ESIREM
ÉCOLE SUPÉRIEURE D'INGÉNIEURS
NUMÉRIQUE ET MATÉRIAUX

École associée Polytech

Université de Bourgogne

TP - Systèmes de télécommunications sans-fils et applications : Carte sans-contact MIFARE Classic

Auteurs :
Clément GHYS
Benjamin MILHET

Professeur :
M. THIVENT

03/01/2023

Table des matières

Introduction

Le but de ce TP est le développement d'une interface graphique pour un lecteur de cartes sans contact Mifare Classic. Le développement se fera avec QT Creator et nous disposons d'une librairie fournie permettant d'utiliser les fonctions associées au lecteur de cartes. L'application doit permettre la connexion entre une carte et le lecteur. L'interface graphique doit comprendre des fonctionnalités permettant d'afficher et de modifier les informations d'identité du processeur de la carte. On souhaite également pouvoir gérer un porte-monnaie et un système d'incrément et de décrément d'unités contenues dans le dit porte-monnaie.

Lien du repo GitHub : TP Carte Sans Contact

Développement de l'interface graphique

2.1 Visuelle de l'application finale

2.1.1 Vision globale

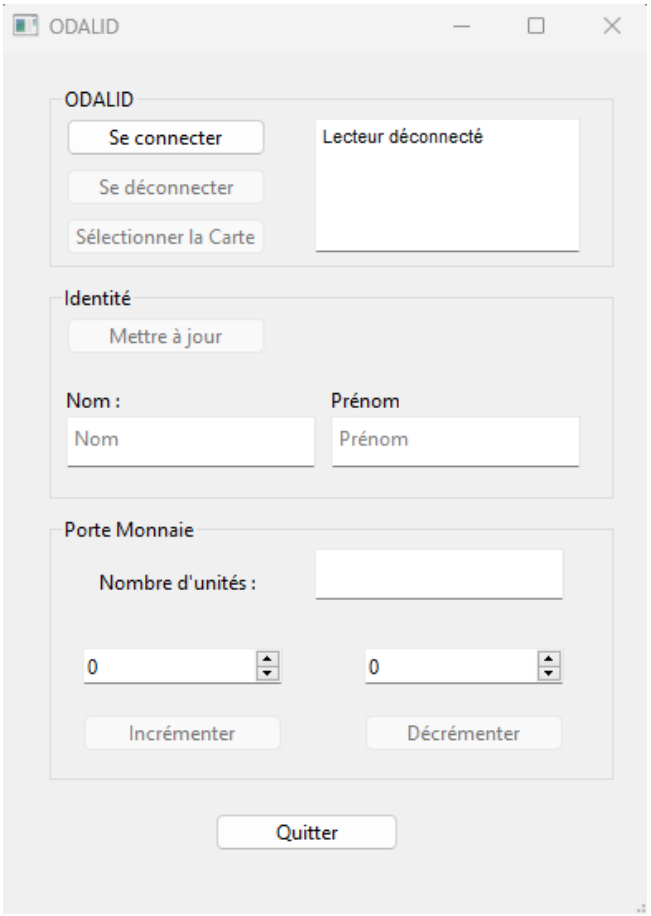

FIGURE 2.1 – Représentation de notre application pour contrôler nôtre lecteur de carte

2.1.2 Détails des différentes parties de l'application

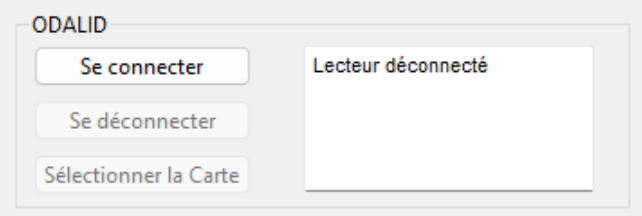

FIGURE 2.2 – Interface pour contrôler le lecteur de carte

Cette partie comporte 3 boutons importants pour se connecter et de déconnecter du lecteur de cartes. Le premier bouton "Se connecter" permet de se connecter au lecteur de cartes. Le bouton "Se déconnecter" permet de se déconnecter du lecteur de cartes. Le troisième et dernier bouton "Sélectionner la carte" permet de récupérer les informations d'une carte sans contact. A droite, il y a une zone d'affichage qui nous permettant d'afficher la version du lecteur ou si nous rencontrons un problème lors de la lecture d'une carte. Tous les boutons sauf le bouton "Se connecter" sont désactiver au lancement de l'application et sont activés au moment où le lecteur de cartes est connecté.

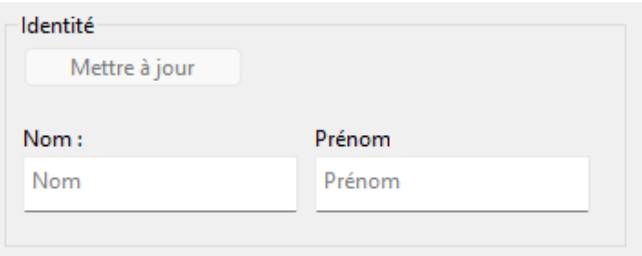

FIGURE 2.3 – Interface de gestion de nom et du prénom du possesseur de la carte

Sur cette interface sont affichés le nom et le prénom du propriétaire de la carte scanner. Il est possible d'éditer directement ces deux paramètres précédents et les mettre à jour en ayant la carte toujours sur le lecteur et d'appuyer sur le bouton "Mettre à jour".

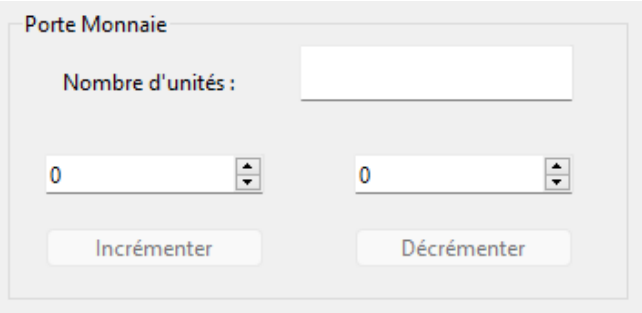

FIGURE 2.4 – Interface de gestion du nombre d'unité sur une carte

Sur cette interface est affiché le nombre d'unité restante sur la carte scanner. Il est possible de renseigner une valeur pour incrémenter ou décrémenter celle-ci puis d'appuyer sur le bouton respectif tout en ayant la carte sur le lecteur.

2.2 Connexion du lecteur de carte

La fonction "on_Connect_clicked", qui s'appelle lors de l'appuie sur le bouton "Se connecter", permet de se connecter au lecteur de carte avec la fonction "OpenCOM" issue de la librairie fournit. Ensuite nous vérifions que celui c'est bien connecter avec la variable status, puis nous pouvons récupérer sa version et permettre de nouvelle action sur l'interface graphique.

```
// Fonction permettant de connecter le lecteur de carte
void MainWindow::on_Connect_clicked() {

    int16_t status = MI_OK;
    MonLecteur.Type = ReaderCDC; // Type de lecteur de carte
    MonLecteur.device = 0; // Numero du lecteur de carte
    status = OpenCOM(&MonLecteur); // Etablissement de la connexion avec le lecteur de carte

    status = Version(&MonLecteur); // Recuperation de la version du lecteur de carte
    if (status == MI_OK){ // Si la connexion est reussie
        ui->Connect->setEnabled(FALSE); // On desactive le bouton de connexion
        status = LEDBuzzer(&MonLecteur, LED_YELLOW_ON); // Allumage de la LED jaune

        // RF field ON
        RF_Power_Control(&MonLecteur, TRUE, 0);

        ui->Affichage->setText(MonLecteur.version); // Affichage de la version du lecteur de carte
        ui->Affichage->update(); // Mise a jour de l'affichage

        this->MelodyBuzzer(); // Jouer une melodie avec le buzzer

        ui->connectCarte->setEnabled(TRUE); // Activation du bouton de connexion avec la carte
        ui->Deconnect->setEnabled(TRUE); // Activation du bouton de deconnexion du lecteur de
            carte

    }
}
```

2.3 Deconnexion du lecteur de carte

La fonction "on_Deconnect_clicked" permet de déconnecter le lecteur de carte avec la fonction "CloseCOM" de la librairie.

```
// Fonction permettant de deconnecter le lecteur de carte
void MainWindow::on_Deconnect_clicked()
{
    RF_Power_Control(&MonLecteur, FALSE, 0);
    status = LEDBuzzer(&MonLecteur, LED_OFF); // Eteindre la LED
    status = CloseCOM(&MonLecteur); // Fermeture de la connexion avec le lecteur de carte

    if (status == MI_OK) { // Si la deconnexion est reussie
        ui->Connect->setEnabled(TRUE); // Activation du bouton de connexion
        ui->Deconnect->setEnabled(FALSE); // Desactivation du bouton de deconnexion
        ui->connectCarte->setEnabled(FALSE); // Desactivation du bouton de connexion avec la carte


        ui->Affichage->setText("Lecteur deconnecte"); // Texte affichant que le lecteur est
            deconnecte
        ui->Affichage->update(); // Mise a jour de l'affichage
    }
}
```

2.4 Lecture d'une carte

La fonction "on_connectCarte_clicked" s'active lors de l'activation du bouton "Sélectionner la carte" sur l'interface graphique et permettant la lecture du contenu d'une carte.

```
// Fonction permettant de recuperer les informations d'une carte MIFARE
void MainWindow::on_connectCarte_clicked()
{
     status = ISO14443_3_A_PollCard(&MonLecteur, atq, sak, uid, &uid_len); // Verification qu'une
         carte est presente

     if (status == MI_OK){ // Si une carte est presente
         ui->Saisie->setEnabled(TRUE); // Activation du bouton de saisie
         ui->incrementButton->setEnabled(TRUE); // Activation du bouton d'incrementation
         ui->decrementButton->setEnabled(TRUE); // Activation du bouton de decrementation

         uint8_t dataPrenom[16];
         status = Mf_Classic_Read_Block(&MonLecteur, TRUE, 9, dataPrenom, AuthKeyA, 2); // Lecture
             du prenom

         uint8_t dataNom[16];
         status = Mf_Classic_Read_Block(&MonLecteur, TRUE, 10, dataNom, AuthKeyA, 2); // Lecture
             du nom


         ui->nom->setText((char*)dataNom); // Affichage du nom
         ui->nom->update(); // Mise a jour de l'affichage

         ui->prenom->setText((char*)dataPrenom); // Affichage du prenom
         ui->prenom->update(); // Mise a jour de l'affichage

         ui->Affichage->setText("Carte detectee"); // Affichage du texte "Carte detectee"
         ui->Affichage->update(); // Mise a jour de l'affichage

        this->actualiserIncrement(); // Actualisation de l'incrementation

        this->tag_hat(); // Jouer un bruit de buzzer de validation
     } else { // Si aucune carte n'est presente
         ui->nom->setText("Erreur lecture carte"); // Affichage du texte "Erreur lecture carte"
         ui->nom->update(); // Mise a jour de l'affichage
```

```
            ui->prenom->setText("Erreur lecture carte"); // Affichage du texte "Erreur lecture carte"
            ui->prenom->update(); // Mise a jour de l'affichage

            ui->nbUnite->setPlainText("Erreur lecture carte"); // Affichage du texte "Erreur lecture
                carte"
            ui->nbUnite->update(); // Mise a jour de l'affichage

            ui->Affichage->setText("Erreur lecture carte"); // Affichage du texte "Erreur lecture
                carte"
            ui->Affichage->update(); // Mise a jour de l'affichage

            ui->Saisie->setEnabled(FALSE); // Desactivation du bouton de saisie
            ui->incrementButton->setEnabled(FALSE); // Desactivation du bouton d'incrementation
            ui->decrementButton->setEnabled(FALSE); // Desactivation du bouton de decrementation
        }
    }
```

2.5 Buzzer et led

La fonction "tag_hat" et une fonction reprise de la documentation fournie permettant lors de son appel de faire "bipper" le buzzer et d'allumer les LEDs.

```
    // Fonction permettant de faire un bruit de buzzer de validation
    void MainWindow::tag_hat(){
        status = LEDBuzzer(&MonLecteur, LED_GREEN_ON+LED_YELLOW_ON+LED_RED_ON+LED_GREEN_ON);
        DELAYS_MS(100);
        status = LEDBuzzer(&MonLecteur, LED_GREEN_ON);
    }
```

2.6 Quitter

La fonction "on_Quitter_clicked" est une fonction qui ferme l'application lorsque l'on clique sur le bouton quitter.

```
    // Fonction permettant de quitter l'application
    void MainWindow::on_Quitter_clicked()
    {
        int16_t status = MI_OK;
        RF_Power_Control(&MonLecteur, FALSE, 0);
        status = LEDBuzzer(&MonLecteur, LED_OFF); // Eteindre la LED
        status = CloseCOM(&MonLecteur); // Fermeture de la connexion avec le lecteur de carte
        qApp->quit(); // Fermeture de l'application
    }
```

2.7 Conversions

Les fonctions "convertirIntToQstring" et "convertirQstringToChar" sont des fonctions convertissent respectivement un en entier en Qstring et une Qstring en char. ces deux fonction permettent des conversions de type qui nous seront utiles lors de l'écriture de données dans la carte.

```
// Fonction permettant de convertir un entier en QString
QString MainWindow::convertirIntToQstring(int monEntier){
    return QString::number(monEntier);
}
```

```
// Fonction permettant de convertir un QString en uint8_t
uint8_t* MainWindow::convertirQstringToChar(QString DataText) {
    char DataIn[240];
    sprintf(DataIn, DataText.toUtf8().data(), 240);
    qDebug() << (uint8_t*)DataIn;
    return (uint8_t*)DataIn;
}
```

2.8 Affichage des informations

La fonction "on_connectCarte_clicked" est la fonction qui permet d'afficher les informations contenues dans la carte. Tout d'abord on rentre dans une condition qui vérifie que le statut et bon ce qui signifie qu'une carte est bien posée sur le lecteur et que l'on peut lire ses données. Si la condition est vérifié on passe les boutons de saisie, d'incrément et décrément en "enable". Il est donc à présent possible d'écrire et de modifier la valeur du porte monnaie. Ensuite on utilise la fonction "Mf_Classic_Read_Block" pour lire le prénom et le nom dans les bon blocs. Enfin on appelle la fonction update pour mettre à jour l'affichage. Si la condition sur le statut n'était pas vérifié les boutons seraient restés désactivés et le message "Erreur lecture carte" se serait affiché dans les zones des saisie.

```
// Fonction permettant de recuperer les informations d'une carte MIFARE
void MainWindow::on_connectCarte_clicked()
{
    status = ISO14443_3_A_PollCard(&MonLecteur, atq, sak, uid, &uid_len); // Verification qu'une
        carte est presente

    if (status == MI_OK){ // Si une carte est presente
        ui->Saisie->setEnabled(TRUE); // Activation du bouton de saisie
        ui->incrementButton->setEnabled(TRUE); // Activation du bouton d'incrementation
        ui->decrementButton->setEnabled(TRUE); // Activation du bouton de decrementation

        uint8_t dataPrenom[16];
        status = Mf_Classic_Read_Block(&MonLecteur, TRUE, 9, dataPrenom, AuthKeyA, 2); // Lecture
            du prenom

        uint8_t dataNom[16];
        status = Mf_Classic_Read_Block(&MonLecteur, TRUE, 10, dataNom, AuthKeyA, 2); // Lecture
            du nom


        ui->nom->setText((char*)dataNom); // Affichage du nom
        ui->nom->update(); // Mise a jour de l'affichage

        ui->prenom->setText((char*)dataPrenom); // Affichage du prenom
        ui->prenom->update(); // Mise a jour de l'affichage

        ui->Affichage->setText("Carte detectee"); // Affichage du texte "Carte detectee"
        ui->Affichage->update(); // Mise a jour de l'affichage

       this->actualiserIncrement(); // Actualisation de l'incrementation

       this->tag_hat(); // Jouer un bruit de buzzer de validation
    } else { // Si aucune carte n'est presente
        ui->nom->setText("Erreur lecture carte"); // Affichage du texte "Erreur lecture carte"
        ui->nom->update(); // Mise a jour de l'affichage

        ui->prenom->setText("Erreur lecture carte"); // Affichage du texte "Erreur lecture carte"
        ui->prenom->update(); // Mise a jour de l'affichage

        ui->nbUnite->setPlainText("Erreur lecture carte"); // Affichage du texte "Erreur lecture
            carte"
        ui->nbUnite->update(); // Mise a jour de l'affichage

        ui->Affichage->setText("Erreur lecture carte"); // Affichage du texte "Erreur lecture
            carte"
        ui->Affichage->update(); // Mise a jour de l'affichage

        ui->Saisie->setEnabled(FALSE); // Desactivation du bouton de saisie
        ui->incrementButton->setEnabled(FALSE); // Desactivation du bouton d'incrementation
        ui->decrementButton->setEnabled(FALSE); // Desactivation du bouton de decrementation
    }
}
```

2.9 Ecriture du Nom et du Prénom

La fonction "on_Saisie_clicked" permet d'enregistrer dans la carte les informations écrites par l'utilisateur dans les cases de saisie du nom et du prénom. Avant toute chose on vérifie que le statut est bon, ensuite on place dans des variables les Qstrings saisient et enfin on écrit dans les blocs correspondants aux nom et prénom les variables correspondantes. Si la condition sur le statut n'était pas vérifiée on affiche "Erreur lecture carte" sur la fenêtre de saisie et on enregistre aucune information.

```
// Fonction permettant d'actualiser les informations d'une carte MIFARE
void MainWindow::on_Saisie_clicked()
{
    status = ISO14443_3_A_PollCard(&MonLecteur, atq, sak, uid, &uid_len); // Verification qu'une
        carte est presente

    if (status == MI_OK) { // Si une carte est presente
        QString nomText = ui->nom->toPlainText(); // Recuperation du nom
        QString prenomText = ui->prenom->toPlainText(); // Recuperation du prenom

        char DataInPrenom[16];
        sprintf(DataInPrenom, prenomText.toUtf8().data(), 16); // Conversion du prenom en char
        auto dataPrenom = (uint8_t*)DataInPrenom;

        char DataInNom[16];
        sprintf(DataInNom, nomText.toUtf8().data(), 16); // Conversion du nom en char
        auto dataNom = (uint8_t*)DataInNom;

        status = Mf_Classic_Write_Block(&MonLecteur, TRUE, 9, dataPrenom, AuthKeyB, 2); //
            Ecriture du prenom dans la carte
        status = Mf_Classic_Write_Block(&MonLecteur, TRUE, 10, dataNom, AuthKeyB, 2); // Ecriture
            du nom dans la carte

        this->tag_hat(); // Jouer un bruit de buzzer de validation
    } else { // Si aucune carte n'est presente
        ui->Affichage->setText("Erreur lecture carte"); // Affichage du texte "Erreur lecture
            carte"
        ui->Affichage->update(); // Mise a jour de l'affichage
    }
}
```

2.10 Incrément et Décrément

La fonction "on_incrementButton_clicked" utilise les fonctions "Mf_Classic_Increment_Value" et "Mf_Classic_Restore pour incrémenter la valeur du porte-monnaie et enregistrer cette valeur dans un bloc de backup de la carte lorsque l'on clique sur le bouton. Idem avec pour la fonction "on_decrementButton_clicked" pour le décrément.

```
// Fonction permettant d'actualiser l'incrementation
void MainWindow::on_incrementButton_clicked()
{
    status = ISO14443_3_A_PollCard(&MonLecteur, atq, sak, uid, &uid_len); // Verification qu'une
        carte est presente
    if (status == MI_OK) { // Si une carte est presente
        auto dataCompteur = ui->increment->value(); // Recuperation de la valeur de
            l'incrementation
        status = Mf_Classic_Increment_Value(&MonLecteur, TRUE, 14, dataCompteur, 13, AuthKeyB, 3);
            // Incrementation de la valeur du compteur
        status= Mf_Classic_Restore_Value(&MonLecteur, TRUE, 13, 14, AuthKeyA, 3); // Restauration
            de la valeur du compteur

        this->actualiserIncrement(); // Actualisation de l'incrementation
        this->tag_hat(); // Jouer un bruit de buzzer de validation
    } else { // Si aucune carte n'est presente
        ui->Affichage->setText("Erreur lecture carte"); // Affichage du texte "Erreur lecture
            carte"
        ui->Affichage->update(); // Mise a jour de l'affichage
    }
}
```

```
// Fonction permettant d'actualiser la decrementation
void MainWindow::on_decrementButton_clicked()
{
    status = ISO14443_3_A_PollCard(&MonLecteur, atq, sak, uid, &uid_len); // Verification qu'une
        carte est presente
    if (status == MI_OK) { // Si une carte est presente
        auto dataCompteur = ui->decrement->value(); // Recuperation de la valeur de la
            decrementation
        status = Mf_Classic_Decrement_Value(&MonLecteur, TRUE, 14, dataCompteur, 13, AuthKeyA, 3);
            // Decrementation de la valeur du compteur
        status= Mf_Classic_Restore_Value(&MonLecteur, TRUE, 13, 14, AuthKeyA, 3); // Restauration
            de la valeur du compteur

        this->actualiserIncrement(); // Actualisation de la decrementation
        this->tag_hat(); // Jouer un bruit de buzzer de validation
    } else { // Si aucune carte n'est presente
        ui->Affichage->setText("Erreur lecture carte"); // Affichage du texte "Erreur lecture
            carte"
        ui->Affichage->update(); // Mise a jour de l'affichage
    }
}
```

2.11 Actualisation de l'incrément

La fonction "actualiserIncrement" permet d'actualiser la valeur du porte-monnaie lorsque l'on souhaite incrémenter ou décrémenter.

```
// Fonction permettant d'actualiser l'affichage de la valeur du compteur
void MainWindow::actualiserIncrement(){
    ui->nbUnite->setPlainText(this->convertirIntToQstring(this->getNbUniteRestante())); //
        Actualisation de l'affichage de la valeur du compteur
    ui->nbUnite->update(); // Mise a jour de l'affichage
}
```

2.12 Lecture du porte monnaie

La fonction "getNbUniteRestante" permet de retourner la valeur du porte monnaie en allant chercher l'information contenue dans la carte grâce à la fonction "Mf_Classic_Read_Value".

```
// Fonction permettant de recuperer la valeur du compteur dans la carte
int MainWindow::getNbUniteRestante(){
    uint32_t dataCompteur = 0; // Initialisation de la variable dataCompteur
    status = Mf_Classic_Read_Value(&MonLecteur, TRUE, 14, &dataCompteur, AuthKeyA, 3); // Lecture
        de la valeur du compteur

    return dataCompteur; // Retourne la valeur du compteur
}
```

2.13 Son de lancement

En bonus nous avons créer un petit son de lancement lorsque la carte se connecte. Il s'agit de la fonction "MelodyBuzzer".

```
// Fonction permettant de jouer une petite melodie de buzzer
void MainWindow::MelodyBuzzer() {
    for (int i = 5; i > 0 ; i--) {
        status = LEDBuzzer(&MonLecteur, LED_GREEN_ON+LED_YELLOW_ON+LED_RED_ON+LED_GREEN_ON);
        DELAYS_MS(30*i);
        status = LEDBuzzer(&MonLecteur, LED_GREEN_ON);
        DELAYS_MS(30*i);
        status = LEDBuzzer(&MonLecteur, LED_GREEN_ON+LED_YELLOW_ON+LED_RED_ON+LED_GREEN_ON);
        DELAYS_MS(15*i);
        status = LEDBuzzer(&MonLecteur, LED_GREEN_ON);
        DELAYS_MS(30*i);
    }
}
```

Conclusion

Ce projet nous a permis d'approfondir nos connaissances avec le langage C++ et le framework Qt Creator et d'utiliser nos connaissances vues pendant les cours afin de réaliser une application visuelle permettant de contrôler un lecteur de cartes sans contact. Nous avions à notre disposition une librairie qui nous était nouvelle et nous devions chercher à travers sa documentation l'ensemble des fonctions qui nous a été utile pur mener à bien ce projet. Ce projet nous a permis de combiner les notions vues en cours et en TD sur le comportement et le fonctionnement des cartes MIFARE et le fonctionnement de leur mémoire.