La Souris HTML D17 (version 17sept2020)

Cette doc présente la souris HTML de la centrale DCC D17.

Cette souris permet de contrôler les locomotives, affiche des TCOs pour commander les accessoires comme les aiguillages et permet d'exécuter des « scripts » afin d'automatiser des séquences de jeu.

Cette souris est en fait un script écrit en javascript qui réside dans une page web délivrée par le serveur web http de D17. Une fois la souris chargée dans votre navigateur, elle communique avec la centrale D17 par websockets. La librairie de l'esp8266 accepte 5 websockets, il peut donc y avoir 5 souris HTML.

Cette approche présente de nombreux avantages. Le premier est que la souris peut tourner sur n'importe quel appareil qui possède un navigateur web (qui supporte les websockets ce qui est le cas des navigateurs depuis plus de 10 ans). Il est ainsi possible d'utiliser la souris sur téléphones (Android et surement Apple (mais je n'ai pas testé), tablettes (Android et surement Apple), PC (sous Windows, Linux, surement MAC) … Le second avantage est qu'il n'y a rien besoin d'installer sur l'appareil se connectant à D17. Il suffit d'ouvrir le navigateur, tapez l'adresse de D17 et la page avec la souris dedans se charge comme une page web « normale ».

Qu'attendons-nous ? Connectons-nous à D17. Dans la configuration par défaut D17 agit comme un point d'accès WIFI. Le nom par défaut est D17-0001 et le mot de passe ulysse31. Vous les aurez surement changés. Il est à noter qu'il est possible de faire des configurations plus complexes comme D17 qui se connecte en temps qu'appareil à votre box internet ou WIFI mais restons simple pour l'instant. Activez le WIFI puis connectez-vous sur le réseau WIFI de D17. Sur certains vieux téléphones activer le WIFI coupe le cellulaire, mais sur les récents les 2 peuvent cohabiter. Il est fort probable que votre appareil vous indique que le réseau ne permet pas d'accéder à internet, c'est normal il permet d'accéder à D17 uniquement.

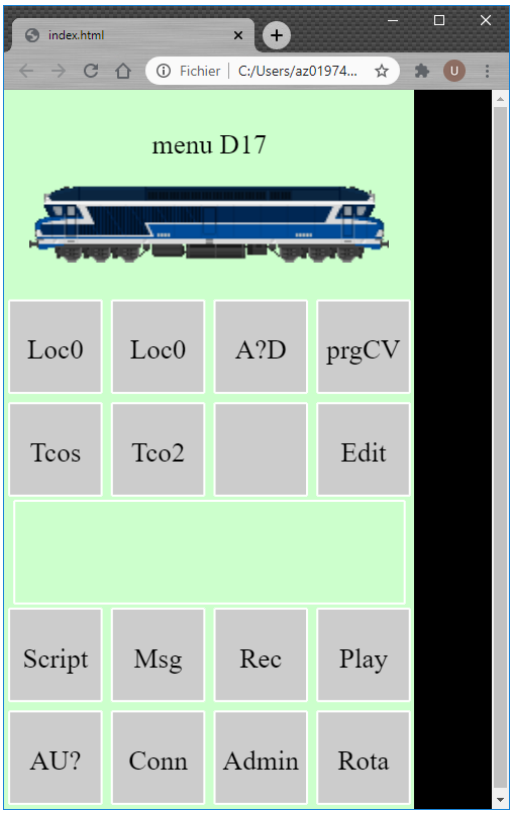
Ouvrez votre navigateur favori et tapez l'adresse de D17 : http://192.168.4.1 . 192.168.4.1 est l'adresse IP par défaut du point d'accès WIFI de D17. http indique que l'on fait une requête http pour avoir une page web. Le port IP n'est pas spécifié, le port standard http par défaut est utilisé (c'est le 80). La page n'étant pas spécifiée, c'est la page par défaut qui est servie (index.html).

Vous devriez voir s'afficher le menu principal de D17, bravo !!!!

Il est à noter que la page n'est servie que si la centrale est en mode AU (arrêt d'urgence = PWR OFF) ce qui est le cas au démarrage. La raison est que servir une grosse page web peut prendre du temps, temps pendant lequel la centrale ne génère plus le signal DCC ce qui peut avoir pour conséquence de faire passer les locos en mode analogique et de les voir toutes partir à fond … Il faut donc passer en AU avant de connecter une nouvelle souris sinon une erreur s'affichera à la place de la page. Vous pouvez si besoin, contrôler l'AU de la centrale avec les requêtes suivantes : http://192.168.4.1/au1 pour passer en AU (PWR OFF) ou http://192.168.4.1/au0 (PWR ON). Il n'y a pas de page web en retour à ces requêtes.

Ne pas utiliser le bouton de retour du navigateur qui redemande la page web, utiliser les liens de la page D17. Tout est géré à partir d'une seule page web pour ne consommer qu'un seul websocket

1) Le menu principal



La souris s'adapte automatiquement à la taille de l'écran tout en conservant la géométrie (les carrés restent des carrés). Cela explique la présence d'une bande noire à droite ou en bas. Sur certains appareils et navigateurs, le fait de faire pivoter l'appareil tourne la fenêtre qui passe du mode portrait au mode paysage et vice versa. A vous de choisir la position la plus agréable. Si ce n'est pas le cas, vous pouvez utiliser le bouton Rota afin de faire pivoter l'affichage.

**

La ligne du bas contient le bouton d'AU (rouge si AU (=PWR_OFF), vert si pas AU (=PWR_ON=PWR), gris si non connu (si la centrale n'est pas connectée). Le bouton Conn en vert indique que la souris est connectée par websocket. En cas de déconnection ou à partir de la 6eme souris qui essaie de se connecter il est gris. Appuyer dessus connecte / déconnecte la souris du websocket. Le bouton Admin accède à une page web permettant de mettre à jour les fichiers du SPIFFS de l'esp8266. Vous pouvez par exemple mettre à jour la souris en remplaçant le fichier index.html, ajouter des images, mettre d'autres page web … Pour info, le SPIFFS qui est la mémoire du système de fichier de l'ESP8266 (comme une clés USB) à une taille de 3MB. Evitez donc de mettre de trop gros fichiers.

**

L'image de la loco en haut est juste là pour faire jolie pour l'instant. Si la CC72000 ne vous plait pas, vous pouvez cliquer dessus pour afficher une CC6500 ou une 141R. Si cela ne vous plait toujours pas, remplacer accueil.gif par une image qui vous plait …

**

La première ligne de bouton est dédiée aux locos. Le premier bouton « Loc » affiche une souris pour contrôler une loco. Le second affiche une seconde souris. Avoir 2 souris est pratique pour certaines manœuvres. Le numéro indique l'adresse de la loco sélectionnée. Un bouton gris et 0 indiquent qu'aucune loco n'est sélectionnée. Un bouton jaune et 1 indique que la souris est utilisée pour un loco analogique. Un buton vert et une adresse de 2 à 9999 indiquent l'utilisation d'une loco DCC.

Le bouton vert « Dcc » indique que la centrale est en mode DCC. Lorsque la centrale est en analogique le bouton est jaune et comporte la mention « Ana ». Lorsque la souris n'est pas connectée le bouton est gris avec la mention « A?D ».

Le bouton « progCV » permet de programmer les CV des décodeurs des locomotives. Pour rappel les CV sont les variables de configuration des décodeurs des locomotives. Par exemple le CV1 contient l'adresse de la locomotive, le 2 permet de fixer le point de démarrage, le 3 l'accélération… Attention toutes les locomotives alimentées seront programmées. C'est pourquoi un code de sécurité est demandé. Il est spécifié dans D17 et vaut 1234 par défaut

**

La seconde ligne de boutons est dédiée aux TCOs. Le premier bouton « TCOs » affiche la liste des 10 TCOs. Vous pouvez alors en sélectionner 1. Le second bouton « TCO1-10 » permet d'accéder directement au sélectionné qui est souvent le dernier TCO utilisé. Enfin, le dernier bouton « Edit » est une aide pour créer facilement des TCOs.

**

Le « rectangle vide » permet à la souris d'afficher des messages. Il est par exemple utilisé par les scripts pour afficher des messages.

**

La 3eme ligne de boutons est dédiée à l'automatisation. Le bouton « Script » permet d'accéder au menu des scripts pour lancer un script ou les arrêter. Il est vert lorsqu'au moins un script est en cours d'exécution.

Le bouton « MSG » permet d'envoyer des commandes à la centrale D17 en respectant le protocole D17. Par exemple t3/ passe l'aiguillage 3 en dévié. Il est utilisé pour la mise au point.

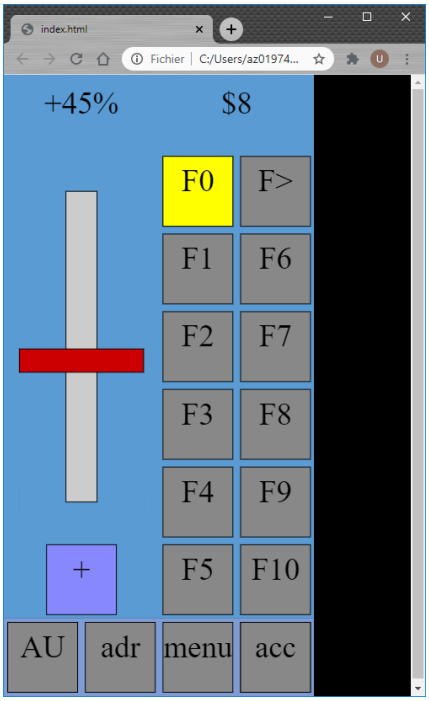
Le boutons « REC » lance un enregistrement. Toutes les commandes envoyées à D17 sont alors enregistrées (de même que l'état des entrées). En enregistrement le bouton est vert. Un second appuie stoppe l'enregistrement et affiche toutes les commandes envoyées et les changements des entrées. Le bouton est alors jaune. Un dernier appuie cache les changements et le bouton redevient gris.

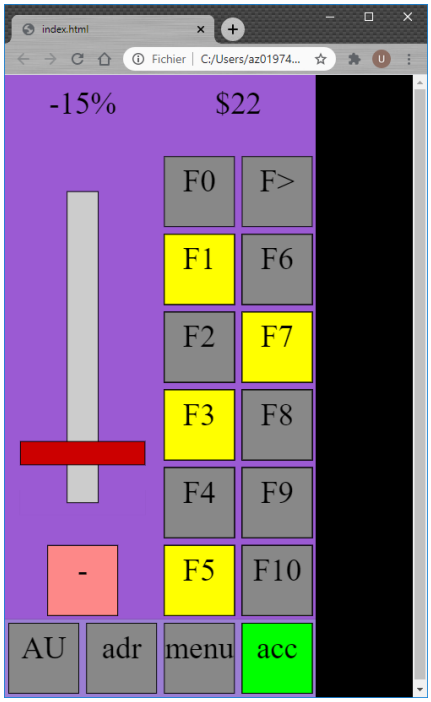
Le bouton « PLAY », permet de réexécuter ce qui a été enregistré avec le bon timing en se synchronisant sur le changement des entrées. Il est ainsi possible par exemple d'enregistrer un tour du réseau avec un train puis de le rejouer. Cette fonctionnalité est encore expérimentale, lorsque ça marche c'est bluffant, mais cela ne marche pas encore dans tous les cas et il n'y a pas encore de sauvegarde.

**

Dans la barre d'adresse apparait c:\... car j'ai lancé la page web de mon PC pour écrire cette doc et mettre au point mes TCOs. En vous connectant sur D17, vous devriez avoir : http://192.168.4.1 ou http://192.168.4.1/index.html

2) Les locos





Le potentiomètre permet de régler la vitesse de la locomotive sélectionnée. Elle est indiquée en pourcentage (0-100%). Le bouton « + » ou « - » permet de changer de sens. L'affichage du + indique que la loco est en marche avant et inversement.

Le bouton « AU » pour arrêt d'urgence coupe l'alimentation en sortie du booster.

Le bouton « acc » pour accéléromètre permet de conduire la loco en penchant la tablette. Bien entendu, il faut que l'appareil et le navigateur supportent cette fonctionnalité. C'est une fonction marrante 😉

Le bouton « adr » permet de choisir la loco à conduire. 0 pour aucune loco, 1 pour les locos analogiques, 2-9999 pour les locos DCC. Pour rappel le choix entre analogique et DCC se fait maintenant par le bouton « DCC/ANA » du menu principal. En DCC seules les souris avec des locos d'adresse 2-9999 auront en effet, en analogique seule les souris avec l'adresse 1 auront un effet. Chaque fois que l'utilisateur change quelque chose, une commande est envoyée à D17. Si plusieurs souris utilisent la même adresse, le dernier changement gagne. Les souris modifient D17, mais l'inverse n'est pas vrai. Plusieurs souris avec la même adresse n'afficheront donc pas nécessairement les mêmes informations.

Les boutons « F0-F28 » changent l'état des fonctions auxiliaires. Par exemple F0 contrôle souvent les phares. « F> » permet d'afficher F1-F10, F11-F20 et F21-F28. F0 est toujours affiché.

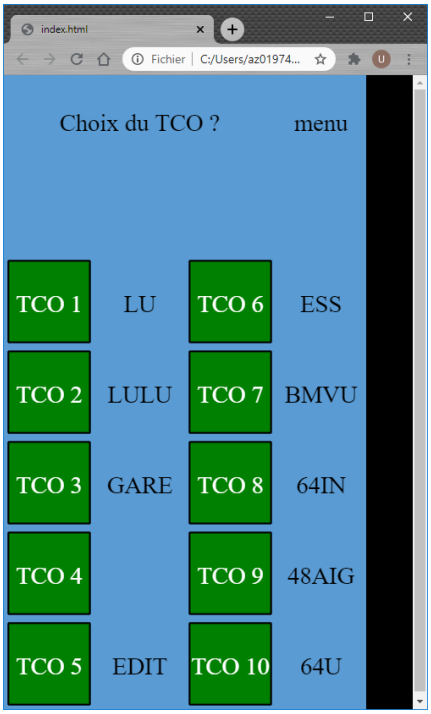
Le bouton « menu » permet de revenir au menu principal.

Tout en haut sont affichés la vitesse en % et l'adresse. Appuyer sur la vitesse passe d'une souris à l'autre permettant de manœuvrer facilement 2 locos. Les 2 souris ont des couleurs de fond différentes afin de les identifier rapidement. Cliquer sur l'adresse affiche le dernier TCO utilisé permettant de passer rapidement de la conduite des locos au control des accessoires comme les aiguillages. Il est possible de placer sur les TCOs une commande pour revenir à la dernière souris des locos utilisée. Il est également possible de mettre une souris dans un onglet du navigateur, voire 2 souris dans 2 onglets et le TCO dans un autre, mais cela consomme un websocket par onglet (sur les cinqs) et à l'utilisation c'est plus lent…

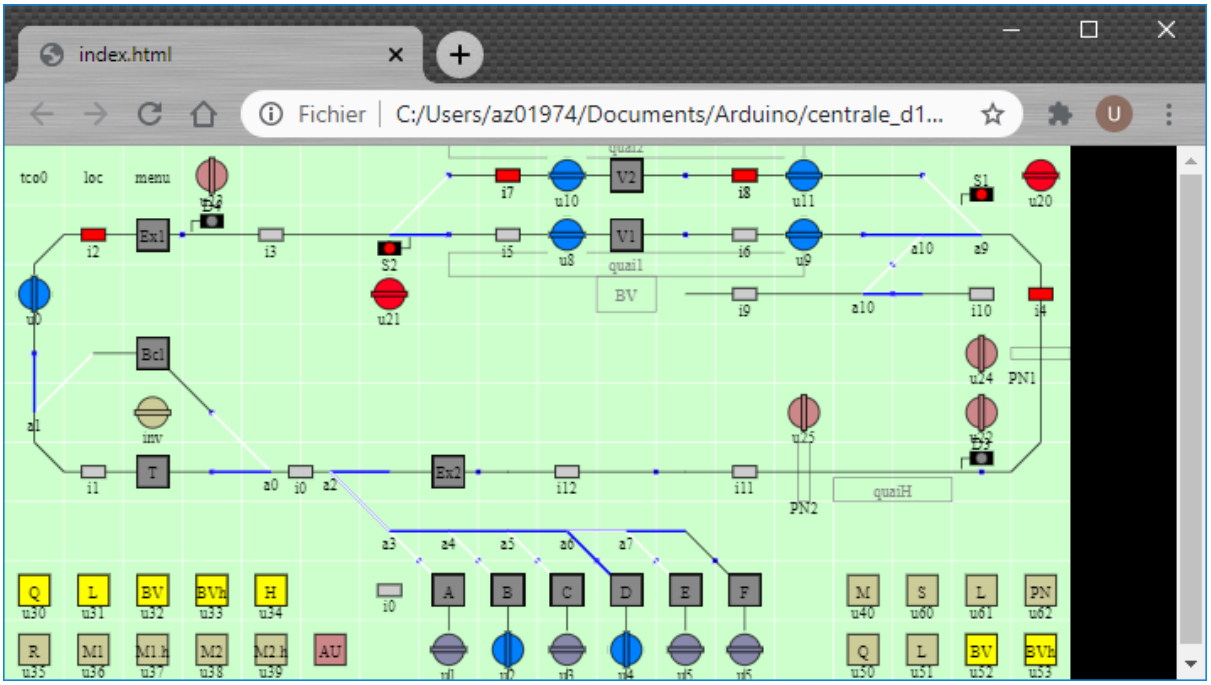
Je serai amené à proposer prochainement d'autres souris de locos, comme une souris double sur la même page, ou une avec un accélérateur et un frein …

3) Les TCOs

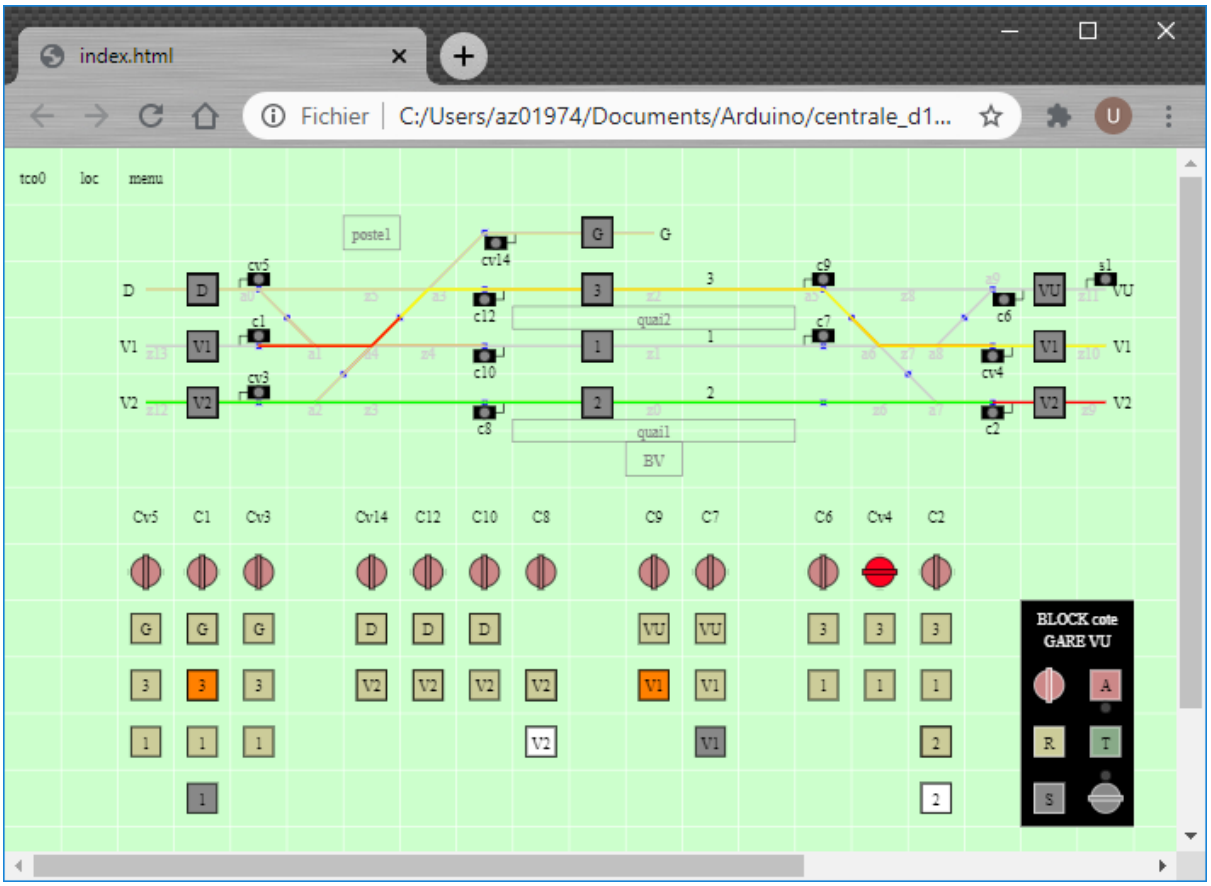
Voici quelques exemples de TCOs (livrés comme exemples que vous remplacerez par les vôtres) :



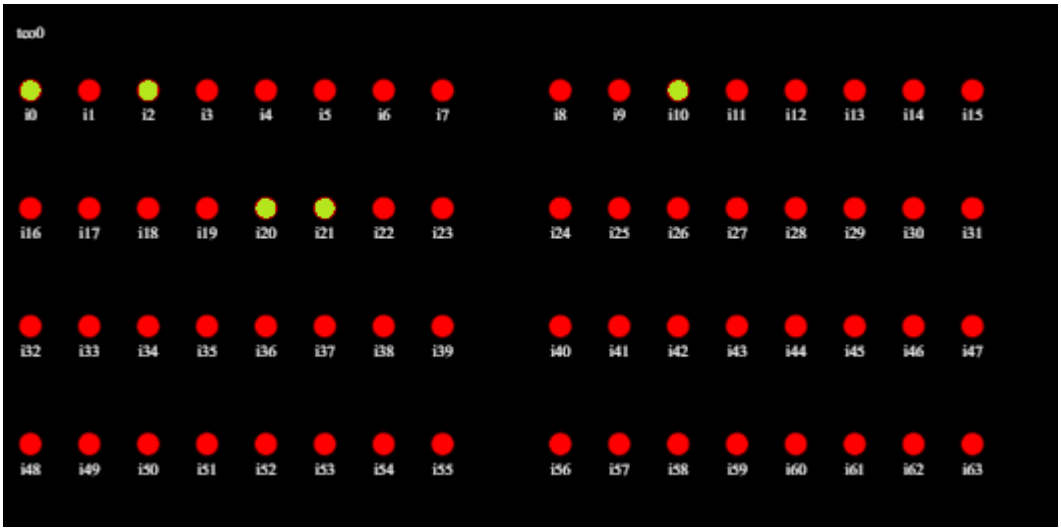
* TCO 0 qui permet de sauter vers un des 10 autres TC0s (1 à 10)



* TCO de l'ensemble d'un réseau. Ce TCO de modélisme n'est pas réaliste, mais très utile pour piloter simplement un réseau. Celui-ci est assez dense pour être utilisable sur l'écran d'un téléphone portable



*TCO assez réaliste d'un poste d'aiguillage PRS (Poste tout Relais à transit Souple). On appuie sur les boutons (oranges) pour établir des itinéraires à destruction automatique DA ou sur les boutons blancs pour des itinéraires à tracé permanent TP. Les sélecteurs permettent de fermer les carrés en cas de problèmes et reprendre les itinéraires. La partie TCO affiche les occupations (rouges), DA (jaunes) et TP (vert), elle est plus en technologie informatique PIPC, mais rien n'empêche de mettre des voyants à la place. Ce TCO est aussi un poste PRG (Poste tout Relais Géographique). Il suffit d'appuyer sur l'origine et la destination de l'itinéraire (ex : V1 puis 3) pour établir un itinéraire. Cela met à jour le bouton PRS correspondant. Enfin un BMVU permet de gérer les circulations sur la voie unique.



*TCO de debug pour afficher l'état des 64 premières entrées de rétrosignalisation

Le programme peut afficher un des 10 TCOs (TCO1-TCO10). Le TCO0 permet de choisir quel TCO afficher. Chaque TCO est composé de cellules sur lesquelles on peut mettre des widgets (lignes(rails), interrupteurs, voyants). Leur apparence (couleur ...), peut dépendre de l'état des entrée, position des aiguillages et variables u. Par exemple colorier en rouge une ligne si un train est détecté. Le TCO permet aussi d'envoyer des commandes à la centrale lorsque l'on clique ou touche une cellule. Le protocole D17 qui est utilisé permet de modifier la position des aiguillages, les leds, sorties et sorties pwm/servo de la centrale ainsi que d'envoyer des ordres aux décodeurs d'accessoires. Voir la doc sur le protocole D17 pour plus d'information

```
i0-95 -->        -->x<1-511>.<0-7>
d0-47 -->        -->y<1-2044>
e0-47 -->  TCO   -->l0-127,c0-127,h0-127
u0-95 <->        -->s0-47,p0-47
t0-47 <--        -->o0-5
```

La centrale envoie en permanence à la page web les variables :

- **i0-95** (input = entrée). Les entrées sont des capteurs (interrupteurs, capteurs de courant) à relier sur des modules S88 du bus de rétrosignalisation.
- **d0-47** et **e0-47** (position des aiguillages directe et évitement). Il y a 2 variables car l'aiguillage peut être en manœuvre (donc ni en direct, ni en évitement). Suivant comment sont gérés les aiguillages, il se peut que la centrale ne connaisse pas la position des aiguillages à l'allumage avant d'avoir envoyé une commande pour les manœuvrer.
- **u0-191**. Ces variables « utilisateur » vous permettent de faire ce que vous voulez. Par exemple vous pouvez décider que u10 allumera un lampadaire, u122 déclenchera un itinéraire. Comme ces variables sont envoyées en permanence, l'ensemble des TCOs (sur chaque appareil est au courant de chaque modification)

Dans le sens inverse, le TCO peut envoyer à la centrale des commandes

- **t0-47**. (t=turnable=aiguillage) pour modifier la position d'un aiguillage. Ex t0- (aiguillage 0 en direct), t22/ (aiguillage 22 en dévié (évitement)), t33^ (aiguillage 33, changement de position)
- **u0-191** pour modifier une variable u
- D'autres ordres pour les leds et sorties pwm/servos de la centrale ou décodeurs d'accessoires. Mais il est préférable d'utiliser les variable u et transformer en ordre la demande de modification de la variable u (pour les leds et sorties pwm/servos de la centrale ou décodeurs) dans le programme de la centrale. L'avantage de passer par les variables u est de pouvoir les utiliser dans les TCOs (par exemple pour colorier un bouton) et ceci en étant à jour sur tous les TCOs des différents appareils. Par exemple en comandant une led directement avec l33^ (changement (allumage ou extinction) de la led 33), les TCOs ne savant pas que cette commande a été envoyée. Avec une variable u (ex u28^), le code de la centrale (en réaction à u28^pourra changer l'état de la led). L'état de la led sera visibles dans tous les TCOs car u28 aura été mis à jour.

Quelques exemples :

Lorsque l'on veut modifier la position de l'aiguillage 0, on envoie la commande t0^. Si on veut le mettre en direct t0- et en dévié t0/. Niveau centrale, la fonction suivante est appelée :

```
void user_notify_aig(byte num, byte cmd) // cmd=0=direct cmd=1=devie
{
    // gerer vos aiguillages ici
    if(num==0)
    {
        if(cmd==0) user_bas_acc_dec_tx(10,0,1);
        if(cmd==1) user_bas_acc_dec_tx(10,1,1);
    }

    user_set_d_e(num, 1 - cmd, cmd);
```

```
    }
```

A vous de gérer vos aiguillages qui peuvent être sur un décodeur, un servo de la centrale ...
Dans cette exemple l'aiguillage 0 est branché sur un décodeur d'accessoires basique d'adresse 10 sur les sorties 0 et 1
user_set_d_e(num, 1 - cmd, cmd); met à jour les variables d pour direct et e pour évitement(dévié) que la centrale envoie régulièrement aux TCOs afin qu'ils les utilisent pour l'affichage

Le même principe est utilisé pour les variables u

```
    void user_notify_u(byte num, byte val)
    {
        if(num==3) { user_led(63,val); }

        if(num==10 && val==1)
        {
              if(user_get_u(11)==0 && user_get_u(12)==0)
              {
                  user_notify_aig(1,0);
                  user_notify_aig(3,1);
                  user_notify_aig(5,0);
             }
         }

        user_set_u(num, val);
    }
```

La première partie contrôle la led 63 par la variable u3.
(On aurait pu utiliser la commande l63, mais dans ce cas les autres TCOs ne verraient pas le changement)

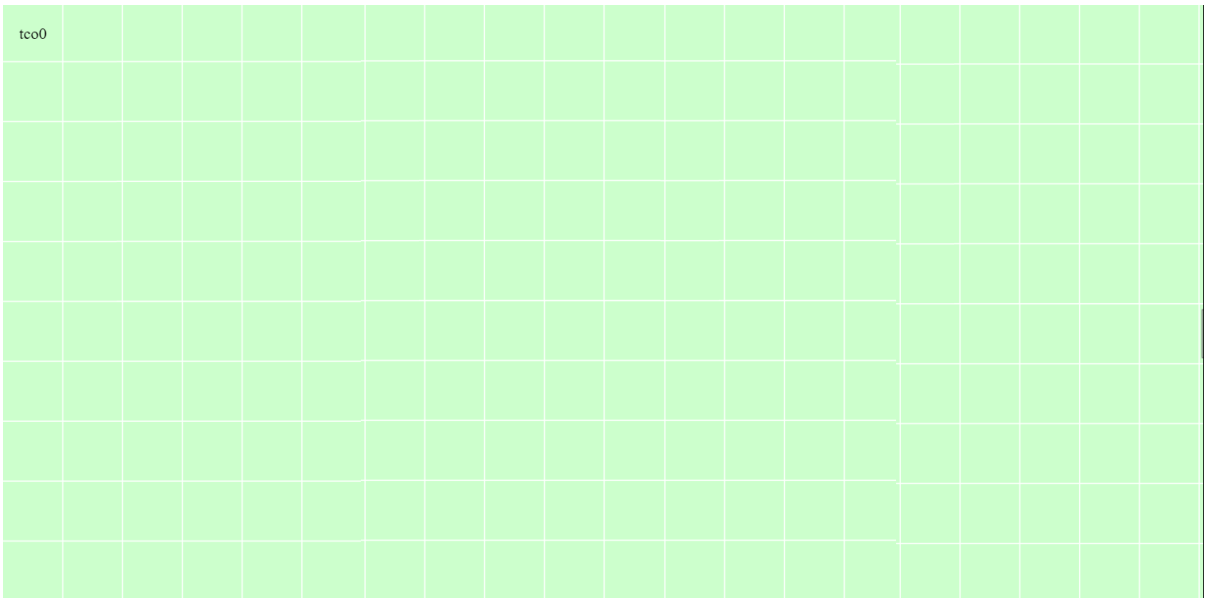
La seconde partie, réalise un itinéraire simplement. u10 met l'aiguillage 1 en direct, 3 en dévié, 5 en direct. Cela seulement si les itinéraires incompatibles u11 et u12 ne sont pas déjà actifs.

La dernière ligne met à jour la variable u qui est envoyé aux TCOs afin qu'ils les utilisent pour l'affichage.

Tous les TCOs de tous les appareils peuvent voire les entrée i la position des aiguillages d,e et les variables u. Alors qu'il n'y a aucun moyen de voir et partager les autres commandes.

Maintenant que nous maitrisons les variables, créons les TCOs. Les TCOs sont définit dans les fonctions tco1_init() à tco10_init(). Il suffit d'ajouter des widgets (composants graphiques) dans les cellules des TCOs. En plus, le TCO0 permet de choisir le TCO à afficher. Ces fonctions se trouvent dans la page web que délivre D17, c'est-à-dire dans le fichier index.html. Ces widgets sont des fonctions à appeler. On distingue les fonctions de base (par exemple pour créer un bouton, une ligne(rail)), les fonctions composées (ex un bmvu composé de plusieurs boutons), les fonctions de l'éditeur qui sont des fonctions composées et vos fonctions que vous pouvez créer pour vous faciliter la tâche. Si la fonction est créé à l'intérieur de la fonction init du TCO alors elle sera utilisable uniquement dans ce TCO, si elle est créée en dehors, elle sera utilisable par l'ensemble des TCOs.

Passons à la pratique avec l'exemple suivant :




```
function tco6_init()
  {
    scale(20,10);
    add_fond(0, 0, 20, 10, color_fond, color_grille);
    add_cmd(0,0,"tco0");
    draw_text(0,0,"ret"); // ret pour retour au tco0
  }
```

La fonction scale demande la création d'un TCO de 20 cellules de large sur 10 cellules de haut. Le programme se débrouille pour ajuster la taille du TCO à la taille de l'écran
add_fond(0, 0, 20, 5, color_fond, color_grille); met un fond à partir de la cellule (0,0) (en haut à gauche) sur 20x5 cellule. Le fond a la couleur color_fond (qui est une constante) et une grille color_fond.

Les couleurs sont définies au format '#RRVVBB' avec RR=rouge, VV=vert, BB=bleue (0=0%, 10, 20, ... 80=50%, 90, A0, .. F0 .. FF=100%) . Ex: '#FF0000" = rouge 100%, '#FFFF00' = jaune(rouge+vert) 100%
Afin de rendre l'utilisation plus facile il est possible de définir des constantes comme cela est fait au début du programme comme: var color_fond = '#ccffcc'; // vert pale
Vous rajouter des constantes pour d'autres couleurs.

Les cellules sont repérées par 2 coordonnées qui commencent à 0. Le premier est pour l'horizontale et le second vertical.
(0,0) = en haut à gauche
 (19,9) =en bas à droite pour un scale de (20,10)

add_cmd(0,0,"tco0"); envoie la commande tco0 lorsque l'on clique sur la cellule (0,0). Cette commande n'est pas envoyée à la centrale mais change de TCO. Il est important de le faire sinon vous ne pourrez plus sortir du TCO.

add_cmd(0,1,"t3^"); demande de changer la position de l'aiguillage3

add_cmd(10,2,"u5^"); demande de changer l'état de la variable u5

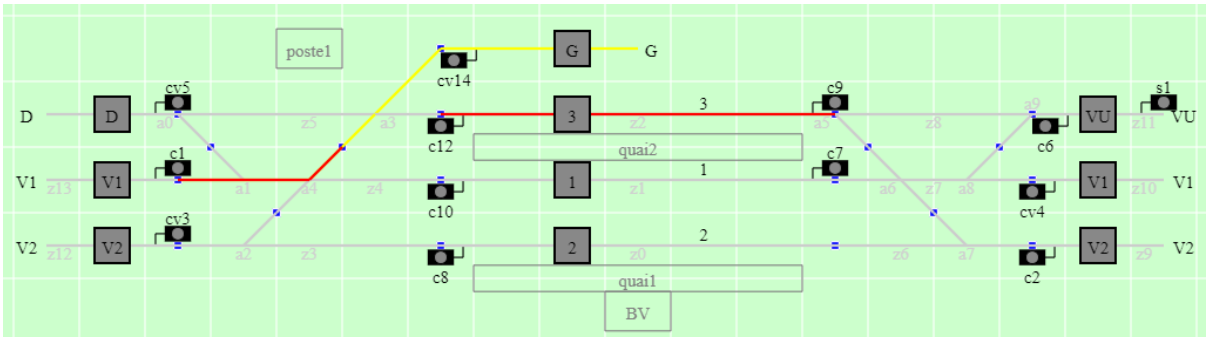
Mais la plupart du temps, les commandes sont directement spécifiées dans les widgets

Maintenant que nous avons un TCO, nous pouvons rajouter des widgets avec les fonctions suivantes. Les fonctions de base proposent de nombreux paramètres afin de convenir à tous les cas. Il est fastidieux et peu claire de tous les mettre. C'est pourquoi si vous ne spécifiés pas tous les paramètres, ceux par défaut (avec le =) seront utilisés. Il vous est également possible de créer vos propres fonctions composées.

---

function **add_line**(x1, y1, x2, y2, size=1, col_off='black', list_svar1="", col1='black', list_svar2="", col2='black', list_svar3="", col3='black', isza=0)
Ajoute une ligne. Par exemple pour représenter des rails.
x1,y1 = cellule de départ
x2,y2 = cellule d'arrivée
size = épaisseur de la ligne en pixels
col_off = couleur (noir par défaut)

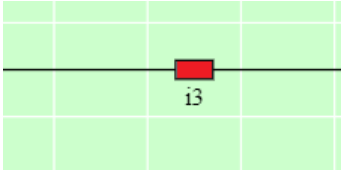
Il est possible de changer la couleur de la ligne si une des variables des liste est vraie
col_off par défaut
col1 si une des variables de list_svar1 est vraie, ex: "u0,u2,u3"
col2 2
col3 3
la couleur3 est la plus prioritaire
col1 est souvent utilisée pour les iti DA (Destruction automatique (souvent jaune))
col2 est souvent utilisée pour les iti TP (Destruction automatique (souvent vert))
col3 est souvent utilisée pour les détections
si is_za=1 (utilisée pour les zones d'aiguillages), la couleur3 d'occupation est uniquement affiché s'il y a au moins une variable vraie dans les listes 1 et 2 (c'est-à-dire un itinéraire actif passant par cette ligne).



* ex avec un itinéraire en destruction automatique (DA) sur la zone d'aiguillages entre V1 et G avec un train détecté sur la première partie. Un autre train est détecté sur la voie 3 qui n'est pas une zone d'aiguillages.
Ex : add_line(1, 1, 5, 1, 1, 'black', "u20,u21", '#ffff00', "u30", '#00ff00', "i2", '#ff0000', 1)

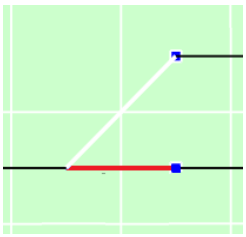
Exemple plus simple d'une ligne qui chnage de couleur en cas de détection d'un train sur i10
add_line(2,5,12,5,1,'black', "i10", '#ff0000');

Il est bien entendu aussi possible de faire des lignes à couleur fixe. Pour détecter les trains, juste mettre un voyant.



Il n’y a pas de fonction de base pour dessiner un aiguillage, en effet il suffit de le créer avec 2 lignes. Certains feront des lignes qui se colorient en fonction de la position des aiguillages, d’autres préféreront utiliser des lignes de couleur fixes et mettre des viyants.
Exemple d’un aiguillage dont la branche active s’allume en rouge et qui change d eposition lorsque l’on clique dessus :

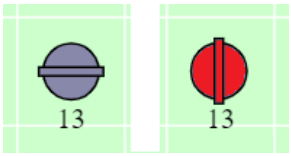
```
add_line(2,5,3,5,3,’black’, "d10", '#ff0000');
add_line(2,5,3,4,3,’black’, "e10", '#ff0000');
add_cmd(2,5,"t10^");
```



---

function **add_sel**(x, y, txt, cmd, varStr, vert_on, col_off, col_on, col_border, col_bg, col_txt)
Affiche un sélecteur (bouton rotatif)

- x,y = cellule
- txt = text à afficher sous le sélecteur
- cmd = commande à envoyer lorsque l'on clique dessus, ex: "u0^"
- varStr = variable utilisée pour l'affichage ex: "u0"
- vert_on: 1 = bouton vertical quand la variable est vraie, 0 = horizontal
- col_off = couleur quand la variable est fausse
- col_on = couleur quand la variable est vraie
- col_border = couleur des bords (‘black’ par défaut)
- col_bg = couleur du fond sur lequel est posé le bouton (color_fond par défaut)
- vol_txt = couleur du texte (‘black’ par défaut)

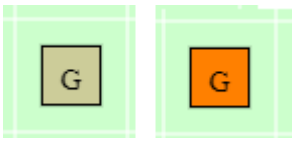


add_sel(1,2, "13","u0^","u0",1,"#888888","#ff0000") ;

---

function **add_bp**(x, y, txt, cmd, varStr, col_off, col_on, col_border='black', col_txt='black')
ajoute un bouton poussoir

- x,y = cellule
- txt = text à afficher dans le BP
- cmd = commande à envoyer lorsque l'on clique dessus, ex: "u0^"
- varStr = variable utilisée pour l'affichage ex: "u0"
- col_off = couleur quand la variable est fausse
- col_on = couleur quand la variable est vraie
- col_border = couleur des bords (‘black’ par défaut)
- vol_txt = couleur du texte (‘black’ par défaut)



add_bp(1,2,"G","u0^","u0",color_orange_off, color_orange_on) ;

*color_orange_on et off sont définies en début de fichier :
var color_orange_off = "#cccc99"
var color_orange_on = "#ff8000"

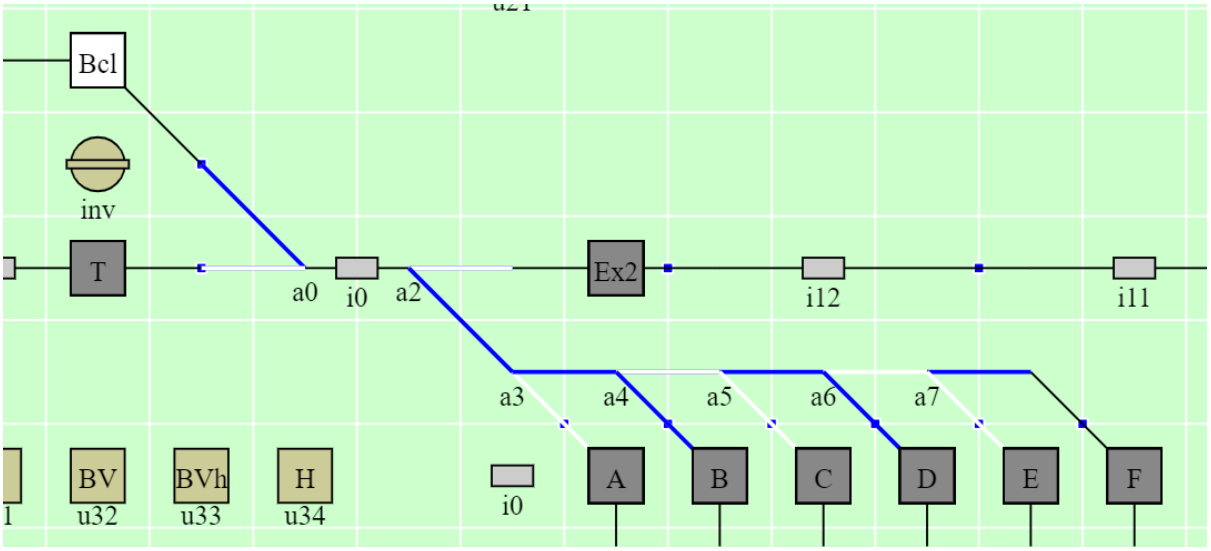
---

function **add_geo**(x, y, txt, num)
Ajoute un bouton géographique pour les TCO de type PRG
x,y = cellule
txt = text à afficher dans le BP
num = numéro du bouton géographique

Les boutons géographiques vont par 2, on appuie sur un premier (par ex en début d'itinéraire) qui s'allume en blanc, puis sur un second (par ex en fin d'itinéraire). La séquence d'appuie déclenche l'envoie d'une commande ajoutée avec la fonction suivante :

add_geo_cmd(geo1, geo2, cmd);

ex: add_geo_cmd(1,3,"u2^"); // l'appuie sur le géo 1 puis 3 demande le changement de l'iti u2
ex: add_geo_cmd2(5,3,"t2-t3/"); // l'appuie sur le géo 5 puis 3 (ou l'inverse grâce à cmd2 à la place de cmd) positionne l'aiguillage 2 en direct et 3 en dévié.

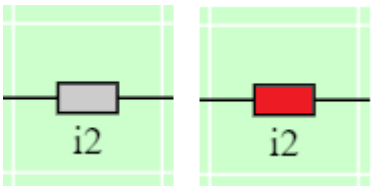


* J'appuie sur Bcl puis B pour positionner les aiguillages entre Bcl et B

---

function **add_voy**(x, y, typeStr, dx, dy, varStr, col_off, col_on, col_border='black', txt="", col_txt='black')
ajoute un voyant
x,y = cellule
typeStr: 'o' rond, 'r' rectangle
dx,dy = taille du voyant (1=100% de la cellule, 0.4=40%)
varStr = variable utilisée pour l'affichage ex: "i0"
col_off = couleur quand la variable est fausse

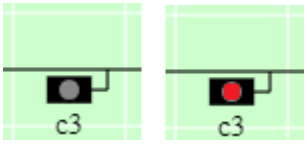
col_on = couleur quand la variable est vraie
col_border = couleur des bords (‘black’ par défaut)
txt = text à afficher sous le voyant (rien par défaut)
col_txt = couleur du texte (‘black’ par défaut)



add_bp(1,2,’r’,.4,.2,"i2", "#cccccc", "#ff0000");
* ici, le voyant est sur une ligne

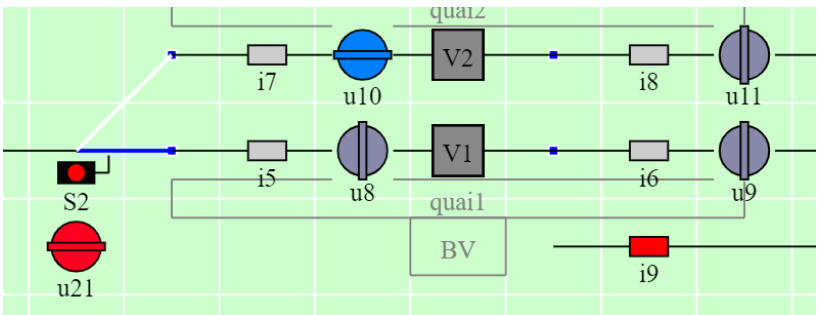
---

function **add_feu**(x, y, varStr, col_off, col_on, txt, col_txt, cible, col_cible)
Ajoute un voyant de feu pour TCO (qui ne s’allume qu’au rouge/violet), comme voyant mais avec une cible
x,y = cellule
varStr = variable utilisée pour l'affichage ex: "i0"
col_off = couleur quand la variable est fausse
col_on = couleur quand la variable est vraie
txt = text à afficher sous le voyant
col_txt = couleur du texte
cible = 0 pas de cible / 1-4=orientation (1=haut>| 2=haut|< 3=bas>| 4=bas|<)
col_cible = couleur du texte (‘black’ par défaut)



add_feu(1,2,"u28", "#cccccc", "#ff0000","c3",’black’,4);

Function **draw_rect**(x, y, dx, dy, col, col_bord, txt, col_txt)
Ajoute un rectangle (par ex pour indiquer un quai, la position de la gare …)
x,y = cellule
dx,dy = taille du voyant (1=100% d’une cellule, 0.4=40%, 10=10cellule)
col = couleur de remplisssage, "" si pas de remplissage
col_bord = couleur du contour, "" si pas de contour
txt = text, "" si pas de texte
col_txt = coleur du texte



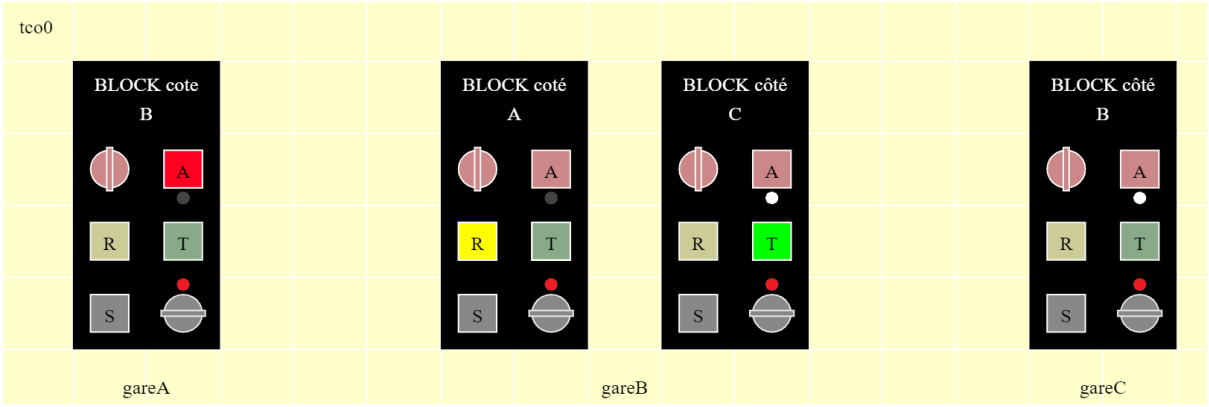
Actuellement, il y a une seule fonction composée. Elle crée un TCO au lieu d'ajouter soit même les 6 boutons et le fond.

function **add_bmvu**(x, y, col_fond, txt1, txt2, col_txt, u)
Ajoute un bmvu
x,y = cellule en haut à gauche
col_fond = couleur du fond du boitier du BMVU
txt1 = texte supérieur (souvent « Block coté »)
txt2 = texte inférieur
u = numéro de la première variable u utilisée.

8 variables sont utilisées par boitier bmvu
Comme il y a 2 boitiers par canton BMVU, il faut 16 variables u par canton bmvu.
Au niveau de la centrale, il faut activer les bmvu avec add_bmvu

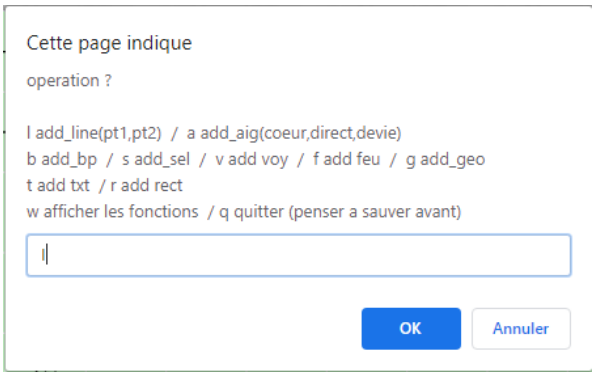
Les variables u sont les suivantes (à ajouter au numéro spécifié dans add_bmvu):
+0 sel blocage
+1 bp redition
+2 bp sonnerie
+3 bp annonce
+4 bp test
+5 sel semaphore
+6 voy semaphore
+7 voy voie libre

Comme, il y a 2 boitiers si on crée un BMVU par exemple avec u20, on créera le complémentaire avec u28 et on n'utilisera pas u20-u35 pour autre chose.



Afin de vous aider dans la conception des TCOs, le bouton edit du menu principal permet de créer simplement un TCO basic. L'éditeur utilise ses propres fonctions composées pour plus de simplicité. A la fin de l'édition, il affiche toute la liste des fonctions appelées. Il vous est ainsi possible de les copier/coller dans le TCO.

Une fois le bouton « Edit » pressé, une fenêtre demande quel TCO éditer. Ensuite un appuie sur la cellule 0,0 (en haut à gauche), affiche le menu d'édition



Par exemple pour ajouter une ligne, il faut appuyer sur la première cellule, puis la seconde … A la fin, appuyez sur la cellule 0,0. Idem pour les autres éléments. Lorsque le résultat convient, choisir w pour afficher la liste des fonctions utilisées. Cette liste étant affichée sous ou à côté du TCO, il faudra peut-être utiliser les barres de défilement pour voire cette fenêtre.

```
add_line(5,7,7,8);
add_line(7,8,9,8);
add_line(9,8,10,9);
add_line(10,9,9,10);
add_line(9,10,7,10);
add_line(7,10,5,9);
add_line(5,9,3,9);
add_line(3,9,2,8);
add_line(2,8,3,7);
add_line(3,7,5,7);
edit_add_btn(7,8,"BP",0);
```

Copier les fonctions, et les coller dans votre TCO. Bien entendu vous pouvez les modifier. Si vous ne rajouter pas ces fonctions, le TCO reste comme il était avant l'édition.

L'édition se termine en quittant le menu d'édition avec la lettre q

Voici les fonctions composées utilisées par l'éditeur :

```
function edit_add_aig(xc, yc, xd, yd, xe, ye, aig)
function edit_add_sel( x, y, u)
function edit_add_btn(x, y, txt, u)
function edit_add_geo(x, y, txt, num)
function edit_add_occ(x,y,i)
function edit_add_feu(x,y,txt,u,pos)
```

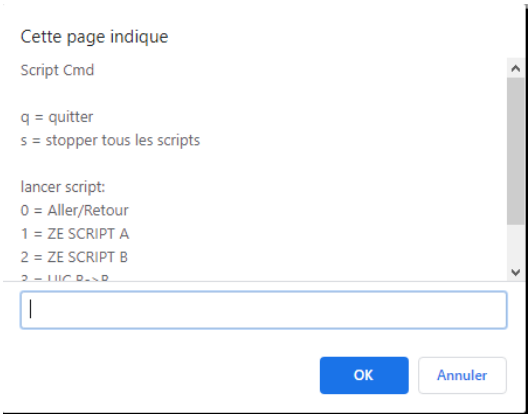
Pour tester, il est bien entendu possible de mettre le fichier dans la centrale (soit en reflashant, soit en utilisant le menu admin pour remplacer le fichier index.html. Il est aussi possible de tester directement sur PC en cliquant simplement sur le fichier index.html qui s'ouvre dans votre navigateur.

Lorsque la page web n'est pas connectée à la centrale, la commande 't' met à jours 'd' et 'e', la commande 'u' met à jour 'u' et il est possible d'envoyer une commande 'i' pour mettre à jour 'i' et ainsi simuler une entrée. Cela permet de tester les aiguillages, boutons, voyants … Bien entendu, pour les cas plus complexes comme les itinéraires, il faudra tester avec la centrale.

Il arrive de faire des erreurs ce qui fait que le TCO ou la page web ne marchent plus. Pour trouver l'erreur, les pro du javascript peuvent utiliser le débogueur javascript. Pour les autres, il est conseillé d'y aller pas à pas et de revenir en arrière lorsque cela ne marche plus.

4) Les scripts

Les scripts permettent d'automatiser des séquences de jeu. Ils sont composés d'une suite d'instructions qui s'exécutent une à après l'autre. Plusieurs scripts peuvent tourner en //. Le bouton Script du menu principal affiche le menu des scripts qui permet de lancer un script en tapant son numéro ou les arrêter.



Pour créer un script, il suffit d'ajouter ses instructions dans la fonction script_init du programme.

Les commandes suivantes sont disponibles :

```
Commande        Description

SCRIPT("nom");   nom et debut d'un nouveau script

CMD("cmd");      envoie la commande la_commande_sans_espace (protocole d17)
                 "a3s+v100f0+" // loco d'adresse 3, avant, vitesse 100%, f0 on
                 "t2-t3^"      // aig2 directe, aig3 inverse
                 "u0+u1-u2^"   // demander la mise à 1 de u0, mise a 0 de u1, invertion de u2
                 "x10,2+"      // activation de la sortie 2 du decodeur d'accessoire d'adresse 10

MSG("msg");      affiche le_message_sans_espace sur l'ecran du menu

TEMPO(sec);      tempo fixe (durée en secondes (min: .25, par pas de .25))

TEMPOR(min,max); tempo aléatoire durée mini_et max seconde

WAIT("var");     attente de la variable
                 "i0-95" pour les entrées,
                 "u0-191" pour les variables u,
                 "d0-47" pour aiguillage en position directe
                 "e0-e7" pour aiguillage en position deviee
                 rajouter ! devant la variable pour attendre qu'elle soit inactive

END()            fin du script

LOOP()           recommencer le script depuis le début

// blabla           commentaire
```

```
/*  blabla          commenatire   */
```

Petit exemple d'animation d'aller-retour de la loco 28 entre les détecteurs in1 et in2 avec attentes variables :

```
//aller retour entre in1 et in2     in1   loco28          in2
SCRIPT("Aller/Retour");

// Aller
MSG("depart aller");
CMD("a28s+v20f0+");
TEMPO(2.5);
CMD("a28v50");
WAIT("i2");
CMD("a28v20");
TEMPO(2);
CMD("a28v0f0-");

// attente variable
TEMPOR(10,20);

// Retour
MSG("depart retour");
CMD("a28s-v20f0+");
TEMPO(2.5); CMD("a28v50");
WAIT("i21");CMD("a28v20");
TEMPO(2);   CMD("a28v0f0-");

// attente variable
TEMPOR(20,60);

// on executec en boucle
LOOP()
```

5) Administration

La page d'administration accessible par le bouton Admin du menu principal ou à l'adresse 192.168.4.1 permet surtout de lister, ajouter ou supprimer les fichiers du serveur web de D17. La taille de cet espace de stockage est de 3MB. Il est par exemple possible de mettre à jour le fichier index.html qui est la page web des souris et TCOs.

Lorsqu'une nouvelle version de la souris D17 est disponible, penser à remplacer les TCOs et scripts d'exemples par les vôtres. Penser à bien sauver vos TCOs et scripts quelque part afin de ne pas les perdre.

Vous savez maintenant tout sur la souris HTML de D17.

Amusez-vous bien !