

# 卒業論文

## LEDマトリクスパネルによる異なる点 滅周波数を用いた可視光通信の実験

An experiment on visible light communication  
using different blinking frequencies with an LED  
matrix panel.

園邊 翔大

立命館大学理工学部  
電子情報工学科

2026年3月



## 内 容 梗 概

本論文では、LED マトリクスパネルを用いた可視光通信の送信部システムの確立を目的とし、表示技術の基礎構築から通信方式の検討までを段階的に行った。まず、HUB75 インタフェースの特性を踏まえ、本研究で採用した点灯方式により 256 階調で画像を表示するシステムを構築した。また、Django と Cloudflare Tunnel を用いて、遠隔地から画像をアップロードし Raspberry Pi で処理した後に表示する Web 連携システムを実装した。さらに、ガンマ補正を導入し、LED マトリクス特有の淡い表示を改善して視覚的品質を向上させた。

可視光通信では、1枚のパネルを4領域に分割し、各領域で同一データを並列送信する方式を採用した。点灯パターンは、時間軸を4区間に分けたときの点灯が2回となる2種類のパターン（1010を1, 1100を0）とし、平均輝度を一致させることで人間の視覚に違和感を与えない設計とした。全体の駆動周波数を480Hz とすることで、ちらつきを抑えつつローリングシャッター由来の縞パターンからビットを推定可能であることを確認した。

提案方式は平均輝度を一致させる設計により、デューティ比に差がある旧方式と比べて色味差やちらつきの低減が期待される。一方で、平均輝度が一致するため受信側は縞パターン抽出と位相推定に依存し、撮影条件に対する頑健性が今後の課題となる。今後は受信アルゴリズムを含めた統合システムの実装と、距離・角度・露光条件に対する通信性能評価を進め、誤り訂正や分割数の最適化を検討する必要がある。



# 目 次

<b>内容梗概</b>	i
<b>第1章 序論</b>	1
1.1 研究背景 . . . . .	1
1.2 本研究の目的 . . . . .	2
1.3 本論文の構成 . . . . .	2
<b>第2章 関連研究</b>	5
2.1 可視光通信の概要と応用 . . . . .	5
2.2 カメラを用いた可視光通信 . . . . .	5
2.3 LED 表示とちらつきの制御 . . . . .	6
2.4 本研究の位置づけ . . . . .	6
<b>第3章 LED マトリクスパネルを用いた画像表示システムの構築</b>	9
3.1 使用デバイスと開発環境 . . . . .	9
3.1.1 HUB75 64 × 32 LED マトリクスの仕様 . . . . .	9
3.1.2 Raspberry Pi 4 と開発環境 . . . . .	12
3.2 256 階調表示システム . . . . .	15
3.2.1 本研究で採用した方式による 256 階調表現 . . . . .	15
3.2.2 一般的な点灯制御方式との比較 . . . . .	16
3.2.3 点灯させるタイミングの計算方法 . . . . .	17
3.2.4 32 × 64 画像へのリサイズとマッピング処理 . . . . .	23
3.3 ガンマ補正による画質改善 . . . . .	25
3.3.1 ガンマ補正 . . . . .	25
3.4 画像アップロードシステムとネットワーク構成 . . . . .	29
3.4.1 画像処理や LED マトリクスパネル制御用の実行ファイル . .	29
3.4.2 Web アプリケーションの構成 . . . . .	30
3.4.3 Cloudflare Tunnel による外部アクセス . . . . .	31
3.4.4 Web サイトの構築 . . . . .	32
3.4.5 Web サイトからの画像アップロード～パネル表示までの処理フロー . . . . .	33

---

<b>第4章 2つの異なる点滅周波数を用いた可視光通信方式</b>	<b>35</b>
4.1 提案方式の概要	35
4.2 可視光通信の仕組み	35
4.2.1 点灯パターンの設計	35
4.2.2 バイナリ変換の方法	37
4.2.3 撮影画像における縞模様	38
4.2.4 分割領域における点灯パターンの設計	39
4.2.5 4分割以上の検証	41
4.3 フレーム構成と復号の設計	41
4.3.1 送信フレームの構成	41
4.3.2 FLAG 検出とシンボル境界推定	42
4.3.3 受信アルゴリズム	42
4.3.4 復号フロー全体	43
<b>第5章 考察および今後の課題</b>	<b>45</b>
5.1 考察	45
5.2 結論および今後の課題	46
<b>参考文献</b>	<b>46</b>
<b>付録A LEDマトリクス表示用プログラム</b>	<b>49</b>
<b>謝辞</b>	<b>55</b>
<b>発表論文リスト</b>	<b>57</b>

# 図 目 次

3.1	HUB75 64 × 32 LED マトリクスパネルの外観 <sup>1</sup>	11
3.2	HUB75 64 × 32 LED マトリクスパネルの点灯例 <sup>2</sup>	12
3.3	Raspberry Pi 4 Model B の外観 <sup>3</sup>	13
3.4	本研究で採用した方式による 256 階調表現	15
3.5	PWM 制御による点灯制御	16
3.6	スロットの均等に配置しないパターン	18
3.7	$R = 200$ のときの累積値の遷移	19
3.8	LED マトリクスパネルに表示した元画像	21
3.9	整数除算でスロット配置を決める方式の場合 1	22
3.10	整数除算でスロット配置を決める方式 2	22
3.11	累積値でスロットを配置した場合 1	22
3.12	累積値でスロットを配置した場合 2	23
3.13	元画像	24
3.14	リサイズ後画像	25
3.15	ガンマ補正の原理 <sup>4</sup>	26
3.16	LED マトリクスパネルでのガンマ特性	27
3.17	ガンマ 2.2 補正画像	28
3.18	ガンマ補正なしの画像（パネル表示）	28
3.19	ガンマ 2.2 補正画像（パネル表示）	29
3.20	Django から実行ファイルへの画像の流れと実行ファイル内の処理 (リサイズ・点灯制御)	30
3.21	Django Web アプリケーションの Web インタフェース	31
3.22	Cloudflare Tunnel の概要図 <sup>5</sup>	31
3.23	Web サイトの画面	32
3.24	システム全体の構成	33
3.25	アップロード前の Web サイトの画面	33
3.26	アップロード後の Web サイトの画面	34
3.27	LED マトリクスパネルに表示された画像	34
4.1	提案方式におけるスロットパターン	37
4.2	1001 というビット列を表現する場合のスロットパターン	38
4.3	縞模様が現れた画像	39

---

4.4	4分割LEDマトリクスの走査制御 . . . . .	40
4.5	LEDマトリクスパネルを4つの領域に分割した画像 . . . . .	40
4.6	LEDマトリクスパネルを8つの領域に分割した画像 . . . . .	41
4.7	送信データの全体設計 . . . . .	42

# 表 目 次

3.1 HUB75 64 × 32 LED マトリクスパネルの製品仕様 <sup>6</sup> . . . . .	10
3.2 Raspberry Pi 4 Model B の主要仕様 <sup>7</sup> . . . . .	14

# 第1章 序論

## 1.1 研究背景

近年, IoT (Internet of Things) 技術の普及やスマートシティの実現に向けた取り組みが活発になる中で, データ通信技術への需要はますます高まっている。特に, 位置情報に基づいた情報配信や, 公共空間における情報提供システムの重要性が増している。従来の無線通信技術 (Wi-Fi, Bluetooth, 携帯電話網など) は広く普及しているものの, 混雑した環境における通信品質の低下や, 電磁波干渉の問題, 更には通信インフラの設置コストといった課題が存在する。

可視光通信 (Visible Light Communication: VLC) は, 人の目に見える可視光線帯域の電磁波を用いて無線通信を行う技術である [1]。LED 等の光源が発する光が届く範囲で通信が可能となるため, 一般に使われている LED 照明を通信手段として流用できる点が特徴である。また, 可視光は生体への影響が小さく安全であり, 電磁波による他機器への悪影響も少ないとされる。

一方, 街中には大型の LED マトリクスパネルを用いた広告看板や情報表示装置が数多く設置されている。これらのパネルは, 高解像度で鮮明な画像や動画を表示できるだけでなく, 高速な点滅制御が可能であることから, 可視光通信の送信端末としての利用が期待されている。特に, スマートフォンのカメラ機能を利用して受信を行う方式は, ユーザーが専用の受信装置を用意する必要がなく, 既存のデバイスで情報を受信できる点で実用性が高い。例えば, 街中の大型広告パネルにスマートフォンのカメラを向けるだけで, その場所に関連する情報 (店舗情報, イベント情報, クーポンなど) を自動的に取得できる可能性がある。

しかしながら, 可視光通信を実用化する上では, いくつかの技術的課題がある。第一に, 人間の視覚に違和感を与えない範囲で情報を埋め込む必要がある。高速な点滅は通信速度の向上に寄与するが, ちらつきとして知覚されると, 表示装置としての機能を損なう可能性がある。第二に, スマートフォンカメラのフレームレートは通常 30~60 fps 程度であり, それよりも高い周波数で点滅させる場合, 受信側での工夫が必要となってくる。第三に, 通信速度を向上させるためには, 単一の領域で送信するよりも複数の領域を並列に利用する必要があるが, その際に人間の視覚に違和感を与えない範囲で情報を埋め込む必要がある。

これらのような技術的課題を解決するためには, LED マトリクスパネルを可視光通信の送信端末として活用する上で, 高品質な画像表示の実現, 遠隔からの表示制御, 視覚に違和感のない情報埋め込み方式の設計など, 段階的に整えるべき

技術要素がある。本研究では、これらの技術的要素を順に構築し、LED マトリクスパネルを用いた可視光通信の送信部システムの構築を目指す。

## 1.2 本研究の目的

本研究の目的は、LED マトリクスパネルを用いた可視光通信の送信部システムを構築することである。本研究は、段階的なアプローチにより、基礎的な表示技術の習得から始まり、最終的な可視光通信システムの送信部の実現に向けて実装を行った。具体的には、以下の 4 つの段階的な目標を設定した。

第一に、本研究で採用した点灯方式により 256 階調で表示するシステムを構築する。これにより、ユーザーが指定した任意の画像を LED マトリクスパネルに表示することができ、実用的な画像表示システムの基盤を確立する。

第二に、Cloudflare Tunnel と Django という Web アプリケーションフレームワークを使用して、遠隔地から画像を指定し、その指定された画像を表示可能なサイズや形式にリサイズして表示するシステムを確立する。これにより、実際の運用環境を想定したシステム構成を実現し、可視光通信システムの基盤となるインフラストラクチャを整備する。

第三に、人間の視覚に違和感を与えない明度に調整するためにガンマ補正の理論を理解し、独自の 256 階調表示システムを改良して補正を適用する。これにより、高品質な画像表示を可能にし、可視光通信システムにおいても視覚的に違和感のない表示を実現する基盤を構築する。

第四に、LED マトリクスパネルを 4 つの領域に分割し、それぞれの領域で異なる点滅周波数を用いて同一のデータを並列に送信する方式を採用する。これにより、単一領域で送信する方式と比較して 4 倍のデータ送信速度を実現することを目指す。

## 1.3 本論文の構成

本論文は以下のように構成されている。

第 2 章では、可視光通信および LED 表示に関連する他者の研究を調査し、関連研究をまとめる。

第 3 章では、LED マトリクスパネルを用いた表示システムの構築について述べる。まず、使用した HUB75 64 × 32 LED マトリクスのハードウェア仕様と、Raspberry Pi を中心とした開発環境について説明する。次に、本研究で採用した方式による 256 階調表示システムの実装方法を述べ、画像のリサイズとマッピング処理、Web サイトからの画像アップロードからパネル表示までの処理フローを説明する。最後に、構築した表示システムの画質改善手法（ガンマ補正など）について議論する。

第 4 章では、LED マトリクスパネルによる異なる点滅周波数を用いた可視光通

信方式について述べる。まず、点灯パターンと、4つの時間区間で輝度が等しくなる2種類のパターン（1010/1100）によるシンボル設計、4分割領域を用いた並列送信の構成を説明する。次に、同期語を用いない前提でのフレーム構成と FLAG 検出によるシンボル境界推定の考え方を整理する。さらに、スマートフォンカメラのローリングシャッターによって生じる縞パターンを利用した受信アルゴリズムの方針を述べる。

第5章では、提案方式の考察および今後の課題をまとめる。第3章・第4章で述べた構成と方式の特徴を踏まえ、実用化に向けた課題と今後の展望について議論する。



# 第2章 関連研究

本章では、LEDマトリクスパネルを用いた可視光通信と画像表示に関する他者の研究をまとめ、本研究の背景と位置づけを述べる。まず可視光通信の概要と応用、次にカメラで受信する方式、さらに表示と通信を両立させるための「ちらつきを抑える」設計についての研究を紹介する。最後に、これらの関連研究と本研究との関係を述べる。

## 2.1 可視光通信の概要と応用

可視光通信（VLC）は、可視光を使って近距離でデータを送る通信方式である。LEDの明るさを変化させることで情報を載せ、光が届く範囲で通信できる [2]。国際的な通信規格である IEEE 802.15.7 では、この方式の通信の仕様が定められており、照明として使うことを前提に、調光やちらつき抑制のための設計の考え方整理されている [3]。可視光通信は電波同士の干渉を受けにくいため、屋内での通信や位置の測定、情報の配信などへの応用が検討されている。商用化に向けた課題と展望も議論されている [4]。

従来の可視光通信では、光を電気信号に変える専用の受信装置が必要であった。一方、スマートフォンのカメラで受信する方式は、特別な装置がなくても既存の機器で情報を得られるため、実用化が期待されている。

## 2.2 カメラを用いた可視光通信

カメラで受信する方式では、送信側で LED の点灯・消灯のパターンを決め、受信側でカメラが撮った画像や動画からそのパターンを読み取り、データを取り出す。このとき、カメラがどのように光を写し取るかが、受信のしやすさに大きく影響する。

多くのスマートフォンのカメラは、画面を上から下へ順に一行ずつ光を読み取る方式である。このため、画面上の縦の位置によって、光を取り込む「時刻」がずれる。点滅している LED を撮影すると、ある行は点灯しているときに写り、別の行は消灯しているときに写るため、画像には明暗の縞模様が現れる。つまり、点滅の「時間の流れ」が、画像の「縦方向の位置」に広がって表れる。この縞模様を解析すれば、1枚の写真からでも、短い時間ごとの点灯・消灯の並び（0 と 1 の

並び) を推定できる可能性がある。

近年は、専用の受信装置が不要な「カメラで受信する方式」が注目され、光を使った近距離通信の国際規格（IEEE 802.15）のなかでも、カメラ受信を扱う検討が進められている [5] [6]. Nguyen らは、LED を  $8 \times 8$  個並べた送信機とカメラを使い、20–100 cm の距離で毎秒 120–960 ビット、最大 1.8 m で毎秒 13.44 kbps の通信を実現したと報告している [7]. また、カメラの撮影枚数（フレームレート）を超える速さでデータを送るため、縞模様からデータを取り出す方式も研究されている [8]. 本研究でも、この縞模様を利用して受信する方針とし、送信側では LED マトリクスパネルを複数の領域に分け、各領域で異なる点滅の速さを使って、同時にデータを送る構成を採用している。

## 2.3 LED 表示とちらつきの制御

可視光通信では、LED の点灯・消灯を速く切り替えて情報を載せるが、「表示」としても使う場合には、人が見たときに「ちらつき」として気にならない範囲に収める必要がある。表示と通信を両立させるには、照明として自然に見える範囲で、点滅の速さや「点灯している時間の割合」を決め、目にわかるちらつきを抑える設計が求められる。IEEE 802.15.7 では、照明として使うときの条件のもとで、データの送り方と明るさの調整・ちらつきの抑制についての考え方方が定められている [2] [3]. 規格で定められた通信方式の設計の意図として、ちらつき・明るさの調整とデータの送り方の関係が整理されている [9].

人の目には、「これより速く点滅すると、ずっと点灯しているように見える」という境界の周波数がある。この値は条件によって異なるが、およそ 60–100 Hz 程度とされる。LED の明るさの調整とデータ伝送を両立させる方法として、点灯時間の長さで表す方式（PWM）や、点滅の回数で表す方式（PFM）などが用いられる。本研究では、「ビット 0 とビット 1 で平均の明るさが違って見えてしまう」ことを避けるため、4つの時間区切りのあいだで点灯が 2 回（点灯している時間の割合が 50%）となるパターン（1010 と 1100）を採用している。このようにすることで、視覚的な違和感を抑えつつビットを送信している。

## 2.4 本研究の位置づけ

以上のように、可視光通信では、カメラで受信する方式、カメラの「上から下へ順に写し取る」性質（ローリングシャッター方式）によってできる縞模様からデータを取り出す方法、および表示と通信の両立のための「ちらつきを抑える」設計が、関連研究として進められている。

本研究では、LED マトリクスパネルを送信側として使い、(1) 256 階調表示と gamma 補正による見やすい画像表示、(2) 2 種類の点滅の速さと、点灯割合が 50% に

## 第 2 章

---

なるパターン（1010/1100）による、見た目に配慮した0と1の送信、(3) パネルを4つの領域に分けて同時に送る構成、を組み合わせた方式を提案している。関連研究で用いられている「縞模様の利用」や「点灯の割合・ちらつきの考慮」と同様の考え方を、LEDマトリクスパネルとその駆動方式（HUB75）を用いた具体的な実装としてまとめ、表示の質と通信の両立を目指している点に、本研究の特徴がある。



# 第3章 LEDマトリクスパネルを用いた画像表示システムの構築

本章では、LEDマトリクスパネルの基本的な仕様を理解した上で、独自の画像表示システムを構築する。256階調での画像表示システムの構築を始めとし、遠隔地から画像をアップロードして表示を行えるシステムや、ガンマ補正を適用した画像表示の品質向上を目指す。

## 3.1 使用デバイスと開発環境

本節では、使用したHUB75 64 × 32 LEDマトリクスパネルのハードウェア仕様と、Raspberry Piを中心とした開発環境についてまとめる。

### 3.1.1 HUB75 64 × 32 LEDマトリクスの仕様

本研究では、HUB75インターフェースを採用した64 × 32ピクセルのLEDマトリクスパネル[10]を使用した。HUB75は、市販のRGB LEDマトリクスが画像データを受け取るために用いる信号インターフェースである[11]。本システムでは、64列×32行の合計2,048個のRGB LEDを制御し、各LEDは赤(R)、緑(G)、青(B)の3色を独立に制御できる。

HUB75インターフェースは、データ信号、クロック信号、ラッチ信号、出力イネーブル(OE)信号、行アドレス信号などから構成され、高速な走査制御を実現する。本システムでは、このHUB75インターフェースを用いて120 Hz相当のフレーム更新を実行し、可視光通信の送信端末として機能させる。使用したLEDマトリクスパネルの製品仕様を表3.1に示す。

### 第 3 章

表 3.1: HUB75 64 × 32 LED マトリクスパネルの製品仕様<sup>1</sup>

製品仕様	
懸念される化学物質 (High-concerned chemical)	なし (None)
カスタマイズされます (is_customized)	はい (Yes)
ピクセル / ピクセルピッチ (Pixels / Pixel pitch)	4mm
表示機能 (Display Function)	ビデオ (VIDEO)
チューブチップの色 (Tube Chip Color)	フルカラー (Full Color)
モデル番号 (Model Number)	P4-64X32-16S or 32S
銘柄 (Brand Name)	OIMG / Starlight
起源 (Origin)	Cn (中国本土, Mainland China)
用途 (Use)	屋内 (Indoor)

本実験で使用した HUB75 64 × 32 LED マトリクスパネルの外観と点灯例をそれぞれ図 3.1 と図 3.2 に示す。

<sup>1</sup><https://ja.aliexpress.com/item/1005010223788711.html> より引用。



図3.1: HUB75 64 × 32 LED マトリクスパネルの外観<sup>2</sup>

<sup>2</sup><https://ja.aliexpress.com/item/1005010223788711.html> より引用。

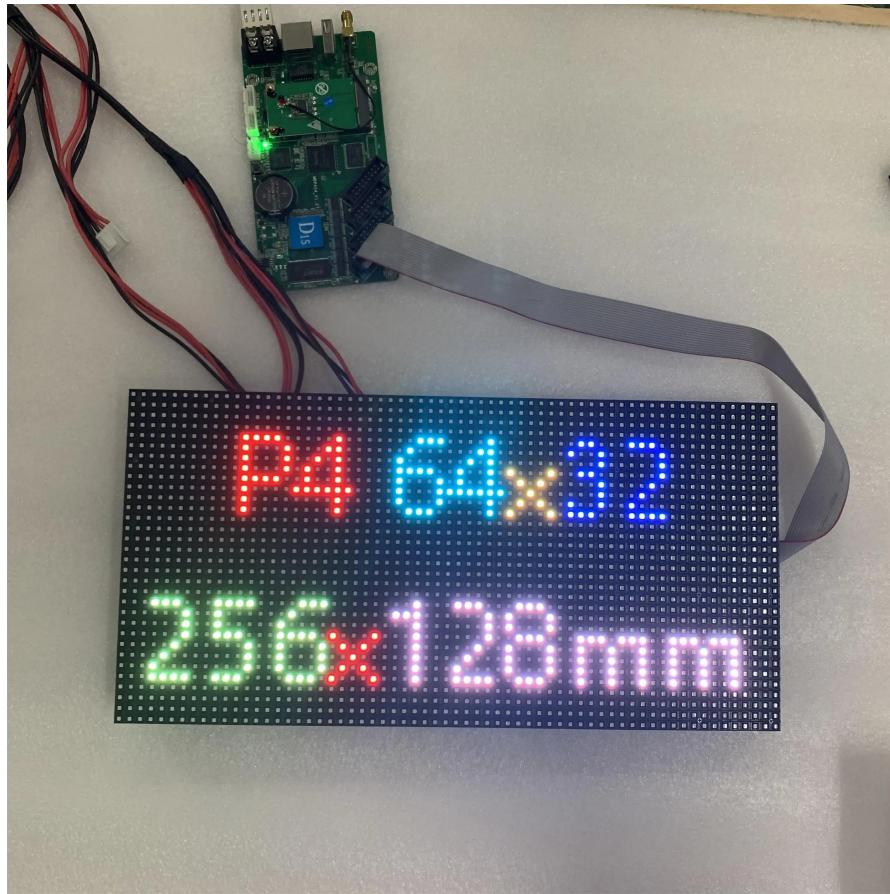


図 3.2: HUB75 64 × 32 LED マトリクスパネルの点灯例<sup>3</sup>

### 3.1.2 Raspberry Pi 4 と開発環境

本研究では、Raspberry Pi 4 Model B を制御用コンピュータとして使用した。Raspberry Pi 4 は、ARM アーキテクチャを採用したシングルボードコンピュータであり、GPIO (General Purpose Input/Output) ピンを用いて HUB75 インタフェースの LED マトリクスパネルを直接制御することが可能である。本システムでは、Raspberry Pi 4 上で C 言語による LED パネル制御プログラムを実行し、高速な点滅制御を実現している。また、Python 環境を併用して、Web サーバの動作も Raspberry Pi 4 上で実現している。

Raspberry Pi 4 Model B の外観と主要仕様をそれぞれ図 3.3 と表 3.2 に示す。

---

<sup>3</sup><https://raspberry-pi.ksyic.com/?pdp.id=552> より引用。



図 3.3: Raspberry Pi 4 Model B の外観<sup>4</sup>

---

<sup>4</sup><https://www.element14.com/raspberry-pi/raspberry-pi-4-model-b/dp/SC0195/item> より引用.

表 3.2: Raspberry Pi 4 Model B の主要仕様<sup>5</sup>

基本仕様	
販売元	element14
製品型番	SC0195/0765756931199
リビジョン	1
SoC	Broadcom BCM2711
CPU	1.5GHz クアッドコア Cortex-A72 (ARMv8, 64bit, L1=データ用 32KB 命令用 48KB/Core, L2=1MB)
GPU	デュアルコア VideoCore VI® 500MHz, OpenGL ES 3.0 対応, ハードウェア OpenVG 対応, H.265 (HEVC) 4Kp60 デコード, H.264 1080p60 デコード / 1080p30 エンコード
メモリー	8GB LPDDR4-3200 SDRAM
電源	USB type C ソケット 5V 3.0A / 2.54mm ピンヘッダー / PoE (要オプション PoE HAT)
消費電力 (本製品単体)	アイドル: 約 3W, ストレス: 約 6.25W
サイズ	85 × 56 × 18mm
生産国	英国
インターフェース	
イーサネット	10/100/1000 Base-T RJ45 ソケット (BCM54213PE)
無線 LAN (WiFi)	IEEE 802.11 b/g/n/ac 2.4/5GHz デュアルバンド (Cypress CYW43455)
Bluetooth	Bluetooth 5.0, Bluetooth Low Energy (Cypress CYW43455)
ビデオ出力	micro HDMI × 2, コンポジット 3.5mm 4 極ジャック (PAL, NTSC), DSI 2-lane (15pin 1mm ピッチ)
オーディオ出力	3.5mm 4 極ジャック, micro HDMI (ビデオ出力と共有) × 2, I2S ピンヘッダー
カメラ入力	2-lane MIPI CSI (15pin 1mm ピッチ)
USB	USB 2.0 × 2, USB 3.0 × 2 (VIA VL805 PCIe)
GPIO コネクター	40 ピン 2.54mm ピンヘッダー (GPIO × 26 3.3V 16mA, UART, I2C, SPI, I2S, PWM, 5V 出力 (使用電源に依存), 3.3V 出力 50mA (GPIO 信号との総和))
メモリー カード スロット	micro SD メモリーカード (SDIO)

<sup>5</sup><https://raspberry-pi.ksyic.com/?pdp.id=552> より引用

## 3.2 256階調表示システム

本節では、256階調表示システムの構築について説明する。256階調表示システムとは、LEDマトリクスパネル上でRGB各色をそれぞれ256段階の明るさで表現し、フルカラー画像を表示するための仕組みである。本研究では可視光通信の送信端末としてLEDマトリクスパネルを用いるため、送信する画像を視覚的に識別しやすく表示する必要があり、そのために256階調での表示を実現した。本節の構成は次のとおりである。まず、自分が今回採用した点灯方式による256階調表現と点灯タイミングの計算方法を述べ、続いて本研究の点灯方式と一般的なPWM制御との比較を行う。その後、画像のリサイズおよびマッピング処理から点灯制御までの処理手順を説明する。

### 3.2.1 本研究で採用した方式による256階調表現

RGB各色を、今回採用した方式（各色を点灯する回数で輝度を表す）により256階調で表示するシステムを構築する。HUB75では、一つのLEDでR,G,Bの3色をそれぞれ点灯させるかしないかを独立に制御が可能である。その仕様を利用して、R,G,Bそれぞれの点灯回数で256階調を表現する。ここで用いるスロットとは、時間軸を等間隔に区切った一区間であり、その区間では「点灯する」か「点灯しない」かのどちらかしか選べない、一回の点灯判定の最小単位である。本研究の方式では、このスロットを256個用意し、そのうち何回のスロットで点灯するか（点灯回数）で輝度を表す。

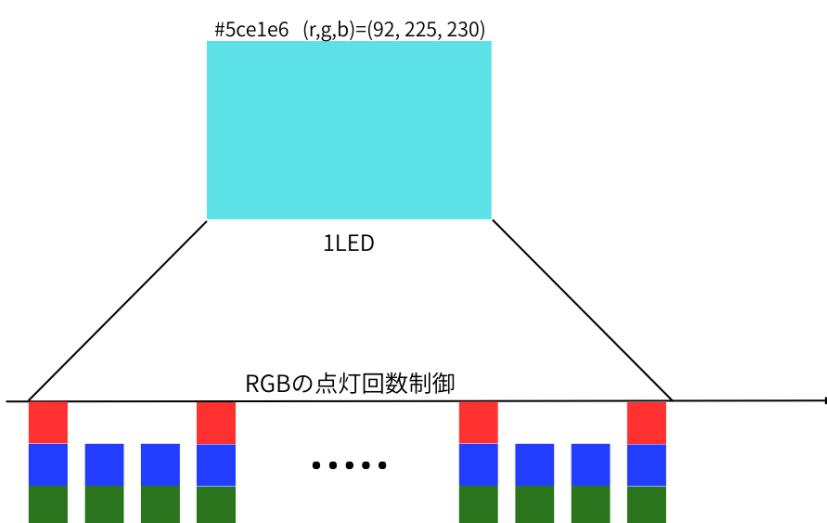


図 3.4: 本研究で採用した方式による256階調表現

例えば、図3.4の例では、シアンを表現しているが、シアンの場合は $(R,G,B)=(92,225,230)$ であり、この場合、256階調のうち92回Rを点灯させ、225回Gを点灯させ、230回Bを点灯させることで表現が可能である。

### 3.2.2 一般的な点灯制御方式との比較

一般的な点灯制御方式としては、PWM (Pulse Width Modulation) 制御がある。PWM制御は、一定周期のパルスのうちONになっている時間の割合（デューティ比）を変えることで見かけの明るさを変える方式である。例えばデューティ比50%であれば1周期の半分の時間だけ点灯し、平均として中間の輝度になる。多くのマイコンやLEDドライバでは、ハードウェアまたは専用回路でPWM波形を生成し、デューティ比を輝度値に応じて設定して多階調表示を実現している。例として以下の図3.5のような画像を表示したい場合を考える。

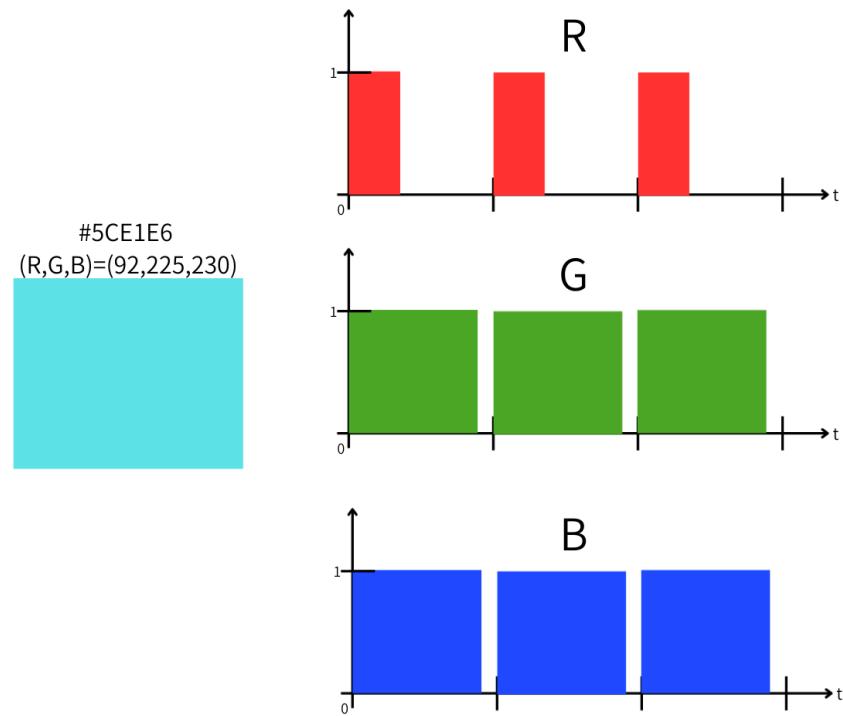


図3.5: PWM制御による点灯制御

図3.5は図3.4と同様に $(R,G,B)=(92,225,230)$ の場合を示している。この場合、 $R=92$ であるためデューティ比は $92/256 \approx 36.3\%$ で点灯させ、 $G=225$ は $225/256 \approx 87.8\%$ 、 $B=230$ は $230/256 \approx 90.2\%$ のデューティ比でそれぞれ点灯させることで表現が可能である。

このように、PWM制御では輝度値をデューティ比に変換して点灯させることで、256階調を表現することができる。

しかし、本研究で採用した方式では、デューティ比ではなく、1フレームを256スロットに分割し、各スロットで点灯するかしないかを整数演算で決めることで、輝度値  $R$  を256スロット中  $R$ 回の点灯として表現する。点灯させるタイミングの決め方については次節で述べる。

本研究の方式はPWMと本質的には変わらず、いずれも時間あたりの点灯時間の割合で輝度を表す。相違点は、PWMがデューティ比（点灯している時間の長さ）で制御するのに対し、本研究の方式はスロットを256個用意し、そのうち何回点灯させるかという回数で制御する点である。この違いにより、次のようなメリットがある。

第一に、輝度値が0~255の整数でそのまま点灯回数に対応するため、後述するガンマ補正後の値をそのまま使え、画像処理との接続が容易である。

第二に、後の章で述べる可視光通信ではスロット単位の点灯パターンでデータを送るため、同じスロット構造を共有でき、表示と通信の両立がしやすい。

また、実装容易性の面では、PWMではハードウェアまたは専用回路でデューティ比に応じた波形を生成する必要があるのに対し、本研究の方式ではHUB75のON/OFF制御のみで、ソフトウェアで各スロットの点灯有無を整数演算で決めればよい。

以上より、本研究ではこの方式を採用した。

### 3.2.3 点灯させるタイミングの計算方法

本研究では1フレームを1スロットとして、256スロット（=256フレーム分）のあいだで点灯回数を制御する。したがって、1枚の画像は256フレームをかけて表現される。しかし、単純に1フレーム内でスロットを配置するだけでなく、スロットをフレーム内で均等に配置する必要がある。その理由を説明するにあたって、均等に配置しないパターンを以下の図3.6に示す。

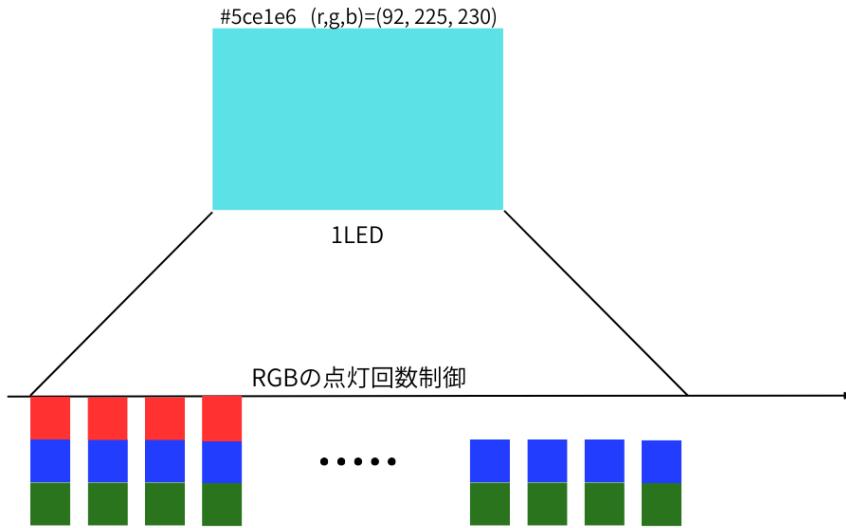


図 3.6: スロットの均等に配置しないパターン

図 3.6 は、シアンを表現するにあたって、スロットの均等に配置しないパターンを示している。図のように均等に配置せずに表示をすると、前半は RGB 全てが点灯されるが、後半赤色が配置されなくなるため、フレームの前半と後半で見える色が違って見える。このように、フレーム内でスロットを均等に配置しないと、フレームの前半と後半で見える色が違って見えるため、画質が損なわれる可能性がある。

そこで、本研究ではスロットを均等に配置する方法を採用した。当初は、スロットを均等に配置する方法として、整数除算でスロットの配置パターンを決める方法を検討した。

例えば、 $R = 128$  の場合、256 スロットを 128 で割った商は 2 であるため、256 スロット中で 2 回に 1 回点灯することになる。しかし、この方法では、 $R$  が 256 の約数でない場合に、スロットを均等に配置することができない。例えば、 $R = 200$  の場合、256 スロットを 200 で割った商は 1 であるため、256 スロット中で 1 回に 1 回、つまり毎回点灯することになってしまう。

そのため、本研究で採用した方法では、各スロットごとに累積値 (accumulator) に目標輝度値  $R$  を加算し、加算結果が 256 を超えたとき（オーバーフローしたとき）に点灯し、累積値を 256 で割った余りに更新する。これにより、 $R$  が 256 の約数でない場合でも、256 スロット中  $R$  回の点灯を時間的に均等に分散でき、階調の線形性と画質の向上が期待できる。本研究では、スロットを均等に配置する方法として、この方法を採用した。以下、まず  $R = 200$  の具体例を示し、続いてこの方法が成り立つ理由を述べる。

具体例として  $R = 200$  の場合を、図 3.7 を使って説明する。

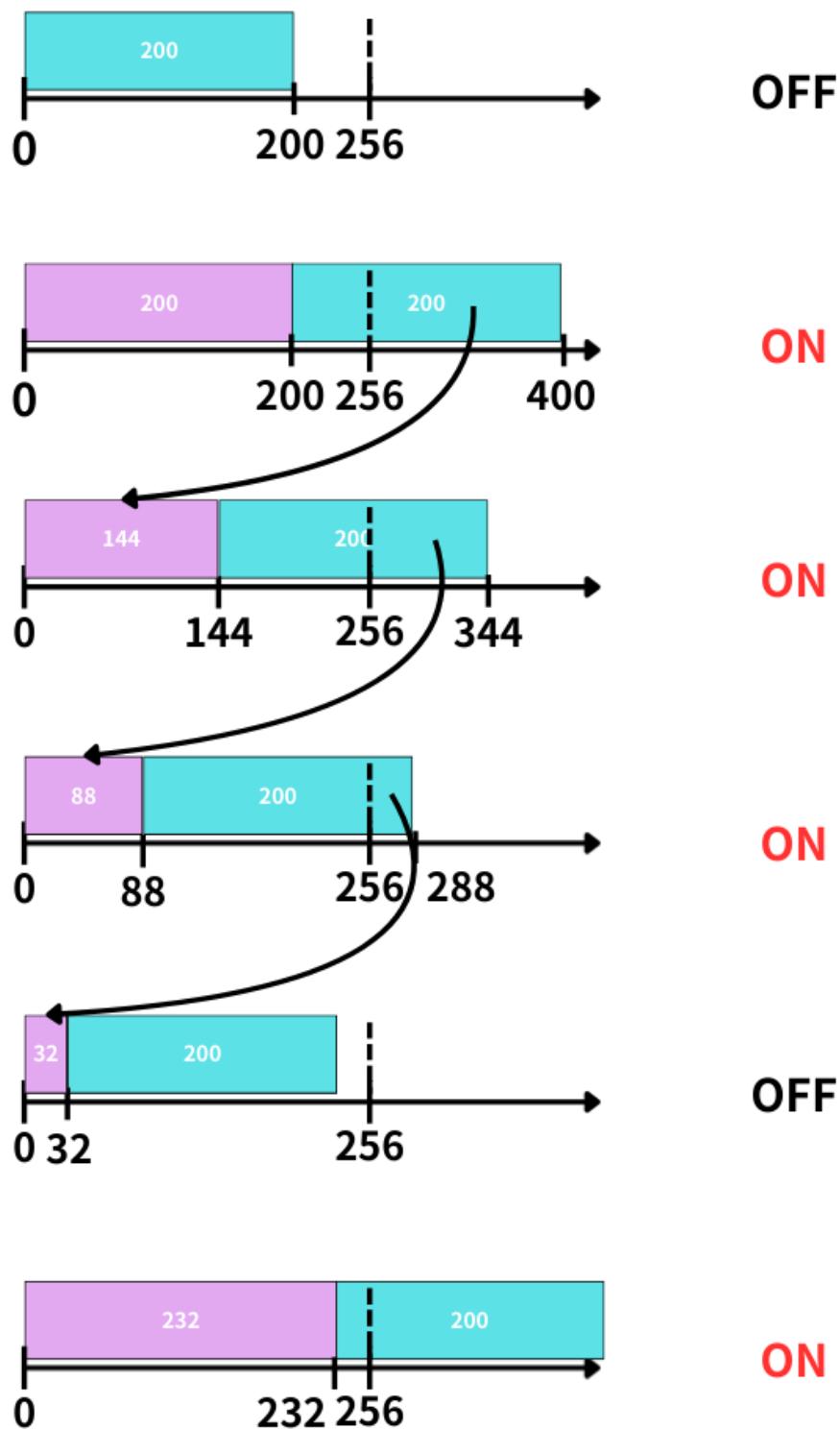
図 3.7:  $R = 200$  のときの累積値の遷移

図 3.7 は、 $R = 200$  のときの累積値の変化を表したものである。図中の紫色の

領域が累積値、水色の領域が目標輝度値  $R$ （この例では 200）を表す。初期値は  $\text{acc}_0 = 0$  である。

各スロットでは「累積値に 200 を足す」を行う。足した結果が 256 未満ならそのまま次の累積値とし、256 以上なら点灯して 256 を引いた余りを次の累積値とする。図 3.7 を見ると、スロット 1 では  $0 + 200 = 200$  で 256 未満のため点灯せず、累積値は 200 になる。スロット 2 では  $200 + 200 = 400$  で 256 以上となるため点灯し、余りは  $400 - 256 = 144$  なので累積値は 144 になる。同様に、スロット 3 では  $144 + 200 = 344$  で点灯し累積値 88、スロット 4 では  $88 + 200 = 288$  で点灯し累積値 32、スロット 5 では  $32 + 200 = 232$  で 256 未満のため点灯せず累積値 232、となる。図中で累積値が「下がる」瞬間が、そのスロットで点灯したことを意味し（256 を引いたため値が小さくなる）。累積値が上がる瞬間が、そのスロットで点灯しなかったことを意味する。

このように、累積値が 256 を超えたスロットで点灯するので、点灯列は「OFF, ON, ON, ON, OFF, ON, ON, ON, ON, OFF, ON, ON, ON, OFF, ON, ...」のようになる。256 スロットのあいだにちょうど 200 回点灯し、かつ点灯が均等に分散するため、輝度 200 の階調を正しく表現できる。

次に、この方法が成り立つ理由を理論的に述べる。

$n$  番目のスロット開始時点の累積値を  $\text{acc}_n$  とし、初期値を  $\text{acc}_0 = 0$  とする。各スロットでは、現在の累積値に  $R$  を加算し、加算結果が 256 以上であれば点灯して 256 を引いた余りを次の累積値とし、256 未満であれば点灯せず加算結果をそのまま次の累積値とする。これを式で表すと

$$\text{acc}_{n+1} = (\text{acc}_n + R) \bmod 256 \quad (3.1)$$

となる（ $R$  は 0 以上 255 以下の整数）。ここで  $\bmod 256$  は 256 で割った余りを表す。加算結果が 256 以上であるとき、余りは 256 未満となり、そのスロットで点灯が発生したことを意味する。

256 スロット中に点灯がちょうど  $R$  回になる理由は次のとおりである。

まず、256 スロットのあいだに累積値へ加算される量の合計は  $256 \times R$  である。その理由は、各スロットで  $R$  を 1 回ずつ加算する操作を 256 回行うためである。点灯は、累積値に  $R$  を加算した結果が 256 以上になるタイミングで発生し、そのたびに 256 を引いた余りを次の累積値とする。したがって、256 スロット中で点灯された回数を  $k$  回とすると、256 を引く操作がちょうど  $k$  回行われ、差し引かれた量の合計は  $256 \times k = 256k$  となる。よって 256 スロット終了時点の累積値は  $256R - 256k$  と表せる。累積値は常に 0 以上 255 以下の範囲に保たれるため、 $0 \leq 256R - 256k < 256$  が成り立つ。両辺を 256 で割ると  $0 \leq R - k < 1$  となり、 $k$  は点灯回数なので整数である。この不等式を満たす整数  $k$  は  $k = R$  のみである。したがって、256 スロット中の点灯回数はちょうど  $R$  回である。

点灯が時間的に均等に配置される理由は次のとおりである。

累積値を  $\text{acc}$  と書く。各スロットでは  $\text{acc} \leftarrow \text{acc} + R$  とし、 $\text{acc} \geq 256$  のとき

点灯して  $acc \leftarrow acc - 256$  とする。したがって  $acc$  は、スロットごとに  $+R$  され、 $acc \geq 256$  になるたびに点灯して、 $acc$  が 0 以上 255 以下の値に更新される、という動きを繰り返す。つまり「次の点灯」が起こるのは、 $acc$  が再び 256 に達するときである。

点灯直後の余りを  $r$  ( $0 \leq r \leq 255$ ) とする。次に 256 に達するまでに必要な加算量は  $256 - r$  である。1 スロットあたりの加算量は  $R$  なので、点灯と点灯のあいだのスロット数  $n$  は

$$n = \left\lceil \frac{256 - r}{R} \right\rceil \quad (3.2)$$

となる。 $0 \leq r \leq 255$  より

$$1 \leq n \leq \left\lceil \frac{256}{R} \right\rceil \quad (3.3)$$

が成り立つ。 $R$  が 256 の約数ならば  $256/R$  は整数であり、 $n = 256/R$  が常に成り立つので、点灯はちょうど  $256/R$  スロットごとに起こる。 $R$  が 256 の約数でない場合でも、 $n$  は式 (3.3) の範囲に収まり、いずれも  $256/R$  の前後である。このように、毎スロット同じ  $R$  を加算し「256 に達したら点灯して余りに更新する」という規則そのものが、点灯を時間軸上に可能な限り均等に並べる。以上より、本方法により 256 スロット中  $R$  回の点灯が保証され、かつ均等に配置されることが示される。実際に点灯する画像を図 3.8、整数除算でスロット配置を決める方式で点灯させ、ランダムなタイミングで撮影を行った画像を図 3.9、3.10、本実験で採用した累積値でスロットを配置させる方式で点灯させ、ランダムなタイミングで撮影を行った画像を図 3.11、3.12 に示す。



図 3.8: LED マトリクスパネルに表示した元画像



図 3.9: 整数除算でスロット配置を決める方式の場合 1



図 3.10: 整数除算でスロット配置を決める方式 2



図 3.11: 累積値でスロットを配置した場合 1



図 3.12: 累積値でスロットを配置した場合 2

前提として、図 3.9, 3.10, 3.11, 3.12 は、撮影タイミングによる見え方の差が生じやすいよう、パネルを 1000 Hz という比較的低い駆動周波数で駆動した条件下で撮影したものである。

図 3.9, 3.10 では、整数除算でスロットの配置を決める方式で点灯させ、ランダムなタイミングで撮影を行った画像であるが、タイミングによって映るもののが全く違うことが確認できる。一方で、図 3.11, 3.12 は、累積値でスロットを配置させる方式で点灯させ、ランダムなタイミングで撮影を行った画像であるが、この方式では、タイミングによって映るものがあまり変わらないことが確認できる。その理由は、累積値でスロットを配置させる方式では、スロットの配置が均一になるため、どのタイミングで撮影しても映るものに差がないためである。

### 3.2.4 $32 \times 64$ 画像へのリサイズとマッピング処理

普段我々が SNS 等で扱う画像は、一般的に  $1920 \times 1080$  や  $1280 \times 720$  などの大きさであるが、LED マトリクスパネルのサイズは  $32 \times 64$  であるため、一般的な画像を LED マトリクスパネルに画像を表示するにあたってリサイズが必要である。今回は  $32 \times 64$  の画像へリサイズするにあたり、各出力ピクセルが元画像上で占める領域を求め、その領域に含まれる画素の RGB 値を単純平均する方法（ボックス平均）を採用した。具体的には、出力画像の座標  $(x, y)$  ( $0 \leq x < 64, 0 \leq y < 32$ ) に対して、元画像の幅を  $W$ 、高さを  $H$  とすると、対応する入力領域を

$$x_0 = \left\lfloor \frac{xW}{64} \right\rfloor, \quad x_1 = \left\lfloor \frac{(x+1)W}{64} \right\rfloor, \quad y_0 = \left\lfloor \frac{yH}{32} \right\rfloor, \quad y_1 = \left\lfloor \frac{(y+1)H}{32} \right\rfloor \quad (3.4)$$

で定義し、この矩形領域  $[x_0, x_1) \times [y_0, y_1)$  に含まれる全画素の RGB 値を加算して画素数で除算することで、出力画素の値を得る。RGB の各成分は独立に平均し、

$$R'(x, y) = \frac{1}{N} \sum_{(i,j) \in \Omega(x,y)} R(i, j) \quad (3.5)$$

$$G'(x, y) = \frac{1}{N} \sum_{(i,j) \in \Omega(x,y)} G(i, j) \quad (3.6)$$

$$B'(x, y) = \frac{1}{N} \sum_{(i,j) \in \Omega(x,y)} B(i, j) \quad (3.7)$$

とした。ここで  $\Omega(x, y)$  は上記入力領域に含まれる画素集合、 $N$  はその画素数である。リサイズ後の RGB 値は、C 言語上で定義した RGB 構造体配列に格納し、LED マトリクスパネル表示処理に入力する。

実際に以下の図 3.13 のような画像をリサイズした結果を図 3.14 に示す。



図 3.13: 元画像



図 3.14: リサイズ後画像

### 3.3 ガンマ補正による画質改善

本節では、3.2節までの処理で、実際に画像を表示した際に色が淡くなり形状が明確でない問題について原因を調査し、ガンマ補正を行って画質を改善する方法について説明する。

#### 3.3.1 ガンマ補正

ガンマとは、コンピュータの画像処理においては中間調（グレー）の明るさを示す用語であり、各機器には固有の発色特性（ガンマ特性）があり、入力された色情報をそのまま素直に出力できない [12]。色情報の入力を  $x$ 、出力を  $y$ 、ガンマ値を  $\gamma$  とすると、

$$y = x^\gamma \quad (3.8)$$

で表される。機器側のガンマ特性に合わせて色情報を調整して帳尻を合わせる仕組みをガンマ補正と呼ぶ [12]。

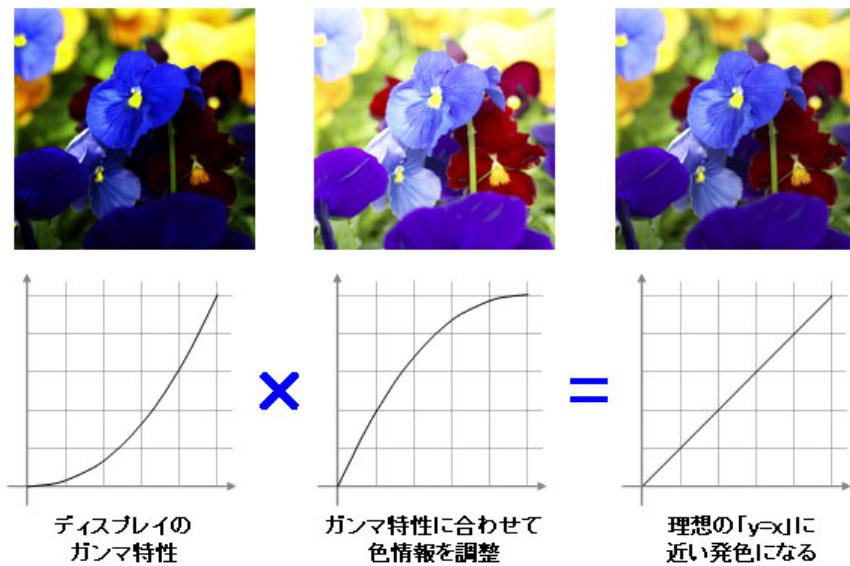
図 3.15: ガンマ補正の原理<sup>6</sup>

図 3.15 では、ガンマ補正の原理を示している。通常の画像ファイルにはガンマ 0.45 の補正がかかっており、この補正によって見た目よりも淡いデータとして保存されている。これを式で説明する。意図する明度（線形な光の強さ）を  $L$  ( $0 \leq L \leq 1$ )、画像ファイルに保存された値を  $s$  とすると、ガンマ 0.45 の符号化は  $s = L^{1/2.2} = L^{0.45}$  である。この  $s$  をディスプレイに入力すると、ディスプレイは式 (3.8) の関係で  $\gamma = 2.2$  の補正をかけて出力する。つまり出力  $y$  は

$$y = s^{2.2} = (L^{0.45})^{2.2} = L^{0.45 \times 2.2} = L^{0.99} \approx L \quad (3.9)$$

となる。したがって、画像ファイルでのガンマ 0.45 とディスプレイでのガンマ 2.2 が打ち消し合い ( $0.45 \times 2.2 \approx 1$ )、最終的に発光強度  $y$  が元の明度  $L$  にほぼ比例する。このため見た目の明度が線形に再現される。

このように、通常のディスプレイではガンマ 2.2 の補正が行われているため、画像ファイルは内部的にガンマ 0.45 の補正が行われている。しかし、LED マトリクスパネルではこの補正が行われないため、ガンマ 0.45 のまま表示することになり、元の画像よりも淡く表示される問題が発生していた。この問題を可視化したもののが図 3.16 である。

<sup>6</sup>[https://www.eizo.co.jp/eizolibrary/other/itmedia02\\_07/](https://www.eizo.co.jp/eizolibrary/other/itmedia02_07/) より引用

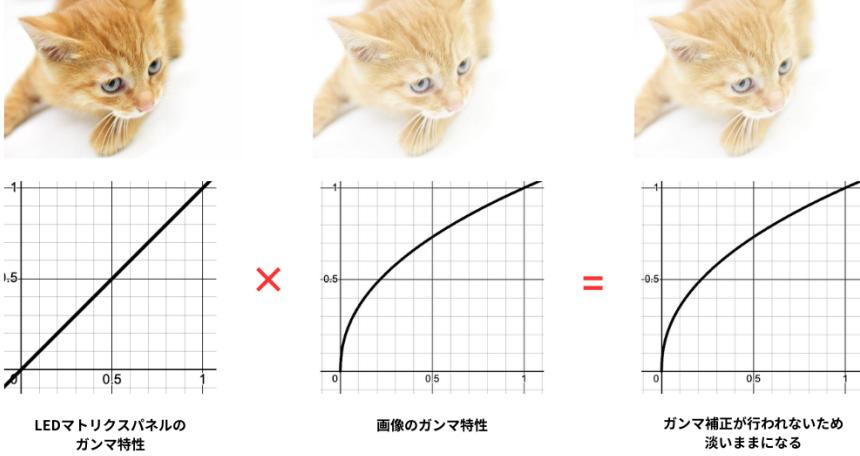


図 3.16: LED マトリクスパネルでのガンマ特性

図 3.16 を見てわかる通り、LED マトリクスパネルにはガンマ特性がないため線形となり、入力された値をそのまま出力する。そのため、ガンマ 0.45 に変換されている画像をそのまま出力することで淡いまま出力してしまう。

そこで本研究では、LED マトリクスパネル上で元画像と近い見た目を得るために、画像データをパネルへ出力する前段で逆ガンマ補正（ガンマデコード）を行った。具体的には、リサイズ後の各画素の RGB 値 (0–255) を 0–1 の範囲に正規化し、各成分  $s$  に対して

$$L = s^{2.2} \quad (3.10)$$

を適用したうえで、再び 0–255 にスケーリングして整数化し、LED 表示用の RGB 構造体配列に格納した。実装上は、画素値  $R$ ,  $G$ ,  $B$  をそれぞれ

$$s_R = \frac{R}{255}, \quad s_G = \frac{G}{255}, \quad s_B = \frac{B}{255} \quad (3.11)$$

とし、

$$R' = \text{round}(255 \cdot s_R^{2.2}) \quad (3.12)$$

$$G' = \text{round}(255 \cdot s_G^{2.2}) \quad (3.13)$$

$$B' = \text{round}(255 \cdot s_B^{2.2}) \quad (3.14)$$

として求めている。これにより、通常はディスプレイ側で行われているガンマ 2.2 の補正をソフトウェア側で補い、LED マトリクスパネル上でも元画像に近い明度分布を再現できるようにした。実際にこの方法でガンマ補正を行った結果は以下の図 3.17 の通りである。

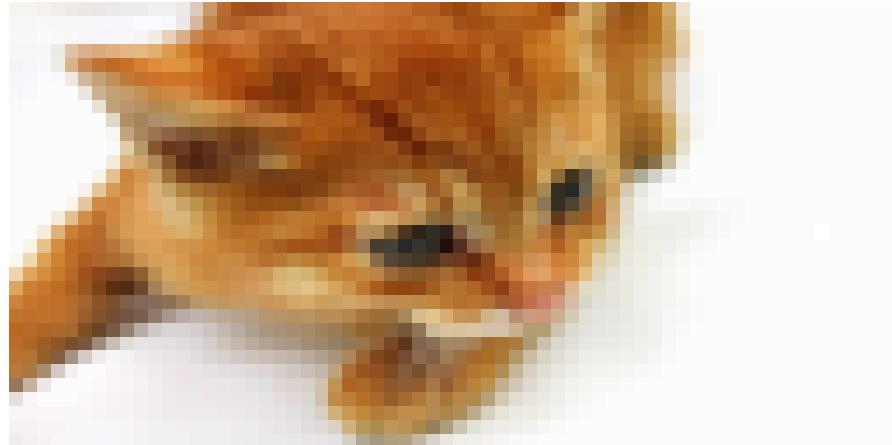


図 3.17: ガンマ 2.2 補正画像

また、ガンマ補正をしてない状態でパネルに表示した結果を図 3.18、ガンマ補正したものをパネルに表示した結果は以下の図 3.19 の通りである。



図 3.18: ガンマ補正なしの画像（パネル表示）

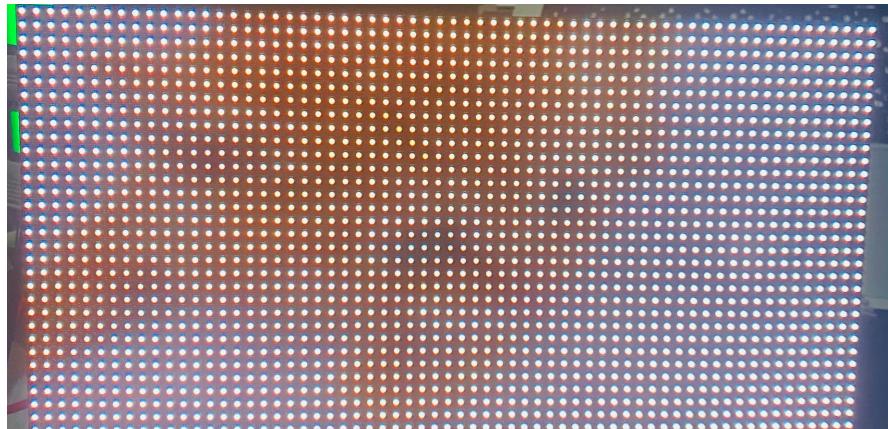


図 3.19: ガンマ 2.2 補正画像（パネル表示）

実際にパネルに表示した結果を見ると、ガンマ 2.2 補正を行ったことで、補正を行っていない画像よりもより鮮明に綺麗に表示されていることが確認できた。

### 3.4 画像アップロードシステムとネットワーク構成

本研究では、LED マトリクスパネルを可視光通信の送信端末として用いることを目指している。送信する画像や表示内容を変更するたびにパネル設置場所へ赴くのは非効率であり、様々な画像で表示を切り替えながら視覚的に確認する実験には不向きである上、実用化後の利便性も損なわれる。そこで、遠隔地から画像をアップロードしてパネルに表示できるようにすることで、研究者や利用者がパネルに物理的に近づかずに表示内容を更新でき、可視光通信の実験や応用を柔軟に進められるようにする。

本節では、遠隔地から画像をアップロードして LED マトリクスパネルに表示するための Web アプリケーションとネットワーク構成について説明する。あわせて、画像処理と LED マトリクスパネル制御を担う実行ファイルについても説明する。

#### 3.4.1 画像処理や LED マトリクスパネル制御用の実行ファイル

本システムでは、C 言語で作成した実行ファイルが画像処理と LED マトリクスパネルの点灯制御を担っている。Django から実行ファイルに画像が渡され、実行ファイル内でリサイズしたのち、LED マトリクスパネルへの点灯制御を行うまでの流れを図 3.20 に示す。

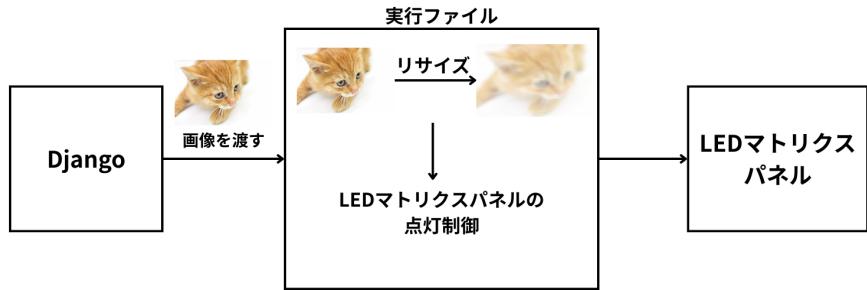


図 3.20: Django から実行ファイルへの画像の流れと実行ファイル内での処理（リサイズ・点灯制御）

実行ファイルが行う処理のうち、画像処理としては 3.2.4 節で述べるリサイズがある。PNG や JPG などでアップロードされた画像はもともと任意の縦横ピクセル数を持つが、今回使用した LED マトリクスピネルは 64 列 × 32 行であるため、実行ファイル内で画像を  $64 \times 32$  ピクセルにリサイズする処理を行っている。また、点灯制御としては 3.2.1 節で述べる 256 階調表現がある。実行ファイルは Raspberry Pi の GPIO (General Purpose Input/Output) を制御し、HUB75 インタフェースを通じて LED マトリクスピネルに点灯パターンを送ることで、RGB 各色の点灯回数により 256 階調で表示する。

### 3.4.2 Web アプリケーションの構成

画像のアップロードと表示制御のための Web アプリケーションとして、Django<sup>7</sup> フレームワークを用いた Web サーバを構築した。Django は Python で記述された Web フレームワークであり、画像アップロード、ファイル管理、API エンドポイントの実装が容易である。本システムでは Django を用いて画像アップロード用の Web インタフェースを提供し、アップロードされた画像を LED マトリクスピネルに表示する。

本システムが提供する Web インタフェースとその処理の流れを図 3.21 に示す。

<sup>7</sup><https://www.djangoproject.com> より引用

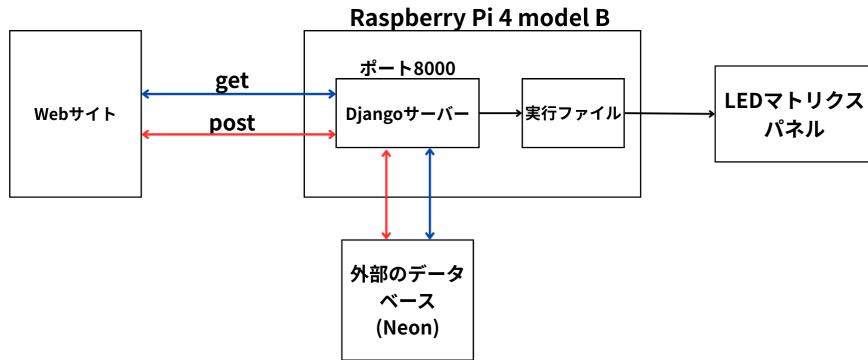
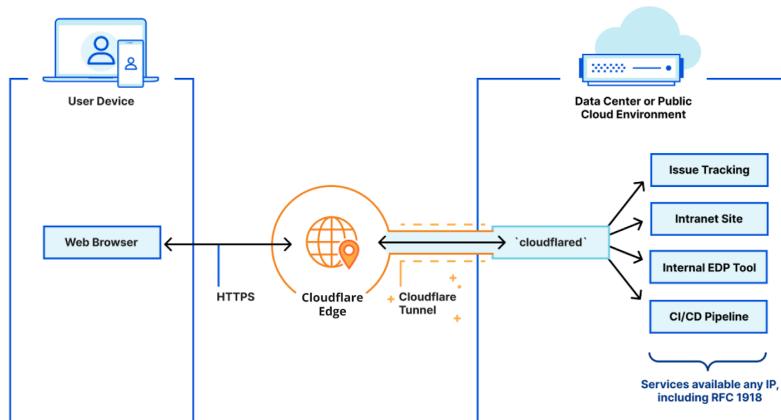


図 3.21: Django Web アプリケーションの Web インタフェース

Django は Raspberry Pi 上でポート 8000 番のサーバとして動作し、常に外部からのリクエストを受け付けている。図 3.21 中の get は GET リクエストを指し、Django はこれを受信すると外部データベースに問い合わせ、現在パネルに表示中の画像を取得して返却する。これは Web サイト上での表示に必要な処理である。post は、新たな画像をアップロードする際に用いる POST リクエストを指す。POST リクエストでアップロードされた画像は Neon [13] のデータベースに保存したのち、Raspberry Pi 4 上で実行ファイルにより処理される。実行ファイルによる処理内容は 3.4.1 節の通りである。

### 3.4.3 Cloudflare Tunnel による外部アクセス

本システムでは、Cloudflare Tunnel [14] を用いて Raspberry Pi 4 上で動作する Django Web アプリケーションに外部からアクセスできるようにした。

図 3.22: Cloudflare Tunnel の概要図<sup>8</sup>

Cloudflare Tunnel は、Cloudflare が提供するトンネリングサービスであり、パブリック IP アドレスやポート開放なしに、ローカルネットワーク内のサービスをインターネット経由でアクセス可能にする。これにより、Raspberry Pi 4 が設置されているローカルネットワークの設定を変更することなく、外部の PC やスマートフォンからブラウザ経由で画像をアップロードし、LED マトリクスパネルに表示することが可能となった。

### 3.4.4 Web サイトの構築

今回は、独自で Web サイトを構築した。なお、Web サイトをインターネット上に公開するには Vercel [15] というサービスを使用した。実際に構築した Web サイトの画面を図 3.23 に示す。



図 3.23: Web サイトの画面

このサイトでは、画像をアップロードして、パネルに表示することができる。サイト内で画像ファイルをアップロードするとその画像は Base64<sup>9</sup>形式にエンコードされ、Raspberry Pi で構築している Django Web アプリケーションに送信される。Django Web アプリケーションでは、常に REST API<sup>10</sup>でリクエストを受け付け、Web サイトから Base64 形式の画像を受け取ってデコードしたのち、実行ファイル（3.4.1 節で述べる、C 言語で作成したプログラム）を呼び出すようにした。実行ファイルによる処理内容は 3.4.1 節の通りである。これにより、Web サイトから画像をアップロードして、パネルに表示することができるようになった。

<sup>8</sup><https://blog.cloudflare.com/getting-cloudflare-tunnels-to-connect-to-the-cloudflare-network-with-quic> より引用

<sup>9</sup>Base64 は、画像データなどを英数字の並びに変換する方式である。画像を文字列として送受信する際に用いられる。

<sup>10</sup>REST API は、Web 上でデータを取得したり送信したりするときのやり取りの決まりに沿った方式である。ブラウザと Web サーバの間でデータのやり取りを行う際に広く用いられる。

### 3.4.5 Web サイトからの画像アップロード～パネル表示までの処理フロー

本システムでは、Web サイトから画像をアップロードして、パネルに表示するシステムを構築した。これにより、遠隔地から画像をアップロードしてパネルに表示することが可能となった。本システムの全体構成を図 3.24 に示す。

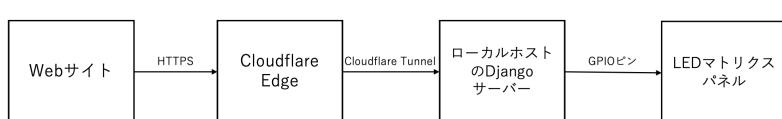


図 3.24: システム全体の構成

ユーザーは、PC やスマートフォンの Web ブラウザから Cloudflare Tunnel 経由で Django Web アプリケーションにアクセスし、画像をアップロードする。アップロードされた画像は、Raspberry Pi 4 上にある実行ファイルが実行され、3.4.1 節で説明した通りの処理をする。この一連の処理フローにより、遠隔地からでも LED マトリクスパネルの表示内容を制御することが可能となっている。以下、Web サイトでの操作からパネル表示までの流れを、図 3.25、図 3.26、図 3.27 を用いて詳しく説明する。



図 3.25: アップロード前の Web サイトの画面

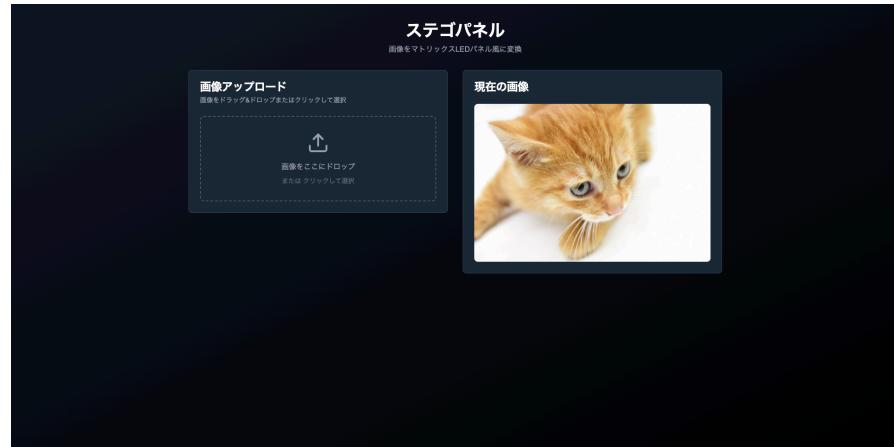


図 3.26: アップロード後の Web サイトの画面

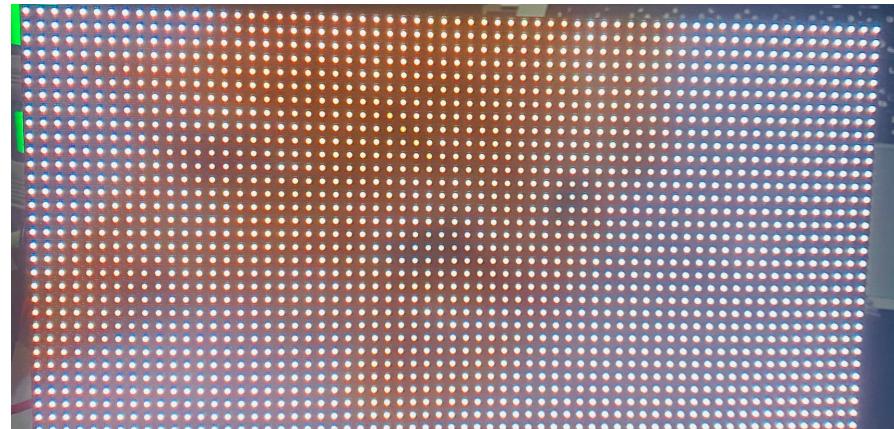


図 3.27: LED マトリクスパネルに表示された画像

図 3.25 は画像をアップロードする前の Web サイトの画面である。ユーザーはこの画面から画像ファイルを選択し、アップロード操作を行う。アップロードが完了すると、Web サイトの画面は図 3.26 のように更新される。この時点で、アップロードされた画像は Django Web アプリケーションを経て Raspberry Pi 上の実行ファイルに渡されている。図 3.27 は、上記の処理を経た画像が LED マトリクスパネルに実際に表示された結果である。Web サイトからアップロードした画像が、リサイズとガンマ補正を伴う点灯制御を経て、パネル上に表示されている様子が確認できる。以上が、Web サイトからの画像アップロードからパネル表示に至る一連の流れである。

# 第4章 2つの異なる点滅周波数を用いた可視光通信方式

第3章では、LEDマトリクスパネルによる256階調表示、ガンマ補正、および遠隔からの画像アップロードと表示を行うシステムの構築について述べた。本章で扱う可視光通信の方式（点灯パターンや分割領域の設計）は、表示システムの実装に直接依存するものではない。一方で、実用化後には同一のパネルで、任意の画像を表示しつつその上に情報を埋め込んだ可視光通信を行うことが想定される。したがって、第3章で整えた「表示として見やすい」基盤の上に、本章の通信方式を組み合わせることで、表示と通信の両立を目指す構成となっている。以下では、2つの異なる点滅周波数を用いた可視光通信方式の具体を述べる。

## 4.1 提案方式の概要

本節では、LEDマトリクスパネルを用いた可視光通信方式について説明する。提案する方式は、2つの異なる点滅周波数を用いてバイナリを表現してデータを送信する。また、一枚のLEDマトリクスパネルを複数の領域に分割し、一回で送信できるデータ量の増加を目指した。これらについて説明する。

## 4.2 可視光通信の仕組み

本節では、具体的な可視光通信の仕組みについて説明する。具体的には点灯パターンと分割領域の設計について説明する。

### 4.2.1 点灯パターンの設計

今回のシステムでは、2つの点滅周波数を用いて、バイナリを表現し、それを受信部で復号する方式を採用した。2つの点滅周波数を用いたバイナリの表現方法について説明する。

まず用語を定義する。スロットとは、時間軸を等間隔に区切った一区間であり、点灯と消灯とを切り替える最小の時間単位である。送信側では、このスロットごとにLEDを点灯するか消灯するかを決めてパターンを生成する。シンボルとは、1

ビット（0 または 1）を表現するために複数のスロットをまとめた単位である。本方式では 1 シンボルを 4 スロットで構成する。以上を踏まえ、以下で点灯パターンの具体を示す。

本方式では、1 シンボルを 4 スロットで構成し、スロットごとの点灯（1）と消灯（0）の並びで 2 種類のパターンを用いる。一方のパターンは 1010（1 スロットごとに点灯と消灯が交互に現れる）とし、他方のパターンは 1100（2 スロット点灯ののち 2 スロット消灯）とする。1010 は 4 スロットのあいだに点滅が 2 回生じるため相対的に高い点滅周波数となり、1100 は点滅が 1 回となるため相対的に低い点滅周波数となる。受信側では、観測した 4 スロットのパターンが 1010 に近ければビット 1、1100 に近ければビット 0 として復号する。このように、2 つの異なる点滅周波数（高・低）をビット 1・0 に対応させることでバイナリを表現する。

当初は、領域を 4 つに分割することを中心に検討していたため、全体を高い方の周波数で駆動させ、低い方の周波数領域は  $1/2$  になるように、2 回に 1 回点灯させるようにする方式を検討していた。しかし、そのやり方では、デューティ比が高い方の周波数の時は 100%，低い方の周波数の時は 50% となり、両者で非対称となる。これは、単位時間あたりで人間の目に入る光量が高周波領域の方が多くなってしまうため、人間の視覚的には激しいちらつきが発生してしまう。そのため、この方式は採用しなかった。（以下、この方式を「非対称デューティ比方式」と呼ぶ）それを考慮して新たに点灯パターンを設計した。具体的には、両方ともデューティ比が 50% となるようにする方式を採用した。

非対称デューティ比方式と本方式を比較したものを以下の図 4.1 に示す。

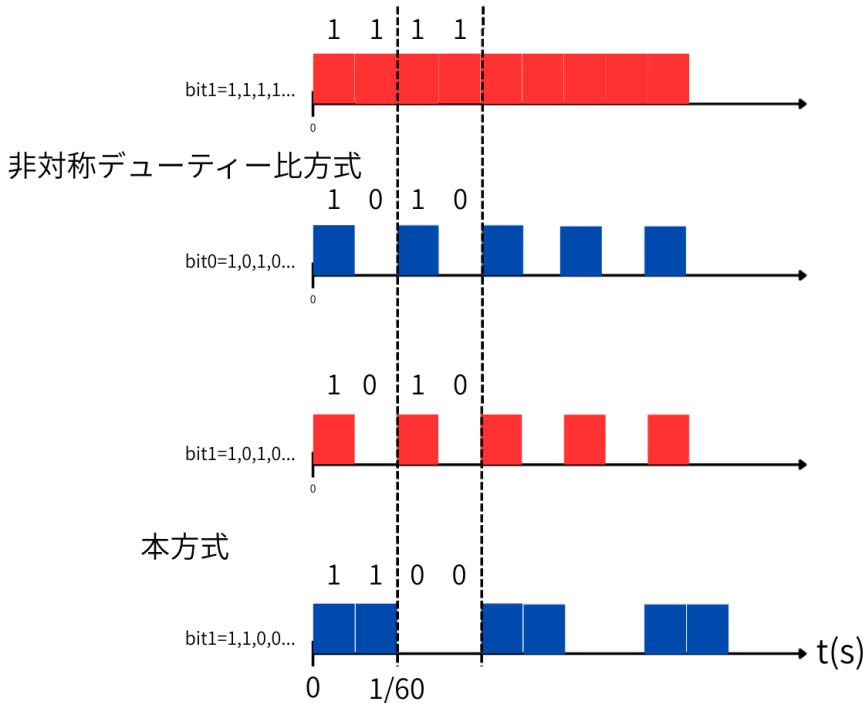


図 4.1: 提案方式におけるスロットパターン

図 4.1では、赤色の領域がビット1を表すスロットパターン、青色の領域がビット0を表すスロットパターンを示している。

非対称デューティ比方式（図上側）では、ビット1が1111（4スロットすべて点灯）、ビット0が1010（2スロット点灯）である。4スロット中の点灯数がビット1では4回、ビット0では2回と異なるため、デューティ比が100%と50%で非対称になる。一方、本方式（図下側）では、ビット1が1010、ビット0が1100であり、いずれも4スロット中で点灯が2回である。したがって両方ともデューティ比が50%となり、対称である。

本方式の2つのスロットパターン（1010と1100）は、4スロットで見たときの輝度が等しくなるように設計されている。以降、これらを「バランス符号」と呼ぶ。

このバランス符号を用いることで、ビット1とビット0のいずれもデューティ比50%となり、4スロットで見たときの輝度が揃うため、人間の視覚に違和感が生じにくい。

#### 4.2.2 バイナリ変換の方法

受信側では、3.2.1節で述べたバランス符号に従い、観測した4スロットのパターンが1010に近ければビット1、1100に近ければビット0として復号する。この処理を時間方向に繰り返し適用することで、点灯パターンからビット列を得る。例

として、1001というビット列を表現する場合のスロットパターンを図4.2に示す。

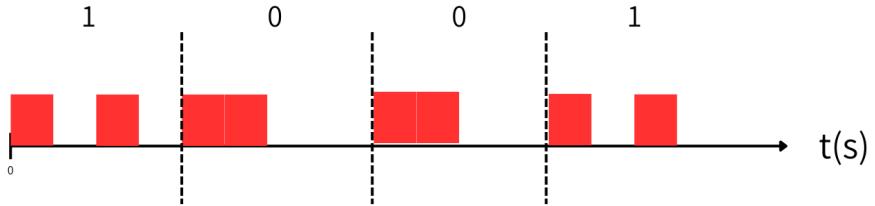


図4.2: 1001というビット列を表現する場合のスロットパターン

図4.2は、ビット列「1001」をバランス符号で表現した例である。左から順にビット1, 0, 0, 1に対応し、それぞれのシンボルが1010, 1100, 1100, 1010のスロットパターンになっている。送信側では、このようにビット列を4スロットごとのバランス符号に置き換え、時間方向に並べることで点灯パターンを生成する。

駆動周波数については、当初は全体を120Hz（1スロット=1/120s）で駆動した。このとき、ビット1のパターン1010は点灯と消灯が2スロットごとに繰り返すため、点滅の1周期は2スロット（=1/60s）であり、実質的な点滅周波数は60Hzとなる。一方、ビット0のパターン1100は4スロットで1周期となるため、点滅周波数は30Hzとなる。したがって、120Hz駆動ではビット1が60Hz、ビット0が30Hzという2つの点滅周波数で送信されていたことになる。しかし、この周波数では、人間の目にちらつきが発生してしまう。

人間の視覚には臨界フリッカ融合周波数（CFF: critical flicker fusion）があり、この周波数以上ではフリッカを感じず連続点灯に見える。CFFは条件により異なるが、およそ60Hzから100Hzとされる。そこで、高い方の点滅周波数が120Hz（CFFを上回り、連続点灯に見えやすい）となるように、全体を480Hzで駆動するようにした。1スロットは1/480sであり、4スロットで1シンボルとなる。

### 4.2.3 撮影画像における縞模様

後述する図4.5のように、点滅するLEDマトリクスピネルをカメラで撮影すると、画面上に縞模様が現れることがある。これは、多くのスマートフォンカメラが採用するローリングシャッター方式によるものである。ローリングシャッターでは、1回の撮影のあいだにセンサの読出しが上から下へ順に行われるため、画面上の縦位置によって光を取り込む時刻が異なる。送信側ではパネルがスロットごとに点灯・消灯を繰り返しているため、読出しの時刻が点灯区間と重なった行は明るく、消灯区間と重なった行は暗く記録される。その結果、画像上では明暗が縞状に並んで見える。実際に縞模様が現れた画像を図4.3に示す。



図 4.3: 縞模様が現れた画像

この縞模様は、1 フレーム内に時間方向の点滅情報が縦方向に展開されたものと解釈できる。したがって、縞のパターンを解析することで、カメラのフレームレート（例：60fps）で区切られた時間分解能を超えて、ビット列を推定できる可能性がある。今回は受信側の復号ロジックまで十分に設計・検証できていないが、今後の課題として、縞パターンからのビット推定や、フレームレートを上回るデータ受信の実現が考えられる。

#### 4.2.4 分割領域における点灯パターンの設計

ここでは、一枚の LED マトリクスパネルを 4 つの領域に分割し、それぞれの領域で異なる点滅周波数を用いて同一のデータを並列に送信する方式を採用した。

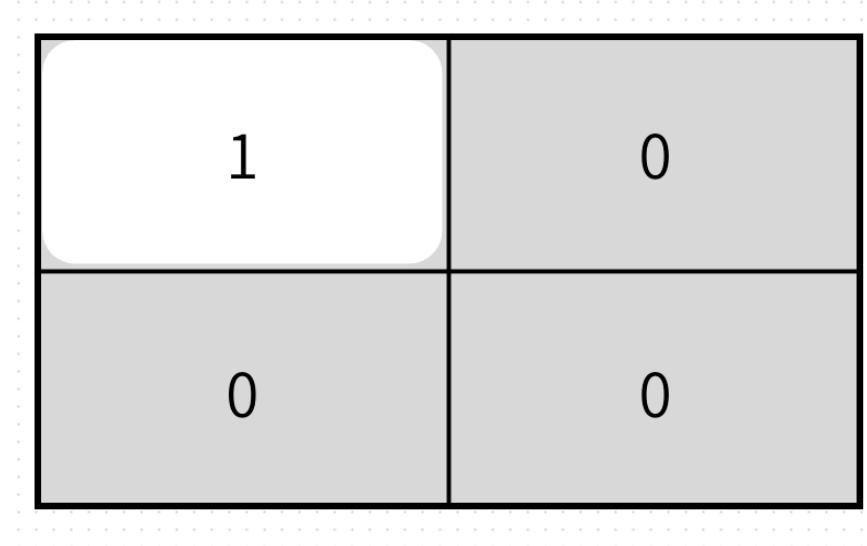


図 4.4: 4分割 LED マトリクスの走査制御

図 4.4 のような場合、受信側では 1001 というデータに変換することができる。このように、分割領域における点灯パターンを設計することで、一回で送信できるデータ量を増加させることができる。この方法で実際に点灯した画像を図 4.5 に示す。

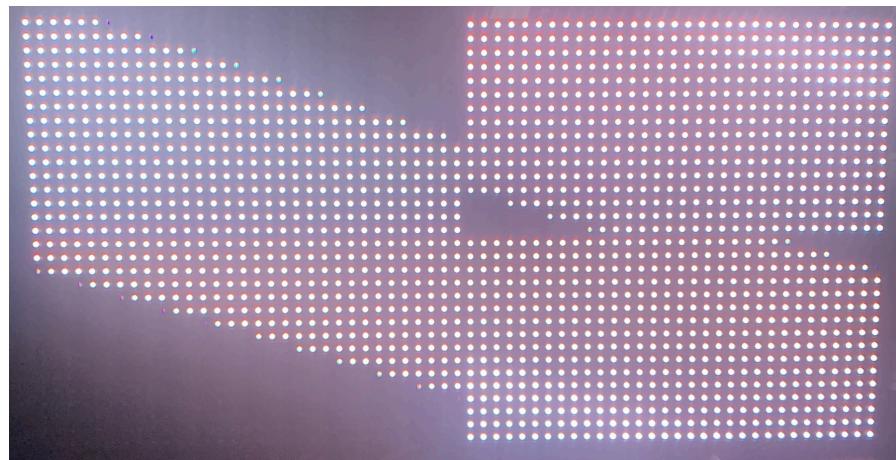


図 4.5: LED マトリクスパネルを 4 つの領域に分割した画像

これまでの 3.2.1 節および 3.2.2 節では、送信側の符号化（バランス符号でビットを 4 スロットの点灯パターンに置き換える）と、受信側の復号の原理（観測した 4 スロットのパターンから 1010 ならビット 1, 1100 ならビット 0 とする）を述べた。しかし、図 4.5 はカメラで撮影した 1 枚の写真である。一般にカメラの 1 フレームの露光時間は、1 スロット ( $1/480\text{s}$ ) より長い。そのため、1 枚の画像では複数スロット分の光が時間方向に積分され、点滅の時間変化が区別できない。し

たがって、この画像だけからはスロットごとの点灯パターンを読み取れず、ビットを推定することはできない。一方で、図 4.5 では縞模様が見える。

#### 4.2.5 4 分割以上の検証

受信側の実装が間に合わず、分割数の増加によって受信側の実装負担がどの程度増えるかを見積もれなかった。そこで、4 分割を前提に設計しつつ、8 分割についても試験的に検証した。

8 分割で点灯した場合、図 4.6 のようになる。

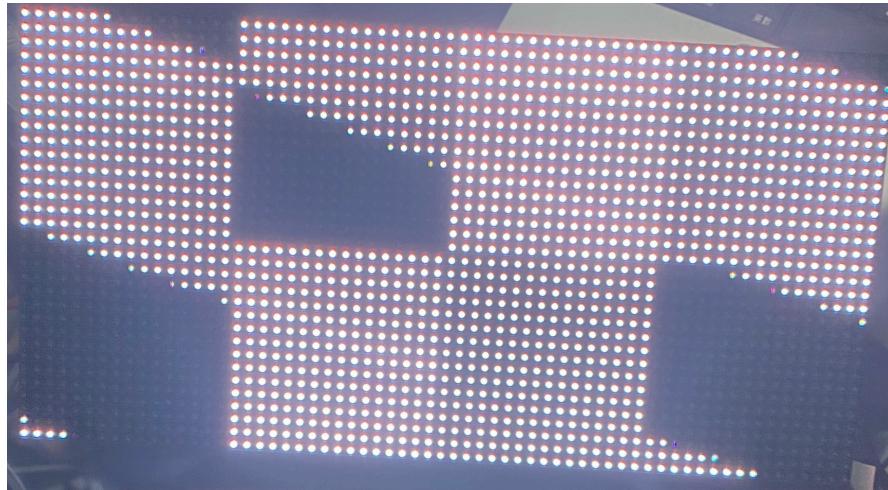


図 4.6: LED マトリクスパネルを 8 つの領域に分割した画像

このように、送信側の技術的には分割数を増やす余地がある。受信側で安定して読み取れるのであれば、分割数をさらに増やすことで、より高速な通信が実現できると考えられる。

### 4.3 フレーム構成と復号の設計

本節では、送信フレームの構成、FLAG 検出とシンボル境界推定、および受信アルゴリズムの方針について述べる。

#### 4.3.1 送信フレームの構成

送信するデータの全体設計は以下の図 4.7 の通りである。



図 4.7: 送信データの全体設計

図 4.7において、FLAG はデータの開始位置と終了位置を示すものであり、Payload は実際に送信するデータの内容であり、CRC はビット誤り検出を行うものである。この設計は HDLC のフレーム同期方式を参考にしている [16]。

### 4.3.2 FLAG 検出とシンボル境界推定

今回的方式では、FLAG を用いてデータの開始位置と終了位置を示す。今回、FLAG は 01111110 として設計したが、ペイロード内に 01111110 が含まれていた場合、そのビット列を FLAG として誤検出してしまう問題がある。そこで本研究ではビットスタッフィングを適用し、ペイロード中に 1 が 5 回連続して現れた場合、その直後に 0 を挿入する。受信側では、1 が 5 回連続した後に現れる 0 を取り除く（デスタッフィング）ことで元のペイロードを復元する。この処理により、ペイロード中に FLAG と同一のビット列が出現しないことが保証され、FLAG の検出が可能となる。

シンボル境界推定について、本研究ではバランス符号として 1100 を 0, 1010 を 1 に割り当て、1 シンボルを 4 スロットで構成する。受信側でシンボル境界（位相）がずれた場合、4 スロット窓で観測されるパターンは 1100 が 1001 や 0011 となるなど、送信時の並びと一致しなくなる。そのため、受信信号を 1 スロットずつシフトさせながら候補パターンとの整合を評価し、最も整合する位相をシンボル境界として採用する。特に 1010 は位相をずらしても 1010 または 0101 の交互パターンとなり、連続する 1 や 0 が生じにくい。この性質は位相推定における識別性を高める点で、本方式のシンボル設計の利点である。

ただし、カメラ撮像ノイズ等により、位相ずれではなく誤りによって 1001 のようなパターンが観測される可能性もある。そこで本研究では、ペイロード末尾に CRC を付与し、受信側で復号結果の整合性を検証する。具体的には、位相候補（0 ~ 3 スロットのシフト）ごとに復号を行い、CRC 検証に合格する候補を正しい復号結果として採用する。これにより、シンボル境界推定の誤りや伝送中のビット誤りを高確率で検出でき、誤ったデータ列を上位層へ渡すことを防ぐ。

### 4.3.3 受信アルゴリズム

本節では、スマートフォンカメラで撮影されたフレーム列から bit 列を復号するアルゴリズムについて述べる。

## フレーム列からの時系列データ生成

受信側では、60fps 程度で取得した動画をフレームに分解し、各フレームに対して4分割領域内の縞パターンを抽出する。本方式は4スロットで平均輝度が一致するように設計しているため、領域内の平均輝度からは bit 差が得られない。そのため、ローリングシャッターによって生じる縞のパターンからビットの推定を行う。得られた縞プロファイルを時間方向に並べることで、領域ごとの時系列データを生成する。

### bit パターンとの類似度評価と位相推定

送信側では1シンボルを4スロット（1スロット=1/480s）で構成しているため、受信側の 60fps では1シンボルあたりの観測点が不足する可能性がある。そこで、縞パターンから得られた時系列データに対して、理論パターン 1010 および 1100 に対応する縞の並びと一致するかを評価する。

また、ローリングシャッターやフレームレートの揺らぎによる位相ずれに対応するため、開始位置を1フレームずつずらして複数の位相候補を評価する。各位相で得られたスコアが最大となるパターンを選択し、1010 なら bit=1、1100 なら bit=0 として復号する。これにより、同一シンボル長でも撮影条件に依存しない復号が可能になる。

### 4.3.4 復号フロー全体

復号フローは、大きく「前処理」「FLAG 検出」「payload 復号」の3段階で構成される。まず前処理としてフレーム列から縞パターン時系列を生成し、正規化を行う。次に、FLAG パターンと一致する区間を探してフレーム境界を決定する。このとき、位相をずらした複数候補を評価し、最も一致度の高い位置を採用する。

境界が確定した後は、payload 領域に対してシンボルごとの類似度評価を行い、bit 列を復号する。誤検出が発生した場合は、FLAG 再検出に戻ることで再同期を行い、長い payload でも復号を継続できる構成とした。



# 第5章 考察および今後の課題

本章では、第3章および第4章で構築してきたLEDマトリクス表示システムおよびフリッカ周波数差を用いた可視光通信方式について、構成と方式の特徴を踏まえて考察し、今後の課題についてまとめる。

## 5.1 考察

今回LEDマトリクスピネルを用いて256階調での画像の表示、外部から画像をアップロードして表示を行えるシステムの構築、LEDマトリクスピネルを用いた可視光通信の実験を行った。第3章および第4章で述べた構成と方式の特徴から、表示品質と通信性能の間には明確なトレードオフがあることが分かる。非対称デューティ比方式ではデューティ比の差により周波数ごとの明るさが不均一となり、ちらつきと色味差が目立った。一方、本方式では4スロット平均で輝度が一致するように設計したため、人間の目には白色の揺らぎが少なくなり、表示装置としての違和感が大きく改善された。

周波数条件については、120Hz/240Hzでは低周波成分が残り、ちらつきが視認できたのに対し、480Hzではほぼ連続点灯に見えた。ただし、高周波化は縞パターンのピッチを細かくし、撮影条件によっては縞のコントラストが低下する可能性がある。すなわち、視覚的な快適さを優先すると通信の検出が難しくなりやすく、逆に検出性を優先するとちらつきが増えるという関係がある。

また、本方式は平均輝度が一致するため、領域内の平均輝度からはビット差が得られず、受信側はローリングシャッターによって生じる縞パターンを用いて復号する必要がある。この設計により視覚的品質は高いが、受信アルゴリズム側の前処理や位相推定に依存する部分が大きく、撮影距離や角度、露光条件の変化に対しても安定して復号ができるようにするという点が今後の課題である。

さらに、4領域並列化は通信速度を向上させる一方、1領域あたりの画素数が減少するため、縞パターンの判別が難しくなる場合がある。実運用を想定する場合は、距離や表示サイズに応じて分割数やシンボル長を調整する必要があると考えられる。

## 5.2 結論および今後の課題

本研究では、LED マトリクスパネルを用いた表示システムの構築と画質改善を行い、その上でフリッカ周波数差を用いた可視光通信方式を提案した。表示面では、256 階調表示とガンマ補正により視覚的な違和感を抑え、通信面では 4 分割領域と本研究で提案した点灯方式を用いることで、ちらつきを抑えつつ情報を埋め込める음을示した。一方で、受信側の実装と実測による通信性能評価は十分に行えておらず、縞パターン抽出や位相推定の頑健性を含めた検証が今後の課題である。

今後は、表示・通信・受信アルゴリズムを統合したシステムとして実装し、撮影距離や角度、露光条件に対する復号性能の評価を進める必要がある。また、誤り訂正符号やシンボル長の最適化、分割数の設計指針の整備を通じて、更なる実用性や安定性の向上を目指したい。

# 参考文献

- [1] 株式会社大塚商会, “Led 通信（可視光通信）とは.” <https://www.otsuka-shokai.co.jp/products/led/knowledge/vlc.html> (最終閲覧日: 2026-01-18) .
- [2] IEEE, “IEEE std 802.15.7-2018: IEEE standard for local and metropolitan area networks—part 15.7: Short-range optical wireless communications,” tech. rep., IEEE, 2018. <https://ieeexplore.ieee.org/document/8697198> (最終閲覧日: 2026-01-24) .
- [3] S. Rajagopal, R. D. Roberts, and S.-K. Lim, “IEEE 802.15.7 visible light communication: Modulation schemes and dimming support,” IEEE Communications Magazine, vol. 50, no. 3, pp. 72–82, Mar. 2012. <https://ieeexplore.ieee.org/document/6163585> (最終閲覧日: 2026-01-24) .
- [4] A. Jovicic, J. Li, and T. Richardson, “Visible light communication: Opportunities, challenges and the path to market,” IEEE Communications Magazine, vol. 51, no. 12, pp. 26–32, Dec. 2013. <https://ieeexplore.ieee.org/document/6685754> (最終閲覧日: 2026-01-24) .
- [5] I. 802.15, “Short-range optical wireless communications tutorial.” IEEE 802.15 Working Group, Mentor 資料. <https://mentor.ieee.org/802.15/dcn/15/15-15-0112-02-007a-short-range-optical-wireless-communications-tutorial.pdf> (最終閲覧日: 2026-01-24) . OCC (カメラ受信) を含む短距離光無線の整理.
- [6] I. 802.15, “IEEE 802.15 TG 15.7r1: Optical camera communications.” IEEE 802.15 Task Group. <https://www.ieee802.org/15/pub/TG7r1.html> (最終閲覧日: 2026-01-24) .
- [7] T. Nguyen, A. Islam, and T. Yamazato, “Optical camera communications for IoT—rolling-shutter based MIMO scheme with grouped LED array transmitter,” Sensors, vol. 20, no. 12, 2020. <https://www.mdpi.com/1424-8220/20/12/3361> (最終閲覧日: 2026-01-24) .
- [8] D. K. Nguyen *et al.*, “Spaced color shift keying modulation for camera-based visible light communication system us-

- ing rolling shutter effect,” Optics Communications, 2019.  
<https://www.sciencedirect.com/science/article/abs/pii/S0030401819304432> (最終閲覧日: 2026-01-24) .
- [9] R. D. Roberts, S. Rajagopal, and S.-K. Lim, “IEEE 802.15.7 physical layer summary,” IEEE GLOBECOM Workshops, pp. 772–776, 2011.  
<https://ieeexplore.ieee.org/document/6162558> (最終閲覧日: 2026-01-24) .
- [10] “Led マトリックス hub75 1/16s 1/32s smd2121 smd2020  
屋内用 p4 256x128mm 64x32 32x64 led モジュール.”  
<https://ja.aliexpress.com/item/1005010223788711.html> (最終閲覧日: 2026-01-24) .
- [11] スイッチサイエンス, “Lmcm (LED matrix controller multinode) .”  
<https://doc.switch-science.com/media/files/d2afe4e3-7003-4de2-97bc-78f818065a8f.pdf> (最終閲覧日: 2026-01-24) .
- [12] EIZO, “液晶ディスプレイの「ガンマ」を知ろう.” ITmedia 流液晶ディスプレイ  
講座 II 第7回, 2009. [https://www.eizo.co.jp/eizolibrary/other/itmedia02\\_07/](https://www.eizo.co.jp/eizolibrary/other/itmedia02_07/) (最終閲覧日: 2026-01-24) .
- [13] “Neon — serverless postgres.” <https://neon.com/> (最終閲覧日: 2026-01-24) .
- [14] Cloudflare, “Cloudflare tunnel.” <https://developers.cloudflare.com/cloudflare-one/networks/connectors/cloudflare-tunnel/> (最終閲覧日: 2026-01-24) .
- [15] “Vercel — build and deploy the best web experiences.” <https://vercel.com/> (最終閲覧日: 2026-01-24) .
- [16] LINEEYE, “通信の基本方式.” 通信の用語集.  
[https://www.lineeye.co.jp/html/term\\_doki.html](https://www.lineeye.co.jp/html/term_doki.html) (最終閲覧日: 2026-01-24) .

# 付録A LEDマトリクス表示用プログラム

本付録では、第3章で述べたLEDマトリクス表示システムの制御に用いたC言語プログラムのソースコードを示す。WiringPiによるGPIO制御、stb\_image系ライブラリによる画像読み込み・リサイズ、および累積値による256階調表示の実装を含む。

```
#define _GNU_SOURCE // POSIX関数を有効にする
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include <math.h>

// stb_image のインクルード
#define STB_IMAGE_IMPLEMENTATION
#include "stb_image.h"
#define STB_IMAGE_WRITE_IMPLEMENTATION
#include "stb_image_write.h"
#define STB_IMAGE_RESIZE_IMPLEMENTATION
#include "stb_image_resize2.h"

#define NEW_BMP_WIDTH    64
#define NEW_BMP_HEIGHT   32

// スロット周波数（設計値）
#define FREQ_LIGHTING   15360

// HUB75 64x32 は通常 1/16 スキャン（行アドレス0..15）
#define SCAN_ROWS        (NEW_BMP_HEIGHT / 2)

// 通常方式（上段/下段を同時点灯）なので *2 は不要
// 1スロット = 1アドレス行の点灯時間
static inline unsigned int calc_slot_time_us(void) {
    unsigned int t = 1000000u / (FREQ_LIGHTING * SCAN_ROWS);
```

```
if (t == 0) t = 1; // 念のため
return t;
}

typedef struct { uint8_t r, g, b; } RGB;

// 画像ファイル読み込み (PNG/JPEG/BMP 等、stb が自動判別)
unsigned char* load_image(const char* filename, int* out_w, int* out_h) {
    int w, h, ch;
    unsigned char *img = stbi_load(filename, &w, &h, &ch, 3);
    if (!img) {
        fprintf(stderr, "画像読み込み失敗: %s\n", stbi_failure_reason());
        return NULL;
    }
    *out_w = w;
    *out_h = h;
    return img;
}

// リサイズ (RGB 構造体配列で返す)
// ※ pow(x, 2.2) による単純ガンマデコード近似を使用
RGB* resize_image(unsigned char* src_rgb, int src_w, int src_h) {
    unsigned char *resized = (unsigned char*)malloc(NEW_BMP_WIDTH * NEW_BMP_HEIGHT * 3);
    if (!resized) return NULL;

    // sRGB を前提にしたリサイズ (ガンマ考慮)
    if (!stbir_resize_uint8_srgb(
        src_rgb, src_w, src_h, 0,
        resized, NEW_BMP_WIDTH, NEW_BMP_HEIGHT, 0, 3)) {
        fprintf(stderr, "リサイズに失敗\n");
        free(resized);
        return NULL;
    }

    RGB *out = (RGB*)malloc(NEW_BMP_WIDTH * NEW_BMP_HEIGHT * sizeof(RGB));
    if (!out) { free(resized); return NULL; }

    for (int i = 0; i < NEW_BMP_WIDTH * NEW_BMP_HEIGHT; i++) {
        double sr = resized[i*3 + 0] / 255.0;
        double sg = resized[i*3 + 1] / 255.0;
        double sb = resized[i*3 + 2] / 255.0;

        int r = (int)lround(pow(sr, 2.2) * 255.0);
        int g = (int)lround(pow(sg, 2.2) * 255.0);
        int b = (int)lround(pow(sb, 2.2) * 255.0);

        if (r < 0) r = 0; if (r > 255) r = 255;
    }
}
```

```
    if (g < 0) g = 0; if (g > 255) g = 255;
    if (b < 0) b = 0; if (b > 255) b = 255;

    out[i].r = (uint8_t)r;
    out[i].g = (uint8_t)g;
    out[i].b = (uint8_t)b;
}

free(resized);
return out;
}

// 点灯判定（累積ディザ / 時間方向誤差拡散）
// 256 スロット中に v 回だけ ON になるよう、スロット k で ON かを返す。
// 判定は「床関数差分」: floor(v*(k+1)/256) != floor(v*k/256)
static inline bool accum_dither_on_u8(uint8_t v, uint8_t k) {
    uint16_t a = (uint16_t)v * (uint16_t)(k + 1);
    uint16_t b = (uint16_t)v * (uint16_t)k;
    return (a >> 8) != (b >> 8);
}

// 通常の HUB75 制御（上段/下段 同時点灯）
void control_led_matrix(RGB* pixels) {
    // ピン定義（BCM GPIO 番号）
    const int pinA      = 22;
    const int pinB      = 23;
    const int pinC      = 24;
    const int pinD      = 25;
    const int pinR1     = 11;
    const int pinR2     = 8;
    const int pinG1     = 27;
    const int pinG2     = 9;
    const int pinB1     = 7;
    const int pinB2     = 10;
    const int pinClock  = 17;
    const int pinOE     = 18;
    const int pinLAT    = 4;

    int pins[] = {pinA, pinB, pinC, pinD,
                  pinR1, pinR2, pinG1, pinG2, pinB1, pinB2,
                  pinClock, pinOE, pinLAT};

    for (int p = 0; p < (int)(sizeof(pins)/sizeof(pins[0])); p++) {
        pinMode(pins[p], OUTPUT);
        digitalWrite(pins[p], LOW);
    }
}
```

```
// OE は HIGH で消灯（一般的な極性）
digitalWrite(pinOE, HIGH);

printf("LED マトリクス制御開始...\n");

const unsigned int slotTimeUs = calc_slot_time_us();
printf("slotTime = %u us\n", slotTimeUs);

uint8_t k = 0; // 0..255

for (long long n = 0; n < 1000000000LL; n++) {

    // ★この1回のnで「スロット k」を表示する:kはフレーム中固定（縛対策）

    for (int address = 0; address < SCAN_ROWS; address++) {

        // 出力無効化（アドレス変更・シフト中は消灯）
        digitalWrite(pinOE, HIGH);

        // 行アドレス設定 (0..15)
        digitalWrite(pinA, (address & 0x1) ? HIGH : LOW);
        digitalWrite(pinB, (address & 0x2) ? HIGH : LOW);
        digitalWrite(pinC, (address & 0x4) ? HIGH : LOW);
        digitalWrite(pinD, (address & 0x8) ? HIGH : LOW);

        const int topRow = address;           // 0..15
        const int botRow = address + SCAN_ROWS; // 16..31

        // 1行分のシフト (64列)
        for (int x = 0; x < NEW_BMP_WIDTH; x++) {

            // 前列の残りを消す
            digitalWrite(pinR1, LOW);
            digitalWrite(pinG1, LOW);
            digitalWrite(pinB1, LOW);
            digitalWrite(pinR2, LOW);
            digitalWrite(pinG2, LOW);
            digitalWrite(pinB2, LOW);

            RGB t = pixels[topRow * NEW_BMP_WIDTH + x];
            RGB b = pixels[botRow * NEW_BMP_WIDTH + x];

            // 上段 (R1/G1/B1)
            if (accum_dither_on_u8(t.r, k)) digitalWrite(pinR1, HIGH);
            if (accum_dither_on_u8(t.g, k)) digitalWrite(pinG1, HIGH);
            if (accum_dither_on_u8(t.b, k)) digitalWrite(pinB1, HIGH);
```

```
// 下段 (R2/G2/B2)
if (accum_dither_on_u8(b.r, k)) digitalWrite(pinR2, HIGH);
if (accum_dither_on_u8(b.g, k)) digitalWrite(pinG2, HIGH);
if (accum_dither_on_u8(b.b, k)) digitalWrite(pinB2, HIGH);

// シフトクロック
digitalWrite(pinClock, HIGH);
digitalWrite(pinClock, LOW);
}

// ラッチ
digitalWrite(pinLAT, HIGH);
digitalWrite(pinLAT, LOW);

// 出力有効化 (点灯)
digitalWrite(pinOE, LOW);

delayMicroseconds(slotTimeUs);

// 次の行へ行く前に確実に消灯
digitalWrite(pinOE, HIGH);
}

// ★フレーム (16 行) 完了後に k を進める
k++;
}

// 念のため消灯
digitalWrite(pinOE, HIGH);
}

int main(int argc, char *argv[]) {
if (argc != 2) {
    fprintf(stderr, "使用方法: %s 入力画像\n", argv[0]);
    return 1;
}

if (wiringPiSetupGpio() == -1) {
    fprintf(stderr, "WiringPi の初期化に失敗\n");
    return 1;
}

int w, h;
unsigned char *raw = load_image(argv[1], &w, &h);
if (!raw) return 1;

RGB *resized = resize_image(raw, w, h);
```

```
    free(raw);
    if (!resized) return 1;

    control_led_matrix(resized);

    free(resized);
    return 0;
}
```

# 謝辞

2026年3月 園邊 翔大



# 研究業績リスト

【公表論文】 【国際会議】