MICHELANGELO SMART GATE

Cancello automatico con Arduino UNO R4 WiFi — **Quarto Michelangelo · 5a AMS · Anno scolastico 2025/2026**

> Modello didattico di cancello scorrevole automatizzato. Il progetto dimostra l'integrazione tra meccanica, sistemi automatici, elettronica e informatica per creare un sistema embedded controllabile da remoto tramite Arduino Cloud.

Indice

Struttura della relazione tecnica sintetica del progetto Michelangelo Smart Gate.

1. Introduzione

Michelangelo Smart Gate è un progetto didattico basato sulla realizzazione di un cancello automatico scorrevole in scala, controllato da una scheda **Arduino UNO R4 WiFi** e monitorabile tramite Arduino Cloud. Il sistema integra elementi meccanici, elettronici e software, simulando il funzionamento di una vera automazione per accessi.

Lo scopo della relazione è descrivere l'idea progettuale, i componenti utilizzati, i collegamenti, la logica di funzionamento, le fasi di sviluppo e le verifiche effettuate. Il progetto è pensato per una presentazione tecnica all'Esame di Stato e mostra come più discipline possano collaborare nella realizzazione di un sistema intelligente.

🔧 Meccanica

Struttura scorrevole con pignone e cremagliera

⚡ Elettronica

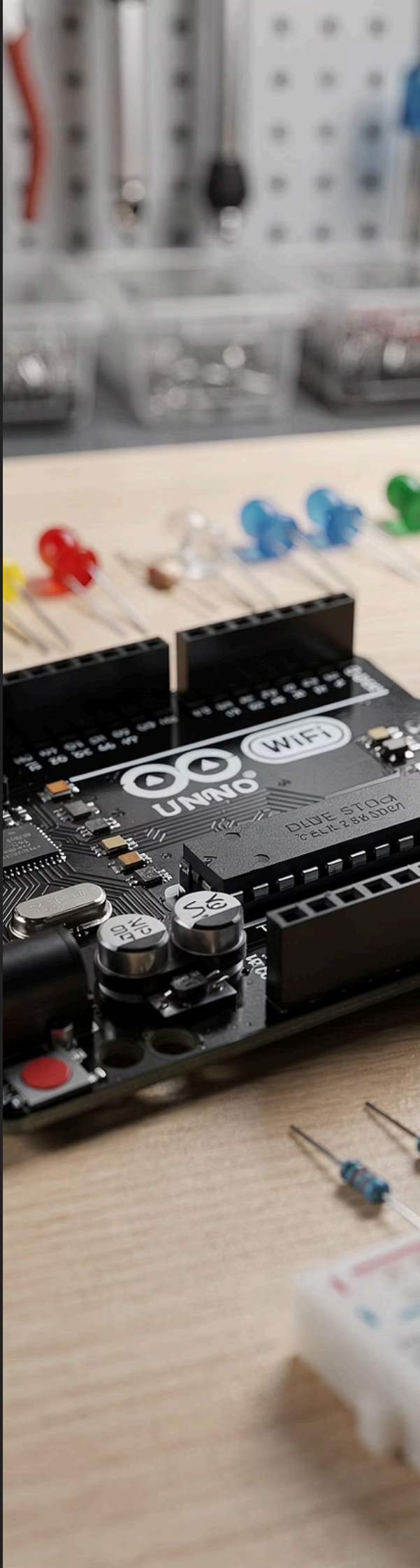
Driver BTS7960, motore DC, sensori e alimentazione

💻 Software

Programmazione Arduino con logica di controllo

☁️ Cloud IoT

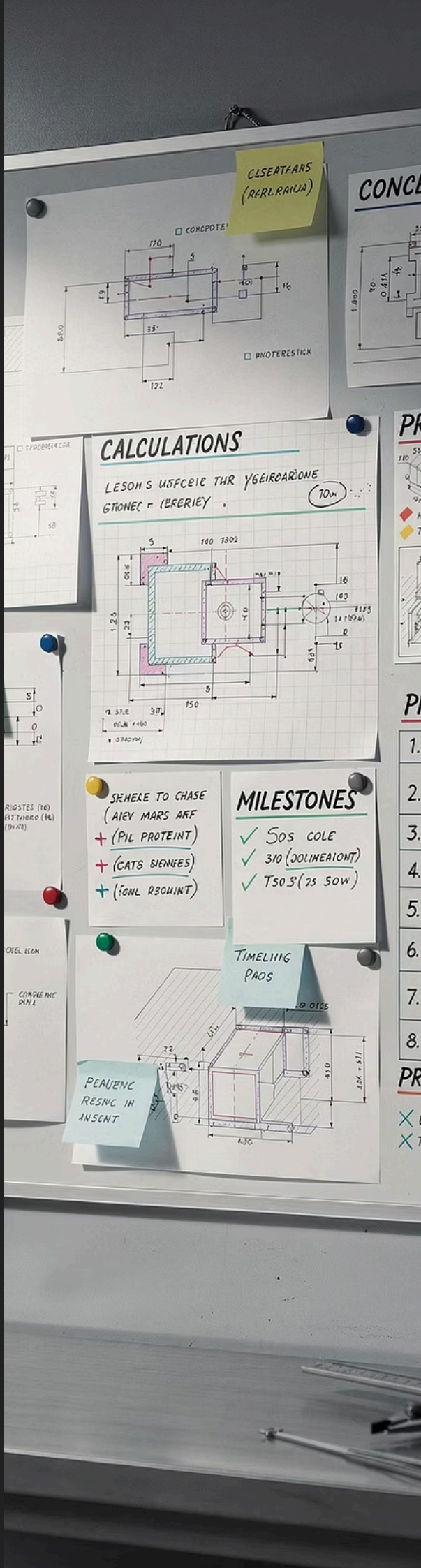
Monitoraggio e controllo remoto via Arduino Cloud



2. Obiettivi del progetto

Il progetto si propone di raggiungere obiettivi tecnici precisi, coprendo l'intero ciclo di un sistema automatico embedded: dalla meccanica al Cloud.

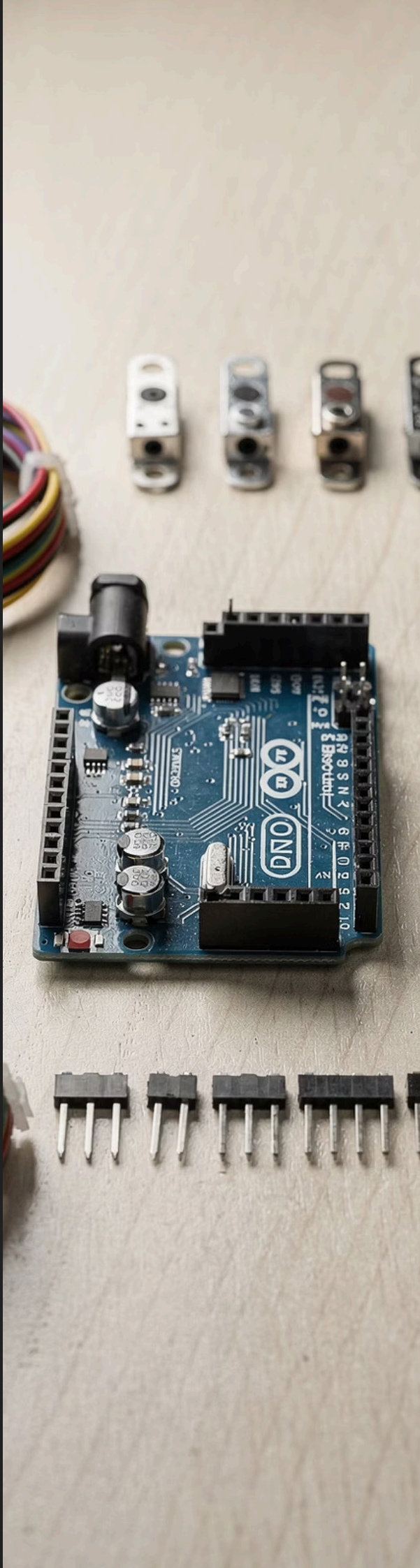
1	**Modello cancello scorrevole** Realizzare un modello di cancello automatico scorrevole controllato elettronicamente.
2	**Gestione comandi** Gestire apertura, chiusura, stop e stato del cancello tramite Arduino.
3	**Driver BTS7960** Utilizzare un driver BTS7960 per pilotare il motore DC in entrambe le direzioni con controllo PWM.
4	**Finecorsa di sicurezza** Usare finecorsa di apertura e chiusura per arrestare il movimento in sicurezza.
5	**Arduino Cloud** Integrare Arduino Cloud per monitoraggio e controllo remoto del sistema.
6	**Presentazione tecnica** Presentare il progetto tramite un sito tecnico e una dashboard simulata.



3. Materiali e componenti

I componenti principali sono stati scelti per rappresentare in modo chiaro le parti fondamentali di un sistema automatico: **controllo, potenza, movimento, sensori e alimentazione**.

Componente	Funzione	Motivo della scelta
Arduino UNO R4 WiFi	Unità di controllo	Integra logica, pin digitali e connessione WiFi.
Driver BTS7960	Controllo potenza motore	Permette inversione di marcia e controllo PWM del motore.
Motore DC 12V con encoder	Movimento cancello	Fornisce coppia e può restituire informazioni su velocità/posizione.
Finecorsa apertura/chiusura	Sensori di limite	Fermano il movimento a fine corsa.
Batteria/alimentatore 12V	Alimentazione motore	Fornisce energia separata al circuito di potenza.
Pignone e cremagliera	Trasmissione meccanica	Trasformano la rotazione del motore in movimento lineare.
LED lampeggiante	Segnalazione visiva	Indica che il cancello è in movimento.
Arduino Cloud	Monitoraggio remoto	Permette comandi e stato via dashboard web/app.



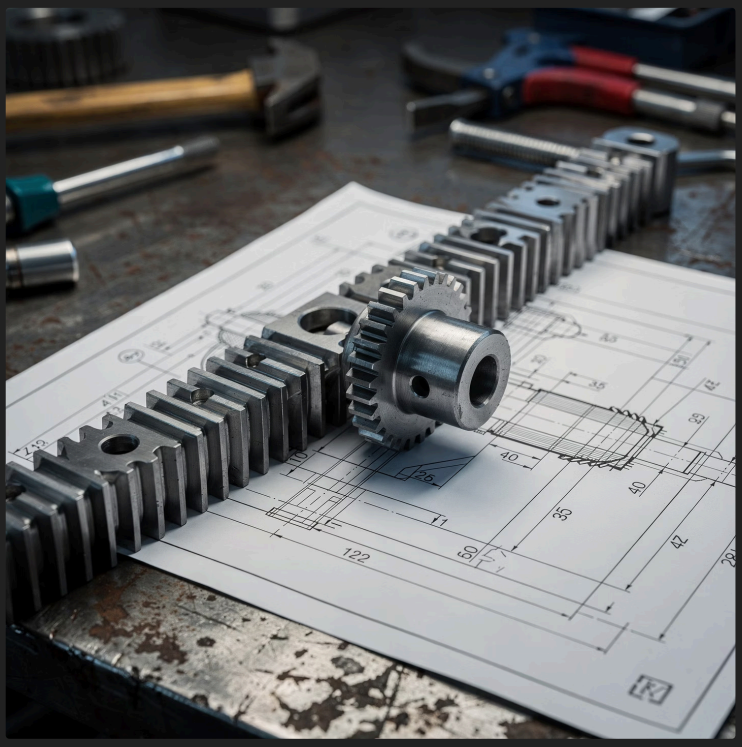
4. Architettura del sistema

Il sistema è organizzato in **quattro blocchi principali**: logica di controllo, potenza, parte meccanica e interfaccia Cloud. Arduino riceve comandi e segnali dai sensori, elabora le condizioni e invia al driver i segnali PWM necessari per comandare il motore.

Panoramica tecnica del sistema: controllo, potenza, sensori, Cloud e alimentazione integrati in un'architettura coerente e modulare.

5. Progettazione meccanica

La parte meccanica è basata su un **cancello scorrevole in scala**. Il movimento lineare è ottenuto tramite una cremagliera fissata alla parte mobile e un pignone collegato all'albero del motore. Il motore trasmette la rotazione al pignone, che spinge la cremagliera e quindi fa scorrere il cancello.



Sistema pignone-cremagliera

La struttura è stata pensata per essere realizzata con elementi stampati in 3D, supporti meccanici, viti e una base rigida.

- **Pignone:** ingranaggio collegato all'albero del motore
- **Cremagliera:** dentatura lineare fissata alla parte mobile
- **Parte mobile:** il pannello del cancello che scorre
- **Finecorsa:** posizionati alle estremità per riconoscere apertura e chiusura
- **Base rigida:** supporto strutturale dell'intero sistema

Dettaglio del sistema meccanico pignone-cremagliera e parte mobile del cancello scorrevole in scala.

6. Schema elettrico e collegamenti

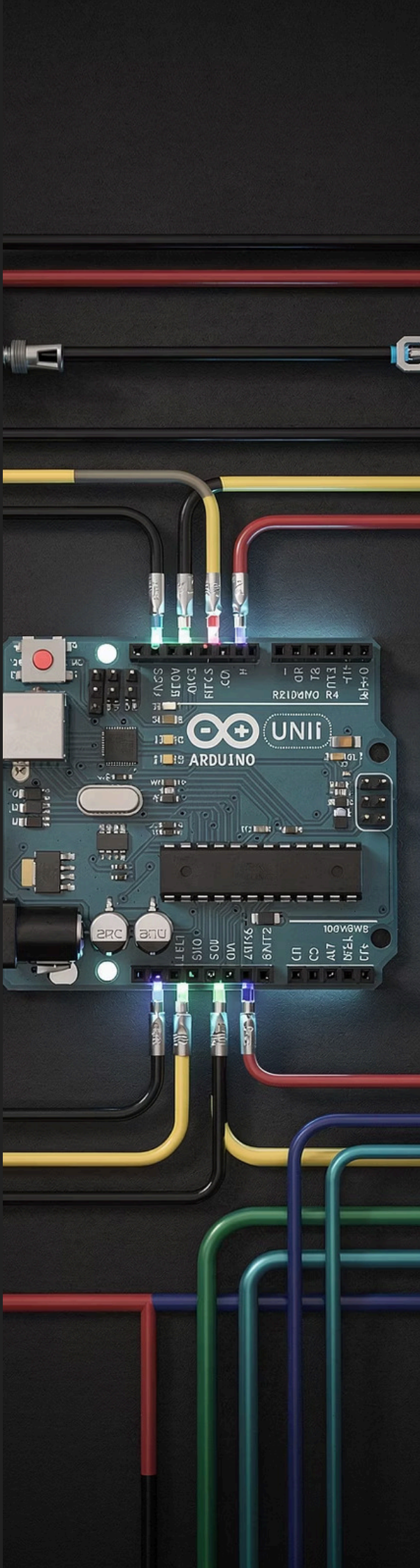
Lo schema elettrico collega **Arduino UNO R4 WiFi** al driver BTS7960, al motore DC, all'encoder, ai finecorsa, al LED lampeggiante e all'alimentazione a 12V. Una regola fondamentale è mantenere tutte le masse GND in comune, in modo che Arduino e il driver abbiano lo stesso riferimento elettrico.

Legenda colori cavi

- Segnali PWM (controllo velocità)
- Segnali Encoder (posizione/velocità)
- Alimentazione 5V (logica)
- GND (massa comune)
- Alimentazione 12V (potenza motore)
- Segnali Digitali (finecorsa, LED)

Note importanti

- Il driver BTS7960 gestisce la potenza del motore
- Arduino legge i finecorsa e l'encoder per controllo posizione e sicurezza
- Il LED lampeggiante indica che il cancello è in movimento
- GND comune obbligatorio tra Arduino e driver



Schema collegamenti Arduino UNO R4 WiFi – BTS7960 – motore DC 12V con encoder.

Tabella dei collegamenti principali

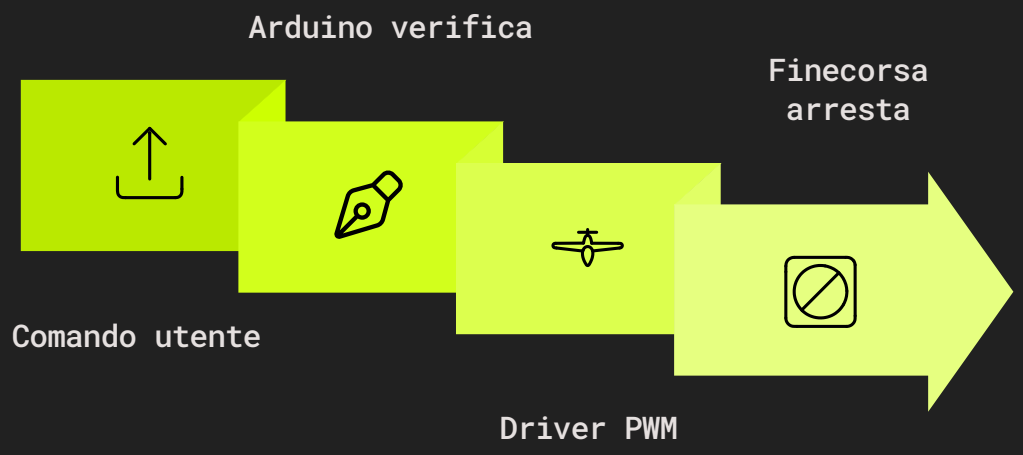
Riepilogo completo di tutti i collegamenti tra i componenti del sistema Michelangelo Smart Gate.

Da	A	Descrizione
Arduino D5	BTS7960 RPWM	Segnale PWM apertura
Arduino D6	BTS7960 LPWM	Segnale PWM chiusura
Arduino 5V	BTS7960 VCC	Alimentazione logica
Arduino GND	BTS7960 GND	Massa comune
Batteria +12V	BTS7960 B+	Alimentazione potenza
Batteria GND	BTS7960 B-	Massa potenza
Motore filo 1	BTS7960 M+	Uscita motore
Motore filo 2	BTS7960 M-	Uscita motore
Encoder VCC	Arduino 5V	Alimentazione encoder
Encoder GND	Arduino GND	Massa encoder
Encoder A	Arduino D2	Canale encoder A
Encoder B	Arduino D3	Canale encoder B
Finecorsa apertura	Arduino D8 + GND	Sensore limite apertura
Finecorsa chiusura	Arduino D9 + GND	Sensore limite chiusura
LED lampeggiante	Arduino D10 + 220Ω + GND	Segnalazione movimento

ⓘ Tutti i GND devono essere collegati in comune tra Arduino, driver BTS7960 e batteria per garantire il corretto riferimento dei segnali elettrici.

7. Funzionamento del sistema

Il funzionamento segue una **logica sequenziale**: l'utente invia un comando, Arduino controlla lo stato dei sensori, attiva il driver, il motore muove il cancello e i finecorsa arrestano il movimento quando viene raggiunto il limite.



🟢 Comando APRI

Arduino attiva il segnale `RPWM` sul driver BTS7960 e il cancello si apre verso destra.

🔴 Comando CHIUDI

Arduino attiva il segnale `LPWM` sul driver BTS7960 e il cancello si chiude verso sinistra.

⏹️ Comando STOP

Il driver viene disattivato e il motore si ferma immediatamente in qualsiasi posizione.

📍 Finecorsa apertura

Arduino arresta il motore e aggiorna lo stato in **APERTO**.

📍 Finecorsa chiusura

Arduino arresta il motore e aggiorna lo stato in **CHIUSO**.

🔄 Encoder

Consente il conteggio degli impulsi e la lettura teorica di velocità e posizione del cancello.

8. Programmazione Arduino

Il programma Arduino gestisce i comandi, controlla il motore tramite **PWM**, legge i finecorsa e aggiorna lo stato del sistema. Nel progetto completo le variabili possono essere collegate ad Arduino Cloud per il controllo remoto.

```
// Michelangelo Smart Gate - logica semplificata
#define RPWM 5
#define LPWM 6
#define ENCODER_A 2
#define ENCODER_B 3
#define FINE_APERTURA 8
#define FINE_CHIUSURA 9
#define LED_MOVIMENTO 10

String statoCancello = "CHIUSO";

void setup() {
  pinMode(RPWM, OUTPUT);
  pinMode(LPWM, OUTPUT);
  pinMode(LED_MOVIMENTO, OUTPUT);
  pinMode(FINE_APERTURA, INPUT_PULLUP);
  pinMode(FINE_CHIUSURA, INPUT_PULLUP);
  pinMode(ENCODER_A, INPUT_PULLUP);
  pinMode(ENCODER_B, INPUT_PULLUP);
  fermaMotore();
}

void apriCancello() {
  if (digitalRead(FINE_APERTURA) == LOW) return;
  statoCancello = "IN APERTURA";
  analogWrite(RPWM, 180);
  analogWrite(LPWM, 0);
  digitalWrite(LED_MOVIMENTO, HIGH);
}

void chiudiCancello() {
  if (digitalRead(FINE_CHIUSURA) == LOW) return;
  statoCancello = "IN CHIUSURA";
  analogWrite(RPWM, 0);
  analogWrite(LPWM, 180);
  digitalWrite(LED_MOVIMENTO, HIGH);
}

void fermaMotore() {
  analogWrite(RPWM, 0);
  analogWrite(LPWM, 0);
  digitalWrite(LED_MOVIMENTO, LOW);
}

void loop() {
  if (digitalRead(FINE_APERTURA) == LOW) {
    fermaMotore();
    statoCancello = "APERTO";
  }
  if (digitalRead(FINE_CHIUSURA) == LOW) {
    fermaMotore();
    statoCancello = "CHIUSO";
  }
}
```

Struttura del codice

- **setup():** configura tutti i pin come INPUT o OUTPUT
- **apriCancello():** attiva RPWM con valore 180, LED acceso
- **chiudiCancello():** attiva LPWM con valore 180, LED acceso
- **fermaMotore():** azzera entrambi i PWM, LED spento
- **loop():** controlla continuamente i finecorsa

> Il valore PWM 180 su 255 corrisponde a circa il 70% della velocità massima del motore.

9. Arduino Cloud e dashboard

Arduino Cloud permette di controllare e monitorare il sistema da remoto tramite una **dashboard web o mobile**. Le variabili principali possono essere configurate come Read & Write per i comandi e Read Only per gli stati.

Esempio di dashboard Arduino Cloud per controllo e monitoraggio remoto del cancello scorrevole.

Variabile	Tipo	Funzione
`apri`	Bool – Read & Write	Comando apertura cancello
`chiudi`	Bool – Read & Write	Comando chiusura cancello
`stopMotore`	Bool – Read & Write	Arresto di emergenza
`statoCancello`	String – Read Only	Mostra lo stato del cancello
`posizione`	Int – Read Only	Percentuale apertura teorica
`rpmMotore`	Int – Read Only	Velocità stimata/encoder

10. Sicurezza

Anche se il progetto è didattico, la **sicurezza è una parte centrale** della progettazione. Un sistema automatico deve poter fermare il movimento quando raggiunge i limiti o quando viene rilevata una condizione anomala.

Finecorsa apertura

Evita che il cancello superi la posizione aperta massima.

Finecorsa chiusura

Evita che il cancello continui a spingere da chiuso.

Comando STOP

Interrompe immediatamente il movimento del motore in qualsiasi condizione.

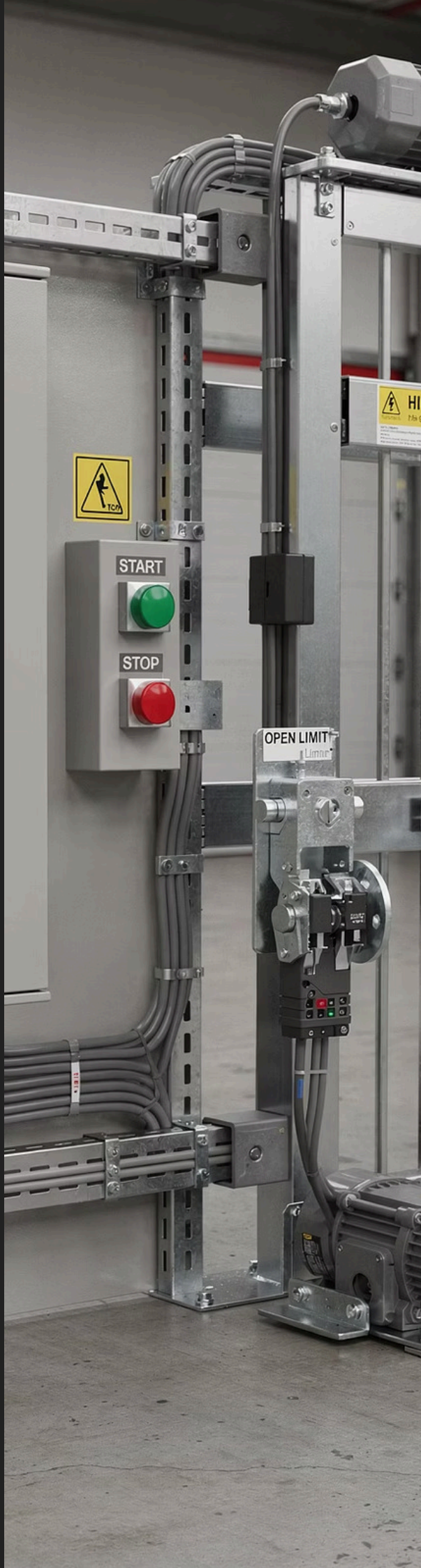
Alimentazione separata

Arduino gestisce la logica; la batteria 12V alimenta il motore separatamente.

GND comune

Necessario per il corretto riferimento dei segnali tra Arduino e driver.

⚠ Il modello ha scopo didattico e non sostituisce un impianto certificato per uso reale. Non utilizzare su cancelli reali senza adeguata certificazione di sicurezza.



11. Fasi di sviluppo – Fase 1 e 2

Il percorso di sviluppo segue le fasi reali di un progetto tecnico: progettazione, ricerca componenti, modellazione 3D, stampa, assemblaggio, programmazione, configurazione Cloud, test e versione finale.

Fase 1 – Progettazione del sistema

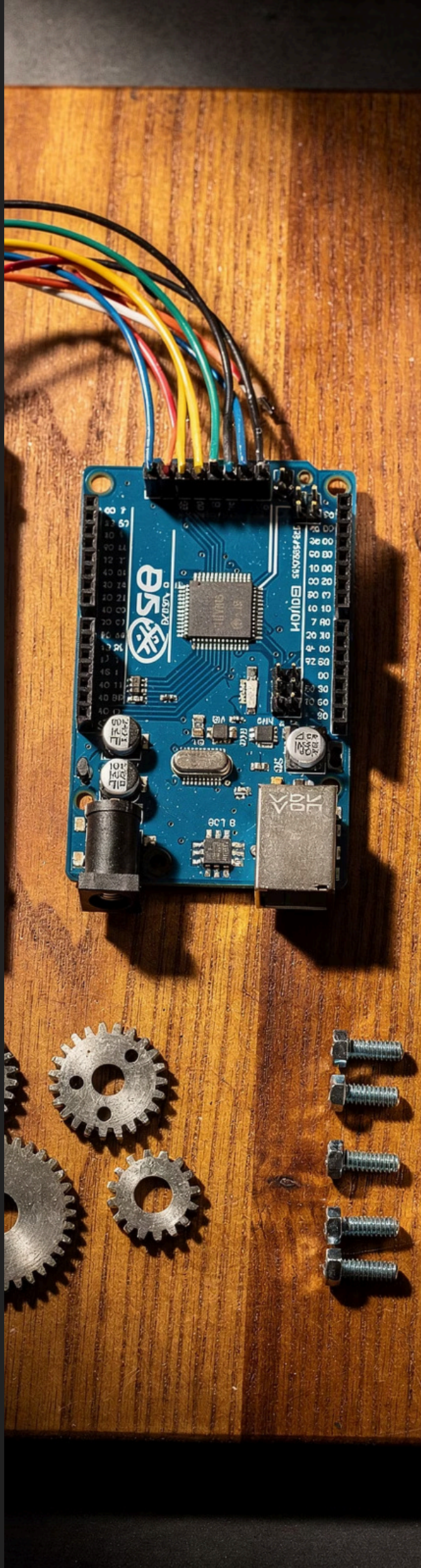
Studio dell'idea progettuale, definizione dell'architettura generale e individuazione delle funzioni richieste.

- Apertura automatica
- Chiusura automatica
- Controllo tramite web app
- Monitoraggio posizione e stato
- Sicurezza con finecorsa e encoder

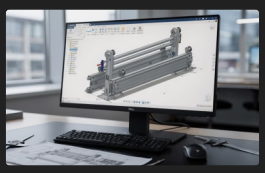
Fase 2 – Ricerca del materiale

Analisi e selezione dei componenti elettronici e meccanici necessari al progetto.

- Arduino UNO R4 WiFi
- Driver BTS7960
- Motore DC con encoder
- Finecorsa di apertura e chiusura
- Batteria 12V
- Cremagliera, pignone e struttura

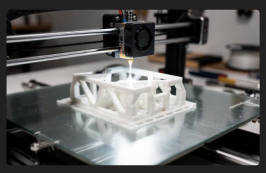


11. Fasi di sviluppo — Fasi 3, 4 e 5



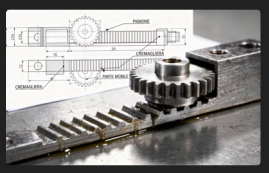
Fase 3 – Prototipazione 3D con Fusion 360

Progettazione CAD della struttura del cancello e definizione delle dimensioni, dei supporti e della posizione dei componenti meccanici ed elettronici.



Fase 4 – Stampa 3D dei componenti

Produzione dei supporti e delle parti strutturali necessarie al montaggio del prototipo, realizzate con filamento PLA su stampante 3D.



Fase 5 – Pignone, cremagliera e parte mobile

Creazione del sistema meccanico responsabile dello scorrimento lineare del cancello: pignone sull'albero motore, cremagliera sulla parte mobile.

11. Fasi di sviluppo – Fasi 6, 7 e 8

Fase 6 – Assemblaggio

1

Montaggio della struttura, installazione del motore, cablaggio dei componenti e posizionamento dei finecorsa sulle estremità della guida.

2

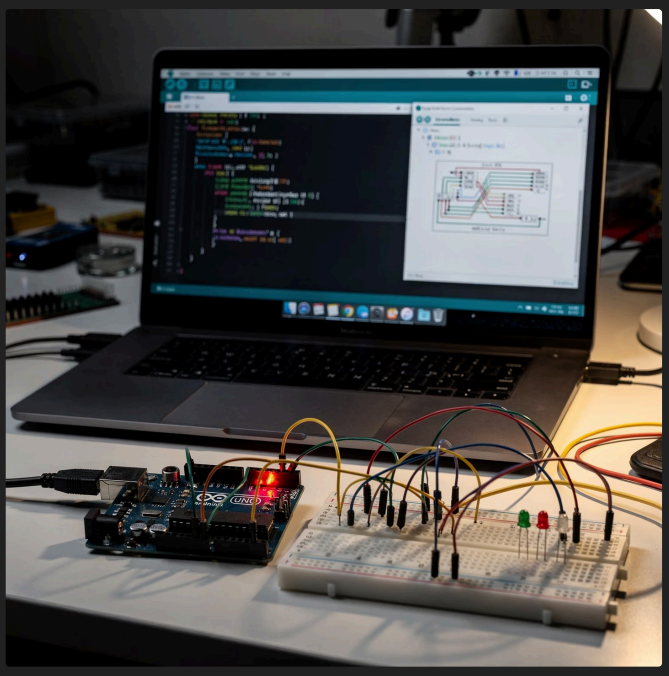
Fase 7 – Programmazione Arduino

Sviluppo del codice per gestire apertura, chiusura, stop, finecorsa, encoder e stato del sistema tramite Arduino IDE 2.2.1.

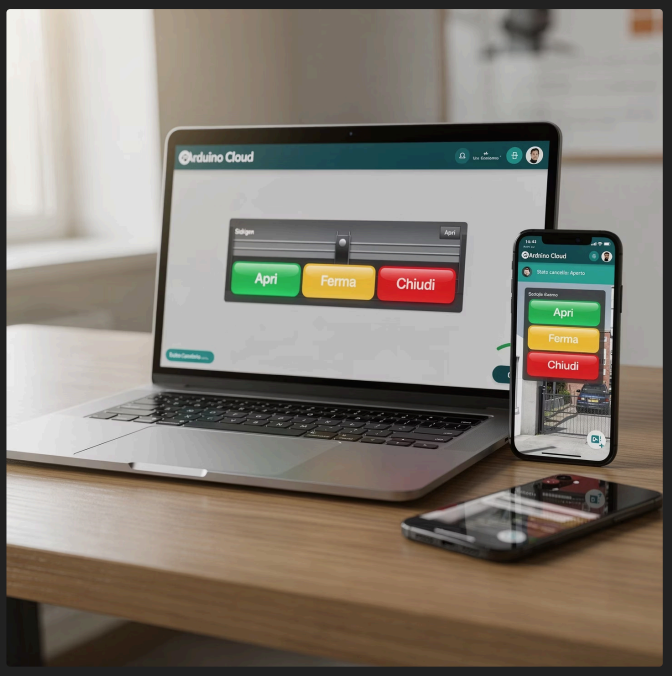
Fase 8 – Configurazione Cloud

3

Creazione della dashboard per controllo remoto, comandi e monitoraggio dello stato del cancello tramite Arduino Cloud.



Programmazione in Arduino IDE 2.2.1



Dashboard Arduino Cloud configurata

11. Fasi di sviluppo — Fasi 9 e 10

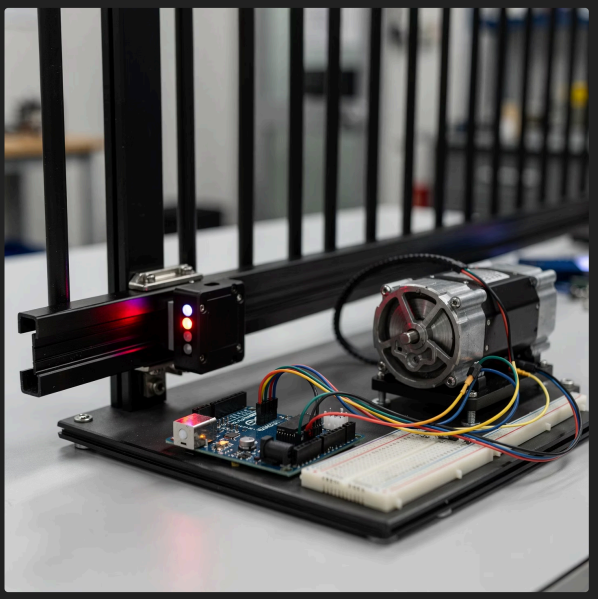
Fase 9 — Test di funzionamento

Verifica dell'apertura, chiusura, stop, finecorsa, encoder e dashboard. Ogni funzione è stata testata singolarmente e poi in sequenza completa.

Test	Esito
Apertura cancello	✓ OK
Chiusura cancello	✓ OK
Finecorsa apertura	✓ OK
Finecorsa chiusura	✓ OK
Arresto emergenza	✓ OK
Controllo da app	✓ OK
Rilevamento ostacoli	✓ OK

Fase 10 — Versione finale

Sistema finale con meccanica, elettronica, software e monitoraggio remoto integrati in un unico prototipo funzionante.



12. Test e collaudo

Il collaudo consente di verificare che il sistema risponda correttamente ai comandi e che le condizioni di sicurezza siano rispettate. I test principali riguardano **movimento, sensori, comunicazione e interfaccia utente**.

Test	Risultato atteso	Esito
Apertura cancello	Il cancello passa da chiuso ad aperto	✅ Superato
Chiusura cancello	Il cancello torna da aperto a chiuso	✅ Superato
Finecorsa apertura	Il motore si ferma a fine apertura	✅ Superato
Finecorsa chiusura	Il motore si ferma a fine chiusura	✅ Superato
Comando STOP	Arresto immediato del movimento	✅ Superato
Dashboard Cloud	Stato e comandi visibili da remoto	✅ Superato

Tutti i test principali sono stati superati con successo. Il sistema risponde correttamente ai comandi e le condizioni di sicurezza sono pienamente operative.

13. Competenze acquisite & Conclusione

Il progetto Michelangelo Smart Gate ha permesso di sviluppare competenze trasversali che spaziano dalla meccanica all'IoT, dimostrando come più discipline possano integrarsi nella realizzazione di un sistema embedded completo.

🔩 Meccanica

Movimento lineare tramite pignone e cremagliera

🖥️ Modellazione 3D

Fusion 360 per struttura e supporti

🖨️ Stampa 3D

Componenti personalizzati in PLA

⚡ Elettronica

Driver, motore, alimentazione, sensori e LED

🤖 Sistemi automatici

Input, elaborazione, output e feedback

💻 Programmazione

Funzioni apertura, chiusura, stop e sicurezza

☁️ IoT e Cloud

Monitoraggio e controllo remoto Arduino Cloud

🧩 Problem solving

Individuazione errori e verifica progressiva

> Il progetto dimostra come meccanica, elettronica, programmazione e IoT possano integrarsi in un sistema embedded funzionante, controllabile da remoto e progettato con attenzione alla sicurezza. Un esempio concreto di ingegneria applicata in ambito didattico.