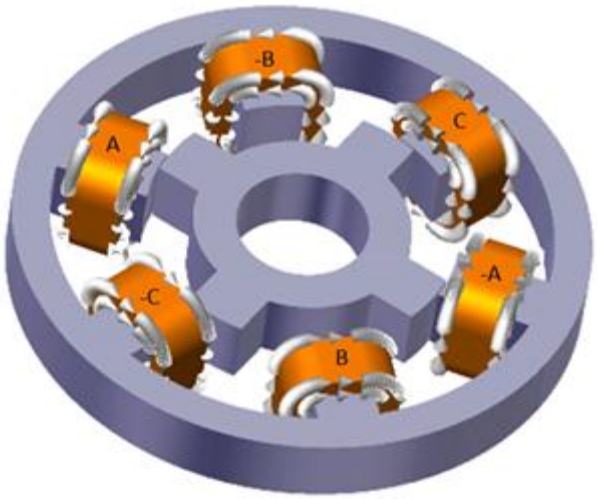
Stepper Motor

A stepper motor is an electromechanical device it converts electrical power into mechanical power. Also it is a brushless, synchronous electric motor that can divide a full rotation into an expansive number of steps. The motor's position can be controlled accurately without any feedback mechanism, as long as the motor is carefully sized to the application. Stepper motors are similar to switched reluctance motors.

The stepper motor uses the theory of operation for magnets to make the motor shaft turn a precise distance when a pulse of electricity is provided. The stator has eight poles, and the rotor has six poles. The rotor will require 24 pulses of electricity to move the 24 steps to make one complete revolution. Another way to say this is that the rotor will move precisely 15° for each pulse of electricity that the motor receives.



Types of Stepper Motor:

There are three main types of stepper motors, they are:

Permanent Magnet Stepper Motor: Permanent magnet motors use a permanent magnet (PM) in the rotor and operate on the attraction or repulsion between the rotor PM and the stator electromagnets.

Variable Reluctance Stepper Motor: Variable reluctance (VR) motors have a plain iron rotor and operate based on the principle that minimum reluctance occurs with minimum gap, hence the rotor points are attracted toward the stator magnet poles.

Hybrid Synchronous Stepper Motor: Hybrid stepper motors are named because they use a combination of permanent magnet (PM) and variable reluctance (VR) techniques to achieve maximum power in a small package size.

Operation of Stepper Motor:

Stepper motors operate differently from DC brush motors, which rotate when voltage is applied to their terminals. Stepper motors, on the other hand, effectively have multiple toothed electromagnets arranged around a central gear-shaped piece of iron. The electromagnets are energized by an external control circuit, for example a microcontroller.

To make the motor shaft turn, first one electromagnet is given power, which makes the gear's teeth magnetically attracted to the electromagnet's teeth. The point when the gear's teeth are thus aligned to the first electromagnet, they are slightly offset from the next electromagnet. So when the next electromagnet is turned ON and the first is turned OFF, the gear rotates slightly to align with the next one and from there the process is repeated. Each of those slight rotations is called a step, with an integer number of steps making a full rotation. In that way, the motor can be turned by a precise. Stepper motor doesn't rotate continuously, they rotate in steps. There are 4 coils with 90^{o} angle between each other fixed on the stator. The stepper motor connections are determined by the way the coils are interconnected. In stepper motor, the coils are not connected together. The motor has 90^{o} rotation step with the coils being energized in a cyclic order, determining the shaft rotation direction. The working of this motor is shown by operating the switch. The coils are activated in series in 1 sec intervals. The shaft rotates 90^{o} each time the next coil is activated. Its low speed torque will vary directly with current.

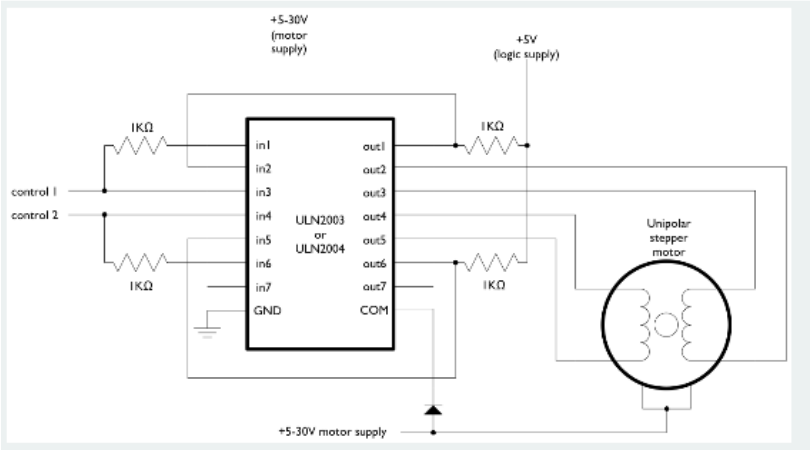
Using A Stepper motor with an Arduino based microcontroller:

Stepper.h Library

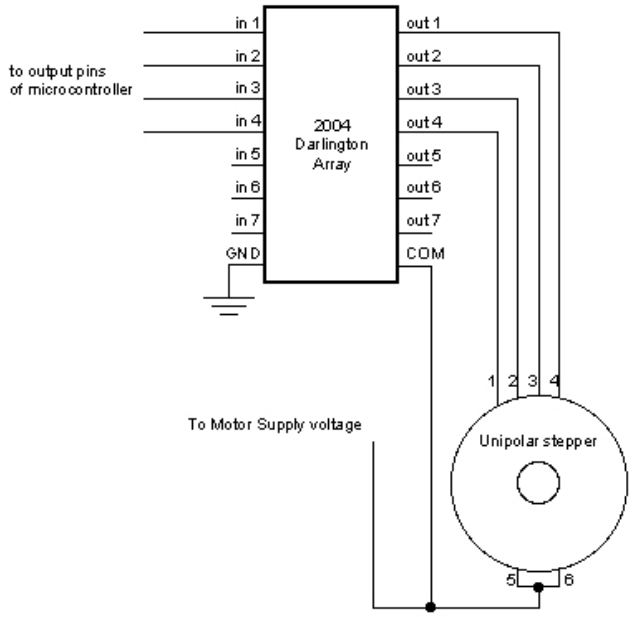
This library allows you to control unipolar or bipolar stepper motors. To use it you will need a stepper motor, and the appropriate hardware to control it.

For unipolar stepper consider this schematic:

2 pins:



4 pins:



Example 1: Controlling a highly accurate stepper motor using a potentiometer.

Stepper motors, due to their unique design, can be controlled to a high degree of accuracy without any feedback mechanisms. The shaft of a stepper, mounted with a series of magnets, is controlled by a series of electromagnetic coils that are charged positively and negatively in a specific sequence, precisely moving it forward or backward in small "steps".

In this example, a potentiometer (or other sensor) on analog input 0 is used to control the movement of a stepper motor using the Arduino Stepper Library. The stepper is controlled by with digital pins 8, 9, 10, and 11 for either unipolar or bipolar motors.

The Arduino or Genuino board will connect to a U2004 Darlington Array if you're using a unipolar stepper or a SN754410NE H-Bridge if you have a bipolar motor.

ULN2001, ULN2002 ULN2003, ULN2004

Seven Darlington array

Datasheet – production data

Features

- Seven Darlingtons per package
- Output current 500 mA per driver (600 mA peak)
- Output voltage 50 V
- Integrated suppression diodes for inductive loads
- Outputs can be paralleled for higher current
- TTL/CMOS/PMOS/DTL compatible inputs
- Inputs pinned opposite outputs to simplify layout

DIP-16

SO-16 (Narrow)

Description

The ULN2001, ULN2002, ULN2003 and ULN 2004 are high voltage, high current Darlington arrays each containing seven open collector Darlington pairs with common emitters. Each channel rated at 500 mA and can withstand peak currents of 600 mA. Suppression diodes are included for inductive load driving and the inputs are pinned opposite the outputs to simplify board layout.

The versions interface to all common logic families:

- ULN2001 (general purpose, DTL, TTL, PMOS, CMOS)
- ULN2002 (14 - 25 V PMOS)
- ULN2003 (5 V TTL, CMOS)
- ULN2004 (6 - 15 V CMOS, PMOS)

These versatile devices are useful for driving a wide range of loads including solenoids, relays DC motors, LED displays filament lamps, thermal printheads and high power buffers.

The ULN2001A/2002A/2003A and 2004A are supplied in 16 pin plastic DIP packages with a copper leadframe to reduce thermal resistance. They are available also in small outline package (SO-16) as ULN2001D1/2002D1/2003D1/ 2004D1

Table 1. Device summary

Order codes	
ULN2001A	ULN2001D1013TR
ULN2002A	ULN2002D1013TR
ULN2003A	ULN2003D1013TR
ULN2004A	ULN2004D1013TR

1 Diagram

Figure 1. Schematic diagram

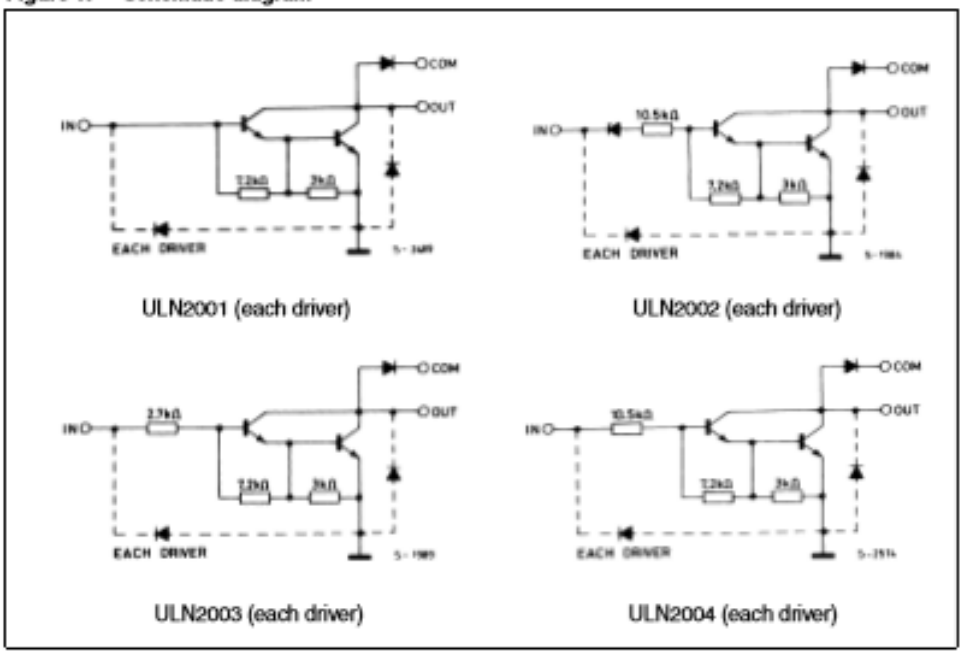


Table 2. Absolute maximum ratings

Symbol	Parameter	Value	Unit
V_O	Output voltage	50	V
V_I	Input voltage (for ULN2002A/D - 2003A/D - 2004A/D)	30	V
I_C	Continuous collector current	500	mA
I_B	Continuous base current	25	mA
T_A	Operating ambient temperature range	- 40 to 85	°C
T_{STG}	Storage temperature range	- 55 to 150	°C
T_J	Junction temperature	150	°C

Table 3. Thermal data

Symbol	Parameter	DIP-16	SO-16	Unit
R_{thJA}	Thermal resistance junction-ambient, Max.	70	120	°C/W

T_A = 25 °C unless otherwise specified.

Table 4. Electrical characteristics

Symbol	Parameter	Test condition	Min.	Typ.	Max.	Unit
I_{CEX}	Output leakage current	V_{CE} = 50 V, (*Figure 3.*)			50	µA
		T_A = 85°C, V_{CE} = 50 V (*Figure 3.*)			100	
		T_A = 85°C for ULN2002, V_{CE} = 50 V, V_I = 6 V (*Figure 4.*)			500	
		T_A = 85°C for ULN2002, V_{CE} = 50 V, V_I = 1V (*Figure 4.*)			500	
$V_{CE(SAT)}$	Collector-emitter saturation voltage (*Figure 5.*)	I_C = 100 mA, I_B = 250 µA		0.9	1.1	V
		I_C = 200 mA, I_B= 350 µA		1.1	1.3	
		I_C = 350 mA, I_B= 500 µA		1.3	1.6	
$I_{I(ON)}$	Input current (*Figure 6.*)	for ULN2002, V_I = 17 V		0.82	1.25	mA
		for ULN2003, V_I = 3.85 V		0.93	1.35	
		for ULN2004, V_I = 5 V		0.35	0.5	
		V_I = 12 V		1	1.45	
$I_{I(OFF)}$	Input current (*Figure 7.*)	T_A = 85°C, I_C = 500 µA	50	65		µA
$V_{I(ON)}$	Input voltage (*Figure 8.*)	V_{CE}= 2 V, for ULN2002 I_C = 300 mA for ULN2003 I_C = 200 mA I_C = 250 mA I_C = 300 mA for ULN2004 I_C = 125 mA I_C = 200 mA I_C = 275 mA I_C = 350 mA			13 2.4 2.7 3 5 6 7 8	V
h_{FE}	DC Forward current gain (*Figure 5.*)	for ULN2001, V_{CE} = 2 V, I_C = 350 mA	1000			
C_I	Input capacitance			15	25	pF
t_{PLH}	Turn-on delay time	0.5 V_I to 0.5 V_O		0.25	1	µs
t_{PHL}	Turn-off delay time	0.5 V_I to 0.5 V_O		0.25	1	µs
I_R	Clamp diode leakage current (*Figure 9.*)	V_R = 50 V			50	µA
		T_A = 85°C, V_R = 50 V			100	
V_F	Clamp diode forward voltage (*Figure 10.*)	I_F = 350 mA		1.7	2	V

SN754410 Quadruple Half-H Driver

1 Features

- 1-A Output-Current Capability Per Driver
- Applications Include Half-H and Full-H Solenoid Drivers and Motor Drivers
- Designed for Positive-Supply Applications
- Wide Supply-Voltage Range of 4.5 V to 36 V
- TTL- and CMOS-Compatible High-Impedance Diode-Clamped Inputs
- Separate Input-Logic Supply
- Thermal Shutdown
- Internal ESD Protection
- Input Hysteresis Improves Noise Immunity
- 3-State Outputs
- Minimized Power Dissipation
- Sink/Source Interlock Circuitry Prevents Simultaneous Conduction
- No Output Glitch During Power Up or Power Down
- Improved Functional Replacement for the SGS L293

2 Applications

- Stepper Motor Drivers
- DC Motor Drivers
- Latching Relay Drivers

3 Description

The SN754410 is a quadruple high-current half-H driver designed to provide bidirectional drive currents up to 1 A at voltages from 4.5 V to 36 V. The device is designed to drive inductive loads such as relays, solenoids, DC and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are compatible with TTL-and low-level CMOS logic. Each output (Y) is a complete totem-pole driver with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled and their outputs become active and in phase with their inputs. When the enable input is low, those drivers are disabled and their outputs are off and in a high-impedance state. With the proper data inputs, each pair of drivers form a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

A separate supply voltage (V_{CC1}) is provided for the logic input circuits to minimize device power dissipation. Supply voltage V_{CC2} is used for the output circuits.

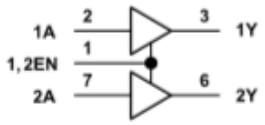
The SN754410 is designed for operation from −40°C to 85°C.

Device Information[1]

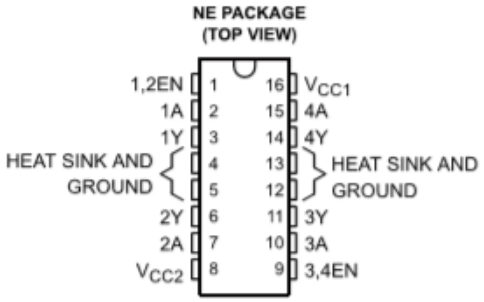
PART NUMBER	PACKAGE (PIN)	BODY SIZE (NOM)
SN754410	PDIP (16)	19.80 mm × 6.35 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

4 Simplified Schematic



6 Pin Configuration and Functions



Pin Functions

PIN		TYPE	DESCRIPTION
NAME	NO.		
1,2EN	1	I	Enable driver channels 1 and 2 (active high input)
<1:4>A	2, 7, 10, 15	I	Driver inputs, non-inverting
<1:4>Y	3, 6, 11, 14	O	Driver outputs
GROUND	4, 5, 12, 13	—	Device ground and heat sink pin. Connect to circuit board ground plane with multiple solid vias
V_{CC2}	8	—	Power VCC for drivers 4.5V to 36V
3,4EN	9	I	Enable driver channels 3 and 4 (active high input)
V_{CC1}	16	—	5V supply for internal logic translation

7.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)[1][2]

		MIN	MAX	UNIT
V_{CC1}	Output supply voltage range	–0.5	36	V
V_{CC2}	Output supply voltage range	–0.5	36	V
V_I	Input voltage	–0.5	36	V
V_O	Output voltage range	–3	V_{CC2} + 3	V
I_P	Peak output current		±2	A
I_O	Continuous output current		±1	A
P_D	Continuous total power dissipation at (or below) 25°C free-air temperature[3]		2075	mW
T_A	Operating free-air temperature range	–40	85	°C
T_J	Operating virtual junction temperature range	–40	150	°C
T_{stg}	Storage temperature range		260	°C

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) All voltage values are with respect to network GND.
(3) For operation above 25°C free-air temperature, derate linearly at the rate of 16.6 mW/°C. To avoid exceeding the design maximum virtual junction temperature, these ratings should not be exceeded. Due to variations in individual device electrical characteristics and thermal resistance, the built-in thermal overload protection can be activated at power levels slightly above or below the rated dissipation.

7.2 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

		MIN	MAX	UNIT
V_{CC1}	Logic supply voltage	4.5	5.5	V
V_{CC2}	Output supply voltage	4.5	36	V
V_{IH}	High-level input voltage	2	5.5	V
V_{IL}	Low-level input voltage	–0.3[1]	0.8	V
T_J	Operating virtual junction temperature	–40	125	°C
T_A	Operating free-air temperature	–40	85	°C

(1) The algebraic convention, in which the least positive (most negative) limit is designated as minimum, is used in this data sheet for logic voltage levels.

Hardware Required

- Arduino or Genuino Board
- 10k ohm potentiometer
- stepper motor
- U2004 Darlington Array (if using a unipolar stepper)
- SN754410ne H-Bridge (if using a bipolar stepper)
- power supply appropriate for your particular stepper
- hook-up wires
- breadboard

Code

For both unipolar and bipolar steppers

```
/*
 * MotorKnob
 *
 * A stepper motor follows the turns of a potentiometer
 * (or other sensor) on analog input 0.
 *
 * http://www.arduino.cc/en/Reference/Stepper
 * This example code is in the public domain.
 */

#include <Stepper.h>

// change this to the number of steps on your motor
#define STEPS 100

// create an instance of the stepper class, specifying
// the number of steps of the motor and the pins it's
// attached to
Stepper stepper(STEPS, 8, 9, 10, 11);

// the previous reading from the analog input
int previous = 0;

void setup() {
  // set the speed of the motor to 30 RPMs
  stepper.setSpeed(30);
}

void loop() {
  // get the sensor value
  int val = analogRead(0);
```

```
  // move a number of steps equal to the change in the
  // sensor reading
  stepper.step(val - previous);

  // remember the previous value of the sensor
  previous = val;
}
```

Example 2: Turn the shaft step by step to check the proper wiring

Code

For both unipolar and bipolar steppers

```
/*
 Stepper Motor Control - one step at a time

 This program drives a unipolar or bipolar stepper motor.
 The motor is attached to digital pins 8 - 11 of the Arduino.

 The motor will step one step at a time, very slowly.  You can use this to
 test that you've got the four wires of your stepper wired to the correct
 pins. If wired correctly, all steps should be in the same direction.

 Use this also to count the number of steps per revolution of your motor,
 if you don't know it.  Then plug that number into the oneRevolution
 example to see if you got it right.

 Created 30 Nov. 2009
 by Tom Igoe

 */

#include <Stepper.h>

const int stepsPerRevolution = 200;  // change this to fit the number of steps per revolution
// for your motor

// initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);

int stepCount = 0;         // number of steps the motor has taken

void setup() {
  // initialize the serial port:
  Serial.begin(9600);
}

void loop() {
  // step one step:
  myStepper.step(1);
```

```
  Serial.print("steps:");
  Serial.println(stepCount);
  stepCount++;
  delay(500);
}
```

Example 3: Altering the speed of the stepper

Code

For both unipolar and bipolar steppers

```
/*
 Stepper Motor Control - speed control

 This program drives a unipolar or bipolar stepper motor.
 The motor is attached to digital pins 8 - 11 of the Arduino.
 A potentiometer is connected to analog input 0.

 The motor will rotate in a clockwise direction. The higher the potentiometer value,
 the faster the motor speed. Because setSpeed() sets the delay between steps,
 you may notice the motor is less responsive to changes in the sensor value at
 low speeds.

 Created 30 Nov. 2009
 Modified 28 Oct 2010
 by Tom Igoe

 */

#include <Stepper.h>

const int stepsPerRevolution = 200;  // change this to fit the number of steps per revolution
// for your motor


// initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);

int stepCount = 0;  // number of steps the motor has taken

void setup() {
  // nothing to do inside the setup
}

void loop() {
  // read the sensor value:
  int sensorReading = analogRead(A0);
  // map it to a range from 0 to 100:
  int motorSpeed = map(sensorReading, 0, 1023, 0, 100);
  // set the motor speed:
  if (motorSpeed > 0) {
```

```
    myStepper.setSpeed(motorSpeed);
    // step 1/100 of a revolution:
    myStepper.step(stepsPerRevolution / 100);
  }
}
```