Università degli Studi di Perugia

Dipartimento di Matematica e Informatica

Relazione / Documentazione Tecnica - P.A.N.

Studenti

Michał Horowski

Danilo Palladino

Dedi Vittorio

Anno Accademico 2024–2025

Introduzione al progetto

Il progetto P.A.N. è nato con l'idea di sviluppare un sistema IoT distribuito che simuli il funzionamento di un braccio robotico saldatore industriale. Il sistema si basa sulla comunicazione wireless tra due schede BBC micro:bit utilizzando componenti del kit Keyestudio. La prima scheda, configurata come fosse un joystick ha tre potenziometri e permette all'utente di impartire comandi di movimento. La seconda scheda riceve questi comandi e li traduce in movimenti fisici attraverso tre servomotori che controllano le articolazioni del braccio robotico. Tutti i dati operativi vengono registrati in tempo reale e visualizzati attraverso una dashboard web interattiva che permette il monitoraggio completo del sistema.

Descrizione del Sistema

Il sistema è basato su due nodi micro:bit comunicanti via radio. La scheda trasmittente integra tre potenziometri analogici collegati ai pin P0, P1 e P2. Poi abbiamo collegato un pulsante digitale sul pin P5, che funge da comando per l'attivazione della simulazione di saldatura (accensione del led). La scheda ricevente controlla tre servomotori standard collegati ai pin P8, P12 e P16. Infine abbiamo un LED collegato al pin P14. Entrambe le schede utilizzano shield Keyestudio per facilitare i collegamenti e pacchi batteria esterni per garantire alimentazione stabile e indipendente. La struttura fisica del braccio robotico è stata progettata utilizzando Fusion 360, questa si compone di tre sezioni principali: la base rotante che ospita il primo servomotore, lo snodo centrale che permette l'elevazione del braccio, e il terminale superiore che controlla l'orientamento finale della parte terminale. Le parti sono state stampate in PLA con colori azzurro e rosa fluorescente. Il design permette l'accesso diretto ai componenti elettronici, ciò ci ha semplificato le operazioni di manutenzione del sistema.

La comunicazione tra le schede utilizza il modulo radio integrato del micro:bit, configurato sul canale 42 alla massima potenza disponibile per garantire affidabilità. Il protocollo sviluppato utilizza messaggi testuali strutturati nel formato "P1:valore1,P2:valore2,P3:valore3,BTN:stato" per trasmettere simultaneamente tutti i dati di controllo. Per ottimizzare l'utilizzo della banda radio ed evitare congestione, la trasmissione avviene solo quando vengono rilevate variazioni significative nei valori dei potenziometri, con un timeout massimo di 100 millisecondi per garantire aggiornamenti regolari anche in condizioni statiche.

Implementazione del software trasmittente

Il software della scheda trasmittente è implementato in MicroPython e segue una logica ciclica caratteristica della scheda con cui stiamo lavorando. All'inizio viene inizializzata e configurata la configurazione radio attraverso la funzione radio.config() che imposta il canale di comunicazione a 42, la potenza di trasmissione al massimo livello disponibile e la lunghezza massima del messaggio a 100 caratteri per garantire compatibilità con il protocollo sviluppato.

Il loop principale esegue letture continue dei tre potenziometri analogici attraverso le funzioni pin0.read_analog(), pin1.read_analog() e pin2.read_analog(), ottenendo valori nel range 0-1023 che rappresentano la posizione angolare desiderata per ciascun servomotore. Questi valori vengono elaborati dalla funzione read_potentiometers() che restituisce una tupla contenente i tre valori analogici letti contemporaneamente.

La gestione del pulsante digitale avviene tramite la configurazione del pin P5 in modalità pull-down, garantendo letture stabili e prive di fluttuazioni casuali quando il pulsante non è premuto. La funzione read_button() interroga lo stato digitale del pin restituendo un valore booleano che indica se il comando di saldatura è attivo.

Il sistema implementa un meccanismo intelligente di trasmissione basato sul rilevamento delle variazioni significative nei valori dei sensori. La logica di change_detected verifica se almeno uno dei tre potenziometri ha subito una variazione superiore a 10 unità rispetto all'ultima trasmissione, oppure se lo stato del pulsante è cambiato. Questo approccio riduce drasticamente il traffico radio trasmettendo solo quando necessario, preservando la banda di comunicazione e riducendo il consumo energetico.

La funzione create_radio_message() costruisce il messaggio strutturato concatenando i valori dei potenziometri e lo stato del pulsante nel formato prestabilito. Il messaggio viene quindi trasmesso attraverso radio.send() che utilizza il protocollo di comunicazione wireless integrato nel micro:bit.

Per fornire feedback visivo all'operatore, il sistema implementa la funzione show_pot_values() che analizza i tre valori dei potenziometri e visualizza sul display LED il numero corrispondente al potenziometro con il valore più alto, permettendo di identificare immediatamente quale asse del braccio robotico sta ricevendo il comando principale.

Il ciclo principale mantiene una frequenza di aggiornamento controllata attraverso la variabile send_interval impostata a 100 millisecondi, garantendo che anche in assenza di variazioni significative, il sistema trasmetta periodicamente per mantenere attiva la connessione con la scheda ricevente. Tutti i dati trasmessi vengono simultaneamente inviati alla porta seriale per consentire il monitoraggio e il debugging del sistema.

Flow chart sistema trasmittente:

INIZIALIZZA sistema trasmittente:

CONFIGURA radio(canale=42, potenza=7)

CONFIGURA potenziometri su PIN0, PIN1, PIN2

CONFIGURA pulsante su PIN5

IMPOSTA pull-down per pulsante

LOOP principale trasmittente:

LEGGI valori potenziometri (0-1023)

LEGGI stato pulsante (0/1)

SE cambiamenti rilevati O timeout intervallo:

CREA messaggio "P1:val1,P2:val2,P3:val3,BTN:stato"

TRASMETTI via radio

AGGIORNA display con potenziometro più alto

LOG valori trasmessi

ATTENDI 100ms

Implementazione del software ricevente

Nel codice della scheda ricevente viene implementato un sistema di ascolto continuo del canale radio dedicato. Il software include funzioni di conversione per tradurre i valori dei potenziometri (0-1023) in angoli servo appropriati (0-180 gradi).

La funzione pot_to_angle() implementa una logica di mappatura inversa che limita il range massimo a 512 per evitare sovraccarichi meccanici sui servomotori, applicando la formula 180 - (valore/512)*180 per ottenere un comportamento intuitivo del controllo.

La conversione successiva in segnali PWM utilizza la funzione angle_to_pwm() che mappa linearmente gli angoli nel range PWM 26-128 appropriato per i servomotori standard.

Il sistema include piccoli meccanismi di sicurezza: un timeout di connessione infatti monitora continuamente la ricezione di dati, e in caso di perdita del segnale radio per oltre 5 secondi, tutti i servomotori vengono automaticamente posizionati in una configurazione sicura a 180 gradi e il LED viene spento.

Tutti i dati ricevuti e le azioni intraprese vengono trasmessi via porta seriale per consentire il monitoring esterno.

Flow chart sistema ricevente:

INIZIALIZZA sistema ricevente:

CONFIGURA radio(canale=42, potenza=7)

CONFIGURA servo su PIN8, PIN12, PIN16 (periodo PWM 20ms)

CONFIGURA LED su PIN14

POSIZIONA tutti servo a 180° (stato iniziale)

FUNZIONI conversione:

pot_to_angle(valore_pot):

LIMITA valore_pot a max 512

RITORNA 180 - (valore_pot/512)*180 // Inversione angolo

angle_to_pwm(angolo):

RITORNA PWM tra 26-128 per angoli 0-180°

LOOP principale ricevente:

SE messaggio radio ricevuto:

PARSA "P1:val,P2:val,P3:val,BTN:val"

PER ogni servo (1,2,3):

angolo = pot_to_angle(valore_potenziometro)

pwm = angle_to_pwm(angolo)

APPLICA PWM al servo

stato_led = 1 - stato_pulsante // Logica invertita

APPLICA stato al LED

AGGIORNA timeout connessione

LOG dati ricevuti e applicati

SE timeout connessione superato:

POSIZIONA tutti servo a posizione sicura (180°)

SPEGNI LED

MOSTRA stato disconnesso

Architettura dell'interfaccia web

L'interfaccia web è costruita su un'architettura moderna che utilizza Flask come microframework backend integrato con Socket.IO per comunicazioni real-time. Il server espone una serie di endpoint REST API che permettono l'accesso ai dati storici e real-time memorizzati in InfluxDB.

L'endpoint /api/latest fornisce i dati più recenti del sistema con latenza inferiore a 100 millisecondi, mentre /api/history permette il recupero di serie temporali configurabili per la generazione di grafici storici. L'endpoint /api/measurements implementa campionamento intelligente dei dati ogni 30 secondi per popolare tabelle di riepilogo senza sovraccaricare l'interfaccia utente.

Il sistema WebSocket mantiene connessioni persistenti con tutti i client connessi, trasmettendo aggiornamenti ogni 2 secondi. Questo approccio elimina la necessità di polling continuo da parte del frontend, riducendo significativamente il carico di rete e migliorando la responsiveness dell'interfaccia.

L'interfaccia utente è sviluppata con tecnologie web standard (HTML5, CSS3, JavaScript ES6) e utilizza la libreria Chart.js per visualizzazioni grafiche avanzate. La dashboard è organizzata in tre sezioni principali accessibili tramite navigazione tab-based.

La sezione "Overview" presenta lo stato real-time del sistema con indicatori visuali per ogni servomotore, mostrando simultaneamente valori angolari, percentuali di attivazione, valori PWM e stato operativo. Gli indicatori implementano animazioni fluide e color-coding per facilitare l'interpretazione immediata dello stato del sistema.

La sezione "Charts" offre visualizzazioni temporali interattive con capacità di zoom e pan per analisi dettagliate. I grafici vengono aggiornati automaticamente senza perdita di continuità visiva, mantenendo uno storico di 50 punti dati per bilanciare performance e utilità analitica.

La sezione "Table" presenta dati campionati in formato tabellare con funzionalità di filtering, sorting e export CSV per analisi offline. Il sistema supporta filtri temporali personalizzabili e refresh automatico configurabile dall'utente.

Conclusioni

Durante lo sviluppo sono emerse diverse sfide tecniche, tra queste abbiamo la comunicazione radio che inizialmente mostrava instabilità con perdita di pacchetti.

Inoltre la gestione del flusso dati seriali ha presentato diverse complessità dovute a rumore e interruzioni nella trasmissione.

Una sfida particolarmente complessa è stata la conversione matematica dei valori trasmessi via radio. I potenziometri forniscono valori analogici nel range 0-1023, che devono essere mappati in angoli servo compresi tra 0 e 180 gradi.

La conversione diretta lineare risultava inadeguata per questo è stata implementata una funzione di mappatura che limita il range massimo di movimento a 512 unità per evitare sollecitazioni meccaniche eccessive sui servomotori.