# Conception d’un système IoT : Gestion des badges

# Fonctionnement de la plateforme de la gestion des badges


Carte / Badge contenant les identifiants

On approche le badge vers le lecteur RFID

Lecteur RFID MFRC522

Envoie les données de la carte (ID de la carte) vers l'Arduino

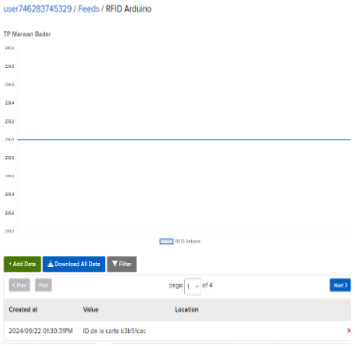
Arduino Uno

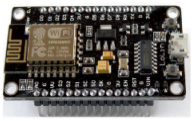
Communication par les pins D4 / D3 et envoi les données de la carte lue vers l'ESP8266

Module Wi-Fi ESP8266 (NodeMCU)

Communication avec l'API Adafruit par WiFi et envoi les données de la carte vers la base de données

Base de données Adafruit.io



# Justification des choix matériels

## Pourquoi a-t-on choisi une carte / un badge avec le protocole NFC?

Une carte / un badge avec le protocole NFC a été choisi car il est simple et facile de l'utiliser et de l'intégrer avec une interaction sans contact, à courte distance. Les communications NFC peuvent être intégrées avec une sécurité comme le chiffrement et l'authentification. Le NFC est une technologie passive et ne nécessite pas une source d'alimentation comme une batterie, il est utilisé dans plusieurs services et domaines. Comme les autres matériels, il a un coût abordable et est donc largement accessible.

## Pourquoi a-t-on choisi un lecteur RFID MFRC522 ?

Le lecteur RFID MFRC522 est choisi pour des projets de contrôle d'accès car il a un faible coût et donc un bon choix pour un budget limité. Il est facile à intégrer avec l'Arduino Uno et possède des librairies / bibliothèques documentées. Il a une basse consommation d'énergie. Sa fréquence est de 13,56 MHz et est donc compatible avec plusieurs cartes et badges NFC.

## Pourquoi a-t-on choisi un Arduino Uno ?

L'Arduino Uno est choisi car il a une facilité d'utilisation, il a un large nombres de broches suffisant d'E/S (entrée/sortie), il est compatible avec plusieurs modules tel que le lecteur RFID ou le module WIFI

ESP8266, il est fiable et stable et a un coût faible pour les budgets limités.

## Pourquoi a-t-on choisi un kit Arduino entier? (Freenove Project Board)

Le kit Freenove Project Board a été choisi car il a des composants supplémentaires comme les LED, les capteurs et peuvent être utilisés pour tester d'autres aspects du projet, il inclut une documentation riche et accessible par tous et comporte les broches E/S (entrée / sortie) permettant de communiquer avec les autres modules présent sur l'interface comme le transfert de données de l'Arduino Uno vers l'ESP8266.

## Pourquoi a-t-on choisi l'ESP8266 à la place d'autres modules Wi-Fi?

L'ESP8266 est choisi à la place d'autres modules Wi-Fi car il a un coût faible, il est simple à intégrer avec l'Arduino Uno, il a une taille compacte et une consommation d'énergie raisonnable.

## Pourquoi ne pas utiliser un Raspberry PI ?

Le Raspberry Pi est un mini-ordinateur complet avec un système d'exploitation. Pour des tâches simples de lecture de badge RFID et de connexion à un réseau Wi-Fi, il est souvent surdimensionné. Il ajouterait une couche de complexité inutile (OS, configuration réseau).

# CODES SOURCE POUR CHAQUE MODULE

## Code source à téléverser dans l'Arduino Uno

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


#define RST_PIN 9  // Broche de réinitialisation
#define SS_PIN 10  // Broche de sélection SPI
const int rouge = 7;
const int vert = 6;
const int bleu = 5;
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Adresse I2C du LCD 16x2
MFRC522 mfrc522(SS_PIN, RST_PIN);    // Création de l'instance
MFRC522

SoftwareSerial EspSerial(2, 3);  // (TX 2 , RX 3)

// UID connu stocké localement sous forme décimale (exemple:
12345678)
unsigned long WhitListUID[] = { 3014991020, 1234567890, 9876543210 };
// Ajoute ici les UID que tu veux
int NombreUID = sizeof(WhitListUID) / sizeof(WhitListUID[0]);
// Taille du tableau d'UID
//****************************************************************
**********************//
void setup() {
  //LED
  pinMode(rouge, OUTPUT);
  pinMode(vert, OUTPUT);
  pinMode(bleu, OUTPUT);
  etd();

  Serial.begin(9600);  // Initialisation de la communication série
  EspSerial.begin(9600); // Initialisation de la communication avec
l'ESP
  SPI.begin();         // Initialisation du bus SPI
  mfrc522.PCD_Init();  // Initialisation du module MFRC522
  // Initialisation de l'écran LCD
  if (!i2CAddrTest(0x27)) {
    lcd = LiquidCrystal_I2C(0x3F, 16, 2);  // Changement de l'adresse
I2C si nécessaire
  }
  // Initialisation de l'écran LCD en affichant "Scannez une carte…"
lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Scannez une ");
  lcd.setCursor(0, 1);
```

```
  lcd.print("carte...");
  EspSerial.println("Arduino On");
}
//*****************************************************************
**********************//
void loop() {
  // Réinitialisation si aucune carte n'est présente
  if (!mfrc522.PICC_IsNewCardPresent()) {
    return;
  }
  // Sélection de la carte
  if (!mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  // Effacer l'écran et afficher "Card UID:"
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Scannez une ");
  lcd.setCursor(0, 1);
  lcd.print("carte...");
  // Convertir l'UID en une seule valeur décimale
  unsigned long scanneUID = 0;
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    scanneUID = scanneUID * 256 + mfrc522.uid.uidByte[i];  //
Conversion des octets de l'UID en nombre décimal
  }
  // Afficher l'UID en décimal sur le LCD
  lcd.setCursor(0, 1);
  lcd.print(scanneUID);
  // Comparer l'UID scanné avec l'UID stocké localement
  if (isUIDKnown(scanneUID)) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Autorise");
    Serial.print("Autorise UID = ");
    Serial.println(scanneUID);

    alumVer();

    EspSerial.println(scanneUID);

  } else {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Refuse");
    Serial.print("Refuse UID = ");
    Serial.println(scanneUID);

    alumRoug();

    EspSerial.println(scanneUID);
  }
  // Pause pour laisser le temps de lire l'UID et l'état
  delay(2000);
  // Remettre l'affichage à "Scan a card..."
  lcd.clear();
  lcd.setCursor(0, 0);
```

```
  lcd.print("Scannez une ");
  lcd.setCursor(0, 1);
  lcd.print("carte...");
  // Arrêter la communication avec la carte RFID
  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();
  etd();
}
//*****************************************************************
**********************//
// Fonction pour tester si une adresse I2C est correcte
bool i2CAddrTest(uint8_t addr) {
  Wire.begin();
  Wire.beginTransmission(addr);
  return (Wire.endTransmission() == 0);
}
void alumRoug() {
  digitalWrite(rouge, LOW);
  digitalWrite(vert, HIGH);
  digitalWrite(bleu, HIGH);
}
void alumVer() {
  digitalWrite(rouge, HIGH);
  digitalWrite(vert, LOW);
  digitalWrite(bleu, HIGH);
}
void etd() {
  digitalWrite(rouge, HIGH);
  digitalWrite(vert, HIGH);
  digitalWrite(bleu, HIGH);
}

bool isUIDKnown(unsigned long scanneUID) {
  for (int i = 0; i < NombreUID; i++) {
    if (scanneUID == WhitListUID[i]) {
      return true;  // L'UID scanné est dans la liste
    }
  }
  return false;  // L'UID scanné n'est pas dans la liste
}
```

# Code source à téléverser dans l'ESP8266

```
#include <ESP8266WiFi.h>           // Bibliothèque pour la connexion
WiFi avec l'ESP8266
#include "Adafruit_MQTT.h"         // Bibliothèque pour gérer le
protocole MQTT
#include "Adafruit_MQTT_Client.h"  // Client MQTT pour la connexion à
Adafruit IO

// Paramètres du réseau WiFi
#define WLAN_SSID "T3ST"      // Le nom du réseau WiFi (SSID)
#define WLAN_PASS "12345678"  // Le mot de passe du réseau WiFi

// Paramètres Adafruit IO
#define AIO_SERVER "io.adafruit.com"               // Adresse du
serveur Adafruit IO (MQTT)
#define AIO_SERVERPORT 1883                        // Port du
serveur MQTT (1883 est le port par défaut pour MQTT)
#define AIO_USERNAME "user746283745329"            // Votre nom
d'utilisateur Adafruit IO
#define AIO_KEY "aio_drYt81nLZP5lMJPm2RYYsjbo6TKb"  // Votre clé API
Adafruit IO

int ledPin = 0;  // La broche où est connectée la LED (D1 = GPIO0)

// Création d'un objet client WiFi pour l'ESP8266 afin de se
connecter au serveur MQTT
WiFiClient client;

// Création du client MQTT en utilisant le client WiFi, le serveur et
les identifiants
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT,
AIO_USERNAME, AIO_KEY);

// Création d'un objet pour publier les données sur Adafruit IO (sur
le feed "test")
Adafruit_MQTT_Publish Attendance = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/rfid-arduino");

// Variable pour stocker un identifiant lu depuis le port série
char ID;

void setup() {
  pinMode(ledPin, OUTPUT);    // Configuration de la broche de la LED
comme sortie
  digitalWrite(ledPin, LOW);  // Éteindre la LED au démarrage
  Serial.begin(9600);         // Initialisation de la communication
série à 9600 bauds
  delay(1000);                // Petit délai pour permettre
l'initialisation

  Serial.println();              // Saut de ligne pour le formatage
dans le moniteur série
  delay(10);                     // Délai court avant de commencer la
connexion WiFi
  Serial.print("Connexion à ");  // Affiche "Connexion à " dans le
```

```
moniteur série
  Serial.println(WLAN_SSID);     // Affiche le SSID du réseau WiFi

  // Tentative de connexion au réseau WiFi
  WiFi.begin(WLAN_SSID, WLAN_PASS);        // Se connecte au réseau
avec le SSID et mot de passe fournis
  while (WiFi.status() != WL_CONNECTED) {  // Attendre jusqu'à ce que
la connexion soit établie
    delay(500);                            // Pause de 500ms
    Serial.print(".");                     // Affiche un point pour
indiquer que la connexion est en cours
  }

  digitalWrite(ledPin, HIGH);                 // Allume la LED une
fois connecté au WiFi
  Serial.println();                           // Saut de ligne pour
le formatage
  Serial.println("Wi-Fi Connecté Succès !");  // Message indiquant
que la connexion WiFi a réussi
  Serial.print("NodeMCU IP Address : ");      // Affiche "NodeMCU IP
Address :"
  Serial.println(WiFi.localIP());             // Affiche l'adresse IP
locale attribuée par le routeur

  // Connexion à Adafruit IO via MQTT
  connect();
}

void connect() {
  Serial.println("Connecting to Adafruit IO... ");  // Affiche un
message pour indiquer la tentative de connexion à Adafruit IO

  int8_t ret;                              // Variable pour stocker le
code de retour de la tentative de connexion
  while ((ret = mqtt.connect()) != 0) {  // Boucle jusqu'à ce que la
connexion réussisse
    // Si la connexion échoue, un message d'erreur est affiché selon
le code de retour
    switch (ret) {
      case 1: Serial.println("Wrong protocol"); break;       //
Protocole incorrect
      case 2: Serial.println("ID rejected"); break;          //
Identifiant rejeté
      case 3: Serial.println("Server unavailable"); break;   //
Serveur non disponible
      case 4: Serial.println("Bad user/pass"); break;        //
Mauvais identifiant ou mot de passe
      case 5: Serial.println("Not authorized"); break;       // Non
autorisé
      case 6: Serial.println("Failed to subscribe"); break;  // Échec
de l'abonnement
      default: Serial.println("Connection failed"); break;   // Échec
de connexion
    }

    if (ret >= 0) mqtt.disconnect();           // Si la connexion
échoue, déconnexion
```

```
    Serial.println("Retrying connection...");  // Affiche "Retrying
connection..." pour indiquer une nouvelle tentative
    delay(5000);                               // Pause de 5 secondes
avant de retenter la connexion
  }

  Serial.println("Adafruit IO Connected!");  // Indique que la
connexion à Adafruit IO a réussi
}

void loop() {
  // Envoie des "pings" réguliers à Adafruit IO pour s'assurer que la
connexion MQTT reste active
  if (!mqtt.ping(3)) {      // Si trois tentatives de ping échouent
    if (!mqtt.connected())  // Si le client MQTT est déconnecté
      connect();            // Reconnecte à Adafruit IO
  }

  // Vérifie s'il y a des données disponibles sur le port série
  if (Serial.available()) {  // Si des données sont disponibles
    char a = Serial.read();  // Lit un caractère depuis le port série
    ID = a;                  // Stocke le caractère lu dans la
variable `ID`
    // Tente de publier le caractère `ID` sur le feed MQTT "test"
    if (!Attendance.publish(ID)) {  // Si la publication échoue
      Serial.println("Failed");     // Affiche "Failed"
    } else {                        // Si la publication réussit
      Serial.println("Sent!");      // Affiche "Sent!" pour indiquer
que les données ont été envoyées
    }
  }
}
```