```
/*
 * created by Rui Santos, https://randomnerdtutorials.com
 *
 * Complete Guide for Ultrasonic Sensor HC-SR04
 *
    Ultrasonic sensor Pins:
        VCC: +5VDC
        Trig : Trigger (INPUT) - Pin11
        Echo: Echo (OUTPUT) - Pin 12
        GND: GND
 */
#include <Adafruit_NeoPixel.h>

#define PIN      2
#define N_LEDS 15
Adafruit_NeoPixel strip = Adafruit_NeoPixel(N_LEDS, PIN, NEO_GRB + NEO_KHZ800);

// end neopixel

int trigPin = 4;    // Trigger
int echoPin = 3;    // Echo
long duration, cm, inches;
#define ACT 5
boolean control_sound = false;

// end sensor

#include <Stepper.h>

//definitions for each command to be recieved via serial port
#define COMMAND_LEFT 'a'
#define COMMAND_RIGHT 'd'
#define COMMAND_FORWARD 'w'
#define COMMAND_BACK 's'
#define COMMAND_STOP ' '

//enter the steps per rev for your motors here
int stepsInRev = 200;

//this sets the value for the for loops and therefore sets the amount of steps in
each call
int num_of_steps = 1;
// setup pins for each driver motor1 ~ IN1, IN2, IN3, IN4; motor2 ~ IN1, IN2,
IN3, IN4
Stepper myStepper1(stepsInRev, 10, 11, 12, 13);
Stepper myStepper2(stepsInRev, 6, 7, 8, 9);
// variable to store the last call to the serial port
char lastCall = ' ';

//to move the motors forward
void forwardStep(int steps){
  Serial.println("forward");
  // step one step each forward
  myStepper1.step(1);
  myStepper2.step(1);
  delay(15);
}

// to move the motors back
void backwardStep(int steps){
  Serial.println("backward");
  // step one step each backward
  myStepper1.step(-1);
  myStepper2.step(-1);
  delay(10);
}


// to move the motors in opposite directions (left)
void leftStep(int steps){
  Serial.println("left");
 // step one step each left
  myStepper1.step(1);
  myStepper2.step(-1);
  delay(10);
}


// to move the motors in opposite directions (right)
void rightStep(int steps){
  Serial.println("right");
  // step one step each right
  myStepper1.step(-1);
  myStepper2.step(1);
  delay(10);
}

// to power down the motor drivers and stop the motors
void allStop(){
  Serial.println("stop");
  // steppers stop
//  PORTD = B00000000; //sets all of the pins 0 to 7 as LOW to power off stepper1
//  PORTB = B00000000; //sets all of the pins 8 to 13 as LOW to power off stepper2
}

// end stepper motor



void setup() {
  //Serial Port begin
  Serial.begin (9600);
  //Define inputs and outputs
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

// end sensor

// SOUND
  pinMode(14, OUTPUT);
  pinMode(15, OUTPUT);
  pinMode(16, OUTPUT);
  pinMode(17, OUTPUT);
  pinMode(18, OUTPUT);
  pinMode(ACT, INPUT_PULLUP);

  digitalWrite(14, HIGH);
  digitalWrite(15, HIGH);
  digitalWrite(16, HIGH);
  digitalWrite(17, HIGH);
  digitalWrite(18, HIGH);
    strip.begin();
}

int getDistance() {
  // The sensor is triggered by a HIGH pulse of 10 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the signal from the sensor: a HIGH pulse whose
  // duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);

  // Convert the time into a distance
  //cm = (duration/2) / 29.1;     // Divide by 29.1 or multiply by 0.0343
  return inches = (duration/2) / 74;   // Divide by 74 or multiply by 0.0135

//  Serial.print(inches);
//  Serial.print("in, ");
//  Serial.print(cm);
//  Serial.print("cm");
//  Serial.println();

//end sound

  myStepper1.setSpeed(60);
  myStepper2.setSpeed(60);

//end stepper motor

}

static void chase(uint32_t c, int t) {
  for(uint16_t i=0; i<strip.numPixels()+4; i++) {
      strip.setPixelColor(i  , c); // Draw new pixel
      strip.setPixelColor(i-4, 0); // Erase pixel a few steps back
      strip.show();
      delay(t);
  }
}

void determineState(int d) {
  int play = -1;

  if(d < 15) {
    chase(strip.Color(255, 0, 0), 5);
    //delay(1000);
    digitalWrite(14, LOW);
    forwardStep(200);
  } else
    digitalWrite(14, HIGH);

  if((d > 15) && (d < 30))  { chase(strip.Color(0, 255, 0), 15);
digitalWrite(15, LOW); leftStep(200); } else digitalWrite(15, HIGH);
  if((d > 30) && (d < 50))  { chase(strip.Color(0, 90, 100), 30);
```

```
digitalWrite(16, LOW); rightStep(200); } else digitalWrite(16, HIGH);
  if((d > 50) && (d < 80))  { chase(strip.Color(60, 90, 100), 45);
digitalWrite(17, LOW); backwardStep(200); } else digitalWrite(17, HIGH);
  if((d > 80) && (d < 100)) { chase(strip.Color(100, 10, 128), 100);
digitalWrite(18, LOW); forwardStep(200); } else digitalWrite(18, HIGH);
  if(d > 150) Serial.println("FAR");
//
//    while(digitalRead(ACT) == LOW) {
//      delay(1500);
//   }


  //digitalWrite(play, HIGH);
  //digitalWrite(14, HIGH);
//  digitalWrite(15, HIGH);
//  digitalWrite(16, HIGH);
//  digitalWrite(17, HIGH);
//  digitalWrite(18, HIGH);
}



void loop() {
  //determineState(getDistance());
  //delay(50);
forwardStep(1);
//end sensor

////check to see if there is serial communication and if so read the data
//if(Serial.available()) {
//char data = (char)Serial.read();
//// switch to set the char via serial to a command
//switch(data) {
//  case COMMAND_FORWARD:
//    forwardStep(num_of_steps);
//    break;
//  case COMMAND_BACK:
//    backwardStep(num_of_steps);
//    break;
//  case COMMAND_LEFT:
//    leftStep(num_of_steps);
//    break;
//  case COMMAND_RIGHT:
//    rightStep(num_of_steps);
//    break;
//  case COMMAND_STOP:
```

```
//    allStop();
//    break;
//}
//// set the 'lastCall' variable to the last call from the serial
//lastCall = data;
//}
//else{
//char data = lastCall;
//switch(data) {
//  case COMMAND_FORWARD:
//    for(int i=0; i<5; i++)
//    {
//      forwardStep(num_of_steps);
//     }
//    break;
//  case COMMAND_BACK:
//    backwardStep(num_of_steps);
//    break;
//  case COMMAND_LEFT:
//    leftStep(num_of_steps);
//    break;
//  case COMMAND_RIGHT:
//    rightStep(num_of_steps);
//    break;
//  case COMMAND_STOP:
//    allStop();
//    break;
//}
//lastCall = data;
}

//end stepper motor
//}
```