令和 2 年度

卒業研究論文

高空風況計測用気球の挙動評価システムの開発

1710950092　長久 泰樹

近畿大学工学部機械工学科

流体エネルギー研究室

目次

第 1 章　緒言

1.1 風況計測の背景

日本はエネルギー消費率が高いにも関わらず，エネルギー自給率が低い国となっている．エネルギー源として使われる石油・石炭などのエネルギー資源が乏しいことなどからエネルギー源を輸入に頼り，エネルギー消費による地球温暖化が進むなか，絶えず資源が補充され枯渇せず，二酸化炭素を排出しないものとして再生可能エネルギーが注目されている．その再生可能エネルギーの中の一つとして，風力発電が挙げられる．しかし，風力発電は安定してエネルギーを生み出す必要があり，発電機材が大型であるため，建設に適した場所が限定されている．近年では，日本ではそのような問題を解決するために，島国の特長を活かして海に風車を設置する洋上風力発電の計画が進んでいる．風力発電において，安定してエネルギーを生み出すための発電予測として発電機建設場所での正確な風況計測は必要不可欠であるため，陸地だけでなく海上での風況計測方法の開発を行う必要がある．

図 1.1　洋上風力発電 [1)]

1.2 高空風況計測の現状

風力発電の発電予測において風況計測は必要不可欠であり，従来の方法である風況観測タワーを用いた計測方法がある．しかし，風況観測タワーの建設には高いコストがかかるだけでなく，設置後のメンテナンスにも長時間かかってしまう．また，発電機の大型化が進むなか，60m 以上になると安全面などの規制があり，建設場所により部品運送が困難となっている．

図 1.2　洋上風力発電設備・観測タワー

1.3 研究の目的

本研究では気球を用いた風況計測システムの開発を行い，計測装置装置の小型化，低コストでの風況計測を目的とする．特に気球の挙動を評価するシステムの開発に着目し，気球によって計測した風速の修正に繋がる手法を提案する [3].

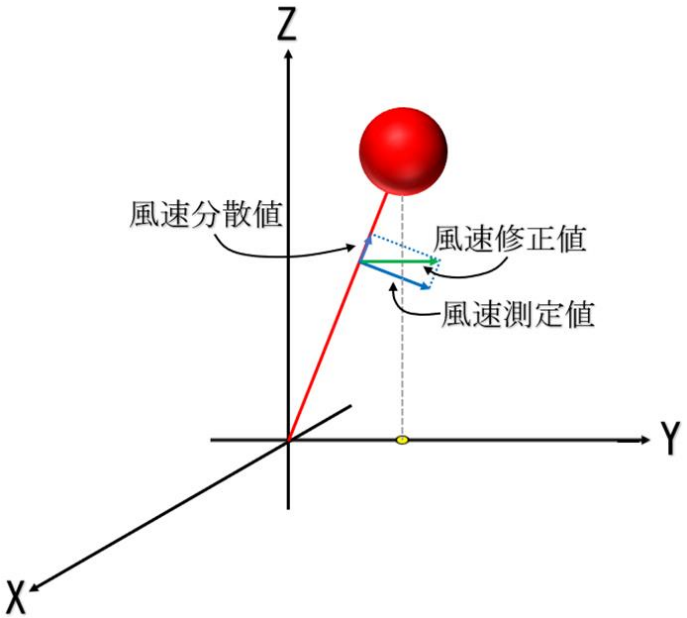

図 2　風速修正

第 2 章　浮揚システムの開発および風況計測方法

風速は地面に対して平行な向きで計測する必要があるが，気球に風速計を取り付けた場合，風によって動いている状態で計測しているため，計測軸が変化してしまう．そこで，気球の位置を検知することで計測軸を予測し，正しい計測軸に修正する．本研究は GPS センサーと IMU センサーによる気球の位置を検知する方法を検証した．

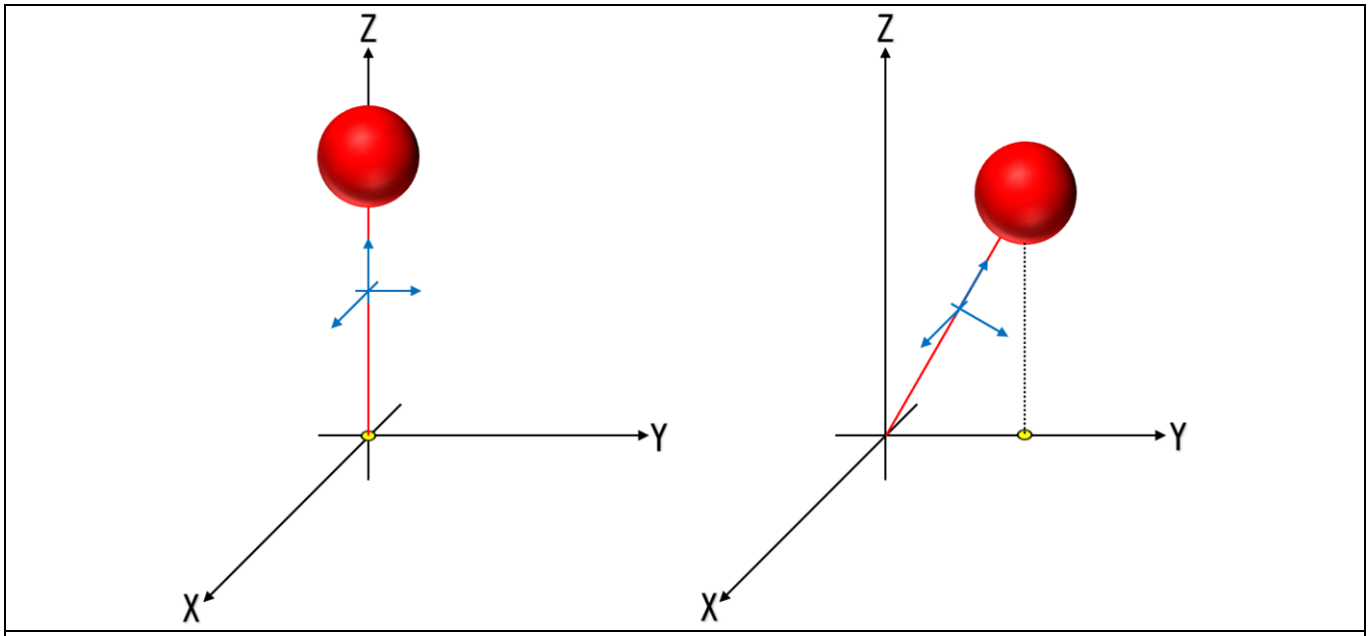

図 2.1　気球の位置検知

2.1 RTK

RTK(Real Time Kinematic)とは相対測位法と呼ばれ，基準局と移動局の 2 つの受信機で同一時間に観測したデータから相対位置を計測する測位法である．スマートフォンやカーナビのように 1 つの受信機を用いてリアルタイムで移動して計測を行うと誤差が数 m 発生するが，RTK では 2 つの受信機の間で情報をやりとりしてずれを補正することで，誤差を数 cm ほどに抑えることができる．

気球が無風状態(ウィンチの上)にあるときの GPS 座標を基準とし，浮揚しているときの座標の差から気球の傾きを計算することで風速を正しい測定軸に修正する．このとき，風速計はウィンチから気球に結ぶひもの直線状に，GNSS 受信機(移動局)は気球の下に取り付ける．なお，基準局は座標が動かないように固定する．

本研究は GNSS モジュール：NEO-M8P (u-blox)，アンテナ：TW2710 (Tallysman)を用いて行った．2 台のパソコンに GNSS モジュール，アンテナを接続し，それぞれ基準局，移動局として RTK 測位を行った [4),5)]．なお，基準局は NEO-M8P-2，移動局は NEO-M8P-0 を用いる．表 2.1，表 2.2 に GNSS モジュール，アンテナの仕様，図 2.2，図 2.3 に基準局，移動局の装置の概要を示す．

表 2.1　GNSS モジュール

		NEO-M8P-2(基準局)	NEO-M8P-0(移動局)
GNSS	GPS/QZSS	●	●
	GLONASS	●	●
	Galileo		
	BeiDou	●	●
インターフェース	UART	●	●
	USB	●	●
	SPI	●	●
	DDC(I2C 互換)	●	●
	基地局	●	
電源電圧	2.7~3.6V	●	●
タイムパルス出力	0.25~10MHz	●	●
測位データ出力レート	5Hz(RTK 測位結果出力)	●	●

表 2.2　アンテナ

アプリケーション	GPS，GLONASS，Galileo，BeiDou
周波数グループ	極超短波(1GHz~2GHz)
周波数	1.561GHz，1.575GHz，1.602GHz
周波数範囲	1.557GHz~1.606GHz

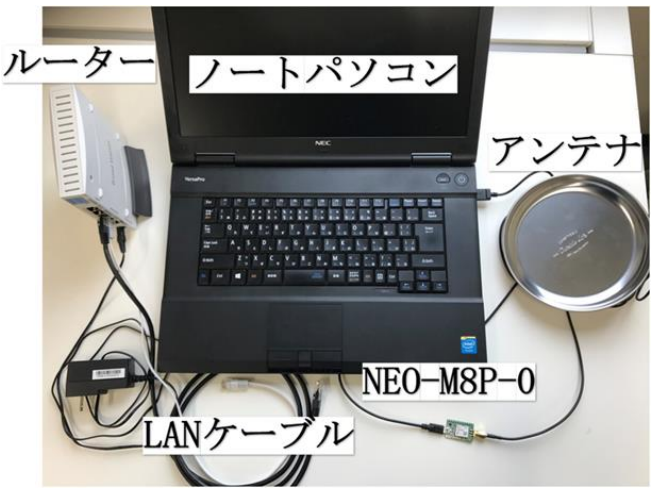

図 2.2　基準局装置

図 2.3　移動局装置

u-center を用いて NEO-M8P の設定を行った. u-center とは，u-blox のウェブサイトから無料でダウンロードすることができる高機能 GPS 評価・グラフィック表示化ツールであり，u-blox 製 GPS レシーバーの設定を行うことができる．以下に設定手順についてまとめる.

- NEO-M8P 設定

(1) u-center を起動する．

(2) Connecting / Disconnecting ボタンから COM ポートを選択し，受信機と接続する．接続が確認されると緑色に変わる．COM ポートはデバイスマネージャーから確認できる．

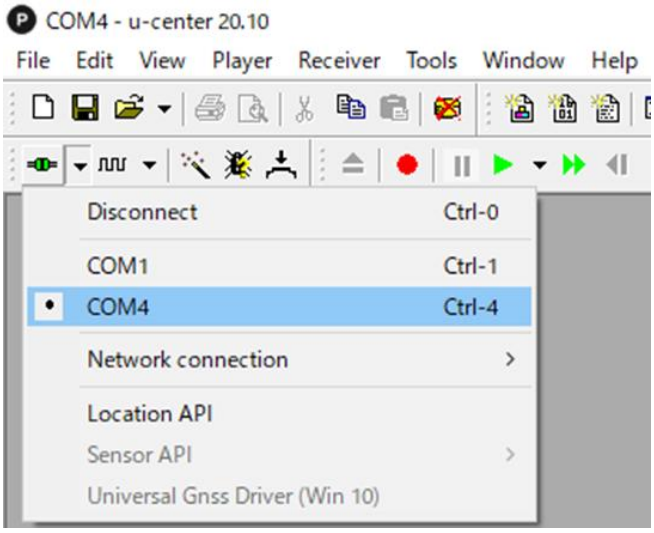

図 2.4　接続設定

(3) メニューから View>Configuration View を選択して開く．

(4) 受信機の設定を初期状態にする．左側のリストから CFG を選択する． Devices の 4 つの項目を全て選択し，Revert to default Configuration にチェックを入れて send ボタンをクリックする．

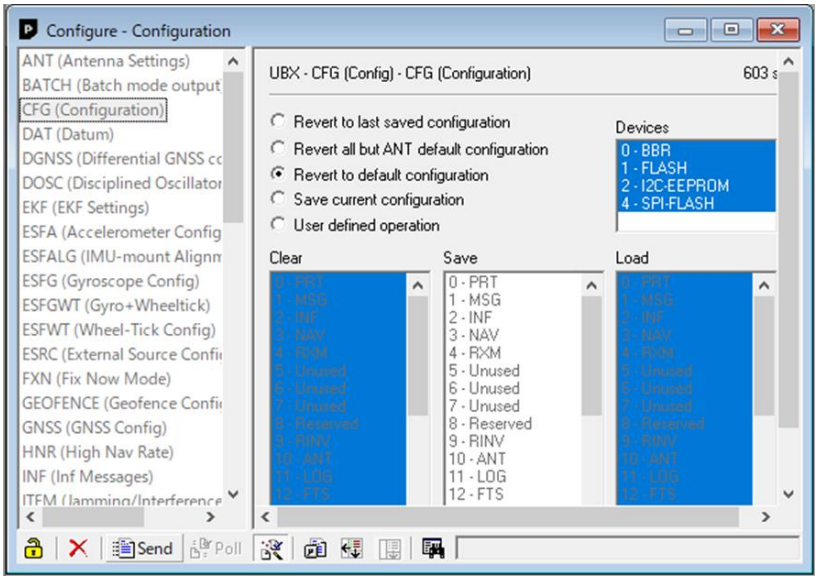

図 2.5　初期化設定

(5) 受信衛生の設定を行う．左側のリストから GNSS を選択する．BeiDou の Enable にチェックを入れる．GLONASS の Enable のチェックを外す．

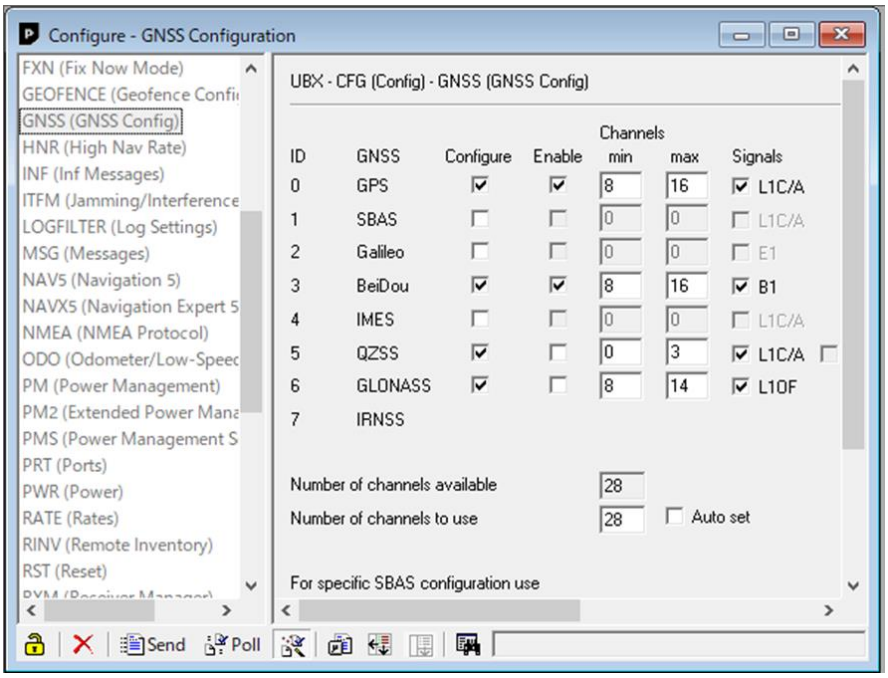

図 2.6　受信衛生設定

(6) 基準局として必要なデータを受信機に出力させる設定を行う．左側のリストから MSG を選択する．Message の項目から 02-13RXM-SFRBX を選択し，USB にチェックを入れて send ボタンをクリックする．(基準局のみ設定)

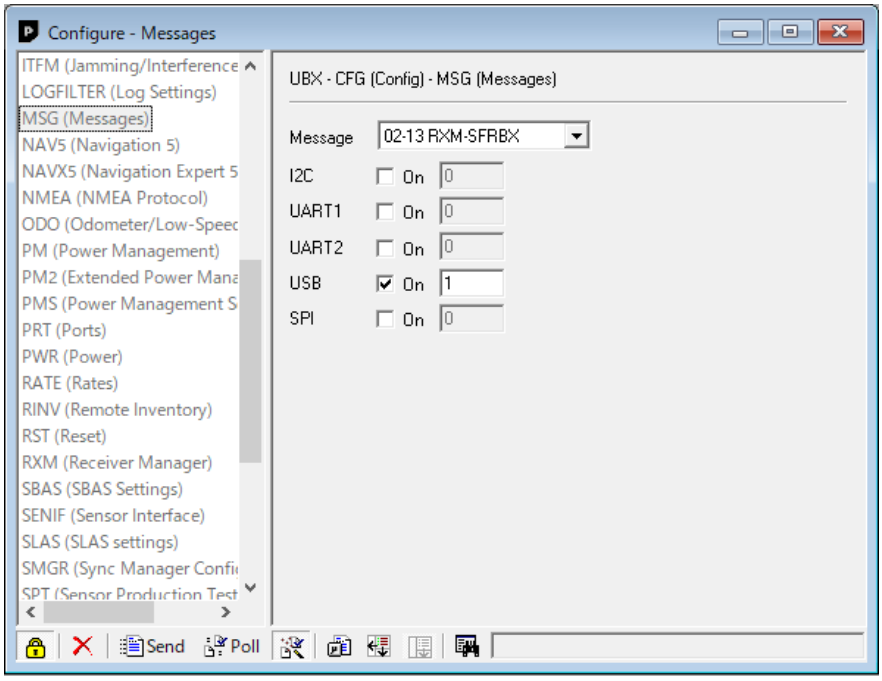

図 2.7　基準局出力設定

(7) 移動局として必要なデータを受信機に出力させる設定を行う．左側のリストから MSG を選択する．Message の項目から 02-15RXM-RAWX を選択し，USB にチェックを入れて send ボタンをクリックする．(移動局のみ設定)

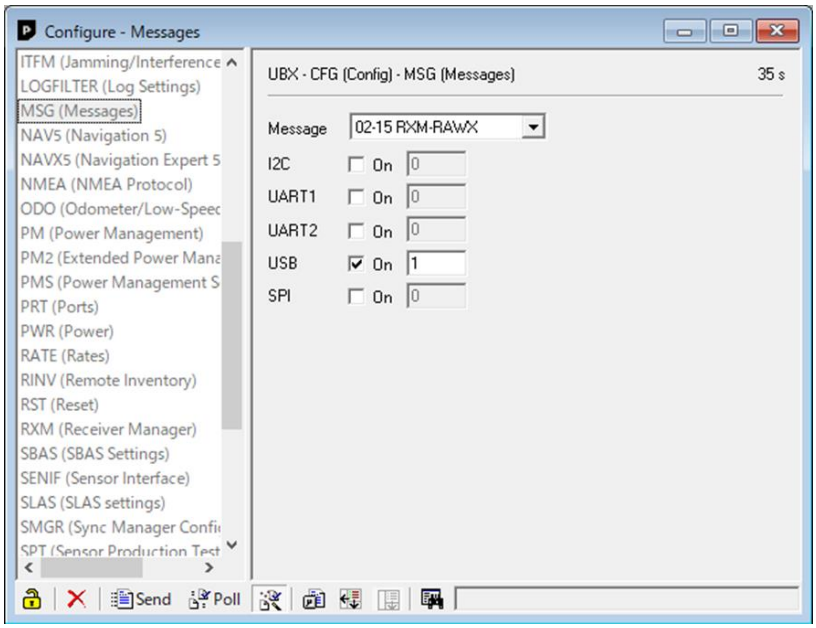

図 2.8　移動局出力設定

(8) 受信機を再起動しても設定が残るように受信機に設定を保存する．左側のリストから CFG を選択する． Devices の 4 つの項目を全て選択し，Save current Configuration にチェックを入れて send ボタンをクリックする．

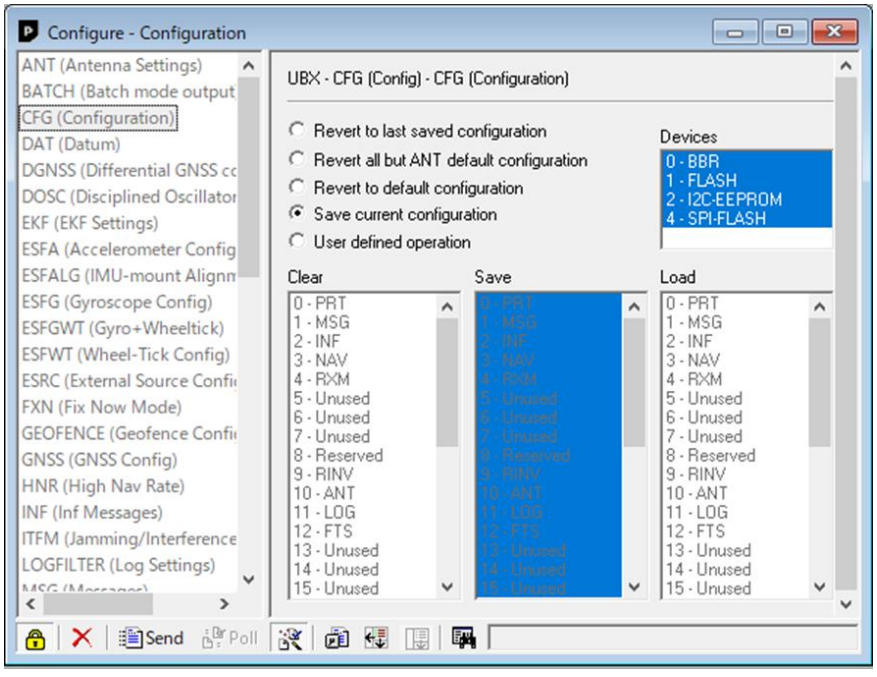

図 2.9　設定保存

(9) u-center を閉じる.

NEO-M8P 設定終了後, RTK 解析ソフトウェアの RTKLIB を用いて RTK 測位を行った. RTKLIB とは Windows パソコンや Raspberry Pi などの小型 Linux マシン上で動作する, オープンソースの RTK ライブラリとそのライブラリを使うアプリケーション・ソフトウェア群である. 表 2.3 に本研究で用いた RTKLIB のプログラムを示す.

表 2.3　RTKNAVI プログラム

機能	ファイル名	詳細
通信サーバー	STRSVR	基地局が観測した生データを取得して伝送する
リアルタイム測位	RTKNAVI	測位計算をリアルタイムに実行する

基準局，移動局をそれぞれルーターに接続し，STRSVR，RTKNAVI を起動して設定を行う. 設定完了後，RTK 測位を開始する．以下に RTK 測位の実験手順についてまとめる.

- RTK 実験手順

(1) STRSVR(基準局)設定

1. RTKLIB 内の STRSVR を起動する.

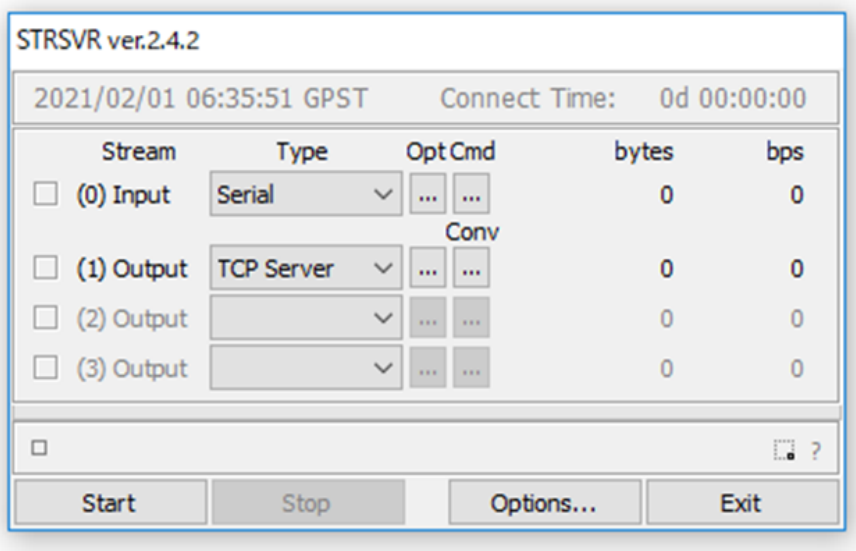

図 2.10　STRSVR

2. Stream-(0)Input の type として Serial を選択する．Option(Opt)ボタンをクリックし，表示された Serial Options ダイアルログで，Port として基地局受信機の接続された COM ポートを選択する．その他は図 2.11 の通りとする.

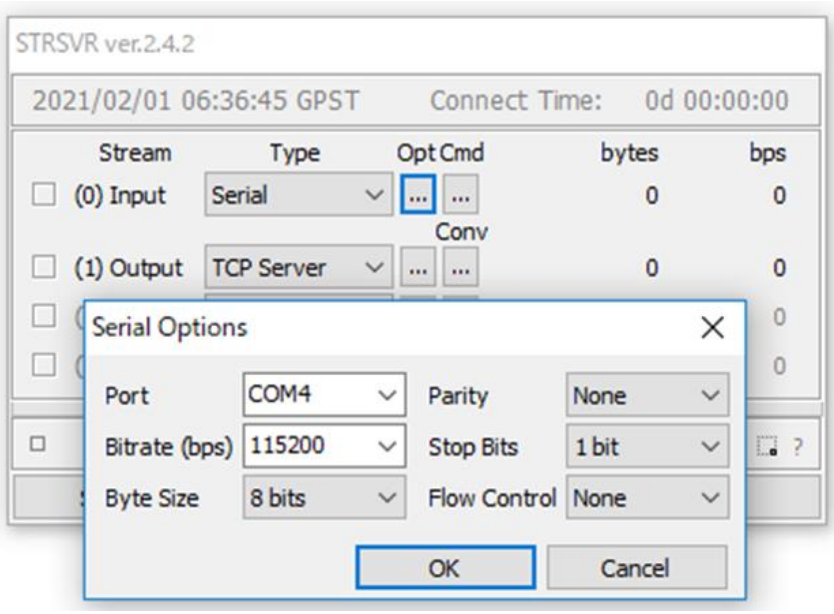

図 2.11　Serial

3. Stream-(1)Input の type として TCP Server を選択する．Option(Opt)ボタンをクリックし，表示された TCP Server Options ダイアルログで，TCP Server Port を Port に入力する．OK をクリックする．

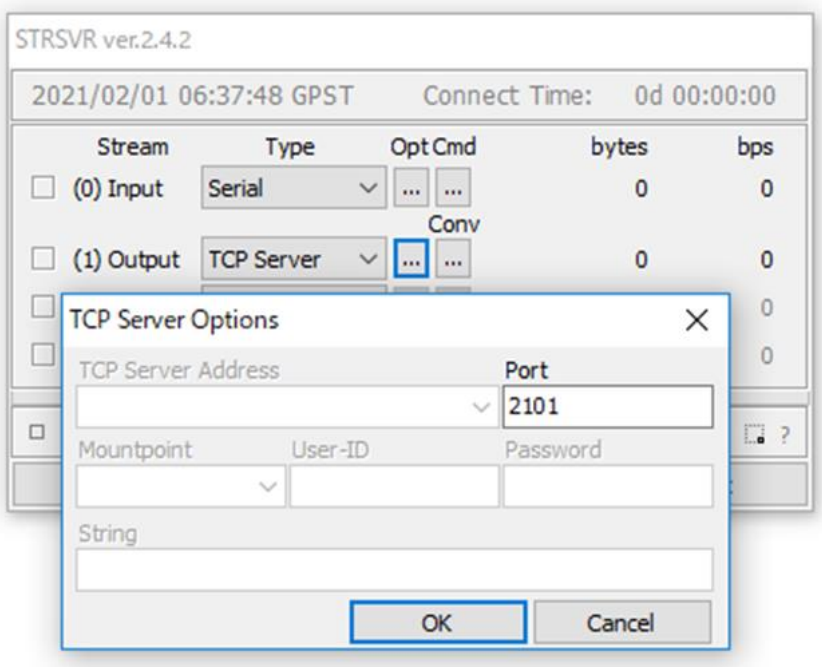

図 2.12　TCP Server

4. STRSVR の Start ボタンをクリックする．(0)Input 左側のデータインジケーターが黄緑に点灯し，右側の bytes，bps 値が上昇することを確認する．(1)Input 左側のデータインジケーターは黄色に点灯して，移動局の接続待ち状態となる．

(2) RTKNAVI(移動局)設定

1. RTKLIB 内の RTKNAVI を起動する.

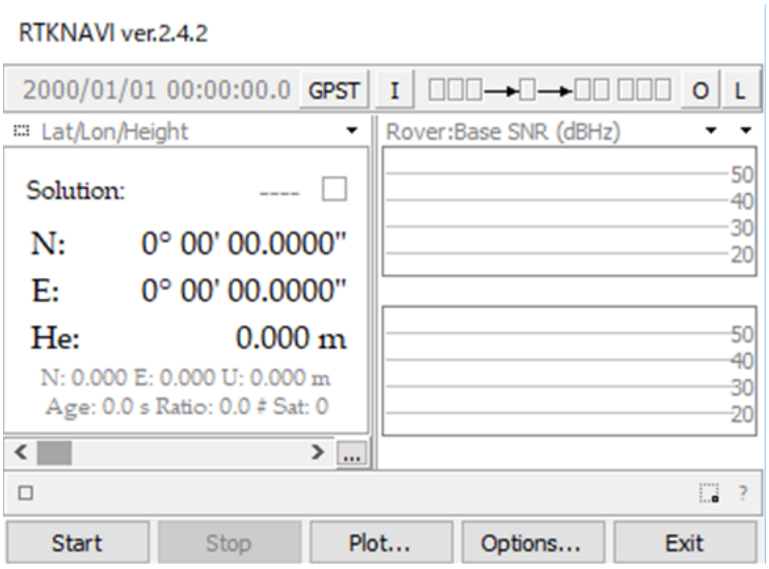

図 2.13　RTKNAVI

2. Rover にチェックを入れ, type として Serial, Format として u-blox を選択する. Option(Opt) ボタンをクリックし，表示された Serial Options ダイアルログで，Port として移動局受信機の接続された COM ポートを選択する．その他は図 2.13 の通りとし，OK をクリックする.

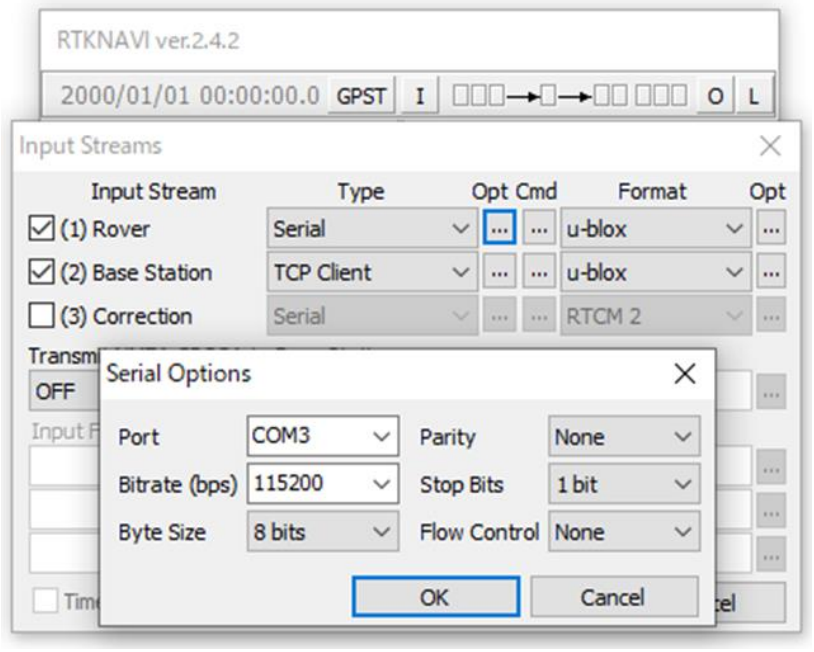

図 2.13　Input Streams-Rover

3. Base Station にチェックを入れ，type として TCP Client，Format として u-blox を選択する．Option(Opt)ボタンをクリックし，表示された TCP Client Options ダイアルログで，TCP Server Address として基地局 PC の IP アドレスを入力する．IP アドレスはコマンドプロンプトにて ipconfig を入力することで確認できる. Port として基地局で設定した TCP Server Port を入力する．OK をクリックする．

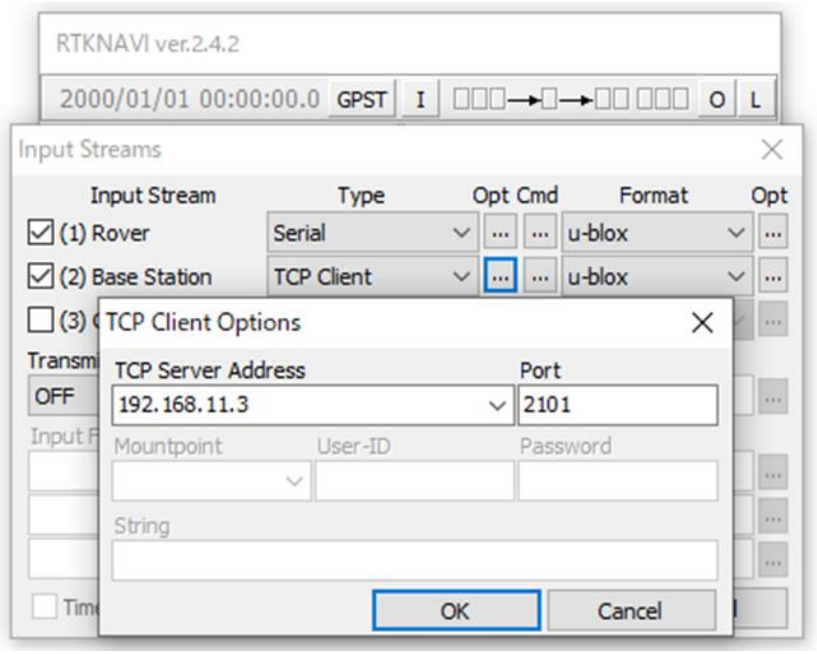

図 2.14　Input Streams-Rover

4. RTKNAVI の Options ボタンをクリックする．

5. Setting1 を選択する．Positioning Mode として Kinematic を選択する．単独測位を行う場合は Single を選択する．Frequencies として L1 を選択する．Elevation Mask として 15 を選択する．Elevation Mask 横の...ボタンをクリックし，Rover と Base Station にチェックを入れ，L1 の値を全て 40 と入力する．Excluded Satellites の項目に C02 と入力する．使用する衛星にチェックを入れる..

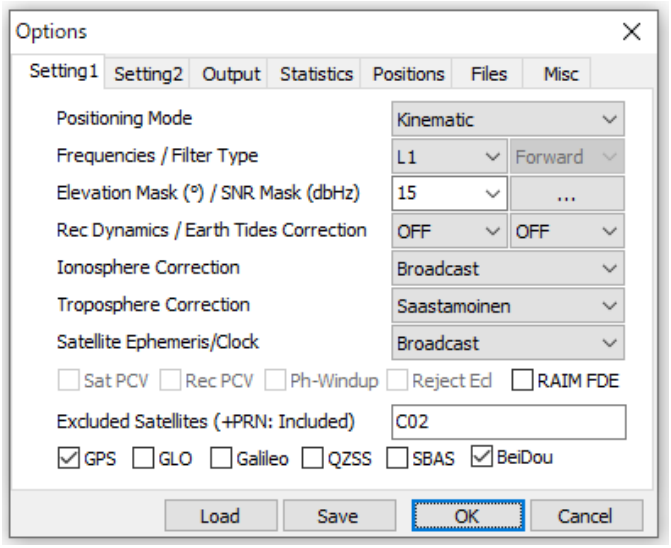

図 2.15　Setting1

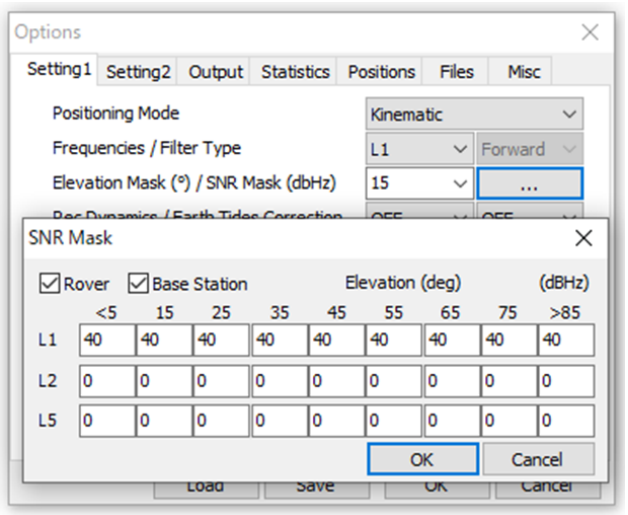

図 2.16　Setting1-SNR Mask

6. Setting1 を選択する．Integer Ambiguity Res として Fix and Hold，OFF，ON を選択する．

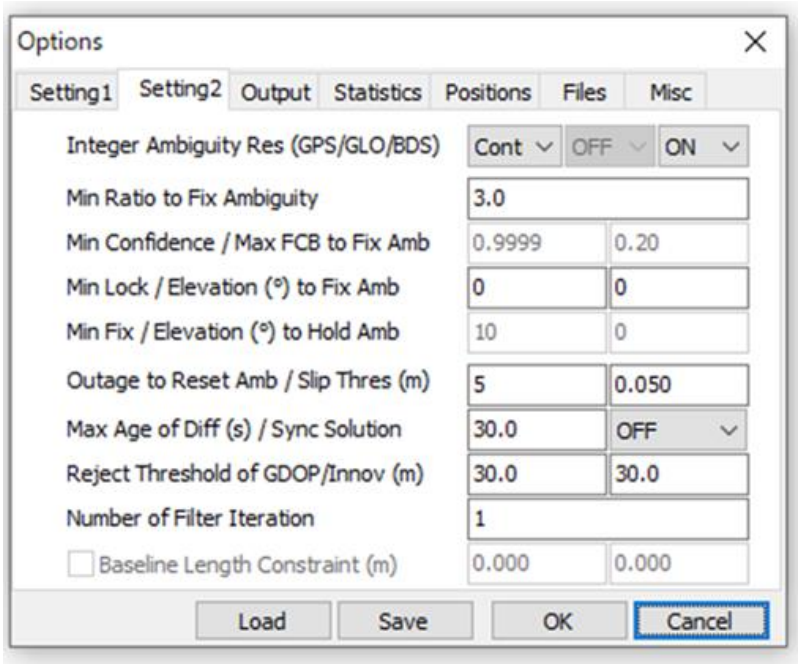

図 2.17　Setting2

7. Positions を選択する．Base Station の設定単位として Lat/Lon/Height(deg/m)を選択する．基地局の緯度，経度，標高を入力する．OK をクリックする．

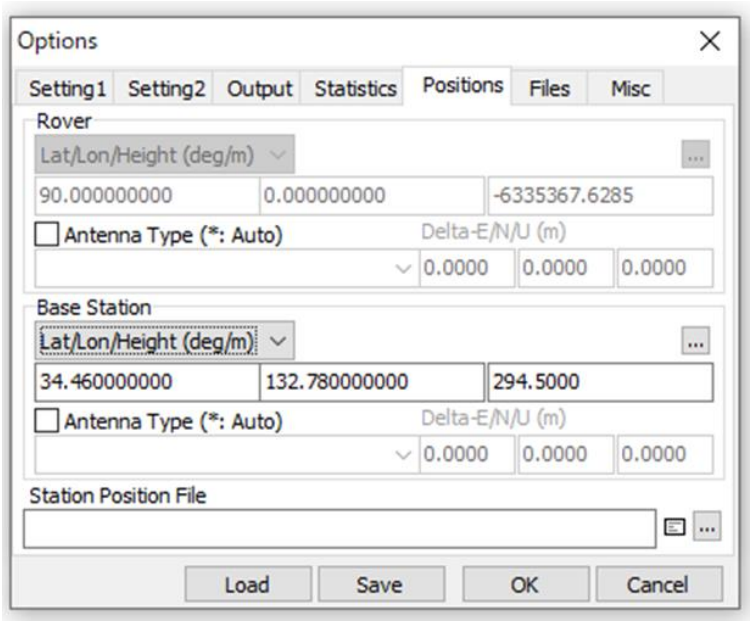

図 2.18　Positions

8. RTKNAVI の Start ボタンをクリックし，RTK 測位を開始する.

2.2 IMU

IMU(inertial measurement unit)とは 3 軸の加速度と 3 軸の角速度を検出する慣性計測装置である. 計測した角速度から気球の姿勢を検知することで風速を正しい計測軸に修正する. このとき，風速計と IMU センターは風の影響を受けても同じ傾き，回転となるように気球の下に直線的に固定する.

本研究は IMU：MPU9250，Raspberry Pi 4 を用いて行った．IMU センサーを Raspberry Pi に接続し，サンプリング周波数 1 Hz で計測した加速度，角速度，地磁気を抽出した．表 2.4，表 2.5 に IMU センサー，Raspberry Pi の仕様を示す.

表 2.4　IMU センサー

センサー	lnvenSense MPU-9250
インターフェース	I2C，SPI
電源電圧　VDD	DC2.4V~3.6V
I/O 電圧 VDDIO	DC1.7V~(VDD)
最大クロック	400kHz(I2C)，1MHz(SPI Read/Write)，20MHz(SPI ReadOnly)
加速度	
測定レンジ	±2 / ±4 / ±8 / ±16g
分解能	16 ビット
感度	1LSB=0.061mg(±2)，0.122mg(±4)，0.244mg(±8)，0.488mg(±16)
ノイズ	300μg/√Hz
LPF	5~260Hz
出力レート	0.24~4000Hz
ジャイロ部	
測定レンジ	±2 / ±4 / ±8 / ±16g
分解能	16 ビット
感度	1LSB=0.00763(±250), 0.01526(±500), 0.03048(±1000), 0.06097(±2000)°/sec
ノイズ	0.01°/√Hz
LPF	5~250Hz
出力レート	4~8000Hz
コンパス部	旭化成エレトロニクス AK8963
測定レンジ	±4800μT
分解能	14 ビット/16 ビット
感度	0.6μT/LSB(14bit)，0.15μT/LSB(16bit)
動作温度	−40℃~+85℃

表 2.5　Raspberry Pi 4

CPU		Broadcom BCM2711,　quad-core Cortex-A72 (ARM v8) 64-bit SoC
CPU クロック		1.5GHz
メモリー		UD-RP4B8：8GB　UD-RP4B4：4GB　UD-RP4B2：2GB
接続コネクター	USB	・USB3.0 Standard A コネクター×2 ・USB2.0 Standard A コネクター×2
	有線 LAN	RJ-45×1：IEEE802.3i(10BASE-T)，IEEE802.3u(100BASE-TX), IEEE802.3ab(1000BASE-T)
	その他	・HDMI 出力(マイクロ)×2 ・microSD カードスロット×1 ・40 ピン GPIO
電源定格		DC 5V

IMU と Raspberry Pi の接続は以下のように行った [6]．図 2.19，図 2.20 に MPU9250 と Raspberry Pi のピン配置，図 2.21 に接続の概要を示す．

MPU9250 1 番 pin (VDD) >> raspi 1 番 pin (3V3 power)
MPU9250 2 番 pin (VDDIO) >> raspi 1 番 pin (3V3 power)
MPU9250 3 番 pin (SCL/SCLK) >> raspi 5 番 pin (SCL)
MPU9250 4 番 pin (SDA/SDI) >> raspi 3 番 pin (SDA)
MPU9250 5 番 pin (AD0/SDO) >> raspi 6 番 pin (Ground)
MPU9250 6 番 pin (~CS) >> raspi 1 番 pin (3V3 power)
MPU9250 10 番 pin (GND) >> raspi 9 番 pin (Ground)

用途	名称	ピン番号	写真	ピン番号	名称	用途
電源 2.4V~3.6V	VDD	1		10	GND	電源・信号グランド
I/O 電源	VDDIO	2		9	INT	割り込み出力
I2C クロック/SPI クロック	SCL/SCLK	3		8	AUX_DA	(外部センサ用 SDA)
I2C データ/SPI データイン	SDA/SDI	4		7	AUX_CL	(外部センサ用 SCL)
I2C アドレス選択/SPI データアウト	AD0/SDO	5		6	~CS	チップセレクト

図 2.19　MPU9250 ピン配置

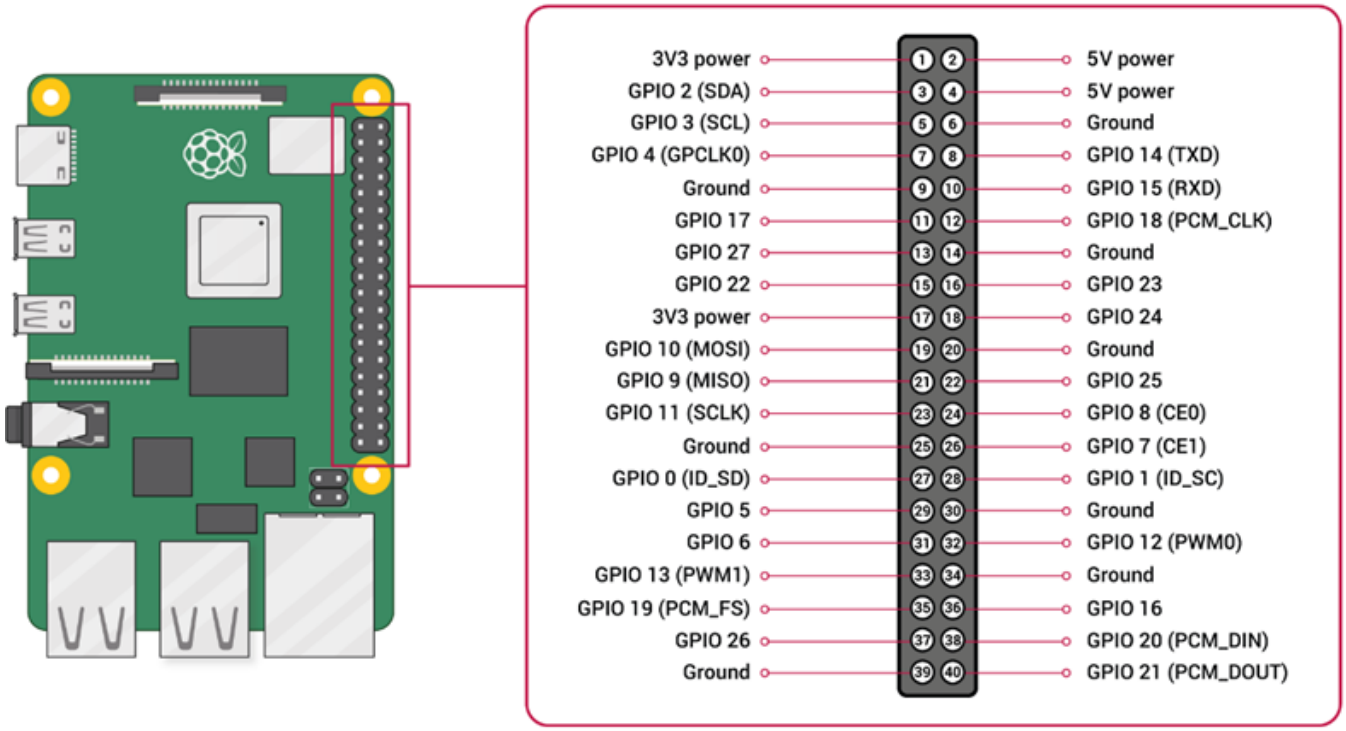

図 2.20　Raspberry Pi ピン配置

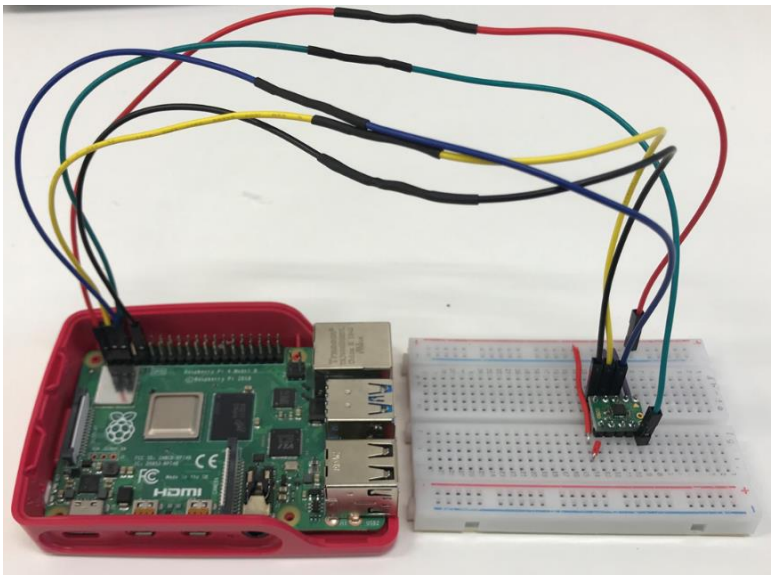

図 2.21　MPU9250-Raspberry Pi 接続

図 2.22，図 2.23. に MPU9250 の加速度・角速度，地磁気の計測軸について示す.

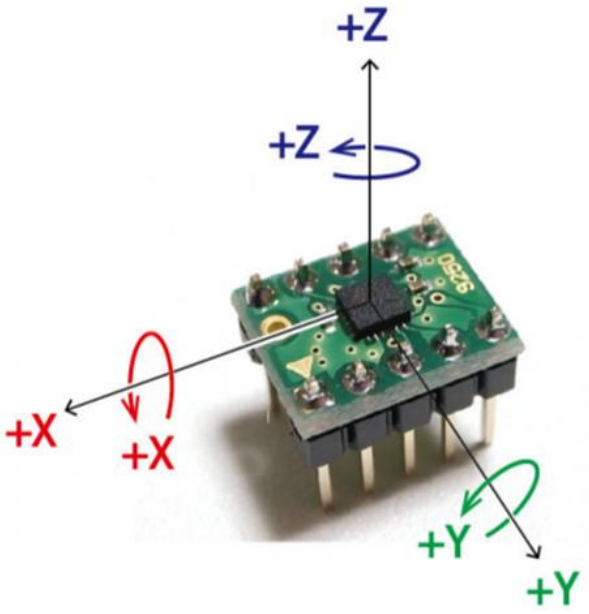

図 2,22　加速度・角速度計測軸

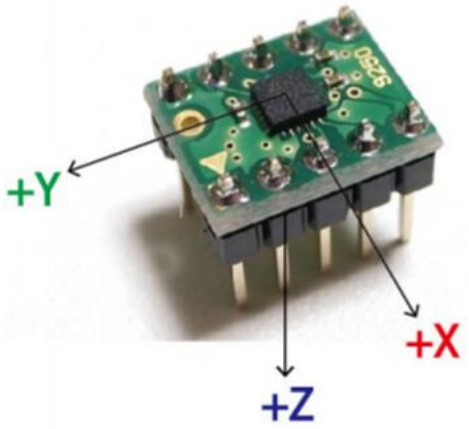

図 2.23　地磁気計測軸

2.3 Python プログラム

IMU での測定，風速修正は Python のプログラムを用いて行う．風速修正は計測した角速度から気球の位置を予測して風速を正しい計測軸に修正する [7]．風によって固定軸であるグローバル座標(X,Y,Z)から 3 次元に回転し，別の座標(x,y,z)に変化するため，IMU センサーの加速度の計測値を用いてそれぞれ 3 軸(ロール，ピッチ，ヨー)を中心に回転させることによって現在の位置を計算し，ベクトル座標をグローバル座標値で示すことで姿勢を検知する．

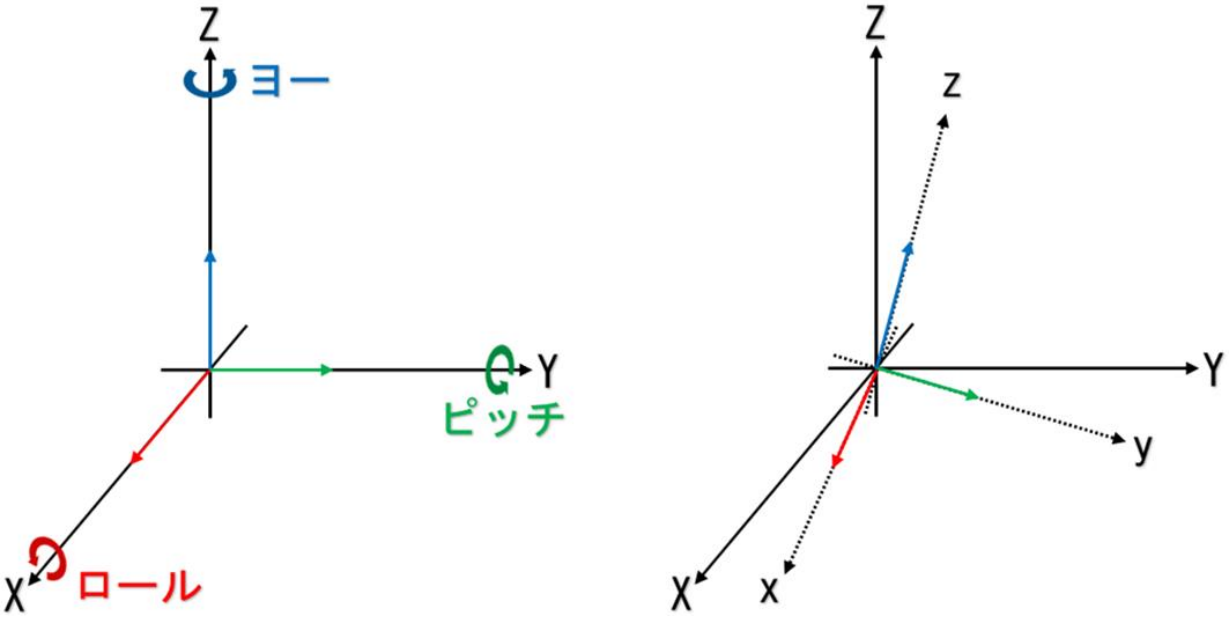

図 2.24　計測軸変化

姿勢検知は前回の位置が必要となるため，気球の浮上前にグローバル座標を定義する．浮揚中の気球のロール，ピッチ，ヨーの回転角をそれぞれ$\theta_r, \theta_p, \theta_y$とすると，それぞれ 3 軸(ロール，ピッチ，ヨー)を中心に回転させる回転行列 R(r,p,y)はそれぞれ式(1)，式(2)，式(3)で表される．

R(r)：ロール軸を中心に回転させる回転行列

$$R(r) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_r & -\sin\theta_r \\ 0 & \sin\theta_r & \cos\theta_r \end{bmatrix} \tag{1}$$

R(p)：ピッチ軸を中心に回転させる回転行列

$$R(p) = \begin{bmatrix} \cos\theta_p & 0 & \sin\theta_p \\ 0 & 1 & 0 \\ -\sin\theta_p & 0 & \cos\theta_p \end{bmatrix} \tag{2}$$

R(y)：ヨー軸を中心に回転させる回転行列

$$R(y) = \begin{bmatrix} \cos\theta_y & -\sin\theta_y & 0 \\ \sin\theta_y & \cos\theta_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

式(1)，式(2)，式(3)の回転行列を用いると，前回の位置からロール，ピッチ，ヨーにそれぞれ$\theta_r, \theta_p, \theta_y$回転した現在の位置を求める式(4)が成り立つ．

$$\overrightarrow{p'} = R(r)R(p)R(y)\,\vec{p} \tag{4}$$

式(4)の 3 軸の回転行列を合成すると式(5)のように表すことができる．

$$\overrightarrow{p'} = \begin{bmatrix} C_pC_y & S_rS_pC_y - S_yC_r & S_pC_rC_y + S_rS_y \\ S_yC_p & S_rS_pS_y + C_yC_r & S_pS_yC_r - S_rC_y \\ -S_p & S_rC_r & C_rC_p \end{bmatrix} \vec{p} \tag{5}$$

(ただし，$S_x = \sin\theta_x$，$C_x = \cos\theta_x$，x : r, p, y)

式(5)を用いて風速修正のプログラムを開発する．以下に本研究で開発した MPU9250 の測定，風速修正のプログラムを示す[8]．

- MPU9250 測定

```
import numpy as np
import FaBo9Axis_MPU9250
import time
import datetime
import sys
```

```
#=====INPUT starts here=================
mpu9250 = FaBo9Axis_MPU9250.MPU9250()
save_folder='output'
save_file='mpu9250_output'
save_file=save_folder+'/'+save_file+'.csv'

site="Kindai Department of Engineering G building rooftop"
device='mpu9250' # Name of the device
sampling_freq=10.0 # [Hz]
g_accel=9.80665 # acceleration due to gravity [m/s^2]
#=====INPUT ends here=====================

header_write1='Site: '+ site
header_write2='Device: '+device+', Sampling frequency= '+ str(sampling_freq)
with open (save_file,'w') as f:
    f.write(header_write1+'¥n')
    f.write(header_write2+'¥n')
    f.write('date time [] ,accel x [m/s^2], accel y [m/s^2], accel z [m/s^2], gyro x [deg/s], gyro y [deg/s], gyro z [deg/s], compass x [uT], compass y [uT], compass z [uT]'+'¥n')

try:
    while True:
        accel = mpu9250.readAccel()
        ax=accel['x']*g_accel; ay=accel['y']*g_accel; az=accel['z']*g_accel; # convert to SI unit from measured values in g
        #print("accel x = ",ax,"    accel y = ",ay,"    accel z = ",az)

        gyro = mpu9250.readGyro()
        gx=gyro['x']; gy=gyro['y']; gz=gyro['z']
        #print("gyro x    = ",gx,"    gyro x    = ",gy,"    gyro x    = ",gz)

        mag = mpu9250.readMagnet()
        mx=mag['x']; my=mag['y']; mz=mag['z']
        #print ("compass x = ",mag['x'],"    compass y = ",mag['y'],"    compass z = ",mag['z'])

        current_time=datetime.datetime.now()
        current_time_write=current_time.strftime("%Y/%m/%d %H:%M:%S.%f")
```

```
            time.sleep(1.0/sampling_freq)

            save_dat=np.array([[current_time_write,ax,ay,az,gx,gy,gz,mx,my,mz]])
            with open (save_file,'ab') as f:
                np.savetxt(f,save_dat,fmt='%s,%s,%s,%s,%s,%s,%s,%s,%s,%s')

except KeyboardInterrupt:
        sys.exit()
```

- 風速修正

```
import numpy as np
from os import path # e.g. to join one or more path components
import datetime # for reading and writing date and time
import os
import warnings # to suppress warning at specific place
import sys
import re # to rearrange number

#=====INPUT starts here=================
read_folder='output'
read_file="mpu9250_output_20201215_20201215" #
file_end=".csv"
read_file=read_folder+'/'+read_file+file_end

header=3 # Number of header lines
footer=0 # Number of footer lines

time_fmt="%Y/%m/%d %H:%M:%S.%f" # "%Y/%m/%d %H:%M"
site="Kindai Department of Engineering G building rooftop"
device='mpu9250' # Name of the device
sampling_freq=10.0 # sampling frequency of the original measured data [Hz]

avg_window= 1.0 # averaging time [second]
save_folder='output'

save_file='mpu9250_output'
file_end='.csv'
```

```
g_accel=9.80665 #重力加速度[m/s^2]

st_date= datetime.datetime(2020,12,15,5,33)
en_date= datetime.datetime(2020,12,15,5,35)

time_on_ground=datetime.datetime(2020,12,15,5,34) # Time until which sensor is on the ground

#=====INPUT ends here=====================


data = np.genfromtxt(read_file,skip_header=header,skip_footer=footer,delimiter=",",dtype=str) #csv ファイルの読み込み

# 0 col: time; 1st col:ax; 2nd col:ay; 3rd col: az; ....acceleration
# 4th col:gx; 5th col:gy; 6th col: gz; ...gyro
# 7th col:mx; 8th col:my; 9th col: mz; ...magnet

date_time = [datetime.datetime.strptime(x,time_fmt) for x in data[:,0]]
date_time = np.array([date_time]).T

data=np.delete(data,np.s_[0:1],axis=1) # delete the time column
data = data.astype(np.float) # Convert data from string to float
data = np.append(date_time,data,axis=1)

date_time=data[:,0] #1 列目 datetime 値
ax=data[:,1] #2 列目 ax 値
ay=data[:,2] #3 列目 ay 値
az=data[:,3] #4 列目 az 値
gx=data[:,4] #5 列目 gx 値
gy=data[:,5] #6 列目 gy 値
gz=data[:,6] #7 列目 gz 値
mx=data[:,7] #8 列目 mx 値
my=data[:,8] #9 列目 my 値
mz=data[:,9] #10 列目 mz 値

#Substract g from the vertical acceleration when the MPU is on the ground.
j_on_ground = (np.abs(date_time[:]-time_on_ground)).argmin()
```

```
for j in range(j_on_ground+1):
    az[j]=az[j]-g_accel

#---mag 値(x,y,z)が 0 のとき 1 つ前の値に置き換える(最初のデータが 0 のときは次の値に置き換える)---
for j in range(data.shape[0]):
    if j==0 and mx[j]==0:
        mx[j]=mx[j+1]
    elif mx[j]==0:
        mx[j]=mx[j-1]

    if j==0 and my[j]==0:
        my[j]=my[j+1]

    elif my[j]==0:
        my[j]=my[j-1]

    if j==0 and mz[j]==0:
        mz[j]=mz[j+1]
    elif mz[j]==0:
        mz[j]=mz[j-1]


#---Time averaging
print('====The time averaging')
#---Set round the start & end time to minute
st_date_file                                                    =date_time[0]-
datetime.timedelta(seconds=date_time[0].second,microseconds=date_time[0].microsecond)
en_date_file        =date_time[-1]+datetime.timedelta(seconds=60-date_time[-1].second,microseconds=-
date_time[-1].microsecond)
#print('st_date_file',date_time[0],st_date_file)
#print('en_date_file',date_time[-1],en_date_file)

if (en_date_file-st_date_file).total_seconds()<avg_window:
    print("Numbe of measured data less than averaging window time of "+ str(avg_window)+" sec")
    sys.exit()

#---Some allocations
```

```
Navg = int((en_date_file-st_date_file).total_seconds()/avg_window)
time_avg =np.empty(Navg,dtype=object)
ax_avg =np.zeros(Navg)
ay_avg =np.zeros(Navg)
az_avg =np.zeros(Navg)
gx_avg =np.zeros(Navg)
gy_avg =np.zeros(Navg)
gz_avg =np.zeros(Navg)
mx_avg =np.zeros(Navg)
my_avg =np.zeros(Navg)
mz_avg =np.zeros(Navg)

win_en_date = st_date_file
for n in range(Navg):
    win_st_date=win_en_date
    win_en_date=win_en_date+ datetime.timedelta(seconds=avg_window)
    i_st = (np.abs(date_time[:]-win_st_date)).argmin() # index window start time
    i_en = (np.abs(date_time[:]-win_en_date)).argmin() # index window end time

    if win_en_date<date_time[i_st]:
        time_avg[n]   =win_en_date
        ax_avg[n]   =0.0
        ay_avg[n]   =0.0
        az_avg[n]   =0.0
        gx_avg[n]   =0.0
        gy_avg[n]   =0.0
        gz_avg[n]   =0.0
        mx_avg[n]   =0.0
        my_avg[n]   =0.0
        mz_avg[n]   =0.0
        continue

    #---some checks
    if date_time[i_st]<win_st_date:
        i_st = i_st+1
    if date_time[i_en]>win_en_date:
        i_en = i_en-1
    if i_en<i_st:
```

```
            time_avg[n]    =win_en_date
            ax_avg[n]    =0.0
            ay_avg[n]    =0.0
            az_avg[n]    =0.0
            gx_avg[n]    =0.0
            gy_avg[n]    =0.0
            gz_avg[n]    =0.0
            mx_avg[n]    =0.0
            my_avg[n]    =0.0
            mz_avg[n]    =0.0
            print('No data available at',time_avg[n])
            continue

        time_avg[n] = win_en_date
        #print(time_avg[n].strftime("%Y/%m/%d  %H:%M:%S"),'        ndata=',(i_en-i_st)) # to estimate
ndat_max for availability. usually commented
        ax_avg[n]    =np.mean(ax[i_st:i_en+1])
        ay_avg[n]    =np.mean(ay[i_st:i_en+1])
        az_avg[n]    =np.mean(az[i_st:i_en+1])
        gx_avg[n]    =np.mean(gx[i_st:i_en+1])
        gy_avg[n]    =np.mean(gy[i_st:i_en+1])
        gz_avg[n]    =np.mean(gz[i_st:i_en+1])
        mx_avg[n]    =np.mean(mx[i_st:i_en+1])
        my_avg[n]    =np.mean(my[i_st:i_en+1])
        mz_avg[n]    =np.mean(mz[i_st:i_en+1])


print("")
print('====Write average data to a file')
save_dat                                                                                                =
np.array([time_avg[:],ax_avg[:],ay_avg[:],az_avg[:],gx_avg[:],gy_avg[:],gz_avg[:],mx_avg[:],my_avg[:],
mz_avg[:]]).T
save_dat[:,0] =[x.strftime("%Y/%m/%d %H:%M:%S") for x in save_dat[:,0]] # reformat time

save_file=save_folder+'/'+save_file+'_'+str(int(avg_window))+'sec_avg'+'_from'+st_date.strftime("%Y-%
m-%d-%H-%M-%S")+'_to_'+en_date.strftime("%Y-%m-%d-%H-%M-%S")+file_end
header_write1='Site: '+ site
header_write2='Device: '+ device+', Original sampling frequency= '+ str(sampling_freq)+', Average
```

```
window: '+ str(avg_window)+ ' second'
with open (save_file,'w') as f:
    f.write(header_write1+'¥n')
    f.write(header_write2+'¥n')
    f.write('date time [] ,accel x [m/s^2], accel y [m/s^2], accel z [m/s^2], gyro x [deg/s], gyro y [deg/s],
gyro z [deg/s], compass x [uT], compass y [uT], compass z [uT]'+'¥n')

with open(save_file,'ab') as f:
    np.savetxt(f,save_dat[:,:],fmt='%s,%s,%s,%s,%s,%s,%s,%s,%s,%s')
#---

'''

R=gxa
P=gya
Y=gza

if R==-90 or R==0 or R==90 :
    Sr=round(np.sin(R*np.pi/180))
    Cr=round(np.cos(R*np.pi/180))

else :
    Sr=np.sin(R*np.pi/180)
    Cr=np.cos(R*np.pi/180)

if R==-90 or P==0 or P==90 :
    Sp=round(np.sin(P*np.pi/180))
    Cp=round(np.cos(P*np.pi/180))

else :
    Sp=np.sin(P*np.pi/180)
    Cp=np.cos(P*np.pi/180)

if R==-90 or Y==0 or Y==90 :
    Sy=round(np.sin(Y*np.pi/180))
    Cy=round(np.cos(Y*np.pi/180))

else :
```

```
    Sy=np.sin(Y*np.pi/180)
    Cy=np.cos(Y*np.pi/180)

RXx=Cp*Cy
RXy=Sy*Cp
RXz=-Sp
RYx=Sr*Sp*Cy-Sy*Cr
RYy=Sr*Sp*Sy+Cr*Cy
RYz=Sr*Cp
RZx=Sp*Cr*Cy+Sr*Sy
RZy=Sp*Sy*Cr-Sr*Cy
RZz=Cr*Cp

BXx=AXx*RXx+AYx*RXy+AZx*RXz
BXy=AXy*RXx+AYy*RXy+AZy*RXz
BXz=AXz*RXx+AYz*RXy+AZz*RXz
BYx=AXx*RYx+AYx*RYy+AZx*RYz
BYy=AXy*RYx+AYy*RYy+AZy*RYz
BYz=AXz*RYx+AYz*RYy+AZz*RYz
BZx=AXx*RZx+AYx*RZy+AZx*RZz
BZy=AXy*RZx+AYy*RZy+AZy*RZz
BZz=AXz*RZx+AYz*RZy+AZz*RZz

'''
```

2.4 気球を用いた風況計測

図 2.25 に本研究で用いた高空風況計測システムの概要を示す．気球の挙動は 9 軸 IMU センサー(MPU9250)を用いて計測した．IMU センサーを Raspberry Pi に接続し，サンプリング周波数 1 Hz で計測した加速度，角速度，地磁気を抽出した．計測用のプログラムは Python を用いて構築した．気球に風速計，IMU センサーを取り付け，空中に浮揚させる．ただし，IMU センサー，風速計は風の影響を受けても同じ傾き，回転となるように直線的に固定した．風速計は超音波風速計(YOUNG モデル 81000)を用いた．

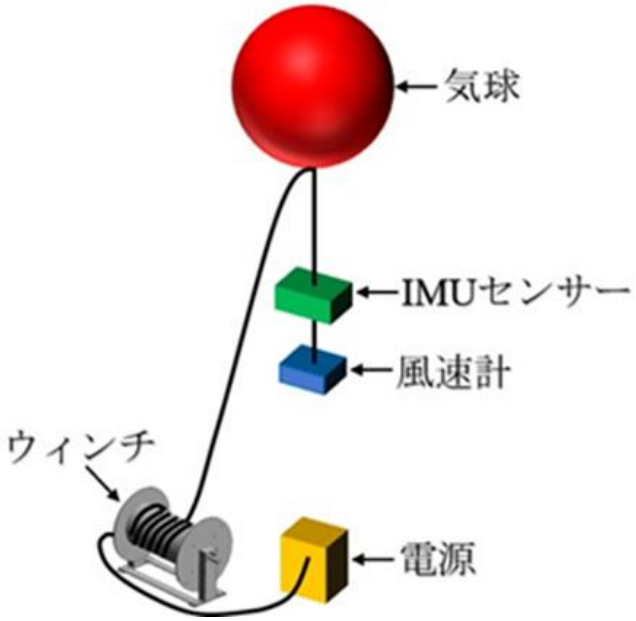

図 2.25　高空風況計測システム

以下に高空風況計測の実験手順をまとめる.

- 高空風況計測実験手順

(1) 気球のひもをウィンチに巻き付ける.

(2) 気球にヘリウムガスを入れる．本研究では直径 2.3m，ガス容量 7.3m^2 である気球を用いた.

(3) Raspberry Pi に接続した IMU センサーを箱の中に固定して入れ，ふたを閉める．IMU と風速計を直線的に固定するために中心に穴をあけ，塩化ビニル管を通して箱の上下をねじで固定する.

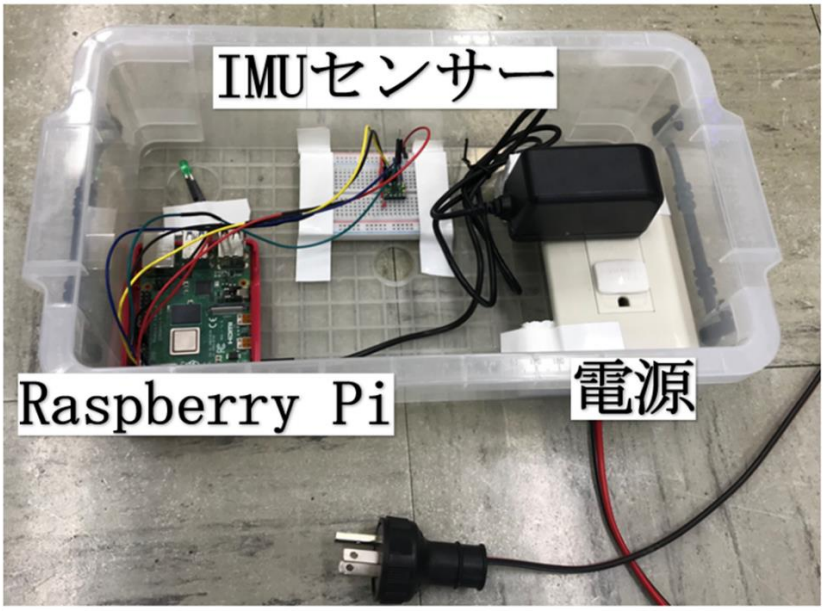

図 2.26　IMU 収納箱機材

(4) Raspberry Pi を起動し，測定用のプログラムを起動させる．加速度，角速度，地磁気のオフセット値を数分計測し，プログラムを終了して結果を保存する．

(5) 再び Raspberry Pi で測定用のプログラムを起動させる．浮揚中は電源を箱の中に繋ぎ，Raspberry Pi を起動状態にしてプログラムが終了しないようにする．

(6) 風速計と IMU センサーを気球の下に取り付ける．本研究ではひもで風速計と IMU センサーを固定した. IMU センサーは箱を固定させた塩化ビニル管のねじ部分にひもを通して固定させた．

図 2.28　風速計-IMU センサー固定方法

(7) 気球を徐々に浮揚させて，計測を開始する．

(8) 一定時間経過後，気球を回収する．

(9) 測定用のプログラムを終了して，結果を保存する．

(10)計測値を用いて実際の風速値に修正する．

第 3 章　実験結果および考察

3.1 RTK 測位結果

単独測位はできたが，RTK 測位ができなかった．図 3.1 に RTK 測位結果を示す．移動局の衛生受信は表示されているが，基準局の衛生受信は表示されていない．衛生の受信が途切れ途切れであるのか，リアルタイムでの座標は計測できず，瞬間における座標が計測された．
RTKNAVI で基準局，移動局をルーターに LAN ケーブルで接続した状態で，コマンドプロンプトにて互いの情報伝送状態を確認したが，異常がなかった．よって，各プログラムでの設定に不備がある，もしくは STRSVR から伝送した現在時間が RTKNAVI 内で一致していないことが原因であると考えられる．

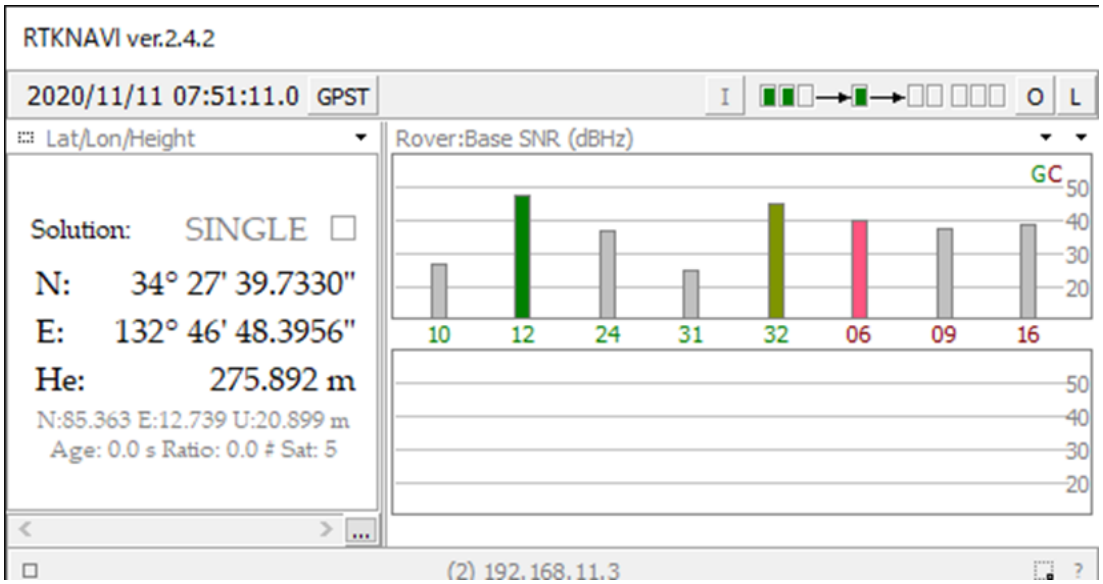

図 3.1　RTKNAVI-RTK 測位結果

3.2 風況計測結果

計測は 2021 年 1 月 9 日 14 時に行った．IMU センサーは計測できたが，風速は浮揚中の計測は強風に煽られて気球が大きく揺れ，周辺機材に衝突して割れたため，1 分ほどしか正確な計測ができていない. 図 3.2 に気球に取り付けた超音波風速計と観測タワーに設置した基準となる超音波風速計による風速計測データの比較を示す．比較にした基準となる超音波風速計はで屋上に設置した YOUNG モデル 81000 を用いた．

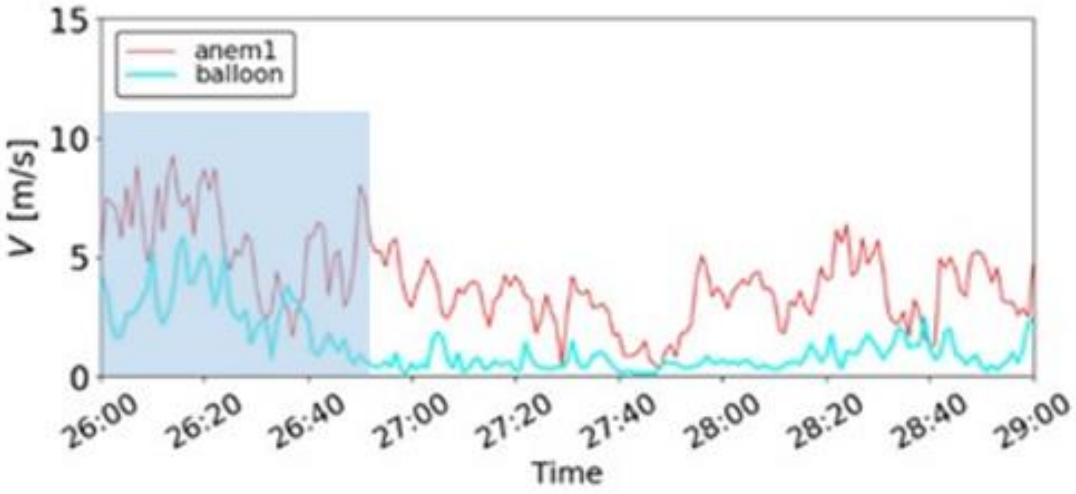

図 3.2　風速計比較．浮上期間：26:00～26:50.

図 3.3　超音波風速計(YOUNG モデル 81000)

気球が浮上している期間，基準の風速計と気球に取り付けられている風速に大きな差があるものの，両風速計が似た傾向を示す．気球に取り付けた風速計は風によって測定軸が変わって分散した値を検知するため，基準値より小さい値が計測された．それ以降は気球が割れて風速計も落下したため，気球に取り付けた風速計が正確に計測出来てない．気球に取り付けた風速計の値は IMU センサーの計測値から風速を修正することで基準となる風速値により近い値となることが予想される．

第 4 章　結言

本研究では発電予測における風況計測を低コストで行うことを目的としている．気球による低コスト風速計測システムの開発を行った，気球の位置から風速を修正する方法を提案し，RTK 測位と IMU センサーでの風況計測を検証した．IMU センサーに関しては IMU センサー(MPU9250)での加速度，角速度，地磁気計測，計測値から風速を修正するプログラムを Python で開発した．

検証結果として，RTK 測位に関して，単独測位は成功したが RTK 測位は失敗した．IMU センサーでの風況計測に関しては短時間での計測に成功した．IMU センサーの計測値から風速を修正することで正しい風速値となることが予想されたが，気球が割れて落下したためデータ数が少なく，定量的な評価ができていない．また，風によって気球が予想よりも大きく揺れるなど問題も発覚した．

今後の課題としては，長期間の計測を行い，提案した手法の検証を行うことである．そのために，RTK 測位は基準局，移動局の情報を同一時刻によるものとするなどして RTK 測位を行う必要がある．IMU センサーは気球の形を変形させ風の抵抗を減らす，気球をウィンチ以外の他の固定地点を設けるなどして，気球の挙動を安定させる必要があると考える． Python のプログラムに関しては研究を進めるにあたり，改良が必要である．

謝辞

本研究にあたり，高空風況計測システム，プログラム開発に関して，ジェイ・プラカス・ゴイト助教，田口誠景様，使用した実験装置製作に関して，梅西浩二様にご指導いただきました．ここに記して謝辞とさせていただきます.

参考文献

1) 宇都宮 智昭　他，浮体式洋上風力発電の実用化に向けて−五島市椛島に おける実証事業-(2014)

2) MST エムエスツデー2013 年 10 月号 お客様訪問記：https://www.m-system.co.jp/mstoday/backnum/2013/10/interview/index.html (参照：2020-1-12)

3) Assessment of a balloon-borne buoy based measurement system with co-located tower and lidar measurements (2015)

4) トランジスタ技術 2018 年 1 月号 地球大実験ピタリ 1cm！新 GPS 誕生［CD 付き］(2018)

5) 川崎 駿，傾斜地果樹園における生育モニタリングシステムの開発-位置測定方法の検討-(2019)

6) Qiita Rasberry pi 3 でストロベリー・リナックス社製の「MPU-9250　9 軸センサモジュール（メーカー品番：MPU-9250）」を使う：https://qiita.com/boyaki_machine/items/915f7730c737f2a5cc79 (参照 2020-11-17)

7) Watako-Lab．3 次元ベクトルの回転「ロール・ピッチ・ヨー」：https://watako-Lab.com/2019/01/23/roll_pitch_yaw/ (参照 2020-11-17)

8) IMU センサーを RaspberryPi とインターフェースする方法：https://maker.pro/raspberry-pi/tutorial/how-to-interface-an-imu-sensor-with-a-raspberry-pi (参照 2020-11-17)

令和二年度　卒業研究論文

高空風況計測用気球の挙動評価システムの開発

長久　泰樹