Équipe FF11

Julia Asselin

Callista Bélanger

Shakira Elmi

Gelbert Simo

Dieu Merci Tshiaba

Livrable F : Prototype 1 et rétroaction

Travail présenté au

professeur Bouendeu

dans le cadre du cours

GNG1503

Université d'Ottawa

Le 2 mars 2025

Table des matières

Introduction

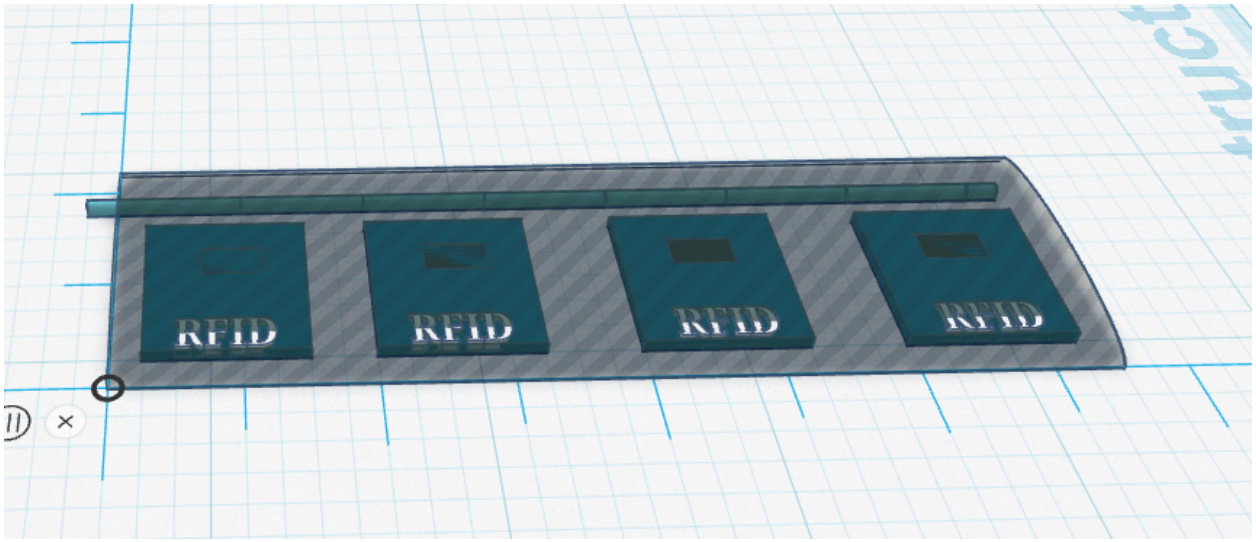
Les puces RFID sont utilisées dans plein d'aspects du quotidien : on peut ainsi profiter de cette technologie pour concevoir un système de détection qui répondra aux besoins de l'ECJM de l'Université d'Ottawa. Ils planifient une course de voitures télécommandées suivie par un système fiable et simple qui enregistre le vainqueur de la course, le nombre de tours des voitures et leurs durées. On pose l'hypothèse que cet objectif sera atteint par le système de puce RFID. La conception de ce système se poursuit avec l'ébauche du premier prototype. La présente documentation décrit le premier prototype, en fait son analyse, énonce la rétroaction obtenue par des utilisateurs, décrit le plan du prochain prototype et fait la mise à jour de la nomenclature des matériaux.

Le prototype 1

Le prototype 1 consiste du code qui servira à interpréter les signaux captés par les lecteurs RFID ainsi qu'un modèle analytique du circuit qui connectera les capteurs au microcontrôleur. La première partie du prototype, le code, a pour but de montrer que les résultats peuvent être compilés et interprétés. Le code est en fait le sous-système qui permet au système d'être utilisé. En effet, s'il n'y avait qu'un circuit de capteurs, il serait impossible de déclarer un vainqueur grâce au système. Le code est disponible à Annexe B.

La deuxième partie du prototype 1, le modèle, est effectuée sur Tinkercad. Le modèle a pour but de montrer le concept mis à jour à la suite de la revue de conception. Initialement, un lecteur de puces RFID devait être placé en face, au-dessus ou à côté de la ligne d'arrivée. Cependant, comme les capteurs RFID ont une courte portée de détection, il serait plus judicieux de placer plusieurs lecteurs sous la ligne d'arrivée. Ceci nécessite l'ajout d'un sous-système : une rampe permettant aux voitures de passer au-dessus des capteurs sans les endommager.

Figure 2- Modèle de l'organisation des capteurs à la ligne d'arrivée



Une représentation 3D du prototype a été réalisée à l'aide du logiciel Web Tinkercad. Le fichier du modèle est disponible à https://www.tinkercad.com/things/0McltCAdLEO-swanky-jarv/edit?returnTo=%2Fthings%2F0McltCAdLEO-swanky-jarv

L'analyse du prototype 1

Les hypothèses

D'abord, des projets utilisant Arduino ont été recherchés pour déceler une structure de base pour le code. Un projet de casse-tête RFID utilisant quatre capteurs tout comme le système Chrono-Tours à détection RFID a servi de point de départ pour le code.

Figure 3a- Code servant de référence au prototype 1 [1]

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN         10
#define SS_1_PIN        4
#define SS_2_PIN        5
#define SS_3_PIN        6
#define SS_4_PIN        7
#define NR_OF_READERS   4
byte ssPins[] = {SS_1_PIN, SS_2_PIN, SS_3_PIN, SS_4_PIN};
MFRC522 mfrc522[NR_OF_READERS];                   // Create MFRC522 instance.
uint8_t reader;

void setup() {
  Serial.begin(9600);
  while (!Serial);    // Do nothing if no serial port is opened
  SPI.begin();        // Init SPI bus
  checkReaders();     // Init the readers
}


void loop() {
  for (uint8_t reader = 0; reader < NR_OF_READERS; reader++) {
    mfrc522[reader].PCD_Init(ssPins[reader], RST_PIN);     // Init each MFRC522 reader
    delay(1);                                              // necessary delay to detect mifare ultralight
    // Look for new cards
    if (mfrc522[reader].PICC_IsNewCardPresent() && mfrc522[reader].PICC_ReadCardSerial()) {
      Serial.print(F("Reader "));
      Serial.print(reader);
      // Show some details of the PICC (that is: the tag/card)
      Serial.print(F(": Card UID:"));
      dump_byte_array(mfrc522[reader].uid.uidByte, mfrc522[reader].uid.size);
      Serial.print("\tPICC type: ");
      MFRC522::PICC_Type piccType = mfrc522[reader].PICC_GetType(mfrc522[reader].uid.sak);
      Serial.println(mfrc522[reader].PICC_GetTypeName(piccType));
    } //if (mfrc522[reader].PICC_IsNewC
    else {
      // no new card found and read
      Serial.println("no valid tag present!");
    }
    // Halt PICC
    mfrc522[reader].PICC_HaltA();
    // Stop encryption on PCD
    mfrc522[reader].PCD_StopCrypto1();
  } //for(uint8_t reader
}
```

Figure 3b- Code servant de référence au prototype 1 [1]

```
void dump_byte_array(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.print(buffer[i], HEX);
  }
}

void checkReaders() {
  boolean allworking = true;
  Serial.println();
  Serial.println("Checking all Readers!");
  for (reader = 0; reader < NR_OF_READERS; reader++) {
    delay(1);
    mfrc522[reader].PCD_Init(ssPins[reader], RST_PIN);        // Init each MFRC522 reader
    byte verbyte = mfrc522[reader].PCD_ReadRegister(mfrc522[reader].VersionReg);
    Serial.print(F("Reader "));
    Serial.print(reader);
    Serial.print(F(": "));
    mfrc522[reader].PCD_DumpVersionToSerial();
    if (verbyte != 0x92) {
      allworking = false;
    }
    mfrc522[reader].PICC_HaltA();
    mfrc522[reader].PCD_StopCrypto1();
  }
  if (allworking == false) {
    Serial.println("There is a Problem!");
  }
  else {
    Serial.println("Readers are working");
  }
  Serial.println();
}
```

Pour que ce code soit bien implémenté par Arduino IDE, il a fallu inclure une bibliothèque MFRC522 [2] compatible avec les puces RFID du type MFRC522. Par ailleurs, ce code n'est pas entièrement adapté à ce que le système Chrono-Tours doit accomplir : il ne fait que identifier l'identité des puces. Pour accomplir les buts premiers du projet (compter le nombre de tours et leur durée), il faut ajouter une structure [3] emmagasinant l'information de chaque puce, soit les données de chaque voiture. Cette structure assigne à chaque puce son identifiant, une variable comptant le nombre de tours et une variable chronométrant la durée des tours. Ensuite, pour mettre une condition d'arrêt, il faut écrire une fonction déterminant si les quatres voitures ont fait leurs 10 tours. Ceci nécessite l'incrémentation des variables

comptant les tours à chaque détection. Il faut donc écrire des fonctions pouvant identifier la puce, déterminer si la puce a déjà une structure associée et ajouter la puce si ce n'est pas le cas.

Ensuite, pour afficher les résultats convenablement, il faut initialiser un dictionnaire dans lequel seront entreposées les données de chaque tour de chaque voiture. Quand la condition d'arrêt est satisfaite, ce dictionnaire est affiché.

La fonction finale du code de référence vérifiant la communication de chaque capteur est gardée presque intacte. Les lignes initialisant la variable du lecteur et imposant la condition de détection d'une puce grâce à cette variable sont également gardées intactes.

Le code est écrit en C/C++ sur Arduino IDE. Ce langage n'a pas de classe dictionnaire, il faut donc trouver et inclure une bibliothèque supportant ce type de données [4].

Les résultats

La méthode d'essais du prototype 1 est simplement la compilation du code sur Arduino IDE (voire Annexe A). La durée estimée de ce test était de deux heures et le critère d'arrêt, que Arduino ne relève pas d'erreur. Le code a été écrit itérativement, en compilant chaque ébauche et en effectuant des ajustements selon les erreurs relevées. La durée cumulative des essais a finalement été de trois heures.

Le résultat principal de cette méthode d'essai est le suivant : le code ne contient aucune erreur de compilation. En outre, voici un résumé de ce que le code devrait être capable d'accomplir durant les prochains tests (l'implémentation du code avec le circuit et le microcontrôleur Arduino) étant donné qu'aucune erreur n'est décelée :

- Établir une variable du maximum de quatre puces pouvant être détectées ;
- Établir une variable du maximum de 10 tours que les voitures feront ;
- Créer une structure emmagasinant l'identité, le nombre de tours et la durée des tours de chaque voiture ;
- Créer un dictionnaire enregistrant tous les résultats ;
- Arrêter le programme et imprimant les résultats lorsque la course est terminée ;
- Initialiser une boucle principale passant à travers les données de chaque capteur, puis identifiant si la puce détectée et incrémentant son nombre de tour en enregistrant la durée du dernier tour dans le dictionnaire.

La rétroaction du prototype 1

En discutant avec quelques professionnels, nous avons pu recueillir de la rétroaction pour notre prototype. Premièrement, nous avons été avisés d'ajouter des traducteurs de tension entre l' Arduino et les détecteurs RFID. Ceci est a cause que même s'il y a une sortie de 3.3V sur le arduino, les I/O sur le arduino fonctionne quand même a 5V donc il se peut que les données des détecteurs ne soient pas bien lus par le microprocesseur. De plus, de cette façon on peut utiliser la sortie 5V et si la V Sortie diminue un peu a cause que le courant devient trop élevé en fournissant plusieurs détecteurs, ils peuvent quand même fonctionner.

Deuxièmement, un client nous a conseillé de vérifier le code en partie en utilisant "unit testing". De cette façon, nous pouvons mieux anticiper les erreurs qui pourraient arriver dans le code. Ceci nous permettrait de le rendre plus efficace et évidemment plus fiable.

Le plan du prototype 2

Le deuxième prototype se focalise sur la construction du circuit. Le prototype 2 consiste du circuit de capteurs et du microcontrôleur Arduino. Ce sous-système sera testé grâce aux essais décrits ci-dessus et à l'aide du code écrit dans le prototype 1. Grâce au code, nous serons capable de déterminer si la métrique du paramètre testé est adéquate ou si des ajustements sont nécessaires. Alors, l'objectif principal de ce prototype est de fixer les métriques de distance entre les capteurs, entre les capteurs et la rampe et de l'épaisseur de la rampe pour assurer la fiabilité maximale du système.

Tableau 1- Plan d'essais du prototype 2

Concept de conception		Système de détection à puces RFID						
Test	**Problème critique probable**	**Objectif du test**	**Description du test**	**Méthode d'analyse**	**Éléments mesurables**	**Métri-ques**	**Niveau et fidélité du prototype**	**Type de prototype**
1	Distance de la piste couverte avec les capteurs	Mesurer la distance/proportion de la piste que les capteurs couvrent afin de déterminer le placement du pont et	Une fois que nous avons les capteurs, nous allons déterminer quelle proportion de la piste ils peuvent	Durée estimée d'environ 2 heures. Détermine la proportion de la poste couverte par les capteurs en tenant compte de leur efficacité. Vérifier, dans un scénario ou	Proportion de la piste couverte par les capteurs.	Proportion de la piste : S.O	Fidélité moyenne, ciblé.	Physique

		assurer qu'il y a assez d'espace pour les voitures pour qu' ils soient tous capables d'être lus sans aucune collision.	couvrir tout en prenant compte qu'ils doivent être efficaces. Il est aussi primordial de pouvoir garder une distance au qu'elle dans un scénario si 4 voiture passe, il n'y a pas de collision. Le test serait beaucoup de tests et erreurs afin de déterminer la meilleur distance des capteurs sur la largeur de la piste.	quatre voiture passent la ligne d'arrivée, il n'y a pas de collision et le capteur est capable de détecter chaque puce sur chacune des voitures. Réaliser plusieurs essais et erreurs pour optimiser l'emplacement des capteurs et une couverture efficace sans interférence.				
2	La distance maximum que les capteurs peuvent détecter.	Puisque nous planifions d' installer un type de pont à la ligne d'arrivée où on installera les capteurs en dessous, il est crucial de déterminer à quelle hauteur on placera le pont. Et si le capteur serait toujours capable de prendre des données à une telle hauteur.	Déterminer le matériel qui fera le pont (surement carton pour sauver de l'argent), puis avant que on construit la forme du pont on vas faire un test ou les capteurs sont par terre puis on vas augmenter la distance entre le capteur et le carton graduellement en prenant en compte si le capteur est toujours capable de lire les données puis lorsqu'on détermine la hauteur max.	Comparer la lecture des données puis déterminer laquelle est la meilleure et sa hauteur correspondante est celle que nous allons choisir.	La distance maximale de rétention du capteur et sa puce a traversé le pont.	Hauteur : mètre	Moyenne, fidélité, cible.	Physique
3	L'épaisseur et le matériel qui compose le pont.	Choisir le meilleur matériel pour le pont qui ne cause pas de problème à notre système global.	Nous avons pu réduire nos choix à 3 matériaux, soit le plastique de la l'imprimante 3D, le carton, ou le bois.	Durée estimée de 1 jour. Les matériaux sélectionnés sont le plastique imprimé en 3D, le carton et le bois. Chaque matériau sera testé pour évaluer sa compatibilité avec les capteurs et son impact sur la lecture des	Épaisseur du matériel choisi. Capacité de support.	Épaisseur du matériel : mètre, pouce Poids : gramme.	Moyenne, fidélité, cible	Physique.

			De façon générale, aucun de ces materiel causent des probleme au capteur au niveau de la lecture des puces. Par contre, évidemment il y a certains matériaux qui sont plus compatibles que d'autres, nous allons avoir un petit morceau de chaque matériaux et évaluer la lecture du capteur de chacun et choisir le meilleur ainsi que celui qui est dans notre budget. Au niveau de l'épaisseur, elle sera déterminée par essai et erreur, nous allons augmenter du matériel choisi tout en évaluant la lecture des puces ainsi que son support de quatre voitures dans le circuit.	puces. L'épaisseur optimale sera déterminée par essais et erreurs en vérifiant la lisibilité des puces et la résistance aux voitures. Le choix final se fera en fonction de la performance, du coût et de la solidité.				

Le plan du prototype 2 évalue tous les tests possibles qui concernent le circuit. Le premier test permet de déterminer la distance que les capteurs couvrent au niveau de la piste, le deuxième nous aide à déterminer la distance (hauteur) maximale entre le capteur et le pont que nous allons installer à la ligne d'arrivée, le dernier est d'identifier le matériel du pont et son épaisseur.

Mise à jour de la nomenclature des matériaux

Tableau 2- Nomenclature des matériaux

Numéro	Description du composant	Quantité	Prix unitaire	Prix Calculé
1	Microcontroleur	1	9,00$	9,00$
2	Fils électriques mâle-mâle	20	0,10$	2,00$
3	Fils électriques mâle-femelle	20	0,10$	2,00$
4	Breadboard	1	2,50$	2,50$
5	Senseur RFID	1	17,99$	17,99$
6	Velcro	1	4,99$	4,99$
7	Ruban adhésif	1	0$	0,00$
8	Carton	2	0$	0,00$
9	Bois	2	0$	0,00$
			Total + taxe et livraison :	40,82$

Les coûts et les quantités des matériaux sont susceptibles de changer tout au long des essais. Il sera possible de garder le fil de ces informations et de les modifier à Nomenclature des matériaux FF11.xlsx

Conclusion

Tout comptes fait, le prototype 1 montre une modélisation parfaite de ce dont nous voulons que le code fasse une fois compilé. En effet, dans cette section nous avons une représentation 3D de notre configuration des senseurs placés à la ligne d'arrivée qui sert aussi de ligne de départ. De plus, lors de la rétroaction nous avons été conseillés d'utiliser "unit testing" pour un meilleur rendement et efficacité du code. Encore, l'inclusion de traducteur de tension entre le microcontrôleur et le senseur a été conseillée pour un bon fonctionnement des cartes électriques. Par ailleurs, le plan du prototype 2 explique comment notre second prototype sera focalisé sur la construction du circuit. Il explique comment nous allons analyser et mesurer les différents tests que nous allons prendre. Enfin une mise à jour de la nomenclature des matériaux qui possède leurs prix et leurs quantités à été faites.

Références

1. DerGeppi. “[solved] problem with reading from multiple MFRC522.” Arduino Forum, mars 2021. https://forum.arduino.cc/t/solved-problem-with-reading-from-multiple-mfrc522/698438?_gl=1*r6jf3n*_up*MQ..*_ga*ODA5NDc5OTc4LjE3NDA2ODA5ODY.*_ga_NEXN8H46L5*MTc0MDY4MDk4NS4xLjAuMTc0MDY4MDk4NS4wLjAuNTEzNDE1NTMy
2. Miguelbalboa. “MFRC522.” GitHub, février 2025. https://github.com/miguelbalboa/rfid
3. Main, John. "Arduino Struct: Easily Organize Your Arduino Code with Structs." *Best Microcontroller Projects*, https://www.best-microcontroller-projects.com/arduino-struct.html.
4. Arkhipenko. “Dictionary.” GitHub, 2022. https://github.com/arkhipenko/Dictionary

ANNEXE A

Tableau 3-Méthode d'essais du prototype 1

Test	Prototype	Objectif	Méthode	Éléments mesurables	Durée	Critère d'arrêt	Résultats
1	Code Arduino pour quatre capteurs MFRC522.	Compter le nombre de détections et les intervalles de détection.	Ébauche itérative du code et compilation.	Erreurs décelées par Arduino IDE.	Deux heures.	Aucune erreur n'est relevée.	Durée finale : trois heures. Aucune erreur n'est décelée. Le code implémenté devrait compter les tours, enregistrer leurs durées et afficher toutes les données à la fin de la course.

ANNEXE B

Figure 1- Script du code du système Chrono-Tours à détection RFID sur Arduino IDE

```
#include <Dictionary.h>
#include <DictionaryDeclarations.h> // Pour enregistrer les données sous forme de dictionnaire

#include <SPI.h> // Communication capterus-Arduino
#include <MFRC522.h> // Pour utiliser les données des capteurs MFRC522

#define RST_PIN         10
#define SS_1_PIN        4 // Définir l'entrée de connection de chacun des capteurs
#define SS_2_PIN        5
#define SS_3_PIN        6
#define SS_4_PIN        7
#define NR_OF_READERS   4 // Nombre de capteurs
#define PUCE_MAX        4  // Maximum de puces détectées
#define TOUR_MAX       10 // Condition d'arrêt

byte ssPins[] = {SS_1_PIN, SS_2_PIN, SS_3_PIN, SS_4_PIN};
MFRC522 mfrc522[NR_OF_READERS];

struct PuceInfo { //Entreposer les variables de nombre de tour et de durée dans un objet PuceInfo
    String puceID; //Variable ID
    int compte; //Variable # de tours
    unsigned long temps; // variable temps depuis le départ de Arduino au moment de la détection
};

PuceInfo liste[PUCE_MAX]; //créer une liste des puces avec 4 éléments objets de PuceInfo struct

Dictionary resultats (4);  // Déclaration du dictionnaire

void setup() {
    Serial.begin(9600);
    while (!Serial); //Aucune action tant que le port n'est pas ouvert
    SPI.begin(); //Initialiser la communication Arduino-capteurs
    checkReaders(); //Initialiser les capteurs
}

void loop() {
    if (toutesLesVoituresOntFini()) { //Condition pour arrêter le programme
        Serial.println("Les quatre coureurs ont fini la course!");
        Serial.println("Les résultats finaux :");
        for (int i = 0; i < resultats.size(); i++) { //Afficher les résultats en passant à travers le dictionnaire avec une boucle
            Serial.print(resultats[i]);  // Imprimer la clé : le ID de la puce
            Serial.print(" : ");
            Serial.println(resultats(resultats[i]));  // Imprimer les valeurs : les numéros des tours et leurs durées
        }
        while (true); // Arrêt du programme
    }

    for (uint8_t reader = 0; reader < NR_OF_READERS; reader++) {  //Dans le code de référence
        mfrc522[reader].PCD_Init(ssPins[reader], RST_PIN);
        delay(1); //délais nécessaire de détection

        if (mfrc522[reader].PICC_IsNewCardPresent() && mfrc522[reader].PICC_ReadCardSerial()) { //Condition de détection
            String puceID = identifier(mfrc522[reader].uid.uidByte, mfrc522[reader].uid.size); // Trouver le ID de la puce, le convertir en UID et trouver sa longueur
            unsigned long tempsactuel = millis(); //Enregistrer le temps écoulé dans la variable tempsactuel
            int index = trouver(puceID); //vérifier si la puce est déjà dans la liste

            if (index == -1) { //Ajouter la puce comme ayant fait son premier tour si elle ne se trouve pas dans la liste
                index = ajouterPuce(puceID);
            }
```

```
            if (index != -1) { //si on a ajouté ou trouvé une puce, l'index est à l'index de la puce dans la liste et on incrémente le compte.
                liste[index].compte++;
                unsigned long dureetour = tempsactuel - liste[index].temps; //trouver la durée du tour à partir du temps actuel et du temps à la dernière détection
                liste[index].temps = tempsactuel; //ajout de la donnée du temps dans l'objet de la puce en question

                resultats(liste[index].puceID, String(liste[index].compte) + ":" + String(liste[index].temps)); //ajout de la paire puce-#tour/temps dans le dictionnaire

                Serial.print(F("Capteur : ")); //Afficher les données instantanément lors de la détection
                Serial.print(reader);
                Serial.print(F(" UID de la puce : "));
                Serial.print(puceID);
                Serial.print(F(" | Nombre de tours : "));
                Serial.print(liste[index].compte);
                Serial.print(F(" | Durée du dernier tour : "));
                Serial.print(dureetour);
                Serial.println(F(" ms"));
            }
        }
        mfrc522[reader].PICC_HaltA(); //arrêter la réponse temporairement
        mfrc522[reader].PCD_StopCrypto1(); //remettre le capteur à 0 pour le préparer à la prochaine lecture
    }
}

bool toutesLesVoituresOntFini() {  //Fonction pour déterminer si chaque voiture a terminé
    for (int i = 0; i < PUCE_MAX; i++) { //pour tester chaque puce de la liste
        if (liste[i].compte < TOUR_MAX) { //condition si la voiture est à 10 tours
            return false;
        }
    }
    return true;

            }

            String identifier(byte *buffer, byte bufferSize) { //fonction convertit le ID en String : prend une liste de bytes et sa longueur
                String puceID = ""; //initialiser la variable puceID
                for (byte i = 0; i < bufferSize; i++) { //convertir le byte array en String
                    puceID += (buffer[i] < 0x10 ? "0" : ""); //ajout de 0 au besoin
                    puceID += String(buffer[i], HEX); //conversion de chaque byte
                }
                return puceID;
            }

            int trouver(String puceID) { //Vérifier si la puce a déjà été détectée
                for (int i = 0; i < PUCE_MAX; i++) { //Traverser la liste
                    if (liste[i].puceID == puceID) {
                        return i;   //arrêter la fonction si la puce se trouve dans la liste
                    }
                }
                return -1;
            }

            int ajouterPuce(String puceID) {
                for (int i = 0; i < PUCE_MAX; i++) {
                    if (liste[i].puceID == "") { //condition si la position dans la liste ne contient pas déjà une puce
                        liste[i].puceID = puceID; //ajout de la puce le cas échéant
                        liste[i].compte = 0; //initialisation du nombre de tours à 0
                        liste[i].temps = millis(); //initialiser le temps avec le temps actuel
                        return i;
                    }
                }
                return -1;
            }

    void checkReaders() { //Fonction dans la référence, vérifie que tous les capteurs fonctionnent et initialise les capteurs
        boolean allWorking = true;
        Serial.println("\nChecking all Readers!");
        for (uint8_t reader = 0; reader < NR_OF_READERS; reader++) {
            delay(1);
            mfrc522[reader].PCD_Init(ssPins[reader], RST_PIN);
            byte verByte = mfrc522[reader].PCD_ReadRegister(mfrc522[reader].VersionReg);
            Serial.print(F("Reader "));
            Serial.print(reader);
            Serial.print(F(": "));
            mfrc522[reader].PCD_DumpVersionToSerial();
            if (verByte != 0x92) {
                allWorking = false;
            }
            mfrc522[reader].PICC_HaltA();
            mfrc522[reader].PCD_StopCrypto1();
        }
        Serial.println(allWorking ? "Readers are working" : "There is a Problem!");
    }
```