



## **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: Trenti Elia***

***Matricola: 209212***

***Studente: Amorth Matteo***

***Matricola: 209615***

***Studente: Pasquini Giuseppe***

***Matricola: 209048***

***Studente: Titton Giulia***

***Matricola: 209359***

***Studente: Paganin Andrea***

***Matricola: 209926***

***Anno Accademico 2021/2022***

# INDICE

<b>1. INTRODUZIONE</b>	<b>3</b>
1.1 Descrizione del Corso	3
1.2 Il Raspberry RP2040 e l'EVB Raspberry Pi Pico	4
1.3 Descrizione generale e tool impiegati	6
1.3.1 KiCad 6	6
1.3.2 Github	8
1.3.3 Firmware - Visual Studio Code	8
<b>2. SPECIFICHE</b>	<b>9</b>
2.1 Il microcontrollore	10
2.2 Circuito LAN	11
2.3 Il circuito di alimentazione	11
2.4 Il modulo RTC	12
2.5 Il modulo GPS	12
<b>3. PROGETTAZIONE CAD SCHEMATIC</b>	<b>13</b>
3.1 Il microcontrollore	13
3.1.1 Il clock	14
3.1.2 La memoria FLASH	15
3.1.3 Le periferiche di interesse	16
3.1.3.1 GPIO	16
3.1.3.2 Timer	17
3.1.3.3 USB	18
3.1.3.4 UART	21
3.1.3.5 I2C	21
3.2 Il circuito LAN	24
3.2.1 Transceiver	26
3.2.2 Trasformatori	28
3.3 Il circuito di alimentazione	28
3.4 Il modulo GNSS	30
3.4.1 SAM-M8Q	32
3.4.2 PAM-7Q	34
3.5 Il modulo RTC	35
3.6 Test Point	38
3.7 Uscite digitali	38
3.8 Ingressi digitali	39
<b>4. PROGETTAZIONE CAD PCB LAYOUT</b>	<b>41</b>
4.1 Assegnamento dei footprint	41
4.1.1 Creazione nuovo footprint	41
4.2 Definizione delle regole di progettazione	42
4.3 Progettazione del layout della scheda	43
4.4 Sbroglio e ottimizzazione	46
4.5 I file Gerber	47
4.6 Alloggiamento	48

<b>5. ASSEMBLAGGIO MANUALE IN LABORATORIO E TEST FUNZIONALI</b>	<b>49</b>
<b>6. PROGRAMMAZIONE FW E UTILIZZO PRATICO</b>	<b>54</b>
6.1 Comunicazione con il transceiver Ethernet LAN8742	55
6.2 Comunicazione con il modulo RTC	56
6.3 Comunicazione con il GPS	58
<b>7. APPROFONDIMENTO - PIO</b>	<b>61</b>
7.1 Struttura di una macchina a stati	61
7.1.1 IO Mapping	62
7.1.2 ISA	62
<b>8. BUG e IMPLEMENTAZIONI MANCANTI</b>	<b>64</b>
8.1 Interfaccia di rete	64
8.2 Monitoraggio orario	64
8.3 Modulo GPS	64
<b>9. ALLEGATI</b>	<b>66</b>
9.1 Assegnamento footprint	66
9.2 File Gerber	68
9.3 Macchina a stati per la lettura delle stringhe del GPS	72
9.4 BOM	73

# 1. INTRODUZIONE

## 1.1 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 I<sup>2</sup>C. 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 NMEA0183) 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:

- i. LAN-NTP
- ii. Sun Tracker
- iii. MPPT Tracker
- iv. 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.

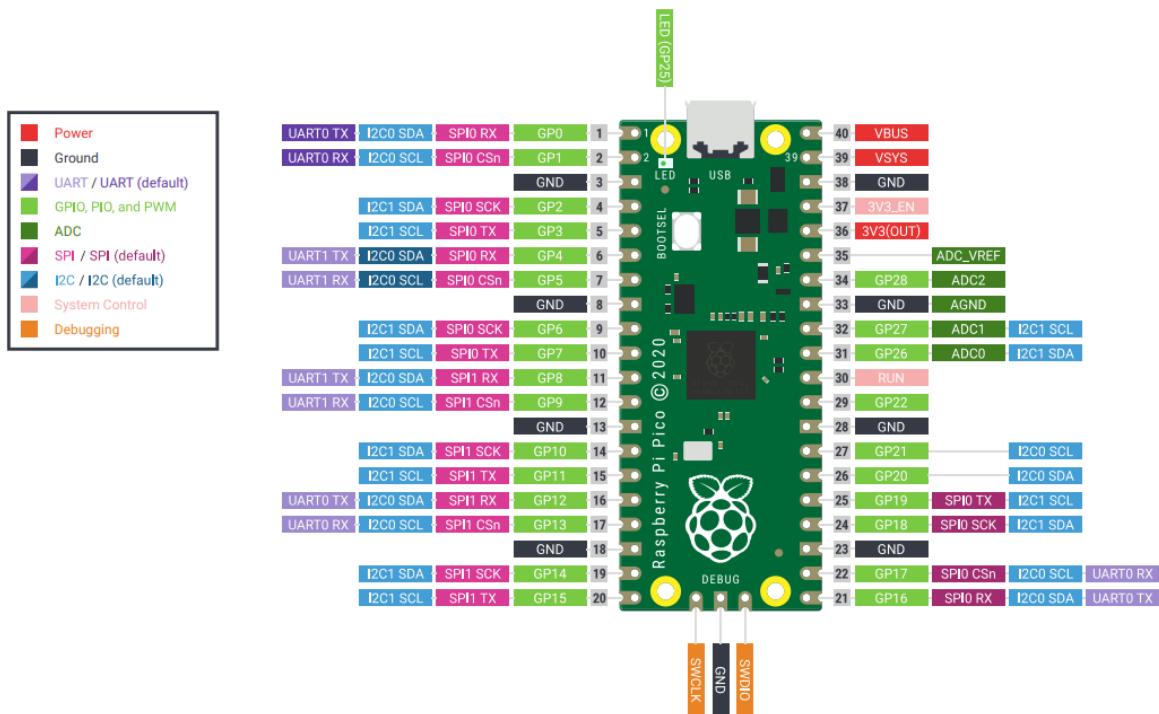
## 1.2 Il Raspberry RP2040 e l'EVB Raspberry Pi Pico

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, una piattaforma di sviluppo a basso costo ma flessibile, 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.



*Figura 1 - Raspberry Pi Pico*

Le caratteristiche principali della demo board sono (da datasheet):

- *RP2040* microcontroller with 2MByte Flash;
- Micro-USB B port for power and data;
- 40 pin 21x51 'DIP' style 1mm thick PCB with 0.1" through-hole pins also with edge castellations
  - Exposes 26 multi-function 3.3V General Purpose I/O (GPIO);
  - 23 GPIO are digital-only and 3 are ADC capable;
  - Can be surface mounted as a module;
- 3-pin ARM Serial Wire Debug (SWD) port;
- Simple yet highly flexible power supply architecture
  - Various options for easily powering the unit from micro-USB, external supplies or batteries;
- High quality, low cost, high availability;
- Comprehensive SDK, software examples and documentation;
- Dual-core cortex M0+ at up to 133MHz
  - On-chip PLL allows variable core frequency;
- 264kByte multi-bank high performance SRAM;
- External Quad-SPI Flash with eXecute In Place (XIP) and 16kByte on-chip cache;
- High performance full-crossbar bus fabric;
- On-board USB1.1 (device or host);
- 30 multi-function General Purpose IO (4 can be used for ADC)
  - 1.8-3.3V IO Voltage (NOTE Pico IO voltage is fixed at 3.3V)
- 12-bit 500ksps Analogue to Digital Converter (ADC);
- Various digital peripherals
  - 2 × UART, 2 × I2C, 2 × SPI, 16 × PWM channels;
  - 1 × Timer with 4 alarms, 1 × Real Time Counter;
- 2 × Programmable IO (PIO) blocks, 8 state machines total
  - Flexible, user-programmable high-speed IO;
  - Can emulate interfaces such as SD Card and VGA;

Pi Pico fornisce circuiti esterni minimi e flessibili per utilizzare il microcontrollore *RP2040*.

La maggior parte dei pin del microcontrollore viene trasferita su pin I/O posti sui lati sinistro e destro della scheda.

Quattro pin sono utilizzati per le funzioni interne: pilotaggio del LED onboard, controllo delle alimentazioni e rilevazione delle tensioni del sistema.

La demo board è stata progettata per utilizzare connettori maschio saldati da 0,1 pollici o può essere utilizzato come "modulo" montabile su superficie, grazie ai fori castellati.

## 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, programma open source per il disegno di circuiti stampati e schematici. La versione è 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).

La realizzazione del circuito stampato è suddivisa in due macro attività: progettazione, prototipazione così suddivise:

- **Progettazione**

- Realizzazione dello schematico;
- Scelta dei simboli e footprint disponibili in libreria;
- Implementazione simboli custom per il progetto;
- Disegno di footprint custom;
- Generazione netlist e BOM;
- Creazione layout e posizionamento componenti;
- Sbroglio circuitale;
- Definizione aree rame;
- Creazione file GERBER e NC-Drill per la produzione del PCB.

- **Prototipazione**

- Produzione PCB (realizzazione esterna);
- Montaggio SMD (Applicazione pasta saldante tramite stencil serigrafico, saldatura tramite piastra termica);
- Montaggio componenti Through Hole;
- Test funzionali.

La figura seguente riporta la procedura di progettazione del circuito stampato in Kicad 6.

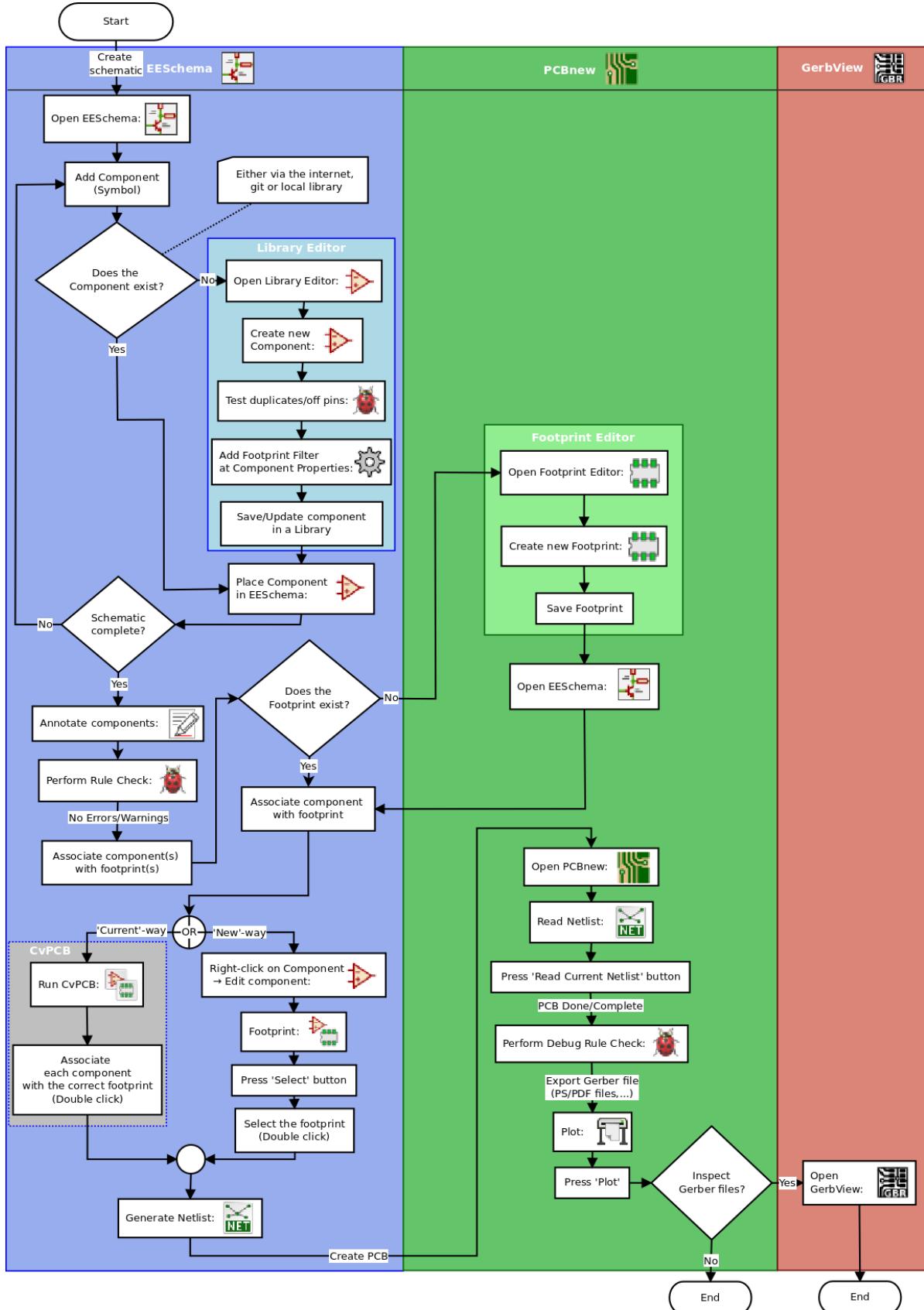
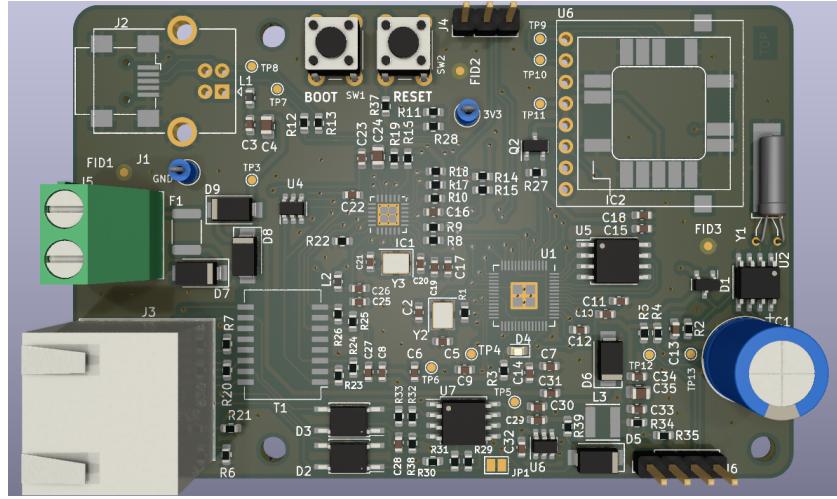


Figura 2 - Kicad Workflow (Fonte: Kicad6)



*Figura 3 - Visualizzazione 3D del PCB completo*

### 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 indicato nella sezione relativa al firmware (sezione 9, Getting started with Raspberry Pi Pico).

## 2. 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, e variano in genere intorno ai 12-15 pF;;
- 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;
- PCB doppio layer (Top/Bottom), dimensioni 85mm x 56 mm, spessore 1,6mm e 35 micron di rame.

Una volta definiti i componenti su descritti, si è passati a definire 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.

## 2.1 Il microcontrollore

Come anticipato, il microcontrollore di interesse per il progetto è l'RP 2040, il quale gestisce la parte di calcolo e implementa le periferiche, come ad esempio I/O, ADC, PWM ed interfacce di comunicazione I2C e USB. La scelta di questo microcontrollore 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.

Le specifiche tecniche sono le seguenti (da datasheet):

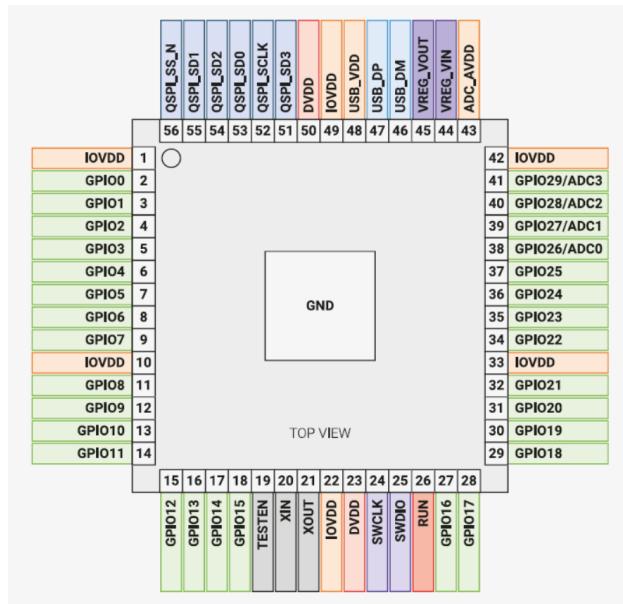
- Dual ARM Cortex-M0+ @ 133MHz;
- 264kB on-chip SRAM in sei banchi indipendenti;
- Supporto fino a 16MB di memoria flash off-chip attraverso bus QSPI dedicato;
- DMA controller;
- Fully-connected AHB crossbar;
- Periferiche di interpolazione e di divisione di interi;
- On-chip programmable LDO per generare la tensione di core;
- 2 on-chip PLLs per generare i clock per l'USB e per il core;
- 30 pin per il GPIO, 4 dei quali possono essere usati come input analogici
  - 4 canali ADC con sensore di temperatura interno, 0.5 MSa/s, 12-bit di conversione;
- 6 IO dedicati all'SPI Flash;
- IO programmabile per supporto alle periferiche;
- Hardware dedicato per le periferiche più utilizzate;
- Periferiche
  - 2 UARTs;
  - 2 SPI controllers;
  - 2 I2C controllers;
  - 16 canali PWM;
  - USB 1.1 controller and PHY, con supporto host e device;
  - 8 PIO state machines.

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”).



*Figura 4 - Pinout microcontrollore RP2040*

## 2.2 Circuito LAN

Una Local Area Network (in italiano rete in area locale, o rete locale), in informatica e telecomunicazioni, indica una rete informatica di collegamento tra più computer, estendibile anche a dispositivi periferici condivisi, che copre un'area limitata.

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 (sezione 3.2.1). 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.

### **2.3 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.

## **2.4 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 0.1F, come da suggerimento del datasheet del componente.

## **2.5 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.

### 3. 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).

#### 3.1 Il microcontrollore

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.

In questo progetto non vengono utilizzati tutti i pin presenti ma solo i necessari:

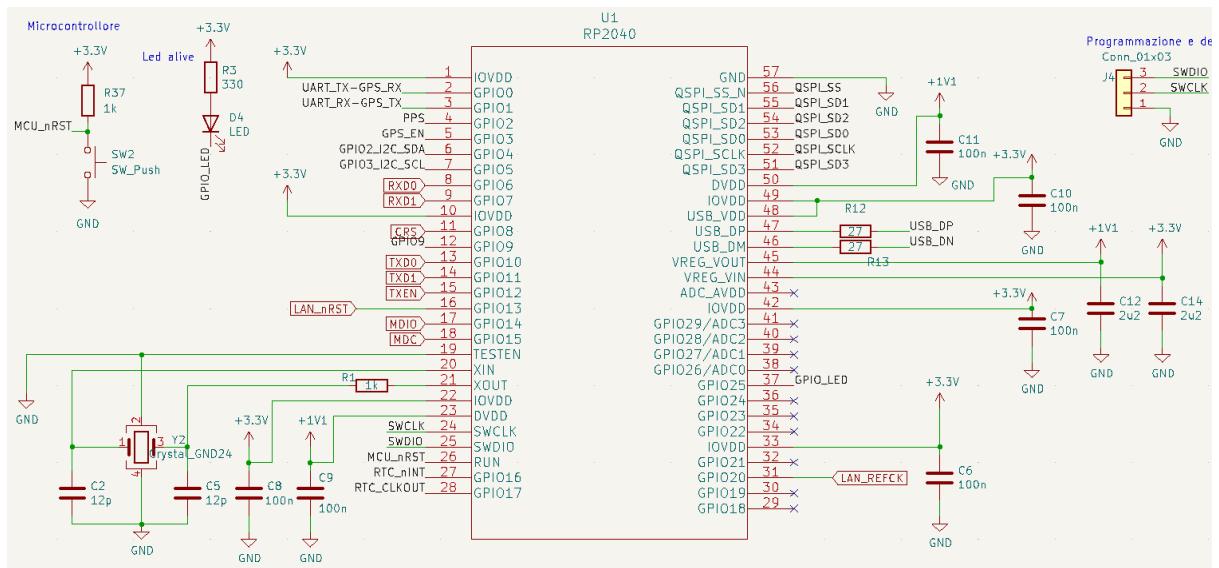
- La comunicazione USB avviene tramite i pin 46 e 47 (USB\_DM, USB\_DP);
- La memoria flash è connessa tramite i pin 51 - 56 rispettivamente QSPI\_SD3, QSPI\_SCLK, QSPI\_SD0, QSPI\_SD2, QSPI\_1, QSPI\_SS;

- Il GPS è collegato mediante le connessioni 2 (GPIO0) - 5 (GPIO3) che rappresentano UART\_TX-GPS\_RX, UART\_TX-GPS\_TX, PPS, GPS\_EN;
  - Il modulo RTC è interfacciato tramite la comunicazione I2C pin 6 (GPIO4) e 7 (GPIO5) oltre a RTC\_CLKOUT (28 - GPIO17) e RTC\_nINT (27 - GPIO16);
  - Il connettore di programmazione utilizza le connessioni SWCLK (24) e SWDIO (25);
  - E' presente un LED di stato sul pin 37 (GPIO 25) necessario per le funzioni base e il debug software;
  - Si è previsto un pulsante di reset normalmente aperto sul pin 6 MCU\_nRST;
  - Il modulo LAN necessita di 10 connessioni con il microcontrollore: LAN\_nRST (16 - GPIO13), CRS (11 - GPIO8) MDIO (17 - GPIO14) RXD1 (9 - GPIO6) TXD1 (14 - GPIO11) MDC (18 - GPIO15) LAN\_REFCK (31 - GPIO20) RXD0 (8 - GPIO6) TXD0 (13 - GPIO10) TXEN (15 - GPIO12).

I pin di alimentazione (IOVDD, DVDD, VREG\_VIN, VREG\_VOUT, USB\_VDD) sono collegati in shunt a condensatori di 100nF e 2.2uF per ridurre il ripple e il rumore dell'alimentazione (bypass delle alte frequenze a massa).

L'alimentazione per gli ingressi analogici non è collegata in quanto non vengono utilizzati, inoltre il microcontrollore non utilizza un riferimento di tensione interno indipendente da VDD (FVR, Fixed Voltage Reference), questo non garantirebbe una corretta lettura dell'ingresso se fossero utilizzati.

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



*Figura 5 - Circuito microcontrollore nello schematico*

### 3.1.1 || 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). L'oscillatore esterno da 48 MHz è collegato tra i pin 20 e 21 ed è necessario per il funzionamento della scheda;
- 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 all'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). Viene utilizzato per fornire il clock dei core durante la fase di boot, ha una frequenza programmabile nel range di 1.8MHz - 12MHz e un valore nominale di 6.5MHz.

### 3.1.2 La memoria FLASH

Nella progettazione si è predisposto l'utilizzo della memoria Windbond W25Q128JV (3V 128M-BIT SERIAL FLASH MEMORY WITH DUAL/QUAD SPI) in package SO8. Durante il montaggio, per la disponibilità in laboratorio, è stata montata la memoria W25Q16JV (3V 16M-BIT SERIAL FLASH MEMORY WITH DUAL/QUAD SPI), la quale offre 16 Mb di spazio di archiviazione ed opera ad una tensione di alimentazione tra 2.7V e 3.6V con un consumo di corrente limitato. Essa differisce dalla prima solamente per la capacità della memoria. Questa variazione non genera problemi durante la prototipazione in quanto il package è il medesimo ed inoltre la quantità di Bit è sufficiente per il nostro scopo.

La comunicazione tra la memoria e 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).

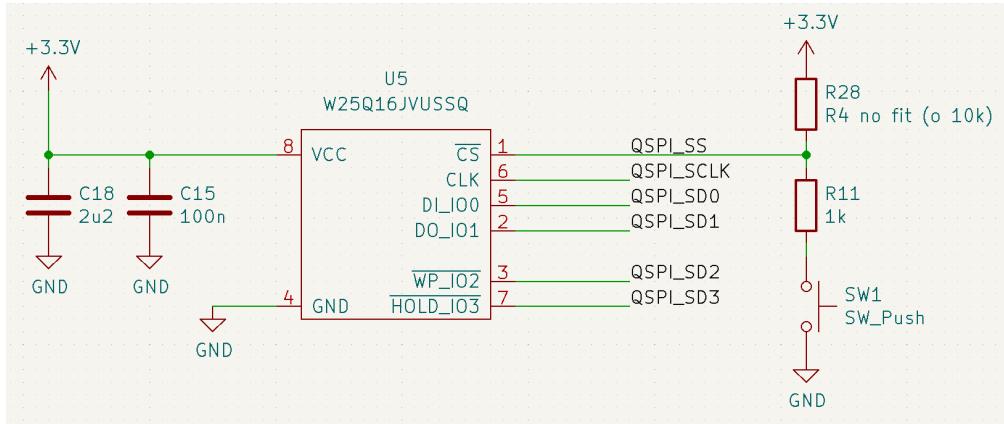


Figura 6 - Circuito memoria FLASH nello schematico

### 3.1.3 Le periferiche di interesse

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

#### 3.1.3.1 GPIO

L'*RP2040* dispone di 36 pin GPIO (General Purpose Input Output), 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).

GPIO	F1	F2	F3	F4	F5	F6	F7	F8	F9
0	SPI0 RX	UART0 TX	I2C0 SDA	PWM0 A	SIO	PIO0	PIO1		USB OVCUR DET
1	SPI0 CSn	UART0 RX	I2C0 SCL	PWM0 B	SIO	PIO0	PIO1		USB VBUS DET
2	SPI0 SCK	UART0 CTS	I2C1 SDA	PWM1 A	SIO	PIO0	PIO1		USB VBUS EN
3	SPI0 TX	UART0 RTS	I2C1 SCL	PWM1 B	SIO	PIO0	PIO1		USB OVCUR DET
4	SPI0 RX	UART1 TX	I2C0 SDA	PWM2 A	SIO	PIO0	PIO1		USB VBUS DET
5	SPI0 CSn	UART1 RX	I2C0 SCL	PWM2 B	SIO	PIO0	PIO1		USB VBUS EN
6	SPI0 SCK	UART1 CTS	I2C1 SDA	PWM3 A	SIO	PIO0	PIO1		USB OVCUR DET
7	SPI0 TX	UART1 RTS	I2C1 SCL	PWM3 B	SIO	PIO0	PIO1		USB VBUS DET
8	SPI1 RX	UART1 TX	I2C0 SDA	PWM4 A	SIO	PIO0	PIO1		USB VBUS EN
9	SPI1 CSn	UART1 RX	I2C0 SCL	PWM4 B	SIO	PIO0	PIO1		USB OVCUR DET
10	SPI1 SCK	UART1 CTS	I2C1 SDA	PWM5 A	SIO	PIO0	PIO1		USB VBUS DET
11	SPI1 TX	UART1 RTS	I2C1 SCL	PWM5 B	SIO	PIO0	PIO1		USB VBUS EN
12	SPI1 RX	UART0 TX	I2C0 SDA	PWM6 A	SIO	PIO0	PIO1		USB OVCUR DET
13	SPI1 CSn	UART0 RX	I2C0 SCL	PWM6 B	SIO	PIO0	PIO1		USB VBUS DET
14	SPI1 SCK	UART0 CTS	I2C1 SDA	PWM7 A	SIO	PIO0	PIO1		USB VBUS EN
15	SPI1 TX	UART0 RTS	I2C1 SCL	PWM7 B	SIO	PIO0	PIO1		USB OVCUR DET
16	SPI0 RX	UART0 TX	I2C0 SDA	PWM0 A	SIO	PIO0	PIO1		USB VBUS DET
17	SPI0 CSn	UART0 RX	I2C0 SCL	PWM0 B	SIO	PIO0	PIO1		USB VBUS EN
18	SPI0 SCK	UART0 CTS	I2C1 SDA	PWM1 A	SIO	PIO0	PIO1		USB OVCUR DET
19	SPI0 TX	UART0 RTS	I2C1 SCL	PWM1 B	SIO	PIO0	PIO1		USB VBUS DET
20	SPI0 RX	UART1 TX	I2C0 SDA	PWM2 A	SIO	PIO0	PIO1	CLOCK GPIO0	USB VBUS EN
21	SPI0 CSn	UART1 RX	I2C0 SCL	PWM2 B	SIO	PIO0	PIO1	CLOCK GPIO0	USB OVCUR DET

Figura 7 - Descrizione pin GPIO

### 3.1.3.2 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. Infatti, è rappresentato da un timer decrescente dedicato e separato dall'esecuzione del programma. Esso viene resettato dopo ogni istruzione, mentre allo scadere del tempo di timing ripristina lo stato iniziale del microcontrollore evitando loop infiniti.

Il microcontrollore *RP2040* gestisce gli interrupt su 32 dei suoi pin divisi in banchi, questa funzione è utilizzata negli eventi che richiedono un'attenzione immediata.

IRQ	Interrupt Source								
0	TIMER_IRQ_0	6	XIP_IRQ	12	DMA_IRQ_1	18	SPI0_IRQ	24	I2C1_IRQ
1	TIMER_IRQ_1	7	PIO0_IRQ_0	13	IO_IRQ_BANK0	19	SPI1_IRQ	25	RTC_IRQ
2	TIMER_IRQ_2	8	PIO0_IRQ_1	14	IO_IRQ_QSPI	20	UART0_IRQ		
3	TIMER_IRQ_3	9	PIO1_IRQ_0	15	SIO_IRQ_PROC0	21	UART1_IRQ		
4	PWM_IRQ_WRAP	10	PIO1_IRQ_1	16	SIO_IRQ_PROC1	22	ADC_IRQ_FIFO		
5	USBCTRL_IRQ	11	DMA_IRQ_0	17	CLOCKS_IRQ	23	I2C0_IRQ		

Figura 8 - Gestione Interrupt

Alla ricezione del segnale che scatena l'interrupt, l'esecuzione “istruzione per istruzione” (polling) del programma si interrompe e viene chiamata la routine di servizio di interrupt (ISR - Interrupt Service Routine).

Al termine dell'esecuzione delle istruzioni di interrupt, il microcontrollore riprende le istruzioni che sono state interrotte dalla riga successiva all'evento.

La chiamata alla routine di servizio di interrupt viene eseguita in tre modi:

- Change: il valore logico del pin cambia stato, da alto a basso o da basso a alto;
- Rising: il valore logico del pin varia da basso a alto;
- Falling: il valore logico del pin varia da alto a basso.

I pin I/O del microcontrollore sono programmabili tramite software per definire la loro funzione di input o output.

Un pin impostato come uscita può essere configurato come push-pull, tramite l'utilizzo di un mosfet PMOS e uno NMOS per portare l'uscita rispettivamente alta o bassa (Vcc - GND). Il cosiddetto tri-state è dato dalla possibilità di settare il pin come alta impedenza.

### 3.1.3.3 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).

Il circuito stampato è predisposto per due connettori USB, in modo tale che durante l'assemblaggio sia possibile montare il tipo B o in alternativa quello Mini B in base alla disponibilità dei componenti in laboratorio.

La parte di filtro e stabilizzazione della tensione di alimentazione è composta dall'induttore in serie (L1) e dai due condensatori in parallelo (C3 non polarizzato, C4 polarizzato).

C3 è un condensatore da 100nF con lo scopo di filtrare il ripple, ovvero filtrare le oscillazioni della tensione, mentre C4 è un condensatore da 10uF che viene utilizzato per la conservazione dell'energia. In generale, i condensatori da 100nF vengono utilizzati vicino ai

componenti attivi al fine di rimuovere le alte frequenze e annichilire i disturbi prodotti dal componente stesso.

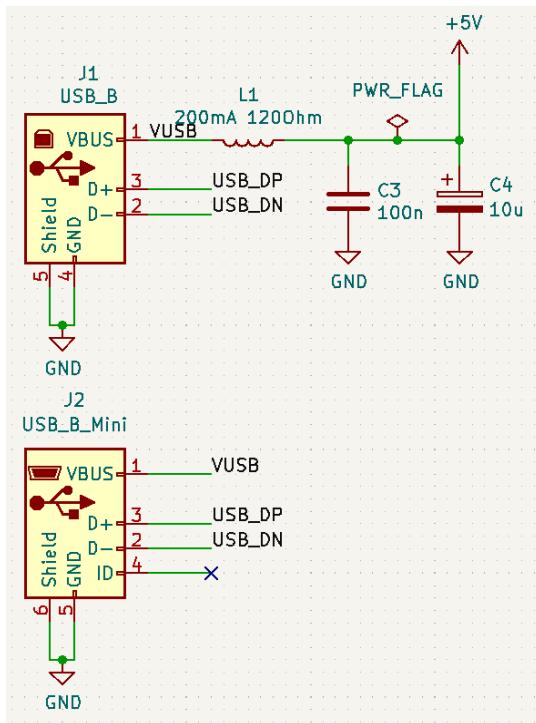


Figura 9 - Circuito USB

In generale, la connessione USB è uno standard di comunicazione seriale dove è garantita sia l'alimentazione tra i due dispositivi sia lo scambio di dati tramite una comunicazione differenziale atta a eliminare i disturbi di modo comune in quanto i segnali D+ e D- sono opposti e il rumore viene eliminato. Il segnale in uscita è dato dalla sottrazione dei segnali differenziali eliminando così i disturbi.

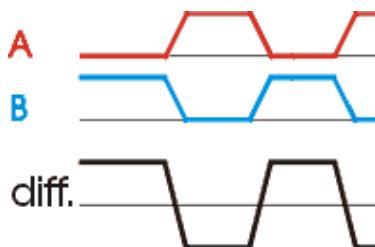


Figura 10 - Segnali della comunicazione USB

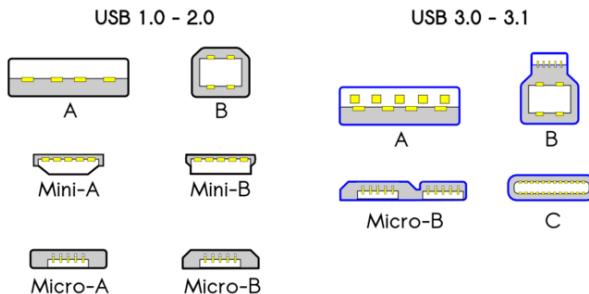
La prima fase successiva alla connessione del dispositivo USB è chiamata enumerazione. In questa fase, tramite un primo dialogo, il device fornisce all'host le informazioni per la connessione come VendorID, ProductID e altri dettagli.

Il trasferimento dei dati avviene tramite pipeline, distinguendosi in diverse modalità:

- Interrupt IRQ: utilizzato se serve immediatezza e reattività;
- Bulk (massivo): utilizzato per il trasporto di una grande quantità di dati;
- Isocrono: è di tipo lossy (send & forget) e può perdere dati. È necessaria la sincronizzazione con le periferiche ma non si tiene in considerazione se il messaggio è stato ricevuto (acknowledge).

Esistono diversi standard USB dal 1.0 al 3.2 che variano in base alla velocità di trasferimento. Nella USB 3.0 sono stati implementati due ulteriori canali differenziali per aumentare la capacità della linea di trasmissione. Tutti i nuovi standard sono retrocompatibili con le vecchie versioni di USB.

Gli standard USB usufruiscono diversi tipi di connettori:



*Figura 11 - Tipologie di connettore USB*

Nel caso del nostro progetto, abbiamo testato il funzionamento dell'USB tramite l'utilizzo dell'oscilloscopio del laboratorio di elettronica del nostro Dipartimento. Si tratta di uno strumento di misura elettronico che consente di visualizzare su un grafico bidimensionale l'andamento nel dominio del tempo dei segnali elettrici. L'oscilloscopio che abbiamo utilizzato è di tipo digitale: il segnale in ingresso viene visualizzato dopo essere stato digitalizzato attraverso la conversione analogico/digitale.

I segnali rilevati dall'oscilloscopio sono visibili in fig. 12 ed è possibile notare come i due canali siano differenziali: quando un bit nel primo canale è al valore alto, il corrispondente bit nel secondo canale è al valore basso.



*Figura 12 - Segnali dell'USB rilevati all'oscilloscopio*

### 3.1.3.4 UART

Il modulo UART (Universal Asynchronous Receiver Transmitter) è una periferica di comunicazione seriale I/O. Contiene i generatori di clock, gli shift registers e i data buffers necessari per il trasferimento dei dati indipendentemente dal programma in esecuzione sul micro.

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. Infatti, la comunicazione seriale avviene su due linee di trasmissione, tra due dispositivi e con velocità di comunicazione che si misura in BAUD=bit/s.

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.

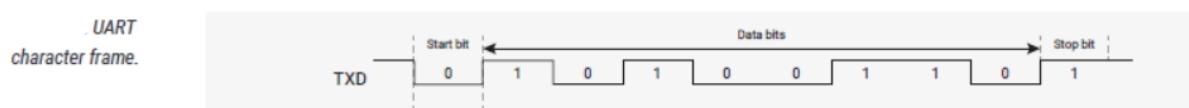


Figura 13 - Funzionamento trasmissione UART

Lo standard UART trasmette e riceve dati tramite il formato standard Non-Return-to-Zero (NRZ), che è implementato tramite due livelli:

- Voltage Output High (VOH) che rappresenta il segnale alto;
- Voltage Output Low (VOL) che rappresenta il segnale basso.

### 3.1.3.5 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.

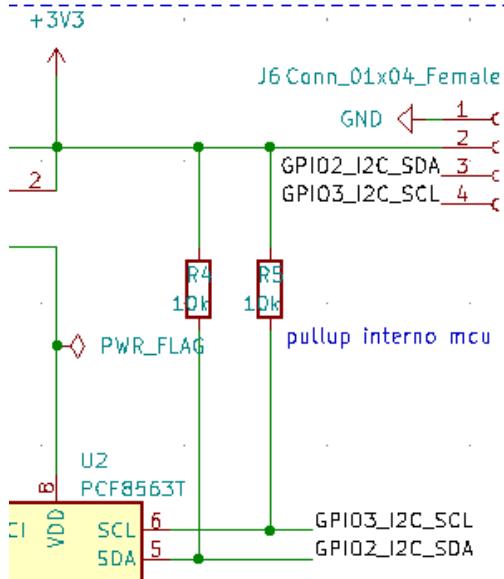


Figura 14 - Resistori di pull up

L'RP2040 implementa sia la modalità I2C master che la modalità slave, i pin utilizzati per il trasferimento dei dati sono:

- Serial clock - SCL;
- Serial data - SDA.

Nella modalità master il microcontrollore emette il segnale di clock ed utilizza gli indirizzi degli slave per la comunicazione, mentre durante il funzionamento come slave, utilizza il pin SCL per sincronizzarsi sul segnale di clock impostato dal master e rimane in attesa dei dati.

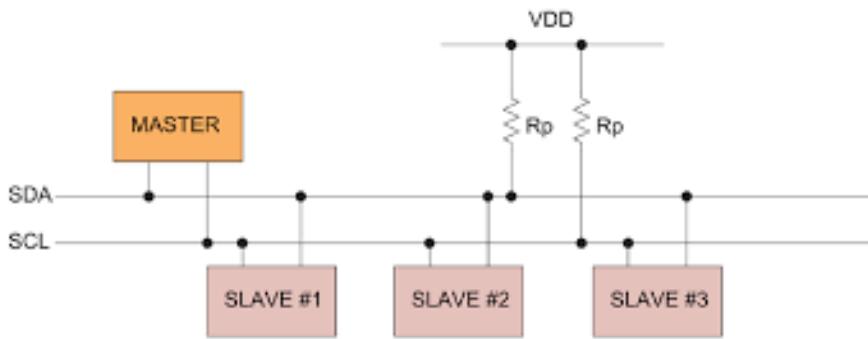
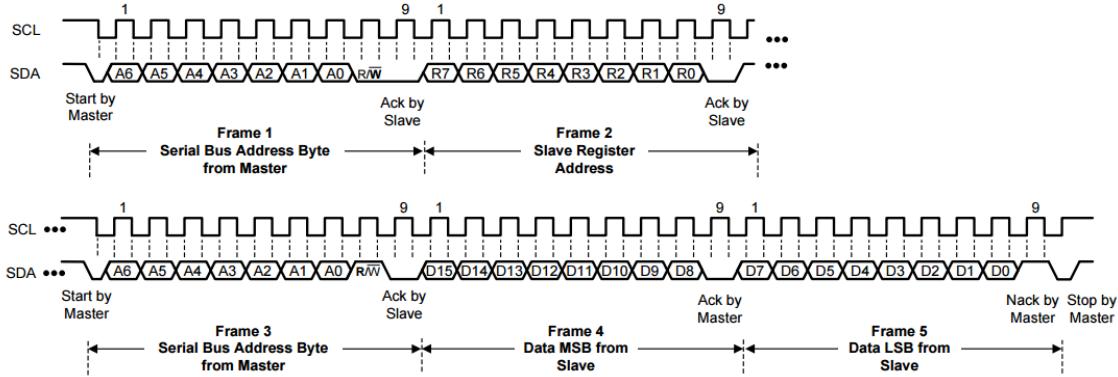


Figura 15 - Comunicazione I2C

La comunicazione inizia dal master, con l'invio del segnale di start, sul canale SDA. Mentre il canale di clock SCL oscilla, il master trasmette i 7 bit dell'indirizzo del dispositivo con cui vuole trasmettere, i quali vengono letti dallo slave a ogni fronte di salita del clock. Successivamente, il master manda un ultimo bit che indica allo slave la volontà di leggere o scrivere e da lì inizia il vero scambio di dati che si concluderà con la trasmissione di un bit di stop da parte del master.



**Figure 17. I2C Read Register Sequence**

*Figura 16 - Protocollo di comunicazione I2C*

La comunicazione tra dispositivi avviene in 4 modi:

1. Il master trasmette – controlla il clock e invia dati agli slave;
2. Il master riceve – controlla il clock, ma riceve dati dallo slave interpellato;
3. Lo slave trasmette – il dispositivo non controlla il clock, ma invia dati al master;
4. Lo slave riceve – il dispositivo non controlla il clock e riceve dati dal master.

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

### 3.1.3.6 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 e garantendo un'esecuzione sequenziale indipendente dallo stato del processore. 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.

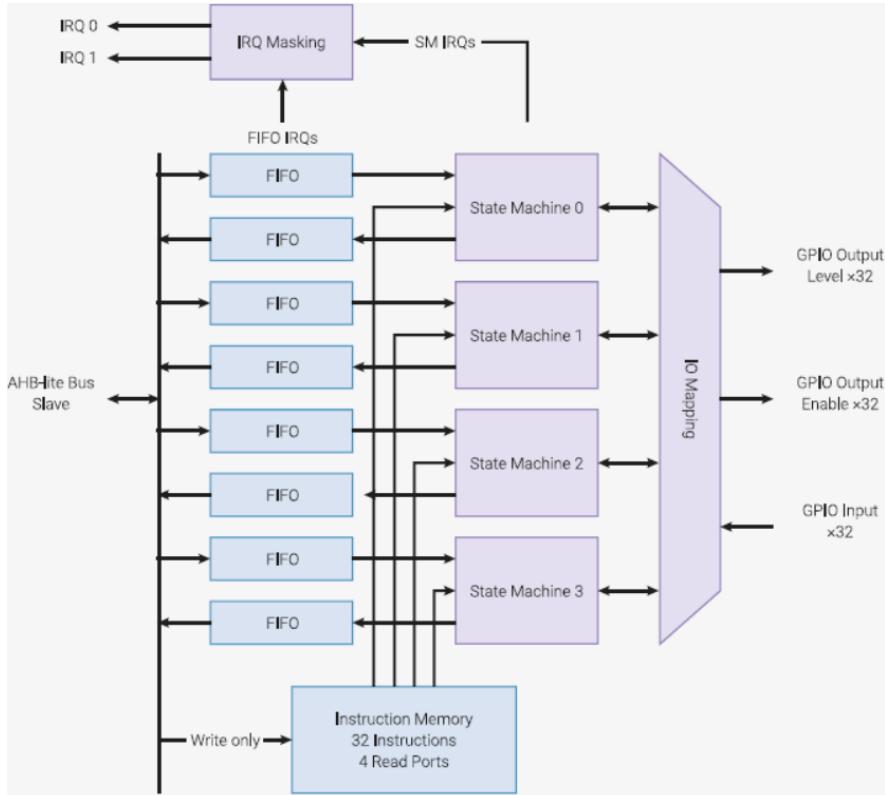


Figura 17 - Architettura PIO

I blocchi PIO sono un interfaccia hardware versatile in grado di supportare diversi standard I/O. In particolare, il datasheet riporta come esempi:

- bus paralleli 8080 e 6800;
- I2C;
- 3-pin I2S;
- SDIO;
- SPI, DSPI, QSPI;
- UART;
- DPI or VGA (via resistor DAC).

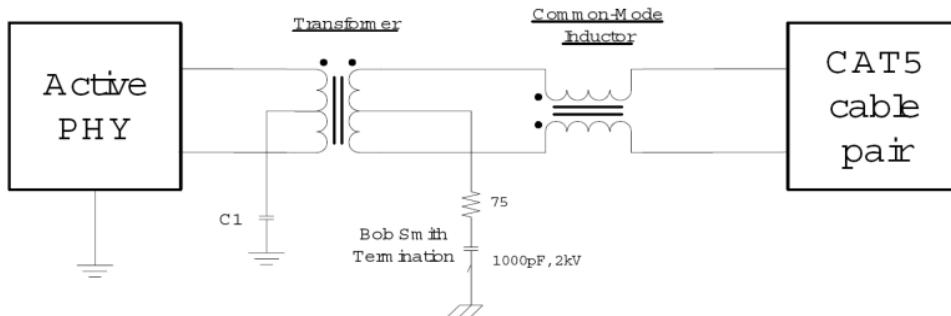
## 3.2 Il circuito LAN

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:

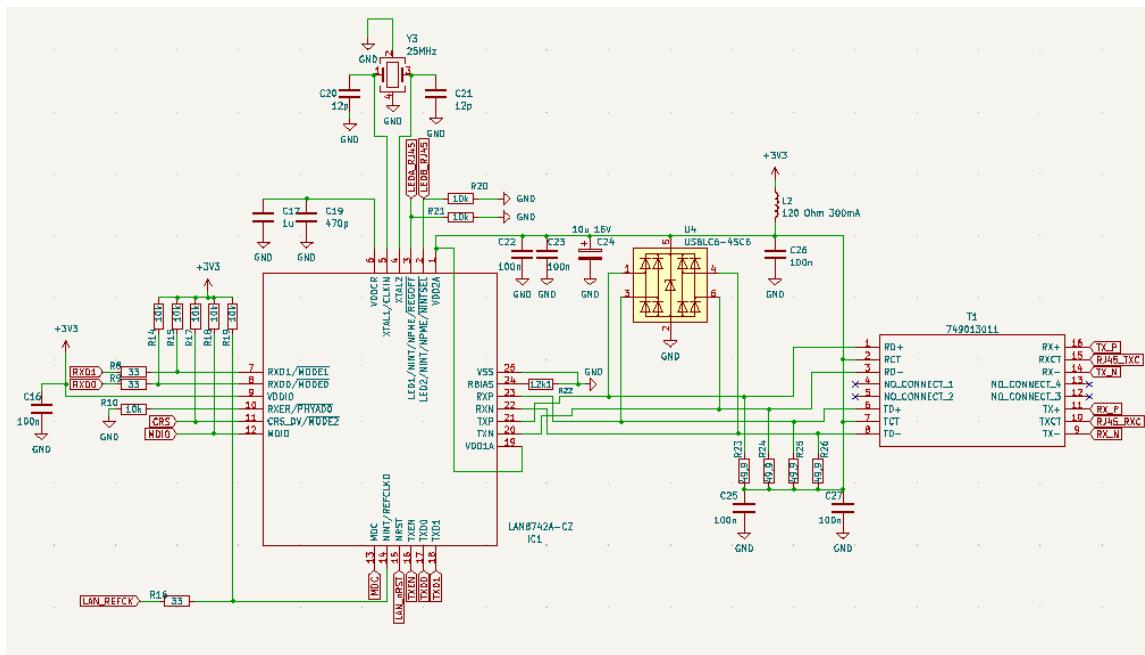
- Connettore RJ45, 8P8C, con led di indicazione dello status Activity e Link;
- Magnetics per standard 10/100BASE-T Wurth Elektronik WE749013011;
- Transceiver LAN8742 o LAN8720.

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.



*Figura 18 - standard 10/100BASE-T*

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



*Figura 19 - Circuito LAN*

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.

### 3.2.1 Transceiver

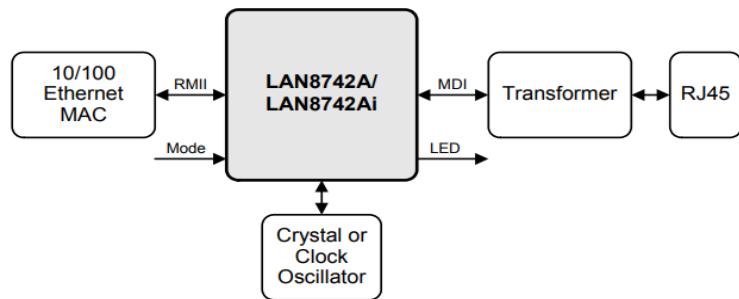


Figura 20 - Funzionamento LAN

Il transceiver utilizzato per il collegamento LAN è il LAN8742A, un low-power 10BASE-T physical layer (PHY) transceiver con tensione di ingresso variabile e conforme agli standard di comunicazione a livello fisico IEEE 802.3 e 802.3u.



Figura 21 - Modulo LAN

10Base-T è una specifica di livello fisico dello standard IEEE 802.3, caratterizzata da velocità di trasmissione di 10 Mbps in banda base su due doppini intrecciati non schermati o tramite Ethernet. Prevede solo collegamenti fisici punto-punto, e quindi consente collegamenti fisici a stella; attraverso dispositivi intermedi, come hub e switch, è possibile ottenere topologie logiche a bus, a stella e ad albero.

Il modulo LAN8742A supporta la comunicazione tramite Ethernet MAC con l'interfaccia standard RMII (Reduced media-independent interface).

Questo modulo è caratterizzato da un basso consumo di potenza e comprende la funzionalità Wake-On-LAN, che garantisce ulteriori modalità di sleep per abbassare i consumi durante le fasi di non utilizzo.

Le sue caratteristiche più rilevanti sono (da datasheet):

- Single-Chip Ethernet PHY
- Flexible power management architecture
- LVCMS variable I/O voltage range +1.6V to +3.6V
- Integrated 1.2V regulator with disable feature
- Cable diagnostics
- Deterministic latency
- Uses a low-cost 25MHz crystal for RMII mode to drive 50MHz output
- Wake-On-LAN
- HP Auto-MDIX support
- Supports commercial (0° to 70°C) and industrial (-40° to 85°C) temperature ranges
- 24-pin, 4x4mm SQFN, RoHS-compliant package

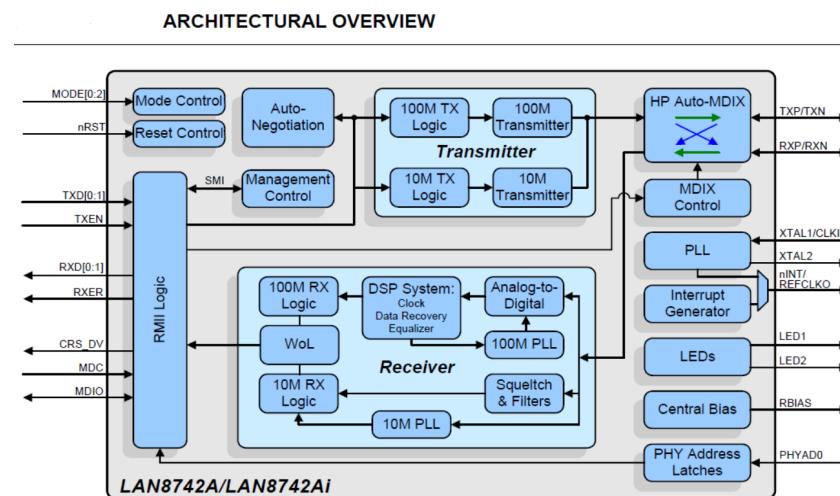


Figura 22 - Overview

### 3.2.2 Trasformatori

I trasformatori di rete LAN sono utilizzati per ridurre i disturbi, includono filtri di modo comune per l'attenuazione del rumore per le applicazioni Ethernet e offrono una tensione di isolamento nell'ordine dei kilo-Volt rms.

Questi trasformatori hanno un rapporto ingresso - uscita di 1:1 per garantire l'isolamento galvanico tra il cavo ethernet e il modulo LAN, mantenendo invariato il segnale.

Per questo scopo si è utilizzato il modulo trasformatori WE749013011 A di Wurth Elektronik  
Caratteristiche (da datasheet):

- Stile di terminazione: SMD/SMT;
- Range di frequenza: 1 MHz to 1000 MHz;
- Induttanza: 350 uH;
- Tensione di isolamento: 1500 VAC;
- Temperatura di lavoro minima: - 40 C;
- Temperatura di lavoro massima: + 105 C;
- Numero di canali: Single;
- Dimensioni: 12.7mm x 9.4mm x 6.35mm;
- Corrente nominale: 350 mA;

Lo schema elettrico è il seguente:

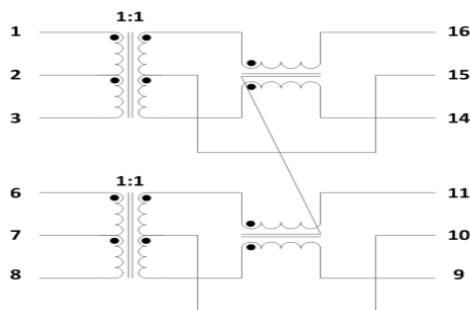


Figura 23 - Schema elettrico trasformatori LAN

### 3.3 Il circuito di 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 il

modello SS16, in package SMA. Esso sopporta una corrente diretta costante pari a 1A, e una tensione inversa massima di 60V.

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:

$$\frac{R2}{R1} = \frac{V_{OUT}}{V_{FB}} - 1$$

I valori di R1 e R2 (R34-R35 in fig. 24) 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:

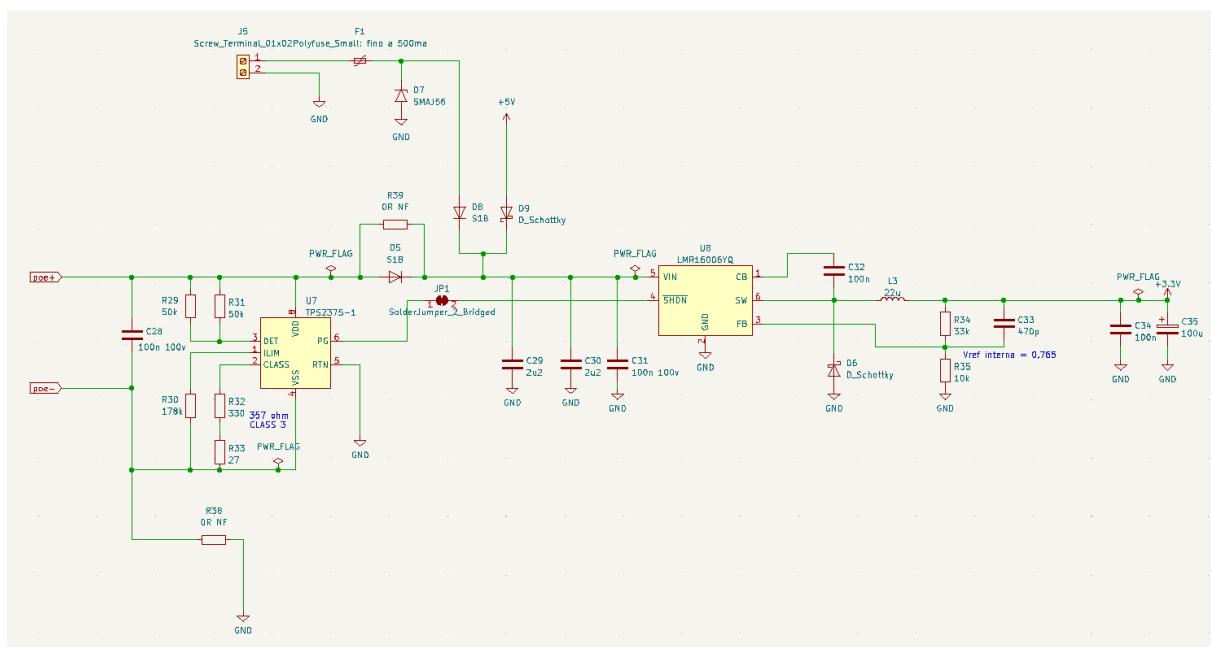


Figura 24 - Circuito di alimentazione e OR a diodi

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 sia di 56 volt, e che il package del componente sia 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 di 357 ohm, ottenuta dalla serie di due resistori, rispettivamente del valore di 330 e 27 ohm.

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

### 3.4 Il modulo GNSS

GNSS è l'acronimo di Global Navigation Satellite System, un sistema per la localizzazione globale. Grazie al GNSS è possibile rilevare le coordinate geografiche del ricevitore, e ottenere dati di navigazione quali, ad esempio:

- Ora UTC
- Data
- Latitudine e longitudine
- Deviazione magnetica della zona
- Direzione rispetto al Nord Vero
- Velocità in nodi
- Numero di satelliti in vista

La geolocalizzazione avviene tramite l'utilizzo di satelliti orbitanti. I moderni ricevitori GNSS riconoscono i segnali provenienti da diverse costellazioni di satelliti. Ne sono un esempio NAVSTAR GPS, GLONASS, BEIDOU, GALILEO.

Il ricevitore GNSS si occupa di ricevere il dato temporale di almeno quattro diversi satelliti, per ottenere il dato di posizione geografica. La conoscenza della posizione in orbita dei satelliti permette, mediante trilaterazione, di ottenere la posizione sulla superficie terrestre.

Lo scopo del GNSS sulla nostra scheda è quello di ricevere l'ora esatta da parte dei satelliti. La ricezione del segnale radio di un singolo satellite da parte del ricevitore è sufficiente per la determinazione dell'orario UTC (Universal Time Coordinated, tempo riferito al meridiano di

Greenwich). Le informazioni di data e posizione richiedono invece l'aggancio di un minimo di quattro satelliti.

Per quanto riguarda il modulo GNSS da impiegare, vengono proposte due alternative, ossia il SAM-M8Q e il PAM-7Q. Le alternative verranno illustrate in modo dettagliato nelle sezioni seguenti.

Nello schematico, si inseriscono i simboli relativi a entrambi i componenti, per permettere l'inserimento dei footprint rispettivi nel layout PCB. Tutti i segnali richiesti vengono connessi tra il microcontrollore ed entrambi i moduli.

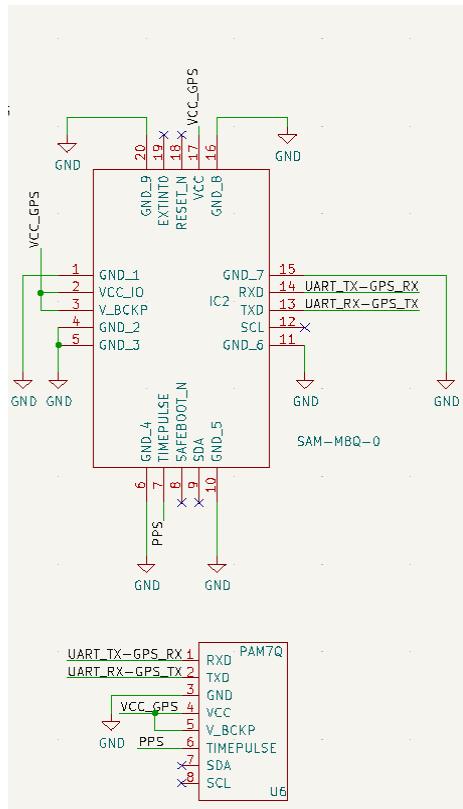


Figura 25 - Schematico localizzazione GNSS

La tensione di alimentazione positiva dei GNSS è fornita tramite un pMOS (NTR4101PT1G) i cui source e drain sono connessi rispettivamente alla linea 3V3 e al pin VCC del modulo GNSS. 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 modulo spento; ponendo a livello basso il segnale di controllo, la tensione S-G positiva fa entrare il transistor in conduzione, e accendere il modulo. Si noti che la RDS-ON del transistor è di pochi milliohm. Si può allora considerare nulla la caduta VSD.

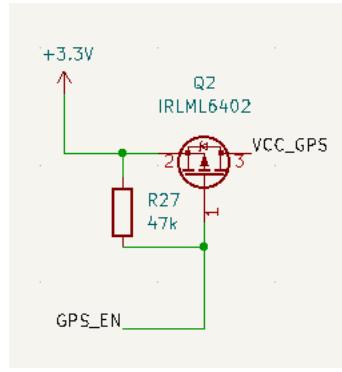


Figura 26 - Transistor PMOS del GNSS

### 3.4.1 SAM-M8Q

Il SAM-M8Q è il modello più recente dei due moduli presi in considerazione e supporta al massimo la ricezione di tre sistemi GNSS (Global Navigation Satellite System) in contemporanea tra GPS, Galileo e GLONASS.

Presenta un' antenna patch integrata ed è in formato SMD. Necessita di un'alimentazione in DC tra i 2.7V e 3.6V tipicamente 3V.



Figura 27 - GPS 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.

Caratteristiche (da datasheet):

- Receiver type: 72-channel u-blox M8 engine GPS L1C/A, SBAS L1C/A, QZSS L1C/A, QZSS L1 SAIF, GLONASS L1OF, Galileo E1B/C;
- Accuracy of time pulse signal

- RMS: 30 ns;
  - 99%: 60 ns;
- Frequency of time pulse signal: 0.25 Hz...10 MHz (configurable);
- Operational limits:
  - Dynamics <= 4 g;
  - Altitude 50,000 m;
  - Velocity 500 m/s;
- Velocity accuracy: 0.05 m/s;
- Heading accuracy: 0.3 degrees;
- GPS, GLONASS, Galileo;
- Horizontal position accuracy:
  - GPS & GLONASS 3 2.5 m;
  - GPS 2.5 m;
  - GLONASS 8.0 m;
  - Galileo TBC 4;
- Max navigation update rate:
  - GPS & GLONASS 10 Hz;
  - GPS 18 Hz;
  - GLONASS 18 Hz;
  - Galileo 18 Hz;
- Time-To-First-Fix (Cold start):
  - GPS & GLONASS 26 s;
  - GPS 29 s;
  - GLONASS 30 s;
  - Galileo 18 Hz TBC 4;

## Block diagram

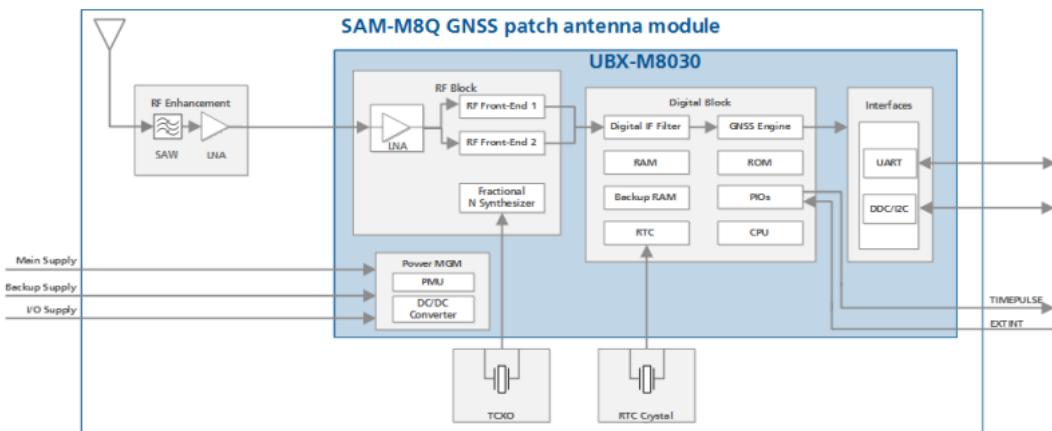


Figura 28 - Diagramma SAM-M8Q

### 3.4.2 PAM-7Q

Durante la prototipazione si è scelto di utilizzare il modello PAM-7Q, a montaggio through hole e provvisto di un'antenna patch 18 x 18 mm con polarizzazione RHCP. Questo modulo utilizza solo il sistema GPS di GNSS e per il corretto funzionamento necessita di un'alimentazione in DC tra i 2.7V e 3.6V tipicamente 3V.

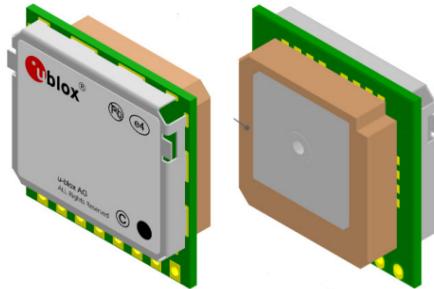


Figura 29 - GPS PAM-7Q

Il PAM-7Q richiede che  $V_{CC}$  e  $V_{BCKP}$  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.

Caratteristiche (da datasheet):

- Receiver type:
  - 56 Channels;
  - GPS L1C/A;
  - SBAS L1C/A;
  - QZSS L1C/A;
- Time-To-First-Fix:
  - Cold Start 29 s;
  - Warm Start 28 s;
  - Hot Start 1 s;
  - Aided Starts 5 s;
- Sensitivity:
  - Tracking & Navigation –161 dBm;
  - Reacquisition –159 dBm;
  - Cold Start –147 dBm;
  - Warm Start –147 dBm;
  - Hot Start –155 dBm;
- Horizontal position accuracy:
  - Autonomous 2.5 m;
  - SBAS 2.0 m;
- Accuracy of time pulse signal:
  - RMS 30 ns;
  - 99% 60 ns;
- Frequency of time pulse signal: 0.25 Hz ... 1 kHz (configurable);
- Max navigation update rate: 10 Hz;
- Velocity accuracy: 0.1 m/s;

- Heading accuracy: 0.5 degrees;
- Operational limits:
  - Dynamics  $\leq$  4 g;
  - Altitude 50,000 m;
  - Velocity 500 m/s;

### Block diagram

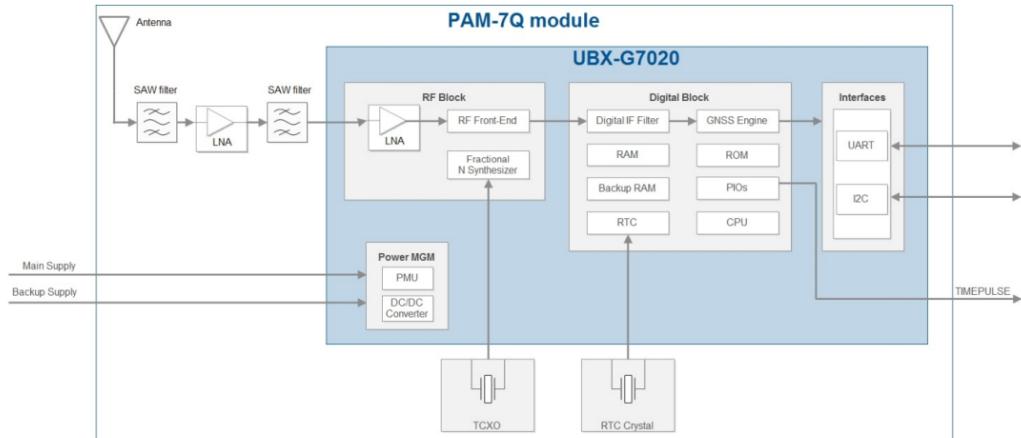


Figura 30 - Diagramma PAM-7Q

## 3.5 Il modulo RTC

Un orologio in tempo reale, noto anche come RTC per il suo acronimo in inglese "Real Time Clock", è un circuito integrato con lo scopo di scandire il tempo con uno scarto infinitesimale, utilizzato spesso nei telefoni cellulari, dispositivi portatili, misurazioni elettronici e prodotti alimentati tramite batteria. E' in grado di contare secondi, minuti, ore, giorni, mesi, anni, e fornisce inoltre un flag del secolo.

Il modello previsto è il Microchip MCP7940M in package "SOIC 8-Lead Plastic Small Outline (SO) - Wide 3.90mm Body" avente un pitch di 1.27 mm BSC (Basic Spacing between Centers, valore teoricamente esatto espresso senza tolleranze), con una memoria SRAM da 64 byte e funzioni di allarme e timer.



Figura 31 - Modulo RTC MCP7940M

La comunicazione con il modulo RTC MCP7940 richiede l'impiego di una interfaccia I2C, con velocità massima del bus di 400 kbit/s. Al fine di prevenire situazioni di stallo, il

dispositivo prevede un watchdog timer che ripulisce l'interfaccia I2C per permettere di continuare a tenere traccia del tempo in caso di stalli.

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 pull-up interna. Per il protocollo, una resistenza di pull-up 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. Il diodo D1, infatti, carica il condensatore da 0.1F quando c'è alimentazione mentre quando non c'è alimentazione impedisce che il condensatore si scarichi sul resto del circuito, permettendo di alimentare solamente il modulo RTC.

Per il suo regolare funzionamento, il modulo richiede la connessione dei pin OSC1 e OSC0 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 1s, partendo da una base di 32.768 kHz.

Il segnale CLKO (pin 7) del modulo RTC indica il clock in uscita, ovvero l'onda quadra programmabile disponibile in uscita dall'RTC. È un circuito open-drain e viene abilitato all'accensione e nel caso venga disabilitato diventa in alta impedenza. La frequenza di default è 32.768kHz, ma possono essere generate anche forme d'onda a 1.024kHz, 32Hz e 1Hz per l'utilizzo come system clock, clock del microcontrollore oppure per la calibrazione dell'oscillatore.

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

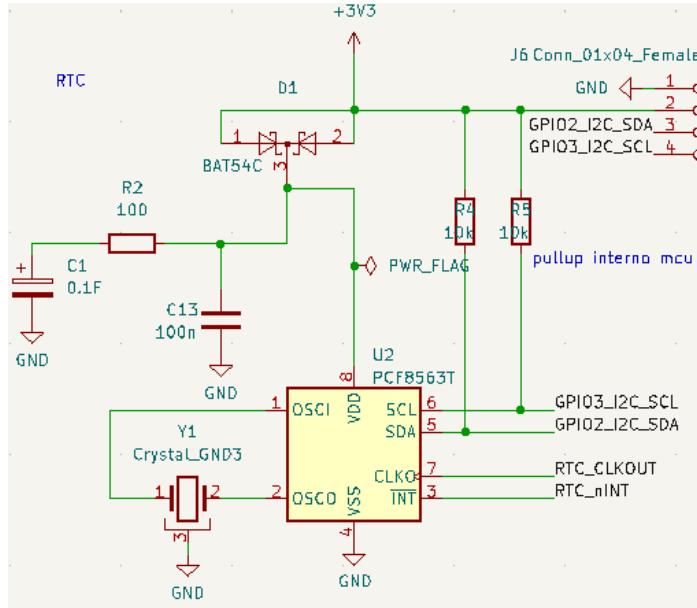


Figura 32 - Circuito RTC

Caratteristiche (da datasheet):

- Timekeeping Features
  - Real-Time Clock/Calendar (RTCC):
    - Hours, Minutes, Seconds, Day of Week, Day, Month, Year;
    - Leap year compensated to 2399;
    - 12/24 hour modes;
  - Oscillator for 32.768 kHz Crystals:
    - Optimized for 6-9 pF crystals;
  - On-Chip Digital Trimming/Calibration:
    - $\pm 1$  PPM resolution;
    - $\pm 129$  PPM range;
  - Dual Programmable Alarms;
  - Versatile Output Pin:
    - Clock output with selectable frequency;
    - Alarm output;
    - General purpose output;
- Low-Power Features
  - Wide Voltage Range:
    - Operating voltage range of 1.8V to 5.5V;
  - Low Typical Timekeeping Current:
    - Operating from VCC: 1.2  $\mu$ A at 3.3V;
- User Memory
  - 64-byte SRAM;
- Operating Ranges
  - 2-Wire Serial Interface, I2C Compatible
    - I2C clock rate up to 400 kHz;
  - Temperature Range:
    - Industrial (I):  $-40^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$ ;

## 3.6 Test Point

Vengono inseriti i test point sull'alimentazione e sui segnali più importanti da monitorare, questo è utile durante lo svolgimento dei test funzionali per controllare il corretto funzionamento della scheda. Si predisponde il controllo per la tensione di alimentazione del GPS e dei suoi segnali di ricezione e trasmissione.

Anche il modulo RTC presenta due test point sui segnali SDA e SCL della comunicazione I2C.

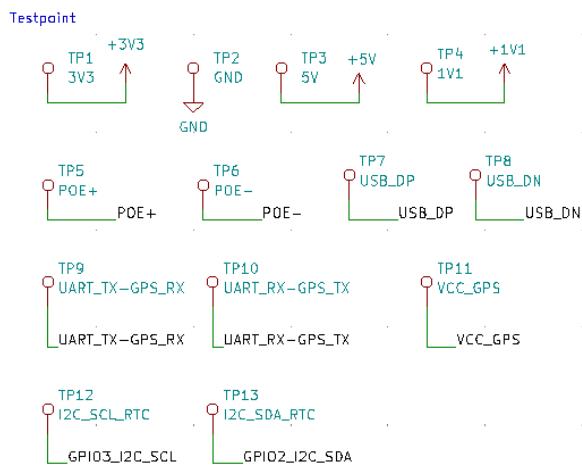


Figura 33 - Testpoint nello schematico

## 3.7 Uscite digitali

Il LED alive è utile per la verifica della corretta esecuzione del programma in fase di debug. Dopo una corretta programmazione è possibile osservare se l'esecuzione delle istruzioni è rimasta bloccata in qualche ciclo infinito oppure le istruzioni sono eseguite correttamente tramite un semplice lampeggio del led. Il led rosso è collegato al pin GPIO25, pin 37 del microcontrollore.

Esso è alimentato a 3V3 ed è connesso in serie ad una resistenza dal valore di 330 Ohm. La connessione in sink (come da schema seguente) sfrutta la conduzione dello stadio N del pin GPIO, data la sua maggiore conducibilità rispetto allo stadio P. Questa connessione fa lavorare il LED in logica negata (uscita alta per led spento, uscita bassa per led acceso).

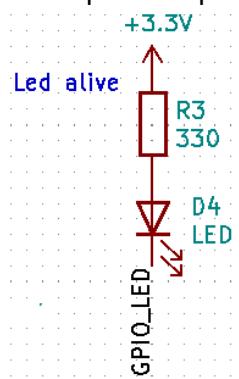


Figura 34 - Circuito LED di alive nello schematico

**NB:** In base al colore del LED di segnalazione, la tensione di forward cambia. Tipicamente si ha una caduta di tensione di 1.8V per il LED rosso e 2.5V per il LED verde, come è possibile vedere nel grafico in figura seguente.

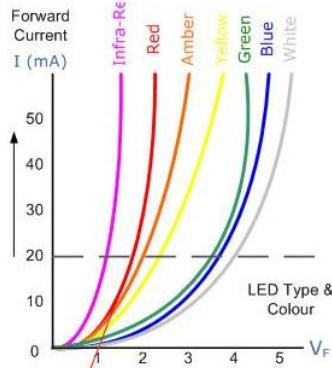


Figura 35 - Tensione di forward dei LED

### 3.8 Ingressi digitali

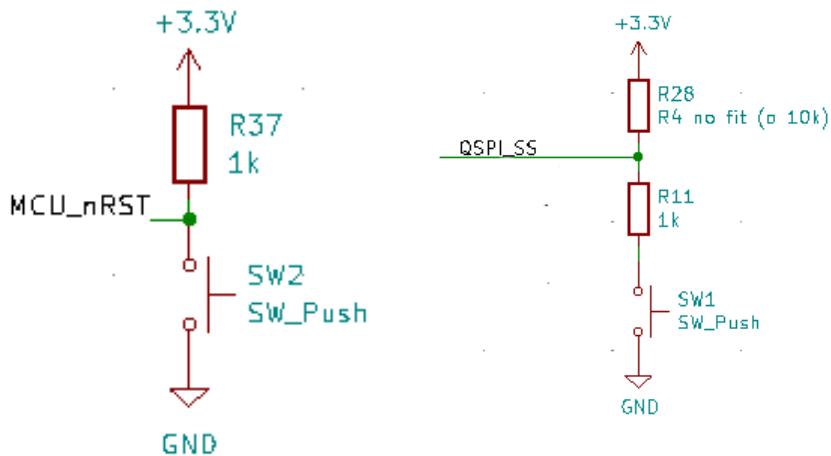


Figura 36 - Pulsante di reset

Figura 37 - Pulsante di boot

Il segnale MCU\_nRST del microcontrollore è connesso a VDD mediante resistore da 1kOhm, e a GND mediante un pulsante NA. La chiusura del pulsante verso massa causa il reset del microcontrollore. Il secondo pulsante è connesso invece al segnale CS della memoria FLASH; lasciando flottante tale segnale (quindi senza la pressione del tasto), il microcontrollore è in grado di gestire la comunicazione con tale integrato per le operazioni di lettura e scrittura. La chiusura a massa del contatto (attraverso un resistore del valore di 1kOhm), all'accensione del microcontrollore, forza l'avvio del microcontrollore stesso in modalità bootloader, permettendo il caricamento del firmware mediante connessione USB. Il resistore di pullup (R28 nello schema) non viene invece montato.

Non sono previsti condensatori in parallelo che hanno la funzione di debouncing durante la variazione di stato dei due pulsanti, in quanto i pulsanti hanno lo scopo di resettare allo stato iniziale il microcontrollore e di avviarlo in modalità di programmazione della memoria flash.

Il condensatore in shunt si predispone quando il pulsante ha lo scopo di ingresso digitale generico in quanto il segnale si considera alto se è maggiore di  $\frac{2}{3}$  del range di tensione massima, mentre è considerato basso quando è pari o inferiore ad  $\frac{1}{3}$  della tensione di alimentazione.

Tramite l'utilizzo del condensatore di debouncing si raggiunge la soglia dei  $\frac{2}{3}$  della tensione in un tempo maggiore rispetto al tempo durante il quale il contatto rimbalza.

Questo tempo è calcolato tramite la formula di carica del condensatore:

$$V(t) = Val * [1 - e^{(-t/\tau)}] \text{ con } \tau = R * C$$

Durante il passaggio da alto a basso il condensatore si scarica in un tempo infinitesimo in quanto il pulsante premuto lo cortocircuita direttamente a massa.

Un altro modo per risolvere questo problema è tramite uno stratagemma software che valuta il cambio di stato solo una volta terminate le oscillazioni 0-1/1-0 dell'ingresso.

# 4. PROGETTAZIONE CAD PCB LAYOUT

Nei paragrafi seguenti tratteremo le fasi di progettazione del PCB, dalla definizione delle regole di progettazione all'ottimizzazione del risultato finale

## 4.1 Assegnamento dei footprint

Innanzitutto, si assegnano i footprint a tutti i componenti del PCB. I footprint sono le impronte dei componenti elettronici sui circuiti stampati. Sono formati dal profilo geometrico dei pin per le interconnessioni elettriche tra componente e piste del PCB, dalla serigrafia del perimetro del componente, dal profilo dei fori, dal campo di testo per il riferimento del componente e da ulteriori indicazioni testuali.

In questo progetto i footprint sono assegnati come in allegato (punto 9.1) ed in base alle specifiche definite al punto 2.

### 4.1.1 Creazione nuovo footprint

Il footprint del GPS PAM7Q è stato realizzato manualmente e salvato in una libreria appositamente creata per il progetto, in quanto non era disponibile né nelle librerie preinstallate di Kicad, né disponibile al download.

Il componente include 8 fori e due pad SMD utilizzati per il GND, come è possibile vedere in figura.

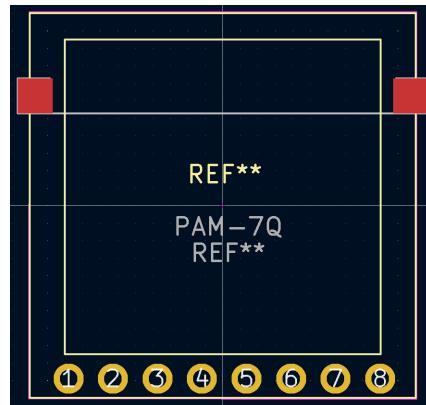


Figura 38 - Footprint realizzato manualmente del PAM-7Q

Le proprietà delle pad sono state assegnate come nelle figure seguenti.

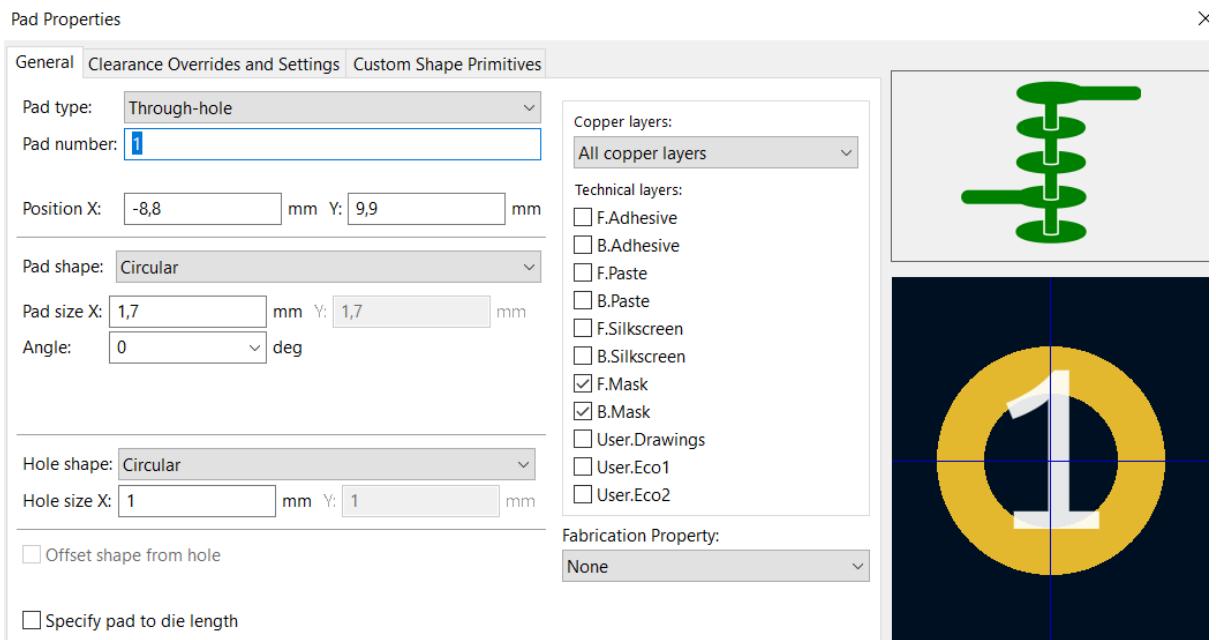


Figura 39 - Proprietà pad through-hole

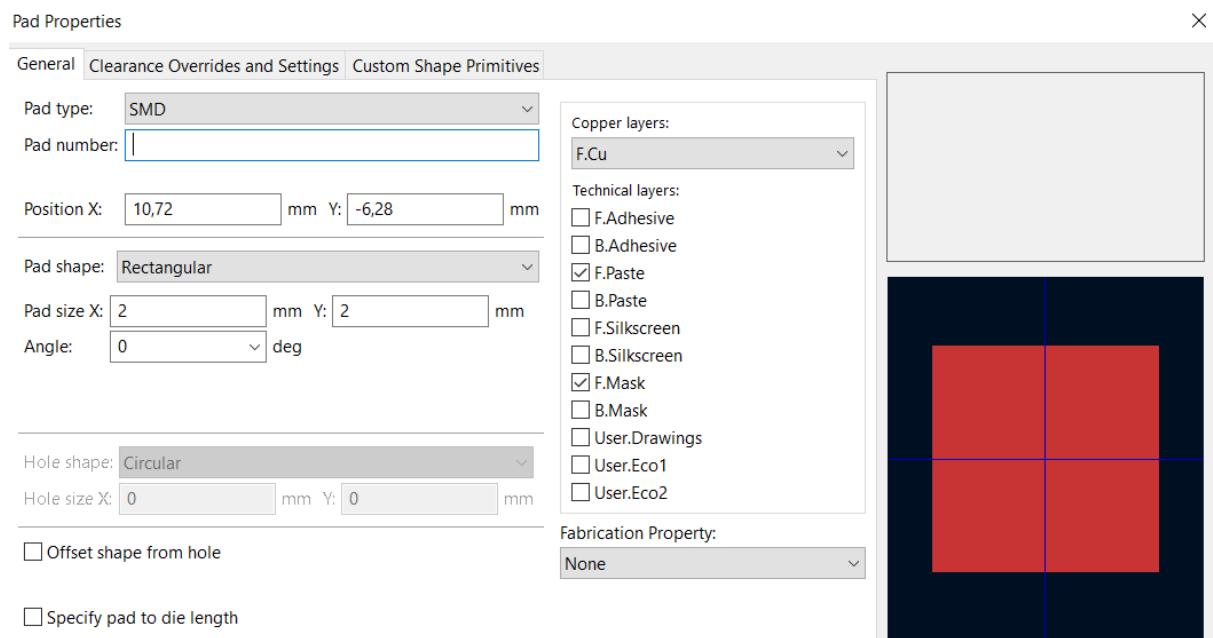


Figura 40 - Proprietà pad SMD

## 4.2 Definizione delle regole di progettazione

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 da due strati (layer, Top e Bottom), e avrà spessore del substrato (FR4) pari a 1.6mm.

Le regole di progettazione della scheda state impostate come segue:

Impostazione scheda

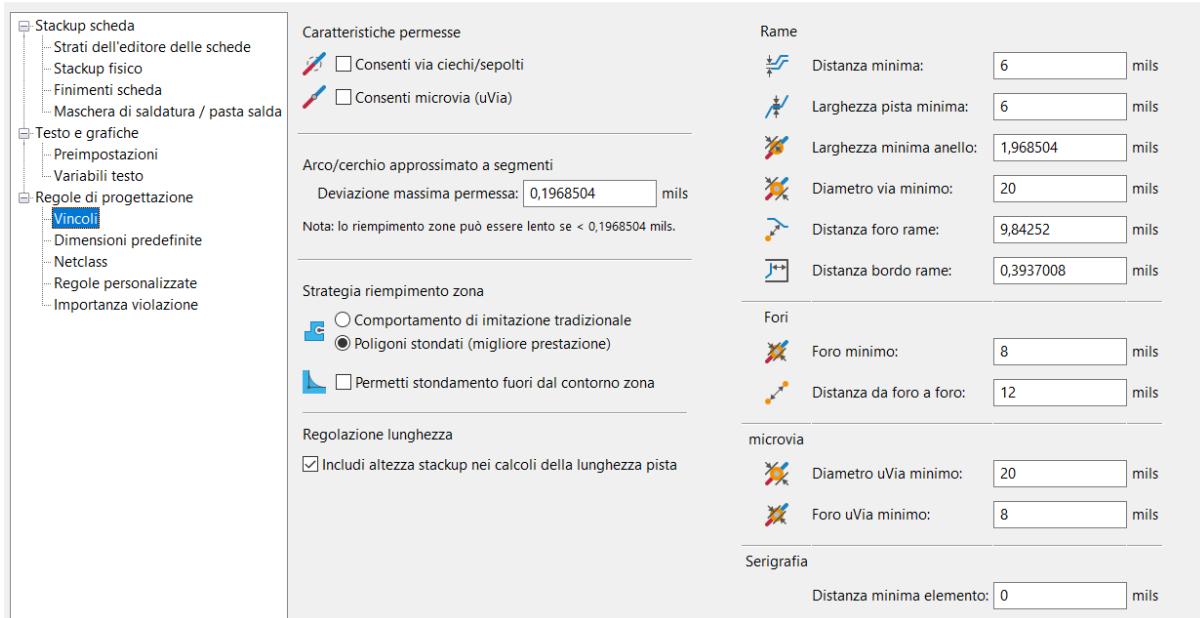


Figura 41 - Regole di progettazione (clearance, dimensioni fori...)

Impostazione scheda

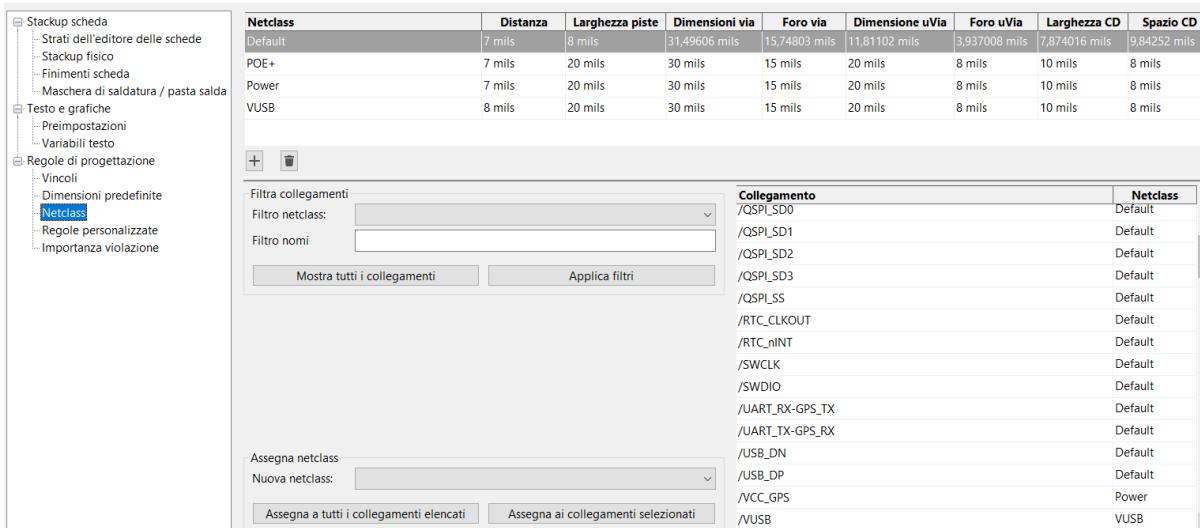


Figura 42 - Definizione e assegnamento delle netclass

## 4.3 Progettazione del layout della scheda

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.

Le dimensioni della scheda sono quindi: 85mm x 56mm x 1.6mm.

In seguito, si importa la 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 RJ45 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 pull-up/pull-down, diodo led, diodo di alimentazione RTC, pMOS di spegnimento del GPS) posti di conseguenza relativamente alla loro funzione, il più vicini possibile al componente corrispondente.

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.

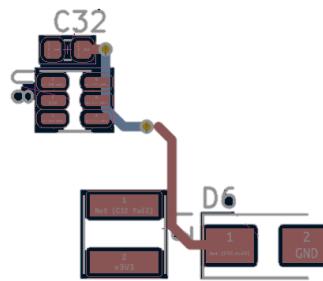


Figura 43 - Posizionamento non ottimale dei condensatori di disaccoppiamento e del diodo di ricircolo del regolatore switching

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.

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 sbroglino, 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.

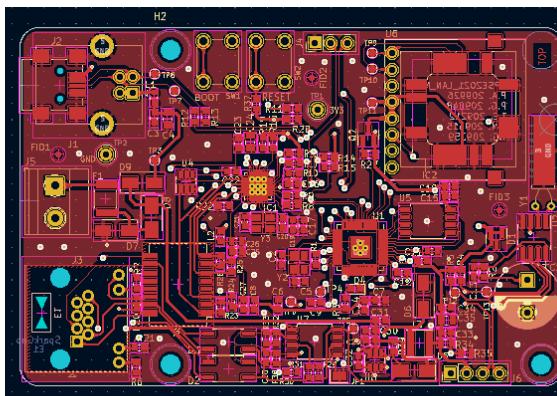


Figura 44 - Area di rame layer TOP

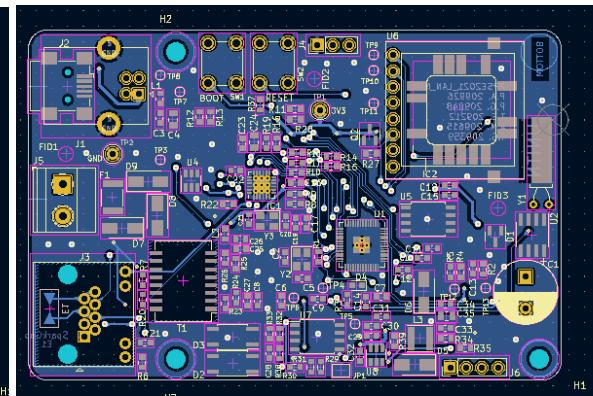


Figura 45 - Area di rame layer BOTTOM

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, non generino problemi quali falsi contatti, dovuti ad esempio a colatura di stagno fuso che causino cortocircuiti tra i pin del componente.

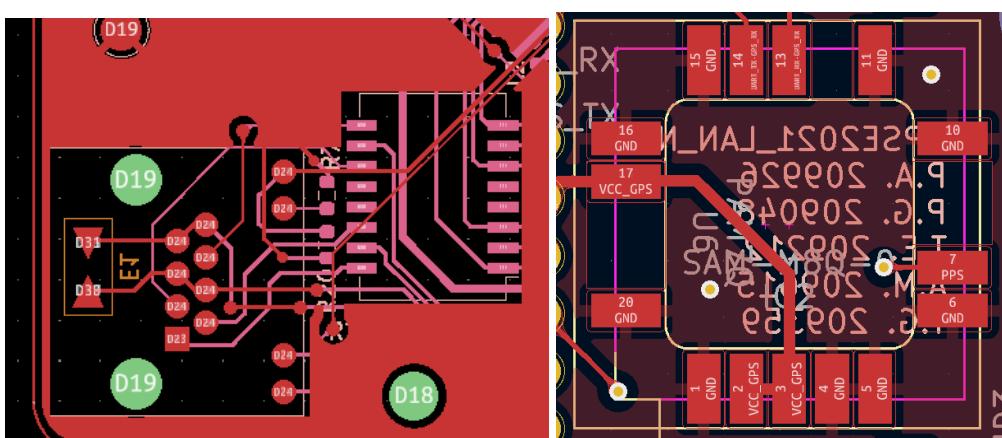


Figure 46-47 - Zone no-fill connettore RJ45, magnetics e SAM-M8Q

Al fine di una revisione veloce ed efficace, sul layer F.Silkscreen sono stati riportati il numero di errori e di avvisi rilevati dalla verifica DRC e l'indicazione dei testpoint con il corrispondente segnale. In questo modo è facile verificare la presenza di errori aggiuntivi non ancora considerati in fase di ottimizzazione del PCB stesso.

```

ERRORI: 11
AVVISI: 8

TESTPOINT
TP1 3V3
TP2 GND
TP3 5V
TP4 1V1
TP5 POE+
TP6 POE-
TP7 USB_DP
TP8 USB_DN
TP9 UART_TX-GPS_RX
TP10 UART_RX-GPS_TX
TP11 VCC_GPS
TP12 GPIO3_I2C_SCL
TP13 GPIO2_I2C_SDA

```

Figura 48 - Commenti aggiuntivi sulla serigrafia

## 4.4 Sboglio e ottimizzazione

Si procede quindi con lo sboglio della scheda tracciando le piste. Si utilizzano le *vias* per passare dal layer top al layer bottom e viceversa, nel caso in cui si incontri un'altra pista oppure un componente, e continuare a tracciare la pista fino alla pad desiderata.

Per quanto riguarda l'ottimizzazione del PCB, si è proceduto allo spostamento delle piste in modo da ottenere i piani di massa il più ampi possibili e le tracce il più corte possibili al fine di ridurre eventuali disturbi che potrebbero interferire con il corretto funzionamento della scheda.

Ad esempio, nel caso riportato in figura, è presente una zona non coperta dall'area di rame. Si può procedere, quindi, spostando la pista di qualche millimetro in modo da riuscire a riempire anche la zona precedentemente vuota.

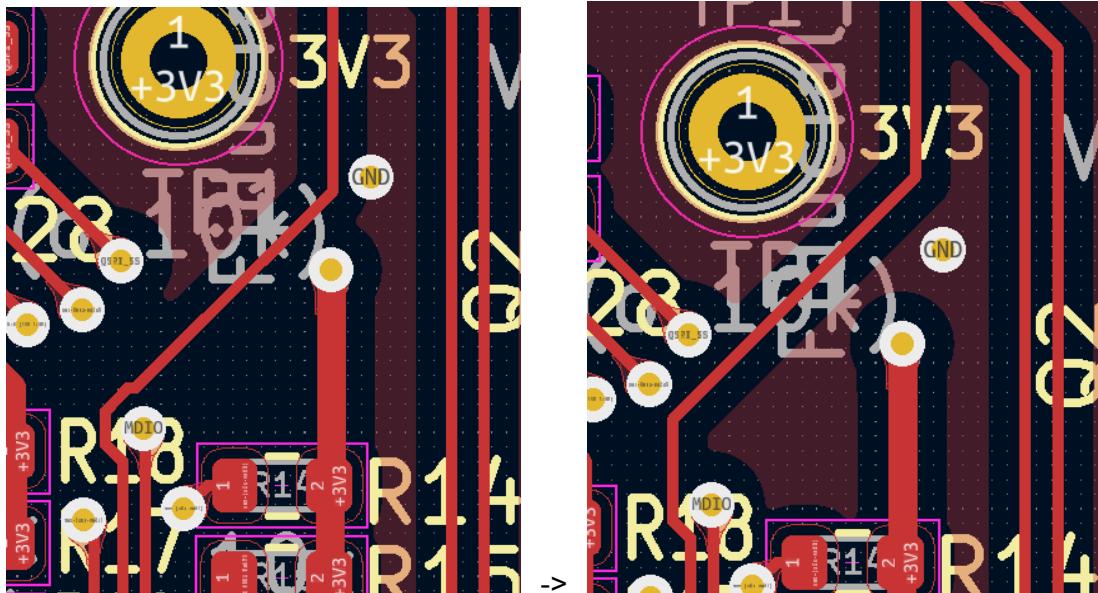


Figure 49-50 - Ottimizzazione delle aree di rame

Inoltre, sono stati aggiunti le teardrop utilizzando l'apposito plugin e i fiducial. Questi ultimi risultano particolarmente utili per il posizionamento del telaio per la stesura della pasta saldante e per le macchine di pick&place. Ne sono stati posizionati 3 in modo asimmetrico.

La scheda completamente sbagliata è visibile in figura 51. Le piste rosse si trovano sul layer top, mentre quelle blu sul layer bottom.

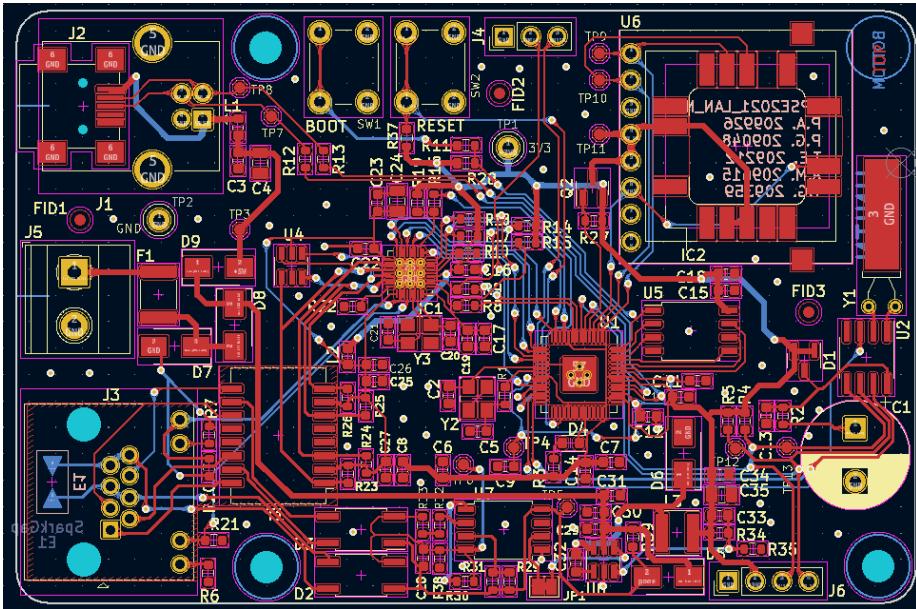


Figura 51 - Scheda completamente sbagliata

A questo punto se il DRC non rileva nessun errore, oppure rileva errori che si possono ignorare, si può passare alla generazione dei file Gerber.

Un esempio di errore trascurabile appena citato potrebbe verificarsi nel caso in cui vengano previsti due componenti da montare in alternativa e i footprint vengano posizionati sovrapposti. Il controllo DRC rileverà, quindi, che i due ingombri si sovrappongono e genererà un errore, da noi ignorabile.

## 4.5 I file Gerber

I file Gerber vengono generati al termine dello sbagliato. Sono scritti in linguaggio testuale, editabile con un qualsiasi editor di testo, e definiscono principalmente la grafica, le geometrie e le lavorazioni necessarie al fine della realizzazione del PCB.

La visualizzazione separata dei piani Gerber, mediante *GerbView*, permette di verificare che tutti gli elementi del layout siano corretti ai fini della produzione.

I file Gerber generati sono di seguito elencati. La visualizzazione dei Gerber è riportata nella sezione Allegati (9.2) ed in tutti è visibile il file Edge\_Cuts (GKO), rappresentante il contorno della scheda.

- F\_Cu (GTL): rappresenta le aree di rame al lato top;
- B\_Cu (GBL): rappresenta le aree di rame al lato bottom;
- F\_Mask (GTS): rappresenta la soldermask al lato top;
- B\_Mask (GBS) rappresenta la soldermask al lato bottom;
- F\_Silkscreen (GTO): rappresenta la serigrafia al lato top;
- B\_Silkscreen (GBO): rappresenta la serigrafia al lato bottom;
- F\_Paste (GTP): rappresenta le aree della pasta saldante al lato top. Il Gerber B\_Paste non viene generato in quanto non sono presenti componenti sul lato bottom e di conseguenza non è necessaria la pasta saldante.

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.

Infine, viene generato anche il file Drill, ovvero il file delle forature, il quale contiene le indicazioni sulla punta da utilizzare per effettuare la foratura e la posizione (x,y) del foro, come si può vedere in figura 52 .

```
T1
X5.3622Y-4.9528
X5.3622Y-4.9921    INCH
X5.3622Y-5.0315    ; #@! TA.AperFunction,Plated,PTH,ComponentDrill
X5.4016Y-4.9528    T1C0.0079
X5.4016Y-4.9921    ; #@! TA.AperFunction,Plated,PTH,ComponentDrill
X5.4016Y-5.0315    T2C0.0138
X5.4409Y-4.9528    ; #@! TA.AperFunction,Plated,PTH,ViaDrill
X5.4409Y-4.9921    T3C0.0150
X5.4409Y-5.0315    ; #@! TA.AperFunction,Plated,PTH,ViaDrill
```

*Figura 52 - File NC Drill*

## 4.6 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.

## 5. 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 (detto serigrafo). 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.

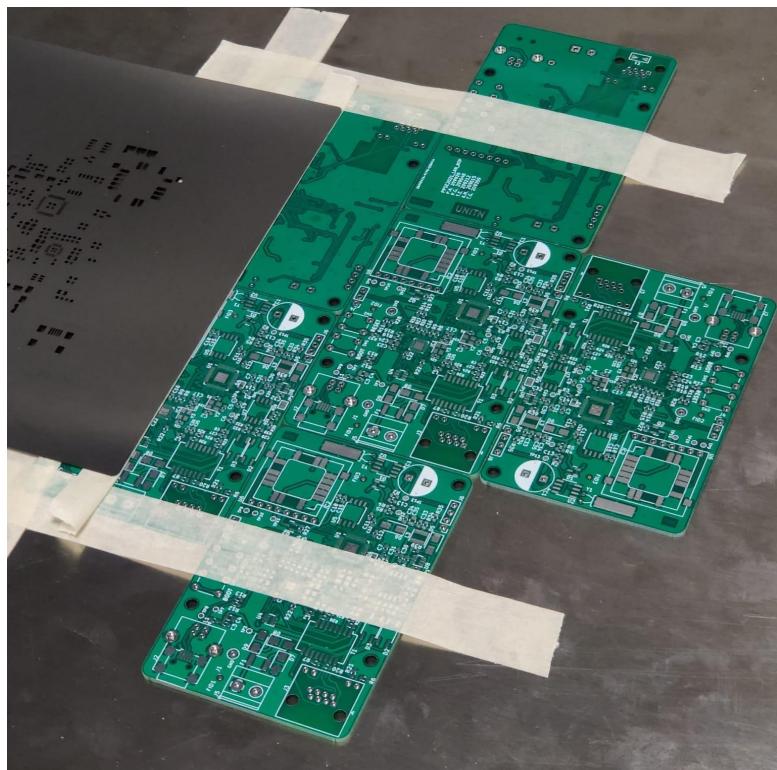
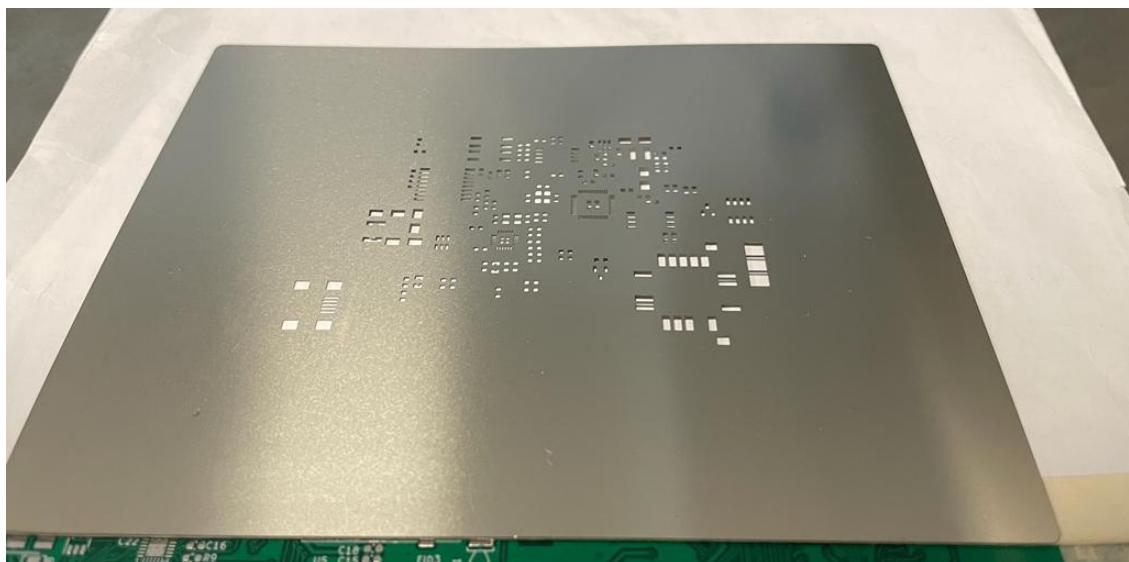
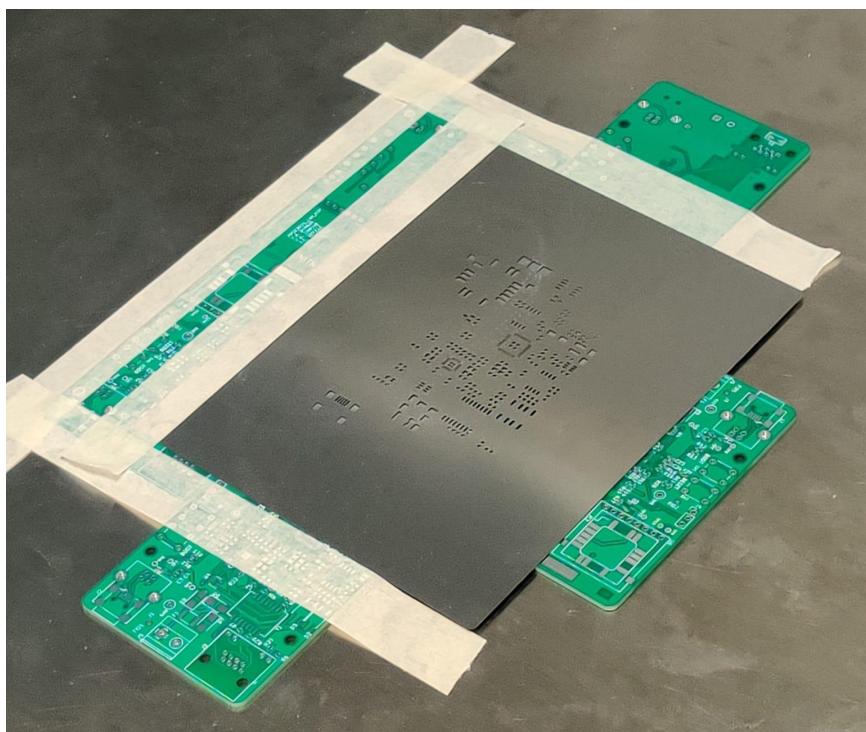


Figura 53 - Fissaggio serigrafo "fai da te"

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.



*Figura 54 - Posizionamento lamina*



*Figura 55 - Fissaggio lamina*

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.



Figura 56 - Stesura pasta saldante

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, consultando lo schematico e la BOM.

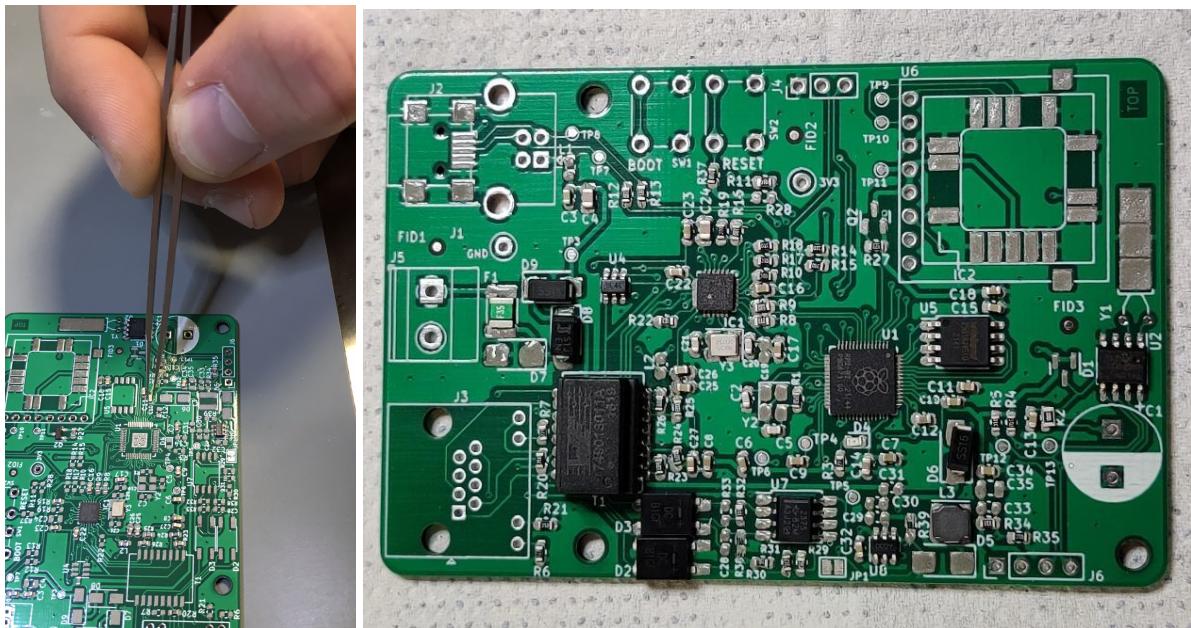


Figure 57-58 - Posizionamento componenti sulla pasta saldante e risultato finale

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).

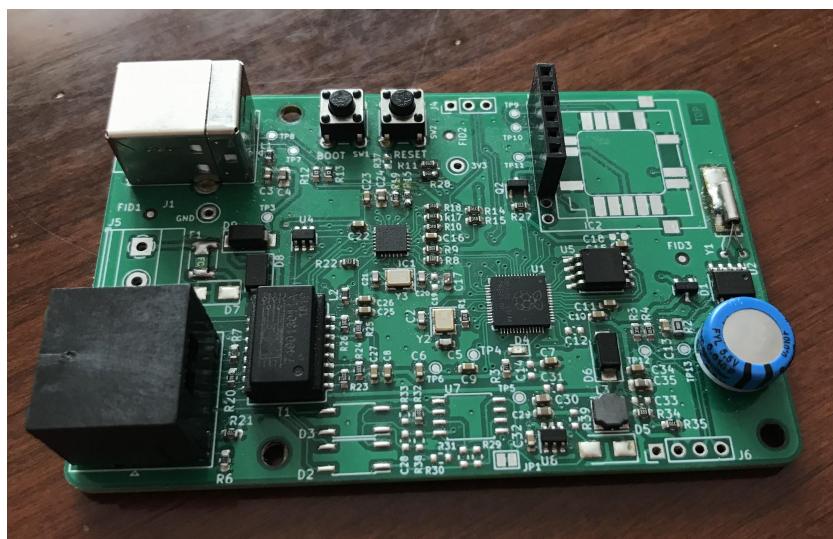
Può verificarsi che un componente non sia esattamente allineato alla sua impronta. Nella maggior parte dei casi questo inconveniente si risolve grazie alla tensione superficiale dello stagno liquefatto, che attua una forza sui pin del componente che lo allinea alle pad, prima di

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 due 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 lega.

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.



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  $33\Omega$  a corredo del transceiver LAN (R8, R9, R16). Sulle restanti quattro schede, tali resistori erano presenti, seppur nel valore di  $49.9\Omega$ . La sostituzione di R16 con il valore di  $270\Omega$  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 la regolare esecuzione 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 realizzazione 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'accurata analisi dello schematico e del codice, si è riscontrato che il segnale di reset del transceiver, pur connesso al microcontrollore, non era gestito dal relativo pin GPIO. Si sottolinea che il segnale di reset è attivo a livello basso; deve pertanto essere forzato a livello alto per permettere il regolare funzionamento dell'integrato. Ponendo, mediante istruzioni firmware, il segnale sul pin relativo (GPIO13) a livello alto, il malfunzionamento si è immediatamente risolto.

## 6. PROGRAMMAZIONE FW E UTILIZZO PRATICO

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, e adattare 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:



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 http. L'applicazione di base fornisce due pagine web (HTML), che vengono compilate esternamente al microcontrollore e gestite attraverso il file system.

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;

- o Macchina a stati software che si occupa della formattazione della stringa corretta per l'estrazione di data e ora;
- RTC
  - o Scrittura dei registri, avvio dell'oscillatore;
  - o Lettura dei registri;
  - o 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 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. Analogamente 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\_ether\_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 pull-up.

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

- i. START (S): transizione 1/0 della linea SDA con SCL a livello alto;
- ii. ADDR+W: indirizzo a 7 bit dello slave e ottavo bit a 0 per indicare una scrittura.
- iii. ACK/NACK dello slave. Acknowledgement è indicato dal nono bit forzato a '0' dallo slave.
- iv. INDIRIZZO del registro in scrittura (8 bit)
- v. ACK/NACK dello slave interpellato
- vi. DATO IN SCRITTURA
- vii. ACK/NACK dello slave interpellato
- viii. STOP (P)

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

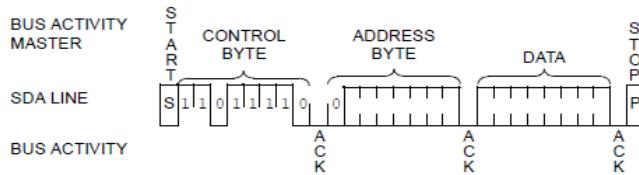


Figura 60 - Scrittura di un singolo registro

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

```

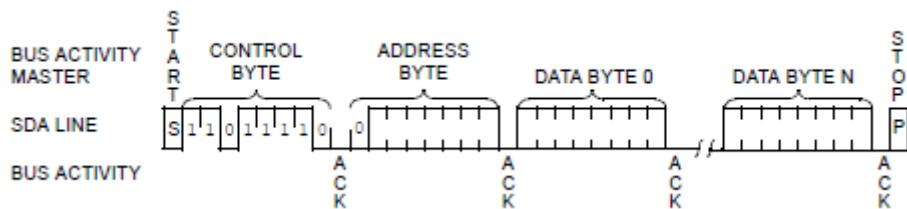


Figura 61 - Scrittura multipla di registri

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).

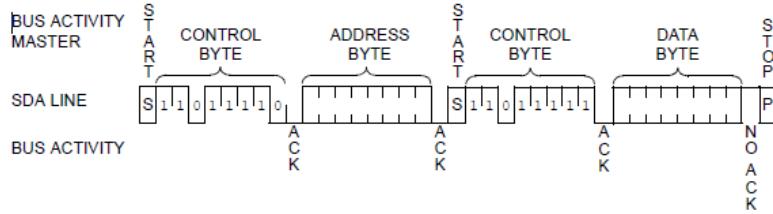


Figura 62 - Lettura

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.

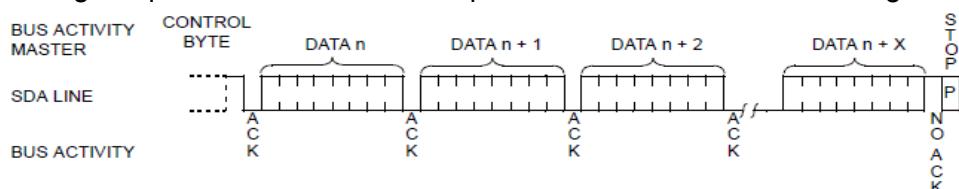


Figura 63 - Lettura con autoincremento dell'indice

### 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):

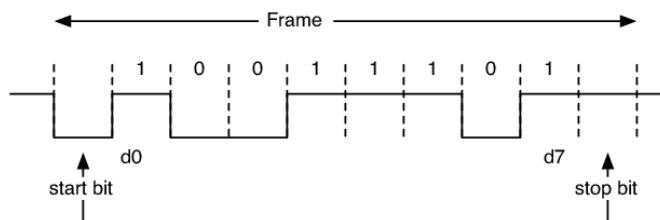


Figura 64 - Trasmissione dati

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 dall'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 inopportuno usare tali funzioni, poiché costringono lo svolgimento del codice a bloccarsi in attesa di dati in ricezione. Tali funzioni sono normalmente dette funzioni blocking.

Per evitare questo inconveniente, conviene impostare e sfruttare l'interrupt generato dalla periferica alla ricezione di ogni byte.

**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, la ricezione di un carattere dall'interfaccia UART.

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, RMC ...);
- 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 (ovvero 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. Nella sezione Allegati (9.3) viene riportato lo schema della macchina a stati implementata.

Il segnale TIMEPULSE, o 1PPS, è rilevato dall'interrupt (IRQ) associato al pin del microcontrollore relativo (GPIO2). 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.

## 7. APPROFONDIMENTO - PIO

Programmable IO è un tipo di periferica integrata nel RP2040 che permette la creazione di interfacce o blocchi funzionali tramite software che viene eseguito indipendentemente dall'esecuzione del codice in esecuzione sulla CPU.

Questo è molto importante perché combina la flessibilità di un'interfaccia completamente personalizzabile con le prestazioni dell'esecuzione in HW.

Ogni PIO è costituito da quattro macchine a stati che possono essere considerate come dei semplici processori specializzati nelle operazioni di movimento di dati.

L'interfacciamento tra il sistema e le macchine a stati avviene in due modi principali:

- **Memoria Istruzioni:** è una memoria condivisa tra le 4 macchine a stati che può contenere fino a 32 istruzioni, ha la caratteristica di poter essere letta contemporaneamente da più macchine senza che si verifichino conflitti. Questa memoria può solamente essere letta dalle macchine a stati ma può essere scritta dalla CPU o da altre periferiche come il DMA, essendo connessa al bus di sistema.
- **FIFO 4X32:** Ogni macchina a stati dispone di due buffer FIFO 4x32 che sono accessibili dal sistema:
  - uno di ingresso: permette di trasferire dati dal sistema alla macchina a stati.
  - uno di uscita: permette di spostare dati dalla macchina a stati al sistema.

### 7.1 Struttura di una macchina a stati

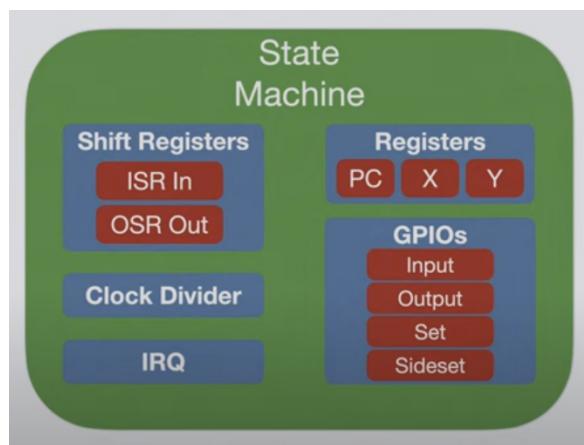


Figura 65 - Struttura macchina a stati

Ciascuna macchina a stati contiene 3 registri principali:

- **PC:** program counter, contiene l'indirizzo in memoria dell'istruzione in esecuzione.
- **X & Y:** sono due registri da 32 bit che possono essere usati liberamente. Oltre a questi tre registri principali vi sono anche due shift register dedicati a l'input e l'output tra le FIFO e i GPIOs. Un ruolo molto importante è quello del IRQ, modulo che permette di lanciare degli interrupt con il fine di allertare la CPU oppure possono essere usati per sincronizzare più macchine a stati tra loro.

### 7.1.1 IO Mapping

La comunicazione con l'esterno invece viene mediata tramite l'IO Mapping che consente di assegnare a ciascuna macchina a stati ciascuno dei 30 pin di input e/o output.



Figura 66 - IO mapping

- Input: è possibile definire quale pin viene indicizzato come primo nella numerazione della macchina a stati.
- Output: è possibile definire quale pin di output viene indicizzato come primo e il numero di pin di output [0-31]
- Set: Analogo ai pin di output [0-5]
- Side set: Pin di output che possono essere gestiti parallelamente all'esecuzione di qualunque istruzione.

### 7.1.2 ISA

L'ISA della macchina a stati è costituita da un assembly dedicato con 9 istruzioni e può essere assemblato tramite l'apposito assemblatore. Le istruzioni disponibili più importanti sono:

- **SET destination[PINS,PINDIRS,X,J], data [0-31]**

*Scrive i dati nella destinazione*

Parametri per destination:

- pins:
- pindirs
- X
- Y

- **JUMP (condition) target [0-31]**

*Salto assoluto condizionale*

Esempi di condizione:

- !X !Y salta se il registro specificato ha valore nullo
- X- - Y- - se il valore del registro non è nullo viene decrementato di uno e il salto viene eseguito
- X!=Y salta solo se i due registri hanno valori differenti
- PIN salta in base al valore logico del pin.
- !ORSE salta se l'output shift register non è vuoto

- **MOV destination, source**

Copia il contenuto di source in destination

Esempi di destinazione:

- pins
- X e Y

- EXEC: fa in modo che il valore copiato sia decodificato e eseguito come istruzione al ciclo di clock successivo.
- PC: program counter
- ISR e OSR: permettono di scrivere negli shift register di input e output

Esempi di source:

- Pind
- X
- Y
- NULL
- STATUS
- ISR
- OSR

- **IN source, bitcount [0-31]**  
shifta i dati in source nel ISR
- **OUT destination, bitcount [0-31]**  
shift data OSR nel registro di destinazione.

E' importante considerare che ogni istruzione non bloccante viene eseguita in un ciclo di clock, i wait ([0-31]) specificano quanti cicli una istruzione deve durare.

Un'altra funzionalità importante è la cosiddetta sideset che permette di cambiare stato dei pin puntati dall'omonimo registro mentre una qualsiasi altra istruzione è in esecuzione.

## 8. BUG e IMPLEMENTAZIONI MANCANTI

Riportiamo in questa sezione i bug presenti nel progetto e le funzionalità mancanti da sviluppare.

### 8.1 Interfaccia di rete

Per quanto riguarda la parte riguardante l'interfaccia di rete, si ha che la scheda presenta diverse criticità. Innanzitutto, l'acquisizione dell'indirizzo IP di rete è possibile unicamente seguendo i seguenti passaggi:

- alimentare la scheda attraverso USB o alimentazione con connettore;
- collegare il cavo di rete (il led verde lampeggiava mentre il led giallo è acceso statico);
- scollegare il cavo di rete **mantenendo** la scheda alimentata;
- riconnettere il cavo di rete (il led verde lampeggiava, il led spento rimane spento).

Nel caso in cui non venga scollegato e ricollegato il cavo di rete, non sarà assegnato un indirizzo IP alla scheda. Per verificare se la procedura di connessione è stata completata correttamente abbiamo due metodi: verificando che il led giallo sia spento (come da protocollo) o attraverso il controllo degli indirizzi IP assegnati sul modem. Una terza alternativa è attivare il debug attraverso un terminale seriale che risulta già presente in libreria.

Da questo problema ne deriva l'impossibilità di sfruttare l'alimentazione PoE da sola, poiché la sospensione dell'alimentazione provocherebbe la disconnessione.

Un'altra criticità è l'impossibilità di utilizzare pagine html dinamiche da utilizzare sul server. Risulta tuttora impossibile il funzionamento del server web se si desidera modificare il file da inviare. Dopo ulteriori test, è stato appurato che nemmeno creare dei file ad hoc con successiva modifica in fase di caricamento ha permesso un corretto funzionamento. In quest'ultimo caso specifico, il server web funziona correttamente alla prima connessione, tuttavia al ricaricamento della pagina smette di funzionare e rispondere. Risulta perciò non disponibile la funzionalità NTP o di riferimento temporale.

### 8.2 Monitoraggio orario

Per quanto riguarda la parte della gestione del tempo è presente un bug dove il caricamento massivo dei dati nei registri dell'integrato MCP non provoca il suo avvio, benché sia presente il campo configurato a tale scopo. Per questo motivo è stata implementata una soluzione ridondante ove, terminato il caricamento massivo, si procede a caricare nuovamente il registro riguardante l'avvio (nel nostro caso quello dei secondi). Risulta comunque corretto l'orario inserito nonostante questo doppio passaggio, come confermato dal modulo GPS che non modifica costantemente l'orario salvato.

### 8.3 Modulo GPS

Il ricevitore GPS fornisce il dato orario in formato UTC, ossia riferito al meridiano di Greenwich, e senza considerare il cambio d'ora. Per ottenere l'ora locale (LT, Local Time) è necessario considerare il fuso orario della propria zona (per l'Italia, UTC+1) e l'adozione del DST o ora legale. Quindi risulta necessaria una routine che regoli l'ora di conseguenza, considerando il periodo di inserimento dell'ora legale che va dall'ultima domenica di marzo

alle ore 2.00, all'ultima domenica di ottobre alle ore 3.00. Anche la data va regolata, una volta che il GPS abbia acquisito almeno quattro satelliti per la determinazione di posizione e data, a seconda dell'orario. Ad esempio, alle 23.30 UTC del 31 dicembre 2022 (ora solare), in Italia sono le 00.30 del 1 gennaio 2023.

A livello di parsing delle sentences ricevute, al momento non si considerano i due caratteri componenti il checksum al termine di ogni stringa. Normalmente, è opportuno confrontare il checksum ricevuto con quello calcolato sulla base del corpo della sentence appena letta, per discriminare le stringhe ricevute correttamente da quelle contenenti errori nei caratteri.

## 9. ALLEGATI

### 9.1 Assegnamento footprint

1	C1 -	0.1F : Capacitor_THT:CP_Radial_D10.0mm_P5.00mm
2	C2 -	12p : Capacitor_SMD:C_0603_1608Metric
3	C3 -	100n : Capacitor_SMD:C_0603_1608Metric
4	C4 -	10u : Capacitor_SMD:C_0805_2012Metric
5	C5 -	12p : Capacitor_SMD:C_0603_1608Metric
6	C6 -	100n : Capacitor_SMD:C_0603_1608Metric
7	C7 -	100n : Capacitor_SMD:C_0603_1608Metric
8	C8 -	100n : Capacitor_SMD:C_0603_1608Metric
9	C9 -	100n : Capacitor_SMD:C_0603_1608Metric
10	C10 -	100n : Capacitor_SMD:C_0603_1608Metric
11	C11 -	100n : Capacitor_SMD:C_0603_1608Metric
12	C12 -	2u2 : Capacitor_SMD:C_0603_1608Metric
13	C13 -	100n : Capacitor_SMD:C_0603_1608Metric
14	C14 -	2u2 : Capacitor_SMD:C_0603_1608Metric
15	C15 -	100n : Capacitor_SMD:C_0603_1608Metric
16	C16 -	100n : Capacitor_SMD:C_0603_1608Metric
17	C17 -	1u : Capacitor_SMD:C_0603_1608Metric
18	C18 -	2u2 : Capacitor_SMD:C_0603_1608Metric
19	C19 -	470p : Capacitor_SMD:C_0603_1608Metric
20	C20 -	12p : Capacitor_SMD:C_0603_1608Metric
21	C21 -	12p : Capacitor_SMD:C_0603_1608Metric
22	C22 -	100n : Capacitor_SMD:C_0603_1608Metric
23	C23 -	100n : Capacitor_SMD:C_0603_1608Metric
24	C24 -	10u 16V : Capacitor_SMD:C_0805_2012Metric
25	C25 -	100n : Capacitor_SMD:C_0603_1608Metric
26	C26 -	100n : Capacitor_SMD:C_0603_1608Metric
27	C27 -	100n : Capacitor_SMD:C_0603_1608Metric
28	C28 -	100n 100v : Capacitor_SMD:C_0603_1608Metric
29	C29 -	2u2 : Capacitor_SMD:C_0603_1608Metric
30	C30 -	2u2 : Capacitor_SMD:C_0603_1608Metric
31	C31 -	100n 100v : Capacitor_SMD:C_0603_1608Metric
32	C32 -	100n : Capacitor_SMD:C_0603_1608Metric
33	C33 -	470p : Capacitor_SMD:C_0603_1608Metric
34	C34 -	100n : Capacitor_SMD:C_0603_1608Metric
35	C35 -	100u : Capacitor_SMD:C_0805_2012Metric
36	D1 -	BAT54C : Package_TO_SOT_SMD:SOT-23
37	D2 -	MB18 : Package_SO:SO-4_4.4x3.6mm_P2.54mm

Figura 67:69 - Assegnazione footprint

---

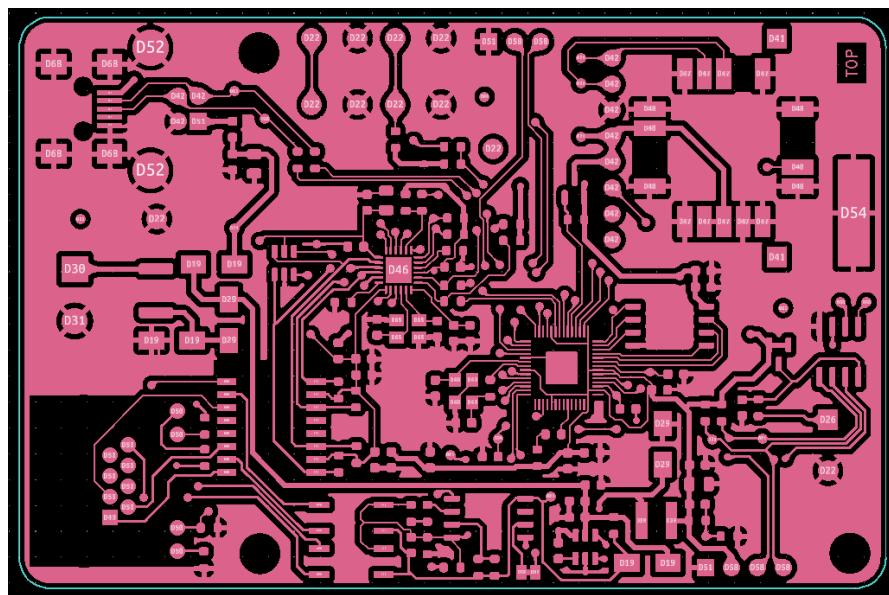
```

38    D3 -          MB1S : Package_SO:SO-4_4.4x3.6mm_P2.54mm
39    D4 -          LED : LED_SMD:LED_0603_1608Metric
40    D5 -          S1B : Diode_SMD:D_SMA
41    D6 -          D_Schottky : Diode_SMD:D_SMA
42    D7 -          SMAJ56 : Diode_SMD:D_SMA
43    D8 -          S1B : Diode_SMD:D_SMA
44    D9 -          D_Schottky : Diode_SMD:D_SMA
45    E1 -          SparkGap : PSE2021:SPARK_GAP_TRIANGLE
46    F1 - Polyfuse_Small: fino a 500ma : Fuse:Fuse_1812_4532Metric
47    H1 -          MountingHole : MountingHole:MountingHole_3mm
48    H2 -          MountingHole : MountingHole:MountingHole_3mm
49    H3 -          MountingHole : MountingHole:MountingHole_3mm
50    IC1 -          LAN8742A-CZ : Package_DFN_QFN:HQQFN-24-1EP_4x4mm_P0.5mm_EP2.5x2.5mm_ThermalVias
51    IC2 -          SAM-M8Q-0 : PSE2021:SAM-M8Q
52    J1 -          USB_B : Connector_USB:USB_B_OST_USB-B1HSxx_Horizontal
53    J2 -          USB_B_Mini : Connector_USB:USB_Mini-B_Wuerth_65100516121_Horizontal
54    J3 -          8P8C_LED : PSE2021:RJ45_Amphenol_RJHSE-5081
55    J4 -          Conn_01x03 : Connector_PinHeader_2.54mm:PinHeader_1x03_P2.54mm_Vertical
56    J5 - Screw_Terminal_01x02 : TerminalBlock_Phoenix:TerminalBlock_Phoenix_MRDS-1,5-2-5.08_1x02_P5.08mm_Hori
57    J6 - Conn_01x04_Female : Connector_PinHeader_2.54mm:PinHeader_1x04_P2.54mm_Vertical
58    JP1 - SolderJumper_2_Bridged : Jumper:SolderJumper-2_P1.3mm_Bridged_Pad1.0x1.5mm
59    L1 -          200mA 120Ohm : Inductor_SMD:L_0603_1608Metric
60    L2 -          120 Ohm 300mA : Inductor_SMD:L_0603_1608Metric
61    L3 -          22u : Inductor_SMD:L_Vishay_IFSC-1515AH_4x4x1.0mm
62    Q2 -          IRLML6402 : Package_TO_SOT_SMD:SOT-23
63    R1 -          1k : Resistor_SMD:R_0603_1608Metric
64    R2 -          100 : Resistor_SMD:R_0603_1608Metric
65    R3 -          330 : Resistor_SMD:R_0603_1608Metric
66    R4 -          10k : Resistor_SMD:R_0603_1608Metric
67    R5 -          10k : Resistor_SMD:R_0603_1608Metric
68    R6 -          270 : Resistor_SMD:R_0603_1608Metric
69    R7 -          270 : Resistor_SMD:R_0603_1608Metric
70    R8 -          33 : Resistor_SMD:R_0603_1608Metric
71    R9 -          33 : Resistor_SMD:R_0603_1608Metric
72    R10 -         10k : Resistor_SMD:R_0603_1608Metric
73    R11 -         1k : Resistor_SMD:R_0603_1608Metric
74    R12 -         27 : Resistor_SMD:R_0603_1608Metric

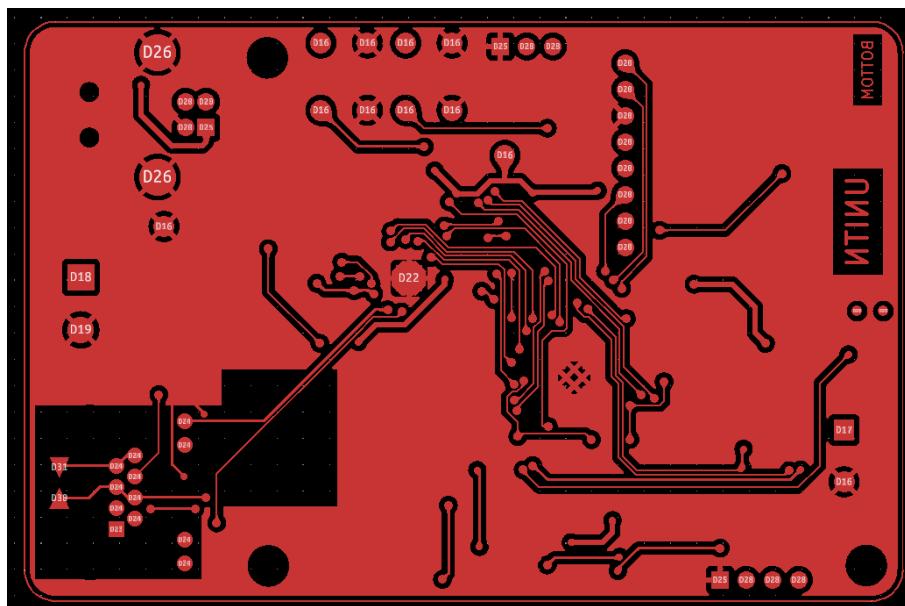
75    R13 -         27 : Resistor_SMD:R_0603_1608Metric
76    R14 -         10k : Resistor_SMD:R_0603_1608Metric
77    R15 -         10k : Resistor_SMD:R_0603_1608Metric
78    R16 -         33 : Resistor_SMD:R_0603_1608Metric
79    R17 -         10k : Resistor_SMD:R_0603_1608Metric
80    R18 -         10k : Resistor_SMD:R_0603_1608Metric
81    R19 -         10k : Resistor_SMD:R_0603_1608Metric
82    R20 -         10k : Resistor_SMD:R_0603_1608Metric
83    R21 -         10k : Resistor_SMD:R_0603_1608Metric
84    R22 -         12k1 : Resistor_SMD:R_0603_1608Metric
85    R23 -         49,9 : Resistor_SMD:R_0603_1608Metric
86    R24 -         49,9 : Resistor_SMD:R_0603_1608Metric
87    R25 -         49,9 : Resistor_SMD:R_0603_1608Metric
88    R26 -         49,9 : Resistor_SMD:R_0603_1608Metric
89    R27 -         47k : Resistor_SMD:R_0603_1608Metric
90    R28 - R4 no fit (o 10k) : Resistor_SMD:R_0603_1608Metric
91    R29 -         50k : Resistor_SMD:R_0603_1608Metric
92    R30 -         178k : Resistor_SMD:R_0603_1608Metric
93    R31 -         50k : Resistor_SMD:R_0603_1608Metric
94    R32 -         330 : Resistor_SMD:R_0603_1608Metric
95    R33 -         27 : Resistor_SMD:R_0603_1608Metric
96    R34 -         33k : Resistor_SMD:R_0603_1608Metric
97    R35 -         10k : Resistor_SMD:R_0603_1608Metric
98    R37 -         1k : Resistor_SMD:R_0603_1608Metric
99    R38 -         OR NF : Resistor_SMD:R_0603_1608Metric
100   R39 -         OR NF : Resistor_SMD:R_0603_1608Metric
101   SW1 -         SW_Push : Button_Switch_THT:SW_PUSH_6mm
102   SW2 -         SW_Push : Button_Switch_THT:SW_PUSH_6mm
103   T1 -          749013011 : Transformer_SMD:Transformer_Ethernet_Wuerth_749013011A
104   TP1 -          3V3 : TestPoint:TestPoint_Loop_D1.80mm_Drill1.0mm_Beaded
105   TP2 -          GND : TestPoint:TestPoint_Loop_D1.80mm_Drill1.0mm_Beaded
106   TP3 -          5V : TestPoint:TestPoint_Pad_D1.0mm
107   TP4 -          1V1 : TestPoint:TestPoint_Pad_D1.0mm
108   TP5 -          POE+ : TestPoint:TestPoint_Pad_D1.0mm
109   TP6 -          POE- : TestPoint:TestPoint_Pad_D1.0mm
110   TP7 -          USB_DP : TestPoint:TestPoint_Pad_D1.0mm

```

## 9.2 File Gerber



*Figura 70 - File Gerber - GTL*



*Figura 71 - File Gerber - GBL*

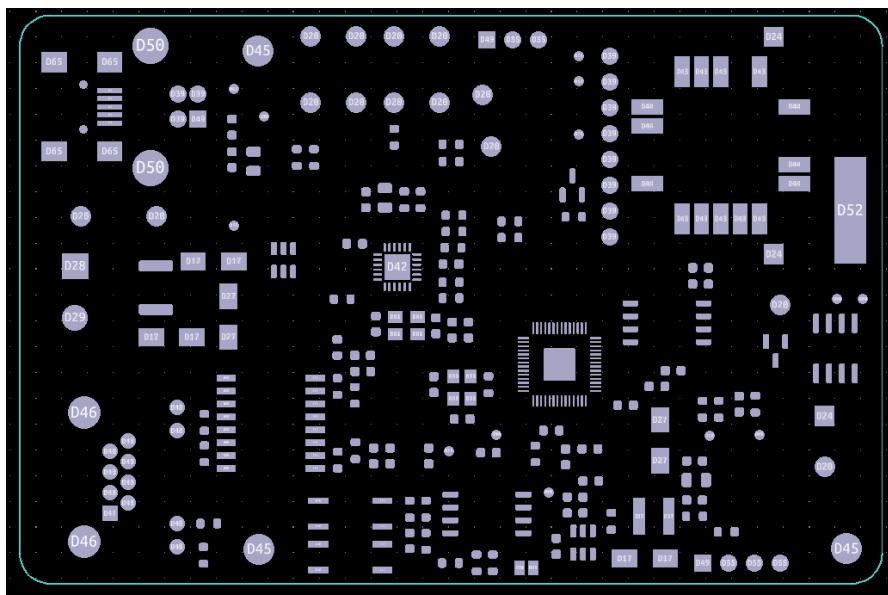


Figura 72 - File Gerber - GTS

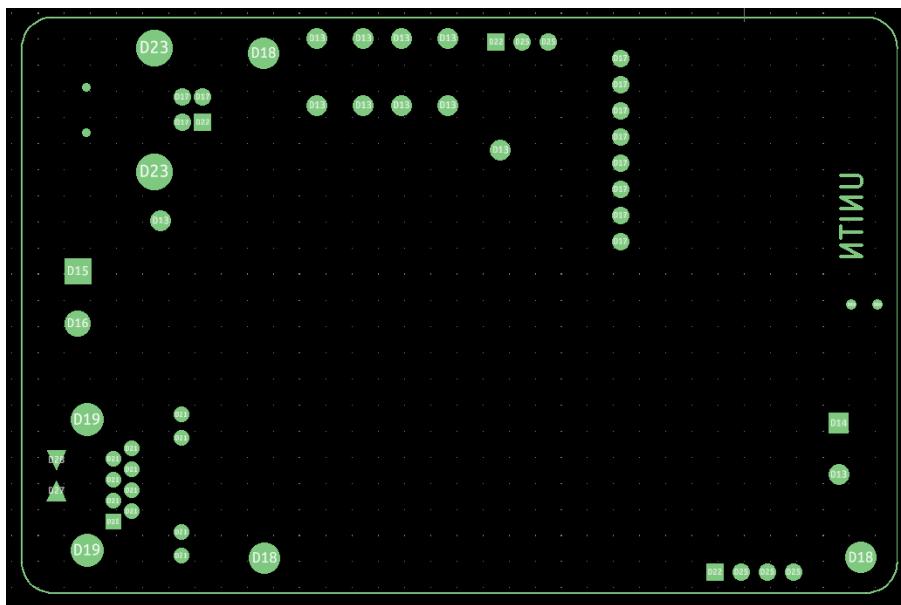


Figura 73 - File Gerber - GBS

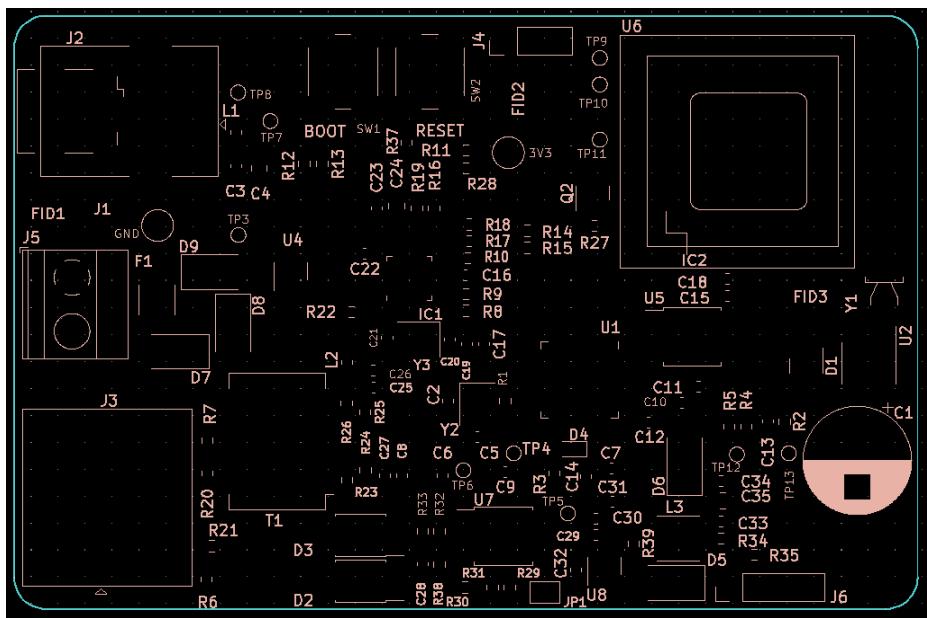


Figura 74 - File Gerber - GTO



Figura 75 - File Gerber - GBO

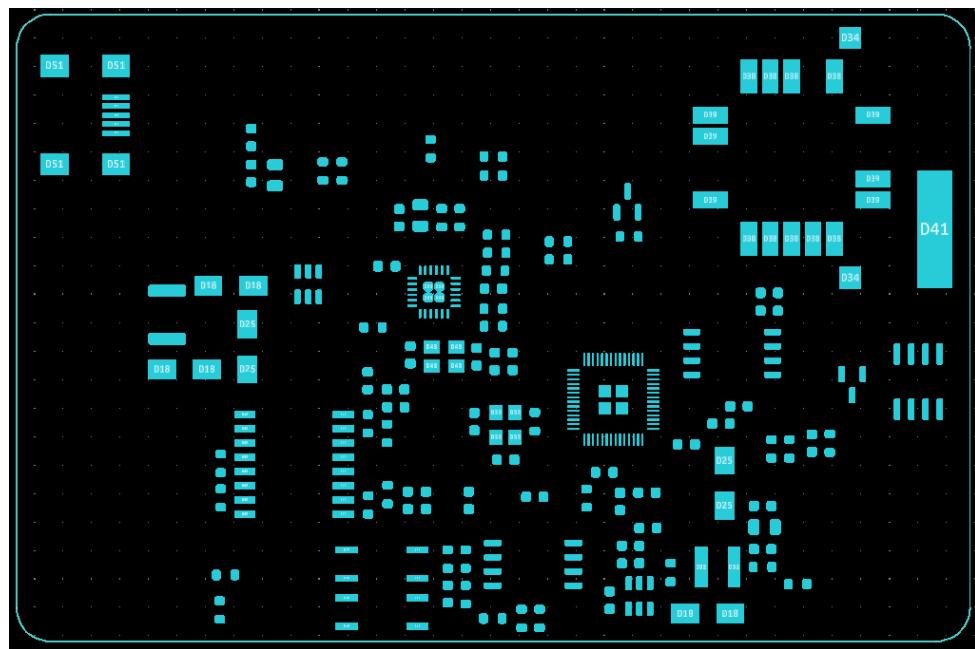


Figura 76 - File Gerber - GTP

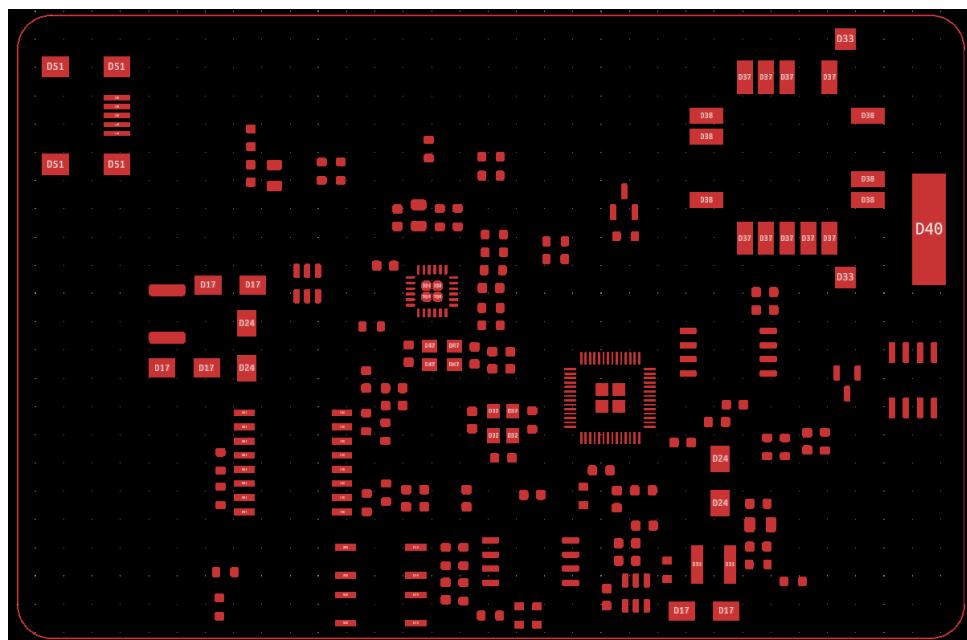


Figura 77 - File Gerber - GTP ridotta del 2%

### 9.3 Macchina a stati per la lettura delle stringhe del GPS

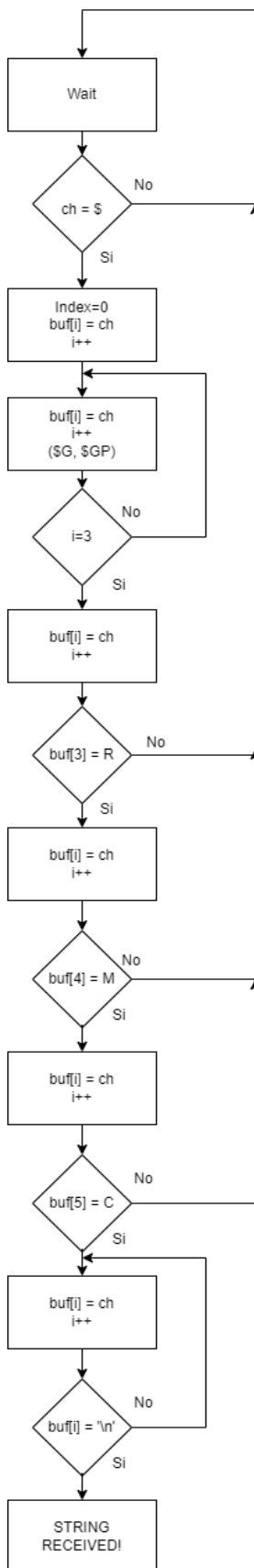


Figura 78 - Macchina a stati GPS

## 9.4 BOM

Component Count: 126

Ref	Qty	Value	Cmp name	Footprint	Description
C1,	1	0.1F	C_Polarized	Capacitor_THT:CP_Radial_D10.0mm_P5.00mm	Polarized capacitor
C2, C5, C20, C21,	4	12p	C	Capacitor_SMD:C_0603_1608Metric	Unpolarized capacitor
C3, C6, C7, C8, C9, C10, C11, C13, C15, C16, C22, C23, C25, C26, C27, C32, C34,	17	100n	C	Capacitor_SMD:C_0603_1608Metric	Unpolarized capacitor
C4,	1	10u	C_Polarized	Capacitor_SMD:C_0805_2012Metric	Polarized capacitor
C12, C14, C18, C29, C30,	5	2u2	C	Capacitor_SMD:C_0603_1608Metric	Unpolarized capacitor
C17,	1	1u	C	Capacitor_SMD:C_0603_1608Metric	Unpolarized capacitor
C19, C33,	2	470p	C	Capacitor_SMD:C_0603_1608Metric	Unpolarized capacitor
C24,	1	10u 16V	C_Polarized	Capacitor_SMD:C_0805_2012Metric	Polarized capacitor
C28, C31,	2	100n 100v	C	Capacitor_SMD:C_0603_1608Metric	Unpolarized capacitor
C35,	1	100u	C_Polarized	Capacitor_SMD:C_0805_2012Metric	Polarized capacitor
D1,	1	BAT54C	BAT54C	Package_TO_SOT_SMD:SOT-23	dual schottky barrier diode, common cathode
D2, D3,	2	MBS1	D_Bridge_+/-AA	Package_SO:SO-4.4x3.6mm_P2.54mm	Diode bridge, +ve/-ve/AC/AC
D4,	1	LED	LED_SMD:LED_0603_1608Metric	Light emitting diode	
D5, D8,	2	S1B	D	Diode_SMD:D_SMA	Diode
D6, D9,	2	D_Schottky	D_Schottky	Diode_SMD:D_SMA	Schottky diode
D7,	1	SMA56	1.5KExxA	Diode_SMD:D_SMA	1500W unidirectional TRANSZORB®
E1,	1	SparkGap	SparkGap	PSE2021:SPARK_GAP_TRIANGLE	Transient Voltage Suppressor, DO-201AE
F1,	1	Polyfuse_Small: fino a 500mA	Polyfuse_Small	Fuse:Fuse_1812_4532Metric	Resettable fuse, polymeric positive temperature coefficient, small symbol
H1, H2, H3,	3	MountingHole	MountingHole	MountingHole:MountingHole_3mm	Mounting Hole without connection
IC1,	1	LAN8742A-CZ	LAN8742A-CZ	Package_DFN_QFN-HVQFN-24-1EP_4x4mm_P0.5mm_EP2.5x2.5mm_ThermalVias	SAM-M8Q RF Receiver Galileo, GLONASS, GPS 1.575GHz, 1.602GHz -165dBm 400kbps
IC2,	1	SAM-M8Q-0	SAM-M8Q-0	PSE2021:SAM-M8Q	
J1,	1	USB_B	USB_B	Connector_USB:USB_B_OST_USB-B1HSxx_Horizontal	USB Type B connector
J2,	1	USB_B_Mini	USB_B_Mini	Connector_USB:USB_Mini-B_Wuerth_65100516121_Horizontal	USB Mini Type B connector
J3,	1	8P8C_LED	8P8C_LED	PSE2021:RJ45_Amphene_RJHSE-5081	RJ connector, 8P8C
J4,	1	Conn_01x03	Conn_01x03	Connector_PinHeader_2.54mm:PinHeader_1x03_P2.54mm_Veutils/schlib/autogen/connector/	(8 positions 8 connected), two LEDs, RJ45 Generic connector, single row, 01x03, script generated (kicad-library-utils/schlib/autogen/connector/)

J5,	1	Screw_Terminal_01x02	Screw_Terminal_01x02	TerminalBlock_Phoenix:TerminalBlock_Phoenix_MKD5-1.5-2.5-08_1x02_P5.08mm_Horizontal	Generic screw terminal, single row, 01x02, script generated (kicad-library-utils/schlib/autogen/connector/)
J6,	1	Conn_01x04_Female	Conn_01x04_Female	Connector_PinHeader_2.54mm:PinHeader_1x04_P2.54mm_Vertical	Generic connector, single row, 01x04, script generated (kicad-library-utils/schlib/autogen/connector/)
JP1,	1	SolderJumper_2_Bridged	SolderJumper_2_Bridged	Jumper:SolderJumper-2_P1.3mm_Bridged_Pad1.0x1.5mm	Solder Jumper, 2-pole, closed/bridged
L1,	1	200mA 120Ohm	L	Inductor_SMD:L_0603_1608Metric	Inductor
L2,	1	25uH	L	Inductor_SMD:L_0603_1608Metric	Inductor
L3,	1	22u	L	Inductor_SMD:L_Vishay_IFSC-1515AH_4x4x1.8mm	Inductor
Q2,	1	IRLML6402	Q_PMOS_GSD	Package_TO_SOT_SMD:SOT-23	P-MOSFET transistor, gate/source/drain
R1, R11, R37,	3	1k	R	Resistor_SMD:R_0603_1608Metric	Resistor
R2,	1	100	R	Resistor_SMD:R_0603_1608Metric	Resistor
R3,	1	R	R	Resistor_SMD:R_0603_1608Metric	Resistor
R4, R5, R10, R14, R15, R17, R18, R19, R20, R21, R35,	11	10k	R	Resistor_SMD:R_0603_1608Metric	Resistor
R6, R7,	2	270	R	Resistor_SMD:R_0603_1608Metric	Resistor
R8, R9, R16,	3	33	R	Resistor_SMD:R_0603_1608Metric	Resistor
R12, R13, R33,	3	27	R	Resistor_SMD:R_0603_1608Metric	Resistor
R22,	1	12k1	R	Resistor_SMD:R_0603_1608Metric	Resistor
R23, R24, R25, R26,	4	49.9	R	Resistor_SMD:R_0603_1608Metric	Resistor
R27,	1	47k	R	Resistor_SMD:R_0603_1608Metric	Resistor
R28,	1	R4 no fit (o 10k)	R	Resistor_SMD:R_0603_1608Metric	Resistor
R29, R31,	2	50k	R	Resistor_SMD:R_0603_1608Metric	Resistor
R30,	1	178k	R	Resistor_SMD:R_0603_1608Metric	Resistor
R32,	1	330	R	Resistor_SMD:R_0603_1608Metric	Resistor
R34,	1	33k	R	Resistor_SMD:R_0603_1608Metric	Resistor
R38, R39,	2	OR NF	R	Resistor_SMD:R_0603_1608Metric	Resistor
SW1, SW2,	2	SW_Push	SW_Push	Button_Switch_THT-SW_PUSH_6mm	Push button switch, generic, two pins
T1,	1	WE749013011	749013011	Transformer_SMD:Transformer_Ethernet_Wuerth_749013011A	
U1,	1	RP2040	RP2040	PSE2021:RP2040-QFN-56	
U2,	1	PCF8563T	PCF8563T	Package_SO:SOIC-8_3.9x4.9mm_P1.27mm	Realtime Clock/Calendar I2C Interface, SOIC-8
U4,	1	USBLCG6-4SC6	USBLCG6-4SC6	Package_TO_SOT_SMD:SOT-23-6	Very low capacitance ESD protection diode, 4 data-line, SOT-23-6
U5,	1	W25Q16UVUSSQ	W25Q16UVUSSQ	Package_SO:SOIC-8_5.275x5.275mm_P1.27mm	
U6,	1	PAM7Q-gpsChip	PAM7Q-gpsChip	PSE2021:PAM-7Q	
U7,	1	TPS2375-1	TPS2375-1	Package_SO:SOIC-8_5.275x5.275mm_P1.27mm	IEEE802.3af PoE Controller, Auto-Retry
U8,	1	LMR16006YQ	LMR16006YQ	Package_TO_SOT_SMD:SOT-23-6	Simple Switcher Buck Regulator, Vin=4-40V, Iout=600mA, Adjustable output voltage, SOT-23-6 package
Y1,	1	Crystal_GND3	Crystal_GND3	Crystal:Crystal_AT310_D3.0mm_L10.0mm_Horizontal_1EP_style1	Three pin crystal, GND on pin 3
Y2,	1	Crystal_GND24	Crystal_GND24	Crystal:Crystal_SMD_3225-4Pin_3.2x2.5mm	Four pin crystal, GND on pins 2 and 4
Y3,	1	25MHz	Crystal_GND24	Crystal:Crystal_SMD_3225-4Pin_3.2x2.5mm	Four pin crystal, GND on pins 2 and 4

Figura 79, 80 - BOM