Ce document décrit les principales **Etapes** du projet **Hélicoïdal 4**

Il se lit de préférence à l'envers (de la fin au début) en tenant compte des dates / Gérald Litzistorf

Déc 2022	**Synthèse des 12 derniers mois**
Nov 2022	**Détecter des anomalies de fonctionnement**
Oct 2022	**En route pour atteindre l'objectif de 20 trains en mouvement simultané**
Eté 2022	**Pause des activités H0 & Nouveaux développements LGB**
Juin 2022	**M83 le sauveur !**
Mai 2022	**Consommation transmise par Bluetooth Low Energy**
Avril 2022	**Encore un BUG !!!**
Mars 2022	**Je me convertis au MFX**
Fév 2022	**Dispositifs Anti-collisions**
Jan 2022	**Bonheur & Malheur**
Déc 2021	**Etape 5 : Niveau 0 (Construction – Logiciels – Tests)**
Oct 2021	**Etape 5 : Niveau 0 (Design – Simulation)**
Sept 2021	**Etape 4 : Niveau –1**
Printemps 2021	**Pause des activités H0 & Nouveaux développements LGB**
Avril 2021	**Mesures de performance**
Mars 2021	**Consommation proche de la limite ?**
Fév 2021	**Etape 3 : Niveau –2**
Jan 2021	**Interface graphique**
Déc 2020	**Etape 2 : Niveau –3**
Nov 2020	**Etape 1 : 2 hélicoïdaux face à face**
Oct 2019	**Simulateur**

Déc 2022 Synthèse des 12 derniers mois

… on apprend chaque jour …
Ci-dessous, les principaux points illustrant l'expertise acquise en 2022 :

1) Les décodeurs **M83** (Turn) fonctionnent à la perfection depuis juin 2022

2) Les détections **S88** sont essentielles pour le bon fonctionnement de la maquette
A certains endroits (hélicoïdal, …), j'ai fraisé le rail pour éviter un court-circuit de l'isolation
Par paresse, j'ai parfois évité cet effort que j'ai remplacé par ce test logiciel :
- au lancement du programme, **tester toutes les rétrosignalisations actives** (=1)
- soit elles sont légitimes en fonction de la position des trains mémorisées dans D:\loc.txt
- soit elles sont signalées sur le GUI et sur la sortie COM4 (putty)

3) Connaître en permanence la **consommation (A) des trains** a toujours été ma devise
La maquette comprend 4 zones isolées comprenant 1 booster DB4 par zone :

Gauche Niv0 & Niv-1	Niv0 & Niv-1 Droite
Gauche Niv-2 & Niv-3	Niv-2 & Niv-3 Droite

3.28 0.01 ● 0 2.55
2.27 0 0 2.05

Les 4 valeurs proches du point bleu indique la valeur instantannée transmise par Bluetooth Low Energy chaque seconde
Les 4 autres valeurs mémorisent la valeur maximum

4) Depuis mars 2022, toutes les locomotives sont commandées avec le **protocole MFX** qui est
- **simple** à utiliser https://gelit.ch/Train/DirectMFX.ino
- **fiable** puisque chaque paquet, protégé par un CRC, sera ignoré par tous les décodeurs en cas d'erreur
- **robuste** car le paquet Periodic doit être envoyé régulièrement à chaque locomotive pour lui transmettre vitesse + F0-F15
- **efficient** puisque la charge maximum n'atteint que 25% lors de l'initialisation (getSID)

5) Les **wagons nettoyeurs Lux** sont utiles et planifiés tous les 3 mois
Des séquences spécifiques (va & vient) ont été développées pour le niveau -3

6) Les rails neufs Märklin ont besoin d'être **rodés** !!!
J'ai observé des accumulations noires autour de certaines roues
Un nettoyage méticuleux, à l'alcool à brûler, montre d'infimes résidus métaliques
Des traces très noires sont visibles sur le chiffon utilisé pour nettoyer ces rails à l'alccol

7) Tous les **frotteurs (Schleifer)** s'usent !!!
Une évidence qui m'a conduit à acquérir 2 exemplaires de réserve pour chaque locomotives et d'en changer 8 (sur 16 locomotives) pour les démos des fêtes de fin d'année

8) Certains wagons (Piko, NME, …) sont magnifiques
… mais exigent patience de rodage et … car ils ne possèdent pas la **qualité de Märklin !!!**

J'ai eu énormément de plaisir à développer les logiciels pour la carte Arduino Due et mon PC Windows 10 … avec parfois des bugs difficiles à corriger !

Les trains sont contrôlés exclusivement par une carte **Arduino Due** supportant les protocoles **MFX** et **MM2**
Les **210 mètres** de rails sont subdivisés en 65 blocs dont 58 autorisent le stationnement
Détection S88 (112 rétro) toutes les 40 ms avec charge CPU = 3-4-6 ms (Min-Moy-Max)
La maquette comprend **54 aiguilles**; 25 sont motorisées et commandées par M83
La **consommation totale maximum** est de l'ordre de **9A** sous 18V

L'interface utilisateur sous Windows :
- affiche l'occupation des blocs pour chaque train
- renseigne en temps réel (chaque seconde) sur la consommation des trains
  des sondes Arduino MKR transmettent l'information via Bluetooth Low Energy
- permet de privilégier les niveaux inférieurs ou supérieurs
- donne accès au mode Parking afin de ranger tous les trains

Logiciels développés par Gérald Litzistorf :
- 2500 lignes de C pour Arduino Due
- 1200 lignes de C + 400 lignes de XAML pour application Windows MFC

Mes vifs remerciements à l'entreprise Märklin pour l'excellente qualité de ses produits : rails, aiguilles, locomotives, wagons, décodeurs M83

Nov 2022 Détecter des anomalies de fonctionnement

La **bonne** gestion des trains consiste à donner à chaque **décodeur** de locomotive **la valeur souhaitée de vitesse au moment voulu.**

4 vitesses sont définies pour chaque locomotive (train) :

- 0 Arrêt
- LS Low Speed pour ralentir en vue d'un arrêt
- MS Medium Speed = vitesse par défaut
- HS High Speed pour certains blocs à certaines conditions

A titre d'exemple, la locomotive 460 simule la circulation d'un train aux vitesses de LS=25 km/h, MS=60 km/h et HS=100 km/h

Cette bonne gestion exige de connaître la position de chaque train ; ce qui est rendu possible avec la détection S88
Voir §5 de http://gelit.ch/Train/Raildue_F.pdf

- Les 210 mètres de rail sont subdivisés en blocs (cantons)
- Chaque bloc est délimité par 2 rétrosignalisations
- Pour un bloc de stationnement, la première rétro détectée permet de ralentir le train qui va s'arrêter à la détection de la seconde rétro
- Le ralentissement est effectif dès la détection ou après que tous les wagons ont franchi la détection
- Le(s) bloc(s) est(sont) **réservé(s)** avant chaque déplacement ; ce(s) bloc(s) doit(doivent) ensuite être **libéré(s)** dès que possible
- Voir la partie Déc 2021 pour l'implémentation logicielle

*Les **aiguilles (Turns)** doivent évidemment être positionnées correctement ce qui est garanti depuis juin 2022 avec l'utilisation des décodeurs M83.*
Voir paragraphe Fév 2022

Il m'a semblé utile de mesurer l'interval de temps pendant lequel une rétro est active
Ce temps dépend de la longueur, de la vitesse du train et de la longueur du rail de détection (de l'ordre de 36 cm)

Mon logiciel va d'abord **apprendre ces valeurs pour chaque train** ; valeurs que je vais ensuite mémoriser pour servir de **référence** afin de détecter un interval de temps inférieur à la référence

De plus, la commande de vitesse depuis l'arrêt n'est possible que si la bonne rétro est active
Je peux ainsi détecter des anomalies telles que :

- Le train a mis plus de temps que prévu pour passer la rétro
- Le train n'est pas parti → contrôler propreté rail, qualtié du frotteur, …
- Le train n'est pas sur la bonne rétro → arrêt de tous les trains en mouvement sur la prochaine détection

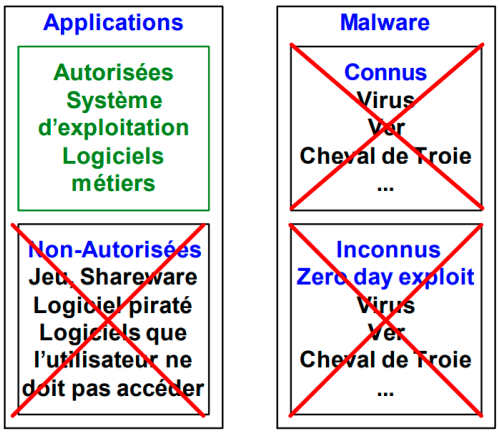
Les temps S88 mesurés en Oct 2022 ne sont pas modifiés : Min=3 ms Moy=4 ms Max=6 ms

Ce type de logiciel (d'algorithme) est de type **liste blanche (white list) :**

- **Tout ce qui est écrit est autorisé (et donc tout ce qui n'est pas écrit est interdit)**
- **on parle de modèle comportemental**

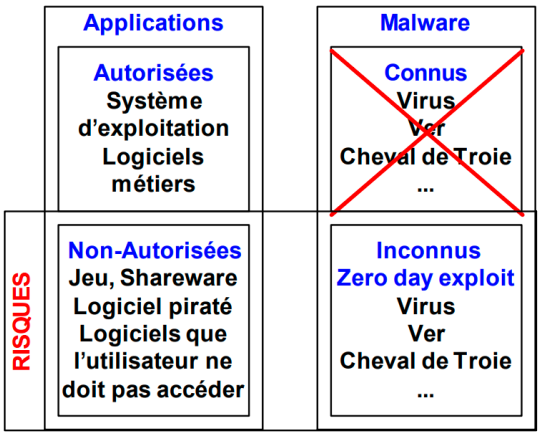
J'en profite pour illustrer ce principe essentiel de sécurité informatique :

Protection de type white list



Seules les applications autorisées sur votre PC ou Smartphone peuvent agir

Protection de type black list



Seuls les malwares connus de votre antivirus sont bloqués

Oct 2022 En route pour atteindre l'objectif de 20 trains en mouvement simultané

Principales activités :

- Mise en service de la boucle **Horaire** capable de supporter **5 trains**
- Mise en service de la boucle **Anti-Horaire** capable de supporter **8 trains**
- Ajout du 7ème module **S88**
  Détection S88 (112 rétro) toutes les 40 ms avec charge CPU = 3-4-6 ms (Min-Moy-Max)
- Ajout du 4ème **booster**
  Consommation totale maximum d'environ 9A sous 18V
- **Optimisation** de certains tronçons des Niv0 & Niv-3 pour éviter des attentes inutiles
- Commande de 3 locomotives pour disposer en 2023 des 20 trains
- Détection de l'absence de mouvement d'une loc. en mode automatique
  Beep + message sur PC pour résoudre manuellement le problème avant de changer le frotteur (Schleifer)
- Simplification des modes de fonctionnement
  Privilégier par défaut les niveaux bas
  Privilégier les niveaux haux
- Offrir à cette belle carrosserie

un chassis Märklin 37304 de qualité … Merci à Charly !!!

Eté 2022 Pause des activités H0 & Nouveaux développements LGB

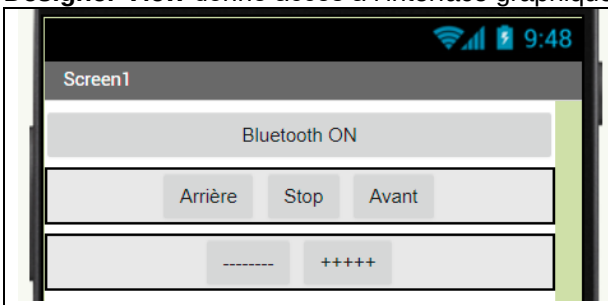
Développement d'une application Android sur la base des réalisations 2021 :

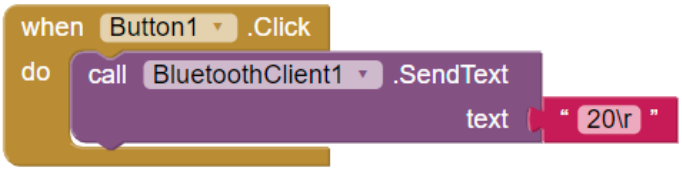
L'environnement http://ai2.appinventor.mit.edu est fabuleux pour le débutant en application mobile que je suis

1. Disposer d'un compte gmail

2. Utiliser un tutoriel http://appinventor.mit.edu/explore/ai2/beginner-videos
   La communauté est très importante avec des personnes parfois très jeunes !

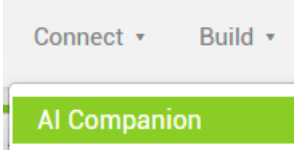
   Designer View donne accès à l'interface graphique ; **Blocks View** à la partie algorythmique





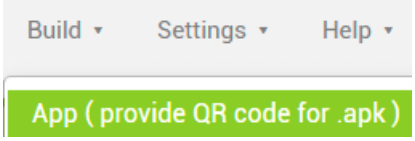
3. Installer MIT AI2 Companion sur le mobile Android
   https://play.google.com/store/apps/details?id=edu.mit.appinventor.aicompanion3&hl=fr_CH&gl=US
   ou utiliser Emulator (qui est lent et permet de s'affranchir d'un mobile)

4. Connect AI Companion



   Je n'ai pas réussi à utiliser l'option USB sur 3 mobiles Android différents
   http://appinventor.mit.edu/explore/ai2/setup

5. Build



6. Utiliser le QR code produit pour charger l'application sur le mobile

En conclusion

- Ce projet inité par Google → https://en.wikipedia.org/wiki/App_Inventor_for_Android permet **d'atteindre très rapidement un résultat** sans devoir passer par toutes les marches que les étudiants doivent gravir parfois dans nos écoles

- Il offre un excellent **niveau d'abstration** et une multitude d'applications rendues possibles grâce au couteau suisse qu'est le smartphone

- Pouvoir **accéder au matériel** pour commander un robot ou un train étend presque à l'infini les champs d'investigation

- Remercions quelques **informaticiens chevronnés** qui ont su développer cet outil !

Centrale DCC

En optant pour l'échelle LGB, j'ai choisi d'utiliser le protocole DCC (que je ne connaissais pas) car il est utilisé fréquemment pour commander les modèles réduits de diverses échelles.

Son protocole (en version de base) est extrèmement simple : https://www.nmra.org/sites/default/files/s-92-2004-07.pdf

Cet excellent lien fournit clarté et concision en vous évitant de devoir étudier la norme précédente
http://www.train35.fr/dcc2.html

Mon code comprend moins de 200 lignes en C
http://gelit.ch/Train/duedcc.ino

Il utilise pleinement les capacités PWM ainsi que le CPU 32 bit – 84 MHz Atmel SAM3X8E ARM Cortex-M3 utilisé sur la carte Arduino Due
De plus, cette carte Arduino Due est presque vendue au prix de l'Arduino Uno (25 x moins puissant)

Booster à la centrale DCC

Mes loc LGB consomment plus de 600 mA à vitesse moyenne ; ce qui est la limite du composant L293D

Je choisis comme booster le modèle BTS7960 qui :
- offre une interface électrique de 6 fils identique à L293D
- permet un courant de 43 A
- possède un refroidisseur
- et une surveillance thermique intégrée

Le circuit intégré utilisé provient de Infineon Allemagne (ex Siemens)

Câblage selon https://www.handsontec.com/dataspecs/module/BTS7960%20Motor%20Driver.pdf

Juin 2022 M83 le sauveur !!!

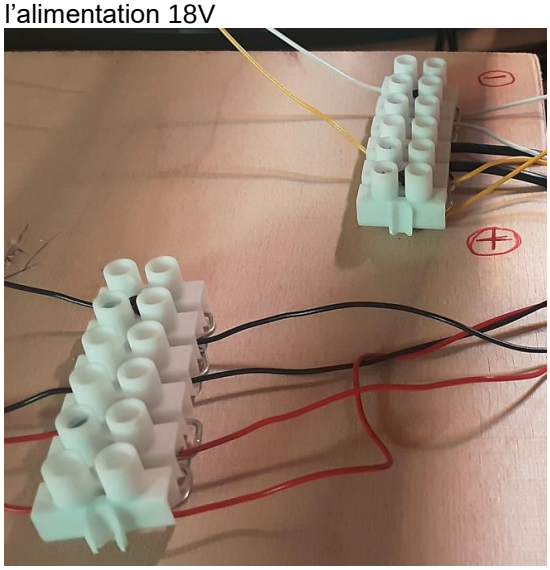
Mes 23 Turns m'ont toujours posé beaucoup de soucis !!!
Un jour sans erreur et le lendemain problèmesssssss alors que ni changement matériel & ni modification logicielle
A se demander si la température …

Je me décide donc à investiguer une autre solution basée sur le **décodeur Märklin M83**
Son **principal avantage** permet d'alimenter les Turns avec une **source continue 18V indépendante**
J'en ai profité pour tester que le Turn ne consomme plus 1.5A sur le rail
→ Plus besoin donc de réserver 1.5 A sur les 4.5 à disposition

Son principal défaut est d'exiger un câblage spécifique :

- 2 fils pour le relier au signal électrique du rail afin que le décodeur puisse recevoir le paquet MM2
- 2 fils pour l'alimentation 18V externe (j'utilise un module Märklin 66360 – 18V 2A pour 3 boîtiers M83)

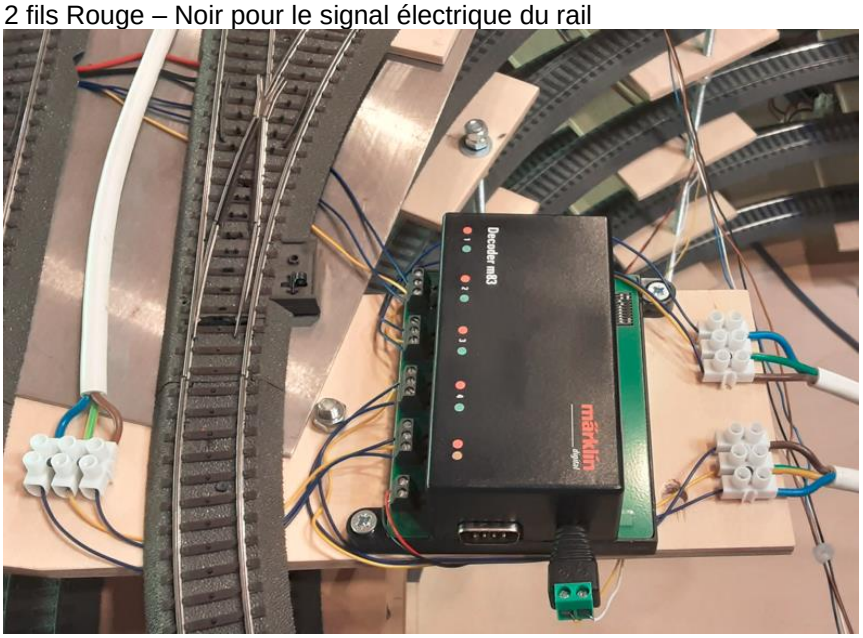
Ci-dessous,
Au premier plan barrette du signal électrique du rail
Au second plan barrette destinée à l'alimentation 18V



Ci-dessous, un module M83 relié à 4 Turns (3 fils par Turn)

Au premier plan connecteur mâle 5.5 x 2.5 pour l'alimentation 18V

2 fils Rouge – Noir pour le signal électrique du rail



En cas de distance importante (supérieure à la longueur des fils bleu – jaune fournis par Märklin), j'utilise un câble d'alimentation 230V de forte section
J'ai testé avec succès une ralonge de 150 cm

Le **bruit** produit par la commutation d'un Turn est révélateur du **courant disponible**
Je dispose d'une alimentation stabilisée (réalisée au début de mes études d'ingénieur)
En limitant le courant à 1 A, le Turn commute mais le bruit produit est différent de celui observé avec une limite de 2A
Il est moins sec (aigü) et dure donc plus longtemps
Ces observations confirment celles de mars 2021

Conclusion après 23 jours de tests :
Les 6 modules M83, mis en service le 6 juin 2022, fonctionnent à la perfection
Je conseille donc l'utilisation des décodeurs M83 pour leur fonctionnement parfait et leur prix inférieur aux décodeurs 74462

De plus j'estime que les boosters Littfinski DB4 ne sont pas compatibles avec ces décodeurs 74462 !!!
Cette installation comprenant 900 m de rails utilise les décodeurs 74462 sans problème MAIS avec des booster Märklin !!!

Mai 2022 Consommation transmise par Bluetooth Low Energy

Le booster Littfinski DB4 utilisé limite le courant consommé à 4,5 A **si JP5 est ouvert**
2 résistances mises en // produisent **225 mV pour 4,5 A**

Sachant qu'un Turn consomme environ 1.5 A (confirmé par mon alimentation 18V stabilisée) pendant 30 ms, j'affiche la consommation instantanée et mémorise la valeur maximale afin de respecter ce plan de consommation :

- 4.5 A TOTAL
- 0.5 A Marge de sécurité
- 1.5 A Consommation d'un Turn (verrou logiciel pour exclure la commutation de plusieurs Turns simultanément)
- 2.5 A A disposition pour la circulation des trains et alimenter les décodeurs
  A l'arrêt, la consommation totale des 15 trains (décodeurs) et des 23 décodeurs de Turn = 2.4 A !!!

La consommation [A] des trains à vitesse moyenne varie : Cargo=0.5 Mak=0.4 Re 4/4=0.3 Gothard=0.2 620=0.1

J'ai surtout appris que mon magnifique **TGV 37793 – 43423 – 43433 – 43443** comprenant 10 modules consomme 1 A en montée et 1.5 A à vitesse élevée

Les mesures de consommation sont essentielles pour garantir la bonne circulation des trains ; surtout au niveau des Turns car il ne pourra pas commuter en cas de surcharge

La maquette actuelle utilise **3 booster**
Les 3 consommations sont envoyées chaque seconde
Les 3 valeurs maximales sont mémorisées

1.68 1.10
1.32 1.00 1.34 2.53

Chaque booster alimente une zone composées de cantons 3 zones 7 zones
Toutes les zones sont isolées entre elles 4 zones

Le câblage en étoile évite les boucles
Les conducteurs ont un diamètre de 1 mm
Tous les Turns sont alimentés au plus prêt

Principaux objectifs :

1. Unifier les canaux de télécom
   Avant : Arduino Due port USB avec séparation galvanique + Arduino MKR & connexion wifi
   Après : 1 seul port USB utilisé pour N boosters
2. Investiguer la technologie Bluetooth Low Energy (**BLE**) supporté avec Arduino MKR
   Excellent document : https://www.silabs.com/documents/public/user-guides/ug103-14-fundamentals-ble.pdf
   Il était prévu initialement d'utiliser le chip Bluetooth de mon portable Lenovo T540p
   Après 2 jours à chercher du code fonctionnel pour Win10, j'ai préféré utiliser la carte MKR après avoir lu que le nombre élevé de dongle USB-Bluetooth utilisé dans le monde était certainement dû à la médiocrité des librairies Windows
3. Améliorer la mesure du courant en fixant la tension de référence à 225 mV
   Avec la résolution 8 bit par défaut, le courant consommé a un pas = 4500/255 = 17.6 mA

Le Bluetooth classique, utilisant par exemple l'excellent composant HC05
https://www.epfl.ch/campus/associations/list/robopoly/kit-prisme-extension/module-bluetooth/ permet une communication permanente point à point et ne permet donc pas de partager un canal de communication entre plusieurs nœuds.

BLE vise un monde de senseurs avec des volumes de données faibles.
Parmi les nombreux exemples proposés par la communauté Arduino, je me suis focalisé sur un mode particulier appelé Beacon → https://en.wikipedia.org/wiki/Bluetooth_low_energy_beacon

Chaque périphérique BLE doit commencer par envoyer un paquet **Advertise**
Un smartphone proche et compatible avec le service (mesure de température par exemple) pourra traiter cette information.

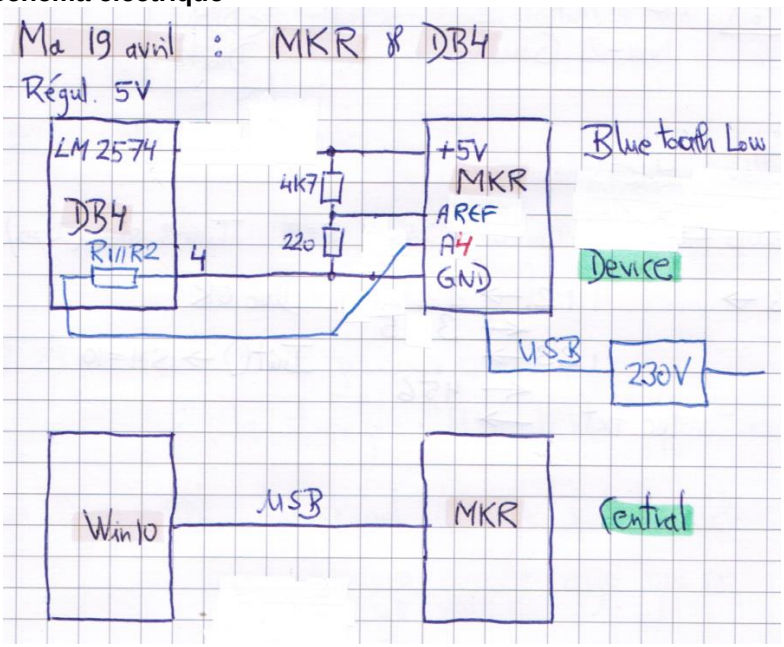
La librairie standard ArduinoBLE permet de transmettre des données dans ce paquet
`BLE.setManufacturerData(data,length)`
mais n'offre aucun moyen de lire ces données du côté opposé (smartphone ou Arduino)

La solution se trouve dans https://github.com/arduino-libraries/ArduinoBLE/pull/53
Il suffit de télécharger le fichier ZIP puis de placer son contenu dans le dossier C:\Users\...\Documents\Arduino\libraries

La carte **Arduino MKR wifi 1010** offre un excellent support matériel https://docs.arduino.cc/hardware/mkr-wifi-1010 :

- Processeur 32 bit cadencé à 48 MHz
- Espace RAM = 32 kByte
- Espace Flash = 256 kByte
- Module Nina W102 uBlox (développé en Suisse) compatible wifi et Bluetooth → https://www.u-blox.com/en

Schéma électrique



Le diviseur de tension fixe AREF = 5 x 220/4920 = 224 mV

Le régulateur 5V LM 2574 ne permet pas d'alimenter la carte MKR car la charge est trop élevée
→ alimentation externe 5V via USB

A4 mesure la chute de tension aux bornes de R1 // R2

Logiciel

Device (code unique pour tous les boosters DB4)

```
// 20 avril 2022
// from https://forum.arduino.cc/t/arduino-nano-33-ble-as-ibeacon/623266

#include <ArduinoBLE.h>

const int DATA_SIZE = 1;
uint8_t data[DATA_SIZE];

int N;
int Sum;
byte Value;
unsigned long  Time;

void setup() {
  pinMode(4,INPUT);
  analogReference(AR_EXTERNAL);  // use AREF pin
  Time = millis();
  N=0; Sum=0; Value=0;
  BLE.begin();
}

void loop() {
  N++;
  Sum=Sum+analogRead(A4);  // 1200 samples / seconde
  if (millis() > Time + 1000) { // sended each seconde
    Time = millis();
    Value = Sum/N;
    data[0] = Value;

    BLE.stopAdvertise();
    BLE.setManufacturerData(data,DATA_SIZE);
    BLE.setAdvertisingInterval(1280); // 0.8 seconde
    BLE.advertise();
    N=0; Sum=0;
  }
}
```

Central

```
// 21 avril
// from Examples - ArduinoBLE - Central - Scan
// Use Scan example to read device address

String Data;

#include <ArduinoBLE.h>

void setup() {
  Serial.begin(115200);

  BLE.begin();
  BLE.scan(true);
}

void loop() {
  BLEDevice peripheral = BLE.available();

  if (peripheral.address() == "84:0d:8e:34:84:6a") {
    Serial.print("G ");
    Send(peripheral);
  }

  if (peripheral.address() == "9c:9c:1f:e1:8b:7e") {
    Serial.print("D ");
    Send(peripheral);}
  }
}

void Send(BLEDevice peripheral) {
  if (peripheral.hasManufacturerData()) {
    Data = peripheral.manufacturerData();
    if (Data.length() == 2) {
      Serial.println(Data);
    }
  }
}
```

Avril 2022 Encore un BUG !!!

La bonne gestion des Turns a toujours été ma principale préoccupation !!!

Comment garantir une démo (avec 20 trains en mouvement simultané) sans problème ?

1. Les changements de pente doivent être doux pour éviter les **Court-Circuits**
2. **La position des Turns doit être garantie**
3. Le CPU doit être suffisament rapide : **Arduino Due OK**
4. Avoir une **alimentation** rigoureuse des rails
5. … ne pas perdre de wagons
6. … avoir des voies libres de tout objet
7. …

Le système de **FIFO** décrit en **déc 2021** a pour but de garantir que chaque commande Turn soit suivie d'une pause (500 ms) avant l'exécution de la prochaine commande Turn ou Speed.
Les autres locomotives ne doivent pas être impactées par cette pause.

Ce système doit être amélioré car il ne gère pas 2 (ou plusieurs) trains qui commandent des Turns !

En mode automatique, les trains se déplacent de bloc en bloc ; 2 cas se présentent :

1. Le train ne traverse aucun Turn
2. Le train traverse un ou plusieurs Turns

Le **cas 2** exige une pause à respecter après chaque commande Turn alors que le **cas 1** ne doit pas subir de retard !

Ce principe reste le même si 2 ou plusieurs trains utilisent des Turns !

- Chaque train utilisant un(des) Turn(s) doit terminer sa séquence : Turn - … - Speed
- Le logiciel (Core) garantit l'isolation des séquences : Loc1 – Loc5 – Loc20 – Loc3 – …

Il suffit donc de disposer de 2 FIFOs pour les procédures Zentrale, getSID, Periodic, Speed, …, Turn :

- Turn et les commandes Speed associées qui doivent respecter les pauses
- Zentrale, getSID,…, Speed

```
void Dequeue() {

  if (EnqN > FiMax-30) {Serial.println("Fifo FULL --> EnqN"); StopAll();}

  else if (Enq != Deq) { // Something ?
    ...
    switch (b) {
      default: Serial3.print("Error Dequeue-case"); Serial3.println(b); break;
      case 1 : ZentraleHW();                                            break;
      case 2 : getSIDHW(c);                                             break;
      case 3 : PeriodicHW(c);                                           break;
      case 4 : ... SpeedHW(c,a);                                        break;
      case 6 : ... FuncHW(c,a,1); ... FuncHW(c,a,0);}                   break;
    }
    Deq++; if (Deq == FiMax) {Deq=0;}  EnqN--;
  }

  if (Pause != 0) {if (millis() > Pause) {Pause=0;}}   // Wait

  else if (EnqTN > FiMax-30) {Serial.println("Fifo FULL --> EnqTN"); StopAll();}

  else if (EnqT != DeqT) {   // Something ?
    ...
    switch (b) {
      default: Serial3.print("Error DequeueT-case"); Serial3.println(b);     break;
      case 4 : ... SpeedHW(c,a);                                            break;
      case 7 : ... TurnHW(a,0); ... {TurnHW(a,1);}   Pause = millis() + 250; break;
                                                      // TURN TIMEOUT = 250 ms
    }
    DeqT++; if (DeqT == FiMax) {DeqT=0;} EnqTN--;
  }
}
```

Ce scénario de validation utilise les locomotives **3**, **7** et **11 avec un Timeout = 500 ms**

EnqN=0 EnqNMax=2 EnqTN=0

```
3 A69_37-10                    Loc 3 to C37 --> T1 T2
EnqT Turn=1                    mise en queue T1
TurnQ=1
EnqT Turn=2                    mise en queue T2
TurnQ=1
EnqT Speed Loc=3  Vit=2        mise en queue Speed
TurnQ=0                        Loc 3 cycle terminé (EnqTN=3)

7 H59_34-89                    Loc 7 to C34 --> T10 T11 T12
EnqT Turn=12                   mise en queue T12
TurnQ=1
EnqT Turn=11                   mise en queue T11
TurnQ=1
EnqT Turn=10                   mise en queue T10
TurnQ=1
EnqT Speed Loc=7  Vit=2        mise en queue Speed
TurnQ=0                        Loc 7 cycle terminé (EnqTN=7)

11 H66_64-13                   Loc 11 to C13 --> NO turn --> utiliser Enq
Enq Speed Loc=11  Vit=2        mise en queue Speed
SpeedHW-11-2                   Loc 11 démarre
1 ms
TurnHW-1 0                     exécution T1
502 ms
TurnHW-2 0                     exécution T2
505 ms
SpeedHW-3-2                    Loc 3 démarre
0 ms
TurnHW-12 0 --> T12            exécution T12
501 ms
TurnHW-11 1 --> T11            exécution T11
501 ms
TurnHW-10 1 --> T10            exécution T10
501 ms
SpeedHW-7-2                    Loc 7 démarre

Enq Speed Loc=11  Vit=0        Loc 11 a atteint la détection de fin du bloc C13
SpeedHW-11-0                   Loc 11 à l'arrêt

11 H64_62-15                   Prochaine destination pour Loc 11
Enq Speed Loc=11  Vit=2
2 ms
SpeedHW-11-2
```

Conclusion :

- **Loc 11 démarre logiquement en premier puisqu'elle n'utlise pas de Turn**
- **Loc 3 démarre avant Loc 7 car Core traite les locs en séquence (for Loc=1, Loc<=LoMax, Loc++) …**
- **5 pauses égales à 500 ms sont présentes ; une par Turn**
- **Le compteur EnqTNMax=7 confirme les 7 mises en queue**
- **Le compteur EnqNMax=2 reste à des valeurs normales**
- **… Ne pas oublier l'envoi de Periodic chaque 25 ms qui n'apparait pas dans ce logs MAIS qui est bien envoyé sinon les loc. refusent d'avancer**

Mars 2022 Je me convertis au MFX

J'ai enfin succombé aux charmes du procotole MFX !!!

MFX est une marque déposée pas Märklin
https://www.maerklin.de/de/produkte/produkt-informationen/modellbahn-steuerung/mfx
Cette technologie impose des décodeurs spécifiques qui possèdent un **identifiant unique** (UID) propre à chaque locomotive.
Elle passe par une phase d'apprentisage dans laquelle la Mobile Station va lire et mémoriser les principaux paramètres d'une locomotive.

Ce projet, démarré le 28 déc 2021, a été très enrichissant !!!
Il m'a permis de comprendre le dialogue entre ma centrale Mobile Station 2 et un décodeur sur la base de **l'excellent document produit par Stefan Krauss et son équipe :** http://www.skrauss.de/modellbahn/Schienenformat.pdf

J'ai d'abord voulu comparer les paquets produits par ma Mobile Station 2 et une locomotive avec ce document.

En réutilisant l'optocoupleur du §3 http://gelit.ch/Train/DirectMM2.pdf , j'ai longtemps cherché à acquérir les bonnes trames !!!
Même avec un code d'interruption minimaliste, il n'est pas possible d'extraire les trames (durée du bit = 100 µs) avec un Arduino Due cadencé à 84 MHz (et un suréchantillonage de 10)

Heureusement, mon analyseur logique https://www.lab-nation.com/ offre des extensions très intéressantes https://wiki.lab-nation.com/index.php/Creating_your_own_Protocol_Decoder qui m'ont parmis d'observer les trames décrites par Stefan.

Fan du PWM implanté dans l'Arduino Due, j'ai pensé le réutiliser !?
A la réflexion, produire le signal MFX sans PWM avec digitalWrite est plus efficient … et plus simple ; surtout pour les changements de polarité.

Je me suis un peu cassé les dents sur l'implémentation du CRC ; voir ligne 316 de http://gelit.ch/Train/DirectMFX.ino

Heureusement que internet regorge de trésors comme le **gigantesque développement de Daniel Sigg** :
https://siggsoftware.ch/wordpress/category/modellbahn/

Mes vifs remerciements vont à Stefan et Daniel pour les précieux conseils qu'ils m'ont donnés !!!

Mon code Arduino Due mis à disposition se veut minimaliste et exige la présence de la Gleisbox pour lire l'UID mémorisé dans le décodeur et pour l'activer en mode MFX si la locomotive fonctionnait auparavent en MM2

Il peut permettre à un hobbyist de démarrer simplement avec :
- 1 aiguillage Adresse=5 en ligne 80
- 2 locomotives dont les UIDs doivent être modifiés (voir page suivante)
- et 16 rétrosignalisations

Les rétrosignalisations S88 (lignes 359 à 376) sont optionnelles
Elles sont lues chaque seconde (ligne 214) alors que mon code personnel lit 96 rétro chaque 20 ms
Le résultat est affiché avec la commande s (=Statistiques)
Les pins 10 à 13 sont utilisées (lignes 13 à 16) :
- `const int CLOCK = 10;`
- `const int DATA = 11;`
- `const int RESET = 12;`
- `const int LOAD = 13;`

Au niveau matériel, ne pas oublier d'ajouter le composant L293 à la carte Arduino Due
Voir §10 de http://gelit.ch/Train/DirectMM2.pdf
Les pins 2 et 3 de Arduino Due sont utilisées :
- `const int Pin2_L293 = 2;`
- `const int Pin7_L293 = 3;`

La pins 8 active ou non le signal électrique sur le rail :
- `const int PowerPin = 8;`

La pins 9 est optionnelle et renseigne sur l'état (PowerOFF ou PowerON) du rail :
- `const int Red_Led = 9;`

Des tests intensifs sur plusieurs semaines ont permis de valider ce code avec ma version qui gère 15 locomotives compatibles MFX.

J'en ai profité pour remplacer le décodeur du modèle 29486 qui ne supporte pas MFX
https://www.maerklin.de/en/products/details/article/29486
Mon choix s'est porté sur le décodeur ESU LokPilot 5 → https://www.esu.eu/en/products/lokpilot/lokpilot-5/
Son UID se trouve en ligne 77.

A propos de http://gelit.ch/Train/DirectMFX.ino :

- Ne pas oublier d'ajouter la librairie DueTimer dans Arduino IDE (Tools – Manage Libraries…)
- Utilisation de machines d'états (SM = State Machine) pour Interrupt et loop
- Dans **loop** (lignes 186-218), la séquence S=2 à S=5 respecte les timings (delay) en émettant les trames MFX après PowerON ; l'utilisateur active PowerON avec Enter.

  Le compteur Za utilisé par Zentrale() est incrémenté après chaque getSID mais n'est pas mémorisé lors du PowerOFF car tous les décodeurs acceptent de toujours démarrer avec Za=1.

  getSID alloue une adresse dynamique de 7 bit à l'UID correspondant.
  Ces UID (lignes 76-77) doivent être modifiés pour correspondre aux locomotives utilisées
  Voir procédure dans §4.1 de http://gelit.ch/Train/Raildue_F.pdf

  L'état S=5 impose l'envoi de Zentrale toutes les 500 ms et Periodic toutes les 50 ms
  Des conflits sur la mémoire tampon B[] sont donc possibles et gérés par Busy

  La procédure Periodic est indispensable au bon fonctionnement ; elle envoie à toutes les locomotives enregistrées les valeurs de vitesse (133), direction (132) et les fonctions F0 à F15 (lignes 137-138)
- La procédure Func (112) est très utile car elle permet d'activer une seule fonction parmi 128 possibles (F0 à F127)
- Speed (ligne 100) n'utilise que 3 bits pour les vitesses Speed(0) à Speed(7)
- **L'interruption** (243-307) est exécuté chaque 50 µs (MFX) ou chaque 104 µs (Turn en mode MM2) ; le signal électrique généré par la carte Arduino présente 3 cas :

  Emission continue du caractère de synchronisation (SM=10 en ligne 282)
  Maintient le synchronisme des décodeurs, fournit l'énergie pour le mouvement, l'éclairage et le son des locs

  Emission d'une trame MFX (SM=20 SM=30)
  Longueur = variable Len

  Envoi d'une commande MM2 destiné à un Turn (SM=3 à 8)
  Pause=6.25 ms + 18 bit à 104 µs + Pause=1.5 ms + 18 bit à 104 µs + Pause=6.18 ms = 17.7 ms

  Il m'a semblé utile de connaître l'occupation due aux trames MFX sur le rail
  Mesuré avec un interval de 1 seconde, les trames MFX occupent un maximum de 40% à l'initialisation (getSID) et ne dépassent pas 20% avec mes 15 trains en mouvement simultané
  L'occupation due aux Turns n'est pas significative car chaque commande subit une pause = 250 ms (ligne 228)
- Les **statistiques** (commande s) affichent :

  Les valeurs des variables
  ```
  Power=1   S=5   SM=10
  ```

  Détail de la dernière trame MFX envoyée
  ```
  Len=48    CRC=93
   1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
   0 1 0 0 1 1
  ```

  Détail de la dernière trame MM2 envoyée
  ```
  MM2= 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
  ```

  Détail des 16 bits de rétrosignalisation
  ```
  S88  1=1 2=1 3=1 4=0 5=0 6=0 7=0 8=0 9=0 10=0 11=0 12=0 13=0 14=0 15=0 16=0
  ```

Fév 2022 Dispositifs Anti-collisions

En mode automatique, avec 15 trains en mouvement simultané, il est malheureusement possible qu'un aiguillage (turn) soit mal positionné !!!

J'ai développé **2 stratégies** pour **détecter** et **éviter** ce type de problème :

1. Chaque train s'arrêtant en fin de bloc, il est facile de **détecter la présence de 2 trains sur le même bloc**

2. Le **Turn 5** sépare 3 blocs ; il est donc possible de **détecter sa mauvaise position grâce aux rétro**

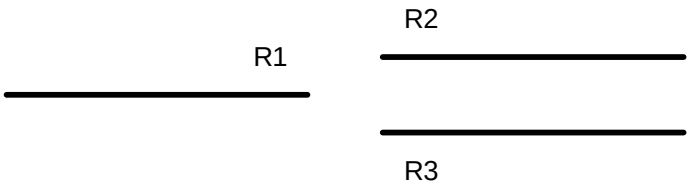


Illustration pour le Turn 5

```
const byte TurnMax = 24;    // 24 physical turn with MOTOR

volatile bool TurnDir[TurnMax+2];
volatile byte TurnLoc[TurnMax+2];

if (TurnLoc[5]!=0 && TurnDir[5]) {
  if (R[2]) {TurnLoc[5]=0;}
  if (R[3]) {Speed(TurnLoc[5],0); Serial.print("Error Turn 5-1");}
}

if (TurnLoc[5]!=0 && !TurnDir[5]) {
  if (R[3]) {TurnLoc[5]=0;}
  if (R[2]) {Speed(TurnLoc[5],0); Serial.print("Error Turn 5-0");}
}
```

Les variables `TurnLoc[]` et `TurnDir[]` sont affectées dans la procédure Turn() :

```
void Turn(byte device, bool dir)
...
TurnDir[device]=dir;
TurnLoc[device]=D;
...
```

Finalement le meilleur moyen de l'éviter est d'alimenter ces turns au plus près du booster car ils consomment énormément de courant → voir Mars 2021

Jan 2022 Bonheur & Malheur

Les plaisirs offerts par mes trains sont multiples :

- Réfléchir à un plan de réseau
  Etudier des solutions sur internet et dans des revues
- Simuler par logiciel ce plan de réseau pour en connaître les limites
  Est-il envisageable d'avoir 20 trains en mouvement simultané ?
  Quel est le coût financier ?
- Construire la maquette étape par étape (étage par étage)
  Garder à l'esprit la vue d'ensemble
  Tester minutieusement les points critiques (pente, aiguille, rétrosignalisation, …)
- **Développer pour chaque étape les logiciels appropriés (mon travail préféré)**
  Prendre en compte les tests unitaires
  Trouver la bonne parade face aux bugs (debug, remontée d'information, …)
- Documenter les points critiques
  Rétrosignalisation, bloc, aiguille, locomotive, …, logiciel
- Effectuer des démos aux invités
  La complexité de l'ensemble n'est plus la même en 2021 qu'en 2017 !
  Une démo parfaite exige un comportement parfait de tous les maillons de la chaîne (loc, rail, turn, logiciels, …
  Alors que les logiciels restent éternels au niveau fonctionnel, le matériel ferroviaire demande du soin (aspirer les voies régulièrement, nettoyer à l'alcool les rails en cas de mauvais contact, ajuster la mécanique d'une aiguille, changer un moteur d'aiguille, … mettre à la poubelle une locomotive)
- Observer l'évolution des petit-fils … l'ainé détecte un bruit anormal (wagon hors rail) à la seconde !!!

Pour conclure ces 5 dernières années, **je dois remercier Märklin pour la qualité des produits vendus !**

- Le logiciel de dessin Gleisplanung facilite grandement l'ébauche de la maquette
- Les rails voie C sont très robustes et s'interconnectent facilement (même pour mon petit fils de 5 ans)
- Les aiguilles sont fiables mécaniquement
- Prévoir des moteurs d'aiguille et des décodeurs de réserve en cas de panne
- Les locomotives sont lourdes et très fiables (le service après-vente répare dans un délai hélas important)
- Les wagons sont parfaits

Je resterai donc fidèle à cette marque et veux témoigner de la médiocrié de certains produits concurrents :

- **Bemo** avec sa magnifique locomotive Glacier Express facturée CHF 450.- le 12 mai 2020

https://www.bemo-modellbahn.de/produkte/produkt-detail.html?tx_userbemocatalogue_rubriclist%5Bitem%5D=1675&tx_userbemocatalogue_rubriclist%5Baction%5D=show&tx_userbemocatalogue_rubriclist%5Bcontroller%5D=Item&cHash=3de974bf47de293f81a89038c4fd2751

Un relais, ajouté pour la lumière dans les wagons, n'était pas bien isolé
Après 3 mois en usine, **elle a fonctionné moins de 20 minutes (après avoir perdu 3 dents)**

Une bonne raison de ne plus perdre de temps avec ce triste produit !!!!!!

Voir les 2 pages suivantes

4 fév 2021

To
Bemo Kundendienst

From
Litzistorf Gérald – litzis@bluewin.ch
gelit.ch

Frage : Kann ich mein Lok RhB Ge 4/4 III 651 modifizieren ?

Guten Tag,

Mein Bemo Lok fährt problemlos wenn die Steigung = 0 oder positiv

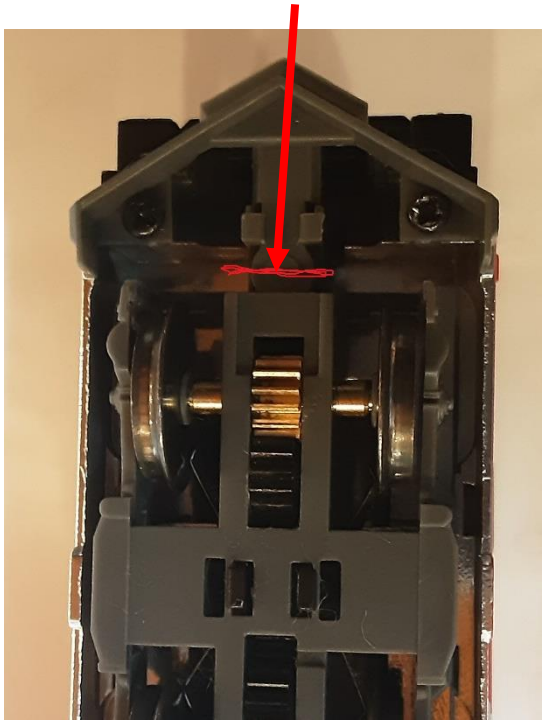
Aber wenn sie mit eine Steigung kleiner als -1% abfährt, entgleist sie oft auf die Weiche
Es ist sehr schade für eine Firma, die **BergLok** produziert !!!

Meine 12 andere Lok (Märklin, Esu, Liliput) haben nicht dieses Problem

Warum ?
Ich mache ein Vergleich mit Märklin 37304 Lion

Spiel = 1,7 mm Preis = 325 CHF	Spiel = 1,4 mm Preis = 450 CHF
	

Frage : Kann ich mein Lok RhB Ge 4/4 III 651 modifizieren; um mehr Spiel zu haben ?
Was passiert wenn ich hier schneide ?



Sie können mein Lok sehr einfach mit die MFX Werte = UID = FA 00 19 4B

Besten Dank für Ihr Antwort

Malgré plusieurs échanges de mail, Bemo n'a jamais voulu reconnaître ce problème !!!!!!!!!!!!!!!!

- **ESU** avec sa magnifique locomotive, facturée CHF 439.- le 7 nov 2015 que j'ai renvoyée à l'usine car elle était incompatible avec la détection S88 pour les systèmes Märklin 3 rails

Elle doit circuler en marche arrière car seuls 2 axes ont été modifiés

- **Liliput SBB Kiss Dosto** facturé CHF 650.- en sept 2015

Extérieur magnifique mais intérieur misérable ; surtout le pauvre moteur
Les bandages quittent les roues rapidement
Les couplages ne sont que source de problème à CHF 100.- la panne
Une misère à l'état pur fabriqué en Chine !

Heureusement tous les produits non Märklin ne sont pas médiocres à l'image du **wagon de mesure Pico** et des produits Littfinski (booster & rétrosignalisation)

Déc 2021 Etape 5 : Niveau 0 (Construction – Logiciels – Tests)

Pose des rails & turns pour réaliser la maquette de la **fig 1**
Longueur de rails ajoutée = 59 mètre pour une **longueur totale = 208 mètre**
Maquette finale prévue pour **20 trains** → chaque train dispose de 10 mètre répartis sur plusieurs blocs

Arduino Due gère le contrôle des 20 trains
Maquette comprend :

- 51 aiguilles (Turn) dont 24 sont motorisées et donc équipées d'un décodeur
- 53 blocs utilisés pour Niv0=9 Niv–1=Niv–2=6 Niv–3=8 HeliG=HeliD=12
  47 possèdent une longueur suffisante pour garer un train de longueur inférieure à 286 cm (futur TGV de 10 modules mesurera 231 cm)
- 84 détections S88 pour Niv0=20 Niv–1=Niv–2=12 Niv–3=16 HeliG=HeliD=12 via 6 modules Littfinski pour créer ces blocs

Principales fonctions :

- **Speed(Loc,Value)**
  Loc:1-20 Value:0=Stop/1=LowSpeed/2=NormalSpeed/3=HighSpeed
- **SeT(Device,Value)**
  Device:1-24 Value:0/1
- **S88 gère les événements suivants :**
  Sto : Stop Loc
  SLO : SLOW → ralentir dès détection
  Slo : Slow → ralentir une fois que le train a passé la détection
  Ren : Rendre Bloc(s)

Chemins :

```
H60_1[]   = "H60_1-250" → sens Horaire du (From) Bloc 60 vers (To) le bloc 1
                          Respectant conditions 250

case 250:  if (!Z) {if   (R[25] && C[18]==0) {Path[W] = P; W++;}}  // Exclusion T22
            else                    {C[18]=D;}
           1ère ligne exécutée pour évaluer si le bloc 18 est libre et si la
           rétrosignalisation 25 (placée au début du bloc 60) est occupée
           2ème ligne exécutée pour bloquer la ressource (bloc 18)

case 1   : Q(&H1_6[0]); Q(&H1_7[0]); Q(&H1_8[0]); Q(&H1_9[0]); Q(&H1_51[0]);
           Loc avec sens Horaire dans bloc 1 : 5 chemins possibles
           Chaque chemin est évalué
           Choix aléatoire si plusieurs chemins possibles
```

Actions :

```
if sens Horaire et
      From Bloc 60 : Ren[25]=From → Détection 25 → rendre Bloc 60
      To   Bloc 1  : SLO[63]=Loc → Détection 63 → Ralentir Loc
                     Sto[52]=Loc → Détection 52 → Stoper Loc
```

Mise à jour du GUI :

```
for (int a=0; a<CMax; a++) {
  if (C[a] != Last_C[a]) { // if change
    if (C[a]==0) {Serial.print(90); if (a<10) {Serial.print("0");} Serial.println(a);}
                  // 90 --> GUI : Canton Hidden
    else         {Serial.print(91); if (a<10) {Serial.print("0");} Serial.println(a); …}
                  // 91 --> GUI : Canton Visible
  }
  Last_C[a] = C[a];
}
```

Principales difficultées rencontrées :

1. Simplifier l'interface utilisateur avec des raccourcis clavier pour fiabiliser la conduite des trains
   L (=Learn) pour déplacer un train sur la prochaine détection afin de connaître le bloc occupé
   0 (=Stop All) pour arrêter tous les trains en cas de problème
   C (=Clear) pour autoriser les trains arrêtés manuellement à terminer leur trajet
   Backspace (=Arrêt d'urgence) qui impose 0 volt sur les rails

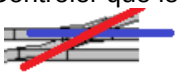
2. Conserver la position des trains sur le PC dans le fichier D:\loc20.txt
   0103A Loc 1 sur Bloc 3 dans le sens Anti-Horaire
   0302H Loc 3 sur Bloc 2 dans le sens Horaire
   1399 Loc 13 pas présente physiquement sur la maquette

3. Tester tous les chemins possibles
   Subdiviser les tests en créant des modes de fonctionnement du Niv 0 :
   1. 1 seule voie par sens
   2. 2 voies par sens
   3. Idem 2 + voie de retournement
   4. Idem 3 + équilibrage (voir §3 ci-dessous)
   5. Idem 4 en occupant Niv-3 puis Niv-2 puis Niv-1 = mode privilégié !
   6. Idem 4 en occupant Niv-1 puis Niv-2 puis Niv-3

4. Autoriser les trains à circuler en vitesse rapide sur le niveau 0 avec raccourci clavier **H (=High)**
   Positionner les détections de freinage en conséquence

5. Assurer un équilibrage des trains (sens horaire & anti-horaire)
   Actuellement avec 17 trains présents, il ne doit pas y avoir plus de 10 trains dans un sens
   A terme (avec 20 trains), l'équilibrage doit être parfait pour que chaque train trouve une place de parking pour la nuit

6. Ajouter le raccourci clavier **P (=Parking)** pour ranger les trains de la manière la plus rapide

7. Contrôler que les verrous de turn empêchent les collisions de trains
   
   Le premier train (bleu par exemple) bloque le train rouge
   Création d'un bloc pour assurer cette exclusion mutuelle

8. Des turns ne sont pas positionnés correctement alors que les commandes **`SeT`** sont justes
   Cas d'erreur observés avec 10 – 15 trains en mouvement et une consommation = 2.4 - 2.8 A
   Détection par logiciel de ces cas d'erreur grâce aux détections S88 puis **Stop All**
   Détection de la présence de 2 trains sur le même bloc grâce aux détections S88 puis **Stop All**
   Confirmation que le booster DB-4 n'a pas la capacité dynamique d'ajouter 1.4 A en quelques ms (voir mesures de mars 2021) alors qu'il délivre 2.4 - 2.8 A
   ➔ **Décision d'ajouter un second booster !!!**

9. La valeur du courant fourni par ce booster est transmise par wifi entre une carte Arduino MKR configuée comme Access Point et le PC

10. Subdiviser l'alimentation des rails en 2 zones (1 booster DB4 par zone)
    Zone **Gauche** = Héli Gauche + Niv0 Gauche + Niv–1 + Niv–2
    Zone **Droite** = Héli Droite + Niv0 Droite + Niv–3

11. Tester la bonne gestion des court-circuits de chaque zone

12. Rendre la gestion des ports S88 indépendante de la gestion du PWM
    Utiliser le câblage proposé dans http://gelit.ch/Train/DirectRailDesktop.pdf
    Les sorties analogiques (A0-A4) sont utilisées en mode digital :
    ```
    pinMode(PowerPin, OUTPUT); digitalWrite(PowerPin,0); // disable L293D
    pinMode(CLOCK, OUTPUT); pinMode(LOAD, OUTPUT); pinMode(RESET, OUTPUT);  // S88
    ```
    Elles ne sont pas concernées par le multiplexage très complexe du processeur SAM3X
    Voir §9.3 de
    https://ww1.microchip.com/downloads/en/devicedoc/atmel-11057-32-bit-cortex-m3-microcontroller-sam3x-sam3a_datasheet.pdf

13. Retrouver la bonne ligne de code dans les
    - 1800 lignes en C dans fichier .ino (Arduino Due)
    - 900 lignes en C# dans fichier .cs (Windows 10)
    - 360 lignes du fichier .xaml (Windows 10)

Structures de données utilisées :

1. Les variables suivantes sont associées à une locomotive :

```
struct LoS {
  int  Pause;          // Attente pour Turn & Function
  bool Status;         // 0=Disable / 1=Enable
  byte Stop;           // 1=Arrêté / 0=En mouvement / 2=...
  bool StopM ;         // Mémoriser Speed if StopAll
  int  Pos;            // Position (bloc ou canton) actuel
  byte Adr;            // Adresse MM2
  bool Dir;            // Direction : 1=Forward / 0=Reverse
  bool Sens;           // 1=Horaire / 0=Anti-horaire
  byte H;              // High   Speed
  byte M;              // Medium Speed
  byte L;              // Low    Speed
  int  Enq;            // Enqueue (voir ci-dessous)
  int  Deq;            // Dequeue (idem)
  bool Light;          // Clignotement en mode Learn / ON si Status=1
  bool Sound;          // 1=Loc est compatible
};
```

2. Mémoires tampon circulaire de type FIFO (First IN – First OUT)

   Chaque locomotive possède son espace fixé à 9 commandes pour mémoriser les commandes d'un déplacement
   `H60_6[]: SeT(21,0); SeT(22,0); Speed(Loc,MS)`

   Ces 3 commandes sont mises en queue (Enqueue) afin d'être exécutées dans cette séquence

```
void Speed(int l, int v) {
  ...
  Cmd[l][Lo[l].Enq] = v;
  Lo[l].Enq++; if (Lo[l].Enq == 9) {Lo[l].Enq=0;}
}
```

   Chaque commande Turn est suivie d'une pause paramétrable fixée à 500 ms (½ sec)
   Les autres trains ne subissent aucune pause car le mécanisme est propre à chaque locomotive

   Ces 20 queues sont évaluées en permanence afin d'exécuter une commande au bon moment (avec Dequeue)

```
void Dequeue() {
  for (D=1; D <= LoMax; D++) {
    if (Lo[D].Pause != 0) {if (millis() > Lo[D].Pause) {Lo[D].Pause=0;}}  // Wait
    else if (Lo[D].Enq != Lo[D].Deq) {    // Cmd to execute
      if      ((Cmd[D][Lo[D].Deq] >= 0) && (Cmd[D][Lo[D].Deq] <= 3))  {SpeedHW(D);}
      ...                                                              {SoundHW(D,1);}
      else {  // Turn Cmd
        int b = Cmd[D][Lo[D].Deq];
        ...
        if (...) {... TurnHW(b,1);} else {TurnHW(b,0);}
        Lo[D].Pause = millis() + 500;    // TURN TIMEOUT = 500 ms
      }
      Lo[D].Deq++; if (Lo[D].Deq == 9) {Lo[D].Deq=0;}
    }
  }
}
```

3. L'ensemble des commandes (issues des 20 locomotives) est placée dans une autre mémoire tampon circulaire afin de les traiter dans le bon ordre temporel en ne privilégiant ainsi aucune locomotive

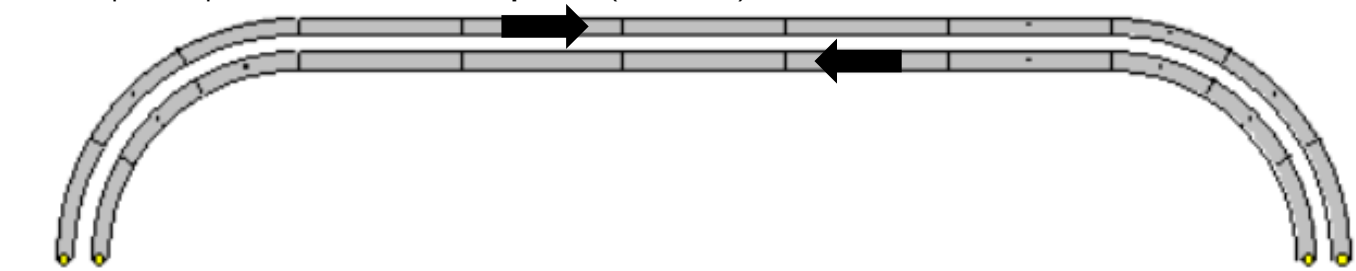
   En bout de chaîne la cellule PWM, capable de prendre en charge une nouvelle commande toutes les 13 ms, les exécute pour produire le signal électrique du rail (voir §3 de http://gelit.ch/Train/DirectRail.pdf)

4. La perfection est obtenue en diminuant l'interval de temps entre les "doppel Paket" de 7 à 4 bit

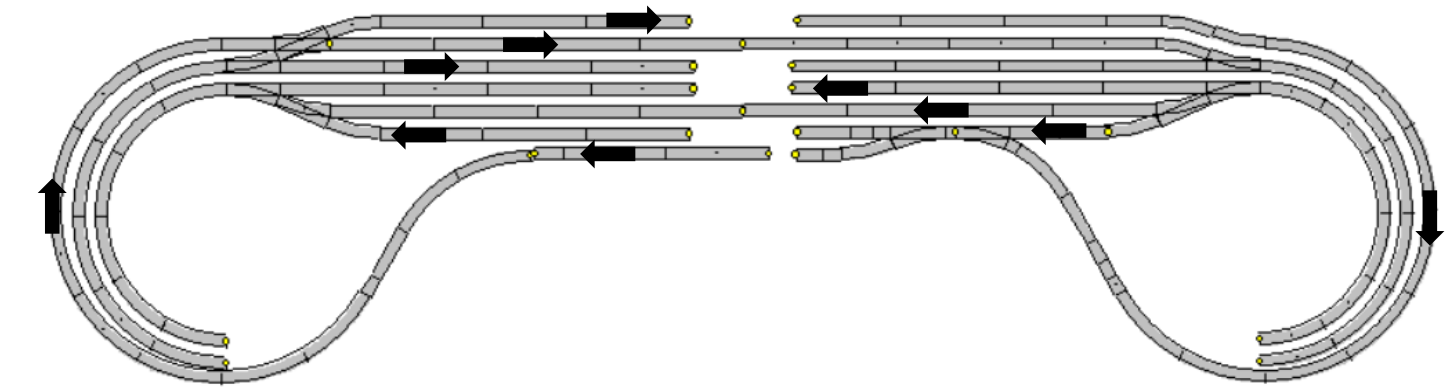
   J'ai mis un certain temps pour comprendre la raison pour laquelle 3 locomotives (les plus chères) ignoraient parfois une commande Speed

Oct 2021 Etape 5 : Niveau 0 (Design)

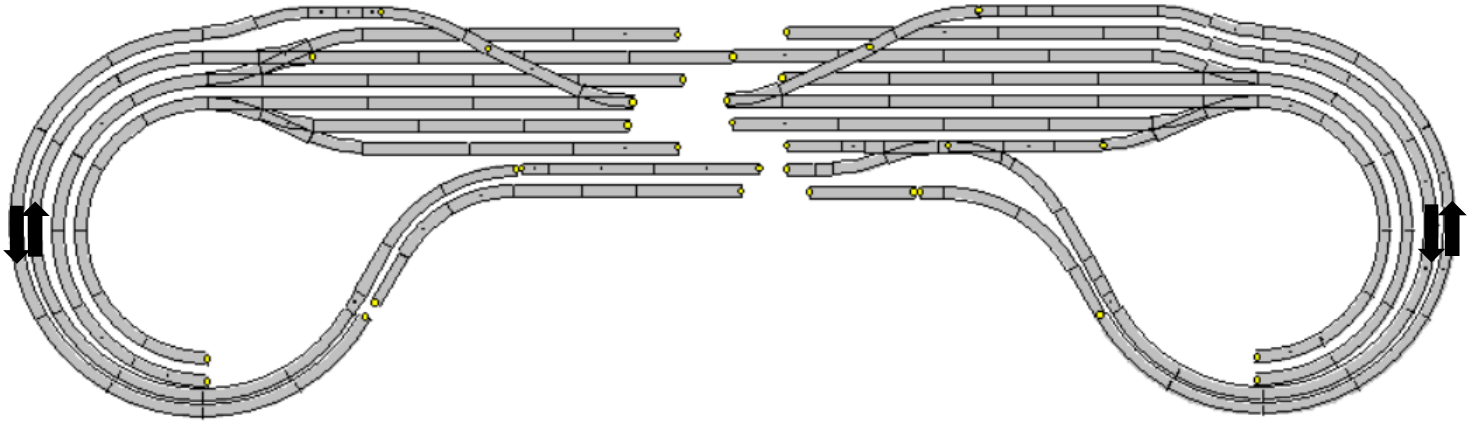
Lors des phases précédentes le **niveau supérieur** (=Niveau 0), reliant les 2 hélicoïdaux, était minimaliste :



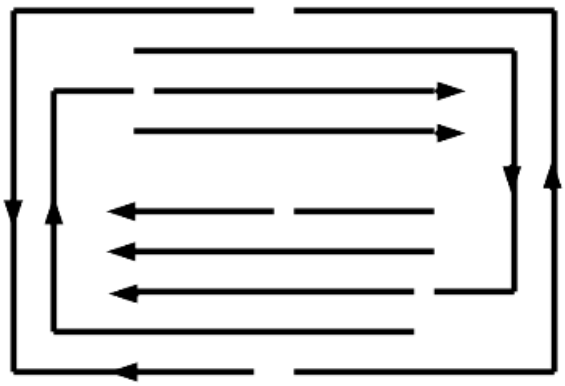
Le plan retenu (**fig 1**) doit doubler les nombre de trains en mouvements simultanés et offrir 2 boucles de retournement :



Il ajoute la circulation de 2 trains supplémentaires dans le sens opposé des boucles de retournement (**fig 2**) :



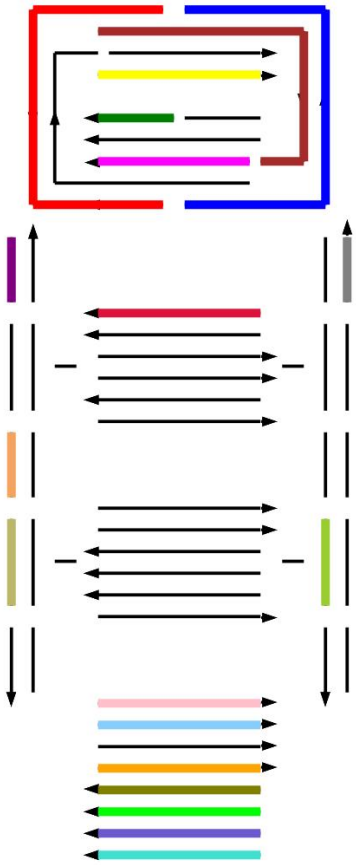
Affichage du Niveau 0 (comprenant 10 blocs) sur le **PC Windows** :



L'objectif étant de **simuler** la circulation des **20 trains** garés aux 3 étages inférieurs

Oct 2021 Etape 5 : Niveau 0 (Simulation)

Dans l'exemple ci-dessous, **20 locomotives sont présentes virtuellement** (puisque le Niveau 0 n'est pas encore construit) :



6 → **6 Loc présentes au Niveau 0**

◄ 12 8 ► **8 Loc circulent dans le Sens Horaire (dont Loc jaune)**

12 Loc circulent dans le Sens Anti-Horaire

- **Loc Grise** se situe dans la montée entre Niv –1 et Niv 0
- **Loc Brune**, présente au Niv 0 change de Sens (Horaire → Anti-Horaire)
- **Loc Violet** se situe dans la descente entre Niv 0 et Niv –1

Par défaut, les trains circulent dans un **mode aléatoire** pour la recherche du prochain chemin :

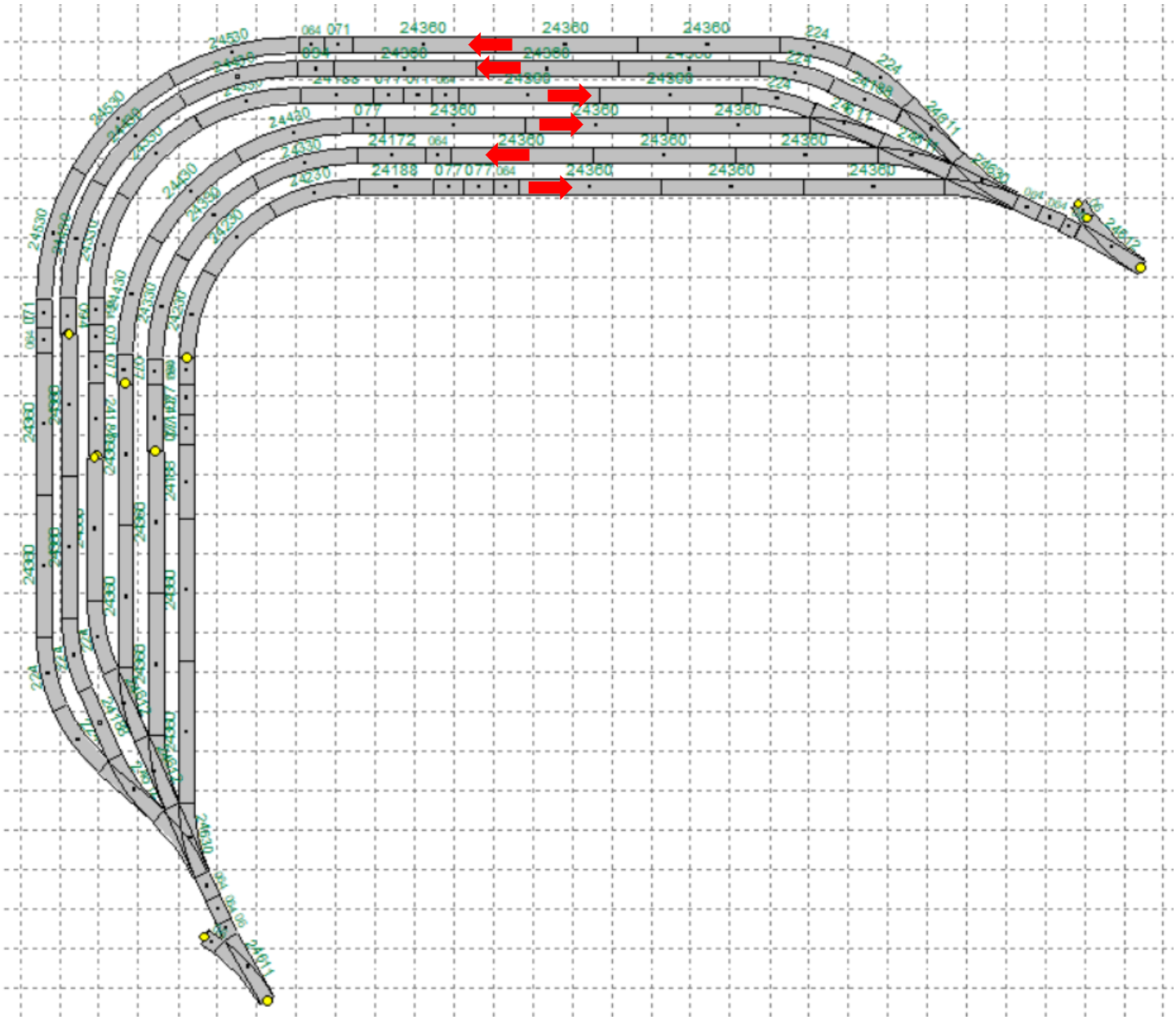
- **Loc Grise**, arrivée au Niv 0, va disposer de 3 blocs libres
- **Loc Vert clair**, arrivée au Niv –2 a 3 choix possibles

Les 3 niveaux inférieurs offrent 20 voies de stationnement (10 par sens)

Dans cette phase transitoire due à l'absence de Niv 0, le logiciel Arduino a dû être modifié (simplifié) en désactivant la détection S88. Les Blocs sont donc rendus artificiellement après un délai.

Sept 2021 Etape4 : Niveau –1

Cette étape ajoute **19 mètres** de rails au Niv –1 pour garer **6 trains** supplémentaires ; soit **20 au total** répartis sur les 3 niveaux inférieurs



Chaque voie est **unidirectionnelle** pour :

- Conserver une symétrie → même nombre de trains circulent dans le **sens Horaire et Anti-horaire**
- Diminuer le risque de collision
- Minimiser le nombre de turns motorisés

De plus une sécurité est implémentée pour ces 20 blocs : tous les trains en mouvement sont arrêtés si les 2 détections d'un bloc sont actives simultanément signifiant qu'un train entre dans un bloc occupé par un autre train.

La longueur maximale des trains est donc limitée par le bloc le plus court :

- **286** cm pour Niv -1
- 297 cm pour Niv -2
- 343 cm pour Niv -3

La maquette comprend 6 modules S88 chaînés (6x16 = 96 détections au maximum) afin de permettre la prochaine construction du Niveau 0

Au niveau des **performances lors d'un arrêt**, **une locomotive reçoit la commande Stop 80 µs** (microseconde) **après la détection S88.** L'écart est de +- 2 µs

Avril 2021 Mesures de performance

Elles vont permettrent de répondre aux questions suivantes (liste non-exhaustive) :

- Quel est l'interval de temps maximum qu'une locomotive doit attendre dans le processus de contrôle ?
- Avec 14 trains en mouvement, quelles parties du code consomment le plus de temps CPU ?
- Qui consomme le plus lors d'une montée en charge ?
- …

Le code Arduino Due actuel comprend une boucle **loop** et un gestionnaire d'interruptions **PWM** :

```
void loop() {
  Learn();       // not used in Automatic mode
  S3_Cmd();      // Debug with Serial3
  S88();         // S88 detection with 4 modules Littfinski (64 inputs)
                 // with Slow&Stop Speed + Free Block
  Core();        // Choix du chemin si loc. activée et à l'arrêt
                 // Commands (Speed, Turn) sended to FIFO (1 FIFO / Loc)
  Dequeue();     // Commands (from FIFO) executed with NON-BLOCKING delay
  CtoPC();       // Arduino to PC (USB) to update GUI
  CfromPC();     // PC to Arduino (USB) to activate Loc
}

void PWM() {...}
```

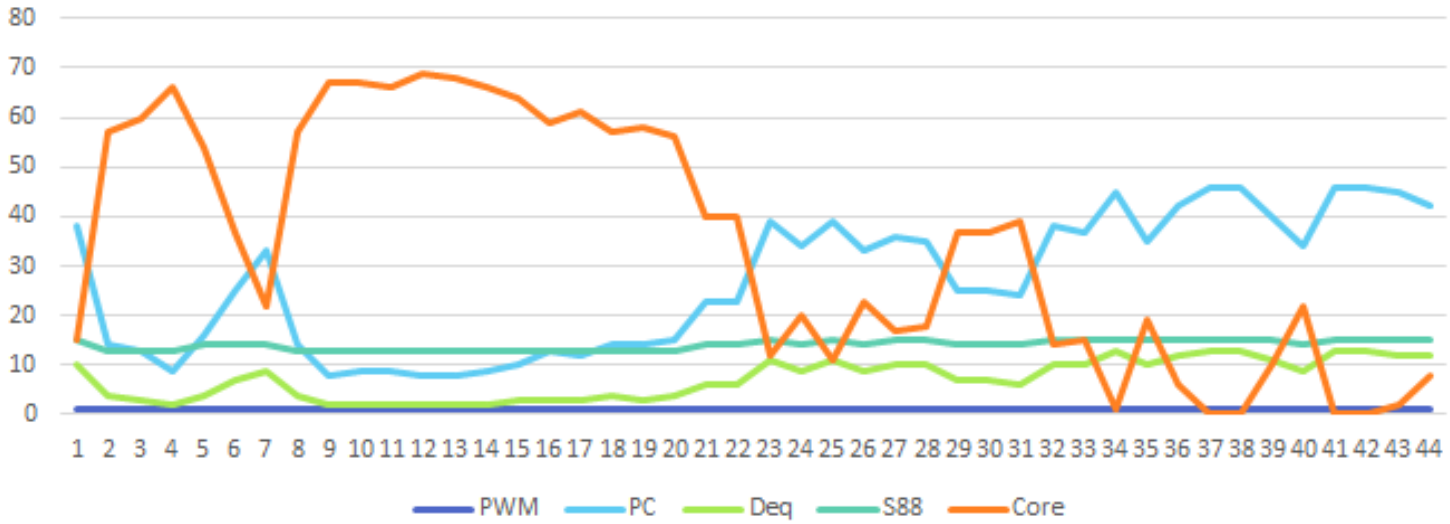
- Aucune attente active (delay) n'est présente
- Toutes les opérations de debug via Serial3 sont désactivées durant les acquisitions

Méthodologie de mesures :

- La fonction `micros()` permet de mesurer facilement un interval de temps
- Elle ne consomme que 15 µs (micro-seconde)
- L'acquisition prend fin si le nombre d'exécution de loop = 100'000 ou si l'interval d'acquisition dépasse 10 secondes.
- Les résultats sont ensuite transférés via Serial3 puis une nouvelle acquisition démarre
- Tous les trains désactivés – **Activer les 14 trains en 2 étapes**

Résultats en % :

- Ces **5 courbes** expriment **le pourcentage du temps total**
- L'acquisition comprend 44 acquisitions (comprises entre 8 et 10 secondes)
- Elle ignore le temps passé à calculer et envoyer ces résultats sur Serial3

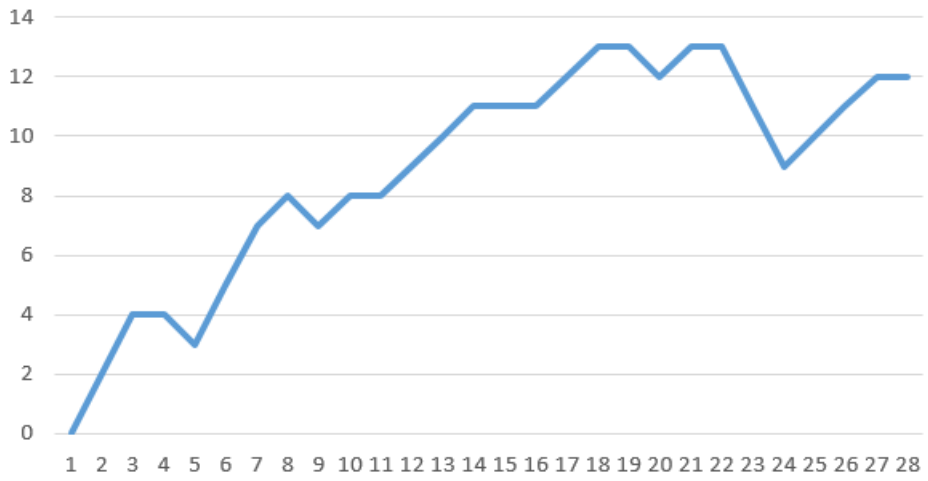


- **PWM** maintient une charge constante (~1%) qui peut être négligée → Merci à la cellue PWM !!!
- **S88** maintient logiquement une charge constante de l'ordre de 14% pour la lecture des 4 modules (64 entrées)
  Salve de 2 ms toutes les 13 ms (période PWM)
  Cette charge peut facilement être divisée par 2 (en passant de 10 à 5 µs pour la durée du bit)
- **Dequeue** indique la charge due à l'exécution des commandes Speed et Turn
- **PC**, qui regroupe les 2 sens de communication USB, montre une charge croissante atteignant 40% pour la mise à jour de l'affichage avec 10 à 14 trains en mouvement
  A optimiser au besoin (le débit actuel étant de 115 kbit/s)

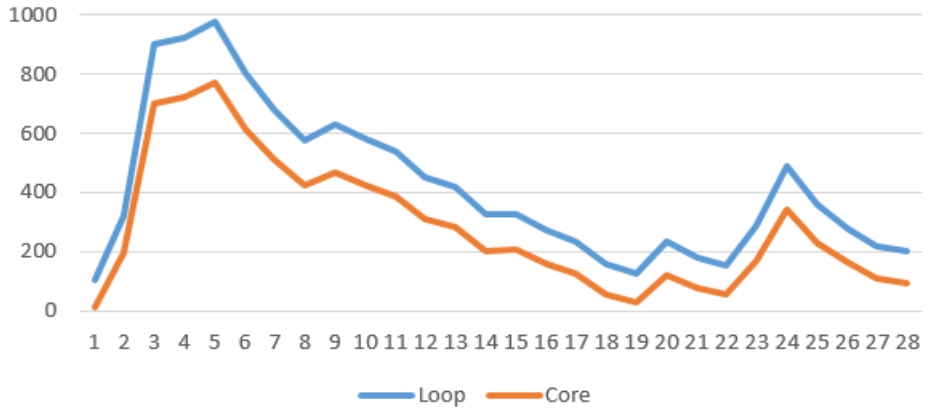
- **Core :**
  dans la première moitié du graphique, beaucoup de trains sont à l'arrêt → **PC** est peu solicité → **Core** est proche de 60% → voir aussi courbe du bas
  dans la seconde moitié, la majorité des trains est en mouvement (confirmée avec **Dequeue**) → l'affichage consomme 40% alors que **Core** est presque au repos

Résultats en valeurs absolues :

- Le scénario de mesure consiste à démarrer 14 trains garés dans les niveaux -3 et -2
- Au début, seuls 4 trains peuvent circuler (monter dans les hélicoïdaux)
- Les autres démarrent lorsque la ressource est libre
- 12 trains sont en mouvement après 17 acquisitions ; soit 1½ min



- La courbe bleu précise l'intervalle de temps qu'une locomotive doit attendre avant d'être servie par Core
- Ce temps (en bleu) dépend principalement du travail effectué par Core (en brun)
- Les temps sont mesurés en **microseconde et restent inférieurs à la milliseconde (ms)**
- La valeur affichée est la moyenne pour une durée d'acquisition égale à 10 secondes.



Conclusion :

- **Arduino Due peut donc contrôler 100 trains sans le moindre problème !!!**
- **… pour autant que le logiciel n'effectue aucune attente active et comprenne des tampons (FIFO)**
- **L'efficacité de la cellue PWM est démontrée avec un charge négligeable de l'ordre du %**
- **Le doublement du nombre de rétrosignalisation S88 se fera sans le moindre problème**
- **La communication PC – Arduino via USB (115 kbit/s) pourrait faire l'objet d'études complémentaires**
- **Il convient de préciser qu'un train H0 (simulant un train réel à 60 km/h) parcourt 15 cm/s**
- **Toutes mes détections S88 ont une longueur minimale de 36 cm !!!**
- **L'avenir semble donc radieux … et je n'aurais jamais assez de trains et de voies pour mettre à genoux mon cher Arduino Due**

Mars 2021 Consommation proche de la limite ?

L'objectif principal de cette étude est d'anticiper d'éventuels problèmes liés à une consommation proche de la limite.

Le booster Littfinski utilisé limite le courant consommé à 4,5 A
2 résistances mises en // produisent **225 mV pour 4,5 A**
Un Arduino Due dédié à cette mesure est utilisé, il n'utilise pas un câble USB conventionnel mais un modèle avec séparation galvanique https://www.wut.de/e-33001-ww-dade-000.php

Pour validation, le résultat affiché sur le **GUI** (produit avec cet Arduino Due) est comparé avec une mesure faite avec un **voltmètre**.

Conditions de mesure	**GUI [mA]**	**Voltmètre [mA]**	**Commentaire**
15 Turn (aiguille)	400	420	1 décodeur Maerklin 74461 consomme 28 mA
Idem + 14 Loc	733	760	1 décodeur de loc consomme 24 mA
Idem + wagon Simplon + Light	**866**	860	**Consommation totale à l'arrêt**

Sachant qu'un Turn consomme ~1,5 A pendant 30 ms → Source = http://gelit.ch/Train/DirectRail.pdf

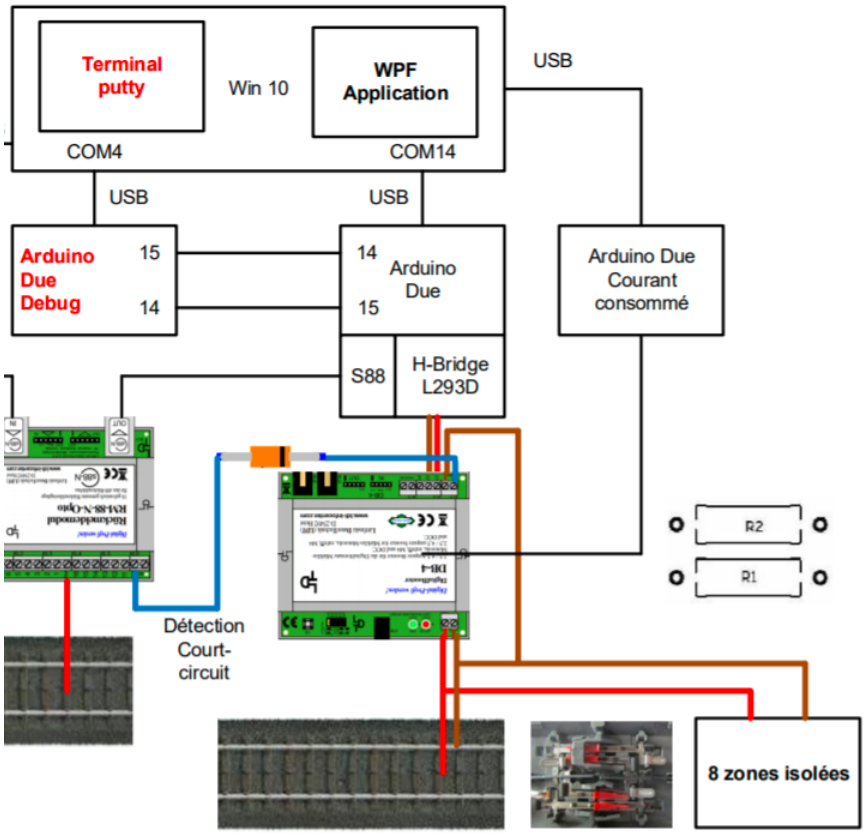
Il ne reste que 4,5 – 0,9 (conso. à l'arrêt) – 1,5 (conso Turn) = **2,1 A pour la circulation des trains**

Conso. actuelle max des 14 trains en mouvement = 1,5 A → ~100 mA / train en mouvement

Conclusion :

1. **L'affichage (GUI) de la consommation est vitale**
2. **Elle ne doit pas dépasser 3 A (afin de garantir le courant nécessaire au Turn)**
3. **Le logiciel de commande n'autorise qu'une seule commutation à la fois suivie d'une pause non-bloquante (paramétrable) de 500 ms**
4. **Il semble réaliste d'avoir simultanément 20 trains en mouvement … l'avenir le dira !**

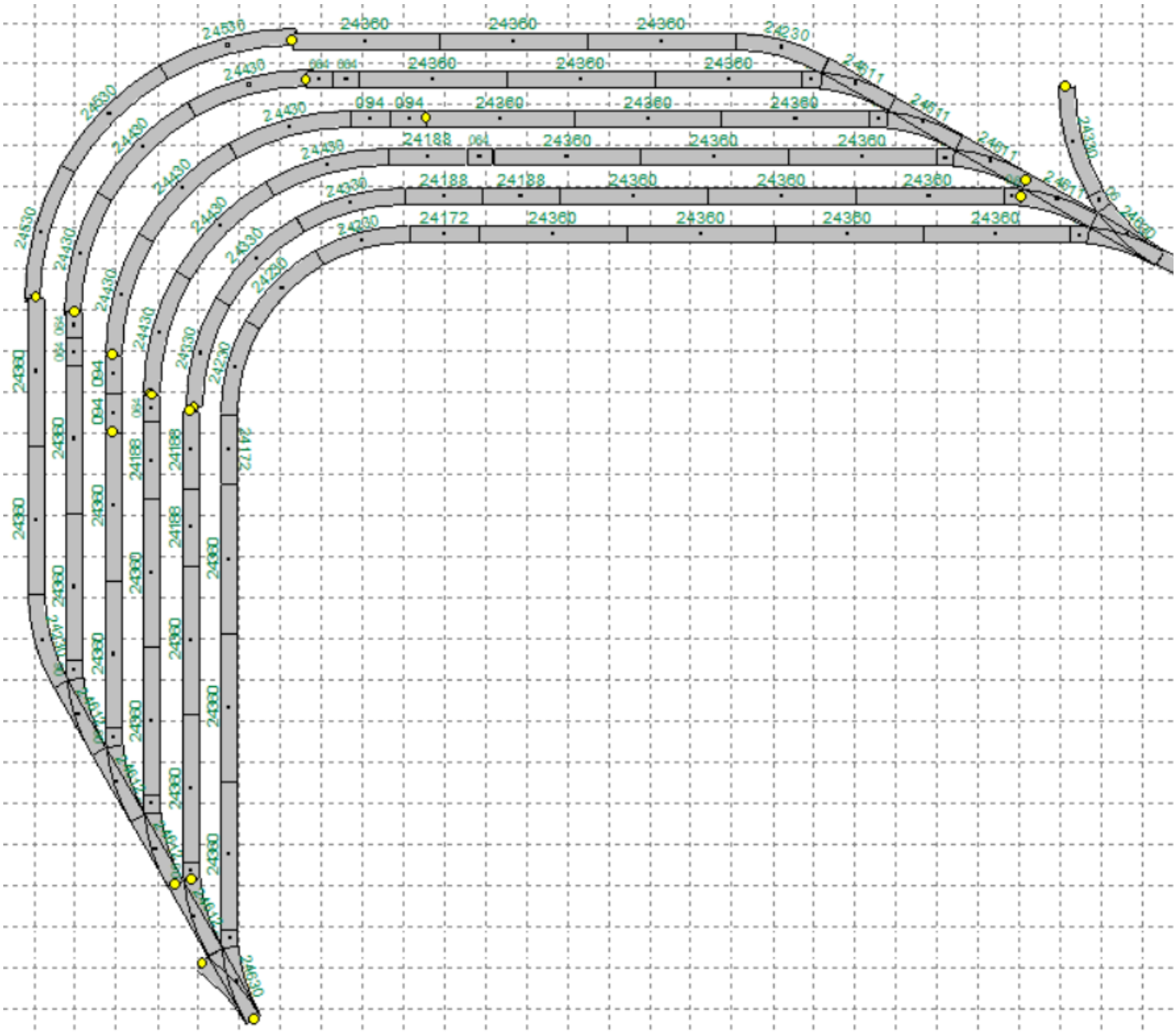
Schéma électrique → Source = http://gelit.ch/Train/DirectRail.pdf



jan 2024
Erreur
Seul le conducteur central du rail doit être isolé afin de maintenir un potentiel de référence commun

Fév 2021 Etape 3 : Niveau –2

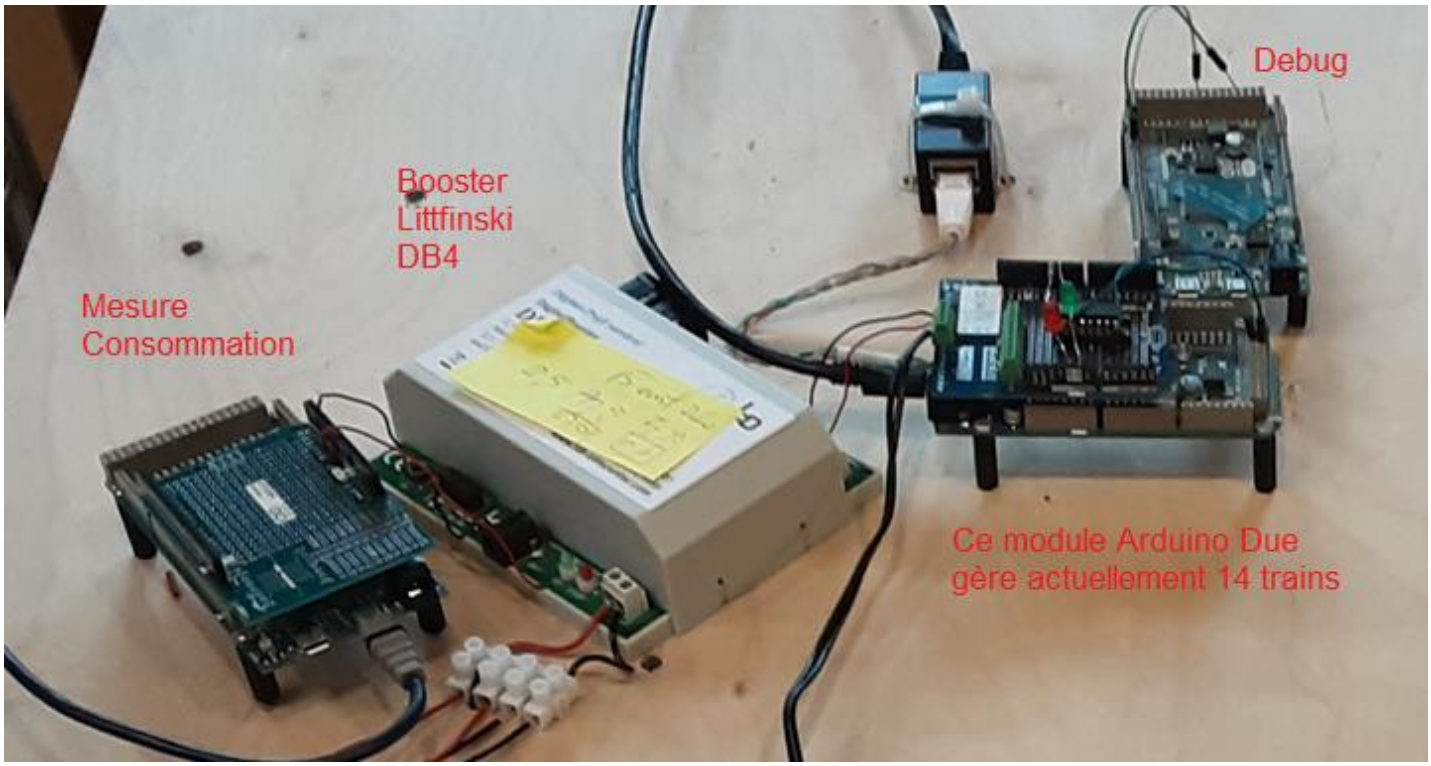
Cette étape ajoute **20 mètres** de rails au Niv –2 pour garer **6 trains** supplémentaires



Un **grand MERCI à Charly** pour m'avoir conseillé d'utiliser une **structure portante Alu Bosch Rexroth !**

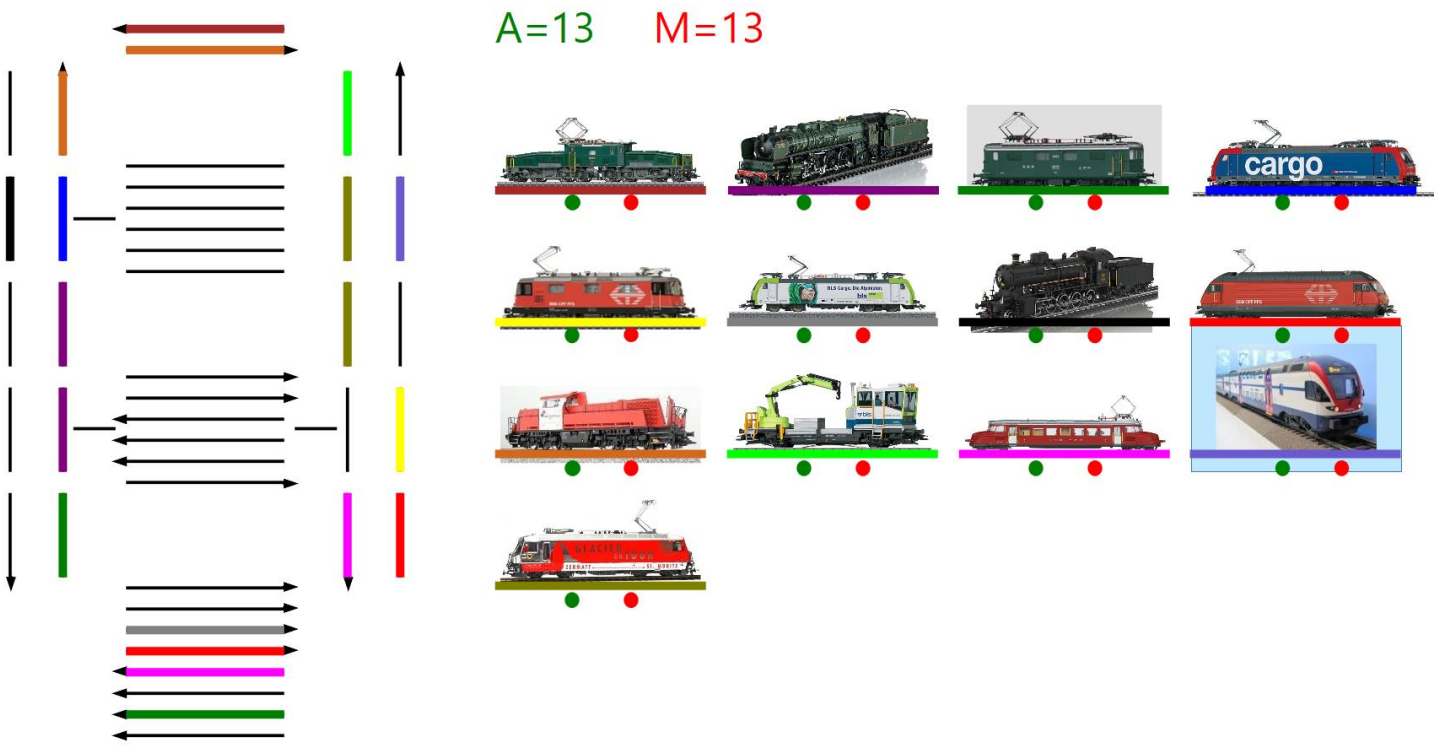
- Elle est facile à construire
- Elle supporte un certain poids
- Elle ne rouille pas
- Elle conserve sa longueur d'origine … mes petits-fils pourront l'utiliser dans 50 ans !
- Elle est facilement extensible
- Elle n'est pas chère

Emplacement des 3 cartes Arduino Due et du booster DB4



Niv –2 terminé

GUI avec 13 trains en mouvement



Prochainement :

- **Boucles de retournement**
- **Performances**
- **Consommation**

Jan 2021 Interface graphique (GUI = Graphical User Interface) pour Etape2 → http://gelit.ch/Train/H4_GUI1.mp4

L'ensemble des rails utilisés (sans prendre en comptes les aiguillages) mesure **108** mètres !

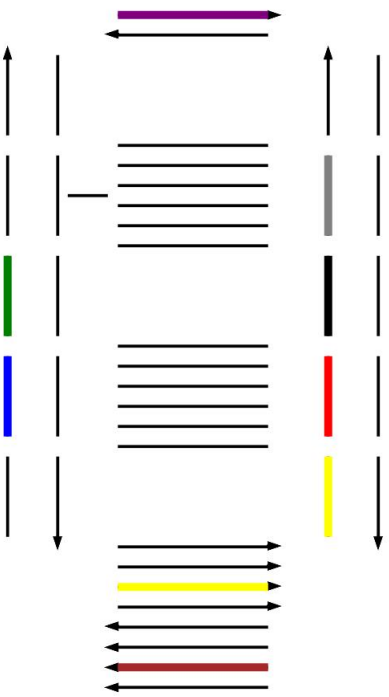
- 5½ tours de cercle extérieur = 324 cm + cercle intérieur = 275 cm = **33** m par hélicoïdal
- Niveau supérieur = 605 cm + 595 cm = **12** m
- Niveau inférieurs = 434 cm + 409 cm + 370 cm + 355 cm + 372 cm + 343 cm + 367 cm + 389 cm = **30** m

Ci-dessous dans la partie gauche, la maquette est représentée par 43 lignes :

- **20 lignes verticales** pour Hélicoïdal situé à gauche et Hélicoïdal situé à droite
- **2 lignes horizontales** pour le niveau supérieur (Niv 0)
- **8 lignes horizontales** pour le niveau inférieur (Niv -3)

Les niveaux intermédiaires (Niv -1 & Niv -2) ne sont pas construits et permettront à terme de garer un total de 20 trains

Chaque ligne constitue un **bloc** (canton) créé avec les **rétrosignalisations** → http://gelit.ch/Train/Raildue_F.pdf



8 locomotives (locs) sont physiquement présentes sur la maquette.
Elles sont identifiées par une **couleur** afin de visualiser la position occupée

Témoin vert : 2 locs sont activées (= autorisées à se déplacer)
Le texte **A=2** indique le nombre de 2 locs actives

Témoin rouge : 2 locs sont en mouvement
Le texte **M=2** indique le nombre de 2 locs en mouvement

La loc identifiée en bleu est activée et bloquée par celle en vert

La loc identifiée en brun a été désactivée ; elle terminera sa course à la fin du bloc

Ce GUI est particulièrement utile en phase de mise au point (debug) ainsi qu'aux personnes qui admirent la maquette.
Il fonctionne sous Windows 10 et comprend 360 lignes de code C# (wpf) + 180 lignes en xaml pour définir lignes, photos,…

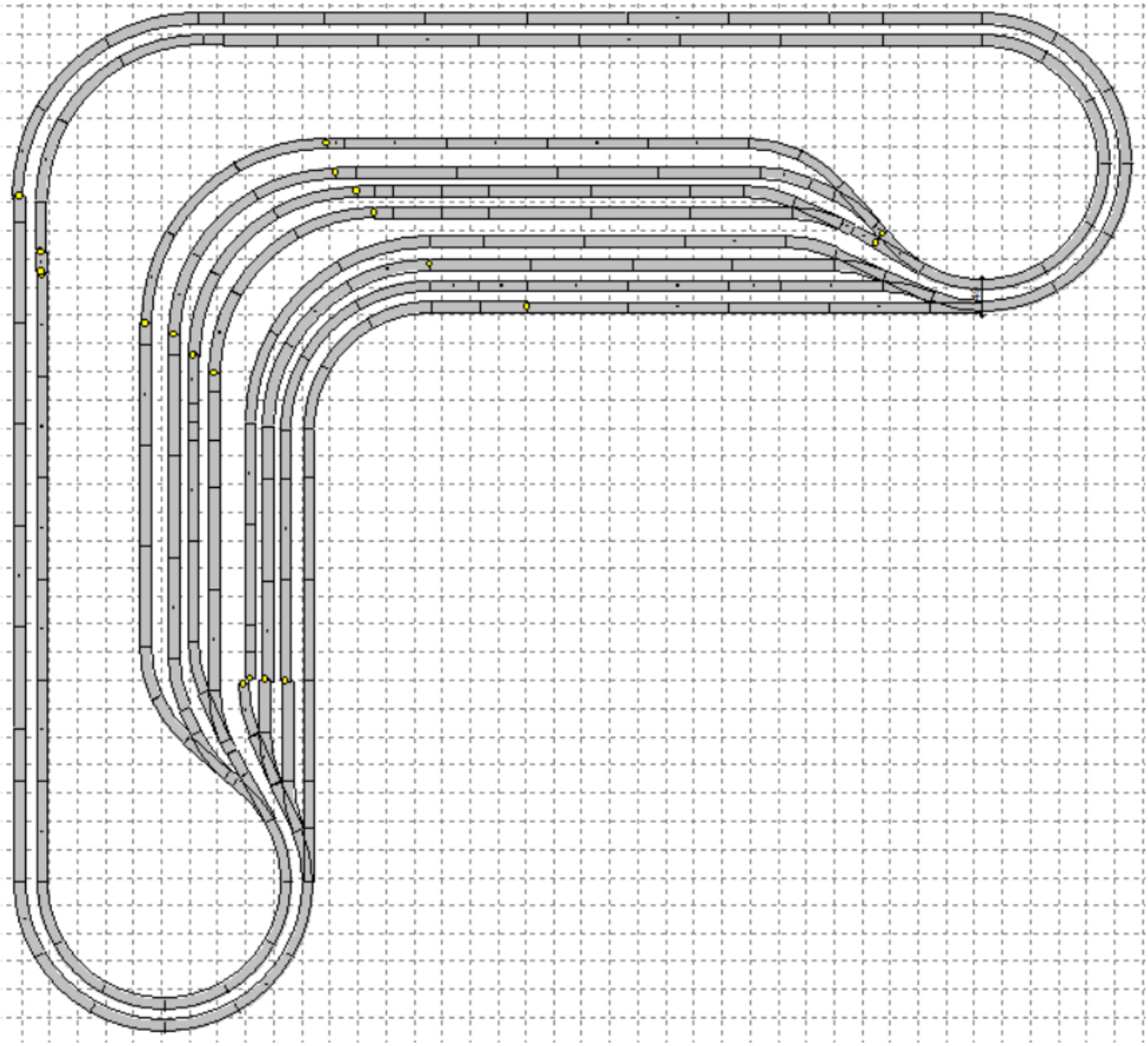
La commande des trains est effectuée par une carte Arduino Due → voir §11 dans http://gelit.ch/Train/DirectRail.pdf
Ce logiciel comprend 1000 lignes de C

Prochainement : quel est le courant consommé (par train, au total, au minimum, au maximum, …) ?
18 déc 2020 : Etape 2 avec vidéo http://gelit.ch/Train/H4_2.mp4

Déc 2020 Etape 2 : Niveau –3

Structures alu

2 hélicoïdaux en position définitive
Niveau –3 terminé



Points de contrôle :

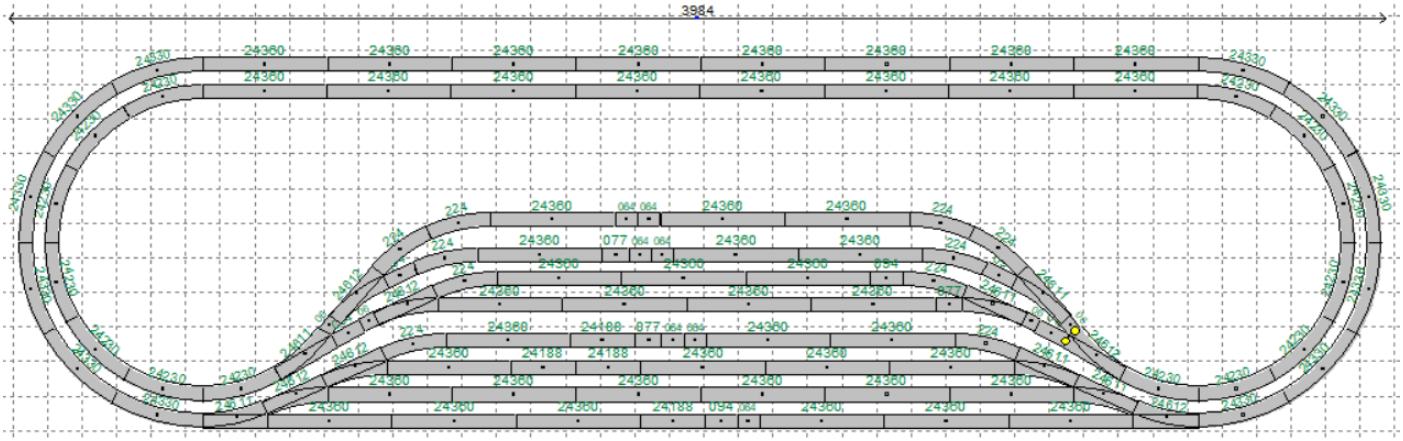
- Comparaison entre plans et réalité

Nov 2020 Etape 1 : 2 hélicoïdaux face à face

2 hélicoïdaux symétriques sont reliés au Niveau 0 et au Niveau -3
10 trains (longueur < 228 cm = 10 x 24230) peuvent circuler simultanément dans chaque hélicoïdal

Points de contrôle :

- Tests des rétrosignalisations – longueur suffisante
- Tests des transitions de pente

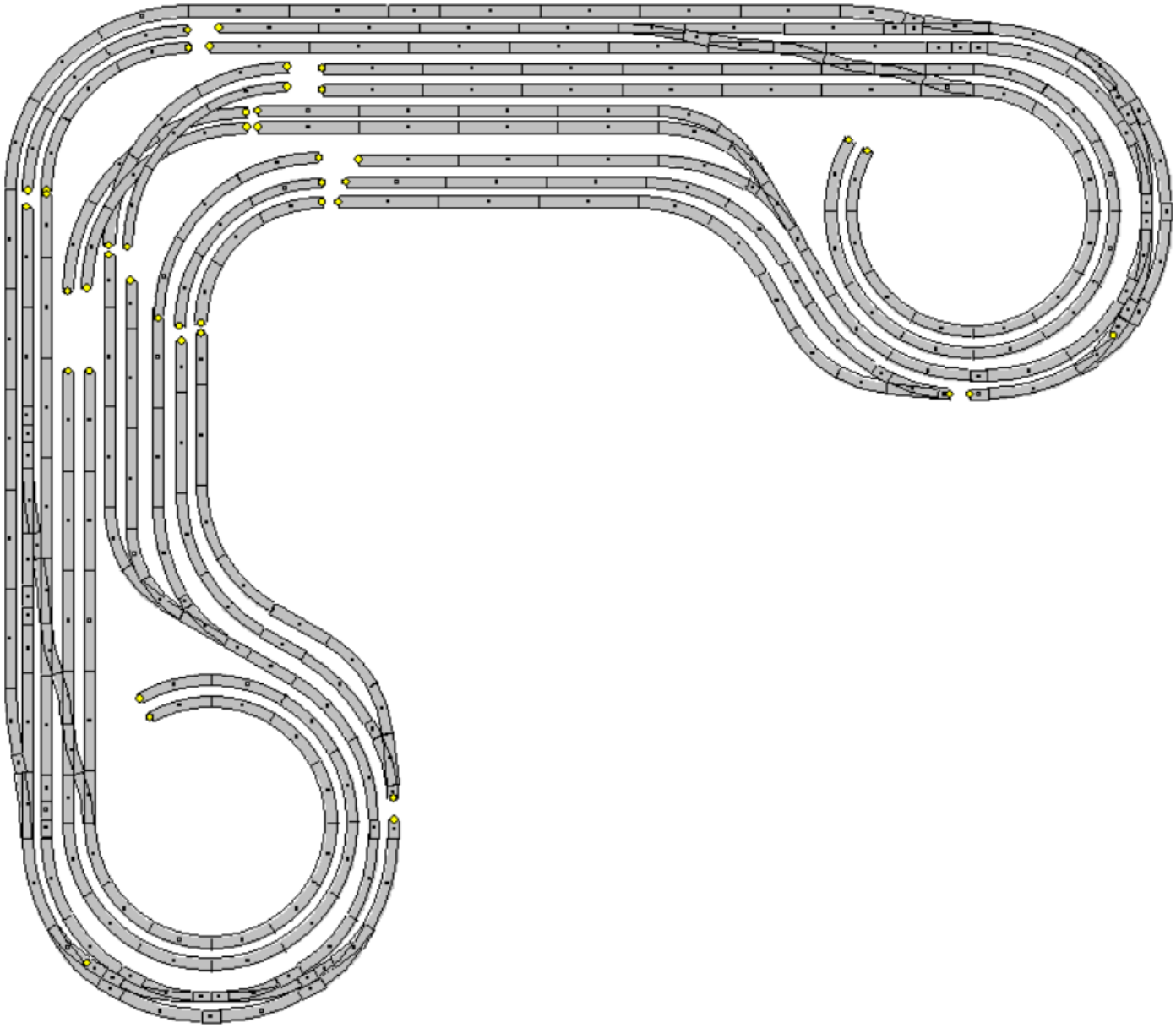


Vidéo http://gelit.ch/Train/H4_1.mp4 montre 10 trains en mouvement ; 5 par sens (Horaire & Anti-horaire)

Oct 2019 Simulateur

Principaux objectifs

1) Modifier hélicoïdal 1 pour lui ajouter **une 2ème voie**
2) Utiliser cette structure en L (440 x 400 x 140 cm) offrant **10 voies** entre les hélicoïdaux



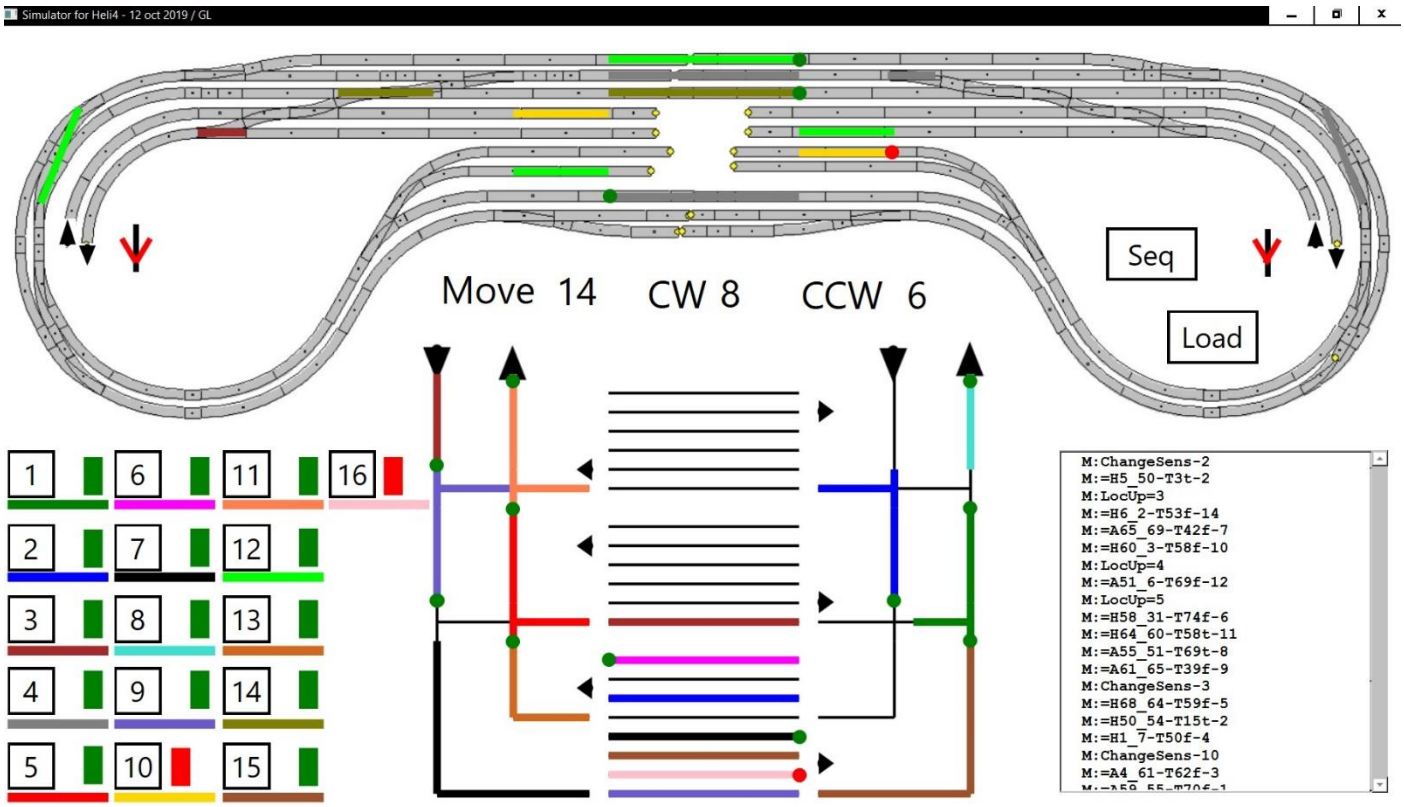
3) Modifier les niveaux inférieurs pour pouvoir garer **20 trains** (6 aux Niveaux -1 & -2 + 8 au Niveau -3)
4) Créer un effet "terrasses décalées" avec les 4 Niveaux

Infrastructure

- 66 cantons dont 42 permettent le stationnement
- 68 aiguilles dont 46 motorisées

Simulateur

- Il doit permettre d'estimer le nombre de trains simultanément en mouvement
- La libération des cantons correpond aux mesures temporelles effectuées sur Hélicïdal 3 ajustées avec les nouvelles longueurs pour une vitesse de 15 cm/s
- La version actuelle simule 16 trains et montre ci-dessous 14 trains en mouvement avec 8 dans le sens horaire et 6 dans le sens anti-horaire



A faire

- Identifier les optimisations possibles
- Sont-elles pertinentes d'un point de vue économique ?