CAREL – Confidential



**REQUIREMENTS SPECIFICATION**

*Gateway Middle End*

rev. vedi cronologia

DRAFT

Index

[Index 2](#_Toc31803582)

[Revisione 3](#_Toc31803583)

[1 Introduzione 4](#_Toc31803584)

[1.1 Scopo della RS 4](#_Toc31803585)

[1.2 Definitioni, acronimi e abbreviazioni 4](#_Toc31803586)

[Modo di funzionamento (macro) 5](#_Toc31803587)

[1.3 Prima accensione modello WiFi 5](#_Toc31803588)

[1.4 Inizializzazione RTC 6](#_Toc31803589)

[1.5 Verifica file system 6](#_Toc31803590)

[Certificati CA 6](#_Toc31803591)

[File di modello (MODEL\_TABLE) 6](#_Toc31803592)

[2 Polling Engine 7](#_Toc31803593)

[3 MQTT Engine 8](#_Toc31803594)

[4 Modelli dispositivo e tipi di dati 9](#_Toc31803595)

[5 Aggiornamento files nel gateway 10](#_Toc31803596)

[5.1 Aggiornamento file di modello 10](#_Toc31803597)

[5.2 Aggiornamento file dei certificati 10](#_Toc31803598)

[5.3 Aggiornamento FW del dispositivo connesso 10](#_Toc31803599)

[6 Interfaccia WiFi 13](#_Toc31803600)

[6.1 Generale 13](#_Toc31803601)

[6.2 Contenuto della pagina di configurazione 13](#_Toc31803602)

[7 ESP32 partizionamento memoria 15](#_Toc31803603)

[8 Alcune note relative all’hardware 16](#_Toc31803604)

[Appendice A – Aggiornamento FW del target 17](#_Toc31803605)

Revisione

|  |  |  |  |
| --- | --- | --- | --- |
| Rev. | Rev. date | Author | Note |
| *0.01* | *06/01/2020* | *A.Bilato* | 1° draft |
| 0.02 | 24/02/2020 | A.Bilato | 2° draft |
| 0.03 | 26/02/2020 | A.Bilato | Aggiunto agg. FW GME |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Introduzione
   1. Scopo della RS

Queste specifiche si riferiscono alla realizzazione di una FW che implementa il Gateway Middle End, in breve un dispositivo che è in grado di :

* monitorare un dispositivo (1)
* trasferire i dati monitorati un MQTT broker.
* ricevere valori di variabili da inviare al dispositivo via MQTT
* ricevere il nome del file che descrive il modello del dispositivo via MQTT e trasferirlo via HTTPS.

Il FW con opportune varianti dovrà essere eseguito sia sul modello WiFi che sul modello 2G.

Le specifiche sono divise in più files per comodità e per poter essere stese congiuntamente con il dipartimento IoT che si occuperà della realizzazione della parte cloud server.

A tale scopo diversa documentazione è disponibile al link

<https://drive.google.com/open?id=1znYUtkt66RGxlrIFfflsu1rycst2YjfG>

Il verrà realizzato su ESP32, e comprende le seguenti parti:

1. Libreria Modbus Master;
2. Libreria MQTT;
3. Libreria CBOR;
4. Routine per il download via HTTPS dei file
5. Polling engine;
6. Sistema di aggiornamento del FW del dispositivo connesso e del gateway stesso in maniera standardizzata
   1. Definitioni, acronimi e abbreviazioni

AP = WiFi access point

GME\_AP = gateway che agisce da access point  
GME\_WIFI = Gateway Middle End versione WiFi

GME = Gateway Middle End   
FW = firmware

IoT = Internet of Things

MonDev = il dispositivo connesso al GME attraverso la RS485

OTA = Over The Air

SW = software  
MODEL\_TABLE = tabella che definisce quali registri Modbus debbano essere letti

Modo di funzionamento (macro)

* 1. Prima accensione modello WiFi

Alla prima accensione il modello WiFi non è in grado di connettersi al cloud in quanto la connettività è fornita da terzi attraverso tipicamente un modem/router.

Per configurare il GME lo stesso mette a disposizione un web server interno che espone le pagine di configurazione i punti salienti sono:

* Il nome dell’AP interno sarà “cgatem\_xxxx”
* La pagina di configurazione sarà raggiungibile all’indirizzo 192.168.0.1

* 1. Inizializzazione RTC

La libreria non gestisce in alcun modo l’orologio ma si appoggia ad un NTP server,

questo è settato su “pool.ntp.org” alla porta standard 123.

Se il sistema non riesce a sincronizzarsi con l’NTC server il sistema attenderà che questo passo sia concluso reiterando le richieste.

Questo è obbligatorio in quanto la validazione dei certificati di sicurezza passa obbligatoriamente per la verifica della validità temporale degli stessi.

* 1. Verifica file system

Il sistema necessita della presenza di alcuni “files”, la loro presenza/consistenza dovrà essere validata alla partenza della macchina.

Certificati CA

E’ previsto che il sistema possa memorizzare n.2 certificati CA, questo per permettere le seguenti funzionalità:

* avere la possibilità di aggiornare il certifica CA se ne sorge la necessità.  
  In pratica il sistema prova ad utilizzare il certificato n.1 se fallisce prova ad usare il n.2, in questo modo è sempre possibile aggiornare i certificati.  
  L’unico caso scoperto è quello di un dispositivo che resti spento per lungo tempo e che perda n.2 aggiornamenti, in questo caso si recupererà la macchina caricando il certificato via seriale anziché via OTA.
* avere la possibilità di utilizzare un certificato non CAREL.  
  Questo sarà possibile tramite un codice di produzione custom che caricherà nel sistema il certificato del committente.  
  TDB in CAREL o USR
* L’onere di aggiornare in tempo o quando necessario i certificati è lato cloud.  
  Se il GME resta inattivo per lungo tempo ed entrambi i CA sono non validi il GME deve essere aggiornato manualmente in CAREL

File di modello (MODEL\_TABLE)

Questo file contiene la definizione delle variabili da leggere ciclicamente via Modbus  
vedi <GME_Modbus_scan_datatable_Flash_ipotesi_2.xlsx>

La verifica consiste nel verificare se il file :

* è un file di modello;

questo si verifica facilmente in quanto nel file i primi caratteri saranno sempre GME\_MBT

* verificare se il checksum del file è corretto;  
  il CRC del file è alla fine del file ed è un semplice CRC16 che usa lo stesso polinomio del Modbus.
* L’informazione se il file è corretto o meno verrà trasferita tramite il topic \hello nel payload alla voce "gid" estraendo il dato dal modello.   
  Vedi file [payload-hello.cbor](https://drive.google.com/open?id=1Fn2AkghjOlY3gKg6rq46WZ9GJqSetOGf) formato del topic /hello

1. Polling Engine

In questa sezione viene descritto come funziona la parte di FW che si occupa di leggere ciclicamente i dati via Modbus.

Nei vari documenti si troveranno riferimenti a “line” che avrà sempre valore 1 in quanto il  
GME ha una unica porta seriale.

La configurazione della linea seriale viene fatta attraverso il comando descritto nel documento [payload-set\_lines\_config.cbor](https://drive.google.com/open?id=1WdL3k1ttpWaz8rz0JBIX7y81Cpo9svfJ).

1. MQTT Engine

Il protocollo usato per trasferire i dati campionati è MQTT nella versione 3.1.1.

I vari topic pubblicati e/o sottoscritti sono codificati attraverso il formato CBOR standardizzato dalla RFC7049.

Il payload dei vari possibili comandi è descritto nel dettaglio nei files presenti a questo link <https://drive.google.com/open?id=1IJpGUBONSAXrd71AtTDmHrDc2wQGiFEX>.

1. Modelli dispositivo e tipi di dati

Il GME campiona dei dati via Modbus e li trasferisce via MQTT al server incapsulati in un messaggio CBOR.  
Il formato del messaggio CBOR <https://drive.google.com/open?id=1BuX2NeawAvz4Py_tf2SRti_enep9hSQe>.

I tipi di dati che è in grado di trattare sono elencati di seguito e troveranno poi applicazione nella tabella di modello (da ora MODEL\_TABLE) descritta in

<GME_Modbus_scan_datatable_Flash_ipotesi_2.xlsx> sia per l’invio di valori via CBOR.

Tale tabella deve contenere un totale massimo di 100 valori da campionare, allarmi compresi, la verifica che tale limite sia rispettato è in carico al cloud che metterà a disposizione anche il tool di selezione delle variabili da campionare.

I dati vengono campionati secondo 3 modalità:

* Allarmi;   
  il polling è best effort ovvero la scansione avviene alla massima velocità possibile compatibilmente con il numero di allarmi in tabella.
* High polling;  
  Il tempo di campionamento minimo è di 60 sec., eccezionalmente e per il solo cliente Rivacold tale tempo sarà ridotto a 30 sec. Ma con un numero di variabili da monitorare non superiore a 15.
* Low polling;  
  Il tempo di campionamento massimo è di 32767 sec.

**Endianess**

Per i dati a 32 bit si deve tener conto anche dell’endianess in quanto i dati sono trasferiti al massimo a registri di 16 bit.  
A questo scopo è predisposto il bit 7 del dei flags in MODEL\_TABLE che indica se la codifica seguita è Little o BigEndian.

**Isteresi**

Uno dei campi presenti in MODEL\_TABLE è il campo isteresi, questa viene usata per determinare se il dato è variato più di un certo ammontare e quindi deve essere inviato via MQTT.  
L’isteresi è sempre espressa come dato a 32 bit Little Endian indipendentemente dall’endianess del dato letto, questo semplifica la generazione del modello e il dato non deve essere convertito ulteriormente.

Se il dato letto è di tipo a 16 bit (1 holding register ad esempio) il dato dell’isteresi sarà

lo stesso a 32 bit ma il valore dovrà essere castato a 16 bit.

**Dati di linearizzazione**

Nel MODEL\_TABLE sono presenti due valori a espressi come dati a 32 bit in Little Endian indipendentemente dall’endianess del dato associato ad essi.

Questi devono essere usati con tutti i tipi di dato ad eccezione dei bool applicando la formula

Valore ritornato = ((Valore letto) \* ”Lin A” ) + “Lin B”

1. Aggiornamento files nel gateway

Il GME per il suo funzionamento richiede alcuni files, questi possono essere installati all’atto della produzione o devono essere scaricati via cloud una volta installato l’apparecchio.

Questi sono :

1. file che descrive il modello del dispositivo collegato, MODEL\_TABLE.
2. file dei certificati CA
3. file di aggiornamento FW del dispositivo target.
4. file di aggiornamento del FW del GME

Il trasferimento di questi files avviene tramite collegamento HTTPS ad un server, di seguito vengono elencate le varie modalità.

* 1. Aggiornamento file di modello

Il GME riceverà il MODEL\_TABLE da scaricare tramite il topic MQTT \req con il comando SET\_DEV\_CONFIG, al cui interno vi sono le informazioni necessarie per eseguire il download.  
Al ricevimento di questo comando il GME deve :

1. bloccare il polling Modbus se attivo
2. fare il flush dei dati campionati se presenti
3. scaricare il file passato via comando tramite HTTPS, le credenziali sono presenti nel payload.
4. alla fine del trasferimento del file il sistema deve fare un reboot, la catena di controllo dei file viene eseguita nuovamente.

Per i dettagli vedi la descrizione del comando in

<https://drive.google.com/open?id=1IJpGUBONSAXrd71AtTDmHrDc2wQGiFEX>

* 1. Aggiornamento file dei certificati

Il GME riceverà il CA da scaricare tramite il topic MQTT \req con il comando UPDATE\_CA\_CERTIFICATE, al cui interno vi sono le informazioni necessarie per eseguire il download.  
Al ricevimento di questo comando il GME deve :

1. puntare al file CA non utilizzato
2. scaricare il file indicato nel payload tramite HTTPS, le credenziali sono presenti nel payload stesso.

Per i dettagli vedi la descrizione del comando in

<https://drive.google.com/open?id=1IJpGUBONSAXrd71AtTDmHrDc2wQGiFEX>

* 1. Aggiornamento FW del dispositivo connesso

Il GME è in grado di aggiornare il FW del dispositivo target ad esso collegato.  
Per far questo utilizza il protocollo Modbus con il comando standard “Write File Record” 0x15. La descrizione del protocollo di trasferimento è descritta nell’[Appendice A](#appendice_a)   
dove vi sono descritti i dettagli implementativi.  
Il GME riceverà il FW da scaricare tramite il topic MQTT \req con il comando UPDATE\_DEV\_FIRMWARE, al cui interno vi sono le informazioni necessarie per eseguire il download.  
Al ricevimento di questo comando il GME deve :

1. bloccare il polling Modbus se attivo
2. fare il flush dei dati campionati se presenti
3. scaricare il file passato via comando tramite HTTPS e trasferirlo al dispositivo target.  
   Il GME riceverà il file del FW da aggiornare ricevendo questo   
   [payload-update\_dev\_firmware-req.cbor](https://drive.google.com/open?id=1JMApiQEPfN7dYzpGY_Prvw4HlR4B4sMf)

Dentro al payload ci sono anche le credenziali per l’accesso al file via HTTPS.  
In sintesi significa che il GME recupera via HTTPS dei chunk di FW e li spedisce al target, in caso di errore tenta i retry come da prassi ma se non ci riesce abortirà la sequenza.  
Vedi anche [Appendice A](#appendice_a).

1. il GME al termine dell’operazione attende il numero di secondi indicati nel payload e poi riprende il polling.

In caso di errato trasferimento e di FW non correttamente installato succederà che il target non risponderà al polling e quindi il GME comunicherà un allarme di dispositivo offline. L’operatore via cloud potrà re-inviare il comando di aggiornamento FW del target e riprovare.

Per i dettagli vedi la descrizione del comando in

<https://drive.google.com/open?id=1IJpGUBONSAXrd71AtTDmHrDc2wQGiFEX>

* 1. Aggiornamento FW del gateway

Il GME è in grado di aggiornare il proprio FW, riceverà il FW da scaricare tramite il topic MQTT \req con il comando UPDATE\_GW\_FIRMWARE, al cui interno vi sono le informazioni necessarie per eseguire il download.  
Al ricevimento di questo comando il GME deve :

1. bloccare il polling Modbus se attivo
2. fare il flush dei dati campionati se presenti
3. memorizzare il CID passato nel comando nella memoria NVM
4. scaricare il file passato via comando tramite HTTPS e trasferirlo nella propria memoria FLASH   
   Il GME riceverà il file del FW da aggiornare ricevendo questo   
   [payload-update\_gw\_firmware-req.cbor](https://drive.google.com/open?id=10R980Hv8OSLNRtEFDWVEYlDnMi7c49ZT)

Dentro al payload ci sono anche le credenziali per l’accesso al file via HTTPS.  
In sintesi significa che il GME recupera via HTTPS dei chunk di FW in caso di errore tenta i retry come da prassi ma se non ci riesce abortirà la sequenza.

1. il GME al termine dell’operazione invierà il messaggio di fine trasferimento file, ed eseguirà un reboot.
2. Alla partenza invierà il messaggio \hello in cui riporterà il CID memorizzato al punto 3, dopodichè azzererà il CID.

Per i dettagli vedi la descrizione del comando in

<https://drive.google.com/open?id=10R980Hv8OSLNRtEFDWVEYlDnMi7c49ZT>

1. Interfaccia WiFi
   1. Generale

Il sistema alla partenza ha la modalità AP attiva e avrà SSID settato a “cgatem-xxxx“ dove xxyyzz sono le ultime 3 cifre del MAC Address.

Avrà il webserver attivo all’indirizzo fisso 192.168.100.1 e un client (smartphone tipicamente) vi si potrà connettere.

Il GME\_WIFI avrà il DHCP attivo, di default, sia lato AP (max. 1 connessione) sia lato STA.

Alla prima accensione l’utente dovrà impostare i parametri di connessione ed una volta salvati il sistema ripartirà applicando i nuovi parametri.

* 1. Contenuto della pagina di configurazione

La pagina/e di configurazione che il webserver espone conterranno le seguenti opzioni

1. La pagina di configurazione deve essere essere protetta  
   Alla prima accensione verrà obbligatoriamente richiesta l’impostazione di   
   “user name” e “password” per il prossimo accesso alle pagine di configurazione,  
   poi nella pagina potranno essere cambiate
2. Modalità di funzionamento AP
   1. SSID trasmesso (default “cgatem\_ xxyyzz vedi sopra)
   2. Password AP
   3. Indirizzo AP/webserver (default 192.168.0.1)
   4. DHCP on/off (default on)
   5. Indirizzo base DHCP (default 192.168.0.10)
3. Modalità di funzionamento STA
   1. nome AP a cui connettersi
   2. Modo WPA/WAP2 (è fissa ma ricorda all’utente
   3. Password AP
   4. modalità indirizzo dato da DHCP on/off (default on)
   5. indirizzo statico
   6. netmask
   7. default gateway
   8. DNS primario/secondario
4. NTP
   1. indirizzo server;  
      alla prima accensione il GME proporrà “pool.ntp.org” e poi l’utente decide se cambiarlo o meno.

Ci sarà la possibilità di premere un tasto sulla pagina per attivare WPS.

1. ESP32 partizionamento memoria

L’ESP32 WROOM è dotato nell’implementazione di CAREL di 4MB di FLASH la memoria

sarà così partizionata

|  |  |  |
| --- | --- | --- |
| **Scopo** | **Dimensione** | **Note** |
| FW applicativo n.1 | ~1 MB | Questo è il FW che esce di fabbrica installato nel dispositivo. |
| FW applicativo n.2 | ~1 MB | Questo è il FW di un eventuale aggiornamento  totalmente auto contenuto che se installato  parte al posto di FW1. |
| SPIFS | 1 MB |  |
| Libero per usi futuri | 1 MB |  |

FW n.1 e n.2 sono ~1MB perché in ogni caso va previsto lo spazio per il bootloader che si incarica di :

1. capire quale dei due FW è abilitato
2. alla partenza calcola il CRC per verificare se è corretto
3. esegue il “jump” coerentemente ai punti precedenti

A seguito di un factory reset il sistema deve sempre puntare al FW n.1 che è garantito funzionante nelle sue parti principali e quindi consente sempre di scaricare un aggiornamento.

Appendice A – Aggiornamento FW del target

Il trasferimento avverrà tramite Modbus file transfer a partire dal file di indice passato.  
La regola di trasferimento è qui sotto descritta, è un estratto dalle specifiche interne CAREL per l’implementazione di un prodotto a cui fare riferimento.

As an optional feature, a slave device can support file transfer feature (commands 0x14 “Read File Record” and 0x15 “Write File Record”). This feature shall be implemented following the “MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3”, 6.14 and 6.15.

According with the document the slave device the default configuration is 10000 records available with max record length 2 byte. File number can be in the range 1 to 0xFFFF, but legacy devices support only the range 1 to 10. If a file is longer than 2bytes\*10000, file index auto-increasing shall be managed.  
By an example this means :

* suppose that the initial file index is 6410
* after transfer 20000 bytes of data file index become 6411
* the 20001 byte will be transferred with Modbus file index 6411 and so on

Upgrade from Gateway: if a Carel device is connected to a Carel Gateway, the device should be upgraded by the gateway using Modbus file transfer functionality. File index will be set by the caller, record length is 2.

Note: the length of the upgrade file is not known by the slave so, in order to recognize that the transfer is completed and the file ready the master (the gateway) shall send a “0 length packet” to let the client able to close the file and start upgrade procedure.