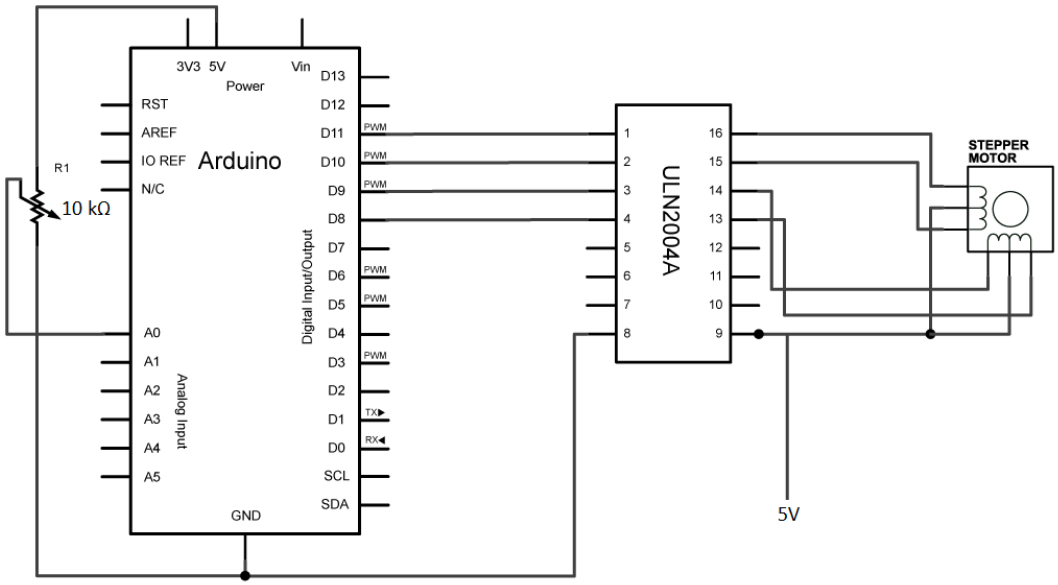
17. Stepper motor

ZADATAK 1. Spoji komponente prema shemi na slici:

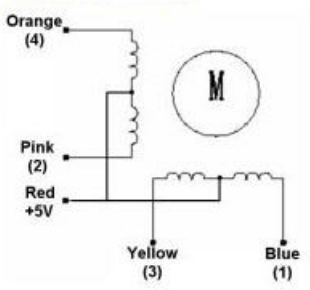


Potrebno je napisati program koji upravlja brzinom vrtnje stepper motora preko potenciometra. Vrijednost potenciometra čitaš sa analognog pina A0 (funkcija *analogRead()*) i na temelju te vrijednosti određuješ brzinu vrtnje.

Na početku programa napiši *#include <Stepper.h>*. Time ukazujemo da želimo koristiti funkcije iz „Stepper“ biblioteke (library-a).
Nakon toga, radi preglednosti, definiraj makro konstantu STEPS: *#define STEPS* 4096 (kasnije isprobaj 64 ili vrijednosti po volji pa prouči što se događa).
Idući korak jest kreirati instancu klase (objekt) *Stepper*, tj. napisati ovo:
Stepper stepper(STEPS, 8, 9, 10, 11);
Primijeti da smo prilikom kreiranja objekta kao argumente prenijeli broj koraka po jednom okretaju, i izlazne pinove Arduina.
Sada nad objektom *stepper* možemo koristiti bilo koju metodu koja nam je dostupna unutar klase *Stepper*.
Primjer korištenja metode *setSpeed()* za postavljanje brzine: *stepper.setSpeed(30);*
Tom metodom ne pokrećemo motor, već samo kažemo kojom brzinom će se okrenuti kad pozovemo metodu *step()*.
Metoda *setp()* okrene motor za broj koraka koji specificiramo u argumentu, na brzini koju smo specificirali zadnjim pozivom metode setSpeed().
Primjer korištenja: *stepper.step(20);*
Te dvije metode bit će nam dovoljne za riješiti ovaj zadatak.

Koristi funkciju *map()*, da bi prilagodio ulaz na praktičan interval brzina, npr.: *map(senzor, 0, 1023, 0, 100)*.

ZADATAK 2. Korištenje gotovih biblioteka je (u pravilu) jednostavno i elegantno, ali to nam često ograničava fleksibilnost i kontrolu našeg programa, a ponekad je i vrlo neefikasno (u smislu tzv *processing time*-a). Zato ćemo sada od nule napisati program koji upravlja stepper motorom.
Od pomoći će nam biti ovaj dijagram i tablica.



Lead Wire Color	---> CW Direction (1-2 Phase)							
	1	2	3	4	5	6	7	8
4 Orange	-	-						-
3 Yellow		-	-	-				
2 Pink				-	-	-		
1 Blue						-	-	-

Prema navedenoj tablici potrebno je postavljati pinove na Arduinu **HIGH** ili **LOW**, tj. **1** ili **0**, u točno prikazanom rasporedu. Crtice označavaju kada pin mora biti postavljen na visoku razinu, tj HIGH (1), a praznina kada je na niskoj razini, LOW (0). Sam odredi je li ti praktičnije za to koristiti funkciju *digitalWrite()* ili direktno pisanje na portove korištenjem DDRx i PORTx registara (*port manipulation*).
Preporuka je koristiti varijablu *int steps* koja pamti na kojem je koraku (ili možemo reći, stanju, tj položaju rotora) stepper (*steps* ide od 0 do 7, pogledaj tablicu) i zasebnu funkciju, *void stepper()*, koja ovisno u kojem je stanju stepper (na temelju varijable *steps*) namješta izlaze Arduina).
Npr. ako je *steps* na 3 (to je 4. korak) izlazi su 0110 (LOW, HIGH, HIGH, LOW).
Onda u *loop()* bloku pozivaš funkciju *stepper()* u pravilnim intervalima, ovisno o brzini (isprobaj to korištenjem funkcije *delay()*, *micros()*… koje je bolje rješenje?).
Kako bi obrnuo smjer vrtnje?
Prilagodi program tako da preko Serial Monitora dobiva brzinu i smjer vrtnje, broj koraka koje treba napraviti…

Za brze tu je…

ZADATAK 3. Treba napisati program koji mjeri brzinu okretaja, tj. brzinomjer (pogledaj na ploči).

HINT pomoću funkcije *milis()* mjeri period signala sa fotootpornika (ulaz A0). Reći ćemo da je period vrijeme između trenutaka kada vrijednost na ulazu prijeđe neku graničnu vrijednost (vjerojatno MAX_VRIJ/2, eksperimentiraj sa različitim vrijednostima, kako to utječe na točnost mjerenja).

Rated voltage : 5VDC
Number of Phase 4
Speed Variation Ratio 1/64
Stride Angle 5.625° /64
Frequency 100Hz
DC resistance 50Ω±7%(25℃)
Idle In-traction Frequency > 600Hz
Idle Out-traction Frequency > 1000Hz
In-traction Torque >34.3mN.m(120Hz)
Self-positioning Torque >34.3mN.m
Friction torque 600-1200 gf.cm
Pull in torque 300 gf.cm

(1 N·m = 10197.1621297793 gf·cm)