第 14 回　みやこ祭

プロジェクションマッピング

引継ぎ資料

資料作成日

2019 年 2 月 7 日

記述内容

配線・ソフトウェア・床面の説明

資料作成者　18272001　関 優志

（システムデザイン学部　知能機械システムコース）

目次

1. プロジェクションマッピング（A,B,C 面）

1.1. システム構成

2018 年度のシステム構成を以下に示す.

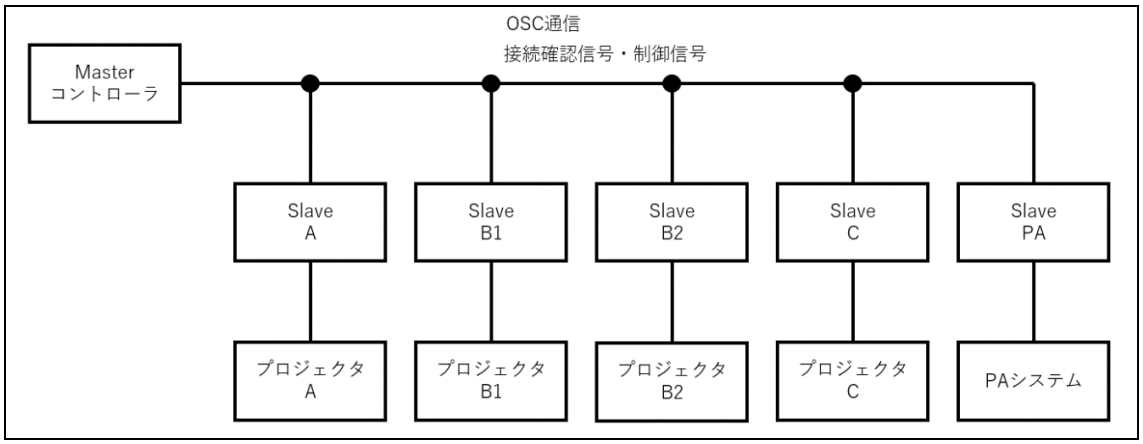

図 1.1　システム構成

映像・音響全体を制御する Master（コントローラ）1 台と，Master からの指令に応じて各プロジェクタに映像を出力する Slave5 台から構成される．Master と Slave となる PC には Mac を使用した．Master と Slave 間は LAN ケーブルで接続し，OSC 通信を用いた各システムの同期を図っている.

Master/Slave のソフトウェア開発には openFrameworks を使用した.

1.2 機器の接続

使用する機器は図 1.2 のように接続する．

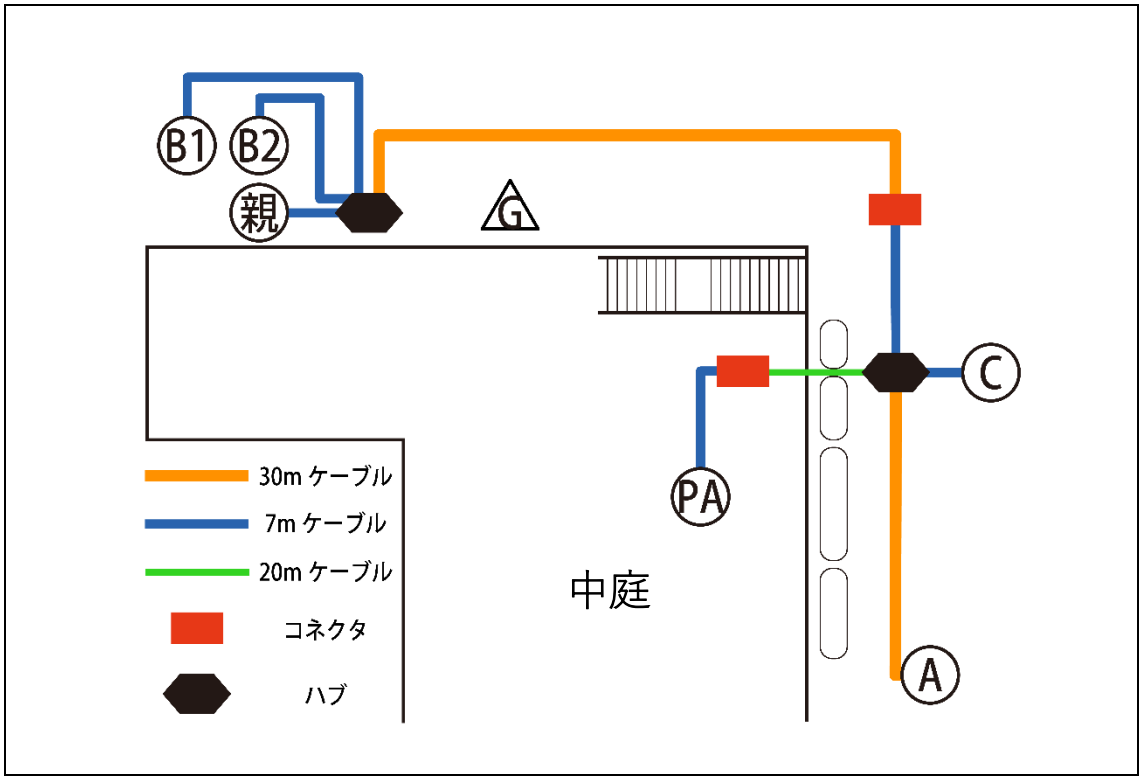

図 1.2　機器の接続

丸印：パソコン（Mac），親：コントローラ，G：床面投影機材

1.3. 操作手順

1.3.1.　接続と確認

接続確認機能により，接続ミスがなく，Master が各 Slave との通信に成功していることを確認できる．これにより，投影開始時になって接続不良が発覚することを防ぐ．

① Master と各スレーブを LAN ケーブルで接続する．
② Master と各スレーブでプログラムを実行する．
③ コントローラ画面の「Connection Check」には「failed」と表示されている．
④ コントローラのキー「c」を押す．
⑤ 「connecting…」と表示され，接続試行中となる．
⑥ 通信が成功すれば，「success」と表示される．

「failed」と表示される場合は，正常に通信が出来ていないことになるので，接続を再確認する．

【チェック】
・PC の WiFi 接続は切られているか？
・ハブの電源は入っているか？
・使用する変換ケーブルは適切か？（Mac の種類により相性の良し悪しがあった）

1.3.2. 投影

コントローラの操作により，全ての Slave を操作することができる．以下に，操作の種類を示す．

表 1.1 コントローラで可能な動作

動作	キー	説明
再生（本編動画）	スペース	本編動画と本編用音声を再生する．
一時停止	p	本編動画，本編用音声，QR 用動画，QR 用音声を一時停止する．
巻き戻し	R	本編動画，本編用音声，QR 用動画，QR 用音声を巻き戻す．
映像の表示・非表示	Q	全プロジェクタ映像の表示・非表示を切り替える．
再生（QR コード）	G	QR 用動画と QR 用音声を再生する．

再生中に「Q」を押した場合，画面への出力を遮断しているだけであり，裏では再生が続いているため，注意する．また，各 Slave はキー操作により，個別に映像の表示・非表示を切り変えることができる．

表 1.2 各 Slave で可能な操作

動作	キー	説明
映像の表示・非表示	Q	全プロジェクタ映像の表示・非表示を切り替える．

なお，操作の優先度が高いのは，コントローラである．
（例）Slave 側で映像を非表示にしても，コントローラ側の操作次第で表示状態にできる．

【チェック】
・映像が最後まで再生されても自動的に巻き戻しはされない．続けて再生する場合は「R」を押して巻き戻す．

1.3.3.　詳細説明

・OSC メッセージの構成

OSC 通信では，URL のように送受信する情報をラベリングできる．今回のシステムで使用した OSC メッセージを図 1.3 に示す．

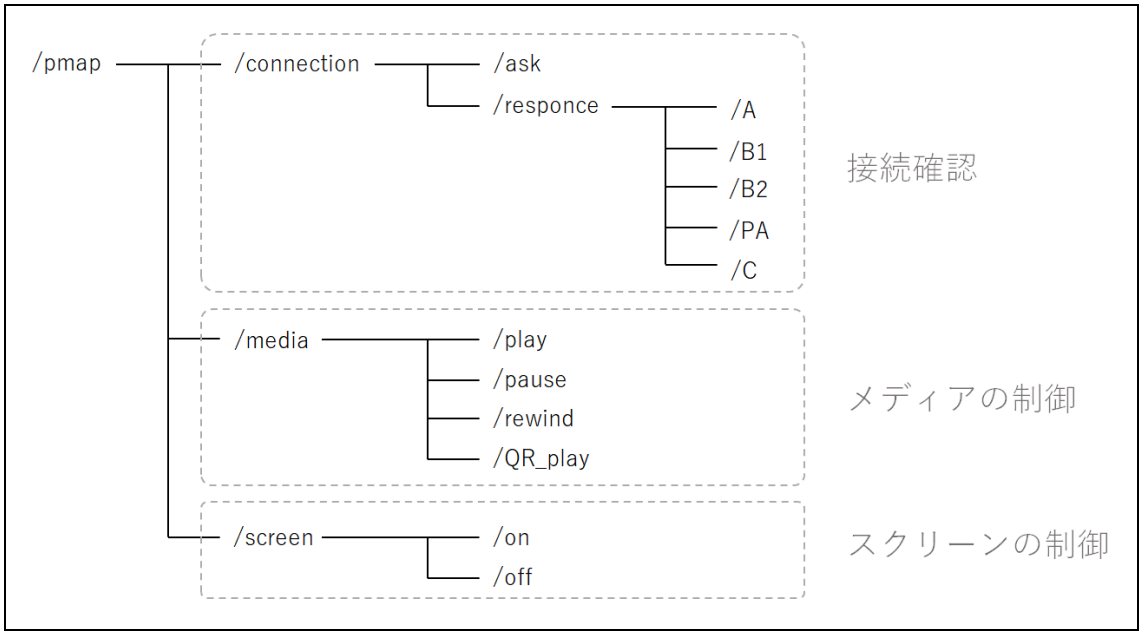

図 1.3　OSC メッセージ

OSC メッセージの階層構造は大きく 3 つに分かれている．基本的に Master から slave 方向に送信されるが，接続確認の「/responce」以下のみ slave から Master 方向に送信される．

・接続確認のしくみ

図 1.4 に接続確認の流れを示す．

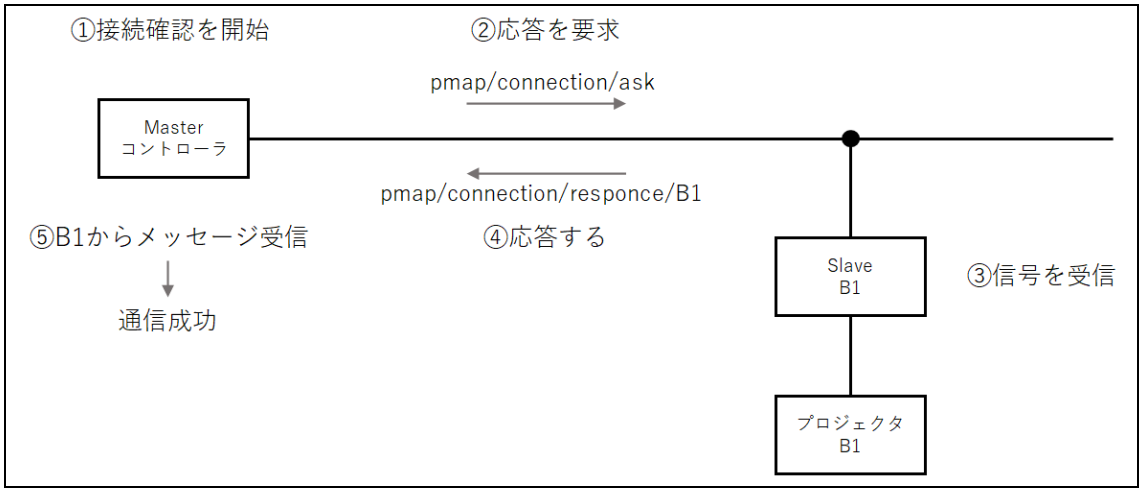

図 1.4　接続確認のしくみ

Slave B1 との接続確認を例にとる．Master で接続確認を開始すると，通信バスに OSC メッセージ「pmap/connection/ask」が流される．このメッセージはすべてのスレーブが受信することができる．Slave は，このメッセージを受信すると，末尾に各 Slave 名の付いた OSC メッセージを送り返すようにプログラムされている．Slave B1 であれば「pmap/connection/response/B1」を送信する．「pmap/connection/response/B1」を Maste が受信すれば，Master と Slave B1 間でメッセージの受け渡しが成功したことになり，通信路の確立が確認されるから，接続成功としている．

・問題点と修正

今回のシステムでは，有線通信を使用しているため，Master からほぼ同タイミングで制御信号を出力しても，各スレーブが受信するタイミングに差が生じる．これにより，映像のタイミングがずれてしまう可能性がある．これを解決するために，slave ごとに信号を受信してから映像を再生するまでの時間を設定できる機能を追加していたが，思っているように動作しいていなかった．仕様では，slave のプログラム「ofApp.h」内の「PLAY_ADJUST」に，遅らせたいフレーム数を設定すると，そのフレーム数が経過した時点で映像の再生が開始する．

【問題点】

問題は，再生の OSC メッセージを受信したときに実行されるメソッド「void ofApp::play()」の中に，「movie.play()」があることである．「movie.play()」を実行するとその時点で映像がはじまる．実際には「PLAY_ADJUST」で制御していたのは画面を非表示から表示に変えるタイミングであり，動画自体の再生開始タイミングの制御はできていなかった．

また，フレーム数をカウントする変数がプログラムを実行中，常にインクリメントされており，オーバーフローによる誤動作を引き起こす可能性がある．（2018 年度の投影時間ではオーバーフローしなかったが，投影時間が長くなるとオーバーフローする可能性がある．）

【変更点】

問題解決のため，以下の 3 カ所を変更する．

① slave のプログラム「ofApp.cpp」内の void ofApp::play()を以下のように変更する．

```
void ofApp::play(){
	movie.setPaused(false);
	qr.setPaused(true);
	videoType = 1;
}
```

② 同ソースファイル内「void ofApp::update()」内の

```
else if (m.getAddress() == "/pmap/media/pause"){
	pause();
}
```

を

```
else if (m.getAddress() == "/pmap/media/pause"){
	pause();
	framecount++;
}
```

に変更する.

③ 同ソースファイル内「void ofApp::update()」内の

```
if(count_start){
	framecount ++;
	if(framecount == PLAY_ADJUST){
		movie_on = true;
	}
}
```

を

```
if(count_start){
	if(framecount == PLAY_ADJUST){
		movie_on = true;
		movie.play();
	}
	else if(framecount < PLAY_ADJUST) {
		framecount++;
	}
}
```

に変更する.

1.4. ソフトウェア解説

1.4.1.　マスター

```
verdana.load("VerdanaRef.ttf", 50)
```

フォントの設定をしている．フリーのフォントデータ VerdanaRef.ttf をダウンロードし，プロジェクトファイルの「bin>data」に入れる．

```
projector_A1.setup(IP_PROJECTOR_A1, PORT_TO_SLAVE);
projector_B1.setup(IP_PROJECTOR_B1, PORT_TO_SLAVE);
projector_B2.setup(IP_PROJECTOR_B2, PORT_TO_SLAVE);
projector_C1.setup(IP_PROJECTOR_C1, PORT_TO_SLAVE);
pa.setup(IP_PA, PORT_TO_SLAVE);

receiver.setup(PORT_TO_CONTROLLER);
```

OSC 通信のポート設定．コントローラ→スレーブのポートは 8000，スレーブ→コントローラのポートは 7000 に設定した．子機の Mac の IP アドレスを調べ，ここで設定する．記述は ofApp.h 内で．

```
cnState_A1 = 2;
cnState_B1 = 2;
cnState_B2 = 2;
cnState_C1 = 2;
cnState_PA = 2;
```

コントローラと各スレーブとの接続状態を示す変数．
0：...（接続試行中），1：接続成功，2：接続失敗

```
getTime()
```

現在時刻を表示する．コントローラの画面上に現在時刻を表示している．

```
if (connectionState == true) {
        frameCount++;
}
```

変数「connectionState」が true の時は，接続チェックを行っている状態である．接続確認中は変数「frameCount」をカウントアップしている．

```
while (receiver.hasWaitingMessages()) {
   ...
}
```

コントローラが，各スレーブから送信された OSC メッセージを受信している間，while 文内の処理を繰り返す．

```
ofxOscMessage m;
receiver.getNextMessage(&m);
```

ofxOscMessage クラスのインスタンス m を宣言し，「getNextMessage(&m)」で m に受信データを代入する．

```
if (m.getAddress() == "/pmap/connection/responce/A") { cnState_A1 = 1; }
else if (m.getAddress() == "/pmap/connection/responce/B1") { cnState_B1 = 1; }
else if (m.getAddress() == "/pmap/connection/responce/B2") { cnState_B2 = 1; }
else if (m.getAddress() == "/pmap/connection/responce/C") { cnState_C1 = 1; }
else if (m.getAddress() == "/pmap/connection/responce/PA") { cnState_PA = 1; }
else;
```

コントローラが受信したデータにより条件分岐させている．メッセージを送信してきた子機の「cnState_xx」変数を 1 に変更し，接続成功の状態に遷移させている．

```
if (frameCount > TIMEOUT) {
            if (cnState_A1 == 0) cnState_A1 = 2;
            if (cnState_B1 == 0) cnState_B1 = 2;
            if (cnState_B2 == 0) cnState_B2 = 2;
            if (cnState_C1 == 0) cnState_C1 = 2;
            if (cnState_PA == 0) cnState_PA = 2;

            connectionState = false;
}
```

接続確認中にカウントアップした変数「frameCount」が「TIMEOUT」の数値より大きくなったら，タイムアウトと判断する．その場合，まだ接続確認中（cnState_xx==0）の機器は，「cnState_xx == 2」に変更し，接続失敗の表示に遷移させる．また，「connectionState」を false に変え，接続確認を終了する．

```
if (cnState_A1 == 1 && cnState_B1 == 1 && cnState_B2 == 1 && cnState_C1 == 1 &&
cnState_PA == 1) connectionState = false;
```

接続確認中かつタイムアウトする前にすべての機器の「cnState_xx」が 1（＝接続成功）になった場合，「connectionState」を false に変え，接続確認を終了する．

void ofApp::draw()内では，コントロール画面への文字や数値の表示処理を行っている．

```
verdana.load("VerdanaRef.ttf", 30); //①
ofSetColor(30, 200, 200); //②
verdana.drawString("表示したい文字", x 座標, y 座標); //③
```

今回は，フォントデータを使っているため表示処理は上記のようになる．①フォントデータをロードし，②文字の色を決め，③表示したい文字列とその座標を記述する．また，スレーブとの接続状態の表示は，変数「cnState_xx」の状態により内容を変えている．

その他，関数の説明を以下に示す．

```
getTime()
```

現在時刻を表示する関数．

```
keyPressed(int key)
```

キーボード入力を判別する関数（openFrameworks 標準機能）．

```
connectionCheck()
```

接続確認処理の初めに実行される．「cnState_xx=0」で，すべてのスレーブとの接続表示を「...」に変更した後，コントローラからスレーブに向けて OSC メッセージを送信する．ofxOscMessage クラスのインスタンス「message」に， setAddress()で OSC メッセージ「ofxOscMessage /pmap/connection/ask」をセットする．sendMessage()でメッセージを送信する．

コントローラから（本編動画，QW コードの）再生，停止，巻き戻し，スクリーンの表示・非表示の指令を出すときも，送信するメッセージは異なるが同じ仕組みである．

```
allPlay()
```

全ての子機に，本編動画再生の指示を出す．

```
allPause()
```

全ての子機に，動画停止の指示を出す．

```
allRewind()
```
全ての子機に，動画巻き戻しの指令を出す．

```
allScreenOff()
```
全ての子機に，スクリーン非表示の指令を出す．

```
allScreenOn()
```
全ての子機に，スクリーン表示の指令を出す．

```
allQR_Play()
```
全ての子機に，QR コード動画再生の指令を出す．

1.4.2. スレーブ

```
ofHideCursor();
CGDisplayHideCursor(NULL);
```
実行画面上でのカーソルを非表示にする．

```
ofSetFrameRate(30)
```
フレームレートの設定．1 秒間に 30 フレーム更新する．

```
movie.load("A_1102.mp4")
qr.load("A_QR.mp4")
```
投影したい動画のデータファイル名を設定する．ファイルは，プロジェクトファイルの中の「bin>data」に入れる．

```
movie.setLoopState(OF_LOOP_NONE)
qr.setLoopState(OF_LOOP_NONE)
```
動画のループをさせない．

```
sender.setup(IP_CONTROLLER, PORT_TO_CONTROLLER)
receiver.setup(PORT_TO_SLAVE)
```
OSC 通信のポート設定．コントローラ→スレーブのポートは 8000，スレーブ→コントローラのポートは 7000 に設定した．値の記述は ofApp.h 内で．

```
//movie_on = true;
//count_start = true;
//play();
//videoType=2;
```

このコメントアウトを外すと，強制的に動画を再生できる．デバック時などに使用.

```
movie.update();
qr.update();
```

動画データの更新.

```
while (receiver.hasWaitingMessages()) {
   …
}
```

スレーブが，コントローラから送信された OSC メッセージを受信している間，while 文内の処理を繰り返す.

```
ofxOscMessage m;
receiver.getNextMessage(&m);
```

ofxOscMessage クラスのインスタンス m を宣言し，「getNextMessage(&m)」で m に受信データを代入する.

```
if (m.getAddress() == "/pmap/media/play" ){…}
else if (m.getAddress() == "/pmap/media/pause"){…}
…
```

スレーブが受信したメッセージによって，条件分岐させている．ここで，スレーブの行う処理を決めている.

```
if(count_start){
        framecount ++;
        if(framecount == PLAY_ADJUST){
            movie_on = true;
        }
}
```

コントローラから再生の指令が来ると「count_start==true」になる．すると，変数「framecount」がカウントアップされ，「framecount==PLAY_ADJUST」となった時，変数「movie_on==true」となる．動画は「movie_on==true」となると表示される．（つま

り，子機ごとに再生の指令を受信してから動画を表示するまでの時間を微調整できる．
「1.3.3.　問題点と修正」で述べたように，このままでは表示開始のタイミングは調整できるが，再生開始のタイミングは調整できない．）

```
if(black == false && movie_on == true){
        if (videoType == 1) {
            movie.draw(0, 0, ofGetScreenWidth(), ofGetScreenHeight());
        }
        if (videoType == 2) {
            qr.draw(0, 0, ofGetScreenWidth(), ofGetScreenHeight());
        }
}
```

変数「black」は，コントローラからのスクリーンの表示・非表示指令，またはキーボード入力の「Q」により変更される．「black == false」かつ「movie_on == true」が満たされると，動画の再生がはじまる．変数「videoType」は動画の種類を表しており，videoType==1 の時は本編動画を，videoType==2 の時は QR コード動画を表示する．

その他，関数の説明を以下に示す．

```
keyPressed(int key)
```

キーボード入力を判別する関数（openFrameworks 標準機能）．

```
if(key == 'q'){
        black = !black;
}
```

キーボードの「Q」が押されたときに，変数「black」の状態を反転している．つまり，スクリーンの表示・非表示を切り替えている．

```
play()
```

本編動画再生の指示を出す．

```
QR_Play()
```

QR コード動画再生の指令を出す．

```
pause()
```

動画停止の指示を出す．

```
rewind()
```
動画を巻き戻す.

```
allScreenOff()
```
スクリーンを非表示にする.

```
allScreenOn()
```
スクリーンを表示する.

```
reply()
```
コントローラからの接続確認に応答する. スレーブからコントローラに向けて OSC メッセージを送信する. ofxOscMessage クラスのインスタンス「message」に, setAddress()で OSC メッセージ「/pmap/connection/responce/x」をセットする. sendMessage()でメッセージを送信する.

```
for (int i = 0; i < 10000; i++);
```
この処理は, 変数 i を一万回カウントするだけの処理であり, 時間稼ぎのために書いた. (delay させる処理が分からなかったため.) 時間稼ぎをする意味は, コントローラがすべてのスレーブに対しての送信処理を確実に終えた後に, スレーブから返信させたかったためである.

1.5. ソースコード (2018 年)

1.5.1. マスター

Master (コントローラ) 用のソースコード (main.cpp/ ofApp.h/ ofApp.cpp) を示す.

main.cpp

```
#include "ofMain.h"
#include "ofApp.h"

int main(){
    ofSetupOpenGL(700,800,OF_WINDOW); //実行画面サイズ

    ofRunApp(new ofApp());
}
```

ソースコード 1.1　main.cpp

ofApp.h

```
#pragma once

#include "ofMain.h"
#include "ofxOsc.h"

//スレーブのIOアドレス設定
#define IP_PROJECTOR_A1 "192.168.11.6"
#define IP_PROJECTOR_B1 "192.168.11.2"
#define IP_PROJECTOR_B2 "192.168.11.3"
#define IP_PROJECTOR_C1 "192.168.11.5"
#define IP_PA       "192.168.11.1"

#define PORT_TO_SLAVE 8000 //コントローラ->スレーブのポート番号
#define PORT_TO_CONTROLLER 7000 //スレーブ->コントローラのポート番号

#define TIMEOUT 50 //接続確認時のタイムアウト時間

class ofApp : public ofBaseApp {

public:
        //メンバメソッド宣言
    void setup();
    void update();
    void draw();

    void keyPressed(int key);

    void connectionCheck(); //接続チェック
    void getTime(); //現在時刻の取得

    void allPlay();
    void allPause(); //一時停止
    void allRewind(); //巻き戻し
    void allScreenOff();
    void allScreenOn();
```

```
    void allQR_Play();

    //メンバ変数宣言
    bool connectionState;
    bool black;

    int cnState_A1;
    int cnState_B1;
    int cnState_B2;
    int cnState_C1;
    int cnState_PA;

    int frameCount;

    //インスタンスの宣言
    ofxOscSender projector_A1, projector_B1, projector_B2, projector_C1, pa; //OSC通信
送信用
    ofxOscReceiver receiver; //OSC通信 受信用
    ofTrueTypeFont verdana; //フォント用

    string time;
    string s, m, h;
};
```

ソースコード 1.2　ofApp.h

ofApp.cpp

```
#include "ofApp.h"

//--------------------------------------------------------------
void ofApp::setup(){

    ofBackground(0, 0, 0); //背景色
    ofSetFrameRate(30); //3フレームレート設定

    //フォント設定
    ofTrueTypeFont::setGlobalDpi(72);
```

```
    verdana.load("VerdanaRef.ttf", 50);
    verdana.setLineHeight(24);
    verdana.setLetterSpacing(1.0);

    //通信設定
    projector_A1.setup(IP_PROJECTOR_A1, PORT_TO_SLAVE); //OSC通信 送信機能 初期化
    projector_B1.setup(IP_PROJECTOR_B1, PORT_TO_SLAVE);
    projector_B2.setup(IP_PROJECTOR_B2, PORT_TO_SLAVE);
    projector_C1.setup(IP_PROJECTOR_C1, PORT_TO_SLAVE);
    pa.setup(IP_PA, PORT_TO_SLAVE);

    receiver.setup(PORT_TO_CONTROLLER); //OSC通信 受信機能初期化

    cnState_A1 = 2; //0:connecting... , 1:success , 2:failed
    cnState_B1 = 2;
    cnState_B2 = 2;
    cnState_C1 = 2;
    cnState_PA = 2;

    connectionState = false;
    frameCount = 0;
    black = false;
}

//--------------------------------------------------------------
void ofApp::update(){

    getTime(); //現在時刻表示

    if (connectionState == true) {
        frameCount++;

                //スレーブからメッセージを受信している間繰り返す
        while (receiver.hasWaitingMessages()) {
            ofxOscMessage m;
            receiver.getNextMessage(&m); //スレーブからのメッセージ取得
```

```
                        //slaveから応答があったら，接続状態をsuccessにする
            if (m.getAddress() == "/pmap/connection/responce/A") { cnState_A1 = 1; }
            else if (m.getAddress() == "/pmap/connection/responce/B1") { cnState_B1 =
1; }
            else if (m.getAddress() == "/pmap/connection/responce/B2") { cnState_B2 =
1; }
            else if (m.getAddress() == "/pmap/connection/responce/C") { cnState_C1 =
1; }
            else if (m.getAddress() == "/pmap/connection/responce/PA") { cnState_PA =
1; }
            else;
        }

                //タイムアウト時に応答待ちならば，接続状態をfailedにして，終了する
        if (frameCount > TIMEOUT) {
            if (cnState_A1 == 0) cnState_A1 = 2;
            if (cnState_B1 == 0) cnState_B1 = 2;
            if (cnState_B2 == 0) cnState_B2 = 2;
            if (cnState_C1 == 0) cnState_C1 = 2;
            if (cnState_PA == 0) cnState_PA = 2;

            connectionState = false;
        }

                //全て接続成功したら，終了する
        if (cnState_A1 == 1 && cnState_B1 == 1 && cnState_B2 == 1 && cnState_C1 == 1 &&
cnState_PA == 1) connectionState = false;
    }
}

//--------------------------------------------------------------
void ofApp::draw(){

        //画面表示
    verdana.load("VerdanaRef.ttf", 30);
```

```
    ofSetColor(30, 200, 200);
    verdana.drawString(time, 530, 50);

    verdana.load("VerdanaRef.ttf", 40);
    ofSetColor(30, 200, 200);
    verdana.drawString("Connection", 50, 450);

    verdana.drawString("Media & Screen", 50, 150 - 40);

    verdana.load("VerdanaRef.ttf", 30);
    ofSetColor(30, 200, 200);
    verdana.drawString("- Projector A1", 150, 550);
    verdana.drawString("- Projector B1", 150, 600);
    verdana.drawString("- Projector B2", 150, 650);
    verdana.drawString("- Projector C1", 150, 700);
    verdana.drawString("- PA", 150, 750);

    verdana.drawString("+ Media Play", 90, 200 - 40); verdana.drawString("'   '", 480,
200 - 40);
    verdana.drawString("+ Media Pause", 90, 250 - 40); verdana.drawString("' p '", 480,
250 - 40);
    verdana.drawString("+ Media Rewind", 90, 300 - 40); verdana.drawString("' r '",
480, 300 - 40);
    verdana.drawString("+ Screen Blind", 90, 350 - 40); verdana.drawString("' q '",
480, 350 - 40);
    verdana.drawString("+ QR code play", 90, 400 - 40); verdana.drawString("' g '",
480, 400 - 40);



    verdana.drawString("+ Connection Check", 90, 500); verdana.drawString("' c '", 480,
500);

    if (cnState_A1 == 1) { ofSetColor(0, 200, 100); verdana.drawString("success", 455,
550); }
    else if (cnState_A1 == 2) { ofSetColor(200, 100, 0); verdana.drawString("failed",
470, 550); }
```

```
    else if (cnState_A1 == 0) { ofSetColor(30, 200, 200);
verdana.drawString("connecting...", 440, 550); }

    if (cnState_B1 == 1) { ofSetColor(0, 200, 100); verdana.drawString("success", 455,
600); }
    else if (cnState_B1 == 2) { ofSetColor(200, 100, 0); verdana.drawString("failed",
470, 600); }
    else if (cnState_B1 == 0) { ofSetColor(30, 200, 200);
verdana.drawString("connecting...", 440, 600); }

    if (cnState_B2 == 1) { ofSetColor(0, 200, 100); verdana.drawString("success", 455,
650); }
    else if (cnState_B2 == 2) { ofSetColor(200, 100, 0); verdana.drawString("failed",
470, 650); }
    else if (cnState_B2 == 0) { ofSetColor(30, 200, 200);
verdana.drawString("connecting...", 440, 650); }

    if (cnState_C1 == 1) { ofSetColor(0, 200, 100); verdana.drawString("success", 455,
700); }
    else if (cnState_C1 == 2) { ofSetColor(200, 100, 0); verdana.drawString("failed",
470, 700); }
    else if (cnState_C1 == 0) { ofSetColor(30, 200, 200);
verdana.drawString("connecting...", 440, 700); }

    if (cnState_PA == 1) { ofSetColor(0, 200, 100); verdana.drawString("success", 455,
750); }
    else if (cnState_PA == 2) { ofSetColor(200, 100, 0); verdana.drawString("failed",
470, 750); }
    else if (cnState_PA == 0) { ofSetColor(30, 200, 200);
verdana.drawString("connecting...", 440, 750); }
}

//--------------------------------------------------------------
//現在時刻の取得
void ofApp::getTime() {
```

```
    if (ofGetSeconds()<10)
        s = "0" + ofToString(ofGetSeconds(), 0);
    else
        s = ofToString(ofGetSeconds(), 0);
    if (ofGetMinutes()<10)
        m = "0" + ofToString(ofGetMinutes(), 0);
    else
        m = ofToString(ofGetMinutes(), 0);

    if (ofGetHours()<10)
        h = "0" + ofToString(ofGetHours(), 0);
    else
        h = ofToString(ofGetHours(), 0);

    time = h + ":" + m + ":" + s;
}

//--------------------------------------------------------------
//キー入力処理
void ofApp::keyPressed(int key){

    switch (key) {
        case 'c':
            connectionCheck();
            break;

        case ' ':
            allPlay();
            break;

        case 'p':
            allPause();
            break;

        case 'r':
            allRewind();
```

```
            break;

        case 'q':
            black = (!black);
            if (black == false) allScreenOn();
            else if (black == true) allScreenOff();
            else;
            break;

        case 'g':
            allQR_Play();
            break;

        default:
            break;
    }
}

//--------------------------------------------------------------
//接続確認
void ofApp::connectionCheck() {

        //表示を「connecting...」に変更
    cnState_A1 = 0;
    cnState_B1 = 0;
    cnState_B2 = 0;
    cnState_C1 = 0;
    cnState_PA = 0;

    //全Slaveにメッセージ送信
    ofxOscMessage message;
    message.setAddress("/pmap/connection/ask");

    //メッセージ送信
    projector_A1.sendMessage(message);
    projector_B1.sendMessage(message);
```

```
    projector_B2.sendMessage(message);
    projector_C1.sendMessage(message);
    pa.sendMessage(message);

    connectionState = true;
    frameCount = 0;
}

//--------------------------------------------------------------
//本編再生
void ofApp::allPlay() {

    ofxOscMessage message;
    message.setAddress("/pmap/media/play");

    projector_A1.sendMessage(message);
    projector_B1.sendMessage(message);
    projector_B2.sendMessage(message);
    projector_C1.sendMessage(message);
    pa.sendMessage(message);
}

//--------------------------------------------------------------
//一時停止
void ofApp::allPause() {

    ofxOscMessage message;
    message.setAddress("/pmap/media/pause");

    projector_A1.sendMessage(message);
    projector_B1.sendMessage(message);
    projector_B2.sendMessage(message);
    projector_C1.sendMessage(message);
    pa.sendMessage(message);
}
```

```
//--------------------------------------------------------------
//巻き戻し
void ofApp::allRewind() {

    ofxOscMessage message;
    message.setAddress("/pmap/media/rewind");

    projector_A1.sendMessage(message);
    projector_B1.sendMessage(message);
    projector_B2.sendMessage(message);
    projector_C1.sendMessage(message);
    pa.sendMessage(message);
}

//--------------------------------------------------------------
//スクリーン非表示
void ofApp::allScreenOff() {

    ofxOscMessage message;
    message.setAddress("/pmap/screen/off");

    projector_A1.sendMessage(message);
    projector_B1.sendMessage(message);
    projector_B2.sendMessage(message);
    projector_C1.sendMessage(message);
    pa.sendMessage(message);
}

//--------------------------------------------------------------
//スクリーン表示
void ofApp::allScreenOn() {

    ofxOscMessage message;
    message.setAddress("/pmap/screen/on");

    projector_A1.sendMessage(message);
```

```
    projector_B1.sendMessage(message);
    projector_B2.sendMessage(message);
    projector_C1.sendMessage(message);
    pa.sendMessage(message);
}

//--------------------------------------------------------------
//QR動画再生
void ofApp::allQR_Play() {

        ofxOscMessage message;
        message.setAddress("/pmap/media/QR_play");

        projector_A1.sendMessage(message);
        projector_B1.sendMessage(message);
        projector_B2.sendMessage(message);
        projector_C1.sendMessage(message);
        pa.sendMessage(message);
}
```

ソースコード 1.3　ofApp.cpp

1.5.2.　スレーブ

スレーブ（各プロジェクタ）用のソースコード（main.cpp/ ofApp.h/ ofApp.cpp）を示す.

main.cpp

```
#include "ofMain.h"
#include "ofApp.h"

int main(){
    ofSetupOpenGL(768,768,OF_FULLSCREEN); //実行画面フルスクリーン

    ofRunApp(new ofApp());
}
```

ソースコード 1.4　main.cpp

ofApp.h

```
#pragma once

#include "ofMain.h"
#include "ofxOsc.h"

#define PORT_TO_SLAVE 8000 //コントローラ->スレーブのポート番号
#define PORT_TO_CONTROLLER 7000 //スレーブ->コントローラのポート番号

#define IP_CONTROLLER "192.168.11.4" //コントローラのIPアドレス

#define PLAY_ADJUST 2 //このプロジェクターの再生開始を PLAY_ADJUST [フレーム]だけ遅らせる

class ofApp : public ofBaseApp{

public:
        //メンバメソッド宣言
    void setup();
    void update();
    void draw();

    void keyPressed(int key);

    void play();
    void pause(); //一時停止
    void rewind(); //巻き戻し
    void screenOff();
    void screenOn();
    void reply(); //接続チェックでコントローラに応答するメソッド
    void QR_play();

        //メンバ変数宣言
    bool black;
    bool count_start;
    bool movie_on;
```

```
    int videoType;
    int framecount;

        //インスタンスの宣言
        ofxOscSender sender; //OSC通信 送信用
        ofxOscReceiver receiver; //OSC通信 受信用
        ofVideoPlayer movie; //本編動画用
        ofVideoPlayer qr; //QR動画用
};
```

ソースコード 1.5　ofApp.h

ofApp.cpp

```
#include "ofApp.h"

//--------------------------------------------------------------
void ofApp::setup(){

    CGDisplayHideCursor(NULL); //カーソルを非表示(Windowsなら不要)
    ofHideCursor(); //カーソルを非表示

    ofBackground(0, 0, 0); //背景色
    ofSetFrameRate(30); //フレームレート設定

    movie.load("A_1102.mp4"); //動画ファイル読み込み
    qr.load("A_QR.mp4"); //音声ファイル読み込み

    movie.setLoopState(OF_LOOP_NONE); //動画ループしないように設定
    qr.setLoopState(OF_LOOP_NONE); //QR動画ループしないように設定

    //OSC通信 送信機能 初期化
    sender.setup(IP_CONTROLLER, PORT_TO_CONTROLLER);

    //OSC通信 受信機能初期化
    receiver.setup(PORT_TO_SLAVE);

    black = false;
```

```
    movie_on = false;
    framecount = 0;
    videoType = 0;

    //強制スタート
    //コントローラからの信号なしで再生させる場合以下の2行を有効に
    //movie_on = true;
    //count_start = true;

}

//--------------------------------------------------------------
void ofApp::update(){

    movie.update();
    qr.update();

        //コントローラからメッセージを受信している間繰り返す
    while (receiver.hasWaitingMessages()) {
        ofxOscMessage m;
        receiver.getNextMessage(&m); //コントローラからのメッセージ取得

        //本編再生
        if (m.getAddress() == "/pmap/media/play" ){
            count_start = true;
            play();
        }
        //一時停止
        else if (m.getAddress() == "/pmap/media/pause"){
            pause();
        }
        //巻き戻し
        else if (m.getAddress() == "/pmap/media/rewind") {
            count_start = false;
            rewind();
        }
```

```
        //スクリーン消灯
        else if (m.getAddress() == "/pmap/screen/off") {
            screenOff();
        }
        //スクリーン表示
        else if (m.getAddress() == "/pmap/screen/on") {
            screenOn();
        }
                //接続チェックの応答
        else if (m.getAddress() == "/pmap/connection/ask") {
            reply();
        }
                //QRコード再生
        else if (m.getAddress() == "/pmap/media/QR_play") {
            count_start=true;
            QR_play();
        }
    }


    if(count_start){
        framecount ++;
        if(framecount == PLAY_ADJUST){ //開始タイミングの微調整
            movie_on = true; //再生開始のフラグを立てる
        }
    }
    if(count_start == false){
        framecount = 0;
        movie_on = false; //再生開始のフラグを倒す
    }
}

//--------------------------------------------------------------
void ofApp::draw(){

    //動画の再生
```

```
         if(black == false && movie_on == true){
                  //本編動画
        if (videoType == 1) {
            movie.draw(0, 0, ofGetScreenWidth(), ofGetScreenHeight());
        }
                  //QRコード動画
        if (videoType == 2) {
            qr.draw(0, 0, ofGetScreenWidth(), ofGetScreenHeight());
        }
    }

}

//--------------------------------------------------------------
//キー入力処理
void ofApp::keyPressed(int key){
    if(key == 'q'){
        black = !black; //画面表示を切り替える
    }
}

//--------------------------------------------------------------
//本編再生
void ofApp::play(){
    movie.play();
    movie.setPaused(false);
    qr.setPaused(true);
    videoType = 1;
}

//--------------------------------------------------------------
//QR再生
void ofApp::QR_play() {
    qr.play();
    qr.setPaused(false);
    movie.setPaused(true);
```

```
    videoType = 2;
}

//--------------------------------------------------------------
//一時停止
void ofApp::pause() {
    movie.setPaused(true); //本編動画を一時停止
    qr.setPaused(true); //QR動画を一時停止
}

//--------------------------------------------------------------
//巻き戻し
void ofApp::rewind() {
    movie.setPosition(0.0); //本編動画を0秒地点に
    movie.setPaused(true); //本編動画を一時停止
    qr.setPosition(0.0); //QR動画を0秒地点に
    qr.setPaused(true); //QR動画を一時停止
    videoType = 0;
}

//--------------------------------------------------------------
//スクリーン非表示
void ofApp::screenOff() {
    black = true; //スクリーン消灯を有効
}

//--------------------------------------------------------------
//スクリーン表示
void ofApp::screenOn() {
    black = false; //スクリーン消灯を無効
}

//--------------------------------------------------------------
//接続確認 コントローラへの応答
void ofApp::reply() {
    ofxOscMessage message;
```

```
    message.setAddress("/pmap/connection/responce/A"); //送信内容を設定

    //時間稼ぎ
    for (int i = 0; i < 10000; i++); //すぐに応答しないように，10000フレーム待機
    sender.sendMessage(message); //コントローラへ送信
}
```

ソースコード 1.6　ofApp.cpp

1.6. 改善案

・複数の映像への対応

投影実験段階では，映像が 1 本にまとまっていないため，複数の映像を流すことができると便利だと思う．

2. プロジェクションマッピング（床面）

中庭の地面にカラフルな円を投影し、人が踏むことにより映像にアクションが起こる体験型作品を制作した．運営側が人の動きに合わせてコントローラを操作し，あたかも人の動きにより映像が変化しているような体験を提供する．USB でパソコンに接続できるコントローラと，プロジェクタで投影する映像のプログラムを製作した．ボタンと映像内の円は配置が対応しており、手元のボタンで映像に動きを与えることができる．人が踏むと円が消えていくタイプと波紋が広がるタイプの 2 作品を作成した．

2.1. ハードウェア

2.1.1. システム構成

コントローラは，USB ケーブルでパソコンに接続して使用する．コントローラに内蔵されたマイコンが，押されたボタンの情報をシリアル通信でパソコンに送信する．また，パソコンから信号を送信することで，コントローラの LED を制御し，ボタンの点灯・消灯などを制御することもできる．これらの入出力信号と連動した映像データをプロジェクタに出力している．

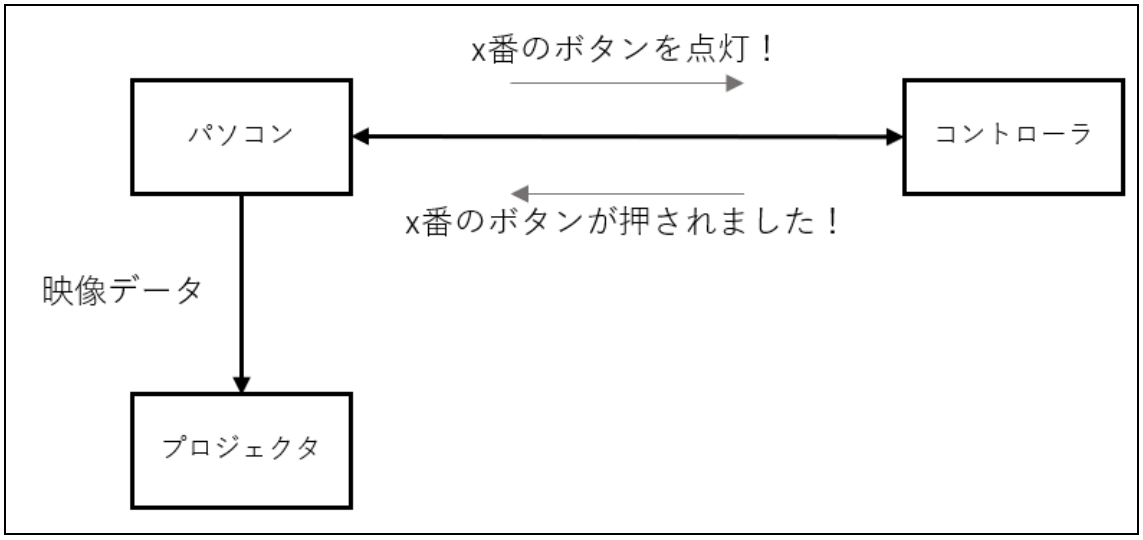

図 2.1　システム構成

ボタンの番号は図 2.2 のように定義している．また，図 2.3 のように，16 個のボタンそれぞれの下に，タクトスイッチ 2 つとフルカラーLED が設置されている．（コントローラの天板は接着していないため外すことができるが，（3D プリントした）ボタンがバラバラに取れるため面倒！設計が悪かった）

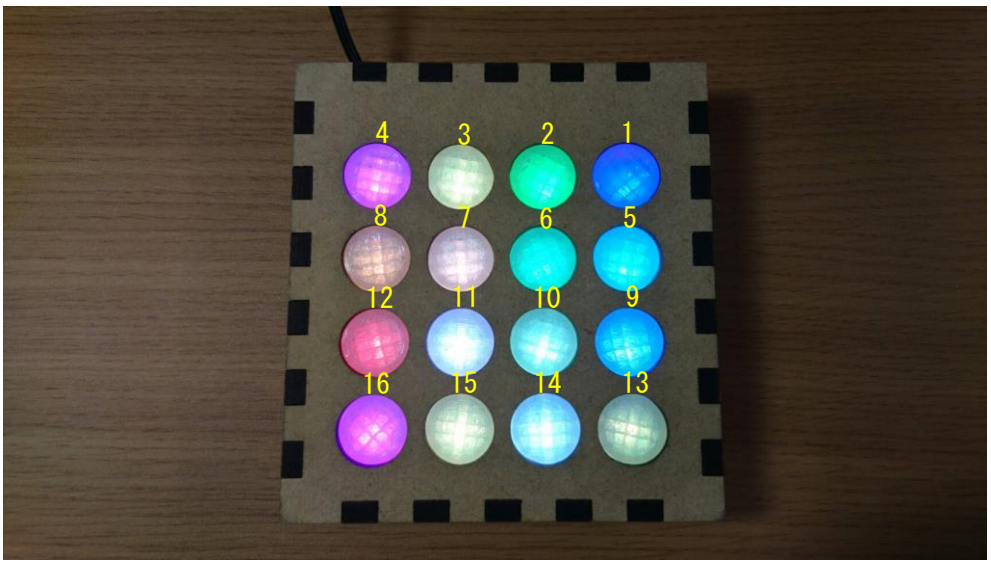

図 2.2　ボタン番号の定義

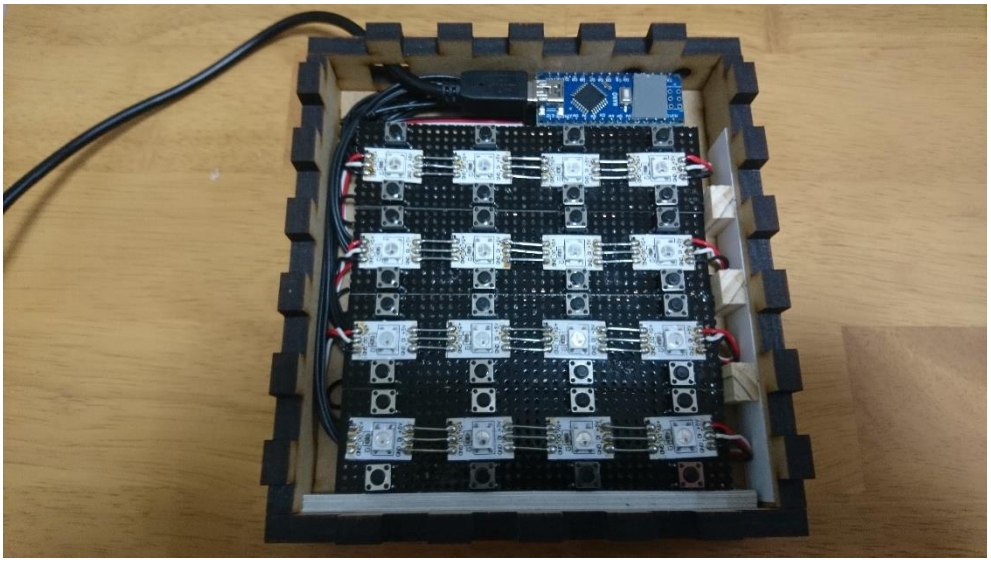

図 2.3　コントローラ内部

2.1.2. 回路図

図 2.4 に，コントローラの回路図を示す．各ボタンの下にはスイッチが 2 つずつ設置されており，どちらか，もしくは両方のスイッチが押されると信号線が GND に接地し，マイコンが認識する．フルカラーLED には個別で色を制御可能な NeoPixel を使用した．

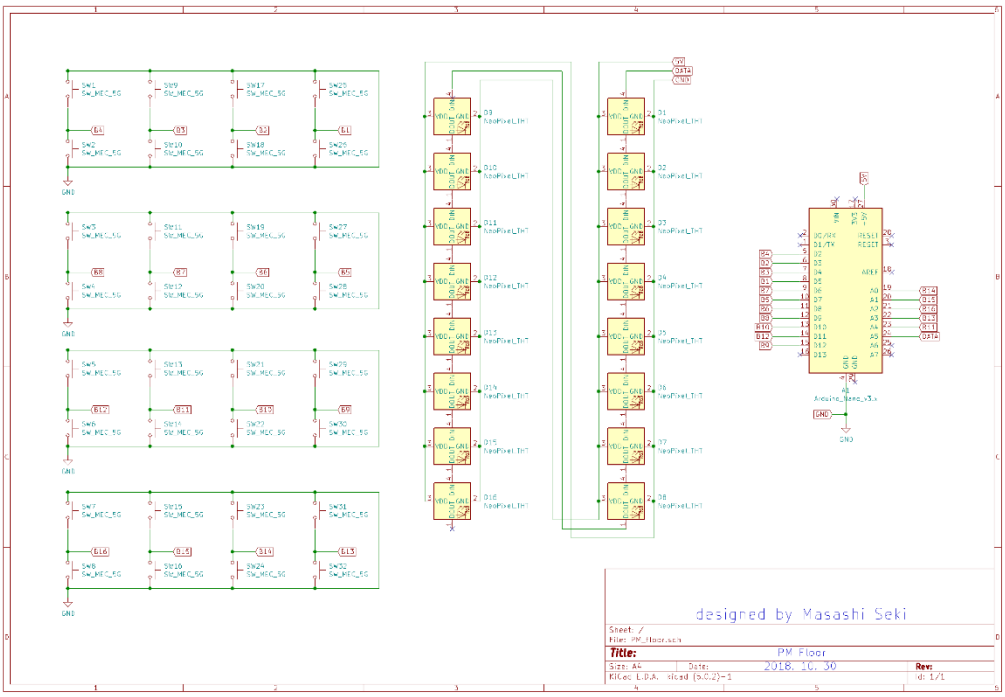

図 2.4　コントローラ回路図

2.2. ソフトウェア解説

パソコンとコントローラ間はシリアル通信により，数値データを互いに送受信している．パソコンとコントローラの各プログラム内で，受信した数値により実行する処理を記述している．

2.2.1. コントローラ側

コントローラのプログラミングには Arduino を使用している．使用しているマイコンボードは Arduino Nano である．コントローラの行っている処理は次の 2 つである．

① パソコンから数値データを受信し，受信したデータによりボタンの LED を変化させる．

② ボタンの状態をチェックし，押されたボタンの番号をパソコンに送信する．

以下に，コントローラ側のプログラムの処理について説明する．

```
#include "button.h"
```

ボタンと Arduino のポート番号の対応関係を記述した「button.h」を使用するためにインクルードしている．

```
#include <Adafruit_NeoPixel.h>
```

NeoPixel LED を使用するために必要なライブラリのインクルード．Adafruit_NeoPixel.h をダウンロードしてパソコン入れないと使用出来ないので注意．LED 関係の処理については，

ライブラリをダウンロードすると入っているスケッチの例「simple」を参照すると分かりやすいと思う.

```
#define PIN  19
#define NUMPIXELS  16
```

Arduino に接続されている LED の DATA 線のピン番号と LED の数の設定. ここは変更なしで OK.

```
double rate = 0.2
```

LED の明るさの指定. LED は 0～255 の範囲で明るさを指定できるが, かなり明るいので, 20%くらいの出力で対応した.

```
int color[16][3]={
     {0 * rate, 0 * rate, 255 * rate},
     {0 * rate, 255 * rate, 0 * rate},
     …
}
```

16 個のボタンの色を設定している. 3 つの数値はそれぞれ RGB である. 明るさ調整のために上述の変数「rate」をかけている. この色の設定が OpenFrameworks 側の円の色と対応しており, 映像データとボタンで配置を統一している.

```
Serial.begin(9600)
```

シリアル通信の開始. 現在は通信速度 9600bps を使用しており, これはパソコン側と同じ通信速度に指定する必要がある.

```
push1 = buttonCheck()
push2 = buttonCheck()
```

ボタン状態の取得.「buttonCheck()」は, 押されているボタンの番号を返す関数である. 押されていない時は 0x00 を返す.

```
if (push1 == push2) {…}
```

ボタン処理に入る. 2 回取得したボタン情報が同じであれば, ボタンを押された時の処理に入る. わざわざボタン状態を 2 回取得して比較しているのは, ノイズやチャタリングによる誤動作を防ぐためである.

```
Serial.write(0x01)
```

パソコンに数値データを送信する．「Serial.write()」の引数に指定した数値をシリアル通信でパソコンに送信する．上記の例なら，数値 0x01 を送信している．

```
if (push2 == 0x01)
else if (push2 == 0x02)
else if (push2 == 0x03)
…
```

Arduino 上で押されたボタンを識別する．変数「push1」と「push2」には押されたボタンの番号が入っているので，push2 の値により条件分岐している．

```
if (Serial.available() > 0) {…}
```

パソコンから Arduino に送られてきた数値データを確認する．「Serial.available()」は，Arduino がデータを受信すると 1 を返すので，この if 文は Arduino がデータを受信した時に実行される．

```
getData = Serial.read()
```

コントローラが受信したデータを取得する．データを受信したときに「Serial.read()」を実行すると，戻り値として1byteの受信データを得ることができる．

```
if (getData == 0x01)
else if (getData == 0x02)
else if (getData == 0x03)
…
```

コントローラが受信した数値データにより，コントローラが行う処理を決めている．パソコンから送られてきた数値は変数「getData」に入るため，この変数の値で条件分岐している．

また，本プログラムには以下の関数が用意されており，組み合わせることでプログラミング可能である．

```
reset_LED()
```

全ての LED を点灯する．

```
allOff_LED()
```

全ての LED を消灯する．

```
off_LED(int no)
```

引数に指定した番号の LED を（個別に）消灯する．

```
on_LED(int no)
```

引数に指定した番号の LED を（個別に）点灯する．

```
flash_LED(int no)
```

引数に指定した番号の LED を（個別に）点滅させる．

```
buttonCheck()
```

コントローラのボタン状態を取得する．ボタンが押されているときはそのボタン番号を返し，押されていない時は 0x00 を返す．

```
LED_blink()
```

全ての LED を 3 回点滅させる．

2.2.2. パソコン側

パソコン側は OpenFrameworks により処理をしている．パソコン側の行っている処理は次の 3 つである．

① コントローラから数値データを受信し，受信したデータにより映像を変化させている．
② コントローラの LED 状態を変化させるために数値データを送信している．
③ キーボードの状態をチェックし，各キーに割り振った数値データをコントローラに送信する．

以下に，パソコン側のプログラムの処理について説明する．

```
ofHideCursor()
CGDisplayHideCursor(NULL)
```

実行画面にカーソルを表示させないための関数．Windows の場合は「ofHideCursor()」を，Mac の場合は「ofHideCursor()」だけでは消えないことがあるため，「CGDisplayHideCursor(NULL)」も記述する．

```
serial.setup("COM3", 9600)
serial.setup("/dev/tty.usbmodem1411",9600)
```

シリアル通信のポートと通信速度の設定．関数の引数は，ポート番号と通信速度である．コントローラを接続してArduinoを認識したCOMポートの番号（場所）に合わせる．通信速度はパソコンとコントローラで合わせる必要があり，現在は9600bpsとなっている．Windows の場合は「serial.setup("COM3", 9600)」のように，Mac の場合は「serial.setup("/dev/tty.usbmodem1411",9600)」のように記述する．

```
getData = serial.readByte()
```
パソコンが受信したデータを取得する．データを受信したときに「serial.readByte()」を実行すると，戻り値として1byteの受信データを得ることができる．

```
serial.writeByte(0x01)
```
コントローラに数値データを送る．「serial.writeByte()」の引数に指定した数値をシリアル通信でコントローラに送信する．上記の例なら，数値0x01を送信している．

```
if (getData == 0x01) {
    cout << "button1 pushed" << endl;
    serial.writeByte(0x01);
}
…
```
OpenFrameworksで，押されたボタンを識別する．getDataに入った数値が押されたボタン番号に対応しているので，if文で条件分岐すればよい．ここでは，ボタンが押されるとコンソール画面に「button x pushed」と表示し，コントローラにボタンの番号を送り返している．（つまり，コントローラのボタンを押すと，そのボタンのLEDが消灯する．）

2.3. ソースコード

2.3.1. パソコン側

OpenFrameworks とコントローラ間の通信を確認するためのプログラムである．簡単のため，映像に関する処理を省略している．

Main.cpp

```
#include "ofMain.h"
#include "ofApp.h"

int main(){
    ofSetupOpenGL(1024,768,OF_WINDOW);
    ofRunApp(new ofApp());
}
```
ソースコード 2.1　Main.cpp

ofApp.h

```
#pragma once
#include "ofMain.h"

class ofApp : public ofBaseApp {
public:
	void setup();
	void update();
	void draw();
	void keyPressed(int key);

	ofSerial serial;
	int getData;
};
```

ソースコード 2.2　ofApp.h

ofApp.cpp

```
#include "ofApp.h"

void ofApp::setup() {
	ofSetFrameRate(60);
	ofHideCursor();
	//CGDisplayHideCursor(NULL);

	serial.setup("COM3", 9600); //Windows
	//serial.setup("/dev/tty.usbmodem1411",9600); //Mac
}

void ofApp::update() {
	getData = 0;
	getData = serial.readByte();

	if (getData == 0x01) {
		cout << "button1 pushed" << endl;
		serial.writeByte(0x01);
	}
```

```
        else if (getData == 0x02) {
                cout << "button2 pushed" << endl;
                serial.writeByte(0x02);
        }
        else if (getData == 0x03) {
                cout << "button3 pushed" << endl;
                serial.writeByte(0x03);
        }
        else if (getData == 0x04) {
                cout << "button4 pushed" << endl;
                serial.writeByte(0x04);
        }
        else if (getData == 0x05) {
                cout << "button5 pushed" << endl;
                serial.writeByte(0x05);
        }
        else if (getData == 0x06) {
                cout << "button6 pushed" << endl;
                serial.writeByte(0x06);
        }
        else if (getData == 0x07) {
                cout << "button7 pushed" << endl;
                serial.writeByte(0x07);
        }
        else if (getData == 0x08) {
                cout << "button8 pushed" << endl;
                serial.writeByte(0x08);
        }
        else if (getData == 0x09) {
                cout << "button9 pushed" << endl;
                serial.writeByte(0x09);
        }
        else if (getData == 0x0A) {
                cout << "button10 pushed" << endl;
                serial.writeByte(0x0A);
        }
```

```
        else if (getData == 0x0B) {
                cout << "button11 pushed" << endl;
                serial.writeByte(0x0B);
        }
        else if (getData == 0x0C) {
                cout << "button12 pushed" << endl;
                serial.writeByte(0x0C);
        }
        else if (getData == 0x0D) {
                cout << "button13 pushed" << endl;
                serial.writeByte(0x0D);
        }
        else if (getData == 0x0E) {
                cout << "button14 pushed" << endl;
                serial.writeByte(0x0E);
        }
        else if (getData == 0x0F) {
                cout << "button15 pushed" << endl;
                serial.writeByte(0x0F);
        }
        else if (getData == 0x10) {
                cout << "button16 pushed" << endl;
                serial.writeByte(0x10);
        }
}

void ofApp::draw() {
        ofSetColor(255, 255, 255);
}

void ofApp::keyPressed(int key) {
        if (key == 'r')
                serial.writeByte(0xFF);
}
```

ソースコード 2.3　ofApp.cpp

2.3.2. コントローラ側

コントローラの Arduino に書き込むプログラムを示す（本番と同じもの）.

PM_floor_v1.0.ino

```
#include "button.h"

#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
#include <avr/power.h>
#endif

#define PIN    19 //A5
#define NUMPIXELS    16
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

double rate = 0.2; //brightness of LED

int buttonState[16];
int color[16][3] = {

  {0 * rate, 0 * rate, 255 * rate},
  {0 * rate, 255 * rate, 0 * rate},
  {255 * rate, 255 * rate, 0 * rate},
  {255 * rate, 0 * rate, 255 * rate},
  {0 * rate, 255 * rate, 255 * rate},
  {0 * rate, 255 * rate, 70 * rate},
  {255 * rate, 183 * rate, 76 * rate},
  {221 * rate, 132 * rate, 22 * rate},
  {67 * rate, 135 * rate, 233 * rate},
  {149 * rate, 255 * rate, 101 * rate},
  {255 * rate, 255 * rate, 255 * rate},
  {255 * rate, 49 * rate, 25 * rate},
  {156 * rate, 167 * rate, 22 * rate},
  {168 * rate, 239 * rate, 175 * rate},
  {244 * rate, 250 * rate, 37 * rate},
```

```
  {225 * rate, 0 * rate, 178 * rate},
};

int push1;
int push2;
byte getData;

void setup() {

#if defined (__AVR_ATtiny85__)
  if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
#endif
  pixels.begin();

  LED_blink(); //LED blink

  for (int i = 0; i < 19; i++) {
    if (i != 13)
      pinMode(i, INPUT_PULLUP);
  }

  Serial.begin(9600);
  reset_LED();
}

void loop() {
  push1 = 0;
  push2 = 0;

  //ボタンチェック
  push1 = buttonCheck();

  if (push1 != 0) {
    delay(50);
    push2 = buttonCheck();
    if (push1 == push2) {
```

```
      //ボタン押された時の処理
      if (push2 == 0x01) Serial.write(0x01);
      else if (push2 == 0x02) Serial.write(0x02);
      else if (push2 == 0x03) Serial.write(0x03);
      else if (push2 == 0x04) Serial.write(0x04);
      else if (push2 == 0x05) Serial.write(0x05);
      else if (push2 == 0x06) Serial.write(0x06);
      else if (push2 == 0x07) Serial.write(0x07);
      else if (push2 == 0x08) Serial.write(0x08);
      else if (push2 == 0x09) Serial.write(0x09);
      else if (push2 == 0x0A) Serial.write(0x0A);
      else if (push2 == 0x0B) Serial.write(0x0B);
      else if (push2 == 0x0C) Serial.write(0x0C);
      else if (push2 == 0x0D) Serial.write(0x0D);
      else if (push2 == 0x0E) Serial.write(0x0E);
      else if (push2 == 0x0F) Serial.write(0x0F);
      else if (push2 == 0x10) Serial.write(0x10);
   }
}

//受信データチェック
if (Serial.available() > 0) {
   getData = 0;
   getData = Serial.read();

   if (getData == 0x01) off_LED(0x01);
   else if (getData == 0x02) off_LED(0x02);
   else if (getData == 0x03) off_LED(0x03);
   else if (getData == 0x04) off_LED(0x04);
   else if (getData == 0x05) off_LED(0x05);
   else if (getData == 0x06) off_LED(0x06);
   else if (getData == 0x07) off_LED(0x07);
   else if (getData == 0x08) off_LED(0x08);
   else if (getData == 0x09) off_LED(0x09);
   else if (getData == 0x0A) off_LED(0x0A);
   else if (getData == 0x0B) off_LED(0x0B);
```

```
      else if (getData == 0x0C) off_LED(0x0C);
      else if (getData == 0x0D) off_LED(0x0D);
      else if (getData == 0x0E) off_LED(0x0E);
      else if (getData == 0x0F) off_LED(0x0F);
      else if (getData == 0x10) off_LED(0x10);

      else if (getData == 0xFF) reset_LED();
      else if (getData == 0xFE) allOff_LED();
      else if (getData == 0xFD) flash_LED(0x04);
      else if (getData == 0xFC) on_LED(0x04);
      //else if (getData == 0xFB) LED_blink();
   }
   delay(1);
}

void reset_LED() {
   for (int i = 0; i < 16; i++)
      pixels.setPixelColor(i, pixels.Color(color[i][0], color[i][1], color[i][2]));
   pixels.show();
}

void allOff_LED() {
   for (int i = 0; i < 16; i++)
      pixels.setPixelColor(i, pixels.Color(0, 0, 0));
   pixels.show();
}

void off_LED(int no) {
   pixels.setPixelColor(no - 1, pixels.Color(0, 0, 0));
   pixels.show();
}

void on_LED(int no) {
   pixels.setPixelColor(no - 1, pixels.Color(color[no - 1][0], color[no - 1][1], color[no -
1][2]));
   pixels.show();
```

```
}

void flash_LED(int no) {
  for (int i = 0; i < 7; i++) {

    pixels.setPixelColor(no - 1, pixels.Color(0, 0, 0));
    pixels.show();
    delay(50);
    pixels.setPixelColor(no - 1, pixels.Color(color[no - 1][0], color[no - 1][1], color[no - 1][2]));
    pixels.show();
    delay(50);
  }
}

int buttonCheck() {
  int push = 0;
  if (digitalRead(B1) == LOW) push = 1;
  else if (digitalRead(B2) == LOW) push = 2;
  else if (digitalRead(B3) == LOW) push = 3;
  else if (digitalRead(B4) == LOW) push = 4;
  else if (digitalRead(B5) == LOW) push = 5;
  else if (digitalRead(B6) == LOW) push = 6;
  else if (digitalRead(B7) == LOW) push = 7;
  else if (digitalRead(B8) == LOW) push = 8;
  else if (digitalRead(B9) == LOW) push = 9;
  else if (digitalRead(B10) == LOW) push = 10;
  else if (digitalRead(B11) == LOW) push = 11;
  else if (digitalRead(B12) == LOW) push = 12;
  else if (digitalRead(B13) == LOW) push = 13;
  else if (digitalRead(B14) == LOW) push = 14;
  else if (digitalRead(B15) == LOW) push = 15;
  else if (digitalRead(B16) == LOW) push = 16;
  return push;
}
```

```
void LED_blink(){
    for (int i = 0; i < 3; i++) {
      for (int i = 0; i < 16; i++)
         pixels.setPixelColor(i, pixels.Color(0, 20, 20)); //green
      pixels.show();
      delay(500);

      for (int i = 0; i < 16; i++)
         pixels.setPixelColor(i, pixels.Color(0, 0, 0));
      pixels.show();
      delay(500);
   }
}
```

ソースコード 2.4　PM_floor_v1.0.ino

button.h

```
#define B1 5
#define B2 3
#define B3 4
#define B4 2

#define B5 7
#define B6 8
#define B7 6
#define B8 9

#define B9 12
#define B10 10
#define B11 18 //A4
#define B12 11

#define B13 17 //A3
#define B14 16 //A2
#define B15 15 //A1
#define B16 14 //A0
```

ソースコード 2.5　button.h