Logo

Description automatically generated

**Università degli Studi di Trento**

**Dipartimento di Ingegneria e Scienza dell’Informazione**

**Corso di Progettazione Sistemi Elettronici**

***Titolare del Corso: Ing. Michele Corrà***

***Relazione finale del corso***

***Studente: Paganin Andrea***

***Matricola: 209926***

***Anno Accademico 2021/2022***

Contents

[1. INTRODUZIONE 4](#_Toc104534932)

[1.1 Descrizione del Corso 4](#_Toc104534933)

[1.2 Il Raspberry RP2040 e l’EVB Raspberry Pi Pico 5](#_Toc104534934)

[1.3 DESCRIZIONE GENERALE E TOOL IMPIEGATI 5](#_Toc104534935)

[1.3.1 KiCad 6 5](#_Toc104534936)

[1.3.2 Github 5](#_Toc104534937)

[1.3.3 Firmware - Visual Studio Code 6](#_Toc104534938)

[2. SPECIFICHE 7](#_Toc104534939)

[2.1 Il microcontrollore 7](#_Toc104534940)

[2.2 Il circuito LAN 8](#_Toc104534941)

[2.3 Il circuito di alimentazione 8](#_Toc104534942)

[2.4 Il modulo RTC 8](#_Toc104534943)

[2.5 Il modulo GPS 8](#_Toc104534944)

[3. PROGETTAZIONE CAD SCHEMATIC 9](#_Toc104534945)

[3.1 Alimentazione 10](#_Toc104534946)

[3.2 Il microcontrollore RP2040 12](#_Toc104534947)

[3.2.1 Il clock 12](#_Toc104534948)

[3.2.2 La memoria FLASH 13](#_Toc104534949)

[3.2.3 Le periferiche di interesse 13](#_Toc104534950)

[3.2.3.1 GPIO 14](#_Toc104534951)

[3.2.3.2 Timer 14](#_Toc104534952)

[3.2.3.3 USB 14](#_Toc104534953)

[3.2.3.4 UART 14](#_Toc104534954)

[3.2.3.5 I2C 15](#_Toc104534955)

[3.2.3.6 PIO 15](#_Toc104534956)

[3.3 Il circuito LAN 16](#_Toc104534957)

[3.4 Il modulo GPS 18](#_Toc104534958)

[3.5 Il modulo RTC 19](#_Toc104534959)

[4. Progettazione CAD PCB layout 20](#_Toc104534960)

[4.1 Piani gerber 20](#_Toc104534961)

[4.2 Alloggiamento 20](#_Toc104534962)

[5. Assemblaggio manuale in laboratorio e test funzionali 20](#_Toc104534963)

[6. Programmazione fw e utilizzo pratico (con o senza sw) 22](#_Toc104534964)

[6.1 Comunicazione con il transceiver Ethernet LAN8742 23](#_Toc104534965)

[6.2 Comunicazione con il modulo RTC 24](#_Toc104534966)

[6.3 Comunicazione con il GPS 26](#_Toc104534967)

[7. Approfondimenti 29](#_Toc104534968)

[8. Allegati 29](#_Toc104534969)

# INTRODUZIONE

## Descrizione del Corso

Il corso “Progettazione e Prototipazione di Sistemi Elettronici”, offerto dal Dipartimento di Ingegneria e Scienza dell’Informazione presso l’Università di Trento e tenuto dall’Ing. Corrà, si svolge nell’arco di due semestri e tratta i passi fondamentali dello sviluppo di un dispositivo elettronico moderno, con particolare focalizzazione sui sistemi a microcontrollore.

Durante il primo semestre, l’attenzione è stata data a una scheda già esistente, chiamata scheda PSE, che presenta i blocchi circuitali di base della maggior parte delle schede attuali, ossia: microcontrollore, I/O digitale, conversione A/D e comunicazione tramite protocolli UART, USB e I2C. Durante il semestre, gli studenti hanno seguito lezioni teoriche, potendo al contempo sviluppare una propria versione del layout della scheda, partendo da uno schematico comune, mediante il tool di progettazione Kicad 5.

Il secondo semestre è stato invece dedicato alla pratica. Gli studenti frequentanti, in tutto una ventina, sono stati suddivisi in quattro gruppi, ai quali sono stati poi assegnati altrettanti progetti diversi.

Scopo comune ai gruppi è quello di realizzare un dispositivo elettronico a microcontrollore nel suo complesso, ossia, partendo dalle specifiche definite a priori con il supporto del Docente, individuare i blocchi circuitali fondamentali, tradurli in schematico, realizzare il layout del PCB, montare i componenti sullo stampato e infine programmare e testare il dispositivo. Nella presente relazione si vedranno nel dettaglio tutti i passaggi appena citati.

Il corso si appoggia al microcontrollore (da qui: MCU) RP2040, prodotto da Raspberry. Ogni gruppo ha quindi svolto il proprio progetto impiegando tale chip e alcuni componenti in comune con gli altri. Un altro componente reso comune ai vari progetti è un modulo GPS, con il quale sperimentare l’acquisizione di messaggi asincroni (stringhe NMEA) e la cadenza dell’evento 1PPS (pulse per second), utile per operazioni di sincronizzazione temporale tra dispositivi dislocati anche a distanze molto elevate.

Per i dettagli realizzativi, invece, la componentistica e le scelte di progettazione sono state definite ad hoc sulla base delle specifiche decise insieme al Docente.

I quattro gruppi e i relativi progetti sono così definiti:

1. LAN-NTP
2. Sun Tracker
3. MPPT Tracker
4. Sensor Fusion

All’interno di ciascun gruppo, ogni studente ha ricevuto e realizzato uno stampato, in modo da garantire a tutti una buona autonomia nella successiva fase di programmazione e test, che sono comunque stati gestiti nel complesso dal gruppo e non individualmente.

La presente relazione è relativa al gruppo LAN – NTP.

## Il Raspberry RP2040 e l’EVB Raspberry Pi Pico

Per il corso di Progettazione si è convenuto di impiegare un microcontrollore di recente produzione (il lancio sul mercato è avvenuto nel 2021), ossia il Raspberry RP2040. Nel codice del componente sono insiti alcuni dettagli descrittivi interessanti:

* **2** indica un’architettura dual cores
* **0** indica la serie dei cores, in questo caso ARM Cortex M0+
* **4** codifica la quantità di memoria RAM statica, precisamente 264 KBytes
* **0** codifica la quantità di memoria flash. Questo dato è fondamentale in quanto il micro è sprovvisto di memoria flash integrata.

L’ultimo dettaglio, come si vedrà più avanti, obbliga il progettista a prevedere l’installazione di una memoria flash esterna (off-chip memory).

A ciascun gruppo viene fornita, a inizio semestre, una evaluation board denominata Raspberry Pi Pico, che monta il medesimo MCU del progetto, insieme alla componentistica base necessaria al funzionamento (vale a dire: quarzo, flash esterna e interfaccia USB, oltre a due file di fori castellati per l’espansione dell’I/O). Tale scheda di sviluppo è utile per i primi tentativi di programmazione firmware svolti preliminarmente al montaggio dello stampato definitivo, durante le prime settimane del corso.

## 1.3 DESCRIZIONE GENERALE E TOOL IMPIEGATI

### 1.3.1 KiCad 6

La fase di stesura dello schematico e sviluppo del layout del PCB avviene per mezzo dell’ambiente di sviluppo CAD KiCad 6, versione successiva a quella impiegata per la prima parte del Corso.

Rispetto a tale versione, Kicad 6 presenta la correzione di alcuni bug e una rinnovata grafica per quanto riguarda il tool PCBNew. Inoltre, l’ambiente integrato semplifica l’installazione di plugins esterni (ad esempio, per la generazione dei teardrops e per la BOM interattiva.

### 1.3.2 Github

La necessità di lavorare al medesimo progetto per diverse persone ha richiesto l’impiego di un sistema che permettesse il mantenimento di una repository condivisa tra i componenti del gruppo, con cui fosse semplice apportare aggiornamenti e avanzamenti al progetto in locale e successivamente ricaricare il progetto aggiornato. Il nostro gruppo ha quindi creato una repository Github dedicata al progetto, contenente il progetto KiCad, il progetto del firmware, e varie cartelle di supporto con la documentazione e le librerie di componenti KiCad. Mediante il software Github Desktop, le operazioni di clonazione della repository, aggiornamento e caricamento delle modifiche sono state rese estremamente rapide e semplici da fare. Il grande vantaggio di Github è il version control, ossia la possibilità di ritornare su delle operazioni svolte precedentemente all’ultima versione dell’avanzamento. È inoltre possibile, partendo da una repository Master, creare dei progetti Branch che seguono uno sviluppo diverso dall’originale. Per il nostro progetto, si è mantenuta l’integrità delle cartelle e lavorato sempre all’originale.

### 1.3.3 Firmware - Visual Studio Code

L’ambiente di sviluppo firmware impiegato per la programmazione del microcontrollore nell’ambito del progetto LAN-NTP è Visual Studio Code (VSC), come consigliato dalle guide ufficiali Raspberry. Il linguaggio di programmazione è il C, estremamente versatile ed efficiente e molto spesso impiegato in sistemi embedded che richiedono alta velocità di esecuzione.

Per lo sviluppo e la compilazione del codice sorgente è richiesta l’installazione dell’apposita toolchain come da riferimento ufficiale. Il dettaglio dell’installazione è fornito nella sezione relativa al firmware.

# SPECIFICHE

Le caratteristiche fondamentali del prototipo sono state definite insieme al Docente, il quale ha comunque lasciato libertà di aggiungere eventuali migliorie alla scheda. Alcune specifiche sono state definite come linea comune per tutti i gruppi, per semplificare lo sviluppo dei quattro prototipi e l’approvvigionamento dei componenti. Le parti comuni sono le seguenti:

* Memoria flash esterna (Winbond W25Q16JVUSSQ), di tipo NOR, capacità 16Mbit, in package SOIC8
* Quarzo da 12 MHz e package 3225 (3.2 x 2.5 mm). Le capacità di carico vanno dimensionate in base al quarzo scelto (C1C2/(C1+C2) + Cstray)
* Modulo GPS U-blox SAM-M8Q oppure PAM-7Q. I segnali necessari sono la trasmissione UART e il segnale di sincronia 1PPS
* Transistor PMOS IRLML6402 , package SOT-23, per l’accensione (duty-cycling) del modulo GPS
* Interfaccia di programmazione e debug USB
* Regolatore step-down switching TPS563201 per la tensione di alimentazione di 3.3V. Per il progetto LAN-NTP si è impiegato invece il LMR16006, avente tensione massima di ingresso di 60V. Questo dato è fondamentale per la soluzione PoE (Power over Ethernet) descritta più avanti.
* Package dei componenti passivi 0603 per resistori dissipanti fino a 0.25W, condensatori fino a 10uF e induttori di filtraggio. Per dissipazioni maggiori, o capacità di 10uF o maggiori, il package da usare è lo 0805 (o 1206 o 1210 per resistori altamente dissipativi).
* Package SMA per diodi di protezione e relativi ai regolatori switching

Una volta definiti i componenti su descritti, si è passati a delineare per ciascun gruppo il progetto nel suo specifico. L’obiettivo finale del nostro dispositivo è una versione semplificata di un server NTP (Network Time Protocol), ossia un apparato connesso in rete che fornisce ai client che lo richiedono l’ora esatta del sistema. Per il progetto ivi descritto, la richiesta principale è la valorizzazione del drift di un RTC (Real Time Clock) installato sullo stampato, in un certo arco temporale. Per tale misurazione, si fa uso di un GPS, che, proprio per il suo principio di funzionamento, garantisce errori nella scansione del tempo dell’ordine del nanosecondo. Il valore misurato deve essere reso disponibile in rete locale, per cui è fondamentale la presenza dell’interfaccia fisica Ethernet.

In generale, il dispositivo LAN-NTP si compone quindi dei seguenti macro-blocchi:

* Microcontrollore
* Circuito di interfaccia LAN
* Alimentazione USB, PoE ed esterna mediante morsetto a vite
* Circuito integrato RTC
* Modulo GPS

## Il microcontrollore

Come anticipato, il microcontrollore di interesse per il progetto è l’RP2040. La scelta è stata fatta per permettere l’approfondimento di una piattaforma moderna, basata su architettura a 32 bit. Si presenta in un unico package QFN-56, con pad di massa centrale al di sotto del corpo del componente. Di seguito si elencano le principali necessità da considerare nello sviluppo dello stampato per quanto riguarda il microcontrollore.

La prima particolarità rilevante è l’assenza di una memoria flash (la memoria non volatile e riscrivibile che contiene le istruzioni in linguaggio macchina) integrata al microcontrollore. Questa particolarità ha, da un lato, l’inconveniente di obbligare il progettista a impiegare un componente esterno, con aumento di costo e ingombro sullo stampato. Dall’altro lato, permette flessibilità nella scelta della quantità di memoria che firmware complessi possano richiedere. Per le operazioni di reset e di caricamento del firmware, sono richiesti due pulsanti (normalmente aperti) connessi al microcontrollore.

La necessità di impiegare la periferica USB per la comunicazione, il caricamento di firmware e il debug, richiede la presenza di un quarzo risonante alla frequenza di 12.000 MHz. Sempre nell’ambito della comunicazione USB, si predisporrà il montaggio di un connettore USB-B o, alternativamente, USB-Micro.

Un led è richiesto per facilitare il debug, e per segnalare il corretto funzionamento del dispositivo durante la sua attività (led “alive”).

## Il circuito LAN

La richiesta di avere il dispositivo connesso in rete rende necessario un circuito di interfaccia tra il microcontrollore e l’access point. In particolare, si è scelta la connettività Ethernet e il transceiver LAN8742. Gli studenti hanno richiesto di implementare un circuito di alimentazione PoE (Power over Ethernet) cercando di massimizzare la qualità del prototipo, come descritto nella sezione relativa.

## Il circuito di alimentazione

Tutti i componenti presenti lavorano ad una tensione di alimentazione di 3,3V. È quindi sufficiente un singolo circuito regolatore di step down. È richiesto che la scheda sia alimentata alternativamente attraverso il cavo USB di programmazione, oppure mediante alimentatore esterno connesso al morsetto a vite apposito. Vista la presenza del connettore RJ45, gli studenti hanno richiesto di implementare un circuito di alimentazione che impieghi tale connettore, sfruttando il protocollo PoE (Power over Ethernet). Il dettaglio di questo circuito è presente nella sezione di descrizione dei blocchi circuitali. Si fa presente che per tale circuito è stato preso esempio da schematici forniti dal Docente.

La coesistenza delle tre modalità di alimentazione, tutte in ingresso allo stesso regolatore, richiede che esse siano connesse al regolatore mediante un circuito di “OR” logico a diodi.

## Il modulo RTC

Su suggerimento del Docente, si è scelto di impiegare il modulo RTC PCF8563. Il riferimento di frequenza è fornito da un quarzo da 32.768 KHz, in package cilindrico 10x3mm. La comunicazione con il MCU host avviene mediante protocollo I2C, e l’alimentazione del modulo a dispositivo spento è garantita da un super condensatore da 0.1 F, come da suggerimento del datasheet del componente.

## Il modulo GPS

I componenti suggeriti e di cui si predisporrà il montaggio sono gli U-blox SAM-M8Q oppure PAM-7Q.

Entrambi sono del tipo AOT (antenna on top) e non richiedono quindi un’antenna esterna e particolari attenzioni a tracce a impedenza controllata per la radiofrequenza. Gli unici segnali da connettere all’RP2040 sono la trasmissione UART e il PPS.

# PROGETTAZIONE CAD SCHEMATIC

Nella prossima sezione verranno descritte nel dettaglio tutte le sezioni circuitali, partendo dallo schematico sviluppato. Il tool impiegato è Eeschema della suite Kicad 6. Si ricorda che i passi da seguire nello sviluppo dello schematico per arrivare al layout dello stampato sono i seguenti:

1. Piazzamento dei simboli dei componenti (shortcut: ‘A’) nel foglio schematico. Nel caso di componenti che avranno una sola istanza sullo stampato, ma possibilità di essere impiegati in diversi package, è necessario prevedere un simbolo per ogni footprint, data la corrispondenza biunivoca tra simbolo e footprint.
2. Connessione pin a pin mediante fili (shortcut: ‘W’).
3. Annotazione dei simboli, ossia l’assegnazione di una label univoca per ogni componente nello schematico.
4. Assegnamento footprint: ad ogni simbolo si fa corrispondere il relativo footprint che verrà in seguito posizionato nel layout manager.
5. Generazione netlist. La netlist è un file testuale che definisce tutte le caratteristiche dei componenti, e le loro interconnessioni. Sarà il file da importare nel tool di sviluppo layout per il suo sviluppo.
6. Generazione bill of materials necessaria per l’approvvigionamento dei componenti.

In breve, si vedrà lo sviluppo del circuito di alimentazione, dei componenti di accompagnamento del microcontrollore (del quale si dettaglieranno anche alcune caratteristiche rilevanti).

## Alimentazione

È già stato accennato nella sezione “Specifiche” al circuito di alimentazione. Viene richiesta la possibilità di alimentare il dispositivo in oggetto, sia mediante cavo USB, sia tramite un connettore separato. In entrambi i casi, la tensione di alimentazione è superiore ai 3.3 V richiesti dai componenti scelti. È perciò indispensabile un circuito di step down di tipo switching per tale regolazione. In aggiunta alle due modalità accennate, si sceglie di poter alimentare il dispositivo anche tramite PoE, sfruttando il cavo di rete e il connettore RJ45. Esso fornisce potenza mediante le linee non utilizzate dai dati (pin 4-5, 7-8 modo B, o con la dc sovrapposta al segnale….). La tensione nominale tra il polo positivo e il negativo è definita dal protocollo PoE pari a 48V. è quindi richiesto un regolatore switching che sostenga tale tensione. Il regolatore scelto è quindi l’LMR16006, che sopporta tensioni di ingresso comprese tra 4 e 60V. Il calcolo dell’induttanza avviene seguendo le linee guida del produttore, e il valore commerciale scelto è di 22 uH. Il diodo Schottky di ricircolo è un …. In package SMA.

Il riferimento interno al regolatore è pari a 0.765V, e per ottenere una tensione regolata di 3V3, il partitore di feedback è allora così calcolato:

I valori di R1 e R2 sono rispettivamente di 10 kOhm e 33 kOhm.

Per le capacità di ingresso e di uscita si segue la linea guida del produttore, e i valori sono rispettivamente di 2,2 uF per la capacità di ingresso e 10 uF per quella di uscita. Entrambi i condensatori hanno in parallelo altrettante capacità da 100 nF. La capacità di bootstrap, connessa tra i pin SW e CB del regolatore, è fissata a 100 nF.

Le tre tipologie di alimentazione esterna sono connesse alla linea di ingresso del medesimo regolatore. È allora necessario realizzare un circuito detto “OR a diodi” per impedire che l’eventuale utilizzo di due o tutti e tre i modi di alimentazione (USB, alimentatore esterno e PoE) possa danneggiare la scheda o uno dei circuiti di alimentazione connessi. Il circuito è schematizzato come segue:

Il diodo relativo alla Vbus del connettore USB è di tipo Schottky, dato che la sua minore tensione diretta di attivazione rispetto a un comune raddrizzatore, assicura che in ingresso al regolatore ci sia sempre la tensione minima di funzionamento richiesta di 4V.

Si nota che l’alimentazione mediante il morsetto a vite prevede delle protezioni aggiuntive: sono infatti inseriti, un polyfuse, con corrente nominale di apertura di 500mA, in package 1812, e un TVS, che sopprime eventuali impulsi nella tensione di ingresso. Il modello suggerito dal Docente, SMAJ56, suggerisce che la tensione alla quale la protezione interviene è di 56 volt, e che il package del componente è un SMA.

L’alimentazione Power over Ethernet è leggermente più complessa e richiede della componentistica aggiuntiva. Tale circuiteria è stata prevista, con la possibilità di alimentare esternamente il dispositivo, bypassandola completamente, nel caso il Poe causasse rallentamenti allo sviluppo del progetto. Il protocollo richiede infatti che un dispositivo alimentato risponda a una fase di riconoscimento. Un dispositivo alimentato tramite PoE prevede la presenza di un “iniettore”, che fornisce tramite cavo Ethernet sia la connessione allo switch di rete, che la potenza elettrica di alimentazione, con una tensione continua nominale di 48V. Alla connessione del Powered Device con l’iniettore, ha inizio una fase di riconoscimento della classe di device PoE. Essa prevede che il dispositivo assorba una quantità di corrente con andamento a rampa, in modo che l’iniettore che lo alimenta riconosca la classe di dispositivo connesso, sulla base dell’assorbimento iniziale e del tempo di salita della rampa. Questo scambio di informazioni in forma analogica è gestito (lato dispositivo alimentato) da un apposito circuito integrato, il TPS2375 nel caso in esame. La classe a cui il nostro dispositivo appartiene è la classe 3, e per definirla, tra il pin CLASS dell’integrato e GND deve essere connessa una resistenza del valore di357 ohm.

Si precisa che per il circuito PoE, il Docente ha fornito degli schematici di appoggio.

## Il microcontrollore RP2040

La componentistica relativa al funzionamento del microcontrollore è già stata in parte introdotta nella sezione “specifiche”. In aggiunta al quarzo da 12.000 MHz in package 3225 (la cui piedinatura vede i terminali del cristallo disponibili sui pin 1 e 3, mentre 2 e 4 vanno connessi a GND), sono necessarie varie capacità di disaccoppiamento connesse ai pin di alimentazione. Per tutte queste capacità si sceglie il valore commerciale di 100nF, come nella maggior parte dei casi in cui si necessita di disaccoppiamento. Oltre ai pin di alimentazione a 3.3V, si connettono altrettante capacità ai pin sui quali è presente una tensione generata dal microcontrollore. Ad esempio, sui pin 23, 45 e 50 dell’MCU si trova la tensione di 1.1V generata internamente, e necessaria all’alimentazione dei cores.

Un componente fondamentale per le operazioni di debug firmware e per segnalare la corretta attività di ogni dispositivo a microcontrollore è un comune led. Ne viene inserito uno la cui resistenza in serie è fissata a 330 ohm. Inoltre, è consigliato avere un tasto di reset del sistema. Questo viene connesso tra il pin nRST del micro e GND, con un resistore di pullup da 1 kOhm connesso tra il pin di reset e VDD.

Di seguito si descrivono brevemente alcune parti interessanti del microcontrollore, e per alcune di esse si spiegheranno le soluzioni circuitali adottate.

### Il clock

Il funzionamento di ogni microcontrollore è basato su circuiti logici e sequenziali, che necessitano di un segnale di sincronismo ad alta precisione. La frequenza di tale clock determina in maniera sostanziale la velocità di esecuzione del codice, e di conseguenza la risposta a condizioni ed eventi che accadono durante il funzionamento. Questo è uno degli aspetti critici di un sistema embedded, detti anche real time, come quello in oggetto.

Il core, la memoria, i timer e le varie periferiche impiegano un determinato segnale di clock, derivato da diverse sorgenti e regolato attraverso un divisore di frequenza detto prescaler, che a seconda dell’impostazione di determinati registri di memoria effettua la divisione della frequenza in ingresso e la rende disponibile al blocco che richiede il segnale di clock.

Tra le varie possibilità per generare i segnali di clock, l’RP2040 dispone delle seguenti:

* Clock esterno: fornito da un oscillatore dedicato, ad esempio un TCXO (oscillatore compensato in temperatura).
* Clock esterno fornito da un oscillatore a rilassamento: si ottiene inserendo un circuito RC serie a due pin appositi del microcontrollore.
* Cristallo di quarzo: risonante a frequenze comprese tra 1 e 15 MHz, va connesso ai pin XIN e XOUT del micro, in parallelo a due capacità di carico. La frequenza generata dall’oscillatore che sfrutta il quarzo è posta in ingresso a due PLL (phase-locked loop) che fungono da moltiplicatori di frequenza. Il primo genera una frequenza impostabile fino a 133 MHz, e viene utilizzato dai cores. Il secondo genera la frequenza di 48 MHz richiesta dalla periferica USB. È importante, per tale scopo, che il quarzo sia risonante a 12 MHz.

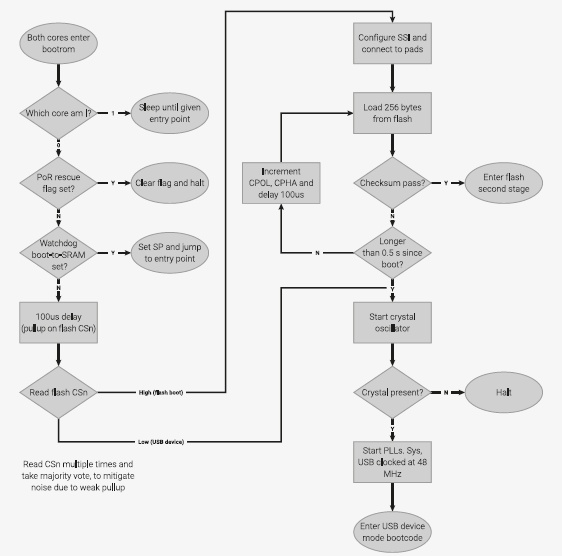
Nel nostro progetto, la sorgente del clock impiegata è costituita da un cristallo risonante a 12.000 MHz per il motivo appena citato.

* Ring oscillator: il chip contiene un ring oscillator (composto quindi da un numero dispari di NOT logici connessi ad anello), attivo dall’accensione dell’unità. Può essere impiegato come sorgente di clock, sebbene poco accurata, fornendo una frequenza variabile, a seconda di temperatura, tensione di alimentazione e processo in corso, tra 4 e 8 MHz (fonte: datasheet).

### La memoria FLASH

La comunicazione con il microcontrollore avviene tramite interfaccia QSPI (quad- SPI) ed è analoga alla Serial Peripheral Interface standard, ma lo scambio di dati avviene su quattro linee dati (sono assenti le linee MISO e MOSI). Questo permette di inviare quattro bit simultaneamente per ogni ciclo di clock. Esiste anche la modalità a doppia velocità, con quattro bit trasmessi ad ogni fronte di clock, per un totale di otto bit per ciclo.

La scrittura su flash avviene forzando a livello logico basso il segnale CS (chip select). Durante l’esecuzione del codice, il microcontrollore si occupa della gestione di tale segnale, per le operazioni di lettura e scrittura. Durante la fase di scrittura (flashing) del firmware, invece, è l’utente (o un programmatore dedicato) che mantiene basso tale segnale. Nella evaluation board Pico, e nel progetto in esame, tale operazione avviene mediante la pressione di un pulsante (BOOTSEL) al momento della connessione al PC. Questo sistema fa avviare la sequenza di bootloading, schematizzata in fig. X (fonte: RP2040 datasheet). In tal modo, alla connessione con il PC, il microcontrollore figura come una memoria di massa, nella quale è possibile caricare il file eseguibile (in formato \*.uf2).



### Le periferiche di interesse

Il microcontrollore RP2040 dispone di molte periferiche. Di seguito si analizzeranno quelle più interessanti dal punto di vista del progetto.

#### GPIO

L’RP2040 dispone di 36 pin GPIO, includendo i sei dedicati all’interfaccia QSPI, dedicata all’interfaccia con la memoria flash esterna. Normalmente, si considerano i pin da GPIO0 a GPIO29. Di questi, i pin da GPIO26 a GPIO29 sono impiegabili come ingressi analogici. Inoltre, tra le varie funzionalità che i pin GPIO possono assumere, si nota la possibilità di impiegarli come interfacce di comunicazione seriale, quali UART, SPI, I2C, USB, output in modulazione PWM, input/output di segnali di clock, e per la connessione con le macchine a stati finiti che costituiscono il PIO (Programmable I/O).

#### Timer

L’unità dispone di un timer basato su contatore a 64 bit (divisi in due registri da 32 bit), e incremento alla frequenza di 1 MHz. Può essere impiegato nella programmazione del firmware come free timer di periodo 1us, e conta liberamente in incremento il tempo di attività del microcontrollore, dalla sua accensione. L’overflow del registro a 64 bit avviene dopo circa 585 mila anni; perciò, sono evitati problemi del tipo “millennium bug”.

È importante notare che la lettura dei due registri avviene in due cicli di clock, necessariamente con un accesso al registro meno significativo (“L”) e poi al più significativo (“H”). In questo modo, avviene il latching del registro H all’istante della lettura del registro L. Questo previene errori di lettura, nel caso la lettura avvenga a cavallo dell’overflow del registro L e dell’incremento del registro H. È comunque possibile accedere ai due registri “RAW” contenenti il valore non latchato del contatore.

Altri timer che possono essere sfruttati sono quelli relativi alla periferica PWM (fino a otto timer diversi), o quello relativo al watchdog. Questa periferica impiega un timer dedicato che deve essere ripristinato via software entro la sua scadenza. Ha la finalità di generare interrupt per risvegliare il microcontrollore da modalità a basso consumo energetico (sleep), o di causare un reset di sistema in caso di loop bloccanti.

#### USB

L’RP2040 dispone di un controller USB capace di operare come device in Full Speed a 12 Mbps, o come host connettendo dispositivi LS o FS. Supporta la versione USB 2.0, fino a 16 endpoints in input e 16 in output, le modalità di trasferimento Control, Isochronous, Bulk e Interrupt. La periferica è fondamentale nel caricamento del file eseguibile in memoria flash (la memoria viene vista come unità di massa in cui trasferire questo file) e nella comunicazione di debug con il PC (lo standard output è indirizzabile sull’interfaccia USB, vista dal PC come COM virtuale).

#### UART

Nel progetto in esame, l’interfaccia UART è di particolare interesse per quanto riguarda la ricezione, in quanto fondamentale per l’input di messaggi asincroni derivanti dal modulo GPS.

Il microcontrollore permette di istanziare due interfacce UART separate, e mappabili mediante firmware su gruppi diversi di pin GPIO. Oltre ai segnali Tx e Rx, possono essere gestiti attraverso altri due pin anche i segnali di controllo CTS e RTS, non utilizzati per il nostro obiettivo.

Per il nostro progetto, i pin dedicati all’interfaccia seriale (uart0, nella mappatura standard) sono i pin GPIO0 (Rx) e GPIO1 (Tx). Essi vengono connessi ai rispettivi pin Tx ed Rx del GPS.

È possibile, previa impostazione firmware, indirizzare l’output dello standard I/O alternativamente attraverso l’interfaccia USB (attraverso l’apertura di una porta COM virtuale, classe CDC), oppure attraverso la seriale asincrona UART.

#### I2C

L’RP2040 dispone di due interfacce I2C e può operare come Master o come Slave. Supporta tutte le velocità previste dal protocollo fino alla Fast Mode plus (fino a 1Mbps). I pin relativi ai segnali SCL e SDA possono essere configurati in modo da avere una resistenza di pullup connessa internamente a Vcc. Il datasheet consiglia comunque di inserire dei resistori esterni per tale scopo.

Per i dettagli del protocollo, si rimanda alla precedente relazione.

Si vedrà nella sottosezione specifica la connessione del modulo RTC con il microcontrollore.

#### PIO

Una periferica estremamente interessante presente nell’RP2040 è la Programmable I/O. Un blocco PIO permette di associare a uno o più pin di I/O fino a quattro macchine a stati finiti in grado di pilotare dei segnali in maniera hardware, ossia senza coinvolgere il programma principale. Il vantaggio è quello di non impiegare la CPU per il pilotaggio dei pin, incrementando la velocità di esecuzione del codice.

Un blocco PIO si compone di due registri a scorrimento, in comunicazione con altrettanti buffer (code), una per i segnali in uscita e uno per i segnali di input. Usa inoltre due registri di appoggio, un divisore di frequenza di clock frazionario, un registro (Program Counter) che punta all’indirizzo dell’istruzione in corso di svolgimento, e una logica di controllo che gestisce il fetching delle istruzioni dalla memoria istruzioni dedicata al PIO.

Ogni MSF si programma mediante nove istruzioni assembly, che usate correttamente permettono la generazione di segnali e quindi l’implementazione di svariati protocolli di comunicazione. Permettono operazioni di lettura/scrittura dei valori logici assunti dai segnali posti in input/output tramite i pin.

Questa periferica è di particolare interesse in quanto impiegata per la generazione dei segnali RMII per il pilotaggio del transceiver Ethernet.

## Il circuito LAN

Di seguito si descrive brevemente la circuiteria che si occupa della trasformazione del livello fisico (PHY) dell’interfaccia Ethernet in segnali digitali per la comunicazione tra il microcontrollore e il transceiver Ethernet. La comunicazione tra questi due componenti rispetta lo standard RMII.

Escludendo la circuiteria PoE, il blocco LAN richiede i seguenti componenti, nell’ordine di connessione tra il connettore e il MCU:

* Connettore RJ45, 8P8C, con led di indicazione dello status Active e Link
* Magnetics per standard 10/100BASE-T Wurth Elektronik WE749013011
* Transceiver LAN8742 o LAN8720

Lo standard 10/100BASE-T richiede l’impiego di due sole coppie di fili intrecciati, connessi ai poli n. 1-2 (prima coppia) e 3-6 (seconda coppia). Detti contatti portano, in questo ordine, i segnali denominati TX\_P, TX\_N, RX\_P, RX\_N. I due poli di ciascuna coppia sono connessi ai due poli di un choke di modo comune. I due poli opposti di tale induttore sono collegati (internamente al componente) ai capi dell’avvolgimento primario di un trasformatore a presa centrale di rapporto 1:1. Il secondario del trasformatore costituisce la connessione con il transceiver. Di seguito uno schematico che spiega quanto descritto, per quanto riguarda una sola coppia di segnali.

Diagram

Description automatically generated

Nello schematico seguente, si vede nel dettaglio la connessione tra i magnetics (componente composto da induttore e trasformatori come descritto sopra), e il transceiver.

Diagram, schematic

Description automatically generated

I resistori di valore 49.9 ohm, connessi alla tensione di alimentazione positiva del transceiver, sono necessari per l’adattamento di impedenza della linea che connette i magnetics con il transceiver stesso, e allo stesso tempo per la traslazione del valore medio del segnale alla tensione di alimentazione positiva.

L’alimentazione del LAN8742/8720 è derivata da quella generale del dispositivo attraverso un filtro LC (composto dall’induttore L2 nello schema, e dal parallelo delle varie capacità presenti per scopi di disaccoppiamento e impedance matching).

L’integrato USBLC6-4SC6 è composto da una rete di sette diodi di protezione. Non è quindi strettamente necessario ai fini del funzionamento di base del circuito LAN, ma protegge l’ingresso analogico del transceiver da picchi di tensione.

La comunicazione tra il microcontrollore host e il transceiver LAN avviene secondo lo standard RMII (Reduced Media-Independent Interface), che si occupa dell’incapsulamento del livello MAC in segnali del livello fisico secondo lo stack TCP/IP. Prevede la presenza dei seguenti segnali:

* REF\_CLK
* TXD0, TXD1
* RXD0, RXD1
* TX\_EN
* CRS\_DV
* MDIO, MDC

Il segnale di sincronismo REF\_CLK è derivato dall’oscillatore a 25MHz del transceiver LAN8742. I segnali TXD0 e TXD1 trasportano i dati da trasmettere, immettendo sulla linea due bit ogni ciclo di clock. Analoga modalità è usata per la ricezione. TX\_EN e CRS\_DV si occupano del controllo della trasmissione (dal dispositivo verso la rete) e del rilevamento dello stato della linea (in caso di altri dispositivi in comunicazione). MDIO e MDC sono segnali di controllo e clock della comunicazione RMII.

## Il modulo GPS

Per quanto riguarda il modulo GPS da impiegare, vengono proposte due alternative, ossia il SAM- M8Q e il PAM-7Q.

Nello schematico, si inseriscono allora i simboli relativi a entrambi i componenti.

La tensione di alimentazione positiva dei GPS è fornita tramite un pMOS (IRLML6402) i cui source e drain sono connessi rispettivamente alla linea 3V3 e al pin VCC dei due moduli GPS. Tra source e gate viene connessa una resistenza di spegnimento del valore di 47kOhm. Il gate viene poi connesso a un pin di controllo (ENABLE\_GPS) del microcontrollore, in configurazione open drain. Questo circuito permette l’accensione e lo spegnimento del GPS mediante un unico segnale di controllo. Infatti, con il segnale ENABLE\_GPS a livello alto, il pMOS è interdetto e il GPS spento; ponendo a livello basso il segnale di controllo, la tensione S-G positiva fa entrare il transistor in conduzione, accendendo il GPS. Si noti che la RDS-ON del transistor è di pochi milliohm. Si può allora considerare nulla la caduta VSD.

* Il modulo SAM-M8Q presenta 20 pad, di cui nove sono connesse a GND, VCC, VCC\_IO e VBCKP sono connesse alla tensione di alimentazione positiva, EXT\_INT, nRESET e nSAFEBOOT sono segnali che non verranno impiegati, e pertanto rimarranno non connessi. Il modulo dispone di un’interfaccia di comunicazione UART standard, come per tutti i ricevitori GPS, i cui segnali Rx e Tx vanno connessi al microcontrollore, e di un’interfaccia I2C, che non viene impiegata e rimane non connessa. Il segnale TIMEPULSE, o 1PPS, è un segnale a onda rettangolare il cui fronte di salita si verifica ad intervalli di un secondo ed elevata accuratezza. Tale segnale viene impiegato per la sincronizzazione del sistema e viene allora connesso al pin GPIO2 del microcontrollore.
* Il PAM-7Q richiede che VCC e VBCKP siano connessi all’alimentazione positiva, GND alla massa del circuito, Tx ed Rx ai corrispondenti Rx e Tx del micro (come per il precedente SAM-M8Q), 1PPS al pin di lettura del segnale di sincronismo. Anche per questo componente, l’interfaccia I2C non viene usata e i pin sono pertanto NC.

Nella progettazione dello schematico, si prevedono un simbolo relativo al SAM-M8Q e uno relativo al PAM-7Q, per permettere l’inserimento dei footprint rispettivi nel layout PCB. Tutti i segnali richiesti vengono connessi tra il microcontrollore ed entrambi i moduli.

## Il modulo RTC

La comunicazione con il modulo RTC MCP7940 richiede l’impiego di una interfaccia I2C. Per tale comunicazione è stata sfruttata l’istanza firmware i2c0, che impiega di default i pin GP2 (SDA) e GP3 (SCL). È comunque possibile assegnare arbitrariamente i pin dell’unità alle funzioni di comunicazione. Sempre mediante firmware, si può abilitare una resistenza di pullup interna. Per il protocollo, una resistenza di pullup è sempre necessaria, in quanto tutti i circuiti integrati in comunicazione hanno le uscite di tipo OD (open drain), ossia presentano la sola rete di pull down. Ogni integrato è allora in grado di forzare il segnale a livello basso. Rilasciando il controllo della linea, il segnale si porta automaticamente a livello alto. Nello schematico relativo alla comunicazione con il modulo RTC, vengono inserite due resistenze fisiche di pullup del valore di 10 kOhm.

Come tutti i circuiti integrati nel circuito, il MCP7940 ha una tensione di alimentazione di 3.3V, fornita attraverso i pin VDD e VSS. Il pin VDD è connesso all’alimentazione positiva attraverso un diodo Schottky. In parallelo al modulo è connessa una rete RC serie composta da un supercap da 0,1F e una resistenza da 100 ohm. Questo circuito, in cui il condensatore viene caricato attraverso il diodo e la resistenza, si occupa di mantenere attivo il modulo RTC a dispositivo spento.

Per il suo regolare funzionamento, il modulo richiede la connessione dei pin OSCI e OSCO ai terminali di un cristallo di quarzo risonante a 32.768 kHz, per la generazione del clock. Tale valore è utilizzato per operazioni di temporizzazione a cadenza pari a un secondo, poiché un divisore di frequenza a 15 bit fornisce un segnale a onda rettangolare di periodo 1 s, partendo da una base di 32.768 kHz.

Il pin denominato VBAT, impiegabile per alimentare il componente mediante una pila di backup, non viene usato. Dato che inizialmente era previsto di utilizzare un PCF8563, la connessione è comunque presente nello schema e nello stampato. Sarà necessario mantenere il pin del MCU connesso in stato di alta impedenza.

Il pin MFP rappresenta un’uscita di tipo open drain. Tale segnale può indicare l’avvenimento di alcuni eventi preimpostabili attraverso l’interfaccia di comunicazione. Questi eventi possono essere la scadenza di un timer, oppure una “sveglia”, ossia il raggiungimento di un orario predeterminato. Nel nostro schematico tale resistore non è stato previsto per un errore di progettazione. Sarà quindi necessario abilitare la resistenza di pullup connessa al pin del microcontrollore per utilizzare i segnali.

La fig. XX riassume quanto esposto finora. Lo schematico è relativo al PCF8563, ma escludendo il pin 3, la connessione è valida anche per l’MCP7490.

Diagram, schematic

Description automatically generated

# Progettazione CAD PCB layout

Una volta completato lo schematico, e definito il package di tutti i componenti che andranno montati sullo stampato, si passa alla progettazione del layout della scheda. La scheda in oggetto sarà composta di due strati (layer, Top e Bottom), e avrà spessore del substrato (FR4) pari a 1.6mm.

Per prima cosa si definiscono le regole di progettazione. Nel nostro caso si rispettano le seguenti linee guida:

veww.

Il passo successivo è quello di impostare il passo della griglia di sbroglio. Nel nostro caso, durante la fase di posizionamento componenti e sbroglio, sono stati impiegati i valori di 0.1mm e 0.05mm.

In seguito si passa alla definizione del bordo scheda. Si replica il design dello stampato del Raspberry Pi4. L’operazione è stata eseguita importando il file \*.dwg riportante la vista in pianta della scheda in questione, tracciando il bordo scheda (layer “Edge-Cuts”) rispettando l’originale, e posizionando i fori di fissaggio concordemente.

Il passo successivo è l’importazione della netlist generata precedentemente, al termine dello sviluppo dello schematico. Tale operazione importa automaticamente i footprint dei componenti, e lega visivamente le varie piazzole secondo le connessioni dello schematico mediante la “rastnet”, ossia delle linee grafiche che indicano le connessioni delle piazzole stesse.

I footprint vengono successivamente posizionati all’interno del bordo scheda seguendo questo ordine generale:

* Microcontrollore, memoria e transceiver LAN posti in posizione centrale;
* Connettori RJ45, USB e morsetto sul medesimo lato corto;
* Circuiteria LAN (magnetics, linee di segnale) posta tra il connettore RJ15 e il transceiver;
* GPS e RTC posti sul lato corto opposto. Il modulo GPS è stato posizionato ad alcuni millimetri di distanza dal bordo scheda, per garantire la presenza di una sufficiente superficie di massa necessaria al corretto funzionamento dell’antenna integrata;
* Circuiteria PoE e regolatore switching posti sul lato lungo inferiore;
* Pulsanti di Boot e Reset posti sul lato lungo superiore;
* OR a diodi e circuito di protezione (TVS, polyfuse) posti tra le tre sorgenti di alimentazione, nelle vicinanze dei connettori;
* Componenti a corredo (condensatori di disaccoppiamento, protezioni, resistori di pullup/pulldown, diodo led, diodo di alimentazione RTC, pMOS di spegnimento del GPS) posti di conseguenza relativamente alla loro funzione.

Si fa presente che, nonostante l’attenzione posta nel posizionamento di alcuni componenti, alcuni condensatori di disaccoppiamento e il diodo di ricircolo del regolatore switching non sono stati posti nella maniera ottimale. Essi vanno infatti mantenuti il più possibile vicini tra loro.

Graphical user interface

Description automatically generated with medium confidence

La maggior parte dei componenti sono a montaggio superficiale; alcuni componenti (connettori, tasti, zoccolo di supporto del GPS) sono invece a montaggio su foro per garantire maggiore tenuta agli sforzi meccanici. Di seguito si riportano le linee guida per quanto riguarda la scelta dei package dei vari componenti. Il dettaglio è comunque fornito nella BOM allegata.

Al termine del posizionamento dei componenti, si generano due zone piene, una su ciascun layer dello stampato, connesse alla net GND. Esse hanno la funzione di semplificare lo sbroglio, in quanto la maggior parte dei componenti ha almeno una connessione a massa; inoltre migliorano la qualità della scheda, rendendola più robusta ai disturbi elettromagnetici, e favorendo la dissipazione del calore generato attraverso la maggiore superficie metallica.

Quando le zone piene sono connesse alle relative piazzole (sia SMD che PTH), il loro contatto fisico con la piazzola avviene per mezzo dei thermal reliefs, piccoli segmenti di rame disposti a croce. Essi vengono ricavati mediante la fresatura del layer di rame come avviene per le tracce e le piazzole, e hanno la funzione di facilitare la saldatura manuale, impedendo la dispersione del calore attraverso tutta la zona piena, quando la punta dello stilo saldante viene posta sulla pad e lo stagno viene fuso per la brasatura.

È possibile posizionare delle zone (poligoni) cosiddette “no-fill”, dove non siano presenti footprint, tracce, via o zone piene. Nel nostro caso, una zona no-fill viene posta in corrispondenza del connettore RJ45 e dei magnetics, in modo da evitare le sole zone piene di massa. Un’altra zona no-fill è posta tra le piazzole non usate del GPS SAM-M8Q, in modo da evitare la creazione di tali piazzole. Questa scelta assicura che piazzole presenti, ma non connesse, possano generare problemi quali falsi contatti, dovuti ad esempio a colatura di stagno fuso che causino cortocircuiti tra i pin del componente.

Scrivere dei commenti fuori scheda riguardo gli errori drc

## 4.1 Piani gerber

La visualizzazione separata dei piani gerber, mediante GerbView, permette di verificare che tutti gli elementi del layout siano corretti ai fini della produzione. Le immagini dei piani sono riportate nella sezione “Allegati”.

Attraverso le impostazioni dell’editor PCB, è stato generato un nuovo piano Top-Solder Paste, ridotto del 2% rispetto al piano creato per la produzione dello stampato. Questo nuovo piano viene impiegato per la generazione della lamina usata per la stesura della pasta saldante da applicare alle piazzole dei componenti SMD.

## 4.2 Alloggiamento

La scelta di un contenitore destinato allo stampato è ricaduta su un generico contenitore per il single-board computer Raspberry Pi 4. Si è allora imposto che le dimensioni e il posizionamento dei fori della nostra scheda fossero identici a tale SBC. Inoltre, i tre connettori, ovvero RJ45, USB e morsetto a vite, devono rispettare il posizionamento dei connettori presenti sul RPi4.

# Assemblaggio manuale in laboratorio e test funzionali

L’assemblaggio dei componenti sul circuito stampato è stato svolto in laboratorio con il supporto del Docente. La presenza di componentistica in package aventi i pin nascosti sotto il corpo ha obbligato a ricorrere al montaggio mediante lamina, pasta saldante e piastra riscaldata.

La procedura prevede solitamente di posizionare la scheda da montare sulla base di un telaio per l’allineamento della lamina. Nel nostro caso, sono stati usati degli altri stampati, fissati al banco di montaggio, per l’allineamento. È importante che la lamina sia perfettamente allineata con lo stampato sottostante, pena l’errata stesura della pasta sulle pad di saldatura SMD. Anche lo spessore degli stampati usati per l’allineamento deve coincidere con quello della scheda da montare, o si corre il rischio di avere una stesura e una conseguente saldatura di scarsa qualità.

Una volta allineato lamina e stampato, si fissa la prima in modo sicuro, e si procede con la stesura della pasta di stagno. Questa sostanza contiene una polvere di stagno amalgamata con il flussante, ed ha una consistenza densa ma facilmente stendibile. Con l’impiego di una spatola o di una lama, si procede a stendere la pasta lungo tutta la lamina, i cui fori in corrispondenza delle pad di saldatura lasciano depositare il giusto quantitativo di pasta. È fondamentale che durante la stesura non si verifichi il disallineamento della lamina.

Una volta stesa correttamente la pasta, si può rimuovere la lamina e passare al montaggio dei componenti. Mediante una pinzetta SMD, si posizionano attentamente i componenti allineando i pin alle relative pad.

Terminata questa fase, la scheda viene posta in forno o su una piastra metallica riscaldata alla temperatura di fusione della pasta di stagno (circa 280°C, come nel nostro caso).

Può verificarsi che un componente non sia esattamente allineato alla sua impronta. Nella maggior parte dei casi questo inconveniente si risolve grazie alla tensione superficiale dello stagno liquefatto, che attua una forza sui pin del componente che lo allinea alle pad, prima di solidificarsi. In altri casi avviene che il package sia allineato a pin adiacenti a quello corretto, causando l’errato montaggio del componente stesso. È allora richiesta una rilavorazione manuale, con la dissaldatura del componente mediante pistola ad aria calda o, se disponibile, a infrarosso.

Anche eventuali saldature di bassa qualità (fredde) o cortocircuiti fra sue o più pin adiacenti, richiedono rilavorazioni successive. Per asportare dello stagno in eccedenza, si ricorre talvolta a una trecciola in rame, che riscaldata (ad aria, o con stilo saldante) rimuove il metallo in eccesso per legame metallico e capillarità.

Altro fenomeno che si verifica talvolta è il cosiddetto tombstoning: nel caso di componenti a due piazzole opposte, come nel caso di passivi, diodi o fusibili, se uno dei due terminali non è ben adagiato alla pasta di stagno, la tensione del metallo liquido applicata al terminale opposto fa sì che il componente si posizioni in verticale (da qui il termine tombstoning) e si saldi al raffreddarsi della pasta.

Componenti through hole, quali il quarzo dell’RTC, e i connettori (RJ45, USB-B, morsetto a vite) vanno saldati mediante saldatore a stilo e filo di stagno. Si precisa che tale filo di stagno è in realtà una lega di stagno-rame, talvolta con una piccola percentuale di piombo. All’interno della sezione del filo si trova una piccola quantità di flussante che aumenta la fluidità del metallo fuso.

Terminata la fase di montaggio e controllo dei componenti saldati, si è passati alla prima connessione ad un PC per la programmazione della memoria. Per tale operazione si è talvolta riscontrato un problema di comunicazione tra le due parti, dovuto al fatto che il microcontrollore rimaneva in stato di reset alla sua connessione mediante cavo USB. Lo stesso problema si presentava su tutte le schede ad esclusione di una, che non presentava alcuna difficoltà di comunicazione/programmazione. Sono allora stati effettuati alcuni tentativi di sostituzione dei valori dei resistori relativi ai circuiti di reset e boot, passando da 10kΩ, a 1kΩ. Tali tentativi non hanno rivelato miglioramenti accettabili, in quanto il malfunzionamento si ripresentava ad intervalli apparentemente casuali.

Confrontando la scheda funzionante con quelle che davano problemi, ci si è resi conto che su essa non erano ancora stati montati i tre resistori da 33Ω (R8, R9, R16). Sulle restanti quattro schede, tali resistori erano presenti, seppur nel valore di 49.9Ω. La sostituzione di R16 con il valore di 270Ω ha immediatamente risolto il problema. La linea in cui R16 è presente è la REF\_CLK del transceiver Ethernet. Ragionevolmente, si pensa che il segnale a 50MHz generato da esso possa interferire con il funzionamento del microcontrollore. Tale problema potrebbe essere amplificato dal cattivo posizionamento dei condensatori di disaccoppiamento vicino ai componenti sulla scheda.

Passati con successo alla programmazione di tutte le schede, si è proceduto al caricamento di un codice di base che si occupasse di far lampeggiare il led a bordo della scheda. Avere un led permette di effettuare facilmente operazioni di debug, basti pensare che prevedendo un ciclo di quattro accensioni ravvicinate indica il reset del microcontrollore, e un lampeggio a frequenza di 1Hz indica il regolare svolgimento del codice caricato. È importante che tutte le operazioni, tra cui la gestione del led, siano gestite come macchina a stati, quindi evitando funzioni bloccanti (cicli) e funzioni di ritardo (delay, sleep). Si impiega allora il confronto di una variabile con un free-running timer, che conta il numero di microsecondi trascorsi dall’avvio della macchina.

A seguito di questo primo test, il firmware caricato è quello reperito come esempio in rete, che gestisce la comunicazione con il transceiver, e implementa i vari protocolli di rete per la costruzione di un server HTTP (si veda la sezione successiva per i dettagli). Al primo avvio, tale comunicazione non funzionava, fatto indicato dalla mancata accensione dei led di link e activity a bordo del connettore RJ45. Dopo un accurato confronto tra lo schematico e il codice firmware, si è riscontrato che il segnale di reset del transceiver, pur connesso al microcontrollore, non era gestito dal suo GPIO. Si sottolinea che il segnale di reset è attivo a livello basso; deve pertanto essere forzato a livello alto per permettere lo sblocco dell’integrato. Ponendo, mediante istruzioni firmware, il segnale sul pin relativo (GPIO13) a livello alto, il malfunzionamento si è immediatamente risolto.

# Programmazione fw e utilizzo pratico (con o senza sw)

La progettazione del firmware è avvenuta in parte in parallelo a quella hardware. Normalmente avviene per quasi tutti i sistemi a microcontrollore, in quanto è necessario adattare le soluzioni firmware a quelle hardware in corso di sviluppo, e viceversa. Ad esempio, può accadere che una periferica obblighi ad impegnare determinati pin del microcontrollore: è quindi il caso di adattare il codice per impiegare tali pin. Nel nostro lavoro di gruppo, si è partiti da un progetto trovato in rete, dedicato alla piattaforma RP2040 e al suo interfacciamento con il chip LAN8720. Tale firmware impiega il PIO per la generazione dei segnali RMII mediante precisi pin del microcontrollore. Si è preferito mantenere invariato il codice, adattando allora le connessioni fisiche nel circuito.

Il codice del progetto fornito si occupa della comunicazione con il transceiver Ethernet citato, e della gestione dei protocolli di livello superiore dello stack TCP/IP, mediante le librerie LwIP. Ricordando brevemente lo stack dei livelli, si ha:

|  |
| --- |
| APPLICAZIONE |
| TRASPORTO |
| RETE |
| COLLEGAMENTO |
| FISICO |

Il progetto viene fornito compreso di tutte le librerie contenenti strutture dati, costanti e funzioni in codice C per la gestione di diversi protocolli di trasporto (TCP e UDP) e di applicazione (http, SMTP, MQTT, DNS). Inoltre, si trova un file sorgente main.c che impiega alcune di tali librerie per la creazione di un server web. L’applicazione di base fornisce due pagine web (HTML), che vengono compilate esternamente al microcontrollore e gestite attraverso il file system http.

L’analisi del codice sorgente fornito, porta alle seguenti conclusioni:

* La comunicazione tra MCU e transceiver è affidata al PIO
* Mediante le code delle FSM del PIO e il DMA vengono estratti e trasmessi i frame Ethernet
* Lo stack LwIP si occupa di incapsulare e deincapsulare i messaggi tra i vari livelli dello stack TCP/IP
* L’applicazione gestisce un server http che fornisce una pagina HTML di indice.

Prima di lavorare al codice fornito nel progetto, si è cercato di sviluppare i driver di comunicazione tra il micro, il GPS e il modulo RTC in modo separato. È quindi stato creato un progetto mediante l’editor VSCode, nel quale sono inseriti file di libreria creati dal gruppo e un file main.c che testa le funzioni dedicate, ossia:

* GPS:
  + Inizializzazione della comunicazione, in rispetto dell’impostazione usata dal modulo;
  + Callback uart\_rx, atta a leggere i dati UART in ingresso; essa viene richiamata ad ogni dato valido ricevuto dalla linea RX;
  + Macchina a stati software che si occupa della formattazione della stringa corretta per l’estrazione di data e ora;
* RTC
  + Scrittura dei registri, avvio dell’oscillatore
  + Lettura dei registri;
  + Confronto del dato orario con il tempo GPS.

## 6.1 Comunicazione con il transceiver Ethernet LAN8742

La comunicazione di controllo del transceiver LAN8742 è gestita mediante le linee MDIO (management data I/O) e MDC (management data clock). I due segnali sono controllati da altrettanti pin GPIO del microcontrollore tramite istruzioni software: il clock viene generato con transizioni 1/0 successive dello stato del pin MDC e la il pin MDIO assume il valore logico di ciascun bit di dato trasmesso/letto sequenzialmente. Una scrittura da parte del microcontrollore avviene allora mediante scorrimento bit a bit dei dati trasmessi, scandito dal segnale di clock. Analogo funzionamento assume la lettura, con la differenza che il pin MDIO assume la funzione di input digitale.

Un esempio di utilizzo della comunicazione di controllo è la lettura tramite polling dello stato del link Ethernet, che avviene ripetutamente con la chiamata alla funzione “netif\_rmii\_ethernet\_mdio\_read” all’interno del ciclo infinito del task “netif\_rmii\_ethernet\_loop”.

**N.B.** il codice sfrutta la capacità dual core del microcontrollore, delegando tutta la gestione della comunicazione Ethernet e dello stack TCP/IP al core 1. Infatti, l’istruzione precedente all’ingresso nel ciclo infinito presente nel programma principale è la chiamata a “multicore\_launch\_core1(netif\_rmii\_ethernet\_loop)” e indica che tutto il processo è delegato al solo core 1, lasciando l’altro disponibile per altre applicazioni.

La gestione dei segnali digitali scambiati tra MCU e transceiver (con i quali esso genera i segnali fisici Ethernet) è affidata al PIO. Nel dettaglio, un blocco PIO utilizza due macchine a stati, una per i segnali in trasmissione (TX0, TX1 e TX\_EN) e una per quelli in ricezione (RX0, RX1 e CRS).

Il PIO si appoggia al blocco DMA (direct memory access), un blocco che si occupa dei trasferimenti da e verso la memoria e le periferiche di dati ad alta velocità, senza interessare il processore. Questo sistema assicura alte velocità di trasferimento ed evita stalli dovuti a istruzioni di salto e trasferimento di dati in memoria da parte della CPU.

La ricezione è gestita ad alto livello dalla funzione “netif\_rmii\_ethernet\_poll”, che si occupa di:

* Valutare lo stato della linea, discriminando se sono presenti dati in entrata;
* Valutare la dimensione dell’eventuale frame Ethernet in entrata;
* Allocare un buffer atto a contenere il frame in arrivo;
* Configurare il DMA per il salvataggio del frame, impostando i pin di ricezione della macchina a stati come sorgente di dati, e il trasferimento dalla RX\_FIFO del blocco PIO al canale DMA corretto;
* Lanciare l’esecuzione della macchina a stati di ricezione per l’acquisizione in tempo reale dei dati.

La trasmissione avviene in maniera simile, mediante la funzione “netif\_rmii\_ethernet\_output” che si occupa di:

* Inizializzare il frame da trasmettere, includendo il payload passato dal livello superiore (TCP o UDP) e calcolarne il CRC per il controllo errori;
* Configurare il DMA passando come parametri il frame Ethernet formattato e la macchina a stati del blocco PIO che si occuperà della trasmissione verso il transceiver.

## 6.2 Comunicazione con il modulo RTC

La comunicazione con il modulo RTC avviene via standard I2C bus, e l’impostazione e la lettura dei dati orari avvengono mediante scrittura e lettura di registri interni al modulo. Di questi fanno parte i registri di controllo, i registri di timekeeping, ossia data e ora gestiti dall’integrato, e i registri di impostazione di allarmi e interrupt dedicati alla gestione di eventi quali la scadenza di un timer, o lo scoccare di un determinato orario.

I registri citati sono di lettura e scrittura. Prendendo in esame l’impostazione e la successiva lettura dei registri orari e di calendario, le due operazioni sono gestite mediante comunicazione in standard seriale I2C. è uno standard sincrono che permette la connessione di molte periferiche.

Il protocollo prevede due linee di segnale, SDA (Data) e SCL (Clock). L’apertura di una transazione è sempre gestita da un master, nel nostro caso il microcontrollore. L’indirizzamento di uno slave si effettua, dopo il segnale di start, mediante i primi 7 bit inviati sulla linea SDA, e l’indicazione di lettura o scrittura è data dal valore dell’ottavo bit di indirizzamento. Dopo ogni dato ricevuto dallo slave, esso risponde ACK (in caso di indirizzamento corretto) o NACK (in caso di indirizzo errato). Il segnale è dato dal nono bit della transazione. Nel caso di ACK, lo slave forzerà basso tale bit, nel caso di NACK, lascerà libera la linea, che si porterà a livello auto data la presenza dei resistori di pullup.

A livello concettuale, una scrittura di un registro avviene nella seguente procedura:

1. START (S): transizione 1/0 della linea SDA con SCL a livello alto;
2. ADDR+W: indirizzo a 7 bit dello slave e ottavo bit a 0 per indicare una scrittura.
3. ACK/NACK dello slave. Acknowledgement è indicato dal nono bit forzato a ‘0’ dallo slave.
4. INDIRIZZO del registro in scrittura (8 bit)
5. ACK/NACK dello slave interpellato
6. DATO IN SCRITTURA
7. ACK/NACK dello slave interpellato
8. STOP (P)

Si riporta un esempio di scrittura di un singolo registro del modulo MCP7940:

Diagram

Description automatically generated

Dalla fig. precedente, si evince che l’indirizzo a 7 bit del modulo è 6F (HEX).

Gli indirizzi dei dati orari sono forniti dal datasheet e sono così mappati:

|  |  |
| --- | --- |
| SECONDI | 0x00 |
| MINUTI | 0x01 |
| ORE | 0x02 |
| GIORNO | 0x03 |
| DATA | 0x04 |
| MESE | 0x05 |
| ANNO | 0x06 |

Per la scrittura del registro secondi, l’operazione di write da parte del microcontrollore sarà strutturata come segue:

**S; ADDR+W; SECS\_REG\_ADDR; VALORE\_SECS; P**

È possibile, secondo le specifiche del MCP7940, eseguire una scrittura multipla dei registri, sfruttando l’auto incremento dell’indice dei registri interno al modulo. La scrittura di tutti i dati orari avviene allora nel seguente modo:

**S; ADDR+W; SECS\_REG\_ADDR; 🡪**

* **VAL\_SECS; VAL\_MINS; VAL\_HRS; VAL\_DAY;**
* **VAL\_WKDY; VAL\_MNTH; VAL\_YR; P**

**A picture containing timeline

Description automatically generated**

Questo tipo di scrittura permette di risparmiare tempo nello svolgimento dell’operazione. Inoltre, poiché i registri subiscono un latching (vengono “congelati”) durante le operazioni di comunicazione, si assicura che non avvengano errori di impostazione o lettura dei dati. È quindi conveniente impiegare sempre letture e scritture multiple quando necessario.

**N.B.** il registro “secondi” contiene un bit di controllo che si occupa dello start del conteggio del tempo: è necessario porre tale bit a 1 per imporre l’avvio dell’oscillatore.

L’operazione di lettura avviene in modo analogo, con la differenza che l’indirizzamento dello slave avviene con l’ottavo bit a livello alto (READ).

Diagram

Description automatically generated

Dopo l’invio del byte ADDR+WRITE, il master trasmette l’indirizzo del registro da leggere (esempio, 0x00 per i secondi). Al termine di tale byte e del relativo ACK dallo slave, il master invia un nuovo segnale di START, seguito da ADDR+READ. Da questo momento lo slave prende il controllo della linea e invia il byte richiesto. Al termine del byte, il master risponde con NACK e P, chiudendo la comunicazione e liberando la linea.

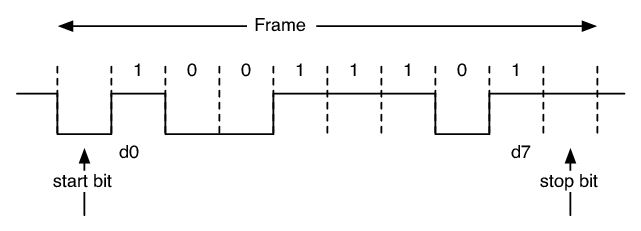
Anche per la lettura è conveniente usare il meccanismo dell’auto incremento dell’indice. Il principio è analogo a quello della scrittura multipla, e viene schematizzato di seguito.

Diagram

Description automatically generated with medium confidence

## 6.3 Comunicazione con il GPS

Il modulo gps è in grado di trasmettere stringhe al microcontrollore attraverso l’interfaccia UART. Di default, il modulo trasmette i dati in formato 9600, 8N1, ossia a un data rate di 9600 baud, con caratteri di 8 bit, nessun controllo parità e 1 bit di stop. Ogni dato è trasmesso come da figura seguente (fig. cc):



Il segnale di start è una transizione 1/0 della linea. In seguito si trovano 8 bit di dato, trasmessi in ordine temporale dal LSB al MSB. Nel tempo del bit successivo all’ottavo, il segnale a livello alto indica il bit di stop.

Le librerie fornite dal SDK dell’RP2040 contengono funzioni di gestione della ricezione UART. Poiché detta comunicazione è però asincrona, e le funzioni sono scritte in modo da eseguire un ciclo di attesa dei dati, è inopporuno usare tali funzioni, poiché costringono lo svolgimento del codice a bloccarsi in attesa di dati in ricezione. Sono cosiddette funzioni blocking.

Per evitare questo inconveniente, conviene impostare e sfruttare l’interrupt generato dalla periferica alla ricezione di ogni byte. In questo capitolo non si vedrà l’impiego del DMA (direct memory access) per la ricezione.

NB: Un interrupt è la gestione software di un evento asincrono rilevato dall’hardware. Tra i vari interrupt si ricordano la scadenza di un timer hardware, l’avvenuta conversione A/D, il cambio di stato logico di un GPIO.

Viene pertanto impostata una funzione (chiamata callback, o handler), che viene automaticamente eseguita all’avvenimento di un evento che scatena un interrupt. L’impostazione data lega l’evento “byte ricevuto dalla periferica” alla funzione “uart\_rx\_callback”. All’interno di questa funzione ci si occuperà di inserire i caratteri ricevuti in un buffer (stringa) da cui si estrarranno i dati di interesse.

I messaggi (sentences) trasmessi dal gps sono formattati in formato ASCII e rispettano lo standard NMEA0183. I vari campi della sentence sono separati dal carattere ‘,’.

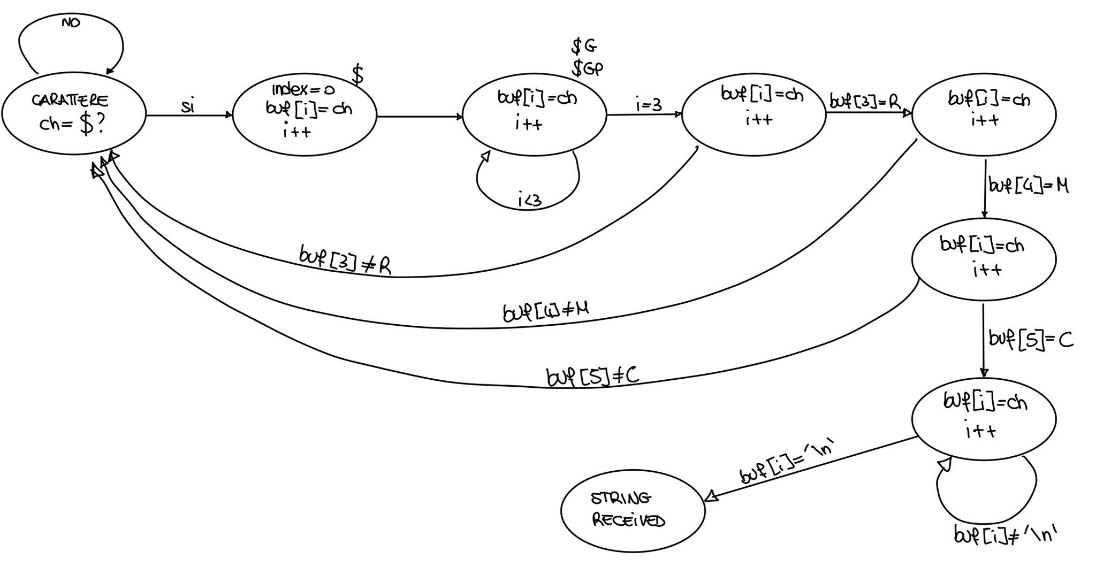
Un messaggio è pertanto strutturato come segue:

* 1 carattere di apertura ‘$’
* Una sottostringa di intestazione di cinque caratteri, che indica la costellazione di satelliti cui fa riferimento il messaggio (primi due caratteri), e l’informazione che esso porta: GP, GL, GN indicano rispettivamente le costellazioni Navstar GPS, Glonass e GNSS. I successivi tre caratteri indicano il tipo di stringa (GGA, GLL, …)
* Serie di campi separati tra loro dal carattere virgola
* Terminatore di stringa ‘\r’, ‘\n’

Il dato necessario per il progetto in analisi è quello dell’orario. L’ora viene fornita come primo campo del messaggio RMC, riferita al tempo UTC (Universal Time Coordinated). Rispetto al fuso orario di Roma, UTC indica un’ora in anticipo. Va considerata anche la presenza del DST, ossia l’impostazione dell’ora legale, che porta la differenza a due ore tra LT e UTC. La data è presentata nel nono campo della stringa.

La procedura all’interno della funzione callback\_uart\_rx si occupa di discriminare i caratteri ricevuti, e realizza una macchina a stati finiti software. Si va pertanto a controllare se il nuovo carattere arrivato è di tipo terminatore, start o separatore. Come caso base del check (stato “0”), si considera che tutti i caratteri ricevuti dallo startup del microcontrollore siano invalidi, e pertanto da scartare: tale scelta assicura che in nessun caso si vadano a considerare caratteri appartenenti a stringhe non complete. Solo dalla ricezione del primo carattere ‘$’ si assume che le stringhe successive siano integre. L’azione da intraprendere allora (stato “1”) è azzerare l’indice del buffer che conterrà la stringa in ricezione, e inserirvi il carattere iniziale (‘$’ appunto) all’indice 0. I caratteri successivi verranno bufferizzati (= salvati temporaneamente) nelle posizioni successive. Il controllo della sentence in corso di ricezione avviene carattere per carattere, verificando se, alla ricezione dei caratteri in posizione 3, 4 e 5 del buffer, i caratteri effettivamente ricevuti sono, nell’ordine, ‘R’, ‘M’ e ‘C’. in caso almeno una delle tre lettere fosse diversa da quella attesa, la stringa viene ritenuta invalida (e da scartare), e la MSF ritorna allo stato “0”, in attesa del prossimo avvio di stringa.

Se i tre successivi controlli sui caratteri R, M, C hanno esito positivo, la MSF procede nella costruzione della stringa, fino alla ricezione di un carattere terminatore ‘\n’, che indica la chiusura della sentence in corso. Nello schema seguente è descritta la macchina a stati impiegata.



Il segnale TIMEPULSE, o 1PPS, è rilevato dall’interrupt (IRQ) associato al pin del microcontrollore relativo (GPXX). In particolare, è impostato in modo da chiamare l’apposita funzione (callback) ad ogni cambio di stato logico da basso ad alto, in modo da rilevare il fronte di salita del segnale PPS. Esso avviene ad intervalli di 1s estremamente precisi, ed è utile per gestire la sincronia di eventi. Ad esempio, si possono sincronizzare operazioni che avvengono simultaneamente in luoghi separati.

# Approfondimenti

fref

# Allegati

frefe