Micro:bit et système Mensura

Pierre Dieumegard
pierre.dieumegard@free.fr

Comme Arduino, Micro:bit est une petite carte à microcontrôleur, connectable à un ordinateur par une prise USB, et permettant de faire des mesures ou de commander des actionneurs (lampes, moteurs, relais...).

Son encombrement est encore moindre que celui d'Arduino : environ la moitié de la taille d'une carte bancaire. Il a moins de connecteurs facilement utilisables qu'Arduino, mais il a cinq connecteurs au format « prise banane », connectables avec des câbles standards.

Contrairement à Arduino, dont le langage de programmation est une variant de C, Micro:bit est programmable en Python (et quelques autres langages). Ceci peut permettre de l'utiliser en lycée, où le langage de programmation habituel est Python.

Enfin, Micro:bit a plusieurs capteurs intégrés, dont les plus faciles à utiliser sont les capteurs de lumière et de température, ainsi que l'accéléromètre à 3 axes.

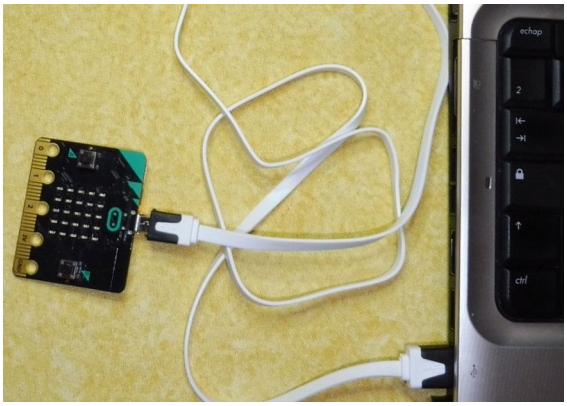


Illustration 1: Micro:bit connecté par un câble (blanc) à une prise USB d'ordinateur portable

Il existe aussi un capteur magnétique (« boussole »), mais qui est un peu plus compliqué à utiliser.

Comment programmer Micro:bit

Sur Internet, la méthode la plus fréquemment utilisée est la programmation par blocs par une application Internet (MakeCode, pour Windows, Macintosh et Linux).

On peut aussi programmer Micro:bit en Python. Une première possibilité est la programmation par blocs (https://app.edublocks.org/#MicroBit) qui donne ensuite un programme en Python. Une autre possibilité est une application Internet (https://python.microbit.org/v/2.0) qui permet une programmation Python par des lignes de programme. Enfin, il existe un logiciel à installer, Mu editor (https://codewith.mu/, pour Windows, Mac et Linux) sur son ordinateur et qui permet la programmation directe en Python d'un Micro:bit relié à cet ordinateur : c'est la manière qui sera développée ici.

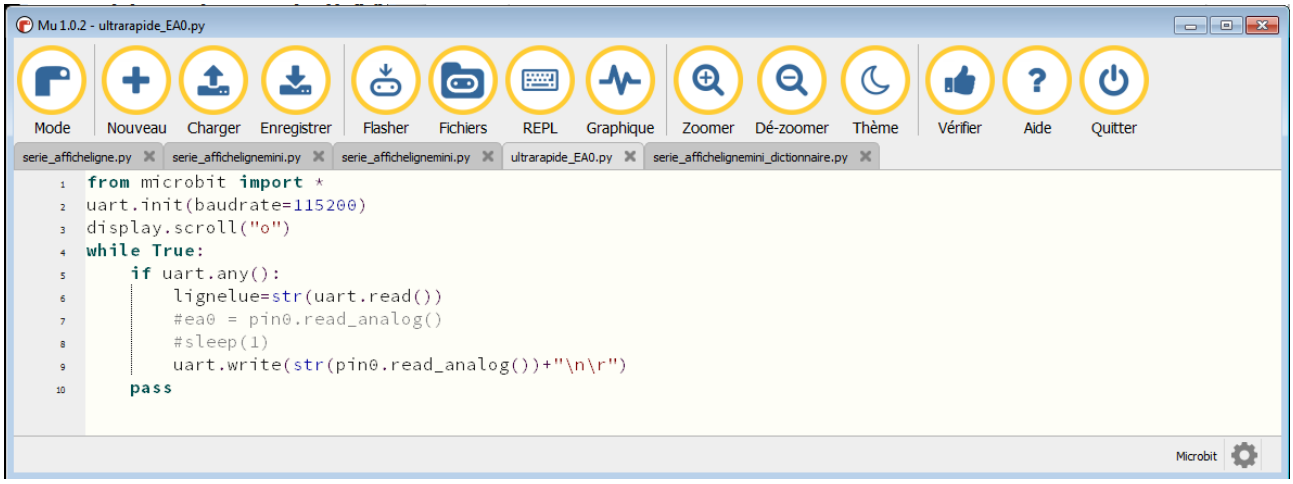


Illustration 2: Editeur Mu pour Python et Micro:bit

Communication en série entre Micro:bit et l'ordinateur

Pour les versions de Windows antérieures à Windows 10 (Windows 8, 7 et antérieurs), il faut installer un pilote (driver) qui permettra à Micro:bit d'être reconnu comme un port série (http://os.mbed.com/media/downloads/drivers/mbedWinSerial_16466.exe).

Ensuite, tous les logiciels de communication par voie série pourront échanger des données avec Micro:bit, à condition que cet appareil ait été programmé correctement.

Le programme ci-dessous initialise la liaison à 115200 bits/seconde (le maximum possible), affiche « o » sur la matrice de diodes lumineuses (simplement pour vérifier que le programme a bien été injecté), puis envoie le résultat d'une mesure en fonction du caractère reçu à partir de l'ordinateur (« 0 » envoie la mesure lue sur le connecteur 0, « 1 » envoie la mesure lue sur le connecteur 1, etc).

```
from microbit import *
uart.init(baudrate=115200)
display.scroll("o")
while True:
    if uart.any():
        lignelue = str(uart.read())
        if lignelue.find("0")>-1:
            uart.write(str(pin0.read_analog())+"\n\r")
        elif lignelue.find("1")>-1:uart.write(str(pin1.read_analog())+"\n\r")
        elif lignelue.find("2")>-1:uart.write(str(pin2.read_analog())+"\n\r")
        elif lignelue.find("L")>-1:
               uart.write(str(display.read_light_level()) + "\n\r")
        elif lignelue.find("X")>-1:
               uart.write(str(accelerometer.get_x())+"\n\r")
        elif lignelue.find("Y")>-1:
               uart.write(str(accelerometer.get_y())+"\n\r")
        elif lignelue.find("Z")>-1:
               uart.write(str(accelerometer.get_z())+"\n\r")
    pass
```

Pour communiquer avec Micro:bit, il faudra un programme de communication sur l'ordinateur, réalisable avec tous les langages normaux de programmation. Le programme en question devra

envoyer un caractère en fonction de la voie qu'on souhaite lire (« 0 » pour la broche 0, « 1 » pour la broche 1, etc), et recevoir une chaîne de caractères correspondant à la mesure effectuée.

Un pilote Mensura en PureBasic pour Micro:bit

Initialisation de la voie série avec les bons paramètres

```
Port$ = "COM6" ; (ou un autre numéro, variable selon les ordinateurs)
vitesse=115200
OpenSerialPort(0, Port$,vitesse, #PB_SerialPort_NoParity, 8, 1,
#PB_SerialPort_NoHandshake, 1024, 1024)
```

Fonctions de mesure (nead et ead)

On définit 7 voies de mesure : les trois entrées numérotées 0 à 2, la mesure de lumière, et les trois axes de l'accéléromètre.

```
ProcedureDLL .s nead(n .w)
Select n
  Case 0 :    nead$="EA0"
  Case 1 : nead$="EA1"
  Case 2 : nead$="EA2"
  Case 3 : nead$="lum"
  Case 4 : nead$="Accel X"
  Case 5: nead$="Accel Y"
  Case 6: nead$="Accel Z"
   Default
    nead$=""
    EndSelect
ProcedureReturn(chaineascii(nead$))
EndProcedure


ProcedureDLL .d ead(n .w)
  chloc$=chainemesures$
  valloc=999
  delai=20 ; il faut un délai pour que Micro:bit fasse la mesure et la renvoie
(au moins 10ms)
   While AvailableSerialPortInput(0)>0
    If ReadSerialPortData(0,@buffer,1) ; on vide le tampon des caractères reçus
      text$=Chr(buffer)
    EndIf
    Wend
  Select n
    Case 0 : WriteSerialPortString(0,"0", #PB_Ascii) : Delay(delai)
     Case 1 : WriteSerialPortString(0,"1", #PB_Ascii) : Delay(delai)
     Case 2 : WriteSerialPortString(0,"2", #PB_Ascii) : Delay(delai)
     Case 3 : WriteSerialPortString(0,"L", #PB_Ascii) : Delay(delai*3)
; la mesure de la lumière prend plus de temps ==> il faut un délai plus long
     Case 4 : WriteSerialPortString(0,"X", #PB_Ascii) : Delay(delai)
     Case 5 : WriteSerialPortString(0,"Y", #PB_Ascii) : Delay(delai)
     Case 6 : WriteSerialPortString(0,"Z", #PB_Ascii) : Delay(delai)
    EndSelect
      text$=""
        While AvailableSerialPortInput(0)>0
          If ReadSerialPortData(0,@buffer,1)
```

```
            text$=text$+Chr(buffer) ; on lit les caractères reçus de Micro:bit
          EndIf
        Wend
      If Len(text$)>0 : valloc=Val(text$):EndIf
ProcedureReturn(valloc)
EndProcedure
```

Une expérience de physique : oscillation du pendule Micro:bit

Montage expérimental

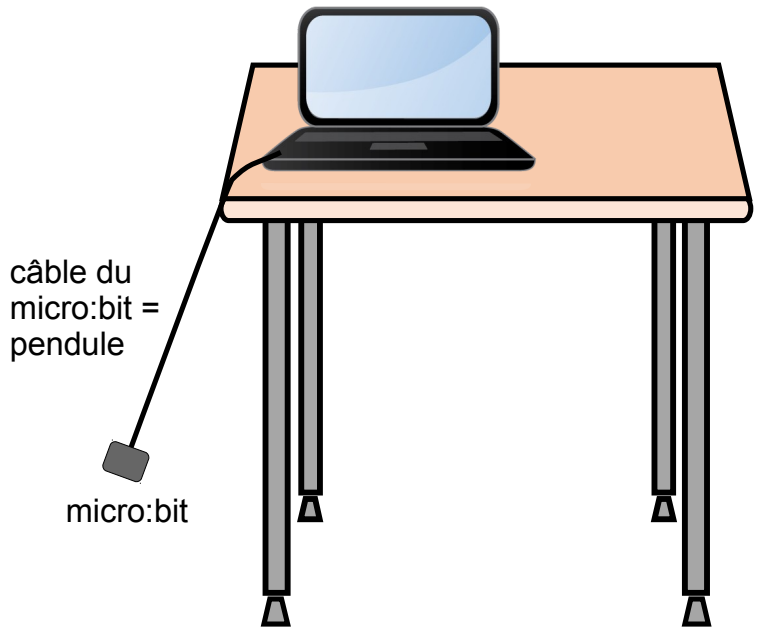


Illustration 3: Dispositif expérimental : micro:bit et son câble servant de pendule

Le capteur d'accélération permet de mesurer l'accélération, dans un axe quelconque (X, Y ou Z). Comme la position du pendule (= micro:bit 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.

Logiciel utilisé et résultats

Les images ci-dessous sont des copies d'écran de MGW32 (logiciel pour Windows) (http://sciencexp.free.fr/logwin32/logwin32.htm). On a réglé la voie principale avec le pilote bibdyn_microbit_PB_rapide.dll (mesures en accélération X = horizontale), puis on a réglé le temps entre 2 mesures à 100 ms, et 500 mesures à faire. Après le déclenchement des mesures, on a écarté le pendule de la verticale, puis on l'a relâché.

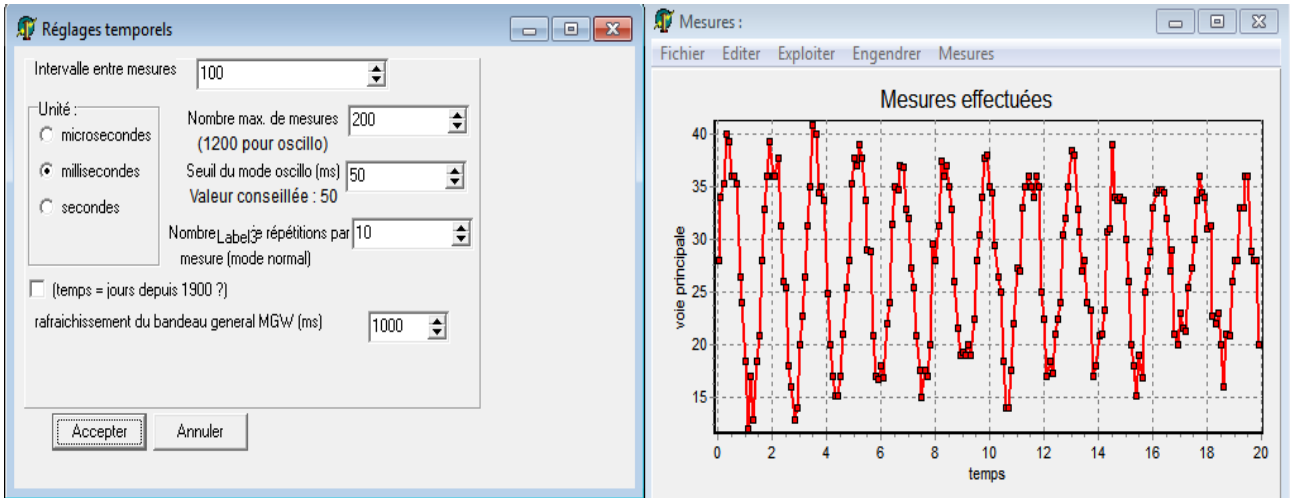


Illustration 4: Expérience du "pendule Micro:bit"

Visuellement, on voit que les oscillations du pendule s'amortissent peu à peu. Il semble que la période ne varie pas ou peu. C'est ce qu'enseigne la théorie : la période du pendule est indépendante de l'amplitude de l'oscillation, et de la masse du pendule, mais ne dépend que de la longueur l du pendule (et de l'accélération de la pesanteur g) : $T = 2\pi\sqrt{(\frac{l}{g})}$

Calcul de la période

On peut calculer la période en divisant l'intervalle de temps entre plusieurs maximums par le nombre de périodes ; pour avoir le maximum de précision, le mieux est de prendre le premier et le dernier maximum de l'enregistrement.

En 20 secondes, il y a eu 12,7 cycles d'accélération horizontale (approximativement). La période est donc 20/12,7 = 1,57 secondes.

(Attention à la composante de l'accélération mesurée : l'accélération verticale a une période deux fois plus petite que l'accélération horizontale !)

Confirmation de l'invariance de la période par le logiciel PAST

PAST est un logiciel d'analyse statistique (https://folk.uio.no/ohammer/past/). On peut coller les valeurs à partir de MGW32 dans PAST :

- Dans le menu de la fenêtre graphique de MGW32, choisir Editer / Copier (virgule décimale)

- Dans PAST, choisir Edit / Paste

Le menu Time Series / Spectral analysis / Simple periodogram permet d'obtenir le « périodogramme » : les abscisses sont les fréquences possibles, et les ordonnées sont la puissance de l'amplitude des sinusoïdes correspondant à ces fréquences.

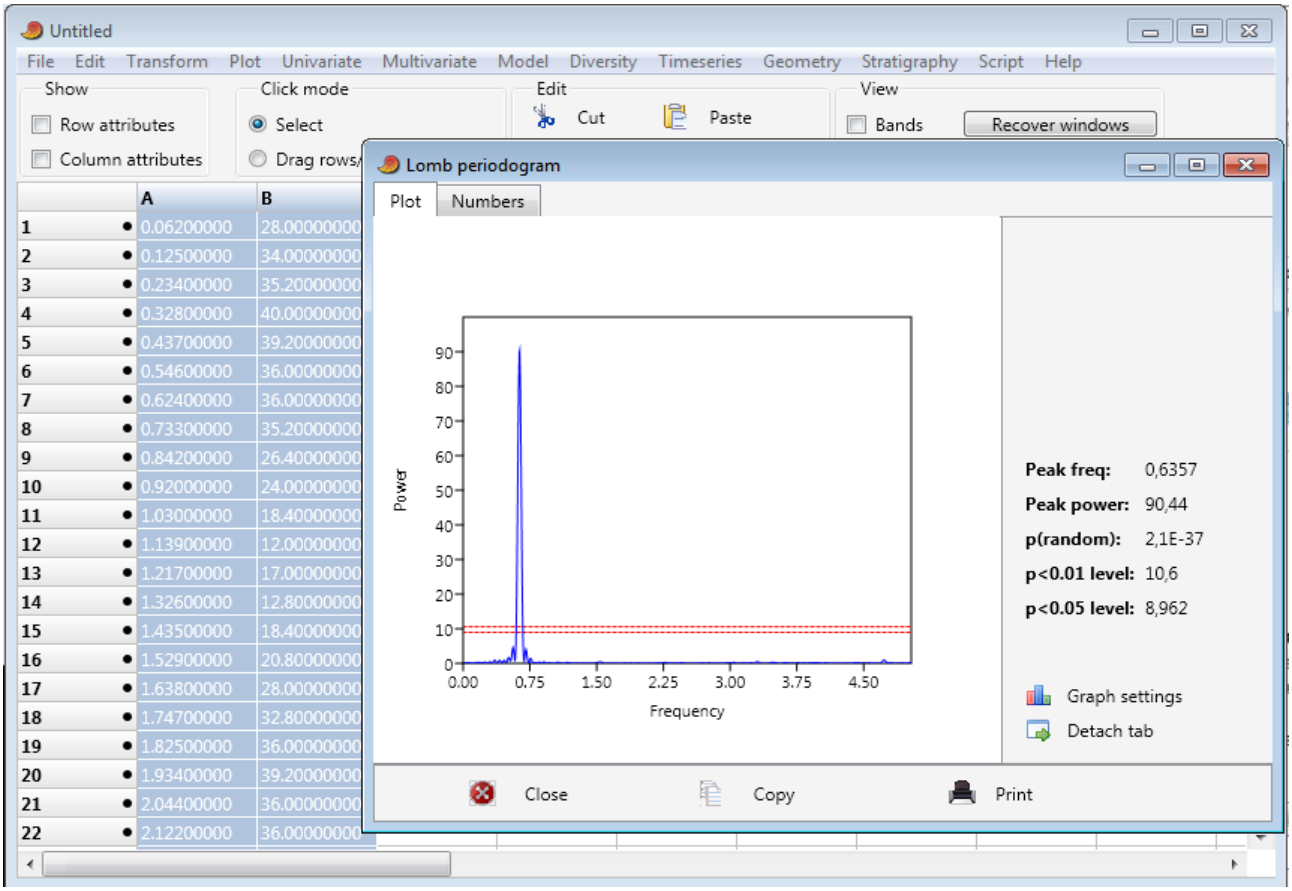


Illustration 5: Détermination de la fréquence du "pendule Micro:bit" par PAST

On voit que le pic de fréquence le plus important est à 0,6357 Hz, donc la période est 1/f, soit 1/0,6357 = 1,573 s, ce qui est proche de la valeur trouvée précédemment, 1,57 s. Comme ce pic est très pointu, on peut dire que la fréquence ne varie pas lors de l'amortissement.

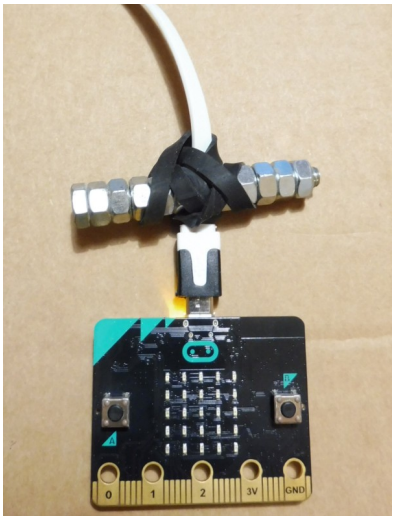


Illustration 6: Adjonction d'une masse (boulon avec nombreux écrous), pour plus de précision sur la longueur du pendule

Pour aller plus loin dans l'expérience du pendule

La carte Micro:bit est très légère, le câble USB aussi, et on ne sait pas très bien où est le centre de gravité du système. Pour essayer de voir que la période est proportionnelle à la racine carrée de la longueur, il faut savoir précisément la longueur du pendule, donc connaître la position du centre de gravité.

On peut fixer une masse assez lourde à l'extrémité du pendule (par exemple un gros boulon fixé à l'extrémité du câble USB, près de la carte Micro:bit). Comme cette masse sera beaucoup plus lourde que l'ensemble (câble + carte), on pourra considérer que la longueur du pendule est la distance entre le bord de la table et la masse ajoutée.

Détermination de g

A partir de la relation $T = 2\pi\sqrt{(\frac{l}{g})}$ on peut calculer g = l * 4* π^2*f^2. Ici, avec une longueur de 0,65 m et une période de 1,57 s, on obtient g=10,37 $m.s^{-2}$; cette valeur est imprécise à cause notamment de l'imprécision sur la longueur du pendule (où est précisément le centre de gravité ? bien que léger par rapport à la masse ajoutée, le câble USB est pesant, de même que la carte Micro:bit).

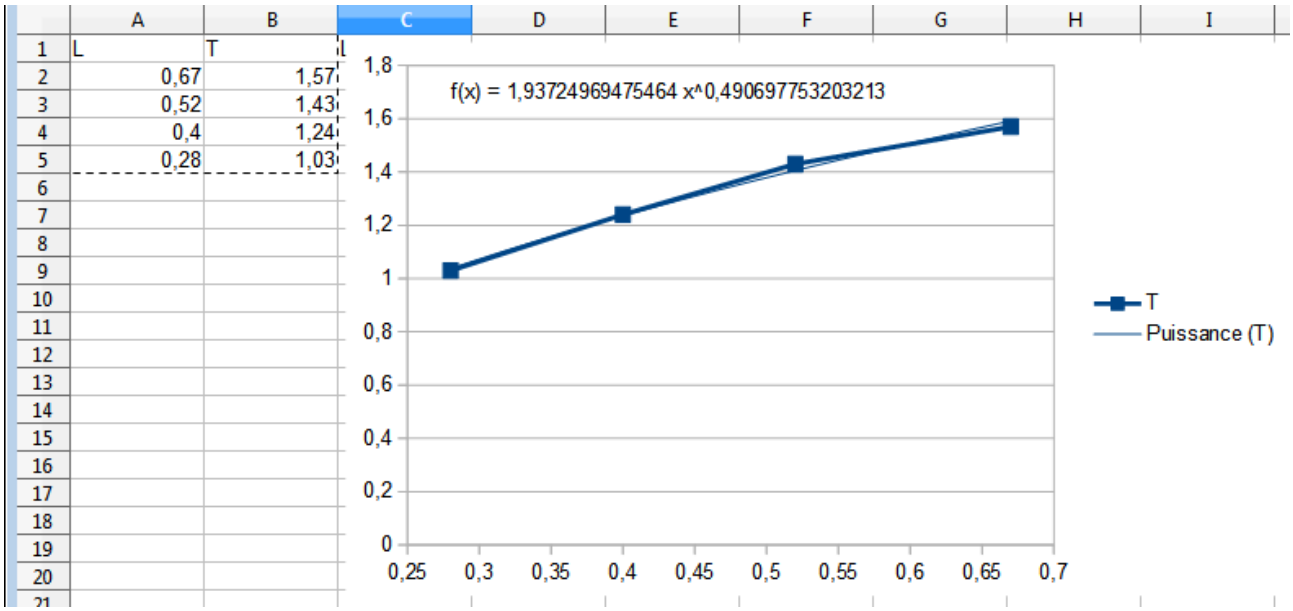
Relation entre période et longueur du pendule

On peut constater assez facilement que plus le pendule est long, plus grande est la période. Mais est-ce que la période est proportionnelle à la longueur du pendule, ou bien est-ce une relation non proportionnelle ?

On peut faire varier la longueur du pendule (= la longueur du câble USB entre la carte Micro:bit et l'axe de rotation). Un exemple de résultats est donné ci-dessous.

L	T
0,67	1,57
0,52	1,43
0,4	1,24
0,28	1,03

La modélisation par un tableur permet de constater que lorsqu'on demande la modélisation par la fonction puissance (Y = a*x^b), la meilleure approximation donne une puissance d'environ 0,5 (le coefficient b précédent vaut 0,5), ce qui signifie que la période est proportionnelle à la racine carrée de la longueur du pendule.



Le coefficient a précédent vaut ici 1,937.

Dans la relation $T = 2\pi\sqrt{(\frac{l}{g})}$ c'est la valeur de $2\frac{\pi}{\sqrt{(g)}}$, qui vaut effectivement environ 2.