Systèmes d'Information et Numérique		TSTI2D
Lycée J-B. VUILLAUME	Comment valider les contraintes techniques ?	SÉANCE 3
	Comment créer un serveur Web embarqué ?	Activité 3

Nom : …… Prénom : …… Note : /20

Durée : 4 H 00

Objectif visé : **O5 – Imaginer une solution, répondre à un besoin**

O6 – Préparer une simulation et exploiter les résultats pour prédire un fonctionnement,

Compétences : **CO 5.7 CO 6.2**

Connaissance visée : **SA 2.4.4. Transmission de l'information**
SA 3.4.3 Inter-opérabilité des produits
SA 4.3.5. Conception informationnelle des produits

Matériel nécessaire : Poste informatique équipé de Proteus ISIS et Arduino

Objectifs de l'activité : L'objectif de cette activité est de mettre en œuvre la programmation d'un serveur web dit "embarqué" sur un système microcontrôleur type Arduino. À partir du cahier des charges et de l'étude, l'élève doit être capable :

- De s'approprier le concept de codage d'une page HTML et de le transcrire pour un système microcontrôleur,
- De programmer la chaine complète d'information en s'appuyant sur un algorithme donné, et de valider la solution retenue par simulation en rapport au cahier des charges fixé,
- D'organiser son travail par étape, et de valider chacune de celles-ci aboutissant au résultat final permettant la communication distante avec un serveur Web embarqué.

♦ PRÉSENTATION DU PROJET :

Situation déclenchante : Présentation PowerPoint

Comme nous l'avons vu dans l'étude préliminaire, un objet connecté peut se commander par différents supports distants (téléphone mobile, interface web, tablette, PC, etc.) aux travers différentes technologies de communication filaires ou sans fil (Bluetooth, Ethernet, Wifi, Zeegbe, etc.)

Nous allons dans cette activité nous intéresser particulièrement à la connectivité de la serrure par le biais du support de communication Ethernet, et voir comment programmer une interface Web simple permettant le contrôle à distance de cet objet, et simuler la solution avec Proteus ISIS et un navigateur Web.

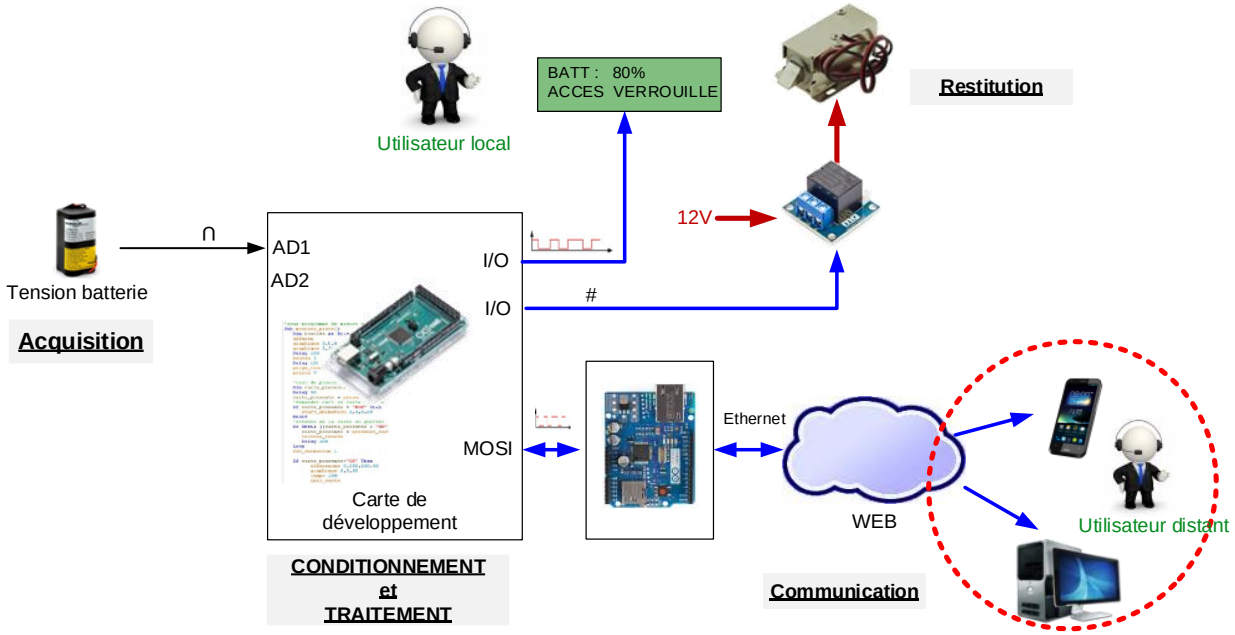
Se pose alors la problématique à laquelle nous allons répondre aux travers de l'activité suivante mettant en œuvre le principe de la communication TCP/IP sous Arduino :

- ***Comment communiquer les informations de la serrure à travers le Web ?***

Pour répondre à cette problématique vous devez :

- Mesurer et afficher le niveau de tension de la batterie,
- Commander par interface de puissance l'accès à l'habitat,
- Communiquer par le web le niveau de tension de la batterie,
- Analyser et traiter les commandes envoyées depuis le Web : gestion de l'accès
- Tester et valider le bon fonctionnement du système

Schéma synoptique du Projet :



♦ CAHIER DES CHARGES

Vous allez devoir réaliser le programme de la carte Arduino afin de prendre en compte le module shield Ethernet, configuré en mode Serveur Web.

Recopiez le dossier **SEANCE 3…\ACTIVITE3…\Ressources**\ dans votre dossier de travail.

Liaison Série RS232 :

- Vitesse de transmission 9600 bauds
- Nombre de bits : 8
- Parité : aucune
- Bit de Stop : 1

Port utilisé :

- **80** : protocole HTTP sous navigateur internet

Paramètres Réseaux :

- **Adresse IP** : 172.17.8. xx ou xxx représente le numéro du poste SIN
- **Adresse MAC** : Ensemble de 6 octets notés sur la face arrière de la carte Ethernet
- **DNS** : 172.17.1.1
- **Passerelle** : 172.17.0.1
- **Masque de sous-réseau** : 255.255.0.0

Afficheur LCD:

Si vous utilisez Ardublock pour la programmation, vous devez modifier votre programme au niveau de l'afficheur. Les broches habituellement utilisées avec Ardublock (12,11,6,5,4,3 et 2) **entrent en conflit avec la module Shield Ethernet** !

Algorithme de programmation du serveur Web :

Déclarations :

Importation des librairies "**Ethernet.h**" et "**SPI.h**" (prototypage) ou "**UIPEthernet.h**" (simulation)
Création de l'objet **server** sur le port **80** à partir de la libraire **EthernetServer**

mac[ ] est un tableau d'octets contenant l'adresse **MAC**
ip[ ] est un tableau d'octets contenant l'adresse **IP** du module
DNS[ ] est un tableau d'octets contenant l'adresse IP du serveur **DNS** du réseau
gateway[ ] = est un tableau d'octets contenant l'adresse IP de la **passerelle** du réseau
subnet[ ] = est un tableau d'octets contenant le **Masque de sous-réseau** du lycée

readString est un objet chaine (String) de 100 caractères maximum

Initialisation (Setup) :

Initialiser le module **Ethernet** par la fonction **begin** en passant les paramètres MAC, IP, DNS, PASSERELLE et MASQUE
Démarrer l'objet **server** avec la fonction **begin**

Programme principal (Loop):

client est un objet **EthernetClient**

client ⇦ données disponibles sur **server**

SI client est présent **ALORS**

- **caract** est un caractère
- **caract** ⇦ lecture du caractère contenu dans le buffer d'entrée de **server**
- **TANTQUE client** est **connecté**
  - **SI client** a **transmis des données**
    - **caract** ⇦ lecture du caractère envoyé par le **client**
    - **readString** ⇦ **readString** + String(**caract**) *// concaténation*
    - **SI caract** contient le caractère ***saut de ligne*** ('\n') **ALORS**
      - Temporisation de 100ms
      - **SI readString** contient **"ACCES" ALORS**
        - **SI readString** contient **"ON" ALORS**
          - Ouverture de l'accès
        - **SINON**
          - Verrouillage de l'accès
        - **FIN**
      - **FIN**

PARTIE 2 du TP

      - Transmission au **client** de l'entête de la page HTML :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta http-equiv= "Refresh" content ="2"  // rafraichir la page toute les 2 secondes
<meta http-equiv= "Connection" content ="close" // ferme la connexion à la fin de l'envoi
<title>Sti2d Serveur Web</title>
</head>
```

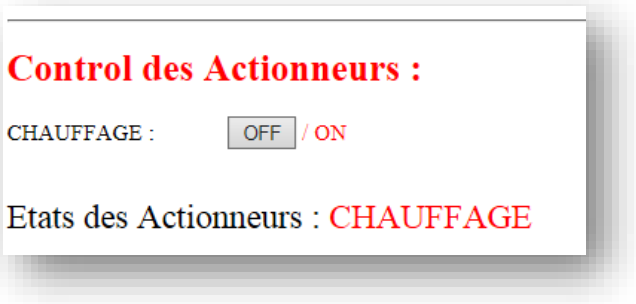
      - Transmission au **client** du corps de la page HTML ***(à FAIRE !!!)***
      - **SORTIR** → *fonction break*
    - **FIN**
  - **FIN**
- **FIN**
- Arrêter la connexion avec le **client** → *fonction client.stop( )*
- Vider la chaine de données : **readString** ⇦ ""

FIN

Temporisation de 30 ms

Interaction avec le Serveur : Ajout de Boutons

Votre interface doit permettre la commande d'actions sur votre Serveur (Arduino Mega), comme par exemple l'action de remise à zéro (RESET), la commande du chauffage, etc...



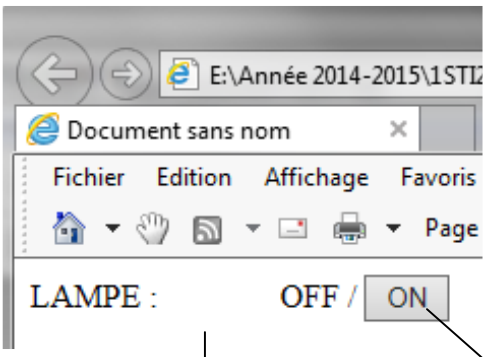
Il est possible en insérant des **champs** "**Bouton**" de pouvoir remédier à ce problème. L'astuce consiste à envoyer au serveur une **requête** d'action grâce à la méthode **GET**

La méthode **GET** est la valeur de méthode par défaut. On l'utilise si l'on veut que certains paramètres soient **ajoutés à l'URL** (dans d'autre cas on utilise POST qui n'affiche pas les valeur envoyées).

Il est donc possible ensuite sous le serveur Web Arduino d'analyser le contenu envoyé et de récupérer les données passées à la page.

La méthode GET en HTML

Exemple : je désire créer un bouton pour allumer ou éteindre une lampe depuis mon serveur Web :



<form method=get name=LAMPE> :
Permet de créer la méthode **GET** avec comme nom **LAMPE**

<input type=**submit** name=**LAMPE** value=**ON**>
Permet de créer le bouton (**submit**) qui actionnera la méthode "**LAMPE**" (donc **GET**) avec comme ajout la valeur "**ON**"

```
<form method=get name=LAMPE>
        LAMPE  :     OFF /  <input type=submit name=LAMPE value=ON>
</form>
```

Lors de l'appui sur le bouton **ON**, le navigateur enverra une requête au serveur Web Arduino en ajoutant à la fin de l'adresse URL le texte " **?LAMPE=ON** " :

htttp:\\172.17.8.5**?LAMPE=ON**

...ET APRÈS ?

Il suffit lors de la réception par le **serveur Web Arduino** de vérifier si la commande "**LAMPE**" est présente, puis si "**ON**" ou "**OFF**" sont également présents.

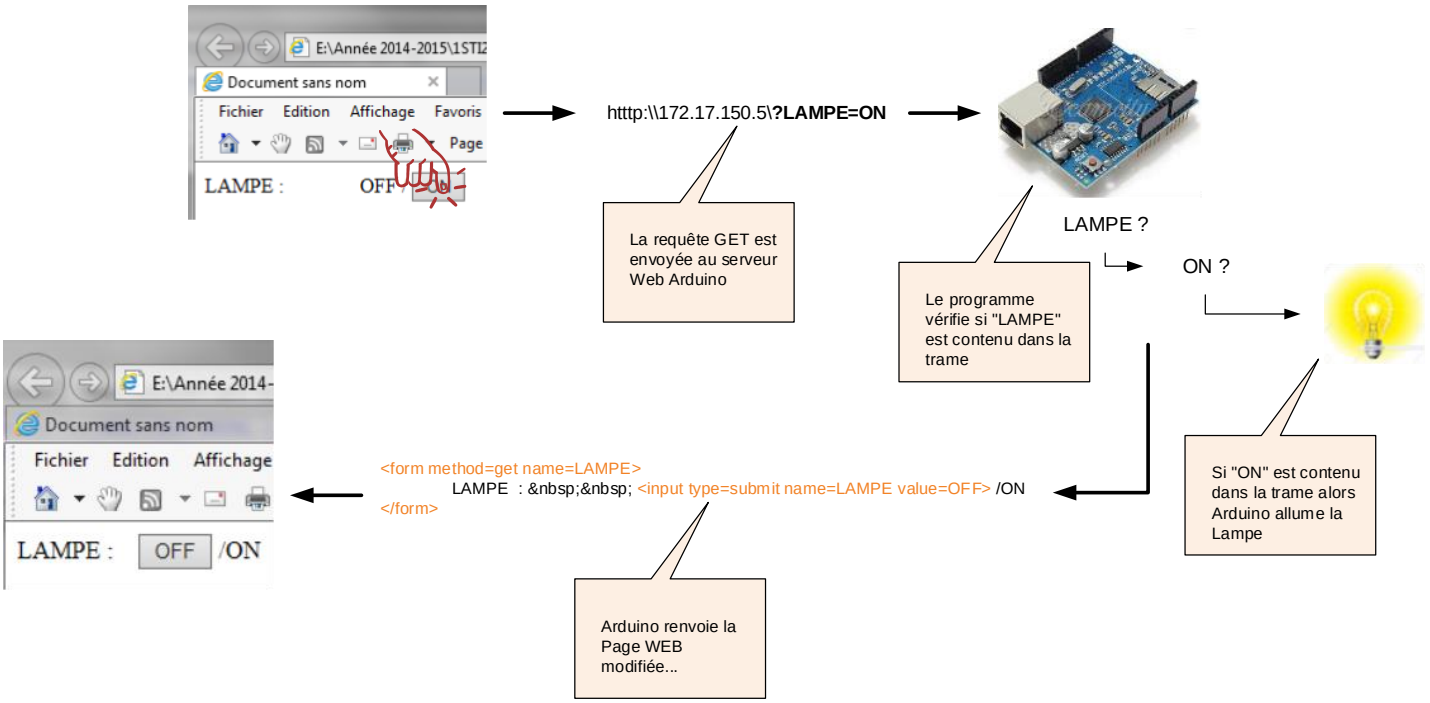
```
SI ReadString contient "LAMPE" ALORS
      SI ReadString contient "ON" ALORS
            On allume la Lampe
            On mémorise l'état par le booléen bLampe = Vrai
      SINON
            On éteint la Lampe
            On mémorise l'état par le booléen bLampe = Faux
      FIN
```

On transmet ensuite la Page Web au Client avec le test de **bLampe** pour modifier la page :

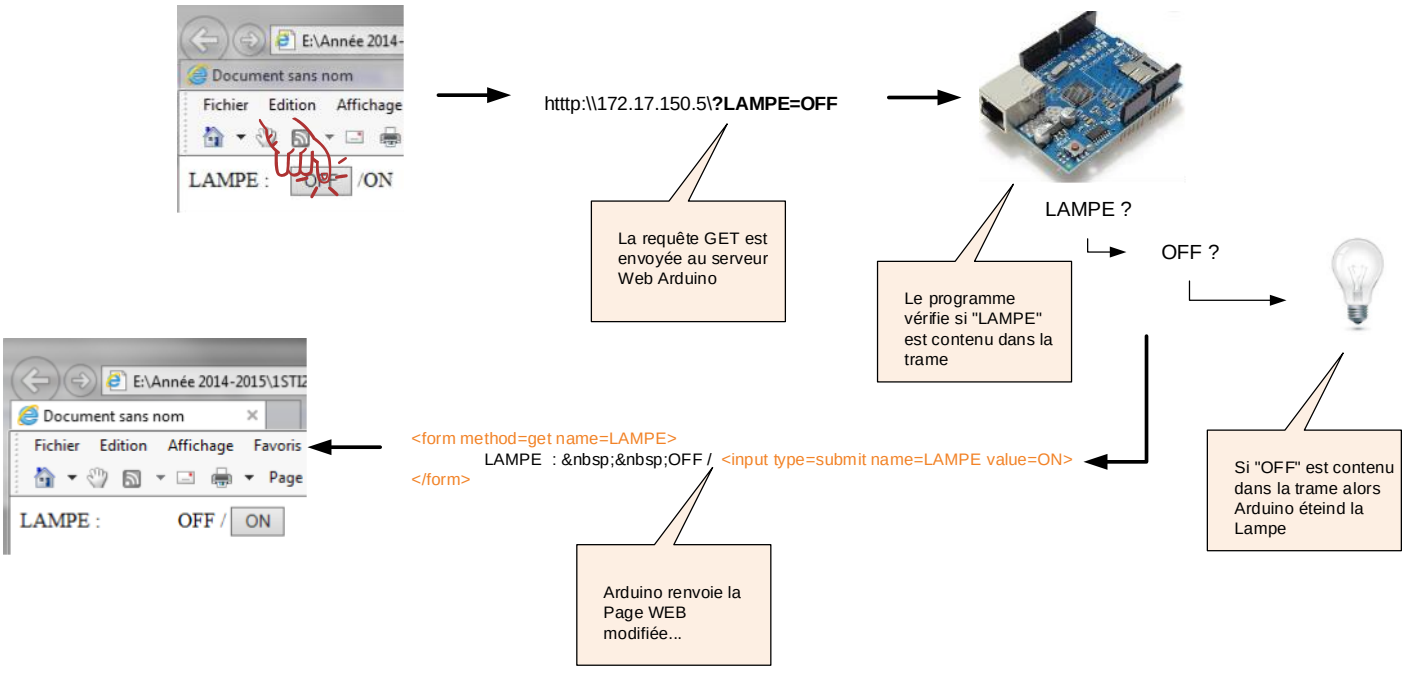
```
SI bLampe = Vrai ALORS
    On envoie "<form method=get name=LAMPE> LAMPE  :     OFF /  <input
               type=submit name=LAMPE value=ON> </form>"
SINON
    On envoie "<form method=get name=LAMPE> LAMPE  :     <input
               type=submit name=LAMPE value=OFF> / ON </form>"
FIN
FIN
```

Le résultat en image :

Allumage de la LAMPE :



Extinction de la LAMPE :



Voici en exemple les deux codes à ajouter pour modifier une interface WEB avec un bouton "**CHAUFFAGE**" pouvant prendre les valeurs ON ou OFF afin de commander le chauffage :

"<form method=get name=BOUTON1>CHAUFFAGE :   <input type=submit name=**CHAUFFAGE** value=**OFF**> / **ON** </form>"

CHAUFFAGE : [OFF] / ON

"<form method=get name=BOUTON1>CHAUFFAGE :   **OFF** / <input type=submit name=**CHAUFFAGE** value=**ON**> </form>"

CHAUFFAGE : OFF / [ON]

À partir des notions expliquées ci-dessus, il vous est possible à partir du programme Arduino d'activer le déverrouillage de l'accès par un simple bouton.

ATTENTION : Il est risqué de tester cette solution sur une Arduino Uno. La mémoire dynamique réservée aux variables temporaires risque d'être insuffisante. Il est préférable d'utiliser l'Arduino **MEGA 2560** bien plus puissante en termes de réservation mémoire et d'entrées/sorties.

♦ TRAVAIL DEMANDÉ

PARTIE 1 : PROGRAMMER UNE PAGE WEB

Vous devez réaliser le programme de base permettant d'afficher une simple interface web embarquée comme représentée ci-dessous :

Utilisez les notions HTML de l'activité 1 pour réaliser le code HTML de cette page avec un éditeur comme NotePad++ ou Dreamviewer, ou tout autre logiciel. Testez dans un premier temps votre codage dans un navigateur. Aidez-vous d'internet et du cours pour le codage de tableaux (balise *table*) ou des caractères spéciaux (é,è,É…) ou des couleurs et polices de caractères.

Une fois au point, ouvrez et complétez le programme ArduinoWebServer_v1.ino puis complétez le programme sur Arduino en respectant **le cahier des charges** et **l'algorithme** donnés. Copiez-collez votre code HTML dans le programme afin de programmer la page Web.

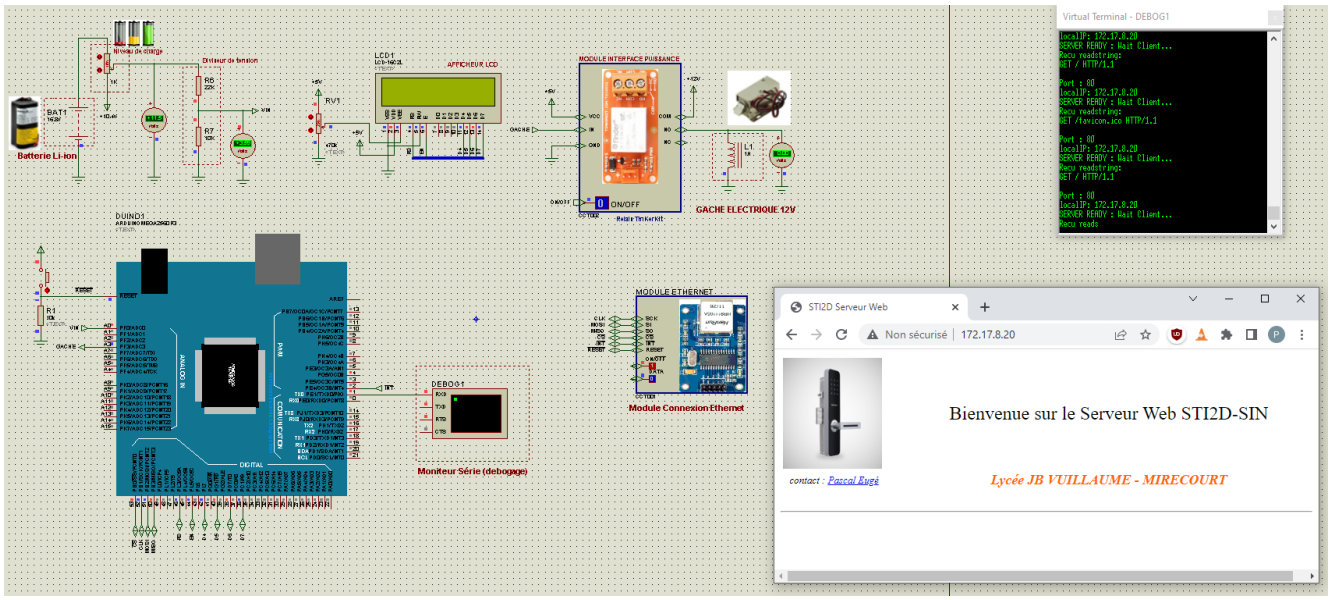
Attention : vous devez double-quotter les " dans le code Arduino, puisque ce caractère est réservé pour la définition de texte dans une chaine (machaine = **"**toto**"**)

Exemple :

```
client.println( " <meta http-equiv= ""Refresh"" content =""5"" /> ");
```

Pour se faire remplacez " par "" dans un éditeur de texte où vous aurez préalablement copié/collé le code HTML de votre page.

Compilez le fichier hexa pour l'importer dans la carte Arduino MEGA de Proteus ISIS (fichier de simulation ServeurWeb_MEGA.DSN) puis avec un navigateur Web testez votre programme en entrant comme lien URL l'adresse IP du module Arduino :



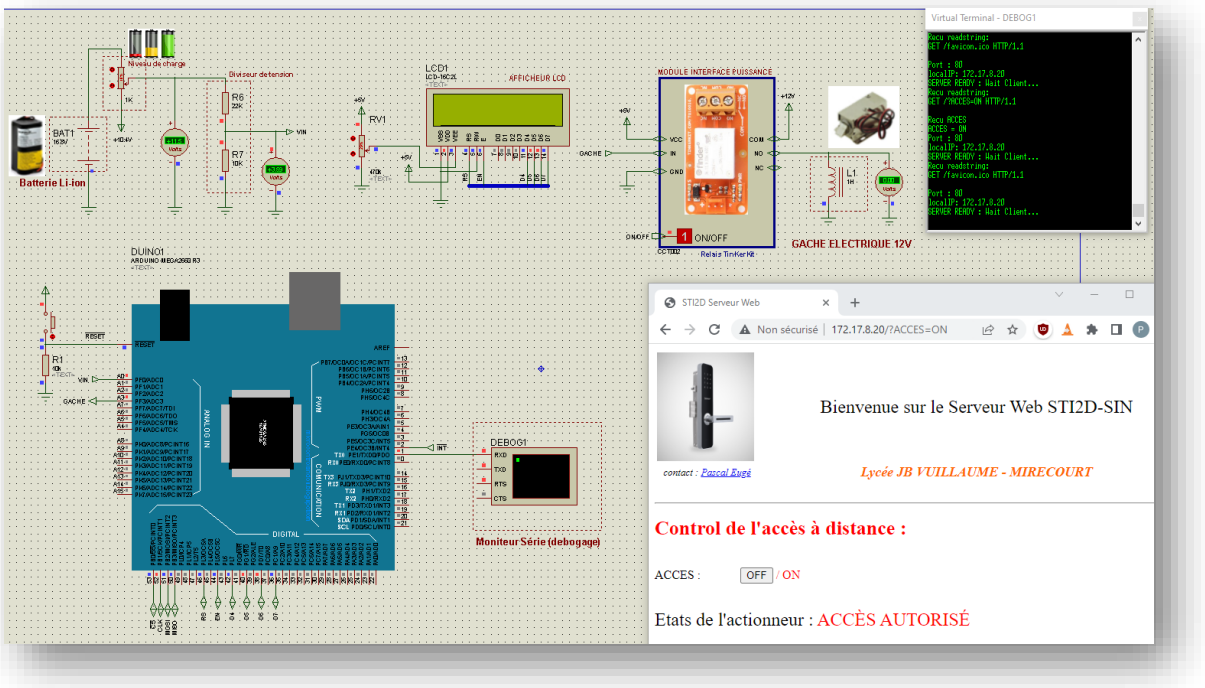
Validation par l'enseignant : /5

PARTIE 2 : GESTION DE L'ACCES

Sauvegardez votre programme en version 2 (ArduinoWebServer_v2.ino) puis joutez à la suite de votre programme la gestion de l'accès par bouton poussoir comme expliqué dans le sujet :

Attention : Avant de transférer au client la page Web, testez si le contenu de la requête envoyée dans la variable `readString` contient "ACCES", puis "ON" ou "OFF" (voir algorithme PARTIE 2). Définissez alors une variable booléenne bACCES pour mémoriser et tester l'état de commande de l'accès.

Compilez le fichier hexa pour l'importer dans la carte Arduino MEGA de Proteus ISIS puis avec un navigateur Web testez votre programme en entrant comme lien URL l'adresse IP du module Arduino. Testez et mettez au point votre programme afin d'obtenir le résultat suivant :

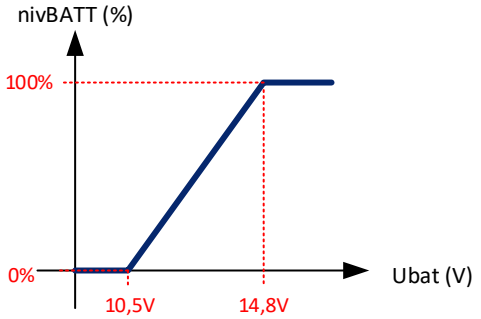


Validation par l'enseignant : /5

PARTIE 3 : GESTION ET AFFICHAGE DU NIVEAU DE BATTERIE

Afin de finaliser entièrement votre interface, vous devez dans cette dernière partie gérer et informer l'utilisateur du niveau de charge de la batterie :

Réinvestissez vos compétences et connaissances vues dans **l'activité 2** de la **SEANCE 2** (*Surveillance du niveau de batterie*) pour la mesure de la tension batterie. À partir de la tension mesurée, **définissez une règle de trois** permettant d'afficher le niveau en % en utilisant la variable nivBATT :



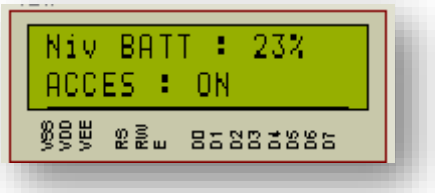
Remarque : Pour une meilleure visibilité du programme, utilisez les onglets "Batterie" et "AffichageLCD" en créant des fonctions qui seront appelées par le programme principal (Loop).

Validation par l'enseignant : /5

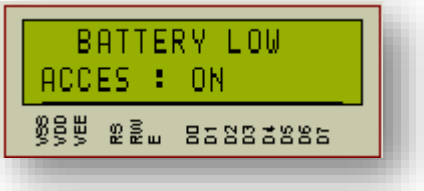
PARTIE 4 : AFFICHAGE LOCAL DES INFORMATIONS

Complétez votre programme en utilisant l'afficheur LCD pour indiquer :

- Le niveau de charge de la batterie
- L'état de l'accès



Lorsque le niveau de charge atteint 0%, l'information "BATTERIE LOW !" doit s'afficher sur l'écran LCD en lieu et place de la valeur en % :



Remarque : Afin d'éviter une lecture et un affichage trop fréquent des informations, il peut être intéressant d'utiliser la fonction TIMER à partir de la librairie `Duinoedu_Utility` :

```
1  #include <Duinoedu_Utility.h>
2
3  void setup()
4  {
5  }
6
7  void loop()
8  {
9    ONLY_EVERY(5000,_ABVAR_1_Repeat)
10      //placez votre code ici
11   END_ONLY_EVERY
12
13  }
```

Validation par l'enseignant : /5

Vous pouvez également valider votre programme en utilisant le navigateur Web d'un autre ilot, puisque le module Proteus ISIS communique en réseau grâce au module ENC28J60 et à la librairie UIPEthernet !