UNIVERSITA' DEGLI STUDI DI FIRENZE

FACOLTA' DI INGEGNERIA

Tesi di Laurea di Primo Livello in Ingegneria Elettronica

PROGETTO E REALIZZAZIONE DI UN COLLEGAMENTO WIRELESS IN BANDA ISM 868 MHZ

Relatore: Prof. Massimiliano Pieraccini

Correlatore : Ing. Giovanni Macaluso

Autore : Laurent Ntibarikure

ANNO ACCADEMICO 2006-2007

SOMMARIO

INTRODUZIONE

Le tecnologie elettroniche odierne offrono la possibilità di realizzare sistemi di telecomunicazioni radiomobili molto efficienti ad un costo relativamente basso. Questi ultimi sfruttano la propagazione di onde elettromagnetiche nello spazio libero per la trasmissione di segnali informativi, distaccandosi dai vincoli fisici imposti ad una comunicazione via cavo. La crescente richiesta del mercato ha portato le industrie del settore ad offrire ai progettisti una vasta gamma di soluzioni sempre migliori. Durante il mio tirocinio presso il Laboratorio di Tecnologie per i Beni Culturali[1], ho avuto l'occasione di studiare alcuni di questi prodotti.

Lo scopo di questa tesi è di illustrare i dispositivi scelti, elencandone le caratteristiche principali, e, riferendosi alle specifiche di progetto, mostrare come essi vengono impiegati nella realizzazione di un collegamento wireless. Quest'ultimo è destinato a sostituire la comunicazione cablata tra due parti di un sistema radar a penetrazione superficiale per la rilevazione di segni vitali.

In una prima parte di questa tesi sarà presentato il sistema radar, con particolare riguardo all'interfaccia di comunicazione già presente. Successivamente verrà spiegato il funzionamento dei componenti utilizzati nella realizzazione del collegamento. L'intera progettazione verrà ripercorsa cronologicamente, dall'analisi della situazione attuale alla realizzazione vera e propria. Infine verrà mostrato il prototipo e commentate le prove eseguite su di esso.

---

[1] Università degli Studi di Firenze, Dipartimento di Elettronica e Telecomunicazioni.

Alla mia famiglia che mi ha sempre supportato durante questi primi anni di università.

Un sentito ringraziamento al professor Pieraccini per aver reso possibile quest'esperienza indimenticabile.

Ai ragazzi del laboratorio, per il loro simpatico ed amichevole sostegno.

CAPITOLO 1 :
LE SPECIFICHE DI PROGETTO

In questo capitolo verrà presentato il sistema nel quale il collegamento wireless si colloca. Si tratta di un radar a penetrazione superficiale GPR[1] per la rilevazione di segni vitali, in grado di individuare eventuali superstiti da catastrofi naturali sepolti sotto le macerie.

1.1 Lo stato attuale del sistema radar

Integrandosi in un sistema radar mobile, il progetto da realizzare si propone di sostituire con un collegamento wireless il cavo di connessione tra l'unità radar e lo schermo di controllo. Per poter rilevare segni vitali sotto diversi metri di macerie, il radar sfrutta la sua grande sensibilità alle vibrazioni. L'operatore deve dunque allontanarsi dal radar, evitando così di influenzare negativamente la rilevazione. Infatti, le macerie, per la loro conformazione irregolare, riflettono in tutte le direzioni le onde generate dal radar. L'operatore rifletterebbe a sua volta queste onde verso l'antenna del radar, provocando un disturbo alla rilevazione. Le indagini devono essere ripetute in più posti e il cavo rende quindi poco pratica quest'operazione. Con un collegamento wireless si isola il radar dall'influenza dell'operatore, offrendo una maggiore maneggevolezza al sistema.

La Figura 1.1 illustra lo stato attuale del sistema radar, nel quale permane il problema della rilevazione legata alla vicinanza dell'operatore.

[1] *Ground Penetrating Radar*

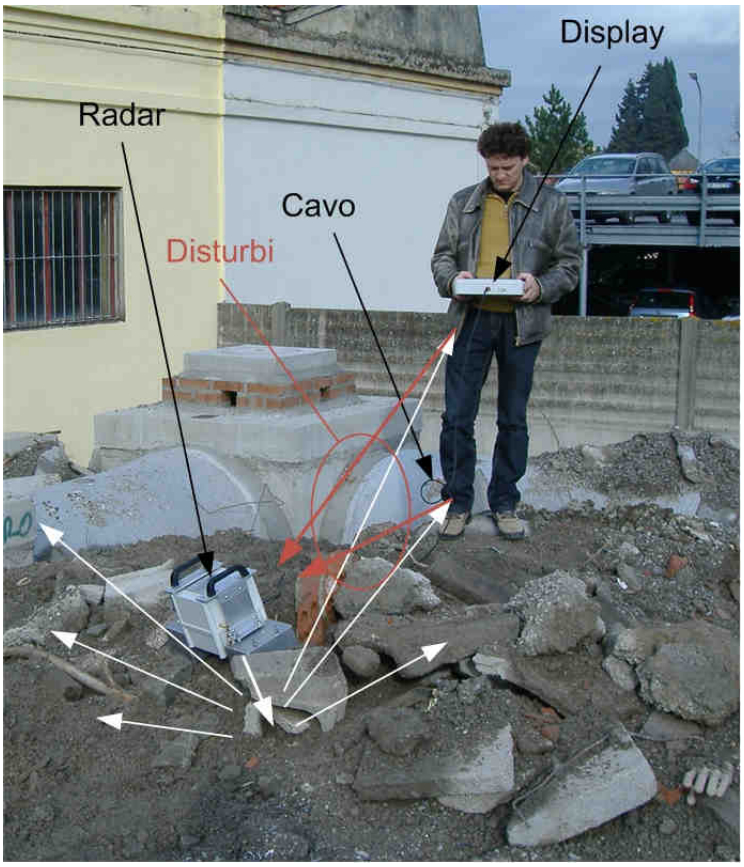


Figura 1.1. Disturbi legati alla presenza dell'operatore

Le due parti del sistema comunicano attraverso interfacce seriali RS232.

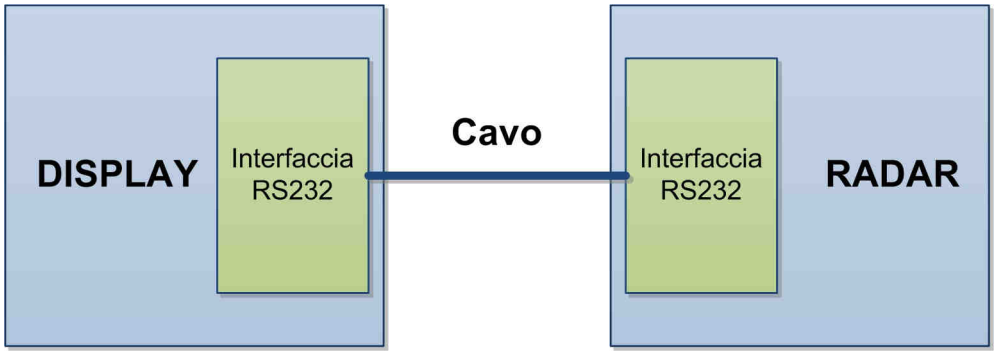


Figura 1.2. Sistema Radar nello stato attuale

L'interfaccia RS232

Le prime specifiche RS232 furono rilasciate nel 1962 dall'EIA, *l'Electronic Industries Alliance* attraverso lo standard EIA 232. Col tempo quest'ultimo si è evoluto, integrando specifiche della TIA, *Telecommunication Industries Association*, e concludendosi con l'EIA/TIA-232-F dell'ottobre 1997. Lo standard nacque

coll'intento di definire il livello fisico della pila protocollare ISO/OSI[2] per la comunicazione tra un DTE, *Data Terminal Equipment* ed un DCE, *Data Comunication Equipment*, permettendo ad un sistema computerizzato, il DTE, di comunicare con altri attraverso il DCE, dispositivo esclusivamente dedicato alla ricetrasmissione di informazioni.

L'RS232 specifica sia le caratteristiche elettriche che meccaniche e funzionali del livello fisico. I segnali, in codifica NRZ[3], sono sbilanciati e unidirezionali (su una linea si hanno soltanto trasmettitore e ricevitore). I livelli di tensione sono simmetrici rispetto a massa: una tensione negativa rappresenta il livello logico alto e, viceversa se positiva, quello basso.

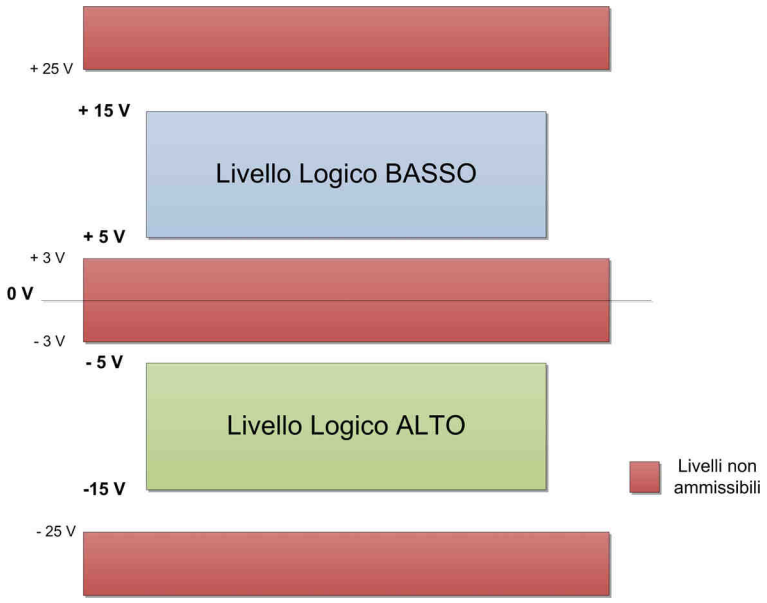


Figura 1.3. Livelli di tensione RS232

Il *bit rate* standard può arrivare a 20 Kbps qualora il cavo lo permetta : lo standard non ne definisce una lunghezza massima ma le capacità ed impedenza di linea massime, rispettivamente 2500 pF e 7 KΩ, che con i comuni cavi utilizzati corrispondono a poco meno di 20 metri. Con alcuni artifici, ad esempio con l'impiego di cavi particolari e dispositivi di pilotaggio particolarmente veloci, la soglia del Megabit al secondo viene facilmente raggiunta, pur mantenendo gli stessi livelli di tensione. Nel nostro caso il *bit rate* è impostato a 38,4 Kbps.

---

[2] International Organization for Standardization / Open Systems Interconnection
[3] Non Return to Zero

Al livello di collegamento (ISO/OSI) viene spesso utilizzata la seguente configurazione: i dati vengono raggruppati in pacchetti di 10 bits permettendo una comunicazione asincrona a trame o *frames*: un byte viene confinato entro bit d'inizio e bit di arresto trasmissione che assicurano un "sincronismo di *frame*".

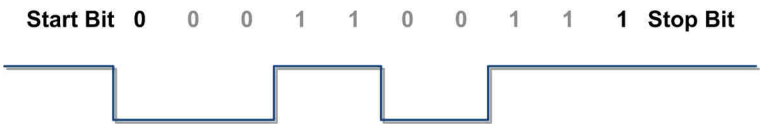


Figura 1.4. *Frame* a 8 bits, 1bit di stop, nessun bit di parità

Questa soluzione viene comunemente usata per la trasmissione di caratteri in codifica ASCII[4]: infatti con la precedente sequenza di bit in Figura 1.4, ossia il *byte* 0x33 (esadecimale) viene trasmesso il carattere "3". La struttura del frame può essere modificata in funzione delle necessità. Per esempio, un ulteriore "bit di parità" permette di ottenere un semplice controllo di integrità dei dati trasmessi. Si può inoltre allungare la durata del bit d'arresto qualora la linea sia particolarmente rumorosa.

Lo standard specifica anche il *pin-out* di un connettore tipo D a 25 linee (DB-25) delle quali 18 sono di segnale. La seguente Tabella 1-1 elenca le linee e loro funzioni di un connettore DB-25 per RS232.

[4] American Standard Code for Information Interchange

# PIN	Nome	Funzione	# PIN	Nome	Funzione
1	---	Protective Ground	14	SBA	2nd Transmitted Data
2	**TXD**	**Transmitted Data**	15	DB	DCE Element Timing
3	**RXD**	**Received Data**	16	SBB	2nd Received Data
4	RTS	Request To Send	17	DD	Received Elem. Timing
5	CTS	Clear To Send	18	---	Unassigned
6	DSR	Data Set Ready	19	SCA	2nd Request To Send
7	**GND**	**Sign. Ground/Common**	20	DTR	Data Terminal Ready
8	CD	Carrier Detect	21	CG	Sign. Quality Detector
9	---	+Voltage	22	RI	Ring Detector
10	---	-Voltage	23	CH/CI	Data Rate Detector
11	---	Unassigned	24	DA	DTE Element Timing
12	SCF	2nd Line Detector	25	---	Unassigned
13	SCB	2nd Clear To Send			

Tabella 1-1. *Pin-out* DB-25 standard

Qualora non siano necessarie tutte le linee per stabilire la comunicazione tra il DTE e il DCE, si può usare il connettore DB-9 ossia quello della “porta seriale” del PC oppure l’RJ-45, entrambi decisamente meno ingombranti.

Sia il radar che l’unità di controllo impiegano le sole linee TXD e RXD su connettori DB-9 :

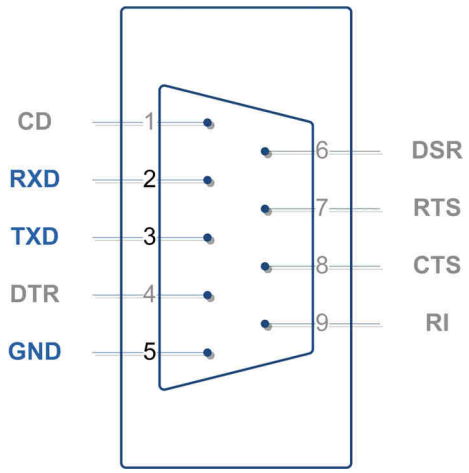


Figura 6. Connettore DB-9 RS232

Per quanto riguarda il controllo del flusso di dati via etere è da escludere quello di tipo hardware che impiega le linee RTS e CTS, rispettivamente “richiesta di trasmissione” (da parte del DTE) e “pronto per trasmettere” (in risposta da parte del

DCE). Un problema che potrebbe insorgere da una tale modifica del mezzo fisico di trasmissione è il verificarsi di *bottlenecks*, ovvero colli di bottiglia da entrambe le parti del sistema, in particolare da quella del radar che trasmette ininterrottamente dati per la visualizzazione su schermo. In effetti, qualora il flusso in ricetrasmissione radio dovesse risultare minore di quello cablato, la trasparenza della comunicazione verrebbe inevitabilmente compromessa.

Appena messo in funzione, il radar inizia a trasmettere dati grezzi circa ogni 5 ms, dati che vengono poi mediati e processati dal display per la loro visualizzazione in tempo reale. Questi dati sono impacchettati in modo particolare: un *token*, il byte 0xAA, avverte che i 4 *bytes* successivi contengono informazioni sulle componenti in fase e in quadratura del segnale radar retrodiffuso rilevato dal sistema.

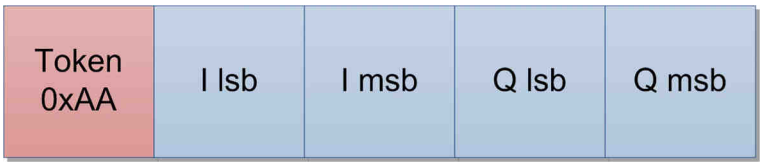


Figura 1.5. Pacchetti provenienti dal radar

I comandi dal display vengono inviati dopo l'intervento dell'operatore, oppure automaticamente qualora il display non sia in grado di visualizzare i dati ricevuti (comandi di *reset*). In effetti, la ricezione di un dato che per certi motivi supera i limiti visualizzabili dallo schermo, dovuto per esempio ad un forte guadagno nella catena di recezione del sistema radar, viene immediatamente notificata al radar con un particolare comando. In questo caso i comandi lunghi solamente un byte, vengono impacchettati diversamente: 3 tipi di *token* (0x63, 0x67 o 0x69) possono precedere il *byte* di comando.

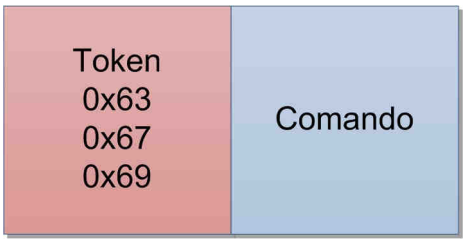


Figura 1.6. Comandi dal display

1.2 Schema del collegamento wireless

La soluzione che verrà presentata comprende la realizzazione di due moduli a radiofrequenza da interfacciare ad entrambe le parti del sistema, garantendone una comunicazione trasparente. Il radar, essendo ad onda continua con una frequenza operativa di 2,42 GHz, potrebbe essere influenzato dalla comunicazione qualora si decidesse di impegnare la medesima banda. Si è dunque deciso di operare in banda a 868 MHz, anche essa dedicata ad applicazioni industriali, scientifiche e biomedicali (ISM) per *Short Range Devices*[5] (SRD).

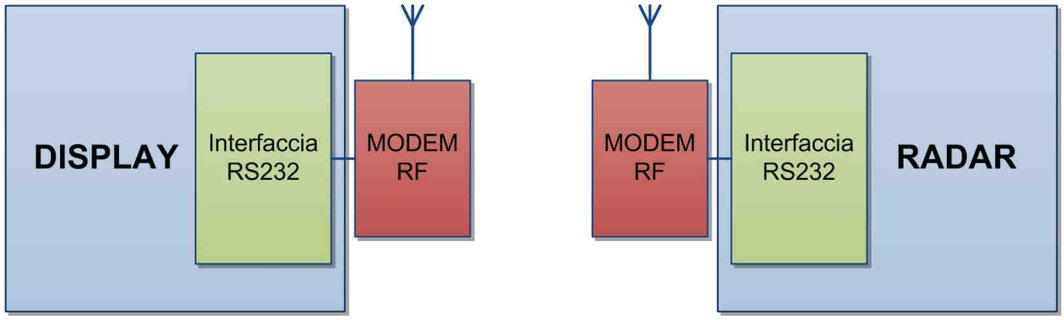


Figura 1.7. Sistema Radar con collegamento wireless

La figure Figura 1.7 e Figura 1.8 illustrano i due moduli di ricetrasmissione da realizzare. Questi moduli, che chiameremo "Modem RF[6]", costituiscono i *front-ends* in una doppia catena di modulazione e demodulazione, sia per i dati provenienti dal radar che per segnali di comando inviati dal display di controllo.

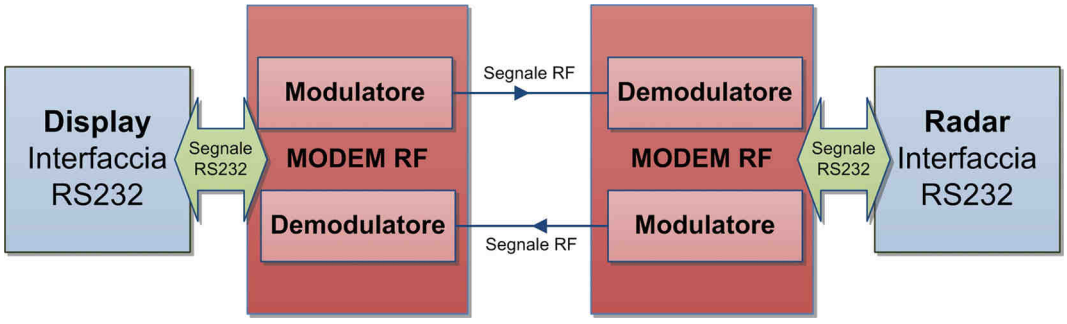


Figura 1.8. Schema del collegamento wireless

[5] "Dispositivi a corto raggio d'azione". Con questa classificazione si intendono apparati radiotrasmittenti non suscettibili di causare disturbi ad altri apparati radio
[6] Modem a RadioFrequenze

Si deve quindi realizzare un sistema di telecomunicazione in grado di riprodurre fedelmente da una parte all'altra del sistema le informazioni che normalmente fluirebbero attraverso il cavo elettrico.

La banda ISM 868MHz

Le bande ISM sono bande libere, che non necessitano di licenze per il loro utilizzo purché vengano rispettati certi requisiti. Sul territorio europeo, l'impiego di apparati radio viene regolamentato dalla direttiva EMC 2004/108/EC sulla compatibilità elettromagnetica e dalla norma armonizzata ETSI[7] EN 300 220. Quest'ultima riprende la norma CEPT[8]/ERC[9]/70-03 che definisce le modalità d'impiego di 13 bande disposte su uno spettro segmentato che va da 6765KHz a 246GHz. I requisiti vengono definiti, in funzione del tipo di apparato ricetrasmittente, in termini di potenza effettiva irradiata o *effective radiated power* (ERP), di frequenza di trasmissione con rispettivo *duty cycle* e di spaziatura tra canali in una medesima banda.

La banda a 868 MHz copre le frequenze da 863 a 870 MHz ed è ulteriormente segmentata in funzione del tipo di applicazione: allarmi, microfoni wireless ed altre apparecchiature audio, sistemi di identificazione a radiofrequenze (RFID) e SRD generici. Il nostro collegamento wireless è classificato come SRD generico. Pertanto la porzione di spettro che ci interessa è di soli 2 MHz, da 868 a 870 MHz. Particolari accorgimenti sarebbero necessari qualora si decidesse di impegnare il segmento di banda superiore: la banda 870-876 MHz è assegnata ad apparecchiature TETRA[10] che potrebbero desensibilizzare e bloccare un segnale in 869,3–870 MHz non adeguatamente filtrato.

[7] European Telecommunications Standards Institute
[8] European Conference of Postal and Telecommunications Administrations
[9] European Radiocommunications Committee
[10] TErrestrial Trunked RAdio

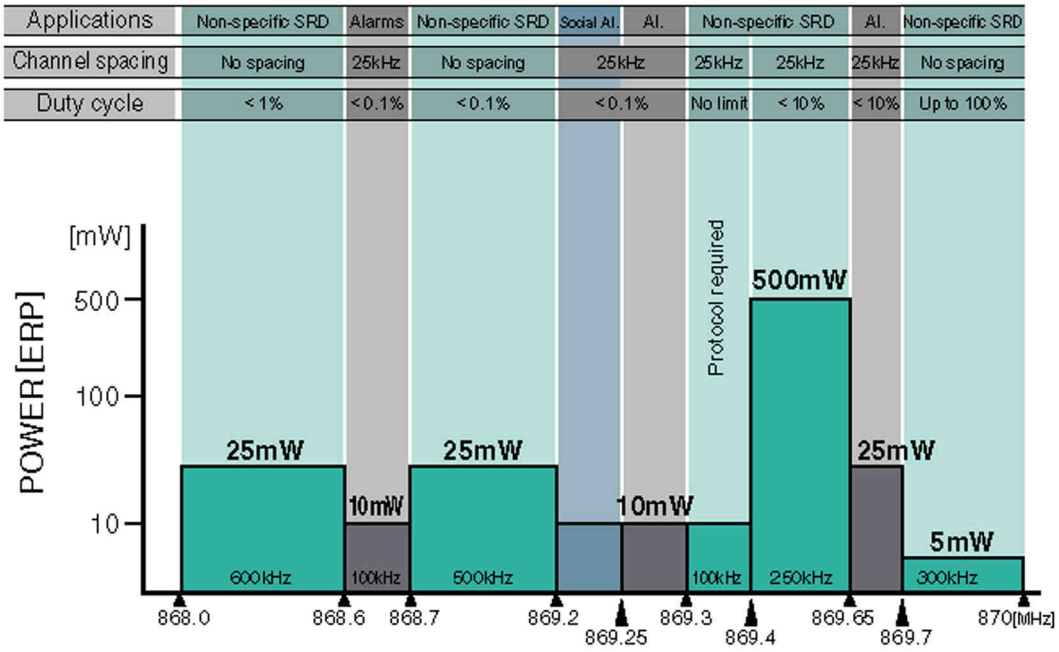


Figura 1.9. Limiti di potenza irradiata nelle bande per SRD generici

CAPITOLO 2 :

IL MODEM A RADIOFREQUENZE

In questo capitolo verrà trattata la fase di progettazione del collegamento wireless. In un primo momento si esporranno i componenti scelti , spiegandone il funzionamento. In seguito verrà mostrato il loro impiego nella realizzazione del Modem RF.

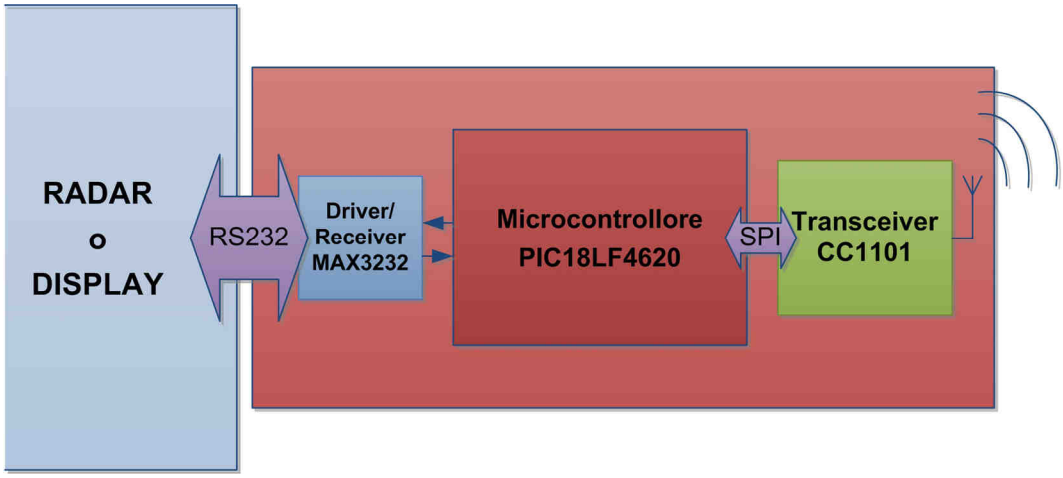


Figura 2.1. Schema a blocchi del Modem RF

La Figura 2.1 illustra lo schema a blocchi del Modem RF, introducendo i componenti scelti. Il *transceiver* CC1101, blocco di ricetrasmissione radio, si interfaccia ad un microcontrollore, il PIC18LF4620, che dovrà controllare il transceiver ed allo stesso tempo comunicare attraverso l'RS232 con il radar o il display. La conversione dei livelli di tensione, da quelli CMOS/TTL del microcontrollore a quelli dell'RS232, verrà affidata al *Driver/Receiver* MAX3232.

2.1 Componenti scelti

La prima fase del progetto richiede un'accurata analisi delle soluzioni offerte dal mercato. La semplicità d'impiego, le dimensioni fisiche, la qualità della documentazione, il supporto che la casa produttrice offre in termini di kit di

sviluppo e di software di progettazione, le prestazioni, sono alcuni dei criteri di scelta da prendere in considerazione.

Sfogliando un catalogo e navigando in rete alla ricerca della soluzione RF ideale, si nota subito che pochi sono i produttori che offrono soluzioni personalizzabili e che non siano prodotti finiti. La *Texas Instruments*, da molti anni leader nel settore dei semiconduttori, mette a disposizione una serie di integrati ricetrasmittenti o *transceivers RF*, la *CC11xx*[1]. Questi chips sono caratterizzati da dimensioni estremamente ridotte (package QLP) e necessitano di pochi componenti esterni per funzionare. Sono inoltre in grado di modulare e demodulare segnali digitali nelle più recenti ed efficienti forme di modulazione: OOK (*On-Off Keying*), simile all'ASK (*Amplitude Shift Keying*) e con ulteriore risparmio energetico, l'FSK (*Frequency Shift Keying*) e la GFSK (*Gaussian FSK*) in cui viene posto l'accento sull'efficienza spettrale ed infine l'MSK, una forma di modulazione O-QPSK (*Offset Quadrature Phase Shift Keying*), che permette di raggiungere *bit rates* molto elevati.

2.1.1 Il transceiver CC1101

Il *transceiver RF* scelto è il CC1101, progettato per lavorare come modulatore e demodulatore nelle bande ISM al disotto di 1 GHz, cioè quelle a 315, 433, 868 e 915 MHz. Le sue caratteristiche principali sono:

- la possibilità di selezionare *data rates* via etere da 1,2 a 500 Kbps nelle varie modulazioni elencate all'inizio del capitolo,
- un'ottima selettività di banda che riduce fortemente il rischio di bloccaggio dovuto a trasmissioni su canali adiacenti,
- un'altissima sensibilità in ricezione, con conseguenza diretta sul *range* di copertura del collegamento wireless,
- un rapido passaggio da trasmissione a ricezione e viceversa grazie al breve tempo di assestamento del sintetizzatore di frequenza

[1] Prodotti della Chipcon, compagnia specializzata in sistemi RF, acquisita dalla Texas Instruments

- infine e non meno importante, un'interfaccia di controllo digitale flessibile, che permette di configurare la radio *ad hoc* ed ottenere informazioni sullo stato del collegamento.

Le ultime due caratteristiche risultano cruciali nella gestione di un terminale *half-duplex* come il CC1101, che deve interfacciarsi al'RS232 di per sé *full-duplex*. Infatti, il transceiver non è in grado di trasmettere e ricevere contemporaneamente. Al contrario, le linee RXD e TXD sono completamente indipendenti l'una dall'altra.

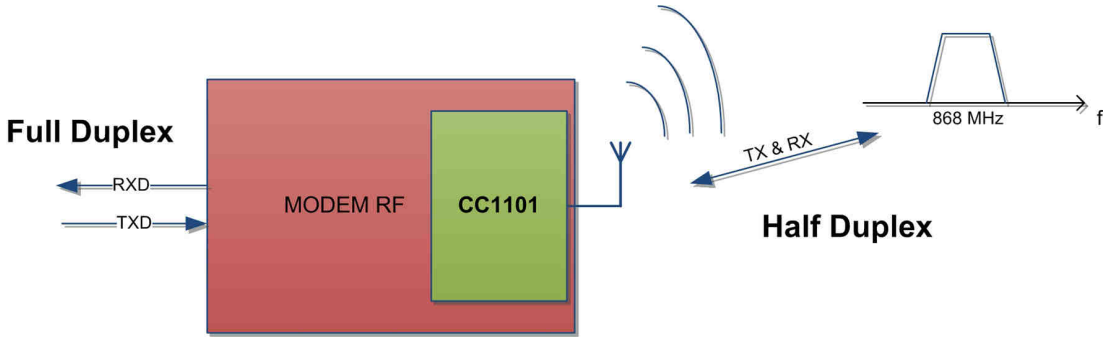


Figura 2.2. Da *Full-Duplex* ad *Half-Duplex*

2.1.1.1 Caratteristiche elettriche

Il transceiver CC1101 viene alimentato con tensioni da 1,8 a 3,6 V ed è in grado di irradiare potenze da -30 a +10dBm. La potenza trasmessa rispetta quindi le norme EMC. Alimentato a 3V ed impostato in modo da trasmettere alla massima potenza (10mW) a 868 MHz, il CC1101 assorbe una corrente di 32 mA, l'equivalente di circa 100mW. In condizioni normali d'impiego e ad 1mW di potenza trasmessa, negli stati attivi (trasmissione o ricezione) la corrente media assorbita è di circa 15 mA.

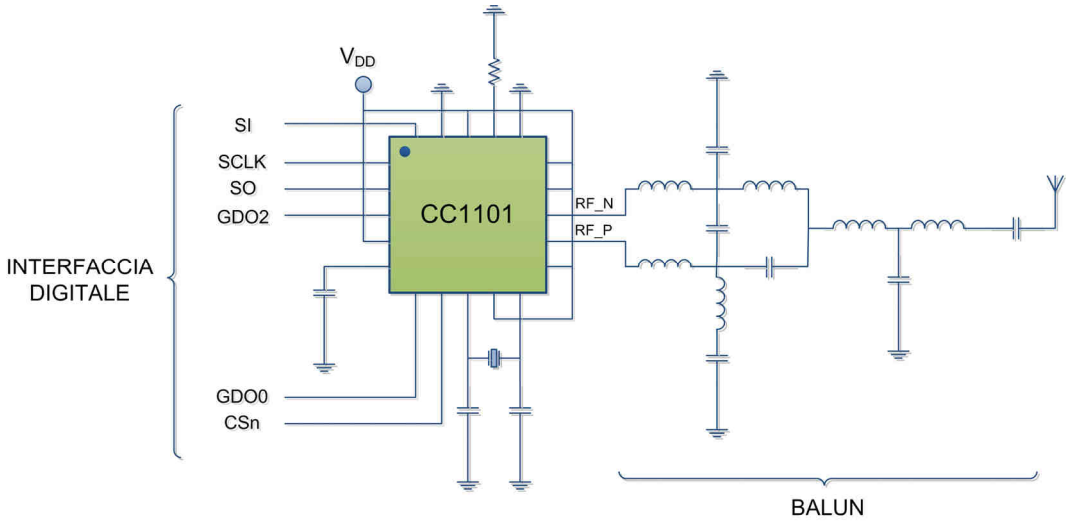


Figura 2.3. Esempio circuitale del CC1101

In Figura 2.3 viene illustrato il circuito di base per il funzionamento del transceiver a 868 MHz. Per disaccoppiare la parte analogica da quella digitale, l'alimentazione deve essere accuratamente filtrata attraverso una serie di capacità di valori e tipi diversi. Un *balun*[2] risulta necessario per collegare al chip un'antenna a quarto d'onda. In effetti, l'uscita bilanciata RF_N / RF_P deve essere sbilanciata per poter alimentare questo tipo di antenne. Il quarzo con frequenza di risonanza compresa tra 26 e 27 MHz non deve superare derive complessive (tolleranza del cristallo e del carico capacitivo, invecchiamento e derive termiche) superiori a 40 ppm, cioè di circa ±1 KHz.

2.1.1.2 Funzionamento

Vediamo adesso il funzionamento del CC1101, analizzando separatamente la parte analogica e quella digitale. Il processo CMOS a 0,18 µm col quale viene realizzato il chip ha permesso di integrare circuiti analogici e complessi blocchi digitali per segnali misti. La Figura 2.4 illustra lo schema a blocchi semplificato del transceiver.

[2] BALunced/UNbalanced

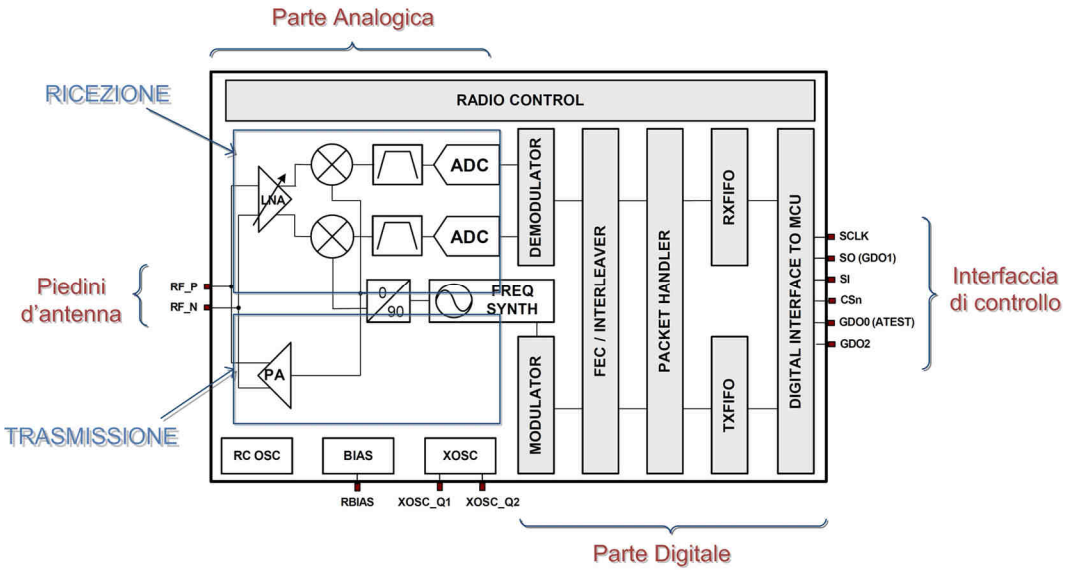


Figura 2.4. Schema a blocchi semplificato del CC1101

Due piedini del CC1101, RF_P e RF_N permettono la connessione all'antenna attraverso una linea bilanciata. La catena di ricezione inizia con un *Low Noise Amplifier* a guadagno variabile, il cui scopo primario è di ridurre al minimo la cifra di rumore dell'intera catena. Seguono poi due *mixer* che riportano in ingresso a due convertitori analogico-digitale le componenti in fase e in quadratura del segnale *down-converted*, opportunamente filtrate in banda intermedia. Questi ADC sono di architettura Δ-Σ (*"delta-sigma"*) e vengono fatti operare in *oversampling* del segnale. A questo punto la rilevazione del segnale quantizzato viene lasciata ad un blocco di demodulazione digitale. Per quanto riguarda la catena di trasmissione, un modulatore digitale gestisce direttamente il sintetizzatore di frequenza per generare forme d'onda a radiofrequenza che verranno successivamente amplificate nel *Power Amplifier* e quindi rimandate sui piedini d'antenna.

Essendo digitale, il demodulatore rende possibile una lunga serie di configurazioni tra le quali l'impostazione del filtro digitale, la compensazione dell'offset di battimento e la stima della potenza di segnale ricevuta. Il sincronismo di bit avviene per meccanismi di estrazione del *clock* (*clock recovery*) da parole di sincronismo. Queste ultime sono due *bytes* impostati a piacere dal progettista (purché non siano una successione di uni e zeri) che vengono inclusi automaticamente all'inizio di una trasmissione di dati.

Il CC1101 offre, oltre ai servizi di modem, ulteriori blocchi digitali per la gestione dei dati. Il blocco *FEC*[3]*/Interleaver* svolge la funzione di codificatore/decodificatore in modo da irrobustire la comunicazione. Il *Packet Handler* si occupa dell' impacchettamento e lo spacchettamento automatico dei dati trasmessi. Questo blocco è di particolare importanza se si vuole un minimo di controllo di integrità sui dati trasmessi. Inoltre, sistemi che sfruttano bande condivise per comunicare devono includere una forma di impacchettamento dei dati, atta a garantire la loro provenienza. Infatti, le normative limitano solamente al *duty cycle* dispositivi sprovvisti di meccanismi *Listen Before Talk* (CSMA/CA) e nulla vieta che ci siano collisioni.

Il tipo di pacchetti che il transceiver è in grado di gestire automaticamente è quello mostrato nella seguente Figura 2.5.

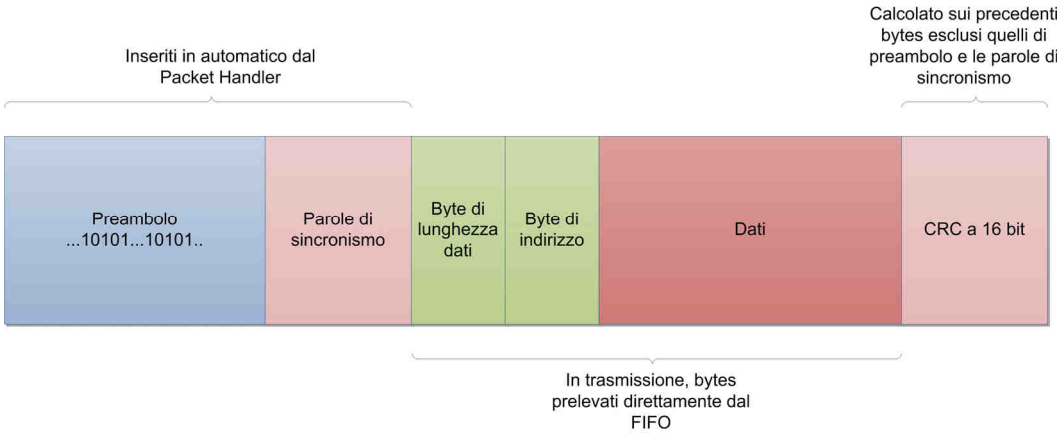


Figura 2.5. Struttura di un pacchetto radio generico

Il tipo di pacchetti che il transceiver è in grado di gestire automaticamente è quello mostrato nella seguente Figura 2.5. Un pacchetto ha dimensioni multiple di un *byte*. Infatti il CC1101 gestisce parole lunghe un *byte* per volta. All'inizio di una trasmissione, il modulatore genera un preambolo: una serie di uni e zeri alternati che servono ad occupare il canale ed avvertire il ricevitore della comunicazione. Poi, le parole di sincronismo aggiunte permettono al ricevitore di individuare l'inizio del pacchetto (sincronismo di pacchetto). Seguono adesso due *bytes* opzionali immessi dal mittente. Al primo corrisponde il numero di *bytes* di dati che il pacchetto sta trasmettendo, se questo è variabile. Altrimenti

[3] *Forward Error Correction*

se la lunghezza è impostata come fissa alla configurazione del *transceiver*, questo *byte* non deve più essere immesso nel pacchetto. Il secondo permette di indirizzare i pacchetti fino a 255 dispositivi diversi che costituirebbero dunque una rete wireless di SRD. Il transceiver è provvisto di meccanismi di filtraggio dei pacchetti non destinatigli: il *Packet Handler* è in grado di scartare pacchetti se il *byte* di indirizzo è diverso da quello assegnato al dispositivo stesso. Dopo questi *bytes* opzionali vengono immessi quelli dei dati per una dimensione massima di 255 *bytes*. Il *Packet Handler* è in grado di effettuare inoltre un controllo *CRC* (*Cyclic Redundancy Check*) sui *bytes* di lunghezza, indirizzo e dati, accodando al pacchetto in trasmissione il risultato a 16 *bit*. In ricezione invece ripete l'operazione e né confronta il risultato con i due *bytes* accodati. Se questi sono diversi allora l'intero pacchetto verrà scartato.

I dati fluiscono attraverso due buffer FIFO da 64 *bytes* ai quali si accede attraverso un'interfaccia di comunicazione seriale, la stessa che viene impiegata per il controllo del *chip*.

2.1.1.3 L'interfaccia digitale

La configurazione del CC1101 avviene attraverso un'interfaccia SPI, *Serial Peripheral Interface*, in cui fluiscono anche i dati ricevuti e da trasmettere. L'SPI è un'interfaccia sincrona e full-duplex: una linea di clock in ingresso, SCLK, temporizza (con prelievo ai fronti di salita) il flusso sulle linee SO (*Serial Output*) e SI (*Serial Input*), rispettivamente per dati in uscita ed in ingresso al *transceiver*. Un ulteriore linea, CSn (*Chip Select attivo basso*), permette ad un microcontrollore (*Master*) di stabilire una comunicazione col transceiver (*Slave*). Oltre alle quattro linee SPI sopra elencate ci sono due linee opzionali configurabili a piacere, GDO0 e GDO2, che permettono di tenere sotto controllo alcune funzioni del transceiver.

La semplicità nell'uso del CC1101 è essenzialmente dovuta ad una serie di registri di un *byte* che mappano le funzioni del *transceiver*.

A questi registri si accede inviando un primo *byte*: 2 *bits* che specificano l'operazione da eseguire (lettura o scrittura, singola o *burst*[4]) sono seguiti da 6 *bits* di indirizzo. In scrittura, questo primo *byte* di accesso viene seguito da quello destinato a sovrascrivere il contenuto del registro. La comunicazione in SPI avviene soltanto a CSn basso, che deve essere riportato al livello alto per interrompere la comunicazione. Il *clock* massimo in modalità *burst* è di 6,5 MHz.

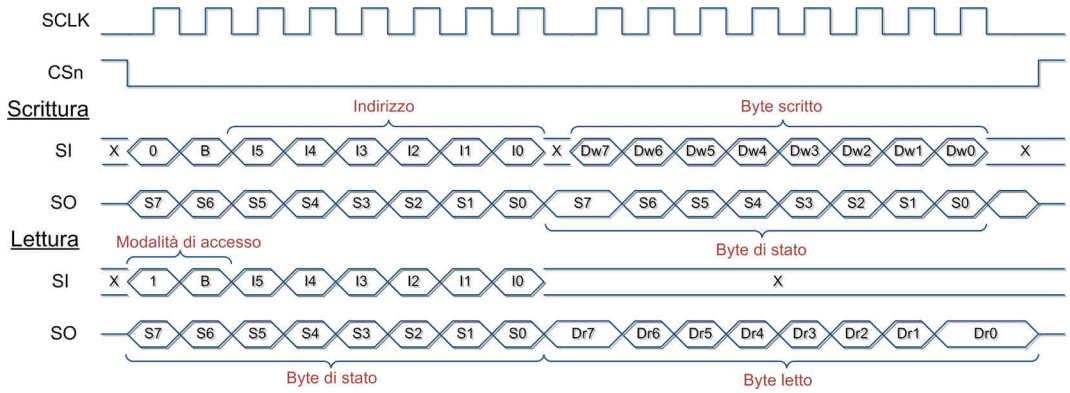


Figura 2.6. Comunicazione su SPI

Durante l'immissione dell'indirizzo viene restituito un *byte* che contiene alcune informazioni sullo stato del *transceiver*.

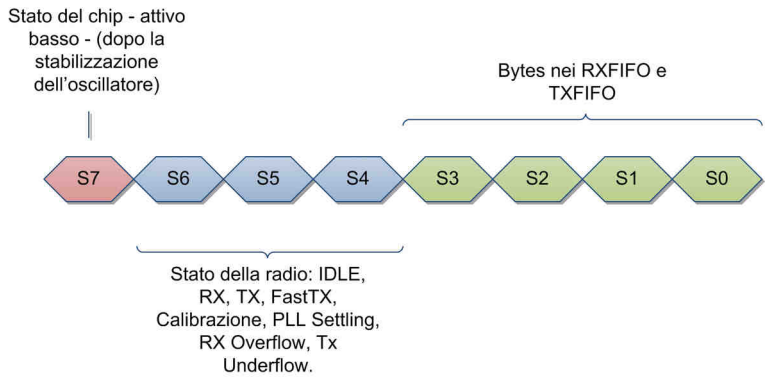


Figura 2.7. Byte di stato del chip

I comandi di controllo dello stato del *transceiver* vengono inviati attraverso un solo *byte* (*command strobe*): viene inviato solamente il *byte* di accesso ad un certo registro, e quest'accesso viene riconosciuto ed interpretato dal *transceiver* come un'operazione da eseguire.

[4] La lettura o scrittura a *burst* si appoggia sull'incremento automatico dell'indirizzo per ripetere l'operazione sul registro successivo

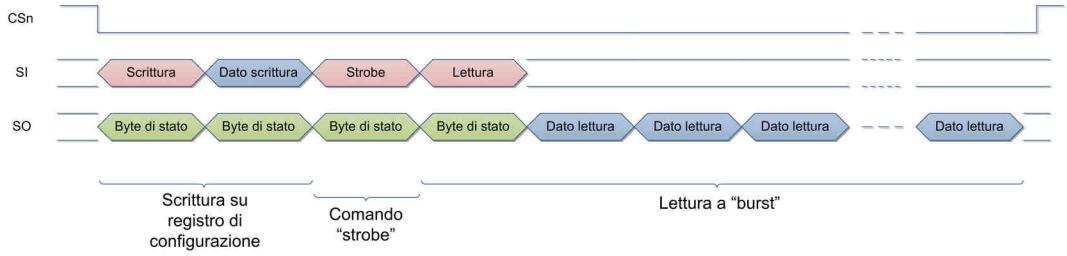


Figura 2.8. Esempio di comunicazione col transceiver

La Figura 2.9 mostra la mappa dei registri del *transceiver*. Il numero dei registri per la configurazione è di 47. Ci sono inoltre 13 *command strobes* possibili per i quali vengono assegnati 13 indirizzi. Altri 13 registri ai quali si accede in sola lettura restituiscono informazioni sullo stato della radio, dei *buffer* FIFO (riempimento) e sulla qualità del collegamento. Un indirizzo viene dedicato alla configurazione del *Power Amplifier*, scegliendo così la potenza fornita all'antenna. Per modulazioni di ampiezza, lo stesso indirizzo mappa 8 registri, accessibili in modalità *burst*, che con valori crescenti del guadagno assicurano un *pulse shaping* del segnale da irradiare e conseguente riduzione della potenza distribuita sui canali adiacenti (*Adjacent Channel Power Ratio*). Infine, l'indirizzo 0x3F permette l'accesso ai *buffer* FIFO, quello in ricezione con un'operazione di lettura e quello in trasmissione impostando il *bit* di scrittura (attivo basso).

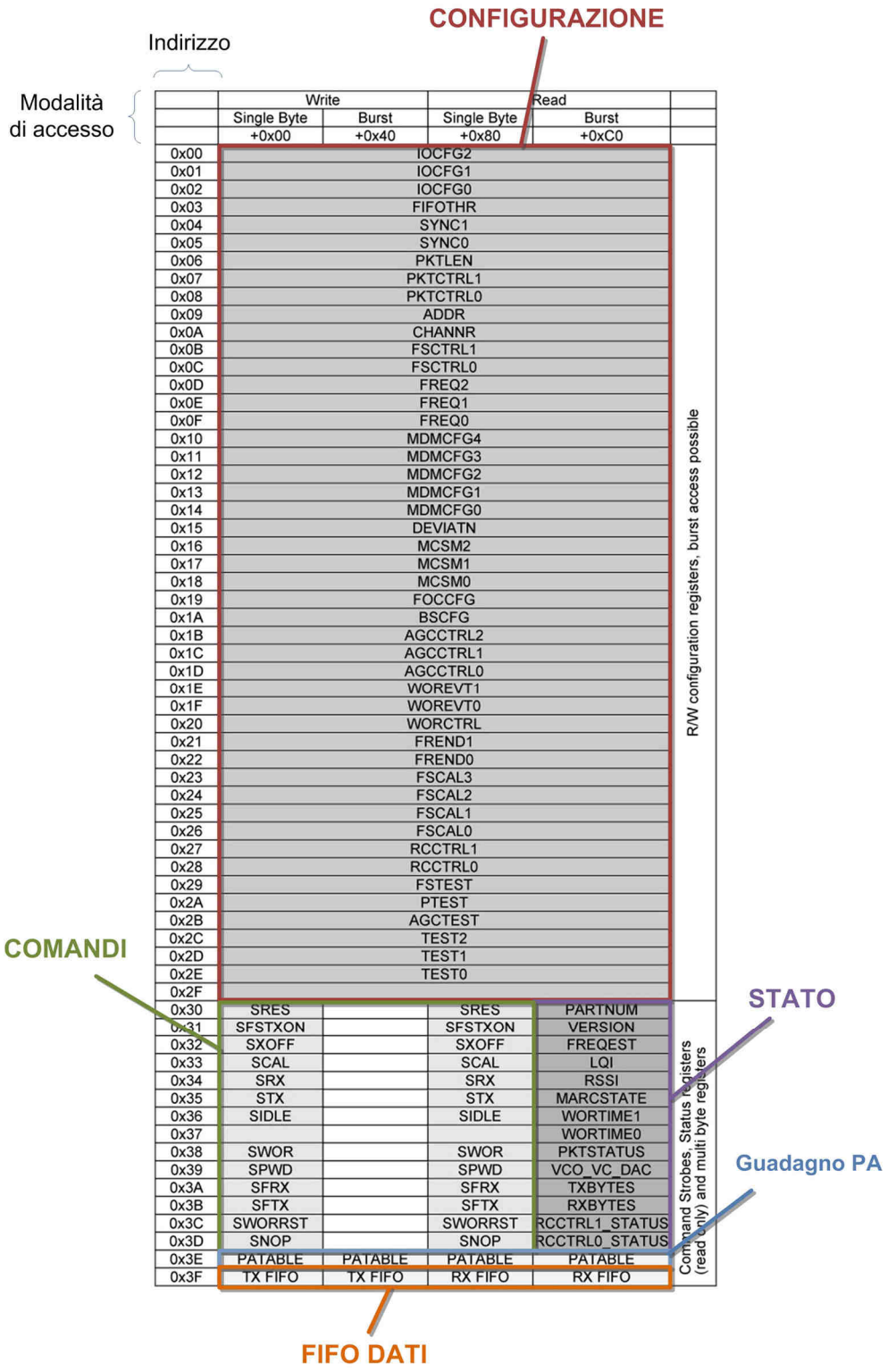


Figura 2.9. Registri del CC1101

2.1.1.4 Stati della radio

Il *transceiver* CC1101 è caratterizzato da diversi stati di funzionamento. Alla messa in tensione esso attiva l'oscillatore locale, aspettando la sua stabilizzazione. Si pone poi nello stato IDLE, ossia in uno stato di attesa inattiva. Da questo stato è possibile, con gli opportuni comandi *strobe* (*PoWerDown* o *Crystal Oscillator Off*), ridurre al minimo il consumo elettrico ponendo il chip in standby.

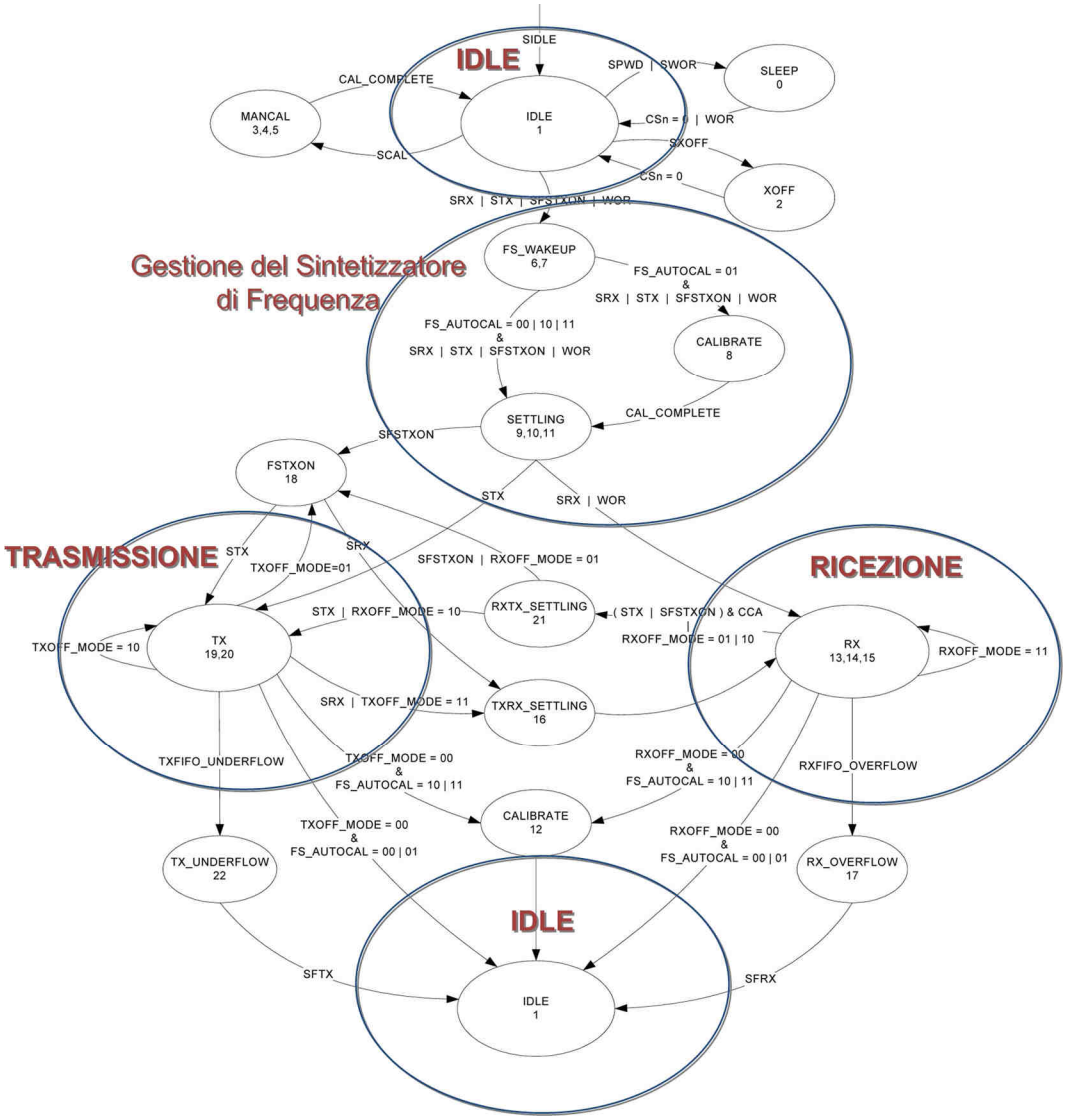


Figura 2.10. Controllo della Radio

Il PLL (*Phase Lock Loop*), blocco fondamentale del sintetizzatore di frequenza, richiede una calibrazione: in questa fase, attraverso meccanismi

automatici di regolazione delle capacità di *tuning* del VCO[5] e delle correnti nella pompa di carica del PLL, vengono impostati i valori ottimali che ne assicurano l'aggancio. Da questo procedimento risulta una comunicazione affidabile in cui la sintonizzazione è la più precisa possibile. La fase di calibrazione impiega tipicamente poco meno di un millisecondo.

I principali stati successivi, controllati dal microcontrollore attraverso gli *strobes*, sono quelli di attivazione e di assestamento del sintetizzatore di frequenza, stati dopo i quali il chip può essere posto in ricezione o in trasmissione. Un uso improprio dei FIFO costringe il chip ad interrompere la comunicazione, ponendosi in *RX Overflow* o *TX Underflow*, stati dai quali si può uscire soltanto con gli *strobe* SFRX e SFTX. Con questa operazione vengono svuotati i FIFO e il chip torna in IDLE.

Vediamo adesso come vengono gestiti i dati da trasmettere o da ricevere, nei corrispondenti stati della radio.

Trovandosi nello stato IDLE, la scrittura di un *byte* nel TXFIFO non basta per iniziare la trasmissione. In effetti, il sintetizzatore di frequenza è a riposo. Inoltre, se i pacchetti radio da formare sono di lunghezza variabile, il *Packet Handler* si aspetta che questo primo *byte* sia quello di lunghezza e dovrà quindi essere seguito dai *bytes* necessari al completamento del pacchetto, evitando così che il *transceiver* entri nello stato *TX Underflow* appena viene mandato il *command strobe* STX di inizio trasmissione. Se invece si è deciso di trasmettere pacchetti di dimensione fissa e tutti i dati sono stati immessi nel TXFIFO, il *Packet Handler* esegue il *checksum* mentre il modulatore inizia a trasmettere il preambolo. La trasmissione dei successivi *bytes* di sincronismo, di dato e di CRC avviene alla fine del calcolo *checksum*, se tuttavia è stato mandato il numero minimo di *bytes* di preambolo. Lo stato successivo alla trasmissione viene automaticamente scelto in funzione del contenuto del registro di configurazione MCSM1 (*Main Radio Control State Machine*, secondo byte). Di default, lo stato torna in IDLE. È comunque possibile immettere nello stato TX il *transceiver* (in modo da impegnare il canale) anche se i FIFO sono

[5] Nel *Voltage Controlled Oscillator* queste capacità sono integrate ad array e costituiscono la capacità variabile per una regolazione fine della frequenza generata

vuoti, a condizione che dal momento in cui viene scritto un *byte* si aggiungano immediatamente quelli necessari a completare il pacchetto radio, altrimenti anche in questo caso va in *TX Underflow*.

Per ricevere un pacchetto radio, il *transceiver* deve trovarsi nello stato RX. In questo stato, il demodulatore rimane in attesa di un preambolo. Se vengono ricevute delle parole di sincronismo valide, allora il *Packet Handler* inizia a gestire i dati immettendoli nell'RXFIFO. Se il pacchetto non è corrotto, ovvero se il risultato del *checksum* in ricezione è identico a quello accodato in trasmissione, allora il *bit* di stato CRC_OK del registro PKTSTATUS indica al microcontrollore che può prelevare il contenuto del pacchetto, altrimenti l'RXFIFO viene automaticamente svuotato (*autoflush*). È necessario prelevare i pacchetti ricevuti affinché il transceiver non vada in *RX Overflow,* stato in cui l'RXFIFO è pieno e da cui si esce col comando SFRX (il *buffer* viene svuotato e la radio entra in IDLE). Un'opzione del *Packet Handler* è quella di aggiungere ai *bytes* ricevuti (lunghezza, indirizzo e dati) due *bytes* di stato, il *Received Signal Strength Indicator* e il *Link Quality Indicator*. Questi contengono informazioni sulla qualità del segnale ricevuto col pacchetto.

2.1.1.5 L'*Evaluation Module* CC1101EM

Il montaggio del chip delle dimensioni di 4x4 mm richiede un'alta precisione, non ottenibile se non con l'ausilio di strumenti automatici. Si è quindi deciso di acquistare l'*Evaluation Module Kit* del CC1101. Il kit comprende, oltre a due schede sulle quali vengono montati il chip e tutti i componenti esterni necessari al suo funzionamento, due antenne a quarto d'onda per un operatività a 868 MHz. Queste schede vengono alimentate e connesse al microcontrollore attraverso due connettori a montaggio superficiale posti sul retro. Il cristallo di oscillatore che ci viene montato è di 26 MHz.

Figura 2.11. Antenna e scheda dell'*Evaluation Module*

2.1.2 Il microcontrollore PIC18LF4620

Come visto nel precedente paragrafo 2.1.1.3, il *transceiver* CC1101 viene configurato e gestito attraverso l'interfaccia SPI di un microcontrollore. La scelta del microcontrollore è risultata particolarmente immediata vista la disponibilità in laboratorio del programmatore. Si tratta di un microcontrollore a 8 bits della Microchip. La serie PIC18 è caratterizzata da ampie memoria Flash per il codice operativo e RAM per i dati volatili, e dalla possibilità di avere fino a 70 piedini di I/O detti *General Purpose Input/Output* per una maggior versatilità d'impiego. Oltre alle funzioni primarie comuni alla maggior parte dei microcontrollori, i PIC18 hanno una moltitudine di periferiche dedicate a particolari funzioni. Due di queste periferiche, i moduli MSSP (*Master Synchronous Serial Port*) e EUSART (*Enhanced USART*[6]) sono di particolare

[6] Universal Synchronous Asynchronous Receiver Transmitter

interesse per il progetto: il primo permette di implementare l'interfaccia SPI e il secondo di realizzare la comunicazione in RS232. Il vantaggio principale dell'inserimento nel *chip* di moduli è di ridurre l'impegno del cuore (*core*) del microcontrollore. La Figura 2.12 illustra lo schema a blocchi del PIC18F4620.

Come per il CC1101, tutte le funzioni del microcontrollore sono mappate attraverso registri, gli SFR (*Special Function Registers*) e i *Configuration Registers*. I registri SFR vengono modificati per gestire il microcontrollore durante il suo funzionamento. Al contrario, i *Configuration Registers*, essendo implementati nella EEPROM, non possono essere cambiati durante l'esecuzione del programma. Un esempio di mappatura che si ottiene con gli SFR è quella dei piedini del microcontrollore: attraverso alcuni registri se ne imposta lo stato (livello alto o basso). Questi registri sono denominati *PORTx* dove *x* è una lettera che rappresenta il gruppo di 8 piedini gestiti. Altri registri, i *TRISx*, impostano i piedini come ingressi o uscite.

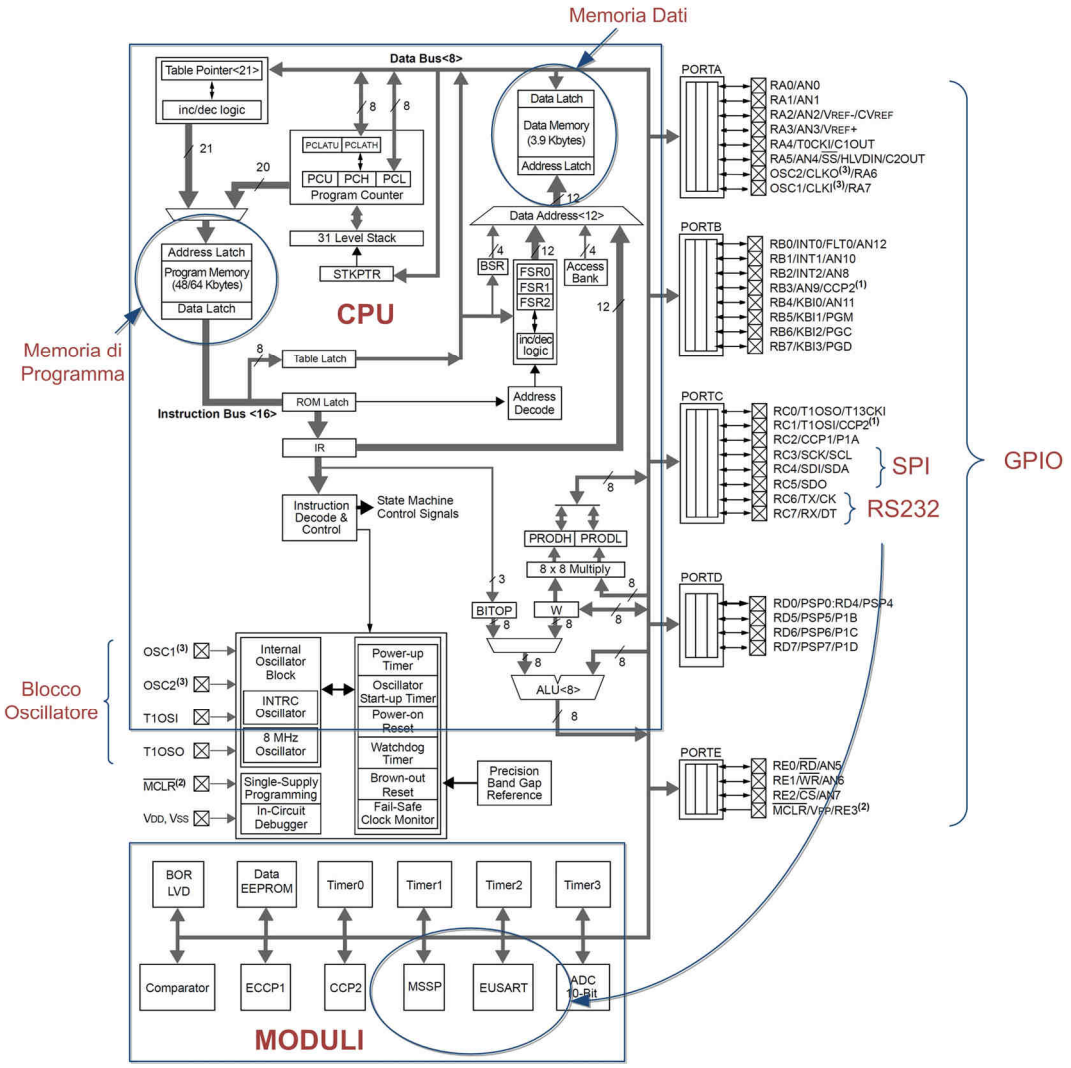


Figura 2.12. Blocchi costitutivi del PIC

2.1.2.1 Il cuore del microcontrollore

Il cuore di un microcontrollore della serie PIC18 è di architettura *Harvard* in cui la memoria di programma viene indirizzata con 21 *bits*, e che potrebbe dunque essere estesa a 2 MBytes. Le istruzioni occupano due celle da un *byte*. La memoria dati, implementata con una RAM statica, è indirizzata con 12 bits permettendo un'allocazione fino a 4 KBytes. Il microcontrollore ha anche una regione di memoria dati non volatile alla quale si può accedere durante l'esecuzione.

Il ciclo di istruzione del microcontrollore è di quattro periodi di *clock*, ma un flusso *pipeline*, in grado di eseguire fino a quattro istruzioni contemporaneamente, riduce l'esecuzione ad un solo effettivo periodo di *clock*.

I PIC18 sono stati sviluppati in modo da supportare diverse modalità di funzionamento, vista l'alta integrazione di periferiche indipendenti. In effetti, nello stato IDLE, la CPU può essere disabilitata mantenendo i moduli attivi. Un'efficace complesso di interruzioni permette di riattivarla. CPU e moduli possono essere, allo stesso tempo, abilitati (stato RUN) o disabilitati (stato SLEEP).

La gestione del consumo durante il funzionamento è affidata alla possibilità di variare la frequenza dell'oscillatore che determina la corrente assorbita. Diverse sono le fonti di *clock (*Figura 2.13*)*: quattro piedini permettono di realizzare oscillatori a quarzo o RC (con resistenza e capacità), oppure possono essere gli ingressi a due generatori esterni autonomi (fino a 40 MHz). Il chip integra anche un blocco oscillatore interno (INTOSC) a 8 MHz ed uno RC a 31 KHz che permettono, dopo opportune scalature (*postscaling*) e moltiplicazioni, di ottenere clock da 31 KHz a 32 MHz. Le moltiplicazioni avvengono per mezzo di un PLL che quadruplica la frequenza di riferimento. Perché il PLL possa funzionare con INTOSC, la frequenza impostata con il *postscaler* deve essere di 4 o 8 MHz, permettendo di avere rispettivamente 16 e 32 MHz.

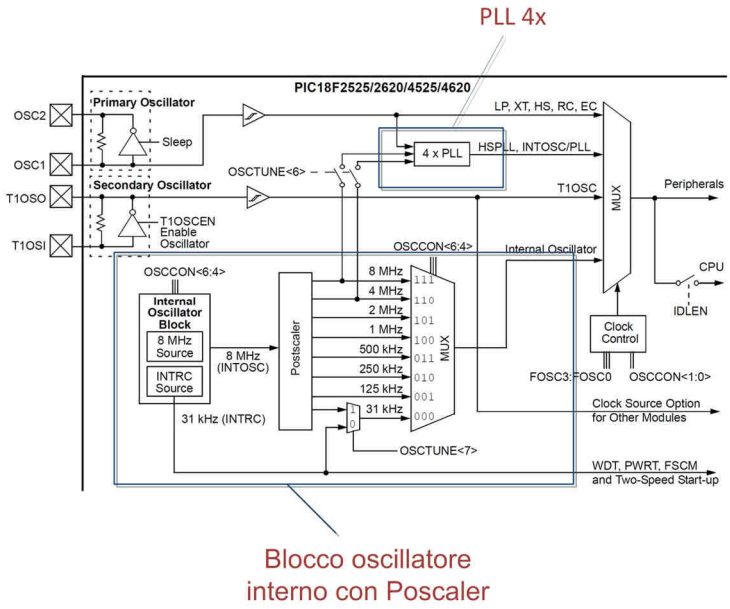


Figura 2.13. Blocco oscillatore

2.1.2.2 Il modulo MSSP

L'MSSP è in grado di realizzare un interfaccia SPI sia modalità *Master* che *Slave*. Con l'ausilio di due registri di configurazione (*SSPSTAT* e *SSPCON1*) si possono impostare la frequenza di *clock* (SCLK) e gli istanti di acquisizione e di restituzione dei dati.

I blocchi funzionali dell'MSSP vengono illustrati in Figura 2.14. Un registro a scorrimento accumula, temporizzato da SCLK, i *bits* in trasmissione ed in ricezione. Un'interruzione alla CPU avverte del completamento della comunicazione, consentendogli di prelevare e memorizzare un *byte* ricevuto.

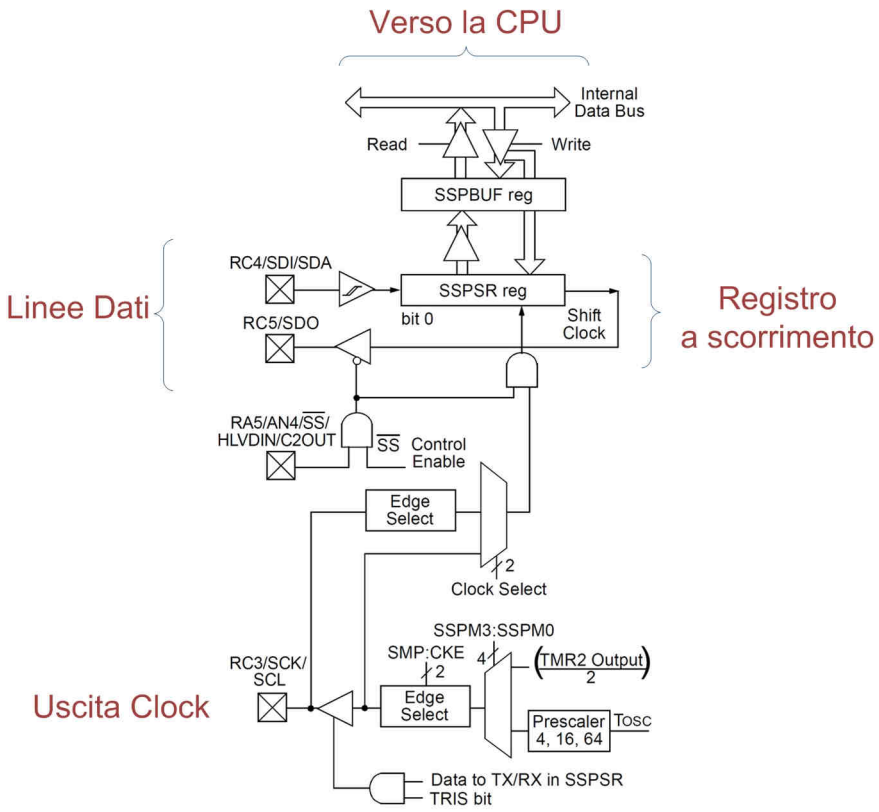


Figura 2.14. Schema del modulo MSSP per l'SPI

Nel nostro caso, il microcontrollore è *Master* per il *transceiver*, e deve poter abilitare la comunicazione SPI attraverso la linea CSn. Si utilizzerà a questo scopo un piedino libero (GPIO) del microcontrollore.

2.1.2.3 Il modulo EUSART

Generalmente noto come SCI (*Serial Communications Interface*), il modulo EUSART è in grado di implementare comunicazioni seriali sia sincrone che asincrone. La Figura 2.15 illustra il funzionamento dell'EUSART Una modalità asincrona analoga a quella dell'RS232 si ottiene impostando i registri di trasmissione e ricezione, rispettivamente TXSTA e RCSTA, in modalità UART. Occorre inoltre configurare il "generatore di *baud rate*", blocco in grado di generare le frequenze standard per la comunicazione asincrona e dunque permettere il sincronismo di *bit*.

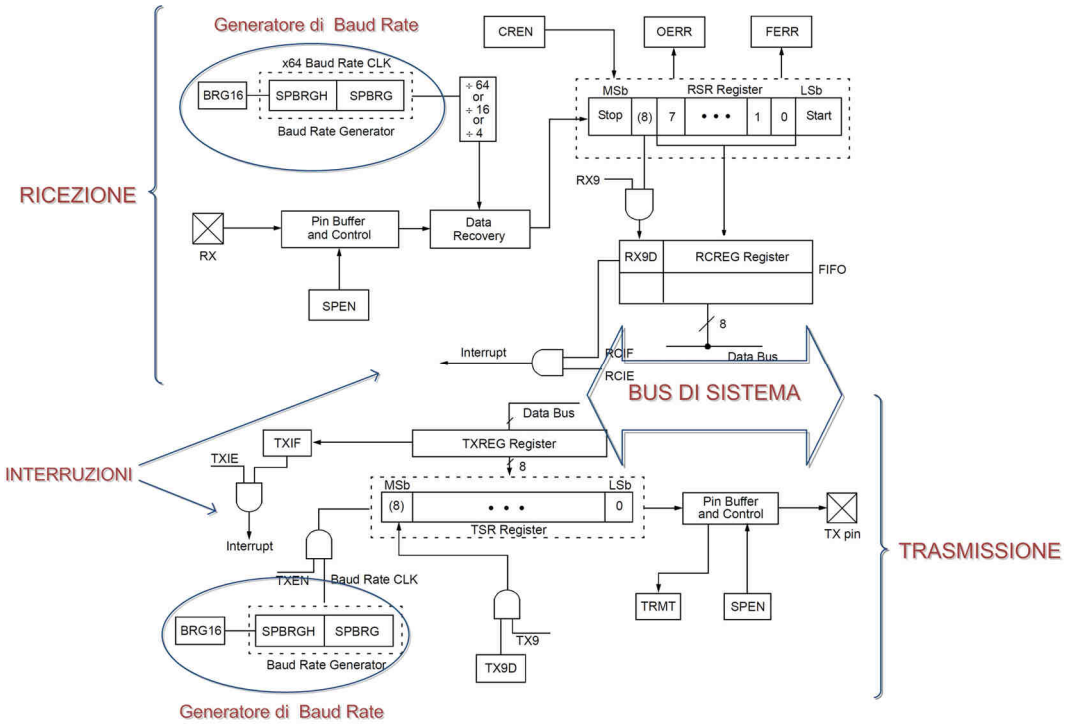
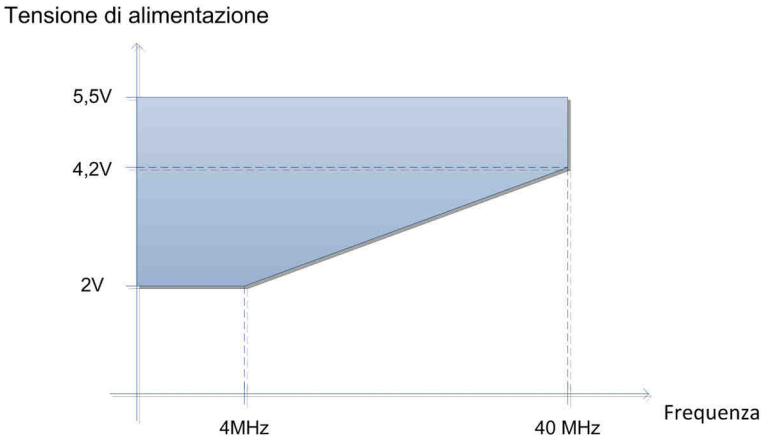


Figura 2.15. Il modulo EUSART

Il *baud rate* viene generato con l'ausilio di un contatore ad 8 o 16 bits, in *free-running* al *clock* di sistema. Poi con divisori di frequenza viene allungato il tempo di segnalazione del contatore, in modo da ottenere il tempo di *bit* adeguato per la comunicazione. Essendo il *baud rate* derivato dall'oscillatore, valori di standard sono ottenibili soltanto con quarzi di frequenze standard o multiple. Negli altri casi vi è un errore sul tempo di *bit* non sempre trascurabile.

2.1.2.4 Considerazioni sull'alimentazione

Il PIC18LF4620 supporta un range di alimentazione da 2 a 5,5 V. Alcune considerazioni sulla frequenza di *clock* devono essere fatte per le tensioni sotto 4,2 V. In effetti, il microcontrollore non vi è più in grado di lavorare con frequenze di *clock* troppo alte. Il vincolo tensione/frequenza segue la legge illustrata in Figura 2.16.



$$f_{MAX} = 16{,}36 \times (V_{DD} - 2) + 4 \quad [MHz]$$

Figura 2.16. Vincolo Tensione/Frequenza

Il *transceiver* CC1101 ammette tensione massima di 3,6V. Per evitare una rottura del componente è necessario alimentare l'intero Modem RF con una tensione minore o uguale. Questo vuol dire che la frequenza massima di *clock* che il microcontrollore (a 3,6V) è in grado di sopportare è di 30 MHz.

2.1.3 Il Driver/Receiver MAX3232

Le tensioni sui piedini collegati al modulo EUSART sono a logica TTL/CMOS. Per rispettare le specifiche RS232 è necessario interfacciare al microcontrollore un dispositivo in grado di portare ad una tensione negativa il livello logico alto e ad una positiva quello basso. Quest'operazione viene svolta da un *Driver/Receiver* come il MAX3232 della *Maxim,* illustrato in Figura 2.17. Quest'ultimo sfrutta il principio della pompa di carica per generare le tensioni giuste, talvolta maggiori di quella di alimentazione. A 3 V di alimentazione, le linee di trasmissione sull'RS232, se caricate leggermente, generano tensioni di pilotaggio di ±5,5 V, sufficientemente alte da poter essere individuate come livelli logici da un DCE standard.

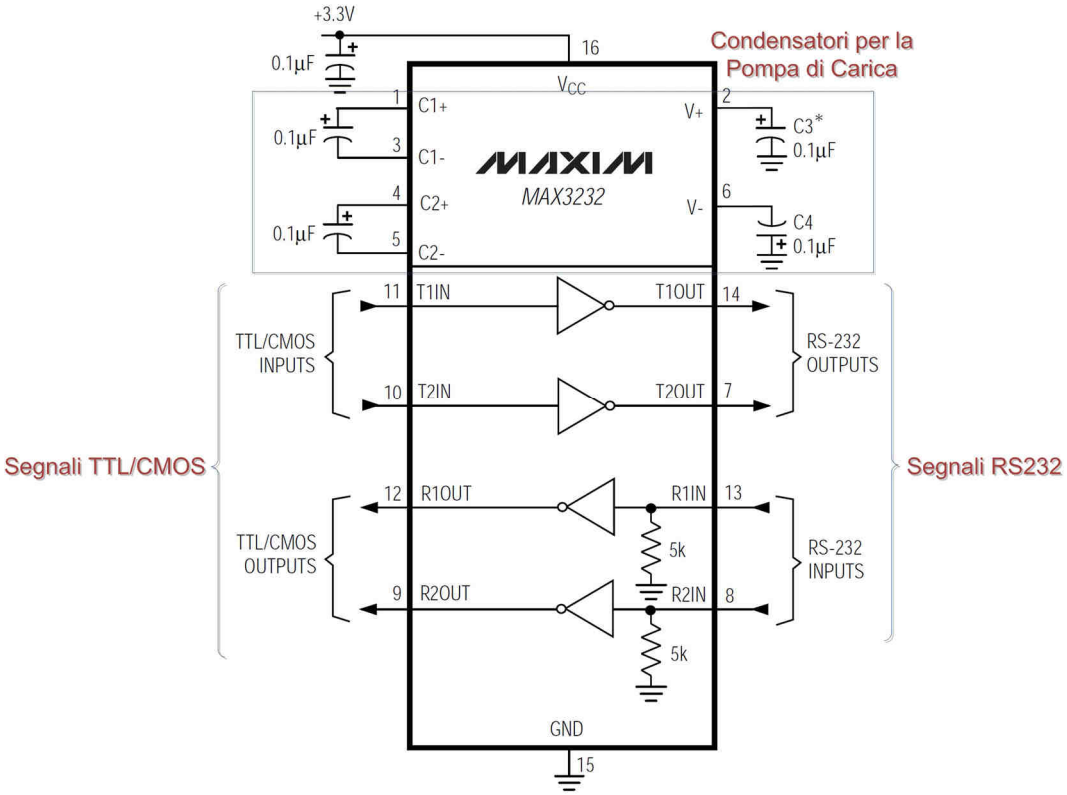


Figura 2.17. Schema circuitale del MAX3232

2.2 Circuito elettrico

La precedente fase di studio dei componenti scelti ci permette adesso di progettare il circuito elettrico del Modem RF. Si deve prima scegliere la tensione di alimentazione comune a tutti componenti. Il limite inferiore è imposto dal MAX3232 che potrebbe non funzionare correttamente con tensioni sotto i 3 V. Quello superiore invece è legato al CC1101, che rischierebbe di guastarsi con tensioni maggiori di 3,6 V. Verrà aggiunto al circuito del Modem RF un classico regolatore a dissipazione che stabilizzerà la tensione a 3,3 V. A questo scopo si è scelto l'LP38692, regolatore *low-dropout* della *National Semiconductors* in grado di erogare fino ad 1 A. Questo limite non verrà mai raggiunto. In effetti, la somma delle correnti massime assorbibili dai tre blocchi costitutivi è minore di 350 mA. Il regolatore ammette in ingresso una tensione massima di 10 V, e per poter mantenere 3,3 V in uscita per qualsiasi corrente di carico richiede un minimo di 4,3 V in ingresso.

Controlliamo adesso il valore della frequenza massima operativa del microcontrollore ricordando che è questa vincolata dalla tensione di alimentazione. Il costruttore del regolatore garantisce una tensione di 3,3V con una tolleranza massima di ±5%. La frequenza massima di clock per il microcontrollore viene dunque calcolata su 3,135 V minimi, e risulta di circa 22,5 MHz (si vedi il paragrafo 2.1.2.4). La frequenza massima dell'oscillatore interno che si potrà utilizzare è di 16 MHz. Vista la lieve variazione della velocità computazionale, un oscillatore esterno risulterebbe superfluo. Si decide dunque di lavorare a 16 MHz con l'oscillatore interno (4 MHz e PLL 4x attivo).

Procediamo tracciando le connessioni tra i piedini dei componenti scelti partendo dal microcontrollore. Quest'ultimo è disponibile in versione DIP (*Dual Inline Package*) a 40 *pin* dei quali 36 sono di I/O e 4 servono all'alimentazione. I moduli MSSP e EUSART essendo attivi, non tutti i piedini saranno disponibili per operazioni di I/O. L'MSSP, come illustrato nello schema a blocchi (Figura 2.14), occupa tre piedini e l'EUSART (Figura 2.15) due. Per completare l'interfaccia SPI, ai 3 piedini (18, 23 e 24) assegnati di c all'MSSP, SCLK e SDO come *outputs* e SDI come *input*, va aggiunto CSn come *output* dal microcontrollore. Le linee opzionali GDO0 e GDO2 del *transceiver* vengono aggiunte come ingressi generici, anche se il costruttore consiglia di utilizzarle per interruzioni al microcontrollore. Questa funzione si ottiene soltanto con i piedini gestiti da PORTB. Le 6 linee sopra elencate vengono portate ad uno dei connettori dell'*Evaluation Module*, l'altro essendo esclusivamente dedicato alla sua alimentazione.

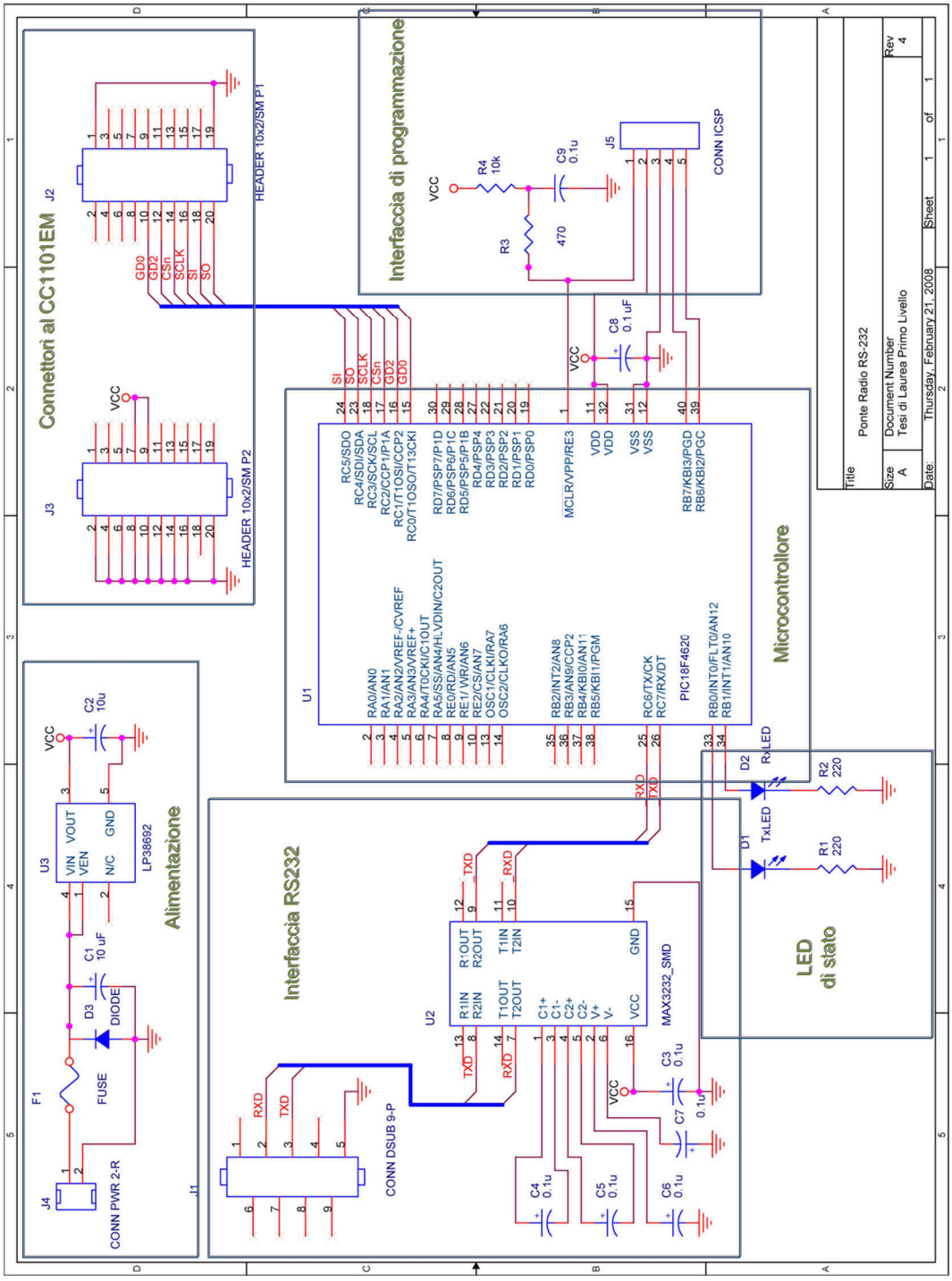


Figura 2.18. Schema elettrico

L'EUSART invece usa automaticamente i piedini 25 e 26, rispettivamente per la trasmissione (*output*) e per la ricezione (*input*) di dati. Queste due linee diventano ingresso e uscita CMOS/TTL per il MAX3232 che le porterà ad un connettore DB-9 per completare l'interfaccia RS232.

Il codice operativo del microcontrollore viene immesso in memoria di programma (di tipo Flash) attraverso una particolare interfaccia di programmazione seriale, l'*In-Circuit Serial Programming*. Col piedino 1, il programmatore fornisce una tensione V_{PP} sufficientemente alta (circa 13 V) necessaria alla scrittura su questo tipo di EEPROM. Poi con le linee PGC (*Programming Clock*) e PGD (*Programming Data*) inizia una comunicazione seriale sincrona attraverso la quale fluisce il codice macchina. Il piedino 1 serve anche a resettare il microcontrollore con una tensione bassa e, dunque, per funzionare il PIC deve avere una resistenza di *pull-up* collegata a questo piedino. I PIC18 ne hanno una interna che è possibile attivare portando a 0 un *bit* di configurazione (MCLRE). In questo modo si è in grado di sfruttare il pin come ingresso. Nel nostro caso si è deciso di porre sul pin, oltre al *pull-up* resistivo, un condensatore di filtro che introduce un ritardo tra la messa in tensione e quella in funzione, garantendo un "*reset* all'avvio".

Due piedini del microcontrollore sono stati dedicati all'accensione di due LED, uno rosso che indicherà lo stato in trasmissione e l'altro verde per la ricezione. Sui piedini di alimentazione dei PIC e MAX3232 vi sono state messe delle capacità di filtro da 0,1 µF in grado di cortocircuitare eventuali transitori indesiderati.

Vediamo infine il circuito di alimentazione basato sull'LP38692: condensatori da 10 µF in ingresso ed in uscita stabilizzano il regolatore, evitando nefaste reazioni positive. Un fusibile protegge il circuito da possibili inversioni della polarità sull'alimentazione: in tal caso il diodo entra in funzione facendo scorrere nel fusibile una corrente maggiore del potere di interruzione.

2.3 Schema di funzionamento

2.3.1 La conversione dei dati

Vediamo adesso il lavoro che il microcontrollore deve svolgere affinché i dati possano fluire in entrambi i versi del collegamento wireless. Si può

considerare il microcontrollore come l'interfaccia di comunicazione tra il transceiver e la parte del sistema radar a cui vi si collega.

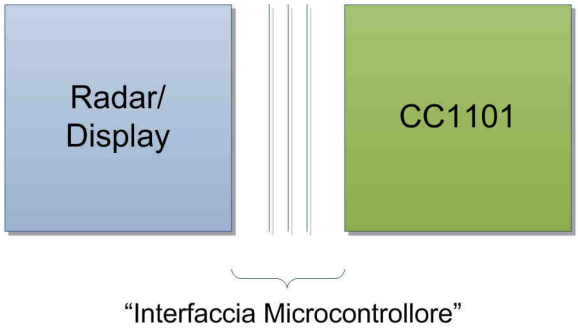


Figura 2.19. Interfaccia Microcontrollore

Questo tipo di interpretazione ci permette di capire che il microcontrollore dovrà, almeno per quanto riguarda i dati da ricetrasmettere, occuparsi della conversione da pacchetti su seriale RS232 a pacchetti su SPI e viceversa da SPI a RS232. Questa conversione si ottiene facilmente con i moduli MSSP e EUSART, visto che entrambi gestiscono automaticamente comunicazioni a *byte*. Inoltre questi moduli possono operare contemporaneamente e le velocità con cui fluiscono i dati posso essere diverse. La CPU preleva il *byte* in arrivo in uno dei moduli e lo stocca opportunamente in memoria. Poi, a pacchetto completato (Figura 1.5 e Figura 1.6) dovrà immetterlo *byte* per *byte* nel secondo modulo. L'interfaccia SPI al CC1101 complica leggermente le operazioni in quanto si deve esplicitamente accedere ai *buffer* FIFO. D'altro canto, la velocità che quest'interfaccia è in grado di supportare è di gran lunga maggiore di quella in UART, rendendo questi accessi trascurabili rispetto al flusso complessivo dei dati.

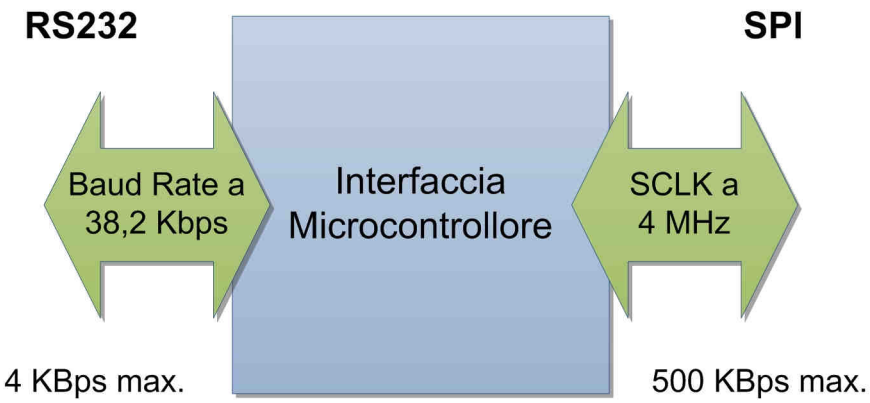


Figura 2.20. Velocità di comunicazione

L'interfaccia microcontrollore lavorerà dunque nel seguente modo, da sinistra verso destra riferendosi alla precedente Figura 2.20:

- All'interruzione di ricezione proveniente dal modulo EUSART, prelievo del *byte* dal *buffer* RCREG con conseguente immissione in una pila FIFO,
- Assegnazione della dimensione del pacchetto se questo costituisce il primo *byte* del pacchetto (*token*),
- Altrimenti attesa del completamento del pacchetto incrementando progressivamente il puntatore in testa alla pila,
- Scrittura nell'ordine di arrivo di ciascun *byte* in SSPBUF, precedendoli dall'indirizzo del *buffer* TXFIFO.
- Infine svuotamento della pila per ripetere l'operazione

Nel verso opposto il meccanismo è lo stesso, cambiano soltanto il tipi di accesso ai moduli: l'MSSP richiede la lettura dei *bytes* in RXFIFO mandando l'opportuno indirizzo. Poi per trasmettere il pacchetto con l'EUSART si esegue una scrittura in TXREG (Figura 2.15).

2.3.2 I pacchetti radio

Si deve decidere adesso il tipo di pacchetti che viaggeranno via etere. La loro forma dipende principalmente dalla quantità di dati che si deve trasmettere e da "quanto spesso" lo si deve fare. Ricordando che esiste un certo ritardo nel passare da trasmissione a ricezione e viceversa, l'alternarsi di questi stati

potrebbe compromettere la comunicazione bidirezionale. La Figura 2.21 illustra quello che potrebbe succedere con pacchetti che si presentano dall'RS232 "troppo spesso" al *transceiver* per la loro trasmissione radio. A fine trasmissione, il transceiver torna in ricezione, ma l'arrivo di un nuovo pacchetto lo riporta immediatamente in trasmissione.

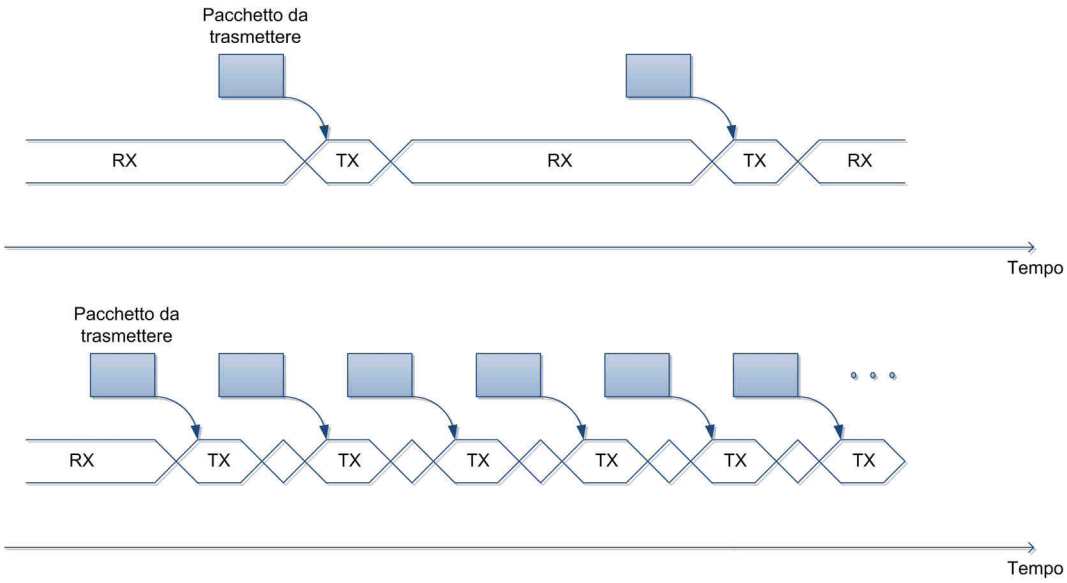


Figura 2.21. Bidirezionalità della comunicazione compromessa

La durata della trasmissione è legata al *baud rate* via etere impostato e alla lunghezza del pacchetto stesso. Per ridurre al minimo il tempo di trasmissione, pur mantenendo una buona sensibilità del demodulatore (circa -95 dBm), si è deciso di impostare il CC1101 in modulazione GFSK a 250 Kbps, ossia circa 7 volte più veloce del *baud rate* in RS232. I ritardi dovuti sia al tempo di assestamento del sintetizzatore di frequenza che all'effettivo allungamento del pacchetto con preambolo, parole di sincronismo e CRC, non influiscono in questo modo nella conversione da f*ull-duplex* ad *half-duplex*.

Come viene illustrato in Figura 2.22, l'impiego di pacchetti di lunghezza variabile che includono più "pacchetti radar" costringe il microcontrollore a confinarli entro dei *byte* per il loro riconoscimento a valle della catena di trasmissione. Infatti per distinguere pacchetti di 5 da quelli di 2 *bytes* è necessario porre delle etichette di intercapedine. Questa soluzione è improponibile in quanto le componenti in fase e in quadratura contenute nel pacchetto radar possono assumere qualsiasi valore (da 0x00 a 0xFF). Il

microcontrollore dovrebbe utilizzare una serie di *byte* per il riconoscimento dei pacchetti radar, ma anche in questo caso sussiste una probabilità di errore.

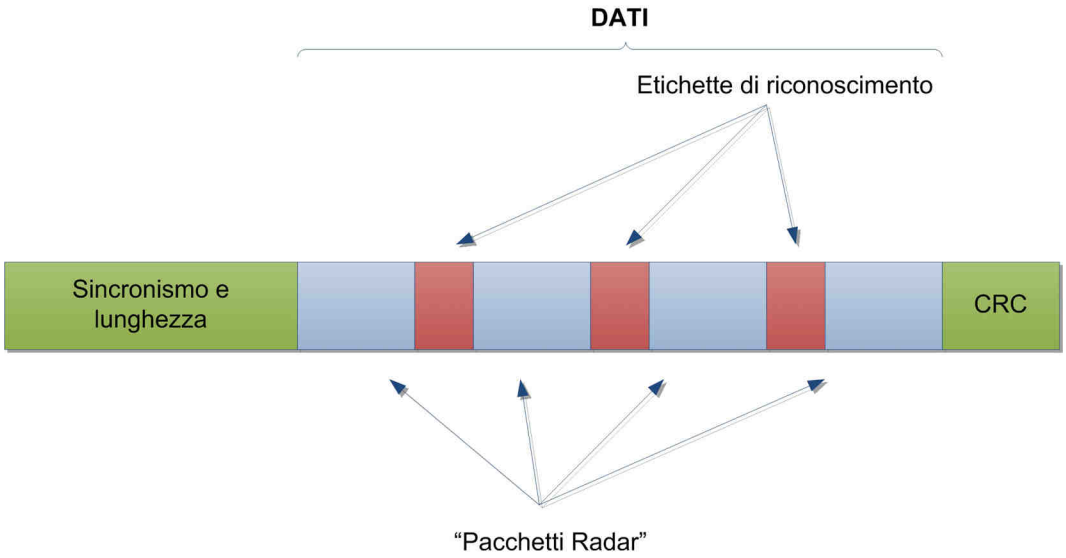


Figura 2.22. Pacchetto con etichette di riconoscimento

La soluzione che si è deciso di utilizzare è quella di pacchetti fissi di 5 *bytes*, come in Figura 2.23, essendo questa la dimensione massima dei pacchetti. In effetti, vista la precedente complicazione legata a pacchetti multipli, è inutilmente dispendioso per il microcontrollore utilizzare pacchetti di dimensioni variabili. Per colmare il deficit che lo schermo di controllo incontra alla trasmissione di comandi verranno aggiunti automaticamente 3 bytes ai 2 già definiti, che verranno poi scartati alla ricezione.

Figura 2.23. Pacchetto Radio

2.3.3 Flusso dei dati

Vediamo adesso, nel diagramma in Figura 2.24, lo schema di gestione del flusso dei dati dal lato radar del collegamento wireless.

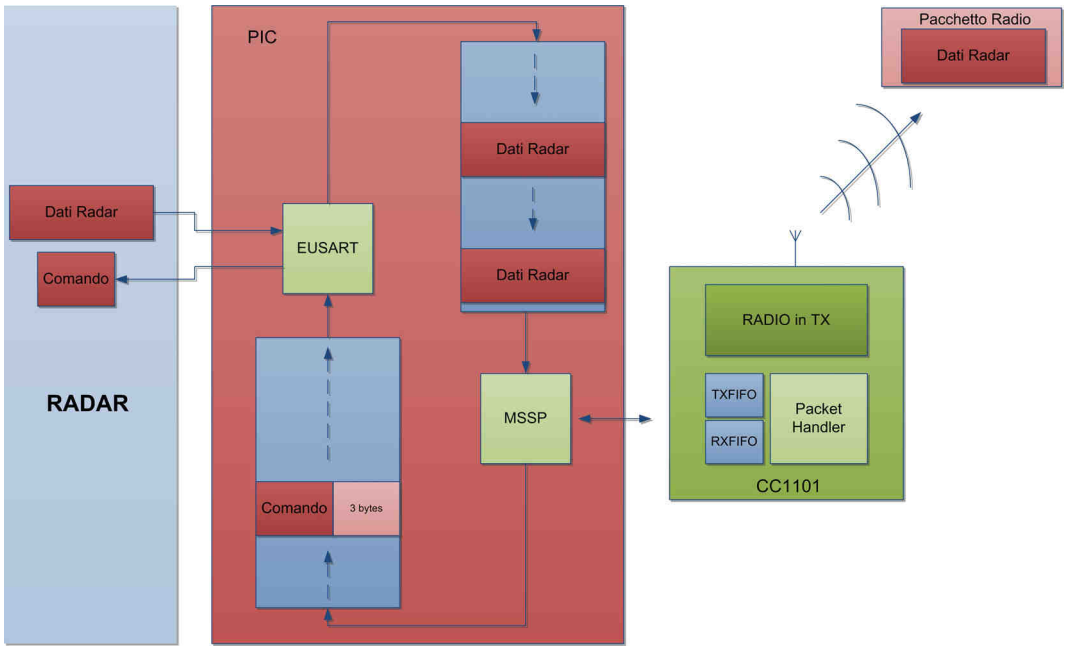


Figura 2.24. Esempio del flusso di dati al Radar

Gli stessi meccanismi vengono riprodotti nel Modem RF lato display. In effetti, i comandi sono stati “normalizzati” a 5 bytes dal microcontrollore in trasmissione e “de-normalizzati” in ricezione.

Il flusso dei dati via etere viene temporizzato dal microcontrollore che decide dello stato del *transceiver*. Attraverso i *comand strobes* (paragrafo 2.1.1.4), riesce a cambiare lo stato della radio da trasmissione a ricezione o viceversa. Una volta immesso un pacchetto nel TXFIFO, il transceiver deve entrare nello stato TX affinché il *Packet Handler* ed il modulatore possano gestirne la trasmissione. Se vi ci si trova già, vuol dire che ha iniziato a mandare il preambolo, occupando così il canale, e soltanto a CRC completato inizia la trasmissione del pacchetto radio. Il *transceiver* posto nello stato RX rimane in ascolto sul canale in attesa di un valido preambolo.

2.4 La configurazione del CC1101

Per poter utilizzare il *transceiver*, il microcontrollore deve opportunamente impostare dei registri del CC1101. Questa fase della progettazione del Modem RF è particolarmente delicata in quanto delle impostazioni non corrette

potrebbero concludersi con un dispositivo non funzionante. Iniziamo con l'impostazione del modulatore e del demodulatore, poi quella del *Packet Handler*.

2.4.1 SmartRF® Studio

Buona parte di questa fase del progetto viene semplificata dal software *SmartRF® Studio* che il produttore mette gratuitamente a disposizione del progettista. Attraverso una semplice ed intuitiva interfaccia grafica si è in grado di impostare la maggior parte dei registri, in particolare quelli dedicati alla parte analogica del *chip* (*Frequency Synthesizer*, *Modulation Formats* , ecc...). L'utilizzo di questo software è fortemente raccomandato dal produttore, soprattutto per ragioni di rispetto delle normative EMC.

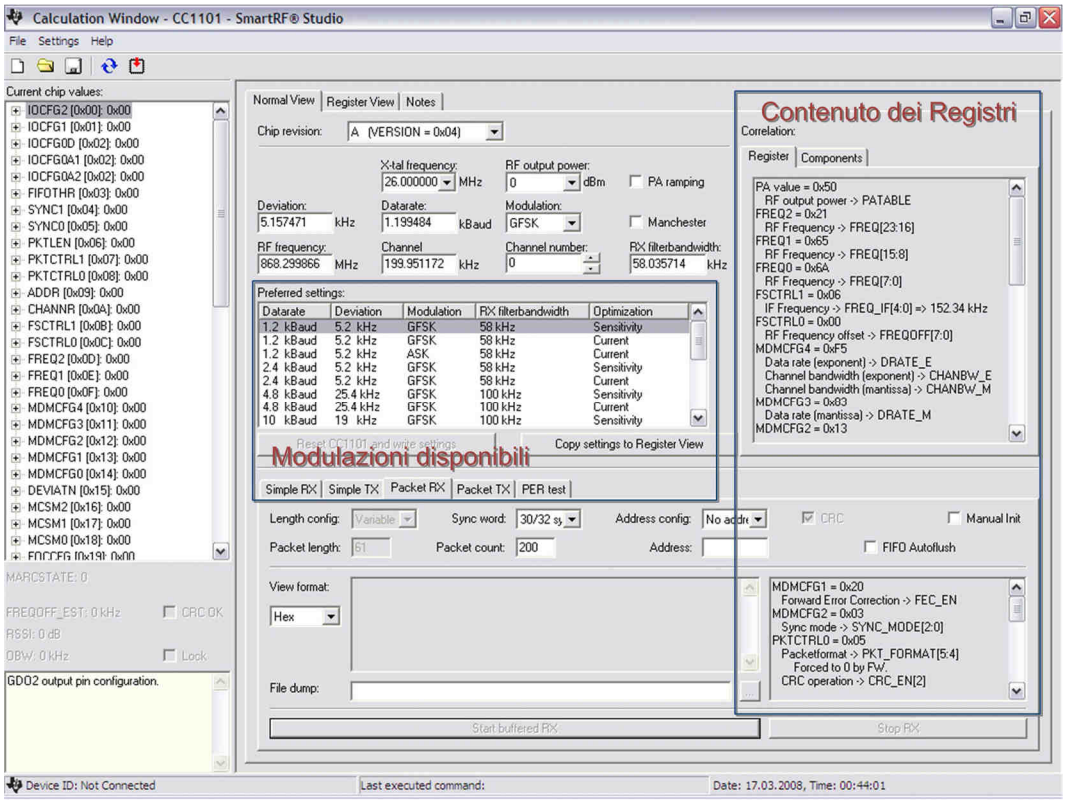


Figura 2.25. *SmartRF® Studio* per la configurazione dei registri

Come si vede nella Figura 2.25, l'interfaccia grafica comprende due parti principali per il controllo dei registri: quella centrale elenca le impostazioni di

default che il produttore raccomanda, e a destra se ne vedono le modifiche al contenuto dei registri.

Selezioniamo dunque l'impostazione a 250 Kbps, 127 KHz di deviazione dalla portante a 868,3 MHz per una modulazione GFSK. L'alto *baud rate* distribuisce la potenza su una banda più larga e pertanto, per raggiungere il livello minimo di potenza necessaria alla demodulazione, il filtro in ricezione viene allargato a 540 KHz. Il software permette inoltre di selezionare un'ottimizzazione della radio per il risparmio energetico o per una maggiore sensibilità in ricezione, ovvero un più ampio raggio di copertura. Viene scelto quest'ultimo.

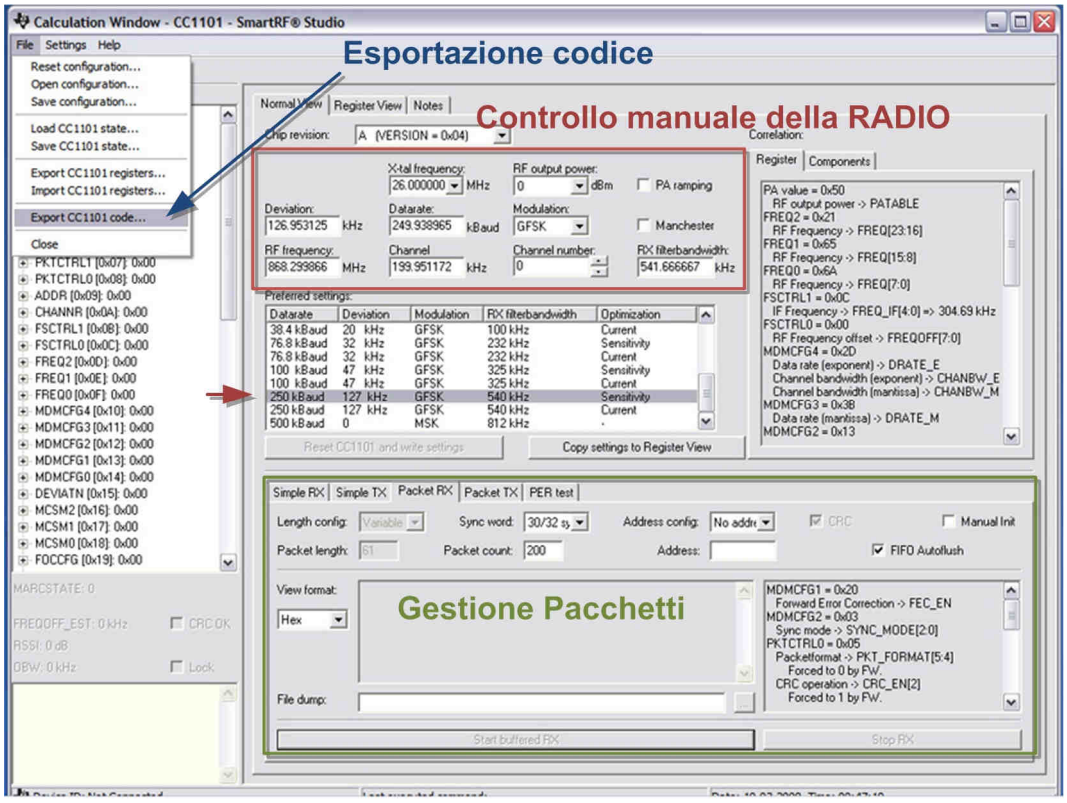


Figura 2.26. Estrazione dell'impostazione scelta

Nel riquadro del controllo manuale della radio, evidenziato nella Figura 2.26, si vedono i parametri effettivi della parte analogica del *transceiver*. Viene selezionato il cristallo di riferimento a 26 MHz. È anche possibile utilizzare una pre-processazione dei dati codificando i dati in modalità *Manchester*. Nel riquadro inferiore della Figura 2.26 invece si denota la gestione dei pacchetti. Viene selezionato il *FIFO Autoflush* affinché il *Packet Handler* scarti

automaticamente i pacchetti corrotti. Il software, essendo progettato per lavorare con l'ambiente di sviluppo *SmartRF® DK,* non ci permette di configurare ulteriormente i pacchetti, ne la potenza in trasmissione. Queste verranno fatte manualmente attraverso accessi singoli ai registri, come verrà spiegato nel paragrafo 0.

Infine, si è in grado di estrarre il contenuto dei registri in un file di testo, oppure di generare il codice da inserire in un file sorgente del programma per microcontrollore.

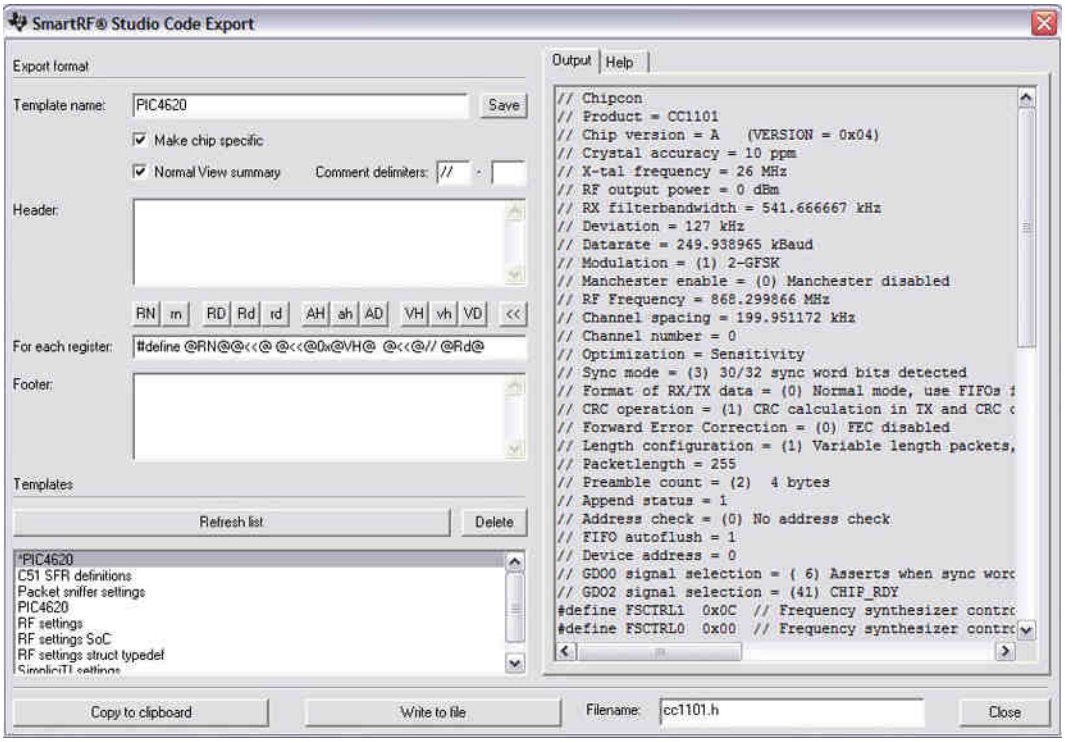


Figura 2.27. Generazione del codice ad hoc

Attraverso la finestra di *Code Export* si imposta il formato del file che si desidera creare, associando il nome del registro col *tag* *@RN@* al suo contenuto in esadecimale, contrassegnato con *@VH@*.

Vediamo nella il codice in linguaggio C che è stato estratto, destinato al compilatore CCS per i microcontrollori PIC18F4620.

```
// Chipcon
// Product = CC1101
// Chip version = A   (VERSION = 0x04)
// Crystal accuracy = 10 ppm
// X-tal frequency = 26 MHz
// RF output power = 0 dBm
// RX filterbandwidth = 541.666667 kHz
// Deviation = 127 kHz
// Datarate = 249.938965 kBaud
// Modulation = (1) 2-GFSK
// Manchester enable = (0) Manchester disabled
// RF Frequency = 868.299866 MHz
// Channel spacing = 199.951172 kHz
// Channel number = 0
// Optimization = Sensitivity
// Sync mode = (3) 30/32 sync word bits detected
// Format of RX/TX data = (0) Normal mode, use FIFOs for RX and TX
// CRC operation = (1) CRC calculation in TX and CRC check in RX enabled
// Forward Error Correction = (0) FEC disabled
// Length configuration = (1) Variable length packets, packet length configured
by the first received byte after sync word.
// Packetlength = 255
// Preamble count = (2)  4 bytes
// Append status = 1
// Address check = (0) No address check
// FIFO autoflush = 1
// Device address = 0
// GDO0 signal selection = ( 6) Asserts when sync word has been sent / received,
and de-asserts at the end of the packet
// GDO2 signal selection = (41) CHIP_RDY

#define FSCTRL1  0x0C  // Frequency synthesizer control.
#define FSCTRL0  0x00  // Frequency synthesizer control.
#define FREQ2    0x21  // Frequency control word, high byte.
#define FREQ1    0x65  // Frequency control word, middle byte.
#define FREQ0    0x6A  // Frequency control word, low byte.
#define MDMCFG4  0x2D  // Modem configuration.
#define MDMCFG3  0x3B  // Modem configuration.
#define MDMCFG2  0x13  // Modem configuration.
#define MDMCFG1  0x22  // Modem configuration.
#define MDMCFG0  0xF8  // Modem configuration.
#define CHANNR   0x00  // Channel number.
#define DEVIATN  0x62  // Modem deviation setting (when FSK modulation is
// enabled).
#define FREND1   0xB6  // Front end RX configuration.
#define FREND0   0x10  // Front end RX configuration.
#define MCSM0    0x18  // Main Radio Control State Machine configuration.
#define FOCCFG   0x1D  // Frequency Offset Compensation Configuration.
#define BSCFG    0x1C  // Bit synchronization Configuration.
#define AGCCTRL2 0xC7  // AGC control.
#define AGCCTRL1 0x00  // AGC control.
#define AGCCTRL0 0xB0  // AGC control.
#define FSCAL3   0xEA  // Frequency synthesizer calibration.
#define FSCAL2   0x2A  // Frequency synthesizer calibration.
#define FSCAL1   0x00  // Frequency synthesizer calibration.
#define FSCAL0   0x1F  // Frequency synthesizer calibration.
#define FSTEST   0x59  // Frequency synthesizer calibration.
#define TEST2    0x88  // Various test settings.
#define TEST1    0x31  // Various test settings.
#define TEST0    0x09  // Various test settings.
#define FIFOTHR  0x07  // RXFIFO and TXFIFO thresholds.
#define IOCFG2   0x29  // GDO2 output pin configuration.
#define IOCFG0D  0x06  // GDO0 output pin configuration. Refer to SmartRF Studio
User Manual for detailed pseudo register explanation.
#define PKTCTRL1 0x0C  // Packet automation control.
#define PKTCTRL0 0x05  // Packet automation control.
#define ADDR     0x00  // Device address.
#define PKTLEN   0xFF  // Packet length.
```

Figura 2.28. Codice estratto da SmarfRF®

2.4.2 Vista dettagliata dei registri

In questo paragrafo vengono presentati i registri di configurazione del CC1101, necessari al conseguimento della comunicazione wireless. In una prima fase si elencheranno quelli configurati col software *SmartRF*, e in seguito si presenteranno quelli configurati manualmente.

2.4.2.1 Registri della radio

I seguenti registri sono stati configurati con software *SmartRF:*

- CHANNR, *Channel Number*, che seleziona un canale nella banda di lavoro incrementando la frequenza del VCO di valori discreti (impostati in MDMCFG). Questo registro viene lasciato a 0x00
- FSCTRL, *Frequency Synthesizer Control*, 2 *bytes* che impostano la frequenza intermedia e un eventuale *offset* da aggiungere alla frequenza di battimento. La frequenza intermedia è cosi posta a 304,7 KHz
- FREQ, *Frequency Control Word*, 3 *bytes* (0x21656A) che impostano la frequenza della portante a 868,3 MHz
- MDMCFG, *Modem Configuration*, con 5 *bytes* imposta la larghezza del filtro in ricezione a 541,6 KHz (variando la frequenza di decimazione dei campioni dell'ADC), il *data rate* a 249,9 Kbps, il tipo di modulazione a GFSK, l'estensione delle parole di sincronismo a 4 *bytes* (i due *bytes* vengono ripetuti), il numero di *bytes* minimo di preambolo a 4 ed infine il *channel spacing* col quale si divide la banda in più canali di 199,9 KHz (salti discreti di CHANNR)
- DEVIATN, *Modem Deviation Setting*, registro che fissa la deviazione di frequenza a 126,9 KHz per la nostra modulazione GFSK
- FOCCFG, *Frequency Offset Compensation Configuration*, imposta il PLL per una compensazione automatica dell'errore di battimento
- BSCFG, *Bit Synchronisation Configuration*, imposta il PLL per il *clock recovery* dalle parole di sincronismo

- AGCCTRL, *AGC Control*, 3 *bytes* per impostare il guadagno dell'LNA a seconda dei casi per ottimizzare automaticamente la sensibilità del ricevitore. Questi fissano anche la soglia minima per il *Carrier Sensing.*

Nella seguente Figura 2.29 viene mostrata la configuzione finale della radio, evidenziando le caratteristiche del tramettitore a sintetizzazione diretta ed il ricevitore col segnale in banda intermedia.

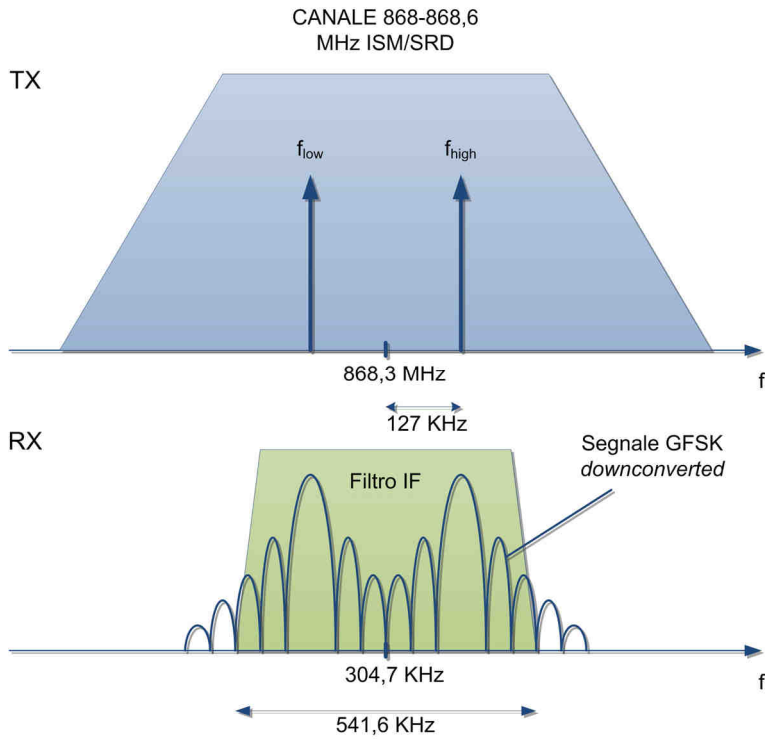


Figura 2.29. Configurazione della radio

2.4.2.2 Registri di controllo dei pacchetti

Il software *SmartRF® Studio* non ci ha permesso di configurare il *transceiver* in modo da gestire dei pacchetti come deciso nel progetto (Paragrafo 2.3.2). Per ovviare a questo inconveniente si deve reperire quelli da modificare, e riferendosi alle note del costruttore (*datasheet*), assegnare i valori giusti. Questi registri sono:

- PKTLEN, *Packet Length*, il cui contenuto precedente era di 0xFF che corrisponde ad una lunghezza massima del pacchetto di 255 bytes. Questo valore verrà portato a 0x05
- PKTCTRL, *Packet Automation Control*, che con 2 *bytes* imposta vari parametri del Packet Handler. In primo luogo si impostano i bit di controllo indirizzo a 0, disabilitando l'opzione. Poi si imposta il tipo di pacchetti a fisso ponendo i 2 bit di lunghezza a 0. Il CRC viene abilitato assieme all'*autoflush* di eventuali pacchetti corrotti. Infine i *bytes* di stato non vengono appesi in quanto il microcontrollore non ne farà uso.

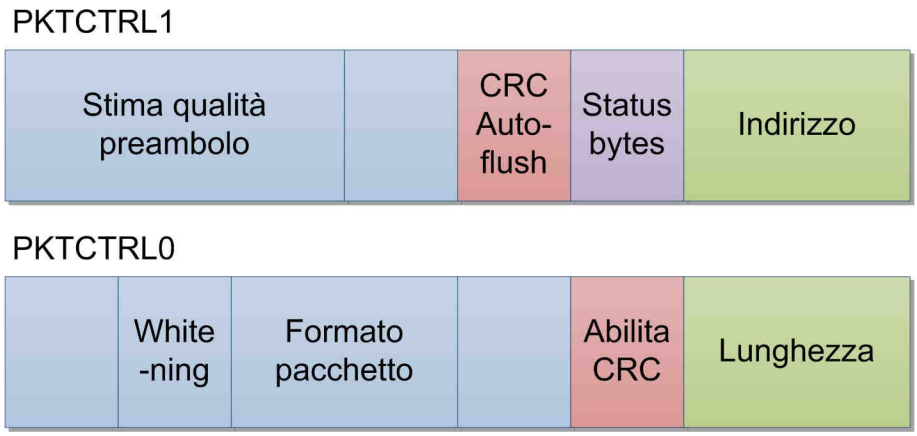


Figura 2.30. I registri di configurazione dei pacchetti

Si scriverà dunque 0x08 e 0x04 rispettivamente in PKTCTRL1 e PKTCTRL0.

2.4.2.3 Registro per la potenza in trasmissione

Per selezionare il guadagno del PA è necessario accedere al registro PATABLE. Con valori da 0x03 a 0xC2 si riesce ad impostare la potenza fornita all'antenna da -30 a 10 dBm. Per ottenere il valore di 0 dBm verrà immesso nel registro il valore 0x50.

2.4.2.4 Registro di controllo automatico dello stato

I registri MCSM, *Main Radio Control State Machine Configuration*, permettono di controllare ad hoc gli automatismi della radio. Questi ultimi permettono ad esempio al transceiver di interrompere una ricezione in funzione della qualità del segnale RF stimata e quella del preambolo. Si impostano inoltre una soglia di risveglio del ricevitore posto in modalità *Wake On Radio* ed un'altra per il *Clear Channel Assessment* prima di trasmettere.

La configurazione che interessa il progetto è quella dell'uscita automatica dagli stati di trasmissione o di ricezione. Il *transceiver* va immediatamente in IDLE, stato in cui le correnti assorbite vengono portate a livelli molto bassi. Inoltre si deve cambiare il contenuto di MCSM0 in modo da evitare che venga effettuata la calibrazione del PLL ad ogni passaggio da IDLE a TX o RX: il sintetizzatore di frequenza verrà calibrato una sola volta all'accensione del Modem RF, operazione necessaria al funzionamento del *transceiver*. Se le condizioni operative del *transceiver* dovessero cambiare, ossia se almeno una delle tensione di alimentazione, temperatura, e frequenza del sintetizzatore di frequenza dovessero variare considerevolmente[7], allora la calibrazione del PLL dovrebbe essere ripetuta.

2.5 Programma del Microcontrollore

In questo paragrafo vedremo come opera il microcontrollore nella gestione del Modem RF. Verrà mostrato con un diagramma di flusso operativo i passi di configurazione e di gestione dei dati.

2.5.1 Configurazione del microcontrollore

La configurazione del microcontrollore comprende:

[7] Rispettivamente fino a 0,5 V , 40 ℃ e 1 MHz di va riazioni

- L’impostazione dell’oscillatore interno a 16 MHz,
- L’abilitazione del modulo EUSART in modalità UART, l’impostazione del *Baud Rate Generator* a 38,4 Kbps e l’abilitazione delle interruzioni legate al modulo,
- L’attivazione del modulo MSSP per l’SPI, impostandone il *clock* a 4 MHz, e l’assegnazione della linea in uscita CSn,
- Infine l’assegnazione delle altre linee al CC1101 e dei LED.

Per poter svolgere la sua funzione di interfaccia (paragrafo 2.3.1), al microcontrollore rimane di implementare i FIFO che serviranno come buffer per i pacchetti radio.

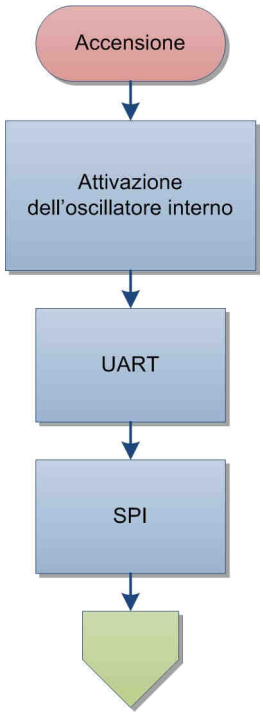


Figura 2.31. Configurazione del microcontrollore all'accensione.

2.5.2 Configurazione del *Transceiver*

In questa fase il microcontrollore effettua la scrittura dei valori di progetto nei registri di configurazione del *transceiver*. Questi, nel numero di 47, verranno acceduti in modalità *burst*, riducendo così i tempi di scrittura.

Dopo l'impostazione del *transceiver* è necessario effettuare la calibrazione del PLL. Questa viene fatta inviando il comando SCAL. A fine operazione lo stato della radio si pone automaticamente in IDLE.

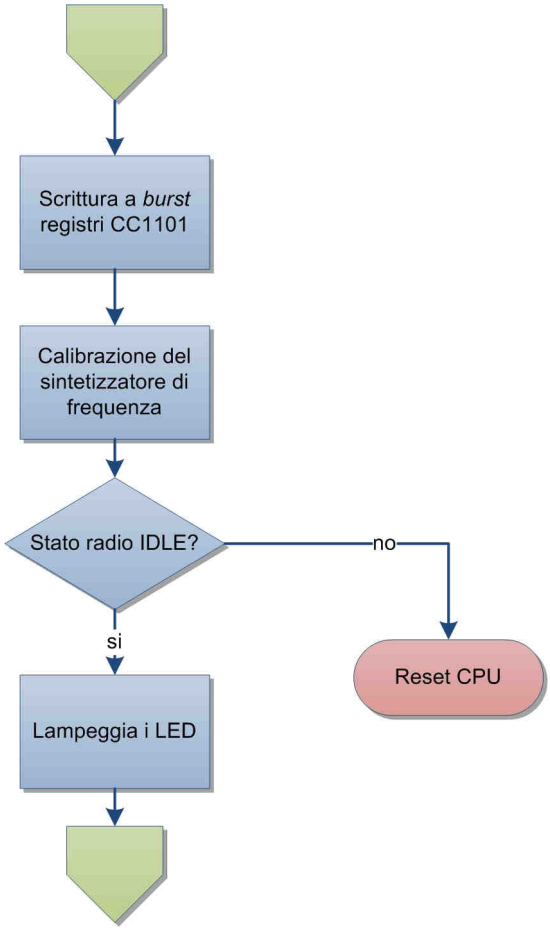


Figura 2.32. Configurazione del CC1101

2.5.3 Ciclo di lavoro

Il microcontrollore entra adesso in un ciclo di esecuzione infinito, durante il quale gestisce lo stato della radio in funzione dei pacchetti in arrivo dall'RS232. Lo stato è prevalentemente in ricezione, e va in trasmissione appena viene ricevuto un pacchetto dall'RS232.

Il prelievo in RXFIFO viene fatto soltanto se il controllo CRC è andato a buon fine, immettendo il pacchetto nella pila di ricezione. Segue poi una scrittura del pacchetto in RS232, controllando opportunamente il numero di *bytes* da stampare in funzione del *token*.

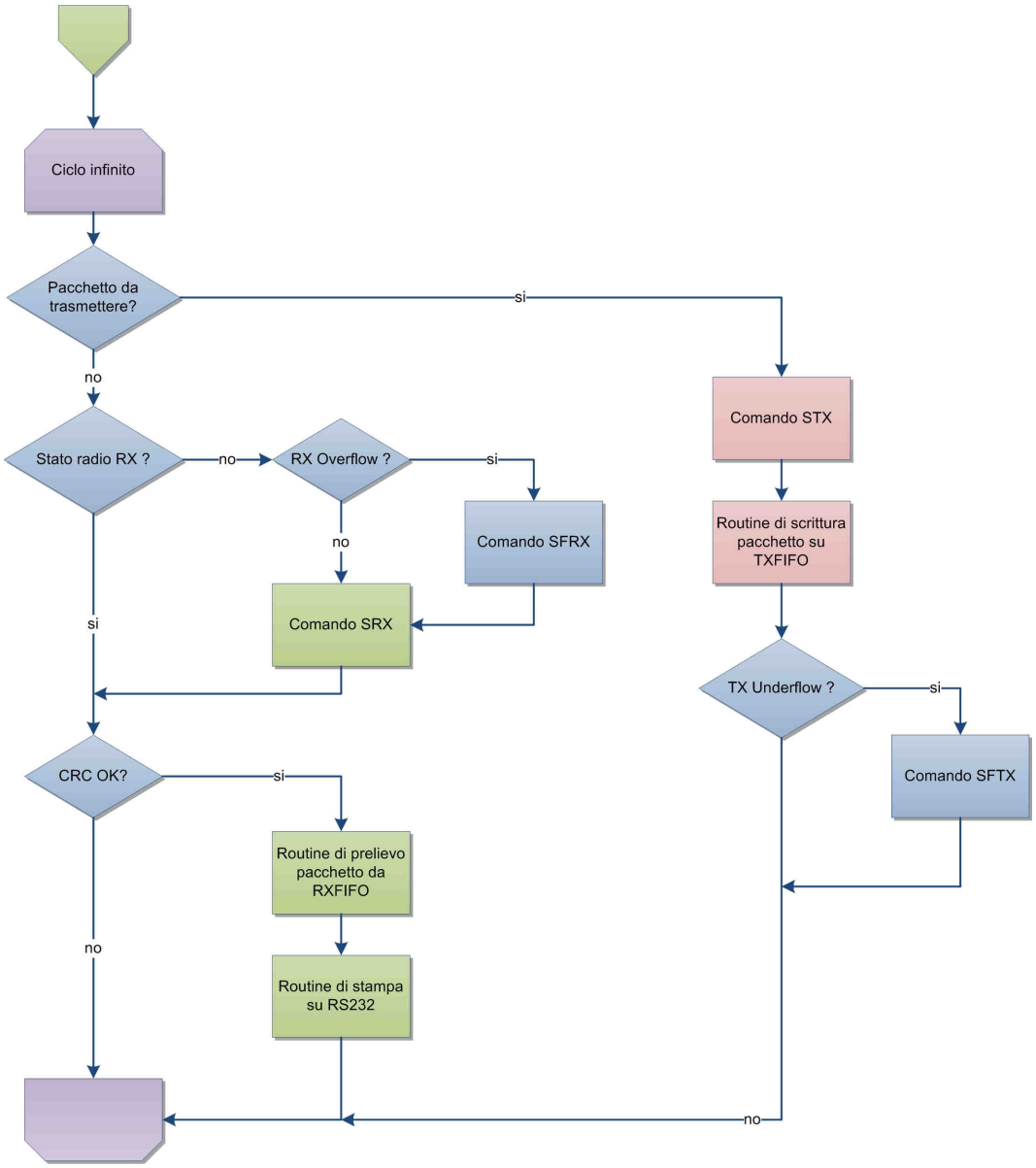


Figura 2.33. Ciclo di lavoro

Un pacchetto viene ricevuto dall'RS232 attraverso l'interruzione di esecuzione del flusso principale, e richiamando una routine di servizio (ISR) colla quale il pacchetto viene immesso nella pila di trasmissione. Viene dunque abilitato un *flag* che permette di entrare in trasmissione nel ciclo successivo di esecuzione.

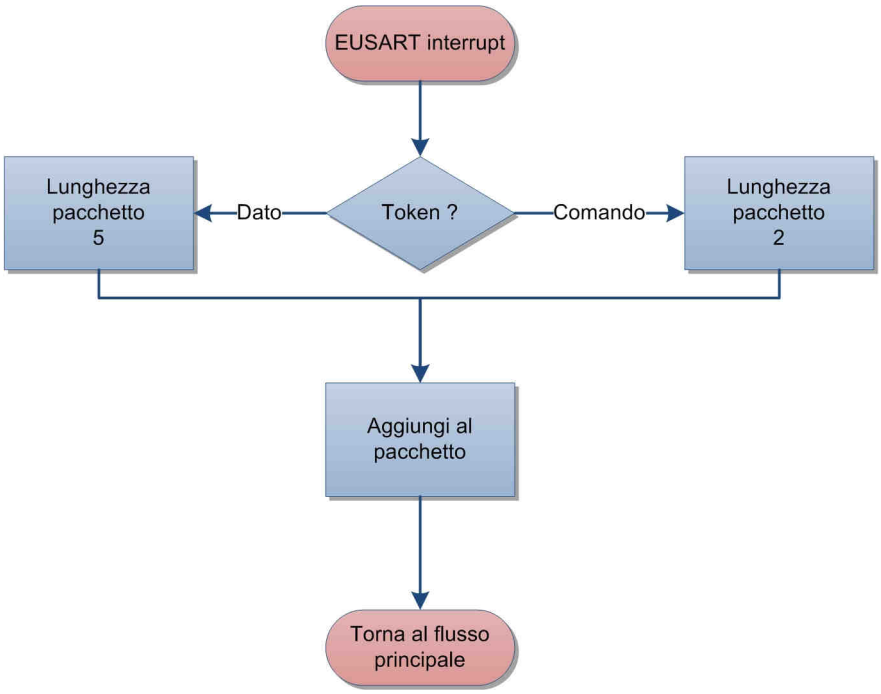


Figura 2.34. Routine di acquisizione RS232

CAPITOLO 3 :
REALIZZAZIONE DEL MODEM

In quest'ultimo capitolo verrà mostrato il Modem RF costruito a fine progetto. Questo è stato poi messo in funzione, dimostrando di realizzare il collegamento wireless come da aspettative. Sono in seguito state eseguite delle prove di copertura, e, dopo un *link budget*, si è ridotta la potenza in trasmissione.

3.1 Le schede di montaggio

Il Modem RF progettato è stato realizzato su un circuito stampato doppia faccia vista la necessità di combinare componenti passanti posti sulla faccia superiore e componenti a montaggio superficiale posti su entrambi i lati della scheda.

Le seguenti figure Figura 3.1 e Figura 3.2 illustrano rispettivamente il lato superiore e quello inferiore delle schede realizzate.

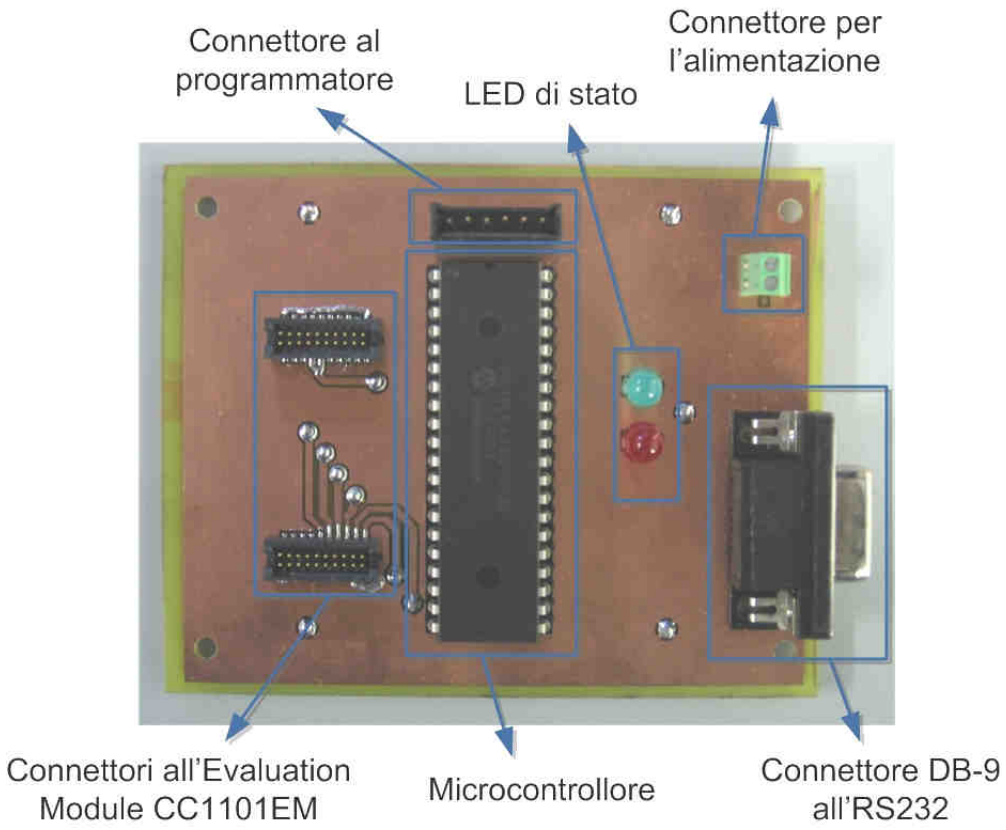


Figura 3.1. Lato superiore della scheda

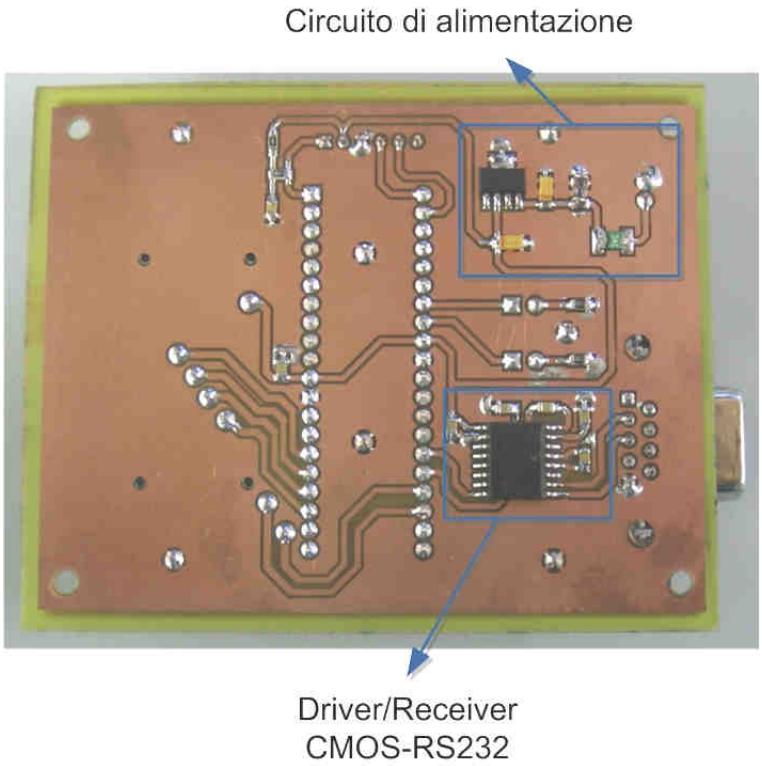


Figura 3.2. Lato inferiore della scheda

Figura 3.3. Prototipo del Modem RF

Una volta programmato il microcontrollore, il prototipo ha dimostrato di funzionare come da specifiche. Infatti, ciascuno dei due Modem RF costruiti è stato arbitrariamente (sono intercambiabili) collegato ad una parte del sistema, realizzando così una comunicazione trasparente tra il radar e lo schermo di controllo.

3.2 Raggio di copertura

Il raggio di copertura è stato valutato impostando il guadagno dell'amplificatore di potenza alla potenza di 8,5 dBm. Il collegamento è risultato persistente fino ad una cinquantina di metri in un ambiente fortemente affetto da riflessioni multiple. Queste distanze non verranno mai raggiunte durante l'impiego del radar e perciò si è deciso

di ridurre la potenza di trasmissione. La scelta della potenza in trasmissione viene fatta in base alla seguente formula di FRIIS:

$$P_R = P_T \, G_R \, G_T \left(\frac{\lambda}{4\pi R} \right)^2 \qquad [W]$$

dove:

- P_R è la potenza ricevuta ai morsetti dell'antenna,
- P_T la potenza irradiata in trasmissione,
- G_R il guadagno dell'antenna in ricezione,
- G_T il guadagno dell'antenna in trasmissione,
- λ è la lunghezza d'onda della radiazione elettromagnetica,
- R la distanza tra il trasmettitore e il ricevitore

I guadagni G_R e G_T sono di circa 3,5 dB per i monopoli che si adoperano, guadagno riferito al radiatore isotropo, antenna ideale sulla quale si basa la formula di FRIIS. λ è la lunghezza d'onda corrispondente alla frequenza f della portante, ossia c/f con c la velocità della luce. A 868 MHz, la lunghezza d'onda è di circa 34,5 cm.

Vista la nostra necessità di ricevere dati entro 20 m dal radar ed essendo la sensibilità del transceiver in ricezione di circa -95 dBm (a 250 kbps), la potenza in trasmissione può esse ridotta a -30 dBm. Infatti, a 20 m, la potenza stimata[1] è di circa -80 dBm sufficienti per la demodulazione. Verrà dunque immesso il valore 0x01 nel registro di configurazione del PA. In Figura 3.4 viene illustrato l'andamento della potenza ricevuta in funzione della distanza relativa.

[1] In effetti, la formula di FRIIS vale per un trasmettitore ed un ricevitore posti nello spazio libero, privo di disturbatori del campo elettromagnetico (riflessioni multiple)

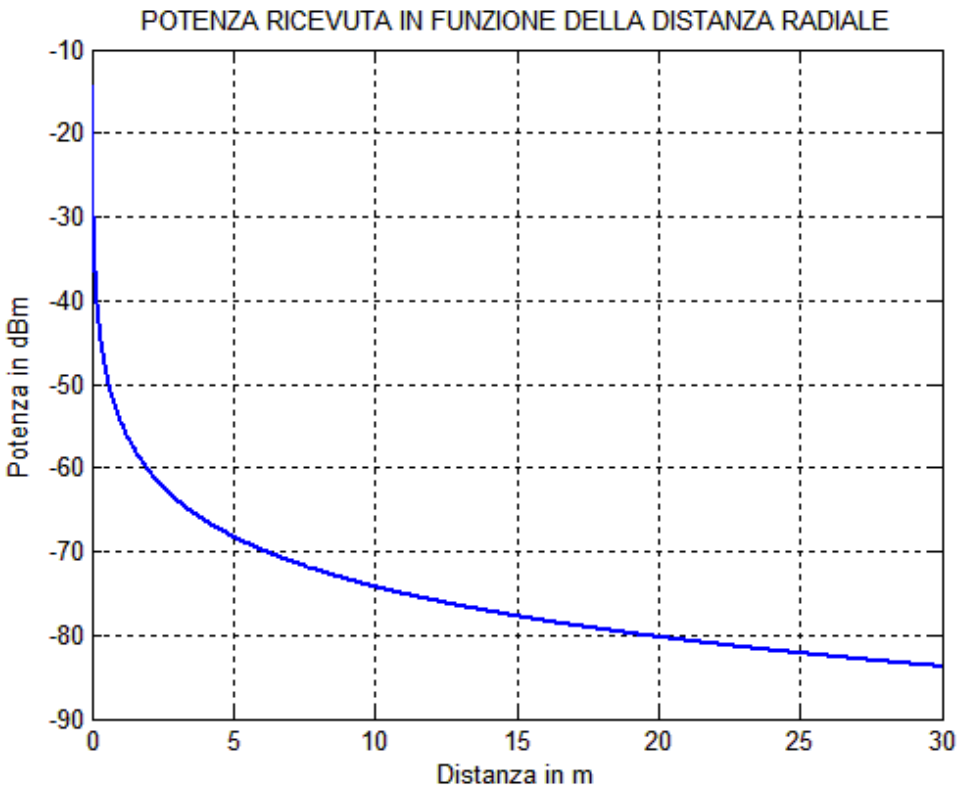


Figura 3.4. Stima della potenza ricevuta

CONCLUSIONE

È stato progettato ed in seguito realizzato un collegamento wireless tra un unità radar GPR e il suo schermo di controllo. Durante la lunga fase di progettazione, è stato necessario studiare la comunicazione già presente nel sistema per poi progettare la soluzione sostitutiva. Quest'ultima ha dimostrato di raggiungere le prestazioni richieste dalle specifiche, offrendo i vantaggi di una comunicazione radio ad un'interfaccia RS232.

Integrando i moduli realizzati al sistema radar, la rilevazione di eventuali superstiti in caso di catastrofi è facilitata poiché vengono meno i disturbi legati alla comunicazione cablata.

Una migliore soluzione al problema trattato potrebbe trovarsi nella miniaturizzazione del Modem RF utilizzando il *System-on-Chip* CC1111, altro integrato della *Texas Instruments* che, oltre ad avere la medesima radio del CC1101 (accorciando quindi i tempi di riprogettazione), è provvisto di un microcontrollore (di architettura 8051) sullo stesso *chip*. Le dimensioni del Modem RF verrebbero ridotte a quelle dell'*Evaluation Module*.

BIBLIOGRAFIA

[1] M. PIERACCINI, G LUZI, D. DEI, L. PIERI, C. ATZENI, *Detection of Breathing and Heartbeat Through Snow Using a Microwave Transceiver*, IEEE Transactions 2008

[2] D. LEROY, *RS-232 Bus Description and EIA232 Pinout,* http://www.interfacebus.com/Design_Connector_RS232.html

[3] Wikipedia article, *RS-232*, http://en.wikipedia.org/wiki/RS-232

[4] C.E. STRANGIO, *The RS232 Standard*, http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html

[5] Commissione Europea*, Guide for the EMC Directive 2004/108/EC,* http://ec.europa.eu/enterprise/electr_equipment/emc/guides/emcguide_may2007.pdf

[6] ETSI EN 300 220-1 V2.1.1 (2006-04), *Electromagnetic compatibility and Radio spectrum Matters (ERM), Short Range Devices (SRD);Radio equipment to be used in the 25 MHz to 1 000 MHz frequency range with power levels ranging up to 500 mW;Part 1: Technical characteristics and test methods,* http://www.etsi.org/WebSite/Standards/StandardsDownload.aspx

[7] ERC RECOMMENDATION 70-03, *RELATING TO THE USE OF SHORT RANGE DEVICES (SRD),* http://www.radiometrix.com/pdf/ERC70-03.pdf

[8] TEXAS INSTRUMENTS, *CC1101 Documentation*, http://focus.ti.com/docs/prod/folders/print/cc1101.html

[9] A. Aktas, M. Ismail, *CMOS PLL Calibration Techniques*, IEEE CIRCUITS & DEVICES MAGAZINE, SEPTEMBER/OCTOBER 2004

[10] B. Watson, *FSK: Signals and Demodulation*, WJ Communications Tech Note 2001, http://www.wj.com/documents/Tech_Notes_Archived/FSK_signals_demod.pdf

[11] C. LANGTON, Intuitive Guides to Principles of Communications, Feb 2002, http://www.complextoreal.com/chapters/fm.pdf

[12] G. BUCCI, Architetture dei calcolatori elettronici, McGraw-Hill 2001

[13] MICROCHIP, *PIC18F4620 Documentation,* http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1335&dDocName=en010304

[14] MICROCHIP, *MPLAB Integrated Development Environment,* http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002

[15] N. Gardner, *PICmicro MCU C® An introduction to programming The Microchip PIC in CCS C,* Bluebird Electronics 2002.