<Inserire il titolo del progetto>

Perini Lorenzo **Matricola 122640**

INDICE

Analisi

1.1 Testo assegnato

Realizzare, un sistema di irrigazione "a getto" intelligente con le seguenti caratteristiche:
1. Due servo del tipo: https://docs.wokwi.com/parts/wokwi-servo dove sono montati due irrigatori a getto, distanti ed orientati su angoli di getto differenti e non interferenti.
2. 5 sensori sensori di umidità, disposti in un terreno a circa 30 gradi l'uno dall'altro per coprire l'arco di 180 gradi.
Il programma deve poter consentire di impostare:
a) Tramite interfaccia seriale, al reset, la data ed ora corrente e fino a 16 orari, all'interno di ciascuna giornate per avvio di un task di controllo ed eventuale irrigazione, specificandone la durata
b) I livelli massimi (troppo umido) e minimi (troppo secco) previsti e quindi decidere, in ciascun ciclo di controllo, se avviare o meno il relay di irrigazione.
L'irrigazione consiste nel far oscillare i due irrigatori con un arco di 180 gradi ad una velocità relativa variabile in funzione dell'umidità rilevate daisensori, in determinate zone radiali.
Il sistema si pone in stato di “Power Down”, una volta terminato un ciclo di attività e potrà essere sveglialo da:

a. dalla premuta di un pulsante per avvio immediato di un ciclo di irrigazione
b. dall’orario di avvio di un ciclo di irrigazione

1.2 Analisi del problema

Il sistema proposto è un’unità automatizzata per l’irrigazione intelligente, progettata per massimizzare l’efficienza idrica e adattarsi alle condizioni del suolo. Dispone di due irrigatori rotanti su servomotori con copertura di 180°, posizionati per evitare sovrapposizioni e garantire una distribuzione uniforme dell’acqua.

Il terreno è monitorato da cinque sensori di umidità, disposti a intervalli di 30°, per rilevare la distribuzione dell’umidità in più aree del terreno. Il sistema consente fino a 16 attivazioni giornaliere, programmabili via interfaccia seriale, ciascuna con orario e durata personalizzabili.

L’irrigazione si attiva se l’umidità è sotto una soglia minima definita dall’utente, mentre è inibita se sopra una soglia massima, per evitare sprechi. L’erogazione è modulata dinamicamente: i getti ruotano più lentamente sulle aree più secche e più velocemente su quelle già umide, dosando l’acqua in modo mirato.

Infine, il sistema entra in modalità "power down" per il risparmio energetico, risvegliabile sia automaticamente agli orari programmati, sia manualmente tramite un pulsante fisico.

Elementi computazionali:

Specifiche numeriche calcolabili:

Il sensore DHT22 restituisce due misure digitali: la temperatura e l'umidità relativa dell'aria. Nel sistema in esame, viene utilizzato esclusivamente il valore relativo all'umidità. Per una corretta implementazione del sensore nel sistema proseguiamo con l'analizzare e definire tutte le specifiche numeriche utili ai fini del progetto.

Carico di memoria

- Ogni valore di umidità = **16 bit** (2 byte)
- Con 5 sensori → **10 byte per ciclo**
- Eventuale salvataggio temporaneo di:
    - Stato logico per ciascuna area (secco / adeguato / umido): **1 bit per zona**, arrotondato a **1 byte ciascuna**
- Totale memoria minima per ciclo = **15 byte**
  (Dati + stati locali, senza buffer storici)

Tempi di acquisizione e scansione

- Ogni sensore DHT22 richiede:
    - Circa 2 ms per l'avvio della comunicazione
    - Fino a 4 ms per il completamento della trasmissione dei 40 bit (temperatura+umidità)
- Con **5 sensori**, la lettura completa richiede:
    - **~25 ms complessivi** per un ciclo di acquisizione
- È possibile calcolare:
    - Il tempo minimo tra due cicli di lettura completi
    - La frequenza massima di aggiornamento dell'umidità per ogni area

Tempi calcolabili per ciclo decisionale

- Tempo di acquisizione: **~25 ms**
- Tempo di confronto + logica decisionale: **~1–2 ms per sensore**

- Tempo totale per decisione su tutte le aree: **~35–40 ms**

Precisione e risoluzione dei dati

- Umidità misurabile: **0–100%**
- Risoluzione interna: **0.1%**
- Valori possibili: **0 – 1000 unità digitali (interi)**

È possibile definire un numero finito di intervalli di irrigazione, ad esempio:

- < 300 → secco
- 300–700 → adeguato
- 700 → troppo umido

Dalla richiesta relativa al sistema da implementare possiamo dedurre che:

- Ogni sensore fornisce un dato autonomo e viene analizzato singolarmente
- Il valore è confrontato con:
    - Una **soglia minima** → indica terreno troppo secco
    - Una **soglia massima** → indica umidità eccessiva
- Il confronto è effettuato per ciascun sensore, senza medie o aggregazioni

Questo consente un controllo puntuale dello stato idrico in ogni area specifica del terreno.

Valori di ingresso

L'Arduino Uno riceve da ciascun sensore DHT22:

- Una sequenza digitale a singolo filo di 40 bit totali (16 bit dedicati all'umidità)
- Ogni sensore è connesso a un pin digitale dedicato
- La comunicazione è temporizzata e gestita in modo sequenziale

Valori di uscita

Per ogni dato ricevuto, Arduino esegue:

- Un confronto diretto con le soglie impostate
- La determinazione dello stato idrico locale:

- Secco
- Adeguato
- Troppo umido

Questi stati logici sono indipendenti per ciascuna area e costituiscono la base per eventuali azioni di controllo.

Casi estremi

Terreno completamente secco

- Tutti i sensori rilevano valori molto al di sotto della soglia minima.
- Il sistema deve irrigare l'intera area, quindi:
  - Entrambi i servo operano sul massimo arco.
  - Massimo tempo di attivazione del relay.
  - Carico massimo su alimentazione, logica e componenti meccanici.

Terreno completamente saturo

- Tutti i sensori rilevano valori superiori alla soglia massima.
- Nessuna irrigazione deve essere avviata.
- Il sistema entra in modalità di inattività subito dopo il controllo.

Distribuzione disomogenea dell'umidità

- Alcune aree sono secche, altre umide.
- Gli irrigatori devono modulare:
  - la durata di sosta su zone diverse,
  - la velocità angolare in modo variabile.
- Richiede calcoli e gestione differenziata zona per zona.

Errore o malfunzionamento dei sensori

- Un sensore non comunica oppure restituisce valori incoerenti.
- Il sistema deve:
    - rilevare l'anomalia,
    - evitare comportamenti imprevisti (es. irrigare inutilmente),
    - eventualmente segnalare il guasto.

Numero massimo di orari programmati

- L'utente imposta tutti i 16 orari disponibili nella stessa giornata.
- Il sistema deve:
    - gestire una tabella piena di eventi,
    - distinguere correttamente tra avvii validi e già eseguiti,
    - evitare sovrapposizioni logiche.

Avvio manuale durante un ciclo attivo

- L'utente preme il pulsante mentre un ciclo di irrigazione è già in corso.
- Il sistema deve decidere se:
    - ignorare la richiesta,
    - accodare un secondo ciclo,
    - interrompere e riavviare.

Mancata sincronizzazione temporale

- All'avvio non viene impostata correttamente la data/ora.
- Tutti gli orari programmati diventano inutilizzabili.
- Serve un meccanismo di fallback o blocco delle attività automatiche.

Vincoli hardware

Nel contesto della progettazione di un sistema, è fondamentale identificare e considerare fin da subito i vincoli imposti dai componenti hardware utilizzati. Tali vincoli possono limitare le prestazioni, influenzare la scelta dell'architettura software e imporre soluzioni specifiche per garantire l'affidabilità del sistema. Analizziamo di seguito l'hardware che verrà utilizzato per estrapolare da esso possibili limitazioni.

Vincoli del sensore DHT22

- Tipo di segnale: digitale
- Tempo minimo tra due letture: 2 secondi
    - Non è possibile interrogare il sensore più frequentemente
- Tensione di alimentazione: 3.3–6 V
    - Compatibile con Arduino Uno (5V)
- Corrente assorbita: ~1–2 mA durante la lettura
- Dimensione dati: 40 bit (di cui 16 per l'umidità)

Limitazioni pratiche:

- Ogni sensore richiede un pin digitale dedicato su Arduino
- Le letture sono bloccanti: Arduino deve attendere la fine della trasmissione (~4–5 ms per sensore)

Vincoli dei servomotori (tipo standard, PWM analogici)

- Alimentazione consigliata: 5–6 V
- Assorbimento di corrente:
    - A vuoto: 100–250 mA
    - Sotto carico: fino a 1 A o più
- Controllo via PWM:
    - Necessitano di segnali da pin PWM (servo library) ogni ~20 ms
    - Arduino Uno può controllare fino a 12 servo con la libreria Servo.h, ma realisticamente 2–4 in modo stabile

Vincoli dell'Arduino Uno

- Microcontrollore: ATmega328P
- Memoria:
    - SRAM: 2 KB → per variabili e dati temporanei
    - Flash: 32 KB → per il codice programma
    - EEPROM: 1 KB → per salvataggi persistenti
- Pin disponibili:
    - 14 pin digitali I/O (di cui 6 PWM)
    - 6 ingressi analogici
- Tensione di funzionamento: 5 V
- Timer limitati

Specifiche non vincolanti

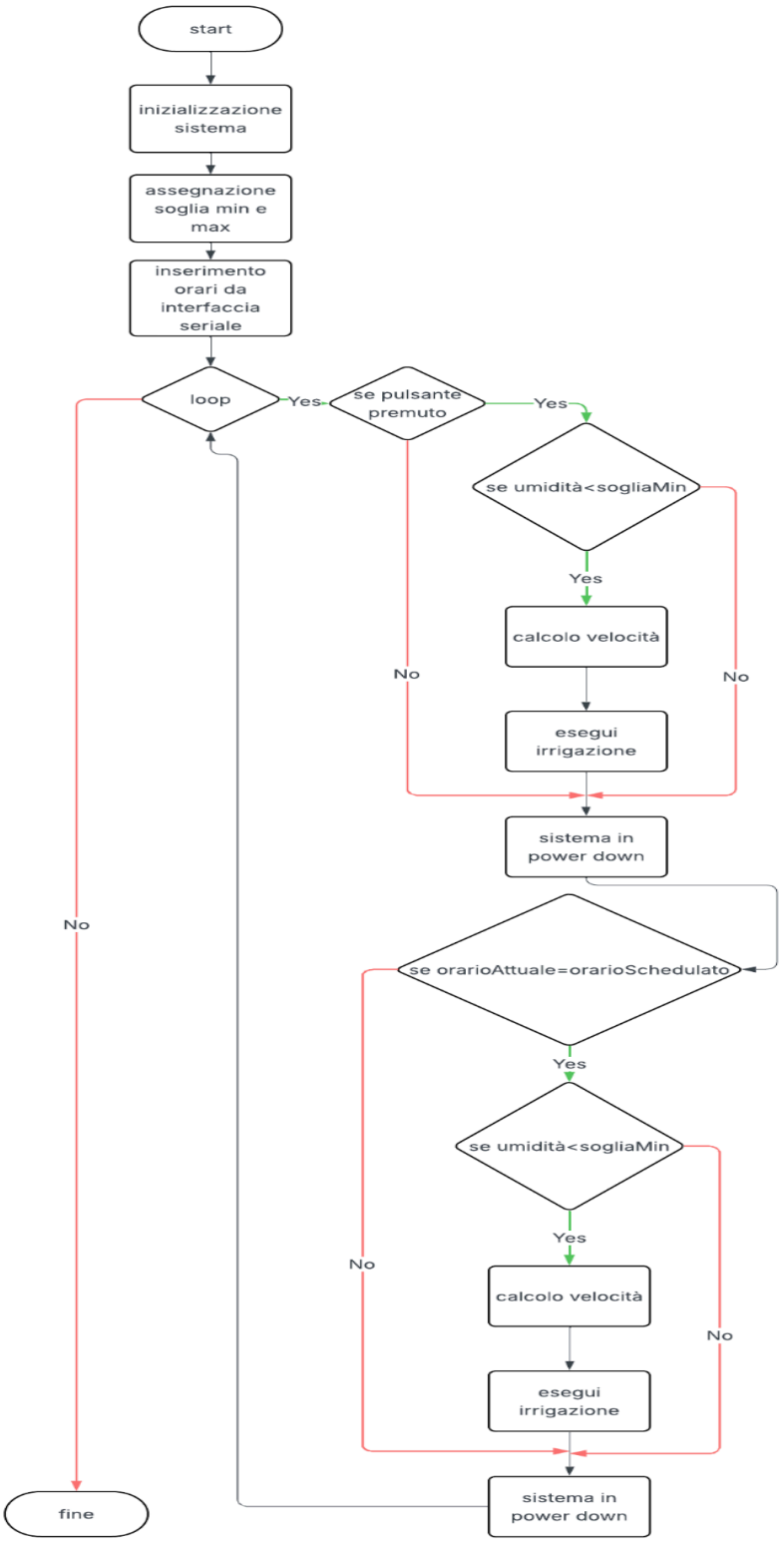
Nel sistema di irrigazione, la **soglia minima** e la **soglia massima** di umidità sono parametri fondamentali da definire per il corretto avvio e arresto del ciclo di irrigazione. Queste soglie non sono imposte direttamente dai componenti hardware, ma devono essere scelte dall'utente in modo tale da garantire un'irrigazione efficace e un buon mantenimento del terreno.

- **Soglia minima**: valore di umidità sotto il quale si avvia il ciclo di irrigazione.
- **Soglia massima**: valore di umidità oltre il quale il terreno non necessita di essere irrigato.

Le soglie devono essere scelte con attenzione per garantire una gestione equilibrata dell'umidità nel terreno, evitando sia l'eccesso che la carenza di acqua. La distanza tra soglia minima e massima è un aspetto importante da considerare per evitare attivazioni e disattivazioni frequenti e ridondanti del sistema.

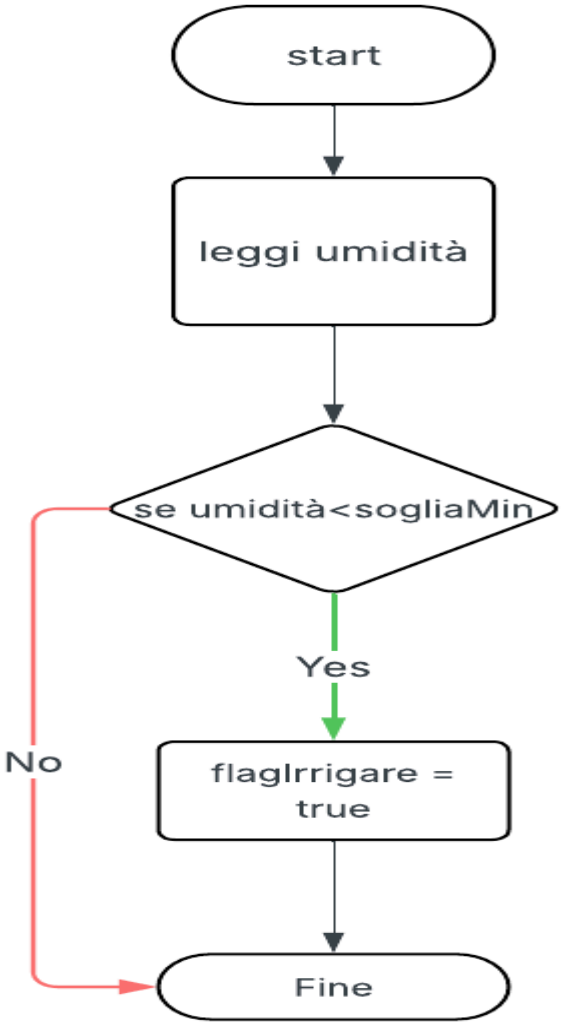
Flusso Logico del Sistema

Flowchart generale

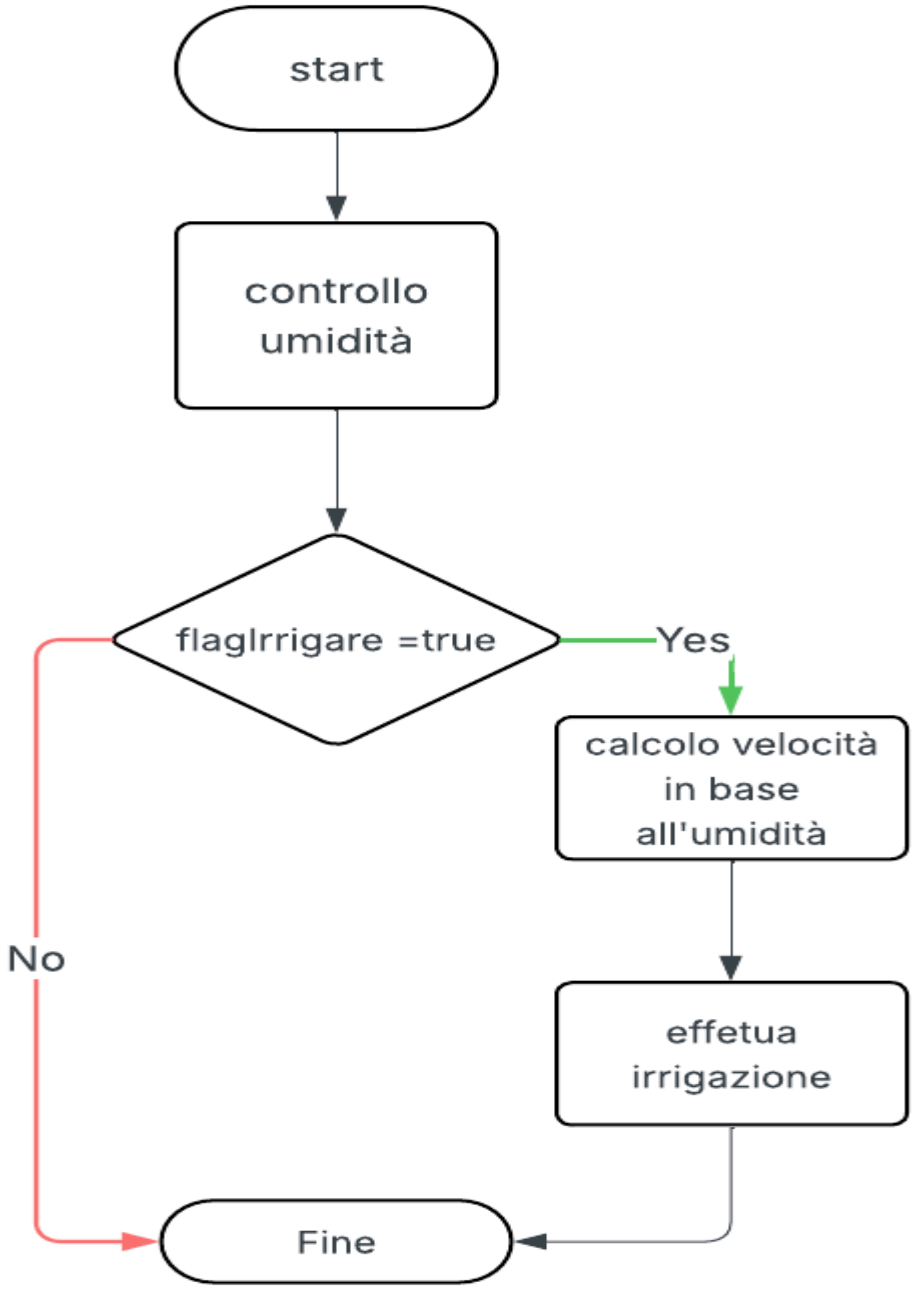


Isolamento dei Componenti logici

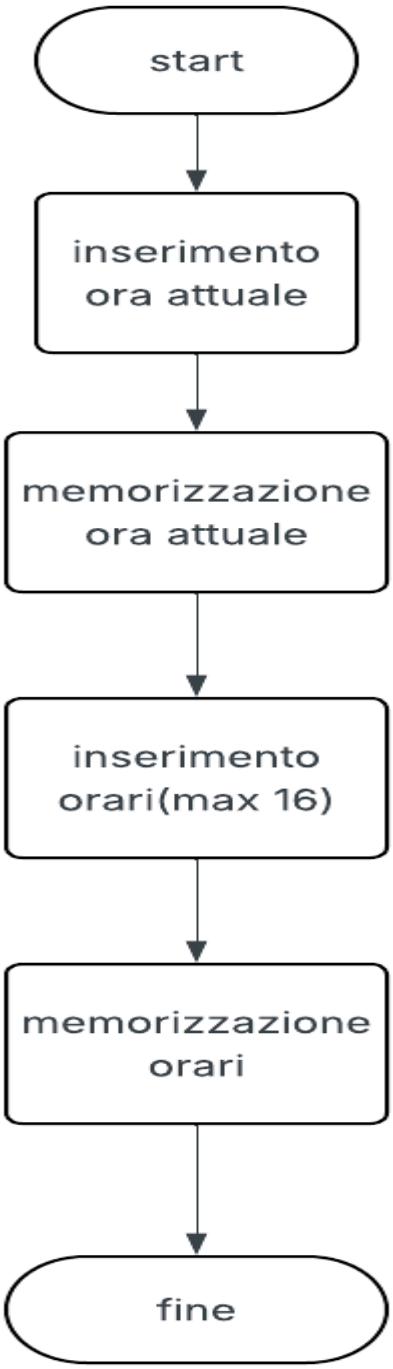
CONTROLLO UMIDITÀ (da intendere per ogni settore)



CICLO IRRIGAZIONE (automatico e manuale)



INPUT DA INTERFACCIA SERIALE



1.3 Analisi della soluzione adottata

Per ottenere il comportamento desiderato del sistema di irrigazione automatico e manuale, è stata adottata una soluzione che privilegia la semplicità progettuale, l'efficienza nell'uso delle risorse hardware (timer, ADC, PWM) e la robustezza operativa.

Il sistema è stato progettato come un controllore ad evoluzione ciclica che alterna stati di attesa passiva con stati di attivazione della procedura di irrigazione, sulla base di condizioni temporali (orari programmati) o su richiesta manuale (pressione del pulsante).

Diagramma di stato complessivo

Il sistema può essere descritto tramite un semplice diagramma di stato a 4 stati principali:

- **Idle / Sleep**: stato di attesa, in cui il sistema aggiorna l'orario interno e verifica la pressione del pulsante o il raggiungimento di un orario di irrigazione.
- **Irrigazione in corso**: attivazione della sequenza di irrigazione, che prevede la lettura dei sensori di umidità e il movimento dei servo per irrigare selettivamente le zone che richiedono acqua.
- **Attesa pre-reset**: eventuale pausa temporale per consentire il completamento della distribuzione dell'acqua.
- **Reset servo**: ritorno delle testine di irrigazione nella posizione di partenza.

Il passaggio tra questi stati avviene in modo deterministico e sincrono, senza uso di multitasking né di interruzioni hardware, semplificando l'implementazione.

Moduli di elaborazione indipendenti

Modulo	Ingressi	Uscite	Funzione principale
Gestione PWM (Timer1)	Angolo richiesto	Segnale PWM ai servo	Generazione segnale PWM per il controllo dei servo
Lettura umidità (ADC)	Canale analogico (A0-A4)	Valore umidità (0-1023)	Conversione analogico-digitale dei sensori di umidità
Gestione orologio interno	millis()	Variabili ora/minuto/secondo	Simulazione di un orologio software senza RTC esterno
Logica di irrigazione	Orario, stato pulsante, umidità	Comando movimento servo	Decisione dinamica di irrigazione zona per zona
Interfaccia utente seriale	Comandi/testo da seriale	Stampa su seriale	Configurazione iniziale e feedback operativo

Componenti fisici rilevanti

- Servomotori standard controllati tramite PWM a 50 Hz. Il controllo viene implementato con Timer1 in modalità Fast PWM per garantire una temporizzazione precisa e stabile.
- Sensori di umidità analogici (o potenziometri per simulazione). La lettura avviene tramite convertitore ADC a 10 bit integrato in ATmega328P.
- Pulsante manuale con pull-up interno abilitato, gestito tramite polling con debounce software.
- LED di stato (pin 13) opzionale, utilizzabile per feedback visivo durante lo sviluppo e il debug.

Algoritmi computazionali

I principali algoritmi sono:

- Gestione orologio software: semplice conteggio di secondi/minuti/ore tramite confronto con millis().
- Controllo irrigazione:
    - Lettura delle umidità per tutte le zone.

- Calcolo della velocità di movimento dei servo per ciascuna zona.
- Movimentazione selettiva dei servo:
  - Se la zona è secca → movimento lento per irrigazione.
  - Se la zona è umida → movimento rapido per saltare la zona.

- Gestione pulsante manuale con debounce:
  - Rilevamento pressione.
  - Avvio immediato di una sequenza di irrigazione manuale.

1.4 Analisi delle soluzioni alternative

Durante la progettazione sono state valutate diverse soluzioni alternative, successivamente scartate per ottimizzare l'impiego delle risorse e semplificare l'architettura.

Controllo dei servomotori

- **Alternativa**: libreria Servo.h.
- **Scartata perché**: maggiore occupazione di memoria (~1 kB), uso di interrupt con possibile jitter.
- **Soluzione adottata**: PWM diretto via Timer1 → maggiore precisione e minor overhead.

Gestione dell'orologio

- **Alternativa**: modulo RTC esterno (es. DS3231).
- **Scartata perché**: hardware aggiuntivo, maggiore complessità, consumo di 2 pin I2C.
- **Soluzione adottata**: orologio software tramite millis(), precisione sufficiente per l'applicazione.

Architettura software

- **Alternativa**: multitasking con RTOS o librerie scheduler.
- **Scartata perché**: incremento consumo RAM (~1–2 kB), complessità non necessaria.

- **Soluzione adottata**: ciclo sequenziale semplice, sufficiente per il comportamento richiesto.

Gestione del pulsante

- **Alternativa**: gestione tramite interrupt esterno (INT0).
- **Scartata perché**: maggiore complessità, gestione concorrente più delicata.
- **Soluzione adottata**: polling con debounce software, adeguato per l'uso previsto.

2. Sintesi

2.1 Sintesi componente ME1 — Gestione PWM per servomotori

Realizzazione:
Il componente ME1 è realizzato tramite configurazione diretta del Timer1 di ATmega328P in modalità Fast PWM, con TOP=ICR1 e uscita non-inverting su OC1A e OC1B (pin 9 e 10).

Comportamento:
Genera segnali PWM a 50 Hz con velocità variabile per controllare l'angolazione dei due servomotori. La funzione angoloServo() calcola dinamicamente il valore di OCR1A/B in funzione dell'angolo richiesto.

Accorgimenti adottati:

- Evitato l'uso della libreria Servo.h per ridurre l'overhead e migliorare la stabilità del PWM.
- Utilizzato un mapping lineare personalizzato per ottenere una risposta angolare calibrata sui servo in uso.

Test effettuato:
Verificato il corretto movimento dei servo su tutto l'arco da irrigare (0°–180°), con tempi di risposta rapportati alla quantità di umidità presente in ogni zona.

2.2 Sintesi componente ME2 — Lettura sensori di umidità

Realizzazione:
Il componente ME2 sfrutta il convertitore ADC a 10 bit integrato in ATmega328P. I canali A0–A4 vengono letti in sequenza tramite la funzione leggiADC().

Comportamento:
Acquisisce il livello di umidità del terreno per ciascuna delle 5 zone. I valori letti vengono memorizzati nell'array hum[5].

Accorgimenti adottati:

- Configurato ADC con riferimento AVcc per massimizzare la stabilità della lettura.
- Inserita una funzione di lettura sequenziale per ridurre la latenza tra i canali.

Test effettuato:
Verificata la lettura stabile e ripetibile su tutti i canali, con variazioni di tensione simulate.

2.3 Sintesi componente ME3 — Orologio interno (software)

Realizzazione:
Il componente ME3 implementa un orologio software basato sulla funzione millis()(Timer0 integrato di Arduino Uno), con aggiornamento delle variabili secondi, minuti e ore.

Comportamento:
Tiene traccia dell'orario corrente, aggiornando le variabili ogni secondo. Fornisce il riferimento temporale per la gestione delle irrigazioni automatiche.

Accorgimenti adottati:

- Gestito overflow di millis() per evitare errori a lungo termine.
- Implementata la logica di rollover sessadecimale di secondi/minuti/ore in modo robusto.

Test effettuato:
Monitorato il drift orario su più ore. Precisione ritenuta adeguata per l'applicazione.

2.4 Sintesi componente ME4 — Logica di irrigazione

Realizzazione:
La logica di irrigazione è implementata nel loop principale. Controlla lo stato dell'orologio e del pulsante e decide quando attivare la sequenza di irrigazione.

Comportamento:
Quando scatta un orario programmato o viene premuto il pulsante, esegue:

- Lettura umidità.
- Calcolo velocità servo.
- Movimentazione selettiva dei servo.

Accorgimenti adottati:

- Implementato meccanismo orarioAttivato[] per evitare ripetizioni durante lo stesso minuto.
- Gestito movimento rapido per le zone umide per garantire fluidità del ciclo.

Test effettuato:
Verificata corretta attivazione sia in automatico che in manuale. Testate sequenze con zone alternate secche/umide per verificare che in qualsiasi combinazione i servo continuino a

funzionare al meglio.

2.5 Sintesi componente ME5 — Gestione pulsante manuale

Realizzazione:
Il pulsante è collegato a PD2, gestito tramite polling e debounce software.

Comportamento:
Se il pulsante viene premuto, avvia una sequenza di irrigazione manuale completa.

Accorgimenti adottati:

- Implementato debounce con ultimoCambioStato e debounceDelay per evitare falsi trigger.
- Gestito correttamente il rilascio del pulsante per consentire nuove attivazioni.

Test effettuato:
Verificata risposta corretta anche in caso di rimbalzi e pressione prolungata.

3. Sistema complessivo

Il sistema realizzato implementa un prototipo funzionale per l'irrigazione smart di una porzione di terreno su piattaforma Arduino, con interfaccia utente tramite porta seriale e attivazione manuale tramite pulsante.

Il progetto completo è allegato in formato .ino nella repository gitHub (repository) ed è compatibile con la piattaforma di simulazione Wokwi.

Il codice completo è stato sviluppato in linguaggio C++ per Arduino, sfruttando direttamente le risorse hardware del microcontrollore.

3.1 Manuale utilizzo

Dati da inserire

All'avvio il sistema richiede di inserire tramite porta seriale:

1. Data corrente (formato: gg/mm/aaaa).
2. Ora corrente (formato: hh:mm).
3. Numero di orari di irrigazione da programmare (da 0 a 16).
4. Per ciascun orario programmato:
    - Orario di irrigazione (formato: hh:mm).

Funzionamento automatico

- Il sistema mantiene un orologio interno software basato su millis(), aggiornato ogni secondo.
- Allo scoccare di ogni minuto controlla se l'orario corrente corrisponde a uno degli orari di irrigazione programmati.
- Se sì:
    - Esegue la lettura dei 5 sensori di umidità.
    - Calcola per ciascuna zona la velocità di movimento dei servo.
    - Esegue la sequenza di irrigazione:

        - Per zone secche → irrigazione lenta con movimento servo.
        - Per zone umide → movimento rapido per saltare la zona.
    - Al termine riporta i servo in posizione di partenza.

Funzionamento manuale

- Premendo il pulsante, il sistema avvia una irrigazione manuale completa indipendentemente dall'orario.
- La sequenza manuale segue lo stesso comportamento di quella automatica.
- Il pulsante è gestito con debounce software per evitare falsi trigger.

Risultati attesi

- In seriale viene mostrato:
    - L'orario corrente (viene mostrato ogni minuto).
    - L'attivazione degli orari di irrigazione.
    - L'esito dell'irrigazione per ciascuna zona (irrigata o saltata).
    - La conferma di avvio e completamento dell'irrigazione manuale.
- I servomotori eseguono i movimenti previsti:
    - Rotazione selettiva per ogni zona, servo1 da 0 a 90 gradi mentre servo2 da 91 a 180.
    - Ritorno alla posizione iniziale al termine.

Accorgimenti d'uso

- Si raccomanda di impostare correttamente i valori di soglia SOGLIA_MIN e SOGLIA_MAX per adattarli ai sensori fisici utilizzati.
- I potenziometri possono essere usati in fase di test per simulare il livello di umidità.
- Il sistema funziona anche senza RTC esterno, ma potrebbe accumulare una piccola variazione oraria nel lungo periodo.

3.2 Considerazioni finali

Il progetto ha consentito di realizzare un sistema di irrigazione completamente autonomo, semplice da configurare e a basso costo.

Sono stati sfruttati in modo diretto:

- il Timer1 per la generazione del PWM dei servo,
- il convertitore ADC per la lettura dei sensori,
- un orologio software auto-aggiornato.

L'architettura a ciclo unico sequenziale ha permesso di evitare la complessità di multitasking o di interrupt, mantenendo un comportamento robusto e prevedibile.

Sviluppi futuri possibili

- Integrazione di un RTC hardware per una maggiore precisione temporale.
- Implementazione di una memorizzazione permanente degli orari.
- Aggiunta di un'interfaccia utente più evoluta.
- Supporto per un numero maggiore di zone.