プログラミング班

電光掲示板製作

３４番　元永　彬弘

３５番　モハマド　ナイム

３６番　八巻　勇介

担当教員　酒井先生

目　　次

1. はじめに …P２
2. 研究概要 …P２
3. 使用機器・材料 …P２
4. 開発環境 …P２
5. 研究内容 …P３
6. 研究成果 …P３
7. マニュアル …P 4
8. 感想 …P 17
9. 参考資料 …P 17
10. プログラム …P 18

１ はじめに

CHaserOnlineを終え、今後の研究課題として電光掲示板を作り、情報技術科で活用できればいいと思い製作する。

２ 研究概要

　自分の表示したい文字列を表示できる。また２列での表示や色の変更、文字の流れる速度などを調整できる。これらの操作は「USBメモリ」または「ブラウザ」から行うことができる。

３ 使用機材・材料

・64×32 RGB LED Matrix - 3mm pitch 2個

・Raspberry Pi 3 Model+ １個

・4A電源 １個

・USBメモリ １個

・アクリル板（９００ｍｍ×６１０ｍｍ） １枚

・ネジ（M２.６×２０ｍｍ） ４本

４ 開発環境

* Mac OS Mojave
* Debian 9
* Windows 10

使用した言語

・ PHP

・ Python

* JavaScript
* Redis

５ 研究内容

文字の表示

マトリックスディスプレイを操作するときは、” hzeller/rpi-rgb-led-matrix”のライブラリを使用する。もとはC言語で書かれたライブラリだが、Pythonから呼び出せるようにラッパーされている。これによってPythonで書かれたブログラムからでも、C言語で書かれたライブラリを呼び出すことができる。このライブラリには文字を表示する関数があり、これを利用することによって自由に文字を表示することが可能だが、日本語の表示ができなかった。そのため日本語を表示できるようにライブラリを書き換えた。また、この関数はフォントを指定することにより、任意のフォントで文字を表示することができる。フォントの指定には、BDF(ビットマップフォント)のみが可能であり、日本語のBDFを見つけることができなかったため、「jirutka/otf2bdf」というOpenType形式のフォントからBDFへ変換を行うプログラムを利用して、日本語フォントを用意した。

USBメモリからの読み込み

Raspberry PiはDebianというOSで動作させている。Linux系のOSなのでUSBメモリなどを接続した場合はディレクトリに「/dev/sd\*」として認識される。プログラムでは「/dev」より下の階層を定期的にチェックし存在している場合は、USBメモリから読み込まれるようになっている。

文字の格納から表示までのプロセス

文字列や設定などはRedisという「キーバリューストア」を利用する。ブラウザまたはUSBメモリから読み込んだ文字や設定をRedisへ格納する。ディスプレイを表示するプログラムはRedisの変更を検知することができるので、リアルタイムに文字や設定を書き換えることができる。

６ 研究成果

　今回の課題研究で制作した電光掲示板は、情報技術科にとどまらず、例として「事務室前に置き、願書を出しに来る中学生のための案内板とし活用しよう」などと、指導教員の酒井先生の提案が出た。これは情報技術科の宣伝になると同時に、行事を円滑に進められるという効果があると考えられる。また、各設定をUSBメモリから変更が可能なので操作をしやすく、長期的な使用が見込める。APIを実装しているため掲示板本体を各部屋に設置しても、APIを利用して集中管理を行うことができることや、ほかのサイトのRSSなどからニュース、天気予報、電車の運転情報などを取得して画面に表示するなど様々なことが可能だと思われる。

　７　マニュアル

**電光掲示板マニュアル**



**目次**

※注意・電源の位置 **P.5**

[1]操作画面の出し方 **P.6**

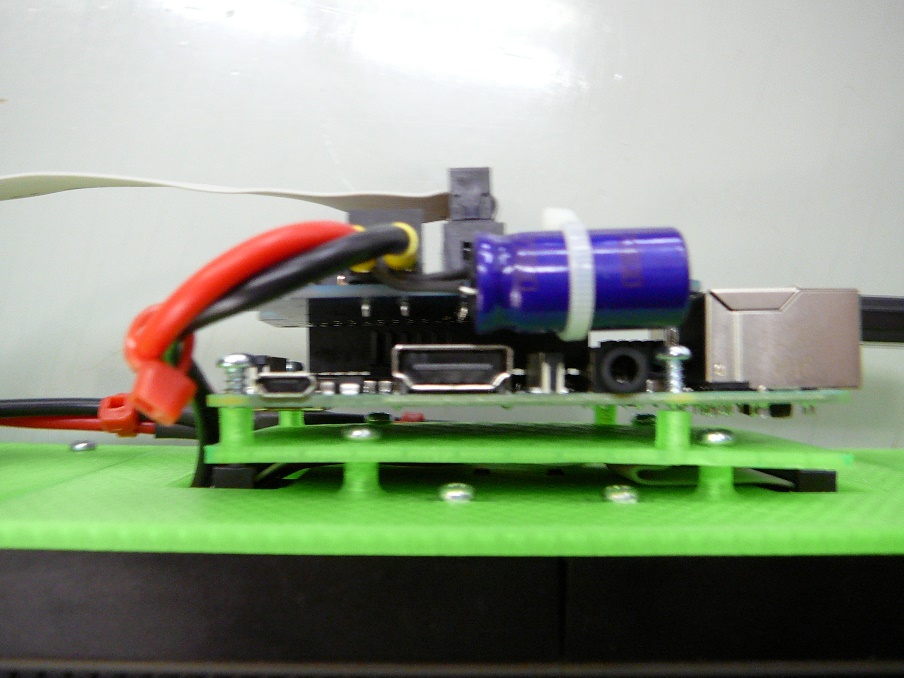
[2]操作について **P.9**

[3]USBからデータを読み込む場合 **P.12**

[4]USBドライブの初期化方法 **P.15**

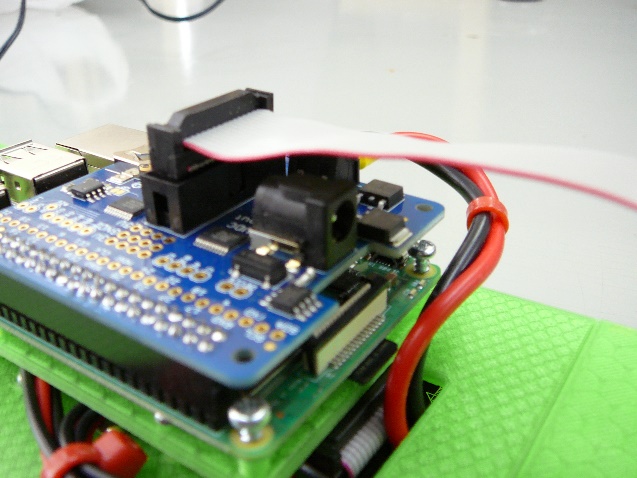
**注意**

**電源の取り扱いには十分に注意してください**



**接続しない！**

Raspberry Piに直接電源を接続しないでください。



**ここに接続**

白の枠線で囲まれた端子部分に電源を接続してください。

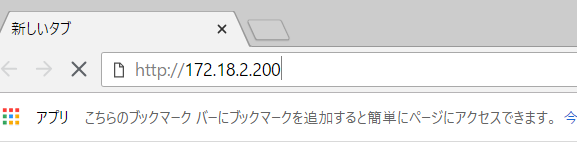
電源は、**5V4A**のものであれば大丈夫です。

**[1]操作画面の出し方**



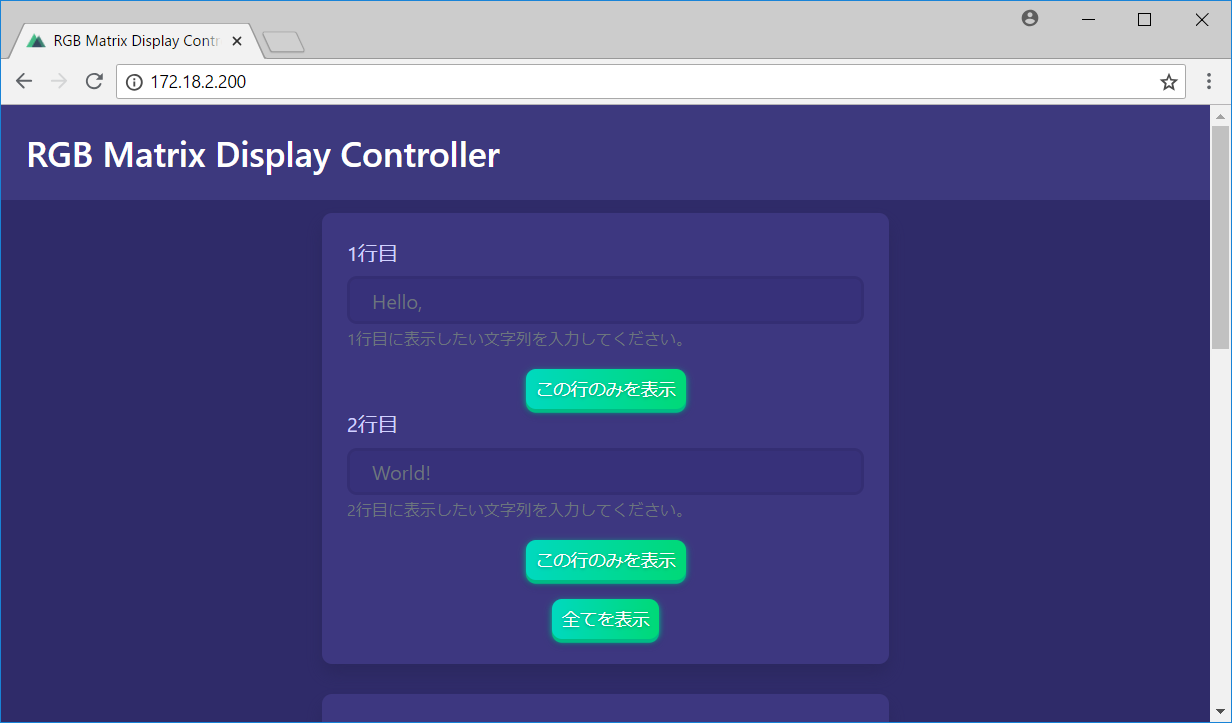
**ダブルクリック**

ブラウザを起動します。今回の例では、「Google Chrome」を起動します。



**IPアドレスを入力**

本体のIPアドレスにアクセスします。このとき本体のIPアドレスは「172.18.2.200」に設定されています。



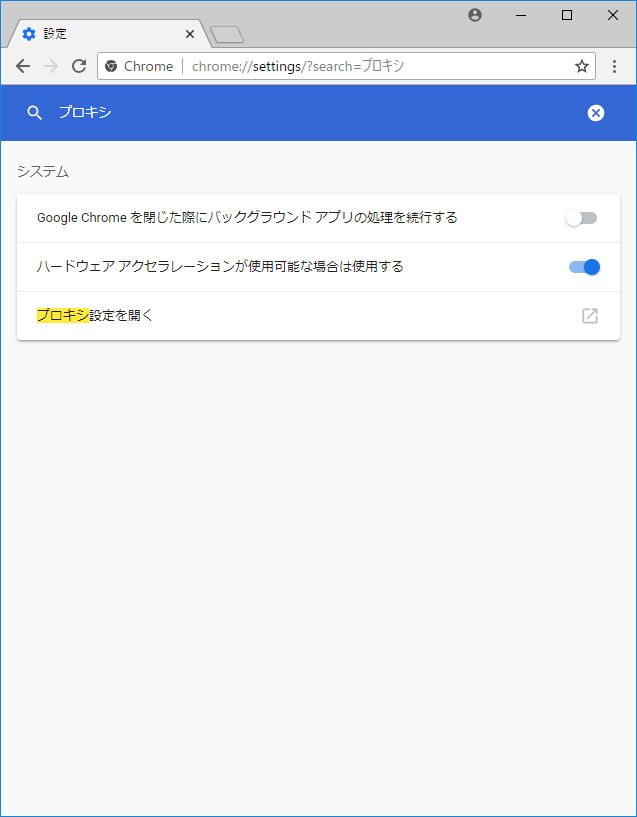
操作画面が表示されます。

※操作画面にアクセスできない場合



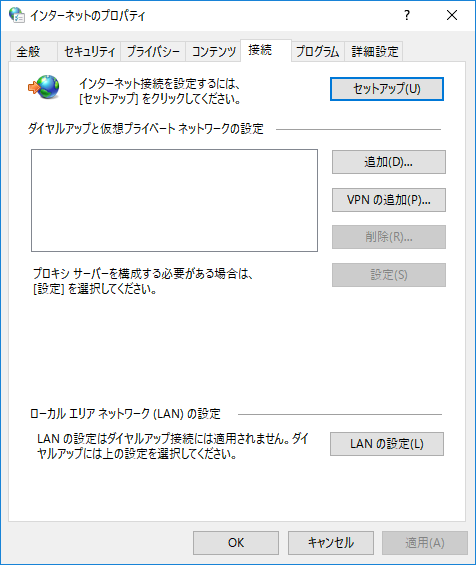
**クリック**

検索バーの右側にあるボタンから「設定」をクリックし、設定画面を開きます。



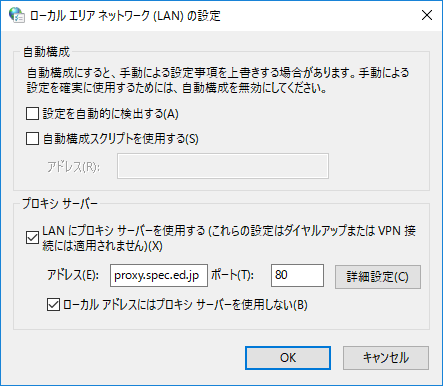
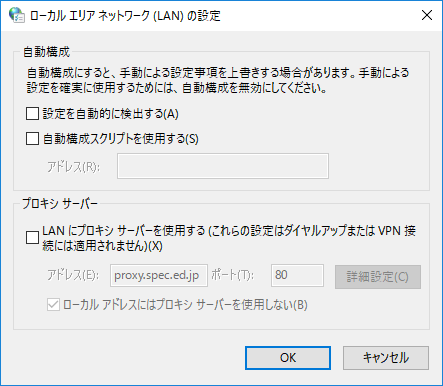
**クリック**

「プロキシ設定を開く」をクリックします。



**クリック**

「インターネットのプロパティ」が開きます。「LANの設定」をクリックします。

**チェックを外す**

**クリック**

「LANにプロキシサーバーを使用する(これらの設定はダイアルアップまたはVPN接続には提要されません)」のチェックを外します。チェックを外した後、「OK」をクリックして画面を閉じ、再度同じ手順を踏みます。

**[2]操作ついて**

操作画面の全体は以下のようになっています。

（１）

(2)

(3)

(1) P.10 文字の表示へ

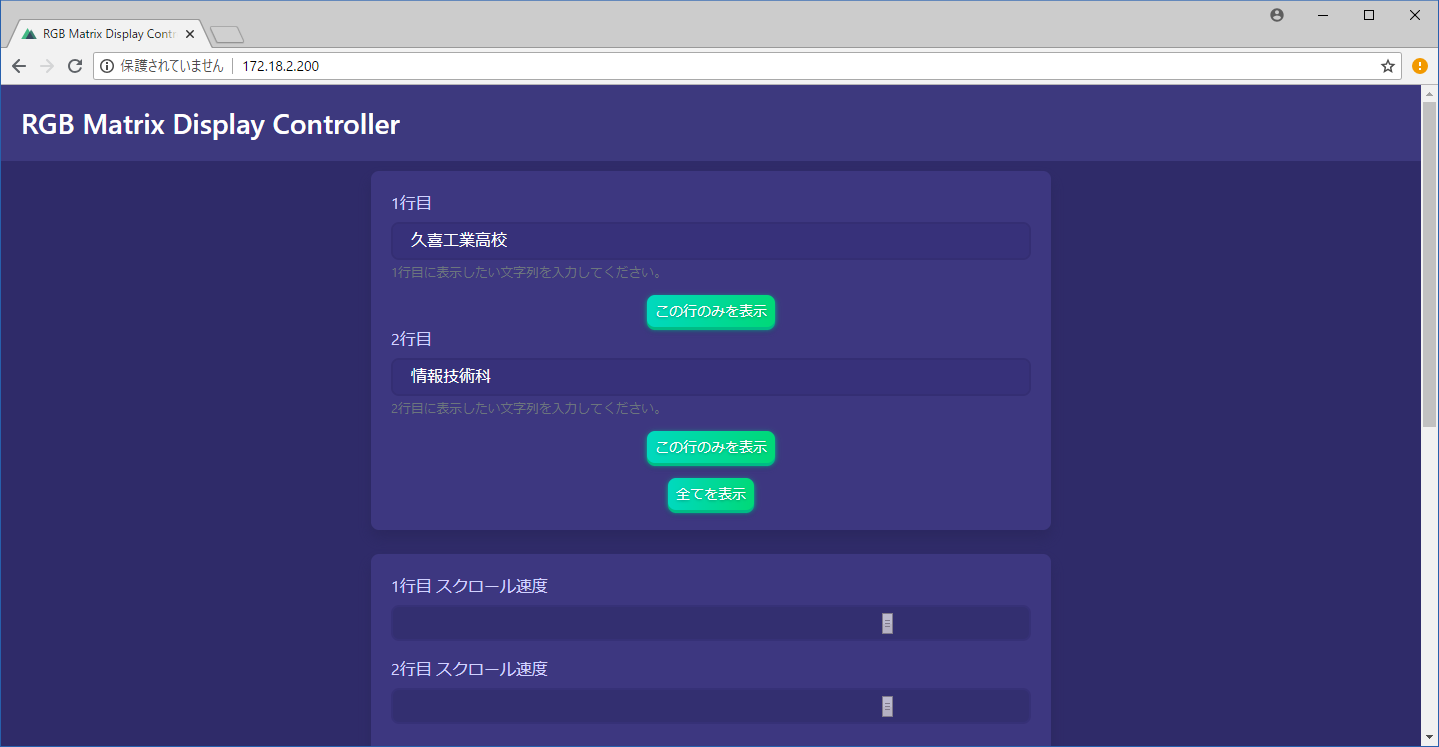
(2) P.11 スクロール速度へ

(3) P.11 文字色の変換へ

(1)文字の表示

起動直後は、「Initialized.」と表示されます。





**二行目の文字列**

**一行目の文字列**

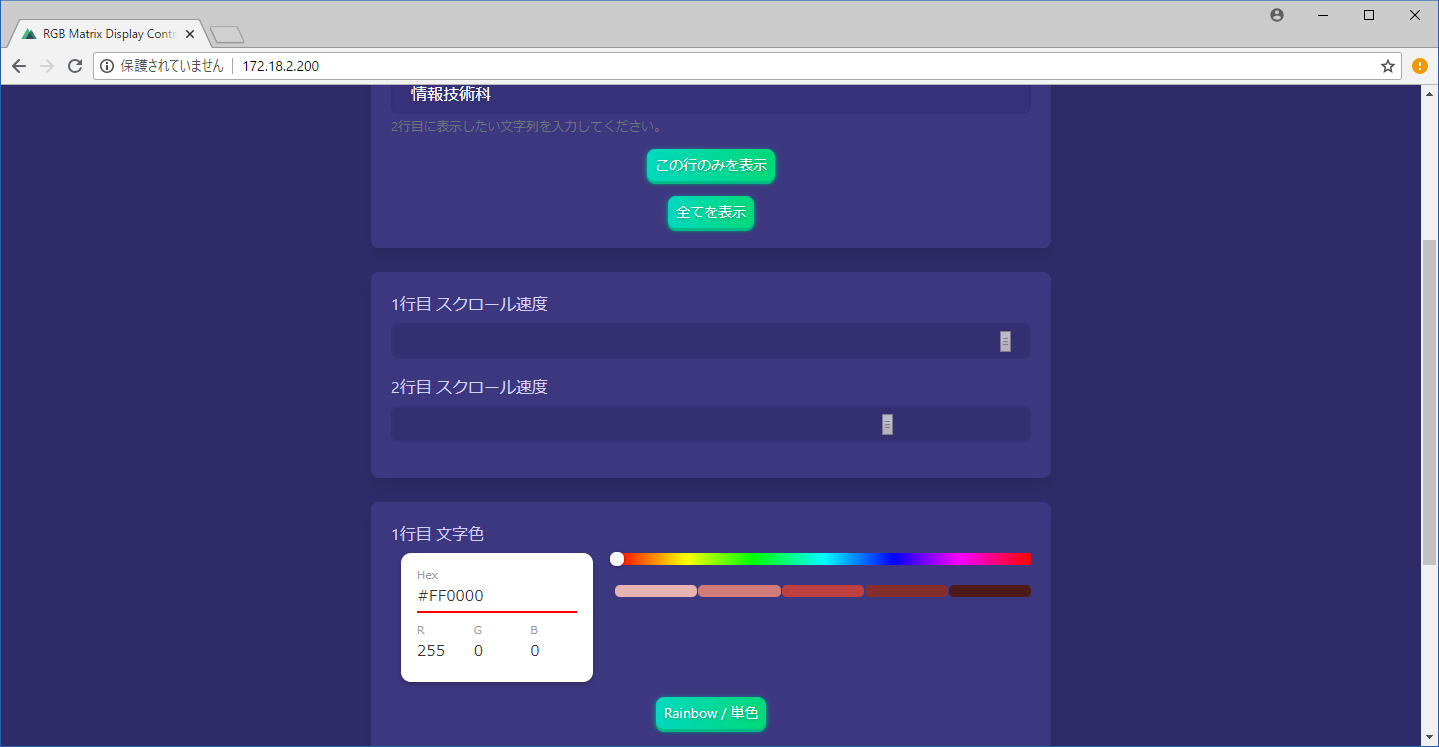
図1

図１には「一行目」、「二行目」、「この行のみを表示」、「すべてを表示」があり、図２は一行目に久喜工業高校、二行目に情報技術科と,入力したものになっています。



図２

(2)スクロール速度

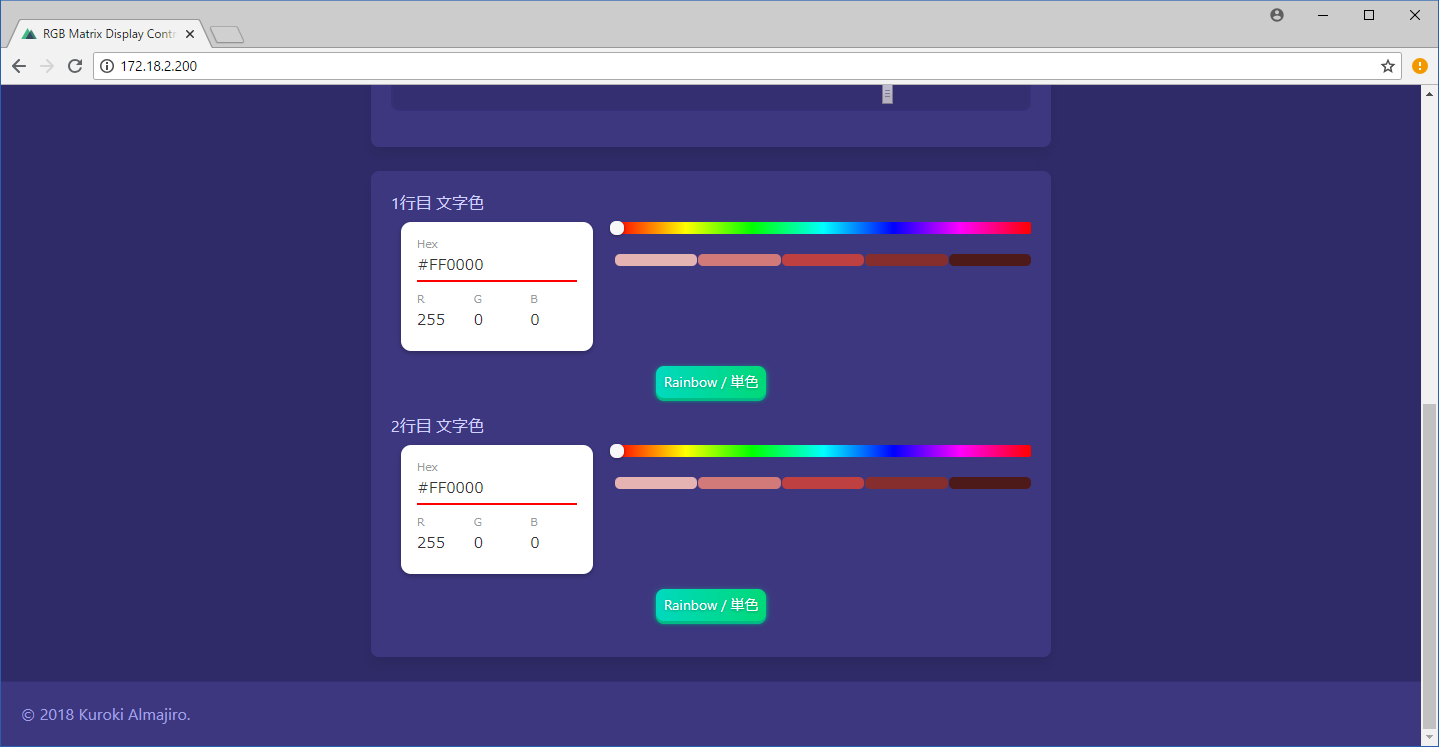


**ドラックで移動**

図３

図３には一行目、二行目のスクロール速度を変えるバーがあり右側に動かすほど早くなります。

(3)文字色の変換



**クリックで切り替え**

**ドラックで移動**

図４

図４では一行目と二行目の色を変えることができます。初期状態では虹色に設定されています。変えたい色の位置にバーを合わせ「Rainbow/単色」のボタンを押すことによって切り替えが可能です。

**[3]USBからデータを読み込む場合**

ルートフォルダ内に以下のファイルを追加することによって管理画面を利用せずに設定を行うことができます。

(1) message.txt

(2) speed.txt

**※USB端子であればどの位置に差しても構いません。**

(3) color.txt

(4) big.txt

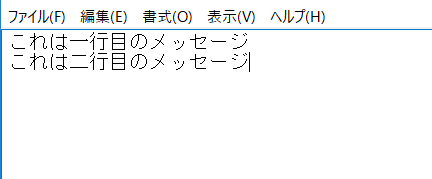
(5) scrollable.txt

(6) direction.txt

(1)message.txtについて

message.txtは表示したいメッセージを入力します。

**一行目の文字列**



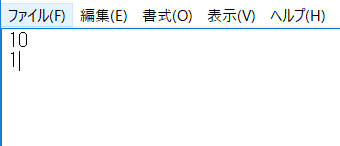
**二行目の文字列**

入力した行に対応して電光掲示板上では表示されます。

二行まで対応しています。

(2)speed.txtについて

speed.txtはメッセージのスクロール速度を入力します。

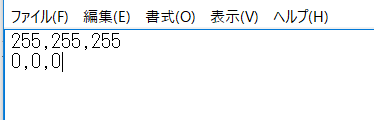


**二行目の速度**

**一行目の速度**

入力した行に対応して電光掲示板上ではスクロールされます。1から10までの数値が設定可能です。二行まで対応しています。速度は1が遅く、数が大きくなるつれて速くなります。

(3)color.txtについて



**青色**

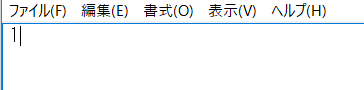
**緑色**

**赤色**

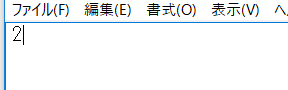
左からR（レッド）、G（グリーン）、B(ブルー)なっており、「,(コンマ)」で区切ってください。画像一行目のように行うと白色になり、二行目のように行うと虹色になります。二行まで対応しています。数値は0から255まで対応しています。

(4)big.txtについて

**一行目のみ表示**



一行目の文字を大きく表示したい場合



**二行目のみ表示**

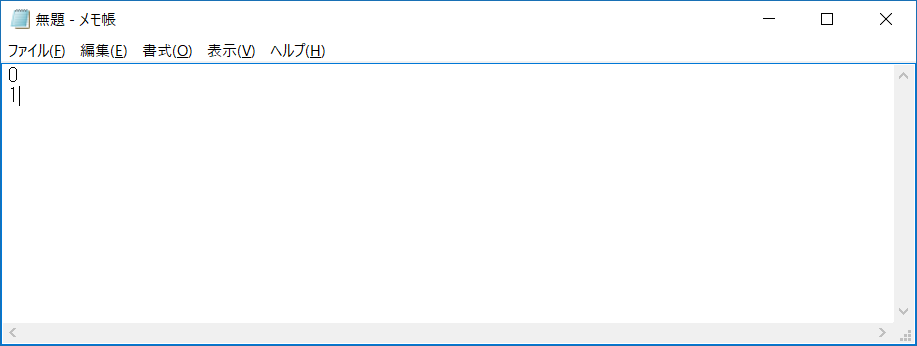
二行目の文字を大きく表示したい場合

一行目を大きく表示したい場合は画像のように１と入力し、二行目を大きく表示したい場合は２を入力してください。二行で表示したい場合は、そのディレクトリ内に「big.txt」を作らないでください。

(5)scrollable.txt について

この場合1行目は静止、

2行目はスクロールします。

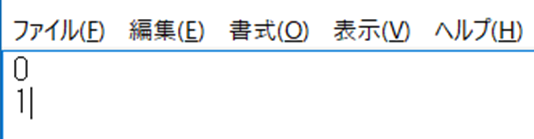


　scrollable.txtでは、スクロールの可否を設定することができます。スクロールさせたい場合は１を、スクロールさせたくない場合は０を、それぞれ対応した行に入力してください。

(6)direction.txt について

この場合1行目は左から右、

2行目は右から左へスクロールします。



direction.txtでは、スクロールの方向を設定することができます。右から左へスクロールさせたい場合は0を、左から右へスクロールさせたい場合は1を、それぞれ対応した行に入力してください。

**[4] USBドライブの初期化方法**

USBドライブから読み込む場合、一つ目のパーティションが参照されます。また一つ目のパーティションは、「FAT32」でフォーマットされている必要があります。

**Linux系のシステムで初期化する**

(1) USBドライブを接続します。

(2) 現在のUSBドライブのパーティションの確認、削除、作成を行います。

「fdisk」コマンドを利用します。ルート権限で実行する必要があるため

「sudofdisk デバイス名」と入力します。

Raspberry Piなどを利用している場合一つ目に接続されたUSBドライブは、

「/dev/sda」として認識されます。その場合は「sudo fdisk /dev/sda」と入力

します。

① パーティションの確認

「p」と入力するとパーティションテーブルが表示されます。

パーティションの一覧が出力された場合は、パーティションを削除してください。

② パーティションの削除

「d」と入力するとパーティションが削除されます。パーティションが一つの場合は自動的に削除されますが、複数ある場合は番号を指定してください。

　 ③ パーティションの作成

「n」を入力するとパーティションが作成することができます。2回続けて「Enterキー」を押してください。

「w」を入力し保存して終了します。

(3) 作成したパーティションを「FAT32」でフォーマットします。

「sudo mkfs.vfat –v –c –F 32 パーティション」を実行します。

Raspberry Piの場合は「sudo mkfs.vfat –v –c –F 32 /dev/sda1」と入力して実行します。

USBドライブの容量によってかかる時間は異なりますが結構かかります。

以上でUSBのフォーマットは完了です。

８ 感想

今回の課題研究の反省点として、チームとしての役割分担の難しさ、マニュアルの作成を通して相手にわかりやすく説明する難しさを学んだ。

分担作業では元永と八巻はアクリル板を切ることや、部品の買い出し、そして書類作成などでプログラム等の作業についていくことができなかった。当人の知識や技術が高ければ、負担をもっと分散できたと思った。

　　　マニュアルの作成では、知識のない方にどのような言い回しをするか、画像はどのように工夫し入れるかなど、とわかりやすく伝える難しさを知ることができた。

研究課題発表の際、質問に対し曖昧な答えを返すことしかできなかったため、情報共有をしっかり行い、どのような質問が来ても対応できるようにしておくべきだった。

今回の研究でPythonがなぜ注目される言語なのかを知ることができたと思う。ライブラリの使いやすさや、プログラムの構文が簡潔で分かりやすいなど、様々な発見があった。また、研究を行うまでC言語のライブラリがラッパーによって、ほかの言語で利用できることを知らなかった。

今回はRedisを利用したが、MySQL等を利用してもパフォーマンスが変わらなかったと思われる。インメモリで動作するので、高速なアクセスが可能な利点を生かそうと思ったが、そこまで高速なアクセスを行う場面がなかった。また、最初からUSBからの読み込みを予定していれば、Redisを利用しなくてもよかったかもしれない。

９ 参考資料

　・https://github.com/almajiro/matrix-display

　・https://github.com/jirutka/otf2bdf

　・https://github.com/hzeller/rpi-rgb-led-matrix

１０　プログラム

##############################################################################

# Matrix Display Daemon

#

# Author: Kuroki Almajiro

# Date: 2018/10/12

##############################################################################

from rgbmatrix import graphics, RGBMatrix, RGBMatrixOptions

from datetime import datetime

import time

import redis

import os

import glob

##############################################################################

# Parameters

##############################################################################

default\_font\_path = "./fonts/default.bdf"

light\_font\_path = "./fonts/light.bdf"

led\_rows = 32

led\_cols = 64

led\_chain = 2

led\_parallel = 1

led\_pwm\_bits = 11

led\_brightness = 100

led\_hardware\_mapping = 'adafruit-hat'

max\_row = 2

margin\_top = 14

margin\_top\_single = 24

mode = 0

mode\_row = 1

scrolling = {0: 1, 1: 1}

direction = {0: 1, 1: 0}

messages = {}

colors = {}

scroll\_speeds = {}

rainbows = {}

detect = True

before\_checked = datetime.now()

version = "2.3.8"

##############################################################################

# Initialize Display

##############################################################################

def initialize():

global options, matrix, canvas, store

global default\_font, light\_font

options = RGBMatrixOptions()

options.rows = led\_rows

options.cols = led\_cols

options.chain\_length = led\_chain

options.parallel = led\_parallel

options.brightness = led\_brightness

options.hardware\_mapping = led\_hardware\_mapping

options.disable\_hardware\_pulsing = True

options.pwm\_lsb\_nanoseconds = 130

# if you're not using raspberry pi 3 comment line below

options.gpio\_slowdown = 2

matrix = RGBMatrix(options=options)

canvas = matrix.CreateFrameCanvas()

default\_font = graphics.Font()

default\_font.LoadFont(default\_font\_path)

light\_font = graphics.Font()

light\_font.LoadFont(light\_font\_path)

store = redis.StrictRedis(host='localhost', port=6379, db=0)

##############################################################################

# Message setter/getter

##############################################################################

def get\_message(row=1):

return store.get('message'+str(row)).decode('utf-8')

def get\_messages():

for i in range(max\_row):

messages[i] = get\_message(i+1)

def set\_message(row=1, message=''):

store.set('message'+str(row), message)

##############################################################################

# Color setter/getter

##############################################################################

def get\_color(row=1):

red = int(store.zscore('message'+str(row)+'\_color', 'red'))

green = int(store.zscore('message'+str(row)+'\_color', 'green'))

blue = int(store.zscore('message'+str(row)+'\_color', 'blue'))

return {'red': red, 'green': green, 'blue': blue}

def get\_colors():

for i in range(max\_row):

\_color = get\_color(i+1)

colors[i] = graphics.Color(\_color['red'], \_color['green'], \_color['blue'])

def set\_color(row=1, value=0, color='red'):

store.zadd('message'+str(row)+'\_color', int(value), color)

def get\_rainbow(row=1):

return int(store.get('message'+str(row)+'\_rainbow'))

def get\_rainbows():

for i in range(max\_row):

rainbows[i] = get\_rainbow(i+1)

def set\_rainbow(row=1, status=0):

store.set('message'+str(row)+'\_rainbow', status)

##############################################################################

# Scroll Speed setter/getter

##############################################################################

def get\_scroll\_speed(row=1):

return int(store.get('message'+str(row)+'\_scroll\_speed').decode('utf-8'))

def get\_scroll\_speeds():

for i in range(max\_row):

scroll\_speeds[i] = get\_scroll\_speed(i+1)

def set\_scroll\_speed(row=1, speed=4):

store.set('message'+str(row)+'\_scroll\_speed', speed)

##############################################################################

# Common Functions

##############################################################################

def set\_standby():

set\_message(1, 'Initialized.')

set\_message(2, 'Initialized.')

#for i in range(2, max\_row+1):

# set\_message(i, '')

for i in range(1, max\_row+1):

set\_scroll\_speed(i, 4)

set\_color(i, 255, 'red')

set\_color(i, 0, 'green')

set\_color(i, 0, 'blue')

set\_rainbow(i, 1)

store.set('changed', 0)

store.set('type', 0)

store.set('mode', mode)

store.set('row', mode\_row)

def get\_display\_parameters():

global mode, mode\_row

for i in range(max\_row):

get\_colors()

get\_scroll\_speeds()

get\_rainbows()

mode = int(store.get('mode'))

mode\_row = int(store.get('row'))

def detect\_usb():

global detect, before\_checked

diff = datetime.now() - before\_checked

if diff.seconds >= 2:

if glob.glob('/dev/sd\*1'):

if detect != True:

detect = True

print("USB Connected !!!!!!")

check\_usb()

store.set('type', 1)

store.set('changed', 1)

else:

if detect == True:

detect = False

print("USB not Disconnected !!!")

before\_checked = datetime.now()

def check():

status = int(store.get('changed'))

if status == True:

store.set('changed', 0)

return True

return False

def check\_type():

get\_type = int(store.get('type'))

return get\_type

def rainbow(color):

if color[0] == 255 and color[2] == 0:

color[1] += 1

if color[1] == 255 and color[2] == 0:

color[0] -= 1

if color[1] == 255 and color[0] == 0:

color[2] += 1

if color[2] == 255 and color[0] == 0:

color[1] -= 1

if color[2] == 255 and color[1] == 0:

color[0] += 1

if color[0] == 255 and color[1] == 0:

color[2] -= 1

return color

def usleep(value):

time.sleep(value / 1000000.0)

def animate():

global canvas, matrix

for times in range(3):

for color\_code in range(256):

if times == 0:

color = graphics.Color(color\_code, 0, 0)

if times == 1:

color = graphics.Color(0, color\_code, 0)

if times == 2:

color = graphics.Color(0, 0, color\_code)

for i in range(led\_rows):

graphics.DrawLine(canvas, 0, i, led\_cols\*led\_chain, i, color)

time.sleep(0.001)

canvas = matrix.SwapOnVSync(canvas)

for color\_code in range(255, -1, -1):

if times == 0:

color = graphics.Color(color\_code, 0, 0)

if times == 1:

color = graphics.Color(0, color\_code, 0)

if times == 2:

color = graphics.Color(0, 0, color\_code)

for i in range(led\_rows):

graphics.DrawLine(canvas, 0, i, led\_cols\*led\_chain, i, color)

time.sleep(0.001)

canvas = matrix.SwapOnVSync(canvas)

def check\_usb():

files = glob.glob('/dev/sd\*1')

if files:

print('USB Drive detected')

print('Mounting device')

os.system('sudo mount ' + files[0] + ' /home/mnt')

print('Device successfully mounted!')

if os.path.exists('/home/mnt/message.txt'):

print('Set Messages from USB')

f = open('/home/mnt/message.txt', 'r')

for i in range(max\_row):

set\_message(i+1, f.readline().strip())

f.close()

if os.path.exists('/home/mnt/speed.txt'):

print('Set Speeds from USB')

f = open('/home/mnt/speed.txt')

\_\_speeds = f.read().replace('\r', '').split('\n')

f.close()

for i in range(len(\_\_speeds)):

set\_scroll\_speed(i+1, \_\_speeds[i])

if os.path.exists('/home/mnt/color.txt'):

print('Set Colors from USB')

f = open('/home/mnt/color.txt')

\_\_colors = f.read().replace('\r', '').split('\n')

f.close()

for i in range(len(\_\_colors)):

if \_\_colors[i] != '':

set\_rainbow(i+1, 0)

\_\_colors\_separated = \_\_colors[i].split(',');

set\_color(i+1, \_\_colors\_separated[0], 'red')

set\_color(i+1, \_\_colors\_separated[1], 'green')

set\_color(i+1, \_\_colors\_separated[2], 'blue')

if \_\_colors\_separated[0] + \_\_colors\_separated[1] + \_\_colors\_separated[2] == '000':

set\_rainbow(i+1, 1)

if os.path.exists('/home/mnt/big.txt'):

global mode, mode\_row

print ('Set Big Font')

f = open('/home/mnt/big.txt')

store.set('mode', 1)

store.set('row', int(f.readline()))

f.close()

else:

store.set('mode', 0)

if os.path.exists('/home/mnt/direction.txt'):

print('Set Direction')

f = open('/home/mnt/direction.txt')

for i in range(max\_row):

direction[i] = int(f.readline().strip())

f.close()

if os.path.exists('/home/mnt/scrollable.txt'):

print('Set Scrollable')

f = open('/home/mnt/scrollable.txt')

for i in range(max\_row):

scrolling[i] = int(f.readline().strip())

f.close()

print("Unmounting device")

os.system('sudo umount /home/mnt')

before\_checked = datetime.now()

return

##############################################################################

# Main

##############################################################################

if \_\_name\_\_ == '\_\_main\_\_':

print('Matrix Display Daemon')

print('Version: '+version)

print('-------------------------')

print('Initializing....')

initialize()

#animate()

set\_standby()

check\_usb()

print('Display initialized.')

try:

print('<Ctrl+C> to Exit')

while True:

get\_messages()

get\_display\_parameters()

if mode == 0:

lengths = {}

positions ={}

counters = {}

rainbow\_colors = {}

rainbow\_counters = {}

for i in range(max\_row):

if not direction[i]:

positions[i] = canvas.width

else:

messages[i] = messages[i][::-1]

positions[i] = -len(messages[i])

counters[i] = 0

rainbow\_counters[i] = 0

rainbow\_colors[i] = [255, 0, 0]

strip\_color = [255, 0, 0]

while True:

if check():

if check\_type():

break

else:

get\_display\_parameters()

canvas.Clear()

detect\_usb()

strip\_color = rainbow(strip\_color)

for i in range(led\_cols \* led\_chain):

canvas.SetPixel(i, margin\_top + 2, strip\_color[0], strip\_color[1], strip\_color[2])

for i in range(max\_row):

if not scrolling[i]:

positions[i] = 0

if rainbows[i] == 1:

rainbow\_counters[i] += 1

if rainbow\_counters[i] == 1:

rainbow\_colors[i] = rainbow(rainbow\_colors[i])

rainbow\_counters[i] = 0

color = graphics.Color(rainbow\_colors[i][0], rainbow\_colors[i][2], rainbow\_colors[i][1])

else:

color = colors[i]

lengths[i] = graphics.DrawText(canvas, default\_font, positions[i], margin\_top \* (i+1), color, messages[i])

if (scroll\_speeds[i] < counters[i]) and scrolling[i]:

if direction[i]:

positions[i] += 1

else:

positions[i] -= 1

counters[i] = 0

if (positions[i] + lengths[i] < 0) and not direction[i]:

positions[i] = canvas.width

if direction[i] and (positions[i] + lengths[i] > canvas.width + lengths[i]):

positions[i] = -lengths[i]

counters[i] += 1

usleep(0.001)

canvas = matrix.SwapOnVSync(canvas)

else:

if not scrolling[mode\_row-1]:

position = 0

else:

position = canvas.width

if direction[mode\_row-1]:

messages[mode\_row-1] = messages[mode\_row-1][::-1]

counter = 0

rainbow\_color = [255, 0 ,0]

rainbow\_counter = 0

while True:

if check():

if check\_type():

break

else:

get\_display\_parameters()

canvas.Clear()

detect\_usb()

if rainbows[mode\_row-1] == 1:

rainbow\_counter += 1

if rainbow\_counter == 1:

rainbow\_color = rainbow(rainbow\_color)

rainbow\_counter = 0

color = graphics.Color(rainbow\_color[0], rainbow\_color[2], rainbow\_color[1])

else:

color = colors[mode\_row-1]

length = graphics.DrawText(canvas, light\_font, position, margin\_top\_single, color, messages[mode\_row-1])

if (scroll\_speeds[mode\_row-1] < counter) and scrolling[mode\_row-1]:

if direction[mode\_row-1]:

position += 1

else:

position -= 1

if (position + length < 0) and not direction[mode\_row-1]:

position = canvas.width

if direction[mode\_row-1] and (position + length > canvas.width + length):

position = -length

counter += 1

time.sleep(0.001)

canvas = matrix.SwapOnVSync(canvas)

except KeyboardInterrupt:

matrix.Clear()

print('See you next time!')