Vol.2012-MBL-64 No.3
Vol.2012-ITS-51 No.3
2012/11/15

ユビキタス機器から構成される電飾アートのプログラム自動生成の集中・分散制御調整について

長岡 佑典[1]　佐野 渉二[1]　寺田 努[1,2]　塚本 昌彦[1]

概要：近年，発光ダイオード (LED: Light Emitting Diode) からなるイルミネーションやディスプレイなどを用いた電飾アートが注目されている．大規模な電飾を光らせるためには大量の LED を多くのマイコンを用いて制御する必要がある．筆者らは，これまでに複数のユビキタス機器を統合的に扱うマクロプログラミングモデルを提案し，これを電飾アートに適用してきた．マクロプログラミングシステムでは，複数あるシステム要件を全て満たすことは困難なため，個々のデバイスの制御方式を変えることが求められる．そこで，本稿では，多数のユビキタスデバイスの制御方式として従来の集中型制御方式に加え，分散型制御方式，中間型制御方式を設計，実装し，制御の集中・分散度の調整をパラメタにより行うシステムを実装した．

Adjustment Centralization and Distribution Control of Generation Program in Illumination with Multiple Ubiquitous Devices

Abstract: Recently, illuminations and information displays using LEDs have attracted a great deal of attention. To construct such illuminations, it is necessary to control many microcomputers connected to LEDs. We designed a macroprogramming model for controlling multiple ubiquitous devices. In this paper, we propose three type controlling methods for satisfying some system requirements, and implement a system for adjustment the degree of centralization and distribution of controlling with some parameters.

1. はじめに

近年，さまざまなモノにコンピュータを内蔵して利用するユビキタスコンピューティング [1] が注目されている．ユビキタスコンピューティングでは，複数のセンサや LED などの入出力機器を制御するデバイス (以下，ユビキタスデバイス) を連携させることで高度な機能を実現する．多数のユビキタスデバイスを統合的に制御する 1 つの手法としてはマクロプログラミング [2], [3], [4], [5], [6] がある．マクロプログラミングシステムでは，ユビキタスデバイス全体に対するプログラムをマクロな視点で記述でき，個々のユビキタスデバイスの動作を考慮しなくてもよいため，使用者はユビキタスデバイス全体に対する動作記述に注力できる．

一方で，近年発光ダイオード (LED: Light Emitting Diode) からなるイルミネーションやディスプレイなどを用いた電飾アートが注目されている．しかし，大規模な電飾アートをマイコンなどを用いて柔軟に制御するためには，多くのマイコンを統合的に制御する必要があり，特に 1 つ 1 つのマイコンにプログラムを書く必要がある場合，電飾の発光パターンの変更には多大な労力と時間がかかる．

筆者らはこれまで，電飾アートで複雑な光り方をより簡単に制御するためのマクロプログラミングモデルを設計してきた．設計したモデルでは，ユビキタスデバイスの位置関係に基づくセンサや LED などの入出力機器の制御を記述するだけで，環境内に存在する複数のユビキタスデバイス向けのプログラムを自動生成し，配置・実行する．設計モデルにより，電飾アート全体としての動作をマクロな視点で記述でき，デザイナなどの分散プログラミングに親しんでいないユーザでも電飾アートの表現記述に注力できる．しかし，ユーザによって電飾アートに求める点は異なる．例えば，通信量の減少や点灯時の見栄えの良さ，各デバイスの消費電力のバランスなどが考えられる．筆者らが

[1] 神戸大学大学院工学研究科
Graduate School of Engineering, Kobe University
[2] 科学技術振興機構さきがけ
PRESTO, Japan Science and Technology Agency

Vol.2012-MBL-64 No.3
Vol.2012-ITS-51 No.3
2012/11/15

これまでに設計したモデルでは複数のユビキタスデバイスの制御方法は 1 通りであり，これらの要件に柔軟に対応することは困難であった．そこで本稿では，複数のユビキタスデバイスの制御方式として異なる特徴を持った 3 つの制御方式を設計，実装し，ユーザの用途に適した制御方式を自動で選択，実行するシステムの実装を行った．また，実際にいくつかのプログラムを各制御方式で実行し，ユビキタスデバイスの通信量，LED 点灯の同時性，消費電力のバランスの 3 点を定量的に計測し，評価・考察を行った．

本稿は以下のように構成されている．第 2 章で関連研究について述べ，第 3 章でマクロプログラミングモデルの設計について説明する．第 4 章で評価・考察を行い，最後に第 5 章でまとめを行う．

2. 関連研究

多くのコンピュータの制御を 1 つのプログラムで行うマクロプログラミングの研究はこれまでにも多数行われている．Gumadi らの Kairos[3] は，多くのコンピュータに対してマクロな視点で個々のコンピュータを制御することに着目し，複数コンピュータにまたがる制御やコンピュータ間のネットワークトポロジを用いた制御を単一のプログラム上で記述できる．Newton らの Regiment[4] では，各コンピュータが取得するセンサなどのデータを関数型プログラミングの記述で扱える．Bischoff などの RuleCaster[5] は，簡単なルール形式の記述により，多くのコンピュータにまたがる処理を行うことができる．これらの研究では，システムはコンピュータ群に対して記述したプログラムを個々のコンピュータ用のプログラムに変換，分配する機能を有し，使用者はコンピュータ全体に対する記述をするだけでよい．

大規模な電飾アートを容易に制御可能なシステムとして，木下らの電飾アートの制御支援システム [7] がある．このシステムでは，大規模な電飾アートを小単位に分割して分散制御し光り方の変更や故障に対する柔軟性を高めている．また，簡単なコマンド入力により，直観的に LED の点滅パターンの設計を行える．一方で，出力のみの制御であるため，センサなどを用いた入力による LED の点滅パターンの設計ができないといった問題がある．大量の LED を用いた電飾の制御として，中田らのプロジェクタとユビキタス光デバイスを用いた電飾制御 [8] がある．これは，プロジェクタから照射された光を，ユビキタスデバイス上の光センサが感知し，LED を点灯させている．点滅パターンの変更は，プロジェクタで照射する光のパターンを PC で変更するだけで行えるため，プログラムの書き換えやユビキタスデバイス間の通信が必要ない．また，プロジェクタの照射できる範囲であればユビキタス光デバイスの制御が行えるため，数百個，数千個の LED の一斉制御ができる．しかし，光センサには指向性があるため，プロジェクタの照射角度や照射光の強弱の影響を受けやすく，設置条件が厳しいという問題がある．さらに，球体などプロジェクタ光の影になる部分を光らせたい場合やプロジェクタの照射範囲を越えるような大規模なもので用いることも難しい．

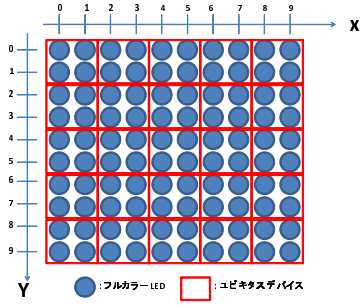

図 1 RGB フルカラー LED と一体となったユビキタスデバイスを格子状に並べた電飾アート

3. マクロプログラミングモデルの設計

本研究では，電飾アートとして図 1 のように複数のユビキタスデバイスを利用して格子状に並べられた多くのフルカラー LED を想定する．大規模なイルミネーションを複数の LED と 1 つのマイクロコンピュータが搭載されたユニットを複数使用することにより，分散制御が可能となる．このような分散制御のメリットとして，イルミネーションのサイズや形の変更が容易となることやイルミネーションが故障しても取り換えが容易にできるため，コストがかからないといったことが挙げられる．しかし，多くのマイクロコンピュータを制御する必要があり，プログラマが 1 つ 1 つのマイクロコンピュータにプログラムする手法では多大な労力と時間を要するといった問題点がある．そこで筆者らはマクロプログラミングを用いてきた．マクロプログラミング (図 2) とは，ユビキタスデバイス群全体に対する単一のプログラムを記述するだけで多数のユビキタスデバイスを制御するプログラミング手法であり，プログラマはユビキタスデバイス群をマクロな視点で制御することができる．マクロプログラミング環境では，ユビキタスデバイス群全体に対して記述されたプログラムから個々のユビキタスデバイスで動作するプログラムを生成して実行される．このユビキタスデバイス群全体に対するプログラムをグローバルプログラムと呼び，個々のユビキタスデバイスで動作するプログラムをローカルプログラムと呼ぶ．電飾アートで求められる以下の要件を満たしながらローカルプログラムを生成する必要がある．

- LED 点灯の同時性
  複数のユビキタスデバイスで 1 つの電飾アートを制御するため，異なるユビキタスデバイス上の複数の LED を同時に光らせたい場合，LED 点灯のタイミングを合わせる必要がある．特に，頻繁に点滅を繰り返すような表現の場合，少しでも点灯のタイミングがずれると

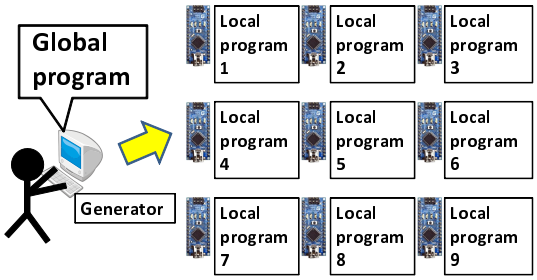

図 2 マクロプログラミング

電飾アートの表現力が低下する可能性がある．そのため，動作のタイミングを常にそろえる同時性が求められる．

- 通信量の削減
  ユビキタスデバイス間で協調して動作を行うためには，メッセージを通信する必要がある．しかし，一般にユビキタスデバイスは電池駆動であるため，通信量が多くなると，通信に要する消費電力が大きくなり，長時間駆動に適さない．このため，通信量を削減することが求められる．
- 消費電力量のバランス
  バッテリー切れにより，電飾アートの一部が動作しなくなった場合，表現力が低下するだけでなく，毎回修理が必要となり，コストがかかるという問題が挙げられる．そこで，各デバイスの消費電力量を可能な限り均等にする必要がある．

3.1 電飾アートのためのプログラミング言語設計

電飾アートを作成するためのプログラミング言語として Processing[10] に似せた言語を用いる．Processing はビジュアルデザインによく用いられる言語であり，プログラミングに慣れていないデザイナなどでもグラフィックを作成するプログラムを容易に記述できる．本研究では，LED の位置に基づいて線や三角，四角，丸のような図形を描く関数を用いて LED 群を制御する．これらの関数には Processing であらかじめ用意されている関数をマクロプログラミング向けに拡張したものや光の制御を直観的に行うために新たに作成した関数が含まれる．表 1 に関数内のパラメータについて，表 2 に関数と機能の一覧を示す．

3.2 マクロプログラミングのための制御方式

提案するマクロプログラミングモデルでは，コンパイラがグローバルプログラムからローカルプログラムを生成する．この時，上述のイルミネーション制御における要件を考慮する必要がある．しかし，すべての要件を満たすのは困難であり，重視する要件に応じて適切な制御方式を使用する必要がある．本稿では集中型制御方式，分散型制御方式，中間型制御方式の 3 つの制御方式を提案する．

表 1 関数内のパラメータ

deviceID	ユビキタスデバイスの ID 番号
x	lED の X 座標
y	LED の Y 座標
width	長方形の幅
height	長方形の高さ
r	円の半径
color	LED の色 (0:白 1:赤 2:緑 3:青 4:桃 5:黄 6:水色)

表 2 関数一覧

関数	機能
setposition(*deviceID,x,y*)	指定した ID のデバイスを座標 (x,y) に設定する．
colorMode(*color*)	点灯する LED の色を color で設定する．
point(*x, y*)	点 (x,y) に位置するデバイスの LED を点灯させる．
line(*x1, y1, x2, y2*)	点 (x1,y1) と点 (x2,y2) を結ぶ線上に位置するデバイスの LED を点灯させる．
triangle(*x1, y1, x2, y2,*	(x1,y1)，(x2,y2)，(x3,y3) の 3 点を結ぶ三角形上に位置するデバイスの LED を点灯させる．
rect(*x, y, width, height*)	点 (x,y) を左上の角として幅 width，高さ height の長方形上に位置するデバイスの LED を点灯させる．
circle(*x, y, r*)	点 (x,y) を中心とした半径 r の円上に位置するデバイスの LED を点灯させる．
fill(*color*)	図形を塗りつぶす色を color で設定する．
noFill()	図形の塗りつぶしを無効にする．
stroke(*color*)	図形の周りの境界上の色を color で設定する．
noStroke()	境界線を無効にする．
low(*x,y*)	点 (x,y) に位置するデバイスの LED を消灯させる．
allHIGH(*color*)	すべての LED を color の色で点灯させる．
allLOW(*color* or ALL)	指定の色のすべての LED を消す．ALL の場合は色に関係なく全ての LED を消す．
read(*x,y*)	点 (x,y) に位置するデバイスのセンサの値を読み取る．

3.2.1 集中型制御方式

1 つのデバイスでグローバルプログラムの処理の流れを管理し，命令を送ることで他のユビキタスデバイスの入出力を制御する方式を集中型制御方式と呼ぶ．グローバルプログラムの処理の流れを管理するユビキタスデバイスをマスタデバイス，それ以外のユビキタスデバイスをスレーブデバイスとする．マスタデバイスはグローバルプログラムに沿って処理を行い，スレーブデバイスの入出力を制御する場合はそのデバイスに命令を送る．スレーブデバイスはマスタデバイスから命令を受けるとその命令に応じて入出力を制御する．

3.2.2 分散型制御方式

各ユビキタスデバイスが独立して入出力を制御する方式

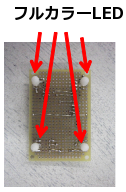

図 3　フルカラー LED 面

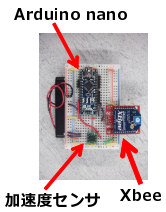

図 4　Arduino nano 面

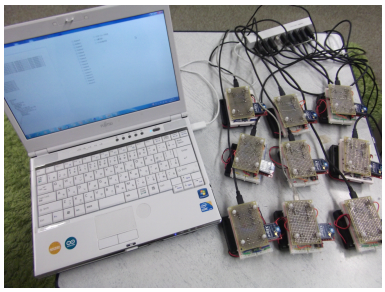

図 5　電飾アートの制御風景

を分散型制御方式と呼ぶ．他のユビキタスデバイスのセンサデータなどの入力値が必要な場合はそのデバイスからメッセージを送ってもらうことで取得する．分散型制御方式では，集中型制御方式のマスタデバイスのように全体の処理を管理するデバイスはなく，各ユビキタスデバイスがそれぞれ必要に応じて同期をとりながら処理を行うことで，ユビキタスデバイス全体の動作としてグローバルプログラムで記述された通りに処理が行われているように見える．

3.2.3　中間型制御方式

集中型制御方式は LED 点灯の同時性を重視した場合に適しているが，通信量の削減には適していない．一方で，分散型制御方式は通信量の削減に適しているが，LED 点灯の同時性を重視する場合は適していない．このように集中型制御方式と分散型制御方式は対極にある．中間型制御方式はこれら 2 つの制御方式の間をとった制御方式である．

3.3　実装

本研究では，ユビキタスデバイスとして図 3，4 に示すようなフルカラー LED4 個と加速度センサ，無線通信モジュールの Xbee を Arduino nano[11] に搭載したものを用いる．Arduino nano とは，マイクロコントローラ (AVR) と入出力ピンなどを搭載した基板である．加速度センサは電飾アートへのインタラクションのために用いる．各ユビキタスデバイス上の Arduino nano の ID はあらかじめ割り当てられているものとし，ユビキタスデバイスへのプログラムのアップロードは図 5 のように 1 つの PC で行う．

3.4　実行環境

マクロプログラミングでは，グローバルプログラムで個々のユビキタスデバイスを動作させるために，グローバルプログラムから個々のユビキタスデバイス用のローカルプログラムを生成する必要がある．そこで，本研究では，言語処理をし，各ユビキタスデバイス用のプログラムを生成するジェネレータを C ＃を用いて作成した．図 6 に提案モデルのインタフェースを示す．このインタフェースにより，ユーザは制御方式を自由に設定できる．例えば，青色のパラメータのタブを左端に設定した場合，LED 点灯の同時性を最も重視した集中型制御方式が実行される．一方で，タブを右端に設定した場合，通信量の削減を最も重視した分散型制御方式が実行される．それ以外の場合は中間型制御方式が実行される．以下，各制御方式でのローカルプログラムへの生成方法について説明する．

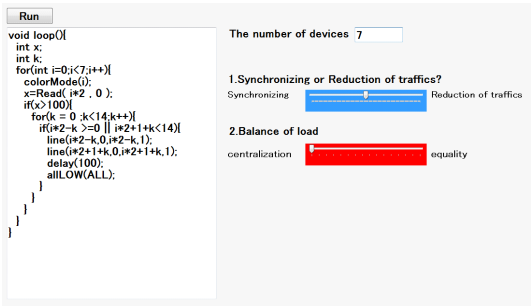

図 6　インタフェース画面

- 集中型制御方式
  マスタデバイス用のローカルプログラムとしては，グローバルプログラムに沿ったプログラムに入出力の関数部分が変換されたプログラムが生成される．いくつかの LED を点灯させるプログラムを記述した場合，マスタデバイスがユーザがあらかじめ設定したデバイスの ID と位置座標の関係により点灯させる LED の ID(localID) とそれを有するデバイスの ID(deviceID) を取得する．localID はユビキタスデバイス上の 4 つの LED を識別するための ID であり，左下，右下，左上，右上の順に 0，1，2，3 とした．ここで，取得した deviceID がマスタデバイス自身の deviceID と一致した場合は，特定の LED を制御する．一方で，取得した deviceID がスレーブデバイスの deviceID である場合は，そのスレーブデバイスに LED を制御させるメッセージを送る．スレーブデバイス用のローカルプログラムとしては，マスタデバイスから送られたメッセージに応じて，localID の LED を制御する．3.1 節で説明した関数から集中型制御方式で各ユビキタスデバイス用に生成されるプログラムの主なものを表 3 にまとめた．センサデータなどの入力を扱う場合は，マスタデバイスが入力値を取得したいセンサを有するデバイスを判別し，それがマスタデバイス自身であれば入力制御を行い，スレーブデバイスであれば入力制御と値の送信をさせるメッセージを送る．スレーブデバイスはマスタデバイスからの命令を受けると，入力データ

表 3　集中型制御方式で関数を生成したプログラム

関数	マスタデバイス（上段） スレーブデバイス（下段）

point(*x, y, color*)	deviceID=getdeviceID_point(x, y);
	localID=getlocalID_point(x, y);
	if(deviceID==MASTER_ID)
	lighting(color, localID);
	else
	Serial.write(data(command,
	deviceID, color, localID));

	case command(lighting): lighting(color(data), localID(data)); break;

line(*x1, y1, x2, y2, color*)	deviceID=getdeviceID_line(
	x1, y1, x2, y2);
	localID=getlocalID_line(
	x1, y1, x2, y2);
	for(int i=0; i num; i++){
	if(deviceID[i]==MASTER_ID)
	lighting(color, localID[i]);
	else
	Serial.write(data(command,
	deviceID[i], color, localID[i]));
	}

	case command(lighting): lighting(color(data), localID(data)); break;

Read(*x, y*)	deviceID=getdeviceID_point(x, y);
	if(deviceID==MASTER_ID)
	value=analogRead(pin);
	else
	value=request_data(ID);

	case command(analogRead): value=analogRead(pin); Serial.write(data(value)); break;

を送り返す．集中型制御方式でのプログラム生成の例を図 7 に示す．この例では，LED4 × 4 の電飾アートの line を点灯させるというプログラムを記述した．記述したプログラムの関数部分 (2, 4 行目) を読み取り，マスタデバイスと各スレーブデバイス用のローカルプログラムを生成する．関数部分以外の記述したプログラム (1, 3, 5 行目) は，マスタデバイスのみに書き込まれる．スレーブデバイスの 1, 2, 3, 7, 8 行目はあらかじめ書かれるプログラムである．

- 分散型制御方式

  3.1 節で説明した関数の中からいくつかを例として分散型制御方式で各ユビキタスデバイス用に生成したプログラムを表 4 にまとめた．図形上にある LED を光らせる場合は，環境内のすべてのユビキタスデバイスが関数 getdeviceID と getlocalID で図形上にある LED を有するユビキタスデバイスの deviceID と LED

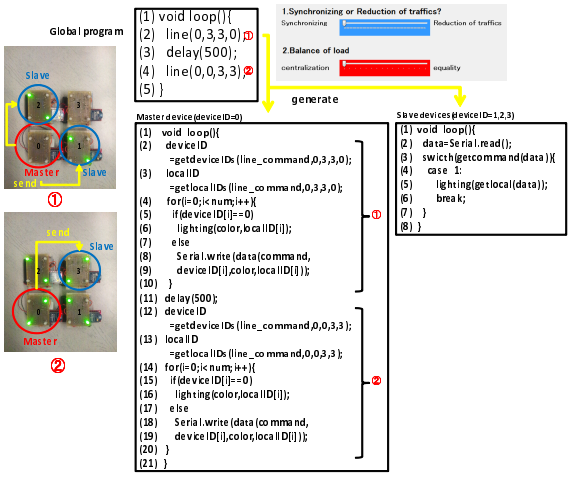

図 7　プログラム生成（集中型制御方式）

表 4　分散型・中間型制御方式で関数を生成したプログラム

関数	ユビキタスデバイス (deviceID = n)

point(*x, y*)	deviceID=getdeviceID_point(x,y);
	localID=getlocalID_point(x, y);
	if(deviceID==n)
	lighting(color, localID);

line(*x1, y1, x2, y2*)	deviceID=getdeviceID_line(
	x1, y1, x2, y2);
	localID=getlocalID_line(
	x1, y1, x2, y2);
	for(int i=0; i < num; i++){
	if(deviceID[i]==n)
	lighting(color, localID[i]);
	}

Read(*x, y*)	deviceID=getdeviceID_point(x,y);
	if(deviceID==n){
	value=analogRead(pin);
	send_data(value);
	}
	else
	value = receive_data();

の localID を取得し，取得した deviceID が自ユビキタスデバイスの deviceID と一致したときに点灯させる．センサデータなどの入力を扱う場合は，各ユビキタスデバイスがセンサデータを読み取るユビキタスデバイスの deviceID を取得し，その deviceID が自身の deviceID と一致した時にセンサデータを読み取り，その他のユビキタスデバイスに入力データを送信する．deviceID が一致しないユビキタスデバイスは入力データを受信するまで待つ．分散型制御方式でのプログラム生成の例を図 8 に示す．記述したプログラムの関数部分 (2, 4 行目) を読み取り，各ユビキタスデバイス用のプローカルログラムを生成する．関数部分以外の記述したプログラム (1, 3, 5 行目) は，環境内のユビキタスデバイスすべてに書き込まれる．また，1 つのユ

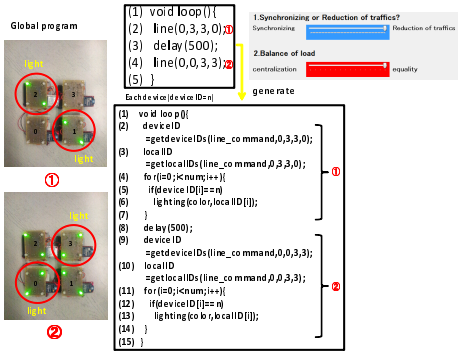

図 8 プログラム生成（分散型制御方式）

ビキタスデバイスの 1 行目には動き始めの命令を送る処理，他のユビキタスデバイスの 1 行目には命令を待つ処理がそれぞれ書かれる．

- 中間型制御方式

プログラムの生成ルールは分散型制御方式と同様である．しかし，パラメータの設定によってプログラムの先頭や delay 関数の後に同期を行うプログラムが生成される．本稿では，異なるパラメータ設定の 3 つの例を挙げ，それぞれ生成されるローカルプログラムについて説明する．1 つ目の例は同時性を重視し，消費電力の均一化を全く考慮しないパラメータ設定である（図 9）．マスタデバイスはローカルプログラムの 2, 10 行目でスレーブデバイスに同期のためのメッセージを送信する．一方で，スレーブデバイスはローカルプログラムの 2, 10 行目はマスタデバイスからのメッセージを受信するまで待つ．これにより，ユビキタスデバイス間で同期をとる．この例では，グローバルプログラム 1 ループにつき 2 回同期を行う．2 つ目の例は通信量の削減を重視し，消費電力の均一化を全く考慮しないパラメータ設定である（図 10）．マスタデバイスはローカルプログラムの 2 行目でスレーブデバイスに同期のためのメッセージを送信し，スレーブデバイスはローカルプログラムの 2 行目でマスタデバイスからのメッセージを受信するまで待つ．マスタデバイスとスレーブデバイスのローカルプログラムの 3, 17 行目はプログラムを 2 回ループさせるため，2 ループにつき 1 回同期を行う．最後に，3 つ目の例は同時性と通信量の削減を同等に重視し，消費電力の均一化を考慮したパラメータ設定である（図 11）．各デバイスの 2 ～ 5 行目で同期回数が自身の deviceID と一致した時にその他のデバイスに同期のためのメッセージを送信し，一致しない時はメッセージを受信するまで待つ．6 ～ 8 行目は同期回数をカウントするプログラムである．この例では，グローバルプログラム 1 ループにつき 1 回の同期制御を行い，同期のたびにメッセージを送信する役割を環境内のデバイスに順番に交代させる．これにより，各デバイスでの処理の負荷が平等になり，消費電力が均一になる．

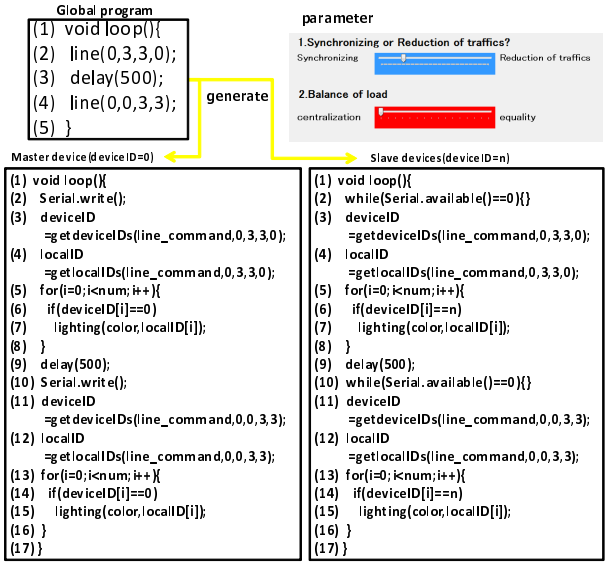

図 9 プログラム生成（中間型制御方式 1）

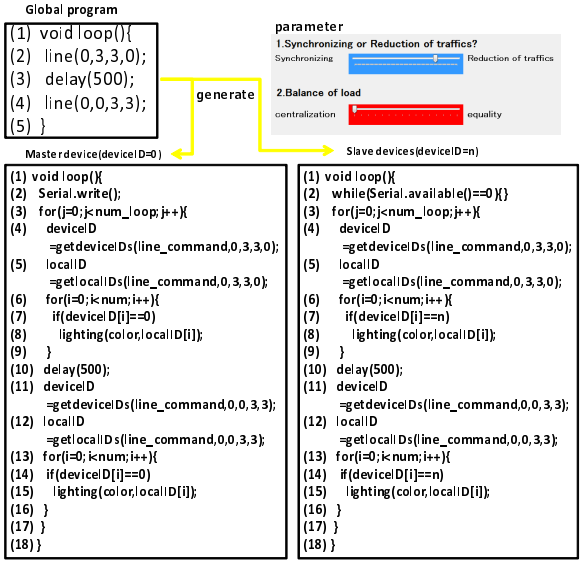

図 10 プログラム生成（中間型制御方式 2）

4. 評価実験

電飾アートの用途に応じて，適した制御方式は異なる．そこでグローバルプログラム例をいくつか挙げ，各制御方式での違いを比較し，考察する．中間型制御方式においては，3.4 節のようにさらに 1. 同時性をより重視したタイプ，2. 通信量の削減をより重視したタイプ，3. 同時性と通信量の削減どちらも均等に重視したタイプに分ける．評価する項目は 3 章で挙げた LED 点灯の同時性，通信量の削減，消費電力のバランスである．評価方法は，同時性に関しては各制御方式で実際に電飾アートを光らせ，デジタルカメラで動画を撮影する．撮影した動画を Windows ムービーメーカー（1 フレーム 0.03 秒）でフレーム毎に確認し，明

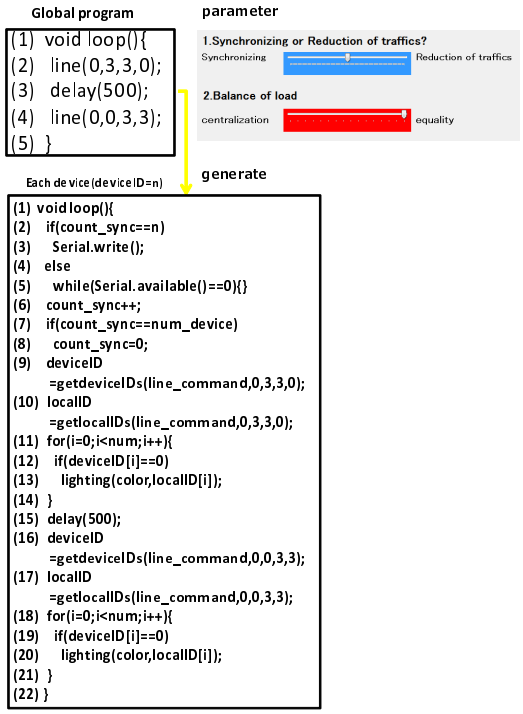

図 11 プログラム生成（中間型制御方式 3）

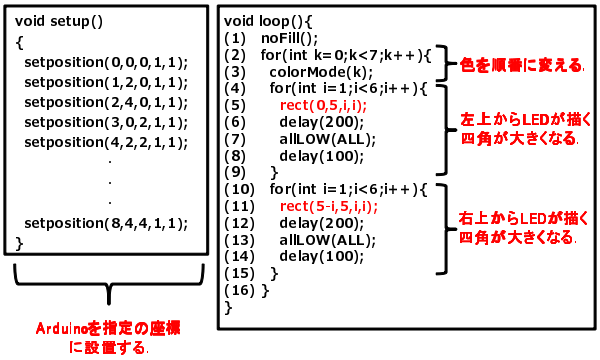

図 12 グローバルプログラム例

滅のタイミングがずれた時のずれフレーム数の総量を実測する．また，最も大きいずれフレーム数も実測する．通信量の削減に関してはグローバルプログラムから手計算でデバイス群での総通信量（バイト）を実測する．消費電力のバランスに関しては消費電力に最も影響がある受信量（バイト）を各デバイスごとに実測し，標準偏差を算出する．この時，消費電力のバランスは同時性や通信量の削減のパラメータに依存しないため，例として中間型制御方式タイプ□で消費電力のバランスのパラメータを 5 段階に分け，比較する．

4.1 グローバルプログラム例

図 12 に長方形を描く 6 × 6 の電飾アートのグローバルプログラム例を示す．プログラムでは，まずコンフィグレーション部分で LED を 6 × 6 の格子状に配置する．(2) で図形の塗りつぶしを無効にし，(4) で色を設定，(5)-(10) の for 文で長方形を左上から大きくしていき（図 13 上図），(10)-(15) の for 文で長方形を右上から大きくしていく（図 13 下図）．

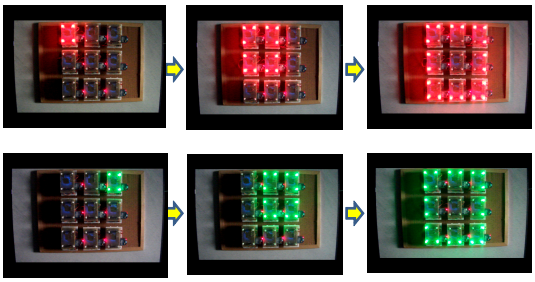

図 13 電飾アート例

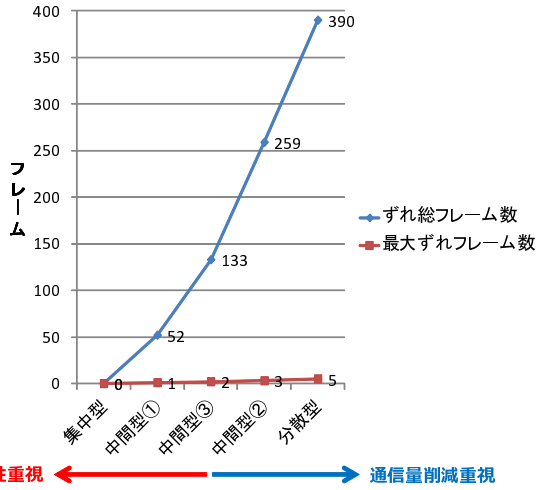

図 14 総ずれフレーム数と最大ずれフレーム数

4.2 結果と考察

- LED 明滅の同時性
  図 14 は各制御方式でグローバルプログラムを 3 ループ動作させた時の総ずれフレーム数と最大ずれフレーム数を実測した結果である．ずれフレーム数が少ないほど同時性が高いと言える．同時性を重視した制御方式つまり同期の回数が多い制御方式ほどずれフレーム数が少ないことが確認できる．
- 通信量の削減
  図 15 は各制御方式でグローバルプログラムを 3 ループ動作させた時のデバイス群での総通信量を表す．集中型制御方式はスレーブデバイスの出力制御のたびに 2byte のメッセージを送るため通信量が非常に多い．一方で，分散型制御方式は出力のみの場合はデバイス間の通信が必要ないため通信量は 0byte である．中間型制御方式は通信量が非常に少ない．
- 消費電力量のバランス
  デバイスの消費電力量は受信量に大きく依存する．図 16 は中間型制御方式タイプ 3 で設定を 5 段階に分けて，それぞれ 9 ループ動作させた時の受信量の標準偏

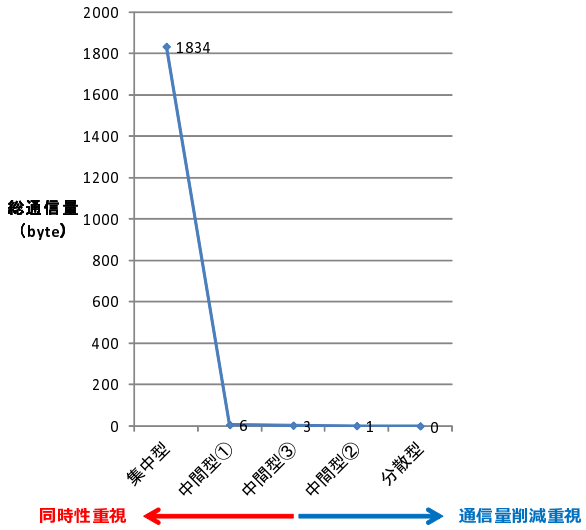

図 15 総通信量

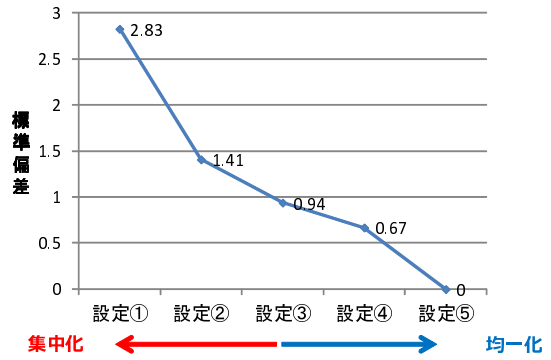

図 16 受信量の標準偏差

差を表す．この値が小さいほど消費電力量のバランスがとれている．均一化を重視した設定ほど標準偏差の値が小さいことが確認できる．

5. おわりに

本稿では，筆者らがこれまでに設計したマクロプログラミングモデルに新たに分散型・中間型制御方式を設計・実装し，パラメタにより適切な制御方式を自動で選択・実行を行うシステムを実装した．これにより，電飾アートの要件に柔軟に対応することが可能となった． また，実際に各制御方式で 1 つのグローバルプログラムを実行し，電飾アートの要件を満たすかどうか定量的に評価することで新たに実装したシステムが有効であることを確認した．今後の課題としては，技術者や非技術者など多くの人に利用してもらうことや実運用を通して，提案モデルの有用性などの検証を行い，さらなる機能の強化を図る予定である．

謝辞

本研究の一部は，文部科学省科学研究費補助金基盤研究 (A)(20240009, 23240010) によるものである．ここに記して謝意を表す．

参考文献

[1] 森川博之: ユビキタスネットワーク社会の構築, 電気学会論文誌C, Vol. 124, No. 1, pp. 12–17, 2004.
[2] R. Newton, Arvind, and M. Welsh: Building up to Macroprogramming: An Intermediate Language for Sensor Networks, *Proc. of 4th International Symposium on Information Processing in Sensor Networks (IPSN2005)*, pp. 37–44, 2005.
[3] R. Gummadi, O. Gnawali, and R. Govindan: Macro-Programming Wireless Sensor Networks Using Kairos, *Proc. of 1st Distributed Computing in Sensor Systems (DCSS 2005)*, pp. 126–140, 2005.
[4] R. Newton, G. Morrisett, and M. Welsh: The Regiment Macroprogramming System, *Proc. of 6th International Conference on Information Processing in Sensor Networks (ISPN2007)*, pp. 489–498, 2007.
[5] B. Urs and K. Gerd: A State-Based Programming Model and System for Wireless Sensor Networks, *Proc. of 3rd International Workshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS 2007)*, pp. 261–266, 2007.
[6] S. R. Madden, M. J. Franklin, J. M. Hellerstein, W. Hong: TinyDB: An Acquisitional Query Processing System for Sensor Networks, *ACM Transactions on Database Systems*, pp. 122–173, 2005.
[7] 木下浩平, 藤田直生, 柳沢 豊, 寺田 努, 塚本昌彦: 分散制御された LED マトリックスを用いた電飾アート制御プラットフォーム, 情報処理学会研究報告, Vol. 2009-EC-12, No. 26, pp. 65–70, 2009.
[8] 中田眞深, 児玉賢治, 藤田直生, 竹川佳成, 寺田 努, 塚本昌彦,: プロジェクタによる一斉制御が可能なユビキタス光デバイスの設計と実装, 情報処理学会論文誌, Vol. 50, No. 12, pp. 2871–2880, 2009.
[9] 長岡佑典, 佐野渉二, 寺田 努, 塚本昌彦 : 複数ユビキタス機器を統合的に扱うためのマクロプログラミングモデルの設計, マルチメディア, 分散, 協調とモバイル (DICOMO2011) シンポジウム, pp. 659-666.
[10] Processing, http://www.processing.org/.
[11] Arduino, http://www.arduino.cc/.