Contents

Things used in this project

Hardware

• Raspberry Pi 4 Computer Modell B, 4GB RAM	x 1
• Argon ONE V2 Case für Raspberry Pi 4	x 1
• Official Raspberry Pi USB-C Power Supply 5,1V / 3,0A, EU, black	x 1
• ESP32 NodeMCU Module Dev Kit C Dev Board with CP2102	x 1
• NodeMCU v2 - ESP8266 Development Board, CP2102	x 1
• GPS-14A, size 2,5x14mm	x 4
• USB cables	x 2
• USB 4 Port Power Supply / 5V / 5.0A with Auto-ID white	x 1

Software

- Rocrail
- Mosquitto
- Mattzobricks Firmware
    - MTC4BT (V0.5)
    - MLC (V0.5)
- Arduino IDE
- VS Code
    - Platform IO

Lego Pieces

Lego set #75955 – Hogwarts Express train	
Lego train track straight	x 8
Lego train track curved	x 4
Custom coal car v2.7	
Extra wagons	x 2
Modified locomotive	
Lego Plate, Modified 1 x 1 with Light Attachment - Thick Ring	x 4

- No: 4081b Alternate Item No: 41632
- Dark bluish gray

Story

While the Lego Power Up Functions iPad app works great for getting your trains running on a layout quickly, it does not allow for the level of automation I wanted. I began researching what other AFOL have done and stumbled upon Mattzobrick's YouTube channel. I was impressed with the flexibility and the collection of Lego trains. They have been constantly updating the project and it seems to be going in a good direction.

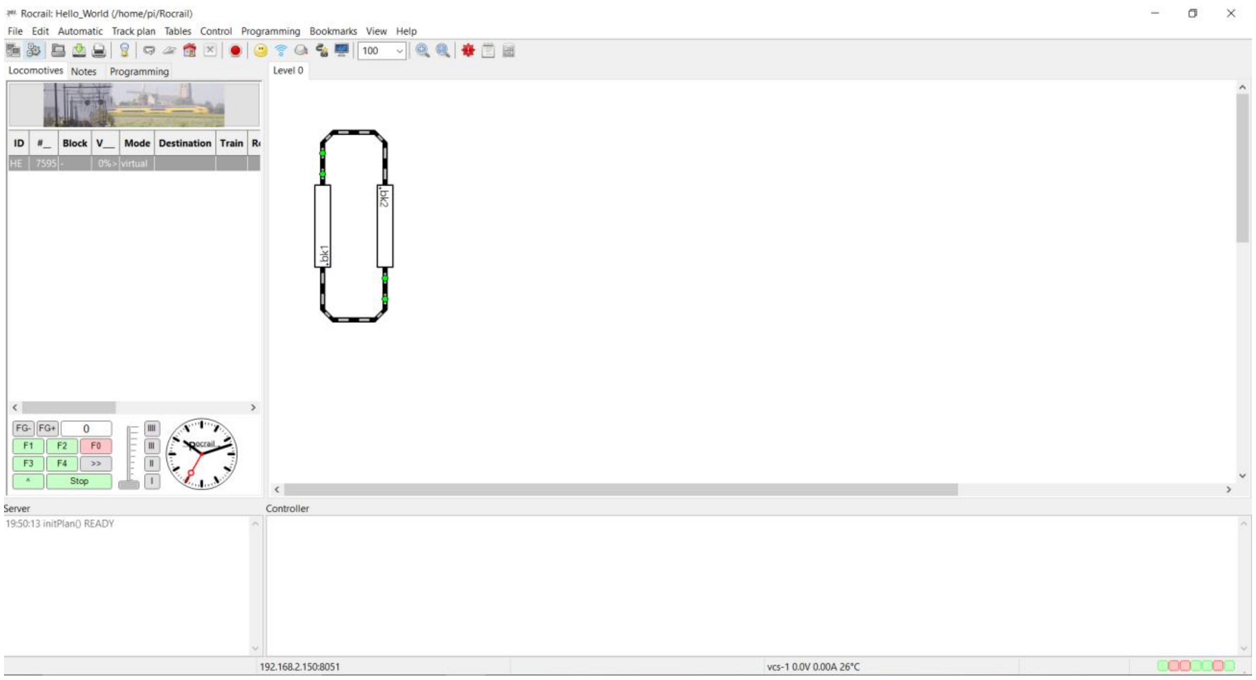
One of the latest changes was to firmware version 0.5, in which they added the MTCBT. This is what I am using to control the locomotives. This allows the ESP 32 board to connect to the Lego Powered Up Bluetooth hub, just like one would with and iPad or the Lego controller. This worked great and as I couldn't get the older version working.

My first goal was to design the minimum viable layout, where I could test enough of the pieces to have a running system. While the Rocrail and Mattzo websites, forums, and documentation are extremely detailed, I wanted to still document the layout I had done. I hope it too can help others interested in trying to automate their trains, because there is no end to the creativity or customization you can two with these products.

How it works

The Mattzo team has a great diagram of the overall system architecture.

The oval is the simplest layout to create in Rocrail and what most Lego trains come with originally. So that is what I modeled in Rocrail. In order for this to work, you need to have a minimum of 2 stations (or blocks) in the layout.


Rocrail: Hello_World (/home/pi/Rocrail)
File Edit Automatic Track plan Tables Control Programming Bookmarks View Help
Locomotives Notes Programming
Level 0
ID #_ Block V_ Mode Destination Train
HE 7595 - 0% > virtual
bk1
bk2
FG- FG+ 0
F1 F2 F0
F3 F4 >>
Stop
Server
Controller
192.168.2.150:8051
vcs-1 0.0V 0.00A 26°C

```
19:50:13 initPlan() READY
```

Step Overview

1. Setup Raspberry Pi
2. Add sensors
3. Upload firmware
4. Configure Rocrail and Rocview

Step 1 Setup Raspberry Pi

Install Rocrail on Raspberry Pi

The first thing to do is Install Rocrail server on RPI 4. You can choose to run the Rocrail server on your PC, but I chose to have it on the Raspberry Pi. This allowed it to be always running.

It is also a good idea to setup the Auto start on RPI so that the Rocrail server boots up, when the Raspberry Pi starts.

The Rocrail software allows for server & client architecture. Since the RPi is running the server, you have to setup the client on a PC. I set up Rocview on my Windows 10 PC.

Install Mosquitto on Raspberry Pi

Follow instructions for setting up a Mosquitto server on Mattzobricks and Rocrail websites.

```
mosquitto version 1.5.7 starting
```

Mosquitto configuration

Where are the logs located? - /var/log/mosquitto

How to check that it is working

topic '#'

Testing mosquitto

```
mosquitto_sub -d -t testTopic
```

```
mosquitto_sub -d -t rocrail/service/command
```

```
mosquitto_pub -d -t testTopic -m "Hello world!"
```

```
sudo systemctl stop mosquitto.service
```

```
sudo systemctl start mosquitto.service
```

```
sudo systemctl restart mosquitto.service
```

https://wiki.rocrail.net/doku.php?id=rocrailini-service-en#mqtt_service

Step 2 Add sensors

Modify Lego to hold sensors

The sensors, that I found on Amazon.de ended up having too large of a diameter for the Lego pieces. Therefore, I had to drill out the Lego pieces, in order for them to fit. I had purchased all of them from a seller on Brick Link. I first drilled out the hole to 3.5 mm and then to 4.0 mm. Then the sensors were able to fit.

Connect sensors to ESP 8266

The layout controller can handle more than 4 sensors, but this was the bare minimum needed to have 2 blocks in Rocrail going a single direction. The sensors do not have polarity. They should all have 1 lead connected to ground and the other connected to an input on the development board.

https://mattzobricks.com/automation/sensors

Step 4 Upload firmware

There currently a view different firmware versions available. For this project I used version 0.5.

Upload Mattzo firmware (MLC) to ESP 8266

To upload the firmware I used the Arduino IDE.

Upload Mattzo firmare (MTC4BT) to ESP 32

For this firmware, the Mattzo team has created a workspace for Visual Studio Code. With the help of some extensions VS Code provides a great IDE for working with the ESP32.

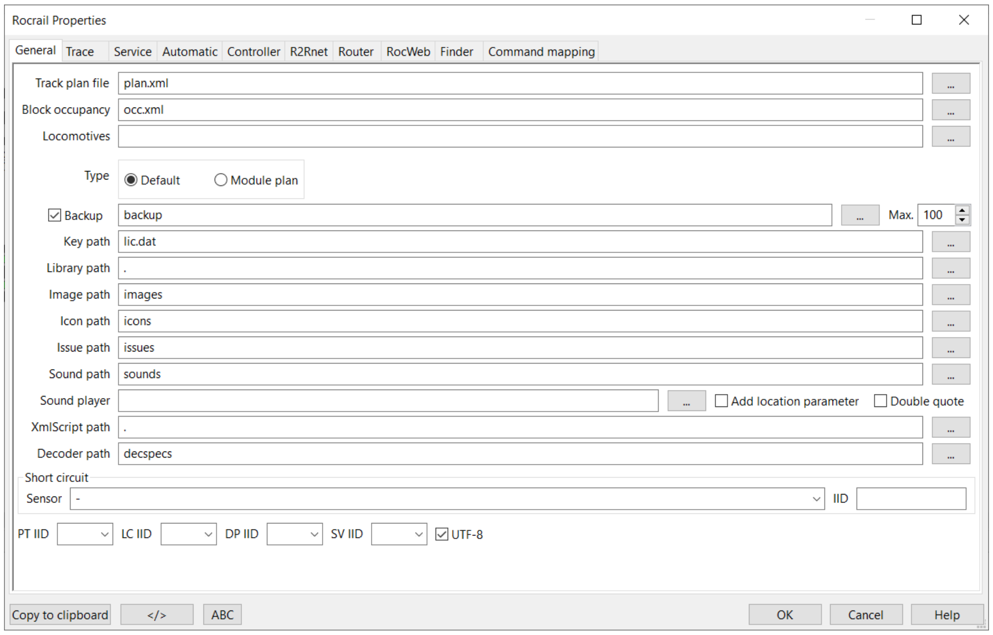
Step 5 Configure Rocrail and Rocview

Here are some helpful videos on how to setup your Rocrail layout.

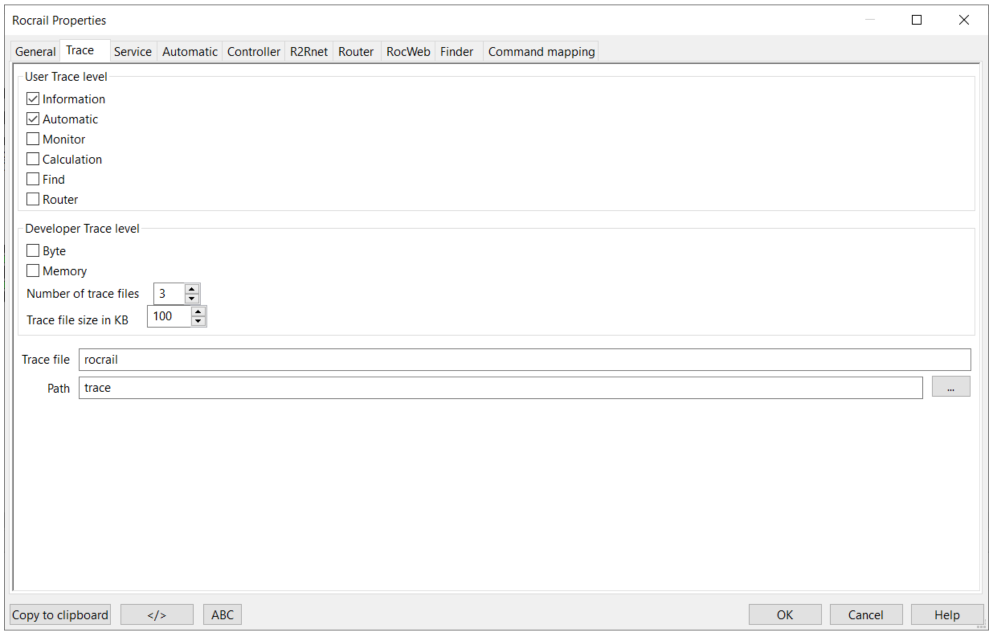
https://www.youtube.com/watch?v=V0APPfda9xA&t=113s

https://www.youtube.com/watch?v=tf_V-Ghm0YE&t=8s

Firstly, you need to configure the Rocrail server properties.



Track plan file: plan.xml
Block occupancy: occ.xml
Locomotives:
Type: [✓ Default] Module plan
[✓ Backup] backup Max. 100
Key path: lic.dat
Library path: .
Image path: images
Icon path: icons
Issue path: issues
Sound path: sounds
Sound player: ☐ Add location parameter ☐ Double quote
XmlScript path: .
Decoder path: decspecs
Short circuit
Sensor: - IID
PT IID LC IID DP IID SV IID [✓ UTF-8]


Rocrail Properties

General | Trace | Service | Automatic | Controller | R2Rnet | Router | RocWeb | Finder | Command mapping

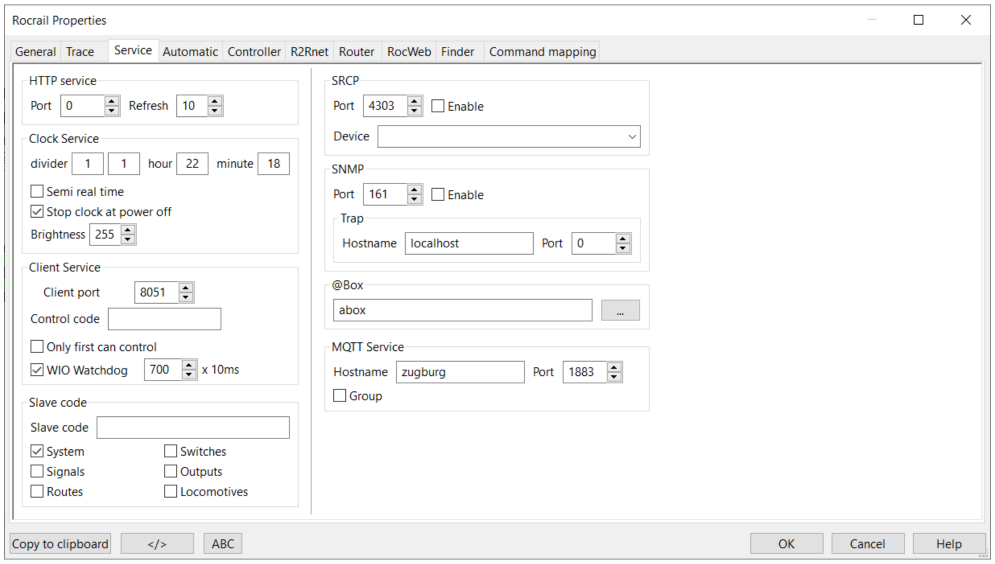
User Trace level
- [x] Information
- [x] Automatic
- [ ] Monitor
- [ ] Calculation
- [ ] Find
- [ ] Router

Developer Trace level
- [ ] Byte
- [ ] Memory

Number of trace files: 3
Trace file size in KB: 100

Trace file: rocrail
Path: trace

The key parameters are under the 'Service' tab for the hostname and port for the MQTT service. You should enter the host name or ip address of the Raspberry pi. The default port for Mosquitto is 1883.


Rocrail Properties

General | Trace | Service | Automatic | Controller | R2Rnet | Router | RocWeb | Finder | Command mapping

HTTP service
Port 0 Refresh 10

Clock Service
divider 1 1 hour 22 minute 18
- [ ] Semi real time
- [x] Stop clock at power off

Brightness 255

Client Service
Client port 8051
Control code
- [ ] Only first can control
- [x] WIO Watchdog 700 x 10ms

Slave code
Slave code
- [x] System
- [ ] Switches
- [ ] Signals
- [ ] Outputs
- [ ] Routes
- [ ] Locomotives

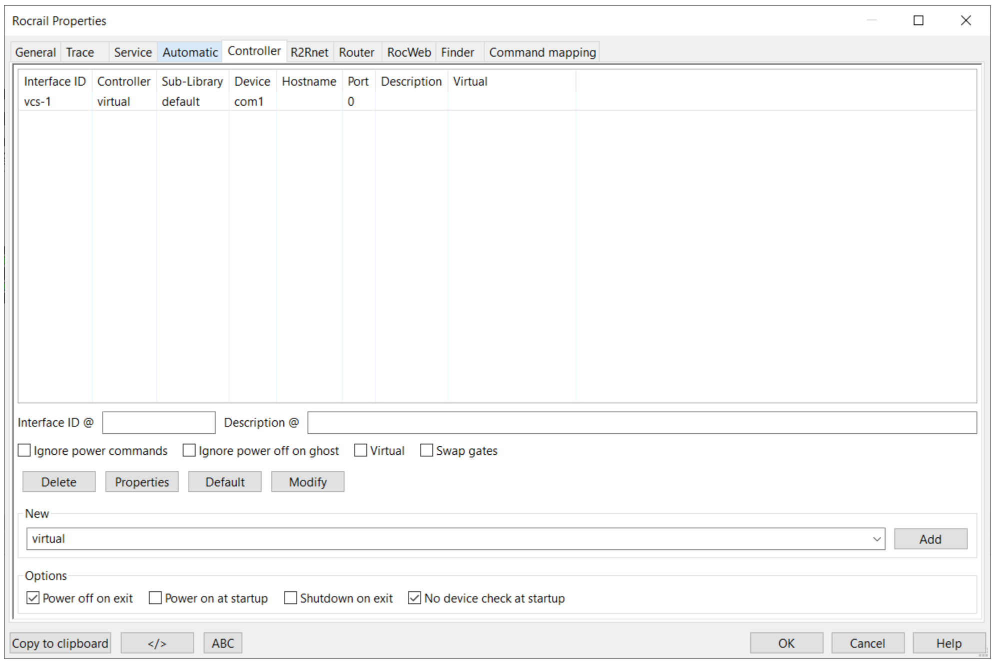
SRCP
Port 4303 [ ] Enable
Device

SNMP
Port 161 [ ] Enable
Trap
Hostname localhost Port 0

@Box
abox

MQTT Service
Hostname zugburg Port 1883
- [ ] Group

On the 'Controller' tab you need to create at least 1 controller. You can see the settings in the following figure.



Rocrail Properties

Interface ID	Controller	Sub-Library	Device	Hostname	Port	Description	Virtual
vcs-1	virtual	default	com1		0		

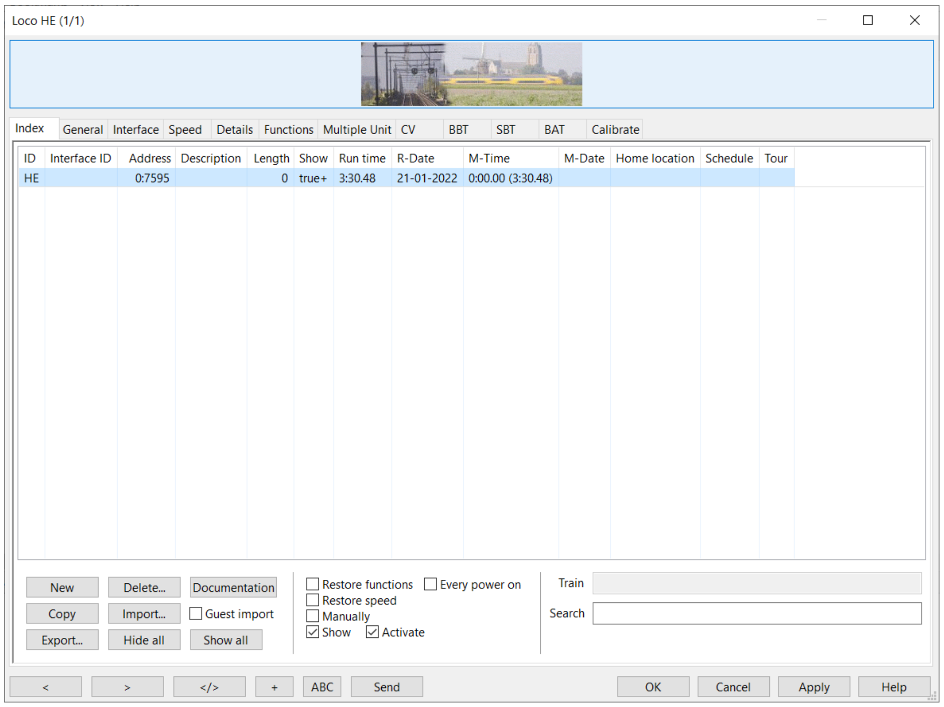
Interface ID @ Description @

☐ Ignore power commands ☐ Ignore power off on ghost ☐ Virtual ☐ Swap gates

New: virtual

Options: ☑ Power off on exit ☐ Power on at startup ☐ Shutdown on exit ☑ No device check at startup

Next you need to setup at least one locomotive. In Rocrail, you can also setup trains, but in this case it isn’t necessary.



Loco HE (1/1)

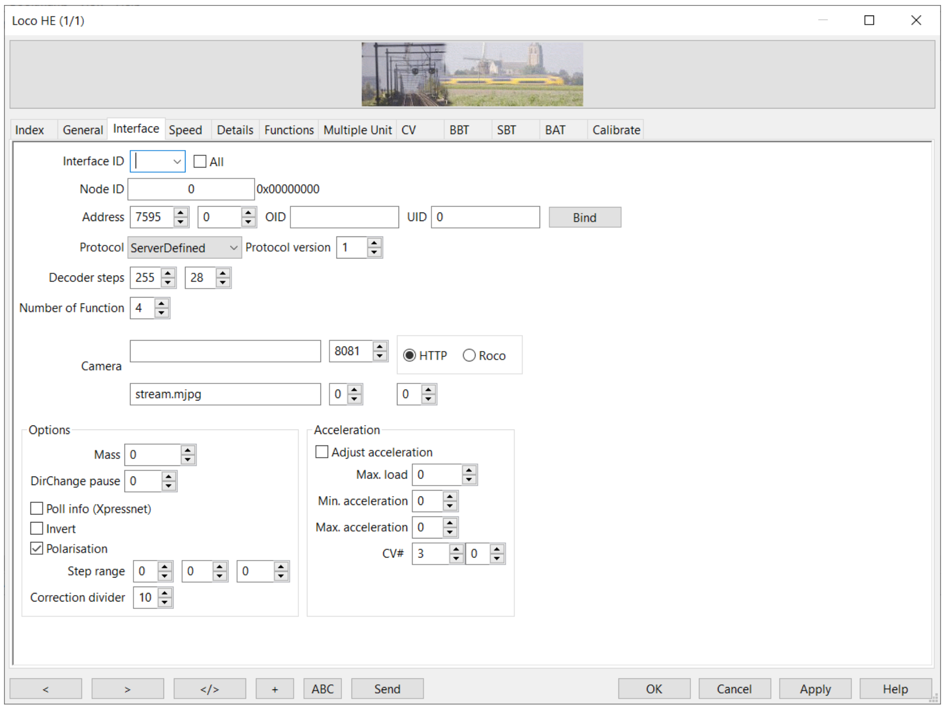
ID	Interface ID	Address	Description	Length	Show	Run time	R-Date	M-Time	M-Date	Home location	Schedule	Tour
HE		0:7595		0	true+	3:30.48	21-01-2022	0:00.00 (3:30.48)				

☐ Guest import

☐ Restore functions ☐ Every power on
☐ Restore speed
☐ Manually
☑ Show ☑ Activate

Train

Search


Loco HE (1/1)

Index | General | Interface | Speed | Details | Functions | Multiple Unit | CV | BBT | SBT | BAT | Calibrate

Interface ID [ ] ☐ All
Node ID 0 0x00000000
Address 7595 0 OID UID 0 Bind
Protocol ServerDefined Protocol version 1
Decoder steps 255 28
Number of Function 4

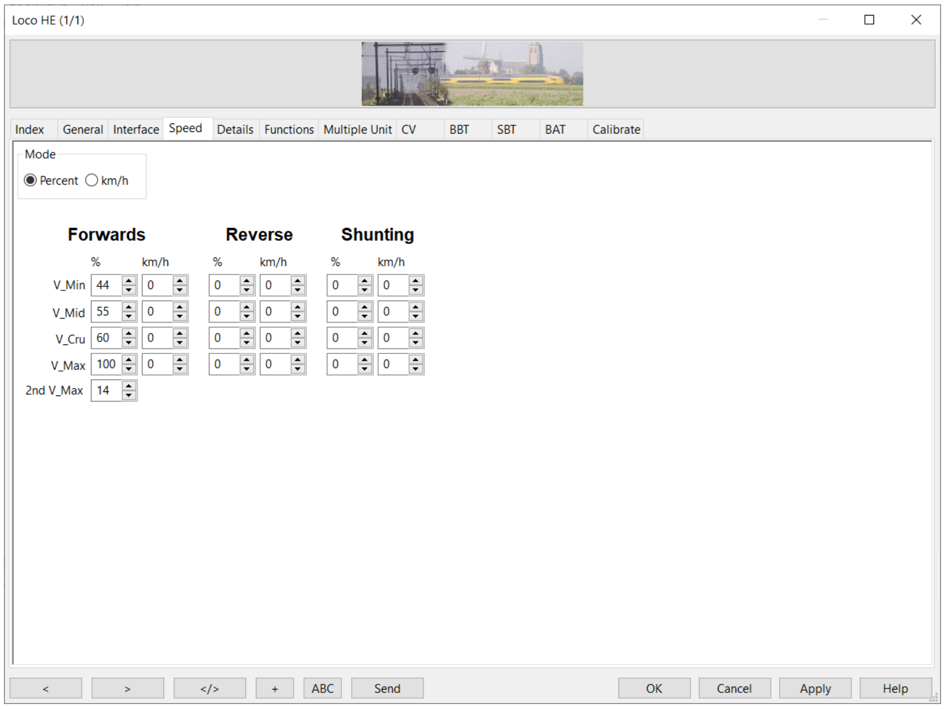
Camera 8081 [● HTTP] ○ Roco
stream.mjpg 0 0

Options
Mass 0
DirChange pause 0
☐ Poll info (Xpressnet)
☐ Invert
☑ Polarisation
Step range 0 0 0
Correction divider 10

Acceleration
☐ Adjust acceleration
Max. load 0
Min. acceleration 0
Max. acceleration 0
CV# 3 0

< > </> + ABC Send OK Cancel Apply Help

The ‘Speed’ of the locomotive can be set under the locomotive or under

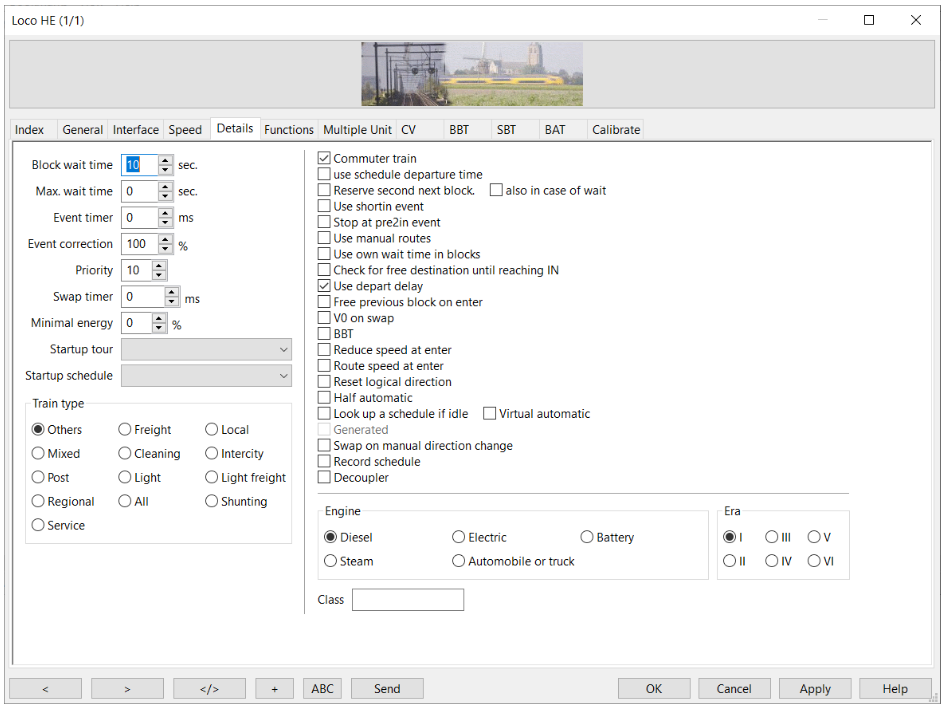

Loco HE (1/1)

Index | General | Interface | Speed | Details | Functions | Multiple Unit | CV | BBT | SBT | BAT | Calibrate

Mode
[● Percent] ○ km/h

	Forwards		Reverse		Shunting	
	%	km/h	%	km/h	%	km/h
V_Min	44	0	0	0	0	0
V_Mid	55	0	0	0	0	0
V_Cru	60	0	0	0	0	0
V_Max	100	0	0	0	0	0
2nd V_Max	14					

< > </> + ABC Send OK Cancel Apply Help



Block wait time 10 sec.
Max. wait time 0 sec.
Event timer 0 ms
Event correction 100 %
Priority 10
Swap timer 0 ms
Minimal energy 0 %
Startup tour
Startup schedule

Train type
● Others ○ Freight ○ Local
○ Mixed ○ Cleaning ○ Intercity
○ Post ○ Light ○ Light freight
○ Regional ○ All ○ Shunting
○ Service

☑ Commuter train
☐ use schedule departure time
☐ Reserve second next block. ☐ also in case of wait
☐ Use shortin event
☐ Stop at pre2in event
☐ Use manual routes
☐ Use own wait time in blocks
☐ Check for free destination until reaching IN
☑ Use depart delay
☐ Free previous block on enter
☐ V0 on swap
☐ BBT
☐ Reduce speed at enter
☐ Route speed at enter
☐ Reset logical direction
☐ Half automatic
☐ Look up a schedule if idle ☐ Virtual automatic
☐ Generated
☐ Swap on manual direction change
☐ Record schedule
☐ Decoupler

Engine
● Diesel ○ Electric ○ Battery
○ Steam ○ Automobile or truck

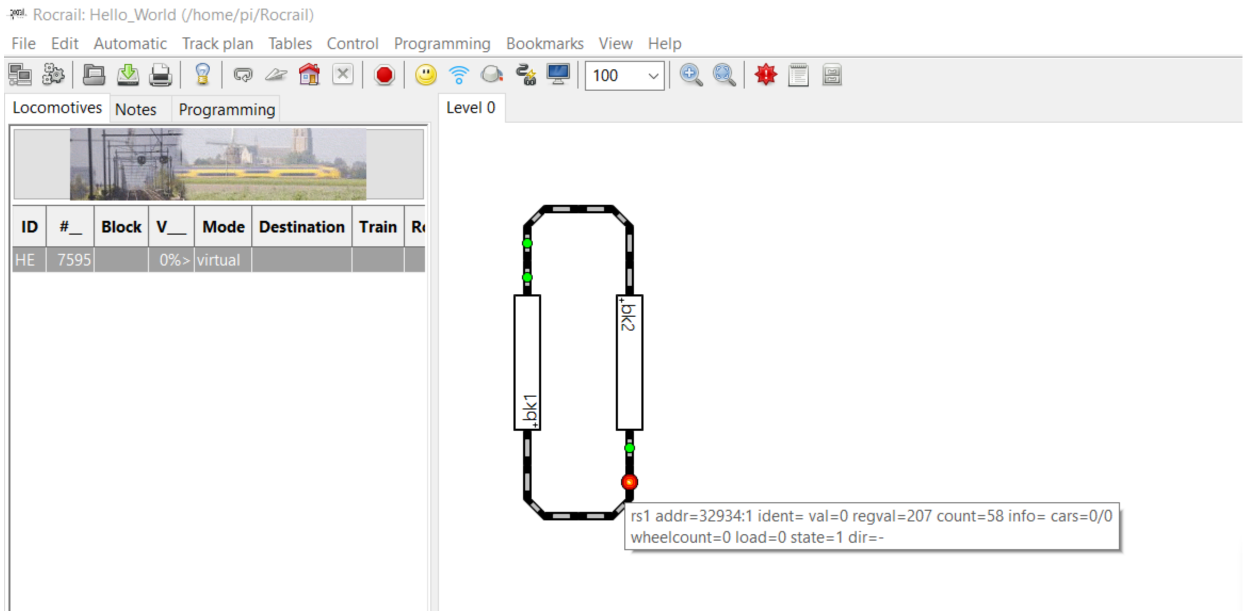
Era
● I ○ III ○ V
○ II ○ IV ○ VI

Class

Then we should setup some sensors. The more sensors the better. The absolute minimum could be two (2) sensors. However, the documentation recommends for two sensors before each block. The first sensor that will be tripped for example rs1 will trigger the '*enter*' event type and then the next sensor rs2, which is closest to the block will trigger the '*in*' event.

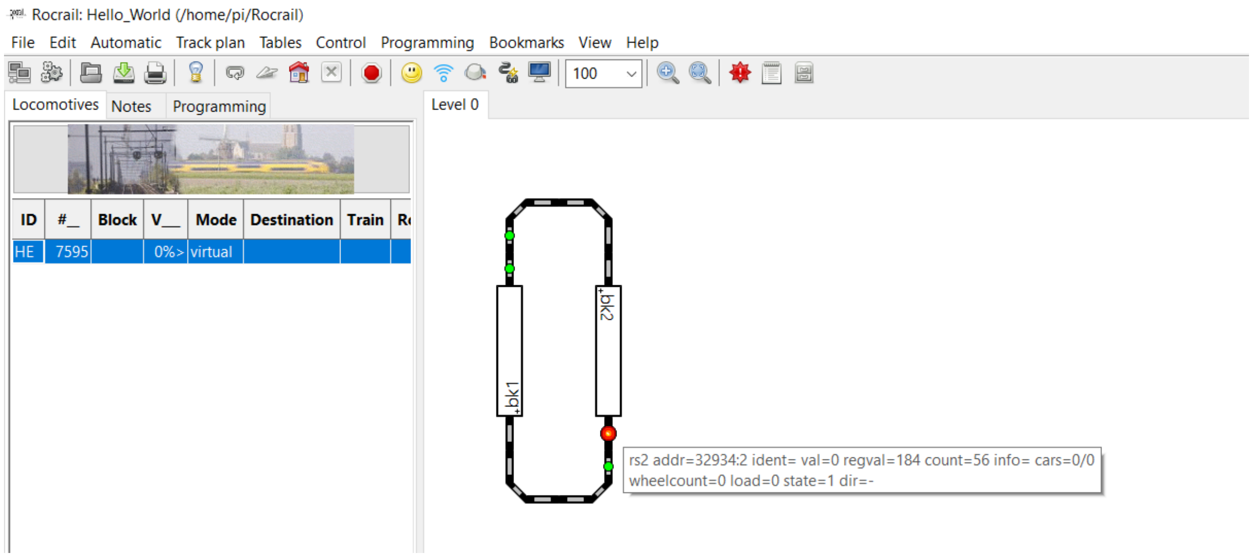
Most times the magnets on the Lego train buffers will not trigger the sensor to switch. However, the Lego motor was consistent enough for my scenario, and this is what I used in this case. There are some suggestions on the Mattzobricks website on how to add magnets to your wagons to have a sensor reading for the first and last wagon in the train. I would definitely go this route, if setting up a more complex layout with multiple trains, directions, and varying lengths of the trains.

The address of the sensor


Rocrail: Hello_World (/home/pi/Rocrail)
File Edit Automatic Track plan Tables Control Programming Bookmarks View Help
100
Locomotives Notes Programming
Level 0
bk1
bk2

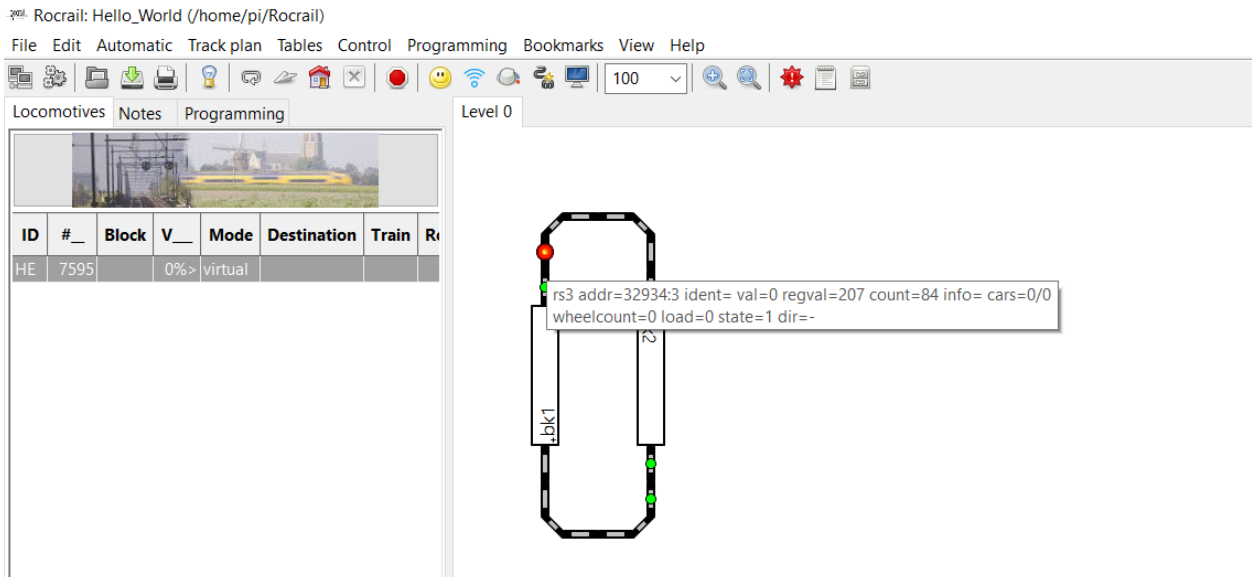
ID	#_	Block	V_	Mode	Destination	Train	R
HE	7595		0%>	virtual			

```
rs1 addr=32934:1 ident= val=0 regval=207 count=58 info= cars=0/0
wheelcount=0 load=0 state=1 dir=-
```


Rocrail: Hello_World (/home/pi/Rocrail)
File Edit Automatic Track plan Tables Control Programming Bookmarks View Help
100
Locomotives Notes Programming
Level 0
bk1
bk2

ID	#_	Block	V_	Mode	Destination	Train	R
HE	7595		0%>	virtual			

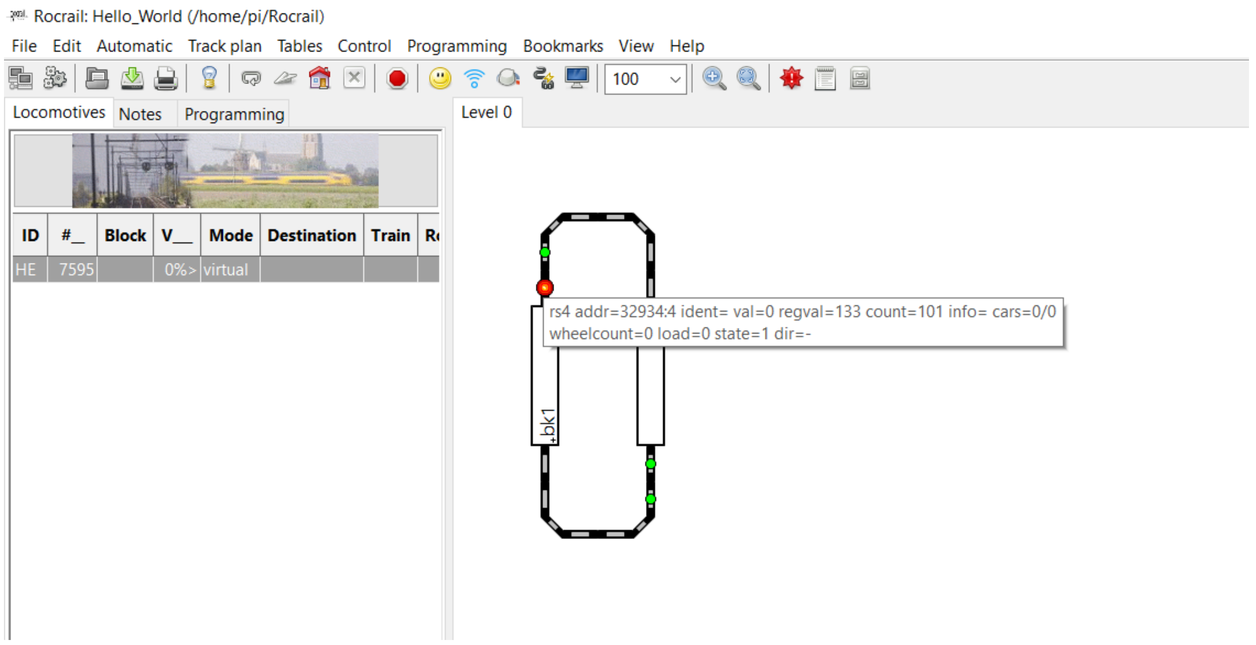
```
rs2 addr=32934:2 ident= val=0 regval=184 count=56 info= cars=0/0
wheelcount=0 load=0 state=1 dir=-
```


Rocrail: Hello_World (/home/pi/Rocrail)

File Edit Automatic Track plan Tables Control Programming Bookmarks View Help
100
Locomotives
Notes
Programming
Level 0
bk1

ID	#_	Block	V_	Mode	Destination	Train	R
HE	7595		0%>	virtual			

rs3 addr=32934:3 ident= val=0 regval=207 count=84 info= cars=0/0
wheelcount=0 load=0 state=1 dir=-

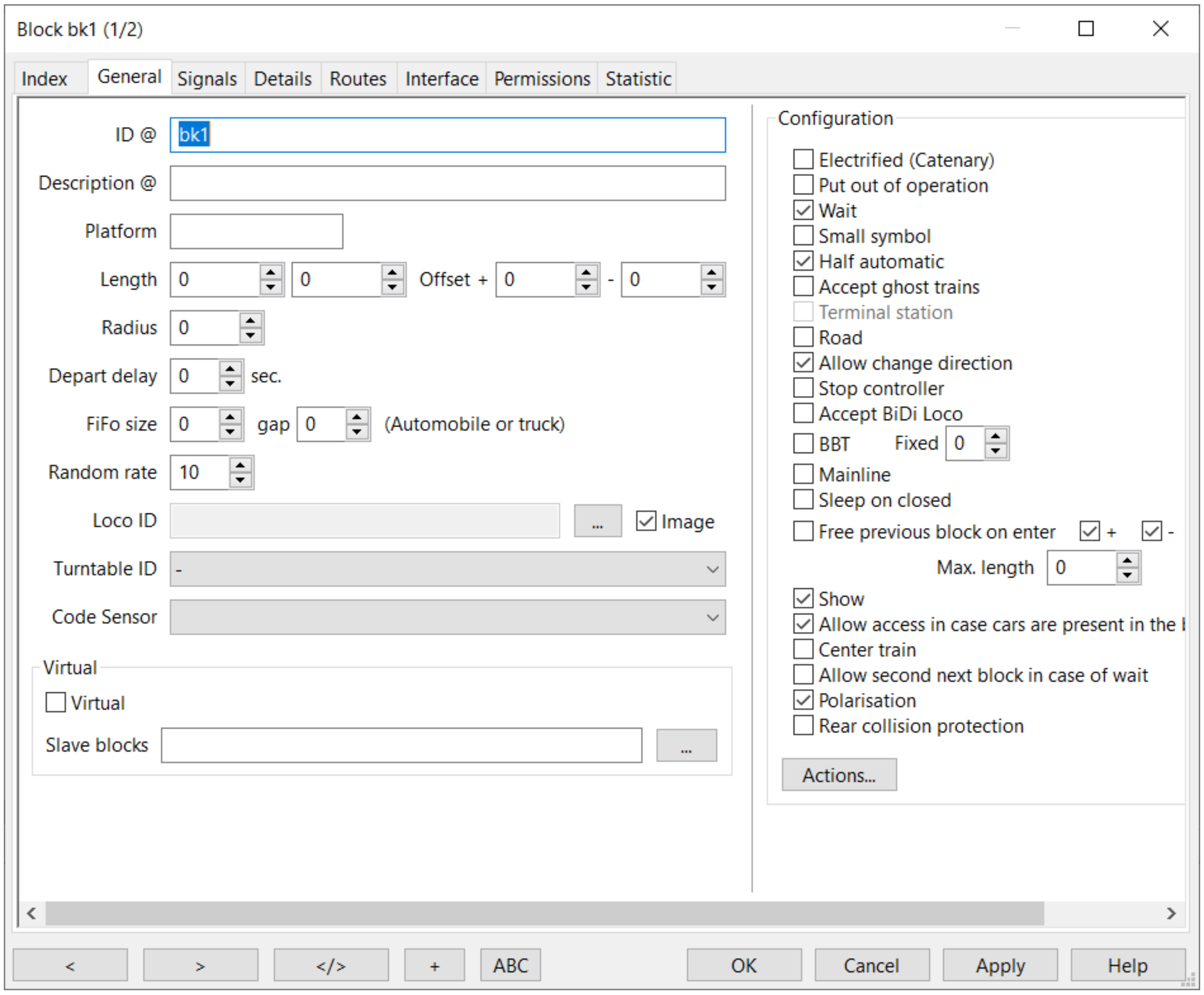

Rocrail: Hello_World (/home/pi/Rocrail)

File Edit Automatic Track plan Tables Control Programming Bookmarks View Help
100
Locomotives
Notes
Programming
Level 0
bk1

ID	#_	Block	V_	Mode	Destination	Train	R
HE	7595		0%>	virtual			

rs4 addr=32934:4 ident= val=0 regval=133 count=101 info= cars=0/0
wheelcount=0 load=0 state=1 dir=-

Next we can setup the blocks.



Block bk1 (1/2)

Index | General | Signals | Details | Routes | Interface | Permissions | Statistic

ID @	bk1
Description @	
Platform	
Length	0 0 Offset + 0 - 0
Radius	0
Depart delay	0 sec.
FiFo size	0 gap 0 (Automobile or truck)
Random rate	10
Loco ID	... ☑ Image
Turntable ID	-
Code Sensor	

Virtual

☐ Virtual

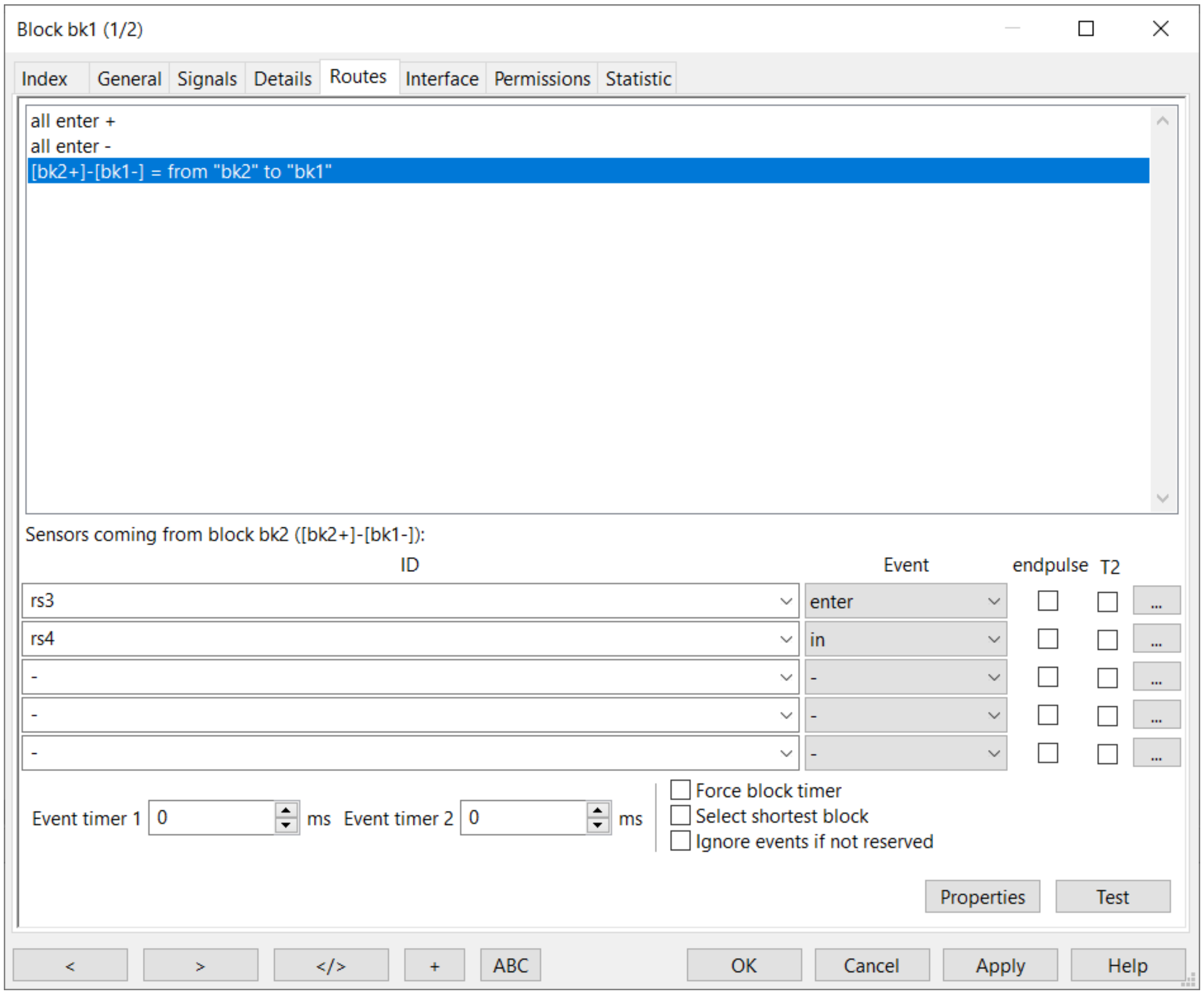
Slave blocks ...

Configuration

- ☐ Electrified (Catenary)
- ☐ Put out of operation
- ☑ Wait
- ☐ Small symbol
- ☑ Half automatic
- ☐ Accept ghost trains
- ☐ Terminal station
- ☐ Road
- ☑ Allow change direction
- ☐ Stop controller
- ☐ Accept BiDi Loco
- ☐ BBT Fixed 0
- ☐ Mainline
- ☐ Sleep on closed
- ☐ Free previous block on enter ☑ + ☑ -

Max. length 0

- ☑ Show
- ☑ Allow access in case cars are present in the
- ☐ Center train
- ☐ Allow second next block in case of wait
- ☑ Polarisation
- ☐ Rear collision protection



Block bk1 (1/2)

Index | General | Signals | Details | Routes | Interface | Permissions | Statistic

all enter +
all enter -
[bk2+]-[bk1-] = from "bk2" to "bk1"

Sensors coming from block bk2 ([bk2+]-[bk1-]):

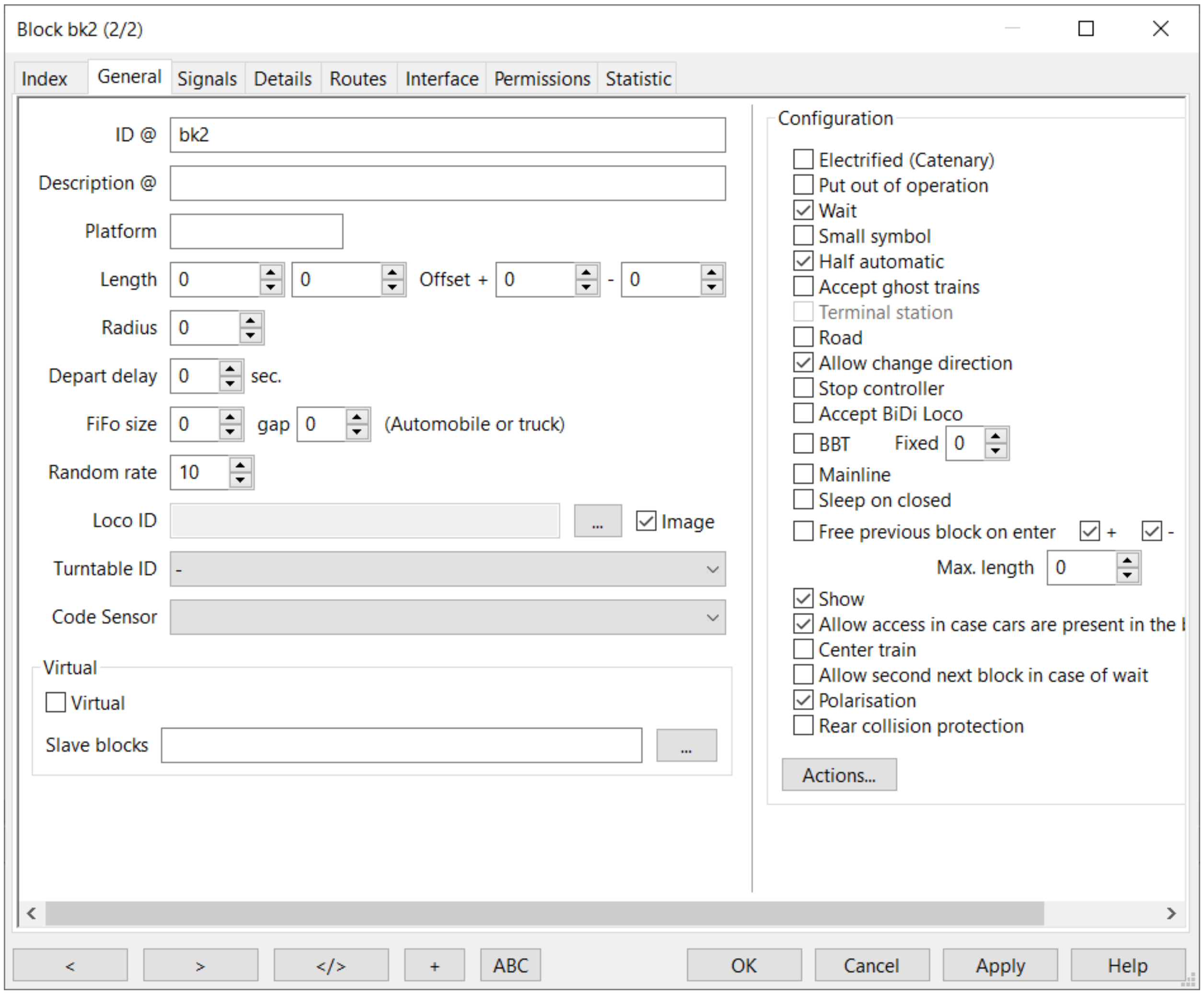
ID	Event	endpulse	T2	
rs3	enter	☐	☐	...
rs4	in	☐	☐	...
-	-	☐	☐	...
-	-	☐	☐	...
-	-	☐	☐	...

Event timer 1 0 ms Event timer 2 0 ms

☐ Force block timer
☐ Select shortest block
☐ Ignore events if not reserved

Properties | Test

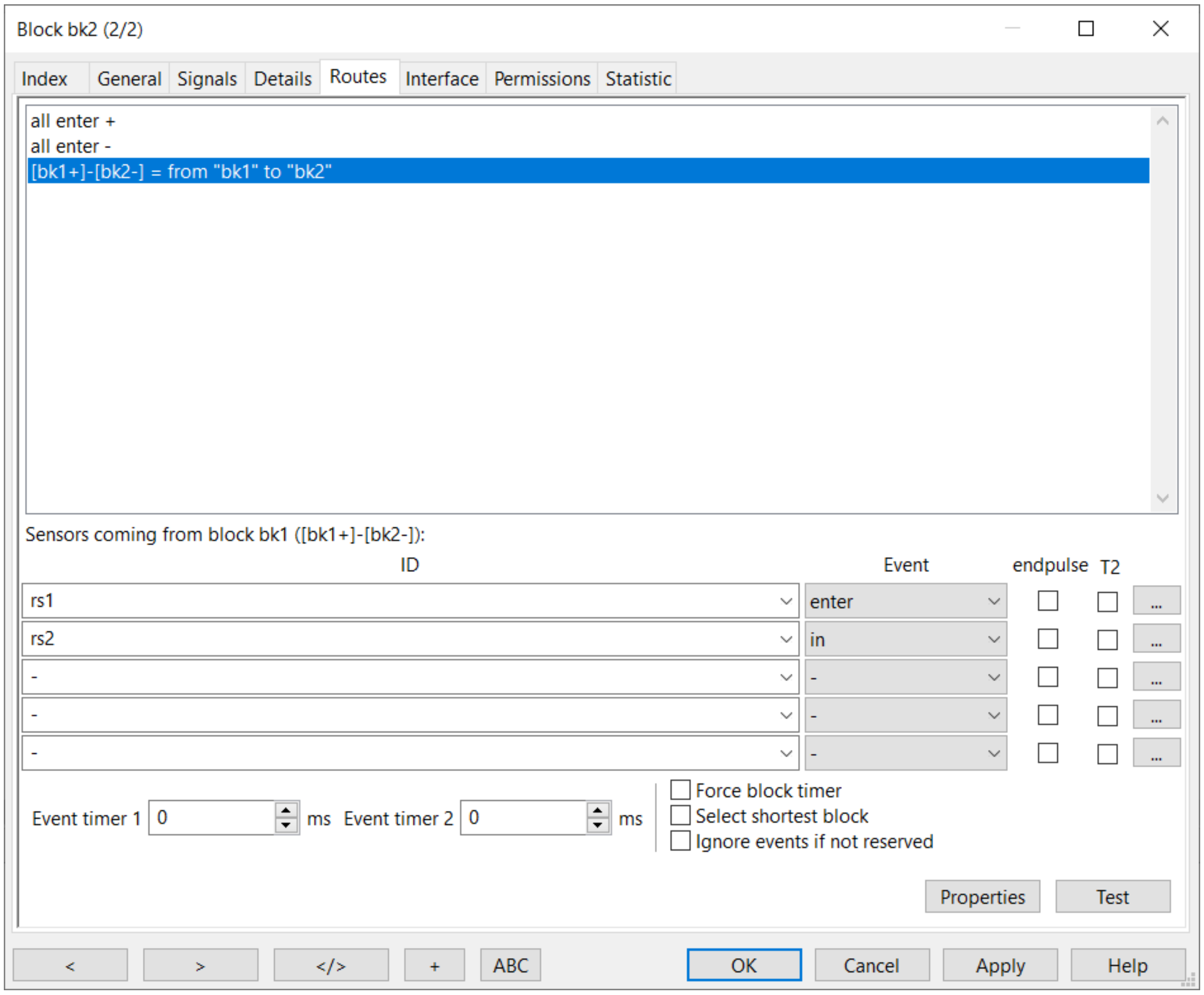
< | > | </> | + | ABC | OK | Cancel | Apply | Help


Block bk2 (2/2)

ID @ bk2
Description @
Platform
Length 0 0 Offset + 0 - 0
Radius 0
Depart delay 0 sec.
FiFo size 0 gap 0 (Automobile or truck)
Random rate 10
Loco ID ... ☑ Image
Turntable ID -
Code Sensor

Virtual
☐ Virtual
Slave blocks ...

Configuration
☐ Electrified (Catenary)
☐ Put out of operation
☑ Wait
☐ Small symbol
☑ Half automatic
☐ Accept ghost trains
☐ Terminal station
☐ Road
☑ Allow change direction
☐ Stop controller
☐ Accept BiDi Loco
☐ BBT Fixed 0
☐ Mainline
☐ Sleep on closed
☐ Free previous block on enter ☑ + ☑ -
Max. length 0
☑ Show
☑ Allow access in case cars are present in the
☐ Center train
☐ Allow second next block in case of wait
☑ Polarisation
☐ Rear collision protection



Block bk2 (2/2)

all enter +
all enter -
[bk1+]-[bk2-] = from "bk1" to "bk2"

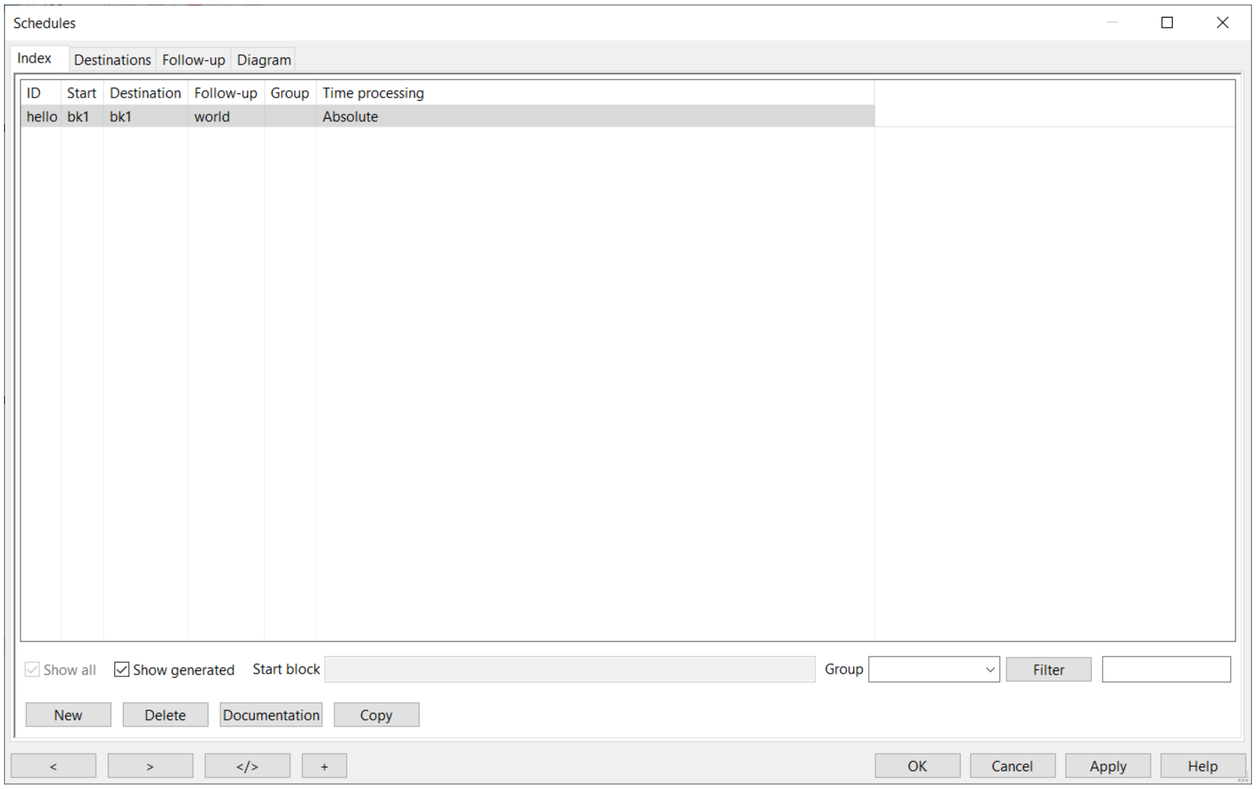
Sensors coming from block bk1 ([bk1+]-[bk2-]):

ID	Event	endpulse	T2
rs1	enter	☐	☐
rs2	in	☐	☐
-	-	☐	☐
-	-	☐	☐
-	-	☐	☐

Event timer 1 0 ms Event timer 2 0 ms

☐ Force block timer
☐ Select shortest block
☐ Ignore events if not reserved

The reason we needed to blocks was to be able to setup a schedule between the two. It is not possible in Rocrail to have a single block and make it go in a loop. The destination can’t equal the current location. In the schedule below, you can see that we start in block 1 then go to block 2 then go to block 1 again.



Schedules

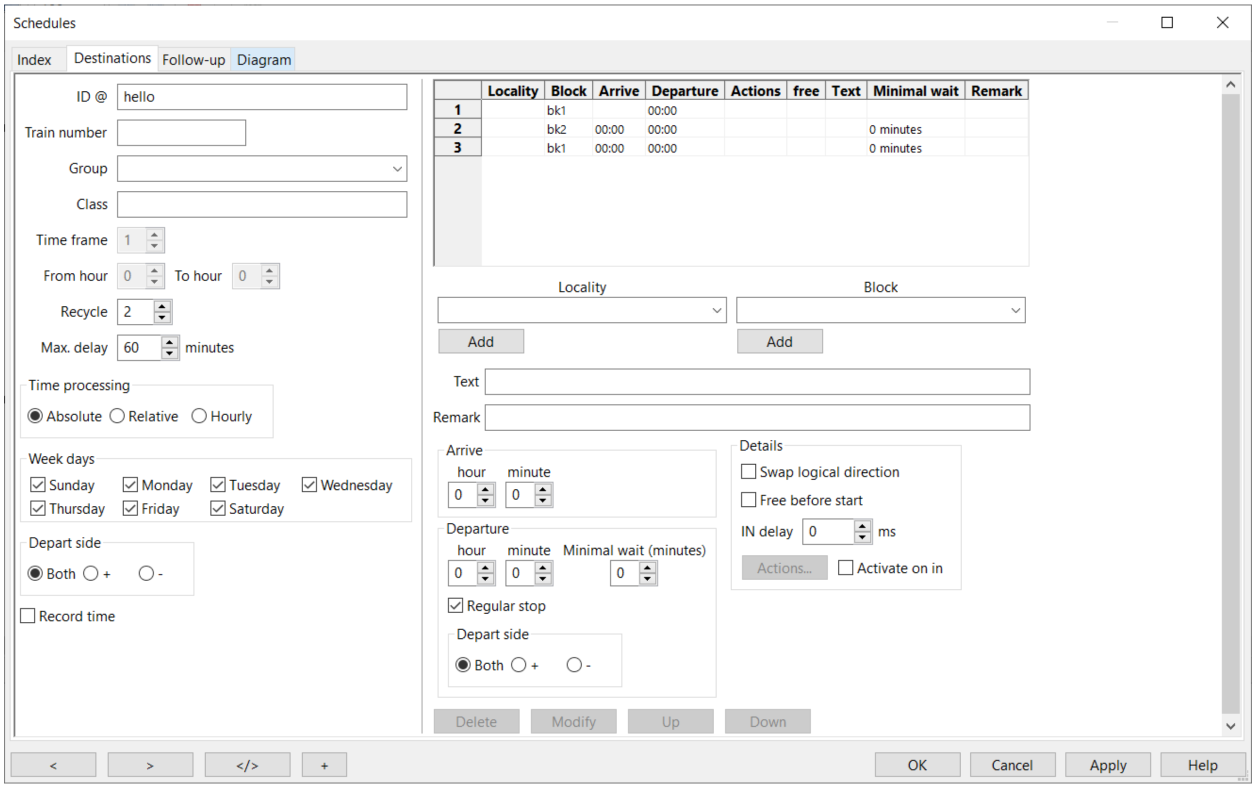
Index | Destinations | Follow-up | Diagram

ID	Start	Destination	Follow-up	Group	Time processing
hello	bk1	bk1	world		Absolute

Show all [✓ Show generated] Start block Group Filter

New Delete Documentation Copy

< > </> + OK Cancel Apply Help



Schedules

Index | Destinations | Follow-up | Diagram

ID @ hello

Train number

Group

Class

Time frame 1

From hour 0 To hour 0

Recycle 2

Max. delay 60 minutes

Time processing: [✓ Absolute] Relative Hourly

Week days: [✓ Sunday] [✓ Monday] [✓ Tuesday] [✓ Wednesday] [✓ Thursday] [✓ Friday] [✓ Saturday]

Depart side: [✓ Both] + -

Record time

	Locality	Block	Arrive	Departure	Actions	free	Text	Minimal wait	Remark
1		bk1		00:00					
2		bk2	00:00	00:00				0 minutes	
3		bk1	00:00	00:00				0 minutes	

Locality Add

Block Add

Text

Remark

Arrive: hour 0 minute 0

Departure: hour 0 minute 0 Minimal wait (minutes) 0

[✓ Regular stop]

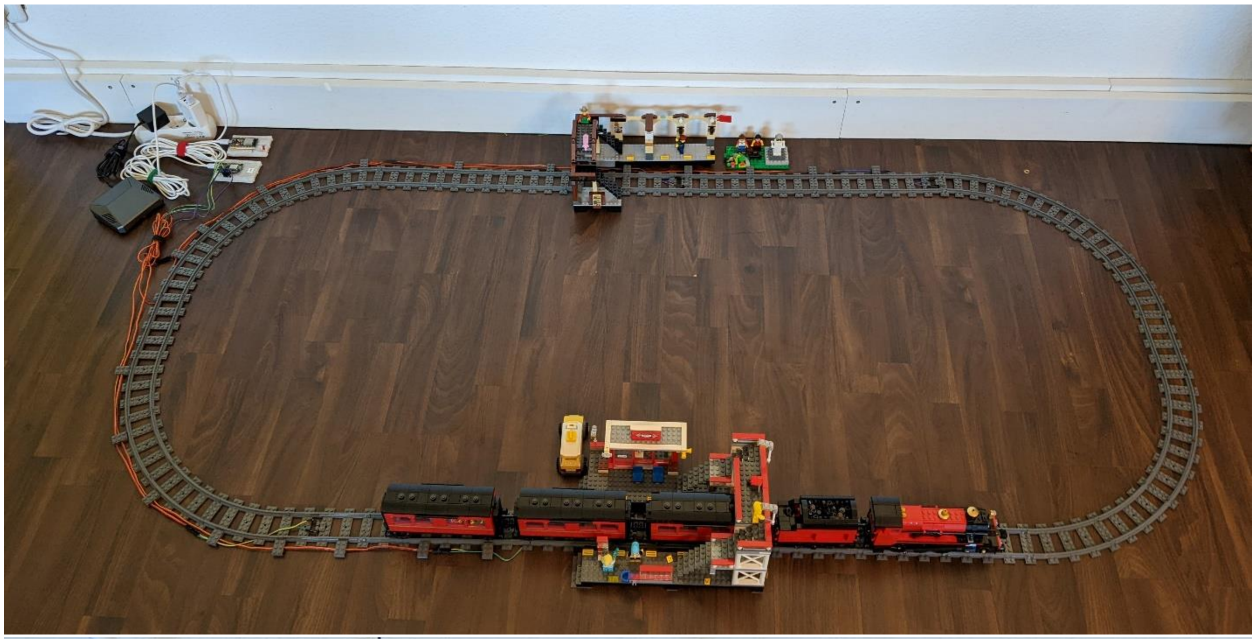
Depart side: [✓ Both] + -

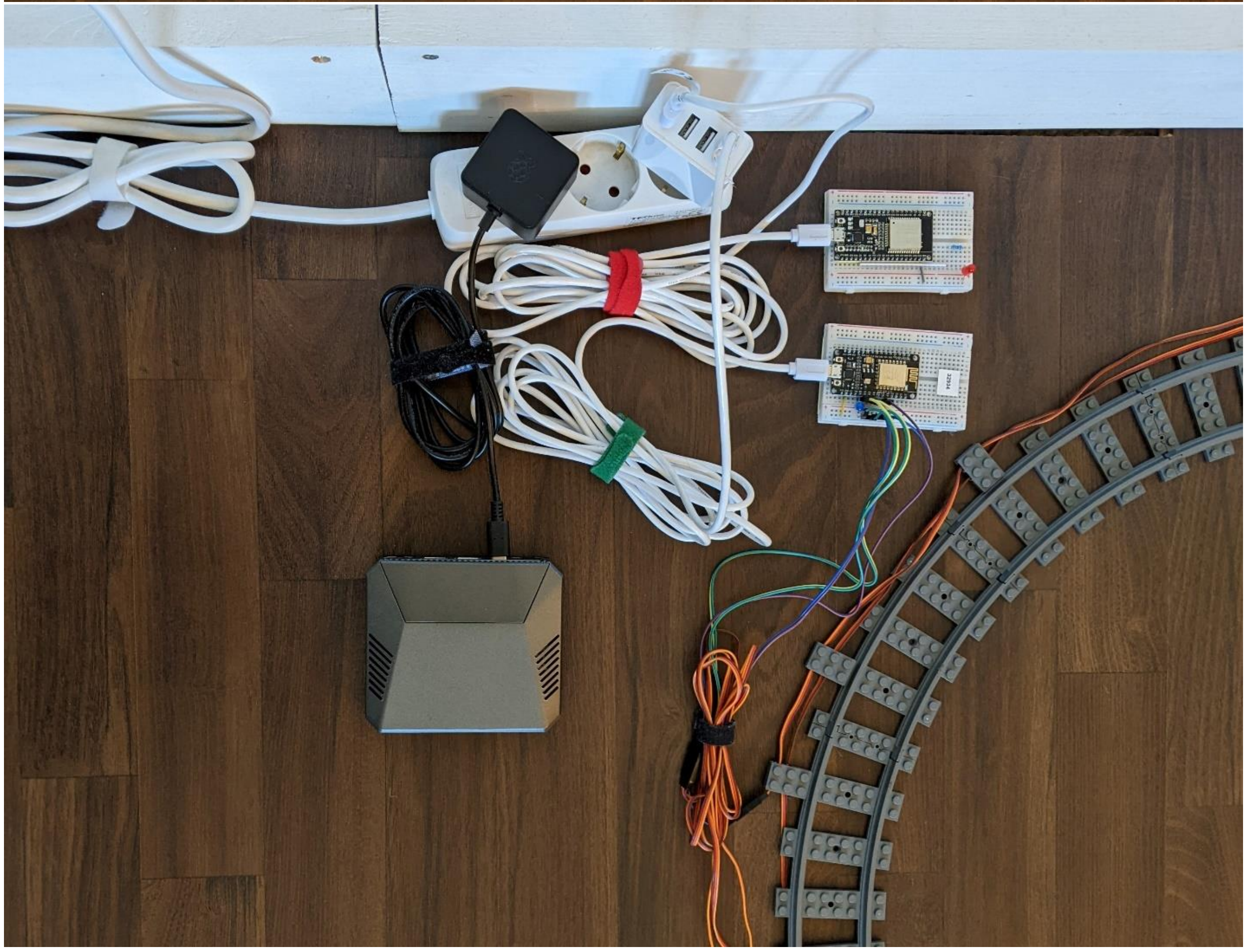
Details: Swap logical direction; Free before start; IN delay 0 ms; Actions...; Activate on in

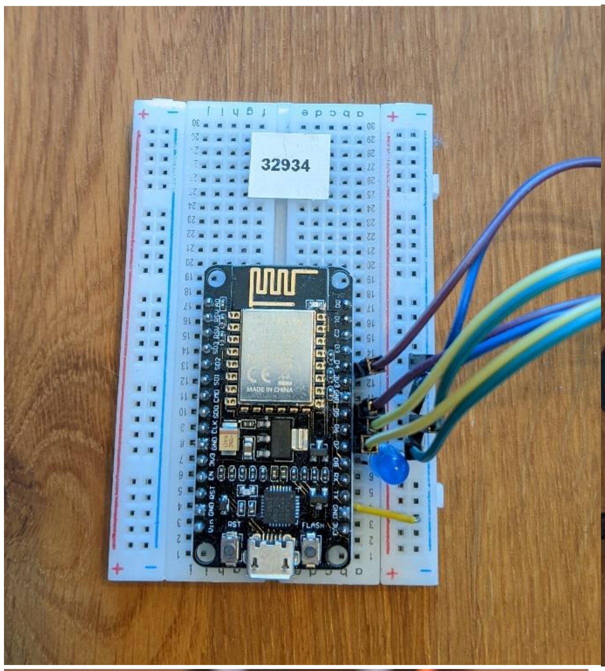
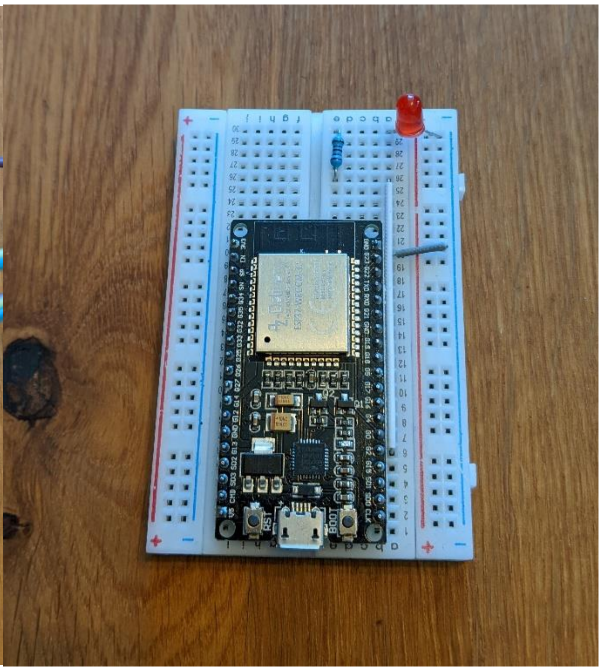
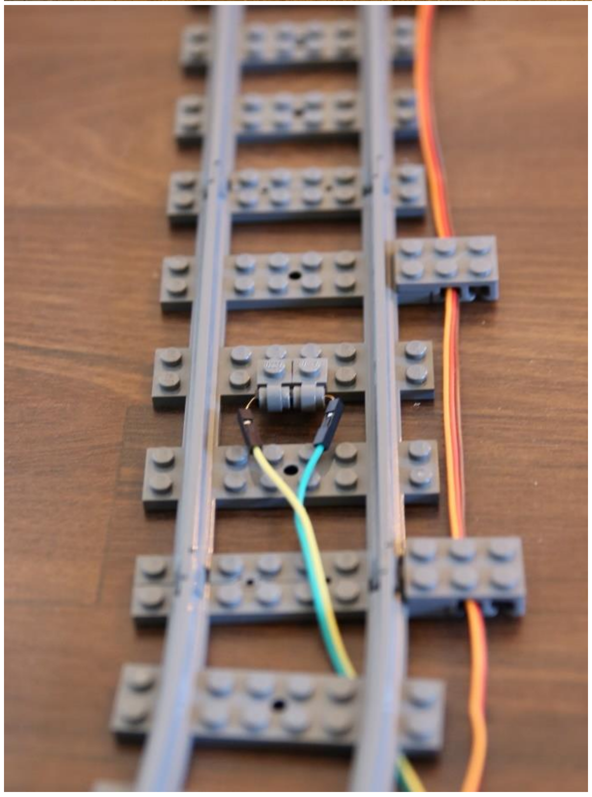
Delete Modify Up Down

< > </> + OK Cancel Apply Help

Demo






32934



Code

The code needed for this project is from a few different sources.

The code need for Rocrail can be found here.

The firmware for the development boards can be found here.

My modified configuration files for the Mattzo controllers can be found here.

References

https://mattzobricks.com/

https://wiki.rocrail.net/doku.php?id=start