



[Onboarding di base]

Sistema
Domotico su
piattaforma
Arduino



AUTORE: ANDREA LANDO

REVISIONE 1.0.0

DATA ULTIMA REVISIONE: 22.05.2026

Sommario

Prefazione	6
Introduzione al percorso di onboarding	6
La filosofia progettuale di DomoManager	6
Perché un onboarding strutturato è essenziale	6
Comprendere l'architettura	6
Acquisire un metodo di lavoro	7
Evitare errori tipici	7
A chi è destinato questo documento	7
Conclusione della prefazione	7
Architettura Generale del Sistema	9
Filosofia generale del sistema	9
MAIN - Punto di ingresso del sistema	9
FRONTEND ENGINE – Orchestratore dei moduli	9
DMSETUP – Inizializzazione hardware e buffer	10
DOMOMANAGER – Il cuore del sistema	11
CICLO PRINCIPALE – Round-robin deterministico	11
DIAGNOSTICA – Watch, Watchdog, DeviceErrors	12
Prerequisiti e Materiale Necessario	13
Obiettivo del capitolo	13
Materiale hardware necessario	14
Unità di controllo	14
Gateway RS485 → Ethernet	14
Dispositivi Modbus RTU	14
Cablaggio RS485	14
Materiale software necessario	14
Schema di rete consigliato	15
Tabella IP consigliati	15
Tabella ID Modbus consigliati	16
Prerequisiti tecnici dell'installatore	16
Checklist pre-installazione	16
Configurazione – Opta + (2) Waveshare + (2) I/O Modbus	17
Configurazione fisica	17
Assegnazione IP ai Waveshare	17
Collegamento RS485 con l'IO remoto Ebyte 4DI+4DO	19
Configurazione logica (OPTA)	24
Configurazione dispositivi in DomoManager (DMSetup.hpp)	24

Definizione aree del buffer	25
Configurazione Toggles e Splits	26
Analisi della configurazione di esempio	27
File DMSetup.hpp	28
File DomoManager.ino	32
File DMFrontend.hpp.....	35
Avvio di DomoManager	36
Diagnostica iniziale	37
Watchdog (da wdConfig in DMSetup.hpp)	37
Aree monitorate	37
Stato dispositivi.....	37
Avvio del sistema	37
Guida per installatori con diagrammi di flussi e dipendenze	39
Flusso di avvio del sistema (BOOT FLOW)	39
Dipendenze tra i moduli (Dependency Graph)	40
Flusso Modbus (Gateway → Opta → Buffer)	41
Flusso di configurazione (DMSetup → DomoManager)	42
Flusso diagnostica (Watchdog, Watch, DeviceErrors)	43
Troubleshooting rapido per installatori.....	44
Nessuna lettura Modbus?	44
Un'area non cambia?.....	44
Troubleshooting Avanzato.....	45
Obiettivo del capitolo	45
Diagramma generale del troubleshooting.....	45
Problemi di rete (Ethernet).....	46
Problemi Modbus TCP	47
Problemi RS485.....	47
Problemi dispositivi Modbus	47
Problemi buffer (DomoManager)	48
Problemi watchdog.....	48
Problemi IP Manager	49
Tabella riassuntiva problemi → soluzioni	49
Glossario Tecnico.....	50
■ Buffer.....	50
■ Area	50
■ Virtual Area	50
■ Toggle.....	50

 Split.....	50
 Route	50
 Scene	50
 Regola dinamica	51
 Trend	51
 Bitmask.....	51
 H24 / RT.....	51
 Watchdog.....	51
 Cooldown IP	51
 Slot Modbus TCP	51
 Profilo dispositivo.....	51
Disclaimer Legale – Limitazione di Responsabilità	52

Prefazione

Introduzione al percorso di onboarding

L'onboarding rappresenta il primo contatto strutturato tra l'utente e l'ecosistema DomoManager. Non è un semplice insieme di istruzioni operative, ma un vero e proprio **percorso di orientamento**, progettato per accompagnare installatori, tecnici e sviluppatori nella comprensione dei principi fondamentali che governano la piattaforma. DomoManager nasce come risposta a un'esigenza concreta: gestire sistemi domotici complessi con un approccio **modulare, scalabile e completamente parametrico**, eliminando rigidità, ridondanze e dipendenze non necessarie. Questo documento introduce la filosofia che guida il progetto e fornisce le basi concettuali per affrontare con consapevolezza i capitoli successivi.

L'obiettivo principale dell'onboarding è creare un terreno comune di conoscenza, affinché ogni figura coinvolta — dall'installatore base allo sviluppatore avanzato — possa muoversi all'interno del sistema con sicurezza, comprendendo non solo *come* configurare un modulo, ma soprattutto *perché* esso è stato progettato in quel modo. La piattaforma, infatti, non si limita a offrire strumenti: propone un **metodo**, una logica di lavoro che privilegia la chiarezza, la coerenza e la totale assenza di hardcoded values, in favore di un'architettura dinamica e adattabile.

La filosofia progettuale di DomoManager

DomoManager è costruito attorno a un principio cardine: **la configurazione è il cuore del sistema**. Ogni sensore, attuatore, zona, area, soglia o limite non è definito nel codice, ma in strutture parametriche esterne, pensate per essere leggibili, modificabili e versionabili. Questa scelta architetturale permette di:

- ridurre drasticamente la complessità del firmware;
- garantire una manutenzione più semplice e sicura;
- consentire aggiornamenti rapidi senza toccare la logica interna;
- favorire l'estensibilità del sistema nel tempo.

Il risultato è un ambiente in cui l'utente non deve "adattarsi" al software, ma può modellarlo sulle proprie esigenze. La piattaforma non impone un modello rigido: offre un **framework** che permette di costruire sistemi domotici di qualsiasi dimensione, mantenendo sempre un elevato livello di controllo e trasparenza.

Questa filosofia si riflette in ogni componente: dai motori di dominio (WeatherEngine, PowerSupervisorEngine, HVAC, Security) ai reader generici dei sensori, dai task operativi ai moduli di comunicazione. Tutto è pensato per essere **modulare, sostituibile, estendibile**.

Perché un onboarding strutturato è essenziale

La potenza di DomoManager risiede nella sua flessibilità. Tuttavia, una piattaforma così ricca richiede un percorso di ingresso chiaro, progressivo e ben organizzato. L'onboarding ha quindi tre obiettivi fondamentali:

Comprendere l'architettura

Prima di configurare un sistema, è necessario capire come i vari moduli interagiscono tra loro. L'onboarding introduce i concetti chiave:

- FrontendConfig come sorgente unica della verità;
- motori di dominio come interpreti dei dati;
- task operativi come orchestratori delle logiche;
- moduli di comunicazione come ponte verso il mondo esterno.

Acquisire un metodo di lavoro

DomoManager non è un insieme di funzioni isolate: è un ecosistema coerente. L'onboarding insegna a:

- definire correttamente la struttura del sistema;
- mappare sensori e attuatori in modo sicuro;
- impostare soglie e limiti dinamici;
- verificare il comportamento tramite log diagnostici.

Evitare errori tipici

Un sistema parametrico può essere potente, ma anche delicato. L'onboarding aiuta a prevenire:

- configurazioni incoerenti;
- mapping errati;
- soglie non allineate ai motori;
- dipendenze circolari;
- comportamenti inattesi dovuti a valori mancanti.

A chi è destinato questo documento

Il documento è pensato per tre categorie di utenti, ciascuna con esigenze e livelli di competenza differenti:

Installatore base: deve comprendere i concetti essenziali e saper configurare un impianto standard.

Installatore avanzato: deve padroneggiare logiche, parametri e personalizzazioni.

Sviluppatore: deve conoscere l'architettura interna per estendere il sistema con nuovi moduli.

L'onboarding è costruito in modo progressivo: parte dai concetti generali e arriva gradualmente ai dettagli tecnici, permettendo a ciascun profilo di trovare il proprio livello di profondità.

Conclusione della prefazione

DomoManager non è solo un software: è un metodo, un ecosistema, una filosofia progettuale che mette al centro la configurazione, la modularità e la trasparenza. Questa prefazione introduce i principi fondamentali che guidano la piattaforma e prepara il lettore a un percorso di apprendimento strutturato, chiaro e progressivo.

L'onboarding è il primo passo per entrare nell'ecosistema DomoManager. Non è un manuale tecnico, ma un percorso guidato che permette a installatori, tecnici e sviluppatori di:

- capire l'architettura reale del sistema
- configurare correttamente un impianto Opta + Waveshare + Modbus
- evitare errori tipici
- acquisire un metodo di lavoro coerente con la filosofia del framework

DomoManager è un sistema **completamente dichiarativo**: nessuna logica è scritta nel firmware, tutto è definito in:

- **FrontendConfig**
- **DMSSetup.hpp**
- **AUTOMATION_JSON**

Questo documento ti accompagna passo per passo nella comprensione e nella messa in servizio di un impianto reale.

Architettura Generale del Sistema

Filosofia generale del sistema

La nuova architettura di DomoManager è costruita su tre principi:

- **Tutto è configurazione**, niente è hardcoded
- **Ogni modulo è indipendente**, orchestrato dal Frontend
- **Il ciclo principale non blocca mai**, grazie a task pesati e rate-limited

Il sistema è diviso in tre livelli:

1. **Main** → carica la configurazione completa (FrontendConfig)
2. **FrontendEngine** → orchestra i moduli (HVAC, Weather, Power, Security, MQTT...)
3. **DomoManager** → gestisce buffer, dispositivi, Modbus, automazioni, watchdog

Questo è il flusso reale del firmware.

MAIN - Punto di ingresso del sistema

📌 File: DomoManager.ino

Il main:

- inizializza hardware
- costruisce FrontendConfig
- avvia FrontendEngine

Esempio reale:

```
FrontendConfig cfg;  
DomoManagerFrontendEngine::Setup(mainConfig);
```

✓ Cosa significa in pratica

Il main **non contiene logica**. Tutta la logica è nei moduli e tutta la configurazione è in mainConfig.

FRONTEND ENGINE – Orchestratore dei moduli

📌 File: DMFrontend.hpp

Il Frontend è il **direttore d'orchestra**.

Cosa fa:

- Inizializza rete, Modbus TCP, MQTT, IOT
- Costruisce DomoManager
- Inizializza i moduli funzionali:
 - SecuritySensorEngine
 - HVACEngine
 - WeatherEngine

- PowerEngine
- PowerSupervisorEngine
- Medie dinamiche
- Watch (diagnostica)
- Registra configurazioni dinamiche
- Esegue il ciclo deterministico (TaskEngine)

Cosa NON fa:

- Non interpreta sensori
- Non gestisce automazioni
- Non gestisce Modbus
- Non gestisce errori dispositivi

Tutto questo è delegato a DomoManager.

DMSETUP – Inizializzazione hardware e buffer

📁 File: DMSetup.hpp

È il **bootstrap hardware** del sistema.

Cosa contiene:

- Lista dispositivi (Modbus TCP → RTU)
- Lista aree del buffer
- Toggles
- Splits
- Routes
- Watchdog config
- Wired sensors config
- MQTT vars
- HVAC zones
- Weather config
- Power config
- Medie dinamiche
- Diagnostic config
- Watch areas

Punto chiave:

DMSetup non contiene logica. Contiene solo configurazione strutturata.

È la “mappa dell’impianto”.

DOMOMANAGER – Il cuore del sistema

📌 File: DM.hpp

È il modulo più importante dell’intero firmware.

Cosa fa:

- Gestisce Modbus (lettura/scrittura)
- Gestisce buffer (aree, virtuali, changed map)
- Gestisce dispositivi (DeviceManager)
- Gestisce automazioni (AutomationEngine)
- Gestisce scheduler asincrono
- Gestisce toggles, splits, routing
- Gestisce watchdog avanzato
- Gestisce diagnostica aree
- Gestisce cicli IP e reset automatici
- Gestisce medie e trend
- Gestisce RTC

Cosa NON fa:

- Non gestisce HVAC
- Non gestisce meteo
- Non gestisce power
- Non gestisce MQTT

Questi sono moduli del Frontend.

CICLO PRINCIPALE – Round-robin deterministico

Il ciclo è:

1. FrontendEngine.loop()
2. DomoManager.update()
3. Moduli funzionali
4. Scheduler asincrono
5. Automazioni dinamiche
6. Watchdog
7. Diagnostica

Il sistema è progettato per **non bloccare mai**.

DIAGNOSTICA – Watch, Watchdog, DeviceErrors

Il sistema integra:

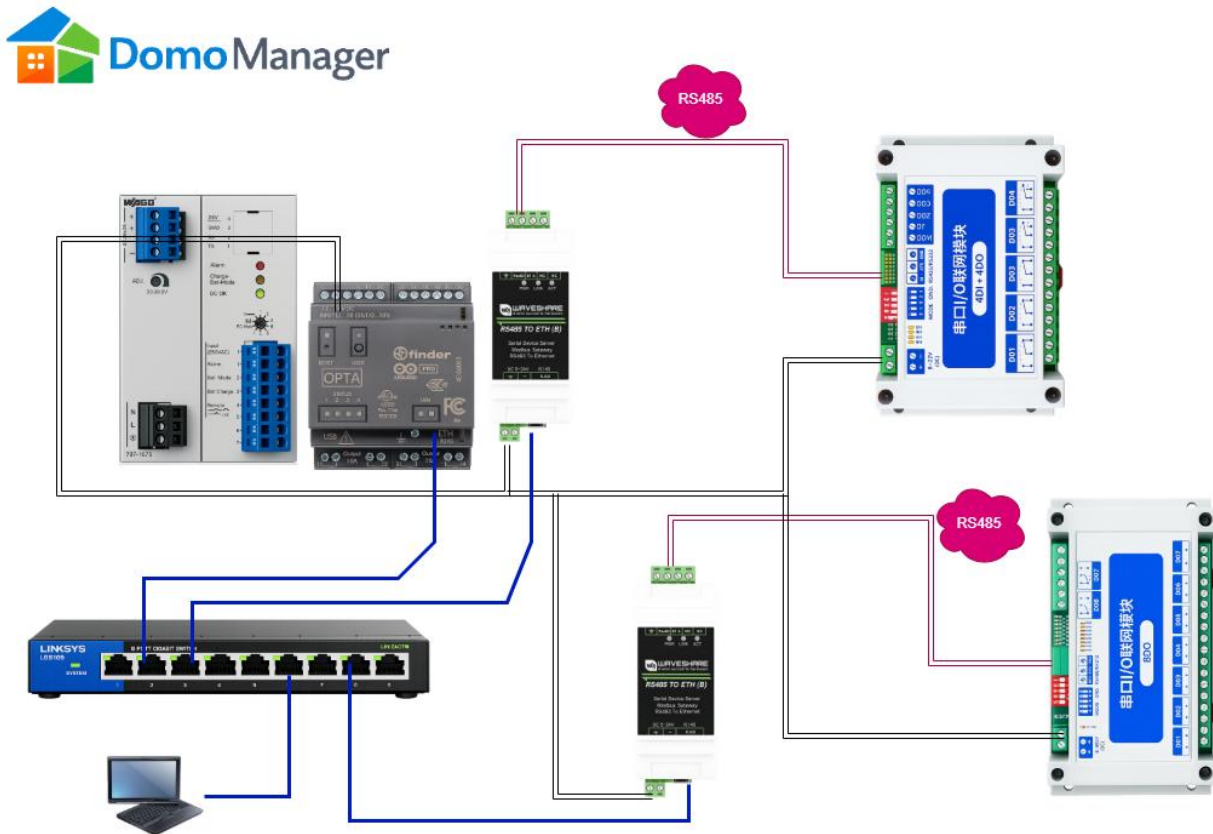
- Watchdog avanzato (EMA Q8.8)
- Spike detection
- Rate limiting
- DeviceHasErrors
- Aree monitorate (Watch)
- Logging dettagliato
- Reset IP automatico

Prerequisiti e Materiale Necessario

Manuale di Onboarding – Impianto Opta + (2) Waveshare + (2) I/O Modbus

Simuliamo un semplice impianto domotico con un Arduino OPTA che controlla attraverso la rete due piani di una casa. Su ogni piano, è installato un convertitore Ethernet\RS485 Waveshare dal quale parte un cavo schermato RS485 con protocollo Modbus che termina ad una scheda di IO. Nel prio piano troviamo una scheda 4DI+4DO Ebyte, al secondo piano troviamo una scheda 8DO sempre Ebyte

Ecco lo schema riassuntivo:



Obiettivo del capitolo

Fornire all'installatore tutto ciò che serve **prima** di iniziare:

- configurazione rete
- configurazione Waveshare
- cablaggio RS485
- preparazione firmware Opta
- preparazione JSON automazioni

Materiale hardware necessario

Unità di controllo

Componente	Q.tà	Note
Arduino Opta	1	Master Modbus TCP, orchestratore
Alimentatore 24V	1	Alimentazione Opta + dispositivi
Cavi Ethernet Cat5e/6	2-3	Opta ↔ Waveshare

Gateway RS485 → Ethernet

Modello	Q.tà	Note
Waveshare RS485→Ethernet	2	Ogni gateway gestisce un ramo RS485

Dispositivi Modbus RTU

Dispositivo	Q.tà	Funzione
Ebyte 4DI+4DO	1	4 ingressi + 4 uscite digitali
Ebyte 8DO	1	8 uscite digitali

Cablaggio RS485

- Cavo twistato schermato (4 x 0.5)
- Terminatore 120 Ω sull'ultimo dispositivo
- Morsettiere a vite

Materiale software necessario

Software	Scopo	Link
Arduino IDE V	Compilazione firmware Opta	arduino.cc/en/software
Waveshare Vircom V.5.68	Configurazione IP gateway	RS485 TO POE ETH (B) - Waveshare Wiki
Modbus Poll / Modbus Doctor	Test comunicazione Modbus	Modbus Doctor Client Modbus Gratuit
Editor JSON (VSCode consigliato)	Modifica automazioni	Visual Studio Code - The open source AI code editor Your home for multi-agent development

Schema di rete consigliato

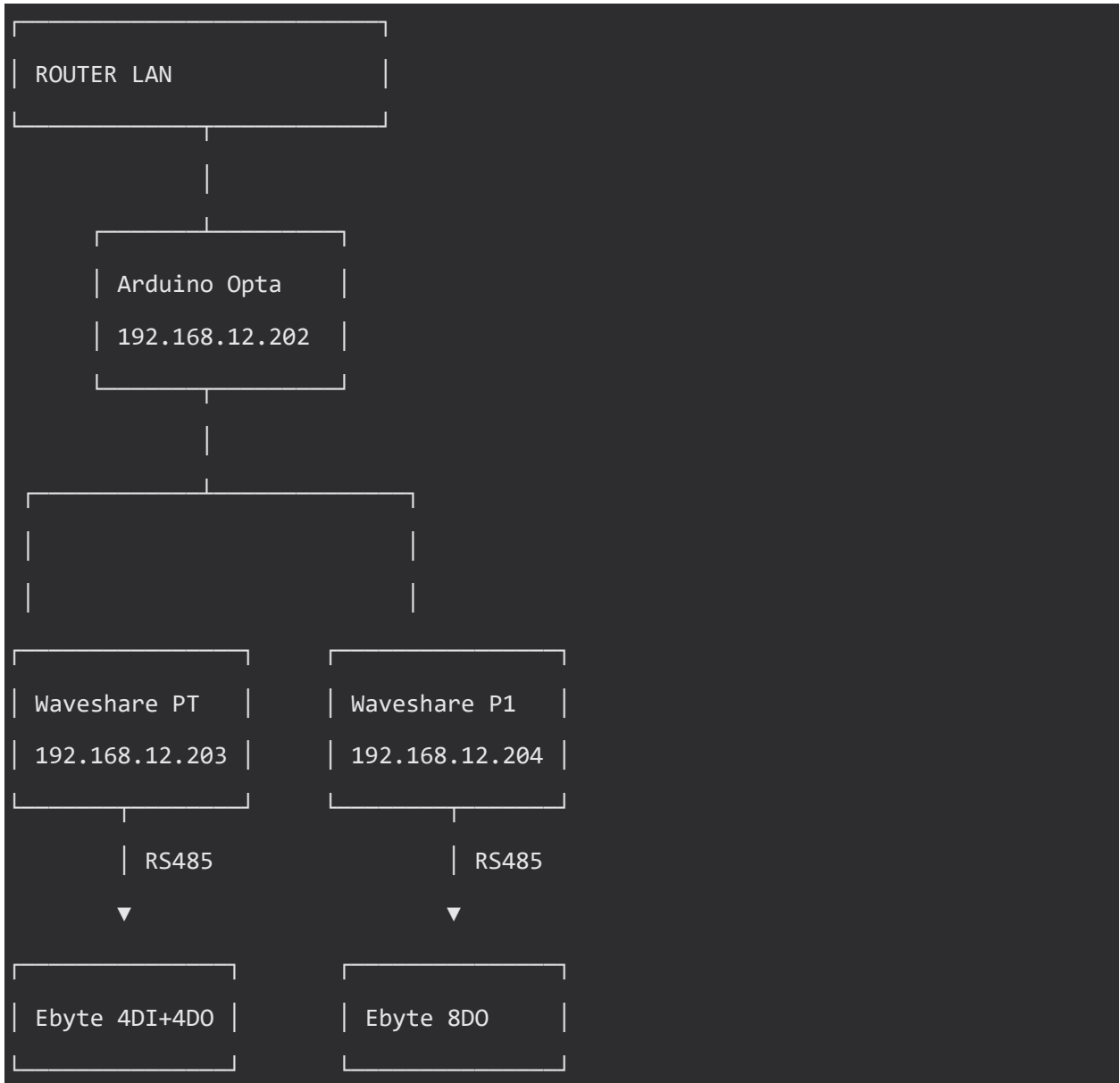


Tabella IP consigliati

Dispositivo	IP consigliato	Note
Opta	192.168.12.202	Master
Waveshare #1	192.168.12.203	Ramo RS485 piano terra
Waveshare #2	192.168.12.204	Ramo RS485 piano 1
PC	192.168.12.100	PC di programmazione

Tabella ID Modbus consigliati

Regola fondamentale: Gli ID devono essere **univoci per ogni gateway**, ma **possono ripetersi su gateway diversi**.

Gateway	Dispositivo	ID
Waveshare PT	Ebyte 4DI+4DO	2
Waveshare P1	Ebyte 8DO	7

Prerequisiti tecnici dell'installatore

L'installatore deve conoscere:

- Cablaggio RS485 (A/B, terminazione, polarità)
- Configurazione IP base
- Modbus RTU/TCP (concetti fondamentali)
- Uso di Arduino IDE
- Lettura di JSON

Checklist pre-installazione

```
[ ] Alimentazione 24V disponibile
[ ] Gateway Waveshare configurati con IP statico
[ ] ID Modbus impostati correttamente
[ ] Cablaggio RS485 completato e terminato
[ ] Firmware Opta compilato e caricato
[ ] File JSON automazioni pronto
[ ] PC con software necessario
```

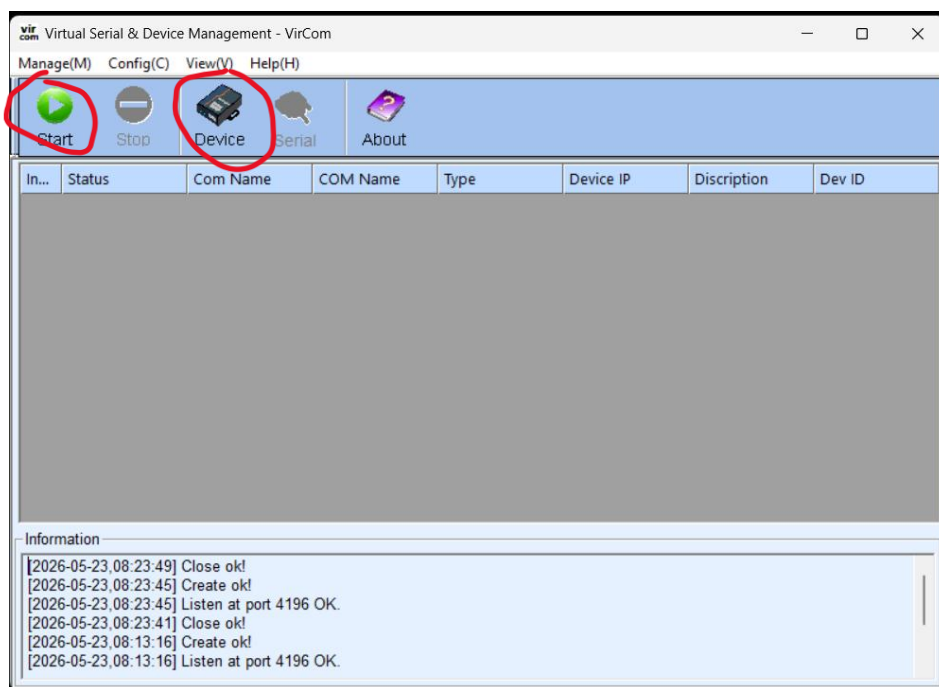
Configurazione – Opta + (2) Waveshare + (2) I/O Modbus

Configurazione fisica

Assicurarsi che il proprio PC sia configurato con l'indirizzo IP deciso in fase di pianificazione della rete (Schema di rete consigliato), e che lo switch ed i dispositivi Waveshare siano alimentati e interconnessi.

Assegnazione IP ai Waveshare

Eseguire il software VirCom, fare click sul pulsante Start e poi su Device

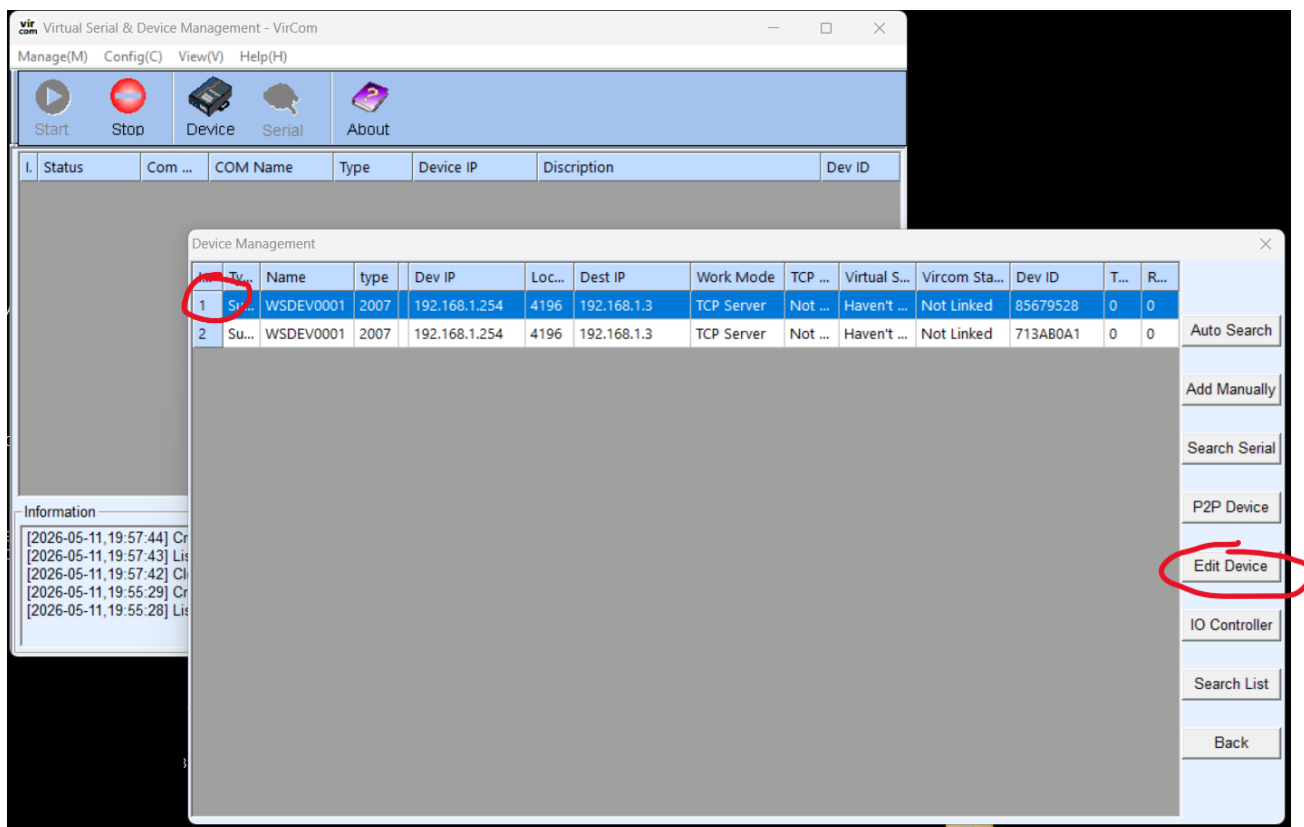


se la rete funziona correttamente dovreste vedere l'elenco dei dispositivi (2) disponibili, come in figura seguente.

Con i dispositivi nuovi si noterà l'indirizzo IP di default che è 192.168.1.254 per entrambe.

Selezionare il primo dispositivo della lista e fare click sul pulsante EDIT DEVICE.

ATTENZIONE: Per non confondersi è meglio accendere e configurare un dispositivo per volta, così da essere sicuri di dare l'indirizzo IP al gateway giusto (Piano Terra e Piano 1).



Apparirà la finestra di configurazione del device nella quale dovremmo inserire i parametri di configurazione.

Impostazioni

- Porta: **502**
- Protocollo di trasferimento: **Modbus TCP**
- Timeout: 1000–2000 ms
- RS485: 9600 / N / 8 / 1

Device Settings

Device Info

Virtual Serial: Not Use

Dev Type:

Dev Name: WS-PT

Dev ID: 2882713AB0A1

Firmware Ver: V1.523

Function of the device

☐ Web Download

☒ DNS System

☒ REAL_COM Protocol

☒ Modbus TCP To RTU

☒ Serial Commnad

☒ DHCP Support

☐ Storage Extend

☒ Multi-TCP Connection

Network

IP Mode: Static

IP Address: 192 . 168 . 12 . 203

Port: 502

Work Mode: TCP Server

Net Mask: 255 . 255 . 255 . 0

Gateway: 192 . 168 . 12 . 1

Dest. IP/Domain: 192.168.1.3 Local IP

Dest. Port: 4196

Serial

Baud Rate: 9600

Data Bits: 8

Parity: None

Stop Bits: 1

Flow Control: None

Advanced Settings

DNS Server IP: 8 . 8 . 4 . 4

Dest. Mode: Dynamic

Transfer Protocol: Modbus_TCP Protocol

Keep Alive Time: 60 (s)

Reconnet Time: 12 (s)

Http Port: 80

UDP Group IP: 230 . 90 . 76 . 1

☐ Register Pkt: ☐ ASCII

☐ Restart for no data every 300 Sec.

☐ Enable send parameter every 5 Min.

More Advaced Settings...

Framing Rule

Max Frame Length: 1300 (Byte)

Max Interval(Smaller will better) 3 (Ms)

Get Default Save As Default Load Default Modify Key Firmware/Config Restart Dev Modify Setting Cancel

Dopo avere inserito i parametri corretti, basterà fare click su MODIFY SETTINGS, quindi riavviare il Gateway levando alimentazione per 20 secondi.

Rialimentato il Gateway, possiamo provare ad eseguire un PING per vedere se l'indirizzo risponde.

Eseguiamo la stessa cosa sul Gateway Waveshare del piano 1, avendo cura di rispettare l'indirizzo IP pianificato ed un nome diverso per il dispositivo.

Collegamento RS485 con l'IO remoto Ebyte 4DI+4DO

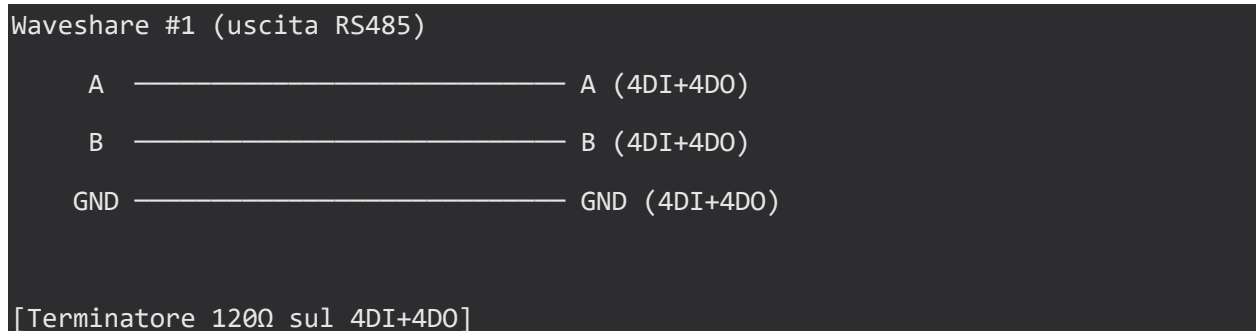
Cablaggio Waveshare <->Ebyte

- A → A
- B → B
- GND comune consigliato
- Topologia **a linea**, non a stella
- Terminatore **120 Ω** SOLO sull'ultimo dispositivo
- Lunghezza massima consigliata: **500 m**
- Cavo twistato schermato, comunque che non passi assieme a linee di alimentazione con tensioni pari o superiori a 50Vca.

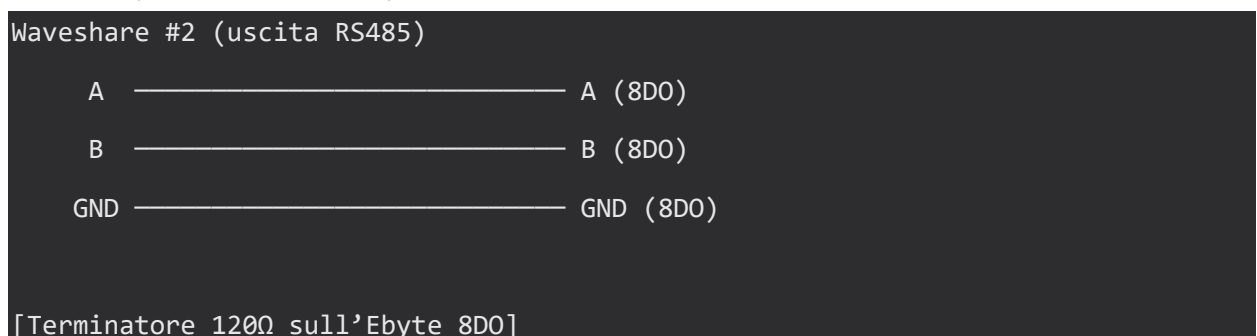
Topologia

Ogni Waveshare gestisce **un ramo RS485 indipendente**.

Schema RS485 per Waveshare PT (Ebyte 4DI+4DO)

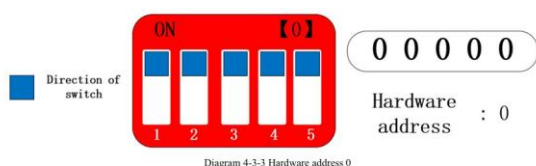


Schema RS485 per Waveshare P1 (Ebyte 8DO)

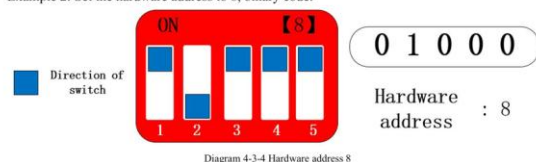


Impostazione indirizzi Modbus (ID)

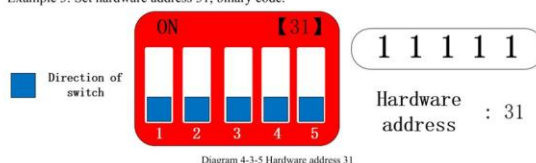
Attenzione: I dispositivi Ebyte hanno una configurazione particolare degli indirizzi: all'indirizzo selezionato sul dip switch, va aggiunto 1 (vedi manuale del prodotto, capitolo 4.3 Device Address). Si riporta di seguito un estratto del manuale che riporta i dip switch in 3 posizioni di esempio per gli indirizzi 0, 31 e 8. In realtà, questi indirizzi verranno interrogati sugli indirizzi 0+1=1, 31+1=32, 8+1=9.



Example 2: Set the hardware address to 8, binary code.

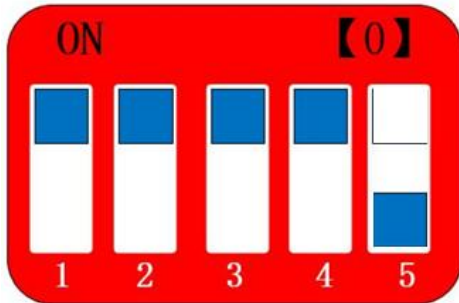


Example 3: Set hardware address 31, binary code.

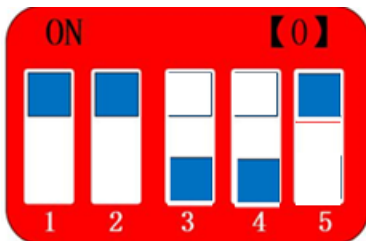


Quindi, impostando il dip switch sull'indirizzo 1, in realtà l'interrogazione Modbus avverrà sul 2, come deciso nella nostra pianificazione. Spostare quindi i dip switch di entrambe gli IO remoti con lo switch 5 a OFF e tutti gli altri ad ON come in figura.

Questo dovrà essere il DIP Switch del dispositivo Ebyte 4DI+4DO, ID=2



Mentre questo dovrà essere il DIP Switch del dispositivo Ebyte 8DO, ID=7

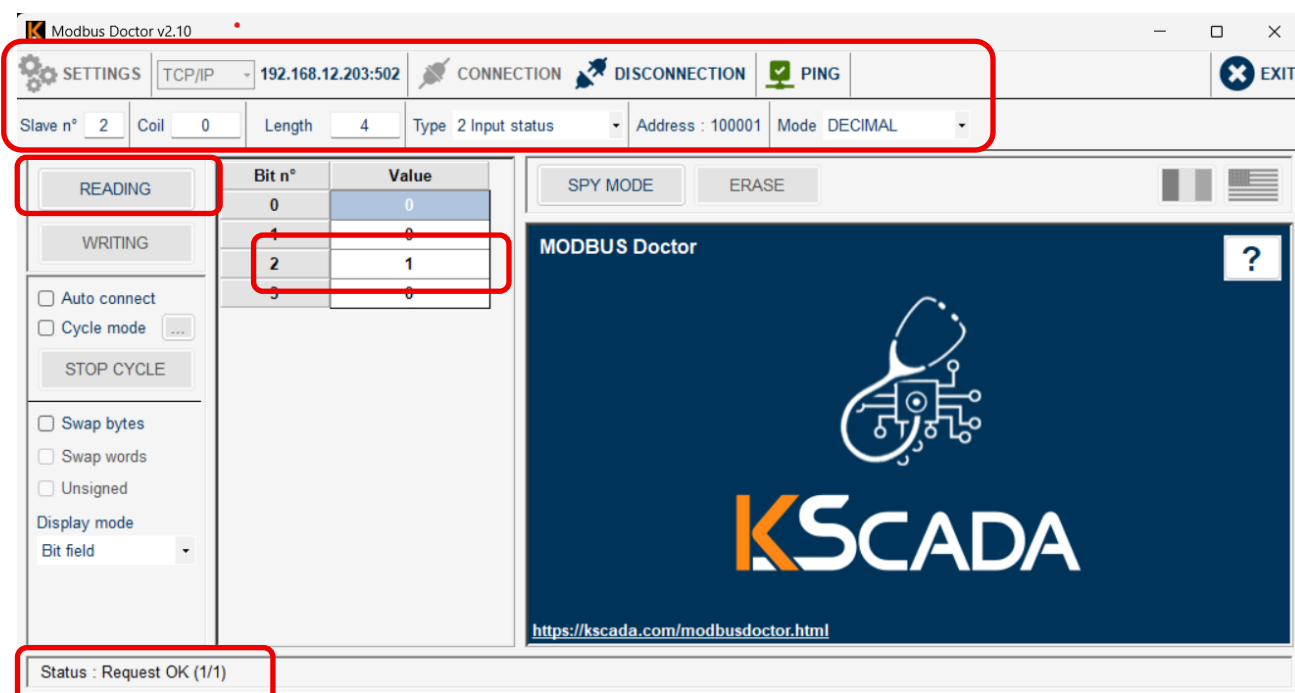


Lo spostamento degli switch di indirizzo, necessitano dello spegnimento e riaccensione dei dispositivi per reinizializzare il nuovo indirizzo.

Test finali

A questo punto, lanciamo l'applicazione di test della comunicazione Modbus Doctor, e testiamo il ramo di impianto di piano terra, quindi 192.168.12.203 e dispositivo Modbus 2, il 4DI+4DO legge gli ingressi sui primi 4 registri modbus (Input register) a partire dall'indirizzo 0, settiamo quindi il software di test con tutti i parametri, premiamo il pulsante READING ed il software effettuerà una lettura degli input digitali, dando il risultato in basso a sinistra (Request Ok, significa che la lettura è avvenuta con successo).

A questo punto se cablo un pulsante sul dispositivo 4DI+4DO e lo premo, forzando una lettura sul software, vedrò il valore del corrispondente bit andare a 1.

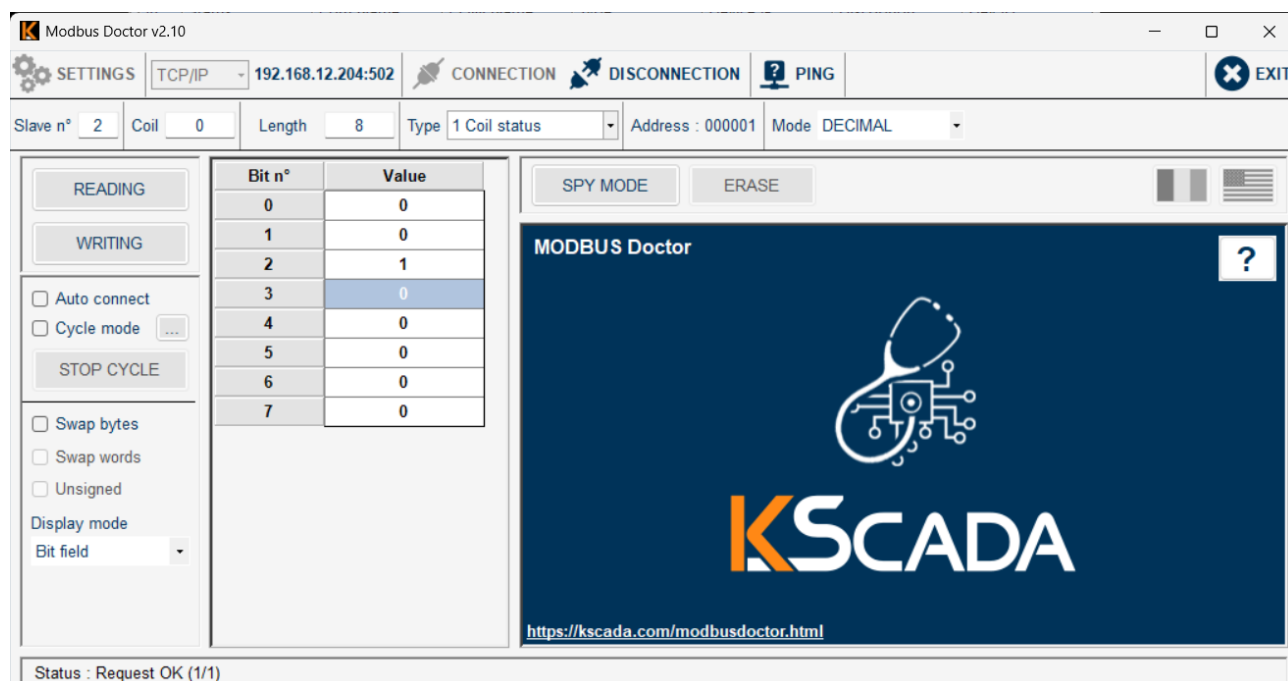


Ora passiamo al test della connessione del piano 1.

Configuriamo sul software di test l'indirizzo IP del gateway e l'indirizzo del dispositivo modbus, 7, premendo su CONNECTION il software si connette al gateway.

Essendo connesso ad un dispositivo a sole uscite (EByte 8DO), configuriamo la scrittura dei registri Coils, 8, partendo da 0.

Modifichiamo ora il valore del bit 2 ad 1, e premiamo sul pulsante WRITING.



Se tutto è stato fatto correttamente, avremo il messaggio in basso a sinistra Request OK ed il relay selezionato sull'IO remoto si ecciterà.

Ecco riassunto in foto, il nostro banco di prova per questo test.



Configurazione logica (OPTA)

Terminati i cablaggi ed i test della parte hardware, passiamo alla configurazione del controller OPTA.

Configurazione dispositivi in DomoManager (DMSetup.hpp)

La mappatura dispositivi fisici, si trova nell'array **mainDevicesConfig**: è definita nel file **DMSetup.hpp** tramite:

```
static const DomoManagerConfig::Devices mainDevicesConfig = {
    {
        { "Scheda 4DI+4DO - Scala", WaveSharePT_Addr, 2, "MA01_XACX0440",
          { AREA_DI_BAGNO, AREA_DI2, AREA_DI3, AREA_DI4, AREA_DO1, AREA_DO2,
            AREA_DO3, AREA_DO_ALIM_BAGNO },
          3, High
        },
        { "Scheda 8DO - Fondo P1", WaveShareP1_Addr, 7, "MA01_AXCX0080",
          { AREA_DO1A, AREA_DO2A, AREA_DO3A, 21, 22, AREA_DO_STRIP1_BAGNO,
            AREA_DO_STRIP2_BAGNO, 25 }, 3, Low
          }
    }
};
```

Ogni dispositivo contiene:

- Nome
- IP del gateway (Waveshare o altro dispositivo), nell'esempio gli IP sono definiti in due variabili

```
arduino::IPAddress WaveShareP1_Addr=IPAddress(192, 168, 12, 204);
arduino::IPAddress WaveSharePT_Addr=IPAddress(192, 168, 12, 203);
```

- ID Modbus RTU (tra 1 e 127)
- Profilo (es. "N4DIH32", "MA01_AXCX0080") dipende dal device
- Lista aree (contiguee, univoche, di quantità dipendente dal device prescelto – **VEDERE MANUALE 5 LIBRERIE DISPOSITIVI**)
- Retry (3 se non meglio specificato)
- Priorità (High, Med, Low) in base a come deve essere aggiornato es: gestione pulsanti High, Low per sensori di bassa importanza.

✚ **DomoManager carica automaticamente questa lista** durante il setup.

Definizione aree del buffer

Il buffer è l'area di scambio di tutte le informazioni in Domomanager. Principalmente, è necessario definire tante aree quanti dispositivi di I/O fisici sono stati definiti precedentemente in **mainDevicesConfig**, in modo tale da **dichiarare per ogni area definita negli I/O la funzione**.

Centrando il discorso su questo esempio di utilizzo, la Scheda 4DI+4DO – Scala, ha 8 aree, 4 di input e 4 di output, e queste sono state definite (come numero), nel device; qui, ogni numero viene ripetuto per definirne meglio le caratteristiche.

Le aree sono definite tramite **mainAreasConfig**.

```
static const DomoManagerBufferEngine::AreasConfig mainAreasConfig = {
{
    // **** 4 DI EbYTE ID=2

    { AREA_DI_BAGNO, 0, "Pulsante bagno", {false,false,false}, {-1,-1} },
    { AREA_DI2, 0, "Pulsante Apri Vasistas", {true,true,false}, {-1,-1} }, //
forwardArea = 57
    { AREA_DI3, 0, "Pulsante Chiudi vasistas", {true,true,false}, {-1,-1}
}, // -----A
    { AREA_DI4, 25, "Pulsante scala", {true,true,false}, {-1,-1} }, //
forwardArea = 59

    // 4 DO EbYTE

    { AREA_DO1, 0, "", {false,false,false}, {-1,-1} },
    { AREA_DO2, 0, "", {false,false,false}, {-1,-1} },
    { AREA_DO3, 0, "", {true,true,false}, {-1,-1} },
    { AREA_DO ALIM_BAGNO, 0, "Alimentatore LED bagno", {false,false,false}, {-
1,-1} },

    // **** 8 DO EbYTE ID=7

    { AREA_DO1A, 0, "Inv. vasistas sx.", {false,false,false}, {-1,-1} },
    { AREA_DO2A, 0, "Ali. vasistas sx.", {false,false,false}, {-1,-1} },
    { AREA_DO3A, 0, "", {false,false,false}, {-1,-1} },
    { 21, 0, "", {false,false,false}, {-1,-1} },
    { 22, 0, "", {false,false,false}, {-1,-1} },
    { AREA_DO_STRIP1_BAGNO, 0, "Led strip 1", {false,false,false}, {-1,-1} },
    { AREA_DO_STRIP2_BAGNO, 0, "Led strip 2", {false,false,false}, {-1,-1} },
    { 25, 0, "Luce scala", {false,false,false}, {-1,-1} },
```

```
{ 26, 0, "Virtual", {false,false,false}, {-1,-1} }
}];
```

Ogni area definisce:

- Numero area
- Forward area (per comandare direttamente un'altra area)
- Etichetta
- Flag (scrittura verso pannello, lettura da pannello, logica negata)
- Soglie (low/high) se il valore è analogico

✚ Le aree non mappate da dispositivi vengono marcate automaticamente come **virtuali**.

Configurazione Toggles e Splits

Toggles

Definiti in mainTogglesConfig:

```
static const DomoManagerToggleEngine::TogglesConfig mainTogglesConfig = {
    { { 54, { BAGNO_IN_P2 } }, { 90, { 82, 55 } }, ... }
};
```

Praticamente il toggle trasforma un ingresso momentaneo comandato da un pulsante, in una sequenza on\off ad ogni pressione dell'ingresso.

Splits

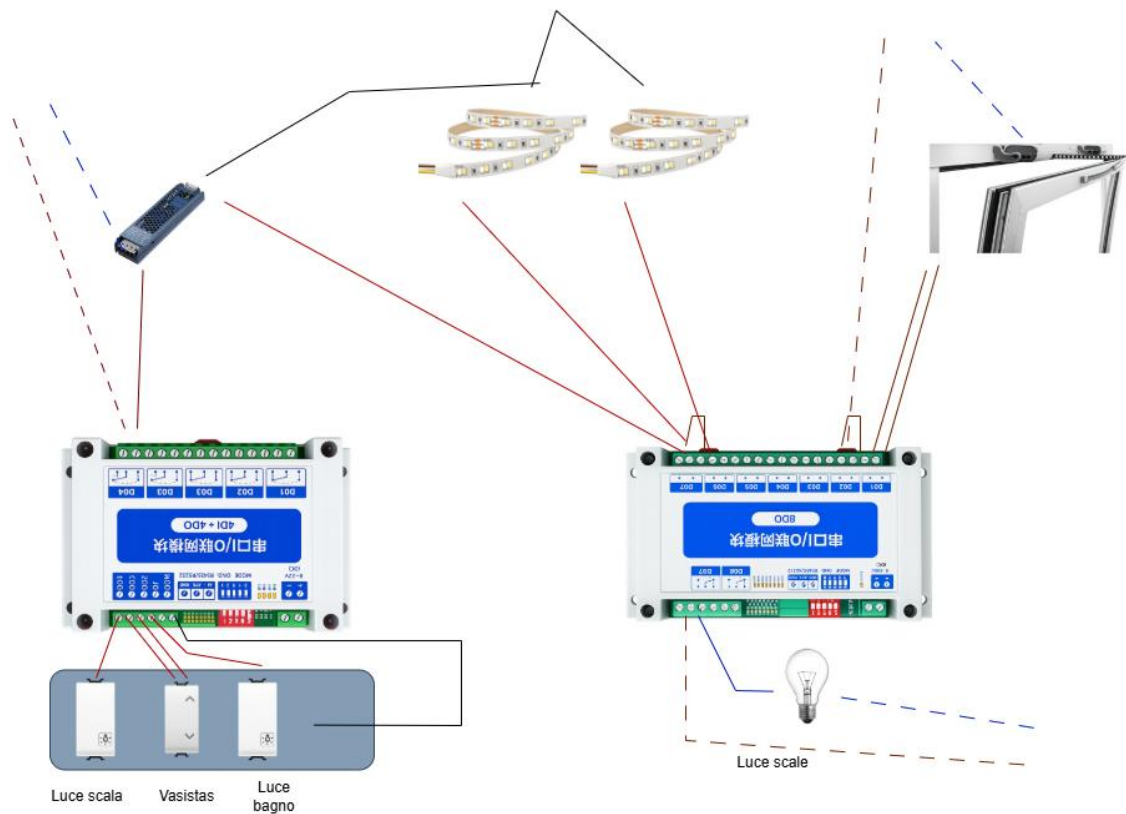
Definiti in mainSplitsConfig:

```
static const DomoManagerSplitEngine::SplitsConfig mainSplitsConfig = {
    { { 78, {114,115}, 30000 }, ... }
};
```

Gli split permettono di comandare piu uscite contemporaneamente al cambio di un ingresso.

Analisi della configurazione di esempio

L'impianto di test del presente documento, presenta i seguenti cablaggi (schema di principio) attestati sugli I/O remoti:



Ebyte 4DI+4DO:

- Ingresso 1: Interruttore bagno, AREA_DI_BAGNO
- Ingresso 2: Interruttore apertura Vasistas, AREA_DI2
- Ingresso 3 Interruttore chiusura Vasistas, AREA_DI3
- Ingresso 4: Interruttore bagno, AREA_DI4
- Uscita 4: Alimentatore LED bagno, AREA_DO ALIM_BAGNO

Ebyte 8DO:

- Uscita 1A: comandi apri\chiudi Vasistas, con comune del relay ponticellato sul relay uscita 2, AREA_DO1A
- Uscita 2A: on/off Vasistas, AREA_DO2A
- Uscite 6A e 7A, alimentazioni Strip LED provenienti da alimentatore LED , AREA_DO_STRIP1_BAGNO e AREA_DO_STRIP2_BAGNO
- Uscita 8A: Luce scala, area 25 (senza DEFINE)

Funzionamento:

- **Pulsante luce scala** AREA_DI_BAGNO, accende luce scale area 25 (e rimane accesa finchè il pulsante viene tenuto premuto).
- **Pulsante luce bagno** AREA_DI4, con toggle, accede Alimentatore AREA_DO ALIM_BAGNO e 2 strip led, AREA_DO_STRIP1_BAGNO e AREA_DO_STRIP2_BAGNO
- **Il doppio pulsante** AREA_DI2, AREA_DI3, comanda le uscite AREA_DO1A o AREA_DO1A e AREA_DO2A rispettivamente per alimentare il vasistas (chiudi) o alimentare il vasistas ed invertire il senso di marcia (apri), tutto con un limite di 30 secondi

File DMSetup.hpp

Si inizia quindi a battezzare le Aree di buffer, a partire dalla nr. 10 in avanti (da 0 a 9 sono riservate); attraverso le DEFINE_AREA si associano nomi agli indirizzi del buffer.

```
// ***** IO AREA IDS*****  
  
// 0..9 RESERVED  
  
DEFINE_AREA(AREA_DI_BAGNO, 10)  
DEFINE_AREA(AREA_DI2, 11)  
DEFINE_AREA(AREA_DI3, 12)  
DEFINE_AREA(AREA_DI4, 13)  
  
DEFINE_AREA(AREA_DO1, 14)  
DEFINE_AREA(AREA_DO2, 15)  
DEFINE_AREA(AREA_DO3, 16)  
DEFINE_AREA(AREA_DO ALIM_BAGNO, 17)  
  
DEFINE_AREA(AREA_DO1A, 18)  
DEFINE_AREA(AREA_DO2A, 19)  
DEFINE_AREA(AREA_DO3A, 20)  
  
DEFINE_AREA(AREA_DO_STRIP1_BAGNO, 23)  
DEFINE_AREA(AREA_DO_STRIP2_BAGNO, 24)
```

In questo esempio abbiamo a posta non battezzato tutte le aree, in modo tale da fare vedere che è possibile riferirsi alle aree di buffer o mediante il loro indice o mediante la definizione del nome.

Le aree vengono agganciate ai device fisici, lasciando l'area 26 virtuale, ovvero non legata ad alcun device.

```

static const DomoManagerBufferEngine::AreasConfig mainAreasConfig = {
{
    // **** 4 DI EbYTE ID=2

    { AREA_DI_BAGNO, 0, "Pulsante bagno", {false,false,false}, {-1,-1} },
    { AREA_DI2, 0, "Pulsante Apri Vasistas", {true,true,false}, {-1,-1} }, //
forwardArea = 57
    { AREA_DI3, 0, "Pulsante Chiudi vasistas", {true,true,false}, {-1,-1}
}, // -----A
    { AREA_DI4, 25, "Pulsante scala", {true,true,false}, {-1,-1} }, //
forwardArea = 59

    // 4 DO EbYTE

    { AREA_DO1, 0, "", {false,false,false}, {-1,-1} },
    { AREA_DO2, 0, "", {false,false,false}, {-1,-1} },
    { AREA_DO3, 0, "", {true,true,false}, {-1,-1} },
    { AREA_DO_ALIM_BAGNO, 0, "Alimentatore LED bagno", {false,false,false}, {-
1,-1} },

    // **** 8 DO EbYTE ID=7 - Vasistas e scuri
    { AREA_DO1A, 0, "Inv. vasistas sx.", {false,false,false}, {-1,-1} },
    { AREA_DO2A, 0, "Ali. vasistas sx.", {false,false,false}, {-1,-1} },
    { AREA_DO3A, 0, "", {false,false,false}, {-1,-1} },
    { 21, 0, "", {false,false,false}, {-1,-1} },
    { 22, 0, "", {false,false,false}, {-1,-1} },
    { AREA_DO_STRIP1_BAGNO, 0, "Led strip 1", {false,false,false}, {-1,-1} },
    { AREA_DO_STRIP2_BAGNO, 0, "Led strip 2", {false,false,false}, {-1,-1} },
    { 25, 0, "Luce scala", {false,false,false}, {-1,-1} },

    { 26, 0, "Virtual", {false,false,false}, {-1,-1} }

}};

```

da notare che l'area AREA_DI4, scrive direttamente sull'area 25:

```

{ AREA_DI4, 25, "Pulsante scala",.....
.....
{ 25, 0, "Luce scala",.....

```

In modo tale che alla pressione del pulsante si ecciti direttamente l'uscita desiderata fino al rilascio del pulsante.

Poi definiamo una route, che si lega all'area AREA_DI_BAGNO (pulsante luce bagno) e che alla pressione comanda 3 uscite contemporaneamente:

```
static const DomoManagerRouteEngine::RoutesConfig mainRoutesConfig = {
    {
        {
            "Bagno - Interruttore luce",
            AREA_DI_BAGNO,
            {
                // Caso: value == 1
                {
                    1,
                    {
                        { AREA_DO_ALIM_BAGNO, 1 },
                        { AREA_DO_STRIP1_BAGNO, 1 },
                        { AREA_DO_STRIP2_BAGNO, 1 },
                        { 26, 150 }
                    }
                },
                // Caso: value == 0
                {
                    0,
                    {
                        { AREA_DO_ALIM_BAGNO, 0 },
                        { AREA_DO_STRIP1_BAGNO, 0 },
                        { AREA_DO_STRIP2_BAGNO, 0 },
                        { 26, 0 }
                    }
                }
            }
        }
    }
};
```

per puro esercizio, oltre alle 3 uscite, comandiamo anche l'area 26 (virtuale) scrivendo il valore 150 all'accensione, e 0 allo spegnimento.

Inoltre, dato che il pulsante deve comportarsi come un interruttore, aggiungiamo un toggle:

```
static const DomoManagerToggleEngine::TogglesConfig mainTogglesConfig = {
    {
        { AREA_DI_BAGNO, { } }
    }
};
```

in ultimo configuriamo le uscite per comandare il Vasistas, attraverso lo split, dove AREA_DI2 comanda 2 uscite per massimo 30 secondi (30000 millisecondi) e AREA_DI3 comanda solo una uscita sempre per massimo 30 secondi, poi leva alimentazione.

```
static const DomoManagerSplitEngine::SplitsConfig mainSplitsConfig = {
    {
        { AREA_DI2, {AREA_D01A, AREA_D02A}, 30000 },
        { AREA_DI3, {AREA_D01A}, 30000 }
    }
};
```

Dove vanno a finire queste dichiarazioni? Nella funzione seguente, nelle parti

- // --- Devices & Areas ---
- // --- Engines ---

```
// ***** LOADER *****
DomoManagerConfig makeDomoConfig() {
    DomoManagerConfig cfg; // usa tutti i default della struct

    cfg.hmi.enabled = false; // opzionale, è già default
    cfg.hmi.port = 502;      // default
    cfg.hmi.pollingMs = 500; // default

    // --- Modbus ---
    cfg.modbusRTU.port = 502;

    // --- Watchdog ---
```

```

    cfg.watchdog = wdConfig;

    // --- Devices & Areas ---
    cfg.devices = mainDevicesConfig;
    cfg.areas   = mainAreasConfig;

    // --- Engines ---
    cfg.routes  = mainRoutesConfig;
    cfg.toggles = mainTogglesConfig;
    cfg.splits  = mainSplitsConfig;

    return cfg;
}

static DomoManagerConfig domoConfig = makeDomoConfig();

```

Sempre in questa funzione troviamo:

- i parametri di gestione del pannello HMI che tratteremo in seguito (cfg.hmi)
- la porta di connessione per il Modbus RTU (cfg.modbusRTU che è al valore di default 502)
- i parametri di watchdog che di norma non vanno modificati

tutti questi settaggi, convogliano nella variabile **domoConfig**.

File DomoManager.ino

domoConfig viene convogliato nella struttura di configurazione del frontend, **mainConfig**, che raccoglie oltre ai parametri di DomoManager, anche i parametri base di sistema ed i parametri dei moduli di Frontend che vedremo più avanti.

```

static FrontendConfig mainConfig = [](){
    FrontendConfig c;

    c.pins.userButton = BTN_USER;
    c.pins.leds = LedController::LedPins(
        LED_D0, //RS485 Read
        LED_D1, //RS485 Write
        LED_D2, // HMI Read/Write
        LED_D3  // Devices in error
    );
}

```

```

// --- NETWORK ---

c.net.mac      = {0x90, 0xA2, 0xDA, 0x0E, 0x94, 0xB5};
c.net.ip       = IPAddress(192, 168, 12, 202);
c.net.gateway  = IPAddress(192, 168, 12, 1);
c.net.subnet   = IPAddress(255, 255, 255, 0);


// --- MODBUS TCP to RTU ---

c.modbus.timeoutMs = 250; //200ms non scendere mai sotto questo valore
15.05.2026, valore ok=250 cautelativo


// --- DomoManager ---

c.domoManager=domoConfig;


// --- DIAGNOSTICS ---

c.diagnostic=diagnosticParams;


// --- WATCH ---

c.watch.enabled = false;

c.watch.aree = watchAreas;

c.watch.count = sizeof(watchAreas) / sizeof(watchAreas[0]);


return c;

}();

```

La struttura contiene la configurazione del pulsante di OPTA e dei suoi LED (se si utilizza un altro controller la configurazione va variata), la configurazione di rete, che deve essere omogenea a quella del PC di programmazione, dell'HMI se viene usato, dei convertitori Ethernet-RS485 per il modbus RTU.

Andando avanti, si trova il timeout dei dispositivi su rete RS485 Modbus

```
c.modbus.timeoutMs = 250;
```

di norma il parametro non va cambiato, è un buon compromesso tra prestazioni e sicurezza.

Andando avanti, troviamo l'aggancio al setup visto prima di DomoManager, i parametri diagnostici che rispondono alla struttura sottostante, **diagnosticParams**

```
static const DiagnosticConfig diagnosticParams = {  
  
    .reportAutomationEngine    = false,  
    .reportScheduler           = false,  
    .reportSplitAnalysis       = false,  
    .reportMissingBufferDefs   = false,  
    .reportDeviceAreas         = false,  
    .reportIneffectiveRules    = false,  
    .reportDeviceErrors        = true,  
    .reportWatchdogs           = false,  
    .reportDeviceProfiles      = false,  
    .reportRTC                 = true,  
    .reportHotStandby          = true,  
    .reportVirtualAreas        = false,  
    .reportUnusedBufferAreas   = false,  
    .reportNeverInitialized    = false,  
    .reportMultipleInitialized = true,  
    .reportAutomationConfig    = false,  
    .reportLogBuffer           = true  
  
};
```

che determina quali informazioni visualizzare alla pressione dal pulsante utente presente su OPTA, nella finestra **Serial Monitor** dell'IDE Arduino.

In ultimo, la configurazione degli autoWatch, ovvero della visualizzazione automatica del valore di una o più variabili, al variare di almeno un valore. Nell'esempio non è abilitato, ma settando

```
c.watch.enabled = true;
```

ed inserendo nell'array l'indice della\ delle aree da visualizzare (o il nome della define), al variare di almeno un area dell'elenco, il serial monitor dell'IDE di Arduino viene fermato e visualizzato il valore delle variabili elencate, fino alla pressione del pulsante utente su OPTA.

```
static const int watchAreas[] = { 26 };
```

In questo caso, abbiamo inserito l'area di esempio Virtuale, che viene scritta dalla routine di Route.

File DMFrontend.hpp

Alla partenza del software, questa configurazione viene passata al frontend che gestisce le classi con i servizi di Engine animate dal cuore DomoManager che scambio dati tra le Aree di buffer e gli IO remoti. Se nell' .ino sono configurati uno o più engines, questi vengono fatti girare in task separati ma tutti connessi mediante le aree di buffer.

E' in questo file, che eventuali componenti di engine dell'utente devono essere dichiarati ed integrati per l'utilizzo.

Avvio di DomoManager

Terminate le configurazioni hardware e software, siamo pronti ad avviare l'OPTA. Aprire l'IDE di Arduino e trasferire il software su OPTA (seguire le guide Arduino); Se la configurazione del file DMSetup.hpp e/o DomoManager.ino fosse errata, sul serial monitor vedreste una indicazione come quella riportata sotto.

```
Output Serial Monitor X
Message (Enter to send message to 'Opta' on 'COM3')

10:02:39.895 -> [2448 ms] [i INFO] [DomoManagerFrontendEngine::SetupEthernet] Ethernet interface started
10:02:39.895 -> [2450 ms] [i INFO] [DomoManagerFrontendEngine::SetupEthernet] My IP address: 192.168.12.202
10:02:39.895 -> [2451 ms] [i INFO] [DomoManagerFrontendEngine::Setup] Modbus TCP to RTU timeout set 250 mSec.
10:02:39.935 -> [2455 ms] [X ERROR] [Config::validateRoutes] Route 'Bagno - Interruttore luce': triggerArea 22 non valida
10:02:39.935 -> [2456 ms] [X ERROR] [Config::validateRoutes] Route 'Bagno - Interruttore luce': targetArea 23 non valida
10:02:39.935 -> [2457 ms] [X ERROR] [Config::validateRoutes] Route 'Bagno - Interruttore luce': targetArea 24 non valida
10:02:39.935 -> [2458 ms] [X ERROR] [Config::validateRoutes] Route 'Bagno - Interruttore luce': targetArea 25 non valida
10:02:39.935 -> [2458 ms] [X ERROR] [Config::validateRoutes] Route 'Bagno - Interruttore luce': targetArea 23 non valida
10:02:39.935 -> [2459 ms] [X ERROR] [Config::validateRoutes] Route 'Bagno - Interruttore luce': targetArea 24 non valida
10:02:39.935 -> [2460 ms] [X ERROR] [Config::validateSplits] Split mainArea 78 non valida
10:02:39.935 -> [2461 ms] [X ERROR] [Config::validateSplits] Split outArea 114 non valida
10:02:39.935 -> [2461 ms] [X ERROR] [Config::validateSplits] Split outArea 115 non valida
10:02:39.935 -> [2462 ms] [X ERROR] [Config::validateSplits] Split mainArea 79 non valida
10:02:39.935 -> [2463 ms] [X ERROR] [Config::validateSplits] Split outArea 114 non valida
10:02:39.935 -> [2464 ms] [X ERROR] [Config::validateSplits] Split mainArea 80 non valida
10:02:39.935 -> [2465 ms] [X ERROR] [Config::validateSplits] Split outArea 113 non valida
10:02:39.935 -> [2465 ms] [X ERROR] [Config::validateSplits] Split mainArea 81 non valida
10:02:39.935 -> [2466 ms] [X ERROR] [Config::validateSplits] Split outArea 113 non valida
10:02:39.935 -> [2467 ms] [X ERROR] [Config::validateSplits] Split outArea 112 non valida
10:02:39.935 -> [2468 ms] [X ERROR] [Config::validateSplits] Split mainArea 86 non valida
10:02:39.935 -> [2468 ms] [X ERROR] [Config::validateSplits] Split outArea 116 non valida
10:02:39.935 -> [2469 ms] [X ERROR] [Config::validateSplits] Split outArea 117 non valida
10:02:39.935 -> [2470 ms] [X ERROR] [Config::validateSplits] Split mainArea 87 non valida
10:02:39.935 -> [2470 ms] [X ERROR] [Config::validateSplits] Split outArea 116 non valida
10:02:39.935 -> [2471 ms] [X ERROR] [Config::validateSplits] Split mainArea 88 non valida
10:02:39.935 -> [2472 ms] [X ERROR] [Config::validateSplits] Split outArea 111 non valida
10:02:39.935 -> [2473 ms] [X ERROR] [Config::validateSplits] Split mainArea 89 non valida
10:02:39.935 -> [2474 ms] [X ERROR] [Config::validateSplits] Split outArea 110 non valida
10:02:39.935 -> [2474 ms] [X ERROR] [Config::validateSplits] Split outArea 111 non valida
10:02:39.935 -> [2475 ms] [X ERROR] [Config::validateToggles] Toggle areaRead 54 non valida
10:02:39.935 -> [2476 ms] [X ERROR] [Config::validateToggles] Toggle forward 55 non valida
10:02:39.935 -> [2476 ms] [X ERROR] [Config::validateToggles] Toggle areaRead 90 non valida
10:02:39.935 -> [2477 ms] [X ERROR] [Config::validateToggles] Toggle forward 82 non valida
10:02:39.935 -> [2478 ms] [X ERROR] [Config::validateToggles] Toggle forward 55 non valida
10:02:39.935 -> [2478 ms] [X ERROR] [Config::validateToggles] Toggle areaRead 63 non valida
10:02:39.935 -> [2479 ms] [X ERROR] [Config::validateToggles] Toggle areaRead 64 non valida
10:02:39.935 -> [2480 ms] [X ERROR] [Config::validateToggles] Toggle areaRead 91 non valida
10:02:39.935 -> [2480 ms] [X ERROR] [Config::validate] CONFIGURAZIONE NON VALIDA -> REPORT DIAGNOSTICO
10:02:39.935 -> [2481 ms] [X ERROR] [DomoManager::setup] AVVIO BLOCCATO PER CONFIGURAZIONE NON VALIDA
```

Con OPTA che visualizza l'ultimo LED a destra sempre acceso (il 4), segnalando errore di configurazione.



Se invece vi fosse una configurazione vuota, alla partenza rimangono accesi fissi i due led centrali (led 2 e 3)

Diagnostica iniziale

Watchdog (da wdConfig in DMSetup.hpp)

Il sistema usa:

```
.activity_block_ms = 4000  
.update_slow_ms = 110  
.max_spikes = 8  
.spikes_window_s = 45  
.rate_limit_min_interval = 50
```

L'installatore deve verificare:

- Nessun ciclo supera 110 ms
- Nessun blocco oltre 4000 ms
- Spike sotto 8 nella finestra di 45 s

Aree monitorate

InitWatchedAreas() registra le aree critiche.

Verificare:

- Le variazioni vengono rilevate
- Nessuna area monitorata è virtuale per errore

Stato dispositivi

DomoManager aggiorna automaticamente:

- AREA_SYSTEM_ERRORS (che fa parte delle aree riservate di sistema)
- LED errori
- Log diagnostico

Verificare:

- Nessun dispositivo in errore
- Nessun gateway in cooldown
- Nessun reset IP imprevisto

Avvio del sistema

1. Alimentare Opta e Waveshare
2. Verificare comunicazione RS485
3. Caricare firmware Opta
4. DomoManager carica:
 - dispositivi

- aree
- toggles
- splits
- routes
- automazioni

5. Verificare log iniziali
6. Testare ingressi e uscite
7. Testare automazioni

Guida per installatori con diagrammi di flussi e dipendenze

Questa guida è pensata per:

- installatori che devono mettere in servizio un impianto Opta + Waveshare + Modbus
- tecnici che devono capire “chi fa cosa”
 - troubleshooting rapido
 - onboarding immediato senza dover leggere codice

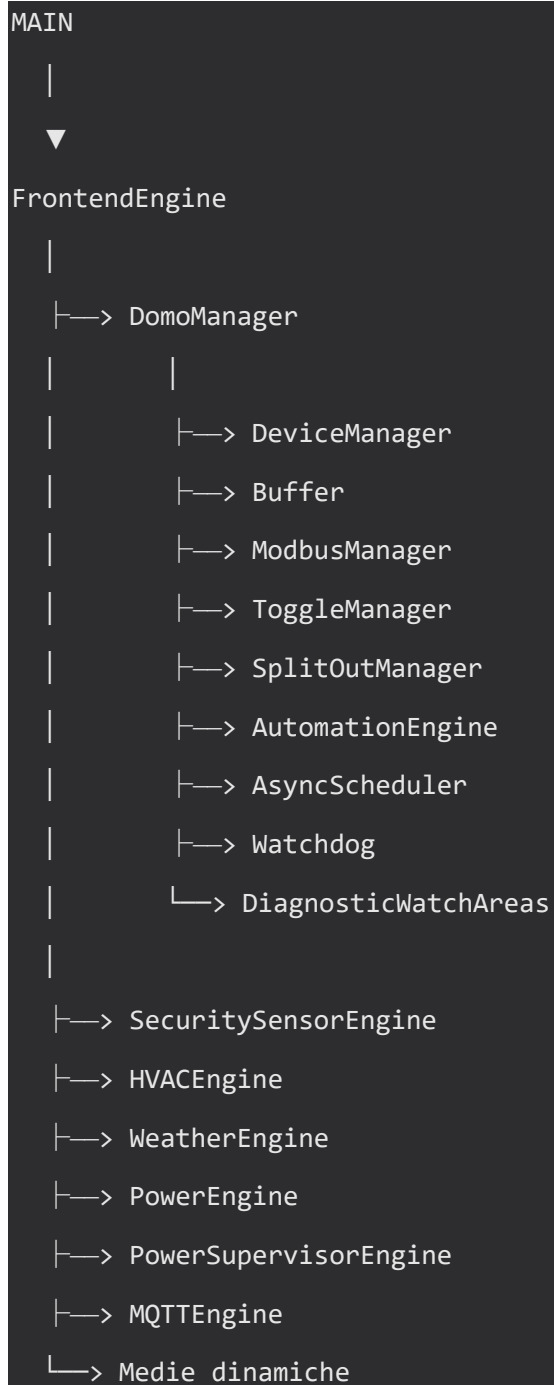
E contiene diagrammi testuali UML-style, perfetti per documentazione tecnica.

Flusso di avvio del sistema (BOOT FLOW)

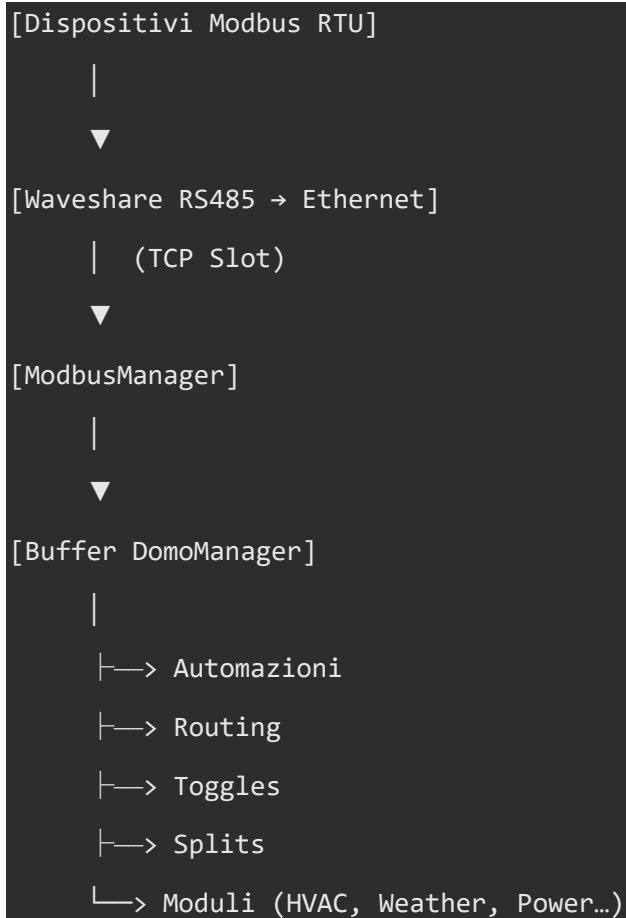
```
[Power ON]
|
▼
[MAIN]
- Inizializza Opta
- Carica FrontendConfig
- Avvia FrontendEngine
|
▼
[FrontendEngine::Setup]
- Setup rete
- Setup Modbus TCP
- Costruisce DomoManager
- Inizializza moduli (HVAC, Weather, Power...)
|
▼
[DomoManager::setup]
- Carica dispositivi (da DMSetup)
- Applica aree buffer
- Applica toggles, splits, routes
- Inizializza ModbusManager
- Avvia automazioni
|
▼
[Sistema operativo]
```

- Loop deterministico
- Polling Modbus
- Automazioni dinamiche
- Watchdog
- Diagnostica

Dipendenze tra i moduli (Dependency Graph)



Flusso Modbus (Gateway → Opta → Buffer)



Regola fondamentale sugli ID Modbus

Gli ID devono essere **univoci per ogni gateway**, ma **possono ripetersi su gateway diversi**, ad es. ID=2 su gateway 1 ed ID=2 su gateway 2.

Flusso di configurazione (DMSetup → DomoManager)

DMSetup.hpp

```
|  
├─ mainDevicesConfig    → lista dispositivi  
├─ mainAreasConfig      → definizione aree buffer  
├─ mainTogglesConfig    → logiche toggle  
├─ mainSplitsConfig     → temporizzazioni split  
├─ mainRoutesConfig     → routing personalizzato  
├─ wdConfig             → watchdog  
├─ wsc[]                → sensori cablati  
├─ MQTT_VARS[]          → variabili MQTT  
├─ hvacZones[]          → zone HVAC  
├─ gruppiMedie[]        → medie dinamiche  
└─ diagnosticParams     → diagnostica
```

DomoManager::setup()

```
|  
├─ mapDevices()  
├─ apply(areas)  
├─ apply(toggles)  
├─ apply(splits)  
├─ apply(routes)  
├─ initBuffer()  
├─ BuildIps()  
└─ ModbusManager.Begin()
```

Flusso diagnostica (Watchdog, Watch, DeviceErrors)

DomoManager

```
|  
├─ Watchdog (EMA Q8.8)  
├─ DeviceHasErrors()  
├─ Aree monitorate (Watch)  
├─ Spike detection  
├─ Rate limiting  
└─ Reset IP automatico
```

Troubleshooting rapido per installatori

Nessuna lettura Modbus?

- Controllare IP Waveshare
- Controllare ID dispositivo
- Controllare slot Modbus TCP
- Controllare terminazione RS485
- Verificare DeviceHasErrors()

Un'area non cambia?

- Verificare che sia mappata in mainAreasConfig
- Verificare che non sia virtuale
- Verificare che il dispositivo la esponga

Troubleshooting Avanzato

Obiettivo del capitolo

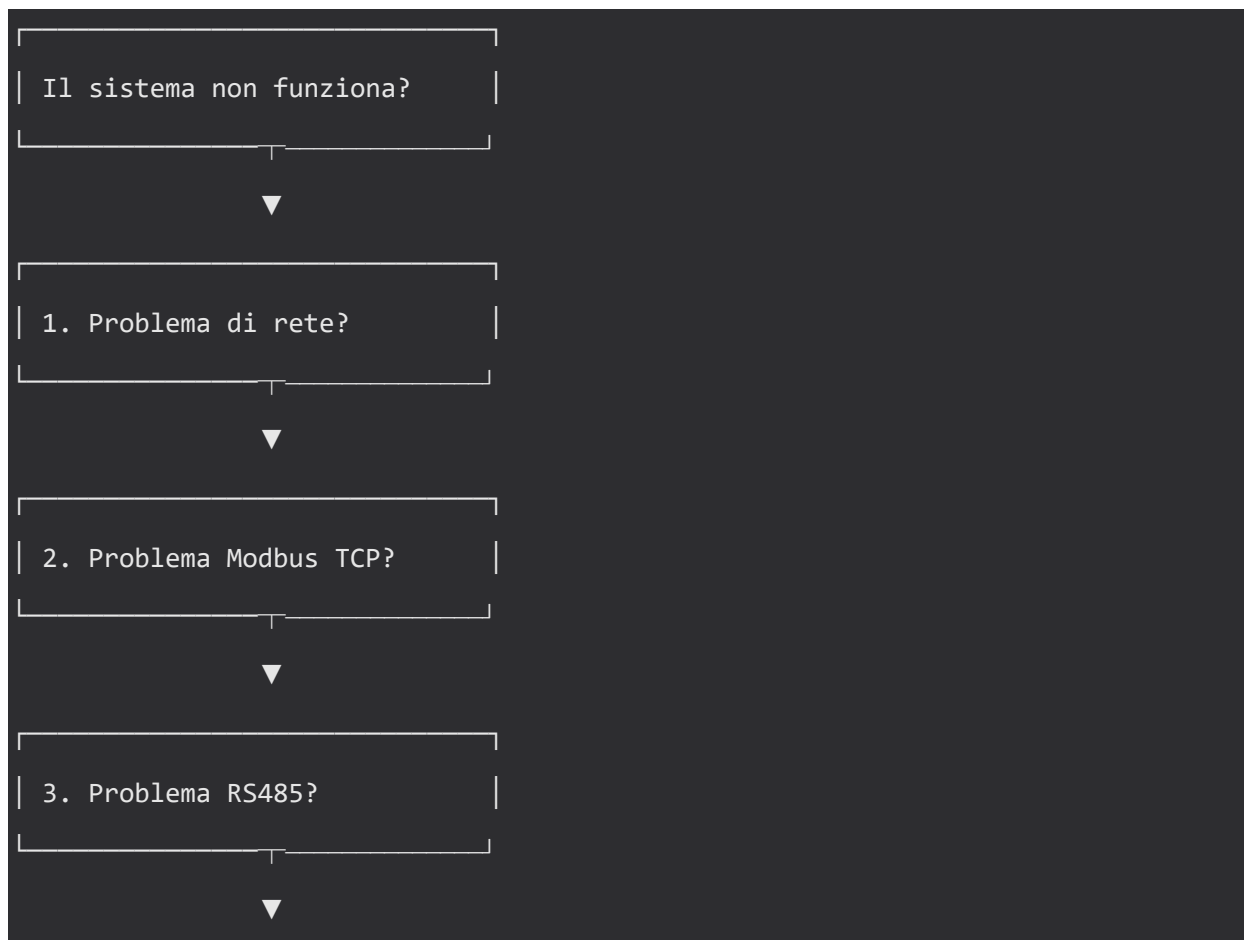
Fornire una guida completa per:

- Identificare rapidamente un problema
- Capire la causa reale
- Applicare la soluzione corretta
- Evitare errori ricorrenti
- Diagnosticare problemi complessi (Modbus, RS485, buffer, automazioni, watchdog, IP Manager)

Obiettivo: fornire una procedura di troubleshooting **deterministica, ripetibile e veloce**, per individuare e correggere problemi in:

- rete Ethernet
- Modbus TCP
- RS485
- dispositivi Modbus
- buffer DomoManager
- AutomationEngine
- Watchdog

Diagramma generale del troubleshooting





Problemi di rete (Ethernet)

! Sintomi

- Opta non risponde al ping
- Waveshare non raggiungibili
- Modbus TCP non si connette
- Timeout immediati su tutti i dispositivi

🔍 Cause probabili

- IP errato
- Gateway errato
- Subnet errata
- Cavo difettoso
- Switch spento

🔧 Soluzioni

- Verificare IP Opta: 192.168.12.202
- Verificare IP Waveshare: 192.168.12.203 / 192.168.12.204
- Verificare subnet: 255.255.255.0
- Verificare gateway: 192.168.12.1
- Verificare che Waveshare sia in **TCP Server**
- Controllare che non ci siano **IP duplicati** nella rete
- Sostituire cavo Ethernet

Problemi Modbus TCP

! Sintomi

- Timeout Modbus
- Errori CRC
- Nessuna risposta dal gateway

🔍 Cause probabili

- Porta TCP errata (deve essere 502)
- Timeout troppo basso
- Waveshare in modalità sbagliata
- Slot Modbus (ID univoco) errato

✂ Soluzioni

- Impostare porta: 502
- Timeout: 200–300 ms
- Modalità Waveshare: **TCP Server**
- Verificare slot configurato in DMSetup

Problemi RS485

! Sintomi

- Nessuna risposta dai dispositivi
- Risposte intermittenti
- Errori CRC frequenti
- Letture che cambiano senza logica

🔍 Cause probabili

- A/B invertiti
- GND non comune
- Terminazione mancante
- Cavo non schermato
- Lunghezza cavo eccessiva

✂ Soluzioni

- Verificare $A \rightarrow A$, $B \rightarrow B$
- Aggiungere GND comune
- Terminatore 120Ω sull'ultimo dispositivo
- Usare cavo twistato schermato
- Evitare topologia a stella

Problemi dispositivi Modbus

! Sintomi

- Un solo dispositivo non risponde
- Risponde ma con valori errati
- Risponde solo a volte

🔍 Cause probabili

- ID Modbus errato
- Alimentazione insufficiente
- Profilo errato in DMSetup
- Slot errato

✂ Soluzioni

- Verificare ID (univoco per gateway)
- Verificare alimentazione 24V
- Verificare profilo: "N4DIH32" / "MA01_AXCX0080" / "CTR4A01"
- Verificare slot Modbus TCP

Problemi buffer (DomoManager)

! Sintomi

- Aree non aggiornate
- Aree virtuali non previste
- Aree duplicate
- Aree mancanti

🔍 Cause probabili

- Area non definita in `mainAreasConfig`
- Area non mappata da dispositivo
- Area definita due volte
- Area fuori range

✂ Soluzioni

- Verificare log:

```
Aree virtuali auto-determinate: ...
```
- Verificare duplicati:

```
ERRORE: Area inizializzata più volte
```
- Controllare mapping dispositivi
- Verificare che nessuna automazione scriva su aree non allocate

Problemi watchdog

! Sintomi

- Log SPIKE frequenti
- Reset imprevisti
- `updateCycle` troppo alto

🔍 Cause probabili

- Automazioni troppo pesanti
- Dispositivo Modbus lento
- Loop congestionato

✂ Soluzioni

Verificare log:

```
[SPIKE] ...  
updateCycle avg=...
```

- Ridurre peso automazioni
- Aumentare timeout Modbus (se >300 e continuano gli errori, verificare cablaggio, terminatori, schermatura cavi)
- Verificare dispositivi in errore

Problemi IP Manager

! Sintomi

- IP in cooldown
- Reset IP frequenti
- Reset totale sistema

🔍 Cause probabili

- Dispositivo non risponde
- Timeout troppo basso
- Rete instabile

🔧 Soluzioni

- Verificare log:

```
IpCooldownLogger
```

- Aumentare timeout Modbus
- Verificare rete fisica
- Sostituire cavo Ethernet
- Verificare connessioni a switch
- Verificare sovrapposizioni di IP con altri dispositivi

Tabella riassuntiva problemi → soluzioni

Problema	Causa probabile	Soluzione
Nessuna lettura Modbus	IP errato	Verifica IP Waveshare
Errori CRC	RS485 invertito	A→A, B→B
Dispositivo non risponde	ID errato	Imposta ID corretto
Area non aggiorna	Area virtuale	Verifica mapping
Spike watchdog	Overload	Ottimizzare automazioni
IP cooldown	Timeout basso	Aumentare timeout

Glossario Tecnico

Manuale di Onboarding – Impianto Opta + Waveshare + Modbus RTU

Questo glossario fornisce definizioni operative dei termini utilizzati nel sistema DomoManager, indispensabili per installatori e tecnici.

Buffer

È la mappa centrale dello stato dell'impianto. Contiene tutte le aree (ingressi, uscite, virtuali, medie, bitmask, ecc.).

Ogni area è un numero intero (es. 78, 110, 155).

Area

È l'unità logica del buffer. Può rappresentare:

un ingresso digitale

un'uscita digitale

un valore analogico

un flag virtuale

una media

un bitmask

Virtual Area

Area non collegata a un dispositivo fisico. È generata automaticamente se non mappata da nessun device.

Toggle

Logica che trasforma un fronte di salita in un cambio di stato.

Esempio: Pulsante → accende/spegne una luce.

Split

Logica temporizzata: Un'area principale attiva una o più aree per un tempo definito.

Esempio: Pulsante → attiva motore vasistas per 30 secondi.

Route

Regola di routing: Quando un'area cambia, altre aree vengono aggiornate.

Esempio: Pulsante bagno → accende LED + luce principale.

Scene

Insiemi di azioni predefinite.

Esempio: Scena "CappaOn" → accende estrattore + LED.

Regola dinamica

Automazione definita nel JSON.

Tipi:

- threshold
- trend
- bitmask
- multi (AND/OR)
- time
- sequence
- composite

Trend

Calcolo della variazione nel tempo di un valore.

Esempio: Umidità bagno in aumento → attiva estrattore.

Bitmask

Valore intero che contiene più flag booleani.

Esempio: Stato sicurezza (porte, finestre, allarmi).

H24 / RT

- H24 → protezione 24H per taglio cavi \ apertura sensore
- RT → sensore con ritardo (Real Time con debounce)

Watchdog

Sistema che monitora prestazioni e stabilità:

- spike
- overload
- updateCycle
- activityLoop

Cooldown IP

Stato in cui un IP viene temporaneamente escluso per errori ripetuti.

Slot Modbus TCP

Numero di canale TCP usato per comunicare con un dispositivo RTU tramite Waveshare.

Profilo dispositivo

Descrive come leggere/scrivere un dispositivo Modbus.

Esempio: "N4DIH32" → 32 ingressi digitali.

Disclaimer Legale – Limitazione di Responsabilità

Il software **DomoManager**, la relativa documentazione tecnica e tutti i moduli associati (inclusi ma non limitati a: *DM.hpp*, *DMFrontend.hpp*, *DMSetup.hpp*, *DMDiagnostic.hpp*, automazioni, configurazioni e profili hardware) sono forniti **“così come sono”**, senza alcuna garanzia espressa o implicita.

L'autore e i collaboratori **non assumono alcuna responsabilità** per:

- danni diretti o indiretti
- malfunzionamenti
- perdita di dati
- interruzioni di servizio
- danni a cose, impianti o persone
- utilizzo improprio o non conforme del software
- modifiche non autorizzate al codice o alla configurazione
- integrazioni con dispositivi di terze parti non certificati

L'utente accetta che:

L'installazione, la configurazione e l'uso del software avvengono sotto la propria esclusiva responsabilità.

Il sistema è progettato per applicazioni domotiche e di automazione non critiche. Non deve essere utilizzato in contesti dove un malfunzionamento potrebbe causare:

- rischi per la sicurezza personale
- danni strutturali
- interruzioni di servizi essenziali
- condizioni pericolose per persone o animali

L'utente è tenuto a:

- verificare la correttezza delle configurazioni
- testare ogni automazione prima dell'uso reale
- assicurarsi che i dispositivi collegati siano conformi alle normative vigenti
- implementare adeguate misure di sicurezza elettrica e informatica

L'autore declina ogni responsabilità derivante da:

- modifiche al codice sorgente
- utilizzo in ambienti industriali o critici
- integrazioni con hardware non supportato
- errori di cablaggio o installazione
- uso improprio delle automazioni o delle scene

L'uso del software implica l'accettazione integrale del presente disclaimer.

DOMO Manager è ciò che davvero conta:

*la continuità, la sicurezza, la sostenibilità e la qualità dell'ambiente in cui viviamo ogni
giorno...*
