令和2年度

卒業研究報告書

研究題目

Node.jsによるRGBマトリクスLED制御システム開発

電気情報工学科

No. 3 氏名 石川 瑛祐

提出年月日

令和3年 2月18日

指導教員

香川 福有

新居浜工業高等専門学校

1. はじめに - 1 -

2. 使用装置 - 1 -

2.1 RGBフルカラードットマトリクスLEDパネル 32x16ドット

2.2 RGB Matrix HAT + RTC

2.3 Raspberry Pi 3 Model B Rev 1.2

2.4使用装置一覧

3. 使用ソフトウェア - 3 -

3.1 Raspberry Pi OS

3.2 Node.js

3.3 node-rpi-rgb-led-matrix

3.4 Vue.js

3.5 UFO

3.6使用ソフトウェア一覧

4. 配線 - 5 -

5. セットアップ - 5 -

6. BDFデコーダの作成 - 5 -

6.1 BDFフォントの書式

6.2 BDFデコーダの作成

7. システムの概要 - 7 -

7.2 バックエンドシステム

7.3 フロントエンドシステム

8. 機能 - 8 -

8.1 テキスト表示

8.2定型文

8.3 RSS

8.4動作風景

9. システムのビルドと開発 - 15 -

9.1 開発用サーバ

9.2 ビルド

9.3 本番環境

9.4 永続化と自動起動

10. 課題 - 16 -

11. おわりに - 17 -

12. 参考文献 - 17 -

# **はじめに**

本研究はRGBマトリクスLEDパネル(以下、LEDパネル)をWebブラウザから制御するシステムを開発するテーマである。Webブラウザから操作を受け付けてLEDパネルにテキストを表示するアプリケーションの作成を行った。

# **使用装置**

本システムは以下の装置によって構成される。

## 2.1 RGBフルカラードットマトリクスLEDパネル 32x16ドット

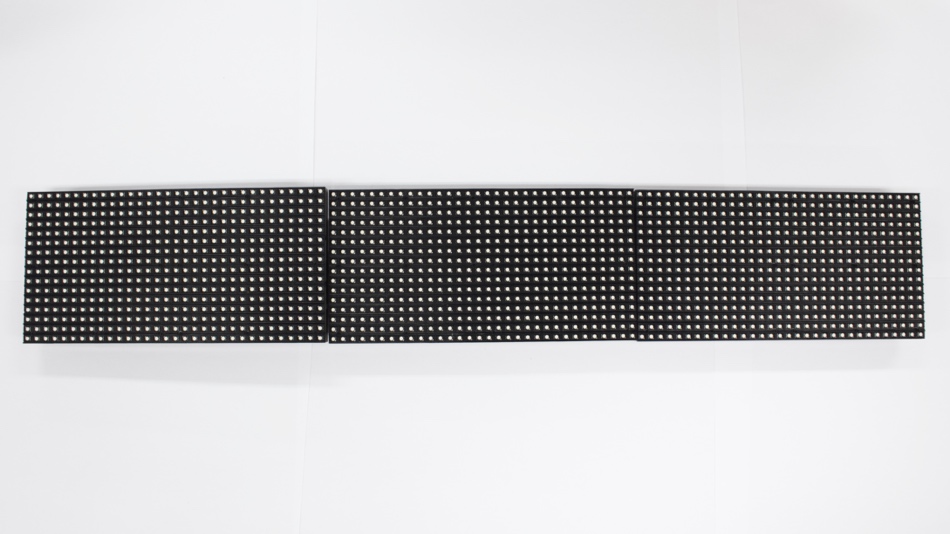
 本システムの主要な出力装置である。使用したパネルを図1に示す。このパネルを横に3枚デイジーチェーンして連続してテキストが表示される。2.2のHATを接続してRaspberry Piからの入力を受け取る。

図 1. RGBフルカラードットマトリクスLEDパネル

## 2.2 RGB Matrix HAT + RTC

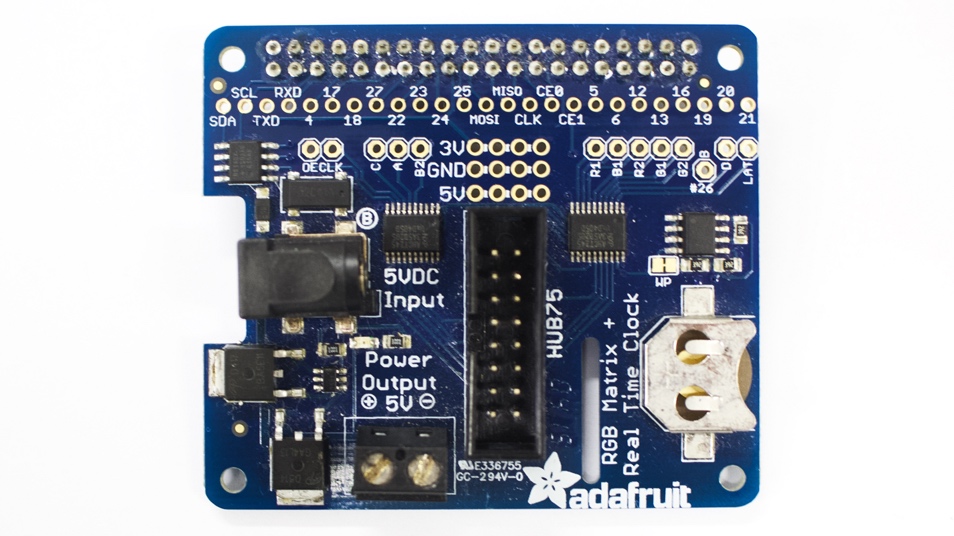
 このRGB Matrix HAT + RTCはRaspberry PiでRGBマトリクスLEDパネルを操作する拡張ボード型のコントローラである。図2に使用したHATを示す。

図 2. RGB Matrix HAT+RTC

Raspberry Pi Foundationが公開しているHAT(Hardware Attached on Top)の仕様に準拠しているため、本研究で使用したモデル以外のRaspberry Piでも利用できる。

装置名にあるようにRTCを搭載しているため、Raspberry Piの電源が切断されていても、時刻の表示などが可能である。

16x32、32x32、32x64、64x64のHUB75規格のパネルに対応しており、デイジーチェーンで接続すれば更なる表示領域の拡張にも対応する。

## 2.3 Raspberry Pi 3 Model B Rev 1.2

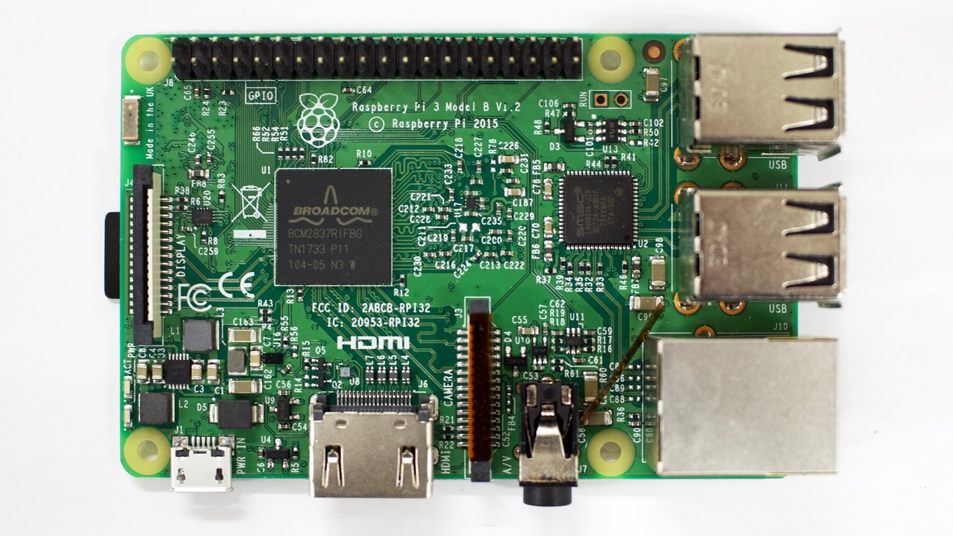
 ARMプロセッサ搭載の教育用シングルボードコンピュータであり、本システムのパネル制御、Web機能の提供を行う。 図3に使用したボードを示す。以下表1は今回使用するモデルの仕様である。

図 3. Raspberry Pi 3 Model B Rev 1.2

表 1. Raspberry Pi 3 Model B+の仕様

|  |  |
| --- | --- |
| 項目 | 仕様 |
| CPU | Broadcom BCM2837 1.2GHz 64-bit quad-core ARMv8 Cortex-A53 |
| RAM | 1GB |
| USB | USB2.0 x4 |
| LAN | RJ-45 x1 (IEEE802.3i(10BASE-T)、IEEE802.3u(100BASE-TX))、無線LAN(IEEE802.11b/g/n(2.4GHz)) |
| その他外部接続 | HDMI、 microSDカードスロット、 3.5mmジャック、 40pin GPIO、 Bluetooth v4.1(Bluetooth Low Energy) |

## 2.4 使用装置一覧

使用装置一覧を表2に示す。

表 2. 使用装置一覧

|  |  |  |  |
| --- | --- | --- | --- |
| 装置名 | メーカ名 | 型番 | 用途 |
| RGBフルカラードットマトリクスLEDパネル | 不明 | 不明 | テキストの表示を行う出力装置。 |
| RGB Matrix HAT + RTC for Raspberry Pi | Adafruit | 2345 | マトリクスLEDに電源や制御信号を供給する。 |
| Raspberry Pi 3 Model B Rev 1.2 | Raspberry Pi Foundation | RASPBERRYPI3MODELB | システム全体を統括する。 |
| ACアダプタ | 東亜無線電機株式会社 | 3A-502DA05 | HAT及びマトリクスLEDへ電源を供給する。 |
| USB ACアダプタ | 秋月電子通商 | AD-B50D200 | Raspberry Piへ電源を供給する。 |
| フラットケーブル | 不明 | 不明 | HATとマトリクスLEDを接続し制御信号を送る。 |
| LANケーブル | 東日京三電線株式会社 | K-LAN-C5 | インターネットやLANへ接続する。 |
| microSDカード(16GB) | SanDisk | SDSQUNS-016G-GN3MN | Raspberry Piの記録メディア。 |

# **使用ソフトウェア**

本システムは以下のソフトウェアによって構成される。

## 3.1 Raspberry Pi OS

OSにはRaspberry Pi OSを採用した。DebianベースのOSでapt等のパッケーマネージャを採用している。使用したバージョンは10(Buster)である。

## 3.2 Node.js

Node.jsはサーバサイドでJavaScriptを実行する環境を提供するソフトウェアである。付属するパッケーマネージャnpm(Node Package Manager)を使用してモジュールを追加してシステムを構築することが可能で、本システムでは3.3のドライバモジュールをはじめとして、複数のモジュールを組み合わせることでLEDパネルを操作している。

## 3.3 node-rpi-rgb-led-matrix

Node.jsからLEDパネルを制御するドライバモジュールである。Raspberry Pi 3 Model Bまたは4にNode.js v8、10、13での動作が検証されている。本モジュールではLEDパネルの制御をLEDの座標と色を指定することでプログラムから行うことができる。また、BDF形式のフォントをサポートしており、テキストの表示も可能である。本システムの主要な機能であるテキスト表示機能はこのモジュールを使用した。

## 3.4 Vue.js

Webブラウザでのユーザインタフェースを構築するためのJavaScriptフレームワークである。SPA(Single Page Application)の作成が可能であるため、ページの遷移をすることのないWebアプリケーションを作成できる。また、コンポーネントごとにWebページ内の要素を独立して作成することが可能なため、コードの再利用などにおいて特に記述が容易になる。本システムのWeb側のフロントエンドはすべてVue.jsで作成した。

## 3.5 UFO

BDF(Bitmap Distribution Format)で作成されたフォントである。本システムでテキストをLEDパネルに表示する機能はこのフォントを採用している。BDFフォントはテキストファイルにビットマップの文字データを16進化して記録されている。

本研究で採用したUFOフォントはUnicodeで記述されたフォントである。これは3.3のモジュールに収録されているBDFフォントを表示する関数がUnicode(UTF)にしか対応しておらず、日本語対応のUnicodeで記述されたBDFフォントは他に類を見なかったためである。Unicodeの基本多言語面の文字をサポートしているため、英語、日本語のみならず、その他の外国語や一部の絵文字も表示可能である。

## 3.6 使用ソフトウェア一覧

表 3. 使用ソフトウェア一覧

|  |  |  |  |
| --- | --- | --- | --- |
| ソフトウェア名 | 開発者名 | バージョン | 用途 |
| Raspberry Pi OS | Raspberry Pi Foundation | 10 (Buster) | Raspberry PiのOS。 |
| Node.js | Ryan Dahl | v12.18.3 | システム全体を管理する他、開発にも付属のパッケージマネージャnpmを使用する。 |
| node-rpi-rgb-led-matrix | easybotics | 0.4.1 | マトリクスLEDのドライバ。 |
| Vue.js | Evan You | 15.1 | システムのフロントエンドを構築する。 |
| MariaDB | MariaDB Foundation | 10.3 | システムのバックエンドで履歴などを保存する。 |
| UFO | akahuku | 1.0.21 | マトリクスLEDに表示するテキストのフォント。 |

その他、本システムに使用しているNode.jsのプラグインに関しては、プロジェクトディレクトリ内のpackage.json[[1]](#footnote-1)を参照されたい。

# **配線**

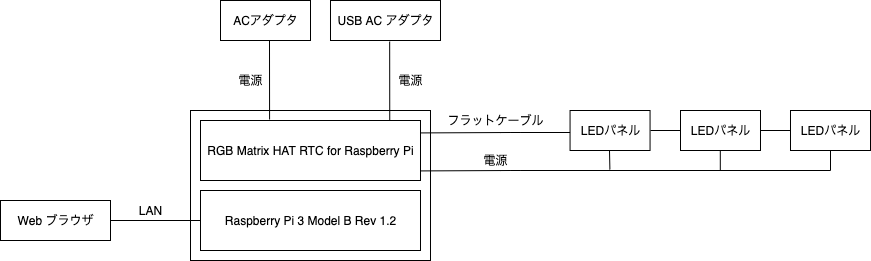
 本システムのハードウェアの配線について説明する。

図4. 配線図

本システムは、Raspberry Piを中心に図4のようにすべてのパーツが接続されている。Raspberry Piの上に接続されたRGB Matrix HAT + RTCからマトリクスLEDの制御用のフラットケーブルと電源用の4pinコネクタを接続している。マトリクスLED間はデイジーチェーンで接続され、今回は3枚接続している。

電源にはRaspberry PiとRGB Matrix HAT + RTCへ2本供給している。

セットアップ時にRaspberry Piへの入力を必要とするため、Raspberry PiにはUSBマウスとキーボードを接続した。セットアップ後はLANケーブル1本でネットワークを介して操作を行う。

# **セットアップ**

本システムにおける配線以降のセットアップに関しては付録のセットアップ手順を参照されたい。

# **BDFデコーダの作成**

テキストの表示にはBDFフォントを使用している。このフォントを利用する上でシステムの開発の前に、人間がフォントのソースファイルを解釈しやすくするためのデコーダを作成した。

## 6.1 BDFフォントの書式

BDFでは各文字はSTARTCHAR行からENDCHAR行までを1文字とし、その中のBITMAP行以下の16進数で記述された部分がビットマップフォントの各ピクセルを表現している。以下に例としてBDFでの「あ」(U+3042)の部分を記載する。

STARTCHAR U+3042

ENCODING 12354

SWIDTH 1000 0

DWIDTH 16 0

BBX 16 16 0 -2

BITMAP

0000

0000

0200

1FF0

0200

0240

0240

07F8

0C44

1482

2482

2302

2504

1808

0070

0000

ENDCHAR

## 6.2 BDFデコーダの作成

上のBITMAP以下16行の16進数を2進化し1となる部分を点灯している部分とみなすと、BDFフォントの文字の形状を復元できるが、これを人力で行うのは大きな負担となるので16進数を2進数に変換して文字として視認しやすい表示を行うソフトウェアを開発した。

これは本システムのプロジェクトディレクトリ内のtools/bdfdecoder.html、またはGitHub[[2]](#footnote-2)上で公開している。

図5にそのソフトウェアの動作画面を示す。

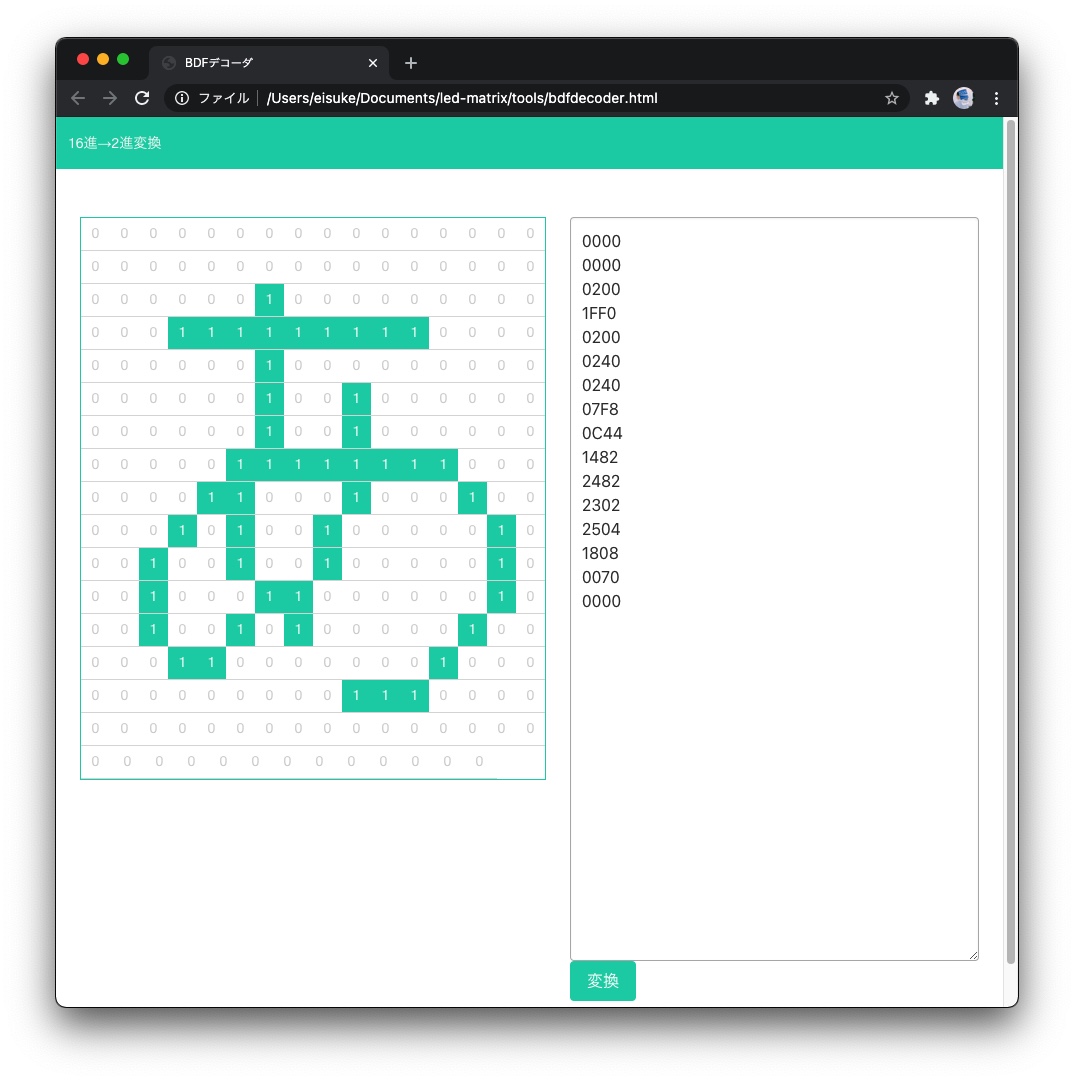
 図5左部が出力結果の表示エリアで右部が16進数のテキスト入力ボックスである。BDFから形状を調査したい文字のBITMAP部をコピーしこのソフトウェアに貼り付けて変換ボタンを押すと、左部のセルの色が変わり出力結果を確認することができる。で入力文字列の2進数への変換処理、各セルの色の変更などはすべてjQueryを使用しており、CSSにはBulmaフレームワークを採用している。各ライブラリはCDNを利用して読み込んでいるため、インターネットアクセスが必要ではあるものの、HTMLファイル単独での実行が可能である。

図5. BDFデコーダ

このソフトウェアを利用して、アルファベット、かな、漢字のみならず、その他の外国語や絵文字の形状についてシステム開発以前に簡単に調査することが可能となった。

# **システムの概要**

本システムは次の図6のような構成となっている。

Webシステムなどに見られるクライアントサーバモデルを採用し、クライアント(Webブラウザ)からのリクエストに応じてバックエンド(Raspberry Pi+Node.js)でLEDパネルの表示を制御している。

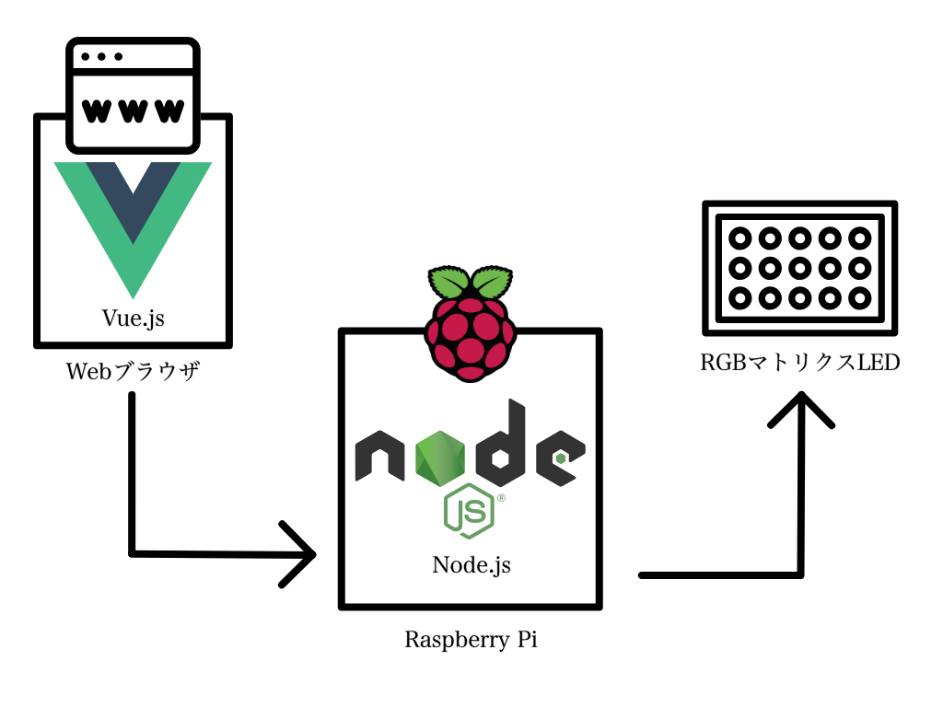


図6. システム構成

## バックエンドシステム

バックエンドではRaspberry Pi上にNode.js(上図中央)を基盤として、Node.jsのWebフレームワークのexpress上でAPIサーバを稼働させている。クライアントからサーバへのリクエストをトリガとしてNode.jsのLEDパネルドライバであるnode-rpi-rgb-led-matrix(easybotics)を通してLEDパネルを制御している。クライアントからサーバへテキストの表示をリクエストすると、その文字列をMariaDBに保存し、次回以降履歴一覧から参照し、繰り返し読み込むことを可能にしている。その他、定型文の登録やRSSのソースを登録するためにデータベースを利用している。

## フロントエンドシステム

フロントエンドではVue.jsを全面的に採用し、SPA(Single Page Application)によるクライアントソフトウェアを構築した。

ページ遷移時に再読み込みをせず、予め各ページのデータを読み込んでおいて必要に応じてAjaxなどで情報を更新するため、従来のWebアプリケーションと比較してネイティブのアプリケーションに近い体験を実現した。

開発においては、Vue.jsはコンポーネント指向であるため、要素毎に独立して開発を進められるためページ内の構造の見通しが立てやすい。

また、UIフレームワークにVuetifyを使用しているため、Material Designを外観に適用している。

# **機能**

本システムに実装している機能について解説する。

## テキスト表示

本システムの最も基本的な機能であるテキスト表示機能について解説する。

http://<Raspberry PiのIPアドレス>/textまたは画面左部のメニューから利用できる。

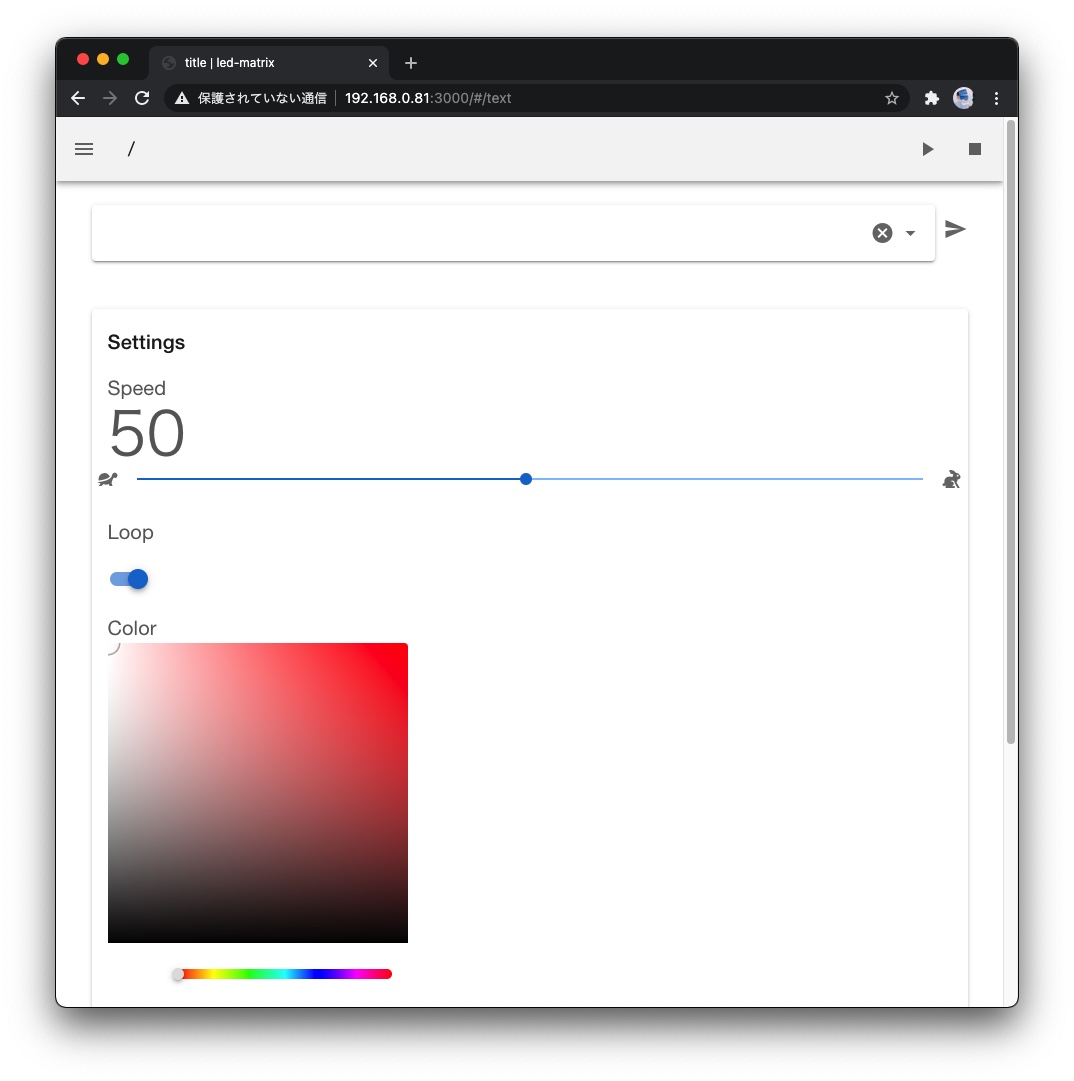
上記URLにアクセスした画面が図7である。

図 7. /text

* バックエンドシステム

バックエンドではテキストの表示する仕組みについてのプログラムを記述している。

図7の画面を操作してサーバに送られた文字列は、controllers/apiController.js内のexports.indexを呼び出してLEDパネルに表示される。

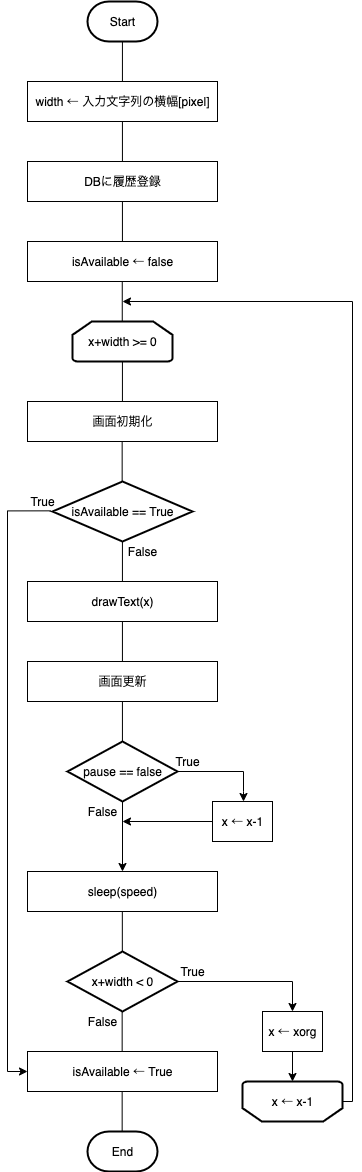
この関数では、以下の図8のフローチャートのような手順で文字の出力を行っている。

図8. テキスト表示処理フローチャート

図8はまず、入力された文字列をUFOフォントで表示した際に必要とする横幅のドット数をstringLength関数によって取得している。これはLEDパネルの右部から文字をスクロールする際に横幅を把握していないと表示終了のタイミングが不明なためである。単純にを横幅とすると、半角文字が混在している場合空白文字が文字列の終盤に大量に発生してしまい、表示のループに不自然な影響が出る。

次にinsertHistory関数によってデータベースに入力文字列を記録している。以後、クライアントからは履歴として呼び出して過去に入力した内容の再利用を可能にしている。

また、isAvailable関数でテキストの表示リクエストを新たに受け付けられるかのフラグを管理している。この変数をfalseとし、テキスト表示中は新たなテキストの受付ができないようにすることで、重複してリクエストを受け付けないようにしている。

文字列の表示の前に一度LEDパネルを初期化してからスクロールが始まる。

文字列はまず、x座標が最大の位置からx=0地点に向かって表示されるたびに次の表示の際に1ドットずれた位置で表示をしてスクロールされていく。文字列の先頭がx=0を超えるとxは負の値に向かって進み、表示範囲から見切れている部分を表示していく。

一度表示が終わるとLEDパネルを初期化してスピードで設定した値に応じて数ms停止した後1列隣りにずれて文字列が表示される。この操作を繰り返し行っている。

動作の途中でポーズのリクエストが送信されるとその位置でxの値の更新を一時停止し、ストップリクエストが送信されるとisAvailableをtrueとして強制的に全テキストの表示を停止してLEDパネルを初期化する。

* フロントエンドシステム

フロントエンドは、src/components/text/Text.vueにコンポーネントとして記述している。図7の上部のテキストボックスはVuetifyのv-autocompleteコンポーネントを使用し、空欄状態でクリックすると、図9のように履歴が展開される。

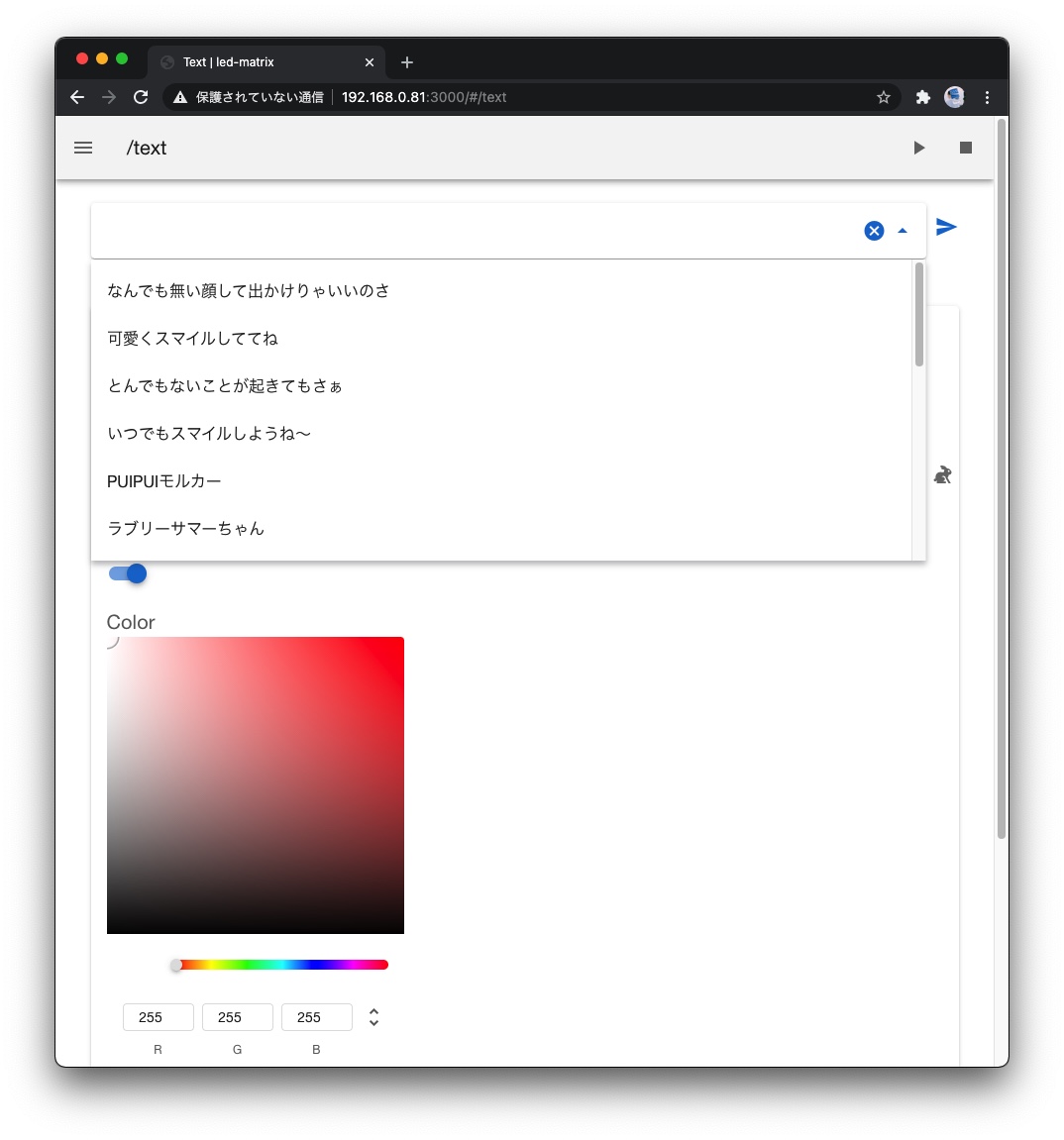
この履歴はVueのライフサイクルフックであるmounted内に記述しているため/textにアクセスされた時点でAjaxによってAPIサーバに問い合わせて履歴を含むjsonデータを読み込んでいる。

図 9. /text(履歴展開)

Settings内にはテキストの状態を変えるコンポーネントを設置した。

Speed以下のスライダはテキストのスクロール速度を示す。1-100の整数倍の100段階で制御可能で、ごとにLEDパネルの表示を一時停止しながらスピードの制御をしている。

Color以下のカラーパレットはv-color-pickerコンポーネントを使用し、任意の色を抽出するとLEDパネルの色が即座に変更される。

Speed、Color共にmounted内に記述しているので/textにアクセスした時点でのサーバの値がクライアントに反映されるほか、クライアント側で変更があった場合もwatchプロパティによりAPIサーバに変更のリクエストを送信している。

## 8.2. 定型文

8.1のテキスト機能を利用するには都度文字列を手作業で入力しなければならない。この手間を省略し、文字列を定型文として登録して再利用しやすくしたものが定型文機能である。

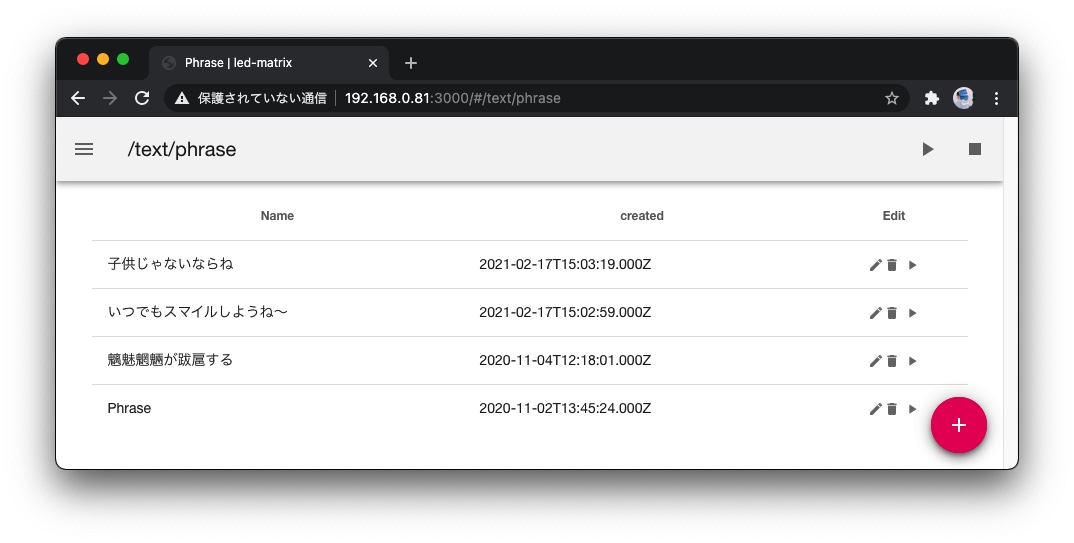
 http://<Raspbery PiのIPアドレス>/text/phraseまたは画面左部のメニューから利用できる。上記URLにアクセスした画面が図10である。

図10. /text/phrase

図10のテーブルに登録済みの定型文が表示され、各行のEdit列のボタンにより、編集、削除、再生を実行できる。

このページの内容もまたテキスト機能同様にアクセス時にmounted内の処理が実行されAPIサーバから登録済み定型文をjsonデータでダウンロードしている。

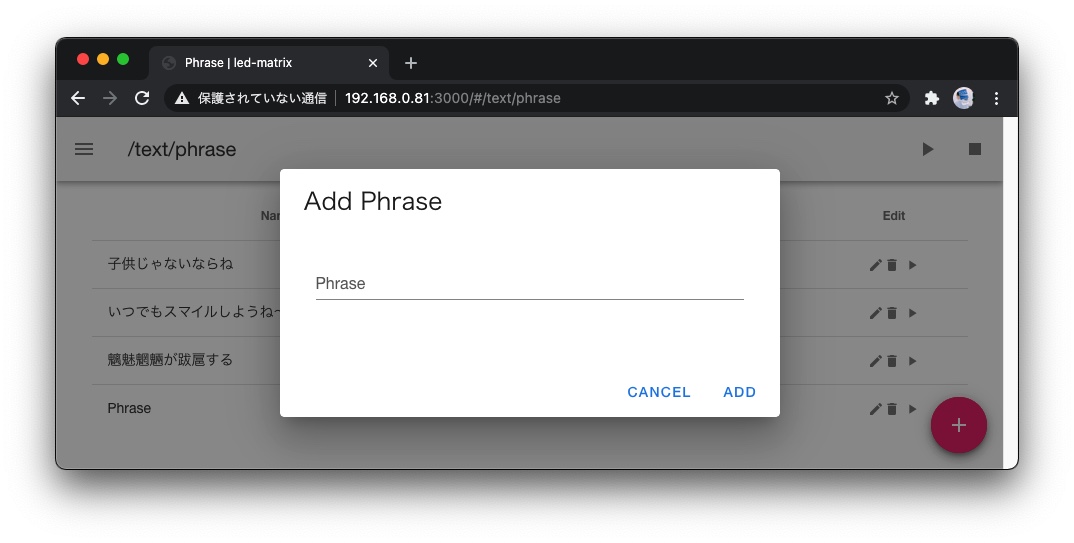
右下の+ボタンを押下すると、新規定型文登録メニューが出現する。以下の図11がそのメニューである。

図11. /text/phrase (新規定型文追加メニュー展開時)

図11のADDボタンを押下すると、APIサーバにリクエストが送信され、apiController.js内のaddPhrase関数が実行される。jsonで送信された入力文字列が関数内で展開され、MariaDBに登録されている。

## 8.3. RSS

インターネットからRSSによるニュース配信などを取得し、表示させる機能である。

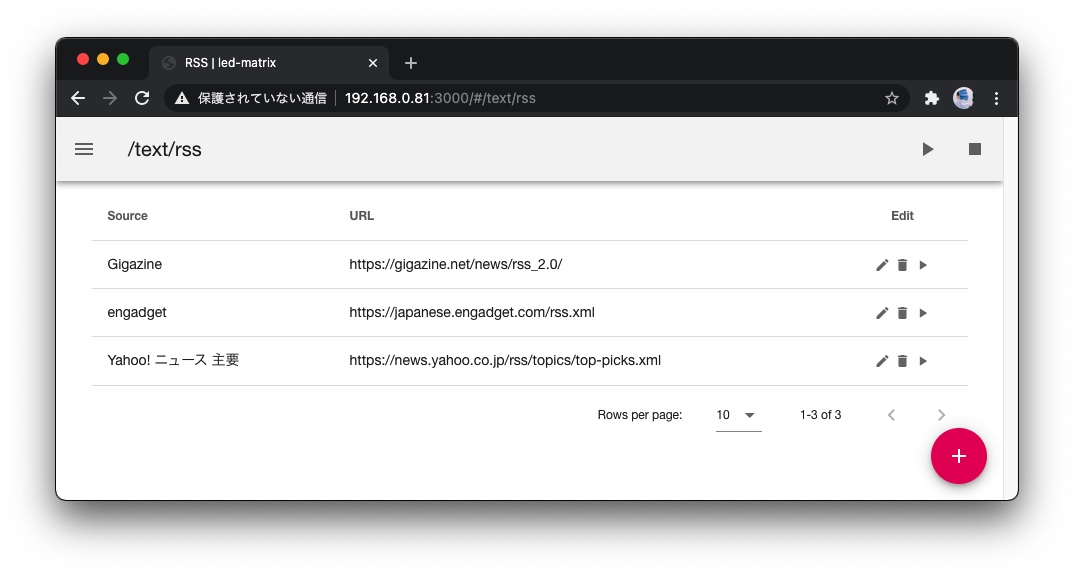
 http://<Raspberry PiのIPアドレス>/text/rssまたは画面左部のメニューから利用できる。上記URLにアクセスした画面が図12である。

図12. /text/rss

定型文機能と同様に登録済みデータが表示され、各業のデータに対して編集、削除、再生を実行できる。

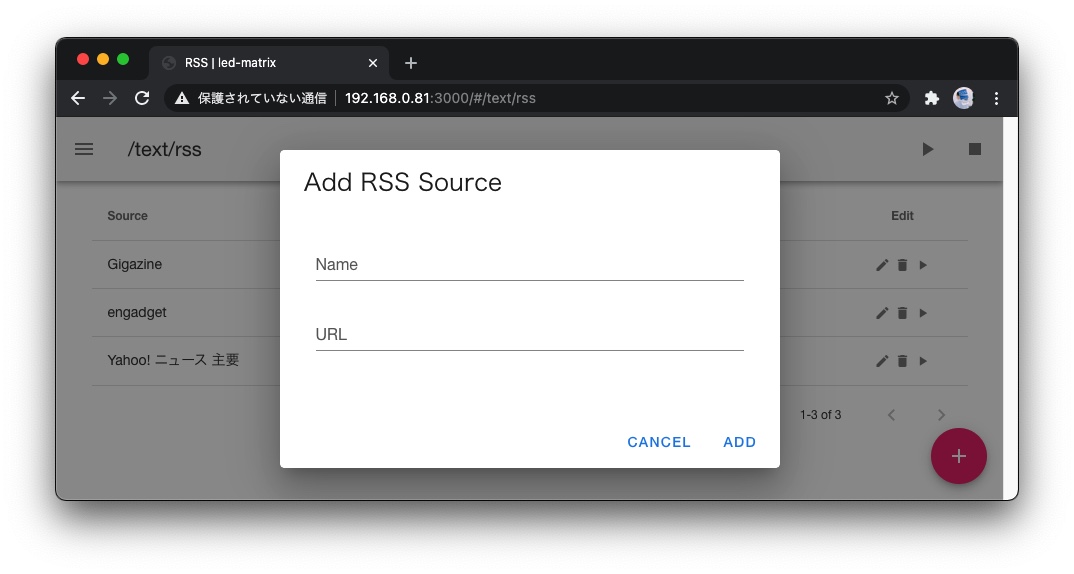
右下の+ボタンを押下すると、新規RSS登録メニューが出現する。以下の図13がそのメニューである。

図13. /text/rss (新規RSS追加メニュー展開時)

/text/rssからRSSの再生ボタンが押下された時、バックエンドでは、apiController.jsのshowNews関数が実行される。この関数では、文字列の表示の前にニュースのテキストをインターネットから取得している。RSSの取得にはcheerio-httpcliモジュールを使用した。RSSによって定義されたXMLでは、ニュースの見出しを取得するには各記事のtitleタグを抽出することで記事一覧の配列を生成できる。

## 8.4. 動作風景

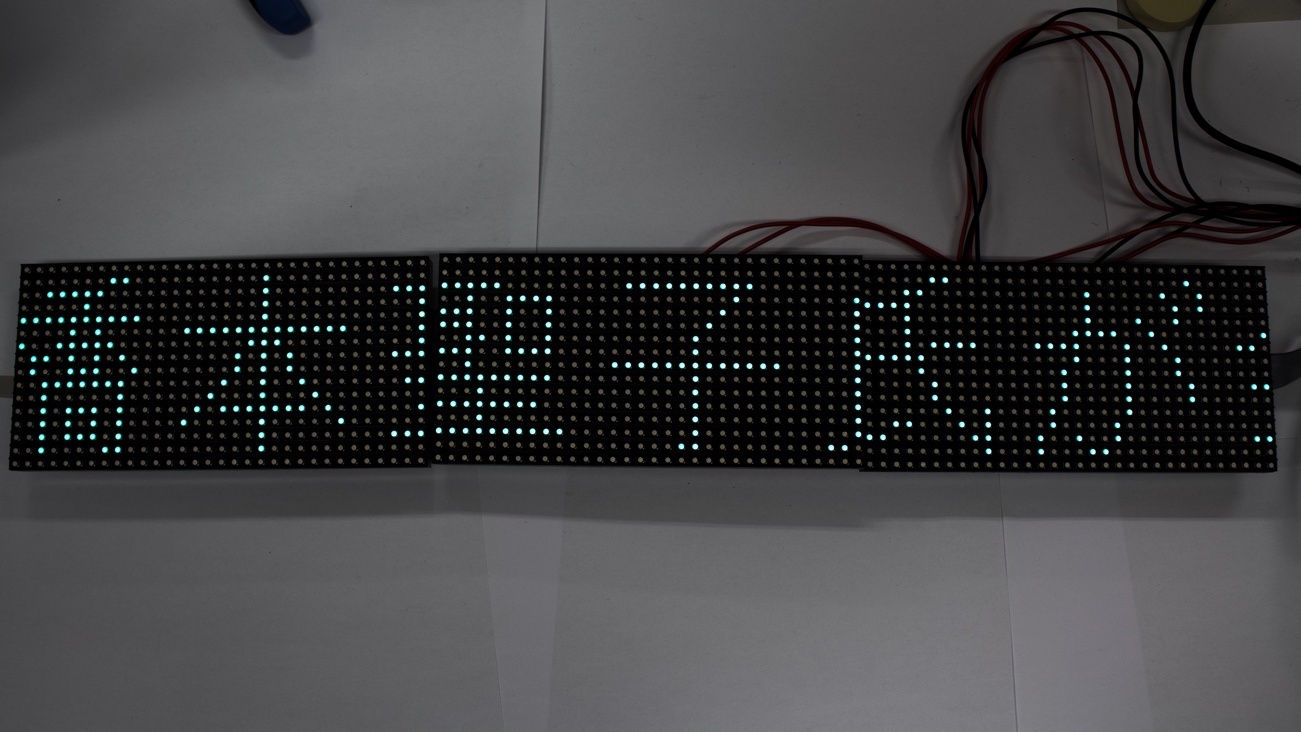
 以下の図14に動作風景を示す。

図14. 動作風景

# **システムのビルドと開発**

## 開発用サーバ

Vue.jsでのSPAは、JavaScriptモジュールバンドラのWebpackの関連機能であるwebpack-dev-serverを主に用いて開発した。これは、Webサーバを別に用意することなく開発できるようにHTMLを提供するサーバを構築するソフトウェアである。ライブ・リローディング機能でソースコードの変更を検知して自動でWebブラウザをリロードする機能を持つ。

プロジェクトディレクトリ内のpackage.json内のscriptsにあるdevの要素にwebpack-dev-server --inline --progress --config build/webpack.dev.conf.jsを記述しているため、開発用サーバは$ npm run devによって起動できる。開発用サーバが起動している間はhttp://<Raspberry PiのIPアドレス>:8080/で本システムが待ち受けている。この8080版ポートはconfig/index.jsのdev内のportで定義されているものである。しかし、開発用サーバから受け取ったフロントエンドからAPIサーバにリクエストを送ると、8080番ポートへリクエストしてしまう。これではポート番号が異なるため、正常にAPIを利用できないので、config/index.jsにproxyTableを追記し、/apiへのアクセスに関してすべて3000番ポートへ変換するように設定した。

## ビルド

Vue.jsで開発したシステムは本番環境へデプロイできるようにビルドする必要がある。ビルドはpackage.json内のscriptsにあるbuildの要素にnode build/build.jsを記述しているため、$ npm run buildでビルドが実行できる。ビルドされて出力されるファイルは、dist/内に保存される。

## 本番環境

ビルド後生成されたファイルの配信手続きとAPIサーバの起動スクリプトはserver.jsに記述されている。フロントエンドは9.2でビルドしたファイルへのルーティングとそのファイルを80番ポートで配信する設定を記述している。

## 永続化と自動起動

本システムは電光掲示板専用機としての利便性向上の観点から、Raspberry Piの電源を投入すれば、すぐに使い始められるように設定している。そのためにはシステムの自動起動と永続化が必要不可欠であり、これの実現のためにrc.localとforeverを使用した。

/etc/rc.localはRaspberry Pi OS起動時にこのファイルに記述されているスクリプトを自動的に実行する特定のファイルである。ここにserver.jsを永続化して起動するスクリプトを記述した。

foreverは、Node.jsのスクリプトを永続化可能にするプラグインである。Foreverによって起動されたNode.jsスクリプトはバックグラウンド処理されながら、異常終了時には自動再起動する。このプラグインよってserver.jsを起動する以下のようなスクリプトをrc.localに記述した。

forever start /home/pi/led-matrix/server.js

以上の手続きによって、自動起動する電光掲示板システムが構成されている。

# **課題**

* stringLength関数の実行時間

テキストの表示に関わるリクエストをNode.jsに送信すると都度受け取った文字列の横幅の長さをstringLength関数で計算している。横幅の長さは適切なスクロールの終了処理のために必要なものであるが、この値の取得に要する時間は文字列が長くなるほど増大している。

RSS機能により令和3年2月17日 15:28に以下の「Yahoo! ニュース・トピックス – 主要[[3]](#footnote-3)」を取得した。(114字)(空白含む)

丸高が妊娠 柿谷2児のパパに

ワクチン接種 日本はG7で最後

手術15回 ダウン症モデルの夢

京王プラザ 30泊21万円プラン

立花氏にNHK受信料支払い命令

森氏の後任 候補者一本化か

自民・白須賀氏 離党届を提出

東京で378人の感染確認

以上の文字列をクライアントから取得するリクエストを送信してからLEDパネルに表示されるまで、14.44[s]を要した。

ユーザからすると、送信ボタンを押下後しばらく操作に対するフィードバックがない状態で待ち続けることになり仕様を把握していないと故障と誤解される恐れがある。

これは入力された文字列から1文字ずつUFOフォントの文字幅を取得する時に、毎回フォントファイルへアクセスし、シーケンシャルに対象文字のSTARTCHAR行を探索しているためである。

対策として、フォントファイルへ都度アクセスすることなく文字幅を計算することが挙げられるが膨大なUFOの文字幅をすべての文字種で把握することは難しい。

* APIサーバの設計

本システムのAPIサーバのルーティングはroutes/api.jsに記述している。

このファイルにはクライアントからの各操作に対して一意なURLを与えることでリクエストを解決しているが、現在の仕様ではAPIに関する処理をすべてPOSTにしているためRESTful APIとしては望ましくない点が課題となっている。

現状でも問題はないが、本来であればRESTful APIのルールに則ってAPIを設計した方が開発に規律が生まれる。

GET、POSTの他にPUT、DELETEなどのメソッドを使い分けてAPIの構築を行うべきであった。

# **おわりに**

約1年間の研究期間のうち、前期は遠隔授業で自宅での研究が続いた。幸いにも在宅でも進めやすいテーマだったが、序盤はNode.jsの取り扱いにも苦戦し、Node.jsの役割を把握していなかったために様々な言語やプラットフォームを右往左往しながら活動したため苦労した。

しかしながら、後期以降はNode.js+Vue.jsというJavaScriptを基盤とした開発環境が確立し、JavaScriptでの開発スキルが大いに成長したと感じたほか、モダンなWeb開発環境に触れることが出来た。

フロントエンドの処理に詰めの甘さや画像や動画の再生機能の未実装が悔やまれるが、卒業研究としてある程度の規模のシステムの稼働まで実現することが出来たことには満足している。

本研究で開発したシステムのソースコードは以下のURLで公開している。

[https://github.com/xiexxa/led-matrix]

# **参考文献**

* Adafruit RGB Matrix HAT + RTC for Raspberry Pi - Mini Kit、Adafruit、[https://www.adafruit.com/product/2345]。
* D.6 BDF形式、富士通、[https://software.fujitsu.com/jp/manual/manualfiles/m200002/b1wd0741/14z200/b0741-d-06-00.html]。
* vue.jsのcreatedとmountedの違いを目で見て理解、REFFECT、[https://reffect.co.jp/vue/vue-js-created-mounted-diffrence]。
* RESTful APIとは何なのか、NagaokaKenichi、[https://qiita.com/NagaokaKenichi/items/0647c30ef596cedf4bf2]。
* Vueのwatchプロパティの書き方・使い方について解説、ELOOP、[https://www.e-loop.jp/knowledges/2/]。
* cheerio-httpcli - Node.js用WEBスクレイピングモジュール、ktty1220、[https://www.npmjs.com/package/cheerio-httpcli]。
* Unicode対応 文字コード表、ASH multimedia lab.、[http://ash.jp/code/unitbl21.htm]。
* Raspberry Piでプログラムを自動起動する5種類の方法を比較・解説、karaage0703、[https://qiita.com/karaage0703/items/ed18f318a1775b28eab4]。
* 【Node.js】foreverコマンドの基本的な使い方まとめ、taneyats、[https://www.taneyats.com/entry/nodejs-forever]。
* node-rpi-rgb-led-matrix、easybotics、[https://github.com/easybotics/node-rpi-rgb-led-matrix]。

1. package.json、[https://github.com/xiexxa/led-matrix/blob/dev/package.json] [↑](#footnote-ref-1)
2. BDFデコーダ、[https://github.com/xiexxa/led-matrix/blob/dev/tools/bdfdecoder.html] [↑](#footnote-ref-2)
3. Yahoo! ニュース・トピックス – 主要、[https://news.yahoo.co.jp/rss/topics/top-picks.xml] [↑](#footnote-ref-3)