卒業研究報告書

題目

滑空回収型小型気球への搭載を目的とした

防水軽量型マルチデータロガーBOX の開発

報 告 者

学籍番号:1240179

氏名:渡部　瑠斗

指 導 教 員

山本　真行　教授

令和 6 年 2 月 16 日

高知工科大学　システム工学群　電子・光工学専攻

目次

第一章　序論

1.1 背景

1.1.1 滑空回収型小型気球を使った気球実験について

滑空回収型小型気球は図 1.1 のような作りになっており、パラフォイルに制御装置をつけた誘導回収型観測機を気象観測気球から吊るしたものである。

これを図 1.2 のように地上から放球し、目標の高度に達したら気球とパラフォイルを切り離し、制御装置でパラフォイルを操ることで上空の風に任せつつも航路の一部を制御して回収可能な範囲で海に着水させ、船で回収し再利用する。

滑空回収型小型気球に観測装置を搭載することにより、空中での長い観測時間、再利用による低コスト化、エンジンの排気などによる大気の擾乱や騒音がない、障害物の少ない場所に落とすことができるなどの利点が得られる。

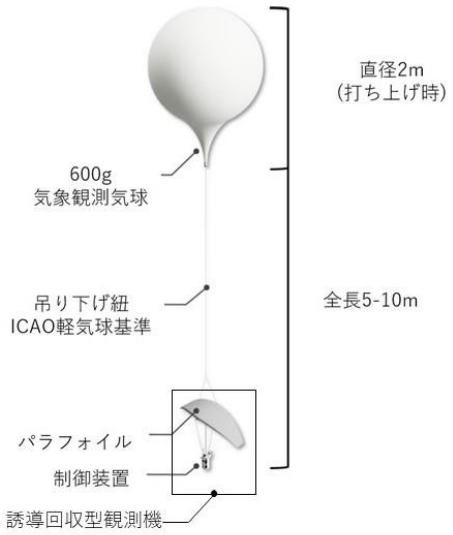

図 1.1　滑空回収型小型気球の全体図[1]

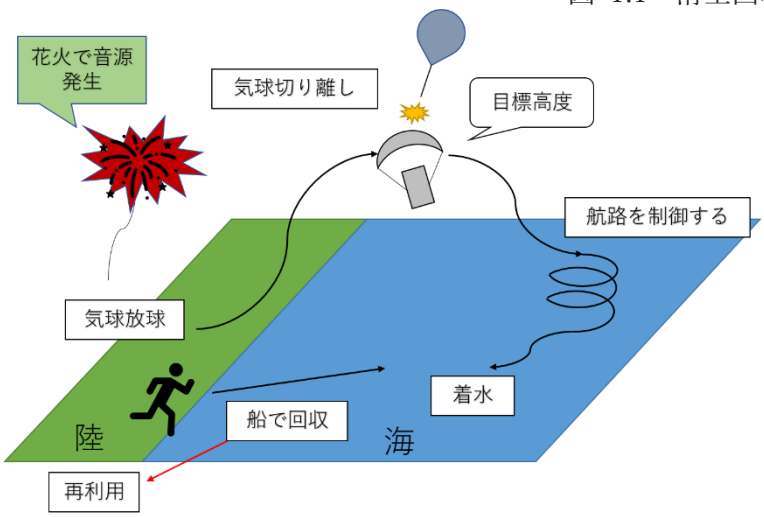

図 1.2　気球実験概要

1.1.2 インフラサウンドについて

インフラサウンドとは周波数 20 Hz 以下の人間の可聴域以下の音波のことであり、インフラサウンドの特徴として長距離伝搬性が挙げられる。著者の所属する高知工科大学システム工学群宇宙地球探査システム研究室（以下、山本研）では 2022 年 1 月 15 日に 8,000 km 離れた南太平洋トンガのフンガ・ドンガーフンガ・ハウパイ火山の噴火による音波を日本の各地で観測した[2]。

インフラサウンドは地震・噴火・津波・雷鳴などの大規模な自然現象や、人為的爆発現象

により発生するため、インフラサウンドを観測・解析することにより災害の検知につながることが期待できる。インフラサウンドの観測方法として本研究では株式会社 SAYA 製の小型センサ INF03S(図 1.3)を使用する。

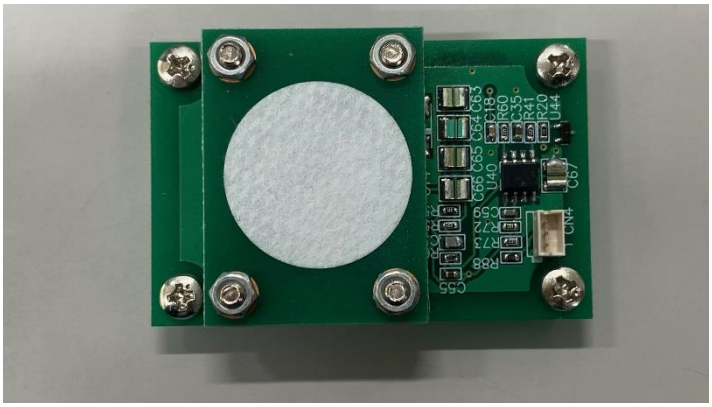

図 1.3　INF03S の外観

1.1.3 音波の伝達経路の解析

音波を解析する際、短距離伝搬の場合は直線伝搬を仮定し解析できるが、長距離伝搬の場合は図 1.4 のような三次元での伝達を考慮して解析する。三次元での解析では音速や風速の高度分布も踏まえて理論計算をするが、観測値と一致しない事例もあるため、実際に空中で音波を観測することで、より詳しい三次元での音波の伝達の理解が期待できる。

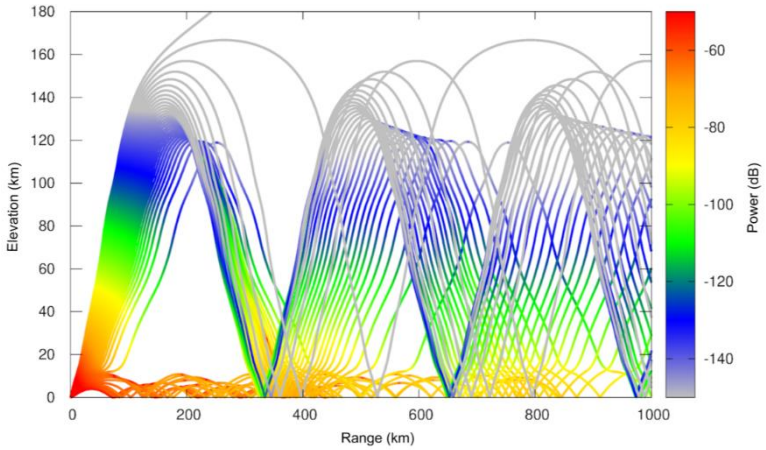

図 1.4　三次元での音波伝搬・減衰の一例[3]

1.2 目的

本研究では 1.1.3 項で述べた内容の理解のため、滑空回収型小型気球実験に興味を持ってくださった福岡大学の高島氏提供の NO2 センサ（Alphasense 製 NO2-B43F)、およびインフラサウンドセンサを気球に載せることにより同時に大気中での音波伝搬の仕方の理解に加え、大気中のその場における情報を得ることを目指す。

大気中の情報を得るための手法として、上記のセンサ類を搭載でき、システム面で小型気

球装置とは独立した防水軽量型データロガーBOX を作り、STRVSN 社が製作している滑空回収型小型気球用の誘導回収型観測機(図 1.5)に取り付ける。そして図 1.2 のように気球実験中に音源発生用の花火を打ち揚げ、上空の目標高度において防水軽量型データロガーBOX 内のセンサで観測する方法を採用する。

本研究では気球搭載用の防水軽量型データロガーBOX の開発と動作検証を目的とする。気球搭載用の防水軽量型データロガーBOX(以下 BOX)の必要な機能を以下に示す。また BOX は図 1.6 のように誘導回収型観測機の制御装置の前方に取り付ける。

1. 質量 350 g 以内
2. 縦 150 mm × 横 107 mm × 高さ 64 mm 以内
3. 長時間のフライト(3 時間程度)でも運用可能なバッテリーとシステムの選定
4. 海に着水させても BOX 内のシステムが水没しないための防水性能
5. BOX 内の各センサが必要なデータを取得できる程度の空気や音圧の提供

また、BOX が回収できなかった場合に備えて通信モジュールを使った地上へのデータのリアルタイム転送も実装したいと考えている。

図 1.5　STRVSN 製の滑空回収型小型気球用の誘導回収型観測機の外観

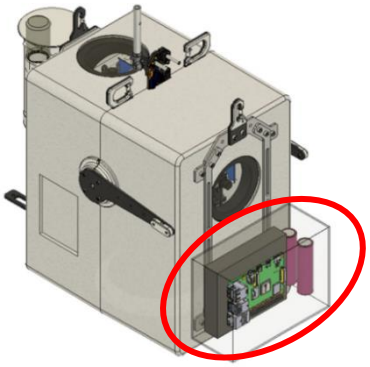

図 1.6　防水軽量型データロガーBOX の誘導回収型観測機用制御装置への取り付け位置(赤丸)[1]

第二章　実験方法

2.1 ドローンを用いた誘導回収型観測機の滑空実験における防水軽量型データロガーBOX の動作確認実験

2.1.1 防水軽量型データロガーBOX のシステムについて

図 2.1 に動作確認実験で使用する BOX の外観を示す。BOX の中は図 2.2 のようになっておりシステムは市販のリチウム一次電池(CR123A, 3.0 V)を 3 つ直列につなぎ 9 V にて動作させる。

BOX の外装(図 2.3)は 3D プリンター(FLASHFORGE 製 Creator Pro2)を用いて ABS フィラメントを使い、充填率 15%の設定で制作した。外装は 2 つの部屋から出来ており、左の部屋には図 2.4 に示すガスセンサの機器類が入り、右の部屋には図 2.5 に示すインフラサウンドセンサ及びデータロガーの機器類が入る。BOX 全体の質量は 320 g である。

図 2.1　防水軽量型データロガーBOX の外観

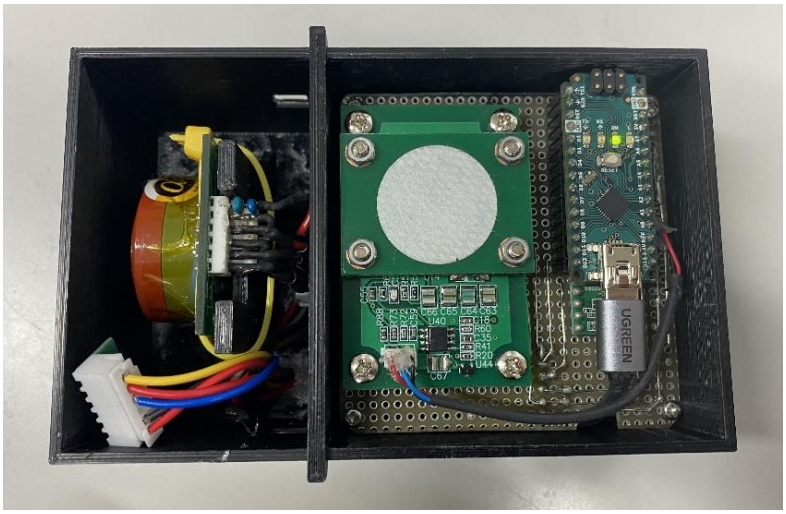

図 2.2　防水軽量型データロガーBOX の中の実装状態

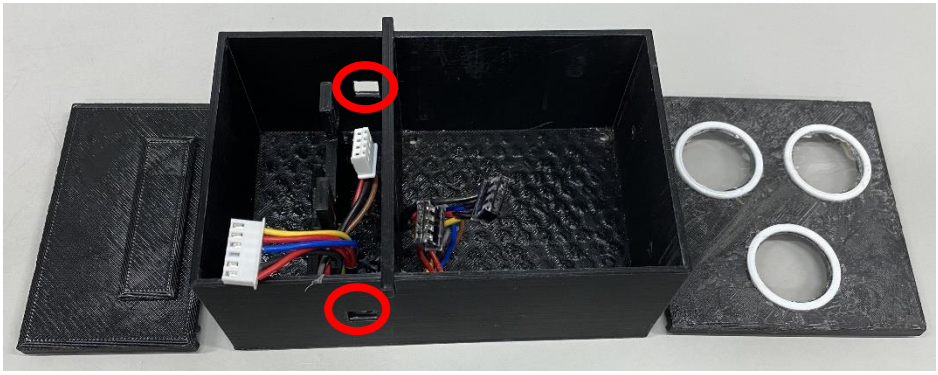

図 2.3　BOX 外装の筐体。BOX の筐体はガスセンサへの空気の供給が出来るよう赤丸のところに四角い穴が開いている部屋(左)とその他の機器への防水のため密閉されている部屋(右)に別れている。

図 2.3 の左の部屋には外気の情報からデータを得る NO2 センサ(NO2-B43F)と温湿度・気圧センサモジュール(株式会社秋月電子通商製 AE-BME280)が入っており、空気の入れ替えが出来るように図 2.3 の赤丸の部分のように壁の左右に 1cm 四方の穴を設けた。

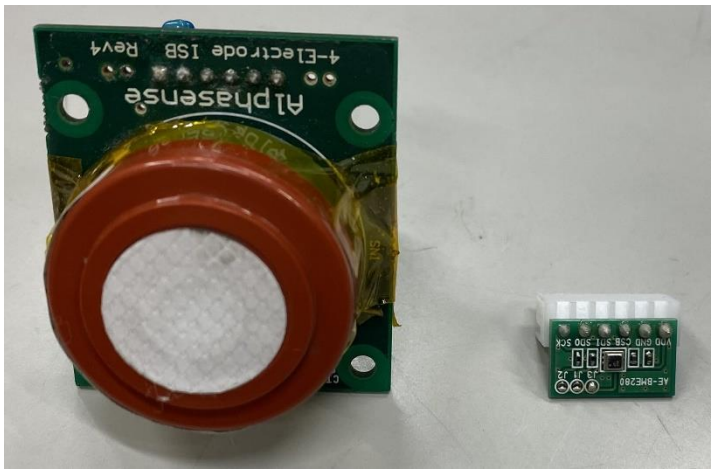

図 2.4　図 2.3 の左の部屋に入っている機器類の外観。左 : NO2 センサ(NO2-B43F)
右 : 温湿度・気圧センサモジュール(AE-BME280)

図 2.3 の右の部屋には分子レベルでの外気情報を必要としない INF03S やデータロガーシステムを動かすための機器類(図 2.5)が二段の層状になって入っている。

気球実験で誘導回収型観測器を海に着水させた際に BOX 内の主要電子部品であるマイコン(Arduino Holding 制 Arduino Nano)や microSD カードが水没すれば上空で観測した貴重なデータが紛失する可能性がある。また繰り返し BOX を使えるようにするために非防水(左)と防水(右)の部屋で隔離する構造とした。蓋を閉めた時の蓋と箱の隙間は今回の実験ではテープを巻くことで水が入らないようにした(図 2.1)。

右の部屋の蓋には 3 つの穴(図 2.1)が開いており、その穴にビニールフィルムをシーリングしてある。これは BOX が固い壁で出来ていたら空気中を伝搬してきたインフラサウンドの音波が減衰し、BOX 内の INF03S まで届かないと考えたからであり、一部ビニールフィルムで柔らかい部分を作ることにより BOX 内まで音波が届くように配慮した。

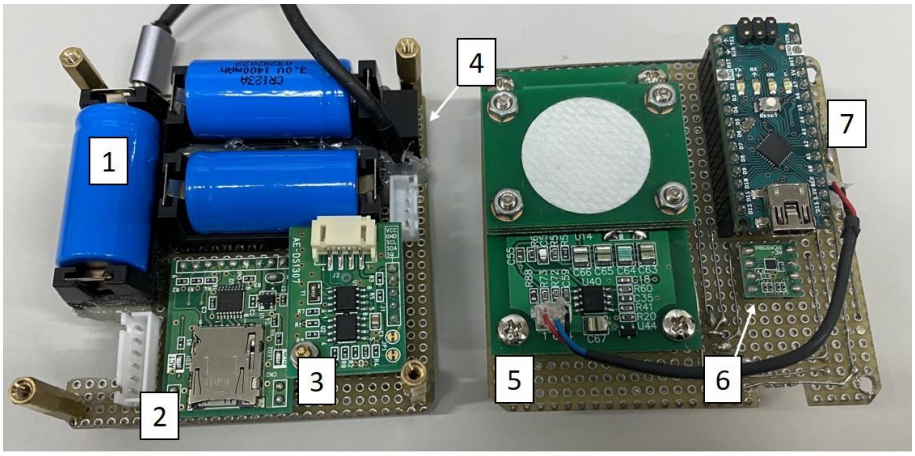

図 2.5 図 2.3 の右側に入っている機器類の外観(左 : 一段目　右 : 二段目)。

1. リチウム電池(CR123A)、2. microSD カードスロット(AE-microSD-LLCNV)
3. RTC モジュール(AE-DS1307)、4. DC-DC コンバータ(M78AR05-1)
5. INF03S 、6. I2C バス用双方向電圧レベル変換モジュール(AE-PCA9306)
7. Arduino Nano

※ 1 : Guangzhou Markyn Battery Co 製、2,3,6 : 株式会社秋月電子通商製
4 : Minmax Technology Co 製

図 2.6 は NO2 センサと AE0-BME280 の回路図である。XH コネクタ_C と XH コネクタ_B は図 2.3 の左右の部屋の仕切りに空いている穴を通して Arduino Nano マイコンにつながっている。仕切りの穴は配線を通した後にグルーガンで埋め、防水加工の部屋に水が浸入しないようにした。

図 2.7 , 2.8 は図 2.5 の基板の回路図であり、ユニバーサル基板にはんだ付けしている。

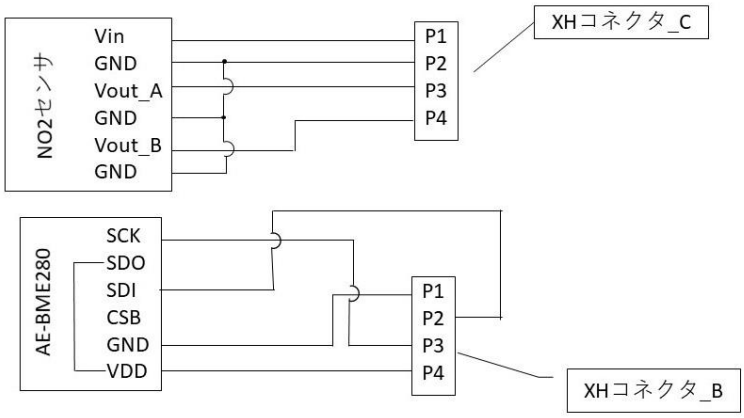

図 2.6　NO2 センサ、AE-BME280 の回路図

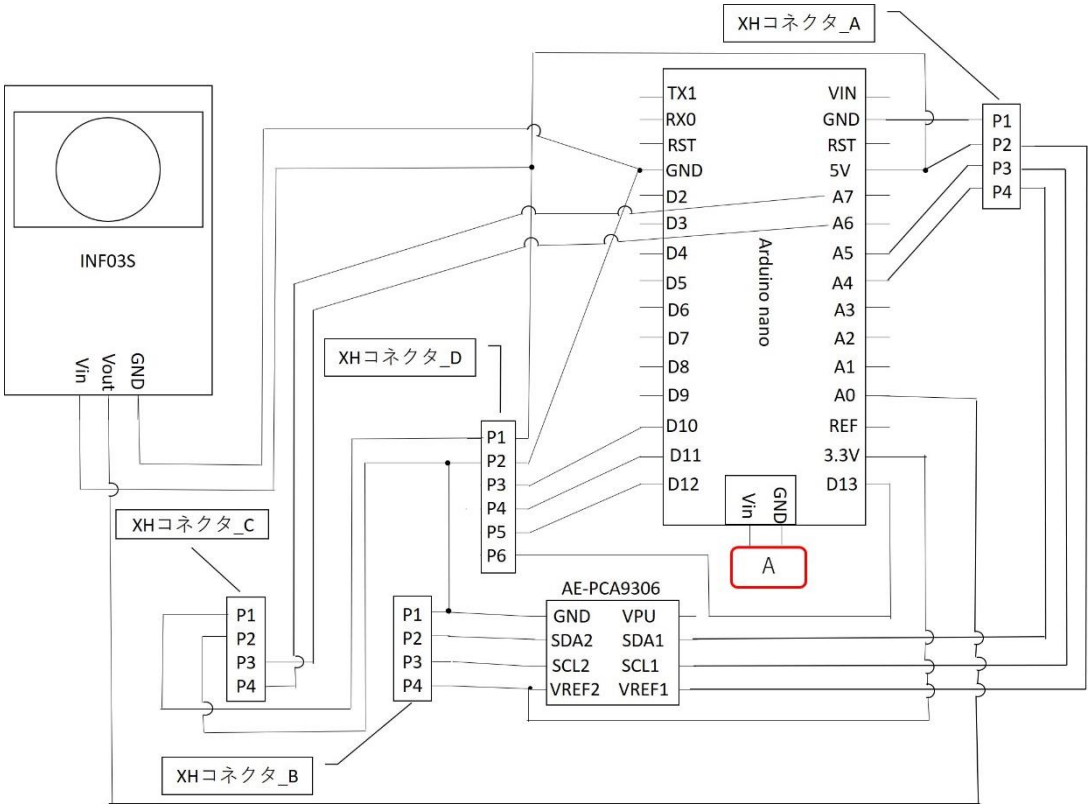

図 2.7　図 2.5 の二段目の基板の回路図（A は図 2.8 の B に接続）

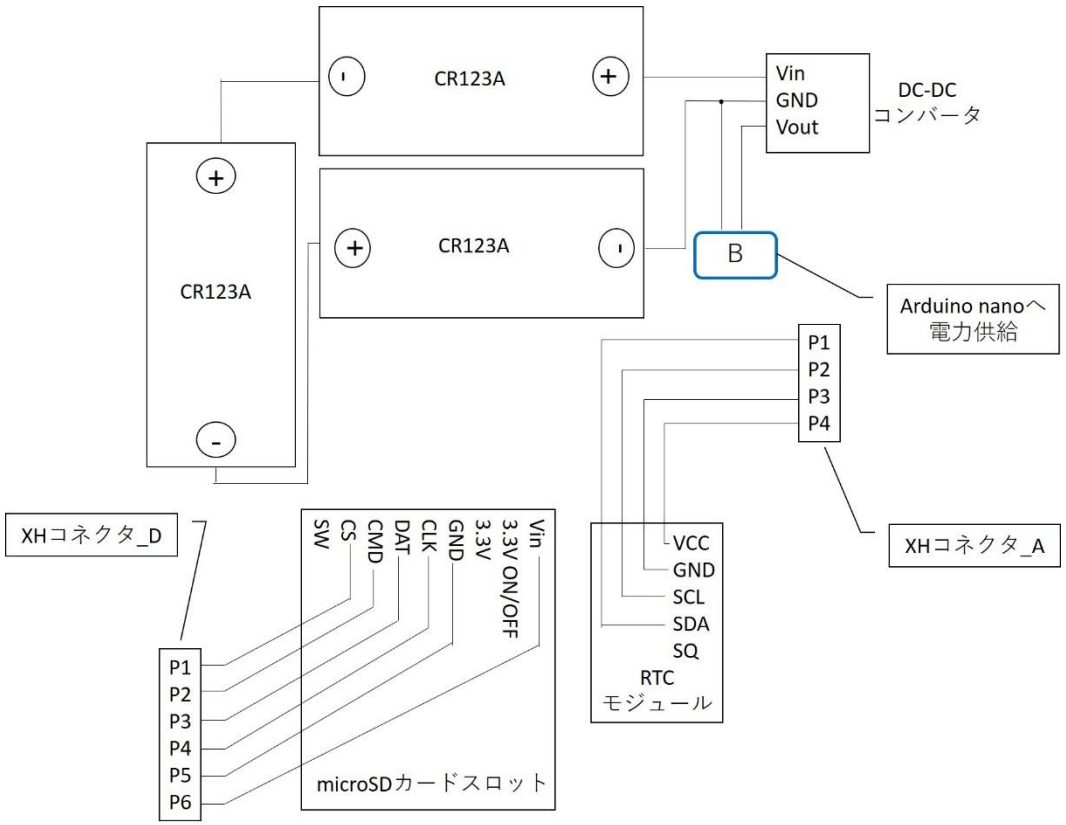

図 2.8　図 2.5 の一段目の基板の回路図(B は図 2.7 の A に接続)

2.1.2 防水軽量型データロガーBOXのプログラムについて

図 2.9 は BOX 内の Arduino Nano マイコンに実装したプログラムの簡略化したフローチャートである。Arduino Nano マイコンに書き込んだ実際のプログラムは付録に示す。

プログラムで設定した INF03S のサンプリング周波数は 40 Hz(周期 25 ms)である。サンプリング周波数を 40 Hz に設定した理由は、取得したデータの周波数成分解析のため、高速フーリエ変換(FFT)する場合、FFT で表せる周波数成分の範囲はサンプリング周波数の 1/2 の周波数(ナイキスト周波数)以下であるため、周波数 20 Hz 以下のインフラサウンドを FFT で周波数成分解析するには周波数 40 Hz 以上のサンプリング周波数でのデータ取得が必要であるためである。

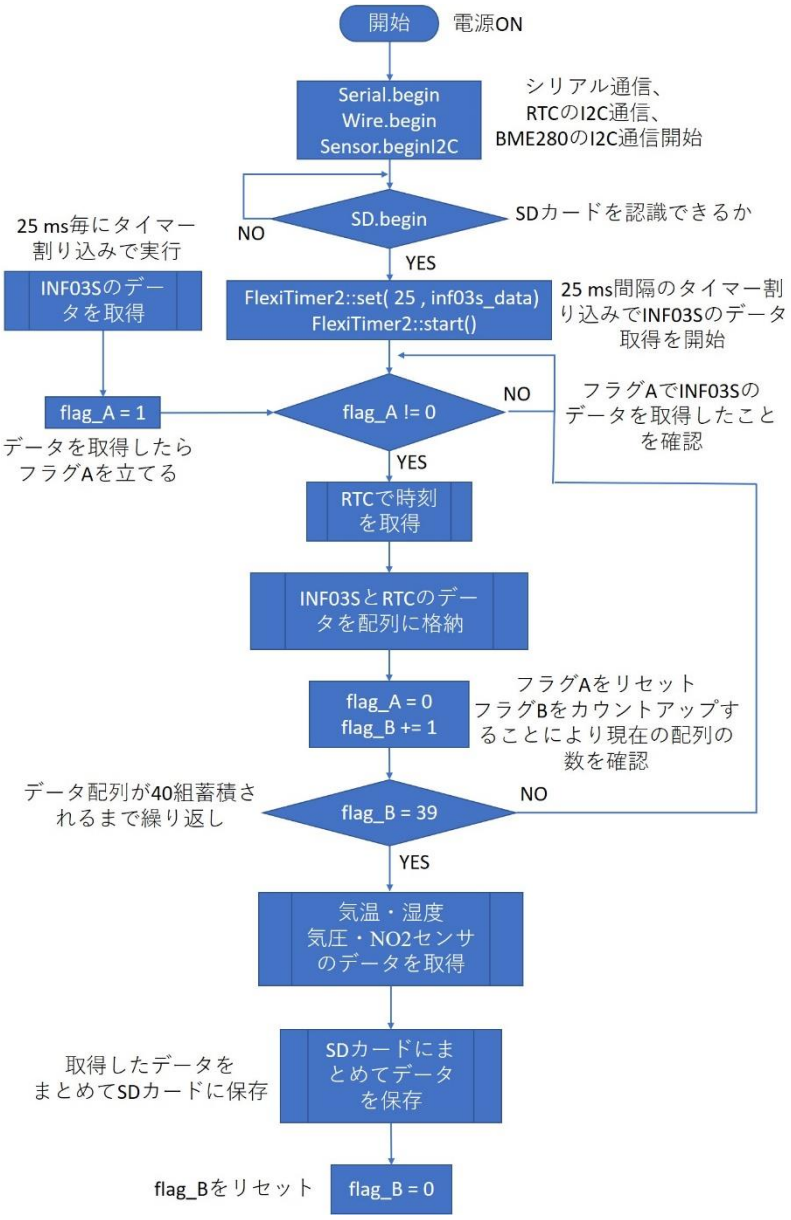

図 2.9　防水軽量型データロガーBOX の Arduino nano マイコンに書き込まれているプログラムのフローチャート

BOX の SD カードへのデータの書き込み書式を図 2.10 に示す。

時	分	秒	INF03S	気温	湿度	気圧	NO2_A	NO2_B
13	41	0	275					
13	41	0	276					
13	41	0	275					
13	41	0	276					
13	41	0	279					
13	41	0	279					
13	41	1	281					
13	41	1	280					
13	41	1	279					
13	41	1	278					
13	41	1	280					
13	41	1	280					
13	40	59	258					
13	40	59	260					
13	40	59	260	15.28	53.03	925	51	49

図 2.10　SD カードへの書き込み書式。時刻と INF03S のデータは 25 ms 毎に取得、気温・湿度・気圧・NO2 センサ(NO2_A、NO2_B)のデータは 1 s 毎に取得している。

2.1.3 防水軽量型データロガーBOX の動作確認実験

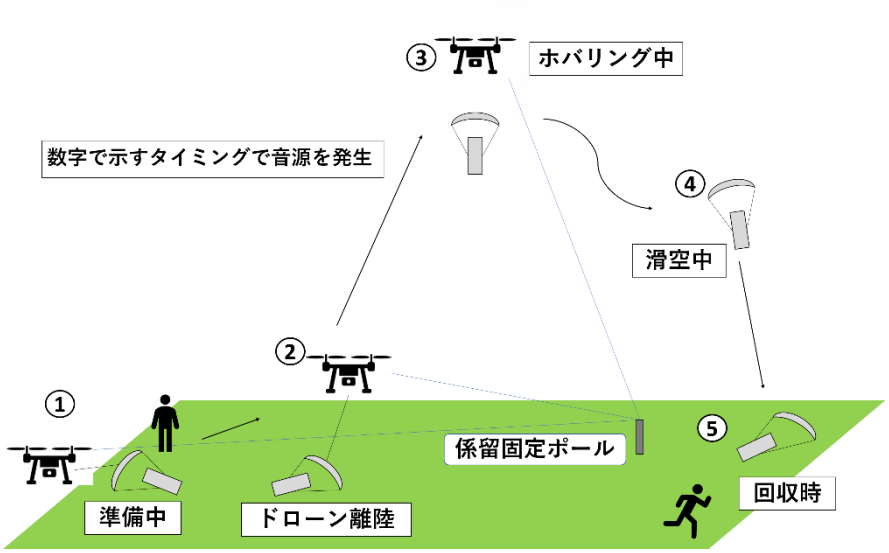

図 2.11 滑空実験の概要とインフラサウンドの音源を発生させるタイミング

実験方法は、図 2.11 に示すように図 2.1 の BOX を取り付けた滑空回収型小型気球用誘導回収型観測機(図 2.12)を係留ドローンから吊るし目標の高さまで上昇させ、ドローンから切り離し制御装置で航路を制御しながら地上に着陸させる。この実験を三回行う。

その実験中にインフラサウンドの音源を図2.11の数字で示すタイミングで発生させBOX内のインフラサウンドセンサで観測し、解析することで正確に動作出来ているか確認する。

NO2 センサの動作確認はこの実験で取得したデータを福岡大学の高島氏に送り確認して

もらうことで動作確認をする。なお、本論文では NO2 センサのデータについては議論しない。

図 2.11 で示したタイミングで発生させるインフラサウンドの発生方法を以下に示す。

・図 2.11 の音源発生方法

① おもちゃの火薬銃　※2, 3 回目の実験では手を三回たたいた
② 車(SUZUKI 製　軽乗用車 EVERY)のトランクのドアを閉め音を発生させる
③ 車のトランクのドアを閉める
④ 車のトランクのドアを閉める
⑤ 手を三回たたく

図 2.13 にデータのロギングが出来ているかの確認方法を示す。

図 2.12　防水軽量型データロガーBOX を取り付けた誘導回収型観測機の外観

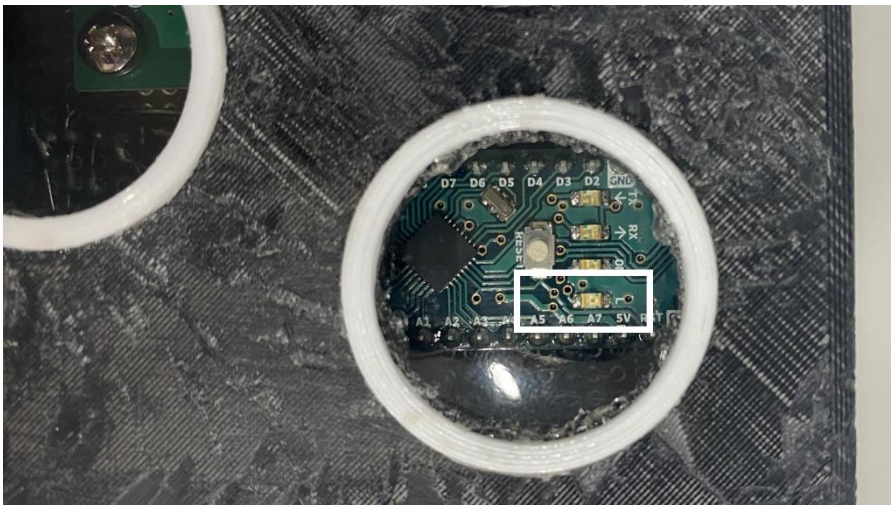

図 2.13　防水軽量型データロガーBOX のフィルム部分の外観白枠の Arduino Nano のLED が 1 秒ごとに短く光っていれば正しくロギングできている。

2.2 環境実験

2.2.1 恒温槽を使った防水軽量型データロガーBOX の動作実験

山本(2023) [1] によると 2024 年度に行う予定の飛翔実験では最高到達高度が成層圏の 15 km と記述されており、高度 15 km では気温は-55 ℃付近まで下がるため BOX は-55 ℃の条件下でも動作可能な必要がある。

上述の条件を満たすかを調べるための環境実験を図 2.14 に示す恒温槽(ESPEC 製 SH-241)を使い行う。BOX は図 2.15 のように設置した。最高高度 15 km のフライトを想定した場合、30 分間程度は-55℃の条件下で動作しなければならないため、30 分間 -55 ℃の恒温槽内で BOX を動作させ機器に問題が発生しないかを確かめた。

また、恒温槽の実験で使用した BOX 内のバッテリーが何時間まで所定の電力を維持できるかについても実験も行った。連続稼働時間の測り方は、電池切れで BOX 内システムが動かなくなるまでデータのロギングを続け、SD カードに保存されている時刻のデータから計算する。

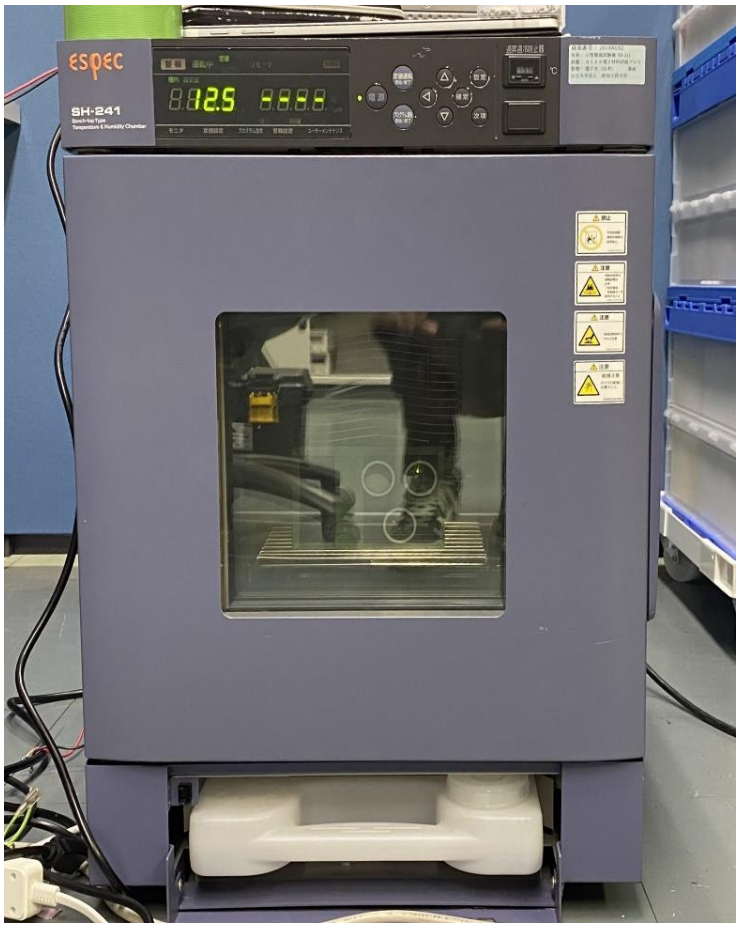

図 2.14　恒温槽(ESPEC 製　SH-241)の外観

図 2.15　恒温槽の内部と BOX を設置した外観

2.2.2 防水軽量型データロガーBOX の防水性能

BOX の防水性能を確かめる方法として、BOX の防水加工が必要な部屋にトイレットペーパーを詰め(図 2.16)蓋をし、蓋と箱の隙間から水が入らないように養生テープで固定した BOX(図 2.17)を水の入ったバケツに浸けた。しばらく時間をおいた後 BOX をバケツから取り出しトイレットペーパーが濡れているかを確認し、濡れている場合はその濡れている場所から原因を考察し、何度かフィードバックを重ねた。

図 2.16　防水性能の確認のため防水軽量型データロガーBOX の防水加工側の部屋(右)にトイレットペーパーを詰めた様子

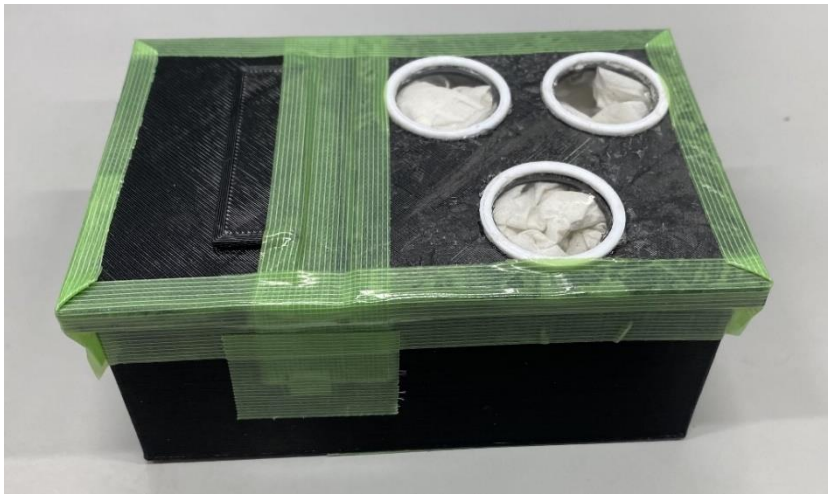

図 2.17　防水軽量型データロガーBOX 内にトイレットペーパーを詰め、蓋と箱を養生テープで固定した様子

第三章　結果

3.1 ドローンを用いた誘導回収型観測機の滑空実験における防水軽量型データロガーBOX の動作確認実験の結果

・取得した時刻データの誤り

図 3.1 は滑空実験一回目で取得したデータである。時・分・秒・INF03S の AD 変換値を見る全て 0 になっている列が 2 列あることが分かる。これは最初 2 回のデータ書き込み時に起こり、全ての滑空実験のデータの最初で確認できる。

図 3.2 では赤で囲っている部分の時刻が誤って、前の秒数の 42 秒より 2 秒早い 40 秒になっていることが分かる。

13	41	0	270					
13	41	0	270					
0	0	0	0					
0	0	0	0	15.28	53.05	925	52	50
13	41	0	271					
13	41	0	271					
13	41	0	271					

図 3.1　データの誤り事例①

左から時・分・秒・INF03S・気温・湿度・気圧・NO2_A・NO2_B

12	57	42	171					
12	57	42	171					
12	57	42	173					
12	57	42	174					
12	57	42	173					
12	57	42	172					
12	57	42	174					
12	57	42	173					
12	57	42	178					
12	57	40	125					
12	57	40	130					
12	57	40	136	12.53	57.97	925.2	47	50
12	57	42	183					
12	57	42	181					

図 3.2　データの誤り事例②

左から時・分・秒・INF03S・気温・湿度・気圧・NO2_A・NO2_B

・データ解析の方法

図 3.3 は滑空実験一回目で取得したデータであり、誘導回収型観測機が着陸するまでのデータである。右の縦軸の気圧の値がデータ番号 26000 個辺りで大きく変動しているのが分

かる。気圧が下がっている時間帯はドローンによって誘導回収型観測機の高度が上がっている時間帯であり、低い気圧が一定で保たれている時間帯はドローンが上空でホバリングしているとき、気圧が上がっているときはドローンから誘導回収型観測機が切り離され滑空中である。滑空実験のデータ結果はこの考えを元に記述していく(図 3.3 参照)。

気圧が925以上で一定	→	地上にある
気圧が922以下で一定	→	ホバリング中のドローンから吊るされている
気圧が負の傾きで下がっている	→	誘導回収型観測機が上昇している
気圧が正の傾きで上がっている	→	滑空中

図 3.3　気圧と BOX を搭載した誘導回収型観測機の位置関係

また、GPS との時刻の同期が出来てなく、上記のようなデータの誤りもあるので横軸は観測した INF03S のデータ個数で表す。INF03S のデータは 25 ms ごとの取得になっているので、データ個数間の時間経過は式(3-1)から求められる。

$$25\ \mathrm{ms} \times \text{データ番号} = \text{経過時間} \quad \cdots\cdot(3\text{-}1)$$

・滑空実験一回目の結果

図 3.4 は目標の高度でドローンから誘導回収型観測機を切り離し、滑空している様子である。図 3.5 は滑空実験一回目で誘導回収型観測機が記録した滑空経路のデータを Google Earth 上にプロットしたものであり、誘導回収型観測機のパラフォイルが滑空中にうまく風をつかんだことで緩やかに滑空している。

図 3.4　ドローンから誘導回収型観測機を切り離した様子

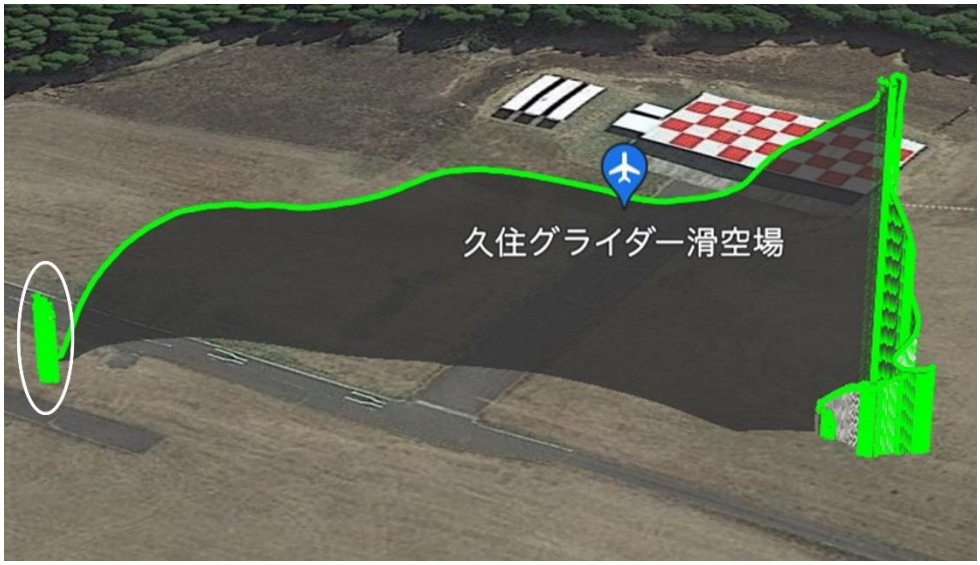

図 3.5　滑空実験一回目の誘導回収型観測機の滑空経路を Google Earth 上にプロットした様子（白丸で囲っている軌跡は着陸の衝撃を観測したと考えられ、実際は飛行していない）

図 3.6 は滑空実験一回目の誘導回収型観測機が着陸するまでのデータである。赤丸部分に V 型の波形が確認できる。

図 3.7 は図 3.6 の気圧の変動部分(24700 ~ 26900 付近)を拡大したものであり、赤丸①では N 型の波形が確認でる。赤丸②では INF03S の値がドローンのホバリング中に比べ、滑空中の振幅の方が大きくなっていることが分かる。赤丸③では誘導回収型観測機の着陸と同時に INF03S の値が急に大きく下がっていることが分かる。

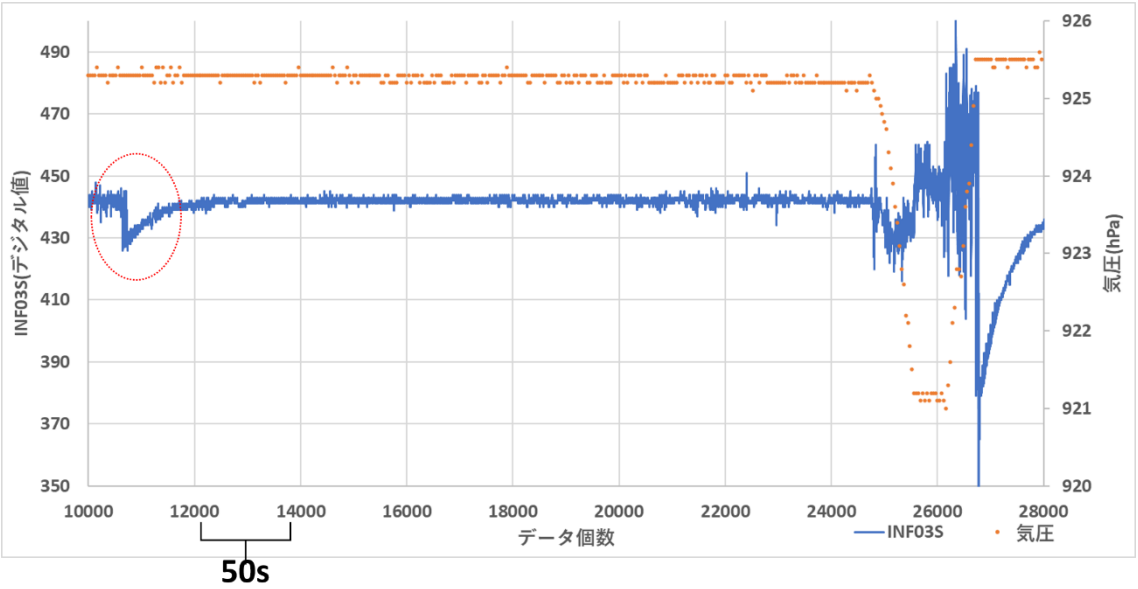

図 3.6　実験一回目の INF03S と気圧の値（誘導回収型観測機着陸まで）

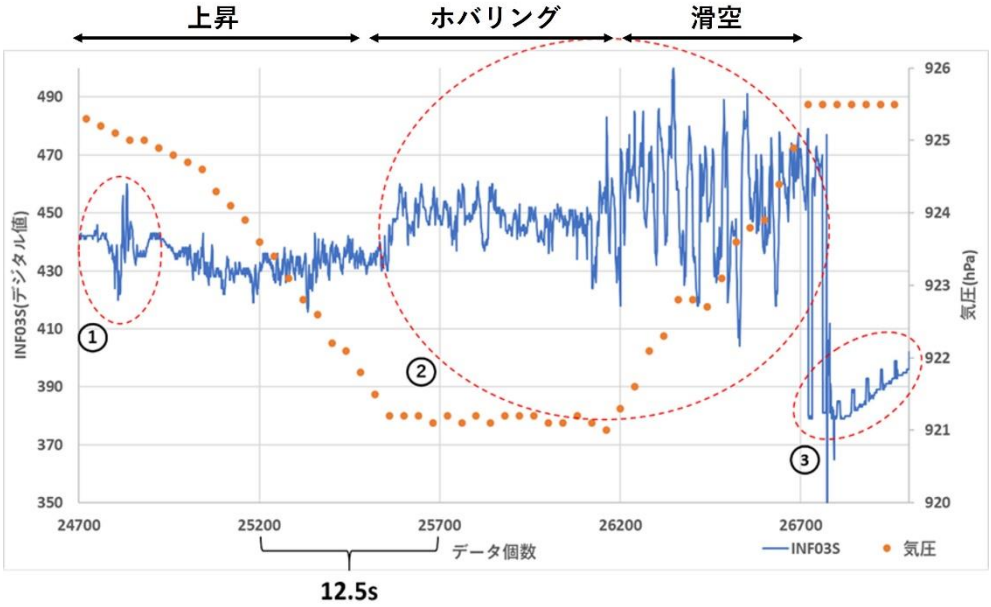

図 3.7　図 3.6 の気圧の変動部分を拡大したデータ(上昇、ホバリング、滑空の波形の特徴がそれぞれで分かれている)

図 3.8 は BOX で計測した滑空中の INF03S (第一軸)と気圧(第二軸) の値、図 3.9 は滑空中の誘導回収型観測機で計測したの移動速度(第一軸)と気圧(第二軸)の値であり、二つの図の第二軸の気圧は使用している気圧センサの違いから 2 hPa 程度の値のずれがあるが、値の傾きの傾向は一緒である。

図 3.8 と図 3.9 の気圧が上昇し始めた辺り、図 3.8 の赤の四角の範囲で INF03S の振幅が小さくなり、図 3.7 のデータ個数 160 辺りで移動速度が 8 m/s まで下がっている(図 3.9、赤の四角)。

図 3.8 のデータ個数 210 個を過ぎた滑空後半辺りから移動速度が小さくなり、図 3.9 の INF03S の波形も滑空後半辺りから振幅が徐々に小さくなっている。

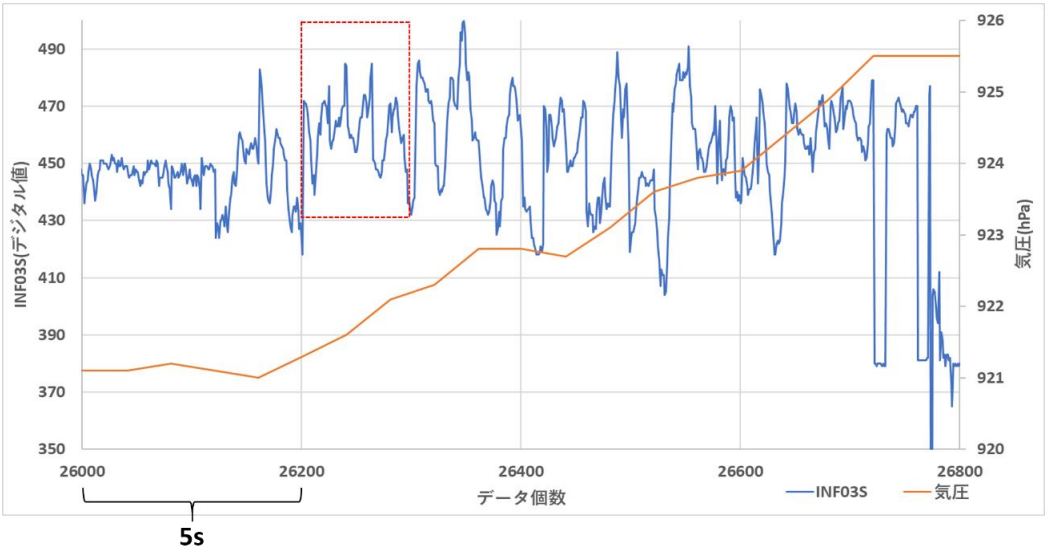

図 3.8　実験一回目の滑空中の INF03S と気圧のデータ

図 3.10 は図 3.6 の続きであり赤丸部分を見ると前後より振幅の大きい波形がある。

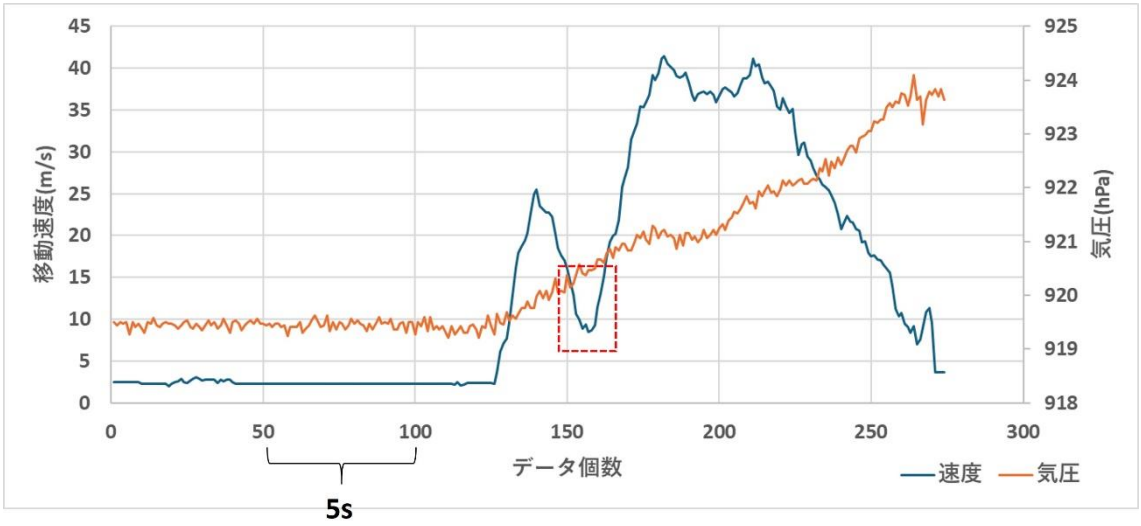

図 3.9 実験一回目の誘導回収型観測機で取得した滑空中の移動速度と高度

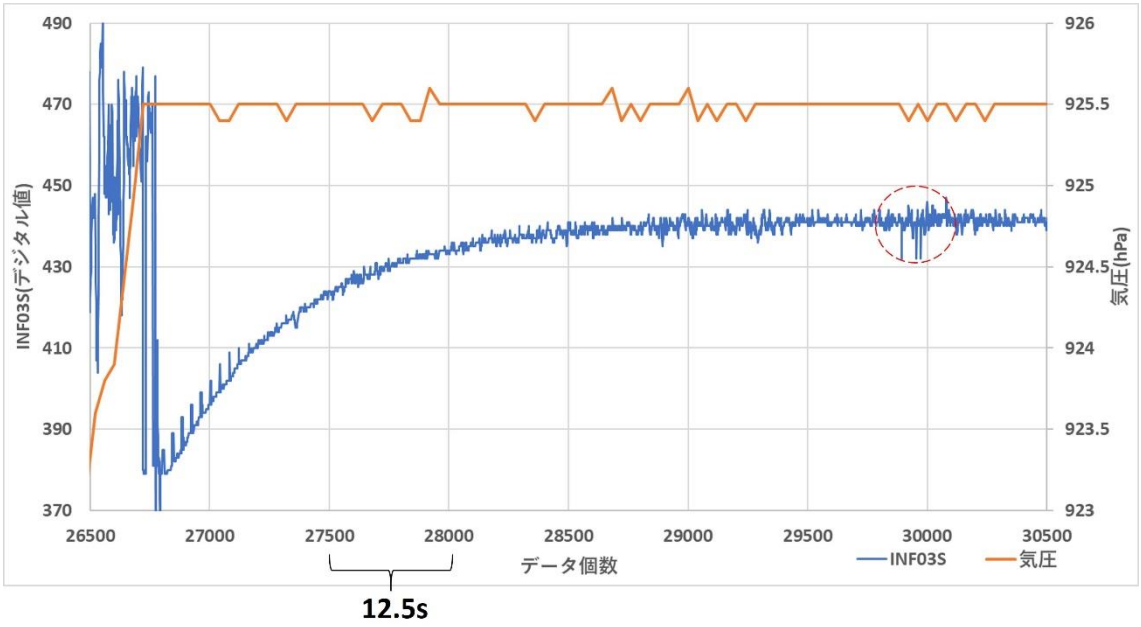

図 3.10　実験一回目の INF03S と気圧の値（誘導回収型観測機着陸後）

・滑空実験二回目の結果

図 3.11 は滑空実験二回目で誘導回収型観測機が記録した滑空経路のデータを Google Earth 上にプロットしたものであり、滑空実験一回目に比べて滑空経路の勾配が急である。

図 3.12 に滑空実験二回目の誘導回収型観測機が着陸するまでのデータを示す。

図 3.13 は図 3.12 の赤丸の拡大図であり、RTC モジュールの時間のずれを考慮し図 2.11 で示したタイミング①の、手を叩いたと予想できる時間帯である。図 3.13 のデータ個数

57800 個あたりから INF03S の振幅が大きくなっていることが分かる(図 3.13 赤の四角)。

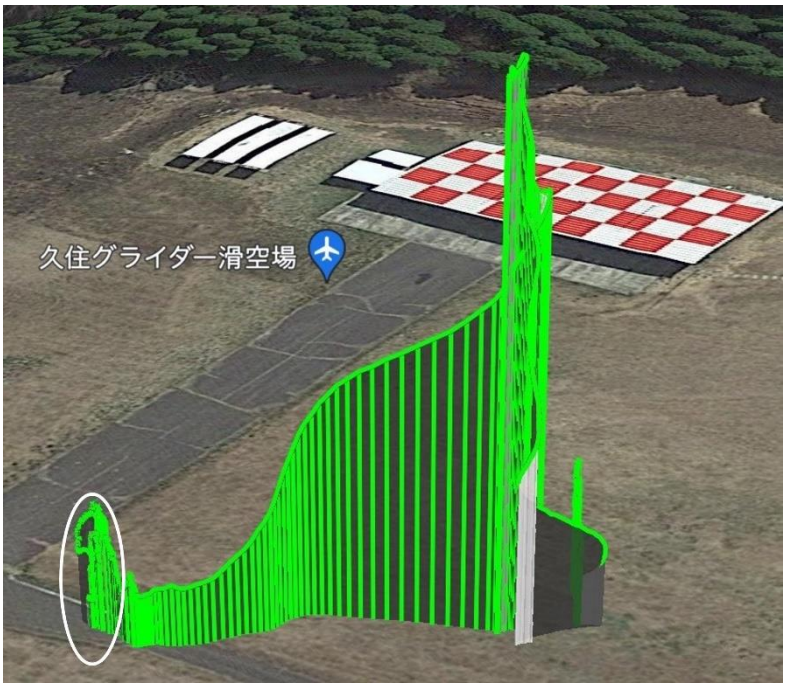

図 3.11　滑空実験二回目の誘導回収型観測機の滑空経路を Google Earth 上にプロットした様子（白丸で囲っている軌跡は着陸の衝撃を観測したと考えられ、実際は飛行していない）

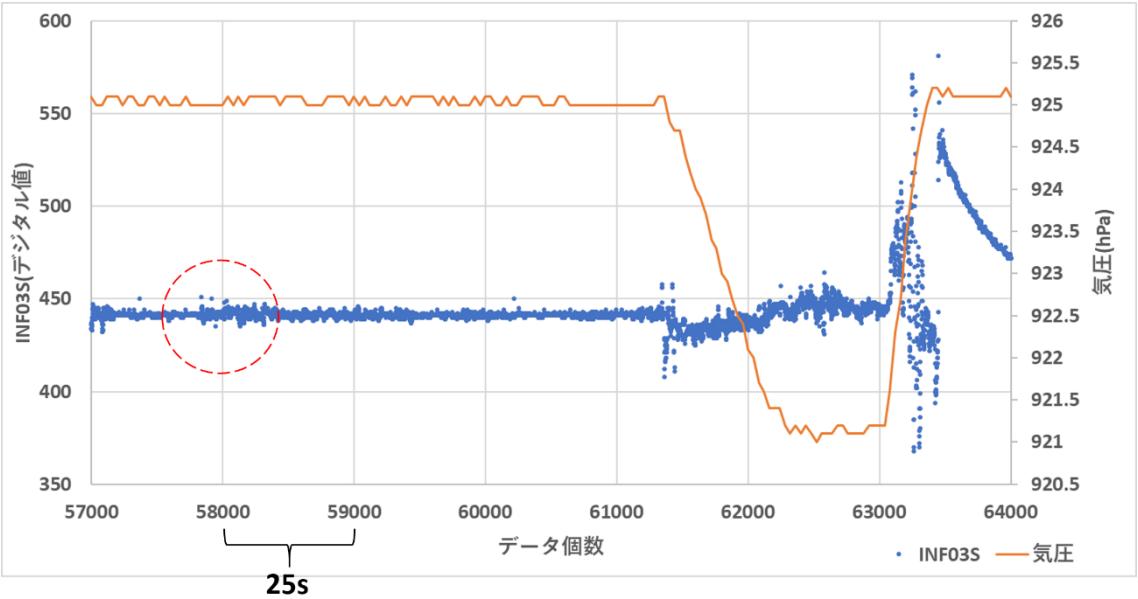

図 3.12　実験二回目の INF03S と気圧の値（誘導回収型観測機着陸まで）

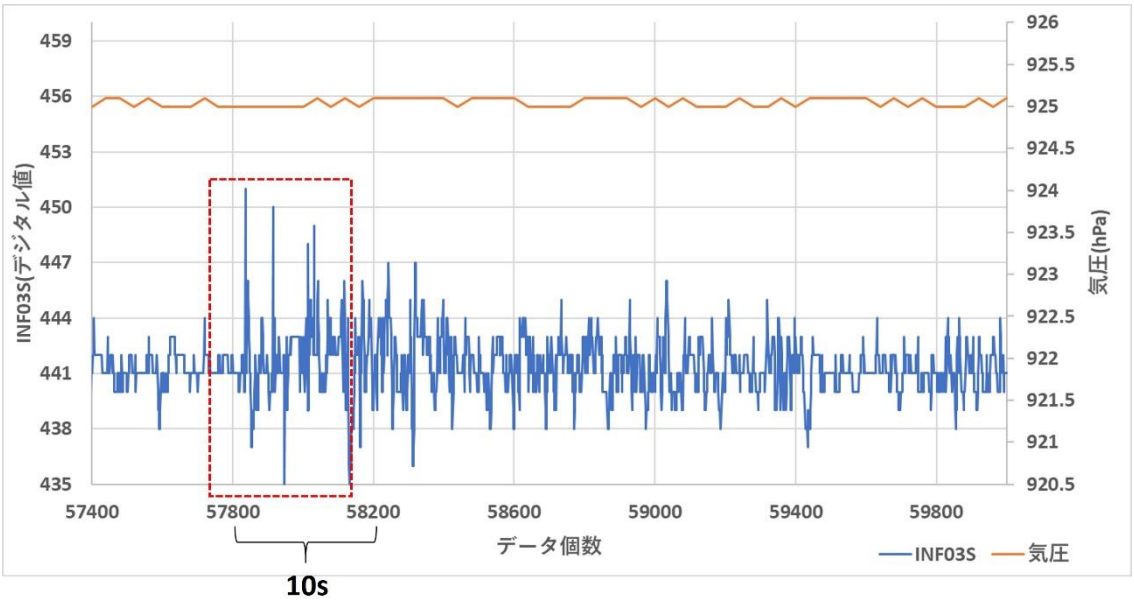

図 3.13　図 3.9 の赤丸の拡大図(手を二回たたいた時間帯)

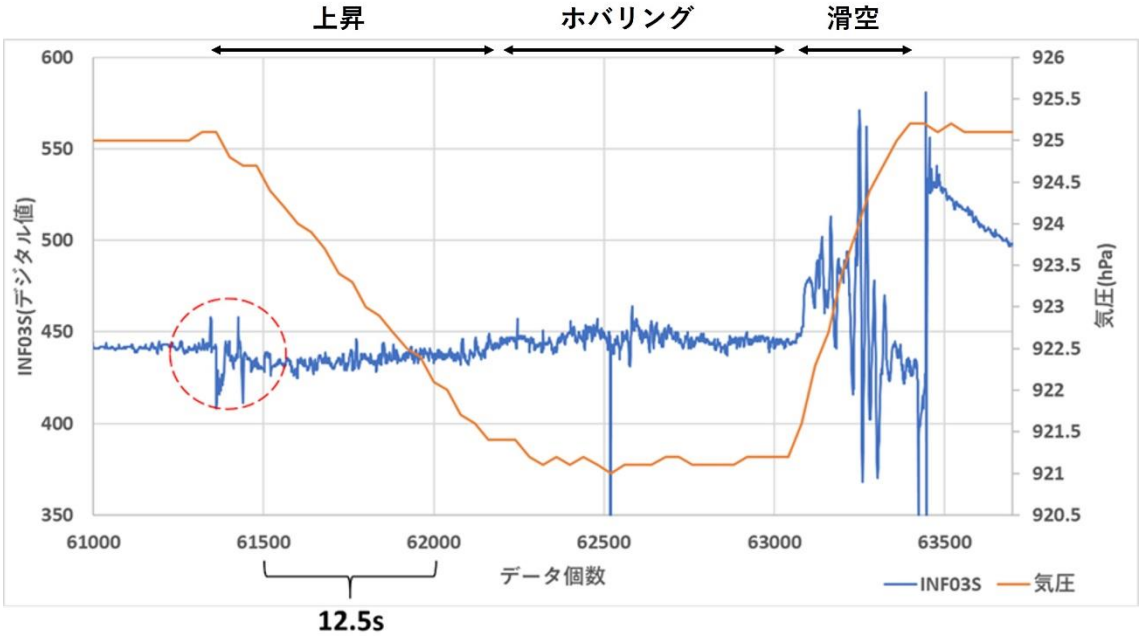

図 3.14　図 3.12 の気圧の変動部分を拡大したデータ

図 3.14 は図 3.12 の気圧の変動部分を拡大したデータであり、赤丸部分に波形がみられる。ドローンのホバリング中は INF03S の振幅が小さく、滑空し始めたら振幅が大きくなっている。着陸衝撃時には値が大きく跳ね上がっている(図 3.13 赤矢印)。

図 3.15 は図 3.12 の続きのデータであり誘導回収型観測機着陸後のデータである。赤丸の部分に INF03S の値が急に下がっている部分がある。図 3.16 に実験二回目の滑空中の INF03S と気圧のデータを示す。

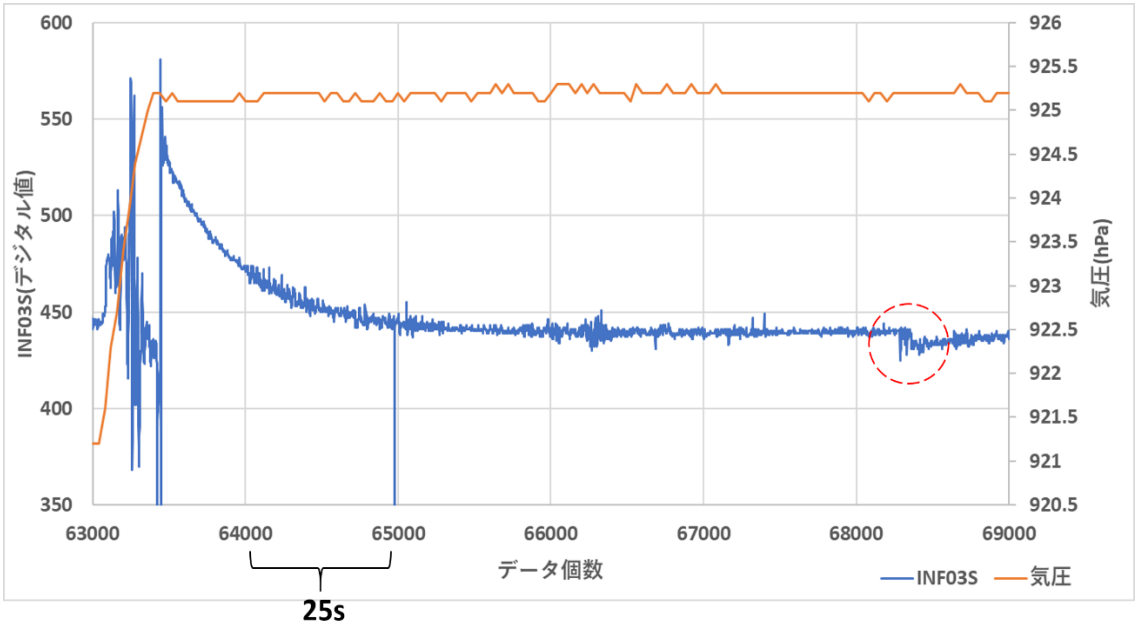

図 3.15　実験二回目の INF03S と気圧の値（誘導回収型観測機着陸後）

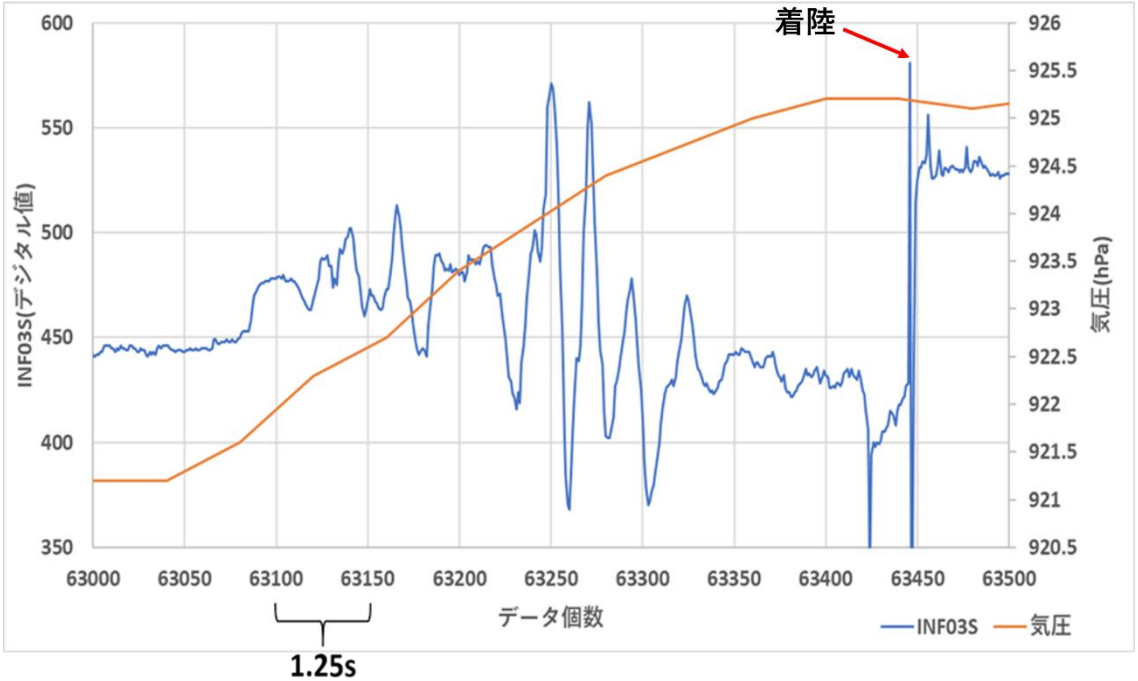

図 3.16　実験二回目の滑空中の INF03S と気圧のデータ

・**滑空実験三回目の結果**

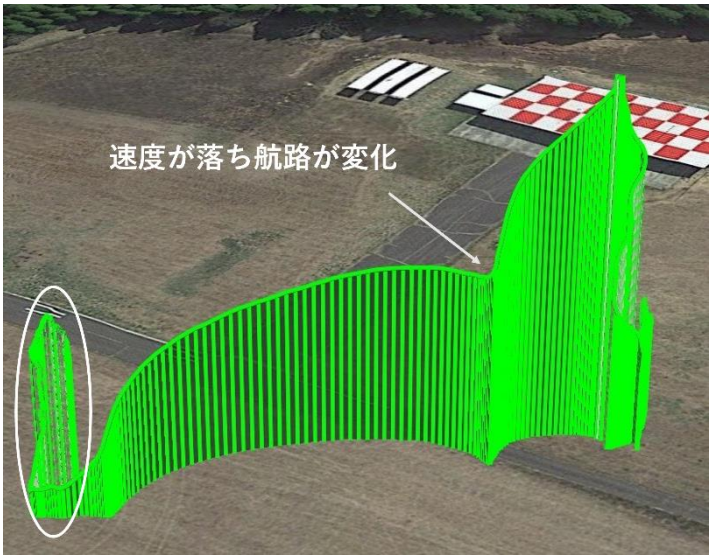

図 3.17　滑空実験三回目の誘導回収型観測機の滑空経路を Google Earth 上にプロットした様子（白丸で囲っている軌跡は着陸の衝撃を観測したと考えられ、実際は飛行していない）

図 3.17 は滑空実験三回目で誘導回収型観測機が記録した滑空経路のデータを Google Earth 上にプロットしたものであり、矢印で示した位置で誘導回収型観測機の速度が落ち航路が変化した。

図 3.18 に滑空実験三回目の誘導回収型観測機が着陸するまでのデータを示す。

図 3.19 は図 3.18 の赤丸で囲まれた範囲の拡大図である。二回目の実験結果と同じように時間のずれを考慮しており、図 2.11 のタイミング①で手を叩いたと考えられる時間帯である。赤丸の部分に振幅の大きい波形が見られる。

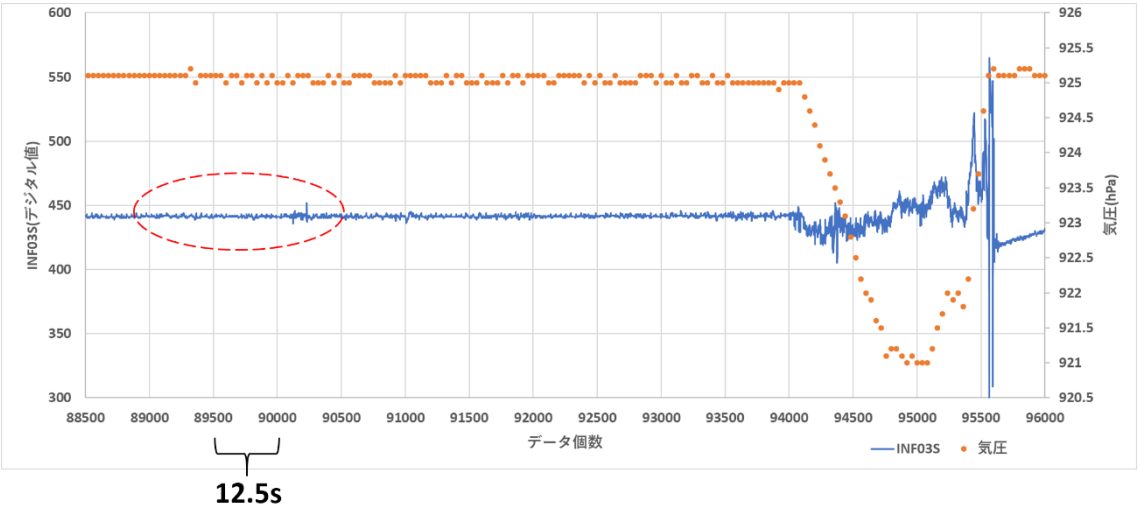

図 3.18　実験三回目の INF03S と気圧の値（誘導回収型観測機着陸まで）

図 3.20 は図 3.19 の気圧の変動部分を拡大したデータであり、赤丸に前後より振幅の大きい波形が確認できる。赤の四角内の時間帯は前後より振幅が小さく、気圧はほぼ一定になっている。

図 3.17 は図 3.14 の続きのデータであり誘導回収型観測機着陸後のデータである。赤丸部分に小さな波形が見られる。

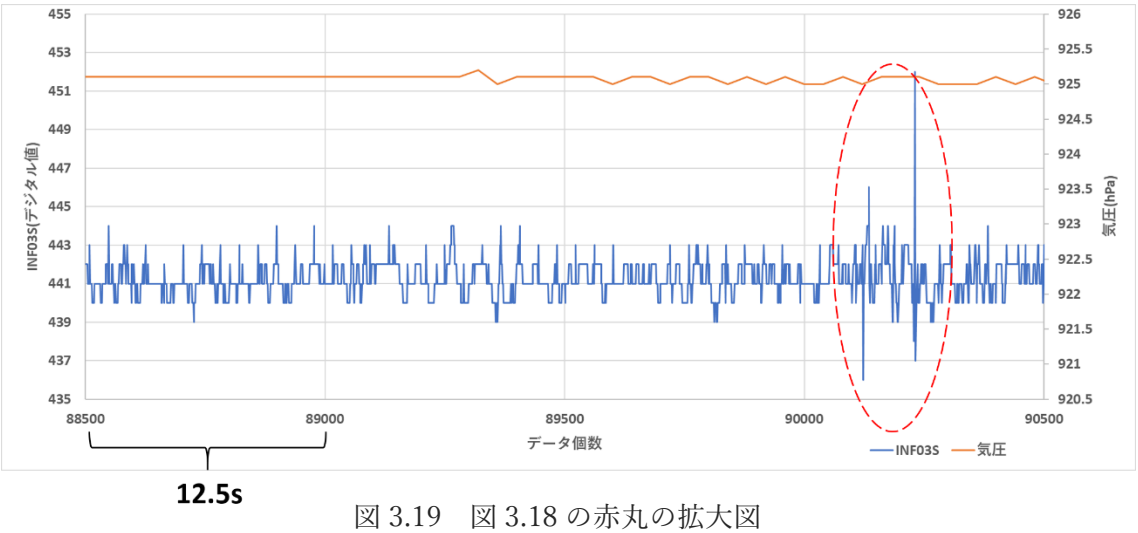

図 3.19　図 3.18 の赤丸の拡大図

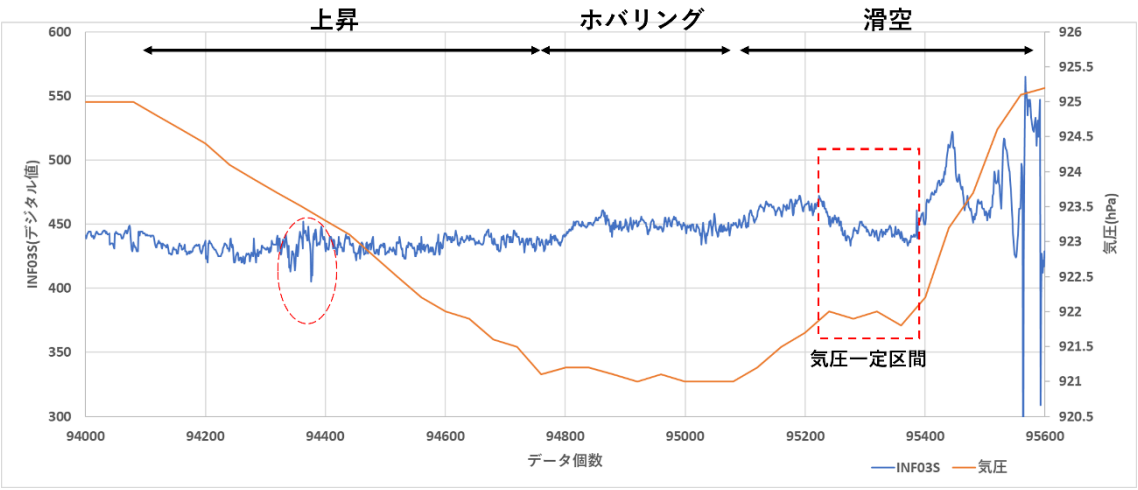

図 3.20　図 3.18 の気圧の変動部分を拡大したデータ

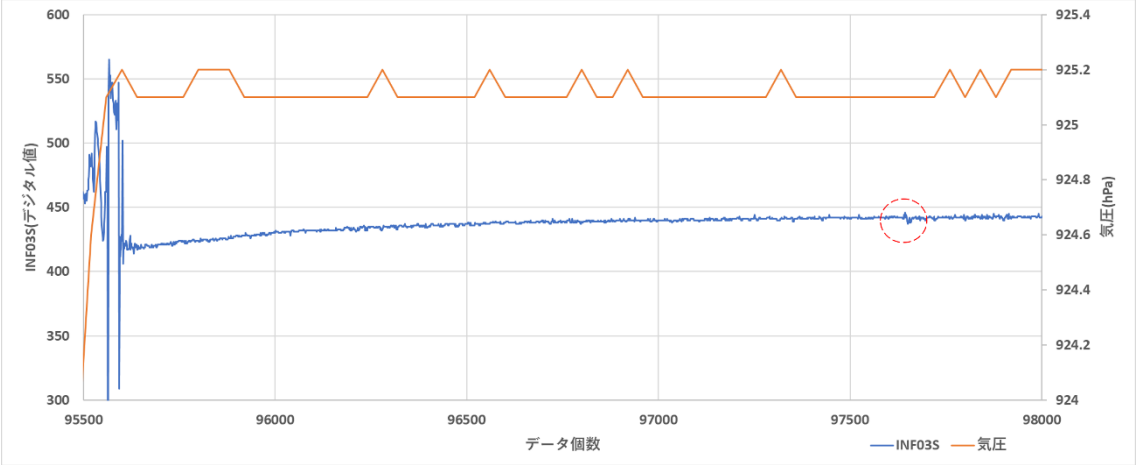

図 3.21　実験三回目の INF03S と気圧の値（誘導回収型観測機着陸後）

3.2 環境実験の結果

3.2.1 恒温槽を使った防水軽量型データロガーBOX の動作実験の結果

図 3.22、図 3.23 に恒温槽を使った環境実験の結果を示す。槽内温度が目標温度の-55℃まで到達できず-38.1℃が最低到達温度となったため-38.1℃の条件下で実験を行った。

図 3.22 は INF03S の値と恒温槽内温度の関係であり、INF03S の振幅が大きいデータ個数 400000 個以下の範囲が恒温槽を動作させ槽内を冷やしている区間である。データ個数 300000 個を超えた辺りから-38℃以下になっている。INF03S の値は恒温槽動作中の赤丸部分のように 100 を下回る値が出ており、恒温槽動作停止後はおよそ 100 分にわたって 100 を下回る INF03S の値が出ている。恒温槽内の温度が上がってきたデータ個数 700000 個を超えてからは INF03S の値が極端な外れ方をすることはなくなった。

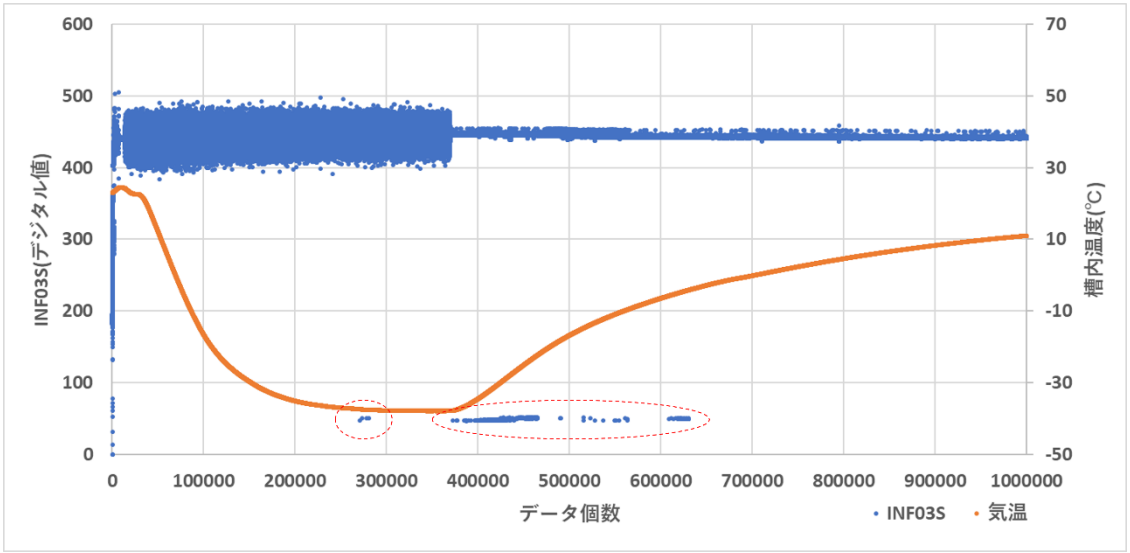

図 3.22　恒温槽実験結果　INF03S と槽内温度の関係

図 3.23 は INF03S と湿度の関係であり、湿度が恒温槽稼働停止後 33.5 %まで急に上がり緩やかに 27 %付近まで低下している。恒温槽稼働停止後、湿度が高くなると INF03S の 100 を下回る値が増えていることが分かる。また、バッテリーが何時間もつか試した実験では 66 時間稼働可能であった。

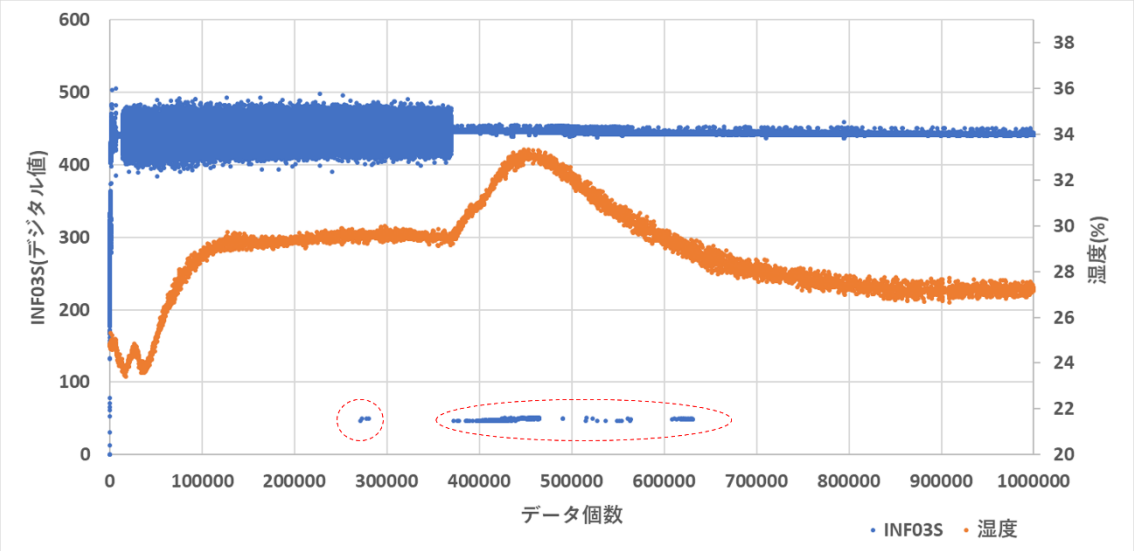

図 3.23　恒温槽実験結果　INF03S と槽内温度の関係

3.2.2 防水軽量型データロガーBOX の防水性能の実験の結果

図 2.14 の BOX を 20 秒間水に浸け、蓋を開けると蓋と箱の接合部付近と蓋のフィルム付近のトイレットペーパーの一部が濡れており BOX 内に水が少し浸入しているのが分かった

第四章　考察と評価

4.1 ドローンを用いた誘導回収型観測機の滑空実験における防水軽量型データロガーBOX　の動作確認実験の考察と評価

・図 2.11 のドローンの準備中(タイミング①)に発生させた音源について

図 3.6、図 3.12、図 3.18 の赤丸部分で確認した波形は図 2.11 のタイミング①、ドローンの準備中に発生させた音源を観測したと考えられる。

タイミング①で発生させた音源の時刻を記録してあり、滑空実験一回目から順に[13 時 10 分]、[14 時 5 分]、[14 時 18 分]である。滑空実験のタイミング①の音源についてそれぞれ考察する。その後のタイミング②、③、④、⑤の発生時刻は記録していない。

・滑空実験一回目(図 3.6)

滑空実験一回目の音源はおもちゃの火薬銃で発生させ、手を三回たたくよりも音が大きいため、図 3.6 のように値が急に変わり緩やかに戻っていく波形(コンデンサマイクが飽和状態になった反応)が出力されていると考えられる。

図 3.6 の波形の立ち上がり(データ個数 29841 付近)の時刻は[13 時 10 分 5 秒]であり、音源を発生させた時刻と 5 秒差で RTC モジュールとの時刻のずれの範囲内と考えられるため、図 3.6 の音波はタイミング①で発生させた音源と考えられる。

・滑空実験二回目(図 3.12)

図 3.12 で確認した波形はデータ個数 57837 付近であり時間は[14 時 5 分 5 秒]と、音源発生時間から 5 秒のずれてあり RTC モジュールの時間のずれの範囲内と考えられるため、図 3.12 の音波はタイミング①で発生させた音源だと考えられる。

・滑空実験三回目(図 3.18)

滑空実験三回目の図 3.18 で確認した波形はデータ個数 90231 付近であり時間は[14 時 18 分 34 秒]と、音源発生時間から 34 秒ずれている。上述の滑空実験一回目と二回目のタイミング①の音源とデータの関係から RTC モジュールの時刻のずれは 5 秒か 6 秒と考えられるため、図 3.18 の波形はタイミング①で発生した音源ではないと考えられる。

・図 2.11 のドローン飛ばしはじめ(タイミング②)に発生させた音源について

図 3.7 赤丸①、図 3.14 赤丸、図 3.20 赤丸の波形はタイミング②のドローンの飛ばしはじめに車のトランク(ハッチバック車の後方ドア)を閉めることで発生させた音源の波形だと考えられる。3 つの図に示された波形はすべてドローンが上昇し始めている(気圧の下がり始めている)辺りで発生しており、三回の実験それぞれで確認できているため、人為的に起こされたものだろうと考えたからである。

・**図 2.11 のドローンのホバリング中(タイミング③)に発生させた音源について**

図 3.7、図 3.14、図 3.20 のホバリング中取得した INF03S のデータを確認したが、特に目立った波形がなく、車のトランクで発生させた音源によってできた波形か、周りの雑音によってできた波形かの区別が出来なかった。ドローンの飛ばしはじめの音源が確認できているのを考慮すると、ホバリング中は上空の誘導回収型観測機と車の距離が離れているため伝わるまでに音波が弱まりドローン動作音を含む雑音に埋もれてしまったのではないかと考える。

滑空実験三回のうち二回目のデータのみタイミング③の音源を観測したと考えられる波形を確認できるので図 4.1 に示す。

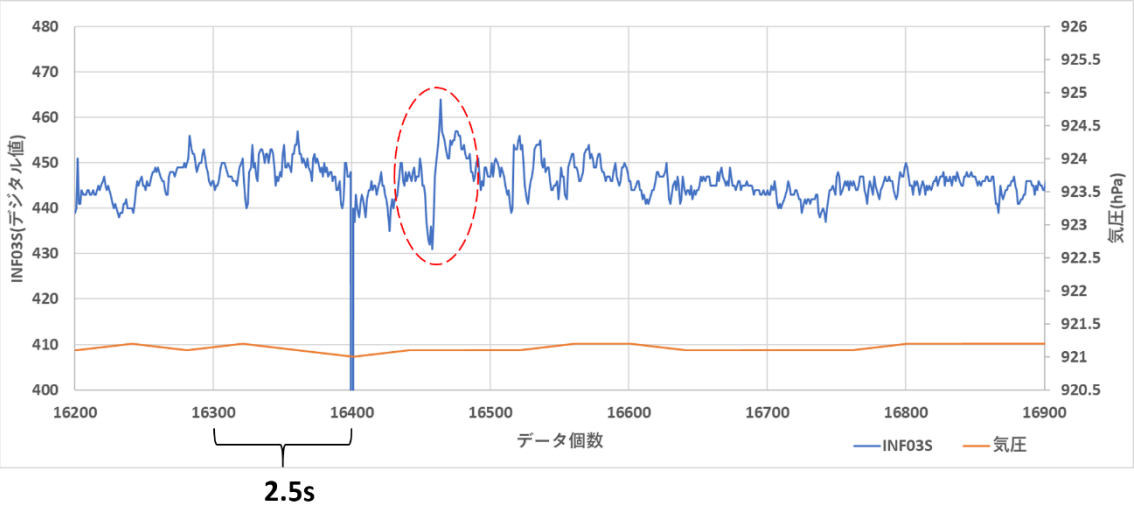

図 4.1　滑空実験二回目のホバー中の INF03S のデータ　赤丸に手順③の音源を観測したと考えられる波形を確認できる

・**図 2.11 の滑空中(タイミング④)に発生させた音源とノイズについて**

図 3.8、図 3.14、図 3.20 は三回の滑空実験の滑空中の INF03S と気圧のデータである。データを見ると滑空中は INF03S の値が乱れていることが分かる。

図 3.8 と図 3.9 の INF03S の値と移動速度の関係を考えると、誘導回収型観測機の移動速度が速い区間はデータの乱れが大きいことや、図 3.20 の気圧の変化が激しい(高度の変動が激しい)区間のデータの乱れが大きいことから、風による振動や音、誘導回収型観測機が揺れることで INF03S のコンデンサマイクが反応し、ノイズが発生していると考えられる。そのためタイミング④で発生させた音源は、ノイズにかき消されて確認することが出来なかったと考えられる。

・**図 2.11 の回収時(タイミング⑤)に発生させた音源について**

図 3.10、図 3.15、図 3.21 を確認すると誘導回収型観測機着陸後には手を三回たたいたと考えられる波形がある。しかしタイミング⑤で発生させた音源と思われる波形が同じ実

験のデータで何個かあり、タイミング⑤の音源を発生させた時刻が分からないため、どの波形が手を三回たたいたと考えられるかが正確にはわからない。

・Arduino nano のアナログ入力分解能について

INF03S の値は 0~5 V の範囲内で出力され、出力された値を Arduino nano 内の AD 変換機能を使い AD 変換した。Arduino nano のアナログ入力分解能は 10 bit であり 0~5 V を 0~1023 の範囲内に置き換えた値で SD カードに保存される。

インフラサウンドのデータ解析は細かい波形の変動まで読み取る必要があるため図 4.2 のようにデータを拡大する必要がある。しかし Arduino の AD 変換の分解能ではデータの解像度が粗く、本来滑らかな波形になるべきデータが図 4.2 に示すように、横ばいのデータになる。より精度の高い解析をする場合は、Arduino nano 搭載の AD 変換機能ではなく外付けの AD 変換を採用するか、Arduino nano 本体を変える必要があると考える。

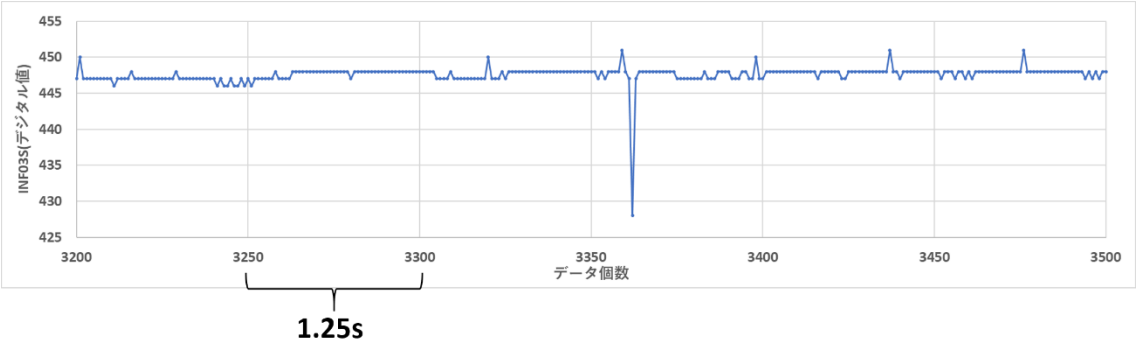

図 4.2　Arduino nano 搭載の AD 変換の分解能についてのデー

4.2 滑空中の風の影響の確認実験

・目的

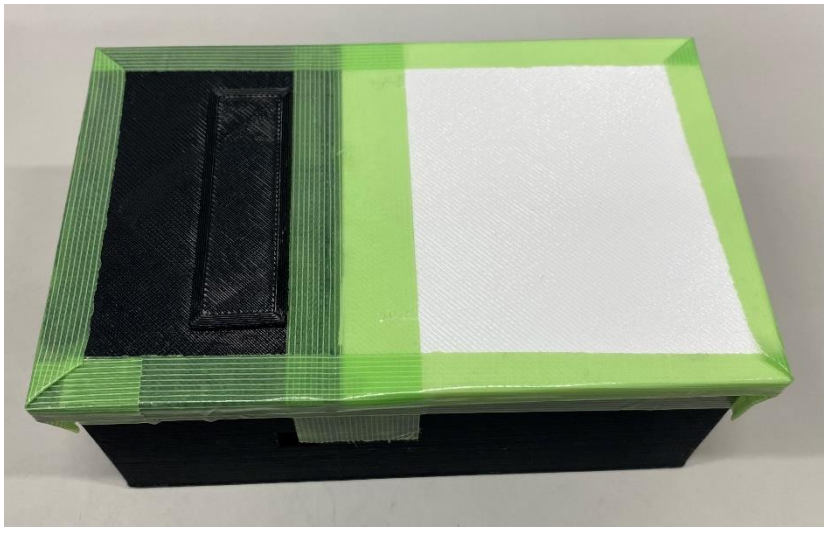

図 4.3　フィルムがない蓋を使用した防水軽量型データロガーBOX

4.1 節で記述したように INF03S のデータを解析する際、風の音が INF03S のデータに重なり、必要とするデータを見つけられないことがある。図 2.1 の BOX の蓋のビニールフィルムが風の音を拾いやすいと考えフィルムの影響を受けない図 4.3 のフィルムがない蓋を 3D プリンター(WEEDO 制 TINA2)で作成し,図 2.1 のフィルムがある蓋と図 4.3 のフィルムがない蓋のどちらが風の影響を受けにくいかを確認することを目的とする。

・実験方法

実験方法は、滑空中の風の影響を再現するために扇風機を首振り動作にして BOX に周期的に強弱のある風を当てる。その風の中で研究室のドアを 1 分ごとに閉めインフラサウンドを発生させる。フィルム有りと無しの BOX それぞれ同じ条件下で実験を行い、取得したデータを見比べることによりフィルムの影響を確認する。

・実験結果

フィルム有りの結果(図 4.4)とフィルム無しの結果(図 4.5)を見比べると両者ともドアの開閉を検知出来ているが、僅かにフィルム無しの方が風による波形の振幅が小さいのが分かる。

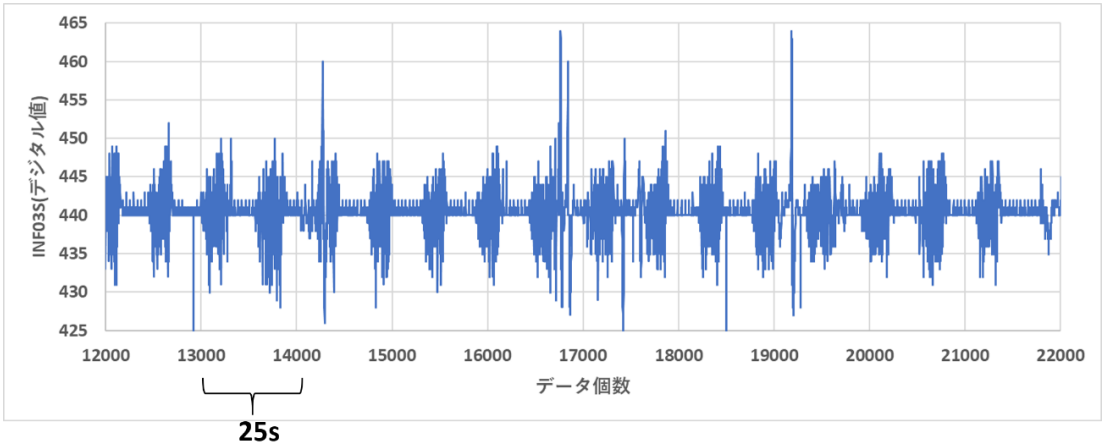

図 4.3　フィルム有りの蓋を使用して風の影響下で観測したデータ

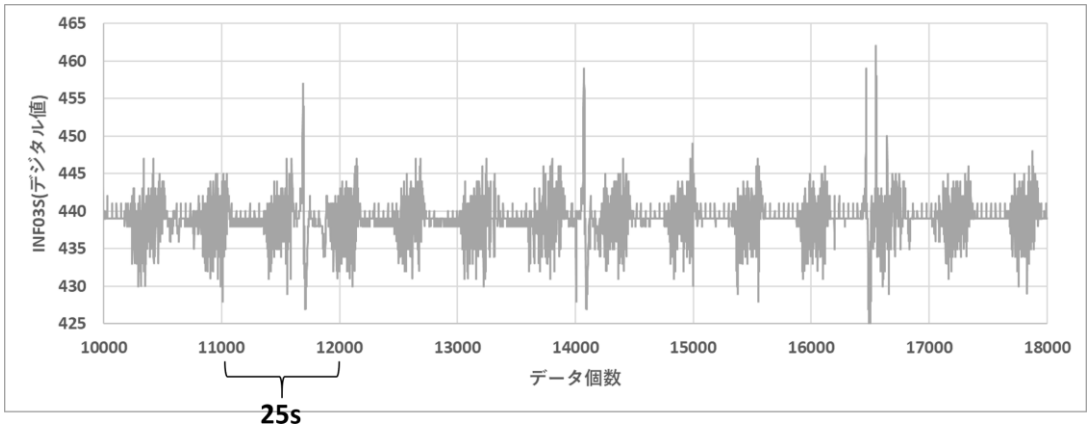

図 4.4　フィルム無しの蓋を使用して風の影響下で取得したデータ

・考察

実験結果よりフィルム有りと無しでは僅かにフィルムがない蓋の方が風の影響を受けにくいことが分かった。よってフィルムだけでなくBOXのボディに風が当たっている音も原因になっているため、BOXの外装に風の音の影響を受けにくいような加工をする必要があると考える。

4.3 環境実験の結果の考察と評価

4.3.1 恒温槽を使った防水軽量型データロガーBOXの動作実験の考察と評価

3.2.1項の実験結果より、極端に値の離れたデータがいくつかあるものの-40℃までなら解析ができる程度のデータの取得が出来ることが分かった。

極端な値の離れ方のデータについては図3.22、図3.23よりBOX内の温度の低下による結露の影響と考えられ、結露対策としてBOX内の温度を低下させないために基板を防災用アルミシートで包装する、BOX内の湿度を下げるために乾燥剤を入れるなどの対策をする必要があると考える。その他にも結露した水滴がセンサの銅線につくと短絡し、データが取得できないためグルーガンなどで銅線や配線をカバーし水滴がつかないようにするのも効果的だと考える。

また、3.2.1項により、現在のバッテリーと消費電力では66時間稼働可能なので、これは目標の3時間を超えておりバッテリーはCR123Aをそのまま採用していいと考える。

4.3.2 防水軽量型データロガーBOXの防水性能の実験の考察と評価

図2.17のBOXに水が入った原因を探るために4.2節で使用した蓋(図4.2)を使用し同じように水の中に沈めた。結果は図2.14のBOXと同じように水が入った。次にテープの補強が弱いのではないかと考え、図4.5のように養生テープでさらに補強するとBOX内に水は入ってこなかったのでテープの接地面不足で水が入ってきたと考えられる。

図2.14のフィルムありの蓋ではフィルム面にテープを貼ることはできないためテープの接地面不足で水が入ってくる可能性があるので完全防水を必要とする場合フィルム無しの蓋を使用した方が良いと考える。

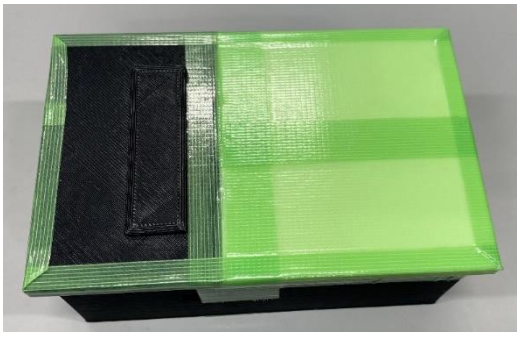

図4.5　テープの補強をした防水軽量型データロガーBOX

第五章　結論

・1.2 節で述べた目的の達成度

まず 1.2 節で記述した防水軽量型データロガーBOX に必要な 5 つの機能について記述する。1 つ目の質量 350 g 以内は完成した BOX が 320 g であり条件を満たしている。2 つ目の BOX の大きさが縦 150 mm × 横 107 mm × 高さ 64 mm 以内であるが、縦 135 mm × 横 86 mm × 高さ 56 mm なので条件を満たしている。3 つ目の長時間のフライト(3 時間程度)が運用可能かについても、に関しては 66 時間動作可能であったため条件を満たしている。4 つ目の海に着水させても BOX 内のシステムが水没しない程度の防水性能はフィルムがない場合は水の中に 20 秒間沈ませても大丈夫だったが、フィルムありの蓋を使う場合は蓋の表面の凹凸を無くすなどしてテープとの密着性を高める必要がある。5 つ目の BOX 内の各センサが必要なデータを取得できる程度の空気や音圧の提供である、達成はできているがまだ風の影響を考えると改善の余地があると言える。プラス α の目標としての通信モジュールを使った地上へのデータのリアルタイム転送であるが、これについては期間内に実装出来なかったので達成できていない。

以上より防水軽量型データロガーBOX として動作可能な程度の目標は達成できているといえる。

・Arduino nano のプログラムについて

プログラムについては不完全な点が 4 つある。一つ目は GPS との同期が出来ていないため RTC モジュールに時間のずれがあることである。二つ目は SD カードに書き込まれる最初の 2, 3 回は時刻が 0 になる列が出来ることである(図 3.1)。三つ目は誤った時刻の書き込みが入ることである(図 3.2)。四つ目はミリ秒の表示が出来なかったことである。INF03S のデータは 25 ms 間隔で行われるのでミリ秒の表示が出来るとより正確なデータ解析が出来るためプログラムに実装したい。

以上 4 つが不完全な点である。第四章で示したような実験結果の考察についてのデータ解析の正確性と信憑性を高めるためにも、より精度の高い時刻の取得が必要である。

・防水軽量型データロガーBOX のシステムについて

データの地上へのデータのリアルタイム転送の実装もあるが、BOX が切り離されてしまった場合に備えて BOX が紛失してしまわないよう、GPS で位置情報が分かる機能を加えるとより便利になると考える。

・滑空実験について

滑空実験については、データの解析時に探している波形らしきものを確認できるが、信憑性をもって探しているデータと特定出来なかったため、すべての音源の発生の時刻のメ

モが必要だった。音源発生の方法についても、トランクのドアより強い音波が出るものを採用するべきだった。また同じ実験をすることがあれば競技陸上用の火薬銃スターターなどで音源を発生させ、より遠くまで音波が届くようにする必要があると考える。

・防水軽量型データロガーBOX の外装について

データ解析する際に風の音が邪魔だったので BOX の形を風の影響を受けにくい形にする、外装に屋外用の可聴音マイクにつけるスポンジ状のウインドジャマーやウインドスクリーンを付けるなどして防風対策を加えるべきである。

・環境実験について

恒温槽が-55℃まで下がらず高度 15 km 付近の環境実験がまだできていないので、-55℃まで下げられる恒温槽で環境実験を行えればよい。

防水性能については完全防水とは言えないものの水しぶき程度では BOX 内に水は入ってこないので必要最低限の防水性は確保できていると考える。

謝辞

本研究を行うにあたり、ご指導いただいた指導教員の高知工科大学 システム工学群 山本真行教授、西川泰弘助教に心から感謝申し上げます。また、滑空実験の実施や防水軽量型マルチデータロガーBOX の開発を手伝ってくださった STRVSN 河野紘基 氏、成蹊大学 枝本雅史 助教、NO_2 センサの BOX への搭載、滑空実験に協力して下さった福岡大学 高島久洋 教授、乙部直人 助教、そして研究へのアドバイスをして下さった山本真行研究室の先輩、後輩、同期に心から感謝申し上げます。ありがとうございました。

参考文献

[1] 山本真行，枝本雅史，令和 5(2023)年度　基盤研究(C)　(一般)研究計画調書, 2023/

[2] Yasuhiro Nishikawa ,Masa-yuki Yamamoto ,Kensuke Nakajima ,Islam Hamama ,Hiroaki Saito ,Yoshihiro Kakinami ,Masumi Yamada & Tung-Cheng Ho , Observation and simulation of atmospheric gravity waves exciting subsequent tsunami along the subsequent tsunami along the coastline of Japan after Tonga explosion event, *Scientific Reports*, 5-6, 2022.

[3] Islam Hamama and Masa-yuki Yamamoto, Infrasonic Earthquake Detectability Investigated in Southern Part of Japan, *Sensors* 2019, 11 - 16, 2021.

・滑空実験で使用した滑空回収型観測機を制作している STRVSN のホームページ

[4] STRVSN，無人気球による、新しい空の観測。
https://strvsn.net/ (2024 年 2 月 21 日参照)

・福岡大学　高島久洋 教授、乙部直人 助教のホームページ

[5] 福岡大学，福岡から診る大気環境研究所
https://www.se.fukuoka-u.ac.jp/fit/ (2024 年 2 月 21 日参照)

・先行研究

[6] 枝本雅史，パラフォイルを用いた小型飛翔体自律誘導システムの基礎開発，高知工科大学平成 27 年度卒業研究報告書, 2015.

[7] 河野紘基，国内運用に最適化された小型高高度気球システムの開発と飛翔評価試験 Development and evaluative flight experiments of small-high altitude balloon system optimized for domestic operation，高知工科大学平成 28 年度特別研究報告書, 2016.

[8] 平塚丘将，小型観測気球用の着陸位置選択式輸送システムの開発と評価 Development and evaluation of selective landing area carrier system for small sounding balloons，高知工科大学平成 31 年度修士論文, 2019.

付録

・Arduino nano のプログラム内容（P14 のデータ誤り改善前）

```
#include<stdio.h>
//I2C ライブラリ
#include <Wire.h>
// タイマー割り込み　ライブラリ
#include<FlexiTimer2.h>
//シリアル通信　ライブラリ
#include<SPI.h>
//SD カード　ライブラリ
#include<SD.h>
//BME280 ライブラリ
#include "SparkFunBME280.h"
//rtc アドレス
#define RTC_address 0x68

//配列番号　指定
const int i = 40;

//BME280 関連
BME280 sensor;

// inf03s_data 関連
unsigned short int flag_A = 0, sound;

// rtc_data 関連
unsigned short int time[7];
unsigned short int rtc_i, rtc_hour,
rtc_minute, rtc_second;

// inf03s_rtc_data 関連
unsigned char  hour[i][2], minute[i][2],
second[i][2];
unsigned short int  inf[i][2];
unsigned short int  data_i = 0, data_j = 0;

// SD カード関連
const  int chipSelect = 10;

// save_SDcard 関連
unsigned short int SD_i = 0, SD_j = 0;

//そのほか
unsigned short int flag_B = 0;

void setup() {

  Serial.begin(115200);

   //I2C 通信開始
  Wire.begin();

  delay(1000);

   //BME280 I2C 開始
  sensor.beginI2C();

  delay(1000);

  //SD カード　準備
  if(!SD.begin(chipSelect)){
    Serial.println("error of setting SD");
    return;
  }else{
    Serial.println("success of setting SD");
  }

  delay(1000);

  // 25ms 毎に inf03s のデータ取得
  FlexiTimer2::set( 25 , inf03s_data);
  FlexiTimer2::start();

}

void loop() {

  //inf03s のデータ取得を確認したら実行
  if(flag_A != 0){

    //時刻取得
    rtc_data();

    //inf03s と rtc の配列を作成
    inf03s_rtc_data();

    flag_A = 0;
    flag_B += 1;
  }

  //SD カードに書き込み
  if(flag_B == i - 1){

  save_SDcard();

  flag_B = 0;

  }

}


//タイマー割り込みで inf03s データ取得
void inf03s_data(){

  unsigned short int inf03_data =
analogRead(0);

  sound = inf03_data;

  flag_A = 1;

}


//rtc データ取得
void rtc_data(){
```

```
  //rtc 通信開始
  Wire.beginTransmission(RTC_address);
  Wire.write(0x00);//Register 先頭アドレス
  Wire.endTransmission();

  //RTC データの読み込み
  Wire.requestFrom(RTC_address,7);

  //データを time 配列に格納
  for(int rtc_i=0 ; rtc_i<=7 ; rtc_i++){
    time[rtc_i]=Wire.read();
    }

  //出力用に変換
  rtc_hour = change_dec(time[2]);
  rtc_minute = change_dec(time[1]);
  rtc_second = change_dec(time[0]);

}

//16 進数を 10 進数に変換
int change_dec(byte dec){
  int data;
  data = (dec >> 4) * 10 + (dec & 0x0F);
  return data;
}


//データを配列に格納
void inf03s_rtc_data(){

  inf[data_i][data_j] = sound;
  hour[data_i][data_j] = rtc_hour;
  minute[data_i][data_j] = rtc_minute;
  second[data_i][data_j] = rtc_second;

  //次の配列番号を指定
  if( data_i != i - 1){
    data_i += 1;
  }else if( data_i == i - 1 && data_j == 0){
    data_i = 0;
    data_j = 1;
  }else{
    data_i = 0;
    data_j = 0;
  }

}


// SD カードに保存
void save_SDcard(){
  File logFile = SD.open("/inf03s.txt",
FILE_WRITE);

  if(logFile){

    for(SD_i = 0; SD_i < i - 1; SD_i++){

      logFile.print(hour[SD_i][SD_j]);
      logFile.print(",");
      logFile.print(minute[SD_i][SD_j]);
      logFile.print(",");
      logFile.print(second[SD_i][SD_j]);
      logFile.print(",");
      logFile.println(inf[SD_i][SD_j]);

    }

      logFile.print(hour[i-1][SD_j]);
      logFile.print(",");
      logFile.print(minute[i-1][SD_j]);
      logFile.print(",");
      logFile.print(second[i-1][SD_j]);
      logFile.print(",");
      logFile.print(inf[i-1][SD_j]);

      //BME280 データ
      logFile.print(",");
      logFile.print(sensor.readTempC(), 2);
      logFile.print(",");

logFile.print(sensor.readFloatHumidity(), 2);
      logFile.print(",");
      logFile.print(sensor.readFloatPressure()
/ 100.0, 1);
      logFile.print(",");

      //ガスセンサ　データ
      logFile.print(analogRead(6));
      logFile.print(",");
      logFile.println(analogRead(7));

    Serial.println("success of logging");

    if(SD_j != 0){
      SD_j = 0;
    }else{
      SD_j = 1;
    }

  }else{
    Serial.println("error of logging");
  }

  logFile.close();

}
```

・Arduino nano のプログラム内容（P14 のデータ誤り改善後）

```
#include<stdio.h>
#include <Wire.h>          //I2C ライブラリ
#include<FlexiTimer2.h>   // タイマー割り込みライブラリ
#include<SPI.h>           //シリアル通信　ライブラリ
#include<SD.h>            //SD カード　ライブラリ
#include "SparkFunBME280.h" //BME280 ライブラリ
#define RTC_address 0x68  //rtc アドレス

//配列番号　指定
const int i = 40;

//BME280 関連
BME280 sensor;

// inf03s_data 関連
unsigned short int flag_A = 0, sound;

// rtc_data 関連
unsigned short int time[7];
unsigned short int rtc_i, rtc_hour,
rtc_minute, rtc_second;

// inf03s_rtc_data 関連
unsigned char  hour[i], minute[i], second[i];
unsigned short int  inf[i];
unsigned short int  data_i = 0, data_j = 0;

// SD カード関連
const  int chipSelect = 10;

// save_SDcard 関連
unsigned short int SD_i = 0, SD_j = 0;

//そのほか
int flag_B = 0;

void setup() {

  Serial.begin(115200);

   //I2C 通信開始
  Wire.begin();

  delay(1000);

   //BME280 I2C 開始
  sensor.beginI2C();

  delay(1000);

  //SD カード　準備
  if(!SD.begin(chipSelect)){
    Serial.println("error of setting SD");
    return;
  }else{
    Serial.println("success of setting SD");
  }

  delay(1000);

  // 25ms 毎に inf03s のデータ取得
  FlexiTimer2::set( 25 , inf03s_data);
  FlexiTimer2::start();

}

void loop() {

  //inf03s のデータ取得を確認したら実行
  if(flag_A != 0){

    //時刻取得
    rtc_data();

    //inf03s と rtc の配列を作成
    inf03s_rtc_data();

    flag_A = 0;
    flag_B += 1;
  }

  //SD カードに書き込み
  if(flag_B == i ){

  save_SDcard();

  flag_B = 0;

  }

}


//タイマー割り込みで inf03s データ取得
void inf03s_data(){

  unsigned short int inf03_data =
analogRead(0);

  sound = inf03_data;

  flag_A = 1;

}


//rtc データ取得
void rtc_data(){

  //rtc 通信開始
  Wire.beginTransmission(RTC_address);
  Wire.write(0x00);//Register 先頭アドレス
  Wire.endTransmission();

  //RTC データの読み込み
  Wire.requestFrom(RTC_address,7);

  //データを time 配列に格納
```

```
  for(int rtc_i=0 ; rtc_i<=2 ; rtc_i++){
    time[rtc_i]=Wire.read();
    }

  //出力用に変換
  rtc_hour = change_dec(time[2]);
  rtc_minute = change_dec(time[1]);
  rtc_second = change_dec(time[0]);

}

//16 進数を 10 進数に変換
int change_dec(byte dec){
  int data;
  data = (dec >> 4) * 10 + (dec & 0x0F);
  return data;
}


//データを配列に格納
void inf03s_rtc_data(){

  inf[flag_B] = sound;
  hour[flag_B] = rtc_hour;
  minute[flag_B] = rtc_minute;
  second[flag_B] = rtc_second;

}


// SD カードに保存
void save_SDcard(){
  File logFile = SD.open("/inf03s.txt",
FILE_WRITE);

  if(logFile){

    for(SD_i = 0; SD_i < i - 1; SD_i++){

      logFile.print(hour[SD_i]);
      logFile.print(",");
      logFile.print(minute[SD_i]);
      logFile.print(",");
      logFile.print(second[SD_i]);
      logFile.print(",");
      logFile.println(inf[SD_i]);

    }

      logFile.print(hour[i-1]);
      logFile.print(",");
      logFile.print(minute[i-1]);
      logFile.print(",");
      logFile.print(second[i-1]);
      logFile.print(",");
      logFile.print(inf[i-1]);

      //BME280 データ
      logFile.print(",");
      logFile.print(sensor.readTempC(), 2);
      logFile.print(",");

logFile.print(sensor.readFloatHumidity(), 2);
      logFile.print(",");
      logFile.print(sensor.readFloatPressure()
/ 100.0, 1);
      logFile.print(",");

      //ガスセンサ　データ
      logFile.print(analogRead(6));
      logFile.print(",");
      logFile.println(analogRead(7));

    Serial.println("success of logging");

  }else{
    Serial.println("error of logging");
  }

  logFile.close();

}
```