

EP1000

Actuators

Actuators

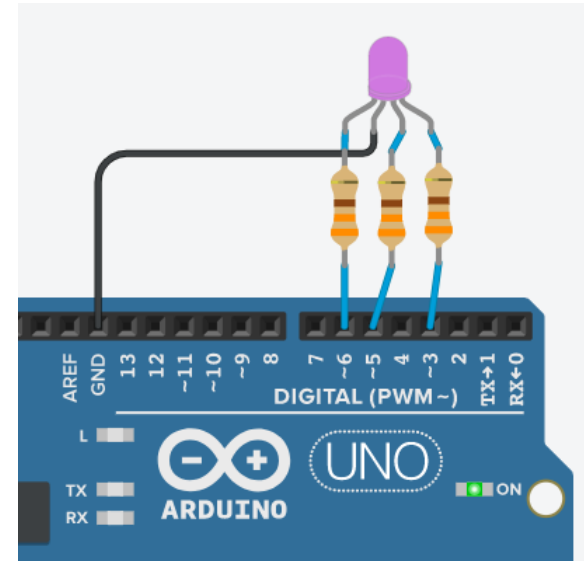
- A component of a machine that moves/controls a mechanism in a system.
- An “output” device.
- Examples:
 - Displays – LED, LCD, Neopixels
 - Motors – DC Motors, Stepper Motors, Servo Motors
 - Valves – Water, gas
 - Solenoids – controls heavier voltages, currents
 - Sounds – buzzers, alarms

Display devices

- The most common output display devices used in projects are:
 - LEDs (indicators)
 - LCD (display panels, information) 16x2, 20x2
 - 7-segment tube displays
 - Dot-matrix displays (running messages)
 - Oled displays
 - NeoPixels (strip lights, multi RGB leds)

LEDs

- Types: single, RGB, 7-segment
- Identify the Anode (or Cathode)
- Ensure there is proper current limiting resistors
- You need a delay to maintain the display (persistence of vision)
- You can use PWM to obtain different intensity levels.



```

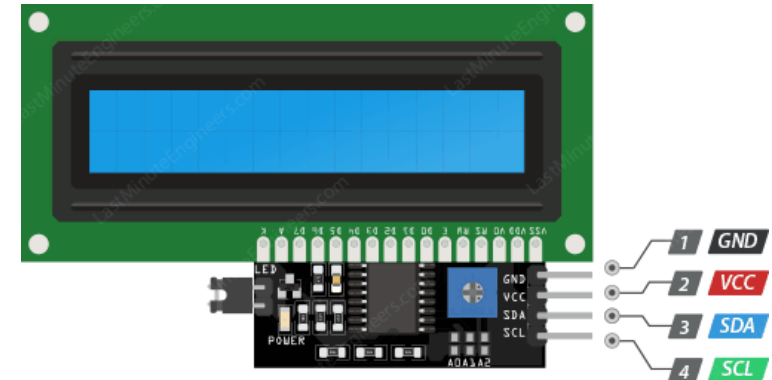
3  const int RED = 6;
4  const int BLUE = 5;
5  const int GREEN = 3;
6
7  void setup()
8  {
9  }
10
11 void loop()
12 {
13     analogWrite(RED, random(256));
14     analogWrite(GREEN, random(256));
15     analogWrite(BLUE, random(256));
16     delay(500);
17 }
18

```

[Uno Random RGB Led using PWM](#)

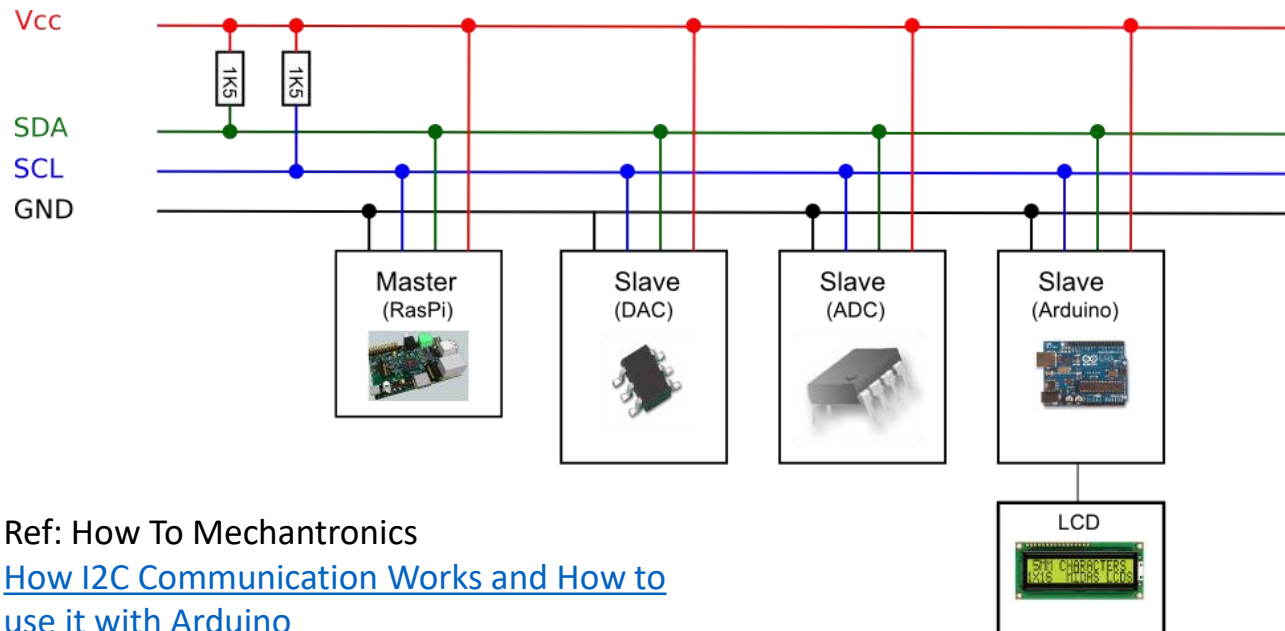
LCD Displays

- Common method of displaying information
- Usually Hitachi HD44780 control requires 11 lines (8 data, RS, RW, En)
- Made simpler using an I2C 1602 controller for the LCD panel
- Library: [LiquidCrystal I2C](#) by Frank de Brabander.
- Reference: [LastMinuteEngineers](#)



I2C Interface

- A method of Serial communication.
- Uses 2 wires (SDA, SCL) to communicate between devices.
- Each device has a unique address which identifies it.
- Can have up to 1024 devices



Ref: How To Mechantronics

[How I2C Communication Works and How to use it with Arduino](#)

Identifying I2c devices

- Use the controller (master) to scan the bus for devices (slaves).
- The addresses will be printed out on the Serial Monitor.
- You can then use the address to write/read from the device.
- Library: Arduino [Wire Library](#)
- Example code is usually provided by the device library.

```
4 // I2C Library
5 #include <Wire.h>
6
7 void setup() {
8     Serial.begin (9600);
9     Serial.println ("I2C scanner. Scanning ...");
10    // initialise I2C
11    Wire.begin();
12    for (byte i = 8; i < 120; i++)
13    {
14        Wire.beginTransmission (i);
15        if (Wire.endTransmission () == 0)
16        {
17            Serial.print ("Found address: ");
18            Serial.print (i, DEC);
19            Serial.print (" (0x");
20            Serial.print (i, HEX);
21            Serial.println (");");
22            delay (1); // maybe unneeded?
23        } // end of good response
24    } // end of for loop
25    Serial.println ("Done.");
26 }
27
28 void loop() {
29     // do nothing here
30 }
```

Writing to I2C LCD display

- Using the LiquidCrystal_I2C library.
- Create the object lcd
- Initial housekeeping
 - Initialise
 - Clear the screen
 - Turn on backlight
- To print messages
 - Position the cursor
 - Print the message

```
3 // LCD I2C library
4 #include <LiquidCrystal_I2C.h>
5 // set the LCD address to 0x3F, 16 chars
6 // and 2 line display
7 LiquidCrystal_I2C lcd(0x3F,16,2);
8
9 void setup() {
10     lcd.init();
11     lcd.clear();
12     lcd.backlight();      // Make sure backlight is on
13
14     // Print a message on both lines of the LCD.
15     // Set cursor to character 2 on line 0
16     lcd.setCursor(2,0);
17     lcd.print("Hello world!");
18     //Move cursor to character 2 on line 1
19     lcd.setCursor(2,1);
20     lcd.print("LCD Tutorial");
21 }
22
23 void loop() {
24     // write your update code here
25 }
26
```


Tube 7-Segment Display TM1637

- 7-segment displays common in projects
- Having multiple 7-segments require conversion and multiplexing
- TM1637 provides 4-digit 7-segment display with only 2 digital I/O lines.
- Library: [TM1637-master](#) by Avishay Orpaz.
- Tutorial:
 - [Last Minute Engineers](#)
 - [Makerguides](#)



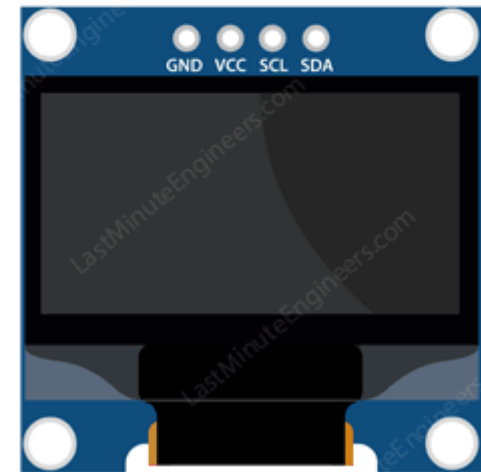
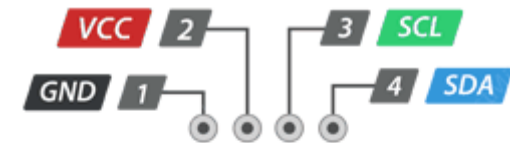
TM1637 Module Pinout



Other versions have 6~8 digits
Simplifies circuitry

Oled Displays SSD1306

- Very small, graphic displays (30x20mm)
- Getting popular, for IOT projects for information display.
- Resolution: 128x128, 128x64, 128x32 pixels
- Uses I2C interface
- Library: [SSD1306](#) by Adafruit
- Tutorials:
 - [Last Minute Engineers](#)
 - [Adafruit](#)



NeoPixels

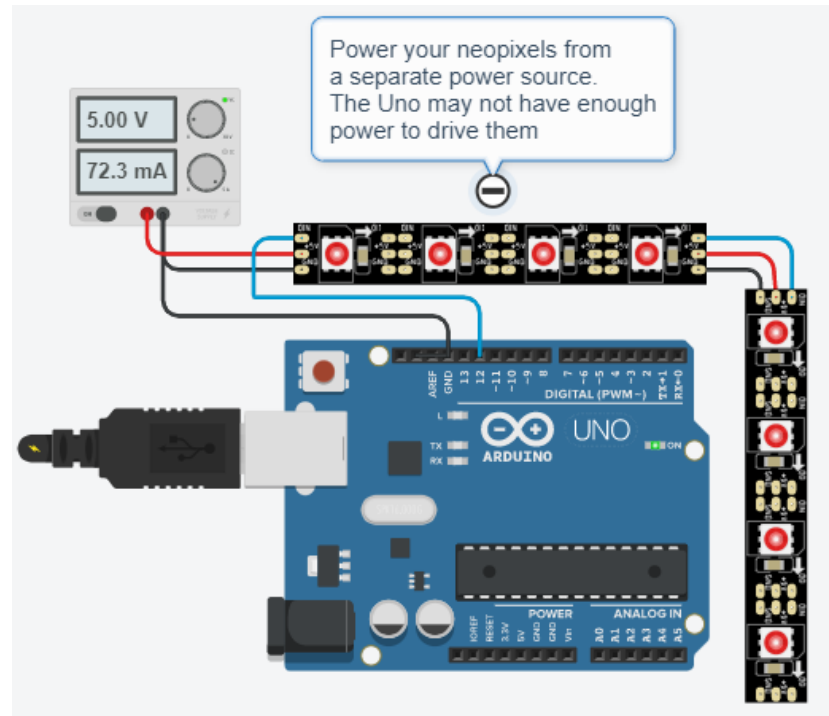
- Individually addressable LEDs housed on a strip and controlled by a single digital I/O pin from the microcontroller.
- Can control
 - Which LED to turn ON/OFF
 - Intensity of light
 - Color (If RGB)
- Good for lighting, ambient light control, effects
- Library: [Adafruit Neopixel](#)
- Tutorial:
 - [Adafruit](#)



Neopixels require quite a lot of current. Always supply the neopixels from a separate power source, not from the microcontroller.

Neopixels - demo

- Always power up neopixels using a separate power source for the extra current required.
- You can join strips of neopixels together.
- [Adafruit Library](#) comes with a number of very useful methods:
 - Color(r, g, b) returns a 32bit color
 - fill() fills neopixels with same color
 - setColor() sets individual neopixels
 - setBrightness() controls intensity
 - clear() blanks out
 - show() must be called before the neopixels can display



[Uno Neopixel Strip](#)

Neopixels – demo code

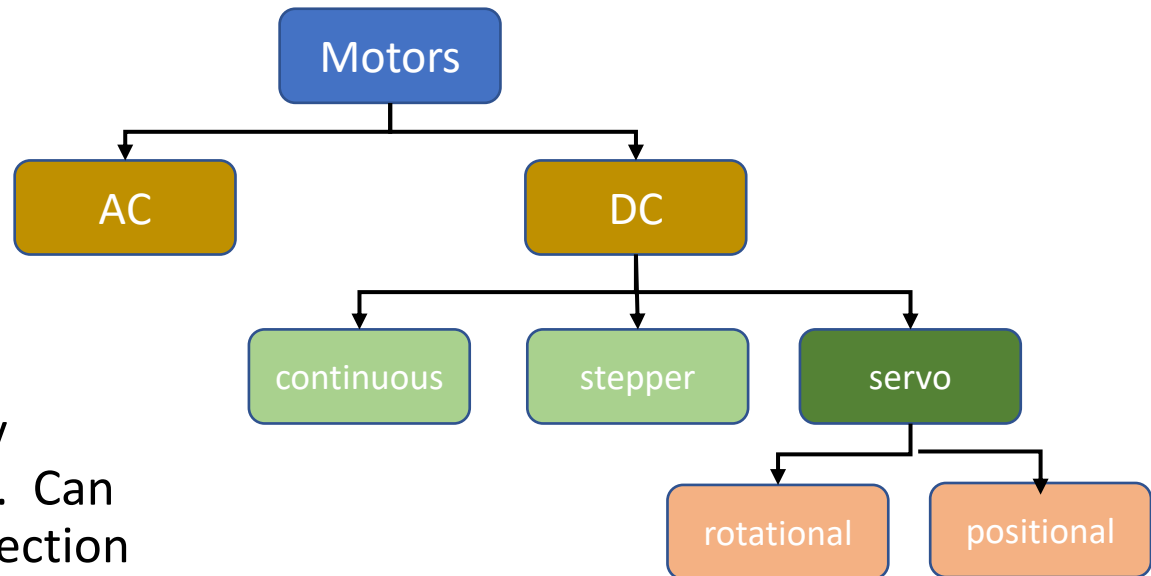
```
2 // using Adafruit Neopixel library
3 #include <Adafruit_NeoPixel.h>
4 const int PIN = 12; // digitalIO pin
5 const int NUMPIX = 8; // num of neopix
6 // create a neopixel object
7 Adafruit_NeoPixel np = Adafruit_NeoPixel(
8     NUMPIX, PIN, NEO_GRB + NEO_KHZ800);
9
10 void setup()
11 {
12     Serial.begin(9600);
13     np.setBrightness(128);
14 }
15
16 void loop()
17 {
18     // flash all, R->G->B
19     unsigned long c = np.Color(255,0,0);
20     flash(c);
21     c = np.Color(0,255,0);
22     flash(c);
23     c = np.Color(0,0,255);
24     flash(c);
25     // scroll random lights
26     for (int i=0; i<10; ++i)
27         runLights();
28 }
```

```
30 void runLights()
31 {
32     const int DLY = 100;
33     int r, g, b;
34     // obtain a random color
35     r = random(256); g = random(256); b = random(256);
36     unsigned long c = np.Color(r,g,b);
37     for (int p=0; p < NUMPIX; ++p)
38     {
39         np.setPixelColor(p, c);
40         np.show();
41         delay(DLY);
42     }
43 }
44
45 void flash(unsigned long c)
46 {
47     np.fill(c, 0, NUMPIX);
48     np.show();
49     delay(500);
50     // clear
51     np.clear();
52     np.show();
53     delay(500);
54 }
```

uint32_t = unsigned long = 32bit

uint16_t = unsigned int = 16 bit

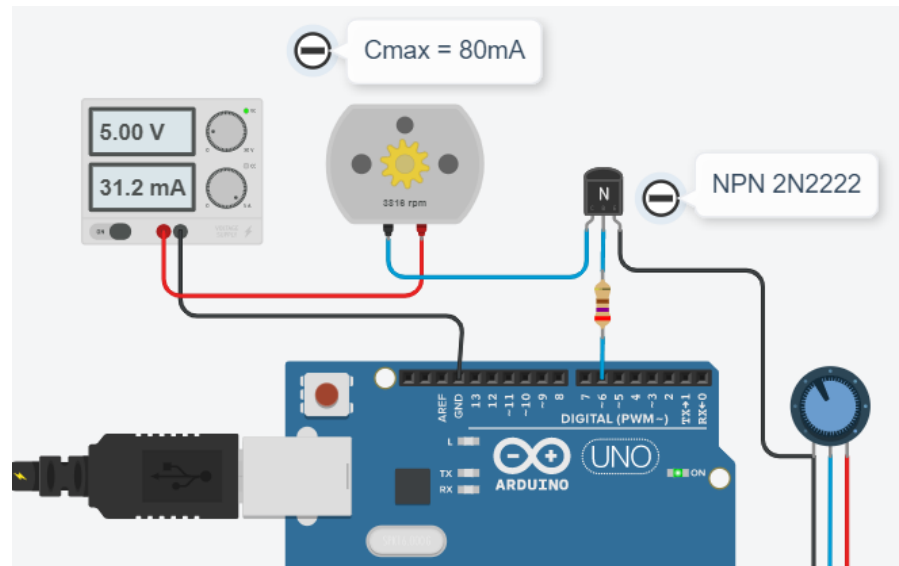
Motors



- DC Motors require a DC current to work.
- DC motors are usually continuously rotating. Can control speed and direction
- Stepper motors can move in 'steps'. Usually done using a controller module, or by changing phase activation.
- Servo motors can be controlled to move, stop/hold, change direction using pulses.
- Motors consume a lot of current, hence requires an interface circuit. Should not be driven from a digital IO pin.

DC Motor control

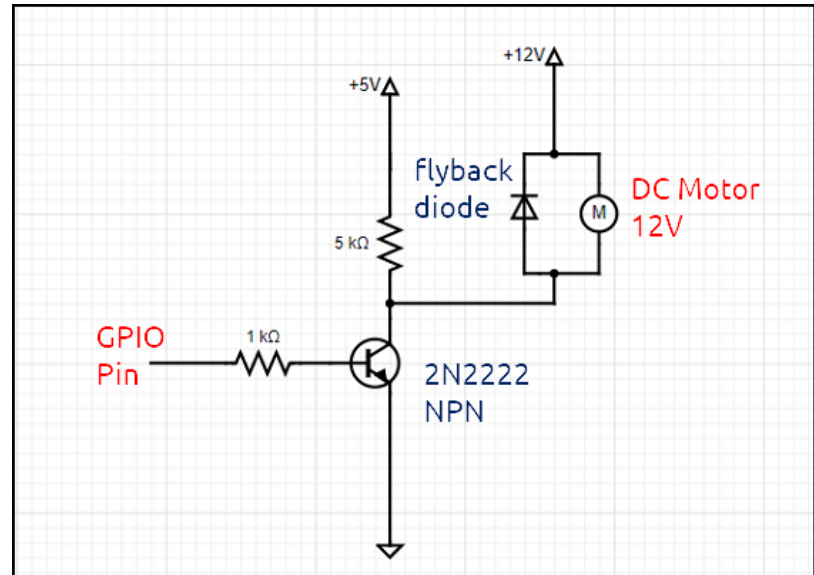
- Use a digital IO pin to control a transistor (Q). Transistor acts like a “tap”
- Qon allows current to flow, which makes motor move.
- Qoff turns off the current.
- Using PWM to control the On/OFF sequence allows control in motor speed.
- PWM at low speeds do not provide much torque.
- This method only allows one direction of motor.



[PWM Motor Control Simulation](#)

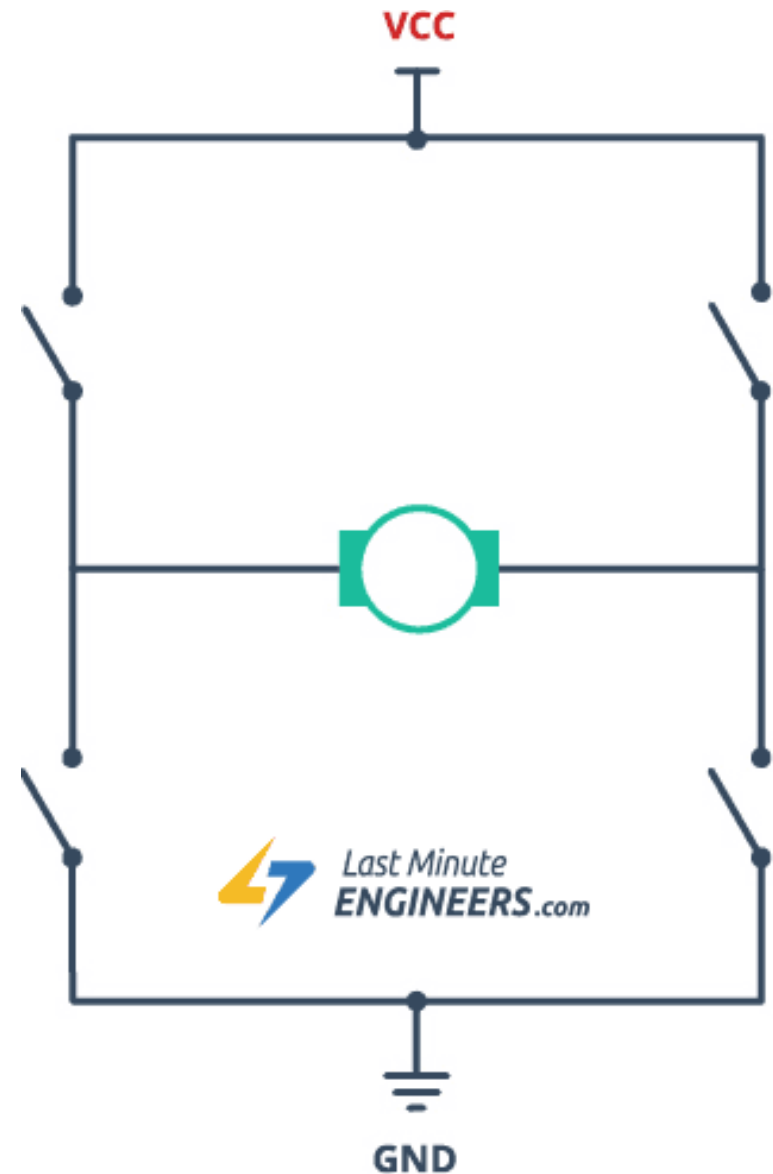
Motor Control with PWM

- Many circuits available
 - [Transistor](#)
 - [MOSFET](#)
- Transistor acts as a “control tap/valve” for current. Used in the saturation mode.
- Problems:
 - Single direction only
 - Need to worry about resistor values
 - Small duty cycles result in lower torque.



H-bridge

- Allows PWM motor control in both directions.
- H-bridge is an IC made up of transistors or MOSFETs
- Most common method of controlling DC motors
- References:
 - [Dronebot Workshop](#)
 - [LastMinuteEngineers](#)



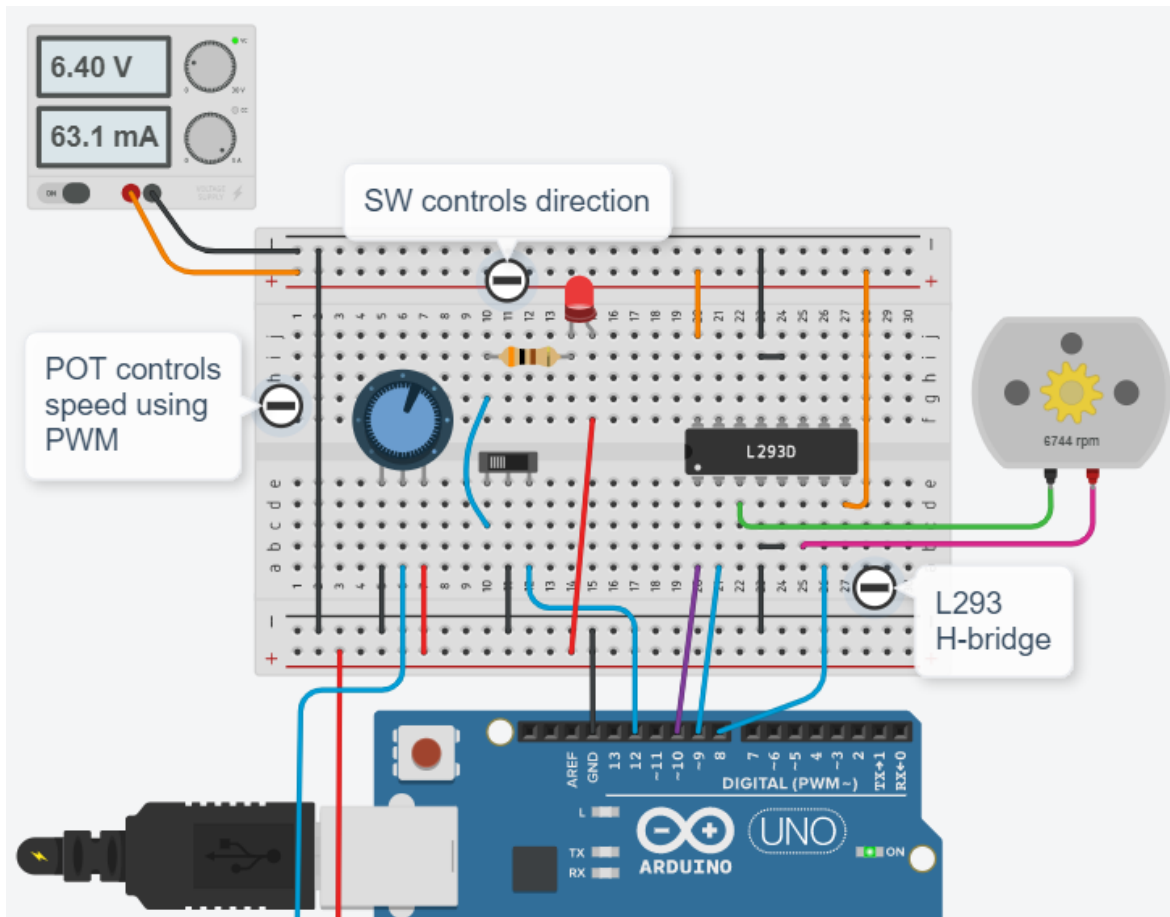
H-bridge connections

- Transistor driven, combined voltage drop is 1.4V.
- If motor is 9V, you will need minimum $9 + 1.4 = 10.4\text{V}$.
- Can drive 2 DC motors
- Enable line controls speed of motors if PWM is applied.
- L298 can be supplied separately from the motor voltages

Control	Purpose
IN1	Input 1 for Motor A
IN2	Input 2 for Motor A
EN1	Enable line for Motor A

IN1	IN2	Direction
0	0	Motor OFF
5V	0	Forward
0	5V	Reverse
5V	5V	Not used

H-bridge Simulation



Simulation

H-bridge (L293) control of a dc motor.

Use SW to control direction of motor (IN1, IN2)

Use POT to control speed of the motor (EN1)

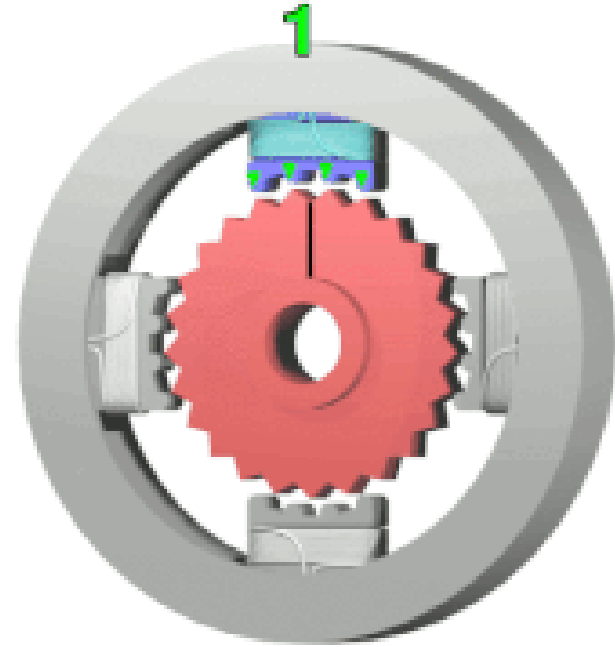
H-bridge code

```
5  const int IN1 = 9;
6  const int IN2 = 8;
7  const int EN1 = 10;
8  const int SW = 12;
9  const int POT = A0;
10 int speed = 0;
11
12 void setup()
13 {
14     pinMode(IN1, OUTPUT);
15     pinMode(IN2, OUTPUT);
16     // pinMode(EN1, OUTPUT);
17     pinMode(SW, INPUT_PULLUP);
18 }
19
20 void loop()
21 {
22     int ain = analogRead(POT);
23     int newspeed = map(ain, 0, 1023, 0, 255);
24     if (newspeed != speed){
25         // change the speed
26         speed = newspeed;
27     }
```

```
28     // set direction
29     if (digitalRead(SW) == HIGH)
30     {
31         // forward
32         digitalWrite(IN1, LOW);
33         digitalWrite(IN2, HIGH);
34     }
35     else
36     {
37         // reverse
38         digitalWrite(IN1, HIGH);
39         digitalWrite(IN2, LOW);
40     }
41     // set the speed using PWM
42     analogWrite(EN1, speed);
43 }
44
```

Stepper Motors

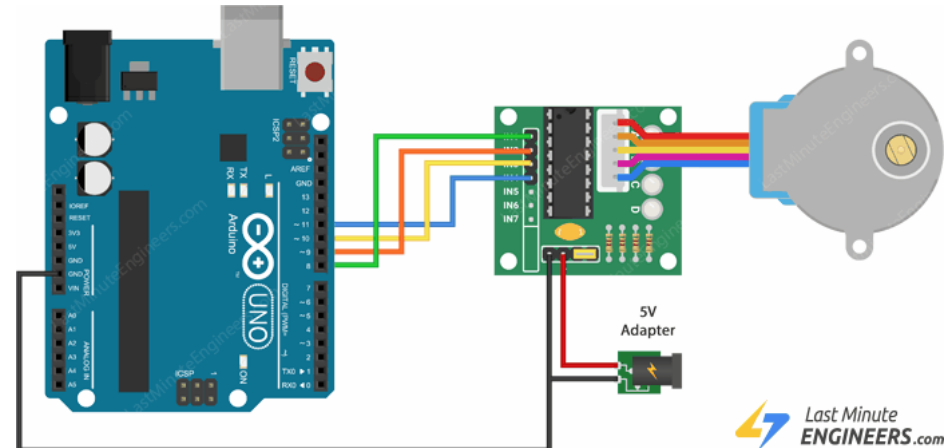
- A motor that can move in steps.
- Each step is a division (in degrees) of a full rotation.
- Done by activation motor coils.
- Stepper Motors are good
 - For exact positional control
 - Movement at specific speeds
 - High torque (even at rest)
- Have Stepper control modules



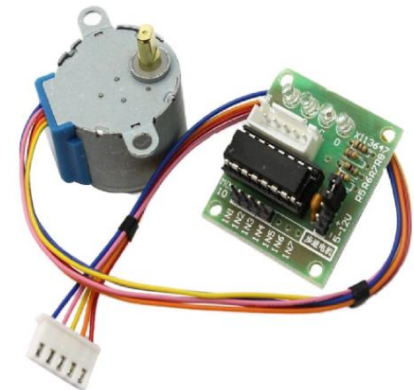
- References:
 - [Wikipedia: Stepper Motor](#)
 - [Dronebot Workshop](#)
 - [LastMinuteEngineers](#)

Simple Stepper Motor: 28BYJ-48

- Common hobby motor with controller board for many projects.
- Able to work with 5V supply (external)
- Can be controlled manually or with a library
- Library: [Arduino Stepper](#)
- References:
 - [LastMinuteEngineers](#)
 - [Dronebot workshop](#) (has manual code and library examples)



 Last Minute
ENGINEERS.com



Stepper Motor Code

```
2
3 //Includes the Arduino Stepper Library
4 #include <Stepper.h>
5 // Defines the number of steps per rotation
6 const int stepsPerRevolution = 2038;
7 // Creates an instance of stepper class
8 // Pins entered in sequence
9 // IN1-IN3-IN2-IN4 for proper step sequence
10 Stepper myStepper = Stepper(stepsPerRevolution,
11                               8, 10, 9, 11);
12
13 void setup() {
14     // Nothing to do
15 }
16
17 void loop() {
18     // Rotate CW slowly
19     myStepper.setSpeed(100);
20     myStepper.step(stepsPerRevolution);
21     delay(1000);
22
23     // Rotate CCW quickly
24     myStepper.setSpeed(700);
25     myStepper.step(-stepsPerRevolution);
26     delay(1000);
27 }
28
```

- Uses stepper motor library
- There is a min and max speed the motor can achieve.

Servo Motors

- Has an internal control circuit with geared motors
- Controlled using PWM
- Pulse width determine
 - Position of servo
 - Direction of rotation
- Two types
 - [Positional \(e.g. FS90\)](#)
 - [Continuous Rotational \(e.g. FS90R\)](#)
- Positional Servo (FS90)
 - 3.0~5.0V, 120° angle
 - Servo pulse 900~1200uS determines position
 - Maintains control position
- Continuous Rotation (FS90R)
 - 3.0~5.0V, 360° rotation
 - Servo pulse 900~1200uS controls direction of rotation or speed
 - For small robots or moving objects

EP1000

Actuators

End