5-2016

Smart Fan

Sang Choi
Trinity University, schoi1@trinity.edu

Shawn Sunday
Trinity University, ssunday@trinity.edu

Smart Fan

ENGR 4367

Dr. Nickels

Group 1

Sang Choi and Shawn Sunday

05/02/16

Table of Contents

1. Design Summary

The conventional desk fans rely on manual user inputs such as buttons or some sort of mechanical inputs that require user's physical contact. The Smart Fan, however, is a fan that is purely operated by sound as user input to adjust speed and directionality of the fan. The device listens for user to provide sound input such as clapping, snapping or tapping through microphones attached on the front and the sides of the device.

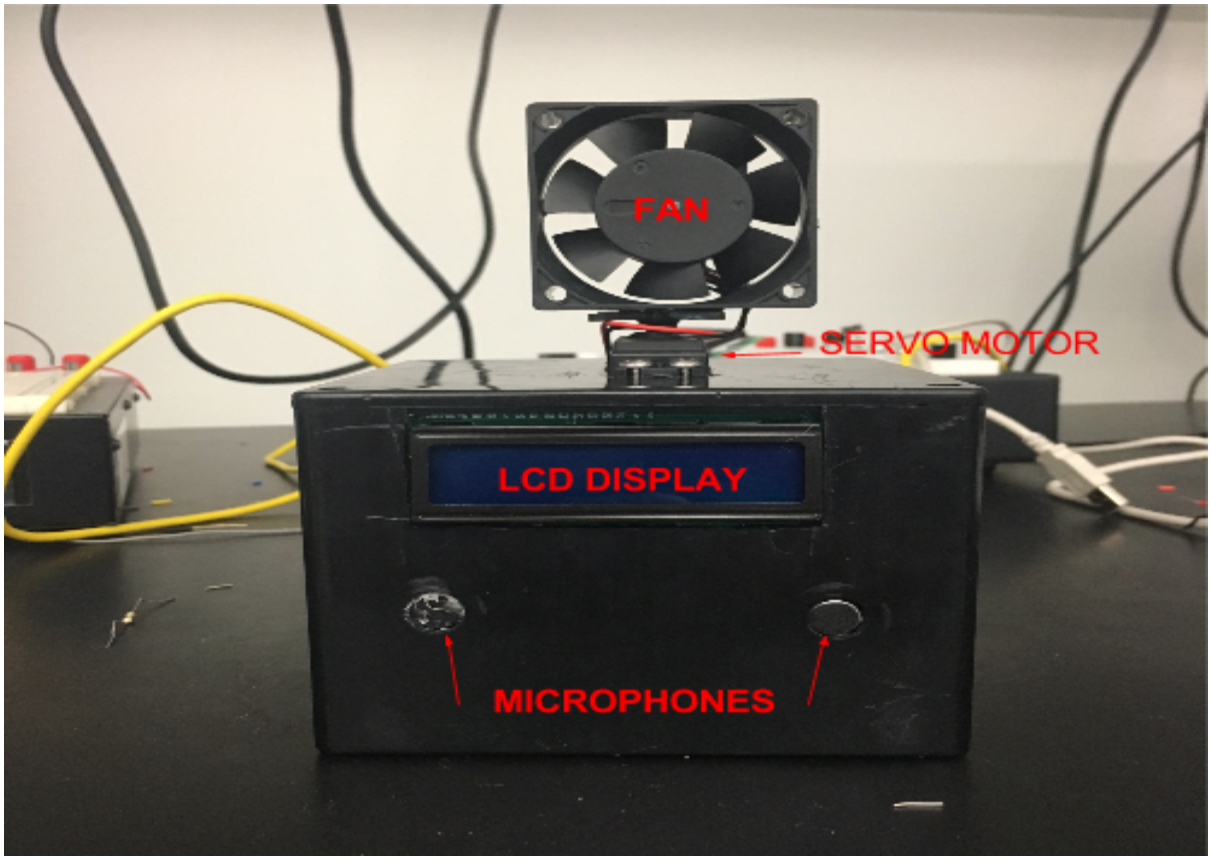

Figure 1: Front View of the Smart Fan

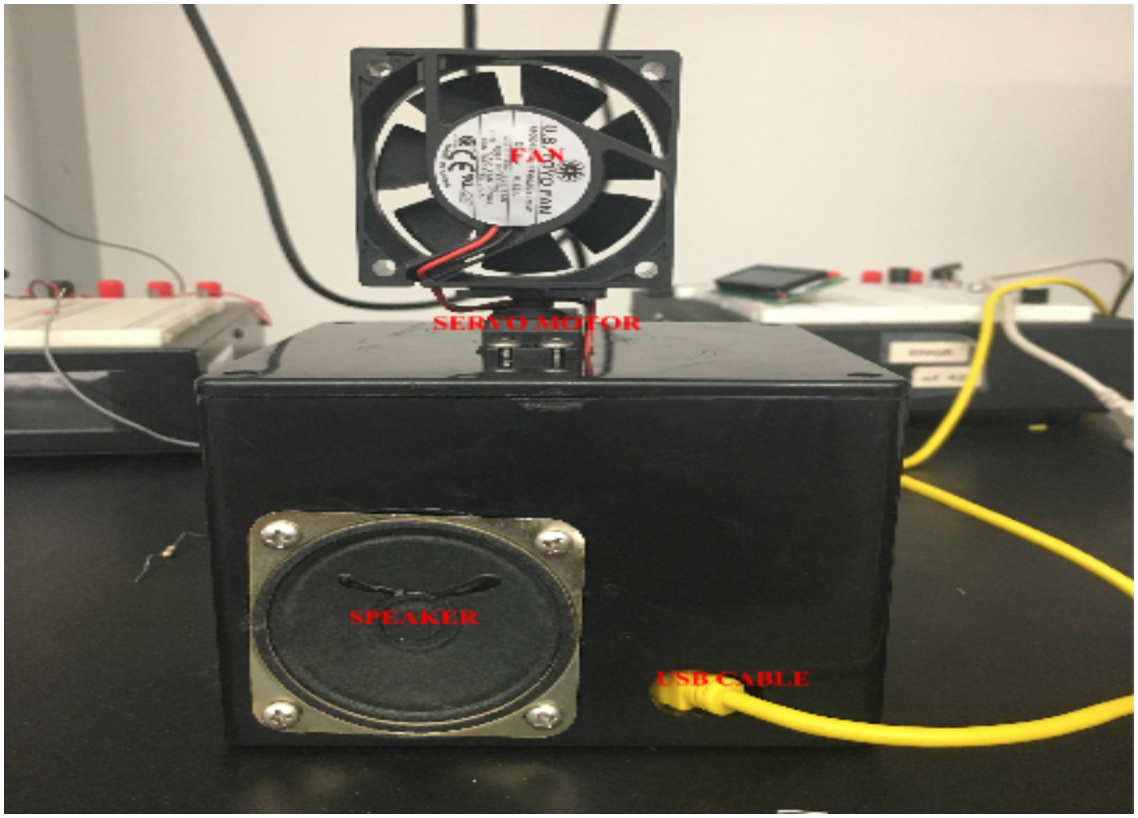

Figure 2: Back View of the Smart Fan

Once the sound is registered, Smart Fan points its fan towards where the sound came from and adjusts its speed to the loudness of the sound.

2. System Details

In figure 1 you can see the fan that rotates and the speed changes on a clap. The servo motor allows the fan to rotate to four different positions according to which microphone detected the clap. The LCD display shows the speed and direction of the fan. Two of the microphones are also shown, they are responsible for recording claps. The other two are located on the sides. The fan is powered by USB that is seen in figure 2. Figure 2 also shows the speaker which plays

a sound when the fan is turned on and when the fan records a clap.`

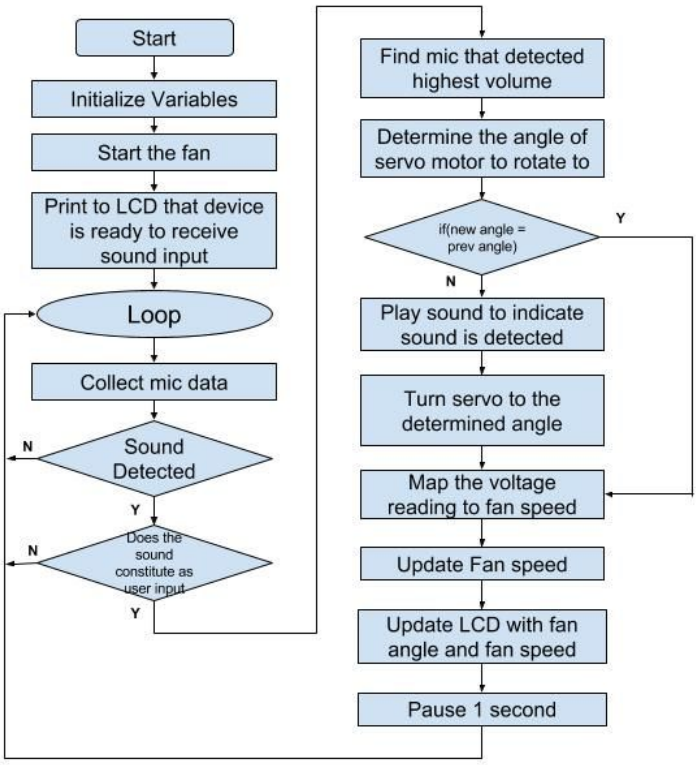

Figure 3: Software Flow Chart

Figure 3 shows the software flow chart of the main code ran on the Arduino that was used for the device. In the initializing mode, the fan is turned on at full speed to overcome the static friction of the fan. Once all the variables and ports are initialized, the microphones start to collect data and after it heard a loud enough sound, the code checks if the sound qualifies as an user input. If so, the fan was turned in the direction of the closest microphone and the speed is determined by how loud the sound was. If the angle that the fan is rotating to is the same as the angle the device was just point at, the device does not update the angle but only update the speed of the fan. However, if this is not the case, the device updates both the angle and the speed of the fan.

Once the properties of the fan are changed, the device waits for 1 second to ensure that fan and the servo are nicely settled.

3. Design Evaluation

The Smart Fan device was a success in that it fulfilled the original intent of the device. The device also satisfied all the functional element categories the instructor provided. The device consists of LCD display as shown in Figure 1. The display was included in the design to give user information what mode the device is on, what the speed of the fan is , and where the fan is currently directed at. As for an audio output, Smart Fan has a speaker on the back as shown in Figure 2. The speaker plays a sound to indicate to the user that the device is in the initializing mode and also when a sound input is registered. A potentiometer is a manual user input of the device that lets the user to adjust contrast of Liquid Crystal Display. The potentiometer is located inside the housing because there's no need to change this configuration frequently. For automatic sensor inputs, there are four microphones attached on the sides and the front of the device. Their functionalities are to collect sound input from the user. As for actuators, mechanisms, and hardware category, the Smart Fan uses a servo motor to rotate a fan to a particular direction of where the sound came from. All the circuitry and components are housed in an elegant black aluminum box. The device uses a microcontroller Arduino Uno, which is programmed to control a servo motor and a fan with sound inputs from microphones. The microcontroller also communicates with a microprocessor (PIC16F88) that controls the speaker.

The group believes the device deserves qualitative grading adjustment on construction quality, aesthetics, and consumer appeal. As seen in Figure 1, the wires and electrical

components that do not need to be exposed to the user are all hidden well inside a black box for aesthetics. Inside the box, all the circuits are nicely soldered to make sure they do not get disconnected anywhere. With an aluminum box to contain all the components as well as soldering to permanently fix all the circuits in place, the device is well protected and requires very low maintenance. The device is also very portable, making it a very appealing product to customers. One can easily move this device to any places they want and all it needs is an USB port for power supply, which is readily available anywhere these days.

4. Partial Parts List

Part	Quantity	Price	Vendor (Source)
U.S. Toyo Fan USTF602012MW	1	$9.26	Trinity University
Phantom YoYo JOSY Sound Sensors (Microphones)	4	$5.69	Amazon
Arduino Uno	1	$24.95	Trinity University

5. Lessons Learned

One of the biggest problems the group faced was getting the microphones to work consistently. We tried using the microphones without the prepackaged circuit board. The microphones without the circuit board would not work at all sometimes. They would be connected just like the datasheet explains but still would only work sometimes by not recording any sounds at all or recording sounds when there was not any. We tried configuring the

microphones in different ways but still the microphones only worked sometimes. We fixed this problem by using the microphones with the prepackaged circuits. These microphones proved to be more consistent. The microphones also constantly produce signals and in order for them to not be recognized as a sound we had to set a certain threshold for each microphone.

The microphones gave out less false signals than the microphones used before but the group would still sometimes receive false signals from all four of the microphones. We resolved this problem by having the software ignore signals from all four of the microphones when they were sent at the same time and only read responses from up to two of the microphones at the same time. The group then decided to have the fan rotate to the microphone with the strongest reading. Sometimes after the microphones picked up a sound the signal from the microphone would linger. This was fixed by adding a delay after a microphone picked up a sound so that it would not register more sounds after the initial sound. Also one of our microphones would pick up an unusual amount of sound. In order to offset this, the threshold for this microphones was set higher than the other microphones. Also in order for the microphones to not record sounds from the motor or the speaker the data collection process is suspended during those operations. If another group were to try to recreate this project I would suggest they use high quality microphones and filters in order to get the best results.

6. Appendix

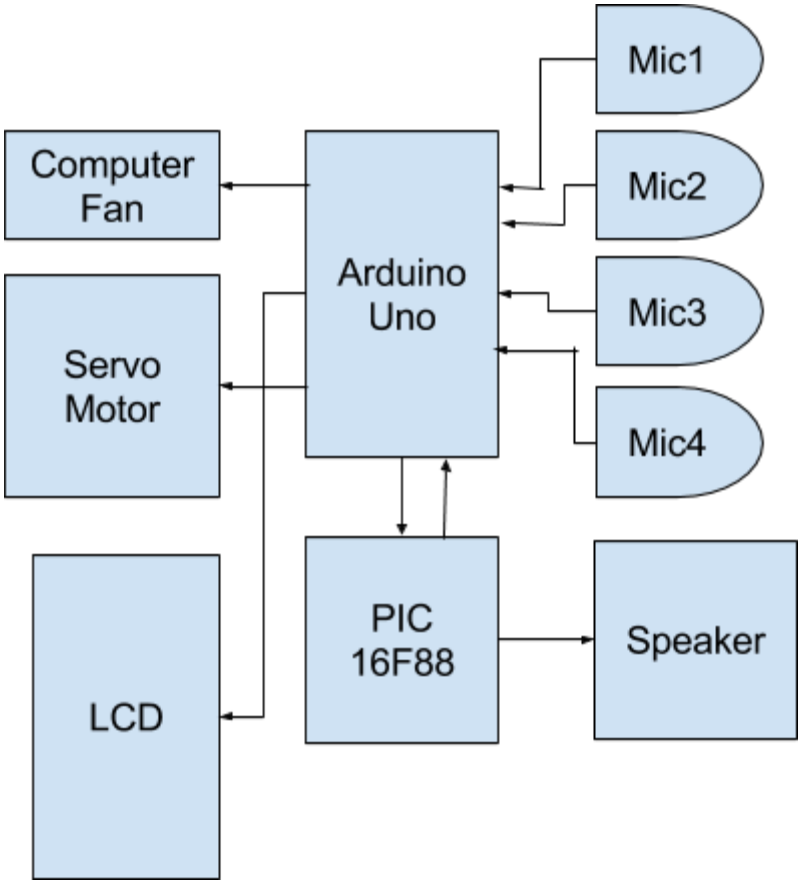

Figure 4: Functional Diagram

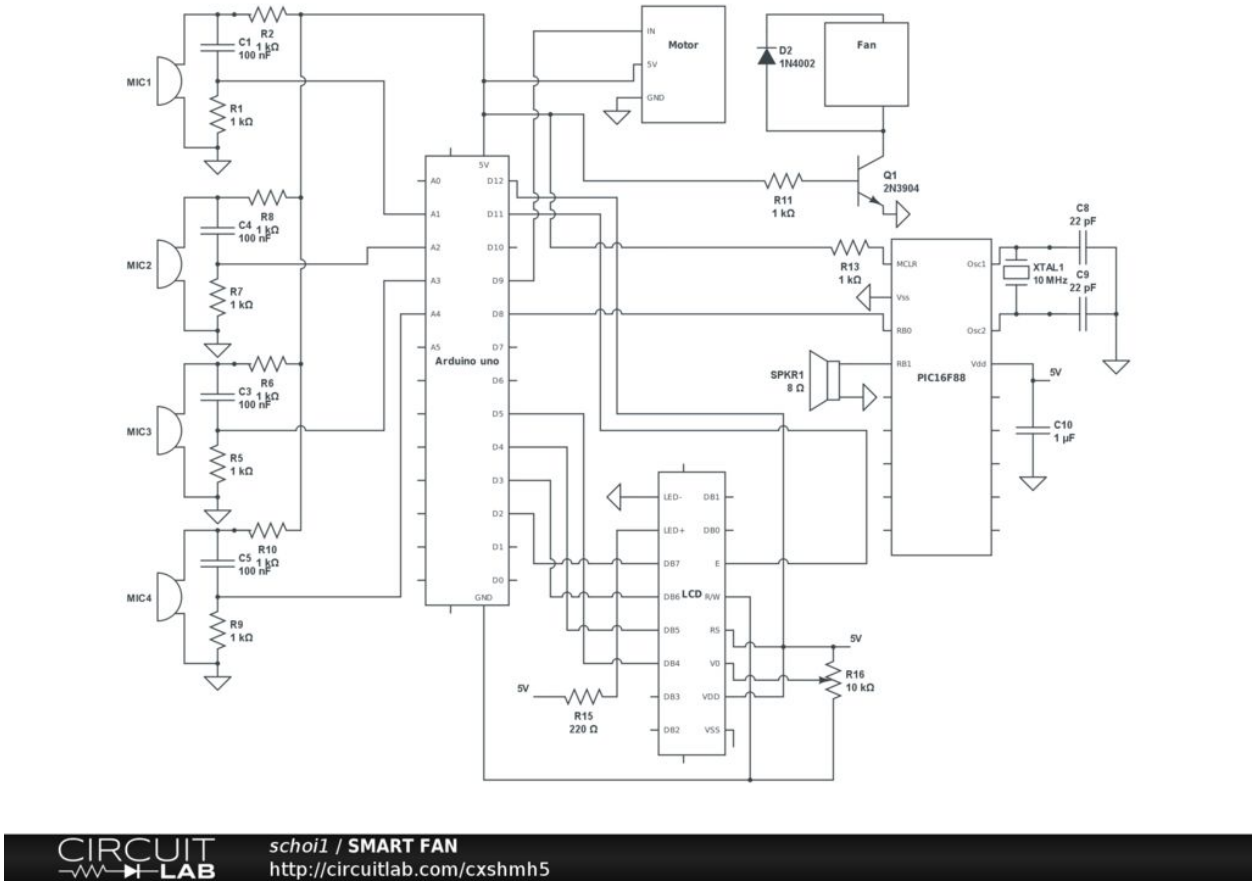

Figure 5: Circuit Diagram

```
#include <Average.h>
#include <arduino.h>
#include <Servo.h>
#include <LiquidCrystal.h>

//pin Assignments
#define fanControl 6
#define soundControl 8
#define servoControl 9

int soundThres = 70;
int currentAngleIndex = 0;
int pastAngleIndex = 0;

Servo servoMotor;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

boolean fanOn  =false;
boolean soundDetected  =false;

int micData[4];
int angleArray[] = {0,60,120,180};

void setup() {
  Serial.begin(9600);
  //Initialize
  servoMotor.attach(servoControl);
  servoMotor.write(90);
  analogWrite(fanControl,255);
  lcd.begin(16,2);
  lcd.clear();
  lcd.print("INITIALIZING");
  lcd.setCursor(0,1);
  lcd.print("SMART FAN");
  delay(3000);
  pinMode(fanControl, OUTPUT);
  pinMode(soundControl, OUTPUT);
  pinMode(servoControl, OUTPUT);
  digitalWrite(soundControl, LOW);
  lcd.clear();
  lcd.print("WAITING FOR");
  lcd.setCursor(0,1);
  lcd.print("SOUND INPUT");
}

void loop() {
  getMicData();
  if(soundDetected  ==true) {
    findServoAngle();
  }
   soundDetected =false;
}
```

```
void getMicData(){
  //Read from all the mics
  int mic1  =analogRead(1);
  int mic2  =analogRead(2);
  int mic3  =analogRead(3);
  int mic4  =analogRead(5);
  //if the sound detected is louder than thet theshold they are considered user input
  if(mic1>soundThres+15 || mic2>soundThres || mic3>soundThres || mic4>soundThres+10){
    micData[0] = mic1;
    micData[1] = mic2;
    micData[2] = mic3;
    micData[3] = mic4;
    Serial.println();
     soundDetected =true; //Sound detected
  }
}

void findServoAngle(){
  int maxIndex = 0;
  int maxValue = micData[0];
  int secondMaxValue = 0;
  int count = 0;
  //count how many mics sensed sound amplitude of 15 or more
  //used for analyzing whether the sound is an user input or not later
  for (int y = 0; y<4; y++){
    if(micData[y]>=15)  count++;
  }

  //Find the mic that detected the highest amplitude and the second highest
  for(int x=1; x<4; x++){
    Serial.println(" ");
    if(micData[x]>=maxValue) {
      secondMaxValue = maxValue;
      maxValue = micData[x];
          maxIndex  =  x;
     }else if(micData[x]<maxValue && micData[x]>secondMaxValue) secondMaxValue = micData[x
  }
  //if the highest amplitude is more than 100 and the scond highest is over 90
  //the sound detected is not an user input
  //these values are derived from obesrvations during testing
  if(maxValue>100 && secondMaxValue>90) {
    Serial.println(maxValue);
    Serial.println(secondMaxValue);
    Serial.println("false signal detected");
    delay(100);
   }
  //mic4  was  found
  /*else if (micData[3] > 100) {
    Serial.println("false signal detected");
    Serial.println("mic4 issue");
      delay(100);
```

```
  */
   }
  //if 3 or more mics detected sound of ampltude 15 or higher
  //the sound is not registerd as an user input
  else if (count >= 3) {
    delay(100);
    Serial.println("flase signal detected");
    Serial.println("three or mmore sensors detected noise");
   }
  //if the signal satisfies all the conditions above
  //the sound detected is an user input
  else{
    pastAngleIndex = currentAngleIndex;
    currentAngleIndex = maxIndex;
    //if the angle to rotate to is not the same as angle before
    //rotate the servo
    //play sound through speaker
    if(currentAngleIndex!=pastAngleIndex){
      digitalWrite(soundControl,HIGH);
      servoMotor.write(angleArray[maxIndex]);
      printSerialData();
      digitalWrite(soundControl,LOW);
      delay(1000);
     }
    //update the lcd with new fan speed and angle
    lcd.clear();
    lcd.print("FAN RUNNING");
    Serial.println(angleArray[maxIndex]);
    Serial.print(currentAngleIndex);
    Serial.print(" ");
    Serial.print(pastAngleIndex);
    lcd.setCursor(0,1);
    lcd.print("SPD:");
    lcd.print(getSpeed(maxValue));
    lcd.print(" DEG:");
    lcd.print(angleArray[maxIndex]);
  }
//map sound amplitude to fan speed
}
int getSpeed(int maxValue){
  int fanSpeed = 0;
  if(maxValue>130) fanSpeed = 255;
  else  fanSpeed  =map(maxValue,soundThres,130,100,255);
  analogWrite(fanControl,fanSpeed);
  return fanSpeed;
}

void printSerialData() {
    Serial.print("mic1: ");
    Serial.print(micData[0]);
    Serial.print(" ");
    Serial.print("mic2: ");
```

```
    Serial.print(micData[1]);
    Serial.print(" ");
    Serial.print("mic3: ");
    Serial.print(micData[3]);
    Serial.print(" ");
    Serial.print("mic4: ");
    Serial.print(micData[4]);
    Serial.print(" ");
}
```

Figure 6: Arduino code of Smart Fan

```
'****************************************************************
'*  Name    : microphones.BAS                                   *
'*  Author  : Sang Choi Shawn Sunday                  *
'*  Notice  : Copyright (c) 2016 [select VIEW...EDITOR OPTIONS] *
'*          : All Rights Reserved                               *
'*  Date    : 4/4/2016                                          *
'*  Version : 1.0                                               *
'*  Notes   : signal processing for microphones                 *
'*          :                                                   *
'****************************************************************

 define OSC 4  ;4MHz Clock
 'Frequency for different notes'
 shg con 1568
 shef con 1244
 shc con 1046
 shbf con 932
 shaf con 830
 hg con 784
 hf con 698
 hef con 622
 hd con 587
 hc con 523
 hbf con 466
 haf con 415
 a con  440
 b con 494
 g con 392
 f con 349
 e con 330
 ef con 311
 d con 294
  c con 262
 bf con 233
 af con 208
 lg con 196
 lf con 174
 lef con 155
 ld con 146
 lc con 130
 lbf con 116
 laf con 103
 slg con 98
 slf con 87
 slef con 78
 sld con 73
 slc con 65

TRISA = %11111111  'port a inputs
ADCON1 = %10000000 ' Right-justify results (lowest 10 bits)
x var word

x = 200     'duration between each notes
pause 100
'freqout 0, x, c
'freqout 0, x, d
'freqout 0, x, e
'freqout 0, x, f
'freqout 0, x, g
'freqout 0, x, a
```

```
'freqout 0, x, b
'freqout 0, x, c

while(1)
' play note c whenever Arduino requests
if (portb.1 == 1)then
    freqout 0, x, c   'send signal to portb.0
    pause(1000)
endif

wend
```


Figure 7: Speaker code for PIC 16F88