

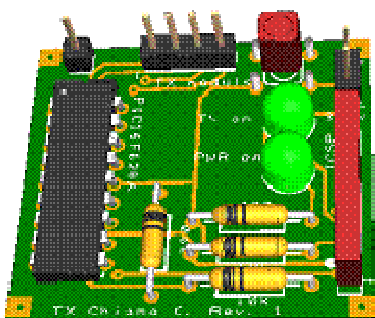
Electronica e Microcontrollori

Costruiamo un trasmettitore per radiocomando a 433MHz con un PIC16F628A e pochi altri componenti.

Autore: Antonio d'Abbieri

Data progetto: 10.05.2011

Microcontrollore: PIC16F628A



Esistono in commercio moduli ibridi di trasmettitori e ricevitori UHF a 433 MHz davvero molto economici con i quali si possono costruire radiocomandi e circuiti per demotica molto interessanti.

In questo articolo descriviamo un semplice TX piccolo ed affidabile che utilizza il ben noto microcontrollore PIC16F628A ed il modulo QAM-TX1 della Quasar UK.

La caratteristica più interessante di questo progetto consiste nell'aver usato componenti di basso costo, infatti il modulo ibrido del TX costa circa 5 euro ma ne esistono altri dal prezzo di oltre 21 euro mentre i moduli RX arrivano addirittura a 40 euro e più. In cosa differiscono? Nel fatto che quelli più costosi contengono anche una logica interna che implementa il protocollo di trasmissione, cioè il software che permette di codificare il segnale da trasmettere in modo opportuno prima di inviarlo al ricevitore, il quale provvederà a decodificarlo e restituirlo in uscita il più possibile "pulito".

Tutto questo processo richiede la presenza di un microcontrollore a bordo del modulo ibrido (uno sul TX ed uno sull'RX) e di un software che svolga l'elaborazione del segnale, quindi è inevitabilmente un deciso aumento dei costi.

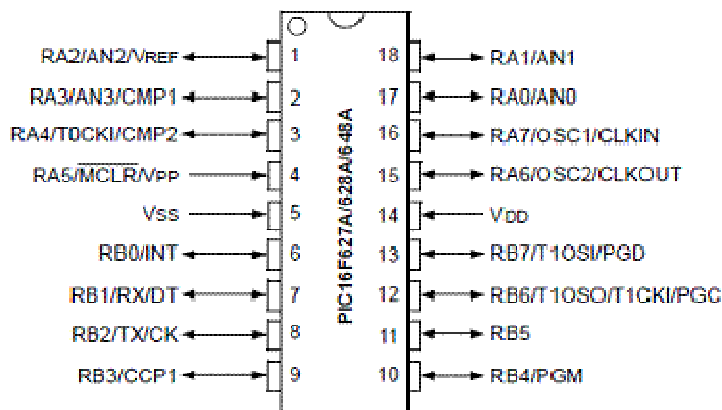
In questo progetto, invece, si è voluto ridurre il costo al minimo, pertanto è stato sviluppato un software in linguaggio C che implementa il protocollo di trasmissione ed allo stesso tempo provvede alle funzioni applicative di un TX per radiocomando. Ciò ha permesso di usare moduli molto economici e di trasformare la complessità hardware in complessità software col vantaggio che il software, una volta sviluppato, può essere inserito in centinaia di trasmettitori a costo zero, realizzando forti economie.

C'è anche un altro vantaggio! Tutta l'intelligenza del circuito è racchiusa nel software quindi, una volta compresa la logica che permette di eseguire una trasmissione sicura ed affidabile, possiamo modificarla ed adattarla alle nostre esigenze cambiando pochissime istruzioni e realizzando circuiti, anche molto complessi, senza difficoltà.

I Componenti e gli Strumenti di lavoro

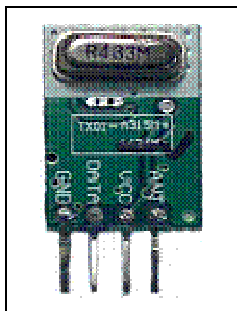
Il Microcontrollore

Per la realizzazione di questo progetto è stato scelto il famoso microcontrollore PIC16F628A. Si tratta di un componente molto economico dotato di numerose funzioni che lo rendono adatto per svariate applicazioni, ed infine è molto facile da reperire in commercio.

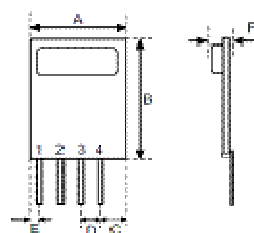


Il Modulo RF

Il modulo scelto è il **QAM-TX1** della **QUASAR UK**. Si tratta di un modulo molto economico e compatto. Poiché tutta la gestione del segnale viene svolta dal nostro software non occorrono moduli più costosi con logica di controllo integrata.



Technical Specifications



Features

- Complete RF Transmitter
- Transmit Range Up To 50m
- CMOS / TTL Input
- No Adjustable Components
- Very Stable Operating Frequency
- Low Current Consumption (Typ 11mA)
- Wide Operating Voltage (1.5-5v)
- ASK Modulation
- Available as 315 or 433 MHz



Il Sistema di sviluppo del software

Il programma per il microcontrollore è stato scritto in linguaggio C usando il sistema di sviluppo **mikroC PRO for PIC** di MikroElektronika, versione 4.60.

Questo prodotto è molto valido, è gratuito ed è liberamente scaricabile da internet. La versione gratuita ha solo la limitazione a 2 Kb della dimensione del programma sorgente, ma questo è un limite difficile da superare per gran parte delle applicazioni hobbistiche.

E' consigliabile scaricare da internet la versione più aggiornata del programma e comunque non usare mai versioni precedenti alla 4.60, per via di alcuni errori nelle librerie, ora corretti.

Il sistema di sviluppo **mikroC PRO** integra il compilatore che crea il codice in linguaggio macchina per i microcontrollori PIC.

Il programmatore di PIC

Una volta creato il codice in linguaggio macchina occorre caricarlo nella memoria flash del microcontrollore. Ciò avviene attraverso un dispositivo chiamato "Programmatore per PIC". In commercio ne esistono molti tipi ed è possibile anche auto-costruirli.

Il programmatore prescelto è il **PicKit2** della Microchip, la stessa casa produttrice dei microcontrollori PIC. E' uno strumento ottimo per via di un valido progetto che ricava tutte le tensioni necessarie dai 5 vcc della porta USB, è protetto contro i cortocircuiti, è piccolo ed il suo prezzo è contenuto.

Il programmatore dispone a corredo di un software attraverso il quale si trasferisce il codice in linguaggio macchina all'interno del microcontrollore.

Descrizione del TX

Il piccolo TX che stiamo realizzando è dotato di un pulsante premendo il quale viene generato un codice inviato via radio. Il ricevitore riconosce il codice ed accende un LED oppure eccita un relé. Questa operazione deve avvenire con elevata affidabilità, senza errori o incertezze.

Un concetto importante

TX ed RX non sono connessi tramite un cavo, il segnale emesso dal TX viaggia nell'etere ed arriva al ricevitore misto a disturbi ed altri segnali radio che numerosissimi affollano la banda usata. Non possiamo quindi pensare che tutti i dati inviati dal modulo TX si ritrovino pari pari all'uscita del modulo RX. Questo è il primo e più importante punto su cui lavorare.

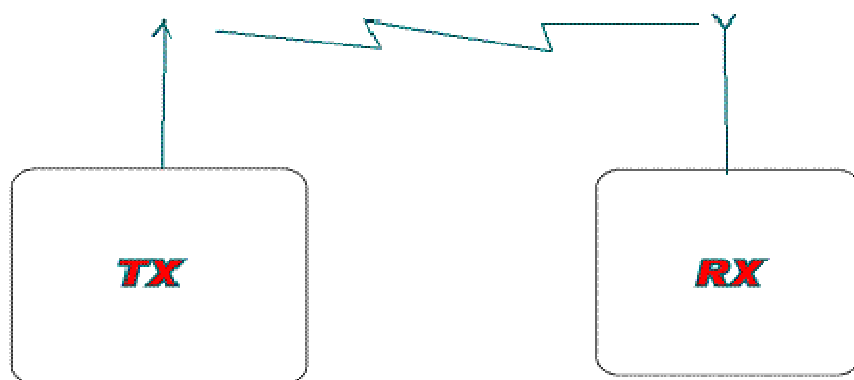
Il programma

Il cuore di tutto il sistema è il programma scritto in linguaggio C ed inserito nel microcontrollore dopo essere stato compilato. Questo software è diviso in due parti:

- ➡ il software di gestione del trasmettitore;
- ➡ il software di codifica del segnale.

Il software di gestione del trasmettitore provvede alle funzionalità del TX, ad esempio, rileva la pressione del pulsante, accende un LED, genera i codici da trasmettere, alimenta e pilota il modulo RF.

Il software di codifica del segnale elabora il codice numerico da trasmettere (il dato) in base ad un protocollo condiviso anche dal ricevitore, in modo che possa essere ricevuto eliminando i disturbi che inevitabilmente lo accompagnano.



La rappresentazione schematica che appare in figura e le informazioni contenute in alcuni datasheet inducono spesso in errore non evidenziando il problema del rumore e delle interferenze. Le due antenne non sono collegate da un filo ma sono immerse in una nuvola di segnali radio.

Se TX ed RX fossero collegati tramite un cavo allora i dati inviati giungerebbero puliti al ricevitore. Il collegamento radio comporta invece il deterioramento del dato trasmesso a causa di due fattori principali: i disturbi e le interferenze. I disturbi sono causati da scariche elettriche che coprono o alterano il debole segnale in arrivo impedendone la ricezione; le interferenze sono causate da altri segnali radio presenti nell'etere che, avendo la stessa frequenza del ricevitore, vengono accettati come validi dagli stadi di sintonia RF ma devono essere scartati perché non emessi dal TX.

Sorge evidente la doppia necessità di:

1. riconoscere i segnali validi separandoli dai disturbi (rumore);
2. rigettare i segnali estranei non rispondenti al protocollo usato.

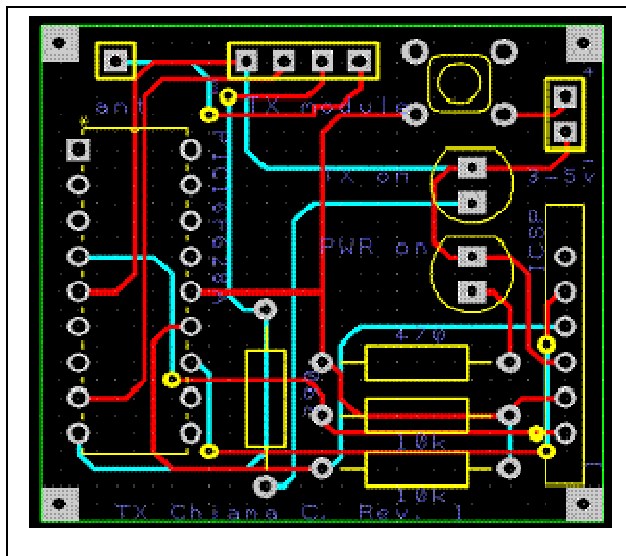
Tutto questo lavoro viene svolto dal software il quale consente di ottenere un funzionamento corretto, preciso, costante e sicuro. In altre parole quando si preme il pulsante sul TX si produce nel ricevitore la reazione prevista (accensione di un LED, attivazione di un relé) senza errori o esitazioni, quindi il sistema si presta a comandare a distanza dispositivi in tutta sicurezza.

Funzionamento

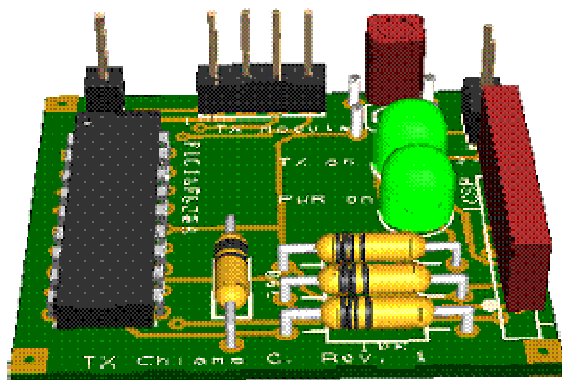
Per questa ragione il microcontrollore invia i dati solo per il tempo necessario lasciando immediatamente libero il canale.

Una seconda osservazione riguarda il modulo TX il quale genera una portante RF anche se non ci sono dati da trasmettere, è quindi necessario disalimentarlo dopo aver inviato i dati altrimenti disturberà gli altri dispositivi posti nelle vicinanze. Il microcontrollore provvede ad alimentare il modulo TX attraverso il pin 9.

Il circuito stampato



Ecco il circuito stampato del trasmettitore. Misura appena 39 x 39 mm. A sinistra si vede il PIC, in alto c'è il connettore per il modulo TX, a destra si nota il connettore per l'alimentazione (2 pin) e quello per il programmatore (6 pin). L'antenna si collega al connettore singolo posto in alto a sinistra.



Il programma

```
/* -----  
Trasmettitore per Comando a Distanza  
Sistema a doppio codice: numero utente - dato  
Autore: Antonio d'Abbieri  
Data: 10.05.2011
```

Nel sistema di sviluppo mikroC PRO impostare:

MCU: PIC16F628A
Clock: 4 MHz
Brown out detect: disabled
Oscillatore: interno
Librerie da usare: UART

Funzionamento:

Il TX invia il codice dell'utente seguito dal dato vero e proprio
La trasmissione dura poco più di 1 secondo poi si fa una pausa di 20 secondi per non saturare la banda.
I dati vengono trasmessi serialmente uno dopo l'altro con un intervallo di 10 ms. per non ingolfare il TX,
infatti senza questa pausa si generano codici anomali.
Il Led 1 si accende durante la trasmissione.
Il circuito è alimentato dando tensione con un interruttore (o pulsante) in modo che quando è spento non consuma nulla.
I codici numerici vanno da 1 a 254 perché si escludono gli estremi.
L'RX controlla il numero utente e, se questo è valido, legge il dato seguente.

```
-----*/
```

```
long i;  
int j;  
char utente, dato;  
void main() {  
    utente=1; // ##### codice utente che usa il sistema #####  
    dato=4; // ##### dato da inviare #####  
    CMCON = 7; // Disabilito i comparatori  
    INTCON = 0; // disabilito interrupt  
    trisb2_bit = 0; // portb2 = output per TX  
    trisb3_bit = 0; // portb3 = output per LED1  
    portb.f3 = 0; // spengo led1  
    UART1_Init(2400); // Inizializzo modulo UART a 9600 bps  
    Delay_ms(100); // attesa per stabilizzare il modulo UART  
    while (1) {  
        portb.f3=1; // accendo led1  
        // invio il codice utente  
        for(i=0;i<=5;i++) {  
            for(j=0; j<10; j++) {
```

```

UART1_Write(utente); // invio 50 volte il codice utente
delay_ms(10); // attesa per non ingolfare la UART
}
}
// invio il dato
for(i=0; i<=5; i++) {
for(j=0; j<10; j++) {
UART1_Write(dato); // invio 50 volte il dato
delay_ms(10); // attesa per non ingolfare la UART
}
}
portb.f3=0; // spengo il led 1
delay_ms(20000); // attendo 20 secondi per lasciare libero il canale
}
}

```

Descrizione

Il programma è ben documentato e non dovrebbero nascere problemi di comprensione.

Fare le seguenti impostazioni sul sistema di sviluppo mikroC:

- MCU: PIC16F628A
- Clock: 4 MHz
- Brown out detect: disabled
- Oscillatore: interno
- Librerie da usare: UART

Il protocollo usato

Come si vede dal programma il protocollo si basa sull'invio in sequenza di due valori numerici compresi tra 1 e 254:

1. il codice dell'utente;
2. il dato vero e proprio.

Il codice dell'utente serve ad escludere tutti i dati prodotti da dispositivi estranei come antifurti, apri cancello, radiocomandi eccetera.

Se il ricevitore non trova il codice dell'utente atteso, non considera validi i dati successivi. In caso contrario accetta il codice utente e si mette in attesa di ricevere il dato che segue il quale è certamente quello trasmesso dal TX.

Questo sistema a doppio codice non garantisce ancora un'elevata reiezione ai falsi allarmi, infatti dall'etere si ricevono in continuazione numeri di tutti i valori (rumore) ed è quindi molto probabile ricevere ben presto anche il codice dell'utente, che poi è un numero compreso tra 1 e 254. In questo caso il ricevitore prenderebbe per buono il primo numero che segue, considerandolo un dato valido.

Per rendere davvero affidabile il sistema si deve separare il rumore dai dati trasmessi dal TX. Il protocollo diventa allora ridondante e trasmette ogni numero (codice o dato) 50 volte di seguito. Si crea un treno di impulsi che arriva al ricevitore frammisto al rumore, ma mentre questo è formato da valori casuali che si ripetono raramente e presenta una bassa frequenza di coppie uguali, quelli generati dal TX presentano un'alta frequenza di coppie uguali

nell'intervallo di tempo esaminato. Allora basta valutare la frequenza di ripetizione di un dato per capire se è un dato valido o casuale.

Dopo aver così aggiornato il protocollo vediamo cosa accade in ricezione: il ricevitore scarta tutti i dati casuali finché non arriva il primo dato valido, cioè un treno di impulsi di lunghezza sufficiente e di valore corrispondente al codice utente atteso. Questa probabilità è già molto bassa. Poi il ricevitore si mette in attesa di un secondo dato valido, cioè un altro treno di impulsi di lunghezza sufficiente che però deve arrivare subito dopo. E' evidente che è molto bassa la probabilità di ricevere in rapida successione due sequenze numeriche valide (cioè di lunghezza adeguata) delle quali la prima corrisponde al codice utente atteso dal ricevitore.

Comunicazione seriale - il protocollo UART

Quello appena descritto è il metodo usato per generare un pacchetto di dati riconoscibile dal ricevitore. Questi dati vengono trasmessi in modo sequenziale usando il ben noto protocollo UART che viene usato in tutte le comunicazioni seriali anche via cavo.

Il sistema di sviluppo MikroC PRO mette a disposizione una libreria per sfruttare questo protocollo ed il microcontrollore PIC16F628A contiene al suo interno la logica che implementa in hardware la UART.

Il protocollo Manchester

Si legge spesso che il protocollo Manchester è il più adatto alla trasmissione di dati via radio, per questo motivo sono stati fatti numerosi tentativi usando la libreria disponibile nel sistema di sviluppo MikroC PRO. Purtroppo tutti i tentativi effettuati sono stati infruttuosi quindi il protocollo Manchester è stato abbandonato ed al suo posto è stato usato il protocollo UART che invece funziona benissimo. Il problema potrebbe forse dipendere da un difetto della libreria usata oppure dalla incompatibilità con i moduli ibridi usati.