卒業論文

団扇を使用した簡易的な 3D 表示装置に関する研究

提出者　　伊藤宏樹

提出年月日　平成 31 年 1 月 24 日

指導教員　　山之上卓 教授

団扇を使用した簡易的な 3D 表示装置に関する研究

情報工学科 5415003 伊藤宏樹

研究概要

この研究では,LED マトリックスパネルを利用して立体映像を作るシステムの開発について述べる. このシステムを用いて簡単な三角錐(ピラミッド)を空間に描画することができた.

このシステムは団扇に貼り付けた LED マトリックスパネルと小型マイコンボードである Arduino と電子ブザーなどで構成されている. Arduino には 3 次元画像を輪切りにして並べた 2 次元画像が記憶されている. LED マトリックスには,この 2 次元画像が表示され,そこから順番に次々と 3 次元画像の輪切りである 2 次元画像が表示されていく. 移動速度を早くすると,人間の目に残像が残り,あたかも 3 次元表示が行われているように見える.

目次

1. まえがき

普段街で見かける電光掲示板は,平面的な表現方法が主流である. しかし,平面的な表示よりも 3 次元的な表示の方が人の目を引き付ける. 団扇に企業情報を記載し配布して広告効果を狙う手法があるが,これを 3D で表示できるようになれば広告効果は格段に上がる.

完成すれば,広告や祭りなどでの出店もできる可能性があり,万人に利用してもらうことができる.

原理としては,強い光による残像を利用して前の残像が残っているうちに次々と発光パターンを変え,さも同時に表示しているかのように見せる.

しかし,既存のものを作ったところで面白くないとの指摘から,2D ではなく,3D で同じようにアイコンや広告,キャラクターなどを表示することができれば,より,印象的でより良い広告や話題になると考え立体表示に挑戦しようと考えた.

さらに,実用性を与えるために動作として似ている団扇を使うこととした.

関連研究として,残像について・DotStae と Arduino を使った Genesis Poi[(1)]があるが,これは 3 次元表示を行うものではない. 3 次元表示装置として,バードンの空間立体描画[(2)]があるが,これは高出力レーザを利用しており,大掛かりであり,注意して利用しないと危険である. 3 次元表示が可能な POV として,チーム 3D-POV の 3D-POV[(3)]がある. これはモータで 2 次元 LED マトリックスを回転させて残像により 3 次元表示を行うものであるが,モータを利用しているので,本システムほど手軽ではない.

団扇と加速度センサーを使用して 3 次元表示を行うもの[(4)]もあるが,本システムでは一定の間隔で扇ぐことを前提として設計しているため加速度センサーを使用する必要がない.

以下,完成系の写真を挙げる.

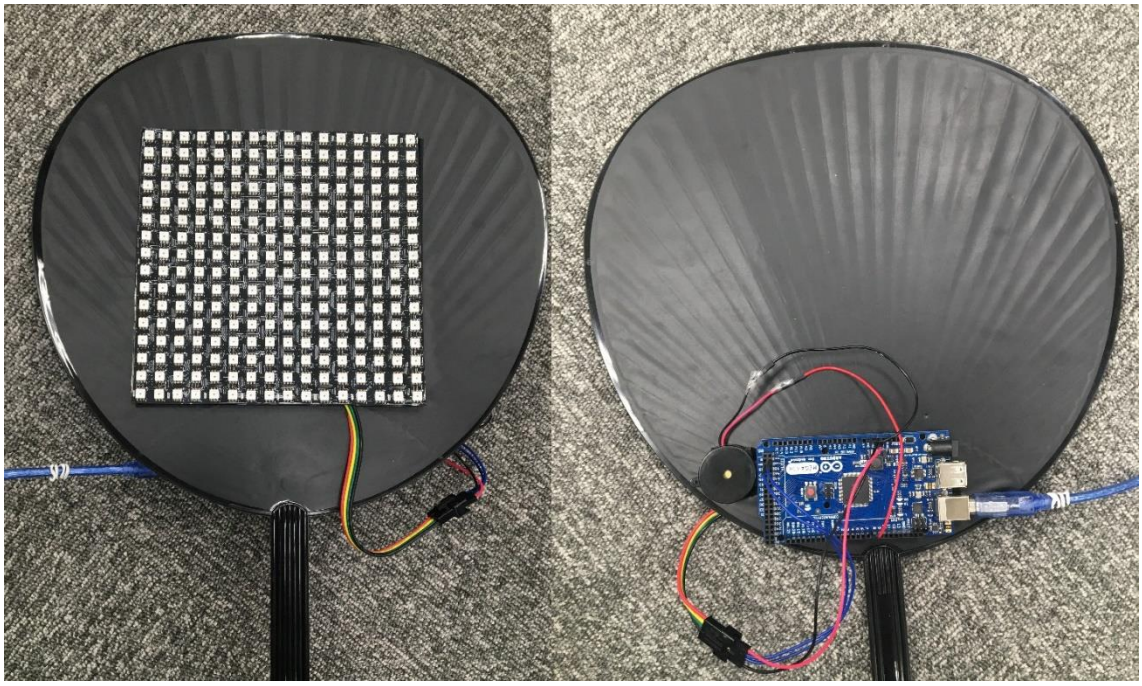

図 1　完成系裏表

2. システム概要

団扇に LED マトリックスパネルを張り付け,表示させたい 3 次元画像を輪切りにした 2 次元画像を連続的に表示させる．2 次元画像を往復描写させることにより,扇ぐ際に残像を残りやすくしている．一定の間隔で往復するよう設計し,電圧ブザーの音によって扇ぐタイミングを合わせるようにした．

Arduino を団扇の手元付近に取り付け,USB ジャックでパソコンに接続している．これにより電力の供給とパソコンからの指令を供給している．

スマホを Arduino に接続し,スマホから編集できるアプリを使用すれば,手軽にデザインを作れ,その場で変更できるようになるであろう．

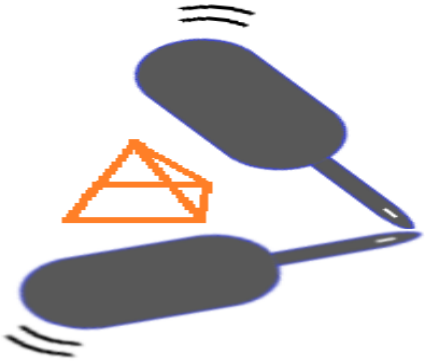

図 2 3 次元画像の表示イメージ

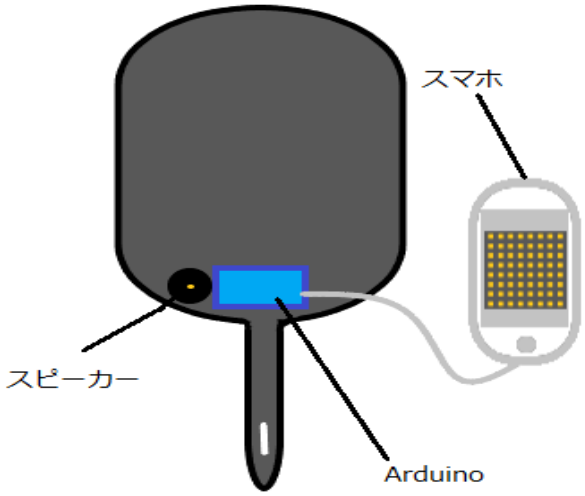

図 3 スマホから Arduino を編集するイメージ

3. ハードウェア

本研究では,団扇に LED マトリックスパネルをつけて扇げば 3 次元画像が映し出される必要があるため,装置自体をコンパクトにまとめる必要があった. よって,簡単に扱えるマイコンボードである Arduino を使用した.

残像をうまく映すためには一定の速度で動かし続けなければならない. そこで,電子ブザーでメトロノームのように一定の間隔で音を出し,タイミングをつかみやすくできるように工夫した.

LED マトリックスパネルには,薄型で,軽量かつ団扇に貼り付けることができる DotStaeMatrix を使用した.

3.1 Arduino(アルディーノ)

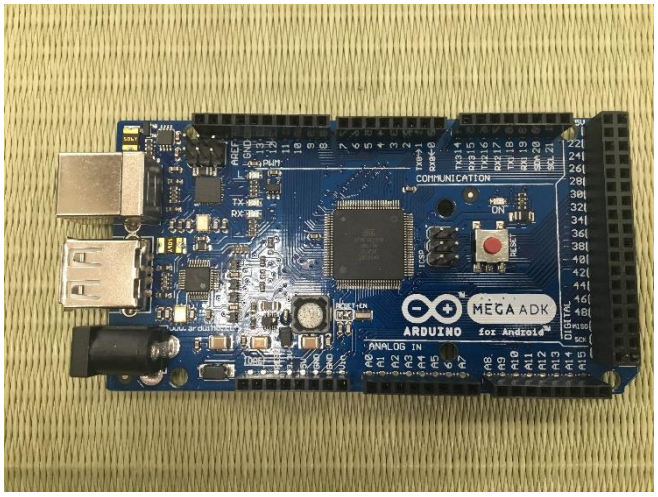

図 4 Arduino(アルディーノ)

Arduino は,入出力のポートを備えたマイクロコントローラ(マイコン)である.

Arduino 専用のソフトを使い,スケッチ(プログラム)をマイコンに書き込むことで,接続したツールを動かすことができる. 言語は C++に似ている Arduino 言語が使用される. 主に,setup 関数と loop 関数を使い setup で loop の準備を行い,loop で,反復実行するような形が多い. スタートや配列への設定など 1 度でよいものを setup で実行し,反復実況したいものを loop で実行する.

3.2 電圧ブザー

図 5 電圧ブザー

電圧ブザーとして,KPE-126 を利用した.

小型で軽量なため,使い勝手が良い.

3.3 DotStaeMatrix

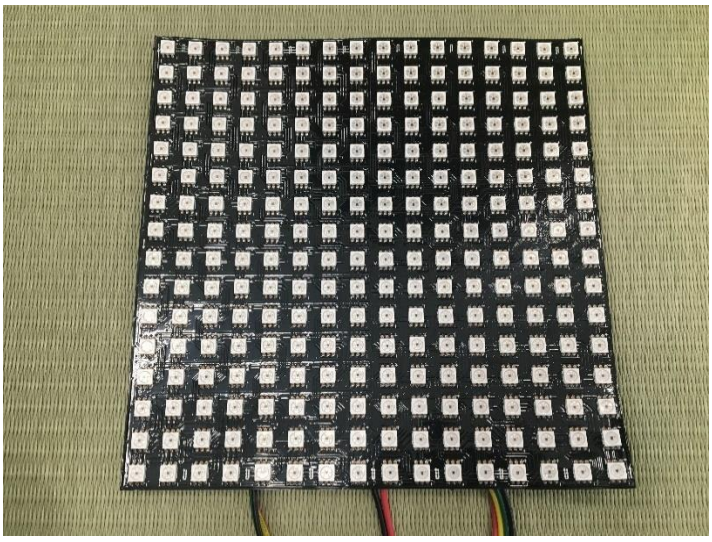

図 6 DotStaeMatrix

2 次元 LED マトリックスパネルとして,DotStaeMatrix を利用した. このパネルは高輝度,高速で LED を点灯させることができるツールである.

パネルに LED が 16＊16 個配列されたものを使用する.

```
Adafruit_DotStarMatrix matrix = Adafruit_DotStarMatrix(
16, 16,
DS_MATRIX_TOP + DS_MATRIX_RIGHT +
DS_MATRIX_ROWS + DS_MATRIX_ZIGZAG,
DOTSTAR_BRG);
```

図 7 DotStar 設定の例

このパネルは,図のようなスケッチを実行することで設定を行うことができる.

2 行目は,左から,縦の LED 数,横の LED 数.

3 行目は,左から,縦のスケッチ開始点,横のスケッチ開始点(TOP/BOTTOM・RIGHT/LEFT)の中から一つずつ選択可能である.

4 行目は,左から,開始点からの移動方向・DotStar の配列の方式を表す.ここで,(縦方向:COKUMNS/横方向:ROWS・列の左から開始し行末から次の行の左から続ける:PROGRWSSIVE/ジグザグに進む:ZIGZAG)の中から 1 つずつを選択可能である.

5 行目は,色使いのモードを表す.ここで,(KHZ800/KHZ400/GRB/RGB/BRG)を選択可能である.

その他の設定を行うため,以下の関数を利用することができる.

setBrightness(n);

n は明るさを表し,0～100 の範囲で明るさのレベルを調節できる.

n の値を極端に大きくすると膨大な電流が流れるため,Arduino が壊れる恐れがあるので注意する. 40 で十分な明るさがあった.

Color(a,b,c);

色使いモードで設定した順番に 0～255 の数値で色を作ることができる.

3.4 配線

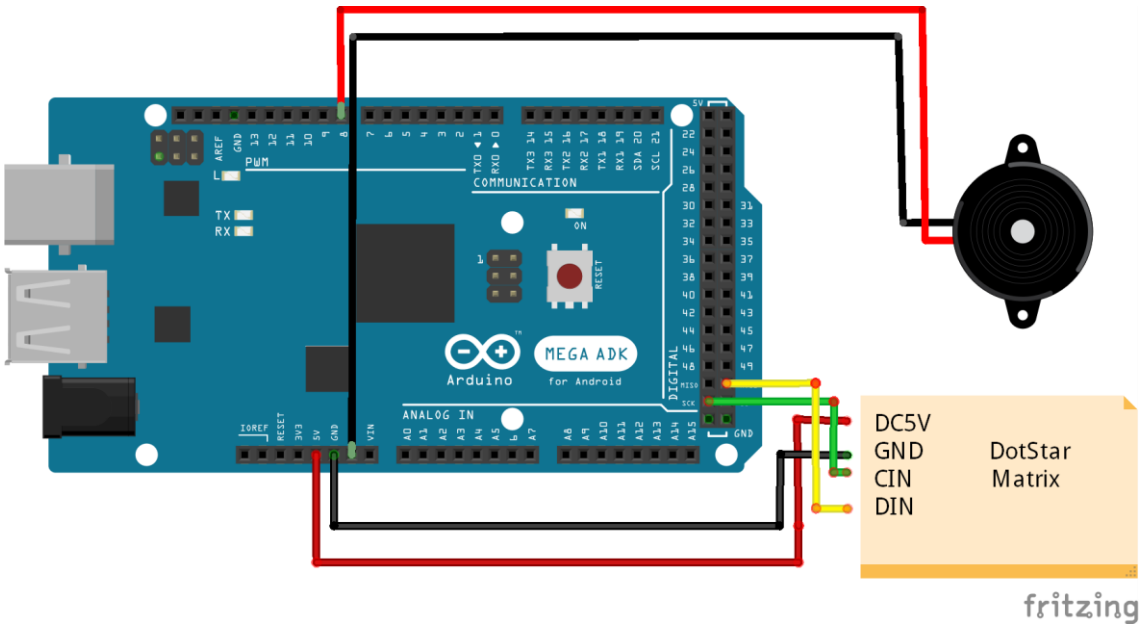

図 8 配線図

配線は図のように行う. DotStar からは電源＋,電源－,と信号線 2 本の 4 本の線が出ている. これと,電圧ブザー,Arduino を以下のように接続する.

MOSI:DATAPIN(DotStar)

SCK:CLOCLIN(DotStar)

5V:DC5V(DotStar)

GND:GND(DotStar)

GND:(電圧ブザー)

8:(電圧ブザー)

4. ソフトウェア

プログラムの大まかな流れとしては,まず描画データを配列に格納して,その後 loop 関数で一つずつ表示させることにした．表示させる際に,描画データ 0～7 を往復させる．こうすることにより団扇を仰ぐ際に残像が残りやすくなっている．往復させる描画データの最初と最後にそれぞれ違う音を出すことにより,扇ぐタイミングをつかめるようになっている．図 8 に loop 関数の中の処理のフローチャートを示す．

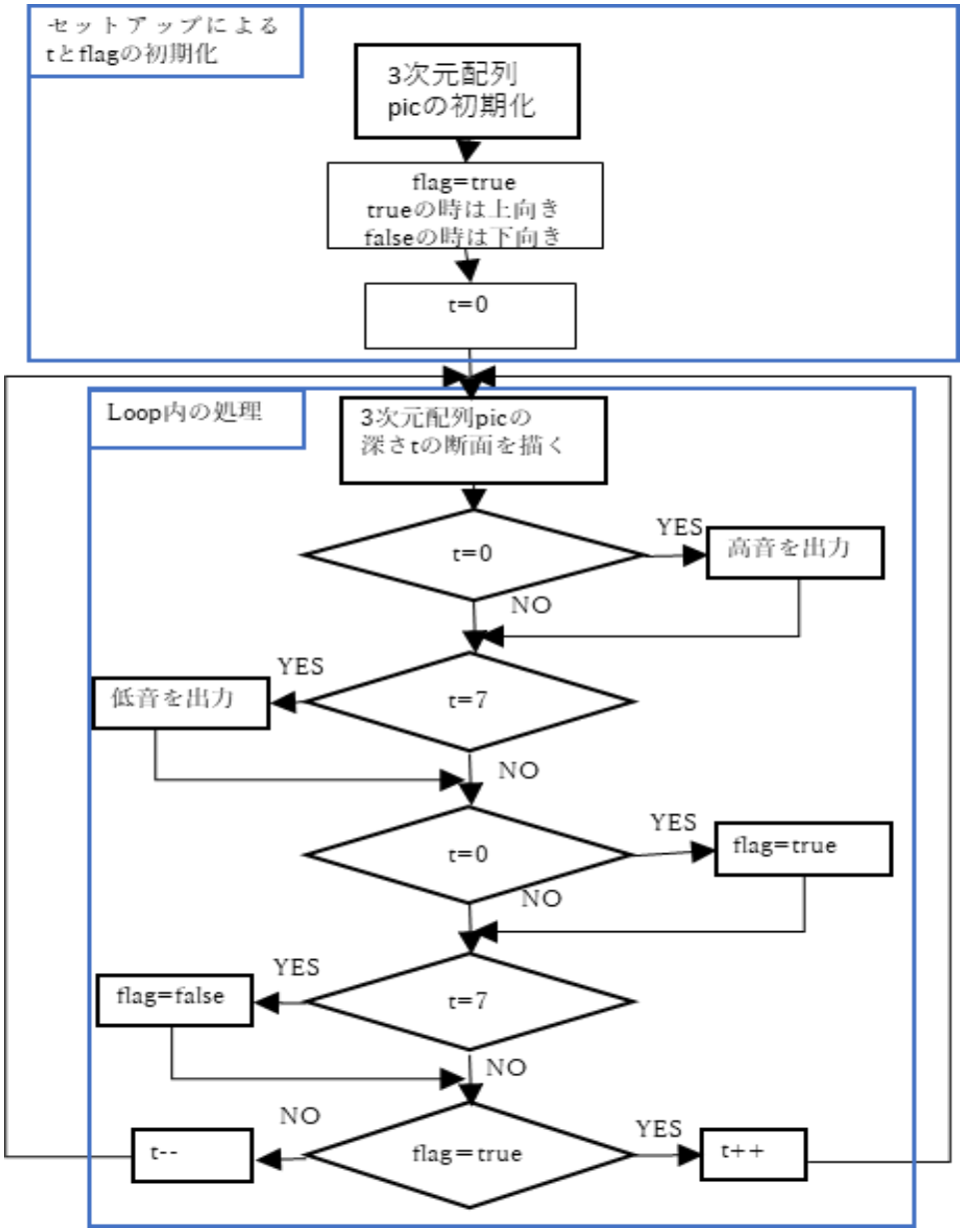

図 9 描画処理フローチャート

4.1 Include/define

プログラムでは以下の include 文によって必要なライブラリが利用できるようにしている.

```
#include <Wire.h>

#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_DotStarMatrix.h>
#include <Adafruit_DotStar.h>

#ifndef PSTR
#define PSTR // Make Arduino Due happy
#endif

#define VWMax 16
#define VHMax 16
#define RWMax 16
#define RHMax 16

#define DATAPIN  4
#define CLOCKPIN 5
```

図 10 Include/define

1 行目は Arduino にジャンパーケーブルなどのワイヤーでの接続を有効にするために必要.

2 つ目の項目は DotStar(Adafruit 製)を使用するために必要.

4 つ目の項目,VWMax・VHMax・RWMax・RHMax は次の項目で説明する. ここで最大値を 16 に設定している.

5 つ目の項目は使用する Arduino のピンの番号である. 実験用に今回の場合は DotStar に描画データを出力するために 4 番ピン,時刻データを出力するために 5 番ピンを使用した.

4.2 描画処理

描画処理のための関数の定義として以下のスケッチを使用した.

```
const uint16_t colors[] = {
  matrix.Color(255, 0, 0), matrix.Color(0, 255, 0), matrix.Color(0, 0, 255) };

uint16_t backGround;
uint16_t realGraphicsArea[RWMax][RHMax];
uint16_t pic[8][RWMax][RHMax];
```

図 11 描画処理の関数定義

const は変数を読み取り専用にするための修飾型で,主に定数の定義として使われ,値の変更を不可能にする.

RWMax は realGraphicsArea の横幅の最大値であり,RHMax は縦幅の最大値である. 同様に VWMax・VHMax は virtualGrphicsArea の横幅と縦幅である. 今回使用したツールの場合 16 が代入される.

7 行目の pic[n][][]の n はつかう画像の数(深さ)で変更する. 今回は 8 枚の四角を描画したので 8 となっているが,グローバル変数nを定義しあとから変更することもできたと思う.

```
for(int i=0;i<RWMax;i++){
  for(int j=0;j<RHMax;j++){

    if(i>=1 && i<=14 && j>=1 && j<=14 ){
      pic[7][i][j]=matrix.Color(0,0,0);
    }
    else if(i>=0 && i<=15 && j>=0 && j<=15 ){
      pic[7][i][j]=matrix.Color(255,255,255);
    }
    else{
      pic[7][i][j]=matrix.Color(0,0,0);
    }
  }
}
```

図 12 描画処理の例

描画処理は指定した場所のLEDを光らせるために,その場所の配列に色を格納する.

指定した場所の明かりを消すためには color(0,0,0)[黒]を格納する.

今回の研究では光る場所を転々と変える必要があるため,次のループで一つ前の指定した範囲を黒で消し,新たに次の個所を光らせるようにした.

4.3 音の処理

音の出力処理のための関数の定義として以下のスケッチ(図)を使用した.

```
void hiSound()
{
  int i = 0;
  while(i<50) {
      digitalWrite(speaker, HIGH);
      delay(1);
      digitalWrite(speaker, LOW);
      delay(1);
      i = i + 1;
  }
}

void lowSound()
{
  int i = 0;
  while(i<50) {
      digitalWrite(speaker, HIGH);
      delay(10);
      digitalWrite(speaker, LOW);
      delay(10);
      i = i + 10;
  }
}
```

図 13 音の処理の関数定義

2 つ目の項目で高い音を出力できるように定義している. while 文で音を出力する時間を設定している.

2 つ目の項目で低い音を出力できるように定義している. 高い音と低い音の出力時間を同じにするために,i の増加量を調節している.

4.4 loop の中の全体の処理

```
{
 matrix.fillScreen(0);

 for(int i=0;i<RWMax;i++){
    for(int j=0;j<RHMax;j++){
      uint16_t c=pic[t][i][j];
      matrix.drawPixel(i,j,c);
    }
 }
 matrix.show();

 if (t == 0) {
   hiSound();
 } else if (t == 7) {
   lowSound();
 }

 if(t==0){
   flag=true;
 }
   else if(t==7){
     flag=false;
   }
 if(flag==true){
   t++;
 }
   else{
     t--;
   }
 }
```

図 14 loop の中の全体の処理

まず上で格納した描画データを 1 つずつ表示させるために変数を用意し(int 型:今回は t)反復させる．描画データが 8 個だけなので 8 個目になったら t をマイナスさせていく．これによりに扇を往復させる時により見やすく表示される．

さらに,描画の最初に高い音,最後に低い音を出力させる．これにより動かすタイミングをつかむことができる．

5. 使い方

この装置は Arduino IDE をパソコンにダウンロードして使用する必要がある. Arduino IDE を起動し Arduino 本体をパソコンに接続してシリアルモニタを起動する.「Type key when ready...」と表示されたら Enter キーを入力すると三角錐を描画するためのプログラムが起動する. プログラム止める場合は,Arduino 本体の RESET ボタンを押すと止まる.

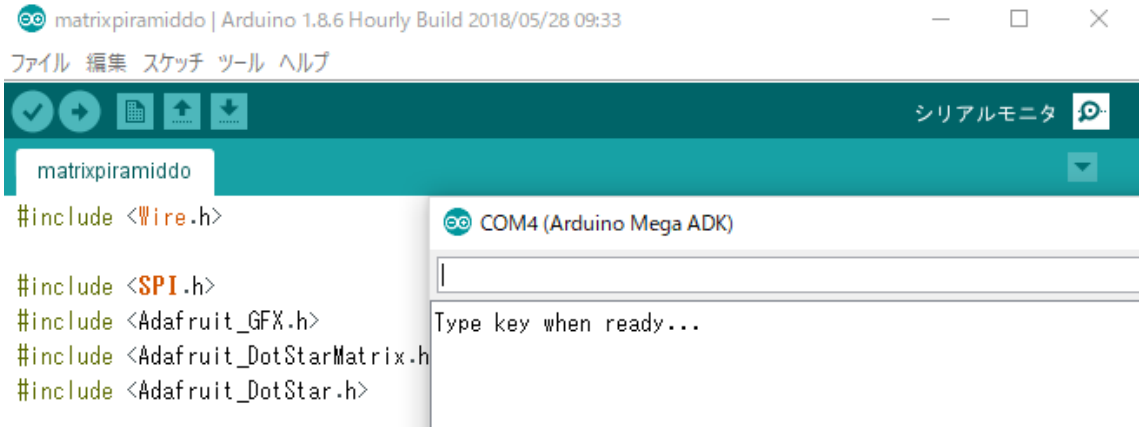

図 15 Arduino AED とシリアルモニタ

6. 実験

本実験では,加速度センサーを使用した物と加速度センサーを使用せず,電子ブザーによるタイミング調整をした場合との比較と,SPI を使用した場合と,使用しなかった場合を比較する.

6.1 加速度センサーを使用したものとの比較

加速度センサーを使用した場合は,一定の速度に達すると描画が初期化され,扇ぐスピードによってはうまく形が表示されなかった.

電子ブザーを使用した場合は,タイミングをつかめば残像がうまく表示され,比較的きれいに映し出された.

図 16 実験結果

6.2 SPI を使った場合との比較

Serial Peripheral Interface(SPI)は,マイクロコントローラにひとつあるいは複数のデバイスを接続する目的で使われる,短距離用の簡便な同期通信プロトコルである[5]. これを使用すると通信速度が速くなり,往復処理が速くなる. これを使う場合とDATAPINとCLOCKPINを自分で指定した場合を比較した.

その結果,SPI を使用した場合は 10 秒間に約 24 往復,使用しない場合は 10 秒間に約 16 往復する結果となった. 残像を効果的に映し出すにはより早く往復させ扇ぐスピードを上げる必要があるため. 本研究では SPI を採用することとなった.

7. この研究の負の側面

物体を表示させる際に光が角度や位置によっては,まぶしく見える傾向があるので,光を加減して表示させる必要がある. LED の激しい点滅により光過敏性発作を起こす可能性がある.

また,扇ぐ動作で物体を表示させることから,簡易的につけられた部品が飛んで周りの人に当たる可能性があり,使用する際に部品の点検が必要である.

8. 関連研究

関連研究として,残像について・DotStae と Arduino を使った Genesis Poi[1]があるが,これは 3 次元表示を行うものではない. 3 次元表示装置として,バードンの空間立体描画[2]があるが,これは高出力レーザを利用しており,大掛かりであり,注意して利用しないと危険である. 3 次元表示が可能な POV として,チーム 3D-POV の 3D-POV[3]がある. これはモータで 2 次元 LED マトリックスを回転させて残像により 3 次元表示を行うものであるが,モータを利用しているので,本システムほど手軽ではない.

団扇と加速度センサーを使用して 3 次元表示を行うもの[4]もあるが,本システムでは一定の間隔で扇ぐことを前提として設計しているため加速度センサーを使用する必要がない.

9. むすび

先行研究よりは簡易的に 3 次元表示ができた. 描画速度について遅すぎると残像が残りづらくなり,速すぎる扇ぐ動作が遅れるため,タイミング調整に苦戦した. 配線を間違えて LED マトリックスパネルを壊してしまうことがあったため, コードの色の確認は慎重に行うべきだった.

今後の課題としては,現時点では片面のみ LED マトリックスパネルを張り付けているが,両面にすることでさらに見やすくなると考えられる. Arduino を団扇の手元付近につけているが,扇ぐ際に団扇のしなりが大きくなるので改良が必要である. スマホ等で気軽に描画する画像を編集できればさらに研究の幅が広がるであろう.

10. 参考文献

(1)Genesis Poi
https://learn.adafruit.com/genesis-poi-dotstar-led-persistence-of-vision-poi/overview

(2)バートンの空間立体描画
https://www.aist.go.jp/aist_j/press_release/pr2007/pr20070710/pr20070710.html

(3)チーム 3D-POV
https://makezine.jp/event/makers2016/チーム_3d-pov/#a0_3d-pov/

(4)松藤一樹:「手軽に,裸眼で,見る場所や向きを自由に移動できる 3 次元表示装置に関する研究」,福山大学工学部情報工学科平成 29 年 2 月 10 日

(5)Arduino 日本語リファレンス
http://www.musashinodenpa.com/arduino/ref/index.php?f=1&pos=539

11. 付録

```
#include <Wire.h>

#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_DotStarMatrix.h>
#include <Adafruit_DotStar.h>

#ifndef PSTR
#define PSTR // Make Arduino Due happy
#endif

#define VWMax 16
#define VHMax 16
#define RWMax 16
#define RHMax 16

#define DATAPIN    4
#define CLOCKPIN 5



Adafruit_DotStarMatrix matrix = Adafruit_DotStarMatrix(
16, 16,
DS_MATRIX_TOP + DS_MATRIX_RIGHT +
DS_MATRIX_ROWS + DS_MATRIX_ZIGZAG,
DOTSTAR_BRG);

const uint16_t colors[] = {
  matrix.Color(255,  0,  0),  matrix.Color(0,  255,  0),
matrix.Color(0, 0, 255) };
```

```
uint16_t backGround;
uint16_t realGraphicsArea[RWMax][RHMax];
uint16_t pic[8][RWMax][RHMax];


void clearRealGraphicsArea(){
  int i,j;
    for( i=0;i<RWMax;i++){
      for( j=0;j<RHMax;j++){
          realGraphicsArea[i][j]=backGround;
      }
    }
}


#define analogInMax 8
#define digitalInMax 6
#define digitalOutMax 14
int analogIns[analogInMax];
int digitalIns[digitalInMax];
int digitalOuts[digitalOutMax];
int digitalVal;

int realX;
int lp; // last received font position on the virtualGraphicsArea
int dp; // displayed font position on the virtualGraphicsArea
int vfmax; // virtual font area max ... = VWMax/16;
int t;
float asam;

const int speaker = 8;
```

```
void setup(void)
{
     Wire.begin();
  int error;
  uint8_t c;
  Serial.begin(9600);

   // スピーカーをつないだピンを出力に設定する
  pinMode(speaker, OUTPUT);

  t=0;
   for(int i=0;i<digitalInMax;i++) digitalIns[i]=i;
   for(int i=0;i<digitalOutMax;i++) digitalOuts[i]=i;
   for(int i=0;i<digitalInMax;i++)
       pinMode(digitalIns[i],INPUT);
   for(int i=digitalInMax+1;i<digitalOutMax;i++)
       pinMode(digitalOuts[i],OUTPUT);
   analogIns[0]=A0;
   analogIns[1]=A1;
   analogIns[2]=A2;
   analogIns[3]=A3;
   analogIns[4]=A4;
   analogIns[5]=A5;
   analogIns[6]=A6;
   analogIns[7]=A7;

   delay(1);
  matrix.begin();
  matrix.setBrightness(40);
  backGround=matrix.Color(0,0,0);
  realX=0;
```

```
lp=0;
dp=0;
vfmax=VWMax/16;

for(int i=0;i<RWMax;i++){
  for(int j=0;j<RHMax;j++){

    if(i>=7 && i<=8 && j>=7 && j<=8 ){
      pic[0][i][j]=matrix.Color(255,255,255);
    }
    else{
      pic[0][i][j]=matrix.Color(0,0,0);
    }
    }
}

for(int i=0;i<RWMax;i++){
  for(int j=0;j<RHMax;j++){

    if(i>=7 && i<=8 && j>=7 && j<=8 ){
      pic[1][i][j]=matrix.Color(0,0,0);
    }
    else if(i>=6 && i<=9 && j>=6 && j<=9 ){
      pic[1][i][j]=matrix.Color(255,255,255);
    }
    else{
      pic[1][i][j]=matrix.Color(0,0,0);
    }
    }
}

for(int i=0;i<RWMax;i++){
```

```
  for(int j=0;j<RHMax;j++){

    if(i>=6 && i<=9 && j>=6 && j<=9 ){
      pic[2][i][j]=matrix.Color(0,0,0);
    }
    else if(i>=5 && i<=10 && j>=5 && j<=10 ){
      pic[2][i][j]=matrix.Color(255,255,255);
    }
    else{
      pic[2][i][j]=matrix.Color(0,0,0);
    }
    }
}

for(int i=0;i<RWMax;i++){
  for(int j=0;j<RHMax;j++){

    if(i>=5 && i<=10 && j>=5 && j<=10 ){
      pic[3][i][j]=matrix.Color(0,0,0);
    }
    else if(i>=4 && i<=11 && j>=4 && j<=11 ){
      pic[3][i][j]=matrix.Color(255,255,255);
    }
    else{
      pic[3][i][j]=matrix.Color(0,0,0);
    }
    }
}

for(int i=0;i<RWMax;i++){
  for(int j=0;j<RHMax;j++){
```

```
      if(i>=4 && i<=11 && j>=4 && j<=11 ){
        pic[4][i][j]=matrix.Color(0,0,0);
      }
      else if(i>=3 && i<=12 && j>=3 && j<=12 ){
        pic[4][i][j]=matrix.Color(255,255,255);
      }
      else{
        pic[4][i][j]=matrix.Color(0,0,0);
      }
      }
}

for(int i=0;i<RWMax;i++){
   for(int j=0;j<RHMax;j++){

      if(i>=3 && i<=12 && j>=3 && j<=12 ){
        pic[5][i][j]=matrix.Color(0,0,0);
      }
      else if(i>=2 && i<=13 && j>=2 && j<=13 ){
        pic[5][i][j]=matrix.Color(255,255,255);
      }
      else{
        pic[5][i][j]=matrix.Color(0,0,0);
      }
      }
}

for(int i=0;i<RWMax;i++){
   for(int j=0;j<RHMax;j++){

      if(i>=2 && i<=13 && j>=2 && j<=13 ){
        pic[6][i][j]=matrix.Color(0,0,0);
```

```
        }
        else if(i>=1 && i<=14 && j>=1 && j<=14 ){
          pic[6][i][j]=matrix.Color(255,255,255);
        }
        else{
          pic[6][i][j]=matrix.Color(0,0,0);
        }
        }
    }

    for(int i=0;i<RWMax;i++){
      for(int j=0;j<RHMax;j++){

        if(i>=1 && i<=14 && j>=1 && j<=14 ){
          pic[7][i][j]=matrix.Color(0,0,0);
        }
        else if(i>=0 && i<=15 && j>=0 && j<=15 ){
          pic[7][i][j]=matrix.Color(255,255,255);
        }
        else{
          pic[7][i][j]=matrix.Color(0,0,0);
        }
      }
    }
      {
      Serial.println("Type key when ready...");
      while (!Serial.available()){}   // wait for a character
    }
}

boolean flag=true;
```

```
void loop(void)
{

 {
  matrix.fillScreen(0);

  for(int i=0;i<RWMax;i++){
      for(int j=0;j<RHMax;j++){
        uint16_t c=pic[t][i][j];
        matrix.drawPixel(i,j,c);
      }
  }
  matrix.show();

  if (t == 0) {
    hiSound();
  } else if (t == 7) {
    lowSound();
  }

  if(t==0){
    flag=true;
  }
    else if(t==7){
      flag=false;
    }
  if(flag==true){
    t++;
  }
    else{
      t--;
    }
```

```
  }
}

void hiSound()
{
  int i = 0;
  while(i<50) {
      digitalWrite(speaker, HIGH);
      delay(1);
      digitalWrite(speaker, LOW);
      delay(1);
      i = i + 1;
  }
}

void lowSound()
{
  int i = 0;
  while(i<50) {
      digitalWrite(speaker, HIGH);
      delay(10);
      digitalWrite(speaker, LOW);
      delay(10);
      i = i + 10;
  }
}
```