Imperial College London

DE2 Electronics 2

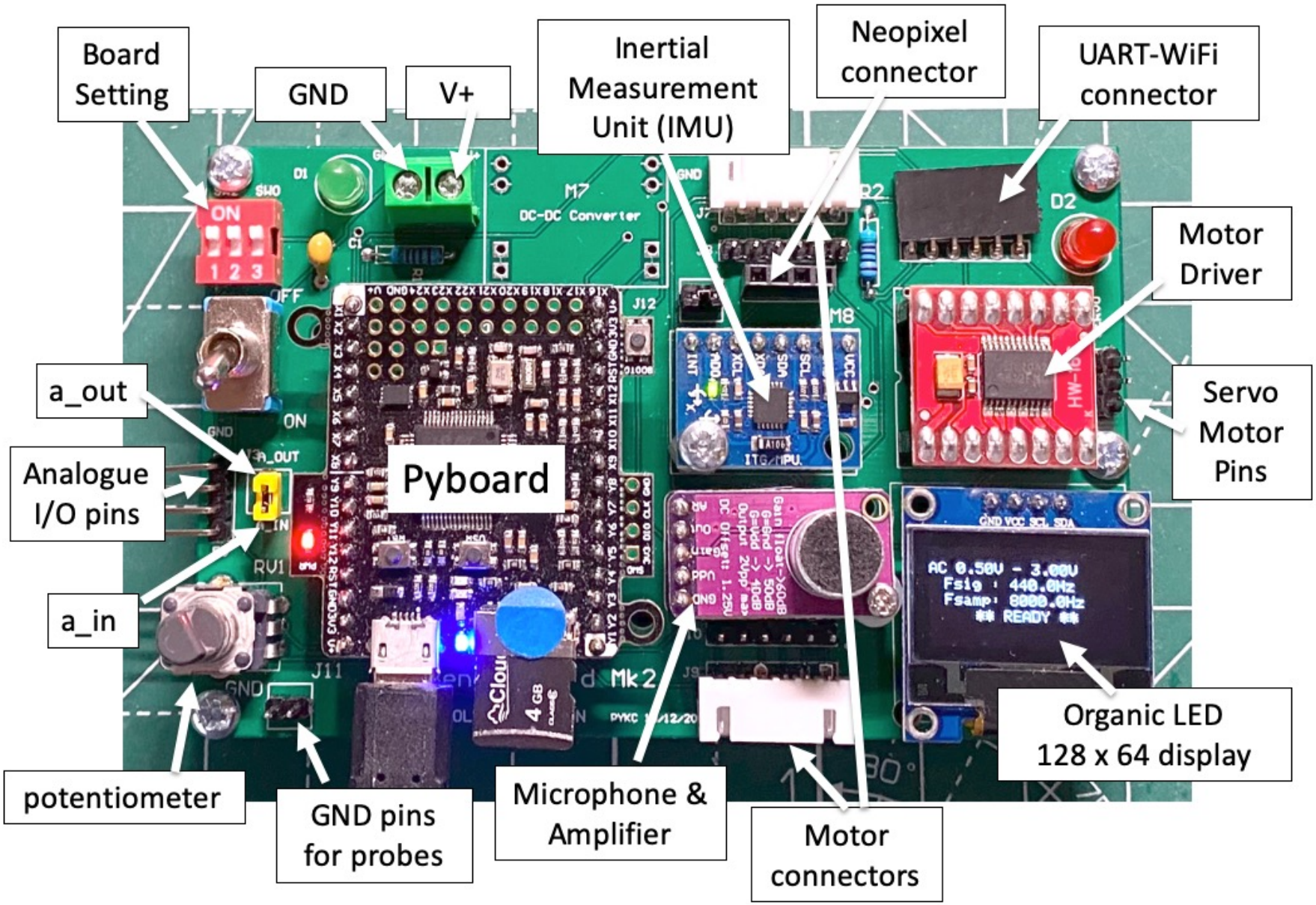
Tutorial 2

PyBench & Lab 2 Explained

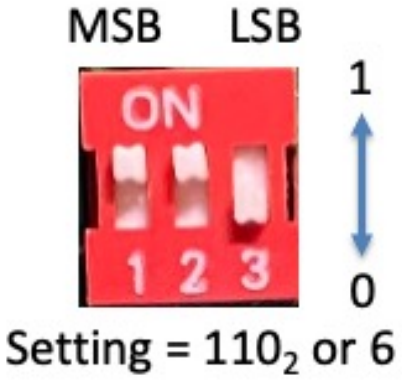
Peter Cheung
Dyson School of Design Engineering

URL: www.ee.ic.ac.uk/pcheung/teaching/DE2_EE/
E-mail: p.cheung@imperial.ac.uk

What's on the Board



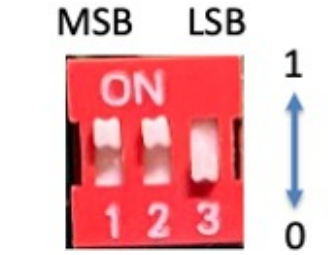
Board setting



Setting = 110_2 or 6

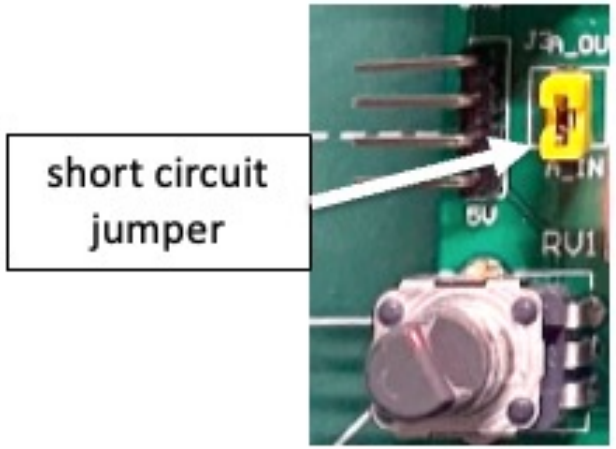
SW[2:0]	Function
000	Run user.py
001	Not used
010	Not used
011	Wifi module Test
100	Spectrum of mic signal
101	Bulb board test
110	Pybench board Test
111	Run pybench.py

Self-test – Verify PyBench system works

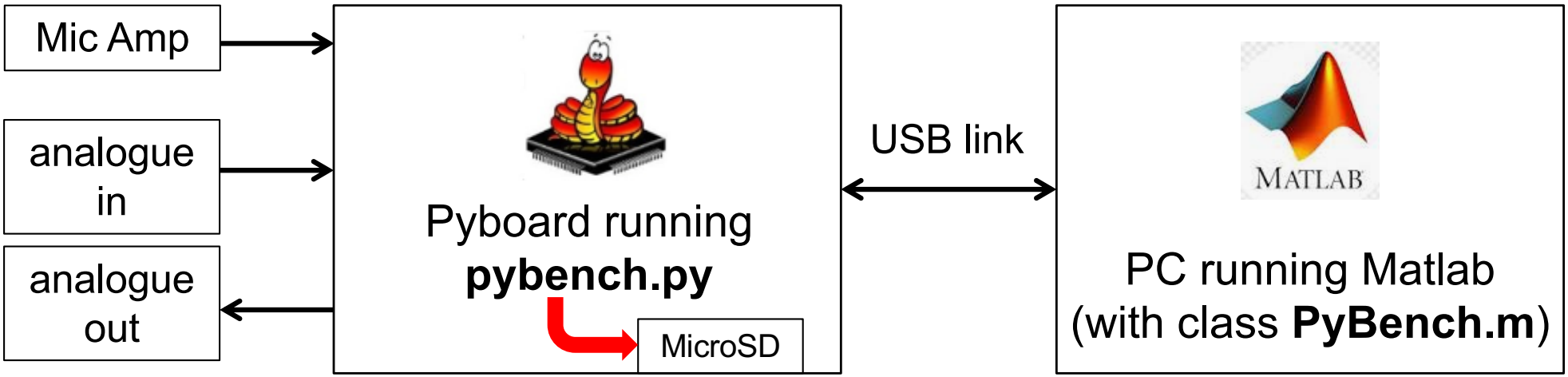


Setting = 110_2 or 6

SW[2:0]	Function
000	Run user.py
001	Not used
010	Not used
011	Wifi module Test
100	Spectrum of mic signal
101	Bulb board test
110	Pybench board Test
111	Run pybench.py



How PyBench works?



- Look for a serial link on computer:

```
ports = serialportlist;      % find all serial port
```

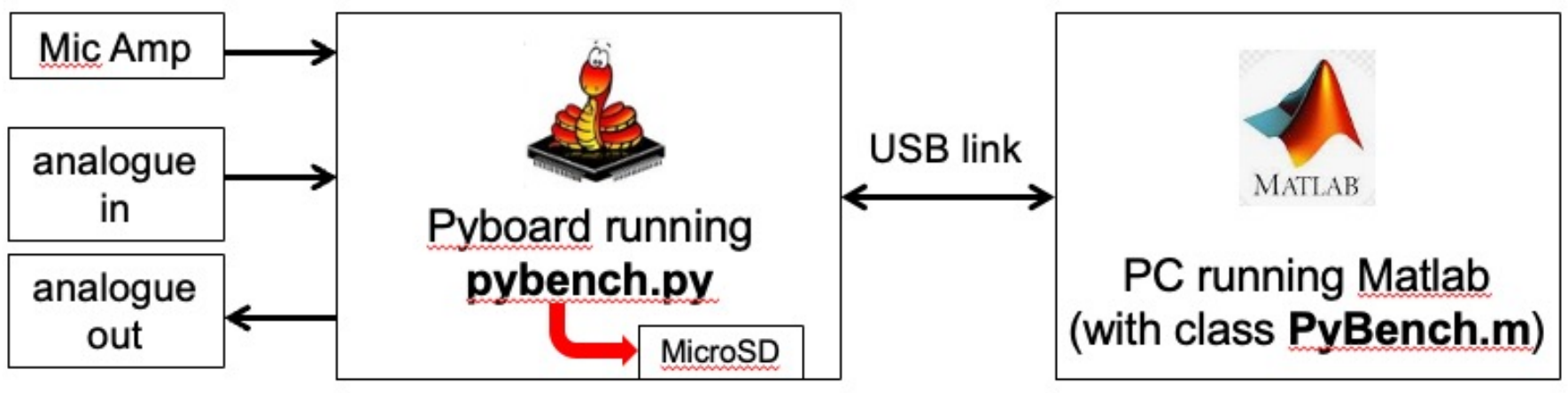
- Last one is usually the one we want to use. The last port is given by: **ports(end).**

- Create an object **pb** for the PyBench Board:

```
pb = PyBench(ports(end));    % create a PyBench object
```

- Control the Board via “methods”, e.g. **pb.set_max_v (2.5).**

pb.set_max_v(2.5) explained

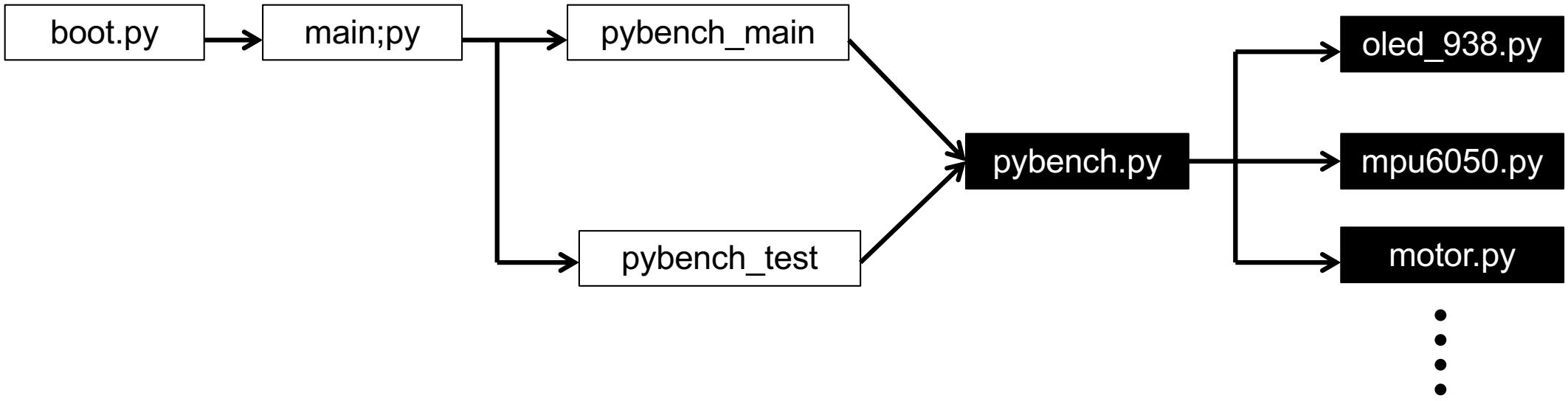


Here is what happens when you used this Matlab command: **pb = pb.set_max_v (2.5).**

- PC sends three bytes to PyBench board via USB link as serial data. First byte is a command character. In this case, 'X', followed by the value of voltage as two bytes. First byte is int (4096 * (v/3.3) / 256), and second byte is int (4096 * (v/3.3)) mod 256.
- All along, PyBoard is running a Python program (pybench.py) listening for a command. The BLUE LED is ON in this state. Waiting for an event such as a character to arrive is known as "**polling**".
- When it receives the command (3 bytes), the pybench.py code sets the maximum voltage of the ADC to 2.5V.

What are stored in the MicroSD card?

Program	Purpose
boot.py	Boot file specifying which is the main program.
main.py	Test the DIP switch setting and execute the corresponding .py file.
pybench_main.py	The controlling program for pybench to interprete commands. Run if SW = 00.
pybench.py	The pybench class library. Can be used in your own application program later.
pybech_test.py	Self-test program for the pybench board to verify the hardware. Run if SW = 11.
oled_938.py	OLED display driver class library.
font.py	Character fonts used by oled_938.py.
mpu6050.py	IMU driver class library – to communicate with the accelerometer and gyroscope.
drive.py	Drive class for the motor driver chip TB6612.



PyBench Methods

PyBench.m must be in the Matlab search path.

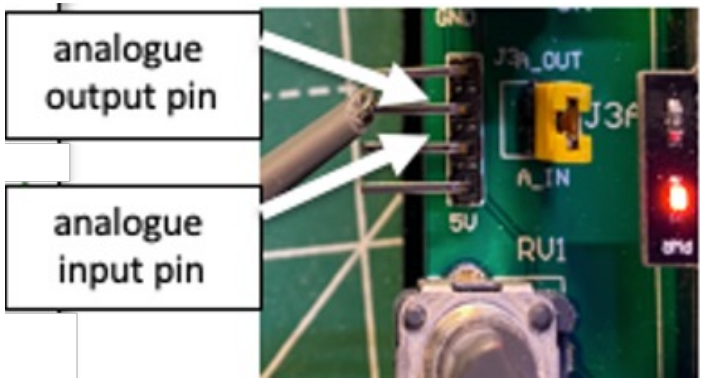
```
clear all
ports = serialportlist;      % find all serial port
pb = PyBench(ports(end));    % create a PyBench object with last port
```

Methods	Purpose
pb.set_sig_freq (f)	Set signal frequency to f. 0.1 Hz ≤ f ≤ 3000 Hz
pb.set_samp_freq (f)	Set sampling frequency to f. 1 Hz ≤ f ≤ 30,000 Hz
pb.set_max_v (v)	Set maximum amplitude to v. 0 ≤ v ≤ 3.3
pb.set_min_v (v)	Set minimum amplitude to v. 0 ≤ v ≤ 3.3
pb.set_duty_cycle (d)	Set duty cycle of a square signal to d. 0 ≤ d ≤ 100
pb.dc (v)	Output a dc voltage v. 0 ≤ v ≤ 3.3
pb.sine ( )	Output a sinusoidal signal at set signal frequency between max_v and min_v.
pb.triangle ( )	Output a triangular signal at set signal frequency between max_v and min_v.
pb.square ( )	Output a square signal at set signal frequency between max_v and min_v, with the set duty cycle.
v = pb.get_one ()	Capture one sample v from analogue input. 0 ≤ v ≤ 3.3
data = pb.get_block (n)	Capture n samples from analogue input. 0 ≤ data ≤ 3.3
data = pb.get_mic (n)	Capture n samples from microphone. 0 ≤ data ≤ 3.3

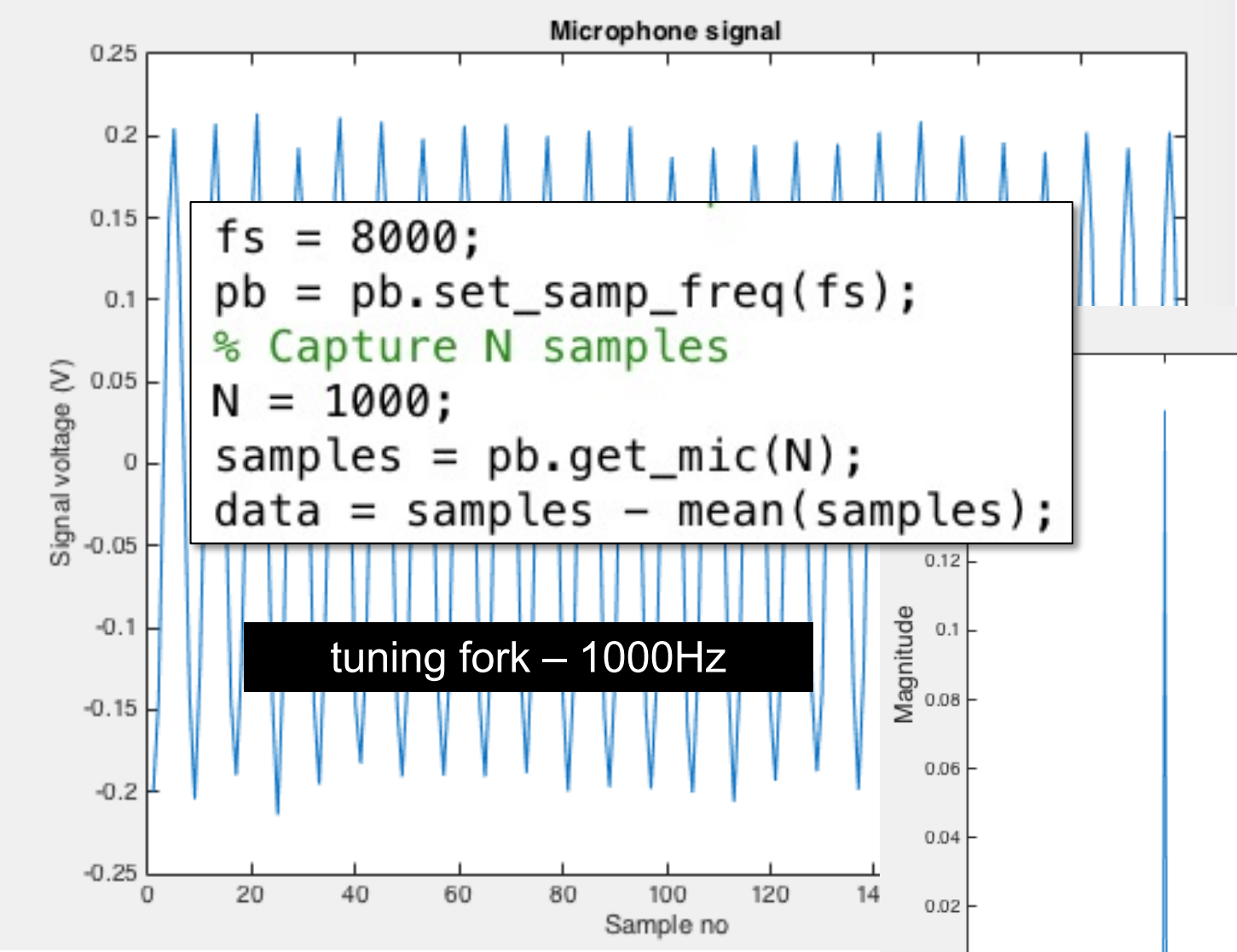
Lab 2 Task 2 – Generate and Capture Signals

```
% Lab 2 - Task 2 - Signal generation and capture with PyBench
%
clear all
ports = serialportlist;      % find all s
pb = PyBench(ports(end));    % create a P
% Set the various parameters
f = 440;                     % signal fre
fs = 8000;                   % sampling f
pb = pb.set_sig_freq(f);
pb = pb.set_samp_freq(fs);
pb = pb.set_max_v(3.0);      % set maximu
pb = pb.set_min_v(0.5);      % set minimu
pb = pb.set_duty_cycle(50);
% Generate a signal
pb.sine();
```

```
% Capture N samples
N = 1000;
samples = pb.get_block(N);
data = samples - mean(samples);
% plot data
figure(1);
plot(data(1:200),'o');
hold on
plot(data(1:200));
xlabel('Sample no');
ylabel('Signal voltage (V)');
title('Captured signal');
hold off
% find spectrum
figure(2);
plot_spec(data,fs);
```

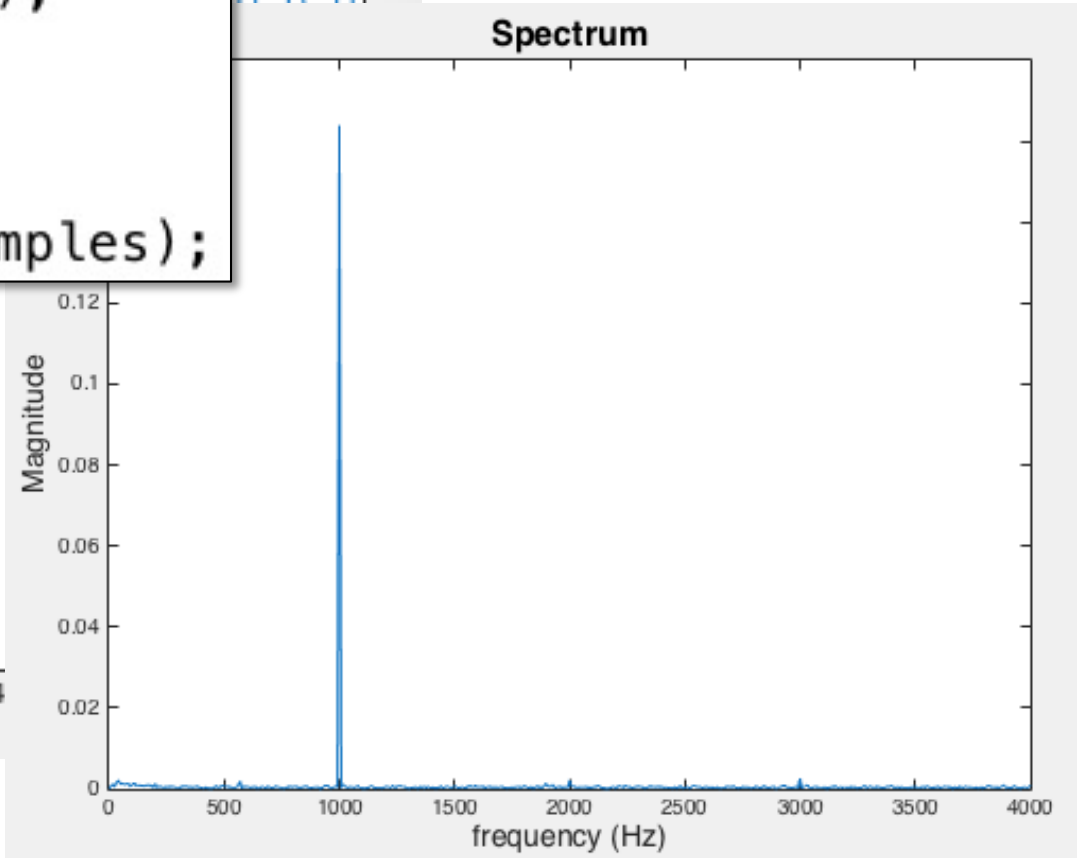


Lab 2 Task 3 – Microphone signal




```
fs = 8000;
pb = pb.set_samp_freq(fs);
% Capture N samples
N = 1000;
samples = pb.get_mic(N);
data = samples - mean(samples);
```

tuning fork – 1000Hz



Lab 2 Task 3 – Repeated capture & plot spectrum

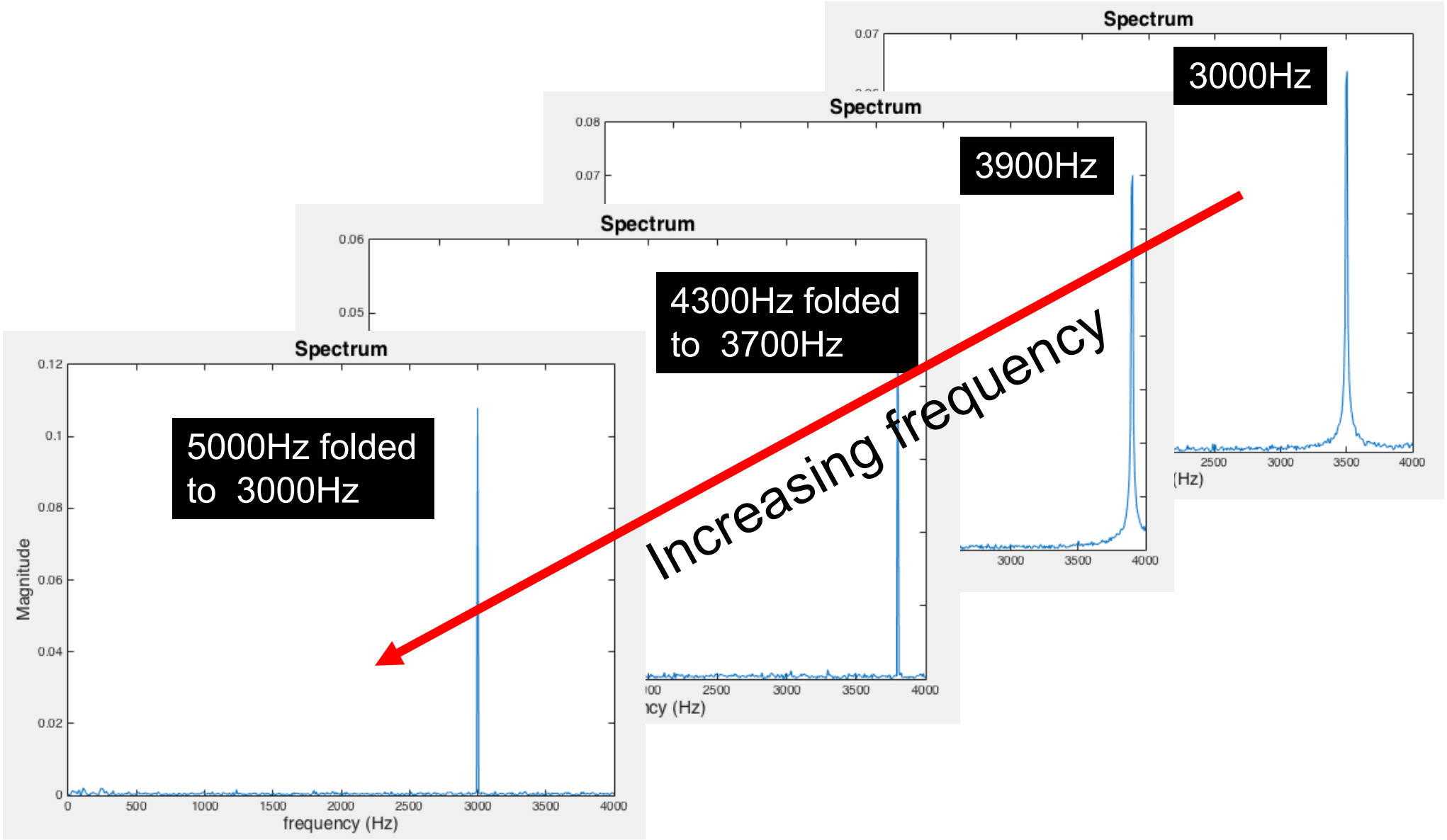
```
% repeat capture and plot spectrum
while true
    samples = pb.get_mic(N);
    data = samples - mean(samples);
    figure(2)
    clf;
    plot_spec(data,fs);
end
```

Warning: Running Matlab in an infinite loop may prevent you from re-gaining control over Matlab or even your computer. There are two things you may try if you want to get back control: 1) Type CTRL+C in the Command Window to interrupt Matlab; 2) kill the Matlab process and restart it again.

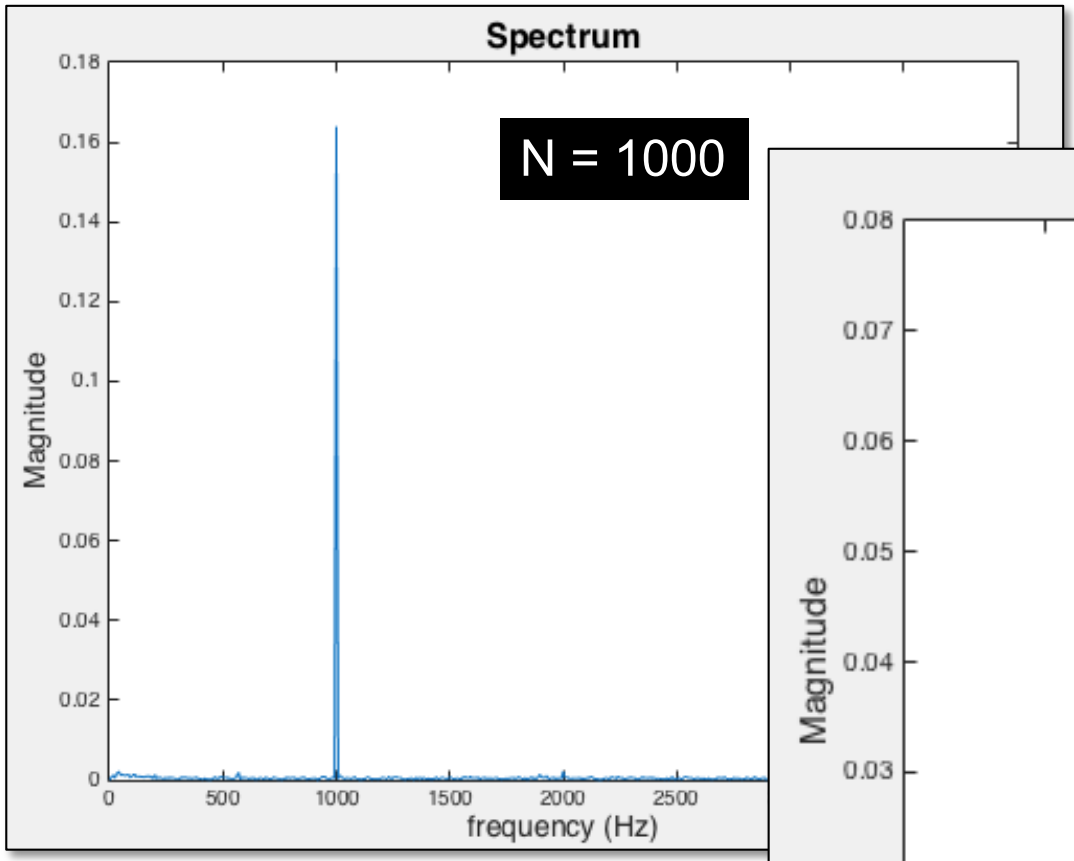
Recover from lost of serial communication:

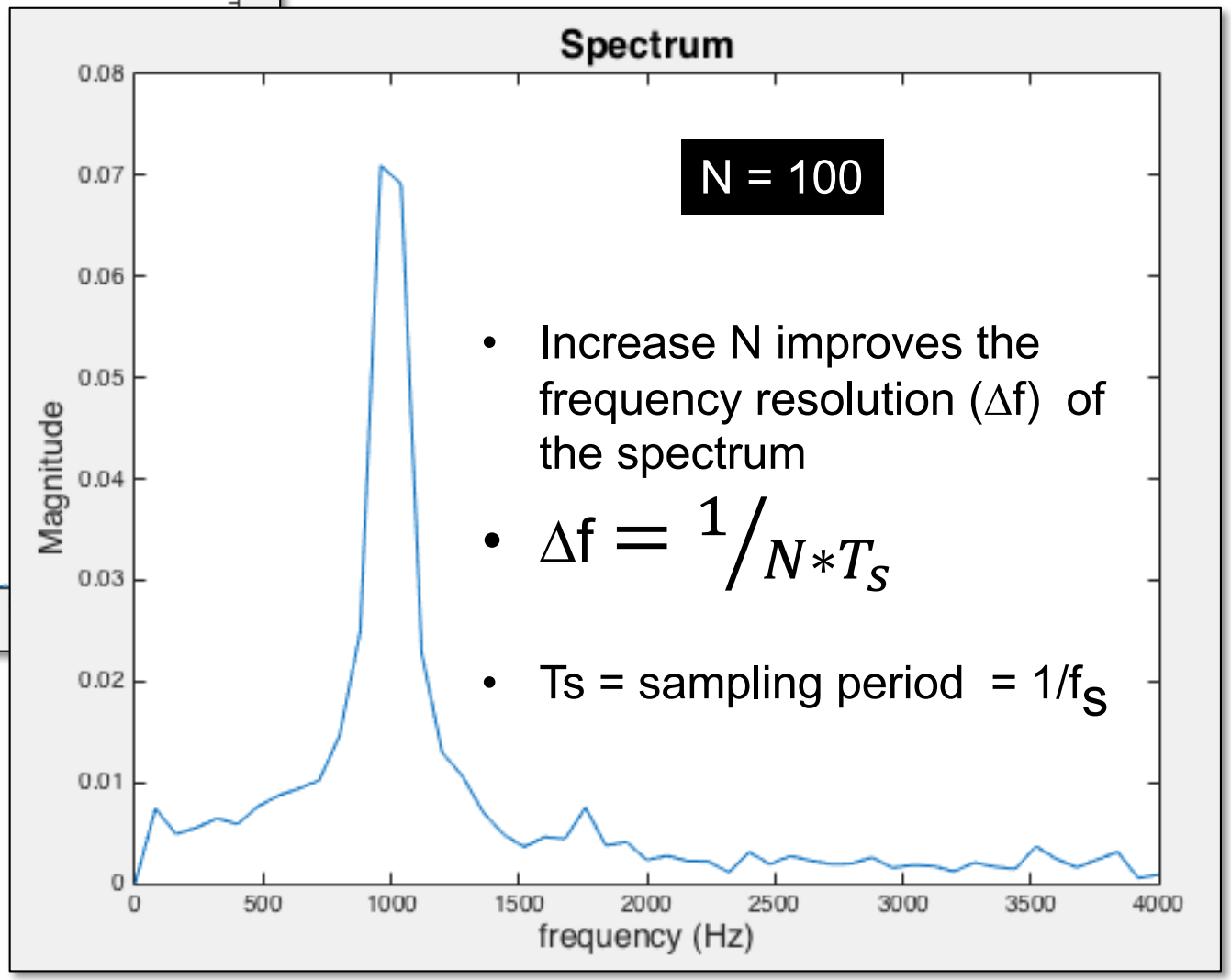
1. Disconnect/reconnect USB; kill & restart Mablab
2. CTRL+C in command window, then type fclose(pb.usb) to shut down usb communication port

Lab 2 Task 3 – Demonstrate spectral folding (aliasing)



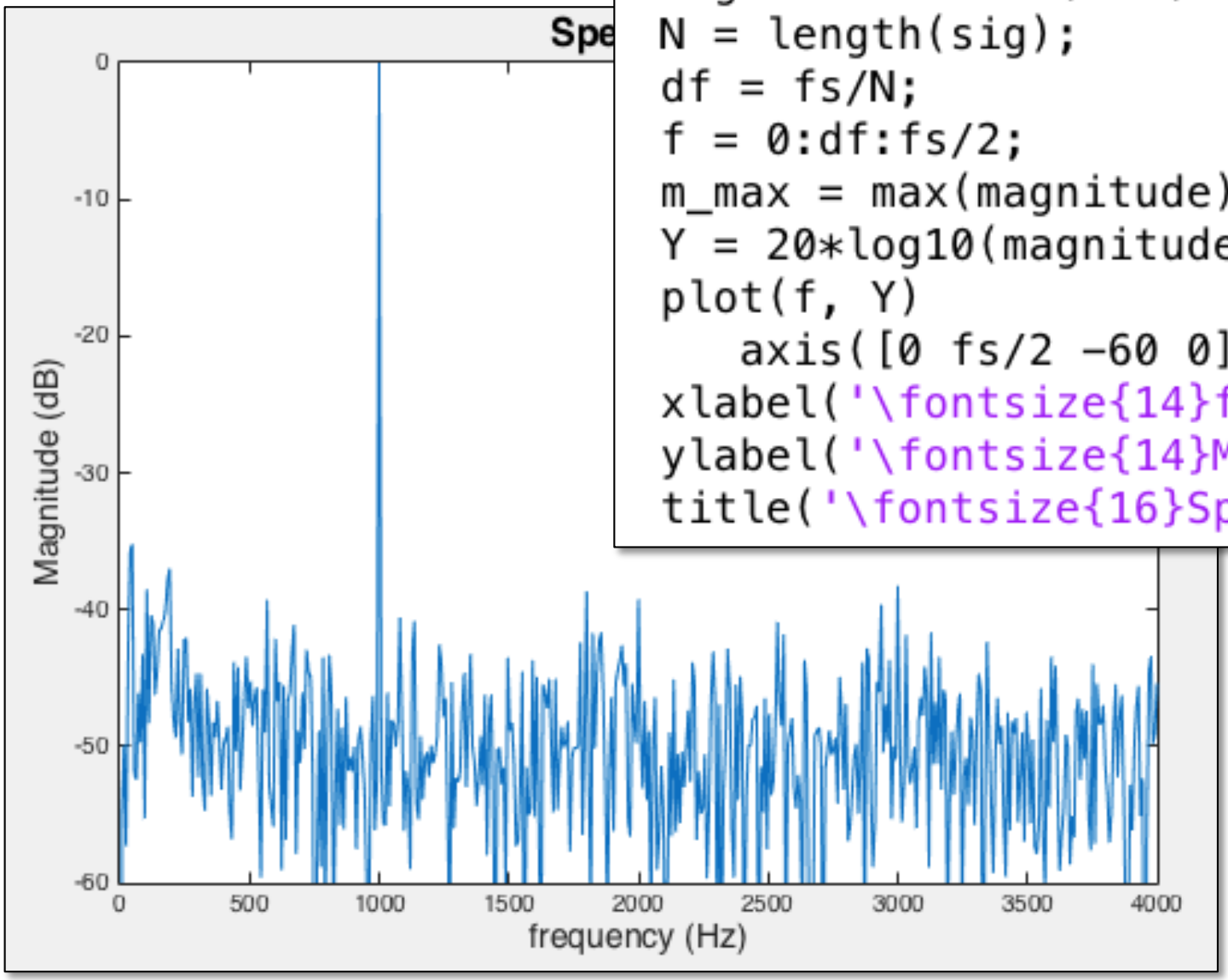
Lab 2 Task 3 – Effect of changing N – no of samples to analyse





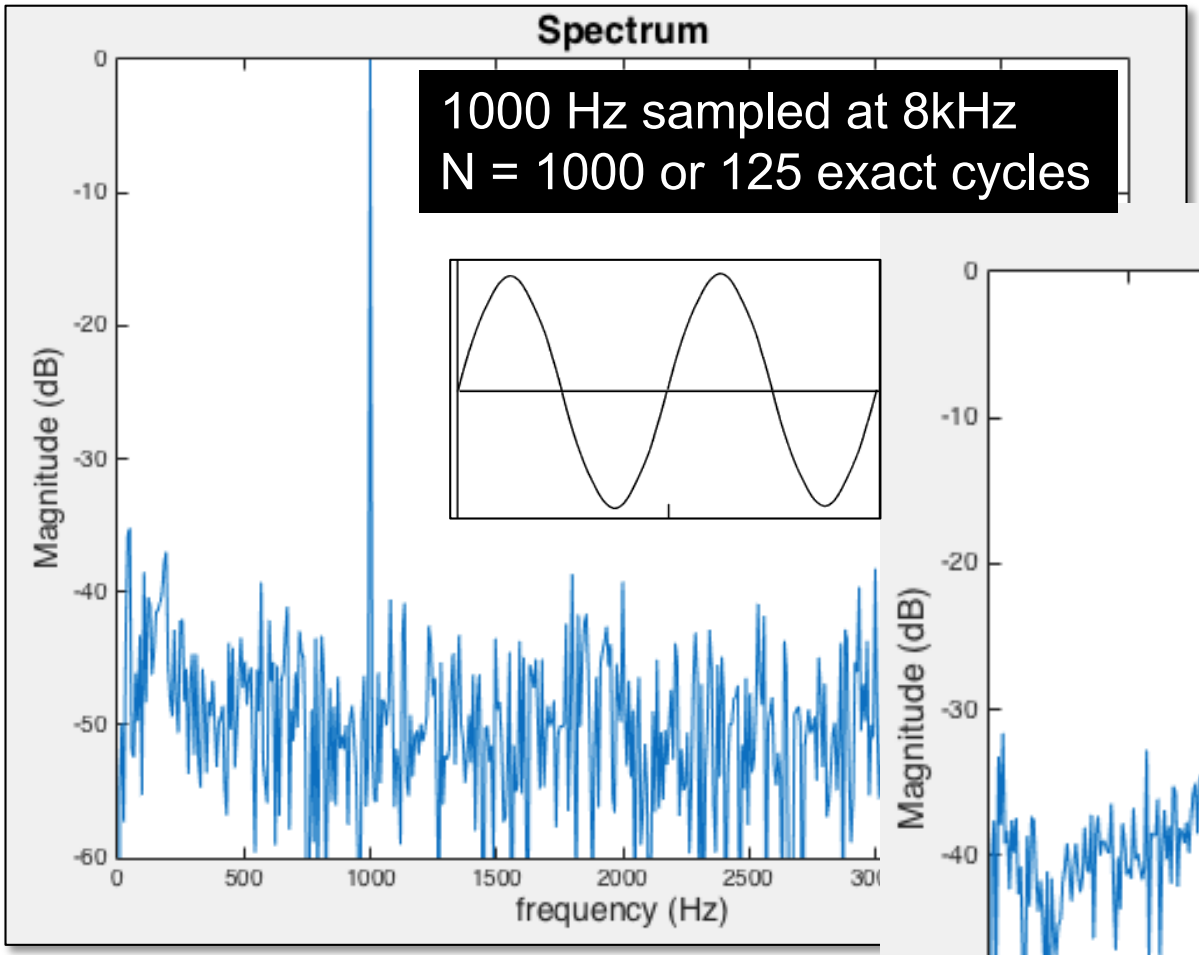
- Increase N improves the frequency resolution (Δf) of the spectrum
- $\Delta f = {}^{1}/_{N*T_S}$
- Ts = sampling period = $1/f_S$

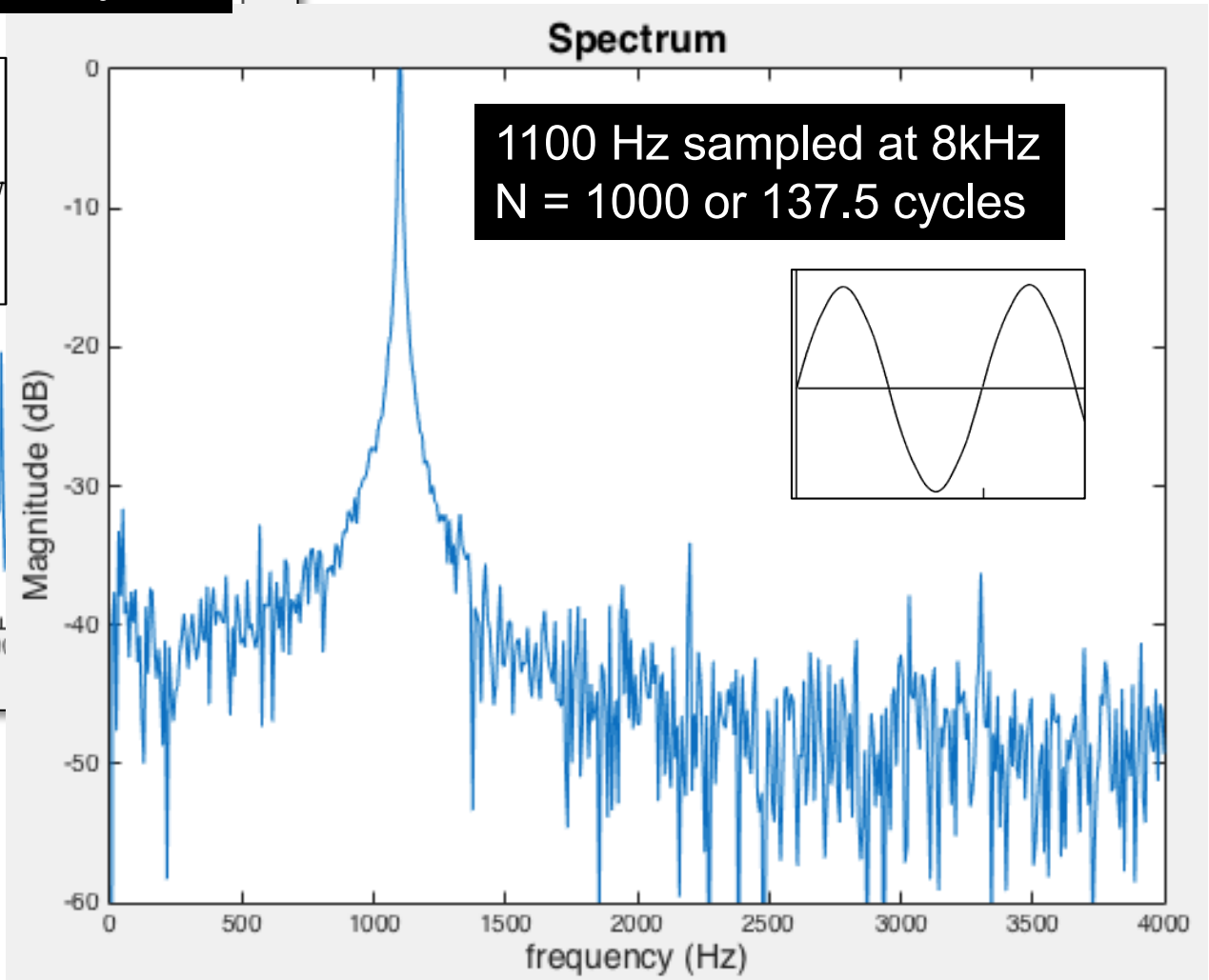
Lab 2 Task 4 – Magnitude in dB



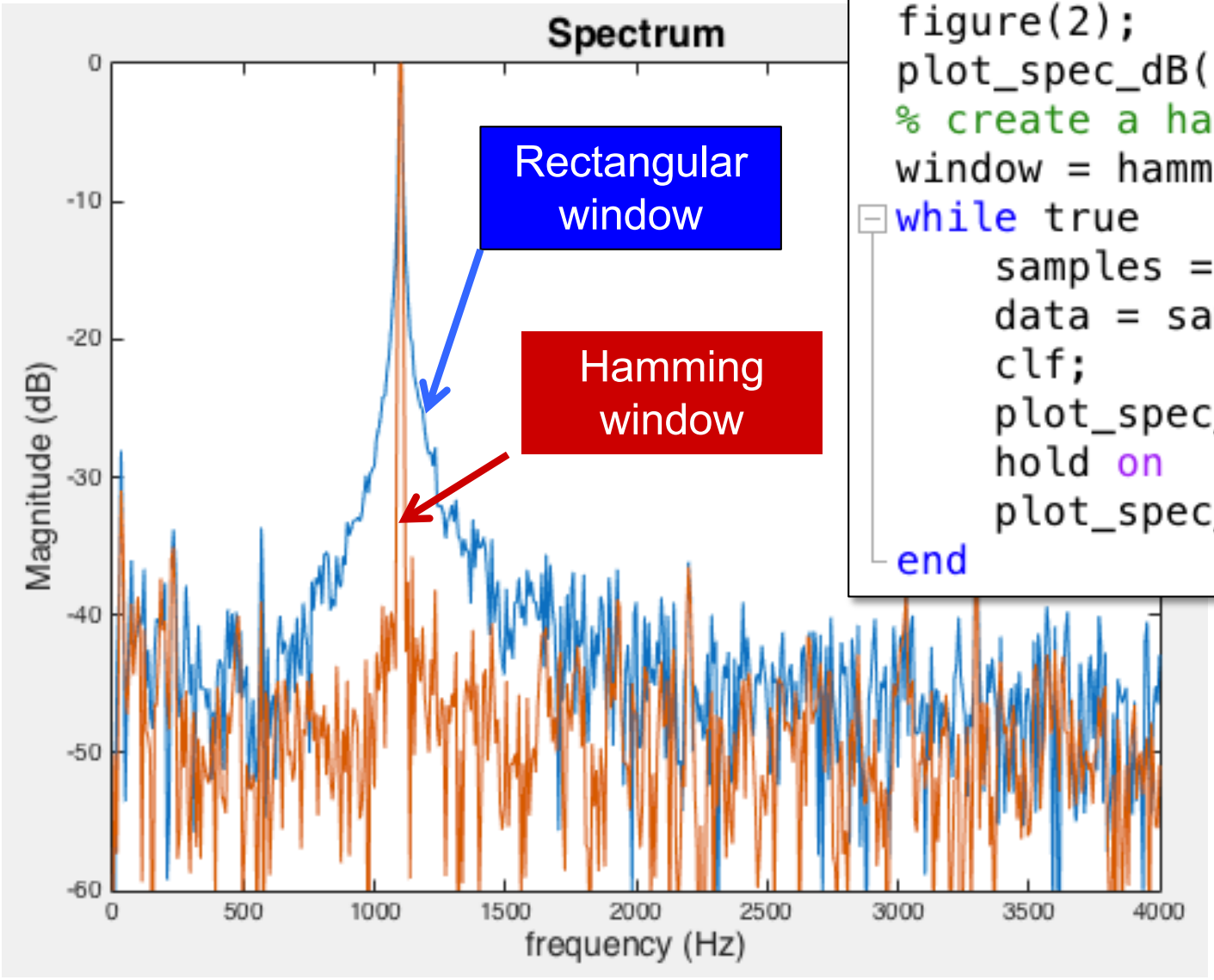

```
magnitude = abs(fft(sig));
N = length(sig);
df = fs/N;
f = 0:df:fs/2;
m_max = max(magnitude);
Y = 20*log10(magnitude(1:length(f))/m_max);
plot(f, Y)
   axis([0 fs/2 -60 0]);
xlabel('\fontsize{14}frequency (Hz)')
ylabel('\fontsize{14}Magnitude (dB)');
title('\fontsize{16}Spectrum');
```

Lab 2 Exercise 4 – Windowing effect





Lab 2 Task 4 – Rectangular vs Hamming Window




```
% find spectrum
figure(2);
plot_spec_dB(data,fs);
% create a hamming window
window = hamming(length(data));
while true
    samples = pb.get_mic(N);
    data = samples - mean(samples);
    clf;
    plot_spec_dB(data,fs);
    hold on
    plot_spec_dB(data.*window,fs);
end
```

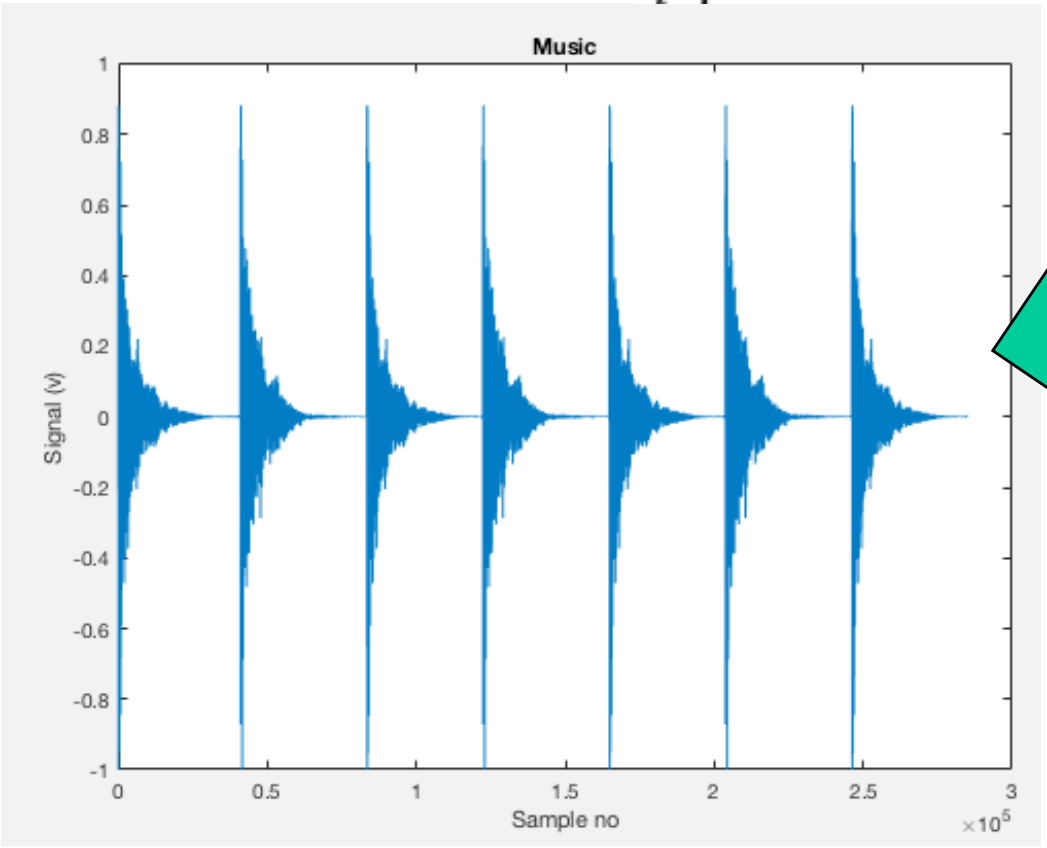
Lab 2 Task 5 – Calculate energy in 20ms segment

```
% Divide signal into se
T = 0.02;          % divid
N = fs*T;          % for t
E = [];
for i=1:N:length(sig)-N+1
    seg = sig(i:i+N-1);
    E = [E seg'*seg];
end
% plot the energy graph and the peak values
figure(2);
clf;
x = 1:length(E);
plot(x, E)
xlabel('Segment number');
ylabel('Energy');
hold on
% Find local maxima
[pks locs] = findpeaks(E);
plot(locs, pks, 'o');
hold off
% plot spectrum of energy
figure(3)
plot_spec(E - mean(E), 1/T);
```

$$\sum_{i=1}^{N} x^2(t)$$ where N is the number of samples in 20ms

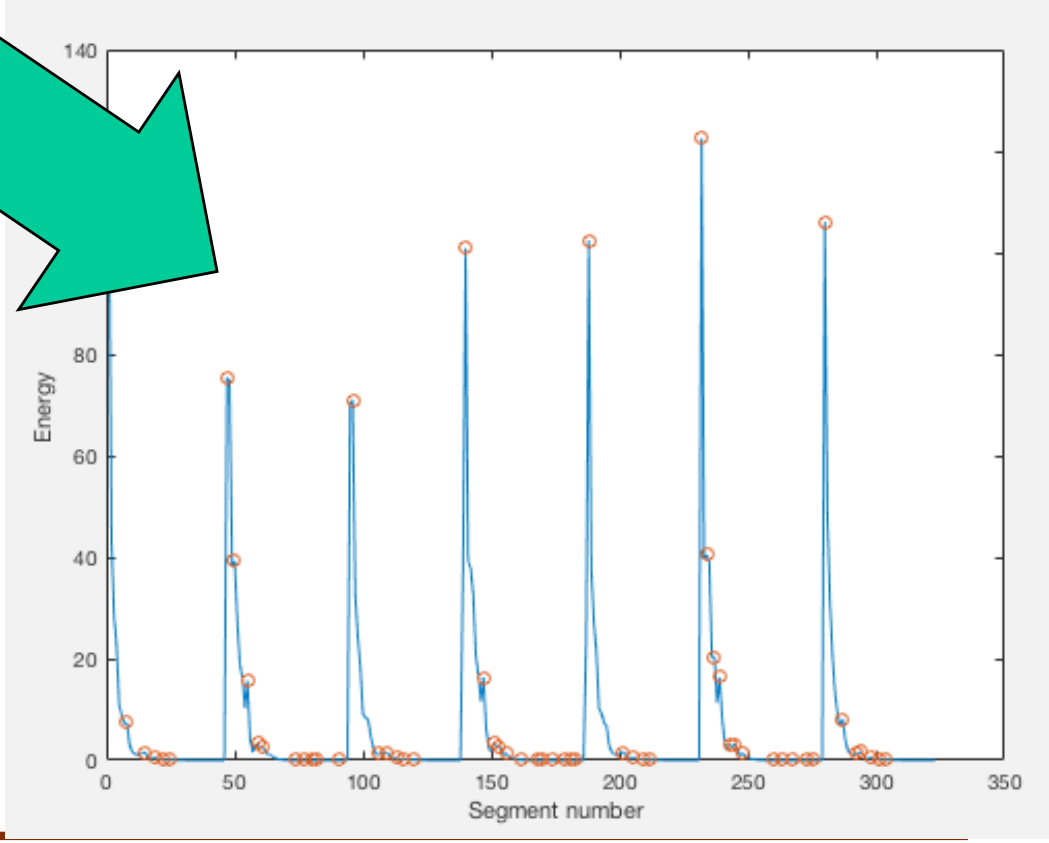
Lab 2 Task 5 – Analyse beat of drum beats (1)

$$\sum_{i=1}^{N} x^2(t)$$ where N is the number of samples in 20ms



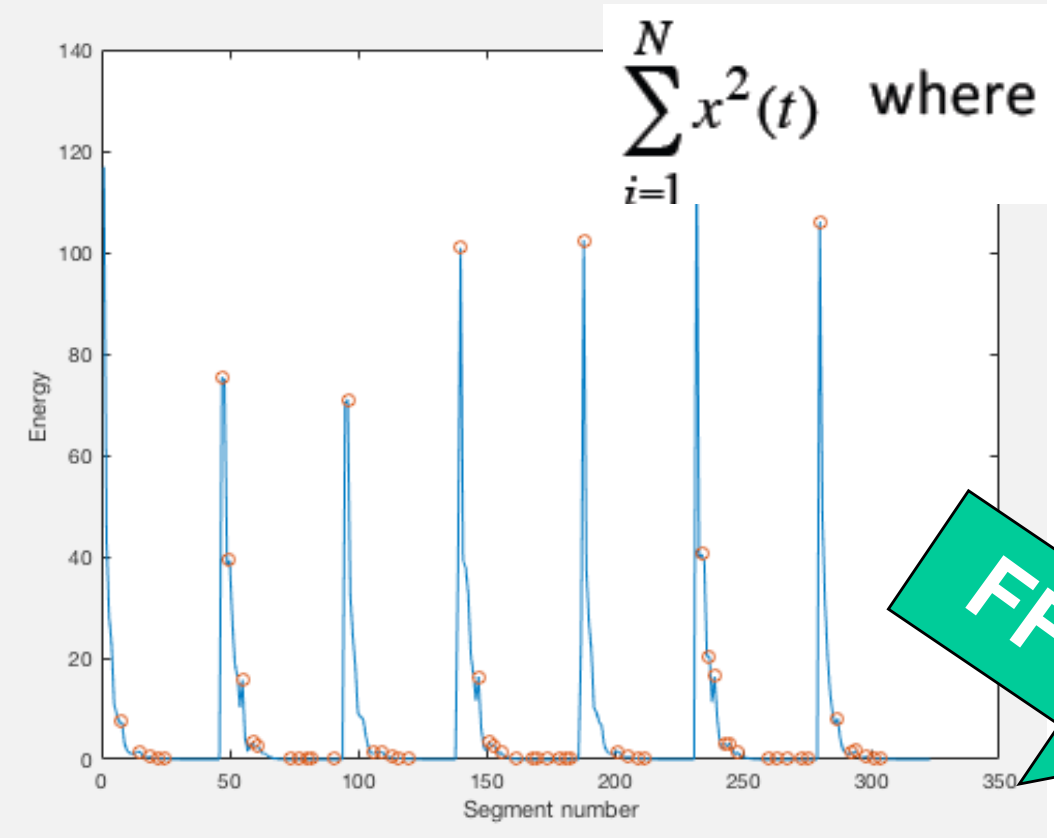
Signal x(t)

Energy of signal



Lab 2 Task 5 – Analyse beat of drum beats (2)

$$\sum_{i=1}^{N} x^2(t)$$ where N is the number of samples in 20ms



Energy vs time

FFT

Spectrum of energy