Massimo Peino (mpeino)

CONTROLLARE UN SERVOCOMANDO A DISTANZA.

30 January 2012

Introduzione

Quest'articolo segue all'articolo "CONTROLLARE UN SERVOCOMANDO CON UN PIC". Secondo me vi conviene darci un occhiata in quanto ci sono alcuni punti in comune.

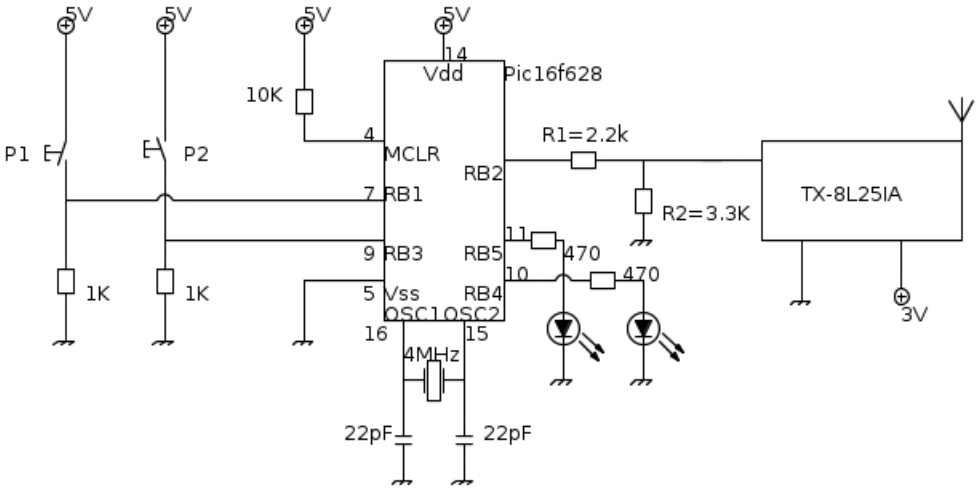
Descrizione generale

Il sistema è formato da un circuito trasmettitore e un circuito ricevitore.

Per entrambi ho utilizzato dei pic e precisamente il **pic 16f628** per il trasmettitore e il **pic 16f876** per il ricevitore (la scelta è casuale, sono quelli che aveva il mio negoziante).

Per il settore radio ho usato dei moduli della Aurel economici ma efficienti.

Schema del trasmettitore



Come si può vedere dallo schema il circuito trasmettitore è il modulo Aurel TX-8L25IA. Questo e un trasmettitore SAW con antenna dedicata che ha una potenza

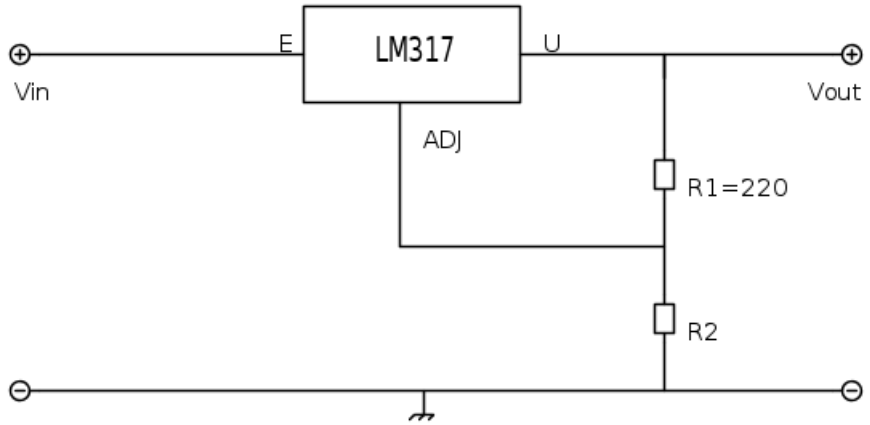
di irradiazione di 25mW. La frequenza di trasmissione è di 868.3MHz, ho scelto questo modulo e non quelli di 430MHz perchè ho pensato che fosse un po più immune ai disturbi.

Il micro funziona a 5V, mentre il modulo a 3V per questa ragione viene utilizzato un partitore resistivo R1-R2 che abbassa a 3V il segnale di 5V proveniente dalla linea di trasmissione del PIC.

Per il calcolo delle resistenze ho usato la seguente formula:

$$V_{out} = V_{in}\left(\frac{R_2}{R_1 + R_2}\right) = 5\left(\frac{3.3\,\mathrm{k\Omega}}{2.2\,\mathrm{k\Omega} + 3.3\,\mathrm{k\Omega}}\right) = 3\,\mathrm{V}$$

Per quanto riguarda l'alimentazione del modulo Aurel avrei voluto utilizzare un MCP1702-3002E un regolatore nella versione a 3 V, ma non avendolo trovato dal mio negoziante ho dovuto ripiegare sul famosissimo LM317.



E con la seguente formula ho calcolato la resistenza R2 per ottenere una tensione di 3 V.

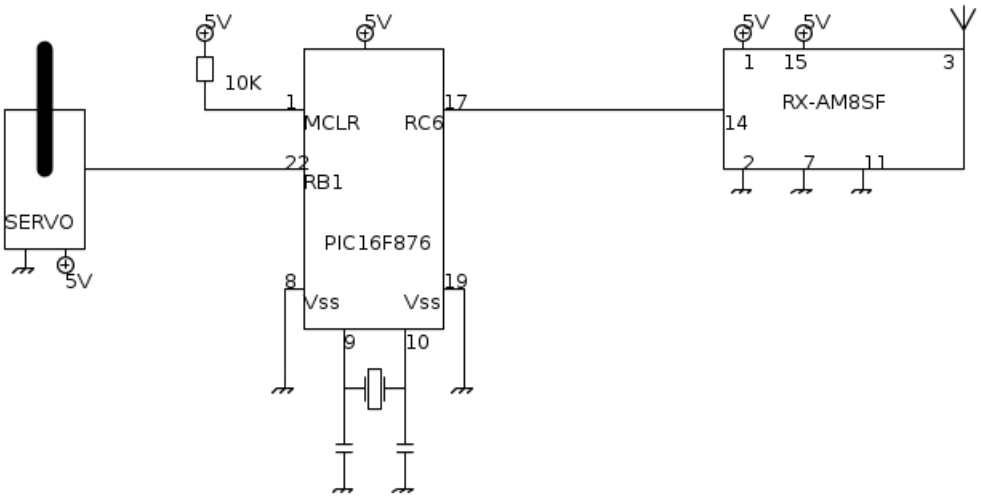
$$R_2 = \left(\frac{V_{out}}{1.25} - 1\right)220 = \left(\frac{3}{1.25} - 1\right)220 = 308\,\Omega$$

Non essendoci in commercio una resistenza da 300Ω ho optato per due da 150 collegate in serie. E' importante dire anche che la tensione in ingresso deve essere almeno di 3V superiore a quella di uscita.

Il pic è collegato al modulo tramite il pin RB2 che è abilitato alla trasmissione dei dati tramite periferica USART.

I due led servono solo a indicare che si sta andando in trasmissione.

Schema del ricevitore



Lo schema del ricevitore è molto piu' semplice in quanto tutti i componenti sono alimentati a 5V.

Il pic è collegato al modulo tramite il pin RC7 che è abilitato alla ricezione dei dati tramite periferica USART.

Con il pin 11 del modulo Aurel si seleziona il livello di sensibilità.
Livello logico 0: -109dBm.
Livello logico 1: -90dBm.

Una veloce panoramica sulla periferica USART

Per fare comunicare il circuito trasmettitore con il ricevitore ho utilizzato la periferica USART (Syncronous Asyncronous Receveir Transmitter).

E' una periferica hardware in grado di trasmettere e ricevere informazioni ad una velocità denominata Baud Rate espressa in bit per secondo (bps).

La periferica può essere configurata in tre modi differenti:

- modalità asincrona;
- modalità sincrona master;
- modalità sincrona slave;

Per il nostro progetto andremo a utilizzare la modalità asincrona.
La periferica è in grado di trasmettere e ricevere dati attraverso due pin che per il PIC16f876 sono RC7/RX e RC6/TX mentre per il PIC16f628 sono RB2/TX e RB1/RX.

I dati che gestisce questa periferica sono formati da un bit di START, otto bit di dati e un bit di STOP.

Come per l'altro progetto ho usato il compilatore MikroC PRO il quale ci mette a disposizione delle librerie che semplificano molto il nostro lavoro.
Per il loro uso bisogna selezionarle attraverso lo strumento Library Manager. Una volta aperta la finestra bisogna mettere una una spunta su UART. UART e non USART perchè queste funzioni supportano solo la comunicazione asincrona e non sincrona.
Analizzeremo adesso solo le funzioni che ho utilizzato per il progetto:

prototipo: void UART1_Init(unsigned long baud_rate);
Questa funzione inizializza la periferica UART con la velocità di trasmissione desiderata. ES: UART1_Init(1000); In questo caso la periferica è stata inizializzata con una velocità di 1000 bps.

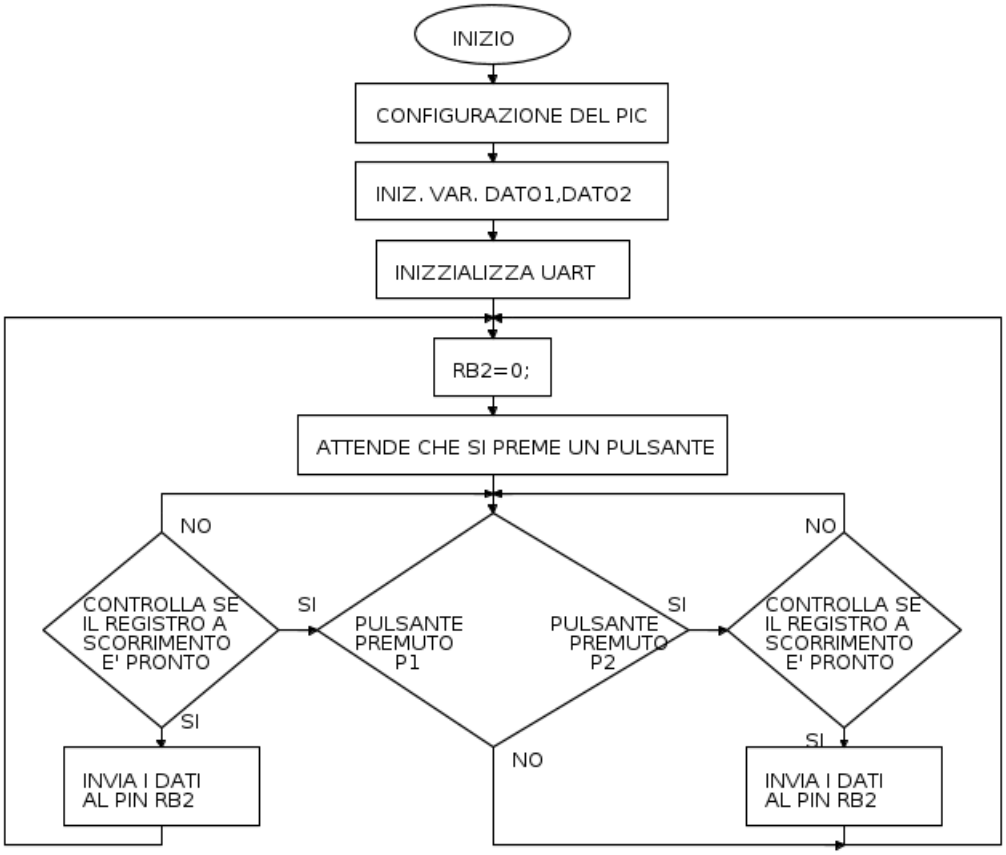
prototipo: char UART1_Tx_Idle();
Ritorna 1 se ci sono dati pronti per la lettura, 0 se non ce ne sono.Questa funzione serve per verificare se il registro a scorrimento di trasmissione è vuoto o no.

Prototipo: void UART1_Write(char_data);
Questa funzione trasmette un byte attraverso il modulo UART.

Prototipo: char UART1_Data_Ready();
Questa funzione serve per verificare se ci sono dati per la lettura. Ritorna 1 se i dati sono pronti per la lettura 0 se non ci sono dati.

Prototipo: char UART1_Read();
Restituisce il byte ricevuto.

Flow chart del trasmettitore



Penso che il flow chart non ha bisogno di ulteriori commenti.

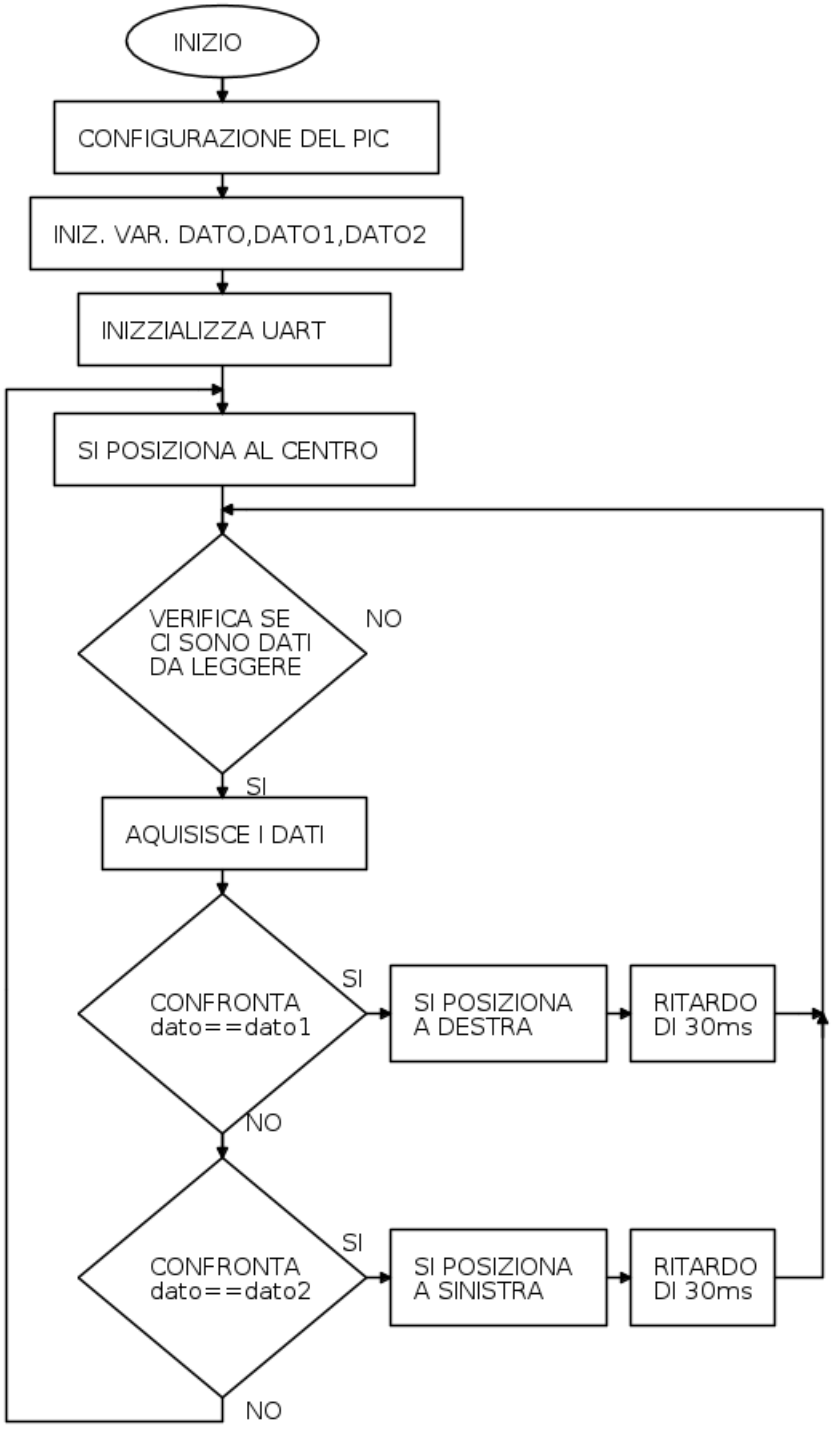
Codice sorgente del trasmettitore

```
char dato1 =0B11110000 ;                    //inizializzazione variabili
char dato2 =0B00001111;                     // dato1,dato2

configura();

UART1_init(1000);                           // inizializzazione modulo UART

while(1) {
        RB4_bit=1;                          //servono solo per controllare che
        RB5_bit=1;                          //che il pic stia trasmettendo
                                            //(vedi schema)

        RB2_bit=0;                          //metto a 0 il pin TX
```

```
while (RB1_bit=0){                          // controlla se il pulsante P1 è prem
           if(RB1_bit=1){
             delay_ms(50);                  // procedura antirimbalzo
              if(RB1_bit=1) {
                 if (UART1_Tx_Idle ()==1){  //controlla se il registro di trasmis
                                            //è pronto
                      UART1_Write (dato1);  //invia dato1 al pin RB2/TX
                      RB4_bit=0;
                                        }
                           }
                     }
                 }

while(RB3_bit=0){                           // controlla se il pulsante P2 è prem
          if(RB3_bit=1){
             delay_ms(50);                  // procedura antirimbalzo
              if(RB3_bit=1) {
                 if (UART1_Tx_Idle ()==1){  //controlla se il registro di trasmis
                                            //è pronto
                      UART1_Write (dato2);  //invia dato2 al pin RB2/TX
                      RB5_bit=0;
                                        }
                            }
                        }
                  }
```

Questo è il cuore del codice.

Flow chart del ricevitore



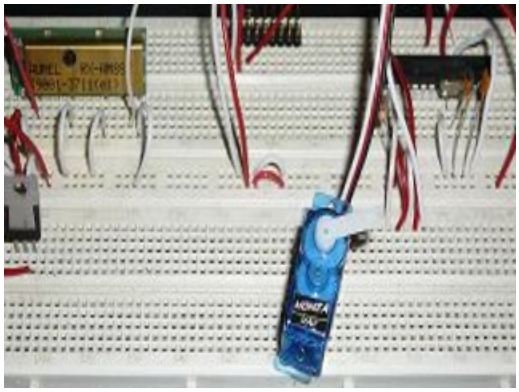
Vorrei fare una premessa. Il fatto che il servo inizialmente si posizioni al centro è una mia scelta personale in quanto questo progetto fa parte di un progetto più grande.
Il ritardo di 30ms è necessario in quanto il servo raggiunge la posizione desiderata

in un determinato tempo.Se non ci fosse quel ritardo il servo oscillerebbe tra la posizione centrale e quella che si intende raggiungere (destra o sinistra) continuamente senza mai raggiungerla.

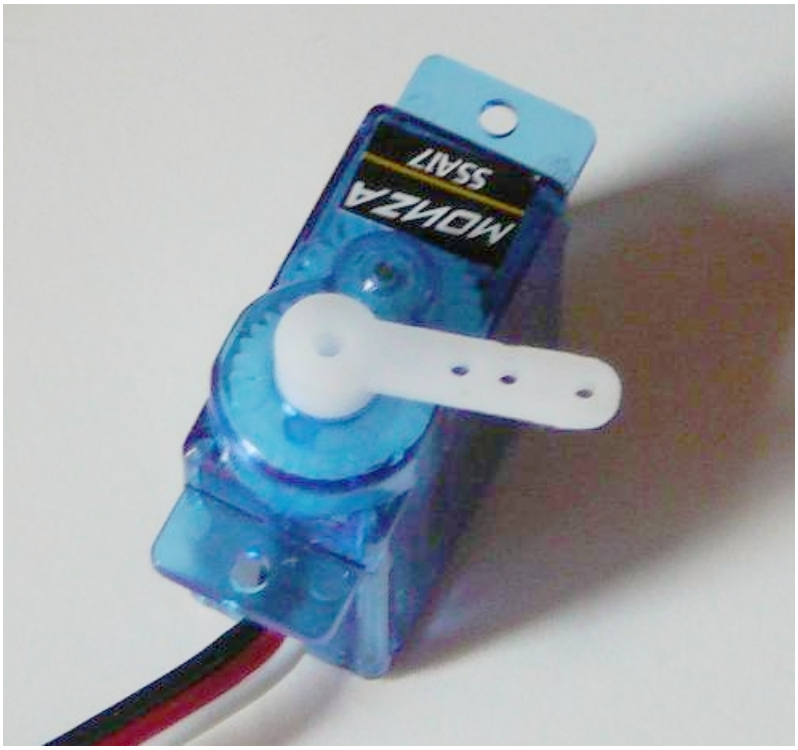
Codice sorgente del ricevitore

```
char dato1=0B11110000;                  //inizializzazioni variabili
char dato2=0B00001111;                  //dato1,dato2 e dato come nel
char dato;                              //trasmettitore

configura();

UART1_init(1000);                       //inizializzazione modulo UART

while(1){
    centro();                           // si posiziona al centro

       while( UART1_Data_Ready()==1){   //controlla se ci sono dati

            dato= UART1_Read();         // se ci sono vengono caricati nella variabile

            if (dato == dato1) {        //confronto tra dato e dato1
               destra();                //chiama la funzione destra()
               delay_ms(30);
               }

            if (dato == dato2) {       //confronto tra dato e dato2
               sinistra();             //chiama la funzione sinistra()
               delay_ms(30);
               }
         }
```

Foto



il montaggio su breadboard



servomecomando

Conclusioni

Ho personalmente provato i circuiti montati entrambi su breadboard agli antipodi di casa mia (una casa di circa 100m²)tenendo conto dei vari divisori e che per antenna c'era un pezzo di cavo, il tutto ha funzionato egregiamente.
Se ci si fa un circuito stampato rispettando le specifiche del costruttore Aurel le cose andranno anche meglio.

Vorrei concludere ringraziando tutta la comunità che partecipa attivamente al forum (solo leggendo i vari post ho imparato tanto).

Un particolare ringraziamento va a **TardoFreak** per la pazienza avuta con me tempo fa per l'acquisto del programmatore e a **Paolino** per i suoi consigli diretti (forum) e indiretti attraverso il suo libro **Pillole di microcontrollori**

Download

Ecco i link per scaricare i file di

- **ricevitore**
- **trasmettitore**

Estratto da "http://www.electroyou.it/mediawiki/index.php?title=UsersPages:Mpeino:controllare-un-servocomando-a-distanza"