Documentation :

Utilisation de relays arduino

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

Cette documentation est une introduction à l’utilisation de relais et à leur pilotage depuis Node-RED. Ces relais peuvent être utiles dans nombre de projets, permettant d’actionner divers circuits et mécanismes selon le serveur.
Exemple : relier les relais à des serrures magnétiques pour les ouvrir ou à des machines pour les protéger

Table des matières

I - But du “Projet”

Ce projet à pour but de permettre la manipulation de relais depuis un serveur NodeRED via une ESP8266 et MQTT.

Le projet se décompose en deux parties distinctes : le serveur NodeRED et le client Arduino sur l’ESP8266. Le serveur et le client sont ici paramétrés pour fonctionner avec une carte comportant 4 relais , mais ils peuvent être adaptés pour fonctionner avec plus ou moins de relais.

Les fonctionnalités proposées par ce “projet” sont les suivantes :

- Controle des relais depuis une interface NodeRED
- Mise à jour automatique de l’état (on / off) de chaque relais dans la même interface
- Mise à jour automatique toutes les minutes de l’état global de la carte (active / inactive) dans la même interface.

Materiel utilisé ici :

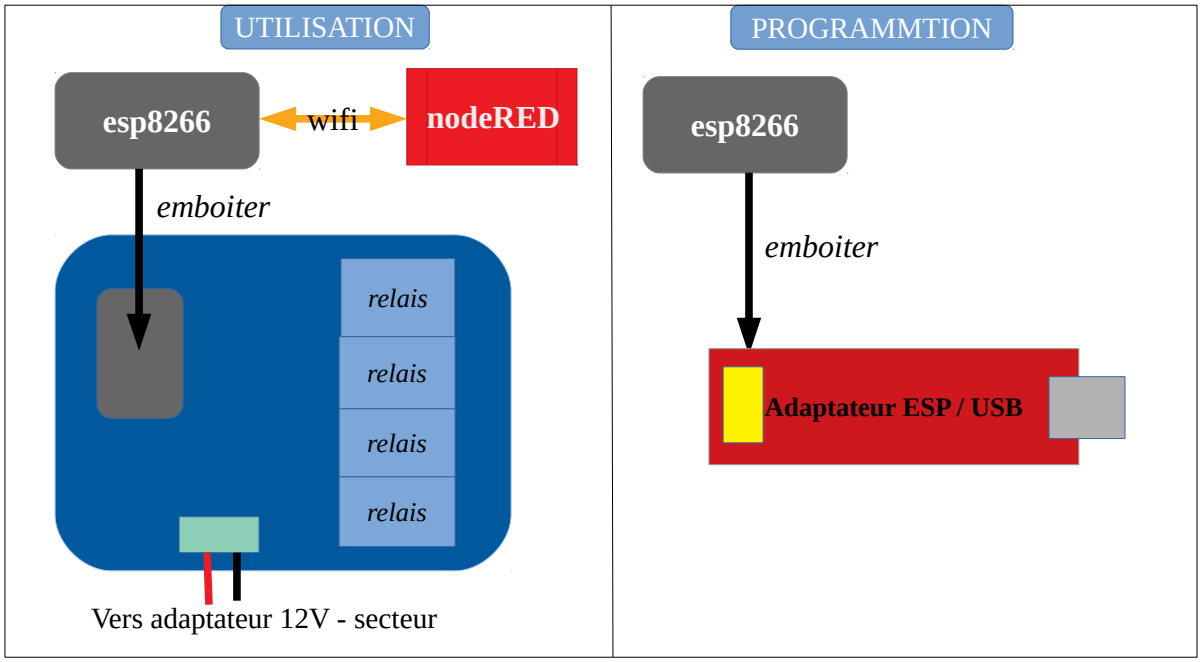
- carte de relais : une *ESP8266 Relay X4* de LC Technology , qui vient avec sa propre mini-esp8266
- adaptateur : un *Open-Smart ESP to USB adapter.*

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 (*Croquis > Inclure une Bibliothèque > Ajouter une Bibliothèque .zip*)
- 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 - Circuit

Plan du circuit :



Explications :

- La carte à relais utilisée contient 4 relais. Elle fonctionne sous 12V, et doit donc être branchée au secteur. Elle se sert du RS232 comme connecteurs
- Elle est pilotée par une ESP8266 simplifiée qui s'emboite sur elle grâce à 8 pattes semblables.
- L'ESP8266 contient le code , elle doit être retirée de la carte à relais et emboitée dans un adaptateur, lequel se branche en USB sur un PC.
- L'adaptateur possède un interrupteur, qui permet de changer de mode entre le mode de programmation – position "prog" - qui permet de téléverser de nouveaux codes sur l'ESP8266 , et le mode d'utilisation – position "vart" - qui permet d'utiliser l'ESP8266 tel quel en lançant le code dessus. (NB : tous les adaptateurs ne possèdent pas cet interrupteur)
- L'ESP doit être ré-emboitée dans la carte à relais pour utilisation. Il est vivement conseillé de débrancher la carte relais du secteur avant de récupérer ou de reposer l'ESP.

III - Code de l'ESP8266

Librairies utilisées :

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

Code :

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

#define CLIENT_ID  "rdc_lab"      // ID du relais , à changer pour chaque relais
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
byte MsgRL1Off[] = {0xA0, 0x01, 0x00, 0xA1};  // ferme le relais #1

byte MsgRL2On[] = {0xA0, 0x02, 0x01, 0xA3};   // ouvre le relais #2
byte MsgRL2Off[] = {0xA0, 0x02, 0x00, 0xA2};  // ferme le relais #2

byte MsgRL3On[] = {0xA0, 0x03, 0x01, 0xA4};   // ouvre le relais #3
byte MsgRL3Off[] = {0xA0, 0x03, 0x00, 0xA3};  // ferme le relais #3

byte MsgRL4On[] = {0xA0, 0x04, 0x01, 0xA5};   // ouvre le relais #4
byte MsgRL4Off[] = {0xA0, 0x04, 0x00, 0xA4};  // ferme le relais #4

//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));
  }
  if (topic != "re:relays" & cmd.substring(0, length).equals("l2_on")) {
    mqttClient.publish("re:relays" , "l2_on");
    Serial.write(MsgRL2On , sizeof(MsgRL2On));
  }
  if ( cmd.substring(0, length).equals("l2_off")) {
    mqttClient.publish("re:relays" , "l2_off");
    Serial.write(MsgRL2Off , sizeof(MsgRL2Off));
  }
  if (topic != "re:relays" & cmd.substring(0, length).equals("l3_on")) {
    mqttClient.publish("re:relays" , "l3_on");
    Serial.write(MsgRL3On , sizeof(MsgRL3On));
  }
  if (cmd.substring(0, length).equals("l3_off")) {
    mqttClient.publish("re:relays" , "l3_off");
    Serial.write(MsgRL3Off , sizeof(MsgRL3Off));
  }
  if (topic != "re:relays" & cmd.substring(0, length).equals("l4_on")) {
    mqttClient.publish("re:relays" , "l4_on");
    Serial.write(MsgRL4On , sizeof(MsgRL4Off));
  }
  if (cmd.substring(0, length).equals("l4_off")) {
    mqttClient.publish("re:relays" , "l4_off");
    Serial.write(MsgRL4Off , sizeof(MsgRL4Off));
  }
}
```

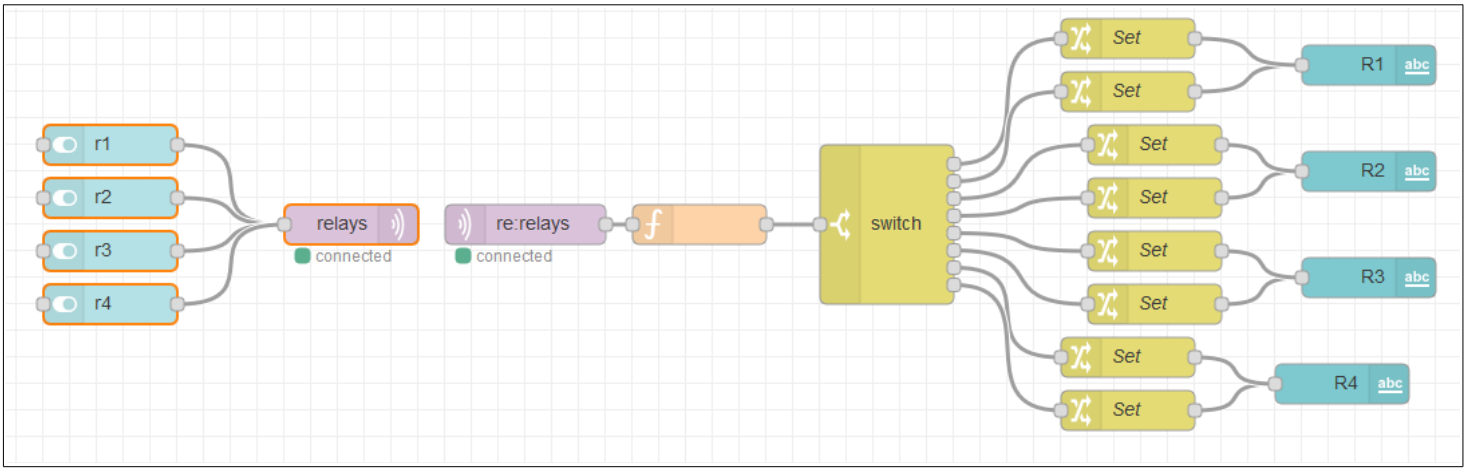
```
/*  SetUp  */
void setup() {

  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);
  }
}
```

Dans arduino, pour choisir la bonne carte , allez dans **Outils > Type de carte**
On utilise ici une ESP8266 générique. 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

- **Flow :**



Pour le refaire :

1. Placez 4 Nodes ‘ switch ’. Renommez-les (optionnel). Paramétrez l’ ui_group (add_new_ui_group > crayon) et l’ui_tab si nécéssaire (new_ui_tab > crayon) Donnez une largeur (Width) au groupe d’au moins 20.
   Paramétrez les messages de sortie. Le premier switch doit avoir ‘l1_on’ comme ‘On Payload’ et ‘l1_off’ comme ‘Off_Payload’. Répétez pour les 3 autres avec l2, l3 et l4.

2. Placez une node mqtt out et reliez-la aux 4 relais. Entrez ‘localhost:1883’ dans le broker (add_new_broker > crayon > server : localhost, port:1883), et entrez ‘relays’ comme topic.

3. Placez une node mqtt in, même broker que pour la node précédente, et ‘re:relays’ comme topic.

4. Placez une node ‘Switch’. Créez 8 conditions, une par message possible (l1_on, l1_off,…)

5. Placez 8 nodes Change. Sur celles reliées aux conditions ‘on’, laissez le choix SET et le champs msg.payload et entrez ‘green’ dans le champs To. Sur celles reliées aux conditions ‘off’, laissez le choix sur SET et le champs msg.payload, et entrez ‘red’ dans le champs To.

6. Placez 4 nodes Text et reliez chaqu'une d'elles aux deux nodes change correspondant au même capteur. Mettez-les dans le même UI_Group que les switchs, laissez le label vide et mettez comme value : .

7. Déployez et accédez au dashboard (panneau de droite > icone de graphe > icone de lien (carré + flèche))

Le serveur peut être adapté pour être utilisé avec une carte contenant un nombre différent de relais (en modifiant le nombre d'interrupteurs et champs texte), voire avec plusieus cartes différentes utilisées en même temps.(en clonant l'interface autant de fois que nécéssaire et en modifiant tous les messages mqtt pour qu'ils incluent l'id de la carte concernée en topic)