ARDUINO

MOTORI E LCD

PROF. NACLERIO PASQUALE

MOTORI PER ARDUINO

ESISTONO MOLTI TIPI DI MOTORI, CIÒ CHE ARDUINO PUÒ DARE IN PIÙ È IL CONTROLLO SU DI LORO DELLA ROTAZIONE E DELLA VELOCITÀ.

PROF. NACLERIO PASQUALE

TIPI DI MOTORI

Due categorie principali:

- Motori in corrente continua DC
- Motori in corrente alternata AC

NOI CI CONCENTRIAMO SUI MOTORI IN DC

- Ci serve precisione
- Controllo (velocità di rotazione e/o angolo di rotazione)
- Facilità di istallazione

MOTORE PASSO – PASSO MONOPOLARE (STEPER MOTOR)

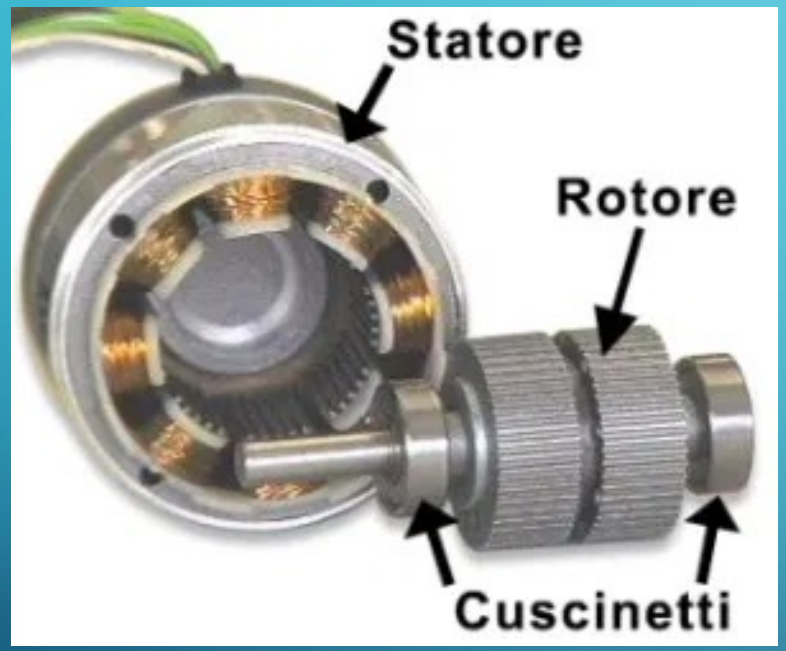
CONTROLLIAMO LA VELOCITÀ

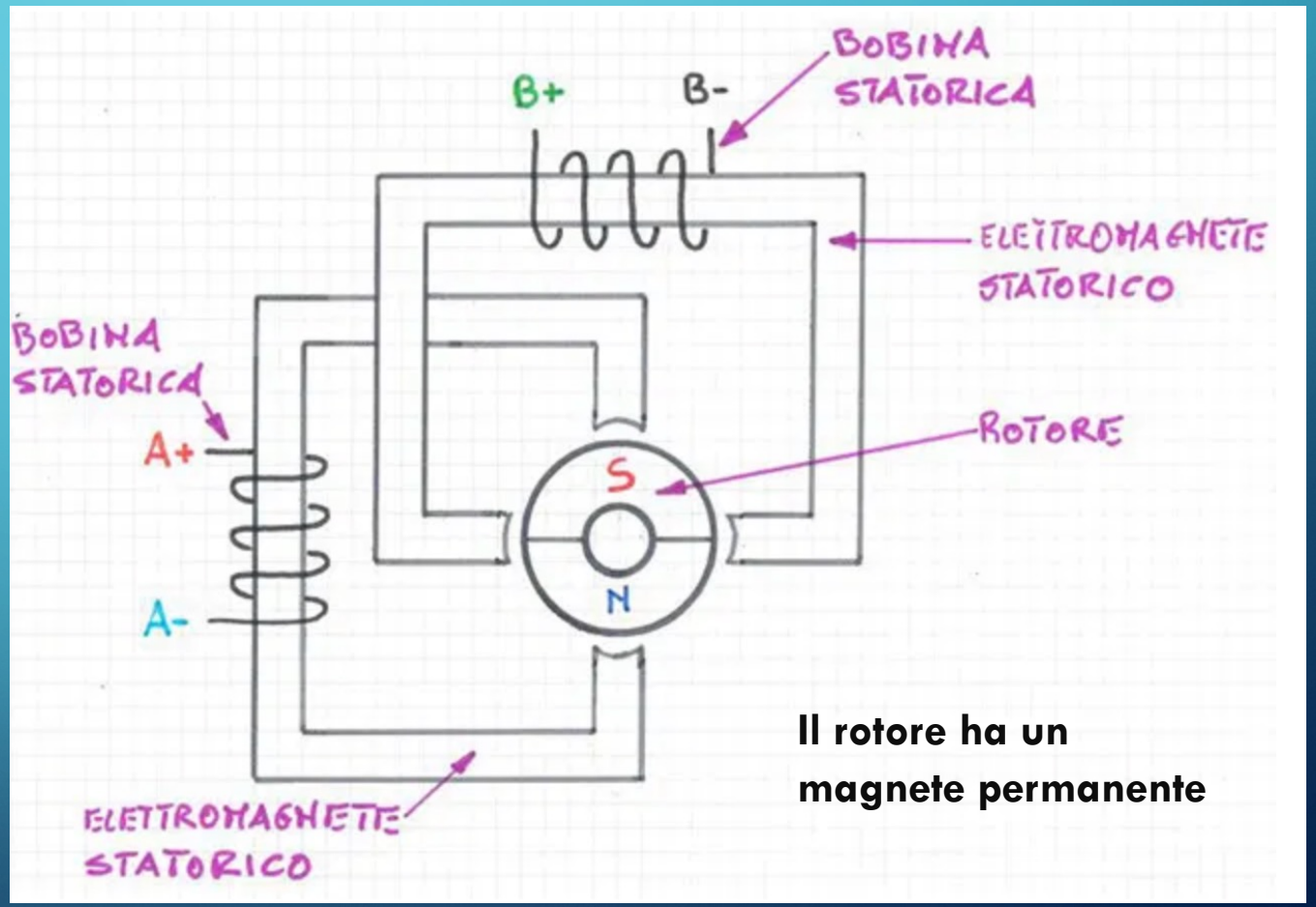
PROF. NACLERIO PASQUALE

STEPER MOTOR

- Motore il cui avanzamento avviene con angoli precisi, per cui è facile controllare con estrema precisione la velocità di rotazione.
- Molto usati nell'automazione industriale, nella robotica, nelle stampanti.
- Mantiene la coppia motrice costante anche se cambia la velocità

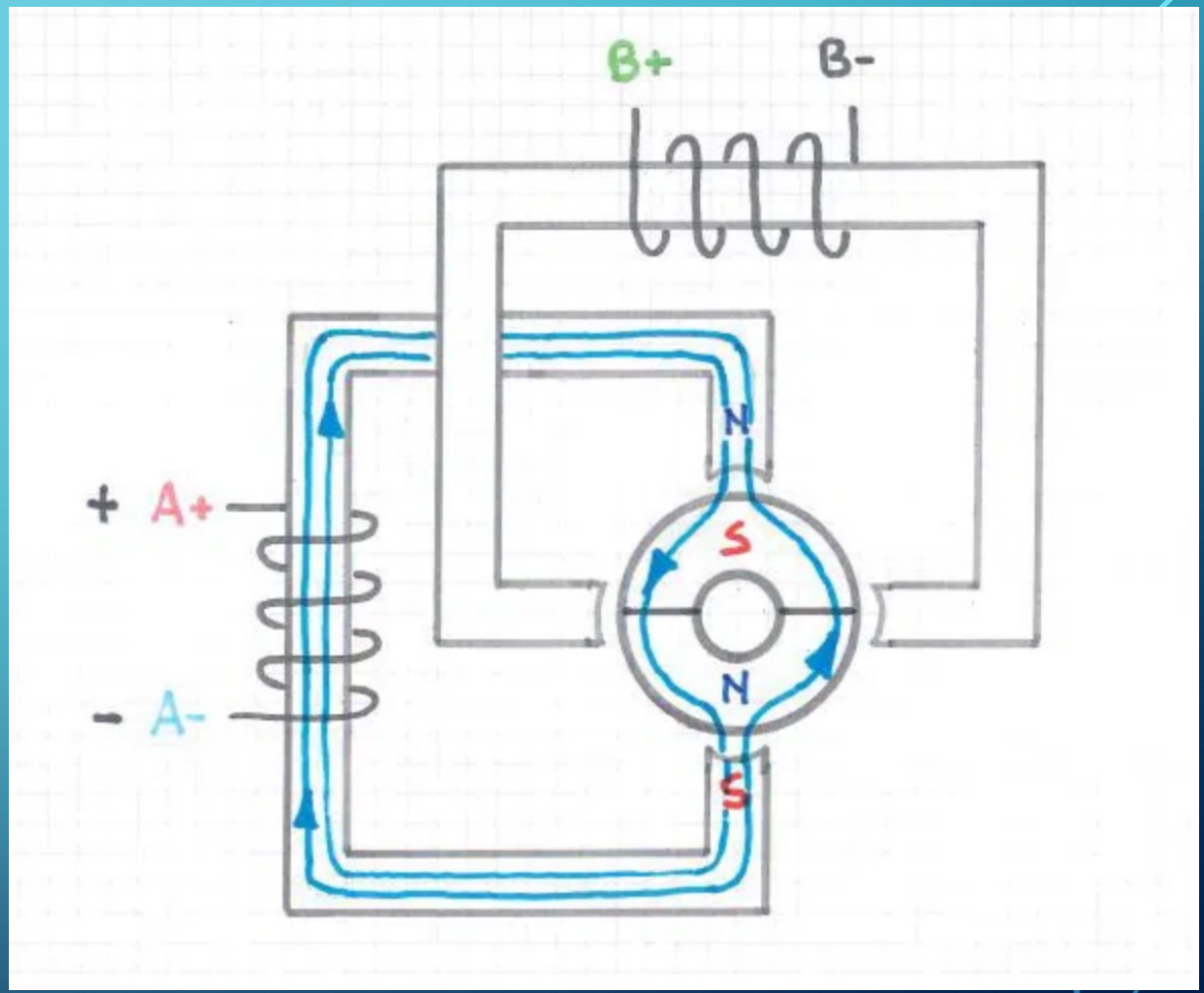
COME È FATTO E COME FUNZIONA





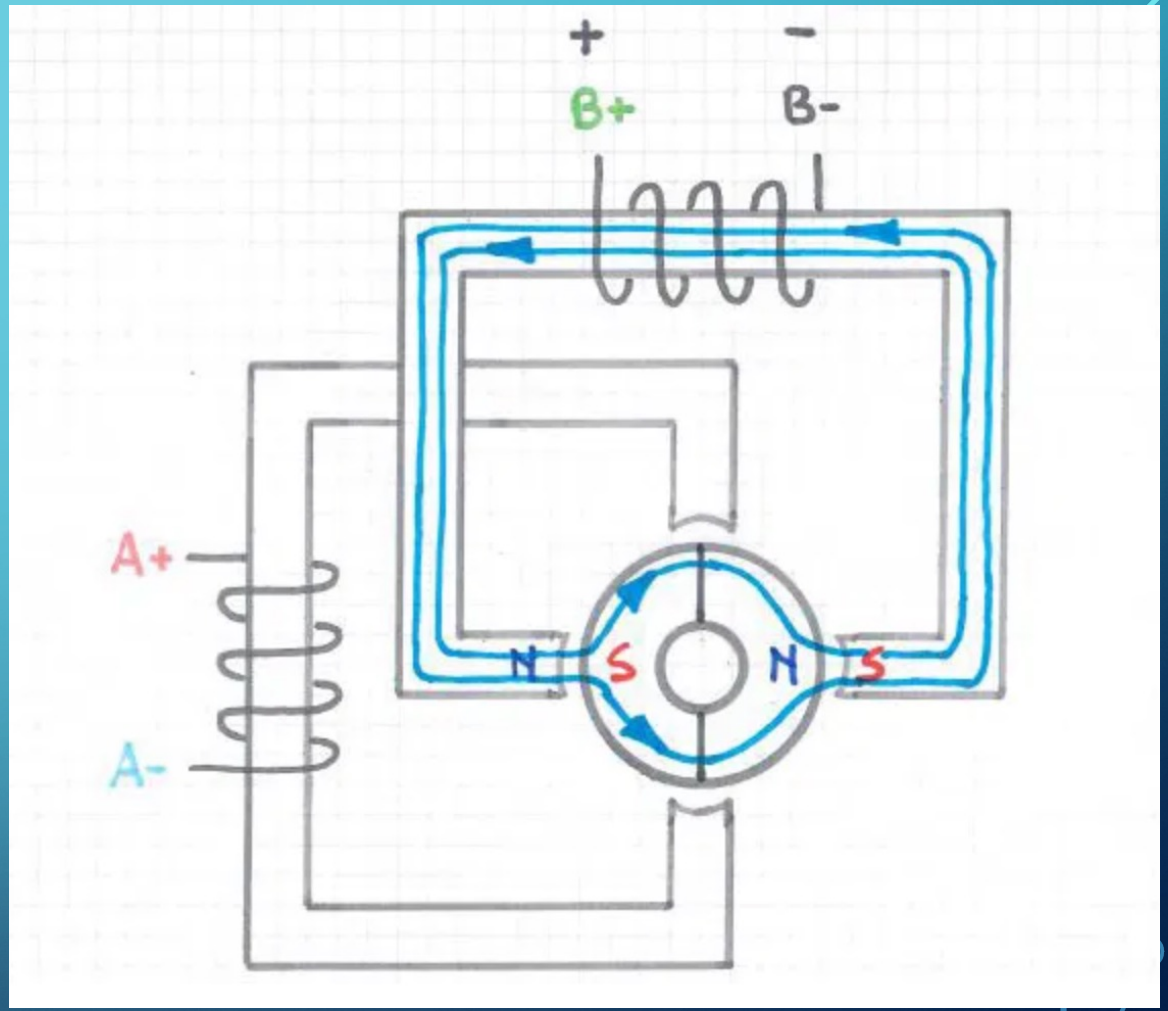
FUNZIONAMENTO 1

Facendo circolare una corrente continua nell'avvolgimento statorico A+ e A- viene generato un campo magnetico che porterà il rotore a ruotare e a bloccarsi nella posizione in cui le polarità magnetiche saranno opposte



FUNZIONAMENTO 2

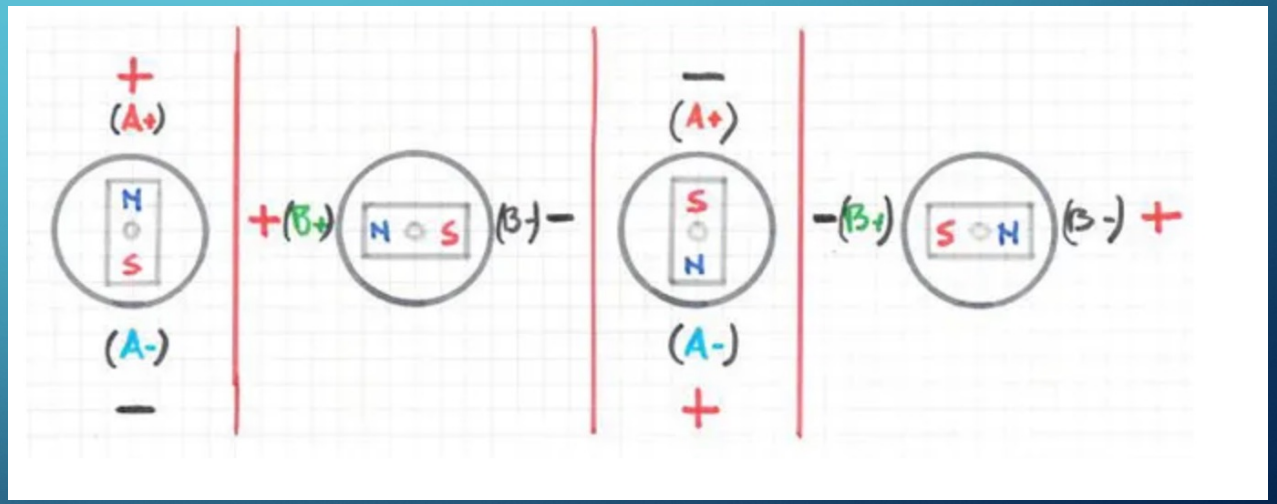
Togliendo alimentazione all'avvolgimento statorico A+ A- e alimentando l'avvolgimento statorico B+ B- si avrà una configurazione differente dei magneti statorici ed una conseguente rotazione di 90° del rotore in senso antiorario



COMANDI

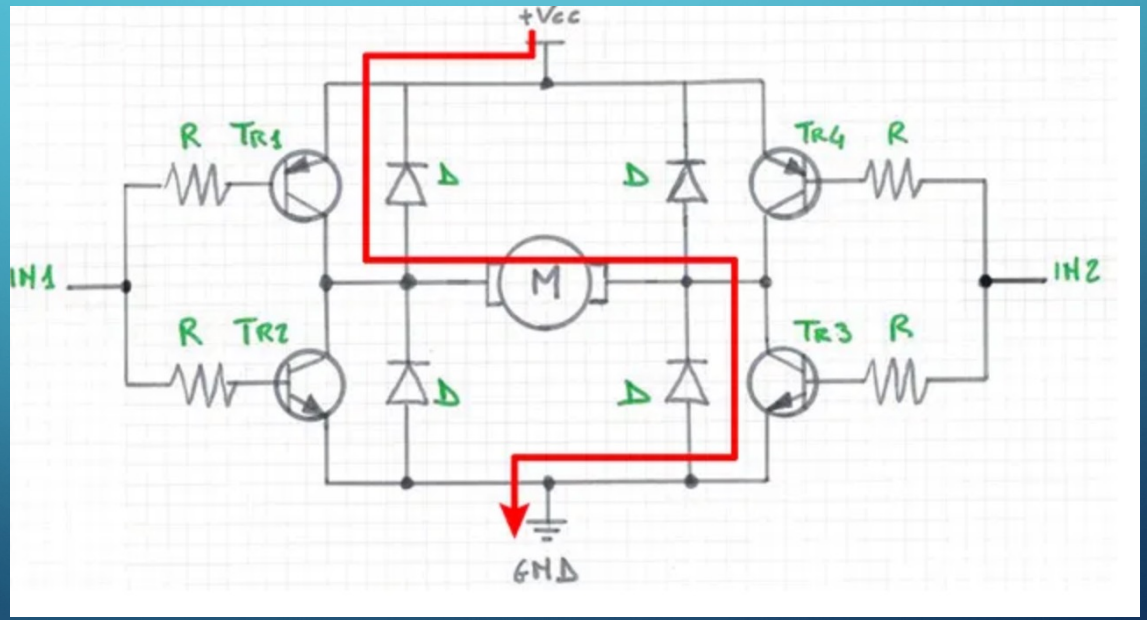
Serve una giusta sequenza di comandi per far ruotare il motore e bisogna seguire la giusta sequenza.

FASE	A+	A-	B+	B-
1	+	-	0	0
2	0	0	+	-
3	-	+	0	0
4	0	0	-	+



IL GIUSTO CIRCUITO

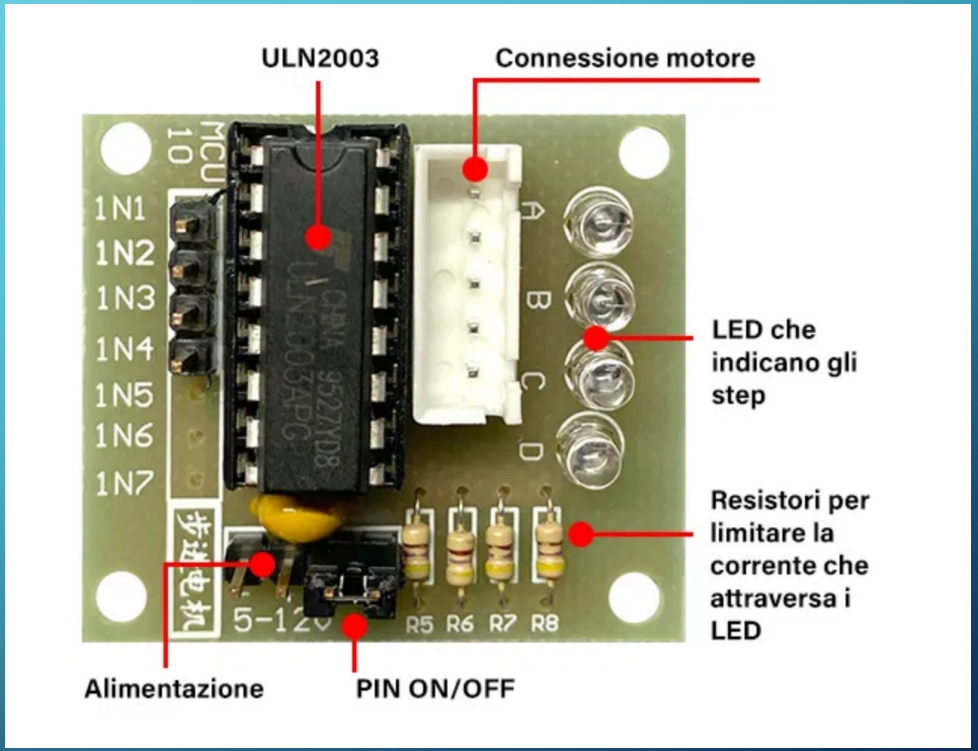
Da queste osservazioni possiamo vedere come servano non solo appositi comandi, ma anche apposti circuiti che permettano a questi motori di ruotare ed essere controllati.



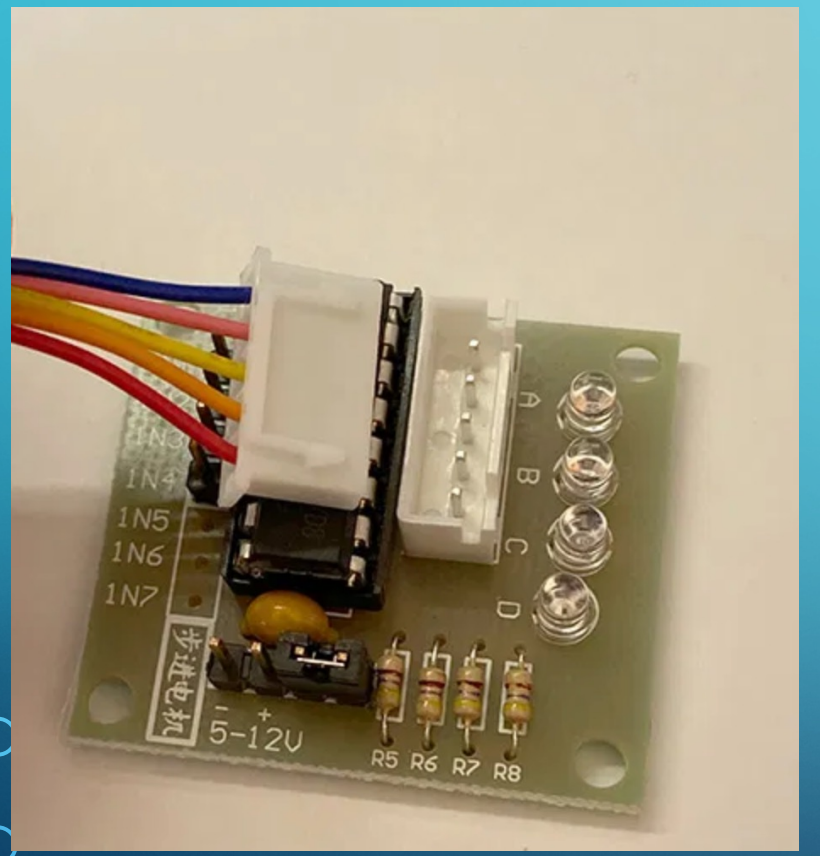
CIRCUITI DI SUPPORTO

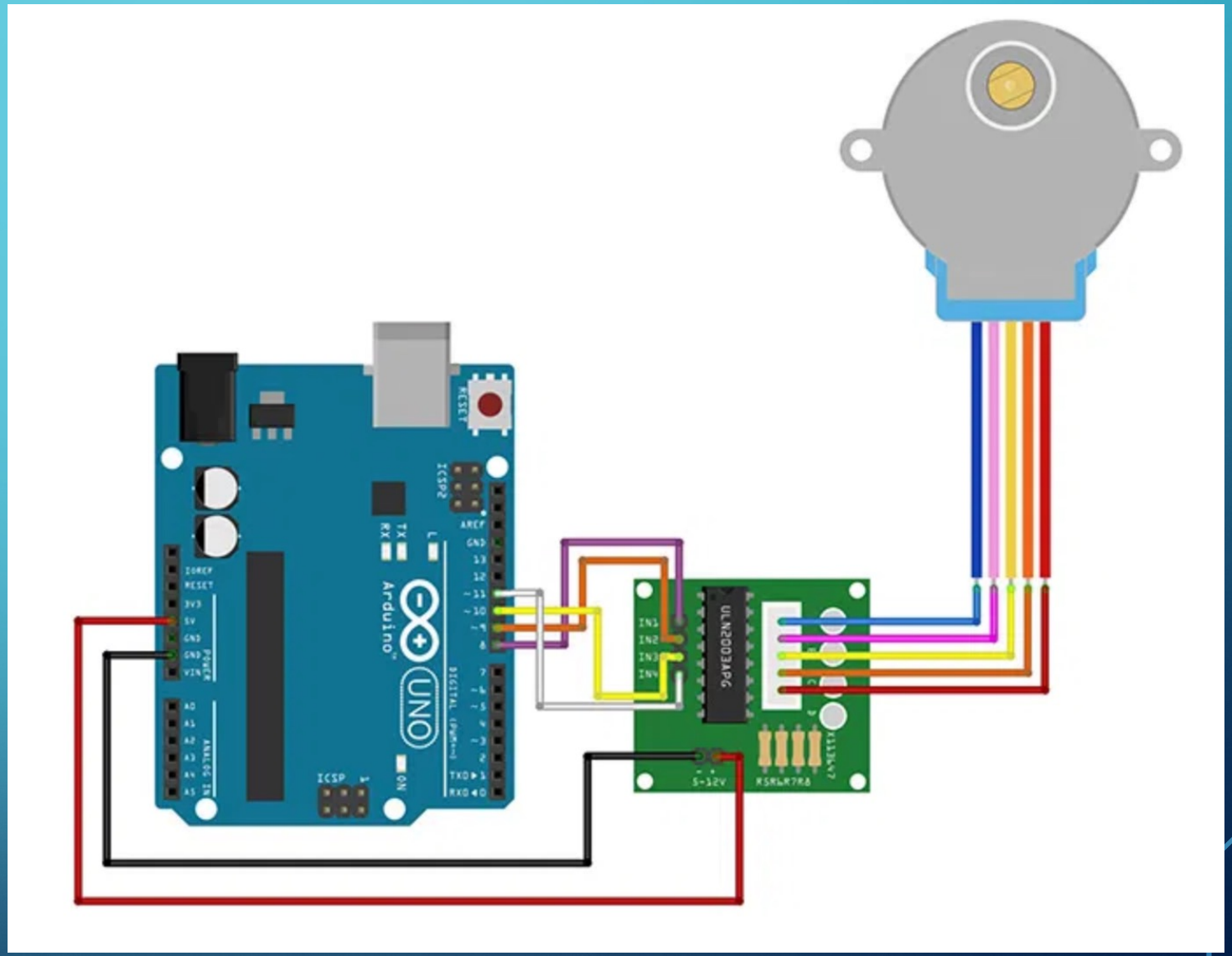
Bisogna dare i giusti segnali con i giusti circuiti. Per fare questo ci vengono in aiuto due cose.

1. Un apposito circuito
2. Le librerie di arduino



COLLEGAMENTO ULN2003A





LIBRERIE DI ARDUINO – ARDUINO STEPPER

I costruttori di Arduino ci hanno già donato tutto il necessario per far funzionare in modo semplice i nostri motori.

La libreria : #include<Stepper.h>

Ora dobbiamo adattare la libreria al nostro motore.

NUMERO DI PASSI

Noi useremo un motore 28BYJ-48 che ad ogni passo fa una rotazione di 11,25°

Quindi una rotazione completa è fatta di 32 passi: 360°/11,25° = 32 passi

Dentro al motore c'è un un riduttore da 1/64 e quindi in realtà fa molti più passi reali cioè 32*64 = 2048 passi teniamo a mente questo numero!

Il tempo per ogni passo è di 2 ms e quindi 1/0,002 passi/s = 500 passi/s è la sua velocità massima.

Questo motore per fare un giro completo ci metterà 2048*0,002 = 4 secondi.

SPIEGHIAMO AD ARDUINO IL MOTORE

Dobbiamo spiegare il tipo di motore ad arduino:

Dobbiamo creare quello che in programmazione si chiama classe

```
Stepper nome_mio_motore( numero_passi_rotazione, pin1, pin2, pin3, pin4);
```

- Dare un nome
- Il numero di passi per rotazione 2048
- A che pin ho collegato il motore IN1 con 8, IN2 con 9, IN3 con 10, IN4 con 11

NB possiamo usare solo i pin digitali e con un ordine preciso.

```
Stepper pino( 2048, 8, 10, 9, 11);
```

APPARECCHIAMO LA TAVOLA

Non serve dare molte altre spiegazioni ad arduino, se non cosa la velocità che dovrà raggiungere il motore.

Quindi dentro il setup dobbiamo dirglielo:

```
void setup() {

  nome_motore.setSpeed(15);

}
```

Il 15 sta per 15rpm = 15 giri/minuto

1 giro/min = 0,0167 giro/sec

1 giro/sec = 60 giri/min

il motore gira a ¼ della sua massima velocità, quindi farà un giro completo in 16 s

QUANTO VUOI FARLO GIRARE?

Ora dobbiamo solo dire al motore quanti passi deve fare:

1024 = 2048/2 fa mezzo giro in senso orario

-1024 fa mezzo in senso antiorario

```
11 void loop() {
12
13  nome_motore.step(passi_che_deve_fare);
14
15
16 }
```

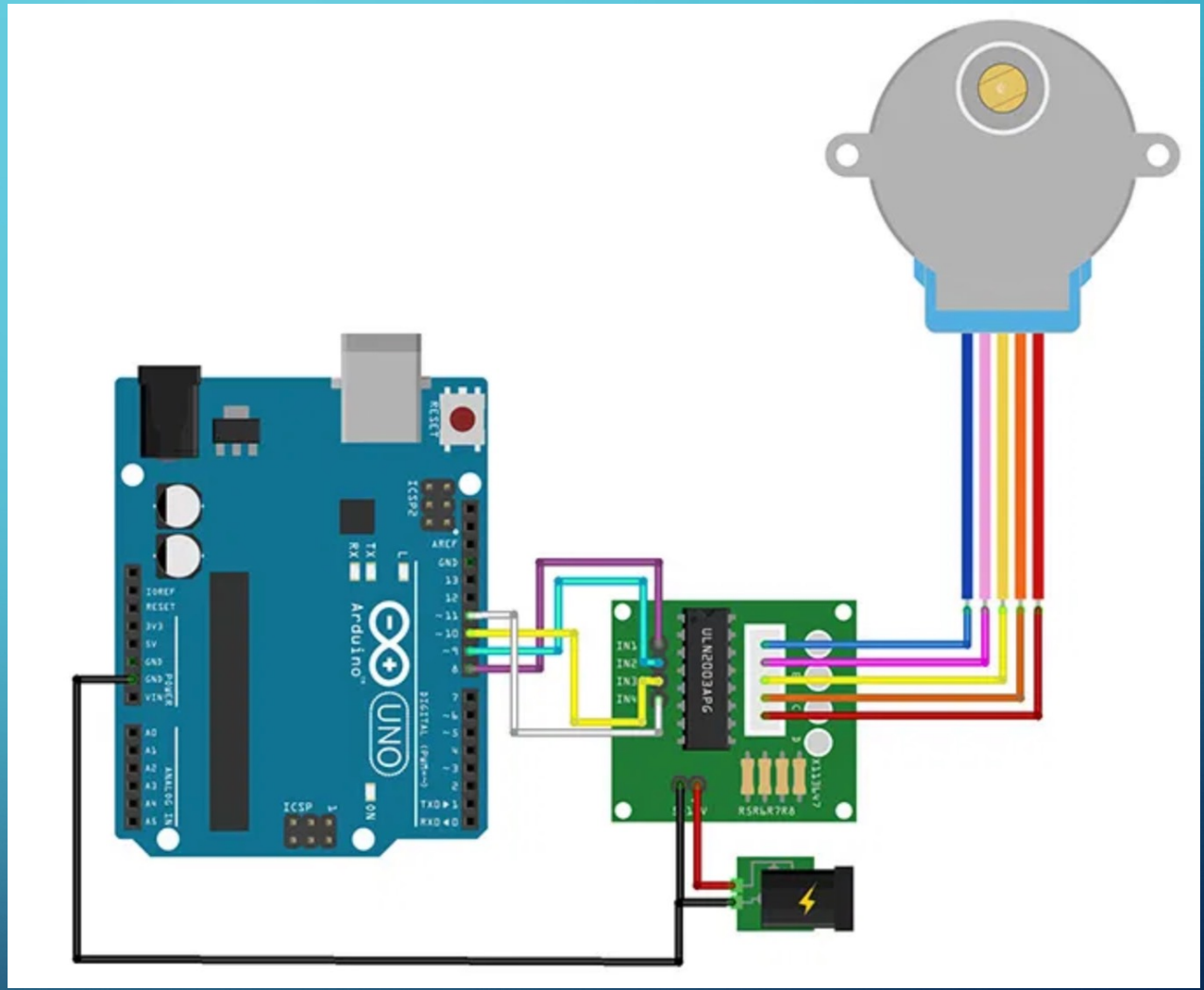
```
11 void loop() {
12
13  pino.step(1024);
14
15  pino.step(-1024);
16
17 }
```

MOTORE CHE VA AVANTI E INDIETRO

```
#include <Stepper.h>

const int stepsPerRevolution = 2048;


Stepper myStepper(stepsPerRevolution, 8, 10, 9, 11);

void setup() {
  myStepper.setSpeed(15);

  Serial.begin(9600);
}

void loop() {
  Serial.println("orario");
  myStepper.step(stepsPerRevolution);
  delay(500);

  Serial.println("antiorario");
  myStepper.step(-stepsPerRevolution);
  delay(500);

}
```

PROF. NACLERIO PASQUALE

A VOLTE SERVE ALIMENTARE I MOTORI DALL'ESTERNO



PROF. NACLERIO PASQUALE

SERVOMOTORE

CONTROLLIAMO L'ANOGOLO

PROF. NACLERIO PASQUALE

CHE COSA È?

È una scatolina che contiene:

- un semplice motore DC
- Un potenziometro per rilevare la posizione
- Ingranaggi per aumentare la potenza meccanica
- Piccolo circuito di pilotaggio

Permette di compiere precise rotazioni di 270° ed è in grado di sollevare qualche kg di peso.

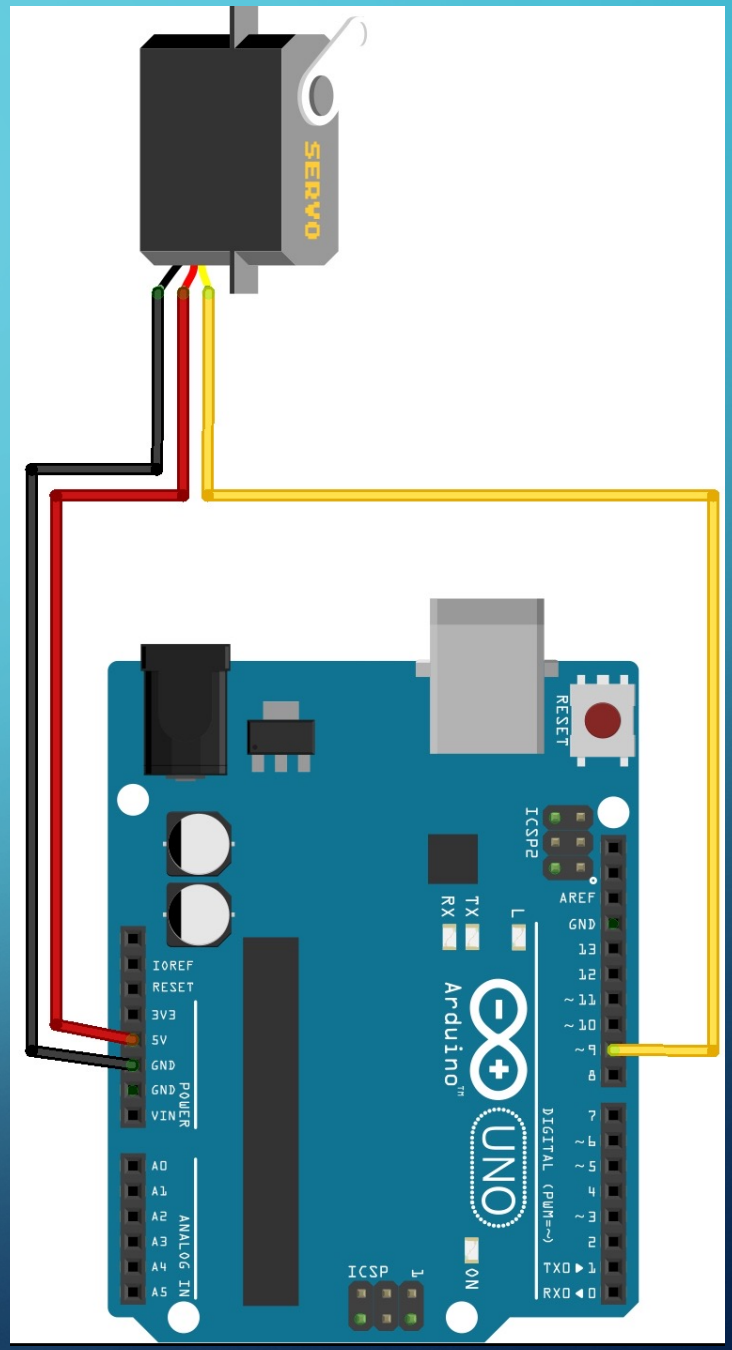
Coppia 2kgm solleva 2kg con un asta di 1m o 4kg con un asta di 50cm.

COLLEGHIAMOLO

il servomotre ha già tutto e quindi basta fare una semplice connessione.

- Alimentazione 5V
- GND
- Pin di comando di tipo PWM

  (9 o 10 o 11)



PROF. NACLERIO PASQUALE

LIBRERIE DI ARDUINO – ARDUINO SERVO

I costruttori di Arduino ci hanno già donato tutto il necessario per far funzionare in modo semplice i nostri motori.

La libreria : #include<Servo.h>

Ora dobbiamo adattare la libreria al nostro motore.

SPIEGHIAMO AD ARDUINO IL MOTORE

Dobbiamo spiegare il tipo di motore ad arduino:

Dobbiamo creare quello che in programmazione si chiama classe

Qui le cose sono molto semplici, basta solo dargli un nome.

```
1 #include <Servo.h>
2
3 Servo nome_servo;
```

APPARECCHIAMO LA TAVOLA

Dobbiamo spiegare ad arduino a che pin è collegato il servo.

Qui si che lo facciamo in void.

Collegato al pin 9

```
5 void setup() {
6
7   nome_servo.attach(9);
8
9 }
```

COMANDIAMO IL SERVO

I servomotori sono perfetti se abbiamo bisogno di impostare con facilità gli angoli a cui si devono posizionare.

Posizionati a -180°

```
11 void loop() {
12
13     nome_servo.write(-180);
14
15 }
```

SERVO CHE VA AVANTI E INDIETRO

```
#include <Servo.h>

Servo rcservo;

void setup() {

  rcservo.attach(9);

}

void loop() {

    rcservo.write(-180);
    delay(200);

    rcservo.write(-90);
    delay(200);

    rcservo.write(0);
    delay(200);

    rcservo.write(90);
    delay(200);

    rcservo.write(180);
    delay(200);

}
```

PROF. NACLERIO PASQUALE

SCHERMO LCD

AGGIUNGIAMO UN DISPLAY LCD AD ARDUINO

PROF. NACLERIO PASQUALE

CRISTALLI LIQUIDI

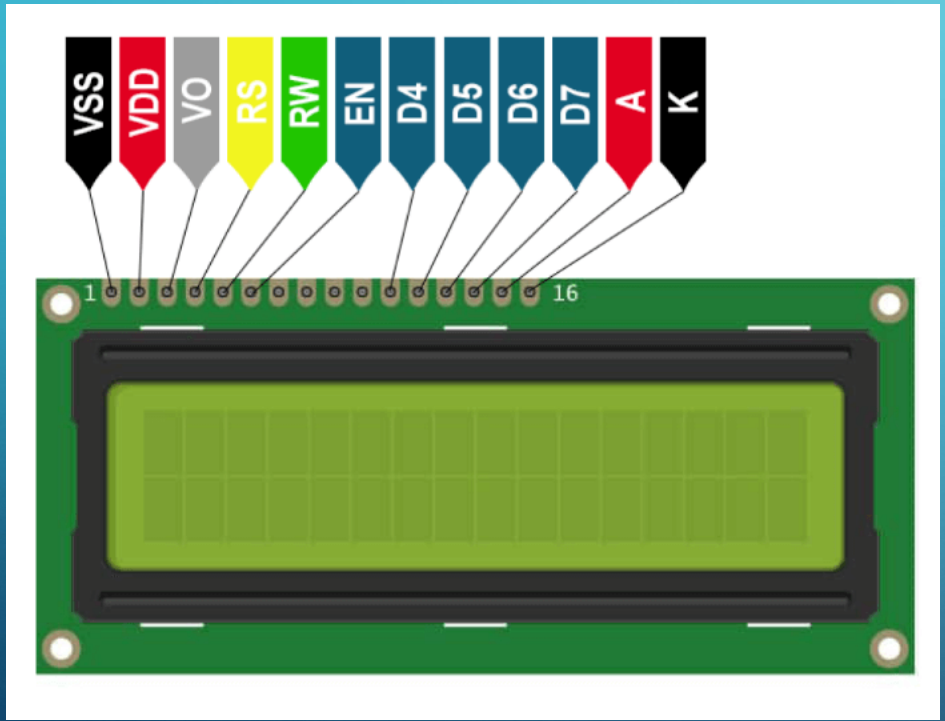
Sono schermi che costano poco e ci permettono di far apparire moltissimi tipi di caratteri.

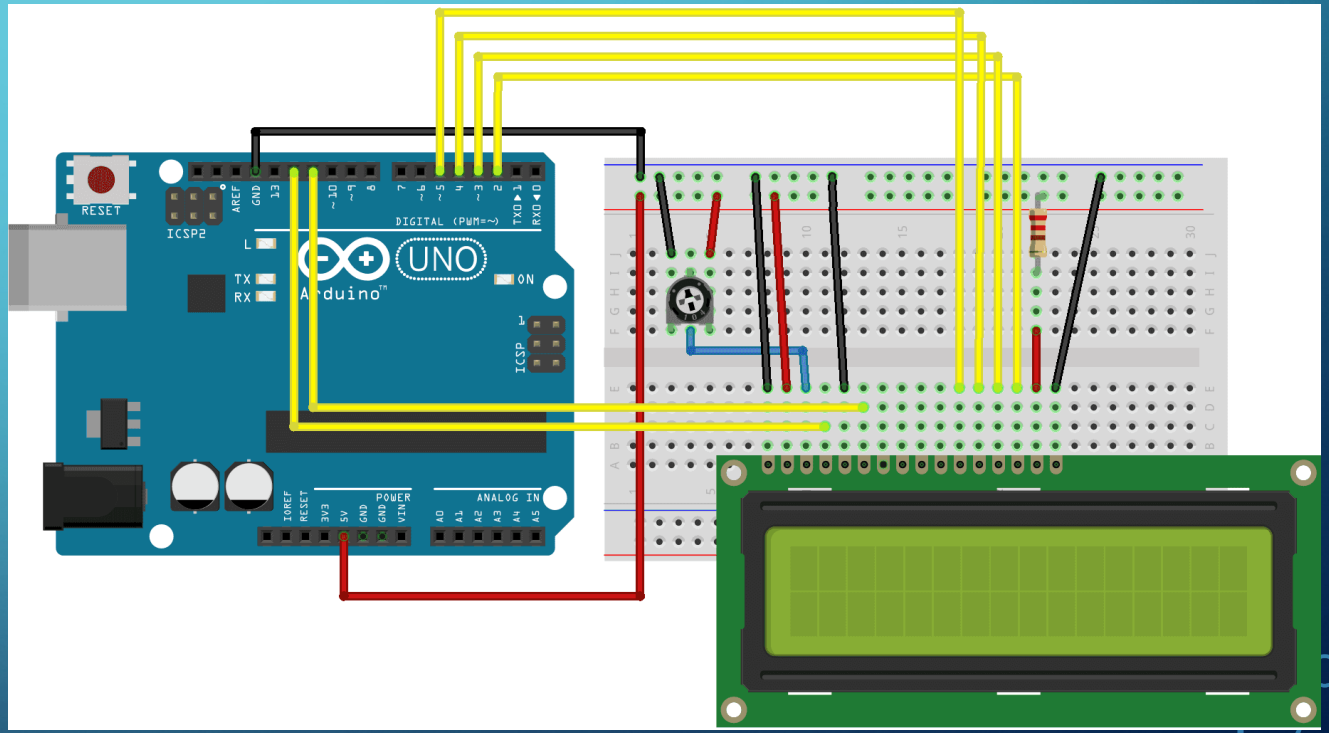
Il limite è che hanno solo 2 righe e 16 caratteri per riga.

Lo schermo più usato è quello della Hitachi 44780 che si è affermato talmente tanto da essere diventato uno standard

IL COLLEGAMENTO DIFFICILE

Questi schermi hanno 16 pin e devono essere collegati nel modo corretto.





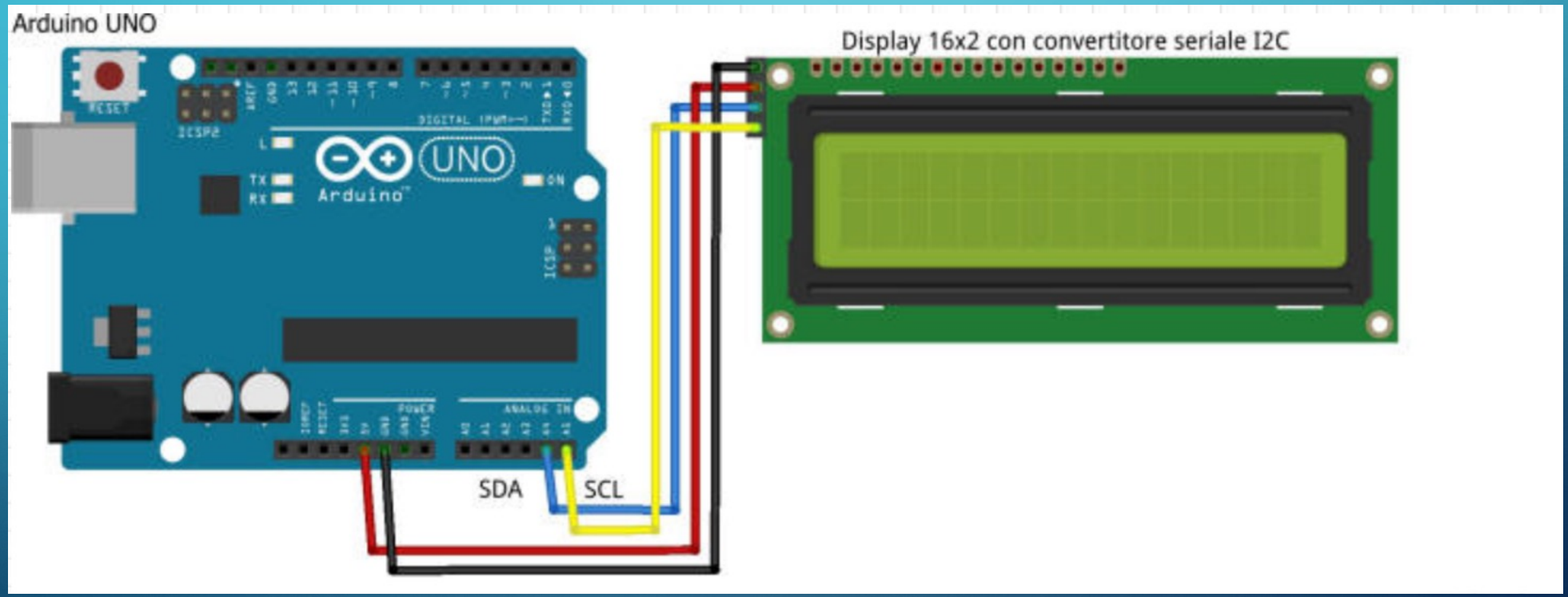
IL COLLEGAMENTO FACILE

Usando una scheda **LCM1602** si può semplificare l'operazione.

È una piccola scheda che va collegata dietro allo schermo e permette di avere solo 4 uscite.

Noi useremo questa, ma l'altro metodo è altrettanto valido

LCM1602 COLLEGAMENTO



E LE LIBRERIE ?

Arduino ci fornisce tutte le librerie per comandare un LCD nel modo classico, ma siccome io voglio usare una comunicazione diversa non trovo queste librerie….

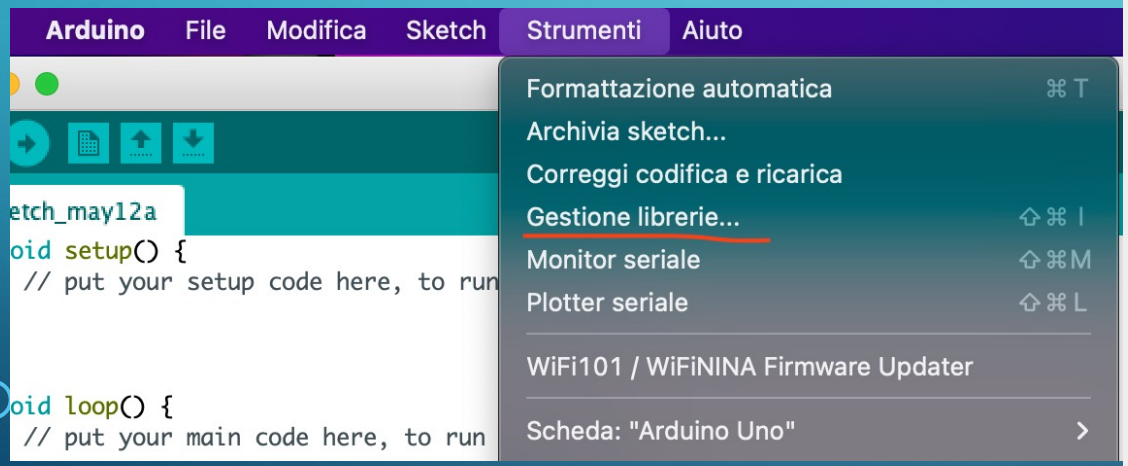
Come risolvere il problema?

1. Mi scrivo io le librerie
2. Cerco in rete se qualcuno le ha già fatte

LA COMMUNITY CI SALVA SEMPRE

Esiste già la libreria che mi serve devo solo trovarla e installarla.

Istallo LiquidCrystal_I2C



COME USARE LO SCHERMO LCD

- Aggiungiamo le librerie

```
1 #include <Wire.h>
2 #include <LiquidCrystal_I2C.h>
```

- Spieghiamo ad arduino come è fatto il nostro LCD

```
3
4 LiquidCrystal_I2C lcd(0x27,16,2);
5
```

Indirizzo di memoria 0x27, numero di colonne, numero di righe

VOID SETUP

In void setup dobbiamo dire come vogliamo lo schermo

- Dove sta ? In A4 e A5, ma è una cosa di default. lcd.init();
- Lo voglio retro iluminato lcd.backlight();
- Posso fare altre cose, ma per noi per adesso vanno bene solo queste istruzioni.

```
6  void setup()
7  {
8    lcd.init();
9    lcd.backlight();
10 }
```

VOID LOOP 1

Ora possiamo usare il nostro lcd

- Una cosa che dobbiamo sempre fare è cancellare ciò che c'era prima

lcd.clear();

```
12 void loop()
13 {
14     lcd.clear();
```

VOID LOOP 2

Alcuni strumenti:

- lcd.setCursor(posizione in riga, posizione in colonna);

lcd.setCursor(4,1); mi vado a mettere nella 4^ casella nella riga 1

- lcd.print(testo);

lcd.print("ciao"); scrivo ciao sullo schermo

ESEMPIO

```
1  #include <Wire.h>
2  #include <LiquidCrystal_I2C.h>
3
4  LiquidCrystal_I2C lcd(0x27,16,2);
5
6  void setup()
7  {
8    lcd.init();
9    lcd.backlight();
10 }
11
12 void loop()
13 {
14     lcd.clear();
15     double lettura1 = 10.0;
16     double lettura2 = 5.0;
17
18     double misura = (lettura1 + lettura2)/2;
19
20     lcd.setCursor(3,0);
21     lcd.print("misura = ");
22     lcd.print(misura);
23
24     lcd.setCursor(0, 1);
25
26     lcd.print("lettura1 = ");
27     lcd.print(lettura1);
28     delay(1000);
29
30 }
```

CIO CHE VEDRO'

_ _ _ misura = 7.50

lettura1 = 10.00

Vado a capo e inizio da 0