

MANUAL DE CONFIGURACIÓN AVANZADA MEDIANTE SCRIPTS PERSONALIZADOS

Para las señales fijas fundamentales de Train Simulator

Por JALTA (Consigna CR-LN)

Edición 1, Diciembre de 2020.



```
--Estado: Anuncio de Precaución
elseif EstadoProximaSenal == ESTADO_APARADA_BIFURCACION then
    Estado = ESTADO_APRECAUCION

--Estado: Anuncio de Precaución con pantalla alfanumérica.
elseif EstadoProximaSenal == ESTADO_APARADA_BIFURCACION_PANTALLA then
    Estado = ESTADO_APRECAUCION
    PantallaAlfa = true

--Estado: Anuncio de Parada por itinerario que se debe hacer a 30km/h
elseif EstadoProximaSenal == ESTADO_APARADA_BIFURCACION_SINPAJARITA then
    Estado = ESTADO_APARADA

--Estado: Preanuncio o Vía Libre si la señal siguiente se encuentra en Anuncio
elseif EstadoProximaSenal == ESTADO_APRECAUCION then
    if gPant==" " then
        Estado = ESTADO_PREANUNCIO
    else
        Estado = ESTADO_VIALIBRE
    end
end

--Estado: Vía Libre
elseif EstadoProximaSenal == ESTADO_PREANUNCIO then
    Estado = ESTADO_VIALIBRE

elseif EstadoProximaSenal == ESTADO_VIALIBRECONDICIONAL then
    Estado = ESTADO_VIALIBRE
```

Índice

Introducción.....	3
1. Instalación del sistema.	4
2. Edición del script personalizado.....	6
2.1 Función “MainConfigPersonal”.....	6
2.2 Función “MainSecuenciaPersonal”.....	7
2.2 Exportación a Train Simulator.....	8
3. Ejemplos prácticos.....	9
3.1 Sucesivas señales en Anuncio de Parada.....	9
3.2 Gestión avanzada del Anuncio de Precaución.....	11
3.3 Simulación de Averías.....	12
4. Listado de variables globales y etiquetas.....	15
Agradecimientos.....	17

Introducción

Tanto las señales como sus caracteres especiales de configuración resuelven el 95% de situaciones existentes en la RFIG. Sin embargo, hay situaciones que no se pueden resolver con éste método ya que para que fueran resueltas, se tendrían que añadir muchos más caracteres que en la mayoría de veces no llegarían a usarse. La solución a estos casos específicos es usar la configuración avanzada mediante scripts personalizados, de tal forma que un script externo creado por el usuario define parte del comportamiento de la señal, ofreciendo mucha más flexibilidad y personalización que los caracteres especiales.

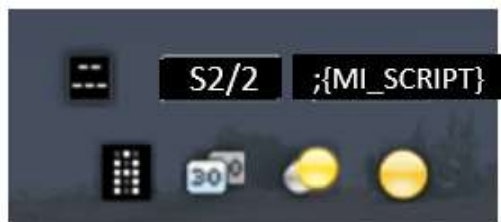


Esta secuencia de señales Rebase Autorizado + Rebase Autorizado no es posible con la configuración de caracteres especiales.

Cuando éste modo es activado, **la configuración por caracteres queda anulada** y unas funciones del script de la señal pasan a ser externas y configurables por el usuario mediante un fichero .lua, de tal forma que queda a disposición del usuario toda la potencia de un script para resolver cualquier situación.

La única dificultad que existe es que el usuario **necesita unos conocimientos muy básicos de programación y de sintaxis del lenguaje Lua**. En internet existe multitud de documentación y tutoriales, pero sobre el lenguaje Lua recomiendo [su web](#).

Para activar dicho modo basta con escribir entre claudators el nombre del script en el segundo cuadro de texto, donde se añaden los caracteres especiales. Por ejemplo si quiero que la señal cargue un script personalizado que se llama MI_SCRIPT.lua deberé escribir en dicho cuadro: “;{MI_SCRIPT}”



Es muy importante **que el nombre esté en mayúsculas**, de no ser así el sistema no va a funcionar y la señal se va a quedar bloqueada. El fichero .lua deberá estar instalado en un directorio específico y deberá contener las funciones externalizadas que se detallaran en éste manual.

Esta funcionalidad solamente se encuentra disponible en las señales fijas fundamentales de 4 y 3 focos (MR LUMINOSA 4 FOCOS, MR LUMINOSA 4 FOCOS LIBERACION, MR LUMINOSA 3 FOCOS, MR LUMINOSA ENTRADA TERMINO, MR MONO 4 FOCOS IZQ /DER, ES_LUM_4F_xV_SM y ES_LUM_3F_xV_SM)

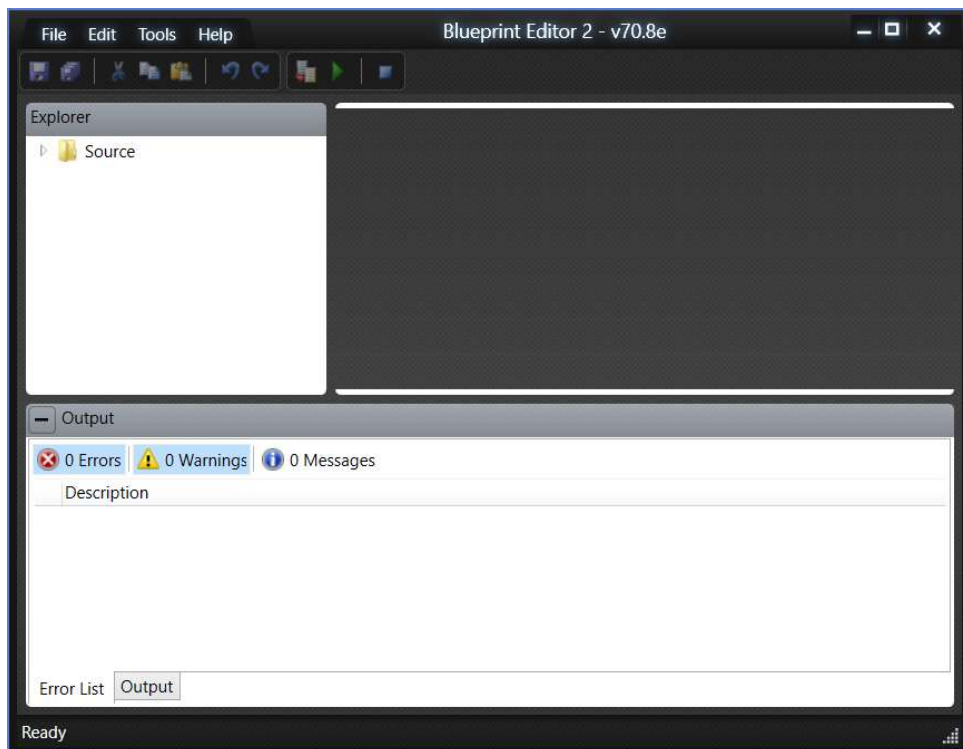
1. Instalación del sistema.

Para la edición y luego exportación del script será necesario el uso del Blueprint Editor del TS y también algún programa de edición de texto Unicode para abrir los ficheros .lua. Si desconocéis lo último yo uso el [Notepad++](#), pero en su defecto y como última opción se puede usar el blog de notas de Windows.

Para abrir el Blueprint Editor hay que ir a la carpeta raíz del simulador (C:\Program Files (x86)\Steam\steamapps\common\RailWorks) y buscar la aplicación **BlueprintEditor2.exe**:

Temp	06/10/2020 18:51	Carpeta de archivos	
TempDump	28/08/2020 23:54	Carpeta de archivos	
action.replay	30/06/2020 16:02	Archivo REPLAY	89 KB
ApplyXS.exe	23/12/2018 13:43	Aplicación	21 KB
BlueprintEditor2.exe	17/09/2020 23:43	Aplicación	797 KB
Clipboard.txt	17/05/2020 20:45	Documento de tex...	4 KB
config.xml	23/12/2018 13:44	Archivo XML	2 KB
config_fullscreen.xml	23/12/2018 13:44	Archivo XML	1 KB
...

Se nos abrirá la ventana del Blueprint Editor y en el explorador tendremos el directorio “Source”, que es donde trabajaremos nosotros:



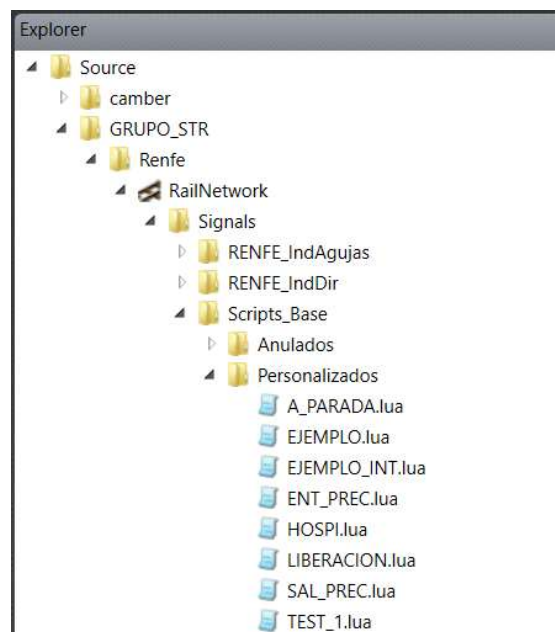
El sistema de producción del simulador funciona de la siguiente forma:

Existen dos carpetas principales, **la primera es la “Assets”**, que es donde se almacena el contenido descargado o proporcionado por los autores. Dentro veremos que hay las carpetas de cada “Provider” y dentro de éstas cada asset descargado. Los archivos que hay ahí son finales, es decir, no es posible editarlos en la mayoría de casos aparte de que se necesitaría el permiso de los desarrolladores para hacerlo.

La segunda carpeta se llama “Source” y es la de los desarrolladores, donde se encuentran los archivos fuente que luego se exportan a archivos finales en la carpeta “Assets” mediante el Blueprint Editor. En ésta carpeta es donde vamos a trabajar utilizando el Blueprint Editor para compilar y exportar los ficheros .lua de nuestros scripts.

Para empezar, con el fichero .zip de descarga nos ha venido un conjunto de carpetas agrupadas dentro de “Source”. Estas carpetas se encuentran vacías (excepto la que contiene los scripts de ejemplo), pero es imprescindible mantener la jerarquía si no se exportaran en otro sitio de la carpeta “Assets”. Debemos copiar y pegar estas carpetas a la carpeta “Source” del simulador ubicada normalmente en C:\Program Files (x86)\Steam\steamapps\common\RailWorks.

A continuación abriremos el Blueprint Editor y desplegaremos la jerarquía de carpetas hasta llegar a “Personalizados”:



La carpeta “Personalizados” es donde vamos a colocar los scripts personalizados que luego llamaremos desde el cuadro de texto de la señal. A pesar de que en la imagen se nos muestran más, de momento solo hay dos: **“EJEMPLO.lua”** y **“EJEMPLO_INT.lua”**

- EJEMPLO.lua: script para señales de mas de un link que habitualmente hacen de señales de entrada y salida en las estaciones.
- EJEMPLO_INT.lua: script para señales de un solo link que habitualmente hacen de señales intermedias.

Haciendo doble clic en ellos se deberían abrir con el editor Unicode que dispongamos. Si no fuera así, haciendo clic derecho en el nombre del directorio podemos abrir la carpeta en el explorador mediante la opción “Show In Explorer” y manipular directamente el archivo desde ahí.

2. Edición del script personalizado.

Abriendo alguno de los dos scripts encontraremos que el fichero se estructura en dos funciones:

```
MainConfigPersonal()
```

y

```
MainSecuenciaPersonal(linkRestrictivo,EstadoProximaSenal)
```

La primera función se dedica a configurar las características de la señal (vendría a ser como la configuración con los caracteres especiales) y la segunda se dedica a configurar el estado y aspecto de la señal en función de la ocupación de sus links, el estado de las agujas situado a continuación y el estado de la señal siguiente.

Como ya he explicado, al ser un Lua script el usuario tiene acceso a todas las funciones de las señales disponibles por lo que el usuario tiene total libertad para hacer y deshacer a partir de esas dos funciones. A parte de eso, el simulador dispone de funciones propias relacionadas la señalización, que podéis consultar en el [Signaling Function Reference](#) del *Train Simulator Development Documentation*.

2.1 Función “MainConfigPersonal”.

Esta función sólo se ejecuta al cargarse el escenario o cuando se reinicia este. No necesita nada ni devuelve nada, pues trabaja directamente con variables globales accesibles desde todas las funciones. **Las variables globales definidas por el usuario se deberán inicializar aquí.** En ella definiremos las características de la señal asignando dichos estados a las variables de configuración:

Variable	Configura	Tipo
gPant	Activa la pantalla de Preauncio al asignarle la velocidad “xx”. Por defecto se debe dejar sin texto “ ”.	String
gAPRX	Indica si se debe mostrar Anuncio de Precaución aunque se vaya a una vía desviada. Nota: hay que editar la función Secuencia Personal para que así sea. Ver Ejemplo 3.2.	Boolean
gParadoxDefecto	Estado de la señal cuando se encuentra en reposo. Normalmente cuando no se aproxima ningún tren las señales suelen ordenar parada. Poniendo “false” anula ésta característica y la señal se mostrará siempre atendiendo la sucesión de señales.	Boolean
gCambioSentido	De acuerdo con el manual, en la configuración normal solo es posible abrir las señales con la tecla TAB después de un cambio de sentido. Esta variable permite que la señal se abra automáticamente después de que la circulación la haya rebasado en sentido contrario. Equivale al carácter de configuración X	Boolean
gPerm	Haz que aparezca el cartelón P. Sólo para señales intermedias. Equivale al carácter de configuración P	Boolean
gBAD	Haz que la señal esté siempre atendiendo la sucesión de señales. Está hecha para las señales intermedias del bloqueo tipo BAD que al ser unidireccional siempre se encuentran atendiendo la sucesión de señales. Equivale al carácter de configuración D	Boolean


2.2 Función “MainSecuenciaPersonal”

Esta función se ejecuta de manera puntual **cuando una actualización del estado de la señal es requerida** ya sea porque la señal siguiente ha cambiado (y le envía un mensaje para indicarle que se actualice porque ha cambiado), las agujas han cambiado (y le envía un mensaje al sistema porque el link conectado ha cambiado), después de inicializarse la señal, etc...en definitiva, cada vez que se produce un cambio que pueda ser susceptible a afectar el estado de la señal.

El prototipo de la función es el siguiente:

```
Estado, Aspecto MainSecuenciaPersonal( linkRestrictivo, EstadoProximaSeñal )
```

Donde:

- Estado: devuelve el estado de la señal.
- Aspecto: devuelve el aspecto de la señal.
- linkRestrictivo: Se trata del link que ha marcado el usuario como restrictivo ()
- EstadoProximaSeñal: Se trata del estado de la Próxima señal.

El estado de la señal se refiere **a como se encuentra**: En Parada, en Rebase, en Anuncio de Parada, etc...Para describirlos se utilizan etiquetas que empiezan por ESTADO_, por ejemplo ESTADO_PARADA.

El aspecto de la señal **es la representación del estado** mediante los focos luminosos y las pantallas alfanuméricas. Al igual que los estados, los aspectos disponen de etiquetas que empiezan por ANIMSTATE_, por ejemplo ANIMSTATE_PARADA.

Al final de éste manual y dentro del script de ejemplo podéis consultar todos los estados y aspectos disponibles así como las variables globales disponibles.

Dicho esto, la función se divide en dos partes: La primera parte sirve para determinar el estado, y la segunda para determinar el aspecto en función del estado.

La determinación del estado se hace mediante la discriminación de casos, de más a menos restrictivos, de tal forma cuando un caso no se cumple, inmediatamente se estudia el siguiente. Para determinar el estado de la señal existen 4 casos:

- Caso 1: condiciones que hacen que la señal se encuentre en **Parada o Rebase Autorizado**. En ésta función no se encuentran todas las condiciones, **ya que por defecto la señal estará en Parada siempre y cuando no haya link conectado (gConnectedLink = -1) o el link conectado conduzca a un trayecto marcado como a Contravía en BAD. En el caso de las intermedias, si hay una ocupación en el cantón**. El resto de condiciones son editables por el usuario en ésta parte de la función mediante el uso de variables globales.
- Caso 2: Estados dependientes de las agujas situadas a continuación de la señal. Si el link conectado es más grande que uno significa que nos vamos a una vía que no es directa (como norma general el link uno es el que se la asigna a una vía directa). Para que la señal anterior nos muestre un Anuncio de Precaución debemos usar un estado específico para ésta situación, el ESTADO_APARADA_BIFURCACION. Existen dos estados más en función de si el usuario ha configurado velocidad y dirección para cada link para que se muestre en la pantalla alfanumérica del Anuncio de Precaución, **pero estos se determinan de manera automática en la función DeterminaEstado**. Obviamente ésta parte sólo es de aplicación si la señal tiene más de un link, de otra forma no tiene utilidad.
- Caso 3: Estados dependientes del estado de la señal siguiente. Aquí asignaremos el estado que debe mostrar la señal en función del estado leído de la señal siguiente. Por ejemplo si la señal siguiente esta en ESTADO_PARADA, la nuestra deberá estar en ESTADO_APARADA, o si la señal siguiente se encuentra en ESTADO_APARADA_BIFURCACION, la nuestra deberá

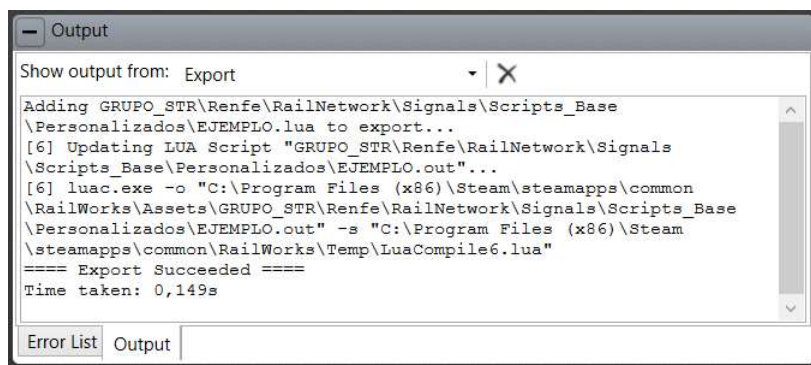
estar en ESTADO_APRECAUCION. Aquí podemos determinar si la señal puede dar vía libre condicional, o preanuncio u otras combinaciones en función del estado siguiente.

- Caso 4: Ninguno de los casos anteriores. Éste caso no es un caso normal, pues no responde a ninguna de las situaciones que se han previsto, como por ejemplo, cuando se lee el estado de la señal siguiente, y éste no se ha tenido en cuenta en el Caso 3. El estado resultante es el que responda a la mayor seguridad posible. El Anuncio de Parada quizás es el mejor cuando desconocemos el estado de la señal siguiente, o directamente Parada si consideramos que el origen de la incidencia esta en la propia señal.

La segunda parte de la función trata de asignar un aspecto a cada estado. En general estado y aspecto suelen coincidir, por ejemplo ESTADO_VIALIBRE suele coincidir con ANIMSTATE_VIALIBRE, pero en otras ocasiones no. Por ejemplo el foco blanco de rebase autorizado puede ser fijo o destellante, para eso seleccionaremos ANIMSTATE_REBASEAUTO o ANIMSTATE_REBASEAUTO_INT. Igual para una vía desviada, normalmente el aspecto asignado suele ser el de Anuncio de Parada, pero también es posible que sea Vía Libre en caso de ir hacia un BLA o BT.

2.2 Exportación a Train Simulator

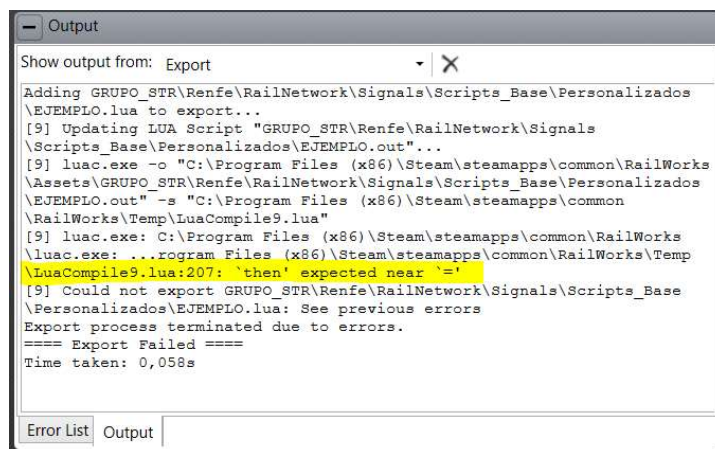
Una vez tengamos ya definido nuestro script personalizado es hora de exportarlo al Train Simulator. Para ello lo vamos a guardar y en el Blueprint Editor haremos clic derecho sobre él y se nos desplegará un menú. Iremos a **Export -> Export with references, o bien pulsaremos F7**. Si todo ha ido bien empezará la exportación y al terminar se nos mostrará en el terminal inferior el texto “export succeeded”:



```

Output
Show output from: Export
Adding GRUPO_STR\Renfe\RailNetwork\Signals\Scripts_Base
\Personalizados\EJEMPLO.lua to export...
[6] Updating LUA Script "GRUPO_STR\Renfe\RailNetwork\Signals
\Scripts_Base\Personalizados\EJEMPLO.out"...
[6] luac.exe -o "C:\Program Files (x86)\Steam\steamapps\common
\RailWorks\Assets\GRUPO_STR\Renfe\RailNetwork\Signals\Scripts_Base
\Personalizados\EJEMPLO.out" -s "C:\Program Files (x86)\Steam
\steamapps\common\RailWorks\Temp\LuaCompile6.lua"
==== Export Succeeded ====
Time taken: 0,149s
  
```

Por el contrario, si ha habido algún error abortará la exportación y el compilador nos indicará donde lo ha encontrado:



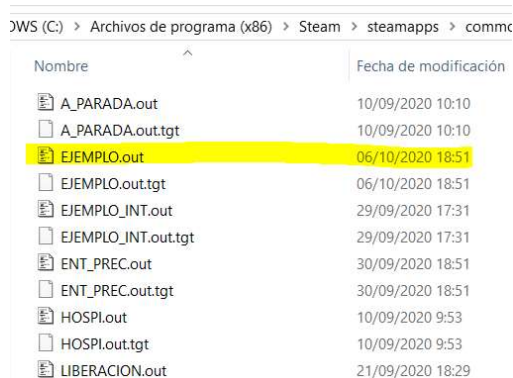
```

Output
Show output from: Export
Adding GRUPO_STR\Renfe\RailNetwork\Signals\Scripts_Base\Personalizados
\EJEMPLO.lua to export...
[9] Updating LUA Script "GRUPO_STR\Renfe\RailNetwork\Signals
\Scripts_Base\Personalizados\EJEMPLO.out"...
[9] luac.exe -o "C:\Program Files (x86)\Steam\steamapps\common\RailWorks
\Assets\GRUPO_STR\Renfe\RailNetwork\Signals\Scripts_Base\Personalizados
\EJEMPLO.out" -s "C:\Program Files (x86)\Steam\steamapps\common
\RailWorks\Temp\LuaCompile9.lua"
[9] luac.exe: C:\Program Files (x86)\Steam\steamapps\common\RailWorks
\luac.exe: ...rogram Files (x86)\Steam\steamapps\common\RailWorks\Temp
\LuaCompile9.lua:207: 'then' expected near '='
[9] Could not export GRUPO_STR\Renfe\RailNetwork\Signals\Scripts_Base
\Personalizados\EJEMPLO.lua: See previous errors
Export process terminated due to errors.
==== Export Failed ====
Time taken: 0,058s
  
```

En éste caso ha encontrado un fallo en la línea 207 y nos dice aproximadamente de qué se trata.

Para comprobar el resultado iremos a la carpeta “Assets” con el explorador de Windows y buscaremos la carpeta del Grupo STR

(C:\Program Files (x86)\Steam\steamapps\common\RailWorks\Assets\GRUPO_STR\Renfe\RailNetwork\Signals\Scripts_Base\Personalizados) Y comprobaremos que efectivamente se ha creado la carpeta “Personalizados” y dentro está el archivo “EJEMPLO.out”:



Nombre	Fecha de modificación
A_PARADA.out	10/09/2020 10:10
A_PARADA.out.tgt	10/09/2020 10:10
EJEMPLO.out	06/10/2020 18:51
EJEMPLO.out.tgt	06/10/2020 18:51
EJEMPLO_INT.out	29/09/2020 17:31
EJEMPLO_INT.out.tgt	29/09/2020 17:31
ENT_PREC.out	30/09/2020 18:51
ENT_PREC.out.tgt	30/09/2020 18:51
HOSPI.out	10/09/2020 9:53
HOSPI.out.tgt	10/09/2020 9:53
LIBERACION.out	21/09/2020 18:29

Bastará en la señal deseada asignarle el script tal y como se cuenta en la Introducción. En éste caso sería el de ;{EJEMPLO}.

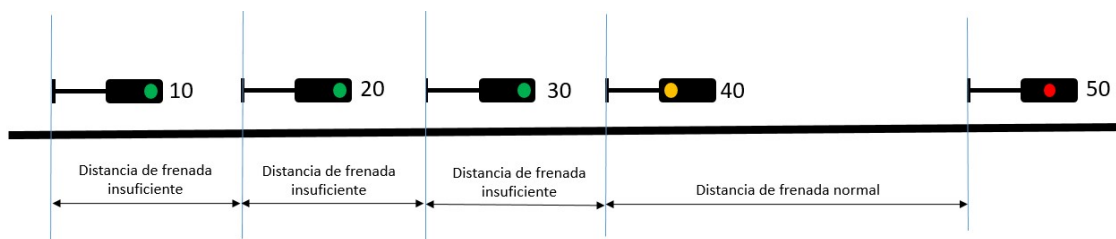
Recordad que para que se hagan efectivos los cambios siempre hay que reiniciar el escenario, y que los aspectos que aparecen en el editor de rutas no se corresponden con los finales en el escenario.

3. Ejemplos prácticos

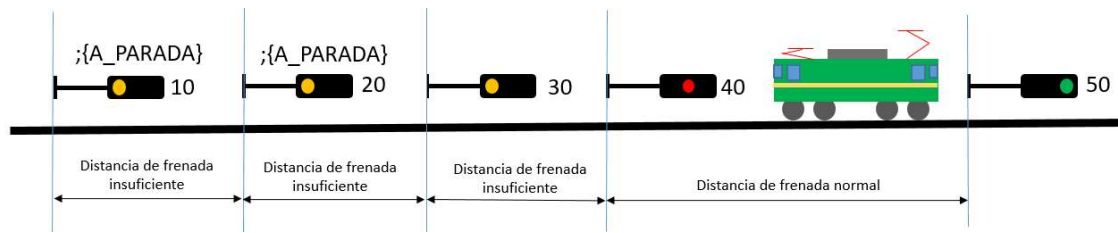
A continuación vamos a describir un conjunto de ejemplos prácticos donde recrearemos situaciones que resultan imposibles de implementar mediante la configuración de caracteres especiales pero resultan bastante sencillas mediante la configuración avanzada con scripts.

3.1 Sucesivas señales en Anuncio de Parada

Las secuencias de señales se pueden alterar siempre y cuando sigan la sucesión, como es en éste ejemplo. En la siguiente figura se representa un tramo de vía de alta densidad de tráfico con diversas señales intermedias. La capacidad del tramo viene determinado por el número de cantones, pero cuanto más cantones hay, menor es su longitud y esto ocasiona que las distancias de frenado sean insuficientes para la velocidad máxima admisible:



Resumiéndolo, la normativa impone una longitud mínima para frenar con seguridad para una determinada velocidad máxima. Existen varias soluciones para mantener la velocidad máxima del trayecto, des de dotar de Preanuncios a las señales (la más lógica) como replicar el Anuncio de Parada a la señal anterior (la opción de salir del paso). En éste ejemplo se presenta ésta última solución, ya que durante mucho tiempo estuvo vigente en los túneles de Barcelona hasta que hicieron un recantonamiento para la instalación del ERTMS N2.



En la figura anterior, la señal 40 se encuentra en Parada porque se encuentra un tren en su cantón. La señal 30 entonces se encuentra en Anuncio de Parada y la señal 20 ya debería encontrarse en vía libre, pero al haber una distancia de frenada insuficiente, el Anuncio de Parada de la señal 30 se replica en la señal 20 y 10. Todas las señales que tengan distancia insuficiente de frenada permanecerán en Anuncio de Parada hasta que se libere el cantón con distancia de frenada normal.

Como ya he explicado, esta situación se solía dar en el túnel de Plaça Catalunya en Barcelona, donde por un error de instalación, los cantones no cumplían con la distancia suficiente y las señales se encontraban siempre en Anuncio de Parada, ya que la circulación de trenes en aquél tramo es muy elevada. Solamente en festivos a primera o última hora del día era posible encontrarse todos los cantones en Vía Libre, pues no tenían ningún tren delante durante todo el tramo. Por suerte ésta situación se ha solucionado con el último recantonamiento que se hizo para preparar la instalación del ERTMS N2.

Para implementar esta situación en nuestras señales partiremos del script de ejemplo EJEMPLO.lua que viene con el pack, así que haremos una copia de él que la renombraremos por ejemplo como A_PARADA.lua. Configuraremos la señal como toque (si es permisiva, etc..) y pasaremos a editar la función MainSecuenciaPersonal.

Debemos hacer que cuando el estado leído de la señal siguiente sea “ESTADO_APARADA”, nuestro estado también sea “ESTADO_APARADA”:

```
elseif EstadoProximaSenal == ESTADO_APARADA then
    --[[Comentar éstas líneas:
    if gPant~="" then
        Estado = ESTADO_PREANUNCIO
    else
        Estado = ESTADO_VIALIBRE
    end]]

    --Y sustituir por:
    Estado = ESTADO_APARADA

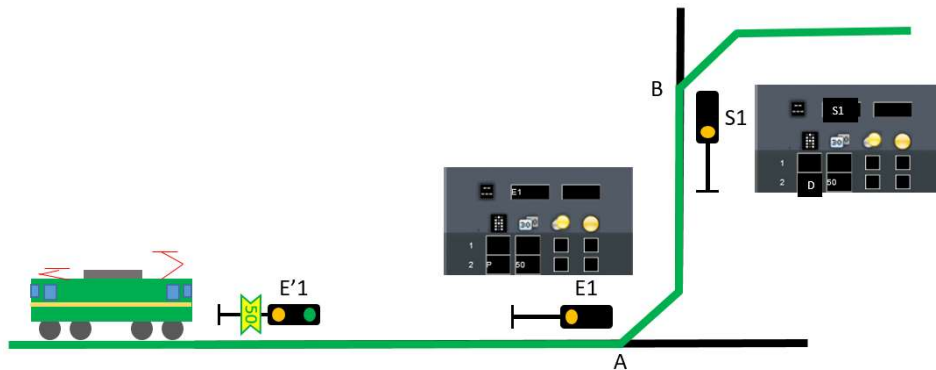
elseif EstadoProximaSenal == ESTADO_APARADA_BIFURCACION then
    Estado = ESTADO_APRECAUCION
```

De ésta forma se conseguirá que cuando la señal siguiente esté en Anuncio de Parada, ésta señal también lo esté.

De igual forma si la señal tuviera Preanuncio, podríamos hacer sucesivos señales en preanuncio, como también sucede en las proximidades de Barcelona, debido al alto volumen de circulaciones, o incluso Vía Libre Condicional cuando la señal siguiente también lo está, para reducir de 200km/h a 160km/h..

3.2 Gestión avanzada del Anuncio de Precaución

En éste ejemplo trataremos una situación que no es posible de reproducir con la configuración normal. La siguiente imagen muestra el esquema de una doble bifurcación:



La velocidad de paso por las agujas A y B es de 50 km/h, y así se ha configurado en las señales E1 y S1. En el caso de la E1 se ha configurado con pajarita, y en la S1 con Pantalla Alfanumérica. Como la E1 es la señal anterior a la S1, ésta debería mostrar también el Anuncio de Precaución con la pantalla alfanumérica, **sin embargo esto no está sucediendo ya que en la E1 se esta dando el Caso 2** (estados dependientes de las agujas situadas a continuación) y se muestra en Anuncio de Parada.

El resultado de ésta secuencia es que el paso por la aguja A es a 50 km/h y el paso por la aguja B es a 30km/h, ya que según el Artículo 2.1.2.5 del RCF, *“En el caso de dos señales sucesivas en anuncio de parada, con agujas a continuación de la segunda señal (no contabilizándose a estos efectos las señales de retroceso), no se excederá la velocidad de 30 km/h al paso por dichas agujas, salvo que haya una señal indicadora de posición de agujas que indique una velocidad superior”*

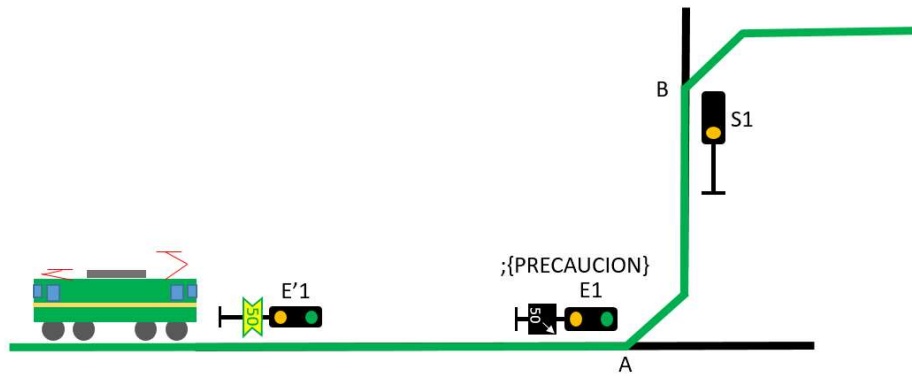
Así que para poder pasar a 50km/h la aguja B podemos hacer dos cosas, o colocar una indicadora de posición de agujas delante de B que indique la velocidad a 50km/h, o bien podemos hacer mediante script que aparezca el Anuncio de Precaución en la señal E1 con su pantalla Alfanumérica. Vamos a tratar aquí ésta segunda opción.

Como siempre partiremos del script de Ejemplo (EJEMPLO.lua) y haremos una copia que la renombraremos como PRECAUCION.lua. A continuación iremos a la segunda parte de la función MainSecuenciaPersonal, donde se relaciona el aspecto con el estado, y cuando el estado sea ESTADO_BIFURACION comprobaremos si el estado siguiente es ESTADO_APARADA_BIFURACION_PANTALLA. Si lo es, entonces mostraremos el Anuncio de Precaución (ANIMSTATE_APRECAUCION_PANTALLA). Si es cualquier otro estado, entonces el aspecto será el habitual ANIMSTATE_APARADA:

```
elseif Estado == ESTADO_APARADA then
    Aspecto = ANIMSTATE_APARADA
    --Aspecto = ANIMSTATE_APARADAINM

elseif Estado == ESTADO_APARADA_BIFURACION then
    --Miramos el estado de la proxima señal.
    --Si es vía desviada con pantalla alfanumérica entonces mostramos el Anuncio de Precaución.
    if EstadoProximaSeñal == ESTADO_APARADA_BIFURACION_PANTALLA then
        Aspecto = ANIMSTATE_APRECAUCION_PANTALLA
        --De otra forma, muestra Anuncio de Parada.
    else
        Aspecto = ANIMSTATE_APARADA
    end
end
```

El resultado es el siguiente:



Nos queda para solucionar un asunto, y es que si editamos sólo ésta parte del código, nos aparecerán los números flotando en el aire, ya que la pantalla no se va a mostrar. Esto es debido a que cuando se carga el escenario y se cargan las señales, hay un intercambio de mensajes y uno de ellos es que se le informa a la señal anterior si es avanzada y si es necesario que aparezca pantalla o pajarita (por ejemplo la E1 manda un mensaje a la E'1 para que le aparezca la pajarita a 50). Este mensaje solo llega por el link 1, ya que por los otros links no sirve ya que predomina el estado de bifurcación, entonces el Anuncio de Precaución no se mostraría nunca y la pantalla siempre estaría apagada. Sin embargo existe una variable que “fuerza” que aparezca una pantalla o pajarita aunque el mensaje llegue por un link diferente del 1. Para ello tenemos que ir a la función `MainConfigPersonal` y poner a “true” la variable `gAPRX`:

```
--¿Debe mostrar Anuncio de Precaución aunque se vaya a una vía desviada?
gAPRX = true
```

3.3 Simulación de Averías.

Una de las posibilidades que ofrece el script personalizado es la simulación de averías de manera aleatoria e impredecible. La avería de una señal (por ejemplo, que no se abra) puede dar bastante jugabilidad sobretodo en señales intermedias con letra “P” ya que no se requiere ninguna autorización para rebasarlas.

En éste ejemplo nos centraremos en dos casos: simular la no apertura de la señal y simular un foco fundido que implique la aparición de un estado más restrictivo.

Como siempre partiremos del script de ejemplo `EJEMPLO.lua` lo copiaremos y lo renombraremos como `AVERIA.lua`.

Para que sea más real añadiremos el factor de la probabilidad de fallo, de tal modo que la señal funcionará normalmente pero a veces se producirá el fallo que le hagamos programado. Para ello utilizaremos la función `math.random` del lenguaje lua:

```
local Averia = 0
--Generamos un número aleatorio entre el 1 y el 10:
Averia = math.random(1,10)
```

La función nos genera un número entero aleatorio comprendido entre el rango de números que le hagamos puesto. Si hemos puesto entre 1 y 10 tenemos un 10% de probabilidades para cada número comprendido en éste rango, así que cambiando el rango variamos ésta probabilidad más o menos. Usaremos éstos números para determinar si la señal está averiada o no. La ubicación de la función debería ser dentro de la `MainconfigPersonal` para que la avería sea determinada al inicio y

perdurando durante todo el escenario. En éste ejemplo se encuentra en la segunda con el rango de fallo del 1 a 10, así de cada 10 veces que se cargue la señal al menos una nos dará fallo.

A continuación vamos a simular la avería de que la señal no se abre (permanece en Parada). Para ello, lo primero que comprobaremos antes de determinar el estado es si el número que hemos elegido coincide con el número generado aleatoriamente. Si es así, el estado será Parada, si no continuará con la determinación normal del estado:

```
--La señal esta averiada y presenta una ocupación intempestiva:
if Averia == 5 then
    Estado = ESTADO_PARADA

--El siguiente estado es para señales intermedias que no son BAD:
elseif gSenallista == SENAL_NOLISTA and gParadoxDefecto == true and gBAD == false then
    --Estado: Anuncio de Parada en estado de reposo por ser una señal avanzada
    if gAvanzada == true then
        Estado = ESTADO_APARADA

    --Estado: Parada por señal no preparada
    else
        Estado = ESTADO_PARADA
    end
end

--2a Parte: Estados dependientes del estado de la siguiente señal. Aquí asignaremos el e
```

En éste caso hemos elegido el 5, pero podríamos haber elegido cualquier otro entre el 1 y 10, pues todos tienen la misma posibilidad de ser elegidos.

En el siguiente ejemplo pasaremos a detallar la segundo tipo de averías que vamos a implementar y que consiste en simular la fusión de la lámpara de un foco.

En las señales modernas la fusión de un foco no implica el apagado de la señal, si no la aparición de un estado mas restrictivo que se permita hacer con los otros focos, por ejemplo si se funde el foco verde la señal pasará a indicar Anuncio de Parada en sustitución de la Vía Libre, y si se funde el foco amarillo la señal pasaría a indicar Parada en sustitución al Anuncio de Parada. En el caso de la fusión del foco rojo no habría más remedio que dejar la señal apagada.

En la siguiente imagen se muestra el código de determinación del estado que se ha modificado para simular la avería del foco verde:

```

elseif EstadoProximaSenal == ESTADO_PARADA then
    Estado = ESTADO_APARADA

elseif EstadoProximaSenal == ESTADO_REBASE then
    Estado = ESTADO_APARADA

elseif EstadoProximaSenal == ESTADO_APARADA then
    --Si tiene pantalla de preanuncio:
    if gPant~="" then
        Estado = ESTADO_PREANUNCIO

        --Si no tiene debería ser vía libre, pero comprobamos si hay avería:
        elseif Averia == 7 then
            Estado = ESTADO_APARADA

        --En caso de no haber avería entonces es vía libre:
        else
            Estado = ESTADO_VIALIBRE
        end

elseif EstadoProximaSenal == ESTADO_APARADA_BIFURCACION then
    Estado = ESTADO_APRECAUCION

elseif EstadoProximaSenal == ESTADO_APARADA_BIFURCACION_PANTALLA then
    Estado = ESTADO_APRECAUCION
    PantallaAlfa = true

elseif EstadoProximaSenal == ESTADO_APARADA_BIFURCACION_SINPAJARITA then
    Estado = ESTADO_APARADA

elseif EstadoProximaSenal == ESTADO_APRECAUCION then
    --Si tiene pantalla de preanuncio:
    if gPant~="" then
        Estado = ESTADO_PREANUNCIO

        --Si no tiene debería ser vía libre, pero comprobamos si hay avería:
        elseif Averia == 7 then
            Estado = ESTADO_APARADA

        --En caso de no haber avería entonces es vía libre:
        else
            Estado = ESTADO_VIALIBRE
        end

    --Si tiene avería ignora los otros estados de vía libre:
    elseif Averia == 7 then
        Estado = ESTADO_APARADA

    elseif EstadoProximaSenal == ESTADO_PREANUNCIO then
        Estado = ESTADO_VIALIBRE

    elseif EstadoProximaSenal == ESTADO_VIALIBRECONDICIONAL then
        Estado = ESTADO_VIALIBRE

    elseif EstadoProximaSenal == ESTADO_VIALIBRE then
        Estado = ESTADO_VIALIBRE




    else
        Estado = ESTADO_APARADA
    end
end

```

Básicamente consiste en comprobar antes de determinar el estado Vía Libre si hay avería en la señal y pasar entonces a estado Anuncio de Parada o Preanuncio de Parada, si existe ésta modalidad en la señal. A parte de esto podemos combinar dos averías distintas en la misma señal simplemente asignando un número diferente para cada avería, pudiendo entonces la señal sufrir averías de distinto tipo cada vez que se genera un número aleatorio.

4. Listado de variables globales y etiquetas

A parte de las variables que se describen en la función `MainConfigPersonal()` existen las siguientes variables disponibles:

Variable	Descripción	Observaciones
gId	Corresponde al nombre que le hemos puesto a la señal.	En formato string
gLinkCount	Corresponde al número de links que tiene la señal, incluyendo el 0	Por ejemplo si <code>gLinkCount</code> es 3, significa que la señal tiene los links 0, 1 y 2.
gConnectedLink	Corresponde al link que conduce el itinerario formado por las agujas situadas a continuación de la señal.	Si vale -1, significa que no hay ningún itinerario válido o que la ruta es incompatible con otra ruta.
gOccupationTable[link]	Tabla de ocupaciones. Se trata de un vector que contiene el nivel de ocupación del link que se ha indexado. Si es superior a 0 significa que hay un vehículo o vehículos ocupando el link de destino.	Índice des del link 0 hasta el link <code>gLinkCount-1</code> . Por ejemplo si queremos saber la ocupación del link 2 la buscaremos por: <code>gOccupationTable[2]</code>
gSpeedTable[link]	Se trata del valor que el usuario ha introducido en la casilla de velocidad  de cada link.	En formato string.
gFeatherTable[link]	Se trata del valor que el usuario ha introducido en la casilla de letra  de cada link.	En formato string y solo almacena una única letra.
gSenalLista	Variable que determina el estado de preparación de la señal, para determinar si está en reposo o si se aproxima un tren a ella mediante la cadena de estados de preparación, explicada en el Manual.	Las etiquetas de cada preparación son: SENAL_NOLISTA SENAL_LISTA SENAL_PREPARADA SENAL_PREPARADA_N1 SENAL_PREPARADA_N2
gRebaseAuto	Ésta variable es “true” cuando se ha solicitado el rebase mediante la tecla TAB y sigue así durante los 2 minutos siguientes hasta que se termina el periodo de rebase.	Recordar que en el manual se informa que cuando se solicita el rebase con la tecla TAB la señal pasa a estar SENAL_LISTA.
gAvanzada	Ésta variable es “true” cuando la señal es avanzada de otra señal.	En caso de no serlo, el valor es “nil”, ya que no se inicializa.
linkRestrictivo	Si el link conectado (<code>gConnectedLink</code>) ha sido marcado por el usuario como link restrictivo  , entonces vale 1, altramente vale 0.	

Las siguientes tablas recogen las etiquetas disponibles para los estados y aspectos:

Tabla de Estados:


Etiqueta	Estado	observaciones
ESTADO_PARADA	Parada	
ESTADO_REBASE	Rebase Autorizado	
ESTADO_APARADA_BIFURACION	Vía desviada, con agujas a 30 km/h.	
ESTADO_APARADA_BIFURACION_PANTALLA	Vía desviada, con velocidad de paso por las agujas y dirección especificadas en las casillas de cada link 	Se cambia automáticamente al detectar dichos valores cuando el estado es ESTADO_APARADA_BIFURACION
ESTADO_APARADA_BIFURACION_SINPAJARITA	Vía desviada, con velocidad de paso por las agujas a 30km/h, aunque haya agujas con velocidad superior marcada en pajarita en la señal anterior. De ésta forma en vez de mostrar Anuncio de Precaución, podrá mostrar Anuncio de Parada. Ver manual	Se cambia automáticamente al detectar dichos valores cuando el estado es ESTADO_APARADA_BIFURACION
ESTADO_APARADA	Anuncio de Parada	
ESTADO_APRECAUCION	Anuncio de Precaución	
ESTADO_PREANUNCIO	Preanuncio de Parada	Necesita que gPant tenga algún número.
ESTADO_VIALIBRECONDICIONAL	Vía Libre Condicional	
ESTADO_VIALIBRE	Vía Libre	
ESTADO_MOVIMAUTO	Movimiento Autorizado	

Tabla de Aspectos:

Etiqueta	Aspecto	Ejemplo	Observaciones.
ANIMSTATE_PARADA	Parada		
ANIMSTATE_REBASEAUTO	Rebase Autorizado		
ANIMSTATE_REBASEAUTOINT	Rebase Autorizado, foco blanco intermitente.		
ANIMSTATE_APARADA	Anuncio de Parada		
ANIMSTATE_APARADAINM	Anuncio de Parada Inmediata		

ANIMSTATE_APRECAUCION	Anuncio de Precaución		La pajarita con la velocidad aparece de manera automática al cargarse el escenario.
ANIMSTATE_APRECAUCION_PANTALLA	Anuncio de Precaución con Pantalla Alfanumérica.		Tanto las flechas y los números aparecen de forma automática en función del link conectado en la señal siguiente.
ANIMSTATE_PREANUNCIO	Preanuncio de Parada.		
ANIMSTATE_VIALIBRECONDICIONAL	Vía libre Condicional		
ANIMSTATE_VIALIBRE	Vía Libre		
ANIMSTATE_MOVIMAUTO	Movimiento Autorizado.		

Agradecimientos

Quisiera agradecer a los autores que han participado en estas señales (**Camber, Xartix, renfe334, Alfredo, Cajoa y Muertez**) por su excelente trabajo y en especial al autor de los scripts BCN06 por cedérmelos de manera totalmente desinteresada por tal de que yo pudiera hacerles alguna mejora.

Por otro lado también agradecer la paciencia de los usuarios por tener que esperar de forma tan prolongada ésta actualización.