NAME

rbserver – Central service to manage readerboards and busylights

SYNOPSIS

rbserver [**−conf** *configfile*]

DESCRIPTION

This program opens connections to all the readerboards and busylight units described in its configuration file and sends commands to them in order to maintain centrally-controlled messages and status indicators on them. It opens an HTTP web service endpoint which provides a simple-to-use interface for other users on the network to request display changes on the managed devices.

OPTIONS

−conf *configfile*

Read the server configuration from the named file. This is a required parameter.

CONFIGURATION

The configuration file is a JSON object with the following fields. All fields have string values unless otherwise indicated.

GlobalAddress *(int)*

Specifies the address, which must be in the range 0−15, that may be sent to the devices to indicate a global command that all devices should respond to. This same address *must* also be configured into each of the devices so they all agree with each other and the server about what the global address should be. By default, the devices use global address 15, but this may be changed by reconfiguring the devices.

Devices *(object)*

The list of all connected Readerboard and Busylight devices is provided here. The value in this case is a JSON object (i.e., key/value map) where the keys are the device IDs and the corresponding values are JSON objects which describe the attributes of that device. The IDs must be numeric and within the range 0–63, and must not be the global address. For devices not directly attached via USB (e.g., ones on RS-485 networks), the device address must also be configured to match the ID number used by the server.

(**N.B.:** in order to comply with the JSON representation standard, these keys must appear in the configuration file as quoted strings, so the entry for device with address 42 would be introduced here with the key **"42"**, not the integer value **42**.)

The fields in the JSON object listed for each device include the following:

DeviceType

This indicates the hardware type for this unit. Valid values as of this writing include **Busylight1**, **Busylight2**, **Readerboard3_RGB**, and **Readerboard3_Monochrome**.

NetworkID

The name of the network or direct connection to which this device is attached. This name must be defined in the **Networks** field (q.v.).

Description

An arbitrary description of the device, its location, purpose, or whatever other information about it is useful to note.

LightsInstalled

If this unit's discrete status LEDs are a non-standard configuration, this field specifies the one-letter codes to represent the color of LED installed at each level, starting with the bottom LED. If left blank or omitted, this will default to the standard configuration, which is "**GyYrRBW**" for busylights and "**GyYrRbBW**" for readerboards.

Serial The serial number of the device. The server will check this by probing the unit and comparing the serial number it reports, to help identify mistakes in the server configuration.

Networks *(object)*
All the serial ports over which the server will communicate with the various devices are enumerated here as a JSON object whose keys are arbitrarily-named network IDs, and the corresponding values describe that connection with the following fields:

ConnectionType
Defines the type of connection (and thus, the protocol which must be used to communicate over it). As of this writing, the valid values are **RS-485** and **USB**.

Device This gives the system device name (e.g., "**/dev/ttyACM0**") of the serial port which the server is to open in order to communicate with the device(s) connected there. If you want the server to try to find the appropriate device, then leave this field out and instead specify **DeviceDir** and **DeviceRegexp** (see below).

DeviceDir
As an alternative to hard-coding a device name in **Device**, This specifies the directory in which all the system's serial port devices may be found.

DeviceRegexp
A regular expression that matches serial ports which may be used for our devices. The first device found in **DeviceDir** which matches this expression and can be successfully opened by the server process will be used for this network.

BaudRate *(int)*
The speed at which to communicate over the serial port.

LogFile
The name of a file where the server should log its state and activity along with debugging and error messages. By default, these messages are sent to the standard output.

PidFile If specified, this names a file in which the server will record its process ID number. This file is deleted when the server shuts down.

Endpoint
This gives the TCP endpoint which is to accept HTTP requests. It has the form [*hostname*]**:***port*.

UserVariables *(object)*
This lists a set of user-defined variables which can be included in posted messages. The value is a map of variable names to their values. A variable name listed here as "**name**" can be inserted into a posted message as "**{$name}**". The server will start with this set of variables defined. More may be added using the **update** web service API call.

Example Configuration

```
{
        "PidFile": "/tmp/rbserver.pid",
        "GlobalAddress": 15,
        "Endpoint": ":43210",
        "Devices": {
                "1": {
                        "DeviceType": "Busylight2",
                        "NetworkID": "directlight",
                        "Description": "Work busy indicator (kitchen)",
                        "Serial": "B1234"
                },
                "2": {
                        "DeviceType": "Readerboard3",
                        "NetworkID": "sparkbus",
                        "Description": "Kitchen status board",
                        "Serial": "RB1234"
                },
                "3": {
                      "DeviceType": "Busylight2",
                      "NetworkID": "sparkbus",
                      "Description": "Downstairs remote busy indicator",
                      "Serial": "B1235"
                }
          },
          "Networks": {
                "directlight": {
                      "ConnectionType": "USB",
                      "Device": "/dev/usbtty0",
                      "BaudRate": 9600
                },
                "sparkbus": {
                      "ConnectionType": "RS-485",
                      "Device": "/dev/usbtty1",
                      "BaudRate": 9600
                }
          },
          "UserVariables": {
                "shift": "day",
                "SecretWord": "albatross"
          }
    }
```

WEB SERVICE API

In the current pre-release version, no authentication protection is implemented, so steps must be taken by the user or system administrator to ensure that the endpoint is not reachable by untrusted systems or users.

The server responds to requests with URLs that begin with the prefix

`http://`*host*`:`*port*`/readerboard/v1/`*command*`?a=`*address(es)*

followed by one of the following commands. All commands accept a parameter **a** which lists one or more device ID numbers which should receive the command. If the global address is in the list, then the command will be sent to all units. If more than one address is given, the addresses are separated from each other with commas.

Simple Device Commands

The following commands carry out the requested operation and do not send a reply unless they need to report a problem. This allows them to be sent with something as simple as a **curl**(1) command at the shell prompt or in a script. In the descriptions that follow, we will omit the "**?a=**..." parameter common to all commands which has already been described above.

alloff Extinguish all LEDs on the device(s), including matrix LEDs and discrete status indicators.

bitmap...[**&merge**][**&trans=***transition*][**&pos=***n*]**&image=***bitmap*

Draw a bitmap pattern on the LED matrix starting at column *n*. If **merge** is specified with no value or the value **true**, the pixels are merged with those already on the display; otherwise the previous contents of the affected columns are lost in favor of the new pattern.

The *bitmap* value consists of a number of planes separated by dollar-signs ("**$**"). Each plane consists of a number of two-digit hex values which provide the pattern of lit pixels for the corresponding column, with the least-significant bit representing the top pixel. On RGB models, there are four planes specified: red, green, blue, and flash; a bit in the red, green, or blue plane adds that color to the corresponding pixel, combining to allow eight colors (including off) to be displayed. Any pixel whose corresponding bit in the flash plane is set will be flashed on and off instead of burning steady. Monochrome displays are similar but only use two planes: the first indicates which pixels are lit and the second indicates which are flashing.

Consider a 4x4-pixel red square filled in with green, which occupies the middle rows like so:

```
....
....
RRRR
RGGR
RGGR
RRRR
....
....
```

To draw this on the readerboard starting at column 0, send the command

> **bitmap?a=***addresses***&pos=0&image=3c24243c$001818$$**

To make the green inner square flash, then the command would instead be

> **bitmap?a=***addresses***&pos=0&image=3c24243c$001818$$001818**

clear Clear the LED matrix so all pixels are off.

color...**&color=***c*
: Sets the current color to the specified color code.

configure-device...**&rspeed=***baud***&uspeed=***baud***&address=***a'***&global=***g'*
: Sets configuration values for the target device, which must be a single device connected via USB, since the devices will refuse to accept this command over RS-485. Generally, these settings would be made via the stand-alone **setsn**(1) program before adding the device to the server.

  This sets the device's address to *a'*, global address to *g'*, and baud rates for USB and RS-485 interfaces.

diag-banners
: Causes the target device to display the power-on banner messages again, showing its configuration values. It will be unresponsive to commands until this display is finished.

dim...**&l=***led***&d=***level*
: Sets the brightness *level* for the specified *led* (either an LED code (q.v.) or "*" to dim all discrete status LEDs or "_" to dim the LED matrix). The values range from 0 (fully off) to 255 (fully on). The LED matrix works best at or near full brightness, but the discrete LEDs can be dimmed to any levels. Note that not all devices support dimming on all LEDs.

flash...**&l=***L0L1...LN*[**&up=***time***&on=***time***&down=***time***&off=***time*]
: Sets a pattern of 0−64 lights which will be illuminated in sequence on the discrete status LEDs. Each value *L0*, *L1*, etc., is either an ASCII digit representing the position in the stack of status lights or a single character which represents the color of the desired light (as defined by each specific target device), or the special character "_", which means no light should be on at that place in the sequence. There are no delimiters between each value in the sequence list. A sequence of zero length turns off the flasher entirely. A sequence of a single value merely blinks that one LED on and off.

  If the **up**, **on**, **down**, and **off** values are given, they specifiy custom flashing and fading effects. Each light in the sequence will fade up from fully off to fully on (or to the set brightness value if the dimmer was adjusted for that light) over the course of the duration specified to the **up** parameter, which is a value in the range 0−63 in units of 1/10 second. Likewise the other parameters indicate how long the light is held on, the time over which it fades down to off, and how long it remains off.

font...**&idx=***f*
: Select the font with ID number *f*.

graph...**&colors=***r0r1r2r3r4r5r6r7*
: Add a data element to the scrolling histogram display where the color of each pixel is given by color codes *r0* (for the top pixel in the column) through *r7* (the bottom pixel). The colors are either a string of eight single-character color codes such as "**RRYYGGGG**" or a comma-separated list of eight color names such as "**red,red,yellow,yellow,green,green,green,green**".

graph...**&v=***n*
: Add a data element to the scrolling histogram display as a stack of *n* lit pixels starting from the bottom of the column.

light...**&l=***L0L1...lN*
: Specifies up to 64 lights in the same fashion as for the **flash** command, but the set of lights are all turned on steady instead of flashing. This command disables the flasher if it was running, but does not affect the strober, thus allowing a steady pattern with a strobe as a secondary status indicator. Busylight units can only light a single LED level at a time, while readerboard units can light any number.

morse...**&t=***message***&l=***led*
: Sends *message* via Morse code on the speaker (if *led* is "_") or by flashing the designated LED. Sends the prosign SK at the end of the message. Other prosigns may be given by putting the following codes in the message (shown here using URL encoding):

Code	**Prosign**
%1802	AR (start message)
%1803	SK (end of message)
%1806	VE/SN (verified)
%1807	KA/CT (attention)
%1811	SOS (distress)
%1812	DDD (relayed distress
%1813	AS (wait)
%181E	BT (break)
%187F	HH (correction)

move...**&pos=***n*
: Move the cursor to the specified absolute column number.

off Turn off the discrete status LED set on the target devices. The flasher and strober are stopped.

save...**&type=***type*
: Save the current settings of the given *type* to EEPROM, so they will remain in effect even after the device is rebooted. The *type* value may be:

D Save all dimmer values.

scroll...[**&loop**]**&t=***text*
: Scrolls *text* across the readerboard. If **loop** is present (with no value or the value **true**), the text will continually repeat in a never-ending scrolling display. Otherwise, it stops scrolling at the end of the text string. The *text* value may contain special codes (these are introduced by control characters, so they are represented here using the URL encoding as normally specified in the URL sent to the server):

%06*f* Switch the font to the one with the specified ID (index) *f*.

%0B*c* Switch to the specified color. This must be a single-character color code.

%18*hh* Insert character with the codepoint given by the two-digit hex number *hh*.

sound...[**&loop**]**&notes=***notes*
: Make a sound on the device's built-in speaker. Some devices are only capable of making a single frequency "beep" (in which case all notes will sound at that pitch), while others can play a range of musical notes. If **loop** is given, the sound will play repeatedly until another **sound** command is sent.

If *notes* is empty or missing, any sound currently playing is stopped. Otherwise it gives a string of notes to be played. Each note is **R** for a rest period or a letter **A–G** to indicate a musical note to play. The note may be followed by **b** or **#** to indicate a flat or sharp, and is then followed by a digit **0–8** to indicate the octave number. Following this is a two-digit hex value giving the duration to play the note (or rest) in 1/100ths of a second. Notes may range from **B0** to **D#8**.

An example three-note attention sequence is "**notes=G432E532C564**".

strobe...**&l=***L0L1...LN*
: Sets a pattern of 0−64 lights which will be strobed (briefly flashed with a pause between each) in sequence on the discrete status LEDs. Each value *L0*, *L1*, etc., is either an ASCII digit representing the position in the stack of status lights or a single character which represents the color of the desired light (as defined by each specific target device), or the special character "_", which means no light should be strobed at that place in the sequence. There are no delimiters between each value in the sequence list. A sequence of zero length turns off the strober entirely. A sequence of a single value merely strobes that one LED.

test Run a test pattern on the target device(s) which demonstrates that the LEDs are all connected and the device's circuitry is functioning properly.

text...[**&merge**][**&align=***alignment*][**&trans=***effect*]**&t=***text*
: Displays *text* on the readerboard, optionally merging the pixels that spell out the letters with existing pixels on the sign and/or transitioning to the message using an effect. The *text* value may contain special codes (these are introduced by control characters, so they are represented here using the URL encoding as normally specified in the URL sent to the server):

%03*p*	Move cursor to column *p* (encoded as a six-bit integer value).
%06*f*	Switch the font to the one with the specified ID *f*.
%08*p*	Move the cursor *p* columns to the left.
%0B*c*	Switch to the specified color.
%0C*p*	Move the cursor *p* columns to the right.
%18*hh*	Insert character with the codepoint given by the two-digit hex number *hh*.

configure-device...**&rspeed=***baud***&uspeed=***baud***&address=***a***&global=***g*
: Command the target device (which must be a single device) to change its baud rates for RS-485 and USB, as well as its own device address and its understanding of the system global address.

Query Commands

The following commands query devices to get information about them, and then report that information back in a JSON-formatted reply as an object with device IDs as keys and JSON-formatted payloads as the corresponding values.

busy Query the device's discrete status LEDs, reporting on which lights are currently illuminated, and the state of the strober and flasher.

query Query the device's full status and report it. This includes the information reported by the **busy** command, as well as hardware and firmware revision numbers, serial number, contents of the LED matrix, etc.

Server Commands

The following commands execute higher-level operations on the server which in turn affects how it manages the state of the connected devices rather than being commands that are sent directly to the devices themselves.

current
: Returns a JSON object that describes the server's current notion of the status indicators on all the devices. This is similar to the **busy** command except it does not take the extra step of individually polling each device to query their status; it reports what state the server last told the devices to show.

post...**&t=***text***&id=***id*[**&trans=***effect*][**&until=***dt*][**&hold=***d*][**&color=***c*][**&visible=***d*][**&show=***d*][**&repeat=***d*]
: Add a new message to the display list for the specified target devices. They will display these messages until they expire. Sending other display-manipulating commands will suspend this display until a **clear** command is issued to the device.

A number of variables, special tokens, and scheduling parameters may be given as documented in the *Readerboard User's Guide*.

postlist...**&id=***id_or_/regex*
: Report back with a list of all the messages in the display queue for the specified device(s) that match the specified *id* or regular expression.

unpost...**&id=***id_or_/regex*
: Removes all messages matching the *id* or regular expression from the display list of the target devices.

update...**&***k0***=***v0*...**&***kn***=***vn*
: Update the value of one or more user variables which may be substituted into posted display lists.

SEE ALSO

setsn(1).

More details about how values are encoded and handled by devices and the server are described in greater detail in the *Readerboard User's Guide* document that accompanies the project source code.

AUTHOR

Steve Willoughby / steve@madscience.zone.

COPYRIGHT