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

長岡 佑典[†1]　佐野 渉二[†2]
寺田 努[†1,†3]　塚本 昌彦[†1]

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 を多くのマイコンなどを用いて制御する必要があるが，1 つ 1 つのマイコンのプログラムを書くような方法では，多大な労力を要するなどの問題がある．本稿では，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設計し，これを電飾アートに用いることにより，各 LED の位置を指定しながら光らせ方を制御したり，観客の動作により電飾アートの光らせ方を変える機能などを容易に記述できる環境を構築する．実際にいくつかの作品を作成することで，提案モデルを用いて電飾アート全体の光り方をマクロな視点で制御できることを確認した．

Design of a Macroprogramming Model for Controlling Multiple Ubiquitous Devices

Yusuke NAGAOKA,[†1] Shoji SANO,[†2]
Tsutomu TERADA[†1,†3]
and Masahiko TSUKAMOTO[†1]

Recently, illuminations and information displays using LEDs have attracted a great deal of attention. Although it is necessary to control a large number of LEDs using many microcomputers to realize large-scale illuminations, it requires a huge effort to write programs for each microcomputer. In this paper, we propose a macroprogramming model for designing illuminations to control blinking patterns flexibly. We confirmed that we could control blinking patterns of a illumination with a macroscopic view.

1. はじめに

近年，計算機の小型化・高性能化に伴い，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が注目されている．ユビキタスコンピューティングとは，Mark Weiser が提唱した概念であり，オフィスや家庭など生活環境中のさまざまなものや場所に埋め込まれた小型のコンピュータを，人が意識することなく利用できるコンピュータの利用形態である[1)]．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．そのため，複数のユビキタスデバイスを個別に制御するのではなく，それらを一斉に制御する必要がある．多数のデバイスを統合的に制御する手法としてはマクロプログラミング[2)–6)] がある．マクロプログラミングは，個々のユビキタスデバイスのネットワークトポロジなどを考慮しなくても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るため，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，既存の電飾アートは，光が変化しない，または光り方が固定されているものが多い．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ためであると考えられる．

そこで本研究では，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する．提案するモデルでは，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．提案モデルにより，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．本論文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

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

計について説明する．第 4 章でプログラミング例を示し，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．本研究もこのマクロプログラミングの考え方を採用しているが，従来のマクロプログラミングの多くはセンサなどのデータをいかに効率よく利用するかという点に着目してシステム設計されているのに対し，本研究では入出力機器の配置関係を考慮しながらいかに簡単に電飾制御を行うかに着目している点で異なる．

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．

3. マクロプログラミングモデルの設計

本研究では，大規模なイルミネーションなどの LED を用いた電飾アートを制御することを目的とする．電飾アートでは，全体の光り方をマクロな視点で制御することが求められる．電飾アートが大規模になるほど，LED とそれの制御を行うマイコンが大量に必要となるが，1 つ 1 つのユビキタスデバイスに直接プログラミングを行うような手法では，それぞれのユビキタスデバイスのプログラムを変えなければならず，多大な労力と時間を必要とし，時間に応じて光らせ方を変えるようなことは困難である．そこで本研究では，電飾アートにおいて複数のユビキタスデバイスを利用する際に，光り方をマクロな視点でプログラミングできるようにマクロプログラミングを用いる．

3.1 ユビキタス機器を扱うマクロプログラミング

マクロプログラミングとは，図 1 のように複数のユビキタスデバイスに対して，個々の通信ネットワークのつながりを考慮せずに，それらの制御方法についてのみ着目して記述できる 1 つのプログラムを用いて複数のユビキタスデバイスを制御することである．しかし，従来のマクロプログラミングの多くは，ユビキタスデバイスで取得したセンサなどのデータをいかに効率よく使うかに着目しており，電飾アートに用いるための入出力機器を有効に使うことはあまり考えられていない．そこで本研究では，電飾アートに用いるための複数の入出力機器を扱うマクロプログラミングモデルを提案する．

使用者がプログラミングにあまり慣れていない電飾デザイナであることも考え，プログラム記述には，簡単な言語で映像表現がプログラミングできる Processing 言語を用いる．本研究では，電飾アートとして図 2 のようにフルカラー 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ものを想定する．以下，本研究で考えるマクロプログラミングの特徴を挙げる．

- 複雑な光らせ方，動きのある光らせ方が可能
  基本的な線，図形，模様や色の変化などを組み合わせ，複雑な光り方が実現できる．また，センサなどの入力に合わせて動的に光り方を変化させるといった機能も記述できる．
- 位置に基づく光らせ方が可能
  LED を制御するユビキタスデバイスには ID を割り当てるが，デバイス ID を指定して LED を光らせるのは使用者が ID を把握しておく必要がある．このため，実世界の座標を用いて光らせたい点や図形の位置を指定できるようにした．
- プログラム記述が容易

図 1　マクロプログラミング

デザイナなどプログラミングに慣れていない人にとってはプログラミングは簡単ではない．そこで，初心者でも比較的容易にグラフィックのプログラミングを行えることで近年注目されている Processing を基盤として用いた．

- 個々のデバイスの負担軽減
  本研究では，1 つのユビキタスデバイスで 1 つまたは少数の LED を制御するとした．そのため，仮にユビキタスデバイスが故障などで動かなくなってもその影響を最小限に抑えられる．
- 電飾アートの大きさや形状に対する柔軟性
  電飾アートの大きさや形状は使用目的や設置場所などにより異なるため，どのような大きさや形状であっても利用できるようにした．

3.2　電飾アートのためのプログラミング言語設計

本研究では，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[9] 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．実際に提案手法を用いて電飾アートを設計する際には，まず電飾アートの大きさや形状を決定し，LED を配置した後に各ユビキタスデバイスの ID に位置座標を割り当てるなどの設定を行う必要がある．このような設定を行う人を本論文では設計者と呼び，設計者が初期設定を行った後に，電飾アートの動作記述を行う人を使用者と呼ぶ．本研究では，設計者はある程度以上のマイコンやプログラミングに関する知識があることを想定する．以下，拡張した関数と新たに作成した関数について説明する．関数内のパラメータについては表 1 でまとめた．

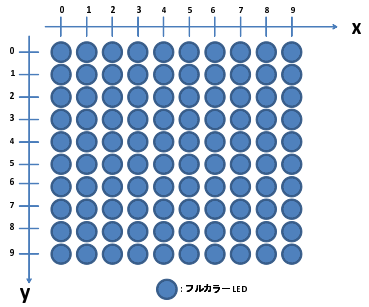

図 2　RGB フルカラー LED と一体となったユビキタスデバイスを格子状に並べた電飾アート

表 1　関数内のパラメータ

ID	デバイスの ID 番号
mode	IO ピンの INPUT/OUTPUT の設定
pin	ピン番号
value	HIGH または LOW，アナログピンの場合は 0〜255 の値
x	x 座標
y	y 座標
width	幅
height	高さ
r	半径
color	LED の色 (red, green, blue)

3.2.1　ユビキタスデバイスの設定を行うための関数

電飾アートを設置した後に設計者はユビキタスデバイスの入出力設定や配置を設定する必要があり，そのための関数を作成した．

- **pinMode.***ID*[**mode**]: 指定 ID のデバイスの IO ピンの入出力を設定する．
- **setposition**(*ID, x, y*): 指定 ID のデバイスの位置座標 (x, y) を設定する．

3.2.2　ユビキタスデバイスを制御するための関数

設計者がユビキタスデバイスの動作確認を行うための関数について説明する．これらの関数は，デバイスの ID を指定できるように Processing の関数を拡張したものである．

- **digitalWrite(***ID, pin, value***)**: 指定 ID のデバイスの pin に HIGH または LOW を出力する．
- **analogWrite(***ID, pin, value***)**: 指定 ID のデバイスの pin にアナログ値を出力する．
- **digitalRead(***ID, pin***)**: 指定 ID のデバイスの pin を HIGH か LOW の 2 値として読み取る．
- **analogRead(***ID, pin***)**: 指定 ID のデバイスの pin をアナログデータとして読み取る．

3.2.3 位置に基づく光の制御を行うための関数

格子状の電飾アートの光の制御を行うための関数について説明する．使用者はユビキタスデバイスの位置情報をもとに光の制御を行う．以下は Processing の図形描画関数を拡張したものであるが，制御構文など Processing の一般の機能は提案システムでも自由に利用できる..

- **colorMode(color)**: 点灯する LED の色を color で設定する．他の描画関数において色の指定が行わなかった場合にはここで指定した色が用いられる．
- **point(***x, y***, color)**: 点 (x, y) に位置するデバイスの LED を color の色で点灯させる．
- **line(***x1, y1, x2, y2***, color)**: 点 (x1, y1) と点 (x2, y2) を結ぶ線上に位置するデバイスの LED を color の色で点灯させる．
- **triangle(***x1, y1, x2, y2, x3, y3***, color)**: (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 color の色で点灯させる．
- **rect(***x, y, width, height***, color)**: 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 color の色で点灯させる．
- **fill(color)**: 図形の塗りつぶし部分上にある LED を color の色で点灯させる．
- **noFill()**: 図形の塗りつぶし部分上にある LED を消灯させる．
- **stroke(color)**: 線や図形の枠線上にある LED を color の色で点灯させる．
- **noStroke()**: 線や図形の枠線上にある LED を消灯させる．

以下は新たに作成した関数について説明する．

- **circle(***x, y, r***, color)**: 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 color の色で点灯させる．
- **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．

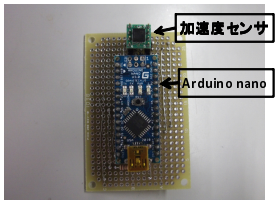

図 3 Arduino nano 面

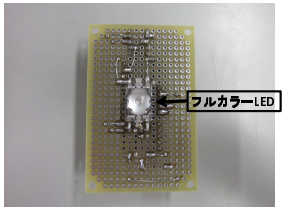

図 4 フルカラー LED 面

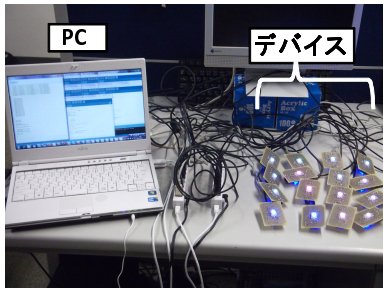

図 5 電飾アートの制御風景

- **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．

3.3 実　　装

3.3.1 電飾アートのためのユビキタスデバイス

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 LED と加速度センサを Arduino nano[10] 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と単純な入出力などを搭載した基板である．Arduino nano は 14 本の入出力ピンおよび USB ポートをもち，プログラム上で役割を決めて自由に使用できる．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間の通信は図 5 のように PC を経由して行う．

3.4 実 行 環 境

マクロプログラミングでは，記述した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，記述したプログラムから個々のユビキタスデバイス用のプログラムを生成する必要がある．そこで，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

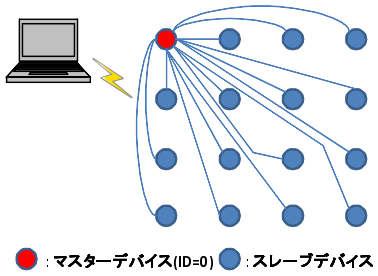

図 6　プログラムの処理方法

ネレータを C ＃を用いて作成した．以下，記述したプログラムによる LED の制御方法と各ユビキタスデバイスで制御するためのプログラムを生成する方法について説明する．

本研究では，電飾アートを構成するユビキタスデバイスのうちの 1 つをマスターデバイスと呼び，マスターデバイス以外のデバイスをスレーブデバイスと呼ぶ．LED を制御する際には，マスターデバイスが点灯を制御する信号を適宜スレーブデバイスに送信する．つまり，図 6 のようにマスターデバイスがすべてのスレーブデバイスに命令を送ることによってユビキタスデバイス全体の動きを制御する．マスターとスレーブ間の通信フォーマットを図 7 に示す．1 つの通信パケットは 2 バイトで構成されており，最初の 1 ビットで 1 バイト目か 2 バイト目かを判断し，1 バイト目のデータなら続けて実行する関数の ID，4 ビットの実行先ユビキタスデバイスの ID が続く．2 バイト目のデータならマスターからスレーブへの命令の場合はユビキタスデバイスのポートの ID，スレーブからマスターへセンサデータを送る場合は 7 ビットのセンサデータが続く．現在のフォーマットでは ID を 4 ビットで表現しているため，17 個以上で通信を行う場合は通信フォーマットを拡張する必要がある．

次に，記述したプログラムから個々のユビキタスデバイスが制御するためのプログラムの生成方法について説明する．各ユビキタスデバイス用のプログラムに生成するのは記述したプログラムの関数部分であり，関数部分以外のプログラムは，マスターデバイスのみに書き込まれる．3.2 で説明した関数の中からいくつかを例として各ユビキタスデバイス用に生成したプログラムを表 2 にまとめた．図形を描く関数の場合は，マスターデバイスは関数 getID で図形上にあるユビキタスデバイスの ID を取得し，取得したユビキタスデバイスに光らせる命令 (senddata) を送るという処理になっている．また，スレーブデバイスの入力制御の関数の場合は，マスターデバイスからの命令 (requestData) を受けると，入力データ (senddata) を送り返すという処理になっている．

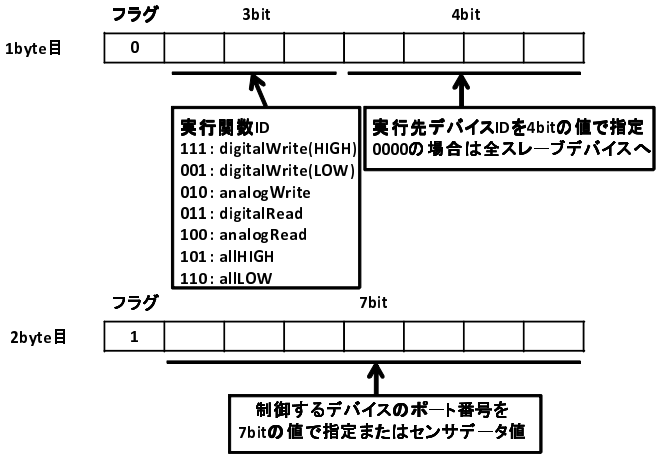

図 7　通信フォーマット

各ユビキタスデバイスへのプログラム生成の例を図 8 に示す．この例では，スレーブ 1 のセンサの値を読み取り，その値によって，スレーブ 2 の LED を光ら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 (3，4，5，8，10，13 行目) を読み取り，マスターデバイスと各スレーブデバイス用の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は，マスターデバイスのみに書き込まれる．スレーブ 1 の 1，2，4〜10 行目と 16〜19 行目部分，スレーブ 2 の 1，2，4〜10 行目と 14〜17 行目部分はあらかじめ書かれるプログラムである．つまり，マスターデバイスに書き込まれるプログラムは，記述したプログラムに沿ったプログラムであり，関数部分のみがマスターデバイス用に生成されたプログラムとなる．一方で，スレーブデバイスに書き込まれるプログラムは，関数部分をスレーブデバイス用に生成したプログラムのみで構成される．

4. マクロプログラミング例

設計したマクロプログラミングモデルを用いる例として，図 9 のように街中で格子状に並べられたクリスマスツリーのような LED イルミネーションを制御する場面を想定する．このようなイルミネーションでは図形を描くように光らせたり，通行人がツリーに触ると光り方が変化するといったインタラクションが有効である．以下の例では，1 つのユビキタスデバイスで制御するツリーを 1 つのフルカラー LED，インタラクションのためのセンサを

表 2　関数を生成したプログラム 1

関数	マスター（上段） スレーブ（下段）
pinMode.ID(mode)	mode　(ID = 0 のとき) mode　(ID != 0 のとき)

setposition(*ID, x, y*)	int getID(int X, int Y){
	if(X == x && Y == y)
	return ID;
	}

digitalWrite(*ID, pin, value*)
or
analogWrite(*ID, pin, value*)
	if(ID == 0)
	digitalWrite or analogWrite(pin, value);
	else
	Serial.write(senddata(command, ID, pin, value), BYTE);
	case Command(digitalWrite) or Command(analogWrite):
	digitalWrite or analogWrite(bitPin, value or bitValue);
	break;

line(*x1, y1, x2, y2*, color)	ID[] = getID(line(x1, y1, x2, y2));
	for(int n=0; n 〈 num; n++){
	if(ID[n] == 0)
	digitalWrite(pin, value);
	else
	Serial.write(senddata(command, ID, pin, value), BYTE);
	}
	case Command(digitalWrite):
	digitalWrite(bitPin, value);
	break;

Read(*x, y*)	ID = getID(x, y);
	if(ID == 0)
	value = analogRead(pin);
	else
	value = requestData(ID, pin);
	case Command(analogRead):
	value = analogRead(bitPin);
	Serial.write(senddata(value));
	break;

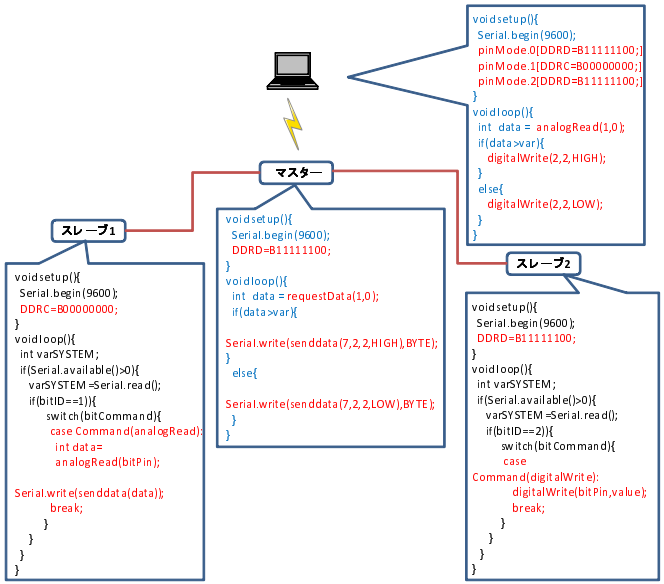

図 8　プログラムの生成の流れ

図 9　街のイルミネーション

加速度センサとみなし，図 3，4 のデバイスに置き換えて考える．

4.1　格子状の電飾アート用プログラム例

図 10 に長方形を描く 4 × 4 の電飾アートの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デバイスを 4 × 4 の格子状に配置する．(2) で図形の塗りつぶしを無効にし，(3) で色を緑色に設定，(4)-(12) の for 文で幅 1, 高さ 1 の長方形を左上 (図 11 の左上図)，右上，左下，右下 (図 11 の右上図) の順番に光らせる．(13) で色を赤色に設定，(14)-(19) の for 文で幅 3, 高さ 1 の長方形を上段，中段，下段の順番に光らせる．(20) で色を青色に設定，(21)-(26) で 1 × 1 の正方形 (図 11 の左下図) と 3 × 3 の正方形 (図 11

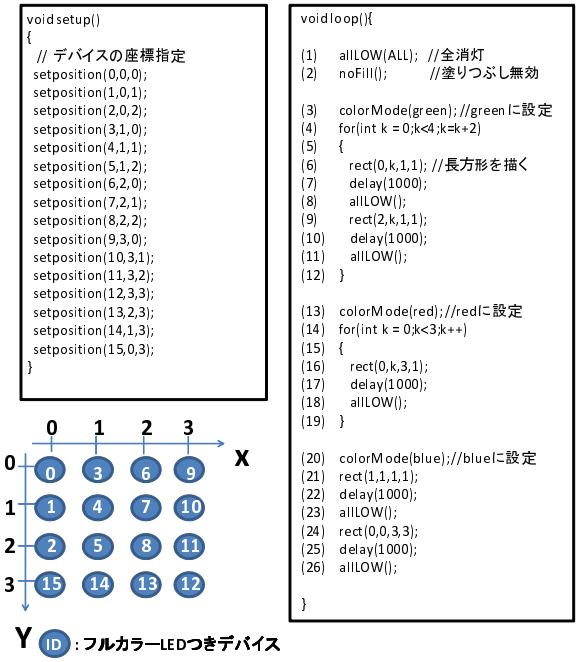

図 10 電飾アートのプログラム例

の右下図) を順番に光らせる．

4.2 センサを用いたプログラム例

図 12 にセンサを用いた制御のプログラム例を示す．この例では，すべてのデバイスの加速度センサの値を読み取り，閾値を越えているデバイスは赤色 LED を光らせる．また，閾値を越えたデバイスの個数を PC に送信する．プログラムでは，まずコンフィグレーション部分でデバイスを 3 × 3 の格子状に配置する．(3)(4) でデバイスの列数と行数を設定し，(5) でデバイスを数えるカウンタを設定する．(6)-(23) で格子状に配置されたデバイスを順番に (10) でセンサの値を読み取る．(11)-(16) で値が閾値を越えていれば (13)(14) で赤色で光らせ，(15) でカウントする．(17)-(21) では閾値を越えていないデバイスは (19)(20) で緑色で光らせる．最後に，(24) で PC に閾値を越えたデバイスの数を送信する．

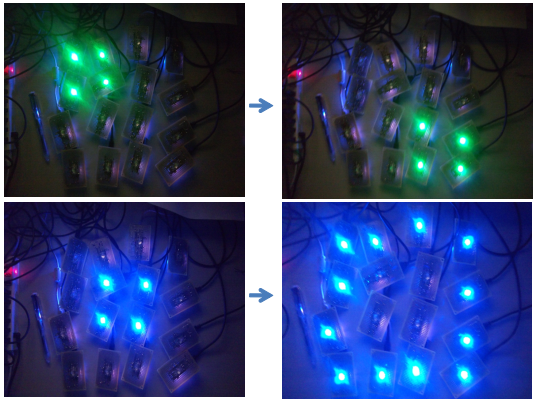

図 11 電飾アート例

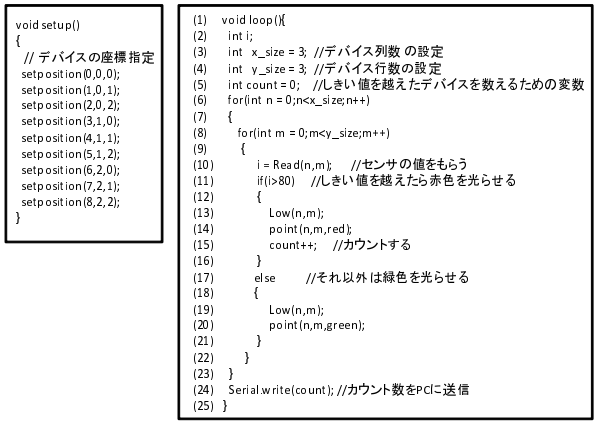

図 12 センサを用いた制御のプログラム例

5. おわりに

本研究では，電飾アートに用いるために複数の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を行った．設計したマクロプログラミングモデルは，制御対

象となるユビキタスデバイスの位置関係，センサや LED などの入出力機器に基づいた制御をマクロな視点で，より柔軟にプログラミングできる．電飾アートとして格子状の電飾アートのための関数を作成または既存の Processing 言語の関数を拡張した．これらの関数を用いて記述したプログラムに対して，各ユビキタスデバイス用のプログラムに生成するジェネレータやユビキタスデバイス全体の処理のための通信フォーマットなどを作成し，実行環境を作成した．

今後の課題としては，さらに関数の拡張や作成を行い，楕円や台形，星形などの複雑な図形の表現を可能にすることや光の色の種類の増加により，より表現力の高い電飾アートを実現することが挙げられる．本研究では評価実験を行っていないため，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，プログラム記述の容易性や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) Processing, http://www.processing.org/.
10) Arduino, http://www.arduino.cc/.