Railway Controller - Devs

Release fc1a257

Sidings Media

Dec 17, 2022

CONTENTS

List of Figures iii

List of Tables v

List of Code Blocks vii

1 Introduction 1
- 1.1 Terminology . . . 1
- 1.2 Symbols . . . 1
  - 1.2.1 Diagrams . . . 1

2 Architecture 3
- 2.1 System Overview . . . 3

3 Nodes 5
- 3.1 Types of node . . . 5
  - 3.1.1 Command node . . . 5
  - 3.1.2 Control node . . . 5
- 3.2 Node Communication . . . 6
  - 3.2.1 Registers . . . 6
  - 3.2.2 Client to Node . . . 7
  - 3.2.3 Node to Node . . . 9

4 Glossary 11

HTTP Routing Table 19

Index 21

Revision fc1a257 on branch main

Revision fc1a257 on branch main

LIST OF FIGURES

1.1 Symbol for client 1
1.2 Symbol for *bridge* 2
1.3 Symbol for *control node* 2
1.4 Symbol for *command node* 2
1.5 Symbol for board interconnects 2

2.1 Architecture of the railway controller system 3

Revision fc1a257 on branch main

Revision fc1a257 on branch main

LIST OF TABLES

Revision fc1a257 on branch main

Revision fc1a257 on branch main

LIST OF CODE BLOCKS

3.1 ABNF specification for a registers address 6
3.2 ABNF specification for register list 6
3.3 ABNF specification for client to node serial communication 8
3.4 ABNF specification for node to node serial communication 9

Revision fc1a257 on branch main

CHAPTER

ONE

INTRODUCTION

This documentation is designed for those working on projects planning to intergrate with the railway control system as well as those just wanting to find out how the system works in more detail.

Our documentation is currently a work in progress. Please bear us while we get it up to standard. Thanks for your patience.

The Sidings Media Team

1.1 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119[1].

1.2 Symbols

1.2.1 Diagrams

The following symbols are used in diagrams relating to the railway control system.

Note: The text inside a shape is used to provide more detail on it’s purpose.

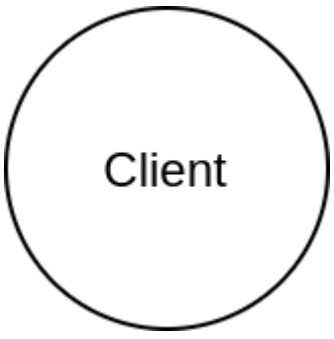

Fig. 1.1: Symbol for client

[1] https://www.rfc-editor.org/rfc/rfc2119.html

Fig. 1.2: Symbol for *bridge*

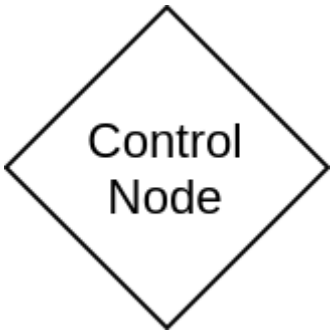

Fig. 1.3: Symbol for *control node*

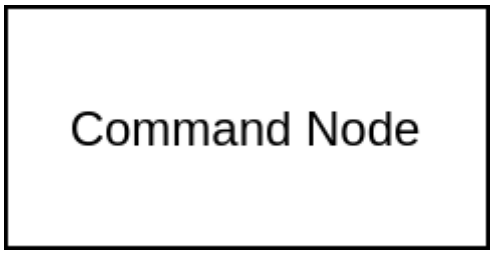

Fig. 1.4: Symbol for *command node*

Fig. 1.5: Symbol for board interconnects

CHAPTER

TWO

ARCHITECTURE

2.1 System Overview

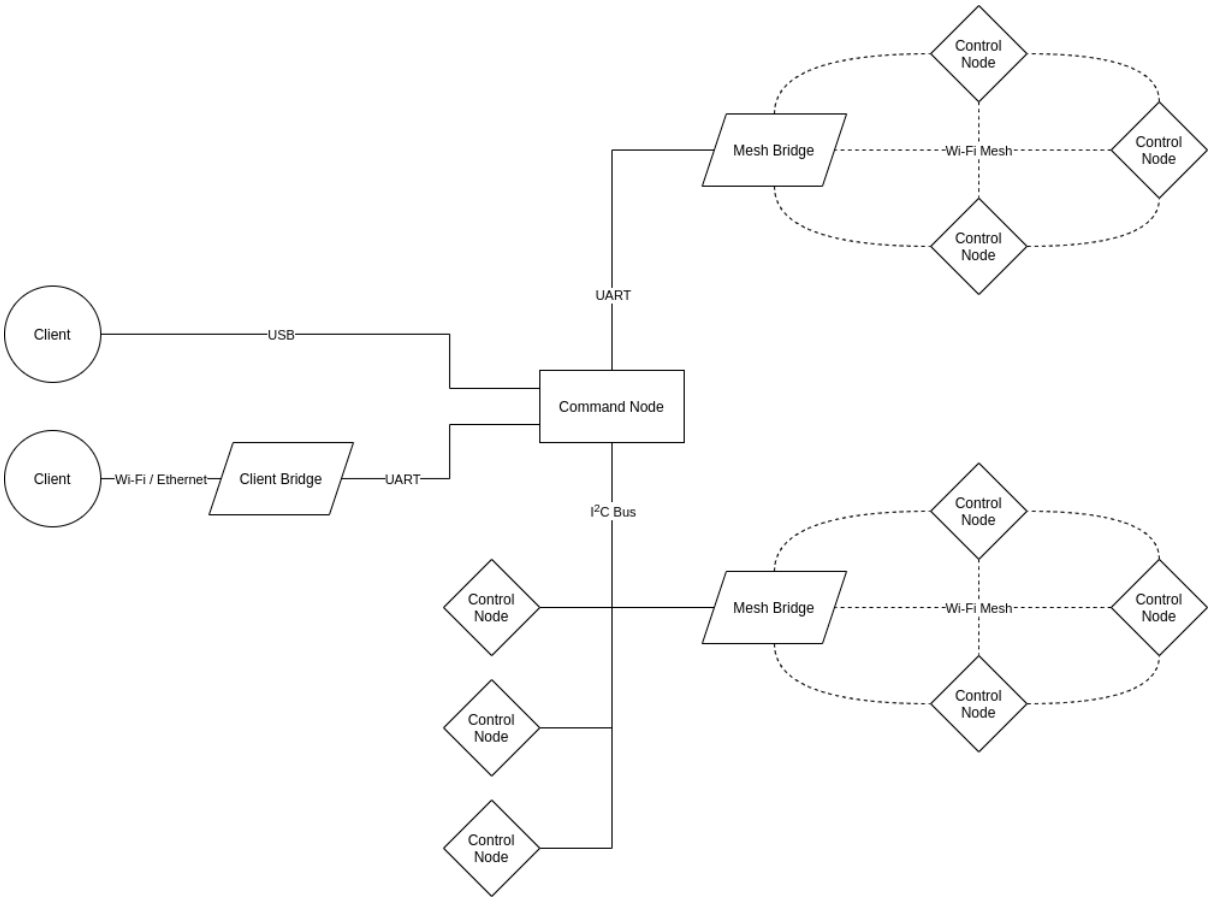

Fig. 2.1: Architecture of the railway controller system

Fig. 2.1 illustrates the overall architecture of the railway controller network. There are 3 types of boards in the network, the *command node*, the *control nodes* and the *bridges*. Only the *command node* MUST be present in all railway controller networks, all other nodes are OPTIONAL although it should be noted, a network with only the *command node* wouldn't be very useful.

Note: The I2C bus MUST support arbitration using SDA[2] to allow for multiple controllers to be on the bus at once.

[2] https://en.wikipedia.org/wiki/I%C2%B2C#Arbitration_using_SDA

CHAPTER

THREE

NODES

Nodes are the core building block of the railway controller system. They provide the physical interface between the user and the railway and are responsible for all aspects of the physical control of the network.

3.1 Types of node

There are two main classes of node withing the railway controller system, the *command node* and *control nodes*. Within the *control node* classification, there are multiple subclasses to indicate the functionality of the node and to dictate the role it has within the control system.

3.1.1 Command node

The *command node* is the central *node* in the railway control system. All commands are either generated by, or pass through the *command node*. It is responsible for making descisions based upon the client inputs, as well as its knowledge of the current state of the railway based upon previous commands as well as optional sensors placed around the network. There MUST only be one control node per railway control network.

3.1.2 Control node

The *control node* is responsible for directly interfacing with the railway infrastructure. *Control nodes* receive commands from the *command node* and adjusts its output accordingly. Each railway controller network will contain multiple *control nodes* that will control different aspects of the railway from the speed of the trains, to signals and points to monitoring position sensors to alert the *command node* when a *block* is occupied. There are different classes of *control node*.

Classes of control node

Speed control node
: The speed control node is responsible for controlling the speed of the trains on the track. Each node can drive one distinct line.

Accessory control node
: The accessory control node is responsible for controlling accessories such as signals and lighting throughout the layout.

Isolation control node
: The isolation control node is responsible for controlling the isolation sections of the network. It contains a number of relays that can be used to supply or cut power to a section of track.

Revision fc1a257 on branch main

3.2 Node Communication

Communication between *nodes* is vital to ensure proper operation of the control system. Due to the distributed nature of the system, a standardised system to communicate between nodes is essential. There are two ways in which nodes communicate within the network. Clients will usually communicate with the *command node* board via the REST API provided by the client bridge. Nodes will usually communicate with each other over serial interfaces such as sockets, I2C and UART. In order to ensure that communication is as smooth as posible, simple standards for both serial communication and communication with the REST API have been developed. The API is documented using the OpenAPI 3.1 specification and the serial communication is defined using Augmented Backus–Naur form (ABNF) as defined by RFC 5234[3].

3.2.1 Registers

Registers are conceptually similar to pigeonholes. In short, they are named locations on each node that store specific pieces of configuration data. These registers can be accessed and modified over the supported communications protocols. These form the basis for each nodes API and the registers are used to control the functionality of each node.

Each register has a locally unique address in the following format:

Listing 3.1: ABNF specification for a registers address

```
register-addr   =       1*(ALPHA / %x2D / %x5F)
                            ; Only support alphanumeric characters as
                            ; well as - and _
```

This address is used by commands to retrieve and modify the data in a specific register.

Note: Register addresses are case insensitive. I.e. speed_channel_1 is the same as SPEED_CHANNEL_1.

In cases where a request is made for the contents of a register but the register is empty, `null` should be returned.

Reserved registers

A number of addresses are reserved for use and MUST be present on all nodes. They are used by the control nodes to establish the specific features that an individual node supports and are essential to the correct interoperation of all nodes.

registers
A comma seperated list of all supported register addresses available on this node. The comma seperated list SHOULD have the following format.

Listing 3.2: ABNF specification for register list

```
register-list   =       *(register-addr %x4C)
                            ; 0 or more register addresses
                            ;   seperated by a comma without
                            ;   any spaces
```

Note: Any whitespace will be removed during processing of the list

serial
A 16 character long string representing the serial number of the node. The serial number is an arbitary string

[3] https://www.rfc-editor.org/rfc/rfc5234.html

that MAY be unique among boards. It is used solely for informational purposes. If no serial number is defined, `null` SHOULD be returned.

model
The model number of this node. The model number is an arbitary string of maximum length 256 characters that does not need to be unique. It is used solely for informational purposes. If no model number is defined, `null` SHOULD be returned.

bootloader
A string representing the current bootloader version installed on this node. This SHOULD be filled on all nodes. It is used to establish compatibility of firmware and supported features.

firmware
A string representing the current firmware version installed on the node. This SHOULD be filled in on all nodes.

3.2.2 Client to Node

REST API

This is used as the main form of communication between a *client bridge* and a *client*. The specification is defined using the OpenAPI 3.1 standard[4] and is listed below. A complete list of HTTP routes is also available at the end of this document. An interactive version[5] of the OpenAPI documentation is also available.

`GET /node/{addr}/register/{register}`

Get value of register

Return the value currently held in the specified register

Parameters

- **addr** (*string*) –
- **register** (*string*) –

Status Codes

- 200 OK[6] – Value of requested register
- 404 Not Found[7] – The requested node or register does not exist
- **default** – General Error

`PUT /node/{addr}/register/{register}`

Set register value

Set the value of the register to the specified value

Parameters

- **addr** (*string*) –
- **register** (*string*) –

Status Codes

- 200 OK[8] – General Success
- 400 Bad Request[9] – Data is invalid
- 404 Not Found[10] – The requested node or register does not exist
- **default** – General Error

[4] https://spec.openapis.org/oas/latest.html
[5] https://docs.railwaycontroller.sidingsmedia.com/projects/dev/en/latest/api/clientbridge.html
[6] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[7] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5

Bootloader

The client may, in order to complete some actions, decide to communicate with the bootloader interface of a *node*. If this is the case, the *reset* command should be issued to the board, and then any character send along the USB interface after approximately 1 second. This is to interupt the boot process. Various commands may then be sent to the bootloader as detailed below.

Listing 3.3: ABNF specification for client to node serial communication

```
; SPDX-FileCopyrightText: 2022 Sidings Media <contact@sidingsmedia.com>
; SPDX-License-Identifier: CC-BY-SA-4.0

; Specification for commands sent a client and a node over serial links.

; Commands

command         =       (update / interrupt / reset) CRLF

update          =       "update"
                        ; Indicate that the client would like to provide
                        ;   a firmware file to update the board

interrupt       =       CHAR
                        ; Any key to cancel normal boot seq and enter
                        ;   bootloader interface

reset           =       "reset"
                        ; Reset the boards microcontroller

; Command option values
string-val      =       1*(ALPHA / %x2D / %x5F)
                            ; Only support alphanumeric characters as
                            ; well as - and _

bin-val         =       "0b" 1*BIT

bool-val        =       "true" / "false"

hex-val         =       "0x" 1*HEXDIG

int-val         =       1*DIGIT

signed-int-val  =       [%x2d] int-val

null-val        =       "null"

;IPv6 Address from RFC5954
IPv6address     =       6( h16 ":" ) ls32
                        / "::" 5( h16 ":" ) ls32
                        / [               h16 ] "::" 4( h16 ":" ) ls32
                        / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
                        / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
                        / [ *3( h16 ":" ) h16 ] "::"    h16 ":"   ls32
                        / [ *4( h16 ":" ) h16 ] "::"              ls32
                        / [ *5( h16 ":" ) h16 ] "::"              h16
```

(continues on next page)

[8] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.1
[9] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.1
[10] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.5

(continued from previous page)

```
                        / [ *6( h16 ":" ) h16 ] "::"

h16             =       1*4HEXDIG

ls32            =       ( h16 ":" h16 ) / IPv4address

IPv4address     =       dec-octet "." dec-octet "." dec-octet "." dec-octet

dec-octet       =       DIGIT                   ; 0-9
                        / %x31-39 DIGIT         ; 10-99
                        / "1" 2DIGIT            ; 100-199
                        / "2" %x30-34 DIGIT     ; 200-249
                        / "25" %x30-35          ; 250-255
```

3.2.3 Node to Node

Serial Commands

Serial commands are used for inter-node communication in almost all cases. Most nodes are connected via serial communication mediums such as I2C, UART and sockets. In these cases, the below specification for serial commands should be used.

These commands are loosely inspired by SQL statements. There are two types of command, the `get` command and the `set` command. As the names suggest, `get` commands retrieve a value from a register and `set` commands set the value of a register.

In most cases, it is required to state the address of the node the command is being sent to. This is to facilitate the command traversing client bridges and interface cards. The only circumstance where the address can be omitted is on commands sent by the *command node* to devices directly connected on the I2C bus. This is possible as the address is already specified by the *command node* when sending the command over the I2C bus.

Listing 3.4: ABNF specification for node to node serial communication

```
; SPDX-FileCopyrightText: 2022 Sidings Media <contact@sidingsmedia.com>
; SPDX-License-Identifier: CC-BY-SA-4.0

; Specification for commands sent between nodes over serial links.

; Commands
command         =       (set / get / reset) [SP addr] CRLF
                            ; SQL like format. CRLF indicates line end.
                            ;   Address is only required when sending
                            ;   commands via an interface card. I.e.
                            ;   when being sent over the network. It is
                            ;   not required for direct I2C interfaces.

get             =       "get" SP register-addr
                            ; GET commands used to retrieve data from
                            ;   registers

set             =       "set" SP register-addr %x3D register-val
                            ; SET commands used to set the value of a
                            ;   register

reset           =       "reset"
                            ; RESET command used to cause system reset
```

(continues on next page)

(continued from previous page)

```
                            ;   and allow access to bootloader interface

addr              =       "at" SP node-addr

; Command option values
register-addr     =       string-val

node-addr         =       hex-val
                          / IPv6address

register-val      =       string-val
                          / bin-val
                          / bool-val
                          / hex-val
                          / int-val
                          / signed-int-val
                          / null-val

string-val        =       1*(ALPHA / %x2D / %x5F)
                              ; Only support alphanumeric characters as
                              ; well as - and _

bin-val           =       "0b" 1*BIT

bool-val          =       "true" / "false"

hex-val           =       "0x" 1*HEXDIG

int-val           =       1*DIGIT

signed-int-val    =       [%x2d] int-val

null-val          =       "null"

;IPv6 Address from RFC5954
IPv6address       =       6( h16 ":" ) ls32
                          / "::" 5( h16 ":" ) ls32
                          / [               h16 ] "::" 4( h16 ":" ) ls32
                          / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
                          / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
                          / [ *3( h16 ":" ) h16 ] "::"    h16 ":"   ls32
                          / [ *4( h16 ":" ) h16 ] "::"              ls32
                          / [ *5( h16 ":" ) h16 ] "::"              h16
                          / [ *6( h16 ":" ) h16 ] "::"

h16               =       1*4HEXDIG

ls32              =       ( h16 ":" h16 ) / IPv4address

IPv4address       =       dec-octet "." dec-octet "." dec-octet "." dec-octet

dec-octet         =       DIGIT                   ; 0-9
                          / %x31-39 DIGIT         ; 10-99
                          / "1" 2DIGIT            ; 100-199
                          / "2" %x30-34 DIGIT     ; 200-249
                          / "25" %x30-35          ; 250-255
```

CHAPTER

FOUR

GLOSSARY

block
: A section of track which should only ever have one train in it at once. It is shown as occupied if there is a train within the block.

bridge
: An individual board that joins different sections of the control network together.

client
: An end users device that is running a form of railway controller software used to interact with the network.

client bridge
: The *bridge* between the *command node* and the clients home network.

cluster bridge
: The *bridge* between the *command node* and the Wi-Fi node cluster.

command node
: The central node in the railway controller system responsible for receiving user inputs and issuing commands based upon them.

control node
: Boards that interface directly with the railway to control the railway harware. E.g. points, speed of trains.

node
: An individual board within the system that can accept and act upon commands.

LIST OF TABLES

Revision fc1a257 on branch main

LIST OF FIGURES

Revision fc1a257 on branch main

LIST OF CODE BLOCKS

Revision fc1a257 on branch main

HTTP ROUTING TABLE

/node

GET /node/{addr}/register/{register}, 7
PUT /node/{addr}/register/{register}, 7

Revision fc1a257 on branch main

INDEX

A

Accessory control node, 5

B

block, 11
bootloader, 7
bridge, 11

C

client, 11
client bridge, 11
cluster bridge, 11
command node, 5, 11
control node, 11

F

firmware, 7

I

Isolation control node, 5

M

model, 7

N

node, 11

R

registers, 6

S

serial, 6
Speed control node, 5

Revision fc1a257 on branch main