# Système RFID :

# Présentation et utilisation simple
# Avec un client ESP8266
# et un serveur Node-RED

DeltaLab

Espace Maison Milon
2 Place E.Colongin
84600 Grillon

deltalabprototype.fr

## Qu’est-ce que DeltaLab ?

DeltaLab est une association ‘loi 1901’ d’intérêt général, dont l’objectif est la création d’un espace dédié à l’innovation, à la production numérique au prototypage et à l’«expression artistique».

Le principe fondateur de l’associtation est **d’apprendre à faire soi-même, pour passer de l’dée à l’objet.**

Deltalab se spécialise en **Objets Connectés**, et est en train de créer un vaste «écosystème digital» entre Drôme et Vaucluse, pour répondre à des besoins non-couverts, mettre à disposition ressources et équipements pour usage professionnels et instaurer des partenariats avec les autres structures et initiatives numériques existantes.

Deltalab est aussi un **FabLab** (*Fabrication Laboratory / Laboratoire de Fabrication* ), un tiers-lieu de type makerspace ou se trouve un atelier qui dispose de machines de fabrication comme des Imprimantes  3D ou des découpeuses Laser.

Deltalab se veut ouvert à tous publics : étudiants, professionnels, associations, inventeurs, designers,artistes, ...

---

## Contexte de cette Documentation

Ce projet est une introduction à la sécrurisation de lieux ou machines via badges RFID. Les cartes / badges RFID sont des badges magnétiques contenant un ID, qui permet de déterminer si ils activent un système ou pas.

DeltaLab compte déployer ce système dans ses locaux de la Maison Milon, cette documentation en est le commencement. Dans le futur, DeltaLab pourrait aider les personnes à installer leur propres systèmes, en fournissant lesd ocumentations et les codes nécéssaires.

# Table des matières

# I - Introduction

Ce projet a pour but de permettre l'utilisation d'un lecteur RFID pour lire des cartes ou badges magnétiques, pour activer un relais. Dans une utilisation réelle, le relais pourrait être relié à une serrure ou une machine à protéger, à une LED, à un buzzer,...

Le projet se décompose en 3 parties distinctes : le lecteur RFID et l'esp8266 associée , le serveur Node-RED et la carte à relais et l'esp8266 associée. Le lecteur RFID ne tient pas en compte le type de carte ou le nombre de relais présents. Le serveur et le client coté relais peuvent être adaptés selon la situation.

Les fonctionnalités proposées par ce projet sont les suivantes :

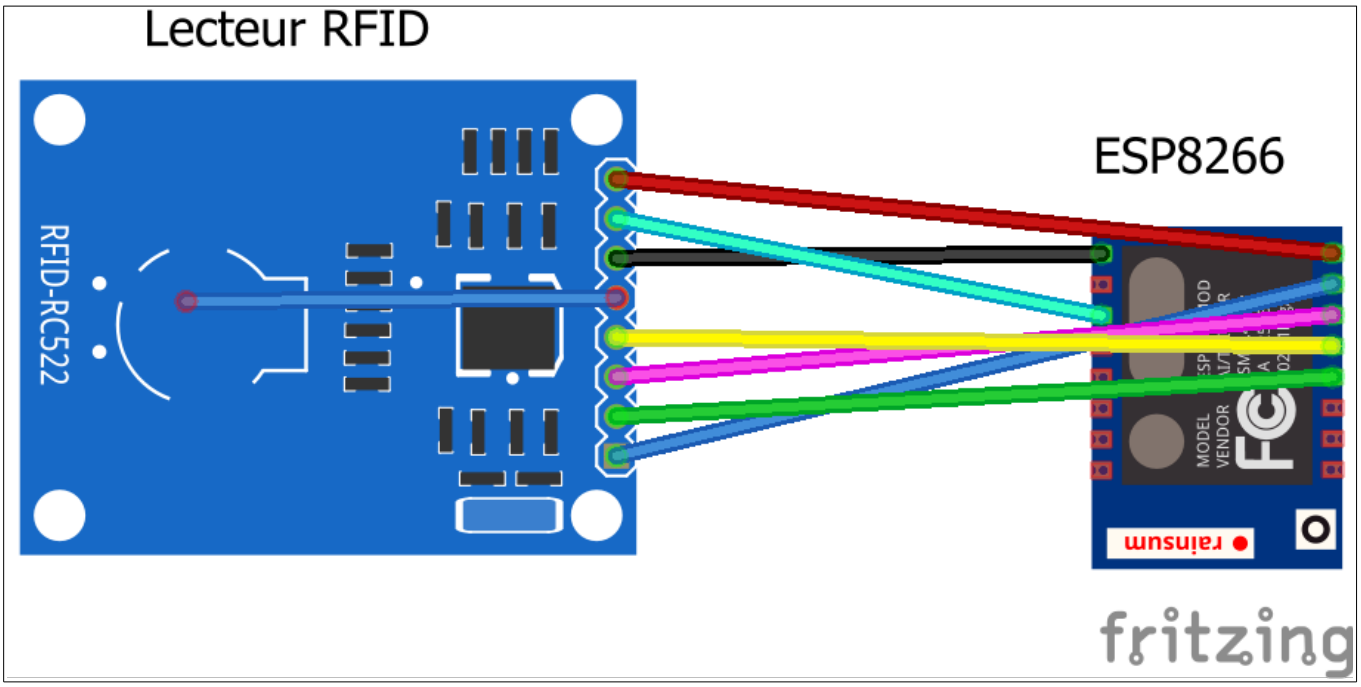
- Utilisation du lecteur et affichage du badge / de la carte utilisées
- Activation d'un relais grâce à l'authenitifcation en "dur" dans le serveur
- Activation d'un relais grâce à l'authenitfication dans une base de donées

**Logiciels :**

- Arduino IDE **:** téléchargeable à : https://www.arduino.cc/en/main/software
- Librairie ESP8266WiFi : à télécharger à https://github.com/esp8266/Arduino Puis à ajouter à Arduino (dézipper, puis dans Arduino : *Croquis > Inclure une Bibliothèque > Ajouter une Bibliothèque .zip* , récupérez la librairie dans le dossier libraries )
- Node-RED : à deltalab , entrez l'@ip ***192.168.1.45:1880*** dans un navigateur
- ESP8266 : Fichier > Préférences > Gestionnaire de cartes. Entrez l'URL http://arduino.esp8266.com/stable/package_esp8266com_index.json **Puis Croquis>Carte>Gestionnaire de cartes , rechercher ESP8266**

# II - Présentation du Circuit

- **Plan du circuit**




**Connexions :**

| Port - Lecteur RFID | Port - ESP8266 | Couleur |
|---|---|---|
| 3,3V | 3V3 | Rouge |
| RST | D3 | Cyan |
| GND | GND | Noir |
| MISO | D6 | Jaune |
| MOSI | D7 | Blanc |
| SCK | D5 | Vert |
| SDA | D8 | Bleu |

## Explication :

Le lecteur RFID est relié à la carte esp8266. Cette dernière y aura accès via le bus SPI. Elle communique avec le serveur Node-RED en wifi via MQTT.

Le serveur MQTT envoie des instructions aux relais selon les données reçues.
Les relais s'activent ou se désactivent , actionnant ou désactionnant les appareils qui peuvent y être reliés.

Les relais peuvent être :

- **indépendants** , sur leur propre carte qui possède sa propre ESP32. Ils n'ont aucun lien avec le lecteur et un code doit leur être téléversé à eux aussi. Pour ce faire, il peut être nécéssaire d'utiliser un adaptateur esp-to-usb , sur lequel l'ESP32 de la carte des relais vient s'emboiter. Ils ont aussi besoin d'une alimentation propre.

- **Branchés à l'ESP32 du lecteur**, soit par des câbles, soit en s'emboitant dessous. C'est l'ESP32 du lecteur qui gère également les relais, et doit donc contenir un code regroupant les codes du lecteur et des relais. Les relais utiliseront aussi l'alimentation de l'ESP32.

# III - Codes Arduino

## 1 - Coté Lecteur

**Librairies utilisées :**

- **SPI - version 1.0**
- **MFRC522 - version 1.4.6**
- **PubSubClient - version 2.7.0**
- **ESP8266WiFi- version 1.0**

**Code :** *(code en rouge : code propre au cas où le relais est branché au lecteur)*

```
#include <SPI.h>
#include <MFRC522.h>
#include "PubSubClient.h"
#include <ESP8266WiFi.h>
#include <WiFiClient.h>

#define ClientID "rdc_lab" // id du lecteur
#define SS_PIN D8
#define RST_PIN D3
#define pin D2

MFRC522 mfrc522(SS_PIN, RST_PIN);
WiFiClient cli;
PubSubClient mqttClient(cli);
const char* ssid = "DeltaLab-Public";          // SSID du point wifi
const char* password = "Milon!Lab";            // mot de passe du point wifi
const char* mqttServer = "192.168.1.45";       // serveur mqtt
const int mqttPort = 1883;                     // port du serveur à utiliser - défaut : 1883


/* Reconnexion au serveur mqtt si déconnecté */
boolean reconnect() {
  if (mqttClient.connect(ClientID)) {
    mqttClient.subscribe("ConnexionRFID");
    mqttClient.loop();
  }
  return mqttClient.connected();
}


/*  Réponse du serveur : carte acceptée ou refusée (uniquement si relais local ) */
void callback(char* topic, byte* payload, unsigned int length) {
   String cmd = String((char*)payload);
    if (cmd.substring(0,5).equals("l1_on")) {
         digitalWrite(pin , HIGH);
    }
    if (cmd.length() >= 6 && cmd.substring(0,6).equals("l1_off")) {
         digitalWrite(pin ,  LOW);
    }     }
```

```
/*
 * Connexion au WiFi et initialisation du lecteur et du MQTT
 */
void setup() {

  Serial.begin(115200);    // Initiate a serial communication
  WiFi.hostname(ClientID);
  WiFi.begin(ssid,password);
  delay(1500);

  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
  }

  mqttClient.setCallback(callback);
  mqttClient.setServer(mqttServer,mqttPort);

  SPI.begin();          // Initiate  SPI bus
  mfrc522.PCD_Init();   // Initiate MFRC522

  Serial.println("Placez votre carte/badge devant le lecteur");
  Serial.println();
}


/*
 * Lecture en boucle du lecteur , formatage des données lues et envoi au serveur
 */
void loop() {

  if ( ! mfrc522.PICC_IsNewCardPresent()) { mqttClient.loop();  return; }
  if ( ! mfrc522.PICC_ReadCardSerial()) { mqttClient.loop(); return; }

  Serial.print("UID tag :");
  String content= "";
  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++) {
     content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
     content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  Serial.println();
  content.toUpperCase();
  String card_id = content.substring(1);
  Envoi(card_id);
  delay(1500);
  if (!mqttClient.connected()) {
        reconnect();
  } else{
        mqttClient.loop();
  }
}
```

```
/*
 * Envoi des données de la carte vers le serveur sous la forme
 * id_de_l'esp/id_de_la_carte , traité ensuite par le serveur
 */
void Envoi(String card_id){

  String contenu = ClientID;
  contenu += "/";
  contenu += card_id;
  char payload[19];
  unsigned int len = contenu.length()+1;
  contenu.toCharArray(payload,len);

  if(mqttClient.connect(ClientID)) {
        mqttClient.publish("connexionRFID",payload);
        mqttClient.subscribe(ClientID);
        mqttClient.loop();
  }
}
```

## 2 - Coté relais (Si les relais sont distants)

**Librairies utilisées :**

- **PubSubClient - version 2.7.0**
- **esp8266WiFi - version 1.0**

**Code :**

```
#include "PubSubClient.h"
#include <ESP8266WiFi.h>

#define CLIENT_ID  "rdc_lab"      // ID de la carte de relais , à changer pour chaque carte

const char* ssid = "DeltaLab-Public";  // SSID du  point wifi - à changer par le SSID personnel
const char* password =  "Milon!Lab"; // Mot de passe du point Wifi - à changer
const char* mqttServer = "192.168.1.45"; //serveur MQTT
const int mqttPort = 1883;              // port du serveur à utiliser - par défaut 1883
char* topic = CLIENT_ID; // Le topic MQTT qui sera utilisé - permet d'isoler chaque relais
int timer = 0; // timer de 3 minutes pour l'envoi de messages

WiFiClient ethClient;
PubSubClient mqttClient(ethClient);

byte MsgRL1On[] = {0xA0, 0x01, 0x01, 0xA2};   // ouvre le relais #1 , seul relais utilisé
byte MsgRL1Off[] = {0xA0, 0x01, 0x00, 0xA1};  // ferme le relais #1
```

```
/* Reconnexion à MQTT si connexion perdue */
boolean reconnect() {
  if (mqttClient.connect(CLIENT_ID)) {
    mqttClient.publish("relais", CLIENT_ID);
    //mqttClient.subscribe(topic);
    mqttClient.subscribe("relays");
    mqttClient.loop();
  }
  return mqttClient.connected();
}


/*
*  Traitement des messages reçus :
*  écrit le bon message en série selon le msg reçu
*/
void callback(char* topic, byte* payload, unsigned int length) {

  String cmd = String((char*)payload);
  Serial.println("cmd : " + cmd.substring(0, length));

  if (topic != "re:relays" && cmd.substring(0, length).equals("l1_on")) {
    mqttClient.publish("re:relays" , "l1_on");
    Serial.write(MsgRL1On , sizeof(MsgRL1On));
  }
  if (cmd.substring(0, length).equals("l1_off")) {
    mqttClient.publish("re:relays" , "l1_off");
    Serial.write(MsgRL1Off , sizeof(MsgRL1Off));
  }




/*  SetUp  */
void setup() {

  WiFi.hostname(CLIENT_ID);
  WiFi.begin(ssid, password);
  Serial.begin(115200);
  delay(1500);

  while (WiFi.status() != WL_CONNECTED) { delay(500); }

  mqttClient.setCallback(callback);
  mqttClient.setServer(mqttServer, mqttPort);
}
```

```
/*
 *   Boucle : si connexion perdue , reconnecter.
 *  Attente de messages mqtt depuis nodeRed
 *   Envoie un message mqtt à nodeRed toutes les 3min pour rappeler que ce relais est vivant
*/
void loop() {

  if (!mqttClient.connected()) {
    reconnect();
  }else{
    mqttClient.loop();
  }

  if ( timer == 600 ) {
    mqttClient.publish("relais" , CLIENT_ID);
    timer = 0;
  } else {
    timer ++ ;
    delay(100);
  }
}
```

Ici, on utilise un relais d'une carte en contenant plusieurs. Il est bien sur possible d'activer les autres selon les mêmes ou d'autres conditions, gérées par le serveur Node-RED

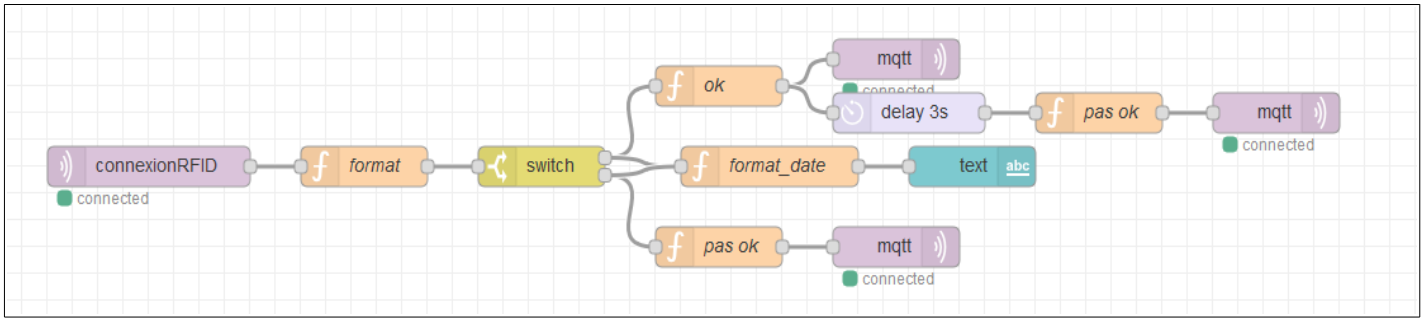
Dans arduino, pour choisir la bonne carte , allez dans **Outils > Type de carte**
On utilise ici une Lolin(Wemos) D1 mini pro pour le lecteur, et une ESP8266 générique pour les relais.Vérifiez le Port et la vitesse d'écriture

Pour téléverser le code sur la carte, allez dans **croquis > Téléverser** , ou cliquez sur la flèche.

# IV - Serveur Node-RED

## 1 - Authentification simple dans le serveur

### Flow :




### Explications :

- A la lecture d'une carte, le lecteur envoie les données à node-red via MQTT.
- Le serveur reçoit le message (***node ConnexionRFID - MQTT in***)
- Le serveur extrait les données du payload du message (***node format - function***). Le payload reçu est de la forme "id_du_lecteur/id_de_la_carte".
- Le message passe dans la node switch. Si l'id de la carte correspond à celui indiqué dans la node, l'authentifiaction est réussie.
- Si l'authentification est réussie , le serveur envoie un message MQTT aux relais concernés , contenant le message permettant d'actionner un relais. Le message est formaté dans la ***node "ok" (function)*** avant d'être envoyé en MQTT dans la ***node "mqtt" (MQTT out)***. Après un certain délais , le serveur envoie le message inverse au relais, pour le refermer (**nodes delay et Relays**), pour simuler de la "sécurité". (Après authentification la serrure se revérouille au bout de 30s même si elle n'est pas touchée)
- Le serveur affiche sur le dashboard que cette carte a été authentifiée avec succès à ce lecteur à cette date. Le message est formaté dans la node ***"format_date" (function)*** , puis est affiché dans une node Text.
- Si l'authentification a échoué, le serveur envoie un message au relais pour le décativer (le fermer) pour la sécurité.

On peut associer à ce Flow celui du projet "Relais", qui permet de savoir si la carte des relais est opérationnelle dans le dashboard. Il est juste copiable sans avoir besoin de changements.

**Pour le refaire :**

1. **MQTT in**
   - broker :
     - serveur : localhost
     - port : 1883
   - topic : connexionRFID

2. **Function format**

```
var id = [];
id = msg.payload.split('/')

return{
 topic:msg.topic,
 payload:{
 id:id[0],
 carte:id[1]
 }
}
```

3. **switch** : condition sur le champs msg.payload.carte. Vous pouvez en ajouter autant que nécéssaire. Creéz une condition 'otherwise' (sinon) à la fin

4. **function ok /pas ok**
   reliez toutes les cartes autorisées à la function ok, et celles non autorisées et la condition otherwise à la function pas ok.

```
return {
        topic:msg.lecteur,
        payload:"l1_on" //function ok - "l1_off" pour la function pas ok
}
```

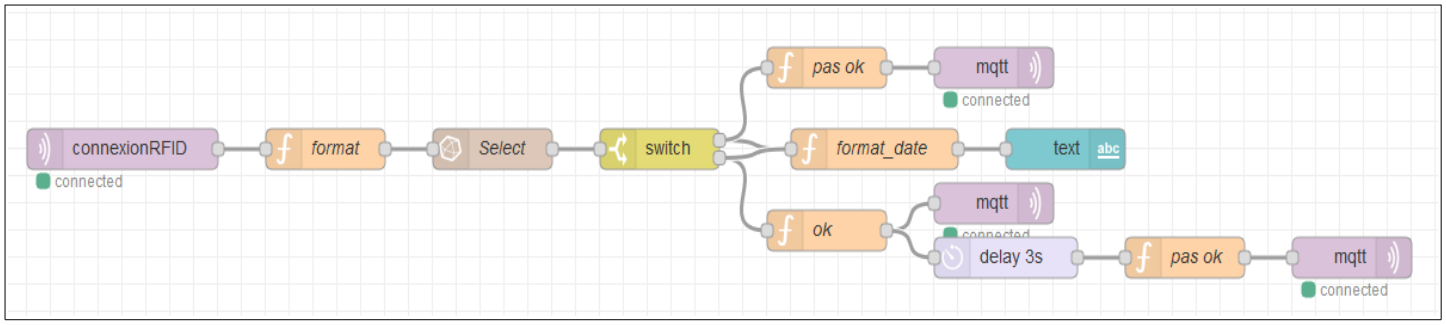
5. **MQTT out**
   - broker : localhost : 1883
   - topic : laisser vide (msg.topic prend le dessus)

6. **function format_date / text**
   La function renvoie la carte , le lieu et la date et l'heure de connexion pour les afficher dans la node text.

```
Var Dat = new Date(Date.now());
Var datheur = Dat.getDate()+"/"+Dat.getMonth()+"/"+Dat.getFullYear()+" - "+
Dat.getHours()+":"+Dat.getMinutes()+":"+Dat.getSeconds();
return {
  payload : "carte : "+msg.payload.carte+" - Lieu : "+msg.lecteur+" -  Date : "
            + datheur
};
```

## 2 - Authentification via une Base de Données

### Flow :




### Explications :

- A la lecture d'une carte, le lecteur envoie les données à node-red via MQTT.
- Le serveur reçoit le message (***node ConnexionRFID - MQTT in***)
- Le serveur extrait les données du payload du message (***node format - function***). Le payload reçu est de la forme "id_du_lecteur/id_de_la_carte".
- Le serveur cherche dans la base de données si la permission est valide. La node Select (postgrestor) récupère un objet vide si la permission n'existe pas, ou un objet comportant au moins un élément si elle existe.
- La node switch teste si le message est vide. Si oui, la permission n'existe pas et l'authentification est ratée, si non la permission existe et l'authentification est donc réussie.
- Si l'authentification est réussie , le serveur envoie un message MQTT aux relais concernés , contenant le message permettant d'actionner un relais. Le message est formaté dans la ***node "ok" (function)*** avant d'être envoyé en MQTT dans la ***node "mqtt" (MQTT out)***. Après un certain délais , le serveur envoie le message inverse au relais, pour le refermer (**nodes delay et Relays**), pour simuler de la "sécurité". (Après authentification la serrure se revérouille au bout de 30s même si elle n'est pas touchée)
- Le serveur affiche sur le dashboard que cette carte a été authentifiée avec succès à ce lecteur à cette date. Le message est formaté dans la node ***"format_date" (function)*** , puis est affiché dans une node Text.
- Si l'authentification a échoué, le serveur envoie un message au relais pour le décativer (le fermer) pour la sécurité.

**Pour le refaire :**

Suivez le même protocole que pour le premier flow, avec comme différences :

- **Node Select** : node de Base de donnée ( postgrestor )
  - serveur
    - host : 127.0.0.1 (localhost)
    - port : 5432
    - database : votre base de donnée
    - user : votre id
    - password votre mot de passe

  - requête

    *SELECT **
    *FROM permissions*
    *WHERE carte = '{{ msg.payload.carte }}'*
    *AND lieu = '{{ msg.payload.id }}';*

- **Node switch :** condition sur le champs msg.payload.rows[0].permission :
  - true : permission accordée – relier à *ok*
  - false : permission refusée – relier à *pas ok*
  - otherwise : permission innexistante - reliez à *pas ok*