

温度データロガーV4

概要

温度データの測定では、用途によっては、長時間(数日間~1ヶ月)の記録をとる場合があります。そこで今回は、出来るだけ、コンパクト、大容量、低消費電力を目指して、温度データロガーを製作してみました。

<仕様>

- 記録容量 約45日分(65536件)EEPROM(24LC1025)
- 電源 単三電池2本~3本
- 消費電流 約5mA
- 測定周期 1分
- 温度センサLM61CIZ
- 測定範囲 -30 ~+100
- 温度係数+10mV/°C
- 測定精度2.44mV(0.41°C)小数点第一位まで(例 “ 12.3 “)
- データ送信RS232C(9600bps)でパソコンに送信

動作原理

<動作モード> 動作モードには、書き込みモードと読み出しモードがあります。

- 書き込みモード スイッチ(SW1)をOFFの状態で、電源を入れます。
- 読み出しモード スイッチ(SW1)をONの状態で、電源を入れます。

<サンプリング時間(1分)> PIC12F683が内蔵しているCCPモジュールをコンペアモードで使用し、0.1秒の割り込みを発生させ、それを更に600回カウントすることにより、正確な1分のサンプリング時間を得ます。

<温度データの測定と記録>

- スイッチ(SW1)をクリックすることにより開始します。
- 温度センサ(LM61CIZ)の出力を、A/D変換で取り込みます。
- A/D変換の基準電圧には、シャントレギュレータ(NJM431L)を使用し、2.5Vを得ます。
- 5msec周期で、50回測定し、その平均値を求めます。
- 求めた値を、EEPROM(24LC1025)に逐次記録していきます。
- 記録を途中で停止する場合には、電源をOFFにします。

<温度データのパソコンへの転送>

- スイッチ(SW1)をクリックすることにより開始します。
- 24LC1025に記録されたデータを、順次読み出します。
- 読み出した値は、電圧値なので、温度値に換算します。(小数点第一位まで)
- 換算した値を、RS232C経由でパソコンに転送します。(速度は、9600bps)
- 転送を途中で停止する場合には、電源をOFFにします。

<パソコン上の操作> Windows付属の通信ソフト(ハイパーターミナル)で、データを受信し、ファイルに

回路図



```

//*****
*
```

```

/*
  < 簡易温度データログ
  1分周期のサンプリングで、約45日間の測定が可能です。
  EEPROMの容量が、131072バイト
  ・ データ1件のサイズは、2バイト
  ・ サンプリング周期が1分
  ・ これより、次式で求めます。
    45日 (131072バイト ÷ 2バイト) ÷ 60分 ÷ 24時間
*/
// *****
*

#define SW1 GPIO.F3
#define SW2 GPIO.F2

#define LED GPIO.F2

#define ON 0
#define OFF 1

#define ACK 1
#define NO_ACK 0

#define WRITE_CYCLE_TIME 5

#define DATA_SIZE_MAX 0x20000

#define SAMPLING_TIME 600

// *****
*

void Delay()
{
    Delay_ms(WRITE_CYCLE_TIME);
}

// *****
*

static unsigned short cnt;

void EEPROM_24LC1025_Page_Write(unsigned long addr, unsigned short
*buf, unsigned short len)
{
    Soft_I2C_Start();
    if ((addr & 0x10000) == 0)
        Soft_I2C_Write(0xA0);
    else
        Soft_I2C_Write(0xA8);
    Soft_I2C_Write((addr >> 8) & 0xFF);
    Soft_I2C_Write(addr & 0xFF);
}

```

```
    for (cnt = 0; cnt < len; cnt++) {
        Soft_I2C_Write(buf[cnt]);
    }
    Soft_I2C_Stop();
    //
    Delay();
}

//*****
*

void    EEPROM_24LC1025_Page_Read(unsigned long addr, unsigned short
*buf, unsigned short len)
{
    Soft_I2C_Start();
    if ((addr & 0x10000) == 0)
        Soft_I2C_Write(0xA0);
    else
        Soft_I2C_Write(0xA8);
    Soft_I2C_Write((addr >> 8) & 0xFF);
    Soft_I2C_Write(addr & 0xFF);
    Soft_I2C_Start();
    if ((addr & 0x10000) == 0)
        Soft_I2C_Write(0xA1);
    else
        Soft_I2C_Write(0xA9);
    for (cnt = 0; cnt < (len - 1); cnt++) {
        buf[cnt] = Soft_I2C_Read(ACK);
    }
    buf[cnt] = Soft_I2C_Read(N0_ACK);
    Soft_I2C_Stop();
    //
    Delay();
}

//*****
*

void    SwitchONcheck()
{
    while (Button(&GPIO, 3, 1, 0) == 0)
        ;
    while (Button(&GPIO, 3, 1, 1) == 0)
        ;
}

//*****
*

static unsigned    int        counter;
```

```
void interrupt()
{
    //0.1sec cycle
    PIR1.CCP1IF = 0;
    //
    if (counter > 0)
        counter--;
}

//*****
*

void Soft_Uart_Write_Str(unsigned short *data)
{
    while (*data != 0x00) {
        Soft_Uart_Write(*data);
        data++;
    }
}

//*****
*

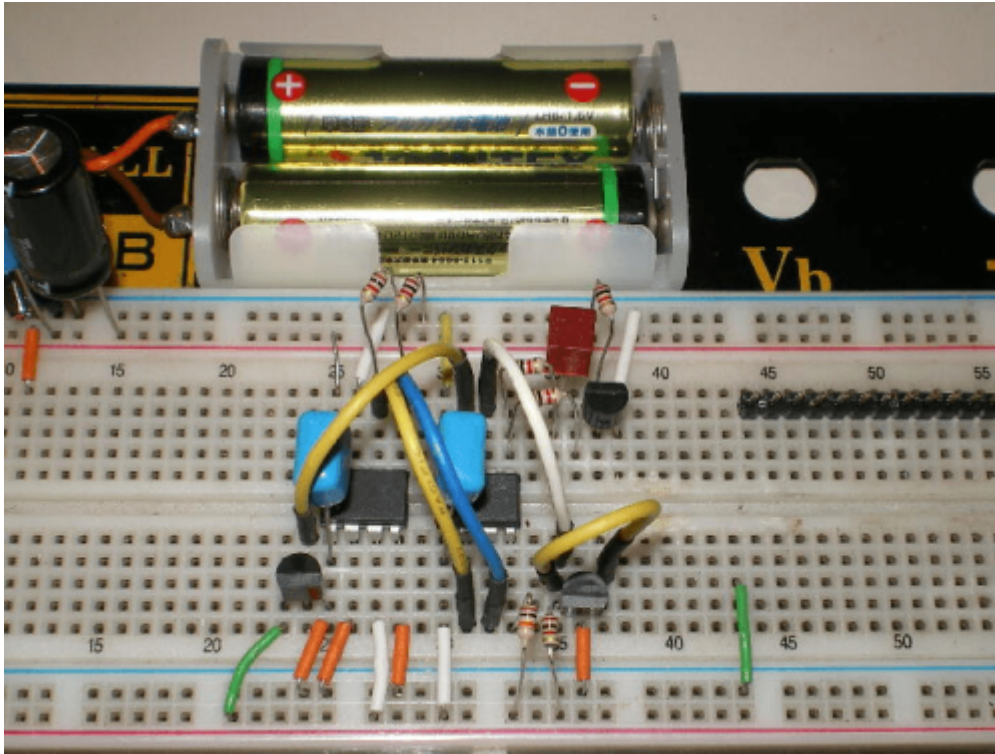
void main()
{
    static unsigned char buf[6], mode;
    static unsigned long addr;
    static unsigned int ad;
    static double thermo;
    //
    OSCCON = 0b01100000;
    CMCON0 = 0b00000111;
    ANSEL = 0b00000011;
    ADCON0.VCFG = 1;
    TRISIO = 0b00001111;
    // CCPの設定
    PIE1.CCP1IE = 1;
    PIR1.CCP1IF = 0;
    CCP1CON = 0b00001011;
    CCPR1L = 0xD4; // 0.1sec...10hz...クロックが4Mhzの時
    CCPR1H = 0x30; // 0.1sec...(1÷4000000)*4*8*12500
    // TIMER1の設定
    PIE1.TMR1IE = 0;
    PIR1.TMR1IF = 0;
    TMR1L = 0;
    TMR1H = 0;
    T1CON.T1CKPS0 = 1;
    T1CON.T1CKPS1 = 1;
    T1CON.TMR1ON = 1;
    //動作モードを決定する。
    LED = OFF;
```

```
mode = 0;
if (SW1 == 0) {
    mode = 1;
    while (SW1 == 0) {
        Delay();
    }
}
//
Soft_Uart_Init(GPIO, 3, 2, 9600, 0);
Soft_I2C_Config(&GPIO, 4, 5);
//
for (cnt = 0; cnt < 5; cnt++) {
    LED = ON;
    Delay_ms(100);
    LED = OFF;
    Delay_ms(100);
}
//
while (1) {
    if (mode == 0) {    //書き込み
        INTCON.PEIE = 1;
        INTCON.GIE = 1;
        //
        SwitchONcheck();
        addr = 0;
        while (1) {
            counter = SAMPLING_TIME;    //サンプリング時間を設定する。
            //
            ad = 0;
            for (cnt = 0; cnt < 50; cnt++) {
                ad += Adc_Read(0);
                Delay();
            }
            ad = ad / 50;
            buf[0] = ad & 0xFF;
            buf[1] = (ad >> 8) & 0xFF;
            EEPROM_24LC1025_Page_Write(addr, buf, 2);
            addr += 2;
            if (addr >= DATA_SIZE_MAX)
                break;
            //
            LED = ON;
            Delay_ms(100);
            LED = OFF;
            //
            while (counter != 0)
                ;
        }
    } else {    //読み出し
        INTCON.PEIE = 0;
        INTCON.GIE = 0;
```

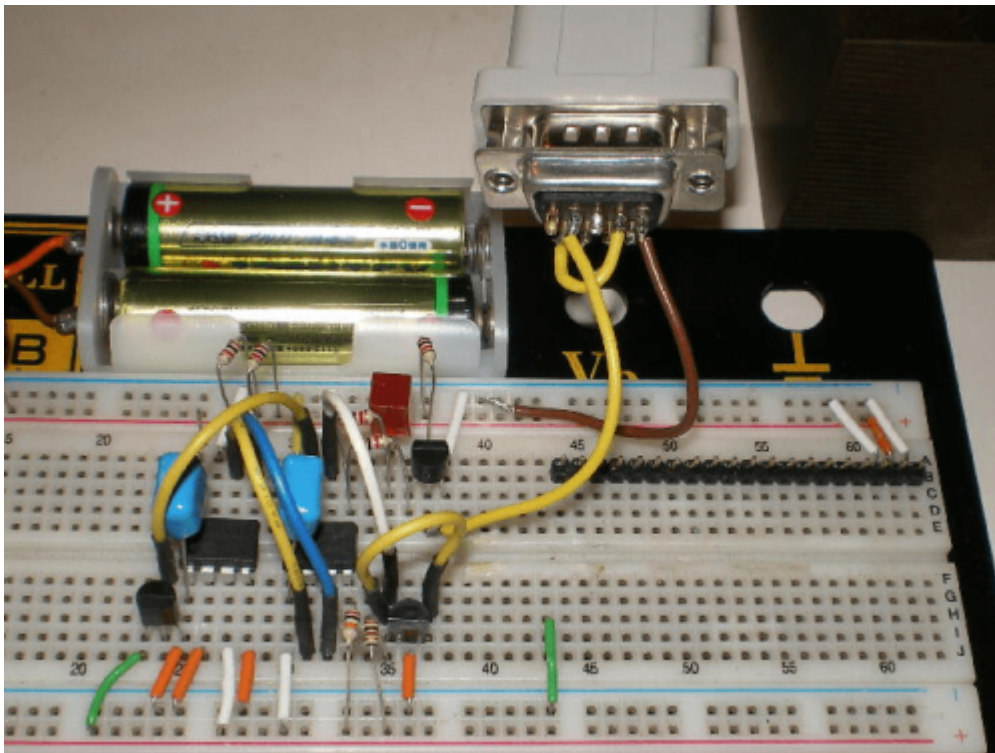
```
//
SwitchONcheck();
addr = 0;
while (1) {
    EEPROM_24LC1025_Page_Read(addr, buf, 2);
    ad = (buf[1] << 8) + buf[0];
    thermo = (double)ad * 2.44140625;
    thermo = ((thermo - 300.0) / 10.0) - 30.0;
    WordToStr(thermo * 10.0, buf);
    buf[0] = buf[1];
    buf[1] = buf[2];
    buf[2] = buf[3];
    buf[3] = '.';
    Soft_Uart_Write_Str(buf);
    Soft_Uart_Write_Str("\r\n");
    addr += 2;
    if (addr >= DATA_SIZE_MAX)
        break;
}
}
}
}
}
//*****
*
```

動作確認

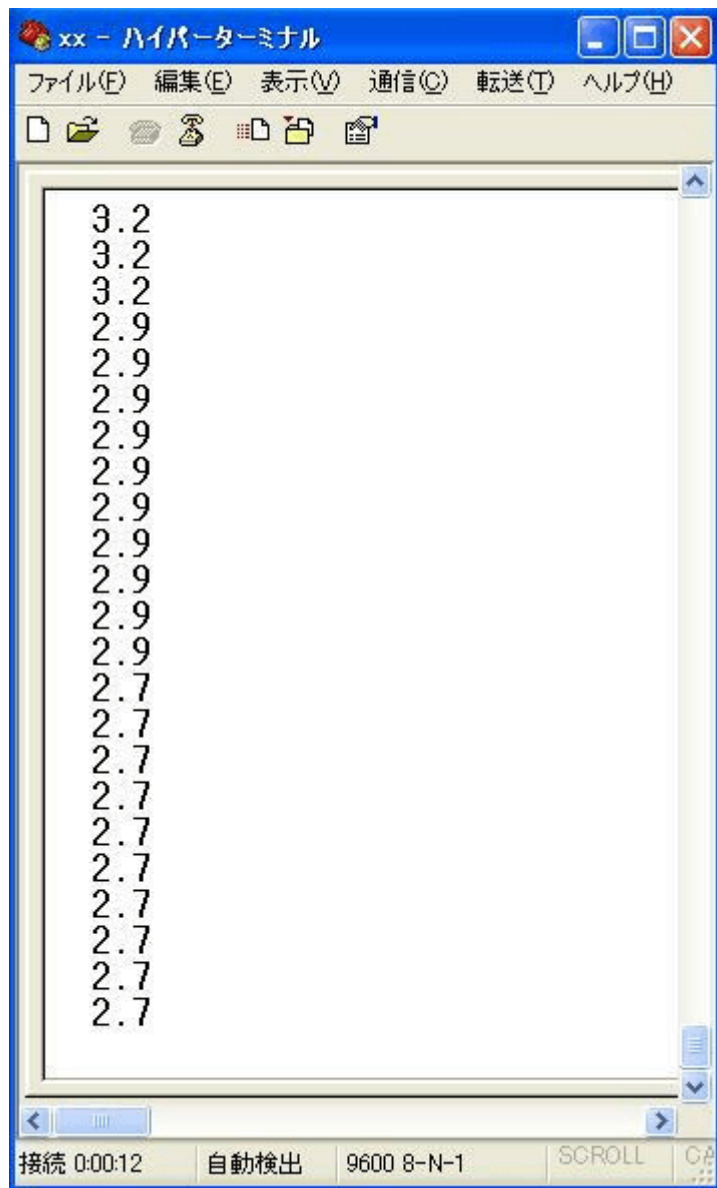
ブレッドボードに実装して、温度を記録してみました。記録した場所は作業部屋です。部品が少ないので、蛇の目基板に実装すれば、かなりコンパクトに仕上がると思います。



記録した温度データを、RS232C経由で、パソコンに転送します。

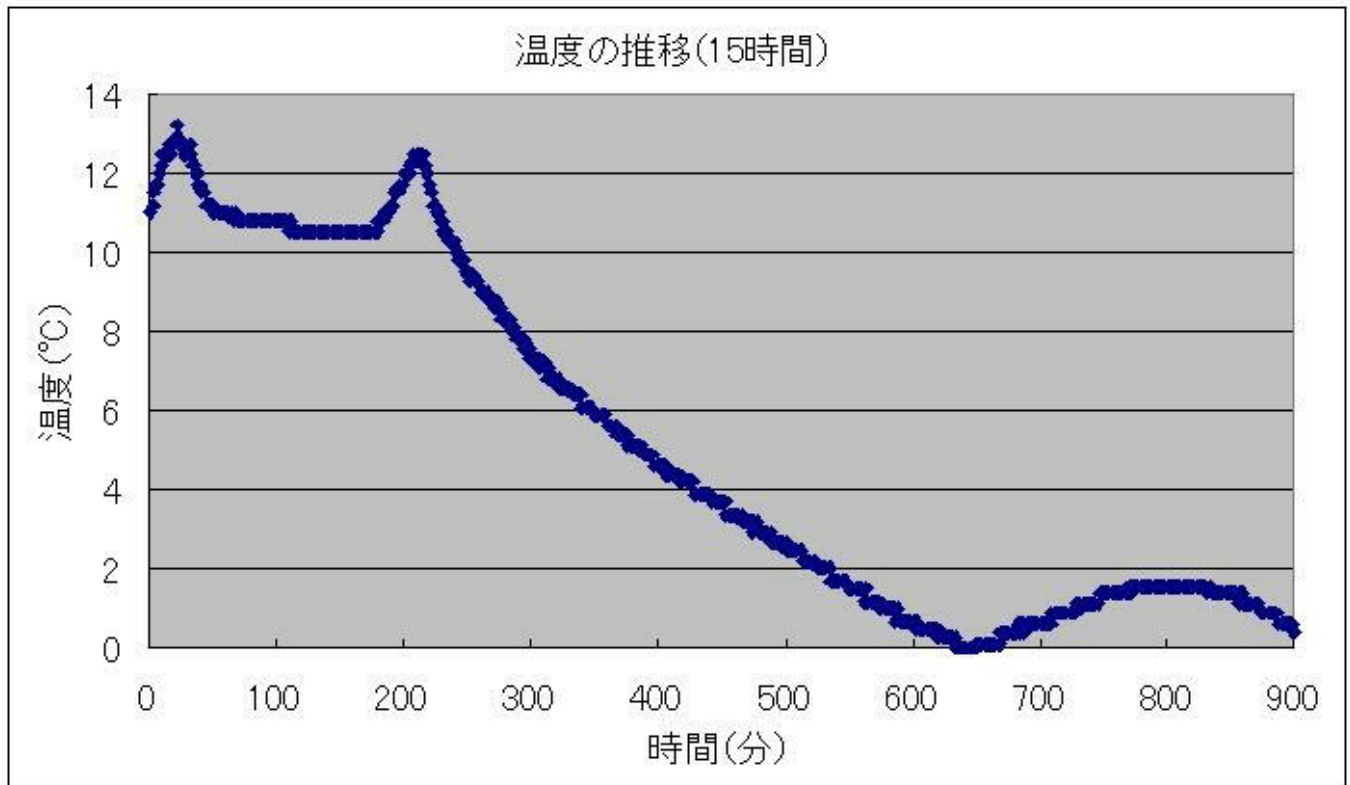


パソコン付属の通信ソフト(ハイパーターミナル)を使用して、データを吸い上げ、ファイルに保存しま



す。

保存したデータを、表計算ソフト(Excel)に取り込んで、グラフ表示させて見ました。記録経過後200分~650分にかけて、温度が徐々に下がっているのは、作業部屋で作業を終えて、ストーブや電気を消して母屋に引き上げたためです。(夜中の1時~8時30分)



消費電流が、約5mAなので、使用する電池によって、連続記録時間が左右されます。ニッケル水素電池(1.2V 1000mAh×3本)の場合、約8日($(1000\text{mAh} \div 5\text{mA}) \div 24\text{時間}$)になります。



リチウム電池(CR2032 3V 210mAh 1本)の場合、約1日半($(210\text{mAh} \div 5\text{mA}) \div 24\text{時間}$)になります。



45日連続して記録する場合には5500mAh以上の電池が必要になります。

著作権表示 copyright notice

このページは稲崎様の閉鎖したHPのコピーで、著作権は稲崎様にあります。[詳細](#) This page is a copy of Mr. Inasaki's closed website, and the copyright is held by him.[Details](#)

From:

<http://deepsky.jp/wiki/> - うごくといいな

Permanent link:

<http://deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic12f683:30>

Last update: **2020/05/01 20:37**

