平成 24 年度

卒業研究報告書

研究題目

多機能気象観測装置の製作

富山高等専門学校
電子制御工学科　立川佑亮
指導教官　由井四海

平成 25 年 3 月 1 日　提出

目次

１．序論

１.１　はじめに

現在、地球規模の大気汚染、地球温暖化、砂漠化などの環境問題が注目されている。これらの問題を解決するためにはまず、どの程度環境が悪くなっているのかを調査する必要がある。

１.２　目的

環境を調査するめには世界各地へ多数の観測装置を設置する必要がある。しかし観測装置は高価であり、コストが高くなってしまう。

そこで本研究では、一般的なセンサとマイコンボードを用いた気象観測装置の製作を目的とした。

これの気象観測装置で観測できるものとしては、緯度・経度・日付・方位・高度・気圧・湿度・気温を目標とし製作する。

２.マイコンボード

２.１　Arduino

装置にはマイコンボードとして、Arduino　Uno を用いた。Arduino は ATmel AVR マイクロコントローラを使用しており、クロック・電圧変換 IC・USB インターフェース IC などが基板上に実装されています。このため、簡単な動作をさせるだけなら外付けの部品も最小限ですみます。外部のインターフェースとしては、アナロ入力ポート×6 本、デジタル入出力ポート×14 本がある。

２.２　Arduino IDE

Arduino のソフトウェアの開発環境はクロスプラットフォームの Java アプリケーションである。その内部では C 言語のコンパイラーgcc やアップロードプログラム avrdude が使用されている。プログラミング言語は C 言語風の構文である。図２.２のプログラムは LED を点滅させるものであり、Arduino 言語では、このような簡単な記述をするだけでよい。

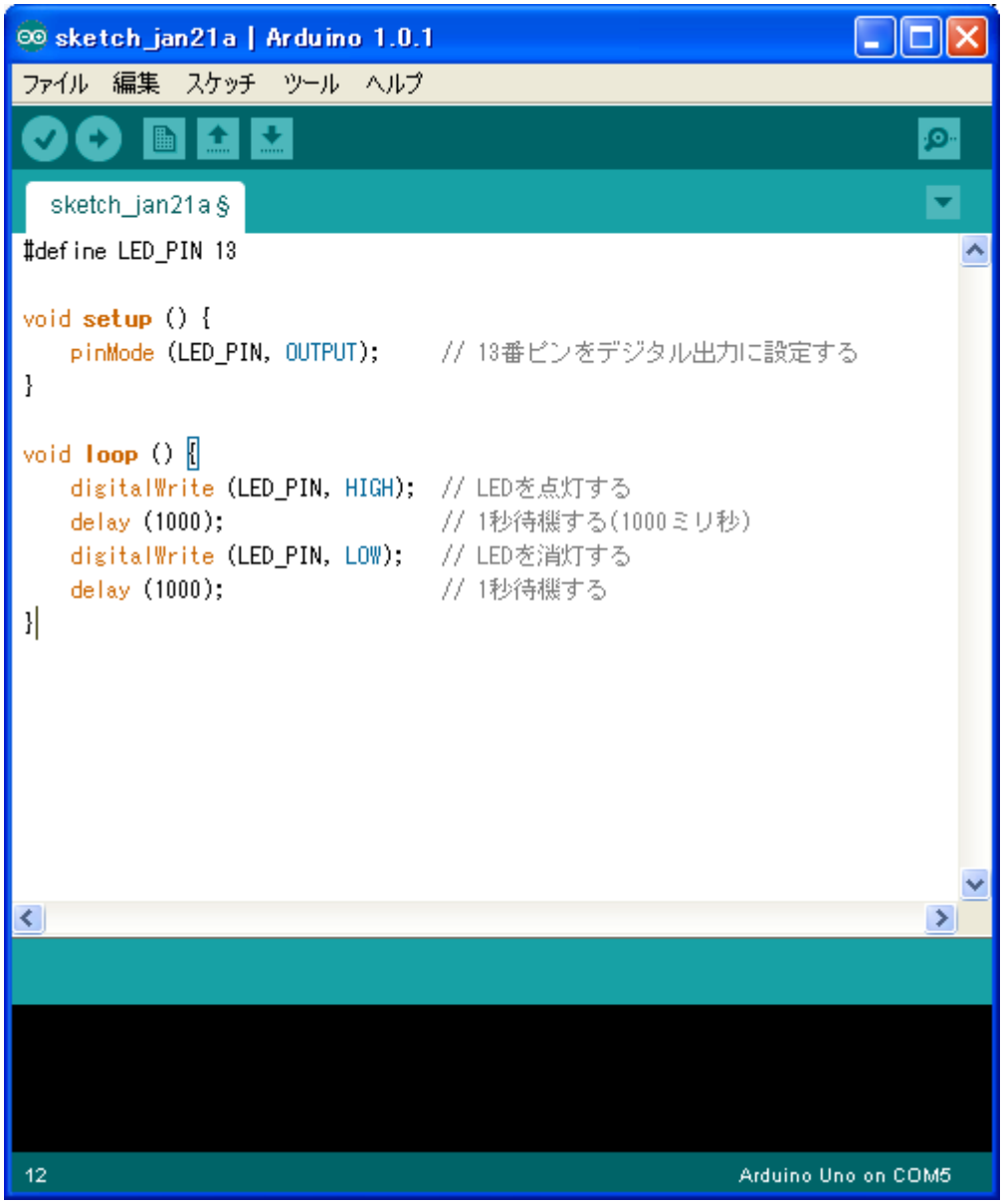

図２.２　LED を点滅させるためのプログラム

２.３　Arduino と PC の接続方法

Arduino と PC 間の通信はシリアル通信のプロトコルがベースとなっています。AVR や PIC などの一般的なマイコンはシリアル通信機能が搭載されていることからプロコトルが簡単に組めるというメリットがあります。このシリアル通信機能はソースプログラムをコンパイルして実行プログラムを Arduino にアップロードする際や、実行時に PC やその他の機器との間で通信を行うときに使われます。Arduino Uno には，USB(B)のコネクタが接続されているので，PC との接続には AB タイプのケーブルが必要となります。

３.大気圧センサ

３.１　使用した大気圧センサ

表３.１.１に大気圧センサの仕様を示す。このセンサは通信方法として I2C インターフェースを利用しています。I2C インターフェースはデータとクロックの 2 線で接続するインターフェースです。接続するデバイスは、１つのマスターと１つ以上のスレーブです。マスターデバイスからスレーブデバイスに対して指示を出し、命令を実行します。スレーブデバイスの選択はアドレスを指定することで決定されます。つまり、マイコンはマスターとして動作し、センサに対してコマンドを送ることで、大気圧を測定し、このデータを取得することができます。

大気圧データや校正用データはセンサのメモリ上に書き込まれます。大気圧は 10 ビットの数値で表されます。校正用データは 96 ビットです。表３.１.２に大気圧センサのメモリマップを示す。10 ビットとなっているものは、16 ビットのうち上位 10 ビットが有効で下位 6 ビットが 0 に設定されている。表３.１.３に大気圧センサで利用可能なコマンドを示す。変換を開始するには、コマンドコードに続けて 0x01 を送信する必要があります。

表３.１.１　大気圧センサの仕様

項目	内容
電源電圧(Vdd)	2.375V～5.5V(標準 3.3V)
測定範囲	50～115kPa(500～1150hPa)
分解能	0.15kPa(1.5hPa)
精度	±1kPa(±10hPa)
通信方式	I2C インターフェース
メーカー	Freescale Semiconductor 社
型番	MPL115A2

表３.１.２　大気圧センサのメモリマップ

アドレス	内容	サイズ(ビット)
0x00-0x01	大気圧データ	10
0x04-0x0f	校正用データ	96

表３.１.３　大気圧センサで利用可能なコマンド

コマンド	コマンドコード
大気圧変換開始	X0010000
大気圧データの MSB 読み取り	X0000000
大気圧データの LSB 読み取り	X0000001
校正用データ読み取り	X0000100

３.２　プログラム

図３.２.１に大気圧センサのフローチャートを示す。連続で動作させるため、校正データは最初の 1 回だけを取得し、その後大気圧データの取得と大気圧の計算の繰り返しとした。また、このセンサは I2C プロトコルを利用するため、Arduino に標準提供されている Wire ライブラリを利用した。表３.２.１に GPS センサの各ピンを示す。7 番と 8 番は，4.7kΩ の抵抗で Vdd に接続する必要がある。

表３.２.１　GPS センサの各ピン

ピン番号	内容
1	Vdd
2	外部キャパシタ
3	GND
4	シャットダウン
5	RST
6	NC
7	データ
8	クロック入力

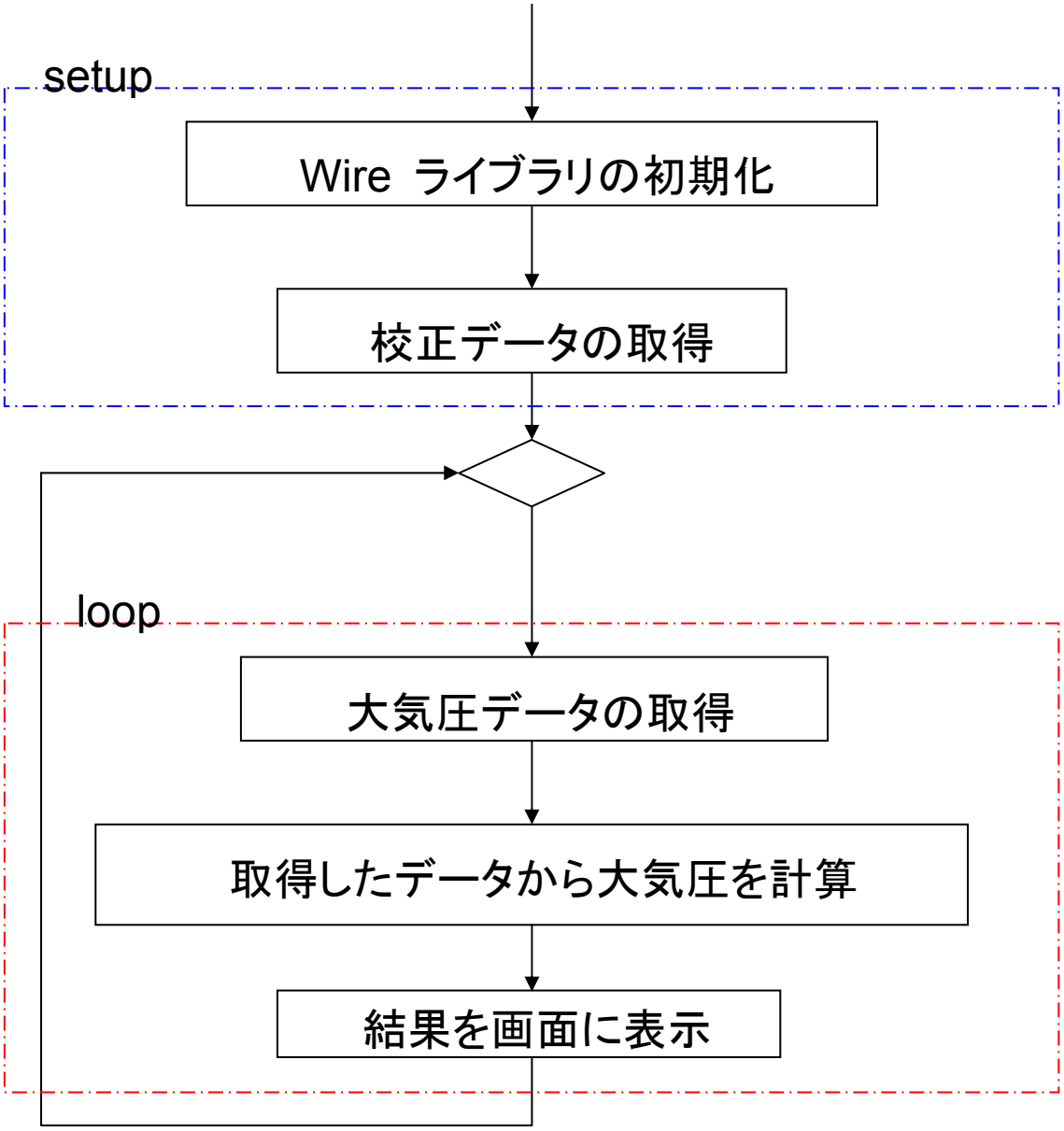

図３.２.１　大気圧センサのフローチャート

３.３　動作確認結果

図３.３.１に大気圧センサの測定結果１を示す。図３.３.２に大気圧センサの測定結果２を示す。2012 年 9 月 12 日 AM1:00～2012 年 9 月 13 日 AM1:00 の 24 時間、1 分刻みで計測を行った。図３.３.１は計測データ、図３.３.２は計測データを 1 時間の平均を算出し、気象庁の伏木のデータと比較したものである。計測し始めたときに約 1.5hPa の差が出力されたが、計測が終わるころには約 0.5hPa の差が出力された。差が変化した要因としては、測定場所が富山高専射水キャンパスと伏木とは距離があったこと、また、室内での測定だったためだと考えられる。

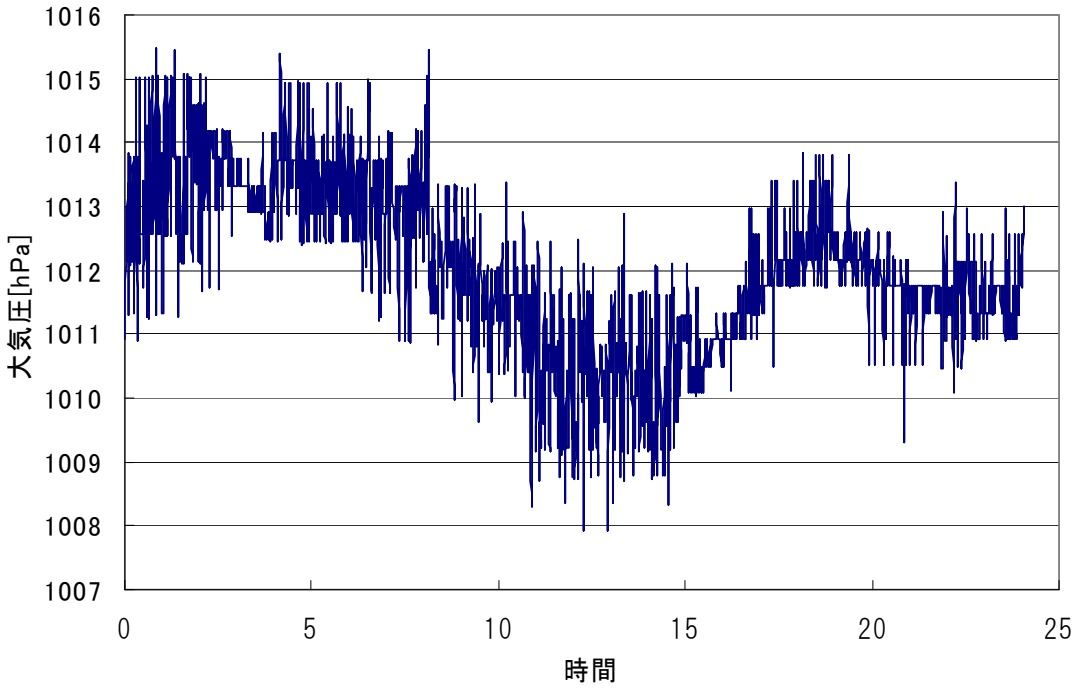

図３.３.１　大気圧センサの測定結果１

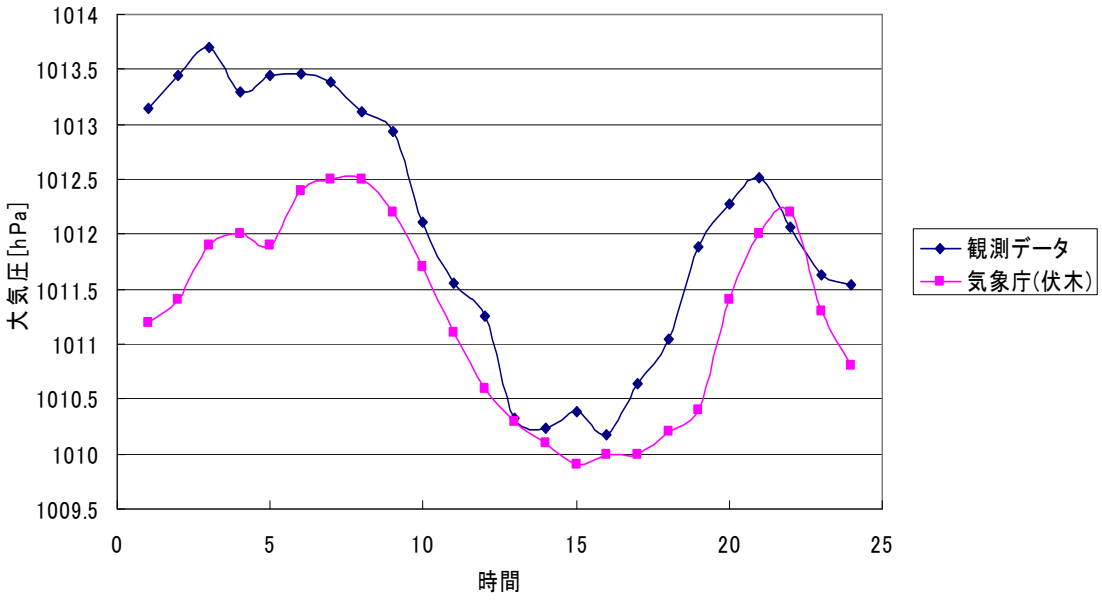

図３.３.２　大気圧センサの測定結果２

４.GPS センサ

４.１　使用した GPS センサ

表４.１.１に GPS センサの仕様を示す。表４.１.２に GPS センサのピン番号を示す。5 番ピン,6 番ピンはデータシート上では NC となっていますが、調べた結果 TTL レベルの出力が出ていた為 Arduino に直接接続することができる。そのため 3 番ピン,4 番ピンではなく、5 番ピン,6 番ピンを使用した。

表４.１.１　GPS センサの仕様

項目	内容
電源電圧(Vdd)	3.8V～8.0V
受信チャンネル	65ch
位置精度	5m
測定範囲	高度 0～18000m　速度 0～515m/s
コールドスタート	約 30 秒
ホットスタート	約 1 秒
動作温度	－40℃～85℃
通信方式	RS-232 シリアル通信
メーカー	CanMore Electronics 社
型番	GT-720F

表４.１.２　GPS センサのピン番号

ピン番号	内容
1	Ground
2	Power
3	Serial Data In 2(RS-232C レベル)
4	Serial Data Out 2(RS-232C レベル)
5	NC(実際は TTL In)
6	NC(実際は TTL Out)

４.２　実験その１

まず、GPS センサがどのようなデータを出力しているのかを調べた。表４.２.１のように接続した。電源を接続し、GPS センサの TTL 出力を Arduino のデジタルの 0 番ピンに接続した。Arduino のデジタルの 0 番ピンはシリアル通信の受信用のピンとなっています。また、シリアルコンソールに文字を表示するため、USB ケーブルで PC と接続した。

表４.２.１　接続方法

GPS センサのピン番号	Arduino のピン番号
1	GND
2	5V
3	接続しない
4	接続しない
5	接続しない
6	デジタル 0(RX)

以下のプログラムを動かすと、GPS センサからデータがシリアルモニタに送信されます。

```
void setup () {
Serial.begin(9600);
}

void loop () {
if (Serial.available()) {
Serial.write(Serial.read());
}
}
```

プログラムをアップロードする際に、Arduino の 0 番ピンと GPS センサとの接続を切っておく必要があり、それはデジタルピンの 0 番と 1 番は PC とのシリアル通信にも利用しているからである。接続したままだとプログラムのアップロードに失敗するため、プログラムをアップロード後 Arduino の 0 番ピンと GPS センサの 6 番ピンとを接続した。

GPS センサから出力された情報を以下に示す。

```
$GPRMC,045052.440,A,3645.4663,N,13709.5606,E,000.0,130.5,080213,,,A*6D
$GPGGA,045052.440,3645.4663,N,13709.5606,E,1,04,3.4,69.7,M,37.2,M,,0000*6D
$GPGSA,A,3,24,09,02,15,,,,,,,,,4.7,3.4,3.2*3E
$GPGSV,3,1,12,15,64,292,15,09,61,165,33,26,59,030,,05,55,095,15*79
$GPGSV,3,2,12,24,29,196,31,21,29,309,,08,19,040,,02,12,164,24*78
$GPGSV,3,3,12,28,11,079,,29,08,245,,18,08,301,,10,04,113,16*7C
$GPVTG,130.5,T,,M,000.0,N,000.0,K,A*0A
```

この出力された情報は NMEA-0183 フォーマットというもので、GGA には主に測位時

刻、緯度、経度、受信栄整数、アンテナ高度、WGS-84 楕円体から平均海水面の高度差が入っており、GSV には通信した衛星の番号とその衛星との仰角、方位、SNR、RMC には測位時刻、緯度、経度、対地速度、進行方向が入っている。今回の製作する多機能気象観測装置では、GPS センサにより得たい情報は緯度、経度、年月日、方位、高度なので必要なのは GPGGA と GPRMC である。

４.３　実験その２

４.３.１　プログラムの流れ

図４.３.１に GPS センサのフロチャートを示す。GPS センサが出力する緯度・経度の情報は 60 進数の数値で出力されている。出力された緯度・経度を google マップで確認するため、10 進数法に変換し、画面に出力する形とした。

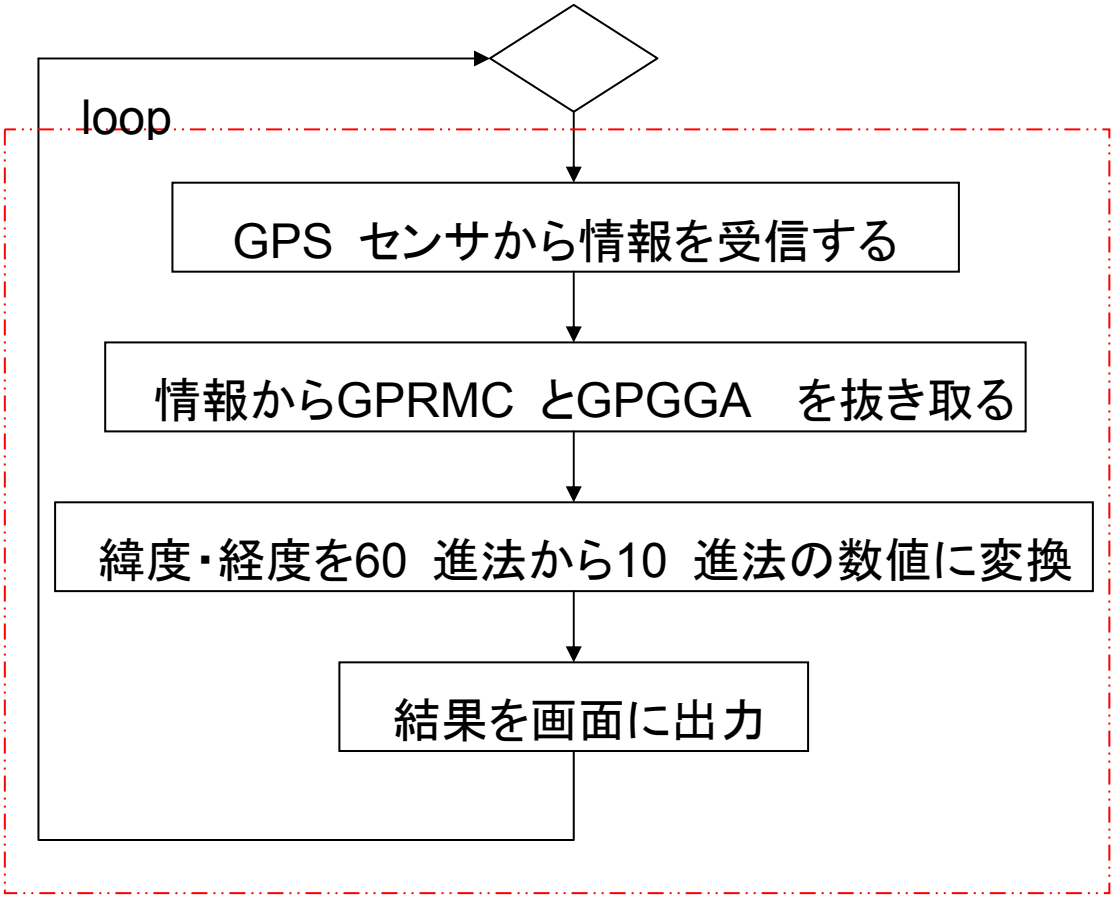

図４.３.１　GPS センサのフロチャート

４.３.２　データ保持について

大気圧センサでの測定のときは、Arduino と PC を USB 接続し PC にデータを保存していたが GPS センサでの測定は車で走行し測定をするため、別のデータ保持の方法をとる必要がある。そこで今回は、マイクロ SD シールドを用いてデータ保持を行なった。このシールドは 3.3V 動作だが電圧変換(3.3<->5V)IC が搭載されているため、このまま使用できる。また、このシールドはピン番号８・１０・１１・１２・１３を使用するので他には使用できない。

４.３.３　結果

・GPRMC

図４.３.２　GPRMC の値を google マップに表示させた結果を示す。緯度・経度に関しては、走行ルートからそれることなく、制度の 5m 以内のデータが出力された。時刻については、GPS センサから出力される時刻は協定世界時間となっていたため、日本標準時間に合わせるため 9 時間進め保存するように設定した。

図４.３.２　GPRMC の値を google マップに表示させた結果

・　GPGGA

表４.３.１に GPGGA の結果を示す。GPGGA に入っている情報は、ジオイド高と WGS－84 楕円体高である。まず標高というのは WGS-84 楕円体高からジオイド高を引いたもので表される。しかし、GPS センサから出力されるジオイド高は、同時に観測した緯度・経

度によりその周辺付近の大まかなジオイド高が出力されることが調べていくうちにわかった。その出力されるジオイド高は GPRMC で出力された緯度・経度から、求めたジオイド高とは約 1m 差があったため、測定では GPRMC で出力された緯度・経度から、ジオイドモデル「日本のジオイド 2000：GSIGEO 2000」を用いて求めたジオイド高を使用した。

結果としては、WGS-84 楕円体高の最低値と最大値の差が大きいことから GPS センサで検出される標高は精度が悪いということがわかった。

表４．３．１　GPGGA の結果

場所	実際の標高	ジオイド高	WGS-84 楕円体高	GPS センサでの標高
富山高専射水キャンパス 4 階	約 10m	36.9156m	45.3～56.7m	8.38～19.78m
二上山金撞堂付近	約 250m	37.5704m	291.2～306.8m	253.63～269.23m

５.湿度センサ

５.１　使用した湿度センサ

表５.１.１に湿度センサの仕様を示す。SHT-71 は SENSIRION 社のデジタル温度・湿度センサ SHT7x シリーズの１つで、全数校正されており、チップ内部のメモリに構成係数が保存されています。センサからこの係数によって校正された値が出力されるようになっています。実際の温度・湿度を得るためには以下の変換式を使って計算をする必要がある。

実際の湿度 ＝ -4.0 + 0.0405 * h - 2.8e-6 * h * h

実際の温度 ＝ -40.0 + 0.01 * t

センサとの通信方法は I2C に似た 2 線式のシリアルインターフェースである。また、データシートによると短い間隔で測定を連続して行うと、自己加熱により精度が落ちるためセンサの動作時間を 10%に抑える必要があります。通常、測定には 400ms ほど時間がかかるため 4 秒間隔ぐらいで測定するのがいいと考えられます。

表５.１.１　湿度センサの仕様

項目	内容
電源電圧(Vdd)	2.4～5.5V
測定範囲(湿度)	0～100%RH
測定範囲(温度)	-40～123.8℃
精度(湿度)	±3.0%RH
精度(温度)	±0.4℃
通信方式	デジタル 2 線式インターフェース
メーカー	Sensirion 社
型番	SHT-71

５.２　プログラムの流れ

図５.２.１に湿度センサのフローチャートを示す。

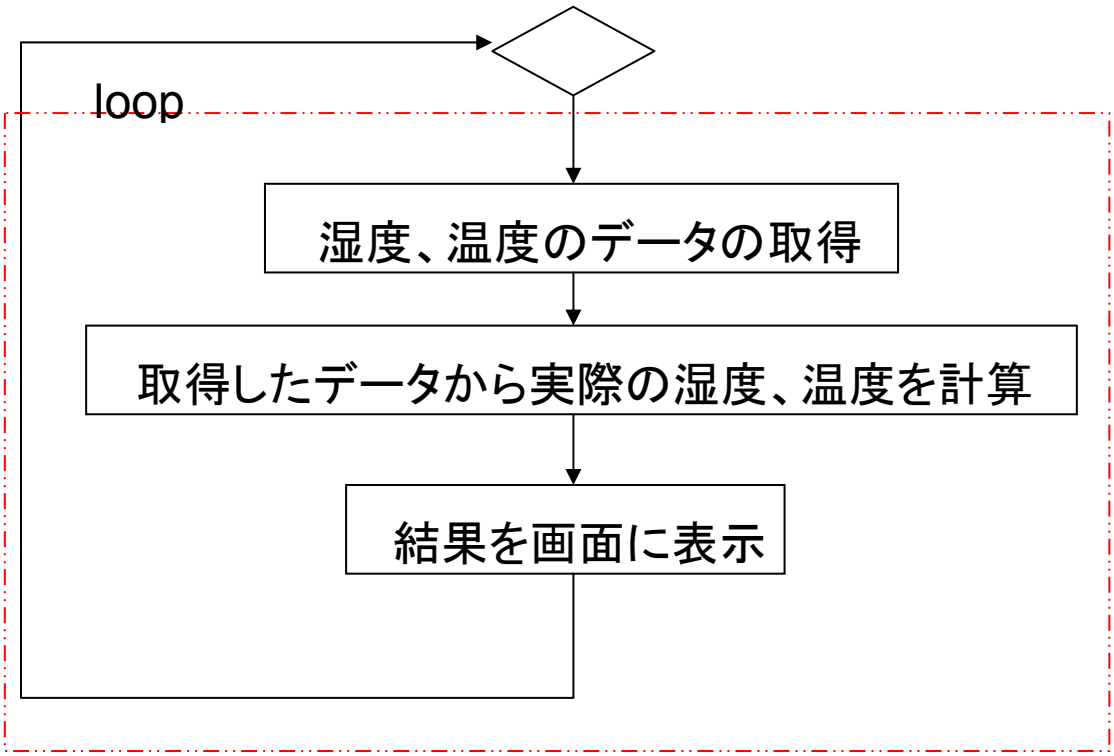

図５.２.１　湿度センサのフローチャート

５.３　動作確認

今回の測定では、湿度センサを比較するため温湿度データロガーTR-72U と一緒に測定した。表５.３.１に温湿度データロガーの仕様を示す。湿度センサの測定は 6 秒おきに測定する設定とし、24 時間計測を行った。温湿度データロガーTR-72U の測定の設定は 30 秒おきとなっている。図５.３.１に湿度センサの結果(湿度)を示す。図５.３.２に湿度センサの結果(温度)を示す。

表５．３．１　温湿度データロガーの仕様

項目	内容
電源	単 3 アルカリ電池(LR6)1 本
測定範囲(湿度)	10～95%RH
測定範囲(温度)	0～50℃
精度(湿度)	±5%RH
精度(温度)	平均±0.3℃
通信方法	USB 通信/シリアル通信(RS-232C)

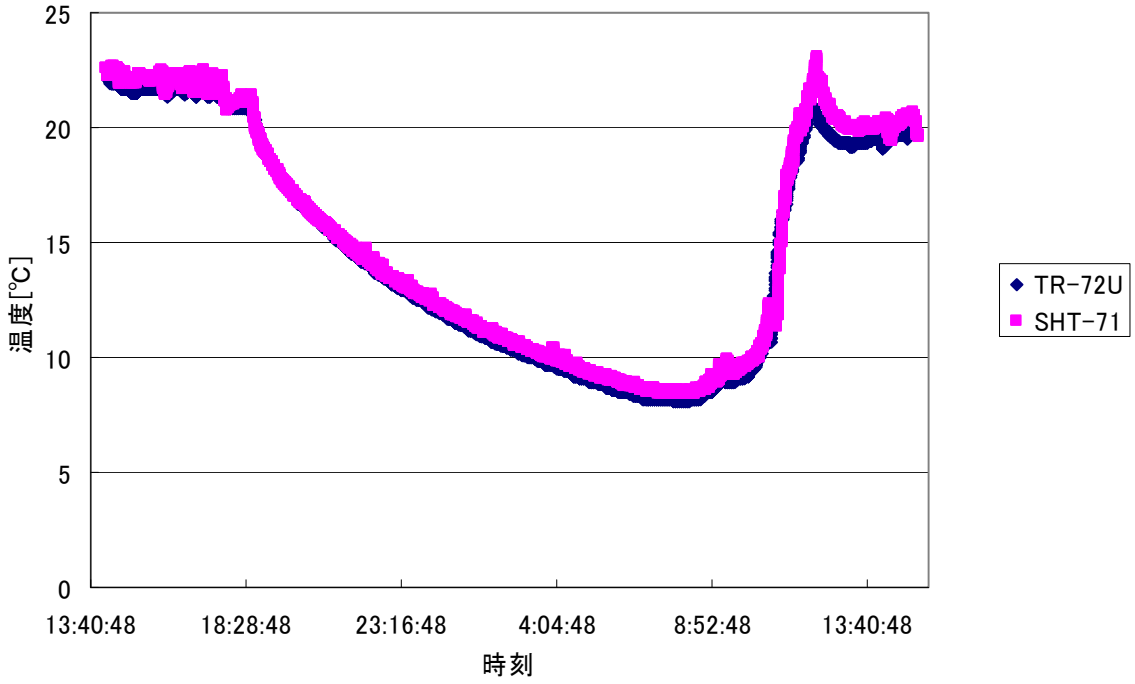

図５．３．１　湿度センサの結果(湿度)

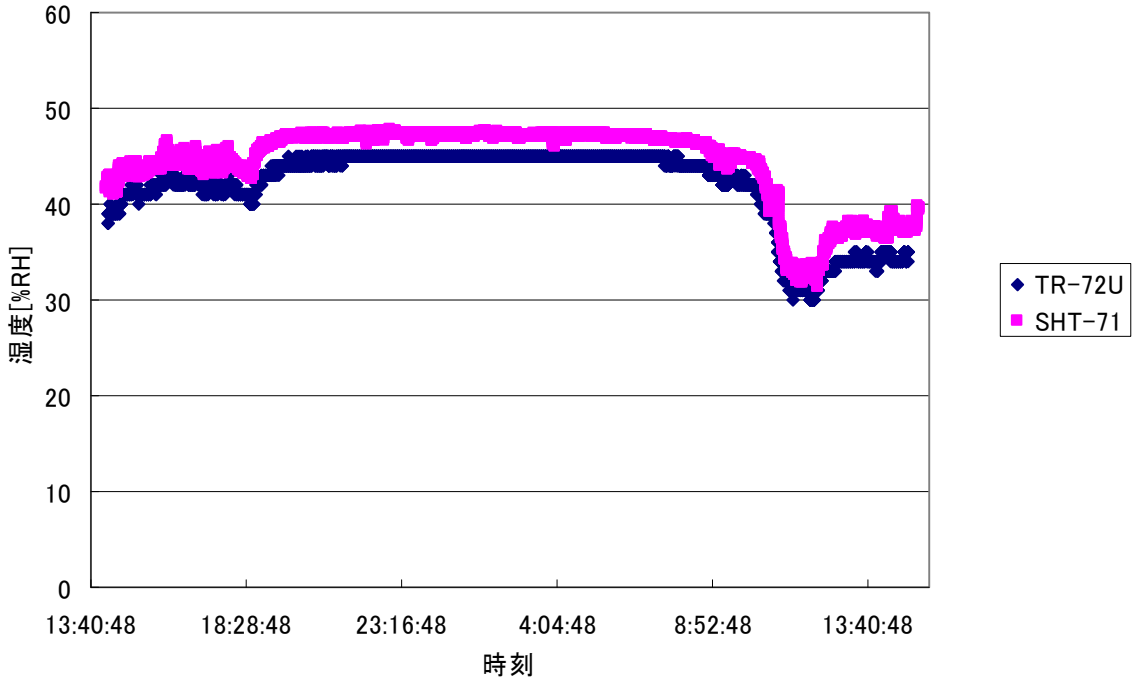

図５．３．２　湿度センサの結果(温度)

６.３つのセンサでの同時測定

６.１　問題点

図６.１.１に多機能気象観測装置のブロック図を示す。まずそれぞれのセンサとArduinoの通信方法はGPSセンサがシリアル通信、大気圧センサがI2C通信、湿度センサがシリアル通信となっている。ここで問題となるのがArduino unoにはシリアル通信用のピンRX、LXが１組しかない。これでは2種類以上のデバイスの通信の際に都合が悪い。

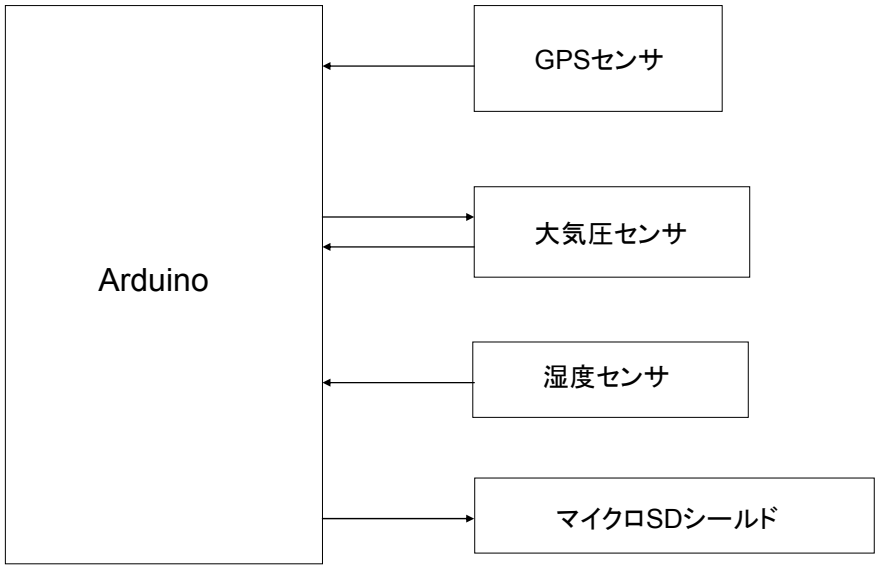

図６.１.１多機能気象観測装置のブロック図

図６.１.１に多機能気象観測装置のブロック図を示す。

６.２　解決方法

図６.２.１に多機能気象観測装置の回路図を示す。図６.２.２にSoftwareSerialのプログラム例を示す。解決方法として、Arduino 標準ライブラリとしてあるSoftwareSerialを使用する。SoftwareSerialとは、特別なピン以外のデジタルポートを使ってシリアル通信機能を実現することができるライブラリである。使い方は簡単で図６.２.２のように、はじめにSoftwareSerial.hをincludeした後、SoftwareSerial sotfSerial(R,T);として仮想ポートを宣言するだけである。宣言するときのRはRX、TはTX相当で、この例だとデジタル2ピンを受信ポート、デジタル3ピンを送信ポート

として使う意味となっている。

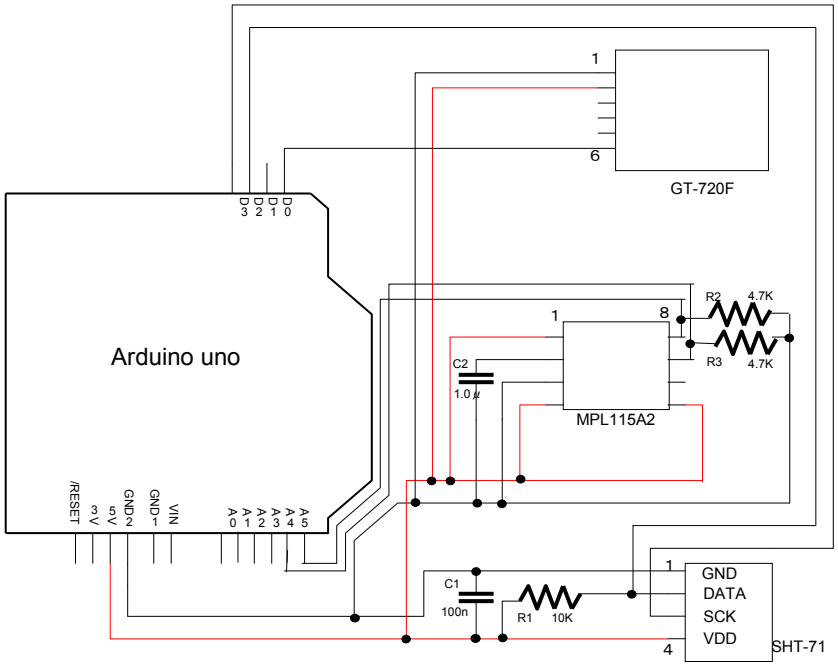

図６.２.１多機能気象観測装置の回路図

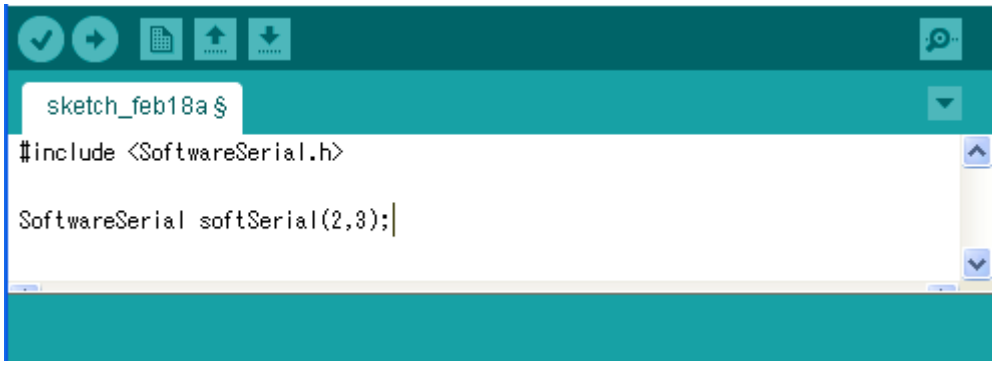

図６.２.２　SoftwareSerial のプログラム例

６.３　測定結果

図６.３.１に３つのセンサでの測定結果を示す。

Year	Month	Day	Hour	Min	Sec	latitude	longitude	Temperatu	Humidity	Atmospheric pressure
2013	2	5	13	27	46	36.75846	137.1596	19.3	43.6	1024.11
2013	2	5	13	27	56	36.75846	137.1596	19.29	43.73	1024.54
2013	2	5	13	28	6	36.75846	137.1596	19.3	43.7	1024.11
2013	2	5	13	28	16	36.75846	137.1596	19.32	43.93	1024.54
2013	2	5	13	28	26	36.75846	137.1596	19.3	43.93	1024.11
2013	2	5	13	28	36	36.75846	137.1596	19.27	43.89	1024.54
2013	2	5	13	28	46	36.75846	137.1596	19.32	43.86	1024.54
2013	2	5	13	28	56	36.75846	137.1596	19.32	43.77	1024.54
2013	2	5	13	29	6	36.75846	137.1596	19.34	43.63	1024.11
2013	2	5	13	29	16	36.75846	137.1596	19.32	43.47	1024.97
2013	2	5	13	29	26	36.75846	137.1596	19.35	43.7	1024.11
2013	2	5	13	29	36	36.75846	137.1596	19.32	43.63	1025.31
2013	2	5	13	29	46	36.75846	137.1596	19.38	43.41	1025.31
2013	2	5	13	29	56	36.75846	137.1596	19.39	43.48	1024.11
2013	2	5	13	30	6	36.75846	137.1596	19.34	43.51	1024.11
2013	2	5	13	30	16	36.75846	137.1596	19.32	43.54	1024.11
2013	2	5	13	30	26	36.75846	137.1596	19.33	43.8	1023.34
2013	2	5	13	30	36	36.75846	137.1596	19.36	43.64	1023.34
2013	2	5	13	30	46	36.75846	137.1596	19.35	43.67	1024.97
2013	2	5	13	30	56	36.75846	137.1596	19.34	43.35	1025.74
2013	2	5	13	31	6	36.75846	137.1596	19.38	43.35	1024.54
2013	2	5	13	31	16	36.75846	137.1596	19.39	43.51	1024.11
2013	2	5	13	31	26	36.75846	137.1596	19.41	43.42	1024.11
2013	2	5	13	31	36	36.75846	137.1596	19.36	43.74	1024.11
2013	2	5	13	31	46	36.75846	137.1596	19.39	43.87	1024.11
2013	2	5	13	31	56	36.75846	137.1596	19.37	43.8	1024.11
2013	2	5	13	32	6	36.75846	137.1596	19.36	43.84	1025.74
2013	2	5	13	32	16	36.75846	137.1596	19.38	43.64	1024.11
2013	2	5	13	32	26	36.75846	137.1596	19.36	43.51	1024.11
2013	2	5	13	32	36	36.75846	137.1596	19.38	43.87	1024.54
2013	2	5	13	32	46	36.75846	137.1596	19.38	43.61	1023.34

図６.３.１　３つのセンサでの測定結果

７.まとめ

大気圧センサは、気象庁のデータとの差が計測の始めには 1.5hPa あったが、計測の終わりには 0.5hPa となっていた。精度が±10hPa ということから規格どおりの値が出力されていることがわかる。GPS センサは、緯度経度に関しては精度 5m の範囲内の値が出力され日時に関しては世界標準時で出力されたため日本標準時に直した。高度に関しては、WGS－84 楕円体高の値の最高値と最低値の差が大きいことから高度の精度は悪いと考える。湿度センサは、2 秒間隔の計測ではデバイスの熱が収まっていなかったので精度がよくなかったが、6 秒間隔の計測では、湿度・温度ともに規格どおりの値が出力された。

問題点としては、現時点でのプログラムの仕様は GPS センサを受信した際に大気圧、湿度センサを動作させるようにしたため、GPS センサが受信する場所でしか測定できないようになっている。また GPS センサの GPRMC と GPGAA を同時に測定することができないようになっている。

参考文献

1) Freescale Semiconductor社,GT-720Fデータシート
2) Sensirion社,MPL115A2データシート
3) CanMore Electronics社,SHT-71データシート

付録

・各センサのサンプルプログラム

GPS センサ: http://garretlab.web.fc2.com/arduino/lab/gps_sensor/index.html

大気圧センサ: http://garretlab.web.fc2.com/arduino/lab/barometer_sensor/index.html

湿度センサ: http://projectsbiotope.blogspot.jp/2010/01/arduinosht-71.html

表 1　使用機器

名称	型番	メーカー
GPS センサ	GT-720F	CanMore Electronics 社
大気圧センサ	MPL115A2	Freescale Semiconductor 社
湿度センサ	SHT-71	Sensirion 社
マイコンボード	Arduino uno	Smart Projects

```
#include <SD.h>
#include <sht1x.h>
#include <stdio.h>
#include <SoftwareSerial.h>
#include <sht1x.h>
#include <Wire.h>
#define dataPin  6
#define clockPin 7
const int gpsRxPin      = 4;
const int gpsTxPin      = 5;

//SHT-71
SHT1x sht1x(dataPin, clockPin);
```

```
//MPL115A2
const int address = 0x60;
float read_coefficients(int total_bits, int fractional_bits, int zero_pad) {
   unsigned char msb, lsb;
   msb = Wire.read();
   lsb = Wire.read();
   return ((float) ((msb << 8) + lsb) / ((long)1 << 16 - total_bits + fractional_bits +
zero_pad));
}
unsigned int read_adc() {
   unsigned char msb, lsb;
   msb = Wire.read();
   lsb = Wire.read();
   return (((unsigned int)msb << 8) + lsb) >> 6;
}
float a0, b1, b2, c12, c11, c22;
SoftwareSerial gt(4,5);
String dataString = "";
void setup () {
   gt.begin(9600);
   Serial.begin(9600);
   analogReference(INTERNAL);
   //Serial.print("Initializing SD card...");
   pinMode(8, OUTPUT);
   if (!SD.begin(8)) {
      //Serial.println("Card failed, or not present");
   }else{
   //Serial.println("Card initialized.");
   }
   Wire.begin();
     Wire.beginTransmission(address);
     Wire.write(0x04);
     Wire.endTransmission();
     Wire.requestFrom(address, 12);
     if (Wire.available()) {
       a0 = read_coefficients(16, 3, 0);
```

```
    b1 = read_coefficients(16, 13, 0);
    b2 = read_coefficients(16, 14, 0);
    c12 = read_coefficients(14, 13, 9);
    c11 = read_coefficients(11, 10, 11);
    c22 = read_coefficients(11, 10, 15);
    //Serial.println("MPL115A2 Starting up");
  }
  Serial.print("Year");
  Serial.print(",");
  Serial.print("Month");
  Serial.print(",");
  Serial.print("Day");
  Serial.print(",");
  Serial.print("Hour");
  Serial.print(",");
  Serial.print("Min");
  Serial.print(",");
  Serial.print("Sec");
  Serial.print(",");
  Serial.print("latitude");
  Serial.print(",");
  Serial.print("longitude");
  Serial.print(",");
  Serial.print("Temperature");
  Serial.print(",");
  Serial.print("Humidity");
  Serial.print(",");
  Serial.println("Atmospheric pressure");



}

void loop () {
  static char s[256];
  static int pos = 0;
```

```
    if (gt.available()) {
        s[pos] = gt.read();
        if (s[pos] == '¥n') {
          s[pos - 1] = '¥0';
          analyze_data(s);
          pos = 0;
        } else {
          pos++;
        }
    }
}

// analyze GPS output.
void analyze_data(char *s) {
    char *type;
    char *time,*year;
    char *latitude, *longitude;
    char *tmp;
    char newlatitude[256], newlongitude[256];
    int riYear = 0, riMonth = 0, riDay = 0,riHour = 0,riMin = 0,riSec = 0;

    type = strtok(s, ",");
    if(strcmp(s, "$GPRMC") != 0) {
      return;
    }
    time = strtok(NULL, ",");//時刻
    tmp  = strtok(NULL, ",");//ステータス
    latitude = strtok(NULL, ",");//緯度
    tmp = strtok(NULL, ",");//北緯 or 南緯
    longitude = strtok(NULL, ",");//経度
    tmp = strtok(NULL, ",");//東経 or 西経
    tmp = strtok(NULL, ",");//移動の速度
    tmp = strtok(NULL, ",");//移動の真方位
    year = strtok(NULL, ",");//年月日
```

```
long hhmmss = atol(time);
riHour = hhmmss / 10000;
riMin =   (hhmmss / 100) % 100;
riSec =   hhmmss % 100;
long ddmmyy = atol(year);
riYear = 2000 + ddmmyy % 100;
riMonth = (ddmmyy / 100) % 100;
riDay = ddmmyy / 10000;
// 日本標準時は、UTC（世界標準時）よりも 9 時間進んでいる。
        riHour += 9;
        if( 24 <= riHour )
        {           // 日付変更
                    riHour - 24;
                    IncrementDay( riYear, riMonth, riDay );
        }
Serial.print(riYear);
Serial.print(",");
Serial.print(riMonth);
Serial.print(",");
Serial.print(riDay);
Serial.print(",");
Serial.print(riHour);
Serial.print(",");
Serial.print(riMin);
Serial.print(",");
Serial.print(riSec);
Serial.print(",");
//iYear
//iMonth
//iDay
//iHour
//iMin
//ISec

//Serial.println(time);
Serial.print(stod(latitude),7);
```

```
  Serial.print(",");
  Serial.print(stod(longitude),7);
  Serial.print(",");
  /*File dataFile = SD.open("datalog.txt", FILE_WRITE);   // define the filename, ファイル名を定義。
  if (dataFile) {                                        //if the file in the SD card was open to wrihte, SD カードの対象ファイルを開くことができれば
    dataFile.print(time);
    dataFile.print(",");
    dataFile.print(stod(latitude),7);
    dataFile.print(",");
    dataFile.println(stod(longitude),7);
    dataFile.close();
    Serial.println();
  }else {
    Serial.println("error opening file");
  }
*/
  SHT71();
  Humidity();
  delay(5000);
}

float stod(char *s) {
  float f;
  float deg, min;

  f = atof(s);

  deg = (int)f / 100;
  min = f - deg * 100;

  return deg + min / 60;
}
void IncrementDay( int& riYear, int& riMonth, int& riDay )
{
```

```
switch( riMonth )
{
case 1:
        if( 31 == riDay){ riMonth = 2; riDay = 1; }
        else                    { riDay++; }
        break;
case 2:
        if( 28 == riDay || 29 == riDay )
        {
                if( IsLeapYear( riYear ) )
                {       // 閏年（2 月は 29 日まである）
                        if( 29 == riDay){ riMonth = 3; riDay = 1; }
                        else                            { riDay++; }
                }
                else
                {       // 閏年じゃない（2 月は 28 日まで）
                        riMonth = 3; riDay = 1;
                }
        }
        else                    { riDay++; }
        break;
case 3:
        if( 31 == riDay){ riMonth = 4; riDay = 1; }
        else                    { riDay++; }
        break;
case 4:
        if( 30 == riDay){ riMonth = 5; riDay = 1; }
        else                    { riDay++; }
        break;
case 5:
        if( 31 == riDay){ riMonth = 6; riDay = 1; }
        else                    { riDay++; }
        break;
case 6:
        if( 30 == riDay){ riMonth = 7; riDay = 1; }
        else                    { riDay++; }
```

```
                break;
        case 7:
                if( 31 == riDay){ riMonth = 8; riDay = 1; }
                else                    { riDay++; }
                break;
        case 8:
                if( 31 == riDay){ riMonth = 9; riDay = 1; }
                else                    { riDay++; }
                break;
        case 9:
                if( 30 == riDay){ riMonth = 10; riDay = 1; }
                else                    { riDay++; }
                break;
        case 10:
                if( 30 == riDay){ riMonth = 11; riDay = 1; }
                else                    { riDay++; }
                break;
        case 11:
                if( 30 == riDay){ riMonth = 12; riDay = 1; }
                else                    { riDay++; }
                break;
        case 12:
                if( 31 == riDay){ riYear++; riMonth = 1; riDay = 1; }
                else                    { riDay++; }
                break;
        }
}
// 閏年の定義
// 西暦年が 4 で割り切れる年は閏年
// ただし、西暦年が 100 で割り切れる年は平年
// ただし、西暦年が 400 で割り切れる年は閏年
int IsLeapYear( int iYear )
{
        if( ( 0 == iYear % 4
            && 0 != iYear % 100 )
          || 0 == iYear % 400 )
```

```
        {
                return 1;
        }
        return 0;
}

void SHT71(void)
{
   float temp_c;
   float temp_f;
   float humidity;
   temp_c = sht1x.readTemperatureC();
   temp_f = sht1x.readTemperatureF();
   humidity = sht1x.readHumidity();
   //Serial.print("Temperature: ");
   Serial.print(temp_c, DEC);
   Serial.print(",");
   //Serial.print("C / ");
   //Serial.print(temp_f, DEC);
   //Serial.print("Humidity: ");
   Serial.print(humidity);
   Serial.print(",");
   //Serial.println("%");
}

void Humidity(void)
{
   Wire.beginTransmission(address);
   Wire.write(0x12);          // Start both conversions(Pressure and Temperature)
   Wire.write(0x01);
   Wire.endTransmission();
   delay(5);
   Wire.beginTransmission(address);
   Wire.write(0x00);   // Read pressure and temperature
   Wire.endTransmission();
   Wire.requestFrom(address, 4); // Request 4 bytes
```

```
  if(Wire.available()) {
    unsigned int Padc = read_adc();
    unsigned int Tadc = read_adc();
    float Pcomp = a0 + (b1 + c11 * Padc + c12 * Tadc) * Padc + (b2 + c22 * Tadc) * Tadc;
    float Pha = Pcomp * 650 / 1023 + 500;
    //Serial.print("Atmospheric pressure: ");
    Serial.println(Pha);


  }
}
```