Phidget Mail Displayer v.1a

Contenu du rapport

Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg

1. Introduction

L'informatique "domestique" ne se limite plus, aujourd'hui, à un traitement de texte et le fameux démineur. L'ordinateur est devenu un acteur à part entière de la vie quotidienne, il sert autant à rechercher une recette de cuisine sur Internet qu'à diffuser sa musique préférée.

Depuis les débuts, on connaissait le clavier. Puis la souris est arrivée, d'abord un outil rudimentaire inventé par Douglas Engelbart en 1963 (voir fig. 1.1), qui est devenu un concentré de technologie doté de laser.

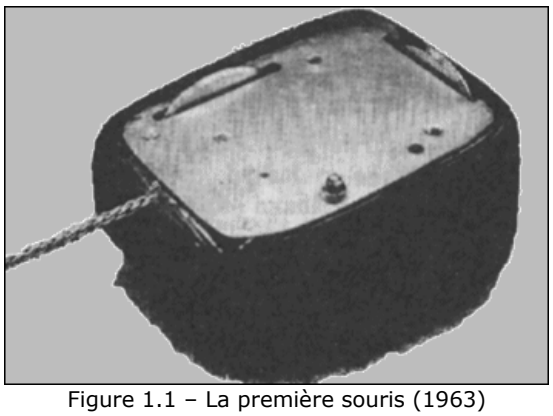


Figure 1.1 – La première souris (1963)

L'évolution veut que l'ordinateur acquière des compétences et des interactions avec le monde extérieur se rapprochant au plus de celles des humains. Le clavier et la souris ne suffisant plus à cela, on y a ajouté la caméra, le micro, et divers périphériques aujourd'hui tout à fait ordinaires.

Les phidgets étendent encore un peu plus ces "sens" qu'on a donnés à l'ordinateur. Grâce à eux, il est capable de mesurer une température ou une accélération, de détecter des objets, etc.

Des applications de toutes sortes peuvent être envisagées, l'imagination de chacun n'a pas de limites ! On pourrait imaginer changer le thème d'affichage d'un site web en fonction de la température du lieu où se trouve le serveur qui l'héberge, une application devant lire un CD-rom dans un ordinateur portable qui stoppe la lecture si l'utilisateur se déplace et provoque trop de vibrations dans le lecteur…

2. Presentation de l'application

Nous avons choisi de créer une application qui libère un petit objet lorsqu'un nouveau message arrive dans la boîte e-mail. Plus tard, lorsque l'utilisateur place l'objet devant un lecteur, le message est à nouveau automatiquement affiché à l'écran.

Trois Phidgets sont nécessaires pour la réalisation de cette application : un servomoteur et deux lecteurs de tags RFID (voir chapitre 3).

Figure 2.1 – Servomoteur

Figure 2.2 - Lecteur de tags RFID

Cette application peut servir pour une personne qui n'a pas la possibilité de relever ses e-mails régulièrement, mais qui reste la plupart du temps près de son ordinateur. Lorsqu'un message arrive dans la boîte e-mail, une action physique concrète est déclanchée, ce qui attire l'attention même dans un environnement bruyant ou agité.

Montage

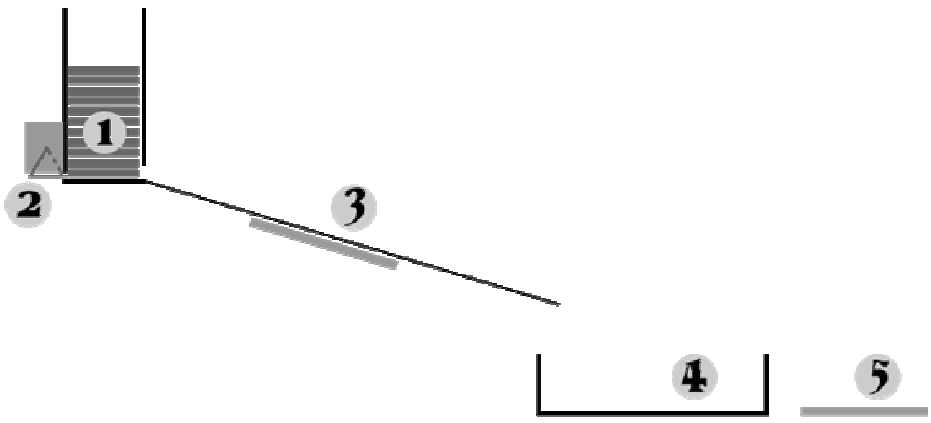


Figure 2.3 - Montage des Phidgets

Fonctionnement

Le scénario de l'application est présenté ci-dessous, en rapport avec le schéma du montage (voir fig. 2.3) et les photos ci-dessous..

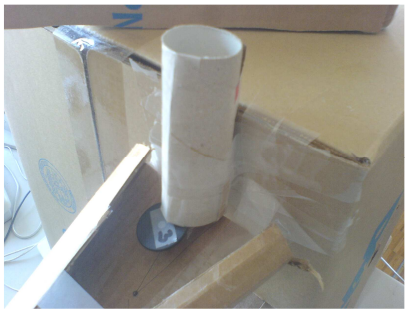
Une pile de tags RFID se trouve dans un tube placé verticalement **(1)**, avec une ouverture au fond sur le devant et une autre sur l'arrière.

Lorsque un nouvel e-mail arrive, le tag du fond est poussé hors de la pile par le phidget servomoteur **(2)**.

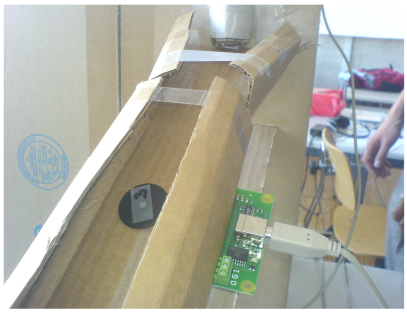
Le tag glisse sur le "toboggan" et est détecté par le phidget lecteur RFID **(3)**. Le numéro du tag est attribué à l'e-mail. Le tag tombe ensuite dans la boîte au fond du toboggan **(4)**.

L'entête de l'e-mail est ajouté à la liste dans la fenêtre principale.

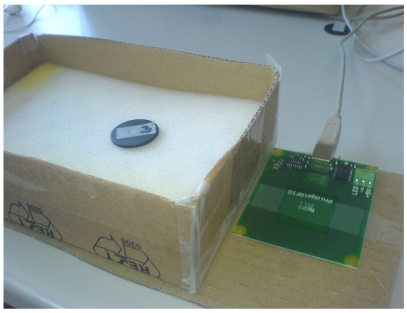
Pour lire l'e-mail, l'utilisateur peut soit cliquer sur l'entête dans la fenêtre principale, ou s'il a les mains sales par exemple, il a la possibilité de saisir le tag et de le passer devant le second lecteur RFID **(5)**, ce qui provoquera l'ouverture automatique de l'e-mail.



(1) et (2)



(3)



(4)

(5)

Ouverture de l'e-mail

3. Presentation des Phidgets utilises

Servomoteur

Le servomoteur se pilote à l'aide de la classe PhidgetServo. Son initialisation demande uniquement de lancer la méthode `boolean open(boolean)` de cette classe. Ensuite il suffit de d'utiliser la méthode `setMotorPosition(int, double)` où l'entier est l'id du moteur (on peut gérer plusieurs moteurs avec une seul classe) et le double la position que le moteur doit prendre. Le moteur utilisé permet un déplacement de la position 0 à la position 200, ce qui correspond à une rotation de 180°.

Lecteur de tag RFID

Ces lecteurs permettre de détecter la présence d'un tag RFID dans leur environ. On les utilise à l'aide de la classe PhidgetRFID. Nous avons cependant remarqué un petit problème : il est impossible de faire fonctionner plusieurs lecteurs à la fois à la manière des PhidgetServo. La solution est l'implémentation d'un thread permettant d'allumer et d'éteindre alternativement les différents lecteurs. Il faut implémenter un listener d'événement afin de réagir dès qu'un tag est détecté. La méthode se nomme `OnTag()`.

Figure 3.1 - Kit Interface 8/8/8 et alimentation pour servomoteur

Figure 3.2 - Lecteur de tags RFID

4. Code source

Classe MDrun

```
package phidgetMailDisplayer;

import javax.swing.SwingUtilities;

/**
 * @author Philippe Galley
 * @author Blaise Grand
 * @author Samuel Rossier
 *
 * Phidgets application : Mail displayer
 */

public class MDrun {

    public static void main(String[] args) {

            new MDsplashScreen().show();
            SwingUtilities.invokeLater(new Runnable() {
                    public void run() {
                            MDview myView = new MDview();
                            myView.setVisible(true);
                    }
            });

    }

}
```

Classe MDsplashScreen

```
package phidgetMailDisplayer;

import java.awt.Dimension;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.MediaTracker;
import java.awt.Rectangle;
import java.awt.Toolkit;
import java.awt.Window;
import java.net.URL;


public final class MDsplashScreen extends Frame {

    private static final String IMAGE NAME = "MDsplashScreen.png";
    private MediaTracker fMediaTracker;
    private Image fImage;

    // A SUIVRE (page suivante)
```

```
    public MDsplashScreen() {
    }

    public void show() {
            splash();
            try {
            Thread.sleep(3000);
            } catch (InterruptedException ie) {
            }
            dispose();
    }

    private void splash() {
            initImageAndTracker();
            setSize(fImage.getWidth(null), fImage.getHeight(null));
            center();
            fMediaTracker.addImage(fImage, 0);
            try {
                    fMediaTracker.waitForID(0);
            } catch (InterruptedException ie) {
                    System.out.println("Cannot track image load.");
            }
            SplashWindow splashWindow = new SplashWindow(this, fImage);
    }


    // Cette méthode permet d'initialiser l'image et le tracker
    private void initImageAndTracker() {
            fMediaTracker = new MediaTracker(this);
            URL imageURL = MDsplashScreen.class.getResource(IMAGE_NAME);
            fImage = Toolkit.getDefaultToolkit().getImage(imageURL);
    }
    // Centre la frame à l'écran
    private void center() {
            Dimension screen = Toolkit.getDefaultToolkit().getScreenSize();
            Rectangle frame = getBounds();
            setLocation((screen.width - frame.width) / 2,
                            (screen.height - frame.height) / 2);
    }

    private class SplashWindow extends Window {
            SplashWindow(Frame aParent, Image aImage) {
                    super(aParent);
                    fImage = aImage;
                    setSize(fImage.getWidth(null), fImage.getHeight(null));
                    Dimension screen = Toolkit.getDefaultToolkit().getScreenSize();
                    Rectangle window = getBounds();
                    setLocation((screen.width - window.width) / 2,
                                    (screen.height - window.height) / 2);
                    setVisible(true);
            }
            public void paint(Graphics graphics) {
                    if (fImage != null) {
                            graphics.drawImage(fImage, 0, 0, this);
                    }
            }
            private Image fImage;
    }
}
```

Classe MDview

```
package phidgetMailDisplayer;

import java.awt.*;
import javax.swing.*;

/**
 * @author Philippe Galley
 * @author Blaise Grand
 * @author Samuel Rossier
 *
 * Phidgets application : Mail displayer
 */

public class MDview extends JFrame {

    private MDcontroller myController;

    private static final int WINDOW_WIDTH = 600, WINDOW_HEIGHT = 400;

    public static final String SW NAME = "PhidgetMailDisplayer v.1a",
                jBgetMailsText = "GET E-MAILS NOW";

    // Composants graphiques

    private JTextField jTFpop3host = new JTextField("mail.siviriez06.ch"),
                jTFpop3port   =   new   JTextField("110"),   jTFpop3user   =   new
JTextField(
                                  "eif-mdtest@siviriez06.ch");

    private JPasswordField jPFpop3passwd = new JPasswordField("eiafr", 20);

    private JButton jBgetMails = new JButton(jBgetMailsText);

    private JLabel jLmailsInfo = new JLabel("Please get e-mails...");

    private JPanel mailListPanel = new JPanel();

    protected int crtDisplayedMail = -1;

    public MDview() {
            mailListPanel.setLayout(new                    BoxLayout(mailListPanel,
BoxLayout.Y AXIS));
            myController = new MDcontroller(this);
            buildIHM();
    }

    private void buildIHM() {

            setTitle(SW NAME);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setDefaultLookAndFeelDecorated(true);
            setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
            Dimension screenDim = Toolkit.getDefaultToolkit().getScreenSize();
            setLocation((screenDim.width - WINDOW WIDTH) / 2,
                        (screenDim.height - WINDOW HEIGHT) / 2);
            setBackground(Color.WHITE);

            Container cont = getContentPane();
            cont.setLayout(new BorderLayout());
            cont.add(new JLabel(new ImageIcon(getClass().getResource(
                        "MDappTitle.png"))), BorderLayout.NORTH);

            JPanel mailPanel = new JPanel(new GridBagLayout());
            GridBagConstraints gbc = new GridBagConstraints(0, 0, 1, 1, 0, 0,
                        GridBagConstraints.NORTH, GridBagConstraints.HORIZONTAL,
                        new Insets(0, 0, 0, 0), 2, 2);
            mailPanel.add(new JLabel("POP3 host : "), gbc);
```

```
            gbc.gridx = 1;
            mailPanel.add(jTFpop3host, gbc);
            gbc.gridx = 2;
            mailPanel.add(new JLabel("    Port : "), gbc);
            gbc.gridx = 3;
            mailPanel.add(jTFpop3port, gbc);
            gbc.gridy = 1;
            gbc.gridx = 0;
            mailPanel.add(new JLabel("Username : "), gbc);
            gbc.gridx = 1;
            mailPanel.add(jTFpop3user, gbc);
            gbc.gridx = 2;
            mailPanel.add(new JLabel("    Password : "), gbc);
            gbc.gridx = 3;
            mailPanel.add(jPFpop3passwd, gbc);
            gbc.gridy = 2;
            gbc.gridx = 0;
            gbc.gridwidth = 4;
            jBgetMails.setActionCommand("getMails");
            jBgetMails.addActionListener(myController);
            mailPanel.add(jBgetMails, gbc);
            gbc.gridy = 3;
            mailPanel.add(jLmailsInfo, gbc);
            gbc.gridy = 4;
            mailPanel.add(mailListPanel, gbc);
            cont.add(mailPanel, BorderLayout.CENTER);
    }

    public void displayMails(String[] mailsList) {
            for (int i = 0; i < mailsList.length; i++) {
                    JButton mailButton = new JButton(mailsList[i]);
                    mailButton.setActionCommand("mail" + i);
                    mailButton.addActionListener(myController);
                    mailListPanel.add(mailButton);
            }
    }

    public String getPasswd() {
            return jPFpop3passwd.getText();
    }

    public String getHost() {
            return jTFpop3host.getText();
    }

    public String getPort() {
            return jTFpop3port.getText();
    }

    public String getUsername() {
            return jTFpop3user.getText();
    }

    public void setGetMailsButtonText(String text) {
            jBgetMails.setText(text);
    }

    public void setMailInfoLabelText(String text) {
            jLmailsInfo.setText(text);
    }

    public void afficheMail(int i) {
            if (crtDisplayedMail != i) {
                    MDmailView mailView = new MDmailView(myController.myModel
                                    .getMailsList()[i], i, this);
                    mailView.setVisible(true);
            }
    }
}
```

Classe MDcontroller

```
package phidgetMailDisplayer;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;

/**
 * @author Philippe Galley
 * @author Blaise Grand
 * @author Samuel Rossier
 *
 * Phidgets application : Mail displayer
 */

public class MDcontroller implements ActionListener {

    private MDview myView;

    protected MDmodel myModel;

    public MDcontroller(MDview view) {
           myView = view;
           myModel = new MDmodel(this, myView);
    }

    public void actionPerformed(ActionEvent ae) {

           String ac = ((JButton) ae.getSource()).getActionCommand();

           if (ac.equals("getMails")) {

                  myView.setGetMailsButtonText("Downloading e-mails...");

                  String host = myView.getHost();
                  String port = myView.getPort();
                  String user = myView.getUsername();
                  String passwd = myView.getPasswd();

                  myView.setGetMailsButtonText(MDview.jBgetMailsText);
                  String emptyString = "";
                  if (!(host.equals(emptyString) || port.equals(emptyString)
                                ||          user.equals(emptyString)          ||
passwd.equals(emptyString))) {
                         try {

    myView.displayMails(myModel.getMailsFromServer(host, port,
                                                  user, passwd));
                         } catch (Exception e) {
                         }
                  } else {
                         myView.setGetMailsButtonText(myView.jBgetMailsText
                                       + " (please fill fields)");
                  }
           } else if (ac.startsWith("mail")) {
                  int crtMail = Integer.parseInt(ac.substring(4));
                  myView.afficheMail(crtMail);
           }
    }
}
```

Classe MDmodel

```
package phidgetMailDisplayer;

import java.io.BufferedReader;
import java.io.Reader;
import java.io.IOException;
import java.net.InetAddress;
import org.apache.commons.net.pop3.*;

import Phidgets.PhidgetServo;

/**
 * @author Philippe Galley
 * @author Blaise Grand
 * @author Samuel Rossier
 *
 * Phidgets application : Mail displayer
 */

public class MDmodel {

    private MDview myView;

    private MDcontroller myController;

    public String[] mailText;

    private MDreaderManager myRM;

    private POP3Client client = new POP3Client();

    private String myHost, myPort, myUsername, myPasswd;

    private boolean isConnected = false, isLogged = false,
                    serverContentOk = false, isReadyToRead = false;

    private String[] mailsList;

    private static final int DEFAULT TIMOUT = 60000, DEFAULT PORT = 110;

    public static final String[] TAG_ID = new String[10];

    public int currentMail = 0;

    public PhidgetServo myMotor;

    public MDmodel(MDcontroller controller, MDview view) {
            for (int i = 0; i < TAG_ID.length; i++)
                    TAG ID[i] = "";
            myView = view;
            myController = controller;
            myRM = new MDreaderManager(this);
            myRM.start();
            myMotor = new PhidgetServo();
            if (myMotor.Open(false) == false) {
                    System.out.println("Could not find a PhidgetServo");
            }
    }

    public String[] getMailsFromServer(String host, String port,
                    String username, String passwd) throws Exception {
            myHost = host;
            myPort = port;
            myUsername = username;
            myPasswd = passwd;

            client.setDefaultTimeout(DEFAULT TIMOUT);
            client.setDefaultPort(DEFAULT PORT);
            isConnected = connect();
            isLogged = login();
```

```
            if (isConnected && isLogged) {
                    POP3MessageInfo msgInfo = client.status();
                    if (msgInfo.number > 0 && msgInfo.size > 0) {
                            myView.setGetMailsButtonText(msgInfo.number
                                            + " e-mails in inbox");
                            POP3MessageInfo[] msg = client.listMessages();
                            if (msg == null)
            myView.setGetMailsButtonText("Could not retrieve messages list !");
                            else {
            myView.setGetMailsButtonText("Please get e-mails...");
                                    Reader reader;
                                    mailsList = new String[msg.length];
                                    mailText = new String[msg.length];

                                    for (int i = 0; i < msg.length; i++) {

                                    if (msg[i] != null && msg[i].number == i + 1
                                                    && msg[i].size > 0) {
                            reader = client.retrieveMessageTop(msg[i].number, 0);
                                                    if (reader != null) {
                                                    mailsList[i] = printMessageInfo(
                                                    new BufferedReader(reader),
                                                                    msg[i].number);
                                                            recieveAMail(i);
                                                    }
                                            }
                                    }
                            }
                    }
            }

            isLogged = !client.logout();
            isReadyToRead = true;
            reset();
            return mailsList;
    }

    private void recieveAMail(int i) {
            try {
                    Thread.sleep(i * 2000);
            } catch (InterruptedException e1) {
            }
            myMotor.SetMotorPosition(0, 200);
            try {
                    Thread.sleep(1000);
            } catch (InterruptedException e) {
            }
            myMotor.SetMotorPosition(0, 0);
    }

    public String[] getMailsList() {
            return mailsList;
    }

    public String printMessageInfo(BufferedReader reader, int id)
                    throws IOException {
            String line, lower, from, subject;

            from = "";
            subject = "";

            while ((line = reader.readLine()) != null) {
                    lower = line.toLowerCase();
                    if (lower.startsWith("from: "))
                            from = line.substring(6).trim();
                    else if (lower.startsWith("subject: "))
                            subject = line.substring(9).trim();
            }

            return Integer.toString(id) + " - From : " + from + " - Subject : "
                            + subject;
    }
```

```
    private boolean connect() throws Exception {
            client.connect(InetAddress.getByName(myHost),
                                Integer.parseInt(myPort));
            if (client.isConnected()
                            && POP3.AUTHORIZATION_STATE == client.getState())
                    return true;
            return false;
    }

    private boolean login() throws Exception {
            if (client.login(myUsername, myPasswd)
                            && POP3.TRANSACTION_STATE == client.getState())
                    return true;
            return false;
    }

    private void reset() throws Exception {
            if (isConnected) {
                    client.disconnect();
                    isConnected = false;
                    isLogged = false;
            }
            client = new POP3Client();
    }

    public boolean isReadyToRead() {
            return isReadyToRead;
    }

    public void afficheMail(int i) {
            myView.afficheMail(i);
    }

}
```

Classe MDmailView

```
package phidgetMailDisplayer;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;

/**
 * @author Philippe Galley
 * @author Blaise Grand
 * @author Samuel Rossier
 *
 * Phidgets application : Mail displayer
 */

public class MDmailView extends JFrame {

    String currentMail;

    private static final int WINDOW WIDTH = 400, WINDOW HEIGHT = 250;

    JButton closeButton = new JButton("Fermer");

    JFrame thisFrame = this;

    MDview myView;
```

```
    public MDmailView(String crtMail, int crtMailNb, MDview view) {
            myView = view;
            myView.crtDisplayedMail = crtMailNb;
            currentMail = crtMail;
            buildIHM();
    }

    private void buildIHM() {
            setTitle(MDview.SW NAME);
            setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
            setDefaultLookAndFeelDecorated(true);
            setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
            Dimension screenDim = Toolkit.getDefaultToolkit().getScreenSize();
            setLocation((screenDim.width - WINDOW WIDTH) / 2,
                            (screenDim.height - WINDOW HEIGHT) / 2);
            setBackground(Color.WHITE);
            Container cont = getContentPane();
            cont.setLayout(new BorderLayout());
            cont.add(new JLabel("Affichage d'un e-mail"), BorderLayout.NORTH);
            cont.add(new JLabel(currentMail), BorderLayout.CENTER);
            closeButton.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                            myView.crtDisplayedMail = -1;
                            thisFrame.hide();
                    }
            });
            cont.add(closeButton, BorderLayout.SOUTH);

    }

}
```

Classe MDreaderManager

```
package phidgetMailDisplayer;

import Phidgets.*;

public class MDreaderManager extends Thread {

    private MDmodel myModel;

    public static final int SERIALS[] = { 16459, 16054 };

    public static final int READERS CONFIGURATION[] = { 2, 2 };

    public static final int READ DELAY = 150;

    public PhidgetRFID readers[] = null; // readers threads list

    public boolean toBeRead[] = null; // used to activate/deactivate a reader

    public int tagCounts[] = null; // number of tags read in a shoot

    public int maxTagCount[] = null; // max number read in one shoot

    private boolean isRunning;

    public boolean initiated = false;

    long bench = 0;

    public MDreaderManager(MDmodel model) {
            myModel = model;
            isRunning = true;
    }
```

```
    public void run() {

            readers = new PhidgetRFID[SERIALS.length];
            toBeRead = new boolean[SERIALS.length];
            tagCounts = new int[SERIALS.length];
            maxTagCount = new int[SERIALS.length];
            for (int i = 0; i < readers.length; i++) {
                    maxTagCount[i] = 1;
            }

            for (int i = 0; i < readers.length; i++) {
                    // ONLY 2 TAG READERS !!!
                    readers[i] = new PhidgetRFID();
                    if (i == 0)
    readers[i].add IPhidgetRFIDEventsListener(new RFIDEventsAdapterNew(myModel));
                    else
    readers[i].add_IPhidgetRFIDEventsListener(new RFIDEventsAdapterOpen(myModel));
                    if (readers[i].Open(false, SERIALS[i]) == false) {
                    System.out.println("Could not find PhidgetRFID " + SERIALS[i]);
                    } else {
                            toBeRead[i] = true;
                            readers[i].SetOutputState(3, true);
                            readers[i].SetOutputState(2, false);
                            readers[i].start();
                    }
            }
            bench = System.currentTimeMillis();
            initiated = true;
            try {
                    while (isRunning) {
                            for (int i = 0; i < readers.length; i++) {
                                    readers[i].SetOutputState(1, false);
                                    readers[i].SetOutputState(3, false);
            readers[(i + 1) % readers.length].SetOutputState(1,toBeRead[i]);
            readers[(i + 1) % readers.length].SetOutputState(3,
                                                    toBeRead[i]);
                                    Thread.sleep(READ DELAY);
                            }

                            Thread.sleep(READ_DELAY);
                            for (int i = 0; i < tagCounts.length; i++) {
                                    tagCounts[i] = 0;
                            }
                    }
            } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            }

            // arrêt propre du Thread
            for (int i = 0; i < readers.length; i++) {
            readers[i].SetOutputState(1, false); // on eteint la LED externe
            readers[i].SetOutputState(2, false); // on eteint la LED onboard
                    readers[i].SetOutputState(3, false); // on eteint l'antenne
                    readers[i].Close();
            }
    }

    public void accountTag(int serial, String tagId) {
            for (int i = 0; i < SERIALS.length; i++) {
                    if (serial == SERIALS[i]) {
                            tagCounts[i]++;
                            if (maxTagCount[i] < tagCounts[i]) {
                                    maxTagCount[i] = tagCounts[i];
                            }
                            return;
                    }
            }
    }

    public void properlyQuit() {
            isRunning = false;
    }
}
```

Classe RFIDEventsAdapterNew

```
package phidgetMailDisplayer;

import Phidgets.*;

/**
 * @author Philippe Galley
 * @author Blaise Grand
 * @author Samuel Rossier
 *
 * Phidgets application : Mail displayer
 */

public class RFIDEventsAdapterNew extends  IPhidgetRFIDEventsAdapter {

    private String lastTag = "";

    private MDmodel myModel;

    public RFIDEventsAdapterNew(MDmodel model) {
            myModel = model;
    }

    public void OnTag(_IPhidgetRFIDEvents_OnTagEvent ke) {
            if (!lastTag.equals(ke.get_TagNumber())) {
                    myModel.TAG ID[myModel.currentMail] = ke.get TagNumber();
                    myModel.currentMail++;
            }
            lastTag = ke.get_TagNumber();
    }

}
```

Classe RFIDEventsAdapterOpen

```
package phidgetMailDisplayer;

import Phidgets.*;

/**
 * @author Philippe Galley
 * @author Blaise Grand
 * @author Samuel Rossier
 *
 * Phidgets application : Mail displayer
 */

public class RFIDEventsAdapterOpen extends _IPhidgetRFIDEventsAdapter {

    private MDmodel myModel;

    public RFIDEventsAdapterOpen(MDmodel model) {
            myModel = model;
    }

    public void OnTag( IPhidgetRFIDEvents OnTagEvent ke) {
            int i;
            for (i = 0; i < myModel.TAG ID.length; i++) {
                    if (myModel.TAG_ID[i].equals(ke.get_TagNumber()))
                            break;
            }
            if (i < myModel.TAG ID.length) {
                    myModel.afficheMail(i);
            }
    }
}
```

5. Tests

Nous avons effectué plusieurs tests avec des nombres d'e-mails variables. Pour tous les scénarios envisagés, l'application a réagi selon nos attentes. Nous ne pouvons pas en dire beaucoup plus dans ce chapitre.

6. Conclusion

Ce travail a été bénéfique car il nous a permis de nous rendre compte des techniques d'interfaçage entre une application et le monde réel. Phidget Mail Displayer est un exemple de logiciel gérant aussi bien les "interactions informatiques normales" telles que les clics et les actions du clavier que des interactions avec des objets réels (dans ce cas-ci les tags RFID).

Cette technologie est très prometteuse, notamment pour permettre un accès plus aisé aux outils informatiques à des personnes souffrant de divers handicapes ou pour une utilisation de logiciels spécifiques dans un environnement pas forcément très adéquat pour des ordinateurs.

Philippe Galley **Blaise Grand** **Samuel Rossier**

7. References

phidgets.com
Site officiel Phidget : téléchargements, exemples, explications

PUBLICITE

Combien de fois aller vous sur votre boîte mail ? Combien de fois devez-vous vous connecter au serveur pour savoir si l'élu de votre coeur ou votre futur employeur vous a envoyé un Courriel ? N'est-ce pas énervant ce réflexe se répétant chaque heure au bureau et le soir à la maison ?

Cela vous pose un problème ? Ca tombe bien, nous avons la solution. En effet grâce à Phidget Mail Displayer, c'est la technologie spatiale au service de votre boîte E-mail.

Le Phidget Mail Displayer vous permet de matérialiser vos E-mail à l'aide de jetons. En effet lorsque vous recevez un E-mail, celui-ci ajoute dans le récipient prévu à cet effet un jeton. Cela vous permet d'avoir visuellement une idée du nombre de vos E-mail. Passez le jeton devant le capteur prévu à cette effet, et miracle !!! Le mail s'affiche sur l'écran de votre ordinateur.