EP375 Computational Physics

Topic 12

SOUND PROCESSING

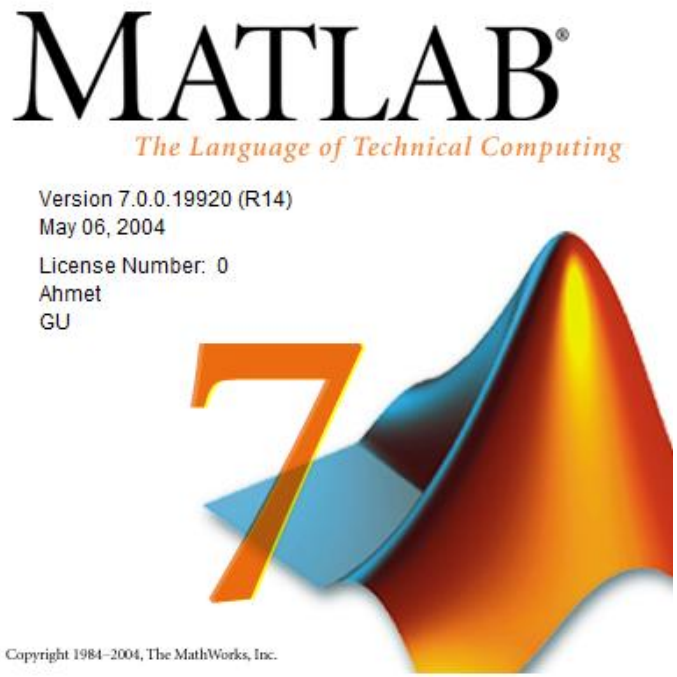
Department of Engineering Physics

Gaziantep University

Apr 2016

Content

1. Introduction
2. Sound
3. Perception of Sound
4. Physics of Sound
5. PC Sound Cards
6. MATLAB Sound Functions
7. Example Applications



1. Introduction

- Sound is a sequence of waves of pressure that propagates through compressible media.

- Sound processing (audio signal processing) is the intentional alteration of sound (auditory signals).
  ==> Audio signals may be electronically represented in either digital or analog format.

- In this section we will consider
  how to read / play / plot / manipulate audio signals in MATLAB.

2. Sound

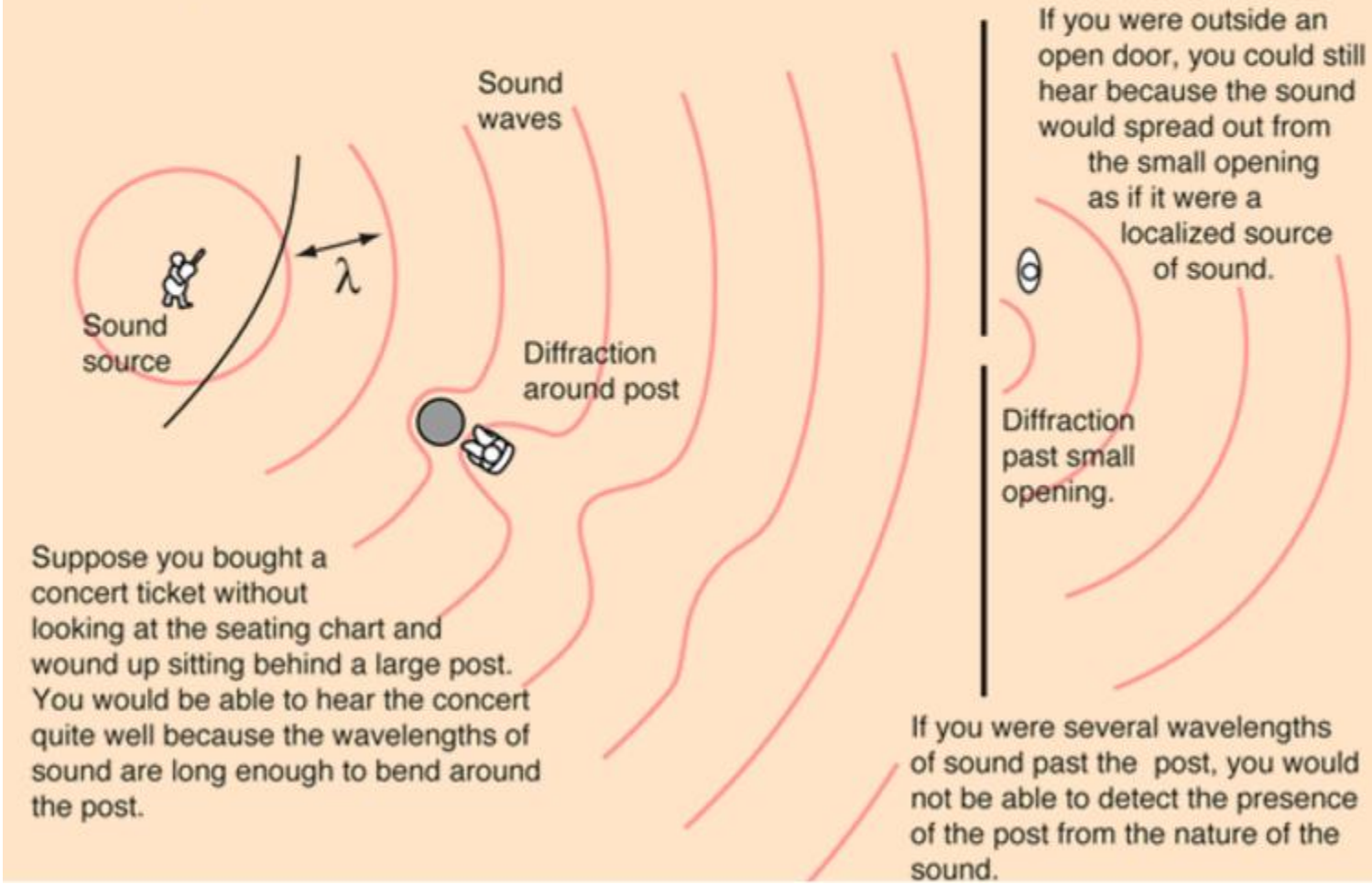
- **Sound** is a mechanical wave that is an oscillation
  - of pressure transmitted through a solid, liquid, or gas, composed of frequencies within the range of hearing and
  - of a level sufficiently strong to be heard.

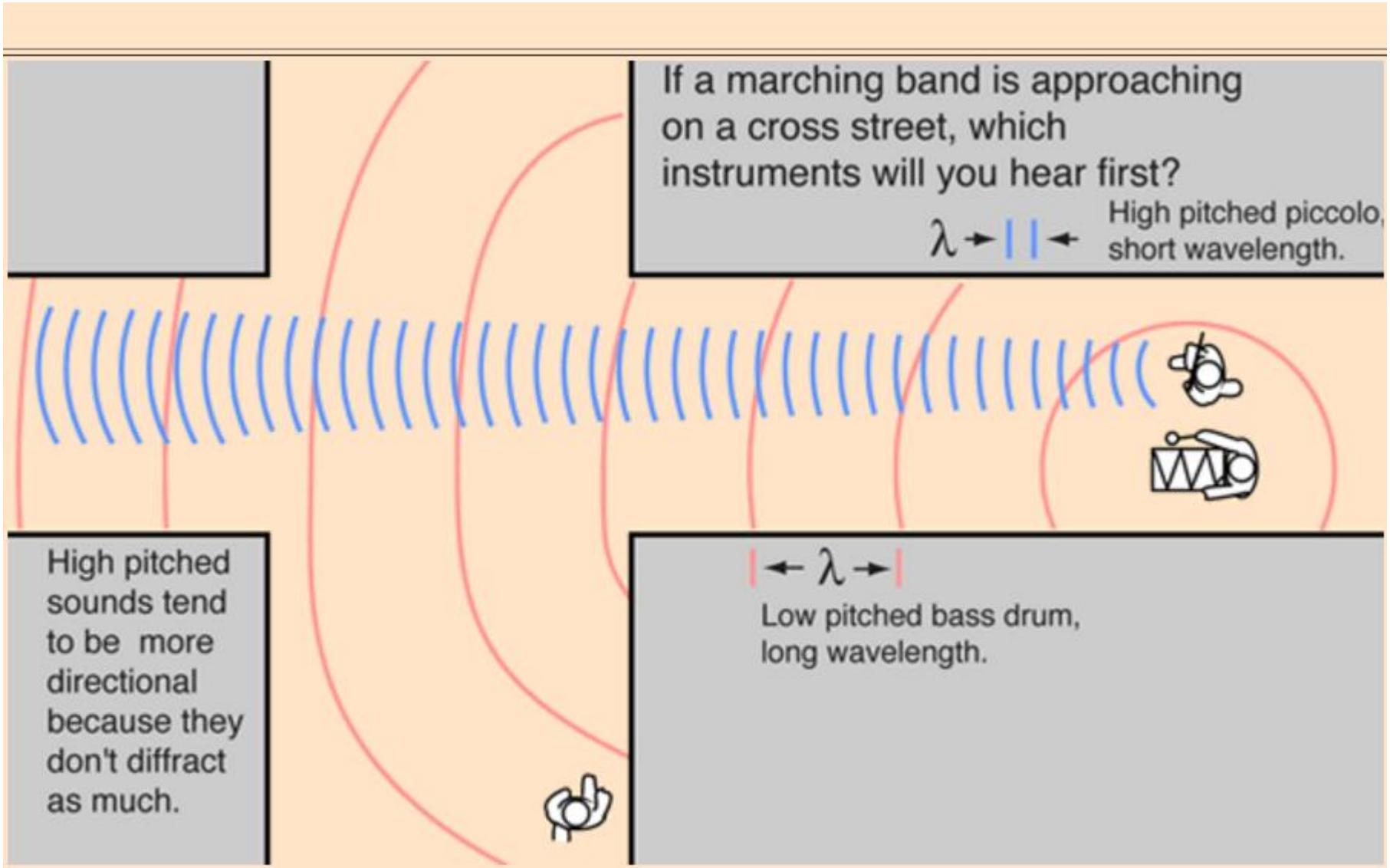
- Sound waves can be reflected, refracted, diffracted, interfered or attenuated by the medium.

Diffraction of Sound

Diffraction: the bending of waves around small* obstacles and the spreading out of waves beyond small* openings.

* small compared to the wavelength





The long wavelength sounds of the bass drum will diffract around the corner more efficiently than the more directional, short wavelength sounds of the higher pitched instruments.

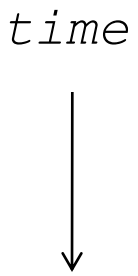
3. Perception of Sound

The perception of sound in any organism is limited to a certain range of frequencies.

- For humans, hearing is normally limited to frequencies between about 20 Hz - 20 kHz.
- The upper limit generally decreases with age.
- Dogs can perceive vibrations higher than 20 kHz, but are deaf to anything below 40 Hz!

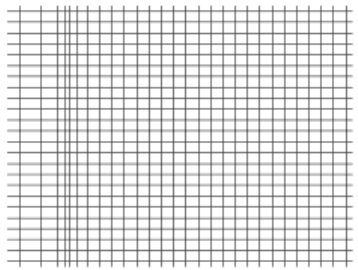
4. Physics of Sound

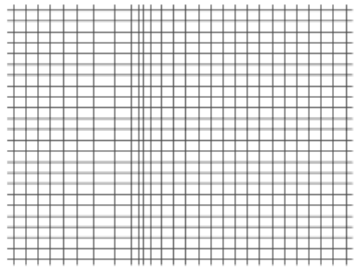
- Sound is transmitted through gases, plasma, and liquids as **longitudinal waves**, also called compression waves.

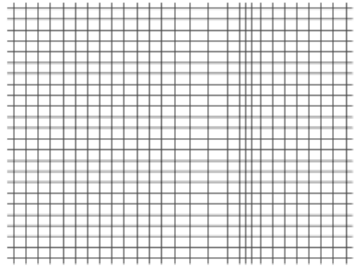


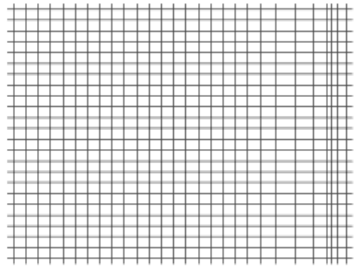
- Through solids, however, it can be transmitted as both longitudinal waves and transverse waves.

- The matter that supports the sound is called the **medium**.
  Sound cannot travel through a vacuum.









- **The speed of sound** depends on the medium the waves pass through.

In general, the speed of sound (v) is given by the Newton-Laplace equation:

$$v = \sqrt{\frac{K}{\rho}}$$

K is a coefficient of stiffness, the bulk modulus*

ρ is the density of the medium.

* *Stiffness is the resistance of an elastic body to deformation by an applied force*.

- **The speed of sound** depends on the temperature.

$$v_{AIR} = 331.3\sqrt{1+\frac{T}{273.15}} \ (m/s)$$

where T is temperature in degrees celcius

v = 343 m/s in dry air at 20 ^{o}C.

v = 1482 m/s in fresh water at 20 ^{o}C

v = 5960 m/s in steel

Effect of temperature on properties of air

Temperature T in °C	Speed of sound c in $m \cdot s^{-1}$	Density of air ρ in $kg \cdot m^{-3}$	Acoustic impedance Z in $N \cdot s \cdot m^{-3}$
+35	351.88	1.1455	403.2
+30	349.02	1.1644	406.5
+25	346.13	1.1839	409.4
+20	343.21	1.2041	413.3
+15	340.27	1.2250	416.9
+10	337.31	1.2466	420.5
+5	334.32	1.2690	424.3
0	331.30	1.2922	428.0
−5	328.25	1.3163	432.1
−10	325.18	1.3413	436.1
−15	322.07	1.3673	440.3
−20	318.94	1.3943	444.6
−25	315.77	1.4224	449.1

Here Z indicates how much sound pressure is generated by the vibration of molecules of a particular acoustic medium at a given frequency.

Acoustic Impedance is pressure per velocity per area.

$$Z = P / (v*A)$$

5. PC Sound Cards

- A PC Sound Card is a computer hardware device that allow you to hear sounds and music through your speakers and provide the input for microphones.

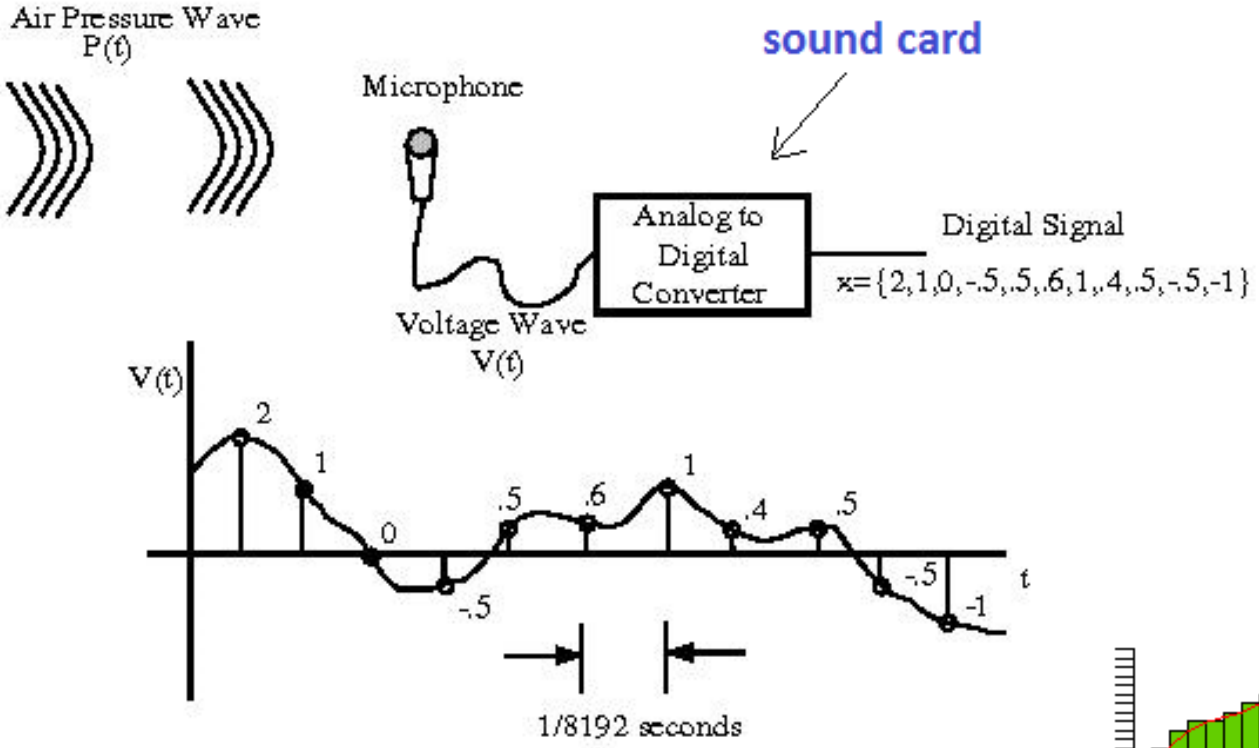
- Before the invention of the sound card, a PC could make one sound - a beep.

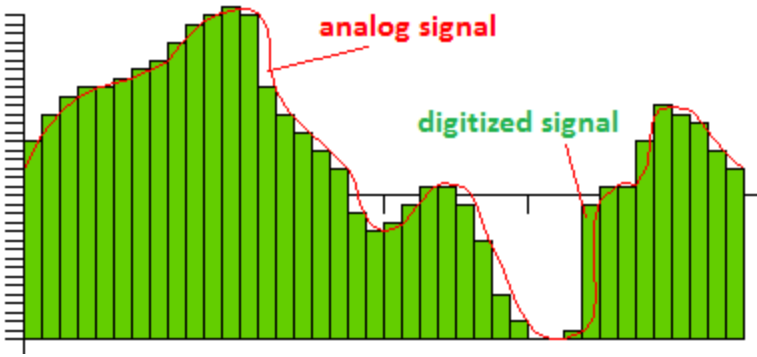
- Nowadays, sound cards include providing the audio component for multimedia applications:
  * music composition,
  * editing video or audio,
  * games etc.

Microphone (input)

Speaker (output)

- Sound card is both digital-to-analog converter (DAC) and analog-to-digital converter (ADC).

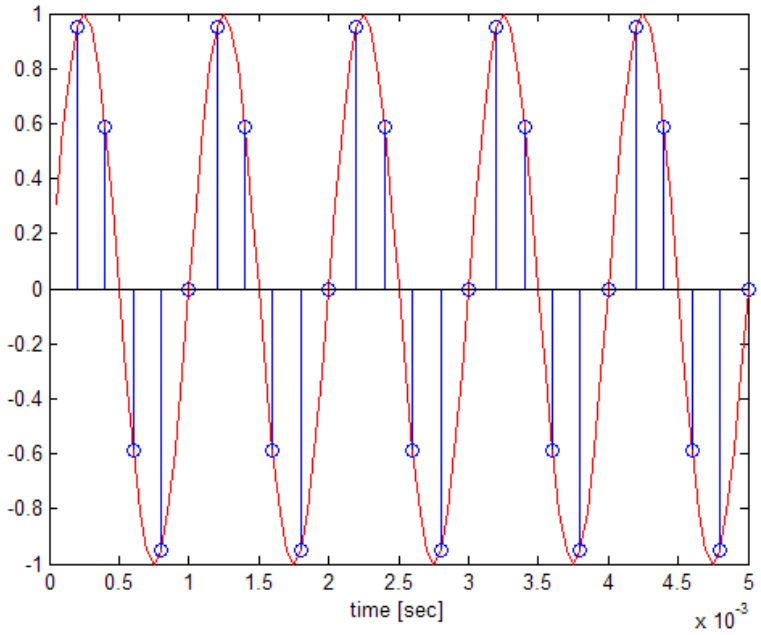




Important properties:

Sampling

- Takes a snapshot of the microphone (sensor) signal at discrete times.
- Read values from a continuous signal equally spaced time interval.



Bit Per Sample

- Specify the number of bits the sound card uses to represent each sample.
- It can be 8, 16, or any value between 17 and 32.
- *If BitsPerSample = 8, the sound card represents each sample with 8 bits. i.e: each sample is represented by a number between 0 and 255.*
- *If BitsPerSample =16, the sound card represents each sample with 16 bits i.e: each sample is represented by a number between 0 and 65535.*

Sampling Rate

- is the number of samples made per unit of time (usually expresses as samples per second or Hertz).
- A low sampling rate will provide a less accurate representation of the analog signal. Sample size is the range of values used to represent each sample, usually expressed in bits. The larger the sample size, the more accurate the digitized signal will be.
- Sound cards commonly use 16 bit samples at sampling rates from about 4000 Hz to 44,000 Hz samples per second.
- The samples may also be contain one channel (mono) or two (stereo).

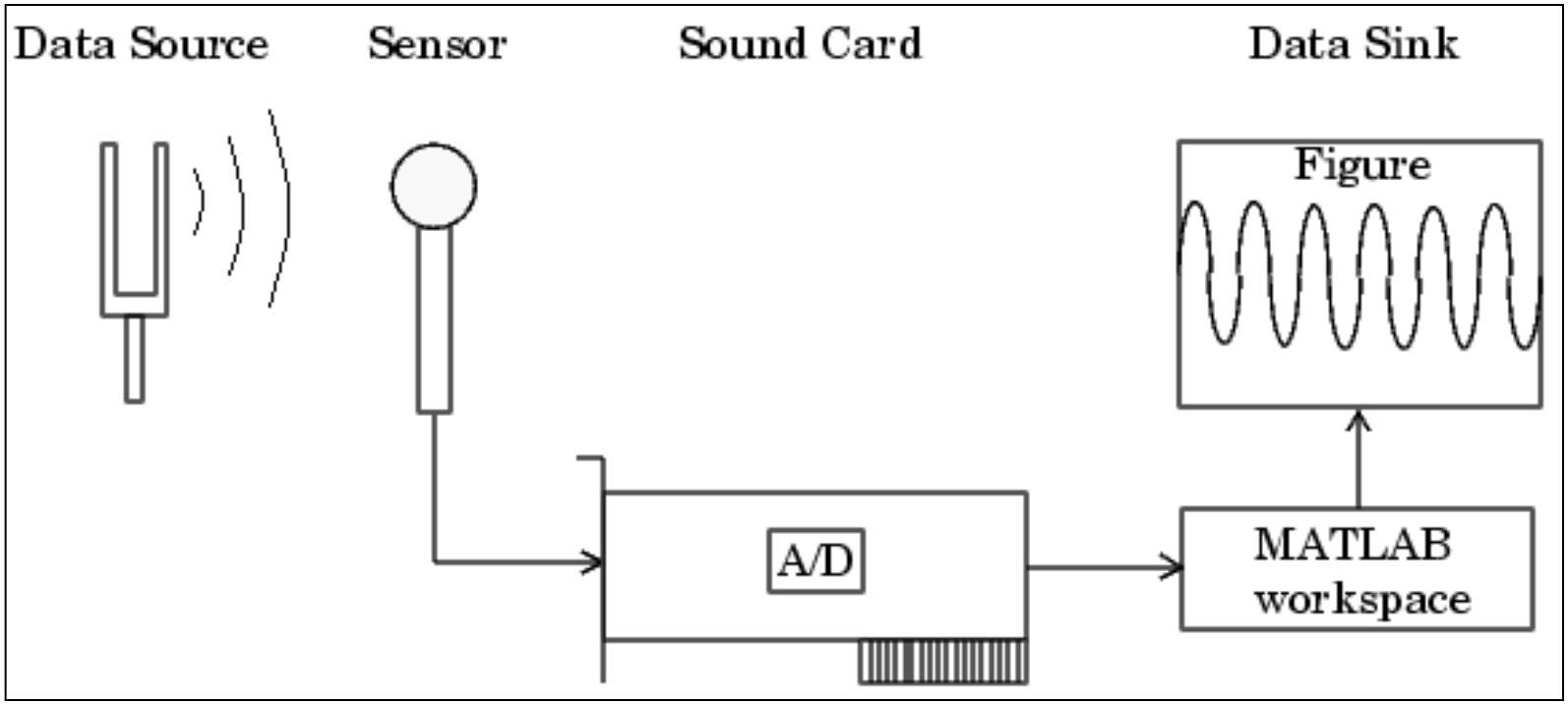
6. MATLAB Functions

- MATLAB provides some functions to process audio signals.

OLD*	NEW
wavread()	audioread()
wavrecord()	audiorecorder()
wavwrite()	audiowrite()
wavplay()	audioplay()
	audioinfo()
	sound()

- *These functions will be removed in a future releases of MATLAB but they are still available in Octave. See backup slides for examples.*

Acquiring Data with a Sound Card



Example Applications [sound generation]

```
% sg1.m
% random noise in the range (0,1)
y = -1 + rand(1,100000);
sound(y,fs)
wavwrite(y)
```

```
% sg2.m
% play a specific sound frequency (f)
clear; clc;
fs = 44100;   % sampling rate
f  = 1000;    % frequency
dt = 1/fs;    % time steps
t = 0:dt:1;   % time
y  = sin(2*pi*f*t); % sine profile
sound(y,fs)         % play the function
```

```
% sg3.m
% play signal and noise
clear; clc;
fs = 44100;
f  = 1000;
dt = 1/fs;
t = 0:dt:1;
y  = sin(2*pi*f*t) + rand(1,length(t))-1;
sound(y,fs)

% write the sound to a file
audiowrite('signal_and_noise.wav', y, fs)
```

- When two sound waves having slightly different frequencies interfere we hear variations in the loudness called beats.

```
% sg4.m
% Beat_frequency = |f1-f2|
clear; clc;
fs = 44100;
f1 = 500;
f2 = 502;
dt = 1/fs;
t  = 0:dt:3;
y  =  sin(2*pi*f1*t) + sin(2*pi*f2*t);% sine profile
plot(t,y)
sound(y,fs)
```

```
% sg5.m
% play left and right
clear; clc;
fs = 44100;
f1 = 500;
f2 = 510;
dt = 1/fs;
t  = 0:dt:3;
y1 = sin(2*pi*f1*t); % left
y2 = sin(2*pi*f2*t); % right
y  = [y1' y2'];
sound(y,fs)
```

```
% sg6.m
% pulse generator
clear; clc;

...
```

Example Applications [accessing sound files]

The following function call assumes that the file '**hucum.wav**' İs in a location in the Matlab path.

```
>> audioinfo('hucum.wav');

Filename: 'C:\Users\Ahmet\Desktop\MATLAB\hucum.wav'
    CompressionMethod: 'Uncompressed'
          NumChannels: 2
           SampleRate: 44100
         TotalSamples: 853776
             Duration: 19.3600
                Title: []
              Comment: []
               Artist: []
        BitsPerSample: 16
```

Reading and playing the sound file

```
>> [y, fs] = audioread('hucum.wav');
>> disp(fs)
44100

>> sound(y,fs)     % play at orig. sampleRate
>> sound(y,fs/2)   % play at half of orig. sampleRate
```

Ploting the sound file

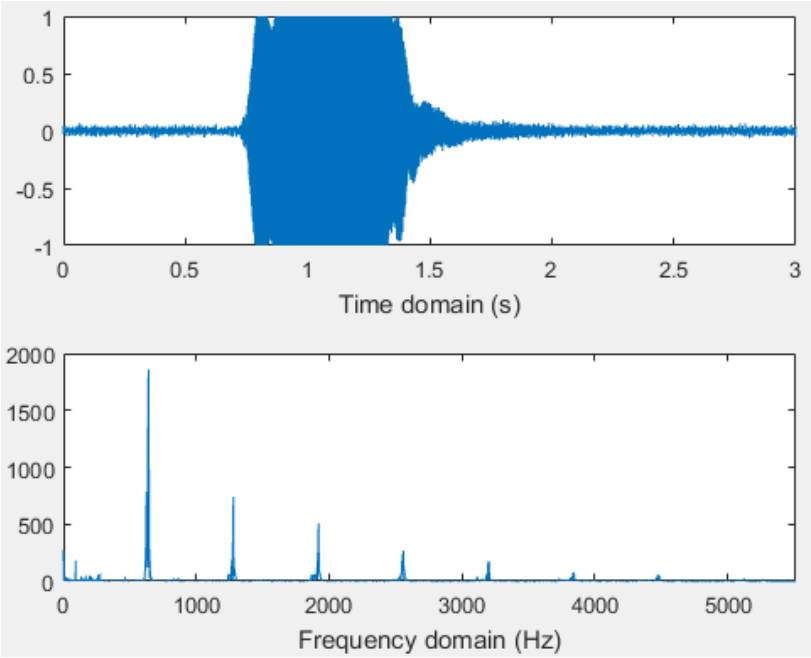
```
>> [y, fs] = wavread('hucum.wav');
>> size(y)              % size of data matrix
ans =   2               % 1: mono, 2: stereo
>> left  = y(:,1);      % left channel signal
>> right = y(:,2);      % left channel signal

>> tmax = length(left)/fs;       % plot entire data
>> t = linspace(0, tmax, length(left));
>> plot(t, left);

>> time = 3000/fs;          % plot a portion of data
>> t = linspace(0, tmax, 3000);
>> plot(t,left(1:3000))
```

Example Applications [Recording sound]

```
% sr1.m
% Record your voice for 3 seconds.
clear; clc;
ar = audiorecorder;
disp('Start speaking...')
recordblocking(ar, 3);
disp('End of recording.');

% Play back the recording.
play(ar);

% Store data in double-precision array.
y = getaudiodata(ar);

% Plot the waveform.
plot(y);
```

```
% sr2.m
% Record your voice and plot it forever
clear; clc;
Fs = 11025; % sampling rate
T  = 0.1;   % total sampling time

% Start recorder for 16-bit, mono(1) [stero(2)]
ar = audiorecorder(Fs,16,2);

while 1
  recordblocking(ar, T);
  y = getaudiodata(ar);          % Store sound array
  t = linspace(0,T,length(y));   % time array
  p = plot(t,y);                 % Plot the waveform
  axis([0 T -0.5 0.5]);          % setup the axis ranges
  title('time-domain sound data')
  xlabel('time (s)')
  %pause(0.01);
  %delete(p);
end
```

```
% sr3.m
% Record sound data plot time and frequency domains
clear; clc;
Fs = 11025; % sampling rate
T  = 0.1;   % total sampling time
 ar = audiorecorder(Fs,16,1);

while 1
  recordblocking(ar, T);
  y = getaudiodata(ar);
  N = length(y);
  t = linspace(0,T,N);
  subplot(2,1,1)
  p1 = plot(t,y);
  axis([0 T -0.5 0.5]);
  xlabel('time (s)');

  % Get and plot the frequency componets
  Y = fft(y);
  df= Fs / length(y);
  f = (1:length(Y)) * df;
  N = uint32(N/2);
  subplot(2,1,2);
  p2 = plot( f(1:N), abs(Y(1:N)) );
  axis([0 Fs/2 0 2000]);
  xlabel('Frequency domain (Hz)');
end
```

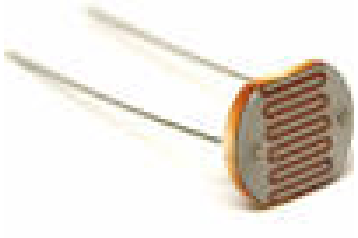


```
% sr4.m
% Record sound data open notepad for a specific frequency
Fs = 11025; % sampling rate
T  = 3;     % total sampling time
ar = audiorecorder(Fs,16,1);
recordblocking(ar, T);
y = getaudiodata(ar);
N = length(y);
t = linspace(0,T,N);
subplot(2,1,1); p1 = plot(t,y); xlabel('Time domain (s)');
axis([0 T -1.0 1.0]);

Y = fft(y);
df= Fs / length(y);
f = (1:length(Y)) * df;
N = uint32(N/2);
subplot(2,1,2); p2 = plot(f(1:N),abs(Y(1:N)));xlabel('Freq. (Hz)');
axis([0 Fs/2 0 2000]);

% get maximum value and its index
[V I] = max(abs(Y(1:N)))
f(I)
if(f(I)>630 & f(I)<650)
   system('start notepad')
end
```

```
% sr5.m
% frequency analysis of specific files
clear; clc; figure;

% Get and play sound data
% you can find the file at:
% http://www1.gantep.edu.tr/~bingul/ep375/wavfiles/
[data Fs] = audioread('instr_piano.wav');
sound(data, Fs)

% Plot sound data
t = (1:length(data)) / Fs;
subplot(1,2,1)
plot(t, data)
xlabel('Time domain (s)')

% Get and plot the frequency componets of the sound data
Y = fft(data);
df = Fs / length(data);
f = (1:length(Y)) * df;
subplot(1,2,2)
plot( f, abs(Y) )
xlabel('Frequency domain (Hz)')
```

Example Application [Use of LDR]

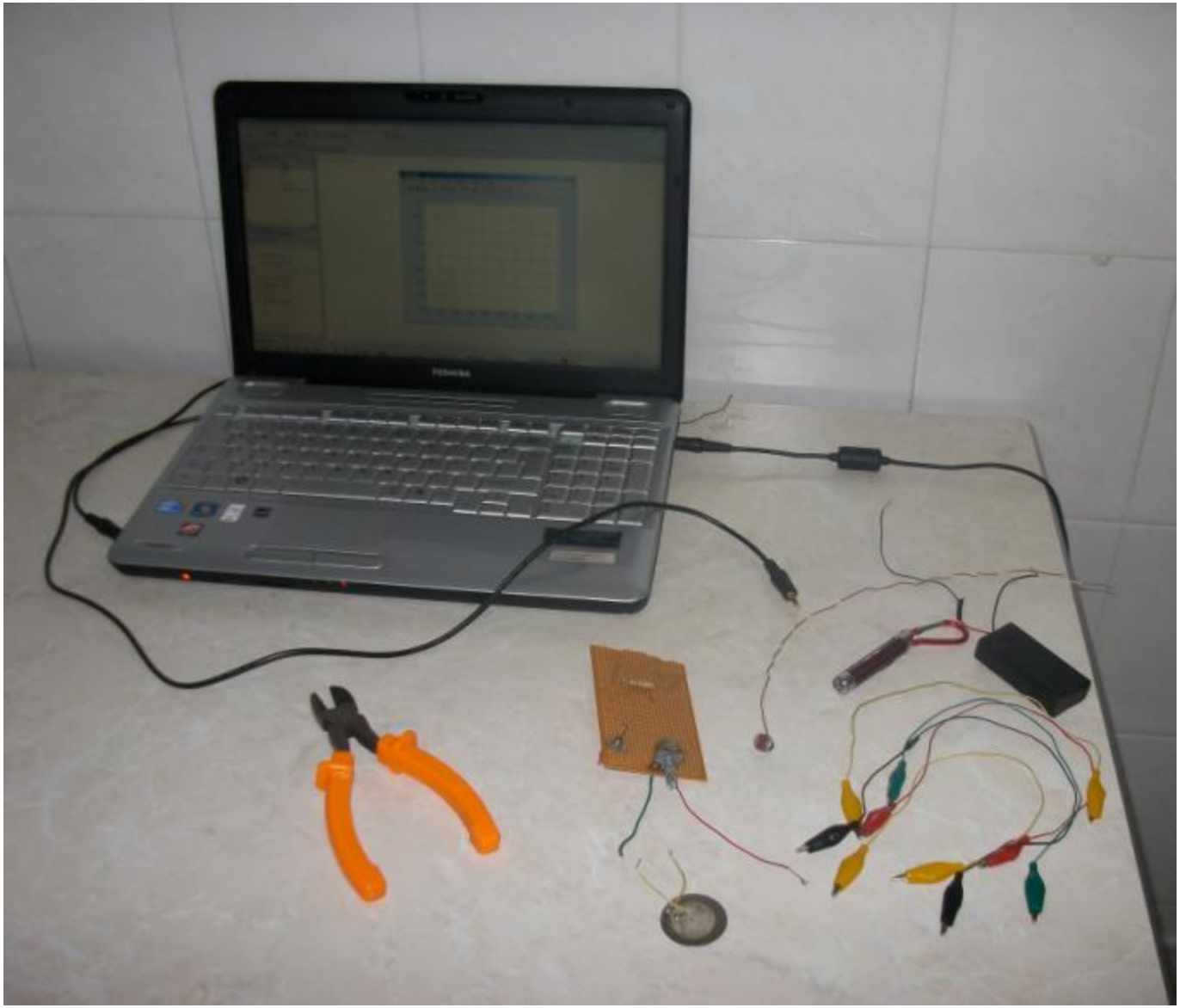
- You can plug in an LDR (Light Dependent Resistor) directly to microphone input.



- This will give you an output as a function of light intensity!

- Finally using the code sr2.m (in the previous slides), you can perform some funny (time and light dependent) projects. (See exercises)

Exercises



HW 1:

Develop a method to measure the speed of sound in dry air.

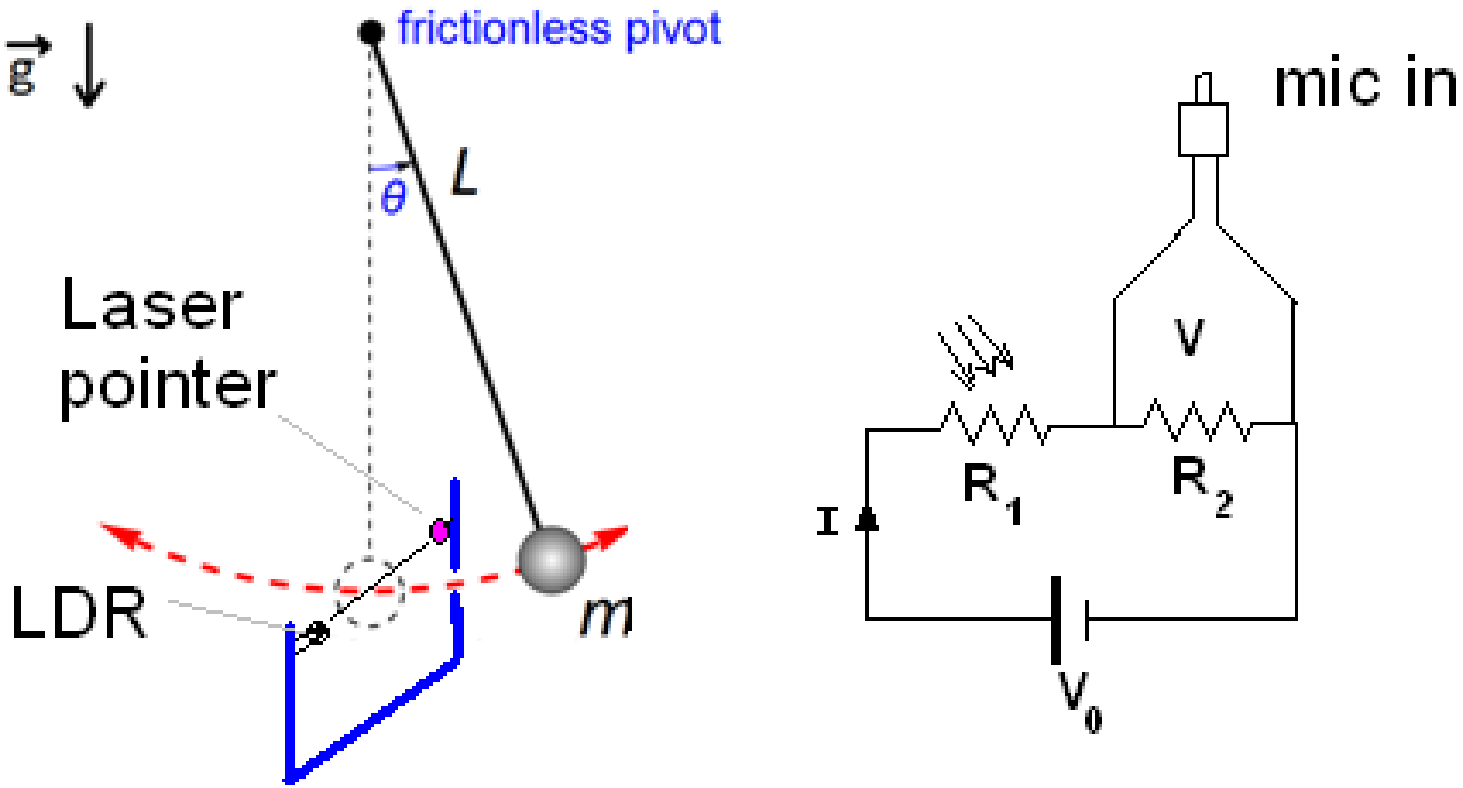
You should implement your method using sound card in a PC.

Write a Matlab m-file for your setup and compare

your measurement with v = 343 m/s.

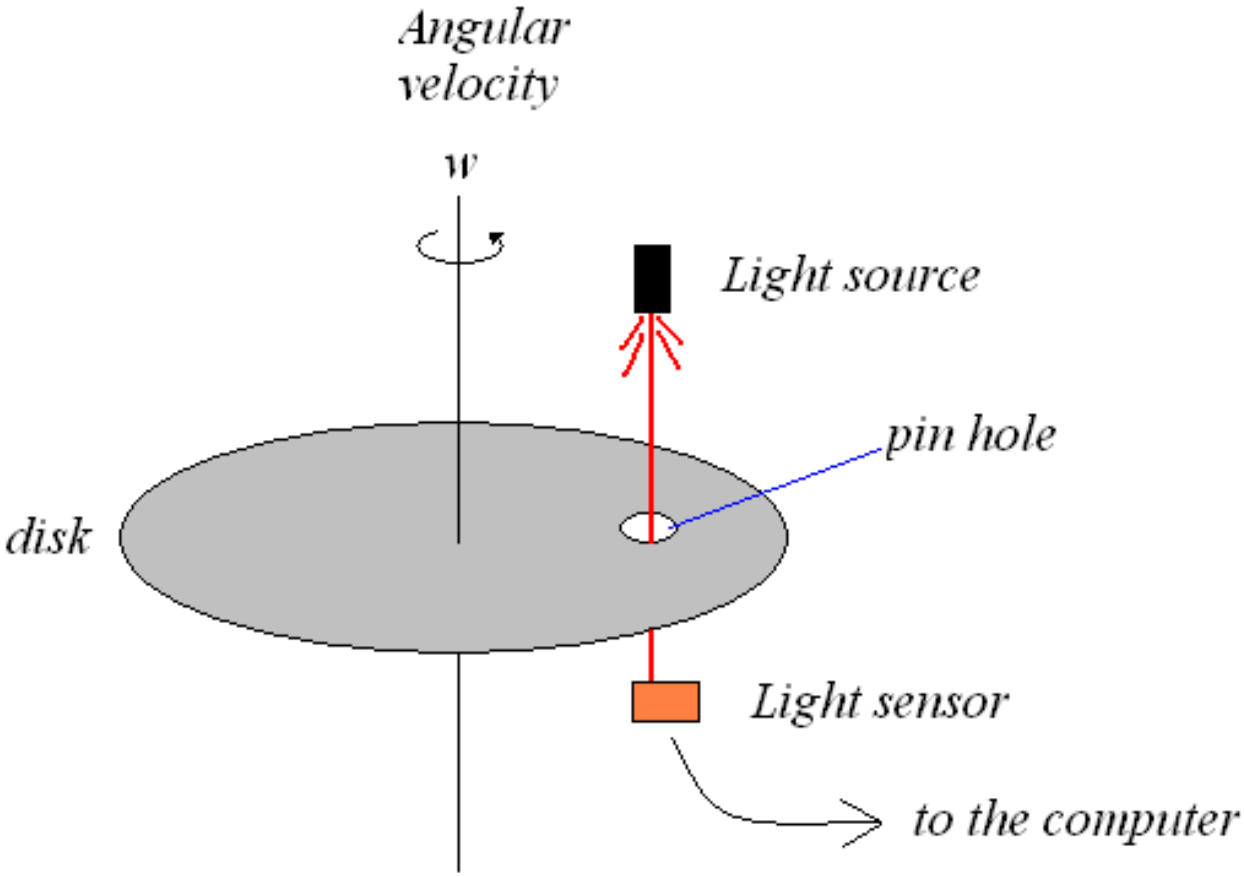
HW 2:

Use the matlab sound card functions to measure the gravitational acceleration using a simple pendulum. Setup is given below. (R1: LDR, R2: a constant resistor and V_0 =1.5 V)



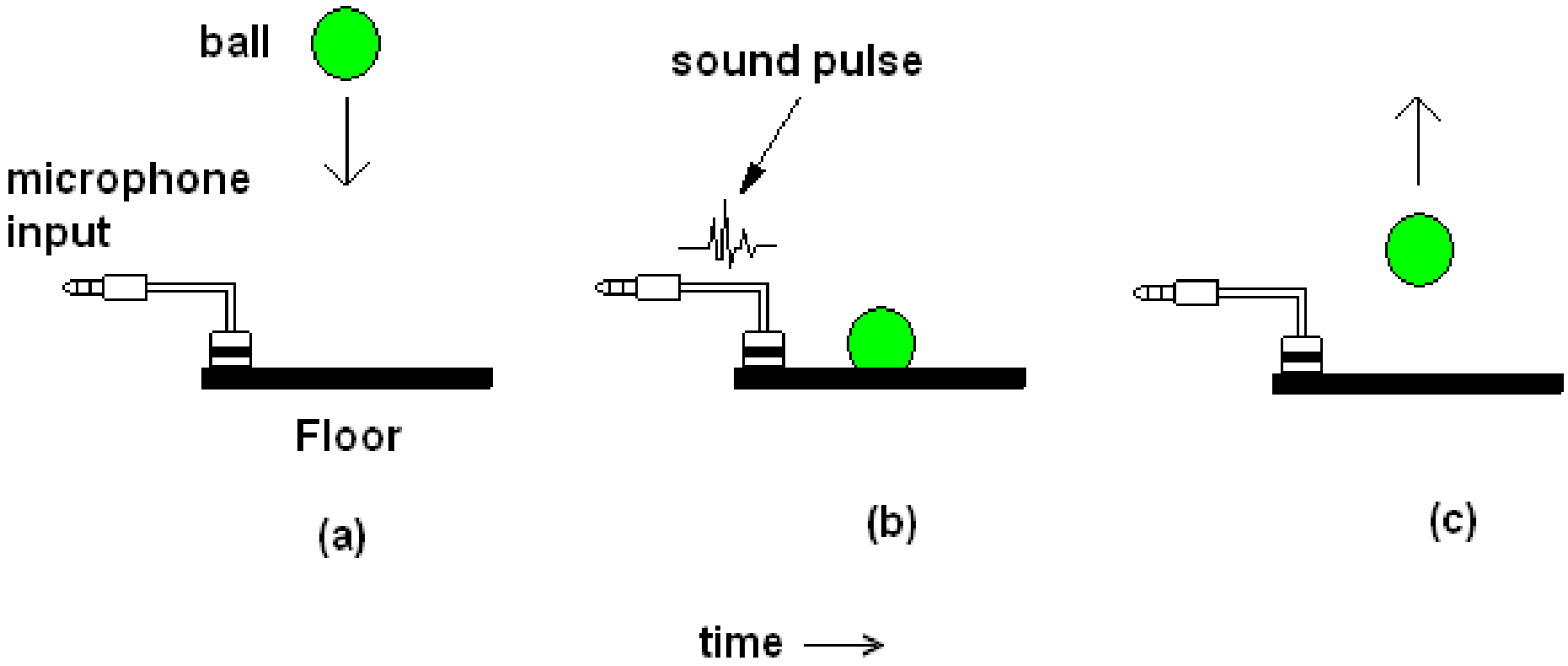
HW 3:

Using sound card functions, write a Matlab script to measure the angular frequency (w) in rad/s and in rpm of a rotating disk. You can use the same circuit in HW2.



HW 4:

Using sound card functions, write a Matlab script to measure the contact time of a tennis ball. Note that, in general, the contact time is in the order of a few milli-seconds. An example setup is given below.



References:

1. http://en.wikipedia.org/wiki/Sound
2. http://en.wikipedia.org/wiki/Sound_card
3. http://tldp.org/HOWTO/Sound-HOWTO/x71.html
4. http://computer.howstuffworks.com/sound-card3.htm
5. http://homepages.udayton.edu/~rhardie1/ECE203/sound.htm
6. http://www.aquaphoenix.com/lecture/matlab10/page4.html
7. http://pcsoundcards.org

BACKUP SLIDES

- Older sound processing functions in MATLAB.
- These are still available in Ocatave 4.0

Example: *Recording and playing back sounds:*

sound_rec.m

```
Fs = 11025; % Set a sampling rate
nbits = 16; % Bit per sample

% Record 3 seconds of 16-bit audio
disp('Recording ...');
y = wavrecord(3*Fs, Fs, 'double');

% Play back the recorded sound
disp('Playing back ...');
wavplay(y, Fs);

% Write it out to a new file.
disp('Writing data to the file ...');
wavwrite(y, Fs, nbits, ...
       'C:\Users\toshiba\Desktop\matlab\newsound.wav');
```

Example: *Ploting dynamic data (from microphone input):*

sound_mic.m

```
clear;
figure;
grid on;
hold on;

Fs = 11025; % sampling rate in Hz
dt = 0.1;   % duration in seconds

while 1
    y = wavrecord(dt*Fs, Fs, 'double');
    p = plot(y);
    axis([0 dt*Fs -0.5 0.5]);
    pause(0.01);
    delete(p);
end
```

Example: *Ploting time & frequency domains*

sound_mic_fft.m

```
clear; figure; grid on; hold on;

Fs = 44100; % sampling rate in Hz
dt = 0.1;   % duration in seconds

while 1
  y = wavrecord(dt*Fs, Fs, 'double'); % Noisy time domain
  Y = fft(y, 512);                    % 512-point fast Fourier trans.
  Pyy = abs(Y).^2;                    % frequency domain
  f = 1000*(0:256)/512;               % Do not draw all data

  subplot(2,1,1);                     % Plot 'time domain' signal
  p1 = plot(y);                       %
  axis([0 dt*Fs -0.1 0.1]);           %

  subplot(2,1,2);                     % Plot 'frequency domain' signal
  p2 = plot(f,Pyy(1:257));            %

  pause(0.01);
  delete(p1);
  delete(p2);
end
```

Example: *Getting peaks (from microphone input):*

sound_peak.m

```
clear; figure; grid on; hold on;
Fs = 4000; % sampling rate in Hz
dt = 5;    % duration in seconds
% get data
y = wavrecord(dt*Fs, Fs, 'double');
% convert to time (sec)
tmax = length(y)/Fs;
t = linspace(0, tmax, dt*Fs);
% plot
plot(t*1000,y);
axis([0 tmax*1000 -0.2 0.2]);
xlabel('time (ms)');
% --- Analysis ---
j = 1;
for i=1:length(y)
  if y(i)>0.15
      pick(j) = 1000*i/Fs;
      fprintf('%3d --> %8.1f ms\n',j, pick(j));
      j=j+1;
  end
end
```