iotProject

Alessio Tommasi

12 febbraio 2025

Indice

Capitolo 1

Introduzione

Il progetto *iotProject* è stato sviluppato nel corso di IoT del Master in Informatica presso SUPSI. Il focus principale è sull'ESP32 e il protocollo Modbus.

1.1 Dipendenze

Driver Per gli utenti Windows, è necessario installare `CP210xDriver`

Compiler Per compilare tale progetto e'stato utilizzato Arduino IDE 2.3.3. disponibile al seguente link: Arduino IDE.

1.2 Configurazione dell'Arduino IDE

Link repo ufficiale: iotProject.

Per compilare i file nelle sottocartelle, è necessario aggiungerli come librerie (`.zip`) all'Arduino IDE. Ho creato una cartella specifica per le librerie dove posizionare o sostituire i file zip. Per una corretta compilazione, importa tutte le cartelle zip presenti in `/Library`.

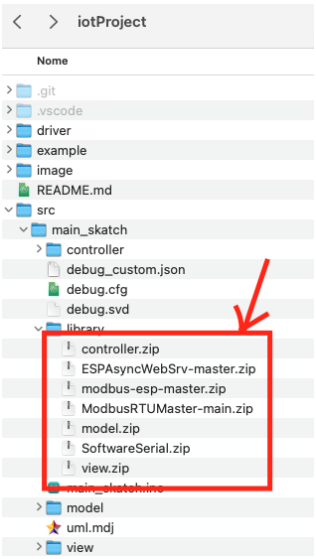

Figura 1.1: Importazione delle librerie nell'Arduino IDE

Altrimenti clonare la versione Portable del progetto disponibile al seguente link: iotProject-portable.

Capitolo 2

Hardware

2.1 Board Esam

2.1.1 Funzionamento

Multiplex

Il dispositivo di multiplexing riulta essere il **CD405xB**.
sviluppato da Texas Insreumet, link alla documentazione ufficiale:
CD405xB.
Siccome successivamente servira diporto di seguito la tabella di verita di tale dispositivo. Figura 5.1

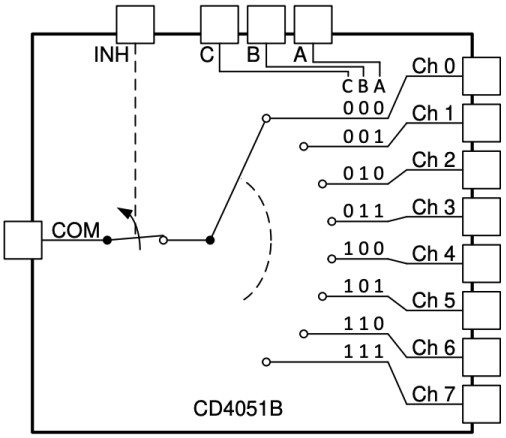

Figura 2.1: Tabella di verita del CD405xB

Collegamenti Multiplexer

I pin di ingresso A, B, C del multiplexer **CD405xB** sono collegati rispettivamente ai pin GPIO 12, 13, 14 dell'ESP32. La selezione dei canali del multiplexer avviene impostando i pin A, B, C come segue:

Canale	**Pin A**	**Pin B**	**Pin C**
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0
4	0	0	1
5	1	0	1
6	0	1	1
7	1	1	1

Tabella 2.1: Configurazione dei pin per la selezione dei canali del multiplexer

Canali Multiplexer

Nella figura nella pagina seguente seguito vennogno riportati i collegamenti dei canali del multiplexer con i collegamenti esterni.

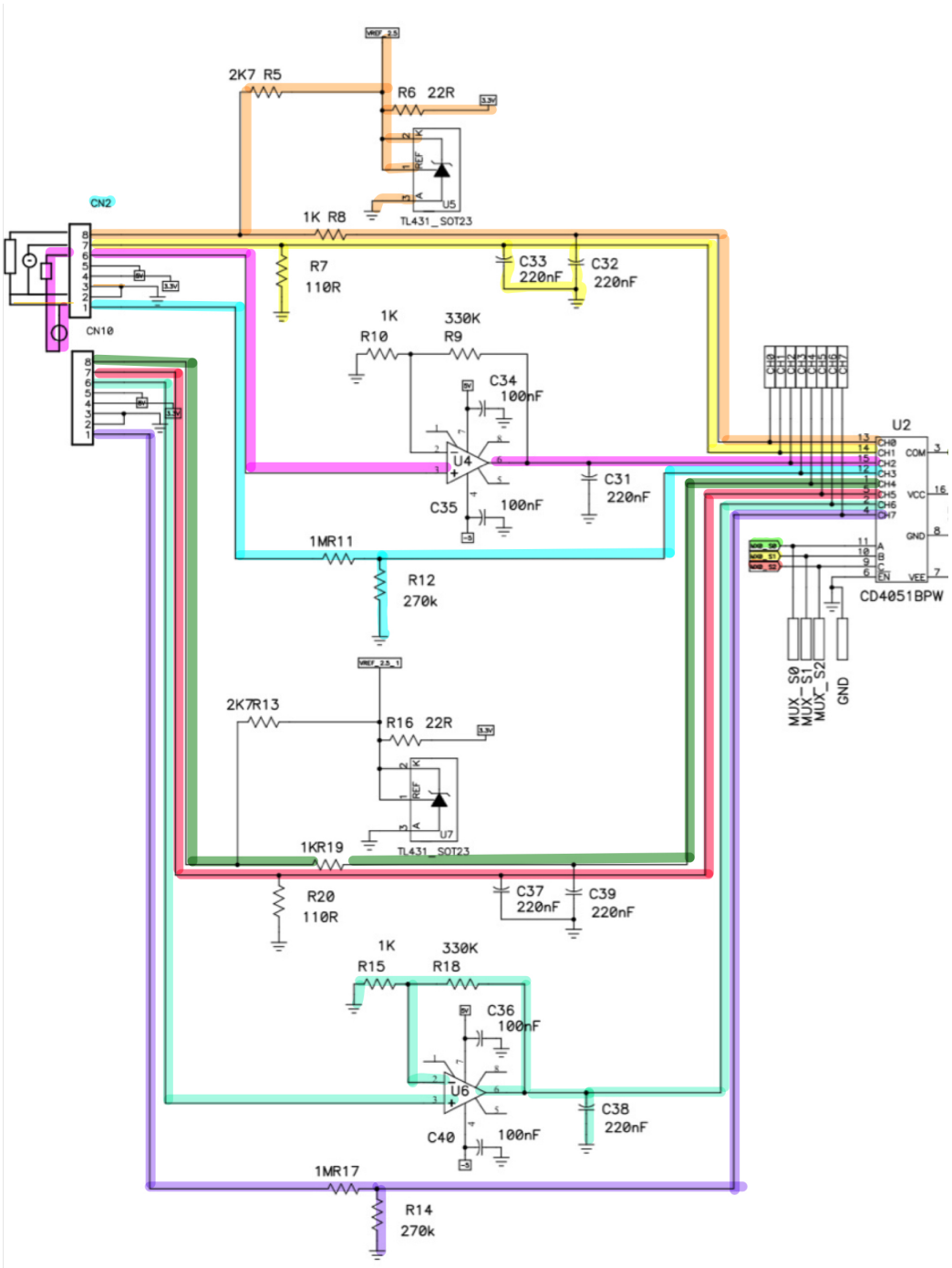

Figura 2.2: Collegamenti dei canali del multiplexer

Ch0 Selezionando Ch0 dall apposita pagina dedicata segue il percorso evidenziato in arancione nella Figura 5.2. E' collegato a pin 8 della morsettiera CN2. Su tale morsetto ci aspettiamo in ingresso un segnale di corrente che il componente TL431 si occupera di regolare e stabilizzare la tensione in ingresso entro dei range indicati. a questo link potete trovare il datasheet di tale componente per maggiori info TL431.

Ch1 percorso evidenziato in giallo nella Figura 5.2.
Possibili funzioni del segnale sul segnale in ingreso al pin 7 della morsettiera CN2.
- **Filtro Passa Basso** Se il segnale applicato su R7 è una tensione alternata (AC), il circuito potrebbe attenuare le alte frequenze, lasciando passare solo le frequenze più basse
- **Stabilizzazione del segnale**

Ch2 percorso evidenziato in viola nella Figura 5.2.
Leggeremo il segnale in ingresso al pin 6 della morsettiera CN2.
Il segnale verra amplificato dall operazionale U4 che e collegato come amplificatore non invertente (maggiori info al link LM358).
Il guadagno di U4 e' dato dalla formula $G = 1 + \frac{R9}{R10} = 1 + 330 = 331$.
inoltre tali alimentatore e alimentato tra 0 e +5v quindi clampera il segnale in ingresso tra 0 e +5v. Il condensatore C31 si occupa di stabilizzare il segnale in uscita.

Ch3 percorso evidenziato in azzurro nella Figura 5.2.
Leggeremo il segnale in ingresso al pin 1 della morsettiera CN2.
tra il segnale sul pin 1 ed ESP e'presente un partitore di tensione composto da R11 = 1M e R12 = 270K dunque
$Vesp = Vpin1 * \frac{R12}{R11+R12} = Vpin1 * \frac{270}{1270} = Vpin1 * 0.2126.$

Ch4 Connessione equivalente a CH0

Ch5 Connessione equivalente a CH1

Ch6 Connessione equivalente a CH3
Tutte le connessioni equivalenti hanno un circuito hw separato per garantire la massima indipendenza tra i canali.

2.1.2 hardware esterno posizionabile sulla board

MAX31865

asd

ESP32 38 Pin

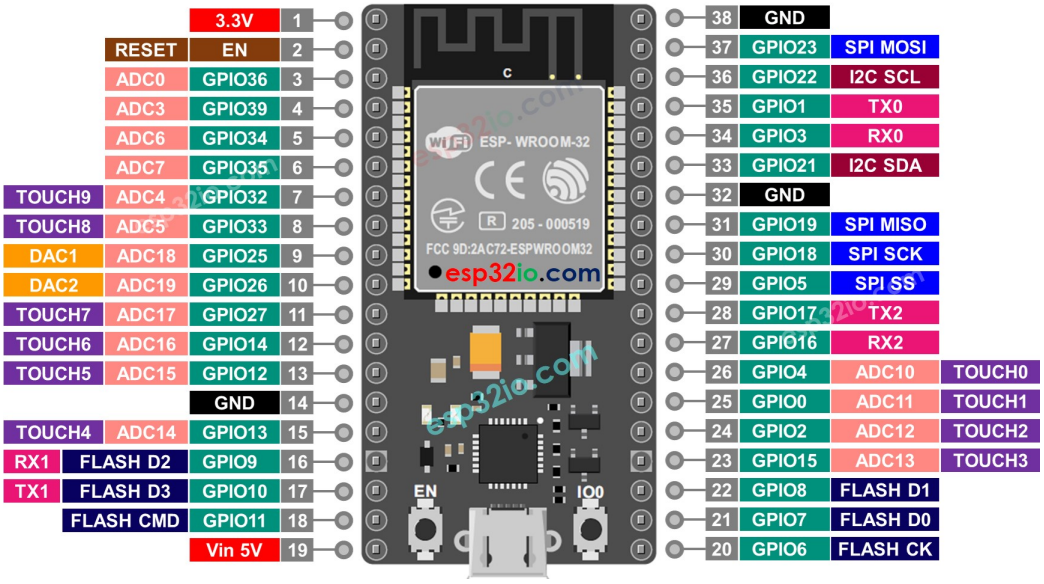

Figura 2.3: Pinout dell'ESP32-DOIT-DEV-KIT v1

Capitolo 3

Software

3.1 Diagramma UML

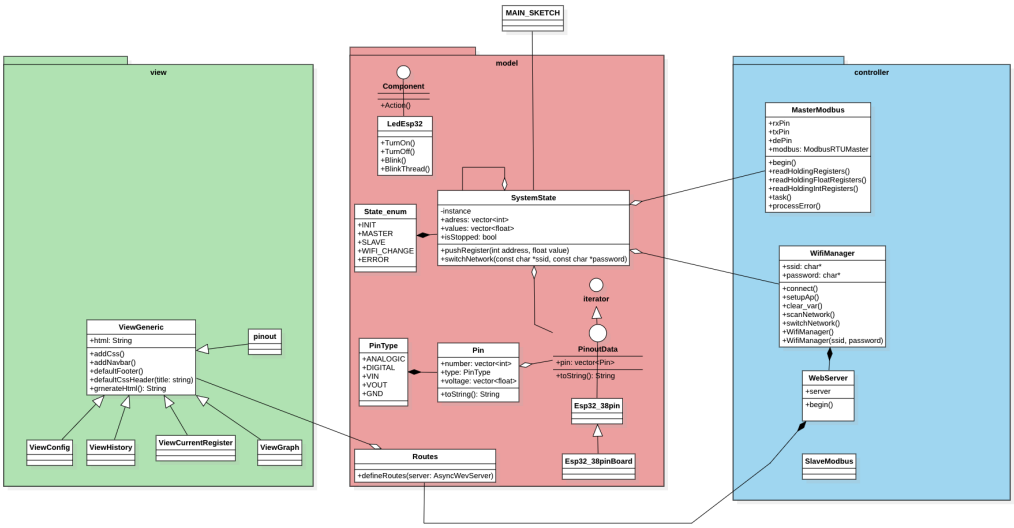

Figura 3.1: Diagramma UML del sistema

3.1.1 Pattern

Model-View-Controller (MVC)

Model Contiene la struttura dei dati e le funzioni per accedere e modificarli. Le classi coinvolte sono `SystemState`, `Pin`, `PinoutData`, `Routes`.

View Si occupa di visualizzare i dati e di interagire con l'utente. principalmente sono classi che si occupano della generazione dei componenti delle pagine web visualizzate dall utente.

Controller Gestisce le richieste dell'utente e aggiorna il modello di conseguenza.

Singleton

utilizzato per garantire unicita e atomicita dei dati `SystemState` e'la classe che implementa tale pattern. Sara particolarmente utile in futuro per l'implementazione di un sistema di datalogging e persistenza dei dati su scheda SD.

3.2 performace

3.3 WebServer

3.4 Modbus

Il file utilizzato per testare lo slave è disponibile qui: modbusSlave2.ino.

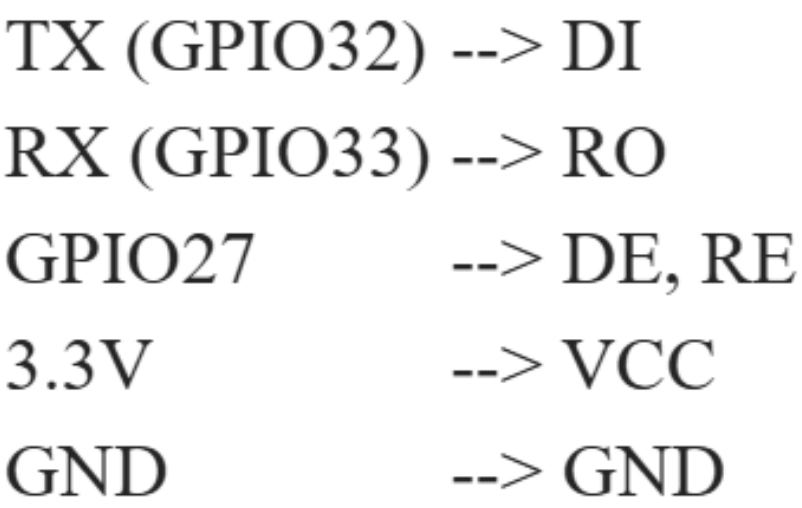

Figura 3.2: Pinout proposto per il dispositivo slave Modbus

Capitolo 4

Attività

Attività	Descrizione
Configurazione sensori di temperatura	Configurare e integrare sensori di temperatura **PT100**, **PT1000** e **termocoppie** utilizzando moduli come **MAX31865** e **MAX31855**.
Lettura segnali analogici	Implementare la lettura di segnali analogici tramite gli ingressi **ADC** dell'ESP32 e eventuali moduli esterni.
Gestione uscite digitali e analogiche	Sviluppare la gestione delle uscite digitali e analogiche tramite l'ESP32.
Comunicazione RS485 (Modbus RTU)	Integrare la comunicazione **RS485** utilizzando il protocollo **Modbus RTU** per interfacciarsi con altri dispositivi.
Server Web (Ethernet TCP/IP)	Sviluppare un server **Web** basato su **Ethernet TCP/IP** per il monitoraggio e controllo remoto dei dati acquisiti.
Datalogging	Implementare un sistema di **datalogging** per salvare e storicizzare i dati raccolti dai sensori.
Test e validazione	Testare e validare il sistema attraverso simulazioni e test su hardware reale.

Capitolo 5

Conclusioni

5.1 Sviluppi futuri

persistenza e datalogging su scheda SD

backup dati su cloud implementazione backend etc

5.2 Ringraziamenti