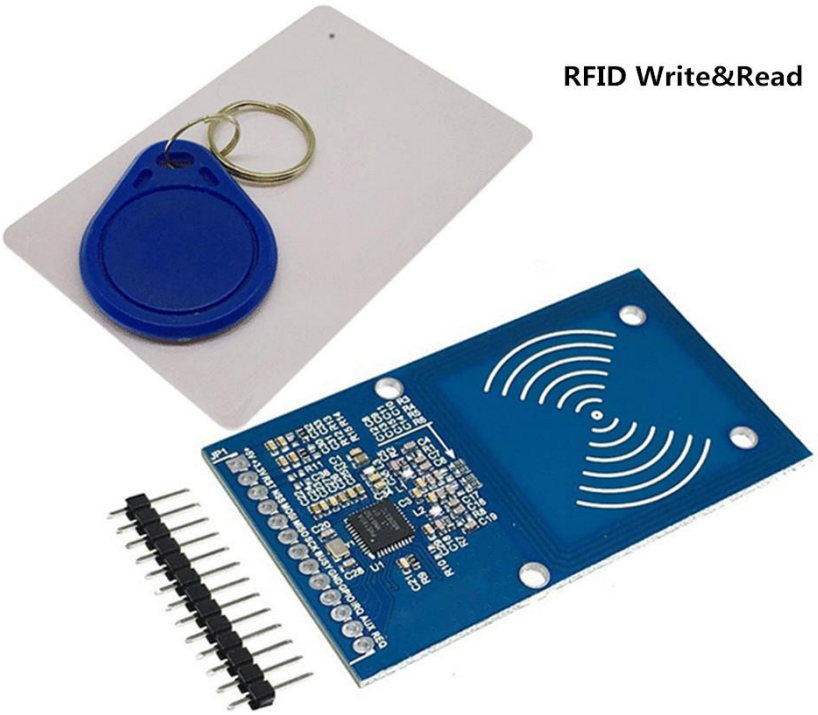
Projet Capteur Lecteur RFID

(rapport_final)

CAUSSE Axel - MARCEAU Axel



SOMMAIRE

I) Informations sur le capteur RFID

1) documentation technique

Les documentations techniques utilisées dans le projet sont disponibles aux liens suivants :

- https://joy-it.net/files/files/Produkte/SBC-RFID-RC522/SBC-RFID-RC522-Manual-09-06-2020.pdf
- https://www.gotronic.fr/pj2-sbc-rfid-rc522-fr-1439.pdf
- http://www.handsontec.com/dataspecs/RC522.pdf
- https://www.data-media.gr/files/KS0067.pdf

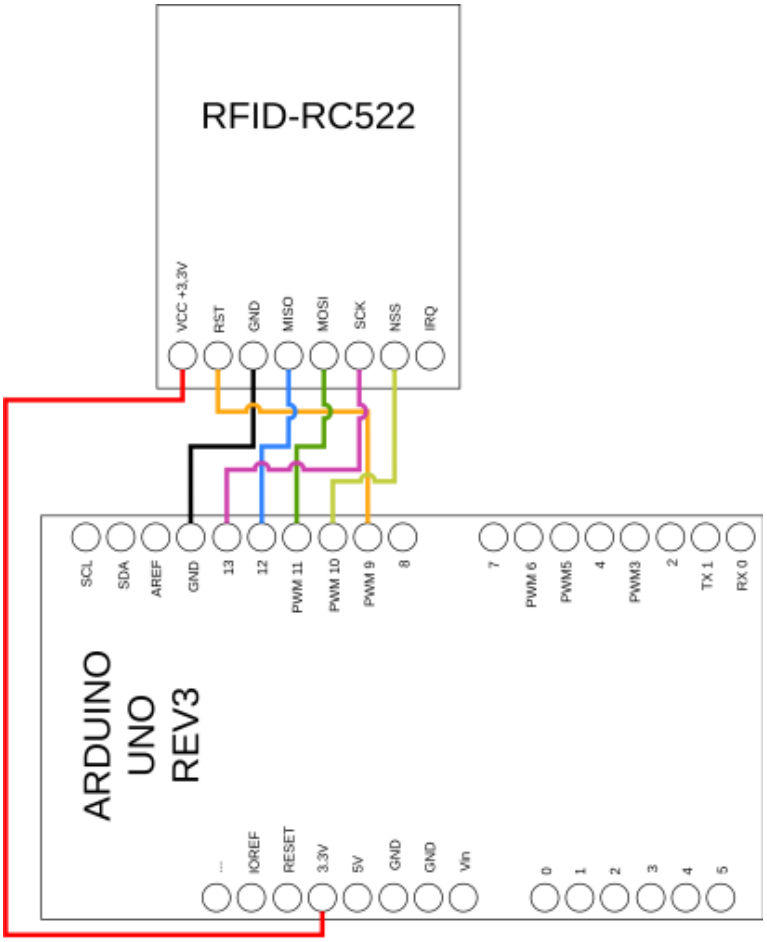
Ces documentations nous ont permis de comprendre le fonctionnement général du capteur RFID ainsi que de nous communiquer un programme basique permettant son démarrage :

```
1 #include <SPI.h>
2 #include <MFRC522.h>
3 #define RST_PIN 9
4 #define SS_PIN 10
5
6  MFRC522 mfrc522(SS_PIN, RST_PIN);
7
8  void setup() {
9    /Serial.begin(9600);
10    wile (!serial);
11    SPI.begin();
12    mfrc522.PCD_Init();
13    mfrc522.PCD_DumpVersionToSerial();
14    Serial.println(F("Scan PICC to see UID, type, and data blocks ..."));
15 }
16
17 void loop() {
18   if(!mfrc522.PICC_IsNewCardPresent()){
19   return;
20  }
21   if(!mfrc522.PICC_IsNewCardSerial()){
22   return;
23  }
24
25  mfrc522.PICC_DumpToSerial(&'mfrc522.uid));
26 }
```

syntax highlighted by Code2HTML, v. 0.9.1

Les datasheets nous on aussi communiqué les différentes librairies requises sur le logiciel Arduino pour une bonne communication avec le capteur, comme par exemple la librairie “MFRC522”, spécialement adaptée à notre capteur.

Figure 1 : Schéma de câblage de la carte du RFID sur l'Arduino UNO



La tension d'alimentation du capteur est égale à 3.3 V en courant continu, autrement dit, on peut alimenter la carte du RFID directement avec l'alimentation de la carte Arduino.

Tout comme l'alimentation, les ports d'entrées/sorties supportent du 3.3V

D'après les datasheets, ce type de capteur peut détecter un badge entre 0 et 10cm, avec une performance optimale de détection à 1cm.
La fréquence de lecture des badge a été réglée à 1Hz.

2) informations annexes

La fonction principale de ce capteur est la détection et la lecture d'informations codées sur des supports adaptés (carte, badge, etc ...) grâce à la technologie RFID.

En général ce type de capteur est utilisé à des fins de sécurité, par exemple pour le contrôle d'identité à l'entrée de zones à accès restreint, dans des ateliers, des entreprises, des bureaux etc.

La communication entre le RFID et l'Arduino s'effectue grâce à un bus de données SPI.

En ce qui concerne les tarifs de ce capteur, il est affiché au prix moyen de 11,90€ chez des fournisseurs tels que "Joy-It" (seul dont nous avons trouvé le nom), ou 1€ chez des revendeurs chinois tiers.

II) Projet et programme

1) Buts du projet

Le but général du projet est d'utiliser ce capteur à des fins de sécurité à l'entrée d'une zone restreinte.
Le capteur détermine si le badge présenté est autorisé ou pas à entrer dans la zone restreinte.

Un bouton permet à un opérateur de répertorier automatiquement un badge présenté dans la base de données d'accès, sans avoir à effectuer une reprogrammation complète du capteur.

Toutes les données des différents badges sont ainsi stockées dans des tableaux en 2 dimensions.

2) Exécution du projet

La première séance nous a permis de découvrir le capteur que nous avions choisi, le RFID.
Elle a aussi permis de coder une première ébauche du programme puis de la tester lors de cette séance.

La recherche de datasheets détaillées s'est avérée assez difficile car la plupart des documentations se concentrent sur l'explication de la mise en route du capteur et non sur les données techniques précises (pour répondre aux questions du rapport préliminaire).

Après avoir fait une demande d'accès à la salle de cours pour travailler, nous nous sommes réparti le travail, 1 de nous travaillant plus sur la programmation pendant que l'autre travaille sur la rédaction du rapport afin de gagner du temps.

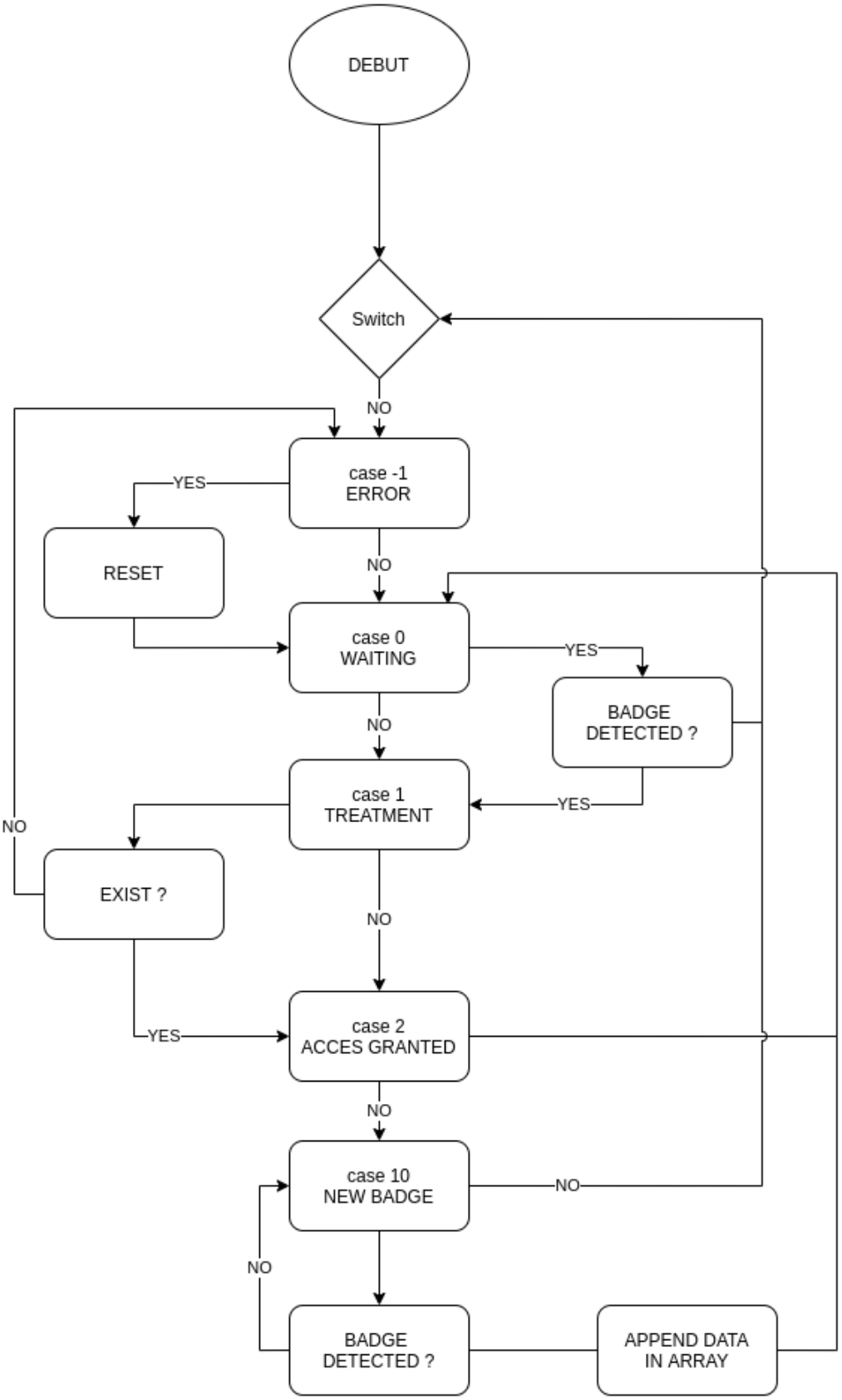
Durant la dernière séance, nous avons finalisé le rapport mais nous avons rencontré beaucoup de difficultés au niveau de la programmation.
En effet, le remplissage du tableau en 2 dimensions avec les valeurs des différents badges nous a donné du fil à retordre, alors qu'une partie du programme a été spécialement conçue pour.
Une grande partie de la séance a donc été remplie par le débuggage.

Le programme a été débuggé et a été testé en fin de séance, il est entièrement fonctionnel.

3) Partage du programme

A) Algorithme

Figure 2: Algorithme du programme (pas totalement représentatif)



Le programme repose sur un *switch case*, cela le rend plus élégant et plus rapide qu'une cascade de *if*. Le traitement de cas permet aussi une meilleure lisibilité du code ainsi que de s'occuper seulement de certains cas particuliers sans tester si les autres cas doivent être fait.

Le stockage de badges se fait dans un tableau de 100 éléments, qui contiennent chacun un octet du code du badge RFID. Le badge RFID possède 4 octets pour le code d'identification, soit un total de 25 badges à attribuer.

Il est possible d'ajouter de nouveau badges dans la liste avec un bouton poussoir branché sur la sortie 4 et connecté à la terre, lors de l'appuis du bouton, le programme se bloque sur le cas 10 et se met en attente d'une lecture d'un badge pour enregistrer celui-ci. **Les badges enregistrés ne sont pas persistants, si l'arduino n'est plus alimenté les données seront perdues à jamais**, une batterie est à prévoir en cas de coupure.

B) Code

Le programme final d'exploitation du capteur RFID est le suivant :

```
#include <SPI.h>     // SPI
#include <MFRC522.h> // RFID

#define NewEntryBT 4
#define SS_PIN 10
#define RST_PIN 9

#define IDArraySize 100
#define NumBytesCard 4

// Déclaration
MFRC522 rfid(SS_PIN, RST_PIN);

byte IDArray[IDArraySize];

int lastCardIndex = 0;

// Etapes du programme
char step = 0;

/*
void displayArray()
{
  for (byte j = 0; j <= IDArraySize; j += 4)
  {

    for (byte i = 0; i < NumBytesCard; i++)
    {
      Serial.print(IDArray[j + i]);
      Serial.print(" ");
    }
    Serial.println("");
  }
}
*/

void setup()
{
  // Init RS232
```

```
40  Serial.begin(9600);
41  Serial.println("IS ON !");
42
43  // Init SPI bus
44  SPI.begin();
45
46  // Init MFRC522
47  rfid.PCD_Init();
48
49  pinMode(NewEntryBT, INPUT_PULLUP);
50 }
51
52 void loop()
53 {
54
55  switch (step)
56  {
57
58  // Badge inconnu
59  case -1:
60    Serial.println("BADGE UNKNOW");
61    Serial.println("CASE -1");
62    step = 0;
63    delay(1000);
64    break;
65
66  //Attente de badge
67  case 0:
68
69    // Si bouton Nouvelle Entrée est appuyé
70    if (!digitalRead(NewEntryBT))
71    {
72      Serial.println("NEW BADGE");
73      step = 10;
74    }
75
76    // Initialisé la boucle si aucun badge n'est présent
77    if (!rfid.PICC_IsNewCardPresent())
78      return;
79
80    // Vérifier la présence d'un nouveau badge
81    if (!rfid.PICC_ReadCardSerial())
82      return;
83
84    Serial.println("CASE 0");
85    step = 1;
86    break;
87
88  // Test l'ID du badge (4 octets)
89  case 1:
90
91    for (byte j = 0; j < IDArraySize; j += 4)
92    {
93
94      for (byte i = 0; i < NumBytesCard; i++)
95      {
96
97        if (IDArray[j + i] == rfid.uid.uidByte[i])
98        {
```

```
99          step = 2;
100       }
101     }
102   }
103
104   if (step != 2)
105     step = -1;
106
107   Serial.println("READING");
108   Serial.println("CASE 1");
109   break;
110
111  // Ouverture porte
112  case 2:
113   step = 0;
114   Serial.println("ACCESS GRANTED");
115   Serial.println("CASE 2");
116
117   delay(1000);
118   break;
119
120  // Enregistrement Badge
121  case 10:
122
123   // Initialisé la boucle si aucun badge n'est présent
124   if (!rfid.PICC_IsNewCardPresent())
125     return;
126
127   // Vérifier la présence d'un nouveau badge
128   if (!rfid.PICC_ReadCardSerial())
129     return;
130
131   // Enregistre le nouveau Badge
132   Serial.print("lastCardIndex = ");
133   Serial.println(lastCardIndex);
134
135   for (byte i = 0; i < NumBytesCard; i++)
136   {
137     IDArray[i + lastCardIndex] = rfid.uid.uidByte[i];
138   }
139   //displayArray();
140   lastCardIndex += 4;
141
142   step = 0;
143   Serial.println("BADGE SAVED");
144   Serial.println("CASE 10");
145   delay(1000);
146   break;
147  }
148 }
```

---

syntax highlighted by Code2HTML, v. 0.9.1