UNIVERSITÀ DEGLI STUDI DI TRENTO

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE,
DELLE COMUNICAZIONI ED ELETTRONICA

Corso di Progettazione Sistemi Elettronici

Relazione finale 2° semestre

Docenti:
Michele Corrà
Stefano Dalpez

Studenti:
Anna Dal Mas
Giovanni Solfa
Giuseppe Webber
Riccardo Cincelli

ANNO ACCADEMICO 2022/2023

Indice

1 Introduzione **2**
1.1 Sequenza di progetto . . . 2
1.2 KiCAD . . . 3

2 Specifiche della scheda prototipo PSE **5**

3 Progettazione CAD - Schematic **7**
3.1 Bill of Materials . . . 7
3.2 Sezioni circuitali . . . 7
3.2.1 Alimentazione . . . 7
3.2.2 Microcontrollore . . . 10
3.2.3 Interfaccia utente . . . 13
3.2.4 Sensori . . . 17
3.2.5 Test points . . . 19
3.2.6 Mounting holes . . . 19
3.3 Editor dei simboli . . . 20
3.4 Da schematico a PCB . . . 20

4 Progettazione CAD - PCB layout **22**
4.1 PCB rules . . . 22
4.2 Generazione e sbroglio PCB . . . 23
4.3 Ottimizzazione e post processing . . . 23
4.4 File Gerber . . . 24

5 Assemblaggio della PCB **28**

6 Programmazione firmware **29**
6.1 IDE . . . 29
6.2 Struttura del codice . . . 29
6.2.1 Stato settings . . . 30
6.2.2 Stato sampling . . . 30
6.2.3 Stato USB . . . 32

7 Allegati **33**
7.1 BOM . . . 33

1 Introduzione

Oggetto di questa relazione sarà la **scheda PSE** e le tecniche e tecnologie utilizzate per la progettazione e la realizzazione di quest'ultima, applicabili chiaramente ad una generica scheda **PCB** (Printed Circuit Board).

Prima di cominciare ad analizzare la scheda, è conveniente discutere sulla così detta sequenza di progetto, ovvero i vari stadi che, partendo dall'idea o dalla richiesta di un cliente, portano alla realizzazione della scheda completa e pronta all'uso.

1.1 Sequenza di progetto

La sequenza di progetto per la realizzazione di una PCB si basa su due concetti che devono essere valutati prima dell'inizio della progettazione della scheda:

- **Le specifiche:** quali sono le funzioni che la scheda deve svolgere? In che contesto si troverà a lavorare la scheda?
- **Lo studio di fattibilità:** sono in grado, dati i mezzi a disposizione, di realizzare il sistema voluto?

Queste valutazioni preliminari ci consentiranno di avere un'idea più chiara e definita di cosa dovremmo andare a realizzare, partendo dai componenti necessari fino ai test che dovremmo effettuare per controllare l'effettivo funzionamento del sistema.

A questo punto la sequenza generale si divide in più fasi, che possiamo riassumere nei seguenti punti:

1. Realizzazione dello **schematico al CAD**;
2. Scelta dei componenti, valutando datasheet;
3. **Sbroglio del PCB** al CAD;
4. Montaggio della scheda, **saldatura** dei componenti;
5. Scrittura del **firmware** e programmazione;
6. Debug e test finali in laboratorio.

Ognuno di questi macro step è in realtà composto da diversi passaggi che verranno poi discussi nei successivi capitoli. Durante tutto il processo faremo uso di diversi strumenti, tra cui sicuramente l'**ambiente CAD** come più importante, in quanto ci consente di progettare e realizzare una PCB partendo dallo schematico. Successivamente si passerà all'utilizzo di vari **strumenti di laboratorio** per il montaggio dei componenti sulla scheda ed, infine, a **software di programmazione** e debugging accanto a strumenti per la misura ed il test della scheda, quali **multimetro e oscilloscopio**.

1.2 KiCAD

KiCAD è un software open-source, che implementa diversi strumenti in un unico ambiente di lavoro e ciò ci consente di passare da una fase di progetto alla successiva in modo veloce ed efficace, senza dover utilizzare diversi software.

KiCAD è caratterizzato da un **workflow** ben definito (Figura 1), diviso tra tre macro aree: l'**editor di schemi elettrici**, l'**editor circuiti stampati** e il **visualizzatore Gerber**.

Editor di schemi elettrici: ci permette di creare lo schematico del nostro circuito, aggiungendo i componenti da librerie preinstallate o di creare nuovi componenti e librerie tramite il tool library editor. Una volta finito lo schematico e aver utilizzato il tool di verifica del circuito, siamo pronti ad associare i footprint ai vari componenti e generare la netlist e passare al prossimo step.

Editor circuiti stampati: importa la netlist dallo schematico e ci permette di procedere con lo sbroglio del circuito, generando infine i file Gerber. In questo ambiente è inoltre possibile creare footprint ad hoc tramite il footprint editor, da associare poi ai componenti nello schematico. Tool esterni possono essere importati per correggere e fare piccole modifiche al PCB.

Visualizzatore Gerber: è un ambiente che consente di modificare manualmente i file Gerber generati. Questo passaggio può essere tralasciato, ma è buona norma controllare i file prima di consegnarli per la realizzazione della scheda.

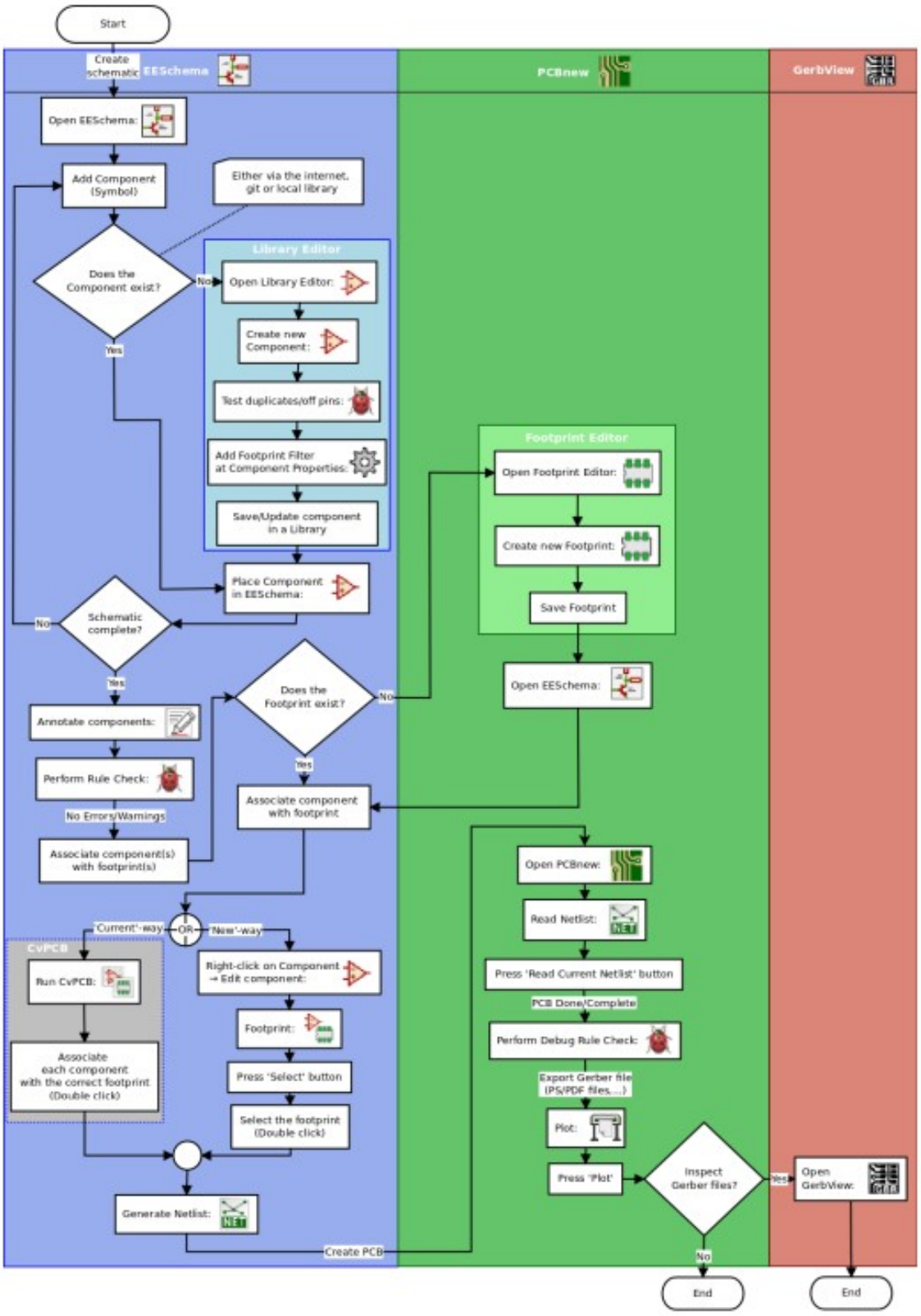

Figura 1: Flowchart del workflow di KiCAD [1]

2 Specifiche della scheda prototipo PSE

La scheda finale dovrà comprendere alcuni blocchi base necessari al funzionamento, tra cui:

- **Microcontrollore** che elaborerà i dati e controllerà le periferiche. Nel nostro caso il microcontrollore utilizzato sarà un RP2040;
- **Input/Output** che saranno collegati a sensori in input e ad attuatori in output per permettere l'interazione con il mondo esterno;
- **Sistema di alimentazione** che può essere di diverso tipo (batteria, cavo, etc.) e con diversi livelli di tensione in base alle necessità;
- **Pin di espansione** per collegare alla scheda altri eventuali sensori o schede di espansione con diverse possibili funzionalità.

In Figura 2 e 3 sono illustrati il package del microcontrollore utilizzato (Figura 2) e il relativo footprint **surface-mount-device** (Figura 3). È importante sottolineare che nella scheda progettata il footprint del microcontrollore è stato "maggiorato", ovvero i piedini sono stati allungati leggermente per facilitare la saldatura.

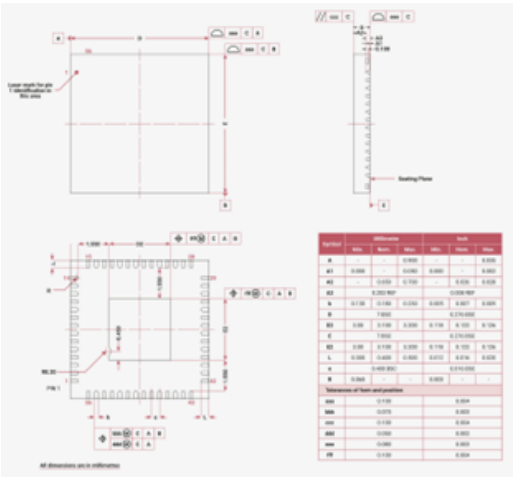

Figura 2: Package QFN-56

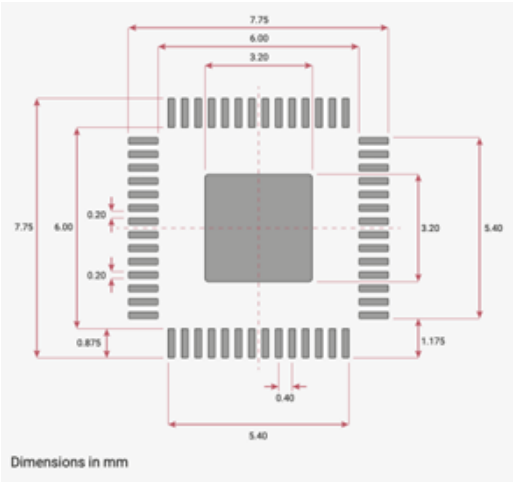

Figura 3: Footprint RP2040 QFN-56

La PCB dovrà poi essere inclusa in un involucro in plastica che lo proteggerà da polvere e acqua. Nel nostro caso la scelta è ricaduta sull'involucro RND 455-00210 in Figura 4, che prevede già sostegni per il montaggio di PCB ed è certificato IP65.

L'impronta dell'involucro in Figura 5 dovrà poi essere importata all'interno nel CAD, così da poter progettare la PCB delle dimensioni adatte ed includere i fori di montaggio.

Figura 4: Involucro utilizzato

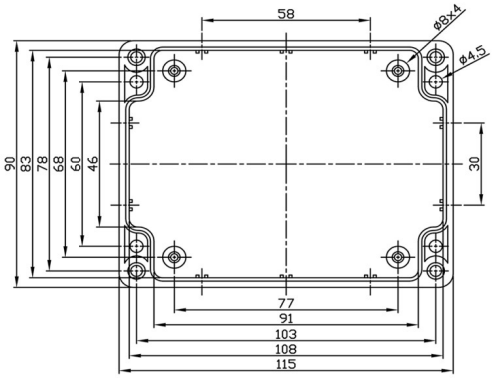

Figura 5: Vista dall'alto con misure

La realizzazione della PCB avverrà su una lastra in vetronite da 1,6 mm a due strati, con layer di rame di spessore 3 μm (Figura 6). Sul top layer verrà applicata la **soldermask**, uno strato di vernice che ha lo scopo di proteggere lo strato di rame dall'ossidazione, lasciando libere solo le pad di saldatura e sul quale verranno infine applicate le serigrafie, come si può osservare in Figura 7.

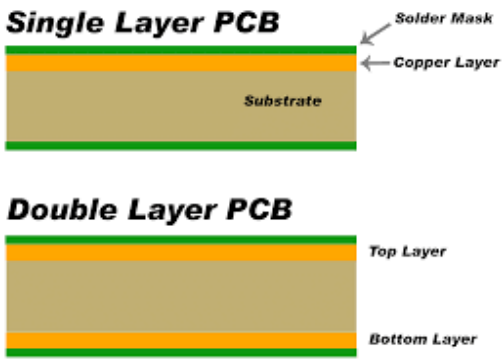

Figura 6: Layers di una PBC a 2 strati

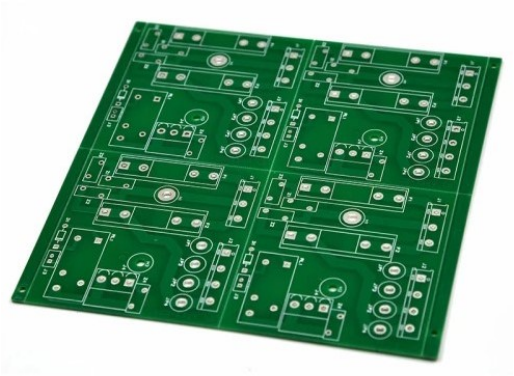

Figura 7: PCB pronta al montaggio

3 Progettazione CAD - Schematic

Il primo passo della progettazione, come detto nei precedenti capitoli, è quello della realizzazione dello schematico al CAD. In questa fase utilizzeremo l'editor di schemi elettrici di KiCAD, importando i componenti dalle librerie preinstallate, o altrimenti se non presenti, realizzando i componenti con il tool editor simboli.

3.1 Bill of Materials

Nello schematico ad ogni componente verranno associati dei parametri, così da renderlo un elemento univoco:

- **Nome di riferimento**: nome composto da una sigla che identifica il tipo di componente (es: Resistore, Condensatore, etc.) e un numero che distingue i vari componenti dello stesso tipo;
- **Valore**: fornisce informazioni aggiuntive sul componente. Nel caso di resistenze, induttore e capacitori deve fornire il valore con unità di misura del componente insieme a tolleranza, tensione di lavoro, etc.

Tutti i componenti utilizzati nello schematico vengono poi raggruppati ed elencati automaticamente dal CAD, in un file chiamato **Bill of Material, BOM** (vedi appendice A.1), che aggiunge ulteriori informazioni legate ad ogni componente:

- **LibPart:** specifica la libreria del CAD da cui proviene il componente. Nel caso di elementi creati dall'utente, indicherà il nome della libreria dell'utente;
- **Footprint**: indica il nome/codice del footprint associato ad ogni componente dello schematico;
- **Data sheet**: se presente, allega il link del data sheet.

3.2 Sezioni circuitali

Vediamo quindi le sezioni circuitali che compongono il nostro circuito.

3.2.1 Alimentazione

- **USB**

  Lo standard **USB (Universal Serial Bus)** definisce le specifiche per i connettori e i protocolli per la comunicazione e l'alimentazione tra dispositivi. La scheda PSE è provvista di un connettore USB, di tipo USB-B-mini illustrato in Figura 8, caratterizzato da 2 pin di alimentazione, da cui vengono ricavati i +5V usati nella scheda, 2 pin differenziali per i dati, con cui è possibile programmare il microcontrollore e 1 pin ID per la selezione della modalità. Nel

circuito è stata inserita una ferrite per ridurre i disturbi prodotti dall'alimentazione, un diodo schottky per bloccare le correnti inverse (che tornano indietro alla USB) e un transistor PMOS tenuto normalmente aperto dalla resistenza R29.

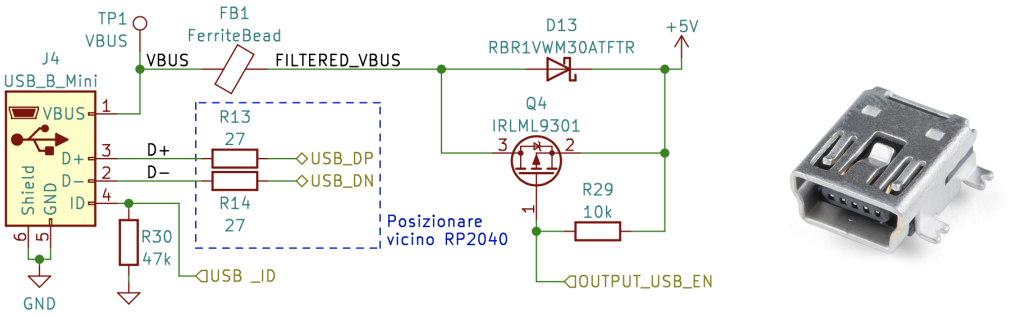

Figura 8: USB mini di tipo B

- **Regolatore di tensione lineare (LDO)**

  Sulla scheda è stato installato un regolatore di tensione lineare (LDO), per convertire i 5V di alimentazione provenienti dalla USB in 3.3V necessari ad alimentare correttamente l'RP2040 (vedi Figura 9). Il circuito è piuttosto semplice: sono presenti due condensatori per rendere l'alimentazione più stabile, un led con la resistenza annessa per segnalare la corretta presenza dei 3.3V ed infine un filtro RC per stabilizzare la linea di tensione analogica.

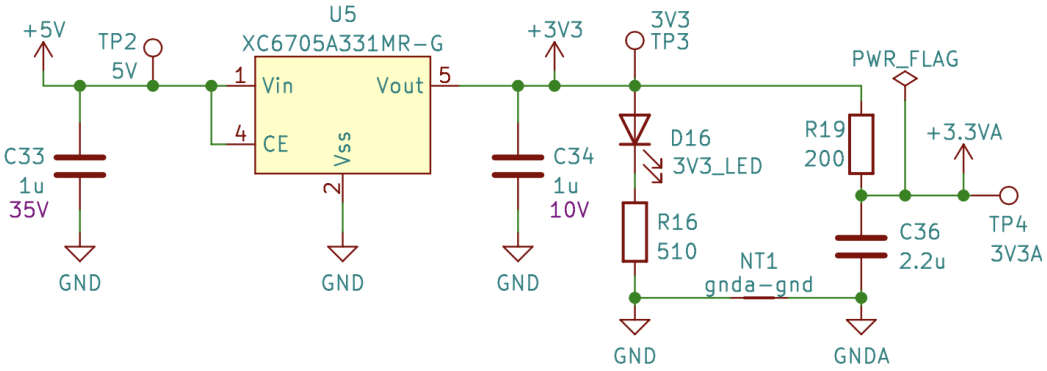

Figura 9: Regolatore lineare LDO

- **Controllo tensione batteria**

  La funzione del circuito in Figura 10, è quella di ritornare in uscita un valore proporzionale alla tensione in entrata dalla batteria, grazie al partitore formato da R20 e R21. A batteria carica, la tensione in ingresso sarà di 12V, che corrisponderà, grazie al partitore, a 3.3V in uscita. Se la tensione della batteria scende, cioè se la batteria si scarica, anche la tensione in uscita si abbasserà. In

questo modo è possibile fare una stima della percentuale di carica della batteria via software, leggendo il valore di tensione con un ADC.

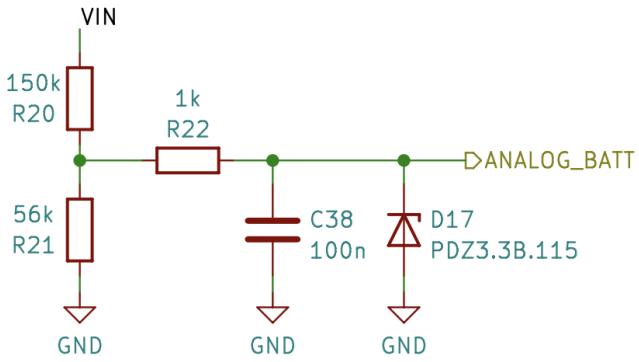

Figura 10: Circuito di controllo tensione batteria

- **Buck converter**

  Tutta la sezione del buck converter è stata progettata per poter utilizzare una batteria, o un qualsiasi altro tipo di alimentazione esterna, al posto della USB. Il buck converter utilizzato (TPS563201) è un particolare convertitore DC-DC di tipo switching che trova impiego per ridurre la tensione (convertitore **step-down**, o riduttore di tensione).

  Nel nostro caso, è stato utilizzato per convertire i 12V in ingresso in 5V, che a loro volta verranno convertiti in 3.3V dal regolatore lineare LDO illustrato in precedenza.

  Il convertitore buck comprende un MOSFET interno per lo switching e presenta una tensione di riferimento interna fissa a 0.765V che viene confrontata con la tensione di feedback in ingresso al pin 4. La tensione V_{out} è la tensione che vogliamo in uscita dal convertitore, nel nostro caso 5V, e tramite la formula riportata nello schematico si determinano le resistenze per il partitore R17 e R18.

  Ai 12V in ingresso sono stati collegati un fusibile, un diodo TVS, in grado di bloccare la tensione ad una certa soglia e un altro semplice diodo di protezione (eventualmente sostituibile con un diodo schottky). Proseguendo verso destra, si osserva che sono stati piazzati 3 condensatori per mantenere stabile l'alimentazione. I diodi D14 e D15 rappresentano una porta OR. Premendo il pulsante di power, viene abilitato il convertitore portando alto il pin enable. Di conseguenza viene attivato il microcontrollore che porterà alto il segnale 5V-enable.

  In uscita sono stati inseriti altri condensatori per il filtraggio delle medie/alte frequenze, un led con la sua resistenza per segnalare la corretta presenza dei 5V e un diodo schottky per bloccare le correnti inverse che tornano indietro verso il convertitore.

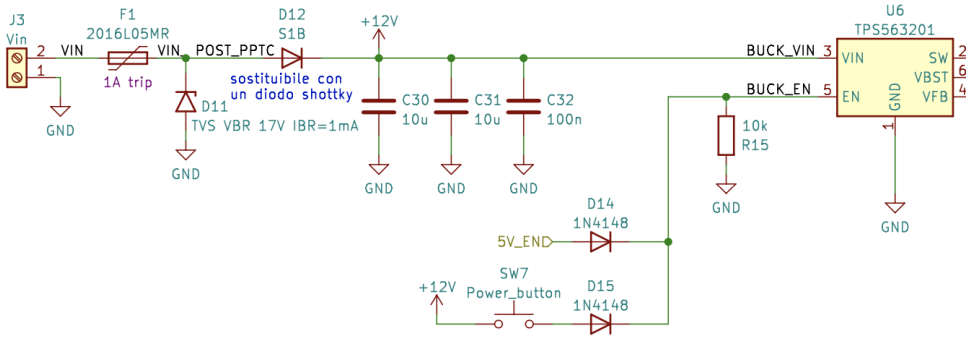

Figura 11: Buck converter sezione input

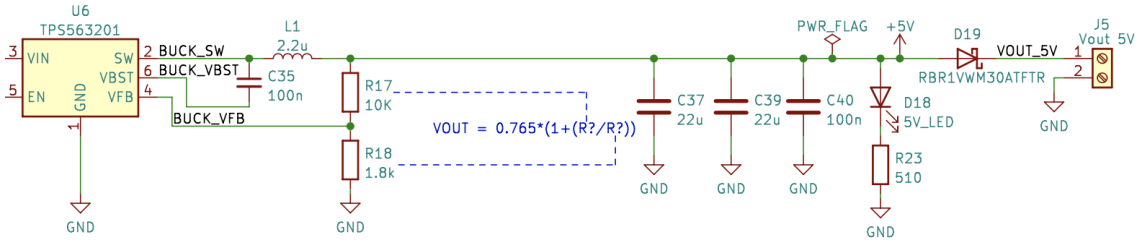

Figura 12: Buck converter sezione output

3.2.2 Microcontrollore

- **RP2040**

  Come spiegato nei capitoli precedenti, la scheda sarà predisposta per il montaggio del **microcontrollore RP2040**.

  L'RP2040 è il primo microcontrollore di Raspberry Pi e include 30 pin GPIO (general-purpose input/output) multifunzione. Questo microcontrollore comprende inoltre due UART (ricevitore-trasmettitore asincrono universale), due SPI (Serial Periferal Interface) e due bus I2C (Inter-Integrated Circuit) per i collegamenti a dispositivi hardware esterni come sensori, display, convertitori digitale-analogico (DAC) e molto altro. Il microcontrollore include anche vari ingressi e uscite programmabili (PIO), che consentono al programmatore di definire nuove funzioni hardware e bus nel software.

  Osservando lo schematico del circuito principale dell'RP2040 riportato in Figura 13, è possibile notare che oltre ai collegamenti con le altre parti di circuito, sono stati inseriti diversi condensatori. In particolare sono stati inseriti tanti condensatori da 100nF quanti sono i pin di alimentazione. Essi infatti sono fondamentali per il filtraggio delle medie/alte frequenze. Ulteriori condensatori di disaccoppiamento sono presenti nello schematico per garantire un'alimentazione stabile al microcontrollore. La resistenza di pull-up R4 collegata tra il pulsante di reset e i 3.3V garantisce uno stato determinato all'ingresso del microproces-

sore quando il pulsante non è premuto. Infine è stato collegato un condensatore in parallelo al pulsante, con lo scopo di mitigare l'**effetto bouncing** causato dalle parti meccaniche del pulsante stesso.

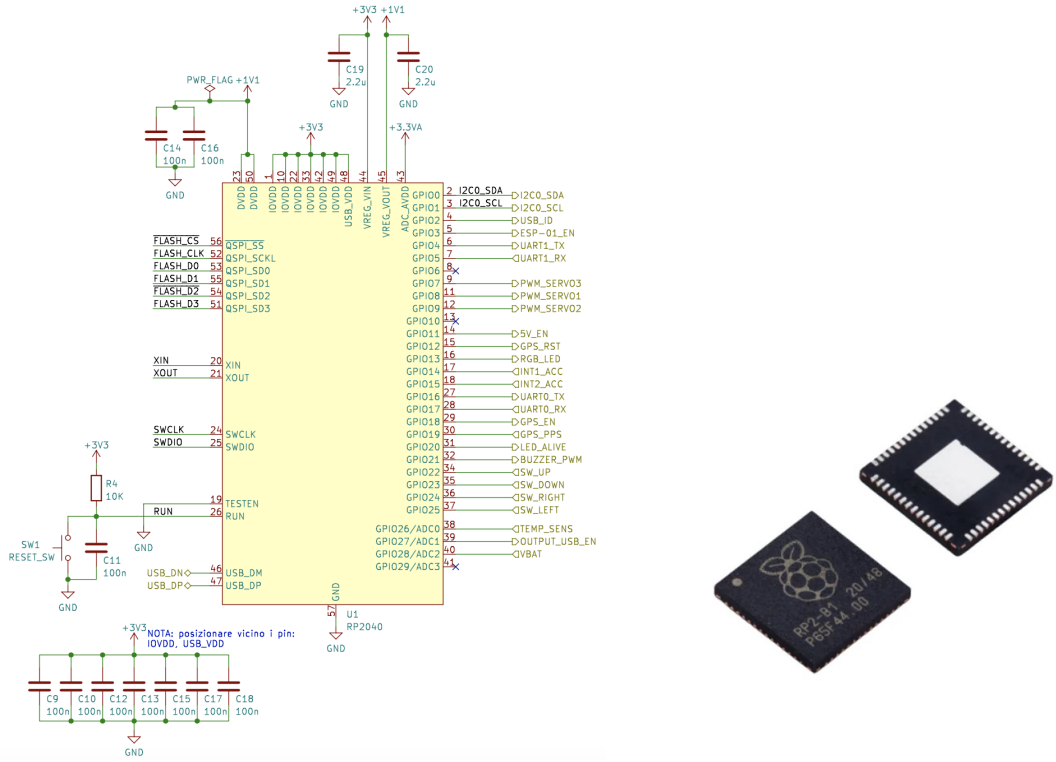

Figura 13: Circuito e package del microcontrollore RP2040

- **Memoria FLASH**

  Il microcontrollore installato sulla scheda necessita di una memoria flash esterna. La memoria utilizzata è la W25Q16JVUSSQ. Nel circuito rappresentato in Figura 14 è presente un condensatore da 100nF che non è stato montato sulla scheda. R5 funge da pull-up e il pulsante SW2 serve per mandare in scrittura la memoria.

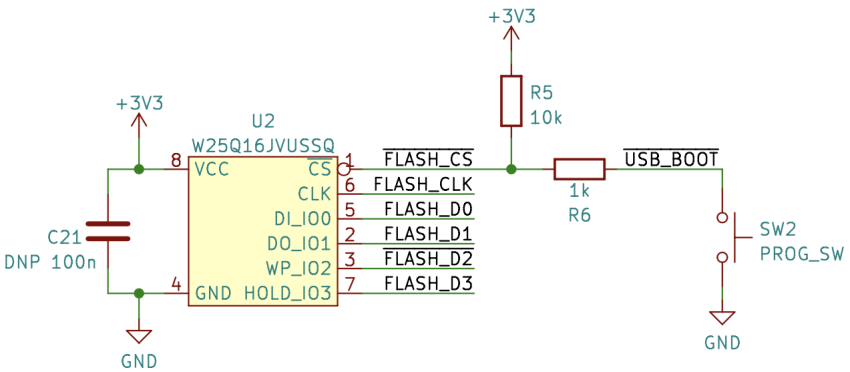

Figura 14: Memoria flash esterna

- **Oscillatore esterno**

  Per quanto riguarda l'oscillatore del microcontrollore (vedi Figura 15), è stato installato un quarzo esterno a 12MHz. Sono stati collegati due condensatori da 18pF, come illustrato nel datasheet.

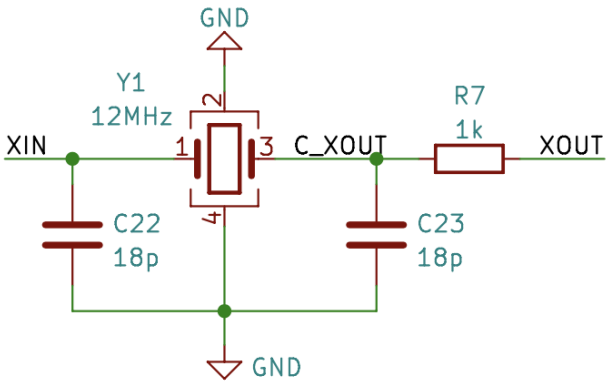

Figura 15: Oscillatore esterno a 12MHz

- **EEPROM**

  Oltre alla memoria flash citata precedentemente è stata anche installata una memoria secondaria. La scelta è ricaduta su una EEPROM: in particolare la Atmel AT24C256C (vedi Figura 16). Essa si interfaccia con il microcontrollore tramite protocollo I2C e alimentata a 3.3V. Da notare che il pin write protect (pin 7) in questo caso è portato a massa. Questa configurazione permette al microcontrollore di avere pieno accesso alla memoria sia in fase di lettura che di scrittura. I pin A0 e A1 servono per impostare l'ultimo bit dell'indirizzo I2C ($GND = 0$, $VCC = 1$).

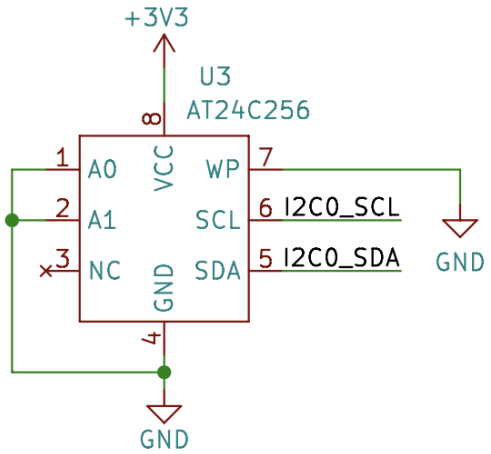

Figura 16: EEPROM

- **Debug**

  Sulla scheda è stato installato anche un connettore di debug, ovvero un connettore diretto all'RP2040 utile in caso si voglia programmare o debuggare la scheda tramite interfaccia SWD.

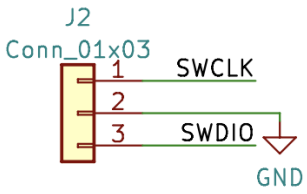

Figura 17: Connettore di debug

3.2.3 Interfaccia utente

- **LED WS2812 RGB**

  Il led tipo WS2812B è un led RGB a controllo digitale. All'interno del package 5050 sono integrati un circuito del LED e un chip di controllo.

  In questo chip è presente un regolatore di luminosità a corrente costante che assicura un colore uniforme anche con differenti tensioni di alimentazione e un'interfaccia di comando seriale a una sola linea. La presenza del chip interno permette ai LED di essere programmati per emettere diversi colori tramite un protocollo ad 1 bit. Per il pilotaggio sono utilizzati solo 3 pin, due per l'alimentazione (+5V e GND) e uno per la programmazione.

  Facendo riferimento allo schematico (vedi Figura 18) si può osservare la presenza di un condensatore da 100nF per ogni pin di alimentazione di ogni singolo LED. Questi condensatori fungono da filtri per la media/alta frequenza, tendenzialmente generata dai segnali digitali. Soffermandosi invece sulla prima parte del circuito, notiamo che è stato inserito un level shifter che permette di passare dai 3.3V in uscita dal microcontrollore ai 5V necessari per garantire una corretta alimentazione dei LED.

  Il level shifter è stato implementato atraverso un MOSFET NMOS polarizzato dalle resistenze R1 e R2. Quando il MOS è aperto (non conduce, segnale DIN in uscita dal micro basso) la linea dato dei LED sarà a livello logico alto, grazie a R3 che fa da pull-up; quando è chiuso (conduce, segnale DIN alto), il MOSFET porta a massa il segnale DIN, quindi la linea dato dei LED avrà livello logico basso. Sostanzialmente è una porta NOT che inverte la logica del segnale in ingresso, in cui i due stati sono 0 o 5V.

  Infine, è stato inserito un connettore per un'eventuale striscia di led esterni e un jumper per far sì che i led interni alla scheda ed esterni fossero scollegati tra loro.

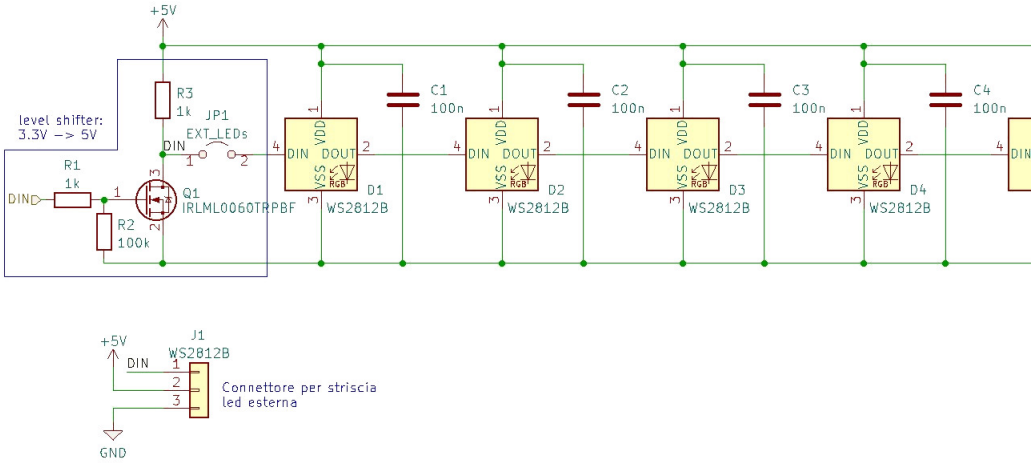

Figura 18: Circuito per i LED WS2812 RGB

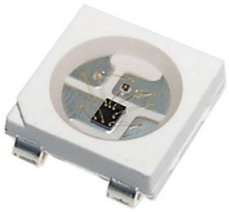

Figura 19: LED WS2812 RGB

- **Display**

  Il display installato sulla scheda PSE è un display OLED controllabile tramite protocollo I2C. In questo caso, oltre al condensatore da 100nF che garantisce una maggior stabilità all'alimentazione, ne è stato aggiunto un altro da 4.7uF a causa di un maggiore consumo rispetto agli altri componenti.

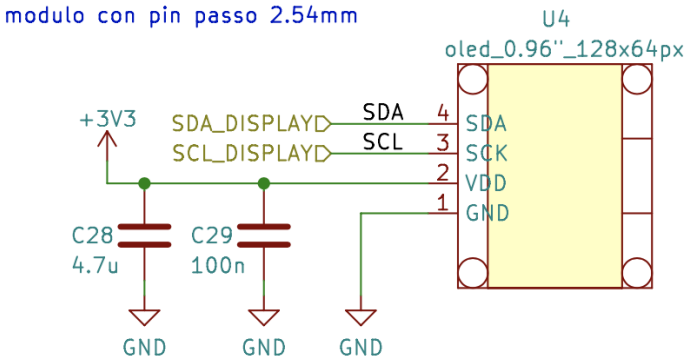

Figura 20: Display OLED con interfaccia I2C

- **LED**

  Tre semplici LED, sono stati aggiunti alla scheda. Questi possono essere utilizzati per segnalare lo stato della scheda in fase di programmazione, come output di un programma o per segnalare la corretta alimentazione.

  Nel nostro caso sono stati montati due LED per segnalare la corretta presenza delle due alimentazioni, ovvero i 5V dell'USB e i 3.3V per il microcontrollore, e un LED in funzione di "LED alive" per l'interfaccia utente. La funzione del led alive è quella di comunicare che la scheda sta funzionando senza problemi.

  In Figura 21 osserviamo il semplice circuito, con la resistenza di polarizzazione calcolata tenendo conto della tensione di alimentazione e della caratteristica I/V del LED.

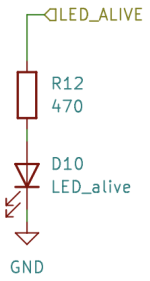

Figura 21: LED alive

- **Pulsanti**

  Per permettere l'interazione con l'utente, sulla scheda sono stati aggiunti sette pulsanti, quattro riportati in Figura 22 per l'utilizzo del display, due per reset e programmazione del micro e uno per l'alimentazione a 12V. Le resistenze di pull-up collegate tra i pulsanti e i 3.3V garantiscono uno stato determinato all'ingresso del microprocessore quando i pulsanti non sono premuti. Il microprocessore avrà quindi un valore basso in ingresso quando il pulsante verrà premuto, alto quando verrà rilasciato.

  Il dimensionamento delle resistenze è stato fatto considerando la resistenza interna degli ingressi del micro che, insieme alla resistenza di pull-up, crea un partitore di tensione. Scegliendo una R troppo grande si rischia di non raggiungere la soglia minima per pilotare la logica CMOS interna, che non riuscirebbe più a rilevare se il pulsante viene rilasciato. Al contrario una R piccola causerebbe una corrente dell'ordine del mA quando il pulsante viene premuto e quindi una grande perdita di potenza.

  Un condensatore è infine stato collegato in parallelo al pulsante, con lo scopo di mitigare l'effetto bouncing causato dalle parti meccaniche del pulsante. Il condensatore infatti introduce una costante di tempo che impedisce alla tensione di raggiungere la soglia logica alta in breve tempo.

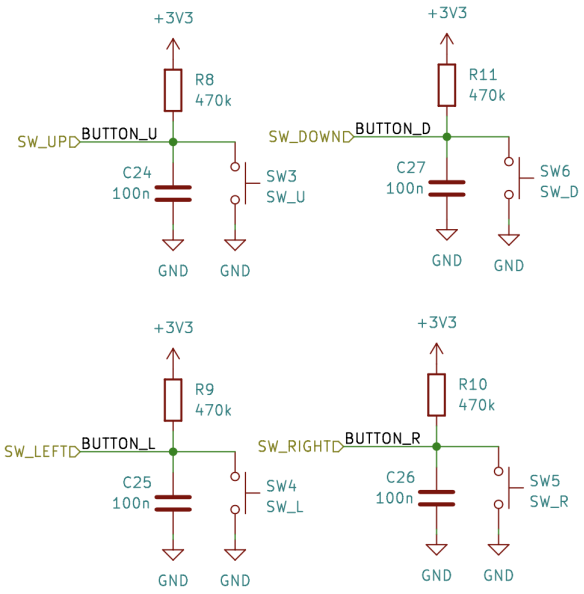

Figura 22: Pulsanti della scheda del corso

- **Buzzer**

  Il buzzer, in questo caso di tipo piezoelettrico, è un dispositivo in grado di generare un suono quando una tensione viene applicata ai suoi capi, sfruttando la deformazione del materiale piezoelettrico al suo interno. Questo tipo di buzzer necessita però di un circuito di pilotaggio, che ne determina anche il suono generato. Il circuito, mostrato in Figura 23, è costituito da un semplice diodo di protezione per evitare di bruciare il transistor BJT o, nel caso peggiore, la porta, e da un commutatore elettronico, in questo caso il DTC114Y, un monolithic IC che include un transistor NPN e le sue resistenze di polarizzazione. Il buzzer verrà pilotato dal microcontrollore tramite la base del BJT, che attivandosi lascerà scorrere una corrente che a sua volta creerà una tensione ai capi del buzzer.

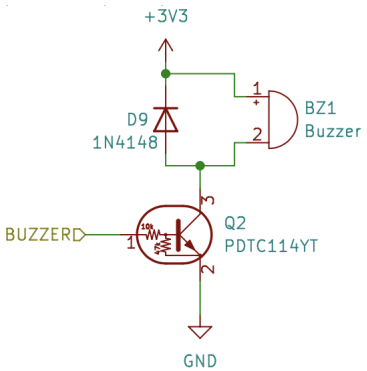

Figura 23: Buzzer

3.2.4 Sensori

- **Accelerometro**

  Un accelerometro è un dispositivo che misura l'accelerazione di un oggetto o di un sistema. All'interno dell'accelerometro sono presenti piccoli sensori, che rilevano i cambiamenti nell'accelerazione grazie a un massa sospesa. Questi sensori convertono l'accelerazione in un segnale elettrico proporzionale, che viene poi elaborato e utilizzato per calcolare l'accelerazione assoluta o relativa all'oggetto.

  Sulla scheda PSE del corso è stato installato un MMA8451Q, ovvero un accelerometro capacitivo a tre assi a basso consumo con 14 bit di risoluzione digitale. Esso sfrutta la variazione di capacità per rilevare lo spostamento della massa, generando un segnale elettrico proporzionale. Lo schema del circuito in Figura 24 è stato progettato seguendo le indicazioni del datasheet del componente.

  L'alimentazione del dispositivo è fornita tramite la linea VDD (da 1.95V a 3.6V). I condensatori C41, C43 da 100nF e C45 da 4.7uF fungono da condensatori di bypass o disaccoppiamento dell'alimentazione e devono essere posizionati il più vicino possibile ai pin 1 e 14 del componente. Essendo interfacciato tramite protocollo I2C, si hanno i segnali di controllo SDA e SCL e il segnale SA0, per settare il bit meno significativo dell'indirizzo I2C del dispositivo. I segnali SDA e SCL sono open drain, quindi a valore alto sono flottanti e a valore basso sono collegati direttamente a massa. Sono infine state inserite le due resistenze di pull-up da 4.7kΩ (R25 e R26) sui bus I2C, necessarie per portare a VDD i segnali che altrimenti rimarrebbero flottanti.

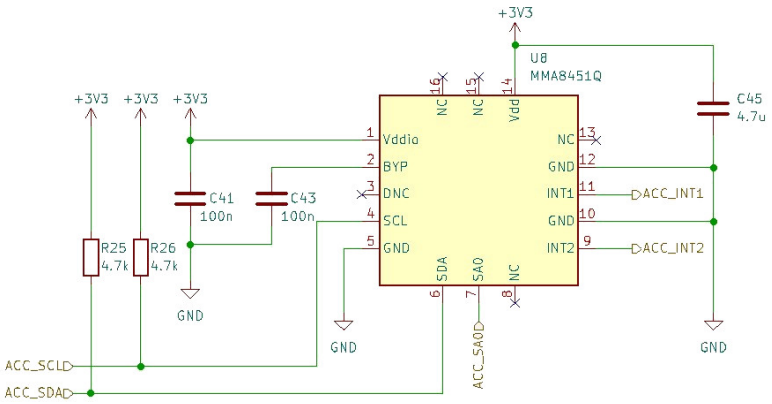

Figura 24: Accelerometro

- **Sensore di temperatura**

  Il sensore TC1047A, scelto per la misura della temperatura sulla scheda PSE, è un sensore lineare basato su un termistore, un elemento semiconduttore che varia molto la sua resistenza al variare della temperatura. Il TC1047A è un sensore di temperatura di uscita lineare la cui tensione di uscita è direttamente proporzionale alla temperatura misurata. Esso può misurare con precisione

la temperatura da -40°C a +125°C. Per il TC1047A, l'intervallo di tensione di uscita è in genere 100mV a -40°C, 500mV a 0°C, 750mV a +25°C e 1,75V a +125°C. Una pendenza di tensione di 10mV/°C consente l'ampia gamma di temperature. Il dispositivo comprende anche il circuito di condizionamento necessario per convertire il valore di temperatura in una tensione che può quindi essere letta direttamente dall'ADC. Il TC1047A si trova in package SOT-23B a 3 pin, rendendolo ideale per l'ottimizzazione dello spazio.

Il circuito in Figura 25 risulta dunque molto semplice: comprende un condensatore in parallelo C46 per garantire un'alimentazione stabile e un semplice filtro passa basso formato da R28 e C48, che riduce i disturbi sul segnale in uscita dal sensore.

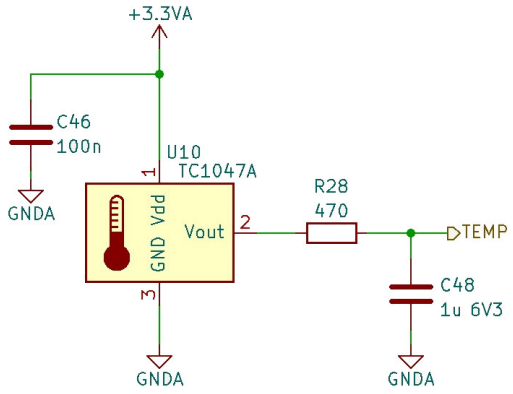

Figura 25: Sensore di temperatura

- **Modulo GPS**

  Per poter acqisire dati relativi alla posizione e al tempo è stato montato un modulo GPS sulla scheda. Il modulo che abbiamo scelto di utilizzare è il SAM-M8Q che offre un'ottima sensibilità e un tempo di acquisizione molto piccolo. Sono supportati vari sistemi di tracciamento quali: GPS, Galileo and GLONASS.

  La comunicazione avviene tramite interfaccia UART, quindi sono necessari un segnale di ricezione (RX) e uno di trasmissione (TX) sia per il microcontrollore che per il modulo. Il baudrate solitamente utilizzato è di 9600 per la trasmissione dei dati.

  L'alimentazione che può andare da 2.7V a 3.6V viene controllata tramite un mosfet di tipo NMOS e un segnale di enable per l'attivazione del GPS. Alcuni condensatori (C42 e C44) fungono da condensatori di bypass e disaccoppiamento e devono essere piazzati molto vicini ai pin di alimentazione 2,3 e 17 del componente.

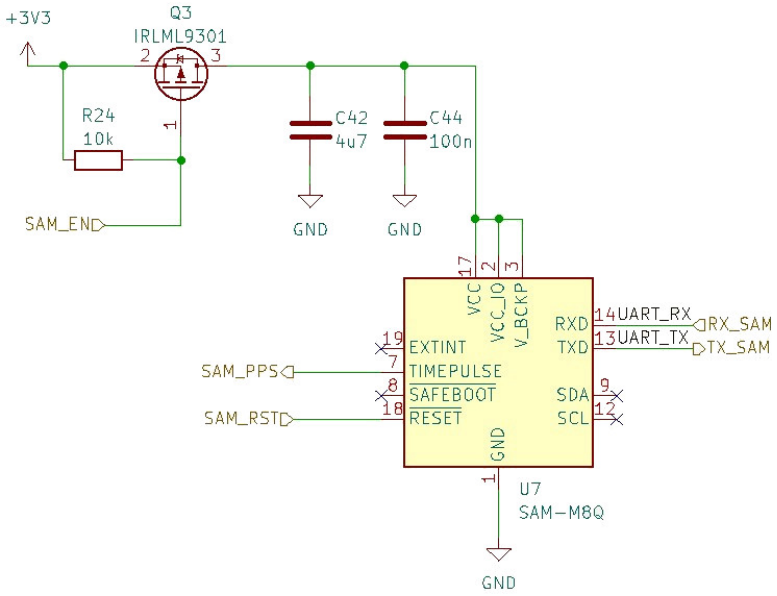

Figura 26: Modulo GPS

- **Espansioni**

  Per poter connettere la scheda con altri dispositivi esterni, come sensori, attuatori o altre schede e riuscire a comunicare tramite protocolli standard come I2C o UART, sono necessari dei connettori. In entrambi i casi i pin necessari saranno 4: 2 di alimentazione e 2 per lo scambio di dati, che si differenziano per i due protocolli, come possiamo notare in Figura 27.

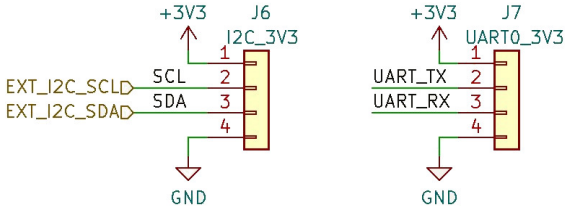

Figura 27: Espansioni UART e I2C

3.2.5 Test points

Per fare test e misurazioni, in fase di debugging o per verificare il corretto funzionamento della scheda, sono stati aggiunti alcuni **test points**, ossia dei jumper facilmente raggiungibili per misurare tensioni con il multimetro. In particolar modo, sono stati inseriti i test points per le alimentazioni (5V e 3.3V).

3.2.6 Mounting holes

L'ultimo elemento che dobbiamo inserire nello schematico sono i **mounting holes**, ossia i fori necessari al fissaggio della PCB all'involucro protettivo. Questi sono

importanti poiché è necessario che il ground della scheda sia connesso all'involucro in cui è contenuto e questo avviene proprio tramite i mounting holes, che sono provvisti di pad di rame consentendo la connessione elettrica tramite la vite con cui verrà fissata la PCB. In questo modo l'intero sistema sarà protetto da **interferenze elettromagnetiche** per il principio della gabbia di Faraday.

3.3 Editor dei simboli

Non sempre le librerie fornite da KiCAD comprendono tutti i simboli degli elementi che dobbiamo inserire nello schematico, come nel caso del fototransistor TEMT6000 usato nella PSE della prima parte del corso per la misura della quantità di luce. In questo caso il CAD ci fornisce lo strumento **editor simboli**, che ci consente di creare i simboli per qualsiasi elemento che vogliamo inserire nel nostro schematico.

Nella finestra di editing è quindi possibile disegnare il nostro simbolo tramite i tool nella barra di destra, aggiungendo poi i pin del dispositivo, stando attenti a selezionare i giusti parametri, tra cui:

- **Tipo elettrico**: specifica se il pin è un ingresso, un'uscita, di potenza, etc;
- **Stile grafico**: importante per specificare pin negati e clock di diverso tipo.

Infine è necessario specificare i nomi dei vari pin e del componente, che verranno poi mostrati quando l'elemento verrà importato nello schematico. L'elemento creato verrà quindi salvato in una libreria e reso disponibile all'aggiunta nello schema elettrico tra le librerie dell'utente.

3.4 Da schematico a PCB

Ultimata la creazione dello schema elettrico, si può procedere con gli ultimi passaggi prima dello sbroglio:

1. **Annotazione dei componenti**: ogni componente aggiunto allo schematico verrà numerato, così da renderlo univoco e distinguibile da altri componenti dello stesso tipo;
2. **Controllo delle regole elettriche**: effettua un controllo delle connessioni tra i componenti e dello schema in generale, evidenziando eventuali errori o avvisi. É inoltre possibile modificare le regole ERC, impostando manualmente su quali tipi di connessione tra pin generare errori/avvisi;
3. **Assegnamento impronte**: ad ogni componente dovrà essere assegnata la sua impronta, o footprint, che rappresenta le dimensioni, la posizione dei pin e il loro tipo (THT o SMD), che verrà poi mostrata nell'editor PCB. Molte impronte per diversi dispositivi sono disponibili nelle librerie fornite da KiCAD, ma nel caso non fossero presenti, il CAD fornisce lo strumento footprint editor che consente di creare o modificare delle impronte basandosi sui datasheet dei componenti;

4. **Generazione Netlist**: contiene una lista con nomi delle net e dei componenti con relativo footprint e le varie connessioni tra questi ultimi;

Siamo ora pronti al passaggio dall'editor di schemi elettrici EESchema, allo strumento **PCB Layout Editor**, passando di fatto alla fase di sbroglio del PCB.

4 Progettazione CAD - PCB layout

In questa fase della progettazione l'obbiettivo è quello di organizzare tutti i componenti sulla scheda PCB, collegarli tra loro tramite piste e infine generare i **file Gerber** della scheda da mandare in produzione.

4.1 PCB rules

Prima di cominciare con lo sbroglio è essenziale definire quelle che sono le regole del PCB, ossia i parametri fisici dei vari elementi della scheda. Ciò è importante per garantire che la PCB sia facilmente fabbricabile dalle aziende di produzione e per limitare problemi di tipo elettromagnetico e di surriscaldamento. Nelle impostazioni della scheda definiamo le regole come mostrato in Figura 28 e Figura 29.

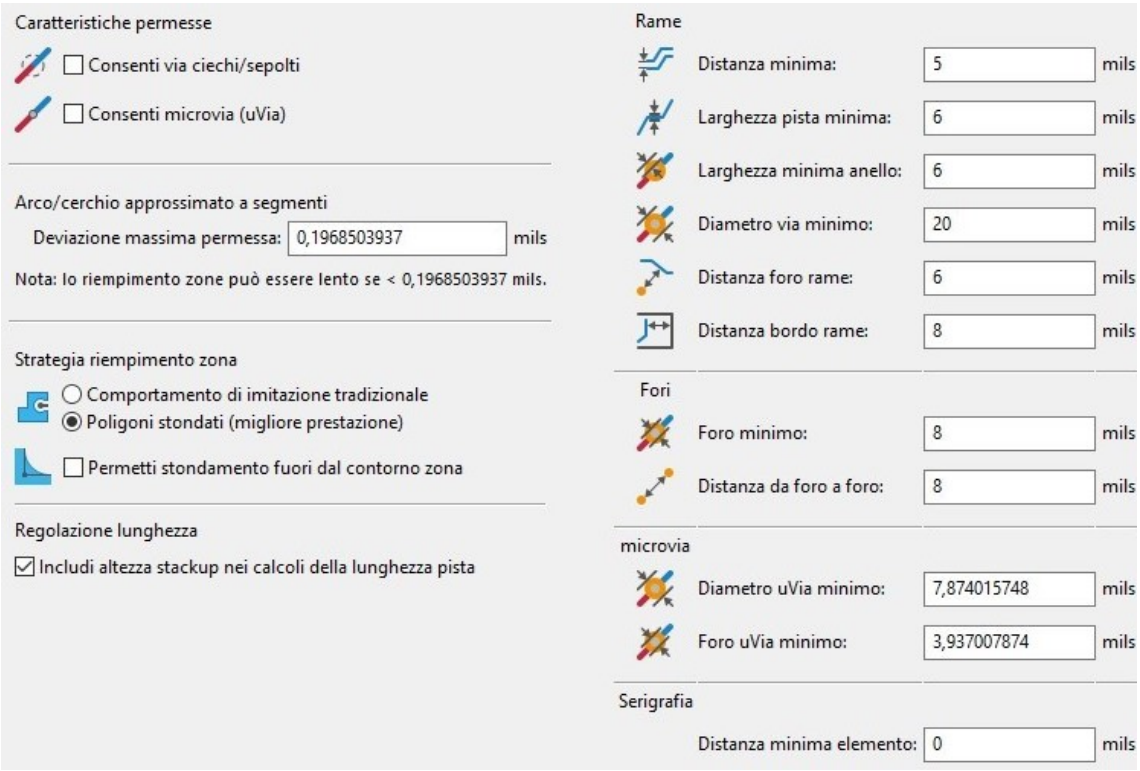

Figura 28: Vincoli minimi per la scheda PSE

Netclass	Distanza	Larghezza piste	Dimensioni via	Foro via	Dimensione uVia	Foro uVia	Larghezza CD	Spazio CD
Default	6 mils	10 mils	32 mils	15 mils	11 mils	4 mils	8 mils	8 mils
POWER	6 mils	20 mils	48 mils	24 mils	11 mils	4 mils	8 mils	8 mils

Figura 29: Regole per la scheda PSE, default e per Netclass di potenza

4.2 Generazione e sbroglio PCB

Siamo quindi pronti a procedere con lo sbroglio. Come primo passo dobbiamo generare le aree di rame al TOP e BOTTOM, ricordandoci di differenziare GND e GNDA. Importiamo l'impronta dell'involucro per delimitare la zona di rame e centrare i mounting holes.

A questo punto rimane solo da procedere con il piazzamento dei componenti sulla scheda e allo sbroglio delle piste.

Oltre ai componenti dello schematico, dobbiamo aggiungere altri elementi:

- **Vias**: sono piccoli fori passanti che fungono da collegamento tra i diversi strati della PCB (in questo caso Top e Bottom). Possono essere usati sia per la connessione tra piste, facendole passare da un lato all'altro, sia come connessione dei piani di massa, come vediamo in Figura 30. In una PCB multistrato possono essere di diverso tipo: pass through, blind o buried;
- **Fiducial**: sono elementi non elettrici che permettono alle macchine pick-and-place di avere dei riferimenti sulla scheda, per poter determinare l'orientamento di quest'ultima. Solitamente sono degli anelli non forati, come in Figura 31.

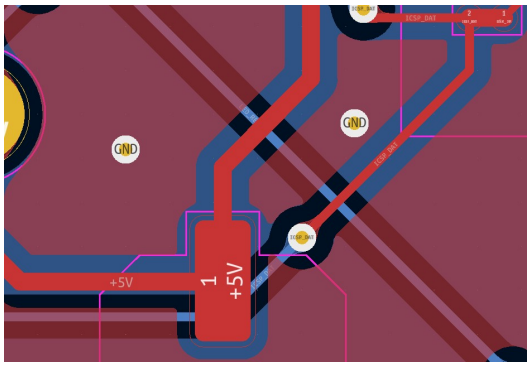

Figura 30: Vias per piste e GND

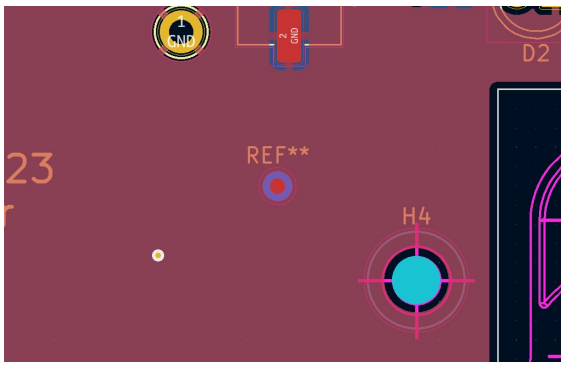

Figura 31: Fiducial della scheda PSE

4.3 Ottimizzazione e post processing

Prima di procedere con la generazione dei file Gerber è buona norma effettuare l'ottimizzazione delle connessioni, in particolar modo aggiungendo i **teardrops**.

I teardrops sono dei rinforzi dei collegamenti tra pista e pad (o vias), che sottoposti a cicli termici, potrebbero indebolirsi e rompersi. Vengono effettuati allargando la pista in prossimità del pad, come possiamo notare in figura Figura 32.

Grazie ad un plug-in esterno da aggiungere a KiCAD, questi verranno automaticamente aggiunti alla PCB, semplicemente impostando alcuni valori, mostrati in Figura 33.

Ulteriori modifiche si possono effettuare una volta generati i file Gerber, modificando a mano il loro contenuto:

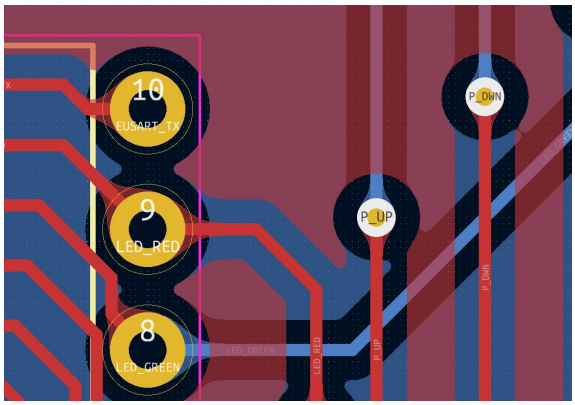

Figura 32: Teardrops sulla scheda PSE

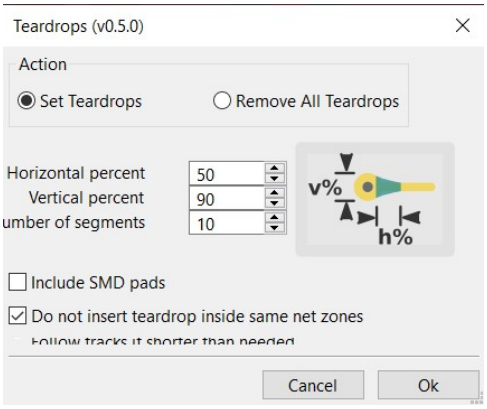

Figura 33: Valori dei teardrops

- Migliorie generali del PCB, rimozione zone di GND fastidiose in radio frequenza;
- **Merge**: unione di diverse schede, che devono però essere compatibili, (numero uguale di strati), in un unico progetto. Queste vengono collegate tramite collegamenti pre-forati detti **mouse bites**;
- **Pannellizzazione**: consiste nella creazione di un unico pannello contenente n schede uguali, necessario per l'ottimizzazione dei costi e tempi di produzione.

4.4 File Gerber

Terminato il lavoro di sbroglio è ora di generare i **file Gerber**, necessari per la produzione del PCB. I file Gerber sono file ASCII, quindi leggibili e modificabili a mano, che elencano gli elementi e definiscono:

- **Geometrie**: come lati Top e Bottom del rame, serigrafie, soldermask, etc;
- **Lavorazioni**: come forature o fresature.

In seguito vengono riportati i **file Gerber** visualizzati tramite lo strumento apposito fornito da KiCAD e il rendering 3D del progetto completo.

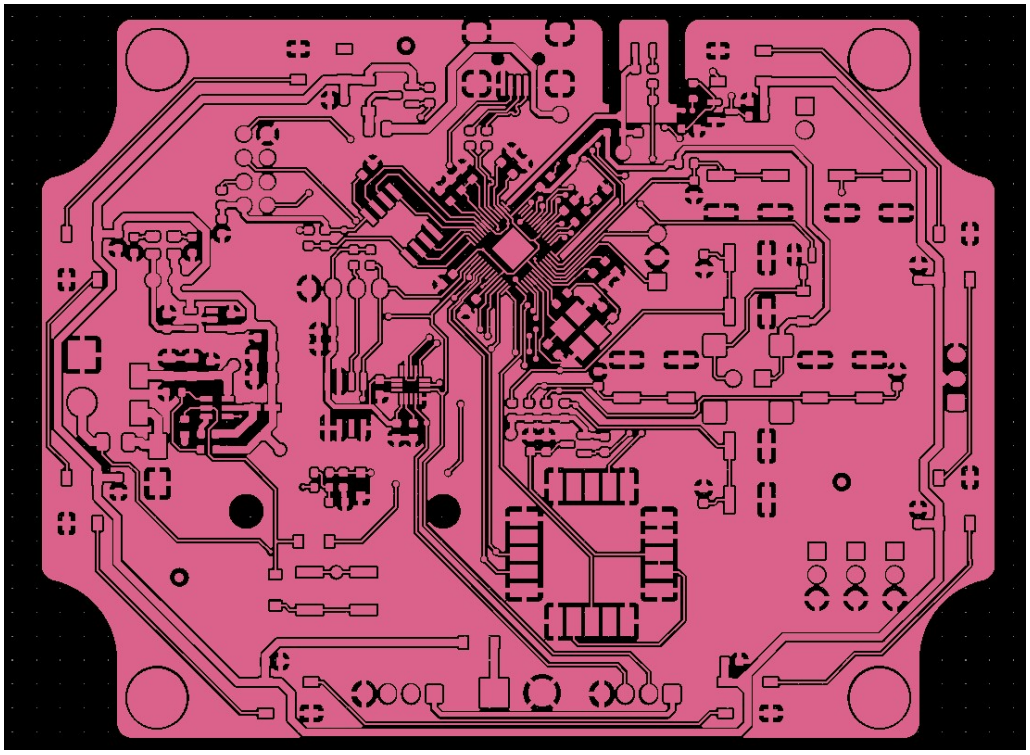

Figura 34: Rame top

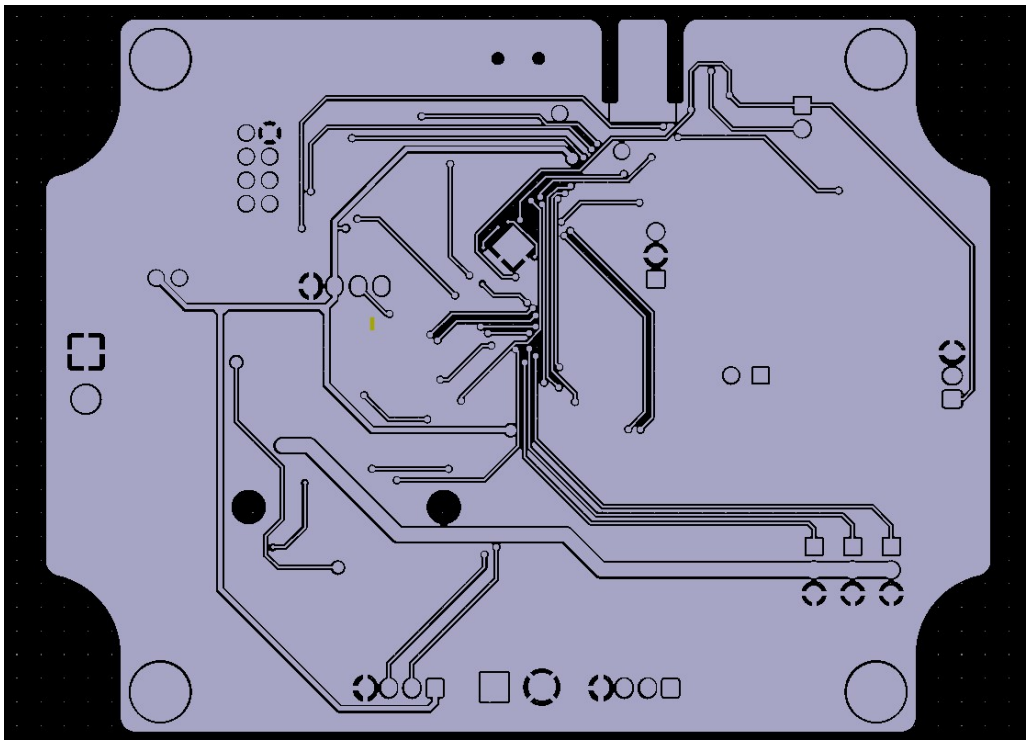

Figura 35: Rame bottom

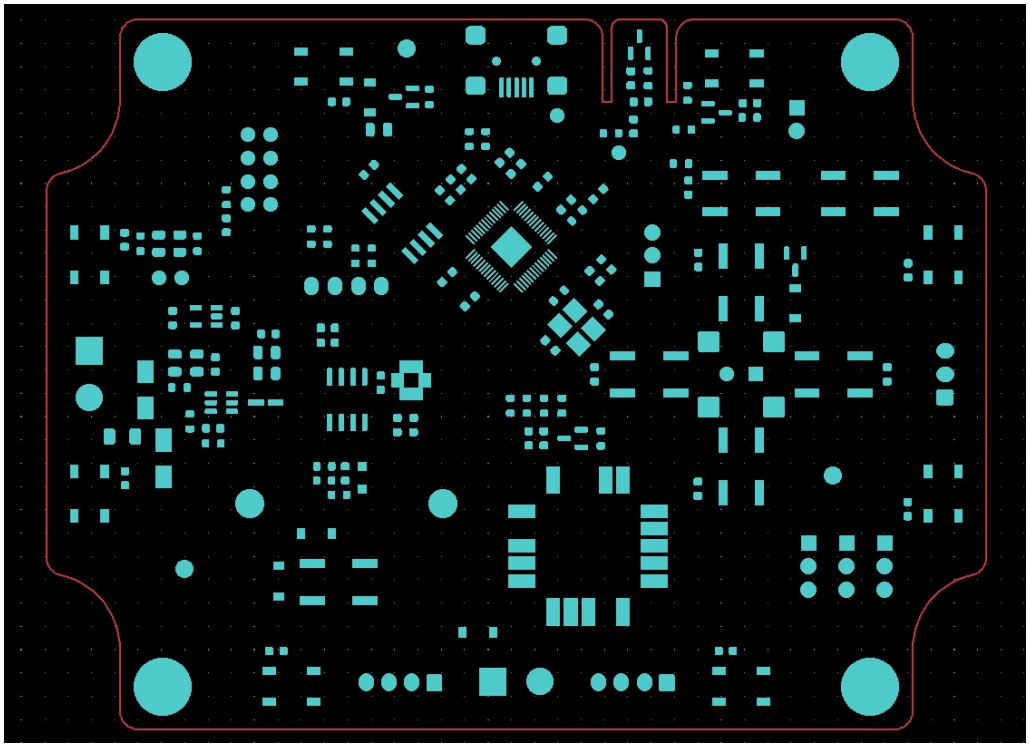

Figura 36: Soldermask top

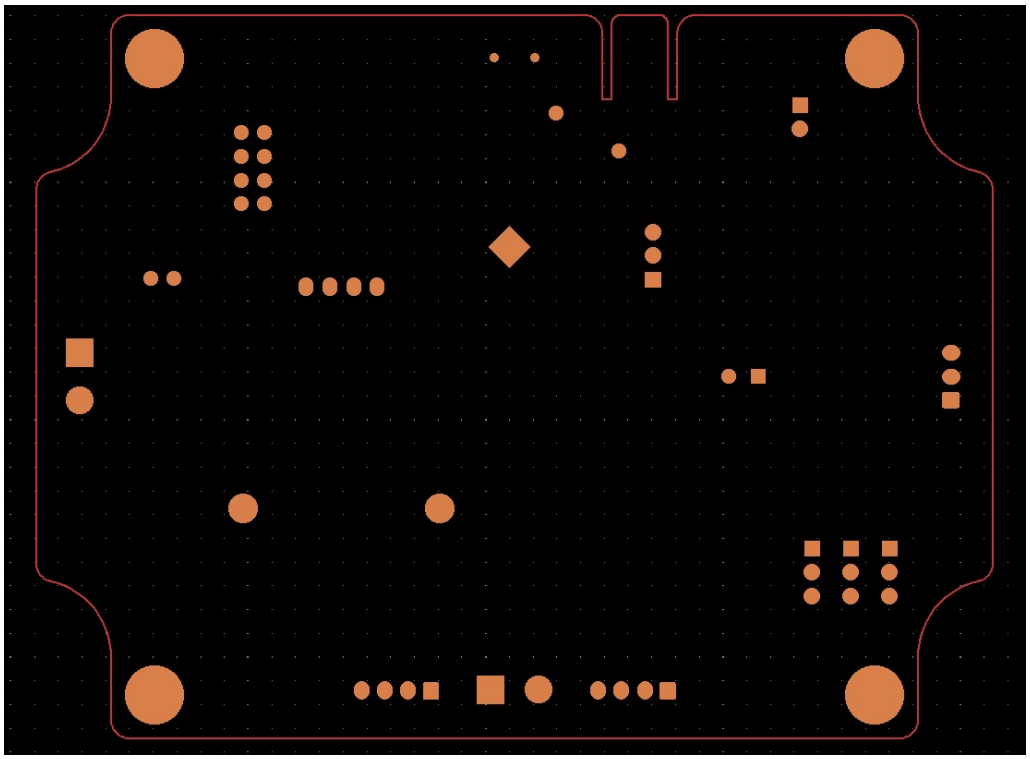

Figura 37: Soldermask bottom

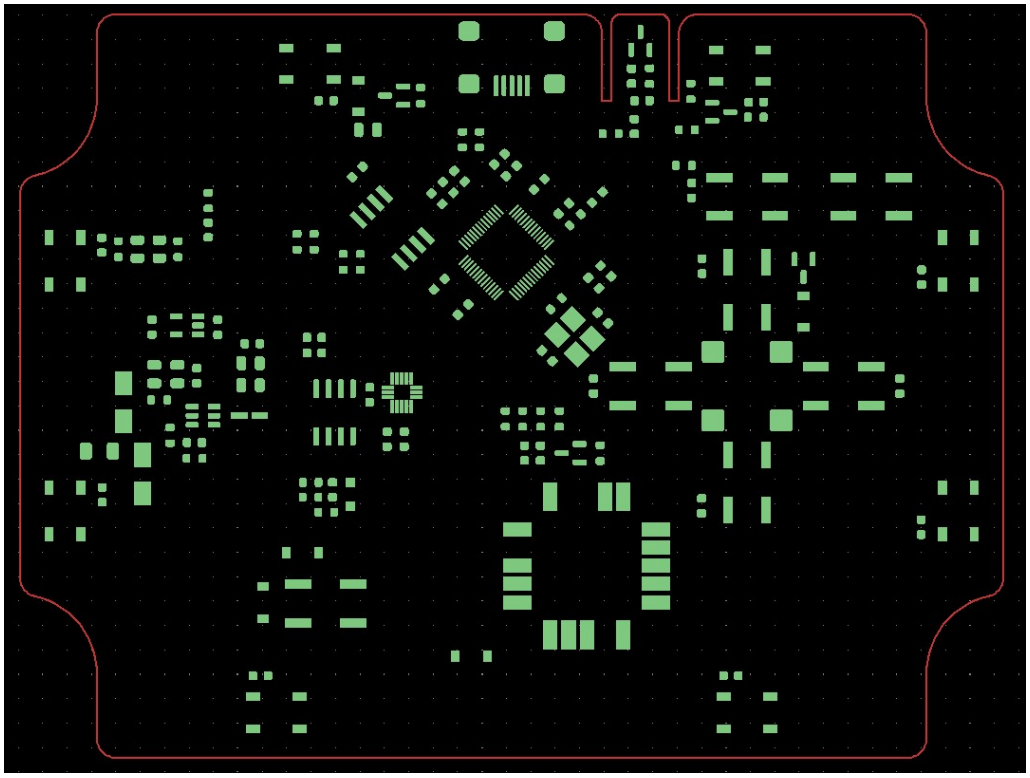

Figura 38: Solderpaste top

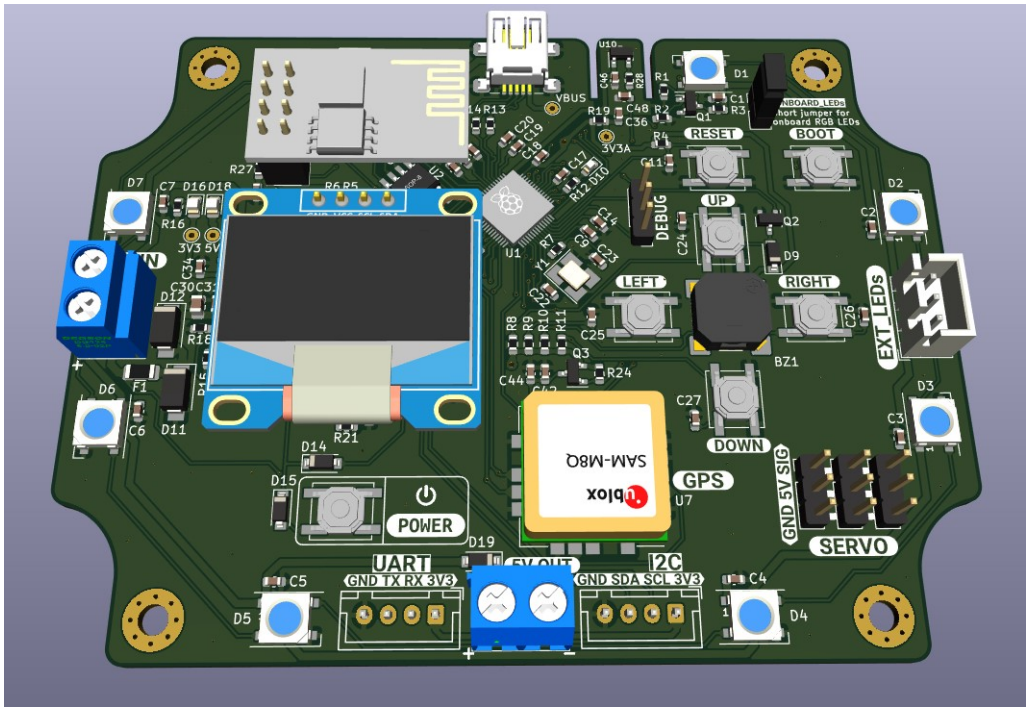

Figura 39: Rendering 3D della scheda PSE

5 Assemblaggio della PCB

L'assemblaggio di una scheda PCB consiste essenzialmente nel montare i componenti nella posizione a loro assegnata e procedere con la **brasatura** al fine di creare una connessione elettrica tra scheda e pin e assicurare una tenuta meccanica.

Definiamo la brasatura come l'unione di elementi metallici tramite un metallo d'apporto, che nei casi delle PCB è tipicamente lo **stagno**, senza la fusione dei componenti da saldare, che ne comprometterebbe il funzionamento.

La saldatura può avvenire in due modi diversi, a seconda del tipo di componente da montare:

- per **componenti THT** si procede con la brasatura dei reofori su lato bottom, dopo averli fatti passare attraverso i pad;
- per **componenti SMD** si procede applicando la pasta saldante nelle aree definite dalla solderpaste, depositando poi il componente e procedendo a scaldare tramite saldatore o pistola riscaldante.

L'assemblaggio di una PCB può essere effettuato manualmente in laboratorio, saldando uno ad uno i componenti sulla scheda, oppure in modo **automatizzato**, per la produzione in serie di grandi quantità di PCB in tempi brevi. Questo secondo metodo sfrutta le macchine **pick-and-place**, che si basano sui file Gerber e sui **file Pick and Place** di un progetto per lavorare e montare una PCB.

Nel nostro caso, l'assemblaggio della scheda PSE è stato fatto manualmente tramite l'utilizzo di un **forno a rifusione**. Alle schede da assemblare è stato prima di tutto applicata la **pasta saldante** utilizzando una maschera metallica ricavata dal layer di solderpaste del gerber. Successivamente sono stati messi in posizione i componenti, facendoli aderire alla pasta. La scheda è stata infine messa nel forno, programmato per eseguire la curva di riscaldamento necessaria alla corretta saldatura dei componenti

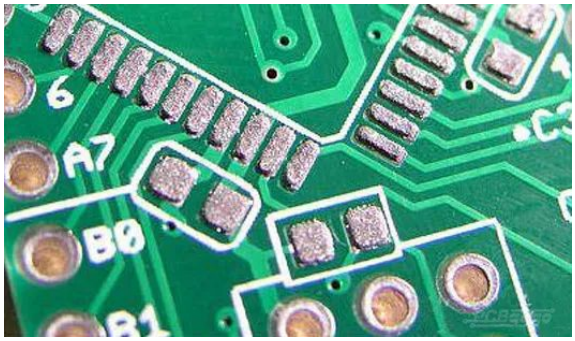

Figura 40: Pasta saldante applicata alla PCB

Figura 41: Forno a rifusione

6 Programmazione firmware

La funzionalità che abbiamo voluto attribuire alla scheda è stata quella di un sistema di acquisizione e memorizzazione dei dati relativi alla posizione, alla temperatura e all'accelerazione. I dati memorizzati possono successivamente essere visualizzati dall'utente collegando la scheda al PC, che verrà riconosciuta come unità Mass Storage. Un file .txt contenente le stringhe di dati, potrà quindi essere aperto e utilizzato dall'utente. Sono inoltre stati implementati dei menù per la scelta della durata e frequenza del campionamento, oltre che ai dati che devono essere salvati nella sessione. Una volta fatto iniziare il campionamento, dei led indicheranno la fase di acquisizione dati, che terminerà dopo il tempo indicato con una suoneria.

6.1 IDE

La programmazione della scheda è stata effettuata tramite Raspberry Pi Pico SDK (Software Development Kit), installato direttamente su Visual Studio Code (VSCode). L'SDK mette a disposizione diverse funzioni, librerie ed esempi in linguaggio C per la programmazione di schede basate sul microcontrollore RP2040. Dopo aver fatto il setup dell'ambiente su VSCode, possiamo creare o prendere dagli esempi un progetto Pico SDK. Questo è caratterizzato da un Makefile, uno speciale file di configurazione che ha il compito di:

- esprimere le dipendenze del progetto verso altri file/directory;
- indicare cosa deve fare il sistema per aggiornare il target se uno dei file nella dependency list è stato modificato;
- fare in modo che tutte le regole descritte nel Makefile vengano applicate automaticamente dal sistema.

Facendo la build del progetto, grazie al Makefile, viene quindi generato il file .uf2, necessario per la programmazione della scheda. Tenendo premuto il pulsante di boot è infatti possibile copiare questo file all'interno della memoria flash della scheda e quindi programmare il microcontrollore.

6.2 Struttura del codice

Il firmware è basato su una serie di macchine a stati finiti (FSM - finite state machine) innestate, le quali chiamano la funzione associata allo stato in cui ogni macchina si trova, determinando così la funzionalità della scheda. La macchina a stati più esterna chiama via via la macchina a stati più interna in modo gerarchico, fino a quando la profondità di chiamata corrisponde al valore di una variabile chiamata "depth". La variabile gestisce la profondità, interrompendo così la serie di chiamate delle funzioni delle FSM innestate. La modifica dello stato corrente e della profondità viene invece gestita dai pulsanti tramite interrupt, ai quali viene applicato un debounce software.

Ogni pulsante, in base allo stato attuale delle FSM e della variabile "depth", modifica lo stato e la profondità, determinando il comportamento della scheda alla prossima chiamata della FSM principale. In particolare, la prima macchina a stati è quella che permette l'inizializzazione di tutto il sistema. Questa FSM a sua volta chiama un'altra FSM che gestisce il menu principale, offrendo la possibilità di entrare in tre stati: settings, sampling e usb.

```
int main() {
    //lasciamo che l'applicazione interrupt based faccia il suo corso
    while(1)
    {
        if(settings.current_state_menu < NUM_STATES_MENU){
            (*fsm_INIT_DEFAULT[settings.current_state_menu].state_function)();     //call to FSM functions
        }
        else{
            /* error */
        }
        while(1){
            if(interrompi == true){
                interrompi = false;
                break;
            }
        }
    }

}
```

Figura 42: Gestione della FSM principale nel main

6.2.1 Stato settings

Il primo menu (settings) permette all'utente di modificare le impostazioni del sampling a proprio piacimento. Accedendo a questo menu, si entra in un'ulteriore macchina a stati innestata che consente di scegliere le modifiche da apportare al sensore di temperatura, all'accelerometro, al GPS o alla velocità di campionamento. Nel caso si selezioni l'accelerometro o il GPS, si accede a un'altra macchina a stati chiamata dalla precedente, nella quale è possibile effettivamente scegliere l'impostazione da cambiare e quindi modificarla. Nei restanti due menù, invece, si accede direttamente allo stato di modifica della frequenza di campionamento e dell'abilitazione/disabilitazione del sensore di temperatura. Lo stato della modifica delle impostazioni viene gestito tramite l'utilizzo di due variabili: una "effettiva" e una temporanea. La variabile "effettiva" viene modificata solo nel momento in cui si conferma la modifica effettuata, mentre la seconda viene modificata in tempo reale dalle modifiche apportate tramite il menu. Nel caso in cui le impostazioni non vengano confermate, la variabile temporanea viene sostituita da una copia della variabile "reale".

6.2.2 Stato sampling

Il menu di sampling permette di visualizzare la frequenza di campionamento e il numero di campionamenti già effettuati. Durante il processo di campionamento, i dati raccolti vengono elaborati e salvati in una sequenza di caratteri, che verrà successivamente utilizzata per creare il file di trasferimento. I possibili valori sono ottenuti dalla giusto utilizzo dei seguenti sensori:

- Il modulo GPS una volta alimentato trasmette via UART in modo continuo i dati ricevuti dai satelliti in quel momento disponibili nel suo raggio di azione. Sarà quindi il microcontrollore, che in base al valore della variabile di campionamento, leggerà i dati provenienti dal modulo GPS utilizzando funzioni per la gestione della comunicazione UART;

```
void check_gps(struct TGPS *GPS)
{
    static unsigned char Line[100];
    static int Length=0;
    while (uart_is_readable(uart1))
    {
        char Character;
        Character = uart_getc(uart1);
        if (Character == '$')
        {
            Line[0] = Character;
            Length = 1;
        }
        else if (Length > 90)
        {
            Length = 0;
        }
        else if ((Length > 0) && (Character != '\r'))
        {
            Line[Length++] = Character;
            if (Character == '\n')
            {
                Line[Length] = '\0';
                printf("%s", Line);
                ProcessLine(GPS, Line, Length);
                Length = 0;
            }
        }
    }
}
```

Figura 43: Funzione di acquisizione dati dal GPS

- L'accelerometro si interfaccia al microcontrollore tramite protocollo I2C. Ad una frequenza stabilita dalla variabile di sampling, il master, che in questo caso è il micro, invierà un segnale di start indirizzandolo all'accelerometro, che a sua volta, una volta terminato il campionamento, risponderà inviando i dati raccolti. Questi verranno infine letti tramite apposite funzioni per la gestione della comunicazione I2C e memorizzati;

```
77  void get_accelerometer(struct Data_storage *Data){
78      float x_acceleration;
79      float y_acceleration;
80      float z_acceleration;
81
82      // Start reading acceleration registers for 2 bytes
83      i2c_write_blocking(i2c_default, ADDRESS, &REG_X_MSB, 1, true);
84      i2c_read_blocking(i2c_default, ADDRESS, bufacc, 2, false);
85      x_acceleration = mma8451_convert_accel(bufacc[0] << 6 | bufacc[1] >> 2);
86      Data->x_acceleration = x_acceleration;
87
88      i2c_write_blocking(i2c_default, ADDRESS, &REG_Y_MSB, 1, true);
89      i2c_read_blocking(i2c_default, ADDRESS, bufacc, 2, false);
90      y_acceleration = mma8451_convert_accel(bufacc[0] << 6 | bufacc[1] >> 2);
91      Data->y_acceleration = y_acceleration;
92
93      i2c_write_blocking(i2c_default, ADDRESS, &REG_Z_MSB, 1, true);
94      i2c_read_blocking(i2c_default, ADDRESS, bufacc, 2, false);
95      z_acceleration = mma8451_convert_accel(bufacc[0] << 6 | bufacc[1] >> 2);
96      Data->z_acceleration = z_acceleration;
97  }
```

Figura 44: Funzione di acquisizione dati dall'accelerometro

- Il sensore di temperatura, a differenza dei precedenti, si interfaccia al microcontrollore tramite un pin analogico e non con un protocollo di comunicazione particolare. All'istante in cui i dati devono essere raccolti, viene dato un segnale di start all'ADC interno al microcontrollore, che campiona il livello di tensione proveniente dal sensore di temperatura e lo converte in valore digitale. Questo viene poi utilizzato via codice per calcolare la temperatura tramite le formule specifiche e il dato viene infine salvato.

```
void get_temperature(struct Data_storage *Data){
    // 12-bit conversion, assume max value == ADC_VREF == 3.3 V
    const float conversion_factor = 3.3f / (1 << 12);
    uint16_t result = adc_read();
    float temperature = (result*conversion_factor - 0.5)/0.01;
    Data->temperature = temperature;
}
```

Figura 45: Funzione di acquisizione dati dal sensore di temperatura

6.2.3 Stato USB

Il menù di trasferimento USB (usb transfer) consente di trasferire i dati salvati durante il sampling su un dispositivo di memorizzazione di massa configurando la scheda in modalità device. Una volta entrati nel menu tramite i pulsanti, il trasferimento USB viene gestito da una libreria chiamata tinyusb. Questa libreria, tramite interrupt, gestisce la comunicazione con il dispositivo host come dispositivo di memorizzazione di massa (mass storage device). Tra le modifiche più importanti ed essenziali apportate a questa libreria vi è la modifica del file di testo da visualizzare e, di conseguenza, del file system che rende possibile tale visualizzazione. Quando si accede allo stato di trasferimento, viene richiamata una specifica funzione che si occupa di elaborare la stringa contenente i dati acquisiti durante il campionamento. La funzione consente quindi di inizializzare il vettore che rappresenta il file system basandosi sulla stringa elaborata. Questo processo assicura che il vettore del file system sia correttamente inizializzato con i dati attuali. Successivamente, viene avviata la connessione USB come dispositivo di archiviazione di massa utilizzando la libreria di supporto e, sempre per mezzo di essa, viene anche gestito il trasferimento del file e la chiusura della connessione.

7 Allegati

7.1 BOM

Tabella 1: BOM della scheda PSE2022-23

Ite	Qt	Reference(s)	Value	LibPart	Footprint
1	1	BZ1	Buzzer	Device:Buzzer	_PSE:MLT-8530
2	31	C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C16, C17, C18, C24, C25, C26, C27, C29, C32, C38, C40, C41, C43, C44, C46, C47	100n	Device:C	Capacitor_SMD:C_0603_1608Metric
3	3	C19, C20, C36	2.2u	Device:C	Capacitor_SMD:C_0603_1608Metric
4	1	C21	DNP 100n	Device:C	Capacitor_SMD:C_0603_1608Metric
5	2	C22, C23	18p	Device:C	Capacitor_SMD:C_0603_1608Metric
6	2	C28, C45	4.7u	Device:C	Capacitor_SMD:C_0603_1608Metric
7	2	C30, C31	10u	Device:C	Capacitor_SMD:C_0805_2012Metric
8	2	C33, C34	1u	Device:C	Capacitor_SMD:C_0603_1608Metric
9	1	C35	100n	Device:C_Small	Capacitor_SMD:C_0603_1608Metric
10	2	C37, C39	22u	Device:C	Capacitor_SMD:C_0805_2012Metric
11	1	C42	4u7	Device:C	Capacitor_SMD:C_0603_1608Metric
12	1	C48	1u 6V3	Device:C	Capacitor_SMD:C_0603_1608Metric
13	8	D1, D2, D3, D4, D5, D6, D7, D8	WS2812B	LED:WS2812B	LED_SMD:LED_WS2812B_PLCC4_5.0x5.0mm_P3.2mm
14	3	D9, D14, D15	1N4148	Diode:1N4148	Diode_SMD:D_SOD-123
15	1	D10	LED_alive	Device:LED	LED_SMD:LED_0603_1608Metric
16	1	D11	TVS VBR 17V	Diode:1.5KExxA	Diode_SMD:D_SMA
17	1	D12	S1B	Device:D	Diode_SMD:D_SMA
18	2	D13, D19	RBR1VWM30ATFTR	Device:D_Schottky	Diode_SMD:D_SOD-123
19	1	D16	3V3_LED	Device:LED	LED_SMD:LED_0805_2012Metric
20	1	D17	PDZ3.3B.115	Device:D_Zener	Diode_SMD:D_SOD-
21	1	D18	5V_LED	Device:LED	LED_SMD:LED_0805_2012Metric
22	1	F1	2016L05MR	Device:Polyfuse	Fuse:Fuse_1206_3216Metric
23	1	FB1	FerriteBead	Device:FerriteBead	Capacitor_SMD:C_0805_2012Metric
24	4	H1, H2, H3, H4	MountingHole	Mechanical:MountingHole	MountingHole:MountingHole_3.2mm_M3_Pad_Via
25	1	J1	WS2812B	Connector_Generic:Conn_01x03	Connector_JST:JST_XH_B3B-XH-A_1x03_P2.50mm_Vertical
26	1	J2	Conn_01x03	Connector_Generic:Conn_01x03	Connector_PinHeader_2.54mm:PinHeader_1x03_P2.54mm_Vertical
27	1	J3	Vin	Connector:Screw_Terminal_01x02	TerminalBlock:TerminalBlock_bornier-2_P5.08mm

28	1	J4	USB_B_Mini	Connector:USB_B_Mini	_PSE:USB mini
29	1	J5	Vout 5V	Connector:Screw_Terminal_01x02	TerminalBlock:TerminalBlock_bornier-2_P5.08mm
30	1	J6	I2C_3V3	Connector_Generic:Conn_01x04	Connector_JST:JST_XH_B4B-XH-A_1x04_P2.50mm_Vertical
31	1	J7	UART0_3V3	Connector_Generic:Conn_01x04	Connector_JST:JST_XH_B4B-XH-A_1x04_P2.50mm_Vertical
32	1	J8	SERVO_1	Connector_Generic:Conn_01x03	Connector_PinHeader_2.54mm:PinHeader_1x03_P2.54mm_Vertical
33	1	J9	SERVO_2	Connector_Generic:Conn_01x03	Connector_PinHeader_2.54mm:PinHeader_1x03_P2.54mm_Vertical
34	1	J10	SERVO_3	Connector_Generic:Conn_01x03	Connector_PinHeader_2.54mm:PinHeader_1x03_P2.54mm_Vertical
35	1	JP1	EXT_LEDs	Jumper:Jumper_2_Open	Connector_PinHeader_2.54mm:PinHeader_1x02_P2.54mm_Vertical
36	1	L1	2.2u	Device:L	_PSE:IHLP1212BZER2R2M11
37	1	NT1	gnda-gnd	Device:NetTie_2	NetTie:NetTie-2_SMD_Pad0.5mm
38	1	Q1	IRLML0060TRPBF	Transistor_FET:IRLML2060	Package_TO_SOT_SMD:SOT-23
39	1	Q2	PDTC114YT	Transistor_BJT:DTC114	Package_TO_SOT_SMD:SOT-23
40	2	Q3, Q4	IRLML9301	Transistor_FET:IRLML9301	Package_TO_SOT_SMD:SOT-23
41	5	R1, R3, R6, R7, R22	1k	Device:R	Resistor_SMD:R_0603_1608Metric
42	1	R2	100k	Device:R	Resistor_SMD:R_0603_1608Metric
43	3	R4, R17, R27	10K	Device:R	Resistor_SMD:R_0603_1608Metric
44	4	R5, R15, R24,	10k	Device:R	Resistor_SMD:R_0603_1608Metric
45	4	R8, R9, R10,	470k	Device:R	Resistor_SMD:R_0603_1608Metric
46	2	R12, R28	470	Device:R	Resistor_SMD:R_0603_1608Metric
47	2	R13, R14	27	Device:R	Resistor_SMD:R_0603_1608Metric
48	2	R16, R23	510	Device:R	Resistor_SMD:R_0603_1608Metric
49	1	R18	1.8k	Device:R	Resistor_SMD:R_0603_1608Metric
50	1	R19	200	Device:R	Resistor_SMD:R_0603_1608Metric
51	1	R20	150k	Device:R	Resistor_SMD:R_0603_1608Metric
52	1	R21	56k	Device:R	Resistor_SMD:R_0603_1608Metric
53	2	R25, R26	4.7k	Device:R	Resistor_SMD:R_0603_1608Metric
54	1	SW1	RESET_SW	Switch:SW_Push	Button_Switch_SMD:SW_SPST_EVQP
55	1	SW2	PROG_SW	Switch:SW_Push	Button_Switch_SMD:SW_SPST_EVQP
56	1	SW3	SW_U	Switch:SW_Push	Button_Switch_SMD:SW_SPST_EVQP
57	1	SW4	SW_L	Switch:SW_Push	Button_Switch_SMD:SW_SPST_EVQP
58	1	SW5	SW_R	Switch:SW_Push	Button_Switch_SMD:SW_SPST_EVQP
59	1	SW6	SW_D	Switch:SW_Push	Button_Switch_SMD:SW_SPST_EVQP
60	1	SW7	Power_button	Switch:SW_Push	Button_Switch_SMD:SW_SPST_EVQP
61	1	TP1	VBUS	Connector:TestPoint	TestPoint:TestPoint_THTPad_D1.5mm_Drill0.7mm
62	1	TP2	5V	Connector:TestPoint	TestPoint:TestPoint_THTPad_D1.5mm_Drill0.7mm
63	1	TP3	3V3	Connector:TestPoint	TestPoint:TestPoint_THTPad_D1.5mm_Drill0.7mm

64	1	TP4	3V3A	Connector:TestPoint	TestPoint:TestPoint_THTPad_D1.5mm_Drill0.7mm
65	1	U1	RP2040	_PSE 2022:RP2040_PSE	_PSE:QFN40P700X700X90-57N-D
66	1	U2	W25Q16JVUSSQ	2022:W25Q16JVUSSQ	_MIA footprint:SOIC8-150mils
67	1	U3	AT24C256	altro:AT24C256	Package_SO:SOIC-
68	1	U4	oled_0.96"_128x64px	_MIA:oled_0.96_128x64px_PSE	_PSE:128x64OLED
69	1	U5	XC6705A331MR-G	2022:XC6705A331MR-G	Package_TO_SOT_SMD:SOT-23-5
70	1	U6	TPS563201	Regulator_Switching:TPS563200	Package_TO_SOT_SMD:SOT-23-6
71	1	U7	SAM-M8Q	RF_GPS:SAM-M8Q	_PSE:SAM_M8Q
72	1	U8	MMA8451Q	MMA8451Q:MMA8451	_PSE:QFN-16
73	1	U9	ESP-01	esp8266:ESP-01	_PSE:ESP-01
74	1	U10	TC1047A	Sensor_Temperature:TC1047AxNB	Package_TO_SOT_SMD:SOT-23
75	1	Y1	12MHz	Device:Crystal_GND24	Crystal:Crystal_SMD_3225-4PC69:F84in_3.2x2.5mm_HandSolde

Riferimenti bibliografici

[1] KiCAD. Getting started in KiCAD, 2021.