RED SMART
research for embedded devices

The Electronic Brand of

Integration Manual

Trace&Follow

Piattaforma di monitoraggio energetico

IDT S.r.l.s.b

Sede operativa: Corso Orbassano 402/6 - 10137 - Torino (TO)
Sede legale: Via Sebastiano Valfrè 16 - 10121 - Torino (TO)

Tel. +39 011 0922786
E-mail amministrazione: monica.giacomelli@dtsolution.com
E-mail ufficio tecnico: alberto.mazzucchelli@idtsolution.com
E-mail ufficio commerciale: margherita.ferragatta@idtsolution.com
Sito: https://idtsolution.com/

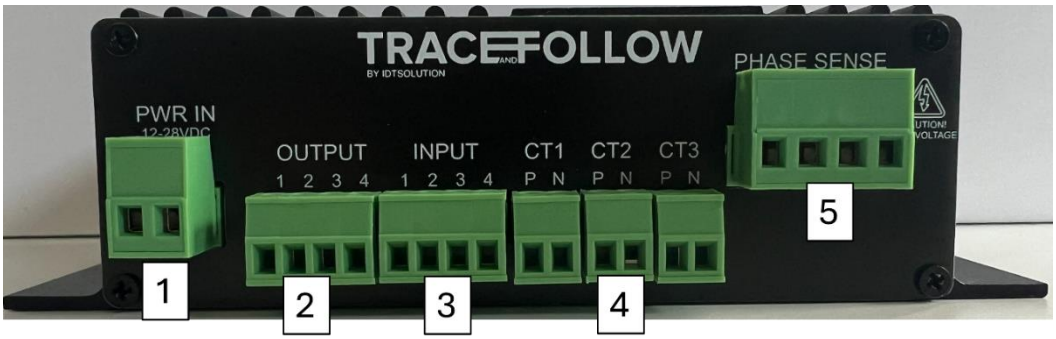
Rev. 4 – 18-07-2024

Sommario

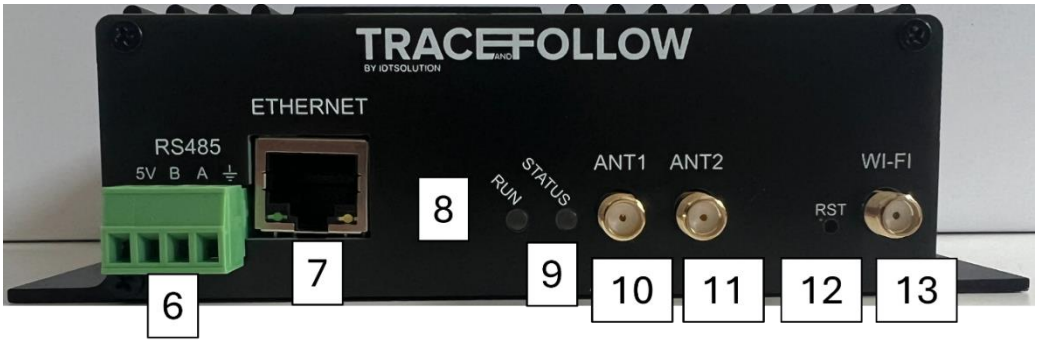
CARATTERISTICHE TECNICHE

Dimensioni (mm)	150 x 95 x 40
Peso (g)	300
Grado di protezione	IP20
Temperatura di utilizzo	0..+50°
Umidità di utilizzo	Massimo 95%, no condensa
Alimentazione	24 VDC +/- 10% 1 A max.
Protocollo di comunicazione	Modbus TCP, MQTT, RS485(espansioni interne)
Connessione	Ethernet, Wi-Fi 2.4g, 4G, LoRaWAN

PANORAMICA COMPONENTI



1. Morsetto di alimentazione
2. Morsetto output digitali
3. Morsetto input digitali
4. Morsetti di collegamento sensori corrente
5. Morsetto di collegamento fasi di rete



6. Connettore RS485
7. Connettore ethernet
8. Led run
9. Led status
10. Connettore antenna principale
11. Connettore antenna ausiliaria
12. Pulsante reset
13. Connettore antenna Wi-Fi

14. Connettore di terra

MONTAGGIO

Il dispositivo è progettato per essere montato in un contenitore DIN con un'altezza di 72 mm oppure in un contenitore flangiato delle dimensioni della scheda. Non sono previste installazioni alternative. Il modulo viene tipicamente consegnato già alloggiato in un contenitore DIN o nel suo contenitore flangiato.

ALIMENTAZIONE

Si raccomanda di alimentare il dispositivo a 24 VDC +/- 10%, il consumo massimo è di 1A che dipende dalla tipologia di espansioni collegate e dal tipo di connessione al cloud utilizzata. Il dispositivo è protetto dall'inversione di polarità.

PANORAMICA FUNZIONALITÀ

Trace&Follow è un dispositivo in grado di monitorare il consumo energetico di macchinari industriali e di inviare i dati acquisiti a una piattaforma cloud. È stata progettata e sviluppata interamente da RedSmart, BU di IDT.

La funzione principale della scheda è di misurare il consumo energetico di macchinari industriali. È in grado di acquisire segnali digitali e pilotare uscite digitali. Tramite il protocollo MQTT la scheda invia i dati acquisiti ad un broker impostabile.

Oltre a queste base la scheda implementa altre funzionalità:

- **Access-point**: la scheda è in grado di generare un access-point Wi-Fi sul quale espone una pagina web ad un indirizzo IP configurabile.
- **Server Modbus TCP**: la scheda espone un server Modbus a cui un dispositivo esterno può collegarsi per leggere/scrivere i dati presenti all'interno della scheda. In questo caso la scheda Trace&Follow è uno slave all'interno della rete.
- **Moduli di espansione**: la scheda Trace&Follow può essere interfacciata a moduli esterni tramite una porta RS485.

- **Multiple interfacce di rete**: la connessione al broker MQTT può avvenire attraverso diverse interfacce di rete: Wi-Fi 2.4g, Ethernet o 4G. È possibile configurare due interfacce MQTT.
  In caso di connessione LoRaWAN la scheda non si collega direttamente ad un broker MQTT, ma invia un pacchetto radio ad un gateway LoRa che lo decodifica. La connessione al broker MQTT in questo caso viene gestita dal gateway.
- **CutOff**: è possibile collegare un relè ad una delle uscite digitali della scheda che interrompa l'alimentazione della macchina a cui è collegata in modo da spegnerla. È possibile impostare una soglia di potenza sotto la quale la macchina viene considerata in stato di "idle" ed un tempo massimo per cui la macchina può rimanere in questo stato. Superato questo intervallo la scheda Trace&Follow commuta il relè e spegne la macchina.
  Nota: se l'alimentazione della scheda è la stessa della macchina, e non è stato previsto un UPS esterno, anche la scheda si spegnerà, non avendo un UPS integrato.

I dati acquisiti dalla scheda si dividono in due categorie:

- Dati periodici: vengono campionati ad intervalli regolari (il tempo è configurabile) e vanno a comporre un pacchetto dati che viene inserito all'interno del buffer di trasmissione. Appena possibile, i dati vengono pubblicati in messaggi in formato JSON sull'interfaccia MQTT con il relativo timestamp di creazione del pacchetto
- Dati non periodici: questi dati vengono trasmessi non appena viene scatenato un evento (es: cambio di stato di un input), a differenza di quelli periodici vengono trasmessi senza timestamp associato.

SETUP

Collegare l'alimentazione, le espansioni necessarie, le antenne e il cavo ethernet se richiesto.

File di configurazione

La configurazione del dispositivo avviene tramite un file di configurazione da caricare direttamente sulla scheda Trace&Follow.
Il file di configurazione può essere caricato e scaricato tramite l'interfaccia web della scheda Trace&Follow.
Il nome del file deve necessariamente essere "**TF_config.json"**, in formato JSON e deve contenere i seguenti campi:

- **"T&F config file":**
  - Descrizione: oggetto contenente i parametri di configurazione
  - Tipo: Oggetto JSON
  - Campi:
    1. **"CREDENTIALS":**
       - Descrizione: oggetto contenente i parametri di configurazione per l'accesso alle funzioni che richiedo credenziali nella pagina web di configurazione
       - Tipo: Oggetto JSON
       - Campi:
         (a) **"user":**
           - Descrizione: username
           - Tipo: Stringa
         (b) **"password":**
           - Descrizione: password
           - Tipo: Stringa
    2. **"AP":**
       - Descrizione: oggetto contenente i parametri di configurazione dell'access-point Wi-Fi generato dalla scheda.
       - Tipo: Oggetto JSON
       - Campi:
         (a) **"name":**
           - Descrizione: nome della rete Wi-Fi generata dalla scheda
           - Tipo: Stringa
         (b) **"password":**
           - Descrizione: password della rete Wi-Fi (da 8 a 63 caratteri)
           - Tipo: Stringa
         (c) **"ip":**
           - Descrizione: indirizzo IP della pagina web della scheda
           - Tipo: Stringa
         (d) **"mask":**
           - Descrizione: maschera di rete
           - Tipo: Stringa
    3. **"WIFI":**
       - Descrizione: oggetto contenente i parametri di configurazione della connessione Wi-Fi
       - Tipo: Oggetto JSON
       - Campi:
         (a) **"use_dhcp":**
           - Descrizione: DHCP abilitato
           - Tipo: Bool
         (b) **"ip":**
           - Descrizione: indirizzo IP fisso della scheda (utilizzato se use_dhcp = 0)
           - Tipo: Stringa (es:"192.168.1.100")

(c) **"mask":**
- Descrizione: maschera di rete (utilizzato se use_dhcp = 0)
- Tipo: Stringa

(d) **"gateway":**
- Descrizione: gateway di rete (utilizzato se use_dhcp = 0)
- Tipo: Stringa

(e) **"dns":**
- Descrizione: dns della rete (utilizzato se use_dhcp = 0)
- Tipo: Stringa

(f) **"ssid":**
- Descrizione: nome della rete a cui collegarsi
- Tipo: Stringa

(g) **"password":**
- Descrizione: password della rete a cui collegarsi
- Tipo: Stringa

(h) **"channel":**
- Descrizione: canale della rete Wi-Fi
- Tipo: Intero [0..14, 0=auto]

(i) **"bssid":**
- Descrizione: bssid della rete Wi-Fi
- Tipo: Stringa

4. **"RTC":**
- Descrizione: oggetto contenente i parametri di configurazione del RTC
- Tipo: Oggetto JSON
- Campi:

(a) **"time_from_network":**
- Descrizione: configurazione inizializzazione timestamp
- Tipo: Bool

(b) **"timezone":**
- Descrizione: timezone da utilizzare per la scheda
- Tipo: Intero [-12..+12]

(c) **"timestamp_rel":**
- Descrizione: forza la pubblicazione dei messaggi con timestamp relativo all'accensione della scheda invece che con timestamp assoluto
- Tipo: Bool

5. **"ETH":**
- Descrizione: oggetto contenente i parametri di configurazione della connessione ethernet
- Tipo: Oggetto JSON
- Campi:

(a) **"use_dhcp":**
- Descrizione: DHCP abilitato
- Tipo: Bool

(b) **"ip":**
- Descrizione: indirizzo IP fisso della scheda (utilizzato se use_dhcp = 0)
- Tipo: Stringa (es:"192.168.1.100")

(c) **"mask":**
- Descrizione: maschera di rete (utilizzato se use_dhcp = 0)
- Tipo: Stringa

(d) **"gateway":**
- Descrizione: gateway di rete (utilizzato se use_dhcp = 0)
- Tipo: Stringa

(e) **"dns":**
- Descrizione: dns della rete (utilizzato se use_dhcp = 0)
- Tipo: Stringa

6. **“MAIN_MQTT”:**
   - Descrizione: oggetto contenente i parametri di configurazione della connessione al broker MQTT (utilizzato in caso di connessione Wi-Fi, Ethernet e 4G)
   - Tipo: Oggetto JSON
   - Campi:
     (a) **“server”:**
       - Descrizione: server MQTT a cui collegarsi
       - Tipo: Stringa
     (b) **“port”:**
       - Descrizione: porta del server a cui collegarsi
       - Tipo: Intero [1..65535]
     (c) **“attribute”:**
       - Descrizione: topic MQTT su cui pubblicare gli attributi
       - Tipo: Stringa (es: ”topic/attribute”)
     (d) **“telemetry”:**
       - Descrizione: topic MQTT su cui pubblicare i dati di telemetria
       - Tipo: Stringa
     (e) **“rpc”:**
       - Descrizione: topic MQTT su cui pubblicare da cui la scheda riceve i comandi rpc
       - Tipo: Stringa
     (f) **“username”:**
       - Descrizione: username autenticazione server MQTT
       - Tipo: Stringa
     (g) **“password”:**
       - Descrizione: password autenticazione server MQTT
       - Tipo: Stringa
     (h) **“connection_int”:**
       - Descrizione: intervallo di tempo (msec) tra tentativi di riconnessione in caso di errore di connessione
       - Tipo: Intero [0..65535]

7. **“SECONDARY_MQTT”:**
   - Descrizione: oggetto contenente i parametri di configurazione della connessione al broker MQTT secondario (utilizzato in caso di connessione Wi-Fi, Ethernet e 4G)
   - Tipo: Oggetto JSON
   - Campi:
     (a) **“active”:**
       - Descrizione: abilitazione MQTT secondario
       - Tipo: Bool
     (b) **“server”:**
       - Descrizione: server MQTT a cui collegarsi
       - Tipo: Stringa
     (c) **“port”:**
       - Descrizione: porta del server a cui collegarsi
       - Tipo: Intero [1..65535]
     (d) **“attribute”:**
       - Descrizione: topic MQTT su cui pubblicare gli attributi
       - Tipo: Stringa (es: ”topic/attribute”)
     (e) **“telemetry”:**
       - Descrizione: topic MQTT su cui pubblicare i dati di telemetria
       - Tipo: Stringa
     (f) **“rpc”:**
       - Descrizione: topic MQTT su cui pubblicare da cui la scheda riceve i comandi rpc
       - Tipo: Stringa

(g) **"username":**
- Descrizione: username autenticazione server MQTT
- Tipo: Stringa

(h) **"password":**
- Descrizione: password autenticazione server MQTT
- Tipo: Stringa

(i) **"connection_int":**
- Descrizione: intervallo di tempo (msec) tra tentativi di riconnessione in caso di errore di connessione
- Tipo: Intero [0..65535]

8. **"LORA":**
   - Descrizione: oggetto contenente i parametri di configurazione della connessione LoRa.
   - Tipo: Oggetto JSON
   - Campi:
     (a) **"app_eui":**
     - Descrizione: app eui LoRa
     - Tipo: Stringa

     (b) **"app_key":**
     - Descrizione: app key LoRa
     - Tipo: Stringa

     (c) **"downlink_time":**
     - Descrizione: tempo di attesa in ricezione dopo un invio
     - Tipo: Intero [0..10000]

     (d) **"max_send_size":**
     - Descrizione: max numero di byte per messaggio. Da settare fisso a 65
     - Tipo: Intero [65]

     (e) **"power":**
     - Descrizione: indice potenza di trasmissione.
     - Tipo: Intero [0..5]

     (f) **"min_transmission_time":**
     - Descrizione: tempo minimo tra due trasmissioni consecutive in millisecondi. Min:15000
     - Tipo: Intero [15000...60000]

9. **"HTTP" (non utilizzato):**
   - Descrizione: oggetto contenente i parametri di configurazione della connessione HTTP
   - Tipo: Oggetto JSON
   - Campi:
     (a) Non sono presenti campi, riservato per future implementazioni

10. **"SD" (non utilizzato):**
    - Descrizione: oggetto contenente i parametri di configurazione dell'utilizzo della scheda SD
    - Tipo: Oggetto JSON
    - Campi:
      (a) Non sono presenti campi, riservato per future implementazioni

11. **"CONNECTION":**
    - Descrizione: oggetto contenente i parametri di configurazione del tipo di connessione
    - Tipo: Oggetto JSON
    - Campi:
      (a) **"type":**
      - Descrizione: tipo di connessione (1-> ethernet, 2-> WiFi, 3-> LoRa, 4-> 4G)
      - Tipo: Intero

      (b) **"machine" (non utilizzato):**
      - Descrizione: riservato (da lasciare fisso a 1)
      - Tipo: Intero

12. **“PACKET”:**
    - Descrizione: oggetto contenente i parametri di configurazione della creazione dei pacchetti
    - Tipo: Oggetto JSON
    - Campi:
        (a) **“packet_int”:**
            - Descrizione: tempo di pacchettizzazione in millisecondi
            - Tipo: Intero [1000..4294967295]
        (b) **“buffer_size”:**
            - Descrizione: massimo numero di pacchetti salvabili nel buffer di trasmissione
            - Tipo: Intero

13. **“POWER”:**
    - Descrizione: oggetto contenente i parametri di configurazione della misurazione dell’energia
    - Tipo: Oggetto JSON
    - Campi:
        (a) **“sampling_int”:**
            - Descrizione: tempo tra un campionamento e un altro dell'energy meter in millisecondi
            - Tipo: Intero
        (b) **“voltage_sensor”:**
            - Descrizione: uso del voltage sampling. 0-> no, 1-> si
            - Tipo: Bool
        (c) **“line_freq”:**
            - Descrizione: frequenza di rete (valore utilizzato se voltage_sensor = 0, altrimenti rilevato dalla scheda)
            - Tipo: Intero
        (d) **“pga_gain”:**
            - Descrizione: Valore da settare in base al CT utilizzato. Vedere la relativa tabella[1].
            - Tipo: Intero
        (e) **“voltage_gain”:**
            - Descrizione: guadagno in ingresso del PGA sulle tensioni
            - Tipo: Intero
        (f) **“gain_ct1”:**
            - Descrizione: calibrazione CT1. Valore da settare in base al CT utilizzato. Vedere la relativa tabella[1].
            - Tipo: Intero
        (g) **“gain_ct2”:**
            - Descrizione: calibrazione CT2. Valore da settare in base al CT utilizzato. Vedere la relativa tabella[1].
            - Tipo: Intero
        (h) **“gain_ct3”:**
            - Descrizione: calibrazione CT3. Valore da settare in base al CT utilizzato. Vedere la relativa tabella[1].
            - Tipo: Intero
        (i) **“pf”:**
            - Descrizione: power factor (valore utilizzato se voltage_sensor = 0, altrimenti rilevato dalla scheda)
            - Tipo: Float
        (j) **“voltage”:**
            - Descrizione: tensione di rete (valore utilizzato se voltage_sensor = 0, altrimenti rilevato dalla scheda)
            - Tipo: Intero
        (k) **“connection_type”:**
            - Descrizione: tipologia di connessone: 0-> Trifase con neutro, 1-> Trifase senza neutro
            - Tipo: Intero

14. **“INPUTS”:**
    - Descrizione: oggetto contenente i parametri di configurazione di input fisici e virtuali
    - Tipo: Oggetto JSON
    - Campi:
        - **(a) “number_phy”:**
            - Descrizione: numero input fisici. Da settare a 4
            - Tipo: Intero
        - **(b) “inputs”:**
            - Descrizione: array di descrizione degli input fisici. Deve contenere number_phy elementi
            - Tipo: Array JSON
            - Campi:
                - **a. “num”:**
                    - Descrizione: numero dell’input, deve contenere un numero progressivo per ogni elemento dell’array a partire dal numero 1
                    - Tipo: Intero
                - **b. “enabled”:**
                    - Descrizione: abilitazione dell’input
                    - Tipo: Bool
                - **c. “mod”:**
                    - Descrizione: modalità input. (1-> normal, 2-> counter, 3-> function, 4-> blinking)
                    - Tipo: Intero
                - **d. “polarity”:**
                    - Descrizione: polarità input (0-> normal, 1-> inverse)
                    - Tipo: Bool
                - **e. “filter_time”:**
                    - Descrizione: filtro input, nessun impulso sotto questa soglia viene considerato in modalità 1
                    - Tipo: Intero [millisecondi]
                - **f. “reset_counter”:**
                    - Descrizione: abilitazione reset del contatore associato all’input dopo l’invio quando questo è configurato in modalità “counter”(2)
                    - Tipo: Bool
                - **g. “add_to_packet”:**
                    - Descrizione: abilitazione inserimento del valore del input quando configurato in modalità 1, 3 o 4 all’interno del pacchetto di dati periodici inviati. Se abilitato, i messaggi asincroni relativi ai cambiamenti di stato di questo input non saranno più inviati
                    - Tipo: Bool

        - **(c) “number_virt”:**
            - Descrizione: numero input virtuali. Da settare in base a quanti input virtuali si vogliono settare.
            - Tipo: Intero
        - **(d) “virtual_inputs”:**
            - Descrizione: array di descrizione degli input virtuali. Deve contenere **number_virt** elementi
            - Tipo: Array JSON
            - Campi:
                - **a. “num”:**
                    - Descrizione: numero dell’input, deve contenere un numero progressivo per ogni elemento dell’array a partire dal numero 1
                    - Tipo: Intero
                - **b. “enabled”:**
                    - Descrizione: abilitazione dell’input
                    - Tipo: Bool

c. **“mod”:**
   - Descrizione: modalità input. (1-> normal, 2-> counter, 3-> function, 4-> blinking)
   - Tipo: Intero

d. **“polarity”:**
   - Descrizione: polarità input (0-> normal, 1-> inverse)
   - Tipo: Bool

e. **“filter_time”:**
   - Descrizione: filtro input, nessun impulso sotto questa soglia viene considerato in modalità 1
   - Tipo: Intero [millisecodi]

f. **“reset_counter”:**
   - Descrizione: abilitazione reset del contatore associato all’input dopo l’invio quando questo è configurato in modalità “counter”(2)
   - Tipo: Bool

g. **“add_to_packet”:**
   - Descrizione: abilitazione inserimento del valore del input quando configurato in modalità 1, 3 o 4 all’interno del pacchetto di dati periodici inviati. Se abilitato, i messaggi asincroni relativi ai cambiamenti di stato non saranno più inviati
   - Tipo: Bool

15. **“OUTPUTS”:**
- Descrizione: oggetto contenente i parametri di configurazione di output fisici e virtuali
- Tipo: Oggetto JSON
- Campi:

(a) “number_phy”:
- Descrizione: numero input fisici. Da settare a 4
- Tipo: Intero

(b) “outputs”:
- Descrizione: array di descrizione degli output fisici. Deve contenere number_phy elementi
- Tipo: Array JSON
- Campi:

a. **“num”:**
   - Descrizione: numero dell’input, deve contenere un numero progressivo per ogni elemento dell’array a partire dal numero 1
   - Tipo: Intero

b. **“polarity”:**
   - Descrizione: polarità input (0-> normal, 1-> inverse)
   - Tipo: Bool

c. **“pin”:**
   - Descrizione: pin input. (fisso: input1=47, input2 = 46, input1=45, input2 = 44)
   - Tipo: Intero

(c) “number_virt”:
- Descrizione: numero output virtuali. Da settare in base a quanti output virtuali si vogliono settare.
- Tipo: Intero

(d) “virtual_inputs”:
- Descrizione: array di descrizione degli output virtuali. Deve contenere number_virt elementi
- Tipo: Array JSON
- Campi:

a. **“num”:**
   - Descrizione: numero dell’output, deve contenere un numero progressivo per ogni elemento dell’array a partire dal numero 1
   - Tipo: Intero

b. **“polarity”:**
   - Descrizione: polarità input (0-> normal, 1-> inverse)
   - Tipo: Bool

c. **“pin”:**
   - Descrizione: riservato, lasciare fisso a 0
   - Tipo: Intero

16. **“CUTOFF”:**
    - Descrizione: oggetto contenente i parametri di configurazione della funzionalità di CutOff
    - Tipo: Oggetto JSON
    - Campi:
      (a) **“is_active”:**
        - Descrizione: abilitazione CutOff
        - Tipo: Bool

      (b) **“pin”:**
        - Descrizione: pin relay cutoff (fisso: input1=47, input2 = 46, input1=45, input2 = 44)
        - Tipo: Intero

      (c) **“threshold”:**
        - Descrizione: threshold potenza
        - Tipo: Intero [W]

      (d) **“idle_time”:**
        - Descrizione: tempo dopo cui scatta il relay se in idle
        - Tipo: Intero [millisecondi]

17. **“LOCALIZATION”:**
    - Descrizione: oggetto contenente i parametri di configurazione della funzionalità di Localizzazione
    - Tipo: Oggetto JSON
    - Campi:
      (a) **“is_active”:**
        - Descrizione: abilitazione localizzazione
        - Tipo: Bool

18. **“MODBUS_TCP”:**
    - Descrizione: oggetto contenente i parametri di configurazione della funzionalità di Localizzazione
    - Tipo: Oggetto JSON
    - Campi:
      (a) **“is_active”:**
        - Descrizione: abilitazione modbus TCP server
        - Tipo: Bool

      (b) **“input_registers_number”:**
        - Descrizione: numero input registers.
        - Tipo: Intero
        - Valori ammessi: il valore deve essere lasciato fisso a 50

      (c) **“holding_registers_number”:**
        - Descrizione: numero holding registers
        - Tipo: Intero
        - Valori ammessi: il valore deve essere lasciato fisso a 60

      (d) **“coils_number”:**
        - Descrizione: numero coils
        - Tipo: Bool
        - Valori ammessi: il valore deve essere lasciato fisso a 0

      (e) **“is_active”:**
        - Descrizione: abilitazione localizzazione
        - Tipo: Bool

(f) **"port":**
- Descrizione: porta su cui è attivo il server
- Tipo: Intero

(g) **"slave_id":**
- Descrizione: id modbus
- Tipo: Intero

(h) **"send_input":**
- Descrizione: se = 1, anche gli input register vengono inviati alla piattaforma cloud
- Tipo: Bool

(i) **"max_client_number":**
- Descrizione: massimo numero di client che possono essere connessi contemporaneamente al server
- Tipo: Intero

(j) **"idle_timeout":**
- Descrizione: tempo dopo il quale, in caso non vengano fatte richieste dal client al server, il client viene disconnesso
- Tipo: Intero [millisecondi]

Qui di seguito è riportato il file di configurazione di default:

```
{
  "T&F config file": {
    "CREDENTIALS": {
      "user": "user",
      "password": "password"
    },
    "AP": {
      "name": "TraceAndFollow",
      "password": "password-TF",
      "ip": "192.168.1.1",
      "mask": "255.255.255.0",
      "gateway": "192.168.1.1"
    },
    "WIFI": {
      "use_dhcp": false,
      "ip": "192.168.1.100",
      "mask": "255.255.255.0",
      "gateway": "192.168.1.1",
      "dns": "192.168.1.1",
      "ssid": "ssid",
      "password": "password",
      "channel": 0,
      "bssid": "0"
    },
    "RTC": {
      "time_from_network": false,
      "timezone": 0,
      "timestamp_rel": true
    },
    "ETH": {
      "use_dhcp": false,
      "ip": "192.168.1.100",
      "mask": "255.255.255.0",
      "gateway": "192.168.1.1",
      "dns": "192.168.1.1"
    },
    "MAIN_MQTT": {
```

```
  "server": "broker.hivemq.com",
  "port": 1883,
  "attribute": "attributes",
  "telemetry": "telemetry",
  "rpc": "rpc",
  "username": "user",
  "password": "password",
  "connection_int": 2000
},
"SECONDARY_MQTT": {
      "server": "broker.hivemq.com",
      "port": 1883,
      "attribute": "attributes",
      "telemetry": "telemetry",
      "rpc": "rpc",
      "username": "user2",
      "password": "password2",
      "connection_int": 2000
},
"LORA": {
  "app_eui": "00000000000000000000000000000000",
  "app_key": "00000000000000000000000000000000",
  "downlink_time": 2000,
  "max_send_size": 65,
  "power": 5,
  "min_transmission_time": 30000
},
"HTTP": {},
"SD": {},
"CONNECTION": {
  "type": 1,
  "machine":3
},
"PACKET": {
  "packet_int": 30000,
  "buffer_size": 100
},
"POWER": {
  "sampling_int": 100,
  "voltage_sensor": 0,
  "line_freq": 50,
  "pga_gain": 42,
  "voltage_gain": 1,
  "gain_ct1": 38904,
  "gain_ct2": 38904,
  "gain_ct3": 38904,
  "pf": 0.95,
  "voltage": 230,
  "connection_type": 0
},
"INPUTS": {
  "number_phy": 4,
  "inputs": [
    {
      "num": 1,
      "enabled": false,
      "mod": 1,
      "polarity": 0,
```

```
      "filter_time": 1000,
      "reset_counter": false,
      "add_to_packet": false
    },
    {
      "num": 2,
      "enabled": false,
      "mod": 1,
      "polarity": 0,
      "filter_time": 5000,
      "reset_counter": false,
      "add_to_packet": false
    },
    {
      "num": 3,
      "enabled": false,
      "mod": 1,
      "polarity": 0,
      "filter_time": 1000,
      "reset_counter": false,
      "add_to_packet": false
    },
    {
      "num": 4,
      "enabled": false,
      "mod": 1,
      "polarity": 0,
      "filter_time": 1000,
      "reset_counter": false,
      "add_to_packet": false
    }
  ],
  "number_virt": 1,
  "virtual inputs": [
    {
      "num": 1,
      "enabled": false,
      "mod": 1,
      "polarity": 0,
      "filter_time": 1000,
      "reset_counter": false,
      "add_to_packet": false
    }
  ]
},
"OUTPUTS": {
  "number_phy": 4,
  "outputs": [
    {
      "num": 1,
      "polarity": 0,
      "pin": 47
    },
    {
      "num": 2,
      "polarity": 0,
      "pin": 46
    },
```

```
        {
          "num": 3,
          "polarity": 0,
          "pin": 45
        },
        {
          "num": 4,
          "polarity": 0,
          "pin": 44
        }
      ],
      "number_virt": 1,
      "virtual outputs": [
        {
          "num": 1,
          "polarity": 0,
          "pin": 0
        }
      ]
    },
    "CUTOFF": {
      "is_active": false,
      "pin": 44,
      "threshold": 200,
      "idle_time": 100000
    },
    "LOCALIZATION": {
      "is_active": false
    },
    "MODBUS_TCP": {
      "is_active": false,
      "input_registers_number": 50,
      "holding_registers_number": 60,
      "coils_number": 0,
      "port": 502,
      "slave_id": 1,
      "send_input": false,
      "max_client_number": 2,
      "idle_timeout": 600000
    }
  }
}
```


1. **Valori di calibrazione Current Transformers e pga.**

Modello	Rapporto	Tensione/Corrente	gain_ctx	pga_gain	Voltage_gain
SCT024TS	200A/100mA	C	31595	21	46604
SCT013	100A/1V	T	44409	21	46604

FASI OPERATIVE

Verifiche Preliminari

All'accensione della scheda, verificare che i led presenti sul box si accendano. In caso contrario verificare l'alimentazione della scheda.

Inizializzazione

All'avvio, la scheda Trace&Follow legge il file di configurazione salvato, configura tutte le periferiche in accordo con esso, avvia l'interfaccia Wi-Fi che genera la sua rete e attiva l'interfaccia di rete selezionata attraverso cui collegarsi alla piattaforma cloud.
Una volta ultimati questi passaggi, il led di "run" presente a bordo scheda comincerà a lampeggiare ad una frequenza di 1Hz.

In concomitanza, il led di status della scheda si accende di bianco, ad indicare che è in corso un tentativo di connessione al cloud tramite l'interfaccia selezionata.

Verificare che i seguenti led siano accesi durante la fase di inizializzazione:

Led **RUN**: lampeggio a frequenza di 1HZ Led **STATUS**: fisso colore BIANCO

Ethernet

Possibili errori:

- Led STATUS di colore rosso: cavo ethernet non correttamente inserito.

Wi-Fi

Possibili errori

- Led STATUS di colore rosso: impossibile connettersi alla rete selezionata. Dopo un breve periodo il led ritorna di colore bianco ad indicare che è in corso un nuovo tentativo di connessione

4G

Possibili errori

- Led STATUS di colore rosso: impossibile connettersi alla rete cellulare. Dopo un breve periodo il led ritorna di colore bianco ad indicare che è in corso un nuovo tentativo di connessione.
  In questo caso controllare:
  - Che l'antenna sia correttamente collegata
  - Che la SIM sia inserita dell'apposito alloggiamento
  - Che la SIM abbia un piano dati attivo

Errori comuni alle tre interfacce:

- Led RUN fisso o acceso o spento: errore generico scheda. Controllare l'alimentazione ed eseguire un reset della scheda. Se il problema persiste contattate il supporto tecnico.

- Led STATUS spento: errore generico scheda. Controllare l'alimentazione ed eseguire un reset della scheda. Se il problema persiste contattate il supporto tecnico.

Una volta che la connessione è stata stabilita il led diventa di colore BLU ad indicare che la scheda è connessa ad internet.
A questo punto la scheda proverà ad ottenere il timestamp corrente dalla rete se necessario (vedi capitolo "Payload dati periodici"). In caso di errore il led STATUS lampeggia di BLU e ROSSO alternati. Dopo 3 tentativi falliti, la scheda riavvia l'interfaccia di rete in automatico finché il timestamp non viene ottenuto.

La scheda infine effettua un tentativo di connessione al broker MQTT configurato (e al secondo se attivo).
In caso di errore il led STATUS lampeggia di rosso. Da file di configurazione è possibile impostare l'intervallo di riconnessione. Avvenuta la connessione al broker MQTT il led STATUS si accende di VERDE e la scheda comincia ad inviare dati. In caso la connessione ad almeno un broker avvenga correttamente, il led si accenderà comunque di verde.

Nota: l'ottenimento del timestamp e la connessione al broker possono avvenire in modo molto rapido, è quindi possibile vedere il led STATUS passare direttamente da bianco a verde.

INTERFACCIA MQTT

La scheda è in grado di inviare e ricevere dati a/da una piattaforma utilizzando una connessione MQTT. I payload scambiati sono tutti in formato JSON.

I dati acquisiti dalla scheda sono divisi in due categorie separate:
- Dati non periodici di configurazione, pubblicati nel topic **"attribute"**
- Dati periodici e non periodici di stato, pubblicati nel topic **"telemetry"**

I dati inviati alla scheda devono essere pubblicati nel topic **"rpc"**, seguendo un formato spiegato in seguito.

Tutti e tre questi topic sono configurabili attraverso il file di configurazione.
La scheda supporta la connessione a due broker MQTT in simultanea, entrambi sono configurabili dal file di configurazione in modo separato. Il primo (MAIN_MQTT) è attivo di default e non è disattivabile, il secondo (SECONDARY_MQTT) va abilitato tramite file di configurazione. Le due connessioni sono parallele ed indipendenti. Gli aggiornamenti OTA sono possibili solo tramite il broker primario.

I dati periodici vengono acquisiti e salvati ad intervalli regolari in strutture dati chiamate pacchetti. La scheda appena ha un pacchetto pronto tenta di inviarlo al broker MQTT. In caso di errore nell'invio questo pacchetto viene salvato all'interno di un buffer dalla scheda. Non appena la scheda riacquista la comunicazione con il broker, invia tutti i pacchetti presenti nel buffer. Se la dimensione del payload risultate è troppo grande, l'invio viene suddiviso in più messaggi, fino a che il buffer non si svuota.
I dati non periodici vengono inviati al broker non appena viene rilevato un loro cambio si stato.

Dati inviati dalla scheda

Dati non periodici:

- **input(N)_type**: campo che indica la modalità in cui un input fisico è configurato.
  - Tipo: Intero
  - Valori ammessi: [1-4]

- **virtual_input(N)_type**: campo che indica la modalità in cui un input virtuale è configurato.
  - Tipo: Intero
  - Valori ammessi: [1-4]

- **packet_time**: campo che indica ogni quanto avviene la creazione di un pacchetto contenente dati periodici in secondi.
  - Tipo: Intero
  - Valori ammessi: [1 …] secondi
- **energy_standby_threshold**: campo che indica la soglia di potenza istantanea sotto la quale la scheda considera la macchina in stato di idle.
  - Tipo: Intero
  - Valori ammessi: [0 …] [W]

- **inactivity_time**: campo che indica dopo quanto tempo scatta il relay di CutOff quando la macchina si trova in stato di idle in secondi. Questa funzione deve essere abilitata dal file di configurazione.
  - Tipo: Intero
  - Valori ammessi: [15 …] [s], se = 0, disabilitato

- **transmission_time**: valore fisso a 0.
  - Tipo: Intero
  - Valori ammessi: 0

- **hardware_version**: campo che indica la versione hardware della scheda
  - Tipo: Stringa

- **firmware_version**: campo che indica la versione firmware della scheda
  - Tipo: Stringa

- **connection_type**: campo che indica l'interfaccia di rete tramite cui la scheda è connessa al broker.
  - Tipo: Intero
  - Valori ammessi: [1-4]

- **timezone:** campo che indica la timezone in cui la scheda si trova. La scheda non la prende automaticamente ma viene settata da file di configurazione.
  - Tipo: Intero
  - Valori ammessi [-12 - +12]

- **timestamp_rel:** campo che indica se la scheda sta usando un timestamp relativo o assoluto per la pubblicazione dei messaggi.
  - Tipo: Bool

- **ip**: campo che indica l'IP della scheda nella rete.
  - Tipo: Stringa

- **gateway**: campo che indica il gateway configurato nella rete.
  - Tipo: Stringa

- **mask**: campo che indica la maschera di rete configurata.
  - Tipo: Stringa

- **dns**: campo che indica il dns configurato.
  - Tipo: Stringa

- **mac**: campo che indica il mac address della scheda di rete.
  - Tipo: Stringa

Dati non periodici di stato:

- **input(N)_value**: campo che indica lo stato corrente dell'input o della funzione ad esso associata quando input(N)_type = 1 (normal), 3 (function), 4 (blinking)
  - stato dell'input fisico.
  - Tipo: Bool
  - Valori ammessi: [0-1]

- **virtual_input(N)_value**: campo che indica lo stato corrente dell'input o della funzione ad esso associata quando virtual_input(N)_type = 1 (normal), 3 (function), 4 (blinking)
  - stato dell'input virtuale.
  - Tipo: Bool
  - Valori ammessi: [0-1]

- **output(N):** stato dell'output fisico
  - Tipo: Bool
  - Valori ammessi: [0-1]

- **virtual_output(N):** stato dell'output virtuale
  - Tipo: Bool
  - Valori ammessi: [0-1]

Dati periodici:

- **energy_consuption**: energia attiva consumata nell'arco di tempo di un pacchetto in Wh.
  - Tipo: Float

- **input(N)_value**: campo che indica il numero di impulsi ricevuti dal relativo input quando input(N)_type = 2 (counter)
  - numero di impulsi nell'arco di tempo di un pacchetto.
  - Tipo: Intero
  - Valori ammessi: [0 …]

- **virtual_input(N)_value**: campo che indica il numero di impulsi ricevuti dal relativo input quando virtual_input(N)_type = 2 (counter)
  - numero di impulsi nell'arco di tempo di un pacchetto.
  - Tipo: Intero
  - Valori ammessi: [0 …]

- **voltage1**: Tensione sulla fase 1.
  - Tipo: Float [V]
- **voltage2**: Tensione sulla fase 2.
  - Tipo: Float [V]
- **voltage3**: Tensione sulla fase 3.
  - Tipo: Float [V]
- **current1**: Corrente sulla fase 1.
  - Tipo: Float [A]
- **current2**: Corrente sulla fase 2.
  - Tipo: Float [A]
- **current3**: Corrente sulla fase 3.
  - Tipo: Float [A]
- **current_tot**: Corrente totale sulle 3 fasi.
  - Tipo: Float [A]

- **active_power1**: potenza attiva istantanea sulla fase 1.
  - Tipo: Float [W]
- **active_power2**: potenza attiva istantanea sulla fase 2.
  - Tipo: Float [W]
- **active_power3**: potenza attiva istantanea sulla fase 3.
  - Tipo: Float [W]
- **active_power_tot**: potenza attiva istantanea sulle 3 fasi.
  - Tipo: Float [W]
- **reactive_power1**: potenza reattiva istantanea sulla fase 1.
  - Tipo: Float [VAR]
- **reactive_power2**: potenza reattiva istantanea sulla fase 2.
  - Tipo: Float [VAR]
- **reactive_power3**: potenza reattiva istantanea sulla fase 3.
  - Tipo: Float [VAR]
- **reactive_power_tot**: potenza reattiva istantanea sulle 3 fasi.
  - Tipo: Float [VAR]
- **pf1**: power factor istantaneo sulla fase 1.
  - Tipo: Float
- **pf2**: power factor istantaneo sulla fase 2.
  - Tipo: Float
- **pf3**: power factor istantaneo sulla fase 3.
  - Tipo: Float
- **pf_tot**: power factor istantaneo sulle 3 fasi.
  - Tipo: Float
- **phase1**: sfasamento fase 1.
  - Tipo: Float [°]
- **phase2**: sfasamento sulla fase 2.
  - Tipo: Float [°]
- **phase3**: sfasamento sulla fase 3.
  - Tipo: Float [°]
- **frequency**: frequenza di rete.
  - Tipo: Float [Hz]
- **reactive_energy_consuption**: energia reattiva consumata nell'arco di tempo di un pacchetto in VARh.
  - Tipo: Float [VARh]

Nota: quando un input è configurato in modalità 1, 3 o 4 la logica del loro invio segue quella dei dati non periodici, ovvero vengono inviati non appena viene rilevato un cambio di stato (se la chiave "add_to_packet" = "false").

Nota: I dati energetici qualitativi vengono inviti solamente se le fasi di alimentazioni sono collegate alla scheda e nel file di configurazione la chiave "voltage_sensor" ha valore "true".

Payload alla prima connessione

All'avvio della scheda, non appena la connessione al broker viene stabilita, la scheda pubblica un messaggio contenente i seguenti campi sul topic **"attribute"**

- packet_time
- transmission_time
- energy_standby_threshold
- inactivity_time
- hardware_version
- firmware_version
- connection_type
- timezone
- timestamp_rel

- input1_type
- input2_type
- input3_type
- input4_type
- virtual_input(N)_type (una chiave per ogni input virtuale configurato)

Esempio di payload iniziale, con topic **attribute** → v1/devices/me/attributes :

Topic: v1/devices/me/attributes QoS: 0


```
{
  "packet_time": 30,
  "transmission_time": 0,
  "energy_standby_threshold": 5000,
  "inactivity_time": 100,
  "hardware_version": "2.1",
  "firmware_version": "1.0.3",
  "connection_type": 1,
  "timezone": 1,
  "timestamp_rel": false,
  "input1_type": 1,
  "input2_type": 0,
  "input3_type": 0,
  "input4_type": 0
}
```


Payload dati di rete

Ogni volta che la scheda si connette al broker MQTT, sia alla prima connessione sia dopo eventuali disconnessioni, viene inviato un payload contenente i seguenti campi sul topic **"attribute":**

- ip
- gateway
- mask
- dns
- mac

Topic: v1/devices/me/attributes QoS: 0


```
{
  "ip": "192.168.1.24",
  "gateway": "192.168.1.1",
  "mask": "255.255.255.0",
  "dns": "192.168.1.1",
  "mac": "B0:B2:1C:D4:50:B7"
}
```


Payload dati periodici

Quando un pacchetto è pronto, e la connessione con il broker è attiva, la scheda pubblica un payload sul topic **"telemetry"** contenente tutti i dati periodici di un pacchetto. Il payload è formato da un array JSON contenente un oggetto JSON per ogni pacchetto inviato.
Il singolo oggetto è composto da due chiavi:

- **"ts":** timestamp associato alla creazione del pacchetto in formato UNIX in millisecondi
  - Tipo: Stringa

- **"values":** oggetto contenente tutte le chiavi del pacchetto

- Tipo: Oggetto JSON

La scheda Trace&Follow è equipaggiata con un RTC con batteria tampone che le permette di mantenere l'ora corrente anche in caso di mancanza di alimentazione.
In base alle impostazioni presenti nel file di configurazione, la scheda può usare o il timestamp presente nel RTC, ottenere dalla rete il timestamp corrente oppure usarne uno relativo all'avvio della scheda:

- "timestamp_rel" = true: la scheda pubblicherà i messaggi utilizzando il timestamp relativo all'avvio
- "timestamp_rel" = false e "time_from_network" = true: la scheda la scheda otterrà il timestamp dalla rete e andrà ad aggiornare automaticamente quello presente nel RTC e lo utilizzerà per i messaggi.
- "timestamp_rel" = false e "time_from_network" = false: la scheda la scheda utilizzerà il timestamp salvato nel RTC e lo utilizzerà per i messaggi.

Nota: in caso di perdita dei dati del RTC, dovuto per esempio alla rimozione o alla scarica della batteria, la scheda userà un timestamp relativo, utilizzando come istante iniziale quello dell'avvio della scheda stessa.
Esempio di messaggio contenente un unico pacchetto, senza dati qualitativi, con topic **telemetry** → v1/devices/me/telemetry:

Topic: v1/devices/me/telemetry QoS: 0


```
[
  {
    "ts": "1709196304146",
    "values": {
      "energy_consumption": 0.497999996,
      "input2_value": 12,
      "virtual_input1_value": 0
    }
  }
]
```


Esempio di messaggio con pacchetti multipli:

Topic: v1/devices/me/telemetry QoS: 0


```
[
  {
    "ts": "1709196514146",
    "values": {
      "energy_consumption": 0.497999996,
      "input2_value": 31,
      "virtual_input1_value": 0
    }
  },
  {
    "ts": "1709196544146",
    "values": {
      "energy_consumption": 0.497999996,
      "input2_value": 0,
      "virtual_input1_value": 0
    }
  },
  {
    "ts": "1709196393401",
    "values": {
      "energy_consumption": 0.499000013,
      "input2_value": 16,
      "virtual_input1_value": 0
    }
  }
]
```

Payload dati non periodici di stato

In caso un input sia configurato in modalità 1, 3 o 4, il cambio di stato viene inviato sul topic **"telemetry"**, ma senza il timestamp associato, e sempre all'interno di un array JSON.

Esempio di input configurato in modalità 1, con topic **telemetry** → v1/devices/me/telemetry:

Payload dati non periodici di configurazione

In caso di dati non periodici modificati durante il funzionamento della scheda, questi verranno inviati sul topic **"attribute"** all'interno di un oggetto JSON.

Esempio con topic **attribute** → v1/devices/me/attributes:

Dati inviati dal server

Il server che gestirà i dati inviati dalla scheda dovrà inviare anch'esso alcuni dati.
L'invio dati lato server può avvenire in due modalità:

- **Server first**: in questo caso il server invierà alcuni parametri di configurazione che la scheda è pronta a ricevere.
- **Board first**: in questo caso la scheda esegue una richiesta al server, il quale dovrà rispondere con i dati richiesti.

Server first

In questo caso, è il server che decide quando inviare un dato alla scheda. L'invio di questo dato avviene tramite l'utilizzo del topic "**rpc**" al quale viene applicato il suffisso **/request/$request_id**
Esempio con topic **rpc** → v1/devices/me/rpc/

v1/devices/me/rpc**/request/**$request_id

Dove $request_id è un intero che identifica la singola richiesta effettuata. Questo id deve incrementare ad ogni richiesta. Quando la scheda effettua una nuova connessione il request_id deve ripartire da 0.

Il payload inviato dal server deve essere composto da un oggetto JSON contenente:

- La chiave **“method”:**
  - Tipo: Stringa
  - Descrizione: il metodo che la scheda deve eseguire
- La chiave **“params”:**
  - Tipo: oggetto JSON
  - Descrizione: contiene le chiavi con i parametri per l’esecuzione del rpc.

Esempio di richiesta Rpc effettuata dal server:

Topic: v1/devices/me/rpc/request/0 QoS: 0

```
{
  "method": "setPacketTime",
  "params": {
    "value": 200
  }
}
```

method	params	Descrizione
setInputFunctionState	**number:** numero dell'input **state:** nuovo stato della funzione associata all'input	Setta lo stato della funzione associata all’input selezionato quando configurato in modalità input
setConfigOutput	**number:** numero dell'output **polarity:** polarità dell'output [0 normale 1 invertita]	Setta la polarità dell’output selezionato
setOutput	**number**: numero dell'output **state:** new output state **time**: tempo in cui l'uscita rimane attiva (0 = infinito) [50 … infinito] millisecondi	Setta lo stato dell’output selezionato. È possibile inviare anche un impulso di durata impostabile utilizzando il parametro “time”
setThreshold	**value**: nuovo valore di soglia per standby [> 0] W	Setta la soglia di potenza istantanea utilizzata nella funzione di CutOff
setInactivityTime	**value**: nuovo tempo di inattività (0 = disattivato) [30 … infinito] secondi	Setta il tempo di idle massimo nella funzione di CutOff
setPacketTime	**value**: nuovo tempo di packet [1 …] s	Setta il tempo tra la creazione di un pacchetto e il successivo
reset	**value**: 1	Esegue un reset della scheda
setInputRegister	**number:** numero del registro [0-9] **value:** valore da inserire nel registro [0-65535]	Setta il valore di un input register associato al server modbus TCP gestito dalla scheda
setCurrentTimestamp	**Value:** timestamp GMT in formato UTC, in secondi	Aggiorna il valore del timestamp salvato all’interno del RTC della scheda

Quando la scheda riceve correttamente la richiesta rpc, essa pubblica lo stesso payload ricevuto dal server sul topic **"rpc"** al quale viene applicato il suffisso **/response/$request_id**

Esempio con topic **rpc** → v1/devices/me/rpc/

"rpc"/response/$request_id

Dove il parametro $request_id ha lo stesso valore di quello ricevuto.

Esempio di risposta inviata dalla scheda:

Topic: v1/devices/me/rpc/response/1 QoS: 0

```
{
  "method": "setPacketTime",
  "params": {
    "value": 200
  }
}
```

Quando un parametro viene modificato, la scheda pubblica un messaggio nel topic **"attribute"** contenente il nuovo valore.

Nota: Nel caso si vada a modificare il tempo di pacchetto (tramite RPC) la scheda interrompe il pacchetto corrente, lo invia al server e ne inizia unno nuovo con il nuovo tempo impostato.

Board first

Al contrario della modalità precedente, in questo caso è la scheda che richiede delle chiavi al server, il quale deve rispondere con un messaggio in formato JSON contenente i valori delle chiavi richieste.
La scheda pubblica la richiesta sul topic **"attribute"** al quale viene applicato il suffisso **/request /$request_id**

Esempio con topic **attribute** → v1/devices/me/attributes/

v1/devices/me/attributes**/request/$request_id**

Dove il parametro $request_id è un intero incrementale che identifica la richiesta.

Il messaggio di richiesta da parte della scheda è composto da un oggetto JSON contenente la seguente chiave:

- **"sharedKeys":**
  - Tipo: Stringa
  - Descrizione: contiene le chiavi dei parametri richiesti separati da una virgola

Esempio di richiesta:

Topic: v1/devices/me/attributes/request/0 QoS: 0

```
{
  "sharedKeys": "fw_title,fw_version,s_packet_time,s_energy_standby_threshold,s_inactivity_time"
}
```

La risposta del server deve essere pubblicata sul topic **"attribute"** al quale viene applicato il suffisso **/response /$request_id**

Esempio con topic **attribute** → v1/devices/me/attributes/

v1/devices/me/attributes**/response/**$request_id

Dove il parametro $request_id deve avere lo stesso valore di quello presente nella richiesta.

Il messaggio di risposta da parte del server è composto da un oggetto JSON contenente la seguente chiave:

- **"shared":**
  - Tipo: oggetto JSON
  - Descrizione: contiene le coppie chiave-valore relative alle chiavi richieste dalla scheda.

Le chiavi richieste dalla scheda (senza considerare quelle relative al firmware) sono alcune di quelle inviate dalla scheda con l'aggiunta del prefisso "s_". questo fa sì che se uno dei parametri viene modificato a scheda spenta, e quindi la richiesta RPC non ha successo, la scheda è comunque in grado di prendersi i valori corretti e aggiornati.

Le possibili chiavi richieste dalla scheda sono le seguenti:

- **s_energy_standby_threshold:**
  - Tipo: Intero
  - Descrizione: soglia di potenza istantanea sotto la quale la scheda considera la macchina in stato di idle in W. Si riferisce al parametro con la chiave inviata dalla scheda **"energy_standby_threshold"**.

- **s_inactivity_time:**
  - Tipo: Intero
  - Descrizione: indica dopo quanto tempo scatta il relay di CutOff quando la macchina si trova in stato di idle in secondi. Si riferisce al parametro con la chiave inviata dalla scheda **"inactivity_time"**.

- **s_packet_time:**
  - Tipo: Intero
  - Descrizione: campo che indica ogni quanto avviene la creazione di un pacchetto contenente dati periodici in secondi. Si riferisce al parametro con la chiave inviata dalla scheda **"packet_time"**.
- **fw_title:**
  - Tipo: String
  - Descrizione: titolo Firmware OTA

- **fw_version:**
  - Tipo: String
  - Descrizione: versione Firmware OTA

- **fw_size:**
  - Tipo: Intero
  - Descrizione: dimensione file binario OTA [bytes]

- **fw_checksum:**
  - Tipo: String
  - Descrizione: checksum firmware OTA calcolato con algoritmo MD5

Esempio di risposta del server:

Topic: v1/devices/me/attributes/response/1 QoS: 0

```
{
  "shared": {
    "fw_title": "TraceAndFollow",
    "fw_version": "1.0.2",
    "s_energy_standby_threshold": 3585,
    "s_inactivity_time": 95,
    "s_packet_time": 57
  }
}
```

Aggiornamenti OTA

Tramite l'interfaccia MQTT è possibile aggiornare il firmware della scheda Trace&Follow.

L'aggiornamento inizia con il server che pubblica nel topic **"attribute"** le seguenti chiavi:

- **fw_title:**
  - Tipo: Stringa
  - Descrizione: titolo Firmware
  - Valori ammessi: **"TraceAndFollow"**, se il nome inviato è diverso da quello riportato qui, il pacchetto non viene installato.

- **fw_version:**
  - Tipo: Stringa
  - Descrizione: versione Firmware. La versione deve essere maggiore di quella precedente perché il pacchetto venga installato.

- **fw_size:**
  - Tipo: Intero
  - Descrizione: dimensione file binario [bytes]

- **fw_checksum:**
  - Tipo: Stringa
  - Descrizione: checksum firmware calcolato con algoritmo MD5

- **fw_tag:**
  - Tipo: Stringa
  - Descrizione: il suo valore è formato da fw_title e fw_version separate da uno spazio.

- **fw_checksum_algorithm:**
  - Tipo: Stringa
  - Descrizione: algoritmo utilizzato per calcolare il checksum.
  - Valori ammessi: **"MD5"** questo campo deve avere valore fisso come riportato

Esempio di messaggio pubblicato dal server:

Topic: v1/devices/me/attributes QoS: 0


```
{
  "fw_title": "TraceAndFollow",
  "fw_version": "1.0.2",
  "fw_tag": "TraceAndFollow 1.0.2",
  "fw_size": 1437520,
  "fw_checksum_algorithm": "MD5",
  "fw_checksum": "ad4d254d429695ad7cfbe2492192a7d4"
}
```


Quando la scheda riceve dai parametri validi, interrompe la normale esecuzione del programma ed entra in Download mode. In questa modalità non accetta più chiamate RPC e non invia più i dati come durante il funzionamento ordinario.
Appena la scheda entra in questa modalità, pubblica un messaggio sul topic **"attribute"** contenente le seguenti chiavi:

- **fw_chunk_size:** indica la dimensione dei chunk con cui il server dovrà rispondere alle richieste della scheda
- **fw_chunk_number:** indica il numero di chunk che la scheda richiederà
- **fw_current_chunk:** indica il chunk che sta attualmente scaricando. In questo primo messaggio il suo valore sarà 0

Esempio di primo messaggio:

Topic: v1/devices/me/telemetry QoS: 0


```
{
  "fw_chunk_size": 15000,
  "fw_chunk_number": 95,
  "fw_current_chunk": 0
}
```


Successivamente la scheda pubblica un altro messaggio sullo stesso topic contenente lo stato dell'update:

Topic: v1/devices/me/telemetry QoS: 0


```
{
  "fw_state": "DOWNLOADING"
}
```

A questo punto inizia il download del file binario. La scheda pubblica sul topic

v2/fw/request/0/chunk/$chunk_number

un messaggio contenente la dimensione del chunk richiesto. Il valore di $chunk_number è un intero incrementale che corrisponde alle successive sezioni del firmware.
Il server dovrà rispondere sul topic

v2/fw/response/0/chunk/$chunk_number

con un numero di byte del file binario uguale al chunk_size. Il valore del parametro $chunk_number dovrà essere lo stesso della richiesta effettuata dalla scheda.

Esempio di risposta del server con $chunk_size = 100 e $chunk_number = 0

Topic: v2/fw/response/0/chunk/0 QoS: 0

```
e905 0220 2825 0840 ee00 0000 0000 0000 00ff ff00 0000 0001 2000 403f d4b9 0300 3254 cdab
0000 0000 0000 0000 0000 0000 7631 2e30 2e31 2d38 2d67 6438 6564 3430 342d 6469 7274 7900
0000 0000 0000 0000 5253 3232 3030 335f 4553 5033 325f 7632 0000 0000
```

Quando il chunk 0 è stato scaricato, la scheda pubblica una richiesta del successivo chunk e così via fino alla fine dei chunk. Ad ogni chunk la scheda pubblica il valore di **“fw_current_chunk”**

Esempio di messaggio pubblicato dopo la ricezione del chunk 1:

Topic: v1/devices/me/telemetry QoS: 0

```
{
  "fw_current_chunk": 1
}
```

Dopo la fine del download la scheda notifica la riuscita dell’aggiornamento settando il valore di “**fw_state”** a “DOWNLOADED”

Topic: v1/devices/me/telemetry QoS: 0

```
{
  "fw_state": "DOWNLOADED"
}
```

A questo punto la scheda si riavvia ed installa l’aggiornamento.

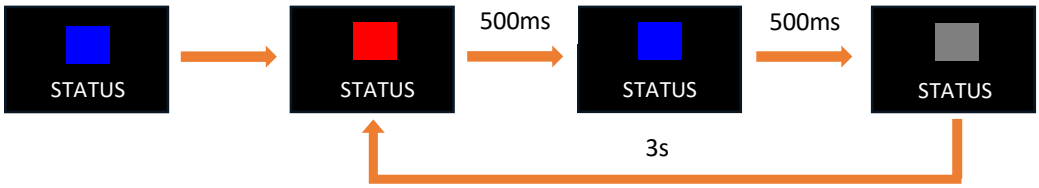
In caso di errore durante una fase dell'aggiornamento, il parametro **"fw_state"** viene settato a "FAILED" e il parametro **"fw_error"** settato con un accordo.

Esempio di un errore in cui la versione firmware non è maggiore di quella già presente:

Topic: v1/devices/me/telemetry QoS: 0

```
{
  "fw_error": "Firmware is already up to date",
  "fw_state": "FAILED"
}
```

Quando l'aggiornamento firmware ha inizio, il led STATUS della scheda diventa di colore BLU.
In Caso di errore il led lampeggia di rosso per 500ms, ritorna blu per altri 500ms, dopodiché rimane spento per 3 secondi e il ciclo è ripetuto per due volte.



GESTIONE I/O

La scheda Trace&Follow ha 4 input fisici e 4 output fisici. Oltre a questi, è predisposta anche per gestire input e output virtuali. Questi I/O virtuali sono utilizzati per unificare la gestione dei valori inviati dalla scheda in modo che, una volta configurato sulla scheda l'I/O, sia possibile avere sull'interfaccia MQTT un set di chiavi standard indipendentemente dalla tipologia di valore monitorato.
Esempio: Tramite la porta di espansione RS485 viene collegato un sensore di pressione che restituisce valori in bar. Il valore viene inviato tramite MQTT con la chiave virtual_input1_value. In un'applicazione analoga, il sensore di pressione viene sostituito da un encoder. La scheda invierà sempre il valore acquisito come virtual_input1_value. In questo modo le chiavi inviate dalla scheda sono sempre le stesse, mentre il significato del valore è configurabile lato server.

Configurazione input

Gli input della scheda possono essere configurati in 4 modalità:

1. **Normal:** l'input viene considerato come un normale input digitale. È possibile impostare un tempo di filtro (chiave "filter_time" nel file di configurazione), i cambi di stato inferiori al tempo di filtro impostato non vengano trasmessi al server, se viene superato il tempo di filtro viene trasmesso messaggio tramite MQTT.

2. **Counter:** in questa modalità la scheda conta gli impulsi che arrivano sul pin. I dati per gli input configurati in questo modo vengono trattati come dati periodici e quindi inseriti nei pacchetti con timestamp associato.

3. **Function:** in questa modalità viene cambiato il valore di una variabile booleana all'interno della scheda quando il segnale cambia di stato. All'avvio la variabile è a 0, un cambio di stato (fronte) del segnale la variabile va a 1. La variabile può essere riportata a 0 se il segnale viene mantenuto alto per almeno 3 secondi. Un esempio di applicazione può essere un tasto di richiesta manutenzione: l'operatore preme il pulsante e porta a 1 la variabile, per resettare lo stato deve premere lo stesso per 3 secondi.

4. **Blinking:** consente l'individuazione di un segnale ad impulsi in input. Esempio: segnale di una colonnina di segnalazione di un macchinario industriale, la luce lampeggiante corrisponde a macchina attiva (valore = 1), luce spenta corrisponde a macchina ferma (valore=0);

Per ogni input è poi possibile configurare anche la polarità dell'input (inversione logica):

- "polarity" = 0: Polarità normale.

- "polarity" = 1: Polarità Inversa.

Esempio: "polarity" = 0, in caso di 0V applicato sull'input, la scheda invia 0 come stato, in caso di 24V applicati, la scheda invia 1. Se "polarity" = 1 invece in caso di 0V applicato sull'input, la scheda invia 1 come stato, in caso di 24V applicati, la scheda invia 0.
Quando l'input è configurato in modalità "Counter", se la chiave d configurazione **"reset_counter"** è impostata a "true", significa che ad ogni pacchetto il valore del contatore viene resettato. Se invece è impostata a "false", il contatore non si resetta fino ad un valore massimo di 4,294,967,295 (intero unsigned a 32 bit).
Se l'input è configurato nelle altre 3 modalità, tramite la chiave **"add_to_packet"** è possibile includerlo o meno come dato periodico invece che inviarlo come dato asincrono. In particolare, se "add_to_packet" = "true", viene inviato come dato periodico, in caso contrario come dato asincrono.

Configurazione output

Per gli output è possibile configurare la polarità:

- "polarity" = 0: Polarità normale.
- "polarity" = 1: Polarità Inversa.

Esempio: "polarity" = 0, in caso di comando di set dell'output al valore 1, l'output verrà settato al valore di VCC, in caso di comando di set dell'output al valore 0, l'output verrà settato al valore di GND.
"polarity" = 1, in caso di comando di set dell'output al valore 1, l'output verrà settato al valore di GND, in caso di comando di set dell'output al valore 0, l'output verrà settato al valore di VCC.

SERVER MODBUS TCP

La scheda è dotata di un server Modbus TCP per la trasmissione e la ricezione di dati tramite il protocollo Modbus TCP.
Del server è possibile configurare:

- porta su cui il server è esposto
- slave id
- il massimo numero di client contemporaneamente connessi
- il tempo di idle prima che un client venga disconnesso (per tempo di idle si intende il tempo per cui il client non invia richieste al server)
- l'attivazione o disattivazione del server

Questi parametri sono contenuti nel file di configurazione.

Il server Modbus TCP può essere usato da un PLC esterno per inserire dati di produzione relativi alle produzioni correnti se la scheda Trace&Follow è utilizzata per monitorare una macchina di produzione.
In questo caso il PLC agisce da client e deve effettuare la connessione al server a bordo scheda.

Il server è esposto sullo stesso IP della scheda nella rete. Perché la scheda sia raggiungibile tramite il suo IP nella reta, essa deve essere connessa in Ethernet o Wi-Fi.

In caso la scheda non sia connessa a nessuna rete, oppure in 4G, è comunque possibile accedere al server Modbus collegandosi all'access point generato dalla scheda. Il server sarà esposto sollo stesso IP settato da file di configurazione su cui viene esposta la pagina web.

I comandi a cui risponde la scheda sono:

- Read Holding Registers (03)
- Read Input Registers (04)
- Write Multiple Registers (16)

Il comando di write scrive sugli Holding Registers.

Input registers

Tramite il server è possibile accedere a molti parametri della scheda, salvati negli input registers secondo la seguente tabella:

Add	significato	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	notes
0	Fisso a 1	1																
1	Versione Hardware	major version								release								
2	Versione Firmware	major v				minor v				release								
3	packet time	Valore in secondi																
4	transmission time	Valore in secondi																
5	inactivity time	Valore in secondi																
6	energy stand-by threshold	Valore in watt																
7	connection type	1-> Ethernet, 2-> Wi-Fi, 3-> LoRa, 4 -> 4G																
8	input type	type in 4				type in 3				type in 2				type in 1				0-> disabilitato, 1-> normal, 2-> counter, 3-> function
9	AP enabled	1-> abilitato, 0-> disabilitato																
10	Nome AP1	carattere 29								carattere 28								Carattere 29 più a sinistra, carattere 0 più a destra. Caratteri non utilizzati settati a 0
11	Nome AP 2	carattere 27								carattere 26								
12	Nome AP 3	carattere 25								carattere 24								
13	Nome AP 4	carattere 23								carattere 22								
14	Nome AP 5	carattere 21								carattere 20								
15	Nome AP 6	carattere 19								carattere 18								
16	Nome AP 7	carattere 17								carattere 16								
17	Nome AP 8	carattere 15								carattere 14								
18	Nome AP 9	carattere 13								carattere 12								
19	Nome AP 10	carattere 11								carattere 10								
20	Nome AP 11	carattere 9								carattere 8								
21	Nome AP 12	carattere 7								carattere 6								
22	Nome AP 13	carattere 5								carattere 4								
23	Nome AP 14	carattere 3								carattere 2								
24	Nome AP 15	carattere 1								carattere 0								
25	IP AP 1	byte 1 IP								byte 2 IP								byte4.byte3. byte2.byte1
26	IP AP 2	byte 3 IP								byte 4 IP								
27	cut-off abilitato	1-> abilitato, 0-> disabilitato																
28	localization abilitata	1-> abilitato, 0-> disabilitato																
29	reserved																	
30	network connection status	1->connesso, 0->non connesso																

31	Stato connessione MQTT	1->connesso, 0->non connesso									
32	IP tab 1	byte 1 IP			byte 2 IP						byte4.byte3. byte2.byte1
33	IP tab 2	byte 3 IP			byte 4 IP						
34	Modbus client massimi consentiti										
35	Modbus client connessi										
36	Modbus idle timeout	value in seconds									
37	Stato IO	riservato	o4	o3	o2	o1	in4	in3	in2	in1	Valori possibili 0/1
38	Potenza istantanea	Valore in watt									
39	riservato										
40	RPC 0	Settato da chiamata RPC									Inviato con indirizzo 3000
41	RPC 1	Settato da chiamata RPC									Inviato con indirizzo 3001
42	RPC 2	Settato da chiamata RPC									Inviato con indirizzo 3002
43	RPC 3	Settato da chiamata RPC									Inviato con indirizzo 3003
44	RPC 4	Settato da chiamata RPC									Inviato con indirizzo 3004
45	RPC 5	Settato da chiamata RPC									Inviato con indirizzo 3005
46	RPC 6	Settato da chiamata RPC									Inviato con indirizzo 3006
47	RPC 7	Settato da chiamata RPC									Inviato con indirizzo 3007
48	RPC 8	Settato da chiamata RPC									Inviato con indirizzo 3008
49	RPC 9	Settato da chiamata RPC									Inviato con indirizzo 3009

I registri RPC 0-9, sono registri il cui valore può essere settato tramite RPC dall'interfaccia MQTT. Il metodo **"setInputRegister"** riportato nella tabella RPC method. Questo fa sì che sia possibile inviare dal server dei valori che possono essere letti dal PLC collegato alla scheda Trace&Follow. Inoltre, nel file di configurazione, nel campo "MODBUS_TCP" è presente la chiave "send_input". Se questa chiave è settata a "true", quando il valore viene scritto tramite RPC, la scheda invia sul topic **"telemetry"** con la chiave **"virtual_input300x_value"** il valore che è appena stato settato.

Esempio di richiesta in arrivo dal server con risposta della scheda quando “send_input” = “true”:

Holding registers

Al contrario degli Input Registers, gli Holding Registers possono essere scritti dal PLC. Questi registri servono a monitorare la produzione della macchina a cui la scheda è collegata.
I registri sono divisi in 5 blocchi, il che permette di monitorare fino a 5 produzioni simultanee.
Per ogni produzione è possibile settare:

- Il **codice** della produzione (Max 10 caratteri): registri x0 – x4, 2 caratteri ASCII per ogni registro.
- Il contatore dei pezzi **OK**: registro x5
- Il contatore dei pezzi **SCARTO**: registro x6
- Il **tempo ciclo** della produzione: registro x7
- Il **target** di produzione: registro x8

È infine presente un registro tramite cui il PLC può inviare al server lo stato della macchina.

Questi sono dati che il PLC scrive e che la scheda invia al server tramite l’interfaccia MQTT.

La scheda pubblica questi dati non appena essi vengono scritti utilizzando le chiavi **“virtual_input(N)_value”** secondo la seguente tabella:

Indirizzo	Significato	Virtual Input Value	Produzione
1	stato macchina	4001	/
10	codice prodotto 1	4014	**PRODUZIONE 1**
11			
12			
13			
14			
15	contatore pezzi OK 1	4015	

16	contatore pezzi scarto 1	4016	
17	tempo ciclo 1	4017	
18	target produzione 1	4018	
20	codice prodotto 2	4024	**PRODUZIONE 2**
21			
22			
23			
24			
25	contatore pezzi OK 2	4025	
26	contatore pezzi scarto 2	4026	
27	tempo ciclo 2	4027	
28	target produzione 2	4028	
30	codice prodotto 3	4034	**PRODUZIONE 3**
31			
32			
33			
34			
35	contatore pezzi OK 3	4035	
36	contatore pezzi scarto 3	4036	
37	tempo ciclo 3	4037	
38	target produzione 3	4038	
40	codice prodotto 4	4044	**PRODUZIONE 4**
41			
42			
43			
44			
45	contatore pezzi OK 4	4045	
46	contatore pezzi scarto 4	4046	
47	tempo ciclo 4	4047	
48	target produzione 4	4048	
50	codice prodotto 5	4054	**PRODUZIONE 5**
51			
52			
53			
54			
55	contatore pezzi OK 5	4055	
56	contatore pezzi scarto 5	4056	

57	tempo ciclo 5	4057	
58	target produzione 5	4058	

Quando il PLC inizia una nuova produzione, deve settare i registri x0 – x4 con il codice associato alla produzione. La scheda a quel punto invierà il codice sotto forma di un'unica stringa con la chiave "virtual_input_40x4_value". È possibile settare anche tempo ciclo e target che verranno inviati in modo analogo.
Durante la fase di produzione il PLC può settare i registri relativi ai contatori in base alla statistiche di produzione. Quando uno di questi registri viene settato ad un valore diverso da 0, la scheda invia al server il corrispondente valore e riporta a 0 quel registro per indicare la corretta trasmissione verso il server.
Una volta che la produzione è terminata, il PLC deve settare i registri del codice di produzione, del tempo ciclo e del target a 0. La scheda invia a 0 i parametri che sono stati resettati e una stringa nulla come codice per indicare che la produzione è terminata.

Esempio inizio produzione con solo target e codice impostati:

Topic: v1/devices/me/telemetry QoS: 0

```
[
  {
    "virtual_input4014_value": "GreenPen1",
    "virtual_input4018_value": 576
  }
]
```

Esempio scrittura contatori:

Topic: v1/devices/me/telemetry QoS: 0

```
[
  {
    "virtual_input4015_value": 16,
    "virtual_input4016_value": 8
  }
]
```

È possibile anche non settare entrambi i contatori. La scheda invia solo quelli che sono stati modificati:

Topic: v1/devices/me/telemetry QoS: 0

```
[
  {
    "virtual_input4015_value": 6
  }
]
```

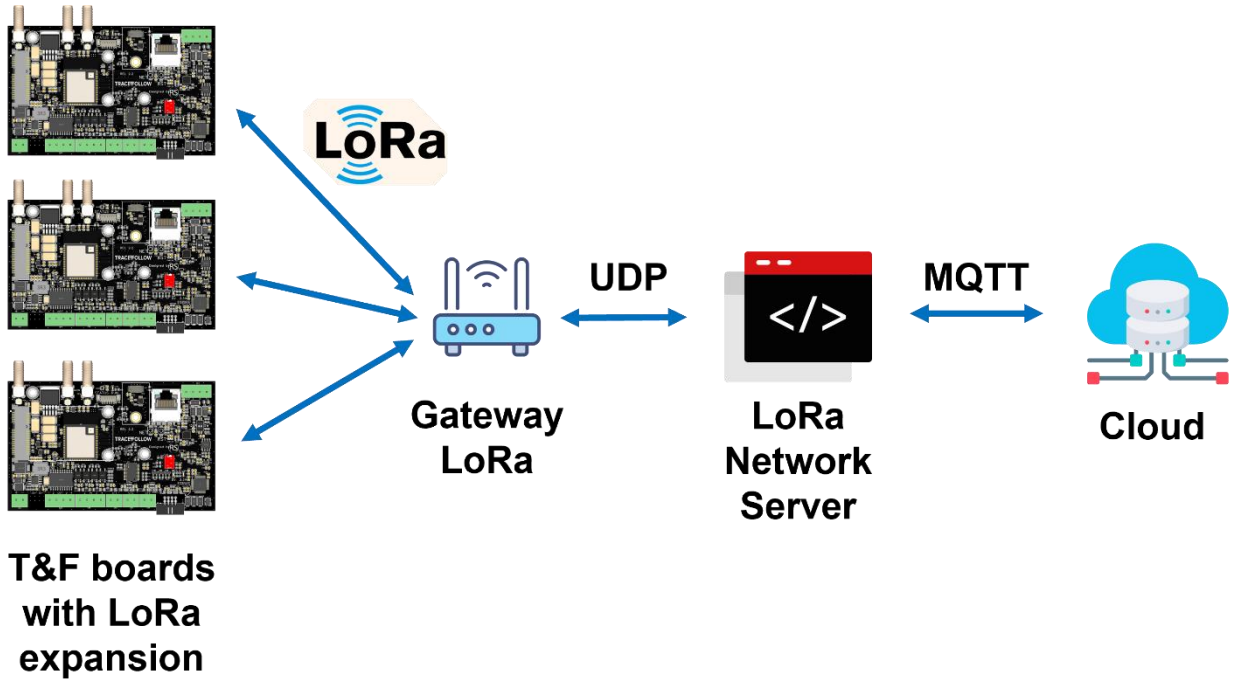
Esempio fine produzione:

INTERFACCIA LoRaWAN

Tramite l'espansione LoRaWAN è possibile configurare la scheda Trace&Follow per lavorare tramite il protocollo LoRaWAN. Questo protocollo permette di inviare e ricevere dati a lungo raggio tramite un gateway LoRa in ambienti in cui non è presente connettività (il gateway deve comunque essere connesso in rete).
Tramite il file di configurazione è possibile impostare i parametri di connessione al gateway.
La connessione avviene tramite la procedura Over-The-Air-Activation (OTAA) in classe A.
I parametri della comunicazione sono:

- LoRaWAN MAC version: 1.0.2
- LoRaWAN Regional Parameters revision: B
- ADR algorithm: Default ADR algorithm (LoRa only)
- Max EIRP: 27
- Uplink Interval: 1

Schema dell'infrastruttura LoRa:



Le schede Trace&Follow inviano tramite pacchetti LoRa codificati in un array di byte che è possibile decodificare attraverso la funzione allegata a questo documento.
È possibile anche inviare dati alla scheda tramite questo protocollo utilizzando la funziona di codifica allegata a questo documento oppure scaricabile tramite il sito https://idtsolution.com/trace-and-follow/.
Le due funzioni andranno implementate sul LoRa network server. La funzione di decodifica restituirà un JSON mentre la funzione di codifica si aspetta in ingresso un JSON. Entrambi sono formattati come descritto nel paragrafo riguardante la connessione MQTT.

Nota: i parametri relativi alla misurazione qualitativa dell'energia non sono disponibili tramite questo protocollo, così come l'interfaccia con il server modbus TCP. Inoltre, l'aggiornamento OTA non è disponibile.

REVISIONI

REVISIONI		
N.	**Descrizione**	**Data**
0	Primo rilascio	04/03/2024
1	Introduzione RTC	14/03/2024
2	Fix tabella modbus TCP	04/04/2024
3	Introduzione dati qualitativi energia, connessione LoRaWAN	11/06/2024
4	Aggiornate unità di misura dati inviati	18/07/2024

Questo documento costituisce una documentazione tecnica; per ulteriori dettagli e informazioni, si rimanda al manuale completo del Trace&Follow™.