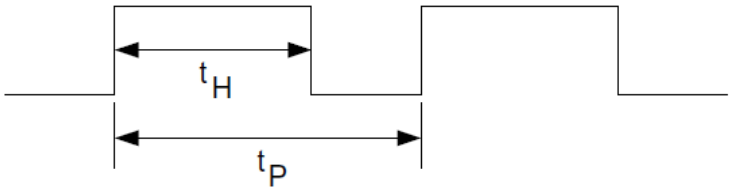
PWM

Il Pulse-Width Modulation (PWM) è un metodo di codifica dei dati che varia la durata della semionda positiva (denominata tH in figura 1) e negativa (desumibile in figura come tP-tH). Il rapporto fra il tempo in cui la forma d'onda è a livello alto (tH) e il tempo necessario per ottenere una forma d'onda completa (tP) si definisce duty cycle. Maggiore risulta il duty cycle è maggiore sarà la corrente che scorre nel carico.
Una importante considerazione deriva dal fatto che la maggior parte dei microcontrollori attuali gestisce le informazioni (ovvero le variabili) con un parallelismo di 8 bit (1 byte) o con i suoi multipli. Si ricorda che, nella scheda Arduino, la variabile di tipo "byte" oppure "char" utilizza 1 byte di memoria mentre la variabile di tipo "int" ne utilizza 2 byte. Considerando la variabile "byte" si hanno a disposizione 2^8=256 differenti valori memorizzabili. Risulta evidente che per semplificare i calcoli si utilizzi il byte o i suoi multipli per suddividere il tempo (tP).



$$dutycycle = \frac{t_H}{t_P}$$

Figura 1. Duty cycle della forma d'onda con il PWM

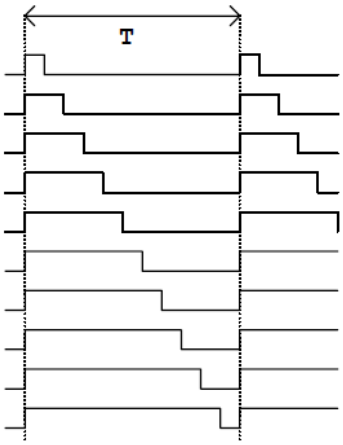


Figura 2. Esempio tipico delle forme d'onda ottenute con il PWM

Ad esempio si consideri la necessità di utilizzare un PWM con un periodo di 2,56 ms (0,00256 s che coincide con una frequenza di 390,625 Hz) e un duty cycle del 10%, 50% e 90%.
Conoscendo il periodo totale (2,56 ms) che verrà suddiviso in 256 parti, in modo da poter memorizzare il valore in un singolo byte (variabile di tipo "byte" da 0 a 255) si ottiene:
0,00256 / 256 = 0,00001 sec. = 10 μs
Con un duty cycle del 20% volendo conoscere per quanto tempo si deve rimanere a livello High si avrà: Ton = (tP * 20)/100 = 0,000512 sec. = 512 μs mentre per
Quindi deve rimanere costante sia la tensione che può assumere nel caso di uscita digitale solo valore alto (High = VCC = +5V) oppure basso (Low = GND = 0V) e la frequenza dell'oscillazione ovvero il tempo tP.
In seguito viene riportato un software per gestire la regolazione di luminosità di un led (Progetto 0).

Progetto 0

```
/*-----------------------------------------------
nome progetto:        PWM-LED
Specifiche del progetto:
Variatore di luminosità di un diodo led collegato al PIN 13 alla cui uscita è collegato l'anodo del led,
mentre il catodo risulta collegato a GND.
Al PIN 12 è collegato un pulsante di tipo n.a. denominato "UP" che serve per incrementare la luminosità.
Al PIN 11 è collegato un pulsante di tipo n.a. denominato "DOWN" che serve per decrementare la luminosità.
Entrambi i pulsanti possiedono una R da 10Kohm di pullup verso il +5V.
Creato da:     G. Carpignano          data:   20/10/2011
Compilatore:   per scheda Arduino 2009       Vers.   022
  ----------------------------------------------- */
unsigned int i, counter, ton, toff;
byte pulsanti;
byte puls_up = 12, puls_down = 11, led = 13;

/* Si consideri un'onda quadra formata da:
                    un SEMIPERIODO POSITIVO denominato TON
                    un SEMIPERIODO NEGATIVO denominato TOFF
Il tempo totale T sarà dato da: T = TON + TOFF, mentre la frequenza sarà l'inverso del tempo totale.
Nel caso in esame si è suddiviso il tempo totale in 256 porzioni. La scelta del valore 256 è riconducibile alla
possibilità di memorizzare i valori con una costante di tipo a 8 bit come l'unsigned byte.
Ovviamente il rapporto tra la semionda positiva e negativa viene definito DUTY CYCLE.
Si prenda in considerazione un Duty Cycle del 20% per una frequenza di 1024 HZ che coincide con un T = 0,976562
msec e si desideri calcolare quanto vale il TON e il TOFF, cioè il semiperiodo positivo e negativo, si dovrà
effettuare il seguente calcolo:
il 20% di 256 sarà uguale a (256 * 20) / 100 = 51.2, che viene arrotondato al numero intero 51, quindi per la
semionda Positiva il PIN 13 deve essere forzato a livello ALTO per un tempo TON = (0,976562 / 256) * 51 =
0,194549 msec.
Per calcolare la Semionda Negativa occorre effettuare il seguente calcolo: TOFF = T - TON.
N.B. -  Ai due estremi occorre prestare attenzione perché se consideriamo un Duty cycle del 100% NON SI OTTIENE
una frequenza ma una tensione SEMPRE a livello ALTO, mentre se il Duty Cycle scende al 0% si ottiene SEMPRE una
tensione a livello logico BASSO sull'output PIN 13. Questo significa che non viene più generata una frequenza;
per tale motivo se il range massimo è stato suddiviso in 256 parti, si otterrà che:
              MASSIMA LUMINOSITA' DEL LED          -->    TON = 255 e TOFF = 1
              MINIMA LUMINOSITA' DEL LED           -->    TON = 1 e TOFF = 255
              CIRCA META' LUMINOSITA' DEL LED              -->    TON = 128 e TOFF = 128
Nei precedenti esempi si evidenzia come la frequenza sia sempre uguale perché f = 1 / (TON + TOFF) dove TON +
TOFF vale sempre 256. */

#define valmax 255    // valore massimo del numero di cicli da effettuare per ottenere una forma d'onda completa
#define valmin 1      // valore minimo del numero di cicli da effettuare per ottenere una forma d'onda completa
#define valoremaxcounter 100  /* contatore del numero di volte che vengono letti i pulsanti prima di effettuare
una modifica alle variabili TON o TOFF. */

void setup()
```

```
{
  pinMode(led, OUTPUT); // inizializza il pin 13 come OUTPUT collegato al LED
  pinMode(puls_down, INPUT); // inizializza il pin 11 come INPUT collegato al pulsante n.a. DOWN
  pinMode(puls_up, INPUT); // inizializza il pin 12 come INPUT collegato al pulsante n.a. UP
  digitalWrite(puls_down, HIGH);  // attiva Rpullup da 10Kohm sul pin11
  digitalWrite(puls_up, HIGH);  // attiva Rpullup da 10Kohm sul pin12
}
/*    funzione per controllare quali pulsanti vengono premuti    */
void controllapulsanti(void)
{
  // memorizza il valore dei due pulsanti UP e DOWN in una unica variabile
  pulsanti = digitalRead(puls_up) * 2 + digitalRead(puls_down) * 1;
  // Le possibili combinazioni della variabile denominata "pulsanti" sono:
  // 1) 00000000B → coincide con entrambi i pulsanti premuti
  // 2) 00000001B → coincide con il pulsante UP premuto e pulsante DOWN non premuto
  // 3) 00000010B → coincide con il pulsante DOWN premuto e pulsante UP non premuto
  // 4) 00000011B → coincide con entrambi i pulsanti non premuti
  switch (pulsanti)
  {
   case 0:            // entrambi i pulsanti sono premuti
        {
          counter = 0;
          break;
        }
   case 2:            // solo il pulsante DOWN collegato al PIN 11 è premuto
        {
          counter++;
          if (counter == valoremaxcounter)
            {
              if (toff != valmax)
                  {
                    toff++; // incrementa TIME OFF (semiperiodo negativo dell'onda quadra)
                    ton--;  // decrementa TIME ON (semiperiodo positivo dell'onda quadra)
                    counter = 0;
                  }
            }
           break;
        }
   case 1:            // solo il pulsante UP collegato al PIN 12 è premuto
        {
          counter++;
          if (counter == valoremaxcounter)
            {
              if (ton != valmax)
                  {
                    ton++;  // incrementa TIME ON (semiperiodo positivo dell'onda quadra)
                    toff--; // decrementa TIME OFF (semiperiodo negativo dell'onda quadra)
                    counter = 0;
                  }
            }
          break;
        }
   case 3:            // nessun pulsante è premuto
        {
          counter = 0;
          break;
        }
  }
}
/*    Programma principale   */
void loop ()
{
  digitalWrite(led, LOW);  // forza basso l'output del PIN 13 (spegni il led)
  counter = 0; // azzera contatore numero letture delle pressioni del tasto
  ton = valmin;       // valore iniziale del semiperiodo positivo
  toff = valmax;      // valore iniziale del semiperiodo negativo

  while (1)           // ripeti il ciclo infinito
   {
     // genera la semionda positiva
          digitalWrite(led, HIGH);  // forza alto l'output del PIN 13 (accendi il led)
          //aspetta per un periodo di tempo prefissato che è dipendente dalla variabile ton
   for (i = 0; i < ton; i++);
          controllapulsanti();              // controlla lo stato dei pulsanti
     // genera la semionda negativa
          digitalWrite(led, LOW);  // forza basso l'output del PIN 13 (spegni il led)
          //aspetta per un periodo di tempo prefissato che è dipendente dalla variabile toff
   for (i = 0; i < toff; i++);
          controllapulsanti();              // controlla lo stato dei pulsanti
   }
}
```

Progetto 1

Controllo della velocità di una ventola con il Computer

Si sviluppi un software in linguaggio C per la scheda Arduino in grado di soddisfare le seguenti richieste:

1) Si deve poter regolare la velocità di rotazione di una piccola ventola collegata sul collettore di un transistor NPN del tipo BC107 mentre la base è collegato al PIN 7 e l'emettitore a GND, tra un valore minimo (ventola ferma) ed un valore massimo (ventola funzionante alla massima velocità) tramite l'utilizzo di un valore numerico compreso tra 0 e 9 inserito dalla tastiera del Personal Computer a cui è collegata la scheda Arduino tramite l'interfaccia USB.
2) Il numero di step necessari per aumentare la velocità di rotazione della ventola è uguale a 256, ovvero tra 0 (velocità nulla → valore inserito da tastiera → 0) e 255 (velocità massima → valore 9 inserito da tastiera).
3) Per tutto il tempo che NON viene inserito un nuovo valore la velocità di rotazione della ventola deve rimanere COSTANTE.

La piccola ventola è possibile recuperarla da un vecchio PC (Figura 1). Abbiamo intenzione di utilizzare una di queste ventole per tenerci al fresco in estate.
Ovviamente, un semplice interruttore on / off non sarebbe in linea con il nostro metodo di risoluzione del problema, per cui la velocità della ventola sarà controllabile con la scheda Arduino tramite un valore numerico digitato sul Personal Computer.

COMPONENTI E STRUMENTAZIONE

Scheda Arduino Uno / Duemilanove
R1 = 270 ohm 1/2W metal film resistor
T1 = BD139 power transistor
M1 = ventola a +12V recuperata da un computer
Alimentatore da 12V / 1A

Hardware

Siamo in grado di controllare la velocità della ventola utilizzando l'uscita analogica (PWM) che controlla un transistor di potenza collegato al motore della ventola. Dal momento che queste ventole del computer utilizzano una tensione di alimentazione solitamente di 12V, useremo un alimentatore esterno per fornire la potenza di azionamento per il ventilatore.
La Figura 2 mostra lo schema del progetto mentre la Figura 3 mostra il layout della basetta.

Software

Si tratta di un software molto semplice (Listato Progetto 1).
In sostanza, occorre solo leggere un valore compreso tra 0 a 9 dalla porta USB e utilizzare un analogWrite al Pin su cui è collegato il motore dopo averlo moltiplicato per 28 in modo da scalare fino a un numero compreso tra 0 e 252. Si consideri che il valore minimo è 0 mentre il massimo è uguale a 255 per la funzione "analogWrite()".

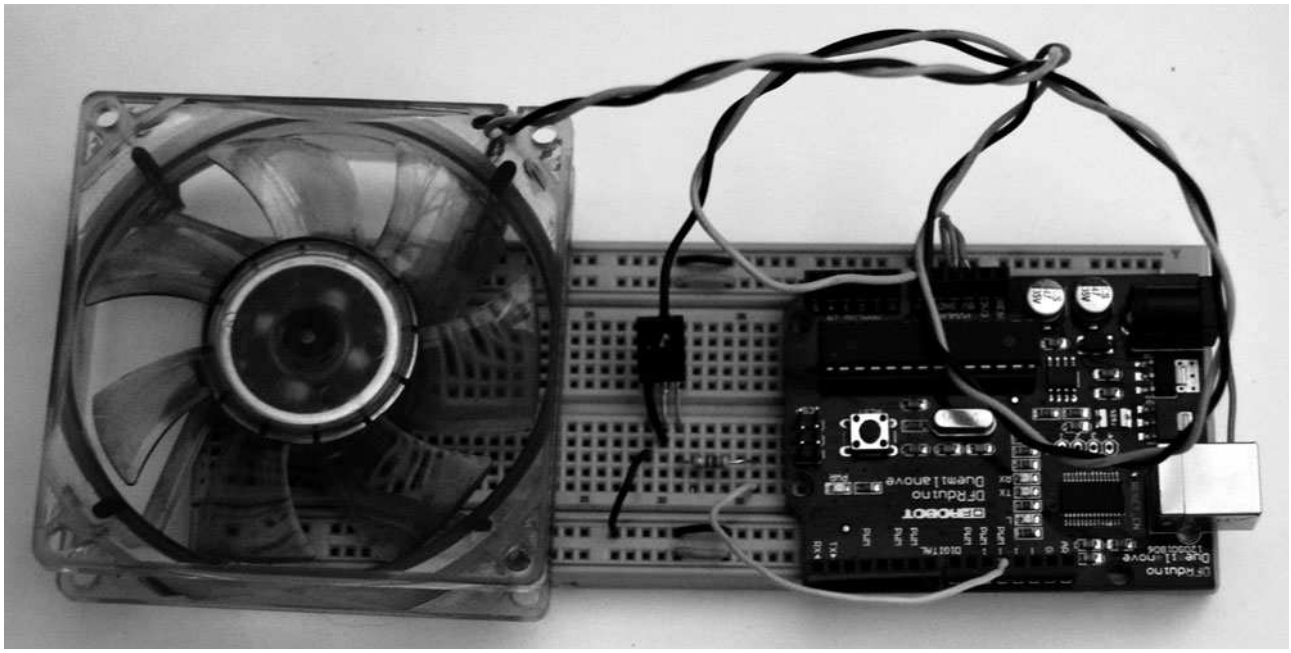


Fig. 1 Progetto 1. Velocità di una ventola controllata dal Computer.

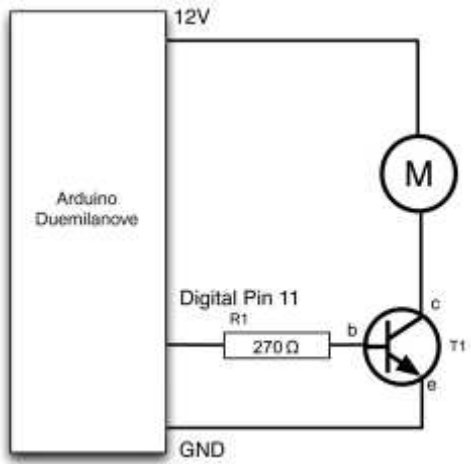


Fig. 2 Schema dei collegamenti per il Progetto 1.

LISTATO PROGETTO 1

```
int motore = 11;  // definisce il PIN 11 collegato al motore della ventola
void setup()
{
  pinMode(motore, OUTPUT); // definisce come OUTPUT il PIN 11 collegato alla ventola
  analogWrite(motore, 0);  // ferma il motore della ventola
  Serial.begin(9600);    // inizializza la seriale a 9600 baud
}
void loop()
{
  if (Serial.available())  // controlla se esiste un carattere sulla seriale in RX
    {
     char ch = Serial.read(); // leggi il carattere dalla seriale
     if (ch >= '0' && ch <= '9') // se il carattere è un numero <> da 0 a 9 non considerarlo
       {
         int speed = ch - '0'; // elimina offset
         analogWrite(motore, speed * 28);
       }
    }
}
```

Occorre caricare il software per il Progetto 1 sul Sketchbook Arduino compilandolo e scaricarlo sulla scheda Arduino. Considerato che ci sono così pochi componenti in questo progetto si può pensare di eliminare il montaggio su una basetta sperimentale ausiliaria.

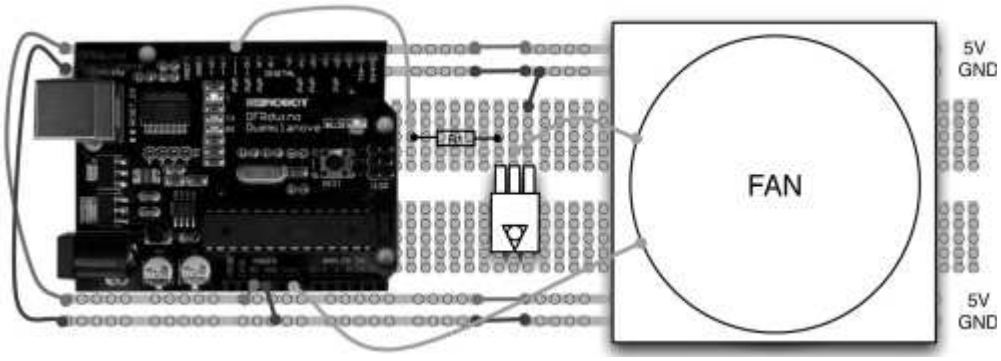


Fig. 3 Breadboard layout for Progetto 1.

Circuito con ponte ad "H" per controllare il senso di rotazione di un motore in corrente continua utilizzando una singola alimentazione

Per cambiare la direzione in un motorino alimentato con una sola tensione di alimentazione, si deve invertire la direzione in cui scorre la corrente. Per fare questo si richiedono quattro interruttori o transistor. La Figura 4 mostra lo schema di principio implementato con dei semplici interruttori.

Nella Figura 4, S1 e S4 sono gli interruttori chiusi mentre S2 e S3 sono gli interruttori aperti. Ciò permette alla corrente di fluire attraverso il motore con il terminale A positivo e il terminale B negativo. Se dovessimo invertire gli interruttori in modo che S2 e S3 sono chiusi e S1 e S4 sono aperti, allora B sarà positivo e A sarà negativo e il motore girerà in direzione opposta.

Tuttavia, potrebbe essere individuato un pericolo con questo circuito. Cioè, se per caso S1 e S2 sono entrambi chiusi, il positivo sarà direttamente collegato all'alimentazione negativa ottenendo il cortocircuito dell'alimentatore. Lo stesso vale se S3 e S4 sono entrambi chiusi al tempo stesso. Nel progetto successivo, useremo dei transistor / mosfet di potenza al posto degli interruttori per controllare il senso di rotazione di un motore elettrico.

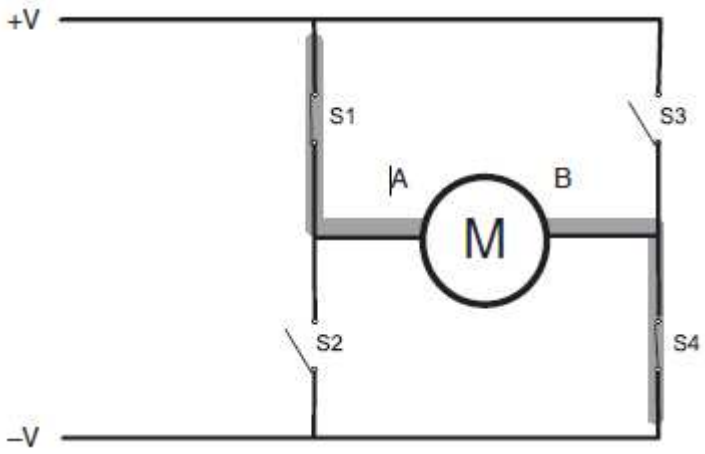


Fig. 4 Schema elettrico di un tpico circuito a ponte di tipo "H" per ottenere l'inversione del senso di rotazione del motorino in continua.

Progetto 2

Ipnotizzatore

Per ipnotizzare non occorre alcun potere particolare come non occorre una formula segreta. L'elemento fondamentale è il rapporto che si viene a stabilire tra due persone. Chi infatti si lascerebbe accompagnare in un territorio sconosciuto da qualcuno rispetto al quale non nutre alcuna fiducia? Come è importante suscitare fiducia è anche necessario dimostrare sicurezza. Il comando della vostra auto lo lascereste ad un pilota insicuro? Considerate poi che lo stato di trance non è una qualche forma di sonno. Non c'è un interruttore da premere imponendo un brusco passaggio da uno (acceso) a zero (spento). Esso è invece un progressivo stato di depotenziamento dell'attività coscio-razionale del cervello. Ciò non è così difficile da ottenere. Sembra difficile solo perchè crediamo di essere molto più capaci a controllare il mondo esterno ed interno di ciò che in realtà accade.
Vediamo come progettare e costruire un semplice circuito per il controllo mentale. Questo progetto (vedi Figura 5) prevede il controllo completo di un motore non solo è in grado di controllare la sua velocità, ma anche di farla girare in senso orario e antiorario. Collegato al motore ci sarà un disco a spirale vorticosa con lo scopo di ipnotizzare le sfortunate e inconsapevoli vittime.

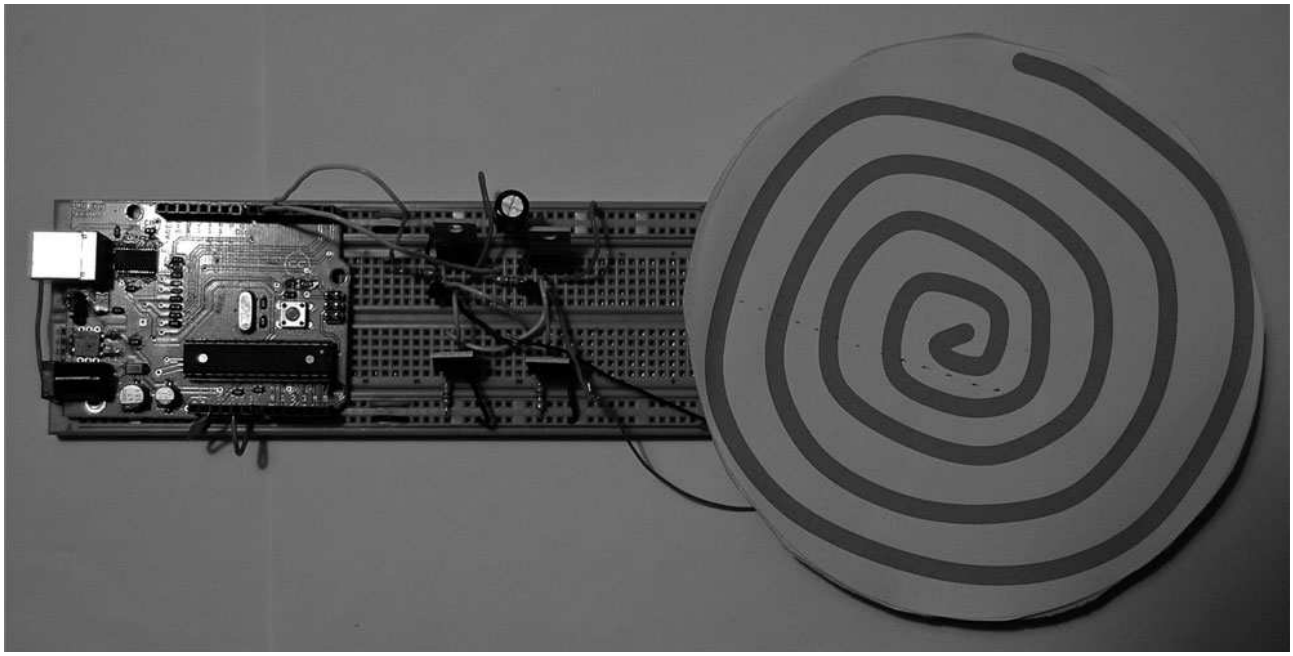


Fig. 5 Progetto 2. L'ipnotizzatore.

COMPONENTI E STRUMENTAZIONE

Scheda Arduino Uno / Duemilanove
T2 = T4 = FQP33N10 N-channel power MOSFET.
T1 = T3 = FQP27P06 P-channel power MOSFET.
T4 = T5 = Transistor tipo BC548
R1 = R2 = R3 = R4 =R5 = R6 = 10 Kohm – 1/2W metal film resistor
M1 = 12V motorino di un lettore CD o ventola recuperata da un computer
12V / 1A Alimentatore

Il motore che abbiamo usato in questo progetto è stato recuperato da un lettore CD di un Personal Computer guasto. Una fonte alternativa a buon mercato sarebbe un motore di un vecchio giocattolo motorizzato con gli ingranaggi della guida di una ruota che si adattano particolarmente bene per essere utilizzati per il disco ipnotico.

Hardware

Lo schema dei collegamenti per l'ipnotizzatore è mostrato in Figura 6. Si utilizza un ponte ad H per ottenere l'inversione della polarità. Si noti che stiamo usando i MOSFET piuttosto che dei tradizioni transistor bipolari per il controllo dell'alimentazione principale. In teoria, questo ci permetterà di controllare anche dei motori di elevata potenza con il vantaggio che i MOSFET a malapena si riscaldano con il piccolo motore utilizzato e, di conseguenza, non avremo bisogno dei dissipatori di calore.

I collegamenti dei MOSFET inferiori sono astutamente collegati alle uscite dei loro transistor diagonalmente opposti, in modo che quando si disattivano T1 e T4 si accendono automaticamente sia T3 che T2 e viceversa.

Le resistenze da R1 a R4 assicurarno che lo stato di default di T1 a T4 siano spenti (off) determinando un livello alto sui collettori di T5 e T6 degli stessi che essendo collegati al gate dei due P-MOSFET permette al motore di non essere percorso da nessuna corrente.

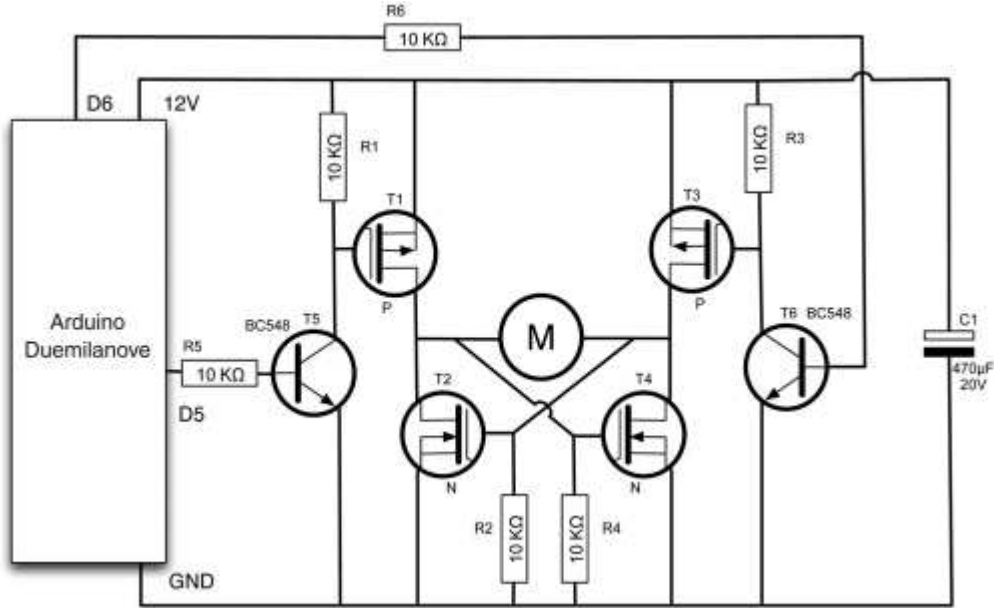


Fig. 6 Schema elettrico per il Progetto 2.

I due transistor T5 e T6 sono transistor bipolari a bassa corrente che sono usati per attivare i MOSFET denominati T1 e T3, rispettivamente. In questo caso, potremmo fare a meno di questi transistor e guidare i gate di T1 e T3 direttamente dalla scheda Arduino. Tuttavia, per fare questo, la logica sarebbe invertita (un livello alto al gate si trasformerebbe nel MOSFET spento). Potremmo far fronte a questo modificando in modo opportuno il software, ma l'altro più importante motivo per l'utilizzo di questi due componenti aggiuntivi è che potremmo utilizzare il circuito per il controllo di motori con tensioni di lavoro superiori ai +5V della scheda Arduino semplicemente utilizzando una tensione superiore alimentazione positiva.

Infine, C1 fornisce un serbatoio ausiliario per l'alimentazione durante i brevi momenti in cui si effettua l'inversione della rotazione del senso di marcia del motore.

La Figura 7 mostra il layout della spirale necessaria per il progetto.

L'ipnotizzatore ha bisogno di un modello di spirale per poter lavorare.

Fig. 7 Esempio di spirale per il circuito dell'ipnotizzatore.

Si può decidere di fotocopiare la Figura 7, ritagliala, e fissarla all'asse del motorino o della ventola. Nel prototipo utilizzato la spirale è stata ritagliata e incollata su un pezzo di cartone rigido che è stato poi incollato al piccolo ingranaggio sul lato del motore.

Software

La parte fondamentale di questo software (Listato Progetto 2) è quello di rendere impossibile che tutti i transistor sia contemporaneamente in conduzione nello stesso intervallo di tempo. Se questo accade, ci sarà un forte odore di bruciato e occorrerà sostituire qualche componente che si sarà bruciato.

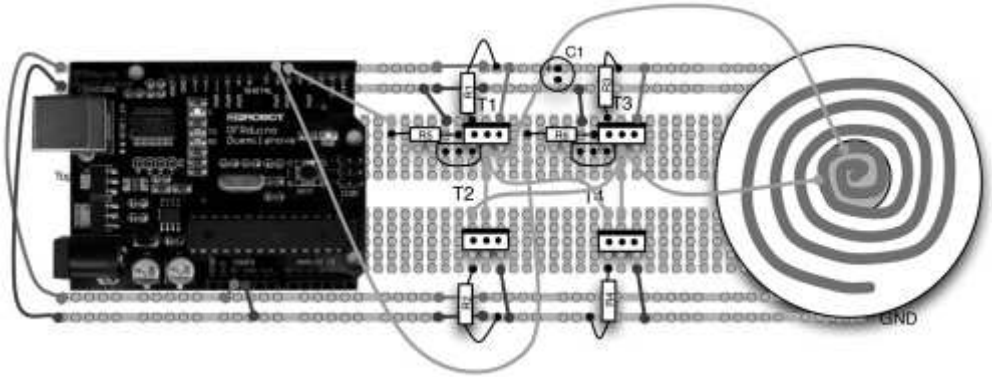


Fig. 8-12 Breadboard layout for Project 24.

LISTATO PROGETTO 2

```
int t1Pin = 5;
int t3Pin = 6;
int speeds[] = {20, 40, 80, 120, 160, 180, 160, 120, 80, 40, 20,
-20, -40, -80, -120, -160, -180, -160, -120, -80, -40, -20};
int i = 0;
void setup()
{
 pinMode(t1Pin, OUTPUT);
 digitalWrite(t1Pin, LOW);
 pinMode(t3Pin, OUTPUT);
 digitalWrite(t3Pin, LOW);
}

void allOff()  // funzione che disattiva tutti i transistor
{
 digitalWrite(t1Pin, LOW); //
 digitalWrite(t3Pin, LOW);
 delay(1);
}

void drive(int speed)  // funzione che imposta la velocità di rotazione del motore
{
 allOff();  // funzione per spegnere il motore
 if (speed > 0)
  {
   analogWrite(t1Pin, speed);
  }
 else if (speed < 0)
  {
   analogWrite(t3Pin, -speed);
  }
}

void loop()  // programma principale
{
 int speed = speeds[i];  // funzione per leggere la nuova velocità da impostare
 i++;   // incremento unitario del contatore
 if (i == 22)  // se e' stato raggiunto la fine dell'array riparte dall'inizio
  {
   i = 0;  // riparte dall'inizio dell'indice
  }
 drive(speed); // funzione per modificare la velocità del motore
 delay(150);   // ritardo per stabilizzare la velocità del motore
}
```

Prima di accendere qualsiasi transistor, tutti i transistor devono essere inizializzati in modalità spento (off) mediante la funzione ALLOFF.
Inoltre, la funzione ALLOFF include un leggero ritardo per assicurare che i transistor siano effettivamente spenti prima che vengano attivati.
Il software utilizza un array per controllare la progressione della velocità del disco. Questo permette al disco di ruotare sempre più velocemente in una direzione per poi rallentare fino a quando si inverte la direzione e poi inizia a ruotare sempre più velocemente in quella direzione e così via. Potrebbe essere necessario modificare questo array per il particolare motore che è stato utilizzato.

Dopo aver caricato il software completo per il Progetto 2 sul Sketchbook e averlo compilato e scaricato sulla scheda Arduino, occorre fare attenzione di controllare il cablaggio prima di applicare l'alimentazione per questo progetto. Ogni collegamento errato è facile che comporti la distruzione di uno o più componenti e in qualche caso anche la distruzione della scheda Arduino.

Servomotori
I servomotori sono piccoli componenti che vengono spesso usati in appararecchiature con radiocomando delle auto, in particolare per il controllo dello sterzo o per il controllo degli alettoni sugli aerei radiocomandati.
Sono disponibili in una varietà di formati e per i diversi tipi di applicazioni e il loro largo impiego nei modelli li rende relativamente poco costosi.
A differenza dei motori normali, non ruotano continuamente, anzi, è questo che si distinguono perché si ottiene un angolo particolare utilizzando un segnale PWM. I servomotori contengono sia la parte elettronica di controllo proprio per fare questo, quindi tutto quello che si deve fornire è una alimentazione a +5V e un segnale di controllo PWM che possiamo generare con la scheda Arduino.
Nel corso degli anni, l'interfaccia dei servi è diventato uno standard. Il servo deve ricevere un flusso continuo di impulsi almeno ogni 20 millisecondi. L'angolo che il servo mantiene è determinato dalla durata dell'impulso. Una larghezza di impulso di 1,5 millisecondi imposterà il servo nel suo punto medio, o 90 gradi. Un impulso di 1,75 millisecondi normalmente può farlo oscillare intorno a 180 gradi, ed un impulso più corto di 1,25 millisecondi, impostare l'angolo di 0 gradi.