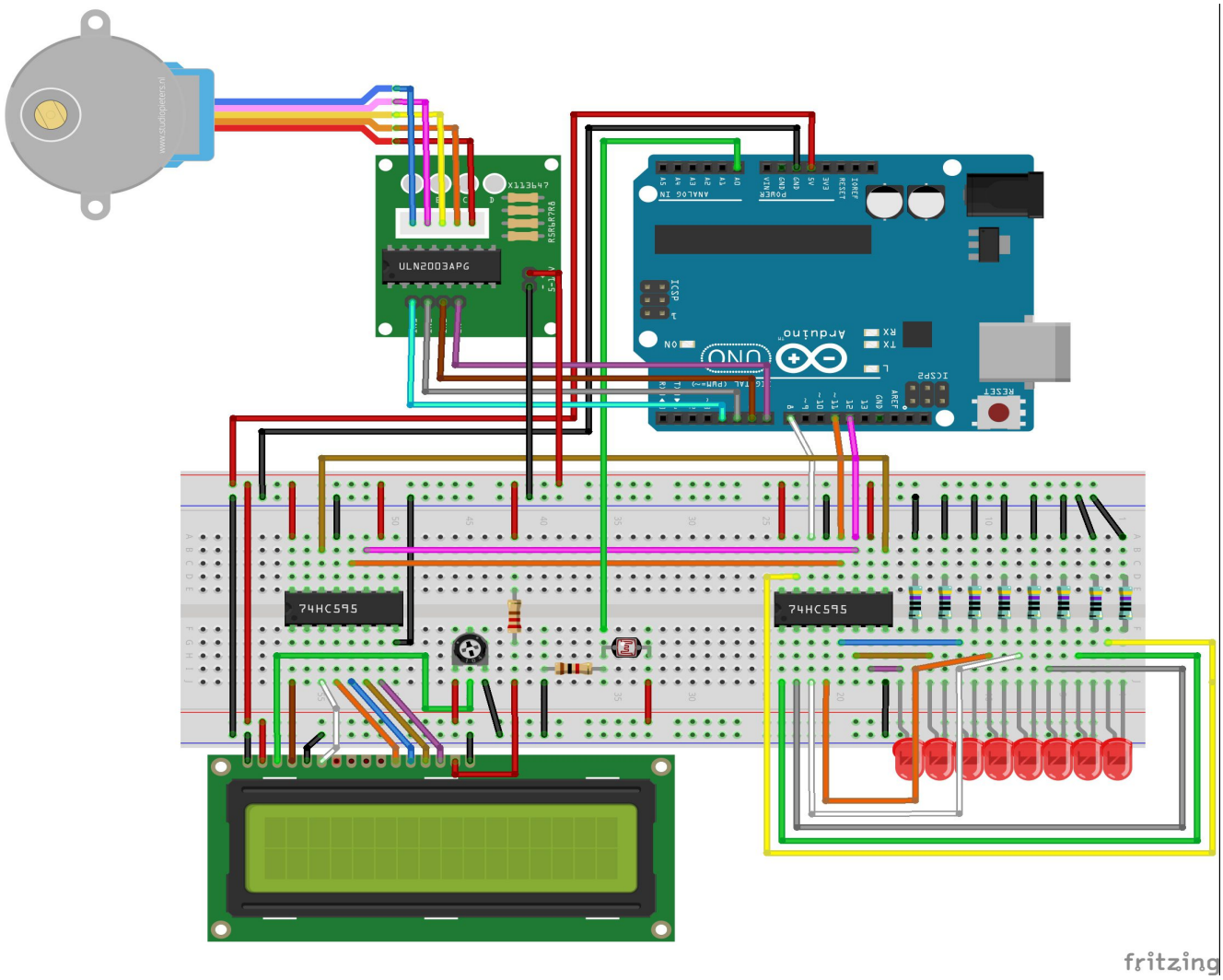
A questo punto diventa molto semplice aggiungendo poche cose, fare in modo che al di sopra di un certo valore di luminosità si metta in moto un motore per abbassare una tenda e al di sotto la renda venga riavvolta oppure che al di sotto di un valore venga accesa una lampada, vediamo il primo caso usando il motore passo passo (stepper motor) e la schedina per controllarlo:



Lo spinotto del motore ha un verso solo per l'inserimento nel modulo di controllo, e i 4 ingressi (IN1...IN4)li colleghiamo ai pin digitali da 4 a 7 di Arduino e questo è lo sketch:

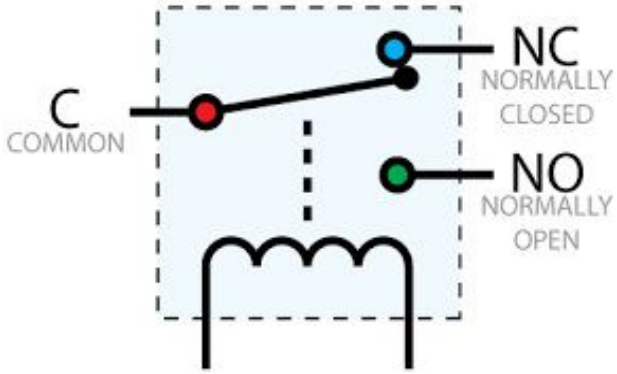
```
/*
 * _74Hc595LedLcd
 * utilizzo la libreria McMajan per gestire gli integrati
 * a cui ho connesso 8 led e un lcd
 * per visualizzare in forma scritta e visuale la luminosità rilevata da
 * una fotocellula connessa al pin A0 e uno stepper motor ai pin 4,5,6,7
 */

#include <Ss_McMajan_Config.h>
#include <Stepper.h>

const int passiPerGiro = 2048;
int velocita = 10;
bool suGiu = 0;

//inizializzo lo stepper collegato da 4 a 7
Stepper myStepper(passiPerGiro, 4, 5, 6, 7);

// creo l'oggetto My595 per gestire i 2 integrati
hc595 My595(11,12,8,2); // latch12,clock11,data14,numero di 74hc595

int ftc = 0;
int luce = 0;
unsigned char leds;

void setup(){
void loop(){
   int leggiFtc = analogRead(ftc);
   int numLedAccesi = leggiFtc / 101;            // max valore ftc= 910 / 9 combinazioni = 101
   if (numLedAccesi > 8) numLedAccesi = 8;
   leds=0;
   for (int i = 0; i< numLedAccesi; i++) {      // per ogni led acceso
      bitSet(leds,i);                           // metto a uno il bit corrispondente in leds
   }
   luce= map(leggiFtc,0,910,0,100);             // trasformo in valore da 0 a 100 la lettura analogica
   regWrite();

   // se il valore di luce è > 60% allora faccio fare 1 giro in senso orario al motore
   // 10 cicli da 21 passi per accelerare poi il resto
   if (luce > 60 && suGiu == 0) {                        // se suGiu = 0 allora la tenda è alzata
      for (velocita = 10; velocita <= 20; velocita++){   // posso abbassarla
         myStepper.setSpeed(velocita);
         myStepper.step(passiPerGiro/100);
         suGiu = 1;                                     // suGiu = 1 tenda abbassata
      }
      myStepper.step(passiPerGiro-210);
   }
   else if (luce < 61 && suGiu == 1){                    // se suGiu = 1 tenda abbassata
      for (velocita = 10; velocita <= 20; velocita++){   // posso alzarla
         myStepper.setSpeed(velocita);
         myStepper.step(-passiPerGiro/100);
      }
      myStepper.step(-(passiPerGiro-210));
      suGiu = 0;                                        // suGiu = 0 tenda alzata
   }
   // metto tutti i pin usati dallo stepper a 1
   for (int i =4 ; i<=7; i++) digitalWrite(i, LOW);     // in questo modo il motore dovrebbe scaldare di meno
   delay(500);
}

void regWrite(){

```

Le funzioni setup e regWrite sono uguali alle precedenti per cui le ho compresse, l'aggiunta va dalla riga 48 alla 63 e poteva benissimo essere messa in una funzione a parte. In questo caso uso una variabile di appoggio (suGiu) per sapere se ho già fatto girare il motore da una parte o dall'altra. Controllo il valore di

luce e se è maggiore del 60% e non ho già abbassato la tenda procedo a dare corrente al motore ed eseguire la rotazione dopodiché imposto suGiu ad 1 cioè tenda abbassata se nei prossimi cicli la luminosità cala al di sotto del 60% se la tenda è abbassata la alzo e reimposto suGiu a 0 cioè tenda alzata. La tenda può essere fatto in molti modi, per esempio con un motore (a corrente alternata o continua) e due sensori di fine corsa uno in fondo ed uno in cima al movimento oppure come in questo caso con uno stepper motor cioè un motore passo passo e sapendo quanti passi sono necessari per effettuare una rotazione completa dell'albero motore e quante rotazioni dobbiamo compiere per abbassare/alzare la tenda, possiamo controllarne la posizione in maniera molto precisa. Ovviamente serve un po di programmazione in più in quest'ultimo caso.

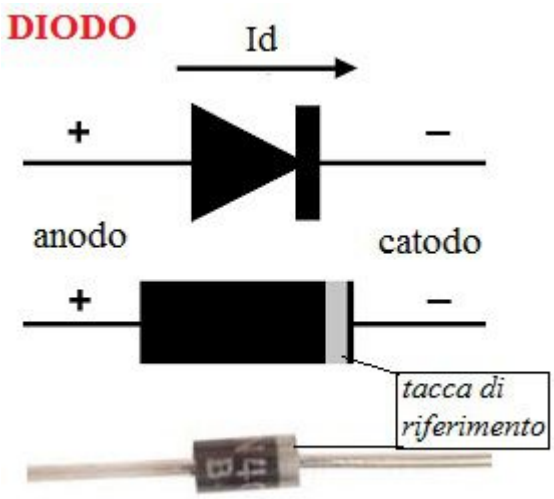
Il motore fornito nel kit è piuttosto piccolo e il moto riduttore interno (64:1) non ci consente di usare velocità superiori a 20-25 rpm (giri al minuto) e partendo con velocità superiori a 15 rpm ogni tanto si blocca, per supplire in parte a questo limite si deve aumentare gradatamente la velocità, in questo esempio ho usato molto sbrigativamente un ciclo for che esegue 11 scalini incrementando ogni volta di 1 unità la velocità, quindi 11 scalini per raddoppiare la velocità iniziale facendo compiere ogni volta 1/100 di rotazione, arrivati alla velocità finale viene completata la rotazione con i passi mancanti. E' buona consuetudine per qualsiasi tipo di motore passo passo utilizzare una rampa di accelerazione/decelerazione accurata, in quanto se ne guadagna in efficienza (minore riscaldamento) e in precisione (salto di passi alla partenza e precisione punto di arrivo).

Vediamo ora come usare un relay o relè.

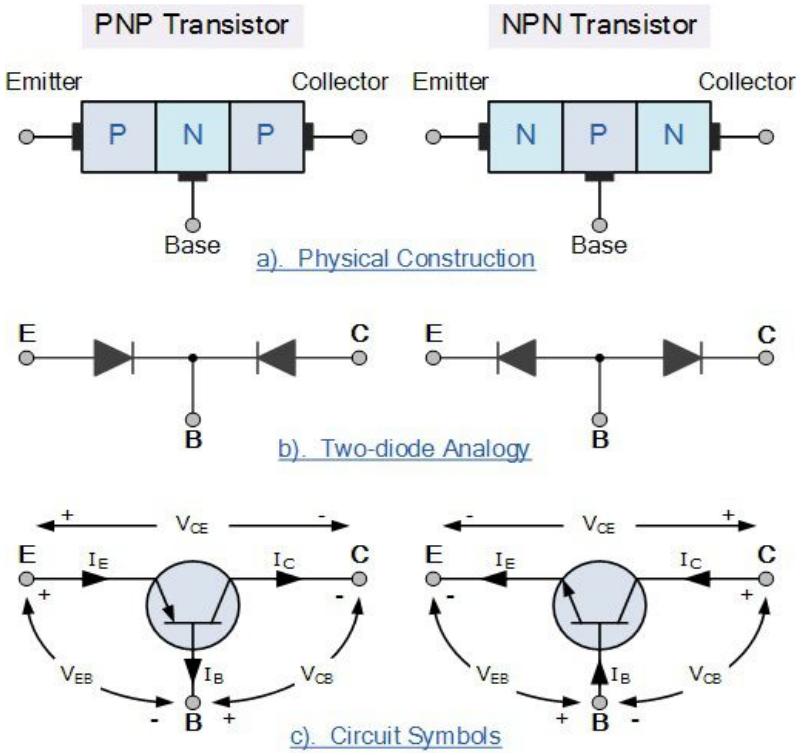


Questa è la rappresentazione di un relay semplice, c'è una bobina che con il suo nucleo forma una elettrocalamita e quando viene attraversata da corrente attira la lamina che chiude il contatto elettrico, quando la corrente cessa la lamina ritorna al suo stato di riposo aprendo il contatto. Esistono moltissimi tipi di relay ma in sostanza funzionano tutti allo stesso modo. Quindi collegando un capo della bobina a massa e l'altro capo ad un pin analogico di Arduino possiamo comandare l'accensione di un circuito esterno. L'inconveniente dell'utilizzo di questi relay risiede nel fatto che, come tutte le bobine (con o senza nucleo di ferrite), quando viene tolta la corrente provocano un picco di tensione di polarità opposta a come sono state alimentate, questa tensione se non viene bloccata può provocare danni all'uscita digitale collegata se non all'intero microcontrollore. Inoltre solitamente i relay necessitano di una corrente superiore ai 20 – 25 mA disponibili da Arduino, per cui nei casi come il nostro, dove abbiamo un relay tal quale, dobbiamo preparare un piccolo circuito, utilizzando un diodo come protezione, per eliminare i picchi di tensione allo sgancio del relay e un transistor da usare come interruttore che ci permette fornendogli una corrente minima (4 – 5 mA) proveniente da Arduino, di pilotare la bobina del relay con la

corrente adeguata (100 – 300 mA).
Vediamo questi due componenti prima di continuare.

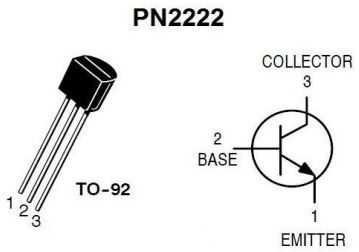


Come si vede il simbolo del diodo è simile a quello dei led, anch'esso ha una polarità da rispettare, in funzione del suo utilizzo, nel kit c'è un diodo con sigla 1N4007 come quello nella figura e la banda argentata ci dice da che parte è il terminale negativo. Se viene inserito in un circuito al rovescio, impedirà il passaggio di corrente come un interruttore aperto.
Transistor: ve ne sono di 2 tipi chiamati PNP (giunzione positivo, negativo, positivo) e NPN (giunzione Negativo, Positivo, Negativo)

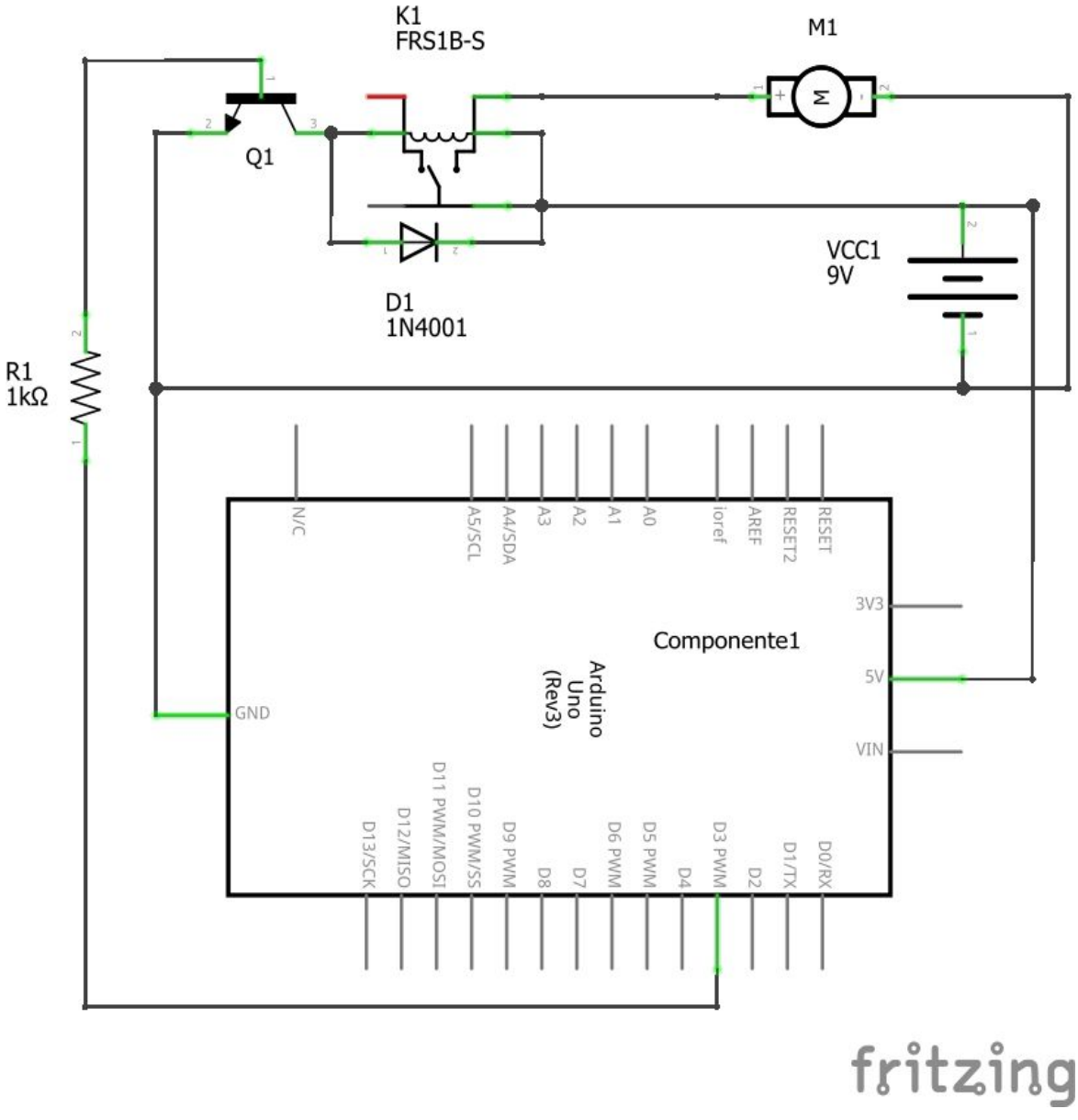


I transistors hanno diverse e più specifiche funzioni, per i nostri scopi il più semplice da usare è il tipo NPN e noi lo utilizzeremo come se fosse un interruttore elettronico, approfittando della sua caratteristica di amplificatore di corrente, infatti applicando una piccola corrente alla base avremo un flusso di corrente tra collettore ed emettitore pari al prodotto della corrente fornita e del fattore di amplificazione (beta)

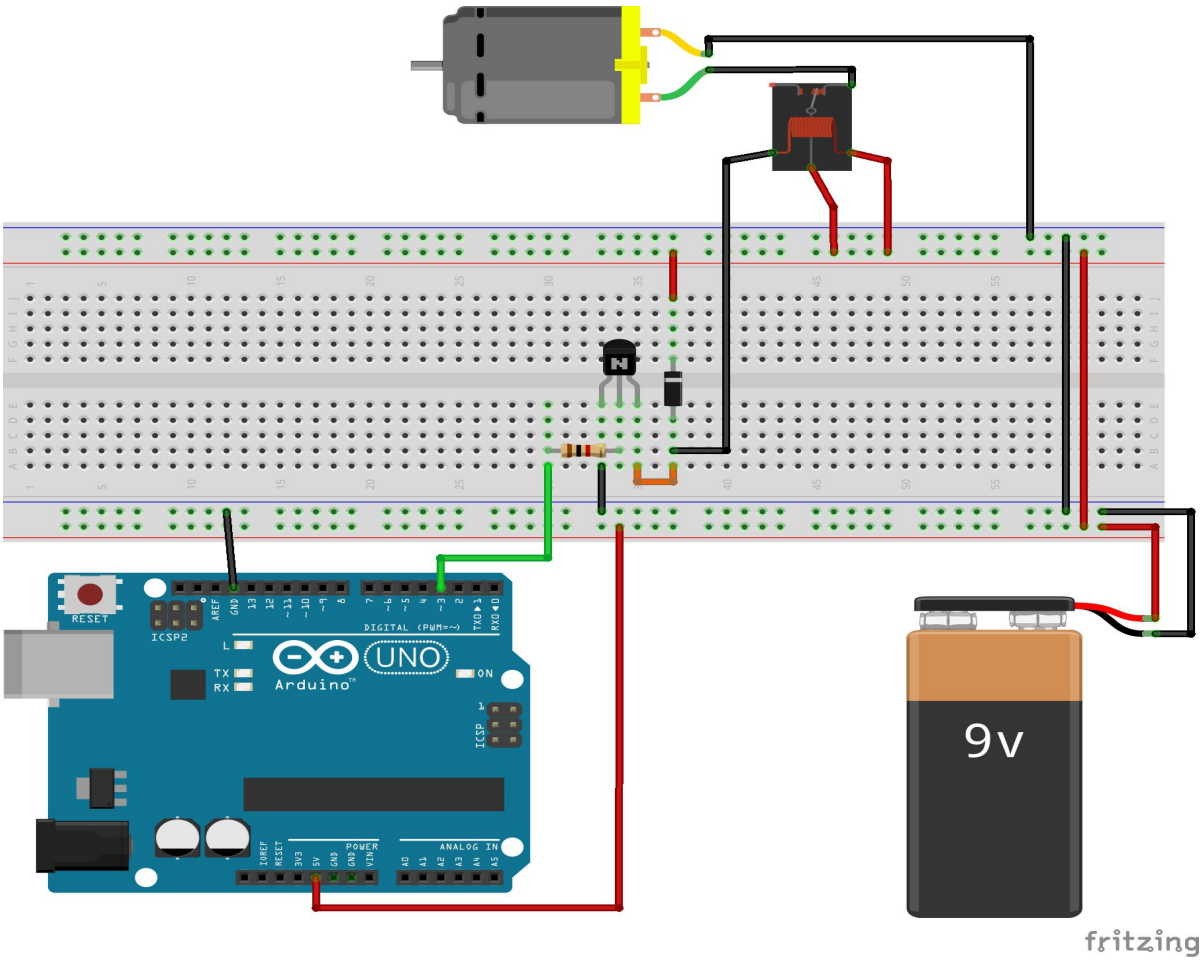
tipico del transistor utilizzato.
Questo è il transistor che abbiamo a disposizione nel kit, si tratta del PN2222.



Il datasheet ci dice che il fattore beta (hFE o DC current Gain) va da un minimo di 35 con correnti di 1 mA fino ad un max di 300 con 150 mA e che si può usare fino a 30V e 600 mA, quindi più che adatto al nostro scopo.
Vediamo lo schema del circuito



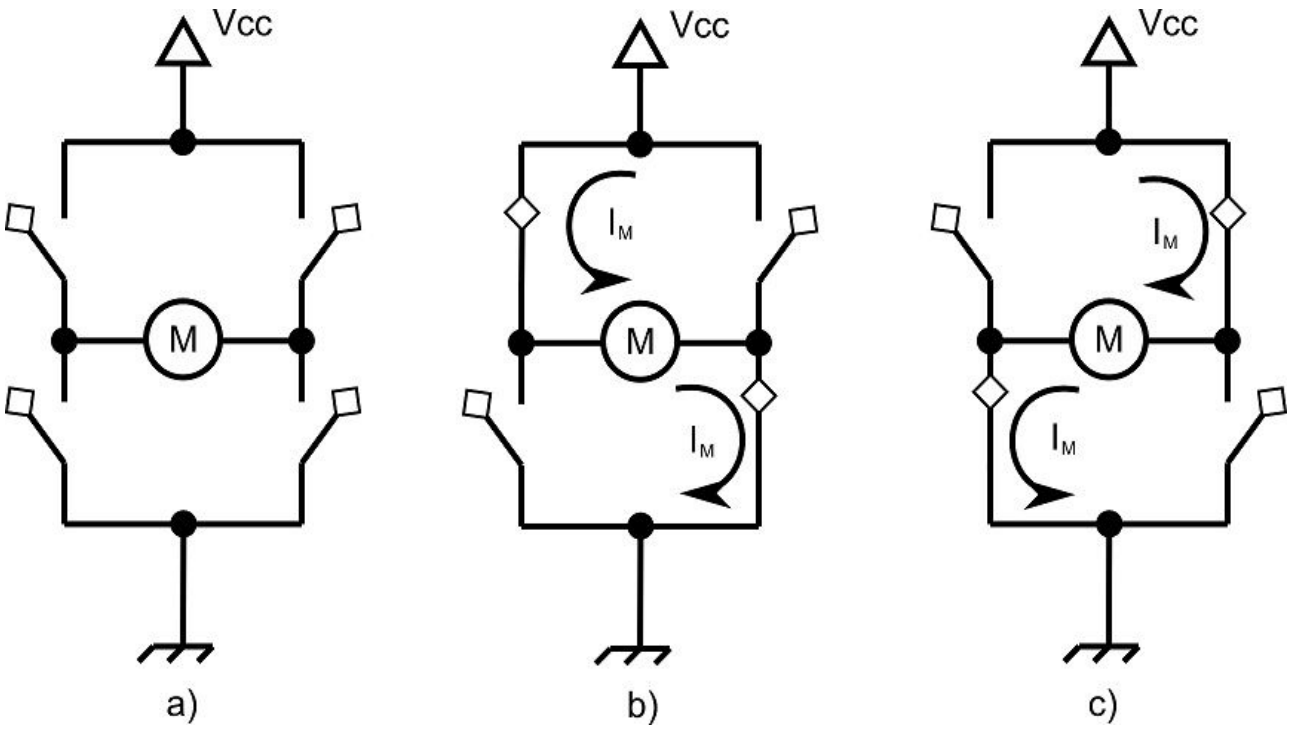
E questo è il circuito, attenzione! Il circuito non va alimentato con i 9 Volt ma con l'apposito convertitore a 5 Volt



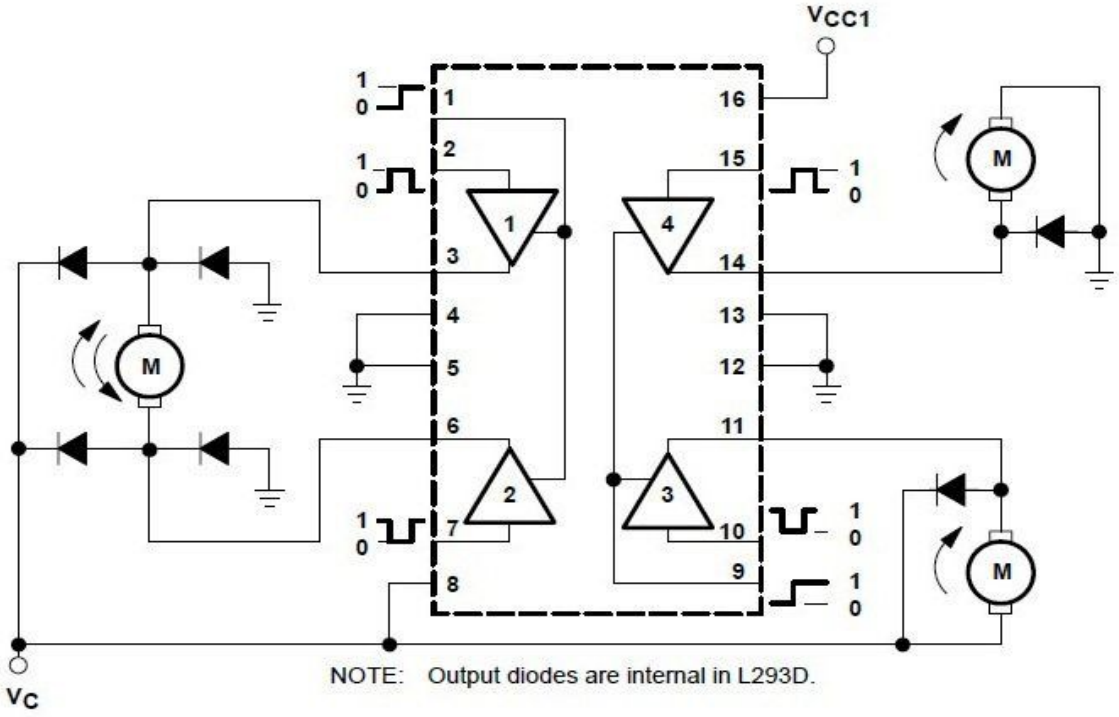
Il programma non presenta nulla di particolare in quanto si tratta solamente di impostare al valore alto l'uscita corrispondente al collegamento con la base del transistor, nel nostro caso il pin 3.

```
/*uso del relè con transistor 2n2222
 * collego la base del transistor al pin 3
 * e lo uso come interruttore per eccitare la bobina del relè
 * al quale collego il carico (qui il motorino dc)
 */

#define baseTr 3

void setup() {
  pinMode(baseTr,OUTPUT);
}

void loop() {
    digitalWrite(baseTr,HIGH); //do corrente alla base
    delay(1750);
    digitalWrite(baseTr,LOW);  //tolgo corrente alla base
    delay(1750);
    digitalWrite(baseTr,HIGH);
    delay(1750);
    digitalWrite(baseTr,LOW);
    delay(3000);
}

```

Rendiamo un po più interessante il controllo del motore in corrente continua utilizzando l'integrato nel kit che si chiama L295D, si tratta di un quadruplo driver a mezzo ponte H ad alta corrente. Cosa significa questo? Semplificando notevolmente si chiama ponte H il sistema per comandare un motore usando quattro connessioni attivabili a comando secondo questo schema:

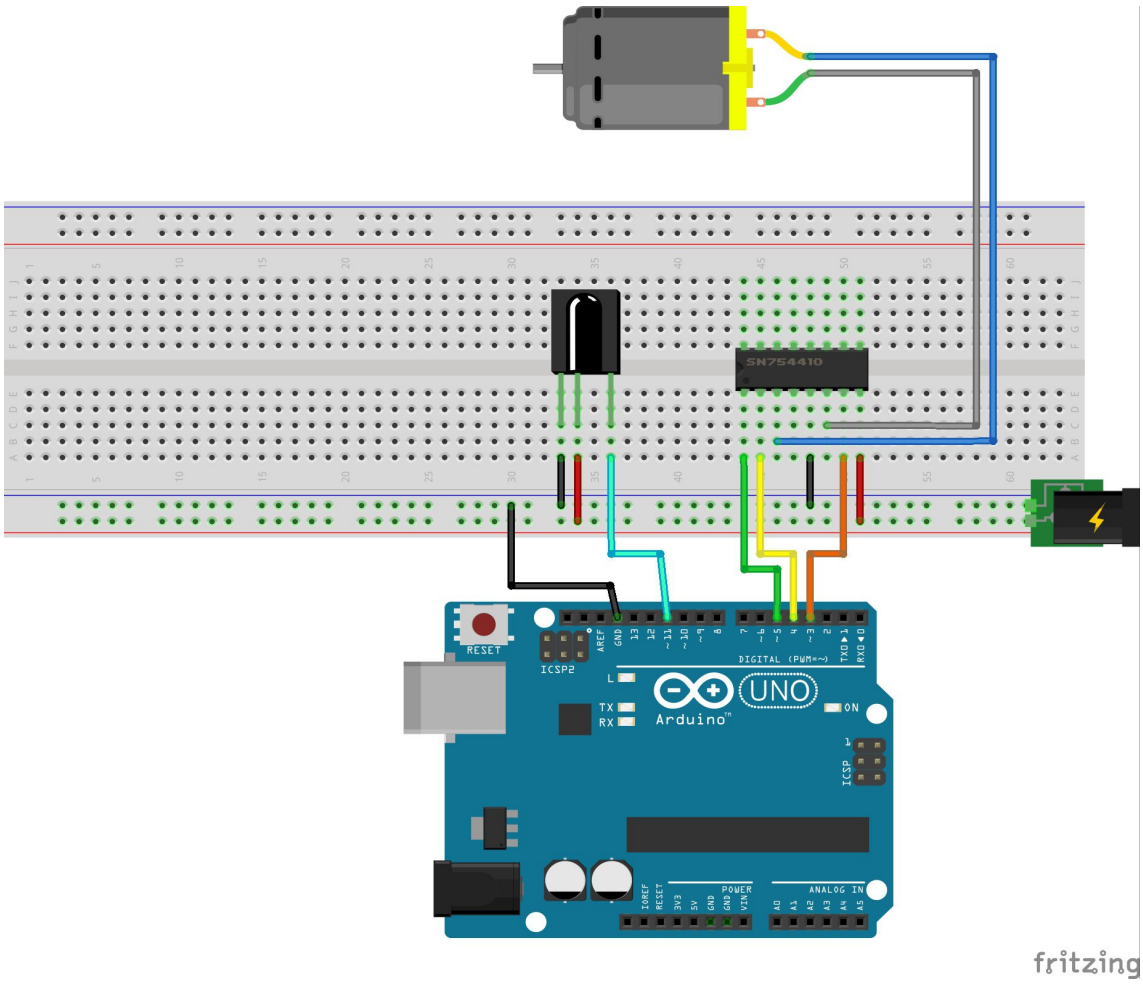


come si vede, dalla figura a) dove non circola corrente si può connettere il motore secondo b) oppure c) ottenendo i due sensi di circolazione, è importante NON connettere mai i due interruttori a destra o i due a sinistra contemporaneamente pena causare un cortocircuito. Il nostro integrato esegue queste connessioni in modo trasparente. Vediamo lo schema interno e come viene connesso ad un motore:



Possiamo notare che è diviso in 4 sezioni (i 4 triangoli) e che ognuna può comandare un singolo motore rotante in un solo verso oppure 2 motori con la possibilità di invertire il senso di rotazione. Useremo le sezioni 1 e 2 collegate al motore DC visto prima come raffigurato nella metà sinistra dello schema qui sopra. Tutti i diodi presenti nello schema sono già inclusi nell'integrato che ha la lettera D finale, se viene usato quello seza la D, vanno aggiunti per proteggere sia l'integrato stesso che Arduino dalle correnti parassite dovute alle bobine del motore (come per il relay). Per complicarci un po la vita, aggiungiamo la possibilità di comandare da remoto il nostro motorino, aggiungendo il sensore ad infrarossi (IR) al nostro circuito.

Si tratta di un sensore molto semplice da usare che possiede solo 3 contatti: positivo (R), negativo (G) e segnale (Y) e va collegato ai pin +5V, GND, e un pin digitale a nostra scelta. In questo modo il circuito diventa:

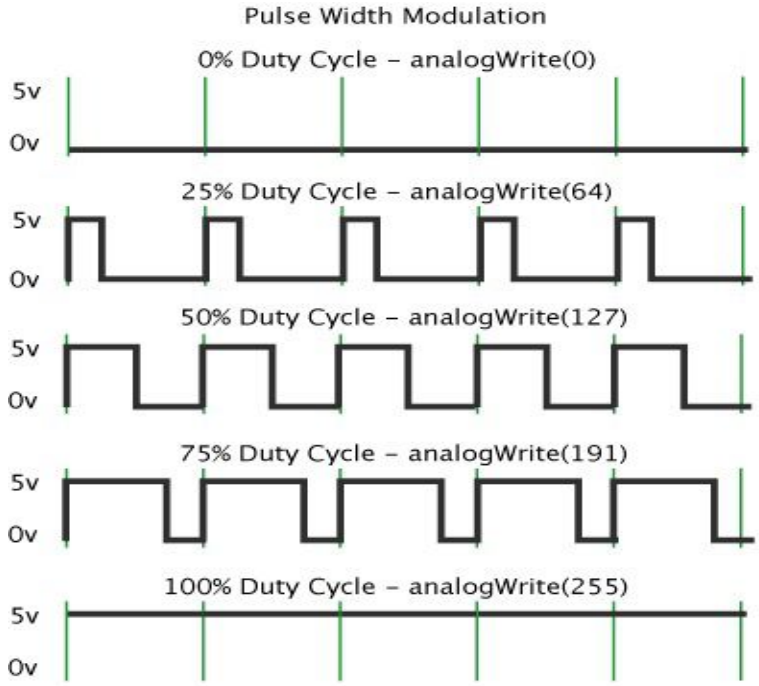


Al quale abbiniamo lo sketch:

```
/*
 * 30-DCMotor
 * Controlliamo il motorino in CC nom l'integrato L293D
 * e lo azioniamo tramite sensore IR e telecomando
 * sensore ir collegato al pin 11
 * motore DC collegato ai pin 3, 4, 5 di arduino
 */

// sezione sensore IR
#include "IRremote.h"
int pinIR = 11;                         //segnale del ricevitore ir sul pin 11
// inizializzazione del sensore IR
IRrecv mioIR(pinIR);                    //creo istanza del sensore ir
decode_results codice;                  //creo instanza per decodifica segnali ricevuti

// sezione motore DC
int startMotore = 0;
int direzione = 0;
int velocita = 9;
const int ENABLE = 5;
const int DIRA = 3;
const int DIRB = 4;

void setup(){
  Serial.begin(9600);
  Serial.println("Decodifica segnali ricevuti dal sensore IR");
  mioIR.enableIRIn();                   // abilito il sensore IR
  pinMode(ENABLE,OUTPUT);               // pin abilitazione motore
  pinMode(DIRA,OUTPUT);                 // pin direzione A
  pinMode(DIRB,OUTPUT);                 // pin direzione B
}

void loop(){
  if (mioIR.decode(&codice)) {      //se abbiamo ricevuto un segnale
    traduciIR();
    eseguiComando();
    mioIR.resume();             /     /rilevo il prossimo segnale
  }
}

void traduciIR(){             //esegue azione a seconda del tasto premuto
  switch(codice.value){
    case 0xFFA25D: Serial.println("POWER"); startMotore = !startMotore;     break;
    case 0xFFE21D: Serial.println("FUNC/STOP");                            break;
    case 0xFF629D: Serial.println("VOL+");                                 break;
    case 0xFF22DD: Serial.println("FAST BACK"); direzione = 1;             break;
    case 0xFF02FD: Serial.println("PAUSE");                                break;
    case 0xFFC23D: Serial.println("FAST FORW."); direzione = 0;            break;
    case 0xFFE01F: Serial.println("DOWN");                                 break;
    case 0xFFA857: Serial.println("VOL-");                                 break;
    case 0xFF906F: Serial.println("UP");                                   break;
    case 0xFF9867: Serial.println("EQ");                                   break;
    case 0xFFB04F: Serial.println("ST/REPT");                              break;
    case 0xFF6897: Serial.println("0"); velocita=0;                        break;
    case 0xFF30CF: Serial.println("1"); velocita=1;                        break;
    case 0xFF18E7: Serial.println("2"); velocita=2;                        break;
    case 0xFF7A85: Serial.println("3"); velocita=3;                        break;
```

```
    case 0xFF10EF: Serial.println("4"); velocita=4;                    break;
    case 0xFF38C7: Serial.println("5"); velocita=5;                    break;
    case 0xFF5AA5: Serial.println("6"); velocita=6;                    break;
    case 0xFF42BD: Serial.println("7"); velocita=7;                    break;
    case 0xFF4AB5: Serial.println("8"); velocita=8;                    break;
    case 0xFF52AD: Serial.println("9"); velocita=9;                    break;
    case 0xFFFFFFFF: Serial.println("REPEAT");                        break;
  default:
    Serial.println(" other button   ");
  }
  delay(500);
}

void eseguiComando(){
  int miaVelocita= map(velocita,0,9,120,255);

  if (startMotore){
    if (direzione==0){
      analogWrite(ENABLE,miaVelocita);          //abilito partenza
      digitalWrite(DIRA,HIGH);                  //eseguo direzione A
      digitalWrite(DIRB,LOW);
    }
    else{
      analogWrite(ENABLE,miaVelocita);          //abilito partenza
      digitalWrite(DIRA,LOW);                   //eseguo direzione B
      digitalWrite(DIRB,HIGH);
    }
  }
  else digitalWrite(ENABLE,LOW);
}
```

Anche in questo caso l'utilizzo di una libreria rende veloce l'utilizzo del sensore IR, usiamole due funzioni IRrecv mioIR(pinIR) e decode_results codice per creare l'oggetto mioIR e memorizzare il codice esadecimale ricevuto dal sensore dopodiché nel setup con il comando mioIR.enableIRIn() abilitiamo il sensore alla ricezione e con mioIR.decode(&codice) recuperiamo il codice del tasto premuto sul telecomando, a questo punto il ricevitore va in pausa, per renderlo operativo e pronto alla ricezione di un nuovo codice dobbiamo usare la funzione mioIR.resume(). Nello sketch ho lasciato tutti i codici esadecimali corrispondenti a tutti i tasti del telecomando che vengono visualizzati sul monitor seriale (se aperto).
La gestione del motore è molto semplice, il pin 5 di Arduino è collegato al pin di abilitazione dell'integrato e quando questo valore è alto (se comandato con digitalWrite()) il motore può funzionare, quindi ponendo alto il pin 4 e basso il pin 3 il motore ruoterà in un senso se invertiamo, pin 4 basso e pin 3 alto, ruoterà all'inverso. Rimettendo a livello basso il pin 5 il motore verrà disabilitato e si fermerà.
Usiamo però un metodo leggermente diverso per abilitare il motore, infatti usiamo tramite il pin 5, un segnale PWM (pulse width modulation) che è un tipo di modulazione digitale che permette di ottenere una tensione media variabile dipendente dal rapporto tra la durata dell'impulso positivo e di quello negativo in questo modo:



Vediamo qui 5 esempi di modulazione PWM con il valore corrispondente da assegnare all'istruzione analogWrite(). Nello sketch converto i valori del telecomando da 0 a 9 in un numero che va da 120 a 255 con l'istruzione map() che abbiamo già visto in passato, parto dal valore 120 in quanto ho verifivato che a valori inferiori il motore non riesce a ruotare, cambiando tipo/qualità di motore il valore può essere sicuramente inferiore.