INTRO TO STEPPER MOTORS

DC motors with precise control of how far they spin
- They have a fixed number of “steps” the take to turn one full revolution
- You can control them one step at a time
- Makes for very precise and repeatable positioning
Why use steppers?

- Very precise
- Much stronger than servos
  - But, they use more current than Arduino can provide
  - So, you need some sort of external power source
- They’re a little tricky to drive
  - So, you need some sort of code library, or external driver board
- We’ll use both – a library, and an external board

They always have multiple wires
How do they Work?

- Like all motors – electro-magnets get energized and push/pull the rotor around.

Steppers have precise internals
Steppers have precise internals
Steppers have precise internals

Stepper Internals

Hybrid Stepper
Different Flavors of Steppers

- 8-wire unipolar
- 6-wire unipolar
- 4-wire bipolar

Bipolar is generally the most powerful and efficient but slightly harder to control.
Make it Turn

- Energize the coils in a very specific sequence

Unipolar Motor and Drive

Bipolar Motor and Drive
## Make it Turn

- Energize the coils in a very specific sequence

### Alternate Full Step Sequence (Provides more torque)

<table>
<thead>
<tr>
<th>Index</th>
<th>1a</th>
<th>1b</th>
<th>2a</th>
<th>2b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Half Step Sequence

<table>
<thead>
<tr>
<th>Index</th>
<th>1a</th>
<th>1b</th>
<th>2a</th>
<th>2b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

## Use a Library

### Functions

- **Stepper**(steps, pin1, pin2)
- **Stepper**(steps, pin1, pin2, pin3, pin4)
- **setSpeed**(rpm)
- **step**(steps)

### Example

- **Motor Knob**
Knob Example

```c
#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;
}
```

Back up: what’s a Class?

- A way of packaging things up into a little code bundle
  - An instance of a class can remember things about itself
  - An instance of a class can do things when you tell it to
  - Each instance of the class can behave differently

- Let’s back up and look carefully at the servo code
Servo + light sensor

```c
#include <Servo.h>

Servo myservo; // create servo object to control a servo
int potpin = 0; // analog pin used to connect the potentiometer
int val; // variable to read the value from the analog pin

void setup()
{
  myservo.attach(9); // attaches the servo on pin 9 to the servo object
}

void loop()
{
  val = analogRead(potpin); // reads the value of the potentiometer

  val = map(val, 0, 1023, 0, 179); // scale it to use it with the servo
  myservo.write(val); // sets the servo position according to the value
  delay(15); // waits for the servo to get there
}
```

Servo Object (class instance)
Stepper Object

Stepper object: mystepper

Internal state:
- What step am I on?
- What sequence do I use?

Drivers

..setSpeed
..step

---

Stepper Object

Stepper object: mystepper

Internal state:
- What step am I on?
- What sequence do I use?
Driver Circuits… (unipolar)

Driver Circuits… (bipolar)

Bipolar Motor and Drive
Driver Circuits... (bipolar)

Stepper Specs

- **Degrees/Step**
  - Common values: 15, 7.5, 3.6, 1.8 deg/step
  - This is the same as 24, 48, 100, and 200 steps/full-rev

- **Coil Resistance**
  - Measured resistance of motor coils

- **Volts/Amps**
  - Rated values for running the motor
  - Amps is the important one!
  - Remember $V=IR$, so $V/R = I$
  - Example: 6VDC, 7.9 $\Omega = .76A$
So far...

- Steppers move very precisely and are relatively powerful

- But are a bit of a pain to drive
  - Four wires from the Arduino
  - External driver circuits
  - Extra power supply to worry about
  - Use stepper library to make stepper “objects” for each one that you use
  - Your program needs to keep track of how far you’ve turned

Easier Motor Driving...

- There are chips specifically designed for driving steppers
  - They manage the sequence of signals
  - They manage the higher voltages of the motors
  - They have “chopper drives” to limit current
  - They can even do “microstepping”
    - This lets you do $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, or $\frac{1}{16}$ step
    - Increases resolution and smoothness, but might reduce power
Pololu A4988 driver

Up to 2A per coil (with heat sink)

8 – 35V on motor
Provides 5v to Arduino

Limits current to a set level

Uses only 2 wires for control: Dir, Step

<table>
<thead>
<tr>
<th>MS1</th>
<th>MS2</th>
<th>MS3</th>
<th>Microstep Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Full step</td>
</tr>
<tr>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Half step</td>
</tr>
<tr>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>Quarter step</td>
</tr>
<tr>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>Eighth step</td>
</tr>
<tr>
<td>High</td>
<td>High</td>
<td>High</td>
<td>Sixteenth step</td>
</tr>
</tbody>
</table>

Dimensions:

- Width: 0.7"
- Height: 1.4"
Pololu A4988 driver

Minimal Connection

Need one driver for each motor

Check REF to see what the current limit will be

Need to set this pot to set current limit

power supply (8–35V)
Current Limit

- Turn pot (use a tiny screwdriver) and check REF
  \[ I_{\text{TripMAX}} = \frac{V_{\text{REF}}}{(8 \times R_s)} \]
- \( R_s = 0.05 \Omega \)

<table>
<thead>
<tr>
<th>V REF</th>
<th>Current Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>.1v</td>
<td>.250A</td>
</tr>
<tr>
<td>.15v</td>
<td>.375A</td>
</tr>
<tr>
<td>.2v</td>
<td>.500A</td>
</tr>
<tr>
<td>.25v</td>
<td>.625A</td>
</tr>
<tr>
<td>.3v</td>
<td>.750A</td>
</tr>
<tr>
<td>.35v</td>
<td>.875A</td>
</tr>
<tr>
<td>.4v</td>
<td>1.000A</td>
</tr>
<tr>
<td>.45v</td>
<td>1.125A</td>
</tr>
</tbody>
</table>

Using a Dir/Step driver

- Set the Dir pin – 0 is one dir, 1 is the other
- Toggle the Step pin up and down
  - You get one step per rising edge

```c
for (int i=0; i < steps; i++) {
  digitalWrite(STEP_PIN, HIGH);
  delayMicroseconds(usDelay);
  digitalWrite(STEP_PIN, LOW);
  delayMicroseconds(usDelay);
}
```
Use StepperDS Library

```
#include "StepperDS.h"

// 1.8 degrees per step = 200 steps per revolution
#define STEPS 200

// define the dir and step pins
#define DIR_PIN 8
#define STEP_PIN 9

// create an instance of the StepperDS class, specifying
// the number of steps of the motor and the pins it's attached to
StepperDS stepper(STEPS, DIR_PIN, STEP_PIN);

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

void setup() {
  // set the speed of the motor to 100 RPMs
  stepper.setPnm(100);
}

void loop() {
  int val = analogRead(8); // get the sensor value
  // save a number of steps equal to the change in sensor reading
  stepper.step(val - previous);
  previous = val; // remember the previous value of the sensor
}
```
Use the StepperDS Library

```cpp
void loop() {
  // rotate a specific number of degrees
  Serial.println("rotateDeg(353) - 100 RPM i.e. clockwise fast");
  stepper.setSpeed(100);
  stepper.stepDeg(353);
  delay(1000);

  Serial.println("rotateDeg(-37.5) - at speed 15 i.e. CW slow");
  stepper.setSpeed(15);
  stepper.stepDeg(-37.5); // reverse
  delay(1000);

  // rotate a specific number of microsteps (default is full-step)
  // 1.8 degree motors have 200 steps per revolution
  Serial.println("rotate(400) CW at half speed two times around");
  stepper.setSpeed(90);
  stepper.step(400);
  delay(1000);

  Serial.println("rotate(-210) CW at quarter speed");
  stepper.setSpeed(45);
  stepper.step(-210); // reverse
  delay(1000);
}
```

Finale

- Use one Pololu driver for each stepper
  - Measure REF, turn pot, to set current limit
  - Set microstepping if desired
  - Use stepperDS library (on web site)
    - Unzip the stepperDS.zip file in your Arduino/libraries folder
- Big motors are around ~1A, smaller are ~.5
  - Both are 1.8 deg/step (200 steps/rev)
Next Assignment

- Use a pair of steppers to make a suspended-style drawing machine
  - Use same pairs as for museum assignment
  - Due 2/28 – 3/1
- Grab a couple motors, and a couple drivers
  - Use timing belt if you like, or design your own mechanism…
  - Make a simple suspended drawing machine, or interpret this in your own way…
  - Draw randomly, or with intent…