MODELSPOORBAAN AUTOMATISERING IN PYTHON

Inleiding over het zelf ontwerpen en maken van **software** om treinen automatisch te laten rijden.

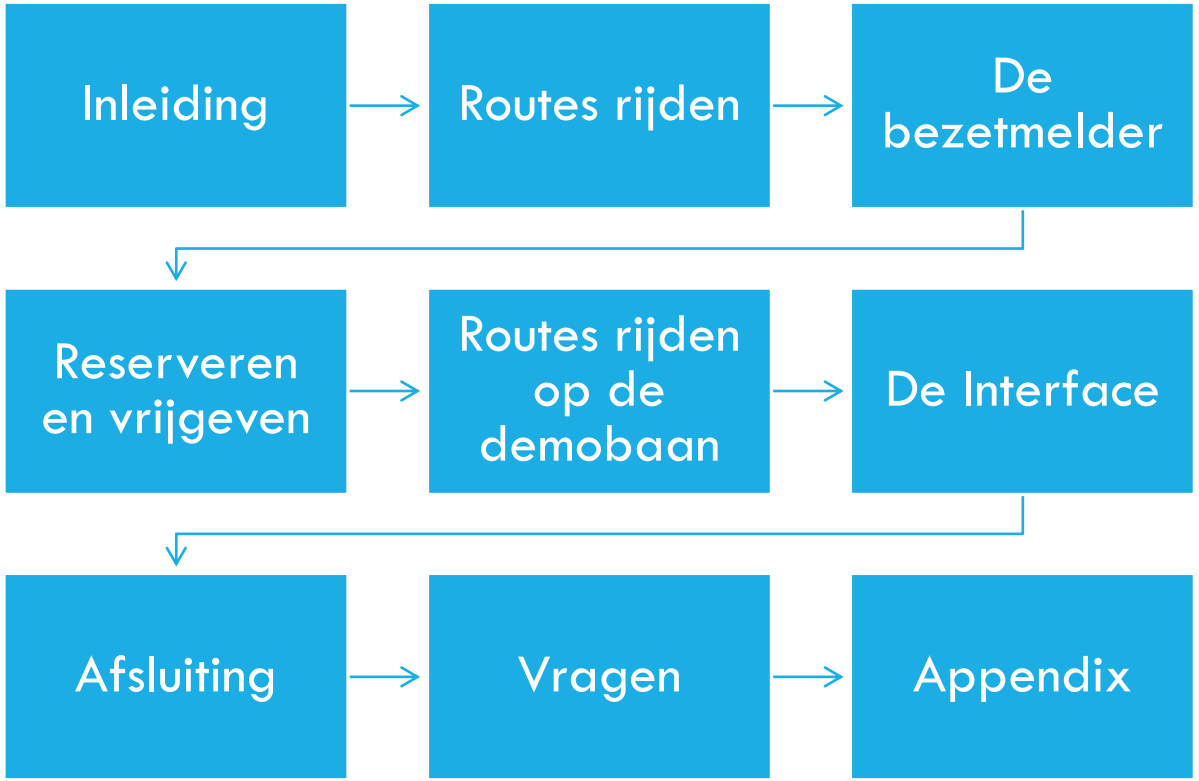
September 2022

Ir. Jan Verhoeven

jantwo@gmail.com

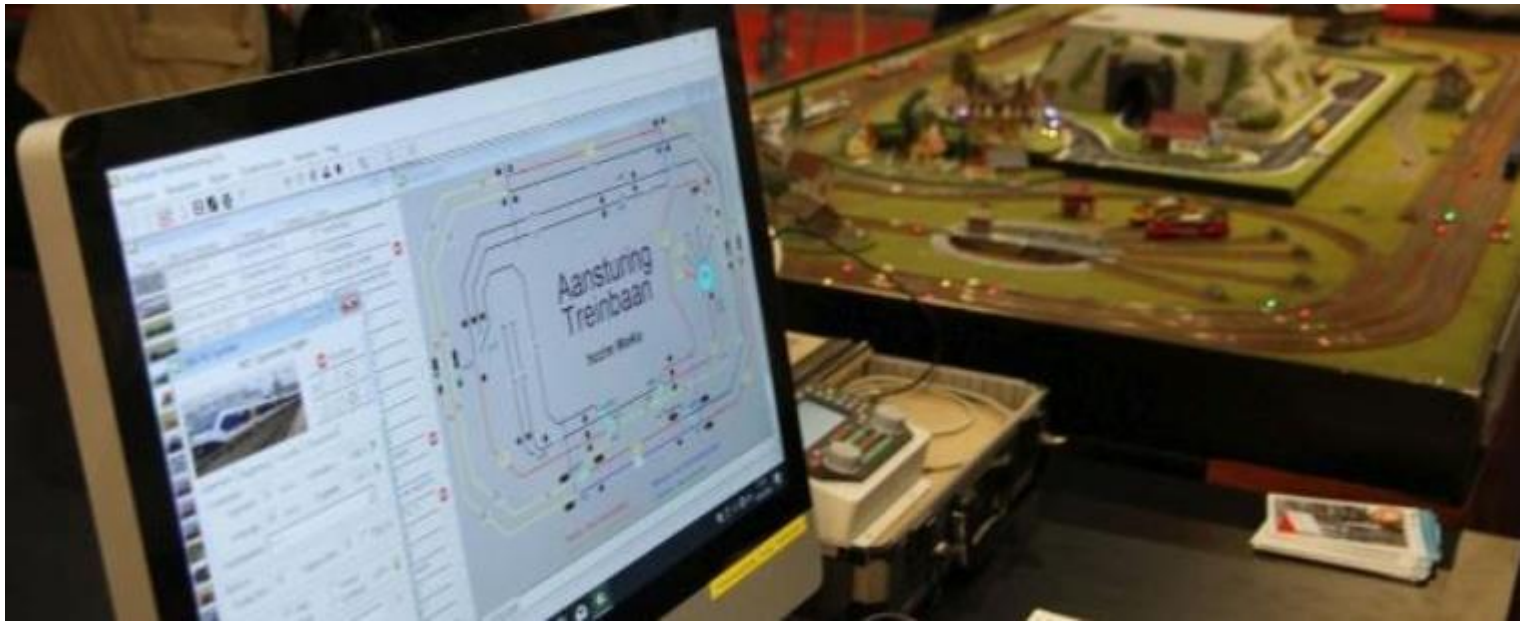
hcc!m

INHOUD



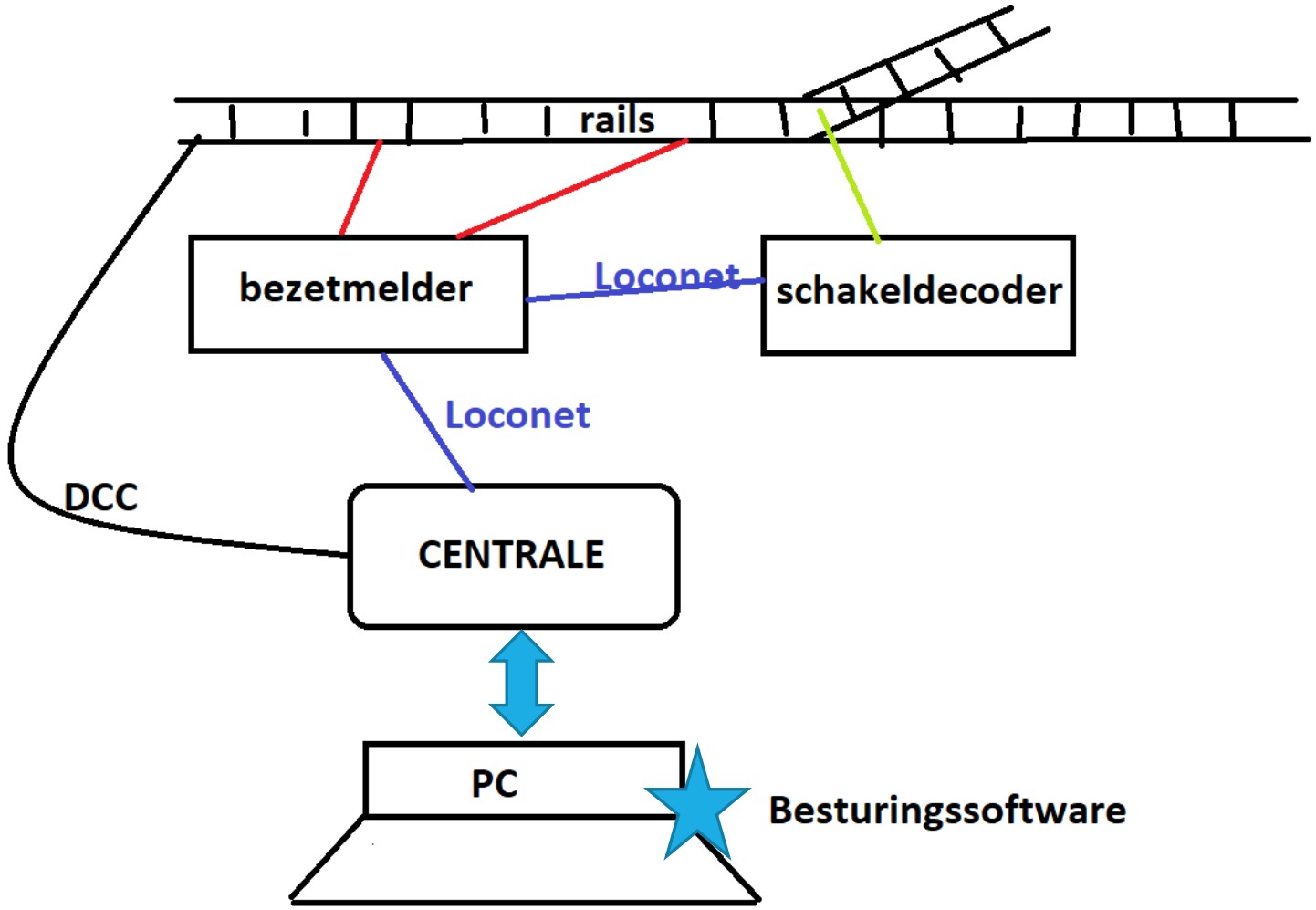
Wil zeggen 'behoorlijk technisch onderwerp'

INLEIDING



MIJN CONFIGURATIE

- Centrale: Intellibox Basic en Z21
- Uhlenbrock en Digikeijs
- Marklin C rails (drierail)
- LocoNet en DCC
- Windows en MacOS
- Besturing: Software in Python en C#



SOFTWARE ARCHITECTUUR

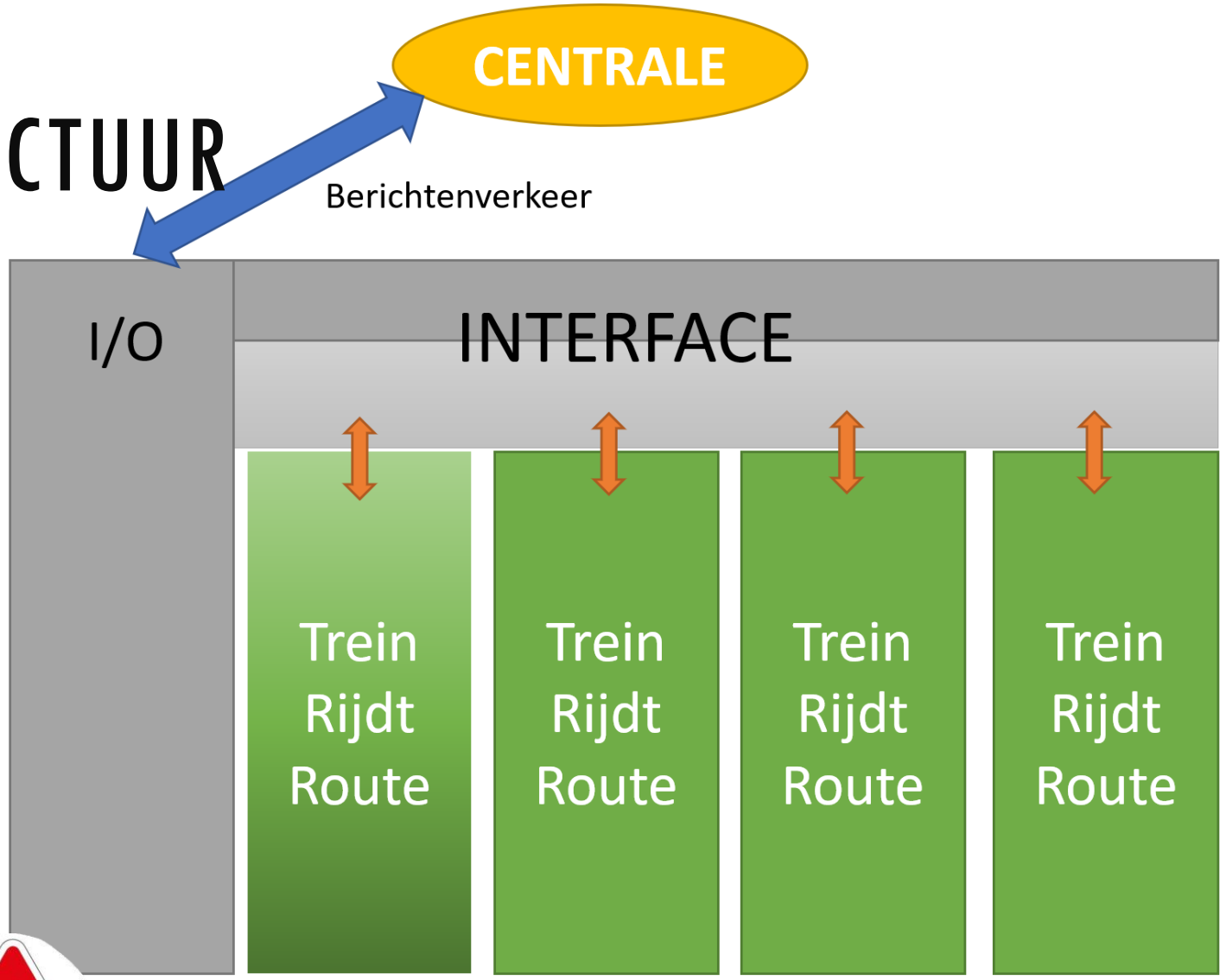
Besturingssoftware:

Python programma met *threads*:

- Interface
- Interface I/O met Centrale
- Per trein een route rijden

Multithreading maakt het mogelijk om meerdere delen van een programma parallel ("tegelijkertijd") uit te voeren.

Zie details over *threads* in https://docs.python.org/3/library/threading.html



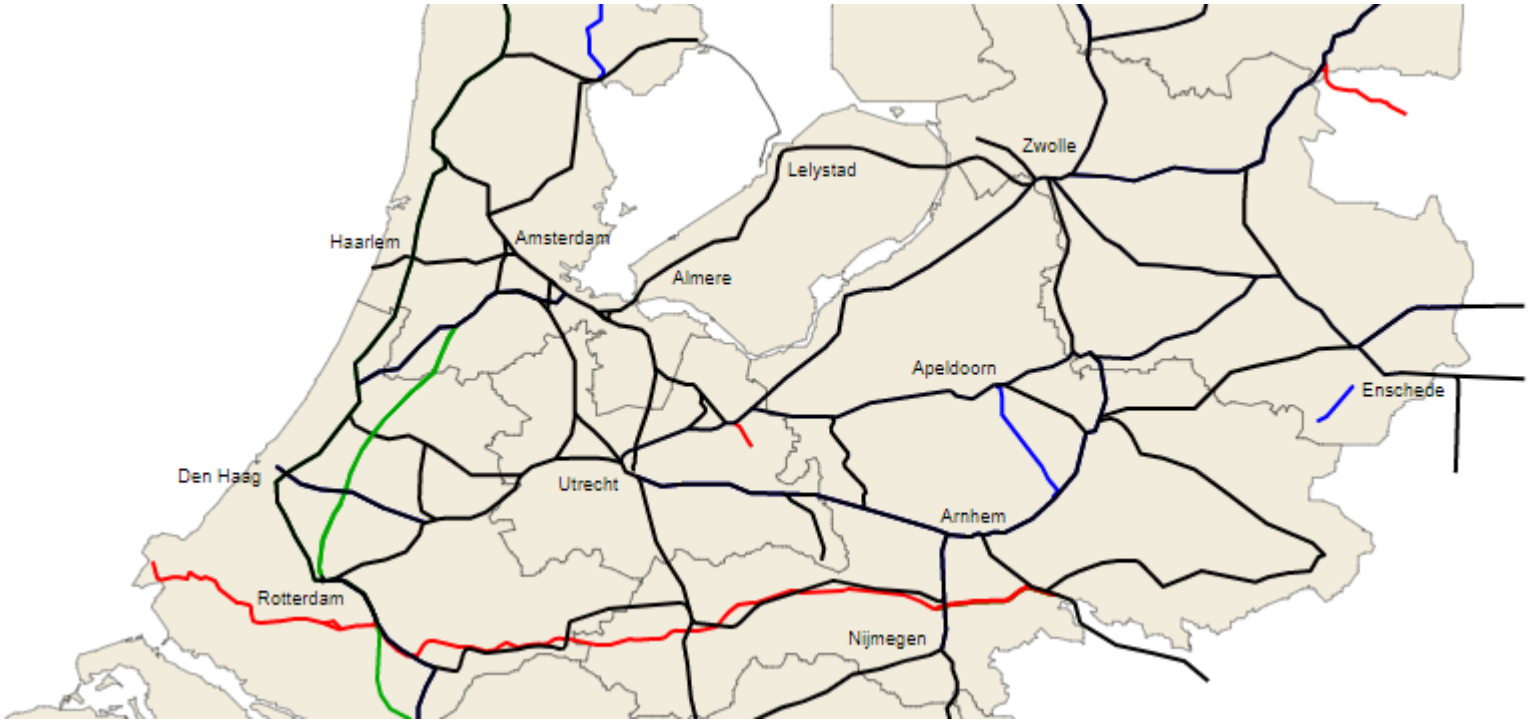
DE INTERFACE IN PYTHON

- Is de intermediair (*interface*) tussen de besturingssoftware en de specifieke centrale.
- Kent alle details van de betreffende centrale.
- Realiseert minimaal de volgende functies:

1. Aanmelden bij de centrale
2. Trein laten rijden (snelheid)
3. Trein laten stoppen (direct, langzaam)
4. Trein richting geven (vooruit, achteruit)
5. Trein functies (licht, geluid, enz.)
6. Wissels, seinen en verlichting schakelen
7. Bezetmelders (*feedback*)
8. Reserveren en vrijgeven

```
1. connect()
2. speed(trein, snelheid)
3. stop(trein)
   halt(trein)
4. forward(trein)
   backward(trein)
5. lights_on(trein)
   lights_off(trein)
6. red(wissel)
   green(wissel)
7. wait_full(feedback)
   wait_empty(feedback)
8.  claim(blok, trein)
    release(blok)
    is_reserved(blok)
```

ROUTES RIJDEN

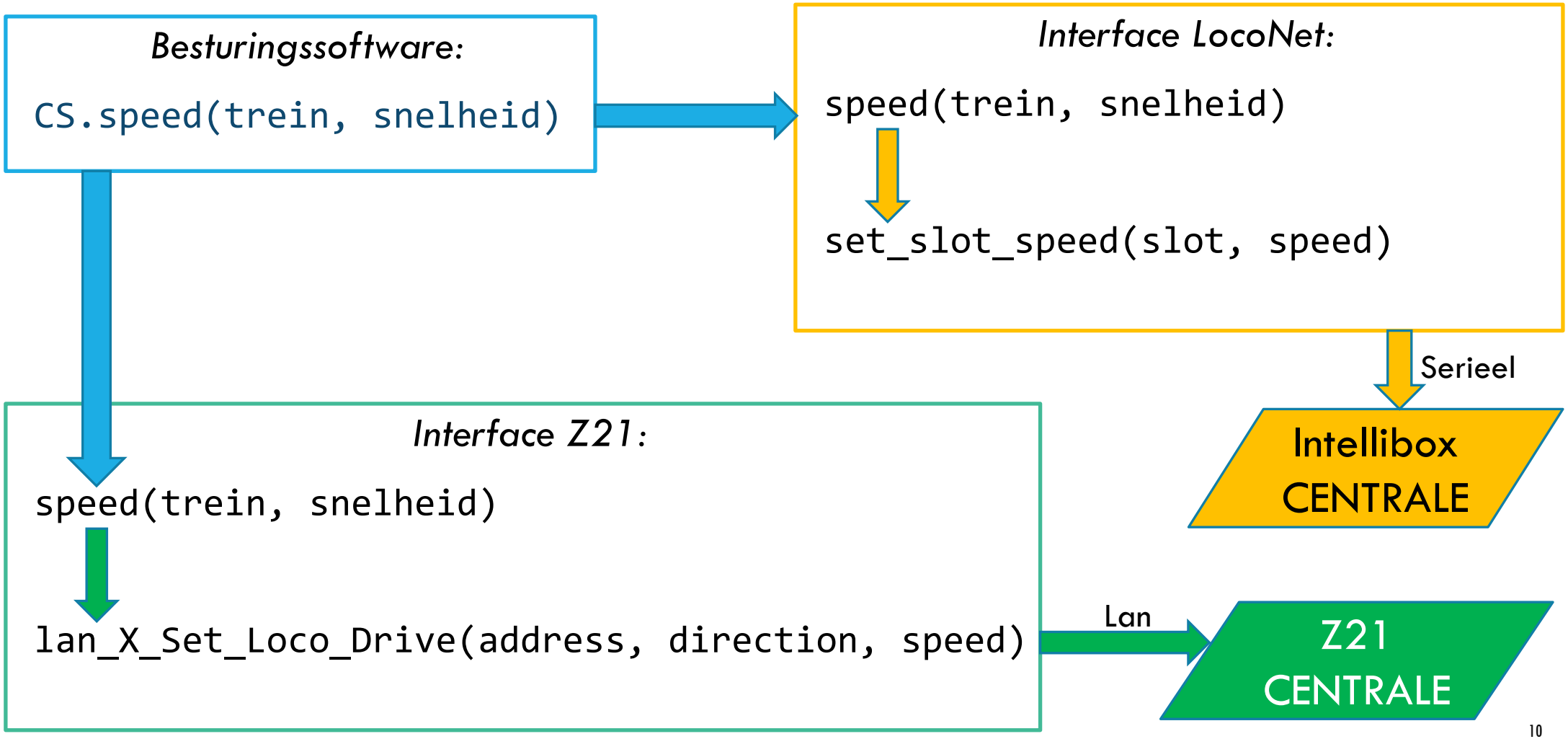


DE EERSTE ROUTE MET EEN TREIN IN PYTHON

```
# CS = CommandStation (is Interface)


for snelheid in [0,20,40,60,80,100,120]:
    CS.speed(trein, snelheid)
    time.sleep(2)


CS.stop(trein)


time.sleep(4)
```

DE EERSTE ROUTE MET EEN TREIN IN PYTHON

INTERFACE FUNCTIE: SPEED



INTERFACE: LOCONET SET SPEED VOORBEELD

4 byte message opcodes format for SET SLOT SPEED is:
<A0><SLOT#><SPEED><CHECKSUM>

```
def set_slot_speed(slot, speed):
    buf = bytearray(4)
    buf[0] = 0xA0
    buf[1] = (slot.to_bytes(1, byteorder='little'))[0]
    buf[2] = (speed.to_bytes(1, byteorder='little'))[0]
    buf[3] = buf[0] ^ buf[1] ^ buf[2] ^ 0xFF
    write_message(buf)
```

INTERFACE Z21: SET SPEED VOORBEELD

Z21 LAN Protocol Specification

4.2 LAN_X_SET_LOCO_DRIVE

Change the speed and direction of a locomotive.

Request to Z21:

DataLen		Header		Data					
				X-Header	DB0	DB1	DB2	DB3	XOR-Byte
0x0A	0x00	**0x40**	0x00	**0xE4**	**0x1S**	**Adr_MSB**	**Adr_LSB**	**RVVVVVVV**	XOR-Byte

Note: loco address = (**Adr_MSB** & 0x3F) << 8 + **Adr_LSB**
For locomotive addresses ≥ 128, the two highest bits in DB1 must be set to 1:
DB1 = (**0xC0** | **Adr_MSB**). For locomotive addresses < 128, these two highest bits have no meaning.

0x1**S** Number of speed steps, depending on the rail format set
S=0: DCC 14 speed steps, or MMI with 14 speed steps and F0
S=2: DCC 28 speed steps, or MMII with 14 real speed steps and F0-F4
S=3: DCC 128 speed steps (aka "126 speed steps" when not counting the stops), or MMII with 28 real speed steps (using light-trit) and F0-F4

RVVVVVVV R ... Direction: 1=forward
V ... Speed: depending on the speed steps S. Coding see below.
If the format MM is configured for the locomotive, the conversion of the given DCC speed stage into the real MM speed stage takes place automatically in the Z21.

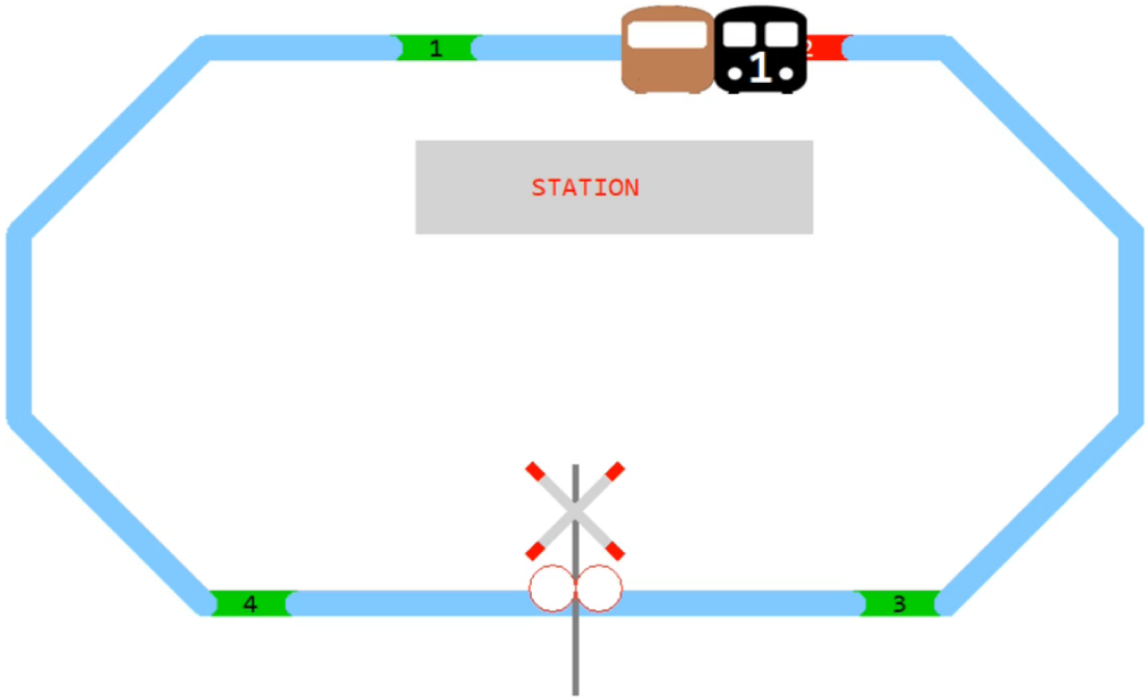
```
def lan_X_Set_Loco_Drive(address, direction, speed): # Z21 SET SPEED
    buf = bytearray(10)
    buf[0] = 0x0A        # dataLen
    buf[1] = 0x00
    buf[2] = 0x40        # header
    buf[3] = 0x00
    buf[4] = 0xE4        # xHeader
    buf[5] = 0x13        # db0
    buf[6] = 0x00        # db1
    buf[7] = (address.to_bytes(1, byteorder='little'))[0]   #db2
    buf[8] = (speed.to_bytes(1, byteorder='little'))[0]     #db3

    if direction == 'forward':
        buf[8] |= 0b10000000  # set bit
    else:
        buf[8] &= ~0b10000000  # clear bit

    buf[9] = buf[4] ^ buf[5] ^ buf[6] ^ buf[7] ^ buf[8]   #xor

    sendCommand(buf)
```

BERICHTENVERKEER INTERFACE ⟷ CENTRALE

```
:54:a0:01:0a:54:a0:01:0f:51:a0:01:0f:51:a0:01:14:4a:a0:01:14:4a:a0:01:19:47:a0:0
1:19:47:a0:01:1e:40:a0:01:1e:40:a0:01:23:7d:a0:01:23:7d:a0:01:28:76:a0:01:28:76:
a0:01:2d:73:a0:01:2d:73:a0:01:32:6c:a0:01:32:6c:a0:01:37:69:a0:01:37:69:a0:01:3c
:62:a0:01:3c:62:a0:01:41:1f:a0:01:41:1f:a0:01:46:18:a0:01:46:18:a0:01:4b:15:a0:0
1:4b:15:a0:01:50:0e:a0:01:50:0e:a0:01:55:0b:a0:01:55:0b:a0:01:5a:04:a0:01:5a:04:
a0:01:5f:01:a0:01:5f:01:a0:01:64:3a:a0:01:64:3a:a0:01:69:37:a0:01:69:37:a0:01:6e
:30:a0:01:6e:30:a0:01:73:2d:a0:01:73:2d:
```

DE BEZETMELDER (FEEDBACK)

PYTHON EN DE BEZETMELDER (FEEDBACK)



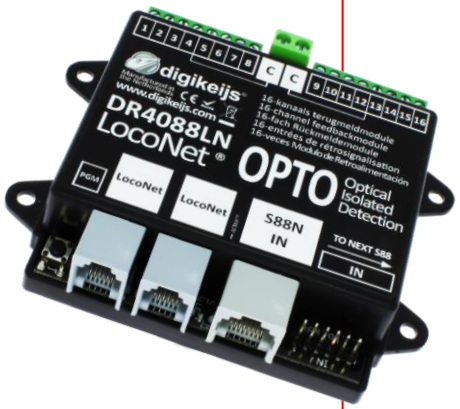
Een trein met een wagonnetje rijdt een aantal rondjes.

Bij het station wordt enige tijd gestopt.

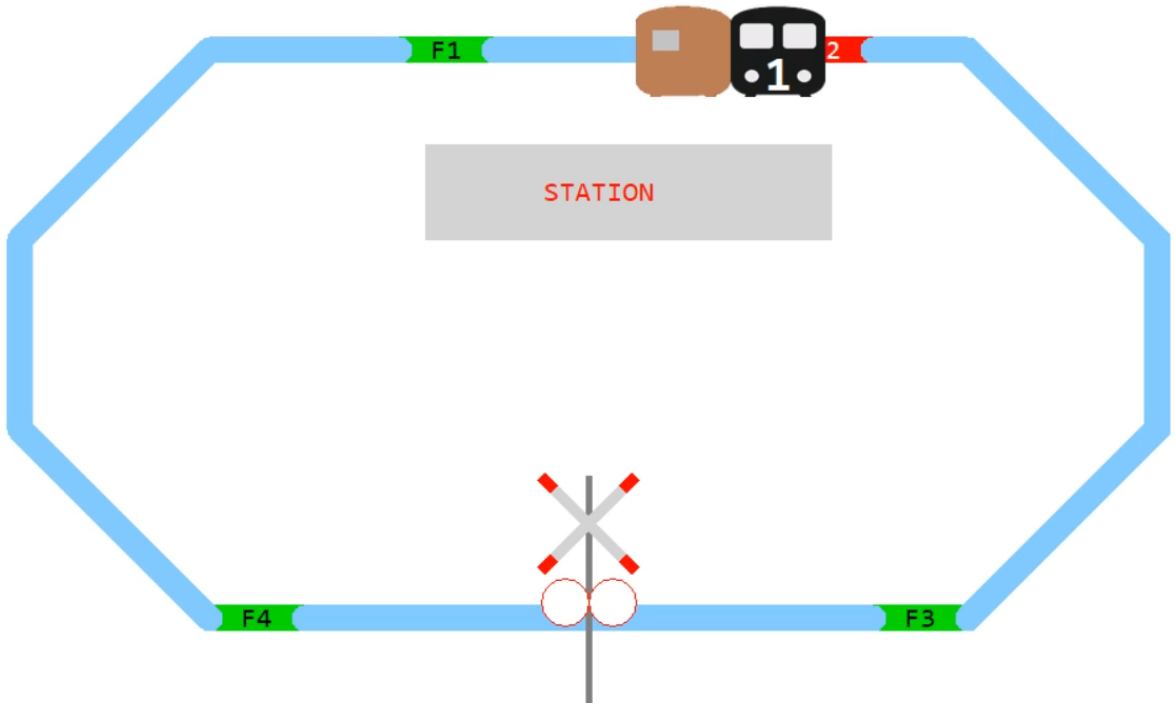
De spoorwegovergang knippert als er een trein aankomt.

Dankzij de bezetmelders (4 stuks)

En dankzij …. Python ;-)



DE BEZETMELDER IN ACTIE




```
CS.speed(trein, 60)
CS.wait_full('F3')
CS.crossing_on()
CS.speed(trein, 100)
CS.wait_full('F4')
CS.crossing_off()
CS.wait_full('F1')
CS.speed(trein, 20)
CS.wait_full('F2')
CS.stop(trein)
time.sleep(2)
```

CS.wait_full() → Proces (*thread*) wacht, maar trein rijdt gewoon door

DE BEZETMELDER IN DE INTERFACE

- Bij status wijziging van een bezetmelder (dat is verandering van bezet naar vrij v.v.), stuurt de Centrale een (LocoNet-)bericht.
- De Interface registreert de actuele toestand van de bezetmelders.
- De Interface heeft twee bezetmelder functies: **wait_full** en **wait_empty**.
- Deze twee functies blokkeren het proces tot de gewenste conditie is bereikt. (De trein wordt niet geblokkeerd, die tuft vrolijk door…)
- De Interface heeft (daarom) voor de veiligheid een tijdslimiet. Wanneer het wachten op de conditie langer dan 15 seconden duurt, stuurt de Interface een **power-off** bericht naar de Centrale.
- Intern wordt gebruik gemaakt van de zogenaamde Python *Event*. Zie details hierover in https://docs.python.org/3/library/threading.html

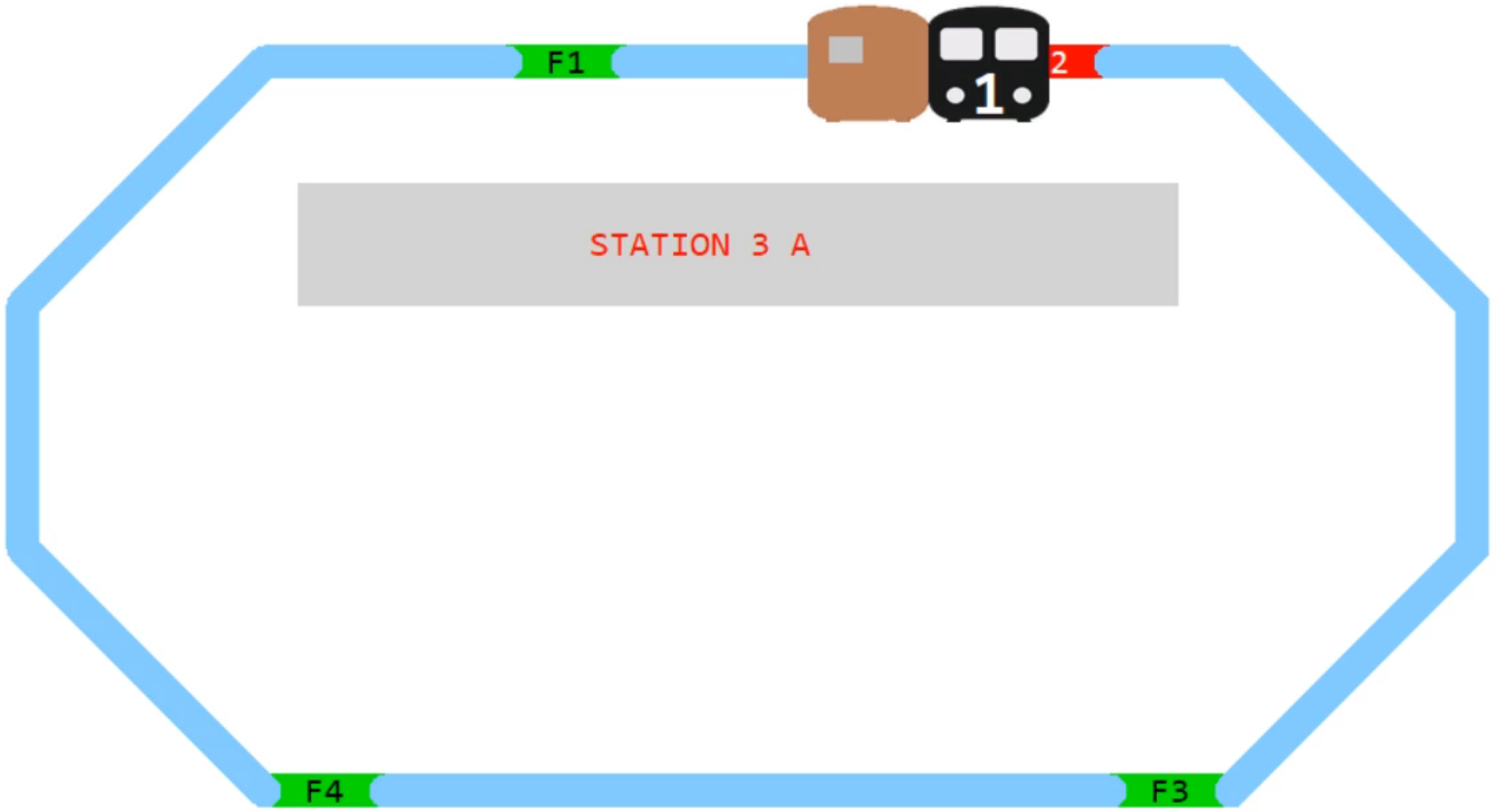
RESERVEREN EN VRIJGEVEN

VOORBEELD RESERVEREN/VRIJGEVEN: 1 TREIN

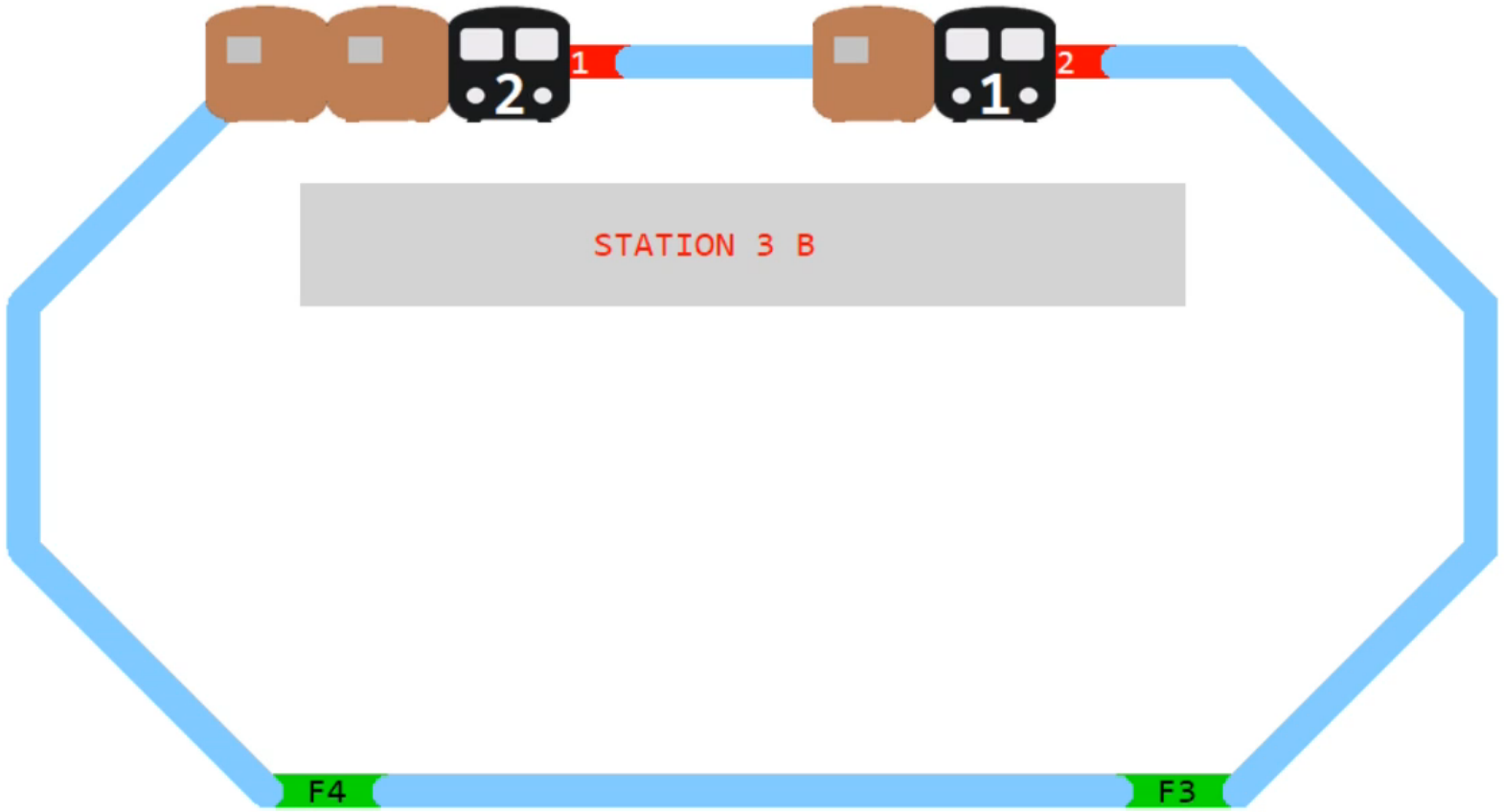
Steeds wordt een volgend blok gereserveerd, dan daar heen gereden en bij aankomst het vorige blok vrijgegeven.

Een trein kan meerdere blokken reserveren. (Vaak afhankelijk van de lengte van de trein)

Een trein heeft alleen toegang tot zijn gereserveerde blok(ken).



VOORBEELD RESERVEREN/VRIJGEVEN: 2 TREINEN



RESERVEREN (VAN EEN "BLOK")

```
CS.claim("B2", trein)
CS.speed(trein, 60)
CS.wait_full("F2")
CS.stop(trein)
CS.release("B1")

CS.claim("B3", trein)
CS.speed(trein, 70)
CS.wait_full("F3")
CS.stop(trein)
CS.release("B2")
```

Schematisch, trein rijdt route blokken 1, 2, 3,

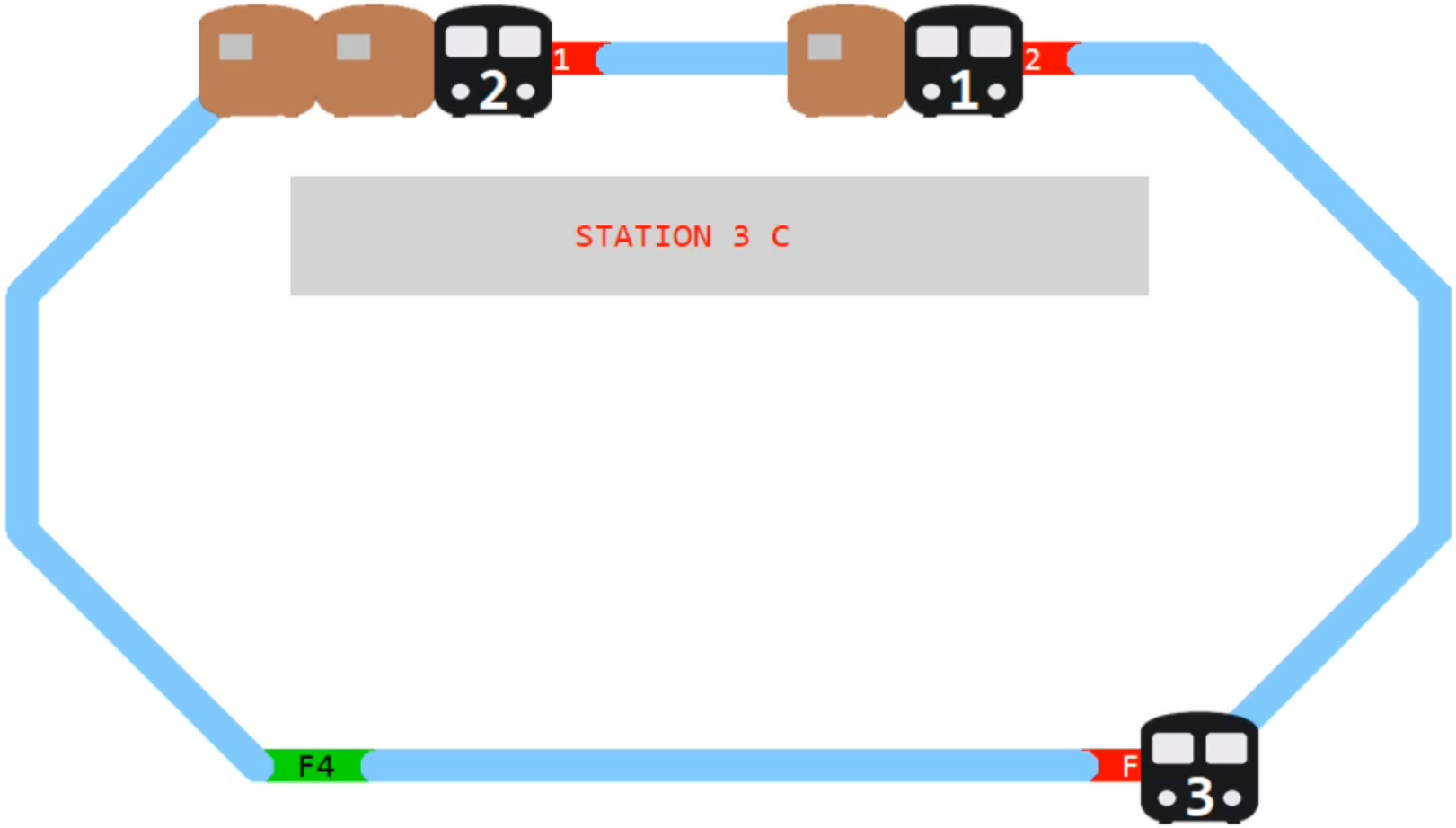
Trein staat stil in het gereserveerde blok 1

Trein reserveert blok 2
Reservering gelukt: Trein gaat rijden
Trein komt aan in blok 2 (bezetmelder)
Trein stopt
Trein geeft vorig blok 1 vrij (vorig blok)

Trein reserveert blok 3
Reservering gelukt: Trein gaat rijden
Trein komt aan in blok 3 (bezetmelder)
Trein stopt
Trein geeft vorig blok 2 vrij (vorig blok)

Enzovoort

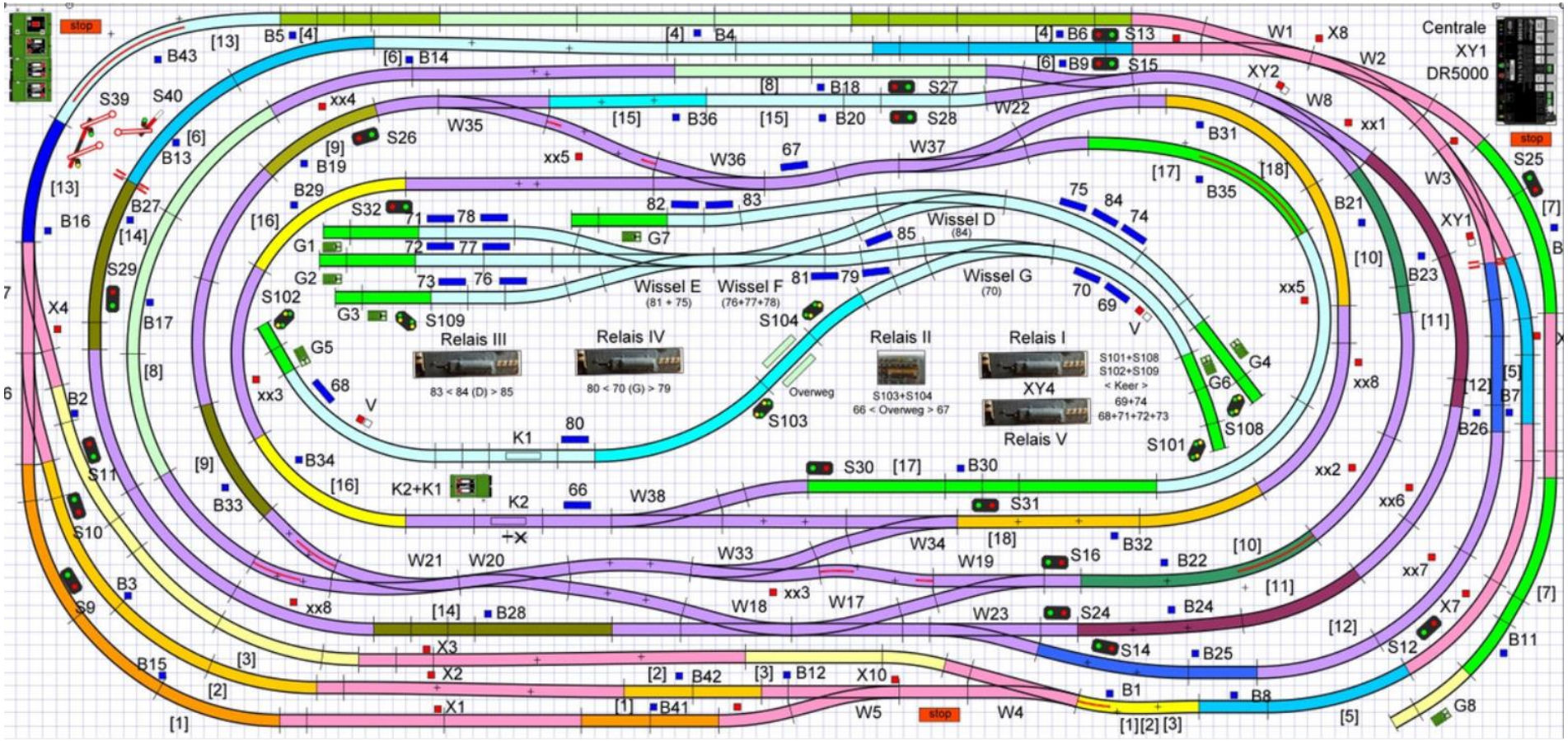
VOORBEELD RESERVEREN/VRIJGEVEN: 3 TREINEN



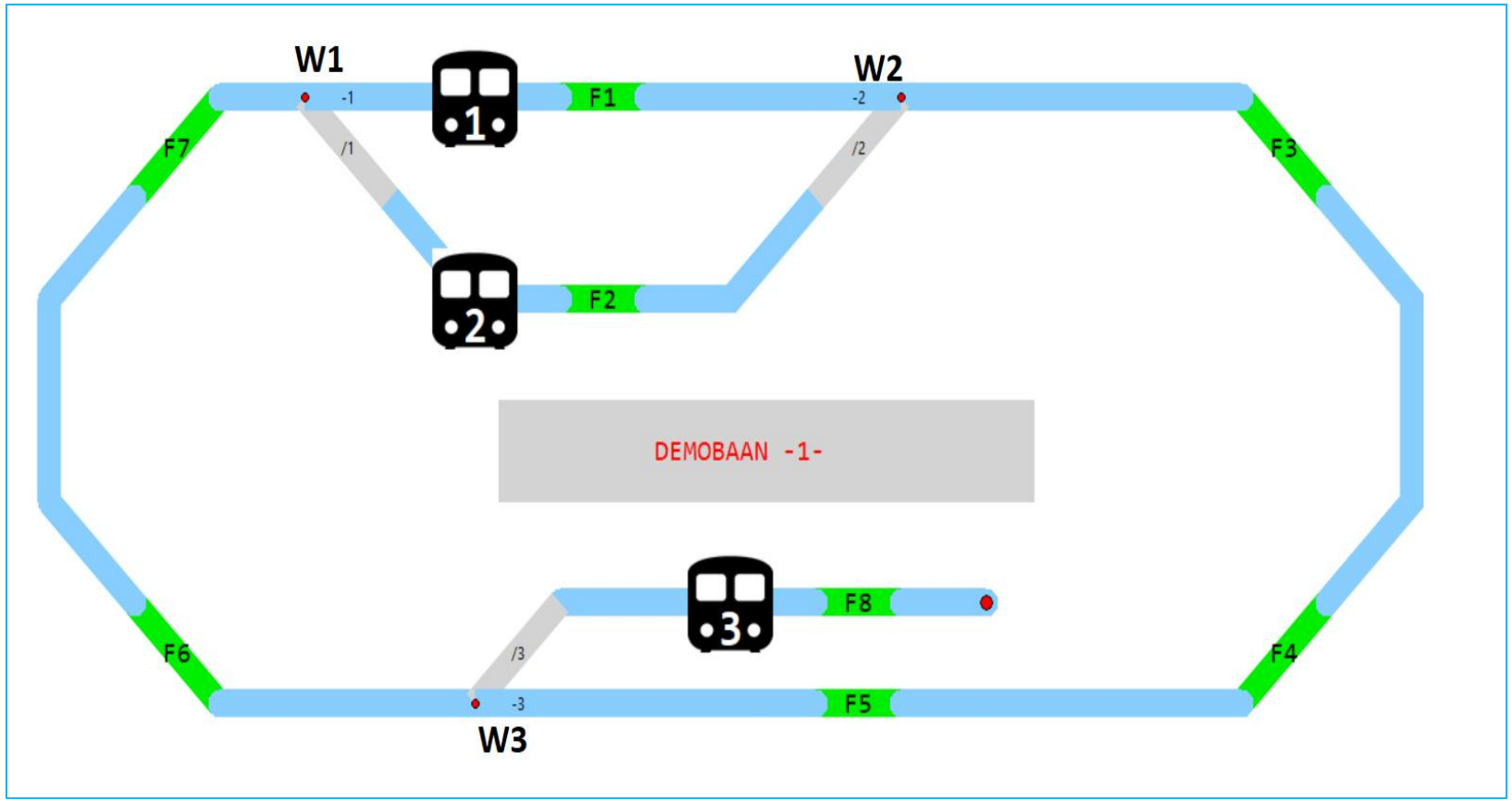
RESERVEREN EN VRIJGEVEN IN DE INTERFACE

- De Interface houdt van alle blokken bij of deze *gereserveerd* of *vrij* zijn.
- Wanneer een trein een blok wil reserveren (via de aanroep van `CS.claim(blok, trein)`) dan **stopt** het proces indien het blok reeds door een ander *gereserveerd* is. Het proces **gaat pas verder** als de reservering voor deze trein is gelukt.
- Reserveren/Vrijgeven gebeurt in de besturingssoftware, er is hierover géén communicatie met de Centrale.
- Deze code is voor alle soorten interfaces (LocoNet, Z21, …) dezelfde.
- Intern wordt gebruik gemaakt van de zogenaamde Python Lock. Zie details hierover in https://docs.python.org/3/library/threading.html

ROUTES RIJDEN OP DEMOBAAN

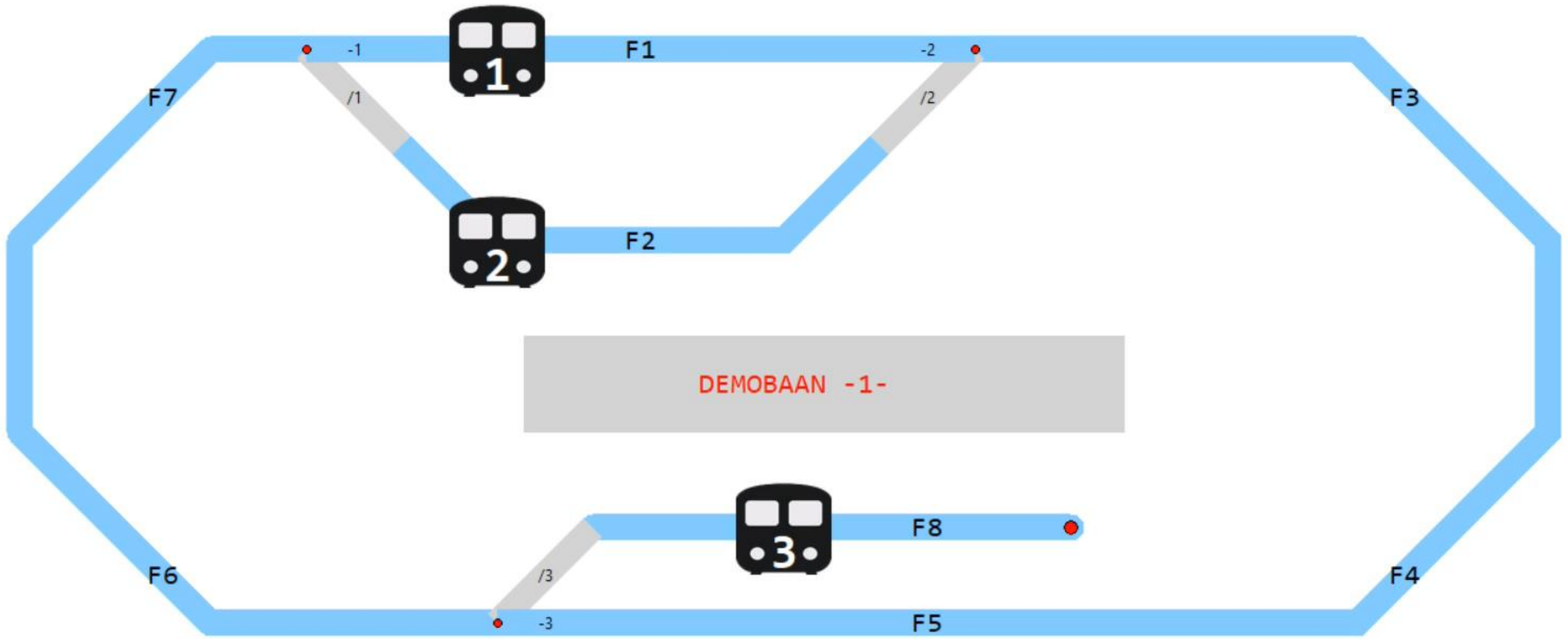


DEMOBAAN 1 ≈ FLYERBAAN



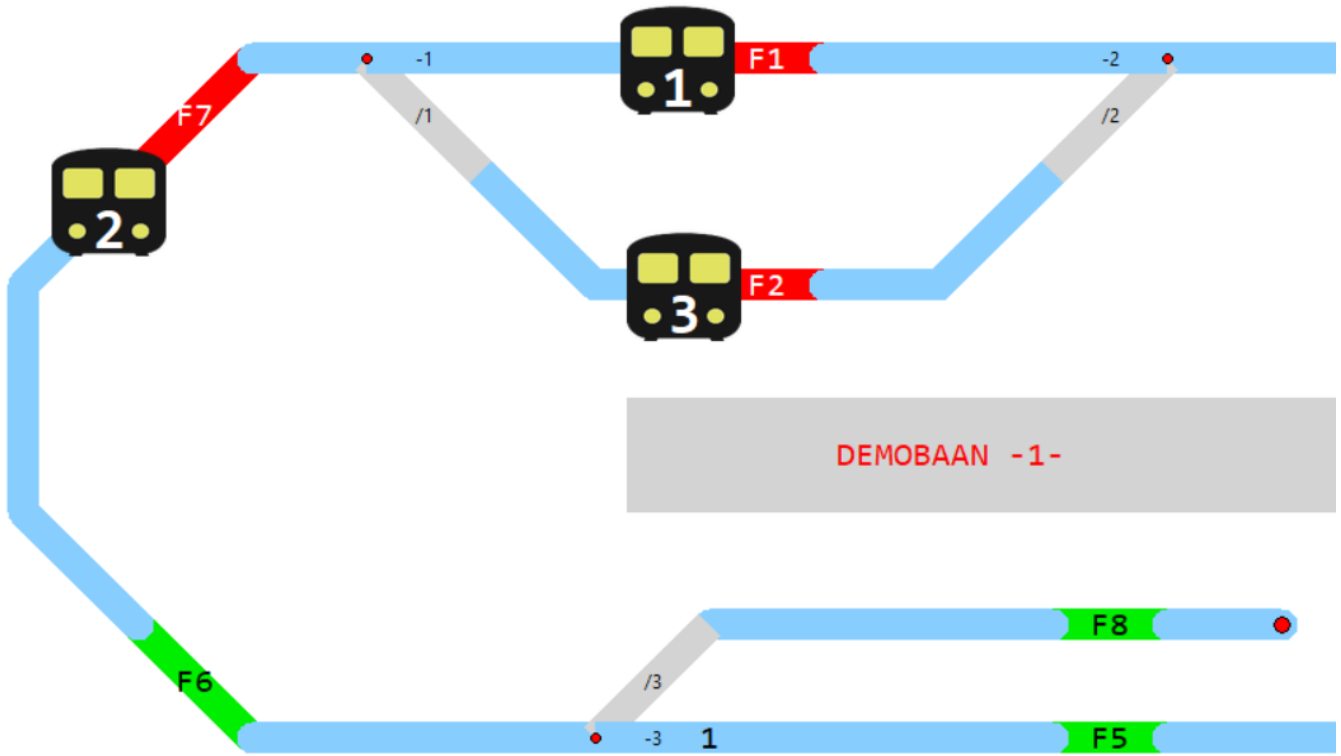
- Let op: in deze voorbeelden hebben alle blokken precies één bezetmelder
- Trein 1 rijdt rondjes via blok 1, 3, ..
- Trein 2 rijdt rondjes via blok 2, 3, .., tenzij blok 2 bezet is, dan blok 1
- Trein 3 pendelt tussen blokken 8 en 2 via 6 en 7.

RIJDEN OP DE DEMOBAAN MET 3 TREINEN

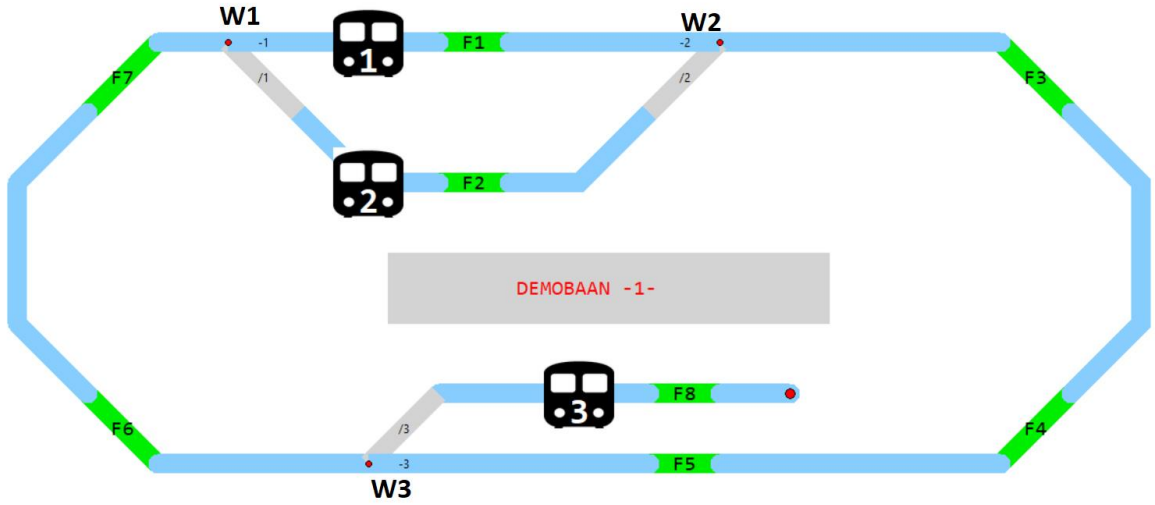


STUKJE PYTHON CODE VAN TREIN 2

```
# Trein (2) staat op B7

if CS.is_reserved('B2'):
    CS.claim('B1', trein)
    CS.red('W1')
    CS.speed(trein, 60)
    CS.wait_full('F1')
else:
    CS.claim('B2', trein)
    CS.green('W1')
    CS.speed(trein, 40)
    CS.wait_full('F2')

CS.stop(trein)
```



ROUTE PENDELEN

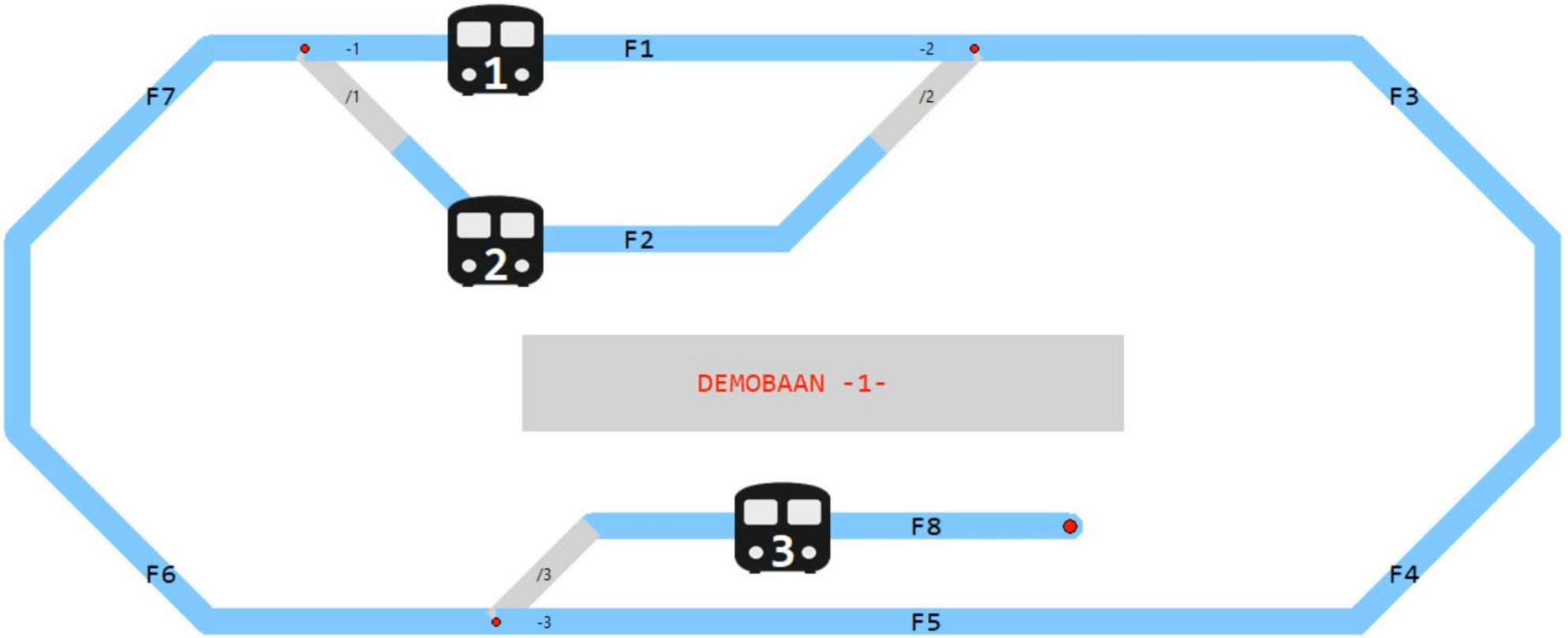


De trein op blok B8 pendelt via B6 en B7 naar blok B2 en na een korte pauze weer terug naar blok B8.

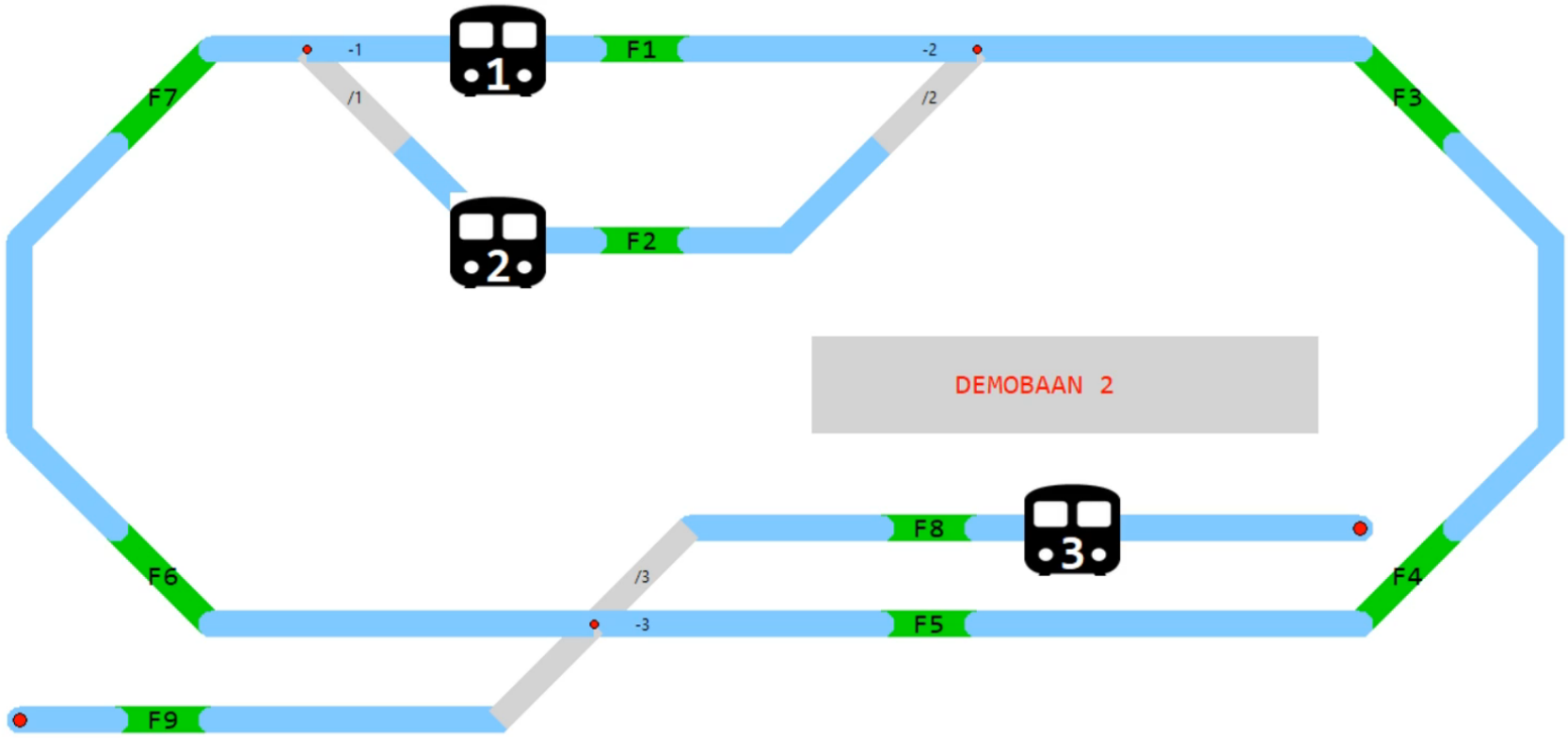
De terugweg gaat tegen de richting van de andere twee treinen in, dat vereist het reserveren van alle blokken tussen B2 en B8, dat zijn de blokken B7 en B6. De volgorde van reserveren is belangrijk!

Fout is om bij de terugweg eerst B7 en daarna B6 te reserveren. Zie volgende simulatie !

DEMO: FOUT PENDELEN → DEADLOCK (IMPASSE)



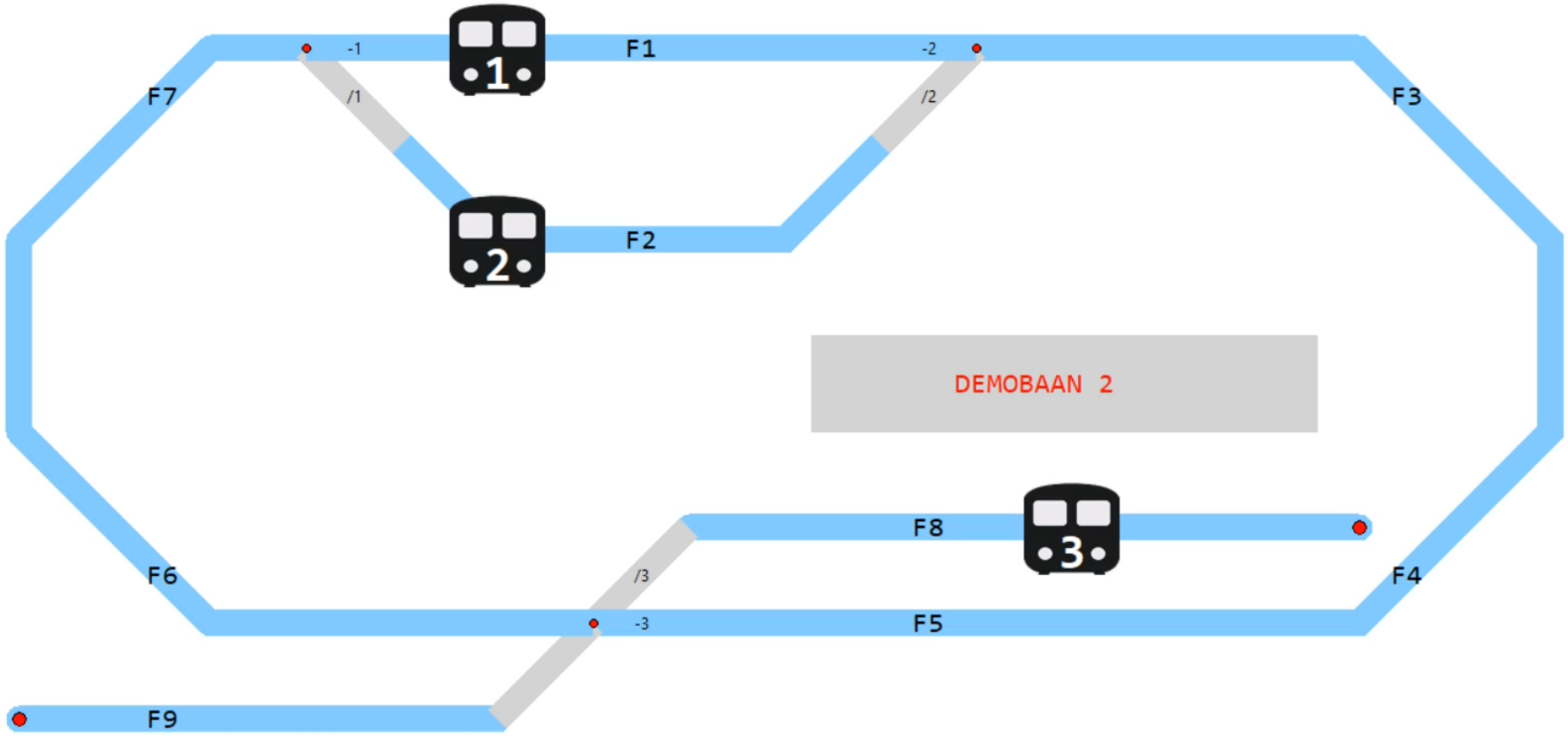
DEMOBAAN 2, NU MET EEN KRUISING



- Let op: in deze voorbeelden hebben alle blokken precies één bezetmelder
- Trein 1 rijdt rondjes via blok 1, 3, ..
- Trein 2 rijdt rondjes via blok 2, 3, .., tenzij blok 2 bezet is, dan blok 1
- **Trein 3 pendelt nu tussen blokken 8 en 9.**

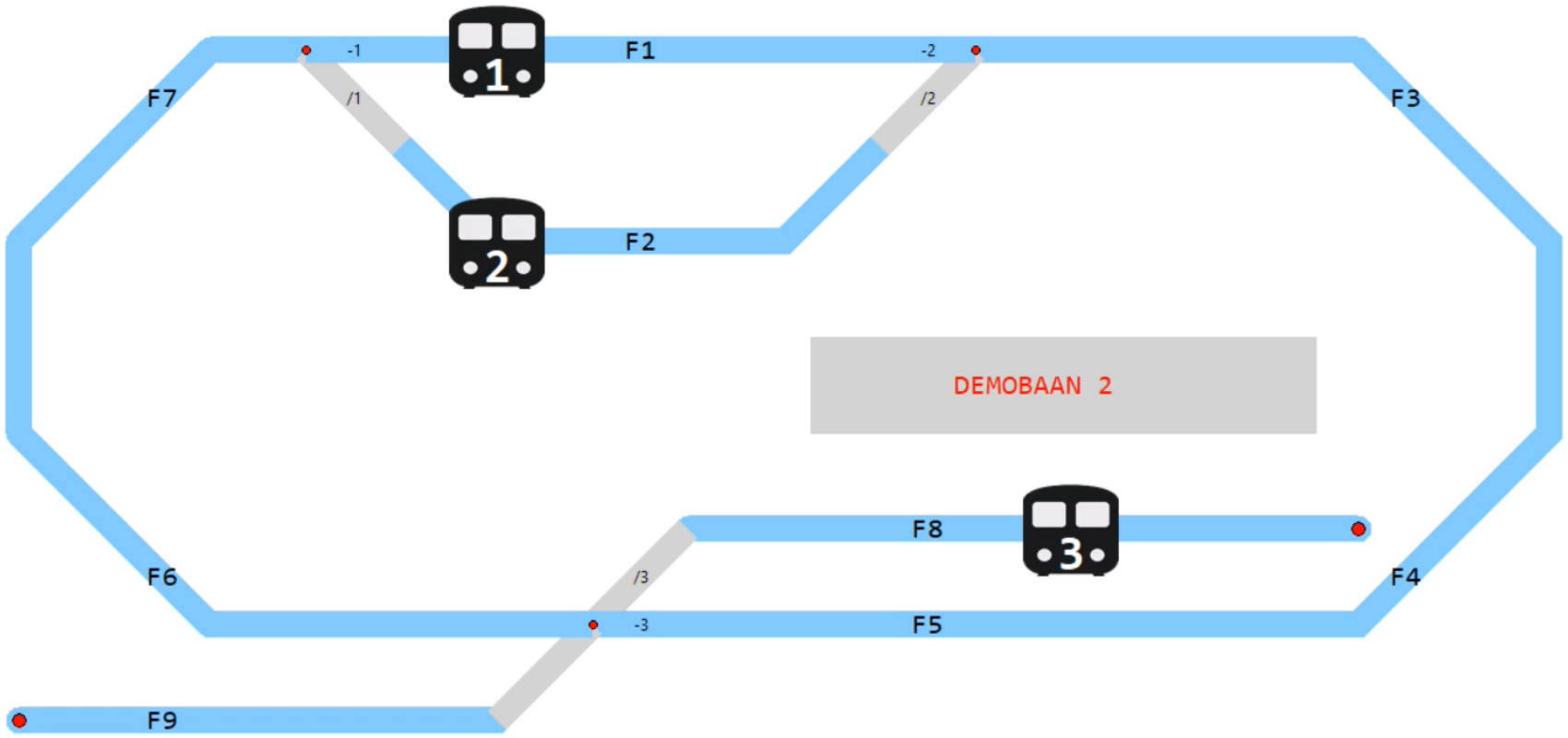
DEMOBAAN 2, RIJDEN MET DRIE TREINEN

GAAT HET GOED??

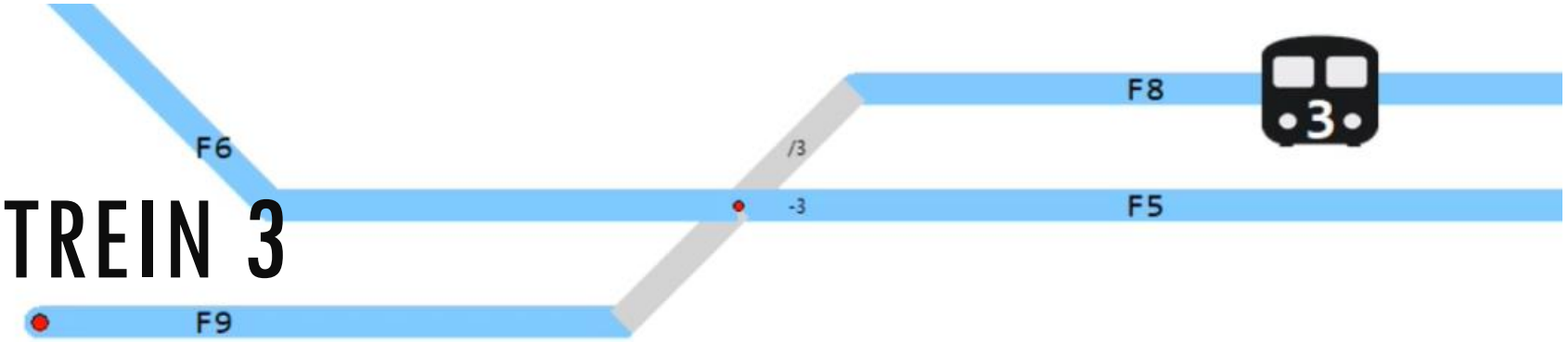


DEMOBAAN 2, MET KRUISING RESERVEREN

GAAT HET FOUT ??

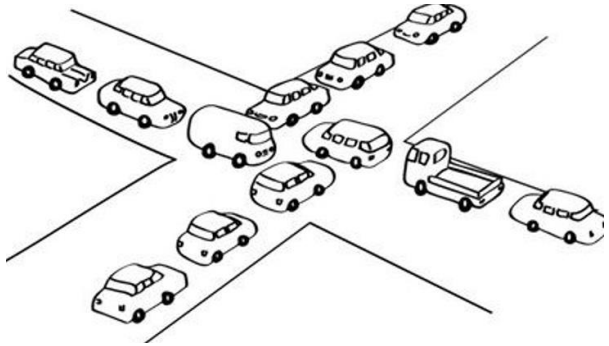


PYTHON CODE TREIN 3



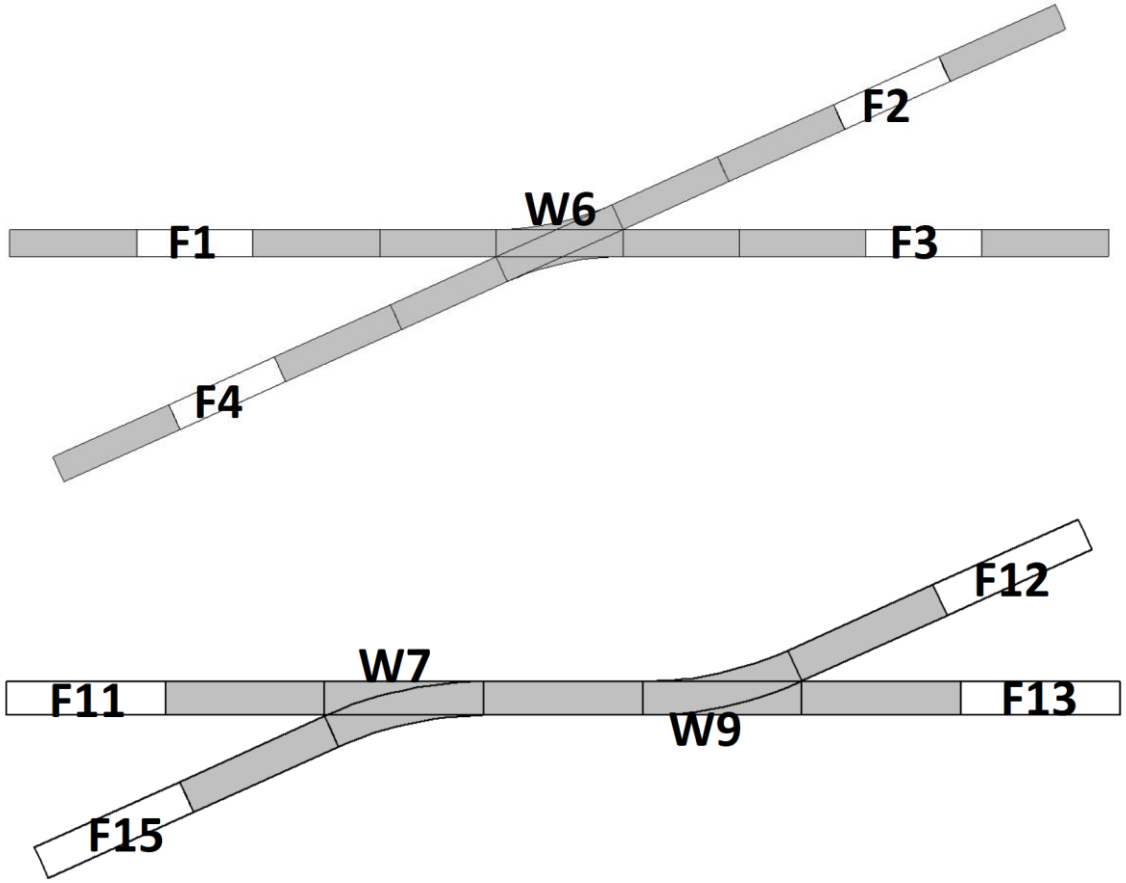

```
# Vooraf: blok 8 gereserveerd
# Heen: van blok 8 naar 9

CS.forward(loc)
CS.claim('B9', loc)
CS.claim('W3', loc)
CS.speed(loc, 60)

CS.wait_full('F9)
CS.release('W3')
CS.release('B8')
CS.stop(loc)
time.sleep(2)
```

```
# Terug: van blok 9 naar 8

CS.backward(loc)
CS.claim('B8', loc)
CS.claim('W3', loc)
CS.speed(loc, 60)

CS.wait_full('F8')
CS.release('W3')
CS.release('B9')
CS.stop(loc)
time.sleep(2)
```



DEADLOCKS

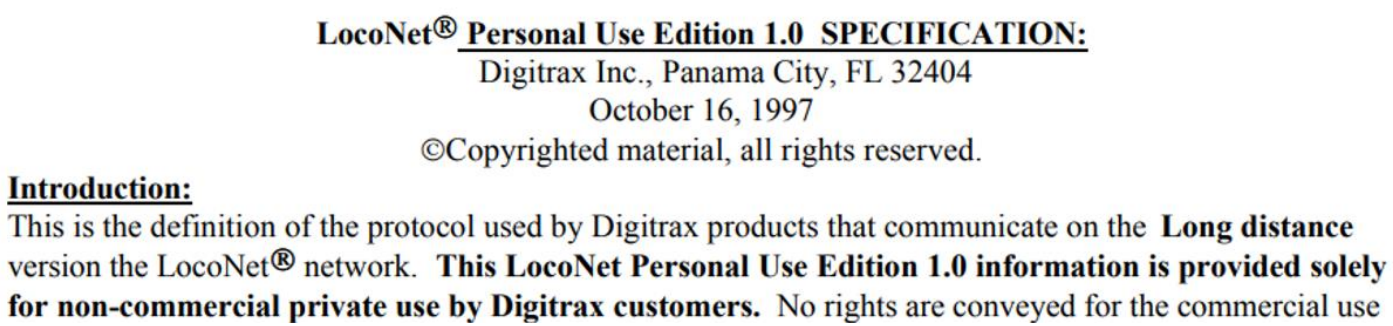
- Wees bedacht op twee richtingen spoor
- Reserveer meerdere opeenvolgende blokken indien nodig
- Reserveer de blokken voor elke trein in steeds dezelfde <u>volgorde</u>
- Let ook op kruisingen of opeenvolgende <u>wissels:</u> reserveren!



DE INTERFACE

INTERFACE DOCUMENTATIE

LocoNet



LocoNet® Personal Use Edition 1.0 SPECIFICATION:
Digitrax Inc., Panama City, FL 32404
October 16, 1997

Introduction:
This is the definition of the protocol used by Digitrax products that communicate on the **Long distance** version the LocoNet® network. **This LocoNet Personal Use Edition 1.0 information is provided solely for non-commercial private use by Digitrax customers.** No rights are conveyed for the commercial use

https://www.digitrax.com/static/apps/cms/media/documents/loconet/loconetpersonaledition.pdf

Z21

Roco Z21 LAN Protocol Specification, versie 1.12 Engelstalig

Roco Z21 Maintenance Tool User Manual, versie 1.17 Engelstalig

Prima documentatie, *up to date* informatie! En eenvoudig te vinden via *Google* …

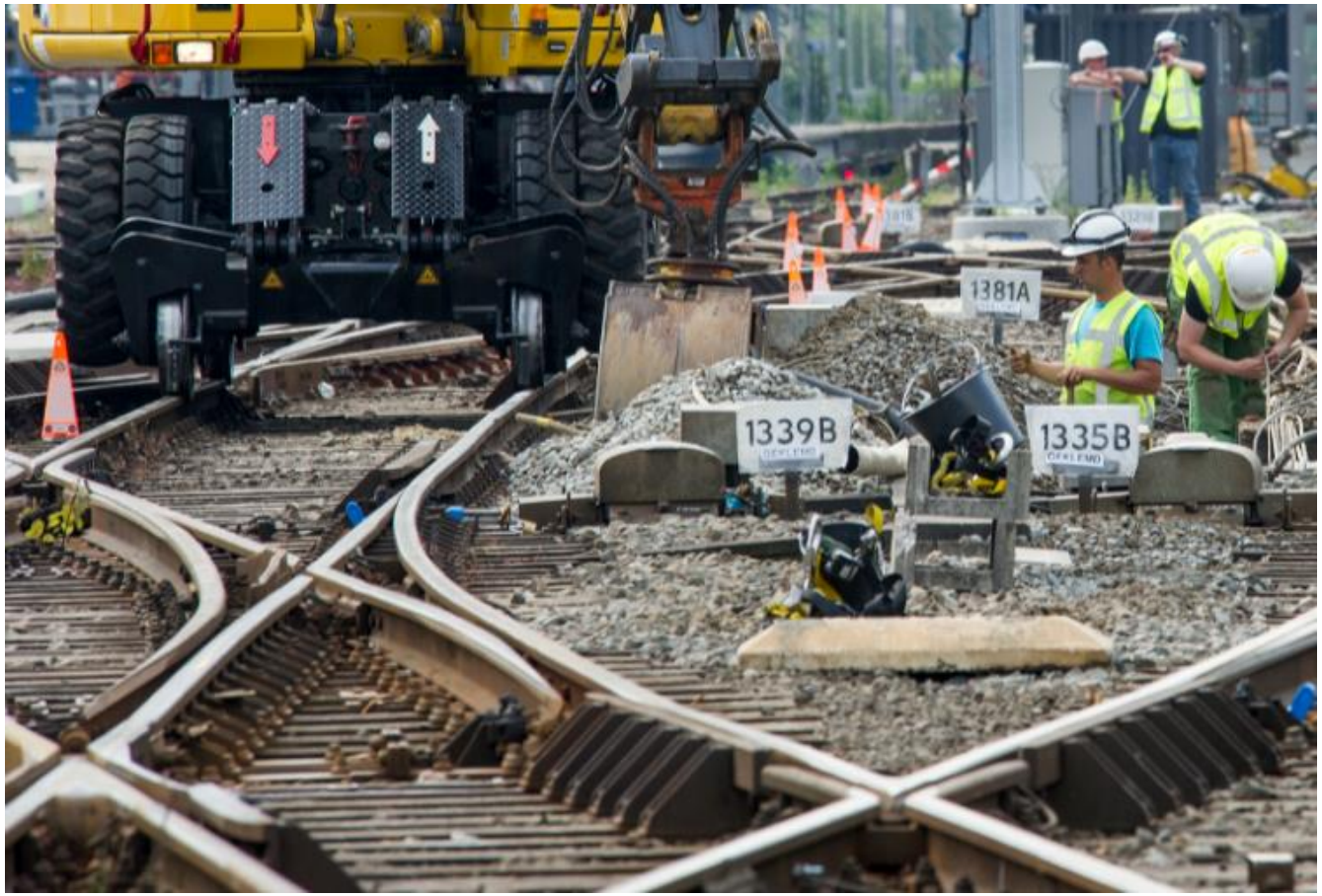
LOCONET INTERFACE: AANMELDEN BIJ INTELLIBOX

```
import serial                              # voor Loconet communicatie
class LocoNetInterface:
    def __init__(self):
        self.ser = serial.Serial()
        self.ser.port = 'COM3'             # kies de juiste port !
                                           # MacOS: /dev/cu.usbserial-0001 ?
        self.ser.baudrate = 115200
        self.ser.timeout = 0
        self.ser.parity = 'N'
        self.ser.stopbits = 1              # standaard voor Intellibox
        self.ser.databits = 8
        self.ser.handshake = 0

    def connect(self):
        self.ser.open()
```

Z21 INTERFACE: AANMELDEN BIJ Z21 LAN

```
import socket                                   # voor Z21 communicatie

class Z21Interface:

    def connect(self):                          # connect to Z21 LAN
       self.soc = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
       self.Z21 = ('192.168.0.111', 21105)

   def sendCommand(self, command):              # send message to Z21
       self.soc.sendto(command, self.Z21)
```

AFSLUITING



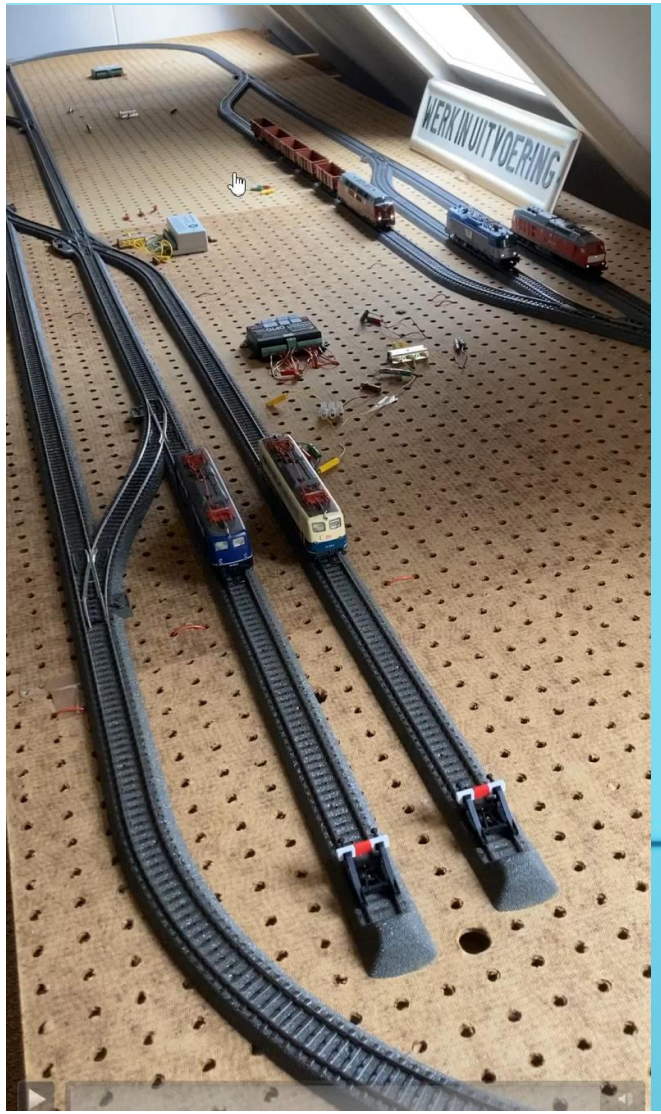
WAAROM ZELF JE SOFTWARE IN PYTHON MAKEN?

- De uitdaging! Het is pittige kost en vereist puzzelen. Niet altijd is makkelijk goede documentatie te vinden.
- Bestaande besturingssoftware is heel breed en generiek, eigen gemaakt is *to the point*.
- Python is een moderne, uitgebreide taal, geschikt voor alle platforms (Windows, Mac, Linux).
- Maar in een andere programmeertaal kan uiteraard ook!

iTrain	Java
Koploper	Delphi/Pascal
RocRail	C/C++
Do It Yourself	C#/Python

MIJN MODELSPOORBAAN

Scenery is nog niet helemaal klaar …



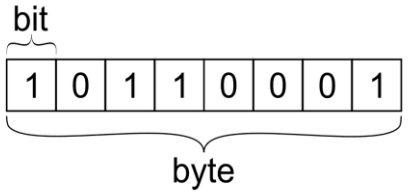
VRAGEN

APPENDIX (ZELFSTUDIE ?)

Inhoud van de volgende pagina's:

- Over bits, bytes en hexadecimaal
- Oefening LocoNet set_slot_speed functie
- Oefening Z21 lan_X_Set_Loco_Drive functie
- Oefening treinen rijden rondjes in Python
- Client GUI voor handbesturing

OVER BITS, BYTES EN HEXADECIMAAL



Bits (8 x)	Byte (hexadecimaal)	Decimaal
0000 0000	00	0
0000 0010	02	2
0000 1001	09	9
0000 1010	0A	10
0000 1111	0F	15
0001 1001	19	25
1010 0000	A0	160

Een **bit** kent alleen de cijfers 0 en 1

Een **byte** bestaat uit 8 bits

Een **hexadecimaal** getal kent de cijfers 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

En een **decimaal** getal bestaat uit de cijfers 0 t/m 9

EEN KLEINE "OEFENING" (VOOR DE LIEFHEBBER)

Wat is de byte-string van de opdracht **set_slot_speed(1, 25)** ?

Antwoord:

Dat zijn de 4 bytes (in hexadecimaal): A0, 01, 19 en 47

```
A0 =  1010 0000
01 =  0000 0001
19 =  0001 1001
47 =  0100 0111   ← checksum (de LocoNet manier)
xor:  1111 1111
```

NOG EEN KLEINE "OEFENING" (VOOR DE LIEFHEBBER)

Wat is de byte-string van de opdracht: **Ian_X_Set_Loco_Drive(1, 'forward', 25)?**

Antwoord:

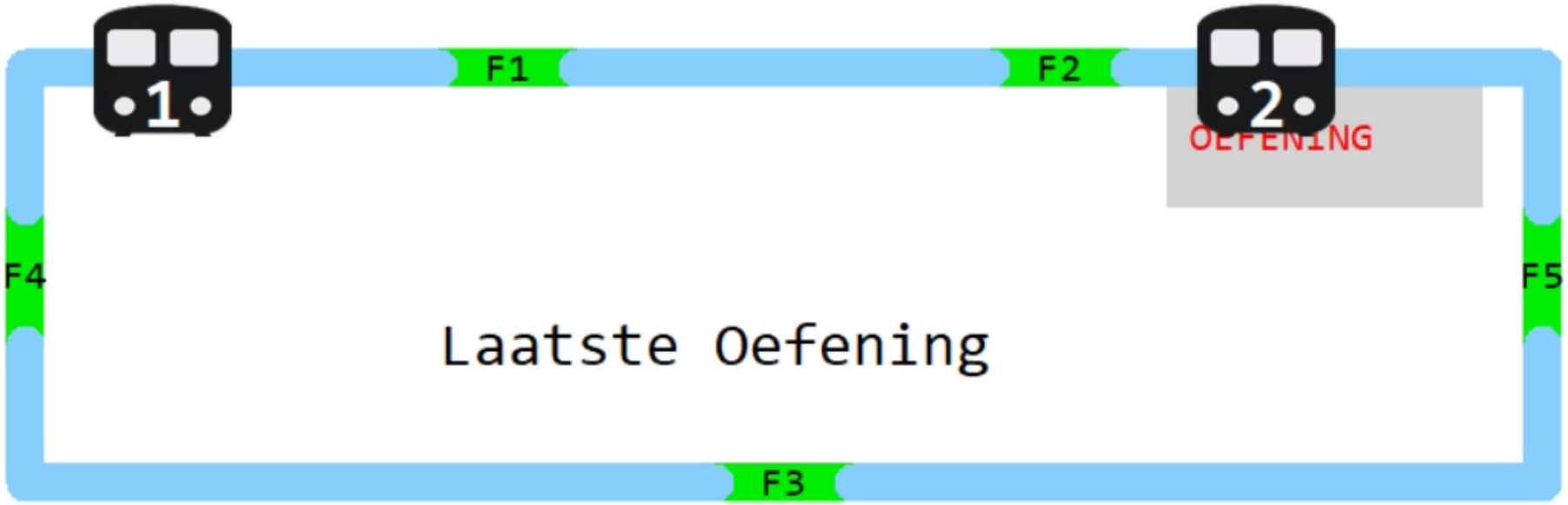
Dat zijn de 10 bytes (in hexadecimaal): 0A, 00, 40, 00, E4, 13, 00, 81, 19 en 6F

```
E4 =  1110 0100
13 =  0001 0011
00 =  0000 0000
81 =  1000 0001
19 =  0001 1001
6F =  0110 1111  ← xor checksum (de Z21 manier)
```

"OEFENING" TREINEN RIJDEN "RONDJES"

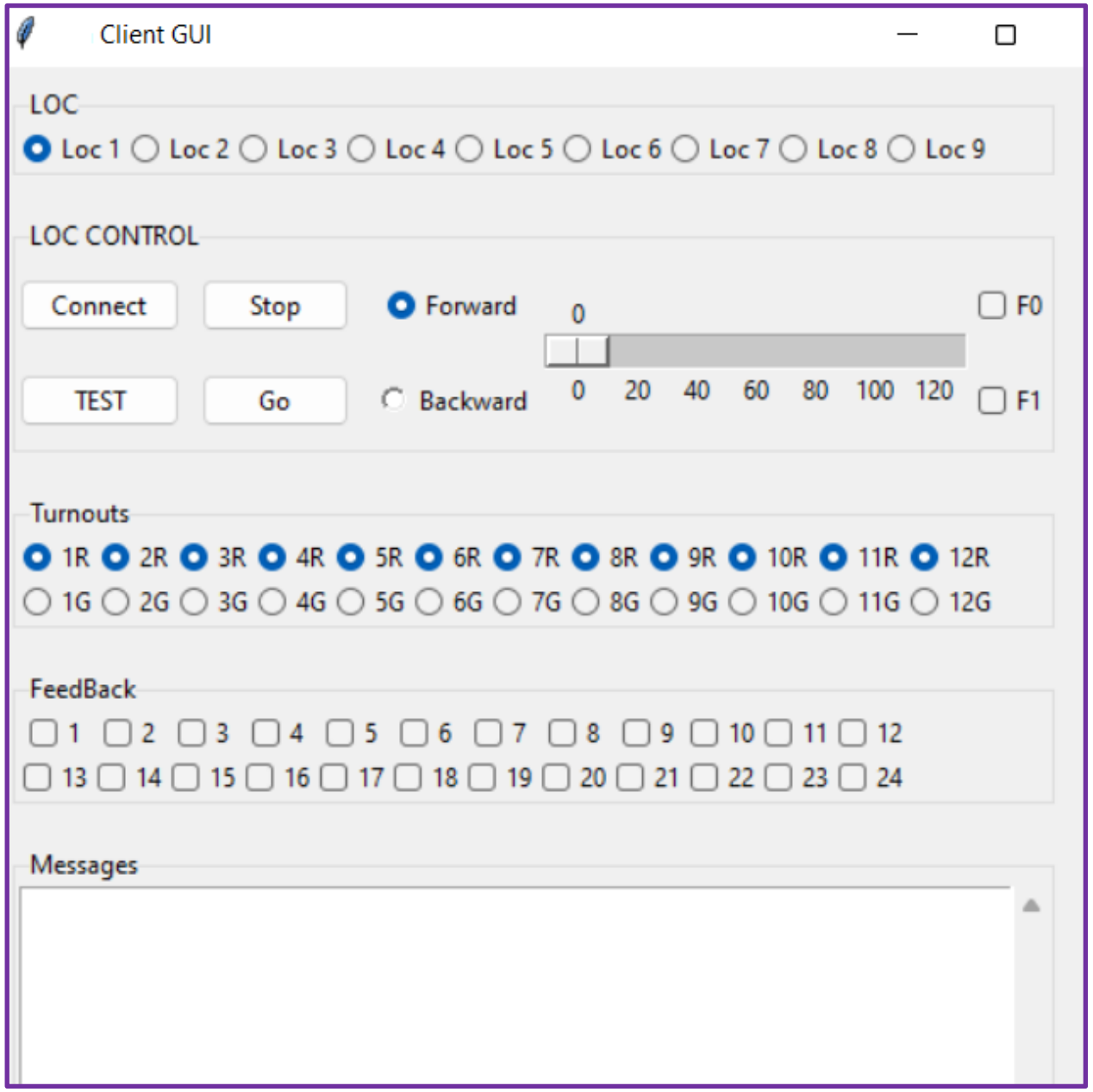
(VOOR DE PYTHON LIEFHEBBER)

Let op: de treinen stoppen bij het station, direct **na** het passeren van bezetmelder F2.



CLIENT GUI IN PYTHON

Altijd handig om te maken en te hebben: een TKInter GUI om de locomotief en wissels en dergelijke handmatig te besturen.



GRAAG TOT ZIENS (BIJ EEN WORKSHOP?)