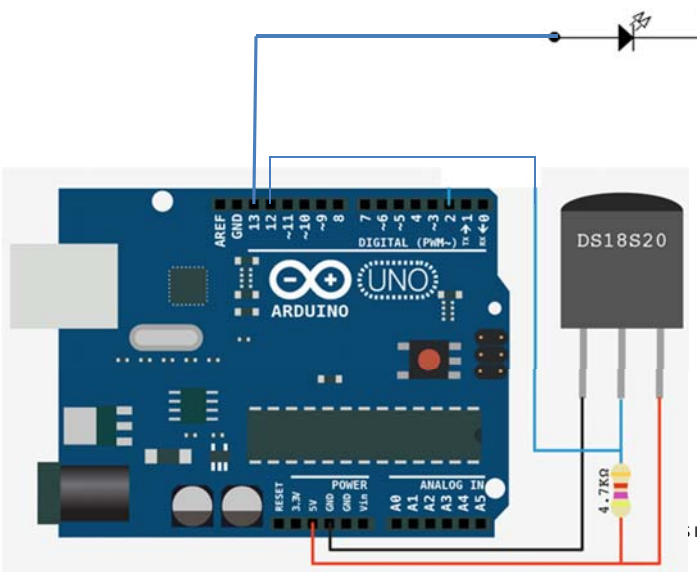
Lecture (04)
Practical Applications on Arduino Uno Board - 4

Dr. Ahmed ElShafee

TemperatureControlledDevice



TemperatureControlledDevice

```
#include <OneWire.h>
#include <DallasTemperature.h>
 #define ONE_WIRE_BUS 12
#define Alarm 13
OneWire
oneWire(ONE_WIRE_BUS);
DallasTemperature
sensors(&oneWire);

void setup(void)
{
  pinMode(Alarm,OUTPUT);
sensors.begin();
}
```

```
void loop(void)
{
sensors.requestTemperatures(); if
(sensors.getTempCByIndex(0)>25)
   digitalWrite(Alarm,HIGH);
 else
   digitalWrite(Alarm,LOW);
}
```


MelodyPlayer

The calculation of the tones is made following the mathematical operation:

```
timeHigh = period / 2 = 1 / (2 * toneFrequency)
```

where the different tones are described as in the table:

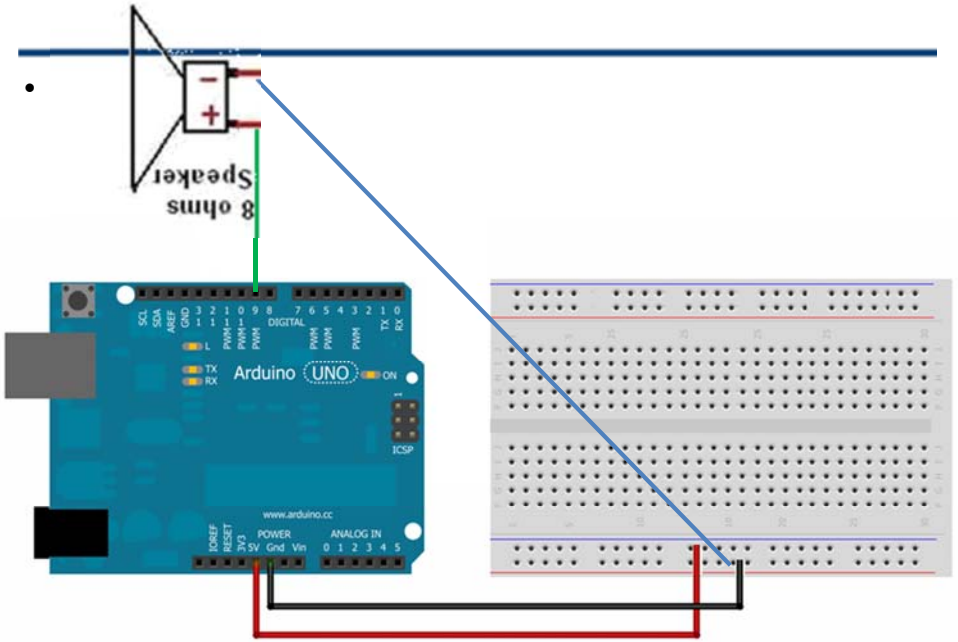
note	frequency	period	timeHigh
c	261 Hz	3830	1915
d	294 Hz	3400	1700
e	329 Hz	3038	1519
f	349 Hz	2864	1432
g	392 Hz	2550	1275
a	440 Hz	2272	1136
b	493 Hz	2028	1014
C	523 Hz	1912	956
		uSec	uSec

- tempo (Italian for time) is the speed or pace of a given piece.
- The tempo of a piece will typically be written at the start of a piece of music, and in modern Western music is usually indicated in **beats per minute** (**BPM**).
- Beats per minute (BPM) is a unit typically used as a measure of tempo in music and heart rate.

- Tempo=300mSec →BPM=60*1000/300=200bpm

Schematic

-




```
int speakerPin = 9;
int length = 15;
// the number of notes
char notes[] = "ccggaagffeeddc ";
// a space represents a rest
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1,
1, 1, 1, 1, 2, 4 };
int tempo = 300;
void playTone(int tone, int duration) {
  for (long i = 0; i < duration * 1000L; i
+= tone * 2)
  //counter 0 → 300 000, step= tone
width uSec, half is high, half is low
  {
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(tone);
    digitalWrite(speakerPin, LOW);
    delayMicroseconds(tone);
  }}
```

```
void playNote(char note, int duration) {
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a',
'b', 'C' };
  int tones[] = { 1915, 1700, 1519,
1432, 1275, 1136, 1014, 956 }; //uSec

  // play the tone corresponding to the
note name
  for (int i = 0; i < 8; i++) {
    if (names[i] == note) {
      playTone(tones[i], duration);
    }
  }
}
```



```
void setup() {
  pinMode(speakerPin, OUTPUT);
}

void loop() {
  for (int i = 0; i < length; i++) {
    if (notes[i] == ' ') {
      delay(beats[i] * tempo); // rest
    } else {
      playNote(notes[i], beats[i] * tempo);
    }

    // pause between notes
    delay(tempo / 2);
  }
}
```

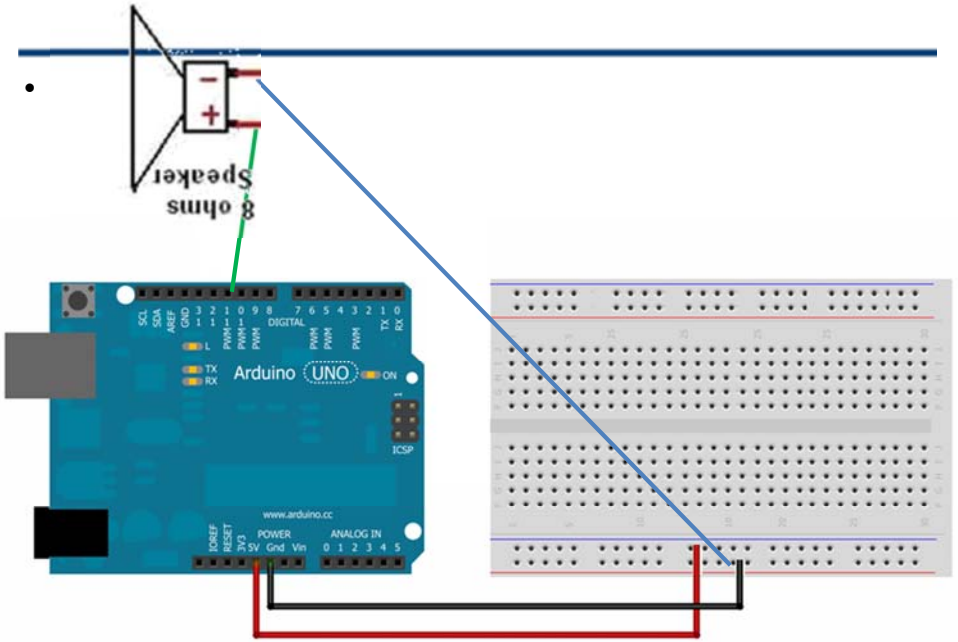
PlaySound

Steps

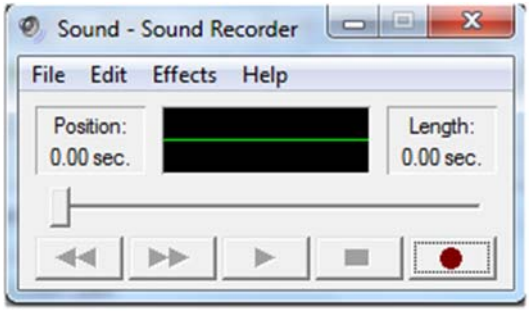
1. Record a wave file (8kHz sampleing rate, 8 bits/sample) 3 seconds
2. Convert file to array

Schematic

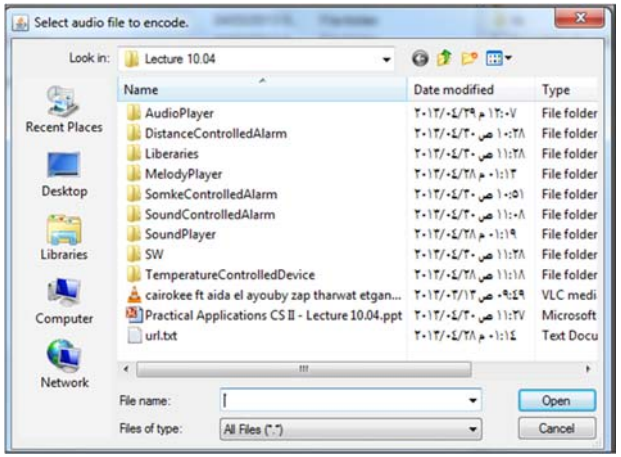
•



1. record



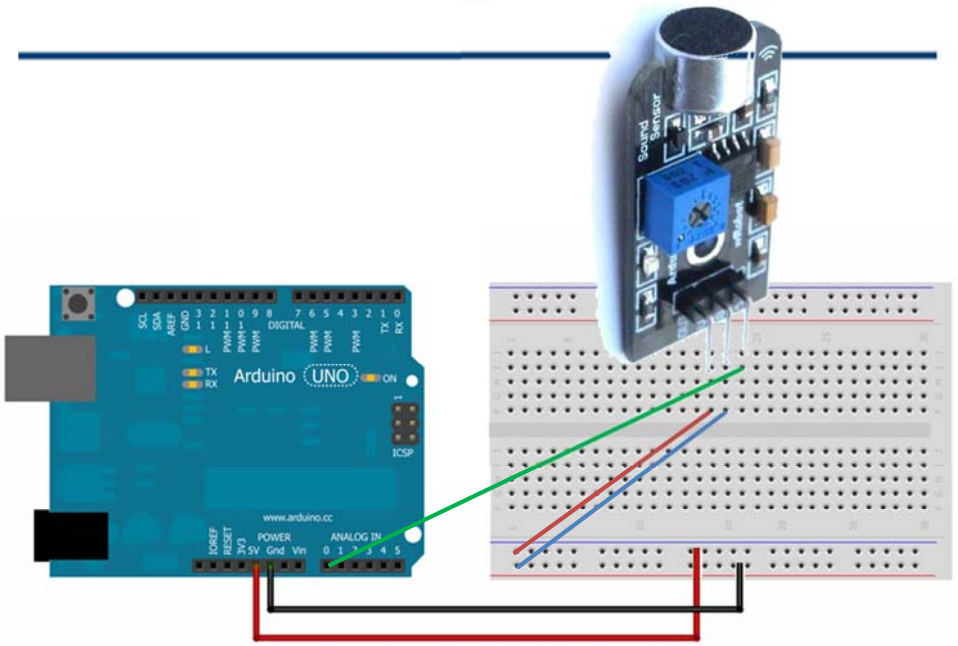
2. Convert file to array



```
#include <PCM.h>
const unsigned char sample[] PROGMEM=
//{126, 127,
128,128,128,128,128,127,128,128,128,129};
void setup(){
}

void loop(){
 startPlayback(sample,sizeof(sample));
 delay (5000);
}
```


SoundControlledAlarm



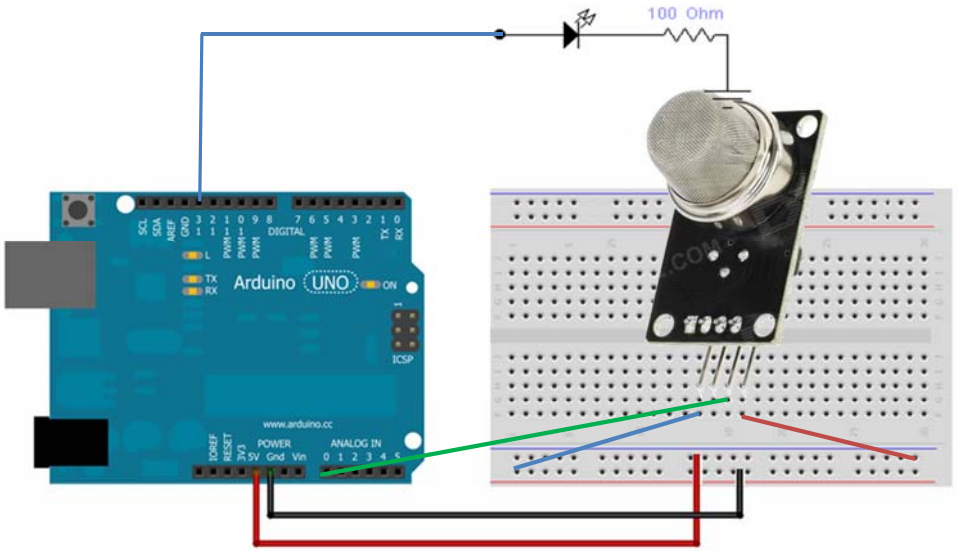

```
#define Alarm 13
#define A0 soundSensor
int val;
int counter=0;
void setup() {
  pinMode(Alarm,OUTPUT);
}

void loop() {
    delay(10);
  val=analogRead(soundSensor);
  // level <60 silent

    if(val>60)
    {
      digitalWrite(Alarm,HIGH);
      counter=100;
    }
```

```
else
    {
     if(counter>0)
       counter--;
     if(counter==0)
       digitalWrite(Alarm,LOW);
    }
}
```


SmokeControlledaAlarm

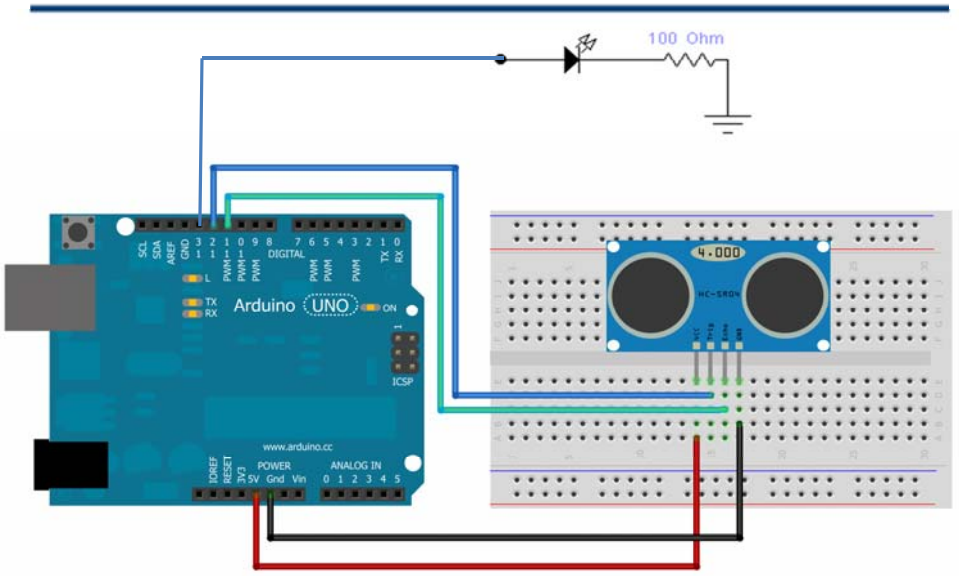


```
#define Alarm 13
#define A0 smokeSensor
int val;

void setup() {
  pinMode(Alarm,OUTPUT);
}

void loop() {
    delay(1000);
  val=analogRead(smokeSensor);
  //smoke val>550
if(val>550)
      digitalWrite(Alarm,HIGH);
     else
      digitalWrite(Alarm,LOW);
}
```


DistanceControlledAlarm



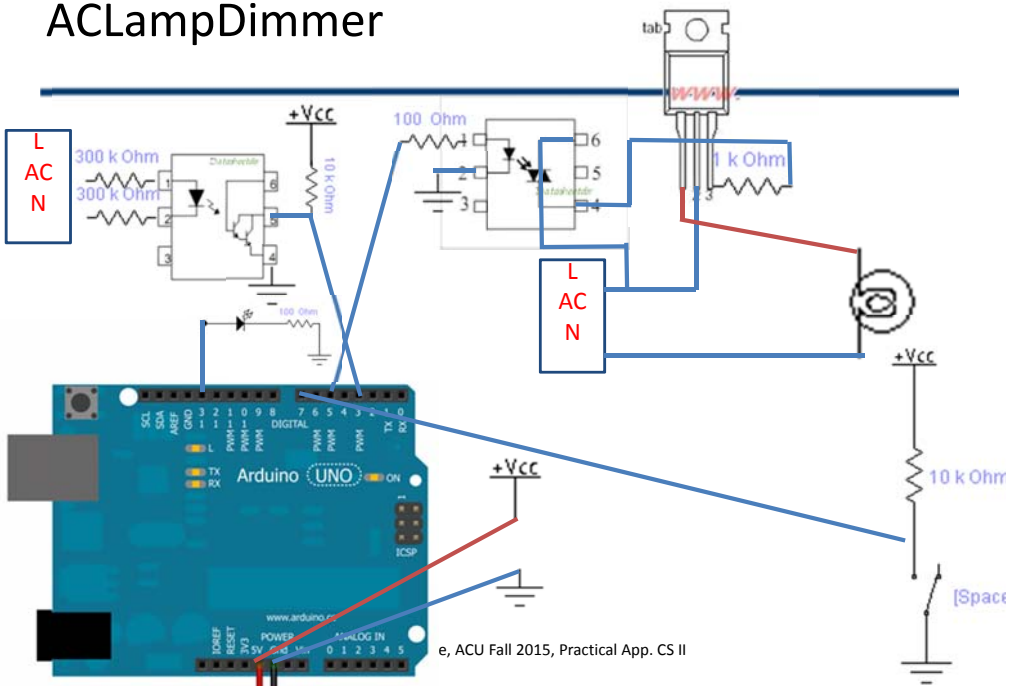

```
#include <NewPing.h>
#define TRIGGER_PIN  12  // Arduino pin tied to trigger pin on the ultrasonic sensor.
#define ECHO_PIN     11  // Arduino pin tied to echo pin on the ultrasonic sensor.
#define MAX_DISTANCE 500// Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-500cm.
#define Alarm 13

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing setup of pins and maximum distance.

void setup() {
  pinMode(Alarm,OUTPUT);
}
```

```
void loop() {
  delay(1000);
  unsigned int uS = sonar.ping(); // Send ping, get ping time in microseconds (uS).
  if(uS<500) //50cm
    digitalWrite(Alarm,HIGH);
  else
    digitalWrite(Alarm,LOW);
}
```


ACLampDimmer



```
#define AC_LOAD 5
int dimming = 128;
#define LED 13
#define BUTTON 7
int LED_status=0;
int count=0;
int Button_status=0;
void setup()
{
pinMode(AC_LOAD, OUTPUT);
attachInterrupt(1, zero_crosss_int,
RISING);
pinMode(LED, OUTPUT);
pinMode(BUTTON,INPUT);
//pin 3 interrupt 1
//pin 2 interrupt 0
}

void loop() {
  for (int i=5; i <= 128; i++)
  {
    dimming=i;
Button_status=digitalRead(BUTTON);
if(Button_status==LOW)
    {
      digitalWrite(LED,HIGH);
      digitalWrite(AC_LOAD,HIGH);
    }
    delay(100);
  }

}

void zero_crosss_int() // function to be
fired at the zero crossing to dim the
light
{
// Firing angle calculation : 1 full 50Hz
wave =1/50=20ms // Every
zerocrossing thus: (50Hz)-> 10ms
(1/2 Cycle) For 60Hz => 8.33ms //
10ms=10000us
// (10000us - 10us) / 128 = 75
(Approx) For 60Hz =>65
int dimtime = (75*dimming); // For
60Hz =>65
delayMicroseconds(dimtime); // Off cycle
digitalWrite(AC_LOAD, HIGH); // triac firing
delayMicroseconds(10); // triac On
propogation delay (for 60Hz use 8.33)
digitalWrite(AC_LOAD, LOW); // triac Off
count++;
if(count==50)
{
  count=0;
  if(LED_status==LOW)LED_status=HIGH;
  else LED_status=LOW;
  digitalWrite(LED,LED_status);
}
}
```


Thanks,

See you next Week, isA