Mesures et expérimentation assistées par ordinateur et téléphone portable

Pierre DIEUMEGARD

pierre.dieumegard@free.fr

Table des matières

Principe : les téléphones portables multifonctions (= smartphones = ordiphones) ont des capteurs physiques intégrés, variables selon les modèles. Parmi les plus répandus, on peut citer les capteurs de lumière, d'accélération (donc de l'accélération de la pesanteur et de la gravité) et de champ magnétique. On peut parfois trouver aussi capteurs de température et de pression.

Il existe déjà des logiciels pouvant afficher sur l'écran du téléphone les valeurs numériques, ou des graphiques correspondant à ces mesures. Mais l'écran du téléphone est petit, et ce n'est pas facile de conserver les mesures et d'en faire des traitements ultérieurs par modélisation.

On peut donc souhaiter envoyer automatiquement les mesures faites par le téléphone vers un ordinateur, plus commode pour traiter les mesures.

Ici sera détaillé le cas d'un téléphone portable fonctionnant avec le système Android, et communiquant avec l'ordinateur soit par un câble USB, soit par Bluetooth, soit par un réseau local wifi, puisque les téléphones portables peuvent servir de point d'accès à Internet par l'un ou l'autre de ces moyens.

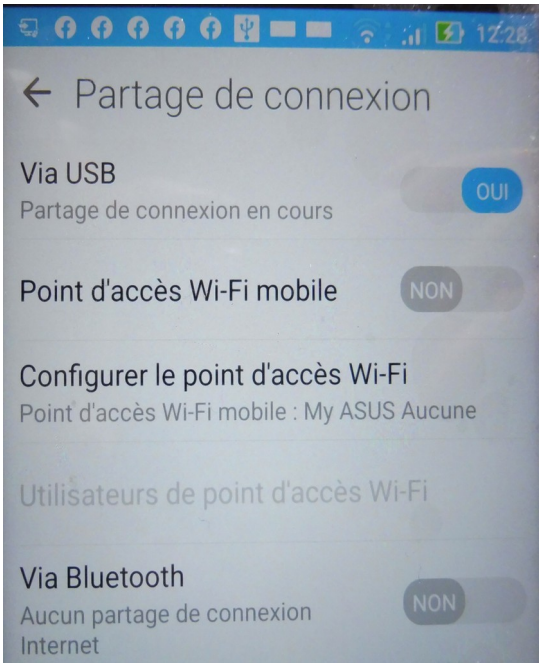


Illustration 1: Copie d'écran pour configuration en connexion par USB (Paramètres / Plus / Partage de connexion)

Le téléphone (ou la tablette) va être le **serveur** : il va fournir des informations à l'ordinateur. L'ordinateur va être le **client** : il va recevoir les informations du téléphone.

Une première étape est d'associer les deux appareils. Le serveur (le téléphone) aura une identification particulière, appelée « adresse IP », utilisée dans les communications par réseau et obtenue automatiquement.Il existe deux variantes « IPV4 » sous une forme de 4 nombres séparés par des points, par exemple « 192.168.42.129 », pour le réseau local, et « IPV6 » sous une forme de 8 groupes de chiffres et lettres, par exemple « 2a01:e0a:33b:e440:a428:749a:a2:df7a % 3 », pour Internet. Dans ce qui suit, il faut obtenir une adresse IPV4. Le serveur a besoin aussi d'un « port de communication », qui est un nombre, à choisir plus ou moins arbitrairement. Ensuite, le client doit appeler le serveur sur la bonne adresse et avec le bon port.

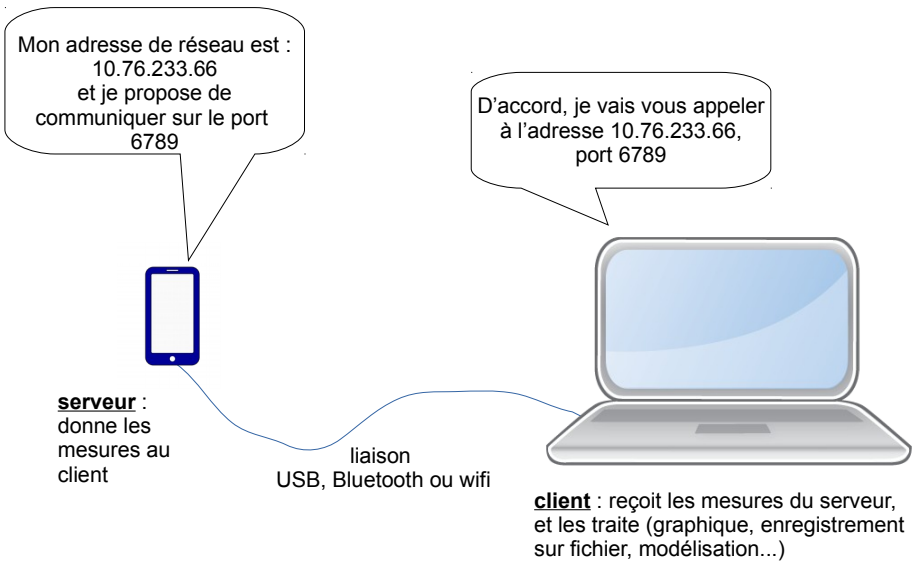
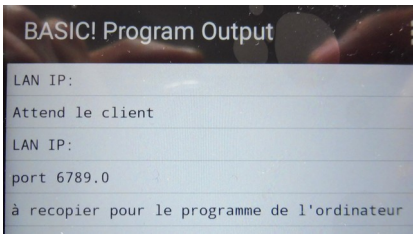
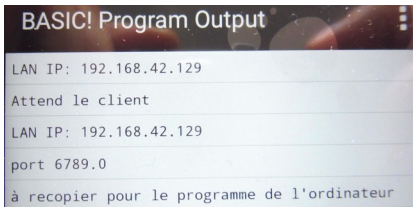


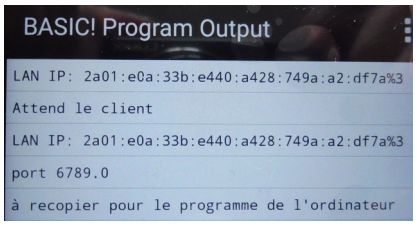
Illustration 2: Communication entre un serveur et un client



Pas d'adresse IP : communication impossible



Adresse IPV4 obtenue : la communication est possible



Adresse IPV6 obtenue : c'est bien pour la communication par Internet, mais non pour le fonctionnement en client/serveur

Illustration 3: Obtention d'adresse IP (programme RFO Basic = Basic!)

Lorsque la communication est bien établie, alors le client peut envoyer des messages au serveur, et inversement. Il faut alors que les programmes qui fonctionnent sur le client soient coordonnés avec ceux du serveur.

Lorsque le téléphone recevra une chaîne de caractères, il fera un ensemble de mesures (sur tous les capteurs disponibles, variables selon les appareils), et l'enverra à l'ordinateur. Ce programme est donc assez simple. Dans la pratique, on peut le faire (au moins) en Python par QPython3, et en Basic par RFOBasic = Basic !, qui sont deux langages facilement disponibles sur Internet.

Sur l'ordinateur, les programmes peuvent être réalisés dans des langages très variés : Python, Pascal (FreePascal et Lazarus), Basic (PureBasic). Si le langage utilisé permet de réaliser des bibliothèques dynamiques, on pourra faire des pilotes pour le système Mensura.

1 Sur le téléphone : petit programme qui renvoie les valeurs mesurées lorsqu'il a reçu une chaîne de caractères

Il existe différents langages pouvant permettre la programmation sur les ordiphones Android. L'important est qu'ils puissent à la fois gérer les capteurs et communiquer par le protocole IP.

Les capteurs utilisables sont variables d'un appareil à un autre. Diverses applications disponibles dans Play Store permettent d'afficher les mesures : Physics Toolbox Suite, ou Capteurs Multi Outils.

Pour un même appareil, certains capteurs peuvent être utilisés par certains langages et non par d'autres. Par exemple, QPython 3 semble capable de gérer moins de capteurs que RFOBasic.

Les programmes ci-dessous commencent par afficher l'adresse IP et le port par lesquels l'ordinateur devra établir la connexion. Ensuite, à chaque fois qu'ils reçoivent une chaîne de caractères, ils renvoient vers l'ordinateur une chaîne ayant la structure suivante :

"accel=", les trois composantes de l'accélération,";mag=",les trois composantes du magnétisme,";lum=",lumière,"temp=",température

1.1 QPython3

```
#-*-coding:utf8;-*-
#qpy:3
#qpy:console

print("module serveur console")

# SERVEUR
import socket, time
from androidhelper import Android
droid=Android()

# Identification réseau de ce programme
IP = ''
PORT = 6789
IP=socket.gethostbyname(socket.gethostname())
s=socket.socket(socket.AF_INET,socket .SOCK_DGRAM)
s.connect(('8.8.8.8',80))
IP=s.getsockname()[0]

ADRESSE = IP, PORT
print(ADRESSE)
input('adresses ci-dessus à fournir au programme-client ; validez
maintenant')
# création d'un canal de communication - socket - de type serveur
serveur = socket.socket() # création
serveur.bind(ADRESSE) # association à l'adresse du programme ...
serveur.listen(1) # #écoute du réseau

# on attend une connexion entrante
client, adresseClient = serveur.accept()
print('Connexion de', adresseClient)
x=777
# Boucle de dialogue
while True:
    recu = client.recv(1024)
    if len(recu) == 0:
        print('Erreur de réception.')
        break
    else:
        recu = recu.decode('utf-8') # décodage du message reçu
        print('Réception de:', recu)
        droid.startSensingTimed(1,300)

        r=droid.sensorsReadAccelerometer().result
        reponse="accel="+str(r[0])+"/"+str(r[1])+"/"+str(r[2])
        r=droid.sensorsReadMagnetometer().result
        reponse=reponse+";mag="+str(r[0])+"/"+str(r[1])+"/"+str(r[2])
        lum=droid.sensorsGetLight()[1]
        reponse=reponse+";lum="+str(lum)
```

```
        print('Envoi de :', reponse+chr(13)+chr(10))
        droid.stopSensing()

        reponse = reponse.encode('utf-8') # encodage du message à émettre
        n = client.send(reponse)
        if n != len(reponse):
            print('Erreur envoi.')
            break
        else:
            print('Envoi ok.')
        if reponse=="":
            break

# si on est là, c'est que la connexion est rompue ;
# il faut alors fermer les canaux de communication
print('Fermeture de la connexion avec le client.')
client.close()
print('On se débranche')
serveur.close()
```

1.2 RFO Basic = Basic !

C'est un langage de type Basic, qui a deux intérêts : d'une part il semble qu'il puisse utiliser plus de capteurs que QPython3, et d'autre part on peut assez facilement transformer un programme RFO Basic en un exécutable Android APK.

```
SOCKET.MYIP ip$
INPUT "numéro du port ?", port, 6789
! crée le serveur au port sélectionné
SOCKET.SERVER.CREATE port

! Connecte le prochain client (= ordinateur)
PRINT "Attend le client"
PRINT "LAN IP: " + ip$
print "port "; port
print "à recopier pour le programme de l'ordinateur"

SOCKET.SERVER.CONNECT 0
DO
 SOCKET.SERVER.STATUS st
UNTIL st = 3

SOCKET.SERVER.CLIENT.IP ip$
PRINT "Connecté à ";ip$
! Connecté à un Client
! Attend que le client envoie un message
! ou dépasse le délai après 10 secondes

sensors.open 1,2,5,6,7
! senseurs pour accel,mag,lum,press,temp
pause 700

nouvelleLigne:
maxclock = CLOCK() + 50000
DO
 SOCKET.SERVER.READ.READY flag
```

```
 IF CLOCK() > maxclock
  PRINT "délai dépassé"
  END
 ENDIF
UNTIL flag

! Message reçu, le lire et l'imprimer puis faire les mesures
SOCKET.SERVER.READ.LINE line$
PRINT line$

sensors.read 5 ,lx,ly,lz
sensors.read 6 ,px,py,pz
sensors.read 7 ,tx,ty,tz
sensors.read 1 ,ax,ay,az
sensors.read 2 ,mx,my,mz
r$="accel="+str$(ax)+"/"+str$(ay)+"/"+str$(az)
r$=r$+";mag="+str$(mx)+"/"+str$(my)+"/"+str$(mz)
r$=r$+";lum="+str$(lx)+";press="+str$(px)+";temp="+str$(tx)

! renvoyer au client le résultat des mesures0
SOCKET.SERVER.WRITE.LINE r$
print "r$="+r$  ! pour controle sur ordiphone
goto nouvelleLigne

sensors.close
! Déconnexion du client, mais en pratique, la boucle est infinie
SOCKET.SERVER.DISCONNECT
PRINT "déconnecté du client"
```

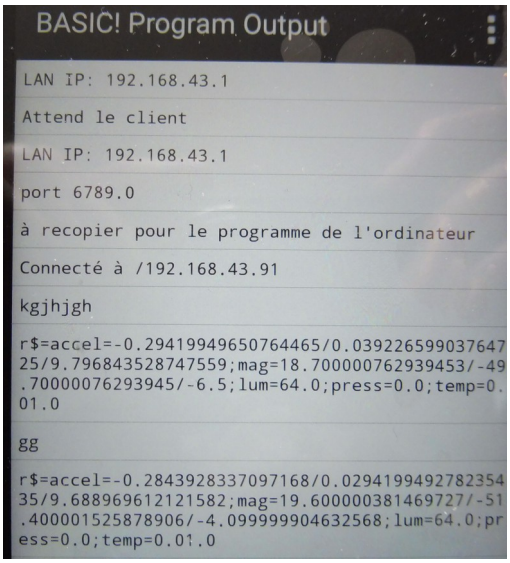


Illustration 4: Copie d'écran du téléphone, après envoi de deux chaines de caractères par l'ordinateur

2 Programmes possibles du côté de l'ordinateur

Les programmes décrits ci-dessous sont volontairement simples, et la chaîne de caractères reçue du téléphone n'est pas décodée. Pour vraiment exploiter les mesures, il faut découper cette chaîne et récupérer les valeurs numériques mesurées.

2.1 Python

```
import socket
import time

HOST='192.168.42.129' # pour RFOBasic
PORT = 6789

input("lancez le programme serveur sur le telephone portable")
client = socket.socket()
client.connect((HOST, PORT))
print('Connexion vers ' + HOST + ':' + str(PORT) + ' reussie.')

while True:
    message=input("entrez la chaine a envoyer")
    if message=="":
        break
    message=message+chr(13)+chr(10)
    print('Envoi de :', message)
    message = message.encode('utf-8')
    n = client.send(message)
    if n != len(message):
        print('Erreur envoi.')
        break
    else:
        recu = client.recv(1024)
        recu = recu.decode('utf-8')
        print('Envoi ok.', 'Reception...', 'Recu :', recu)
    if message=="":
        print("on arrete !")
        break

print('Deconnexion.')
client.close()
```

2.2 PureBasic

```
If InitNetwork() = 0
  MessageRequester("Erreur", "Initialisation du réseau impossible !", 0)
  End
EndIf

OpenConsole()
Port = 6789

Procedure.s ReceiveNetworkString(CID.i)
  Protected longueur.l, *Donnees = AllocateMemory(2000)
  longueur = ReceiveNetworkData(CID, *Donnees, 2000)
  ProcedureReturn PeekS(*Donnees, longueur,#PB_Ascii  )
```

```
EndProcedure

Global ConnectionID = OpenNetworkConnection("192.168.42.129", 6789)
If ConnectionID
    PrintN("connexion faite")
    For i=1 To 20
      SendNetworkString(ConnectionID, "xxx"+Chr(13)+Chr(10),#PB_Ascii  )
      Delay(100)
      chaine$=receivenetworkstring(ConnectionID)
      PrintN("chaine$="+chaine$)
    Next i

    CloseNetworkConnection(ConnectionID)
Else
  MessageRequester("PureBasic - Client", "Ne peut pas trouver le
serveur", 0)
EndIf

CloseConsole()
```

Un des intérêts de PureBasic est qu'il est un langage compilé, et donc peut permettre de faire des bibliothèques dynamiques (DLL), qui servent de pilotes d'appareils de mesure pour le système Mensura.

2.3 Pascal, et plus précisément Lazarus

Lazarus est aussi un langage de programmation compilé, et il peut donc permettre de faire des bibliothèques dynamiques pour Mensura. On peut aussi faire de grosses applications exécutables.

```
program testLazarus;
{ d'après un exemple de Michael van Canneyt et Peter Vreman }
uses Sockets, sysutils;

procedure PError(const S : string);
begin     writeln(S,SocketError);    halt(100); end;

Var
  SAddr    : TInetSockAddr;
  Buffer   : string [255];
  S        : Longint;
  Sin,Sout : Text;
  i        : integer;

begin
  S:=fpSocket (AF_INET,SOCK_STREAM,0);
  if s=-1 then  Perror('Client : Socket : ');
  SAddr.sin_family:=AF_INET;
  SAddr.sin_port:=htons(6789);
  Saddr.sin_addr:=StrToNetAddr('192.168.42.129');
  if not Connect (S,SAddr,Sin,Sout) then
     PError('Client : Connect : ');
  Reset(Sin);
  ReWrite(Sout);
  for i:=1 to 10 do
```

```
      begin
           Buffer:='h';
           Writeln(Sout,Buffer);
           sleep(300);
           Flush(Sout);
           sleep(300);
           Readln(SIn,Buffer);
           WriteLn(Buffer);
      end;
  Close(sout);
end.
```

3 Système Mensura : pilotes et programmes d'application

Avec PureBasic ou Lazarus, on peut faire des bibliothèques dynamiques pour les principaux systèmes d'exploitation, Linux et Windows (et probablement Mac). Ces bibliothèques peuvent être appelées par un très grand nombre de langages de programmation : les langages précédents, mais aussi beaucoup d'autres.

Consultez http://sciencexp.free.fr, et plus précisément http://sciencexp.free.fr/index.php?perma=1584807541

Vous y trouverez une application APK à mettre directement dans un téléphone Android, un pilote pour le système Mensura, pour Linux et Windows, avec source en PureBasic et divers logiciels généralistes (MGW32 pour Windows, Mensurasoft-PB et Mensurasoft-LZ pour Linux).

4 Exemples d'application : période du pendule

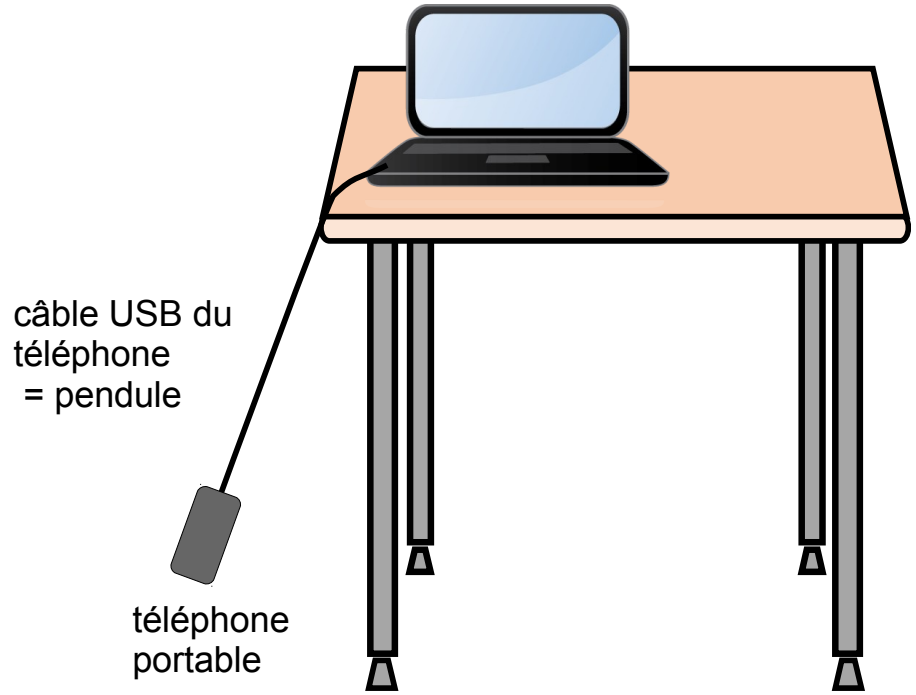


Illustration 5: Dispositif expérimental : pendule formé par le téléphone et son câble

Le câble USB qui relie le téléphone à l'ordinateur peut servir de pendule (attention ! fixer solidement le câble au téléphone pour éviter de détériorer ce dernier !).Le capteur d'accélération permet de mesurer l'accélération, dans un axe quelconque (X, Y ou Z). Comme la position du pendule (= téléphone et son câble) varie en fonction du temps, l'accélération mesurée varie aussi, et on peut mesurer l'oscillation (amplitude et fréquence) de ce pendule.

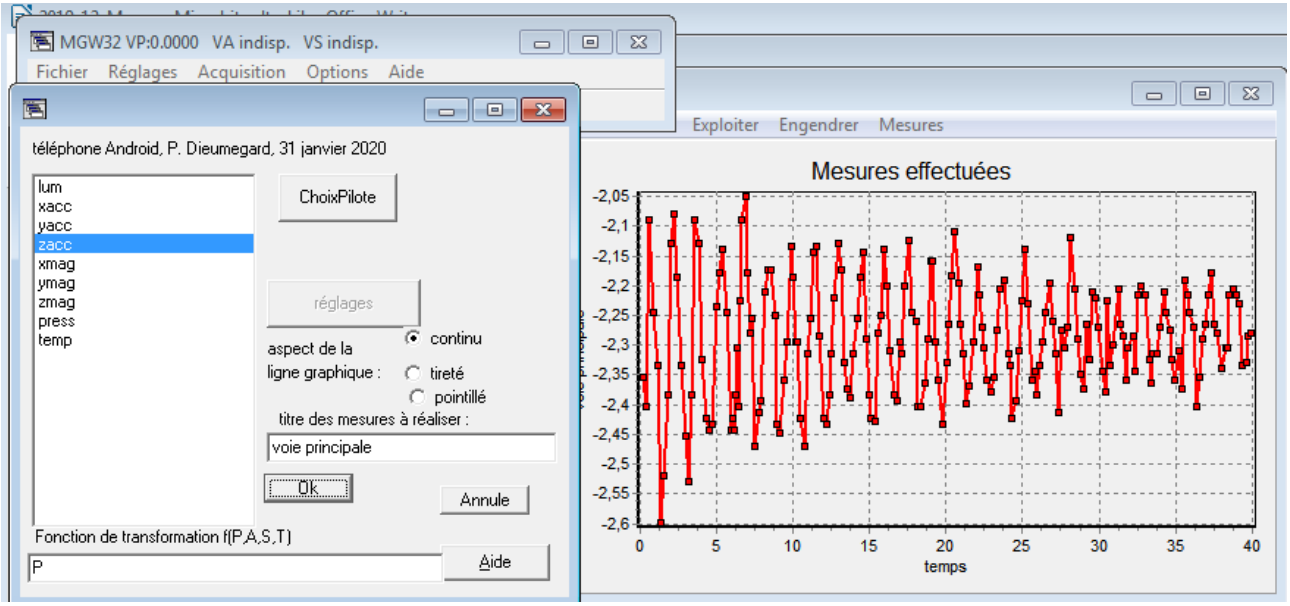
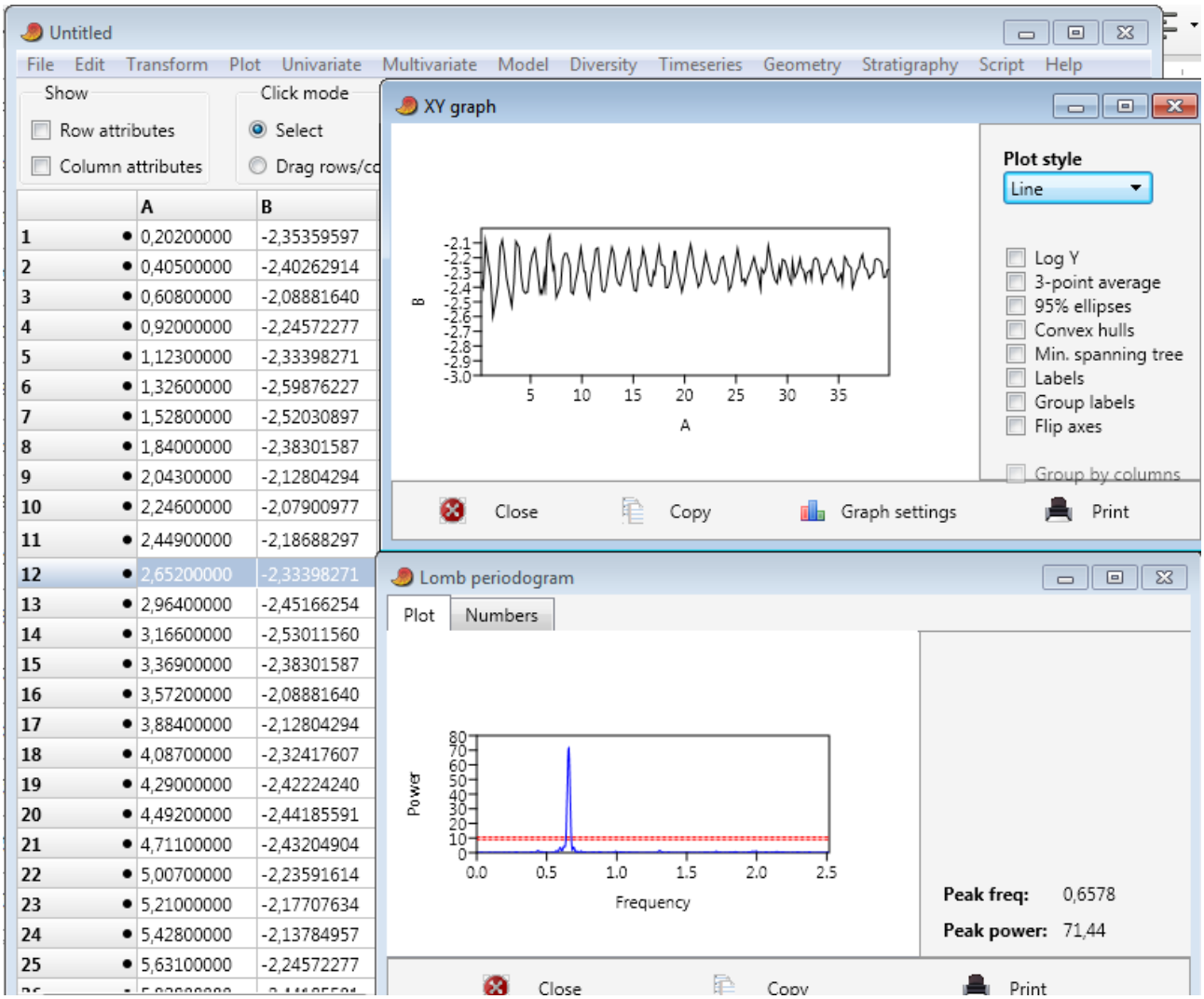


Illustration 6: Acquisition de l'accélération par un téléphone portable (logiciel MGW32)

Comme le câble USB a une longueur de l'ordre du mètre, la période est de l'ordre d'une seconde : il faut plusieurs acquisitions par seconde, et on doit régler l'intervalle entre deux mesures à environ 200 ms.

On voit que l'accélération varie cycliquement, et que l'amplitude des variations diminue peu à peu (oscillations amorties). Pour calculer la fréquence (et donc la période), le mieux est d'utiliser un logiciel spécialisé, par exemple PAST. Dans le logiciel MGW32, le menu Editer / Copier permet d'envoyer la série de mesures dans le presse-papier, et il suffit ensuite de les coller dans le logiciel PAST.



	A	B
1	0,20200000	-2,35359597
2	0,40500000	-2,40262914
3	0,60800000	-2,08881640
4	0,92000000	-2,24572277
5	1,12300000	-2,33398271
6	1,32600000	-2,59876227
7	1,52800000	-2,52030897
8	1,84000000	-2,38301587
9	2,04300000	-2,12804294
10	2,24600000	-2,07900977
11	2,44900000	-2,18688297
12	2,65200000	-2,33398271
13	2,96400000	-2,45166254
14	3,16600000	-2,53011560
15	3,36900000	-2,38301587
16	3,57200000	-2,08881640
17	3,88400000	-2,12804294
18	4,08700000	-2,32417607
19	4,29000000	-2,42224240
20	4,49200000	-2,44185591
21	4,71100000	-2,43204904
22	5,00700000	-2,23591614
23	5,21000000	-2,17707634
24	5,42800000	-2,13784957
25	5,63100000	-2,24572277

Illustration 7: Détermination de la fréquence d'un phénomène périodique par PAST

On voit un seul pic de fréquence(0,6578 Hz) : celle-ci est donc constante pendant toute l'expérience.

Pour aller plus loin, on peut calculer la valeur de l'accélération de la pesanteur (g), en utilisant la relation $T = \frac{1}{F} = 2\pi\sqrt{(\frac{l}{g})}$. On peut calculer $g = l * 4* \pi^2 * F^2$ où l est la longueur du pendule, et F la fréquence des oscillations.

Ici, la longueur est de l'ordre de 60 cm, mais est difficile à estimer précisément, car le centre de gravité de l'ensemble (câble + téléphone) est difficile à déterminer. La relation précédente donne une valeur de g = 10,25, ce qui n'est pas très éloigné de la valeur théorique g = 9,81 $m.s^{-2}$.