Le codage RFID à la loupe

DIDIER LE PAPE *

Le codage numérique de l'information fait partie des programmes de SI (« système de numération, codage ») et de STI2D (« traitement de l'information »). L'expérimentation reste la voie la plus ludique afin de bien comprendre les subtilités de conversion. Nous vous proposons d'aborder, à travers cet article, l'étude d'un module RFID.

MOTS-CLÉS

programmation, codage

RFID est l'acronyme de *Radio Frequency IDentification*. En français, cela correspond à « identification par radiofréquence ».

Cette technologie permet, grâce à un émetteur/récepteur adapté, d'identifier un objet sur lequel est disposé un « tag RFID » (appelé aussi « transpondeur ») contenant les informations à transmettre (un numéro d'identification de carte, par exemple) 1.

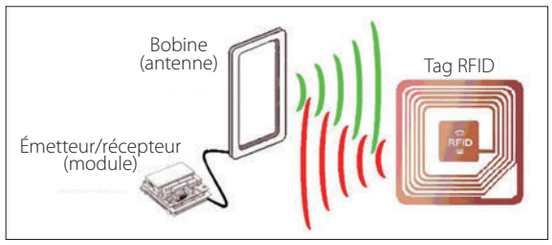
Le principe est le suivant : l'émetteur crée un champ magnétique grâce à une bobine alimentée en courant alternatif. En retour, le tag RFID module ce champ afin de transmettre une information qui sera reconnue par le récepteur (voir aussi l'article « Les technologies RFID », *Technologie*, n° 195).

Les applications sont multiples : cartes ou badges d'accès, étiquettes pour vêtement, étiquettes adhésives pour colis… 2.

Mise en œuvre du module RFID et premier décodage

Le support de travail est le lecteur RFID 125 kHz de SeeedStudio 3. Il sera utilisé avec un des tags RFID vendus par lot (trois badges et deux cartes) 4. Le code fourni par SeeedStudio est le suivant (voir ci-contre).

Afin de récupérer un signal, il faut connecter le lecteur RFID à la broche D2 de la base Shield implantée sur la carte Arduino.



1 Contexte d'utilisation

```
#include <SoftwareSerial.h>

SoftwareSerial SoftSerial(2, 3);
unsigned char buffer[64];
int count=O;

void setup()
{
    SoftSerial.begin(9600);
    Serial.begin(9600);
}
void loop()
{
    if (SoftSerial.available())
    {
        while(SoftSerial.available())
        {
            buffer[count++]=SoftSerial.read();
            if{count == 64)break;
        }
        Serial.write{buffer,count);
        clearBufferArray();
        count = O;
    }
    if (Serial.available())
    SoftSerial.write(Serial.read());
}
void clearBufferArray()
{
    for {int i=O; i<count;i++)
    {
        buffer[i]=NULL;
        }
}
```

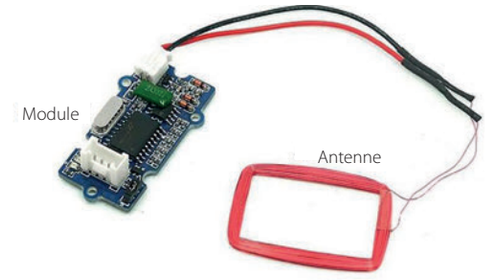
Le test est effectué avec une carte identifiée par le numéro 0010249295 5. Lorsqu'on passe la carte sur l'antenne du lecteur, une LED s'allume sur le module et, simultanément, un code est affiché sur le moniteur série de l'IDE Arduino 6. Afin de relier ce code à celui de la carte, il est nécessaire de lire le document constructeur du lecteur « sen11425p » 7.

D'après cette documentation, le code affiché est composé :

– de dix octets de données correspondants à 10 caractères en code ASCII (« 10 ASCII Data Characters »);

* Enseignant agrégé SII, option Ingénierie mécanique, lycée Louis-le-Grand, Paris.

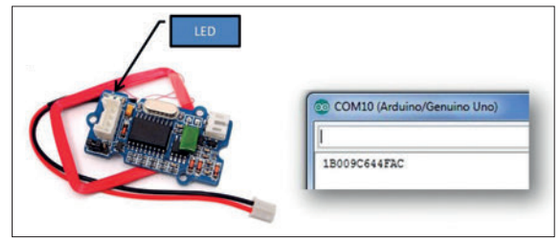
2 Exemples d'utilisation (avec l'aimable autorisation des entreprises Micro-Be, Décathlon et Inotech)



3 Lecteur RFID de SeedStudio

4 Tags RFID

5 Carte utilisée pour les tests



6 Résultat d'identification de la carte

– de deux octets de contrôle (« Checksum – 2 bytes ») obtenus en appliquant la fonction logique « OU exclusif » (« XOR ») successivement à des paires de chiffres en hexadécimal.

On prend comme hypothèse que les octets de début (*start*) et de fin (*end*) ne sont pas transcrits pour affichage.

Vérifions tout d'abord la valeur des deux octets de contrôle : (0x**1B**) XOR (0x**00**) XOR (0x**9C**) XOR (0x**64**) XOR (0x**4F**) = 0x**AC**.

Cette opération peut être effectuée grâce à la calculatrice Windows en mode « programmeur » 8. Le résultat obtenu, 0x**AC**, correspond bien aux deux derniers caractères hexadécimaux du code affiché (les caractères **A** et **C** étant transmis en ASCII étendu, il s'agit bien de deux octets).

Vérifions ensuite le code de la carte : il est écrit que ce sont les quatre derniers octets (*later 4 bytes*) transcrits en décimal qui constituent le code de la carte. Il faut ici comprendre qu'il s'agit des quatre derniers octets constitués par les valeurs hexadécimales **00 9C 64 4F**. En décimal, ce nombre vaut 10 249 295, ce qui correspond bien au code écrit sur la carte (de la même façon que précédemment, cette conversion peut être effectuée grâce à la calculatrice Windows).

On remarque qu'il y a un deuxième code écrit sur la carte : 156,25679. La signification de ce code peut être retrouvée en coupant le code hexadécimal en deux : les deux premiers octets (0x**009C**) ont pour valeur décimale 156 et les deux autres (0x**644F**) ont pour valeur décimale 25 679.

Output Data Format

0x02	10ASCII Data Characters	Checksum	0x03

0x02 - 1 byte start flag

10 ASCII Data Characters – Card number info

Checksum - 2 bytes

0x03 - 1 byte end flag

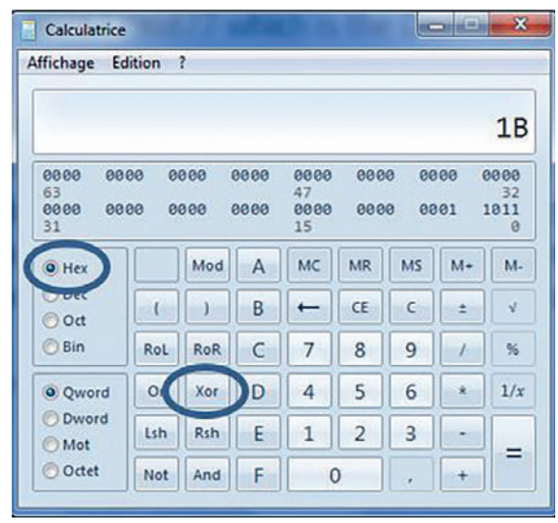
Example

Card number: 62E3086CED

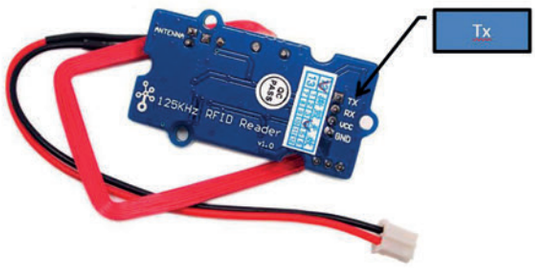
Checksum:(62H)XOR(E3H)XOR(08H)XOR(6CH)XOR(EDH)=08H

Note: The 10 ASCII characters grouped as 5 hex data needs to be further processed as you may find that the 5 hex data is not equal to the number marked on the tags in Decimal. Actually the tag number is equal to the later 4 bytes in decimal. For example, the card number is 62E3086CED, the corresponding number marked on the tag should be 60717296877 which is the Decimal format of E3086CED.

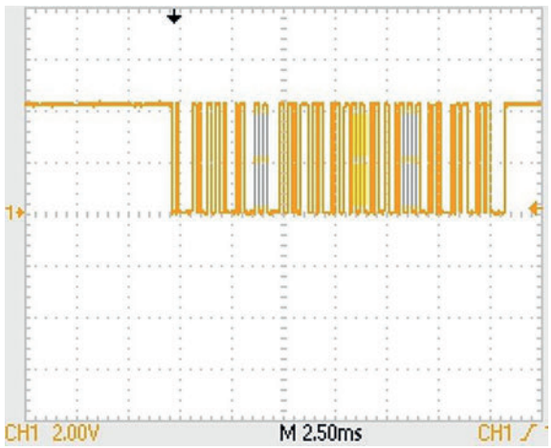
7 Extrait du document constructeur « sen11425p »



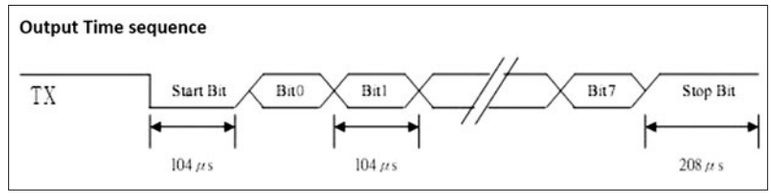
8 Opération « OU exclusif » avec la calculatrice Windows



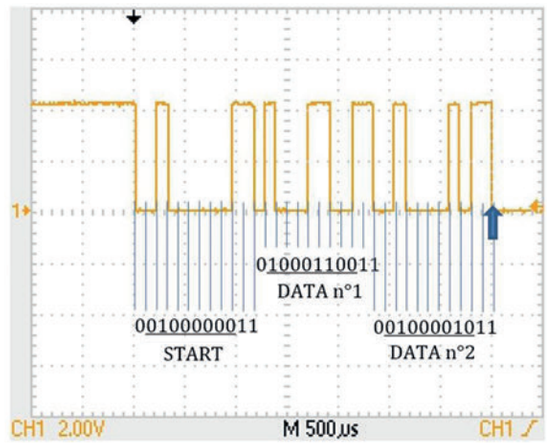
9 Point de mesure

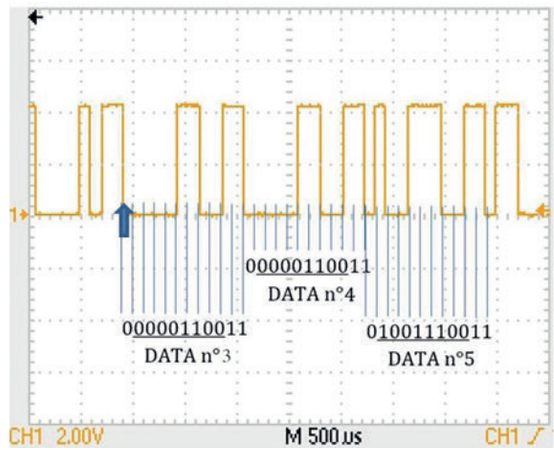


10 Trame RFID



11 Extrait du document constructeur « sen11425p »





12 Début et suite de trame

Relevé de la trame numérique avec un oscilloscope

D'un point de vue physique, on doit pouvoir retrouver par mesure la transmission des informations numériques. Pour cela, on relève avec un oscilloscope le signal sur la broche Tx en prenant la masse sur le GND (*Ground*) de la carte Arduino 9. Une trame est générée à chaque fois que la LED présente sur le module clignote (environ une fois par seconde) 10.

Pour savoir comment sont codés les 0 et les 1 dans la trame, il faut une nouvelle fois lire le document constructeur du lecteur « sen11425p » 11. D'après la documentation, la transmission d'un octet s'effectue :

– avec un bit de *start* et deux bits de *stop* ;
– en envoyant le LSB en premier.

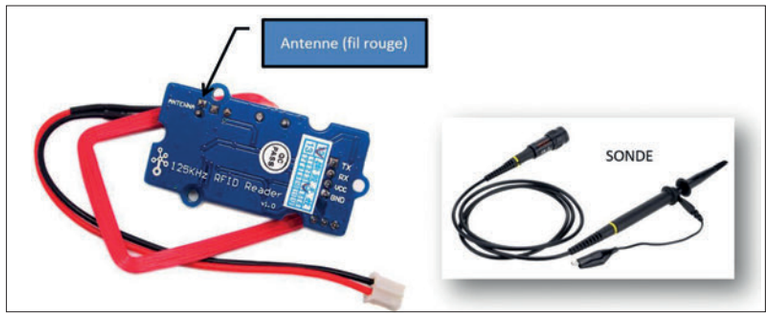
On rappelle que la trame complète commence par un octet de START, puis continue avec dix octets correspondant aux dix caractères de DATA codés en ASCII. En paramétrant la base de temps à 500 ms, le début de transmission est illustré en figure 12.

En prenant comme hypothèse un codage à 1 pour un niveau haut, on obtient les relevés du tableau 13 pour les dix octets de données.

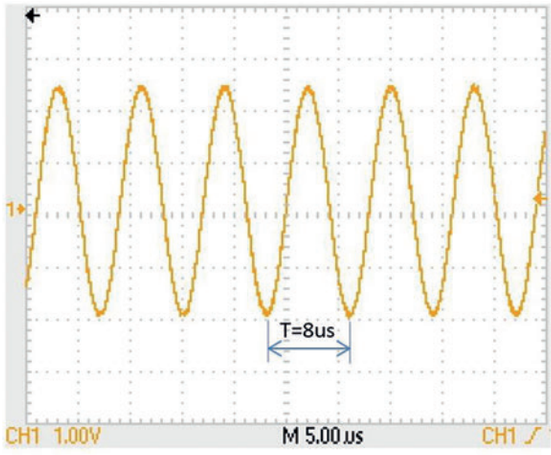
On retrouve bien le code déjà affiché sur le moniteur série lors de la mise en œuvre du module : **1B 00 9C 64 4F**.

Octet DATA	Code relevé	Caractère ASCII correspondant
n° 1	0011 0001	1
n° 2	0100 0010	B
n° 3	0011 0000	0
n° 4	0011 0000	0
n° 5	0011 1001	9
n° 6	0100 0011	C
n° 7	0011 0110	6
n° 8	0011 0100	4
n° 9	0011 0100	4
n° 10	0010 0110	F

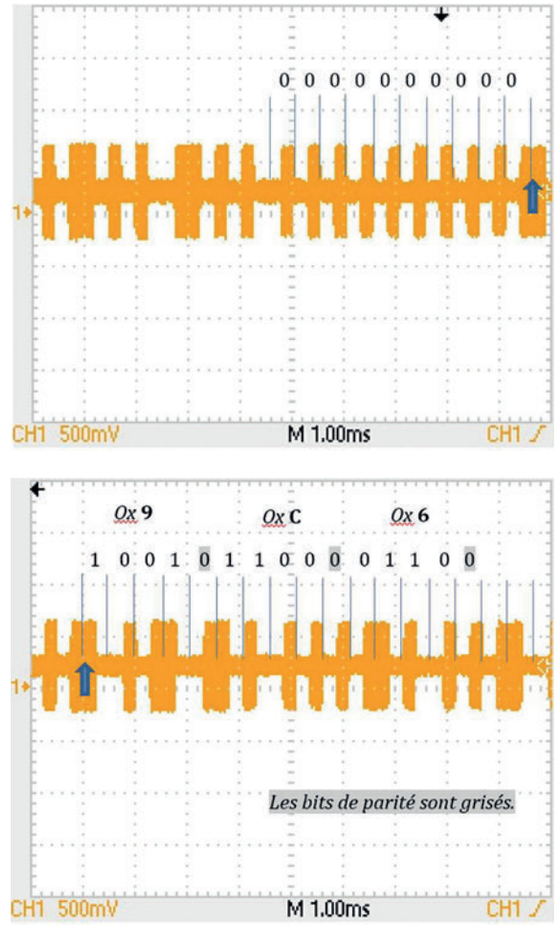
13 Relevé des valeurs sur la trame



14 Point de mesure de l'antenne



15 Signal modulant



16 Début et suite du signal RFID modulé

Relevé du signal modulé avec un oscilloscope

Il est possible de remonter la chaîne d'information pour se placer au niveau de l'antenne de réception du module. On devrait relever un signal alternatif modulé de telle façon que le code correspondant à la carte soit transmis avant son traitement par le module.

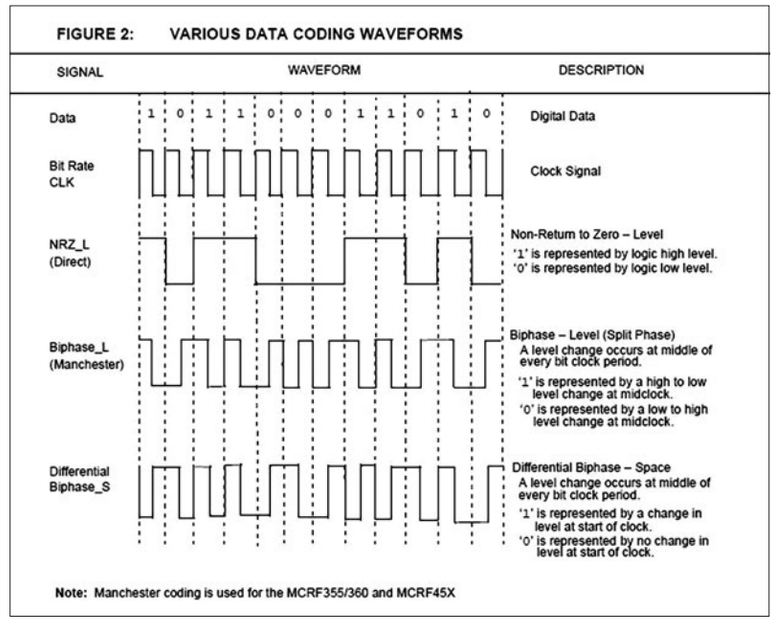
Pour les relevés, il faut cette fois se placer au niveau du fil rouge de l'antenne et utiliser une sonde de mesure (atténuation de 10 nécessaire, car sinon le signal sort de la fenêtre de l'oscilloscope). On prend une nouvelle fois la masse sur le GND de la carte Arduino 14.

Si on ne place aucun tag RFID sur l'antenne, le signal n'est pas modulé. Ainsi, on doit obtenir un signal alternatif de fréquence fixe 125 kHz (d'où le nom du module !).

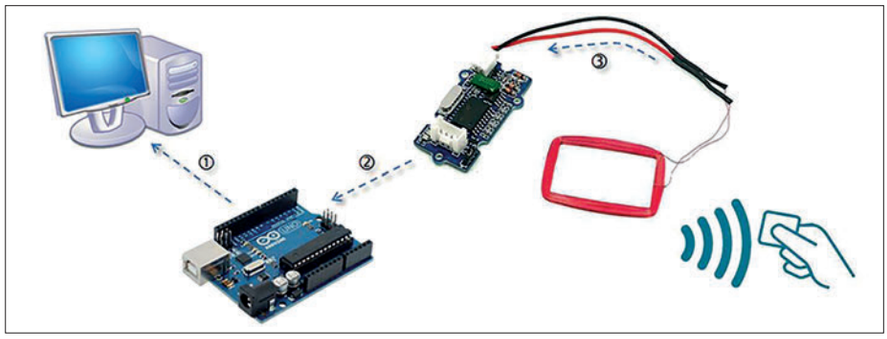
C'est bien le cas d'après le relevé 15, puisqu'on observe une période de 8 µs, soit une fréquence de 125 kHz.

REMARQUE

La sensibilité verticale de 1 V correspond en réalité à 10 V, car on utilise une sonde avec une atténuation de 10. On peut demander à l'oscilloscope d'ajouter une amplification de 10 pour retrouver le niveau réel, mais cela peut créer une confusion inutile, d'autant plus que le niveau réel de tension n'est pas important pour le décodage. Cette remarque reste vraie pour tous les relevés suivants.



17 Extrait du document constructeur « microID 125 kHz RFID »



18 Synthèse des études effectuées

Si on place la carte contenant le tag RFID à proximité de l'antenne, le signal est modulé en permanence. C'est une modulation d'amplitude qui ne peut donc pas se voir avec une base de temps aussi faible que précédemment. En augmentant la base de temps à 5 ms puis en bloquant la fenêtre avant de passer à 1 ms, on peut visualiser le signal par morceaux 16.

D'après la forme obtenue, la modulation est bien une modulation d'amplitude. Pour savoir quel est le codage utilisé, il faut se référer au document constructeur « microID 125 kHz RFID » relatif à la puce électronique contenue dans le tag RFID 17.

En observant la forme de nos relevés, on voit donc qu'il y a deux possibilités pour le codage utilisé : *Biphase-Level (Manchester)* ou *Differential Biphase-Space*.

À ce stade, une autre question se pose : quelle forme prend la trame ? En d'autres termes : y a-t-il huit bits envoyés, avec un bit de *start*, un bit de *stop*, un bit de parité, en ASCII, en hexadécimal, etc. ?

En cherchant judicieusement sur internet, on peut recouper certaines informations et apporter la réponse suivante : le codage est en Manchester, chaque caractère est codé directement en hexadécimal (donc sur quatre bits) avec, en plus, un bit de parité paire.

Et ça marche ! Le début de la transmission peut être repéré grâce à une succession de 10 bits à 0. Suivent les codes hexadécimaux relatifs à **9C6**, chaque code possédant un bit de parité. La suite de la trame, non représentée, permet de retrouver les codes relatifs à **44F**.

Conclusion

Comme on peut le voir, le module RFID étudié est très riche d'un point de vue codage numérique. L'étude peut être réalisée à différents niveaux de complexité, en remontant la chaîne d'information. Ainsi, le travail de décodage peut concerner le numéro affiché sur l'écran de l'ordinateur (repère ①), la trame numérique après traitement par le module (repère ②) ou le signal modulé avant traitement (repère ③) 18.

Pour quelques euros, ce petit module peut donc s'avérer être un vrai support d'expérimentation, d'autant plus qu'il est l'image d'une technologie actuelle et en développement. ■

EN LIGNE

www.centrenational-rfid.com

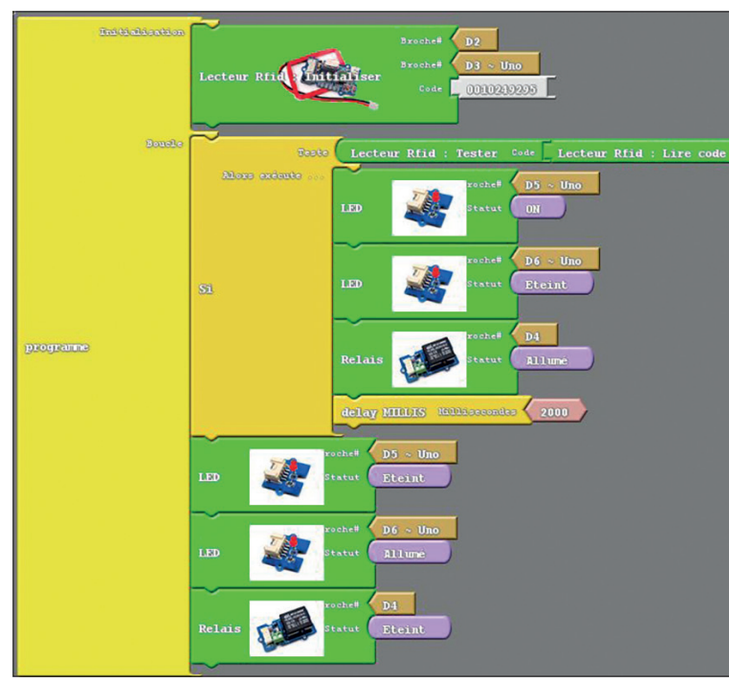
www.seeedstudio.com/wiki/Grove-125KHz_RFID_Reader

Tous les liens sur **http://eduscol.education.fr/sti/revue-technologie**

Lecture RFID avec ArduBlock

Pour ceux qui utilisent plus facilement la programmation par blocs, voici un exemple réalisé avec ArduBlock : nous mettons ici en évidence comment actionner un relais et commander deux LED lorsque la carte est reconnue par le module RFID.

L'initialisation du module connecté aux broches D2 et D3 comprend le numéro de la carte ; il suffit simplement de comparer ce numéro avec celui de la carte en présence dans la boucle centrale pour allumer une LED verte sur D5, éteindre une LED rouge sur D6 et actionner un relais sur D4 pendant 2 s 19. À défaut, la LED verte est éteinte, la rouge allumée et le relais au repos.



19 Code Ardublock de notre exemple