Faculté des Sciences
كلية العلوم

Fascicule des travaux pratiques
Filière : Master Génie Logiciel pour le cloud
Module: IoT and cloud computing
Année Universitaire : 2023-2024

Application I : Gestion de feux de carrefour via (ESP8266)

Enoncé du problème :

On désire gérer les feux de carrefour (figure 1) selon plusieurs modes de fonctionnements dont ces modes sont décrits ci-dessous.

ACTIONNEURS		CAPTEURS	
Désignation	**Symbole**	**Désignation**	**Symbole**
Lampe rouge voie A LR_A	R_A	Commutateur feux normaux /clignotants N/CL	N
Lampe verte voie A LV_B	V_A	Commutateur feux Intelligents I	I
Lampe orangé voie A LO_A	O_A	Commutateur marche-Arrêt CM/A	E
Lampe rouge voie B LR_B	R_B	Capteur ultrasonique 1 (Voie A)	CapA
Lampe verte voie B LV_B	V_B	Capteur ultrasonique 2(Voie B)	CapB
Lampe orangé voie B LO_B	O_B		

Nomenclature :

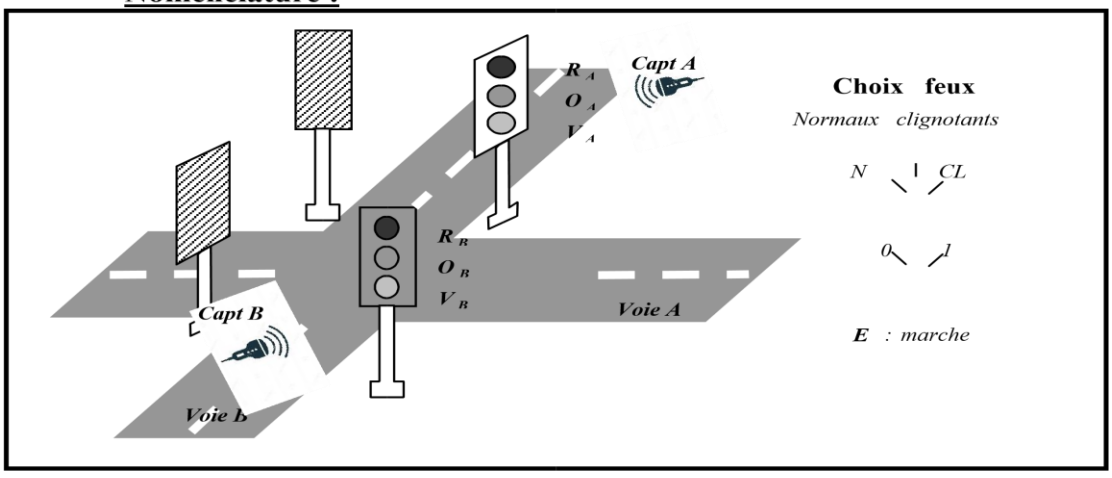


Figure 1 - Schéma de principe du carrefour.

Manipulation 1 : 1er mode de Fonctionnement.

Objectif : L'objectif de cette première Manipulation est d'écrire un programme en langage Python qui utilise les LEDs et les boutons poussoirs, d'implanter ce programme sur une carte ESP8266 et de faire des investigations pertinentes.

TRAVAIL DEMANDE : (Mode de fonctionnement normale d'un carrefour).

On désire automatiser la gestion de feux tricolores, réglant la circulation d'un carrefour à deux voies à l'aide d'un ESP8266 . Le cahier des charges stipule la possibilité de moduler les temporisations par l'opérateur en fonction de la densité de la circulation. L'étude est simple, simule les feux de circulation sur 4 groupes de 3 leds rouge-jaune-vert.

Les feux du carrefour doivent fonctionner en deux modes : tricolores dit feux normaux N et clignotants CL. Le choix d'un mode de fonctionnement est confié à l'agent de service. Le choix, pour N=1, mène le fonctionnement suivant :

- ☞ de 6H00 à 23H00 → feux tricolores FN ;
- ☞ de 23H00 à 6H00 → feux clignotants CL ;

Pour les feux tricolores les durées d'allumage sont fixées comme suit : Rouge : 10 s , Orangé : 3 s et Vert : 10 s.

On peut, sur demande de l'agent de service, pendant le jour, passer instantanément en mode clignotant N=0.

- ☞ Clignotement de la lampe orange sur une durée de 2 s.

Préparation du montage et programmation de la carte ESP8266 :

- Réaliser le montage de gestion de carrefour sur une plaque d'expérimentation.
- Proposer un programme Python répondant à ce cahier de charge.
- Implémenter le programme sur une carte ESP8266.
- Ensuite, tester le programme.

Manipulation 2 : 2éme mode de Fonctionnement.

Objectif : L'objectif de cette Manipulation est d'écrire un programmes en langage Python qui gère des LEDs, des boutons poussoirs et des capteurs ultrasoniques, puis d'implanter ce programme sur une carte ESP8266 et de faire des investigations pertinentes.

TRAVAIL DEMANDE : (Mode de fonctionnement intelligent d'un carrefour).

On désire automatiser la gestion de feux tricolores, réglant la circulation d'un carrefour à deux voies à l'aide d'un ESP8266 . Le cahier des charges stipule la possibilité de moduler les temporisations par système en fonction de la densité de la circulation on se basant sur des capteurs ultrasoniques installés sur les deux voies. L'étude simule les feux de circulation sur 4 groupes de 3 leds rouge-jaune-vert.

Les feux du carrefour doivent fonctionner en trois modes :

- Tricolores dit feux normaux N.
- Clignotants CL.
- Intelligent I.

Le choix d'un mode de fonctionnement est confié à l'agent de service. Le choix, pour N=1 et I=0, mène le fonctionnement suivant :

- ☞ De 6H00 à 23H00 → feux tricolores FN ;
- ☞ De 23H00 à 6H00 → feux clignotants CL ;

Pour les feux tricolores les durées d'allumage sont fixées comme suit : Rouge : 10 s, Orangé : 3s et Vert : 10 s.

On peut, sur demande de l'agent de service, pendant le jour, passer instantanément en mode clignotant N=0 et I=0.

- ☞ Clignotement de la lampe orange sur une durée de 2 s.

Le choix du mode de fonctionnement intelligent (N=1, ∀M) entraine :

- ☞ Etape 1 : Etape initiale caractérisant l'état avant le blocage de la route. Tous les feux sont éteints.
- ☞ Etape 2 : Etape de mise en marche de l'installation. Les feux rouges sont allumés sur les deux voies afin de permettre de dégager la voie par les véhicules bloqués au milieu de la route.
- ☞ Etape 3 : Présence de véhicule sur la voie A. On doit maintenir le rouge pendant une durée de 10 secondes pour les mêmes raisons indiquées sur l'étape 2.
- ☞ Etape 4 : Allumage du voyant vert de la voie1 pendant une durée de 20 secondes au minimum.
- ☞ Etape 5 : Maintien du vert à la fin des 20 secondes jusqu'à la détection d'un véhicule sur la voie B ou mise à l'arrêt du mode Intelligent.
- ☞ Etape 6 : Présence de véhicule sur la voie B. On doit maintenir le rouge pendant une durée de 10 secondes pour les mêmes raisons indiquées sur l'étape 2.
- ☞ Etape 7 : Allumage du voyant vert de la voie B pendant une durée de 20 secondes au minimum.
- ☞ Etape 8 : Maintien du vert à la fin des 20 secondes jusqu'à la détection d'un véhicule sur la voie A ou mise à l'arrêt du mode Intelligent. **Préparation du montage et programmation de la carte ESP8266 :**

- Réaliser le montage de gestion de carrefour sur une plaque d'expérimentation.
- Proposer un programme ESP8266 répondant à ce cahier de charge.
- Implémenter le programme sur une carte ESP8266 .
- Ensuite, tester le programme.

Guide de connexion et manipulation du capteur de distance ultrason hc-sr04 sur esp8266 .

1. Présentation du Capteur :

Le capteur à ultrasons HC-SR04 utilise le principe du sonar pour déterminer la distance à un objet (Figure 2). Il offre une mesure sans contact avec une bonne précision et des lectures stables. Sur le module, nous trouvons un émetteur à ultrasons et un récepteur. Le capteur a une portée de 2 à 400 cm. La précision de la mesure est de +/- 0,5 cm.

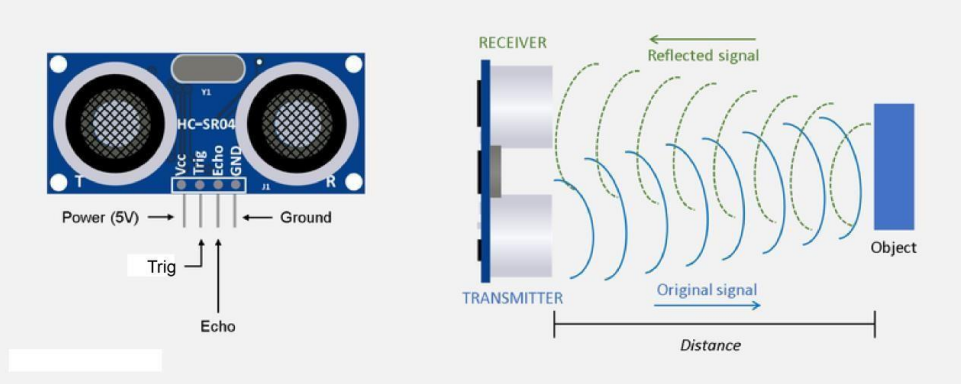


Figure 2 - Composants et fonctionnement du capteur à ultrasons.

Le capteur utilise des signaux sonores ultrasoniques pour définir la distance entre le capteur et l'objet :

- L'émetteur envoie un signal ultrason dans la direction de l'objet.
- Lorsque le signal atteint l'objet, il est renvoyé au capteur.
- Le récepteur peut détecter le signal réfléchi.

Nous pouvons calculer la différence de temps entre le moment où l'émetteur envoie le signal et le moment où le récepteur capte le signal. Puisque la vitesse du son est connue (343,3 m / s ou 34330 cm / s), nous pouvons maintenant calculer la distance comme ceci :

Distance = vitesse*temps Ou

dans notre cas :

Distance = 34330*temps/2

Où

Distance =17165*temps

Remarque : le temps mesuré prend en compte la distance du capteur à l'objet et la distance de l'objet au capteur, il faut donc diviser ce temps par 2.

Soyez prudent ! Avant de brancher des câbles sur les broches GPIO de votre ESP8266 , assurez-vous d'éteindre correctement le et de retirer le câble d'alimentation de la carte mère !

2. Mise en place de la partie matérielle

Le capteur ultrason contient 4 broches : o Broche VCC alimente le capteur. o Broche GND est la masse. o Broche Trigger déclenche l'envoi d'un train d'impulsions à 40 kHz.

o Broche Echo fournir une impulsion + 5v dont la durée est proportionnelle à la distance si le module détecte un objet.

Pour relier le capteur ultrason HC-SR04 à la carte ESP8266 il faut suivre le étapes suivantes :

- Reliez la broche VCC du capteur à une broche 5V (fil rouge)
- Reliez la broche Trig du capteur à la broche 23
- Connectez la broche Echo du capteur à la résistance de 1 kΩ, connectez l'autre extrémité de la résistance à la broche 24 (fil jaune)
- Connectez le GND du capteur à la rangée "-" de la planche à dessin et connectez également le GND GPIO à la rangée "-" de la planche à pain (câble noir)

Application II : Manipulation de la Technologie RFID

Les tags RFID (Radio Frequency Identification) ne sont pas nouveaux mais ils sont plus que jamais au cœur des systèmes connectés et de l'IoT. Ces puces électroniques supplantent les étiquettes imprimées en permettant une localisation précise des produits. L'association du cloud et des objets connectés a remis ces étiquettes d'identification au-devant de la scène.

I. Objectifs de la Manipulation.

- Comprendre le fonctionnement et l'utilité de la technologie RFID
- Savoir les caractéristiques techniques du module RFID RC522
- Se familiariser à l'utilisation du module RC522 dans un projet du contrôle d'accès
- Se familiariser à l'utilisation de la librairie dédiée au module RC522 Autres astuces pratiques

II. Introduction à la Technologie RFID

La présente application est une introduction à une série des projets qui abordent la technologie RFID. «Radio-Frequency IDentification» est une technologie pour laquelle les données numériques codées dans des étiquettes RFID ou « tags ». Elles sont capturées par un lecteur via des ondes radio. La RFID est similaire aux codes-barres dans la mesure où les données d'une étiquette sont capturées par un appareil qui stocke les données dans une base de données. La RFID présente toutefois plusieurs avantages par rapport aux systèmes utilisant un logiciel de suivi des actifs de codes-barres. En effet, le code barre nécessite un scanner optique.

La RFID appartient à un groupe de technologies appelées Automatic Identification and Data Capture (AIDC). Les méthodes AIDC identifient automatiquement les objets, collectent des données les saisissent directement dans des systèmes informatiques avec une intervention humaine minime. Les méthodes RFID utilisent des ondes radio pour y parvenir. À un niveau simple, les systèmes RFID se composent de trois composants : une étiquette RFID « RFID tag », un lecteur RFID et une antenne. Les étiquettes RFID contiennent un circuit intégré et une antenne, qui permettent de transmettre des données au lecteur RFID. Le lecteur convertit ensuite les ondes radio en une forme de données plus utilisable. Les informations collectées à partir des étiquettes sont ensuite transférées via une interface de communication vers un système informatique hôte pour les traitements ultérieurs. La technologie RFID basée sur des transferts d'énergie par liaison radio en utilisant des antennes électromagnétiques. Pour la mettre en

application, il est nécessaire de disposer de marqueurs (étiquettes, tags ou puces RFID) et d'un lecteur RFID.

1. Les différents domaines d'applications de la technologie RFID.

- Gestion de l'inventaire
- Suivi des actifs
- Suivi du personnel
- Contrôle de l'accès aux zones restreintes
- Badge d'identification
- Gestion de la chaîne logistique
- Prévention de la contrefaçon (par exemple dans l'industrie pharmaceutique)

2. Le module RFID RC522 à 13.56MHz

Le module est un lecteur de la puce RFID basé sur le circuit MFRC522 à faible coût est facile à utiliser. Il peut être utilisé dans une large gamme d'application. Le MFRC522 est un circuit intégré de lecture / écriture hautement intégré pour la communication sans contact à 13,56 MHz. Ci-dessous les caractéristiques du module :

- Basé sur le circuit MFRC522
- Fréquence de fonctionnement : 13,56 MHz
- Tension d'alimentation : 3.3V
- Courant : 13-26mA
- Portée de lecture : Environ 3 cm avec la carte et le porte-clés fournis
- Interface de communication : SPI
- Taux de transfert de données maximum : 10 Mbit / s
- Dimensions : 60mm × 39mm
- MFRC522 Datasheet : https://www.electronique-mixte.fr/wp content/uploads/2019/08/MFRC522-Datasheet

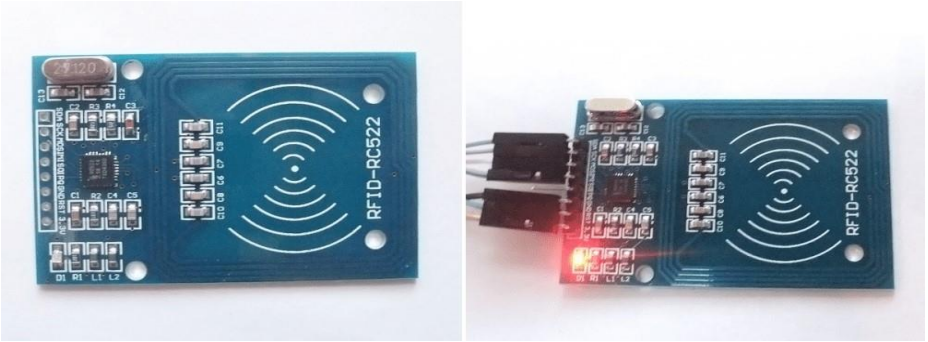


Figure 1: Le module RFID RC522

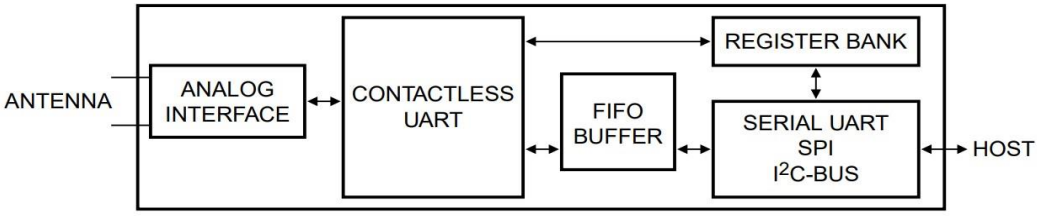


Figure 2 : Blocs simplifies du module RFID RC522

III. Projet : contrôle d'accès par badge avec Arduino

L'application consiste l'ouverture d'une porte en utilisant un badge. Le lecteur RFID couplé à la carte Arduino permet de détecter un badge enregistré ou non. Lorsque l'utilisateur est reconnu, le système déclenche l'ouverture de la porte ou une alarme dans le cas échéant. L'utilisateur à droit de trois tentatives. Le nombre de tentatives est ajustable par le programme Arduino. Ci-dessous les éléments constituant le projet ainsi leurs fonctionnements.

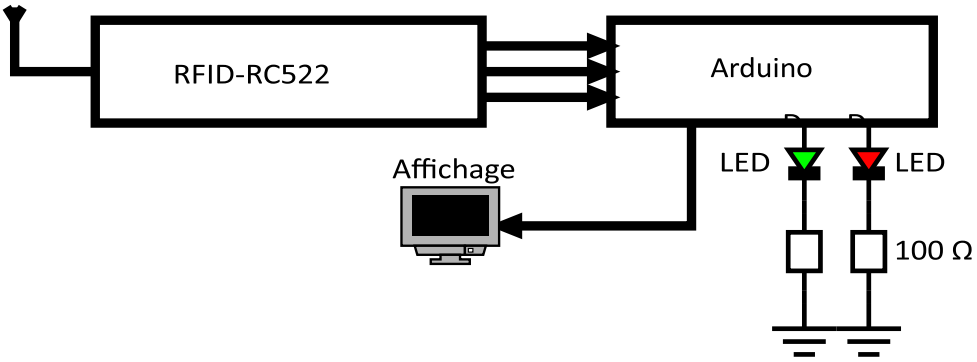


Figure 3: Les constituants du projet

- **RFID-RC522** : Lecteur du badge
- **Carte Arduino** : Elle est couplée avec le lecteur RFID. Elle permet de détecter la présence du badge, reconnaitre son identifiant (code du badge). Elle sert également à activer l'ouverture de la porte ou l'alarme
- **LED verte** : Voyant indiquant l'ouverture de la porte. La LED s'allume pendant une seconde lorsqu'un badge reconnu est détecté. Elle reste éteinte dans le cas contraire
- **LED rouge** : Voyant indiquant la détection d'un Fau badge (identifiant non reconnu du badge). La LED rouge s'allume pendant une seconde puis s'éteint pour chaque fausse détection. Lorsque le nombre de tentatives est atteint, la LED rouge clignote en boucle infinie en état d'alarme. Aucune moyenne n'est possible pour réactiver le système à part la réinitialisation de la carte Arduino.

Note : On peut ajouter un bouton de réinitialisation (non lisible pour l'utilisation) dédié à la désactivation de l'alarme.

RS232 : Le système communique l'état de la porte ou la présence de l'alarme par la liaison RS232. On affiche « Ouverture de la porte » lorsque la LED verte est allumée. Et « Alarme » en boucle lorsque le nombre de tentatives est attient.

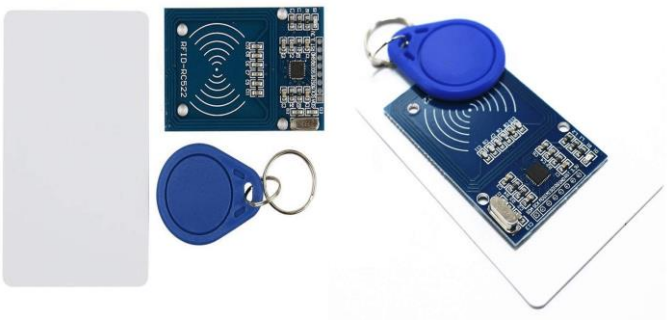


Figure 4 : Les tags du module RFID RC522

1. Câblage des composants

- LED rouge: D2
- LED verte: D3

RFID-ARDUINO:

Signal	RFID-RC522	ARDUINO
RST/Reset	RST	D9 5
SPI SS	SDA(SS)	D10 53
SPI MOSI	MOSI	D11 51
SPI MISO	MISO	D12 50
SPI SCK	SCK	D13 52

2. La lecture de l'ID d'un badge

Le module RFID est accompagné de deux badges de formes différentes (voir la figure 4) : l'un se forme d'une carte et l'autre d'une clé. Pour l'instant on ne connaît pas les identifiants de chacun d'entre eux. La première étape consiste à reconnaitre les ID pour les opérations à venir. Nous avons besoin de télécharger la librairie RFID (MFRC522.h). On considère le même schéma de câblage. Ci-dessous les étapes importantes de déclaration, initialisation, lecture et affichage de l'ID d'un badge.

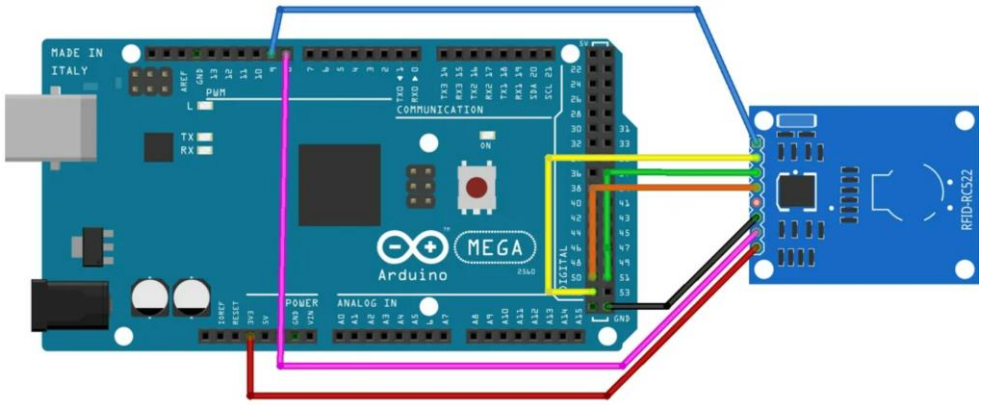


Figure 5: schème de câblage.

3. Le programme de La lecture de l'ID d'un badge

➢ **Déclaration**

```
#include <SPI.h> // SPI
#include <MFRC522.h> // RFID
#define SS_PIN 10
#define RST_PIN 9
// Déclaration
MFRC522 rfid(SS_PIN, RST_PIN);
// Tableau contentent l'ID
byte nuidPICC[4];
```

➢ **Initialisation**

```
void setup()
{
// Init RS232
Serial.begin(9600);
// Init SPI bus
SPI.begin();
// Init MFRC522
rfid.PCD_Init();
}
```

➢ **Attente d'un nouveau badge**

```
void loop()
{
  // Initialisé la boucle si aucun badge n'est présent
  if ( !rfid.PICC_IsNewCardPresent())
    return;
  // Vérifier la présence d'un nouveau badge
  if ( !rfid.PICC_ReadCardSerial())
    return;
```

➢ **Enregistrement de l'ID**

```
// Enregistrer l'ID du badge (4 octets)
  for (byte i = 0; i < 4; i++)
  {
    nuidPICC[i] = rfid.uid.uidByte[i];
  }
```

➢ **Affichage de l'ID**

```
  // Affichage de l'ID
```

```
  Serial.println("Un badge est détecté");
  Serial.println(" L'UID du tag est:");
  for (byte i = 0; i < 4; i++)
  {
    Serial.print(nuidPICC[i], HEX);
    Serial.print(" ");
  }
  Serial.println();
```

- **Re-Initialisation RFID**

```
// Re-Init RFID
  rfid.PICC_HaltA(); // Halt PICC
  rfid.PCD_StopCrypto1(); // Stop encryption on PCD
}
```

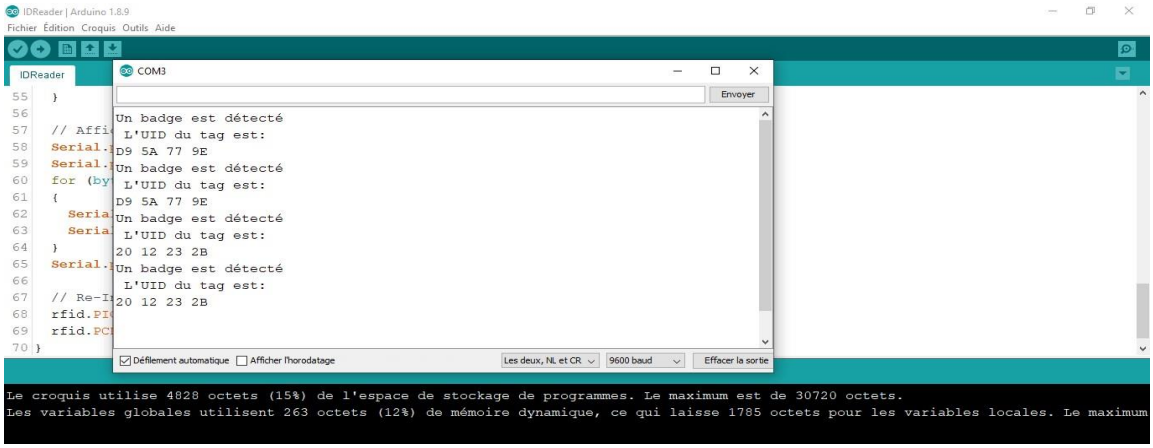
- **Programme complet**

```
 #include <SPI.h> // SPI
#include <MFRC522.h> // RFID
#define SS_PIN 10
#define RST_PIN 9
// Déclaration
MFRC522 rfid(SS_PIN, RST_PIN);
// Tableau contentent l'ID
byte nuidPICC[4];
void setup()
{
  // Init RS232
  Serial.begin(9600);
  // Init SPI bus
  SPI.begin();
  // Init MFRC522
  rfid.PCD_Init();
}

void loop()
{
  // Initialisé la boucle si aucun badge n'est présent
  if ( !rfid.PICC_IsNewCardPresent())
    return;
  // Vérifier la présence d'un nouveau badge
  if ( !rfid.PICC_ReadCardSerial())
    return;
  // Enregistrer l'ID du badge (4 octets)
  for (byte i = 0; i < 4; i++)
  {
    nuidPICC[i] = rfid.uid.uidByte[i];
  }
  // Affichage de l'ID
  Serial.println("Un badge est détecté");
  Serial.println(" L'UID du tag est:");
```

```
  for (byte i = 0; i < 4; i++)
  {
    Serial.print(nuidPICC[i], HEX);
    Serial.print(" ");
  }
  Serial.println();
  // Re-Init RFID
  rfid.PICC_HaltA(); // Halt PICC
  rfid.PCD_StopCrypto1(); // Stop encryption on PCD
}
```

4. Les ID des badges obtenus.



5. Programme du contrôle d'accès par un badge

On reprend les mêmes étapes du programme de lecture. En revanche, on garde en mémoire l'ID du badge {0x20, 0x12, 0x23, 0x2B}. Le programme compare en permanent le nouvel ID avec l'ID de base. Si les deux ID sont identiques, on déclenche l'ouverture de la porte, sinon l'alarme après trois tentatives. On fera des tests en utilisant le bon badge ayant le même ID enregistré et un nouveau badge ayant un ID différent. Ci-dessous le programme principal du projet.

Nous utiliserons une nouvelle fonction GetAccesState() qui prend en argument les deux ID (ID enregistré et le nouvel ID), puis elle retourne « 1 » si les deux ID sont identiques et 0 dans le cas contraire. Les variables à l'entrée sont deux tableaux de tailles 4 aux formats byte (la taille du code est égale à 4 octets).

6. Programme principal

```
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 10
#define RST_PIN 9

#define AccesFlag_PIN 2
#define Gate_PIN 3
#define Max_Acces 4
```

```
byte Count_acces=0;
byte CodeVerif=0;
byte Code_Acces[4]={0x20, 0x12, 0x23, 0x2B};
MFRC522 rfid(SS_PIN, RST_PIN); // Instance of the class
// Init array that will store new NUID
byte nuidPICC[4];
void setup()
{
  // Init RS232
  Serial.begin(9600);
  // Init SPI bus
  SPI.begin();
  // Init MFRC522
  rfid.PCD_Init();
  // Init LEDs
  pinMode(AccesFlag_PIN, OUTPUT);
  pinMode(Gate_PIN, OUTPUT);

  digitalWrite(AccesFlag_PIN, LOW);
  digitalWrite(Gate_PIN, LOW);
}
void loop()
{
  // Initialisé la boucle si aucun badge n'est présent
  if ( !rfid.PICC_IsNewCardPresent())
    return;
  // Vérifier la présence d'un nouveau badge
  if ( !rfid.PICC_ReadCardSerial())
    return;
  // Afffichage
  Serial.println(F("Un badge est détecté"));
  // Enregistrer l'ID du badge (4 octets)
  for (byte i = 0; i < 4; i++) {
    nuidPICC[i] = rfid.uid.uidByte[i];
  }
  // Vérification du code
  CodeVerif= GetAccesState(Code_Acces,nuidPICC);
  if (CodeVerif!=1)
  {
    Count_acces+=1;
    if(Count_acces==Max_Acces)
    {
     // Dépassement des tentatives (clignotement infinie)
     while(1)
     {
      digitalWrite(AccesFlag_PIN, HIGH);
      delay(200);
      digitalWrite(AccesFlag_PIN, LOW);
      delay(200);
```

```
      // Affichage
      Serial.println("Alarme!");
     }
    }
    else
    {
      // Affichage
      Serial.println("Code érroné");

      // Un seul clignotement: Code erroné
      digitalWrite(AccesFlag_PIN, HIGH);
      delay(1000);
      digitalWrite(AccesFlag_PIN, LOW);
    }
  }
  else
  {
    // Affichage
    Serial.println("Ouverture de la porte");

    // Ouverture de la porte & Initialisation
    digitalWrite(Gate_PIN, HIGH);
    delay(3000);
    digitalWrite(Gate_PIN, LOW);
    Count_acces=0;
  }
  // Affichage de l'identifiant
  Serial.println(" L'UID du tag est:");
  for (byte i = 0; i < 4; i++)
  {
    Serial.print(nuidPICC[i], HEX);
    Serial.print(" ");
  }
  Serial.println();
  // Re-Init RFID
  rfid.PICC_HaltA(); // Halt PICC
  rfid.PCD_StopCrypto1(); // Stop encryption on PCD
}
byte GetAccesState(byte *CodeAcces,byte *NewCode)
{
  byte StateAcces=0;
  if ((CodeAcces[0]==NewCode[0])&&(CodeAcces[1]==NewCode[1])&&
  (CodeAcces[2]==NewCode[2])&& (CodeAcces[3]==NewCode[3]))
    return StateAcces=1;
  else
    return StateAcces=0;
}
```

Application III: Développement d'une application téléphonique pour le contrôle des LED

Cette manipulation vise l'exploration et la maîtriser des concepts fondamentaux du contrôle à distance en combinant les puissantes fonctionnalités de la plateforme Blynk et du microcontrôleur ESP8266. L'utilisation de Blynk comme plateforme de développement offre une approche simplifiée et conviviale pour la création d'interfaces utilisateur interactives, tandis que l'ESP8266, en tant que microcontrôleur, constitue le pont essentiel entre le monde physique des composants électroniques et l'univers numérique des applications mobiles. Cette combinaison permet aux étudiants d'acquérir des compétences pratiques dans la programmation Arduino, de comprendre les mécanismes de la communication sans fil, et de développer une appréciation concrète des principes sous-jacents à l'Internet des Objets (IoT).

I. Objectifs de la Manipulation.

- Comprendre le fonctionnement et l'utilité Blynk
- Maîtrise de l'ESP8266
- Intégration matérielle et logicielle
- Résolution de problèmes réels

II. Matériel nécessaire :

- ESP8266 (par exemple, NodeMCU)
- LED
- Résistance (pour limiter le courant à travers la LED)
- Breadboard et fils de connexion
- Accès à Internet

III. Étapes à suivre :

1. Configuration de l'Environnement :

a. Installer les bibliothèques :

Dans l'Arduino IDE, assurez-vous d'avoir installé les bibliothèques nécessaires :

ESP8266WiFi

Blynk

Vous pouvez installer ces bibliothèques via le Gestionnaire de Bibliothèques de l'Arduino IDE.

2. Création du Projet Blynk :

a. Créer un Compte Blynk :

Créez un compte sur https://blynk.io/ si vous n'en avez pas.

b. Nouveau Projet Blynk :

Créez un nouveau projet dans l'application Blynk. (**Suivez les étapes décrites dans la vidéo suivante :** https://www.youtube.com/watch?v=DEaDy4ki9E8)
Notez le Token d'authentification généré.

3. Câblage du Matériel :

Connectez une LED à la broche de sortie de votre choix sur l'ESP8266 via une résistance.
Connectez le GND de l'ESP8266 à la cathode (le côté court) de la LED.

4. Programmation de l'ESP8266 :

Utilisez le code suivant comme base pour votre programme. Assurez-vous de remplacer VotreTokenBlynk, VotreSSID, et VotreMotDePasse par vos propres informations.

```
#define BLYNK_TEMPLATE_ID "TMPL25pLK9wjh"
#define BLYNK_TEMPLATE_NAME "test0124"
#define BLYNK_AUTH_TOKEN "8bxQ0A889BDbX4e8fwUNrMw5sZGAJVZm"
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = "8bxQ0A889BDbX4e8fwUNrMw5sZGAJVZm";
char ssid[] = "iPhone de Abdelkader";
char pass[] = "12345678900";
int pinLED = D4;  // Remplacez D4 par le numéro de la broche que vous
utilisez pour la LED
void setup() {
  Blynk.begin(auth, ssid, pass);
  pinMode(pinLED, OUTPUT);
}
BLYNK_WRITE(V0) {  // Widget Bouton sur l'application Blynk
  int value = param.asInt();
  digitalWrite(pinLED, value);
}
void loop() {
  Blynk.run();
}
```

5. Téléversement du Code :

Sélectionnez la carte ESP8266 et le port dans l'Arduino IDE.
Téléversez le code sur votre ESP8266 (NodeMcu 1.0).

6. Configuration de l'Application Blynk :

Ajoutez un widget "Bouton" (Button) sur l'écran dans l'application Blynk.
Associez le widget à la broche virtuelle V1.

7. Test et Contrôle à Distance :

Ouvrez l'application Blynk sur votre smartphone.
Appuyez sur le bouton ajouté à l'interface pour allumer/éteindre la LED à distance.

IV. Travaux demandés :

Manip 1 :

Concevoir un système permettant la gestion à distance de trois LEDs individuelles en utilisant la plateforme Blynk et le microcontrôleur ESP8266.

Manip 2 :

Concevoir un système permettant l'affichage de l'état réel d'un bouton poussoir sur votre téléphone en utilisant la plateforme Blynk et le microcontrôleur ESP8266.

Manip 3 :

Concevoir un système permettant la gestion à distance d'un capteur de température et d'humidité et qui déclenche une alarme si la température dépasse 26°C en utilisant la plateforme Blynk et le microcontrôleur ESP8266.

Manip 4 :

Concevoir un système permettant la gestion à distance d'un capteur de Lumière LDR en utilisant la plateforme Blynk et le microcontrôleur ESP8266.

Manip 5 :

Concevoir un système permettant la gestion à distance d'un capteur ultrasonic en utilisant la plateforme Blynk et le microcontrôleur ESP8266.

Manip 6 :

Concevoir un système permettant la gestion à distance d'un capteur d'humidité de sol en utilisant la plateforme Blynk et le microcontrôleur ESP8266.

Manip 7 :

Concevoir un système permettant la gestion d'un réseau de capteurs à distance de votre choix en utilisant la plateforme Blynk et le microcontrôleur ESP8266.