Équipe FF11

Julia Asselin

Callista Bélanger

Shakira Elmi

Gelbert Simo

Dieu Merci Tshiaba

Livrable H : Prototype 3 et rétroaction des clients

Travail présenté au

professeur Bouendeu

dans le cadre du cours

GNG1503

Université d'Ottawa

Le 23 mars 2025

Table des matières

Introduction **2**
Description et analyse du prototype **2**
Stade 1 2
Stade 2 3
Stade 3 4
Documentation des essais **5**
Rétroaction de clients **7**
Mise à jour de spécifications de conception technique et de la nomenclature des matériaux **8**
Conclusion **9**
ANNEXE A **10**
ANNEXE B **11**

Introduction

Pour suivre une course de quatre voitures télécommandées, il faut un système de détection qui déclare un vainqueur tout en comptabilisant les tours et leurs durées. Pour bâtir un tel système, l'équipe FF11 a généré une solution utilisant la technologie RFID. Le système consiste donc de capteurs RFID, de puces RFID, d'un microcontrôleur, de fils pour relier toutes les composantes, d'un code pour enregistrer les données des capteurs et d'une rampe pour faciliter le passage des voitures. Le prototype final rassemble les trois sous-systèmes : le circuit Arduino, le code et la rampe. Le document présent a pour but de présenter le prototype final, les résultats des essais sur ce prototype et les leçons apprises. Suivant ces informations seront énoncés les rétroactions de clients par rapport au système et une mise à jour finale des spécifications de conception technique et de la nomenclature des matériaux.

Description et analyse du prototype

Les prototypes précédents comprenaient l'écriture du code et l'assemblage du circuit à quatre capteurs. L'ébauche du code a été une réussite : un code fonctionnel satisfaisant aux besoins principaux du produit en a résulté. L'assemblage du circuit n'a pas été fructueux. Les capteurs ne communiquaient pas avec le microcontrôleur. Comme solution, il a été convenu que le microcontrôleur devait être remplacé et que des convertisseurs de voltage seraient ajoutés.

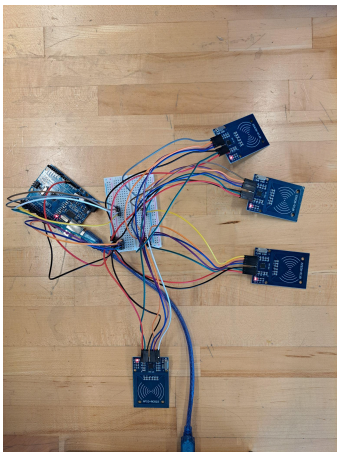
La conception du prototype 3 a donc pris trois formes au fil des essais et des ajustements requis. Le premier stade de ce prototype correspond à un circuit à quatre capteurs RC522 : ceci est plutôt la continuité du prototype 2 sur laquelle les tests ont permis de déterminer qu'un circuit à un capteur est plus réalisable. Ce deuxième stade du prototype 3 est donc un circuit à un capteur RC522 à un seul capteur fonctionnant avec une nouvelle version du code. Le stade final de la conception correspond à ce circuit soudé et fixé à la rampe en carton sur laquelle les voitures passeront une à une.

Stade 1

Ce circuit contient comme éléments quatre capteurs RC522, un microcontrôleur Arduino Uno, un fil USB, un breadboard, 28 fils mâle-femelle et sept fils mâle-mâle. L'objectif de ce circuit est de tester les connexions pour s'assurer que les capteurs reçoivent suffisamment de courant pour fonctionner, et de tester à quelles distances horizontale et verticale les capteurs peuvent détecter des puces. Théoriquement, ce circuit permet de déterminer où précisément chaque composante doit être placée sur la ligne d'arrivée. En pratique, il faut d'abord régler les problèmes rencontrés lors de prototypage 2 (à savoir l'absence totale de réponse des capteurs) pour accomplir ces objectifs. Une solution possible à ces problèmes était de

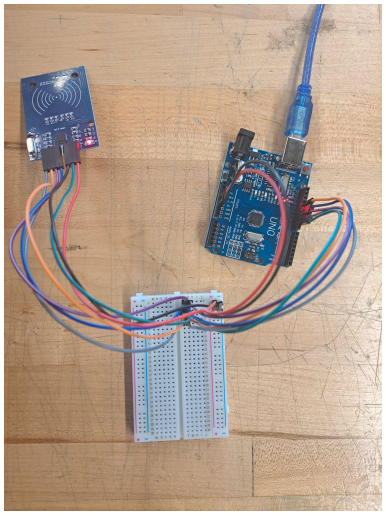
remplacer les pièces défectueuses, le microcontrôleur dans ce cas. Toutefois, le microcontrôleur remplacé, les mêmes problèmes persistent. En effet, la recherche effectuée aux fins de résoudre ce problème montre que bien qu'en théorie un circuit où l'on branche quatre composantes RC522 à un seul microcontrôleur fonctionne, ces composantes doivent partager un SPI, ce qui fonctionne rarement sans défaut en pratique. Les essais se sont donc poursuivis avec un circuit à un seul capteur.

Figure 1- Circuit à quatre capteurs RFID



Stade 2

Figure 2- Circuit à un capteur RFID



Le nouveau circuit est composé d'un microcontrôleur Arduino Uno, d'un capteur RC522, de sept fils mâle-femelle, de sept fils mâle-mâle, d'un fil USB et d'un breadboard. Ce prototype doit vérifier que le

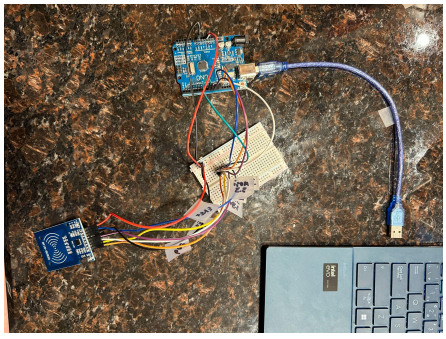
capteur peut détecter les puces RFID. Une fois ceci accompli, il sert à déterminer les distances et la vitesse de détection. Avec ce circuit, il n'est plus nécessaire de tester la détection multiple spontanée, car le concept à un capteur implique la détection d'une voiture à la fois. Le code modifié pour ce nouveau concept est disponible à Annexe B. Alors, le concept initial est modifié pour pallier cette défaillance : les concurrents seront forcés à passer un à la fois sur une rampe seulement assez large pour accommoder une voiture. Le circuit à un capteur performe mieux que la première ébauche, mais ne fonctionne pas parfaitement.

En outre, comme les capteurs RC522 fonctionnent à 3,3 V, il faut utiliser la sortie de 3,3 V du Arduino Uno. Toutefois, les autres *pins* du microcontrôleur fonctionnent à 5 V et cette différence de voltage peut causer des défauts du traitement de l'information. Le circuit est donc testé en ajoutant deux convertisseurs de tension à quatre canaux. Les convertisseurs de tension permettent d'utiliser la sortie à 5 V du microcontrôleur sans endommager le capteur tout en assurant le bon fonctionnement des autres sorties. De plus, si le courant diminue à cause de la résistance, il y aura toujours suffisamment de courant alimentant les capteurs, car la tension est diminuée par les convertisseurs peu importe.

Stade 3

Figure 3A- Rampe du prototype final

Figure 3B-Circuit soudé du prototype final



Le prototype final est l'assemblage du circuit et de la rampe. Il permet de vérifier que l'ensemble du système accomplit les fonctions importantes, soit de détecter les voitures et de compter les tours et leur durée. Ainsi, il faut tester que la rampe peut supporter le poids d'une voiture et permet encore de détecter les voitures. Cette rampe est faite en carton puisque ce matériau est très abordable et facilement accessible. La rampe est conçue selon les dimensions des voitures Tamiya TT02 (36 x 18 cm) : le plateau mesure 50 cm par 30 cm et la montée et la descente mesurent 30 cm par 20 cm pour minimiser

l'inclinaison. En effet, la rampe ayant une hauteur de 5 cm, pour calculer la longueur de la montée avec une inclinaison raisonnable de 15°, on utilise la trigonométrie des triangles rectangles :

$$l = 5\,cm/sin15 = 19,32\,cm$$

Pour que celle-ci soit le plus robuste possible, elle est renforcée avec des piliers de forme triangulaire. Enfin, afin que les voitures embarquent sur la rampe sans faille, le côté montant sera fixé au sol au moment de l'installation à l'aide de ruban adhésif.

Documentation des essais

Tableau 1- Essais effectués avec chaque partie du prototype

Circuit à quatre capteurs	• Tests de détection avec les exemples de code MFRC522 sur Arduino IDE
Circuit à un capteur	• Tests de détection avec les exemples de code MFRC522 sur Arduino IDE • Test de distance de détection verticale
Circuit à un capteur et deux convertisseurs de tension quatre canaux	• Tests de détection avec les exemples de code MFRC522 sur Arduino IDE • Test de la tension
Rampe	• Test de résistance à la force verticale

Tableau 2- Résultats des essais avec le circuit à quatre capteurs

Essai	Objectif	Résultat	Interprétation/modification apportée
1	Vérifier si les capteurs sont bien connectés en utilisant les exemples de code de la bibliothèque MFRC522 (voire Annexe A).	Aucune réponse du circuit.	• Vérification des connexions. • Couper l'alimentation et réessayer.
2	*Id.*	*Id.*	• Modification du circuit pour utiliser un seul capteur.

Tableau 3- Résultats des essais avec le circuit à un capteur

Essai	Objectif	Résultat	Interprétation/modification apportée
1	Vérifier si le capteur est bien connecté en utilisant les exemples de code de la bibliothèque MFRC522.	Communication avec le capteur et scan d'une puce. Le moniteur affiche le UID de la puce et l'information emmagasinée dans la mémoire.	● Résultat souhaité. ● La puce est détectée à courte distance, 5 cm au maximum. ● Itération pour assurer la fiabilité.
2	Vérifier la reproductibilité des scans pour assurer la fiabilité du système.	Aucune puce détectée.	● Ajout des convertisseurs de canaux pour optimiser le fonctionnement
3	Voir si les convertisseurs de canaux rendent le capteur plus performant.	Aucun courant n'est acheminé au capteur.	● Test du fonctionnement des convertisseurs de canaux.
4	Vérifier que les convertisseurs font bel et bien passer la tension de 5 V à 3,3 V en utilisant un multimètre branché en parallèle sur les fils où seraient le ground et l'entrée 3,3 V du capteur.	Tension nulle, puis tension d'environ 2,8 V après avoir vérifié les connexions.	● Les convertisseurs fonctionnent mais la tension est un peu diminuée quand elle arrive au capteur : celui-ci n'a pas assez de courant pour fonctionner. ● Les convertisseurs de tension ne seront pas utilisés.
5	Vérifier que le circuit soudé final peut	Le circuit peut bien identifier les puces,	● Le code a dû être modifié

	scanner les puces.	mais il a fallu échanger le microcontrôleur, car il ne fonctionnait pas. Les puces sont détectées à chaque fois.	pour être plus compatible avec le circuit. ● Le capteur détecte les voitures et le code affiche les résultats.
6	Tester la vitesse de détection en passant les puces rapidement devant le capteur.	Les puces sont détectées.	● Le système peut détecter les voitures lorsqu'elles passent.

Tableau 4- Résultats des essais avec la rampe

Essai	Objectif	Résultat	Interprétation/modification apportée
1	Tester la résistance de la rampe. Elle doit soutenir une voiture de 1,6 à 1,8 kg.	Déformation de la rampe en son centre.	● Ajout de piliers triangulaires au centre et aux extrémités.
2	Tester la résistance de la rampe. Elle doit soutenir une voiture de 1,6 à 1,8 kg.	La rampe supporte un poids d'environ 2 kg.	● Résistance adéquate de la rampe.

Rétroaction de clients

Un utilisateur a été consulté par rapport à la solution finale. Il propose des améliorations à la solution. Bien qu'il soit content du fonctionnement fiable, il trouve le concept d'une seule voiture passant à la fois limitant. Si le but de la course était de mesurer le tour le plus rapide, ce concept serait adéquat. En revanche, si les organisateurs de la course veulent une compétition de style F1, il serait nécessaire que toutes les voitures puissent passer en même temps. Sa proposition serait de créer quatre circuits avec leur microcontrôleur indépendant. Ainsi, la défaillance causée par le partage d'un SPI sera minimisée et quatres voitures peuvent arriver en même temps. Cette solution avait initialement été mise de côté par l'équipe de conception, mais vu l'excès de budget en cette fin de projet, il s'agit d'une amélioration possible pour de futurs travaux.

Mise à jour de spécifications de conception technique et de la nomenclature des matériaux

Tableau 5- Spécifications de conception technique

Critère	Relation	Valeur	Unités
Poids à supporter par la rampe	>=	2	kg
Vitesse détectable	>	30	km/h
Précision du chronométrage	+/-	1	s
Coût	<=	100	$
Couverture de la largeur de la piste	>=	20	cm
Hauteur de la rampe	<=	5	cm
Épaisseur de la rampe	<=	1	cm

Tableau 6- Nomenclature des matériaux

Numéro	Description du composant	Quantité	Prix unitaire	Prix Calculé
1	Microcontroleur	1	9,00$	9,00$
2	Fils électriques mâle-mâle	7	0,10$	0,70$
3	Fils électriques mâle-femelle	28	0,10$	2,80$
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	3$	3,00$
8	Carton	1	2$	2,00$
			Total :	43,98$

Conclusion

Pour conclure, la conception du prototype 3 à pris trois formes au fil des essais et des ajustements. Le premier stade a été la conception d'un circuit qui comporte quatre capteurs RC522. À la suite des résultats du prototype 2, nous avons alors conclu qu'un système à un capteur est plus fiable. Ceci nous a alors amené au deuxième stade du prototype 3 qui a été la conception d'un circuit à un capteur RC522 qui fonctionne avec une version différente du code. Enfin, le dernier stade de la conception finale est la réalisation de la rampe ainsi que la soudure du circuit. De plus, dans la documentation des essais, nous avons expliqué comment lors du deuxième stade du prototype 3, on a remarqué que l'obtention de traducteur de tension qui avait pour but de transformer 5 volt en 3,3 volts pour les capteurs RC522 s'est avérée inutile puisque après les tests seuls 2,8 volts de tension étaient mesurés. Également, lors des essais, on a vu que le capteur peut détecter les voitures lorsqu'elles passent. Finalement, on a déterminé que la rampe peut supporter jusqu'à 2 kilogrammes, ce qui est adéquat. En outre, un utilisateur nous a reproché le concept d'avoir une voiture qui passe à la fois car selon lui ce concept est limitant malgré le fait qu'il est fiable. Il propose de pouvoir détecter plusieurs voitures qui passent à la fois à l'aide de 4 circuits indépendants. Finalement, une mise à jour de la nomenclature des matériaux qui possède leurs prix et leurs quantités à été faite.

ANNEXE A

Figure 4- Exemples de code de la bibliothèque MFRC522

DumpInfo.ino

```
/*
* ---------------------------------------------------------------------------------------------------------------
* Example sketch/program showing how to read data from a PICC to serial.
* ---------------------------------------------------------------------------------------------------------------
* This is a MFRC522 library example; for further details and other examples see: https://github.com/miguelbalboa/rfid
*
* Example sketch/program showing how to read data from a PICC (that is: a RFID Tag or Card) using a MFRC522 based RFID
* Reader on the Arduino SPI interface.
*
* When the Arduino and the MFRC522 module are connected (see the pin layout below), load this sketch into Arduino IDE
* then verify/compile and upload it. To see the output: use Tools, Serial Monitor of the IDE (hit Ctrl+Shft+M). When
* you present a PICC (that is: a RFID Tag or Card) at reading distance of the MFRC522 Reader/PCD, the serial output
* will show the ID/UID, type and any data blocks it can read. Note: you may see "Timeout in communication" messages
* when removing the PICC from reading distance too early.
*
* If your reader supports it, this sketch/program will read all the PICCs presented (that is: multiple tag reading).
* So if you stack two or more PICCs on top of each other and present them to the reader, it will first output all
* details of the first and then the next PICC. Note that this may take some time as all data blocks are dumped, so
* keep the PICCs at reading distance until complete.
*
* @license Released into the public domain.
*
* Typical pin layout used:
* -----------------------------------------------------------------------------------------
*             MFRC522      Arduino       Arduino   Arduino    Arduino          Arduino
*             Reader/PCD   Uno/101       Mega      Nano v3    Leonardo/Micro   Pro Micro
* Signal      Pin          Pin           Pin       Pin        Pin              Pin
* -----------------------------------------------------------------------------------------
* RST/Reset   RST          9             5         D9         RESET/ICSP-5     RST
* SPI SS      SDA(SS)      10            53        D10        10               10
* SPI MOSI    MOSI         11 / ICSP-4   51        D11        ICSP-4           16
* SPI MISO    MISO         12 / ICSP-1   50        D12        ICSP-1           14
* SPI SCK     SCK          13 / ICSP-3   52        D13        ICSP-3           15
*
```

DumpInfo.ino

```
 * More pin layouts for other boards can be found here: https://github.com/miguelbalboa/rfid#pin-layout
 */

#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN         9          // Configurable, see typical pin layout above
#define SS_PIN          10         // Configurable, see typical pin layout above

MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance

void setup() {
  Serial.begin(9600);   // Initialize serial communications with the PC
  while (!Serial);    // Do nothing if no serial port is opened (added for Arduinos based on ATMEGA32U4)
  SPI.begin();      // Init SPI bus
  mfrc522.PCD_Init();   // Init MFRC522
  delay(4);       // Optional delay. Some board do need more time after init to be ready, see Readme
  mfrc522.PCD_DumpVersionToSerial();  // Show details of PCD - MFRC522 Card Reader details
  Serial.println(F("Scan PICC to see UID, SAK, type, and data blocks..."));
}

void loop() {
  // Reset the loop if no new card present on the sensor/reader. This saves the entire process when idle.
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }

  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }

  // Dump debug info about the card; PICC_HaltA() is automatically called
  mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```

ANNEXE B

Figure 5- Code modifié pour le circuit à un capteur

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

#define RST_PIN 9
#define SS_PIN 10
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance

// Define constants
#define TOTAL_LAPS 10
#define REQUIRED_CARDS 4

// Declare global variables
int currentLap = 1;
String scannedCards[REQUIRED_CARDS]; // To store the UIDs of scanned cars
unsigned long lapTimestamps[REQUIRED_CARDS]; // To store the lap timestamps for sorting

// Structure to store car data
struct Car {
  String uid;
  unsigned long firstScanTime;
  unsigned long secondScanTime;
  bool scannedOnce = false;
  unsigned long lapTimes[TOTAL_LAPS]; // To store lap times for each lap
  int lapCount = 0; // Track how many laps the car has completed
};

// Array to store the cars data
Car raceCars[REQUIRED_CARDS];

void setup() {
  Serial.begin(9600);
  SPI.begin(); // Initiate SPI bus
  mfrc522.PCD_Init(); // Initialize MFRC522
  Serial.println(F("RFID Reader Initialized. Scan cards to start the race."));
}

void loop() {
  if (currentLap > TOTAL_LAPS) {
    announceWinner();
    while (true);  // Stop the program
  }

  Serial.print("Lap ");
  Serial.println(currentLap);



int numCards = 0;

while (numCards < REQUIRED_CARDS) {
  if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
    String uid = getCardUID(mfrc522.uid.uidByte, mfrc522.uid.size);
    unsigned long currentTime = millis();  // Get the current time

    bool alreadyScanned = false;
    for (int i = 0; i < numCards; i++) {
      if (scannedCards[i] == uid) {
        alreadyScanned = true;
        break;
      }
    }

    if (!alreadyScanned) {
      // First scan of the car
      scannedCards[numCards] = uid;
      lapTimestamps[numCards] = currentTime;
      raceCars[numCards].uid = uid;
      raceCars[numCards].firstScanTime = currentTime;
      raceCars[numCards].scannedOnce = true;
      raceCars[numCards].lapTimes[currentLap - 1] = currentTime;  // Store the time for the current lap
      numCards++;
      Serial.print(F("Card UID: "));
      Serial.println(uid);
    } else {
      // Second scan of the car
      for (int i = 0; i < REQUIRED_CARDS; i++) {
        if (raceCars[i].uid == uid && raceCars[i].scannedOnce && raceCars[i].lapCount == currentLap - 1) {
          raceCars[i].secondScanTime = currentTime;
          unsigned long lapTime = raceCars[i].secondScanTime - raceCars[i].firstScanTime;

          // Store the lap time for the current lap (if it's the car's first lap)
          if (currentLap <= TOTAL_LAPS && raceCars[i].lapCount == currentLap - 1) {
            raceCars[i].lapTimes[currentLap - 1] = lapTime;
            raceCars[i].lapCount++;  // Increment lap count after this scan
          }

          Serial.print(F("Car UID: "));
          Serial.print(uid);
          Serial.print(F(" Second scan time: "));
```

```
            Serial.println(currentTime);
            break;
          }
        }
      }

      mfrc522.PICC_HaltA();
      mfrc522.PCD_StopCrypto1();
    }
  }

  // Now that all cars are scanned, print and sort lap times for the current lap
  sortAndPrintLapTimes(scannedCards, lapTimestamps, numCards);

  // Proceed to the next lap
  currentLap++;
  delay(1000);  // Wait before starting the next lap
}

void sortAndPrintLapTimes(String scannedCards[], unsigned long lapTimestamps[], int numCards) {
  // Sort cars based on their lap times for the current lap
  for (int i = 0; i < numCards - 1; i++) {
    for (int j = i + 1; j < numCards; j++) {
      if (lapTimestamps[i] > lapTimestamps[j]) {
        // Swap the lap times and car UIDs
        String tempCard = scannedCards[i];
        scannedCards[i] = scannedCards[j];
        scannedCards[j] = tempCard;

        unsigned long tempTime = lapTimestamps[i];
        lapTimestamps[i] = lapTimestamps[j];
        lapTimestamps[j] = tempTime;
      }
    }
  }

  // Print sorted lap times for the current lap in seconds
  Serial.println("Lap Order (sorted by time):");
  unsigned long firstCarTime = lapTimestamps[0];  // Time of the first car

  for (int i = 0; i < numCards; i++) {
    // Convert lap time from milliseconds to seconds
    float lapTimeInSeconds = lapTimestamps[i] / 1000.0;
    Serial.print(i + 1);
    Serial.print(F(") "));
```

```
    Serial.print(scannedCards[i]);
    Serial.print(F(" - Time: "));
    Serial.println(lapTimeInSeconds, 3);  // Print with 3 decimal places
  }
}

String getCardUID(byte *uidBytes, byte size) {
  String uidString = "";
  for (byte i = 0; i < size; i++) {
    uidString += String(uidBytes[i] < 0x10 ? "0" : "");
    uidString += String(uidBytes[i], HEX);
  }
  return uidString;
}

void announceWinner() {
  Serial.println(F("Race is over!"));
  // Assuming the first car in the sorted list is the winner
  Serial.print(F("The winner is: "));
  Serial.println(scannedCards[0]);
}
```