CO2 センサを用いた換気扇自動制御デバイス

第 3 回生活デザインコンテスト応募作品

作品 URL：http://idea.linkdata.org/idea/idea1s1923i

研究室 URL：http://smart-ilab.sakura.ne.jp/

メンバー：数野　翔太
武内　一晃

神奈川工科大学 工学研究科 電気電子工学専攻

目次

第 1 章　はじめに

1．1　開発背景

日常生活をオフィス空間や学校のような閉ざされた空間で過ごす場合、人間の呼吸により室内空気が汚染され人間の健康に被害を及ぼす。空間中の二酸化炭素濃度が上昇することで、濃度が 1000ppm 以上で眠気や集中力・思考力の低下、2000ppm 以上で頭痛や肩こりの症状が発生する。快適に生活を送るには室内空気を清潔に管理する必要である。空気を清潔にする方法として「換気」が効果的である。

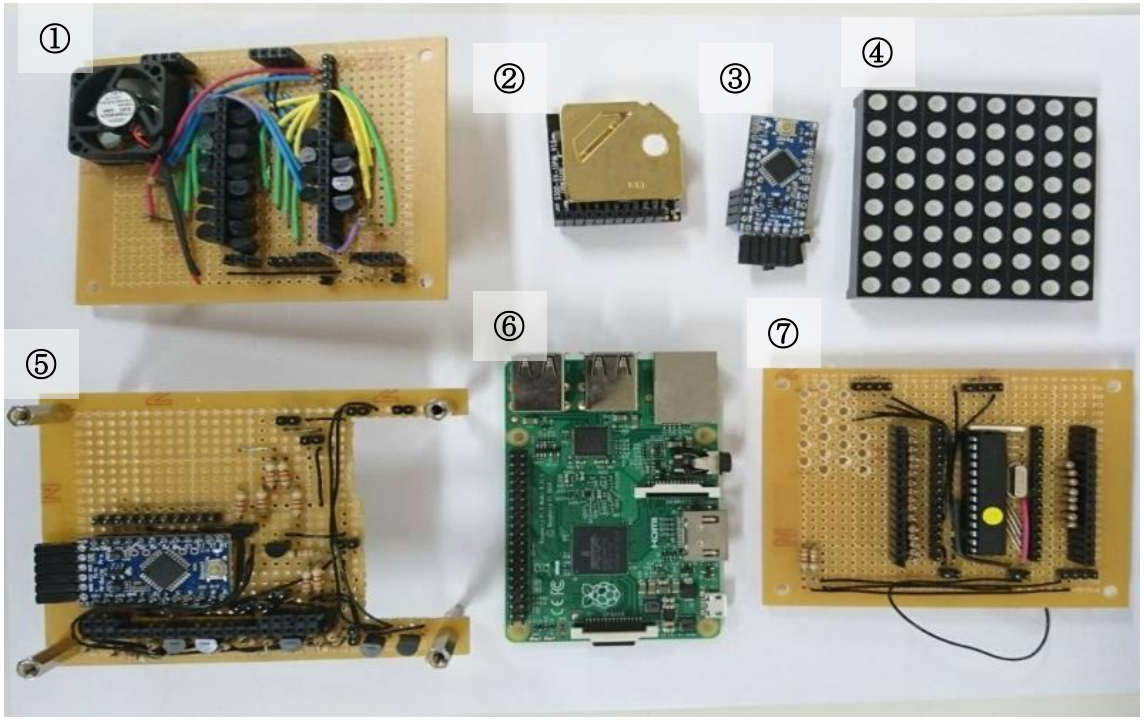
近年、スマートメータや HEMS という言葉をメディアで耳にすることが多くなった。スマートメータと HEMS を繋ぐ標準プロトコルとして、2011 年に ECHONET Lite が認定された[1]。ECHONET Lite の機器オブジェクトには空調関連機器クラスや住宅・設備関連機器クラス、調理・家事関連機器クラスがある[2]。その中でセンサ関連機器クラスを活用したデバイスは少ない。

1．2　目的

CO2 濃度が人間に与える問題を解決する、CO2 センサを利用した換気扇自動制御デバイス「VFcom」を開発する。「VFcom」は CO2 センサに応じて ECHONET Lite より換気扇の自動制御を行い、新鮮な空気を取り込むことで「豊かな」「楽しい」生活を実現するデバイスである。またインターフェイスに LED マトリクスを使用し、空気環境を可視化することで新しい生活をデザインする。

第 2 章　VFcom の概要

2. 1　構成部品



①ファン

②CO2 センサ（S-300）

③Arduino3 (Arduino pro mini)

④8×8 LED matrix (KRM2388AURPGB1)

⑤Arduino2 (Arduino pro mini)

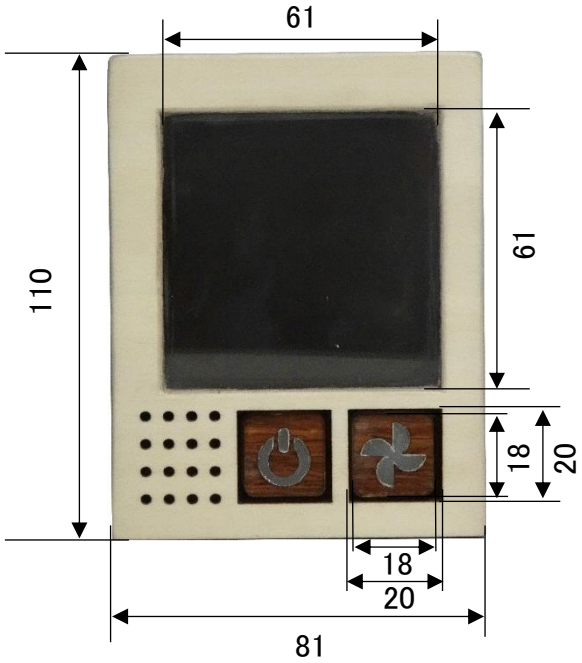
⑥Raspberry pi 3 Model B

⑦Arduino1 (ATMEGA328P-PU)

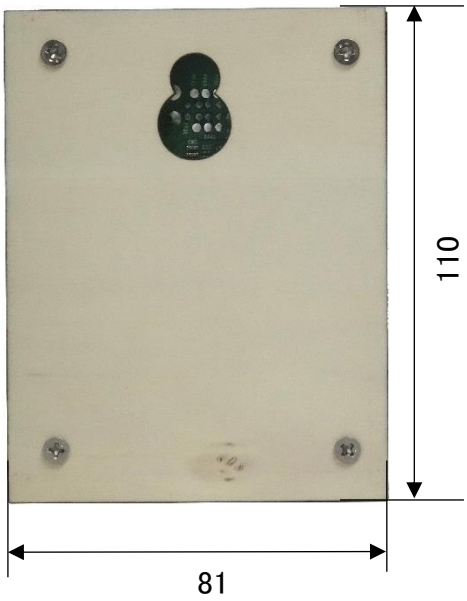
その他部品	個数
2SC1815	29
抵抗10kΩ	22
抵抗510Ω	8
抵抗100Ω	8
抵抗220Ω	8
抵抗1kΩ	2
ダクトスイッチ	2
LED	2
16MHz発振子	1

2. 2　VFcom の寸法

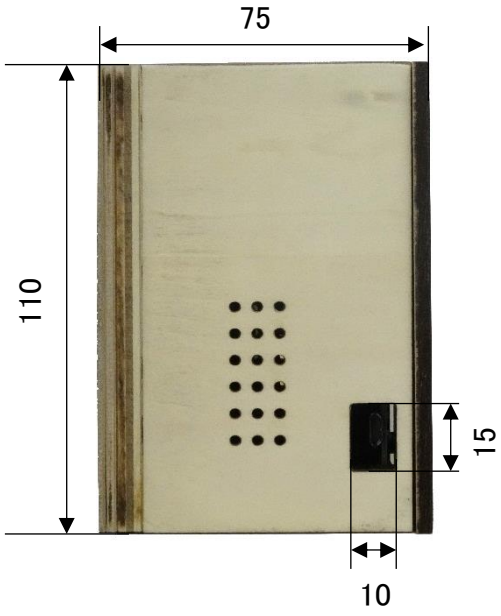
正面



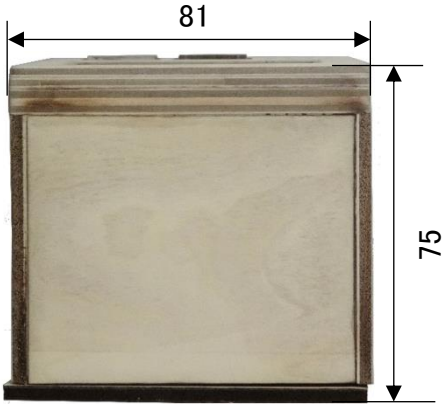
背面



側面

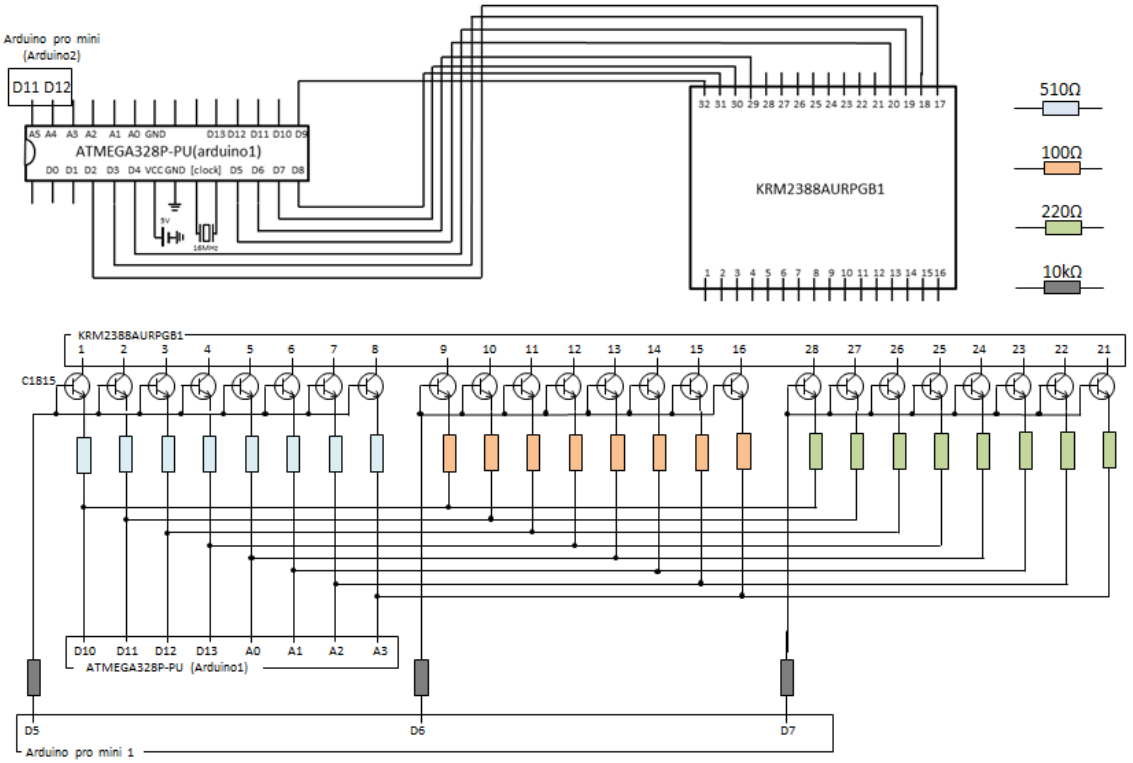


平面



2. 3　VFcom の回路図

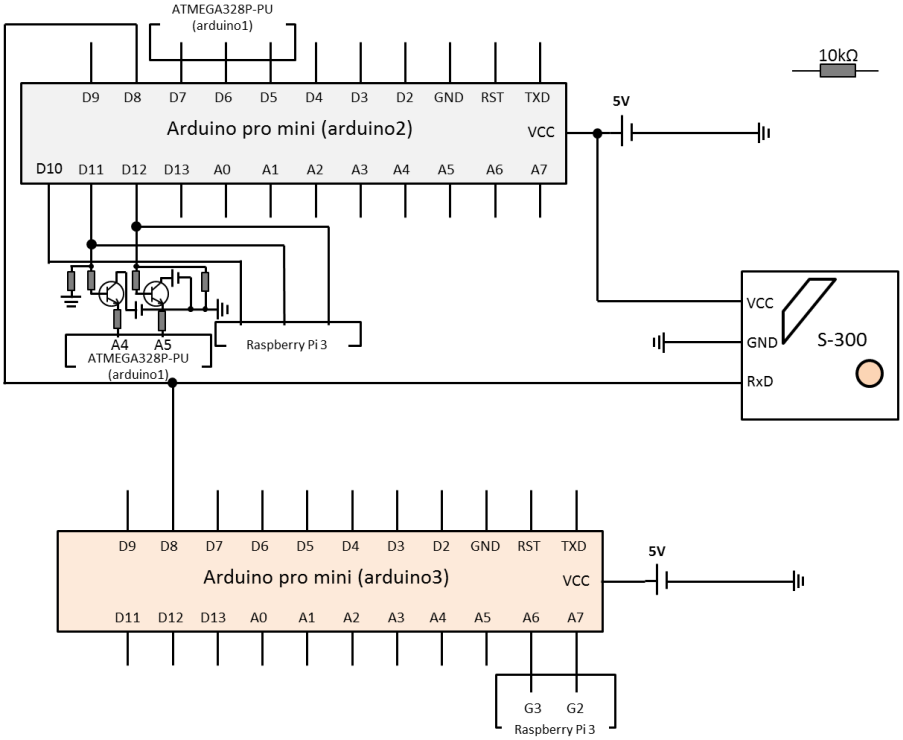
2.3.1　Arduino 1 の回路図



2SC1825 トランジスタを 8 個づつ各色ごとにコレクタを接続し、エミッタを Arduino1 に接続します。色ごとに使用したトランジスタのベースを接続することで、電圧を印可すると全体の色を変えることができますしかし、色の制御は Arduino1 ではピン数が足りないため、次の Arduino 2 のピンを使用します。Arduino1 の A4,A5 ピンはスイッチ制御に使用し、各信号どちらかが HIGH になるか両方とも LOW である時にアニメーションを切り替えます。

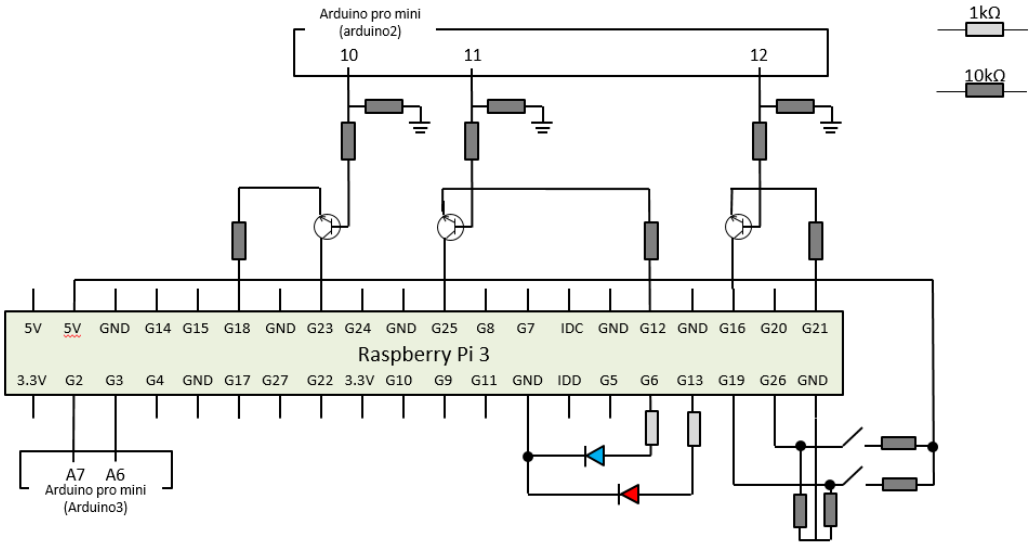
Arduino1, Arduino2, Arduino3, Raspberry Pi3 へのプログラムは以下のシステムの詳細で説明します。（プログラムは「http://smart-ilab.sakura.ne.jp/wp/wp-content/uploads/2016/07/VFcom1.zip」）にもあります。

2.3.2　Arduino2 Arduino3 の回路図



上図の水色のマイコンが Arduino pro mini(Arduino2)になります。ここでは、CO2 センサ（S-300）と UART 通信を行い空気中の CO2 濃度を取得します（ATMEGA328P-PU マイコンでは UART 通信がうまくいかなかったため)。このセンサ値から Arduino2 のピンを制御し、先に接続された Arduino1 の A4、A5 ピンのスイッチの ON,OFF を行い、さきほどのトランジスタのベースに接続することで LED マトリクスのカラー制御も同時に行います。

2.3.2　Raspberry pi の回路図



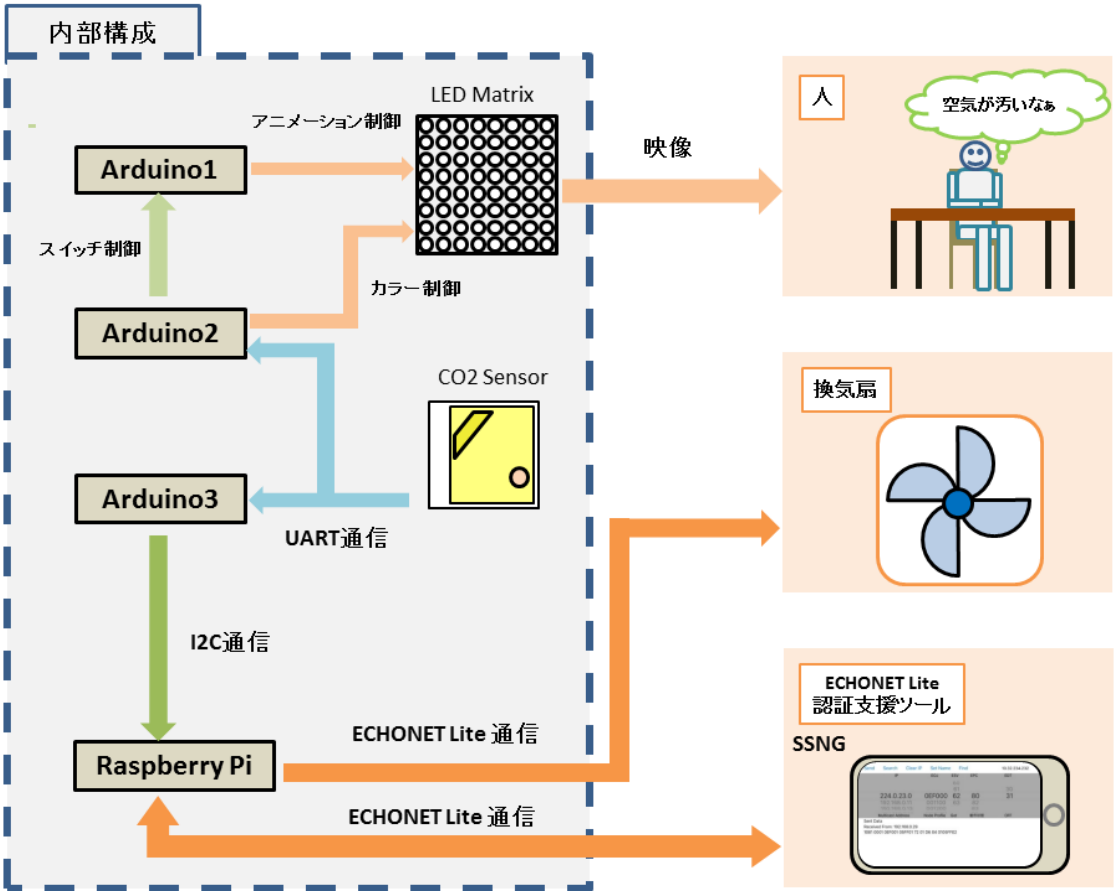
Raspberry Pi は Arduino2 と Arduino3 と接続します。以上で回路は完成です。

*完成後の写真等は「 http://smart-ilab.sakura.ne.jp/wordpress1/vfcom/」に載せてありますので詳細はこちらにアクセスしてください。

第 3 章　システムの詳細

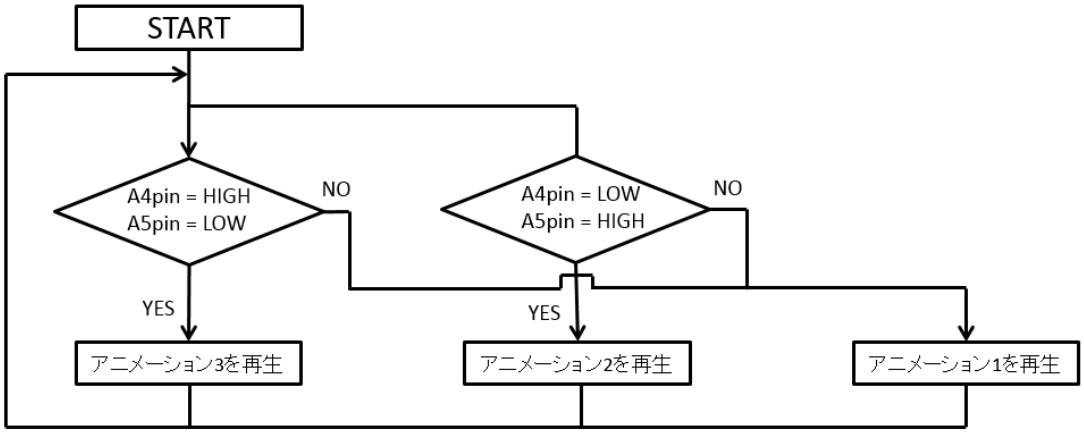
3. 1　システム構成

3.1.1　システム構成図

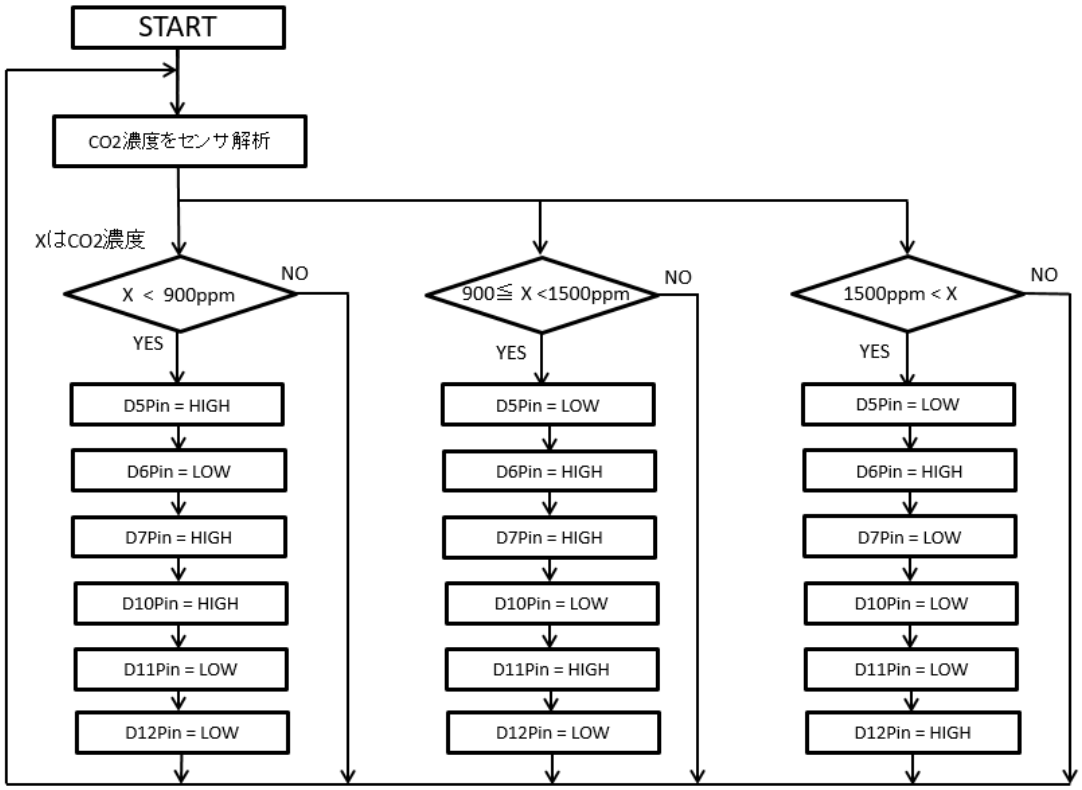


3.1.2　Arduino と Raspberry pi のプログラムのフローチャート

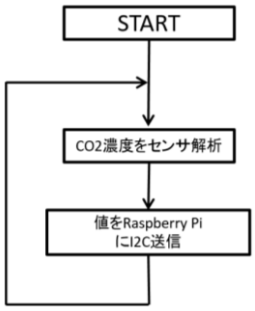
Arduino1



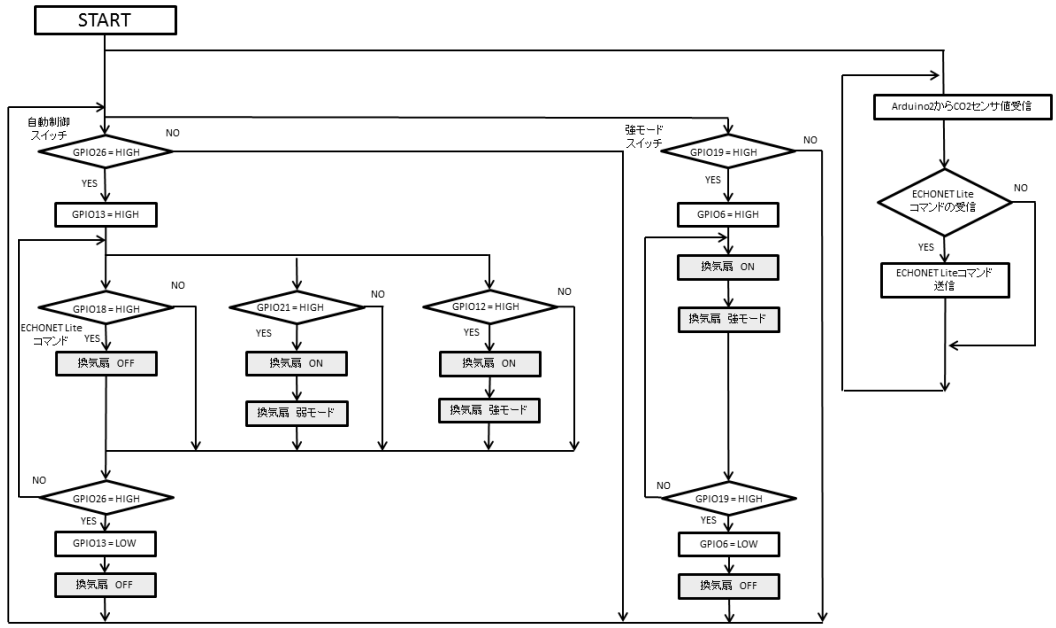
Arduino2



Arduino3



Raspberry Pi



3. 2 Arduino プログラム

3.2.1 8×8LED マトリクス点灯パターン(Arduino1)

```
boolean m1[8][8] = {
   {0, 0, 0, 0, 1, 0, 0, 0},
   {0, 0, 0, 0, 1, 0, 0, 0},
   {0, 0, 0, 1, 1, 0, 0, 0},
   {1, 1, 1, 1, 1, 1, 0, 0},
   {0, 0, 1, 1, 1, 1, 1, 1},
   {0, 0, 0, 1, 1, 0, 0, 0},
   {0, 0, 0, 1, 0, 0, 0, 0},
   {0, 0, 0, 1, 0, 0, 0, 0}
};

boolean m2[8][8] = {
   {0, 0, 0, 1, 0, 0, 0, 0},
   {0, 0, 0, 1, 0, 0, 0, 0},
   {0, 0, 0, 1, 1, 0, 0, 0},
   {0, 0, 1, 1, 1, 1, 1, 1},
   {1, 1, 1, 1, 1, 1, 0, 0},
   {0, 0, 0, 1, 1, 0, 0, 0},
   {0, 0, 0, 0, 1, 0, 0, 0},
   {0, 0, 0, 0, 1, 0, 0, 0}
};
boolean m3[8][8] = {
   {0, 0, 1, 1, 0, 0, 0, 0},
   {0, 0, 0, 1, 0, 0, 0, 0},
   {0, 0, 0, 1, 1, 0, 0, 1},
   {0, 0, 1, 1, 1, 1, 1, 1},
   {1, 1, 1, 1, 1, 1, 0, 0},
   {1, 0, 0, 1, 1, 0, 0, 0},
   {0, 0, 0, 0, 1, 0, 0, 0},
   {0, 0, 0, 0, 1, 1, 0, 0}
};

boolean m4[8][8] = {
   {0, 1, 1, 0, 0, 0, 0, 0},
   {0, 0, 1, 1, 0, 0, 0, 1},
   {0, 0, 0, 1, 1, 0, 1, 1},
   {0, 0, 1, 1, 1, 1, 1, 0},
   {0, 1, 1, 1, 1, 1, 0, 0},
   {1, 1, 0, 1, 1, 0, 0, 0},
   {1, 0, 0, 0, 1, 1, 0, 0},
   {0, 0, 0, 0, 0, 1, 1, 0}
};
boolean m5[8][8] = {
   {1, 0, 0, 0, 0, 0, 1, 1},
   {1, 1, 0, 0, 0, 1, 1, 0},
   {0, 1, 1, 1, 1, 1, 0, 0},
   {0, 0, 1, 1, 1, 1, 0, 0},
   {0, 0, 1, 1, 1, 1, 0, 0},
   {0, 0, 1, 1, 1, 1, 1, 0},
   {0, 1, 1, 0, 0, 0, 1, 1},
   {1, 1, 0, 0, 0, 0, 0, 1}
};
```

```
boolean m6[8][8] = {
   {0, 0, 0, 0, 0, 0, 1, 0},
   {1, 1, 0, 0, 0, 1, 1, 0},
   {0, 1, 1, 1, 1, 1, 0, 0},
   {0, 0, 1, 1, 1, 1, 0, 0},
   {0, 0, 1, 1, 1, 1, 0, 0},
   {0, 0, 1, 1, 1, 1, 1, 0},
   {0, 1, 1, 0, 0, 0, 1, 1},
   {0, 1, 0, 0, 0, 0, 0, 0}
};

boolean m7[8][8] = {
   {0, 0, 0, 0, 1, 1, 0, 0},
   {0, 0, 0, 0, 1, 0, 0, 0},
   {1, 0, 0, 1, 1, 0, 0, 0},
   {1, 1, 1, 1, 1, 1, 0, 0},
   {0, 0, 1, 1, 1, 1, 1, 1},
   {0, 0, 0, 1, 1, 0, 0, 1},
   {0, 0, 0, 1, 0, 0, 0, 0},
   {0, 0, 1, 1, 0, 0, 0, 0}
};

boolean m8[8][8] = {
   {0, 0, 0, 0, 0, 0, 1, 0},
   {0, 0, 1, 0, 0, 1, 1, 1},
   {0, 0, 0, 0, 1, 1, 1, 1},
   {0, 0, 0, 0, 0, 1, 1, 1},
   {0, 1, 0, 0, 0, 0, 1, 0},
   {1, 1, 1, 0, 0, 0, 0, 0},
   {0, 1, 0, 0, 1, 0, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 0}
};

boolean m9[8][8] = {
   {0, 0, 1, 0, 0, 0, 0, 0},
   {0, 1, 1, 1, 0, 0, 1, 0},
   {0, 0, 1, 0, 0, 1, 1, 1},
   {0, 0, 0, 0, 0, 0, 1, 0},
   {0, 0, 0, 0, 0, 0, 0, 0},
   {0, 1, 0, 0, 1, 0, 0, 0},
   {0, 0, 0, 1, 1, 1, 0, 0},
   {0, 0, 0, 0, 1, 0, 0, 0}
};

int huu;

void setup() {
   pinMode(1, OUTPUT);
   for (int i = 2; i <= 17; i++) {
      pinMode(i, OUTPUT);
      pinMode(0, OUTPUT);
      pinMode(A4,INPUT);
      pinMode(A5,INPUT);
      Serial.begin(9600) ;
   }
}

void loop() {
```

```
   if(analogRead(4)>=300){
huu=4;
}
else if(analogRead(5)>=300){
   huu=8;
}
else {
  for(int i=0; i<2; i++){
   for(int i=0; i<50; i++){
   display(m1);}
   for(int i=0; i<50; i++){
   display(m2);    }
   for (int i=0; i<1; i++){
  }}
for(int i=0; i<4; i++){
      for(int i=0; i<25; i++){
   display(m8);}
   for(int i=0; i<25; i++){
   display(m9);    }
   for (int i=0; i<1; i++){
  }}

  huu=0;

}

   for(int i=0; i<huu; i++){
   display(m1);}
   for(int i=0; i<huu; i++){
   display(m2);}
   for(int i=0; i<huu; i++){
   display(m3);}
   for(int i=0; i<huu; i++){
   display(m4);}
   for(int i=0; i<huu; i++){
   display(m5);}
   for(int i=0; i<huu; i++){
   display(m6);}
   for(int i=0; i<huu; i++){
   display(m7);}
   }

void display(boolean matrix[8][8]) {
   for (int y = 2; y <= 9; y++) {
      digitalWrite(y, HIGH);
      for (int x = 10; x <= 17; x++) {
         if (matrix[y - 2][x - 10] == 1) {
            digitalWrite(x, LOW);
         } else {
            digitalWrite(x, HIGH);
            }
      }
      delay(2);
      clearMatrix();
   }
}

void clearMatrix() {
   for (int y = 2; y <= 9; y++) {
```

```
    digitalWrite(y, LOW);
  }
  for (int x = 10; x <= 17; x++) {
    digitalWrite(x, HIGH);
  }
}
```

3.2.3　CO2 センサとカラー制御とアニメーション選択(Arduino2)

```
#include <SoftwareSerial.h>
#include<stdlib.h>
SoftwareSerial S300(8, 9);

void setup() {
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  Serial.begin(38400);
  S300.begin(38400);
}

void loop() {
  String hogehoge = S300.readStringUntil('\n');
  int i = hogehoge.toInt();
  delay(1500);
  Serial.println(i, DEC);
  if (i < 900) {
    digitalWrite(5, HIGH);
    digitalWrite(6, LOW);
    digitalWrite(7, HIGH);
    digitalWrite(10, HIGH);
    digitalWrite(11, LOW);
    digitalWrite(12, LOW);
    delay(1000);
  }

  else if (i > 1500) {
    digitalWrite(5, LOW);
    digitalWrite(6, HIGH);
    digitalWrite(7, LOW);
    digitalWrite(10, LOW);
    digitalWrite(11, LOW);
    digitalWrite(12, HIGH);
    delay(1000);
  }

  else  {
    digitalWrite(5, LOW);
    digitalWrite(6, HIGH);
    digitalWrite(7, HIGH);
    digitalWrite(10, LOW);
    digitalWrite(11, HIGH);
    digitalWrite(12, LOW);
    delay(1000);
```

```
    }
}
```

3.2.3　CO2 センサと I2C 通信 (Arduino3)[3]

```
#include <Wire.h>
#include <SoftwareSerial.h>
int SLAVE_ADDRESS = 0x04;
SoftwareSerial S300(8, 9);

void setup() {
    pinMode(11, OUTPUT);
    Serial.begin(38400);
    S300.begin(38400);
    Wire.begin(SLAVE_ADDRESS);
    Wire.onReceive(processMessage);
    Wire.onRequest(sendAnalogReading);
}

void loop() {
}
void processMessage(int n) {
    char ch = Wire.read();
    if (ch == '1') {
    }
}

void sendAnalogReading() {
    String hogehoge = S300.readStringUntil('\n');
    int reading = hogehoge.toInt();
    Wire.write(reading >> 4);
}
```

3. 3　Raspberry pi プログラム

3.3.1　自動制御スイッチと強モードスイッチ

```
import RPi.GPIO as GPIO
import time
import socket

host = "224.0.23.0"
port = 3610

EDT-ON = "1081000005ff010130016001800130"
on = EDT-ON.decode("hex")

EDT-OFF = "1081000005ff010130016001800131"
off = EDT-OFF.decode("hex")

EDT-weak = "1081000005ff010130016001A00132"
weak = EDT-weak.decode("hex")

EDT-strong = "1081000005ff010130016001A00134"
storong = EDT-strong.decode("hex")


GPIO.setmode(GPIO.BCM)

#自動制御モード
GPIO.setup(18, GPIO.IN)
GPIO.setup(21, GPIO.IN)
GPIO.setup(12, GPIO.IN)
#スイッチ
GPIO.setup(19, GPIO.IN)
GPIO.setup(26, GPIO.IN)

#自動制御モード
GPIO.setup(23, GPIO.OUT)
GPIO.setup(25, GPIO.OUT)
GPIO.setup(16, GPIO.OUT)
#スイッチ
GPIO.setup(6, GPIO.OUT)
GPIO.setup(13, GPIO.OUT)


try:
    while True:
        flag1 = 1
        flag2 = 1
        flag3 = 1
        if GPIO.input(26) == GPIO.HIGH:
            GPIO.output(13, GPIO.HIGH)
            GPIO.output(6, GPIO.LOW)
            time.sleep(1)

            while True:
                if GPIO.input(18) == GPIO.HIGH and flag1 == 1:
                    GPIO.output(23, GPIO.HIGH)
```

```
                sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
                sock.sendto(off, (host, port))
                time.sleep(1)
                flag1 = 0
                flag2 = 1
                flag3 = 1


            elif GPIO.input(21) == GPIO.HIGH and flag2 == 1:
                    GPIO.output(25, GPIO.HIGH)
                    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
                    sock.sendto(on, (host, port))
                    sock.sendto(weak, (host, port))
                    time.sleep(1)
                    flag1 = 1
                    flag2 = 0
                    flag3 = 1


            elif GPIO.input(12) == GPIO.HIGH and flag3 == 1:
                    GPIO.output(16, GPIO.HIGH)
                    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
                    sock.sendto(on, (host, port))
                    sock.sendto(strong, (host, port))
                    time.sleep(1)
                    flag1 = 1
                    flag2 = 1
                    flag3 = 0

            elif GPIO.input(26) == GPIO.HIGH:
                    GPIO.output(13, GPIO.LOW)
                    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
                    sock.sendto(off, (host, port))
                    time.sleep(1)
                    break

            else:
                    GPIO.output(23, GPIO.LOW)
                    GPIO.output(25, GPIO.LOW)
                    GPIO.output(16, GPIO.LOW)
                    time.sleep(1)


    elif GPIO.input(19) == GPIO.HIGH:
        GPIO.output(6, GPIO.HIGH)
        GPIO.output(13, GPIO.LOW)
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        sock.sendto(on, (host, port))
        sock.sendto(storong, (host, port))
        time.sleep(1)
        while True:
            if    GPIO.input(19) == GPIO.HIGH:
                    GPIO.output(6, GPIO.LOW)
                    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
                    sock.sendto(off, (host, port))
                    time.sleep(1)
                    break

            else:
```

```
                    GPIO.output(23, GPIO.LOW)
                    GPIO.output(25, GPIO.LOW)
                    GPIO.output(16, GPIO.LOW)
                    time.sleep(1)

        else:
            GPIO.output(6, GPIO.LOW)
            GPIO.output(13, GPIO.LOW)
            time.sleep(1)

except KeyboardInterrupt:
  GPIO.cleanup()
```

3.3.2　ECHONET Lite 化用 [4]

```
#main.py
import echonet_lite

node = echonet_lite.Node()
node.add_object(echonet_lite.GeneralLighting())
node.loop(debug=True)
```

```
#_init_.py
#echonet_lite フォルダ内にあるプログラム
import socket
import smbus

class Frame:
    ''' ECHONET Lite Frame'''

    ESV_STR = {
        0x60: 'SetI',    0x61: 'SetC',    0x62: 'Get',    0x63: 'INF_REQ',                                0x6E: 'SetGet',
        0x71: 'Set_Res',    0x72: 'Get_Res', 0x73: 'INF',    0x74: 'INFC', 0x7A: 'INFC_Res', 0x7E: 'SetGet_Res',
        0x50: 'SetI_SNA', 0x51: 'SetC_SNA', 0x52: 'Get_SNA', 0x53: 'INF_SNA',
        0x5E: 'SetGet_SNA'
    }

    def __init__(self, data):
        if type(data) == bytearray:
            self._decode(data)
        elif type(data) == list:
            if len(data) < 6:
                self.valid = False
                return
            self.protocol_type = 'ECHONET_Lite'
            self.format = '1'
            self.EHD1 = data[0]
            self.EHD2 = data[1]
            self.TID = data[2]
            self.SEOJ = data[3]
            self.DEOJ = data[4]
            self.ESV = data[5]
            self.properties = []
            self.valid = True
        else:
            self.valid = False

    def _decode(self, data):
        if len(data) < 12:
            self.valid = False
            return
        self._decode_header(data[0:4])
        self._decode_data(data[4:])
        self.valid = True

    def _decode_header(self, data):
        self.EHD1 = data[0]
        self.EHD2 = data[1]
        self.TID = data[2:4]
        if self.EHD1 == 0x10:
```

```
            self.protocol_type = 'ECHONET_Lite'
        elif self.EHD1 >= 0x80:
            self.protocol_type = 'ECHONET'
        else:
            self.protocol_type = 'UNKNOWN'
        if self.EHD2 == 0x81:
            self.format = '1'
        elif self.EHD2 == 0x82:
            self.format = '2'
        else:
            self.format = 'UNKNOWN'

    def _decode_data(self, data):
        self.SEOJ = data[0:3]
        self.DEOJ = data[3:6]
        self.ESV = data[6]
        num_of_properties = data[7] # OPC
        self.properties = []
        offset = 8
        for i in range(num_of_properties):
            prop = Property(data[offset:])
            self.properties.append(prop)
            offset += len(prop)

    @staticmethod
    def create_response(frame):
        if frame.ESV == 0x61: # SetC
            ESV = 0x71
        elif frame.ESV == 0x62: # Get
            ESV = 0x72
        else:
            return Frame()
        return Frame([frame.EHD1, frame.EHD2, frame.TID, frame.DEOJ, frame.SEOJ, ESV])

    def get_bytes(self):
        array = bytearray([self.EHD1, self.EHD2])
        array = array + self.TID + self.SEOJ + self.DEOJ
        array.append(self.ESV)
        array.append(len(self.properties)) # OPC
        for prop in self.properties:
            array = array + prop.get_bytes()
        return bytearray(array)

    def __str__(self):
        if not self.valid:
            return "echonet_lite.Frame(invalid)"
        return "echonet_lite.Frame(protocol_type={}, format={}, TID={}, SEOJ={}, DEOJ={}, ESV={},
OPC={})".format(
            self.protocol_type, self.format, repr(self.TID), repr(self.SEOJ), repr(self.DEOJ),
            Frame.ESV_STR[self.ESV] if self.ESV in Frame.ESV_STR else '0x{:x}'.format(self.ESV),
len(self.properties)
        )

class Property:
    '' ECHONET Property ''


    EPC_STR = {
        0x80: ",          0x81:",          0xE2:"
```

```
    }

    def __init__(self, data):
        if type(data) == bytearray:
            self.EPC = data[0]
            len_edt = data[1]
            self.EDT = data[1:1+len_edt]
        elif type(data) == list:
            self.EPC = data[0]
            self.EDT = data[1]

    def get_bytes(self):
        array = bytearray([self.EPC, len(self.EDT)])
        array = array + self.EDT
        return array

    def __len__(self):
        return 2 + len(self.EDT)

    def __str__(self):
        return 'echonet_lite.Property(EPC=0x{:x}, PDC={}, EDT={})'.format(self.EPC, len(self.EDT), repr(self.EDT))

class Object:
    ''' ECHONET Object '''

    def __init__(self, group, cls):
        self.group = group
        self.cls = cls
        self.id = None
        self.EOJ = None

    def set_instance_id(self, id):
        self.id = id
        self.EOJ = bytearray([self.group, self.cls, self.id])

    def service(self):
        pass

class GeneralLighting(Object):
    ''' General Lighting Object (group=0x02, class=0x90) '''

    def __init__(self):
        Object.__init__(self, 0x02, 0x90)

    def service(self, frame):
        if frame.ESV == 0x61: # SetC
            new_frame = Frame.create_response(frame)
            for prop in frame.properties:
                if prop.EPC == 0x80: # power (0x30=ON, 0x31=OFF)
                    new_frame.properties.append(prop)
            return new_frame

class Node:
    ''' ECHONET Lite Node '''

    def __init__(self):
        self.objects = {}

    def add_object(self, obj):
```

```
    if obj.group not in self.objects:
        self.objects[obj.group] = {}
    if obj.cls not in self.objects[obj.group]:
        self.objects[obj.group][obj.cls] = []
    my_class = self.objects[obj.group][obj.cls]
    my_class.append(obj)
    obj.set_instance_id(len(my_class))

def _deliver(self, frame):
    if frame.DEOJ == bytearray(b'\x0e\xf0\01'):
        return self.service(frame)
    group = frame.DEOJ[0]
    cls = frame.DEOJ[1]
    id = frame.DEOJ[2]
    if group in self.objects and cls in self.objects[group]:
        if id == 0:
            return # TODO: broadcast to this class
        if id <= len(self.objects[group][cls]):
            return self.objects[group][cls][id-1].service(frame)

def service(self, frame):
    if frame.ESV == 0x62: # Get
        new_frame = Frame.create_response(frame)
        for prop in frame.properties:
            if prop.EPC == 0xd6: # instance list
                new_frame.properties.append(self._create_object_list_property())
        return new_frame

def _create_object_list_property(self):
    array = [0]
    for group in self.objects:
        for cls in self.objects[group]:
            for id in range(1, len(self.objects[group][cls])+1):
                array += [group, cls, id]
                array[0] += 1
    return Property([0xd6, bytearray(array)])

def _bind_socket(self):
    local_address = '0.0.0.0'
    multicast_group = '224.0.23.0'
    port = 3610
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind((local_address, port))
    sock.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP,
                            socket.inet_aton(multicast_group) + socket.inet_aton(local_address))
    return sock

def loop(self, debug=False):
    sock = self._bind_socket()
    print "wait..."
    while True:
        recv_msg, addr = sock.recvfrom(1024)
        if debug:
            print addr
        frame = Frame(bytearray(recv_msg))
        if debug:
            print_frame(frame)
        new_frame = self._deliver(frame)
```

```
#0x80
        if frame.get_bytes() == bytearray(b'\x10\x81\x00\x00\x05\xff\x01\x00\x1b\x01b\x01\x80\x00'):
            multicast_group = '224.0.23.0'
            port = 3610
            a = "1081000005ff01001B017201800130"
            z = a.decode("hex")
            sock.sendto(z, (multicast_group, port))
#0x81
        elif frame.get_bytes() == bytearray(b'\x10\x81\x00\x00\x05\xff\x01\x00\x1b\x01b\x01\x81\x00'):
            multicast_group = '224.0.23.0'
            port = 3610
            b = "1081000005ff01001B017201800100"
            y = b.decode("hex")
            sock.sendto(y, (multicast_group, port))
#0x82
        elif frame.get_bytes() == bytearray(b'\x10\x81\x00\x00\x05\xff\x01\x00\x1b\x01b\x01\x82\x00'):
            multicast_group = '224.0.23.0'
            port = 3610
            c = "1081000005ff01001B017201800400004800"
            x = c.decode("hex")
            sock.sendto(x, (multicast_group, port))
#0x83
        Elif frame.get_bytes() == bytearray(b'\x10\x81\x00\x00\x05\xff\x01\x00\x1b\x01b\x01\x83\x00'):
            multicast_group = '224.0.23.0'
            port = 3610
            d = "1081000005ff01001B017201800100"
            w = d.decode("hex")
            sock.sendto(w, (multicast_group, port))
#0x8E
        elif frame.get_bytes() == bytearray(b'\x10\x81\x00\x00\x05\xff\x01\x00\x1b\x01b\x01\x8E\x00'):
            multicast_group = '224.0.23.0'
            port = 3610
            e = "1081000005ff01001B017201800407E0060f"
            v = e.decode("hex")
            sock.sendto(v, (multicast_group, port))
#0xE0
        elif frame.get_bytes() == bytearray(b'\x10\x81\x00\x00\x05\xff\x01\x00\x1b\x01b\x01\xE0\x00'):
            multicast_group = '224.0.23.0'
            port = 3610
            bus = smbus.SMBus(1)
            SLAVE_ADDRESS = 0x04
            reading = int(bus.read_byte(SLAVE_ADDRESS))

            CO2 = format((reading)*16,'x')

            f = '1081000005ff01001B01720180020'+str(CO2)+"
            u = f.decode("hex")
            sock.sendto(u, (multicast_group, port))

#node-profile
#0x80
        elif frame.get_bytes() == bytearray(b'\x10\x81\x00\x00\x05\xff\x01\x0e\xf0\x01b\x01\x80\x00'):
            multicast_group = '224.0.23.0'
            port = 3610
            a = "1081000005ff01001B017201800130"
            z = a.decode("hex")
            sock.sendto(z, (multicast_group, port))
#0x81
        elif frame.get_bytes() == bytearray(b'\x10\x81\x00\x00\x05\xff\x01\x0e\xf0\x01b\x01\x81\x00'):
```

```
            multicast_group = '224.0.23.0'
            port = 3610
            b = "1081000005ff01001B017201800100"
            y = b.decode("hex")
            sock.sendto(y, (multicast_group, port))
#0x82
        elif frame.get_bytes() == bytearray(b'\x10\x81\x00\x00\x05\xff\x01\x0e\xf0\x01b\x01\x82\x00'):
            multicast_group = '224.0.23.0'
            port = 3610
            c = "1081000005ff01001B017201800400004800"
            x = c.decode("hex")
            sock.sendto(x, (multicast_group, port))
#0x83
        elif frame.get_bytes() == bytearray(b'\x10\x81\x00\x00\x05\xff\x01\x0e\xf0\x01b\x01\x83\x00'):
            multicast_group = '224.0.23.0'
            port = 3610
            d = "1081000005ff01001B017201800100"
            w = d.decode("hex")
            sock.sendto(w, (multicast_group, port))
#0x8E
        elif frame.get_bytes() == bytearray(b'\x10\x81\x00\x00\x05\xff\x01\x0e\xf0\x01b\x01\x8E\x00'):
            multicast_group = '224.0.23.0'
            port = 3610
            e = "1081000005ff01001B017201800407E0060f"
            v = e.decode("hex")
            sock.sendto(v, (multicast_group, port))
#0xE0
        elif frame.get_bytes() == bytearray(b'\x10\x81\x00\x00\x05\xff\x01\x0e\xf0\x01b\x01\x81\x00'):
            multicast_group = '224.0.23.0'
            port = 3610

        else:
            pass


        if new_frame is not None:
            if debug:
                print_frame(new_frame)
            sock.sendto(new_frame.get_bytes(), addr)

def print_frame(frame):
    print frame
    for prop in frame.properties:
        print prop
    print repr(frame.get_bytes())
```

参考文献

[1] ECHONET Lite – Wikipedia
https://ja.wikipedia.org/wiki/ECHONET_Lite

[2] ECHONET Lite 規格 - エコーネットコンソーシアム APPENDIX ECHONET 機器オブジェクト詳細規定 Release H のダウンロード ファイルリスト
https://echonet.jp/spec_object_rh/

[3] Raspberry Pi と Arduino を I2C で接続する ,
http://robocad.blog.jp/archives/723986.html

[4] kminami/python-echonet-lite – GitHub
https://github.com/kminami/python-echonet-lite

[5]第 3 回生活デザインコンテスト,
http://lifedesign.tech/lifedesigncontest_3/

[6] VFcom 紹介動画
https://youtu.be/_gnJ7sGup-I

[7] VFcom 制作方法
http://smart-ilab.sakura.ne.jp/wordpress1/vfcom/