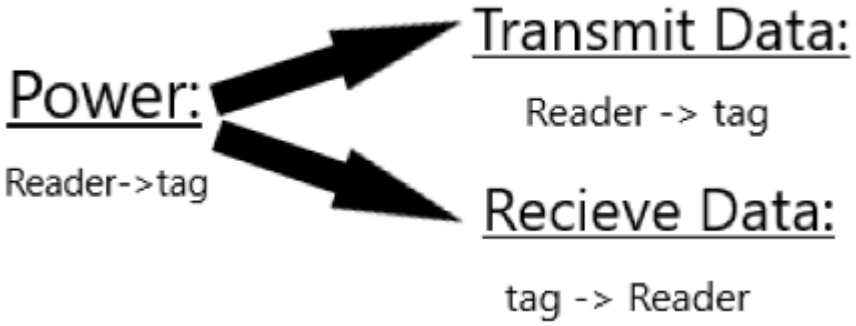
Isnard Achil
Pocholle Alexy

SE

Compte-Rendu TP de communication sans-fils

Fonctionnement transmission RFID:



Pour commencer nous devons établir une connection avec le lecteur:

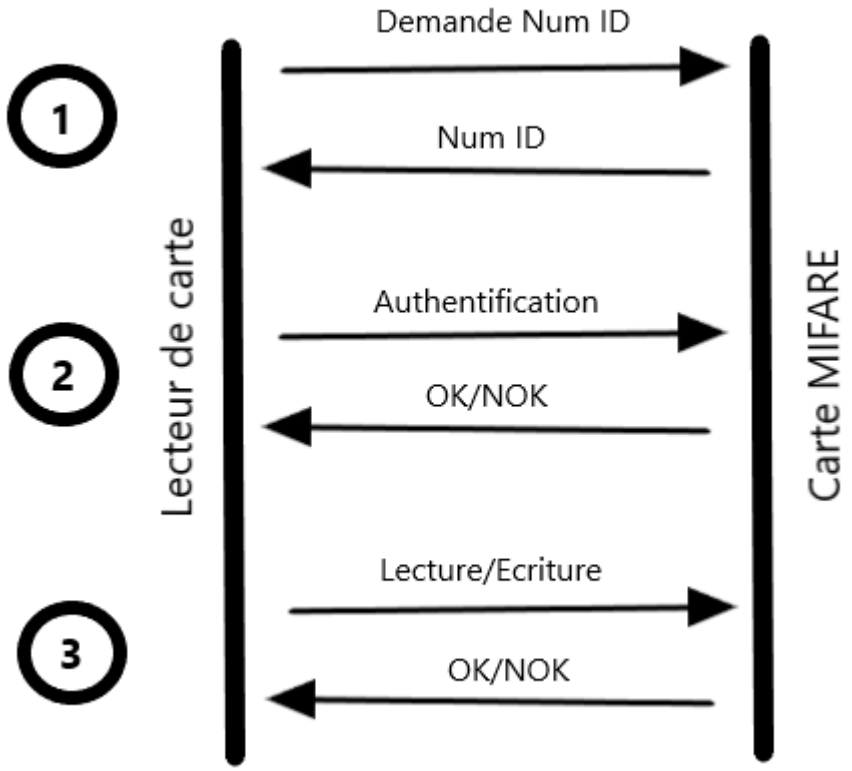
```
OpenCOM(&MonLecteur);
```

Ensuite nous devon activer(TRUE) le lecteur:

```
RF_Power_Control(&MonLecteur, TRUE, 0);
```

remarque:Pour le désactiver lorsque nous quittons l'application nous renvoyons FALSE à la place de TRUE;

Étapes à réaliser pour l'échange d'information entre le lecteur et la carte:



1) Prise de contacte avec la carte: *ISO14443_3_A_PollCard*

```
ISO14443_3_A_PollCard(&MonLecteur, atq, sak, uid, &uid_len);
```

Cette fonction nous renvoie un statut (= 0 prise de contact réussi).

2)Authentification:

pas de demande d'authentification (Mf_Classic_Authenticate) car nous passons un boolean "TRUE" dans nos fonctions lecture et écriture

3) Opérations sur la mémoire: Il faut à chaque fois préciser le **Secteur** et le **Bloc** sur lequel on opère. Les clefs d'authentification(**AuthKey**) changent en fonction des opérations réalisées on utilise KeyA ou KeyB. Toutes nos fonctions nous renvoient un **statut** qui nous indique si l'opération a été réalisée. Nous utilisons ce statut pour afficher dans la console le bon fonctionnement des opérations.

- **Read:**

```
char dataText[240] = {0};
Mf_Classic_Read_Block(&MonLecteur, TRUE, 10, (uint8_t*)dataText, AuthKeyA, 2);
```

On transmet l'adresse de nos Data (dataText) à la fonction pour qu'elle modifie directement cette variable.

- **Write:**

```
char DataIn[16];
Mf_Classic_Write_Block(&MonLecteur, Authentication=TRUE, Block, (uint8_t*)DataIn,
AuthKey, Secteur);
```

On transmet l'adresse de nos Data (DataIn) à la fonction pour qu'elle modifie directement cette variable.

- **Increment:**

Avant d'incrémenter notre compteur on le sauvegarde dans la block de backup :

```
uint32_t dataNum = 0;
Mf_Classic_Read_Value(&MonLecteur, Authentication=TRUE, Block du compteur,
&dataNum, AuthKeyA, Secteur);
```

-> Lecture

```
Mf_Classic_Write_Value(&MonLecteur, Authentication=TRUE, block de la Backup,
dataNum, AuthKeyB, Secteur);
```

-> Ecriture

```
Mf_Classic_Increment_Value(&MonLecteur, Authentication=TRUE, Block, value, Block,
AuthKeyB, Secteur);
```

-> Incrémentation

- **Decrement:**

Avant de décrémenter notre compteur on le sauvegarde dans la block de backup :

```
uint32_t dataNum = 0;
Mf_Classic_Read_Value(&MonLecteur, Authentication=TRUE, Block du compteur,
&dataNum, AuthKeyA, Secteur);
```

-> Lecture

```
Mf_Classic_Write_Value(&MonLecteur, Authentication=TRUE, block de la Backup,
dataNum, AuthKeyB, Secteur);
```

-> Ecriture

```
Mf_Classic_Decrement_Value(&MonLecteur, Authentication=TRUE, Block, value, Block,
AuthKeyB, Secteur);
```

-> Décrémentation.

- **Restore:**

Cette fonction nous permet de récupérer les data dans le bloc backup et de l'insérer dans le bloc du compteur:

```
Mf_Classic_Restore_Value(&MonLecteur, Authentication=TRUE, block du Compteur, block de la Backup, AuthKeyB, Secteur);
```

- **Fonction de buzzer:**

  Cette fonction nous permet de faire un signal sonor lorsque notre opération est correctement réaliser:

```
void buzzer(){
	LEDBuzzer(&MonLecteur, LED_GREEN_ON+LED_YELLOW_ON+LED_RED_ON+LED_GREEN_ON);
	DELAYS_MS(10);
	LEDBuzzer(&MonLecteur, LED_RED_ON);
}
```

Conclusion:

Ce tp nous a permis de mieux comprendre les fonctions présentées en cours. Il nous a aussi permis de comprendre les échanges d'informations entre le lecteur et la carte vu en cours.