ものづくりやろう!

第六回 SWR 表示器の製作

JH3RGD 葭谷安正

■はじめに

先日シャックの模様替えをしました。机の横に棚を置いて、机と棚の上に 190cm 長の化粧板を渡して使用可能な面積を大きくし、そこに無線機を置きなおしました。操作性はよくなりましたが、その時にアンテナチューナを落下させてしまいました。チューナはマッチングがとれ大丈夫でしたが、パワー計と SWR 計が機械的ショックで動きがおかしくなってしまいました。SWR を計測するのに必要な進行波電圧と反射波電圧を検出する回路は、該当端子にマルチメータをあてて直流電圧を計測したところ動いていることが判明。またパワー出力もこの進行波電圧の端子からもらっていることが回路図からわかり、どうやら故障個所はメータの模様。このメータは電力計と SWR 計を兼ねているメータで、今は販売していないのではと思い、ネットでの検索もしていません。このチューナには IC-705 と TS-120 をつないでいますので、IC-705 を使うときは IC-705 の SWR 表示機能を使えばよいのですが、TS-120 ではトランシーバ上で SWR を表示できませんのでマッチングのために SWR 計が動いてくれるとありがたい状態です。進行波電圧と反射波電圧の測定回路が生きているのですから、とりあえず手持ちの Arduino の AD 変換端子から両電圧を読み込み、Arduino に SWR を計算・表示させることができます。そこで、Arduino を使った SWR 表示器を作りました。あくまでもメータを入手するまでの仮の回路として使用していますが、このまま永続的な使用になりそうです。作るというよりは、配線とソフトウェア製作になります。インターネット上にもたくさん参考資料がありました。

■SWR の測定回路

進行波・反射波・電力取得用端子

アンテナチューナ FC-700 の回路図を眺めて、どの端子を使えば SWR や出力電力の情報を取得できるのか確認しました。図 1 は FC-700 の取扱説明書中に記載のあった SWR と電力表示に関係する回路図です。これをたどっていくと、トランシーバからの送信出力は J01 から入り、CM カップラユニットを通過してマッチング回路に入り、アンテナに接続されます。

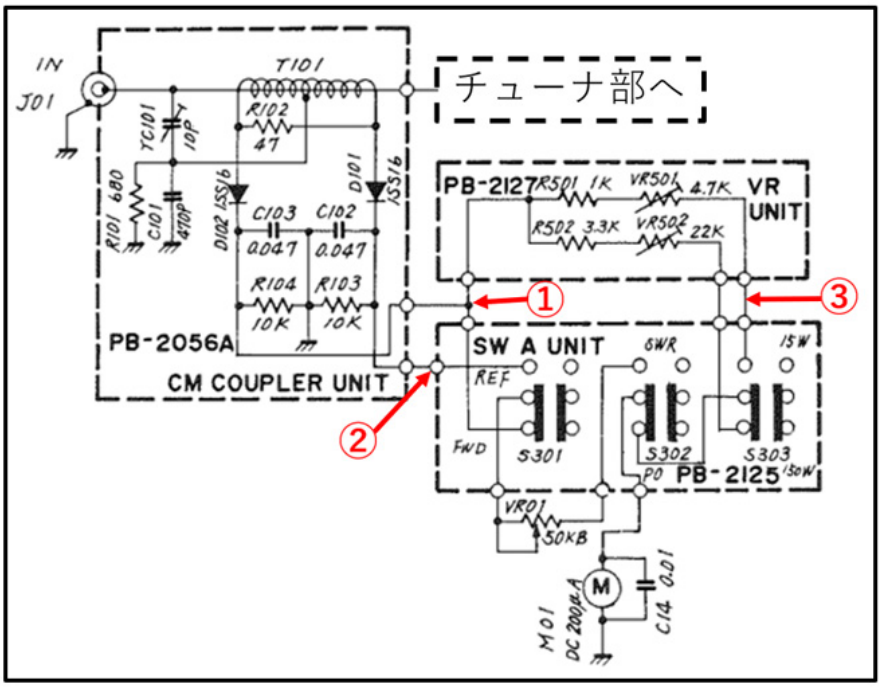

図 1 CM カップラユニットの回路図

SWR の測定には、進行波の電圧と反射波の電圧が必要になります。進行波の電圧は、T101 で高周波結合されて、進行波成分がダイオード D102 と RC 回路で整流されます。図 1 中の①端子にその電圧が出てきます。

また、反射波の電圧も同様にダイオード D011 と RC 回路で整流されます。図 1 中の②端子にその電圧が出てきます。この 2 つの直流成分をメータに導いて SWR を表示しています。そこでこれをマイコンで計算する場合は Arduino の AD 変換器で、端子①、端子②の電圧を読み取り、SWR(ρ)は右の定義式で計算して求めればよいことになります。

$$\rho = \frac{1 + \left| \frac{②の電圧}{①の電圧} \right|}{1 - \left| \frac{②の電圧}{①の電圧} \right|}$$

電力表示については、回路図から図 1 中の①端子の電圧を電力に換算して表示してことがわかります。また、送信出力が 15 ワット以下と 150 ワット以下をプッシュスイッチで切り替えて選択していますが、最大出力が 15 ワットでも 150 ワットでも同じメータで表示できるようメータ直前に進行波出力の電圧を調整するための回路が挿入されており(R501,VR501 と R502,VR502)、2 つの回路の出力をプッシュスイッチで選択しています。私の使用方法ではこのチューナに接続する無線機の出力はいずれも 10 ワット以下ですので最大出力 15 ワットの回路だけで十分です。そこで 150 ワット用の出力端子は使用しないことにし、このため図 1 中の端子③から電力に比例する電圧情報 Arduino で読み込み、その値から出力電力を Arduino で計算して表示することにしました。計算式は加減乗除と絶対値の計算ですから Arduino にまかせてしまいます。

■SWR 表示器の構成と接続図

SWR 表示器の構成と接続図を図 2 に示します。SWR 値や出力電力を表示するために液晶表示器(LCD)を使いました。部品箱に LCD 用の I2C インターフェースがありましたので LCD とこれをブレッドボードで接続して使用することにしました。

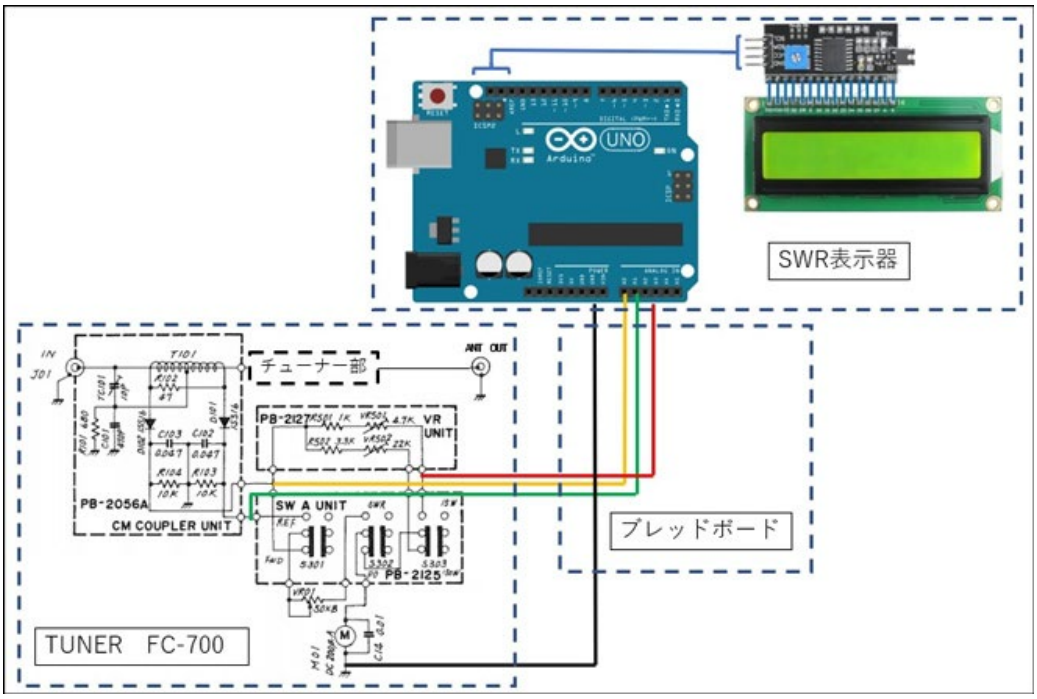

図 2 SWR 表示器の構成と接続図

■事前調整

Arduino の AD 変換部を使って SWR や電力を求めるために、事前に簡単なプログラム(リスト 1)を作成して、電圧と電力の関係やデータ変動についてチェックをおこないました。チューナと SWR 表示器との接続は図 2 のように Arduino と I2C ボード、LCD をワイヤー線で接続しました。またチューナの端子①、端子②、端子③にミノムシクリップを使用して簡易的に接続し、そのミノムシクリップの反対側はブレッドボード経由で Arduino と接続できました。この状態で次のリスト 1 に示すプログラムを使用してつぎの項目のチェックをおこないました。

リスト 1 データ変動確認用プログラム

```
#include <LiquidCrystal_I2C.h>
#define Cnt 50
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
  lcd.init();
  lcd.backlight();
}
void loop() {
  int i, v1, v2, IVpwr;
  float const1 = 5.0 / 1023.0;
  float Vfs, Vrs, Vf, Vr, Vp, Vpwr, Vpwr2, swr, aswr ;
      Vfs=Vrs=Vpwr=0.0;
      for(i=0 ; i<Cnt ; i++){
        Vfs  = Vfs+const1 *analogRead(A0);
        Vrs  = Vrs+const1 * analogRead(A1);
        Vpwr = Vpwr+const1 * analogRead(A3);
        delay(5);
      }
      Vfs=Vfs/Cnt;Vrs=Vrs/Cnt;Vpwr=Vpwr/Cnt;
      swr = (Vfs + Vrs) / (Vfs - Vrs);
      Vpwr2 = 15.055*Vpwr*Vpwr+8.3*Vpwr+0.0011;
      lcd.setCursor(0, 0) ; lcd.print("Vf=");lcd.print(Vfs,2) ;
                            lcd.print(" Vr=");lcd.print(Vrs,2) ;
      lcd.setCursor(0,1);lcd.print("                ");
      lcd.setCursor(0,1);lcd.print("Vp=");lcd.print(Vpwr2,1);
      lcd.print("S=");lcd.print( (Vfs+Vrs) / (Vfs-Vrs), 1);
      delay(500);
}
```

(1) 進行波の電圧(端子①)の安定化

(2) 進行波の電圧と出力電力の関係

・「(1)進行波の電圧(端子①)の安定化」について

リスト 1 のプログラムを動かして、Arduino で読み込んだデータが変動しているので、そのデータ変動を抑えるためにデータ平均を行いましたが、何個のデータの平均をとれば変動を抑えられるかについて確認できたことを記載します。リスト 1 のプログラムが測定していることは、進行波電圧測定用の端子電圧(端子①の電圧値、変数 Vfs に格納)、反射波電圧測定用の端子電圧(端子②の電圧値、変数 Vrs に格納)、電力測定用の端子電圧(端子③の電圧値、変数 Vpwr に格納)を、リスト 2 行目の Cnt の数値件のデータの平均値を表示し続けています。このプログラムを動かし、無線機のモードを CW に設定

し、５秒程度電鍵を ON にしたままの状態にすると LCD 上に上記３つの端子電圧の平均値が表示されます。この平均値を目視チェックしたところ、平均する個数が少ない場合に電圧変動がかなり見受けられました。これは進行波や反射波などの交流成分をダイオードで直流化してもその上に高周波成分が残っているためでないかと推測しました(オシロがあれば検証できるのですが、私のオシロ、壊れてしまっていますので･･･)。

このため、複数データを読み込み、その平均値を求めることで変動抑制をしてみました。これはデジタル信号処理でローパスフィルタに相当するものです。

平均するデータの数と平均後の値の変動を示したのが次の表 1 です。表 1 の中で、加算平均するデータ件数が 5 個まではディスプレイ上の値が変動していることが目視確認できました。それ以上の個数のデータ平均では小数点第 2 桁目がたまに変動するくらいで大きな変動はありませんでした。表 1 のデータから、10 個以上のデータを平均したものを使用することでノイズ成分が除去できることが確かめられました。

データ数	端子①電圧の変動
1	0.47～0.59
3	0.51～0.57
5	0.54～0.56
10	0.54
20	0.56
50	0.54
100	0.55

表 1 データ平均数とデータ変動の確認表

・「(2)進行波の電圧と出力電力の関係」について

リスト 1 のプログラムを動かして、進行波電圧と出力電力の関係について確認できたことを記載します。

表 2 出力電力と端子電圧との関係

IC-705の出力電力設定値(W)	進行波電圧端子の電圧 (①端子電圧)(V)	電力端子の電圧 (③端子電圧)(V)
1	0.10	0.10
2	0.18	0.18
3	0.25	0.25
4	0.31	0.30
5	0.37	0.36
6	0.41	0.40
7	0.46	0.45
8	0.50	0.50
9	0.54	0.53
10	0.59	0.59

表２は IC-705 の出力可変ダイアルを回して１ワットから 10 ワットまで変化させた時、進行波電圧端子の電圧と電力端子の電圧がいくらになるかということを測定しました。本来ならば標準信号発生器などの校正用機器を基準にして、出力電力と端子電圧の関係を求めればよいのですが、そのような機器が手元にありませんので、基準出力を IC-705 として端子電圧を測定したものです。表は 7.001MHz の無変調信号(CW 信号)を入力しました。また、電圧値はすべて出力を固定したときに、50 個のデータの平均を連続して観測し、またアンテナの SWR は IC-705 の SWR 計を基準にして SWR=1.0 の状態(反射電圧、すなわち端子②の電圧が 0V)で進行波電圧端子の電圧と電力端子の電圧を表にまとめました。

表１の進行波電圧端子の電圧を x、IC-705 の出力電力設定値を y として散布図(x-y グラフ)を描くと図３のようになりました。Arduino で出力電力を表示するためには、端子③の電圧(端子①電圧とほぼ同じ)からこの図の関係を使って、出力電力を端子③電圧から求めます。図３から、電力と電圧の関係は二次関数で近似できそうですので Excel の近似機能を使って近似式を求めると図３中に見える次のような式になります。

$$y=15.055x^2+8.2992x+0.0011$$

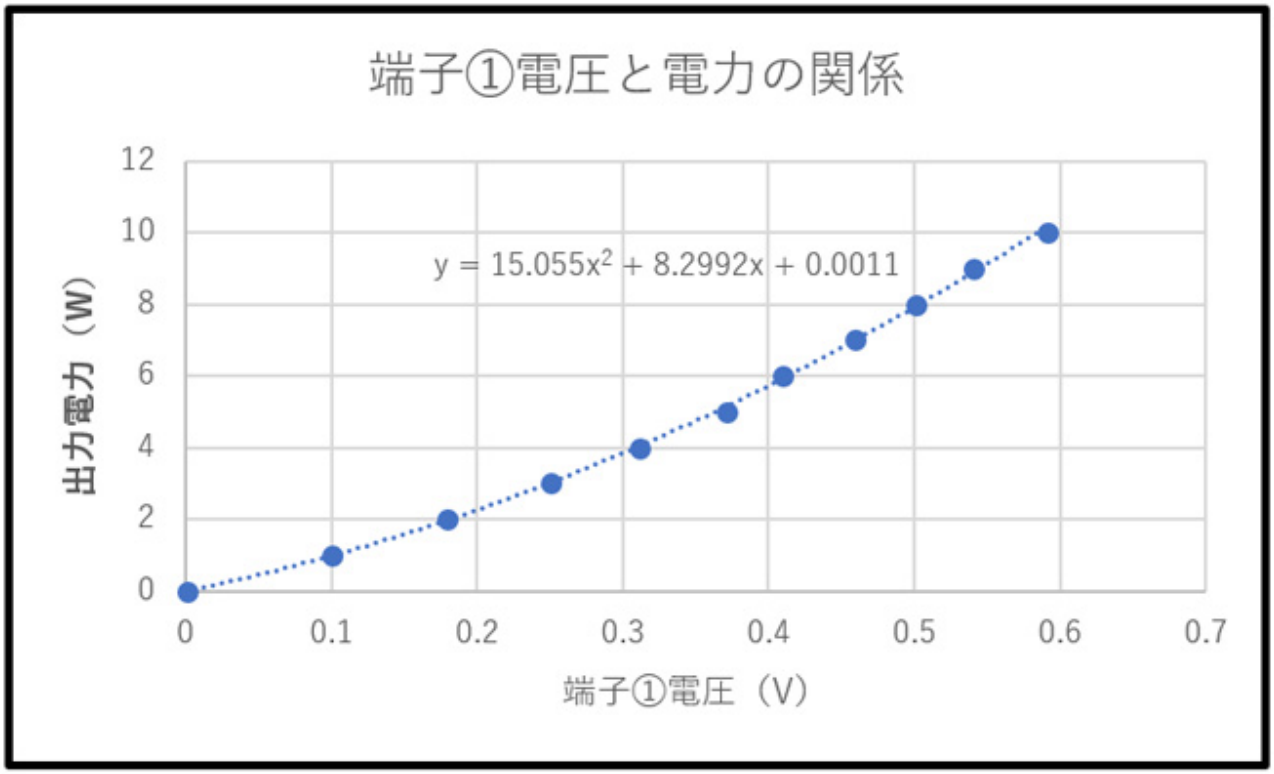

図３ 端子①電圧と出力電力の関係

この式で、x は①端子電圧に相当し、y は出力電力に相当します。端子③の電圧が測定できれば、この式の x に端子③の電圧をいれることで、その時の出力電力がわかるわけで Arduino のプログラム中にこの式を入れます。(図３では「端子①電圧･･･」と記載してありますが、端子①の電圧と端子③の電圧は、表２のようにほぼ同じ値でしたのでこのような説明をさせていただきました。)

この式を使って y を計算により求めたものが表 3 です。表 3 の 2 列目の数値を x としてもとめたのが表 3 の 3 列目の出力電力の推定値です。1 列目の数値(IC-705 の出力設定値)の近似値であることが確認できます。

電力（W）	端子③電圧（V）	推定値
0	0	0.0
1	0.1	1.0
2	0.18	2.0
3	0.25	3.0
4	0.31	4.0
5	0.37	5.1
6	0.41	5.9
7	0.46	7.0
8	0.5	7.9
9	0.54	8.9
10	0.59	10.1

表 3 式 y による電圧から電力の推定値を求める

また SWR を測定するためには 4 ワット以上の出力電力がないと誤差が大きいようであることも判明しました。

■プログラム

以上をもとに SWR と出力電力を求めるプログラムを作成しました。ソースプログラムは参考として記事の最後に記載しました。

■配線

Arduino と LCD、そしてチューナをつなぐことがメインになります。

SWR 表示器

Arduino と LCD を接続しました(図 4)。これをケースにいれるのですが適当なケースがないので段ボール箱に格納しました。

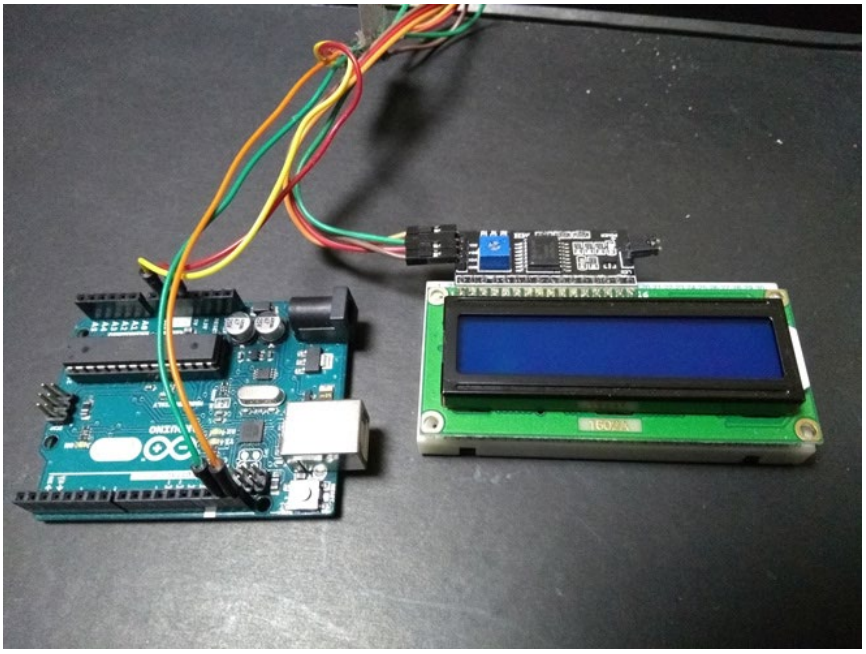

図 4 組み込み前の Arduino と LCD

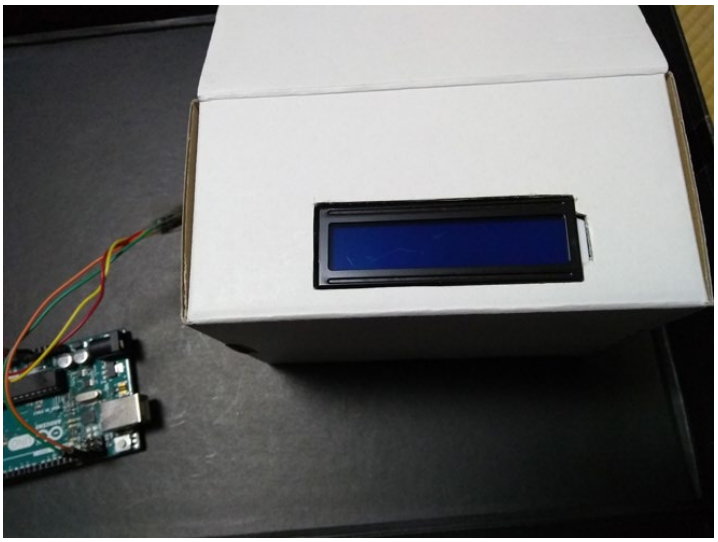

図 5 Arduino と LCD の組み込み(1)

図 5 のように段ボール箱に LCD が入るように切り抜き、内部には Arduino を入れるため、構造物で梁をつくりました。

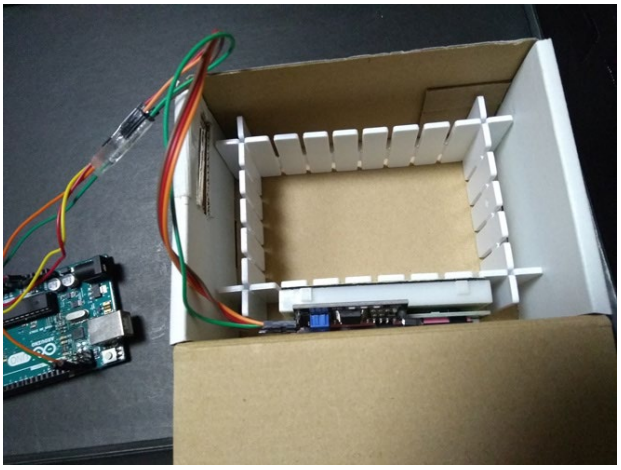

図 6 Arduino と LCD の組み込み(2)

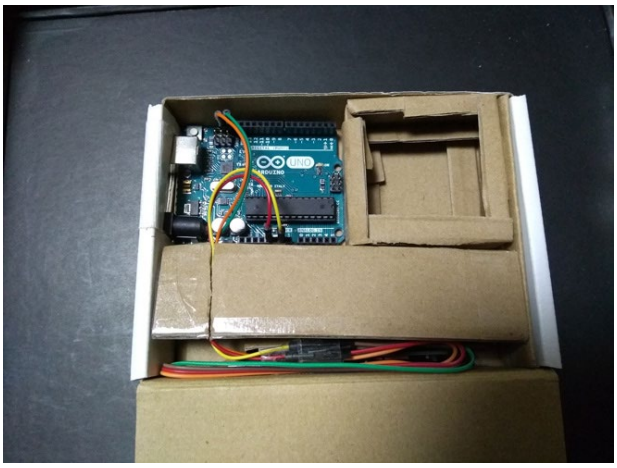

図 7 Arduino と LCD の組み込み(3)

外部に線を引き出し、できあがりです。

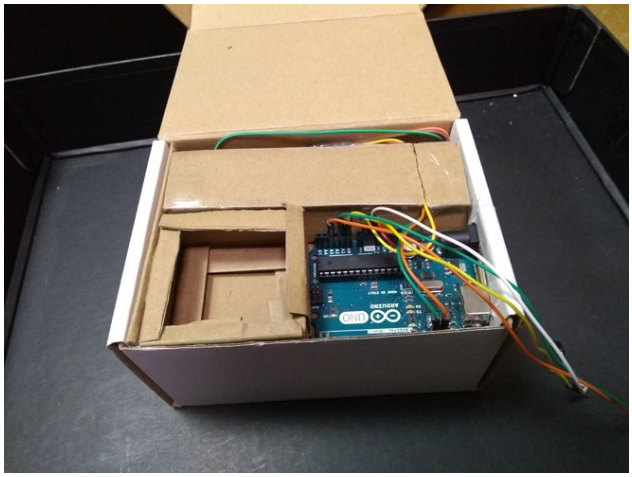

図 8 Arduino と LCD の組み込み(4)

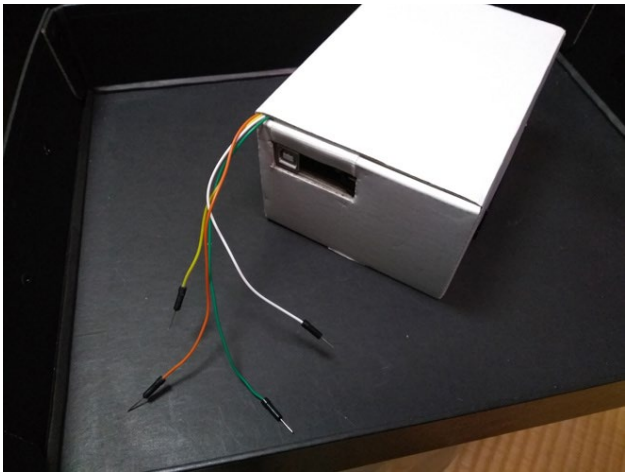

図 9 SWR 表示器結線終了

電源は USB 端子を使います。実はこの箱はパドル(CW ONE)が入っていた箱です。

チューナ内配線

加工前のチューナ内の状況です。

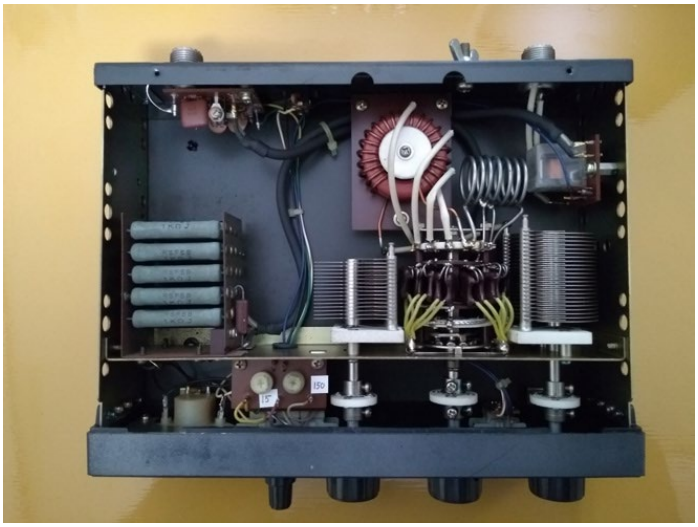

図 10 チューナ内部

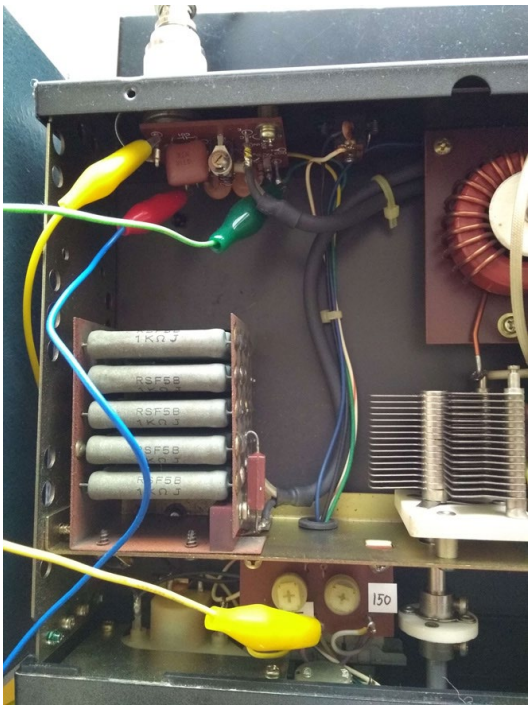

図 11 チューナ内部の仮配線

事前調整の時には図 11 のようにミノムシクリップでつかんで行いました。下部黄色のミノムシクリップは電力測定のための端子③です。上部黄色が端子①(進行波測定用電圧端子①)、緑色が端子②(反射波測定用電圧端子)、赤色がグランドです。このままでは危険ですので、安全のためちゃんと半田付けします。図 12 のような端子を作成し、この端子に結線を集約して外部に出しました。

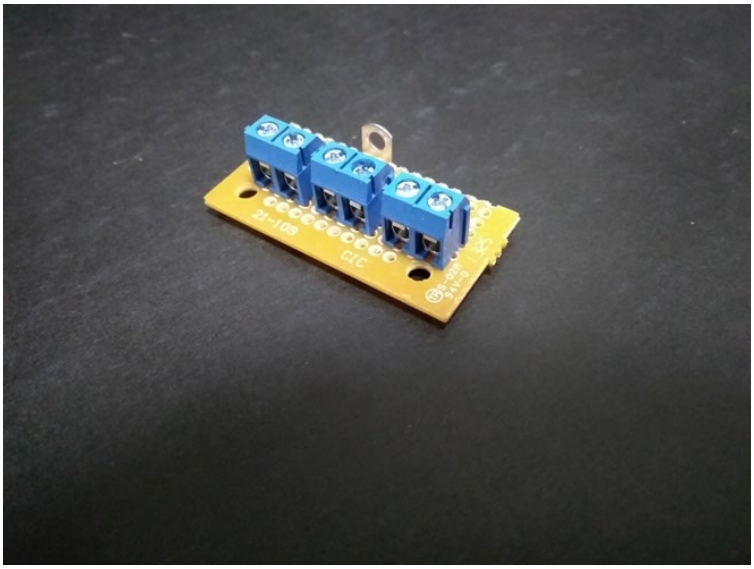

図 12 配線用端子

図 13 がチューナ内配線を端子へ集約したものです。

図 13 チューナ内部配線(1)

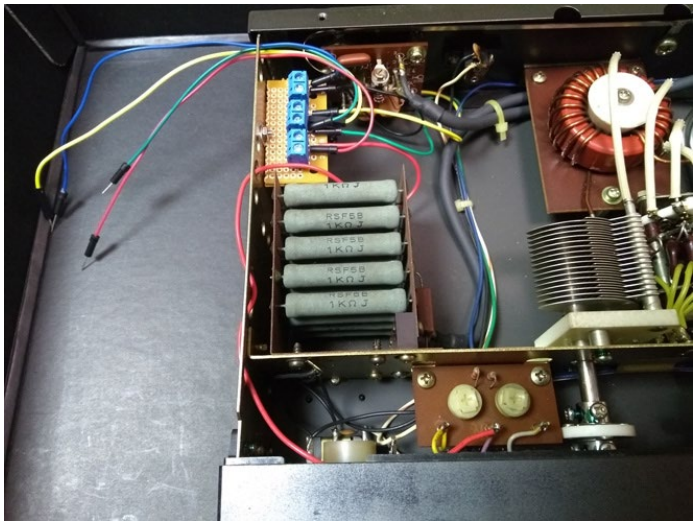

図 14 チューナ内部配線(2)

集約した端子からは、チューナの既存の穴を利用してそこから外部に配線を引き出しました。

チューナのふたをします。あとは表示部との結合です。

図 15 チューナ内部配線(3)

チューナの上にブレッドボードをテープで貼り付け、そこに SWR 表示器からの配線とチューナからの配線を結合しました。

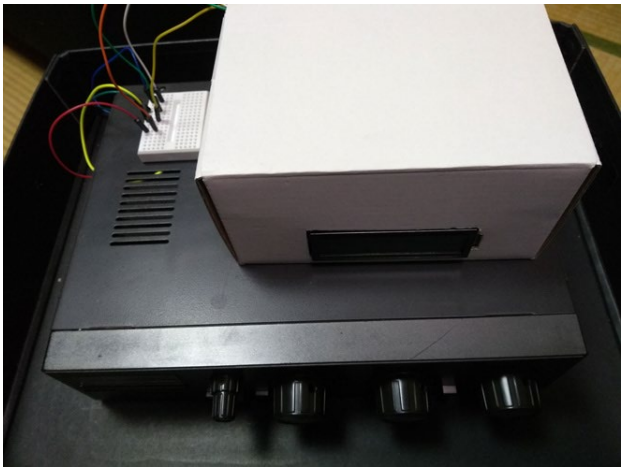

図 16 LCD と Tuner の結線

図 17 チューナと SWR 表示器

動作

動作時の表示例を図 18 に示します。

LCD の上段には、P:電力(反射電力) Swr:SWR 値を表示します。

LCD の下段には、SWR 値を視覚化するため四角(■)を表示しました。

LCD16 文字表示できますので、SWR 値が 3 の時 8 文字目まで■が表示されるようにプログラミングしました。指定時間(現プログラムでは 10 秒)経過すると LCD が消えます。

図 18 SWR 表示器の表示例

図 19 指定時間経過後のスリープ状態

問題点

運用してみるといくつかの問題がでてきました。一番困ったことは、「バンドによっては大きな誤差がある」ことです。製作した SWR 表示器は 7MHz、10MHz、18MHz では IC-705 の SWR 値と比較的近い値を表示するのですが、14MHz では大きく異なってしまいます。原因不明です。いつか直したいと思いながらずっとそのままです。コロナが収束する前に原因究明と対策を施したいと思いますが、いつになることやら。

■ソースリスト(参考)

記事の中に記載していない下記の機能をいれてあります。

- 省エネのため、タイマーを利用して出力がなくなってから 10 秒経過(Ktime の値）すると LCD のバックライトが off になります。バックライトのない LCD ではこの機能が働きません。
- 最大電力が 150 ワットを計測できるようにプログラム構造を準備してあります。(チューナと Arduino の配線を追加する必要があります)

```
#include <LiquidCrystal_I2C.h>
#include <MsTimer2.h>
#define Cnt 100
#define Ktime 10000
boolean timeron = true ;
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
  MsTimer2::set(Ktime, flash); // タイマー時間を Ktime ms にセット
  timeron = false;
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);lcd.print("SWR display     ");
  lcd.setCursor(0,1);lcd.print("for FC-700      ");
  delay(5000);
}
// the loop routine runs over and over again forever:
void ftimeron() {  // Timer start
  MsTimer2::start();
  lcd.backlight();
  timeron = true;
}
void flash() {  // Timer stop
  //lcd.noCursor();
  MsTimer2::stop();
  timeron = false;
}
void loop() {
  int i, v1, v2, IVpwr;
  float const1 = 5.0 / 1023.0;
  float Vfs, Vrs, Vf, Vr, Vp, Vpwr, Vpwr2, swr, aswr ;
  float Vfmax, Vfmin, Vrmax, Vrmin;
  char FF=255;
```

```
  while (!timeron) { // Timer off から Timer On への切替条件
    lcd.noBacklight();
    Vf = const1 * analogRead(A0);
    if (Vf > 0.05) {
      ftimeron();
    }
    delay(300);
  } // end of **while(!timeron){
  while (timeron) { // Timer on 時の処理内容
    Vf = const1 * analogRead(A0);
    Vpwr = const1 * analogRead(A3);
    delay(10);
    if (Vf > 0.05) {
      Vfs = 0; Vrs = 0; Vpwr = 0; Vpwr2 = 0;
      for (i = 0; i < Cnt; i++) {
        Vfs  = Vfs + const1 * analogRead(A0);
        Vrs  = Vrs + const1 * analogRead(A1);
        Vpwr = Vpwr + const1 * analogRead(A3);
        Vpwr2 = 0; //Vpwr2= Vpwr2+const1*analogRead(A5);
      }
      swr = (Vfs + Vrs) / (Vfs - Vrs);
      if (Vpwr > Vpwr2) {
        Vpwr = Vpwr / Cnt;
        lVpwr = int(15.055 * Vpwr * Vpwr + 8.2992 * Vpwr + 0.0011 + 0.5);
      }
      else {
        Vpwr = Vpwr2 / Cnt;
        lVpwr = int(15.055 * Vpwr * Vpwr + 8.2992 * Vpwr + 0.0011 + 0.5);//式変更の必要
あり
      }  // end of **if (Vpwr > Vpwr2)
    }  // end of **if (Vf > 0.05) {
    lcd.setCursor(0, 1) ; lcd.print(″                ″);
    if (!isnan(swr)) {
      lcd.setCursor(0, 0);
      lcd.print(″                ″);
      lcd.setCursor(0, 0);
      aswr = abs(swr);
      lcd.print(″P:″ + String(lVpwr) + ″(″); lcd.print((lVpwr * (1.0 - (aswr - 1) / (aswr + 1))), 1);
lcd.print(″)″);
      lcd.print(″ Swr:″); lcd.print((abs(swr)), 1);
```

```
        int k = min((swr - 1) * 5, 16);
        lcd.setCursor(0, 1);
        lcd.print("                ");
        for (i = 0; i < min(k, 16); i++) {
          lcd.setCursor(i, 1); lcd.print(FF);//lcd.print(">");
        }
        delay(100);
      }
    }  // end of **if (Vf>0.05)
}  // end of  **while(timeron)
```