簡易静電容量スイッチ

概要

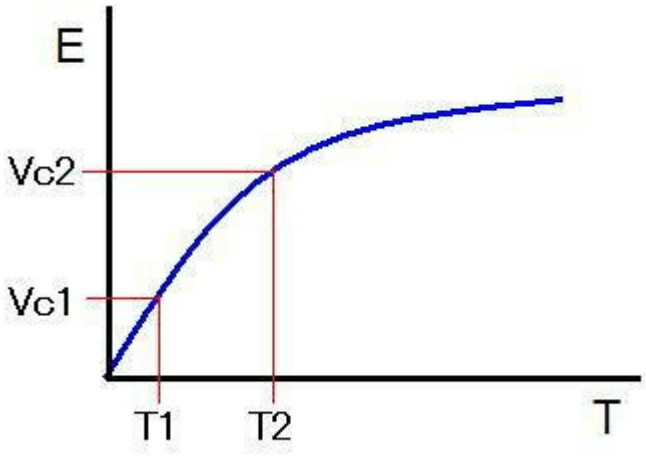
静電容量の変化を検出して、制御(今回は、LEDとブザーを接続)を、ONまたはOFFする小型のユニットを 製作しました。 用途としては、人体や水などによる静電容量の変化による制御が考えられますが、他にも静電容量の変化が 得られる(例えば、接地抵抗など)のであれば活用範囲が大いに広がるのではないでしょうか。

動作原理

静電容量(キャパシタンス:capacitance)の測定には、コンデンサの充電特性における、充電時間(T1□T2)を 計測することにより求めます。

- 充電時間は、静電容量の大きさに比例する。
- 静電容量が小さいときは、充電時間は短い。
- 静電容量が大きいときは、充電時間は長い。

基本的な考え方は、以前に製作した、「キャパシタンスの測定」を参照してください。



<コンデンサの充電特性>

「キャパシタンスの測定」では□PIC16F88を使用したので、コンパレータを2個使用することが出来ました。 しかし、今回はPIC12F683を使用したので、コンパレータを1個しか使用することが出来ません。 そこで、コンパレータに与える規準電圧(CVref)を切り替える(ソフト的に)ことにより、同等の機能を実現する ことにしました。

今回の製作では、次の二つの機能を搭載しました。

- 静電容量スイッチ
  静電容量の変化(±5%)検出し、LEDをONまたはOFFする。
- 静電容量計
  静電容量を測定し、RS232C経由でパソコンに送り、通信ソフトで受信し表示させる。

浮遊容量をキャンセルする機能も搭載しました。

<充電時間(T2-T1)を測定する処理の流れ>

1. CVrefをVc1の値に設定する。
2. コンデンサを短絡(トランジスタ=ON)させて、電荷を放電させる。
3. コンデンサの電圧が、CVref(Vc1)に達するまで待つ。
4. CVrefをVc2の値に設定する。
5. タイマーを開始させる。
6. コンデンサの電圧が、CVref(Vc2)に達するまで待つ。
7. タイマーを停止して、ターマー値を得る。

<静電容量スイッチとしての処理の流れ>

1. 起動時(またはSW押下時)に充電時間を測定し、その値を基準値(TM1)とする。
2. 更に充電時間を測定し、その値を実測値(TM2)とする。
3. TM1とTM2を比較し、スイッチのON/OFFを判定する。
   TM1に対してTM2が±5%の範囲内であればLED=消灯、ブザー=OFFとする。
   TM1に対してTM2が±5%の範囲外であればLED=点灯、ブザー=ONとする。
4. 2.へ戻る。

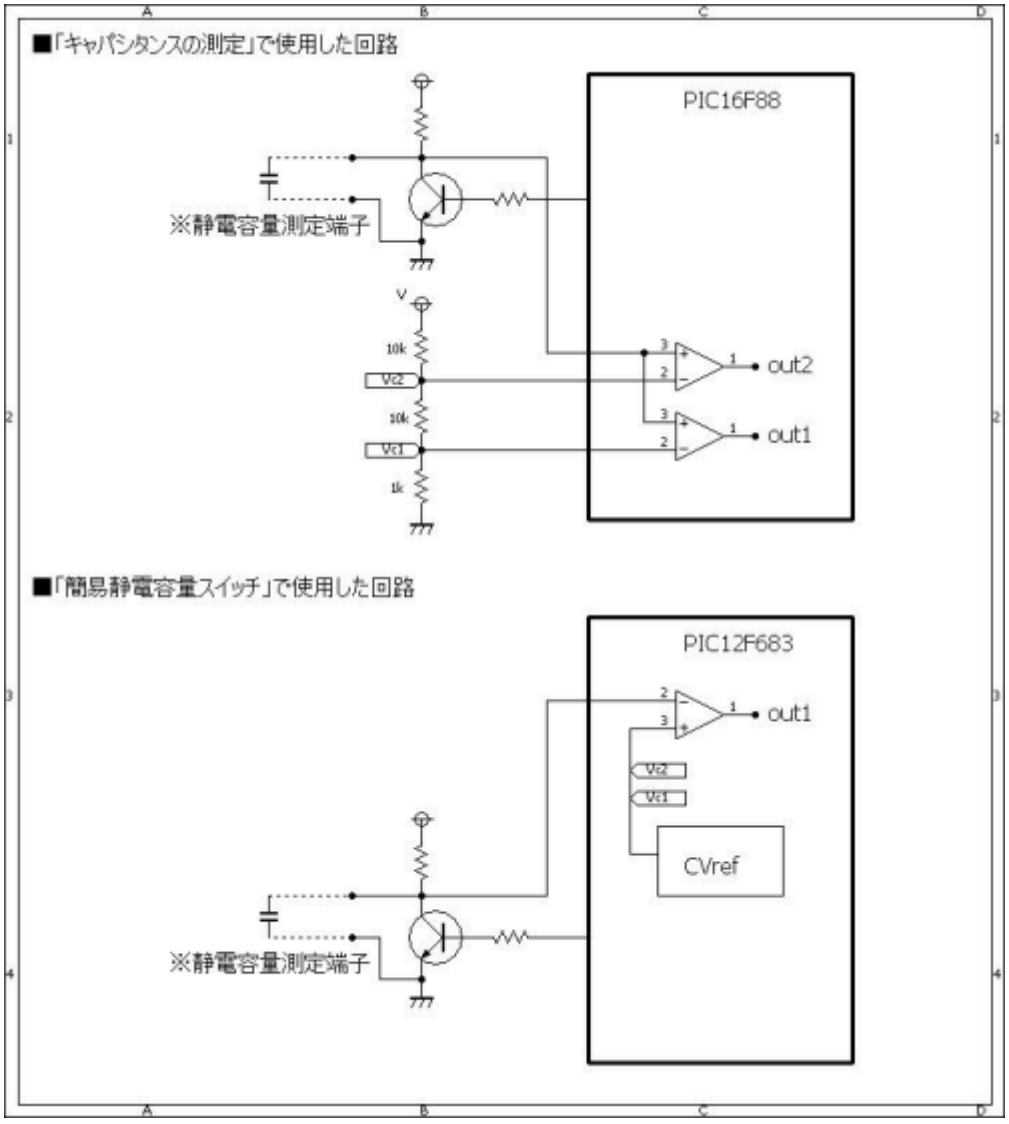
<静電容量計としての処理の流れ>

1. 静電容量測定端子に何も接続しない状態で、SWを押下しながら電源を入れる。
2. SWが離されると、充電時間を測定し、その値を浮遊容量値(TM0)とする。
3. SWが離されている間、LEDを点滅させる。
4. 基準となる容量(1000pF)のコンデンサを測定端子に接続する。
   ※出来るだけ精度の高いコンデンサを使用してください。
5. SWが再度押下されると、充電時間を測定し、その値を基準容量値(TM1)とする。
6. 更に充電時間を測定し、その値を実測容量値(TM2)とする。
7. TM0、TM1、TM2の値より、容量値(pF)を求める。
   容量値(pF)=(TM2-TM0)/[1)]

1)
TM1-TM0)/1000.0)

1. 容量値(pF)を文字列に変換してRS232C(9600bps)経由でパソコンに送信する。
2. パソコンの通信ソフト(ハイパーターミナル等)でデータを受信する。
3. 6.へ戻る。

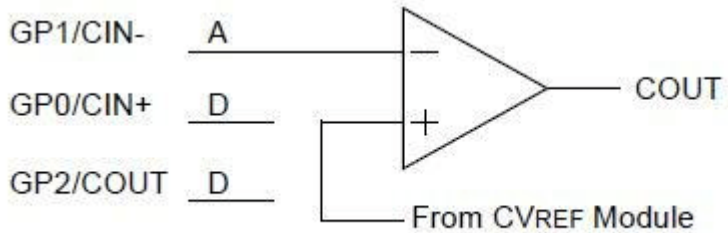
<コンパレータ1個で、2電圧値を判断する方法>



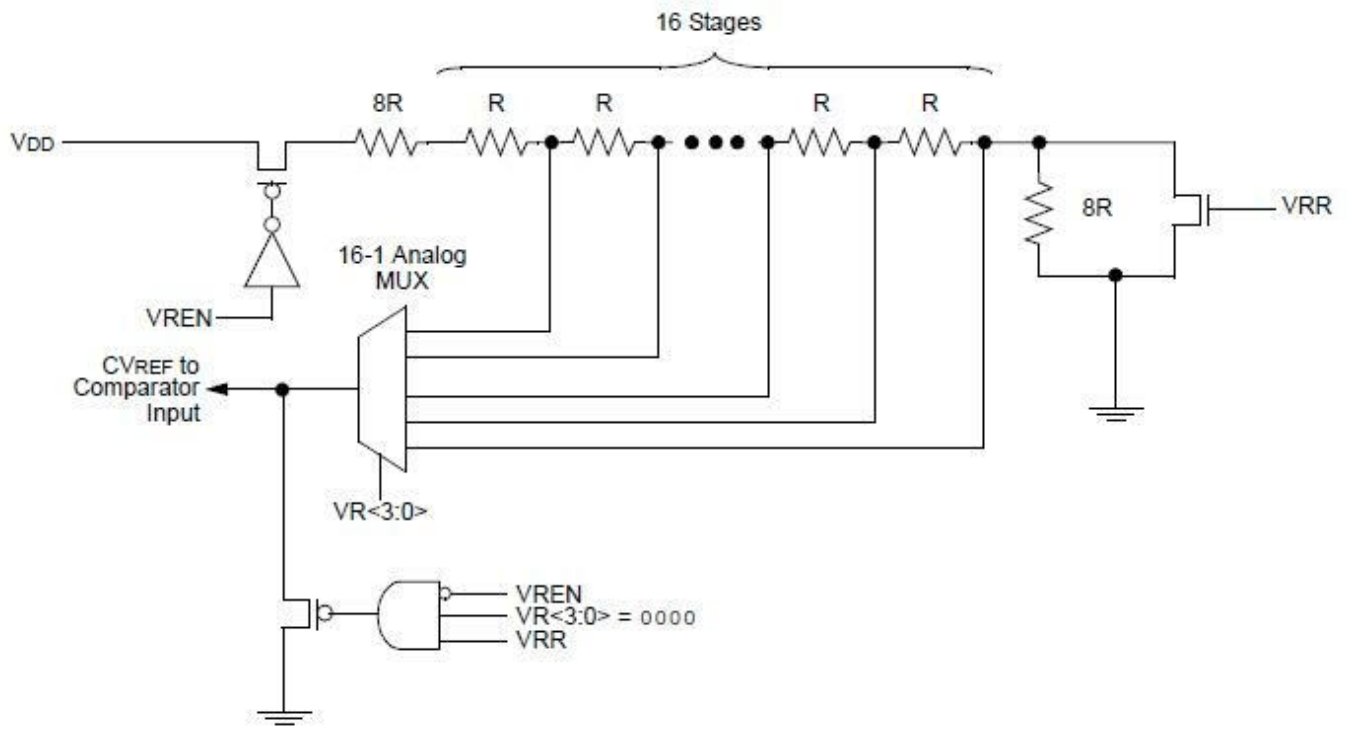
<コンパレータの使用モード> 基準電圧はCVrefを使用する。

Comparator w/o Output and with Internal Reference

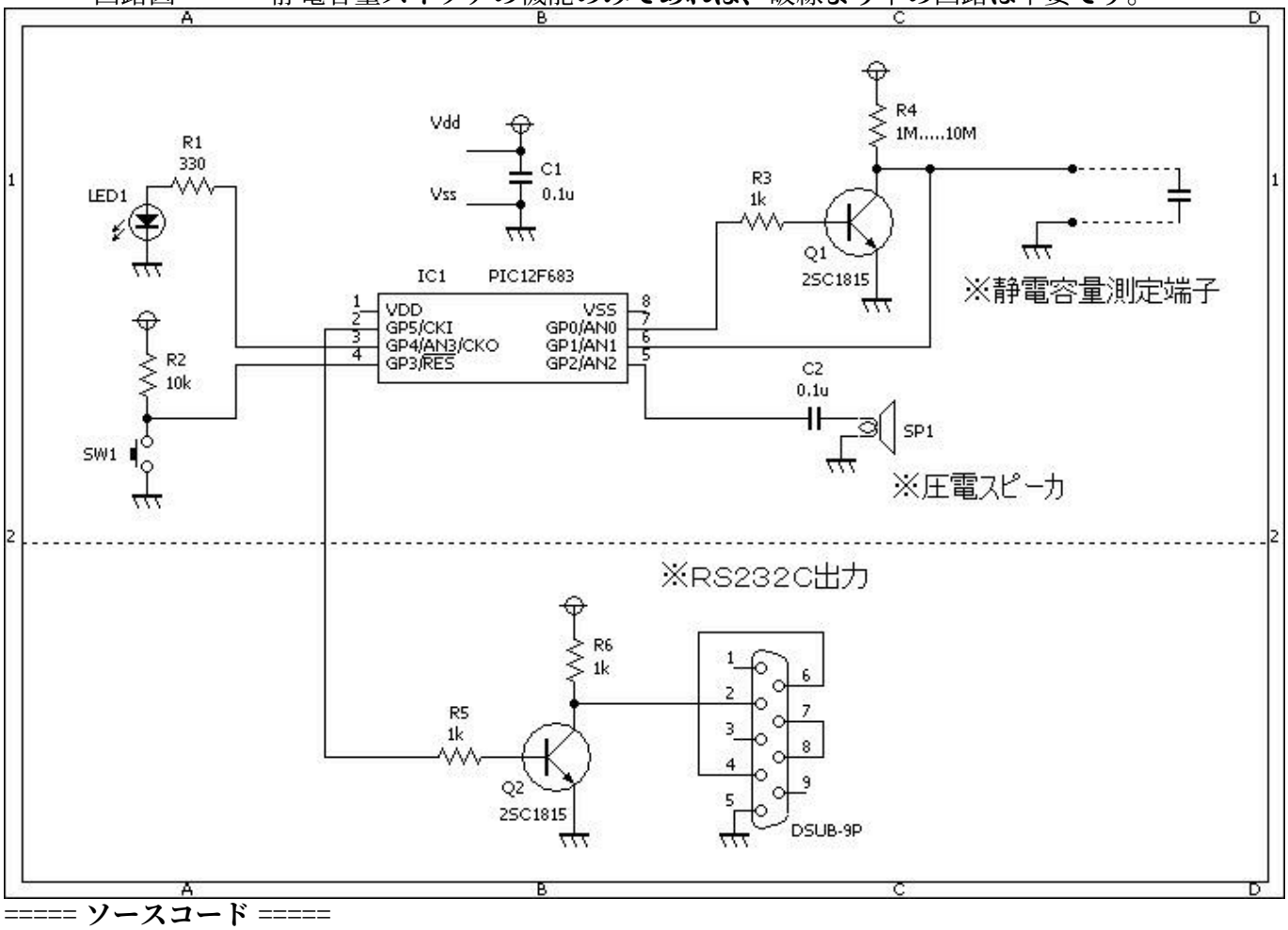
CM<2:0> = 100



<コンパレータ基準電圧のブロックダイアグラム> VRR=1(Low range)の時、0V~3.125Vまでを16段階で切り替えが可能です VRR=0(High range)の時、1.25V~2.34Vまでを16段階で切り替えが可能です。



===== 回路図 ===== 静電容量**スイッチの機能のみであれば、破線より下の回路は不要です。**



===== **ソースコード** =====

capacitanceSwitch.c

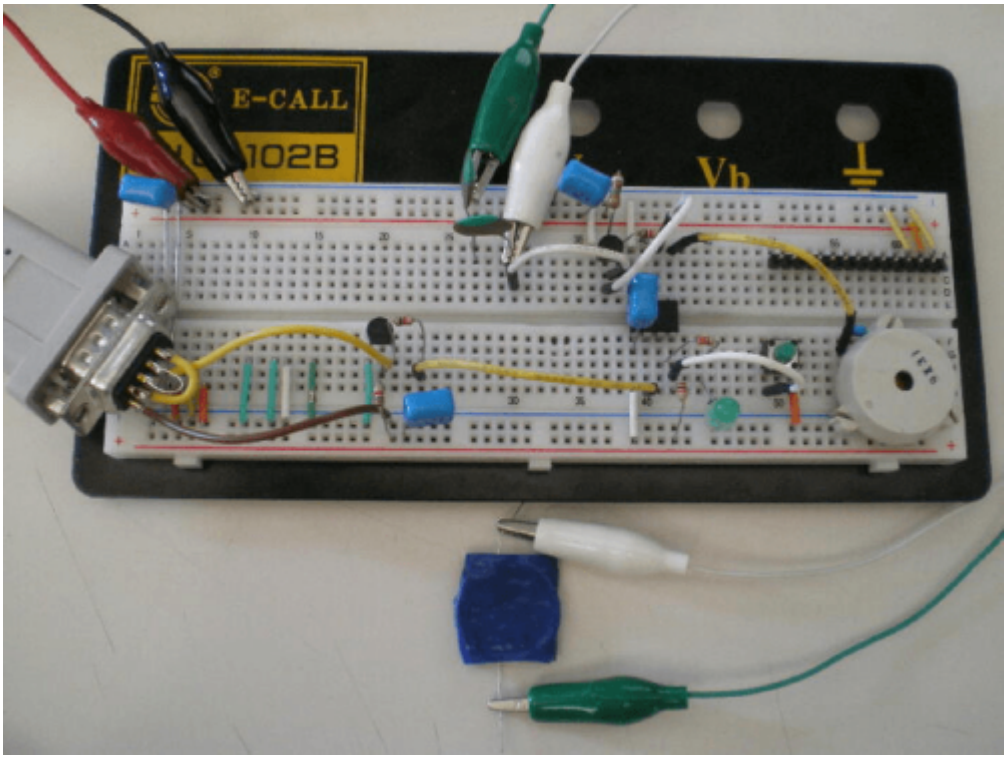
```
//*********************************************************************
*
```

```
/*
    <静電容量スイッチ>
*/
//*********************************************************************
*

#define       LED        GPIO.F4
#define       SW        GPIO.F3

//*********************************************************************
*

void   Pwm_Change_DutyEx(unsigned int duty_ratio)
{
    CCPR1L = duty_ratio >> 2;
    CCP1CON.DC1B0 = duty_ratio & 0b00000001;
    CCP1CON.DC1B1 = (duty_ratio & 0b00000010) >> 1;
}

//*********************************************************************
*

unsigned   int    measurement()
{
    static    unsigned    int    tm;
    //
    TMR1L = 0;
    TMR1H = 0;
    PIR1.TMR1IF = 0;
    T1CON.TMR1ON = 0;
    //
    VRCON.VR3 = 0;
    VRCON.VR2 = 0;
    VRCON.VR1 = 0;
    VRCON.VR0 = 1;
    //
    GPIO.F0 = 1;
    Delay_us(100);
    GPIO.F0 = 0;
    //
    while (CMCON0.COUT == 1)
        ;
    //
    VRCON.VR3 = 1;
    VRCON.VR2 = 1;
    VRCON.VR1 = 1;
    VRCON.VR0 = 1;
    //
    T1CON.TMR1ON = 1;
    while (CMCON0.COUT == 1)
        ;
```

```
    //
    T1CON.TMR1ON = 0;
    tm = TMR1H << 8;
    tm |= TMR1L;
    //
    return(tm);
}

//*********************************************************************
*

unsigned   int    Average(int cnt)
{
    static    unsigned    long    tm;
    static    unsigned    int        i;
    //
    tm = 0;
    for (i = 0; i < cnt; i++) {
        tm += measurement();
    }
    return(tm / cnt);
}

//*********************************************************************
*

void   Soft_Uart_Write_Str(char* str)
{
    while (*str != 0x00) {
        Soft_Uart_Write(*str);
        str++;
    }
}

//*********************************************************************
*

void main()
{
    static    long    tm0, tm1, tm2;
    static    short    mode, cnt, on_cnt;
    static    char    buf[10];
    static    double    val;
    //
    OSCCON = 0b01110000;     //クロックは8Mhz
    CMCON0 = 0b00000100;     //コンパレータを使用する。
    ANSEL  = 0b00000000;     // □□□変換は使用しない。
    TRISIO = 0b00001010;
    GPIO = 0b00000000;
    //
```

```
    T1CON.T1CKPS1 = 0;
    T1CON.T1CKPS0 = 0;
    //
    Pwm_Init(1000);
    Pwm_Change_DutyEx((PR2 * 4) / 2);
    Pwm_Stop();
    //
    Soft_Uart_Init(GPIO, 3, 5, 9600, 0);
    //
    VRCON.VREN = 1;
    VRCON.VRR = 1;
    //
    LED = 0;
    tm0 = 0;     //浮遊容量
    tm1 = 0;     //基準値
    tm1 = 0;     //実測値
    //動作モードの設定
    if (SW == 1) {
        mode = 1;     //静電容量スイッチ
    } else {
        while (SW == 0) {
            Delay_ms(50);
        }
        mode = 0;     //静電容量計
        Soft_Uart_Write_Str("C-Meter\r\n");
        //浮遊容量の測定
        tm0 = Average(100);
    }
    //基準値の測定
    if (mode == 0) {
        while (SW == 1) {
            LED = 1;
            Delay_ms(50);
            LED = 0;
            Delay_ms(50);
        }
    }
    tm1 = Average(100) - tm0;
    //
    while (1) {
        if (mode == 1) {
            on_cnt = 0;
            for (cnt = 0; cnt < 10; cnt++) {
                //実測値の測定
                tm2 = Average(10) - tm0;
                //基準値と実測値の差を求める。
                if (labs(tm1 - tm2) > (tm1 / 20)) {
                    on_cnt++;
                }
            }
            if (on_cnt > 5) {
```

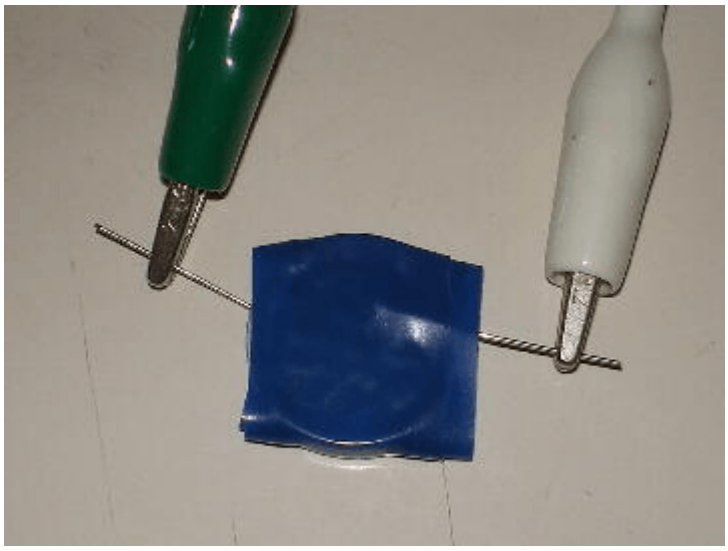
```
                LED = 1;
                Pwm_Start();
            } else {
                LED = 0;
                Pwm_Stop();
            }
        } else {
            //実測値の測定
            tm2 = Average(100) - tm0;
            val = (double)tm2 / ((double)tm1 / 1000.0);
            //
            WordToStr(val, buf);
            Soft_Uart_Write_Str(buf);
            Soft_Uart_Write_Str("pF\r\n");
            //
            LED = 1;
            Delay_ms(100);
            LED = 0;
        }
        //基準値を再設定する。
        if (SW == 0) {
            tm1 = tm2;
        }
    }
}

//*****************************************************************************
*
```

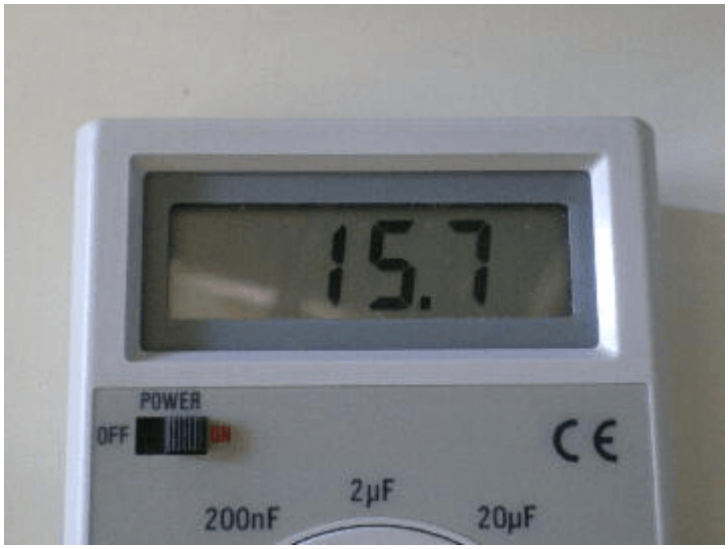


===== 動作確認 =====1円玉を2個使ったセンサーです。 2個の1円玉の間をビニールテープで絶縁し、リード線を挟み込んで、

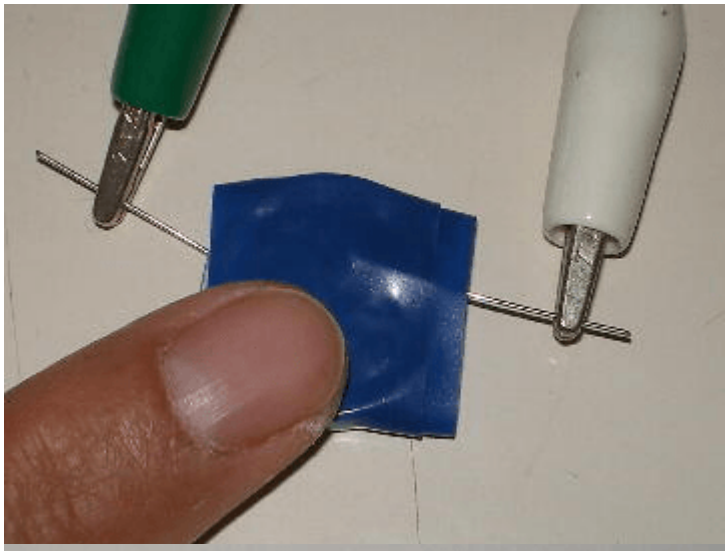
更に、全体をビニールテープで巻いて 固定します。

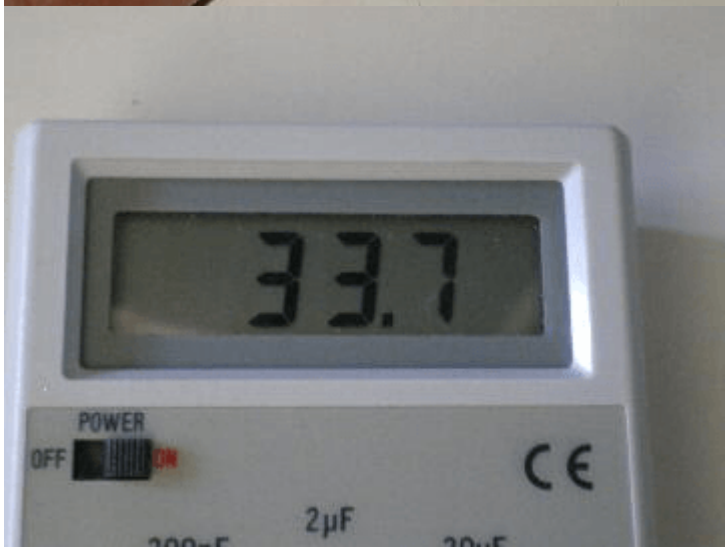
容量計で測定してみました。約16pFあります。





指で触れると約34pFに増えます。

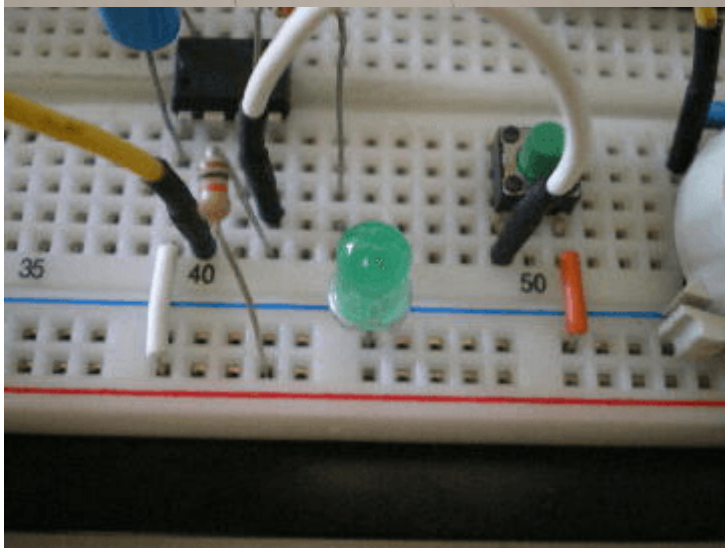




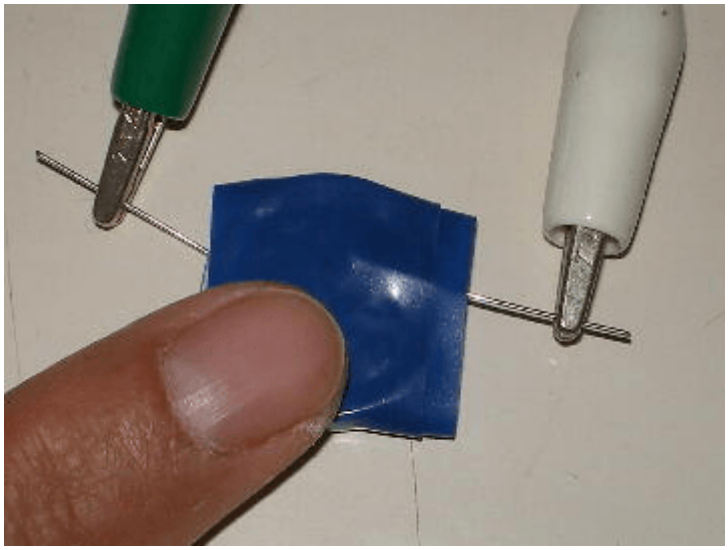
1円玉センサーを本回路の静電容量測定端子に接続します。 このままでは感度が良すぎる(2倍(200%)の変化)ので、並列に100pFのコンデンサを接続し、感度を下げ ます。 そうすることにより、約16%の変化になります。これは本回路の検出精度(5%)に十分な変化率となります。

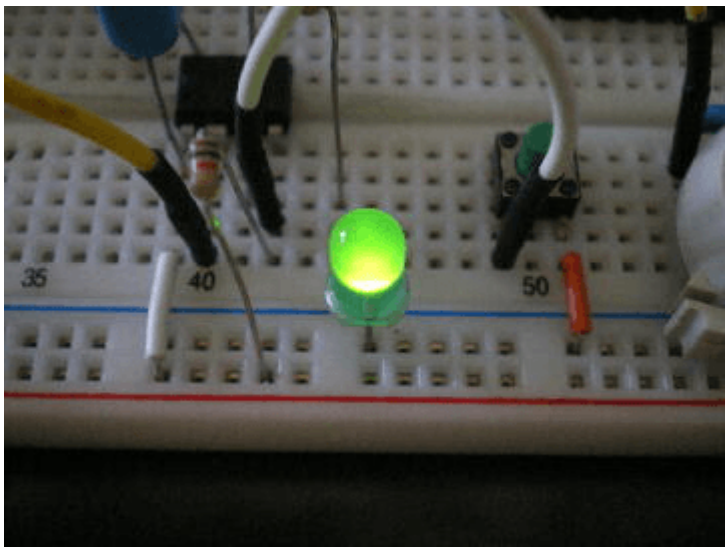
- 指で触れない時→約116pF
- 指で触れた時→約134pF

指で触れない時の容量をSWを押下することにより記憶させます。(この時はLEDはOFF状態です)

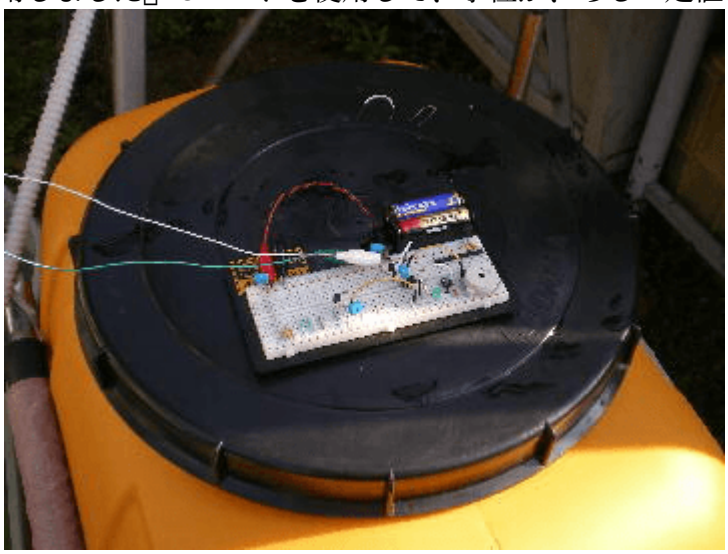


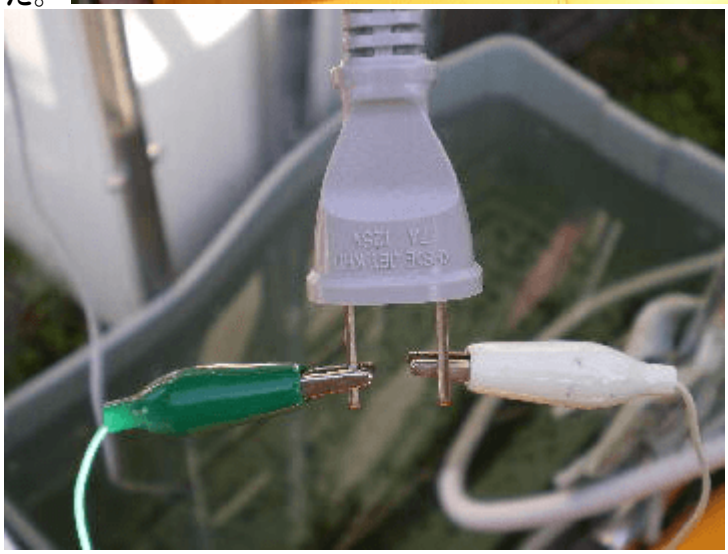
指で触れると、容量が変化し、LEDが点灯します。



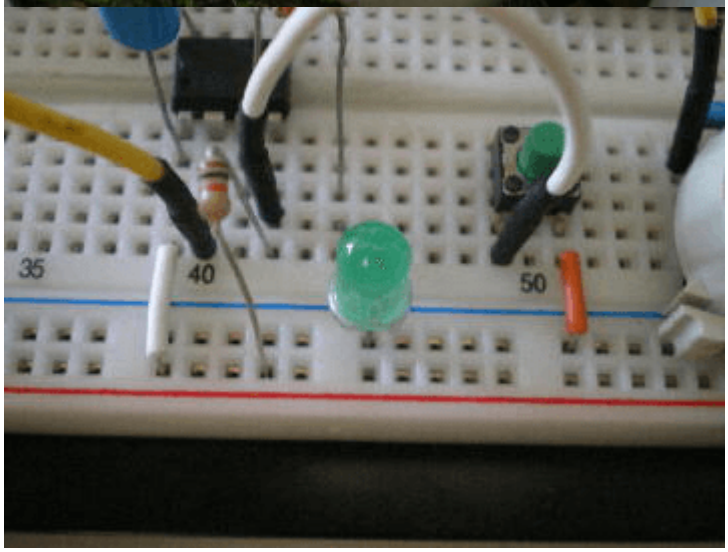


同様に「簡易水位センサー(静電容量方式)」でも使用しましたACコードを使用して、水位が、ある一定値になるとLEDが点灯することを確認しました。

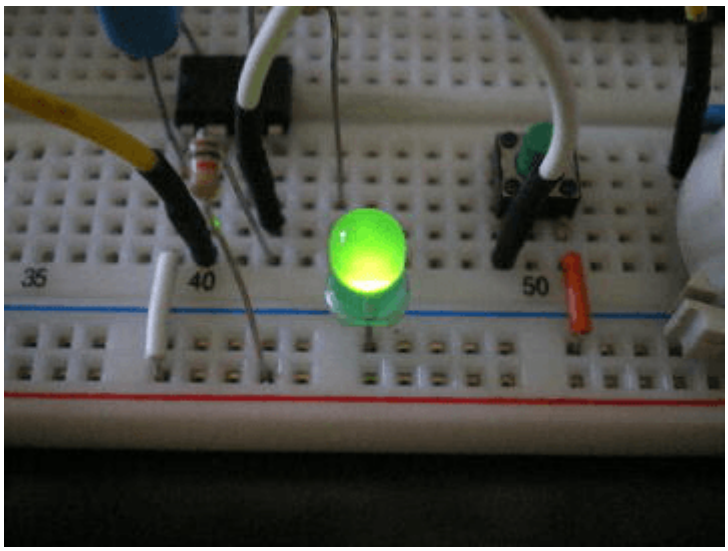




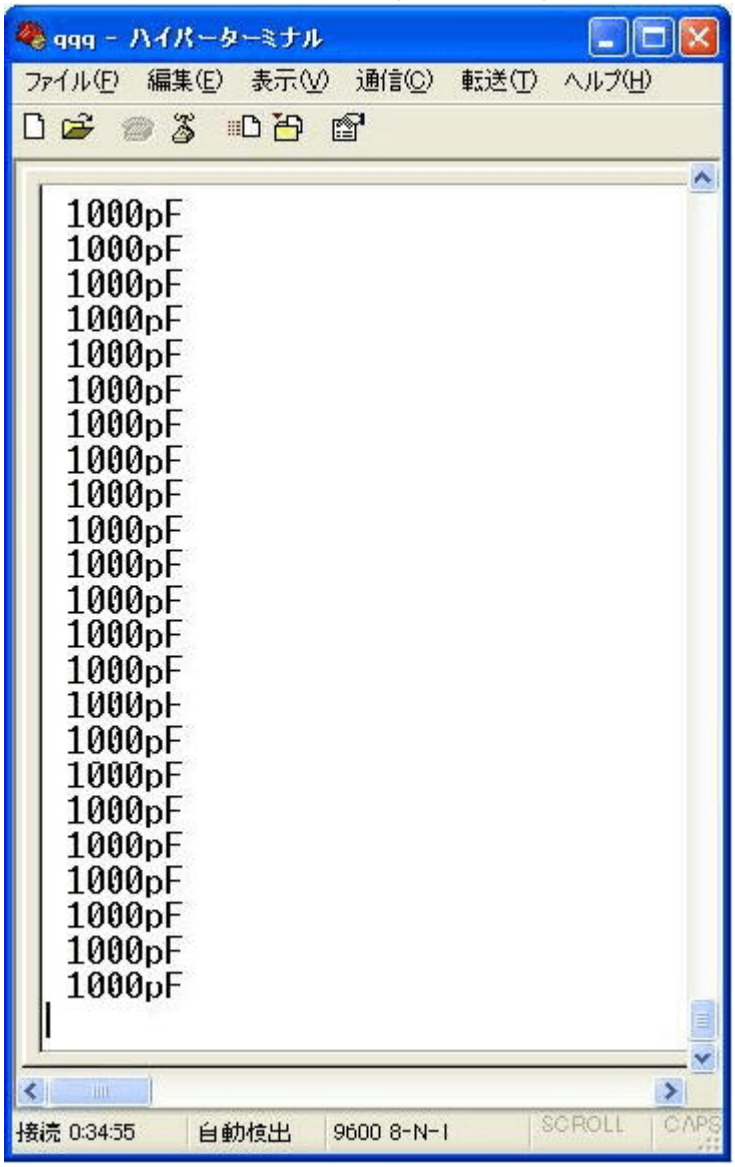
ACコードを水に浸さない時の容量をSWを押下することにより記憶させます。(この時はLEDはOFF状態です)

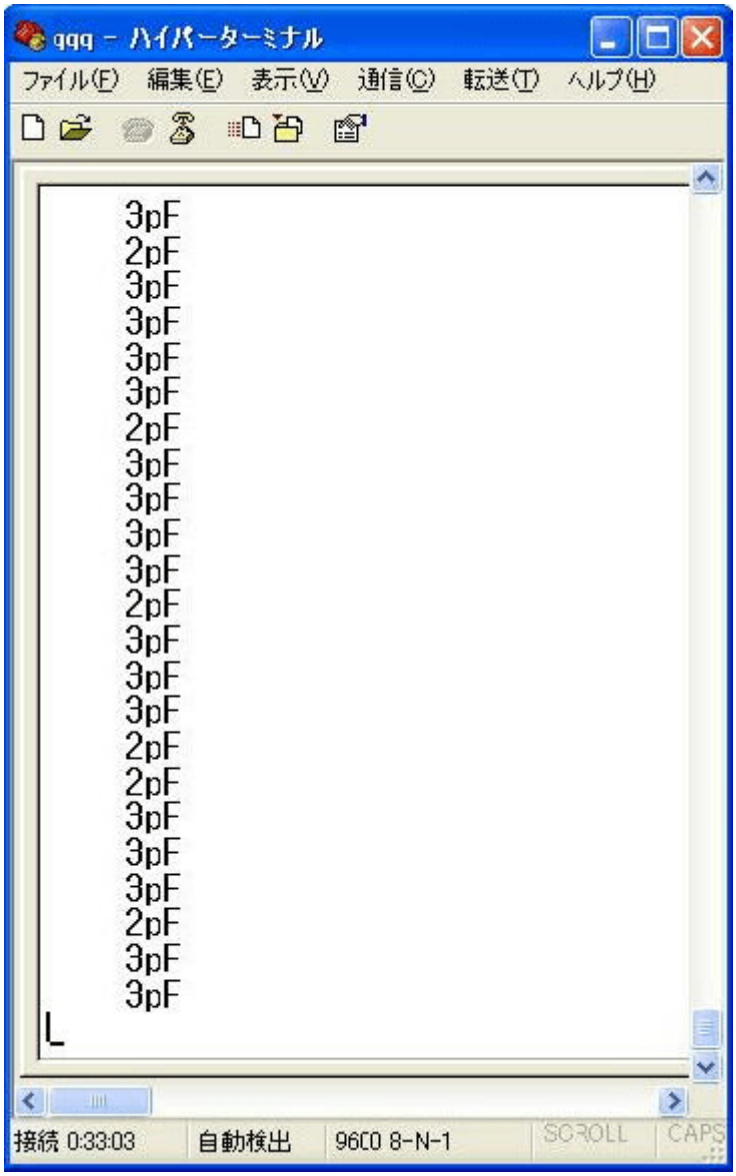


ACコードを水に浸す(約5cm~8cm)と、容量が変化し、LEDが点灯します。

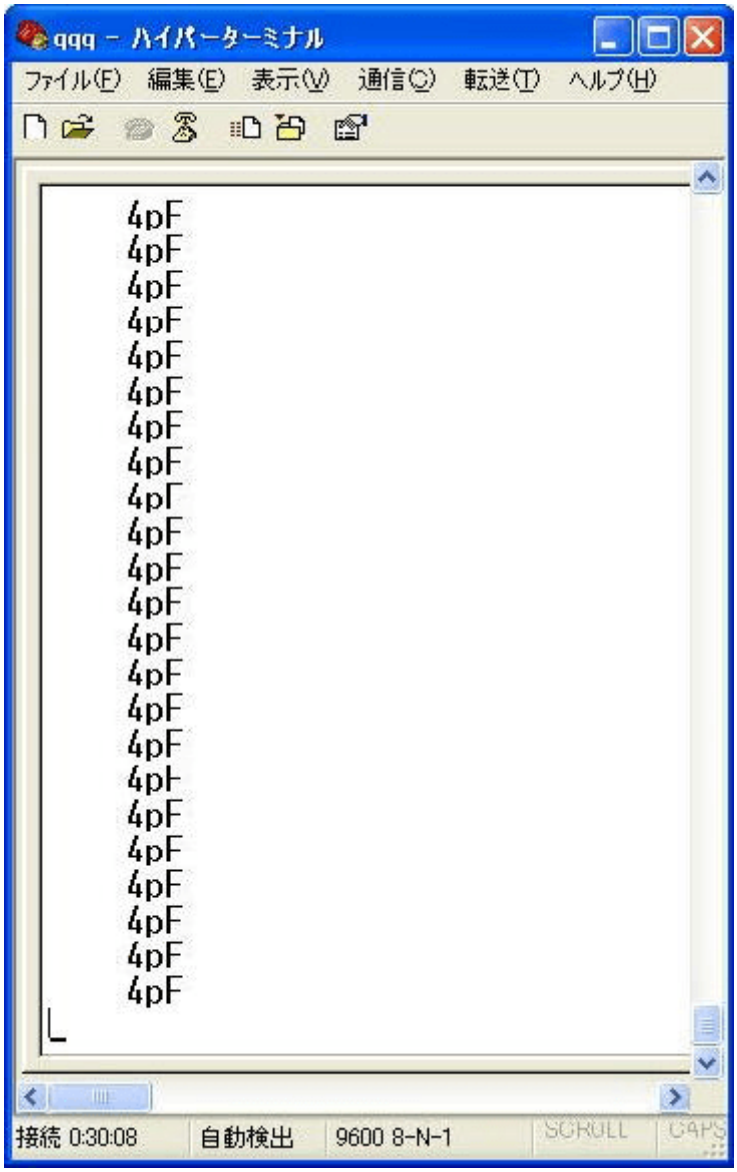


静電容量計の機能で、手持ちのコンデンサを測定してみました。 左側=1000pF□右側=3pFのコンデンサを測定した時の結果です。

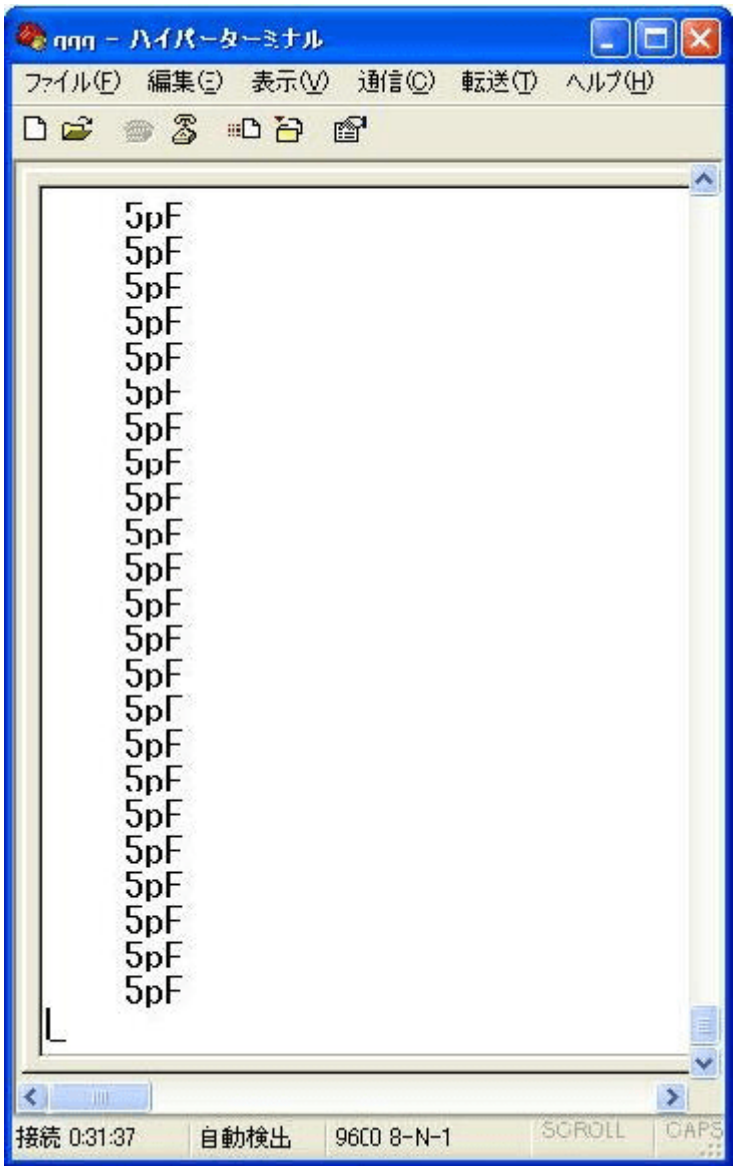




左側=4pF□右側=5pFのコンデンサを測定した


qqq - ハイパーターミナル
ファイル(F) 編集(E) 表示(V) 通信(C) 転送(T) ヘルプ(H)
4pF
4pF
4pF
4pF
4pF
4pF
4pF
4pF
4pF
4pF
4pF
4pF
4pF
4pF
4pF
4pF
4pF
4pF
4pF
4pF
4pF
接続 0:30:08
自動検出
9600 8-N-1
SCROLL
CAPS

時の結果です。

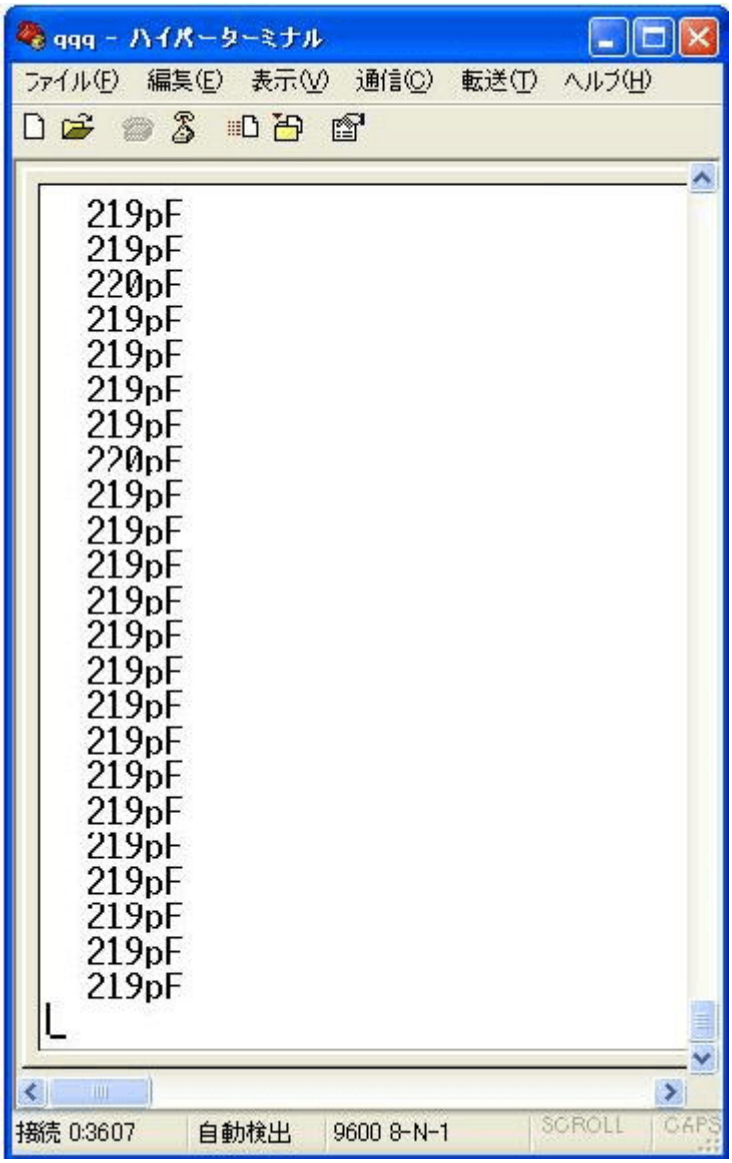


左側=27pF□右側=56pFのコンデンサを測定し

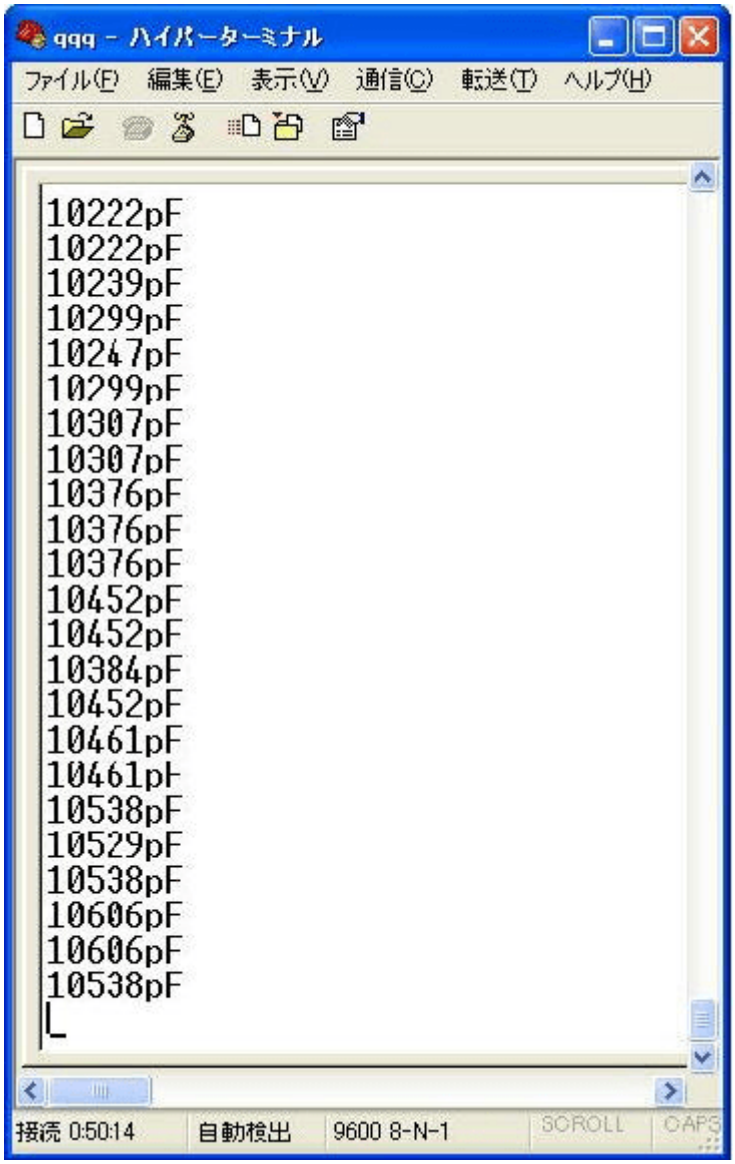

qqq - ハイパーターミナル
ファイル(F) 編集(E) 表示(V) 通信(C) 転送(T) ヘルプ(H)
26pF
26pF
26pF
26pF
26pF
26pF
26pF
26pF
26pF
26pF
26pF
26pF
26pF
26pF
26pF
26pF
26pF
26pF
26pF
26pF
接続 0:3752
自動検出
9600 8-N-1
SCROLL
CAPS

た時の結果です。

左側=220pF□右側=10000pF(0.01uF)のコンデ



ンサを測定した時の結果です。



如何ですか? 簡単な回路の割には、精度もそこそこに高い結果が得られます。 身の回りの静電容量に注力すると応用範囲も広がるのではないでしょうか?