COSTANZO MARTINO

Digital Twin per la Produzione di PCB

Simulazione, IoT e Dashboard Web

Descrizione

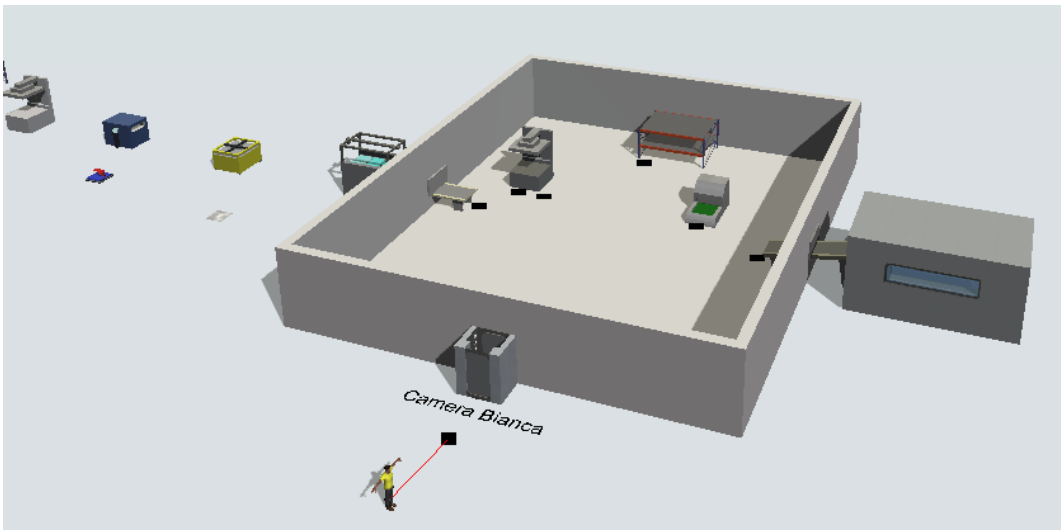
Progetto di Digital Twin per la simulazione di un processo industriale di produzione di circuiti stampati (PCB).

Le macchine sono modellate in SketchUp e integrate in FlexSim, con sincronizzazione dei dati tramite sensori IoT e API PHP.

La dashboard web consente il monitoraggio dei KPI in tempo reale.

Tecnologie

FlexSim, SketchUp, Arduino Nano 33 IoT, PHP, MySQL, JavaScript, HTML, CSS, Sensori IoT



Introduzione

Descrizione del contesto e degli obiettivi del progetto

Il progetto ha l'obiettivo di realizzare un Digital Twin per la simulazione del processo industriale di produzione di circuiti stampati (PCB).

L'intero flusso produttivo, dall'approvvigionamento delle materie prime allo stoccaggio dei PCB, è modellato in FlexSim per analizzare il comportamento del sistema e supportare l'ottimizzazione dei processi.

Il Digital Twin è integrato con un'infrastruttura IoT che consente il monitoraggio in tempo reale dei parametri ambientali critici, come temperatura, umidità e qualità dell'aria.

Gli obiettivi principali del progetto sono:

- Simulare il processo produttivo dei PCB
- Monitorare le condizioni ambientali
- Integrare dati reali tramite sensori IoT
- Rendere i dati disponibili tramite API e dashboard web

Requisiti funzionali

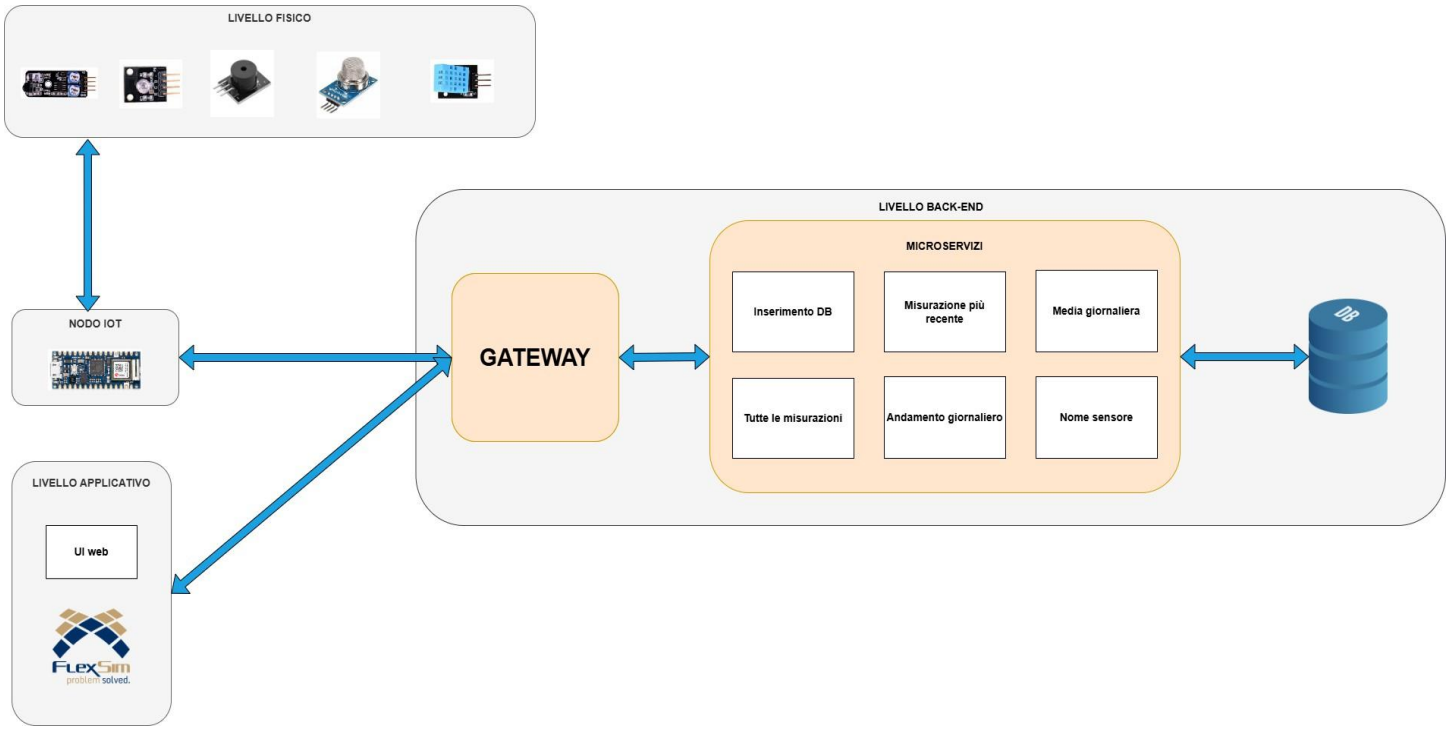
Il sistema prevede l'utilizzo di sensori e attuatori per il monitoraggio dell'ambiente produttivo e la gestione degli allarmi.

- Sensori di movimento: rilevano la presenza degli operatori per attivare specifici processi automatizzati
- Sensori di temperatura: monitorano la temperatura e attivano allarmi in caso di superamento delle soglie
- Sensori di umidità: controllano i livelli di umidità ambientale
- Sensori di CO: misurano la concentrazione di monossido di carbonio
- LED: segnalazione visiva dello stato dei sensori
- Buzzer: allarme sonoro in caso di condizioni critiche
- Microcontrollore: gestisce i sensori e invia i dati al database

ARCHITETTURA COMPLESSIVA

Descrizione dell'architettura

È stata definita un'architettura a microservizi con gateway.



Componenti principali dell'architettura:

Livello fisico:

- Sensori: Dispositivi che monitorano parametri fisici come temperatura, umidità, movimento e monossido di carbonio.
- Attuatori: Componenti che eseguono azioni fisiche in risposta ai comandi provenienti dalla simulazione.

Nodo IoT:

- Arduino Nano 33 IoT: Raccoglie dati dai sensori collegati, li elabora e li invia al gateway.

Livello applicativo:

- FlexSim: Software di simulazione 3D che rappresenta il sistema di produzione, visualizzando e analizzando i dati in tempo reale.
- Web dashboard: pagina HTML/PHP che mostra i dati monitorati dai sensori.

Livello Backend:

- Gateway: assicura che le richieste siano inoltrate al microservizio appropriato.
- Microservizi: gestiscono la logica tra Arduino Nano e FlexSim.
- Database relazionale: Il sistema utilizza un database relazionale basato su SQL ospitato su un server Altervista per la gestione dei dati raccolti dai sensori e utilizzati nell'ambito della simulazione.

Descrizione dei microservizi

Il sistema utilizza un'architettura basata su microservizi REST per la gestione e l'esposizione dei dati provenienti dai sensori IoT.

I servizi comunicano con l'esterno tramite un gateway API e utilizzano un database MySQL per la persistenza delle misurazioni.

Ogni microservizio ha una responsabilità specifica ed espone endpoint dedicati.

Microservizi di acquisizione dati:

insertMisurazione.php
Gestisce l'inserimento delle misurazioni nel database.

- Riceve dati in formato JSON
- Valida il payload
- Inserisce le misurazioni (temperatura, umidità, CO, movimento)
- Associa timestamp e sensore
- Restituisce una risposta JSON con l'esito dell'operazione

Microservizi di consultazione dati

getLatestMeasurement.php
Recupera l'ultima misurazione disponibile per un sensore specifico.

- Parametri via query string
- Query sul database
- Risposta JSON con valore o messaggio di errore

getMisurazioni.php
Restituisce tutte le misurazioni presenti nel database.

- Query completa sulla tabella *Misurazioni*
- Risposta in formato array associativo JSON

getAndamentoGiornaliero.php
Fornisce le serie temporali delle ultime 24 ore.

- Estrae dati per temperatura, umidità e CO
- Include timestamp e valori
- Utilizzato per aggiornare i grafici della dashboard

getNomeSensore.php
Restituisce il nome del sensore a partire dall'ID.

- Query per ID sensore
- Risposta JSON per il frontend

getMisurazioniOggi.php
Calcola le medie giornaliere delle misurazioni ambientali.

- Aggregazione dei dati del giorno corrente
- Calcolo delle medie
- Risposta JSON per la dashboard

Web UI – Dashboard

dashboard.php

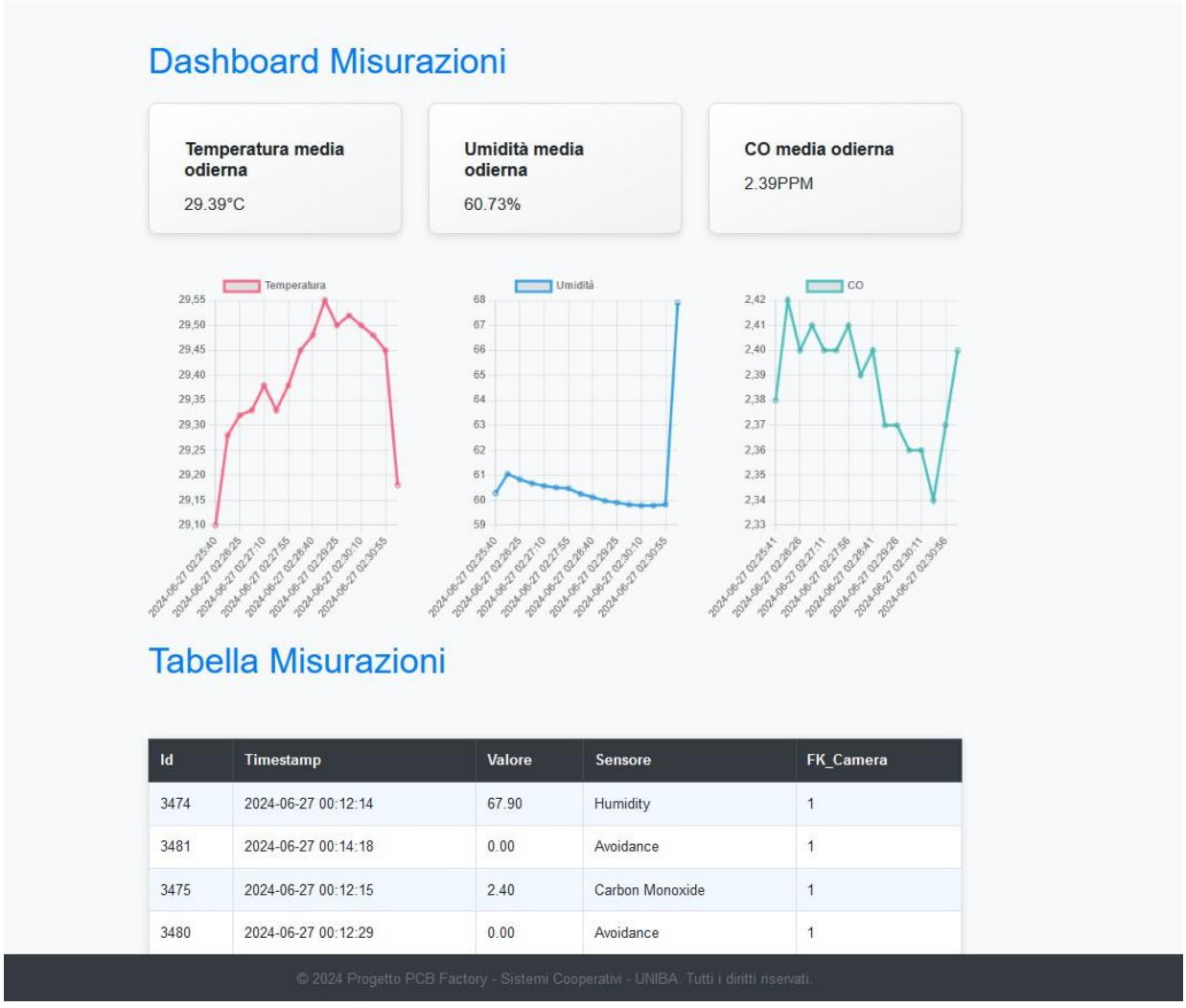
La dashboard web fornisce una visualizzazione completa dei dati raccolti.

Funzionalità principali:

- Visualizzazione delle medie giornaliere tramite card informative
- Grafici interattivi per l'andamento di:
  - Temperatura
  - Umidità
  - Monossido di carbonio
- Tabella dettagliata con:
  - Timestamp
  - Valori
  - Sensore
  - Camera di riferimento

La dashboard consente un monitoraggio continuo e una rapida interpretazione delle condizioni ambientali

https://imarlito.altervista.org/iMarlito/SistemiCooperativi/dashboard.php



Il file gateway.php implementa il Gateway Pattern e funge da punto di accesso centralizzato per la gestione delle operazioni sui dati dei sensori.

Attraverso una classe dedicata, il gateway:

- Gestisce la connessione al database
- Incapsula le query SQL
- Fornisce metodi per:
    - Inserire nuove misurazioni
    - Recuperare l'ultima misurazione di un sensore
    - Ottenere tutte le misurazioni

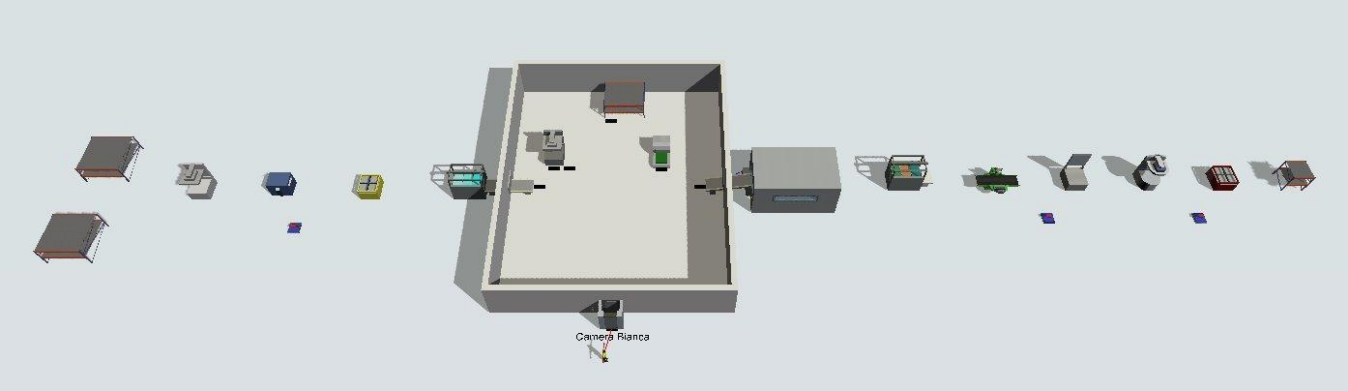
Questo approccio migliora:

- Modularità del codice
- Manutenibilità
- Riutilizzabilità
- Separazione delle responsabilità

La centralizzazione delle operazioni di database riduce la duplicazione del codice, facilita l'estensione dei servizi e consente una gestione più sicura e controllata degli errori.

PROGETTAZIONE E IMPLEMENTAZIONE DEI MODELLI DI SIMULAZIONE

Layout dei modelli di simulazione



Il layout include i seguenti elementi:

Rack

1. RackLastreRame: Contiene l'item LastreRame.
2. RackLastreVetroresina: Contiene l'item LastreVetroresina.
3. RackNastriPellicola: Contiene l'item NastriPellicola.
4. RackPCBFinite: Contiene l'item LastreRame, con applicata la ShapeFrame PCB.

Processor

5. IspezioneRaggiX: Macchinario che scansiona i WaferGrezzo con raggi X.
6. Foratura: Macchina che fora i WaferGrezzo, restituendo WaferForato.
7. BagniSuccessivi: Macchina che lava i WaferForato dai residui di rame e vetroresina post foratura.
8. IncisioneCircuitiLaserUV: Macchina che incide tramite un laser UV la pellicola fotosensibile che non copre i circuiti su WaferPellicola, restituendo WaferInciso.
9. RimozionePellicolaSuperflua: Macchina che rimuove la pellicola superflua su WaferInciso, restituendo WaferPellicolaRimossa.
10. IspezioneAOI (Automatic Optical Inspection): Macchina che trasporta i WaferConCircuiti, sui quali verranno effettuati controlli visivi.
11. ApplicazioneResina: Macchina che applica una resina verde al WaferConCircuiti, restituendo WaferCompleto.
12. IspezioneConducibilità: Macchina che verifica che il WaferCompleto abbia la corretta conducibilità.
13. Taglio: Macchina che taglia il WaferCompleto in singole PCB.

Combiner

14. Pressa: Macchina che pressa due LastreRame con una LastreVetroresina, restituendo un WaferGrezzo.
15. ApplicazionePellicola: Macchina che applica, all'interno della Camera Bianca, NastriPellicola sul WaferForato, restituendo WaferPellicola.

Separator

16. ImmersioneInAcido: Macchina che elimina rame e pellicola da WaferPellicolaRimossa, restituendo WaferConCircuiti.

Sink

17. WaferDifettosi: Sink che elimina i wafer difettosi identificati da IspezioneRaggiX.
18. LastreDifettose: Sink che elimina le lastre che non superano IspezioneAOI.
19. WaferNonConduc: Sink che elimina le lastre con problemi di conducibilità dopo la fase di IspezioneConducibilità.

Operator

20. Operator: Operatore che lavora all'interno della Camera Bianca.

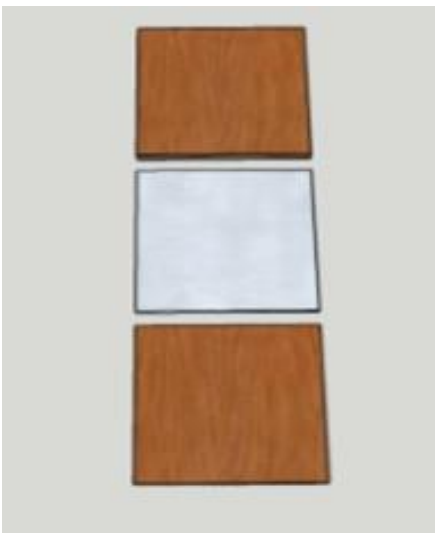
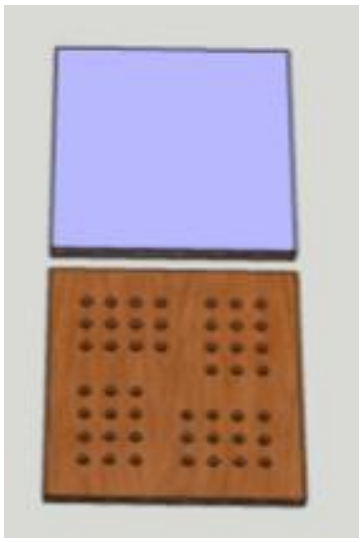
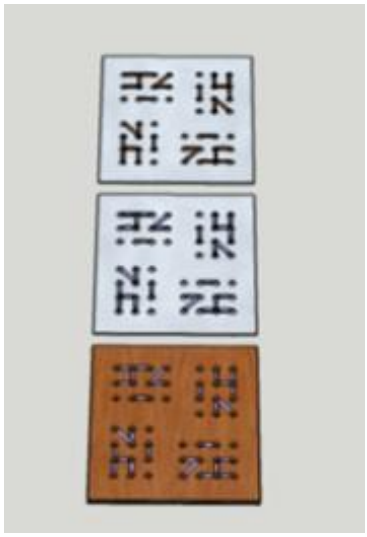
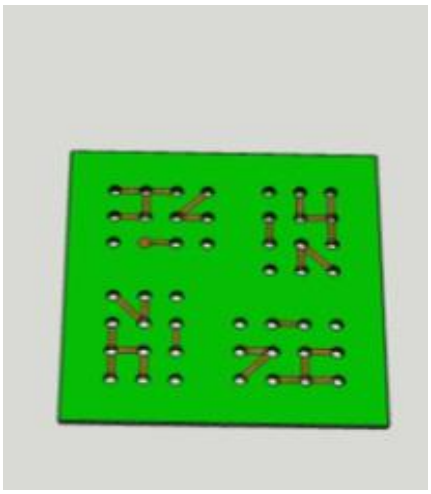
Item utilizzati

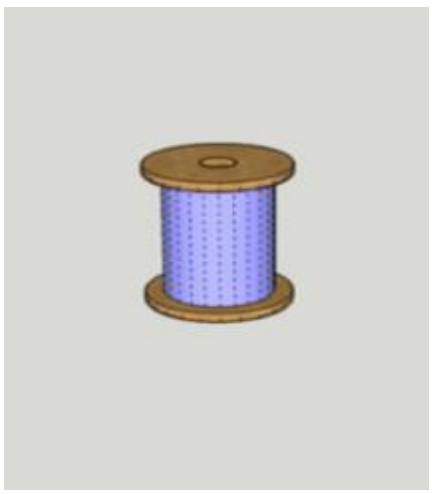
21. LastraRame: Lastra di rame, ha 10 ShapeFrame:

21. LastraRame.
22. WaferGrezzo.
23. WaferForato.
24. WaferPellicola.
25. WaferInciso.
26. WaferPellicolaRimossa.
27. WaferRameRimosso.
28. WaferConCircuiti.
29. WaferCompleto.
30. PCB.

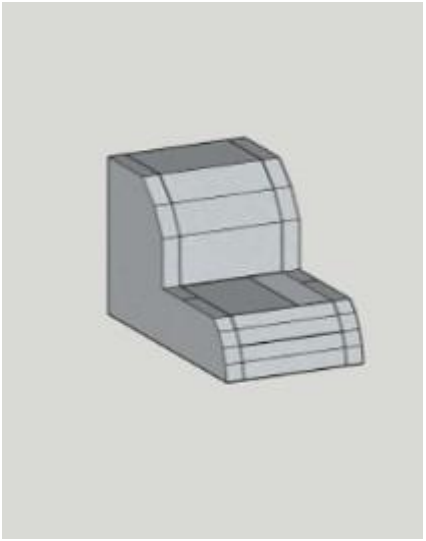
31. LastraVetroresina: Lastra di vetroresina.
32. NastroPellicola: Bobina di pellicola.

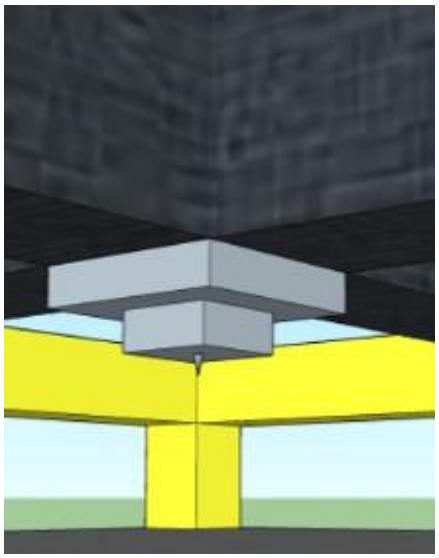
Nel layout non sono presenti Source perché gli elementi vengono generati tramite il process flow.
Ventitré dei trentadue elementi elencati sono stati modellati in 3D utilizzando il software di modellazione SketchUp. Alcuni di questi elementi sono costituiti da più parti modellate separatamente, per garantire una simulazione accurata anche dei movimenti delle macchine, rendendo il tutto più realistico possibile.
Elementi modellati in 3D:

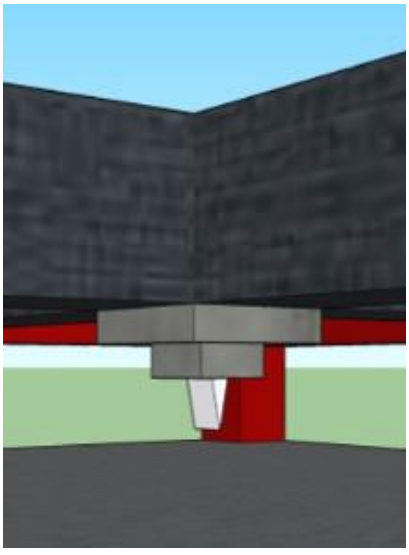





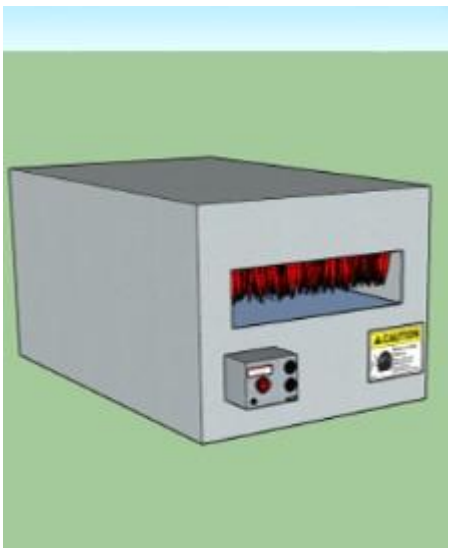












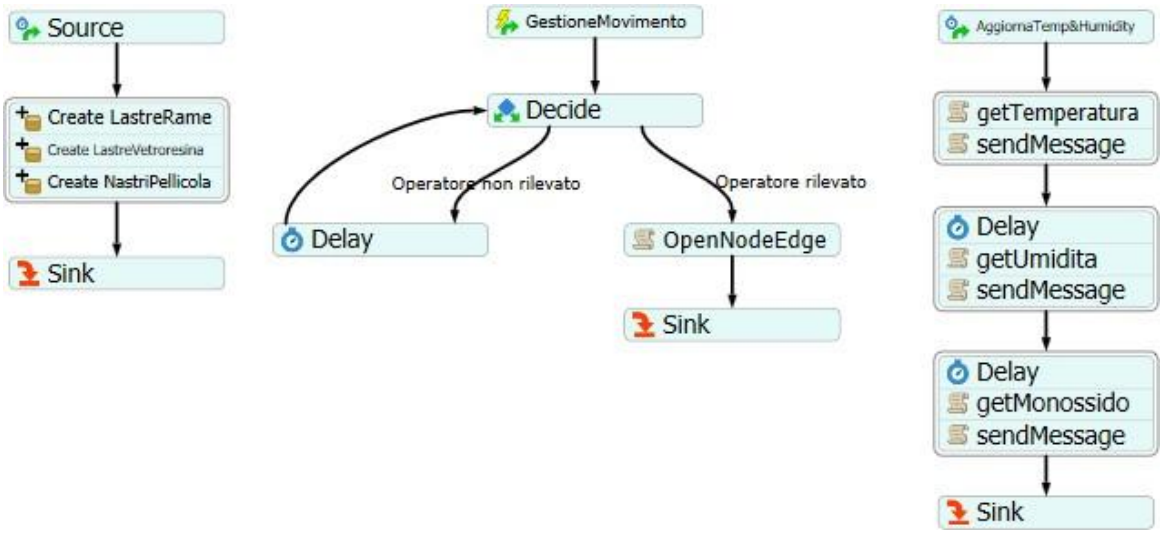
Regole di comportamento e logiche implementate

La simulazione segue un percorso sequenziale, con le macchine che operano in modo continuo. L'operatore presente nel layout esegue i suoi compiti seguendo percorsi predefiniti, definiti tramite delle Travel Network.

Ogni macchina che comporta un cambiamento fisico sull'item in lavorazione, modifica nome e ShapeFrame dell'oggetto tramite un apposito trigger. Gli elementi che escono dalle macchine vengono posizionati in specifiche aree all'interno della macchina o su superfici circostanti, al fine di rendere la simulazione più realistica.

L'accesso alla Camera Bianca è consentito all'operatore solo quando quest'ultimo si trova di fronte all'ingresso della stessa.

Per ottimizzare la gestione nel nostro progetto abbiamo deciso di utilizzare il Process Flow.



Il Process Flow realizzato presenta tre generatori di token:

- Il generatore *Source* produce regolarmente gli item LastraRame, LastraVetroresina e NastroPellicola all'interno dei rispettivi rack.
- Il generatore *GestioneMovimento* crea un token non appena l'operatore raggiunge e si ferma sul nodo NN1 della travel network. All'interno di Decide viene eseguita una richiesta HTTP getMovimento al database, che può restituire 0 o 1. Se il valore è 0, il sensore di movimento non ha rilevato alcun operatore e viene effettuata una nuova richiesta dopo un determinato intervallo di tempo. Se il valore è 1, il nodo NN1 viene aperto per permettere all'operatore di entrare nell'anticamera, vestirsi in maniera opportuna, disinfettarsi e iniziare a lavorare.
- Il generatore *AggiornaTemp&Humidity* crea un token a intervalli regolari per eseguire tre richieste HTTP, ciascuna distanziata per evitare il sovraccarico delle richieste. Dopo ogni richiesta, viene inviato un messaggio alla Billboard corrispondente per permettere allo statistics collector di generare il time plot.

COMUNICAZIONE TRA LA SIMULAZIONE E IL MONDO REALE

La comunicazione tra FlexSim e l'infrastruttura IoT consente di integrare il Digital Twin con dati reali provenienti dai sensori Arduino.

Questo collegamento permette di sincronizzare il modello di simulazione con le condizioni ambientali effettive del sito produttivo.

L'architettura è basata su microservizi REST che gestiscono la raccolta, l'elaborazione e la visualizzazione dei dati.

Tecnologie utilizzate

La comunicazione tra la simulazione in FlexSim e il mondo reale è realizzata attraverso un'architettura a microservizi. Le principali tecnologie utilizzate includono:

- API REST: Utilizzate per lo scambio sincrono dei dati tra i componenti del sistema..
- JSON:Formato utilizzato per serializzare i dati dei sensori in modo leggero e strutturato.
- PHP: Linguaggio utilizzato per sviluppare i microservizi backend e l'API Gateway.
- MySQL: Database per la persistenza dei dati.

Messaggi scambiati

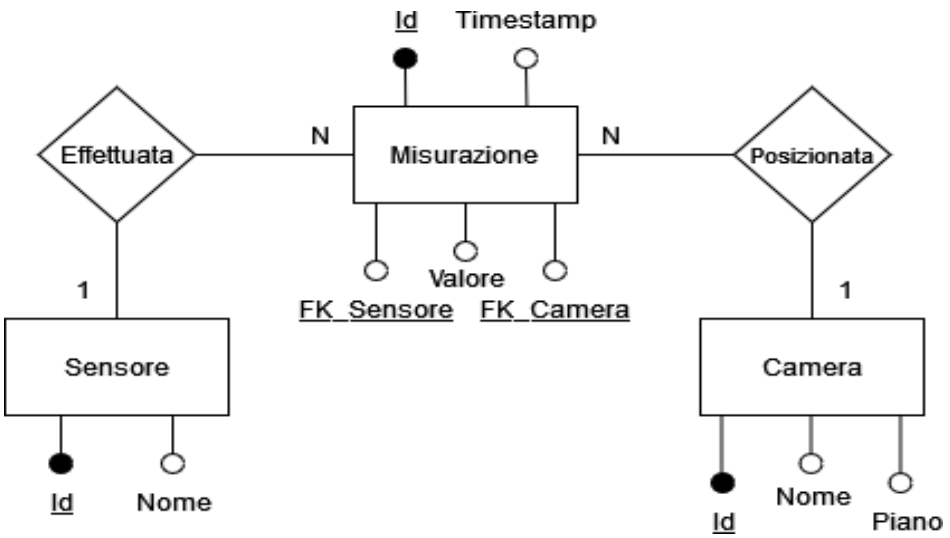
I sensori Arduino raccolgono i dati dal campo (ad esempio, temperatura, umidità, monossido di carbonio) e li inviano ai microservizi tramite richieste HTTP POST.

I messaggi scambiati sono:

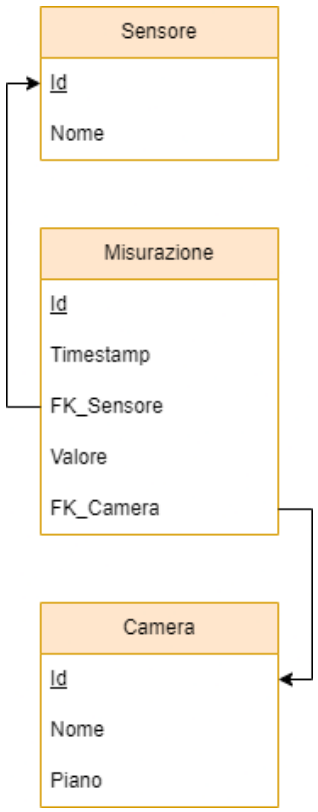
- Scrittura dati dei sensori: I messaggi contengono le letture provenienti dai sensori installati nel mondo reale.
- Il microcontrollore acquisisce i valori rilevati e li invia al gateway tramite protocollo HTTP.
- Il payload è strutturato in formato JSON e viene successivamente memorizzato nel database.
- Richieste di lettura dal database: FlexSim invia richieste al database attraverso il gateway per leggere i dati raccolti dai sensori. Queste richieste possono essere eseguite periodicamente o in risposta a determinati eventi o condizioni operative.
- Dati di risposta dal database: Il database restituisce i dati a FlexSim.
- Le informazioni ricevute vengono utilizzate per aggiornare la simulazione e attivare scenari specifici.

DATABASE RELAZIONALE CONDIVISO

Modello EER



Modello Relazionale



Dizionari

- Sensore
  - Id: Identificativo univoco del sensore (INT, AUTO_INCREMENT, PRIMARY KEY)
  - Nome: Nome del sensore (VARCHAR(255), NOT NULL)
- Camera
  - Id: Identificativo univoco della camera (INT, AUTO_INCREMENT, PRIMARY KEY)
  - Nome: Nome della camera (VARCHAR(255), NOT NULL)
  - Piano: Numero del piano in cui si trova la camera (INT, NOT NULL)
- Misurazione
  - Id: Identificativo univoco della misurazione (INT, AUTO_INCREMENT, PRIMARY KEY)
  - Timestamp: Data e ora della misurazione (DATETIME, NOT NULL,DEFAULT CURRENT_TIMESTAMP)
  - Valore: Valore della misurazione (DECIMAL(5, 2))
  - FK_Sensore: Identificativo del sensore associato alla misurazione (INT, NOT NULL, FOREIGN KEY che fa riferimento a Sensore(Id))
  - FK_Camera: Identificativo della camera associata alla misurazione (INT, NOT NULL, FOREIGN KEY che fa riferimento a Camera(Id))

GESTIONE DEGLI EVENTI DAL CAMPO

Gli eventi dal campo sono rilevati attraverso i sensori fisici.

Sensori utilizzati

Sono stati utilizzati vari sensori e attuatori compatibili con Arduino Nano 33 IoT per monitorare e controllare le condizioni ambientali all'interno della camera bianca:

Avoidance Module

Funzione: Il sensore viene utilizzato per rilevare la presenza dell'operatore all'arrivo nella cabina dell'anticamera.
Quando viene rilevato un movimento:

1. Arduino acquisisce il dato dal sensore
2. Il microcontrollore invia il valore al gateway tramite HTTP
3. Il gateway memorizza il dato nel database
4. FlexSim riceve l'informazione e simula:
    - Il movimento dell'operatore
    - L'apertura delle porte
    - L'avvio del ciclo di pulizia

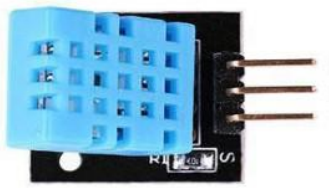
Questo processo garantisce che l'operatore entri nella camera bianca senza introdurre contaminanti.

Tecnologia: Il sensore è composto da:

- Un LED a infrarossi (IR) che emette radiazione
- Un ricevitore IR che rileva la radiazione riflessa

Quando un operatore si avvicina, la radiazione IR viene riflessa dal corpo e captata dal ricevitore, che segnala la presenza attivando il processo di pulizia.

Temperature and Humidity Module (DHT11)



Funzione: Il sensore DHT11 viene utilizzato per monitorare temperatura e umidità all'interno del sito produttivo, assicurando che i parametri ambientali rimangano entro i limiti specificati durante il processo di produzione.

Il controllo di questi valori è fondamentale per prevenire difetti nei PCB e garantire un ambiente di lavoro stabile.

Tecnologia: I DHT11 è un sensore digitale che integra:

- Un elemento resistivo per la rilevazione dell'umidità
- Un termistore NTC (Negative Temperature Coefficient) per la misurazione della temperatura
- Un microcontrollore interno per l'elaborazione dei dati

I valori vengono forniti in formato digitale e possono essere facilmente acquisiti da Arduino Nano 33 IoT.

Carbon Monoxide Sensor (MQ9)

Funzione: Il sensore MQ-9 viene utilizzato per monitorare i livelli di monossido di carbonio (CO) all'interno della camera bianca.

Il CO è un gas incolore e pericoloso, pertanto la sua concentrazione deve essere costantemente controllata per garantire la sicurezza degli operatori e il corretto funzionamento dell'ambiente produttivo.

Tecnologia: Il sensore MQ-9 rileva la presenza di gas tramite la variazione della resistenza interna in presenza di monossido di carbonio.

Il funzionamento si basa su:

- Elettrodi riscaldati
- Reazioni chimiche con il gas rilevato
- Misurazione della corrente elettrica risultante

La resistenza del sensore varia in base alla concentrazione del gas, permettendo ad Arduino di stimare i livelli di CO presenti nell'aria.

LED RGB

Funzione: Il LED RGB viene utilizzato per fornire indicazioni visive immediate sullo stato dei principali parametri ambientali all'interno della camera bianca:

- Verde – Livello di monossido di carbonio entro i limiti di sicurezza
- Rosso – Superamento della soglia di CO, possibile situazione di pericolo
- Blu – Presenza di un operatore rilevata dal sensore di prossimità

Questo sistema consente una segnalazione chiara e immediata delle condizioni operative.

Tecnologia: Il LED RGB è composto da tre diodi luminosi:

- Rosso
- Verde
- Blu

La combinazione delle tre componenti permette di generare diversi colori in base allo stato del sistema, controllati tramite Arduino Nano 33 IoT.

Passive Buzzer (Cicalino)

Funzione: Il buzzer passivo viene utilizzato per fornire allarmi acustici immediati quando i parametri ambientali della camera bianca superano le soglie di sicurezza definite (ad esempio temperatura, umidità o livelli di CO).

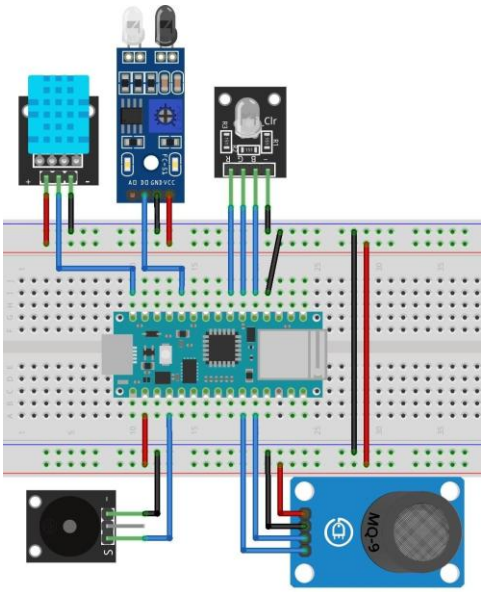
Il segnale sonoro consente agli operatori di individuare rapidamente situazioni critiche anche in assenza di segnalazioni visive.

Tecnologia: Il buzzer utilizzato è di tipo passivo, privo di oscillatore interno.

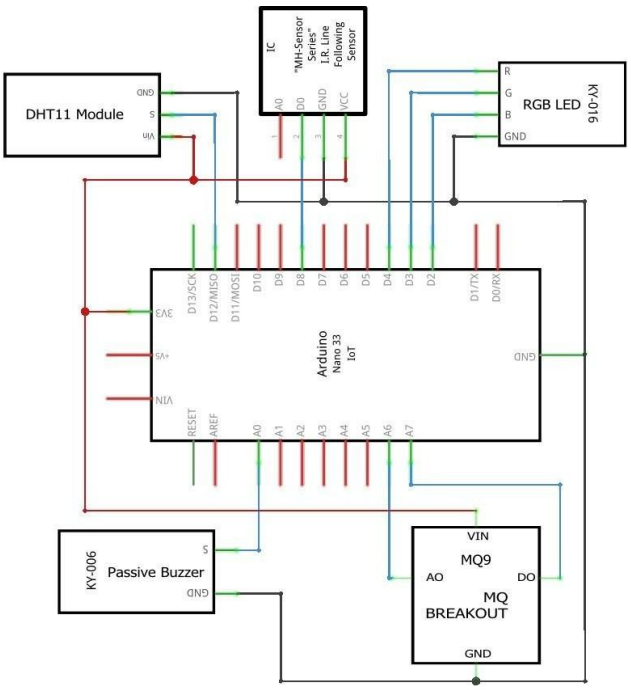
Per emettere un suono, richiede un segnale ad onda quadra con una frequenza compresa tra 2 kHz e 5 kHz, generato direttamente da Arduino Nano 33 IoT.

Nodi IoT che interfacciano i sensori

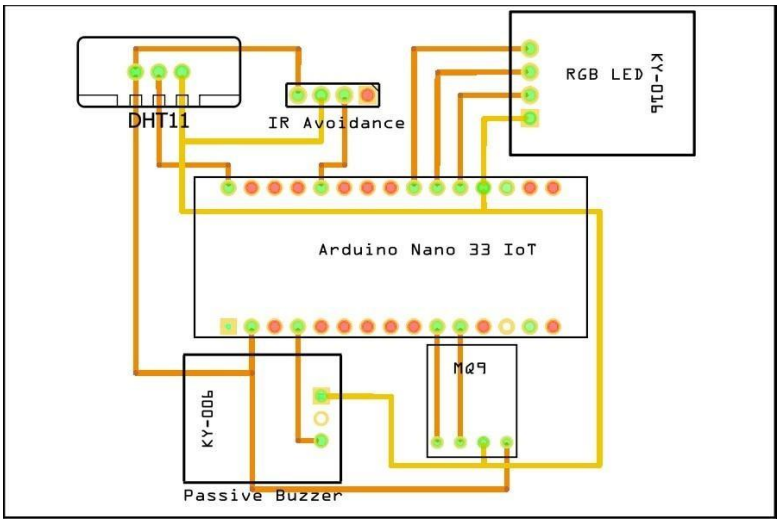
Breadboard



Schema elettrico



PCB



Sketch su microcontrollore

Descrizione Generale

Il programma sviluppato per Arduino Nano 33 IoT gestisce la raccolta, l'elaborazione e la trasmissione dei dati provenienti dai sensori installati all'interno del sito produttivo verso un server remoto.
Il sistema:

- Legge i valori dei sensori a intervalli predefiniti
- Calcola la media delle misurazioni per ridurre il rumore e migliorare l'accuratezza dei dati
- Verifica il superamento delle soglie ambientali
- Attiva segnali di allarme visivi e acustici

- Invia i dati al database tramite richieste HTTP

Controllo delle soglie

Il microcontrollore monitora costantemente:

- Temperatura e umidità
  Se i valori superano le soglie impostate, viene attivato un allarme sonoro tramite buzzer.
- Monossido di carbonio (CO)
  - CO sottosoglia → LED verde
  - CO sopra soglia → LED rosso

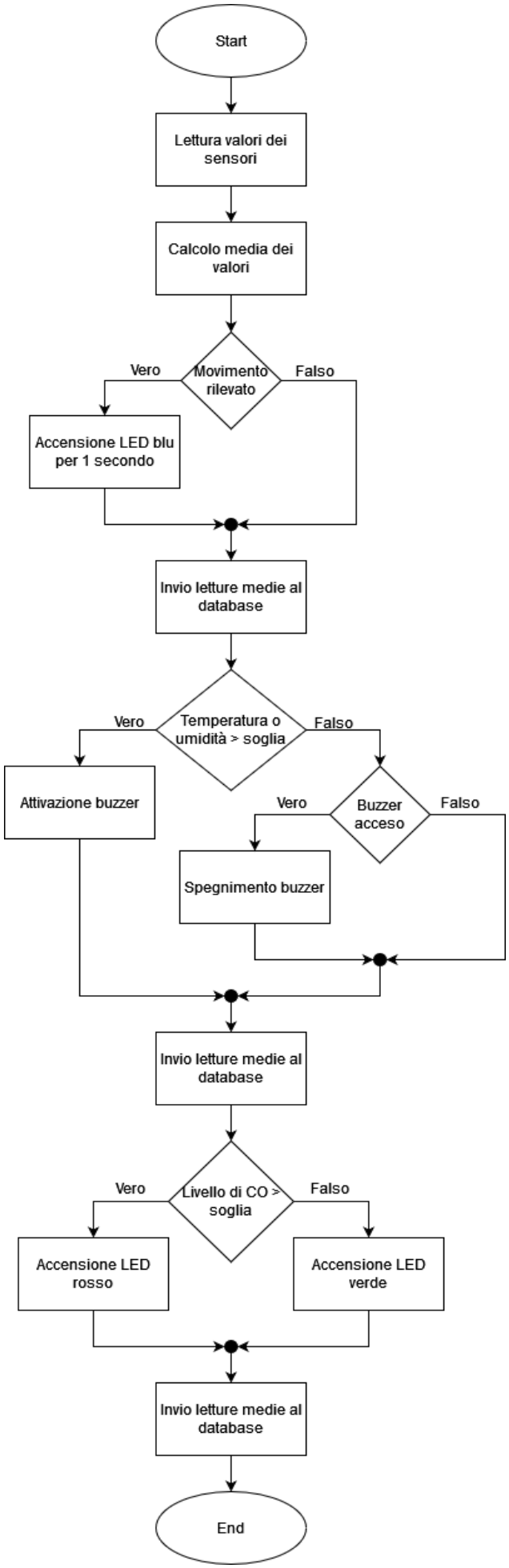
Questi segnali permettono una rapida identificazione delle condizioni critiche all'interno della camera bianca.

Trasmissione dei dati

Ogni sensore invia le proprie letture in modo indipendente al gateway API, che si occupa della memorizzazione nel database remoto.

I dati vengono trasmessi in formato JSON tramite protocollo HTTP, consentendo l'integrazione con:

- Microservizi PHP
- Database MySQL
- Dashboard web
- Simulazione FlexSim



Flow chart Lettura e invio misurazioni

Architettura software – Arduino

Lo sketch per Arduino Nano 33 IoT gestisce:

- Connessione WiFi
- Lettura dei sensori
- Controllo attuatori
- Invio dati al server tramite API REST

Librerie principali

- WiFiNINA – Gestione connessione WiFi
- ArduinoHttpClient – Invio richieste HTTP
- ArduinoJson – Serializzazione JSON
- DHT – Lettura sensore temperatura/umidità
- Adafruit_Sensor – Astrazione sensori
- WiFiConnection – Gestione credenziali (custom)

Configurazione hardware

Il microcontrollore utilizza:

- Sensore di prossimità (Avoidance)
- Sensore DHT11
- Sensore CO (MQ-9)
- LED RGB
- Buzzer passivo

Ogni componente è collegato a pin digitali o analogici dedicati.

Logica di funzionamento

Il sistema:

1. Legge i sensori a intervalli predefiniti
2. Calcola la media dei valori
3. Controlla le soglie
4. Attiva LED e buzzer
5. Invia i dati al server in JSON

Funzione	Scopo
Setup()	Inizializzazione WiFi, pin e sensori
Loop	Gestione ciclo di lettura e invio dati
readAvoidanceSensor()	Rileva presenza operatore
readDhtSensor()	Legge temperatura/umidità
readCarbonSensor()	Legge CO e aggiorna LED
sendData()	Invia JSON al server
mapFloat()	Conversione valori analogici

Event-driven data pipeline

Il sistema implementa un flusso di gestione degli eventi basato su architettura a microservizi e comunicazione API REST:

1. Data acquisition
   I sensori acquisiscono i dati a intervalli configurati:
   - DHT11: 5s
   - MQ9: 3s
   - Avoidance Module: 1s
2. Data preprocessing
   I valori vengono filtrati tramite media mobile per ridurre rumore e anomalie:
   - DHT11: finestra di 20s
   - MQ9: finestra di 12s
   - Avoidance Module: tempo reale
3. API data ingestion
   Il microcontrollore, connesso via WiFi, invia i dati in formato JSON al backend tramite API REST.
4. Backend processing & persistence
   Il gateway applicativo riceve i payload e inoltra i dati al microservizio di persistenza, che li salva nel database.
5. Simulation data sync
   FlexSim accede ai dati tramite endpoint API per sincronizzare lo stato della simulazione con i valori real-time.
6. Scenario execution
   La simulazione attiva scenari dinamici in base alle condizioni rilevate, abilitando comportamenti adattivi del Digital Twin.

TEST E VALIDAZIONE

Per garantire affidabilità, robustezza e corretto funzionamento dell'architettura a microservizi e della simulazione Digital Twin, sono state eseguite attività di test su più livelli: funzionale, di integrazione e architetturale.

Attività di test eseguite

Functional Testing – FlexSim Components

- Verifica del corretto comportamento delle *Fixed Resources* all'interno dell'ambiente di simulazione.
- Validazione delle transizioni di stato (shape, animazioni, logiche operative).
- Controllo delle richieste HTTP verso il backend API.
- Esecuzione di test su scenari multipli per garantire copertura funzionale.

Microservices Architecture Validation

- Verifica dell'interazione tra microservizi e componenti esterni.
    - Analisi di:
    - Scalabilità
    - Prestazioni
    - Resilienza
    - Sicurezza
- Utilizzo di approcci di test mirati per ogni dimensione architetturale.

System Integration Testing

- Verifica dell'integrazione end-to-end tra sensori, backend, database e simulazione.
- Simulazione di scenari realistici per replicare condizioni operative di produzione.

Risultati del test

Le attività di test hanno confermato che la simulazione e l'architettura a microservizi soddisfano i requisiti di progetto in termini di:

- Funzionalità
- Prestazioni
- Sicurezza
- Affidabilità

Durante le fasi di validazione sono stati individuati e risolti alcuni bug minori, contribuendo al miglioramento della stabilità e della qualità complessiva del sistema.

Conclusioni

Il processo di test ha fornito un elevato livello di confidenza sulla robustezza dell'architettura e sul corretto funzionamento della simulazione.
Il sistema risulta idoneo per un contesto di produzione, garantendo il rispetto dei requisiti tecnici e degli obiettivi operativi.