POLYTECH LILLE

DEPARTEMENT IMA
2011-2012

RAPPORT DE PROJET

Rédigé par

Ibrahima SOGOBA et Long ZHANG

R O B O T D E G U I D A G E

Tuteur de projet : M. Thomas Vantroys

Ecole Polytechnique universitaire de Lille (POLYTECH'LILLE)
Département : Informatique – Microélectronique – Automatique
4ème année

PLAN

1/ Présentation du projet

Ce projet s'inscrit dans le cadre de notre quatrième année d'étude au département IMA de l'école d'ingénieur Polytech'Lille. Il a été proposé par le département R&D de l'entreprise Oxylane dans le cadre d'une coopération avec le département. L'objectif de ce projet est de créer un robot de guidage qui devrait être utilisable dans un magasin quel qu'il soit. Ce robot devrait être capable de se localiser dans un magasin suivant les produits recherchés, le principe étant que le client puisse sélectionner les produits sur une tablette tactile associée au robot. Ce dernier commandé par un contrôleur Arduino relié à la tablette devrait utiliser des points de repères afin de déterminer l'emplacement de chaque rayon. L'ensemble du matériel nécessaire (robot, tablette, capteur etc...) à ce projet sont commandés via des fournisseurs et sur proposition des étudiants au tuteur de projet. Les étapes de réalisation de ce projet sont donc tributaires du matériel mis à disposition au

Matériel requis

- Robot + capteurs
- Tablette
- Arduino
- Outils de programmation

Matériel à acheter

- Platine ADK
- Robot
- capteur
- quincaillerie

2/ Analyse du problème

Comme indiqué précédemment, l’objectif du projet est de concevoir un robot permettant de servir de guide dans un magasin, ceci de manière autonome (c’est-à-dire sans communication avec un système à distance).

Les premiers problèmes qui se posent à nous sont les suivant :

- Doit-on adapter le magasin au robot ou l’inverse ?
- Quels moyens peut-on utiliser pour permettre au robot de se localiser?
- Quel type d’application Android doit-on créer pour ce type d’utilisation ?
- Comment gérer l’autonomie du robot ?

La première question est intéressante car elle constitue la base du projet. En effet le robot ne peut pas fonctionner de manière cohérente sous une organisation quelconque d'un magasin. Afin d'optimiser son utilisation et simplifier sa programmation deux possibilités s'offrent à nous :

- le mode de fonctionnement du robot devra définir le type répartition des produits dans le magasin
- L'organisation du magasin définira la programmation du robot. Dans ce cas le robot devra s'adapter à la complexité du magasin.

Dans notre cas, nous décidons de suivre la première idée. Ne disposant pas d'un modèle de magasin, nous avons établi un modèle simple nous permettant de programmer le robot.

Afin de se localiser et ainsi résoudre notre second problème, nous choisissons d'abord de fonctionner avec des lignes marquées au sol. Il est également possible de travailler avec un système RFID de marquage au sol dans notre cas.

L'application Android que nous proposons pour ce robot se fonde sur l'utilisation de 2 bases de données (une pour le magasin et une pour les choix du client). L'utilisateur doit être capable de faire une liste de choix de produit et suivre le robot.

Cette liste de choix implique que le robot devra fonctionner sur une longue distance et sur une longue durée, ce qui impose une autonomie suffisante et la multiplicité de points de rechargement. Il est par conséquent nécessaire d'en tenir compte dans le choix du robot.

3/ ETAPE1 : Le choix du robot

Comme analysé précédemment certains critères d'autonomie, de capacité de programmation, de localisation doivent être pris en compte dans le choix du robot. Mais une contrainte s'impose à ce projet. En effet l'utilisateur doit pourvoir utiliser un robot suffisamment grand et suffisamment stable. Alors pour répondre à se 2 exigences, nous pensons rajouter des barres fixables du le robot pour hausser l'emplacement de la tablette ainsi que des roulettes que nous placerons de chaque coté et à l'arrière du robot qui nous permettront de maintenir son équilibre dans ces déplacements.

Nous avons comparé différents modèles Robot et avons choisi un robot NEXUS mobile FB004 fabriqué par FiveBot évalué chez Robotshop à 751.83 euros.

Kit Robot Mobile à 2 Roues Motrices Compatible Arduino

Ce robot possède entre autre les caractéristiques indiquées ci dessous.

Longueur : 357mm Largeur : 313mm Hauteur : 267mm Vitesse max : 0.8m/s Propulsion : 2 roues motrices différentielles, 1 roue libre Pente franchissable : 20° Charge utile : 10kg Motorisation Type : Faulhaber 12V DC sans fer Puissance : 17W Courant en charge : 1400mA Rapport de boîte de vitesse : 64 :1 Carte de contrôle principale	Microcontrôleur ATMEGA328 Électronique de commande moteur 2A max Alimentation : USB ou alimentation externe 7-12V Ports d'alimentations disponibles : 5V/3.3V stabilisées et alimentation batterie Carte d'extension Contrôle de 2 moteurs supplémentaires Support RS-485 Support module Xbee Programmation Environnement de programmation Arduino AVR Studio avec programmation par USB

Ainsi afin d'atteindre une taille d'environ 1,5m pour un utilisateur humain, nous rajoutons à cela des barres en aluminium commandées chez radiospares (Code RS 493-8319 Fabricant Bosch Rexroth).

Le choix du robot sera par la suite modifié avec un robot à 3 motrices du même modèle et plus couteux.

4/ ETAPE2 : Le système de localisation et l'application

Pour concevoir notre application, nous sommes partis sur l'idée d'un magasin comportant 10 rayons et chaque rayon 2 produits.

A l'aide du logiciel Eclipse, nous avons pu coder sous Android 3.1 pour raison de compatibilité avec la tablette Motorola XOOM qui est utilisée pour ce projet.

Tablette Motorola XOOM

Nous avons démarré par un apprentissage de la programmation sous android avec des programmes simples du type « affichage d'un texte sur un écran ».

A partir de là et pour suivre notre raisonnement consistant à utiliser une base de données, nous avons pensé utiliser Sqlite qui est un moyen simple et efficace de créer et de travailler sur des bases de données avec Android.

Pour cela nous créons une classe MaBaseSQLite qui permet de créer une table de produits contenant 3 colonnes, l'identifiant (numérotation), le code (qui permet de localiser le produit) et le nom du produit.

Le code la classe est le suivant :

```
public class MaBaseSQLite extends SQLiteOpenHelper {
		private static final String TABLE_PRODUITS = "table_produits";
		private static final String COL_ID = "ID";
		private static final String COL_CODE = "CODE";
		private static final String COL_NAME = "name";
		private static final String CREATE_BDD = "CREATE TABLE " + TABLE_PRODUITS + "
("
+ COL_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " + COL_CODE + " TEXT NOT
NULL, "	+ COL_NAME + " TEXT NOT NULL);";
public MaBaseSQLite(Context context, String name, CursorFactory factory, int version) {
super(context, name, factory, version);
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
		//supprimer la table et la recréer
		//en changeant la version les id repartent de 0
		db.execSQL("DROP TABLE " + TABLE_PRODUITS + ";");
		onCreate(db);
	}
	@Override
	public void onCreate(SQLiteDatabase db) {
		//on crée la table
		db.execSQL(CREATE_BDD);
	}
}
```

Nous créons ensuite une autre classe qui produitBDD qui nous permet de travailler sur la base de donnée et d'effectuer certaines opérations telles que l'ajout, la suppression, la mise à jour où la recherche. Le code commenté est donné ci-dessous avec des explications pour chaque opération

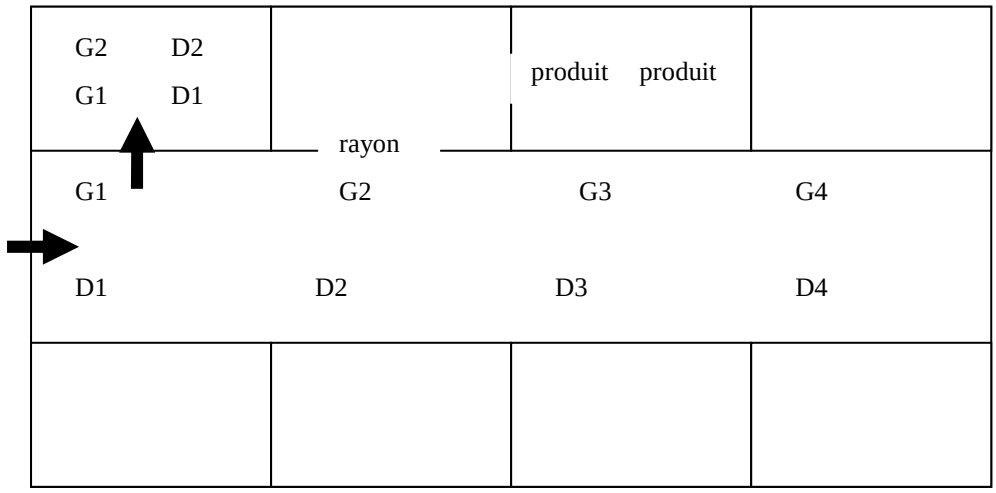
```
public class ProduitsBDD {
	private static final String TABLE_PRODUITS = "table_produits";
	private static final String COL_ID = "ID";
	private static final int NUM_COL_ID = 0;
	private static final String COL_CODE = "CODE";
	private static final int NUM_COL_CODE = 1;
	private static final String COL_NAME = "NAME";
	private static final int NUM_COL_NAME = 2;
	private SQLiteDatabase bdd;
	private MaBaseSQLite maBaseSQLite;

	public ProduitsBDD(Context context, int VERSION_BDD, String NOM_BDD  ){
		//On crée la BDD et sa table
		maBaseSQLite = new MaBaseSQLite(context, NOM_BDD, null, VERSION_BDD);
	}
	public void open(){
		//on ouvre la BDD en écriture
		bdd = maBaseSQLite.getWritableDatabase();
	}
	public void upgrade(){
		//on ouvre la BDD en écriture
		maBaseSQLite.onUpgrade(bdd, 1, 1);
	}
	public void close(){
		//on ferme l'accès à la BDD
		bdd.close();
	}
	public SQLiteDatabase getBDD(){
		return bdd;
	}
	public long insertProduit(Produit produit){
		//Création d'un ContentValues (fonctionne comme une HashMap)
		ContentValues values = new ContentValues();
		//on lui ajoute une valeur associée à une clé (qui est le nom de la colonne dans
laquelle on veut mettre la valeur)
		values.put(COL_CODE, produit.getCode());
		values.put(COL_NAME, produit.getname());
		//on insère l'objet dans la BDD via le ContentValues
		return bdd.insert(TABLE_PRODUITS, null, values);
	}

	public int updateProduit(int id, Produit produit){
		//La mise à jour d'un produit dans la BDD fonctionne plus ou moins comme une
insertion, il faut simplement préciser quel produit on doit mettre à jour grâce à l'ID
		ContentValues values = new ContentValues();
		values.put(COL_CODE, produit.getCode());
		values.put(COL_NAME, produit.getname());
		return bdd.update(TABLE_PRODUITS, values, COL_ID + " = " +id, null);
	}

	public void removeProduitWithID(int id){
		//Suppression d'un produit de la BDD grâce à l'ID
		int i=0;
		Produit fin = new Produit("fin","fin");
		for (i=id;i<getcountproduit(); i++)
		{ updateProduit(i, getProduitWithid(i+1));}
		updateProduit(getcountproduit(), fin);
```

```
    }
    public int getcountproduit()
    {
          Cursor c = bdd.query(TABLE_PRODUITS, new String[] {COL_ID, COL_CODE,
COL_NAME},null, null, null, null, null);
          return c.getCount();
     }
          //Recherche d'un produit grace à son id
    public Produit getProduitWithid(int id){
          //Récupère dans un Cursor les valeurs correspondant à un produit contenu dans la
BDD (ici on sélectionne le livre grâce à son nom)
          Cursor c = bdd.query(TABLE_PRODUITS, new String[] {COL_ID, COL_CODE, COL_NAME},
COL_ID +" = "+id  , null, null, null, null);
          return cursorToProduit(c);
    }
    //Cette méthode permet de convertir un cursor en un produit, ce cursor nous permet de
trouver les informations sur le produit correspondant
    private Produit cursorToProduit(Cursor c){
          //si aucun élément n'a été retourné dans la requête, on renvoie null
          if (c.getCount() == 0)
                return null;
          //Sinon on se place sur le premier élément
          c.moveToFirst();
          //On créé un produit
          Produit produit = new Produit();
          //on lui affecte toutes les infos grâce aux infos contenues dans le Cursor
          produit.setId(c.getInt(NUM_COL_ID));
          produit.setCode(c.getString(NUM_COL_CODE));
          produit.setname(c.getString(NUM_COL_NAME));
          //On ferme le cursor
          c.close();
          //On retourne le produit
          return produit;
    }
   }
```

Pour utiliser cette classe il est nécessaire de pouvoir définir des variables de type produit. On crée alors la classe Produit permettant cela

```
    public class Produit {

          private int id;
          private String code;
          private String name;
                //permet de déclarer un produit avec 2 paramètres
          public Produit(String code, String name){
                this.code = code;
                this.name = name;
          }
          public int getId() {return id;  }
          public void setId(int id) {this.id = id;}
          public String getCode() { return code;}
          public void setCode(String code) {this.code = code;}
          public String getname() { return name;}
          public void setname(String name) {this.name = name;}
                      //Convertit en string les données du produit
          public String toString(){ return "CODE : "+code+"\nRayon-produit : "+name;}
          public String toString2(){return "CODE : "+code+"\nRayon-produit : "+name;}
    }
```

Pour continuer la conception de notre application, nous avons défini un mode de localisation des produits. Le schéma suivant modélise ce mode de localisation :



Comme indiqué, l'application définit la position d'un produit à partir d'une position initiale en affectant une direction à chacun d'eux.

La position est alors définie par la position du rayon dans le magasin puis celle du produit dans le rayon. Dans notre modèle simplifié avec des rayons à droite et à gauche et des produits à droite et à gauche, On localise un produit par la direction du rayon (g ou d) puis le numéro de positionnement du rayon (1,2,3,4 …) , la direction d'un produit dans le rayon (g ou d) et de la même manière son numéro de positionnement.

Par exemple : g1d2 serait le code d'un produit dans la table des produits.

Cette modélisation nous permet de finaliser l'application avec les différents paramètres de l'interface utilisateur (voir main.xml en annexes)

On définit 2 tables de produits :

```
ProduitsBDD produitBdd = new ProduitsBDD(this, 1, "magasin");
ProduitsBDD choixBdd = new ProduitsBDD(this, 1, "client");
```

Et une liste de produit aléatoire afin de remplir la base de données :

```
Produit produit1 = new Produit("d1d1", "fruits-cerise");
Produit produit2 = new Produit("d1g1","fruits-orange");
Produit produit3 = new Produit("g1d1", "légumes-aubergine");
Produit produit4 = new Produit("g1g1", "légumes-pommedeterre");
Produit produit5 = new Produit("d2d1", "vetements-chemise");
Produit produit6 = new Produit("d2g1", "vetements-pantalon");
Produit produit7 = new Produit("g2d1","chaussures-botte");
Produit produit8 = new Produit("g2g1", "chaussures-basket");
Produit produit9 = new Produit("d3d1", "high tech-tablette");
Produit produit10 = new Produit("d3g1", "high tech-pc");
Produit produit11 = new Produit("g3d1", "cuisine-four");
Produit produit12 = new Produit("g3g1", "cuisine-marmite");
Produit produit13 = new Produit("d4d1", "petit déjeuner-pain");
Produit produit14 = new Produit("d4g1", "petit déjeuner-lait");
```

```
Produit produit15 = new Produit("g4d1", "hygiene-Mr propre");
Produit produit16 = new Produit("g4g1", "hygiene-Mir");
Produit produit17 = new Produit("d5d1", "beauté-maquillage");
Produit produit18 = new Produit("d5g1", "beauté-shampoing");
Produit produit19 = new Produit("g5d1", "boisson-soda");
Produit produit20 = new Produit("g5g1","boisson-vin");
```

Chaque produit est ensuite inséré dans la base de données

```
produitBdd.insertProduit(produit1);…
```

Pour choisir un produit à ajouter à la liste on récupère le numéro du produit sélectionné sur l'écran dans la galerie (voir main.xml, imageadaptergallery et code de l'application en annexe)qui correspond à l'ID du produit dans produitBDD que l'on passe en paramètre à la fonction chooser. Cette dernière lance une requête dans produitBDD, récupère le produit correspondant et remplit choixBDD.

```
protected void chooser(int num) {
 produitBdd.open();
 choixBdd.open();
 Produit produitchoisiFromBdd = produitBdd.getProduitWithid(num+1);
 GridView gridview1 = (GridView) findViewById(R.id.gridview1);
       if(produitchoisiFromBdd != null){
       choixBdd.insertProduit(produitchoisiFromBdd) ;
       //On affiche les infos du produit dans un Toast
 Toast.makeText(this,produitchoisiFromBdd.toString(),Toast.LENGTH_LONG).show();

        gridview1.setAdapter(new Image2Adapter(this, num));
       }
 remplirtable();
       //On affiche la liste des choix
 affichageliste();
  produitBdd.close();
  choixBdd.close();
}
```

Pour l'affichage et pour disposer d'une table de données accessible sans requete sur la base de données on déclare une table

```
ArrayList<HashMap<String, String>> listItem = new ArrayList<HashMap<String, String>>();
```

Cette liste nous permettra de disposer à tout moment des choix et transférer cette liste au robot.
La fonction remplirtable() permet alors de remplir la table listItem en se servant d'une boucle allant de 1 à au nombre de produit dans choixBDD.

```
protected void remplirtable()
{
 Produit produitchoisiFromBdd;
 choixBdd.open();
 listItem.clear();
 int i=1;
       //On affiche les infos du produit dans un Toast
       for(i=1;i<=choixBdd.getcountproduit();i++)
       {
             produitchoisiFromBdd = choixBdd.getProduitWithid(i);
             if (produitchoisiFromBdd != null){
```

```
            HashMap<String, String> map = new HashMap<String, String>();
            map.put("titre", produitchoisiFromBdd.toString2());
            map.put("description", "Produit");
            map.put("img", String.valueOf(R.drawable.x));
            listItem.add(map);
              }
        }
         choixBdd.close();
}
```

Le code complet (voir annexe) permet d'avoir un apercu comme suit

La première version de l'application (le bouton « valider » est remplacé par un bouton ON/OFF et le bouton visualiser a été supprimé.

Fonctionnement :

L'utilisateur dispose d'une galerie de produits. Un click sur un produit permet de l'ajouter à la liste et également de visualiser certaines informations sur le produit. La liste s'affiche au fur et à mesure des choix. Pour supprimer un produit de sa liste l'utilisateur doit cliquer sur le choix qu'il souhaite supprimer. La suppression est alors automatique.

L'utilisateur peut également supprimer entièrement la liste.

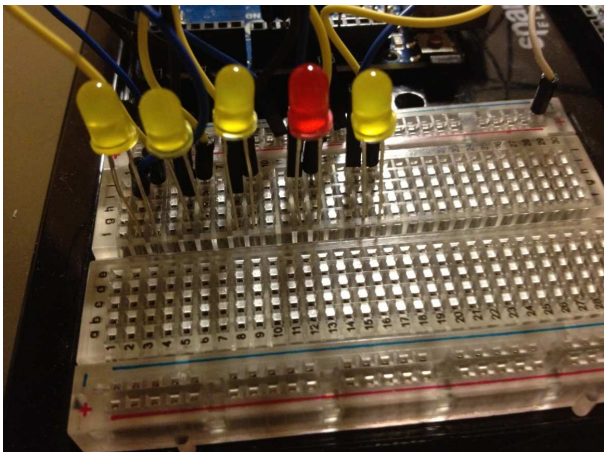
Dans la suite le bouton ON/OFF permet de valider la liste et de donner l'ordre au robot.

5/ ETAPE3 : La communication Arduino

La communication Arduino est faite par câble USB.

La tablette envoie un des octets correspondant au code ascii des codes de produits et le arduino récupère ces données pour les tester.

Ne disposant pas du robot, nous avons affiché sur des leds les résultats que nous pouvions espérer. Ainsi suivant le code entré le arduino exécute une commande correspondant à chaque fois à un octet (un caractère g , d ou un numéro).



Le code est le suivant :

```
#include <Max3421e.h>
#include <Usb.h>
#include <AndroidAccessory.h>
#define  LED_PIN  13
AndroidAccessory acc("Manufacturer",
            "Model",
            "Description",
            "1.0",
            "http://yoursite.com",
                "0000000012345678");
int led1=13;
int led2=12;
int led3=11;
int led4=10;
int led5=9;

void setup()
{

  Serial.begin(9600);           // start serial for output
  pinMode(led5,OUTPUT);
  pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(led3,OUTPUT);
  pinMode(led4,OUTPUT);
  // set communication speed
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT);
  acc.powerOn();
}


void loop()
{
  delay(100);
}

// dans la boucle on récupère le message et on test suivant sa valeur
```

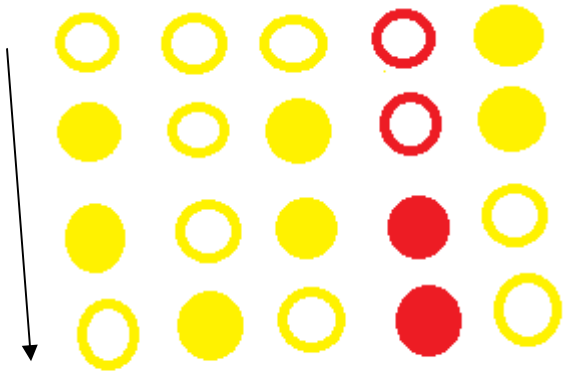
```
void loop()
{
  byte msg[0];
  if (acc.isConnected()) {
    int len = acc.read(msg, sizeof(msg), 1); // read data into msg variable
    if (len > 0) {
        if(msg==48)
   {
     digitalWrite(led1,LOW);
     digitalWrite(led2,LOW);
     digitalWrite(led3,LOW);
     digitalWrite(led4,LOW);
     digitalWrite(led5,LOW);

   }
   else if(msg==49)
   {
     digitalWrite(led3,LOW);
     digitalWrite(led4,LOW);
     digitalWrite(led5,HIGH);
   }
   else if(msg==50)
   {
     digitalWrite(led3,LOW);
     digitalWrite(led4,HIGH);
     digitalWrite(led5,LOW);
   }
   else if(msg==51)
   {

     digitalWrite(led3,LOW);
     digitalWrite(led4,HIGH);
     digitalWrite(led5,HIGH);
   }
   else if(msg==52)
   {

     digitalWrite(led3,HIGH);
     digitalWrite(led4,LOW);
     digitalWrite(led5,LOW);
   }
   else if(msg==53)
   {
     digitalWrite(led3,HIGH);
     digitalWrite(led4,LOW);
     digitalWrite(led5,HIGH);
   }
   else if(msg==103)
   {
     digitalWrite(led1,HIGH);
     digitalWrite(led2,LOW);

   }
   else if(msg==100)
   {
     digitalWrite(led1,LOW);
     digitalWrite(led2,HIGH);
   }

    }
  }
}
```

DROITE		GAUCHE
G1	C	D1
GAUCHE		DROITE
DROITE	O	GAUCHE
G2		D2
GAUCHE	U	DROITE
DROITE		GAUCHE
G3	L	D3
GAUCHE		DROITE
DROITE	O	GAUCHE
G4		D4
GAUCHE	I	DROITE
DROITE		GAUCHE
G5	R	D5
GAUCHE		DROITE

Exemple :

Allumé Eteind Eteind Allumé

Si le arduino reçoit g5d2, les LED doit change etape par etape simulant le comportement du robot.



Nous imaginons ensuite qu'un capteur envoie un signal lorsque le robot arrive en face d'un produit.

Dans ce cas la carte UNO(Slave) envoie un signal 'arrivé' vers la carte ADK (master), si le robot n'arrive pas encore, la carte UNO envoie un signal '0' vers la carte ADK. Un autre cas est celui où la carte ADK reçoit un signal 'arrivé' et va envoyer un signal 's'à UNO pour l'arrêter, a ce moment là toutes les LED sont désactivé.

Dans ce cas de figure les codes ne sont plus les mêmes et les tests sur les codes produits sont effectué à la fois sur les arduinos UNO et ADK

Partie du code ajouté à la carte Master

```
Wire.requestFrom(4,6);
 While(Wire.available())
{
        char c= Wire.read();
        Serial.print(c);
        If(c!=48)
        Wire.beginTransmission(4);
        Wire.write(“s”);
        Wire.endTransmission(4);
}
```

Partie du code ajouté à la carte slave

```
void requestEvent()

{

if(capteur==1)

Wire.write("arrive");

else

Wire.write("0");

}
```

6/ ETAPE4 : La commande par la tablette

La base de données choixBdd nous permet de transmettre après appui sur le bouton ON/OFF les différents codes de produit.

On converti grace à l’arborescence choixBdd.getProduitWithid(i).getCode().getBytes() le code du produit en tableau d’octet que l’on transmet un par un au arduino.

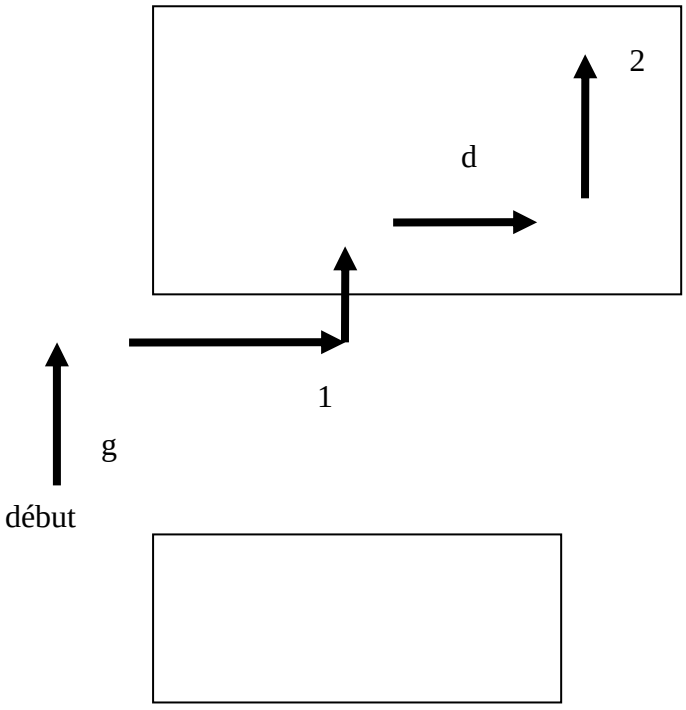
Le code global de la fonction est le suivant (voir code complet en annexe)

```
public void blinkLED(View v){

      int i = 0;
      int l = 0;

      byte buffer1[] = new byte[1];
            if(buttonLED.isChecked())
            {     buffer1[0]=(byte)0; // button says on, light is off
            try {
            mOutputStream.write(buffer1);} catch (IOException e) {
                Log.e(TAG, "write failed", e);
                } }
```

```
            else
            {
                  for (i=1;i<=choixBdd.getcountproduit();i++)
                        {

                        if(choixBdd.getProduitWithid(i)!=null){
                              byte buffer2[] =
choixBdd.getProduitWithid(i).getCode().getBytes();
                              if (mOutputStream != null) {
                                    try {
                                          for(l=0; l<buffer2.length; l++)

      mOutputStream.write(buffer2[l]);

                                          } catch (IOException e) {
                                                Log.e(TAG, "write failed", e);
                                                }
                              }
                                    }
                        }
            }
    }
```

Un exemple de commande est donné ci-dessous :

On choisit un produit dont le code est g1d2. Selon notre algorithme le robot suit alors la trajectoire suivante



7/ Conclusion

Ce projet, très intéressant, nous a permis d'observer une collaboration entre la programmation informatique, l'électronique et la robotique.

L'absence du robot durant la totalité du déroulement du projet a été préjudiciable mais ne nous a pas freinés dans la création de notre application et dans l'élaboration de solution globale aux problèmes.

Les livrables pourront être améliorés par la suite avec par exemple la création d'une galerie de rayon donnant accès à une galerie de produits, la création de plusieurs table de sélection de produits par rayon et les commandes associées au robot.

La communication entre le robot et la tablette devra être revue de manière plus complète cette fois avec des dispositifs de localisation du robot (Ligne blanche, tag RFID ou autres) adaptés

8/ Sources

http://www.robotshop.com/eu/kit-robot-mobile-2-roues-motrices-compatible-arduino-2.html

http://nexusrobot.com/

http://developer.android.com

http://allaboutee.com/2011/12/31/arduino-adk-board-how-send-data-from-the-board-to-the-android-device/

http://www.arduino.cc/

9/ Annexes

Image2adapter

```
package com.android.projet;
import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;

public class Image2Adapter extends BaseAdapter {
    private Context mContext;
    int k;
    public Image2Adapter(Context c, int m) {
        mContext = c;
        k=m;
    }
    public int getCount() {
        return 1;
    }
    public Object getItem(int position) {
        return null;
    }
    public long getItemId(int position) {
        return 0;
    }
    // create a new ImageView for each item referenced by the Adapter
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView;
        if (convertView == null) {  // if it's not recycled, initialize some attributes
            imageView = new ImageView(mContext);
            imageView.setLayoutParams(new GridView.LayoutParams(85, 85));
```

```
            imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
            imageView.setPadding(8, 8, 8, 8);
        } else {
            imageView = (ImageView) convertView;
        }
        imageView.setImageResource(mThumbIds[k]);
        return imageView;
    }
// references to our images
    private Integer[] mThumbIds = {
                    R.drawable.a, R.drawable.b,
                    R.drawable.c, R.drawable.d,
                    R.drawable.e, R.drawable.f,
                    R.drawable.g, R.drawable.h,
                    R.drawable.i, R.drawable.j,
                    R.drawable.k, R.drawable.l,
                    R.drawable.m, R.drawable.n,
                    R.drawable.o, R.drawable.p,
                    R.drawable.q, R.drawable.r,
                    R.drawable.s, R.drawable.t,
    }
}
```

Imageadaptergallery

```
package com.android.projet;
import android.content.Context;
import android.content.res.TypedArray;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;

public class ImageAdaptergallery extends BaseAdapter {
    int mGalleryItemBackground;
```

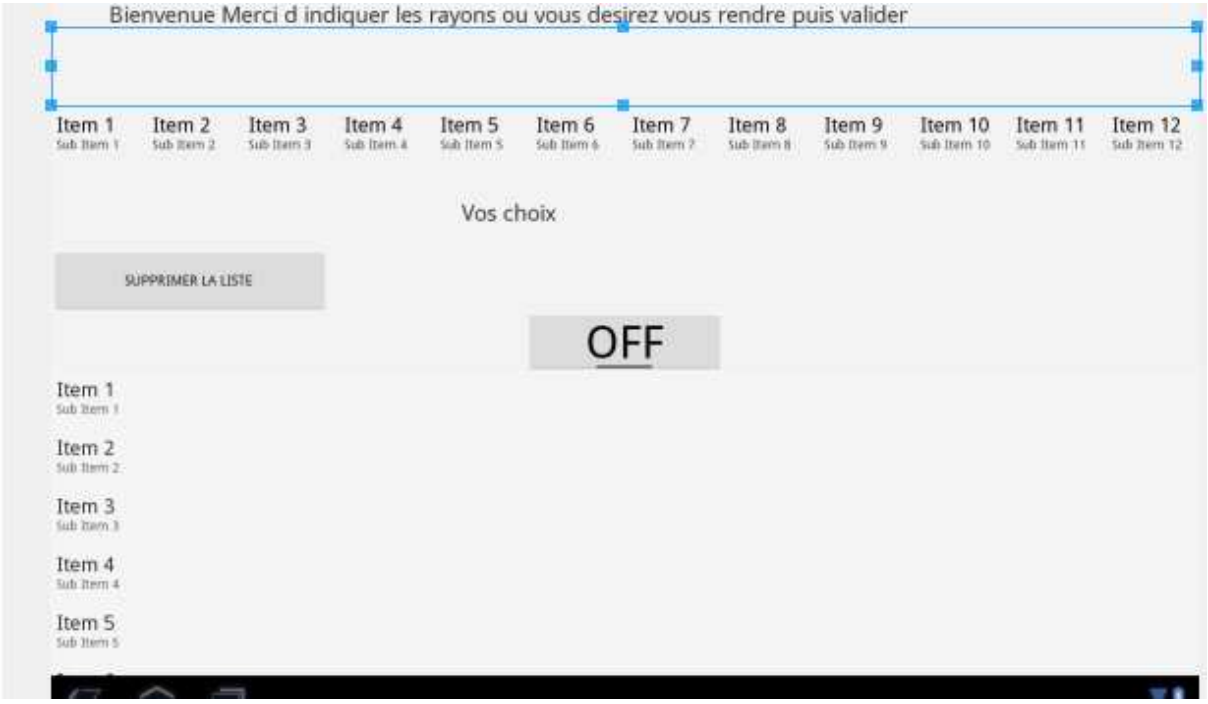
```
    private Context mContext;
    private Integer[] mImageIds = {
                    R.drawable.a, R.drawable.b,
                    R.drawable.c, R.drawable.d,
                    R.drawable.e, R.drawable.f,
                    R.drawable.g, R.drawable.h,
                    R.drawable.i, R.drawable.j,
                    R.drawable.k, R.drawable.l,
                    R.drawable.m, R.drawable.n,
                    R.drawable.o, R.drawable.p,
                    R.drawable.q, R.drawable.r,
                    R.drawable.s, R.drawable.t,
    };
    public ImageAdaptergallery(Context c) {
       mContext = c;
       TypedArray attr = mContext.obtainStyledAttributes(R.styleable.HelloGallery);
       mGalleryItemBackground = attr.getResourceId(
             R.styleable.HelloGallery_android_galleryItemBackground, 0);
       attr.recycle();
    }
    public int getCount() {
       return mImageIds.length;
    }
    public Object getItem(int position) {
       return position;
    }
    public long getItemId(int position) {
       return position;
    }
    public View getView(int position, View convertView, ViewGroup parent) {
       ImageView imageView = new ImageView(mContext);
       imageView.setImageResource(mImageIds[position]);
       imageView.setLayoutParams(new Gallery.LayoutParams(150, 100));
       imageView.setScaleType(ImageView.ScaleType.FIT_XY);
       imageView.setBackgroundResource(mGalleryItemBackground);
       return imageView;
```

```
    }
}
```

Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >


    <TextView
        android:layout_width="1020dp"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="@string/hello"
        android:textSize="25dp" />



    <Gallery
        android:id="@+id/gallery1"
        android:layout_width="match_parent"
        android:layout_height="89dp" />


    <GridView
        android:id="@+id/gridview1"
        android:layout_width="wrap_content"
        android:layout_height="77dp"
        android:columnWidth="90dp"
        android:gravity="center"
        android:horizontalSpacing="10dp"
        android:numColumns="auto_fit"
        android:stretchMode="columnWidth"
        android:verticalSpacing="10dp" >

    </GridView>

    <TextView
        android:layout_width="1020dp"
        android:layout_height="81dp"
        android:gravity="center"
        android:text="@string/choix"
        android:textSize="25dp" />

    <Button
        android:id="@+id/button2"
        android:layout_width="309dp"
        android:layout_height="70dp"
        android:gravity="center"
        android:text="SUPPRIMER LA LISTE"
        android:textSize="15dp" />




    <ToggleButton
        android:id="@+id/toggleButtonLED"
```

```
        android:layout_width="221dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:onClick="blinkLED"
        android:text="ToggleButton"
        android:textSize="50px" />

    <ListView
        android:id="@+id/listview"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </ListView>

</LinearLayout>
```



Code de l'application projet

```
package com.android.projet;

import java.io.FileDescriptor;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;

import android.util.Log;
import android.view.*;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
```

```
import android.content.Intent;
import android.content.IntentFilter;
import android.hardware.usb.UsbAccessory;
import android.hardware.usb.UsbManager;
import android.os.Bundle;
import android.os.ParcelFileDescriptor;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.*;

public class ProjetActivity extends Activity {


      //déclaration pour toggle button
       private static final String TAG = "ArduinoAccessory";

    private static final String ACTION_USB_PERMISSION = "com.android.projet";

    private UsbManager mUsbManager;
    private PendingIntent mPermissionIntent;
    private boolean mPermissionRequestPending;

    UsbAccessory mAccessory;
    ParcelFileDescriptor mFileDescriptor;
    FileInputStream mInputStream;
    FileOutputStream mOutputStream;
    int j=0;
    private ToggleButton buttonLED;

    //Création d'une instance de ma classe ProduitsBDD

      private ListView maListViewPerso;

      ProduitsBDD produitBdd = new ProduitsBDD(this, 1, "magasin");
      ProduitsBDD choixBdd = new ProduitsBDD(this, 1, "client");


    //Création de la ArrayList qui nous permettra de remplir la listView
    ArrayList<HashMap<String, String>> listItem = new ArrayList<HashMap<String,
String>>();



      /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //Création d'un produit
        Produit produit1 = new Produit("d1d1", "fruits-cerise");
        Produit produit2 = new Produit("d1g1","fruits-orange");
        Produit produit3 = new Produit("g1d1", "légumes-aubergine");
        Produit produit4 = new Produit("g1g1", "légumes-pommedeterre");
        Produit produit5 = new Produit("d2d1", "vetements-chemise");
        Produit produit6 = new Produit("d2g1", "vetements-pantalon");
        Produit produit7 = new Produit("g2d1","chaussures-botte");
        Produit produit8 = new Produit("g2g1", "chaussures-basket");
        Produit produit9 = new Produit("d3d1", "high tech-tablette");
        Produit produit10 = new Produit("d3g1", "high tech-pc");
        Produit produit11 = new Produit("g3d1", "cuisine-four");
        Produit produit12 = new Produit("g3g1", "cuisine-marmite");
        Produit produit13 = new Produit("d4d1", "petit déjeuner-pain");
        Produit produit14 = new Produit("d4g1", "petit déjeuner-lait");
```

```
        Produit produit15 = new Produit("g4d1", "hygiene-Mr propre");
        Produit produit16 = new Produit("g4g1", "hygiene-Mir");
        Produit produit17 = new Produit("d5d1", "beauté-maquillage");
        Produit produit18 = new Produit("d5g1", "beauté-shampoing");
        Produit produit19 = new Produit("g5d1", "boisson-soda");
        Produit produit20 = new Produit("g5g1","boisson-vin");
        //mise à jour de la base de données
        choixBdd.open();
        choixBdd.upgrade();
        choixBdd.close();

        //On ouvre la base de données pour écrire dedans
        produitBdd.open();

        produitBdd.upgrade();
        //On insère le produit que l'on vient de créer
        produitBdd.insertProduit(produit1);
        produitBdd.insertProduit(produit2);
        produitBdd.insertProduit(produit3);
        produitBdd.insertProduit(produit4);
        produitBdd.insertProduit(produit5);
        produitBdd.insertProduit(produit6);
        produitBdd.insertProduit(produit7);
        produitBdd.insertProduit(produit8);
        produitBdd.insertProduit(produit9);
        produitBdd.insertProduit(produit10);
        produitBdd.insertProduit(produit11);
        produitBdd.insertProduit(produit12);
        produitBdd.insertProduit(produit13);
        produitBdd.insertProduit(produit14);
        produitBdd.insertProduit(produit15);
        produitBdd.insertProduit(produit16);
        produitBdd.insertProduit(produit17);
        produitBdd.insertProduit(produit18);
        produitBdd.insertProduit(produit19);
        produitBdd.insertProduit(produit20);


        //galerie d'affichage des produits
        Gallery gallery = (Gallery) findViewById(R.id.gallery1);
        gallery.setAdapter(new ImageAdaptergallery(this));
        gallery.setOnItemClickListener(new OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent, View v, int position,
long id) {
             chooser(position);
            }
        });


        //bouton supprimer liste
        final Button button2 = (Button) findViewById(R.id.button2);
        button2.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
             choixBdd.open();
             choixBdd.upgrade();
              choixBdd.close();
              listItem.clear();
              affichageliste();
              j=0;


            }
        });
```

```
        produitBdd.close();


        // bouton toggle  permettant de valider l'envoi de données

        mUsbManager = (UsbManager) getSystemService(Context.USB_SERVICE);

        mPermissionIntent = PendingIntent.getBroadcast(this, 0, new
Intent(ACTION_USB_PERMISSION), 0);
        IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
        filter.addAction(UsbManager.ACTION_USB_ACCESSORY_DETACHED);
        registerReceiver(mUsbReceiver, filter);

        if (getLastNonConfigurationInstance() != null) {
        mAccessory = (UsbAccessory) getLastNonConfigurationInstance();
        openAccessory(mAccessory);
        }
        buttonLED = (ToggleButton) findViewById(R.id.toggleButtonLED);


    }



////////////////////////////////////////////////////////////////////////////////
//////////////////

////////////////////////////////////////////////////////////////////////////////
//////////////////

////////////////////////////////////////////////////////////////////////////////
//////////////////


    //bouton toggle valider liste

    public void blinkLED(View v){

       int i = 0;
       int l = 0;

       byte buffer1[] = new byte[1];
             if(buttonLED.isChecked())
             {      buffer1[0]=(byte)0; // button says on, light is off
             try {
             mOutputStream.write(buffer1);} catch (IOException e) {
                 Log.e(TAG, "write failed", e);
                 } }
             else
             {
                   for (i=1;i<=choixBdd.getcountproduit();i++)
                         {

                         if(choixBdd.getProduitWithid(i)!=null){
                               byte buffer2[] =
choixBdd.getProduitWithid(i).getCode().getBytes();
                               if (mOutputStream != null) {
                                     try {
                                           for(l=0; l<buffer2.length; l++)

       mOutputStream.write(buffer2[l]);

                                           } catch (IOException e) {
```

```
                                                Log.e(TAG, "write failed", e);
                                                }
                                    }
                                          }
                                }
                    }
          }



    private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {
      @Override
      public void onReceive(Context context, Intent intent) {
      String action = intent.getAction();
      if (ACTION_USB_PERMISSION.equals(action)) {
            synchronized (this) {
                  //UsbAccessory accessory = UsbManager.getAccessory(intent);
                  UsbAccessory accessory = (UsbAccessory)
intent.getParcelableExtra(UsbManager.EXTRA_ACCESSORY);
                  if (intent.getBooleanExtra(
                              UsbManager.EXTRA_PERMISSION_GRANTED, false)) {
                        openAccessory(accessory);
                  } else {
                        Log.d(TAG, "permission denied for accessory "
                                    + accessory);
                  }
                  mPermissionRequestPending = false;
            }
      } else if (UsbManager.ACTION_USB_ACCESSORY_DETACHED.equals(action)) {
      //UsbAccessory accessory = UsbManager.getAccessory(intent);
            UsbAccessory accessory = (UsbAccessory)
intent.getParcelableExtra(UsbManager.EXTRA_ACCESSORY);

      if (accessory != null && accessory.equals(mAccessory)) {
            closeAccessory();
      }
      }
      }
      };


      public Object onRetainNonConfigurationInstance() {
            if (mAccessory != null) {
                  return mAccessory;
            } else {
                  return super.onRetainNonConfigurationInstance();
            }
      }



      public void onResume() {
            super.onResume();

            if (mInputStream != null && mOutputStream != null) {
                  return;
            }

            UsbAccessory[] accessories = mUsbManager.getAccessoryList();
            UsbAccessory accessory = (accessories == null ? null :
accessories[0]);
            if (accessory != null) {
                  if (mUsbManager.hasPermission(accessory)) {
```

```
                        openAccessory(accessory);
                } else {
                        synchronized (mUsbReceiver) {
                                if (!mPermissionRequestPending) {

mUsbManager.requestPermission(accessory,mPermissionIntent);
                                        mPermissionRequestPending = true;
                                }
                        }
                }
        } else {
                Log.d(TAG, "mAccessory is null");
        }
}


public void onPause() {
        super.onPause();
        closeAccessory();
}

public void onDestroy() {
        unregisterReceiver(mUsbReceiver);
        super.onDestroy();
}

private void openAccessory(UsbAccessory accessory) {
        mFileDescriptor = mUsbManager.openAccessory(accessory);
        if (mFileDescriptor != null) {
                mAccessory = accessory;
                FileDescriptor fd = mFileDescriptor.getFileDescriptor();
                mInputStream = new FileInputStream(fd);
                mOutputStream = new FileOutputStream(fd);
                Log.d(TAG, "accessory opened");
        } else {
                Log.d(TAG, "accessory open fail");
        }
}


private void closeAccessory() {
        try {
                if (mFileDescriptor != null) {
                        mFileDescriptor.close();
                }
        } catch (IOException e) {
} finally {
        mFileDescriptor = null;
        mAccessory = null;
        }
}
```

```
////////////////////////////////////////////////////////////////////////////////
//////////////////

////////////////////////////////////////////////////////////////////////////////
//////////////////

////////////////////////////////////////////////////////////////////////////////
//////////////////
```

```
    //permet de sélectionner les produits dans la base de donner et de remplir
choixBdd
    protected void chooser(int num) {
       produitBdd.open();
       choixBdd.open();
       Produit produitchoisiFromBdd = produitBdd.getProduitWithid(num+1);
       GridView gridview1 = (GridView) findViewById(R.id.gridview1);
       //nombre maximal de produits à selectionner
       //if (choixBdd.getcountproduit() < 10)
       {
              if(produitchoisiFromBdd != null){
                     choixBdd.insertProduit(produitchoisiFromBdd) ;
                     //On affiche les infos du produit dans un Toast
                     Toast.makeText(this, produitchoisiFromBdd.toString(),
Toast.LENGTH_SHORT).show();
                    gridview1.setAdapter(new Image2Adapter(this, num));
                    remplirtable();
                     affichageliste();
                     }
       }
              //else {
              //     Toast.makeText(this,"LISTE PLEINE !!!!\n Veuillez valider
votre liste ou la supprimer", Toast.LENGTH_LONG).show();
              //     }

        produitBdd.close();
        choixBdd.close();
    }


//permet de remplir listItem

    protected void remplirtable()
    {
       Produit produitchoisiFromBdd;
       choixBdd.open();
       listItem.clear();
       int i=1;
              //On affiche les infos du produit dans un Toast
              for(i=1;i<=choixBdd.getcountproduit()+j;i++)
              {
                     produitchoisiFromBdd = choixBdd.getProduitWithid(i);
                     if (produitchoisiFromBdd != null){

                  HashMap<String, String> map = new HashMap<String, String>();
                  map.put("titre", produitchoisiFromBdd.toString2());
                  map.put("description", produitchoisiFromBdd.toString());
                  map.put("img", String.valueOf(R.drawable.x));
                  listItem.add(map);
                     }
              }
               choixBdd.close();
    }



    protected void affichageliste (){
       maListViewPerso = (ListView) findViewById(R.id.listview);
```

```
        //Création d'un SimpleAdapter qui se chargera de mettre les items présents
dans notre list (listItem) dans la vue affichageitem
        SimpleAdapter mSchedule = new SimpleAdapter
(ProjetActivity.this.getBaseContext(), listItem, R.layout.affichageitem,
                new String[] {"img", "titre", "description"}, new int[] {R.id.img,
R.id.titre, R.id.description});
        //On attribut à notre listView l'adapter que l'on vient de créer
        maListViewPerso.setAdapter(mSchedule);

                maListViewPerso.setOnItemClickListener(new OnItemClickListener() {
                    @SuppressWarnings("unchecked")
            public void onItemClick(AdapterView<?> a, View v, int position, long
id) {
                            //on récupère la HashMap contenant les infos de notre
item (titre, description, img)
                    HashMap<String, String> map = (HashMap<String, String>)
maListViewPerso.getItemAtPosition(position);
                    //on créer une boite de dialogue
                    AlertDialog.Builder adb = new
AlertDialog.Builder(ProjetActivity.this);
                    //on attribut un titre à notre boite de dialogue
                    adb.setTitle("suppression produit");
                    //on insère un message à notre boite de dialogue, et ici on
affiche le titre de l'item cliqué
                    adb.setMessage("selection : "+map.get("titre"));
                    //on indique que l'on veut le bouton ok à notre boite de
dialogue
                    adb.setPositiveButton("Ok", null);
                    //adb.setNegativeButton("Annuler", null);
                    //on affiche la boite de dialogue
                    adb.show();

                    choixBdd.open();

      choixBdd.removeProduitWithname(listItem.get(position).get("titre"));
                choixBdd.close();


                    listItem.remove(position);
                    j=j+1;
                remplirtable();
                    affichageliste();

            }
         });

    }

}
```