



UNIVERSITÀ
degli STUDI
di CATANIA

Sistemi Cloud e Laboratorio SolarCar Cloud Dashboard

Giuseppe Napoli - 1000012802

A.A. 2023/24

Docenti

- Prof. Giuseppe Pappalardo
- Prof. Andrea Francesco Fornaia

Indice

1	Introduzione	4
2	Architettura	5
2.1	Data sender	5
2.2	Backend	6
2.3	Google Cloud Pub/Sub	7
2.4	Google Cloud Storage	8
2.5	Kubernetes	8
2.6	Considerazioni sull'Architettura	9
3	Dashboard	10
3.1	Temperature	10
3.2	Tensioni	11
3.3	Motori	12
3.4	MPPTs	13
3.5	Considerazioni sulla Dashboard	14
4	Costi dei servizi Cloud	15
4.1	Google Kubernetes Engine (GKE)	15
4.1.1	Costi di Google Kubernetes Engine	15
4.2	Google Cloud Pub/Sub	16
4.2.1	Costi di Google Cloud Pub/Sub	16
4.3	Google Cloud Storage	16
4.3.1	Costi di Google Cloud Storage	16
4.4	Google Cloud Functions	17
4.4.1	Costi di Google Cloud Functions	17
4.5	Stime Complessive dei Costi	17
4.6	Considerazioni sui Costi di Google Cloud	18
5	CI/CD: Continuous Integration e Continuous Deployment	19
5.1	Introduzione alla CI/CD	19
5.2	Strumenti utilizzati	19
5.3	Pipeline CI/CD	19
5.4	Considerazioni su CI/CD	22

6	Conclusioni	23
6.1	Sviluppi Futuri	23
6.2	Conclusioni Finali	24

1 Introduzione

Il progetto *SolarCar Cloud Dashboard* nasce dall'esigenza di creare un sistema distribuito, scalabile e cloud-based per la gestione e il monitoraggio di dati provenienti da un'auto alimentata a energia solare. L'idea alla base del progetto è quella di sviluppare un'infrastruttura che permetta la raccolta, l'elaborazione e la visualizzazione dei dati tramite una dashboard accessibile via web.

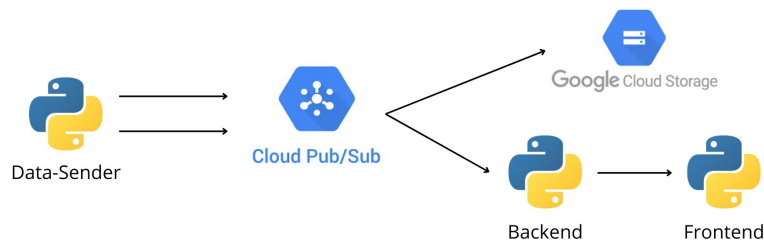


Figure 1: Pipeline del progetto, le frecce indicano il flusso di dati.

Attraverso l'uso di tecnologie cloud all'avanguardia come Kubernetes, Google Cloud Pub/Sub e Google Cloud Storage, è stato possibile progettare un sistema modulare, capace di simulare la ricezione di pacchetti dati CANbus, gestirli in tempo reale e presentare i risultati all'utente finale. Questo approccio non solo garantisce un'alta affidabilità e scalabilità, ma riduce i costi operativi grazie all'utilizzo efficiente delle risorse cloud.

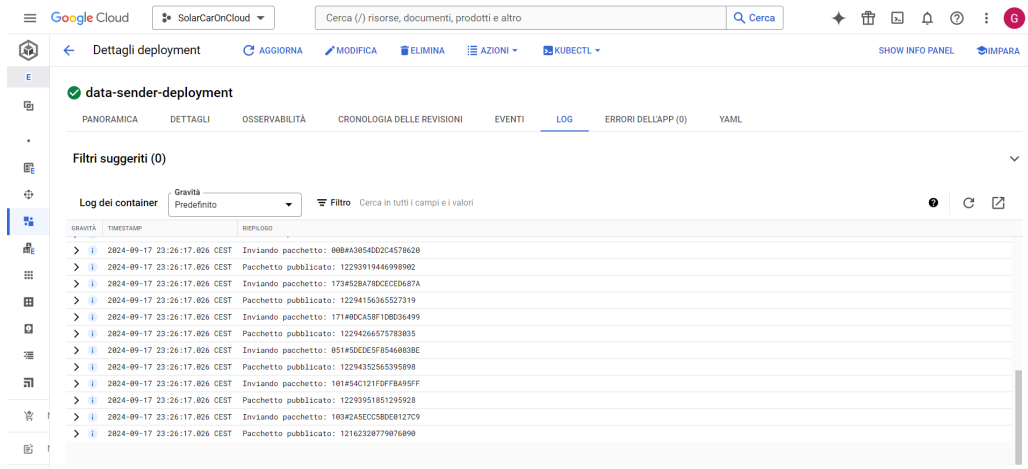
L'integrazione tra il backend, che decodifica e memorizza i dati, e il frontend, che visualizza informazioni come tensione, temperatura e stato del motore, crea un ecosistema completo e user-friendly. Inoltre, il progetto è pensato per evolversi e supportare nuove funzionalità future, migliorando così la gestione e la supervisione di veicoli elettrici solari o altri sistemi analoghi.

2 Architettura

L'architettura del progetto *SolarCar Cloud Dashboard* si basa su una complessa integrazione di vari componenti software e servizi cloud che lavorano insieme per raccogliere, elaborare e visualizzare dati. L'obiettivo principale del sistema è emulare l'invio di messaggi di dati provenienti da un veicolo a energia solare tramite un sistema CANbus simulato, processare tali dati, organizzarli e renderli disponibili in una dashboard accessibile tramite il frontend. In questo contesto, ogni componente dell'architettura ha un ruolo specifico e fondamentale per il corretto funzionamento dell'intero sistema. Di seguito, analizziamo nel dettaglio ogni parte dell'architettura.

2.1 Data sender

Il data-sender è responsabile dell'emulazione dei messaggi CANbus, che simulano i dati inviati dal veicolo. Ma cosa è esattamente un sistema CANbus? Il CANbus (Controller Area Network) è un protocollo di comunicazione progettato per consentire la trasmissione di dati tra diversi dispositivi elettronici, soprattutto in contesti automobilistici. Viene ampiamente utilizzato per collegare centraline elettroniche nei veicoli, permettendo loro di comunicare tra di loro in tempo reale. Questo sistema è altamente affidabile e resistente alle interferenze, il che lo rende ideale per ambienti dove è richiesta una comunicazione sicura e veloce tra diversi sensori e attuatori, come motori, sistemi di frenata o batterie.



GRAVITÀ	TIMESTAMP	RIEPILOGO
>	2024-09-17 23:26:17.826 CEST	Inviando pacchetto: 008A43054D00C45786209
>	2024-09-17 23:26:17.826 CEST	Pacchetto pubblicato: 12293919446998982
>	2024-09-17 23:26:17.826 CEST	Inviando pacchetto: 1734528A78DCECEd687A
>	2024-09-17 23:26:17.826 CEST	Pacchetto pubblicato: 12294156365527319
>	2024-09-17 23:26:17.826 CEST	Inviando pacchetto: 17148DCA58F108D06499
>	2024-09-17 23:26:17.826 CEST	Pacchetto pubblicato: 12294266575783835
>	2024-09-17 23:26:17.826 CEST	Inviando pacchetto: 05145DCE5F85468838E
>	2024-09-17 23:26:17.826 CEST	Pacchetto pubblicato: 12294332565395898
>	2024-09-17 23:26:17.826 CEST	Inviando pacchetto: 181454C121F0FFBA95FF
>	2024-09-17 23:26:17.826 CEST	Pacchetto pubblicato: 12293951851295928
>	2024-09-17 23:26:17.826 CEST	Inviando pacchetto: 18342A5ECC580E8127C9
>	2024-09-17 23:26:17.826 CEST	Pacchetto pubblicato: 12162328779876898

Figure 2: Logs del data-sender. Tengono traccia dei messaggi inviati.

Nel contesto del progetto, il data-sender invia messaggi simulati, ognuno dei quali rappresenta un pacchetto di dati in formato CANbus. Un tipico messaggio ha la struttura **153#90B4B8CC1F9B35BA**, dove la parte **153** rappresenta l'identificatore del messaggio e **90B4B8CC1F9B35BA** rappresenta i dati effettivi, codificati in esadecimale. L'obiettivo del data-sender è trasmettere questi messaggi in modo casuale e continuo, simulando l'attività di un veicolo reale.

2.2 Backend

Il backend è il cuore del sistema, responsabile della decodifica dei messaggi ricevuti dal data-sender. La decodifica avviene in base all'ID contenuto nei messaggi, che permette di classificare i dati in diverse categorie: *temperature*, *voltages* (tensioni), *engines* (motori), e *mppts* (maximum power point trackers). Questo processo di organizzazione è essenziale per poter interpretare i dati ricevuti in modo utile e significativo.

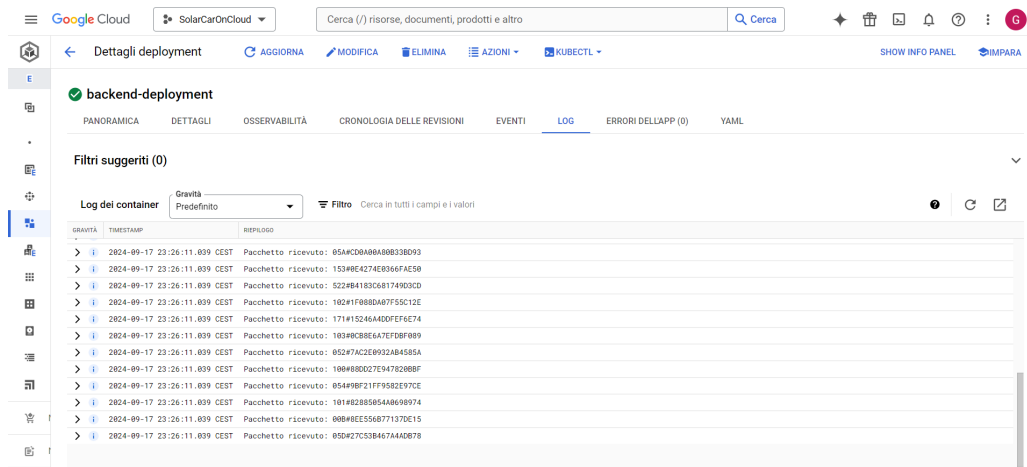


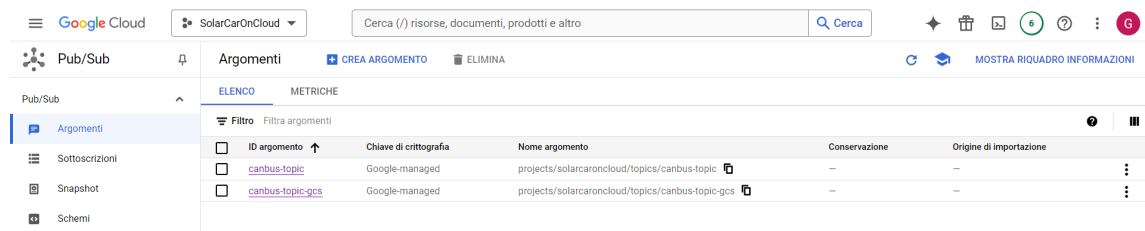
Figure 3: Logs del backend. Conferma la recezione dei messaggi dal topic Pub/Sub.

Il backend, una volta decodificati i messaggi, rende questi dati disponibili attraverso quattro diversi endpoint API. Ciascuno degli endpoint corrisponde a una delle quattro categorie sopra menzionate e può essere richiamato dal frontend per ottenere dati aggiornati. Ad esempio, l'endpoint */temperatures* restituisce le informazioni relative alle temperature rilevate, mentre */voltages* fornisce i valori di tensione raccolti.

Il backend utilizza Google Cloud Pub/Sub per ricevere i messaggi inviati dal data-sender e aggiorna i dati ogni volta che viene ricevuto un nuovo pacchetto. Pub/Sub è un servizio di messaggistica asincrona che facilita la comunicazione tra sistemi distribuiti, come in questo caso tra il data-sender e il backend.

2.3 Google Cloud Pub/Sub

Google Cloud Pub/Sub è uno dei pilastri centrali dell'architettura del progetto. Si tratta di un servizio di messaggistica in cui i dati vengono inviati (pubblicati) su un argomento (topic) e possono essere ricevuti (sottoscritti) da uno o più consumatori. In questo progetto, il data-sender pubblica i messaggi CANbus su un topic Pub/Sub, e il backend è uno dei consumatori (subscriber) che ascolta costantemente per ricevere nuovi messaggi.



ID argomento	Chiave di crittografia	Nome argomento	Conservazione	Origine di importazione
canbus-topic	Google-managed	projects/solarcaroncloud/topics/canbus-topic	—	—
canbus-topic-gcs	Google-managed	projects/solarcaroncloud/topics/canbus-topic-gcs	—	—

Figure 4: Topic Pub/Sub sui quali scrive il data sender, dal primo legge il backend, mentre il contenuto del secondo viene scritto sul bucket.

Una caratteristica fondamentale di Pub/Sub è la sua capacità di scalare in modo quasi illimitato, gestendo grandi quantità di dati in tempo reale. Questo lo rende particolarmente adatto per applicazioni distribuite su larga scala come questa. Rispetto ad altre tecnologie di messaggistica come Kafka, Google Cloud Pub/Sub è spesso preferito in ambienti Google Cloud per la sua perfetta integrazione con altri servizi Google, la gestione semplificata e il modello di pricing basato sull'utilizzo effettivo.

Nel contesto di questo progetto, è stato configurato un doppio topic. Il primo topic è collegato direttamente al backend tramite una sottoscrizione pull, in cui il backend preleva attivamente i messaggi. Il secondo topic, invece, è collegato a Google Cloud Storage tramite una sottoscrizione push e una Cloud Function, che scrive i dati dei messaggi ricevuti direttamente in uno storage bucket. Questa architettura a doppio topic garantisce sia l'elaborazione in tempo reale (tramite il backend) sia la persistenza dei dati (tramite Cloud Storage).

2.4 Google Cloud Storage

Google Cloud Storage è il servizio di storage object-based di Google, utilizzato per archiviare i dati in modo durevole e accessibile. In questo progetto, i dati dei messaggi ricevuti tramite Pub/Sub vengono salvati in un bucket di Cloud Storage. Prima di essere salvati, i messaggi vengono convertiti in un formato più leggibile rispetto all'originale *application/octet-stream*. Questo consente una più facile consultazione e analisi dei dati in futuro.

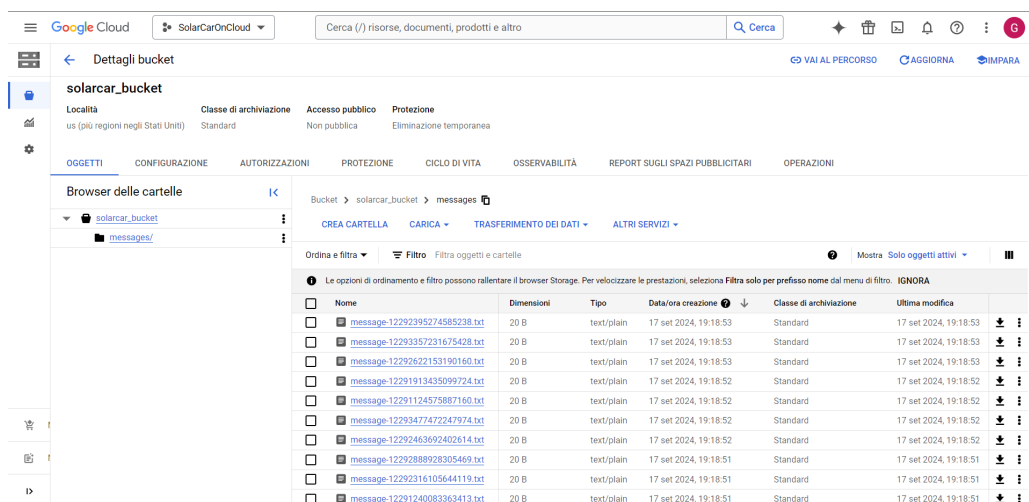


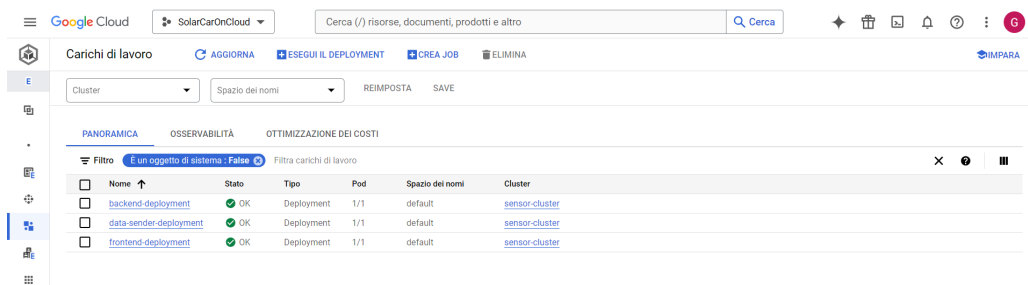
Figure 5: Bucket su Google Cloud Storage. Contiene tutti i messaggi che sono stati inviati dal sender.

La scelta di utilizzare Cloud Storage come sistema di persistenza garantisce affidabilità e accessibilità ai dati raccolti, anche in caso di fallimenti o interruzioni temporanee del sistema. Inoltre, Cloud Storage offre una scalabilità automatica e una gestione semplificata, rendendolo una soluzione ideale per archiviare grandi quantità di dati non strutturati come quelli generati dai messaggi CANbus.

2.5 Kubernetes

L'ultimo componente chiave dell'architettura è Kubernetes, una piattaforma di orchestrazione per container che automatizza il deployment, la gestione e la scalabilità delle applicazioni containerizzate. Kubernetes è stato scelto per gestire il deployment di tutti i componenti del sistema, inclusi il data-sender, il backend e il frontend.

Il vantaggio principale di utilizzare Kubernetes in questo contesto è la sua capacità di gestire automaticamente le risorse, garantendo che ogni servizio sia sempre disponibile e scalabile in base alle necessità. In un ambiente cloud, Kubernetes permette di bilanciare il carico tra diverse istanze, ottimizzando l'utilizzo delle risorse e migliorando l'affidabilità del sistema. Questo approccio è particolarmente utile per gestire applicazioni distribuite come questa, dove ogni componente deve essere gestito e monitorato in modo indipendente.



Nome	Stato	Tipo	Pod	Spazio dei nomi	Cluster
backend-deployment	OK	Deployment	1/1	default	sensor-cluster
data-sender-deployment	OK	Deployment	1/1	default	sensor-cluster
frontend-deployment	OK	Deployment	1/1	default	sensor-cluster

Figure 6: Lista dei deployments presenti nel cluster Kubernetes.

Un ulteriore vantaggio di Kubernetes è la facilità con cui è possibile integrare servizi come Google Cloud Pub/Sub e Google Cloud Storage, grazie alle numerose integrazioni native offerte da Google Cloud. In questo modo, l'intera architettura può essere gestita in modo uniforme e centralizzato, riducendo la complessità operativa e migliorando la manutenzione del sistema.

2.6 Considerazioni sull'Architettura

L'architettura del progetto è strutturata per essere flessibile, scalabile e altamente affidabile. L'integrazione di componenti cloud come Google Cloud Pub/Sub, Google Cloud Storage e Kubernetes garantisce la robustezza del sistema, mentre il data-sender e il backend forniscono una base solida per l'elaborazione dei dati CANbus. L'utilizzo del cloud non solo semplifica la gestione e il deployment, ma permette anche di risparmiare sui costi operativi, grazie alla scalabilità automatica e al modello di pagamento basato sull'utilizzo. Il frontend, che verrà discusso più avanti, permette di visualizzare i dati in modo interattivo, ma è l'infrastruttura cloud che rende possibile tutto questo, offrendo un sistema in grado di gestire grandi volumi di dati in tempo reale e di adattarsi alle esigenze future del progetto.

3 Dashboard

La dashboard del progetto *SolarCar Cloud Dashboard* è stata sviluppata utilizzando HTML, CSS e JavaScript, con un'architettura semplice e performante per la visualizzazione in tempo reale dei dati provenienti dai sensori della vettura. È stata progettata specificamente per essere visualizzata su uno schermo di piccole dimensioni, che si presume venga installato accanto al volante del veicolo, consentendo al conducente di avere una panoramica dei dati critici della macchina mentre guida.

La dashboard è suddivisa in quattro pagine, ciascuna dedicata a una delle quattro principali categorie di dati raccolti dal veicolo: **temperature**, **voltages**, **engines** e **mppts**. Ogni pagina raccoglie i dati dai rispettivi endpoint del backend e li visualizza in modo chiaro e leggibile. L'obiettivo è fornire al conducente informazioni utili e tempestive, che aiutino a monitorare lo stato del veicolo e a intervenire in caso di anomalie.

3.1 Temperature

La prima pagina della dashboard è dedicata al monitoraggio delle temperature delle batterie del veicolo. Le informazioni vengono recuperate dall'endpoint **/temperatures**, che restituisce i dati relativi alle temperature di ogni batteria installata nel sistema.

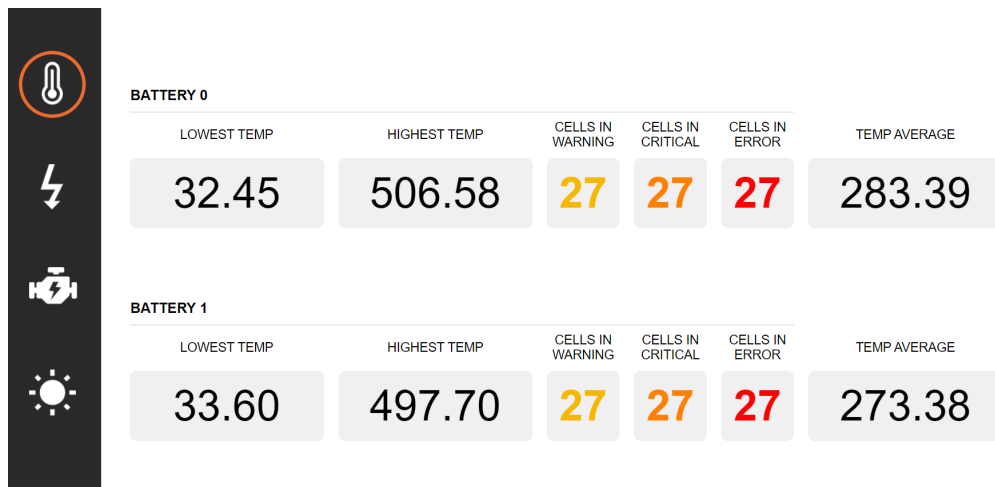


Figure 7: Prima schermata della dashboard.

I dati visualizzati includono:

- Temperatura media del pacco batterie (*avg_temp*): fornisce una panoramica della temperatura media delle batterie.
- Temperatura massima e minima del pacco batterie (*max_temp*, *min_temp*): indicano i valori estremi di temperatura raggiunti.
- Numero celle in warning, critical ed error (*warning_count*, *critical_count*, *error_count*): mostrano il numero di batterie con temperature elevate all'interno del pacco batterie. In base alla temperatura sono stati pensati tre diversi livelli: warning, critical o error. Vanno da una situazione non allarmante ad una di pericolo.

In questa pagina, l'obiettivo è quello di tenere sotto controllo la temperatura delle batterie, che è uno dei parametri più importanti per garantire la sicurezza e la durata del sistema di accumulo dell'energia.

3.2 Tensioni

La seconda pagina della dashboard visualizza le informazioni relative ai voltaggi delle batterie, raccolte dall'endpoint **/voltages**. Questo endpoint fornisce una visione dettagliata della situazione delle batterie in termini di tensione e potenza.

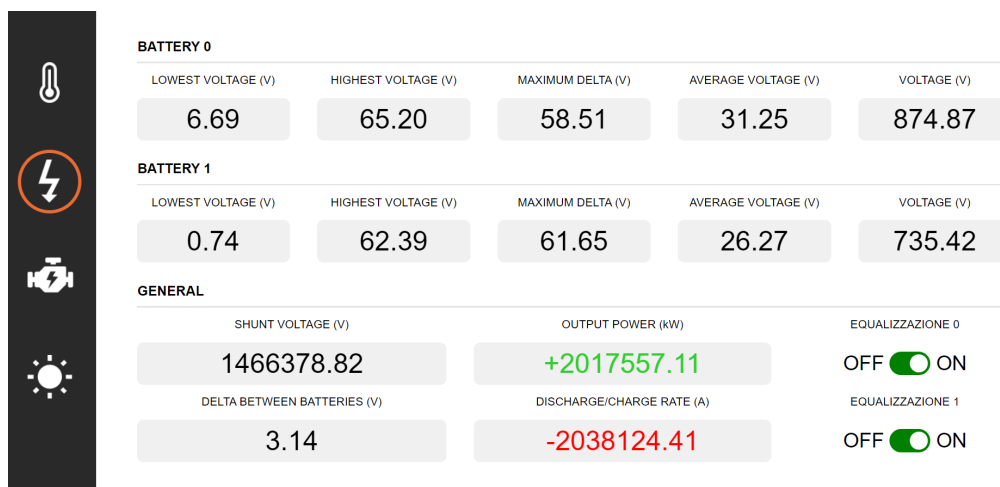


Figure 8: Seconda schermata della dashboard.

I dati principali includono:

- Voltaggio medio e attuale (*average_voltage*, *current_voltage*): mostrano lo stato attuale e storico delle batterie, fondamentale per monitorare l'efficienza del sistema.
- Voltaggio massimo e minimo (*max_voltage*, *min_voltage*): forniscono i valori di riferimento per controllare l'efficienza energetica.
- Delta tra le batterie (*delta_between_batteries*): rappresenta la differenza di tensione tra le batterie, utile per identificare eventuali problemi di bilanciamento.
- Potenza in uscita e shunt voltage (*output_power*, *shunt_voltage*): informazioni che consentono di monitorare la potenza effettiva che viene erogata dal sistema.

Questa pagina permette di avere un quadro preciso dello stato energetico del veicolo, essenziale per la gestione ottimale delle risorse e per garantire l'efficienza durante il funzionamento.

3.3 Motori

La terza pagina si occupa del monitoraggio dei motori elettrici della vettura, utilizzando i dati forniti dall'endpoint `/engines`. I dati riportati sono cruciali per garantire un corretto funzionamento dei motori e per rilevare eventuali problemi in tempo reale.

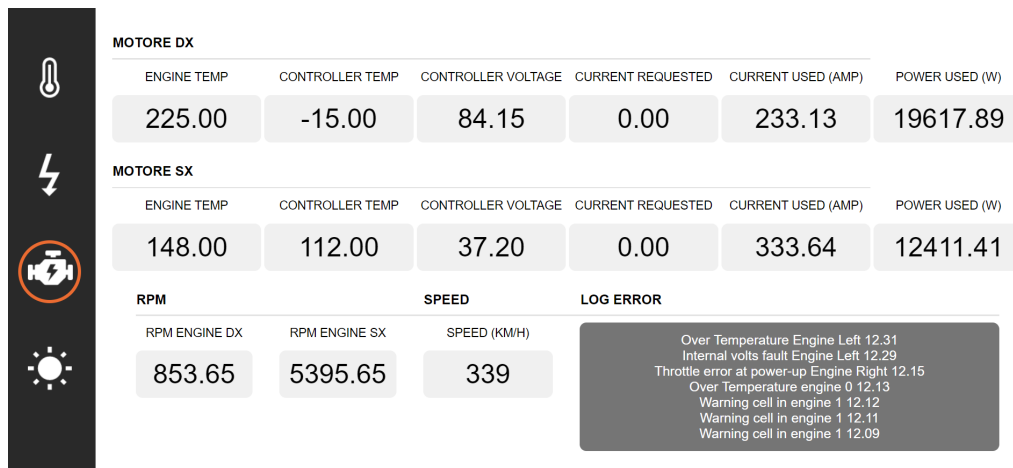


Figure 9: Terza schermata della dashboard.

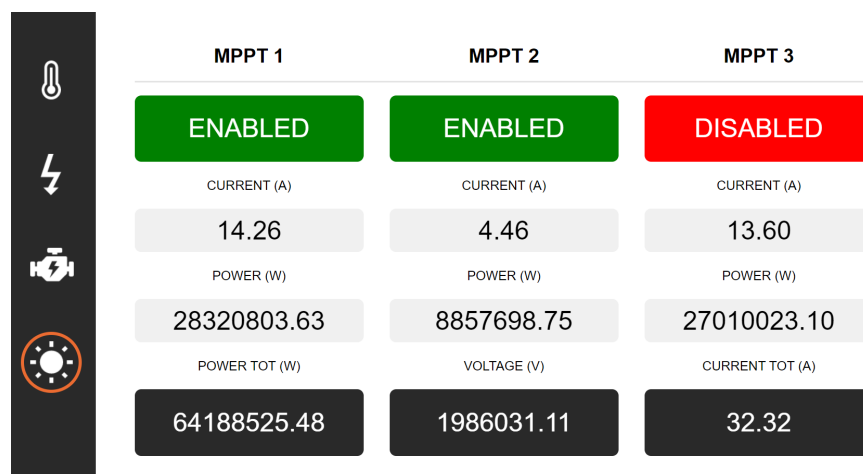
I dati chiave includono:

- Temperature dei controller e dei motori (*controller_temp*, *engine_temp*): forniscono informazioni sulla temperatura sia del motore che del controller, che è importante monitorare per evitare surriscaldamenti.
- Corrente richiesta e utilizzata (*current_requested*, *current_used*): indicano quanta corrente il sistema richiede rispetto a quanta viene effettivamente consumata, utile per valutare l'efficienza dei motori.
- Velocità di rotazione (RPM) (*rpm*): permette di monitorare la velocità di rotazione del motore, dato cruciale per le prestazioni del veicolo.
- Log degli errori (*log_error*): mostra gli errori registrati nei motori, fornendo informazioni dettagliate sugli eventuali malfunzionamenti.

In questa pagina, il focus è sul garantire che i motori funzionino correttamente, evitando surriscaldamenti o malfunzionamenti che potrebbero compromettere le prestazioni o la sicurezza del veicolo.

3.4 MPPTs

L'ultima pagina della dashboard è dedicata agli MPPT (Maximum Power Point Tracker), dispositivi che massimizzano l'efficienza del sistema fotovoltaico del veicolo. I dati sono raccolti dall'endpoint `/mppts` e forniscono informazioni sul rendimento dei pannelli solari e sul flusso di energia verso il sistema di accumulo.



MPPT 1	MPPT 2	MPPT 3
ENABLED	ENABLED	DISABLED
CURRENT (A)	CURRENT (A)	CURRENT (A)
14.26	4.46	13.60
POWER (W)	POWER (W)	POWER (W)
28320803.63	8857698.75	27010023.10
POWER TOT (W)	VOLTAGE (V)	CURRENT TOT (A)
64188525.48	1986031.11	32.32

Figure 10: Quarta schermata della dashboard.

I dati visualizzati includono:

- Corrente e potenza per ogni MPPT (*current*, *power*): misurano quanta energia viene generata dai pannelli solari e gestita da ogni MPPT.
- Stato degli MPPT (*status*): indica se l'MPPT è attivo o meno.
- Potenza totale e voltaggio (*power_tot*, *voltage*, *current_tot*): rappresentano la quantità totale di energia generata, il voltaggio corrente, e la corrente totale.

Questa pagina è fondamentale per monitorare l'efficienza del sistema solare, fornendo un feedback in tempo reale su quanta energia viene generata e se viene utilizzata al meglio delle sue capacità.

3.5 Considerazioni sulla Dashboard

La dashboard, con le sue quattro pagine, fornisce una visione chiara e organizzata dei dati più critici del veicolo, consentendo agli operatori o al conducente di monitorare costantemente lo stato di salute del sistema. Il design semplice e intuitivo, insieme all'uso di tecnologie web standard, rende la dashboard leggera e facilmente accessibile su diversi dispositivi, con particolare attenzione alla sua usabilità su uno schermo di piccole dimensioni montato accanto al volante.

4 Costi dei servizi Cloud

Nello sviluppo e nell'implementazione del progetto *SolarCar Cloud Dashboard*, ho utilizzato vari servizi di Google Cloud per garantire una piattaforma scalabile, affidabile e performante. In questa sezione, analizzerò i principali costi associati all'utilizzo di questi servizi, basandomi sulle stime relative ai consumi durante lo sviluppo e le possibili spese operative nel lungo termine.

I servizi principali coinvolti nel progetto sono:

- **Google Kubernetes Engine (GKE)**
- **Google Cloud Pub/Sub**
- **Google Cloud Storage**
- **Google Cloud Functions**

4.1 Google Kubernetes Engine (GKE)

Google Kubernetes Engine è stato utilizzato per gestire i container delle diverse componenti del progetto, inclusi il backend, il frontend e i data-sender. Kubernetes è essenziale per garantire un ambiente distribuito e orchestrato, con gestione automatica del carico e ridondanza.

4.1.1 Costi di Google Kubernetes Engine

I costi di GKE dipendono principalmente dalle risorse computazionali utilizzate (CPU, memoria e numero di nodi) e dal tempo in cui i cluster sono attivi. Nel mio caso, per gestire l'applicazione durante lo sviluppo, ho optato per un cluster standard con due nodi, sufficienti a gestire le richieste simulate dai data-sender e le richieste di aggiornamento da parte del frontend.

- **Costo per cluster di nodi:** Google addebita un costo fisso di circa \$0.10 per ora di uptime del cluster, a cui si aggiunge il costo per ciascun nodo utilizzato.
- **Costo per nodo:** Ogni nodo dipende dalla configurazione scelta, in particolare dalle risorse computazionali (CPU e memoria). Una configurazione con macchine virtuali di tipo e2-medium, che offre 2 vCPU e 4 GB di memoria, ha un costo di circa \$0.033 per vCPU all'ora e circa \$0.004 per GB di memoria all'ora.

Con un uptime medio di 24 ore giornaliere e due nodi attivi per circa 30 giorni, i costi stimati per l'utilizzo di GKE si aggirano attorno a **\$90-\$100 al mese**, variabili a seconda dell'effettiva necessità di risorse e carichi di lavoro.

4.2 Google Cloud Pub/Sub

Google Cloud Pub/Sub è stato il servizio utilizzato per gestire la messaggistica asincrona tra i data-sender e il backend. I messaggi inviati dal data-sender venivano pubblicati su un topic Pub/Sub, che permetteva al backend di processarli e di aggiornarli nei diversi endpoint, oltre a memorizzare alcuni messaggi su Google Cloud Storage tramite una Cloud Function.

4.2.1 Costi di Google Cloud Pub/Sub

Il modello di pricing di Pub/Sub si basa su due fattori principali:

- **Costo per messaggio pubblicato:** Google Cloud offre una quota gratuita di 10 GB al mese per la pubblicazione e il trasporto dei messaggi.
- **Costo per messaggio recuperato:** Anche per il recupero dei messaggi è prevista una quota gratuita di 10 GB al mese.

Considerando che i pacchetti CANbus inviati dai data-sender sono di piccole dimensioni (circa 20 byte per messaggio), durante le fasi di sviluppo non ho superato la quota gratuita di 10 GB/mese. Tuttavia, in un ambiente produttivo, a seconda del volume di dati trasmessi, i costi possono variare sensibilmente. Per un volume limitato di messaggi, i costi stimati rimangono **\$0** se contenuti nella soglia gratuita, ma possono aumentare in base al numero e al peso dei messaggi.

4.3 Google Cloud Storage

Google Cloud Storage è stato utilizzato come archivio per la memorizzazione permanente dei messaggi CANbus ricevuti. Una Cloud Function si attivava su ogni nuovo messaggio Pub/Sub e lo memorizzava all'interno di un bucket di storage, suddividendo i messaggi per categorie in sottocartelle.

4.3.1 Costi di Google Cloud Storage

Il modello di costo di Google Cloud Storage si basa su diversi fattori, tra cui:

- **Spazio di archiviazione utilizzato:** il costo dipende dalla classe di storage scelta (Standard, Nearline, Coldline, Archive). Per il progetto, ho utilizzato la classe Standard per garantire accesso rapido ai dati. Il costo medio per lo storage Standard è di circa \$0.026 per GB al mese.
- **Operazioni di archiviazione e lettura:** ogni operazione di scrittura (PUT) o lettura (GET) sui file memorizzati ha un costo associato, sebbene le operazioni iniziali siano coperte da una quota gratuita.

Considerando che i messaggi CANbus sono di piccole dimensioni e che vengono archiviati in formato più leggibile, il volume di dati memorizzato è relativamente ridotto. Per un utilizzo medio di 10-20 MB di dati al mese, i costi di Cloud Storage si stimano essere inferiori a **\$1 al mese**, rendendolo un servizio altamente conveniente per il progetto.

4.4 Google Cloud Functions

Le Cloud Functions sono state utilizzate per automatizzare il processo di scrittura dei messaggi ricevuti su Pub/Sub all'interno di Google Cloud Storage. La funzione veniva eseguita ogni volta che un nuovo messaggio veniva pubblicato sul topic Pub/Sub.

4.4.1 Costi di Google Cloud Functions

I costi di Cloud Functions si basano sul numero di esecuzioni e sul tempo di esecuzione, misurato in gigabyte-secondi (GB/s) e gigahertz-secondi (GHz/s). Ogni mese, Google offre una quota gratuita di 2 milioni di invocazioni e 400.000 GB/s di esecuzione gratuita.

Durante la fase di sviluppo e test, le invocazioni della funzione sono state limitate a poche centinaia al giorno, il che rientra abbondantemente nella soglia gratuita. Anche in un ambiente produttivo, con un volume maggiore di messaggi da processare, i costi di Cloud Functions rimangono bassi, con una stima mensile che non supera i **\$2-\$3 al mese** per l'esecuzione delle funzioni.

4.5 Stime Complessive dei Costi

Sommando i costi stimati dei vari servizi utilizzati nel progetto *SolarCar Cloud Dashboard*, le spese operative mensili si suddividono approssimativamente come segue:

- **Google Kubernetes Engine (GKE): \$90 - \$100/mese**
- **Google Cloud Pub/Sub: \$0 - \$5/mese** (a seconda del volume di messaggi)
- **Google Cloud Storage: \$0.50 - \$2/mese**
- **Google Cloud Functions: \$1 - \$3/mese**

Il totale complessivo stimato per il mantenimento dell'infrastruttura cloud è compreso tra **\$95 e \$110 al mese**, con variazioni legate al volume di dati e alla scalabilità dell'applicazione. Questi costi possono crescere all'aumentare dell'uso del sistema e dell'accesso ai dati, ma rimangono comunque competitivi grazie alla possibilità di gestire i carichi in modo dinamico e scalabile su Google Cloud.

4.6 Considerazioni sui Costi di Google Cloud

Rispetto ad altre piattaforme cloud, Google Cloud offre una combinazione bilanciata di prestazioni e prezzi competitivi. Servizi come Pub/Sub e Cloud Functions sono particolarmente vantaggiosi per i progetti basati su microservizi, grazie al loro modello di prezzo basato sull'utilizzo effettivo, che riduce i costi operativi in fasi di sviluppo o per carichi di lavoro ridotti.

5 CI/CD: Continuous Integration e Continuous Deployment

5.1 Introduzione alla CI/CD

Nel contesto del progetto **SolarCar Cloud Dashboard**, è stata implementata una pipeline di **Continuous Integration (CI)** e **Continuous Deployment (CD)** con l'obiettivo di automatizzare il processo di build, testing e rilascio delle modifiche al codice. La pipeline CI/CD, eseguita tramite **GitHub Actions**, assicura che ogni modifica venga automaticamente integrata e distribuita, riducendo i tempi di rilascio e mantenendo l'infrastruttura sempre aggiornata.

5.2 Strumenti utilizzati

La pipeline utilizza una serie di strumenti chiave per il deployment e la gestione dell'applicazione in un ambiente cloud:

- **Google Cloud SDK**: per interagire con Google Cloud e gestire le risorse.
- **Docker**: per il packaging delle applicazioni in container.
- **Google Container Registry (GCR)**: per memorizzare le immagini Docker delle diverse componenti.
- **Kubernetes**: per orchestrare i container in esecuzione su Google Kubernetes Engine (GKE).
- **kubectl**: per gestire i deployment su Kubernetes.
- **GitHub Actions**: come piattaforma CI/CD per automatizzare il flusso di lavoro.

5.3 Pipeline CI/CD

Ogni volta che viene effettuato un push sul branch main, la pipeline CI/CD viene eseguita per effettuare la build, il push delle immagini su Google Cloud e il deployment delle modifiche nel cluster Kubernetes.

La pipeline è configurata come segue:

1. **Checkout del codice:** La pipeline inizia con il checkout del codice dal branch main

```
– name: Checkout code
  uses: actions/checkout@v2
```

2. **Set up Google Cloud SDK:** Viene configurato l'ambiente per interagire con Google Cloud SDK, assicurando che sia disponibile la versione più recente

```
– name: Set up Google Cloud SDK
  uses: google-github-actions/setup-gcloud@v2
  with:
    project_id: solarcaroncloud
    version: "latest"
```

3. **Autenticazione a Google Cloud:** L'autenticazione avviene utilizzando una chiave di servizio salvata come segreto nel repository GitHub

```
– name: Authenticate to Google Cloud
  uses: google-github-actions/auth@v2
  with:
    credentials_json: ${secrets.GCP_SA_KEY}
```

4. **Autenticazione di Docker su Google Cloud Registry:** Viene configurato Docker per effettuare il push delle immagini sul Google Container Registry (GCR)

```
– name: Authenticate Docker to Google Cloud
  Registry
  run: gcloud auth configure-docker
```

5. **Verifica dell'autenticazione** Un passaggio di verifica per controllare che l'autenticazione sia avvenuta correttamente

```
– name: Check GCP authentication
  run: |
    gcloud auth list
    gcloud config list project
```

6. **Build delle immagini Docker:** Vengono create le immagini Docker per ogni componente del progetto (backend, frontend e data-sender)

```

- name: Build Docker images
  run: |
    docker build -t
      gcr.io/solarcaroncloud/backend:latest
      ./Backend
    docker build -t
      gcr.io/solarcaroncloud/frontend:latest
      ./Frontend
    docker build -t
      gcr.io/solarcaroncloud/sender:latest
      ./Data-Sender

```

7. **Push delle immagini su Google Container Registry:** Una volta completata la build, le immagini vengono pushate su GCR

```

- name: Push Docker images to Google Container
  Registry
  run: |
    docker push
      gcr.io/solarcaroncloud/backend:latest
    docker push
      gcr.io/solarcaroncloud/frontend:latest
    docker push
      gcr.io/solarcaroncloud/sender:latest

```

8. **Set up kubectl:** Viene configurato kubectl, lo strumento per la gestione dei deployment Kubernetes

```

- name: Set up kubectl
  uses: azure/setup-kubectl@v4

```

9. **Recupero delle credenziali del cluster GKE:** Le credenziali del cluster GKE vengono recuperate per consentire a kubectl di interagire con esso

```

- name: Get GKE credentials
  run: |
    gcloud container clusters get-credentials
      sensor-cluster --region us-central1-a

```

10. **Installazione del plugin di autenticazione GKE:** Questo step garantisce che il plugin di autenticazione GKE sia installato

```
– name: Install gke-gcloud-auth-plugin
  run: gcloud components install
      gke-gcloud-auth-plugin
```

11. **Deploy su Kubernetes** Viene effettuato il deploy delle applicazioni nel cluster Kubernetes utilizzando i file di configurazione

```
– name: Deploy to Kubernetes
  run: |
    kubectl apply -f backend-deployment.yaml
    kubectl apply -f frontend-deployment.yaml
    kubectl apply -f sender-deployment.yaml
```

12. **Rollout Restart dei deployment:** Infine, viene eseguito il rollout restart per applicare le modifiche in ogni componente

```
– name: Rollout Restart Deployments
  run: |
    kubectl rollout restart deployment
    backend-deployment
    kubectl rollout restart deployment
    frontend-deployment
    kubectl rollout restart deployment
    data-sender-deployment
```

5.4 Considerazioni su CI/CD

L'integrazione di una pipeline CI/CD completamente automatizzata ha permesso di semplificare e velocizzare il processo di deployment per il progetto SolarCar Cloud Dashboard. Ogni modifica al codice viene automaticamente buildata, testata e distribuita nel cluster Kubernetes, garantendo continuità nel processo di sviluppo e aggiornamenti rapidi e affidabili in produzione.

6 Conclusioni

Il progetto *SolarCar Cloud Dashboard* rappresenta un sistema innovativo per la gestione e il monitoraggio di dati critici relativi a un veicolo alimentato a energia solare. Utilizzando una combinazione di tecnologie cloud, come Google Kubernetes Engine (GKE), Pub/Sub, Cloud Storage e Cloud Functions, il progetto ha implementato un'architettura scalabile e affidabile, capace di processare in tempo reale i dati simulati da un modulo CANbus virtuale (data-sender) e renderli accessibili tramite una dashboard interattiva.

Il sistema, nella sua forma attuale, simula l'invio di pacchetti CANbus da parte di un data-sender, ma con poche modifiche potrebbe essere adattato per ricevere i dati direttamente da un veicolo reale. La dashboard, sviluppata in HTML, CSS e JavaScript, è progettata per essere visualizzata su uno schermo di piccole dimensioni, posizionato vicino al volante, e permette di monitorare parametri vitali della vettura, come temperature, tensioni delle batterie, stato dei motori e performance dei pannelli solari (MPPTs).

Dal punto di vista tecnologico, l'infrastruttura cloud si è dimostrata flessibile ed efficiente, con la capacità di scalare automaticamente in base al carico di lavoro, garantendo al tempo stesso un basso costo operativo. Questo rende la soluzione non solo adatta a veicoli solari, ma potenzialmente estensibile a una vasta gamma di veicoli intelligenti che richiedono monitoraggio remoto e gestione avanzata dei dati.

6.1 Sviluppi Futuri

Un passo logico successivo per il progetto sarebbe l'integrazione del sistema con una macchina reale, sostituendo il data-sender simulato con il modulo CANbus effettivo del veicolo. Questo aggiornamento permetterebbe di raccogliere dati reali sulle performance della vettura, che potrebbero essere processati e visualizzati in tempo reale sulla dashboard.

Il ruolo del cloud in questo scenario diventerebbe cruciale per due motivi principali:

- **Monitoraggio remoto:** L'infrastruttura cloud potrebbe essere utilizzata per inviare i dati raccolti dal veicolo non solo allo schermo interno, ma anche a una piattaforma remota. In questo modo, gli operatori potrebbero monitorare lo

stato della macchina da qualunque luogo, analizzando le prestazioni del veicolo e rilevando eventuali anomalie anche quando il veicolo è in movimento.

- **Analisi e ottimizzazione:** I dati storici raccolti e memorizzati in Google Cloud Storage potrebbero essere analizzati nel tempo per identificare pattern e tendenze di utilizzo, fornendo informazioni utili per migliorare l'efficienza della vettura. Inoltre, il sistema potrebbe generare report di manutenzione predittiva, suggerendo interventi prima che si verifichino problemi critici.

Con l'integrazione della macchina reale e una maggiore automazione, il sistema *Solar-Car Cloud Dashboard* potrebbe rappresentare una soluzione altamente efficiente per la gestione e il monitoraggio di flotte di veicoli solari, consentendo una gestione più intelligente delle risorse energetiche e un miglioramento delle prestazioni complessive del mezzo.

6.2 Conclusioni Finali

In conclusione, il progetto ha dimostrato come l'adozione di tecnologie cloud possa portare vantaggi significativi nel monitoraggio in tempo reale di un veicolo a energia solare. Grazie alla flessibilità della dashboard e alla scalabilità della soluzione cloud, il sistema è in grado di adattarsi a diverse esigenze, sia per singoli veicoli che per flotte più complesse.

Gli sviluppi futuri del progetto possono estenderne ulteriormente l'efficacia, permettendo di sfruttare appieno il potenziale del cloud per applicazioni automotive avanzate, migliorando così la sicurezza, l'efficienza e la sostenibilità delle vetture.