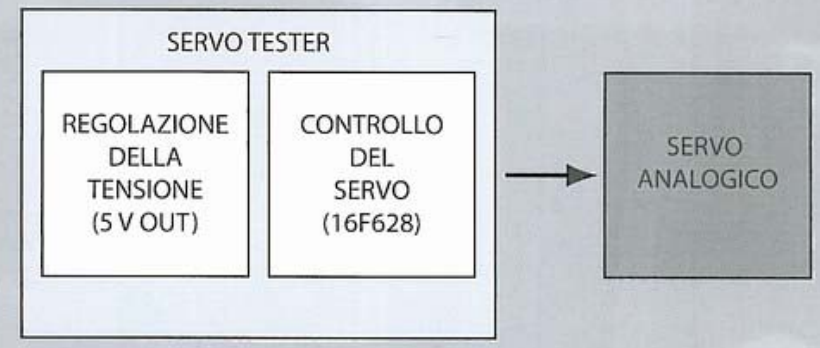
REALIZZIAMO IL SERVO TESTER

In questo Workshop completiamo il sistema di test dei servocomandi visto nel precedente fascicolo, realizzandone la struttura elettronica.

Dopo aver analizzato il firmware di controllo dei servocomandi, passiamo ora ad affrontare le questioni legate alla **struttura elettronica del circuito di test.** Come già avvenuto in passato, esso sarà suddiviso in **due macroblocchi,** il primo dedicato alla **regolazione della tensione** per mezzo del circuito integrato **2940** e il secondo composto dal **PIC 16F628** e dai componenti necessari al controllo del servocomando. La struttura del regolatore di tensione sarà praticamente identica a quella impiegata nei circuiti dei fascicoli precedenti e, in maniera per certi versi analoga, ciò varrà anche per il blocco di controllo costituito dal PIC 16F628 e da tre circuiti di input digitale (sempre composti da un micropulsante e da un resistore, come visto nel Workshop 62).

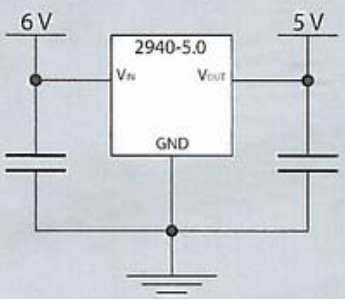


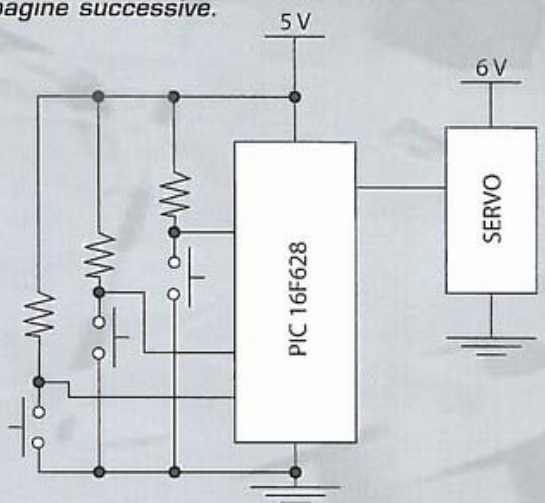
Sopra, lo schema a blocchi del nostro circuito di test, composto dal circuito di regolazione della tensione e dal microcontrollore.

IL CIRCUITO FINALE ›››

Vediamo a destra uno **schema di massima del circuito** che realizzeremo. Sebbene tale struttura sia praticamente identica a quella degli esempi precedenti, come già anticipato nel fascicolo 64, puoi notare come in questo esperimento vengano utilizzate **due distinte tensioni di alimentazione:** la prima di circa **6 V** viene fornita direttamente dal **pacco batterie** e **alimenta il servocomando analogico** e il **regolatore di tensione 2940.** La seconda, invece, è data dai **5 V** ottenuti utilizzando il **regolatore di tensione** e viene impiegata per **alimentare il microcontrollore** e per ottenere i **segnali di input digitale.**

Nello schema è mostrata la struttura elettronica dei due macroblocchi del circuito di test, che potrai realizzare nello StepbyStep delle pagine successive.





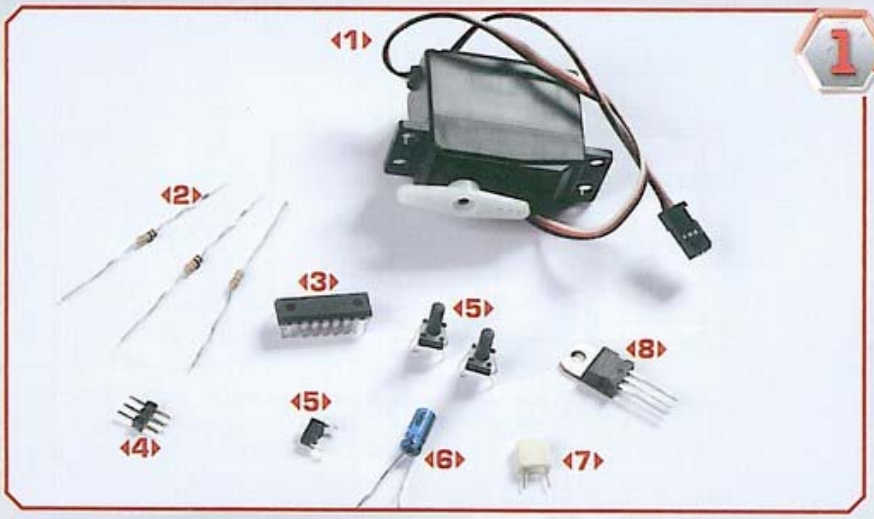
STEPbySTEP

IL CIRCUITO DI TEST

Vediamo ora come realizzare il circuito di test descritto nella prima pagina di questo Workshop. Per poter proseguire dovrai, ovviamente, compilare il firmware descritto nel fascicolo 64 all'interno dell'ambiente mikroC e caricare il file binario ottenuto in un PIC 16F628.

I componenti richiesti per realizzare il circuito di test sono i seguenti:

1

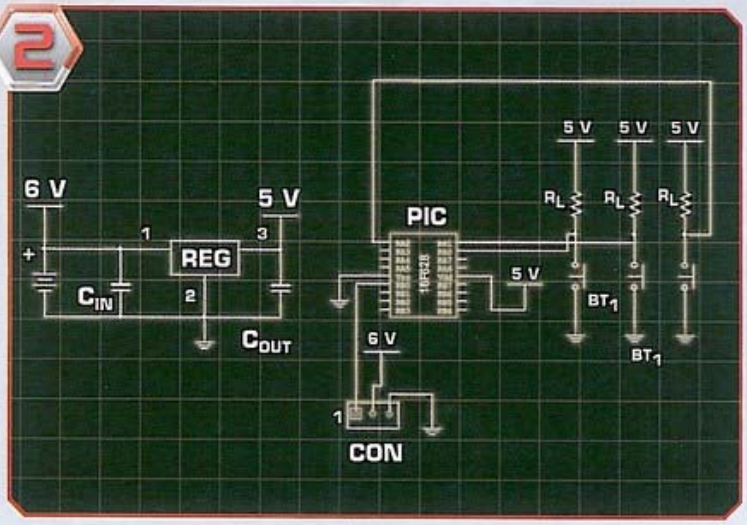


‹1› un servocomando analogico
‹2› 3 resistori da 10 kohm (R_L)
‹3› un microcontrollore 16F628 (PIC)
‹4› una porzione di strip line maschio da 3 unità (puoi ricavarla tagliando una normale strip line maschio) (CON)
‹5› 3 micropulsanti (puoi usarne anche di due tipi differenti, per differenziare visivamente i movimenti) (BT_1, BT_2, BT_3)
‹6› un condensatore da 22 µF (C_{OUT})
‹7› un condensatore da 0,47 µF (C_{IN})
‹8› un regolatore 2940-5.0 (REG)

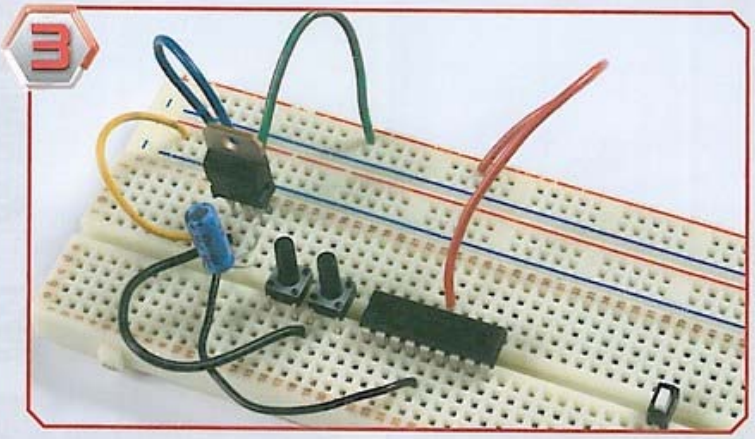
Oltre a questi componenti, sarà come sempre necessaria una breadboard con i relativi fili e un pacco batterie da 4 stilo.

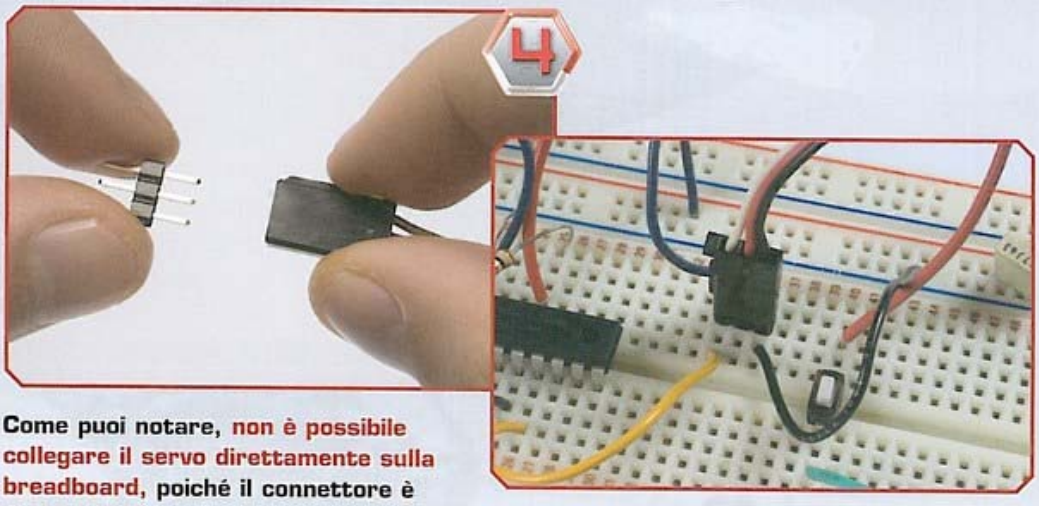
Nello schema a lato puoi osservare il circuito che dovrai realizzare. Come detto in precedenza, l'alimentazione del servocomando dovrà essere fornita direttamente dal pacco batterie. Per la piedinatura del connettore del servocomando puoi far riferimento alle indicazioni mostrate a pag. 5 del fascicolo 11.

2

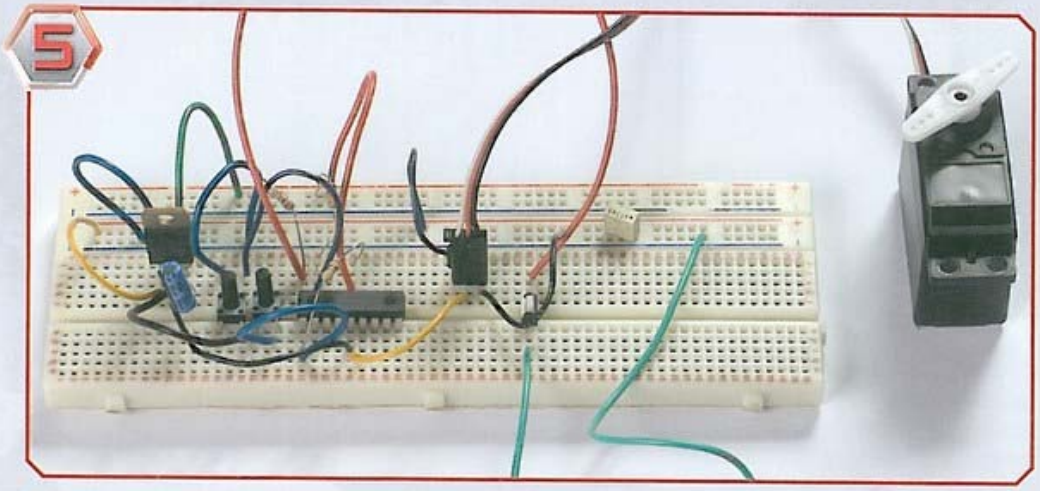


Inizia ad assemblare il circuito a partire, come sempre, dal posizionamento del PIC e del blocco di regolazione della tensione. Aggiungi in seguito i pulsanti e crea i collegamenti utilizzando i fili per breadboard.





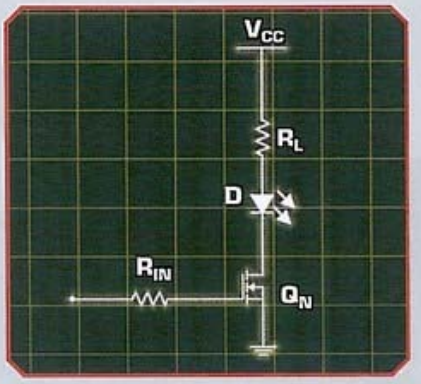
Come puoi notare, non è possibile collegare il servo direttamente sulla breadboard, poiché il connettore è dotato di un attacco di tipo femmina che non può essere inserito nei fori della basetta. Per ovviare al problema inserisci una porzione di strip line maschio da tre unità nel connettore del servocomando, in modo da poterlo collegare alla breadboard. Innesta il connettore e crea i collegamenti con le linee di alimentazione e il PIC.



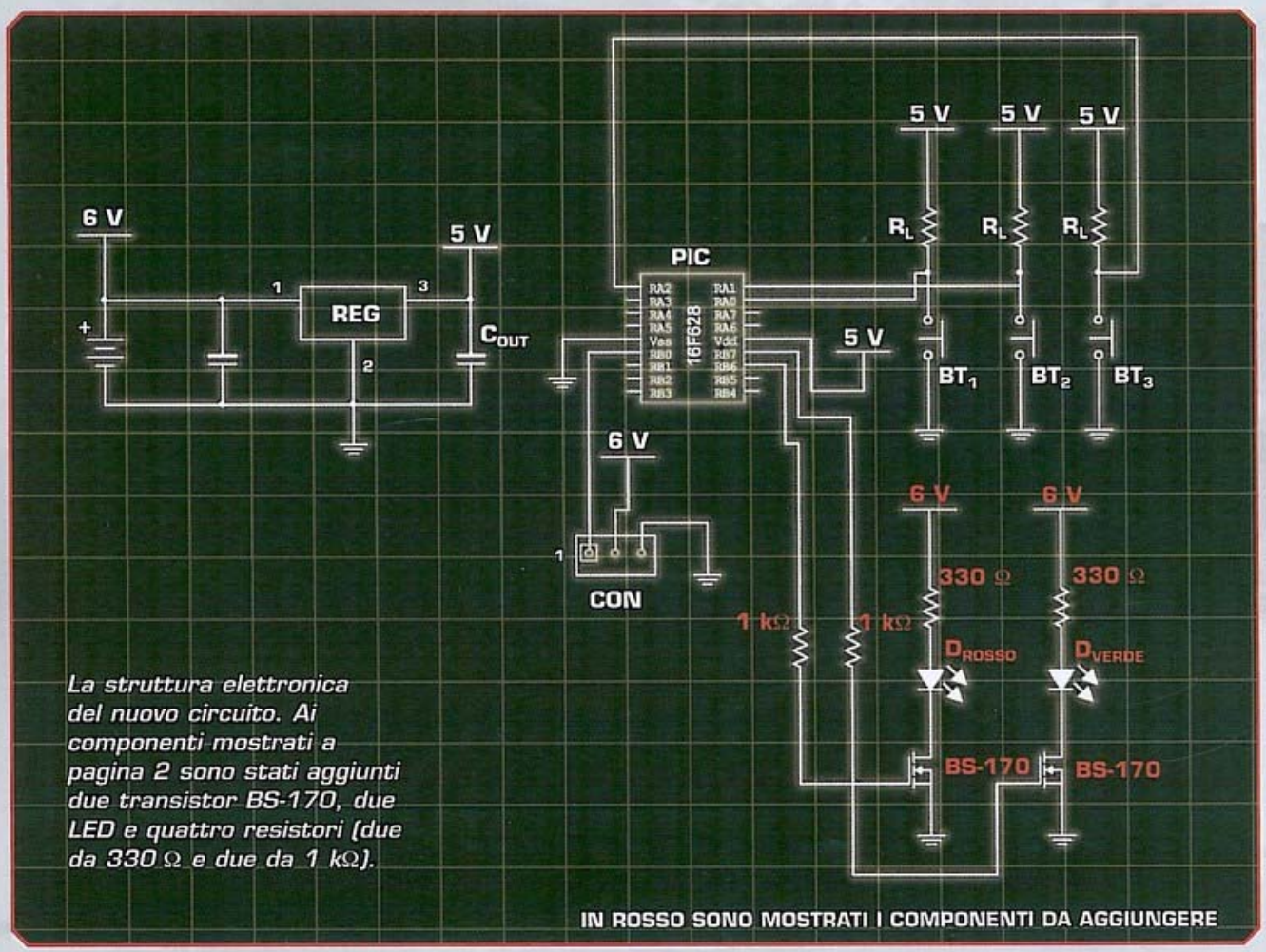
Il circuito è ultimato. Non ti resta che collegare il pacco batterie per accenderlo. Se tutto è stato montato correttamente, non appena alimenterai il circuito vedrai il servocomando irrigidirsi e muoversi per raggiungere una posizione di centratura. Cliccando i tre pulsanti di interazione potrai, poi, variare la durata dell'impulso, cambiando con essa l'orientamento dell'albero del servocomando.

INSERIAMO DUE LED DI 'FINE CORSA' ›››

In questa seconda parte del Workshop ti illustreremo un **'upgrade'** del sistema appena mostrato, attraverso cui ti sarà possibile ricevere una **segnalazione luminosa del raggiungimento degli estremi di durata dell'impulso.** La visualizzazione luminosa sarà possibile aggiungendo **una coppia di LED** (uno rosso e uno verde, in modo da differenziare i due estremi) comandati da **due transistor BS-170** attraverso i pin della porta B (vedi schema sotto). L'utilizzo dei transistor di attivazione dei LED è, per certi versi, una novità rispetto a quanto fatto nei fascicoli precedenti, nei quali abbiamo sfruttato direttamente le correnti di source del microcontrollore. Questa scelta, tuttavia, ci consente di **limitare la corrente assorbita dal PIC,** rendendo possibile pilotare una maggior quantità di dispositivi elettronici. Lo schema elettrico qui sotto ti mostra come collegare le due nuove porzioni di circuito al PIC, sfruttando **i pin 6 e 7 della porta B, già configurati come uscite nel corso del precedente esperimento.** Nelle due pagine che seguono puoi vedere il sorgente del nuovo firmware. Per rendere più semplice e immediata la comprensione delle modifiche apportate, esso viene mostrato interamente in colore nero (rimuovendo così la colorazione di parole chiave e direttive), evidenziando in rosso tutti i blocchi di codice che sono stati aggiunti e modificati.



Il circuito di attivazione del LED che aggiungeremo al circuito originale.



La struttura elettronica del nuovo circuito. Ai componenti mostrati a pagina 2 sono stati aggiunti due transistor BS-170, due LED e quattro resistori (due da 330 Ω e due da 1 kΩ).

IL CODICE ›››

Come puoi notare, la principale variazione rispetto alla prima versione del firmware è stata l'aggiunta di un **nuovo blocco di codice** in chiusura al ciclo, nel quale viene verificata la durata dell'impulso. Nel caso in cui si verifichi in una condizione in cui l'impulso venga generato con una delle due durate limite, il firmware attiverà il LED corrispondente, portando alto il livello logico degli appositi pin.

Ovviamente, **per poter sperimentare questo nuovo sorgente dovrai ricompilarlo in mikroC e caricare il binario così ottenuto nella memoria del PIC**, aggiungendo in ultimo i nuovi componenti.

Il firmware aggiornato

```
/* definizione dei parametri del programma */
#define CENTRATURA 15
#define T_MIN 11
#define T_MAX 19
#define PREMUTO 0
#define RILASCIATO 1

/* funzione principale */
void main()
{

  /*indice per la gestione dei cicli */
  int i;

  /* durata dell'impulso */
  int durata_us = CENTRATURA;


  /* stato dei tre pulsanti di ingresso */
  int stato_RA0 =  RILASCIATO;
  int stato_RA1 =  RILASCIATO;
  int stato_RA2 =  RILASCIATO;


  /*configurazione delle porte del PIC. Porta A con in tre pin meno significativi
    in input e porta B configurata con tutti i pin in uscita. */
  TRISA = 0b00000111;
  TRISB = 0b00000000;

  /* disattivazione dei comparatori del 16F628 */
  CMCON = 0b00000111;


  while(1)
  {

    /*inizio dell'impulso con l'innalzamento dell'output RB0 */
    PORTB.F0 = 1;


    /*generazione del ritardo di impulso */
    for(i=0; i<durata_us; i++) Delay_us(100);


    /*termine dell'impulso con l'abbassamento dell'output RB0 */
    PORTB.F0 = 0;
```

```
    /*inizio gestione dei pulsanti */

    /*rilevamento dei rilasci se la variabile di stato dei pulsanti ha registrato la
      pressione, ma il bit corrispondente segnala il pulsante rilasciato, E' stato
      ultimato il click */


    /* pulsante di decremento */
    if(stato_RA0 == PREMUTO && PORTA.F0==RILASCIATO)
    {
       durata_us--; /* decremento la durata */
       /* se la durata dell'impulso ha superato il limite minimo, la correggo */
       if (durata_us<T_MIN) durata_us = T_MIN;
    }


    /* pulsante di centratura */
    if(stato_RA1 == PREMUTO && PORTA.F1==RILASCIATO)
    {
       durata_us = CENTRATURA; /* decremento la durata */
    }


    /* pulsante di incremento */
    if(stato_RA2 == PREMUTO && PORTA.F2==RILASCIATO)
    {
       durata_us++; /* decremento la durata */
    }

    /* se la durata dell'impulso ha superato il limite massimo, la correggo */
    if (durata_us>T_MAX) durata_us = T_MAX;
    }


    /*rilevamento della pressione dei pulsanti e memorizzazione degli stati */
    stato_RA0 = PORTA.F0;
    stato_RA1 = PORTA.F1;
    stato_RA2 = PORTA.F2;

    /* controllo degli estremi di impulso */
    if(durata_us == T_MAX)
       PORTB.F7 = 1;
    else
       PORTB.F7 = 0;

    /* controllo degli estremi di impulso */
    if(durata_us == T_MIN)
       PORTB.F6 = 1;
    else
       PORTB.F6 = 0;

     /* generazione del ritardo di chiusura */
     for(i=0; i<RITARDO; i++) Delay_us(100);

 } /* si chiude il ciclo di funzionamento del firmware */

}
```