



UNIVERSITÉ
DE LORRAINE



Henri Poincaré
LONGWY

MODULE : Supervision et contrôle à distance

Mise en œuvre de système IOT

Réalisation d'un système de contrôle
d'accès par RFID et Web pour une salle.

PROFESSEUR M . Mehdi SELMANI



Etudiants

LAMAH Severin

FIGUEROA Carlos



12 DE FÉVRIER DE 2021



Sommaire

Objectif.....	2
Cahier de Charge.....	2
Matériel électronique et schéma Electric.....	3
Logiciel	4
Programmation.....	5
Conclusion.....	9

Objectif

Comprendre et sensibiliser à la problématique liée à l'IOT, comprendre la chaîne de mesure de données et le développement d'une application WEB, impliquant :

- a) La conception de système de mesure communicant (Arduino)
- b) La programmation de système embarquée et Web
- c) La connectivité à Internet et la transmission de données via internet
- d) La mise en œuvre de plateforme web et la mise à disposition de données sur le Web

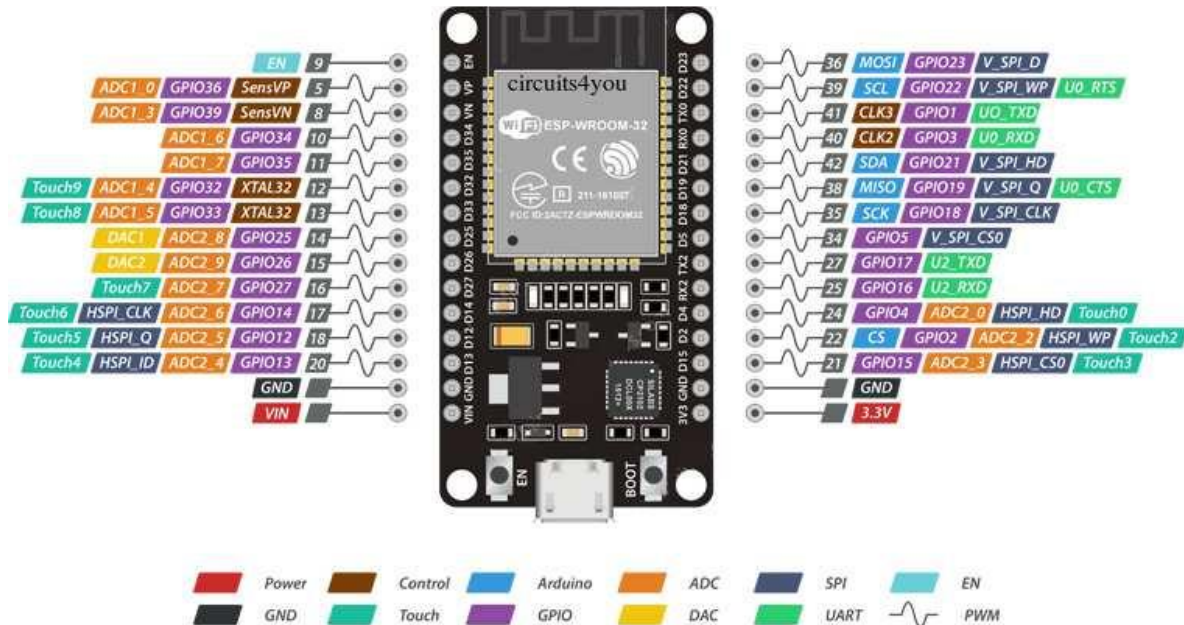
Cahier de Charge

Le projet comprendra les éléments suivants

1. Adaptations à l'interface d'Arduino
2. Créer et programmer un module de lecture RFID.
3. Développer une interface graphique WEB accessible à l'utilisateur.
4. Contrôler l'accès par lecture RFID
5. Contrôler l'accès par un mode manuel
6. Rédiger un rapport du Mini Project

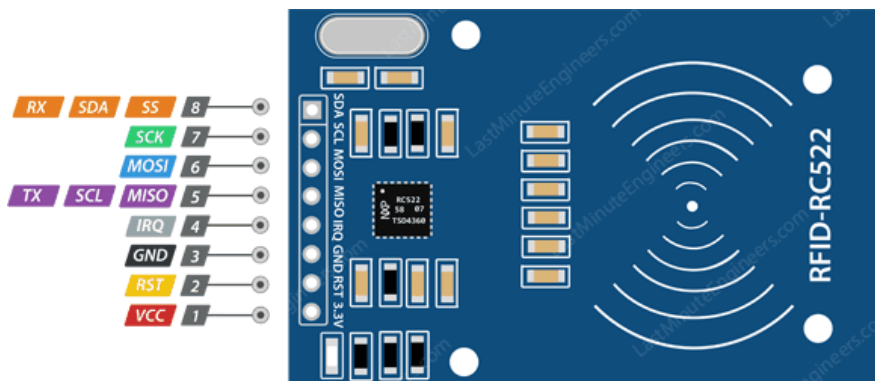
Matériel électronique et schéma Electric

Le cœur de notre projet c'est ESP32 que c'est est une série de microcontrôleurs de type système sur une puce (SoC) d'Espressif Systems, basé sur l'architecture Xtensa LX6 de Tensilica , intégrant la gestion du Wi-Fi et du Bluetooth (jusqu'à LE 5.0 et 5.11) en mode double, et un DSP. C'est une évolution d'ESP8266.



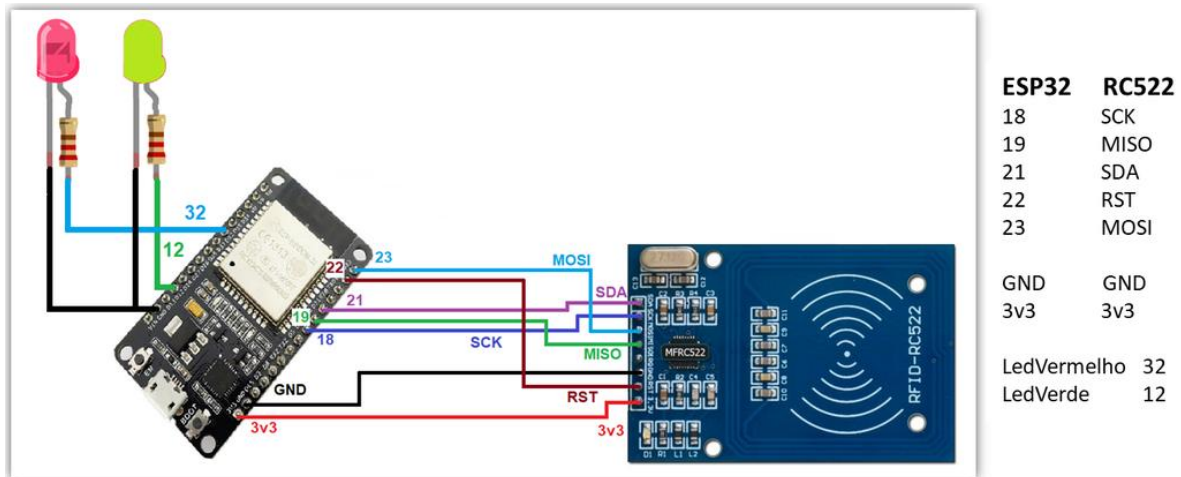
ESP32 Dev. Board Pinout

Ensuite pour faire la lecture nous utilisons une module RFID-RC522 de Joy-it qu'est compatible avec Arduino et Raspberry Pi, basée sur le circuit MFRC-522 de NXP et est utilisée pour lire et écrire sur des cartes ou badges RFID de type Mifare.



RC522 Pinout

Les connexions de l'Arduino, des leds et du lecteur RFID sont les suivantes



Logiciel

La programmation pour le microcontrôleur se compile sur Arduino pour Windows



Et pour la création de la page web nous utilisons un editor de codage HTML très connu DreamViewer 2020



Programmation

Premièrement pour faire la programmation sera nécessaire d'installer de libraires pour contrôler de manière correcte tous les périphériques

- ISP.h
- RFID.h
- WiFi.h
- FS.h
- ESPAsyncWebServer.h

En général, les bibliothèques doivent connecter rapidement différents périphériques ou mieux gérer un processus est le cas des 2 derniers, qui sont responsables de l'optimisation du fonctionnement d'un site Web sur ESP32 et la gestion de l'information entre elle

Pour commencer, nous créons d'abord la communication sans fil grâce au wifi intégré de l'ESP32.

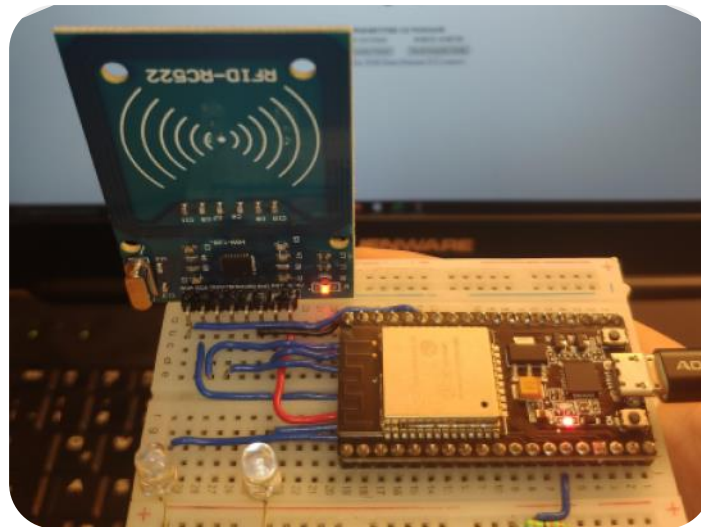
```
#include <WiFi.h>
// Parametres WiFi
const char *ssid = "FIZET";
const char *password = "fizet_mx";
```

Puis nous ajoutons le fonctionnement de la bibliothèque RFID et l'initialisation des variables

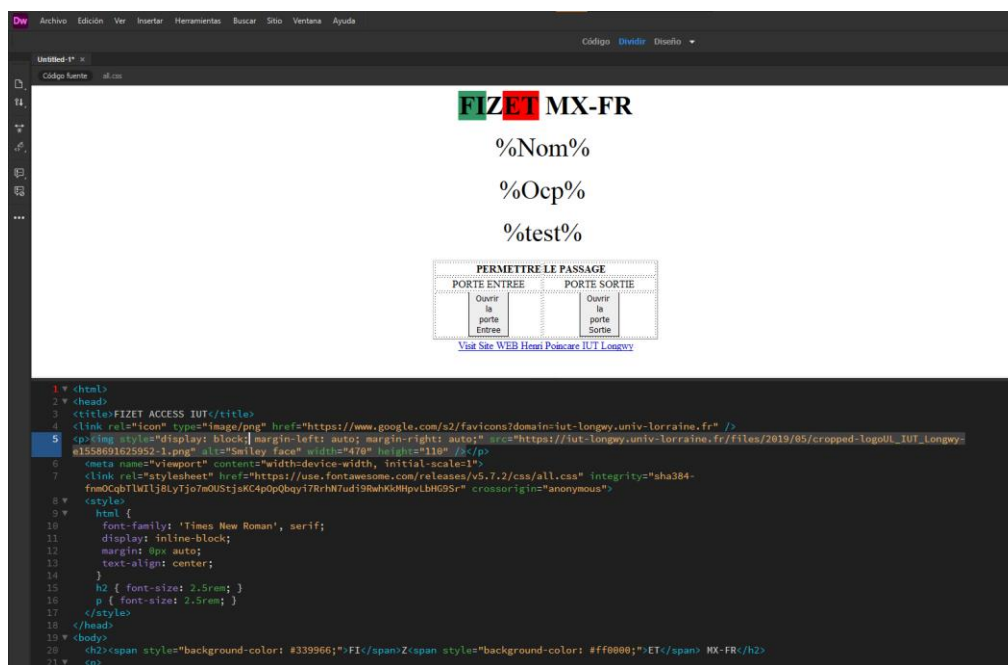
```
/// Librerie de RFID
#include <SPI.h>
#include <RFID.h>
//Parametres de configuration de pin pour RFID
#define SS_PIN 21
#define RST_PIN 22

RFID rfid(SS_PIN, RST_PIN);
```

Nous procédons ensuite à la connexion pour commencer les tests.



Pour la création d'une interface graphique programmée en HTML, DreamViewer a donc été utilisé de manière très intuitive pour atteindre le résultat souhaité.



Plus tard, on ajoute la lecture de la carte RFID et en même temps on fait la comparaison avec un if avec beaucoup de conditions à lire, dans ce cas ce qui est en train de comparer c'est que l'ID de la carte NFC est avec celui qui est dans la programmation, si oui, les variables changent de valeur par le nom de la personne et active la LED en simulation d'une porte.

```
//RFID Lecture
if (rfid.isCard()) {
  if (rfid.readCardSerial()) {
    Serial.println(" ");
    Serial.println("Carte Trouvé");
    Serial.println(" ");
    Serial.println("Nombre Carte:");
    Serial.print("Dec: ");
    Serial.print(rfid.serNum[0], DEC);
    Serial.print(", ");
    Serial.print(rfid.serNum[1], DEC);
    Serial.print(", ");
    Serial.print(rfid.serNum[2], DEC);
    Serial.print(", ");
    Serial.print(rfid.serNum[3], DEC);
    Serial.print(", ");
    Serial.print(rfid.serNum[4], DEC);
    Serial.println(" ");
    // Différenciation des cartes
    if (rfid.serNum[0] == ul_0
        && rfid.serNum[1] == ul_1
        && rfid.serNum[2] == ul_2
        && rfid.serNum[3] == ul_3
        && rfid.serNum[4] == ul_4) {

      Serial.println("Portable FIZET");

      nom = "Bienvenue Carlos Figueroa";
      ocp = "Etudiant";
      /*
      temp_init_milis = millis();
      ///Action
      digitalWrite(led_1, HIGH);
      ///
      while (millis() < temp_init_milis + temp_milis) {
        // atteindre [temp_milis] milisegundos
      }
      digitalWrite(led_1, LOW);
      */
    }
  }
}
```

En même temps, voici une ligne de code intéressante, la commande milis, qui consiste à utiliser le timer interne de l'ESP32 pour empêcher le microcontrôleur d'utiliser un délai commun et de couper la connexion wifi.

En plus d'afficher à tout moment les informations sans avoir besoin de rafraîchir la page, nous utilisons un script sur le même programmation HTML dans le but de rafraîchir uniquement ces valeurs à un certain moment sans affecter les autres valeurs.

```
// Script pour mise a jour les donnees
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("Nom").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/Nom", true);
  xhttp.send();
}, 1000 ) ;
```


Presque pour finir d'ajouter un contrôle manuel, nous devons ajouter un bouton de la programmation HTML et renvoyer une action dans le même protocole https pour que l'ESP32 l'exécute.

```
<a href="/p_entree"><button>Ouvrir la porte Entree</button></a></td>  
<a href="/p_sortie"><button>Ouvrir la porte Sortie</button></a></td>
```

Dans ce cas, il s'agit d'envoyer un appel dans le même serveur à l'adresse /p_entree ou /p_sortie selon le cas.

Et ensuite dans la lecture du serveur à l'intérieur de l'ESP32 on ajoute l'action à effectuer en cas d'appel de cette adresse

```
// pour mise à jour les bouton (entree)  
server.on("/p_entree", HTTP_GET, [] (AsyncWebServerRequest * request) {  
    digitalWrite(led_1, HIGH);  
    delay(2000);  
    digitalWrite(led_1, LOW);  
    request->send_P(200, "text/html", index_html, processor);  
});  
server.on("/p_sortie", HTTP_GET, [] (AsyncWebServerRequest * request) {  
    digitalWrite(led_2, HIGH);  
    delay(2000);  
    digitalWrite(led_2, LOW);  
    request->send_P(200, "text/html", index_html, processor);  
});
```


Conclusion

Finalement, le cahier des charges a été rempli avec succès



FIZET MX-FR

 Bienvenue Roberto Zetina

 Directeur

 42

PERMETTRE LE PASSAGE

PORTE ENTREE

PORTE SORTIE

Ouvrir la porte Entree

Ouvrir la porte Sortie

[Visit Site WEB Henri Poincare IUT Longwy](#)

Dans ce mini projet, nous avons utilisé tout ce que nous avons vu en classe pour avoir une base solide de connaissances sur l'IoT, grâce aux pratiques précédentes, nous avons atteint avec plus de facilité un développement optimal.

Ce projet nous a permis de comprendre les bases de la lecture des cartes RFID tout en comparant les différents identifiants pour effectuer des tâches spécifiques. D'autre part, nous avons compris l'utilisation d'éditeurs web pour améliorer l'entité graphique et en même temps l'utilisation de scripts spécifiques pour améliorer les performances de la page web et enfin, en utilisant des références résultant de la pression d'un bouton sur la page web pour ajouter ultérieurement une action spécifique à la page web, nous avons pu comprendre comment utiliser un éditeur web pour améliorer les performances de la page web.

En général, nous avons découvert le grand potentiel que l'IoT a dans ce monde connecté.

