PiRail

A raspberry pi controlled model railway

Contents

[General Overview 2](#_Toc506196246)

[UDP Z21 2](#_Toc506196247)

[WebSocket Protocol 2](#_Toc506196248)

[Admin Controls (0x80 Flag) 2](#_Toc506196249)

[Train (0x40 Flag) 3](#_Toc506196250)

[Track (0x20 Flag) 4](#_Toc506196251)

[General (0x10 Flag) 5](#_Toc506196252)

[Circuit of RailNet 6](#_Toc506196253)

[Protocol of RailNet 7](#_Toc506196254)

[General 7](#_Toc506196255)

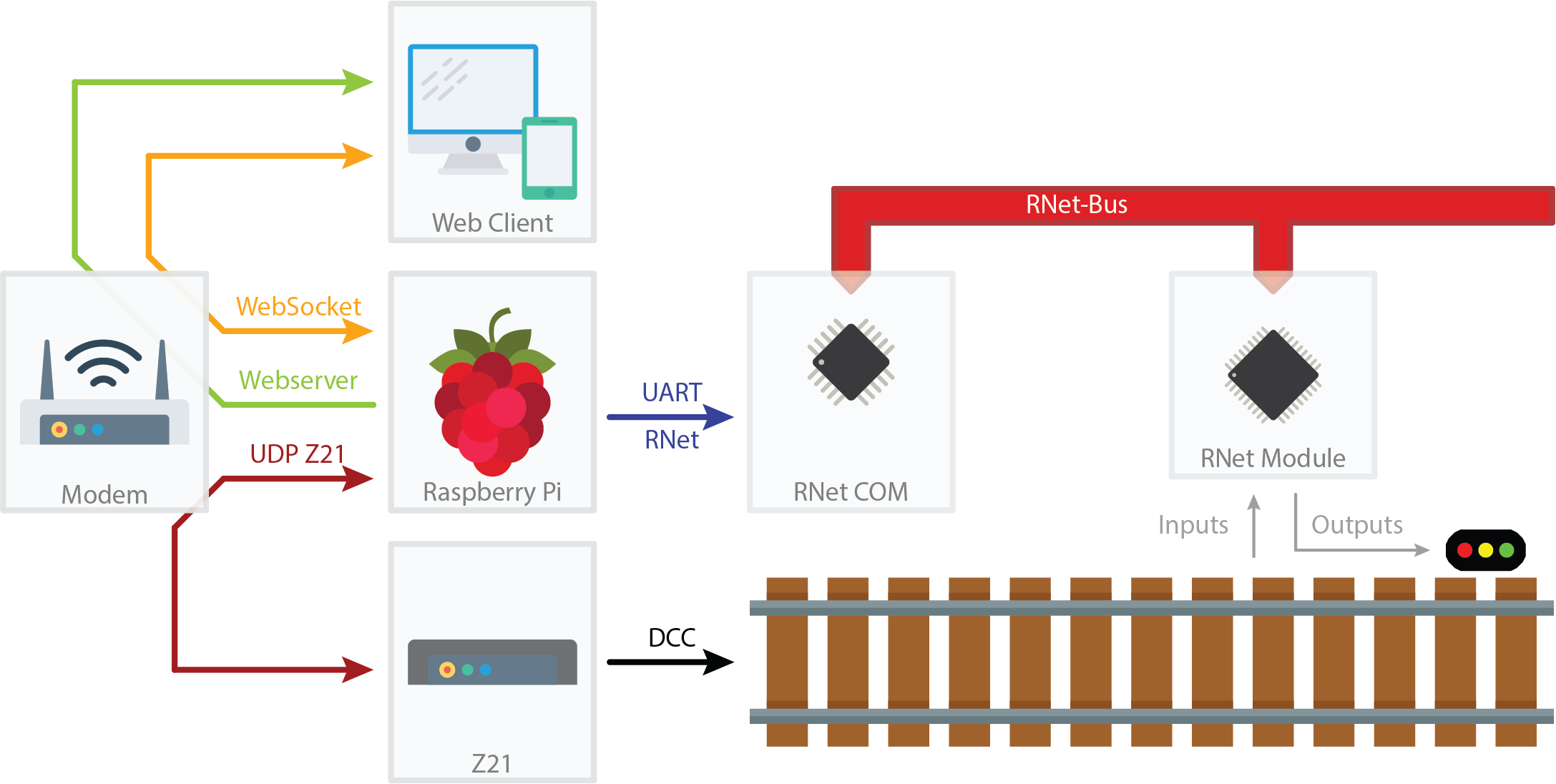
[Output 7](#_Toc506196256)

[Input 8](#_Toc506196257)

[Set Device Parameters 9](#_Toc506196258)

[EEPROM on Arduino 10](#_Toc506196259)

# General Overview



# UDP Z21

The Z21 uses its own predefined protocol. It can be studied from the “Z21\_LAN\_Protokoll V1.05.pdf” file, it is only published in German.

# WebSocket Protocol

*WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011, and the WebSocket API in Web IDL is being standardized by the W3C.*

Al binary representation have the MSB at the left side, otherwise it is explicitly stated.

Each Websocket message is send as a binary packet.

You can register to certain topics as a Websocket client using the protocol properties. 25510 (1111 11112) registers you to all the topics, e.g. “Sec-WebSocket-Protocol: 255”

The flags are setup as following:

ABCD EFGH

A – C = Unused  
D = Admin subscription  
E = Messages  
F = Switches  
G = Track  
H = Trains

The messages are divided into 4 different categories. The system checks for the first set bit. The packet belong to a category if the first bit equals the flag.

## Admin Controls

All messages that has a format of 1xxx xxxx (0x80 flag) is reconised as an admin control packet

### Clear track

*Message from client to server and from the server to every client*Delete all the track from current layout, or notify all members that the track has been cleared.

|  |
| --- |
| 0x80 |

### Reload track

Rescan the track for modules. They still need to be ‘joined’ together.

0x81

### Reload Previous track

Reload a layout from a previous setup. They don’t have to be ‘joined’ together

0x82

### Track info

*Message from client to server*Request data from the track

|  |
| --- |
| 0x83 |

*Message from server to client*

Get the info of the track: voltage, current, uptime, downtime. NOT IMPLEMENTED

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0x83 | Int16 A | Uint16 B | Uint16 C | Uint16 D |

A Track Voltage (mV) 16 bit Integer  
 B Track Current (mA) 16 bit Unsinged integer  
 C Uptime (sec) 16 bit Unsigned  
 D Downtime (sec) 16 bit Unsigned

### Reset Switches

Reset all switches to default

0x84

### Force Switch

Force a switch to a state independent of reserved state.

0x85

### Toggle (Light) Output

Toggle output of modules on or off.

0x86

### All trains back to depot

Send all trains back to depot, only if depot space is available.

0x87

### Emergency Stop, Admin authority

Set emergency stop. Can only be release with admin authority (the send code).

0x90 Admin Code 2 bytes (0-65534)

### Emergency Release, Admin authority

Release Emergency stop.

0x91 Admin Code 2 bytes (0-65534)

### Disable Admin authority for this connection

*From client to server*Logout from the admin services. The server will return with 0x15 (Broadcast change) message to confirm the request.

0xFE

### Enable Admin authority for this connection

*From client to server*Login as admin to the services. The server will return with 0x15 (Broadcast change) message, confirming or denying the request.

0xFF Password hash length Password hash

Train (0x40 Flag)

### New train in library

0x40

### Link train

*Message from client to server*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0x41 | AAAA AAAA | BBBB BBBB | xxxC CCCC | CCCC CCCC |

AAAA AAAA = Follow ID  
 BBBB BBBB = Real ID  
 xxxC CCCC CCCC CCCC = Message ID

*Message from server to client*

|  |  |  |
| --- | --- | --- |
| 0x41 | AAAA AAAA | BBBB BBBB |

AAAA AAAA = Follow ID  
 BBBB BBBB = Real ID

### Speed control

*Data from client to server*

|  |  |  |
| --- | --- | --- |
| 0x42 | AAAA AAAA | BCCC CCCC |

AAAA AAAA = Follow ID  
 Bxxx xxxx = Direction, set = forward, unset = reverse  
 xCCC CCCC = Speed

### Function control

*Data from client to server*

0x43

### Operation change

*Data from client to server*

0x44

### Train data from Z21

*Data from server to client*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0x45 | AAAA AAAA | DB0 | … | DBn |

AAAA AAAA = Train ID

DB0 – DBn = Z21 Databits *LAN\_X\_LOCO\_INFO*

### Add Route

0x46

### Delete Route

0x47

## Track (0x20 Flag)

### Toggle Switch

*Mostly from client to server*Toggle a switch from diverging to straight or vice versa

|  |  |  |
| --- | --- | --- |
| 0x20 | AAAA AAAA | xBBB BBBB |

AAAA AAAA = Module number of switch  
 xBBB BBBB = Switch ID

### MSSwitch toggle up

*Mostly from client to server*Increase the state of a MSSwitch

|  |  |  |
| --- | --- | --- |
| 0x21 | AAAA AAAA | xBBB BBBB |

AAAA AAAA = Module number of switch  
 xBBB BBBB = MSSwitch ID

### MSSwitch toggle down

*Mostly from client to server*Decrease the state of a MSSwitch

|  |  |  |
| --- | --- | --- |
| 0x22 | AAAA AAAA | xBBB BBBB |

AAAA AAAA = Module number of switch  
 xBBB BBBB = MSSwitch ID

### Set switch

*Mostly from (no gui) client to server  
Set a (MS)Switch to a specific state. 0 = Straight, 1 = Diverging.*

|  |  |  |  |
| --- | --- | --- | --- |
| 0x23 | AAAA AAAA | BCCC CCCC | DDDD DDDD |

AAAA AAAA = Module number of switch  
 Bxxx xxxx = Set if MSSwitch  
 xCCC CCCC = (MS)Switch ID  
 DDDD DDDD = New state

### Set switch reserved

0x24

### Set switches for route

*Mostly from client to server  
Server Calculates a route from point A to point B and set the switches accordingly and reserves them.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0x25 | AAAA AAAA | BBBB BBBB | CCCC CCCC | DDDD DDDD |

AAAA AAAA = Module number of point A  
 BBBB BBBB = Block of point A  
 CCCC CCCC = Module number of point B  
 DDDD DDDD = Block of point B

### Broadcast track occupation

*Only from server to client*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0x26 | AAAA AAAA | BBBB BBBB | CxxD EEEE | FFFF FFFF | AAAA AAAA… |

AAAA AAAA = Module number of block  
 BBBB BBBB = Block ID  
 Cxxx xxxx = Direction of block, set when block is reversed  
 xxxD xxxx = Set when block is occupied  
 xxxx EEEE = State of block: 0000 free, 0001 amber, 0010 red, 0011 unknown, 0100 ghost, 0101 blue  
 FFFF FFFF = ID of train, 0 if not occupied  
A – F is repeated for all blocks

### Broadcast states of switches

*Only from server to client*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0x27 | AAAA AAAA | BCCC CCCC | DDDD DDDD | EEEE EEEE | AAAA AAAA… |

AAAA AAAA = Module number of switch  
 Bxxx xxxx = Set if it is a Multi-state switch  
 xCCC CCCC = (MS)Switch ID  
 DDDD DDDD = New State  
 EEEE EEEE = Number of states, only for a MSSwitch, skip for a normal switch  
A – E is repeated for all (ms)switches

## General (0x10 Flag)

### Emergency stop

*Can be send or received*Server sends this message when the stop button has been pressed. Or client sends this when the button is pressed, server will confirm with same opcode.

|  |
| --- |
| 0x10 |

### Short Circuit stop

*Always from server to client  
Server sends this message at a short circuit detection, and stopping all traffic.*

|  |
| --- |
| 0x11 |

### Emergency release

*Can be send or received  
Server sends this message when the stop button has been released. Or client sends this when the button is released, server will confirm with same opcode.*

|  |
| --- |
| 0x12 |

### New Message

*Always from server to client  
Server sends a message to all clients to inform them of something.*

|  |  |  |  |
| --- | --- | --- | --- |
| 0x13 | AAAB BBBB | BBBB BBBB | Data |

AAAx xxxx = Message type  
 xxxB BBBB BBBB BBBB = Message ID

Data is depended on the type:   
*A new train has been put on the tracks (type 0)*

|  |  |  |  |
| --- | --- | --- | --- |
| Header | CCCC CCCC | DDDD DDDD | EEEE EEEE |

CCCC CCCC = and has this follow ID  
 DDDD DDDD = Train found on this module  
 EEEE EEEE = on this block  
*A train has split on the main tracks (type 1)*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Header | CCCC CCCC | DDDD DDDD | EEEE EEEE | FFFF FFFF | GGGG GGGG |

CCCC CCCC = follow ID of train  
 DDDD DDDD = Part A on this module  
 EEEE EEEE = on this block  
 FFFF FFFF = Part B on this module  
 GGGG GGGG = on this block

### Message update

0x14

### Message Clear

0x15 Message ID uint16\_t

### Change Broadcast flags

*From client to server*Client request a change in the flags. Server will send current status if ‘New flags’ is set to zero. This doesn’t change the actual broadcast flags

|  |  |
| --- | --- |
| 0x16 | New Flags |

*From server to client*Server confirms/updates the change in flags settings.

|  |  |
| --- | --- |
| 0x16 | Flags |

# Circuit of RailNet

There are 4 RNet devices in the network: COM interface, RailNet controller, RailNet output module, RailNet input module.

# Protocol of RailNet

A message must contain an opcode and a checksum. Some packet do not have a predefined length, then it is mandatory to send a length byte after the opcode.

## General

### Report ID

*From device to master*Each device sends this message at startup, so that the controller/master know which devices are in the network.

|  |  |  |
| --- | --- | --- |
| 0x00 | DevID | Checksum |

### Set Emergency STOP

*From device to maste*

A device that has an emergency button connect can send this message to call for an emergency stop on the layout.

|  |  |
| --- | --- |
| 0x01 | Checksum |

### Release Emergency STOP

*From device to master*A device that has an emergency button connect can send this message to release an emergency stop on the layout.

|  |  |
| --- | --- |
| 0x02 | Checksum |

### Set Power ON

*From device to master*A device that has a layout power switch connected can send this message to enable the track power.

|  |  |
| --- | --- |
| 0x03 | Checksum |

### Set Power OFF

*From device to master*A device that has a layout power switch connected can send this message to disable the track power.

|  |  |
| --- | --- |
| 0x04 | Checksum |

### Reset All Devices

From master to devices  
Resets the device to the initialization state, and resends the DeviceID packet.

|  |  |
| --- | --- |
| 0x05 | Checksum |

### Set Acknowledge

*From device to master*The device sends this message to acknowledge the changes in the device parameters.

|  |  |  |
| --- | --- | --- |
| 0x7F | DevID | Checksum |

## Output

### Toggle Single Address

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0x10 | DevID | Output L | Output H | Checksum |

Output Low: ID & 0x7F  
Output High: (ID >> 7) & 0x7F

Toggles the output of the specified ID.

### Pulse Single Address

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0x11 | DevID | Output L | Output H | Checksum |

Output Low: ID & 0x7F  
Output High: (ID >> 7) & 0x7F

Send a pulse on the output of the specified ID.

### Toggle Blink Single Address

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0x12 | DevID | Output L | Output H | Checksum |

Output Low: ID & 0x7F  
Output High: (ID >> 7) & 0x7F

Toggles the output of the specified ID.

### Toggle Multiple Addresses

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0x13 | Length | DevID | Output1 L | Output1 H | … | Checksum |

Output Low: ID & 0x7F  
Output High: (ID >> 7) & 0x7F

Toggles multiple addresses

### Set All output Addresses

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0x14 | Length | DevID | Output1 | Output2 | … | Checksum |

Output1: Output & 0x7F  
Output2: (Output >> 7) & 0x7F  
Output3: (Output >> 14) & 0x7F

Set all output to the specified bytes. Output 1 contains the ID of 0-6, output 2 contains 7-13, and so on. NOT IMPLEMENTED

### Set Blink mask of all addresses

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0x15 | Length | DevID | Output1 | Output2 | … | Checksum |

Output1: Output & 0x7F  
Output2: (Output >> 7) & 0x7F  
Output3: (Output >> 14) & 0x7F

Set the blink mask to the specified bytes. Output 1 contains the ID of 0-6, output 2 contains 7-13, and so on. NOT IMPLEMENTED

### Post all Output

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0x16 | Length | DevID | Output1 | Output2 | … | Checksum |

### Post Blink Mask

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0x17 | Length | DevID | Output1 | Output2 | … | Checksum |

### Request Read all Output states and Blink Mask

|  |  |  |
| --- | --- | --- |
| 0x47 | DevID | Checksum |

***Response:*** Post all Output (0x07)

## Input

### Post Single Input Address

|  |  |  |  |
| --- | --- | --- | --- |
| 0x06 | DevID | Output | Checksum |

DevID: ID & 0x7F

Output Low: ID & 0x7F  
Output High: (ID >> 7) & 0x7F

### Post Multiple Addresses

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0x07 | Length | DevID | Output1 | Output2 | … | Checksum |

DevID: ID & 0x7F

Output Low: ID & 0x7F  
Output High: (ID >> 7) & 0x7F

### Post All input Addresses

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0x08 | Length | DevID | OutReg1 | OutReg2 | … | Checksum |

DevID: ID

### Request Read all Inputs

|  |  |  |
| --- | --- | --- |
| 0x4C | DevID | Checksum |

DevID: ID & 0x7F

***Response:*** Post all Input (0x07)

## Set Device Parameters

***The device acknowledges all the changes to its parameters with the acknowledge packet***

### Change Device ID

|  |  |  |  |
| --- | --- | --- | --- |
| 0x50 | Old ID | New ID | Checksum |

### Set number of input modules and output modules

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0x51 | Dev ID | Input Modules | Output Modules | Checksum |

### Set blink Interval

|  |  |  |  |
| --- | --- | --- | --- |
| 0x52 | Dev ID | Interval scalar | Checksum |

Blink interval (ms) =

### Set Pulse length

|  |  |  |  |
| --- | --- | --- | --- |
| 0x53 | Dev ID | Pulse scalar | Checksum |

Pulse duration (ms) =

### Set Check input Interval

|  |  |  |  |
| --- | --- | --- | --- |
| 0x54 | Dev ID | Interval scalar | Checksum |

Check interval (ms) =

### Post All EEPROM variables

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0x55 | Length | Dev ID | Byte 0 | Byte 1 | … | Checksum |

### Request All EEPROM Values

|  |  |  |
| --- | --- | --- |
| 0x59 | Dev ID | Checksum |

## EEPROM on Arduino

|  |  |  |
| --- | --- | --- |
|  | 0x00 | 0x10 |
| 0x0 |  |  |
| 0x1 | Device ID |  |
| 0x2 | Input Devices |  |
| 0x3 | Output Devices |  |
| 0x4 | Blink scalar |  |
| 0x5 | Pulse scalar |  |
| 0x6 | Check scalar |  |
| 0x7 |  |  |
| 0x8 |  |  |
| 0x9 |  |  |
| 0xA |  |  |
| 0xB |  |  |
| 0xC |  |  |
| 0xD |  |  |
| 0xE |  |  |
| 0xF |  |  |