



第10章 LEDの波長確認や炎色波長の計測に！ 美味しい焼き加減やマスクの殺菌効果もわかる？

感度波長340 nm～850 nmの ハンディ光スペクトラム・アナライザ

C12880MA(浜松ホトニクス)は小型分光器で、340 nmから850 nmまでの波長の光を分光できます。本分光器とM5Stackを利用すると、単一電源で動作し、パッケージ・サイズもコンパクトで持ち運びが容易な光スペクトラム・アナライザを作ることができます。

写真1に示すのは、M5Stackに接続したC12880MAです。1眼レフ・カメラ用のマウントを利用してC12880MAの基板と、M5Stackを固定し、測定位置や角度を容易に決められるようにしています。1眼レフ・カメラ用の三脚固定用のパーツは光学測定用の微動機構に比べて2～3万円と価格が安く、こういった用途で位置決めするときに便利です。写真2は、本器で炎色反応のスペクトラムを測定しているところです。

スペック

- 感度波長範囲：340 nm～850 nm,
波長分解能：12 nm (typ)
- 総画素数：288 ピクセル

用途

- LEDの波長確認
- 炎色反応の波長計測
- カラー・バランスの確認

分光器C12880MAの特徴

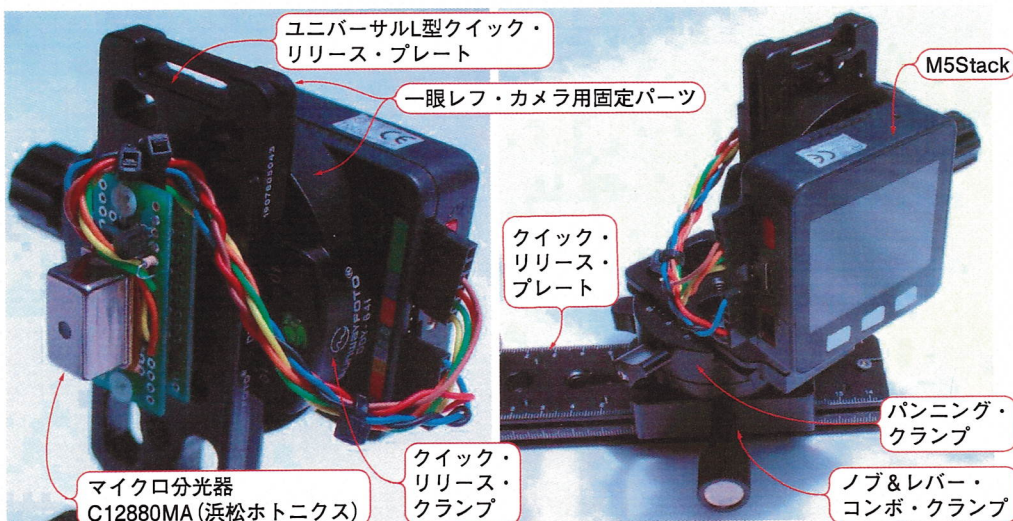
- サイズ：20.1×12.5×10.1 mm
- 質量：5 g, 電源電圧：5 V
- 制御が容易：クロックとスタート・パルス
- アナログ出力：0.8～4.3 V



写真2 油が落ちて炎があがったときのスペクトラムを測定しているところ

写真1 本章ではM5Stackと小型分光器C12880MA(浜松ホトニクス)を利用して光スペクトラム・アナライザを製作する

カメラ用の三脚は450G-7(SLIK), ユニバーサルL型クイック・リリース・プレートはDPL-06R(SUNWAYFOTO), パニング・クランプはDDH-05N(SUNWAYFOTO), クイック・リリース・クランプはDDH-05N(SUNWAYFOTO), クイック・リリース・プレートはBPL-300L(MENGES, 販売中止品。PU200などで置き換え可能), ノブ&レバー・コンボ・クランプはDLC-60LX(SUNWAYFOTO), M5Stackの固定はDDY-64i(SUNWAYFOTO)を使った



(a) M5Stackと分光器の取り付け部

(b) 一眼レフカメラ用の固定パーツで測定位置と角度を調整できるようにしている

ハードウェア製作

● キー・デバイス「小型分光器 C12880MA」のメカニズム

▶ 構造

図1に示すのは、使用したC12880MAの内部構造です。スリットを通過した光をブレード・グレーティングと呼ばれるのこぎりの刃の形状の回折格子で分光し、1次元のCMOSイメージ・センサで電気信号に変換しています。回折格子とCMOSイメージ・センサの位置関係は出荷検査で個々に測定され、C12880MAを購入すると波長変換係数が検査成績書に添付されています。

波長変換係数は、6個の定数でCMOSイメージ・センサの画素に対応する波長を計算する5次式の係数に使用します。

▶ 動作タイミング

図2に示すのは、C12880MAの動作タイミングです。制御入力はクロックとストローブ信号で、センサからの出力はアナログ・ビデオ信号とアナログ出力のトリガ信号とセンサ・スキャン終了信号です。トリガ出力の立ち上がりがアナログ出力のサンプリング・タイミングです。今回紹介するソフトウェアでは、トリガ信号とセンサ・スキャン終了信号は使っていません。

アナログ出力のトリガ信号は、クロック入力を反転したタイミングに近い位相で出力されていたため、A-Dコンバータのサンプリング・タイミングはクロックの立ち下りのタイミングにしています。

CMOSイメージ・センサの画素数は288画素です。ビデオ出力はスタート信号の立ち下りから88クロックで開始されます。CMOSイメージ・センサの感度調整は蓄積時間で制御します。蓄積時間はスタート・パルス幅+48クロック分の時間になります。ソ

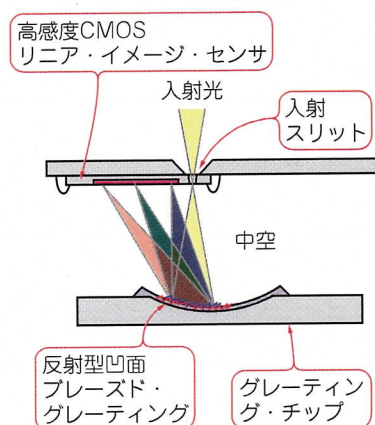


図1 C12880MAの構造

スリットを通過した光を凹型のこぎり刃状（ブレード）の回折格子で分光し、1次元のイメージ・センサで電気信号に変換する

フトウェアではクロックの周波数を制御することで、センサの感度を調整しています。

カタログではクロックのデューティ比を45%から55%と規定していますが、制御ソフトウェアでは、出力ポートから簡易的に1/0を出力してクロックにしているため、デューティ・サイクルが50%からかなりはずれることがあるのですが動作しています。製品を開発する場合は、推奨動作タイミングを守る必要があります。

● 制御回路

図3に示すのは、外付け回路とM5Stackへの接続です。C12880MAは5V電源で動作させるため、M5Stack側のロジック入力とA-Dコンバータで受けるC12880MAのアナログ・ビデオ出力でレベル変換が必要になります。5Vロジックは簡易的に1kΩと2kΩの分圧回路で3.3Vロジックに変換しています。M5Stackのロジック制御出力は3.3Vのままで制御できます。本器のソフトウェアでは、センサからのTRGとEOSのロジック出力は使っていないので、抵抗分圧器と配線はなくても動作します。

また、センサのロジック出力の負荷を軽くすることで、イメージ・センサ周辺の回路の発熱を減らすことができます。

● ビデオ出力バッファ

C12880MAのビデオ出力にはセンサ内部の発熱を抑えるため、バッファ・アンプを取り付けることが推奨されています。ビデオ出力の振幅は最大4.5V程度になるので、M5StackのA-Dコンバータの入力範囲にあわせて分圧する必要があります。ロジック出力と同じく抵抗による分圧でも動作すると思いますが、トランジスタのエミッタ・フォロワによるバッファ・アンプと分圧を兼ねた回路にしました。エミッタ・フォ

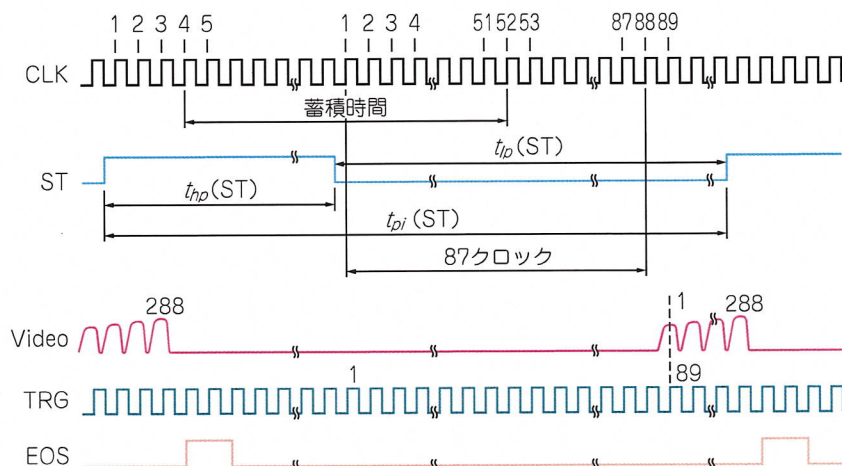
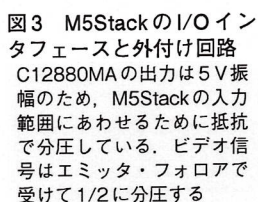
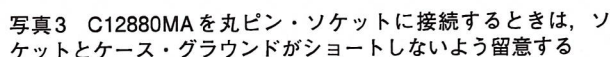


図2 C12880MAの動作タイミング



ビデオ信号出力の最大電圧を4.5 Vとすると、トランジスタのベース-エミッタ間電圧降下0.7 Vで3.8 Vになり、100 Ω 2個の分圧回路で1.9 V振幅がM5StackのA-Dコンバータの入力になります。

写真3に示すのは、M5StackとC12880MAを接続



C12880MAの基板は1眼レフ・カメラ用に使っているクイック・リリース・プレートDPL-06R(SUNWAYFOTO社)にM3ネジとワッシャで固定しています。M5Stackは同じくクイック・リリース・クランプのDDY-64iに両面テープで張り付けています。回転角の調整や位置決め固定にはパニング・クランプのDDH-05Nやスライド・プレートのDMC-200Lを利用しています。

リスト1の10行目から15行目が今回使用したC12880MAの波長変換係数です。この値はC12880MAを購入したときに添付される検査成績書のC12880MAのシリアル・ナンバに対応する数値を入力してください。

リスト1 ヘッダ・ファイルや定数の宣言

```

1 //
2 // C12880MA HAMAMATSU PHOTONICS Micro
   Spectrometer C12880MA SN17100439
3 //
4 #include <M5Stack.h>
5 #include <driver/adc.h>
6 #include <esp_adc_cal.h>
7 #include <math.h>
8
9 // 波長補正係数 A0+B1*pix+B2*pix^2+B3*pix^3+B4*
   pix^4+B5*pix^5
10 const double pA0 = 3.118600107e2;
11 const double pB1 = 2.701222708;
12 const double pB2 = -1.427535306e-3;
13 const double pB3 = -5.295943464e-6;
14 const double pB4 = 4.081650994e-10;
15 const double pB5 = 1.723681828e-11;
16
17 bool clear = 1; //Screen clear flag
18 bool test = 0; //Acquisition flag
19 int Plot_switch = 0; //Line color
20 const int COLOR_SELECT = 4;
21 const unsigned short PLOT_COLORS[COLOR_
   SELECT] = {TFT_WHITE, TFT_GREEN, TFT_RED, TFT_
   YELLOW};
22 int gain = 0; //clock_width control
23 const int CLOCK_SELECT = 6;
24 const int CLOCK_WIDTH[CLOCK_SELECT] =
   {1, 5, 10, 50, 100, 500};

```

● グラフ画面描画

リスト2に示すのは、M5Stackの画面描画に関する機能をまとめたクラスです。

27行目から29行目はM5Stackの画面に描くグラフの位置を表す構造体の定義です。画面上のゼロ点位置とグラフの縦横サイズとメモリの間隔を設定しています。

メンバ関数は3個、init、line、plot_valueで初期化、グラフの線の描画、画面上に文字の表示です。

初期化(init)では画面のxy軸と目盛りを描画します。y軸はA-Dコンバータの出力電圧を表示し、x軸は光の波長を表示しています。光の波長は55行目の計算式でピクセル値から波長に変換しています。そのため、メモリに表示している光の波長の値はきりの良くない数字になっています。

55行目の補正係数式の逆関数を用意してグラフのメモリにはきりの良い数値を表示するべきなのですが、その処理を記述できませんでした。

▶ 線分描画

60行目から68行目が線分描画関数です。入力はxの値、yの値、描画開始フラグ、線の色です。

入力を画面上のピクセル位置に変換して、描画開始フラグが1の場合は描画点の座標を記憶するだけで終了し、次の点から前回の点との間に線分を指定された色で描画します。

▶ 数値表示

69行目から72行目が数値表示関数です。指定され

た画面位置に文字列と数値を表示します。ゲイン切り替え時にゲインの数値を画面右上に表示するために使っています。

● Setup関数とLoop関数

リスト3の75行目以降がArduinoのSetup関数とloop関数です。setup関数ではM5.beginと入出力ポートの定義をしています。loop関数では、M5Stackのフロント・パネルの3つのボタンに対する動作を記述しています。右ボタンを押すとClearフラグを1にして90行目からの96行目の画面クリア動作を実行します。

84行目では中央ボタンが押されたときにtestフラグを1にします。85行目から89行目は左ボタンが押されたときのゲイン切り替え処理で86行目でゲインの設定数値をインクリメントして、87行目でゲインの設定種類の上限に達したらゼロに戻す処理をして、88行目で画面右上にゲインの設定値を表示しています。

▶ 計測処理

リスト3の97行目からが計測処理です。C12880MAの画素数、ストロブ信号のパルス幅、ビデオ出力開始のタイミングなどを設定し103行目からのループでC12880MAのクロックとストロブ信号を供給します。クロックのHigh期間とクロックの立ち下がりは105行目から107行目、クロックのLow期間は112行目から114行目のループで出力しています。ストロブ信号のパルスは110行目で生成して111行目で出力しています。クロック幅はリスト1の24行目で設定している定数配列で定義しています。C12880MAのビデオ信号はクロックの立ち下がりタイミングの109行目のA-Dコンバータ入力関数で受けています。

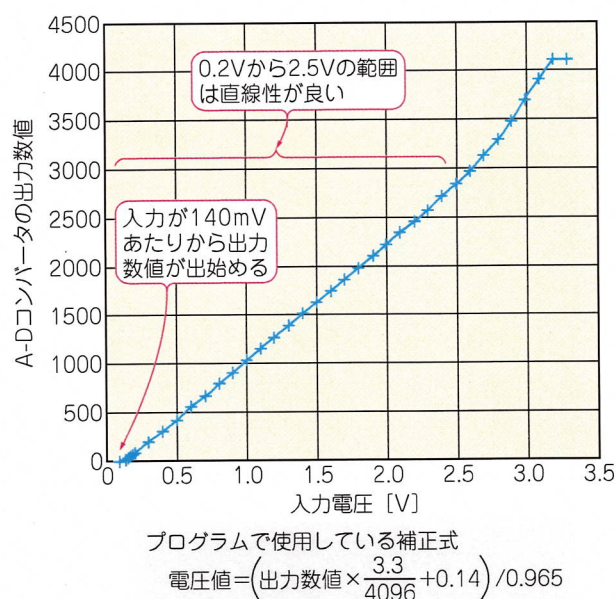


図4 M5Stack内蔵のA-Dコンバータの特性

リスト2 画面描画クラスの定義

```

26 struct { // Screen public class
27     typedef struct { unsigned short ox, oy, wx, hy, tx, ty; } Axis;
28     const Axis Plot={40,10,280,200,20,20}; // Screen size 320*240
29     short x,y,px,py;
30     float X_Div,V_Div;
31
32 void init(float X_Div_i,float V_Div_i){ //画面初期化
33     X_Div = X_Div_i; //
34     V_Div = V_Div_i;
35     M5.Lcd.fillScreen(TFT_BLACK);
36     M5.Lcd.fillRect(Plot.ox,Plot.oy,Plot.wx,Plot.hy,TFT_BLUE); // Screen background color
37     for(int i=Plot.ox;i <= (Plot.ox+Plot.wx);i+=Plot.tx){ //Y軸グリッド
38         M5.Lcd.drawLine(i,Plot.oy,i,Plot.oy+Plot.hy,TFT_BLACK);
39     }
40     for(int i=Plot.oy;i <= (Plot.oy+Plot.hy);i+=Plot.ty){ //X軸グリッド
41         M5.Lcd.drawLine(Plot.ox,i,Plot.ox+Plot.wx,i,TFT_BLACK);
42     }
43     M5.Lcd.setCursor(1,Plot.hy/2); //Y軸単位文字表示位置調整
44     M5.Lcd.setTextSize(2); //文字サイズ
45     M5.Lcd.printf("V"); // V(電圧)
46     M5.Lcd.setCursor(Plot.wx/2+20,Plot.hy+23); //X軸単位文字表示位置調整
47     M5.Lcd.printf("nm");
48     M5.Lcd.setTextSize(1);
49     for(int yaxis = 0; yaxis <= Plot.hy/Plot.ty;yaxis+=2){ //Y軸目盛
50         M5.Lcd.setCursor(10,Plot.hy+5-yaxis*Plot.ty);
51         M5.Lcd.printf("%2.1f",yaxis*V_Div);
52     }
53     for(int pix_count=0;pix_count < Plot.wx;pix_count+=(Plot.tx*2)){ //X軸目盛
54         float pix = (float)pix_count; //ピクセルと光の波長の関係の計算
55         float wavelength = pA0+ pB1*pix+pB2*pow(pix,2)+pB3*pow(pix,3)+pB4*pow(pix,4)+pB5*pow(pix,5);
56         M5.Lcd.setCursor(Plot.ox + pix_count-10,Plot.hy+15); // -8 X offset, 15: Y offset
57         M5.Lcd.printf("%d", (int)(wavelength+0.5)); //波長描画
58     }
59 } //init
60 void line(float h,float v,bool start_flag,unsigned short color){ //線分描画
61     y = Plot.oy+Plot.hy-(short)(v * (float)Plot.ty /V_Div);
62     x = Plot.ox+(short)(h * (float)Plot.tx /X_Div);
63     if(!start_flag & (y > 0)){
64         M5.Lcd.drawLine(px,py,x,y,color);
65         M5.Lcd.drawLine(px,py+1,x,y+1,color);
66     }
67     px=x;py=y;
68 } //line
69 void plot_value(int x_i,int y_i, char *p,int in1){ //文字列と数値表示
70     M5.Lcd.setCursor(x_i,y_i);
71     M5.Lcd.printf("%s %d¥n",p,in1);
72 }
73 } Screen;

```

▶ 計測結果の表示と M5Stack 内蔵 A-D コンバータの リニアリティ補正

リスト3の115行目のif文でクロック・サイクルが
ストロブ幅+ビデオ出力のディレイ値を超えたら
CMOSイメージ・センサの画素数分のデータでグラ
フを描画します。

図4に示すのは、使用したM5Stack BASICのA-D
コンバータの入力電圧に対する出力数値です。入力電
圧が0.14 Vまでは出力が0付近で変化がなく、0.14 V
以上で数値が変化し始めます。入力電圧が2.5 V程度
までは比較的直線性がよく、3 Vから3.2 Vにかけて
特性が曲がり3.2 V以上(4095)で値が飽和します。

この測定から、ソフトウェアでは出力数値に3.3/4096
をかけた値に0.14 Vのオフセットを加えて、ゲイン補
正で0.965で割っています。ソフトウェアではリスト

3の117行目で補正しています。浮動小数点の掛け算
や割り算はソフトウェア的な負荷が高いため、今回の
用途のように光スペクトラムのy軸の光パワーを校正
しない用途では、計算しなくてもよいのですが、参考
のためにy軸を電圧として表示するようにしました。

補正した電圧値とCMOSイメージ・センサのピク
セル・カウントの値を用いて、119行目でグラフを描
画しています。

測定してみる

● ネオン・ランプや紫外線LEDの光スペクトラムを
みてみる

写真4はネオン・ランプの発光スペクトラムを測定
したところです。M5Stackの中央ボタンを押すと測

リスト3 setup関数とloop関数

```

74 //-----
75 void setup() {
76   M5.begin();
77   pinMode(16, OUTPUT); //R2 clock出力
78   pinMode(17, OUTPUT); //T2 strobe出力
79   pinMode(2, INPUT); //G2 EOS入力(未使用)
80   pinMode(5, INPUT); //G5 TRG入力(未使用)
81 }
82 void loop() {
83   if (M5.BtnC.wasPressed()) { clear = 1; } //右ボタンで画面クリア
84   if (M5.BtnB.wasPressed()) { test = 1; } //中央ボタンで測定
85   if (M5.BtnA.wasPressed()) {
86     gain++;
87     if (gain >= CLOCK_SELECT) gain = 0; //左ボタンで測定ゲイン切り替え
88     Screen.plot_value(250, 30, "Gain", gain);
89   }
90   if (clear) { // 画面クリア
91     Screen.init(20, 0, 0.2); //X:20/div Y:0.2V/div
92     M5.Lcd.setCursor(0, 0);
93     M5.Lcd.setTextSize(1);
94     Screen.plot_value(250, 30, "Gain", gain);
95     clear = 0;
96   }
97   if (test) { //測定
98     const int Pix_total = 288; //C12880MAの画素数
99     const int Strobe_width = 6; //strobe信号の幅
100    const int Video_out_delay = 87; //ビデオ出力開始クロック数
101    bool start_flag = 1; //線分描画用の始点フラグ
102    int pix_count = 0; //ピクセルカウンタ
103    for (int ck = 0; ck < (Pix_total + Strobe_width + Video_out_delay); ck++) { //クロックループ
104      bool trig_flag = 0;
105      for (int ii = 0; ii < CLOCK_WIDTH[gain]; ii++) { //クロックH幅ループ
106        digitalWrite(16, 1); //R2 clock H 出力
107      }
108      digitalWrite(16, 0); //R2 clock L 出力
109      int ad_in = analogRead(36); //ADコンバータ入力
110      bool strobe_out = (ck < (Strobe_width)) ? 1 : 0; //ストローブ信号作成
111      digitalWrite(17, strobe_out); //T2 ストロブ信号出力
112      for (int ii = 0; ii < CLOCK_WIDTH[gain]; ii++) { //クロックL幅ループ
113        digitalWrite(16, 0); //R2 clock L出力
114      }
115      if (ck > (Strobe_width + Video_out_delay)) {
116        if (pix_count < Pix_total) {
117          float Vil = (3.3/4096.0 * ((float)ad_in) + 0.14) / 0.965; //ADコンバータの誤差補正
118          unsigned short color = PLOT_COLORS[Plot_switch];
119          Screen.line(float(pix_count), Vil, start_flag, color); //グラフ描画
120          start_flag = 0; //線分描画始点フラグクリア
121        }
122        pix_count++; //ピクセルカウント
123      }
124    }
125    Plot_switch++; //グラフの描画の色の切り替え
126    if (Plot_switch >= COLOR_SELECT) Plot_switch = 0;
127    test = 0;
128  }
129  M5.update();
130 }

```

定します。測定して波形が出ない場合は、左ボタンを押してゲインを切り替え、再度中央ボタンを押して再計測してゲインを調整します。ネオン・ランプからはネオンの放電に伴う決まったスペクトラムの光が放射されるため、分光器の動作確認や校正に使用できます。

写真5は自作の紫外線LED付きマスク乾燥機を利用してマスクの透過光を測定しているところです。測定されたスペクトラムには、紫外線LEDのスペクト

ラムと室内の照明がマスクに反射したスペクトラムが表示されています。この乾燥機でマスクを乾かしたあとにすぐマスクをつけるとオゾンの匂いがしていたので、効果があるのではないかと考えていたのですが、紫外線LEDの波長は415 nmなので、殺菌やウイルスの破壊に効果がある短波長の波長はなさそうでした。

写真6はろうそくのスペクトラムで700 nmを中心広い範囲に分布しています。カメラのパーツを使う

C12880MAは缶パッケージで静電シールドされているため、ネオン・ランプを近づけてもノイズの影響を受けにくい

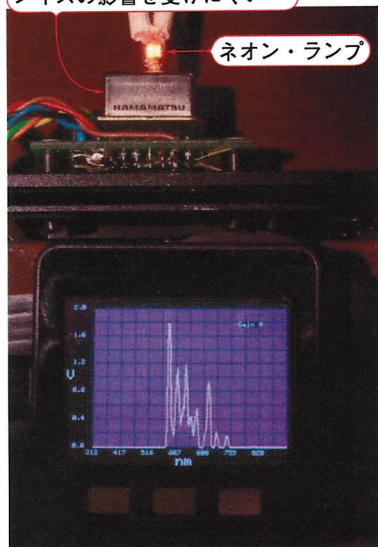
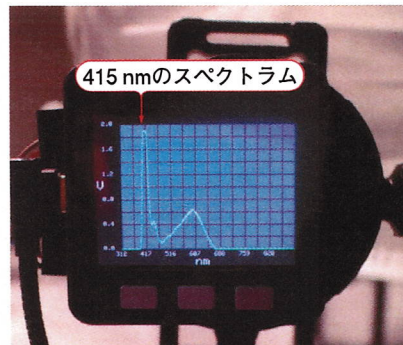


写真4 ネオン・ランプの放電発光スペクトラムは分光器の校正に使える
本写真では584 nm, 614 nm, 640 nm, 703 nmの波長が確認できた



(a) 測定中



(b) 測定されたスペクトラム

写真5 紫外線LEDのスペクトラム

使用しているLEDの波長が415 nmでその下の波長がでていないので、菌やウイルスに対する効果はあまりなさそう

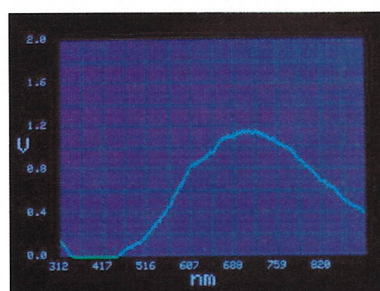


写真6 ろうそくの炎のスペクトラム
700 nmを中心とした幅広い範囲にスペクトラムが分布しているあたたかい光

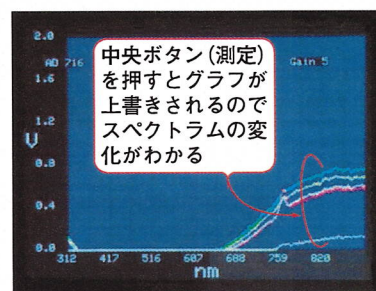


写真7 写真1の炭火のスペクトラム
波長700 nm以上の赤外線が出ている

ことでセンサの位置決めが容易なため短時間でロウソクの蠟がケーキに落ちる前に測定できました。

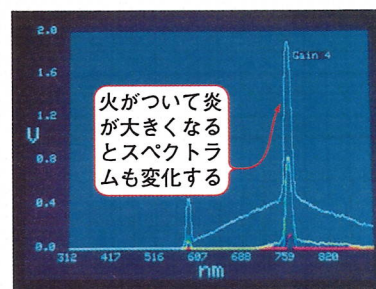
● 炎色反応の波長確認

写真7は七輪の炭火のスペクトラムで700 nm以上の赤外線領域のスペクトラムが確認できました。おそらくスペクトラムは2 μ m以上の波長まで伸びているはずで、この赤外線が美味しく焼ける秘訣です。765 nmのところ少しでているスペクトラムは、木炭にふくまれる炭酸カリウムの炎色反応と思われます。

写真1は炭火の上に油が落ちて炎が上がったときのスペクトラムです。写真8に示すように590 nmと765 nmのスペクトラムがでました。炎色反応の590 nmの波長を調べてみるとナトリウムで、油に含まれるナトリウムとカリウムのスペクトラムのようです。この実験をする場合は水などを用意して消火の準備をして火災に十分に注意してください。測定に気を取られていると黒焦げになって食べられなくなります。

カリウムのスペクトラムの766 nmを周波数にすると約390 THzの電波です。つまり、炭火に油を滴下するタイミングで390 THzの電波のON/OFF制御(A1A変調)ができるので、390 THzでモルス信号が送信できます。これは炎の有無による通信なので古代の松明の光による通信と同じことです。将来宇宙空間で孤立したときに助けを呼ぶのに使えるかもしれません。

写真8 カリウムとナトリウムと思われる炎色反応のスペクトラム
炭に油が落ちて火が付くと590 nm(508 THz)と765 nm(395 THz)あたりのスペクトラムが出る



* *

M5Stackはロジック制御とD-A/A-Dコンバータを組み合わせた計測制御と測定結果を表示する画面表示がコンパクトにまとまっていて、思いついたことをすぐ実現できます。C12880MAも制御がシンプルで簡単に本器を作ることができました。

普段撮影用カメラで使っているクイック・リリース・クランプやパン用の回転機構が、ここで製作したようなセンサの実験での位置決め固定に便利だということがわかりました。カメラ用のパーツは光学定盤の上で使う精密位置決め機構に比べて精度は落ちますが、距離や角度の設定ができて剛性も十分です。価格も光学位置決め機器に比べて安価なので実験用にそろえておくと便利だと思います。

〈鮫島 正裕〉

●参考文献●

●浜松ホトニクス, C12880MA データシート