```
/*
  AnalogReadSerial

  Reads an analog input on pin 0, prints the result to the Serial
Monitor.
  Graphical representation is available using Serial Plotter (Tools >
Serial Plotter menu).
  Attach the center pin of a potentiometer to pin A0, and the outside
pins to +5V and ground.

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/AnalogReadSerial
*/
const int trigPin = 11;           //connects to the trigger pin on the
distance sensor
const int echoPin = 12;           //connects to the echo pin on the
distance sensor
// the setup routine runs once when you press reset:

float distance = 0;               //stores the distance measured by
the distance sensor

int speakerPin = 10;               //the pin that buzzer is connected
to


void setup()
{
  Serial.begin (9600);        //set up a serial connection with the
computer

  pinMode(trigPin, OUTPUT);   //the trigger pin will output pulses of
electricity
  pinMode(echoPin, INPUT);    //the echo pin will measure the duration
of pulses coming back from the distance sensor
  pinMode(speakerPin, OUTPUT);    //set the output pin for the speaker

}

// the loop routine runs over and over again forever:


void loop() {

  distance = getDistance();   //variable to store the distance
measured by the sensor

  Serial.print(distance);     //print the distance that was measured
  Serial.println(" in");      //print units after the distance
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);   // 0 - 1023  (0 to 5V)

  // make brightness change according to sensorValue
  // analogWrite(9, sensorValue/4 );  // 0 - 255  (0 to 5V)
  // better way:
  // brightness = map(sensorValue, 0, 890);
  //  0,255)
  // print out the value you read:

  Serial.print(sensorValue);
  Serial.print(" & ");
  Serial.println(distance);

  delay(500);        // delay in between reads for stability
//
  if (distance<=1) {

  play('g', 2);       //ha
  play('g', 1);       //ppy
  play('a', 4);       //birth
  play('g', 4);       //day
  play('C', 4);       //to
  play('b', 4);       //you

  play(' ', 2);       //pause for 2 beats

  play('g', 2);       //ha
  play('g', 1);       //ppy
  play('a', 4);       //birth
  play('g', 4);       //day
  play('D', 4);       //to
  play('C', 4);       //you

  play(' ', 2);       //pause for 2 beats

  play('g', 2);       //ha
  play('g', 1);       //ppy
  play('G', 4);       //birth
  play('E', 4);       //day
  play('C', 4);       //dear
  play('b', 4);       //your
  play('a', 6);       //name

  play(' ', 2);       //pause for 2 beats

  play('F', 2);       //ha
  play('F', 1);       //ppy
  play('E', 4);       //birth
  play('C', 4);       //day
  play('D', 4);       //to
  play('C', 6);       //you

  }
  Serial.print(sensorValue);
  Serial.print(" & ");
  Serial.println(distance);

  delay(500);        // delay in between reads for stability
}




   void play( char note, int beats)
{
  int numNotes = 14;  // number of notes in our note and frequency
array (there are 15 values, but arrays start at 0)

  //Note: these notes are C major (there are no sharps or flats)

  //this array is used to look up the notes
  char notes[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C', 'D', 'E',
'F', 'G', 'A', 'B', ' '};
  //this array matches frequencies with each letter (e.g. the 4th note
is 'f', the 4th frequency is 175)
  int frequencies[] = {131, 147, 165, 175, 196, 220, 247, 262, 294,
330, 349, 392, 440, 494, 0};

  int currentFrequency = 0;    //the frequency that we find when we
look up a frequency in the arrays
  int beatLength = 150;   //the length of one beat (changing this will
speed up or slow down the tempo of the song)

  //look up the frequency that corresponds to the note
  for (int i = 0; i < numNotes; i++)  // check each value in notes
from 0 to 14
  {
    if (notes[i] == note)             // does the letter passed to the
play function match the letter in the array?
    {
      currentFrequency = frequencies[i];   // Yes! Set the current
frequency to match that note
    }
  }

  //play the frequency that matched our letter for the number of beats
passed to the play function
```

```
  tone(speakerPin, currentFrequency, beats * beatLength);
  delay(beats * beatLength);  //wait for the length of the tone so
that it has time to play
  delay(50);                  //a little delay between the notes makes
the song sound more natural

}





//RETURNS THE DISTANCE MEASURED BY THE HC-SR04 DISTANCE SENSOR
float getDistance()
{
  float echoTime;                   //variable to store the time it
takes for a ping to bounce off an object
  float calculatedDistance;         //variable to store the distance
calculated from the echo time

  //send out an ultrasonic pulse that's 10ms long
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  echoTime = pulseIn(echoPin, HIGH);      //use the pulsein command to
see how long it takes for the
  //pulse to bounce back to the sensor

  calculatedDistance = echoTime / 148.0;  //calculate the distance of
the object that reflected the pulse (half the bounce time multiplied
by the speed of sound)

  return calculatedDistance;              //send back the distance
that was calculated
}
```