ANNEXE

CODE ARDUINO (Programme qui permet la rotation du moteur en fonction des instructions , et envoie les données du capteur en fonction du temps en ms

```
// bibliothèque pour gérer le shield et les moteurs
#include "AFMotor.h"


// quelques constantes
#define MIN_PWM 0
#define MAX_PWM 255

#define NB_VAL 20

// code du retour à la ligne
#define LF 10

// création de l'objet permettant de gérer le moteur sur le shield
// (via bibliothèque)
AF_DCMotor moteur(3, MOTOR12_1KHZ);

// variable pour stocker la vitesse
int vitesse = 0;


/*
 * Cette fonction est exécutée une fois à l'initialisation du code
 */
void setup() {
  // commencer la communiquation avec l'ordinateur
  Serial.begin(9600);
}


/*
 * Cette fonction est exécutée ensuite en boucle infinie
 */
void loop() {
  // on dit au moteur de tourner
  moteur.run(FORWARD);

  // si on a reçu des valeures pour la vitesse du moteur, on les récupères
  if(Serial.available()>0){
    vitesse = Serial.parseInt();
  }

  // lire les valeures en provenance du capteur
  int value = analogRead(A5);

  // envoyer les valeures du temps en millisecondes ( fonction millis() ) et du
capteur
  Serial.print(millis());
  Serial.print(",");
  Serial.println(value);

  delay(1); // attendre un peu l'acquisition de la valeure du capteur

  // régler la vitesse du moteur
  moteur.setSpeed(vitesse);
}
```

ANNEXE

Code divisé en 2 parties : La partie 1 permet de communiquer avec l'ARDUINO et de dessiner la courbe, et la partie 2 est la fenêtre de contrôle permettant d'envoyer les instructions du moteur et des acquisitions

PARTIE 1 :

```
import javax.swing.*;
import java.awt.GridLayout;
import java.awt.FlowLayout;
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import processing.serial.*;
import java.util.*;
import java.io.*;
import processing.core.*;

/* ========================================================================================== *
 * Bibliothèques & Classes :                                                                  *
 *  - javax.swing : Graphiques de a fenêtre de contrôles                                      *
 *  - java.awt.GridLayout : Disposition des graphiques en grille sur la fenêtre               *
 *  - java.awt.event.ActionEvent et Listener : gestion des événements de cette fenêtre        *
 *  - javax.swing.event.ChangeEvent et Listener : pareil pour d'autres sortes d'évènements   *
 *  - processing.serial : communication avec Arduino                                          *
 *  - java.util.* : pour obtenir des liste extensible.                                        *
 *  - processing.core.* : permettra d'obtenir des boîtes de dialogues pour l'enregistrement  *
 * ========================================================================================== */

Serial myPort; // communication avec Arduino
ArrayList<Integer> yPos; // liste des ordonnées des point de l'acquisition
ArrayList<Integer> xPos;// et de l'abcisse du temps

//ATTENTION avec ceci: (c'est le numéro de l'élément de la liste des ports USB branchés sur l'
ordinateur)
short portIndex=0; // 0 est le premier port

// constantes de communication depuis l'arduino vers l'ordinateur (pour les graphiques)
final char DEBUT = 'D';
final char VAL   = 'V';

final char COMMENT = 'K';

// constantes de communication depuis l'ordinateur vers l'arduino (pour changer les valeurs de
 l'arduino)
final char RUN     = 'R';
final char SLOW    = 'S';
final char BACK    = 'B';
final char ACQUIRE = 'A';
final char STOP    = 'X';

final short LF = 10; // retour à la ligne (linux et windows) = "\n"
final short CR = '\r'; // retour à la ligne suite (windows et mac) = "\r"

int yPosition = 1; // valeur reçue de l'arduino depuis le capteur de force.
int serialValue; //valeur convertie en ordonnée pour le graphique
int xPosition = 1; // valeur reçue de l'arduino depuis le compteur de temps.
int serialValue2; //valeur convertie en abscisse pour le graphique
// note : le compteur de temps peux aussi se faire dans processing.
// À voir où le faire.

long compteur = 0; //compteur pour ralentir les graphiques
int vitesseDiminuePar = 1; // en changeant une fois sur <5>

FenControle fen; // fenêtre des graphiques (non-initialisée ici)

// mémorisation des options graphiques
boolean grilleEnabled = false; //grille affichée ?
boolean grilleMovementEnabled = false; // grille en mouvement ?
int grilleSize = 64; //taille de la grille (en pixels)

boolean acquisition = false;
boolean defilement = true;

int leMin = 0;
```

Glisser sur du sable

```
void setup(){ // fait une fois au début
  size(700,700); //taille de la fenêtre par défaut
  surface.setTitle("Graphique de la force en fonction du temps");
  surface.setResizable(true);

  int nbPort = 1;
  boolean continuer = false;

  background(0);
  color(255);
  text("En attente de branchement d'une carte Arduino", 20, 20);

  // connection à l'arduino :
  do{ // on répète (en faisant au moins une fois) :
    nbPort = Serial.list().length;
    if (nbPort!=0){
      println("liste des ports :");
      println((Object[]) Serial.list()); // liste des ports USB branchés
    }

    try{ //esssayer de se connecter à l'arduino
      println("Connexion à ->" + Serial.list()[portIndex]);
      myPort = new Serial(this, Serial.list()[portIndex], 9600); // vitesse de transmission
des donnée de 9600 bauds

      yPos = new ArrayList<Integer>(); //initialisation des listes des ordonnées des point
d'acquisition
      xPos = new ArrayList<Integer>();
      continuer = true; // aller à la suite

    }catch (Exception e){ // en cas d'erreur : recommencer (instruction "continuer=false;")
     // e.printStackTrace();
      //println("Nouvel essai dans 10? secondes...");
      continuer = false;
      //delay(10000);
    }

  } while (continuer==false); // boucler

  println("Connexion établie avec Arduino"); //connexion réussie !
  fen = new FenControle(this); // création de la fenêtre de contrôles
  delay(500);

  myPort.clear();
  myPort.readStringUntil(LF);
}

void draw() { //répété en boucle
  while (myPort.available()>0){ // y a-t-il des données en provenance de l'arduino ?

    String message = myPort.readStringUntil(LF); //lire ces données jusqu'au retour à la
ligne.
    if (message != null && defilement){ // si il y a un message
      print(message); // on l'affiche dans la console

      String [] data = message.split(","); // on le découpe au niveau des virgules (",")
      try{
        if (data[0].charAt(0) == DEBUT){ // si c'est le message du début
            println("Programme Arduino (ré-)initialisé.");
            synchronized (xPos) {
              while(!xPos.isEmpty()){
                xPos.remove(0);
                yPos.remove(0);
              }
            }
        }
      }catch (Exception e) {
        e.printStackTrace();
      }

      try{
        if ((data.length>1) && defilement){ // si c'est une valeur de temps et de force
          try { // on essaye de l'obtenir
            int temps = Integer.parseInt(data[0]); // conversion en nombres
            int force = Integer.parseInt(data[1]);

            serialValue2= height-(int)map(force, 0, 1023, 0, width); // puis en ordonnées
            serialValue = height-(int)map(temps, 0, 1023, 0, width);  // et abcisse
d'affichage
```

```
            xPos.add(temps);
            yPos.add(force);

            println("Valeur :"+temps+" ,"+force);//affichage dans la console

          }catch (Exception e) { //en cas d'erreurs:
            e.printStackTrace(); // on affiche l'erreur pour debug
          }
        }
      }catch ( Exception e) { //en cas d'erreurs:
        //e.printStackTrace();// on affiche l'erreur // pas fait
      }
    }
  }

  compteur++; // incrémentation du compteur "de tours"
  if (compteur % vitesseDiminuePar == 0 && defilement){ // faire bouger la courbe 1 fois sur
<1>

    /* dessins */
    background(0); //fond en noir (efface tout ce qui est déjà dessiné au passage)

    /* Grille */
    if (grilleEnabled){ // si grille affichée
      stroke(0, 0, 255); // dessiner en bleu

      long nombrePxl = 0;
      if (grilleMovementEnabled){ // si en mouvement
        nombrePxl = compteur / vitesseDiminuePar;
      }

      // dessin de la grille
      for(int i = 0; i < width; i++) {
        if ((i + nombrePxl) % grilleSize == 0){
          line(i, 0, i, height); // traits verticaux
        }
      }

      for(int i =0; i<height; i+=grilleSize){
        line(0, i, width, i); // traits horizontaux
      }
    }

    /* Traits force TODO en fonction du temps */
    stroke(255,255,0); // jaune pour la force
    strokeWeight(1);

    float yPosPrev = 0, xPosPrev = 0;

    int leMax = leMin + width;

    synchronized(xPos){
      for(int i: xPos){
        if (i > leMax)
          leMax = i;
      }
    }

    synchronized(xPos){
      for(int n = 0; n < xPos.size(); n++){
        float leX, leY;
        leX = map(xPos.get(n), leMin, leMax, 0, width);
        leY = map(yPos.get(n), 0, height, height, 0);

        if (n>0){
          // nouvelle version (garde-t-on cela ?) : temps calculé dans Arduino.
          line(xPosPrev, yPosPrev, leX, leY); // on trace (chaque point est relié au précédent
pour ne pas faire de coupures)
        }
        xPosPrev = leX;
        yPosPrev = leY;
      }
    }
  }
}


/* Méthodes pour changer les valeurs des options graphiques ou d'arduino */
void setGrilleEnabled(boolean value){ //grille active ?
  grilleEnabled = value;
}
```

```
void setGrilleMovement(boolean value){ // grille en mouvement
  grilleMovementEnabled = value;
}

void setGrilleSize(int taille){ // taille de la grille
  grilleSize = taille;
}


void commencerAcquisition(){
  acquisition = true;
  defilement = true;

  for(int i: xPos){
    if (i > leMin)
      leMin = i;
  }

  synchronized (xPos) {
    while(!xPos.isEmpty()){
      xPos.remove(0);
      yPos.remove(0);
    }
  }
}

void stopperAcquisition(){
  acquisition = false;
  defilement = false;
}

void moteurLent(){
  myPort.write("220\n");
}

void moteurMarche(){
  println("moteur marche 2");
  myPort.write("255");
}

void moteurReculer(){
  myPort.write("-235");
}

void moteurStopper(){
  myPort.write("0");
}

void exporterCSV(){
  selectOutput("Exporter CSV", "exporterCSV"); // le 2e doit être le nom de la fonction qui va
gérer l'enregistrement (fonction ci-dessous).
}

void exporterCSV(File leChemin) {
  FileOutputStream lefichier=null;
  BufferedOutputStream text = null;
  try {
    lefichier = new FileOutputStream(leChemin);
    text = new BufferedOutputStream(lefichier);

    synchronized(xPos){
      for(int i = 0; i < xPos.size(); i++){
        String txt = xPos.get(i) + "," + yPos.get(i);

        for(int j = 0; j < txt.length(); j++){
          text.write(txt.charAt(j));
        }

        text.write(CR);
        text.write(LF);
      }
      text.flush();
      lefichier.flush();
    }

  } catch(IOException e) {
    e.printStackTrace();
  }

  finally {
```

```
    try{
      lefichier.close();
      text.close();
    }catch(Exception e){
      e.printStackTrace();
    }
  }
}

void reset(){
  acquisition = false;
  defilement = true;

  //leMin = xPos.get(xPos.size());
  for(int i: xPos){
    if (i > leMin)
      leMin = i;
  }
  synchronized (xPos) {
    while(!xPos.isEmpty()){
      xPos.remove(0);
      yPos.remove(0);
    }
  }
}
```

PARTIE 2 :

```
class FenControle implements ActionListener, ChangeListener {

  private processing_du_programme7 parent;

  private JFrame fen;

  private JPanel contenuFenetre;
  private FlowLayout disposition1;

  /* zone gestion des graphiques */
  private JPanel zoneGraphiques;
  private GridLayout dispositionGraphiques;

  /* zone grille*/
  private JPanel zoneGrille;
  private GridLayout dispoGrille;

  /*
   * La grille est géré par 2 cases à cocher :
   *   1 -> visible ?
   *   2 -> en mouvement ?
   */
  private JCheckBox grilleActive;
  private JCheckBox grilleActiveMouvement;

  /* géré par un Slider et/ou un Entry spinner */
  private JSlider tailleGrille;
  private JLabel labelTailleGrille;
  //private JSpinner tailleGrilleEntry; //(reste a créer)

  /* zone Arduino :
   *   dans cette zone, il y a :
   *     - Une zone pour démarrer stopper l'acquisition
   *     - Une zone pour gérer le moteur
   *     - Une zone pour exportations
   */
  private JPanel zoneArduino;
  private GridLayout dispoArduino;

  /* zone acquisition */
  private JPanel zoneArduinoAcquisition;
  private GridLayout dispoArduinoAcquisition;

  /* Dans cette zone, il y a :
   *  - Un Bouton pour commencer
   *  - Un Bouton pour stopper
   */
  private JButton arduinoBoutonCommencerAcq;
  private JButton arduinoBoutonStopperAcq;
  private JButton arduinoBoutonSResetAcq;
```

```
/* zone gérer moteur */
private JPanel zoneArduinoMoteur;
private GridLayout dispoArduinoMoteur;

/* Dans cette zone, il y a :
 *  - 4 boutons pour avancer, lentement, reculer, stopper
 *  - un Entry+Slider et un bouton pour changer précisément,
 *  - un Label avec la vitesse actuelle.
 */
private JButton boutonMoteurMarche;
private JButton boutonMoteurLent;
private JButton boutonMoteurStopper;
private JButton boutonMoteurReculer;

/* zone moteur précis */

private JPanel zoneMoteurPrecis;
private GridLayout dispoMoteurPrecis;
private JSpinner spinnerMoteurVitesse;
private JButton boutonMoteurVitesse;

/* zone exportation */
private JPanel zoneExportation;
private GridLayout dispoExportation;

private JButton boutonExportationCSV;
private JButton boutonExportationODS;
private JButton boutonExportationXLS;

/* Bordures de Cadres */
private FlowLayout dispoBorder;
private JPanel borderGraphique;
private JPanel borderArduino;

@SuppressWarnings("unused")
private JPanel borderMoteur;

public FenControle(processing_du_programme7 parent){

  this.parent = parent;

  // création du contenu de la fenêtre

  /* zone fenêtre */
  contenuFenetre = new JPanel();
  disposition1 = new FlowLayout();
  contenuFenetre.setLayout(disposition1);

  dispoBorder = new FlowLayout(FlowLayout.LEFT, 1, 1);

  /* zone Graphiques */
  zoneGraphiques = new JPanel();
  dispositionGraphiques = new GridLayout(1,1);
  zoneGraphiques.setLayout(dispositionGraphiques);

  borderGraphique = new JPanel();
  borderGraphique.setLayout(dispoBorder);
  borderGraphique.setBackground(new Color(0, 0, 0));

  borderGraphique.add(zoneGraphiques);
  contenuFenetre.add(borderGraphique);

  /* et zone Arduino */
  zoneArduino = new JPanel();
  dispoArduino = new GridLayout(2,1);
  zoneArduino.setLayout(dispoArduino);

  borderArduino = new JPanel();
  borderArduino.setLayout(dispoBorder);
  borderArduino.setBackground(new Color(0, 0, 0));

  borderArduino.add(zoneArduino);
  contenuFenetre.add(borderArduino);

  /* zone Graphiques/Grille */
  zoneGrille = new JPanel();
  dispoGrille= new GridLayout(2, 2);
  zoneGrille.setLayout(dispoGrille);

  grilleActive = new JCheckBox("Grille Active ?");
  grilleActiveMouvement = new JCheckBox("Grille en mouvement ?");
```

```
labelTailleGrille = new JLabel("Taille de la grille :");
tailleGrille = new JSlider(SwingConstants.HORIZONTAL, 16, 512, 64);

zoneGrille.add(grilleActive);
zoneGrille.add(grilleActiveMouvement);
zoneGrille.add(labelTailleGrille);
zoneGrille.add(tailleGrille);

grilleActive.addActionListener(this);
grilleActiveMouvement.addActionListener(this);
tailleGrille.addChangeListener(this);

zoneGraphiques.add(zoneGrille);

/* zone arduino */
/* zone arduino/Acquisition */
zoneArduinoAcquisition = new JPanel();
dispoArduinoAcquisition = new GridLayout(1, 3);
zoneArduinoAcquisition.setLayout(dispoArduinoAcquisition);

arduinoBoutonCommencerAcq = new JButton("Démarrer Acquisition");
arduinoBoutonStopperAcq = new JButton("Stopper Acquisition");
arduinoBoutonSResetAcq = new JButton("Reset");

zoneArduinoAcquisition.add(arduinoBoutonCommencerAcq);
zoneArduinoAcquisition.add(arduinoBoutonStopperAcq);
zoneArduinoAcquisition.add(arduinoBoutonSResetAcq);

arduinoBoutonCommencerAcq.addActionListener(this);
arduinoBoutonStopperAcq.addActionListener(this);
arduinoBoutonSResetAcq.addActionListener(this);

/* zone arduino/ contrôler le moteur */
/* 4 différents boutons d'activation */
zoneArduinoMoteur = new JPanel();
dispoArduinoMoteur = new GridLayout(2, 2);
zoneArduinoMoteur.setLayout(dispoArduinoMoteur);

boutonMoteurMarche  = new JButton("Activer le moteur");
boutonMoteurLent    = new JButton("Moteur lent");
boutonMoteurStopper = new JButton("Stopper le moteur");
boutonMoteurReculer = new JButton("Reculer lent");

zoneArduinoMoteur.add(boutonMoteurMarche);
zoneArduinoMoteur.add(boutonMoteurLent);
zoneArduinoMoteur.add(boutonMoteurStopper);
zoneArduinoMoteur.add(boutonMoteurReculer);

boutonMoteurMarche.addActionListener(this);
boutonMoteurLent.addActionListener(this);
boutonMoteurStopper.addActionListener(this);
boutonMoteurReculer.addActionListener(this);

boutonMoteurReculer.setEnabled(false);

/* zone arduino/exportation */
zoneExportation = new JPanel();
dispoExportation = new GridLayout(3, 1);
zoneExportation.setLayout(dispoExportation);

boutonExportationCSV = new JButton("Exporter CSV");
boutonExportationODS = new JButton("Exporter ODS");
boutonExportationXLS = new JButton("Exporter XLS");

zoneExportation.add(boutonExportationCSV);
zoneExportation.add(boutonExportationODS);
zoneExportation.add(boutonExportationXLS);

boutonExportationCSV.addActionListener(this);
boutonExportationODS.addActionListener(this);
boutonExportationXLS.addActionListener(this);

boutonExportationODS.setEnabled(false);
boutonExportationXLS.setEnabled(false);


/* zone Moteur précis */
zoneMoteurPrecis = new JPanel();
dispoMoteurPrecis = new GridLayout(2, 1);
zoneMoteurPrecis.setLayout(dispoMoteurPrecis);
```

```
    spinnerMoteurVitesse = new JSpinner(new SpinnerNumberModel(220, 0, 255, 1)); //val, min,
max, step
    boutonMoteurVitesse = new JButton("Changer la vitesse");

    boutonMoteurVitesse.addActionListener(this);

    boutonMoteurVitesse.setEnabled(false);

    zoneMoteurPrecis.add(spinnerMoteurVitesse);
    zoneMoteurPrecis.add(boutonMoteurVitesse);

    /* Ajout des zones */
    zoneArduino.add(zoneArduinoMoteur);
    zoneArduino.add(zoneArduinoAcquisition);
    zoneArduino.add(zoneMoteurPrecis);
    zoneArduino.add(zoneExportation);



    /* zone finale de la fenêtre */
    fen = new JFrame("Fenêtre de contrôles");
    fen.setContentPane(contenuFenetre);
    fen.pack();
    fen.setVisible(true);
    fen.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }

  public void actionPerformed(ActionEvent evenement) {

    if (evenement.getSource() instanceof JCheckBox){
      /* code des cases à cocher */
      JCheckBox item = (JCheckBox) evenement.getSource();
      /* pour les grilles */
      if (item==grilleActive) {
        parent.setGrilleEnabled(item.isSelected());

        if (item.isSelected()) {
          grilleActiveMouvement.setEnabled(true);
          tailleGrille.setEnabled(true);
          labelTailleGrille.setEnabled(true);
        }else {
          grilleActiveMouvement.setEnabled(false);
          tailleGrille.setEnabled(false);
          labelTailleGrille.setEnabled(false);
        }

      }else if (item==grilleActiveMouvement) {
        parent.setGrilleMovement(item.isSelected());

      }
      /* pout les boutons */
    } else if(evenement.getSource() instanceof JButton){
      JButton item = (JButton) evenement.getSource();

      /* arduino -> Acquisition */
      if (item==arduinoBoutonCommencerAcq) {
        parent.commencerAcquisition();
      }
      if (item==arduinoBoutonStopperAcq) {
        parent.stopperAcquisition();
      }
      if (item==boutonMoteurLent) {
        parent.moteurLent();
      }
      if (item==boutonMoteurMarche) {
        println("moteur marche 1");
        parent.moteurMarche();
      }
      if (item==boutonMoteurReculer) {
        parent.moteurReculer();
      }
      if (item==boutonMoteurStopper) {
        parent.moteurStopper();
      }
      if (item==boutonExportationCSV) {
        parent.exporterCSV();
      }
      if (item==arduinoBoutonSResetAcq) {
        parent.reset();
      }
    }
```

```
  }

  public void stateChanged(ChangeEvent evenement) {
    /* code du "Glisseur" */
    if (evenement.getSource() instanceof JSlider){
      JSlider item = (JSlider) evenement.getSource();

      if (item==tailleGrille){
        int val = item.getValue();
        parent.setGrilleSize(val);
      }
    }
  }
}
```

ANNEXE

Schéma du circuit utilisé avec le moteur, le capteur de force, et la carte ARDUINO