ALMA MATER STUDIORUM – UNIVERSITA DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA IN INGEGNERIA
ELETTRONICA PER L'ENERGIA E L'INFORMAZIONE

TITOLO DELL'ELABORATO

PROGETTO DI UN NODO SENSORE A NANOCORRENTI BASATO SU MICROCONTROLLORE

Tesi in:
ELETTRONICA DEI SISTEMI DIGITALI

Relatore:
Prof. Aldo Romani

Correlatore:
Dott. Davide Fabbri

Presentato da:
Davide Fazi

Sessione III
Anno accademico 2015-16

Sommario

Introduzione

Le prestazioni e la densità di integrazione dei circuiti integrati hanno avuto una incredibile crescita negli ultimi decenni, seguendo la cosiddetta "Legge di Moore" secondo la quale il numero di transistor che possono essere integrati in uno stesso chip sarebbe cresciuto esponenzialmente nel tempo. [1]
Questo incredibile sviluppo ha portato ad una enorme crescita delle applicazioni dei circuiti elettronici, fino ad arrivare a pensare di poter rendere "smart" e connesso ad internet ogni oggetto di uso comune, grazie alla cosiddetta "Internet of Things" e cioè l'interconnessione di dispositivi fisici alla rete internet mediante l'integrazione di sistemi elettronici.
La IoT consente agli oggetti di essere monitorati e controllati in maniera remota utilizzando però le già esistenti infrastrutture di rete. [2]
In questo panorama trovano dunque sempre maggior spazio i sensori, indispensabili al fine di monitorare i parametri d'interesse negli oggetti da controllare.

Un altro aspetto molto importante è quello di ridurre il consumo dei dispositivi. Gli oggetti da connettere ad internet nella maggior parte dei casi saranno da alimentare a batteria; questo comporta di dover scegliere componenti sempre più ottimizzati al fine di ridurre il consumo di potenza, così da poter consentire una maggior durata della batteria a parità di capacità.
Il consumo di potenza è ora infatti uno dei problemi più importanti dell'industria del silicio e per questo, il mondo della ricerca in ambito elettronico, è sempre più orientato verso applicazioni in grado di contenere l'assorbimento di corrente.

Lo scopo di questo elaborato è quello di sviluppare un nodo sensore con annesso un datalogger mediante l'utilizzo di un microcontrollore. L'obiettivo principale è di ridurne il consumo al minimo, con correnti di standby dell'ordine delle decine di nA, così da poter alimentare tale circuito per un periodo di diversi anni con la batteria più piccola possibile.

Il circuito sarà non solo progettato ma sarà anche realizzato su PCB, al fine di poterne collaudare il funzionamento reale e di poterlo confrontare con quello previsto.
Un dispositivo di questo tipo, oltre ad essere un utile caso di studio per mostrare i valori minimi di consumo verso i quali sia possibile spingersi, potrebbe trovare applicazione in situazioni in cui si renda necessario monitorare determinati parametri in condizioni tali da rendere difficoltosa la sostituzione della batteria e

quindi anche il monitoraggio real-time che richiederebbe necessariamente un maggior consumo.

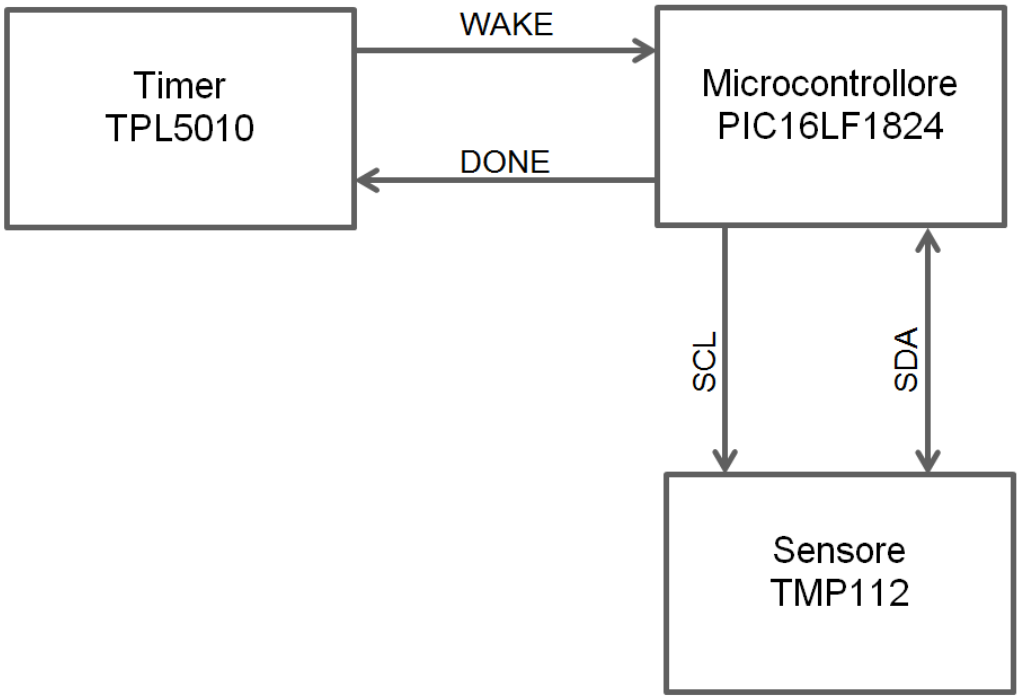

Figura 1 Schema a blocchi del progetto

Basilarmente il progetto si compone di un microcontrollore che, oltre a gestire la lettura di un sensore, si occupa di implementare la funzione di datalogger. Un altro componente fondamentale è un timer, avente lo scopo di gestire il periodo di attività del µC.

1. Il Consumo di Potenza

In questo primo capitolo si farà una panoramica delle cause dei consumi di potenza nei dispositivi elettronici e delle strategie utilizzate per ridurre tali consumi.

1.1. Il consumo di potenza nei dispositivi CMOS

I moderni circuiti integrati sfruttano per lo più la tecnologia CMOS. Qualora sia lecito trascurare i fenomeni di leakge, in tale tecnologia la corrente è assorbita principalmente nella fase di commutazione, mentre è molto ridotta nella fase statica in cui i livelli logici sono mantenuti dalle capacità parassite.
La potenza assorbita può essere divisa in tre contributi: contributo statico (o di leakage), contributo dinamico e contributo di cortocircuito. [3]

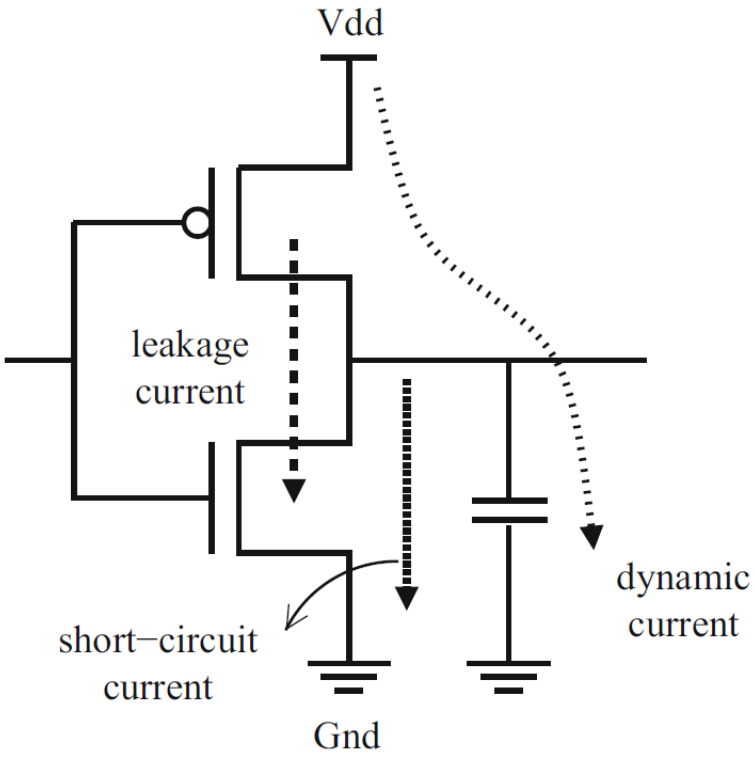

Figura 2. Le componenti della dissipazione di potenza di un transistor **[3]**

1.1.1. Potenza Dinamica

Con riferimento alla Figura 2, ogni volta che la capacità è caricata attraverso il transistor pMOS, la sua tensione sale da 0 a V_{DD} e parte della potenza in gioco è dissipata sul transistor. Analogamente nella transizione alto-basso la capacità è scaricata verso massa e l'energia immagazzinata precedentemente è dissipata sull'nMOS. Ne consegue che tale consumo di potenza si verifica solo durante le fasi di carica e risulta valere:

$$P_{din} = C_L \, {V_{DD}}^2 \, f \quad [1]$$

Per ridurlo è quindi possibile operare in tre modi:

- Effettuare uno scaling del componente riducendo così la capacità di carico (C_L)
- Ridurre la tensione di alimentazione
- Diminuire la frequenza del circuito

Negli ultimi anni si è lavorato molto sui primi due aspetti, arrivando a produrre transistor con processo produttivo inferiore ai 20nm e tensioni di alimentazione inferiori agli 1.8V. Tuttavia, la riduzione spinta delle geometria accentua in maniera rilevante i fenomeni di leakage. E' da notare che la tensione di alimentazione ha un effetto quadratico sulla potenza e quindi si ottengono ottimi risultati riducendola.
Non sempre invece risulta possibile intervenire sulla frequenza poiché diminuendola purtroppo si riducono le prestazioni. Tale parametro ha infatti subito un trend crescente proprio al fine di massimizzarle. Tuttavia, in applicazioni specifiche, sarà generalmente accettabile una riduzione della frequenza di funzionamento.

1.1.2. Potenza statica

[3] Il consumo stazionario di un circuito è espresso dalla relazione:

$$P_{stat} = I_{stat} \, V_{DD}$$

Dove I_{stat} è la corrente assorbita in assenza di commutazioni.
Idealmente tale consumo dovrebbe essere nullo, dato che i due transistor non sono mai accesi simultaneamente. In realtà esistono vari contributi che fanno sì

che tale potenza non sia nulla, creando dei collegamenti conduttivi tra alimentazione e massa:

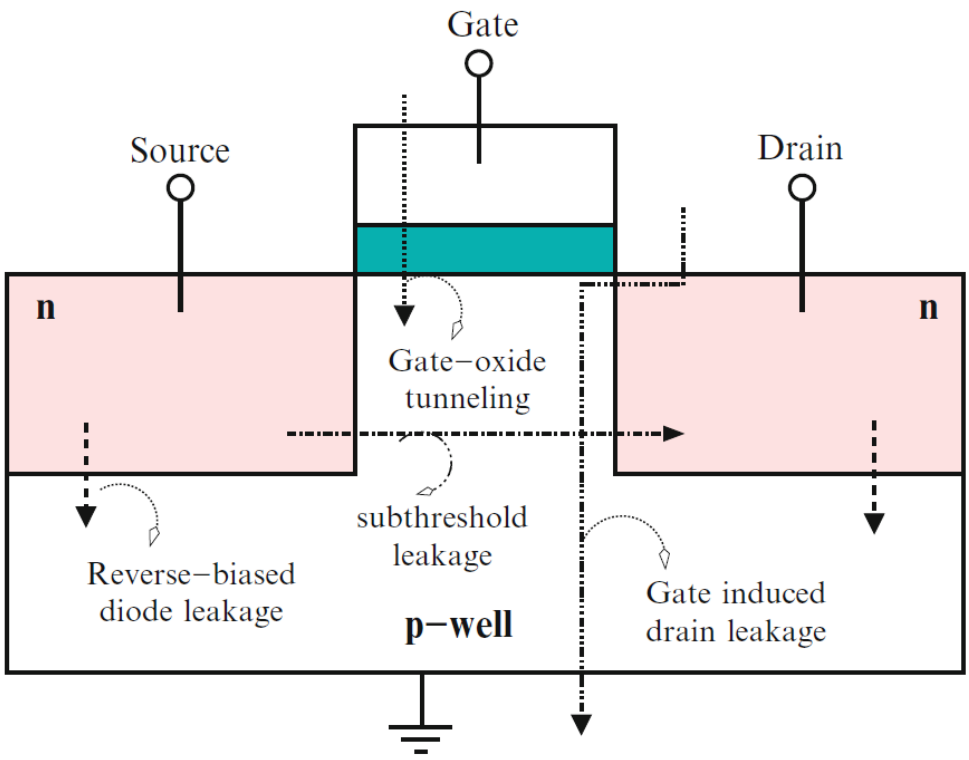

Figura 3. Componenti della Potenza Statica **[3]**

Reverse-biased diode leakage (corrente di perdita della diffusione di drain/source)

E' la potenza persa a causa del flusso di corrente presente tra le giunzioni drain-bulk e/o source-bulk che si comportano come diodi collegati inversamente.
Il fenomeno è dunque dovuto alla diffusione dei portatori minoritari, alla corrente di deriva e alla generazione di coppie elettrone-lacuna nella regione di svuotamento a seguito di tale polarizzazione.

Subthreshold Leakage (Corrente di sottosoglia)

La corrente di sottosoglia è dovuta al fatto che in realtà il transistor non si spegne completamente ma conduce anche per V_{GS} inferiori alla soglia.
Più la tensione di soglia è in modulo prossima a 0V, più influente risulta essere questo effetto. Per questa ragione la tensione di soglia è normalmente abbastanza elevata (0.5-0.6V).
Ridurre la tensione di alimentazione (per andare incontro all'evoluzione tecnologica) mantenendo però invariata la soglia, si traduce in una riduzione

delle prestazioni. La scelta della tensione di soglia rappresenta dunque un importante compromesso tra prestazioni e consumo.
Nei moderni processi tecnologici, a seguito della continua riduzione della tensione di alimentazione, si sta sempre più spingendo la tensione di soglia verso valori più bassi rendendo la corrente di sottosoglia la componente più rilevante del consumo statico.

Gate Oxide Tunneling
La corrente di effetto tunnel scorre attraverso l'isolamento dell'ossido, andando dunque dal gate al bulk. Questa corrente è il risultato del passaggio degli elettroni nella banda di conduzione dell'ossido in seguito all'applicazione di un grande campo elettrico ai capi di quest'ultimo. Tale contributo aumenta al diminuire dello strato di ossido.

Gate Induced Drain Leakage (GIDL)
Questa perdita è causata dal grande effetto di campo presente nella giunzione di drain. In un transistor nMOS quando il gate è polarizzato per formare uno strato di accumulazione, il piccolo strato di silicio sulla superficie del drain diventa fortemente polarizzato di tipo P. Quando il gate è ad un potenziale negativo o nullo e il drain è connesso all'alimentazione, può esserci un grande aumento dell'effetto valanga e dell'effetto tunnel. I portatori minoritari sotto al gate sono così portati al substrato, dando luogo alla corrente GIDL.
Questo effetto è molto meno influente dei tre visti in precedenza e può spesso essere trascurato.

1.2. Il consumo medio di potenza

Il consumo di potenza di un sistema elettronico definisce l'energia assorbita dal circuito in relazione a ciascuna operazione e, quindi, quanto calore viene dissipato dal circuito stesso.
In un circuito alimentato a batteria è importante considerare il consumo medio di potenza, definito come segue:

$$P_{media} = \frac{1}{T}\int_0^T p(t)dt = \frac{V_{DD}}{T}\int_0^T i_{DD}(t)dt$$

Dove V_{DD} è la tensione di alimentazione, i_{DD} è la corrente assorbita dall'alimentazione e T è l'intervallo di tempo in cui è calcolata la media della potenza. [1]
Come suggerito dal nome, tale figura di merito consente di sapere il consumo medio del circuito, tenendo conto di tutte le potenze viste precedentemente in relazione al tempo in cui sono presenti e questo permette, conoscendo la capacità di una batteria, di determinarne la sua durata.

1.3. Andamento del consumo di potenza

Con l'aumento della velocità e della densità di integrazione, il consumo di potenza è aumentato enormemente negli ultimi anni facendo sì che questo aspetto sia diventato uno dei vincoli di progetto più importanti, aprendo nuove strade di ricerca.
Se per molto tempo la potenza dinamica è stata la componente principale della dissipazione di potenza in un circuito, ora la potenza statica inizia a rappresentare il problema maggiore.
Come già visto infatti se da un lato lo scaling e la riduzione dell'alimentazione hanno ridotto il consumo in commutazione, dall'altro il leakage è aumentato in seguito alla riduzione della dimensione dei componenti e all'abbassamento della tensione di soglia per i motivi già visti.

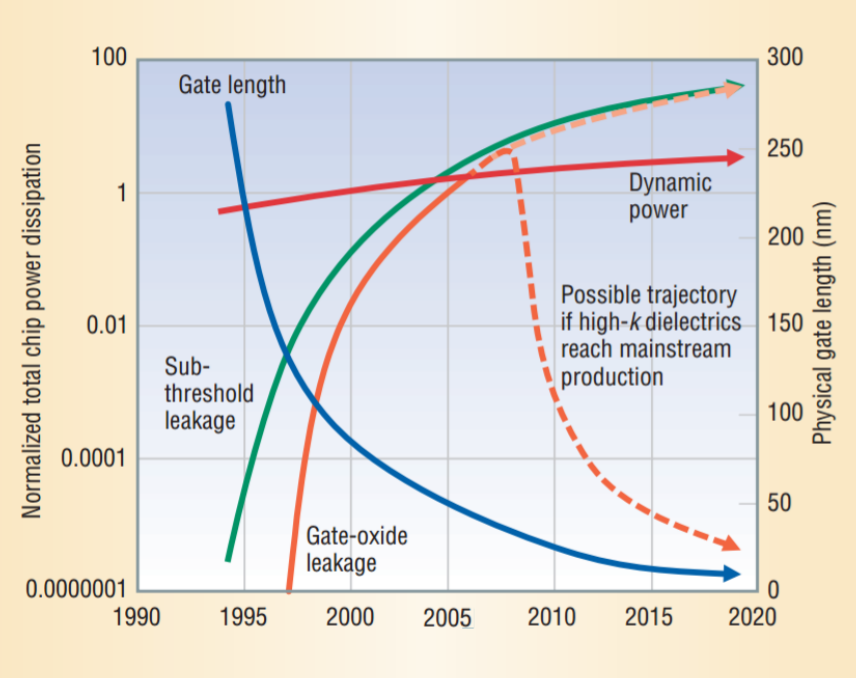

Figura 4. Andamento della dissipazione totale (statica e dinamica) dei chip **[4]**

1.4. Riduzione del consumo nei microcontrollori

Anche nell'ambito dei microcontrollori (uC) le case produttrici di componenti stanno cercando di ridurre i consumi. Oltre all'utilizzo degli accorgimenti progettuali visti nei paragrafi precedenti, per ridurre i consumi è possibile utilizzare delle modalità di risparmio energetico che spengono determinate periferiche mantenendo attive solo quelle indispensabili. L'operatività di una periferica sarà così garantita per il tempo minimo necessario al corretto funzionamento.
Normalmente i uC hanno varie modalità di standby, che mantengono attive più o meno funzionalità. Tipicamente questo dipende dal produttore e dalla tecnologia utilizzata.

2. Scelta dei componenti

In questo secondo capitolo è mostrato come sono stati scelti i componenti utili alla realizzazione del progetto. A questa fase è infatti stata data molta importanza poiché si è reso necessario porre l'attenzione non solo sulle funzionalità dei componenti, ma anche e soprattutto sul consumo di corrente di questi, sia in condizioni operative che di standby.
Non sono mostrati unicamente i componenti utilizzati nel progetto finale, ma anche altri dispositivi ritenuti degni di nota poiché sono stati oggetto di studio durante questa fase.
Una più approfondita descrizione dell'hardware impiegato, segue nei capitoli successivi.

2.1. Scelta dell'alimentazione

2.1.1. Batteria

Da specifiche il progetto prevede che l'alimentazione sia garantita da una batteria. Si è scelto di utilizzare una CR2032, una batteria con tecnologia Litio-diossido di manganese caratterizzata da una tensione nominale pari a 3V, una capacità tipica compresa tra i 210 e i 240 mAh (a seconda del produttore) e un coefficiente di scarica annuo dell'ordine dell'1%.
Tale dispositivo, oltre ad essere molto diffuso, è pensato per l'alimentazione di sistemi elettronici a basso consumo e presenta una capacità discretamente elevata per la sua categoria, in grado di garantire il periodo di vita desiderato per questa applicazione.

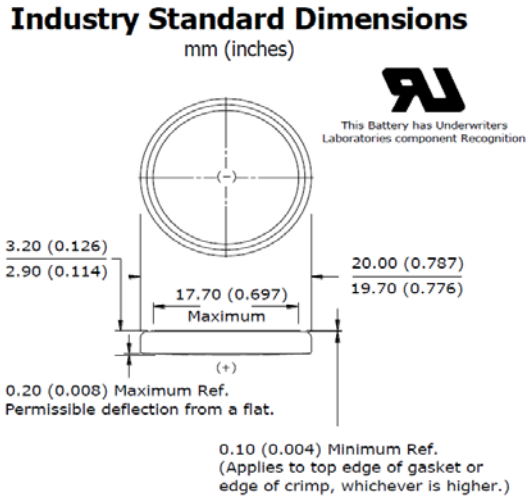

Figura 5 Dimensioni della batteria CR2032 **[5]**

2.1.2. Convertitore DC/DC

Nonostante la tensione fornita dalla batteria sia discretamente bassa, i dispositivi odierni sono in grado di lavorare con tensioni inferiori garantendo così, come già visto, assorbimenti di potenza minori.

L'utilizzo di un regolatore lineare risulta da scartare a priori dato che, a fronte di una riduzione della tensione in uscita, si ha una caduta sul regolatore stesso che corrisponde alla differenza di tensione tra ingresso e uscita.

Si è pensato inizialmente dunque di inserire un convertitore di tipo switching (ad esempio un Chopper-Buck). Questo è infatti in grado di garantire ottimi rendimenti di conversione, anche superiori al 90%.
Ai fini di questo progetto si è riscontrato però un grosso limite: la corrente di quiescenza, anche nei dispositivi più evoluti e pensati per le basse potenze, risulta essere dell'ordine delle centinaia di nA. Dovendo, come specifica di progetto, essere il consumo a riposo del nodo sensore al massimo dell'ordine delle decine di nA, l'impiego di un convertitore sarebbe risultato controproducente.

Prendendo ad esempio in considerazione l'integrato commerciale ADP5300, un convertitore integrato di tipo buck, la corrente di quiescenza tipica risulta essere di 180nA e quindi eccessivamente elevata come già spiegato.
Questo componente, come molti altri, dispone di una modalità di shutdown che ne ridurrebbe il consumo a soli 18nA. In questo progetto però alcuni dispositivi, come si vedrà in seguito, necessitano di essere sempre alimentati e dunque neppure questa modalità di funzionamento risulterebbe essere utile.

Ne consegue che la scelta migliore è quella di non interporre nessun componente tra la batteria ed il resto del circuito, selezionando componenti in grado di operare nel range di tensioni fornite dalla batteria durante il suo ciclo operativo.

2.2. Scelta del microcontrollore

Sono state confrontate tre diverse teconlogie, proposte da altrettante compagnie:

- Microchip PIC XLP (eXtreme Low Power)
- Texas Instruments MSP430 basati su FRAM (Ferroelectric RAM)
- AmbiqMicro Apollo basato su tecnologia SPOT (Subthreshold Power Optimized Technology)

Segue una tabella con un confronto tra tre dispositivi delle aziende concorrenti.

	Sleep [nA]	WDT [nA]	1MHz run [µA]	Dimensione architettura
Microchip PIC16LF18xx	20	300	39	8 bit
Texas Instruments MSP430FR59xx	20	250	100	16 bit
Ambiq Micro Apollo1	143	419	35	32 bit

Tabella 1 Confronto per i consumi dei dispositivi Microchip, Texas Instruments e Ambiq Micro

E' da sottolineare che il confronto è impari: Apollo vede implementata una tecnologia avanzata basata su ARM Cortex-M4F a 32 bit, MSP430FR59xx a 16 bit mentre PIC16LF18xx a soli 8 bit. Inoltre andranno tenuti in considerazione i cicli di clock per istruzione (CPI) di ogni architettura. Ciò nonostante, ai fini dell'applicazione, una tecnologia a 8 bit risulta essere più che sufficiente, dato che non è richiesta un'elevata potenza di calcolo, mentre l'attenzione dev'essere posta sul consumo di potenza, in particolare quello in standby.

Il dispositivo della Ambiq Micro è stato dunque scartato per il maggior consumo in modalità di sleep, mentre quello della Texas Instruments per il maggior consumo/Mhz durante l'utilizzo della CPU.

Si è scelto dunque di orientarsi sui microcontrollori della Microchip della linea XLP (eXtreme Low Power). Questi sono dei dispositivi progettati appositamente per essere utilizzati in applicazioni che necessitano di ridurre al minimo i consumi. Dispongono infatti di una funzione di deep-sleep che consente di ridurre drasticamente i consumi quando non si devono eseguire operazioni. In questa modalità è inoltre garantito il mantenimento dei dati presenti nella memoria RAM e questo, unito al fatto che oltre l'80% delle operazioni sono

eseguite in un solo ciclo di clock, consente di mantenere attivo il dispositivo per un tempo il più possibile contenuto.
Si può uscire dallo stato di sleep grazie ad interrupt forniti dalla maggior parte delle periferiche.

Il componente scelto è il PIC16LF1824 caratterizzato da un consumo minimo di 20nA (che diventano 30nA con alimentazione a 3V); è risultato infatti essere il miglior compromesso per minimizzare la corrente di sleep, prestando però attenzione anche a quella di lavoro.
Questo microcontrollore dovrà restare in standby, per poi essere svegliato da un watchdog timer (WDT) sulla base di una determinata temporizzazione. Dato che il timer interno comporta un assorbimento di corrente di 500nA si è scelto di utilizzarne uno esterno più efficiente, che andrà poi a svegliare il PIC grazie al pin dedicato agli interrupt esterni.

2.3. Scelta del timer esterno

Il timer deve essere in grado di gestire una temporizzazione di decine di minuti e di generare al termine del conteggio un segnale interpretabile come interrupt dal PIC.

La scelta è ricaduta sul TPL5010, in grado di gestire un intervallo di tempo fino a 7200s e di generare al suo termine il passaggio di una porta dallo stato 0 allo stato 1, tutto ciò con un consumo di 35nA durante la fase di conteggio (che diventano circa 40 con alimentazione a 3V).

Un altro dispositivo che era stato preso in considerazione è il TPL5110 caratterizzato dallo stesso funzionamento, fatto salvo che al posto della generazione di un interrupt si propone di gestire uno switch/mos così da poter interrompere l'alimentazione del microcontrollore.
Data la necessità di inserire uno switch e visto il basso consumo in sleep del PIC scelto, questa seconda soluzione risulta non essere la migliore.
Considerando per esempio lo switch consigliato dal datasheet del componente, vale a dire il TPS22860, la corrente di leakage tipica è di 12nA ma la massima è di 150nA. Inoltre con questa soluzione si perderebbe la memory retention della RAM durante la fase di non attività del microcontrollore.
Questa seconda configurazione, utile in caso di utilizzo di uC più potenti e caratterizzati da correnti di sleep maggiori, non troverà però applicazione nel seguito di questo elaborato.

	TPL5010	TPL5110 +TPS22860	
Stato uC in fase di non operatività	Sleep	Spento	
Memory retention	Sì	No	
Consumo con V_{DD}= 3V	70nA	Typ: 52nA	Max: 190nA

Tabella 2 Confronto tra i consumi nel caso di utilizzo dei timer TPL5010 e TPL5110 con switch TPS22680

2.4. Scelta del sensore

A titolo puramente esemplificativo, si è scelto di fare uso di un sensore di temperatura.
Il PIC16LF1824 contiene al suo interno un sensore di temperatura descritto nel documento AN1333 [6]. Si tratta di un trasduttore analogico, che produce in uscita una tensione variabile in funzione della temperatura, basandosi sulla tensione di soglia di un transistor interno.
Il segnale analogico generato viene poi portato all'ingresso dell'ADC interno del microcontrollore e convertito in un dato digitale a 10 bit.
L'assorbimento di corrente, durante la conversione è pari a 250µA, mentre la durata della conversione è di 11µs. Nonostante l'elevato consumo di corrente, la breve durata della conversione rende il sensore interno descritto nel documento AN1333 un ottimo candidato dal punto di vista dei consumi. Il produttore consiglia però di attendere almeno 200µs dopo l'accensione dell'ADC prima della lettura del primo dato per dare tempo al sensore di stabilizzarsi. Dal momento che l'applicazione richiede una conversione singola seguita da un periodo di inattività, questo diventa il parametro temporale da considerare.
Quello che ha portato ad abbandonare la scelta di questo sensore è però la scarsa precisone. Nonostante non ne sia specificata esattamente l'accuratezza, è però possibile vedere dal grafico sottostante come anche in presenza di una calibrazione a due punti, l'errore sia dell'ordine di alcuni gradi, valore inaccettabile nella maggioranza delle applicazioni.

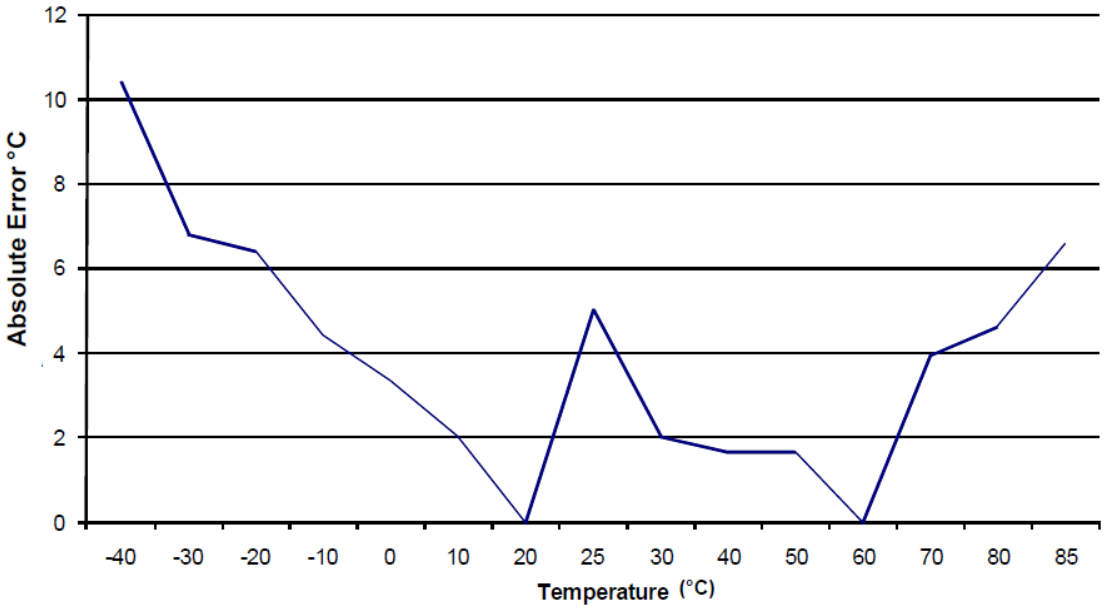

Figura 6 Andamento dell'errore di temperatura dell'AN1333 in presenza di calibrazione a due punti eseguita a 20°C e 60°C **[6]**

Per le ragioni precedentemente viste, la scelta è ricaduta su un sensore esterno, per l'esattezza il TMP112, che dispone di un'accuratezza massima di 0,5°C in assenza di calibrazione, una corrente di quiescenza durante la conversione pari a 40µA, e una corrente di standby di 1µA quando il bus seriale risulta inattivo.
L'unico punto negativo del dispositivo è la corrente di standby che, non risulta essere in linea con i requisiti del progetto, aspetto che ha portato alla decisione di fornire l'alimentazione tramite microcontrollore, così da poter accendere e spegnere il sensore al bisogno.
E' bene sottolineare che nell'intero panorama dei trasduttori di temperatura, non sono stati reperiti attualmente dispositivi in grado di soddisfare i requisiti di standby richiesti.

	Corrente conversione	Tempo 1^{a} conversione	Risoluzione [bit]	Accuratezza	Necessità calibrazione
AN1333+ADC	250µA	200µs	10	N.D.	necessaria
TMP112	40µA	26ms	12	±0.5°C	facoltativa

Tabella 3 Confronto tra i sensori AN1333+ADC interno e TMP112

3. Funzionamento dei componenti

3.1. Microcontrollore PIC16LF1824

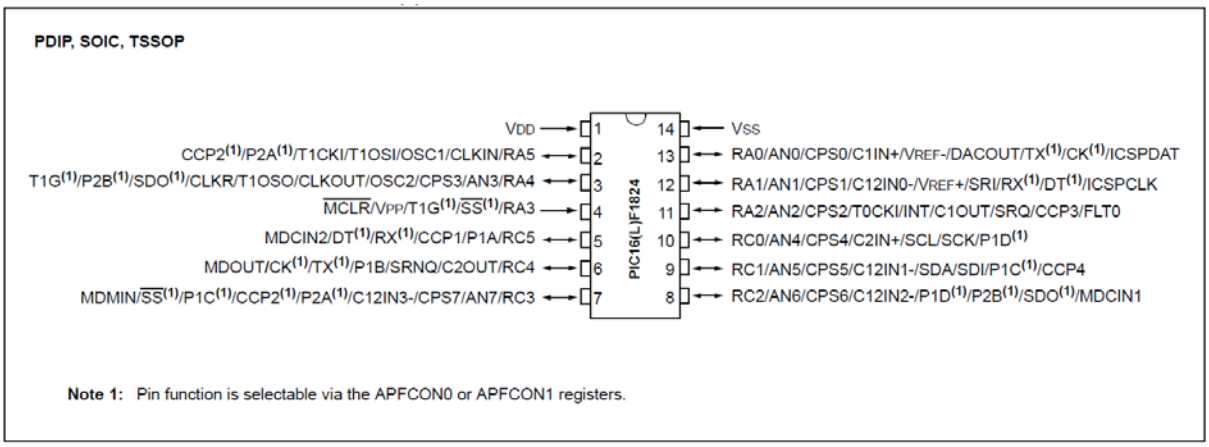

Figura 7 Pinout PIC16LF1824 **[7]**

Il PIC16LF1824 fa parte della famiglia PIC16 di microcontrollori a 8 bit prodotti dalla Microchip. La CPU appartiene alla categoria enhanced mid-range ed è dotata di 49 istruzioni tutte eseguite in un solo ciclo di clock, fatta eccezione per le istruzioni di branch.
Dispone di una memoria programma di tipo flash da 8 kByte nella quale sono salvati il programma scritto dall'utente e le configuration word, una memoria dati RAM da 256 byte ed una memoria dati EEPROM da 256 byte.

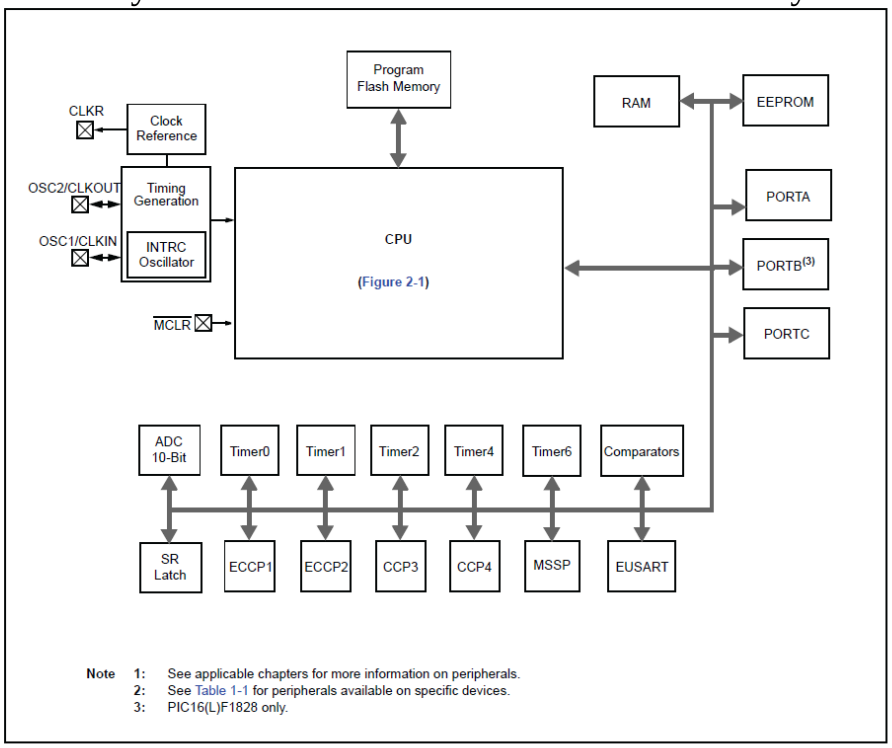

Figura 8 Schema a blocchi PIC16LF1824 **[7]**

Come mostrato nello schema a blocchi sono presenti numerose periferiche sia analogiche che digitali, selezionabili e configurabili grazie a degli appositi registri, tra le quali ricordiamo, poiché utilizzate nel progetto:

- Una porta MSSP (Master Synchronous Serial Port) configurabile come protocollo SPI o I^2C
- Un pin in grado di generare un interrupt a fronte di un cambiamento di stato
- Una porta ICSP (In-Circuit Serial Programming) per consentire la programmazione seriale
- Tre oscillatori interni: uno con frequenza regolabile, mediante prescaler, fino a 32MHz, gli altri due con frequenza fissata rispettivamente a 500kHz e 31kHz (bassa potenza)
- Dei pull-up interni connessi ai pin

E' importante notare, come mostrato nel pinout del componente, che ad ogni pin sono associate diverse configurazioni, che vanno dalla generica porta I/O ad una specifica funzione legata direttamente ad una determinata periferica. Tali configurazioni possono essere selezionate mediante i registri APFCON0 e APFCON1 e le funzioni associate a ciascun pin sono mostrate nella tabella sottostante.

I/O	14-Pin PDIP/SOIC/TSSOP	16-Pin QFN	A/D	Reference	Cap Sense	Comparator	SR Latch	Timers	ECCP	EUSART	MSSP	Interrupt	Modulator	Pull-up	Basic
RA0	13	12	AN0	VREF- DACOUT	CPS0	C1IN+	—	—	—	TX(1) CK(1)	—	IOC	—	Y	ICSPDAT ICDDAT
RA1	12	11	AN1	VREF+	CPS1	C12IN0-	SRI	—	—	RX(1) DT(1)	—	IOC	—	Y	ICSPCLK ICDCLK
RA2	11	10	AN2	—	CPS2	C1OUT	SRQ	T0CKI	CCP3 FLT0	—	—	INT/ IOC	—	Y	—
RA3	4	3	—	—	—	—	—	T1G(1)	—	—	$\overline{SS}$(1)	IOC	—	Y	$\overline{MCLR}$ VPP
RA4	3	2	AN3	—	CPS3	—	—	T1G(1) T1OSO	P2B(1)	—	SDO(1)	IOC	—	Y	OSC2 CLKOUT CLKR
RA5	2	1	—	—	—	—	—	T1CKI T1OSI	CCP2 P2A(1)	—	—	IOC	—	Y	OSC1 CLKIN
RC0	10	9	AN4	—	CPS4	C2IN+	—	—	P1D(1)	—	SCL SCK	—	—	Y	—
RC1	9	8	AN5	—	CPS5	C12IN1-	—	—	CCP4 P1C(1)	—	SDA SDI	—	—	Y	—
RC2	8	7	AN6	—	CPS6	C12IN2-	—	—	P1D(1) P2B(1)	—	SDO(1)	—	MDCIN1	Y	—
RC3	7	6	AN7	—	CPS7	C12IN3-	—	—	CCP2(1) P1C(1) P2A(1)	—	$\overline{SS}$(1)	—	MDMIN	Y	—
RC4	6	5	—	—	—	C2OUT	SRNQ	—	P1B	TX(1) CK(1)	—	—	MDOUT	Y	—
RC5	5	4	—	—	—	—	—	—	CCP1 P1A	RX(1) DT(1)	—	—	MDCIN2	Y	—
VDD	1	16	—	—	—	—	—	—	—	—	—	—	—	—	VDD
VSS	14	13	—	—	—	—	—	—	—	—	—	—	—	—	VSS

Figura 9 Tabella di allocazione pin PIC16LF1824 **[7]**

3.1.1. Protocollo I^2C

Si tratta di un sistema di comunicazione seriale, multi-master multi-slave, con clock esplicito, sviluppato dalla Philips.
Questo protocollo prevede l'utilizzo di un bus formato da due linee dette SCL (Serial Clock) e SDA (Serial Data).
La temporizzazione è comandata sempre dal master che è anche l'unico a poter iniziare una comunicazione e, per farlo, deve indirizzare lo slave cui vuole trasmettere. Quest'ultimo risulta essere identificato da un indirizzo a 7 o più raramente a 10 bit, univoco all'interno della stessa rete.
Entrambe le linee sono di tipo bidirezionale open-drain e questo richiede che siano connesse all'alimentazione attraverso delle resistenze di pull-up. In questa maniera si troveranno ad un valore alto-debole, corrispondente ad un 1 logico, fino a quando le porte dei dispositivi saranno poste in alta impedenza, per passare ad uno 0 logico quando queste costituiranno un collegamento verso massa.
In questo modo se due dispositivi tentassero di comandare la stessa linea contemporaneamente, imponendo rispettivamente un 1 e uno 0, il secondo prevarrebbe sul primo. Ci si trova dunque davanti ad un meccanismo di arbitraggio che consente di prevenire conflitti elettrici.
Non è necessario che i dispositivi siano alimentati con la stessa tensione purché ci sia compatibilità di livelli.
Con questo protocollo è possibile raggiungere una velocità di 400kbit/s in modalità standard e di 3.4Mbit/s in modalità high-speed.

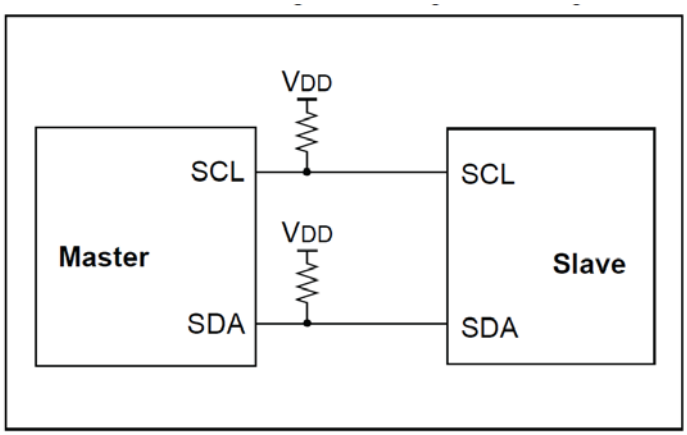

Figura 10 Esempio di connessione I^2C in presenza di un solo master e un solo slave **[7]**

La comunicazione inizia con la generazione di un segnale di start da parte del master, determinato da una transizione alto-basso della linea SDA mentre SCL è lasciata a livello alto.
Il primo byte è trasmesso dal master ed è costituito nei primi 7 bit dall'indirizzo dello slave con cui si intende comunicare, mentre l'ultimo bit detto R/W serve per indicare se si vuole leggere (bit a 1) o scrivere (bit a 0).
Segue un bit di acknowledge lasciato a 1 dal master e quindi comandato dallo slave (portato a 0 in caso di corretta ricezione).
Seguono poi le trasmissioni di dati comandate dal master o dallo slave in base al precedente valore di R/W. Ogni dato è lungo 8 bit e trasmesso dal bit più significativo a quello meno significativo. Ad ogni dato segue un ACK comandato da chi sta ricevendo.
La comunicazione è interrotta dal master con una condizione di stop (facendo cioè variare la linea SDA dallo stato 0 all'1 in corrispondenza di un 1 su SCL) oppure con un nuovo start.

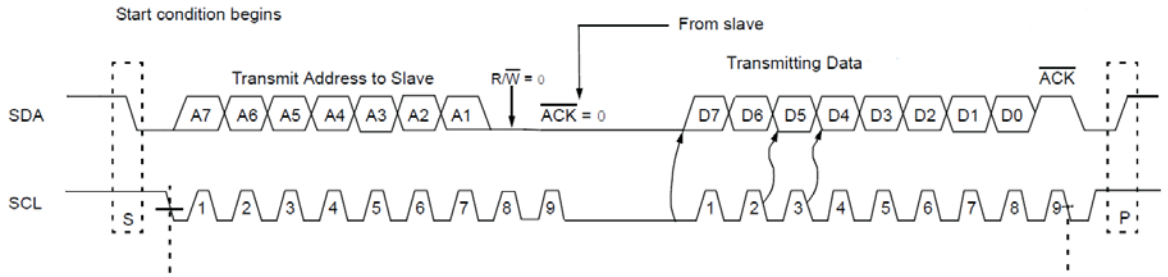

Figura 11 Esempio di trasmissione di un dato da parte del master **[7]**

La porta MSSP del PIC16LF1824 può essere impostata per funzionare con questo protocollo sia in modalità master che slave. Ai fini del progetto si è utilizzata la modalità master che può funzionare in ricezione, quando cioè si vuole leggere da una determinata periferica, oppure in trasmissione quando si vuole cioè inviare dati a quest'ultima.
I registri che permettono di controllare la porta MSSP sono i seguenti:

- SSP1STAT: SSP1 status register
- SSP1CON1: SSP1 control register 1
- SSP1CON2: SSP1 control register 2
- SSP1CON3: SSP1 control register 3
- SSP1MSK: SSP1 mask register

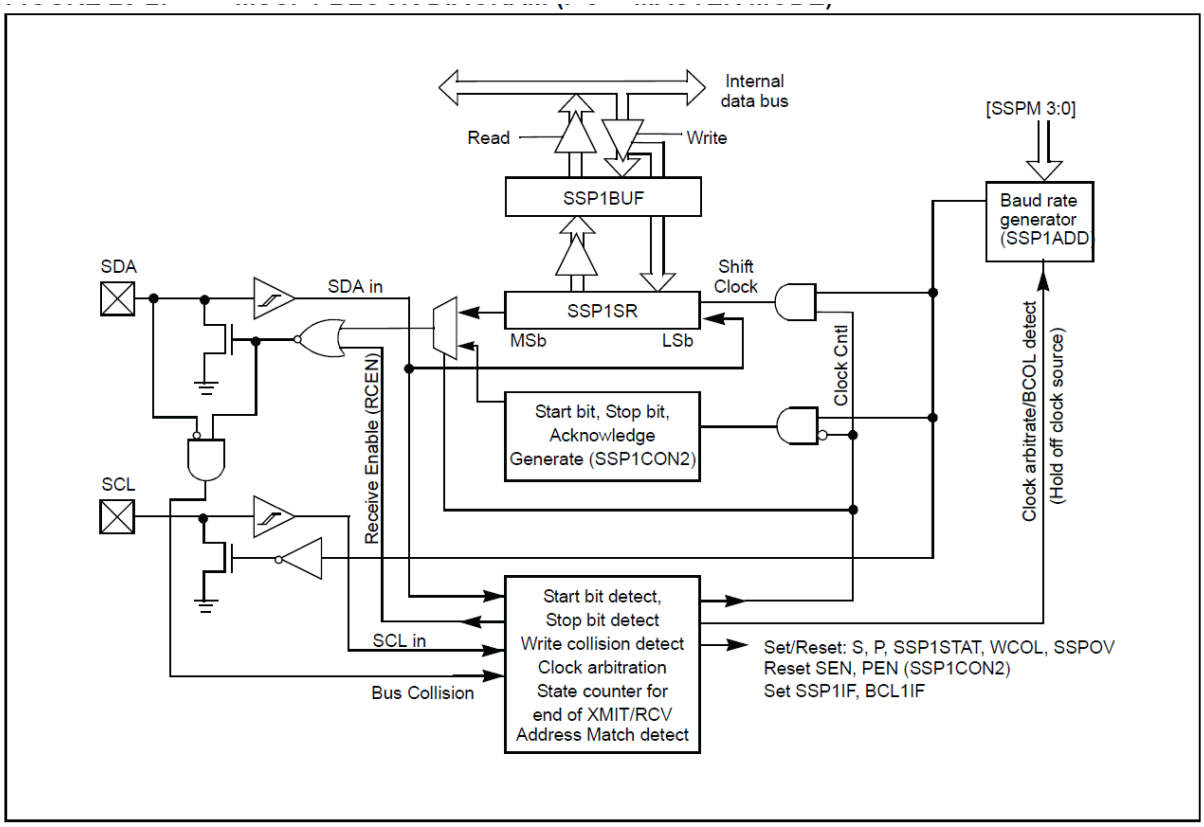

Figura 12 MSSP Diagramma a blocchi I^2C master mode **[7]**

3.1.2. Modalità Power-Down (Sleep)

E' una modalità che consente di ridurre drasticamente il consumo entrando in uno stato di risparmio energetico in cui il processore è spento mentre le periferiche possono essere arrestate o meno a seconda del singolo caso.
Per entrare in questa modalità è presente un'apposita istruzione assembler chiamata appunto "sleep", mentre è possibile uscirne in diversi modi; quelli utilizzati nel progetto sono due:

- Reset esterno sul pin MCLR (pin 4)
- Interrupt esterno sull'apposito pin (pin 11).

Di particolare interesse è il secondo caso: in presenza di una commutazione, viene generato un interrupt.
Nel caso in cui il bit di configurazione globale degli interrupt (GIE) sia attivo, viene avviata l'apposita routine, terminata la quale si torna all'istruzione successiva allo sleep.

Se invece GIE non fosse attivo si uscirebbe direttamente dallo sleep passando all'istruzione successiva.

Al fine di massimizzare il risparmio di energia è necessario che tutte le periferiche siano spente preventivamente, che non ci siano pin flottanti e neppure pin che assorbono corrente. Affinché i piedini non utilizzati non restino flottanti è possibile settare dei pull-up interni definiti "weak pull-ups" mediante apposita configurazione dei registri di PORTA e PORTC.
Se tutti questi requisiti sono soddisfatti il consumo dichiarato per tensione di alimentazione pari a 3V è di soli 30nA.

3.1.3. Oscillatore

L'oscillatore dispone di una varietà di modalità di funzionamento:

- EC (External Clock) → il clock è fornito da un circuito esterno.
- LP, XT, HS → è presente un quarzo esterno che, connesso alla circuiteria interna, costituisce un oscillatore di Pierce. Le tre sigle indiano in ordine le modalità a bassa potenza, medio guadagno e alto guadagno.
- RC → viene connesso un circuito RC esterno.
- INTOSC → è selezionato uno dei tre oscillatori interni LFINTOSC (31kHz), MFINTOSC(500kHz), HFINTOSC (regolabile fino a 32MHz).

E' possibile selezionare le modalità di interesse ed i settaggi ad essa collegati mediante l'uso di tre registri:

- OSCON (Oscillator Control Register)
- OSCSTAT (Oscillator Status Register)
- OSCTUNE (Oscillator Tuning Register)

Per scegliere l'oscillatore da utilizzare si è deciso, come per tutti gli aspetti del progetto, di basarsi sul criterio del consumo, non avendo altri particolari vincoli da rispettare.
Il seguente grafico mostra l'assorbimento di corrente in relazione ad alcune delle possibili modalità di funzionamento.

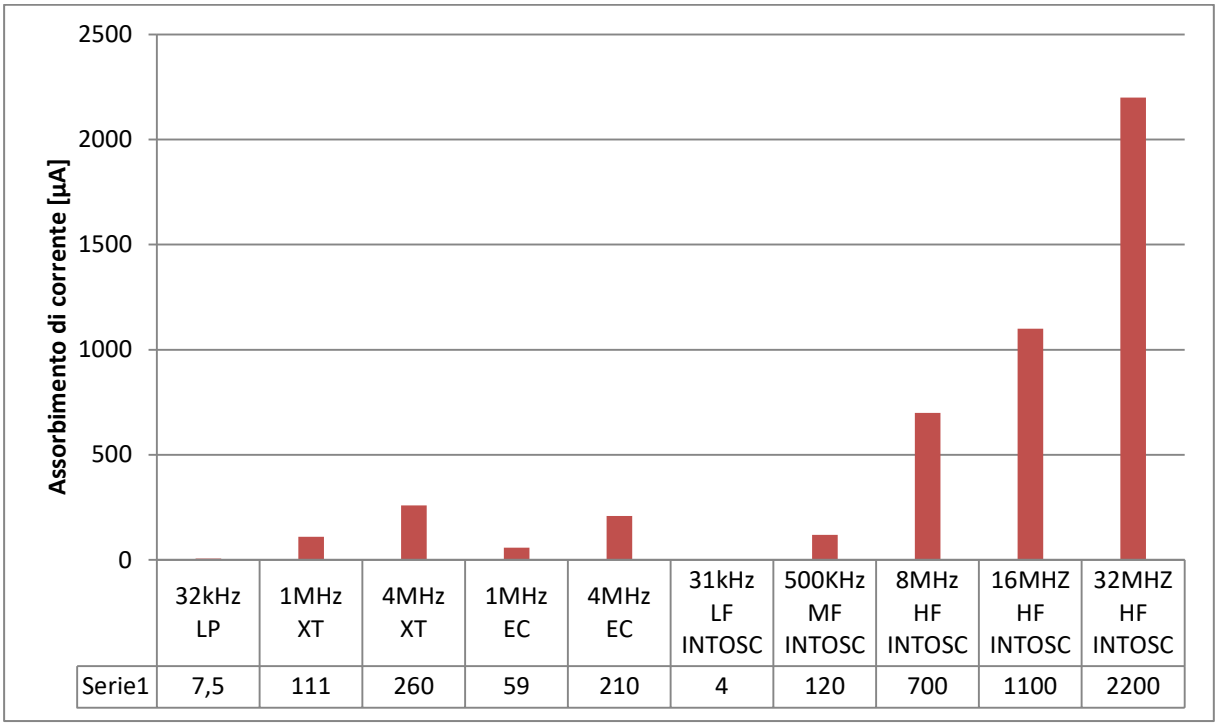

Figura 13 Assorbimento di corrente in funzione dell'oscillatore scelto

Dal grafico sembra facile dedurre che la scelta migliore per ridurre il consumo sia l'oscillatore interno a bassa frequenza, con un assorbimento di soli 4µA.
La precedente considerazione non tiene però conto del fatto che, per ogni oscillatore, il tempo necessario al microcontrollore per svolgere un'operazione è diverso.
Ad esempio è vero che il consumo dell'oscillatore interno a bassa frequenza è nettamente più basso di quello a 32MHz, ma quest'ultimo impiega un tempo molto più basso per svolgere un'operazione, assorbendo la corrente indicata per un tempo inferiore.
Diventa dunque interessante fare delle considerazioni energetiche piuttosto che di assorbimento di corrente. Con il prodotto tra la corrente assorbita e la tensione di alimentazione si ottiene la potenza dissipata, che moltiplicata per il periodo di clock, dato dall'inverso della frequenza, fornisce proprio l'energia assorbita per compiere un'istruzione.

$$E_{istr} = V_{CC} * I * T_{CK}\ [J]$$

Nel grafico sottostante sono mostrati i valori di energia associati alle già viste possibili configurazioni.

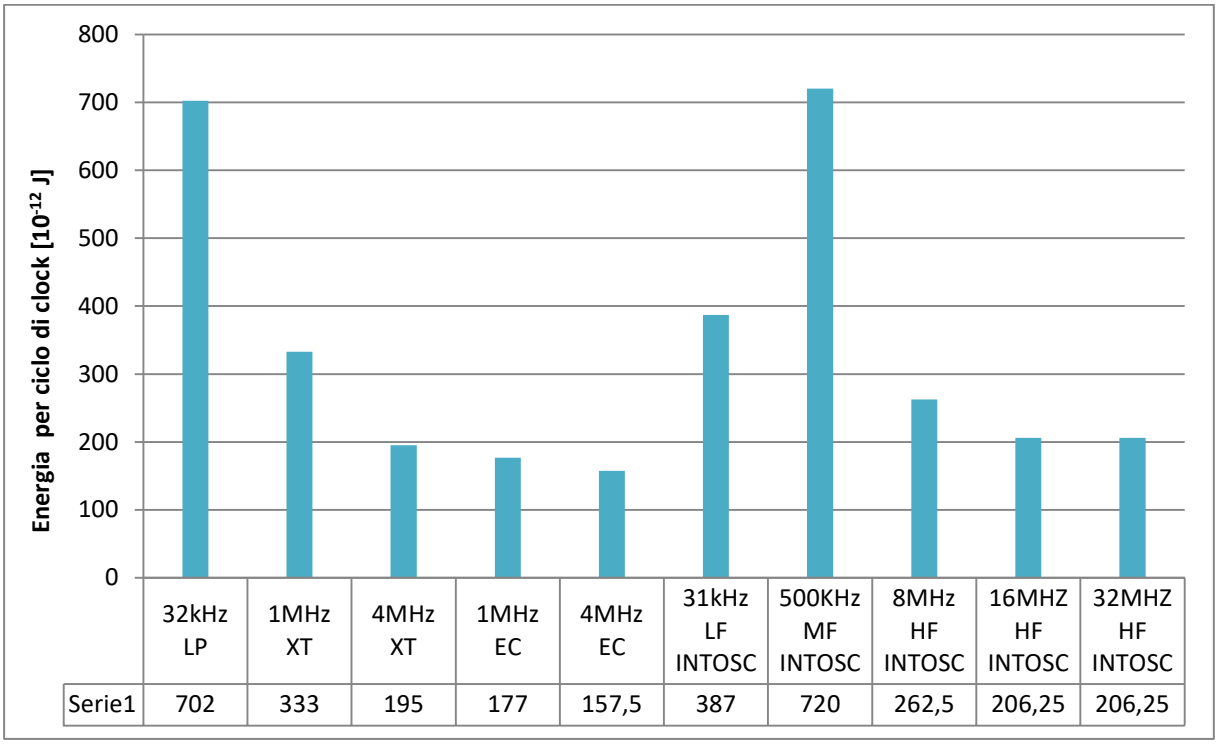

Figura 14 Energia per ciclo di clock in funzione dell'oscillatore scelto

Il grafico dell'energia risulta essere notevolmente diverso dal precedente, mostrando come l'oscillatore a HFINTOSC a 32MHZ che dal punto di vista della corrente risultava il peggiore, sia invece uno dei migliori dal punto di vista energetico, grazie al breve tempo in cui viene svolta un'operazione. La situazione è opposta per l'LP a 32kHz.

Anche i dati dell'energia non risultano sufficienti da soli per prendere una decisione. Lo sarebbero se nell'esecuzione del programma non ci fossero dei tempi fissi, impostati con dei "delay" e necessari al fine del corretto funzionamento degli altri componenti (tempi di acquisizione, tempi di lettura…).
Si è deciso dunque di rappresentare un ulteriore grafico, riportante una stima del consumo di energia totale per ogni volta che il microcontrollore si sveglia, usando le seguenti ipotesi, in linea con le specifiche del progetto:

- Un tempo fisso imposto dai ritardi $T_F = 50ms$
- Un numero di istruzioni da eseguire $n_{istr} = 500$

Per il calcolo dell'energia impiegata è sufficiente sommare il tempo fisso al tempo di esecuzione delle istruzioni e moltiplicarlo poi per la potenza assorbita:

$$E = (T_F + n_{istr} * T_{CK}) * V_{CC} * I$$

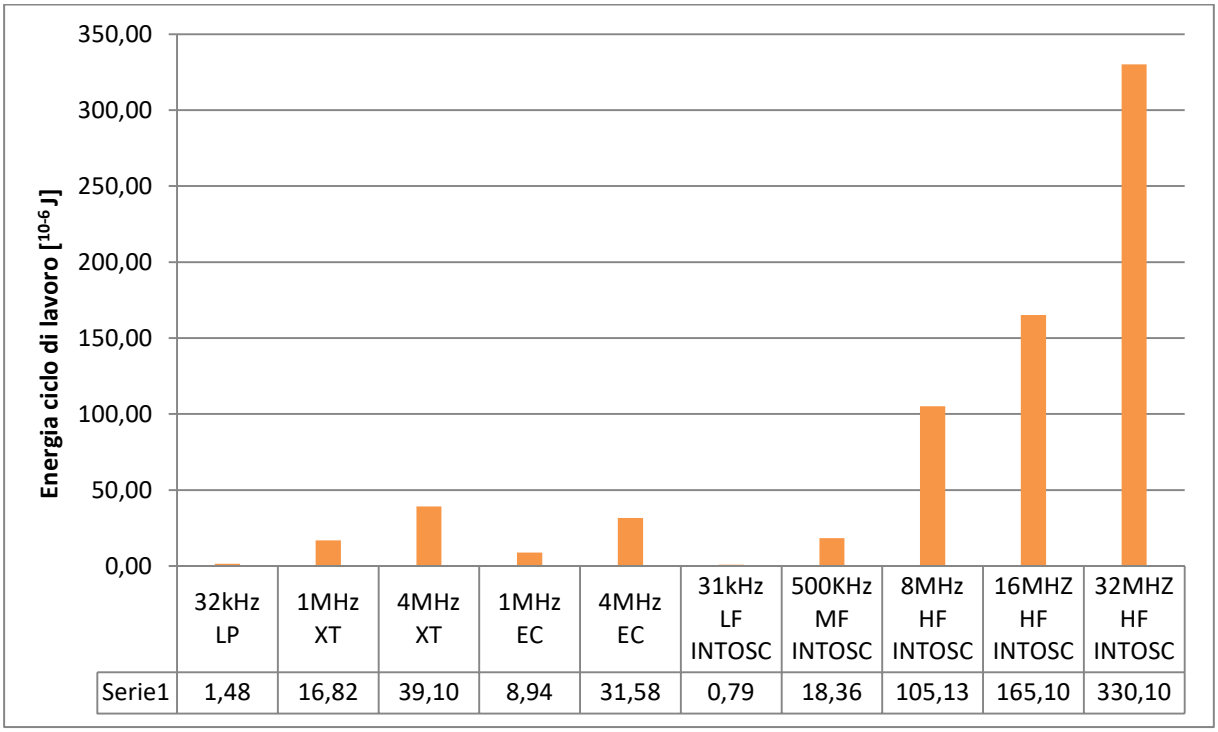

Figura 15 Energia per ciclo di lavoro in base all'oscillatore scelto

Questi nuovi dati sono in linea con quelli presentati nel primo grafico a causa dei ritardi relativamente lunghi rispetto al numero di istruzioni.
Questo porta alla decisione definitiva di scegliere l'oscillatore LF INTOSC a 31kHz.

3.1.4. Data EEPROM

La EEPROM (Electrically Erasable Programmable Read-Only Memory) è una memoria non volatile utilizzata per memorizzare piccole quantità di informazioni che cambiano frequentemente, ma che devono essere mantenute anche in assenza di alimentazione. Può essere scritta e letta mediante i registri EEDATL (eeprom data register), EECON1 (eeprom control 1 register) e EECON2 (eeprom control 2 register).
Questa periferica può essere utilizzata per implementare la funzione di data-logger prevista dal sistema.
Ha una dimensione di 256 byte.

3.1.5. In-Circuit Serial Programming (ICSP)

ICSP consente di programmare il microcontrollore anche quando questo è già saldato sul PCB consentendo di aggiornarlo con firmware recenti.
I pin coinvolti nella programmazione sono i seguenti:

- ICSPCLK: input del clock
- ICSPDAT: input/output dati
- MCLR/V_{PP} : selezione della modalità di programmazione
- V_{DD}
- V_{SS}

E' possibile impiegare due modalità di programmazione denominate rispettivamente "low voltage" e "high voltage".
Nel primo caso sul pin MCLR è fornita una tensione pari a V_{DD} e questo elimina il rischio di danneggiare altri componenti collegati. Tuttavia questa modalità è sconsigliata poiché non consente di sfruttare le funzionalità di debugging dei programmatori.
Nel secondo sul pin MCLR è presente una tensione più elevata pari a 9V che potrebbe causare danni a componenti esterni. Nel progetto si utilizzerà questa modalità facendo uso di un jumper in fase di programmazione, per scollegare componenti che potrebbero subire danni.
Per programmare il microcontrollore si farà uso del PICkit3, un programmatore/debugger economico ma sufficientemente potente, in grado di gestire la programmazione ICSP.
E' importante sottolineare che i pin V_{DD} e V_{SS} possono essere utilizzati per fornire alimentazione al uC o per leggerla. Nel primo caso è importante non siano presenti altre fonti di alimentazione.

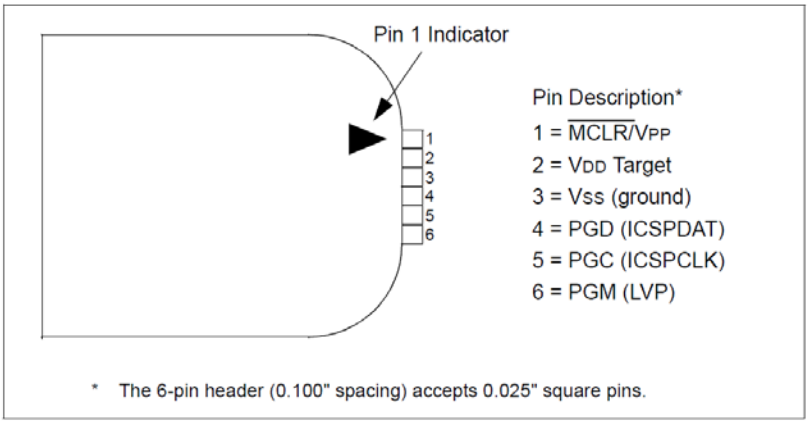

Figura 16 Pinout PICkit 3 **[7]**

3.2. Timer: TPL5010

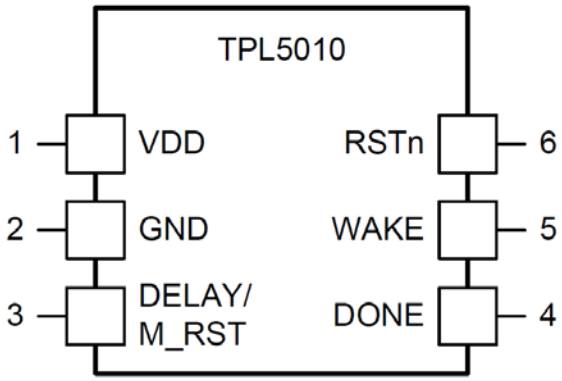

Figura 17 Pinout TPL5010 **[8]**

Il TPL5010 è un timer a basso consumo di potenza che implementa la funzione di watchdog. E' ideato per essere utilizzato in applicazioni alimentate a batteria in cui il microcontrollore rimane per la maggior parte del tempo in modalità di sleep. Tale timer è utilizzabile per attivare il uC a intervalli costanti sostituendo quello interno, caratterizzato tipicamente da un consumo di corrente maggiore.
Il consumo dichiarato alla temperatura di 25°C e con alimentazione a 3V è pari a 40nA e l'accuratezza dell'oscillatore contenuto al suo interno è dello 0.5%.

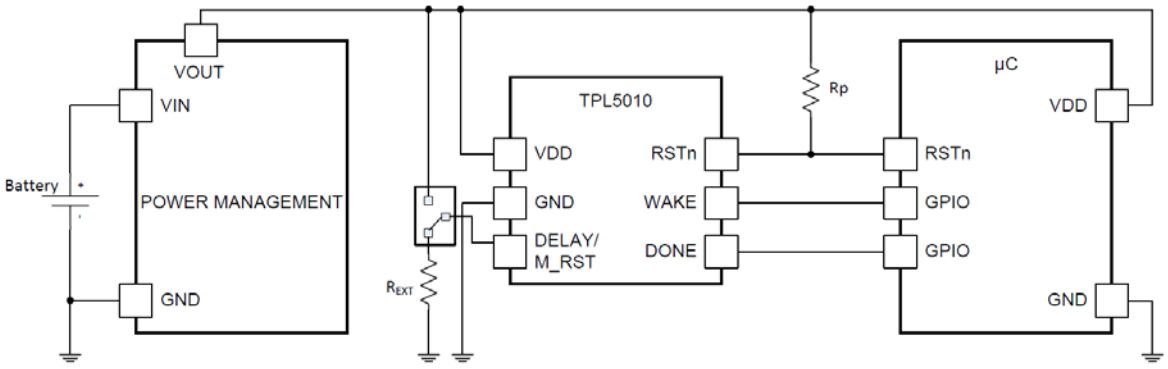

Figura 18 Esempio di collegameno tipico del TPL5010 **[8]**

3.2.1. Descrizione del funzionamento

Pin	Tipologia	Funzionalità associate
DELAY/M_RST	I	La resistenza tra questo pin e massa determina l'intervallo. Se connesso a V_{DD} causa un reset
WAKE	O	Segnale per risvegliare il microcontrollore ad intervalli prestabiliti
DONE	I	Segnale generato dal µC per indicare l'avvenuto risveglio
RSTn	O	Segnale digitale per il reset del µC, necessita di resistenza di pull-up

Tabella 4 Funzionalità associate ai pin del TPL5010

Startup

Durante lo startup il dispositivo esegue una misurazione del valore di resistenza collegato al pin DELAY/M_RST al fine di determinare l'intervallo tra gli interrupt per il risveglio del microcontrollore.

Questa fase ha una durata di 100ms, durante la quale l'assorbimento di corrente tipico è di 200µA. Il pin RSTn è tenuto ad un valore basso, con lo scopo di resettare il microcontrollore.

Questa fase si verifica all'accensione o in seguito al reset del dispositivo.

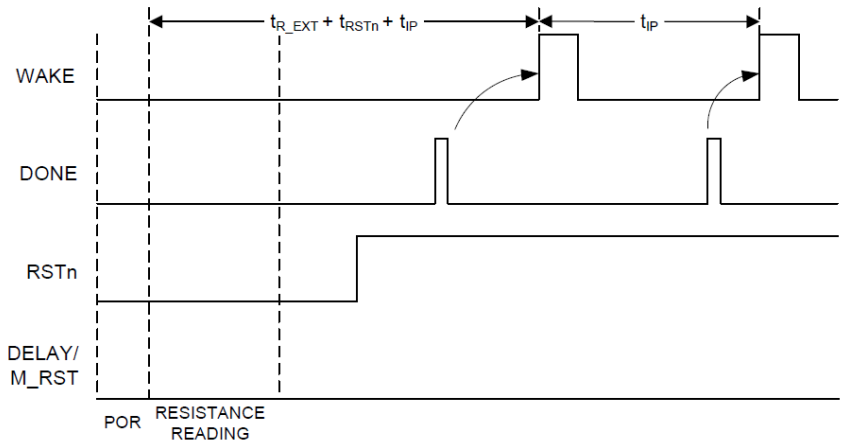

Figura 19 Startup TPL5010 **[8]**

Modo operativo normale

In questa fase il timer fornisce impulsi periodici sul pin WAKE in risposta a validi impulsi di DONE provenienti dal microcontrollore.

E' necessario che il microcontrollore mandi un impulso di DONE della durata di 20ms almeno 20ms prima del successivo impulso di WAKE. Se questo si verifica il tempo tra un impulso di WAKE e l'altro è dovuto alla temporizzazione scelta. In caso contrario, al termine del conteggio, si presenta una transizione alto-basso sul pin RSTn con lo scopo di resettare il µC.

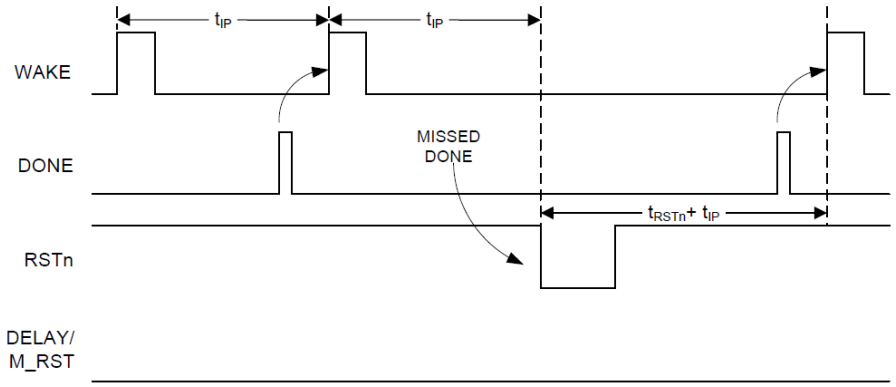

Figura 20 Modo operativo normale TPL5010 **[8]**

Reset manuale

Se al pin DELAY/M_RST viene applicata una tensione pari all'alimentazione per un tempo di almeno 20ms, il dispositivo rileva una condizione di reset manuale che gli fa perdere la temporizzazione memorizzata. Tipicamente a questo piedino è connesso uno switch che consente di passare da V_{CC} alla resistenza utilizzata per determinare la temporizzazione.

In corrispondenza di un reset manuale il pin RSTn è portato ad uno stato logico basso al fine di resettare anche il microcontrollore.

3.2.2. Impostazione della temporizzazione

Per selezionare la temporizzazione dev'essere inserita una resistenza di valore compreso tra 500Ω e 170kΩ tra il pin DELAY/M_RST e massa. Durante la lettura della resistenza il pin è connesso ad un circuito analogico tramite un mux. Al termine questo circuito viene spento ed il pin viene connesso ad un circuito digitale.

La formula per determinare il valore della resistenza da inserire in funzione dell'intervallo desiderato è la seguente:

$$R_{EXT} = 100\frac{-b + \sqrt{b^2 - 4a(c - 100T)}}{2a} \ [\Omega]$$

Dove *T* è l'intervallo di tempo desiderato, R_{EXT} la resistenza da inserire nel circuito, *a*, *b*, *c* sono coefficienti dipendenti dall'intervallo e specificati nel datasheet del componente.
Nel nostro caso, volendo *T=1800s,* la formula diviene:

$$R_1 = 100\left(\frac{136.2571+\sqrt{(-136.2571)^2-4*0.3177(34522.4680-100*1800)}}{2*0.3177}\right) = 92.4k\Omega$$

Non essendo disponibile tale valore di resistenza, si è scelto un valore di *91k*Ω. Invertendo la formula e inserendo al posto di R_1 l'effettivo valore scelto è possibile trovare il valore di T:

$$T = a * {R_1}^2 * 10^{-6} + b * R_1 * 10^{-4} + c * 10^{-2} \cong 1745\ s$$

Dato che la precisione del timer dipende inevitabilmente dall'incertezza della resistenza, si è scelto per R_1 di utilizzare un dispositivo con tolleranza dello 0.1%.

3.3. Sensore di temperatura TMP112

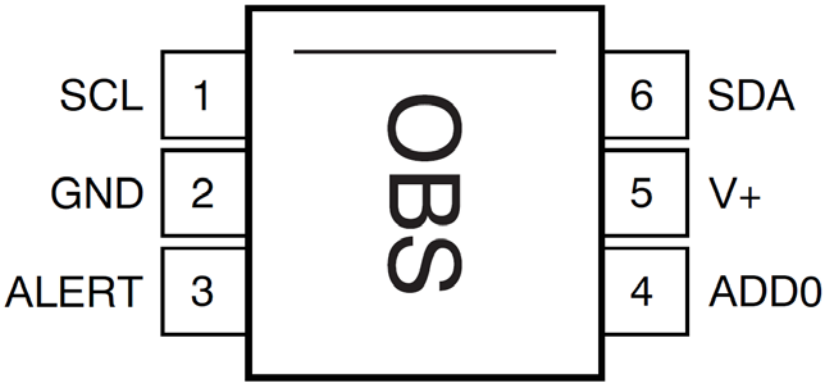

Figura 21 Pinout TMP112 **[9]**

Pin	Tipologia	Funzionalità associate
SCL	I	Ingresso del clock seriale; è open-drain e necessita di una resistenza di pull up
SDA	I/O	Dato seriale; è open-drain e necessita di una resistenza di pull up
ALERT	O	Allarme di sovratemperatura, è open-drain e necessita di una resistenza di pull up
ADD0	I	Selezione dell'indirizzo per la comunicazione

Tabella 5 Funzionalità dei pin del TMP112

Il TMP112 è un sensore di temperatura digitale con un'accuratezza di ±0.5°C senza necessità di calibrazione (in presenza di calibrazione l'accuratezza può arrivare a ±0.17°C). L'uscita del sensore è pressoché lineare e questo evita di dover utilizzare tabelle per comprendere il valore di temperatura.
Il sensore è il chip stesso: la temperatura è rilevata sia dai pin che dal package ma quest'ultimo costituisce il percorso termico principale per via della bassa resistenza termica dei pin.
L'ADC integrato a 12 bit offre una risoluzione di 0.0625°C (LSB).
L'uscita è di tipo digitale ed è compatibile con le interfacce SMBus, two-wire e I^2C.
Può funzionare in varie modalità, ma all'avvio è impostata quella di conversione continua. Il dispositivo esegue delle conversioni di durata 26ms consumando 40µA di corrente. Queste avvengono con frequenza 4Hz e tra una e l'altra il consumo si riduce a 10µA.

Esiste anche un modo di conversione detto One-Shot, durante il quale viene eseguita una sola conversione con un consumo pari a 20µA. Successivamente il dispositivo passerà in standby con un consumo di soli 0,5µA risparmiando quindi sui consumi.
Però, dal momento che per ridurre i consumi, si intende spegnere il sensore dopo ogni misurazione, e che all'avvio parte automaticamente la modalità di conversione continua che esegue immediatamente il primo campionamento, non si utilizzerà questa modalità apparentemente più conveniente.

3.3.1. Interpretazione dell'uscita digitale

L'uscita digitale ad ogni misurazione di temperatura è salvata in un registro di sola lettura, che può essere configurato come registro a 12 bit o a 13 bit per aver rispettivamente un intervallo di temperature misurabili -55÷128°C oppure -55÷150°C. Si utilizzerà la prima di queste due modalità dato che non risulta necessario estendere il range; a questo scopo il bit EM (Extended Mode) del configuration register dev'essere posto a 0 (valore di default).
Per convertire la parola di 12 bit nel corrispondente valore di temperatura è necessario lavorare in due modi diversi a seconda che il dato sia positivo (MSB=0) oppure negativo (MSB=1):

- **Caso positivo**: la parola binaria dev'essere convertita in decimale e poi moltiplicata per la risoluzione

  Es: $0011\ 0010\ 0000 = 800; \quad 800 * 0.0625 = 50°C$

- **Caso negativo**: dev'essere fatto il complemento a due della parola (rappresentando così in binario il valore assoluto della temperatura); la parola ottenuta dev'essere moltiplicata per la risoluzione e cambiata di segno

  Es. $1110\ 0111\ 0000 \rightarrow$ complemento a $2 \rightarrow 0001\ 1001\ 0000$

  $0001\ 1001\ 0000 = 400; \quad 400 * 0.0625 * (-1) = -25°C$

3.3.2. Programmazione e registri

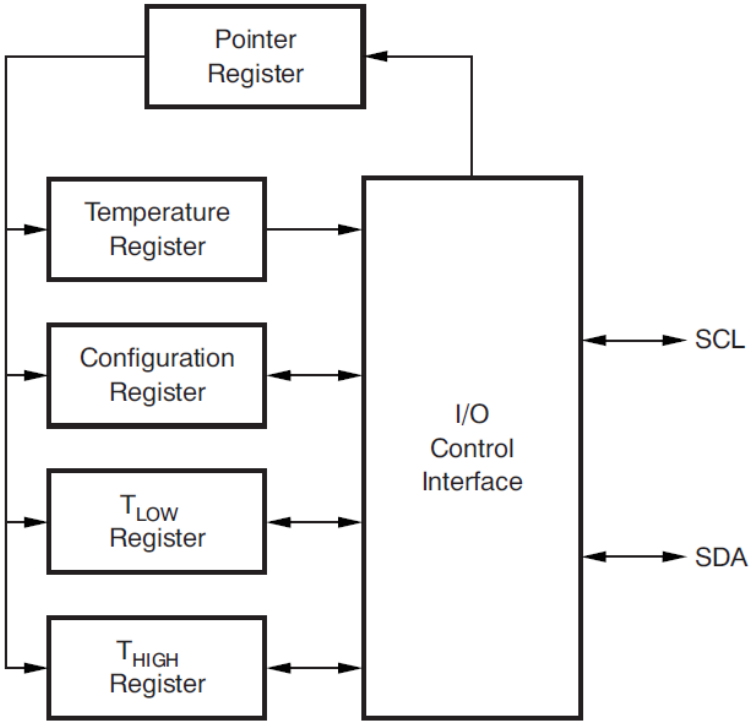

Figura 22 Struttura interna dei registri **[9]**

Pointer Register

In figura è mostrata la struttura interna dei registri. Al fine di accedere ad uno dei 4 registri di scrittura/lettura è necessario l'uso del pointer register. A seconda del valore dei suoi due bit meno significativi, è infatti possibile selezionare quale sia il registro connesso all'interfaccia di I/O che consente di comunicare con l'esterno. I bit più significativi sono invece sempre lasciati a zero. Di default il suo valore è "00" e pertanto è indicizzato il Temperature Register.

Tabella 6 Byte del Pointer Register

P7	P6	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	Bit di registro	

Bit di registro:

- 00 → Temperature Register;
- 01 → Configuration Register
- 10 → T_{LOW} Register (non di interesse)
- 11 →T_{HIGH} Register (non di interesse)

Temperature Register

E’ un registro di sola lettura composto da due byte ed è dove vengono memorizzati i valori di temperatura. Nel primo sono presenti gli 8 bit più significativi, nella metà sinistra del secondo sono presenti i gli altri 4 bit mentre in quella destra sono presenti dei bit posti a 0

Tabella 7 Byte 1 e 2 del Temperature Register

BYTE	D7	D6	D5	D4	D3	D2	D1	D0
1	T11	T10	T9	T8	T7	T6	T5	T4
2	T3	T2	T1	T0	0	0	0	0

Configuration Register

E’ un registro a 16 bit che consente di selezionare le modalità di funzionamento del sensore. In seguito sono descritti i soli bit di interesse

Tabella 8 Byte 1 e 2 del Configuration Register

BYTE	D7	D6	D5	D4	D3	D2	D1	D0
1	OS	R1	R0	F1	F0	POL	TM	SD
2	CR1	CR0	AL	EM	0	0	0	0

- EM (Extended Mode) → consente di selezionare il formato dati a 12 bit (EM=0) o 13 bit (EM=1)
- SD (Standby Mode) → consente di selezionare la modalità di standby (SD=1) con consumo tipico di 0.5µA ; quando SD=0 la modalità selezionata è quella di conversione continua
- OS (One-Shot) → Trovandosi in modalità di standby, ponendo OS=1 è possibile avviare una singola conversione .

3.3.3. Interfaccia seriale

Si farà funzionare il dispositivo in modalità I^2C, non utilizzando il pin ALERT che consentirebbe di implementare le funzionalità aggiuntive del protocollo SMbus. Quanto detto sulle regole del protocollo I^2C nella sezione dedicata al PIC16LF1824 vale anche in questo caso, in cui siamo però in presenza di uno SLAVE. Durante la trasmissione/ricezione l'assorbimento di corrente è pari a 15µA.
Il TMP112 può avere 4 diversi indirizzi (a 7 bit) che vengono selezionati sulla base del segnale cui è collegato il pin ADD0. L'indirizzo corrispondente al collegamento verso massa è "1001000".

Modalità trasmettitore
E' usata per leggere dal registro di memoria indicizzato nel Pointer Register e necessita della seguente sequenza, in cui per brevità non sono indicate le condizioni di start, stop e acknowledge previste dal protocollo:

- Il master invia l'indirizzo 1001000 ponendo il bit R/W a 1 (lettura)
- Lo slave invia il byte più significativo del registro
- Lo salve invia il byte meno significativo del registro

Modalità ricevitore
E' usata per scrivere nei registri:

- Il master invia l'indirizzo 1001000 ponendo il bit R/W a 0 (scrittura)
- Il master invia il valore da assegnare al Pointer Register
- Il master invia il byte più significativo del registro
- Il master invia il byte meno significativo (se necessario)

4. Il Circuito

4.1. Schema elettrico e funzionamento

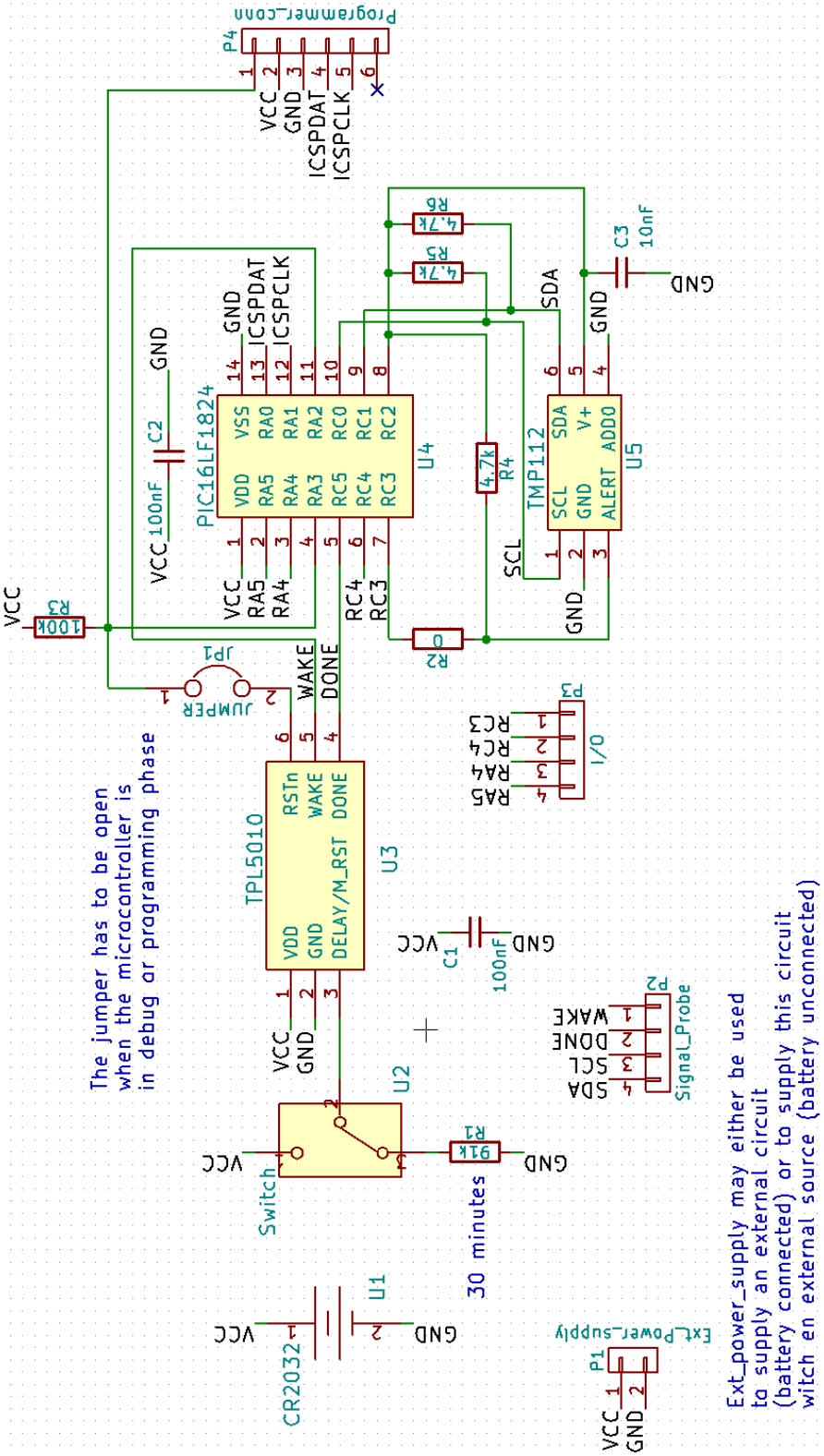

Figura 23 Schema elettrico del circuito

Quello rappresentato in figura è lo schematico del circuito ottenuto dalla connessione dei componenti precedentemente descritti uniti ad alcuni connettori e componenti passivi.

- All'avvio il microcontrollore è acceso e, dopo aver fornito l'alimentazione al sensore e alle resistenze di pull-up, inizia un'operazione di lettura della temperatura tramite una comunicazione seriale mediante l'uso del protocollo I^2C. Terminata la lettura salva il dato ricevuto nella memoria EEPROM interna.
  Allo steso tempo il TPL5010 legge il valore della resistenza R_1 per programmare l'intervallo con cui dovrà fornire gli impulsi di WAKE al PIC (come si è visto lo farà ogni 1745s).
  Terminata questa fase iniziale il microcontrollore spegne il TMP112 per risparmiare energia e poi va in fase di sleep, in attesa della ricezione del segnale di WAKE.
  Quando questo si verifica, il microcontrollore esegue nuovamente la lettura e il salvataggio di un valore di tensione nella stessa maniera vista precedentemente. Al termine invia un segnale di DONE al timer, per indicare che l'interrupt generato è andato a buon fine, per poi tornare in standby fino a nuovo ordine.

- Lo switch (U_2) consente un reset manuale del TPL5010 che a sua volta provvederà a resettare il microcontrollore portando a livello basso il pin RSTn.

- I 3 condensatori C_1, C_2, C_3 sono detti di bypass e servono per evitare eventuali disturbi sull'alimentazione degli integrati.

- R_3 è un resistore di pull-up necessario ai fini del reset del microcontrollore, mentre R_4, R_5, R_6 sono resistori di pull-up necessari per la comunicazione seriale. Questi tre non sono portati direttamente a V_{DD} ma controllati da un pin del microcontrollore. E' da sottolineare però che la connessione per il pin ALERT è solo predisposta ma non utilizzata: infatti R_2, corrispondente ad un cortocircuito, non sarà inserito. E' stata fatta questa scelta progettuale per consentire l'utilizzo di questa funzionalità qualora lo si ritenesse opportuno.

- Il connettore P_2 serve per fornire all'esterno del circuito i principali segnali, così che questi possano essere controllati mediante oscilloscopio o altra strumentazione.

- P_4 è il connettore di programmazione, cui si collega il PicKit3 per programmare il microcontrollore. E' importante che durante la fase di programmazione/debugging, il jumper JP_1 sia aperto, per evitare che le alte tensioni applicate possano danneggiare il timer. Quando JP_1 è aperto si perde però la funzionalità di reset manuale del PIC.

- P_3 fornisce un collegamento con i pin inutilizzati del µC, rendendoli disponibili all'esterno della scheda e consentendo il loro utilizzo in caso di necessità.

- U_1 (CR2032) è la fonte di alimentazione principale del circuito. E' direttamente collegata con P_1, il che consente di fornire l'alimentazione all'esterno, oppure di connettere una diversa fonte di energia, cosa possibile solo nel caso in cui la batteria non sia collegata.

4.2. Il Firmware

Il Firmware con cui programmare il microcontrollore è stato realizzato con MPLAB X, un IDE (Integrated Development Environment) messo a disposizione dalla stessa Microchip. Grazie all'aggiunta del compilatore XC8 è stato possibile programmare in linguaggio C anziché in assembler.
E' stato inoltre utilizzato MPLAB Code Configurator (MCC), un ambiente di programmazione grafico che consente di generare del codice C, pronto per essere richiamato nel main, contenete le impostazioni dei registri desiderati.
Questo strumento genera anche delle librerie per la gestione delle periferiche selezionate.

Per prima cosa risulta necessario impostare i registri CONFIGURATION WORD 1 e CONFIGURATION WORD 2 grazie ai quali, in particolare, risulta possibile:

- selezionare il clock interno;
- disabilitare il watchdog timer;
- attivare la funzionalità di reset collegata al pin MCLR;
- selezionare la modalità di programmazione "high voltage"

```
// CONFIGURATION WORD 1
#pragma config FOSC = INTOSC
#pragma config WDTE = OFF
#pragma config PWRTE = OFF
#pragma config MCLRE = ON
#pragma config CP = OFF
#pragma config CPD = OFF
#pragma config BOREN = OFF
#pragma config CLKOUTEN = OFF
#pragma config IESO = OFF
#pragma config FCMEN = OFF

// CONFIGURATION WORD 2
#pragma config WRT = OFF
#pragma config PLLEN = OFF
#pragma config STVREN = ON
#pragma config BORV = LO
#pragma config LVP = OFF
```

Con il registro OSCON è invece possibile impostare la frequenza dell'oscillatore interno a 31kHz.

Si passa poi all'impostazione dei pin:

- TRIS(A-C) → per selezionare il verso;
- APFCON0,APFCON1 → per selezionare la funzionalità associata;
- ANSEL(A-C) → per selezionare l'ingresso come analogico;
- WPU(A-C) → per impostare i weak pullup.

In questo caso tutti i pin sono impostati come digitali, verso e funzionalità sono stati selezionati sulle base delle singole esigenze, mentre i weak pull-up sono stati abilitati soltanto sui pin non utilizzati.

Per quanto riguarda l'I^2C il PIC è stato impostato per lavorare in modalità master con una frequenza di 1.55kHz

OSCON	SPLLEN	IRCF<3:0>				-	SCS<1:0>	
	0	0	0	0	0	0	1	0
TRISA	-	-	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
	0	0	1	1	1	1	1	1
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
	0	0	0	1	1	0	1	1
ANSELA	-	-	-	ANSA4	-	ANSA2	ANSA1	ANSA0
	0	0	0	0	0	0	0	0
ANSELC	ANSC7	ANSC6	-	-	ANSC3	ANSC2	ANSC1	ANSC0
	0	0	0	0	0	0	0	0
WPUA	-	-	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
	0	0	1	1	1	0	1	1
WPUC	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0
	0	0	0	1	1	0	0	0
APFCON0	RXDTSEL	SDOSEL	SSSEL	—	T1GSEL	TXCKSEL	—	—
	0	0	0	0	0	0	0	0
APFCON1	—	—	—	—	P1DSEL	P1CSEL	P2BSEL	CCP2SEL
	0	0	0	0	0	0	0	0
SSP1STAT	SMP	CKE	D/A	P	S	R/W	UA	BF
	1	0	0	0	0	0	0	0
SSP1CON1	WCOL	SSPOV	SSPEN	CKP	SSPM<3:0>			
	0	0	1	0	1	0	0	0
SSP1CON3	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
	0	0	0	0	1	0	0	0
SSP1ADD	ADD7	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0
	0	0	0	0	0	1	0	0

Tabella 9 Configurazione dei bit nei registri per l'inizializzazione

E' poi possibile passare alla stesura del main, sotto riportato, e qui spiegato per punti:

1. Sono incluse delle librerie che richiamano funzioni di utilità generale per il linguaggio C, insieme alla libreria specifica del componente. Inoltre sono inclusi i file e le librerie generate da MCC. E' inoltre associata una parola all'indirizzo del sensore.
2. E' inizializzato il dispositivo e insieme ad esso gli interrupt.
3. Sono create/inizializzate alcune variabili, inoltre è portato ad un valore basso il segnale di DONE.
4. Il microcontrollore accende il sensore e concede il tempo a quest'ultimo di convertire il segnale.
5. Viene letto il valore di temperatura presente nell'apposito registro del sensore. E' anche eseguito un controllo e qualora la comunicazione non vada a buon fine, la si riavvia fino ad un massimo di dieci volte.
6. Si spegne il sensore.
7. Si scrive il dato corrispondente alla temperatura nei primi due byte disponibili della memoria EEPROM
8. Viene segnalata al timer la corretta ricezione del segnale di WAKE (anche al primo ciclo la segnalazione è necessaria sebbene non ci sia stato alcun interrupt).
9. Il microcontrollore entra in fase di sleep, in attesa di un interrupt esterno.

1	`#include <stdio.h>`
	`#include <stdlib.h>`
	`#include <pic16lf1824.h>`
	`#include "mcc_generated_files/mcc.h"`
	
	`#define TMP112_addr 0b1001000`

2	`void main(void) {`
	
	`    //Initialize the device`
	`    SYSTEM_Initialize();`
	`    INTERRUPT_GlobalInterruptEnable();`
	`    INTERRUPT_PeripheralInterruptEnable();`
	`    EXT_INT_InterruptEnable();`

3	`    unsigned char timeout;`
	`     // I2C initial status`
	`    I2C_MESSAGE_STATUS status = I2C_MESSAGE_COMPLETE;`
	`     //set initial eeprom address`
	`    uint8_t     eeprom_address = 0x00;`
	`     //a buffer where read data will be stored`
	`    uint8_t     Buffer[2];`
	`     //set DONE signal (RC5) low`
	`    DONE_SetLow();`

4	`    while(1){`
	`          //Put RC2 high to activate TMR112`
	`        LATCbits.LATC2 = 1;`
	`          //conversion time`
	`        __delay_ms(30);`

5	`        //This function reads a data temperature`
	`        //(2 bytes) from the TMP112.`
	`        //In case of mistakes it tries for 10 times`
	`        //and then goes out from the routine`
	
	`        timeout=0;`
	`        if (status == I2C_MESSAGE_COMPLETE){`
	`            while(timeout<10){`
	`                    //read 2 byte from slave`
	`                I2C_MasterRead(Buffer, 2, TMP112_addr,`
	`&status);`
	`                 //wait for the message to be sent`
	`                //or status has changed`

```
                while(status == I2C_MESSAGE_PENDING);
                //check if the message was
                //correctly sent
                if (status == I2C_MESSAGE_COMPLETE)
                    break;
                else
                    timeout++;
            }
        }
```

```
        //Put RC2 low to turn off TMR112
        LATCbits.LATC2 = 0;
```

```
        //This function writes temperature
        //value in eeprom memory

        //write MSB
        DATAEE_WriteByte(eeprom_address, Buffer[0]);
        if (eeprom_address<255)
            eeprom_address++;
        else
            eeprom_address=0x00;
          //write LSB
        DATAEE_WriteByte(eeprom_address, Buffer[1]);
        if (eeprom_address<255)
            eeprom_address++;
        else
            eeprom_address=0x00;
```

```
        //set DONE signal (RC5) high
        //in response to TPL5010 interrupt
        DONE_SetHigh();
          //set DONE signal (RC5) low
        __delay_ms(20);
        DONE_SetLow();
```

```
        //pic is in sleep mode till
        //external interrupt is detected
        SLEEP();

    }
     return;
}
```

Sotto è riportata la routine di interrupt che si limita ad azzerare l'omonimo flag per poi tornare all'istruzione successiva allo SLEEP.
In realtà non sarebbe stato necessario abilitare il Global Interrupt dato che non è indispensabile per uscire dalla condizione di standby e che comunque la routine di interrupt non svolge operazioni.
Tuttavia anche le librerie del protocollo I^2C fanno uso degli interrupt e quindi è stato obbligatorio abilitarlo.

```
void INT_ISR(void)
{
    EXT_INT_InterruptFlagClear();
    // Callback function gets called
    //everytime this ISR executes
    INT_CallBack();
}
```

4.3. Stima dei consumi e durata della batteria

Per stimare i consumi si è deciso di definire delle fasi di operatività, ciascuna caratterizzata da un proprio assorbimento di corrente:

- Avvio → si verifica solo all'accensione e coinvolge il solo timer, dal momento che avviene parallelamente alle altre fasi. Ha una durata di 100ms.
- Conversione → avviene ogni volta in cui il sensore converte un dato di temperatura. Ha una durata di 26ms.
- Trasmissione → si verifica durante la trasmissione seriale dei dati tra microcontrollore e sensore. La sua durata è dipendente dalla frequenza scelta e dal quantitativo di informazioni scambiate. Dal momento che la frequenza è stata selezionata pari a 1,55kHz e che in una comunicazione vengono scambiati 3 pacchetti da 9 bit (indirizzo+ 2 byte di informazione con il relativo ACK):

$$t_{trasmissione} = \frac{3 * 9}{1.55 * 10^3} = 17.419\ ms \sim 18\ ms$$

- Elaborazione → tempo in cui il microcontrollore è acceso senza che ci si trovi in una delle due fasi precedenti. La durata dev'essere stimata: dopo aver controllato il register memory del µC si suppongono 200 istruzioni macchina per implementare la routine associata, alla frequenza di 31kHz; cui occorre sommare 24 ms di ritardi imposti:

$$t_{elaborazione} = \frac{200}{31 * 10^3} + 24 = 30.45 \sim 31\, ms$$

- Standby → è la fase in cui il sensore è spento, il PIC è in sleep e il timer conta. Ha una durata di 1745s.

Segue una tabella in cui sono riepilogati gli apporti all'assorbimento di corrente di ciascun componente.

			Assorbimento di corrente			
Fase	**Durata**	**Ciclica**	**PIC16LF1824**	**TMP112**	**TPL5010**	**Totale**
Avvio (100ms)	100ms	No	-	-	200µA	200µA
Conversione	26ms	Sì	4µA	40µA	40nA	44.04 µA
Trasmissione	18ms	Sì	4µA	15µA	40nA	19.04 µA
Elaborazione	31ms	Sì	4µA	spento	40nA	4.04µA
Standby	1745s	Sì	30nA	spento	40nA	70nA

Tabella 10 Riepilogo dell'assorbimento di corrente di ciascun componente in relazione alla fase

E' ora possibile procedere alla stima dei consumi calcolando la corrente media in un periodo. Un periodo (T) è definito dalla somma delle durate di tutte le fasi ad eccezione dell'Avvio, dal momento che questa si verifica una sola volta.
Diventa necessario dunque capire se l'energia assorbita da questa fase sia rilevante o meno:

$$E_{avvio} = \frac{I_a t_a}{3600} = 5.55 nAh$$

Rispetto alla capacità della batteria (210mAh), l'assorbimento di tale energia è irrilevante e per questo sarà trascurato.

La corrente media risulta:

$$I_{medio} = \frac{I_c t_c + I_t t_t + I_e t_e + I_s t_s}{T} = 70.1\, nA$$

Si nota come l'assorbimento di corrente totale sia praticamente identico a quello in fase di standby. Questo è dovuto al fatto che la durata di questa fase è circa cinque ordini di grandezza maggiore rispetto alle altre.

Sapendo che la capacità di una batteria CR2032 è almeno C=210mAh, è possibile stimarne la durata come segue:

$$t = \frac{C}{I_{medio}} = 2960901{,}204h \sim 123371\ giorni \sim 338\ anni$$

La durata ottenuta è incredibilmente alta e non è realmente raggiungibile a causa del coefficiente di scarica annuo che risulta essere più influente dell'assorbimento dovuto al circuito. Inoltre, anche l'invecchiamento della batteria nel corso degli anni limita il tempo di vita del sistema. Allo stato attuale, i produttori di batterie possono garantire tempi di vita massimi di circa 10 anni.

Diventa dunque interessante provare ad ipotizzare l'utilizzo di una batteria con una minore capacità.
Scegliendo una CR1216, avente sempre una tensione nominale di 3V, ma una capacità minima di 25mAh, si ottiene t ~ 40 anni. Un risultato comunque molto elevato, superiore al tempo di vita stimato del sistema e della batteria stessa.

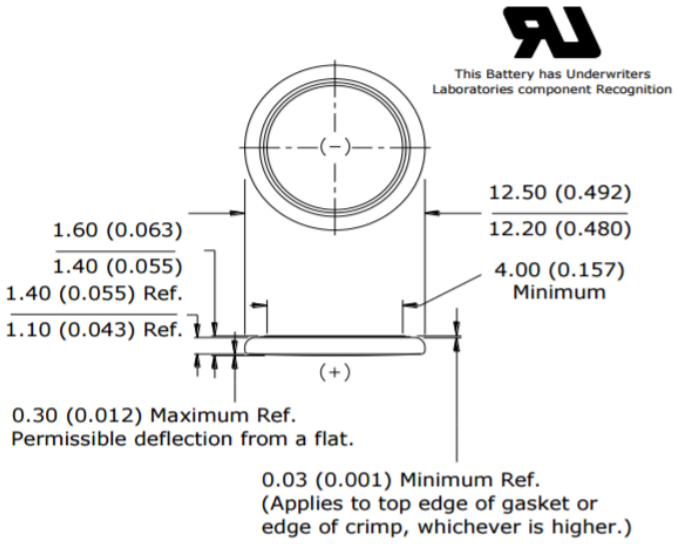

Figura 24 Batteria CR1216 **[10]**

Segue un grafico in cui sono riportati i tempi di vita delle due batterie prese in considerazione, in funzione di diverse durate della fase di standby.

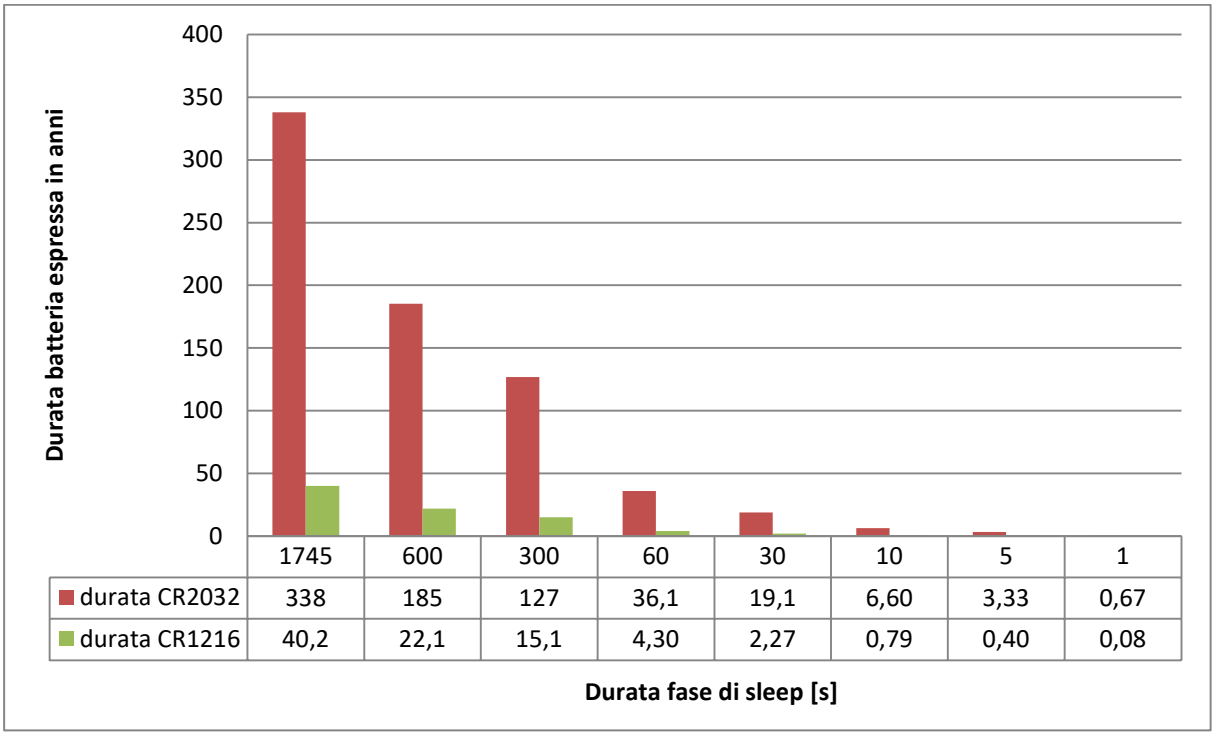

Figura 25 Durata delle batterie CR2032 e CR1216 in funzione del tempo di sleep

Dal grafico è facile vedere come il tempo di scarica della batteria dipenda fortemente della durata della fase di standby.
Riducendola infatti aumenta la frequenza con cui il PIC si sveglia e conseguentemente il consumo.

4.4. Progetto del PCB

Per la progettazione del PCB (Printed Circuit Board) è stato utilizzato “KiCad”, una software di EDA (Electronic Design Automation) open source. Al suo interno sono contenuti dei tool che consentono di gestire tutte le fasi del progetto.

4.4.1. EESchema

Con questa prima applicazione è stato possibile disegnare lo schematico riportato al paragrafo 4.1.

Il tool mette a disposizione delle librerie contenenti i simboli di numerosi componenti. Ciò nonostante si è reso necessario creare delle proprie librerie al fine di aggiungere alcuni simboli mancanti .

Disegnati i componenti questi sono stati opportunamente collegati mediante l'utilizzo di linee vere e proprie ma anche grazie allo strumento "etichette". Questo consente di connettere dei pin solamente associando loro lo stesso nome, così da rendere più leggibile lo schematico.

E' stato infine utilizzato uno strumento che consente di verificare che non siano violate regole elettriche.

4.4.2. CvPcb

CvPcb consente di associare ai componenti riportati nello schematico l'opportuna impronta, dipendente dal package. Una volta fatto questo è possibile generare una netlist contenente tutte le informazioni utili alla realizzazione del PCB.
Segue una tabella in cui sono riportati tutti i componenti presenti con i relativi package.

Componente	Nome	Valore	Package
C_1	Condensatore	100nF± 10%	SMD_0805
C_2	Condensatore	100nF± 10%	SMD_0805
C_3	Condensatore	10nF± 10%	SMD_0805
JP_1	Jumper	-	strip_straight_1x02_Pitch2.54mm
P_1	Connettore	-	strip_straight_1x04_Pitch2.54mm
P_2	Connettore	-	strip_straight_1x04_Pitch2.54mm
P_3	Connettore	-	strip_straight_1x04_Pitch2.54mm
P_4	Connettore	-	strip_angled_1x06_Pitch2.54mm
R_1	Resistore	91kΩ ±0.1%	SMD_0805
R_2	Resistore	0Ω	SMD_0805
R_3	Resistore	100kΩ ±1%	SMD_0805
R_4	Resistore	4.7kΩ ±1%	SMD_0805
R_5	Resistore	4.7kΩ ±1%	SMD_0805
R_6	Resistore	4.7kΩ ±1%	SMD_0805
U_1	CR2032	-	Socket_S8421-45R
U_2	Switch	-	strip_straight_1x03_Pitch2.54mm
U_3	TPL5010	-	SOT23(6 pin)
U_4	PIC16LF1824	-	TSSOP(14 pin)
U_5	TMP112	-	SOT563(6 pin)

Tabella 11 Elenco dei componenti con i rispettivi package

Anche in questo caso non tutte le impronte erano presenti e alcune sono state realizzate con l'apposito editor, seguendo le indicazioni riportate sui datasheet dei componenti. Nel realizzare i pad, oltre a tener presenti le effettive dimensioni dei pin, è stato lasciato sufficiente spazio per poter effettuare una saldatura a mano.
Segue un esempio di impronta realizzata a partire dalle geometrie dell'integrato.

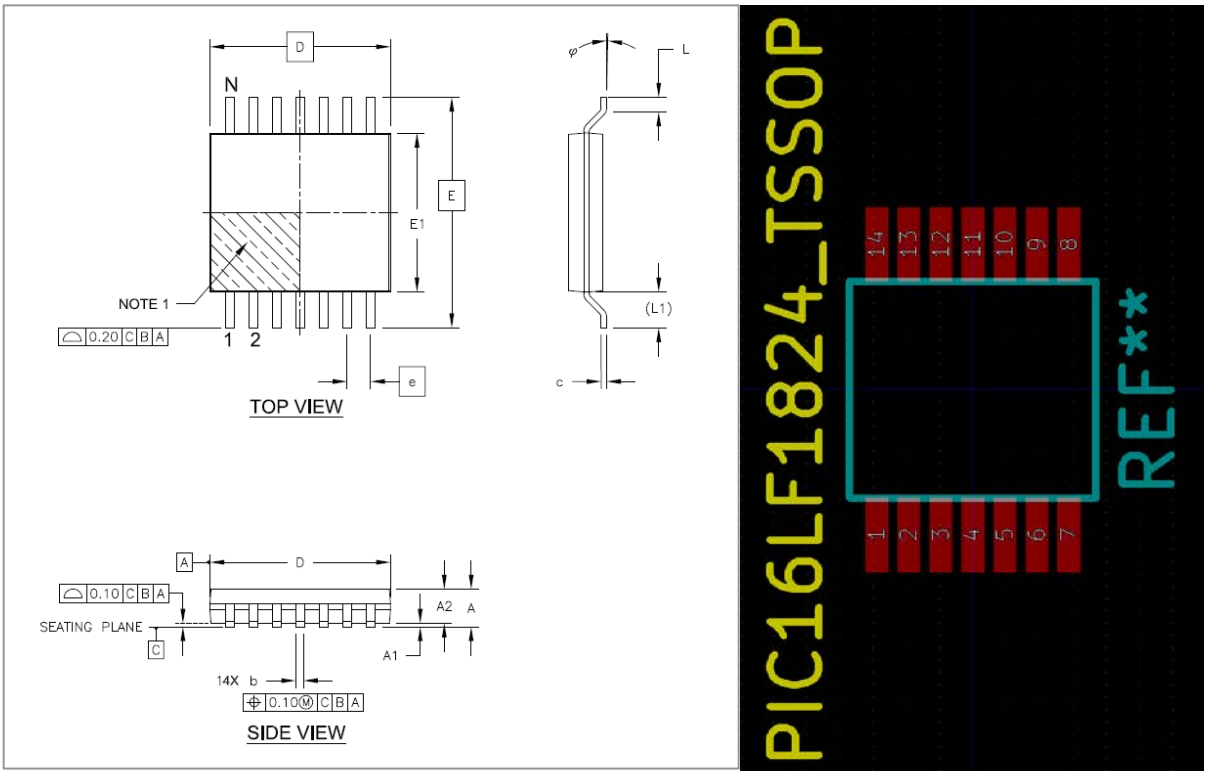

Figura 26 A sinistra: Geometrie del PIC16LF1824 con package TSSOP **[7]**; A destra: Impronta realizzata con KiCad

4.4.3. Pcbnew

Con Pcbnew si procede al disegno del circuito stampato vero e proprio. Importando la netlist precedentemente creata vengono in automatico inserite tutte le impronte dei componenti; inoltre i pin sono già opportunamente connessi da linee rette dette "rat's nests" o "airwires" che indicano quali siano i pin da connettere manualmente tra loro tracciando le piste.

Si è scelto di utilizzare una basetta a due strati dato che, non essendo elevato il numero dei componenti, non è stata ritenuta necessaria la presenza di una copiia di layer di alimentazione.

Le regole utilizzate nella disposizione dei componenti sono state:

- Posizionare i componenti attivi sulla faccia frontale e i passivi su quella posteriore, fatta eccezione per i condensatori di bypass, posizionati invece in prossimità degli integrati di riferimento
- Mettere i connettori sui bordi della scheda così da renderli accessibili dall'esterno
- Minimizzare la distanza dei componenti così da ridurre le dimensioni del PCB
- Orientare i componenti in maniera da evitare il più possibile l'incrocio delle rat's nest

Per quanto riguarda i collegamenti invece:

- Rispettare i vincoli tecnologici imposti tal produttore. Sono state scelte:
  - larghezza delle piste: 0.2mm;
  - isolamento minimo tra le piste: 0.2mm;
  - vias di diametro 0.6mm e foro di 0.2mm;
  - distanza dal piano di massa 0.2mm;
  - distanza del piano di massa e delle piste dai bordi: 0.4mm;
- Eseguire cambi di direzione nelle piste con due angoli di 45° anziché uno di 90° per ridurre le capacità parassite
- Ridurre l'area delle maglie di corrente, in particolare per l'alimentazione
- Evitare i loop di massa
- Disegnare dei powerplane su entrambe le facce, per ridurre l'effetto dei disturbi, cercando di evitare che siano interrotti, in modo che le correnti di ritorno abbiano percorsi il più possibile senza ostruzioni
- Posizionare dei vias per rendere il più possibili equipotenziali i piani di massa

Nelle figure seguenti sono mostrate le due facce del PCB.

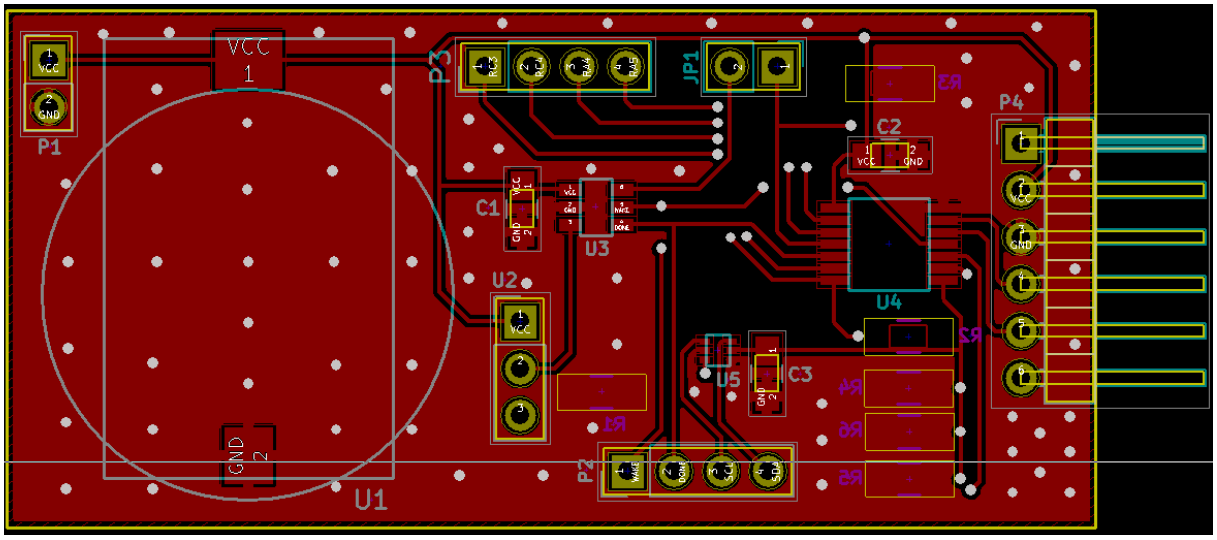

Figura 27 Front layer del PCB

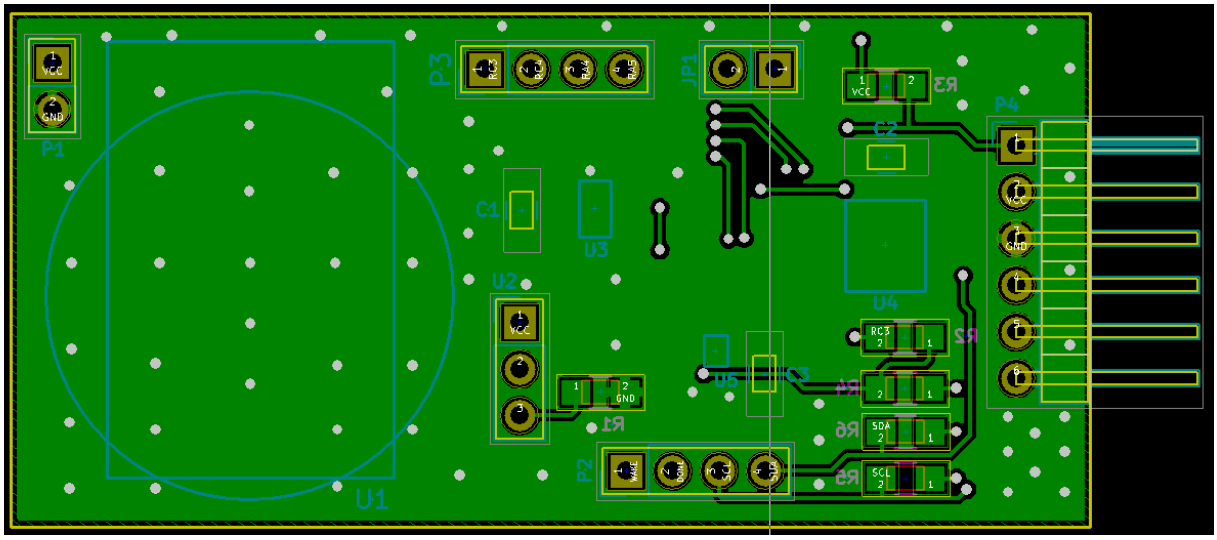

Figura 28 Bottom layer del PCB

4.5. Lettura dati dal datalogger

Nel progetto non è presente un sistema di lettura dei dati. E' evidente che questo risulta necessario qualora si scelga di andare a leggere i valori di temperatura salvati nella memoria EEPROM.
Per fare questo non risulta possibile l'utilizzo di tecniche di trasmissione a distanza basate sulla trasmissione attiva di onde radio. L'aggiunta di un modulo per implementare tale funzionalità infatti, comprometterebbe il basso consumo di corrente tanto ricercato.
Bisogna trovare un'altra modalità di comunicazione. Risultano praticabili tre possibilità:

- **Trasmissione seriale mediante conduttore**.
  La prima possibilità è quella di connettersi alla porta seriale del microcontrollore mediante conduttori metallici, collegando all'altra estremità un dispositivo di acquisizione dati.

- **Modulo NFC**
  NFC (Near Field Communication) è una tecnologia che consente una comunicazione bidirezionale tra due dispositivi a distanza non superiore ai 10 cm. Collegando un "tag" NFC alla porta seriale, risulta possibile intraprendere una comunicazione con un dispositivo che dev'essere in grado a sua volta di gestire questa tecnologia. L'aspetto interessante è che il tag non necessita di essere alimentato, poiché preleva l'energia direttamente dal lettore, non andando a compromettere i consumi del datalogger.
  Nel tag è infatti presente un avvolgimento elicoidale, nel quale viene indotta dal campo elettromagnetico generato dal lettore, una tensione in grado di alimentare il chip di cui dispone.
  La comunicazione avviene mediante l'uso della tecnologia di backscattering e cioè tramite la modulazione dell'impedenza dell'avvolgimento, che avviene tramite l'apertura e la chiusura di un interruttore. Questo causa una riflessione dell'onda trasmessa che può essere rilevata dal lettore stesso.

- **Modulo RFID**
  RFID (Radio-Frequency IDentification) è una tecnologia concettualmente simile alla precedente, in cui sono presenti dei chip transreciver che si

autoalimentano. Anche in questo caso si fa uso di tecniche di back-scattering.
Si differenzia però dall'NFC poiché basata su far-field communication ovvero su una comunicazione di "campo lontano". Ad esempio l'RFID di seconda generazione, basato su UHF (Ultra High Frequency), supporta comunicazioni ad una distanza massima di 50m.
Si differenzia dalla precedente anche per l'utilizzo di antenne al posto degli avvolgimenti.

5. Realizzazione e test del PCB

5.1. Montaggio dei componenti

- Ultimata la progettazione del PCB, i file gerber generati da KiCad, sono stati inviati ad un'azienda per produrre il circuito stampato.
- I componenti, preventivamente ordinati, sono poi stati saldati manualmente sul PCB, con l'ausilio di un microscopio, viste le ridotte dimensioni degli integrati.
- E' stato caricato il firmware sul microcontrollore mediante l'uso del programmatore PicKit 3.

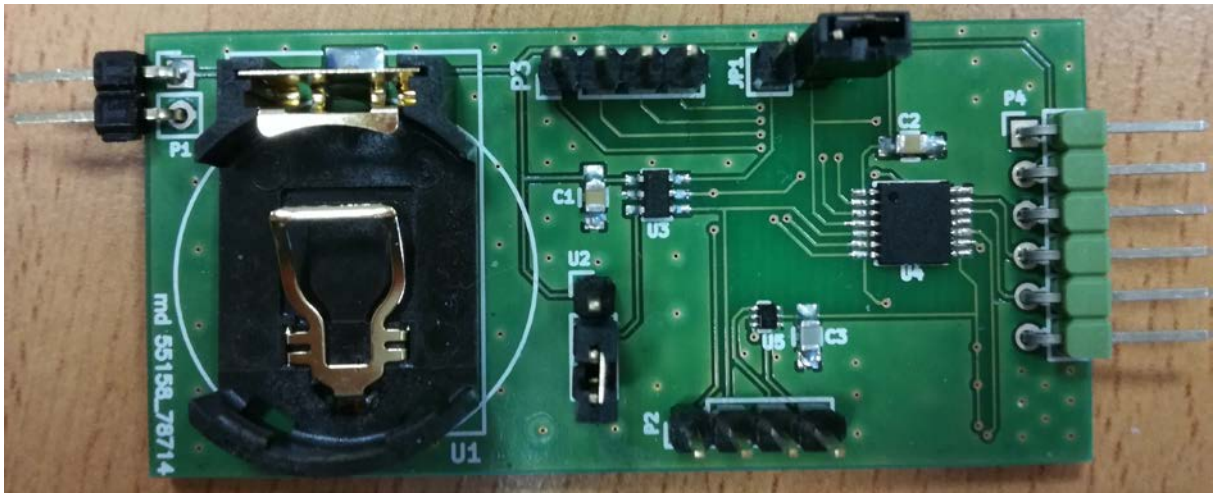

Figura 29 Faccia frontale della scheda

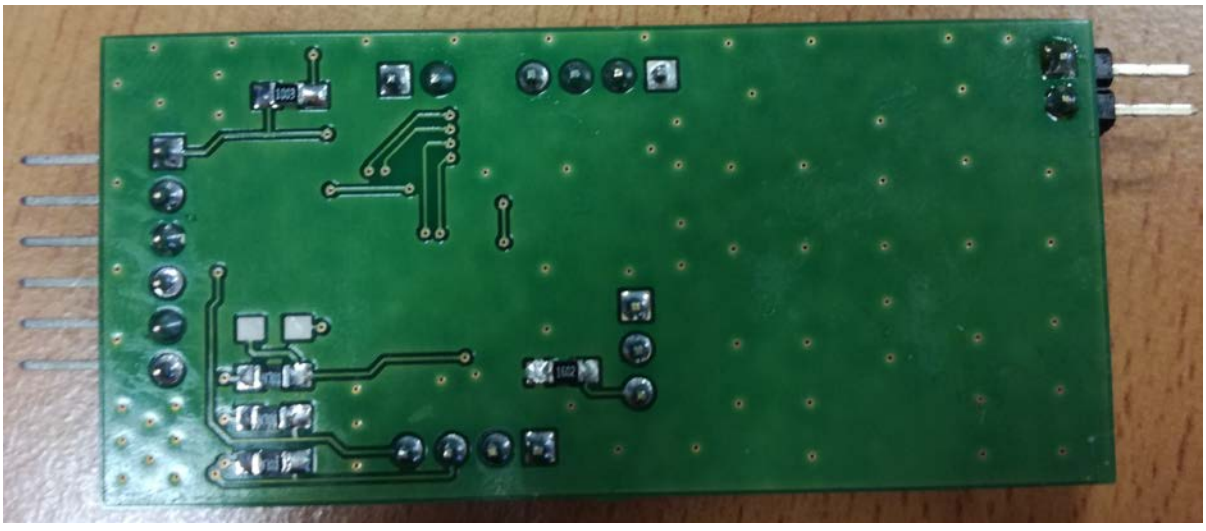

Figura 30 Faccia posteriore della scheda

5.2. Collaudo

E' stato poi collaudato il circuito. Questa fase è suddivisibile in due passaggi:

- Test delle funzionalità
- Verifica dei consumi

Per agevolare la procedura di collaudo, in questa fase si è deciso di sostituire la resistenza R_1 da 91kΩ, con una da 16kΩ, garantendo una temporizzazione del TPL5010 di circa 30 secondi, per l'esattezza:

$$T = a * {R_1}^2 * 10^{-6} + b * R_1 * 10^{-4} + c * 10^{-2} \cong 26.5\ s$$

Con *a*, *b*, *c* forniti dal datasheet.

In questa maniera è stato possibile ridurre il tempo di sleep tra una fase attiva e l'altra, consentendo di eseguire le misure in tempi più brevi.

5.2.1. Test delle funzionalità

Il circuito è stato alimentato con una sorgente esterna dotata di limitazione di corrente, per evitare, al primo avvio, possibili sovracorrenti dovute ad errori nel montaggio. Una volta scongiurata questa eventualità, sono stati collegati i 4 canali di un oscilloscopio ai pin del connettore P_2: WAKE, DONE, SCL e SDA.
In questo modo è stato possibile verificare il corretto comportamento dei detti segnali.
I campioni così ottenuti dall'oscilloscopio sono poi stati importati in Matlab per essere ripuliti dal rumore e graficati con una maggiore risoluzione.

Nel grafico sottostante è mostrata una trasmissione completa.

- In seguito alla ricezione di un segnale di WAKE il microcontrollore attiva i pullup delle linee dell'I^2C che vengono portate pertanto a livello alto.
- Inizia poi la comunicazione seriale costituita da un segnale di inizio, uno di fine e dalla trasmissione delle informazioni secondo il protocollo già visto al paragrafo 3.1.1
- Al termine della trasmissione vengono disattivati i pullup
- Alla fine il microcontrollore invia un segnale di DONE

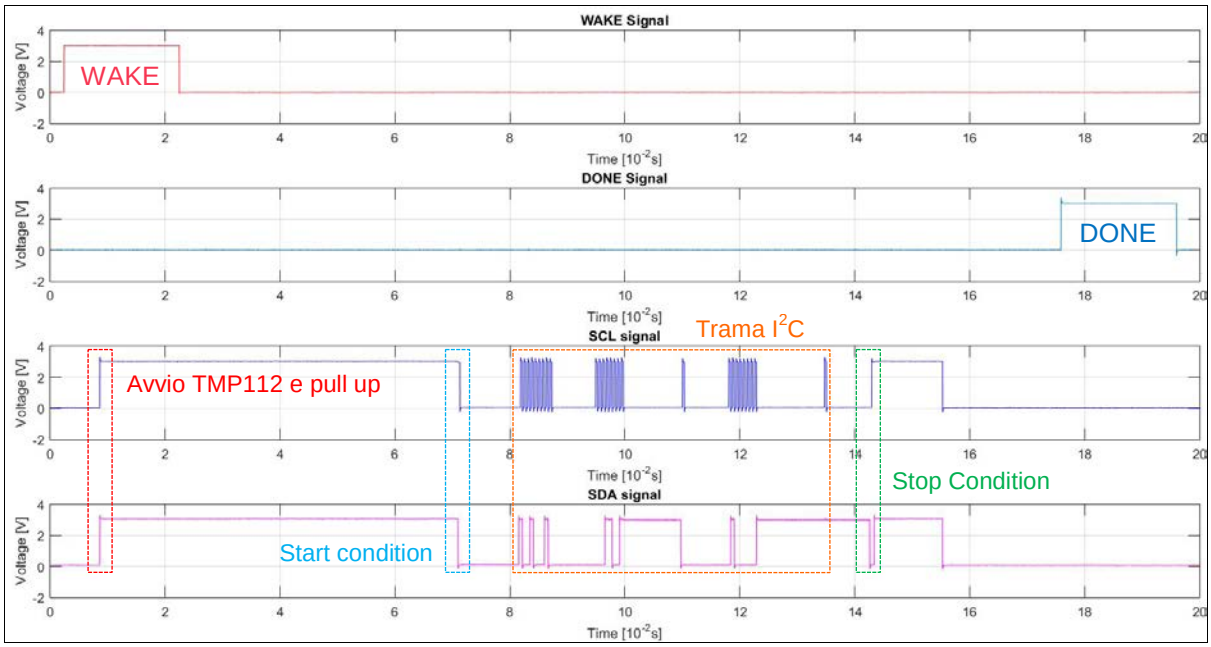

Figura 31 Vista dei 4 Canali dell'oscilloscopio

Segue uno zoom delle condizioni di start e stop che risultano essere in linea con quanto indicato in letteratura:

- START: mentre SCL è alto, transizione alto-basso di SDA
- STOP: mentre SCL è alto, transizione basso-alto di SDA

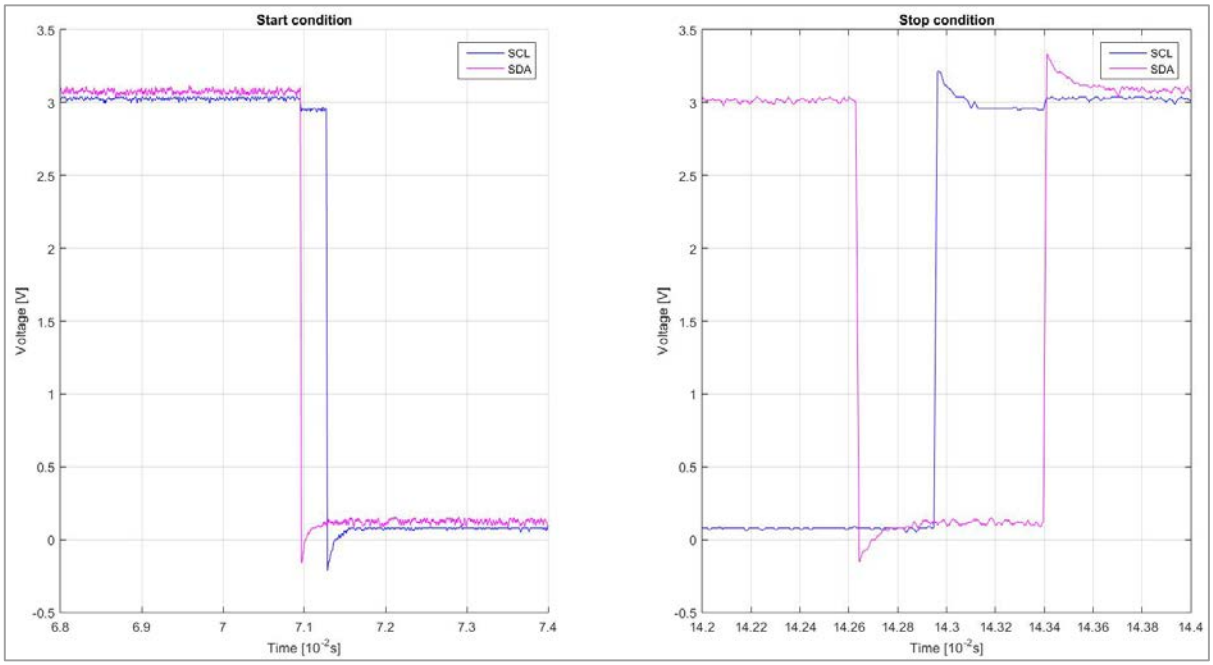

Figura 32 Zoom sulle condizioni di start e di stop

Risulta interessante anche analizzare la trama dei messaggi, che come visibile, è costituita da 3 pacchetti. Questo risultato risulta conforme alle aspettative, dal momento che erano attesi l'invio dell'indirizzo e di due byte contenenti i dati di temperatura.

Nel grafico sottostante è analizzato il primo pacchetto, formato dalla seguente sequenza: 100100010.
I primi sette byte corrispondono all'indirizzo dello slave, l'ottavo, posto a 1, indica che il successivo pacchetto corrisponderà ad una lettura ed il nono indica la corretta ricezione da parte dello slave.

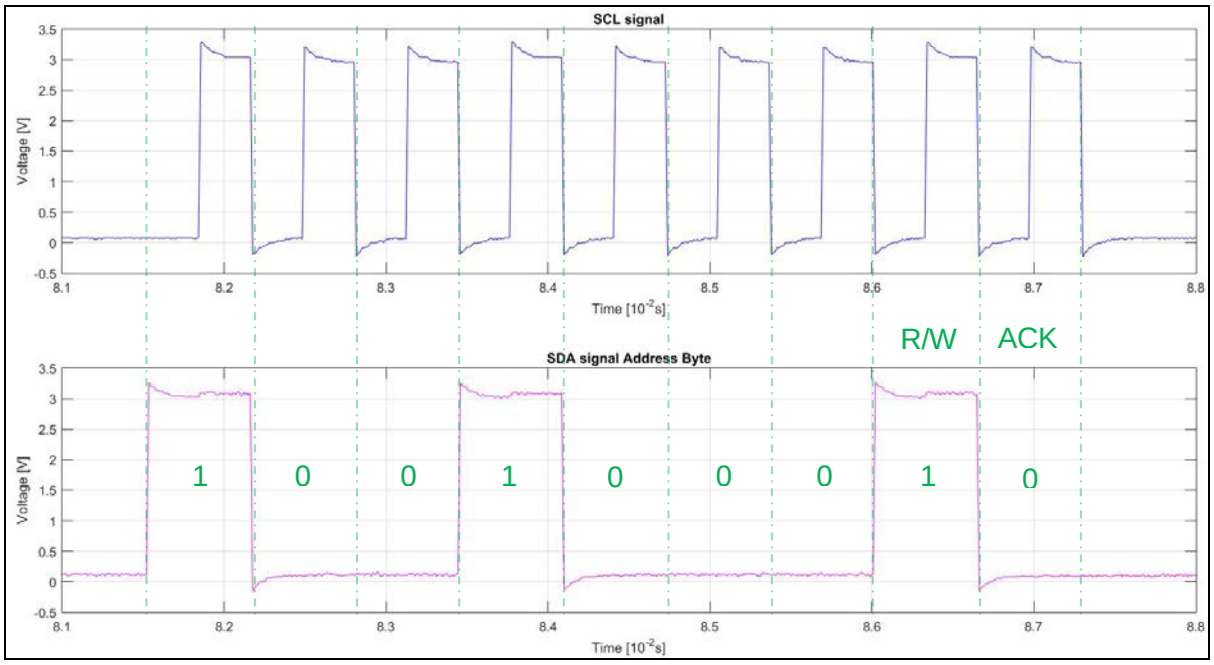

Figura 33 Trasmissione dell'indirizzo dello slave

Tutti i quattro segnali ed in generale la trasmissione, sono dunque conformi alle aspettative.

Per completare l'analisi funzionale è ancora necessario vedere se il payload e dunque i due byte successivi, corrispondono effettivamente ad un valore coerente alla temperatura della stanza in cui è posizionato il circuito.

Per questa seconda prova si è scelto di ricollegare il circuito al programmatore, in modo da poter controllare la memoria EEPROM attraverso la funzionalità "Memory View" disponibile su Mplab.
In questo modo, è stato possibile verificare anche se il microcontrollore riesce a scrivere su detta memoria, completando così l'analisi funzionale.

Nella figura sottostante, estratta appunto dall'ambiente MPLAB, è mostrato lo stato della EEPROM dopo che è trascorso un tempo sufficiente a riempirla completamente.
Essendoci 256 celle ed occupando ciascuna lettura due di queste, con una lettura circa ogni 26.5s, si ha:

$$t_{riempimento} = \frac{256}{2} * 26.5 \cong 3392s \sim 56\ minuti$$

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	18 (1)	10	17	F0	17	F0	17 (2)	E0	17	E0	17	E0	18	40	18	40
10	18	40	18	40	18	30	18	20	18	20	18	20	18	10	18	10
20	18	10	18	10	18	10	18	10	18	10	18	00	18	00	17	F0
30	18	00	18	00	18	00	19	60	19	60	19	60	19	50	19	50
40	19	50	19	40	19	30	19	30	19	30	19	30	19	30	19	30
50	19	30	19	30	19	30	19	40	19	30	19	30	19	20	19	20
60	19	20	19	30	19	30	19	30	19	30	19	20	19	30	19	20
70	19	20	19	20	19	20	19	20	19	10	19 (3)	00	19	00	18	F0
80	18	E0	18	E0	18 (4)	E0	18	E0	18	E0	18	F0	18	E0	18	D0
90	18	D0	18	D0	18	D0	18	D0	18	D0	18	D0	18	D0	18	C0
A0	18	C0	18	C0	18	C0	18	C0	18	D0	18	C0	18	C0	18	D0
B0	18	D0	18	C0	18	C0	18	C0	18	C0	18	C0	18	B0	18	B0
C0	18	B0	18	A0	18	A0	18	A0	18	A0	18	90	18	90	18	90
D0	18	90	18	90	18	80	18	80	18	80	18	80	18	70	18	70
E0	18	70	18	70	18	70	18 (5)	60	18	60	18	60	18	60	18	60
F0	18	60	18	60	18	60	18	60	18	60	18	60	18	60	18	60

Figura 34 Memoria EEPROM PIC16LF1824 vista da MPLAB

Trasformando il contenuto di una coppia di celle dal valore esadecimale al corrispondente binario ed eliminando le quattro cifre meno significative , si riottiene il dato a 12 bit generato dal sensore che può essere opportunamente convertito come visto al paragrafo 3.3.1.
Segue la conversione dei 5 dati evidenziati in tabella.

1. $0x1810 = 0b0001100000010000 \rightarrow 0b0b000110000001 = 385$
   $385 * 0.0625 = 24.06°C$

2. $0x17E0 \rightarrow 23.88°C$

3. $0x18F0 \rightarrow 24.94°\ C$

4. $0x18D0 \rightarrow 24.81°\ C$

5. $0x1860 \rightarrow 24.38°\ C$

Questi valori, verificati con un termometro a contatto con il package del sensore, differiscono al massimo di 0.5°C dall'effettivo valore di temperatura.
Le leggere oscillazioni sono dovute a variazioni seppur minime della temperatura ambiente. E' da notare che la differenza tra due valori vicini è sempre inferiore all'accuratezza del sensore che è pari a 0.5°C.

E' quindi ragionevole sostenere che il circuito svolge correttamente tutte le funzioni previste.

5.2.2. Verifica dei consumi

Per misurare i consumi è stata utilizzata una scheda della STMicroelectronics chiamata Power Monitoring, in grado di interfacciarsi mediante porta USB con un computer, in modo da graficare la corrente istantanea assorbita dal circuito che vi è connesso.

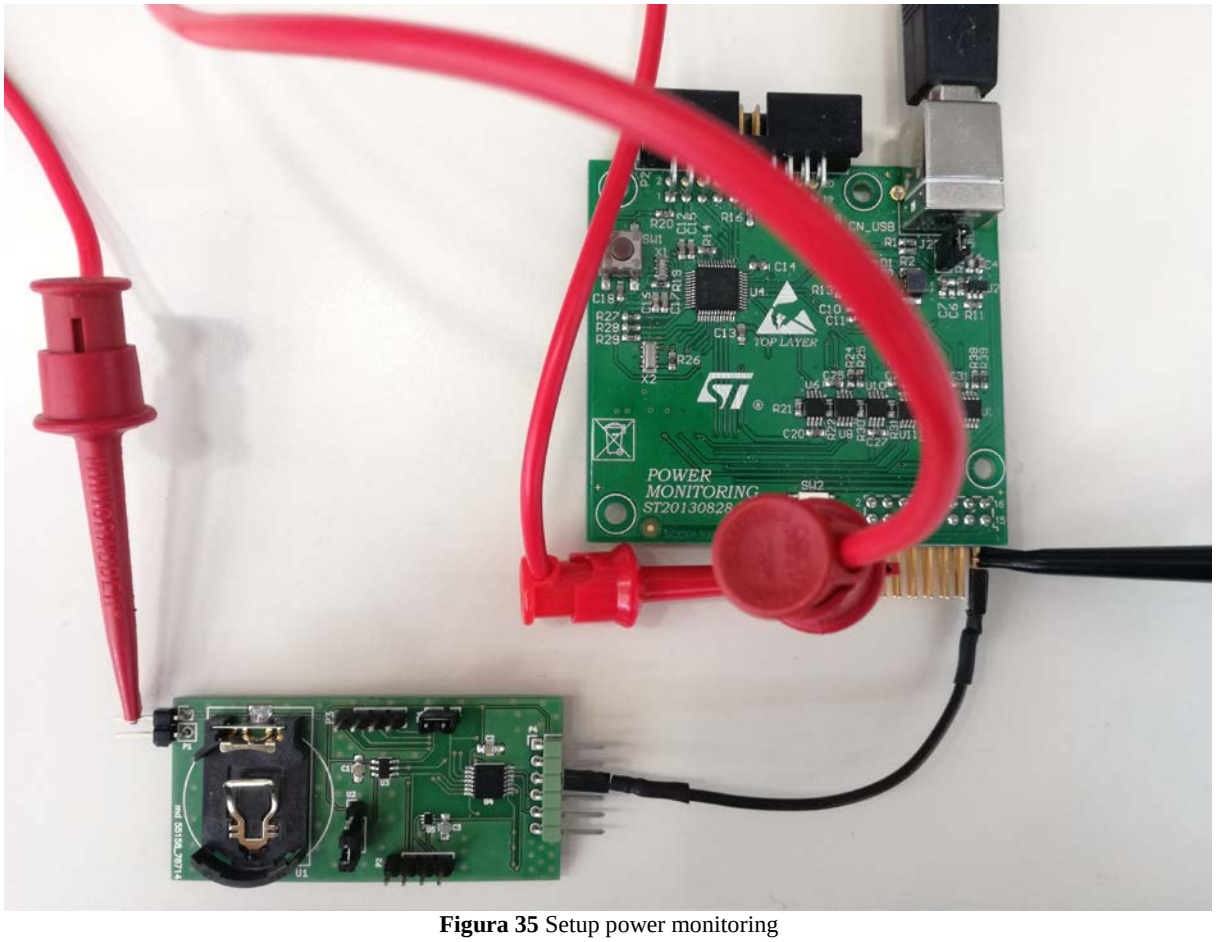

Figura 35 Setup power monitoring

Segue una descrizione dei consumi osservati

- Durante la fase di sleep una misura della corrente non risulta possibile a causa della non sufficiente precisione dello strumento: a vuoto è mostrata una corrente, dovuta al rumore, che oscilla attorno al µA, valore ben più alto della corrente che si vorrebbe osservare.
  Il valore visualizzato (attorno appunto ad 1µA) consente comunque di distinguere questa fase dalle altre.
- E' presente un picco di corrente che sta ad indicare il risveglio del microcontrollore, al quale segue una fase di elaborazione con un consumo di 6 µA.
- Un ulteriore picco, più alto del precedente, indica l'accensione del sensore. Questo è seguito da una fase di durata 26ms in cui il consumo è di 50 µA; tale fase corrisponde alla conversione della temperatura da parte dell'ADC del sensore.
- Segue un ulteriore fase di elaborazione in cui il consumo, dato che il sensore rimane acceso, sale a 8 µA.
  In questa fase (non mostrata interamente nei grafici visto il consumo pressochè costante) è inclusa anche la trasmissione, con un consumo comunque pari a 8 µA. Per ottenere questo risulato è stato però necessario aumenare notevolmente i valori delle reistenze di pullup delle linee di trasmissione, passando da 4.7kΩ a 1MΩ. In questa maniera è stato infatti quasi annullato l'assorbimento di corrente dovuto alla loro presenza. E' da sottolineare che questa sostituzione non ha comportato cambiamenti in termini di funzionalità.

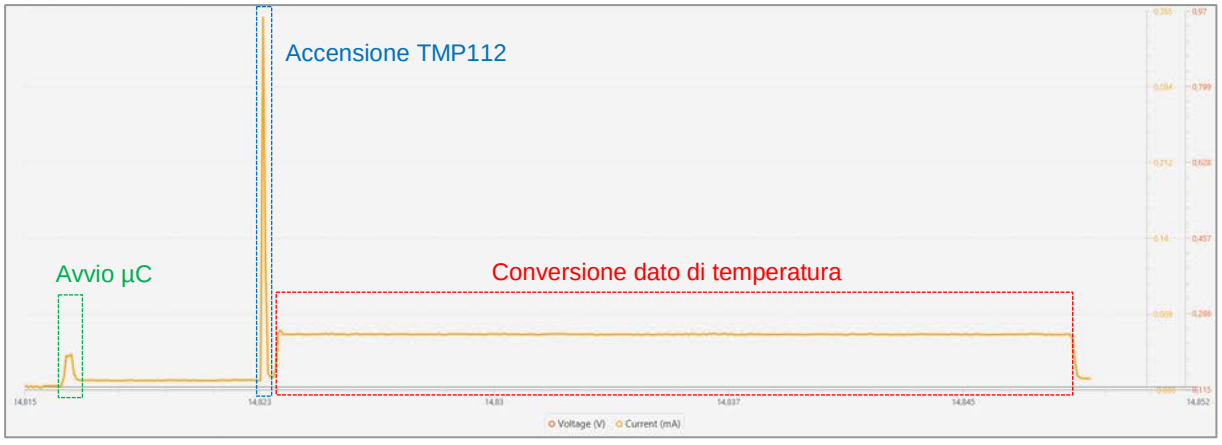

Figura 36 Assorbimento di corrente avvio, accensione TMP112 e conversione

- In seguito alla trasmissione il TMP112 viene spento ed il consumo torna ad attestarsi sui 6 µA.

- Un consumo di corrente che non era stato considerato in fase di progetto, poiché non indicato nel datasheet del microcontrollore, è quello legato alla scrittura sulla EEPROM. Dato che questa operazione comporta il cambio di stato di alcune celle di memoria non-volatile, è inevitabile che ci sia un consumo aggiuntivo di corrente.
  Prima di scrivere un nuovo dato in EEPROM è sempre necessario cancellare il precedente (operazione erase-write); questo è ben visibile dal grafico dei consumi dal momento che i due assorbimenti di corrente risultano ben distinti.
  Quest'operazione, di durata complessiva 3µs e consumo 90µA per la cancellazione e 100µA per la scrittura, è ripetuta due volte, per consentire l'inserimento del MSB e del LSB.
- Il consumo diventa poi nuovamente pari a 6 µA per circa 20ms (tempo imposto da un delay), dopodiché il microcontrollore tornerà in sleep, con una corrente non apprezzabile.

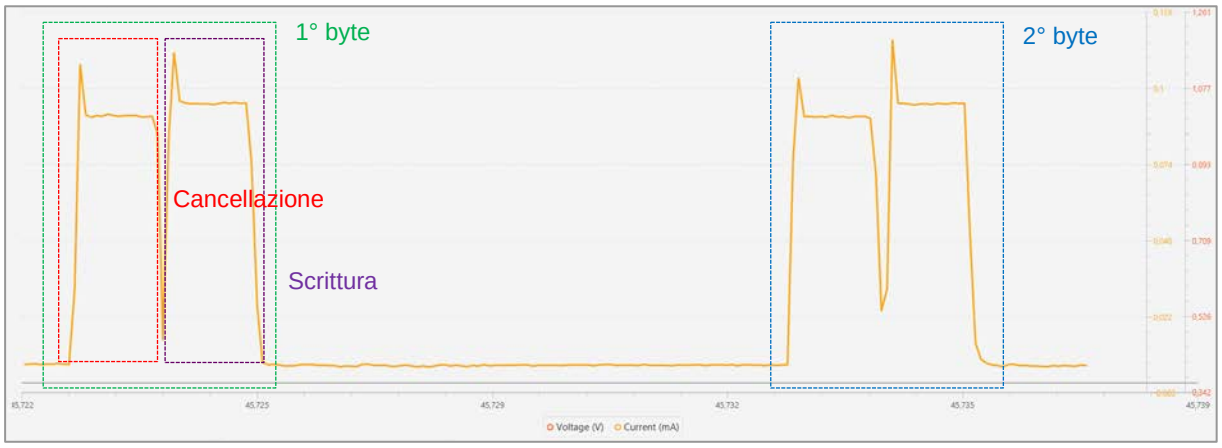

Figura 37 Assorbimento di corrente cancellazione-scrittura EEPROM

I consumi osservati sperimentalmente, risultano essere molto simili a quelli stimati per via teorica, risultando in alcuni casi perfino più bassi del previsto.
Si osserva però che la fase attiva dura complessivamente circa 200ms, un tempo superiore al doppio di quello previsto.

La corrente di standby è stata invece misurata grazie all'ausilio di un multimetro da banco a 6 cifre e ½, l'Agilent 34401A. Per misurare con precisione una corrente dell'ordine delle decine di nA, sarebbe stato necessario uno strumento ancora più preciso, non disponibile però in laboratorio.
Ciò nonostante, impostando il multimetro in modalità "slow", aumentando cioè il tempo di campionamento, è stato possibile apprezzare che l'assorbimento di corrente è attorno a i 60nA.

Segue una tabella con un confronto tra l'assorbimento teorico e quello sperimentale di ciascuna fase.

	Valori teorici		Valori sperimenali	
Fase	**Durata**	**Assorbimento**	**Durata**	**Assorbimento**
Conversione	26ms	44.04 µA	26ms	50 µA
Trasmissione	18ms	19.04 µA	65ms	8 µA
Elaborazione	31ms	4.04 µA	100ms	8 µA
Scrittura EEPROM	Non prevista		6 ms	95 µA
Standby	1745 s	70nA	1745 s	60nA

Tabella 12 Confronto tra i consumi teorici e quelli sperimentali

Conclusioni

I risultati mostrano che l'obiettivo della tesi è stato raggiunto. E' infatti stato realizzato un nodo sensore capace di funzionare, in fase di standby, con correnti dell'ordine di decine di nA.
Sì è visto anche come, scegliendo opportune temporizzazioni, i consumi della fase attiva, essendo concentrati in un breve lasso di tempo, siano poco influenti ai fini del calcolo del consumo medio.

E' stato mostrato anche come tale circuito sia alimentabile con una batteria di dimensioni contenute, garantendo una durata teorica di decine o centinaia di anni a seconda delle soluzioni. Ne consegue dunque che il limite tecnologico della batteria, costituito dal coefficiente di scarica annuo, sia più influente dell'assorbimento di corrente dovuto al funzionamento del circuito.
Sarebbe possibile dunque aggiungere funzionalità al circuito, magari inserendo altri trasduttori in grado di convertire segnali di varia natura, mantenendo comunque durate teoriche superiori alla vita di stoccaggio della batteria.

Predisponendo un sistema di lettura come quelli menzionati, un dispositivo di questo tipo potrebbe trovare applicazione in campi quali il monitoraggio ambientale in zone remote, dove risulta necessario avere dispositivi autonomi per lungo tempo.

E' inoltre stato un interessante caso di studio per poter valutare quale sia lo stato dell'arte nell'ambito della dissipazione di potenza nei moderni dispositivi a microcontrollore.

Bibliografia

[1] J. M. Rabay, A. Chandrakasan e B. Nikolic, CIRCUITI INTEGRATI DIGITALI, Milano: Pearson Education Italia, 2005.

[2] «Wikipedia,» [Online]. Available: https://en.wikipedia.org/wiki/Internet_of_things.

[3] P. Panda e t.al., «Basic Low Power Digital Design,» in *Power-efficient System Design*, Springer, 2010.

[4] N. Kim, T. Austin e D. Baauw, «Leakage Current:Moore's law meets static power».

[5] I. Energizer Holdings, *Product Datasheet: Energizer CR2032.*

[6] Microchip, *Datasheet AN1333.*

[7] Microchip, *Datasheet:PIC16(L)F1824/1828.*

[8] Texas Instruments, *Datasheet: TPL5010.*

[9] Texas Instruments, *Datasheet: TMP112.*

[10] Energizer, *Product datasheet: CR1216.*