adenosina, un neurotrasmettitore che segnala al cervello il bisogno di riposo: l'equivalente biologico dell'acido lattico nei muscoli. Solo il sonno la smaltisce davvero. La caffeina, che tanti studenti (e tanti programmatori) consumano nelle sessioni notturne, non elimina la fatica: blocca i recettori dell'adenosina e ne maschera gli effetti. Il conto si paga dopo, con gli interessi. Non è un caso che uno dei linguaggi più diffusi al mondo si chiami Java, e abbia per logo una tazza di caffè fumante: un omaggio involontario all'adenosina bloccata.

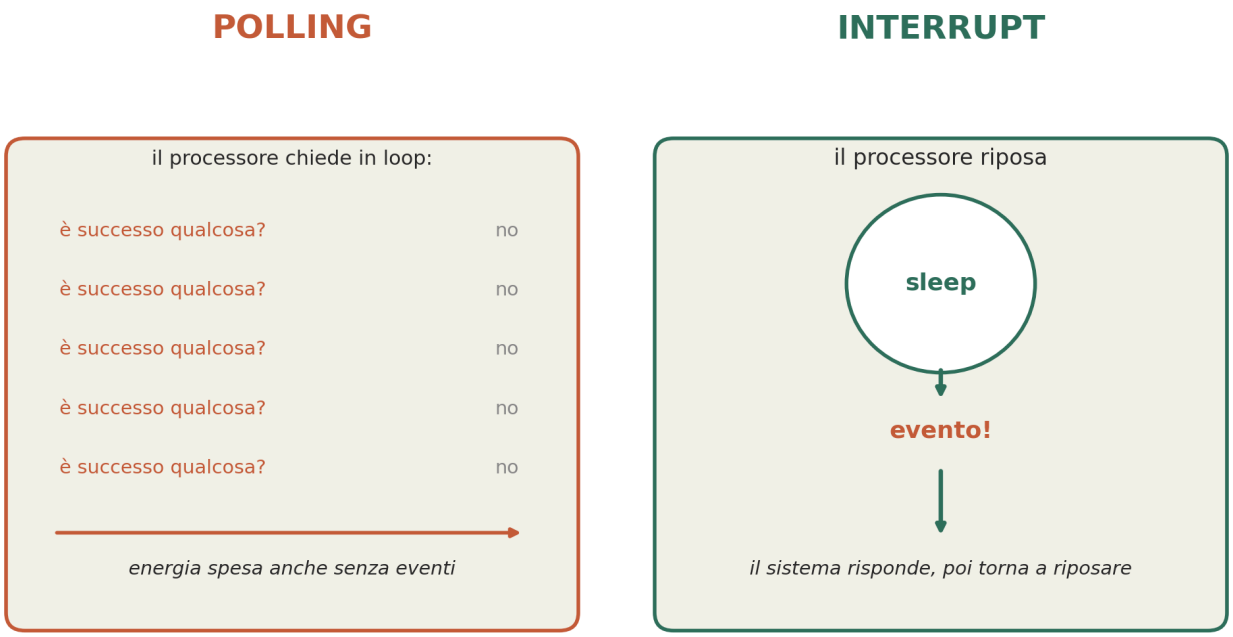


Fig. 1 - Due modi diversi di restare in ascolto

La regola del venerdì

Tra sistemisti e amministratori di rete circola una regola non scritta: non si rilasciano aggiornamenti software il venerdì pomeriggio. Chi lo fa rischia un malfunzionamento scoperto nel weekend, senza nessuno disponibile a risolverlo con calma. Non è superstizione: è il riconoscimento che i sistemi complessi hanno bisogno di tempo per assestarsi dopo un cambiamento, e che non tutti i momenti sono equivalenti per le stesse azioni.

Negli stessi termini, le norme sulla sicurezza del lavoro al videoterminale prevedono pause obbligatorie ogni due ore, e i tempi di guida e di riposo degli autotrasportatori impongono una sosta ogni quattro ore e mezza. Non sono concessioni sindacali: sono il riconoscimento che il corpo, come il sistema, non è un processo infinito.

Anche le tradizioni di pensiero più antiche, ben prima dell'informatica, avevano già individuato questo bisogno: fermarsi non per pigrizia, ma per non perdere il legame con se stessi. Il giorno di riposo settimanale, presente in culture diverse con nomi diversi, non nasce come premio per il lavoro svolto, ma come promemoria che la persona non si esaurisce nella sua produttività. È un'intuizione che la tecnica, con il suo linguaggio di *scheduler* e *interrupt*, sta riscoprendo da un'altra strada: il corpo e la mente che lavorano hanno bisogno di un ritmo che alterni fare e fermarsi, non per essere più efficienti, ma per restare integri.

L'attività in classe: il diario degli interrupt

A metà percorso, dopo che ogni coppia ha scritto e confrontato una prima versione del proprio *sketch* in *polling* e una in *interrup*t, si propone ai partecipanti un piccolo diario per i giorni che separano un incontro dal successivo. Ogni volta che controllano la chat del gruppo, la posta o i social senza un motivo preciso, segnano un evento di *polling*. Ogni volta che vengono interrotti da un messaggio realmente urgente, o che si fermano volontariamente per una pausa programmata, segnano un evento di *interrup*t. All'incontro successivo, prima di tornare al progetto Arduino, si dedicano dieci minuti al confronto tra i diari, attorno a tre domande: quanti dei controlli erano davvero necessari, in quali momenti della giornata si concentrano gli eventi di *polling*, e cosa succederebbe se si stabilisse, come regola condivisa, una finestra quotidiana di silenzio in cui nessuno controlla nulla.

Non è un esercizio di produttività, anche se i progetti finiscono più ordinati. È un esercizio per riconoscere, con un linguaggio già familiare perché appena praticato sul microcontrollore, la differenza tra essere presenti ed essere semplicemente svegli. Il sistema che fa *polling* sembra vigile e invece consuma se stesso senza necessità. Il sistema che gestisce *interrup*t riposa per davvero, e risponde meglio quando serve.



Fig. 2 - Il diario della settimana

Una pausa non è un guasto

A chi progetta oggetti connessi si insegna, giustamente, che ogni dispositivo deve essere reattivo, affidabile, sempre pronto a rispondere. Ma nessun sistema reale è sempre attivo al massimo, e quelli più affidabili sono proprio quelli che sanno quando entrare in *sleep*. Lo *scheduler* di un sistema operativo, su scala diversa, fa lo stesso: non manda avanti tutti i processi insieme, li mette in pausa a turno, perché senza interruzione non c'è stabilità nella distribuzione delle risorse. Vale anche per chi quei sistemi li progetta: una pausa non è un guasto nel processo di apprendimento, è una sua componente strutturale, tanto quanto lo è in un dispositivo ben progettato.

Trasferire questa idea a un percorso di elettronica e IoT non significa aggiungere un modulo di benessere accanto alla didattica tecnica. Significa mostrare che il linguaggio tecnico appena appreso, *polling*, *interrupt*, *sleep*, contiene già gli strumenti concettuali per leggere il proprio rapporto con il lavoro e con le pause. Non serve importare la *mindfulness* dall'esterno come argomento estraneo alla materia: basta restituire alla materia la sua capacità di parlare anche del corpo che la pratica.

Del resto l'informatica ha sempre funzionato così: prende in prestito parole e idee dalla vita di tutti i giorni, una pila di piatti per descrivere una struttura dati, un evento che interrompe per descrivere un comportamento del processore, e a forza di usarle per progettare macchine finisce per restituirle arricchite di senso, pronte a essere riapplicate da dove erano partite. Il *polling* e l'*interrupt* sono nati per i sensori. Tornano utili, raccontati in classe, anche per chi i sensori li progetta.



Pierangelo Indolfi

pierangelo.indolfi@poliba.it

Politecnico di Bari; ITS Academy Apulia Digital

Insegna Laboratorio di Progettazione Software e Informatica per l'ingegneria al Politecnico di Bari e Informatica all'ITS Academy Apulia Digital. Ha insegnato informatica per ventinove anni nelle scuole superiori di Bari. È vincitore del MIT App Inventor Foundation Teacher Excellence Award 2023.