Manipulateur de traction - Cerclo

Le 08/04/2026

Ce document indique comment câbler un manipulateur de traction de BB7200 ou BB7600, pour l'utiliser avec mon programme Arduino de gestion du poste de conduite.

La description du poste de conduite se trouve ici : https://www.la-tour.info/uts/uts_page15.html

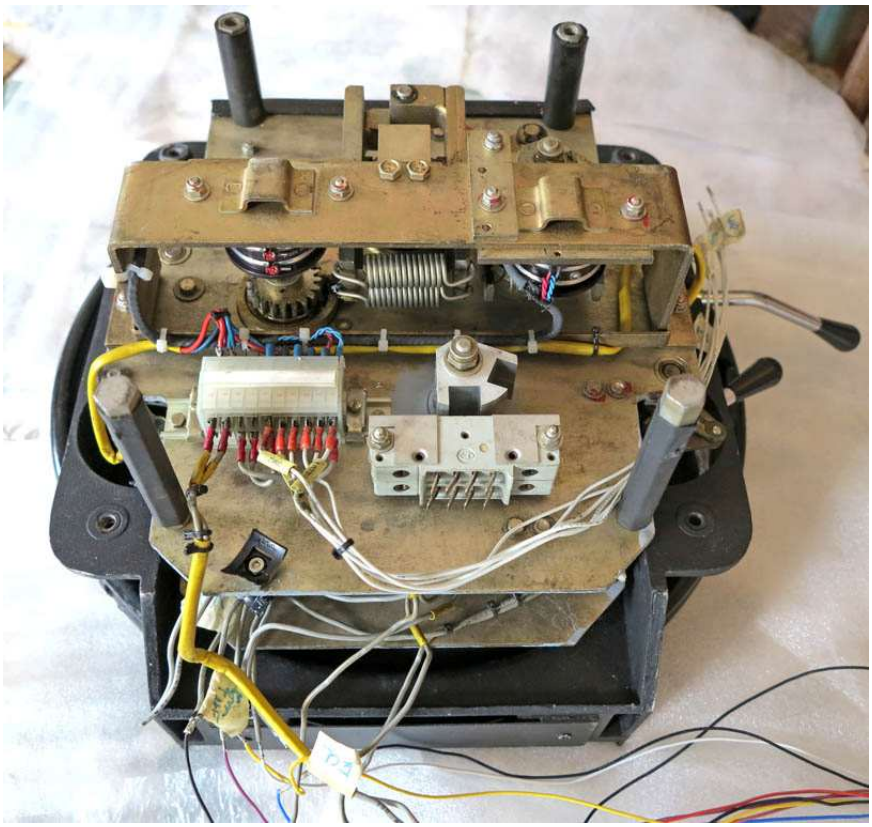
Ce manipulateur de traction possède deux potentiomètres. Un pour le réglage de la vitesse, et un autre pour le réglage de la vitesse imposée (VI).

Il n'y a pas de contact sur le petit levier de déverrouillage. Ce n'est pas évident de rajouter un contact sur ce petit levier. Il sera simulé dans l'Arduino, avant chaque déplacement du levier de traction.

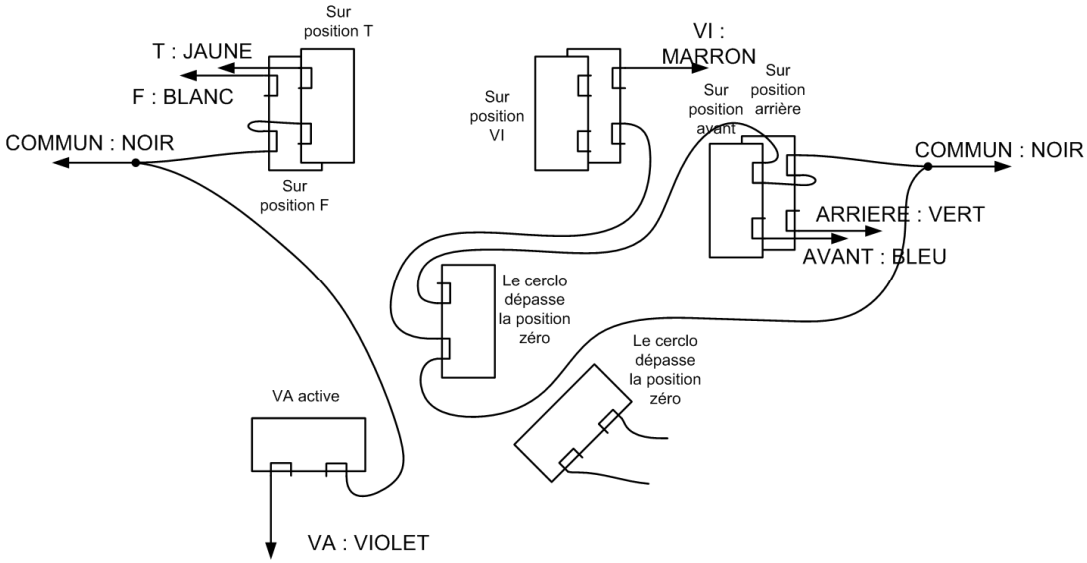
Pour plus de détail sur le code, voir la description du poste de conduite qui se trouve ici : https://www.la-tour.info/uts/uts_page15.html#conduite

Poser vos questions sur le forum RMF, par exemple ici : https://www.rmf-magazine.com/phpBB/viewtopic.php?t=203075

On peut garder les fils déjà branchés sur les contacteurs, ou refaire tout le câblage.
J'ai choisi de conserver tous les fils déjà en place.



Le schéma de câblage, vue de dessous :



Remarques :

On n'a pas de contact directement utilisable pour la position 'VI'. Il est en série avec un autre contact.

Pour corriger cela, sur le contacteur situé au centre, du cerclo qui dépasse la position zéro, on branche le fil qui sort, vers le fil du commun. On obtient ainsi sur le fil violet, un contact quand la manette est sur la position 'VI'.

Les fils du deuxième contacteur, du cerclo qui dépasse la position zéro ne sont pas utilisés.

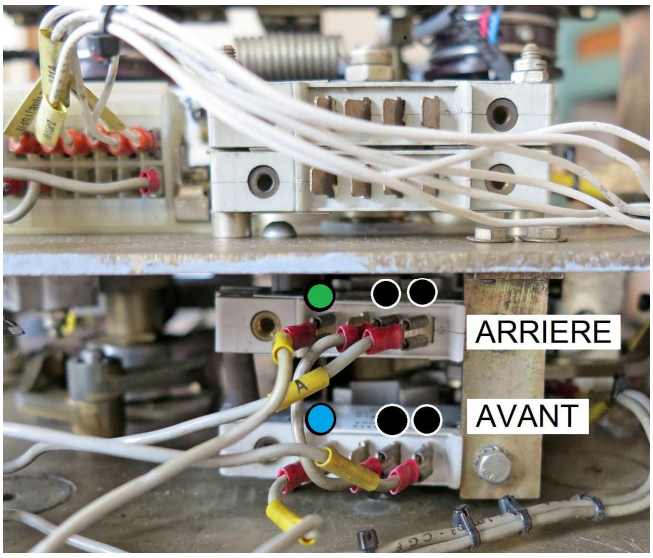
J'ai donné la couleur suivante aux fils en sortie :

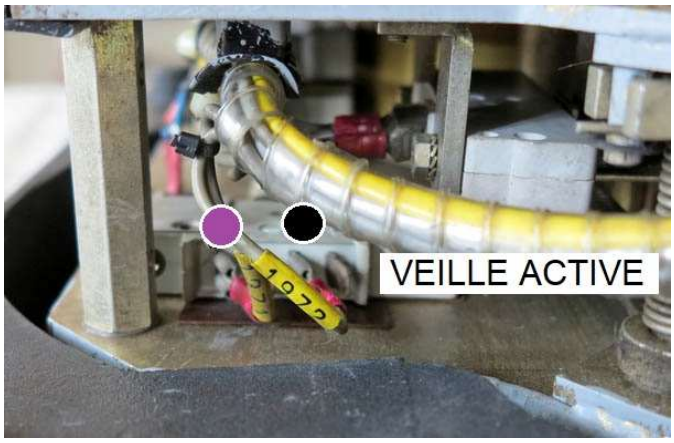
ARRIERE : VERT
AVANT : BLEU
VA : VIOLET
F : BLANC
T : JAUNE
VI : MARRON
COMMUN : NOIR
COMMUN : NOIR
ECLAIRAGE -12V : NOIR
ECLAIRAGE +12V : ROUGE

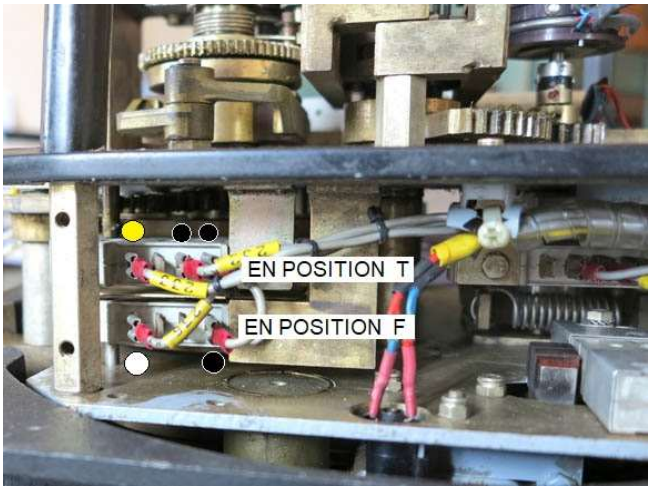
Il y a trois fils pour le commun des contacts. Je les ai reliés ensemble.

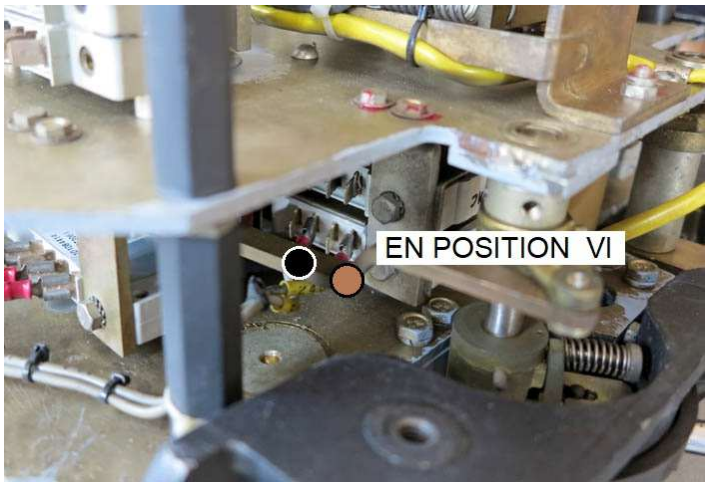
Ne pas relier le fil -12 Volts (noir) de l'éclairage aux autres fils de commun (noir). En cas de problème, on pourrait se retrouver avec du 12 Volts sur les contacts d'entrée.

Voici la position de tous ces fils sur les contacteurs.





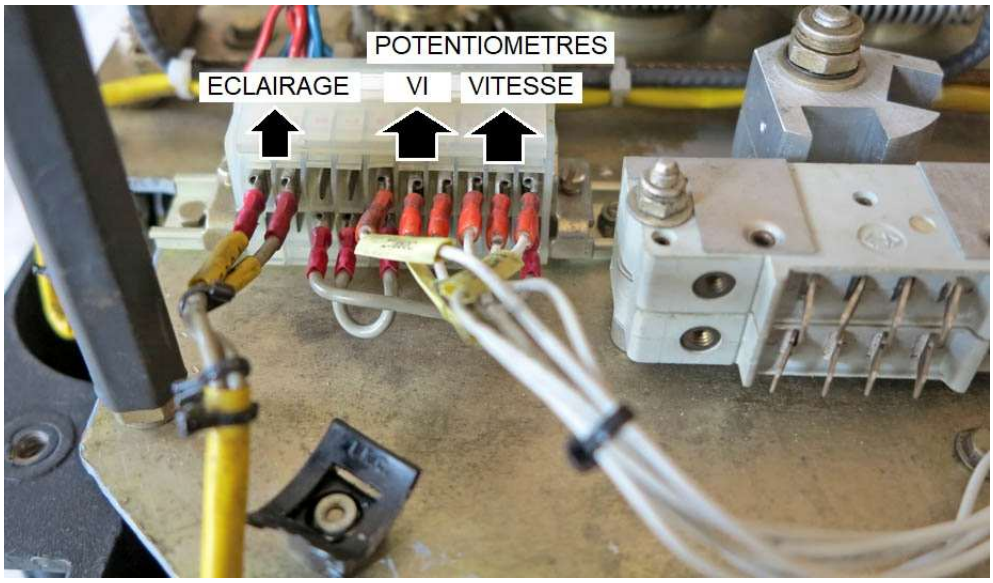




L'éclairage et les potentiomètres sont accessibles sur le bornier du dessus.

On a les connexions pour le potentiomètre de l'intensité du courant (*Vitesse*), et pour le potentiomètre de VI.

VITESSE 0V
VITESSE MILIEU
VITESSE 3,3V
VI 0V
VI MILIEU
VI 3,3V



<u>Les potentiomètres :</u>

Les trois fils des deux potentiomètres sont directement accessibles.

Les potentiomètres ont une résistance de 100 Ohms, ce qui est très faible.

Sur mon circuit imprimé, (https://www.la-tour.info/uts/uts_page15.html#conduite) j'avais placé par défaut une résistance de 22 Ohms, pour limiter le courant à 150 mA en cas de court-circuit entre les plots 3,3V et GND, des entrées analogiques.

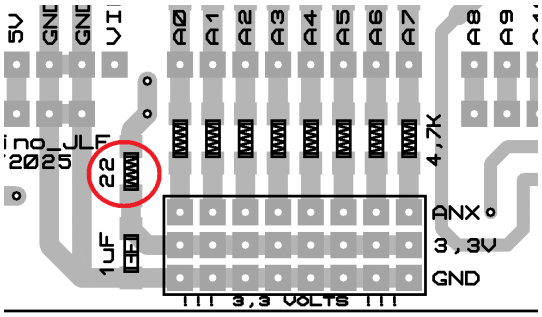
Si l'on branche les deux potentiomètres de 100 Ohms, la tension va passer de 3,3 Volts à 2,2 Volts et trop réduire la plage de conversion des signaux.

Le courant maximum que peut délivrer la carte DUE en sortie 3,3 Volts, est de 850 mA.

Je vais remplacer la 22 Ohms par une 2,2 Ohms.

Le courant de court-circuit sera alors de 1,5 Amp, et la tension sur les plots "3,3V" de 3,17 Volts.

Remplacer la résistance de 22 Ohms par une de 2,2 Ohms :



Les deux potentiomètres consomment 65 mA en tout.

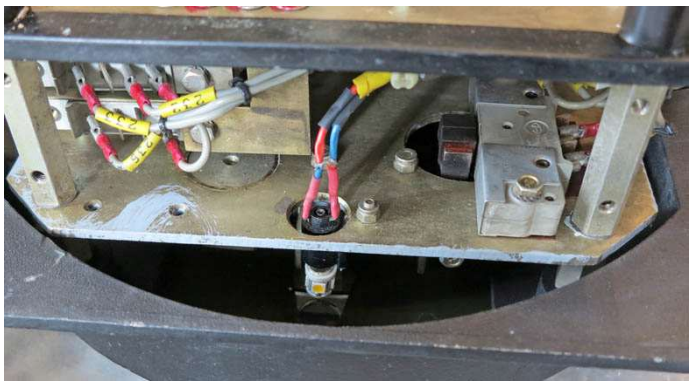
Attention de ne pas mettre en contact un des fils des potentiomètres ou une des entrées analogique au +5 Volts. Ces entrées ne tolèrent pas plus de 3,3 Volts.

Ne pas inverser le plot central des potentiomètres avec le +3,3V ou le 0V. En buté, on aurait un court circuit sur le 3,3V.

Au moment de souder ou de travailler sur les potentiomètres, il faut que le connecteur analogique des entrées de la carte Arduino soit débranché. Ces entrées analogiques sont très fragiles.

L'éclairage 12 Volts

J'ai remplacé les deux ampoules 12 Volts 4 Watts BA9S, par des 12 Volts à leds.



Une des deux lampes ne s'allumait pas, car l'arrivé des fils était inversée sur une des deux lampes.

J'ai du intervertir les fils à l'arrivée d'une lampe, pour que le (+) 12 Volts arrive sur le plot des deux lampes.

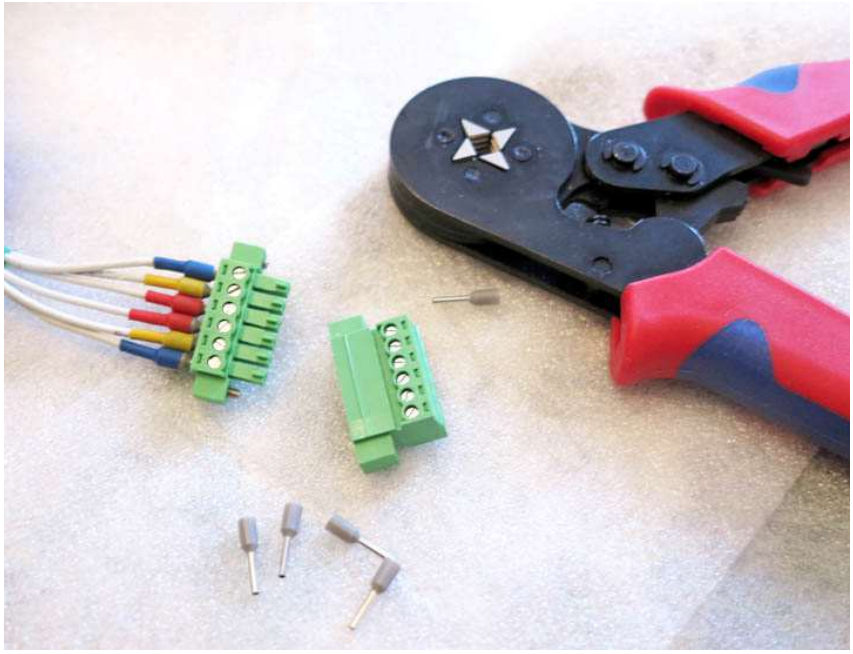
Les connexions

Afin de pouvoir retirer l'ensemble du manipulateur de traction pour maintenance, je l'ai équipé de prises.

On de doit pas pouvoir inverser les fils des potentiomètres avec le reste, car la tension sur ces fils ne doit pas dépasser 3,3 Volts.

Pour les connecteurs, j'utilise des embouts à sertir. Ca permet de bien tenir un fil souple avant de le placer dans un connecteur. C'est beaucoup moins cassant que d'étamer ces fils souples.

J'ai utilisé un connecteur à 6 contacts pour les potentiomètres, et à 10 contacts pour le reste.

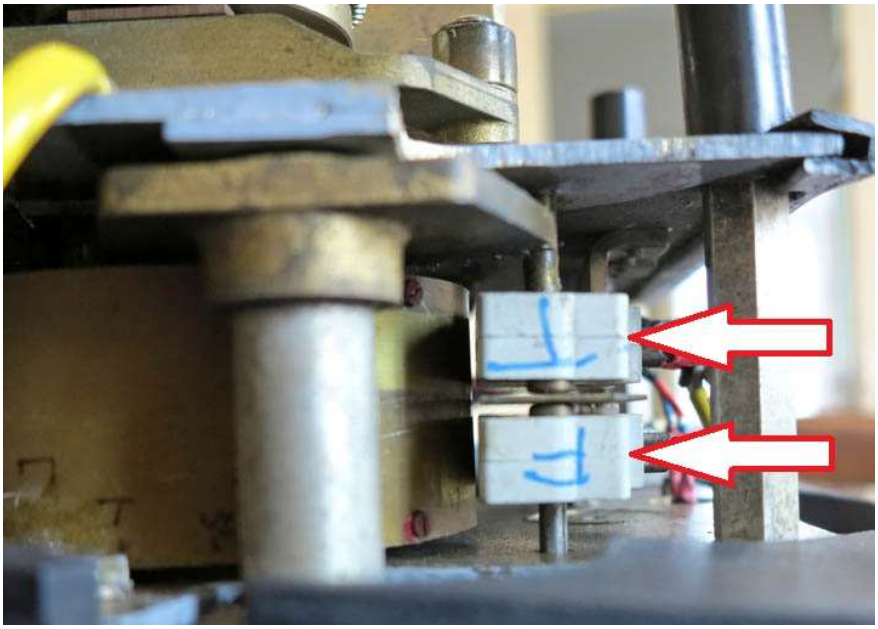


Maintenance et dépannage

En manipulant cet appareil, les contacts 'F' et 'T' ne fonctionnaient plus très bien.

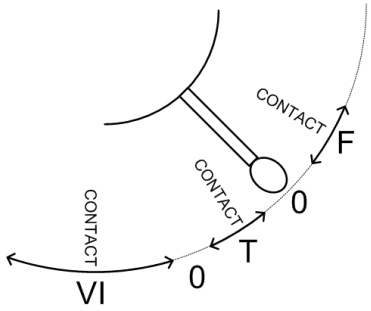
A force de travailler dessus, j'ai du tirer un peu trop sur les câbles et déplacer les contacteurs gris.

Les blocs gris des contacteurs, peuvent se déplacer. On peut appuyer sur leur coté pour les rapprocher du gros cylindre. Il faut garder un espace de moins d'un mm avec le cylindre tournant, sans les faire frotter.



Amélioration du fonctionnement du levier de traction

Avec ce câblage utilisant directement les contacts existants, on se retrouve avec cette configuration :



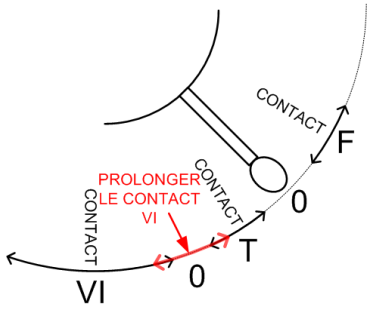
Position '0' → 'F' : Ok, le contact 'F' se ferme.
Position 'F' → '0' : Ok, le contact 'F' s'ouvre.
Position '0' → 'T' : Ok, le contact 'T' se ferme.
Position 'T' → '0' : Ok, le contact 'T' s'ouvre.

Pour ces trois positions 'T', '0' et 'F', l'Arduino et le simulateur savent exactement dans quelle position est le levier de traction.

Mais, quand l'on passe de la position 'T' à la position 'VI', en début de zone 'VI', il n'y a pas de contact d'établi. Il faut descendre un peu plus la manette pour faire ce contact 'VI'.

Dans, ce cas de figure, l'Arduino ne sait pas si l'on est passé de la position 'T' à la position '0', ou de la position 'T' sur la position 'VI'.

Pour corriger cela, et permettre à l'Arduino de connaitre exactement la position du levier de traction, il faut prolonger le contact 'VI.



Si l'on prolonge la plage du contact VI en rouge, l'Arduino saura déterminer la position exacte du levier.

Il faut que cette plage en rouge dépasse la plage des contacts 'VI' et 'T', pour faire complètement disparaitre la zone sans contact assimilable à la position '0'.

Pour cela, on va ajouter un petit interrupteur ILS.

J'ai utilisé un petit aimant de 4 x 5 x 1 mm, et un ILS de 14 mm de long.

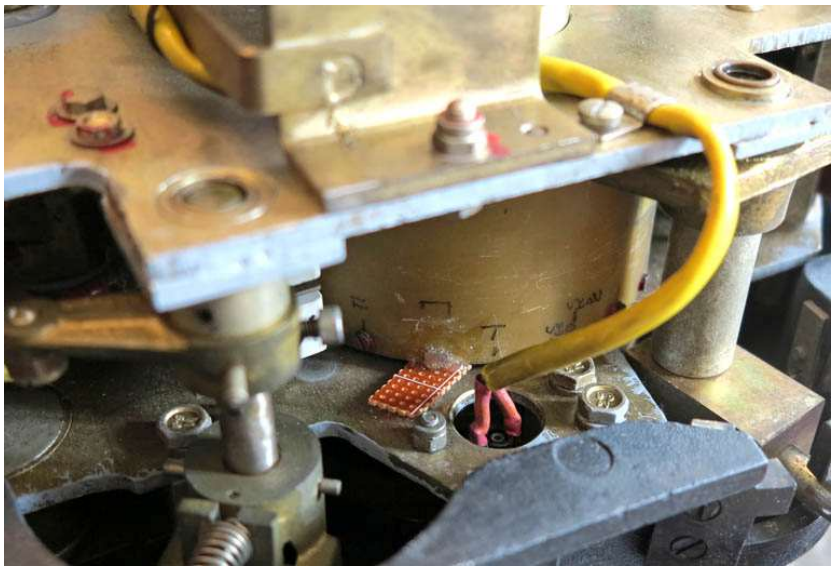
Il faudra positionner l'aimant en bas du cylindre, pour qu'il passe sans toucher les autres équipements.

Pour définir la position de l'aimant, on commencera par le fixer au double face très fin.

Apres avoir nettoyé à l'acétone, l'aimant et la zone de collage, je l'ai collé à la néoprène transparente.

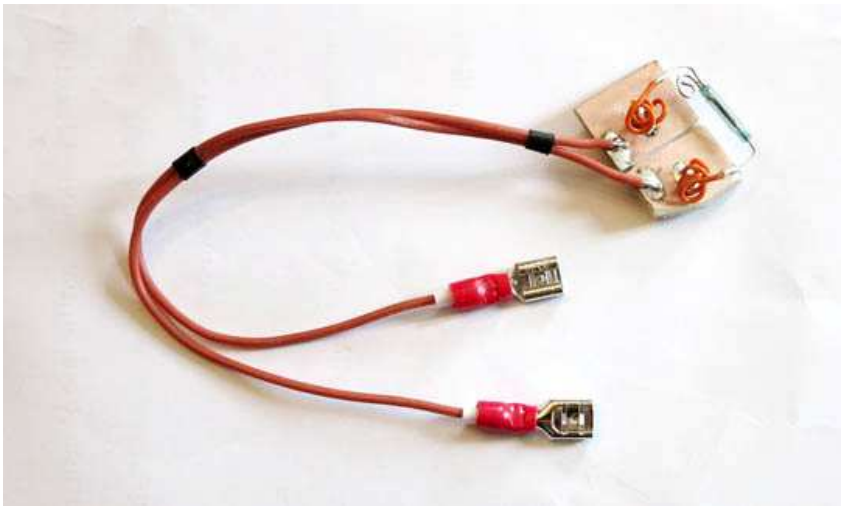
Mettre une plaque sous l'aimant, pour éviter qu'il ne glisse vers le bas le temps du collage, car il est attiré par cette masse métallique.

Le levier est ici en position début de la zone 'VI'. L'aimant sera positionné en face du début de l'ILS



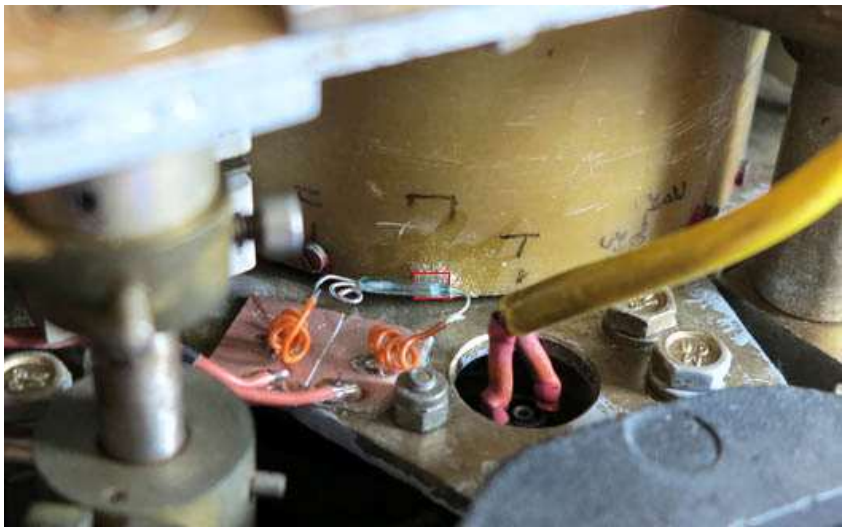
L'ILS est soudé sur une plaque de circuit imprimé de 2 x 3 cm. J'ai utilisé du fil rigide pour pouvoir le positionner précisément.

Dans la version définitive, un deuxième fil est branché sur une des deux cosses, pour aller vers le contacteur 'VI'.



Nettoyer la plaque et la zone de collage à l'acétone. Je l'ai collé à la néoprène transparente.

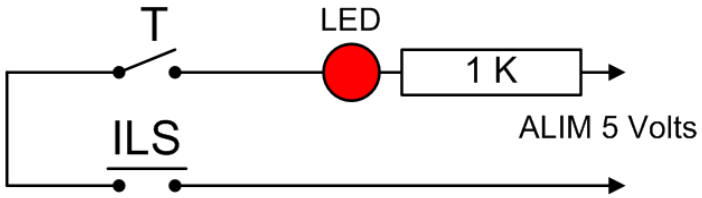
Sur la photo, le levier est en position **début de la zone 'VI'**. L'aimant est positionné en face du début de l'ILS



Une fois le tout collé, on règlera la position définitive de l'ILS.

Au passage de la position 'T' à 'VI', le contact doit s'établir.
Il ne doit pas y avoir de trou entre la position 'T' et la position 'VI'.

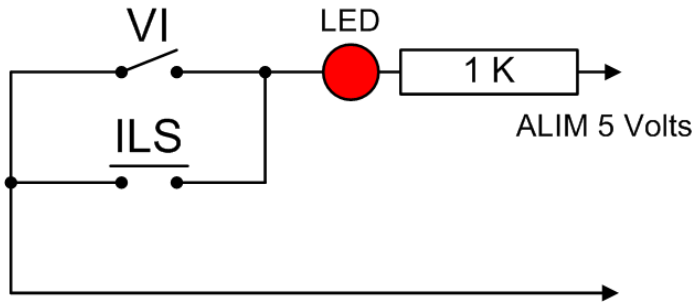
Important, pour vérifier la position définitive de l'ILS, on utilisera une led témoin, et les contacts 'T' et ILS en série.



Au passage 'T' vers 'VI', la led doit s'éclairer brièvement.
Au passage 'VI' vers 'T', la led doit également s'éclairer brièvement.
Si ce n'est pas le cas, il faut déplacer l'ILS vers la droite. Il faut qu'il y ait un recouvrement des contacts.

L'ILS fait contact sur une faible zone de déplacement du levier. Même avec un petit aimant, le contact 'VI' doit se fermer avant que le contact de l'ILS s'ouvre.

Important, on vérifiera aussi avec les contacts 'VI' et ILS en parallèle, que la led est allumée sur toute la plage 'VI'.



Si la led s'éteint, il faut déplacer l'ILS vers la gauche, ou prendre un aimant plus puissant.

Optimisation du fonctionnement du levier de traction

Je présente mon programme de gestion du poste de conduite, ici : https://www.la-tour.info/uts/uts_page15.html#conduite

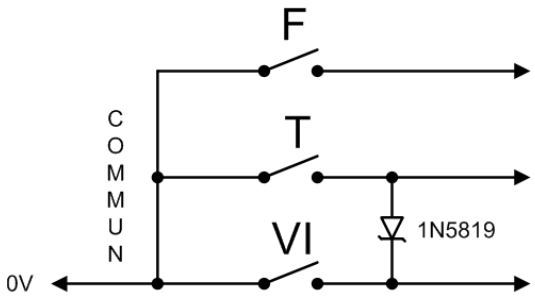
Tous les contacts des entrées sont regroupés sur des cartes périphériques I2C.

Le code est optimisé, pour détecter et traiter tout changement d'état.

Pour cet appareil, même une fois l'ILS ajouté, quand levier de traction dépasse le début de la zone 'VI', le contact 'T' s'ouvre.

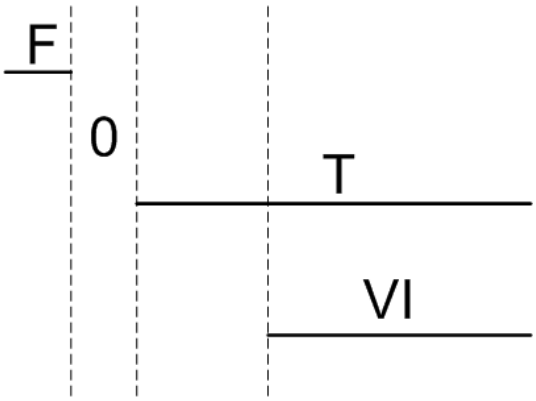
Ce changement d'état est géré par d'Arduino, alors que cela n'est pas nécessaire, et peut amener des confusions dans le programme.

Pour éviter cela, on ajoute une diode Schottky 1N5819.



Comme cela pour le levier de traction, on n'a que quatre états.

Et surtout, une seule entrée change de valeur, entre deux états.



C'est plus facile à gérer dans l'Arduino. On n'a pas besoin de temporisation pour stabiliser la lecture des entrées, et l'on sait exactement dans quelle position est le levier.

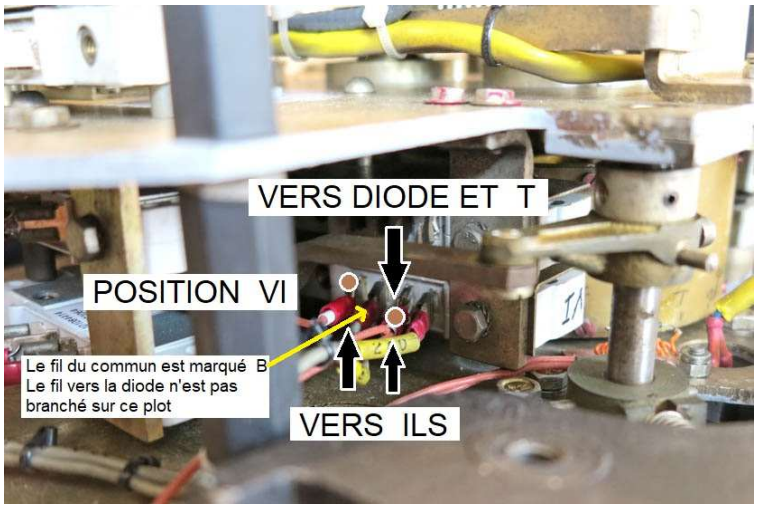
Dans l'Arduino, j'ai le calcul suivant en fonction des entrées : 'Position levier' = 1 - F + T + VI.

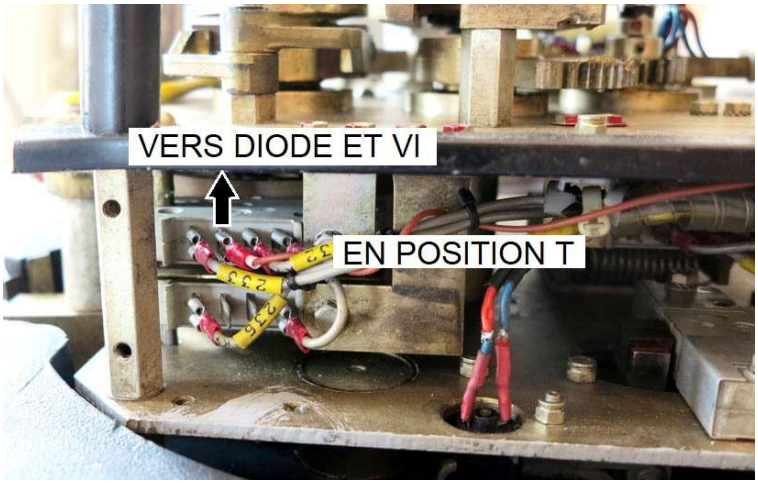
Position levier : 0 = 'F', 1 = '0', 2 = 'T', 3 = 'VI'

J'ai utilisé la formule :

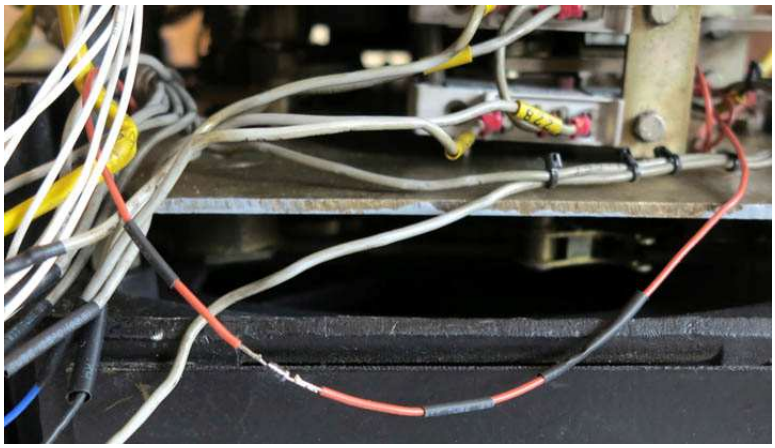
```
ltrac_position = 1 - bitRead(i2c_in1_val, 11) + bitRead(i2c_in1_val, 12) + bitRead(i2c_in1_val, 13);
```

Branchement des fils de l'ILS sur les plots marron, et de la diode 1N5819 sur le fil de liaison :





La diode insérée dans le fil de liaison :



Une fois le manipulateur de traction branché à ma carte Arduino, j'ai relevé les valeurs suivantes :
Avec la résistance de 2,2 Ohms en protection sur la sortie 3,3 Volts, de ma carte de poste de conduite.

Avant la résistance : 3,27 Volts, après la résistance 3,13 Volts. Ce qui est conforme à la prévision de consommation de 60 mA pour les deux potentiomètres.

La valeur de conversion de 3,13 Volts, pour 3,27 Volts pleine échelle, donne la valeur 979, pour 1023 pleine échelle.

Valeur de conversion mesurée avec cette carte :
Pour le potentiomètre I : 0 à 943
Pour le potentiomètre VI : 0 à 976

Avec mon montage de gestion du poste de conduite, Sur l'écran LCD, choisir le menu "(4]AFFICHAGE VAL. / ANALOGIQUE IN" pour afficher la valeur de conversion de 0 à 1023.

Avec cette faible tension et un courant relativement important, il faut de bons contacts pour les prises vers les potentiomètres.

Si la valeur lue change quand l'on bouge les prises, il est conseillé de passer un coup d'aérosol pour contact.

Dans le programme Arduino

Mon système de gestion de poste de conduite, est présenté ici : https://www.la-tour.info/uts/uts_page15.html#conduite

Le code est celui pour la BB7200 de SimExpress, avec Train Simulator Classic.

Voici les lignes de code pour l'ensemble des commandes de traction, adapté au programme Train Simulator Classic.

Pour le levier de traction à 4 positions :

```
// Levier de traction : Frein / 0 / Traction / Vitesse Imposée.
// Le petit levier de déverrouillage situé à gauche, doit être actionné lors du passage de : '0' vers 'F', 'T' vers 'VI' et 'VI' vers 'T'.
// Il n'y a pas de contact sur ce petit levier de déverrouillage. On simule son action dans le code (Touche KEY_RETURN).
  ltrac_position = 1 - bitRead(i2c_in1_val, 11) + bitRead(i2c_in1_val, 12) + bitRead(i2c_in1_val, 13);  // 0='F',  1='0',  2='T',  3='VI'.
  bitWrite(i2c_in1_val_ref, 11, bitRead(i2c_in1_val, 11));                                  // Pour indiquer la position de référence, pour comparer avec le simulateur.
  bitWrite(i2c_in1_val_ref, 12, bitRead(i2c_in1_val, 12));
  bitWrite(i2c_in1_val_ref, 13, bitRead(i2c_in1_val, 13));
  if (ltrac_position != ltrac_position_old) {
    // Le levier de traction vient de changer de position, donc sur le pupitre le cerclo était en buté à droite.
    // Sur le simulateur, on donne un petit coup à droite au cerclo, dès fois qu'il ne soit pas complètement à droite, pour déverrouiller le levier de traction.
    tch_env(NORMAL, 'd', 300); ltrac_val_ref = 0; regul_val_ref = 0; // Et la position de référence des potentiomètres 'Courant' et 'VI' est à zéro.
    // On teste d'abord les cas de passage où l'on doit activer le levier de déverouillage.
    if     ((ltrac_position_old == 1) && (ltrac_position == 0)) { tch_env(NORMAL, KEY_RETURN, PRESS); tch_env(NORMAL, 'q', duree_touche_400);
tch_env(NORMAL, KEY_RETURN, RELEASE); ltrac_position_old--; } // '0' ->  'F'.
    else if ((ltrac_position_old == 2) && (ltrac_position == 3)) { tch_env(NORMAL, KEY_RETURN, PRESS); tch_env(NORMAL, 'e', duree_touche_200);
tch_env(NORMAL, KEY_RETURN, RELEASE); ltrac_position_old++; } // 'T' ->  'VI'.
    else if ((ltrac_position_old == 3) && (ltrac_position == 2)) { tch_env(KEY_LEFT_SHIFT, 'e', 400); // Pour si besoin, arriver en buté haute du levier 'VI'.
                                                  tch_env(NORMAL, KEY_RETURN, PRESS); tch_env(NORMAL, 'q', duree_touche_200); tch_env(NORMAL, KEY_RETURN,
RELEASE); ltrac_position_old--; } // 'VI' ->  'T'.
    else if ((ltrac_position_old == 3) && (ltrac_position == 1)) { tch_env(NORMAL, KEY_RETURN, PRESS); tch_env(NORMAL, 'q', duree_touche_200);
tch_env(NORMAL, KEY_RETURN, RELEASE); ltrac_position_old--; } // 'VI' ->  '0'.
    // On teste ensuite les cas de passage de deux positions à la fois.
    if     (ltrac_position < ltrac_position_old - 1) { tch_env(NORMAL, 'q', duree_touche_200); ltrac_position_old--; }
    else if (ltrac_position > ltrac_position_old + 1) { tch_env(NORMAL, 'e', duree_touche_200); ltrac_position_old++; }
    // On teste enfin les cas de passage d'une position à la fois.
    if     (ltrac_position < ltrac_position_old) { tch_env(NORMAL, 'q', duree_touche_200); ltrac_position_old--; }
    else if (ltrac_position > ltrac_position_old) { tch_env(NORMAL, 'e', duree_touche_200); ltrac_position_old++; }
    ltrac_position_old = ltrac_position;
  }
```

Pour l'inverseur de traction à 3 positions :

```
// Inverseur de traction : AVANT / 0 / ARRIERE.
  invtrac_position = 1 - bitRead(i2c_in1_val, 14) + bitRead(i2c_in1_val, 15);  // 0='AV',  1='0',  2='AR'.
  bitWrite(i2c_in1_val_ref, 14, bitRead(i2c_in1_val, 14));                  // Pour indiquer la position de référence, pour comparer avec le simulateur.
  bitWrite(i2c_in1_val_ref, 15, bitRead(i2c_in1_val, 15));
  if (invtrac_position != invtrac_position_old) {
    // On teste d'abord les cas de passage de AV <-> AR directement.
    if     (invtrac_position < invtrac_position_old - 1) { tch_env(NORMAL, 'w', duree_touche_400); invtrac_position_old--; } // w pour z  Inverseur de traction, en
avant. 600 msec.
    else if (invtrac_position > invtrac_position_old + 1) { tch_env(NORMAL, 's', duree_touche_400); invtrac_position_old++; } //     s     Inverseur de traction, en
arrière. 400 msec.
    // On teste ensuite les cas de passage de AV <-> 0 <-> AR.
    if     (invtrac_position < invtrac_position_old) { tch_env(NORMAL, 'w', duree_touche_400); invtrac_position_old--; } // w pour z  Inverseur de traction, en
avant. 600 msec.
    else if (invtrac_position > invtrac_position_old) { tch_env(NORMAL, 's', duree_touche_400); invtrac_position_old++; } //     s     Inverseur de traction, en
arrière. 400 msec.
    invtrac_position_old = invtrac_position;
  }
```

Pour le potentiomètre du régulateur de courant :

```
// Gestion du potentiomètre du régulateur de courant.
// Potentiomètre branché en entrée AN0.
// De 0 = Complètement à droite, à 943 = A fond à gauche.
// Si il y a un grand déplacement rapide, on bouge de 4 crans à la fois.
// Quand on arrive en bout, pour assurer la position d'extrémité, on bouge de 7 crans à la fois.
// Levier de traction : ltrac_position = 0=Frein / 1='0' / 2=Traction / 3=Vitesse Imposée.
// regul_val_old = 0 à 1023.
// regul_val     = 0 à 31.
if (millis() > lect_can_milli) {              // Pour ne pas passer 25000 fois par seconde dans ce code. On économise du cpu par boucle.
```

```
 lect_can_milli = millis() + 100;              // On passe ici toutes les 100 msec.
 regul_val = analogRead(pin_regul_broche_bb);   // 0 à 1023.
 if ((regul_val - regul_val_old > 10 ) || (regul_val - regul_val_old < -10)) { // Un peu d'hystéresis, pour que la valeur ne bouge pas sans arrêt.
  regul_val_old = regul_val;
 }
 regul_val = (regul_val_old>>5) + (regul_val_old>>8);        // >>5 : 0 à 31 = 32 Pas de régulation. >>8 pour rattraper l'écart 943 à 1023.
 regul_val = min(regul_val, 31);
 if (regul_val != regul_val_ref) {
  if     (regul_val == 31)             { tch_env(NORMAL, 'a', 200); regul_val_ref = regul_val_ref + 7; }  // a pour q Traction : (+)  <--   On arrive en butée.
  else if (regul_val ==  0)             { tch_env(NORMAL, 'd', 200); regul_val_ref = regul_val_ref - 7; }  //    d     Traction : -->  (-)   On arrive en butée.
// Voir si besoin de passer 4 crans à la fois.
////       else if (regul_val >= regul_val_ref + 4) { tch_env(NORMAL, 'a', 100); regul_val_ref = regul_val_ref + 4; }  // a pour q Traction : (+)  <--
////       else if (regul_val <= regul_val_ref - 4) { tch_env(NORMAL, 'd', 100); regul_val_ref = regul_val_ref - 4; }  //    d     Traction : -->  (-)
   else if (regul_val >  regul_val_ref)    { tch_env(NORMAL, 'a',  25); regul_val_ref++; }                 // a pour q Traction : (+)  <--
   else if (regul_val <  regul_val_ref)    { tch_env(NORMAL, 'd',  25); regul_val_ref--; }                 //    d     Traction : -->  (-)
   regul_val_ref = min(regul_val_ref, 31);
   regul_val_ref = max(regul_val_ref,  0);
   ic2_in_comp_ref_milli = ic2_in_comp_ref_milli + duree_touche_200; // Si le cerclo a bougé sur le pupitre, on attendra un peu plus avant de vérifier les valeurs du simu.
 }
```

Pour le potentiomètre du levier de traction en position VI :

```
// Gestion du potentiomètre VI entrainé par le levier de traction en position VI.
// Le potentiomètre est branché en entrée AN1.
// De 0 = en haut, à 976 en bas.
//
//  Levier de traction : Trois contacts utilisés :            F(in1,9)         T(in1,10) VI(in1, 11)
//  Positions du levier de traction    |  nom de variable   |    F    |   O    |   T    |   VI    |
//  ------------------------------------+--------------------+---------+--------+--------+---------+-----------------------------------------
//  Retour position levier traction    |  traction_control  |    1    |  0,666 |  0,333 | variable | Pas significatif si L_VI_control = 1
//  Retour positio aiguill compteur VI |  aigu_VI_control   |    0    |   0    |   0    | 0 à 160 |  Vitesse Imposée en km/h
//  Retour valeur du potentiomètre     |  L_VI_control      |    0    |   0    |   0    |    1    |  1 = Levier sur Vitesse Imposée en cours
//  Retour info si position VI actif   |  visi_trac_control |variable |variable |variable |  0 à 10  |  0 = VI minimum,   10 = VI maximum
//
//  Trois contacts utilisés : F(in1,9)  T(in1,10) et VI(in1, 11). Le simulateur est très lent à réagir. La durée d'appui des touches est en conséquence très élevée.
//  Contraintes sur le levier de vérrouillage (in1,0) :
//   - F -       ^          -
//               | Verrouillé  | Libre
//   - O -  ^    -          v
//          | Libre
//   - T -  v    ^                   ^
//               | Verrouillé. Pour le sens | il faut avoir rélaché le levier de verrouillage entretemps.
//   - VI -      v                   -
//
////  if (visi_trac_control == 1) {   // 'visi_trac_control' = Info du simulateur. On est en position VI, sinon pas besoin de lire ce potentiomètre.
  if (ltrac_position == 3) {     // 'ltrac_position' = Info local. On est en position VI, sinon pas besoin de lire ce potentiomètre. 0='F',  1='0',  2='T',  3='VI'.
   ltrac_val = analogRead(pin_ltrac_broche_bb);  // 0 à 1023.
   if ((ltrac_val - ltrac_val_old > 10 ) || (ltrac_val - ltrac_val_old < -10)) { // Un peu d'hystéresis, pour que la valeur ne bouge pas sans arrêt.
    ltrac_val_old = ltrac_val;
   }
   ltrac_val = (ltrac_val_old>>5) + (ltrac_val_old>>9);  // 0 à 31 = 32 Pas de régulation. >>9 pour rattraper l'écart 976 à 1023.
   ltrac_val = min(ltrac_val, 31);
   if (ltrac_val != ltrac_val_ref) {
    if     (ltrac_val == 31)             { tch_env(KEY_LEFT_SHIFT, 'q', 800); ltrac_val_ref = ltrac_val_ref + 7; }  // Q pour A Traction : (+)  <--   On arrive en butée.
    else if (ltrac_val ==  0)             { tch_env(KEY_LEFT_SHIFT, 'e', 800); ltrac_val_ref = ltrac_val_ref - 7; }  //    E     Traction : -->  (-)   On arrive en butée.
// Voir si besoin de passer 4 crans à la fois.
////       else if (ltrac_val >= ltrac_val_ref + 4) { tch_env(KEY_LEFT_SHIFT, 'q', 400); ltrac_val_ref = ltrac_val_ref + 4; }  // Q pour A Traction : (+)  <--
////       else if (ltrac_val <= ltrac_val_ref - 4) { tch_env(KEY_LEFT_SHIFT, 'e', 400); ltrac_val_ref = ltrac_val_ref - 4; }  //    E     Traction : -->  (-)
    else if (ltrac_val >  ltrac_val_ref)    { tch_env(KEY_LEFT_SHIFT, 'q', 100); ltrac_val_ref++; }                 // Q pour A Traction : (+)  <--
    else if (ltrac_val <  ltrac_val_ref)    { tch_env(KEY_LEFT_SHIFT, 'e', 100); ltrac_val_ref--; }                 //    E     Traction : -->  (-)
    ltrac_val_ref = min(ltrac_val_ref, 31);
    ltrac_val_ref = max(ltrac_val_ref,  0);
    ic2_in_comp_ref_milli = ic2_in_comp_ref_milli + duree_touche_200; // Si le levier a bougé sur le pupitre, on attendra un peu plus avant de vérifier les valeurs du simu.
   }
 }
```

A+