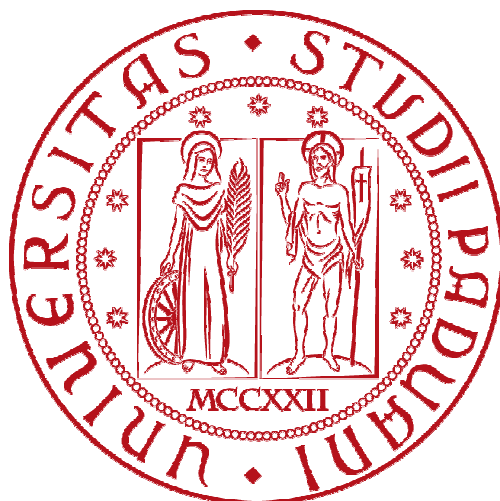


UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA ELETTRONICA



Tesi di Laurea Magistrale in Ingegneria Elettronica

PROGETTAZIONE E SVILUPPO DI
UN SISTEMA ELETTRONICO PER LA
CARATTERIZZAZIONE DI BIOSENSORI ORGANICI
MEDIANTE STIMOLAZIONE E RILEVAZIONE DELLA
RISPOSTA CELLULARE

Relatore: Prof. Andrea Cester

Correlatore: Dott. Nicola Wrachien

Laureando: Andrea Bianchi

ANNO ACCADEMICO 2015 – 2016

ABSTRACT

Negli ultimi anni lo sviluppo dell'elettronica organica ha portato ad esplorare nuove tecnologie anche per quanto riguarda i biosensori.

In questa tesi viene preso in considerazione un particolare tipo di sensore OTFT, utilizzato per la stimolazione e la rilevazione di segnali neuronali. Dato che la teoria sul funzionamento di questi dispositivi è ancora in fase di studio, sono necessarie diverse misure per la sua caratterizzazione e modellizzazione.

Ci si pone quindi, l'obiettivo di progettare e sviluppare un sistema elettronico in grado di fornire impulsi di ampiezza e forma variabile che simulino l'attività neuronale e di leggere la risposta del sensore. La circuiteria della scheda deve essere pilotata da un microcontrollore in grado di far funzionare il sistema sia in modalità stand-alone sia in modalità remota (interfacciata a un PC ed eventualmente altra strumentazione).

Il progetto dello schematico del circuito è stato eseguito tramite software OrCAD Capture, mentre per il layout fisico della scheda si è utilizzato il tool PCB Editor. Il firmware del microcontrollore utilizzato, un AVR AT32UC3A3256, è stato progettato tramite il software dedicato Atmel Studio in linguaggio C.

Sono state implementate tre tipi di misurazioni, nella prima viene letta la risposta del sensore ad un gradino di tensione, la seconda si tratta di una ciclovoltammetria, mentre la terza è una vera e propria simulazione del sistema cellulare nella quale il sensore viene stimolato da un impulso identico al potenziale d'azione del neurone.

Infine sono state eseguite diverse prove di misura su dei dispositivi presenti in laboratorio.

INDICE

INTRODUZIONE	7
CAPITOLO 1 – BIOSENSORI ORGANICI	9
1.1 SVILUPPO E TECNOLOGIE.....	9
1.1.1 OTFT – Organic Thin Film Transistor	13
1.2 MISURE DI CARATTERIZZAZIONE	16
1.2.1 SAMPLING	16
1.2.2 CICLOVOLTAMMETRIA.....	17
1.2.3 FIRING & SENSING	18
CAPITOLO 2 – PROGETTAZIONE CIRCUITO	21
2.1 PROGETTAZIONE SCHEMATICO.....	21
2.1.1 SMU PER LA MISURA DI CORRENTE	22
2.1.2 SMU PER LA MISURA DI TENSIONE	28
2.1.3 POLARIZZAZIONE DUT	31
2.1.4 ADC – Analog/Digital Converter	32
2.1.5 DAC – Digital/Analog Converter	34
2.1.6 SPI – Serial Peripheal Interface.....	35
2.1.7 MICROCONTROLLORE	37
2.1.8 PERIFERICHE.....	39
2.1.9 ALIMENTAZIONI.....	46
2.2 PROGETTAZIONE LAYOUT.....	51
CAPITOLO 3 – PROGRAMMAZIONE FIRMWARE	55
3.1 STRUTTURA FIRMWARE	56
3.2 DRIVER DISPOSITIVI	61
3.2.1 adc.c	62
3.2.2 dac.c	63
3.2.3 flash.c	64
3.2.4 oled.c	64
3.2.5 port_expander.c.....	65
3.2.6 ps2.c	66
3.2.7 rtc.c	66

3.2.8 sdram.c	66
3.3 IMPLEMENTAZIONE MISURE	67
3.4 COMANDI.....	77
CAPITOLO 4 – MISURAZIONI E RISULTATI.....	79
4.1 MISURE SAMPLING.....	80
4.1 MISURE CICLOVOLTAMMETRIA	81
4.2 MISURE FIRING & SENSING.....	85
CONCLUSIONI	89
BIBLIOGRAFIA	91

INTRODUZIONE

Questo elaborato è suddiviso in quattro capitoli, che seguono il processo di esecuzione della tesi.

Nel primo capitolo si introducono i biosensori, ovvero i dispositivi per i quali è stato necessario tutto il lavoro. Lo sviluppo tecnologico nei vari anni ha portato a sensori sempre meno invasivi e più biocompatibili, per cercare di risolvere le varie problematiche dovute all'introduzione di un corpo estraneo all'interno di un organismo vivente. Questa evoluzione ha portato allo studio di particolari biosensori basati su dei transistor organici, gli OTFT.

Nel secondo capitolo si passa alla progettazione circuitale dello strumento di misura. La prima parte riguarda la stesura dello schematico con le varie scelte progettuali, mentre nella seconda si discute il layout e la disposizione dei componenti nella scheda.

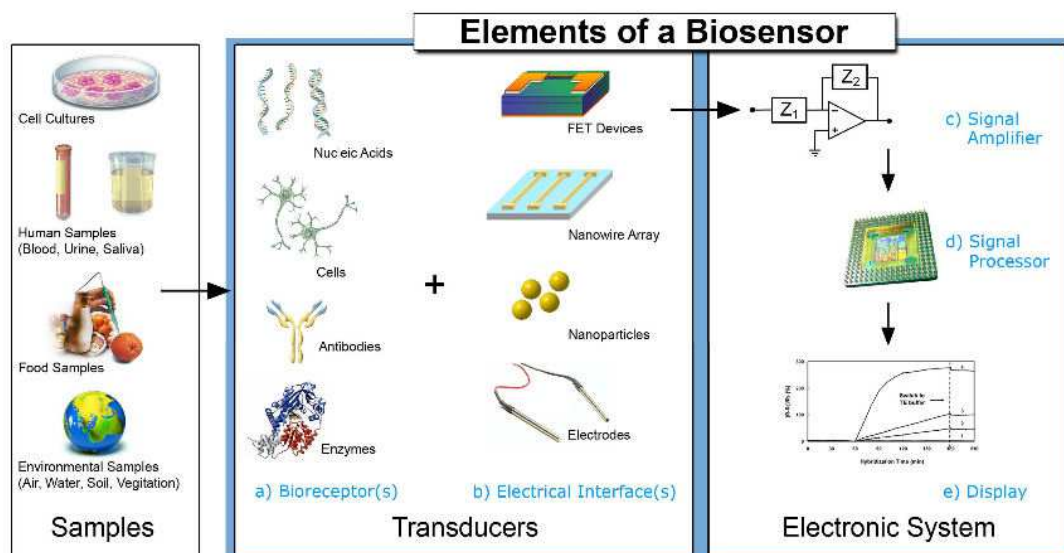
Il terzo capitolo riguarda il firmware. Si spiega la struttura che è a "livelli", dal più basso dove vengono configurati i vari segnali, al più alto dove vengono implementate le misurazioni, viene illustrato il diagramma di flusso, e i vari driver dei singoli componenti interfacciati con il microcontrollore. Una sezione riguarda i comandi disponibili per le varie funzioni che la scheda è in grado di eseguire.

Nel quarto, e ultimo capitolo, vengono riportati i risultati delle diverse misure effettuate.

CAPITOLO 1

BIOSENSORI ORGANICI

Un biosensore è un particolare dispositivo che utilizza un elemento sensibile biologicamente attivo a contatto con un trasduttore e da una parte elettronica. L'elemento biologico interagisce con il substrato da analizzare e il trasduttore converte il segnale biologico in un segnale elettrico misurabile.



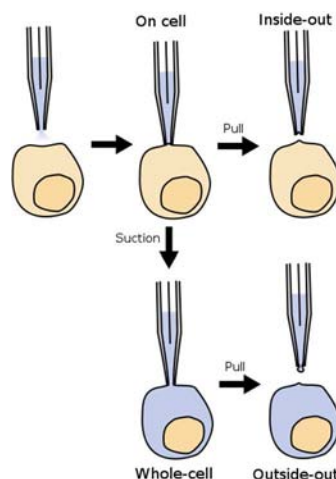
1.1 Elementi di un biosensore

1.1 SVILUPPO E TECNOLOGIE

Nel corso degli anni, le neuro-scienze hanno dedicato sempre più attenzione alla comprensione delle cause dell'alterazione nell'attività bioelettrica delle cellule neuronali, con lo scopo di diagnosticare e ripristinare le funzionalità cerebrali perse a seguito di incidenti o malattie neuro-degenerative come Alzheimer e Parkinson. È quindi necessario lo sviluppo di avanzate tecniche di indagine per la rilevazione di segnali neuronali.

Nel tempo c'è stato uno sviluppo delle tecnologie che permettessero un'interfaccia neuronale, necessaria per poter studiare da vicino i segnali nervosi, stimolando o registrando segnali dal tessuto nervoso stesso. Le interfacce neuronali sono realizzate con elettrodi impiantabili (quindi a tutti gli effetti degli agenti esterni al corpo) che trasducono segnali bioelettrici in segnali elettrici e viceversa. Un'interfaccia ideale, per questo motivo, deve essere biocompatibile e affidabile, deve cioè essere il più possibile integrata nell'organismo che la ospita e deve poter funzionare in modo corretto per molto tempo. Le specifiche che deve rispettare riguardano soprattutto la temperatura massima che può raggiungere, la quale non deve superare quella dell'ambiente circostante, e avere una dimensione tale da non interferire con i movimenti o altre attività fisiologiche. Gli elettrodi neuronali si sono evoluti quindi in termini di materiali, chimica, forma e dimensione per modificare le proprietà elettriche e meccaniche, per raggiungere la biocompatibilità con il tessuto nervoso.

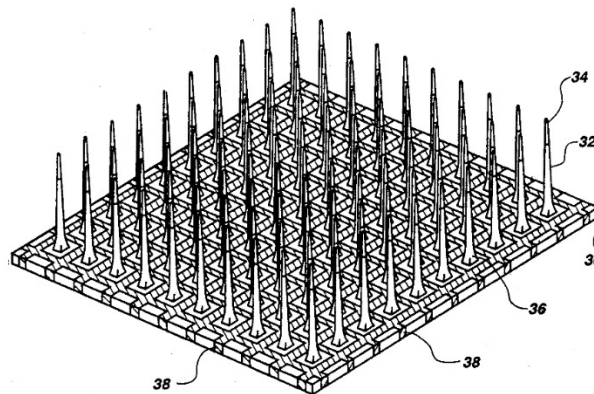
Le prime indagini sono state svolte a partire dal 1976 grazie alla tecnica della patch clamp, dove l'estremità di una pipetta di vetro con un diametro di 1µm viene fatta aderire perfettamente a una membrana cellulare e in questo modo è possibile manipolare chimicamente o elettricamente i canali ionici in modo da ricavarne le proprietà attraverso lo studio della corrente che passa nei canali ionici stessi. Questa tecnica però, offre informazioni di un solo neurone, è invasiva, e risulta valida solo per esperimenti in vitro.



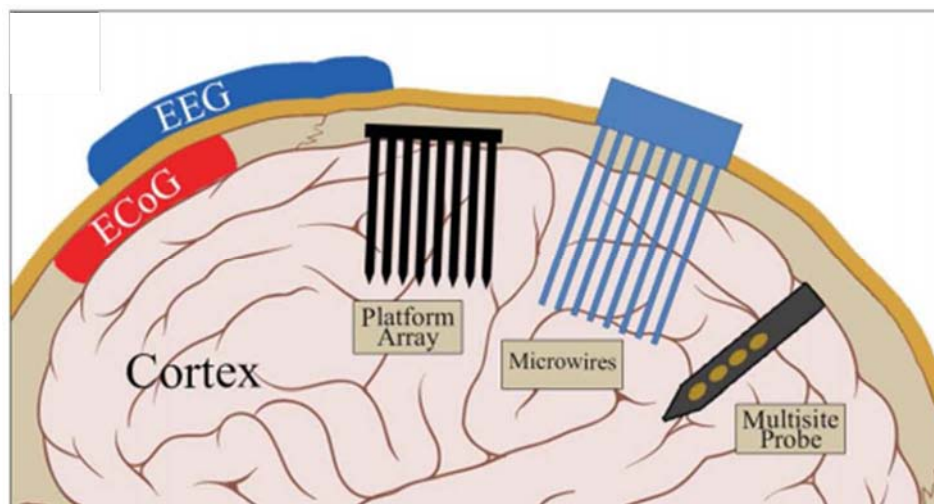
1.2 Patch-clamp

Già a partire dagli anni '70 però, si è iniziato ad utilizzare una tecnologia che risolvesse questi problemi, la tecnologia MEAs (Multi-Electrode Arrays). Questi MEAs sono dispositivi formati da diverse micropunte che interagiscono col segnale neuronale. Il problema principale è che

richiedono una connessione transcutanea che potrebbe causare complicazioni chirurgiche, e si possono deformare durante l'impiantazione.



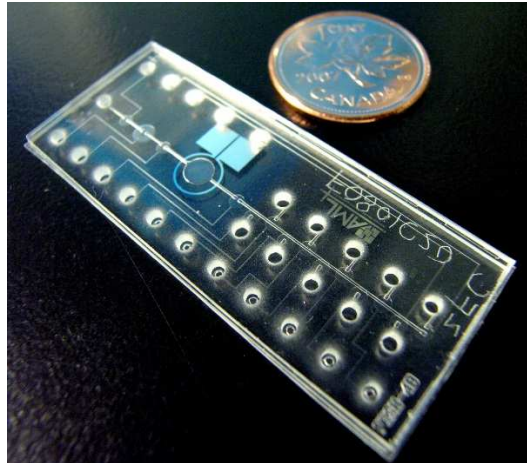
1.3 Esempio di MEAs



1.4 MEAs nel cervello

Si è passati poi alla tecnologia MEMS (Micro Electro-Mechanical System) che permettono un aumento del numero delle sonde ed essendo dispositivi più piccoli risolvono il problema dei MEAs.

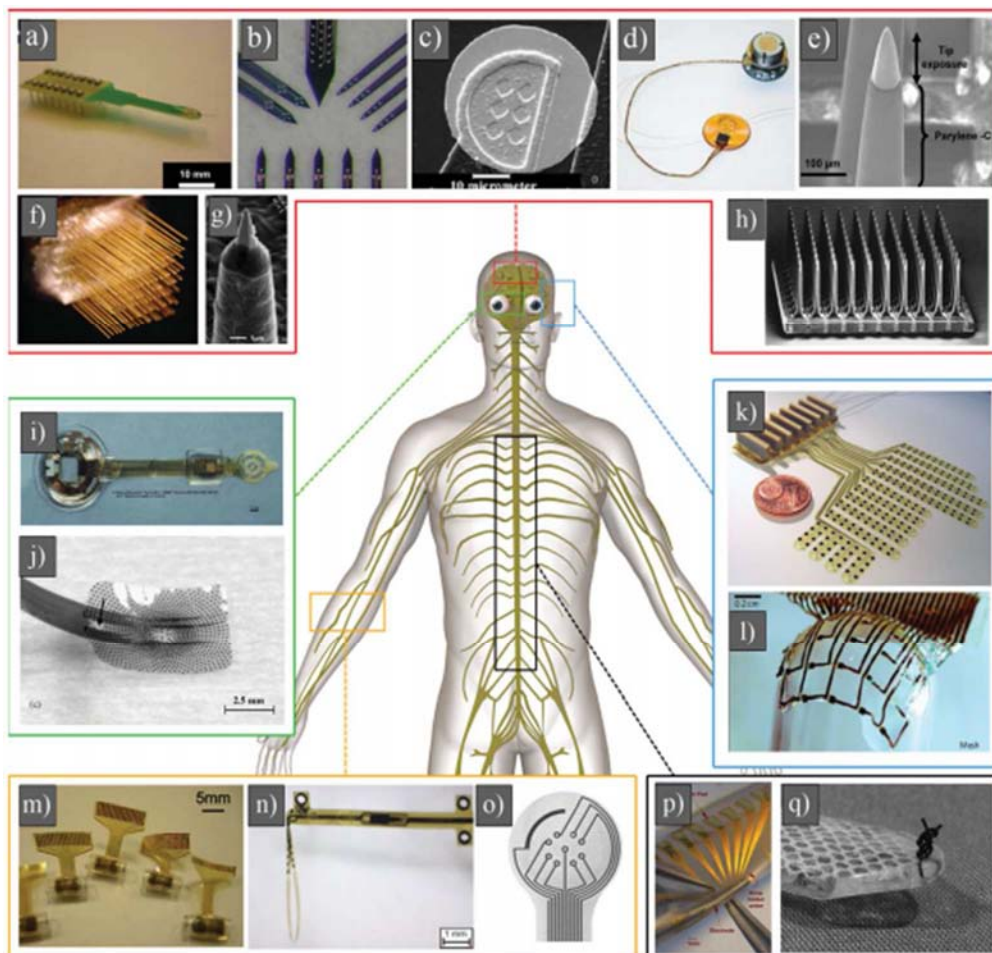
Tutte queste tecnologie però hanno un problema fondamentale, si tratta di elettrodi completamente rigidi che non riescono ad adattarsi alla superficie morbida e curva del tessuto nervoso.



1.5 MEMS

Per fare ciò si è passati ad utilizzare elettrodi ultrasottili in polyimide su film di seta bioassorbibile, ma anche questa tecnologia, come le altre, comporta un certo grado di infiammazione del tessuto.

Per ridurre questi traumi è necessario quindi modificare la superficie dell'elettrodo in modo da migliorare l'interfaccia tra il tessuto e l'elettrodo stesso, e per fare ciò si è passati all'uso di polimeri conduttori (CPs).



1.6 Esempi di biosensori

L'uso di dispositivi a semiconduttore organico ha portato numerosi vantaggi nelle applicazioni biomedicali grazie al fatto che si tratta di dispositivi flessibili e molto più biocompatibili rispetto alle tecnologie tradizionali inorganiche, oltre al fatto che si tratta di dispositivi di piccole dimensioni e con costi molto contenuti. Queste caratteristiche fanno sì che siano dispositivi in grado di essere impiantati anche all'interno di organismi viventi per monitorare, replicare o rigenerare determinate funzionalità vitali.

Fin dagli anni '80 è stato studiato un particolare dispositivo per l'interfacciamento biologico e che funge da biosensore, l'OTFT (Organic Thin Film Transistor).

1.1.1 OTFT – Organic Thin Film Transistor

L'OTFT è un transistor organico ad effetto di campo. Si tratta di un dispositivo a tre terminali, chiamati rispettivamente Gate, Drain e Source, a differenza dei tradizionali transistor inorganici MOSFET non ha un contatto di substrato. Le altre differenze principali sono nel funzionamento. In un transistor organico lo strato attivo è costituito da un sottile film organico quasi intrinseco e lavora in accumulazione, ovvero se l'OTFT è a canale n significa che ha un substrato di tipo n e si accende appena si forma uno strato di accumulazione di portatori maggioritari sotto il dielettrico, invece in un MOSFET si ha passaggio di corrente quando si va in inversione di portatori maggioritari all'interfaccia Si/SiO₂. Un'altra differenza sta nella definizione di tensione di soglia. In un OTFT non esiste l'inversione di portatori, quindi viene definita la tensione di soglia come la tensione di gate alla quale la concentrazione di portatori liberi in banda di trasporto eguaglia la concentrazione di portatori intrappolati nei difetti all'interfaccia. Per quanto riguarda la mobilità dei portatori si ha anche qui una sostanziale differenza: in un MOSFET l'aumento della tensione di gate causa il confinamento dei portatori in una regione molto ristretta che porta ad una diminuzione della mobilità dovuta all'aumento degli urti all'interfaccia, mentre in un OTFT l'aumento della tensione di gate fa aumentare la carica nel canale e di conseguenza la mobilità.

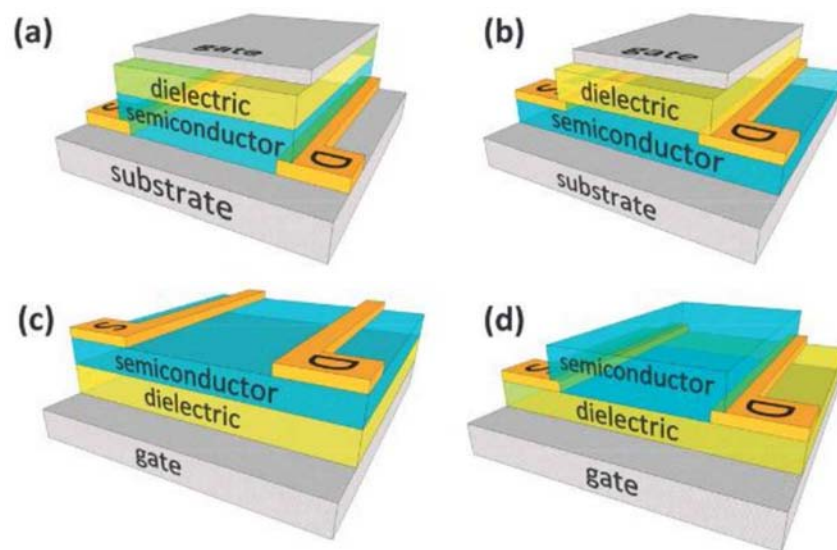
Il drain e il source si trovano a contatto con il semiconduttore organico e permettono il passaggio di corrente quando il dispositivo è acceso. Questa corrente è modulata per effetto di campo dal gate che è isolato dal semiconduttore attraverso un dielettrico. Sono possibili diverse strutture per l'OTFT che dipendono dal posizionamento del gate e dei contatti

rispetto al film organico. Si può avere infatti, partendo dall'alto verso il basso:

- Top-gate, dove si ha substrato, film organico, isolante e gate;
- Bottom-gate, dove si ha substrato, gate, isolante, e film organico.

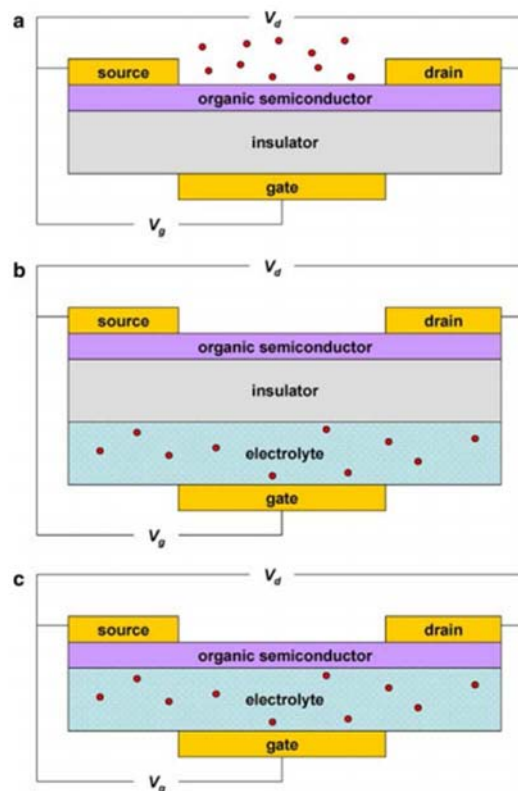
Mentre in base alla posizione dei contatti si può avere:

- Top-contact, source e drain sono sopra il film organico;
- Bottom-contact, source e drain sono sotto il film organico.



1.7 Strutture OTFT

Questo tipo di OTFT è anche detto OFET (Organic Field Effect Transistor) per il tipo di modulazione della corrente nel canale. Sono possibili altri meccanismi di modulazione della corrente che classificano diversi tipi di OTFT; se avviene attraverso un fenomeno di ossido-riduzione che modifica la tensione di gate si parla di OECTs (Organic Electrochemical Transistors), se viene posto uno strato elettrolita tra gate e semiconduttore si hanno gli EGOFETs (Electrolyte-Gated Organic Field Effect Transistors), se a questi viene aggiunto uno strato di dielettrico che isola l'elettrolita dallo strato semiconduttore si hanno gli ISOFETs (Ion-Sensitive OFETs).

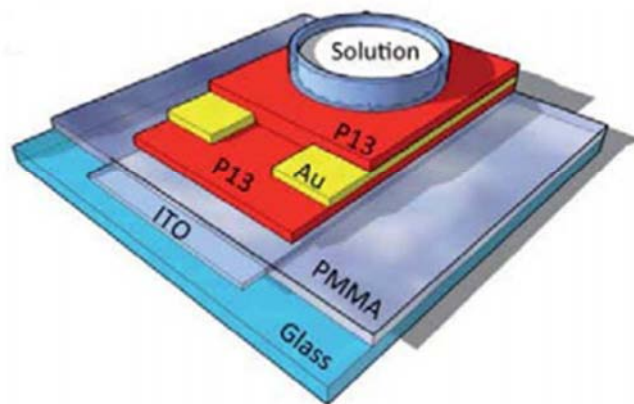


1.8 Tipologie di OTFT

Un particolare transistor usato per il biosensing è l'OCST (Organic Cell Stimulating and Sensing Transistor), ovvero l'unione di un OFET e un OECT.

Permette lo studio dei segnali neuronali sia in vitro che in vivo con una sensibilità maggiore rispetto a tutte le altre tecnologie odierne, attraverso la stimolazione e la rilevazione della risposta cellulare.

La sua struttura è bottom-gate/top-contact ed è un transistor basato su un derivato del perilene, il P13. L'OECT di questo transistor, oltre a permettere l'interfacciamento della parte elettronica con la parte biologica, riduce la tensione di soglia e grazie a ciò è possibile lavorare con basse tensioni ($<1V$) tipiche del mondo biologico, mentre attraverso l'OFET è possibile stimolare la cellula polarizzando l'interfaccia semiconduttore/neurone modulando l'elettrodo di gate. Il neurone, tramite l'emissione di un segnale cellulare stimola la modulazione del canale del dispositivo che causa una variazione della corrente di drain.



1.9 OCST

1.2 MISURE DI CARATTERIZZAZIONE

Per studiare le caratteristiche elettriche di questo tipo di transistors organici e capire come rispondono alle stimolazioni cellulari è stato deciso di implementare e di effettuare tre tipi di misurazioni.

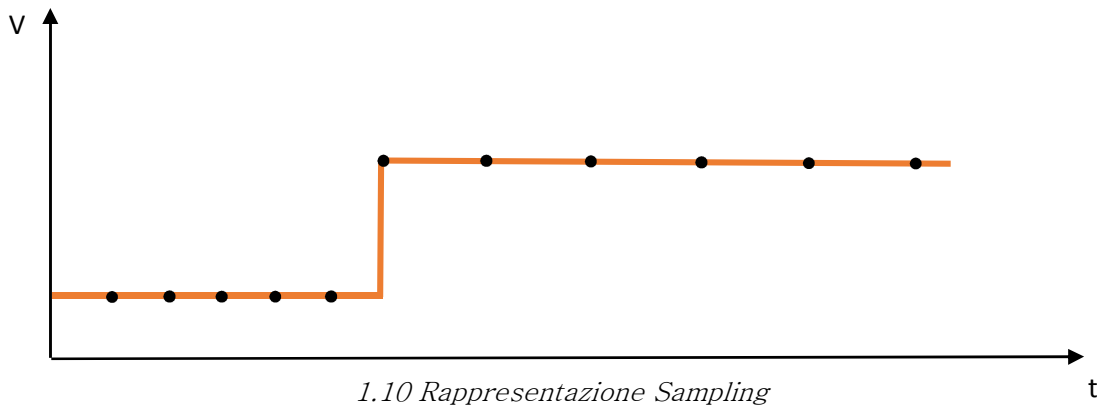
Si vuole poter ricavare il maggior numero di informazioni, quindi deve essere possibile scegliere i vari parametri di misura.

1.2.1 SAMPLING

La prima misurazione, chiamata “Sampling”, è la più semplice.

Si vuole poter imporre sul dispositivo in esame (Device Under Test), un gradino di tensione ad uno dei terminali e leggerne la risposta in corrente e/o in tensione.

Si vuole decidere la tensione di partenza e quella del gradino e per quanto tempo imporre queste due tensioni. Inoltre si vuole poter decidere il numero di campioni (sample) per poter leggere la risposta del DUT.



1.2.2 CICLOVOLTAMMETRIA

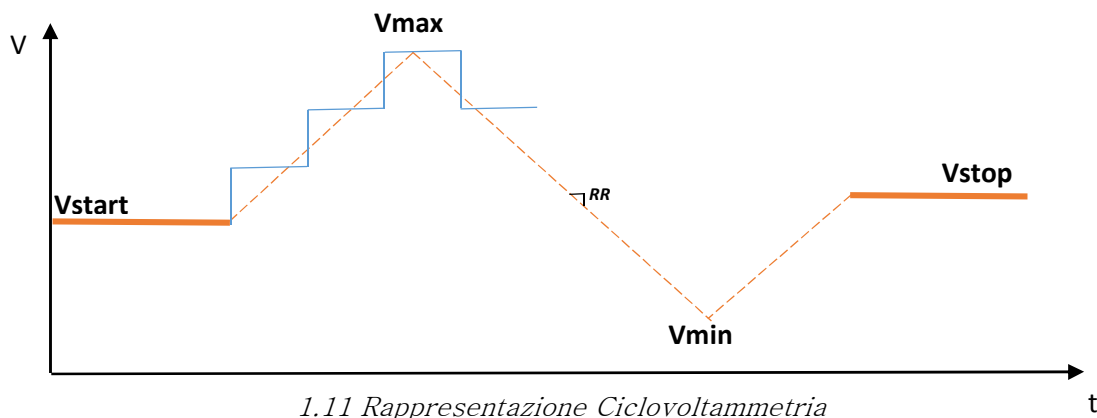
La ciclovoltammetria (CV) è una particolare tecnica utilizzata soprattutto per caratterizzare i biosensori attraverso una tensione con una forma d'onda triangolare imposta ad un terminale del DUT.

Si vuole poter scegliere la tensione di partenza, la tensione di arrivo, quella massima e la minima. Dalla tensione di partenza si può poi andare verso la tensione massima o quella minima preimpostando la direzione dell'onda triangolare.

Attraverso un parametro chiamato Ramp Rate (RR) si decide la “velocità” di variazione della tensione [V/s].

Si è dimostrato che imporre una tensione a gradini, campionare a grande velocità e mediare i valori acquisiti invece che imporre una rampa perfettamente triangolare, al fine del risultato è una buona approssimazione. Si sceglie quindi il numero di gradini di tensione che si vuole avere nella prima rampa, ovvero tra la tensione di partenza e quella massima (o minima, in base la direzione).

Campionando il segnale ad ogni gradino con un numero di campioni impostato, come nella misurazione precedente, si può ricavare la risposta del dispositivo a questo tipo di sollecitazione.

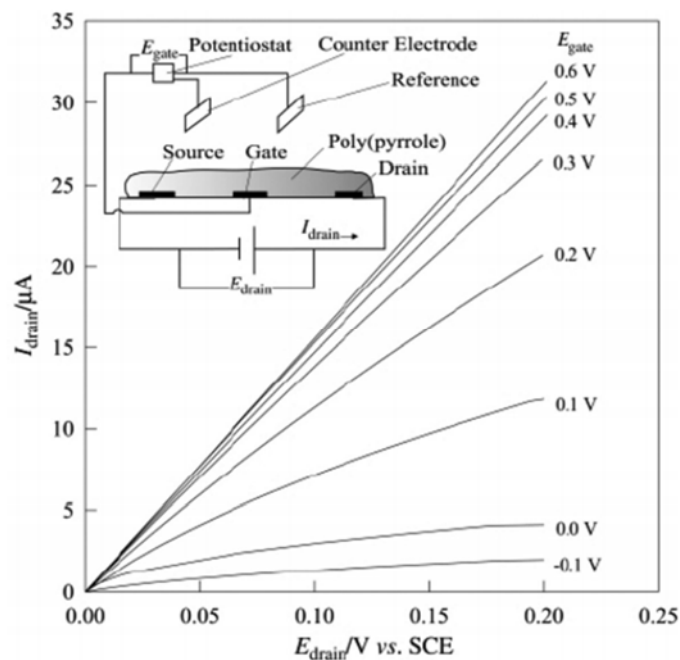


1.2.3 FIRING & SENSING

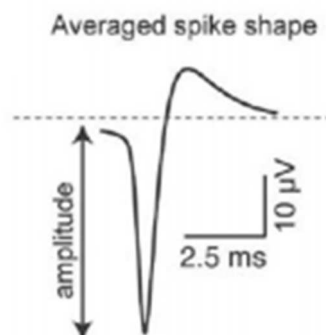
Questa è la misurazione più caratteristica che viene eseguita, infatti le precedenti due si possono applicare a qualsiasi tipo di dispositivo, mentre questa è stata implementata appositamente per i transistor usati nel biosensing.

Si tratta di una vera e propria simulazione dell'attività cellulare.

Tutti i terminali del dispositivo sono utilizzati, alcuni per effettuare la stimolazione, altri per la rilevazione della risposta.



1.12 Esempio set-up misura



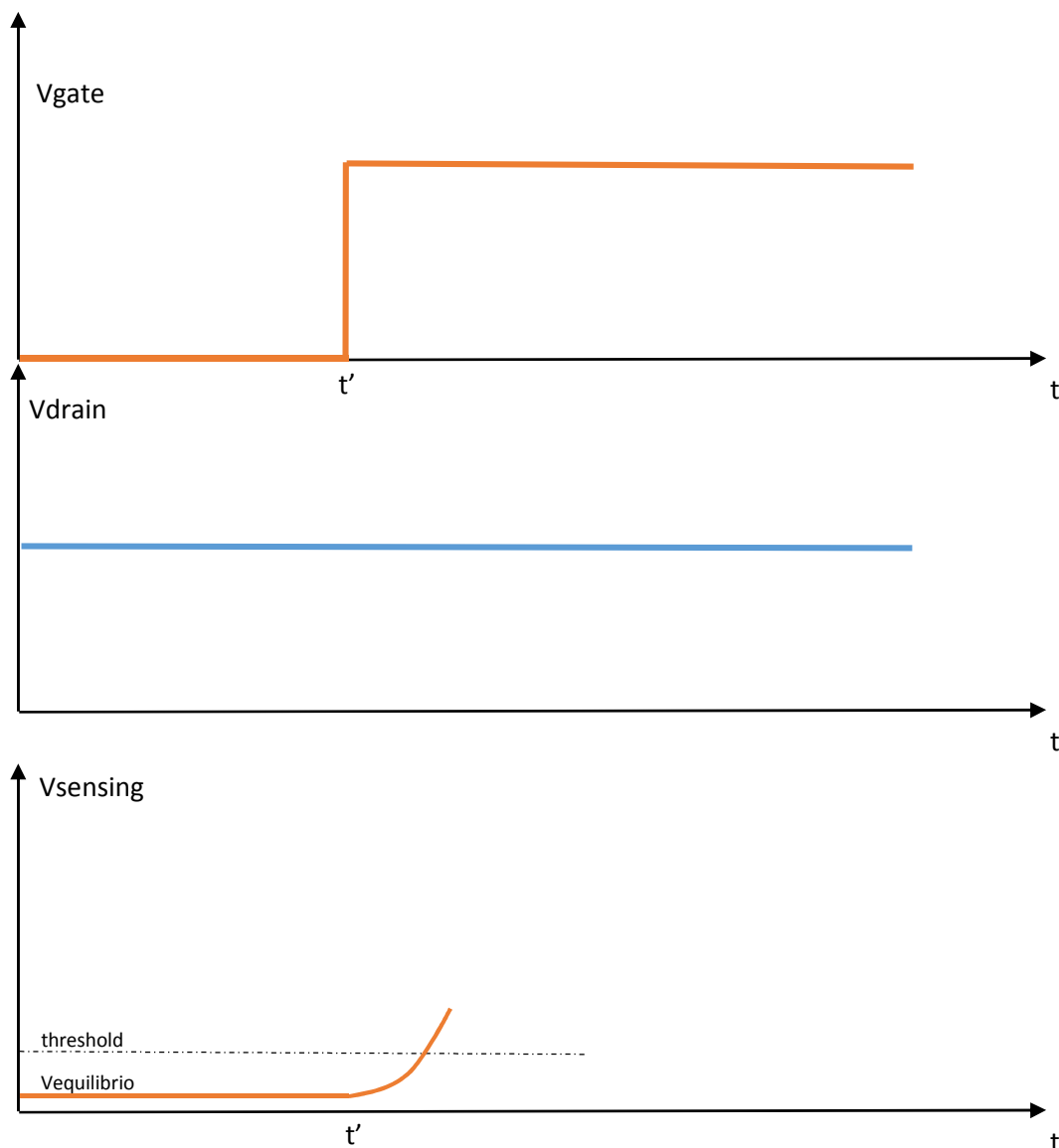
1.13 Potenziale d'azione neuronale

Al terminale di drain deve essere applicata una tensione costante di polarizzazione, mentre il terminale di source deve essere connesso ad un potenziale di riferimento, la massa.

Al terminale di gate va applicato un gradino di tensione che parte da 0V e dopo un certo tempo arriva ad un valore preimpostato e rimane a questo valore per

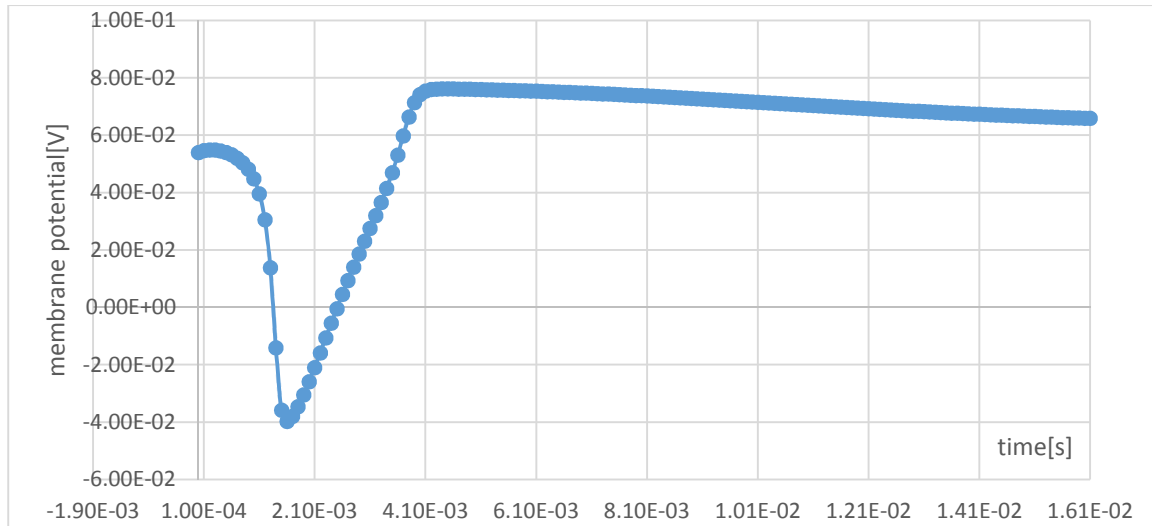
un tempo anch'esso preimpostato. Durante la fase iniziale il dispositivo si assesta ad un potenziale di equilibrio, questo deve essere letto e salvato. Per fare questo quindi è necessario poter leggere la tensione di risposta del DUT. Quando al terminale di gate, la tensione passa al valore del gradino, la risposta in tensione deve essere continuamente letta, e quando questa tensione letta supera una tensione di soglia definita, si deve imporre al dispositivo una serie di impulsi che simulano il potenziale d'azione della cellula neuronale il quale modula la corrente di canale. È necessario quindi leggere anche la risposta in corrente del dispositivo.

Per una maggiore completezza si vuole poter leggere le correnti di gate, drain e source e la tensione di risposta agli impulsi neuronali.



1.14 Rappresentazione Firing&Sensing

Quando la V_{sensing} supera la tensione di soglia partono, fino alla fine del gradino imposto al gate e traslati per partire dalla tensione di equilibrio, degli impulsi che simulano gli impulsi neuronali di questo tipo:



1.15 Valori imposti dalla scheda per l'impulso

CAPITOLO 2

PROGETTAZIONE CIRCUITO

2.1 PROGETTAZIONE SCHEMATICO

Per poter implementare le diverse tecniche di misura precedentemente descritte, si deve progettare un circuito in grado di imporre delle tensioni pre impostate e di leggere la risposta dei dispositivi sia in tensione sia in corrente.

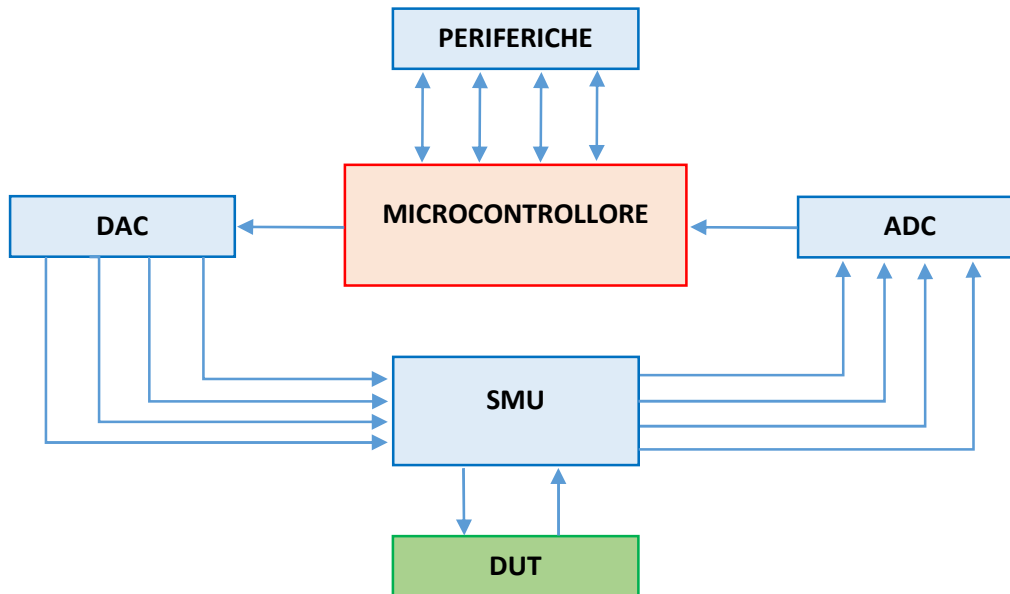
È necessario quindi interfacciare un microcontrollore ad un DAC, che converte una sequenza di bit proveniente dal microcontrollore, in una tensione analogica che viene imposta al DUT; e ad un ADC che converta nel modo più veloce possibile e con una certa risoluzione la tensione di risposta in un segnale digitale per poterlo salvare nelle memorie del sistema.

Il sistema elettronico progettato si divide quindi in due parti, una parte *analogica* e una *digitale* e si è cercato di mantenere separati il più possibile i segnali digitali da quelli analogici per evitare interferenze e disturbi.

La parte analogica è composta principalmente da tre *SMU* per la misura di *corrente* e una *SMU* per la misura di *tensione*.

La parte digitale comprende il cuore della scheda, ovvero il *microcontrollore* che verrà in seguito programmato per eseguire delle funzioni stabilite, e una serie di periferiche ad esso collegate. Alcune periferiche come *DAC* e *ADC* sono necessarie per elaborare i segnali analogici e digitali, altre sono necessarie per memorizzare i dati come ad esempio la *memoria* SDRAM, la memoria FLASH, e la scheda di memoria SD, inoltre sono presenti un *RTC* (Real Time Counter) per tenere memorizzata e aggiornare data e ora, un *port expander* per aumentare il numero di ingressi/uscite del microcontrollore, un *display OLED*, un connettore *PS/2* al quale viene collegata una tastiera esterna ed infine un ingresso *USB* per poter programmare il microcontrollore e per alimentare l'intera scheda con una tensione di 5V.

Il sistema inoltre deve potersi interfacciare con l'utente, che deve poter inviare comandi allo strumento e ricevere un feedback.

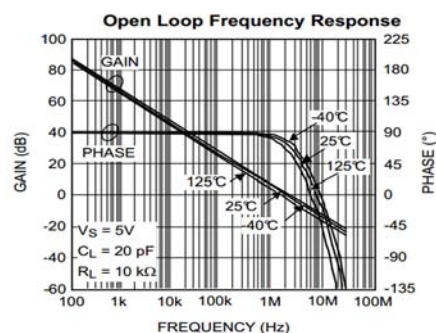


2.1 Schema a blocchi del sistema

2.1.1 SMU PER LA MISURA DI CORRENTE

Per poter leggere la risposta in corrente del DUT è necessario un circuito che misuri la caduta di tensione su una resistenza nota ed invii all'ADC questa tensione per la conversione. Questo perché l'ADC riceve in ingresso dei segnali in tensione, si ricava quindi il valore della corrente in fase di programmazione dividendo semplicemente il valore letto per la resistenza nota.

Il circuito è formato da uno stadio di ingresso composto da amplificatore operazionale 926-LMP7701MA/NOPB della Texas Instrument. Questo opamp è stato scelto principalmente per l'alimentazione a $\pm 5V$ e per la sua massima corrente di uscita di 86mA, in quanto questa corrente dovrà scorrere nel DUT quando viene collegato, ed essere misurata dal circuito. Altra caratteristica fondamentale per il corretto funzionamento del circuito è la stabilità che viene garantita dalla sua funzione di trasferimento.



2.2 Guadagno open loop OpAmp

±5V Electrical Characteristics ⁽¹⁾

Unless otherwise specified, all limits are ensured for $T_A = 25^\circ\text{C}$, $V^+ = 5\text{V}$, $V^- = -5\text{V}$, $V_{CM} = 0\text{V}$, and $R_L > 10\text{ k}\Omega$ to 0V .

Boldface limits apply at the temperature extremes.

Parameter		Test Conditions	Min ⁽²⁾	Typ ⁽³⁾	Max ⁽²⁾	Units
V_{OS}	Input Offset Voltage	LMP7701		± 37	± 200 ± 500	μV
		LMP7702/LMP7704		± 37	± 220 ± 520	
TCV_{OS}	Input Offset Voltage Temperature Drift	⁽⁴⁾		± 1	± 5	$\mu\text{V}/^\circ\text{C}$
I_B	Input Bias Current	^{(4) (5)} $-40^\circ\text{C} \leq T_A \leq 85^\circ\text{C}$		± 0.2	1 ± 50	pA
		^{(4) (5)} $-40^\circ\text{C} \leq T_A \leq 125^\circ\text{C}$		± 0.2	1 ± 400	
I_{OS}	Input Offset Current			40		fA
CMRR	Common Mode Rejection Ratio	$-5\text{V} \leq V_{CM} \leq 5\text{V}$ LMP7701	92 88	138		dB
		$-5\text{V} \leq V_{CM} \leq 5\text{V}$ LMP7702/LMP7704	90 86	138		
PSRR	Power Supply Rejection Ratio	$2.7\text{V} \leq V^+ \leq 12\text{V}$, $V_O = 0\text{V}$	86 82	98		dB
CMVR	Common Mode Voltage Range	CMRR $\geq 80\text{ dB}$ CMRR $\geq 78\text{ dB}$	-5.2 -5.2		5.2 5.2	V
A_{VOL}	Open Loop Voltage Gain	$R_L = 2\text{ k}\Omega$ (LMP7701) $V_O = -4.7\text{V}$ to 4.7V	100 98	121		dB
		$R_L = 2\text{ k}\Omega$ (LMP7702/LMP7704) $V_O = -4.7\text{V}$ to 4.7V	100 94	121		
		$R_L = 10\text{ k}\Omega$ (LMP7701) $V_O = -4.8\text{V}$ to 4.8V	100 98	134		
		$R_L = 10\text{ k}\Omega$ (LMP7702/LMP7704) $V_O = -4.8\text{V}$ to 4.8V	100 97	134		
V_{OUT}	Output Voltage Swing High	$R_L = 2\text{ k}\Omega$ to 0V LMP7701		90	150 170	mV from V^+
		$R_L = 2\text{ k}\Omega$ to 0V LMP7702/LMP7704		90	180 290	
		$R_L = 10\text{ k}\Omega$ to 0V LMP7701		40	80 100	
		$R_L = 10\text{ k}\Omega$ to 0V LMP7702/LMP7704		40	80 150	
	Output Voltage Swing Low	$R_L = 2\text{ k}\Omega$ to 0V LMP7701		90	130 150	mV from V^-
		$R_L = 2\text{ k}\Omega$ to 0V LMP7702/LMP7704		90	180 290	
		$R_L = 10\text{ k}\Omega$ to 0V LMP7701		40	50 60	
		$R_L = 10\text{ k}\Omega$ to 0V LMP7702/LMP7704		40	60 110	
I_{OUT}	Output Current ^{(6) (7)}	Sourcing $V_O = 0\text{V}$ $V_{IN} = 100\text{ mV}$ (LMP7701)	50 35	86		mA
		Sourcing $V_O = 0\text{V}$ $V_{IN} = 100\text{ mV}$ (LMP7702/LMP7704)	48 33	86		
		Sinking $V_O = 0\text{V}$ $V_{IN} = -100\text{ mV}$	50 35	84		

2.3 Caratteristiche elettriche OpAmp

L'operazionale riceve in ingresso al morsetto positivo la tensione imposta dal DAC. Dato che stiamo lavorando nella banda dell'operazionale, si può approssimare un corto circuito virtuale tra il morsetto positivo e quello negativo, quindi questa tensione va a polarizzare il dispositivo (*DUT*), il quale viene collegato al morsetto negativo dello stesso operazionale attraverso un relè comandato da un segnale inviato dal microcontrollore.

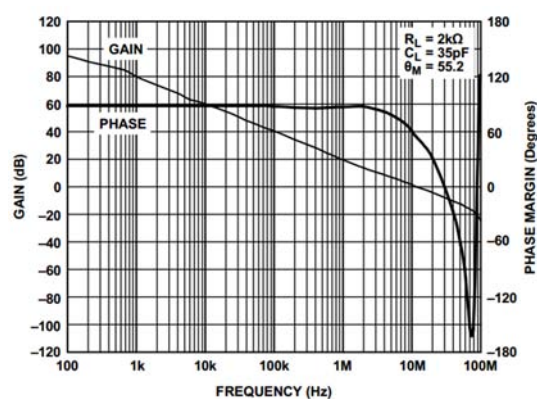
Quando viene aperto il relè, la corrente dell'induttore viene fatta ricircolare in un diodo zener.

Il circuito di feedback dell'operazionale è composto da delle resistenze di diverso valore collegate in parallelo. Queste resistenze, e tutte le resistenze della parte analogica sono state scelte con una tolleranza dello 0.1% sul valore nominale.

In serie alle resistenze di feedback è collegato un *foto-accoppiatore* per ogni diverso valore di resistenza che definisce un diverso range di misura della corrente. Questi foto-accoppiatori fungono da interruttori, sono pilotati da un segnale proveniente dal microcontrollore e la loro funzione è quella di selezionare la resistenza di feedback per effettuare la misura della corrente. È stato scelto di utilizzare questo tipo di interruttori per mantenere un isolamento galvanico tra la parte digitale del circuito e quella analogica.

Conoscendo la caduta di potenziale sulla resistenza di feedback e conoscendo il valore della resistenza selezionata, è possibile ricavare il valore della corrente che scorre sulla resistenza e quindi nel DUT attraverso il semplice rapporto $i_{DUT} = (V_{out} - V_{force})/R_{sense}$.

La tensione di uscita dell'operazionale e quella imposta al DUT vengono quindi prelevate e mandate in ingresso a due buffer di tensione e successivamente ad un amplificatore differenziale. Per i due buffer è stato usato lo stesso tipo di operazionale del primo stadio, ad alta corrente di uscita, mentre per lo stadio di uscita del circuito di misura è stato scelto un 584-AD8675ARMZ della Analog Devices.



2.4 Guadagno open loop OpAmp

ELECTRICAL SPECIFICATIONS

$V_s = \pm 5.0\text{ V}$, $V_{CM} = 0\text{ V}$, $V_O = 0\text{ V}$, $T_A = +25^\circ\text{C}$, unless otherwise specified.

Table 2.

Parameter	Symbol	Test Conditions/Comments	Min	Typ	Max	Unit
INPUT CHARACTERISTICS						
Offset Voltage	V_{OS}	$-40^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$		10	75	μV
				12	240	μV
Offset Voltage Drift	$\Delta V_{OS}/\Delta T$	$-40^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$		0.2	0.6	$\mu\text{V}/^\circ\text{C}$
Input Bias Current	I_B	$-40^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$	-2	+0.5	+2	nA
			-5.5	-2	+5.5	nA
Input Offset Current	I_{OS}	$-40^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$	-1	+0.1	+1	nA
			-2.8	+0.1	+2.8	nA
Input Voltage Range	IVR		-3.0		+3.0	V
Common-Mode Rejection Ratio	CMRR	$V_{CM} = -3.0\text{ V to }+3.0\text{ V}$	105	130		dB
		$-40^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$	105			dB
Open-Loop Gain	A_{VO}	$R_L = 2\text{ k}\Omega$ to ground, $V_O = -3.5\text{ V to }+3.5\text{ V}$	120	126		dB
		$-40^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$	117			dB
OUTPUT CHARACTERISTICS						
Output Voltage High	V_{OH}	$R_L = 10\text{ k}\Omega$ to ground	4.90	4.95		V
		$-40^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$	4.85			V
		$R_L = 2\text{ k}\Omega$ to ground	4.80	4.90		V
		$-40^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$	4.75			V
Output Voltage Low	V_{OL}	$R_L = 10\text{ k}\Omega$ to ground		-4.98	-4.90	V
		$-40^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$			-4.85	V
		$R_L = 2\text{ k}\Omega$ to ground		-4.91	-4.86	V
		$-40^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$			-4.82	V
Short-Circuit Limit	I_{SC}			± 35		mA
POWER SUPPLY						
Power Supply Rejection Ratio	PSRR	$V_s = \pm 5.0\text{ V to } \pm 15.0\text{ V}$	120	140		dB
		$-40^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$	120			dB
Supply Current/Amplifier	I_{SR}	$V_O = 0\text{ V}$		2.3	2.7	mA
		$-40^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$			3.4	mA
DYNAMIC PERFORMANCE						
Slew Rate	SR	$R_L = 2\text{ k}\Omega$		2.5		V/ μs
Gain Bandwidth Product	GBP			10		MHz
NOISE PERFORMANCE						
Voltage Noise	$e_{n,p-p}$	0.1 Hz to 10 Hz		0.1		$\mu\text{V p-p}$
Voltage Noise Density	e_n	$f = 1\text{ kHz}$		2.8		nV/ $\sqrt{\text{Hz}}$
Current Noise Density	i_n	$f = 10\text{ Hz}$		0.3		pA/ $\sqrt{\text{Hz}}$

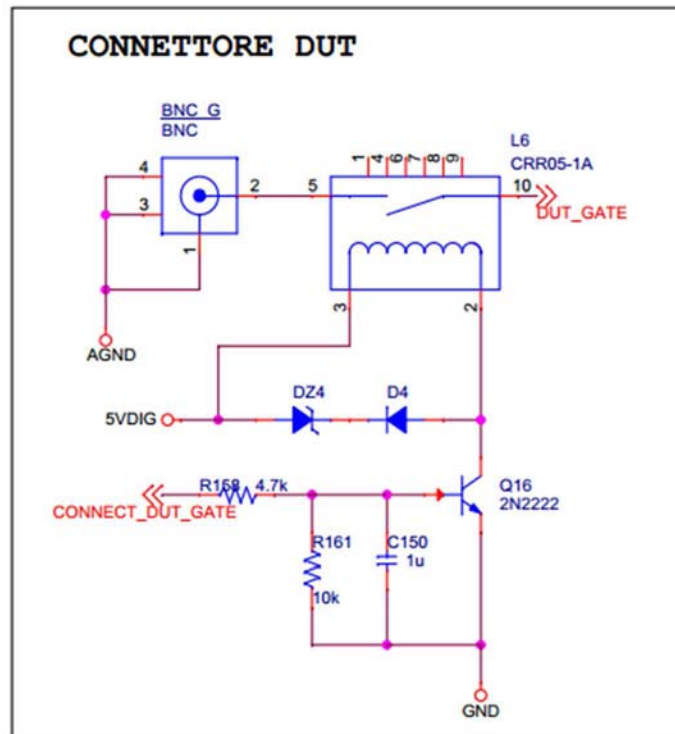
2.5 Caratteristiche elettriche OpAmp

Questo operazionale è stato utilizzato principalmente per la piccolissima tensione di offset che al massimo è di 75 μV . Come prima l'alimentazione è a $\pm 5\text{V}$.

La sua tensione di ingresso viene amplificata e la tensione di uscita viene inviata all'ADC per poterla elaborare successivamente dal microcontrollore.

La caduta di tensione sulla resistenza deve essere minore o al massimo di 1V e dato che l'ADC riceve in ingresso tensioni di $\pm 5\text{V}$, lo stadio di uscita formato dall'amplificatore differenziale amplifica il segnale di ingresso di un fattore 4.7.

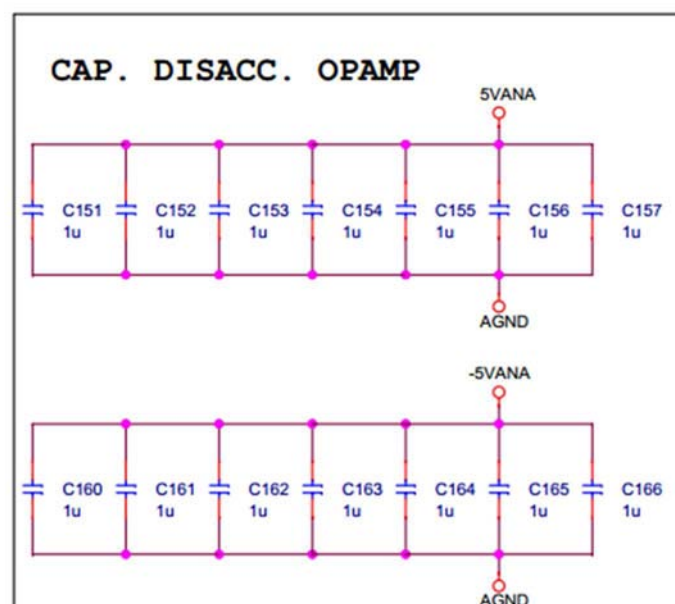
La minima corrente misurabile quindi è di 1V diviso il valore massimo di resistenza di range che è stata scelta di 3M Ω , quindi si può misurare una corrente di 0.33 μA .



2.7 Collegamento DUT

Vengono collegate tra pin di alimentazione degli opamp e massa due capacità di bypass da 1uF. Queste hanno la funzione di cortocircuitare a massa i disturbi ed il rumore che possono essere presenti sulle linee di alimentazione e che, entrando nell'operazionale possono causare errori. I condensatori devono essere collegati vicini ai piedini di alimentazione degli operazionali per ridurre l'impedenza della connessione.

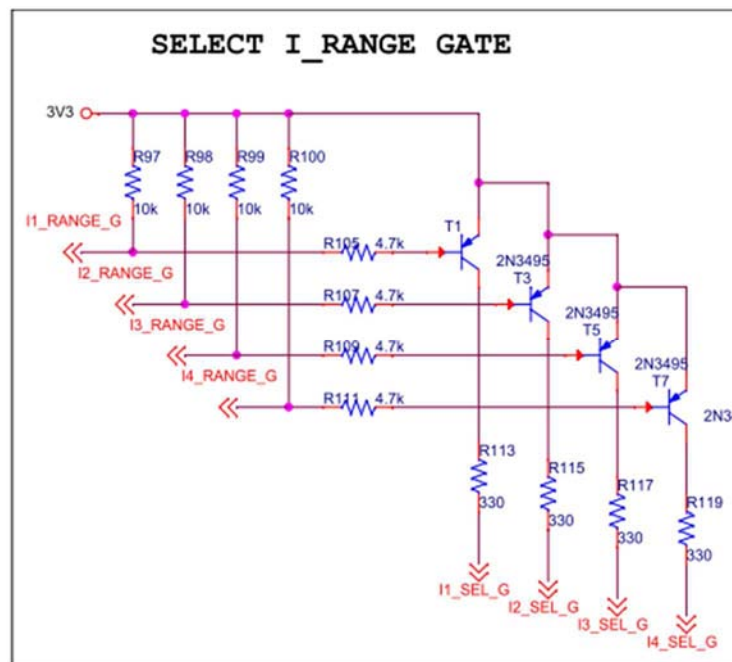
Vengono usate queste capacità in tutto il circuito per ogni alimentazione di ogni componente.



2.8 Capacità bypass alimentazioni OpAmp

Per poter selezionare il range di misura si deve attivare il foto-accoppiatore corrispondente alla resistenza voluta e chiudere lo switch del MAX326CSE+ che permette di collegare l'uscita desiderata. Per fare ciò si utilizza un segnale digitale proveniente dal port-expander che va ad attivare o disattivare un BJT npn che permette il passaggio di corrente nel foto-accoppiatore selezionato.

Accendendo il foto-accoppiatore del range 1, si va a selezionare la resistenza di feedback da $3\text{M}\Omega$, il range 2 seleziona la resistenza da $3.3\text{k}\Omega$, il range 3 quella da $1\text{k}\Omega$ e il range 4 da 100Ω . Si è fatto il modo che la differenza di potenziale sulla resistenza non superi il valore di 1V . in questo modo selezionando il range 4 si può misurare una corrente massima di 10mA , con il range 2 di 0.3mA , con il range 3 di 1mA , mentre con il range 1 si può misurare una corrente massima di $0.33\mu\text{A}$.



2.9 Circuito pilotaggio per scelta range

2.1.2 SMU PER LA MISURA DI TENSIONE

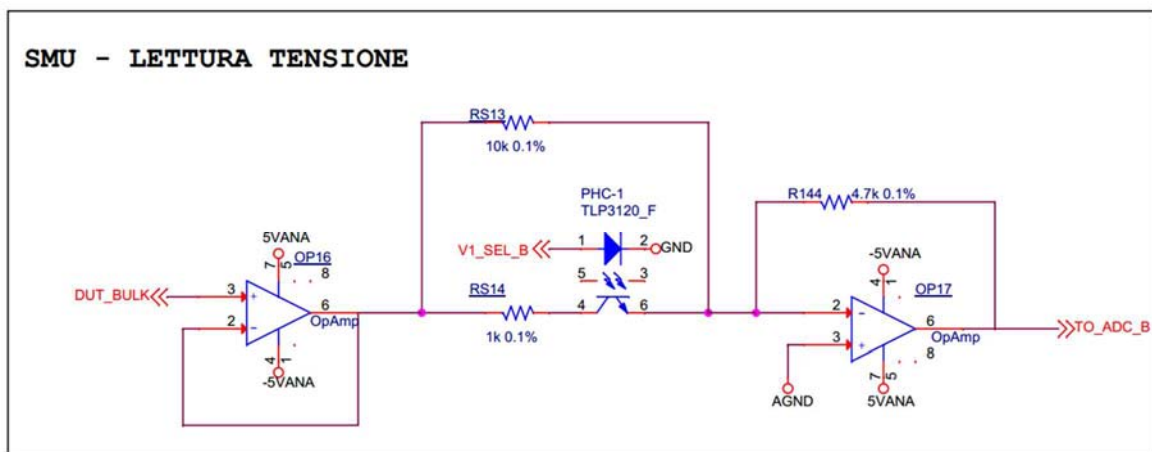
Per poter leggere la risposta in tensione del sensore è sufficiente prendere in ingresso la tensione ed amplificarla al valore più opportuno per una conversione ottimale dell'ADC.

Il circuito per la misura di tensione è composto semplicemente da un buffer e da un amplificatore operazione in configurazione invertente.

Il buffer riceve in ingresso la tensione da leggere del DUT, mentre l'amplificatore invertente amplifica la tensione in ingresso per poterla successivamente inviare all'ADC.

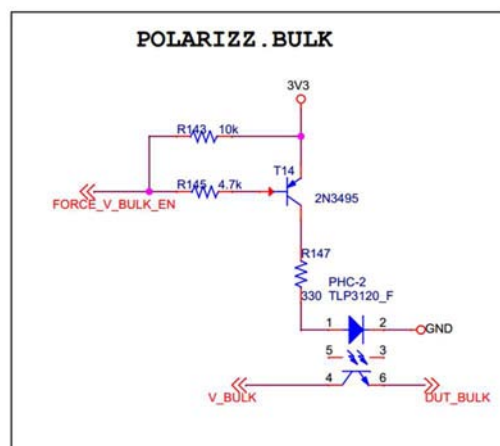
Un foto-accoppiatore comandato dal microcontrollore permette di scegliere il guadagno dell'amplificatore collegando o scollegando la resistenza in ingresso al morsetto negativo dell'operazionale, esattamente come nel caso della SMU di corrente.

Si possono selezionare due range, chiudendo o aprendo un foto-accoppiatore. Se viene selezionato il range 1 il foto-accoppiatore viene chiuso e si ha un fattore di amplificazione di $(-4.7k/(1k||10k))$, mentre se viene selezionato il range 2 si ha un fattore di amplificazione di $(4.7k/10k)$. L'opamp scelto per il buffer è un 926-LMP7701MA/NOPB, questa scelta è stata effettuata per la sua bassa corrente di leakage, mentre per la configurazione invertente è stato utilizzato un 584-AD8675ARMZ per la sua piccola tensione di offset.



2.10 Circuito lettura tensione

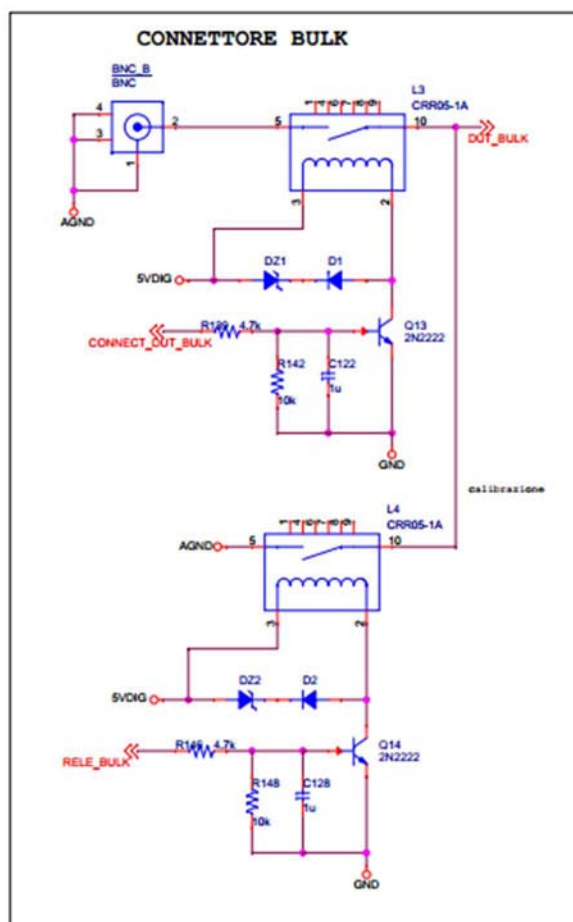
È possibile polarizzare il DUT con la tensione desiderata attraverso la tensione di uscita del DAC e misurare la tensione di risposta del dispositivo collegandolo alla SMU attraverso un relè.



2.11 Collegamento per polarizzazione DUT

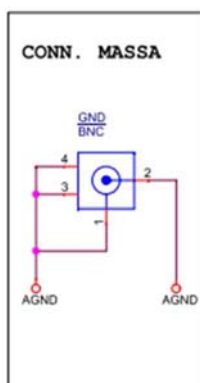
È presente anche un circuito che permette la calibrazione del DAC e dell'ADC. L'ingresso della SMU può essere collegato a *GND* attraverso la massa analogica della scheda.

Si può fare questo attivando un relè, in questo modo la tensione letta dall'ADC, se risulta diversa da 0, è dovuta solo al rumore totale del circuito di elaborazione del segnale e sarà quindi uguale all'offset da sottrarre durante l'elaborazione dei dati acquisiti in fase di programmazione.



2.12 Collegamento DUT a circuito lettura e calibrazione

Per effettuare alcune misure è necessario collegare un elettrodo del dispositivo a massa, per questo motivo è stato aggiunto al circuito un connettore BNC collegato direttamente alla massa analogica dello strumento.



2.13 Connettore GND

2.1.3 POLARIZZAZIONE DUT

Il circuito di polarizzazione elabora la tensione in uscita dal DAC e impone la tensione in ingresso alle varie SMU.

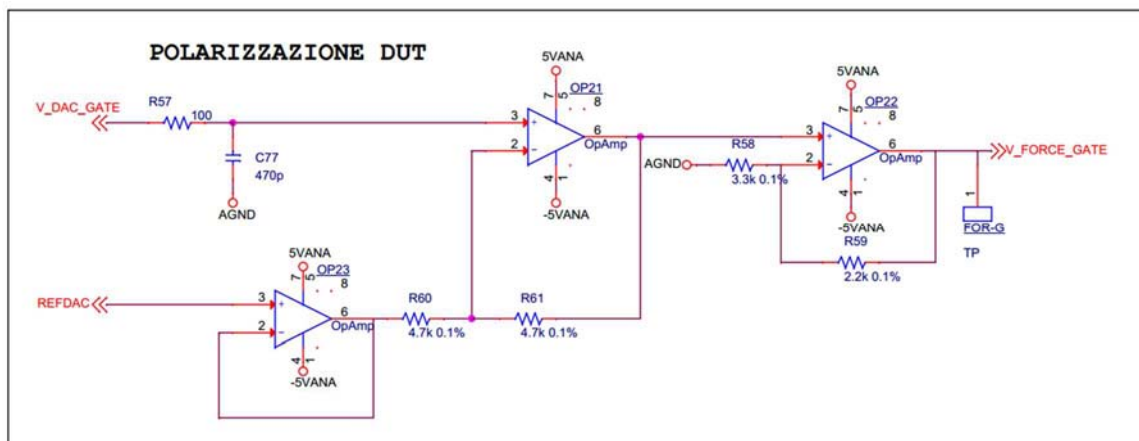
Il primo stadio di elaborazione è composto da un amplificatore operazionale che riceve in ingresso al morsetto positivo la tensione di uscita del DAC e in ingresso al morsetto negativo la tensione di riferimento interna del DAC precedentemente bufferizzata.

$$V_{out1} = V_{DAC} \left(1 + \frac{4.7k}{4.7k} \right) + V_{REFDAC} \left(-\frac{4.7k}{4.7k} \right)$$

Questa tensione entra in un amplificatore operazionale in configurazione non invertente con guadagno $(1 + 2.2k/3.3k)$.

La tensione di uscita del DAC e la tensione di riferimento entrano in due operazionali 584-AD8675ARMZ a basso offset, mentre l'operazionale di uscita ad alta corrente di uscita un è stato scelto un 926-LMP7701MA/NOPB.

È stato inserito questo circuito per poter avere una tensione bipolare in uscita dal DAC.



2.14 Circuito polarizzazione DUT

2.1.4 ADC – Analog / Digital Converter

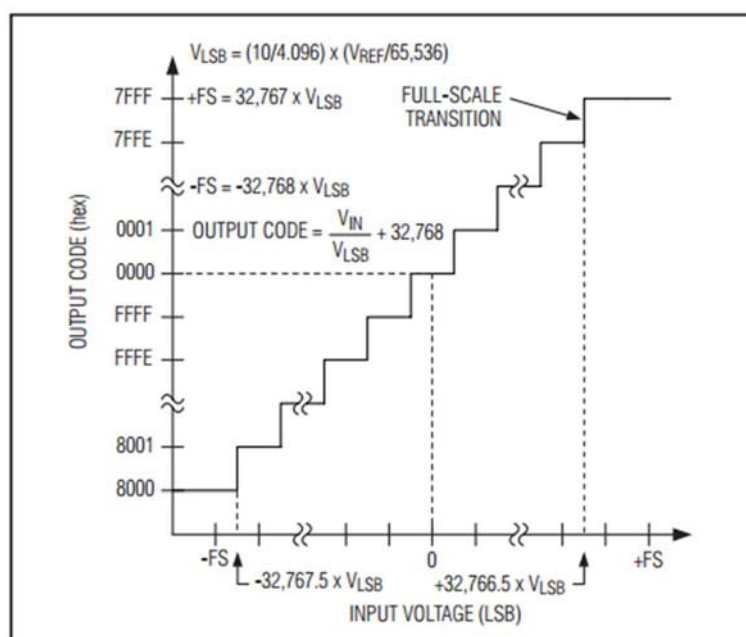
È stato scelto l'ADC *MAX11044BECB+* della Maxim Integrated.

È alimentato da una singola alimentazione analogica di +5V e una singola alimentazione digitale a 3.3V.

È costituito da quattro canali di ingresso per le tensioni analogiche le quali provengono direttamente dalle uscite delle quattro SMU.

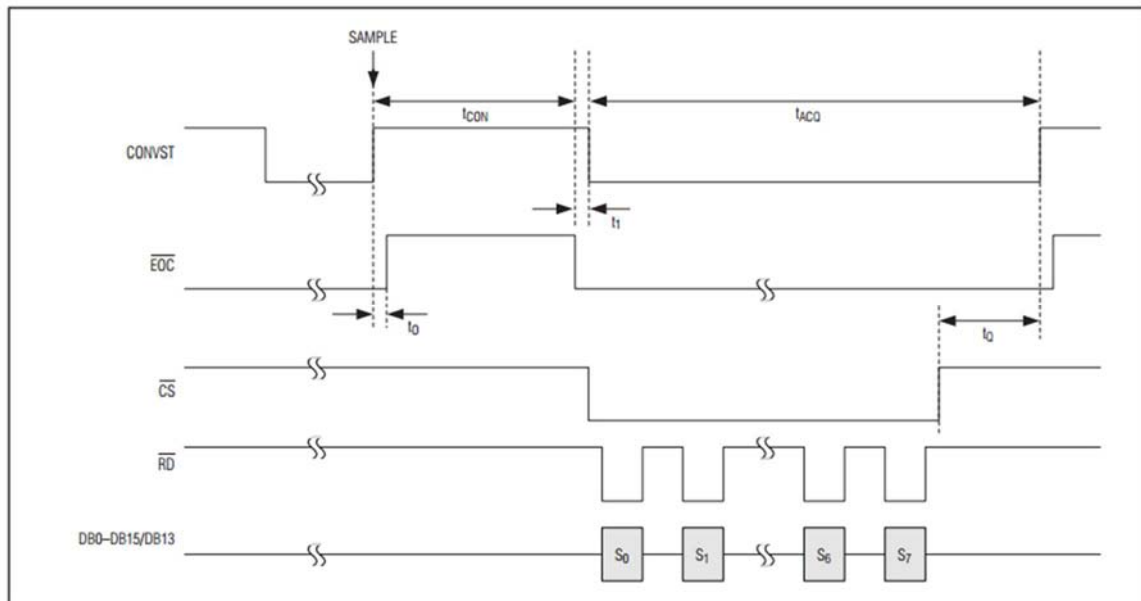
Il range delle tensioni di ingresso è bipolare e varia tra $\pm(1.22 + V_{REF10})$, la tensione di riferimento è stata utilizzata quella interna dello strumento di 4.096V.

La conversione avviene ad una velocità massima di campionamento di 250ksps per ogni canale con una risoluzione di 16bit. È stato scelto un tipo di ADC con uscita parallelo per avere una conversione più rapida possibile.



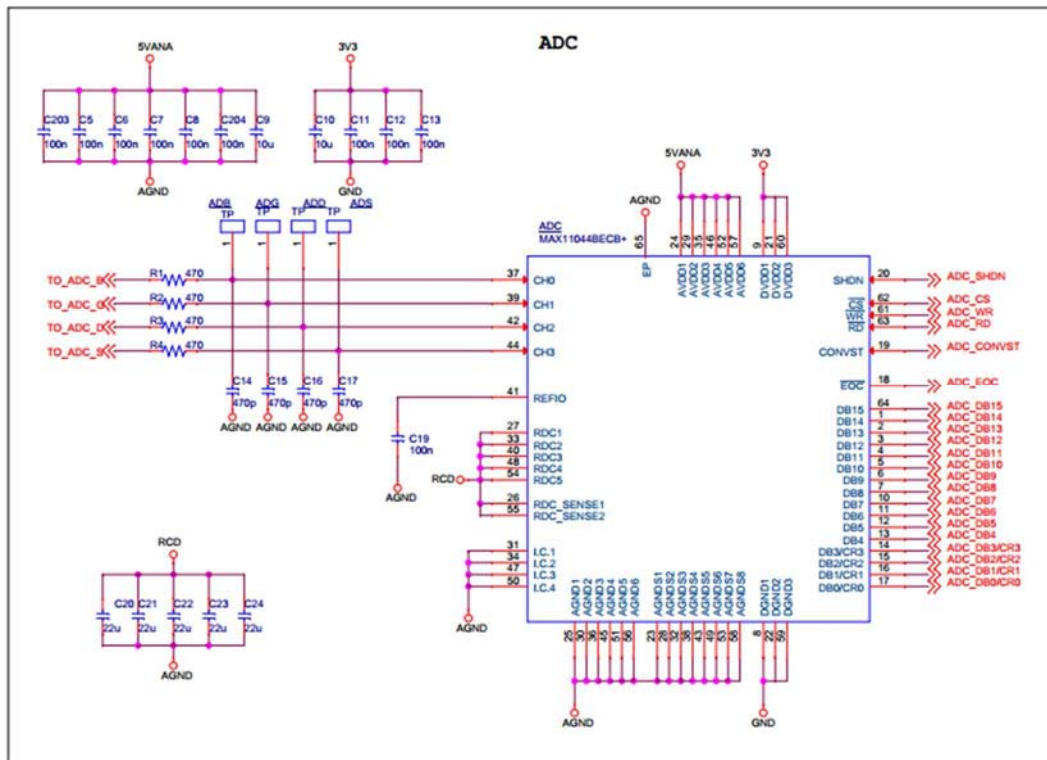
2.15 Caratteristica in/out ADC

Il tempo di conversione è di 3 μ s, mentre il tempo di acquisizione può variare da 1 μ s a 1ms.



2.16 Caratteristica temporale acquisizione/conversione ADC

Oltre ai quattro segnali di ingresso e i sedici di uscita, sono presenti altri segnali che comunicano con il microcontrollore. Il segnale ADC_SHDN (Shutdown) se mantenuto a livello logico alto spegne il dispositivo, quindi per effettuare qualsiasi operazione è necessario porre a 0V questo ingresso. L'ADC_CS (Chip Select) serve per selezionare il dispositivo, l'ADC_WR (Write) per poter scrivere dati nell'ADC, mentre l'ADC_RD (Read) per poter leggere. Altri due segnali sono l'ADC_CONVS (Conversion Start) che serve per iniziare la conversione e l'ADC_EOC (End Of Conversion) che indica la fine della conversione del segnale.



2.17 Circuito ADC

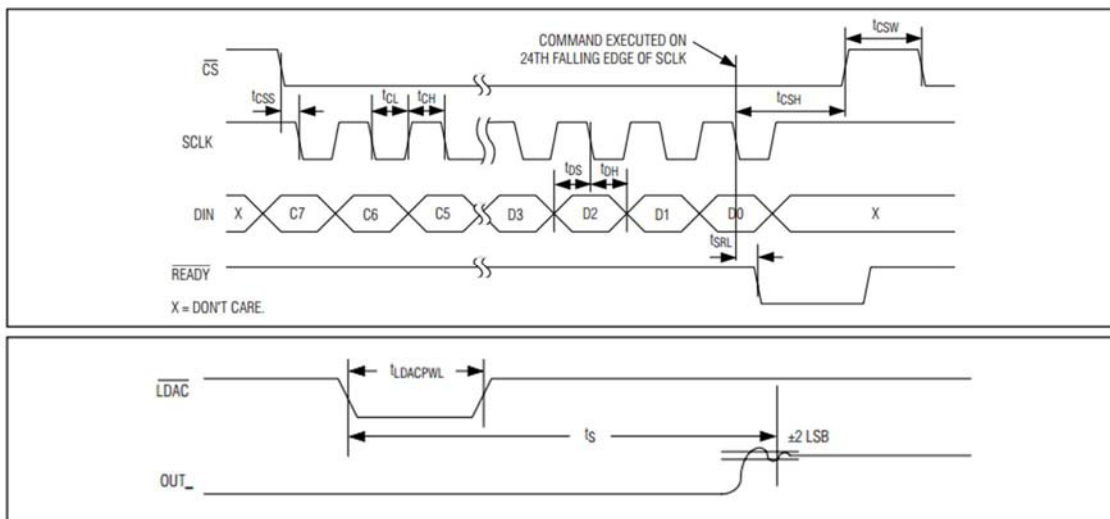
2.1.5 DAC – Digital / Analog Converter

È stato utilizzato il DAC *MAX5134AGUE+* della Maxim Integrated. Questo convertitore ha un'alimentazione analogica di +5V e un'alimentazione digitale di +3.3V. Dispone di quattro uscite, utilizzate per polarizzare le quattro SMU presenti nel progetto. Ha una tensione interna di riferimento di 2.44V e si tratta di un convertitore a 16bit.

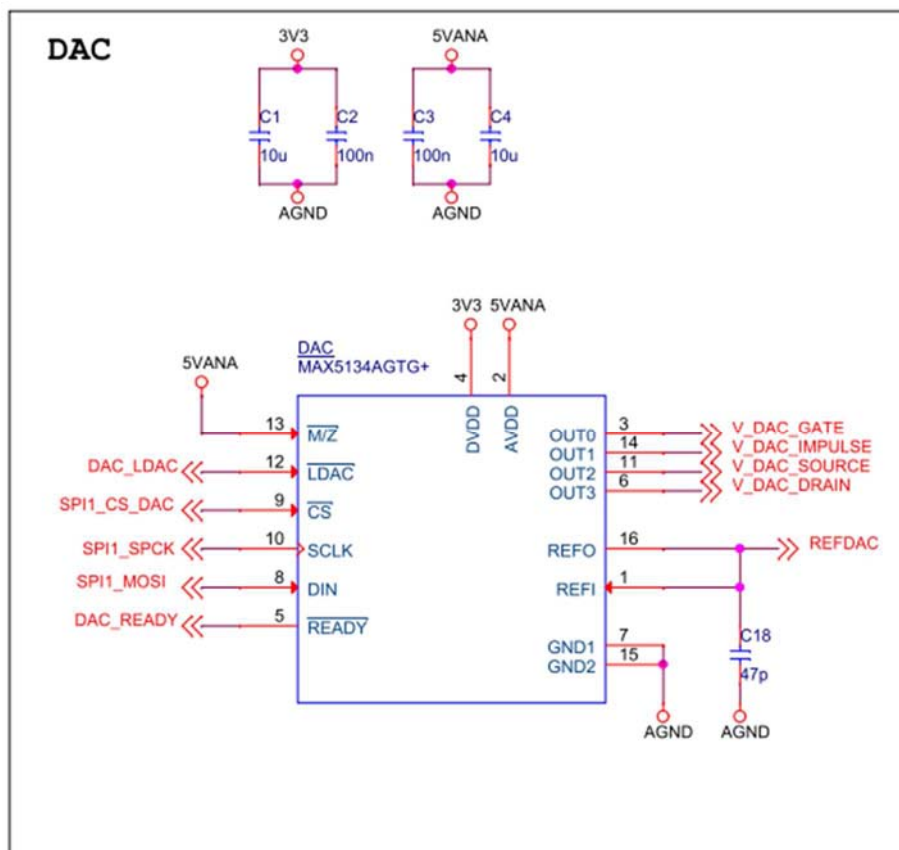
DAC LATCH CONTENTS		ANALOG OUTPUT, $V_{OUT_}$
MSB	LSB	
1111 1111 1111 1111		$V_{REF} \times (65,535/65,536)$
1000 0000 0000 0000		$V_{REF} \times (32,768/65,536) = 1/2 V_{REF}$
0000 0000 0000 0001		$V_{REF} \times (1/65,536)$
0000 0000 0000 0000		0

2.18 Caratteristica in/out DAC

La tensione di uscita può variare in un range che va da 0V a V_{REF} . La comunicazione tra il DAC e il microcontrollore avviene tramite tre segnali di *SPI* più un segnale di LOAD con in quale viene erogata la tensione in uscita, e un segnale di READY che indica che la configurazione del DAC è completata.



2.19 Caratteristica temporale segnali DAC



2.20 Circuito DAC

2.1.6 SPI – Serial Peripheral Interface

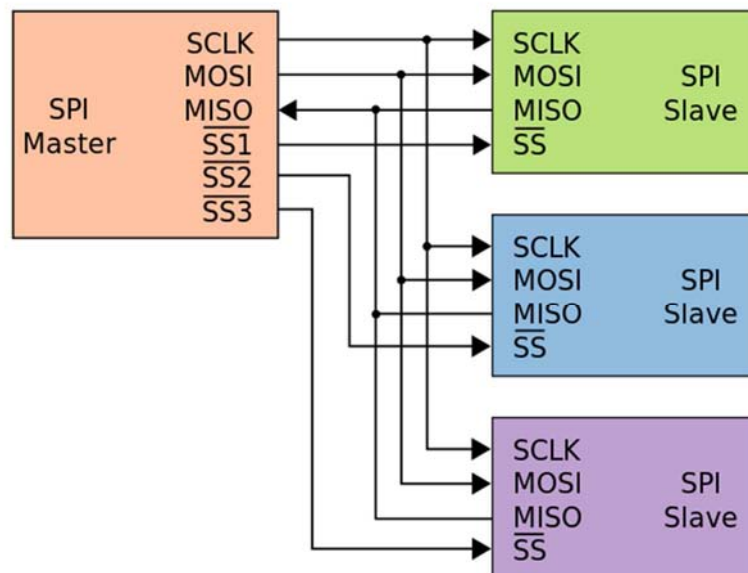
L'SPI è un bus dati di tipo *seriale*, *sincrono* e *full duplex*.

Il bus comunicazione SPI è un semplice protocollo di trasferimento dati ad alta velocità che avviene tra un dispositivo detto master (microcontrollore) e uno o più dispositivi detti slaves, questi sono i motivi principali per cui è stato scelto questo tipo di comunicazione tra i vari dispositivi del progetto.

Il master controlla il bus di comunicazione attraverso il segnale di clock, ha la possibilità di inviare e ricevere dati e comandi e di iniziare la sessione di trasmissione.

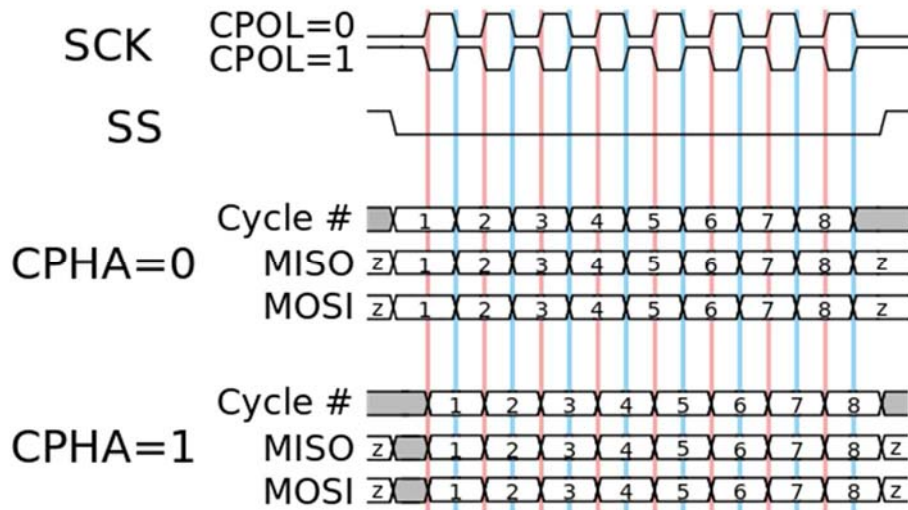
Lo slave è un dispositivo periferico che può ricevere e inviare dati, ma non può inviare comandi, né iniziare una nuova sessione di trasmissione.

La comunicazione avviene su una linea dati chiamata *MOSI* (*Master Output Slave Input*) ed è sincronizzata grazie alla linea di clock chiamata *SCK* (*Serial Clock*). I vari slaves dispongono di una linea di *CS* (*Chip Select*) che il master dovrà comandare per attivare la comunicazione con lo slave desiderato. La linea MOSI è unidirezionale, ma è possibile comunicare anche dallo slave al master grazie alla linea *MISO* (*Master Input Slave Output*). Grazie a ciò è possibile realizzare sistemi di comunicazioni full duplex dove master e slave si inviano dati contemporaneamente su queste due linee.



2.21 Esempio collegamento Master/Slaves

La codifica del segnale ricevuto dipende da due parametri che dipendono dallo slave selezionato, la polarità del clock (CPOL) e dalla fase del clock (CPHA). Questi parametri determinano i fronti del clock su cui sono sincronizzati e campionati i dati. Quindi se gli slave hanno parametri di clock diversi, il master dovrà modificare la codifica di trasmissione e ricezione dei dati per ogni slave a cui è collegato.



2.22 Configurazioni CPOL/CPHA

2.1.7 MICROCONTROLLORE

È stato scelto il microcontrollore a 32 bit AVR AT32UC3A3256 della ATMEL.

L'alimentazione di questo microcontrollore è singola a 3.3V.

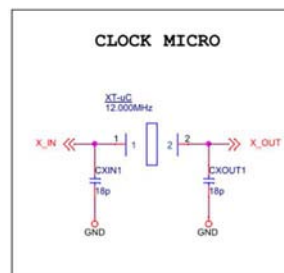
Possiede una memoria flash interna ad alta velocità da 256kbytes, e una memoria SRAM da 64kbytes.

Supporta memorie esterne di tipo SDRAM, flash e MMC, infatti tutte queste memorie sono state utilizzate nel progetto.

Dispone inoltre del bus USB 2.0 (480Mbit/s) usato per la programmazione, di due timer (TC) a 16bit a tre canali ciascuno utilizzati nella gestione degli interrupt e nel conteggio in varie funzioni nel firmware, di due linee di comunicazione SPI utilizzate per scambiare dati con alcune periferiche e di 110 GPIO (General Purpose Input Output) ovvero segnali di ingresso o uscita ai quali può essere data una certa funzione indicata nel datasheet.

È presente l'interfaccia JTAG per la programmazione.

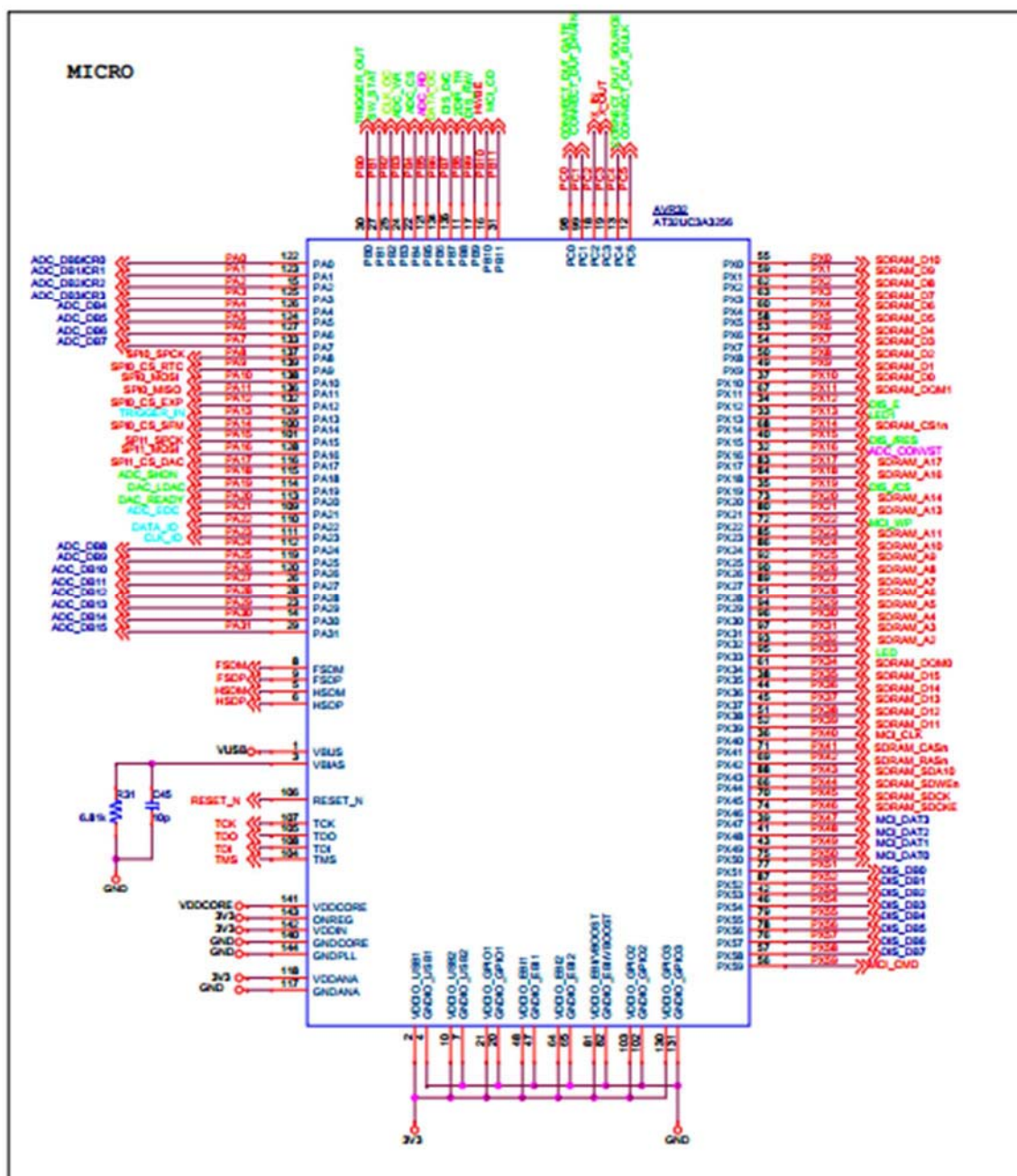
Per il progetto è stato collegato al microcontrollore un clock esterno da 12MHz.



2.23 Clock microcontrollore

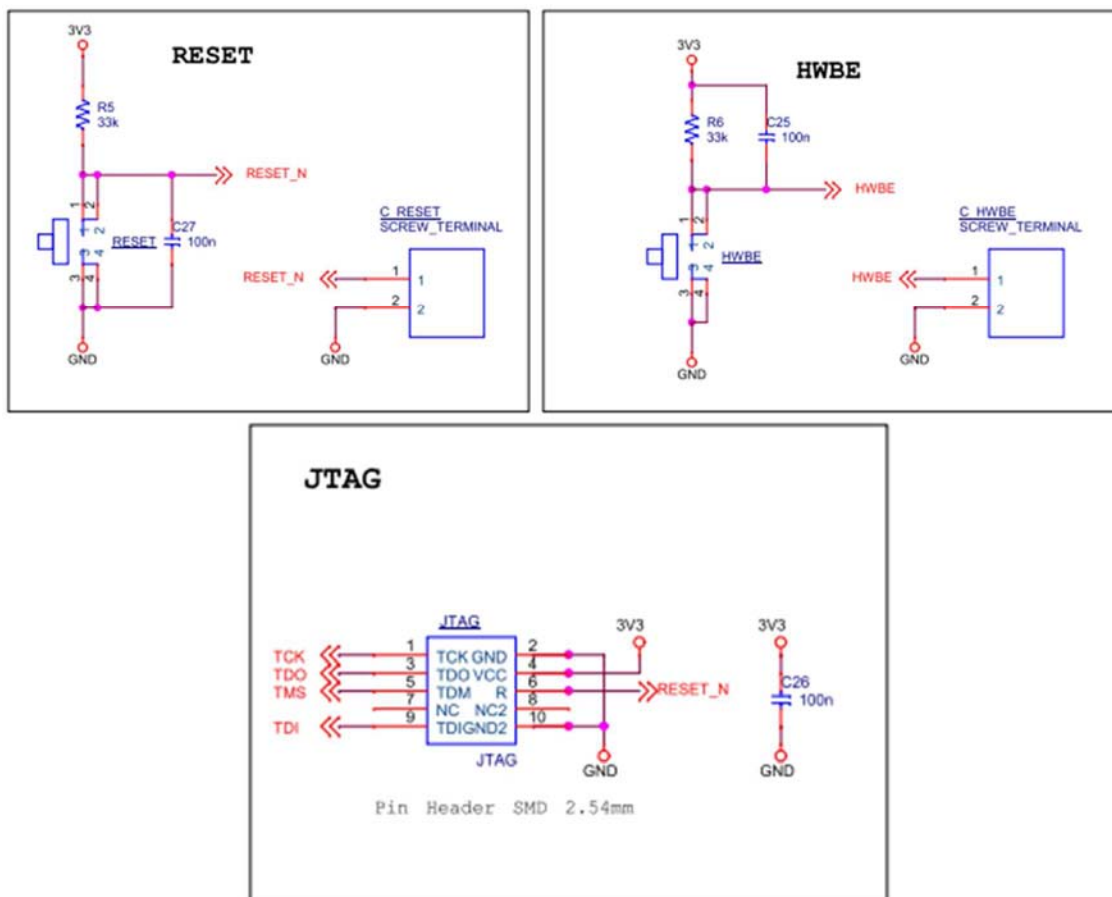
Questo microcontrollore è suddiviso in tre porte da 32bit ciascuna. Alcuni segnali sono stati collegati alla stessa porta per comodità nella

Segnali come l'ADC_CONVST e l'ADC_RD invece, sono stati collegati a dei timer del microcontrollore, mentre segnali di SPI, SDRAM esterna e MCI sono stati connessi ai pin dedicati del dispositivo, il resto dei pin è stato assegnato a GPIO generici.



2.24 Collegamenti microcontrollore

Per la programmazione del microcontrollore sono necessari due pulsanti, uno collegato al pin di RESET e uno collegato a quello di HWBE. Inoltre è stato inserito un connettore header a 10poli per la programmazione tramite JTAG.



2.25 Connettori e pulsanti: RESET, HWBE, JTAG

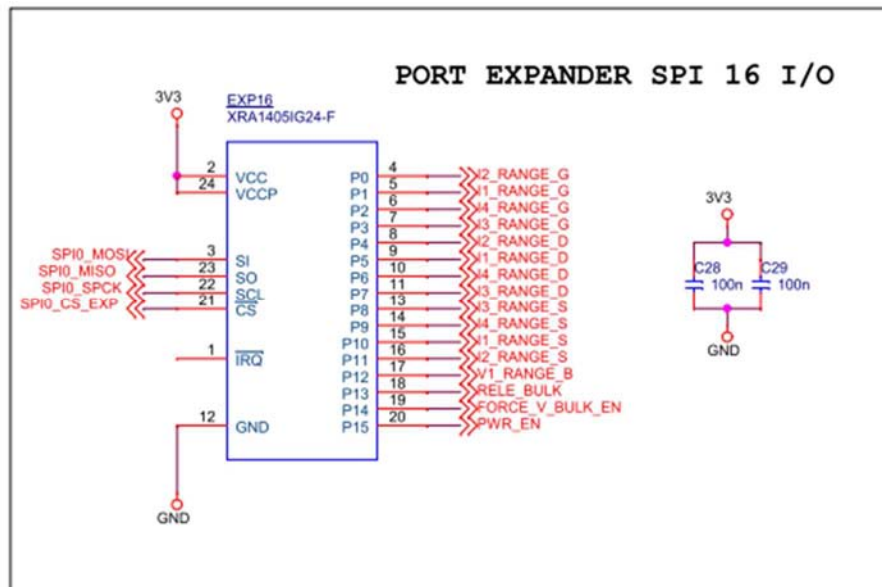
2.1.8 PERIFERICHE

Sono state collegate al microcontrollore diverse periferiche. Tre di queste usano una comunicazione SPI e sono un port-expander, un RTC e una memoria flash. Attraverso il segnale di chip-select è possibile scegliere con quale di queste periferiche iniziare uno scambio di dati.

Il port-expander XRA1405IG24-F della EXAR permette di avere ulteriori sedici GPIO collegati al microcontrollore. Viene alimentato con una tensione di + 3.3V ed è possibile configurare, scrivere o leggere attraverso specifici comandi descritti nel datasheet i vari GPIO che sono divisi in 2 porte da 8 segnali ciascuna.

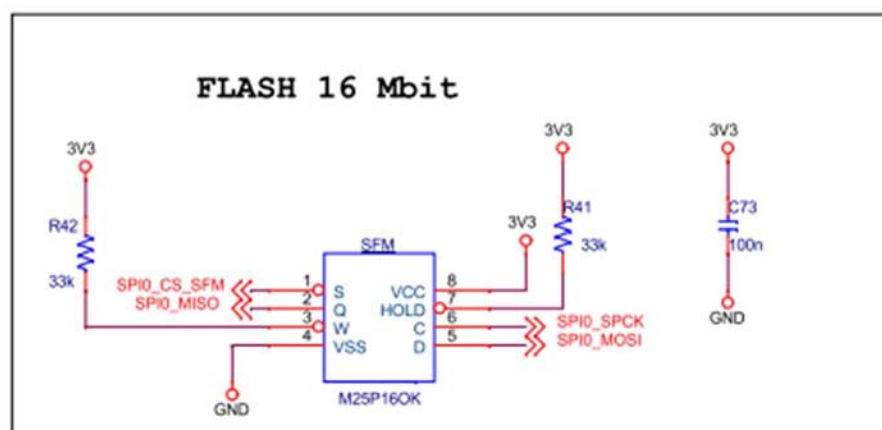
In questo progetto tutti i segnali sono stati configurati come delle uscite e oltre al segnale PWR_EN che abilita nel circuito l'alimentazione della parte

analogica, tutti gli altri segnali sono i comandi di chiusura e apertura dei vari relè e foto-accoppiatori.



2.26 Connessioni port-expander

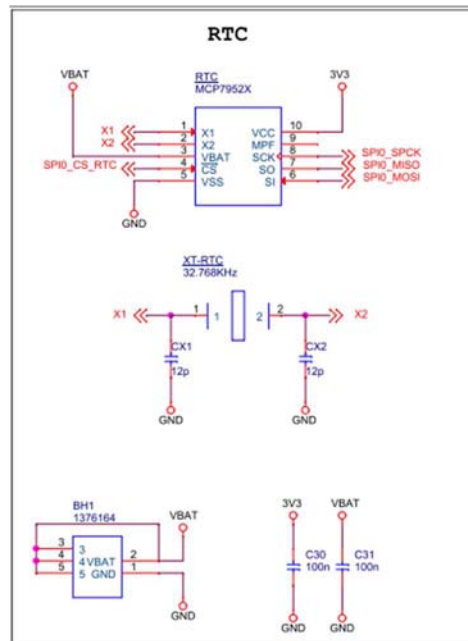
La memoria flash M25P16-VMN6P della Micron è una memoria da 16Mb non volatile ed è utilizzata per memorizzare le impostazioni del programma e i dati che vengono riutilizzati per le varie misure. L'alimentazione è a +3.3V. la programmazione della memoria avviene per pagine da 256byte e la frequenza massima di clock della comunicazione SPI è di 75MHz.



2.27 Connessioni Flash

L'altra periferica SPI è un RTC, il MCP79520-IMS della Microchip. Utilizza un clock esterno da 32.768KHz. L'alimentazione anche per questo dispositivo è di +3.3V ed è in grado di mantenersi aggiornato grazie ad una batteria da +3V anche quando viene a mancare l'alimentazione della scheda. La sua funzione principale è quella di tenere aggiornata la data e

l'ora reale per poterle salvare, insieme ai dati, durante le diverse misurazioni.

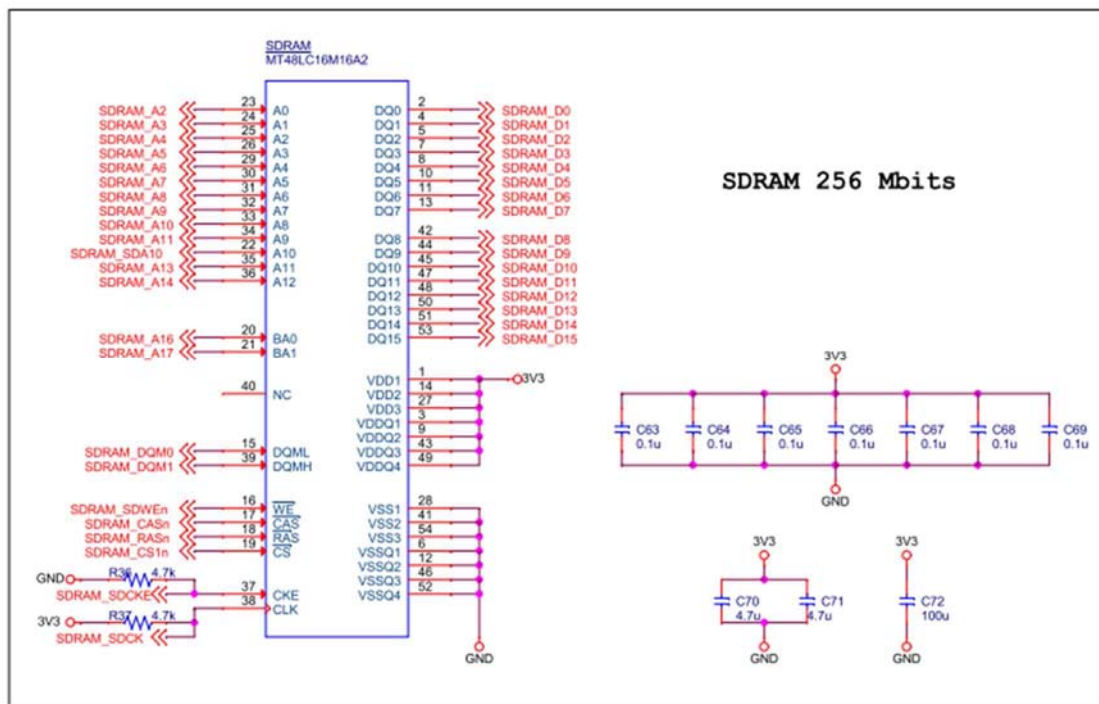


2.28 Connessioni RTC

Dato il grande numero di dati che devono essere memorizzati durante l'esecuzione del programma, come ad esempio i campioni acquisiti dall'ADC in fase di misurazione o i valore che il DAC deve imporre alle varie SMU, la memoria interna del microcontrollore non è sufficiente.

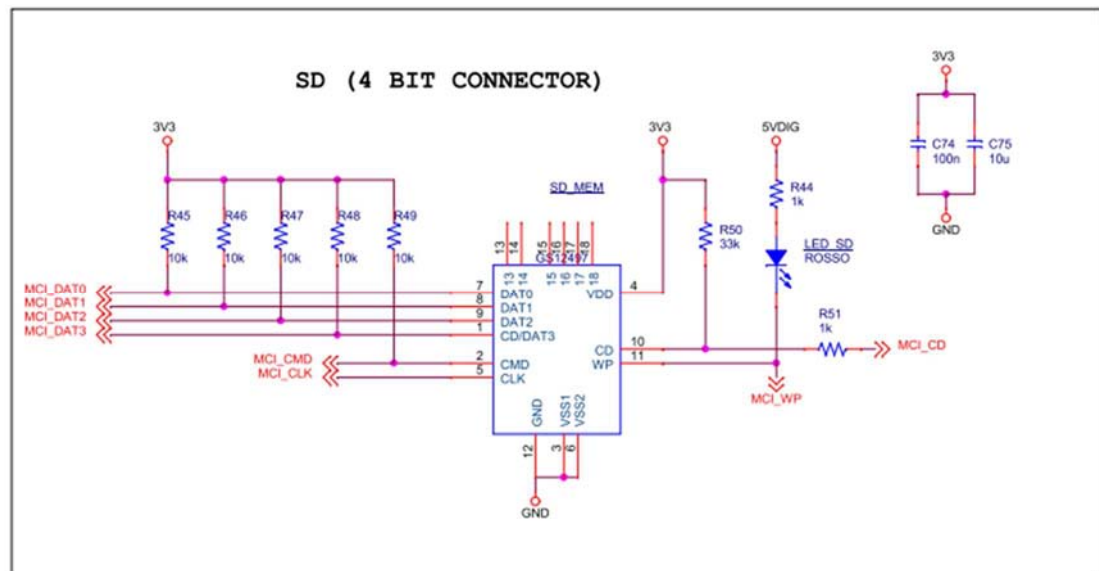
I vari dati devono poter essere salvati o letti da una memoria nel modo più veloce possibile, per questo viene utilizzata una memoria SDRAM da 256Mb.

È stata scelta una MT48LC16M16A2P della Micron con la solita alimentazione a +3.3V. In questa memoria vengono salvati, durante le misure, tutti i valori letti dall'ADC e le successive elaborazioni, le varie tensioni che il DAC deve imporre alle SMU e anche i caratteri inseriti da tastiera che vengono interpretati dal programma come comandi esterni per effettuare misure o altre funzioni memorizzate.



2.29 Connessioni SDRAM

Un altro dispositivo utile per poter trasferire in modo comodo dati da pc alla PCB, è la scheda di memoria SD, per questo motivo è stato installato un connettore per questo tipo di memoria.

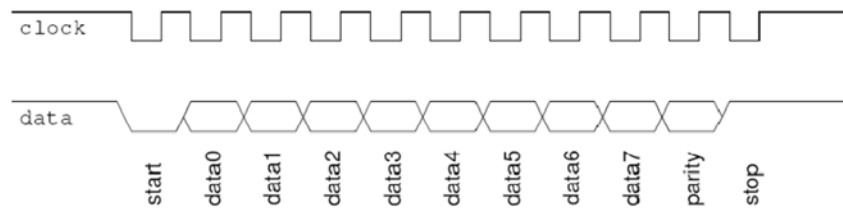


2.30 Connessioni memoria SD

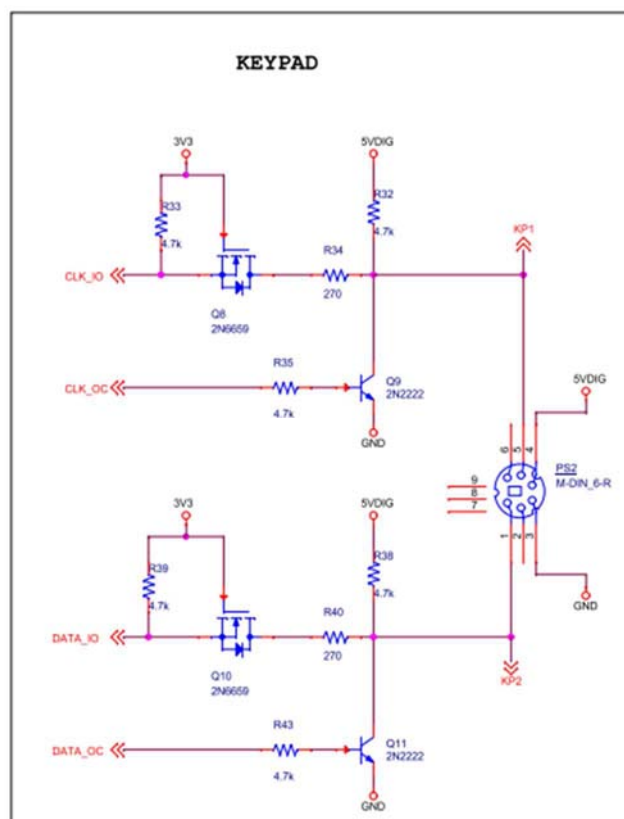
Per poter inserire i comandi è stato predisposto un connettore per tastiera PS/2. La connessione è seriale e prevede una linea di clock e una di dati, oltre che all'alimentazione e massa. La comunicazione tra microcontrollore e tastiera può essere bidirezionale, ma in questo progetto viene usata solamente la comunicazione da tastiera verso il microcontrollore. In questo

caso la tastiera invia i dati un byte alla volta e ogni byte viene inserito in un frame di 11bit:

- 1 bit di start,
- 8bit di dati,
- 1bit di parità,
- 1bit di stop.



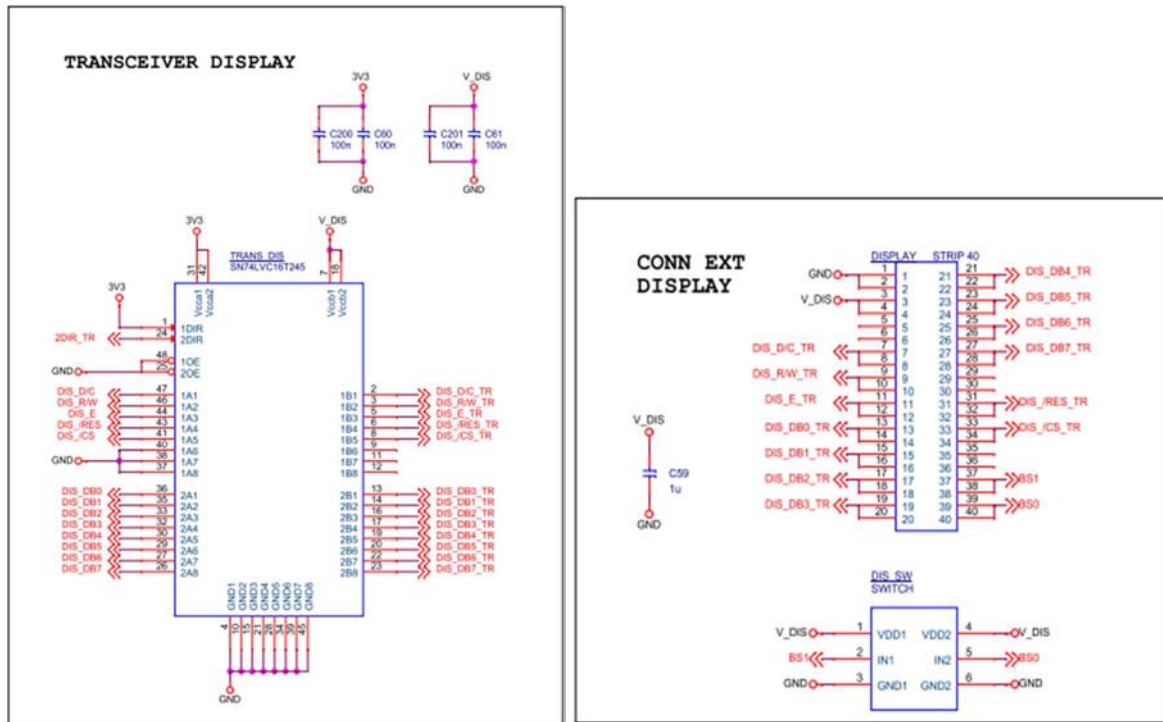
2.31 Protocollo PS2



2.32 Collegamenti PS2

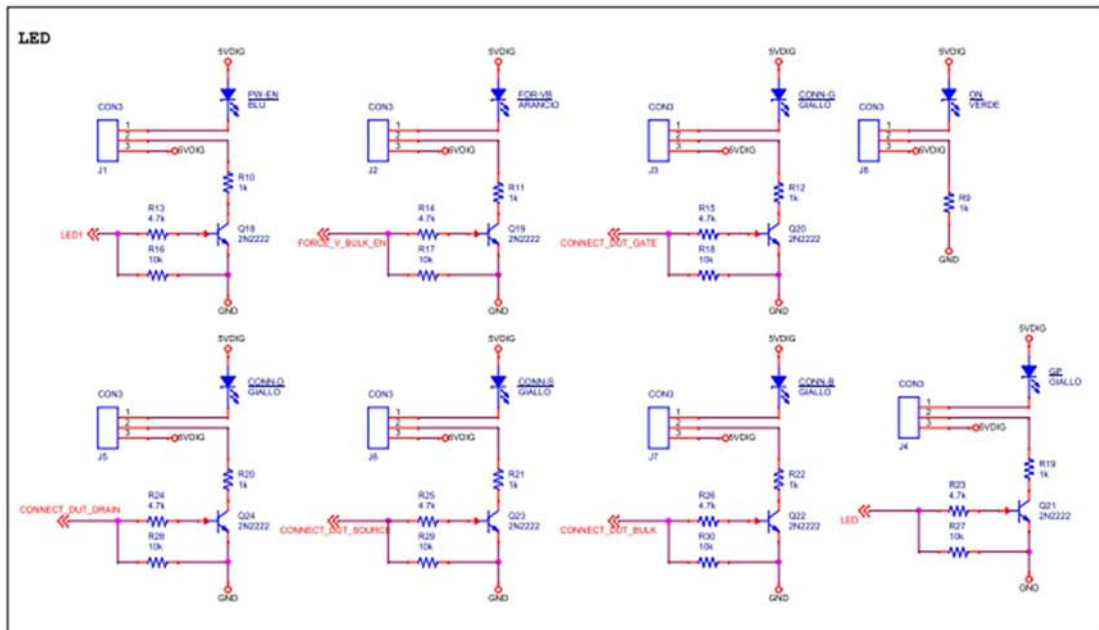
Soprattutto in fase di debug ma anche per avere un'informazione immediata riguardante lo stato del sistema, sia durante la programmazione che durante il funzionamento finale, sono stati inseriti un display OLED e alcuni LED SMD sulla scheda.

Il display NHD-2.8-25664UCB2 della NewhavenDisplay necessita di un'alimentazione a +2.9V ed è stato necessario quindi interporre tra il microcontrollore e il display un dispositivo, il SN74LVC16T245DGGR ovvero un dual-supply bus transceiver. Questo dispositivo è un traslatore di livello, riceve in ingresso le linee provenienti dal microcontrollore e in uscita invia gli stessi segnali con la tensione modificata adattata al dispositivo in uscita. Richiede quindi una doppia alimentazione, a +3.3V e a +2.9V.



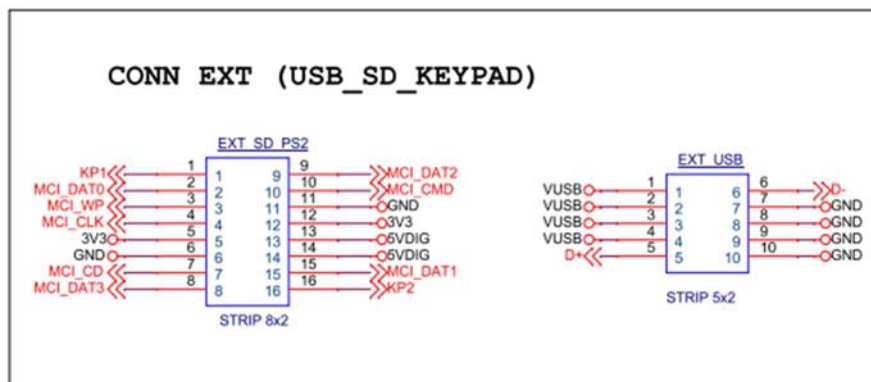
2.33 Collegamenti display

Sono presenti otto LED sulla scheda. Quattro sono usati per indicare quando vengono connesse al rispettivo BNC le quattro SMU di lettura, un LED è usato per indicare quando la SMU di lettura della tensione impone la tensione all'esterno, uno è stato utilizzato come general purpose durante la fase di debug del firmware, uno per indicare la connessione dell'alimentazione alla parte analogica e l'ultimo per indicare quando la scheda viene alimentata.



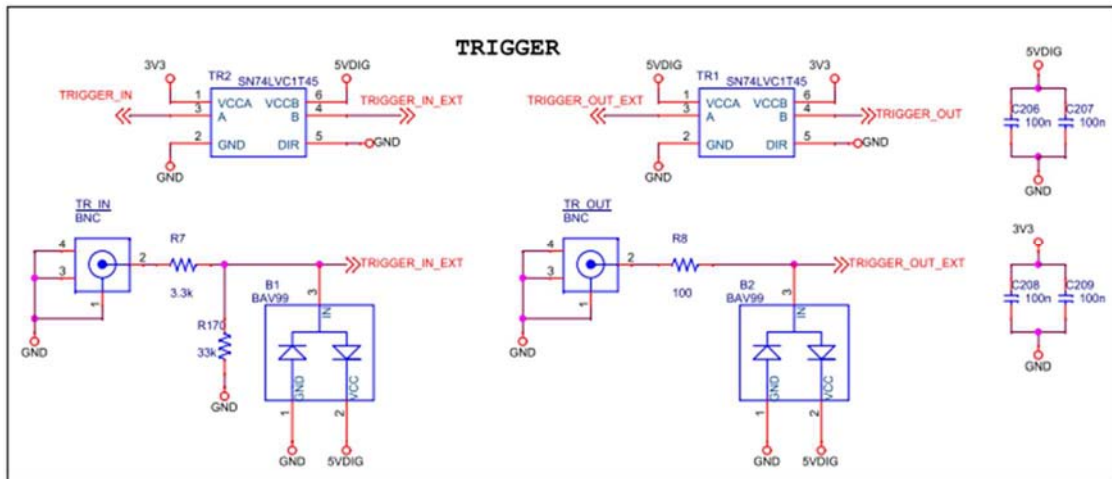
2.34 Collegamenti LEDs

È possibile attraverso un connettore e dei jumper, portare all'esterno della scheda i segnali della USB, della memoria SD, della keyboard e dei LED.



2.35 Connettori per segnali USB, SD, PS2

La scheda è stata predisposta di un circuito di trigger. Durante le misure è possibile sia ricevere dall'esterno, sia generare e inviare a un altro strumento un segnale di trigger. Anche in questo caso, come per il display, il segnale entrante e uscente necessita di un traslatore di livello da 3.3V a 5V, per questo è stato utilizzato un SN74LVC1T45DBVR. Inoltre sono stati inseriti due dispositivi di sicurezza contro le scariche elettrostatiche, i BAV99.



2.36 Circuiti di trigger

2.1.9 ALIMENTAZIONI

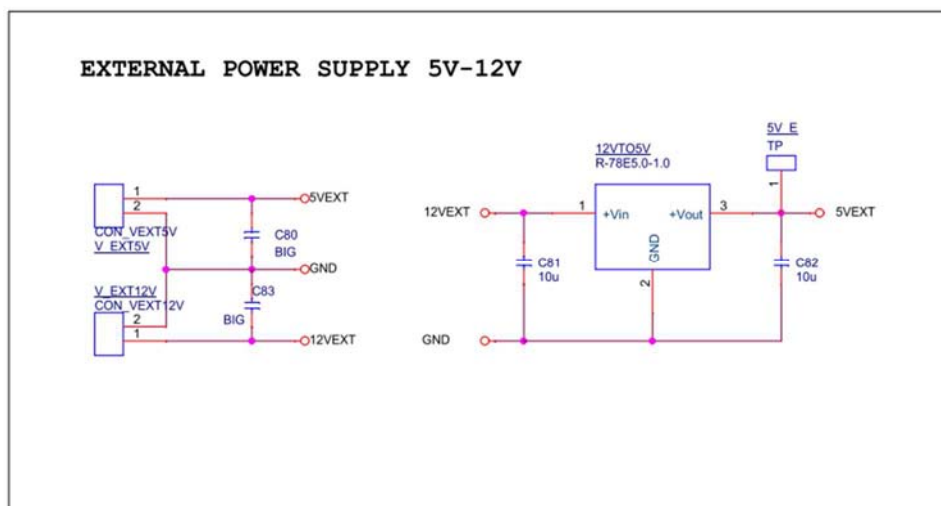
L'alimentazione del circuito può essere scelta tra un'alimentazione esterna da + 12V oppure direttamente dalla USB da + 5V.

La tensione di 12V è stata scelta per avere la possibilità di collegare alla scheda attraverso opportuni connettori delle ventole di raffreddamento per una miglior dissipazione di calore.

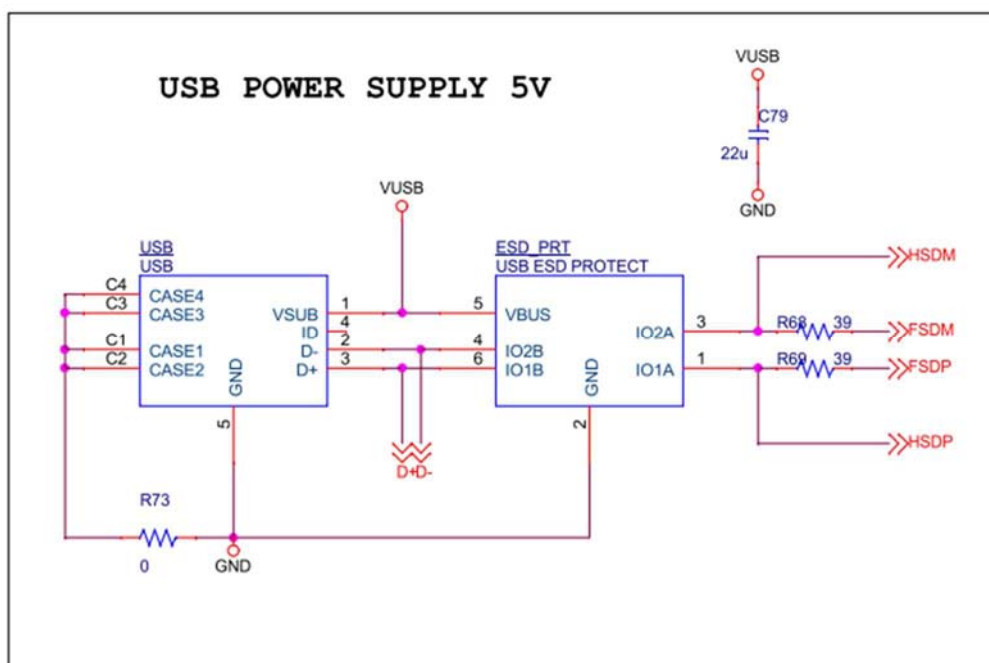
Questa tensione entra direttamente in un Recom, un Buck Converter a tre pin che abbassa la tensione a 5V.

Dal connettore USB invece, è possibile ricavare direttamente i 5V necessari.

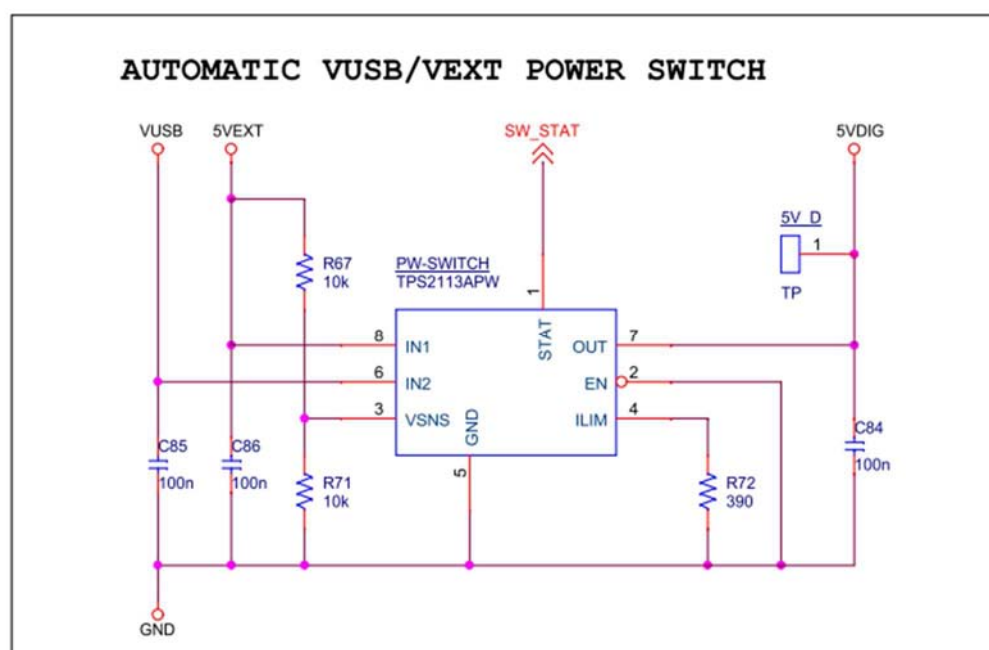
Entrambe queste tensioni entrano in uno switch TPS2113APWR, il quale, se entrambe le tensioni sono presenti, seleziona la tensione esterna come prioritaria. In uscita da questo switch si hanno quindi i 5V necessari per alimentare il circuito.



2.37 Ingresso tensione 12V



2.38 Ingresso connettore USB



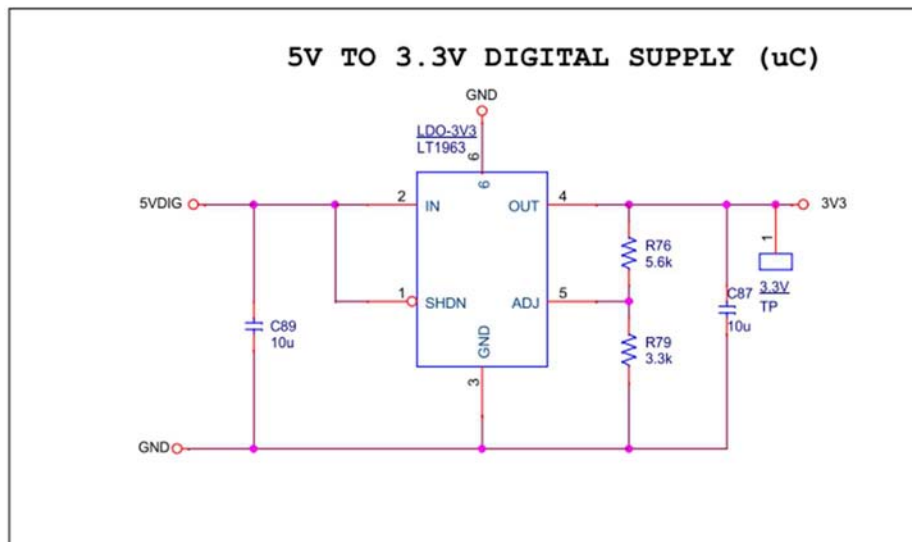
2.39 Power switch EXT/USB

Per alimentare la parte digitale del circuito è necessario ricavare i 3.3V dalla tensione di ingresso. Per fare ciò è stato utilizzato un regolatore lineare LT1963AEQ della Linear Technology. Si tratta di un LDO (Low Drop Out) ovvero un dispositivo che abbassa la tensione di ingresso ad una tensione di uscita di poco inferiore con una relazione tra ingresso/uscita di tipo lineare. Questo permette di avere una tensione di uscita stabile e precisa.

Per poter selezionare la tensione di uscita del regolatore è sufficiente selezionare il rapporto di due resistenze:

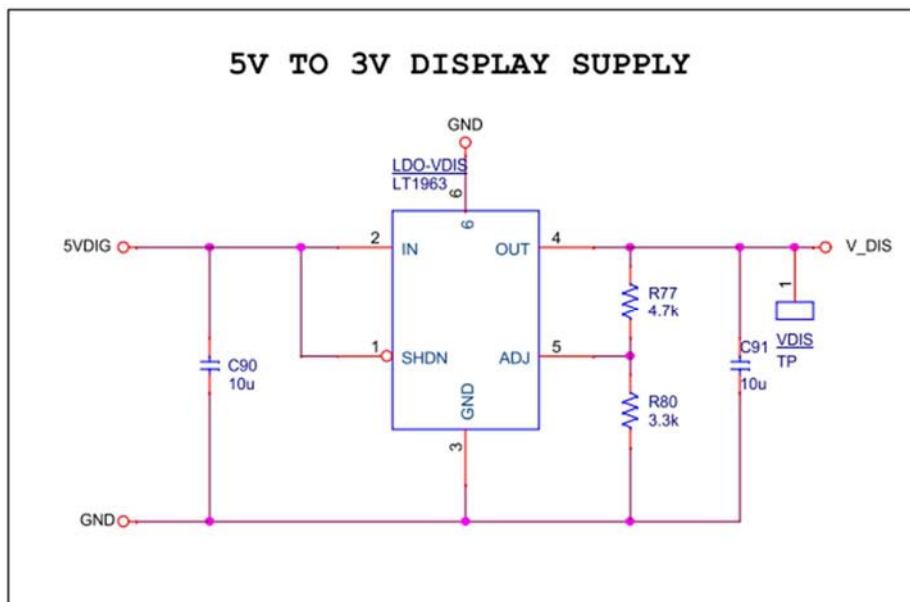
$$V_{OUT} = V_{REF} \left(1 + \frac{R_{76}}{R_{79}} \right), V_{REF} = 1.21V$$

È stato scelto di cortocircuitare il pin di SHDN con l'alimentazione di ingresso per mantenere il dispositivo sempre attivo.



2.40 Generazione 3.3V

Un altro circuito identico a parte il valore delle resistenze del partitore è stato inserito per poter alimentare a 2.9V il display oled.



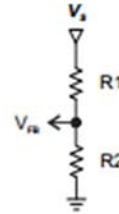
2.41 Generazione alimentazione display

Nella parte analogica del circuito sono necessari i +5V e i -5V per alimentare tutti gli operazionali, e per poter disporre di queste tensioni sono necessari due diversi circuiti.

I +5V provenienti dall'alimentazione esterna entrano in un Boost con controllo di corrente di picco, un TPS61085DGKR. Attraverso un partitore costituito da due resistenze si ricava una tensione di riferimento che entra nel feedback del boost e la tensione di uscita si porterà a questa tensione di riferimento che in questo caso è stata scelta di circa +5.6V.

$$R2 = \frac{V_{FB}}{70\mu A} \approx 18k\Omega \quad R1 = R2 \cdot \left(\frac{V_s}{V_{FB}} - 1 \right)$$

$$V_{FB} = 1.238V$$

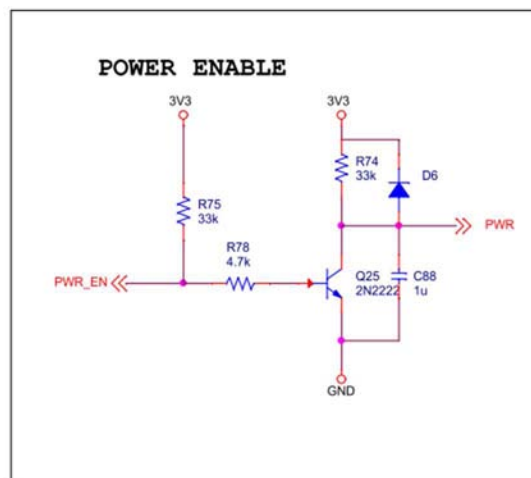


È stata fatta questa scelta per poter utilizzare in uscita un regolatore lineare come in precedenza. La tensione in uscita dal regolatore sarà nuovamente abbassata a +5V e filtrata, in questo modo si ha una tensione precisa e stabile necessaria per alimentare i circuiti analogici.

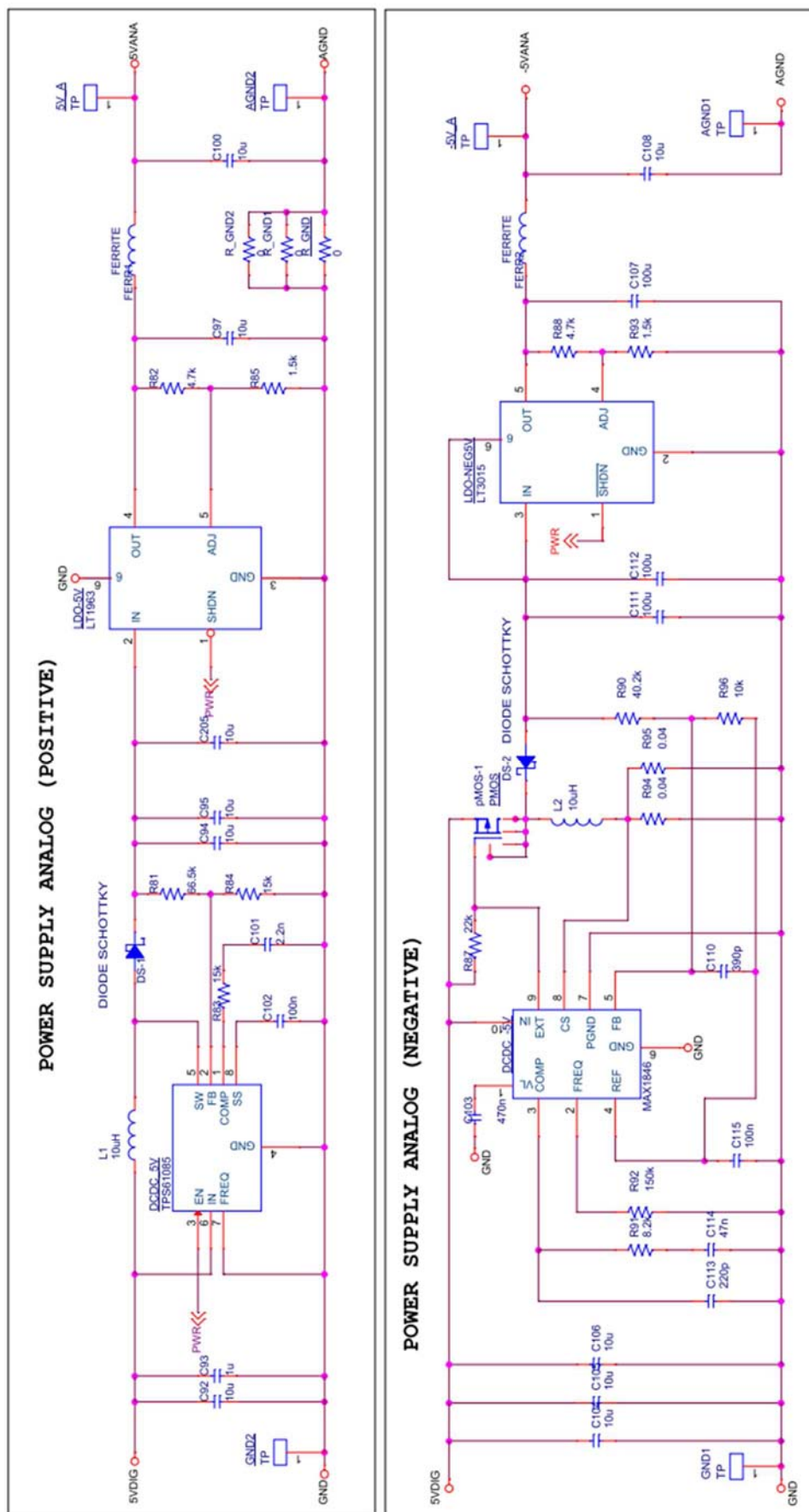
La tensione negativa è stata ricavata da un convertitore Buck-Boost. Questo circuito riceve in ingresso la tensione di 5V esterna e come per il boost, andando a selezionare il rapporto del partitore di resistenze in uscita porta la tensione a un valore poco inferiore ai -5V.

$$R_1 = R_2 \left(-\frac{V_{OUT}}{V_{REF}} \right), V_{REF} = 1.25V, R_2 = 10k\Omega$$

Un regolatore lineare LT3015EQ infine, stabilizza la tensione a -5V. Entrambi i regolatori hanno un segnale di abilitazione controllato dal microcontrollore. Solo quando questo segnale viene pilotato a livello logico basso, il circuito di alimentazione da tensione alla parte analogica.



2.42 Abilitazione alimentazione analogica



2.43 Alimentazione analogica

2.2 PROGETTAZIONE LAYOUT

Una fase molto importante per il corretto funzionamento della scheda, è la progettazione del layout fisico. Si decidono i package e la disposizione dei componenti, la lunghezza e la larghezza delle piste, e i piani di alimentazione e di massa.

Per prima cosa sono stati scelti i package dei dispositivi e i connettori da utilizzare. Sono stati disegnati i pad e successivamente il footprint di tutti i componenti. Per poter collegare i DUT allo strumento si vogliono utilizzare dei cavi coassiali, quindi sono stati inseriti dei connettori BNC per collegare le varie smu ai dispositivi di misura.

Il PCB progettato è composto da quattro layers per la disposizione delle piste.

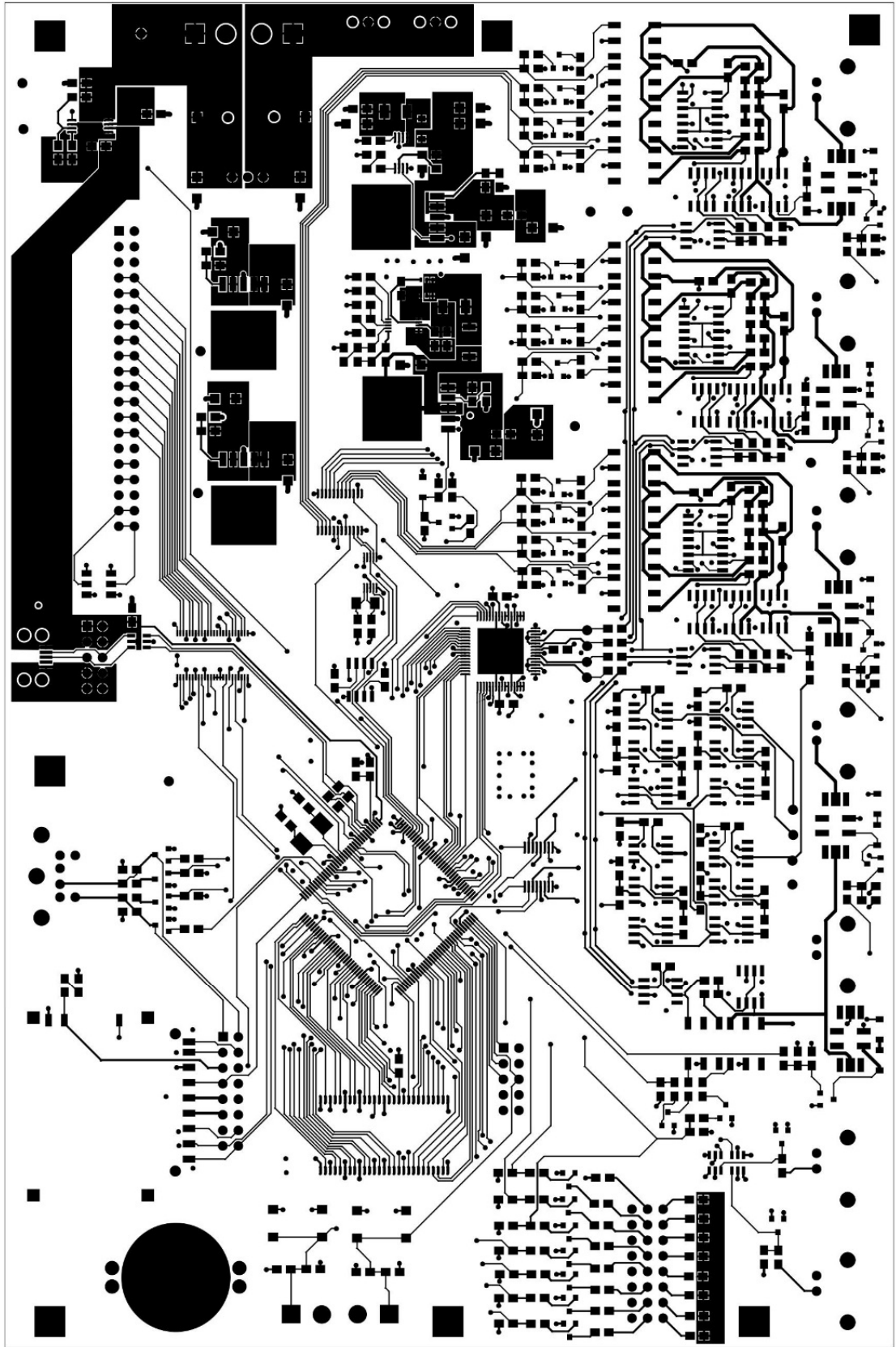
Il primo layer, ovvero il top della scheda, è stato utilizzato per disporre gran parte dei componenti e delle piste e i copper di alimentazione esterna di +12V, +5V e l'alimentazione proveniente dall'USB. La parte analogica del circuito è stata inserita tutta in questo lato della scheda per avere le piste più corte ed avere quindi meno disturbi. Oltre a fare le piste corte, nella parte analogica si è usata una larghezza di 10mils per i rami di circuito dove si ha il maggior passaggio di corrente, mentre per le altre si è usata una larghezza di 8mils. Per le piste di alimentazione si è cercato di farle più larghe possibili, di solito larghe quanto il pad a cui sono collegate.

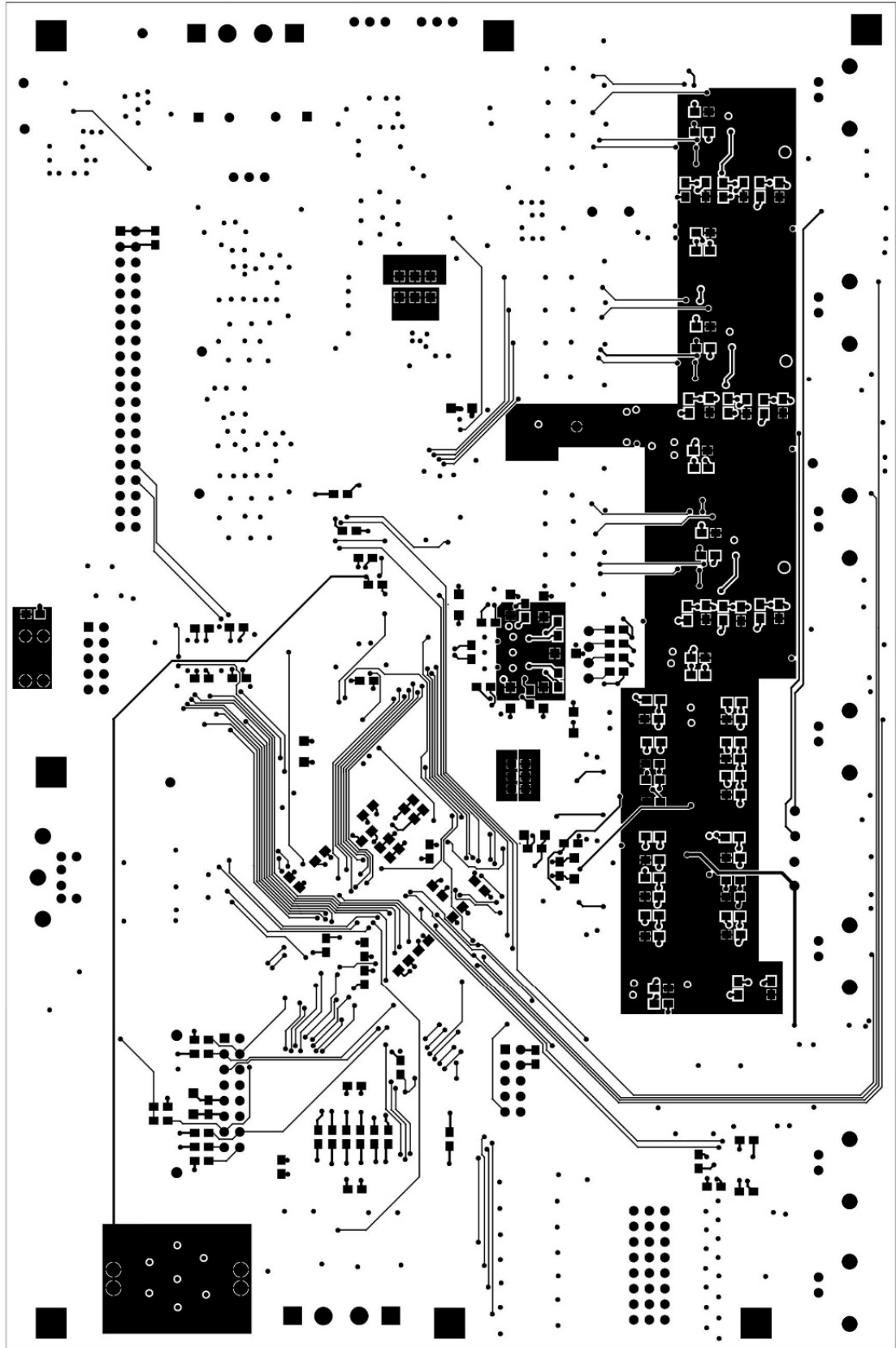
Nel quarto layer, il bottom, sono stati collocati i condensatori di by-pass, alcune piste, e il copper dell'alimentazione analogica di -5V.

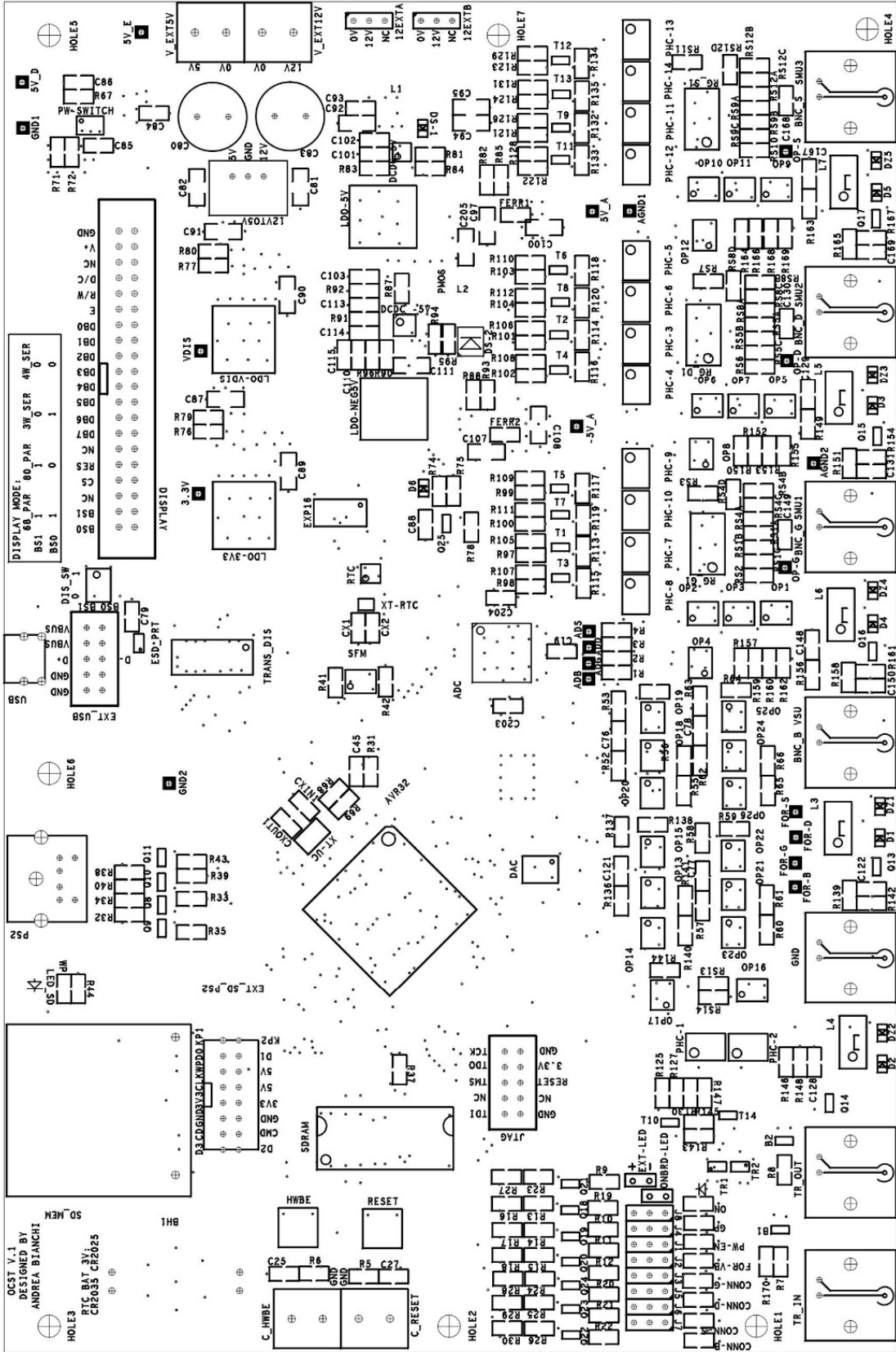
Nel secondo layer, uno dei due layer interni, sono presenti due piani di massa, quello per la massa digitale e quello per la massa analogica. I componenti del circuito analogico sono stati collocati sopra il piano di massa analogica, mentre i componenti della parte digitale sul piano di massa digitale. I due copper sono collegati solo in un punto della scheda da delle resistenze da 0Ω , questo per separare il più possibile i segnali analogici dai disturbi provenienti dai segnali digitali.

Nel terzo layer sono stati inseriti i vari piani di alimentazione, i +5V esterni, i +3.3V per la maggior parte dei dispositivi digitali, i +2.9V per il display e i +5V per la parte analogica.

Una volta finito il layout si è completato il lavoro con l'aggiunta della silkscreen, utile poi nella saldatura dei componenti e in fase di debug, e della soldermask. Sono stati creati quindi i file gerber e inviati ad una azienda specializzata per la realizzazione fisica della PCB.



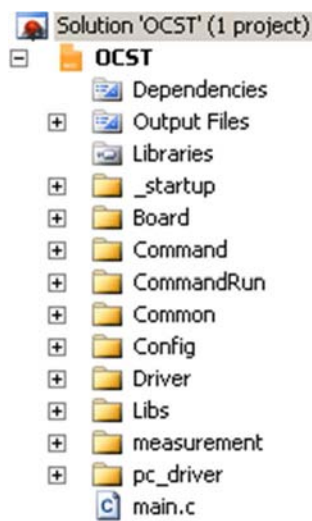




CAPITOLO 3

PROGETTAZIONE FIRMWARE

Il programma è stato organizzato in diversi file. Ognuno di questi file contiene funzioni, costanti e variabili che riguardano una certa parte del programma.



Immagine

Nella cartella *Board* sono presenti i file di configurazione e inizializzazione dei vari pin del microcontrollore e dei dispositivi, vengono inizializzati gli interrupt ed è presente un file `system.c` dove sono presenti le funzioni che gestiscono i timer del microcontrollore, il dispositivo RTC e dove viene dichiarata la SDRAM e inizializzati tutti gli array della memoria stessa.

In *Command* si ha la gestione di inserimento comandi da USB e da PS2, mentre in *CommandRun* sono presenti le funzioni che eseguono i comandi.

Nella cartella *Common* sono incluse le librerie standard del linguaggio C e sono presenti tutte le funzioni a basso livello del linguaggio di programmazione.

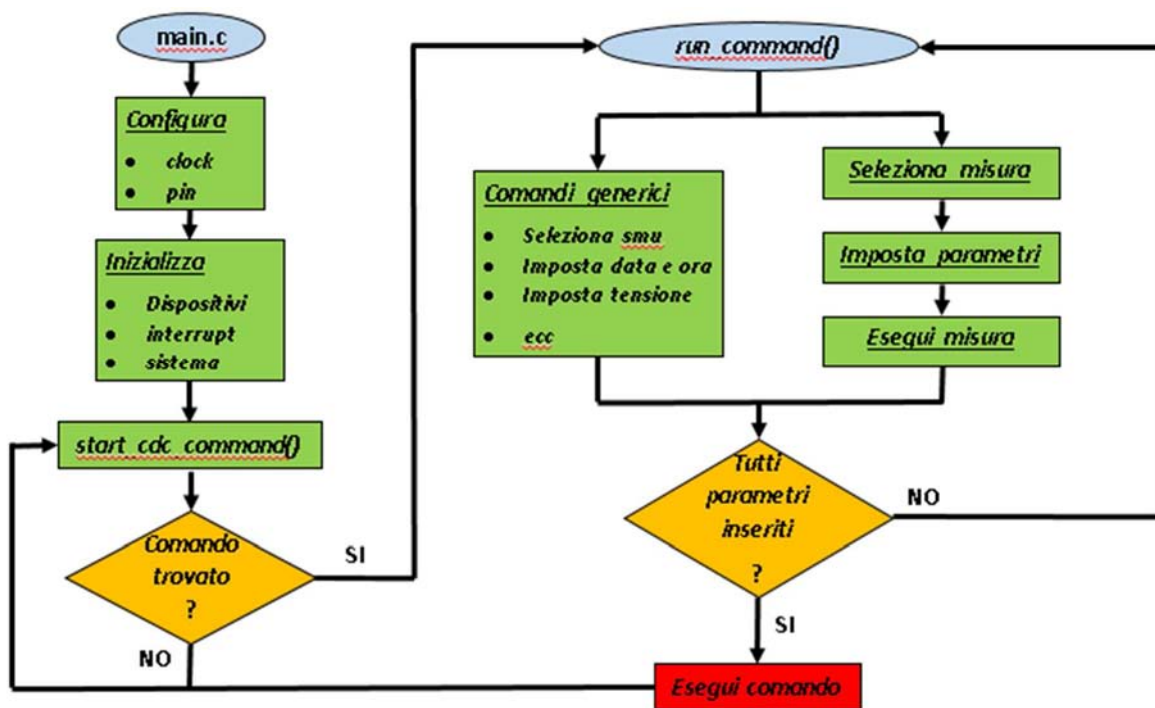
In *Config* si hanno i driver di accesso alla memoria e alcune funzioni di configurazione di registri del microcontrollore.

Nella cartella *Driver* sono presenti tutte le funzioni che gestiscono ogni periferica presente nel circuito mentre in *Libs* sono presenti diverse librerie che vengono usate nel progetto.

Infine nella cartella *measurement* vengono dichiarate, inizializzate e implementate le funzioni che eseguono le tre diverse misure.

3.1 STRUTTURA FIRMWARE

Il firmware è stato implementato in linguaggio C++ ed è composto da diversi file che contengono le funzioni che servono per far comunicare tra loro i dispositivi della scheda. La programmazione avviene attraverso diversi livelli, dal più basso, ovvero la programmazione del microcontrollore e dei protocolli di comunicazione, fino alle funzioni che implementano le misure da effettuare.



Il file principale è il **main.c** che contiene variabili globali di configurazione, come ad esempio la frequenza di clock del microcontrollore, e la funzione **main()**, ovvero la funzione principale del programma. Da qui vengono richiamate le funzioni di inizializzazione e di configurazione dell'intero sistema.

```

main.c

#include "board\board.h"
#include "Driver\sram\sram.h"

// Variabili globali di configurazione (dichiarate in config\conf_system.h)
DWORD fcpu_hz = 60000000;
DWORD fpba_hz = 60000000;
DWORD fpbb_hz = 60000000;

extern void _evba; // Exception Vector Base Address.

int main(void)
{
    __builtin_mtsr(AVR32_EVBA, (int)&_evba); // Carica il vettore degli interrupt.
    Enable_global_exception();                // Enable exceptions.
    INTC_init();                              // Initialize interrupt handling.

    configure_clock();                        // Configure system clocks.
    configure_usb_clock();                    // Configurazione clock USB
    configure_pin();                          // Configurazione dei pin del micro
    devices_init();                          // Inizializzazione di tutti i dispositivi
    enable_interrupt();                       // Inizializzazione interrupt
    system_init();                           // Inizializzazione sistema
    start_cdc_command();                     // Modalità di inserimento comandi

    return 0;
}

```

3.1 Funzione main

Le funzioni *INTC_init()*, *configure_clock()* e *configure_usb_clock()*, sono il livello più basso di configurazione, infatti con queste funzioni si vanno ad inizializzare le caratteristiche di comunicazione principali del microcontrollore.

Successivamente nella funzione *configure_pin()* tutti i pin del microcontrollore vengono prima resettati, poi configurati come ingressi o uscite o con una particolare funzione e infine, se necessario, inizializzati ad un valore logico alto o basso di default. Di seguito viene riportato l'esempio per configurare i pin che gestiscono la connessione delle varie SMU attraverso il relè.

```

// RESET
void reset_pin_configuration()
{
    int i;
    for (i = 0; i < N_PORT; i++)
    {
        AVR32_GPIO.port[i].gpers = 0xFFFFFFFF; // Setta tutto come gpio
        AVR32_GPIO.port[i].oderc = 0xFFFFFFFF; // Setta tutto come ingresso
        AVR32_GPIO.port[i].puers = 0xFFFFFFFF; // Abilita i pull up
    }
}

```

3.2 Reset dei pin del uC

```

// CONNECT DUT
void configure_connectDUT_pin() //default: nessuna smu connessa
{
    set_pin_as_gpio(CONNECT_DUT_GATE);
    clear_gpio(CONNECT_DUT_GATE);
    set_pin_as_output(CONNECT_DUT_GATE);

    set_pin_as_gpio(CONNECT_DUT_DRAIN);
    clear_gpio(CONNECT_DUT_DRAIN);
    set_pin_as_output(CONNECT_DUT_DRAIN);

    set_pin_as_gpio(CONNECT_DUT_SOURCE);
    clear_gpio(CONNECT_DUT_SOURCE);
    set_pin_as_output(CONNECT_DUT_SOURCE);

    set_pin_as_gpio(CONNECT_DUT_BULK);
    clear_gpio(CONNECT_DUT_BULK);
    set_pin_as_output(CONNECT_DUT_BULK);
}

```

3.3 Esempio configurazione pin

Dopo aver configurato tutti i pin, vengono inizializzati i dispositivi attraverso la funzione *device_init()*. In questa funzione vengono inizializzati tutti i segnali dei vari dispositivi al livello di default e le varie caratteristiche delle SPI utilizzate nel circuito, come ad esempio la polarità, la fase e la frequenza del clock e il numero di bit che vengono inviati per ogni singola trasmissione.

```

// CONFIGURA DEVICES
void devices_init()
{
    adc_init();
    dac_init();
    mci_resources_init(); // Inizializza la SD

    spi_master_init(&AVR32_SPI0); // Inizializza l'SPI0 come master
    spi_csr0_config(&AVR32_SPI0, RTC_CLK, RTC_POL, RTC_PHASE, RTC_N_BIT); //Inizializzo le periferiche
    spi_csr1_config(&AVR32_SPI0, SFM_CLK, SFM_POL, SFM_PHASE, SFM_N_BIT);
    spi_csr2_config(&AVR32_SPI0, EXP_CLK, EXP_POL, EXP_PHASE, EXP_N_BIT);
    spi_master_init(&AVR32_SPI1);
    spi_csr1_config(&AVR32_SPI1, DAC_CLK, DAC_POL, DAC_PHASE, DAC_N_BIT);

    port_expander_init();
    SDRAMC_Init(fpbb_hz);
    oled_init();
    oled_start(&display);
    init_kb();
    real_time_clock_init(0);
}

```

3.4 Configurazione dispositivi

Nella *enable_interrupt()* vengono poi inizializzati gli interrupt, ovvero segnali asincroni che interrompono il processo per svolgere una particolare funzione con una maggiore priorità di una periferica. Nel programma vengono utilizzati gli interrupt per far partire l'acquisizione dell'ADC attraverso un segnale modulato da un timer, o in generale per indicare la fine del conteggio di un timer o la ricezione di un particolare tipo di segnale.

```

enable_interrupt void enable_interrupt()
void enable_interrupt()
{
    Disable_global_interrupt();
    INTC_register_interrupt(interrupt_adc_acq, AVR32_EIC_IRQ_0, AVR32_INTC_INT1);
    INTC_register_interrupt(interrupt_ps2, AVR32_EIC_IRQ_2, AVR32_INTC_INT0);
    INTC_register_interrupt(dac_timer_cv, AVR32_TC1_IRQ1, AVR32_INTC_INT2);
    INTC_register_interrupt(dac_timer_impulse, AVR32_TC0_IRQ2, AVR32_INTC_INT2);
    INTC_register_interrupt(dac_timer_step, AVR32_TC1_IRQ2, AVR32_INTC_INT2);
    INTC_register_interrupt(system_rtc, AVR32_RTC_IRQ, AVR32_INTC_INT0);
    Enable_global_interrupt();

    //interrupt EIC su fronte discesa EOC
    AVR32_EIC.MODE.int0 = 0; // Interrupt sul fronte
    AVR32_EIC.IER.int0 = 1; // Interrupt enable register
    AVR32_EIC.EN.int0 = 1; // Enable register
    AVR32_EIC.EDGE.int0 = 0; // Interrupt triggers on falling edge
    // INTERRUPT PS2
    AVR32_EIC.MODE.int2 = 0;
    AVR32_EIC.IER.int2 = 1;
    AVR32_EIC.EN.int2 = 1;
    AVR32_EIC.EDGE.int2 = 0;
    // INTERRUPT TIMER DAC CV
    timer_enable_interrupt_c(&AVR32_TC1, 1);
    // INTERRUPT TIMER IMPULSE
    timer_enable_interrupt_c(&AVR32_TC0, 2);
    // INTERRUPT TIMER STEP
    timer_enable_interrupt_c(&AVR32_TC1, 2);
    // INTERRUPT RTC PS2
    AVR32_RTC.IER.topi = 1;
}

```

3.5 Configurazione interrupt

La funzione *system_init()* è la funzione che inizializza il sistema al livello più alto. Qui vengono inizializzati i puntatori che puntano alle diverse locazioni di memoria della sdram per poter scrivere e leggere i memoria, e vengono resettati i valori delle diverse misure. Viene inoltre stampato sul display il tipo di alimentazione della scheda, ovvero se la si sta alimentando con un alimentatore esterno o tramite USB.

```

void system_init()
{
    if (is_gpio_high(SW_STAT)) { oled_printf(&display,15,FALSE,"Alim. USB\n"); }
    else { oled_printf(&display,15,FALSE,"Alim. EXT\n"); }

    sdram_p = (volatile SDRAM_ALLOC *)SDRAM_Address;
    init_usb_buffer();
    init_ps2_buffer();
    init_adc_buffer();
    init_val_adc();
    init_adc_range_buffer();
    init_dac_buffer();

    measure_init();
}

```

3.6 Configurazione sistema

L'ultima funzione del main è *start_cdc_command()*. Questa funzione inizializza l'interrupt dell'USB, configura i comandi per l'esecuzione ed

entra in attesa del comando proveniente da USB, quindi da terminale, o da PS/2, cioè da tastiera esterna, grazie alla funzione `cdc_command()`.

```
void start_cdc_command()
{
    usb_task_init();
    device_cdc_task_init();
    config_commands(); // Configura i comandi
    cdc_command();
}
```

3.7 Gestione comandi

In questa funzione il programma entra in un loop infinito attraverso il costrutto `while(1)` dove attende l'inserimento del comando da parte dell'operatore.

Per inserire un comando come regola generale è stata scelta quella di scrivere una stringa identificativa del comando seguita da uno spazio e da tutti i valori delle variabili che vengono modificate attraverso quel specifico comando separate da uno spazio.

Il programma entra in uno dei due costrutti `if()` attraverso le funzioni `is_cmd_usb_ready()` o `is_cmd_ps2_ready()` che vengono settate al valore 1 se viene ricevuto un carattere di fine comando rispettivamente da USB o tastiera esterna, o se il buffer che contiene i caratteri di comando è pieno. Se si entra all'interno di uno di questi `if()` il comando è stato inserito e si passa alla decodifica della stringa attraverso la funzione `command_decode(volatile char *command_buffer)` a cui viene passato l'indirizzo del buffer di caratteri che vengono inseriti in SDRAM. Qui viene assegnato al buffer di caratteri `command[]` la stringa del comando e viene fatto puntare il puntatore `arg_p` al primo carattere inserito dopo lo spazio che segue il comando, ovvero al valore della prima variabile.

Se la stringa inserita corrisponde ad un comando si passa alla funzione `run_command(char *command)` che attraverso uno `switch() case:` richiama una funzione per l'esecuzione effettiva del comando.

```

cdc_command.while.if  if(ps2_ric == 1)
void cdc_command()
{
    while(1)
    {
        if (system_rtc_ms == 0)
        {
            system_rtc_ms = 1000;
            system_rtc_update();
        }
        usb_task();
        device_cdc_task();
        if (is_cmd_usb_ready())
        {
            command_decode(usb_buffer);
            if (run_command(command) == 0)
                usb_printf("Comando non eseguito: %s\n\n", command);

            usb_buffer_counter = 0;
            cmd_usb_ready = FALSE;
        }
        if(ps2_ric == 1)
        {
            ps2_ric = 0;
            if (ps2_ch == ASCII_CR || ps2_ch == ASCII_LF || ps2_ch == ASCII_NULL || ps2_buffer_counter == MAX_USB_BUFFER)
            {
                ps2_buffer[ps2_buffer_counter] = '\0';
                if (ps2_buffer_counter > 0)
                    cmd_ps2_ready = TRUE;
            }
            else { ps2_buffer[ps2_buffer_counter++] = ps2_ch; }
        }

        if (is_cmd_ps2_ready())
        {
            command_decode(ps2_buffer);
            run_command(command);

            ps2_buffer_counter = 0;
            cmd_ps2_ready = FALSE;
        }
    }
}

```

3.8 Attesa comando

3.2 DRIVER DISPOSITIVI

Durante l'esecuzione del programma e dei comandi, vengono richiamate diverse funzioni che riguardano i vari dispositivi del progetto. Nella cartella *Driver* sono presenti, oltre alle funzioni che gestiscono il microcontrollore, tutte quelle funzioni che permettono di interagire con questi dispositivi:

- adc
- dac
- flash
- oled
- port expander
- ps2
- rtc
- sdram

- trigger

In ognuno di questi file si gestisce a basso livello il dispositivo, viene cioè configurato e sono presenti le funzioni che verranno usate quando si dovrà svolgere un determinato compito in altre funzioni a più alto livello.

3.2.1 adc.c

In questo file per prima cosa vengono configurati i registri dell'ADC e viene inizializzato al tipo di acquisizione voluto. Questo è fondamentale per poter utilizzare l'ADC correttamente, queste funzioni vengono eseguite appena viene accesa la scheda.

Viene inoltre definito l'interrupt per l'acquisizione e la conversione dei dati. Questo è gestito attraverso l'uscita di un timer del microcontrollore, che attiva l'interrupt sul fronte di discesa del segnale di EOC e salva i dati direttamente in un buffer della sdram.

Il segnale ADC_CONVST è stato assegnato ad un pin di uscita di un timer del microcontrollore. Questo segnale deve rimanere a livello logico alto per un tempo pari a 3 μ s che è il tempo di conversione dei dati acquisiti durante la fase di acquisizione che avviene quando ADC_CONVST è a livello logico basso. Questo tempo di acquisizione varia in base al tempo di campionamento imposto dall'utente e settato tramite la funzione *adc_set_timer()*. Il fronte di salita di ADC_CONVST dà il via alla conversione settando a 1 il segnale ADC_EOC. Quando la conversione è terminata questo segnale viene riportato a livello logico basso e viene generato l'interrupt_adc() che permette di salvare i dati acquisiti in SDRAM attraverso la lettura del valore dei 16 bit ricevuti alla porta PA del microcontrollore.

```
__attribute__((__interrupt__)) void interrupt_adc_acq()
{
    volatile int adc_i;
    AVR32_EIC.ICR.int0 = 1;
    clear_gpio(ADC_CS);
    for(adc_i=0; adc_i<ADC_NUM_IN; adc_i++)
    {
        clear_gpio(ADC_RD);
        wait_ns(ADC_RD_FALL_TO_DATA_VALID);
        adc_DB = AVR32_GPIO.port[ADC_PORT_UC].pvr;
        set_gpio(ADC_RD);

        sdram_p->adc_val[adc_i][n_sample_counter] = ((adc_DB&ADC_MASK_1)>>ADC_OFFSET)|(adc_DB&ADC_MASK_2);

        wait_ns(ADC_RD_HIGH_TIME);
    }
    set_gpio(ADC_CS);
    n_sample_counter++;
}
```

3.9 Interrupt acquisizione dati

Salendo di livello, sono state implementate una funzione di lettura, *adc_read(int adc_in, int n_sample_to_read, int sample0)* e successivamente una funzione che calcola la media dei campioni acquisiti, *adc_avg(int adc_in, int n_sample_tot, int n_sample_avg)*.

La prima funzione permette di leggere un certo numero di campioni, *n_sample_to_read*, acquisiti dall'ingresso dell'adc, *adc_in*, partendo dal campione salvato in posizione *sample0*, e di scrivere in un buffer della sdrum il valore esatto della tensione o della corrente che viene letta dalla SMU con il range selezionati.

La seconda funzione permette di selezionare il numero totale di campioni acquisiti e di calcolarne la media solo di alcuni, *n_sample_avg*.

```
void adc_read(int adc_in, int n_sample_to_read, int sample0)
{
    if (n_sample_to_read > MAX_ADC_BUFFER)
    {
        oled_printf(&display, 15, FALSE, "Errore numero campioni\n");
        return;
    }
    int i;
    switch (adc_in)
    {
        case GATE_ADC:
            for(i = sample0; i < sample0 + n_sample_to_read; i++)
            {
                switch (adc_gate_range_p[i])
                {
                    case RFB_3M:
                        adc_gate_buffer_p[i] = -(sdrum_p->adc_val[GATE_ADC][i]*ADC_V_LSB + voltage_offset)/(4.7*3000000.0);
                        break;
                    case RFB_100K:
                        adc_gate_buffer_p[i] = -(sdrum_p->adc_val[GATE_ADC][i]*ADC_V_LSB + voltage_offset)/(4.7*100000.0);
                        break;
                    case RFB_3K3:
                        adc_gate_buffer_p[i] = -(sdrum_p->adc_val[GATE_ADC][i]*ADC_V_LSB + voltage_offset)/(4.7*3300.0);
                        break;
                    case RFB_100:
                        adc_gate_buffer_p[i] = -(sdrum_p->adc_val[GATE_ADC][i]*ADC_V_LSB + voltage_offset)/(4.7*100.0);
                        break;
                }
            }
        break;
    }
}
```

3.10 Esempio lettura corrente

3.2.2 dac.c

Dato che il dac del progetto utilizza un interfaccia SPI, per inizializzarlo è sufficiente che sia stata inizializzata la periferica SPI del microcontrollore.

In questo file quindi, si trovano solo le funzioni ad alto livello, ad esempio per abilitare e disabilitare il dac o per scrivere una tensione da imporre e una funzione per caricare la tensione preimpostata.

La funzione *dac_write(int num_dac_out, float v_smu)*, riceve come parametri il numero del dac a cui viene chiesto di imporre la tensione e la

tensione che deve essere imposta all'ingresso della SMU. La funzione calcola la corretta tensione di uscita del dac che entra nel circuito di polarizzazione per avere la tensione desiderata alla SMU, converte questo valore nella WORD a 16bit che viene poi inviata al dac tramite la SPI del microcontrollore.

Per dare effettivamente questa tensione in uscita è necessario abilitarla tramite la funzione `dac_load()` che semplicemente abbassa e rialza il segnale DAC_LDAC.

3.2.3 flash.c

Come il dac la flash è un dispositivo SPI. In questo file sono presenti funzioni per abilitare, disabilitare la memoria e per leggere il codice identificativo del dispositivo, utile per verificare che la comunicazione avvenga in modo corretto.

3.2.4 oled.c

Come prima cosa vengono definiti i segnali di ingresso e uscita del display e i pin della porta del microcontrollore alla quale sono collegati.

Sono presenti i driver base per inviare e ricevere un comando, viene definita la scala di grigi per selezionare il colore di stampa e viene inizializzato il display e la sua memoria RAM.

Inoltre sono presenti funzioni che permettono di inviare caratteri e stringhe al display, tutte queste poi vengono utilizzate nella funzione di `oled_print(CLI *console, int color, BIT inverted, char *fmt, ...)` con la quale è possibile scrivere e stampare a display qualunque dato. È stata utilizzata spesso questa funzione in fase di debug del firmware per verificare quale punto del programma si stava eseguendo.



3.11 Esempio debug con display

3.2.5 port_expander.c

Il port-expander utilizzato è una periferica SPI ed aggiunge al microcontrollore 16 GPIO, ovvero è formato da due porte da 8bit ciascuna.

Come prima cosa è necessario configurare le porte del port-expander ad uno stato di default che viene impostato all'accensione della scheda. La funzione `port_expander_init()` permette di fare ciò. Attraverso l'invio di byte sulla SPI è stato possibile impostare le due porte come uscite e il livello logico di ogni uscita.

Altre funzioni a più basso livello permettono di scrivere o leggere le porte del port-expander, mentre altre a più alto livello permettono di cambiare lo stato di una singola uscita.

```
void port_expander_init()
{
    exp_enable();
    spi_send_byte(&AVR32_SPI0, EXP_CS, EXP_GCR1W); //GCR1
    spi_send_byte(&AVR32_SPI0, EXP_CS, 0x00); //abilito tutta la porta[1] come output
    exp_disable();

    exp_enable();
    spi_send_byte(&AVR32_SPI0, EXP_CS, EXP_OCR1W); //OCR1
    spi_send_byte(&AVR32_SPI0, EXP_CS, 0xBB); //0xBB inizializzo la porta[1]
    range_now_g = RFB_100;
    range_now_d = RFB_100;
    exp_disable();

    exp_enable();
    spi_send_byte(&AVR32_SPI0, EXP_CS, EXP_GCR2W); //GCR2
    spi_send_byte(&AVR32_SPI0, EXP_CS, 0x00); //abilito tutta la porta[2] come output
    exp_disable();

    exp_enable();
    spi_send_byte(&AVR32_SPI0, EXP_CS, EXP_OCR2W); //OCR2
    spi_send_byte(&AVR32_SPI0, EXP_CS, 0xDD); //0xDD inizializzo la porta[2]
    range_now_s = RFB_100;
    range_now_b = RB_1K;
    //set_gpio(LED_US1);
    exp_disable();
}
```

3.12 Inizializzazione segnali port-expander

```
void enable_pwr_en()
{
    old_port = exp_read_port(2);
    BYTE part = (old_port & PWR_EN_MASK);
    exp_write_port(2, (part & 0xF0) | (old_port & 0x0F));
    set_gpio(LED_US2);
}

void disable_pwr_en()
{
    old_port = exp_read_port(2);
    exp_write_port(2, (old_port & 0x80));
    clear_gpio(LED_US2);
}
```

3.13 Esempio gestione segnali port-expander

3.2.6 ps2.c

In questo file viene dichiarato l'interrupt per il protocollo ps2 di ricezione dei dati e viene inizializzata tastiera.

È presente inoltre una funzione per la decodifica dei tasti premuti e una funzione che ritorna il valore 1 se viene ricevuto un carattere di fine comando, che viene utilizzata nella funzione `cdc_command()`.

3.2.7 rtc.c

L'rtc viene inizialmente inizializzato attraverso la scrittura nei registri attraverso la funzione `rtc_set_setting()`. Inviando comandi e dati attraverso l'SPI è possibile settare o leggere data e ora memorizzate nell'rtc e in questo file sono presenti tutte questo tipo di funzioni che permettono anche di stampare sul display la data e l'ora aggiornate.

```
void rtc_set_settings()
{
    rtc_settings.calsgn = 0;    // Positive calibration
    rtc_settings.h24 = 0;      // Formato 24h
    rtc_settings.pm = 0;
    rtc_settings.vbat = 0;
    rtc_settings.vbaten = 1;    // VBAT abilitato
}
```

3.14 Esempio inizializzazione segnali RTC

```
void rtc_set_date(int date, int month, int year, int day)
{
    date_and_time.year = year;
    date_and_time.month = month;
    date_and_time.date = date;
    date_and_time.day = day;
}

void rtc_set_time(int hours, int minutes, int seconds, int hseconds)
{
    date_and_time.hours = hours;
    date_and_time.minutes = minutes;
    date_and_time.seconds = seconds;
    date_and_time.hseconds = hseconds;
}
```

3.15 Esempio funzioni RTC

3.2.8 sdram.c

La sdram viene inizializzata attraverso la funzione `SDRAMC_Init(unsigned long hsb_hz)` dove vengono impostati tutti i registri. Inoltre è presente una funzione di test che ritorna il valore 0 se tutti i collegamenti e tutti i registri sono stati configurati correttamente.

Grazie alla inizializzazione della sdrum è possibile utilizzarla come una semplice struttura di buffer. Questa struttura viene dichiarata nel file system.h.

```
typedef struct __attribute__((__packed__)) //alloco la sdrum
{
    char usb_buffer[MAX_USB_BUFFER];
    char ps2_buffer[MAX_USB_BUFFER];

    float adc_gate_buffer[MAX_ADC_BUFFER];
    float adc_drain_buffer[MAX_ADC_BUFFER];
    float adc_source_buffer[MAX_ADC_BUFFER];
    float adc_bulk_buffer[MAX_ADC_BUFFER];

    float adc_gate_avg_buffer[MAX_AVG_BUFFER];
    float adc_drain_avg_buffer[MAX_AVG_BUFFER];
    float adc_source_avg_buffer[MAX_AVG_BUFFER];
    float adc_bulk_avg_buffer[MAX_AVG_BUFFER];

    short adc_val[4][MAX_ADC_BUFFER];

    int adc_gate_range[MAX_ADC_BUFFER];
    int adc_drain_range[MAX_ADC_BUFFER];
    int adc_source_range[MAX_ADC_BUFFER];
    int adc_bulk_range[MAX_ADC_BUFFER];

    float dac_fire_buffer[MAX_USB_BUFFER];
} SDRAM_ALLOC;
```

3.16 Dichiarazione arrays SDRAM

Attraverso dei puntatori si può accedere direttamente alle locazioni di memoria della sdrum e dei buffer dichiarati in essa. In questo modo è possibile salvare una maggiore quantità di dati senza doverli memorizzare nella memoria interna del microcontrollore.

3.3 IMPLEMENAZIONE MISURE

In questo paragrafo vengono illustrate le tre funzioni che eseguono le diverse misurazioni. Vengono dichiarate attraverso delle *struct* dove all'interno sono presenti tutti i parametri impostabili. Le misure vengono eseguite solo dopo che l'utente ha impostato tutti i dati e i parametri con i rispettivi comandi che vengono esposti in seguito.

La prima misura implementata è la misura di Sampling.

I parametri impostabili sono la smu che polarizza il dispositivo, il range di misura della tensione di partenza (base) e quella del gradino (meas), le tensioni, il tempo per il quale queste tensioni vengono applicate, e il nmero di campioni.

```
typedef struct __attribute__((__packed__))
{
    int smu;
    int range_base;
    int range_meas;
    float v_base;
    float v_meas;
    float t_meas;
    float t_base;
    int n_sample_meas;
    int n_sample_base;
    float hold_time;
} SAMPLING;
```

3.17 struttura Sampling

Come prima operazione si deve aver impostato la smu alla quale viene imposto il gradino di tensione, la tensione di base e quella del gradino, il tempo per il quale viene tenuta la tensione rispettiva, il numero di campioni da acquisire e i range di misura.

Quando viene lanciata la misura, viene abilitata l'alimentazione analogica e viene inizializzato a 0 il buffer della SDRAM dove vengono salvati i campioni acquisiti dall'ADC. Successivamente viene fatto imporre dal DAC la prima tensione e mantenuta, senza essere campionata, per un tempo pari all'*hold time*, un altro parametro impostato dall'utente. Passato questo tempo di attesa, viene impostato e fatto partire il timer che abilita l'interrupt di acquisizione dell'adc e inizia quindi il campionamento con un tempo di campionamento pari a t_base/n_sample_base .

Finita questa fase, il dac impone la seconda tensione, *v_meas*, e come per la tensione di base, inizia l'acquisizione con un diverso tempo di campionamento.

Infine si leggono i buffer della sdram con i campioni acquisiti e il range corrispondenti per poter ricavare i valori effettivi della misura, dopodichè si disabilita l'alimentazione analogica.

```

//SAMPLING
void run_meas_sampl()
{
    usb_printf("run sampling meas..\n");
    enable_pwr_en();
    volatile int port_dac = 0;
    volatile int adc_in = 0;
    n_sample_counter = 0;

    if (measure_sampling.smu == GATE_SMU) { port_dac=GATE_DAC; adc_in=GATE_ADC; }
    if (measure_sampling.smu == DRAIN_SMU) { port_dac=DRAIN_DAC; adc_in=DRAIN_ADC; }
    if (measure_sampling.smu == SOURCE_SMU) { port_dac=SOURCE_DAC; adc_in=SOURCE_ADC; }
    if (measure_sampling.smu == BULK_SMU) { port_dac=BULK_DAC; adc_in=BULK_ADC; }

    volatile int a;
    for (a=0; a < MAX_ADC_BUFFER; a++)
    {
        adc_gate_buffer_p[a] = 0;
        adc_drain_buffer_p[a] = 0;
        adc_source_buffer_p[a] = 0;
        adc_bulk_buffer_p[a] = 0;
    }

    dac_write(port_dac, measure_sampling.v_base);
    dac_load();

    dac_write(port_dac, measure_sampling.v_meas);

    wait_us(1000000*measure_sampling.hold_time);

    set_range(BULK_SMU, 2);
    select_range(measure_sampling.smu, measure_sampling.range_base);

    adc_set_timer(measure_sampling.t_base/measure_sampling.n_sample_base);
    start_timer_convst();
    while (n_sample_counter < (measure_sampling.n_sample_base))
    {
    }
    stop_timer_convst();

    for(a=0; a < measure_sampling.n_sample_base; a++)
    {
        adc_val[GATE_ADC][a][0] = range_now_g;
        adc_val[DRAIN_ADC][a][0] = range_now_d;
        adc_val[SOURCE_ADC][a][0] = range_now_s;
        adc_val[BULK_ADC][a][0] = range_now_b;
    }

    set_range(GATE_SMU, measure_sampling.range_meas);

    adc_set_timer(measure_sampling.t_meas / measure_sampling.n_sample_meas);
    dac_load();
    start_timer_convst();
    while (n_sample_counter < measure_sampling.n_sample_meas + measure_sampling.n_sample_base)
    {
    }
    stop_timer_convst();

    for(a = measure_sampling.n_sample_base; a < measure_sampling.n_sample_base + measure_sampling.n_sample_meas; a++)
    {
        adc_gate_range_p[a] = range_now_g;
        adc_drain_range_p[a] = range_now_d;
        adc_source_range_p[a] = range_now_s;
        adc_bulk_range_p[a] = range_now_b;
    }

    adc_read2(GATE_ADC, measure_sampling.n_sample_base + measure_sampling.n_sample_meas, 0);
    adc_read2(DRAIN_ADC, measure_sampling.n_sample_base + measure_sampling.n_sample_meas, 0);
    adc_read2(SOURCE_ADC, measure_sampling.n_sample_base + measure_sampling.n_sample_meas, 0);
    adc_read2(BULK_ADC, measure_sampling.n_sample_base + measure_sampling.n_sample_meas, 0);

    disable_pwr_en();
}

```

3.18 Implementazione Sampling

Per la misura di ciclovoltammetria, la tensione viene imposta su un terminale del dispositivo quindi il setup è identico.

Come già spiegato in precedenza si vuole imporre una tensione triangolare a gradini e campionare su ogni gradino di tensione imposto.

I parametri che si possono impostare sono le tensioni, di partenza, di arrivo, la tensione massima e quella minima, il ramp rate, ovvero la velocità di variazione dell'onda triangolare [V/s] e la direzione dell'onda, ovvero se dalla tensione di partenza si passa al valore massimo e poi minimo o viceversa. Inoltre si imposta il numero di gradini di tensione che si hanno nella prima rampa, ovvero quella per andare dalla tensione di partenza a quella massima (se la direzione impostata è LTH *low to high*) e il numero di campioni per gradino.

```
typedef struct __attribute__((__packed__))  
{  
    int smu;  
    float ramp_rate;  
    int n_step_ramp0;  
    int range;  
    float v_start;  
    float v_stop;  
    float v_min;  
    float v_max;  
    int direction;  
    int n_sample_step;  
    float hold_time;  
} CV;
```

3.19 Struttura Ciclovoltammetria

Quando viene richiamata la funzione di misura di ciclovoltammetria come prima cosa viene chiamata una funzione che calcola i vari parametri delle rampe. Viene calcolato il delta di tensione di ogni rampa, il delta di tensione di ogni gradino, il tempo di durata di ogni gradino, il numero di gradini per ogni rampa e il numero totale di campioni.

```

void impostazioni_rampa_cv()
{
    oled_printf(&display,15,FALSE,"impostaz rampa\n");
    if ((measure_cv.v_start > measure_cv.v_min)&&(measure_cv.v_stop > measure_cv.v_min)&&
        (measure_cv.v_start < measure_cv.v_max)&&(measure_cv.v_stop < measure_cv.v_max))
    {
        if (measure_cv.direction == LTH)    // Low To High
        {
            d[0] = measure_cv.v_max - measure_cv.v_start;
            d[1] = measure_cv.v_max - measure_cv.v_min;
            d[2] = measure_cv.v_stop - measure_cv.v_min;
        }

        if (measure_cv.direction == HTL)    // High To Low
        {
            d[0] = measure_cv.v_start - measure_cv.v_min;
            d[1] = measure_cv.v_max - measure_cv.v_min;
            d[2] = measure_cv.v_max - measure_cv.v_stop;
        }

        n_step_ramp[0] = measure_cv.n_step_ramp0;
        volt_increm = d[0] / n_step_ramp[0];
        t_step = volt_increm / measure_cv.ramp_rate;
        n_step_ramp[1] = d[1] / volt_increm;
        n_step_ramp[2] = d[2] / volt_increm;
        n_sample_tot = measure_cv.n_sample_step * (n_step_ramp[0]+n_step_ramp[1]+n_step_ramp[2]);
    }
}

```

3.20 Calcolo parametri Ciclovoltammetria

Successivamente viene abilitata l'alimentazione analogica, vengono inizializzati i buffer e viene impostata la tensione di partenza. A questo punto inizia il campionamento per un numero di campioni impostato e alla fine di questo prima step inizia la prima rampa formata da il numero di gradini preimpostati. Quando viene raggiunta la tensione massima (o minima) si passa alla seconda rampa, e così via.

Alla fine vengono letti tutti i campioni e viene disabilitata la tensione analogica.

Nell'immagine successiva viene riportato il codice fino alla fine della prima rampa, per le successive due l'algoritmo di ripete, cambia solo il numero di step e la tensione imposta.

```

void run_meas_cv()
{
    impostazioni_rampa_cv();
    enable_pwr_en();

    float v_imposta[1000];
    int a;
    int b;
    int i;
    for (a=0; a<MAX_ADC_BUFFER; a++)
    {
        adc_gate_buffer_p[a] = 0;
        adc_drain_buffer_p[a] = 0;
        adc_source_buffer_p[a] = 0;
        adc_bulk_buffer_p[a] = 0;

        adc_gate_range_p[a] = 0;
        adc_drain_range_p[a] = 0;
        adc_source_range_p[a] = 0;
        adc_bulk_range_p[a] = 0;

        for (b=0; b<4; b++)
        {
            sdram_p->adc_val[b][a] = 0;
        }
    }

    volatile int port_dac = 0;
    volatile int adc_in = 0;
    switch (measure_cv.smu)
    {
        case GATE_SMU:
            port_dac=GATE_DAC;
            adc_in=GATE_ADC;
            break;
        case DRAIN_SMU:
            port_dac=DRAIN_DAC;
            adc_in=DRAIN_ADC;
            break;
        case SOURCE_SMU:
            port_dac=SOURCE_DAC;
            adc_in=SOURCE_ADC;
            break;
        case BULK_SMU:
            port_dac=BULK_DAC;
            adc_in=BULK_ADC;
            break;
        default:
            oled_printf(&display, 15, FALSE, "\n\rErrore smu\n\r");
            break;
    }

    n_sample_counter = 0;

    v_step_i = measure_cv.v_start;
    dac_write(port_dac, v_step_i);
    dac_load();
}

```

```

wait_us(1000000*measure_cv.hold_time);

select_range(measure_cv.smu, measure_cv.range);
select_range(BULK_SMU,2);

adc_set_timer(t_step/measure_cv.n_sample_step);
//adc_set_timer(MIN_T_ACQ);
start_timer_convst();
while (n_sample_counter < measure_cv.n_sample_step)
{ }
stop_timer_convst();

for(a = 0; a < measure_cv.n_sample_step; a++)
{
    adc_gate_range_p[a] = range_now_g;
    adc_drain_range_p[a] = range_now_d;
    adc_source_range_p[a] = range_now_s;
    adc_bulk_range_p[a] = range_now_b;
}

v_imposta[0] = v_step_i;

//RAMP 0-----

switch (measure_cv.direction)
{
    case LTH:
        v_step_i = v_step_i + volt_increm;
        dac_write(port_dac, v_step_i);
        break;
    case HTL:
        v_step_i = v_step_i - volt_increm;
        dac_write(port_dac, v_step_i);
        break;
    default:
        usb_printf("\n\rImpostare direzione rampa\n\r");
        break;
}

for (n_step_counter = 0; n_step_counter < n_step_ramp[0]; n_step_counter++)
{
    i = n_sample_counter;
    dac_load();

    v_imposta[n_step_counter+1] = v_step_i;

    select_range(measure_cv.smu, measure_cv.range);

    adc_set_timer(t_step/measure_cv.n_sample_step);
    start_timer_convst();
    while (n_sample_counter < (i + measure_cv.n_sample_step))
    { }
    stop_timer_convst();

    for(a = measure_cv.n_sample_step*(n_step_counter+1);
        a < ((measure_cv.n_sample_step * (n_step_counter+1))+ measure_cv.n_sample_step); a++)
    {
        adc_gate_range_p[a] = range_now_g;
        adc_drain_range_p[a] = range_now_d;
        adc_source_range_p[a] = range_now_s;
        adc_bulk_range_p[a] = range_now_b;
    }

    if ((n_step_counter+1) < n_step_ramp[0])
    {
        switch (measure_cv.direction)
        {
            case LTH:
                v_step_i = v_step_i + volt_increm;
                dac_write(port_dac, v_step_i);
                break;
            case HTL:
                v_step_i = v_step_i - volt_increm;
                dac_write(port_dac, v_step_i);
                break;
        }
    }
}
}

```

3.21 Implementazione Ciclovoltammetria

Per la misura di Firing&Sensing vengono utilizzate tutte e quattro le smu. La smu di GATE impone una gradino di tensione al terminale di gate del sensore. La smu di DRAIN impone una polarizzazione continua tra drain e source del DUT, viene quindi collegata al drain del dispositivo mentre il source viene connesso al BNC di massa dello strumento. La smu di SOURCE viene utilizzata come firing, è quella che impone l'impulso neuronale simulato, mentre la smu di BULK è quella che legge la risposta in tensione del transistor.

```
typedef struct __attribute__((__packed__))
{
    //DRAIN
    float v_drain;
    int drain_range;
    //GATE
    float v_base;
    float t_base;
    float amp_step;
    float time_step;
    int gate_range;
    float hold_time;
    //BULK
    int neuron_sens_range;
    //SOURCE
    int neuron_imp_range;
    //n sample
    int n_sample_step;
    int n_sample_base;
} FIRING;
```

3.22 Struttura Firing&Sensing

Quando viene eseguita la funzione di misura, dopo aver abilitato l'alimentazione analogica vengono inizializzati i buffer in sdram e disabilitate le smu. Poi vengono impostate le tensioni a cui viene polarizzato il DUT. Per un tempo preimpostato si campiona la tensione alla quale si porta il dispositivo. Questa misura deve avvenire per un tempo abbastanza lungo in modo che la tensione arrivi al suo valore di equilibrio e viene fatta una media degli ultimi 5 campioni per misurare appunto questa tensione di equilibrio, $v_{equilibrio}$. I successivi impulsi verranno fatti partire da questo valore. Finita questa fase la tensione al gate viene impostata ad un diverso valore e si continua a campionare e acquisire sia le varie correnti che la tensione sul dispositivo. Quando questa tensione letta supera un valore di soglia pre impostato, la smu di SOURCE impone al DUT una tensione ad impulso che simula il potenziale di azione del neurone. Lo strumento campiona i segnali di risposta con un tempo di campionamento impostato dall'utente più fitto rispetto alla fase in cui si misura il potenziale di equilibrio.

I valori di tensione dell'impulso sono salvati in un buffer, derivano da una simulazione eseguita con un apposito software e corrispondono ad un "campionamento" del segnale simulato con un tempo di 100us tra un

campione e l'altro. È quindi preferibile scegliere questo tempo di campionamento anche durante la misura per poter avere una simulazione dell'impulso più precisa.

A fine misura come nelle precedenti, vengono letti i buffer con i campioni acquisiti, viene disabilitata l'alimentazione analogica e scollegate le smu.

```
//FIRING
void run_meas_fire()
{
    usb_printf("run sampling meas..\n");

    enable_pwr_en();
    n_sample_counter = 0;
    int n_sample_imp = 163;

    volatile int a;
    volatile int b;
    for (a=0; a<MAX_ADC_BUFFER; a++)
    {
        adc_gate_buffer_p[a] = 0;
        adc_drain_buffer_p[a] = 0;
        adc_source_buffer_p[a] = 0;
        adc_bulk_buffer_p[a] = 0;

        adc_gate_range_p[a] = 0;
        adc_drain_range_p[a] = 0;
        adc_source_range_p[a] = 0;
        adc_bulk_range_p[a] = 0;

        for (b=0; b<4; b++)
        {
            sdram_p->adc_val[b][a] = 0;
        }
    }

    deselect_smu(GATE_SMU);
    deselect_smu(DRAIN_SMU);
    deselect_smu(SOURCE_SMU);
    deselect_smu(BULK_SMU);

    float v_drain = measure_fire.v_drain;
    dac_write(DRAIN_DAC, v_drain);
    dac_write(SOURCE_DAC, 0);
    dac_write(GATE_DAC, measure_fire.v_base);
    dac_load();

    set_range(GATE_SMU, measure_fire.gate_range);
    set_range(DRAIN_SMU, measure_fire.drain_range);
    set_range(SOURCE_SMU, measure_fire.neuron_imp_range);
    set_range(BULK_SMU, measure_fire.neuron_sens_range);

    int p;
    for(p=0; p < measure_fire.n_sample_base + measure_fire.n_sample_step; p++)
    {
        adc_gate_range_p[p] = range_now_g;
        adc_drain_range_p[p] = range_now_d;
        adc_source_range_p[p] = range_now_s;
        adc_bulk_range_p[p] = range_now_b;
    }
}
```

```

dac_write(GATE_DAC, measure_fire.amp_step);

select_smu(GATE_SMU);
select_smu(DRAIN_SMU);
select_smu(BULK_SMU);

wait_us(1000000*measure_fire.hold_time);

int imp_i = 0;
int buffer_indice[10] = { 0,0,0,0,0,0,0,0,0,0 };

adc_set_timer(measure_fire.t_base/measure_fire.n_sample_base);

wait_ms(1000);
start_timer_convst();
while (n_sample_counter < measure_fire.n_sample_base)
{ }
stop_timer_convst();

adc_read2(BULK_ADC, 5, n_sample_counter-6);
float avg = 0;
int y;
for (y=6; y>=1; y--)
{
    avg = avg + adc_bulk_buffer_p[n_sample_counter-y];
}
v_equilibrio = avg/5;
oled_printf(&display,15,FALSE,"\n\rVequilibrio=%g\n\r", v_equilibrio);

adc_set_timer((measure_fire.time_step/measure_fire.n_sample_step)-0.000075);
dac_load();

int j = 0;
float soglia = 0;
int bo = n_sample_counter-1;
while (n_sample_counter < (measure_fire.n_sample_step + measure_fire.n_sample_base))
{
    start_timer_convst();
    while ((n_sample_counter <= bo) && (n_sample_counter < measure_fire.n_sample_step + measure_fire.n_sample_base))
    { }
    stop_timer_convst();
    bo = n_sample_counter;

    if (j == n_sample_imp)
    {
        j = 0;
        soglia = 0;
        deselect_smu(SOURCE_SMU);
    }

    if (soglia > THRESHOLD_FIRE)
    {
        adc_read2(BULK_ADC, 1, n_sample_counter-1);
        soglia = adc_bulk_buffer_p[n_sample_counter-1];
    }

    if (j >= 1)
    {
        dac_write(SOURCE_DAC, (-imp_buff[j] + imp_buff[299] + v_equilibrio));
        dac_load();
        j++;
    }

    if ((soglia <= THRESHOLD_FIRE) && j == 0)
    {
        dac_write(SOURCE_DAC, (-imp_buff[j] + imp_buff[299] + v_equilibrio));
        dac_load();
        select_smu(SOURCE_SMU);
        j++;
        buffer_indice[imp_i] = n_sample_counter;
        imp_i++;
    }
}

```

```

disable_pwr_en();
deselect_smu(GATE_SMU);
deselect_smu(DRAIN_SMU);
deselect_smu(SOURCE_SMU);
deselect_smu(BULK_SMU);

adc_read2(GATE_ADC, measure_fire.n_sample_base + measure_fire.n_sample_step, 0);
adc_read2(DRAIN_ADC, measure_fire.n_sample_base + measure_fire.n_sample_step, 0);
adc_read2(SOURCE_ADC, measure_fire.n_sample_base + measure_fire.n_sample_step, 0);
adc_read2(BULK_ADC, measure_fire.n_sample_base + measure_fire.n_sample_step, 0);
}

```

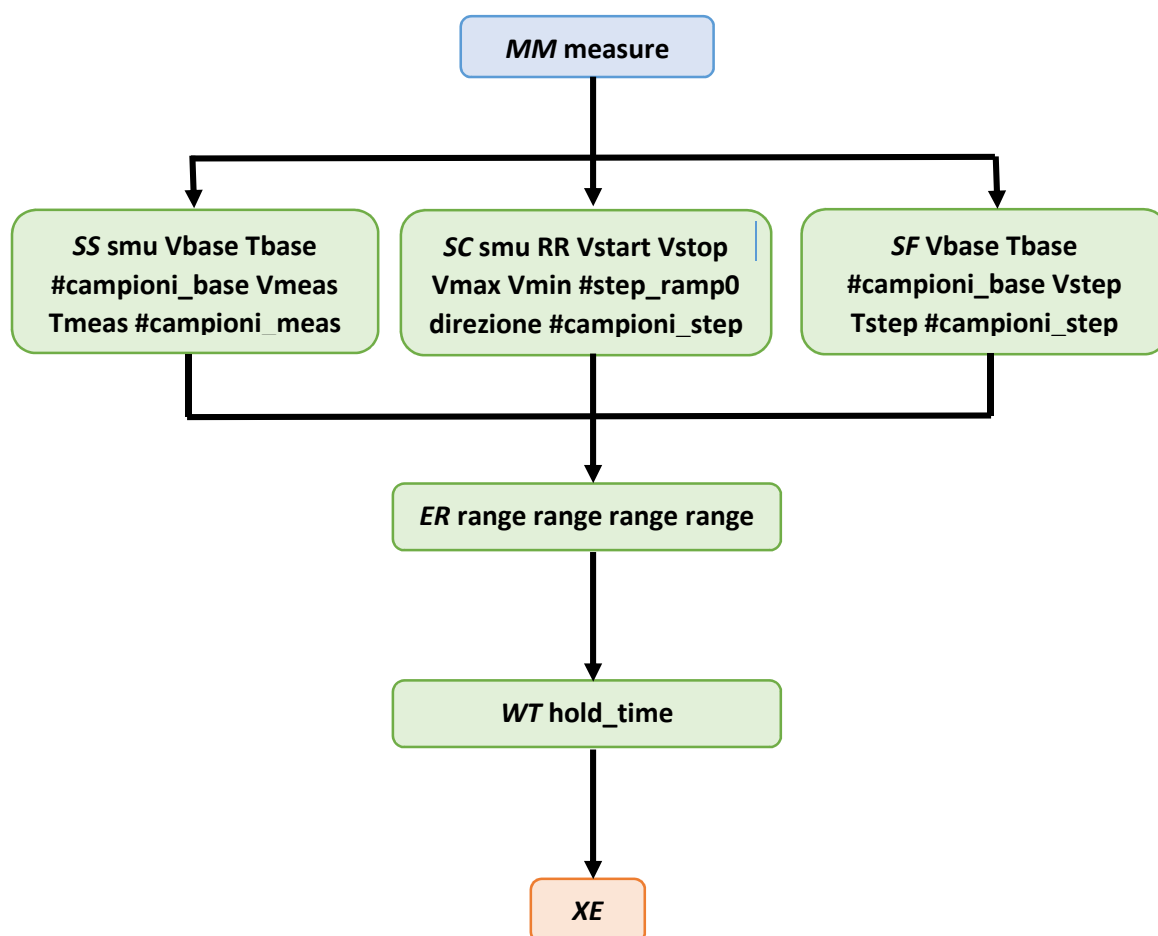
3.23 Implementazione Firing&Sensing

3.4 COMANDI

Come descritto in precedenza, si possono inviare comandi da terminale o da tastiera. Vengono qui di seguito elencati i comandi possibili con i parametri da settare per ogni comando.

funzione	comando	descrizione
reset_system()	*RST	resetta il sistema
id()	*ID	ritorna l'ID della scheda
in_zero_volt(arg_p)	IN smu	impone 0V alla smu selezionata
force_voltage(arg_p)	DV smu range voltage	impone tensione e range all smu
disable_channel(arg_p)	CL smu	disabilita la smu
enable_channel(arg_p)	CN smu	abilita la smu
set_time(arg_p)	HM ora minuti	imposta ora e minuti
set_date(arg_p)	DMY giorno mese anno	imposta giorno mese anno
self_calibration_dac(arg_p)	CAD smu	misura la tensione di offset del dac
set_hold_time(arg_p)	WT hold_time	imposta hold time
loadv(arg_p)	LOADV a b c d e f...[V]	carica in sdram i valori di tensione
measure_mode(arg_p)	MM measure	seleziona il tipo di misura
set_sampling(arg_p)	SS smu Vbase Tbase #campioni_base Vmeas Tmeas #campioni_meas	imposta i parametri per lil sampling
set_ramp_cv(arg_p)	SC smu RR Vstart Vstop Vmax Vmin #step_ramp0 direzione #campioni_step	imposta i parametri per la ciclovoltammetria
set_firing(arg_p)	SF Vbase Tbase #campioni_base Vstep Tstep #campioni_step	imposta i parametri per il firing & sensing
enable_range(arg_p)	ER range range range range	abilita i range
execute_meas()	XE	esegui la misura

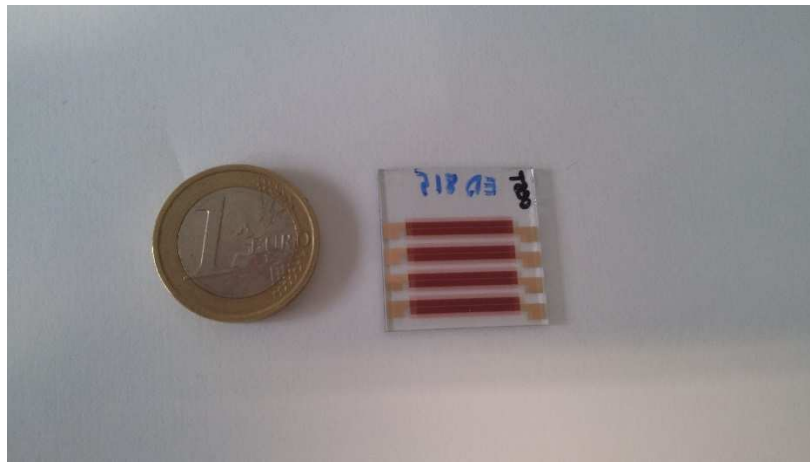
Per poter effettuare una misura si deve seguire questo schema di comandi e inserire i parametri necessari.



CAPITOLO 4

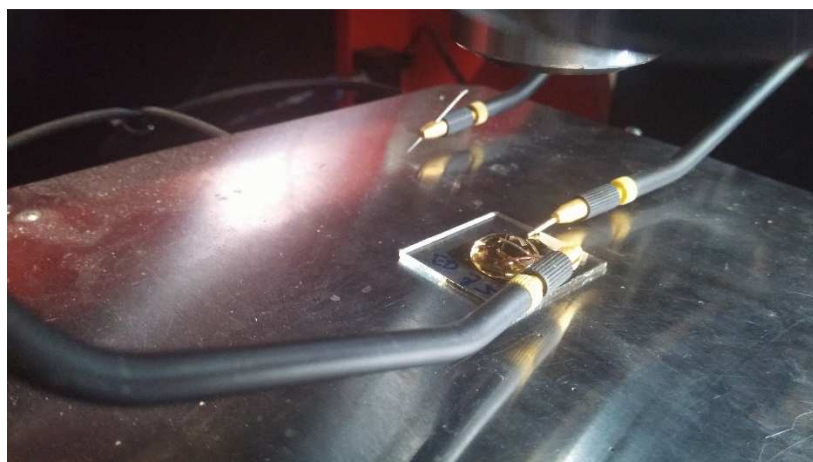
MISURAZIONI E RISULTATI

In quest'ultimo capitolo vengono riportati i risultati delle tre diverse misurazioni effettuate su dei dispositivi presenti in laboratorio.



4.1 Esempio di dispositivo

Nell'immagine successiva vengono illustrati gli elettrodi in platino collegati al dispositivo. Un elettrodo ha la funzione di imporre la tensione che l'utente imposta e che il DAC della scheda impone alla SMU sul terminale del dispositivo, mentre l'altro è collegato alla SMU di lettura e ne legge la risposta in tensione o corrente.



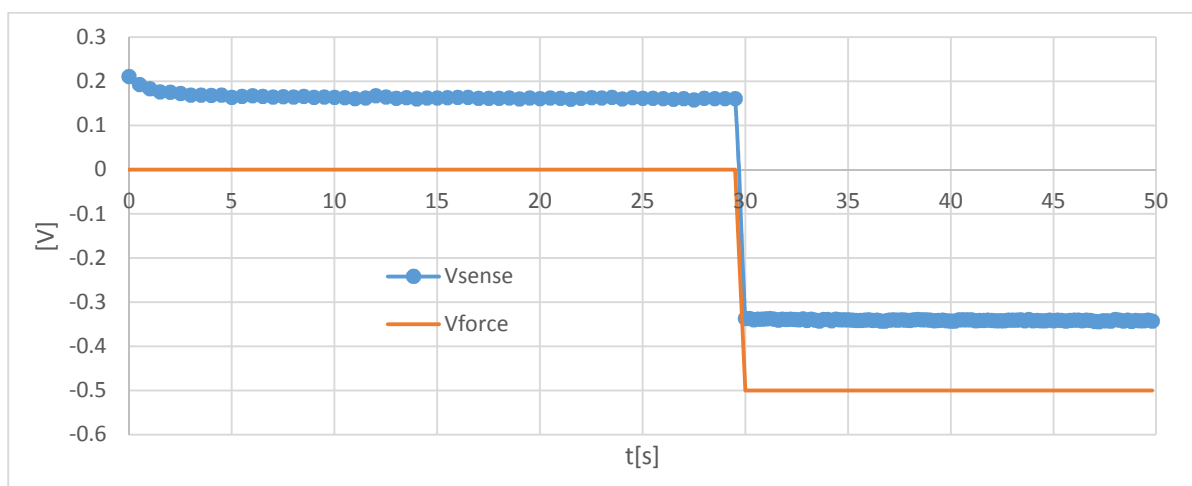
4.2 Elettrodi di polarizzazione e sensing

4.1 MISURA SAMPLING

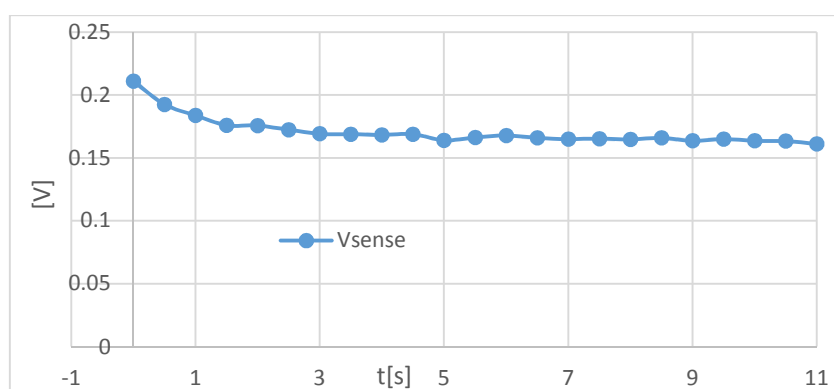
Nella prima misura di sampling effettuata, si impone al terminale di gate una tensione pari a 0V per 30s e una tensione di -0.5V per 20s. Si acquisiscono 60 campioni per la prima fase e 100 campioni nella seconda.

Quello che si nota, acquisendo la risposta in tensione del dispositivo, è un sfasamento verso l'alto della curva di tensione letta rispetto alla tensione imposta di circa 150mV. Questo è dovuto alla differenza delle funzioni lavoro tra l'oro (Au) del contatto e il platino (Pt) dell'elettrodo.

Un'altra caratteristica che si nota è l'andamento esponenziale decrescente della tensione del dispositivo che si assesta ad un valore di equilibrio dopo un certo tempo.

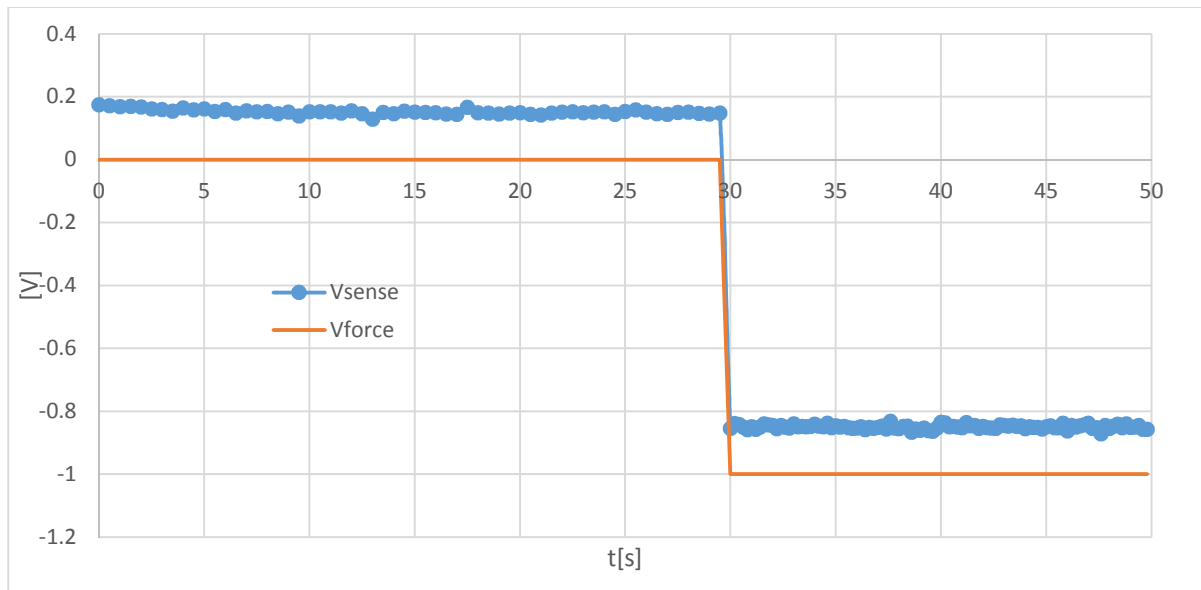


3.3 Misura 1 Sampling

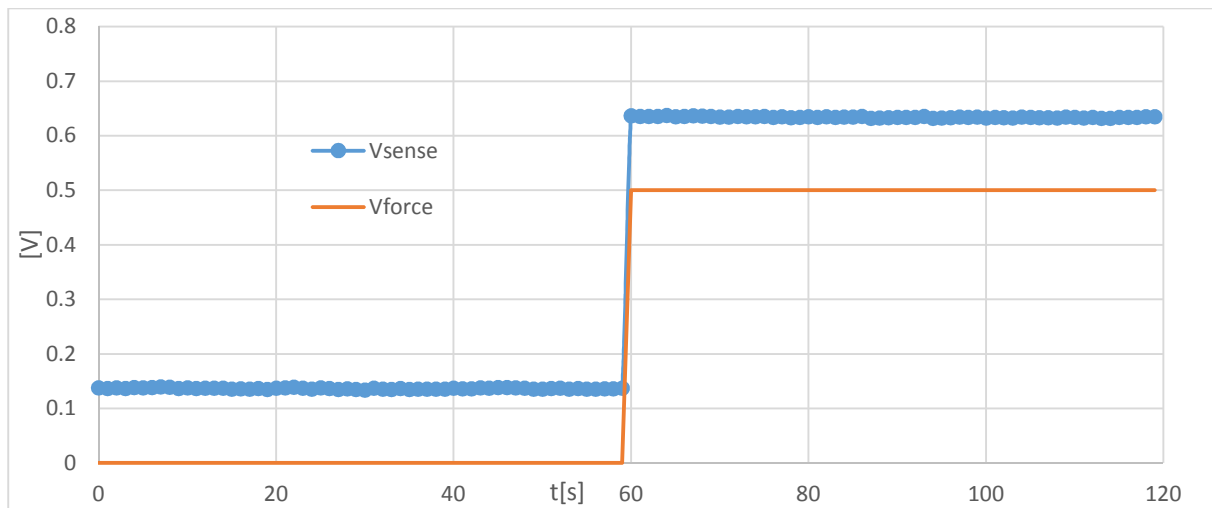


3.4 Fase esponenziale

La seconda misura invece di -0.5V si è imposto -1V, mentre nella terza +0.5V. Le risposte sono riportate qui di seguito e si nota sempre lo stesso sfasamento verso l'alto della tensione.



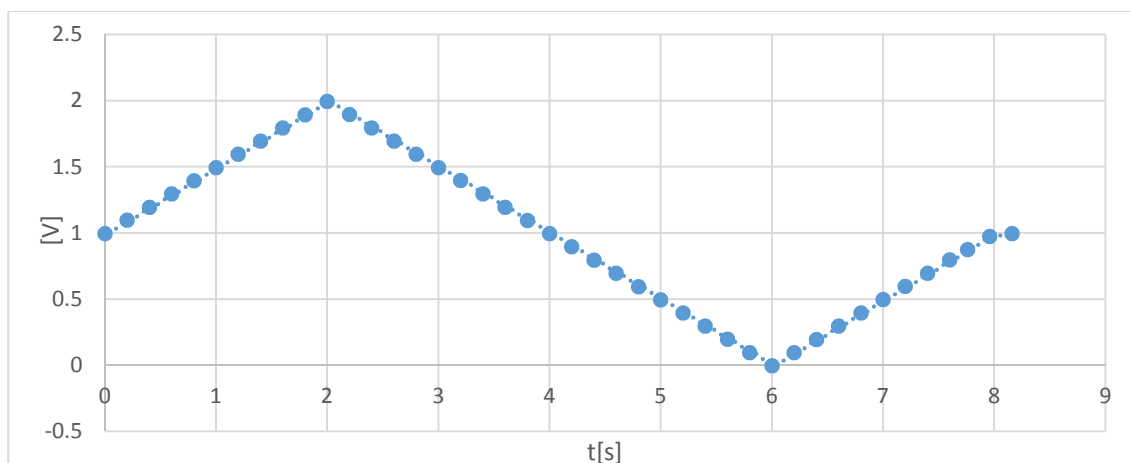
3.5 Misura 2 Sampling



3.6 Misura 3 Sampling

4.2 MISURA CICLOVOLTAMMETRIA

Nella prima misura di ciclovoltammetria riportata di seguito è stata imposta una tensione triangolare a gradini al contatto di gate del dispositivo. Viene riportata la tensione media imposta di ogni gradino in funzione del tempo, il ramp rate si nota essere di 0.5V/s, tensione di partenza 1V, tensione di stop 1V, tensione massima 2V e minima 0V. sono presenti 10gradini perla prima rampa e la direzione è LTH (Low to High).



3.7 Media gradini tensione imposta

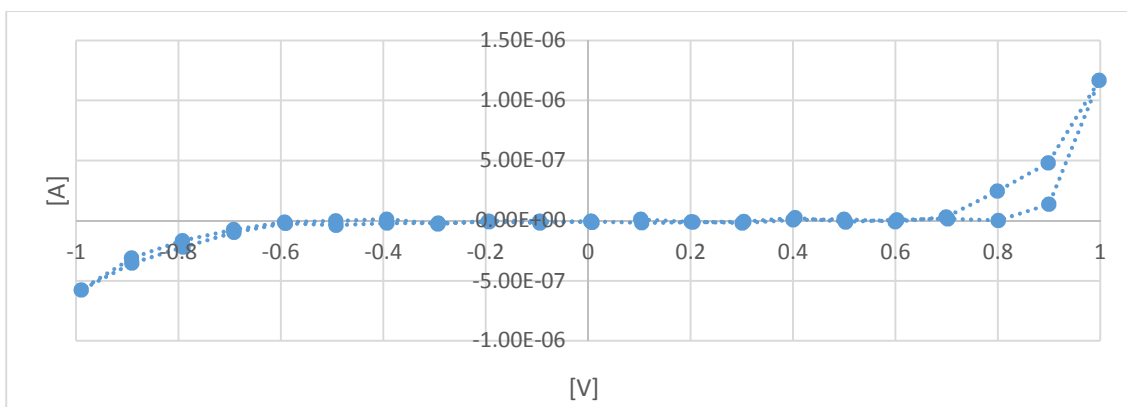
Nelle successive misure è stata imposta al terminale di gate del dispositivo una forma d'onda triangolare con le rispettive tensioni riportate nella seguente tabella. Viene inoltre riportato il tipo di dispositivo sul quale viene fatta la misura, il ramp rate, il numero di gradini della prima rampa, la direzione e il numero di campioni per gradino.

MIS	DISPOS	SMU	RR[V/s]	START[V]	STOP[V]	MAX[V]	MIN[V]	#STEP0	DIR	#SAMPLE
1	Au P13	1	0.1	0	0	1	-1	10	0	5
2	Au P13	1	0.1	0	0	1	-1	20	0	5
3	Au P13	1	1	0	0	1	-1	20	0	4
4	Au P13	1	0.01	0	0	1	-1	20	0	4
5	Au P13	1	10	0	0	1	-1	20	0	4
6	Au P13	1	10	0	0	2	-2	40	0	3
7	Au P13	1	10	0	0	2	-2	40	1	3
8	Au	1	10	0	0	2	-2	40	1	3
9	Au	1	10	0	0	2	-2	40	0	3

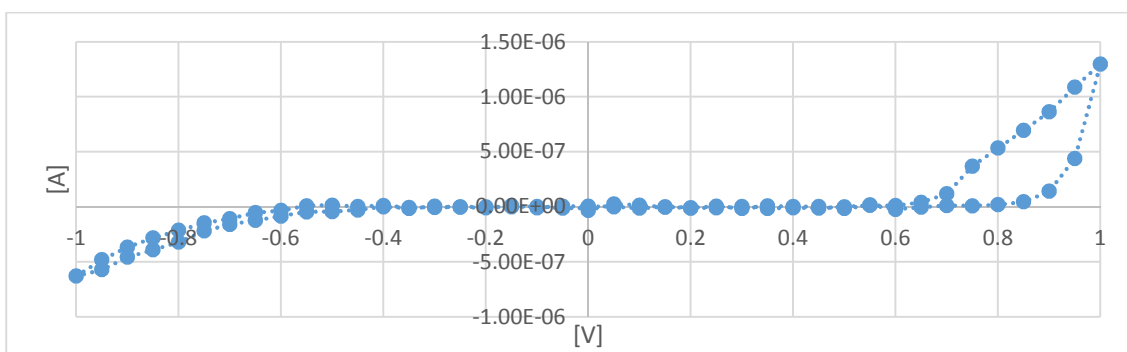
I grafici seguenti riportano la corrente di gate letta in funzione della tensione imposta.

Con il primo dispositivo, dove è presente il P13, si nota che per bassi ramp rate (0.1 e 0.01) la corrente ha andamento esponenziale per tensioni che si avvicinano alla tensione massima e minima applicata, mentre per il resto della curva è circa pari a 0V.

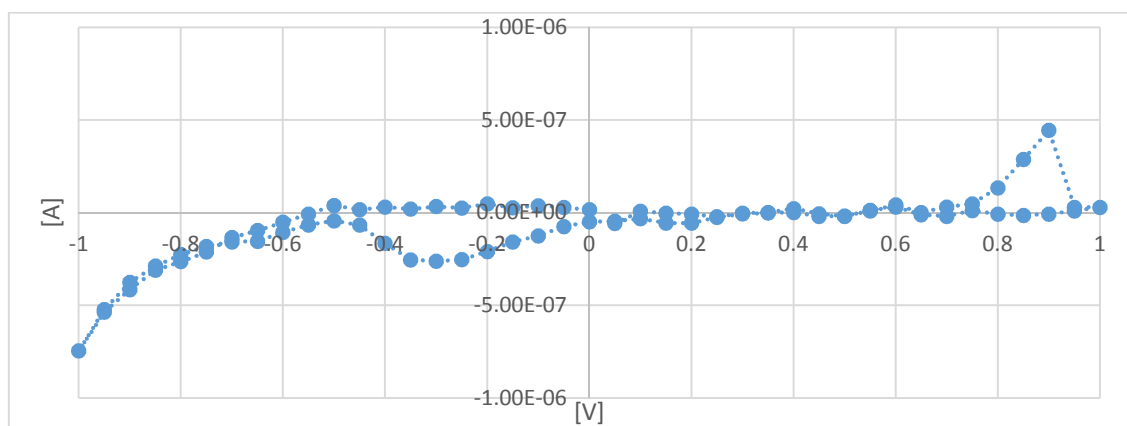
Aumentando il ramp rate (1 e 10) invece, si nota anche una “gobba” nella curva della corrente in un intorno di -0.5V applicato. Questa non è presente nei dispositivi dove non è presente il P13 e in questi inoltre, si può notare anche un’isteresi più ampia.



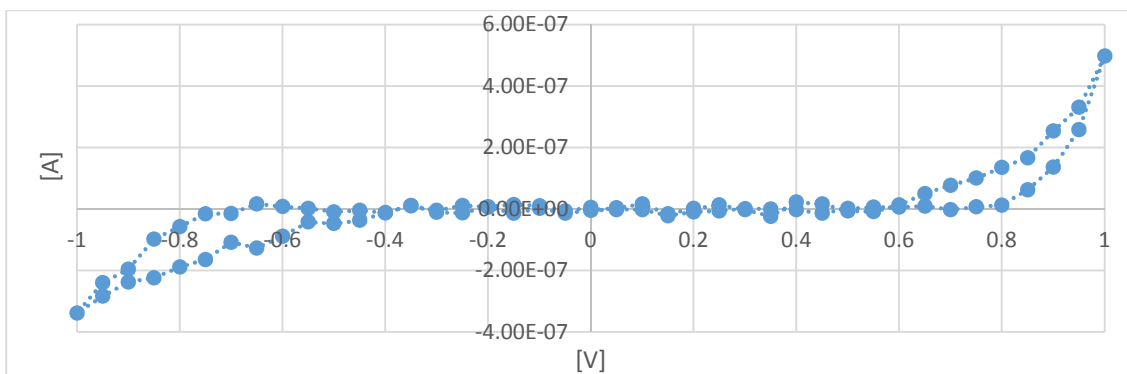
3.8 Misura 1 CV



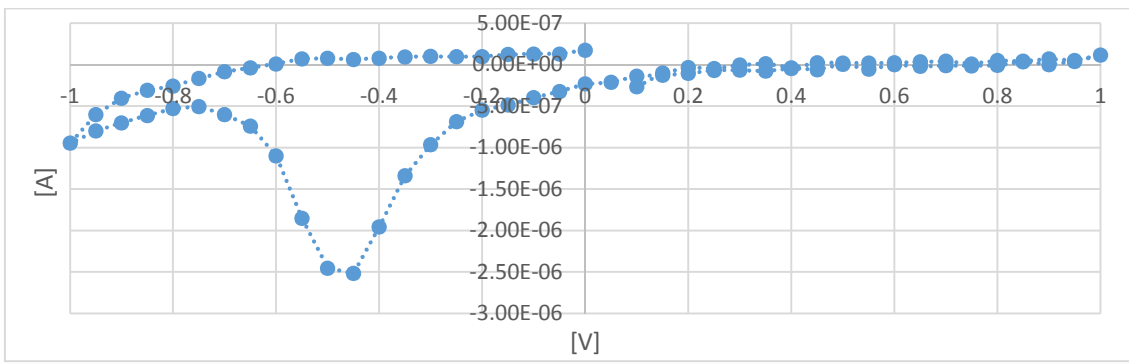
3.9 Misura 2 CV



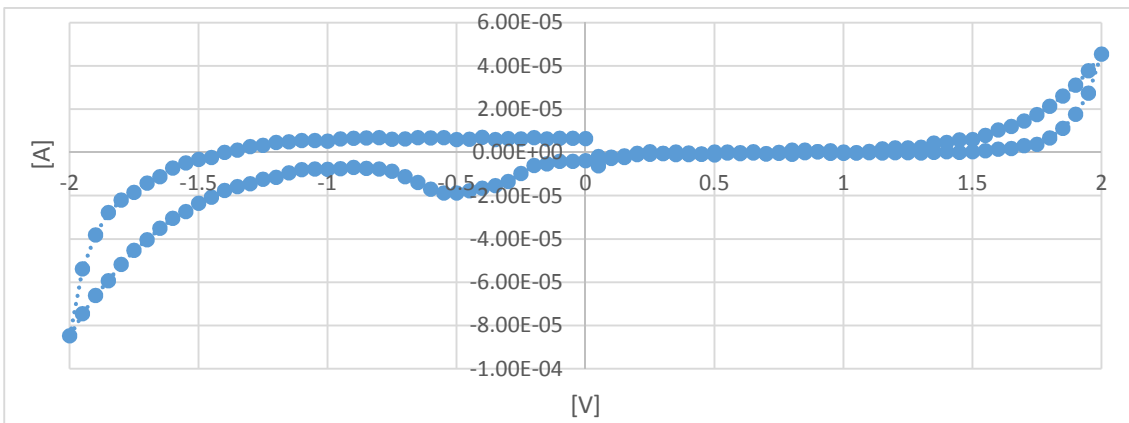
3.10 Misura 3 CV



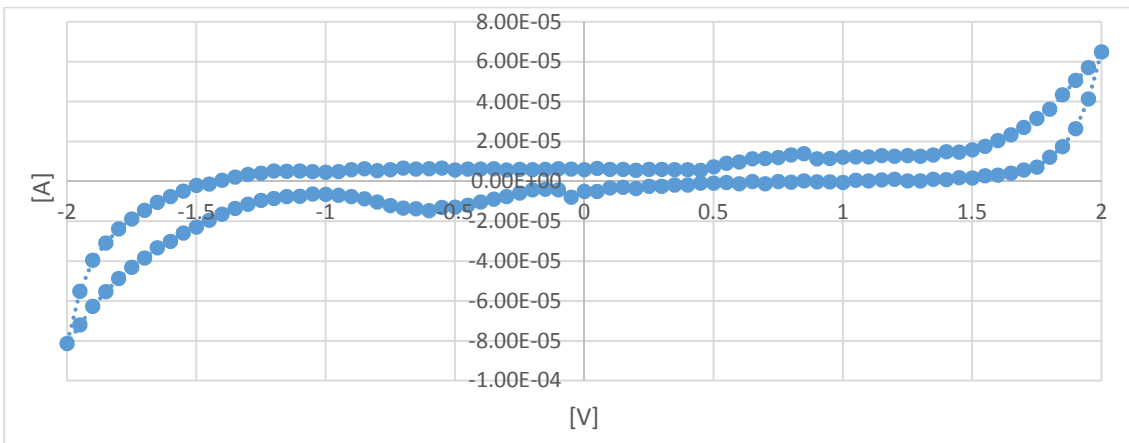
3.11 Misura 4 CV



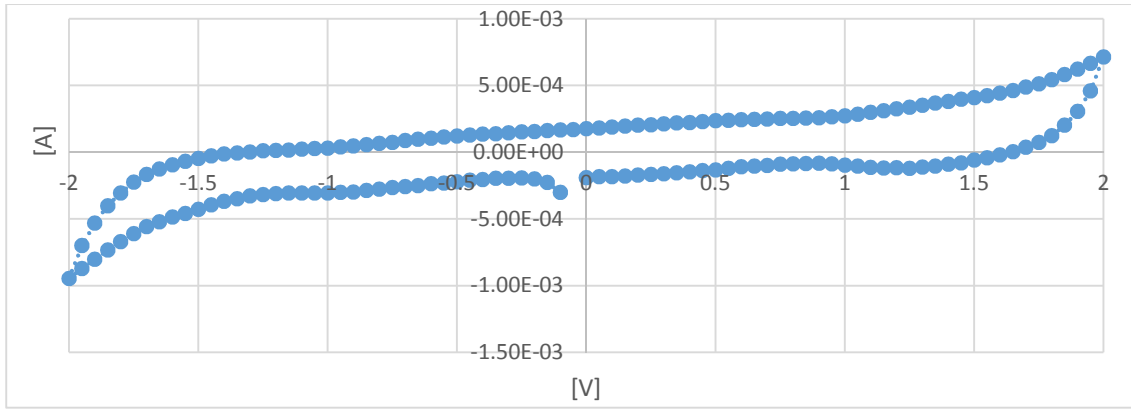
3.12 Misura 5 CV



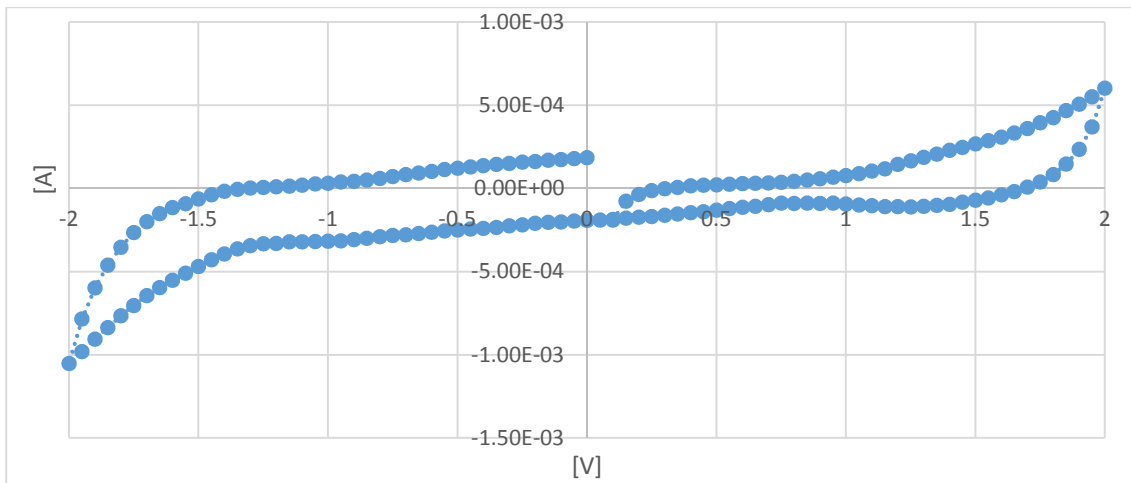
3.13 Misura 6 CV



3.14 Misura 7 CV



3.15 Misura 8 CV



3.16 Misura 9 CV

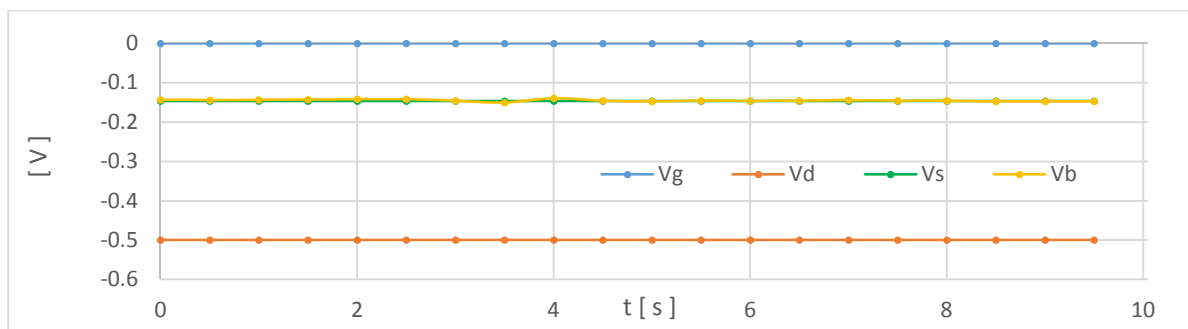
4.3 MISURA FIRING & SENSING

In questa misura il terminale di source del dispositivo viene collegato alla massa della scheda, al drain viene imposta una tensione continua di polarizzazione di $-0.5V$, mentre al gate viene imposta una tensione pari a $0V$ per 10 secondi. In questi dieci secondi si acquisiscono 20 campioni, degli ultimi 5 campioni della misura di tensione si calcola la media, e il risultato viene attribuito alla tensione di equilibrio a cui si è portato il dispositivo.

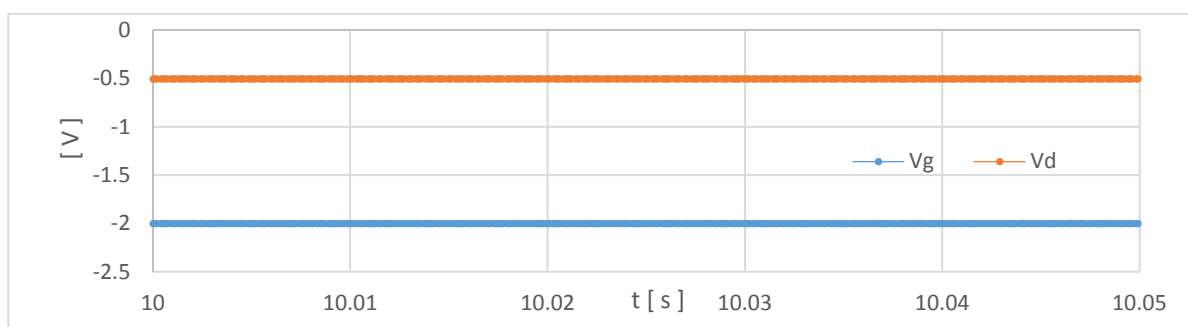
Passati questi 10 secondi la tensione di gate passa a $-2V$, la tensione letta quindi ad un certo punto diventa maggiore in modula della soglia preimpostata di $-0.1V$ e quindi partono gli impulsi di simulazione neuronale attraverso la SMU3 che partono dalla tensione di equilibrio precedentemente calcolata, per tutto il tempo in cui la tensione di gate rimane a $-2V$. Questo tempo è stato impostato a $50ms$ e vengono acquisiti 400 campioni.

Vengono qui riportati i grafici delle tensioni imposte e acquisite e delle correnti acquisite durante questa misura.

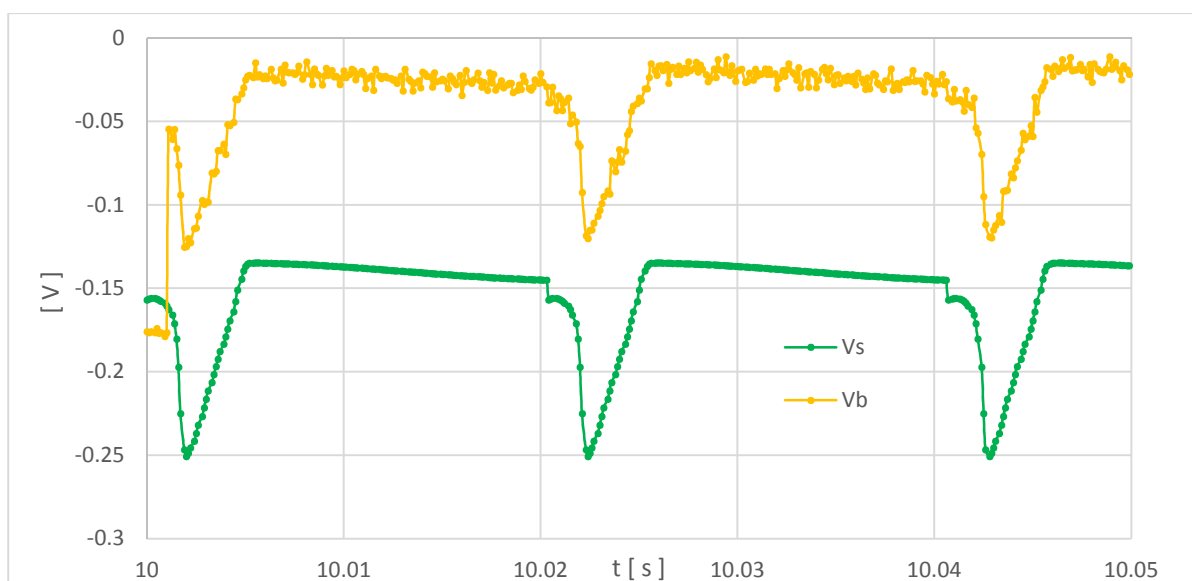
La V_g e la V_d sono rispettivamente la tensione di gate e di drain imposte, la V_s è la tensione che simula gli impulsi neuronali, mentre la V_b è la risposta in tensione acquisita. I_g , I_d , I_s e I_b sono le rispettive correnti acquisite.



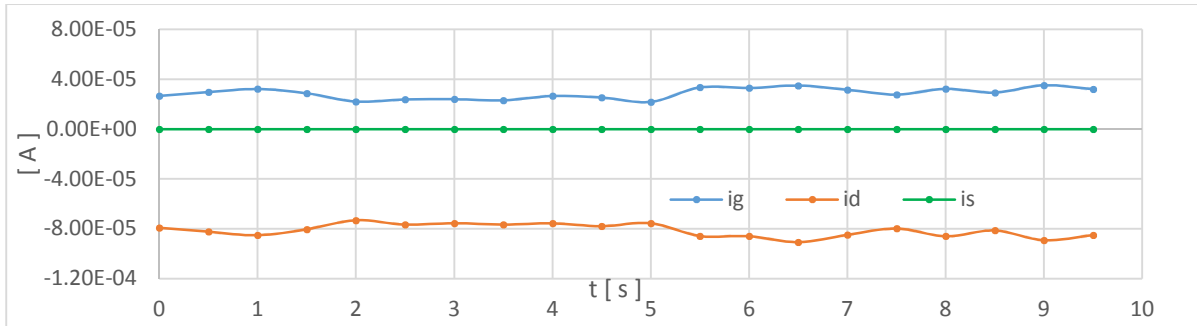
3.17 Tensioni prima fase F&S



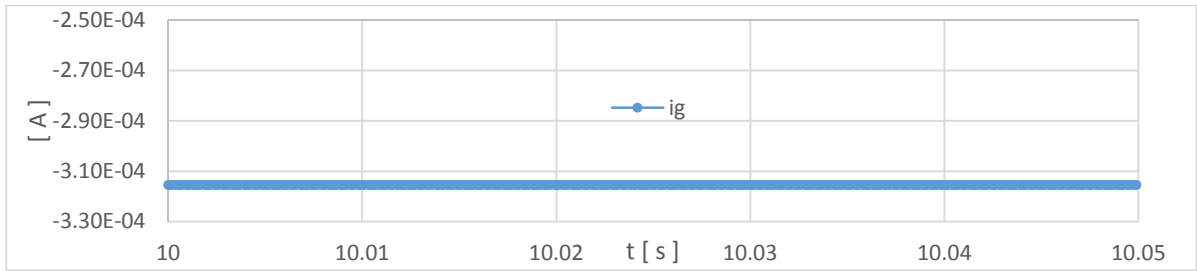
3.18 Tensioni polarizzazione seconda fase F&S



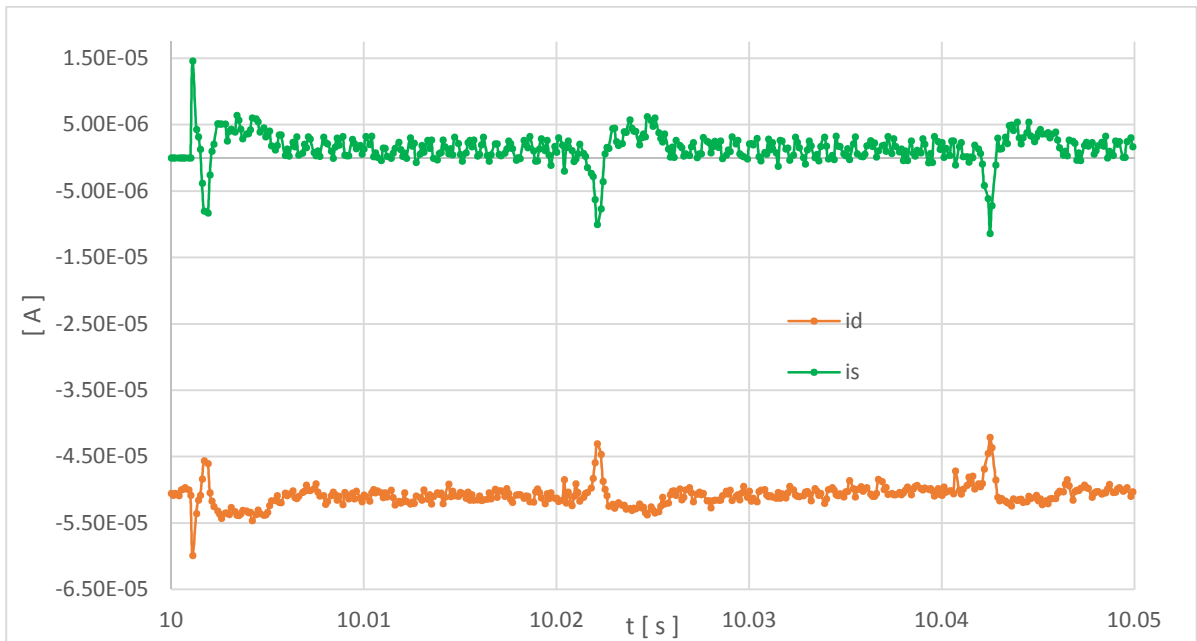
3.19 Impulsi imposti e letti



3.20 Correnti prima fase F&S



3.21 Corrente di gate prima fase F&S



3.22 Correnti drain e source seconda fase F&S

CONCLUSIONI

In questo lavoro di tesi è stato affrontato l'intero percorso progettuale e realizzativo di una scheda elettronica necessaria per lo studio di un particolare dispositivo.

Si è partiti da una problematica, cioè la volontà di effettuare delle misure su dei biosensori organici per poterli studiare e caratterizzare, e da questo si è passati alla stesura delle specifiche dello strumento da progettare.

Seguendo le specifiche, si è progettato dapprima lo schematico del circuito e successivamente il layout della scheda.

Una fase molto importante è stata anche la programmazione del firmware che permette di utilizzare il circuito nella maniera corretta.

Come ultima fase c'è stata la validazione del lavoro tramite delle misurazioni effettuate su dei dispositivi.

Questo progetto è uno strumento utile per lo studio degli OTFT utilizzati come biosensori. Il lavoro di modellizzazione e caratterizzazione è ancora in fase di studio, ma quando si avrà un'idea più chiara sul funzionamento di questi dispositivi sarà più semplice capire il miglior modo per realizzarli e come utilizzarli in un campo così delicato come la riabilitazione e la diagnosi di problemi neurologici.

BIBLIOGRAFIA

- “Dispense di Elettronica Organica”, Andrea Cester.
- “OrganicThin-Film Transistor (OTFT)-Based Sensor”, D.Elkington, N.Cooling, W.Belcher, P.C.Dastor, X.Zhou.
- “Chemical and biological sensors based on organic thin-film trnasistors”, J.T.Mabech, G.G.Malliaras.
- “A review of Organic and Inorganic Biomaterials for Neural Interfaces”, P,Fattahi, G.Yang, G.Kim, M.R.Abidian.
- “Organic Thin-Film Transtors for Chemical and Biological Sensing”, P.Lin, F.Yan.
- “Manuale di Elettronica e Telecomunicazioni”, G.Biondo, E.Sacchi – Hoepli.
- “Linguaggio C”, B.W.Kernighal, D.MRitchie.
- “Electrochemistry, Spectroscopy and Electrogenated Chemiluminescence of Perylene, Terrylene, and Quaterrylene Diimides in Aprotic Solution”, S.K.Lee, Y.Zu, A.Herrmann, Y.Geerts, K.Mllen, A.J.Bard.
- “Measuring Surface-related Currents using Digital Staircase Voltammetry”, Application note rev. 4.0 4/28/2015