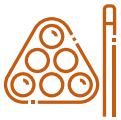
TRAVAUX D'INITIATIVE
PERSONNELLE ENCADRÉS
T.I.P.E. 2024

JEUX/SPORTS

AUTO BOULES BILLIARD

ABB



- *Réaliser par:*
  Erraki Bilal
- *Code CNC:*
  OJ047T

PLAN

1. ***Mise en situation***
2. ***Analyse fonctionnelle***
3. ***Etude de ABB-Tool***
   - Choix des composants.
   - Simulation.
   - Réalisation.
4. ***Etude de ABB:***
   - Choix des composants.
   - Asservissement.
   - Simulation.
   - Réalisation.
5. ***Conclusion***

MISE EN SITUATION

Le billard est un jeu historique, pratiqué à l'échelle mondiale, avec des organisations et des tournois internationaux. Et étant un passionné et joueur quotidien de ce jeu, j'ai choisi de m'y investir et de l'améliorer.

MISE EN SITUATION

Le principe de fonctionnement des tables de billards actuel

Coin operated System

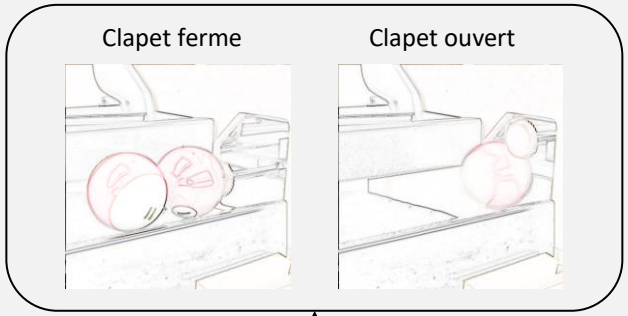
MISE EN SITUATION

Problématique

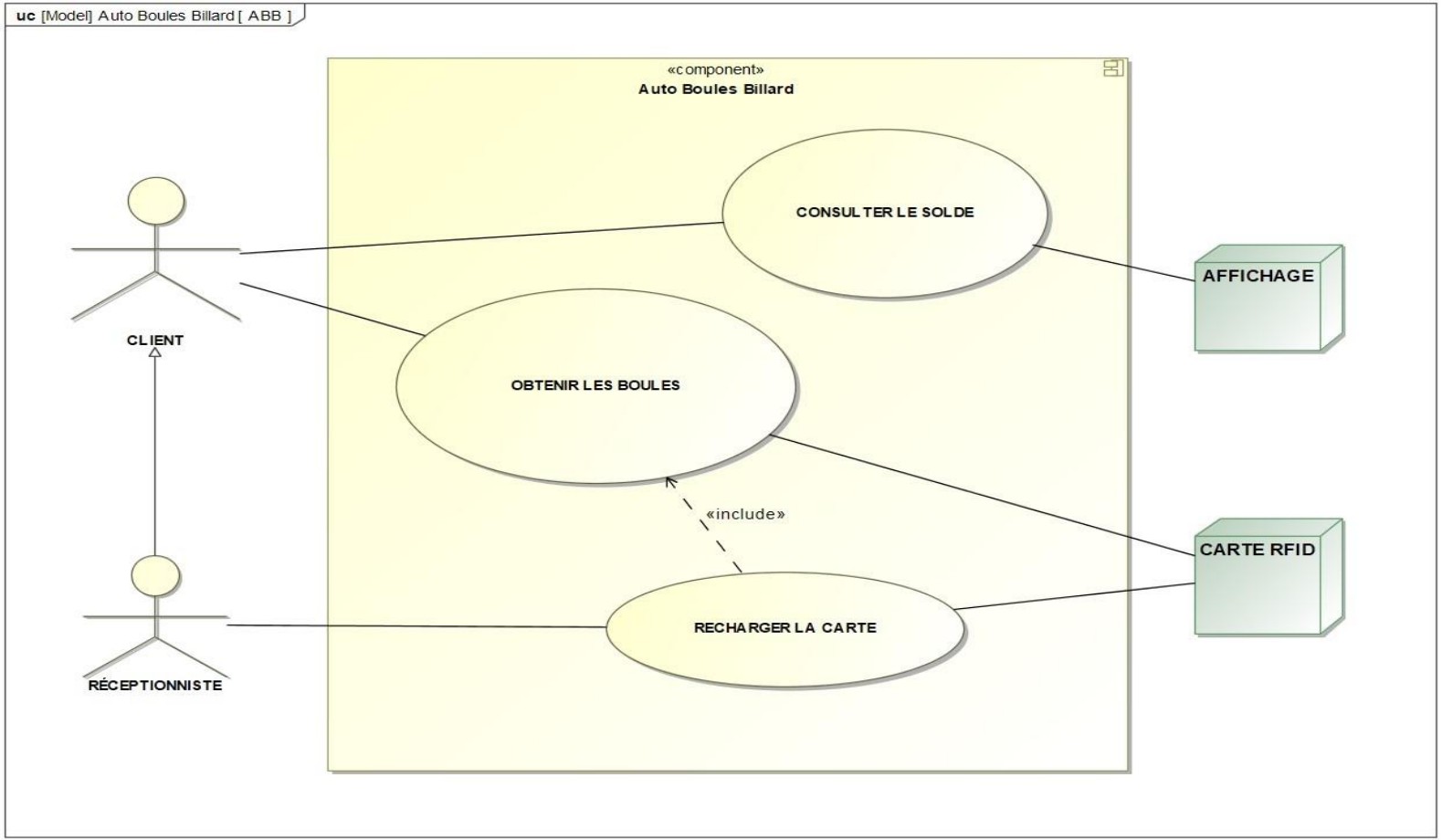
Comment moderniser le système de paiement traditionnel par jetons utilisé sur les tables de billard pour surmonter ses problèmes techniques et pratiques, et ainsi offrir une gestion optimisée et une expérience utilisateur améliorée aux joueurs ?

SOLUTION PROPOSÉE

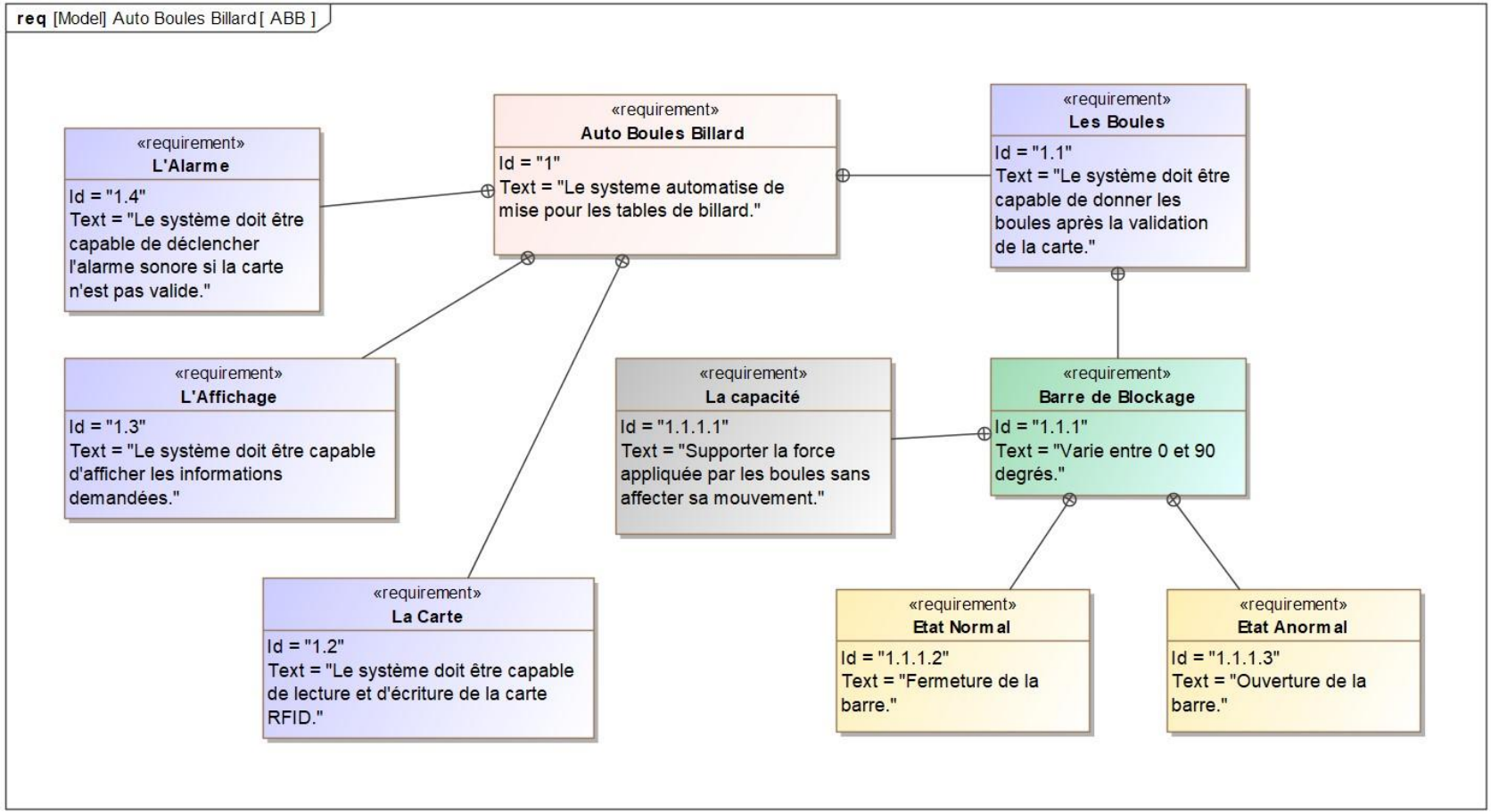
Remplacer le système actuel des tables de billard, par un système de paiement électronique basé sur des cartes rechargeables et un mécanisme électromécanique pour la distribution des boules



ANALYSE FONCTIONNELLE DU SYSTÈME COMPLET



ANALYSE FONCTIONNELLE DU SYSTÈME COMPLET



ANALYSE FONCTIONNELLE DU SYSTÈME COMPLET

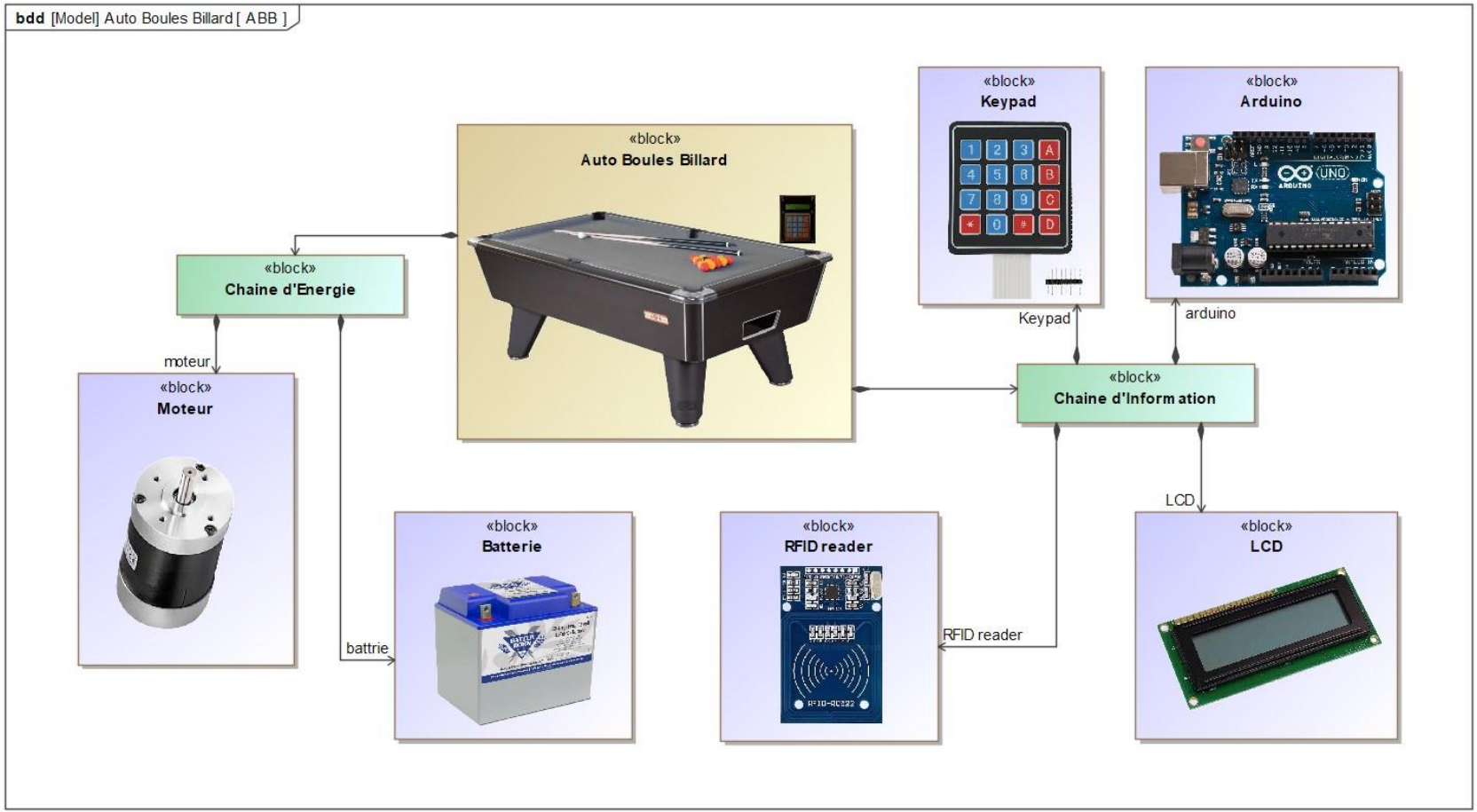
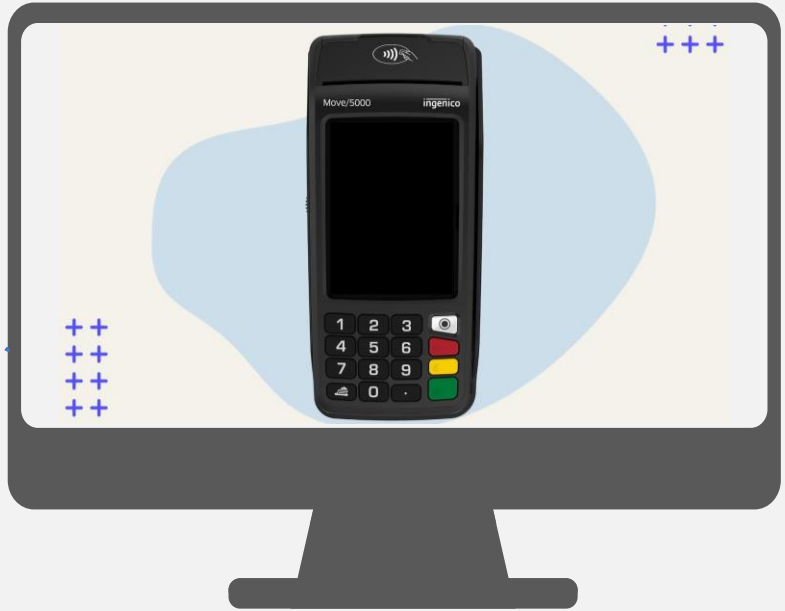


ABB-TOOL

L'exigence d'utiliser des cartes rechargeables sur les tables de billard nous a obligés à construire un appareil capable de gérer le solde des cartes. Nous l'avons appelé ABB-TOOL



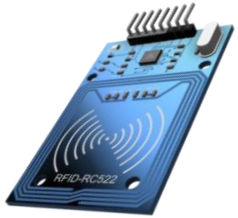
CHOIX DES MATÉRIAUX

Arduino Nano

Caractéristiques:

- Microcontrôleur : ATmega328
- Alimentation : 5V
- Vitesse de l'horloge : 16 MHz
- Analog IN Pins : 8
- Dimensions : 18 x 45 mm

RFID-RC522



Caractéristiques:

- Alimentation : 13mA / 3.3V
- Fréquence : 13.56MHz
- Distance de lecture : 5 cm
- Protocoles : ISO/MIFARE
- Data Transfer Rate: 424 kbps

LCD 16x2 avec I2C

Caractéristiques:

- Alimentation : 5V
- 2 lignes de 16 caractères.
- Rétro-éclairage bleu
- Adresse I2C : 0x20
- Dimensions : 36x79x20 mm

KeyPad 4x4

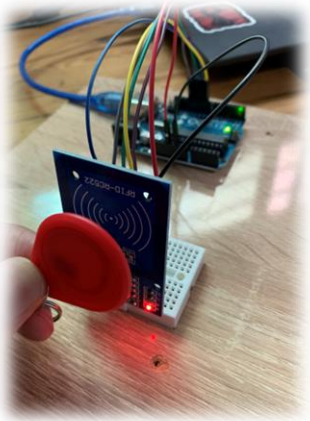
Caractéristiques:

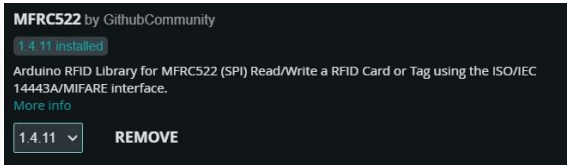
- Alimentation : 3.3V à 5V
- Config : 4 lignes / 4 colonnes
- Nombre de touches : 16
- Rebond de contact : <= 5 ms
- Dimensions : 69x76.9x0.8mm

CHOIX DES CARTES RECHARGEABLES

RFID Tag Mifare 1Kb

- Technologie : RFID MIFARE® Classic 1K EV1
- Norme : ISO14443A
- Fréquence : 13,56 Mhz
- Identifiant : 4 Byte NUID
- Anneau : 2 cm de diamètre





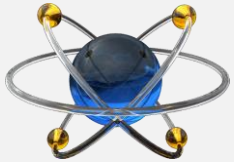
File => Exemples => MFRC522 => DumpInfo

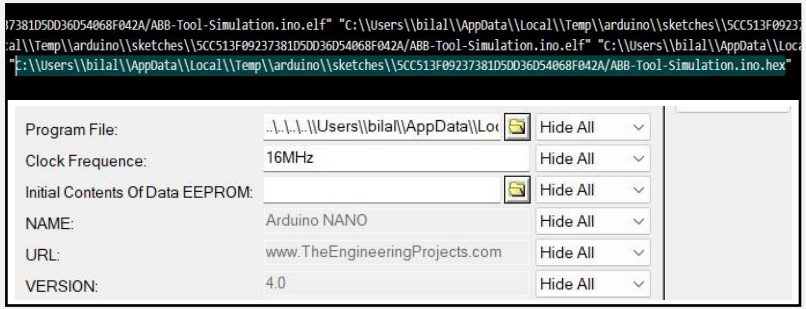
```
18:32:19.037 -> Firmware Version: 0x92 = v2.0
18:32:19.068 -> Scan PICC to see UID, SAK, type, and data blocks...
18:33:02.576 -> Card UID: D7 D0 60 7B
18:33:02.632 -> Card SAK: 08
18:33:02.643 -> PICC type: MIFARE 1KB
18:33:02.643 -> Sector Block   0  1  2  3   4  5  6  7   8  9 10 11  12 13 14 15  AccessBits
18:33:02.739 ->    15      63   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
18:33:02.805 ->            62   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
18:33:02.902 ->            61   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
18:33:02.970 ->            60   35 38 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
18:33:03.073 ->    14      59   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
18:33:03.141 ->            58   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
18:33:03.207 ->            57   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
18:33:03.304 ->            56   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
18:33:03.370 ->    13      55   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
```

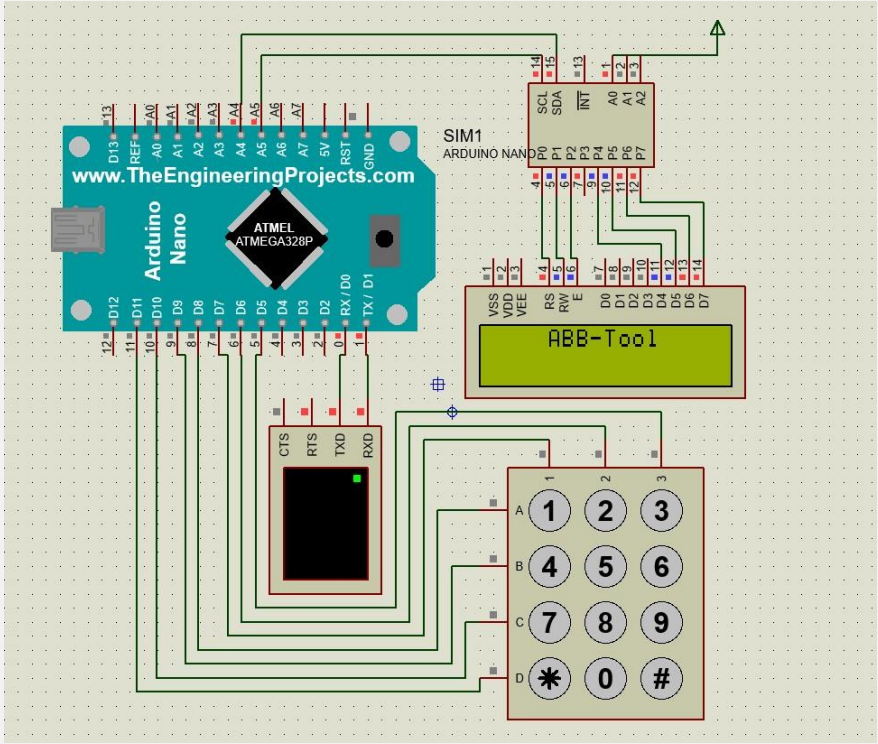
SIMULATION DU ABB-TOOL

- Nous avons implémenté la programmation sur Arduino IDE.

- Et nous avons effectué la simulation sur Proteus.



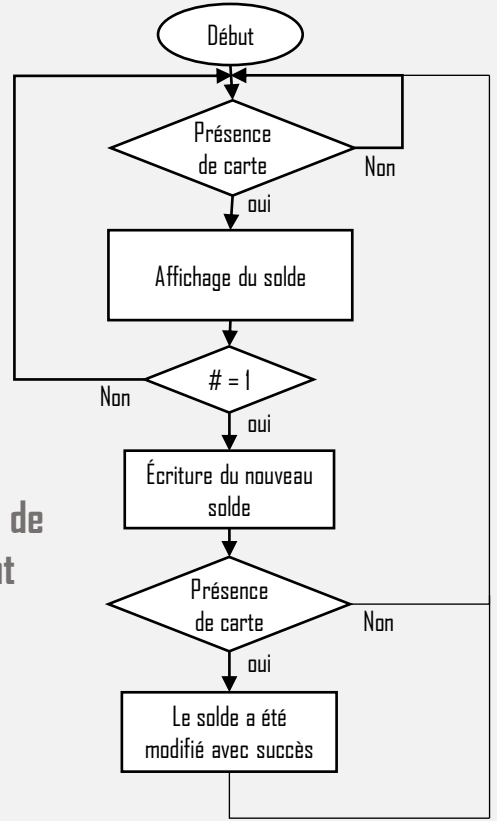




RÉALISATION DU ABB-TOOL

Le code de fonctionnement se trouve dans l'annexe 2

L'organigramme de fonctionnement



AUTO BOULES BILLARD

Le système automatisé intégré dans les tables de billard, capable de distribuer les boules par paiement avec des cartes électroniques

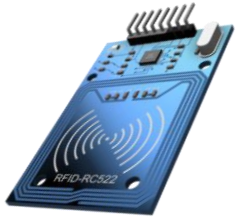
CHOIX DES MATÉRIAUX

Arduino Uno

Caractéristiques:

- Microcontrôleur : ATmega328
- Alimentation : 5V
- Vitesse de l'horloge : 16 MHz
- Analog IN Pins : 6
- Dimensions : 68 x 53 mm

RFID-RC522



Caractéristiques:

- Alimentation : 13mA / 3.3V
- Fréquence : 13.56MHz
- Distance de lecture : 5 cm
- Protocoles : ISO/MIFARE
- Data Transfer Rate: 424 kbps

Buzzer

Caractéristiques:

- Alimentation : 5V
- Décibel : > 85db/10cm
- Fréquence : 2500Hz
- Taille : 10 mm

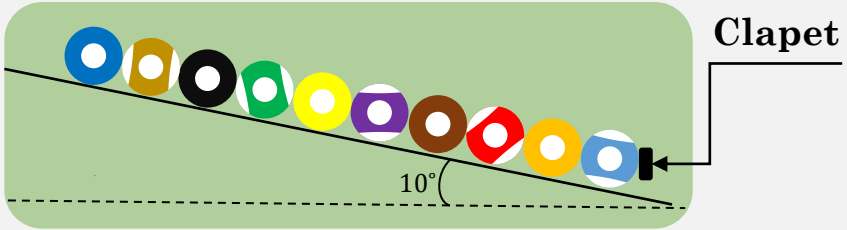
Bouton Poussoir

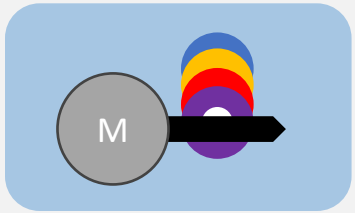
LCD 16x2 avec I2C

Caractéristiques:

- Alimentation : 5V
- 2 lignes de 16 caractères.
- Rétro-éclairage bleu
- Adresse I2C : 0x20
- Dimensions : 36x79x20 mm

CHOIX DU MOTEUR





Clapet

Faire une rotation entre 0° et 90°

Pour effectuer le choix, il est nécessaire de calculer le couple requis pour effectuer une rotation en présence des boules et du clapet.

❖ Données pour le Billard :

Masse de boule	0.16 kg
Longueur de clapet	0,1 m
Masse de clapet	0.3 kg
Vitesse angulaire en 0,5 seconde.	w = 1 rad/s

❖ Etape 1: Calcul du couple pour tourner le clapet :

➢ Le Moment d'inertie:

$$I = \frac{1}{3}\, m\, L^2 \quad \Rightarrow \quad I = 0,001\ kg \cdot m^2$$

➢ L'accélération angulaire :

$$\alpha = \frac{\omega}{\Delta t} \quad \Rightarrow \quad \alpha = 2\ rad/s^2$$

CHOIX DU MOTEUR

➢ Le couple Cp :

$$Cp = \alpha . I \quad \Rightarrow \quad \mathbf{Cp} = 0{,}002\ Nm$$

❖ **Etape 2: Calcul de la force exercée par les boules :**

➢ La force due a gravite sur chaque boules :

$$Fg = \text{m x g} = 0.16 \text{ x } 9.81 \quad \Rightarrow \quad \mathbf{Fg} = 1.569\ N$$

➢ La force totale applique a clapet le long de la pente :

$$Ft = \text{Fg x } 15 \text{ x } \sin(10) \quad \Rightarrow \quad \mathbf{Ft} = 4.05\ N$$

❖ **Etape 3: Calcul du couple totale nécessaire :**

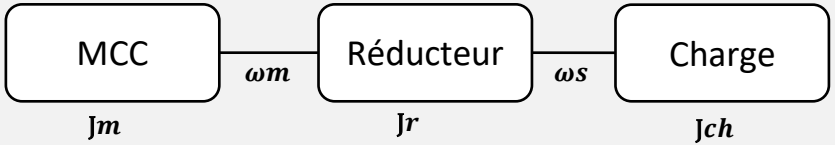
$$Ct = Cp + (Ft \text{ x } a) \quad \Rightarrow \quad \mathbf{Ct} = 0.1235\ N.m$$

- **a = 3 cm** : c'est la distance entre point d'application de la force Ft et l'axe de rotation du moteur

❑ **Moteur de choisi :** NF123G

Tension nominal	12 V
Le couple nominal	0.2 Nm
La vitesse de rotation	2500 tr/min
Le courant nominal	0.5 A

CHOIX DU MOTEUR



❑ Calcul de réducteur :

Étant donné la haute vitesse du moteur, l'utilisation d'un réducteur est probablement nécessaire.

- Angle de rotation : $\frac{\pi}{2}$ **(90°)**
- Le temps de réponse : **1 s**

Donc la vitesse de sortie : $\omega s = \theta/t = 1.57 rad/s$

- La vitesse angulaire moteur :

$$\omega m = 261.8 \text{ rad/s}$$

❖ Calcul du rapport de réduction :

Le rapport de réduction **r** nécessaire pour obtenir une vitesse de 1.57 rad/s à partir de 261.8 rad/s est :

$$r = \frac{\omega m}{\omega s} \Rightarrow r = 167$$

❖ Le couple de sortie Cs :

$$Cs = r \cdot Cm \Rightarrow Cs = 33.4 Nm$$

Ce couple est suffisamment pour entrainer le clapet et aussi à la force appliquée par les boulles de Billard

❖ Moment d'inertie totale :

$$J = jm + jr + \frac{I}{r^2} \Rightarrow J = 0.027\ 10^{-3}\ Kg.m^2$$

CHOIX DE LA BATTERIE

❖ **Choix de batterie se fait par :**

- La capacité en Ah.
- La tension nominale.

❖ **Calcul de la capacité de la batterie nécessaire :**

- le courant consommé par moteur : 0.5A
- l'autonomie demandé : 6 jours
- la durée de fonctionnement : 12h/jours

Par définition : $Cbat = I.\Delta t$

D'ou : $Cbat = 36\ Ah$

Choix : $Cbat = 50\ Ah$

Pour la raison de la sécurité et la Valeur consommé par les composants électronique

❖ **la batterie choisie :**

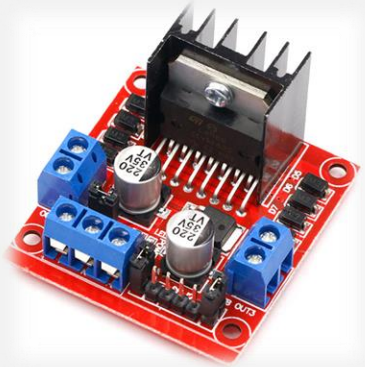
Battle Born 50Ah 12V LiFePO4 Deep Cycle Battery

Electrical Specification	
Voltage	12V
Capacity	50AH
Operating Temperature	- 4°F to 135°F (-20°C to 57.2°C)
Efficiency	99%
Self Discharge	2-3% per month
Maximum Series Voltage	48V
Cycles	3K-5K
Built-in BMS	Internal
Resistance	16 mΩ
Usable DoD	100%

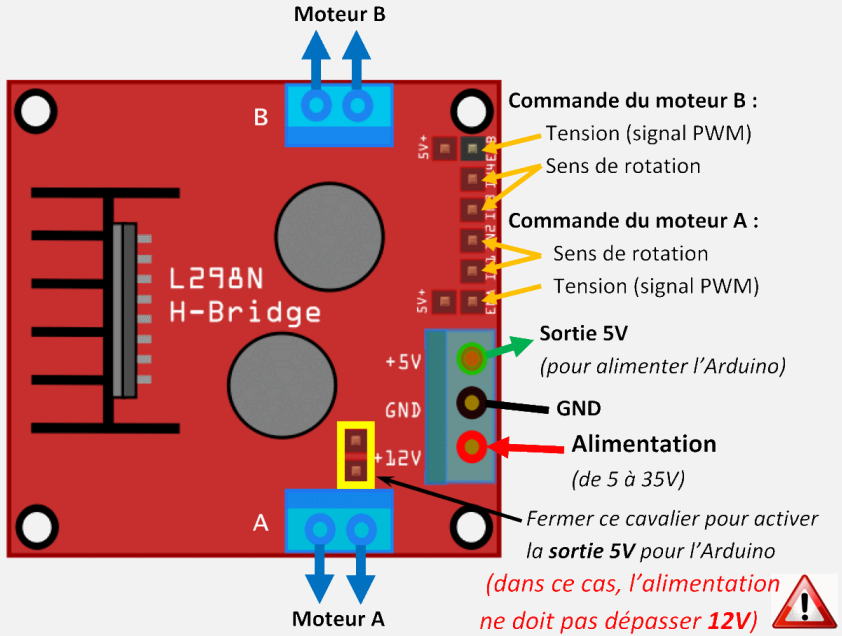
CHOIX DU HACHEUR

Pont en H L298N

Ce circuit, très populaire et bon marché, offre un bon moyen de piloter jusqu'à deux moteurs à courant continu.



❖ Description :

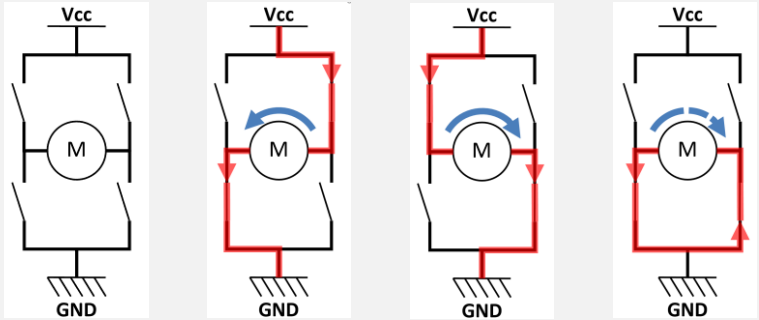


CHOIX DU HACHEUR

❖ Caractéristiques technique :

- Alimentation de la charge : de +6V à +35V
- Courant Max en pointe : 2A
- Tension de commande logique Vss : de +5 à +7V

❖ Schéma de principe :



❖ Câblage :

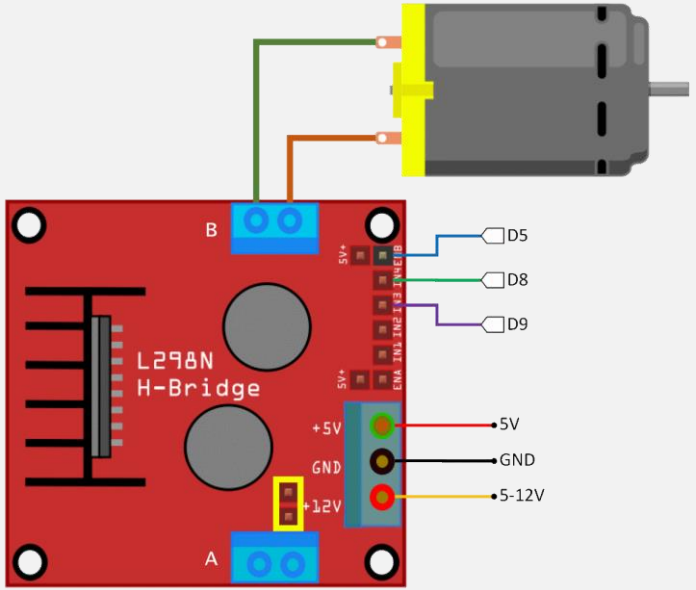
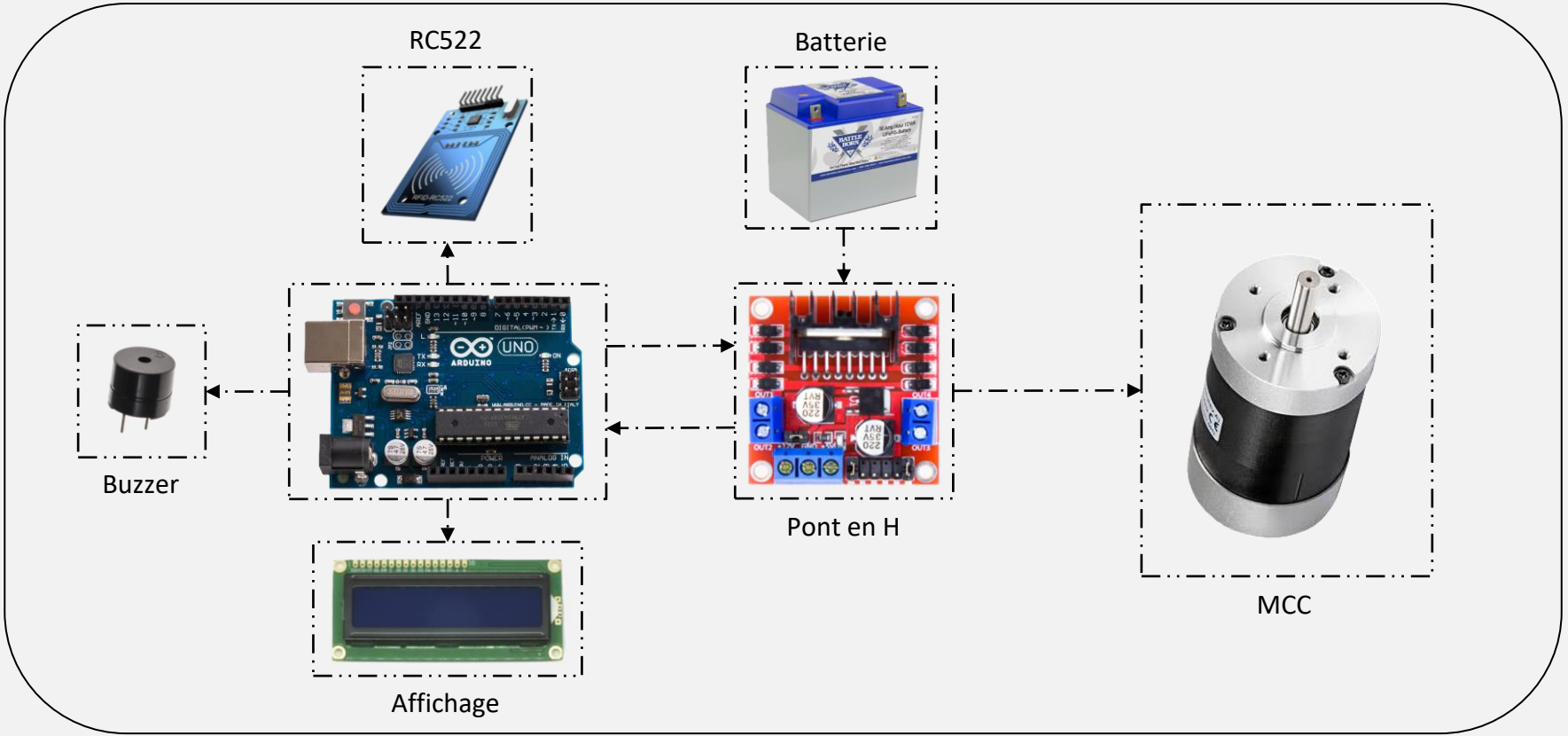


SCHÉMA D'INSTALLATION FINALE



ASSERVISSEMENT DE POSITION DE CLAPET

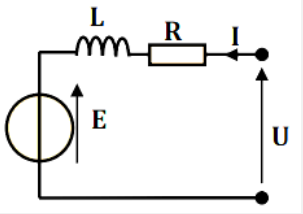
❑ Modélisation de la machine à CC :

❖ Caractéristique de la MCC :

Constante de couple	0.071 Nm/A
La résistance d'induit	1.55 Ω
Inductance d'induit	3.39 mH
Le moment d'inertie	0.27 Kg. cm^2

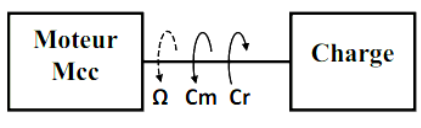
❖ Mise en équations :

Modèle électrique



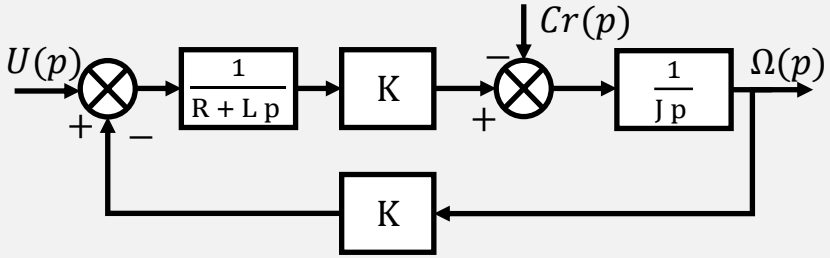
- $\mathbf{u(t) = e(t) + R\,i(t) + L\frac{di(t)}{dt}}$
- $\mathbf{e(t) = K\,\Omega(t)}$

Modèle mécanique

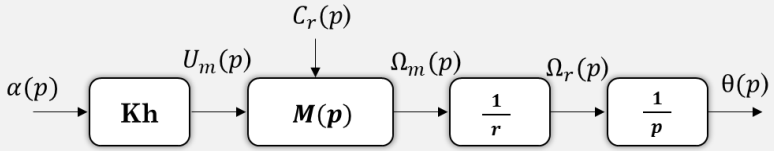


- $\mathbf{j\frac{d\Omega(t)}{dt} = Cm(t) - Cr(t)}$
- $\mathbf{Cm(t) = K\,i(t)}$

❖ Modèle de la machine à courant continu:



❖ Schéma bloc chaîne directe :

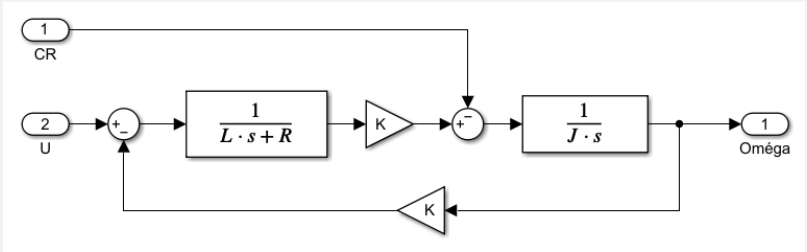


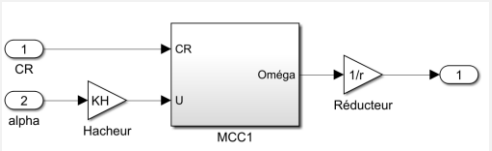
ASSERVISSEMENT DE POSITION DE CLAPET

❖ Objectifs :

- ❑ Asservissement et régulation de la boucle de vitesse de MCC
- ❑ Asservissement et régulation de la boucle de position pour asservir la position du clapet

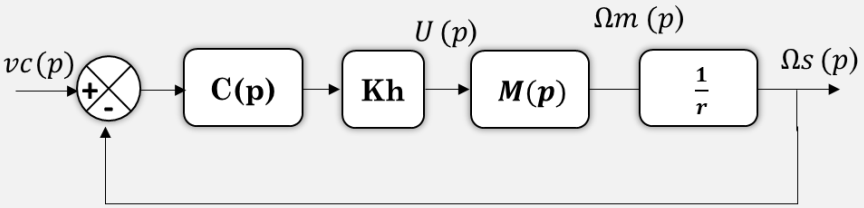
❖ la machine MCC sous MATLAB-Simulink :



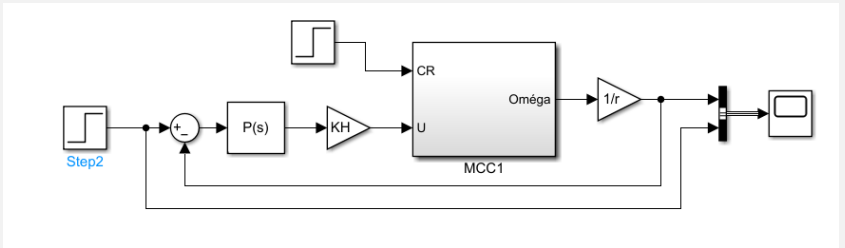


❑ Asservissement de la boucle de vitesse :

❖ Simulation de système non corrigé : C(p) = 1 :

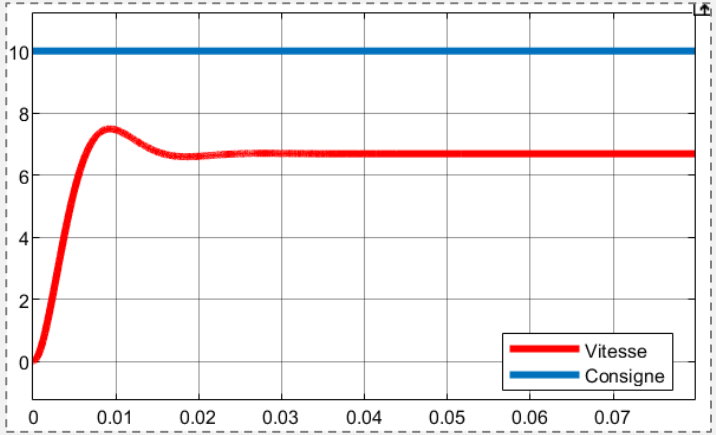


- ○ KH = 12
- ○ r = 167

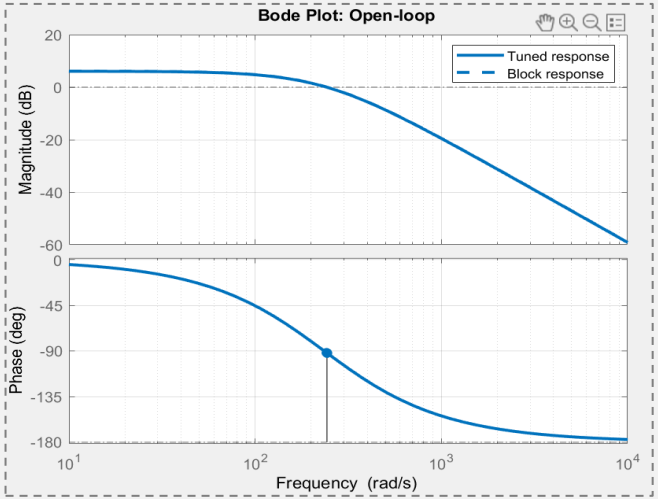


ASSERVISSEMENT DE POSITION DE CLAPET

❖ Résultats de simulation :



Dépassement (%)	11.9
Temps de réponse (ms)	14.3
Erreur statique (rad/s)	3



Marge de gain	infini
Marge de phase (°)	87
La bande passante (rad/s)	244

- Système n’est pas précis et très rapide.
- Le système est purement stable.

- Diminuer la bande passante (rapidité).
- Annuler l’erreur statique et le dépassement

ASSERVISSEMENT DE POSITION DE CLAPET

❖ Calculer le correcteur :

Stabilité	Le système doit être stable
Rapidité	Diminuer la rapidité à $\omega 1 = 120\ rad/s$
Précision	L'erreur statique doit être nulle
dépassement	Le dépassement doit être nul

Il est possible de satisfaire la plupart des exigences mentionnées en utilisant un correcteur de type PI.

○ *Fonction de transfert de correcteur :*

$$C(p) = Kp\left(1 + \frac{1}{Ti\ p}\right)$$

○ *Fonction de transfert en B.O :*

$$FTBO(p) = Kp\,\frac{1+Ti\,p}{Ti\,p}\,\frac{Ko}{(1+Tm.p)(1+Te.p)}$$

$Ko = \frac{KH}{r.\ K}$

$Ko = 2.02$ $Tm = 8.3\ ms$ $Te = 2.2\ ms$

Compensation de pôles : on choisi : **Ti = Tm**

$$\Rightarrow \quad FTBO(p) = \frac{Ko\ Kp}{Tm\ p.(1+Te.p)}$$

○ *Fonction de transfert en B.F :*

$$FTBF(p) = \frac{FTBO}{1+FTBO} = \frac{1}{1 + \frac{Te}{Ko\ Kp}p + \frac{Te\ Tm}{Ko\ Kp}p^2}$$

$$\Rightarrow \quad \omega'o = \sqrt{\frac{Ko\ Kp}{Te\ Tm}}$$

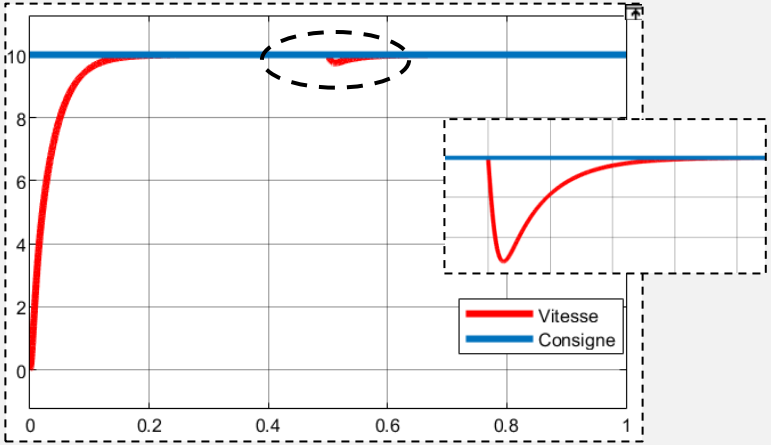
- on pose $\omega'o = \omega 1 = 120\ rad/s$ ⇨ $Kp = 0.13$

○ *Paramètres du correcteur :*

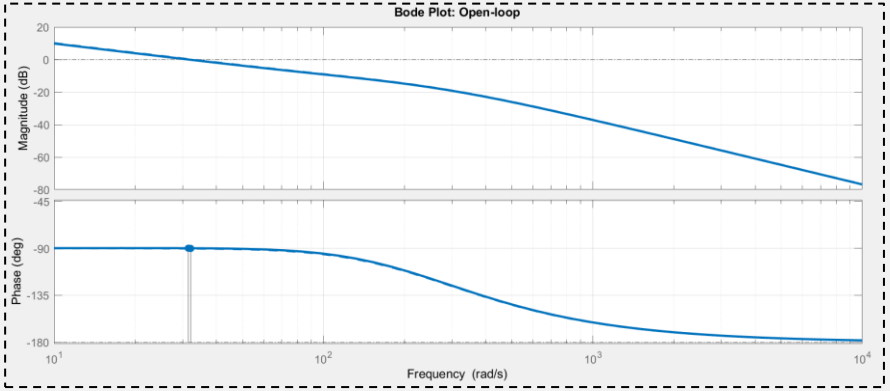
$Kp = 0.13$ $Ki = 1/Ti = 120$

ASSERVISSEMENT DE POSITION DE CLAPET

❖ Résultats de simulation :



Dépassement (%)	0
Temps de réponse (s)	0.125
Erreur statique (rad/s)	0



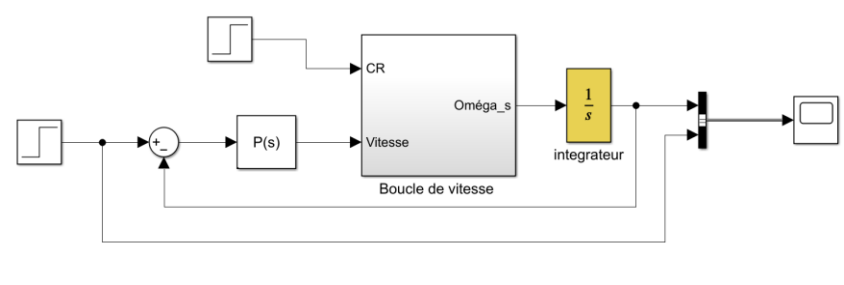
Marge de gain	infini
Marge de phase (°)	89

- Système est précis et rapide.
- Le système est purement stable

La boucle de vitesse est asservie et réglée en vitesse

ASSERVISSEMENT DE POSITION DE CLAPET

❑ Asservissement de la boucle de position :



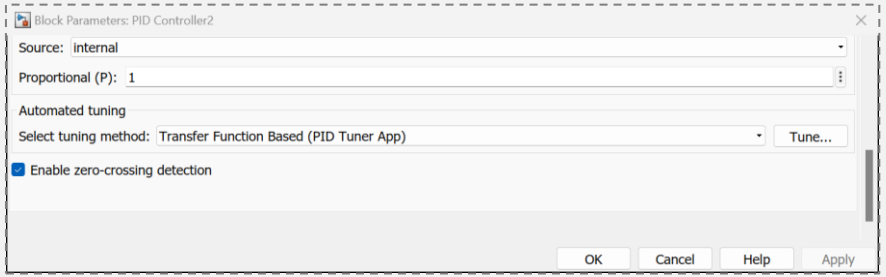
❖ Exigence d'asservissement et choix de correcteur :

Stabilité	Le système doit être stable
Rapidité	Temps de réponse **inférieur à 1 s**
Précision	L'erreur statique doit être nulle
dépassement	Le dépassement doit être nul

Il est possible de satisfaire la plupart des exigences mentionnées en utilisant un correcteur de type P.

❖ Calcul de correcteur :

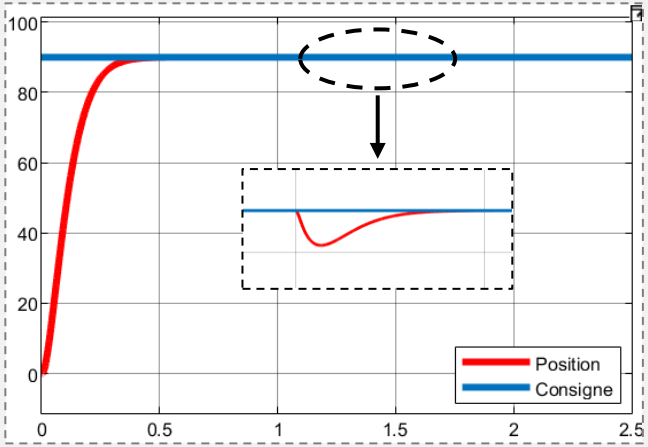
Dans cette partie on va utiliser le calcul de correcteur par la méthode numérique à travers logiciel Matlab.



⇨ La valeur de P : $Kp = 8.67$

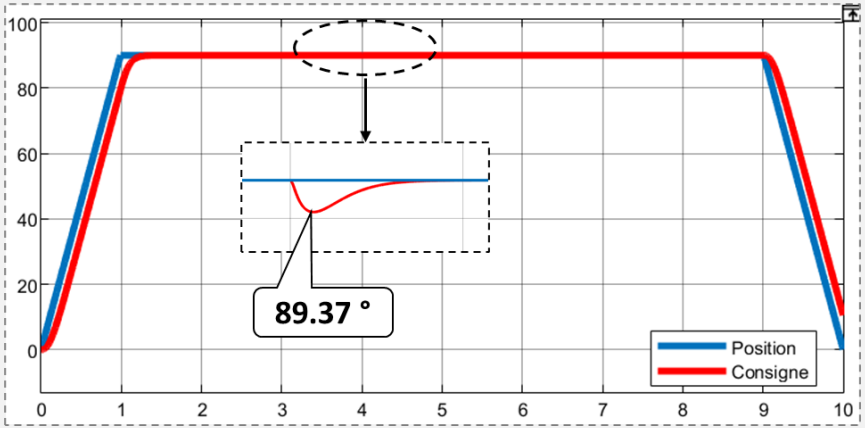
ASSERVISSEMENT DE POSITION DE CLAPET

❖ **Résultats de simulation :**



Temps de réponse (ms)	0.32
Erreur statique (rad/s)	0
Marge de gain (dB)	37
Marge de phase (°)	70
Dépassement (%)	0

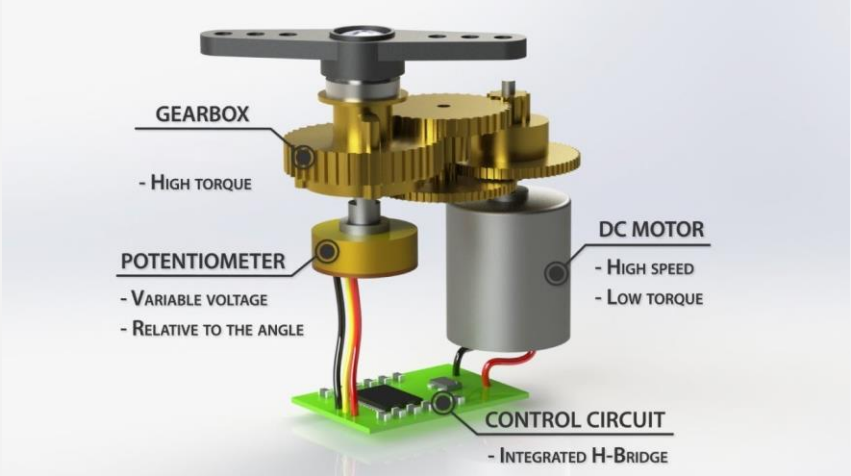
- Essai de notre système pour une consigne rampe de 0 à 90°:



SIMULATION ET RÉALISATION DU SYS-ABB

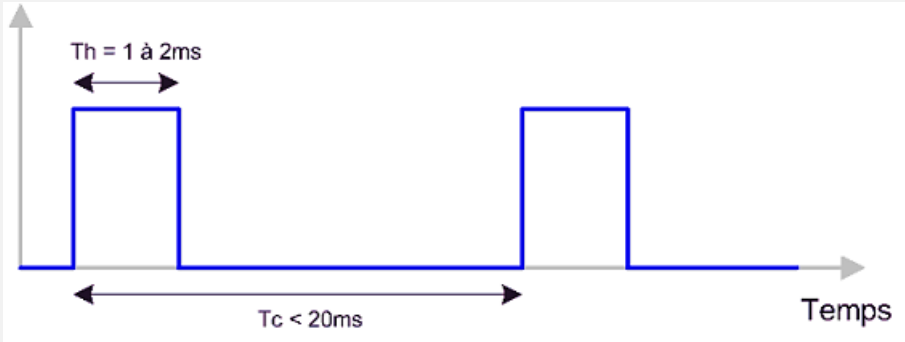
En raison du manque de matériel, la simulation et la réalisation complète contiendront un servo-moteur utilisé comme moteur à courant continu régulier en position.

❖ Description d'un servo-moteur :

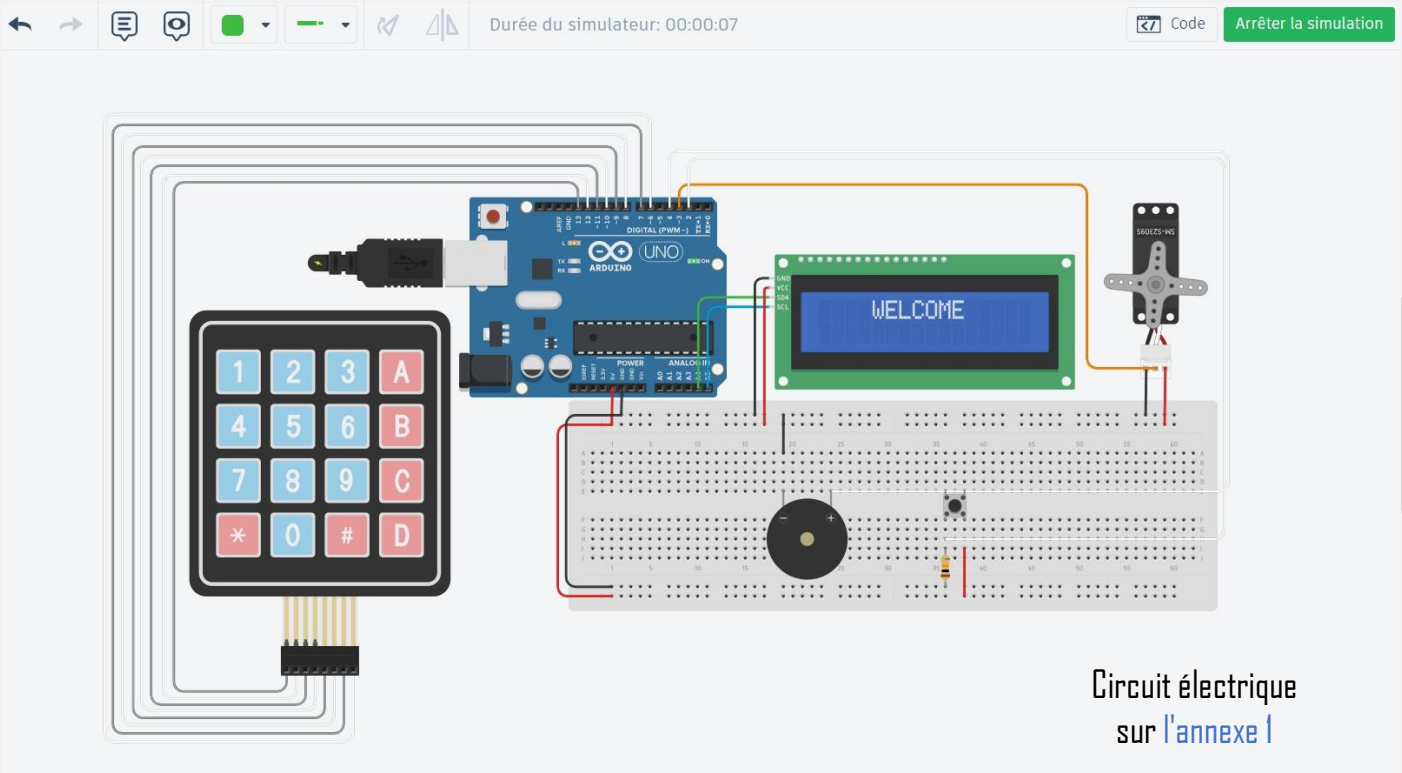


La commande d'un servomoteur nécessite la fourniture d'impulsions sur sa broche de commande. Ces impulsions devront être répétées à un intervalle de temps inférieur à 20ms:

- Une impulsion de 1,5 ms génère un angle de 0°
- Une impulsion de 1ms génère un angle de –90°
- Une impulsion de 2ms génère un angle de +90°



SIMULATION DU SYS-ABB

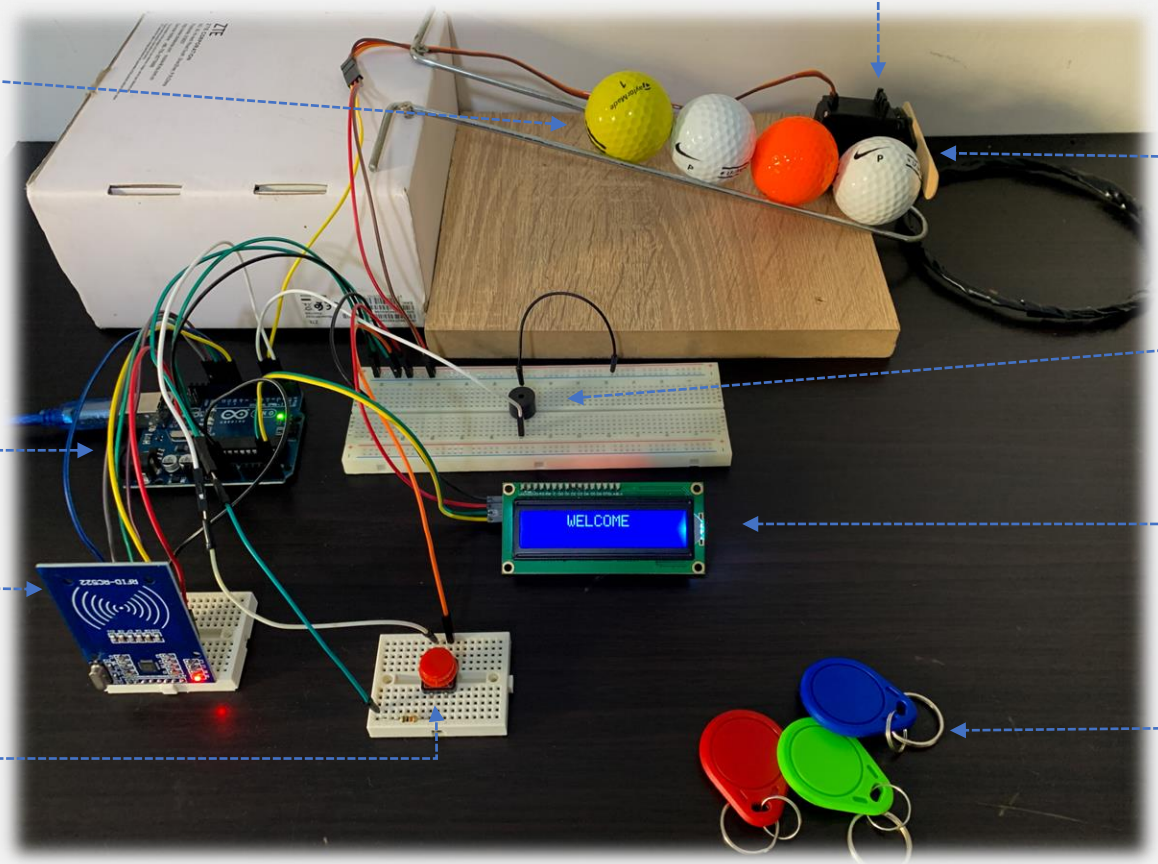


Circuit électrique sur l'annexe 1

TINKERCAD AUTODESK® TINKERCAD®

https://www.tinkercad.com/things/27jbd23vGF1-bilal-abb?sharecode=MyUk-8Ul-uaVWre2BhZQhGyGzMG1ZjsuBLrM_oYw6Wk

PROTOTYPE



Moteur

Boules

Clapet

Arduino

Buzzer

LCD

RC522

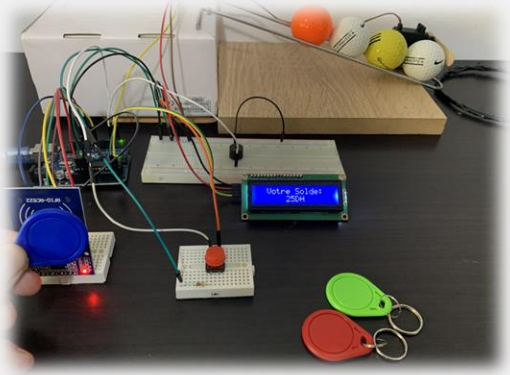
RFID tags

Bouton

PROTOTYPE

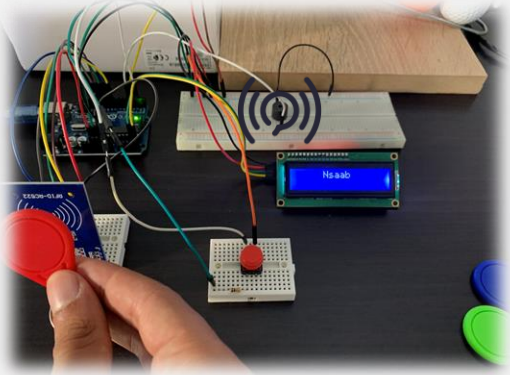
1

Consulter le Solde si la Carte est Valide



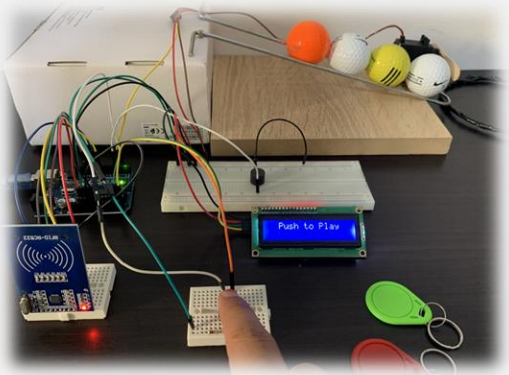
2

Si la carte n'est pas valide, déclencher l'alarme



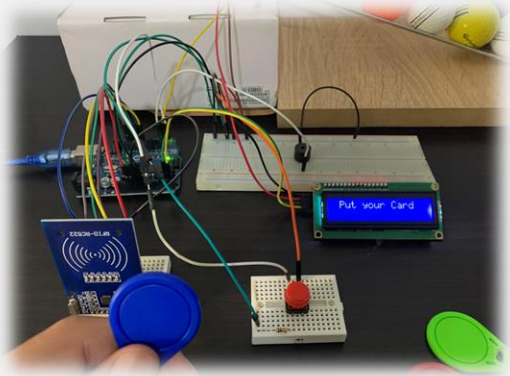
3

Appuyer sur le bouton pour jouer



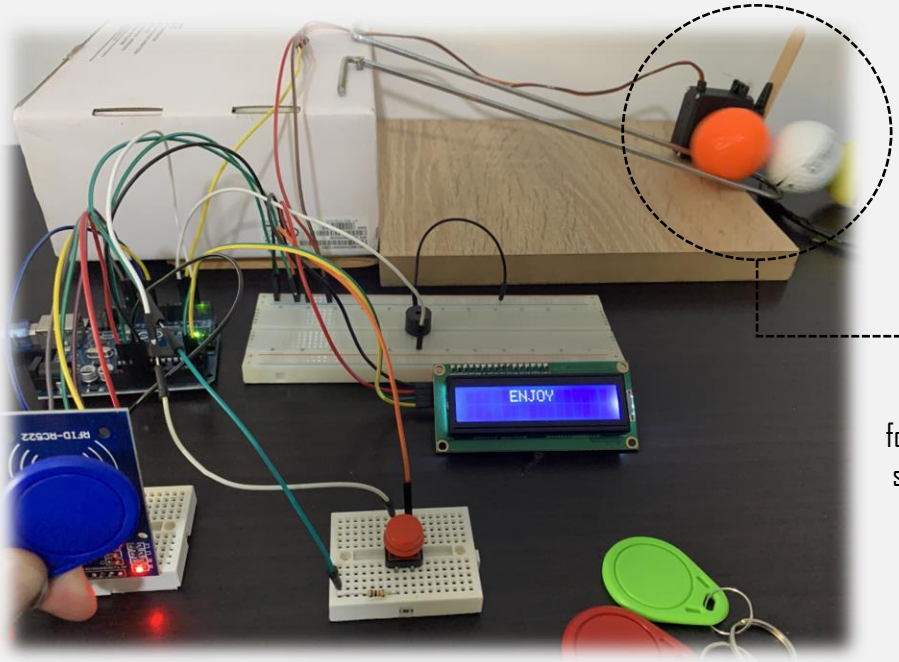
4

Placez la carte pour terminer le processus

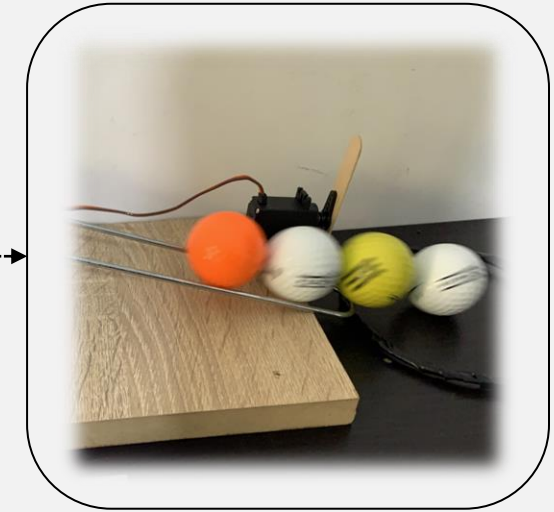


PROTOTYPE

Ouverture du clapet et sortie des boules

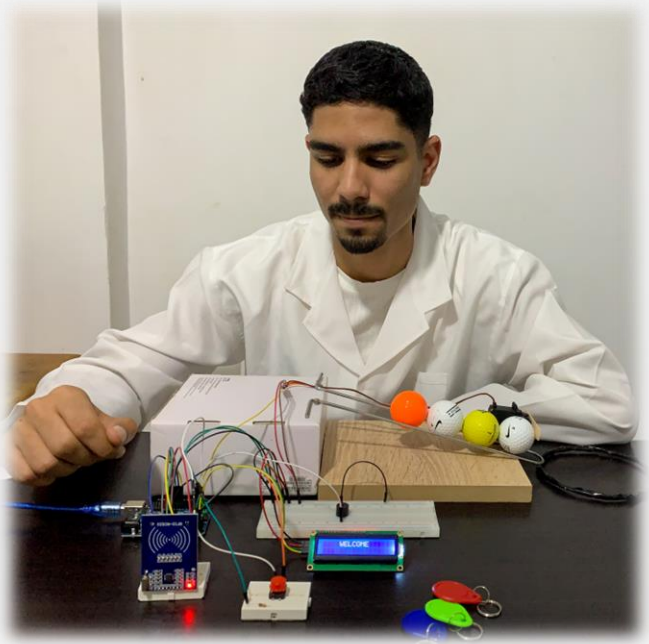


Le code de fonctionnement sur l'annexe 3



CONCLUSION

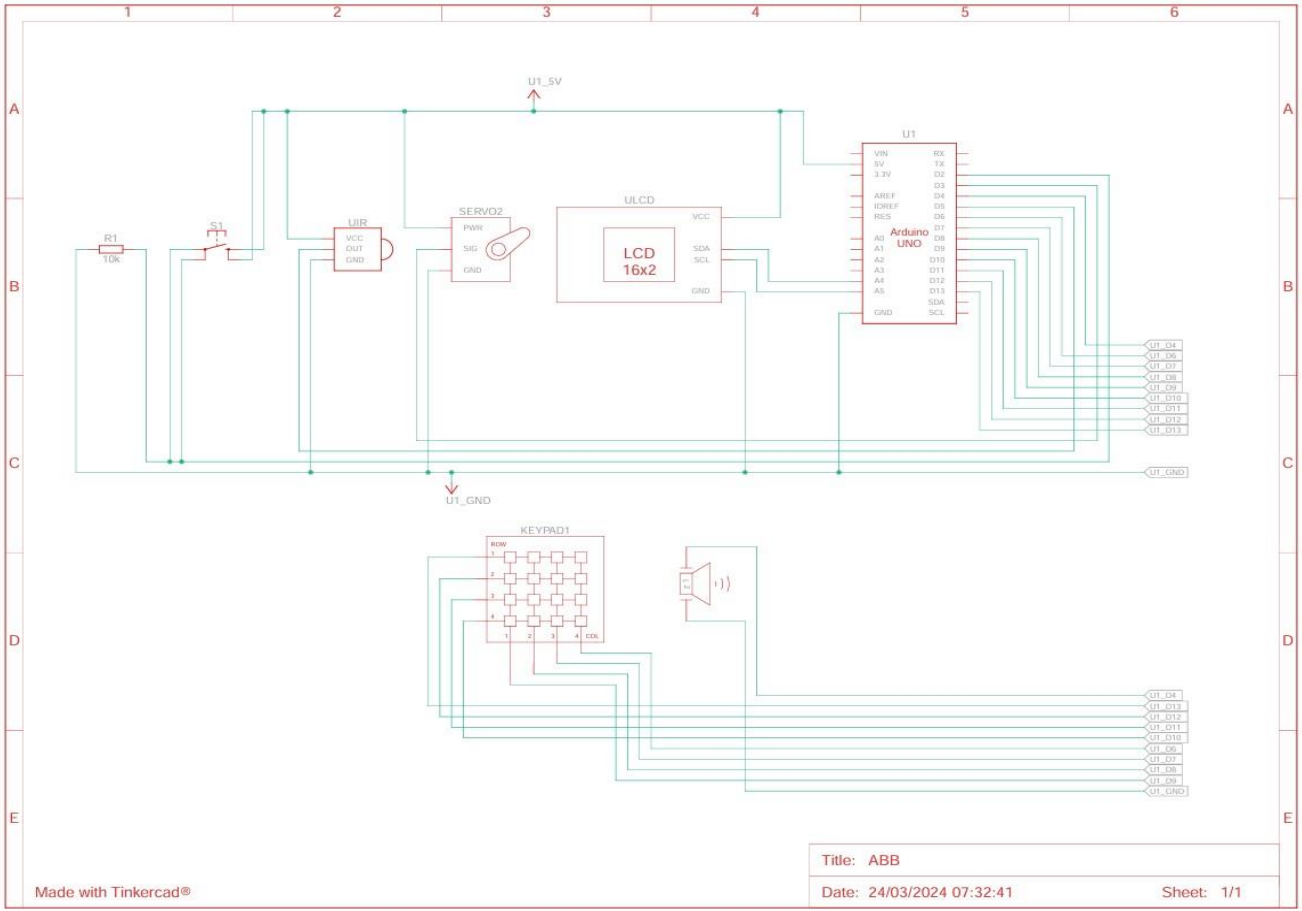
Mon projet a modernisé avec succès les tables de billard en intégrant un système de paiement électronique automatique. La transition des systèmes traditionnels vers des systèmes automatisés représente une avancée significative dans la technologie récréative, garantissant une expérience fluide et agréable pour tous les utilisateurs



MERCI POUR VOTRE

ATTENTION

ANNEXE 1


U1_5V
U1
R1
10k
S1
UIR
VCC
OUT
GND
SERVO2
PWM
SIG
GND
ULCD
VCC
SDA
SCL
GND
LCD
16x2
Arduino
UNO
U1_GND
KEYPAD1
ROW
COL
Made with Tinkercad®
Title: ABB
Date: 24/03/2024 07:32:41
Sheet: 1/1

ANNEXE 2

```
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <MFRC522.h>
#include <SPI.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
MFRC522 rc522(10, 9);
MFRC522::MIFARE_Key key;

const byte ROWS = 4;
const byte COLS = 3;
const char keys[ROWS][COLS] = {
  { '1', '2', '3' },
  { '4', '5', '6' },
  { '7', '8', '9' },
  { '*', '0', '#' }
};
byte rowPins[ROWS] = { 8, 7, 6, 5 };
byte colPins[COLS] = { 4, 3, 2 };
Keypad pad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

const byte PROGMEM IDs[3][4] = {
  { 0xB3, 0x60, 0x12, 0xAA },  //WHITE
  { 0xD3, 0x92, 0x83, 0xB7 },  //BLUE
  { 0xC9, 0xBB, 0xF3, 0xC0 }   //GREEN
};

int validecard() {
  for (int i = 0; i < 3; i++) {
    if (memcmp_P(rc522.uid.uidByte, IDs[i], 4) == 0) {
      return i;
    }
  }
  return -1;
}
```

```
int readcard() {
  MFRC522::StatusCode status;
  status = rc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 63, &key, &(rc522.uid));
  if (status != MFRC522::STATUS_OK) {
    lcd.clear();
    lcd.print("Auth failed");
    return;
  }

  byte buffer[18];
  byte size = sizeof(buffer);
  status = rc522.MIFARE_Read(60, buffer, &size);
  if (status != MFRC522::STATUS_OK) {
    lcd.clear();
    lcd.print("Read failed");
    return;
  }

  char block60Data[3] = {buffer[0], buffer[1], '\0'};
  return atoi(block60Data);
}

void writecard(String myData) {
  byte dataBlock[16] = {0};

  for (int i = 0; i < myData.length() && i < 16; i++) {
    dataBlock[i] = myData[i];
  }

  MFRC522::StatusCode status;
  status = rc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 63, &key, &(rc522.uid));
  if (status != MFRC522::STATUS_OK) {
    lcd.clear();
    lcd.print("Auth failed");
    return;
  }
```

```
  status = rc522.MIFARE_Write(60, dataBlock, 16);
  if (status != MFRC522::STATUS_OK) {
    lcd.clear();
    lcd.print("Write failed");
    return;
  }
}
void setup() {
  SPI.begin();
  rc522.PCD_Init();
  lcd.init();
  lcd.backlight();
  lcd.setCursor(4, 0);
  lcd.print("ABB-Tool");
  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }
}
void loop() {
  if (rc522.PICC_IsNewCardPresent() && rc522.PICC_ReadCardSerial()) {
    if (validecard() != -1) {
      int S = readcard();
      lcd.clear();
      lcd.setCursor(2, 0);
      lcd.print("Solde: ");
      lcd.print(S);
      lcd.print("DH");
      lcd.setCursor(0, 1);
      lcd.print("*:Return #:Edit");
      char key_pressed = NO_KEY;
      while (key_pressed != '#' && key_pressed != '*') {
        key_pressed = pad.getKey();
      }
      if (key_pressed == '#') {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("New Solde:");
        lcd.setCursor(14, 0);
        lcd.print("DH");
        lcd.setCursor(3, 1);
        lcd.print("#: Confirm");
        key_pressed = NO_KEY;
```

```
        String NS;
        lcd.setCursor(11, 0);
        while (key_pressed != '#' && key_pressed != '*') {
          key_pressed = pad.getKey();
          if (key_pressed != '#' && key_pressed != '*' && key_pressed != NO_KEY
          && NS.length() < 2) {
            lcd.print(key_pressed);
            NS += key_pressed;
          }
        }
        SPI.begin();
        rc522.PCD_Init();
        if (key_pressed == '#') {
          lcd.clear();
          lcd.setCursor(2, 0);
          lcd.print("Put your card");
          while (!rc522.PICC_IsNewCardPresent() ||
          | !rc522.PICC_ReadCardSerial()) {
          }
          writecard(String(NS));
          delay(500);
          lcd.clear();
          lcd.setCursor(3, 0);
          lcd.print("New Solde:");
          lcd.setCursor(6, 1);
          lcd.print(NS);
          lcd.print("DH");
          delay(4000);
        }
      }
    } else {
      lcd.clear();
      lcd.setCursor(5, 0);
      lcd.print("Nsaab");
      delay(6000);
    }
    lcd.clear();
    lcd.setCursor(4, 0);
    lcd.print("ABB-Tool");
    rc522.PICC_HaltA();
    rc522.PCD_StopCrypto1();
  }
```

ANNEXE 3

```
#include <LiquidCrystal_I2C.h>
#include <MFRC522.h>
#include <SPI.h>
#include <Servo.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
MFRC522 rc522(10, 9);
MFRC522::MIFARE_Key key;
Servo myservo;

const byte PROGMEM IDs[3][4] = {
  { 0xB3, 0x60, 0x12, 0xAA },  //WHITE
  { 0xD3, 0x92, 0x83, 0xB7 },  //BLUE
  { 0xC9, 0xBB, 0xF3, 0xC0 }   //GREEN
};
int validecard() {
  for (int i = 0; i < 3; i++) {
    if (memcmp_P(rc522.uid.uidByte, IDs[i], 4) == 0) {
      return i;
    }
  }
  return -1;
}
int readcard() {
  MFRC522::StatusCode status;
  status = rc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
  63, key, &(rc522.uid));
  if (status != MFRC522::STATUS_OK) {
    lcd.clear();
    lcd.print("Auth failed");
    return;
  }
  byte buffer[18];
  byte size = sizeof(buffer);
  status = rc522.MIFARE_Read(60, buffer, &size);
  if (status != MFRC522::STATUS_OK) {
    lcd.clear();
    lcd.print("Auth failed");
    return;
  }

  byte buffer[18];
  byte size = sizeof(buffer);
  status = rc522.MIFARE_Read(60, buffer, &size);
  if (status != MFRC522::STATUS_OK) {
    lcd.clear();
    lcd.print("Read failed");
    return;
  }

  char block60Data[3] = {buffer[0], buffer[1], '\0'};
  return atoi(block60Data);
}

void writecard(String myData) {
  byte dataBlock[16] = {0};

  for (int i = 0; i < myData.length() && i < 16; i++) {
    dataBlock[i] = myData[i];
  }

  MFRC522::StatusCode status;
  status = rc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 63, &key, &(rc522.uid));
  if (status != MFRC522::STATUS_OK) {
    lcd.clear();
    lcd.print("Auth failed");
    return;
  }

  status = rc522.MIFARE_Write(60, dataBlock, 16);
  if (status != MFRC522::STATUS_OK) {
    lcd.clear();
    lcd.print("Write failed");
    return;
  }
```

```
void loop() {
  if (rc522.PICC_IsNewCardPresent() && rc522.PICC_ReadCardSerial()) {
    if (validecard() != -1) {
      int S = readcard();
      if (S > 0) {
        digitalWrite(4, HIGH);
        delay(100);
        digitalWrite(4, LOW);
        lcd.clear();
        lcd.setCursor(2, 0);
        lcd.print(F("Votre Solde:"));
        lcd.setCursor(6, 1);
        lcd.print(S);
        lcd.print(F("DH"));
        delay(3000);
        lcd.clear();
        lcd.setCursor(2, 0);
        lcd.print(F("Push to Play"));
        SPI.begin();
        rc522.PCD_Init();
        unsigned long startTime = millis();
        while (millis() - startTime < 8000) {
          if (digitalRead(2) == HIGH) {
            lcd.clear();
            lcd.setCursor(1, 0);
            lcd.print("Put your Card");
            while (!rc522.PICC_IsNewCardPresent() |
            | !rc522.PICC_ReadCardSerial()) {
            }
            digitalWrite(4, HIGH);
            delay(100);
            digitalWrite(4, LOW);
            writecard(String(S - 5));
            delay(300);
            myservo.write(90);
            delay(30);
            lcd.clear();
            lcd.setCursor(5, 0);
            lcd.print(F("ENJOY"));
            delay(5000);
            myservo.write(180);
            delay(30);
            break;
          }
        }

      } else {
        digitalWrite(4, HIGH);
        delay(100);
        digitalWrite(4, LOW);
        delay(100);
        digitalWrite(4, HIGH);
        delay(100);
        digitalWrite(4, LOW);
        lcd.clear();
        lcd.setCursor(4, 0);
        lcd.print(F("Kiwaalo"));
        lcd.setCursor(6, 1);
        lcd.print(S);
        lcd.print(F("DH"));
        delay(5000);
      }
    } else {
      lcd.clear();
      lcd.setCursor(5, 0);
      lcd.print(F("Nsaab"));
      for (int i = 1; i < 50; i++) {
        digitalWrite(4, HIGH);
        delay(100);
        digitalWrite(4, LOW);
        delay(100);
      }
    }
    lcd.clear();
    lcd.setCursor(4, 0);
    lcd.print(F("WELCOME"));
    rc522.PICC_HaltA();
    rc522.PCD_StopCrypto1();
  }
}
```