Progetto finale

UF17 – Reti e bus di campo

Arkeos: Controllo distribuito con Node-RED, Raspberry Pi, ESP32 e PLC Omron.

Alessandro Miori

MAT Meccatronica e Robotica Industriale
Luglio 2025

Indice

Indice

00 Introduzione

Viviamo in un'epoca in cui **automazione**, **connettività** e **facilità d'uso** si incontrano, diventando elementi **essenziali** di qualsiasi sistema moderno. L'integrazione di dispositivi come microcontrollori, PLC industriali e dispositivi mobili rappresenta oggi un **traguardo significativo** nel campo dell'automazione.

Arkeos nasce con l'intento di raggiungere questo traguardo, offrendo una **soluzione semplice**, modulare ed efficace, in grado di unire il mondo dell'elettronica embedded, il controllo industriale e la gestione intelligente e visiva dei processi.
Il tutto inserito in un **ecosistema compatto**, configurabile e accessibile tramite un'**interfaccia web**.

L'intero progetto è stato sviluppato partendo dalle reali necessità di **controllo remoto**, **comunicazione sicura**, **gestione centralizzata** ed estetica moderna, fino a giungere ad una struttura completa e funzionante. Le tecnologie scelte si basano sulla necessità di utilizzare standard **open-source** e protocolli leggeri, per garantire la massima flessibilità e **interoperabilità** tra i vari sistemi.

Il progetto ha l'obiettivo di realizzare un sistema di controllo **modulare** che possa essere interfacciato a **qualsiasi macchina**, stazione robotica o isola industriale standardizzata, per renderne la supervisione più semplice ed intuitiva.

Arkeos è infatti pensato per poter essere **personalizzato** in base alla soluzione richiesta, essendo dotato di una dashboard programmata ad hoc, e sensoristica personalizzata on-board e off-board.

Sfruttando ed interfacciando **3 principali dispositivi**, Arkeos si colloca in uno scenario caratterizzato dall'interconnessione e la **decentralizzazione** dei sistemi, tema molto importante e che lascia spazio a modelli più distribuiti e liberi.

01 Architettura

Il sistema si basa su un'architettura a **3 livelli** che garantisce il **controllo** e il **monitoraggio** in **tempo reale** della macchina attraverso una dashboard web costruita appositamente per la soluzione.

Il tutto è orchestrato da un **server Node-RED**, installato e operato da un **Raspberry Pi5**, che funge da nodo centrale per la comunicazione e gestione logica di Arkeos. I valori ambientali della macchina sono trasmessi in tempo reale attraverso **ESP32** con protocollo **MQTT**, ai seguenti topic:

- Temperatura : `arkeos/env/temp`
- Umidità : `arkeos/env/hum`
- Presenza di gas : `arkeos/env/gas`
- Vibrazioni : `arkeos/env/vib`

Un **PLC** (in questo caso Omron NX102-1200) comanda la stazione fisica e comunica a sua volta con Arkeos; quest'ultimo infatti è in grado di attivare/fermare il ciclo produttivo, mandare in emergenza il sistema automaticamente in base a valori ambientali critici e comunicare lo stato dei singoli stati del ciclo, il tutto tramite protocollo **OPC UA**.

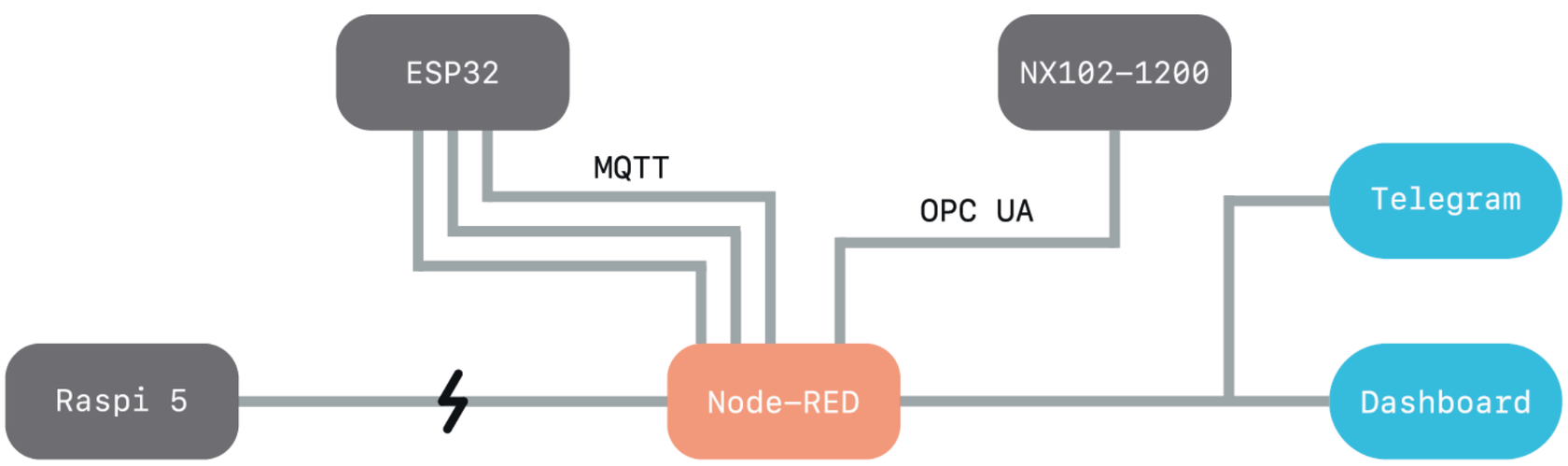

Figura **01.00**: Struttura logica di Arkeos.

Arkeos è pensato per avere una **struttura modulare**: ogni componente può essere aggiunto o rimosso senza impattare il sistema centrale, affinchè Pi5 e Node-RED siano sempre in comunicazione tra loro. Possono essere quindi interfacciati più microcontrollori come ESP32 o Arduino per la raccolta dati o attuazione a bordo macchina.

02 Tecnologie e protocolli

Arkeos sfrutta una combinazione di tecnologie open-source e protocolli industriali per garantire la facilità d'integrazione del sistema all'interno di una linea di produzione già attiva.

02.1 Node-RED

Node-RED è una piattaforma di sviluppo open-source basata su **flussi** (flows), pensata per la programmazione **event-driven**[1]; in Arkeos Node-RED svolge il ruolo di **motore logico centrale**:

- Riceve dati da ESP32
- Interroga o scrive dati sul PLC
- Gestisce e aggiorna in tempo reale la dashboard
- Esegue tutte le funzioni di logica che gestiscono la comunicazione e l'handling dei dati

I plugin esterni utilizzati sono:

- `node_red_dashboard` : Gestione della dashboard e dei widget.
- `node_red_contrib_telegrambot` : Comunicazione con Arkeosbot (vedi sez. **06**).
- `node_red_contrib_opcua` : Comunicazione con il PLC via OPC UA.

02.2 MQTT

Message Queuing Telemetry Transport, è un protocollo di comunicazione open-source leggero basato sulla logica del **publish/subscribe**, ideale per dispositivi embedded come ESP32; in Arkeos viene utilizzato per **inviare dati** dai sensori a bordo macchina collegati ad ESP32 e trasmetterli al sistema in tempo reale.

02.3 OPC UA

Open Platform Communications Unified Architecture, è un protocollo di comunicazione ormai standardizzato tra i sistemi di automazione industriale; in Arkeos viene utilizzato per **comunicare con il PLC** che comanda la macchina.

[1]**event-driven**: ogni azione avviene in risposta ad un evento, come un messaggio ricevuto o un cambio di stato. Il flusso si attiva solo quando serve, rendendo il sistema reattivo ed efficiente.

03 Funzionalità

03.1 Controllo remoto degli attuatori (Non attivo)

Tramite la dashboard, l'utente può **attivare** o **disattivare** in tempo reale **relè** collegati all'ESP32 o segnali digitali/analogici del PLC.

03.2 Monitoraggio in tempo reale

I dati provenienti dai sensori sono visualizzati e messi a **grafico** sulla dashboard con un refresh rate settabile (default=1s).

03.3 Automazioni basate sulla logica

Se per esempio la temperatura supera una certa soglia, viene attivato un relè o segnale al PLC che mette in funzione un sistema di raffreddamento. La stessa cosa vale per le **emergenze**, che vengono **attivate automaticamente** in base ai parametri ambientali.

03.4 Controllo attraverso bot Telegram

Arkeos può essere **monitorato** e **controllato** quasi completamente attraverso il bot di Telegram (Arkeosbot), che permette di pilotare lo start e lo stop del ciclo, richiedere lo **stato** della macchina, gli **allarmi** attivi, i **log** completi della macchina, e ricevere **notifiche** in tempo reale in caso di allarmi o emergenze.

03.5 Logging dei dati e allarmi

Tutti le azioni che vengono effettuate attraverso Arkeos, vengono **salvate in un log** che viene visualizzato sulla dashboard e che può essere svuotato o salvato come file .csv; lo stesso vale per gli allarmi, che sono salvati anche in un **database** a parte dedicato solo ad esse, anche quest'ultimo può essere svuotato o salvato come file .csv.

04 Logica del sistema: flussi Node-RED

La struttura dei flussi di Node-RED di Arkeos è divisa in **4 sezioni** principali:

- `04.01` **Ricezione dati**: flussi che gestiscono la ricezione dei dati dall'ESP32 con MQTT e li **salvano**, **controllano** in tempo reale i valori per generare allarmi, emergenze, o automazioni collegate ai parametri ambientali. Sono a loro volta suddivisi in **temperatura**, **umidità**, **gas** e **vibrazioni**, ma possono essere personalizzati.

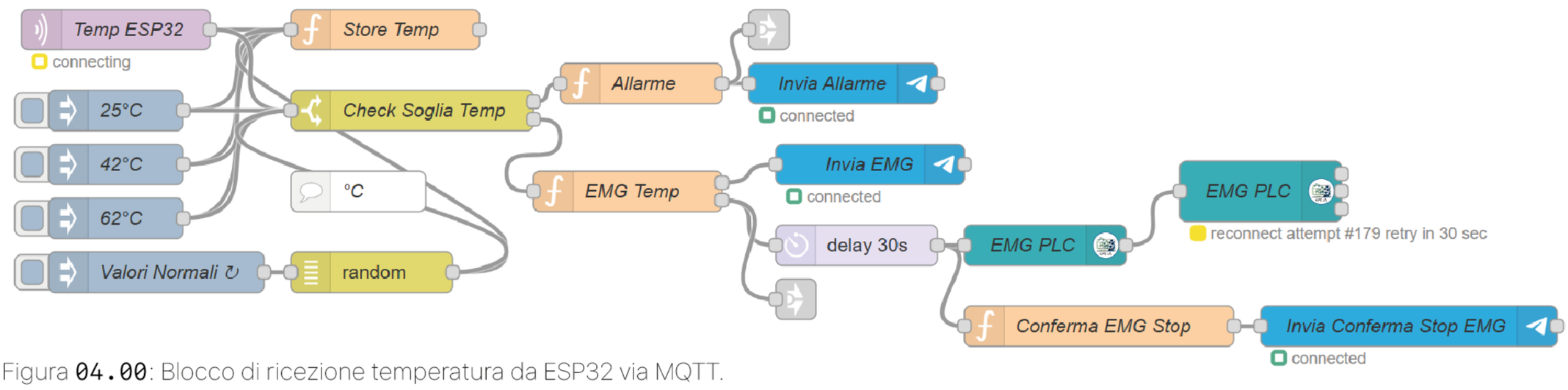

Figura `04.00`: Blocco di ricezione temperatura da ESP32 via MQTT.

Questo è uno dei 4 blocchi dedicati alla ricezione dei dati: composto da un `mqtt_in` che riceve informazioni sul topic `arkeos/env/temp`, e le salva nella variabile globale `last_temp`, attraverso la funzione **Store Temp**:

```
flow.set('last_temp', msg.payload);
```

Il valore viene controllato tramite uno **switch node**, che rileva **anomalie** nei valori rispetto a delle **soglie impostate** generando in questo caso un'allarme se la temperatura supera i 40°C, inviando una notifica all'utente tramite Arkeosbot, oppure un'emergenza nel caso in cui la temperatura superi i 60°C, lasciando un avviso all'operatore che il sistema si spegnerà in automatico dopo 10s.

04 Logica del sistema: flussi Node-RED

- **04.02 Gestione bot Telegram**: flussi che controllano i messaggi in entrata ad Arkeosbot, nel caso in cui venga riconosciuto un comando valido, le funzioni **Risposta /comando** rispondono in chat con le informazioni richieste dall'utente.

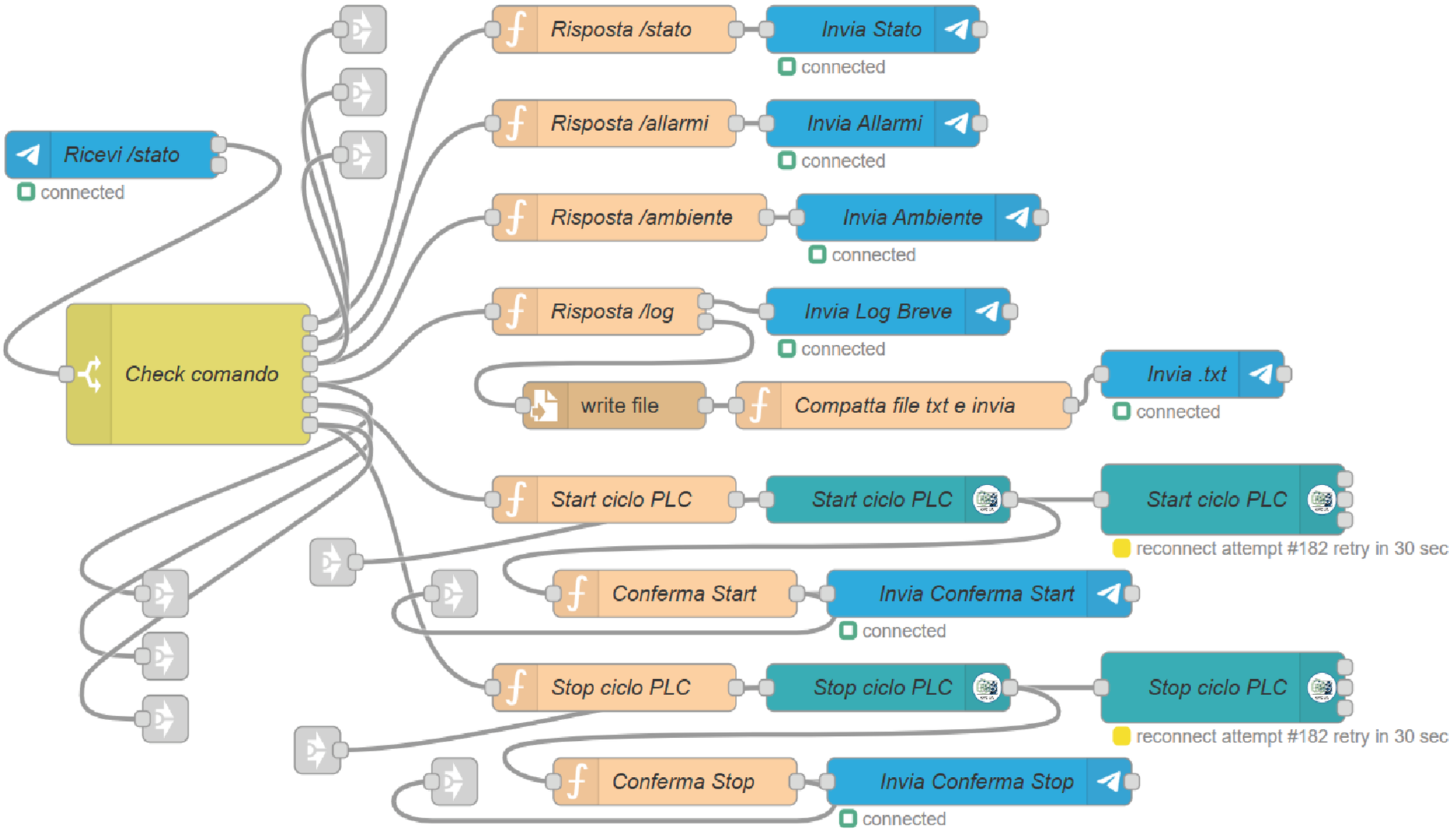

Figura 04.01: Blocco di gestione dei comandi di Arkeosbot.

Il blocco è formato da un `telegram_receiver`, che ritorna in output l'ultimo messaggio ricevuto dall'utente in formato JSON.
Il flow **Check comando** è un parser che estrae **`msg.payload.content`** (il contenuto effettivo del messaggio) e lo confronta con la lista di comandi validi.

Una volta riconosciuto e verificato il comando, si procede con la risposta da parte del bot, che può essere una semplice risposta come nel caso di `/status`, o un processo più complesso che comprende la **scrittura di una variabile** sul PLC o il **download di un file**.

04 Logica del sistema: flussi Node-RED

• **04.03 Variabili, log e allarmi**: le variabili sono gestite da una struttura che alloca dei valori ad esse rendendole leggibili/scrivibili da ogni blocco funzione nel progetto: vengono utilizzati

```
flow.set('variabile', valore);
flow.get('variabile');
```

`set` per scrivere una variabile e `get` per leggerla da qualsiasi altro blocco.

I gruppi di ricezione dati scrivono sulle variabili `last_temp`, `last_hum`, `last_gas` e `last_vib`.

Gli stati sono salvati sulle variabili `statoMacchina`, `connessionePLC`, `connessioneESP`, `connessionePi`.

Gli allarmi sono scritti su un array composto da variabili:

```
let nuovoAllarme = {
      messaggio: `AL101 Temperatura elevata (${temp}°C)`,
      timestamp: new Date().toLocaleString()
};
```

che vengono aggiunte ogni volta che un'azione è stata compiuta con un massimo di 20 entrate:

```
allarmi.push(nuovoAllarme);
if (allarmi.length > 20) allarmi.shift();
flow.set('allarmi', allarmi);
```

Il log è gestito da 2 array globali: `logs` viene popolato con un massimo di 20 entrate e quando va in overflow le variabili inizieranno ad essere spinte su `logsFull`, che rappresenta una variante estesa e completa del log.

Entrambi gli array vengono riempiti attraverso delle entrate:

```
let entry = `${now}-AL101 Temperatura elevata (${temp}°C)`;
logs.push(entry);
if (logs.length > 20) logs.shift();
logsFull.push(entry);

flow.set('logs', logs);
flow.set('logs_full', logsFull);
```

04 Logica del sistema: flussi Node-RED

- **04.04** Dashboard: Flussi che strutturano, costruiscono, popolano, e gestiscono la dashboard in tempo reale, a loro volta suddivisi in Init (inizializzazione), Welcome (mostra le informazioni generali sullo stato della macchina), Ambiente (contiene i valori ambientali e i loro grafici), Allarmi (contiene il log degli allarmi), Logs (contiene il log di tutte le azioni di Arkeos) e infine Comandi Manuali (contiene start/stop ciclo e info su Arkeosbot).

Il gruppo d'inizializzazione contiene i due blocchi principali che gestiscono il CSS della dashboard, Arkeos Custom CSS contiene tutto il codice che modifica la parte grafica della dashboard, mentre Logo Arkeos gestisce il posizionamento del logo sopra ad essa.

Il gruppo welcome è composto da una serie di **ui_text** che vengono aggiornati (default=1s) con delle funzioni che scrivono e leggono gli stati della macchina e dei valori ambientali.

Figura **04.02**: Gruppo welcome della dashboard.

04 Logica del sistema: flussi Node-RED

Il gruppo ambiente è composto da 3 blocchi che aggiornano i valori ambientali e li mostrano su un grafico nel tempo , mostrando un LED che ne stabilisce lo stato (verde OK, giallo Allarme, rosso Critico).

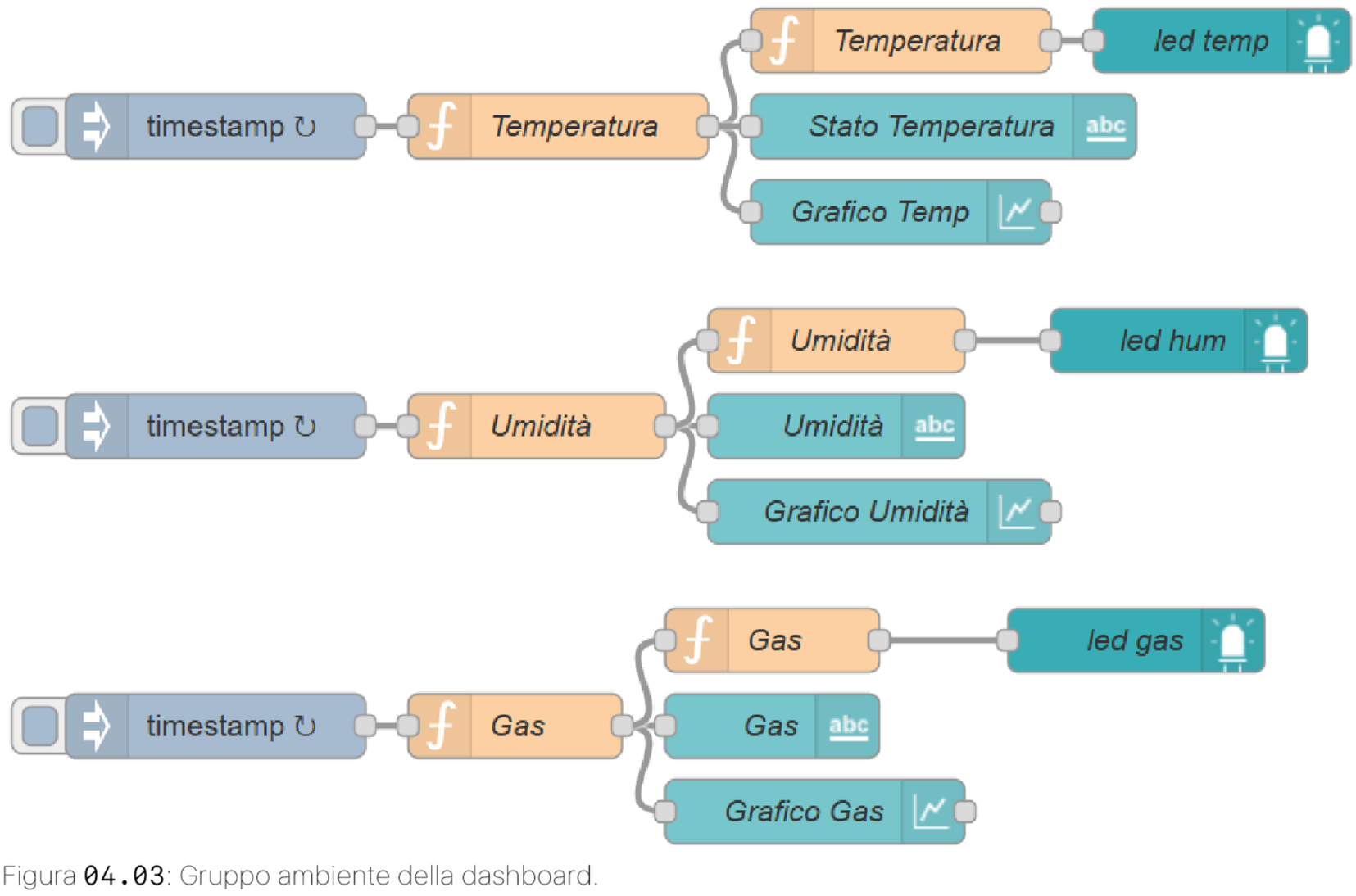

Figura 04.03: Gruppo ambiente della dashboard.

04 Logica del sistema: flussi Node-RED

Il gruppo allarmi è composto da una **tabella** che mostra lo **storico degli allarmi** di Arkeos, una **stringa** che indica l'**ultimo allarme** attivo nella macchina, due pulsanti che permettono di svuotare lo storico oppure scaricare il file completo e infine un campo di testo che permette di inserire il percorso per il download.

Figura 04.04: Gruppo allarmi della dashboard.

Il gruppo logs è composto ugualmente al gruppo allarmi, con l'assenza della stringa dell'ultimo log, poichè molto poco utile.

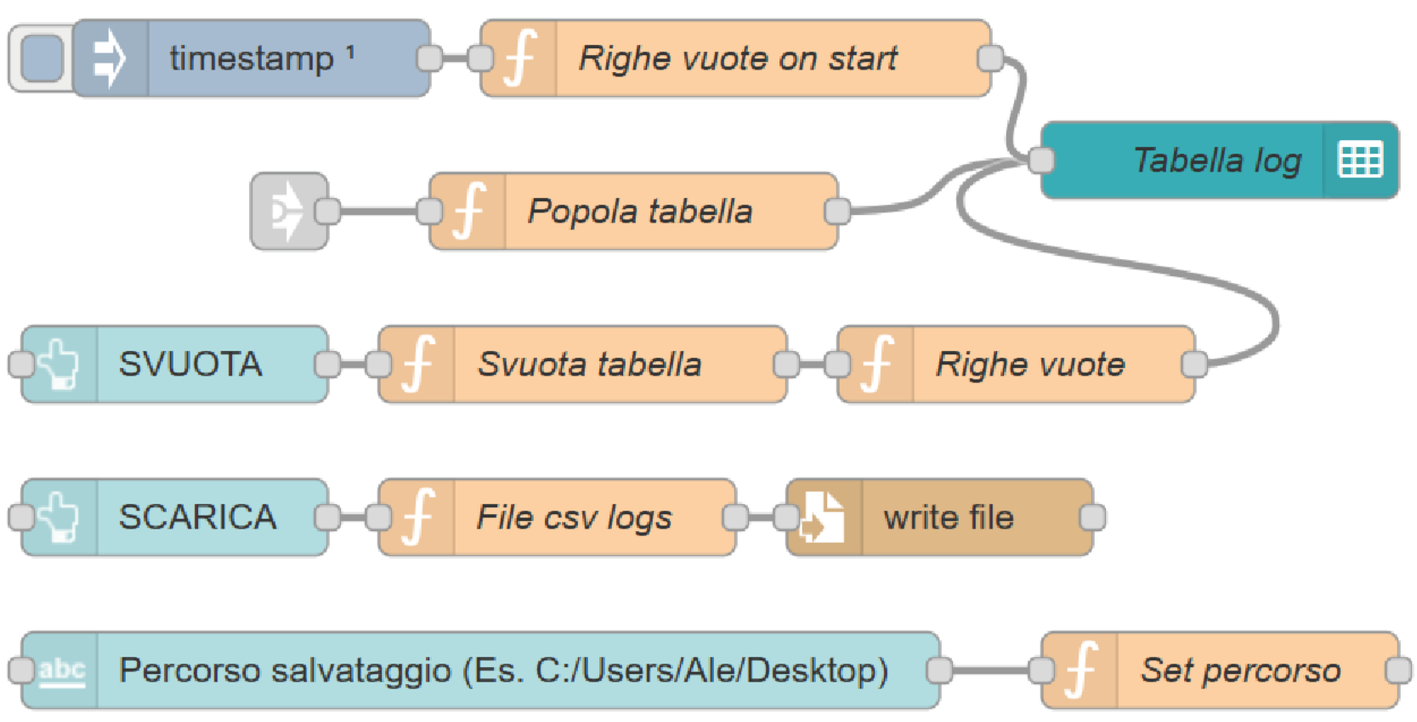

Figura 04.05: Gruppo logs della dashboard.

04 Logica del sistema: flussi Node-RED

Il gruppo comandi manuali è composto da due pulsanti di **start ciclo** e **stop ciclo**, seguiti da uno speciale meccanismo di controllo per la chat con Arkeosbot.

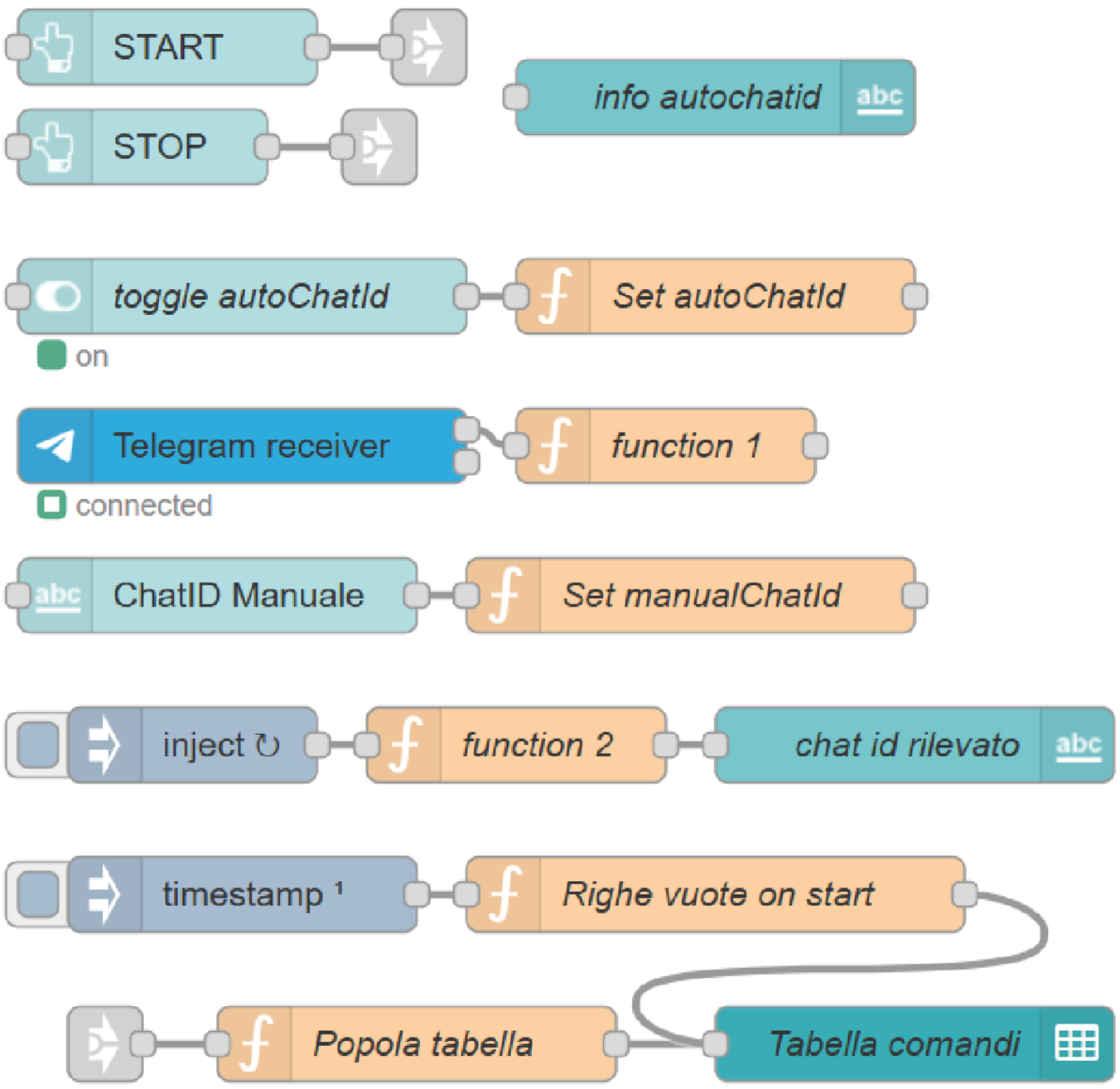

Figura 04.04: Gruppo comandi manuali della dashboard.

Switch **AutoChatID**: se ON, Arkeos riconoscerà **automaticamente** il ChatID[1] dell'utente grazie al suo **ultimo messaggio** inviato al bot, mentre se OFF Arkeos utilizzerà il ChatID che verrà inserito **manualmente** nel campo di testo ChatID Manuale.

`
È inoltre presente una tabella che salva uno storico dei comandi inviati ad Arkeosbot, insieme ai ChatID degli invianti.

[1]ChatID: un numero univoco che identifica una conversazione specifica su Telegram.
Il bot ha bisogno del chatID per sapere a chi inviare messaggi o da chi li riceve.

05 Dashboard

Uno degli aspetti fondamentali di Arkeos è la realizzazione di una **dashboard web-based**, progettata per essere **semplice**, **intuitiva** ed accessibile da qualsiasi dispositivo in rete.

È stata realizzata interamente con **Node-RED,** sfruttando i flows disponibili di default e alcuni esterni come ui_led2 e ui_table.

L'obiettivo è quello di interfacciare Arkeos con l'utente fornendo un esperienza moderna, fluida, e **personale**.

La dashboard è divisa in 7 cards, la prima contiene il **gruppo welcome** e tutti gli stati della macchina e dei suoi parametri, le seguenti 3 contengono i **valori ambientali** e i grafici, altre due contengono **allarmi e logs**, mentre l'ultima contiene i **comandi manuali** e le informazioni riguardo ad Arkeosbot.

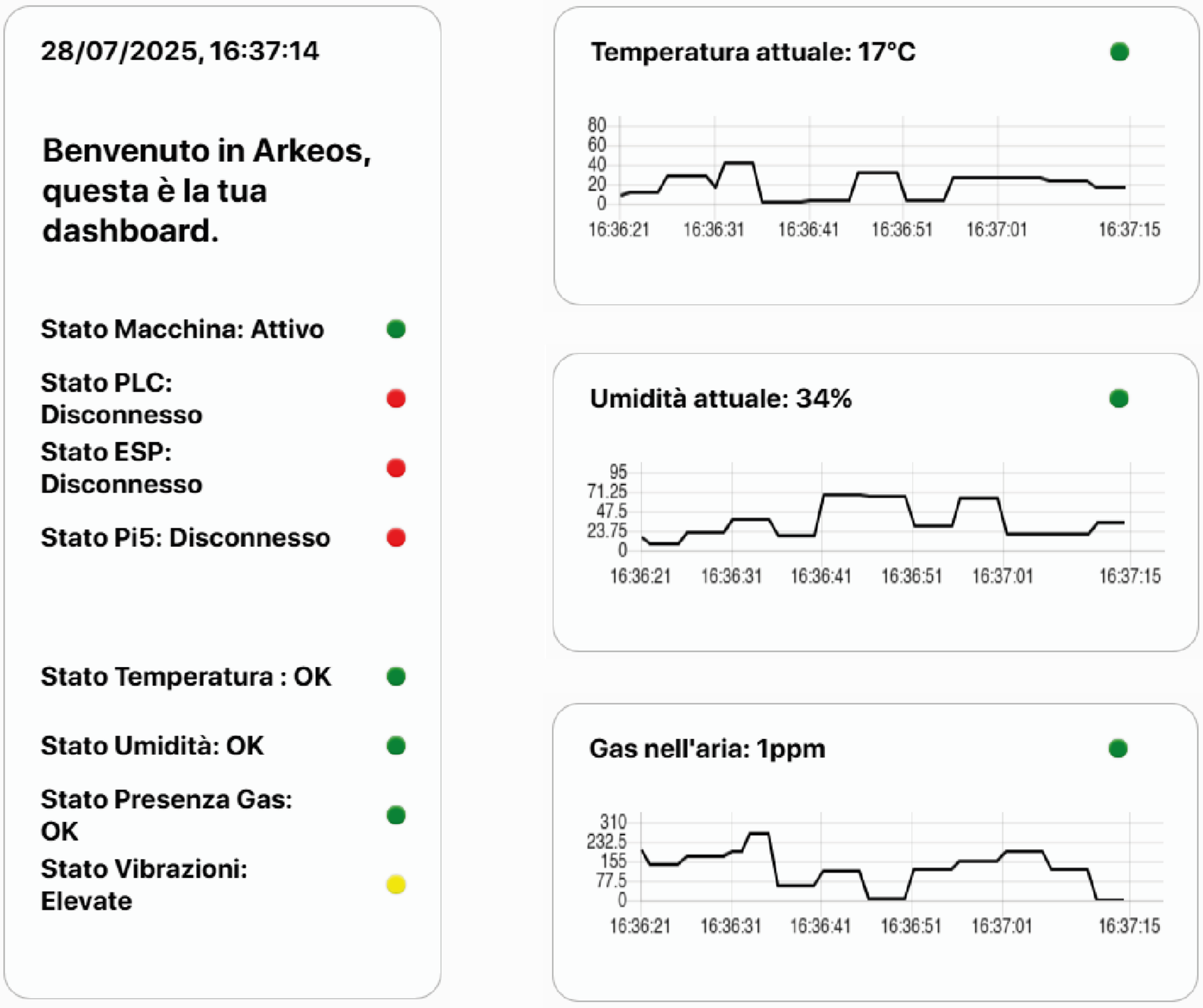

Figura 05.00: Card di status generale e valori ambientali.

05 Dashboard

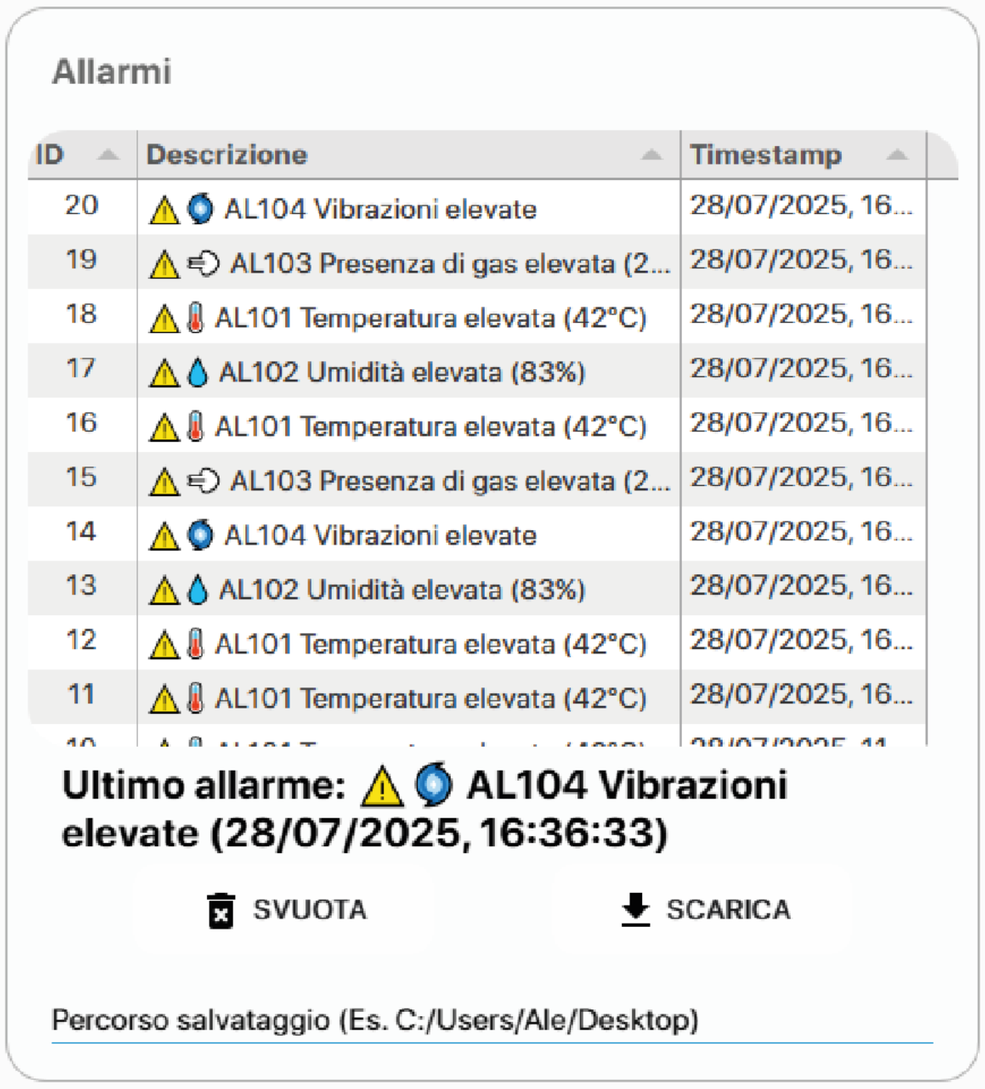

Figura 05.01: Card degli allarmi.

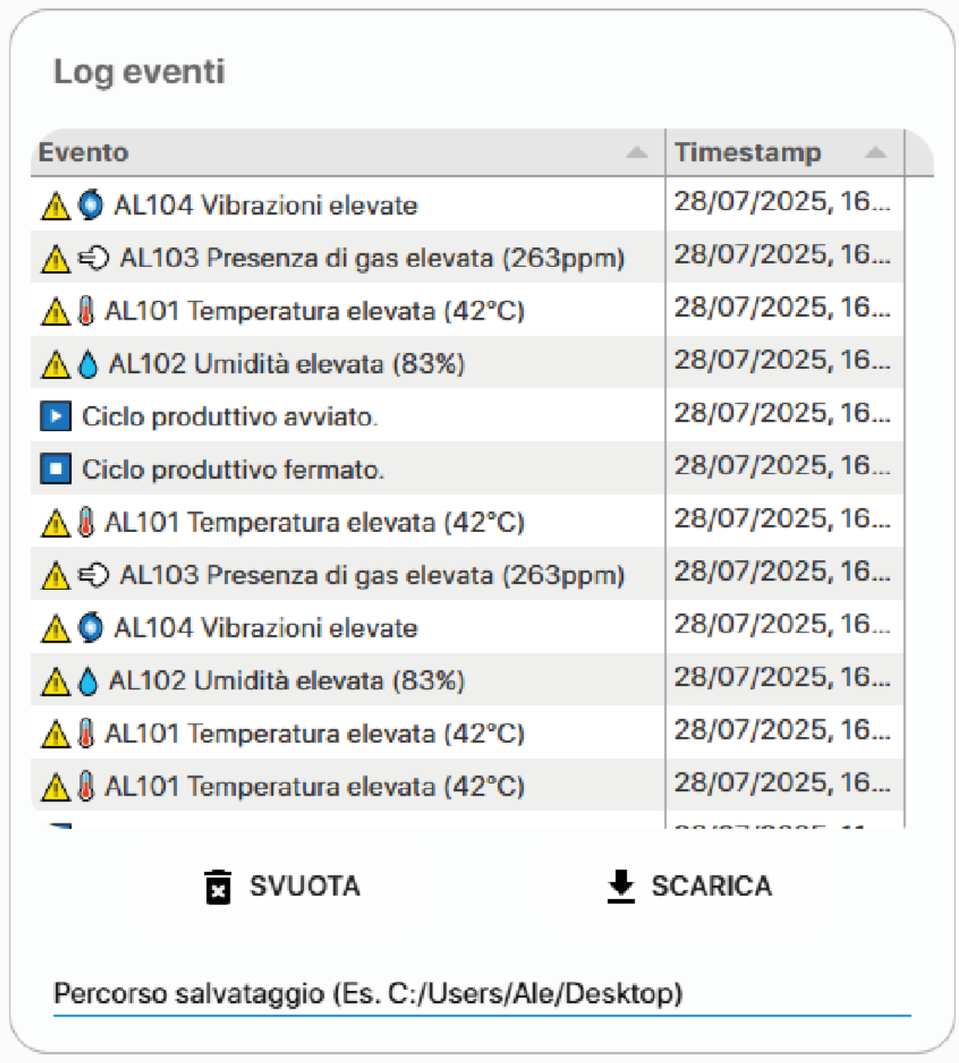

Figura 05.02: Card del log.

Figura 05.03: Card dei comandi manuali.

06 Arkeosbot

Arkeos è fornito di un **bot Telegram personale**, integrato come sistema di **notifica automatica** e come interfaccia remota e istantanea tra utente e macchina, Arkeosbot.

Ciò che lo rende unico è la possibilità di connettersi, pilotare e monitorare la macchina da **remoto**, e da qualsiasi dispositivo supporti Telegram.

I comandi disponibili sono i seguenti:

- `/stato` - Richiedi lo stato attuale della macchina.
- `/allarmi` - Richiedi lo storico degli allarmi attivi.
- `/ambiente` - Richiedi i valori ambientali della macchina.
- `/ log` - Richiedi lo storico dei log della macchina, e scarica una versione completa come file .txt.

- `/startcycle` - Avvia il ciclo produttivo.
- `/stopcycle` - Ferma il ciclo produttivo.

@ @arkeosbot
Username

ⓘ Arkeos è un sistema di automazione industriale con diagnostica remota, notifiche e controllo in tempo reale.
Bio

07 Risorse

Arkeos crede fermamente nell'**open-source**, proprio per questo tutto il codice sorgente è disponibile per essere **scaricato**.

Tutti i flow di Node-RED, la logica scritta in JavaScript, la grafica in CSS e la gestione della dashboard in HTML, possono essere scaricati dal seguente link:

https://github.com/hert1zm/Arkeos

UF17 - Reti e bus di campo

Luglio 2025, **Arkeos** - un progetto di Miori Alessandro

alessandro.miori@mat.tn.it