Fabrication de la mascotte SEIS Junior

Wulfran Fortin, professeur agrégé de physique, lycée Jean Mermoz, Saint Louis, France.

Résumé
Une maquette simplifiée mais fonctionnelle du sismographe SEIS est fabriquée à partir d'éléments simples à trouver dans un magasin de bricolage. Cette maquette est capable de détecter des chocs sur une table en bois grâce à une capsule piézoélectrique fixée dans un des trois pieds. L'acquisition du signal peut se faire via des dispositifs scolaires (centrales d'acquisition de lycée) ou grâce à une électronique dédiée (pré amplification, platine Arduino et logiciel de présentation des données). La décoration de cette maquette lui donne un aspect très charmant et enfantin, qui attire spontanément petits et grands lors d'expositions (Fête de la Science, Portes ouvertes, etc. ...) et permet d'avoir une excellente accroche pour présenter les principes de l'étude de la sismicité.

1. L'expérience SEIS à bord d' InSight

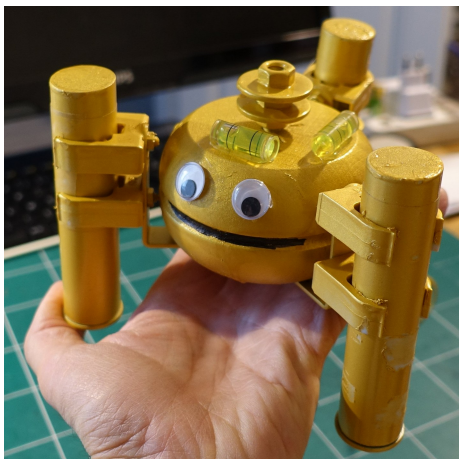
L'expérience SEIS (*Seismic Experiment for Interior Structure*) emportée par InSight (*Interior Exploration Using Seismic Investigations, Geodesy and Heat Transport*) est un petit instrument d'environ 40 cm de diamètre et 30 cm de haut. Il se compose notamment de trois sismomètres à longue période fabriqués par l'Institut de Physique du Globe de Paris (IPGP)[1].

Le sismomètre martien SEIS en salle blanche au Centre spatial de Toulouse. (Crédits : © CNES/GRIMAULT Emmanuel, 2015)

Cette expérience est fondamentale pour comprendre et décrire précisément la structure interne de la planète Mars en analysant la propagation des ondes matérielles (ondes sismiques) à travers la planète qui auront pour origines les impacts de météorites à sa surface.

2. La maquette SEIS Junior

Si l'importance de l'analyse des ondes sismiques de Mars est évidente pour le milieu de la recherche en planétologie, pour le grand public et pour les scolaires, la vue d'un sismogramme ne déclenchera pas en général un enthousiasme débordant chez le néophyte à cause du coté aride et austère d'un simple graphe en zigzag.



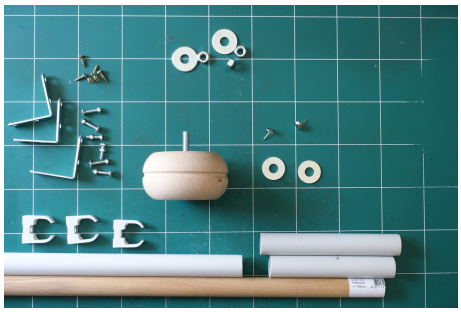
Maquette de SEIS Junior

Il convient donc d'associer la représentation d'un sismogramme à une émotion positive et chaleureuse pour permettre au commun des mortels de s'enthousiasmer pour la mission et déclencher l'envie d'en savoir plus.

Le rôle de cette maquette est donc de faciliter le contact entre le grand public et la recherche en planétologie, et principalement chez les enfants.

3. Matériel utilisé

La photo ci contre décrit les principaux matériaux utilisés pour la confection de la maquette.



Matériaux utilisés pour la fabrication du corps de SEIS Junior

Le corps de SEIS est fait à partir d'une boule de pied de lit d'environ 9 cm de diamètre.

Les trois pieds du sismographe sont découpés dans de la gaine électrique (tube IRL 25) et dans des tourillons en bois (diamètre 22 mm).

Trois équerres métalliques et des supports pour la gaine électrique (6 supports) permettent de fixer les pieds.

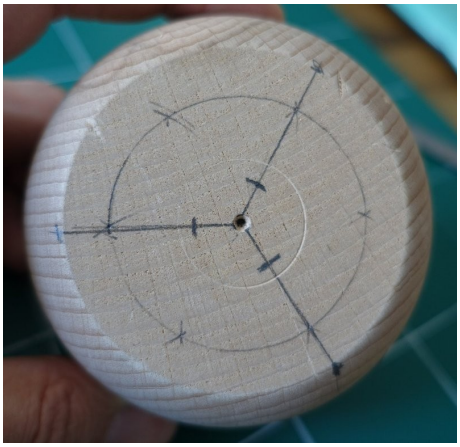
Des rondelles en acier de diamètre 26 mm et des clous à tête ronde de tapissier seront les contacts avec le sol sous les pieds.

Pour la décoration, les yeux ont été achetés dans un magasin d'arts décoratifs et les deux niveaux à bulle furent récupérés dans des outils bas de gamme.

Un des trois pieds est équipé d'un capteur piézoélectrique (disque buzzer de 27 mm de diamètre) soudé en parallèle avec une résistance de 1 MΩ.

4. Fabrication de SEIS Junior

On fixe les trois équerres sous la boule de lit de façon symétrique à 120 ° l'une de l'autre. On tracera au compas les deux cercles permettant le repérage du perçage des trous de vis, ainsi que la découpe d'un cercle en trois secteurs de 120°.



Repérage des perçages

Les trois pieds sont découpés dans de la gaine IRL 25. Chaque tronçon de tube a une longueur de 11 cm. Deux de ces tubes pourront être pleins, on collera à l'intérieur un morceau de tourillon en bois de 22 mm de diamètre et 11 cm de long. À la base de deux pieds, on colle une rondelle de 26 mm de diamètre et un clou de tapisser à tête ronde.

Fabrication de deux pieds de SEIS Junior

Sur trois équerres, on fixe six supports de tube de gaine IRL 25.

Il faudra éventuellement retailler un peu le plastique du support afin qu'il soit bien en contact à plat sur la patte en acier de l'équerre.

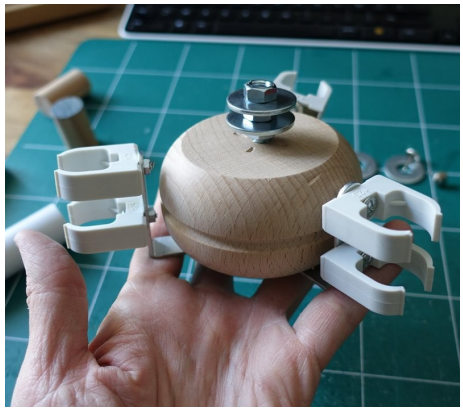
La fixation se fait avec un boulon et un écrou, éventuellement, on ajoute une rondelle pour avoir un serrage uniforme sur le plastique.

Fixation des pieds de SEIS Junior

Les trois équerres seront ensuite vissées sous le pied de lit.

Fixation des supports sur le corps de la maquette

On ajoute deux rondelles et trois écrous pour représenter le crochet au sommet du sismographe qui permet au bras robot d'InSight de saisir et déposer le sismographe sur le sol martien.

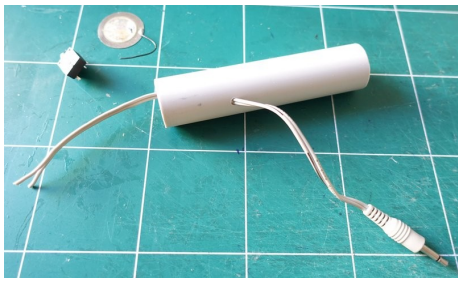


Le corps de SEIS Junior et les trois supports de pieds

La fabrication du troisième pied est un peu plus longue, car on y place un capteur piézoélectrique destiné à détecter les vibrations du support sur lequel sera posé SEIS Junior. Le capteur piézoélectrique est soudé en parallèle avec une résistance de 1 MΩ .

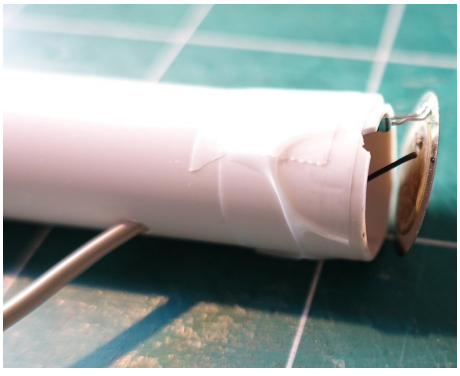
Capteur piézoélectrique et résistance de 1 MΩ

Un câble permet de récupérer la tension électrique générée lors de la déformation de la capsule piézoélectrique.



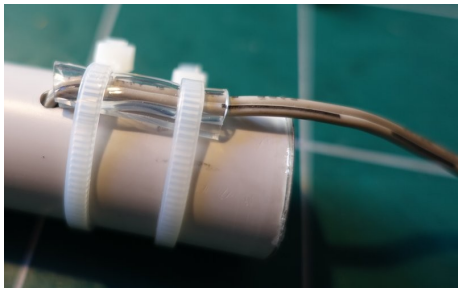
Le câble qui sera connecté au capteur

On colle soigneusement le bord de la capsule sur le bord du tube plastique, après avoir soudé la résistance sur la capsule et le câble de connexion.



Collage de la capsule piézoélectrique sur un pied de SEIS Junior

Ce câble de connexion sera fermement fixé au pied pour éviter qu'en le manipulant, on arrache les soudures sur la capsule piézoélectrique.

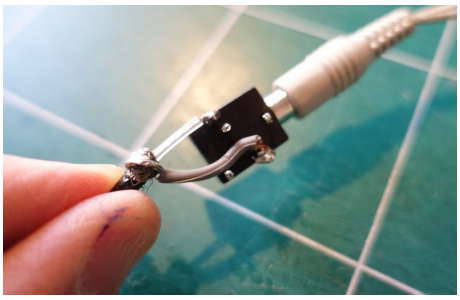


Fixation du câble de connexion sur la troisième jambe de SEIS Junior

Enfin, on colle sous la capsule une tête ronde de clou de tapissier pour permette un contact avec le sol sur la partie centrale du disque piézoélectrique, ce qui permet d'augmenter sa déformation lors des vibrations.

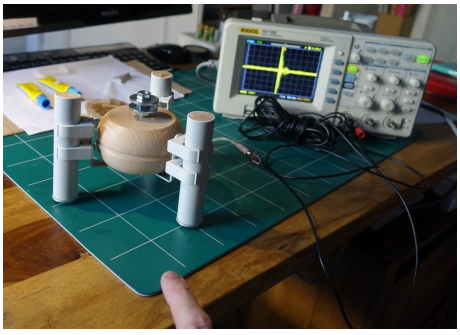
Finition du pied sous le capteur

Pour mesurer la tension variable générée par le disque piézoélectrique lors de la détection d'une vibration dans le pied, on relie ce capteur à un instrument de mesure (oscilloscope numérique dans l'exemple suivant) à l'aide d'un câble coaxial (récupération au laboratoire de physique). Ce câble est blindé, et donc on ne capte pas le rayonnement parasite à 50Hz du réseau électrique domestique.



Câble coaxial relié au capteur piézoélectrique

On peut alors assembler les trois pieds sur le corps de l'instrument SEIS Junior et effectuer un premier test du capteur piézoélectrique en tapotant la surface d'une table en bois.



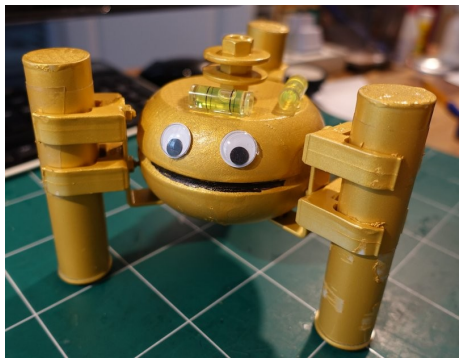
Test du capteur avant mise en peinture

Quand on est sur du bon fonctionnement électrique du capteur, on peut passer à la mise en peinture et à la décoration de la mascotte SEIS Junior.

On peut ainsi lui ajouter des yeux et un sourire, ainsi que deux niveaux à bulle pour rappeler la nécessité de la station horizontale du sismomètre et la présence sur le vrai instrument de trois pieds de longueur variable destinés à réaliser le nivellement de l'instrument.

La mascotte SEIS Junior est prête à être utilisée, il suffit de la relier par exemple, à une centrale d'acquisition de donnée disponible dans tous les lycées français.

On peut aussi développer sa propre interface d'acquisition du signal, c'est le but de la prochaine section.



Le détecteur SEIS Junior décoré

5. Acquisition de données, mise en forme du signal

La capsule piézoélectrique va fournir une tension variable qui pourra atteindre une dizaine de volts (positifs et négatifs). On ne peut donc pas la relier directement à une platine Arduino qui n'accepte que des tensions comprises entre 0 et 5 volts.

De plus, les tensions sont faibles pour les petits chocs sur la table, et donc on ne les détecterait pas bien.

On va donc fabriquer un circuit de protection et d 'amplification pour numériser le signal sismique avec une Arduino.

La protection se fait avec deux diodes en parallèles, montées tête bêche. Toute tension supérieure en valeur absolue à 0,8 volt (tension de seuil des diodes) ne sera pas transmise à la suite du circuit.

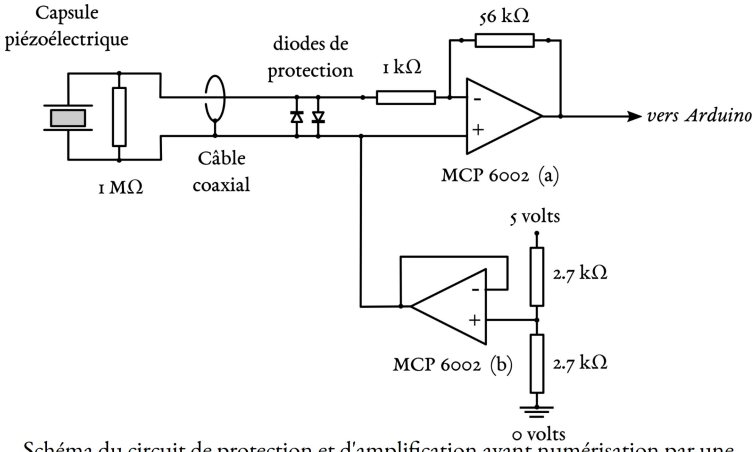


Schéma du circuit de protection et d'amplification avant numérisation par une platine Arduino

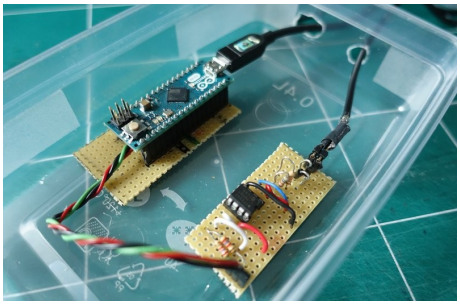
Ensuite, on a un montage avec deux amplificateurs opérationnels alimentés en 5V par la platine Arduino.

Ce sont des MCP6002 de Microchip qui fournit gratuitement des échantillons aux enseignants [2].

Un premier AOP permet d'avoir une masse virtuelle à 2,5 volts par rapport à laquelle on va mesurer le signal électrique de la capsule piézoélectrique.

Le deuxième AOP est monté en amplificateur non inverseur avec un gain de 50 environ, il amplifie le signal mesuré autour de 2,5 volts qui correspond à un signal nul, c'est la référence.

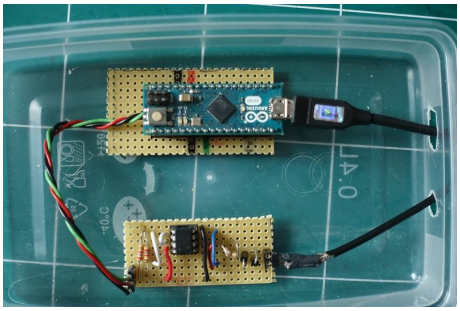
La platine Arduino pourra ainsi numériser un signal compris entre 0 et 5 volts, 2,5 volts correspondant au zéro du signal sismique.



Exemple d'intégration du montage avec une Arduino Micro

Le montage électronique a été réalisé sur une plaquette d'essais à bandes de cuivre. La photo suivante montre un exemple d'intégration du montage dans une boite de transport, avec une Arduino Micro. On y voit le câble USB qui permet l'alimentation en 5 volts du montage et la transmission des données vers un ordinateur, ainsi que le câble coaxial qui apporte le faible signal du capteur qui sera amplifié. Le composant MCP6002 contient les deux AOP du circuit.

Le câble qui relie l'Arduino à l'amplification fournit le 5 volts (rouge), la masse (noir) et récupère la tension à numériser entre 0 et 5 volts (vert).



Vue verticale du montage

6. Acquisition de données, code source Arduino

L'acquisition est très simple, un point de mesure toutes les 2 ms, envoi vers le port série à 19200 bauds.

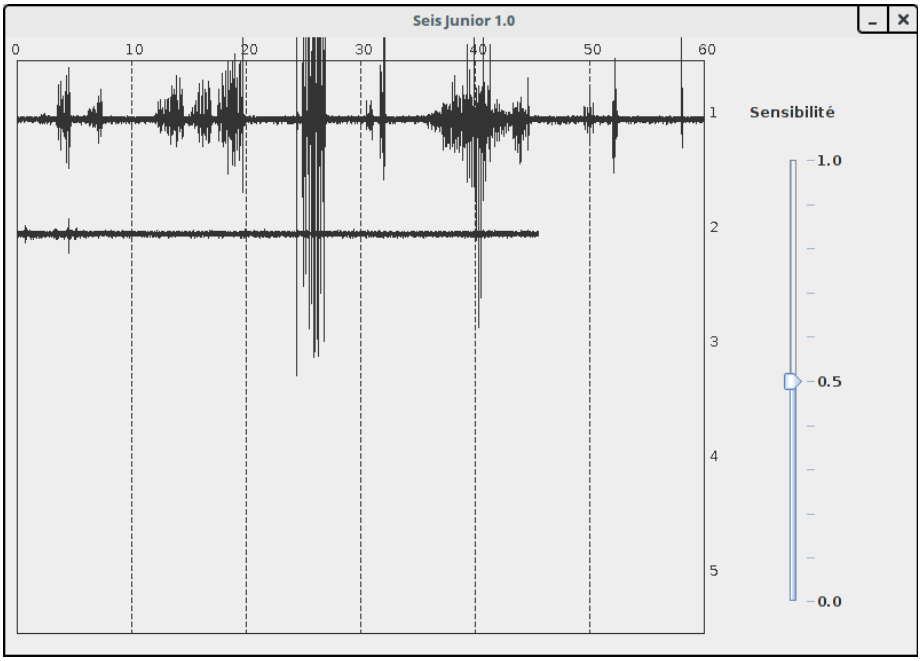
```
/*
 *
 *      SEIS Junior
 *
 */

//*************************************  VARIABLES
int analogPin = 5;
int val = 0;

//*************************************   PARAMETRAGE et INITIALISATION
void setup() {
  Serial.begin(19200);
  while (!Serial) ; // nécessaire pour Arduino Micro - voir doc officielle
}
//*************************************   BOUCLE PRINCIPALE
void loop() {
  val = analogRead(analogPin);    // lecture du signal toutes les 2ms
  Serial.println(val);
  delay(2);
}
```

7. Acquisition de données, affichage sur un écran de PC

Selon les compétences en informatique disponibles, on choisira un langage de programmation pour faire l'acquisition et l'affichage du signal en temps réel. On peut utiliser Python. Dans mon cas, j'utilise Java.



Il y a deux classes `SeisJunior` qui est l'interface graphique principale et `Graphe` qui permet d'afficher les sismogrammes. On stocke les données dans une table et toutes les 100 mesures, on met à jour le graphe. On évite ainsi une saturation du microprocesseur car le flux de données est assez rapide et l'affichage ne pourrait pas se rafraîchir à une telle cadence.

```
package fr.wfn.seis.junior;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Hashtable;
import java.util.List;
import javax.swing.JFrame;
import javax.swing.JLabel;
```

```
import javax.swing.JSlider;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

import jssc.*;

public class SeisJunior extends JFrame implements SerialPortEventListener, ChangeListener {
	private static final long serialVersionUID = 1L;
	static SerialPort serialPort;

	private int MAX_DATA_TIME = 300000;
	private double t_0 = 0.0;
	private double t = 0.0;
	private double d = 0.0;
	private int COUNTER = 0;
	private String dataSerie = "";
	private List datas = Collections.synchronizedList(new ArrayList());
	private Graphe graphe = null;
	private JSlider echelle = null;

	public SeisJunior(String port) {
		super.setTitle("Seis Junior 1.0");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		this.setLayout(null);

		graphe = new Graphe(datas);
		graphe.setBounds(0, 0, 650, 580);
		this.add(graphe);

		echelle = new JSlider(JSlider.VERTICAL, 0, 100, 50);
		echelle.setBounds(680, 100, 50, 400);
		echelle.addChangeListener(this);
		Hashtable labelTable = new Hashtable();
		labelTable.put(new Integer(0), new JLabel("0.0"));
		labelTable.put(new Integer(100), new JLabel("1.0"));
		labelTable.put(new Integer(50), new JLabel("0.5"));
		echelle.setLabelTable(labelTable);
		echelle.setPaintLabels(true);
		echelle.setMajorTickSpacing(10);
		echelle.setPaintTicks(true);
		this.add(echelle);
		graphe.setEchelle(echelle.getValue() / 100.0);

		JLabel sensibilite = new JLabel("Sensibilité");
		sensibilite.setBounds(650, 50, 150, 30);
		this.add(sensibilite);

		t_0 = System.currentTimeMillis();

		this.setSize(800, 570);
		this.setResizable(false);
		this.setVisible(true);
		this.connectionSEIS(port);

	}

	private void connectionSEIS(String port) {
		serialPort = new SerialPort(port);
		try {
			serialPort.openPort();
			serialPort.setParams(SerialPort.BAUDRATE_19200, SerialPort.DATABITS_8,
SerialPort.STOPBITS_1,
					SerialPort.PARITY_NONE);
			serialPort.setFlowControlMode(SerialPort.FLOWCONTROL_RTSCTS_IN |
SerialPort.FLOWCONTROL_RTSCTS_OUT);
			serialPort.addEventListener(this, SerialPort.MASK_RXCHAR);
		} catch (SerialPortException ex) {
			System.out.println("Problème de connection sur le port série : " + ex);
			System.exit(0);
		}
	}

	private void nouvelleValeur(String v) {
		try {
```

```
                        int val = Integer.parseInt(v);
                        COUNTER++;
                        t = System.currentTimeMillis() - t_0;
                        d = new Double(val);

                        d = d - 512;

                        double[] p = { d, t };
                        datas.add(p);

                        if (COUNTER > 100) {
                              COUNTER = 0;
                              graphe.nouvelleValeurs();
                        }

                        if ((System.currentTimeMillis() - t_0) > MAX_DATA_TIME) {
                              datas.clear();
                              t_0 = System.currentTimeMillis();
                        }

                  } catch (NumberFormatException e) {
                        System.out.println(e);
                  }

            }

            @Override
            public void serialEvent(SerialPortEvent e) {
                  if (e.isRXCHAR()) {
                        try {
                              int nbrCar = e.getEventValue();
                              for (int i = 0; i < nbrCar; i++) {
                                    String receivedData = serialPort.readString(1);
                                    if (receivedData.equals("\n") || receivedData.equals("\r")) {
                                          if (dataSerie.length() > 0) {
                                                nouvelleValeur(dataSerie);
                                                dataSerie = "";
                                          }
                                    } else {
                                          dataSerie = dataSerie + receivedData;
                                    }

                              }

                        } catch (SerialPortException ex) {
                              System.out.println("Error in receiving string from COM-port: " +
ex);
                        }
                  }

            }

            public static void main(String[] args) {
                  if (args != null) {
                        String port = args[0];
                        new SeisJunior(port);
                  } else {
                        System.out.println("Spécifier le port série de connection dans la ligne de
commande !");
                  }
            }

            @Override
            public void stateChanged(ChangeEvent e) {
                  JSlider source = (JSlider) e.getSource();
                  if (!source.getValueIsAdjusting()) {
                        int ech = (int) source.getValue();
                        graphe.setEchelle((double) (ech) / 100.0);
                  }
            }
}
```

Classe correspondant à l'affichage dynamique du graphique en temps réel. Une partie du code gérant l'affichage est protégée des accès concurrents sur l'objet `datas` par l'instruction `synchronized`. Cette précaution est importante à respecter, elle devra aussi être faite en Python ou tout autre langage de programmation.

```
package fr.wfn.seis.junior;

import java.awt.BasicStroke;
import java.awt.Stroke;
import java.awt.geom.Line2D;
import java.awt.Graphics;
import java.awt.Graphics2D;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import javax.swing.JPanel;

public class Graphe extends JPanel {
	private static final long serialVersionUID = 1L;
	private List datas = null;
	private int MAX_NEW_POINTS = 10;

	private int X_0 = 10;
	private int Y_0 = 20;
	private double x, y;

	private double scale = 0.2;
	private double offsetY = 100;
	private double LIGNE_X = 600.0;

	final static float dash1[] = { 5.0f, 2.0f };
	final static BasicStroke dashed = new BasicStroke(1.0f, BasicStroke.CAP_BUTT,
BasicStroke.JOIN_MITER, 10.0f, dash1,
			0.0f);
	final static BasicStroke solid = new BasicStroke(1.0f);

	public Graphe(List l) {
		datas = l;
	}

	public void setEchelle(double s){
		scale = s;
	}
	public void paintComponent(Graphics g) {
		super.paintComponent(g);
		Graphics2D g2 = (Graphics2D) g;

		g2.setStroke(dashed);
		for (int i = 0; i < 7; i++) {
			g2.draw(new Line2D.Double(X_0 + LIGNE_X * i / 6, Y_0, X_0 + LIGNE_X * i /
6, Y_0 + offsetY * 5));
			g2.drawString(Integer.toString(i * 10), (int) (X_0 + LIGNE_X * i / 6 - 5),
(int) Y_0 - 5);
		}
		g2.setStroke(solid);
		g2.draw(new Line2D.Double(X_0, Y_0, X_0, Y_0 + offsetY * 5));
		g2.draw(new Line2D.Double(LIGNE_X + X_0, Y_0, LIGNE_X + X_0, Y_0 + offsetY * 5));
		g2.draw(new Line2D.Double(X_0, Y_0, X_0 + LIGNE_X, Y_0));
		g2.draw(new Line2D.Double(X_0, Y_0 + offsetY * 5, X_0 + LIGNE_X, Y_0 + offsetY *
5));

		for (int i = 1; i < 6; i++) {
			g2.drawString(Integer.toString(i), (int) (X_0 + LIGNE_X + 5), (int) (Y_0 +
offsetY * (i - 0.5)));
		}

		double x = -1.0;
		double x_i = -1.0;
		double y = 0.0;
		int X = 0;
		int Y = 0;
		synchronized (datas) {
```

```
                        double[] p = null;
                        if (datas.size() > 0)
                              p = (double[]) datas.get(0);
                        if (p != null) {
                              x_i = p[1] / 100;
                        }

                        Iterator i = datas.iterator();
                        int Xprev = X_0;
                        int Yprev = Y_0;
                        while (i.hasNext()) {
                              p = (double[]) i.next();
                              x = p[1] / 100 - x_i;
                              y = p[0];

                              X = X_0 + (int) (x - ((int) (x / LIGNE_X)) * LIGNE_X);
                              Y = Y_0 + (int) (offsetY * ((int) (x / LIGNE_X) + 0.5) - y * scale);
                              if(Xprev - X < 10){
                                    g2.drawLine(Xprev,Yprev,X,Y);
                              }
                              Xprev = X;
                              Yprev = Y;
                        }
                  }

            }

            public void nouvelleValeurs() {
                  this.repaint();
            }

      }
```

8. Bibliographie

[1] http://www.ipgp.fr/fr/pss/insight

[2] http://ww1.microchip.com/downloads/en/DeviceDoc/21733j.pdf