PROJET | PRISE CONNECTÉE

PRÉSENTÉ PAR :

LÉO DEL GIUDICE | RUBEN OBERLI | ARTHUR GILBERT | THOMAS BOCQUET | TOM HEIM

SOMMAIRE

I. Raspberry
   i. Apache2 & SSL & PHP
   ii. Broker MQTT
   iii. Python
   iv. Site WEB

II. ESP8266
   i. Bouton & LED
   ii. Wi-Fi
   iii. MQTT
   iv. Lien site / app

III. Application
   I. Page de Login
   II. MQTT

IV. Prise connectée
   I. Liste des composants
   II. Schéma
   III. Routage

V. Conclusion

RASPBERRY | SERVEUR APACHE2 SSL & PHP PYTHON & BROKER

APACHE

HTTP SERVER PROJECT

Installation sous Raspberry

Serveur HTTP
Version 7 de PHP

Mise en place d'HTTPS
Redirection HTTP → HTTPS

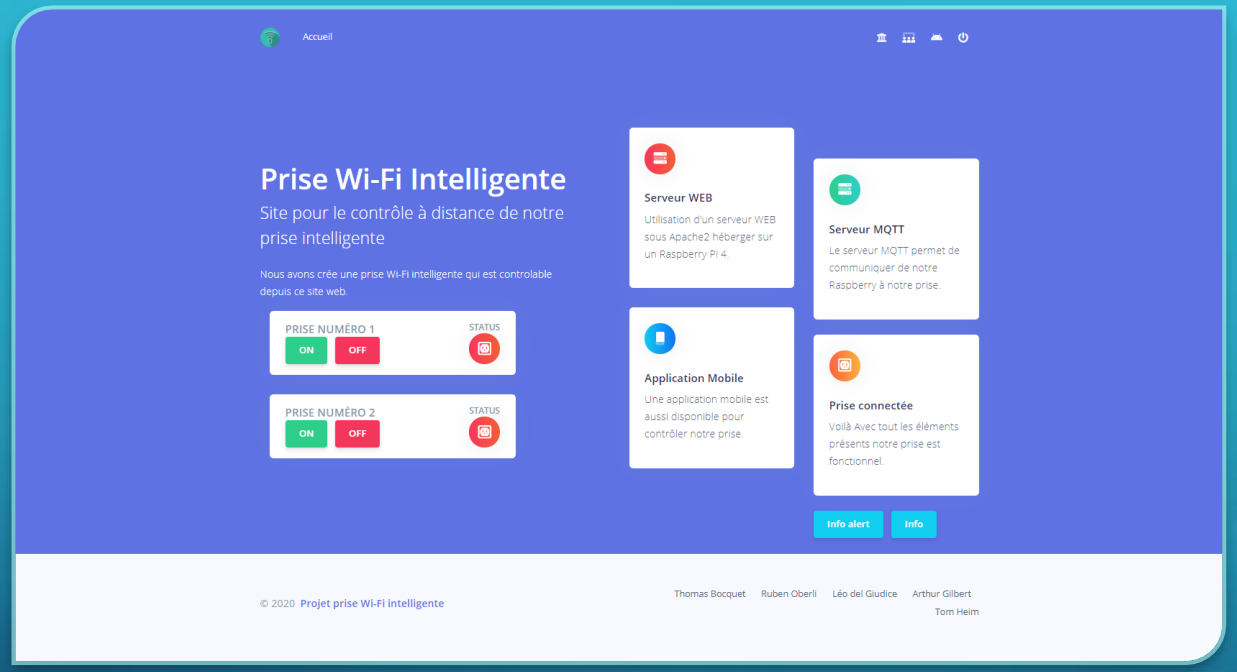
Création des scripts

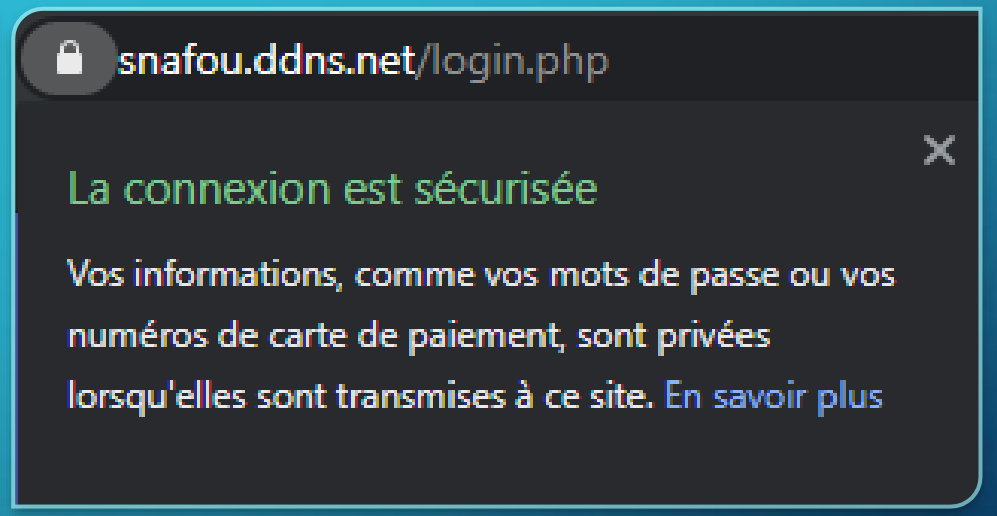
CONFIGURATION APACHE2 AVEC HTTPS

```
<VirtualHost *:80>
        Servername snafou.ddns.net

        ServerAdmin contact@liberty-host.com
        DocumentRoot /var/www/html

        ErrorLog /var/www/logs/error.log
        CustomLog /var/www/logs/access.log combined

        RewriteEngine on
        RewriteCond %{SERVER_NAME} =snafou.ddns.net
        RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
```

```
<IfModule mod_ssl.c>
<VirtualHost *:443>
        ServerName snafou.ddns.net

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog /var/www/logs/error.log
        CustomLog /var/www/logs/access.log combined

        Include /etc/letsencrypt/options-ssl-apache.conf
        SSLCertificateFile /etc/letsencrypt/live/snafou.ddns.net/fullchain.pem
        SSLCertificateKeyFile /etc/letsencrypt/live/snafou.ddns.net/privkey.pem
</VirtualHost>
</IfModule>
```

IMAGE DE BON FONCTIONNEMENT





RASPBERRY | BROKER MQTT

Installation de Mosquitto

Création de crédenciales

Topic par objet connectée

Message distinctif

PYTHON

```
Connected = False
broker_address= "snafou.ddns.net"
port = 1883
user = "dutrt2020"
password = "dutrt2020"

client = mqttClient.Client("Listener MQTT PX")
client.username_pw_set(user, password=password)
client.on_connect= on_connect
client.on_message= on_message
client.connect(broker_address, port=port)
client.loop_start()

while Connected != True:
    time.sleep(0.1)
    client.subscribe("pX_re")
try:
    while True:
        time.sleep(1)

except KeyboardInterrupt:
    print "exiting"
    client.disconnect()
    client.loop_stop()
```

```
import paho.mqtt.client as mqttClient
import time

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connection reussi !")
        global Connected
        Connected = True
    else:
        print("Connection echouer !")

def on_message(client, userdata, message):
    print "Message recu: "  + message.payload
    if message.payload == 'on':
        fichier = open("lib/return/pX.txt", "w")
        fichier.write("on")
        fichier.close()
    if message.payload == 'off':
        fichier = open("lib/return/pX.txt", "w")
        fichier.write("off")
        fichier.close()
```

Script autonome

Condition et stockage de l'information

```
../lib/return/
├── p1.txt
└── p2.txt
```

SITE WEB

Script PHP → Commande Shell → Mosquitto

Fichier on.php

```php
<?php
        $output = shell_exec('mosquitto_pub -h 192.168.1.201 -p 1883 -u dutrt2020 -P dutrt2020 -t p1_se -m "off"');
        echo "<pre>$output</pre>";
?>
```

Fichier off.php

```php
<?php
        $output = shell_exec('mosquitto_pub -h 192.168.1.201 -p 1883 -u dutrt2020 -P dutrt2020 -t p1_se -m "on"');
        echo "<pre>$output</pre>";
?>
```

SITE WEB

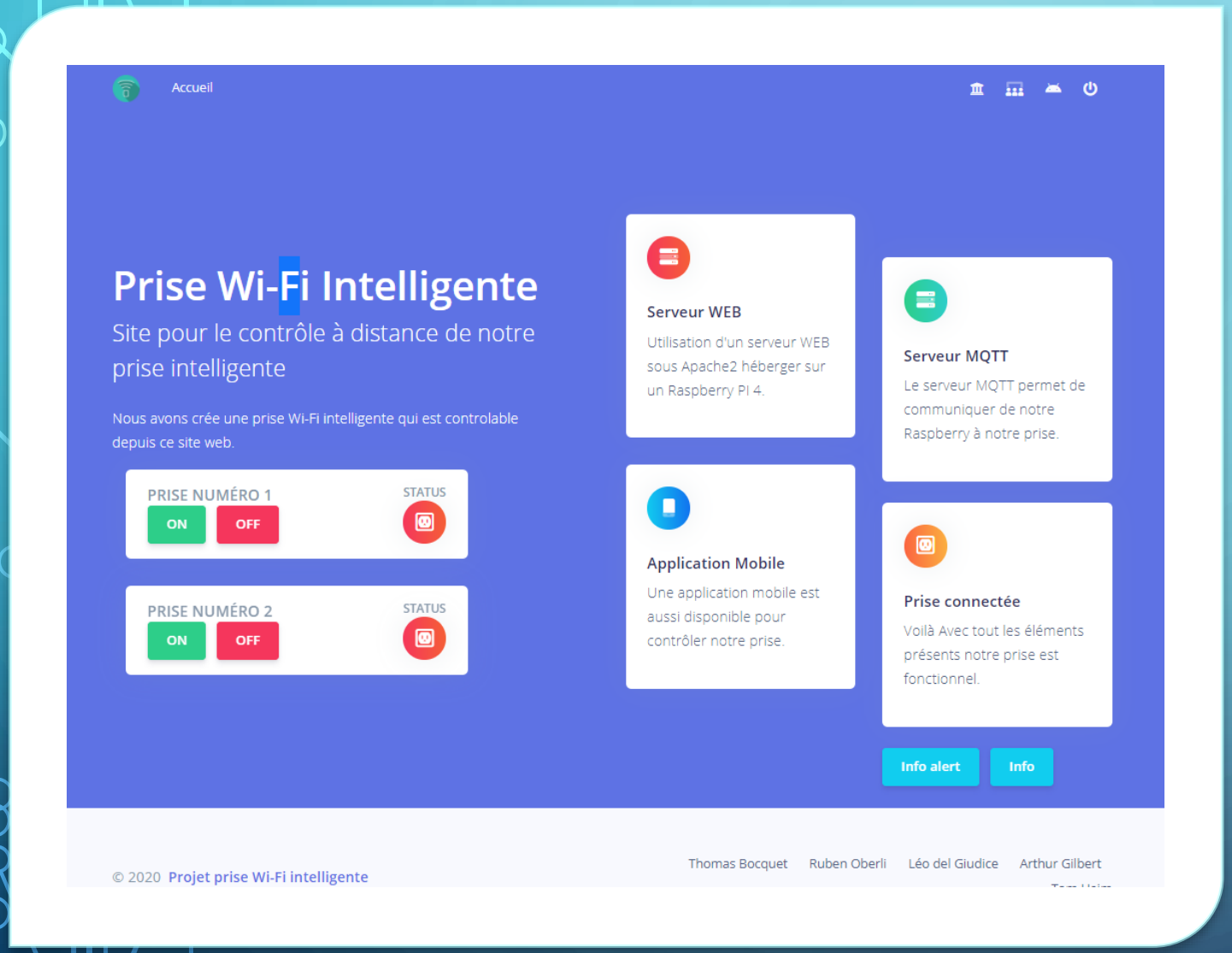
```
$.ajax('https://snafou.ddns.net/lib/return/p1.txt').done(function(responseP1) {
    console.log('Réponse requête P1: ', responseP1);

    var switchElement = $('#status_p1');

    if (responseP1 === 'on') {
        switchElement.addClass('bg-gradient-green');
        console.log('DeBug : P1 ON')
    } else if (responseP1 === 'off') {
        switchElement.addClass('bg-gradient-red');
        console.log('DeBug : P1 OFF')
    }
});

$.ajax('https://snafou.ddns.net/lib/return/p2.txt').done(function(responseP2) {
    console.log('Réponse requête P2: ', responseP2);

    var switchElement = $('#status_p2');

    if (responseP2 === 'on') {
        switchElement.addClass('bg-gradient-green');
        console.log('DeBug : P2 ON')
    } else if (responseP2 === 'off') {
        switchElement.addClass('bg-gradient-red');
        console.log('DeBug : P2 OFF')
    }
});
```

```
function reFresh() {
    location.reload(true)
}
window.setInterval("reFresh()",5000);
```

Script JS

- Actualisation automatique.
- Récupération des informations stocker par python.
- Exécution du script PHP par bouton.

```
// Prise 1
$("#p1_on").on("click", e => {
  $.post("lib/php/p1/on.php")
});
$("#p1_off").on("click", e => {
  $.post("lib/php/p1/off.php")
});

// Prise 2
$("#p2_on").on("click", e => {
  $.post("lib/php/p2/on.php")
});
$("#p2_off").on("click", e => {
  $.post("lib/php/p2/off.php")
});
```

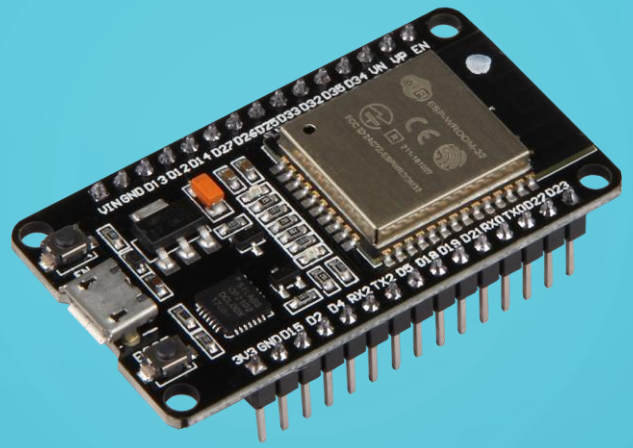


SITE WEB

RENDU FINAL DU SITE.

ESP8266

- Commande Bouton → LED
- Connexion Wifi
- Connexion Broker
  - ↳ Réception information
  - ↳ Envoie d'information



ESP8266 | BOUTON ⟶ LED

```
// Configuration LED
int bouton = 14; // bouton sur la PIN 14
int led = 13; // led sur la PIN 13
int etatbouton = 0; // variable etat bouton (appuie ou non appuie)
int dernieretatbouton = 0; // variable memoire derniere position du bouton
int etatled = 0; // varibale de la led, soit éteinte soit allumée
```

```
void ledd() {
  // Stockage de la position du bouton dans la variable etatbouton
  etatbouton = digitalRead(bouton);
  if (etatbouton != dernieretatbouton) {
    if (!etatbouton) {  // si etatbouton est différent de 1 (je rappelle que les états sont inversés dûs à la résistance de PULLUP)
      if (etatled) {    // et que si etatled est à 1
        etatled = 0;    // nous passons etatled à 0
        WifiOffP1();    // envoi du message off dans le channel MQTT depuis la fonction WifiOffP1()
      }
      else {
        etatled = 1;    // sinon nous le passons à 1
        WifiOnP1();     // envoi du message on dans le channel MQTT depuis la fonction WifiOnP1()
      }
    }
    dernieretatbouton = etatbouton;
  }
  Serial.println (etatled);
  digitalWrite(led, etatled);
  delay(500);
}
```

```
void loop() {
  ledd();
```

ESP8266 | WIFI

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

// WIFI
WiFiClient espClient;
```

```
// on demande la connexion au WiFi
WiFi.begin(ssid, password);
Serial.println("");
// on attend d'etre connecte au WiFi avant de continuer
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
// on affiche l'adresse IP qui nous a ete attribuee
Serial.println("");
Serial.print("IP address: ");
Serial.println(WiFi.localIP());
```

```
// Configuration WiFi
const char* ssid = "arthur"; // SSID de votre WiFi
const char* password = "........"; // mot de passe de votre WiFi
```

ESP8266 | MQTT - CONNEXION

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

```
// MQTT server
char mqtt_server[] = "86.243.17.132";  // adresse IP serveur
PubSubClient MQTTclient(espClient);
#define MQTT_USER "dutrt2020"
#define MQTT_PASS "dutrt2020"
```

```
// Connection au server MQTT ainsi que le port
MQTTclient.setServer(mqtt_server, 1883);
// Inisialisation du service callback MQTT sous le nom de MQTTcallback
MQTTclient.setCallback(MQTTcallback);
```

ESP8266 | MQTT - RECONNEXION

```
MQTTclient.loop();
//
static uint32_t  lastTimeMqtt = 0;
// Connecte le serveur MQTT
if (!MQTTclient.connected()) {
  MQTTconnect();
}
if (millis() - lastTimeMqtt >= 10000)  // toutes les 10 secondes
{
  lastTimeMqtt = millis();
}
```

```
void MQTTconnect() {

  // Boucle d'attente de connexion au serveur MQTT
  while (!MQTTclient.connected()) {
    Serial.print("Attente  MQTT connection...");
    String clientId = "Client_MQTT";   // création d'un id client unique

    // test connexion
    if (MQTTclient.connect(clientId.c_str(), MQTT_USER, MQTT_PASS)) {
      Serial.println("connected");

    } else {  // si echec affichage erreur
      Serial.print("ECHEC, rc=");
      Serial.print(MQTTclient.state());
      Serial.println(" nouvelle tentative dans 5 secondes ");
      delay(5000);
    }
    // Se connecte au chanel de réception MQTT
    MQTTclient.subscribe("p1_se");
  }
}
```

ESP8266 | LIEN SITE / APP

MQTT

↳ Topic

↳ Écoute

↳ Réponse

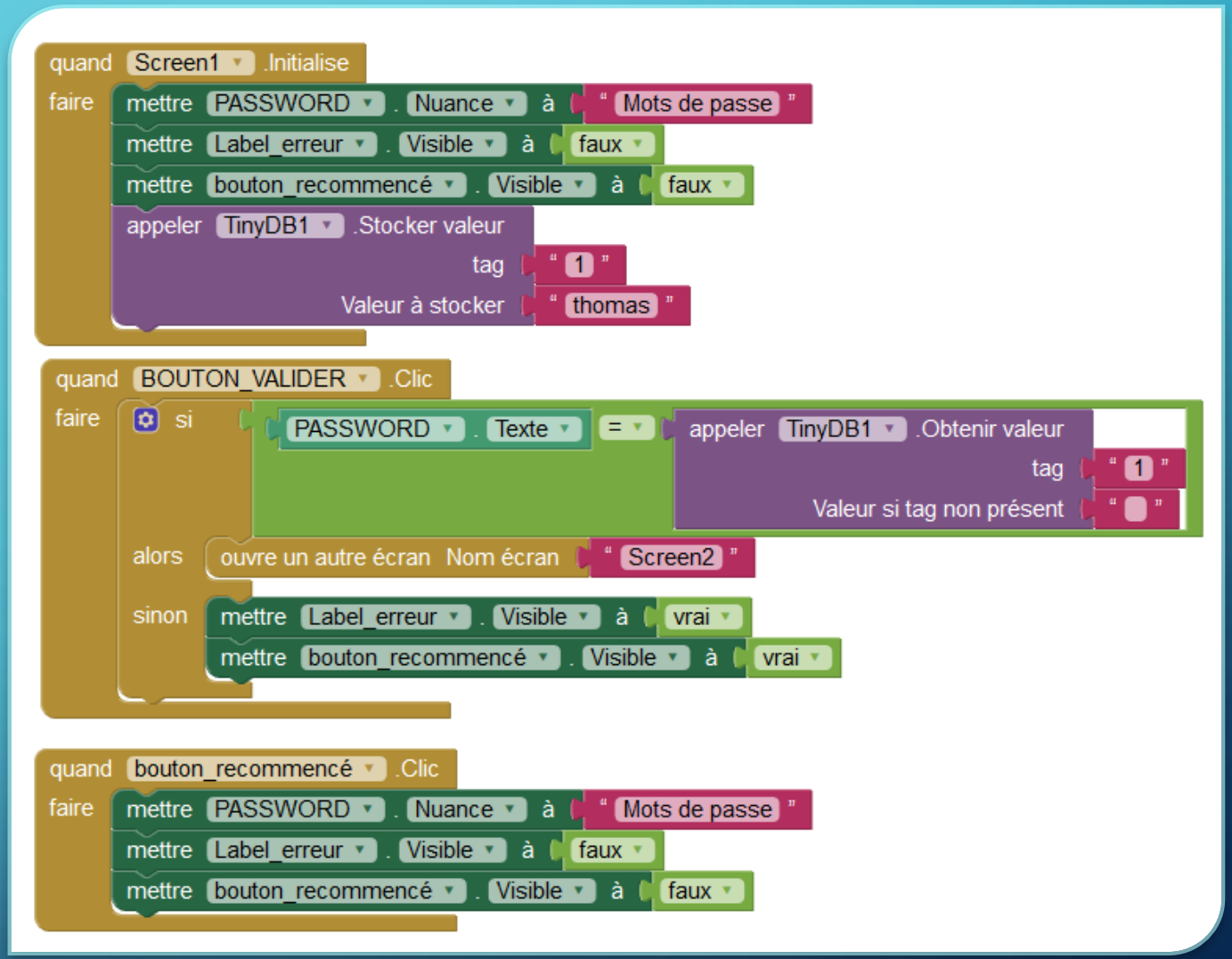
Différent par équipement

```
// Se connecte au chanel de réception MQTT
MQTTclient.subscribe("p1_se");
```

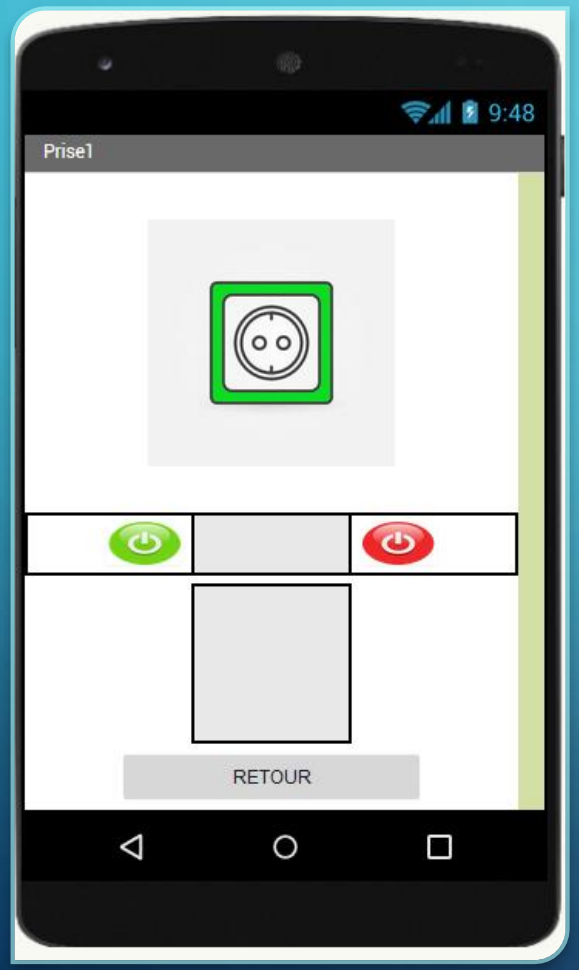
```
// Récupère les réponses et les formates dans le bon format
void MQTTcallback(char* topic, byte* payload, unsigned int length) {
  payload[length] = '\0';
  String s = String((char*)payload);
  Serial.println(s);
  // Si la réponse est on, la led passe à 1 en appelant le void WifiOnP1() pour changer l'état du site
  if (s == "on") {
    etatled = 1;
    WifiOnP1();
  }
  // Si la réponse est off, la led passe à 0 en appelant le void WifiOffP1() pour changer l'état du site
  if (s == "off") {
    etatled = 0;
    WifiOffP1();
  }
}
```

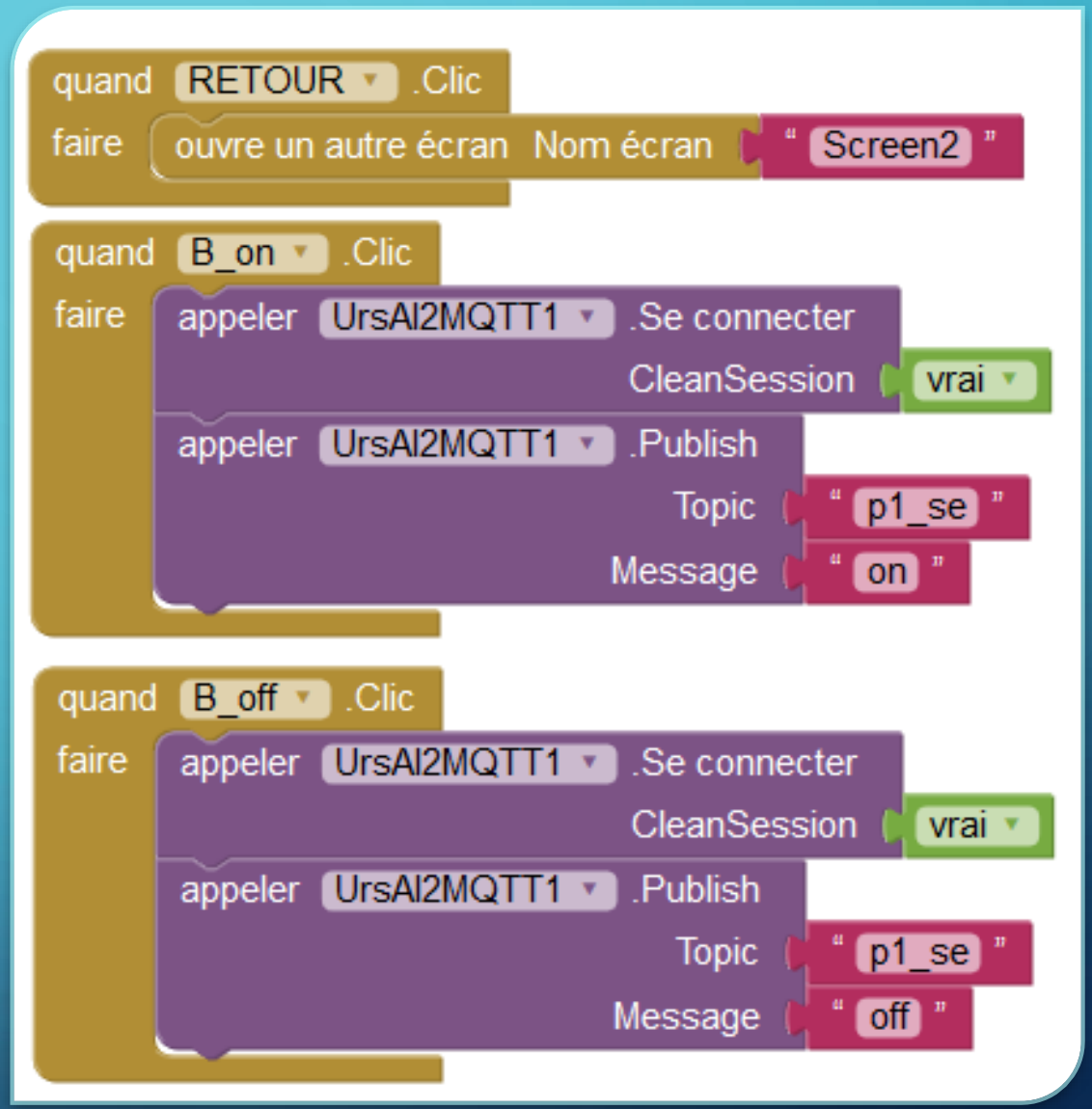
```
// Message d'envoie pour le site web (P1)
// Message ON
void WifiOnP1() {
  String reponse = "on";
  MQTTclient.publish("p1_re", reponse.c_str());
}
// Message OFF
void WifiOffP1() {
  String reponse = "off";
  MQTTclient.publish("p1_re", reponse.c_str());
}
```

APPLICATION | PAGE DE LOGIN

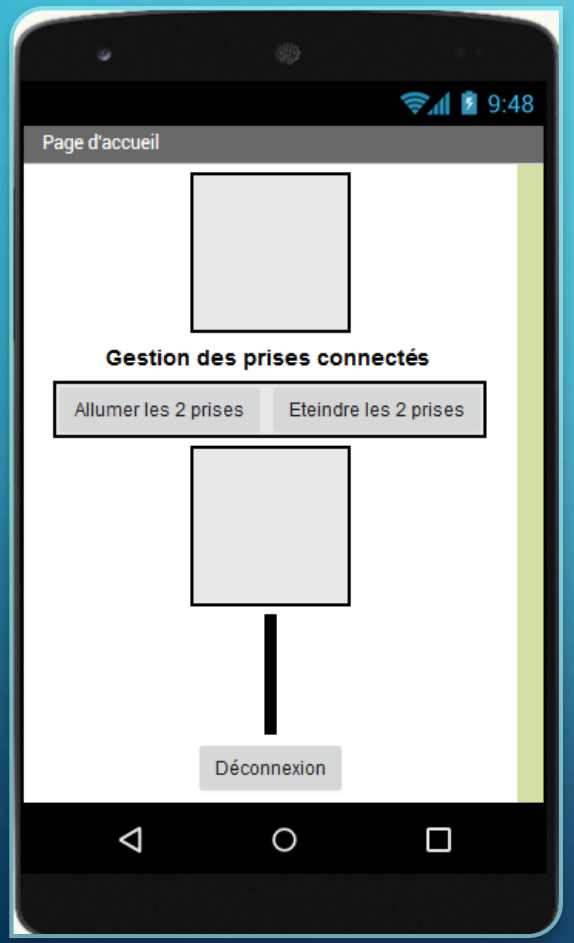


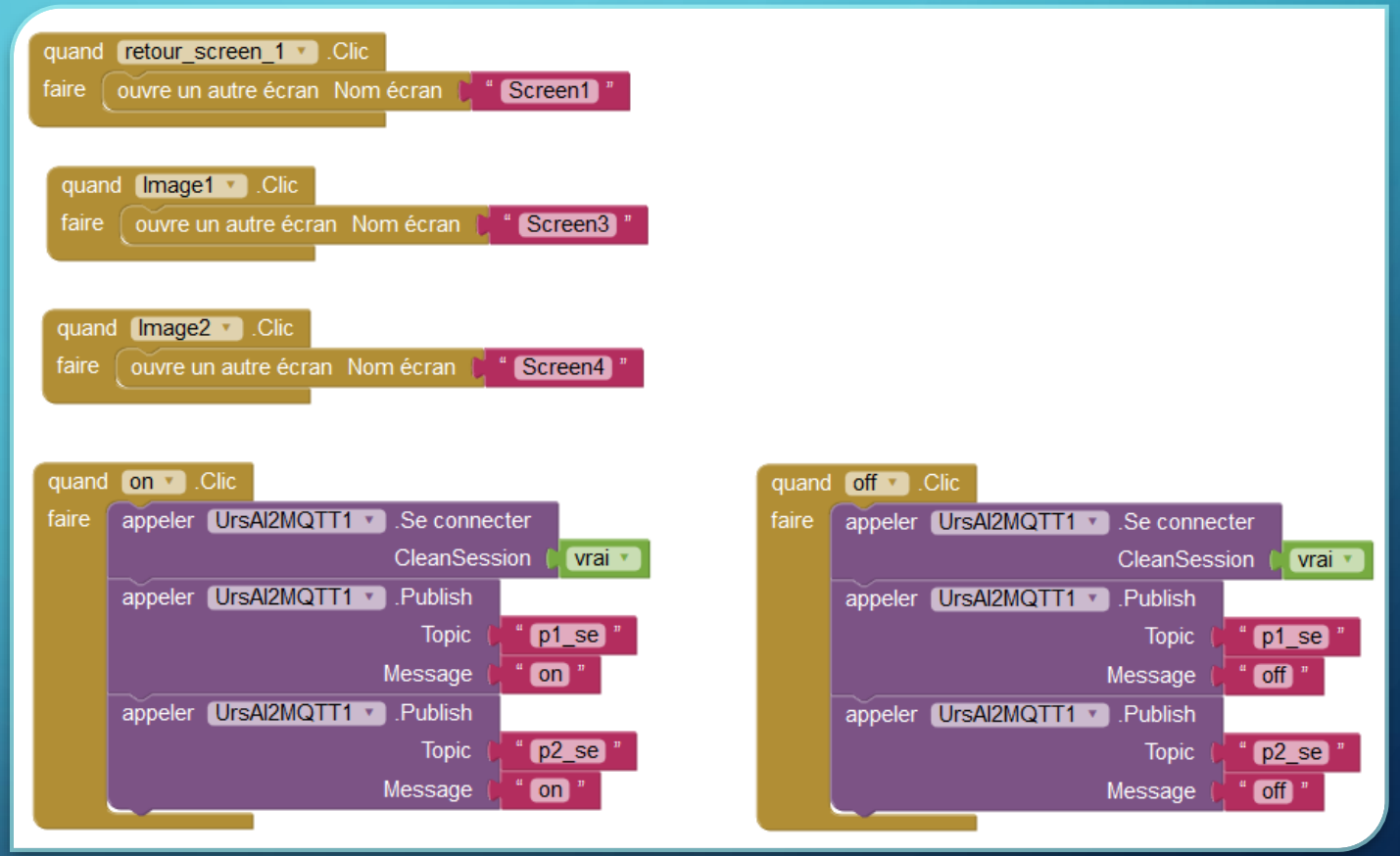
APPLICATION | MQTT





APPLICATION | MQTT





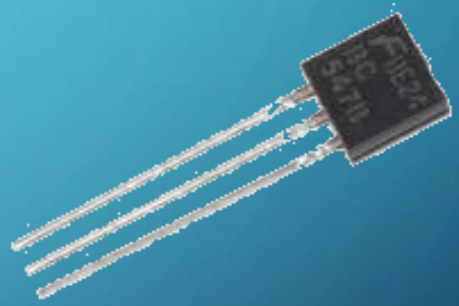
PRISE CONNECTÉE | LISTE DES COMPOSANTS

Node MCU – ESP8266

Prise mal - femelle

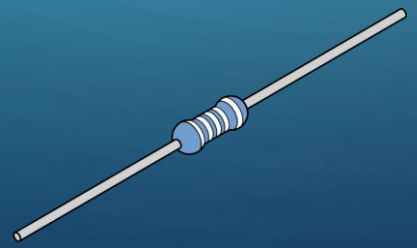
LED

Transistor



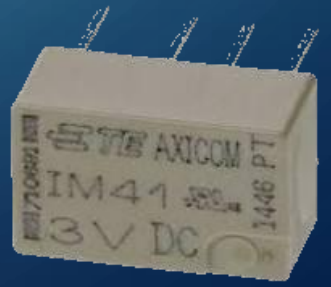
Bouton poussoir

Résistance

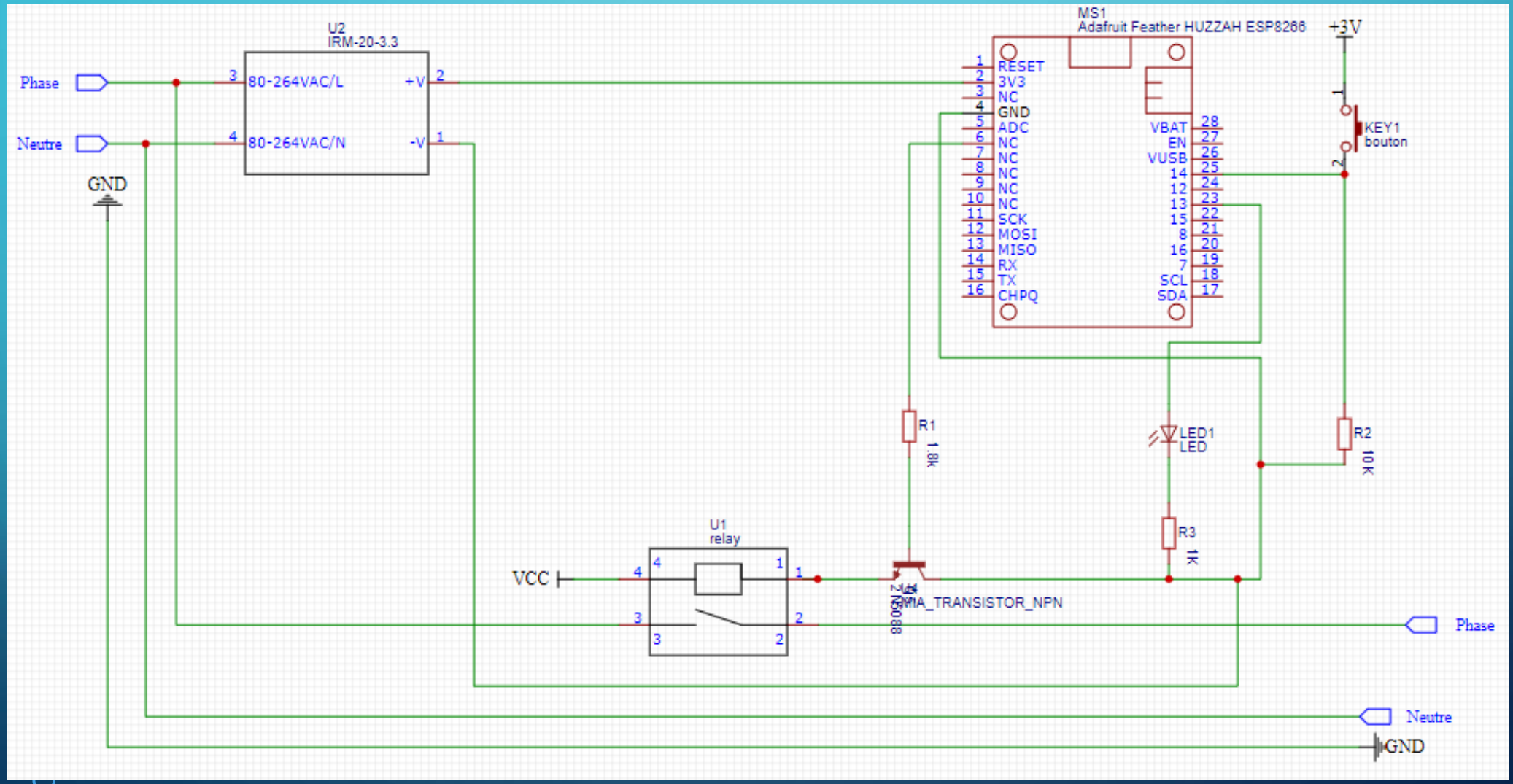


Transformateur 3.3V

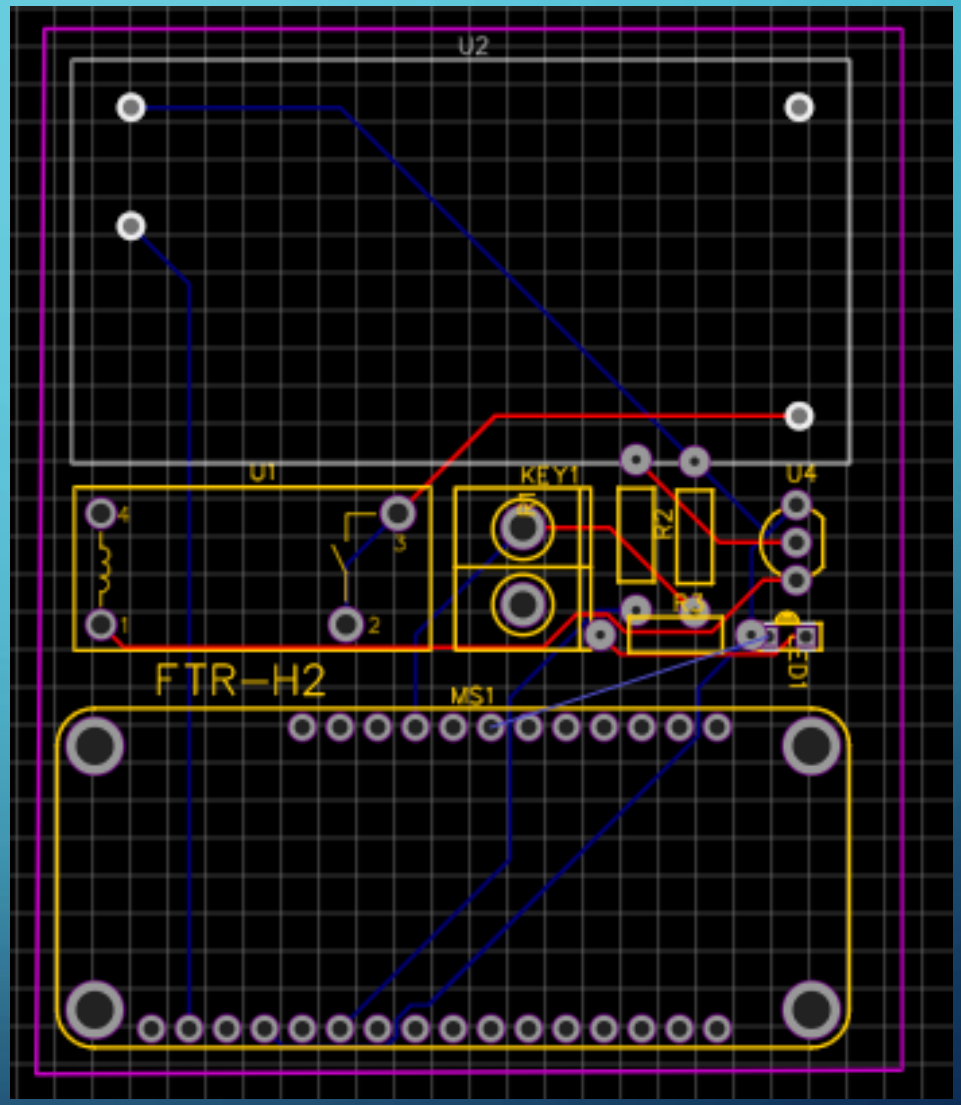
Relais



PRISE CONNECTÉE | SCHÉMA



PRISE CONNECTÉE | ROUTAGE



CONCLUSION

Merci de nous avoir écouter !

Nous vous écoutons pour tout vos question.