Submeter intelligente

Studente/i

Marinoni Marco

Relatore

Rivola Davide

Correlatore

Medici Vasco

Committente

Corso di laurea

Ingegneria Elettronica

Modulo

Progetto Diploma M-P6060.1

Anno

2019/2020

Data

26 agosto 2020

Indice

Elenco delle figure

Capitolo 1

Abstract

In questo progetto si va a realizzare un prototipo di submeter intelligente utilizzabile ad esempio in una palazzina in cui vi è un Master centrale e almento due slave corrispondenti agli appartamenti della palazzina che comunicano tra di loro per mezzo del protocollo modbus, tipico di queste applicazioni. I vari slave si occupano della misurazione di alcuni parametri fondamentali tra cui corrente, tensione, potenza, cosphi e di inviare queste info al master che le processa e le inserisce in un database. Quest'ultimo sarà poi utilizzato da un editor grafico per visualizzare i parametri su una pagina web e tenere monitorato a proprio piacimento tutto quello che succede. Il Master possiede un piccolo Oled in grado di visualizzare instantaneamente le misure effettuate e tramite un pulsante di switchare la visualizzazione di tali. Da lato slave vi è un rtc in grado di mantenere l'orario del dispositivo anche in assenza di alimentazione. Il sistema Modbus è testato sia in modalità half-duplex per il primo slave, sia in modalità full duplex per il secondo slave per dimostrare i possibili casi di funzionamento.

Capitolo 2

Design di sistema

2.1 Descrizione

In Svizzera, la generazione solare sta raggiungendo livelli considerevoli di penetrazione, e dovrebbe aumentare ulteriormente grazie a un'ampia gamma di fattori ambientali, sociali, tecnici ed economici (non da ultimo grazie alla "Strategia energetica 2050"). Da questa crescita, emerge una serie di problemi relativi all'operazione e agli impatti economici della generazione fotovoltaica (FV) sulla rete elettrica. La nuova ordinanza sull'energia, in vigore dal 1 gennaio 2018, consente la creazione di comunità di autoconsumo (SCC), in cui l'energia può essere scambiata internamente tra i loro membri, senza essere soggetti a tariffe di rete e tasse sulla rete. Le SCC rappresentano quindi una soluzione efficiente per la riduzione dei problemi di rete. Aumentando il numero di carichi che hanno accesso alla generazione locale (e possibilmente allo stoccaggio) si aumenta il potenziale di autoconsumo, ottenendo inoltre un controllo più preciso dell'iniezione e del consumo di energia, migliorando così anche al redditività del fotovoltaico e dello stoccaggio. Per la gestione di queste comunità di autoconsumo si vuole realizzare un submeter intelligente per comunità in autoconsumo. Base di partenza è un prototipo di smart meter esistente (basato su Raspberry Computing Module + metering chip ADE9000). Il circuito verrà ottimizzato per essere inserito in un case DIN (adatto per le installazioni elettriche), realizzando due moduli:

- submetering module: acquisizione dei dati elettrici relativi dei singoli partecipanti nella comunità
- computing module: elaborazione dati ricevuti dai submeters

Il computing module potrà quindi leggere dati da n-submeters installati localmente.

2.2 Compiti

- Definizione delle funzionalità e scelta dei componenti
- Progettazione, configurazione e realizzazione del measure module e il computing module
- Programmazione firmware per l'acquisizione delle misure
- Calibrazione e test del sistema

2.3 Obbiettivi

- Design elettronico su altium completato
- Sviluppo, produzione e montaggio dei prototipi PCB
- Assemblaggio delle schede in un case DIN
- Test di un set di funzionalità di base: I,P,V,cosphi
- Calibrazione

2.4 Tecnologie

- Raspberry compute module

Figura 2.1: Raspberry compute module

- PCB (Altium)

Figura 2.2: Altium logo

- Python

Figura 2.3: Python logo

- ADE9000

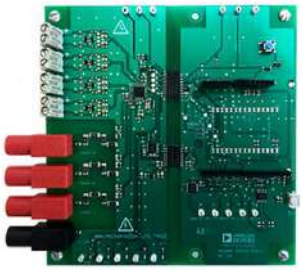

Figura 2.4: Ade9000 shield

2.5 Valutazione Smart dei requisiti

I requisiti per risultare tali devo essere SMART, ciò significa che devono includere tutte le seguenti caratteristiche per evitare fraintendimenti e definire in modo corretto le necessità del progetto:

- SPECIFIC: Chiaro e specifico per evitare fraintendimenti
- MEASURABLE: Misurabilità
- ATTAINABLE: Raggiungibile, ragionevole, credibile
- RELEVANT: Rilevante, realistico
- TIMELY: Limite temporale assegnato

Definizione delle funzionalità e scelta dei componenti	S = Si M = Si, scelto o non scelto A = Si R = Si T = Si, Entro 28/09/20
Measure module e il computing module	S = Si M = Si, funzionante o no A = Si R = Si T = Si, Entro 28/09/20
Programmazione firmware per l'acquisizione delle misure	S = Si M = Si, funzionante o no A = Si R = Si T = Si, Entro 28/09/20
Calibrazione e test del sistema	S = Si M = Si, C'è o non c'è A = Si R = Si T = Si, Entro 28/09/20

Tabella 2.1: Tabella SMART

2.6 Gantt

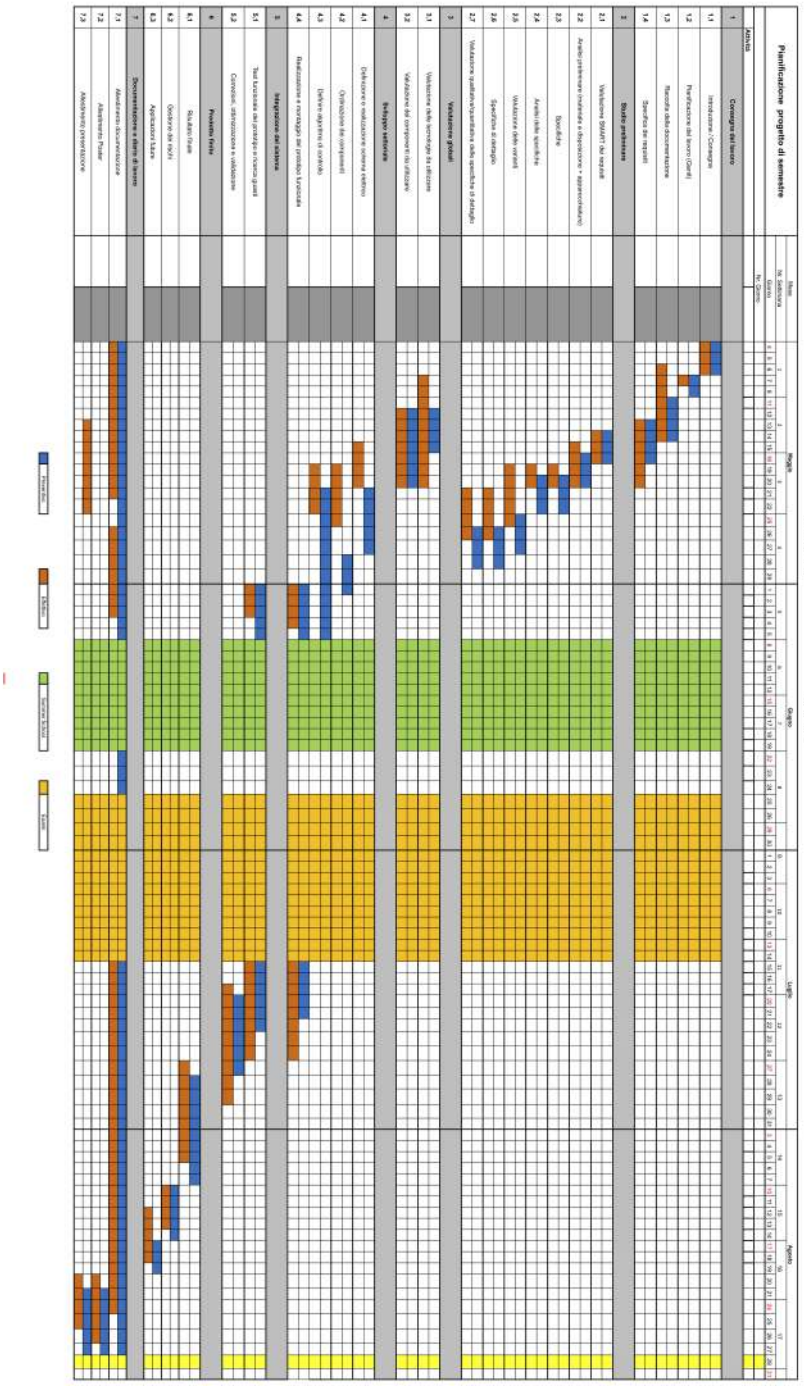

Figura 2.5: Gantt

Capitolo 3

Analisi teorica progetto-schema a blocchi

L'idea di questo progetto è di creare un sistema Master con N-slave che comunicano tra di loro via modbus rs485 half o full duplex, e si scambiano informazioni riguardandi le varie misure che gli slave eseguono per poi andare ad analizzarle tramite il master e capire lo stato della rete per eventualmente intervenire su sistema ed eseguire possibili modifiche. Il primo schema a seguire indica la richiesta per questo progetto a grandi linee, dove vi è raffigurato lo slave con RTC e ADE9000 e dall'altra il master con la raspberry Compute Module che riceve i vari dati. Nel secondo schema è analizzata più in dettaglio la parte slave dove vi sono presenti tutti i vari collegamenti eseguiti con i vari componenti che fanno parte dello slave(inizialmente per delle prove tutte evaluation board) con corrispettivi link e un esempio di codice da applicare per fare delle prime prove. L'ultimo Schema a blocchi invece rappresenta il lato raspberry Master dove è evidenziato il fatto che vi sono più possibilità per collegare via modbus la raspberry e lo shield per la connessione ethernet visto che il Compute Module non è previsto di porta internet. Questi sono solo schemi raffigurativi utilizzati in partenza per una prima realizzazione del progetto. Infatti poi come circuiti finali sono stati aggiunti un display oled con pulsante per variare le schermate, led di presenza tensioni e alimentatore diretto ac/cc 220/5V dc.

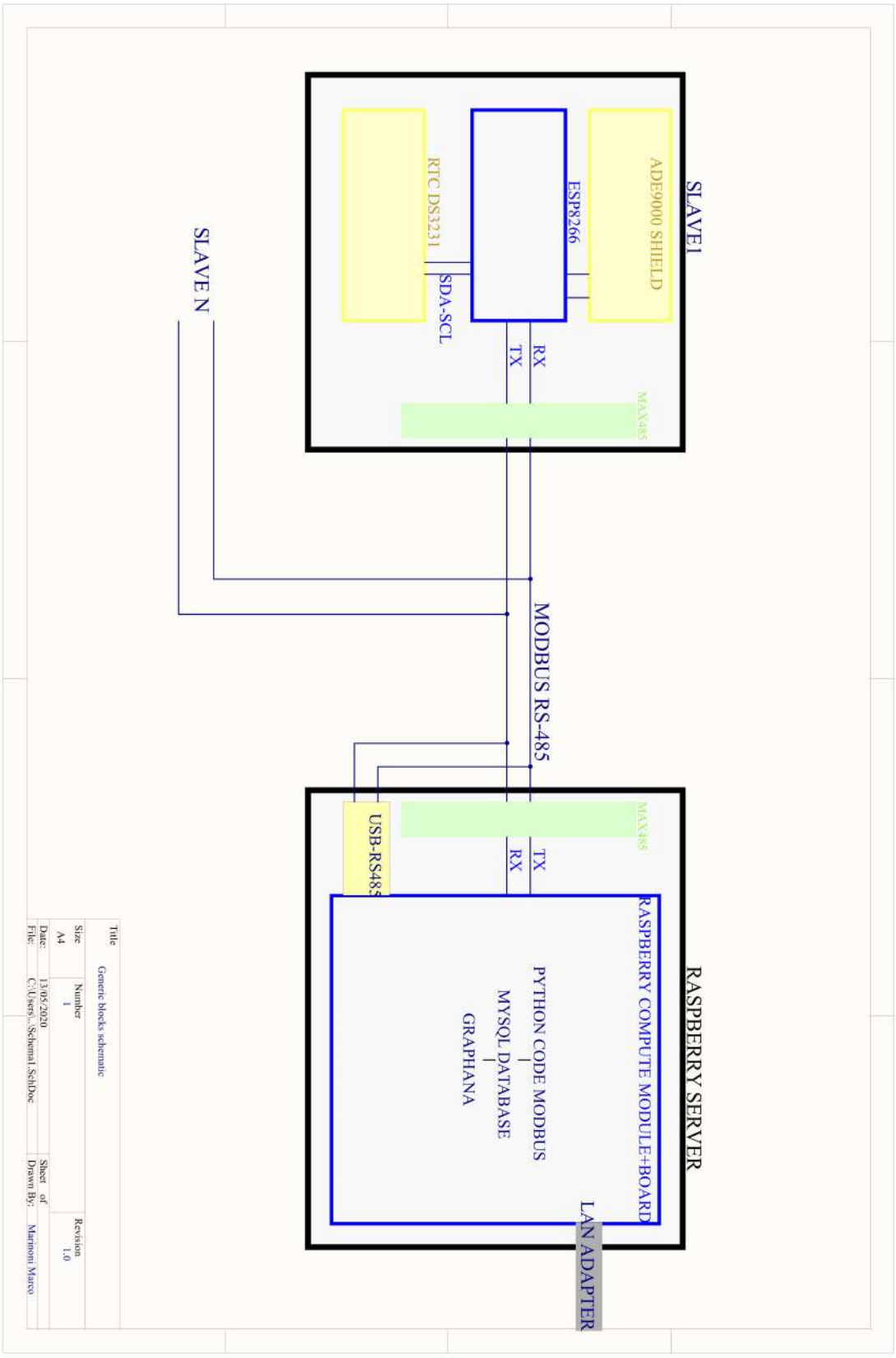

Figura 3.1: Schema a blocchi riassuntivo

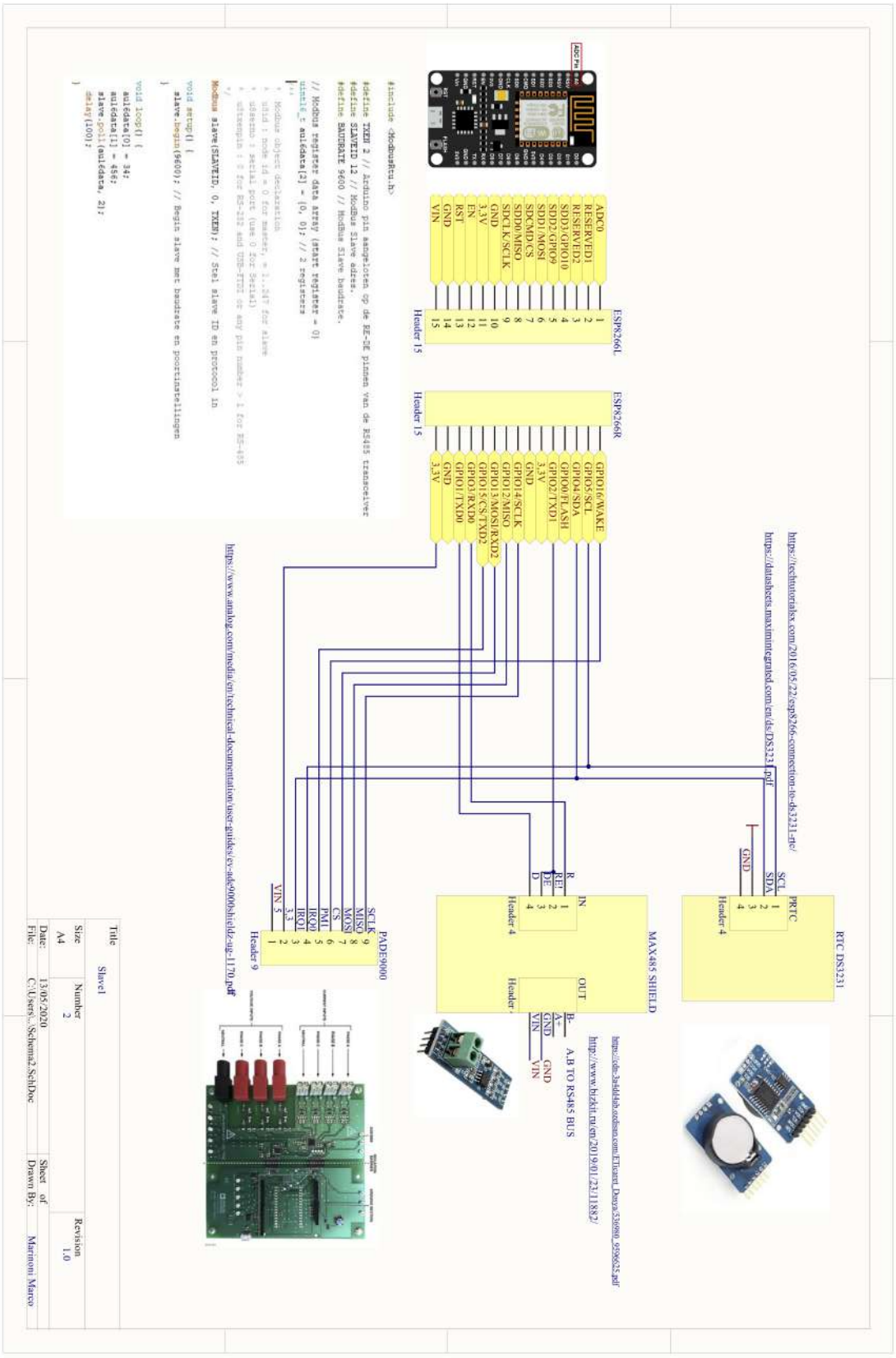

Figura 3.2: Schema a blocchi slave

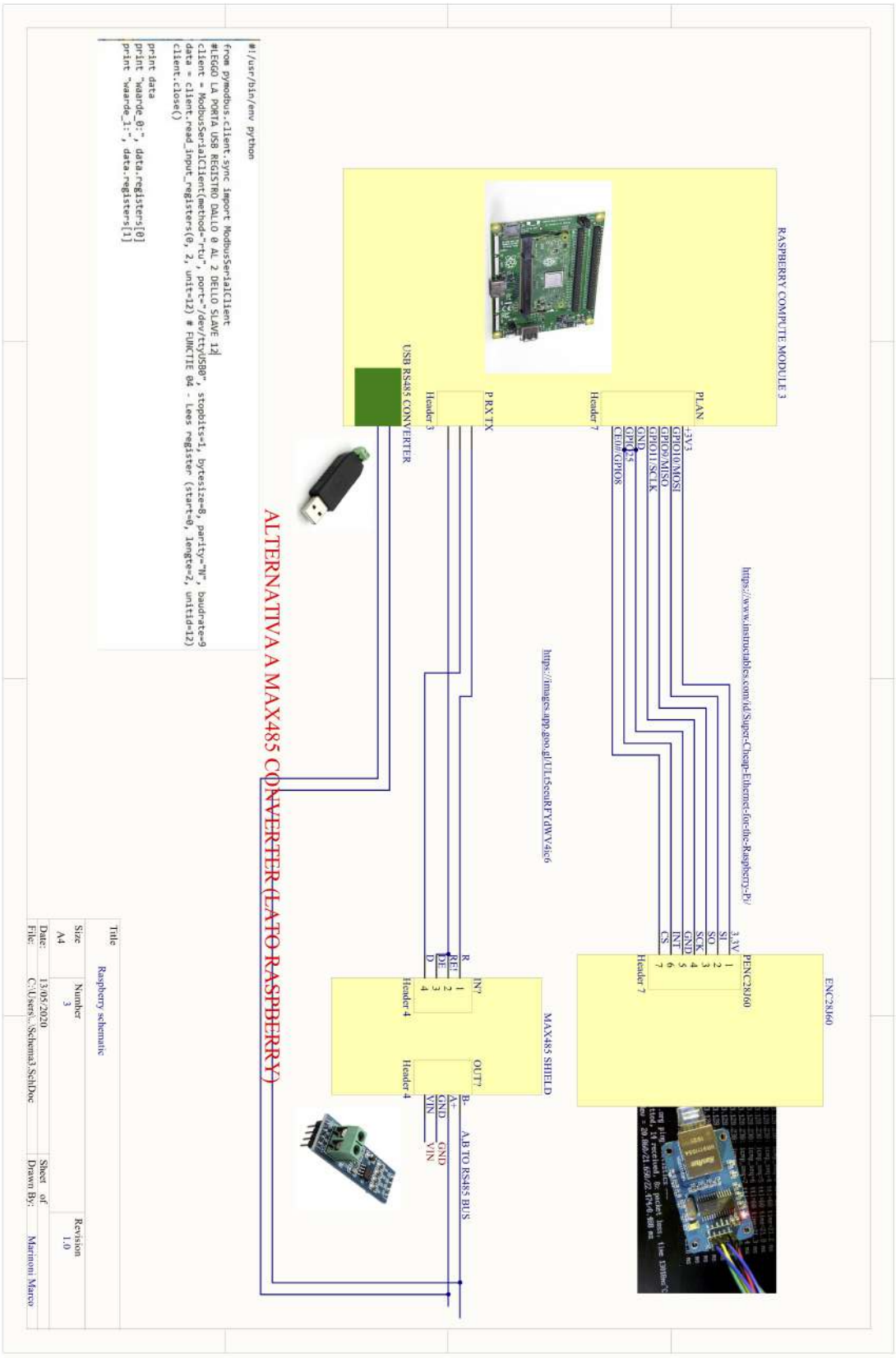

Figura 3.3: Schema a blocchi Master

Capitolo 4

Protocollo di comunicazione MODBUS

4.1 Introduzione

Il Modbus è ancora oggi uno dei protocolli di comunicazione più diffusi nel settore dell'automazione industriale nonostante le sue origini risalgano al 1979. La sua solida e duratura presenza nella vasta panoramica dei protocolli industriali è dovuta al fatto che il Modbus è un protocollo semplice e diretto. Le risorse hardware e software richieste dal protocollo Modbus sono piuttosto esigue e quindi esso è utilizzabile anche in sistemi basati su piccoli ed economici microprocessori. La semplice ed essenziale struttura dei dati trasmessi permette una comunicazione efficace con un buon rapporto tra la parte utile dell'informazione ed il numero totale di bytes trasmessi. Il protocollo si presta anche ad implementazioni parziali dei comandi, limitate alle specifiche esigenze, nonché all'espansione con l'aggiunta di comandi custom da parte del produttore del dispositivo. Molto spesso i produttori di dispositivi per l'automazione "cadono" nella tentazione di sviluppare un protocollo di comunicazione completamente proprietario pensando di fare la cosa più semplice ed efficace. Tuttavia questi protocolli sono frequentemente delle "brutte copie" di un protocollo già semplice e diretto come il Modbus ma non sono però compatibili ne con questo ne con altri protocolli standard. Ciò può costituire un vantaggio se si intende vincolare i clienti al proprio protocollo ma esclude la interoperabilità con tanti altri dispositivi commerciali e la possibilità che il prodotto sia esso stesso più commerciabile. Il protocollo Modbus è basato su bus di campo e questo significa che è adatto allo scambio di informazioni tra apparecchiature fisicamente distinte e posizionabili anche a notevoli distanze tra loro. La realizzazione di un macchina automatica o di un impianto mediante un insieme di dispositivi tra loro interconnessi ha molteplici vantaggi come:

- Possibilità di realizzare l'impianto di automazione utilizzando dispositivi commerciali, realizzati da diversi produttori e facilmente sostituibili tra loro.

- Distribuzione dei vari dispositivi in posizioni strategiche per l'impianto, anche remote, con notevole riduzione e risparmio nei cablaggi degli I/O.
- Frazionamento dell'impianto con conseguente semplificazione dello sviluppo progettuale e delle operazioni di manutenzione e riparazione.

I dispositivi sono connessi tra loro mediante cavi dotati di pochi poli conduttori per formare una rete distribuita anche su elevate distanze. Il protocollo Modbus prevede diverse tipologie di connessione fisica tra i dispositivi. In particolare i mezzi di connessione più diffusi sono:

- Rete RS485 - Protocollo Modbus seriale RTU
- Rete Ethernet - Protocollo Modbus TCP/IP

Il tipo di connessione adottata definisce quello che normalmente si chiama "Physical layer" del protocollo ossia l'hardware sul quale avviene lo scambio delle informazioni. Qui verrà trattato il protocollo Modbus su rete seriale RS485. Una seriale di questo tipo necessita di cavi con solo due poli per far transitare dati da un qualsiasi dispositivo all'altro.
Il successivo livello del protocollo, il "Data Link layer", è invece di tipo software e comprende tutte le specifiche relative allo scambio dei frames di dati (sequenze di bytes) tra un dispositivo e l'altro. Questa parte prescinde dal contenuto informativo dei bytes e si occupa esclusivamente del loro invio sul bus di campo, del controllo delle temporizzazioni e degli errori mediante checksum.
Il protocollo Modbus è di tipo Master/Slave e quindi nella rete è presente sempre e solo un dispositivo Master che gestisce la comunicazione nei confronti di uno o più dispositivi Slave. Ogni scambio di informazioni è originato dal Master il quale invia un frame di bytes sul bus di campo contente una particolare richiesta, normalmente un comando di lettura o di scrittura delle informazioni contenute in uno degli Slave. Tutti gli Slave sono normalmente in ricezione ed ascoltano le richieste del Master. Solo lo specifico Slave interrogato cattura le informazioni inviate dal Master, provvede all'esecuzione del comando e risponde al Master inviando a sua volta le proprie informazioni sulla rete.
Le modalità, secondo le quali le possibili richieste del Master e le relative risposte degli Slave sono codificate all'interno dei frames di comunicazione, sono definite dal successivo livello del protocollo, detto "Application protocol", anch'esso di tipo software. E' con questo livello finale del protocollo che interagisce l'applicazione specifica del dispositivo, nel caso del Master solitamente il programma di automazione del PLC, nel caso degli Slave il firmware di gestione degli specifici I/O del dispositivo. Questo livello costituisce l'interfaccia del protocollo con tutta la parte restante del software del dispositivo.
La descrizione del protocollo frazionata in successivi livelli di implementazione è definita dal modello ISO/OSI frequentemente utilizzato per rappresentare i protocolli di comunicazione:

Layer	ISO/OSI Model	
7	Application	Modbus Application Protocol
6	Presentation	Not used
5	Session	Not used
4	Transport	Not used
3	Network	Not used
2	Data Link	Modbus Serial Line Protocol
1	Physical	EIA/TIA-485 standard

Figura 4.1: Layer ISO

4.2 Protocollo Modbus su RS485 - Physical layer

Il protocollo Modbus su rete seriale RS485 è sicuramente una delle implementazioni più diffuse grazie alla sua semplicità, economia ma anche affidabilità in ambiente industriale. Essa è realizzata mediante un cavo bifilare che unisce in parallelo tutti i dispositivi presenti sulla rete. Essendo un protocollo di tipo Master/Slave saranno presenti sempre un solo dispositivo Master ed uno o più dispositivi Slave:

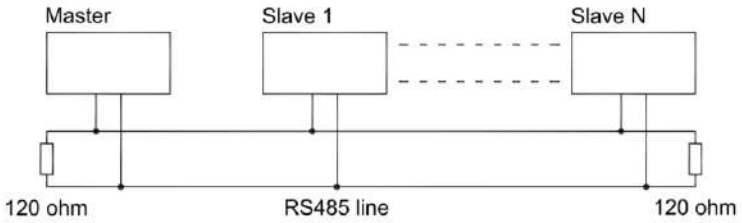

Figura 4.2: Modbus

La connessione bifilare è la più economica possibile ma al tempo stesso offre delle ottime prestazioni in termini di velocità di comunicazione ed immunità ad eventuali segnali di disturbo elettromagnetico. Questo grazie alla particolare tecnica di utilizzo dei segnali elettrici applicati alla coppia di fili che verrà descritta nel seguito. La comunicazione dei bits 0/1 tra i dispositivi avviene applicando sui due fili della coppia una tensione continua di piccola entità la cui polarità cambia in funzione del livello logico 0/1 da trasmettere:

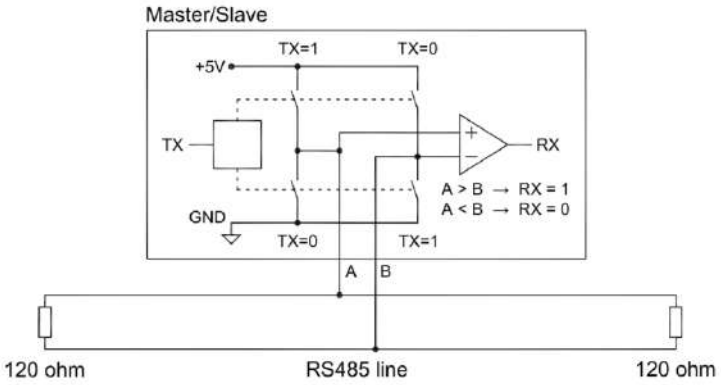

Figura 4.3: Modbus

Normalmente, in assenza di trasmissione da parte del dispositivo, tutti quattro gli interruttori del blocco di trasmissione sono OFF e quindi entrambi i conduttori della linea RS485 non sono connessi ad alcun potenziale elettrico. Tuttavia è preferibile non lasciare la linea fluttuante e per questo si applica sempre un potenziale di default, positivo su A e negativo su B. Questa polarizzazione è ottenuta collegando due resistenze, una tra l'alimentazione +5V ed il segnale A ed un'altra tra il segnale B ed il riferimento GND. Quando un dispositivo deve trasmettere dei bits, esso prende momentaneamente il controllo della linea accendendo i propri interruttori a due a due in diagonale. Per trasmettere uno 0 logico la linea viene forzata con A=GND e con B=+5V, mentre per trasmettere un 1 logico la linea viene forzata con A=+5V e con B=GND. In questo modo la tensione misurata tra il conduttore A ed il conduttore B sarà +5V oppure -5V rispettivamente per trasmettere il bit 1 oppure il bit 0.
Ogni dispositivo comprende anche un blocco di ricezione che può determinare la polarità del segnale A relativamente a quella del segnale B. Per questo l'interfaccia di ricezione è in grado di misurare la differenza di tensione tra il polo A ed il polo B. Se tale differenza è positiva (A > B) la ricezione corrisponde allo stato logico 1, se la differenza è negativa (A < B) lo stato logico che si sta ricevendo è 0. Il blocco di ricezione permette quindi di eseguire il processo inverso, decodificando l'informazione 0/1 precedentemente codificata con il blocco trasmettitore.
NOTA: il blocco trasmettitore è stato rappresentato idealmente con degli interruttori. In realtà questi interruttori sono realizzati con transistors in grado di commutare i segnali ad elevata velocità. Una vasta gamma di apposti circuiti integrati sono stati sviluppati come driver di interfaccia RS485 inserendo al loro interno sia la parte di trasmissione che di ricezione.
Questa tecnica di pilotaggio della linea seriale da origine al nome di "segnale differenziale" riferendosi a quello applicato ai fili A/B della linea. Si noti che in una connessione differenziale non è strettamente necessario collegare il riferimento comune GND di un dispositivo con quello degli altri come normalmente avviene quando si connettono dei segnali tra due apparecchiature distinte. Infatti nel caso del segnale differenziale quello che conta non è la

tensione di A o B rispetto a GND ma la differenza relativa tra A e B. I segnali differenziali sono molto utilizzati, come ad esempio nella connessione USB, proprio per le loro elevate prestazioni abbinate alla massima semplicità del supporto fisico. Un grande vantaggio derivante dai segnali differenziali lo si ha realizzando il cavo mediante una coppia di conduttori tra loro strettamente intrecciati, ottenendo un elevato grado di uguaglianza dei due fili in termini di posizione fisica nello spazio. Una eventuale interferenza, causata dalla vicinanza della linea ai cavi di altri sistemi, indurrà una tensione di disturbo su entrambi i fili della coppia, sovrapponendo ad essi delle tensioni indesiderate con la stessa polarità (positiva o negativa) rispetto al riferimento:

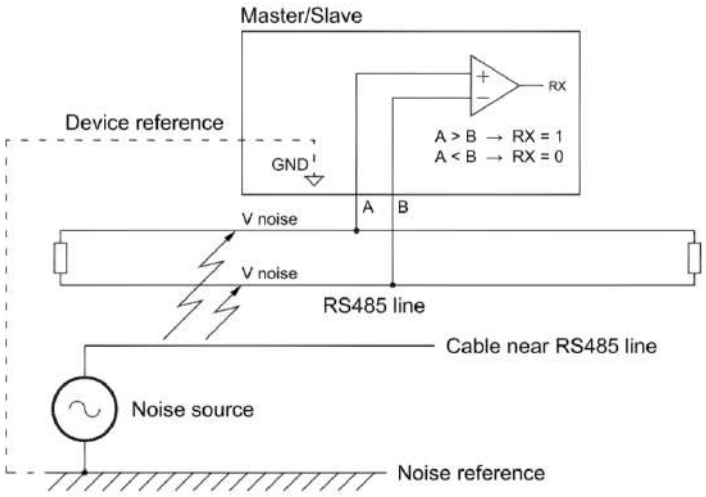

Figura 4.4: Modbus with noise

I ricevitori di ogni dispositivo, che effettuano una valutazione della differenza di tensione tra i fili A e B, saranno quindi in grado di eliminare questa componente di disturbo proprio perché di uguale segno ed entità su entrambi i fili. Per questo è importante che l'eventuale disturbo interferisca allo stesso modo con entrambi i conduttori della linea.
Negli schemi precedenti sono state evidenziate anche due resistenze da 120 ohm poste agli estremi della linea RS485. Lo scopo di queste resistenze è di definire l'impedenza di carico nelle parti terminali della linea per evitare che, nella propagazione dei segnali lungo i cavi, si generino, in corrispondenza alle estremità aperte, delle riflessioni indietro con conseguente alterazione della forma d'onda dei segnali stessi.
Questo fenomeno è tanto più evidente quanto più alte sono le frequenze dei segnali ma già alle più basse frequenze di utilizzo del bus di campo si possono notare questi effetti. Quindi l'inserimento delle resistenze di terminazione della linea è richiesto tassativamente in tutte le applicazioni.
Oltre alla corretta terminazione della linea differenziale è molto importante che la connessione di questa ai vari dispositivi avvenga seguendo un unico percorso lineare ossia senza

diramazioni. Occorre prestare attenzione a questa regola anche nel punto di connessione della linea ai due appositi morsetti di ogni dispositivo. La linea di cavo va interrotta solo in corrispondenza della morsettiera di ogni dispositivo, serrando il cavo in arrivo con quello in partenza direttamente nei stessi morsetti. Solo il primo ed ultimo dispositivo, indipendentemente se Master o Slave, devono anche avere la connessione alla resistenza di terminazione. Questa resistenza è solitamente incorporata nei dispositivi ed abilitabile da un'apposito interruttore.
La rete seriale RS485 permette di realizzare impianti distribuiti posizionando i vari dispositivi anche a notevoli distanze tra loro. La lunghezza massima della connessione dipende da molti fattori come la velocità di comunicazione (generalmente da 9600 a 115200b/s per il protocollo Modbus), dalla qualità dei cavi e dalle caratteristiche del driver hardware dei dispositivi. Utilizzando cavi specifici per le linee differenziali, la distanza massima raggiungibile è di 1200m con velocità di comunicazione nel range da 10Kb/s a 100Kb/s mentre con velocità crescenti fino a 1Mb/s la distanza scende proporzionalmente fino a 120m. Questi sono dati caratteristici specificati per i soli cavi ma esistono tanti altri fattori, dipendenti dalla particolare applicazione, come l'ambiente circostante e l'accuratezza dell'installazione, che possono portare ad una significativa riduzione della distanza massima raggiungibile per una determinata velocità di comunicazione. Il numero massimo di dispositivi collegabili alla rete RS485 dipende principalmente dalle caratteristiche dei circuiti integrati driver adottati nelle interfacce dei singoli nodi. La specifica EIA/TIA-485 richiede che il circuito trasmettitore sia capace di pilotare fino a 32 unità di carico dove una unità di carico (UL) equivale ad una impedenza di circa 12Kohm. Le tecnologie dei circuiti integrati driver, in continua evoluzione in termini di prestazioni, hanno permesso di introdurre nel mercato dispositivi capaci di corrispondere a frazioni dell'unità di carico per cui il numero massimo di dispositivi può essere maggiore di 32. Per esempio componenti con 1/2UL permettono di connettere tra loro fino a 64 dispositivi, mentre con 1/8UL si può arrivare a 256 nodi. Si consideri comunque che questi dati si basano esclusivamente sulla valutazione dell'unità di carico, senza considerare le resistenze di polarizzazione del bus e tanti altri elementi della reale applicazione, in particolare la qualità e lunghezza delle connessioni ed anche la velocità di comunicazione, che possono ridurre in modo sostanziale il numero massimo dei dispositivi.

4.3 Protocollo Modbus su RS485 - Data Link layer

Il “Data Link layer” rappresenta il primo dei livelli di tipo software del protocollo Modbus e definisce come i bytes vengano scambiati tra un dispositivo e l’altro prescindendo tuttavia dal loro specifico significato. Questo livello prevede due diversi metodi di trasmissione delle informazioni. La trasmissione di tipo RTU comporta uno scambio di frames binari dove i bytes possono assumere tutti i possibili valori 0255. Non potendo identificare l’inizio ed il termine di ogni frame di comunicazione in base a valori specifici dei bytes (caratteri di sincronizzazione) è necessario ricorrere alla gestione di timers che controllino l’esatta temporizzazione della trasmissione. In alcuni sistemi la gestione di queste temporizzazioni, i cui tempi possono essere anche molto brevi alle più elevate velocità di comunicazione, può essere onerosa e costituire un problema. La trasmissione ASCII invece è ottenuta codificando il valore di ogni byte dell’informazione con la sequenza di due caratteri di testo esadecimali (cifre 09, AF). Il protocollo ASCII è meno efficiente del protocollo RTU in quanto ogni byte di informazione è codificato da due caratteri. Tuttavia questa codifica consente di riservare dei caratteri speciali allo scopo di essere utilizzati come indicatori (caratteri di sincronizzazione) di inizio e fine frame. Questo semplifica notevolmente la gestione della trasmissione in particolare in sistemi più lenti o non dotati di particolari dispositivi hardware per implementare il controllo di temporizzazione necessario invece al protocollo RTU. In questa serie di articoli sarà trattato solo il protocollo seriale Modbus RTU. La comunicazione su rete seriale RS485 è di tipo half-duplex e ciò significa che solo un dispositivo alla volta può trasmettere informazioni sul bus. Questo richiede che la trasmissione effettuata dai vari nodi della rete sia coordinata in modo tale che non ci siano sovrapposizioni del pilotaggio della linea differenziale da parte di più di un dispositivo alla volta. Il coordinamento della comunicazione è ottenuto assegnando ad uno dei nodi il ruolo di Master mentre a tutti gli altri quello di Slave. In genere la funzione di Master viene assunta dal dispositivo di controllo della logica dell’impianto, ad esempio il PLC, mentre tutti i dispositivi periferici e di espansione degli I/O operano come Slave. Il Master controlla la comunicazione nel senso che è l’unico autorizzato ad iniziare uno scambio di informazioni. Periodicamente o su necessità, il Master inizia la comunicazione con uno degli Slave, inviando un pacchetto di bytes sul bus contenente l’indirizzo dello Slave interessato, la funzione da eseguire, gli eventuali dati associati alla funzione ed il checksum di controllo del pacchetto:

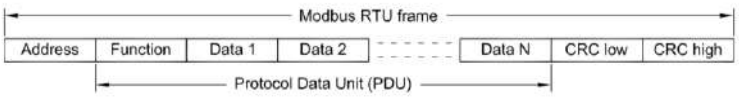

Figura 4.5: Modbus RTU frame

Il campo Address è costituito da un solo byte contenente l’indirizzo dello Slave interessato.

I valori ammessi dal protocollo sono nel range 0247.
Anche per il campo Function è utilizzato un solo byte e questo contiene il codice comando della specifica richiesta allo Slave. Al codice funzione possono essere associati anche alcuni bytes di Dato specifici del comando fino ad un massimo di 252 bytes. I valori dei codici funzione ed il significato dei dati associati al comando saranno definiti nel livello successivo del protocollo (Application layer). L'insieme del campo funzione e del campo dati viene denominato Protocol Data Unit (PDU) e rappresenta la parte utile della comunicazione. Al termine del frame sono aggiunti due bytes contenenti la parte bassa e la parte alta della word di checksum (CRC) calcolata su tutti i bytes precedenti. Questa word costituisce una firma relativa al contenuto del frame al fine di identificare eventuali errori nella trasmissione dei bytes. Tutti gli Slave, normalmente in "ascolto" sul bus, riceveranno il frame trasmesso dal Master ma solo quello indirizzato continuerà la comunicazione mentre gli altri ignoreranno la richiesta. Lo Slave interessato risponderà al Master inviando a sua volta un frame con la stessa struttura di quello appena descritto. Il Master, una volta terminato lo scambio dei frames con uno Slave, potrà iniziare una nuova sequenza di comunicazione con lo stesso Slave oppure con uno degli altri procedendo allo stesso modo:

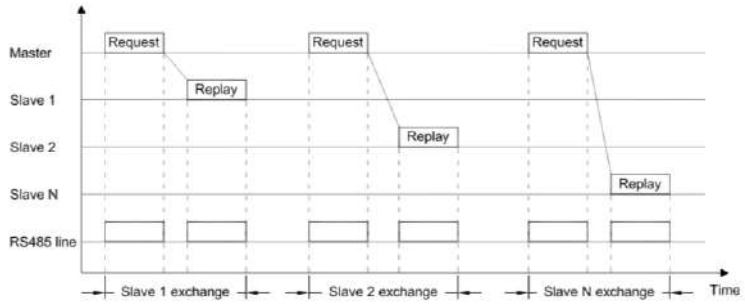

Figura 4.6: Modbus rs485

Lo Slave interrogato deve rispondere entro il più breve tempo possibile anche per non rallentare le operazioni di comunicazione con tutti gli altri Slave. Inoltre è importate gestire nel Master un tempo massimo prefissato (Timeout) entro il quale lo Slave deve rispondere per evitare di bloccare la comunicazione nel caso di Slave assente o guasto. Il protocollo Modbus non specifica come eseguire l'aggiornamento delle informazioni nei confronti degli Slave per cui è possibile che il Master esegua le richieste in modo arbitrario, a seconda delle necessità, oppure in modo periodico (polling) per mantenere costantemente aggiornato la stato delle risorse degli Slave. Ciascun byte del frame è trasmesso sul bus di campo RS485 mediante l'invio seriale di un totale di 11 bits, comprendendo il bit di start, il bit di parità ed il bit di stop:

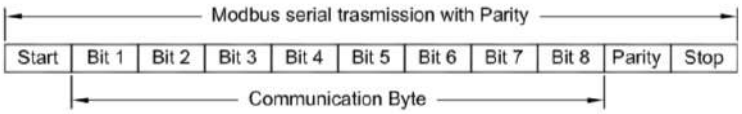

Figura 4.7: Modbus con parity

Il bit di start vale sempre 0 e costituisce un riferimento di sincronismo per inizializzare la ricezione di tutti i bits del carattere. Dopo il bit di start sono inviati gli 8 bits del byte da trasmettere iniziando dal bit meno significativo. Successivamente è trasmesso un bit di parità per il controllo degli errori su un bit del dato ed infine un bit di stop (valore fisso ad 1). Il protocollo Modbus permette di utilizzare una delle seguenti modalità per il bit di parità:

Parity	Rule for define the parity bit
No	Fixed to 1
Even	Total of bits (data + parity) equal to 1 is even
Odd	Total of bits (data + parity) equal to 1 is odd

Figura 4.8: Modbus parity

In realtà la specifica Modbus, anche se lascia piena libertà all'utilizzatore del dispositivo nella scelta della parità, obbliga al produttore di implementare nel dispositivo almeno la parità Even mentre raccomanda la disponibilità anche della configurazione "No parity". Si noti che, nel caso di "No parity" il bit di parità è comunque presente in decima posizione e fisso al valore logico 1. Considerato anche il bit di stop in undicesima posizione, il carattere trasmesso ha quindi sempre un totale di 11 bits, anche in assenza di parità, come se la configurazione "No parity" corrispondesse alla presenza di due bits di stop:

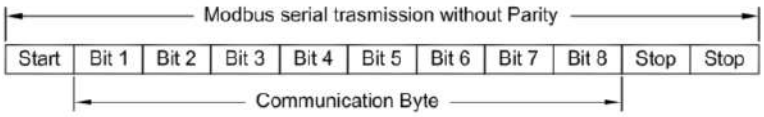

Figura 4.9: Modbus without parity

E' piuttosto comune, in varie implementazioni del protocollo Modbus, trovare una configurazione di bits del carattere al di fuori della specifica ufficiale. Infatti, nel caso di "No parity", alcuni dispositivi considerano la totale assenza del bit di parità portando a 10 il numero di bits totali del carattere. Nel protocollo Modbus RTU, tutti i bytes appartenenti allo stesso frame devono sottostare a delle rigide specifiche temporali, proprio perché, non disponendo di caratteri speciali da utilizzare come sincronismo, i pacchetti vengono riconosciuti ed isolati gli uni dagli altri in base esclusivamente alle loro temporizzazioni:

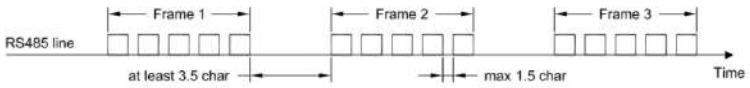

Figura 4.10: Modbus rs485 line

Ogni frame deve necessariamente avere tutti i propri bytes vicini tra loro con un tempo massimo, tra la fine di trasmissione di un byte e l'inizio di trasmissione del successivo, pari a 1.5 volte la durata del carattere stesso. Inoltre per distinguere un frame da quello adiacente occorre che, dopo la trasmissione di un frame, ci sia una pausa pari a 3.5 volte la durata di un carattere. Questa pausa a fine frame, abbinata alla "compattezza" dei bytes dello stesso frame, garantisce la possibilità di distinguere e separare tutti i frames tra loro. Un dispositivo in trasmissione dovrà quindi assicurare l'invio contiguo dei bytes del frame, per esempio preparando tutti i bytes fino al checksum in un buffer di trasmissione ed attivando successivamente l'invio dell'intero buffer sulla linea seriale. In ricezione i dispositivi dovranno "catturare" immediatamente tutti i bytes di un frame riattivando ad ogni byte ricevuto un timer (a 1.5 caratteri) per controllare la contiguità dei bytes ed un timer (a 3.5 caratteri) per verificare l'effettiva fine di un frame. Dopo la ricezione di tutto il frame verrà controllata la correttezza del checksum e la corrispondenza dell'indirizzo di Slave. Solo il livello successivo del protocollo (Application layer) analizzerà la correttezza della PDU ricevuta e in caso positivo provvederà a processarla. Riguardo al campo indirizzo del frame di comunicazione, il protocollo Modbus utilizza i valori nel range 1247 per comunicare con un solo specifico Slave alla volta (modalità Unicast) mentre l'indirizzo speciale 0 è utilizzato per la modalità Broadcast. Una richiesta del Master con indirizzo 0 viene processata da tutti gli Slave ma nessuno di questi provvederà a rispondere al Master proprio per evitare conflitti di trasmissione sul bus. Questa modalità è utilizzata per inviare lo stesso comando contemporaneamente a tutti gli Slave senza però ricevere risposta e quindi senza alcuna conferma dell'avvenuta esecuzione del comando da parte degli Slave. Infine gli indirizzi da 248 a 255 sono "riservati" e consentono al produttore di implementare delle specifiche funzionalità. Il campo terminale del frame è costituito da due bytes corrispondenti alla parte bassa e alta della word di checksum (CRC). Questo valore permette di verificare l'integrità dei bytes del pacchetto in quanto costituisce una firma relativa ai valori dei bytes che precedono il campo CRC. In trasmissione il checksum è calcolato su tutti i bytes dal primo (indirizzo) all'ultimo del campo Dati ed aggiunto al termine del frame. In ricezione viene calcolato il checksum su tutti i bytes del frame ricevuto (campo CRC compreso). In tal caso, solo un risultato pari a zero indica la correttezza del frame ricevuto. Per il calcolo del CRC si possono adottare due differenti approcci. Il primo è quello di un algoritmo completo che elabora i valori di tutti i bytes considerati e determina il valore a 16 bit del CRC. Il secondo è più semplice e veloce dal punto di vista del calcolo ma richiede l'utilizzo di una tabella di 256 bytes costanti precalcolate e questo su microprocessori con poca memoria programma può risultare più oneroso.

Per maggiori dettagli su questi algoritmi fare riferimento al documento ufficiale del protocollo Modbus seriale: http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf Durante la comunicazione dei frames possono verificarsi situazioni anomale come la ricezione di un frame incompleto oppure con checksum errato. In tal caso i dispositivi Slave devono scartare questi frames e non rispondere mentre il Master, se verifica degli errori in una risposta degli Slave, potrebbe decidere di ritentare lo scambio dei frames.

4.4 Protocollo Modbus su RS485 - Application layer

Il secondo ed ultimo livello software del protocollo Modbus, detto “Application layer“, si occupa della Protocol Data Unit (PDU) ossia della parte utile di informazione contenuta nei frames scambiati tra il Master e gli Slave. Infatti nel precedente livello (Data Link layer) sono definite solo le parti del protocollo relative alla trasmissione delle PDU senza considerarne il loro significato. La PDU contiene tutte le informazioni necessarie affinché il Master possa effettuare delle operazioni su uno Slave, per esempio la scrittura o la lettura di un parametro. Per questo i bytes di una PDU sono suddivisi in primo campo (di lunghezza 1 byte) che definisce la particolare operazione da svolgere (codice funzione) ed un eventuale secondo campo Dati specifico del comando da eseguire:

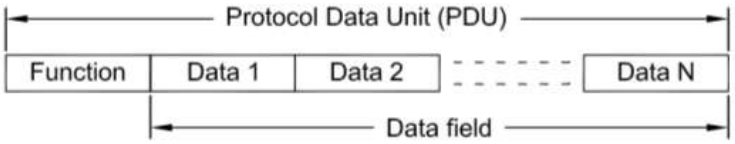

Figura 4.11: Modbus PDU

Il campo funzione contiene un valore predefinito per ogni comando implementato. Il protocollo Modbus prevede che solo i valori da 1 a 127 siano utilizzati per codificare il comando, lasciando i valori da 128 a 255 per la codifica dello stesso comando (ma con il bit più alto ad 1) allo scopo di segnalare, nella risposta dello Slave, un eventuale “eccezione” ossia un errore rilevato durante la sua esecuzione.
Il protocollo Modbus suddivide l’insieme dei codici numerici in due gruppi. I codici compresi nei campi 164, 7399 e 111127 sono utilizzati per le funzioni definite “pubbliche” ossia documentate ufficialmente e quindi univoche nell’ambito del protocollo. I rimanenti codici nei campi 6572 e 100110 sono liberi per l’implementazione di funzioni custom del protocollo.
Il significato e la lunghezza del campo Dati dipendono dallo specifico comando e questa parte può anche non essere presente. La lunghezza massima del campo Dati è di 252 bytes in quanto la somma complessiva dei bytes di un frame, secondo la specifica, è di 256 bytes. Il protocollo gestisce valori di tipo 16bit-word utilizzando la notazione “big-Endian” la quale richiede che il byte più significativo della word sia trasmesso per primo. Si noti che ciò non vale per l’invio del CRC che va trasmesso con il byte meno significativo per primo. I comandi standard del protocollo Modbus relativi alla lettura e scrittura dei valori delle risorse degli Slave fanno riferimento al concetto di “Oggetto” anziché all’indirizzo reale di memoria del dispositivo. Questo permette di svincolare l’indirizzamento delle risorse da parte del protocollo dalle effettive posizioni fisiche delle stesse nell’ambito della memoria interna dello Slave. Per questo ogni dispositivo dovrà definire quelli che sono gli oggetti accessibili al protocollo assegnandogli un indirizzo Modbus e dovrà di conseguenza occuparsi della loro

associazione con le variabili effettive interne nella propria memoria. Gli oggetti sono stati raggruppati in 4 tipologie differenti con evidente riferimento al tipo di risorse che offrivano i dispositivi del periodo nel quale era stato definito il protocollo:

Primary tables	Object type	Access type	Comments
Discrete Inputs	Single bit	Read-Only	Provided by an I/O slave
Coils	Single bit	Read-Write	Can be alterable by Master
Input Registers	16-bit word	Read-Only	Provided by an I/O slave
Holding Registers	16-bit word	Read-Write	Can be alterable by Master

Figura 4.12: Modbus tabella primaria

Le risorse di tipo “Discrete Inputs” fanno riferimento a valori di ingresso digitale (ON/OFF) mentre le risorse “Coils” a valori di uscita digitale quindi forzabili in scrittura ma al tempo stesso leggibili. Le risorse di tipo “Input Registers” e “Holding Registers” sono delle versioni analogiche (valori continui nel range 0-65535) delle precedenti ed utilizzate per ingressi/uscite analogiche o qualsiasi altro valore numerico del dispositivo come ad esempio un parametro di lavoro. Ai suddetti oggetti sono associati sia dei numeri identificativi che degli specifici indirizzi al fine di poter accedere a queste risorse tramite le PDU del protocollo. Si ricordi che l’indirizzo della risorsa non coincide necessariamente con l’indirizzo di memoria fisica all’interno del dispositivo. E’ compito di quest’ultimo associare, al volo, l’indirizzo Modbus con la variabile della memoria interna, utilizzando per esempio delle tabelle di corrispondenza. Queste tabelle definiscono quello che si chiama “Modello dati” dello specifico dispositivo e costituiscono una sorta di “manuale di riferimento del programmatore” in quanto elencano tutte le possibili risorse disponibili nello Slave e quelli che sono gli indirizzi ai fini del protocollo. Gli indirizzi Modbus degli oggetti, essendo definiti mediante una word, possono estendersi nell’intero range da 0 a 65535 mentre per il numero identificativo degli oggetti si utilizza la numerazione da 1 a 65536. Tuttavia la numerazione degli oggetti è solo formale mentre ciò che conta per l’accesso alla risorsa è esclusivamente l’indirizzo inserito nella PDU. Per ognuna delle 4 aree di suddivisione degli oggetti è possibile utilizzare l’intero range 065535 di indirizzamento in quanto sarà lo stesso codice funzione a definire a quale delle quattro aree si riferisce l’indirizzo. A questo modo esteso, che permette di indirizzare 65536 oggetti per ognuna delle quattro aree, si affianca un’altra convenzione definita originariamente dal protocollo e che è tuttora utilizzata. In questa convenzione si numerano le risorse in un range ristretto di valori da 0001 a 9999 aggiungendo un offset, multiplo di 10000, a seconda dell’area interessata:

Primary tables	Type prefix	Object Number	PDU Address
Coils	0x	00001÷09999	0000÷9998
Discrete Inputs	1x	10001÷19999	0000÷9998
Input Registers	3x	30001÷39999	0000÷9998
Holding Registers	4x	40001÷49999	0000÷9998

Figura 4.13: Modbus tabella primaria[1]

In questo modo è il numero di oggetto a definire a quale delle quattro aree si riferisce e conseguentemente con quale codice funzione eseguire le relative operazioni.
I codici funzione standard predefiniti dalla specifica Modbus sono pochi e buona parte di questi sono usati per leggere o scrivere le variabili di tipo bit e le variabili di tipo word:

Function code	Related area	Operation
1	Coils	Read up to 2000 contiguous memory bits
2	Discrete Inputs	Read up to 2000 contiguous input bits
3	Holding Registers	Read up to 125 contiguous memory words
4	Input Registers	Read up to 125 contiguous input words
5	Single Coil	Write one memory bit
6	Single Register	Write one memory word
15	Coils	Write up to 1968 contiguous memory bits
16	Holding Registers	Write up to 123 contiguous memory words
23	Holding Registers	Read up 125 and write up 121 memory words

Figura 4.14: Modbus tabella funzioni

Function code 1 - Read Coils			
Request	Function code	1 byte	0x01
	Starting address	2 bytes	0x0000 to 0xFFFF
	Quantity of Coils	2 bytes	1 to 2000 (0x001 to 0x7D0)
Replay	Function code	1 byte	0x01
	Bytes count	1 byte	N/8 or N/8+1 (N is the Quantity of Coils)
	Coils status	n bytes	n=N/8 or n=N/8+1 if (remainder of N/8) ≠ 0
Notes	The first bit addressed is in the bit 0 position of the first byte of replay. If N is not a multiple of eight the remaining bits in the last byte are 0 padded.		

Figura 4.15: Modbus Funzione 1

Function code 2 - Read Discrete Inputs			
Request	Function code	1 byte	0x02
	Starting address	2 bytes	0x0000 to 0xFFFF
	Quantity of Inputs	2 bytes	1 to 2000 (0x001 to 0x7D0)
Replay	Function code	1 byte	0x02
	Bytes count	1 byte	N/8 or N/8+1 (N is the Quantity of Inputs)
	Inputs status	n bytes	n=N/8 or n=N/8+1 if (remainder of N/8) ≠ 0
Notes	The first bit addressed is in the bit 0 position of the first byte of replay. If N is not a multiple of eight the remaining bits in the last byte are 0 padded.		

Figura 4.16: Modbus Funzione 2

Function code 3 - Read Holding Registers			
Request	Function code	1 byte	0x03
	Starting address	2 bytes	0x0000 to 0xFFFF
	Quantity of Registers	2 bytes	1 to 125 (0x01 to 0x7D)
Replay	Function code	1 byte	0x03
	Bytes count	1 byte	2 x N
	Registers value	2N bytes	N is the Quantity of Holding Registers

Figura 4.17: Modbus Funzione 3

Function code 4 - Read Input Registers			
Request	Function code	1 byte	0x04
	Starting address	2 bytes	0x0000 to 0xFFFF
	Quantity of Registers	2 bytes	1 to 125 (0x01 to 0x7D)
Replay	Function code	1 byte	0x04
	Bytes count	1 byte	2 x N
	Registers value	2N bytes	N is the Quantity of Input Registers

Figura 4.18: Modbus Funzione 4

Function code 5 - Write Single Coil			
Request	Function code	1 byte	0x05
	Coil address	2 bytes	0x0000 to 0xFFFF
	Coil value	2 bytes	0x0000 for OFF or 0xFF00 for ON
Replay	Function code	1 byte	0x05
	Coil address	2 byte	0x0000 to 0xFFFF
	Coil value	2 bytes	0x0000 for OFF or 0xFF00 for ON

Figura 4.19: Modbus Funzione 5

Function code 6 - Write Single Register			
Request	Function code	1 byte	0x06
	Register address	2 bytes	0x0000 to 0xFFFF
	Register value	2 bytes	0x0000 to 0xFFFF
Replay	Function code	1 byte	0x06
	Register address	2 byte	0x0000 to 0xFFFF
	Register value	2 bytes	0x0000 to 0xFFFF

Figura 4.20: Modbus Funzione 6

Function code 15 - Write Coils			
Request	Function code	1 byte	0x0F
	Starting address	2 bytes	0x0000 to 0xFFFF
	Quantity of Coils	2 bytes	1 to 1968 (0x001 to 0x7B0)
	Bytes count	1 byte	N/8 or N/8+1 (N is the Quantity of Coils)
	Coils status	n bytes	n=N/8 or n=N/8+1 if (remainder of N/8) ≠ 0
Replay	Function code	1 byte	0x0F
	Starting address	2 byte	0x0000 to 0xFFFF
	Quantity of Coils	2 bytes	1 to 1968 (0x001 to 0x7B0)
Notes	The first bit addressed is in the bit 0 position of the first byte of request. If N is not a multiple of eight the remaining bits in the last byte are 0 padded.		

Figura 4.21: Modbus Funzione 15

Function code 16 - Write Holding Registers			
Request	Function code	1 byte	0x10
	Starting address	2 bytes	0x0000 to 0xFFFF
	Quantity of Registers	2 bytes	1 to 123 (0x01 to 0x7B)
	Bytes count	1 byte	2 x N
	Registers value	2N bytes	N is the Quantity of Holding Registers
Replay	Function code	1 byte	0x10
	Starting address	1 byte	0x0000 to 0xFFFF
	Quantity of Registers	2 bytes	1 to 123 (0x01 to 0x7B)

Figura 4.22: Modbus Funzione 16

Function code 23 - Read/Write Holding Registers			
Request	Function code	1 byte	0x17
	Read start address	2 bytes	0x0000 to 0xFFFF
	Quantity to read	2 bytes	1 to 125 (0x01 to 0x7D)
	Write start address	2 bytes	0x0000 to 0xFFFF
	Quantity to write	2 bytes	1 to 121 (0x01 to 0x79)
	Write bytes count	1 byte	2 x N
	Write Registers value	2N bytes	N is the Quantity of writing Registers
Replay	Function code	1 byte	0x17
	Read bytes count	1 byte	2 x n
	Read Registers value	2n bytes	n is the Quantity of reading Registers

Figura 4.23: Modbus Funzione 23

Il protocollo Modbus prevede una gestione degli errori anche a livello del “Application layer” controllando che le richieste del Master siano lecite. Se uno Slave riceve una PDU con valori non compatibili con le proprie risorse, esso non eseguirà il comando richiesto e risponderà al Master con una PDU particolare (Exception) contenente nel campo Function lo stesso comando richiesto ma con il bit più significativo ad 1. Successivamente al campo Function, trasmetterà un campo dati, con lunghezza 1 byte, contenente il codice del particolare errore verificato:

Exception			
Replay	Function code	1 byte	The function code of request + 128
	Exception code	1 byte	0x00 to 0xFF

Exception code	Name of error	Comments
1	ILLEGAL FUNCTION	Function code is not valid or implemented.
2	ILLEGAL DATA ADDRESS	Object address is not valid for the Slave.
3	ILLEGAL DATA VALUE	Writing value is not valid for the addressed object.
4	SLAVE DEVICE FAILURE	Fatal error occurred during the requested operation.

Figura 4.24: Modbus eccezioni

Il protocollo Modbus definisce anche altri codici funzione utilizzati per funzioni di servizio, lettura dello stato, diagnosi ed identificazione del dispositivo ed anche ulteriori altri codici di errore.

4.5 Configurazioni RS-485

Realizzare una rete RS485 è possibile grazie ad una vasta gamma di ricetrasmettitori disponibili. I dispositivi RS485 sono sviluppati per venire incontro ai requisiti dell'operatività full-duplex o half-duplex, per fornire tensioni fino a 2.5V, protezione ESD fino a +/- 15kV, velocità di trasferimento dati da 250kbps a 12Mbps, operatività fail-safe in diverse configurazioni e operatività shut-down a basso consumo. Successivamente volevo mostrare un piccolo esempio di tempi di risposta modbus.

4.5.1 velocità di trasferimento e di risposta

Ad un certo punto, durante la progettazione di un sistema RS-485, deve essere determinata una velocità di dati appropriata per quel tipo sistema. Quindi, si rende necessario scegliere dei dispositivi RS485 in grado di funzionare a quella velocità. Optare per dispositivi veloci potrebbe essere una soluzione sempre conveniente, ma ci sono degli aspetti da valutare. Sebbene questi funzionino per dati sia ad alte che a basse velocità, l'utilizzo di dispositivi che siano più veloci di quanto necessario può causare emissioni di radiazioni più elevate e maggiore suscettibilità. Questi problemi vengono enfatizzati quanto sono più lunghi i cavi e alte le frequenze. La limitazione della velocità (slew rate) può aiutare i sistemi a bassa velocità in entrambe le condizioni menzionate sopra e si attua rallentando i limiti del segnale di RS-485 e poi riducendo i componenti ad alta frequenza del segnale.
Nella figura sia a che b mostrano segnali RS-485 che trasmettono a 250kbps (125 Khz). Il dispositivo è un MAX3485E non limitato è in grado di raggiungere velocità fino a 12Mbps. Nella figura b il dispositivo MAX3483E è limitato ad una velocità massima di 250kbps. Nella prima figura la trasformata di Fourier mostra componenti dalla frequenza oltre i 2Mbps.

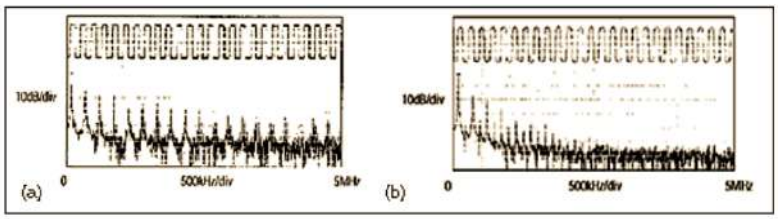

Figura 4.25: Rs485

Questi componenti ad alta frequenza sono necessari per produrre i bordi squadrati richiesti per velocità di dati più elevate ma la trasformata di Fourier mostra che i componenti a frequenza limitata sono più convenienti; un sistema progettato attorno a parti limitate nella velocità, tenderà ad emettere meno radiazioni e sarà meno suscettibile ad una errore di accoppiamento nelle terminazioni. La tabella successiva compara tre differenti componenti di RS-485: MAX3485 è il più veloce della famiglia (fino a 12Mbps), MAX3483 e limitato a 250Kbps e MAX3486 è il compromesso tra i due, con una velocità massima di 2.5Mbps.

	MAX3483	MAX3485	MAX3486
Maximum data rate	.25Mbps	12Mbps	2.5Mbps
Transition time	1200nS max	25nS max	60nS max
Propagation delay	1500nS max	35nS max	70nS max

Figura 4.26: Rs485

4.5.2 Full-duplex e Half-duplex

La RS-485 è progettata in modo che un solo trasmettitore su un doppino (twisted pair) può essere attivato per volta. Come mostrato in figura successiva, il Box A può trasmettere dati al Box B e viceversa, ma entrambi non possono svolgere questa funzione contemporaneamente.

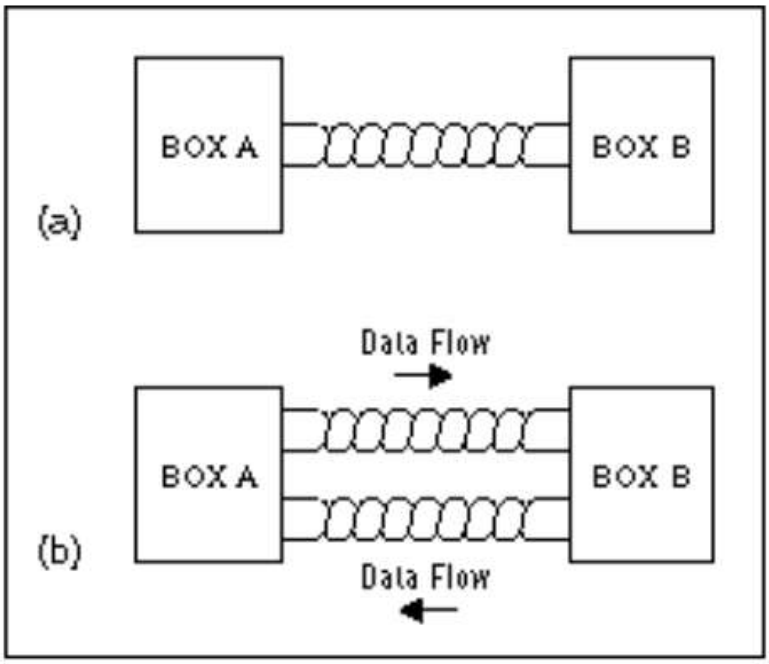

Figura 4.27: Rs485 schematic

Questa costrizione è definita half duplex. Da qui si deduce che un sistema full duplex permetta le comunicazioni in due direzioni nello stesso momento. Visto che una configurazione half duplex che essere in grado di trasmettere e ricevere sulla stessa twisted pair, i pin di output del trasmettitore e i pin di input del ricevitore devono essere collegati. Nella configurazione full-duplex, invece, il ricevitore e il trasmettitore sono separati. Inoltre la parte half duplex avrà sempre un pin driver-enable, perché il trasmettitore deve essere tri-stated (alta impedenza) mentre riceve i dati. I componenti full duplex posso avere o no un pin driver-enable, a secondo se sono progettati o meno per un sistema multidrop.
La configurazione full duplex a 4 fili può essere denominata RS422.

Capitolo 5

Protocollo di comunicazione I2c

5.1 Introduzione

In questo progetto la comunicazione I2C è stata utilizzata per visualizzare i dati misurati su un piccolo schermo OLED da 0,96 pollici. Vediamo ora come funzionano questo tipo di protocollo

5.2 I2C

E' un sistema di comunicazione seriale tra circuiti integrati. Il bus IC è composto da almeno un master ed uno slave. L'architettura nel nostro sistema è composto da un master e un slave.

Figura 5.1: I2C logo

5.2.1 Dettagli

Il protocollo I2C è stato creato dalla Philips Semiconductors nel 1982; la sigla, comunemente indicata anche con con I2C, sta per Inter-Integrated Circuit. Il protocollo permette la comunicazione di dati tra due o più dispositivi I2C utilizzando un bus a due fili, più uno che consiste nella massa, comune a tutti i dispositivi.
In tale protocollo le informazioni sono inviate seriale dal filo **SDA (Serial Data)**. Invece il filo **SCL (Serial Clock)** è utilizzato per il clock (per la presenza di questo segnale l'I2C è un bus sincrono)

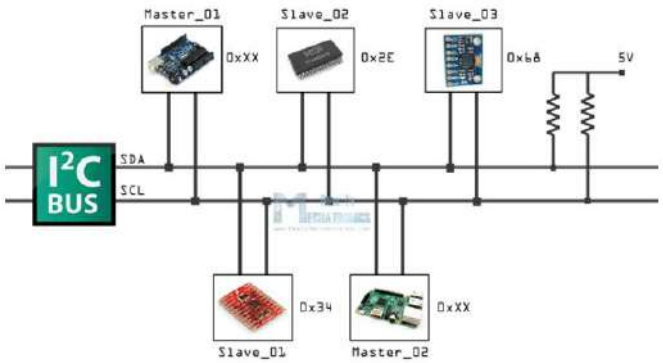

immagine presa dal sito
https://howtomechatronics.com/tutorials/arduino/how-i2c-communication-works-and-how-to-use-it-with-arduino/

Figura 5.2: Generic I2C architecture

In generale ci sono 4 distinti modi di operare:

- un master trasmette – controlla il clock e invia dati agli slave
- un master riceve – controlla il clock e riceve dati dallo slave
- uno slave trasmette – il dispositivo non controlla il clock e invia dati al master
- uno slave riceve – il dispositivo non controlla il clock e riceve dati dal master.

Il dispositivo **master** è semplicemente il dispositivo che controlla il bus in un certo istante; tale dispositivo controlla il segnale di Clock e genera i segnali di START e di STOP. I dispositivi **slave** semplicemente "ascoltano" il bus ricevendo dati dal master o inviandone qualora questo ne faccia loro richiesta.

5.2.2 Trasferimento dati

Una sequenza elementare di lettura o scrittura di dati tra master e slave segue il seguente ordine:

1. Invio del bit di START (S) da parte del master.
2. Invio dell'indirizzo dello slave (ADDR)7 ad opera del master.
3. Invio del bit di Read (R) o di Write (W), che valgono rispettivamente 1 e 0 (sempre ad opera del master).
4. Attesa/invio del bit di Acknowledge (ACK).
5. Invio/ricezione del byte dei dati (DATA).
6. Attesa/invio del bit di Acknowledge (ACK).
7. Invio del bit di STOP (P) da parte del master.

I passi 5 e 6 possono essere ripetuti così da leggere o scrivere più byte.

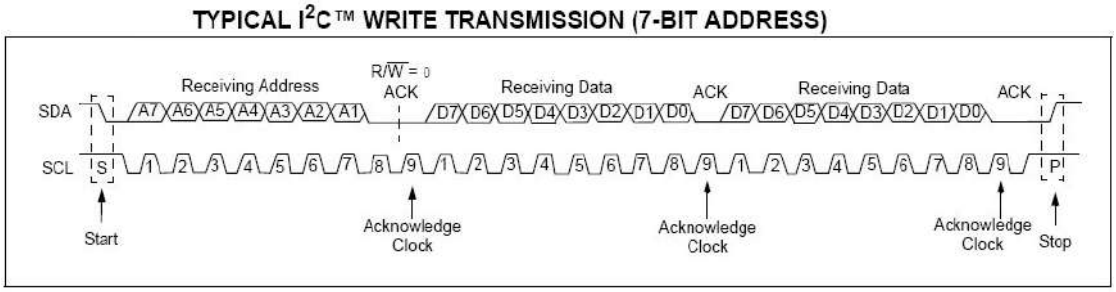

Figura 5.3: i2c comunication

5.2.2.1 Bit di Start e bit di Stop

Una comunicazione avviene all'interno di due sequenze di segnali che il master invierà sul bus IC;queste sono il comando di **START** e di **STOP**.La regola fondamentale da tenere sempre ben presente è che nella trasmissione dei dati, **il segnale SDA può cambiare stato soltanto mentre il segnale SCL è basso**, il valore viene letto durante il fronte di salita o di discesa del segnale SCL.

5.2.2.1.1 Segnale di start Prima di inviare il segnale di start, lo stato del bus IC si trova nella condizione in cui entrambi i segnali SDA ed SCL sono al livello logico alto. Per inviare il comando di start, il master pone a livello basso il segnale SDA mentre SCL è a livello logico alto, in questo modo segnala agli altri dispositivi che sta per iniziare una comunicazione.

5.2.2.1.2 Segnale si stop Prima di inviare il segnale di stop, sono stati trasferiti dei dati per cui lo stato del bus IC si trova nella condizione in cui entrambi i segnali SDA ed SCL sono al livello logico basso. Per inviare il comando di stop, il master pone prima a livello alto il segnale SCL e poi il segnale SDA. In questo modo segnala allo slave che la comunicazione è terminata

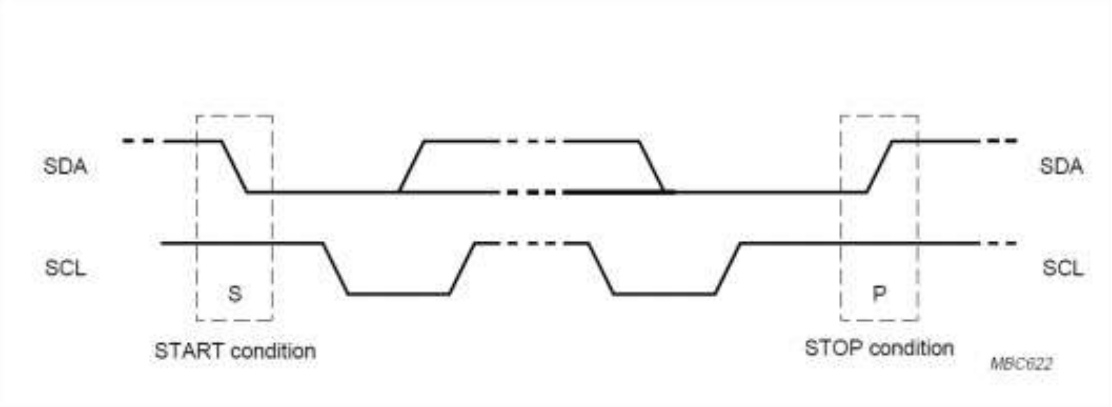

Figura 5.4: Start and Stop condition

5.2.2.2 Invio e ricezione di un bit

5.2.2.2.1 Invio di un bit da parte del master Poichè il segnale SDA può cambiare valore esclusivamente quando il segnale SCL è basso, il master in questo momento imposta il valore che deve trasmettere su SDA. Dopodichè manda alto il segnale SCL ed infine lo riporta basso.

5.2.2.2.2 Ricezione di un bit da uno slave In questo caso è lo slave che impone lo stato del segnale SDA, quindi il master deve rilasciare questo bus e poi leggerne il valore. Quando il master pone a livello basso il segnale SCL lo slave stabilizza SDA al valore che vuole comunicare; il master, quindi, pone alto SCL, legge SDA e ripone a livello basso SCL.

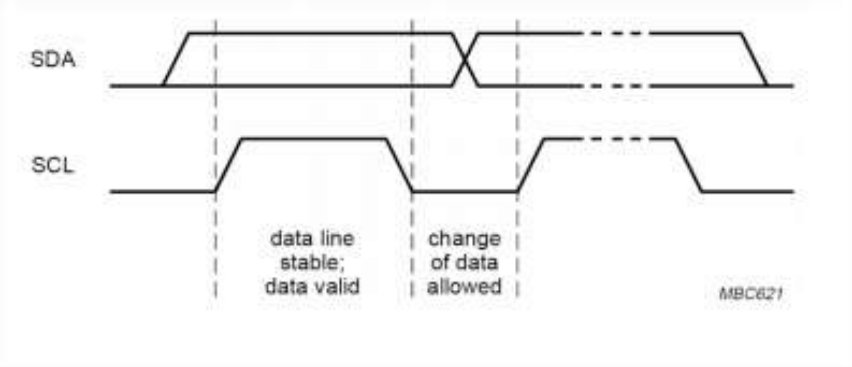

Figura 5.5: invio e ricezione bit i2c

5.2.2.3 Trasferimento di dati dal master allo slave

Il master invia la sequenza **S, ADDR, W,** quindi aspetta il bit di Acknowledge (A) dallo slave identificato dall'indirizzo inviato dal master. Se tale bit viene ricevuto correttamente dal master, questo invia il byte dei dati e aspetta un altro Acknowledge dallo slave. Dopo aver trasferito tutti i byte il master genera un bit di stop

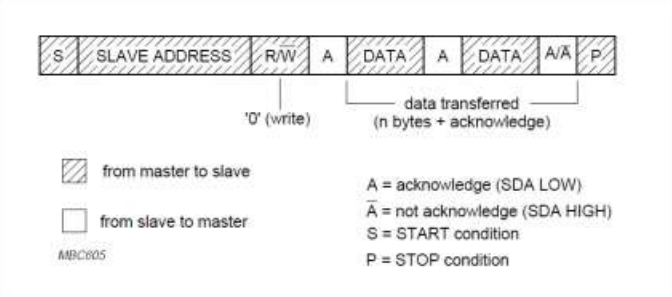

Figura 5.6: data transfer from master to slave

5.2.2.4 Trasferimento di dati dallo slave al master

Un processo analogo si verifica quando il master legge dei dati dallo slave, ma in tal caso, sarà inviato un **R** invece di un W. Dopo che i dati sono trasmessi dallo slave, sarà il master ad inviare il bit di Acknowledge; se invece il master non vuole altri dati, deve inviare un **not-Acnowledge**, che indica allo slave che deve liberare il bus. Questo permette al master di inviare il bit di stop. I dispositivi nascono generalmente con un proprio indirizzo che solitamente può essere cambiato dall'utente attraverso una opportuna sequenza reperibile nei data-sheet del produttore.

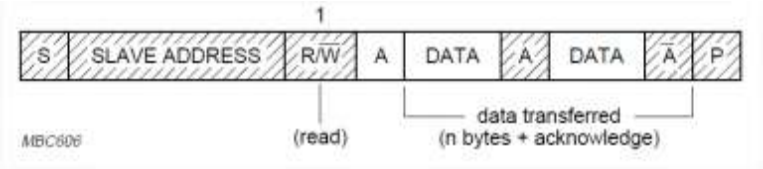

Figura 5.7: data transfer from slave to master

Capitolo 6

Protocollo di comunicazione SPI

6.1 Introduzione

Il protocollo SPI è quello utilizzato dal chip ADE9000 per comunicare con l'ESP8266. Vediamo a seguito brevemente come funziona.

6.2 Descrizione

Serial Peripheral Interface o SPI indica un sistema di comunicazione seriale tra un microcontrollore e altri circuiti integrati o tra più microcontrollori.
È un bus standard di comunicazione la cui trasmissione avviene tra un dispositivo detto master e uno o più slave. Il master controlla il bus, emette il segnale di clock, decide quando iniziare e terminare la comunicazione.

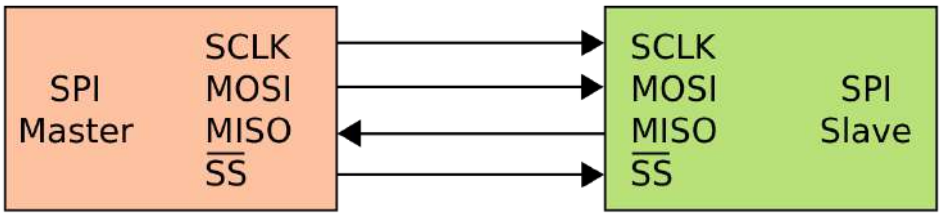

Figura 6.1: Spi comunication

Per quanto riguarda la velocità di scambio dei dati (in pratica la frequenza del clock) non vi è un limite minimo: possono mantenere se alimentati uno stato logico per un tempo indefinito) ma vi è un limite massimo che va determinato dai datasheet dei singoli dispositivi connessi e dal loro numero in quanto ogni dispositivo collegato al bus introduce sulle linee di comunicazione una capacità parassita. Il sistema di comunicazione di solito serve per

lo scambio di dati tra dispositivi montati "sulla stessa scheda elettronica" (o comunque tra schede elettroniche vicine tra di loro) in quanto non prevede particolari accorgimenti hardware per trasferire informazioni tra dispositivi lontani connessi con cavi soggetti a disturbi. Il sistema è comunemente definito a quattro fili. Con questo si intende che le linee di connessione che portano i segnali sono in genere quattro. Va però tenuto conto che vi deve comunque essere una connessione di riferimento (0 Vdc comunemente indicata con GND, nome che a rigore è improprio) e che fisicamente quindi i fili sarebbero cinque. Discorso analogo si può fare su altri bus seriali definiti a due fili (I2C).

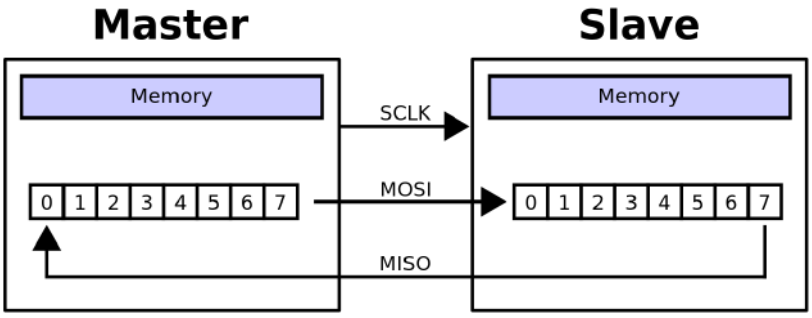

Figura 6.2: Spi master-slave

IL bus è di tipo seriale sincrono per la presenza di un clock che coordina la trasmissione e ricezione dei singoli bit e determina la velocità di trasmissione e anche full-duplex in quanto trasmissione e ricezione possono avvenire in contemporanea.

6.3 Segnali principali

Esso si basa su 4 segnali (si riportano i nomi dei segnali che possono variare a seconda del costruttore. Consultare il datasheet del componente che si intende utilizzare in caso di dubbi):

- SCLK - SCK: Serial Clock (emesso dal master)
- SDI – MISO – SOMI – DI - SO: Serial Data Input, Master Input Slave Output (ingresso per il master ed uscita per lo slave)
- SDO – MOSI – SIMO – DO – SI: Serial Data Output, Master Output Slave Input (uscita dal master)
- CS – SS – nCS – nSS – STE: Chip Select, Slave Select, emesso dal master per scegliere con quale dispositivo slave vuole comunicare

È facile confondersi nei nomi, spesso ambigui, sui segnali di trasferimento dei dati (ingressi e uscite). In ogni caso la consultazione dei datasheet dei dispositivi scelti dovrebbe chiarire

ogni dubbio. Le denominazioni MOSI (Master Output Slave Input), MISO (Master Input Slave Output) sembrano essere quelle che comportano meno equivoci. Il segnale SCLK è il clock seriale che scandisce gli istanti di emissione e di lettura dei bit sulle linee di dati. È un segnale emesso dal master ed è quindi quest'ultimo a richiedere di volta in volta la trasmissione di una "parola". Il segnale SDI/MISO è la linea attraverso cui il dispositivo (master o slave) riceve il dato seriale emesso dalla controparte. Sullo stesso fronte di commutazione del clock, il dispositivo emette, con la stessa cadenza, il suo output ponendo il dato sulla linea SDO/MOSI (linea di output di dato).

6.4 Slave controllati singolarmente

Vantaggi: comunicazione più rapida tra master e singoli slave. Svantaggi: necessità di avere un pin SS per ogni dispositivo slave. La linea CS è dedicata all'abilitazione del dispositivo slave da parte del master, il quale può abilitare un qualsiasi dispositivo slave connesso a trasmettere. La linea CS, normalmente attiva bassa, in caso di disabilitazione (livello logico alto) lascia il dispositivo slave con uscita in alta impedenza e quindi isolato completamente dal bus indifferentemente dall'esistenza del segnale di clock. Il numero di dispositivi slave che si possono connettere al bus è limitato esclusivamente dal numero di possibili linee di chip select gestibili dal dispositivo master. La frequenza di clock, e di conseguenza la velocità del bus, può raggiungere, con questo standard, livelli anche elevati nell'ordine delle decine di MHz ed anche oltre.

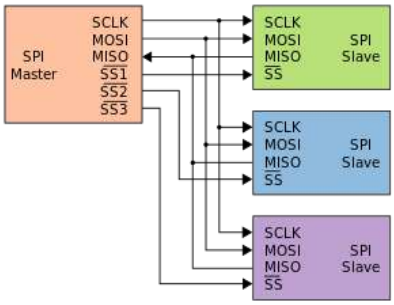

Figura 6.3: Spi slave singoli

6.5 Slave connessi a catena

Vantaggi: uso di un unico pin per selezionare i dispositivi. Svantaggi: minore velocità di aggiornamento dei singoli slave, il guasto di un elemento può causare un'interruzione del segnale negli altri dispositivi. In questo caso la linea SS serve per indicare agli slave quando campionare il dato presente nel registro (il master inietta i bit sulla linea MOSI, partendo dal bit più significativo da inviare all'ultimo slave. Così facendo, una volta trasmessi tutti i bit

destinati a tutti gli slave in questa sequenza, può segnalare agli slave che i dati in loro possesso nel registro di comunicazione sono effettivamente quelli destinati a loro). Se lo Slave Select fosse collegato, per esempio, fisso a massa, gli slave non potrebbero sapere se il dato presente nel loro registro di comunicazione è completo e da campionare, o magari destinato ad uno slave successivo, oppure non allineato alla dimensione del registro stesso. In questa configurazione è anche impossibile avere conflitto sulla linea MOSI del master (cosa che invece accadrebbe nella connessione di dispositivi slave controllati singolarmente dove una eventuale abilitazione di più slave creerebbe un conflitto sulla linea MISO creando problemi nella comunicazione (oltre ad un uso dei dispositivi con uscite in conflitto che non giova alla loro affidabilità nel tempo)

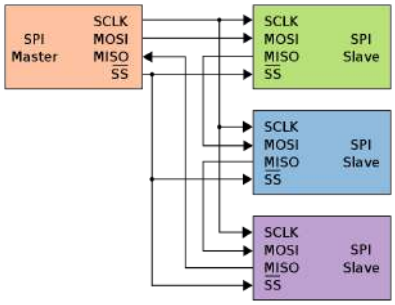

Figura 6.4: Spi daisy chain

6.6 Comunicazione

La trasmissione dei dati sul bus SPI si basa sul funzionamento dei registri a scorrimento (shift register). Ogni dispositivo sia master che slave è dotato di un registro a scorrimento interno i cui bit vengono emessi e, contemporaneamente, immessi, rispettivamente, tramite l'uscita SDO/MOSI e l'ingresso SDI/MISO. Il registro può avere dimensione arbitraria (ma uguale per i dispositivi master e slave) anche se usualmente ha la dimensione di 8 bit. Il registro a scorrimento è un'interfaccia completa mediante la quale vengono impartiti comandi e trasmessi dati che arrivano in modo seriale ma che internamente sono prelevati, a fine trasmissione, in modo parallelo. Ad ogni impulso di clock i dispositivi che stanno comunicando sulle linee del bus emettono un bit dal loro registro interno rimpiazzandolo con un bit emesso dall'altro interlocutore. La sincronizzazione è fatta sui fronti di clock di salita o di discesa regolata da 2 parametri impostabili. Molti microcontrollori dispongono di un hardware dedicato per la gestione dell'SPI programmabile nei dettagli. Questo non è strettamente necessario per poter comunicare con un dispositivo slave SPI (come ad esempio una memoria EEPROM): in ogni caso si potrà implementare nel firmware delle funzioni dedicate al colloquio (che forse risulteranno più lente e occuperanno più spazio nel firmware di quelle disponibili con un blocco hardware dedicato) ma che comunque permetteranno di comu-

nicare in modo efficiente con la periferica SPI, in quanto non vi è, nella temporizzazione dei dati, un limite di tempo massimo da rispettare (si tratta infatti di dispositivi statici, che possono cioè sospendere a tempo indeterminato la comunicazione senza per questo avere perdita di dati; in altre parole hanno dei limiti per quanto riguarda la velocità massima di trasmissione ma che possono funzionare anche in modo lento quanto si vuole).

Capitolo 7

Raspberry Compute Module 3

Questa scheda è pensata principalmente per un utilizzo industriale e per essere installata in sistemi embedeed. Questa soluzione basata sul processore Broadcom BCM2837 (1.2GHz, 64bit, quad-core, 1GB Ram) permette di avere un modulo molto più performante del precedente (presentato nel 2014 e basato sul processore BCM2835) mantenendo, in gran parte, la retro compatibilità col Compute Module 1. La versione standard del CM3 è dotata di un modulo di memoria eMMC da 4GB che permette di ospitare il sistema operativo mentre la versione Lite è sprovvista di tale memoria e quindi sarà necessaria una memoria SD esterna come avviene per la Raspberry PI 3.

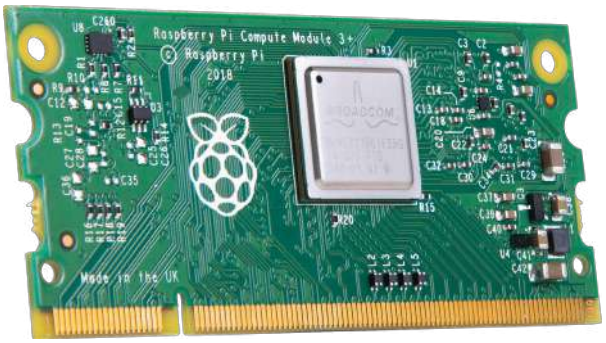

Figura 7.1: Raspberry CM

Per utilizzarla è necessaria una board apposita come si può vedere dalla figura successiva.

Figura 7.2: Board Raspberry

Per utilizzare questa raspberry è indispensabile scaricare il file immagine del sistema da caricare tramite il suo apposito software. Tutta la documentazione la si trova sul sito ufficiale raspberry dove vengono spiegati i vari passaggi da eseguire per un primo avvio in modo molto semplice e veloce. Visto la memoria di 4Gb interna e la non neccessità di avere un'interfaccia grafica ho installato la versione lite del sistema operativo raspbian che occupa solo 1,5 Gbyte.
Il Compute Module 3 è molto compatto, le sue dimensioni sono quelle di una classica memoria DDR2 SoDIMM. Questo form factor permette l'installazione del modulo tramite un socket SoDIMM montato sul proprio pcb.
Per risparmiare spazio e per avere un costo molto inferiore alla Raspberry PI 3, nel modulo non sono stati implementati i circuiti wireless, ethernet e i vari connettori come HDMI, camera e DSI. Molti pin del SoC BCM2837 sono collegati direttamente ai pin del modulo SoDIMM. E' necessaria una certa accortezza nell'interfacciamento del modulo alla circuiteria esterna e nell'implementazione delle funzionalità wifi e ethernet.

7.1 Primo avvio

Una volta avviata per la prima volta e configurato subito l'adapter per la rete Ethernet e impostato un indirizzo Ip fisso in modo da poter accedere via ssh da remoto (link con una buona guida ***https://www.ionos.it/digitalguide/server/configurazione/raspberry-pi-assegnazione-di-un-indirizzo-ip-fisso/)***, è fondamentale eseguire alcuni passi per l'aggiornamento della raspberry e l'aggiunta di alcune librerie per lo sviluppo del programma.
Qui riportati i vari comandi:

- 1) aggiornamento e installazione aggiornamenti tramite ***sudo apt-get update sudo apt-get upgrade***
- 2) installazione pipe ***sudo apt-get install python-pip***
- 3)installazione libreria modbus tramite questa guida ben fatta con libreria direttamente scaricabile da github ***https://github.com/riptideio/pymodbus***
- 4) ovviamente per l'installazione di librerie git serve avere installato il comando git ***sudo apt-get install git***
- 5) prossimo passo è l'installazione della libreria gpio che viene usata nel nostro caso per gestire input del pulsante che genera interrupt ***sudo apt install raspi-gpio***
- 6) importantissima è anche l'aggiunta di alcuni pacchetti che servono al sistema per far girare tutto il programma scritto in modo corretto e funzionante ***pip install –upgrade setuptools python -m pip install –upgrade pip sudo apt-get install libffi-dev sudo apt-get install libtool***
- 7) per quanto riguarda la libreria per gestire il piccolo display Oled occorre installare la libreria dal link ***https://github.com/adafruit/ġit*** e successivamente anche un pacchetto pil ***sudo apt-get install python-pil***
- 8) gli ultimi passaggi sono l'installazione di mysql e grafana di cui farò un breve capitolo in cui ne parlo brevemente

Capitolo 8

ESP8266 NODE MCU

Basata sul modulo transceiver Wi-Fi ESP8266 e sul chip convertitore USB CH340, questa compatta board (Open Source) di sviluppo e prototipazione è ideale per applicazione IoT (Internet of Things). Il modulo Wi-Fi è compatibile con lo standard 802.11 b/g/n a 2,4 GHz, dispone di stack TCP/IP integrato, potenza di uscita di 19.5 dBm, interfaccia dati (UART / HSPI / I2C / I2S / Ir Remote Control GPIO / PWM) e antenna su PCB. Dispone inoltre di connettore micro USB e pulsante di reset. Programmabile con IDE Arduino, include interpreti per l'elaborazione di comandi per linguaggi come LUA.

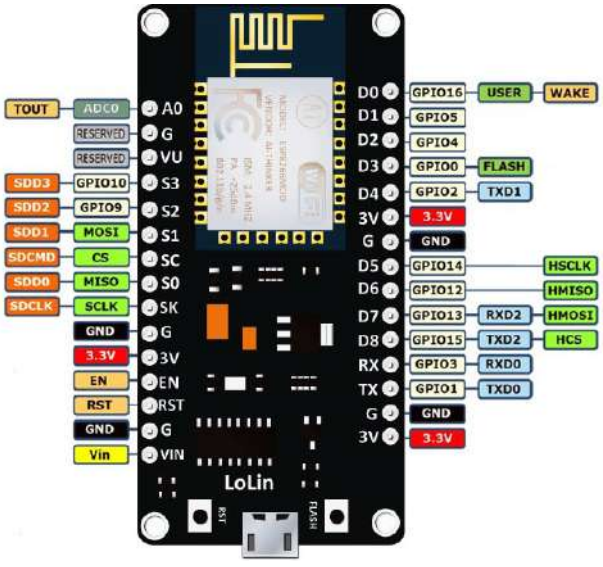

Figura 8.1: ESP8266

8.1 Caratteristiche

- Modello: ESP8266-12E
- Standard Wireless: 802.11 b/g/n

- Range di frequenza: 2,4 GHz - 2,5 GHz (2400M-2483.5M)
- Modalità Wi-Fi: Station / SoftAP / SoftAP+station
- Stack: TCP/IP integrato
- Potenza di uscita: 19.5dBm in modalità 802.11b
- Interfaccia dati: UART / HSPI / I2C / I2S / Ir Remote Control GPIO / PWM
- Supporta la modalità di protezione: WPA / WPA2
- Crittografia: WEP / TKIP / AES
- Alimentazione: da 4,5 VDC a 9 VDC (VIN) o tramite connettore micro USB
- Consumo: con Wi-Fi in trasmissione continua circa 70 mA (200 mA MAX) - in standby < 200A
- Temperatura di lavoro: da -40C a +125C
- Dimensioni (mm): 58x31,20x13
- Peso: 10 grammi

8.2 Configurazione

Il modo più semplice per la programmazione è quello di utilizzare Arduido IDE il quale è molto semplice. Importante è fare alcune modifiche come l'aggiunta del driver ch340 tramite: "File > Impostazioni" si aprirà la finestra "Impostazioni".

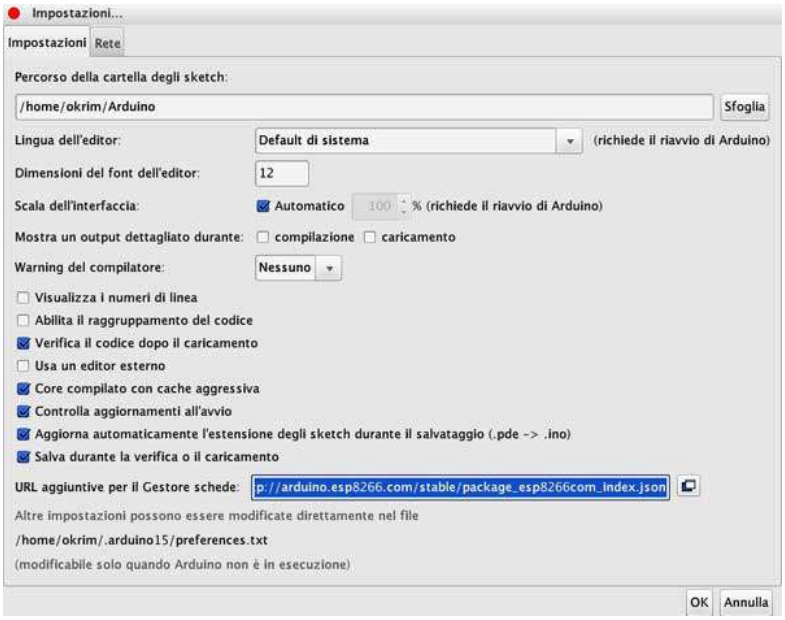

Figura 8.2: Impostazioni Arduino IDE

Dopodichè va inserito questo URL “http://arduino.esp8266.com/stable/package_esp8266com_index.json” nella casella di testo inferiore della finestra (“URL aggiuntive per il gestore di schede”). Clicca OK per continuare.

Dopo aver fatto questo primo passaggio possiamo installare la scheda, cliccando dalla schermata principale dell’IDE “Strumenti > Scheda > Gestore schede...” si aprirà la finestra “Gestore schede” e cercare ’esp8266’. Una volta installata potremo cominciare la nostra programmazione in modo molto semplice.

Capitolo 9

Software e sistemi utilizzati

9.1 Arduino IDE

Figura 9.1: Arduino IDE

La sigla IDE se vi state chiedendo cosa sia è l'acronimo di Integrated Development Environment.L' IDE più essere scaricato in modo gratuito sul proprio computer scaricandolo dalla pagina ufficiale di Arduino molto velocemente.
L'IDE di Arduino comunicherà con le varie schede attraverso la porta USB del computer permettendo il caricamento delle nuove istruzioni che devono essere eseguite dalla scheda ma anche fornendo tutta una serie di comunicazioni da e verso la scheda attraverso una funzione chiamata monitor seriale.
L'IDE di Arduino è in grado di comunicare e programmare tutte le diverse schede della famiglia Arduino (ed i suoi cloni) ma anche schede totalmente diverse come le Nodemcu o le 8266 o le D1. E' quindi un software molto potente diventato ormai la soluzione più comune per la programmazione delle schede IOT (internet of things). Una volta aperto il programma vi troverete davanti ad una schermata simile a questa (la mia ha già un po' codice presente. Nell'ampia parte superiore a sfondo bianco c'è l'area dove scrivere il codice

Nella parte inferiore più piccola a sfondo nero c'è invece l'area dove il sistema comunica all'utente su eventuali errori nel codice (la scritte son in arancione) o informazioni relative alla dimensione e l'upload del codice (scritte in bianco).

9.2 Putty

Figura 9.2: Putty logo

PuTTY è un client SSH, Telnet e rlogin combinato con un emulatore di terminale per la gestione in remoto di sistemi informatici (es. computer, server ecc..). È una suite di software libero, originariamente disponibile solo per sistemi Microsoft Windows, ma in seguito anche per vari sistemi Unix e Unix-like. Molto comodo per applicazione come ad esempio questo progetto in cui si può accedere alla raspberry e comandarla molto facilmente avendo accesso remoto al terminale. Il software è gratuito e disponibile per Windows ma non per MAC OS.

9.3 Altium

Figura 9.3: Altium logo

Altium Designer 20 rappresenta decenni di innovazione e sviluppo incentrati sulla creazione di un ambiente di progettazione veramente unificato, che consente agli utenti di connettersi facilmente con ogni aspetto del processo di progettazione PCB.
Raggiungendo il perfetto equilibrio fra potenza e facilità di utilizzo, Altium Designer si è saldamente affermata come la soluzione di progettazione PCB più utilizzata sul mercato.
Vi è la possibilità di realizzare PCB in 3D e poterli vedere gia realizzati virtualmente direttamente sul proprio pc.
Molto pratico da usare e facile da imparare grazie a vari tutorial presenti in rete.

9.4 MySql

Figura 9.4: Mysql logo

MySQL o Oracle MySQL è un relational database management system (RDBMS) composto da un client a riga di comando e un server, entrambi disponibili sia per sistemi Unix e Unix-like sia per Windows; le piattaforme principali di riferimento sono Linux e Oracle Solaris.
Software libero rilasciato a doppia licenza, compresa la GNU General Public License, sviluppato per essere il più possibile conforme agli standard ANSI SQL e ODBC SQL. I sistemi e i linguaggi di programmazione che lo supportano sono molto numerosi: ODBC, Java, Mono, .NET, PHP, Python e molti altri.
Le piattaforme LAMP e WAMP incorporano MySQL per l'implementazione di server per gestire siti web dinamici, inoltre molti dei content management system di successo come WordPress, Joomla, Drupal e TikiWiki nascono proprio con il supporto predefinito a MySQL.
Alcuni fork sono nati in critica verso la poca apertura di MySQL ai contributi dei volontari esterni e la lentezza della pubblicazione dei fix dei bug segnalati. Il fork che ho utilizzato per questo progetto è MariaDB. Il nome MariaDB è dovuto al fatto che inizialmente questo fork si focalizzava soprattutto sullo sviluppo dello storage engine Aria, il cui vecchio nome era Maria in dedica alla terza figlia di Widenius, una sorta di evoluzione di MyISAM.
Sono state incluse patch realizzate da terze parti, in particolare prelevate dai fork di MySQL sviluppati da Google, Facebook e Twitter, nonché storage engine sviluppati da terze parti. Inoltre altre migliorie al server e alcuni storage engine aggiuntivi sono stati sviluppate appositamente, alcune evoluzioni sviluppate per MySQL sono state importate nel fork e alcuni bug presenti nel programma originale sono stati corretti. Esistono molti altri fork presenti in rete e funzionanti.
In questo sito "https://www.w3schools.com/python/python_mysql_create_db.asp" sono descritte molto bene tante operazioni che possono essere fatte con questo Database.

9.5 Grafana

Figura 9.5: Grafana logo

Grafana è un'applicazione web di analisi interattiva e visualizzazione interattiva multipiattaforma. Fornisce grafici, grafici e avvisi per il Web quando è collegato a origini dati supportate. È espandibile attraverso un sistema plug-in.
Ci sono molte ragioni per cui usare Grafana:

- è un progetto open source, stabile e dotato di una documentazione decisamente esauriente
- è estremamente configurabile ed espandibile
- è attivamente sviluppato e mantenuto
- è utilizzato da molti big del settore – StackOverflow per dirne uno
- i grafici e le visualizzazioni generate sono decisamente gradevoli alla vista: anche l'occhio vuole la sua parte era già in uso per la monitorizzazione di alcuni servizi secondari

Capitolo 10

Componentistica principale

10.1 Energy metering Chip

10.1.1 Premessa

Per quanto riguarda la scelta dell'energy metering chip si è inizialmente fatta una ricerca sul web per verificare quali tipi di chip diffententi esistessero, paragonandone le performance, le caratteristiche varie e tutte le possibili misure che potessero fare. Uno dei criteri base era di misurare corrente, tensione, potenza, frequenza di un sistema trifare con 3 fasi e un neutro. Per quanto riguarda l'aspetto della comunicazione con un microprocessore, bastava avesse comunicazioni come SPI, UART, I2c, che sono le classiche che tutti i micro presentano. Per quanto riguarda i vari modelli si è trovato un chip della Microchip (ATM90E32AS) e uno della Texas Instrument, molto interessanti che facevano da buona concorrenza all'Ade9000 proposto dal professore. I vantaggi principali dell'utilizzo dell'ade9000 dopo vari confronti è stato che il chip dell'analog Devices riuscisse a fare molte più misure in più rispetto agli altri con una maggiore precisione, e quindi per un'analisi dettagliata della rete potrebbe essere davvero interessante l'utilizzo di quest'ultimo. Infine il vantaggio che ci ha maggiormente vincolati è stato che Analog Deviced producesse una evaluation board per Ade900 cosa che gli altri produttori di energy metering non facevano per i chip scelti e vista la situazione COVID si è stati maggiormente convinti dell'acquisto di una board per fare i test abbastanza costosa.

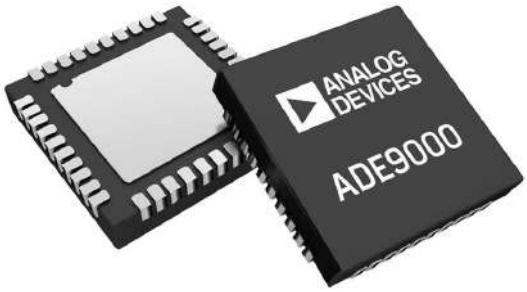

Figura 10.1: Ade9000 chip

10.1.2 ADE9000

L'evaluation board dell'Analog Devices Ade9000 è una scheda di dimensioni circa 15x15 cm costituita da 4 ingressi di tensione (3 fasi + 1 neutro) e 4 entrate per misurare la corrente. Visto che il neutro è considerato come massa delle 3 tensioni, il chip permette l'analisi di 7 input analogici (avendo appunto 7 ADC) che corrispondono ai 4 di corrente e 3 di fasi. Il tutto include anche un sensore di temperatura che è stato aggiunto sulla scheda ma che nel nostro caso non andiamo ad utilizzare. Per quanto riguarda invece la comunicazione con il processore viene utilizzata una SPI.

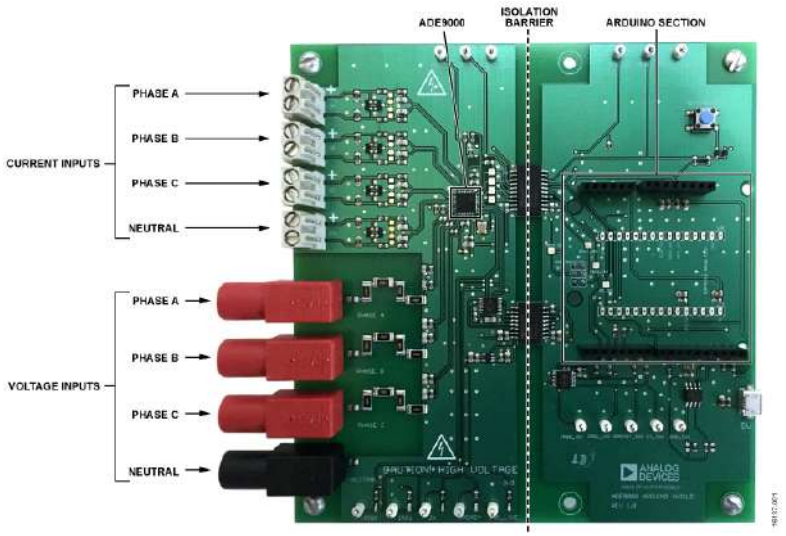

Figura 10.2: ADE9000 evaluation board

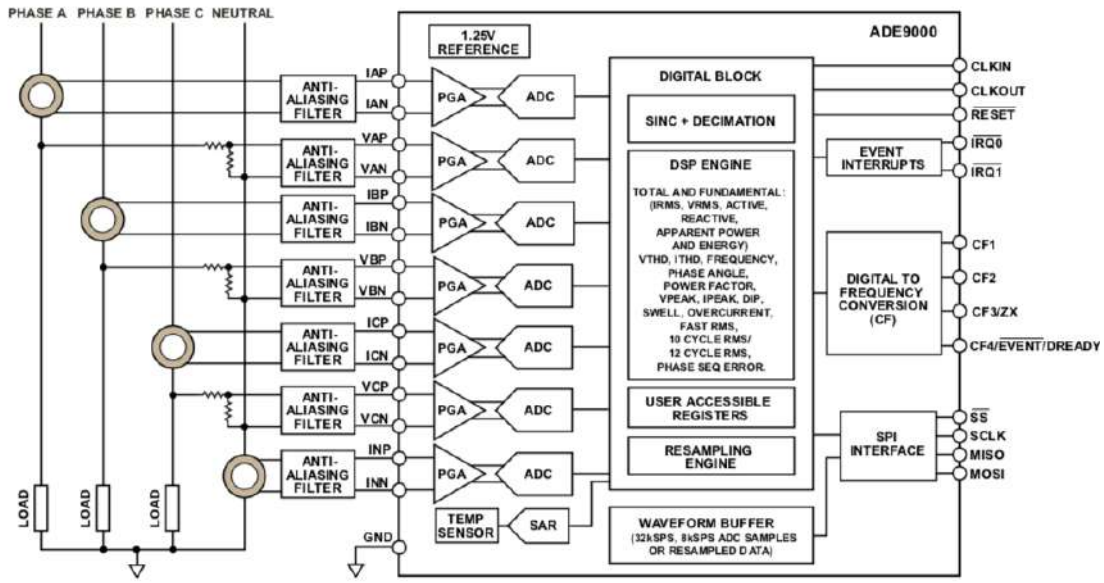

Figura 10.3: ADE900 SCHEMATIC

Il tutto detto precedentemente è rappresentato nello schematico del datasheet Ade9000 fornito dalla casa produttrice.

10.1.2.1 ADC Analog to digital converter

In questa sezione andiamo un po' ad analizzare meglio il funzionamento dell'ADC del chip in questione. Prima di tutto bisogna sapere in cosa consiste l'ADC cioè in una conversione di un segnale analogico in digitale tramite alcuni procedimenti. In un convertitore, maggiore è il numero di bit e più alta è la precisione del nostro dispositivo, in particolare nell'ADE ogni ogni ADC possiede 24 bit il che permette di avere 2 alla 24 livelli cioè ben 1677216 livelli di quantizzazione. Il passaggio successivo è l'amplificazione programmabile, cioè la possibilità di scegliere se applicare un guadagno di 1,2 oppure 4 del valore misurato in ingresso. Ogni ADC può ricevere al massimo 0.6V in ingresso per questo motivo è fondamentale avere in ingresso un particolare partitore che abbassi la tensione a questo livello.

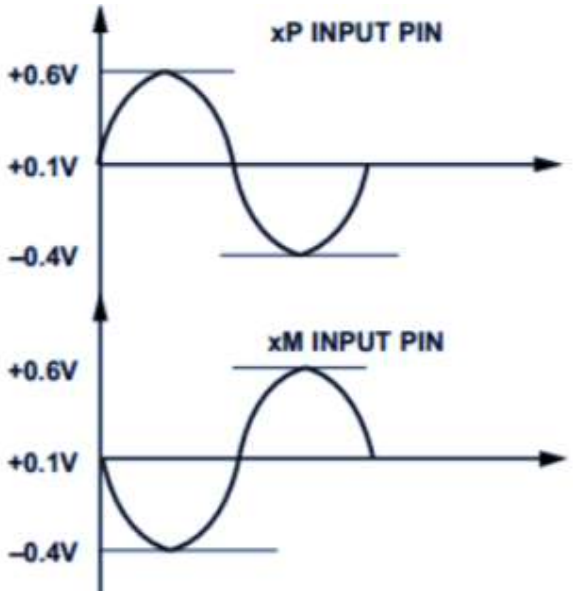

Figura 10.4: Max Input voltage ADC

Come si vede bene da questa immagine, l'ingresso positivo e negativo per ogni tensione può avere come massimo 0,6 e negativo -0,4V.
Per quanto riguarda invece la misura della corrente, vengono utilizzati dei CTS cioè current trasformers o anche chiamato bobina di rogowski. Per l'interfacciamento dei sensori di corrente, una particolare funzione è svolta dai filtri antialisasing (Il seguente fenomeno spiega che due segnali diversi durante la campionatura possano diventare simili se non uguali tra loro). Per questo motivo sia sugli ingressi di tensione che corrente sono aggiunti per formare un filtro a 7khz tramite un semplice RC, in modo da evitare errori di fase.

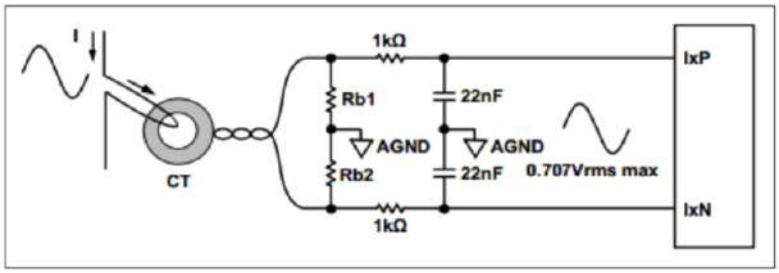

Figura 10.5: CTs circuit

Nella figura sopra si può ben vedere come è fatto il filtro e lo schema ti collegamento del cts agli ingressi ADC mentre nel successivo vi è un partitore resistivo con filtro RC.

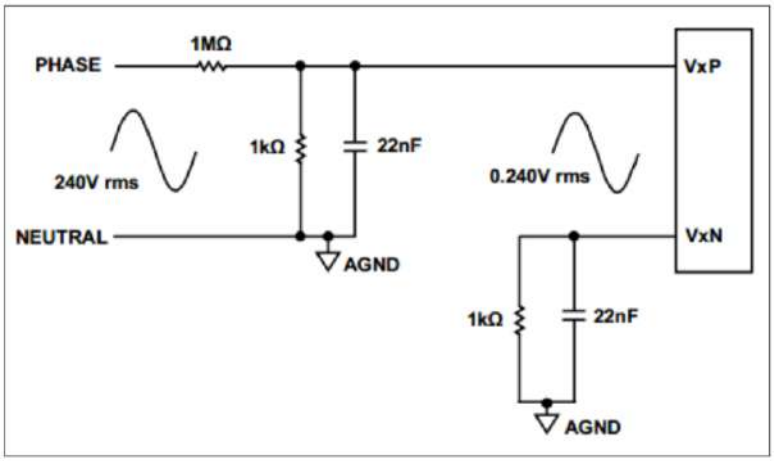

Figura 10.6: Voltge circuit

Ora invece vediamo meglio altri aspetti del chip ADE9000. Come ben sappiamo ci sono due parti ben distinte (control path e data path). Nel datasheet possiamo trovare uno schema del datapath che ci dice che questa è l'unità di calcolo con cui vengono ricavate tutte le misure per andare poi a realizzare la forma d'onda che viene memorizzata all'interno del buffer del chip. In questo schema vi è un modulatore ADC che campiona ad una frequenza di 2.048 MHz e che ci restituisce in uscita una seguenza da 1 bit alla volta dove il corrispettivo valore intero corrisponde appunto alla misura fatta. Il filtro SINC4 presente in figura ci serve per ottenere la risoluzione di 24 bit.

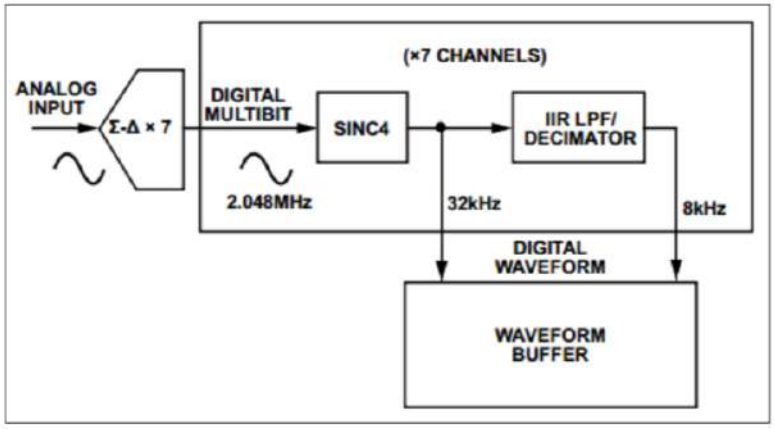

Figura 10.7: Ade9000 schematic datapath

Ciò che si vede denominato con IIR LPF non è altro che un filtro passa basso digitale con la stessa funzione di un passa basso analogico RC con la funzione di antialiasing. Infatti entrambi i filtri sono collegati al buffer del chip per andare a realizzare la forma d'onda raccolta dai due filtri. Il buffer citato precedentemente è composto da ben 32 bit. i dati che vengono shiftati di 4 bit verso sinistra ad ogni misura. La tensione di riferimento usata dall'ADE per

la misura e calibrazione è di 1,25V. Se non si volesse utilizzare questo riferimento vi è la possibilità di utilizzare in pin esterno e settare dei bit nel registro CONFIG1.

10.1.2.2 Registri Prime misure

Register Name	Description	Update Rate
AI_SINC_DAT	IA sinc4 filter output	32 ksps
BI_SINC_DAT	IB sinc4 filter output	32 ksps
CI_SINC_DAT	IC sinc4 filter output	32 ksps
NI_SINC_DAT	IN sinc4 filter output	32 ksps
AI_LPF_DAT	IA sinc4 + IIR LPF filter output	f_{DSP} = 8 ksps
BI_LPF_DAT	IB sinc4 + IIR LPF filter output	f_{DSP} = 8 ksps
CI_LPF_DAT	IC sinc4 + IIR LPF filter output	f_{DSP} = 8 ksps
NI_LPF_DAT	IN sinc4 + IIR LPF filter output	f_{DSP} = 8 ksps
AI_PCF	Instantaneous current on IA	f_{DSP} = 8 ksps
BI_PCF	Instantaneous current on IB	f_{DSP} = 8 ksps
CI_PCF	Instantaneous current on IC	f_{DSP} = 8 ksps
NI_PCF	Instantaneous current on IN	f_{DSP} = 8 ksps
AIRMS	Filtered-based total rms of IA	f_{DSP} = 8 ksps
BIRMS	Filtered-based total rms of IB	f_{DSP} = 8 ksps
CIRMS	Filtered-based total rms of IC	f_{DSP} = 8 ksps
NIRMS	Filtered-based total rms of IN	f_{DSP} = 8 ksps
ISUMRMS	Filtered rms of vector sum (AI_PCF + BI_PCF + CI_PCF ± NI_PCF); see the Neutral Current RMS, Vector Current Sum section	f_{DSP} = 8 ksps
IPEAK	Peak current channel sample; see the Peak Detection section	f_{DSP} = 8 ksps
ANGLx_xxx	Voltage to current or current to current phase angle; see the Angle Measurement section	CLKIN/24 = 1024 ksps

Figura 10.8: Ade Register

Sopra si possono notare alcuni dei registri che vengono utilizzati per le misure. Come evidenziato in figura si può dedurre che le stesse misure possono essere fatte utilizzando diverse opzioni e filtraggio.

10.1.2.3 Sequenza di avvio ADE9000

Nel chip presente vi sono due ingressi, denominati PM0 e PM1 dal quale di possono impostare le modalità PSM0 o PSM3. In pratica se viene attivato PSM0 viene attivato l'ade9000 insieme agli adc, spi e tutto il resto, mentre se si imposta PSM3 il chip entra in uno stato di riposo dove tutto viene messo in stand-by. Una particolare funzione è quella di disabilitare i singoli ADC scrivendo il un registro adibito a ciò.

PSMx Power Mode	Description	PM1 Pin	PM0 Pin	Functions Available	SPI Available?
PSM0	Normal mode	0	0 or 1	All functions	Yes
PSM3	Idle	1	1	None	No

Figura 10.9: Modalità ADE9000

E' fondamentale che avvengano determinati passaggi durante l'avvio, i quali riportati qui sotto in sequenza:

- Viene atteso l'interrupt RSTDONE e i pin IRQ0 e IRQ1 che avdano a 0
- Viene impostata la modalità PSMx
- Il reset viene portato attivo alto
- Vengono impostati tutti i guadagni e le calibrazioni
- Viene scritto un comando 'RUN' nel registro RUN

10.1.2.4 Alimentazione ade9000

L'ade è alimentato con tensione di 3,3V con un massimo limite di 3,96V il quale è altamente sconsigliato raggiungerlo. Altre tensioni presenti sul chip sono AVDD di 1,7V e DVDD di 1,9V che vengono utilizzate per effettuare un controllo del funzionamento del chip. La corrente che circola all'interno del chip è inferiore ai 17mA. Per la corretta sincronizzazione delle varie periferiche è consigliato da datasheet rispettare un certo andamento riportato successivamente.

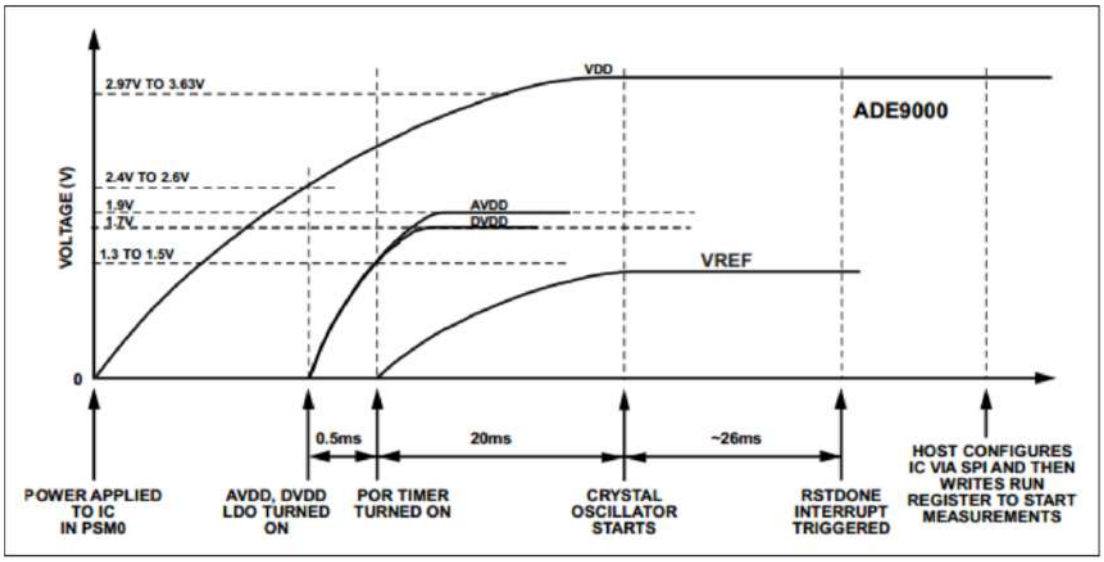

Figura 10.10: Sequenze alimentazione PSM0

10.1.2.5 Calibrazione

Questo passaggio diventa fondamentale per avere una corretta lettura dal proprio strumento. In particolare la calibrazione deve essere fatta inizialmente tramite un altro strumento di misura gia calibrato in modo da paragonare le misure e ricevere valori corretti. Ovviamente è da considerare che l'accuratezza dello strumento dipende dall'accuratezza del dispositivo che viene utilizzato per la calibrazione per questo motivo se si volessero fare particolari calcoli, è fondamentale tenere conto di tutte queste varianti. A livello software la calibrazione è molto semplice, cioè basta inserire un riferimento in alcuni registri che vengono poi sfruttati dall'ade per farci leggere i valori finali corretti.

10.1.2.6 ADE9000 schema e collegamenti

I collegamenti corretti per la nostra scheda di controllo è il seguente che troviamo sul datasheet

ESP8266 Pin	ADE9000 Signal	Type
D5 (GPIO 14)	SCLK	SPI
D6 (GPIO 12)	MISO	SPI
D7 (GPIO 13)	MOSI	SPI
D0 (GPIO 16)	CS (pulled up)	SPI, output
D8 (GPIO 15)	PM1 (pulled down)	Output
D1 (GPIO 5)	IRQ0	Input
D2 (GPIO 4)	IRQ1	Input
3.3 V	Not applicable	Power
5 V	Not applicable	Power

Figura 10.11: Collegamento esp8266 a ade9000

A seguire invece nelle pagine successive gli schematici dell'evaluetion board utilizzata, dove possiamo riconoscere gli ingressi di corrente, tensione e ade9000 della prima pagine, mentre nella seconda un'interessante utilizzo di dispositivi ad accoppiamento magnetico per separare i segnali derivanti dalla parte della scheda connessa alla parte di 'potenza' alla parte invece di controllo. Oltre a ciò vi sono i collegamenti per l'interfacciamento sia dell'esp8266 utilizzato da noi sia per arduino zero.

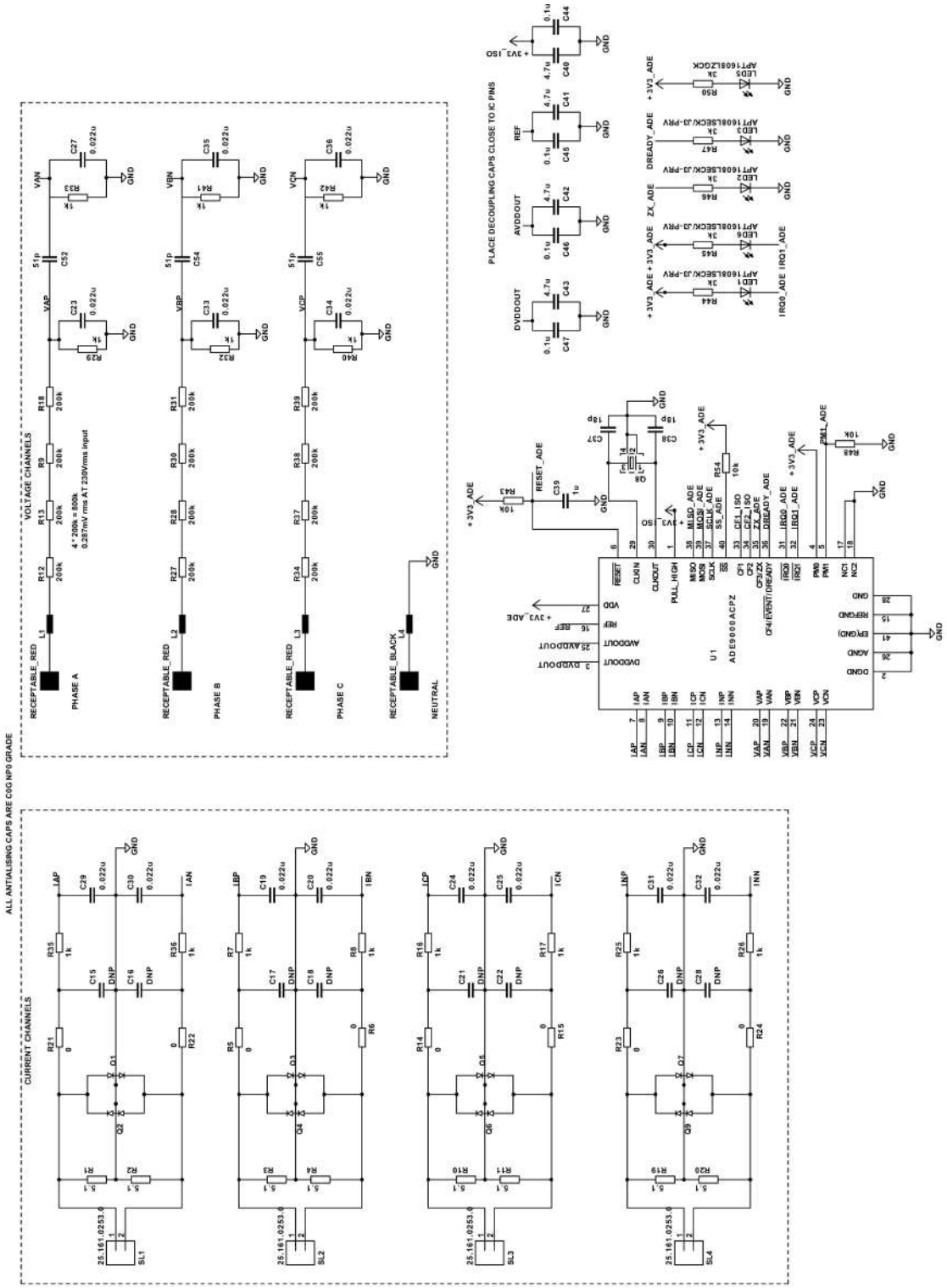

Figura 10.12: Schematico Ade9000 1' parte

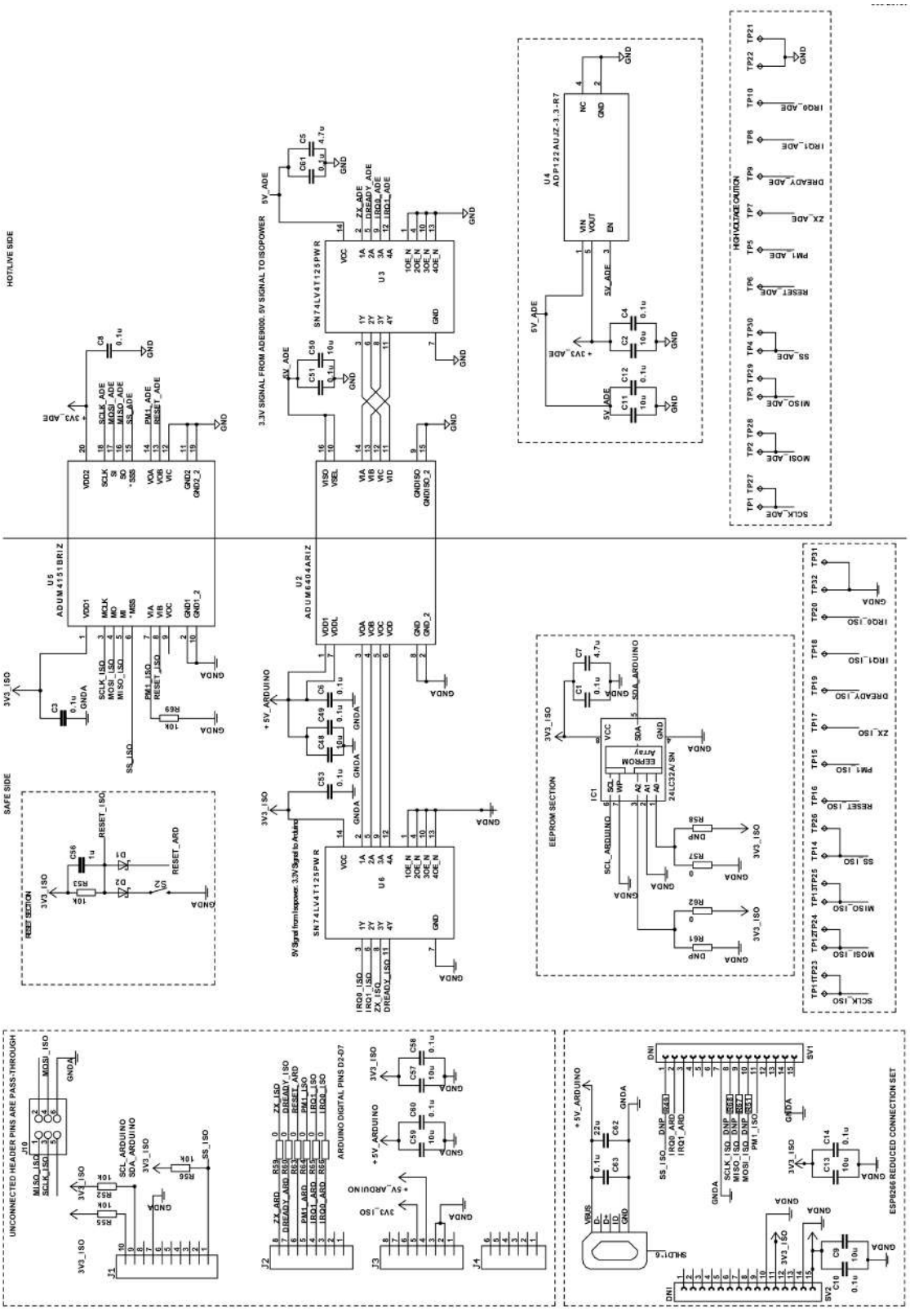

Figura 10.13: Schematico Ade9000 2’ parte

10.2 RTC ds1302

Il modulo MH-Real-Time Clock Module si basa sull' integrato DS1302 al cui interno è presente un orologio in tempo reale / calendario e 31 byte di RAM statica.

Figura 10.14: Ds1302

L'integrato DS1302 è simile al modello DS1307 dal quale si differenzia per alcune particolarità:

- Il DS1302 ha un'interfaccia SPI mentre il DS1307 ha un'interfaccia I2C;
- Il DS1302 può ricarica la batteria, mentre il DS1307 non lo fà;
- Il DS1307 è dotato di un'uscita ad onda quadra programmabile.

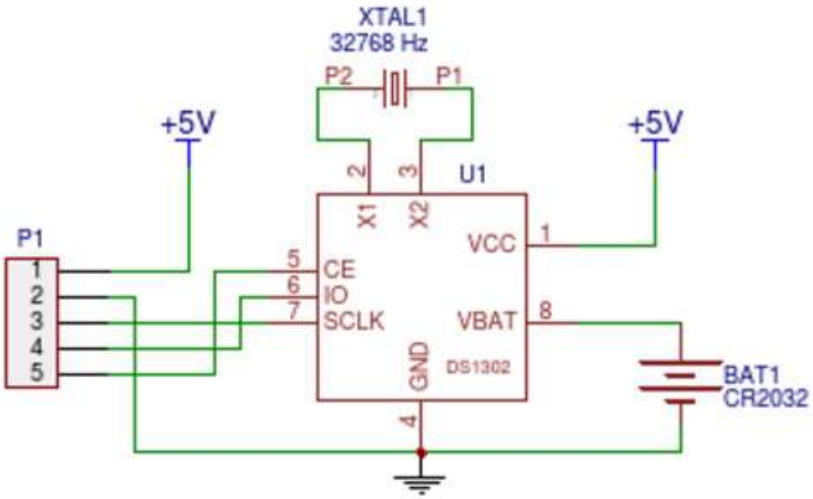

Figura 10.15: Ds1302 circuito

Qui potete vedere lo schema elettrico per il DS1302 Real Time Clock. VCC è impostato per accettare l'alimentazione primaria, questa è tipicamente 3.3V, ma può utilizzare 5V. Si raccomanda di non applicare più di 7V, in quanto ciò può danneggiare l'unità. VBAT viene utilizzata per l'alimentazione di backup, fornita dalla batteria 3.3V tipo CR2032. Il cristallo di quarzo è posto tra i pin 2 e 3, mentre i Pin 5-6-7 sono utilizzati per la comunicazione dei dati tra il modulo e il microcontrollore, è inoltre collegato il pin CE indicato come RST.

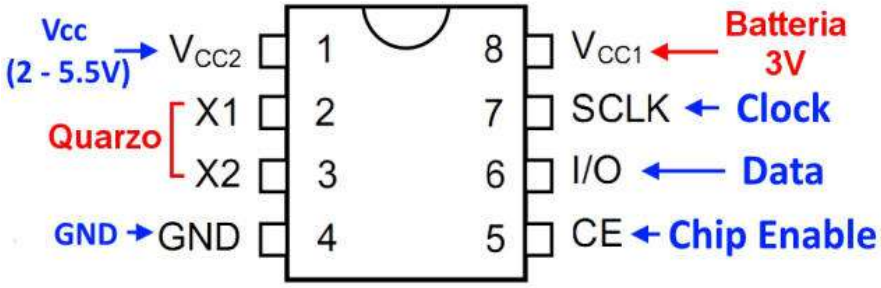

Figura 10.16: Ds1302 pin

Per l'interfaccia, oltre all'alimentazione, sono necessari solo tre pin : CE (RST), I/O (linea dati), e SCLK (serial clock). I dati possono essere trasferiti da e verso l'orologio/RAM 1 byte alla volta o in un inico invio fino a 31 byte. Il DS1302 è progettato per funzionare con una potenza molto bassa e per conservare dati e informazioni di clock con meno di 1W. Il DS1302 ha pin di alimentazione doppi, una primaria (Pin 1 4) e un altro per la batteria di backup (Pin 8 4), corrente 260mAh, batteria non ricaricabili. tempo di conservazione dei dati teorica è più di 10 anni.

10.3 MAX485 shield

Questo piccolo shield è il circuito per utilizzare il chip MAX485 che trasforma un segnale ttl in rs485 in modo molto semplice da utilizzare.

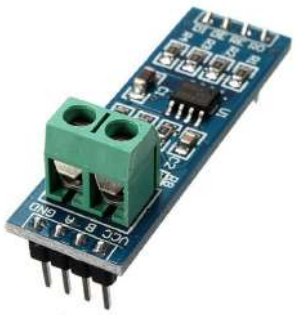

Figura 10.17: MAX485 shield

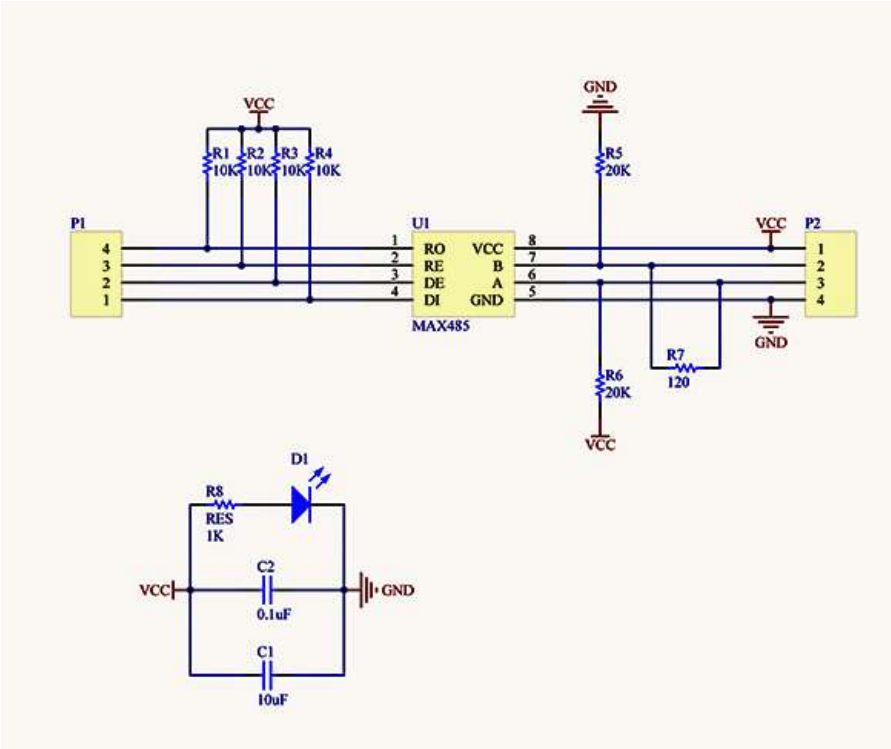

Figura 10.18: MAX485 schematico

Come si può ben vedere dallo schema elettrico, i collegamenti oltre che alimentazione (Vcc-gnd) vi è l'uscita A e B per rs485 e gli ingressi r0 e di che corrispondono a tx e rx e i corrispettivi enable.

10.4 Ethernet ENC28j60

Questa piccola evaluation board è utilissima per quando riguarda l'interfacciamento della rete alla nostra raspberry compute module 3 visto che non è prevista di connessione di rete.

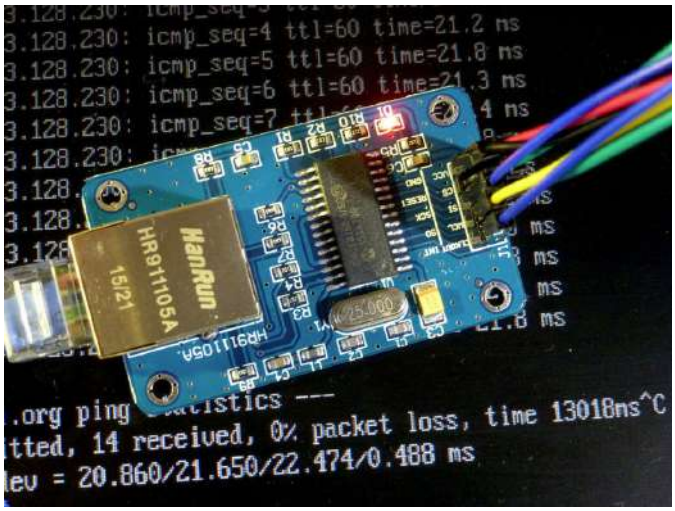

Figura 10.19: ENC28j60

I collegamenti da eseguire per la raspberry sono elencati nella tabella sottostante:

Pi	ENC28J60	Colour
+3V3	VCC	Red
GPIO10/MOSI	SI	Green
GPIO9/MISO	SO	Yellow
GPIO11/SCLK	SCK	Blue
GND	GND	Black
GPIO25	INT	Blue
CE0#/GPIO8	CS	Green

Figura 10.20: ENC28j60 connessione raspberry

Una volta eseguiti correttamente i collegamenti bisogna recarsi nel terminale e digitare ***'ls /boot/overlays/enc28*'***. Se al comando Raspberry non da errore vuol dire che rileva il collegamento corretto tramite una risposta. Fatto ciò bisogna recarsi in ***'sudo nano /boot/config.txt'*** e modificare le ultime righe come qui a seguire:
'dtparam=spi=on dtoverlay=enc28j60'

Quello che viene fatto è di abilitare una connessione alla rete tramite SPI. Infine basta fare un semplice reboot e la nostra rete sarà operativa.

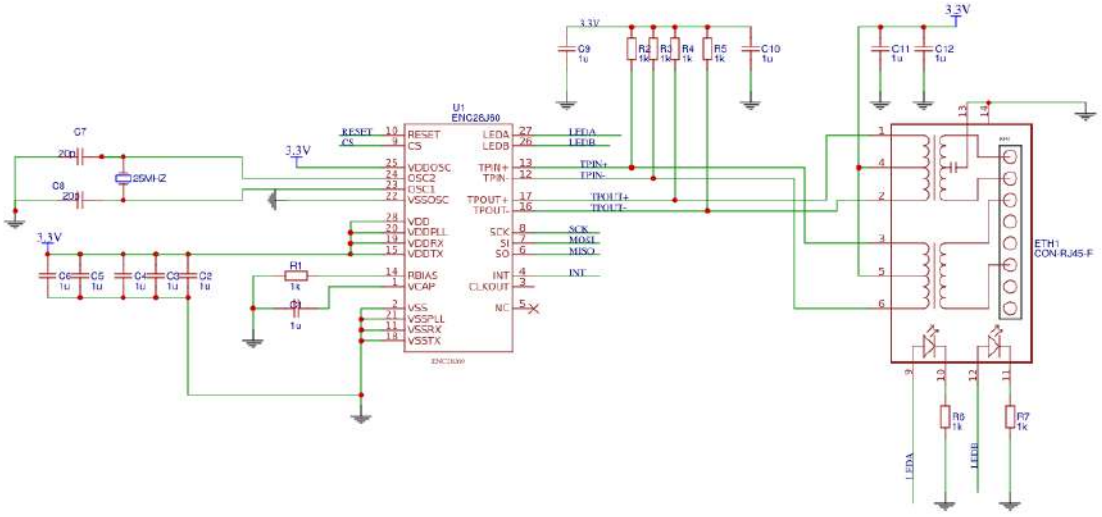

Figura 10.21: ENC28j60 schematico

10.5 Schermo OLED 0,96" adafruit

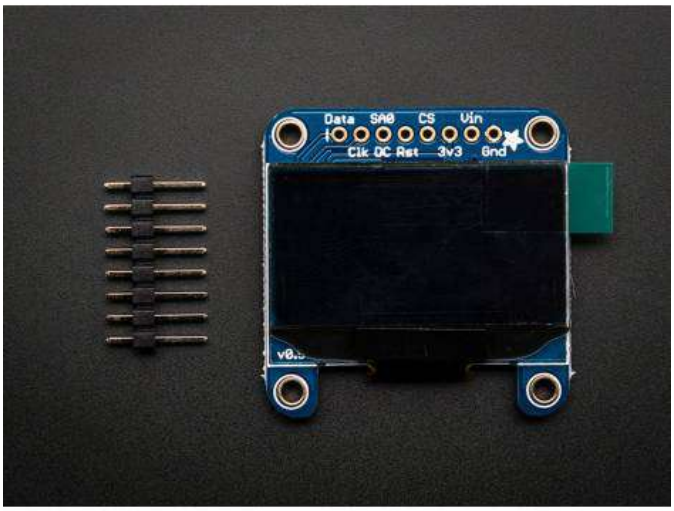

Figura 10.22: Display Oled

Per quanto riguarda la parte master del mio sistema ho trovato interessante utilizzare un piccolo display OLED in cui poter stampare i vari valori letti dai vari slave. E' molto semplice da utilizzare e tramite l'ausilio di un circuito con un pulsante che genera un interrupt sulla raspberry, vengono cambiate le schermate visualizzando tensioni, correnti e altri valori dei vari slavi collegati.

Capitolo 11

Regola per alta tensione su PCB

11.1 Requisiti in termini di distanze

Non tutti i progetti di PCB hanno gli stessi requisiti rigorosi in termini di distanze che invece sono richiesti nei progetti ad alta tensione. In generale, se il normale voltaggio operativo del vostro prodotto raggiunge o supera i 30 VAC o i 60 VDC, dovete prestare particolare attenzione alle regole sulle distanze nel progetto del vostro circuito. Se avete una scheda ad alta densità e le tensioni sono elevate dovete fare ancora più attenzione. L'alta densità rende la corretta spaziatura più difficile da ottenere ed è fondamentale per la protezione.
La spaziatura riveste un ruolo ancora più importante nei progetti ad alta tensione perché il voltaggio presente sui componenti della scheda rende più probabile l'insorgere di archi elettrici tra due elementi conduttivi. Qualsiasi arco elettrico costituisce un pericolo serio sia per il prodotto che per l'utilizzatore. Per ridurre il rischio esistono degli standard da rispettare in termini di distanze di isolamento in aria e superficiali.

11.2 Distanza di isolamento in aria

La distanza di isolamento è la distanza minima in aria tra due conduttori. Mi ricordo di questa definizione quando penso alla distanza di sicurezza sopra la mia testa: quanto spazio c'è prima che possa andare a sbattere contro qualcosa. Se la distanza di isolamento in un qualunque punto del PCB è troppo bassa una sovratensione può causare un arco elettrico tra due componenti conduttivi vicini.
Le regole su questa distanza cambiano in funzione del materiale del PCB, del voltaggio e delle condizioni ambientali. Le condizioni ambientali, in particolare, giocano un ruolo significativo. L'umidità modifica la tensione di breakdown dell'aria e aumenta la probabilità di un arco elettrico. La polvere è un altro fattore importante dato che le particelle che si accumulano sulla superficie del PCB possono andare a formare una traccia riducendo la distanza tra i conduttori.

11.3 Distanza di isolamento superficiale

Similmente a quanto visto prima la distanza di isolamento superficiale si riferisce alla distanza tra i conduttori i un PCB. Anziché la distanza in aria questa volta quello che viene misurato è la distanza minima lungo la superficie del materiale di isolamento. Anche in questo caso il materiale della scheda e i fattori ambientali hanno una notevole influenza. L'umidità o l'accumulo di particelle sulla scheda possono ridurre tale distanza, esattamente come visto prima.
Quando la densità sulla scheda è elevata può essere difficile rispettare il requisito relativo alla distanza minima superficiale. Muovere le tracce è raramente un'opzione percorribile, tuttavia esistono un paio di trucchi per aumentare la distanza. Aggiungere una scanalatura tra le tracce o una barriera verticale di isolamento può aumentare, in modo significativo, la distanza di isolamento superficiale senza modificare il layout della scheda.

11.4 Considerare l'indice di inseguimento comparativo (CTI) del materiale

Dopo la tensione operativa il fattore più importante per rispettare i requisiti in termini di distanze di isolamento in aria e superficiali è legato alle proprietà del materiale del vostro PCB. Il livello di isolamento elettrico dei materiali è indicato dal cosiddetto "Indice di inseguimento comparativo" o CTI. Il CTI è espresso come tensione ed è determinato da un test standard che misura quando la superficie del materiale si rompe.
Esistono 6 categorie di materiali, da 0 in 5, in funzione del valore di questo indice. I valori di isolamento richiesti sono sempre basati su queste categorie. La categoria 5 è quella più bassa, con valori inferiori a 100 V. Con tensioni di rottura superiori a 600 V i materiali che appartengono alla categoria 0 sono i più resistenti ma anche, molto spesso, i più costosi.

11.5 Come determinare quale materiale e spaziatura utilizzare

Nel progetto di PCB ci sono tantissime variabili e materiali da scegliere per cui, per rispettare i requisiti di sicurezza, è meglio rivolgersi direttamente alla fonte. Esistono due standard che vengono applicati comunemente. Il primo è lo standard IPC-2221 che determina i requisiti in termini di distanze di isolamento in aria e superficiali per i casi più generali e il secondo è il IEC-60950-1 (seconda edizione). Lo standard IEC è quello da rispettare per i prodotti IT con alimentazione AC o a batteria, specialmente se volete vendere i vostri prodotti sul mercato internazionale.
Dato che le conseguenze di una spaziatura non corretta sono molto serie dal punto di vista legale e includono danni e persino il decesso vale la pena spendere del tempo per familia-

rizzare con gli standard applicabili al progetto. Inoltre evita che gli studenti siano soggetti a scosse elettriche.
Indentificare ed includere questi standard può essere un processo tedioso per cui è molto utile disporre di un buon software di progettazione. I software migliori per il progetto di PCB consentono di creare specifiche regole di progetto e vi aiutano ad indentificare i problemi già nelle fasi iniziali.

Capitolo 12

Slave - submetering module

In questo capitolo analizzeremo bene tutto il processo che sta dietro agli slave di questo sistema di submetering. In pratica il cervello dello slave è costituito dall' ESP8266 che si occupa di gestire tutte le informazioni che arrivano e comunicare con un master. L'ade9000 si occupa di sviluppare tutte le misure richieste dal micro con eventuali calibrazioni e impostazione di registri, mentre RTC collegata si occupa(dopo aver impostato ad un primo avvio l'orario corretto) di inserire l'orario nei registri modbus per poi essere controllato da master e poter capire se vi è una corretta sincronizzazione. Gli slave sono collegati al master tramite modbus, e per arricchire di più il sistema si è deciso di testare uno slave con modbus half-duplex e l'altro con modalità full-duplex, rispettivamente con collegamenti a 2 o 4 fili. Il sistema intero sarà poi unito su un unico PCB in cui sarà poi impostabile tramite dei 'ponticelli' la modalità di modbus preferita. A livello software sono anche state usate due librerie modbus differenti, anche qua per dimostrare che non vi è una sola libreria funzionante ma più che riescono a collaborare correttamente. Per quanto riguarda il software, vengono eseguite le operazioni secondo un determinato ordine:

- 1) Dichiarazione librerie
- 2) Definizione di registri e variabili globali
- 3) Inizializzazione seriale con creazione dei registri modbus richiesti
- 4) Utilizzo data RTC
- 5) Main loop con esecuzione della funzione ADE9000 che si occupa di leggere i dati e restituirli al sistema
- 6) Fuzione RTC che imposta i valori di tempo negli appositi registri

Di seguito vedremo alcune piccole parti interessanti del software per capire a grandi linee cosa viene fatto.

```
//--------------------------------------------------------------------------------------------
// INCLUSIONE LIBRERIE MODBUS, RTC E ADE9000
//--------------------------------------------------------------------------------------------
// LIBRERIE PER ADE9000
#include <SPI.h>
#include <Hoomaluo_ADE9000CalibrationInputs.h>
#include <Hoomaluo_ADE9000RegMap.h>
#include <Hoomaluo_ADE9000.h>
#include <stdint.h>
#include <math.h>

// LIBRERIE RTC
#if (defined(__AVR__))          //Definizione architettura per libreria
#include <avr/pgmspace.h>
#else
#include <pgmspace.h>
#endif
#include <ErriezDS1302.h>     //Libreria rtc


// LIBRERIA MODBUS
#include <ModbusRTU.h>        //Libreria modbus modbus-esp8266 master
//--------------------------------------------------------------------------------------------
```

Figura 12.1: Dichiarazione librerie

Come si può vedere sono state usate varie librerie, sia per modbus, che ade9000 sia per RTC di cui si fa un breve controllo dell'architettura. Le librerie modbus come specificato prima sono due diverse ma qui è riportato solo un esempio

```
//DEFINE ADE9000-----------------------------------------------------------------
//*********GLOBAL VARIABLES****************
float current_time;                    //for tracking time interval for acquire
float last_time = 0;
float acquire_time = 5;                //number of seconds between acquire
double avrmsaccum,bvrmsaccum,cvrmsaccum,airmsaccum,birmsaccum,cirmsaccum;  //Accumulators
double awattaccum,bwattaccum,cwattaccum,avaaccum,bvaaccum,cvaaccum;
int dccount = 1;                 //counter
//******CALIBRATION********
const int irms_os = 0;          //ADC offset at low current reading
const int vrms_os = 0;          //ADC offset at low voltage reading
float vrms_cal = 0.000017183*0.62;   //Use calibrated multimeter to read voltage and adjust
float irms_cal = 0.000004269*0.736;   //Same as above...use calibrated clamp meter to adjus
float watt_cal = 0.00989166*0.924;    //After calibrating vrms and irms, multiply them toge
//**********INITIALIZE ADE9000*************
ADE9000Class ade9000;          //FIRST POWER MONITOR OBJECT
//ADE9000Class ade9000_2;       //SECOND POWER MONITOR OBJECT uncomment to use more boards
#define SPI_SPEED 500000     //SPI Speed. The mini dev board seems to have problems at high
#define CS_PIN 16            // SS PIN FOR POWER MONITOR 1
//#define CS_PIN 31            // SS PIN FOR POWER MONITOR 2 just example of how to enable
#define VAL_RUN 1
//***********FUNCTION PROTOTYPES************
void init_ADE_regs(); //INITIALIZE GAIN REGISTERS FOR ADE9000
```

Figura 12.2: Prima inizializzazione Ade9000

In questa parte di setup oltre a dichiarare alcune strutture e alcuni parametri, vi è la parte di calibrazione in cui vengono inseriti dei valori nei registri appositi per l"aggiustamento' della misura effettuata. In pratica la calibrazione è molto semplice, cioè viene moltiplicato il valore iniziale per un numero che serve per calibrare la misura in modo corretto.

```
void setup() {
  // Inizio comunicazione seriale
  Serial.begin(9600, SERIAL_8N1);
  // Assegno alla seriale il modbus
  mb.begin(&Serial);
  // Imposto numero id slave
  mb.slave(SLAVE_ID);
  // Aggiungo tutti i registri che ci occorrono
  mb.addHreg(Va_s1);
  mb.addHreg(Vb_s1);
  mb.addHreg(Vc_s1);
  mb.addHreg(Ia_s1);
```

Figura 12.3: Parte di setup

In questa breve parte di setup vi è oltre all'avvio della seriale e impostazione della seriale come mezzo di comunicazione modbus, l'impostazione dell'indirizzo dello slave che è stato precedentemente definito con il numero 1 o 2. Infine l'operazione mb.addHreg() serve per aggiungere il registro che ho denomitato con un certo nome ma che è stato definito numericamente.

```
void init_ADE_regs()
{
  delay(1000);
  ade9000.SPI_Init(SPI_SPEED,CS_PIN);
  ade9000.SPI_Write_16(ADDR_RUN,0x0000);
  ade9000.SPI_Write_32(ADDR_AVGAIN,0x00000002);
  ade9000.SPI_Write_32(ADDR_BVGAIN,0x00000002);
  ade9000.SPI_Write_32(ADDR_CVGAIN,0x00000002);
  ade9000.SPI_Write_32(ADDR_AIGAIN,0x00000002);
  ade9000.SPI_Write_32(ADDR_BIGAIN,0x00000002);
  ade9000.SPI_Write_32(ADDR_CIGAIN,0x00000002);
  ade9000.SPI_Write_32(ADDR_APGAIN,0x00000002);
  ade9000.SPI_Write_32(ADDR_BPGAIN,0x00000002);
  ade9000.SPI_Write_32(ADDR_CPGAIN,0x00000002);
  //ade9000.SPI_Write_32(ADDR_AWATTOS,-14831);
  ade9000.SPI_Write_32(ADDR_ACCMODE,0x0000);
  delay(500);
  ade9000.SPI_Write_16(ADDR_RUN,VAL_RUN);
}
//-------------------------------------------------
```

Figura 12.4: Inizializzazione Registri Ade

Nell'inizializzazione dei registri ade vengono scritti dei valori che sono stati controllati sul datasheet per permettere al chip ADE di funzionare. Dopo aver impostato alcuni parametri vi è la scrittura del comando run per avviare le misure elettriche.

```
// Aggiungiamo il tutto nei registri modbus nelle loro posizioni
mb.Hreg(Va_s1, Va);
mb.Hreg(Vb_s1, Vb);
mb.Hreg(Vc_s1, Vc);
mb.Hreg(Ia_s1, Ia);
mb.Hreg(Ib_s1, Ib);
mb.Hreg(Ic_s1, Ic);
mb.Hreg(F_s1, f);
```

Figura 12.5: Impostazione Registri modbus

L'operazione mb.Hreg(x,y) permette di mettere nel registro x il valore y. In questo modo tutti i dati letti sia derivanti dall'ADE9000 che da RTC vengono inseriti negli Holding registers.

Capitolo 13

Master - computing module

La funzione principale del master è di ricevere i dati provenienti dalle linee modbus half e full duplex e di inserirli in un database mysql che è stato creato in precedenza. Su Raspberry vi è installato anche Grafana a cui è stato associato il database mysql cosi facendo si può molto facilemente visualizzare graficamente i risultati ottenuti per poi poterli analizzare. In più è stato aggiunto alla parte del master un piccolo Oled da 0.96 pollici dove escono schermate riportando i valori misurati al momento. Le varie schermate e lo spegnimento del led sono pilotate da un pulsante che comanda un interrupt alla raspberry. Di seguito possiamo vedere alcune parti di codice importanti per il funzionamento del nostro meter principale.

```
import Adafruit_GPIO.SPI as SPI
import Adafruit_SSD1306
from RPi import GPIO
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(20, GPIO.IN)

import subprocess
import time
from pymodbus.client.sync import ModbusSerialClient as ModbusClient
#from pymodbus.client.async.serial import AsyncModbusSerialClient
#from pymodbus.client.async import schedulers
from pymodbus.register_read_message import ReadInputRegistersResponse
import mysql.connector
#import random
import time
#import datetime from datetime
import datetime
```

Figura 13.1: Librerie Python

Qui si possono vedere le librerie e tutti gli oggetti importati da esse. Inoltre vi è la dichiarazione del pin che gestisce l'interrupt cioè il GPIO20 mentre la parte finale è riguardante la libreria modbus

```
def funz(channel):
        global t
        t = t +1
        print 'eccomi'
        if t == 1:
                draw.rectangle((0,0,width,height), outline=0, fill=0)
                draw.text((x, top),       str('Slave1 V[V]'),  font=font, fill=255)
                draw.text((x, top+8),     str(Va_s1)+"  V", font=font, fill=255)
                draw.text((x, top+16),    str(Vb_s1)+"  V",  font=font, fill=255)
                draw.text((x, top+25),    str(Vc_s1)+"  V",  font=font, fill=255)
                disp.image(image)
                disp.display()
        elif t == 2:
                draw.rectangle((0,0,width,height), outline=0, fill=0)
                draw.text((x, top),       str('Slave1 I[mA]'),  font=font, fill=255)
                draw.text((x, top+8),     str(Ia_s1)+"  mA", font=font, fill=255)
                draw.text((x, top+16),    str(Ib_s1)+"  mA",  font=font, fill=255)
                draw.text((x, top+25),    str(Ic_s1)+"  mA",  font=font, fill=255)

                disp.image(image)
                disp.display()

        elif t == 3:
                draw.rectangle((0,0,width,height), outline=0, fill=0)
                draw.text((x, top),       str('Slave1 P[W]'),  font=font, fill=255)
                draw.text((x, top+8),     str(Pa_s1)+"  W", font=font, fill=255)
                draw.text((x, top+16),    str(Pb_s1)+"  W",  font=font, fill=255)
                draw.text((x, top+25),    str(Pc_s1)+"  W",  font=font, fill=255)

                disp.image(image)
                disp.display()
        elif t == 4:
                draw.rectangle((0,0,width,height), outline=0, fill=0)
                draw.text((x, top),       str('Slave2 V[V]'),  font=font, fill=255)
                draw.text((x, top+8),     str(Va_s2)+"  V", font=font, fill=255)
                draw.text((x, top+16),    str(Vb_s2)+"  V",  font=font, fill=255)
                draw.text((x, top+25),    str(Vc_s2)+"  V",  font=font, fill=255)
                disp.image(image)
                disp.display()
        elif t == 5:
                draw.rectangle((0,0,width,height), outline=0, fill=0)
                draw.text((x, top),       str('Slave2 I[mA]'),  font=font, fill=255)
                draw.text((x, top+8),     str(Ia_s2*100)+"  mA", font=font, fill=255)
```

Figura 13.2: Gestione schermo Oled

In questa immagine si vede una parte della funzione 'funz' che è la parte di codice richiamata quando avviene l'interrupt cioò viene premuto il pusante. In poche parole ogni volta che viene premuto il pulsante viene incrementata una variabile che commuta le schermate che vengono visualizzate sull'OLED. L'ultimo valore poi spegne lo schermo e fa riniziare il conteggio da 0. La gestione del display OLED via I2C è molto semplice e veloce grazie alla libreria rilasciata da Adafruit.

```
mydb = mysql.connector.connect(
        host="localhost",
        user="grafana_user",
        passwd="password",
        database="PowerMeter")
mycursor = mydb.cursor()

client_half_duplex = ModbusClient(method='rtu',port='/dev/ttyUSB0',stopbits=1,bytesize=8,parity='N',baudrate=9600,timeout=10)
#time.sleep(1)
client_full_duplex = ModbusClient(method='rtu',port='/dev/ttyAMA0',stopbits=1,bytesize=8,parity='N',baudrate=9600,timeout=10)
#time.sleep(1)

connection = client_half_duplex.connect()
#time.sleep(1)

connection = client_full_duplex.connect()
```

Figura 13.3: Modbus e mysql

In questo piccolo screenshot sono presenti 2 parti, la prima che permette al programma di connettersi al database creato da terminale mentre una seconda parte che stabilisce la connessione tramite Modbus ai due client.

```
dt2 =datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')

value1 = client_full_duplex.read_holding_registers(10,17,unit=1)
print(value1.registers)
#time.sleep(1
Va_s1 =( value1.registers[0]/100.00)
Vb_s1 = value1.registers[1]/100.00
Vc_s1 = value1.registers[2]/100.00
Ia_s1 = value1.registers[3]
Ib_s1 = value1.registers[4]
Ic_s1 = value1.registers[5]
F_s1  = value1.registers[6]*1.00
aPf_s1 = value1.registers[7]/100
Pa_s1 = value1.registers[8]/100.00
Pb_s1 = value1.registers[9]/100.00
Pc_s1 = value1.registers[10]/100.00
Ora_s1 = value1.registers[11]
Minuti_s1 = value1.registers[12]
Secondi_s1 = value1.registers[13]
Giorno_s1 = value1.registers[14]
Mese_s1 = value1.registers[15]
Anno_s1 = value1.registers[16]
print  Va_s1


#Vb_s1 = value1.registers[3]
sql1 = "INSERT INTO SLAVE1 (dt,Va,Vb,Vc,Ia,Ib,Ic,Pa,Pb,Pc,Fr,Pf) VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
val1 = (dt2 ,Va_s1,Vb_s1,Vc_s1,Ia_s1,Ib_s1,Ic_s1,Pa_s1,Pb_s1,Pc_s1,F_s1,aPf_s1)
mycursor.execute(sql1, val1)
```

Figura 13.4: Modbus e mysql

Quest'ultimo screenshot possiamo invece vedere come tutte le variabili vengano denominate in modo da essere più comprensibili e vengono inserite nel nostro database secondo le giuste colonne di come è stato creato il database.

Capitolo 14

Schemi elettrici e PCB

14.1 Slave

In questo capitolo andremo a vedere tutti gli schematici realizzati per la progettazine del nostro pcb (lato slave) con la presenza di un convertitore ac/dc 220/5V dc in modo da alimentare tutto il nostro sistema. Aggiunta vi è tutta la parte riguardante l'ADE9000 con rtc, max485 per la trasmissione modbus su rs485 e la connessione con ESP8266. Il circuito a livello di pcb è molto complesso in quanto le dimensioni sono 13,5x 7,5 cm visto che la scheda dovrà entrare eventualmente in un contenitore che verrà fissato su una barra DIN e con queste dimensioni di 13,5 cm occuperà circa 8 moduli. I fori ai 4 lati e quelli centrali sono studiati per un evenutale sostegno o case DIN. Le prime pagine sono i vari schematici mentra a seguire la parte di pcb con annessa rappresentazione in 3D.

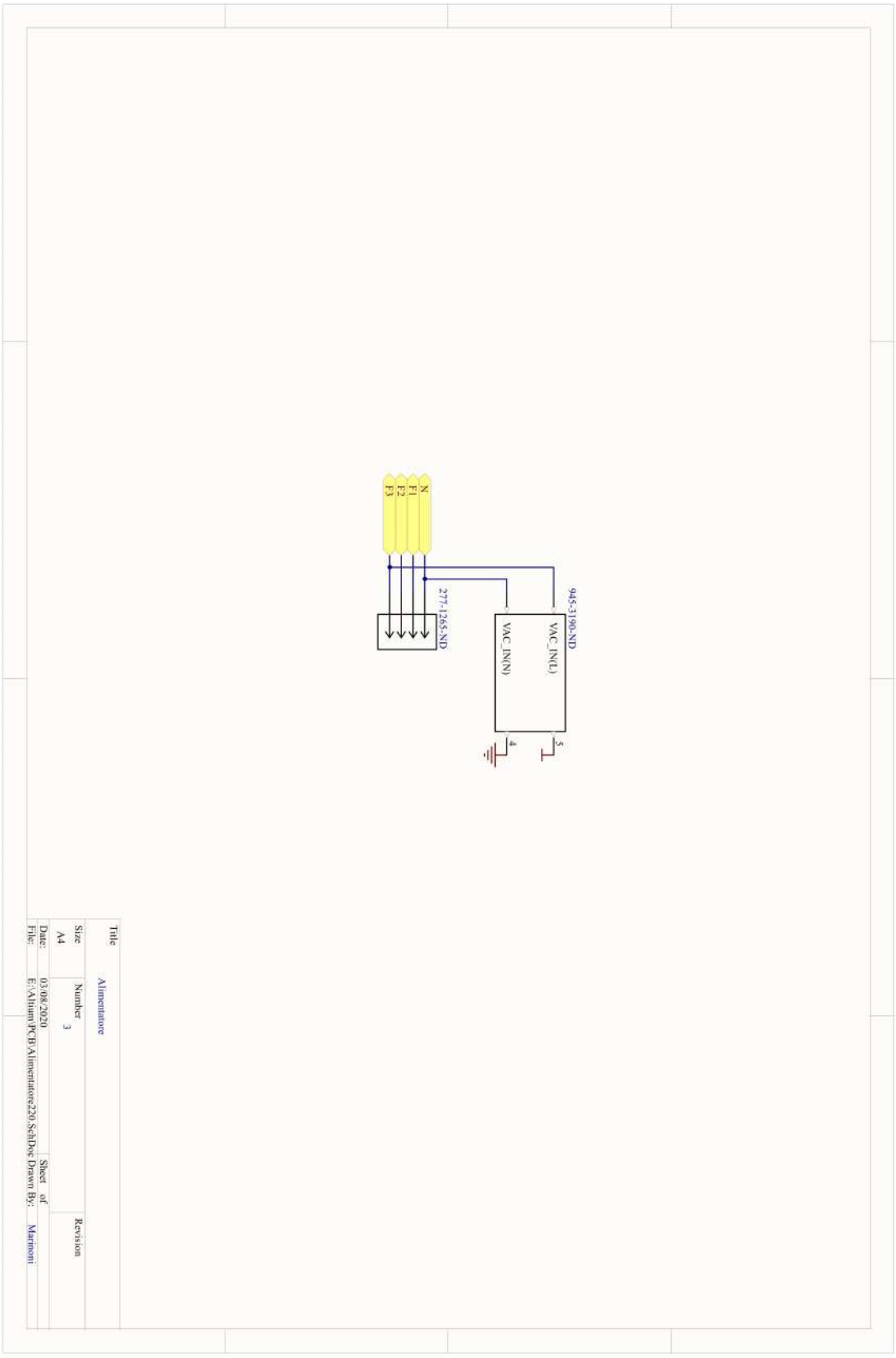

Figura 14.1: Circuito alimentazione

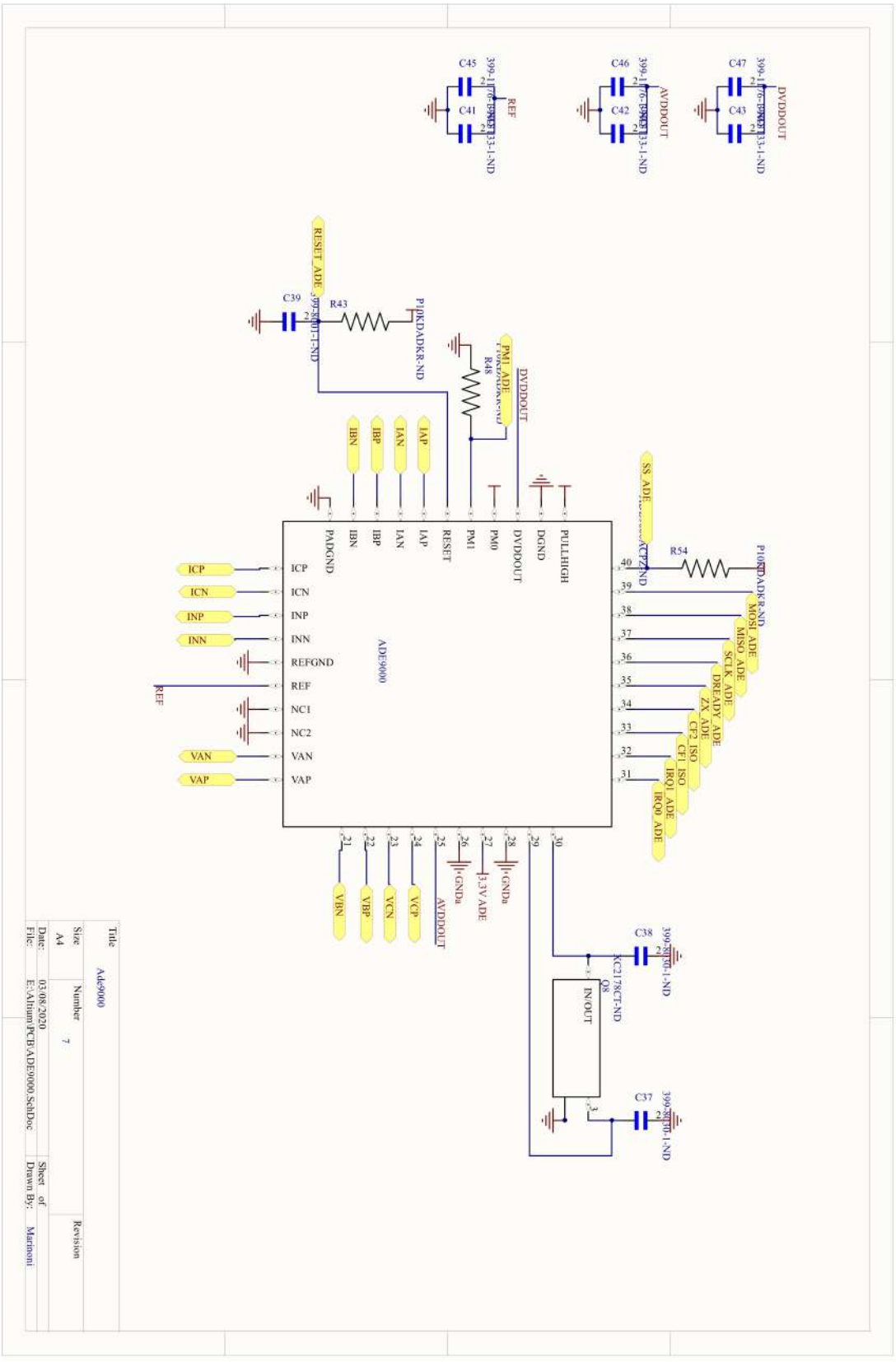

Figura 14.2: Circuito ADE9000

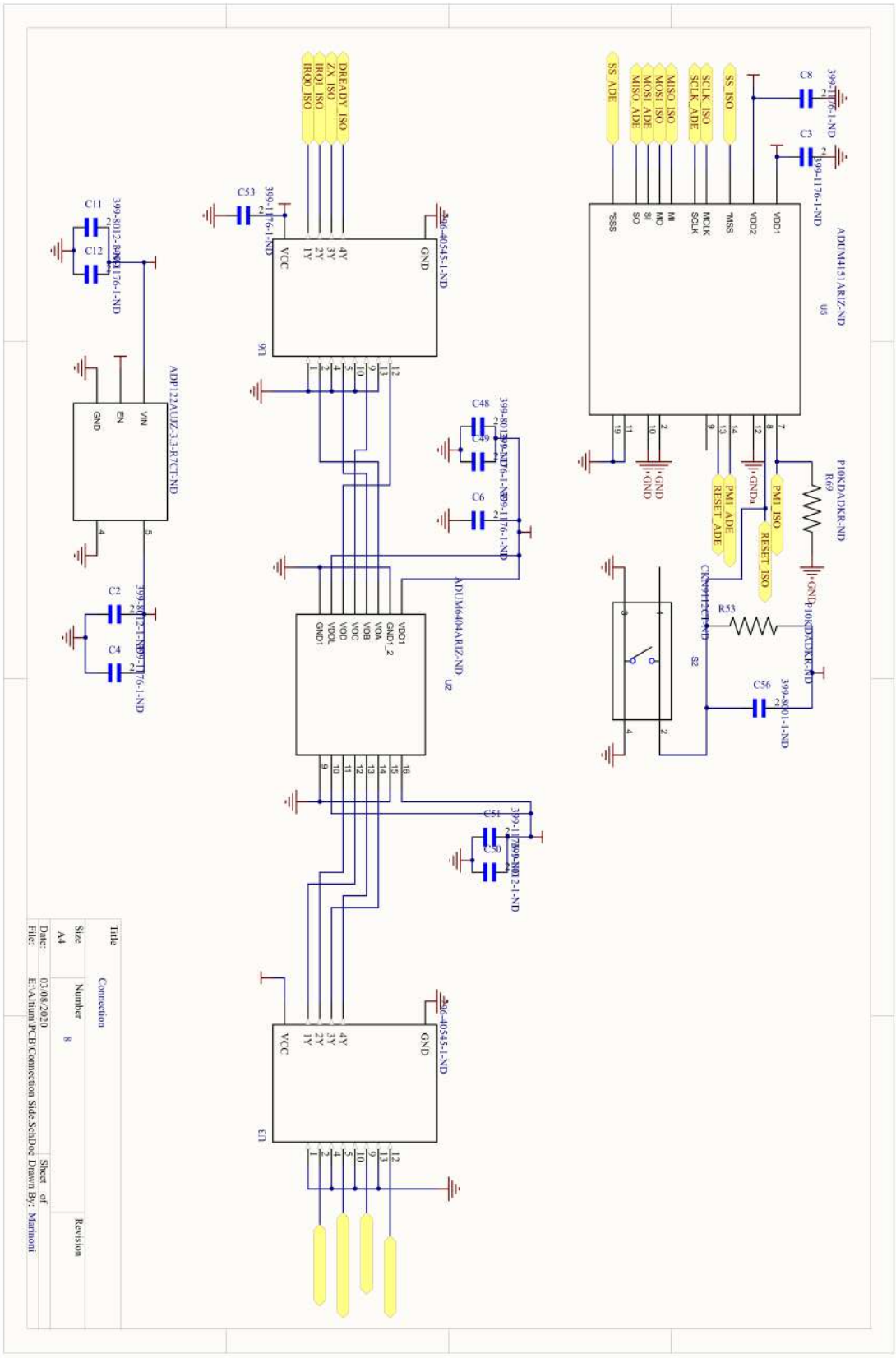

Figura 14.3: Circuito Ade9000 parte 2

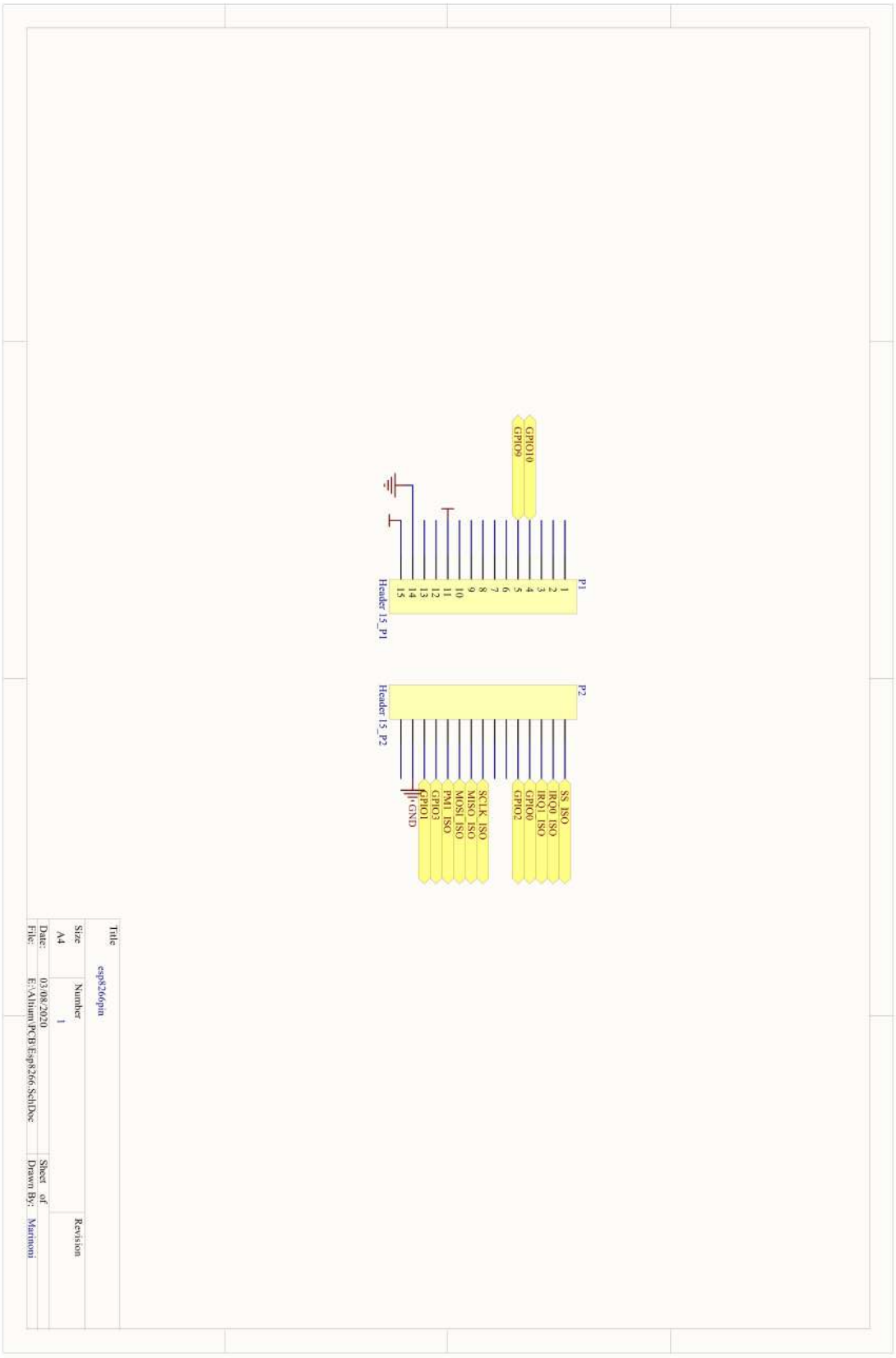

Figura 14.4: Circuito ESP8266 connection

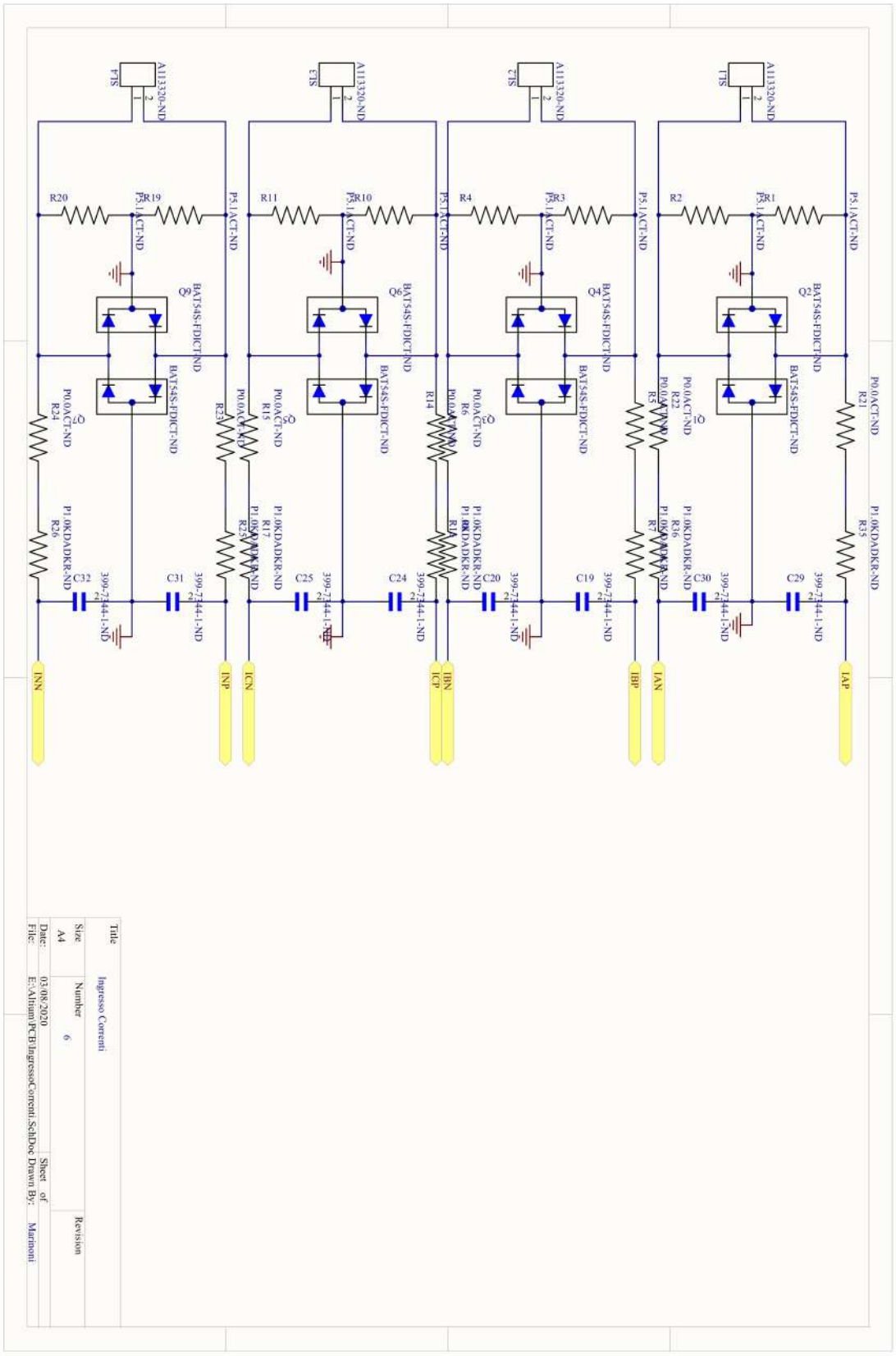

Figura 14.5: Circuito Ingresso Correnti Ade9000

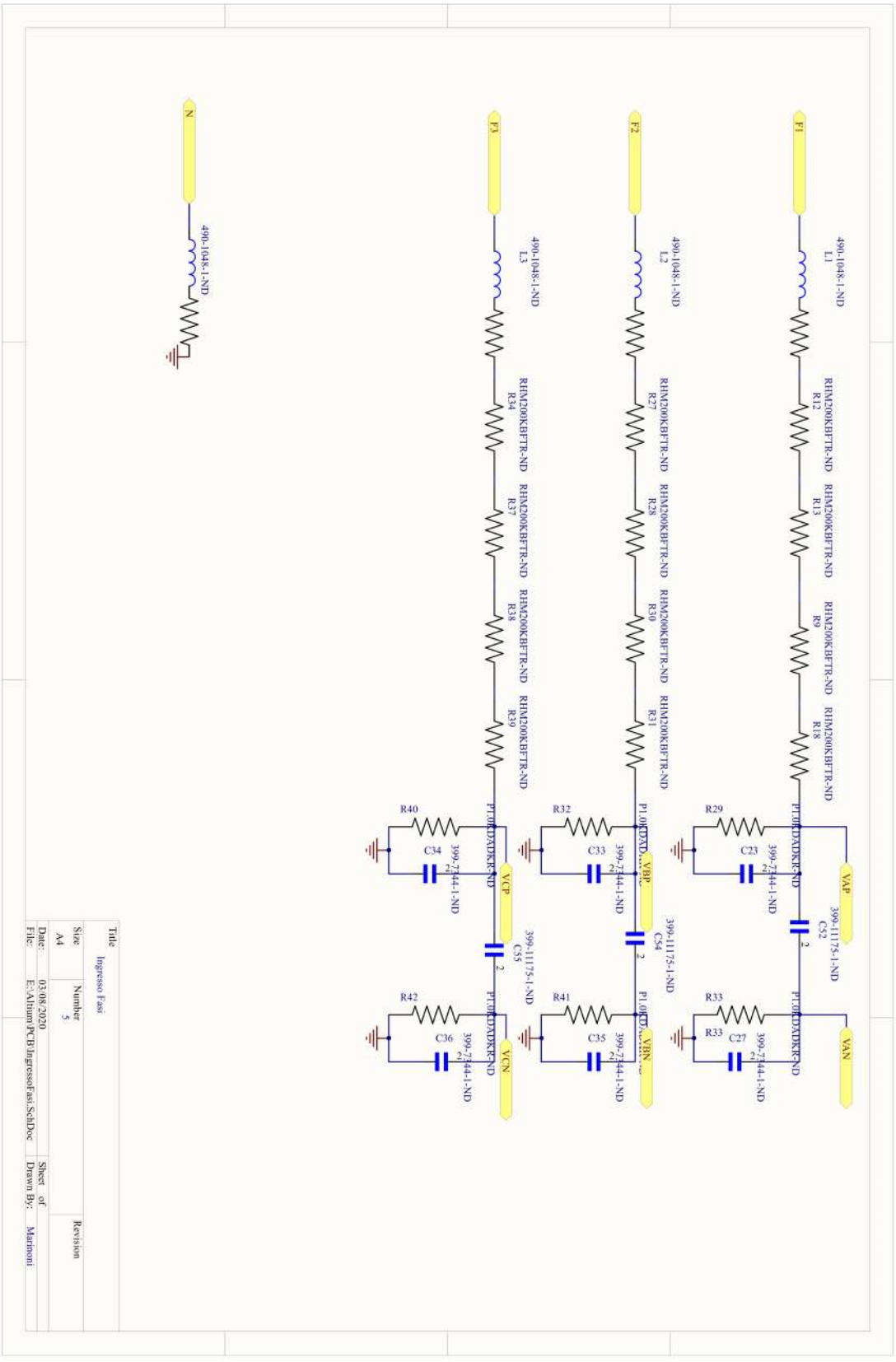

Figura 14.6: Circuito Ingresso Fasi Ade9000

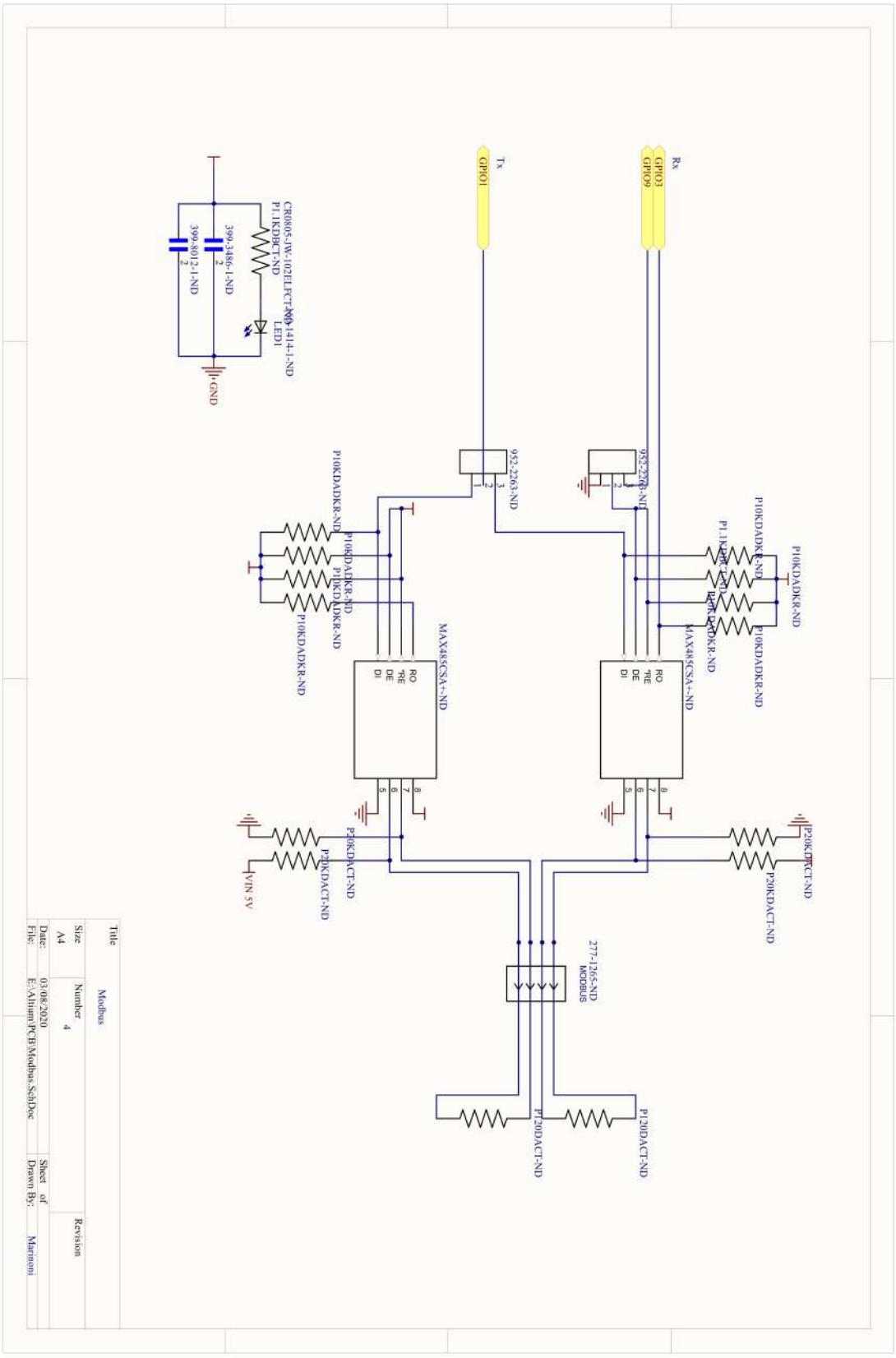

Figura 14.7: Circuito Rs485

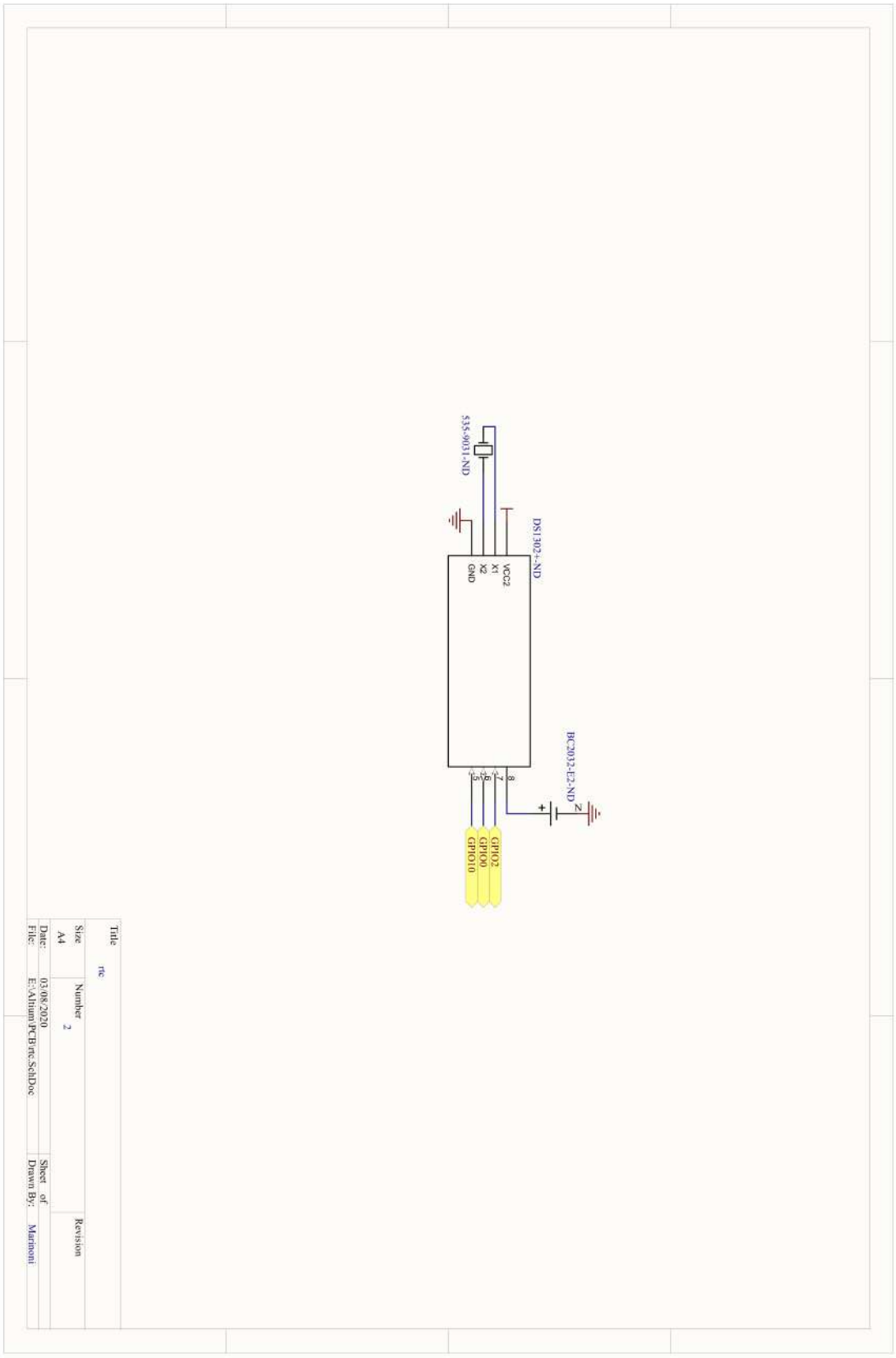

Figura 14.8: Circuito RTC

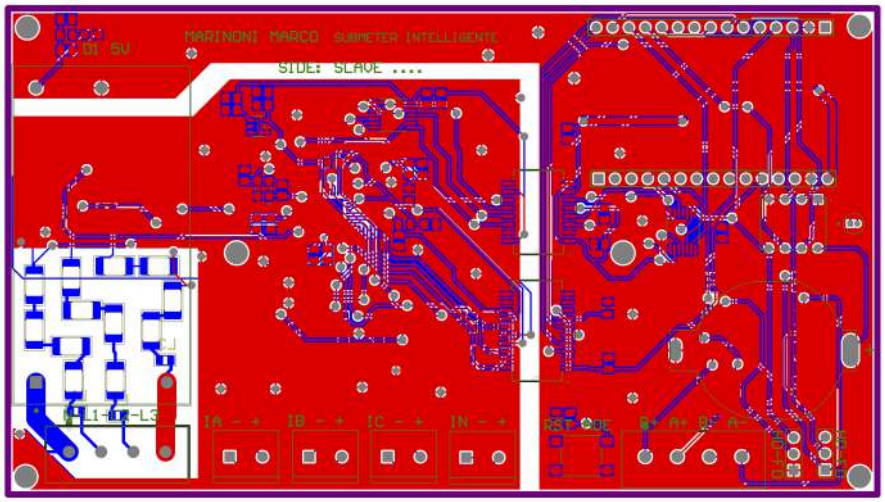

Figura 14.9: PCB top Layer

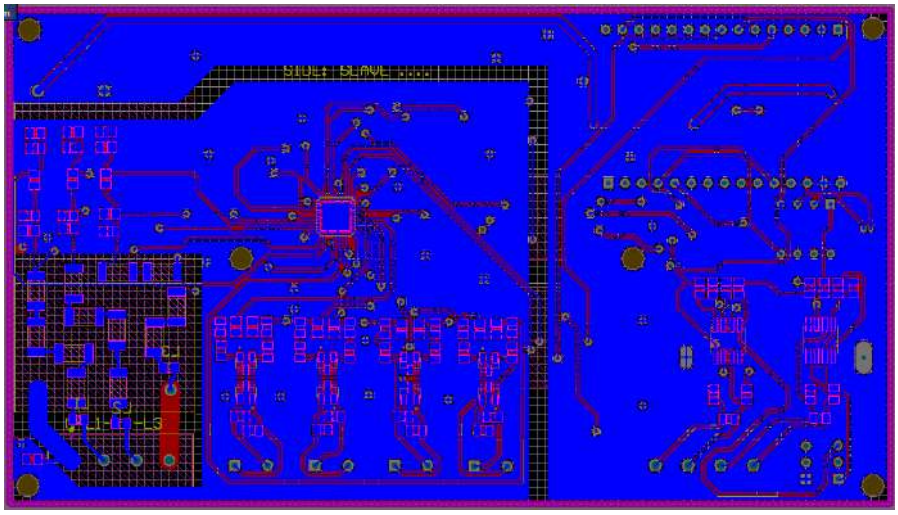

Figura 14.10: PCB bottom layer

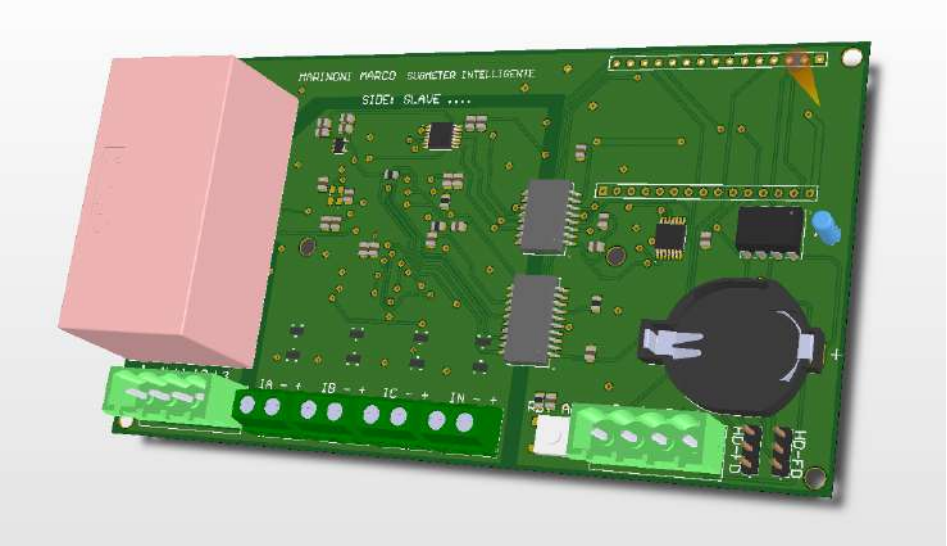

Figura 14.11: Modello 3d PCB slave

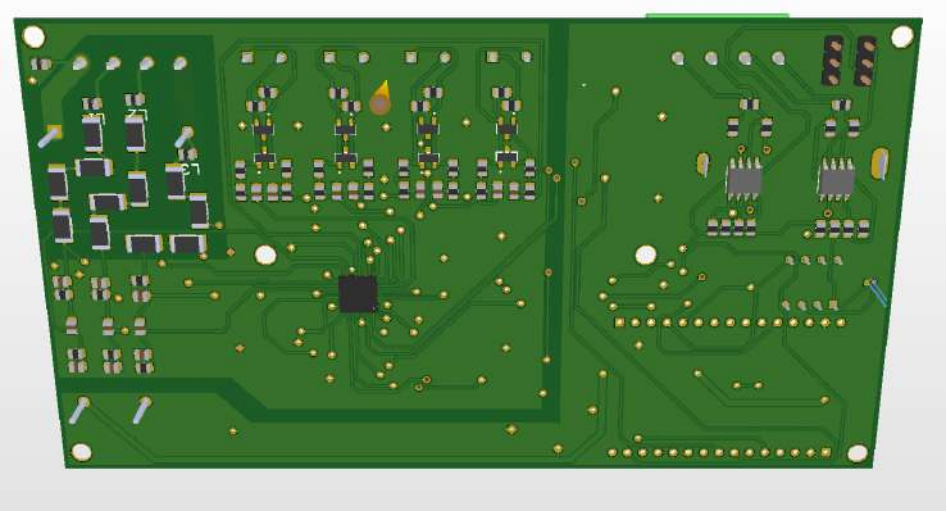

Figura 14.12: Modello 3d PCB slave-bottom

14.2 Master

La parte di master riguarda tutto ciò che ha ache fare con il circuito per l'alloggio del Compute Module, la ricezione del modbus rs485, adattamento per un collegamento alla rete internet, alimentazione come nel caso precedente il tutto riportato in modo più semplice rispetto allo slave su un unico pcb di dimensioni pari alle precedenti visto che anche questo deve essere adibito ad una installazione su barra DIN. E' previsto anche l'alloggio per lo schermo OLED e il pulsantino per switchare le varie schermati del nostro display. Anche in questo caso le prime pagine riguardano gli schematici mentre alla fine la parte di pcb con un buon modello 3D che rappresenta come effettivamente verrà realizzato. Lo schematico dell'alimentazione non è riportato in quanto è lo stesso dello slave quindi è solo una ripetizione.

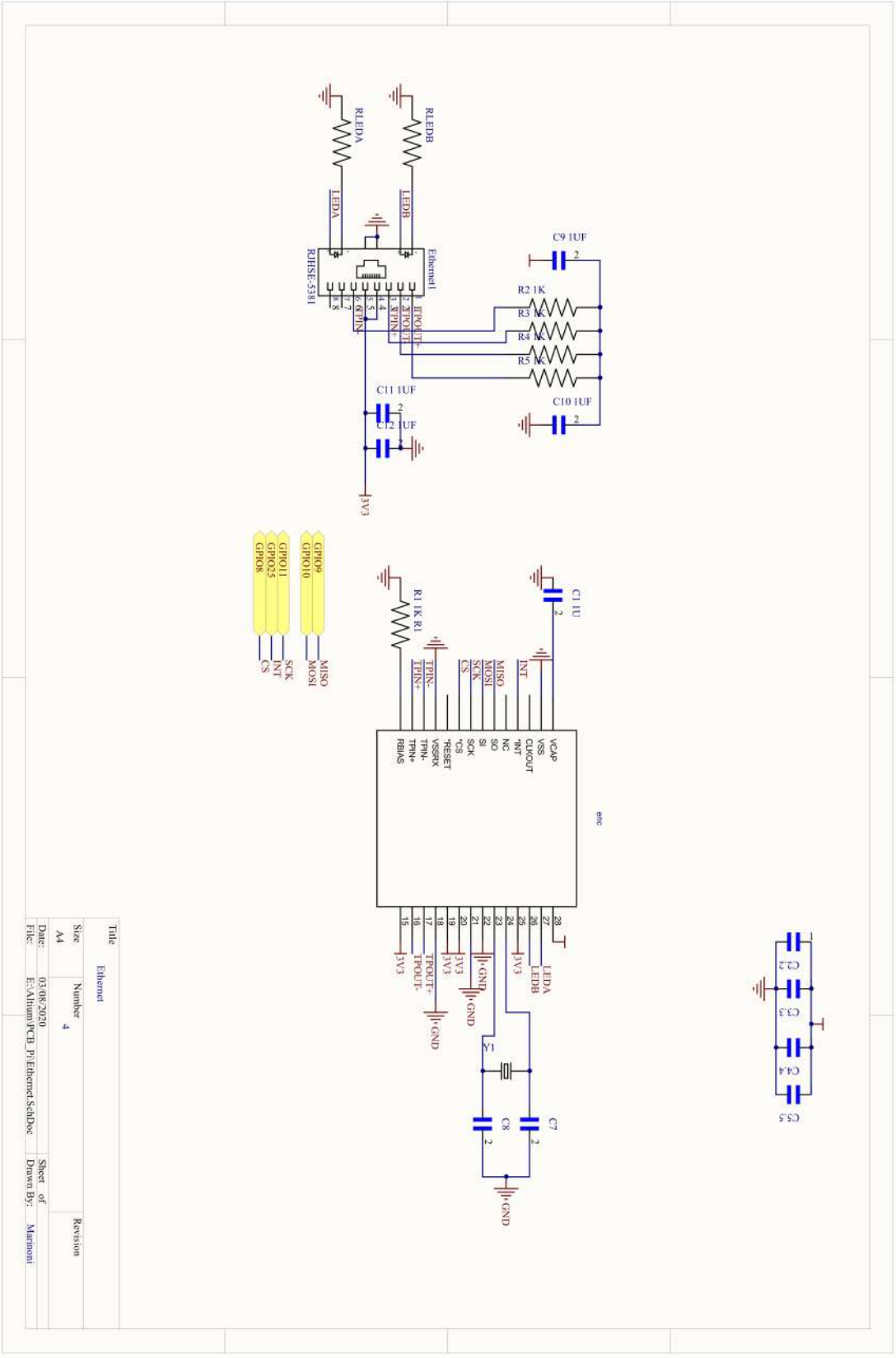

Figura 14.13: Collegamento ethernet ENC28j60

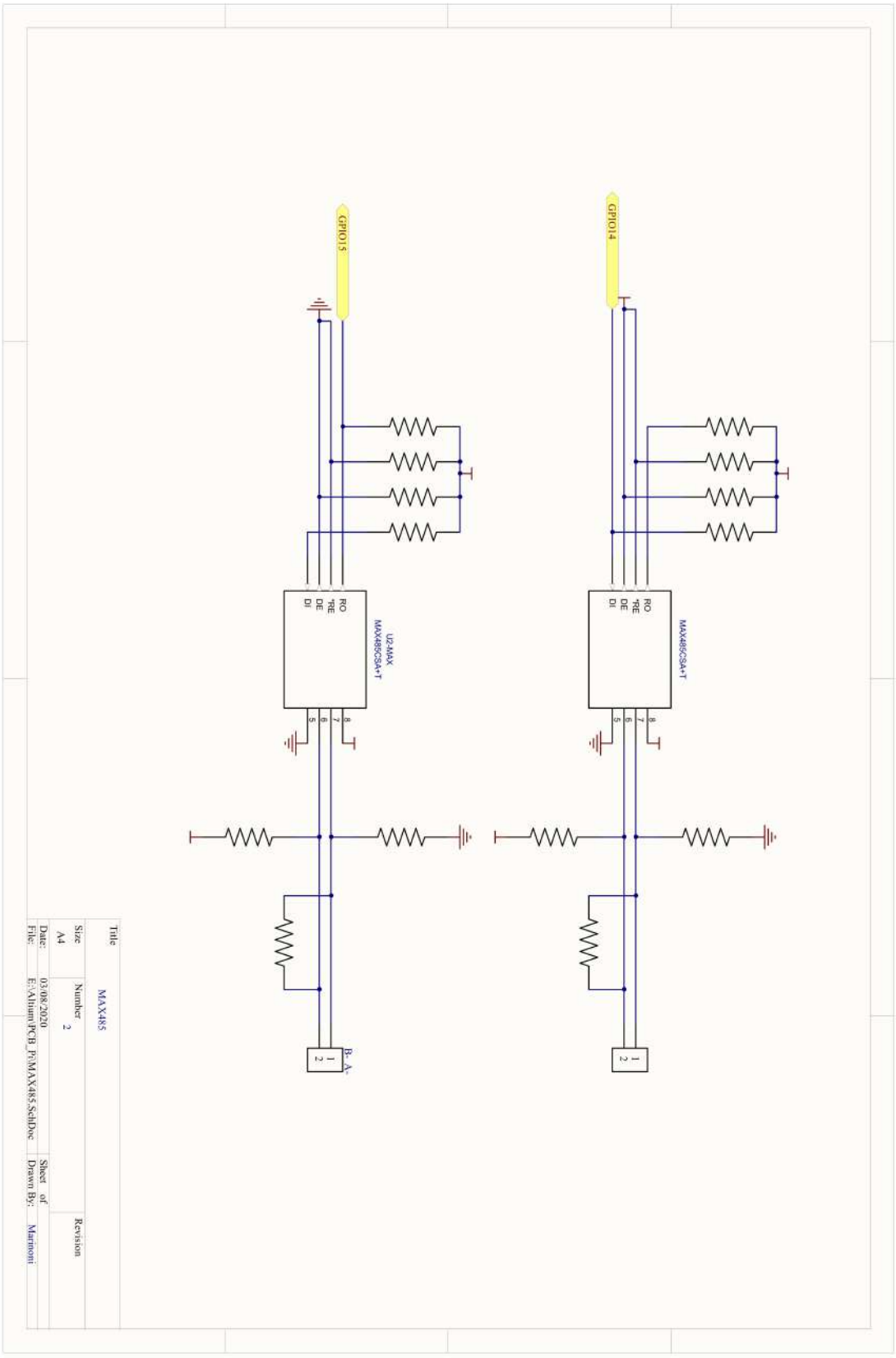

Figura 14.14: MODBUS rs485

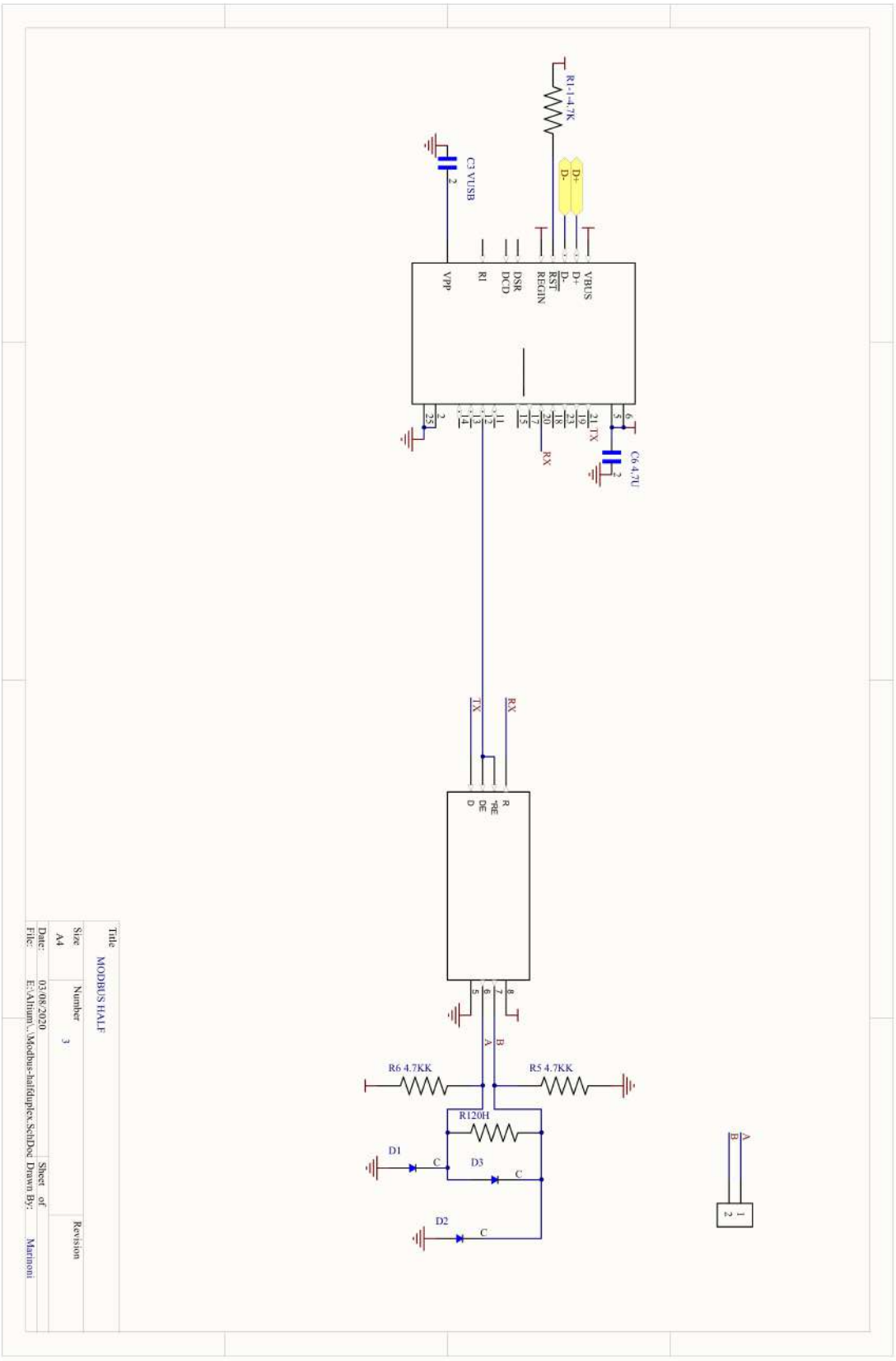

Figura 14.15: MODBUS half-duplex

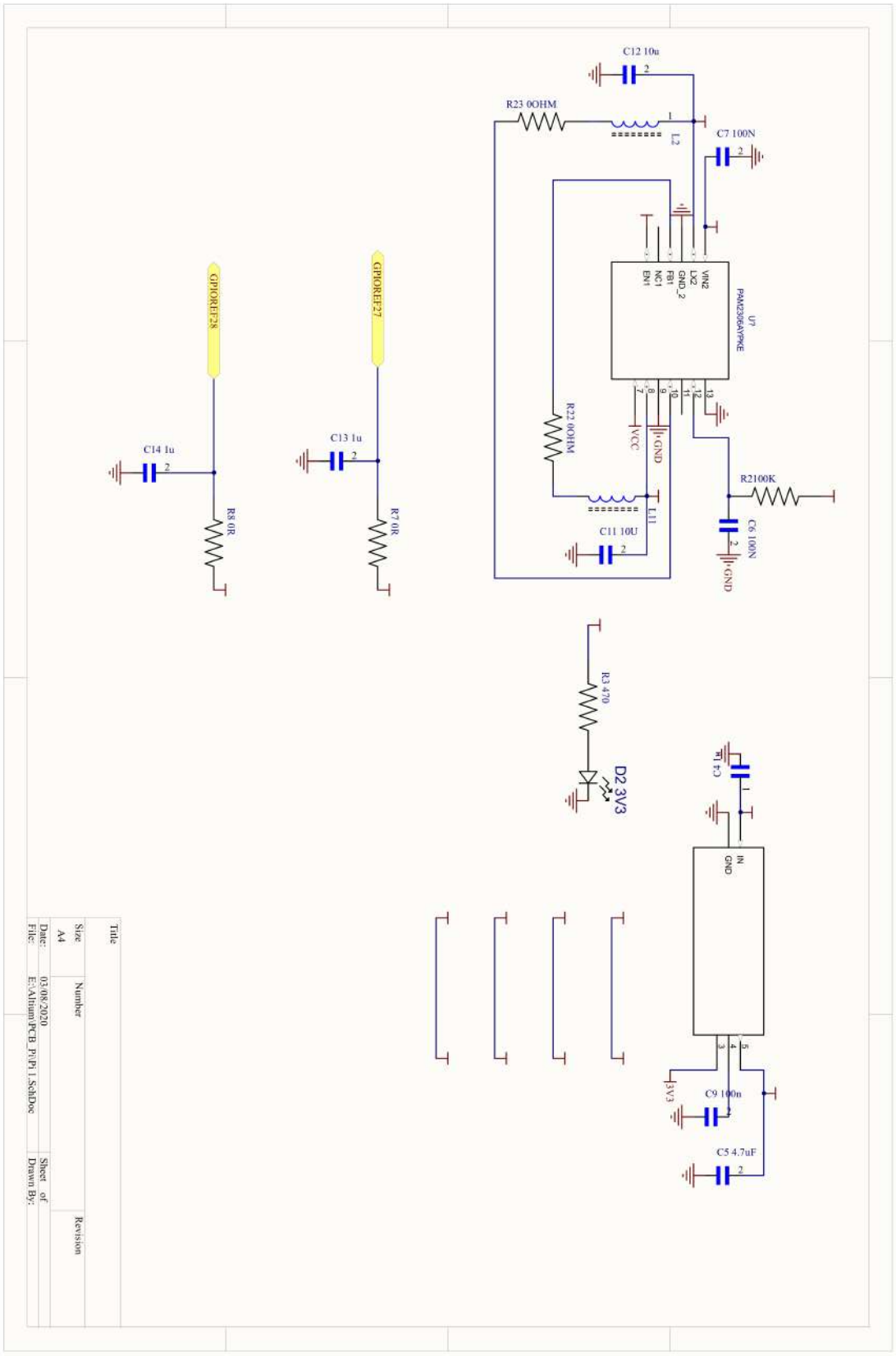

Figura 14.16: Raspberry board

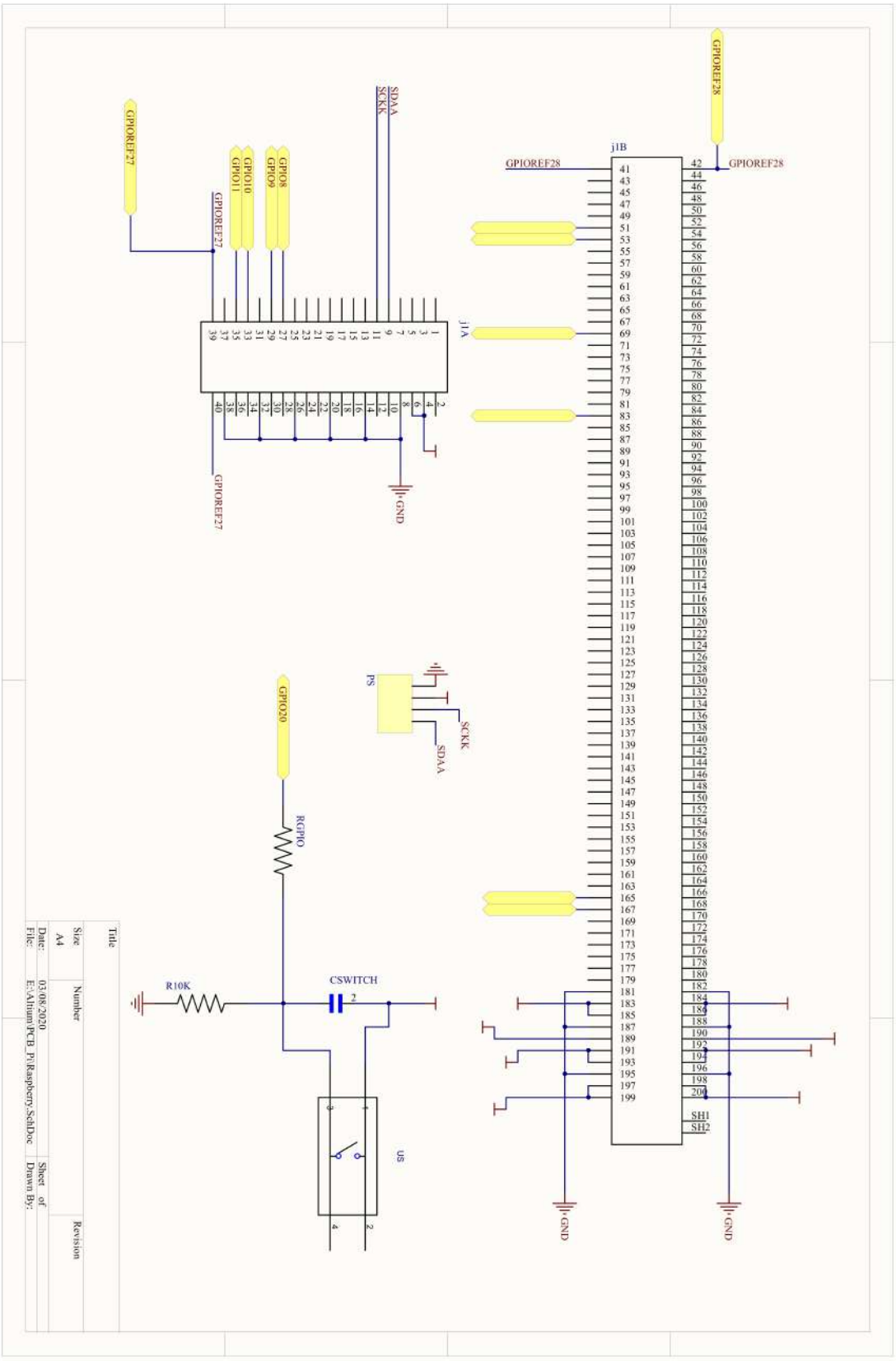

Figura 14.17: Raspberry pin

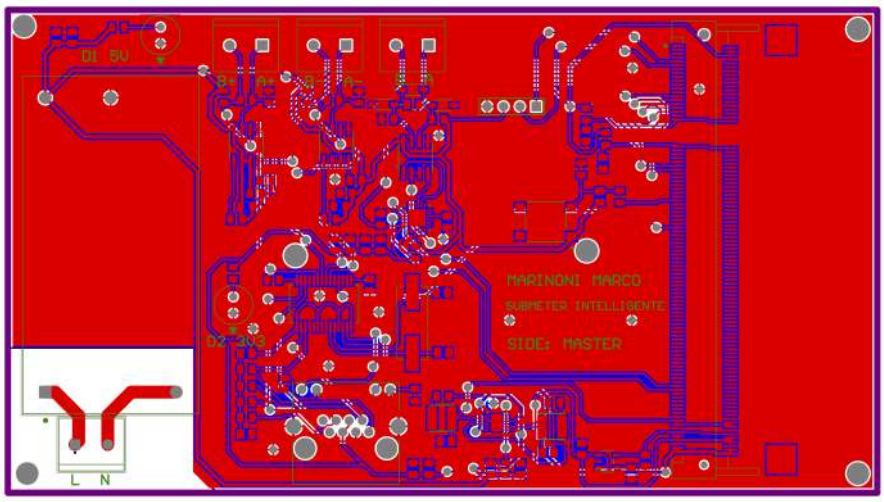

Figura 14.18

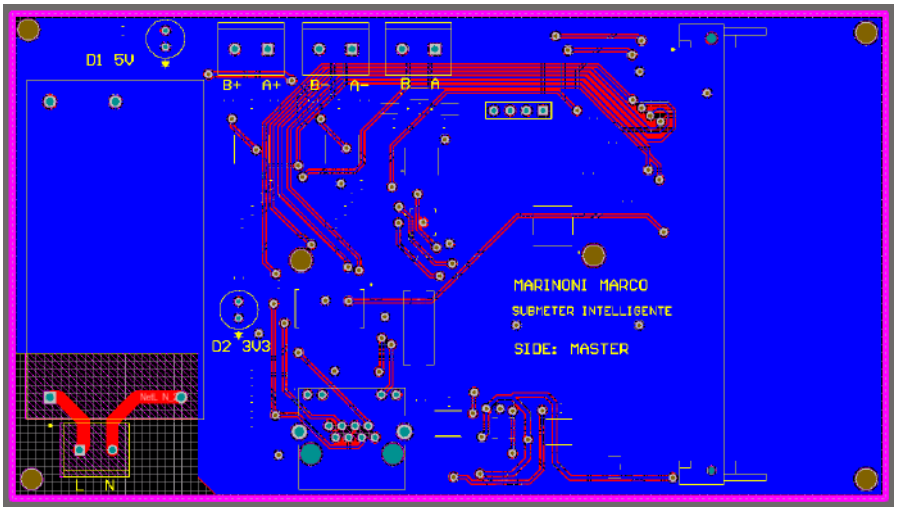

Figura 14.19

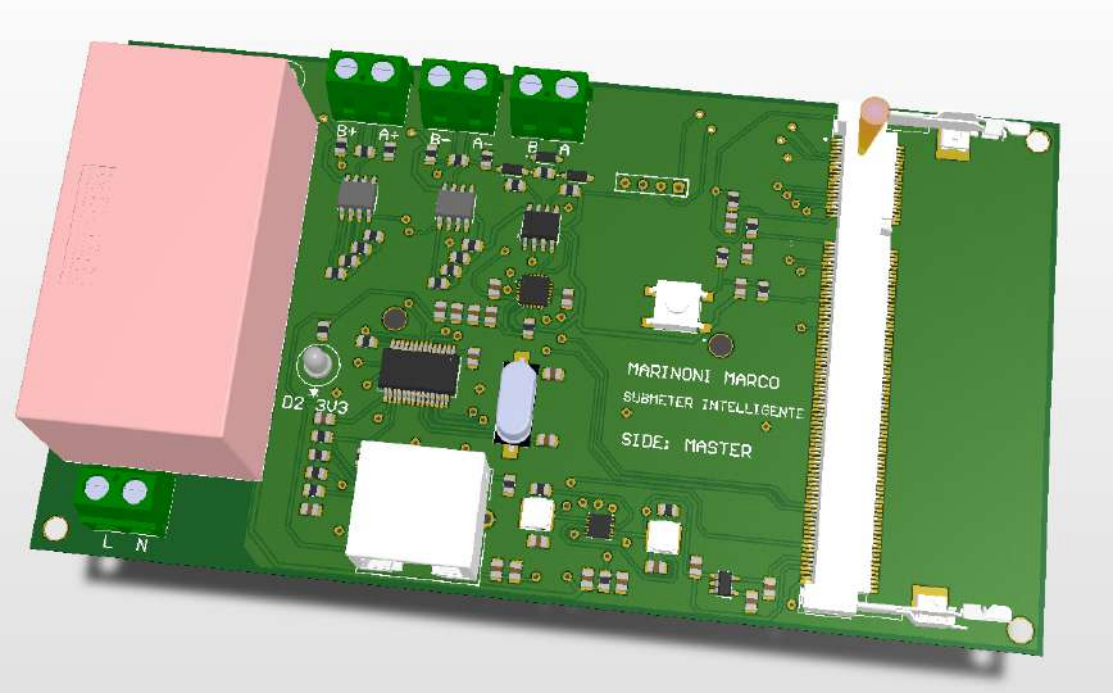

Figura 14.20: Modello 3D Master

Capitolo 15

Tabella registri Modbus Submeter

In questo capitolo è rappresentata la tabella dove vi sono elencati i corrispettivi indirizzi che ho assegnato a questo submeter al protocollo modbus. In pratica, sapendo che gli indirizzi modbus generali HOLDING REGISTERS vanno da 40001 a 49999, e che per usarli dato che si dichiara l'utilizzo degli holding resisters, si usano gli indirizzi riferiti solo a questo gruppo che vanno da 0000 a 9998. Ho deciso per scelta di utilizzare dal 10 al 50 per il primo slave, e per il secondo slave dal 51 al 90. In realtà non sono stati riempiti tutti al completo ma ho predisposto maggiore spazio per ulteriori modifiche e misure aggiunte che possono essere fatte. L'idea di questo progetto è solo di far capire come funziona l'intero sistema e tutte le modifiche apportabili. Per un elenco ancora più dettagliato bisognerebbe prendere come riferimento un datasheet di un'altro dispositivo modbus e creare un ordine simile dei registri. Nel mio caso ho circa 15/16 valori per slave che inserisco nel mio protocollo Modbus e i valori misurati sono denomitati con 's1' o 's2' a simboleggiare il fatto che lo studio è stato fatto su due slave, uno half duplex e l'altro full-duplex.

INDIRIZZI HOLDING REGISTERS	40001	49999
PARZIALI HOLDING REGISTERS	0	9998

REGISTRO	CONTENUTO
10	Va_s1
11	Vb_s1
12	Vc_s1
13	Ia_s1
14	Ib_s1
15	Ic_s1
16	F_s1
17	Cos_s1
18	Pa_s1
19	Pb_s1
20	Pc_s1
21	ora_s1
22	minuti_s1
23	secondi_s1
24	giorno_s1
25	mese_s1
26	anno_s1
50	Va_s2
51	Vb_s2
52	Vc_s2
53	Ia_s2
54	Ib_s2
55	Ic_s2
56	F_s2
57	Cos_s2
58	Pa_s2
59	Pb_s2
60	Pc_s2
61	ora_s2
62	minuti_s2
63	secondi_s2
64	giorno_s2
65	mese_s2
66	anno_s2

Figura 15.1: Tabella registri Submeter

Capitolo 16

Test e Risultato finale

In questo capitolo andremo un po' a vedere come si presenta il progetto che si è andati a realizzare. Vi sono due versioni, una prima demo con delle evaluation board perfettamente funzionanti,e una seconda demo fatta dai PCB creati da me che riporta alcuni piccoli problemi che verranno descritti successivamente. I primi test fatti sono stati per quanto riguarda l'utilizzo indiiduale del RTC con visualizzazione orario e dello schermo OLED con la visualizzazione di alcune scritte. Il passaggio successivo è stato creare un sistema MODBUS con dei registri in cui venivano inseriti dei valori. Qui di seguito è riportato lo screenshot di un programma (QMODBUS) che veniva utilizzato per leggere i vari registri e verificare che gli slave funzionassero correttamente. In pratica si inserisce ID dello slave, il tipo di registro da leggere, e quanti se ne vogliono leggere, e il gioco è fatto.

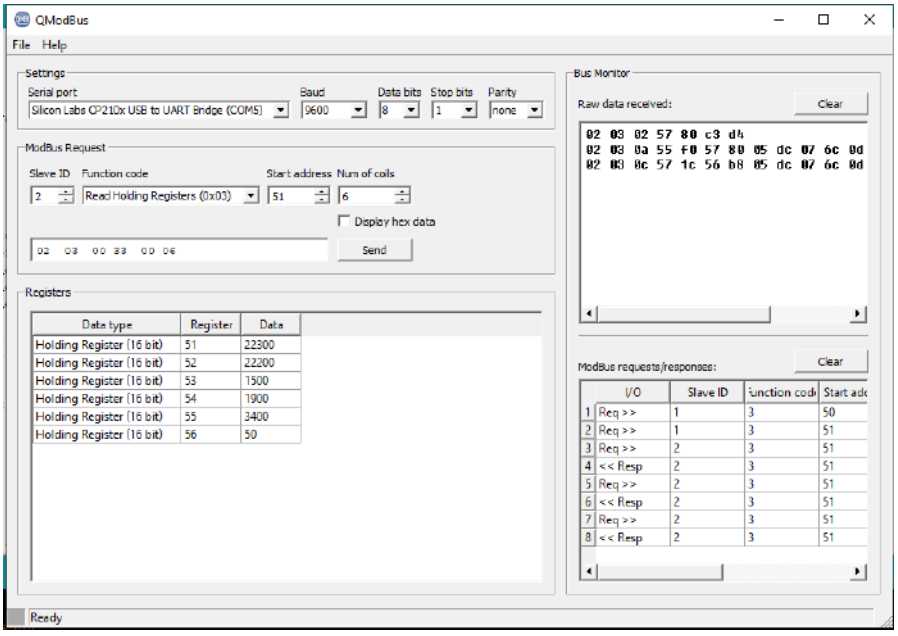

Figura 16.1: Qmodbus

Una volta verificato questo passaggio si è provato a realizzare la comunicazione modbus tramite slave e master(raspberry) nelle due modalità, full e half duplex. Ovviamente si è passati immediatamente alla realizzazione del database dei dati che venivano letti da modbus. Qui riportata è una piccola parte di database dove si possono vedere bene le colonne con i vari valori della tabella.

```
MariaDB [PowerMeter]> select* FROM SLAVE1;
+---------------------+---------+---------+---------+--------+------+------+---------+---------+---------+------+------+
| dt                  | Va      | Vb      | Vc      | Ia     | Ib   | Ic   | Pa      | Pb      | Pc      | Fr   | Pf   |
+---------------------+---------+---------+---------+--------+------+------+---------+---------+---------+------+------+
| 2020-07-16 14:18:08 |       0 |       0 |       0 |      0 |    0 |    0 |       0 |       0 |       0 |   50 |    0 |
| 2020-07-16 14:18:08 | 214.431 | 221.742 |  220.56 |     90 |  170 |  190 | 19.2988 | 37.6961 | 41.9063 |   50 |    0 |
| 2020-07-16 14:18:09 | 214.431 | 221.742 |  220.56 |     90 |  170 |  190 | 19.2988 | 37.6961 | 41.9063 |   50 |    0 |
| 2020-07-16 14:18:10 | 214.431 | 221.742 |  220.56 |     90 |  170 |  190 | 19.2988 | 37.6961 | 41.9063 |   50 |    0 |
| 2020-07-16 14:18:10 | 214.431 | 221.742 |  220.56 |     90 |  170 |  190 | 19.2988 | 37.6961 | 41.9063 |   50 |    0 |
| 2020-07-16 14:18:11 | 214.431 | 221.742 |  220.56 |     90 |  170 |  190 | 19.2988 | 37.6961 | 41.9063 |   50 |    0 |
| 2020-07-16 14:18:11 | 214.431 | 221.742 |  220.56 |     90 |  170 |  190 | 19.2988 | 37.6961 | 41.9063 |   50 |    0 |
| 2020-07-16 14:18:12 | 214.431 | 221.742 |  220.56 |     90 |  170 |  190 | 19.2988 | 37.6961 | 41.9063 |   50 |    0 |
| 2020-07-16 14:18:13 | 214.431 | 221.742 |  220.56 |     90 |  170 |  190 | 19.2988 | 37.6961 | 41.9063 |   50 |    0 |
| 2020-07-16 14:18:13 | 226.709 | 231.448 | 229.706 |    100 |  170 |  200 | 22.6709 | 39.3462 | 45.9412 |   50 |    0 |
| 2020-07-16 14:18:14 | 226.709 | 231.448 | 229.706 |    100 |  170 |  200 | 22.6709 | 39.3462 | 45.9412 |   50 |    0 |
| 2020-07-16 14:18:15 | 226.709 | 231.448 | 229.706 |    100 |  170 |  200 | 22.6709 | 39.3462 | 45.9412 |   50 |    0 |
| 2020-07-16 14:18:15 | 226.709 | 231.448 | 229.706 |    100 |  170 |  200 | 22.6709 | 39.3462 | 45.9412 |   50 |    0 |
| 2020-07-16 14:18:16 | 226.709 | 231.448 | 229.706 |    100 |  170 |  200 | 22.6709 | 39.3462 | 45.9412 |   50 |    0 |
| 2020-07-16 14:18:16 | 226.709 | 231.448 | 229.706 |    100 |  170 |  200 | 22.6709 | 39.3462 | 45.9412 |   50 |    0 |
| 2020-07-16 14:18:17 | 226.709 | 231.448 | 229.706 |    100 |  170 |  200 | 22.6709 | 39.3462 | 45.9412 |   50 |    0 |
| 2020-07-16 14:18:18 | 226.709 | 231.448 | 229.706 |    100 |  170 |  200 | 22.6709 | 39.3462 | 45.9412 |   50 |    0 |
| 2020-07-16 14:18:18 | 226.709 | 231.448 | 229.706 |    100 |  170 |  200 | 22.6709 | 39.3462 | 45.9412 |   50 |    0 |
| 2020-07-16 14:18:19 |  226.73 |  231.51 | 229.872 |    100 |  170 |  200 |  22.673 | 39.3567 | 45.9744 |   50 |    0 |
| 2020-07-16 14:18:19 |  226.73 |  231.51 | 229.872 |    100 |  170 |  200 |  22.673 | 39.3567 | 45.9744 |   50 |    0 |
| 2020-07-16 14:18:20 |  226.73 |  231.51 | 229.872 |    100 |  170 |  200 |  22.673 | 39.3567 | 45.9744 |   50 |    0 |
| 2020-07-16 14:18:20 |  226.73 |  231.51 | 229.872 |    100 |  170 |  200 |  22.673 | 39.3567 | 45.9744 |   50 |    0 |
| 2020-07-16 14:18:21 |  226.73 |  231.51 | 229.872 |    100 |  170 |  200 |  22.673 | 39.3567 | 45.9744 |   50 |    0 |
| 2020-07-16 14:18:22 |  226.73 |  231.51 | 229.872 |    100 |  170 |  200 |  22.673 | 39.3567 | 45.9744 |   50 |    0 |
| 2020-07-16 14:18:22 |  226.73 |  231.51 | 229.872 |    100 |  170 |  200 |  22.673 | 39.3567 | 45.9744 |   50 |    0 |
| 2020-07-16 14:18:23 |  226.73 |  231.51 | 229.872 |    100 |  170 |  200 |  22.673 | 39.3567 | 45.9744 |   50 |    0 |
| 2020-07-16 14:18:23 |  226.73 |  231.51 | 229.872 |    100 |  170 |  200 |  22.673 | 39.3567 | 45.9744 |   50 |    0 |
| 2020-07-16 14:18:24 |  226.74 | 231.583 | 229.965 |    100 |  170 |  200 |  22.674 | 39.3691 |  45.993 |   50 |    0 |
| 2020-07-16 14:18:25 |  226.74 | 231.583 | 229.965 |    100 |  170 |  200 |  22.674 | 39.3691 |  45.993 |   50 |    0 |
| 2020-07-16 14:18:25 |  226.74 | 231.583 | 229.965 |    100 |  170 |  200 |  22.674 | 39.3691 |  45.993 |   50 |    0 |
| 2020-07-16 14:18:26 |  226.74 | 231.583 | 229.965 |    100 |  170 |  200 |  22.674 | 39.3691 |  45.993 |   50 |    0 |
| 2020-07-16 14:18:26 |  226.74 | 231.583 | 229.965 |    100 |  170 |  200 |  22.674 | 39.3691 |  45.993 |   50 |    0 |
| 2020-07-16 14:18:27 |  226.74 | 231.583 | 229.965 |    100 |  170 |  200 |  22.674 | 39.3691 |  45.993 |   50 |    0 |
| 2020-07-16 14:18:28 |  226.74 | 231.583 | 229.965 |    100 |  170 |  200 |  22.674 | 39.3691 |  45.993 |   50 |    0 |
| 2020-07-16 14:18:28 |  226.74 | 231.583 | 229.965 |    100 |  170 |  200 |  22.674 | 39.3691 |  45.993 |   50 |    0 |
```

Figura 16.2: Mysql database

Nel mio caso nelle colonne vi sono presenti valori di tenzione, corrente ecc... .
Una volta fatto ciò si è passati all'utilizzo di ADE9000 che ha riportato una serie di problemi ma che un po' alla volta si sono risolti tutti. In questo modo è bastato mettere tutti insieme (RTC,oled,modbus,ade9000,mysql,grafana) per avere una prima demo completa e funzionante.
L'intero sistema è riportato qui nell'immagine a seguire.

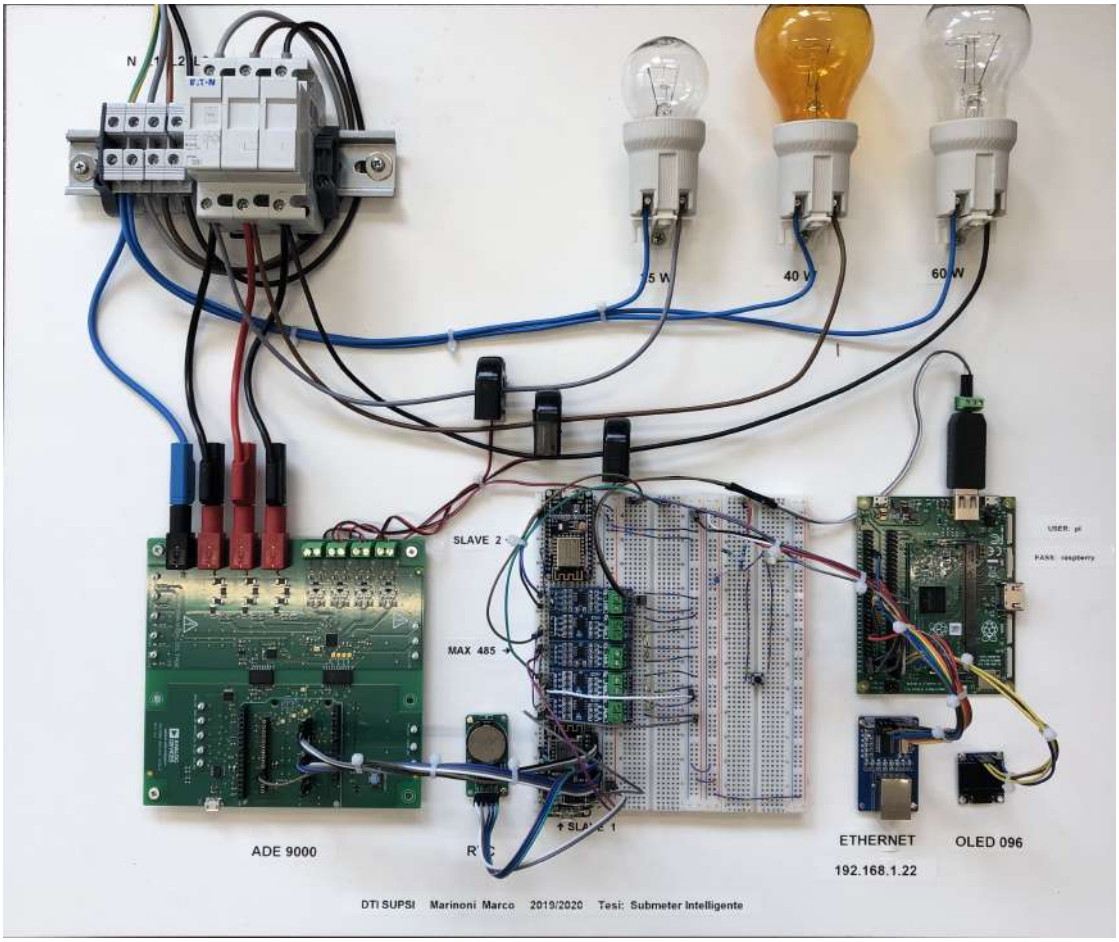

Figura 16.3: Demo

Nella Demo vi sono dei morsetti di partenza e una protezione tramite fusibili da 2A (correnti basse, carichi piccoli). Il test legge tensione, corrente, fase, frequenza e potenza insieme a carichi diversi rispettivamente da 25,40 e 60W.

16.1 PCB

Qui di seguito riportiamo la foto dei pcb, dove si può vedere in alto il master, mentre in mezzo e in basso il fronte e retro dei due slave.

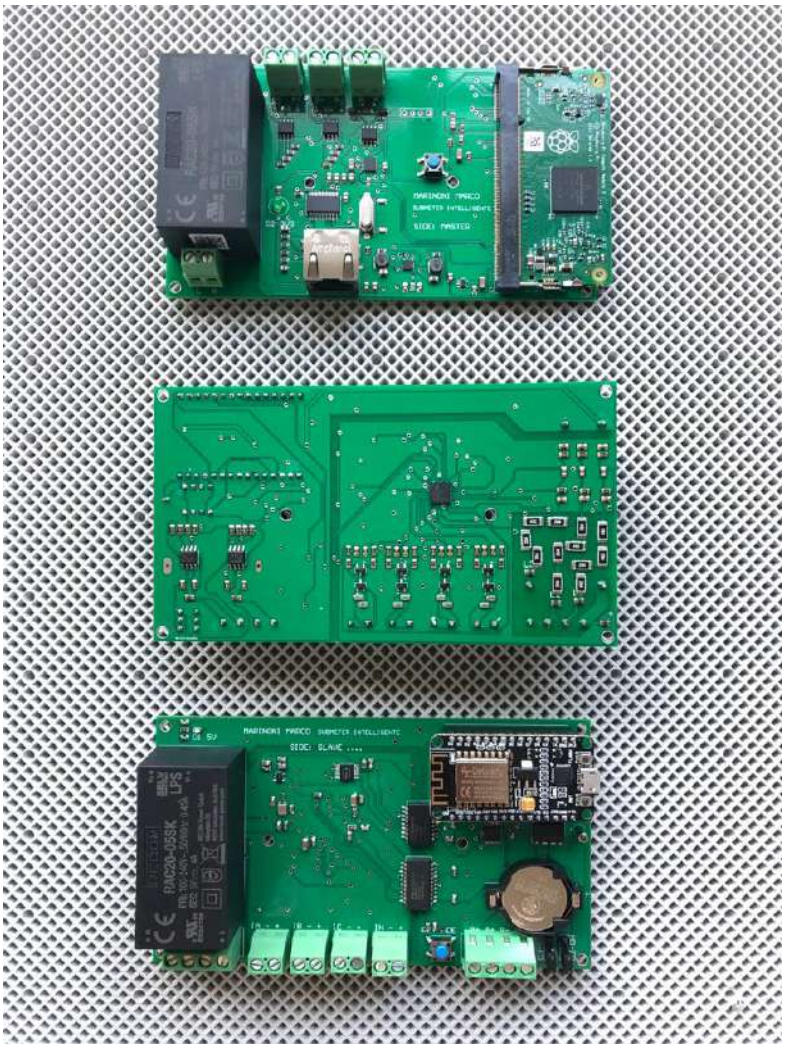

Figura 16.4: Demo

Gli slave possiedono tutti i componenti che ha la demo con la possibilità di selezionare tramite jumper la modalità del modbus da utilizzare. Il master invece ha tutti gli ingressi modbus, alimentazione 220V e ingresso per la rete ethernet con l'aggiunta di un pulsante e schermo OLED da disporre a distanza su case DIN. Questi PCB non sono funzionanti completamente ma riportano alcuni problemi. Sono stati fatti errori di tempistica visto che si pensava di far realizzare i PCB da PCBWAY con saldatura compresa ma visto che ci mettevano troppo tempo ho dovuto fare il tutto a mano e alcuni componenti risultano essere davvero molto piccoli. La scelta di utilizzare una alimentazione dal 220V è dovuta al fatto che questi dispotivivi sono installati dove vi sono di norma tensioni comuni e non basse

tensioni come 12 o 24V. Avrebbe avuto senso avere ingresso a 24 o 12V nel caso in cui esempio in una palazzina con tanti slave, si ha un trasformatore nudo e crudo che generi bassa tensione e che venga distribuita ovunque risparmiando n. alimentatori 220/5V di n. slave. Nel nostro caso la parte più sensata è stata avere input ad alta tensione visto che comunque gia viene utilizzata per la misura.

16.2 Case DIN

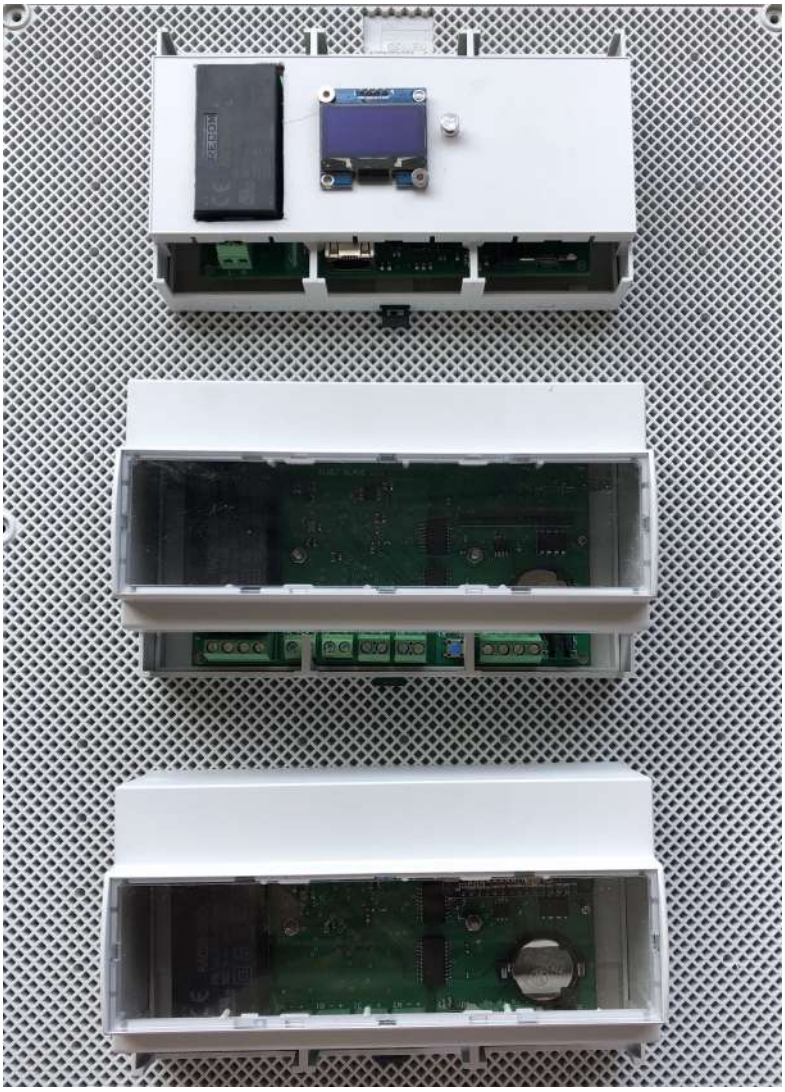

Figura 16.5: Case DIN

Come Case DIN si è riusciti a trovare un'azienda che si chiama ItalTronic, che realizza case plastici DIN di ogni dimensione e forma e per ogni utilizzo. Nel mio caso ho trovato case da 9 Moduli, alti per gli slave e bassi per il master, perfetti per l'utilizzo e pratici per il collegamente dei vari cavi ai connettori. Un piccolo dettaglio è la realizzazione dello 'schermo' in plexiglass trasparente per gli slave che oltre a isolare dalla scheda permette di vedere cosa vi è all'interno. Le dimensioni diversificate ovviamente sono state fatte per provare i diversi tipi e sperimentare come si potrebbe adattare la scheda per occupare diversi spazi ridotti.

16.3 Grafana

Abbiamo gia spiegato brevemente nei capitoli precedenti cosa sia grafana e vediamo ora in alcuni brevi immagini cosa si può fare.

Figura 16.6: Grafana image 1

Come si vede qui sopra, vi è una prima visualizzazione delle tensioni lette nell'ultimo secondo per quanto riguarda la fase 1,2 e 3 e sotto si può leggere la corrente in mA della fase uno con relativo grafico costante visto che si tratta di accensione e spegnimento di una lampadina.

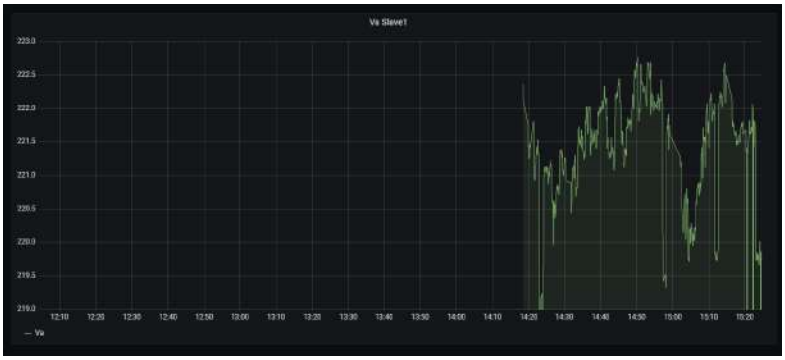

Figura 16.7: Grafana image 2

In questa immagine invece è presente una lettura di circa un'ora della tensione in cui si possono ben vedere gli sbalzi che avvengono e che confermano che la tensione non è sempre costante nel tempo ma ha delle variazioni.
Tramite grafana si può visualizzare ogni genere di dato e 'printarlo' graficamente nel modo più strano e particolare che vogliamo.
A seguire nella prossima immagine alcuni dettagli di analisi più specifiche fatte sulla rete. Oltre a tutto ciò ve ne sono molti altri che non sto a riportare ma che sono presenti sul server grafana.

Figura 16.8: Tensioni trifase

Figura 16.9: Dati Slave

Capitolo 17

Sitografia

Qui di seguito riportati alcuni link utili utilizzati durante la realizzazione del progetto. I link includono sia datasheet, siti web e tutorial con video per apprendere alcuni concetti.

- **ESP8266 con ade9000**
  https://www.analog.com/media/en/technical-documentation/user-guides/ev-ade9000shieldz-ug-1170.pdf
- **Creare Database MYSQL**
  https://www.youtube.com/watch?v=iycNe-ZThOM
- **Utilizzo ENC28j60**
  https://www.instructables.com/id/Super-Cheap-Ethernet-for-the-Raspberry-Pi/
- **Assegnazione IP fisso Raspberry**
  https://www.ionos.it/digitalguide/server/configurazione/raspberry-pi-assegnazione-di-un-indirizzo-ip-fisso/
- **Guida per Raspberry**
  https://www.raspberrypi.org/documentation/hardware/computemodule/cm-peri-sw-guide.md
- **Istruzioni Mysql**
  https://www.w3schools.com/python/python_mysql_create_db.asp
- **Utilizzo Mysql**
  https://www.programmareinpython.it/video-corso-python-intermedio/ambienti-virtuali-con-venv-virtual-environments/
- **Serie per spiegazione modbus**
  https://www.youtube.com/watch?v=YBiXag_Dg0A
- **ADE9000 Analog Devices**
  http://www.analog.com/en/products/ade9000.htmlproduct-overview

- **ADE9000 datasheet**
  http://www.analog.com/media/en/technical-documentation/data-sheets/ADE9000.pdf
- **Guida Altium**
  https://techdocs.altium.com/printpdf/231428
- **Guida installazione Raspbian**
  https://www.raspberrypi.org/documentation/installation/installing-images/README.md
- **ADE9000 manuale tecnico**
  http://www.analog.com/media/en/technical-documentation/user-guides/ADE9000-UG-1098.pdf
- **MAX 485 datasheet**
  https://datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf

Capitolo 18

Budget

Il budget messo a disposizione dal professore per una demo base è tra i 600 e 700CHF. Sono stati spesi tutti per riuscire a realizzare un primo prototipo. Successivamente per avere un progetto maggiormente completo si è voluti procedere con i PCB e quindi con ulteriori spese. Il mio relatore Davide Rivola si è molto gentilmente messo a disposizione di sostenere lui le spese aggiuntive per non dare carico alla scuola.
Qui di seguito riportate le spese fatte da amazon, digikey e altri siti.

Pos	Order Code	Description	Qtty	Pr/Un	-(%)	Total
1	https://www.ama	ESP8266 (x3)	1	18,99		18,99
2	https://www.ama	USB RS485 CONVERTER	1	7,99		7,99
3	https://www.ama	RS485 TO TTL CONVERTER (x5)	1	7,99		7,99
4	https://www.ama	ENC28J60	1	8,99		8,99
5	https://www.ama	RTC DS3231	1	6,49		6,49
6						0,00
7						0,00
8						0,00
9						0,00
10						0,00
11						0,00
12						0,00
13						0,00
14						0,00
15						0,00
						50,45
				Total (EUR)		50,45

Figura 18.1: Ordini Amazon

Pos	Order Code	Description	Qtty	Pr/Un	-(%)	Total
1	EV-ADE9000SHIELDZ-ND	ADE9000 ARDUINO EVALUATION BOARD	1	330,67		330,67
2	1690-1018-ND	COMPUTE MODULE RASPBERRYPI3 I/O	1	119,35		119,35
3	1690-1016-ND	RASPBERRY PI COMPUTE 3 BCM2837	1	47,53		47,53
4						0,00
5						0,00
6						0,00
7						0,00
8						0,00
9						0,00
10						0,00
11						0,00
12						0,00
13						0,00
14						0,00
15						0,00
				Subtotal (EUR)		497,55
				IVA	22,00%	109,46
				Total (EUR)		**607,01**

Figura 18.2: Ordini Digikey 1

NOME	N.	QUANTITA'	PREZZO SINGOLO	TOT	
PIN HEADER 15 POS	2	4	0,83	3,32	https://ww
DS1302+	1	2	3,29	6,58	https://ww
PORTA BATTERIA	1	2	0,8	1,6	https://ww
32,7 KHZ CRISTALLO	1	2	0,18	0,36	https://ww
ALIMENTATORE	1	3	10,09	30,27	https://ww
CONNETTORI 4 POS	2	4	2,22	8,88	https://ww
3PIN	2	8	0,15	1,2	https://ww
MAX485	2	6	2,67	16,02	https://ww
LED	1	2	0,26	0,52	https://ww
FERRITE 1,5 KHM	4	8	0,16	1,28	https://ww
CONN 2 PIN	4	8	0,81	6,48	https://ww
DIODE ARRAY MMBD41	8	16	0,26	4,16	https://ww
ECS-245.7-12-33Q-JE	1	2	0,63	1,26	https://ww
ADE9000	1	2	11,39	22,78	https://ww
ADUM4151BRIZ	1	2	8,81	17,62	https://ww
PULSANTE	1	3	0,24	0,72	https://ww
SN74LV4T125PWR	2	4	0,74	2,96	https://ww
ADUM6404ARIZ	1	2	10,68	21,36	https://ww
ADP122AUJZ-3.3-R7	1	2	0,98	1,96	https://ww
SN75176ADR	1	1	1,21	1,21	https://ww
CP2104-F03-GMR		1	1,4	1,4	https://ww
1N4148W-7-F	5	10	0,14	1,4	https://ww
RETE		1	1,28	1,28	https://ww
ENC28		1	2,41	2,41	https://ww
ECS-250-18-5PX-F-TR		1	0,28	0,28	https://ww
DDR2 SOD		1	5,47	5,47	https://ww
AP7115-25SEG-7		1	0,32	0,32	https://ww
REG 3V3 1V8	1	2	0,61	1,22	https://ww
INDUTTORE	1	2	0,4	0,8	https://ww
RES				5	
			TOT	170,12	

Figura 18.3: Ordini Digikey 2

Come ultima tabella è riportato il totale con aggiunta delle spese avuto da PCBWAY, digikey e Italtronic per l'acquisto di case DIN.

Marinoni Marco		Giovedi' 6 Agosto		
Budget stabilito con relatore:			600/700CHF	
	Ordini effettuati		**Spese [€]**	**Data**
1	Digikey		610,85	13/05/20
2	Amazon		50,36	14/05/20
		Tot	661,21	EURO
	Ordini Da effettuare		**Spese [€]**	**Data**
1	Digikey		182	
2	PCBway		75	
3	Eventuale acquisto case DIN		50	
		Tot	257	EURO
		TOTALE	918,21	EURO

Figura 18.4: Ordini Totale

Ci troviamo di fronte ad un superamento del budget. Inizialmente durante i primi colloqui con il relatore si era stabilita una cifra base ma senza preoccupazioni del caso di andare a superare il limite in quanto visto la situazione iniziale di COVID non si sapeva come si poteva sviluppare il tutto.

Capitolo 19

Conclusioni

Sono orgoglioso del lavoro di tesi svolto. In particolare, apprezzo il fatto di essere riuscito a terminare la demo nei limiti di tempo richiesti dal docente nonostante le difficoltà avute dato questo periodo del 2020.
L'unica parte non funzionante al 100% di questo progetto sono i PCB creati e sviluppati. Per la parte slave si ha avuto difficoltà per quanto riguarda la saldatura di componenti molto piccoli tra cui ADE9000. La parte di modbus e RTC è stata testata e funzionante per entrambi, slave numero 1 e 2. I circuiti e i componenti utilizzati sono corretti visto che si sono seguiti i vari schematici della case madri dei componenti. Purtroppo è stato fatto un errore in quanto si pensava di fare realizzare i PCB compresa la saldatura dei componenti da PCBWAY e nonostante si era già pronti per procedere un mese prima della consegna del progetto, le festività di agosto ,le tempistiche (PCBWAY impiega tra i 20/25 giorni per realizzare pcb e saldarli) e i pagamenti (dovuta a superamento del budget) hanno reso impossibile ciò per cui si è stati costretti a saldare a mano alcuni componenti di dimensioni parecchie ridotte soprattutto visto che con il forno presente nel laboratorio di scuola non è fattibile la realizzazione di schede dual-layer visto che i componenti sul lato posteriore si vanno a staccare per gravità. Per la parte di master vi è solo il problema di un solo componente di cui non si è riuscito bene a saldare ma non è stato fatto un test completo visto che avendo terminato il pcb solo pochi giorni prima della consegna, si voleva evitare di inserire la raspberry compute module visto che poteva subire danni e non si dispone di un altro articolo gia pronto per la sostituzione. Per ovviare a questi problemi, oltre che a far realizzare i PCB direttamente da fabbriche professioniste, si potrebbero cercare componenti di dimensioni maggiori in modo da saldarli a mano in modo più semplice. Per riassumere basterebbero piccoli accorgimenti e qualche giorno in più per dei PCB funzionanti completamente e quindi un progetto che soddisfa le richieste al 100%.

Inoltre sono soddisfatto delle nuove abilità e conoscenze apprese, oltre a essere riuscito a gestire il tutto anche lavorando da solo.

Sono state incontrate alcune difficoltà riguardanti la gestione del modbus visto che gli slave

non riuscivano e inserire correttamente i valori nei registri corretti. Il tutto è stato risolto studiando al meglio due delle tante librerie modbus trovare in rete e adattate al proprio progetto per funzionare al meglio. Un'atra difficoltà incontrata che ha portato via tanto tempo è stata la comunicazione errata con lo shield ADE9000 in quanto vi erano molti registri da impostare correttamente e si è risolto studiando registro per registro durante l'inizializzazione e inserendo i parametri corretti.
Inoltre ho creato un mio GitHub personale, cioè una piattaforma online su cui è disponibile il mio progetto svolto con schematici e codici scritti visto che si tratta comunque di argomenti di attualità. Non è ancora disponibile pubblico visto che è ancora in fase di aggiunta di ultimi particolari e sistemazione delle cartelle visto che prima si è data la priorità a terminare il progetto per la data di consegna.
Per questo progetto sarebbe interessante anche aggiungere eventuali modifiche:

- Una serie di led su pcb che indicano determinate azioni
- Con un budget molto più alto una riduzione della dimensione del pcb utilizzando ad esempio più layer e componenti sempre più piccoli e saldati direttamente dai produttori di pcb.
- Un ulteriore implementazione di altre misure che ADE9000 è in grado di fare con calcoli delle relative incertezze
- La realizzazione di PCB con micro direttamente ONBOARD e non tramite dei connettori.