公立はこだて未来大学 2021 年度 システム情報科学実習
グループ報告書

Future University-Hakodate 2021 System Information Science Practice

Group Report

プロジェクト名

めざせ宇宙開発 - 自律移動ロボット飛行プロジェクト

Project Name

Flying Autonomous Robot Project

グループ名

グループ A

Group Name

Group A

プロジェクト番号/Project No.

15-A

プロジェクトリーダ/Project Leader

成田凪　Nagi Narita

グループリーダ/Group Leader

佐藤大地　Daiti Sato

グループメンバ/Group Member

富樫幹生　Mikio Togashi
鈴木進太郎　Shintaro Suzuki
大岩穂峻　Hodaka Oiwa
西殿大輝　Daiki Nisidono
杉山宏輝　Hiroki Sugiyama

指導教員

大沢英一　三上貞芳　和田雅昭

Advisor

Ei-Ichi Osawa　Sadayoshi Mikami　Masaaki Wada

提出日

2022 年 1 月 19 日

Date of Submission

January 19, 2022

概要

CanSat とは缶サイズの模擬小型人工衛星のことを指し、本グループでは、CanSat の競技会であるスペースプローブコンテストへの出場を目的とし、競技会のレギュレーションに沿った CanSat の設計、運用を検討してきた。「カムバックコンペティション」と呼ばれる方式を取る本コンテストでは空中や地上での機体の制御や機体の耐久性が CanSat 開発に際する問題点として挙げられる。これらの問題に対する解決策を検討したうえでコンテストに出場し、その結果をもとに機体のアップグレードと屋外での実験を行った。コンテストへの参加と姿勢制御や機体の耐久性が確認できたものの空中でのパラフォイルの展開における安定性が課題として残った。

キーワード　CanSat, 超小型模擬人工衛星, カムバックコンペティション

（※文責: 佐藤大地）

Abstract

CanSat is a can-sized simulated small satellite, and this group has been studying the design and operation of CanSat according to the regulation of the competition for the purpose of participating in the Space Probe Contest, which is a CanSat competition. In this contest, which is called "comeback competition", attitude control in the air or on the ground and durability of the CanSat are the problems in the development. After studying the solutions to these problems, we participated in the contest, and based on the results, we upgraded the CanSat and conducted outdoor experiments. Although we were able to participate in the contest and confirm the attitude control and durability of the aircraft, the stability of the parafoil deployment in the air remained an issue.

Keyword CanSat, nano-simulated satellite, comeback competition

（※文責: 佐藤大地）

目次

第 1 章　はじめに

この章では、本プロジェクトの背景、従来の問題点、課題について述べる。

（※文責: 富樫幹生）

1.1　背景

現在、宇宙利用の形態は多岐に渡っており、科学的な分野である宇宙探査だけでなく人工衛星による地球観測や測位事業など産業的な面も大きくなっている。このように、我々が普段利用している GPS や気象予報など、ごく身近な物も宇宙開発の影響を受けている [1]。

しかし、宇宙開発そのものは身近なものにはなっていない。これは、宇宙開発が巨大化、複雑化、高コスト化、高信頼化し、簡単に試作や実験を行うことができないためである。そのため、現状では学生が本格的な宇宙工学を学ぼうと思っても困難であることが多い。そこで、学生の宇宙工学における教育を容易にすることを目的として始められたのが CanSat プロジェクトである。

（※文責: 富樫幹生）

1.2　CanSat について

CanSat とは空き缶サイズの模擬人工衛星を指し、実際の衛星と類似の開発プロセスで作製される。この CanSat を用いて学生の宇宙工学における教育を行おうと開始されたのが先述した CanSat プロジェクトである。CanSat プロジェクトの起源は、1998 年ハワイにて JUSTSAP 会議と連携して開催された University Space Systems Symposium (USSS) においてスタンフォード大学の RobertTwiggs 教授によって提唱されたことに由来する [2]。それ以降は、アメリカと日本を主に各国の学生が競技会に参加するなど盛んにプロジェクトが行われている。

CanSat プロジェクトにおける競技会は CanSat を上空から投下し、降下中または着地後にミッションを行う形式で行われる。競技会には、大きく分類してカムバックコンペティションとミッションコンペティションの 2 種目が存在する。カムバックコンペティションとは、地上制御と飛行制御のどちらか、またはその両方を用いて地上に設けられた目標地点を目指す種目である。ミッションコンペティションとは、自らミッションを設定しそのミッションの達成度とそのアイデア性を競う種目である。

本グループではカムバックコンペティションを採用した競技会に参加することを目的として CanSat の作製を行う。

（※文責: 富樫幹生）

1.3 従来の例

カムバックコンペティションに用いられる CanSat には、350ml クラスとオープンクラスの 2 つのサイズ規定がある。350ml クラスは重量 350g 以下、高さ 240mm 以下、直径 66mm 以下であり、オープンクラスは重量 1050g 以下、高さ 240mm 以下、直径 146mm 以下である。本グループではオープンクラスに準拠した規定を用いた競技会に参加するため、オープンクラスの規定に従って作製を行う。

ここで、過去にカムバックコンペティションを行った CanSat の例を ARLISS2012 東京大学の報告書 [3] から紹介する。ARLISS2012 では各チームが規定のサイズに収めた CanSat をアマチュアロケットに搭載し、上空約 4,000m へ打ち上げる。その後、アマチュアロケットから CanSat が放出され、CanSat は制御機構を用いて目標地点へ到達する。ARLISS2012 での東京大学の CanSat は地上制御を一切行わないフライバックを採用していた。GPS の取得データを軸にパラフォイルの紐をサーボモータで引っ張ることで進行方向を決定する機構であった。

さらに、本プロジェクトで昨年度作製した機体を紹介する。

昨年度の A グループでは、350ml クラスサイズの CanSat を設計・運用した。ミッションは、CanSat の着地後の走行による軌道によって図形を描くというものである。ミッション達成のために、描いた軌跡を画像認識で線として出力することを試みた。実験の流れとしては、まずダムから CanSat を落下させ、パラシュートを展開させる。CanSat が地面に着地後地面を走行しつつ、その軌道で図形を描いた後、GPS センサによって与えられた目標地点を目指して走行するというものであった。

昨年度の B グループでは、オープンクラスサイズの CanSat を設計・運用を検討した。ミッションは、壊れやすいものを壊れないように運ぶというものである。ミッション達成のために、運搬物を入れる箱を作製した。運搬物としてチョークを採用し、これを衝撃吸収材とともに収納できるように設計した。実験の流れとしては、まずダムから機体を落下させ、グライダーによる空中移動を行う。緩衝材で衝撃を抑えて軟着陸し、コンテナから CanSat が出てきたら、センサ類の情報から自律走行をはじめ、あらかじめ決めておいた目標地点を目指して走行するというものであった。

（※文責: 鈴木進太郎）

1.4 従来の問題点

従来、カムバックコンペティションを行う CanSat における問題点の一部として以下があげられる。

姿勢制御

カムバックコンペティションでは、降下中の制御にパラシュートを用いる機体が多い。このような機体ではパラシュートを展開する際に機体に紐が絡まってしまい、制御が難航という問題点がある。また、風などの影響を受けて機体の姿勢が変わってしまい、制御に支障が出るという事例も報告されている [3]。

滞空性

機体の重量や減速機構などの設計ミスにより、パラシュートやパラフォイルによって落下中の速度を減速しきれずに十分な滞空を行えない場合がある。このような事態が発生すると滞空時間が短くなっていまい制御可能な時間も短くなってしまう。

センサの誤作動

降下時または地上走行時において、センサの誤作動によって想定外の動作をしてしまう場合がある。このように想定外の動作をしてしまうと、目標地点への到達が困難になる。

機体の耐久性

パラシュートや制御翼などの減速機構で減速を行っても、着地時の衝撃で機体及び内部モジュールが破損してしまうことがある。この場合ミッション中のログが取得できなくなる。

サイズ規定

CanSat 競技会は CanSat の重量、高さ、直径などレギュレーションが規定されている。よって、搭載モジュールや電気回路、機体、パラシュートの作製に必要最低限な材料に加えて、軽材料を選別し、設計する必要がある。

このように、カムバックコンペティション参加における CanSat の作製には多くの問題点があり、段階実験を通して検証、改善していく必要がある。

（※文責: 鈴木進太郎）

第 2 章　活動内容

この章では、プロジェクトの目的について述べる。

（※文責: 西殿大輝）

2.1　プロジェクトの目的

本プロジェクトの目的は、CanSat の設計・構築・運用を通して宇宙工学に関連した回路設計作製技術、飛行制御技術、無線通信技術、プロジェクト運用法を学習することである。

（※文責: 大岩穂峻）

2.1.1　プロジェクト学習で行うことの利点

本学の科目「システム情報科学実習」のシラバスを参照し、到達目標から以下の 3 点において本プロジェクトの利点を示す。

プロジェクト遂行に必要となる技術を学習する

本プロジェクトでは CanSat の作製を通してハードウェア・ソフトウェア両面での技術を獲得できる。機体の外装や電子回路、各種センサやマイコン、無線による通信など様々な技術的要素が詰め込まれており、総合的な能力を身に着けることができる。

プロジェクトを自主的に管理・運営する方法を学習する

複数のメンバーと協力して 1 つのプロダクトを作り上げるため、タスクの管理や全体の進行管理が必須である。また、本プロジェクトでは他と比べて必要部品が多く、かつ破損しやすため、予算の都合上機体の構想やスケジュールについて詳細に決めておく必要があり、早い段階から管理・運営についての手法を学習することができる。

成果を内外に公表し，大学および地域社会に貢献する

本グループではスペースプローブコンテストに参加することで、外部への発表の機会を得ることができ、大学の知名度の向上や CanSat 界隈、ひいては宇宙開発分野の盛り上げにつながる。

（※文責: 大岩穂峻）

2.2　グループの目的

本グループでは、CanSat 競技会に出場することを目的とした。今回スペースプローブコンテストに出場した。

（※文責: 西殿大輝）

2.2.1　スペースプローブコンテストについて

スペースプローブコンテストとは、植松電機株式会社が主催している CanSat 競技会であり、2021 年 9 月 18 日にリモートで開催される [4]。参加者はチームを組んで開発を行い、スペースプローブコンテストで制定されるミッションに対して取り組むこととなる。本グループはチーム「ナオ・キソ―マ」として参加した。今年はフライバックコンペティションに取り組む。上空から CanSat を投下し目標地点を目指し空中制御を行い、着地点と目標地点の距離の近さや芸術性で競う競技である。

（※文責: 西殿大輝）

スペースプローブコンテストの流れ

スペースプローブコンテスト当日、作製した CanSat についてスライドを用いて発表を行う。その後、ドローンを用いて高度 100m からの投下審査を行う。最後に、投下審査の際の動画やセンサのデータ解析などをもとに結果について発表を行う。

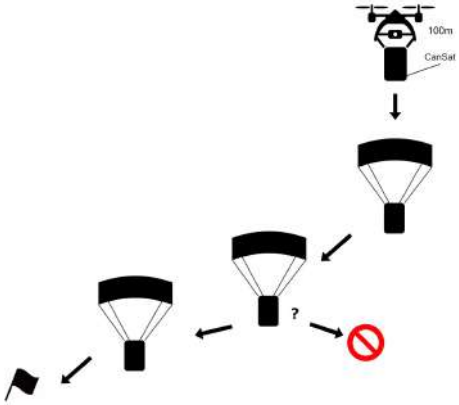

図 2.1　コンテストの流れ

（※文責: 西殿大輝）

2.2.2　サクセスクライテリア

目標を達成するための到達レベルとしてサクセスクライテリアを設定した。レベルの低い順にミニマムサクセス、ミドルサクセス、フルサクセス、アドバンスサクセスとなっている。

ミニマムサクセス (60%)
前提目標を遂行・達成するうえで必ずクリアしなくてはならない基礎的な目標 最低限 CanSat の制御がなされていることが確認できる。

ミドルサクセス (80%)
前提目標を達成できる状態での最低目標 CanSat を制御して目標地点の半径 10 m以内に着地することができる。

フルサクセス (100%)
前提目標を達成できる状態での本来の目標 CanSat を制御して目標地点の半径 5m 以内に着地できる。

アドバンスサクセス (120%)
前提目標が達成可能である状態でのより高度な達成目標 スペースプローブコンテストの評価基準において着地精度以外でも高評価を獲得できる。

（※文責: 西殿大輝）

2.3　課題の設定

ここでは、前項で挙げられた目標を達成するために解決すべき課題を記述する。

1. 動作のタイミング CanSat が投下され、パラシュートが開いてから制御を開始させるため、パラシュートが開いたかどうかの判定をする必要がある。

2. 風による影響風による機体の急激な軌道変化などの外的要因も加味したアルゴリズムの作成が必要である。

3. 着地時の衝撃当機体はマイコンやその他電子モジュールによって制御され、SD カードなどの繊細な部品も搭載するため、着地時の衝撃を最小限に押さえ、部品が破損しないようにする必要がある。

4. マイクロコンピュータによる制御 GPS や加速度センサによる値から進行方向を決め、マイクロコンピュータでサーボモータを制御し目標地点まで飛行させる。値の取得方法やサーボの制御方法などを間違うと目標地点へたどり着けないため、各モジュールの使用方法や実装方法を理解する必要がある。

5. 動作ログ実験やスペースプローブコンテストの結果を定量的に分析するため、各モジュールの数値的な履歴を残しておくことが必要である。その方法の 1 つとして SD カードへの書き込みがあげられるが、設定や書き込み方法が難解である。

（※文責: 西殿大輝）

2.4 課題の割り当て

各人の得意分野および要望をもとに以下の 2 班に分かれ、前項であげられた課題を役割ごとに振り分けた。

（※文責: 佐藤大地）

2.4.1 機体班

機体班では CanSat 競技における機体構造の検討、作製を行う。

本グループでは減速機構としてパラフォイルを採用し、サーボモータにより紐を引くことで移動することとした。

スペースプローブコンテストでは飛行制御のみで目標地点まで近づくため、空中での機動性を高める必要がある。課題としては、パラフォイルや機体の材料や設計はどうするのか、紐を引く長さによりどれほどの移動量が確保できるのかあらかじめ実験を行うなどの検討があげられる。

また、スペースプローブコンテストの規定により 6m/s 以上の降下速度を維持する必要があるため、着地時の衝撃により基盤やマイコンが破壊されないようにする必要がある。

なお、重量や形状なども競技会によってレギュレーションが異なり、後述する制御班との連携を取りつつ作製を進める必要がある。

（※文責: 佐藤大地）

2.4.2 制御班

制御班では CanSat 競技における目標地点へ向かう機体の制御やその実装を担当する。主にアルゴリズム・プログラム・電装という 3 種類の役割をもち、相互に連携しながら開発を進める。

アルゴリズム

GPS や各種センサの値から目標地点へ向かう制御をおこなうためのアルゴリズムを検討する。風などの外的要因による機体への影響も考慮したアルゴリズムの構築が課題である。

電装

使用するモジュールの選定や基盤の設計、作製を行う。他のメンバーと連携を取り、重量などを考慮して動作に必要なモジュールを決めた。

プログラム

各種モジュールの制御やアルゴリズムをマイコンに組み込むためのプログラムを作成する。使用する部品の仕様を理解し、意図した動作をさせることが課題である。

（※文責: 杉山宏輝）

2.5 スケジュール

2.5.1 スペースプローブコンテスト前

5 月

グループ内の役職を決め、マネジメントのためのツールの導入や、スペースプローブコンテストに向けての機体・制御の構想、機材の選定と発注を行った。

6 月上旬

スペースプローブコンテストの申し込みを行った。マイコンの書き込み回路の作製や PIC データシートを読み込み、主要な機能や設定方法などについてまとめたほか、減速機構についての選定に関する実験を公立はこだて未来大学体育館で行った。

6 月中旬

PIC の動作確認を行い、サーボモータを PWM 制御で動作させたほか、簡易的な機体のモデルを作製し、はこだて未来大学体育館にて降下実験を行った。実験の結果から当初予定していた、減速機構を引っ張る紐の長さが足りないことが発覚したため、機体の構造を再検討した。並行して、報告書の内容の構成を決定した。

6 月下旬

減速機構をパラフォイルに決定。作製のための資料集めを行ったのち作製へ取り掛かった。また、中間発表に向けて web サイトや発表スライド、ポスターの制作を進めたほか、サクセスクライテリアを設け、本グループの目標を明確化した。

7 月上旬

中間発表に関する制作物を完成させ、発表会に臨んだ。

7 月中旬

主に書類の作成に当たった。中間報告書の執筆項目を各メンバに割り当て、Notion 上で作成した。グループ内での締め切りを設け、仮完成後全メンバで確認・修正を行い、TeX に書き起こした。並行して、学習ポートフォリオや学習フィードバックを作成した。その他、実験申請書やスペースプローブコンテストに関わるメディア承諾書・機体三面図・設計仕様書などの書類を作成し、提出した。

また、笹流れダムでの実験日を決定し、申請書を作成するなど実験に向けて準備を進めたほか、機体の構造についての検討やアルゴリズムの検討を行った。また、スペースプローブコンテストで

は目標地点の GPS 座標が知らされない可能性が発覚したため、ビーコンや無線機など、目標地点を特定するための方法を検討した。

7 月下旬

期末提出物をすべて完成させ、提出した。

また、夏季休暇中の活動について話し合い、タスクを洗い出した。その結果、機体班では衝撃吸収機構・機体の内部構造の作製とパラフォイルの格納と展開方法の検討、制御班ではサーボモータ制御基板の作製、各モジュールの動作確認、アルゴリズムの検討を行うこととなった。

さらに、笹流れダムでの投下実験を行い、その様子を動画に記録した。システムの実装が間に合わなかったためパラフォイルの動作確認のみとなった。

8 月上旬

工房利用ができなかったこともあり、主に実験結果の考察を行った。紐の長さや重さなどの条件を変え、計 7 度の投下を行ったが、ほぼパラフォイルが展開せずに落下していたため、パラフォイルの素材と構造について見直した。

8 月中旬

制御班では PIC による SD カードや 9 軸センサへのアクセスに関して知識不足を感じたため勉強に費やした。一応の動作確認はできたが、扱いにくさを感じたためマイコンの変更を検討した。

機体班では新たにセルを取り入れたパラフォイルを作製し、大学の体育館で投下実験を行った。実験結果から改善案を出し、本番用に側面を湾曲させたパラフォイルを作製したほか、機体に使用する素材を MDF に決定し、作製に取り掛かった。

8 月下旬

集中講義やインターンシップがあったためグループで集まって作業を行う時間が確保できなかった。

9 月上旬

9 月上旬も集中講義やインターンシップによりグループで集まって作業を行うことが出来なかった。制御班は PIC での実装が間に合わないと判断したため、使用マイコンを RaspberryPI に変更した。また、本格的にシステムを実装するにあたり、チームでのコード管理を容易にするため GitHub を導入した。機体班はパラフォイルと機体の作製を進めたほか、スペースプローブコンテストに向け、最終版の機体三面図や設計仕様書、投下手順書を作成した。

9 月中旬

スペースプローブコンテストに用いるプレゼン資料とプレゼン動画を制作した。また、機体を完成させ、発送した。18 日にコンテストが行われ、スライドを用いた発表と質疑応答を行った。当日は雨天のため投下が延期となり、事後プレゼンも後日となった。

（※文責: 杉山宏輝）

2.5.2　スペースプローブコンテスト後

9 月下旬

コンテストの投下が延期となり、機体が手元にないため全てオンラインでの活動となった。シミュレート用の仮想モジュールの作製に着手し、実機が無くてもシステムの動作確認ができるような環境作りを開始した。

10 月上旬

延期されていた投下が 2 日に行われた。当日は主催がホストの ZOOM にて落下の映像を共有し、各チームで質疑応答が行われた。後日、落下位置と投下の映像が共有されたため、それをもとに事後プレゼン動画を制作し、提出した。

10 月中旬

コンテストの結果から、パラフォイルの展開機構と落下検知に不備が見つかったため修正作業を行ったほか、後期の笹流ダムで行う実験の申請を行った。

10 月下旬

システムの実装期間が短かったことから、コードが整理されていなかったためリファクタリングを行った。また、使用していない余計なファイルを削除し、煩雑化していた階層構造を整理した。

11 月上旬

笹流ダムでの投下実験を行った。数度の投下により座標や加速度値などの各データが記録されていること、サーボモータの作動により軌道が変わったことから、正常に制御が働いていることを確認した。また、ドローンを使った投下実験について検討と航空法の確認を行った。

11 月中旬

引き続きリファクタリングを行った。主に行動決定アルゴリズムに使用していた関数の機能を細分化し、可読性を上げることに成功した。また、メインファイルに集約していたコードを分割し、機能ごとにモジュール化した。ドローンを使った実験は降雪までに許可が下りるか不明なので中止に決定した。

プロジェクト全体の活動として 12 月の成果発表会に向けて発表の班分けを行った。また、Web サイトやポスターの担当者を決め、期末提出物の作成を進めた。

11 月下旬

コンテストや実験の成果をもとにポスターや Web サイト、発表スライドの制作を行った。著作権まわりの修正をいくつか行いつつ完成させ、提出した。

12 月上旬

成果発表会を行った。

12 月中旬

fun キャリがあったため活動日はほとんど無かった。

12 月下旬

最終報告書の執筆にあたった。

1 月

最終報告書を完成させ提出する。

（※文責: 杉山宏輝）

第 3 章　課題解決のプロセス

この章では、2.4 で割り振られた課題に対して行った作業について述べる。

（※文責: 大岩穂峻）

3.1　各班の担当課題

3.1.1　機体班

機体班では主に落下の際に使用する減速機構と各種モジュールを収める機体を作製した。減速機構は飛行を主な目的とするパラフォイル、パラフォイルの展開補助を目的とするパラシュートの二種類を採用した。

パラフォイル

本グループは機体の制御を行うにあたってパラフォイルを採用した。

1. 設計

機体を制御することができる減速機構には以下のような条件が要求される。

- 減速機構の紐を引くことで旋回制御が行える。
- 姿勢を安定状態に保って飛行できる。
- 機体やモジュールが破損しないよう落下速度を抑えられる。

円形、六角形のパラシュートと、パラフォイルの試作型を大学体育館で条件ごとに比較したところ、パラフォイルの機体制御の有効性が勝ったため、パラフォイルを使用するに至った。

これらを踏まえ、以下のようなパラフォイルを作製した。

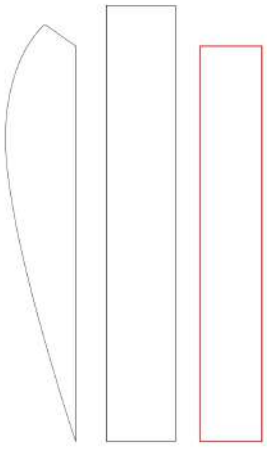

図 3.1　パーツ設計図

図 3.2　作製したパラフォイル

まず基本的な構造を説明する。図 3.1 がパラフォイルのパーツ設計図であり図 3.2 が作製したパラフォイルである。パラフォイルは試作型をいくつか作製、実験したのち過去の CanSat 競技会で使用されたパラフォイル [5] や既存のカイト [6] を参考にして新たに設計した。

素材にはリップストップポリエステル生地を使用し、耐久性の向上と軽量化を図った。作製方法としては生地をパーツ設計図に基づいて切り出し、縫い合わせることで作製した。翼に取り付けた制御紐はタコ糸を使用し、翼の部位ごとに先端をまとめて機体に接続した。その際に使用するタコ糸の基本の長さは 1000mm とした。

パラフォイルには空気を取り入れるセルを片側 7 つずつ設けた。これはパラフォイルの展開時に風を取り込むことでパラフォイル全体が膨らみ、機体の姿勢を安定させることを目的としたものである。この空気セルを導入したことで後述する旋回制御の際に翼がつぶれてきりもみ落下するリスクを軽減することができた。

また、紐の接続部には補助用のパーツを新たに作製し、通常飛行時に前方にやや重心が偏るようにした。これにより進行方向を常に前方に固定することを可能にした。さらに、翼を前方に傾けることで飛行時に空気セルにより風が入りやすくなり、安定性の向上も確認された。

2. 制御

機体の旋回制御はパラフォイルの紐を引くことで行う。

原理としては、紐を引くことで翼の後縁を下げ、翼全体を傾けることによってパラフォイルにかかる揚力が重力と遠心力の合力とつり合うようにして旋回するというものである [7]。

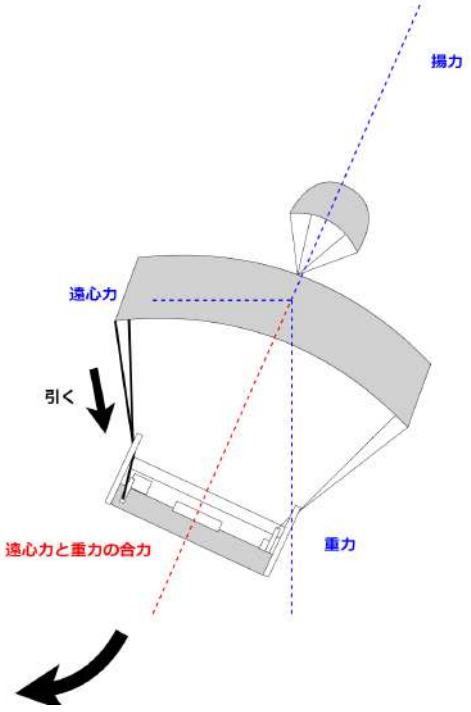

図 3.3　パラフォイルの旋回挙動

この制御方法ではパラフォイルの翼を傾ける必要があるが、当初は翼の片翼に接続している紐すべてを引くことで翼を傾ける想定であった。しかし、実際に飛行実験を行ったところ、旋回時に翼が変形し墜落してしまうなど安定性に欠ける結果が多く信頼性に欠けるものであった。

そこで、以下のような条件を比較した実験を行い最も安定したものを採用することとした。

条件 1. 最も外側の紐計 12 本を左右それぞれで束ねたもののみを引く
条件 2. 翼内側の 4 セルの紐計 12 本を左右それぞれで束ねたもののみを引く
条件 3. 翼前方の紐計 14 本を左右それぞれで束ねたもののみを引く
条件 4. 翼後方の紐計 14 本を左右それぞれで束ねたもののみを引く

この条件のパラフォイルに約 120g の重りをつけ、未来大体育館の約 9m の高さから翼を開いた状態で投下し、安定性と旋回の度合いを比較した。また、紐を引く長さは 50mm, 100mm, 150mm の 3 種類で実験を行った。

結果は条件 4 がどの紐の長さにおいても最も安定し、かつ大きく旋回するというものだった。

条件 1 は旋回開始時すぐに翼が変形し墜落するという大きく安定性に欠ける結果になった。

条件 2 は 50mm の長さでは最も安定した飛行を見せたが、100mm,150mm の長さでは旋回時に翼が変形しする確率が高く、実際の制御で採用するには信頼性に欠ける結果であった。

条件 3 も条件 1 と同じく旋回開始時に翼が変形し、墜落することが多く安定性に欠ける結果となった。

条件 4 はどの紐の長さにおいても旋回時に翼の大きな変形は見られず、墜落することもなかった。また、紐を 100mm 引いた際に約 9m 下の床に到達するまでに機体の向きが投下方向から見て完全に逆方向を向くなど旋回性に関しても十分な結果となった。

この実験からパラフォイルを用いた CanSat では翼後方を引くことで旋回を行うのが安定した制御方法であることがわかった。本グループで使用したパラフォイルはこの機構を採用して制御を行うことに決定した。

図 3.4　条件 4

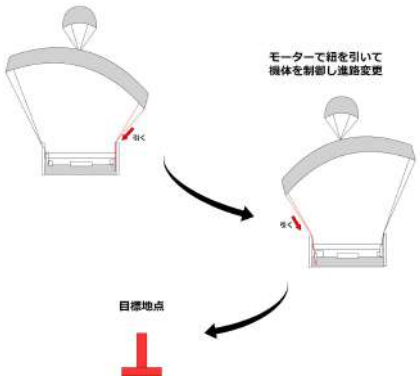

図 3.5　パラフォイル制御図

（※文責: 大岩穂峻）

パラシュート

本グループは、パラフォイルとは別に展開を促す目的で展開用パラシュートを作製した。また、パラシュートの作製にあたり、宇宙開発でも使用できるような岩谷式パラシュートの構造を参考にした [8]。 形状としては、正六角形の形で真ん中に飛行を安定されるための通気口を開けたものである。

図 3.6　作製したパラシュート

1. 設計・作製

パラフォイルが展開用パラシュートにより引っ張られ、パラフォイルが均一に広がるようなパラシュートを作製する必要がある。

パラシュートの作製は以下のようなステップを踏んで行った。

STEP 1 ナイロン素材を六角形に切り取る

始めに、ナイロン粗大に定規と三角定規の 60 度部分を使い、六角形を描いていった。上の写真はパラフォイルとパラシュートで、パラシュートはパラフォイルの上部に取り付けている形になっている。大きさは、対角線が約 40 センチとなるように設計した。展開用のため、大きく作りすぎるとパラフォイルの制御を妨げる可能性があるため、このサイズを選択した。

STEP2 6 つの穴を開ける

次に、六角形のそれぞれの角に対して紐を通すための穴を開けていった。穴を開ける際は、アイスピックを使用し適当な位置に穴を開けた。そして、6 か所に穴を開けたら、補強用の保護シールを貼って紐に引っ張られた際にも耐えれる仕様に工夫した。

STEP3 紐をパラシュート本体に取り付ける

最後に、穴を開けた 6 か所に紐を通していった。紐の長さは、対角線の長さ 40 センチより少し長めに設定した。6 本の紐を一つにまとめて完成である。

2. 効果

パラシュートは、パラフォイルの上部に取り付け、上部の画像のように展開しパラフォイルの完全な展開、飛行安定性を支える役割を果たす。具体的な展開方法は、投下時にパラシュートがパラフォイルより先に展開して、機体に収納された飛行用のパラフォイルを釣り出すことでパラフォイルの展開の支えになる。

図 3.7　展開の様子

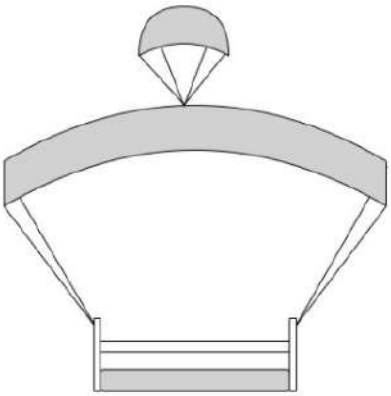

図 3.8　パラシュート概要図

3. 性能評価

性能を確かめるために、体育館と笹流ダムで実験を行った。手順として 2 通りの投下方法を示す。1 つ目は、投下の際にパラフォイルを開いた状態で落とし、2 つ目は、投下の際にパラフォイルとパラシュートを折り畳み機体に収納した形で落とした。結果として、1 つ目の条件ではうまく展開用のパラシュートが機能しパラフォイルが完全に開くことを補助した。2 つ目の条件は、折りたたんだ状態であるためパラシュートが上手く開かず自由落下することが実験を行うことで判明した。ゆえに、折りたたんだ際にパラシュートが展開できるための機構を作製する必要があることを確認した。

（※文責: 富樫幹生）

機体

本グループでは機体の飛行誘導を行うためにサーボモータでパラフォイルの制御紐を引っ張り、進行方向を制御することにした。以下に示すのが制御の方法を加味した機体の概要図である。外形は 150mm、最大長 300mm で設計した。サーボモータの動作に干渉しないよう、また、スペースプローブコンテストで定められる既定の重量に沿わせる目的で複数穴をあけた。また、この穴は空気孔としての役割も同時に担っており、パラフォイルの展開の際に必要になる空気を真下から取り入れる目的もある。

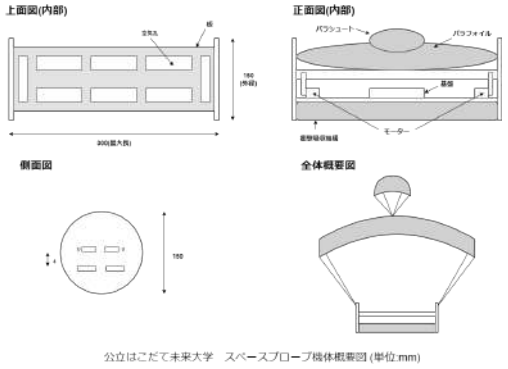

図 3.9　機体概要図

前期では機体内部で紐が絡まらないよう、また、他の部品に干渉することを防ぐために滑車を採用し、滑車を機体内部に取り付けるための部品を 3D プリンターで作製した。形状のモデリングは教育機関用 fusion360 をもちいて作製した [9]。滑車は「シングルプーリー」の名称で市販されていたものを用いて作製し、部品に滑車が収まることを確認した。実際に部品に滑車を収めた状態の図を以下に示す。

図 3.10　シングルプーリー 横

図 3.11　シングルプーリー 縦

機体の形状について、スペースプローブコンテストの定めるレギュレーションに沿うよう内部構造を検討し、円柱を倒した形状で内部を二層に分けた機体を作製した。この二層はマイコン等を配置する階層と、投下前にパラフォイルを格納しておく階層からなる構造となっている。

実際に作製した機体は以下の通りである。

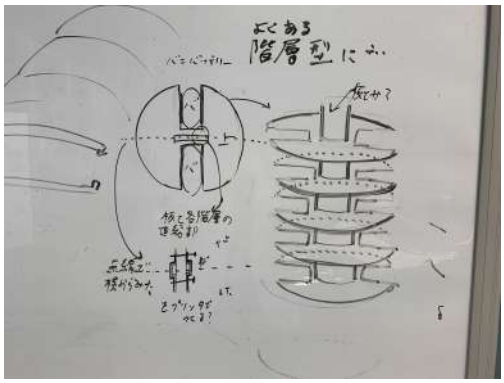

図 3.12　当初検討していた構造のイラスト

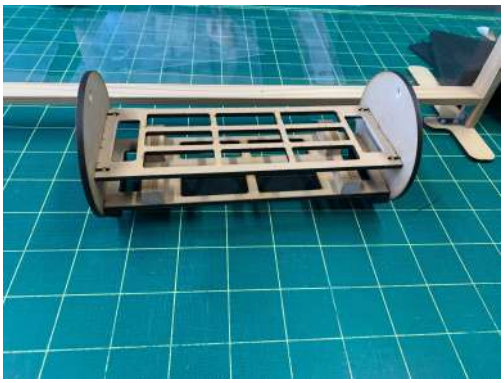

図 3.13　実際に作製した機体の構造

機体の素材には MDF を使用した。MDF を使用した理由として、市販しているため材料に困らないこと、レーザーカッターで加工がしやすいこと、木が原料ではあるが MDF は木の細胞の大きさにまで小さく解体、再び接着剤で固めたものであるため木の芯や木目を考慮する必要がないことが挙げられる。 Illustrator でパーツごとの図面を作製し、レーザーカッターを用いて切り出し、組み立てて作製した。以下に示すのが実際に部品を作製した際の図面である。なお、学生のレーザーカッターの使用には事前の講習と許可証が必要になるため事前にレーザーカッターの使用に関する講習を受けてから作製を進めた。

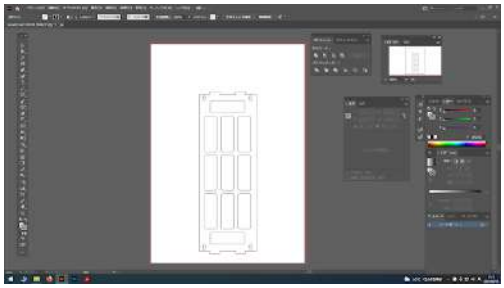

図 3.14　実際に作成した図面

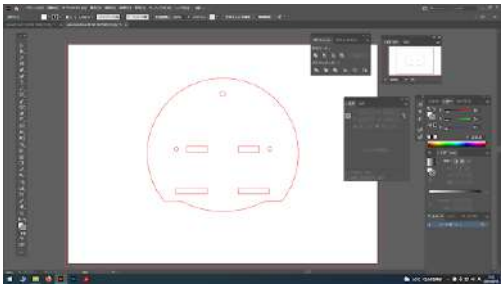

図 3.15　実際に作成した図面

また、この際ナットを MDF の中に埋め込める形で MDF を切り出すことで、本来ネジ止めが難しい方向からのネジ止めを可能にし、機体のパーツ同士の接続の強度を確保した。加えてなるべく角を少なくすることで衝撃を受け流せるように設計した。

図 3.16　ナットを埋め込んだ際の実際の図

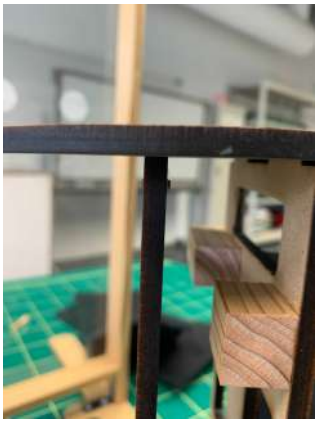

図 3.17　ナットを埋め込んだ際の実際の図

パラフォイルを投下前に格納しておく階層は図 3.19 で示される上の階層である。機体側面にはプラスチック製の透明なブレードを取り付けることでパラフォイルが固定位置がずれる、投下前に落ちてきてしまうなどの危険性を極力減らせるよう努めた。

なお、実際にパラフォイルを格納したものを図 3.20 に示す。投下の際は図 3.20 の状態で投下する形となる。

この外見がボンレスハムのように見えることから機体名はボンレスハム号となった。

図 3.18　機体の階層

図 3.19　実際にパラフォイルを格納した図

衝撃吸収機構について、機体下部に衝撃を吸収するための機構を備えた。「市販の衝撃吸収材を使用する」案を採用し、作製した。使用した衝撃吸収材は商品名「スポンジ」として販売されていた素材であり、これをカッターで適切な大きさに切り出し市販の接着剤と硬化促進剤を用いて固定した。

図 3.20　機体下部の衝撃吸収機構

（※文責: 佐藤大地）

軌道分析

後期に行った実験において、機体の投下の様子を撮影し、飛行時の軌道、スピード、位置について解析を行った。解析には windows 向けのフリーの動作分析ソフトである Kinovea[10] の ver0.9.5 を用いた。

1. 軌道分析手順

ここでは Kinovea 上で行った解析の操作方法について解説する。

Kinovea 起動時の画面は以下に示すとおりである。

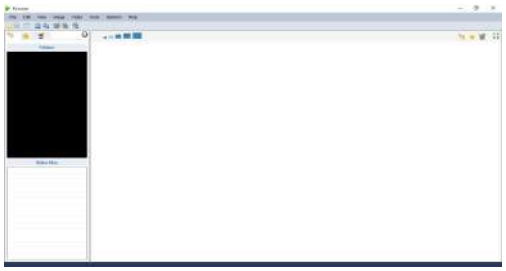

図 3.21　Kinovea(ver0.9.5) 起動時画面

この画面から左上部の「File」→「Open video」を選択し、軌道を解析したい動画ファイルを選択する。

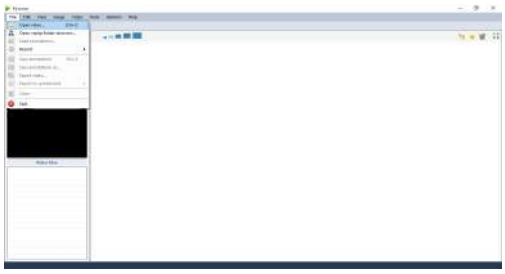

図 3.22　動画読み込み手順

動画を読み込むと以下のような画面が表示される。

図 3.23　動画を正常に読み込めた際の画面

次に投下した瞬間のフレームにカーソルを合わせ、画面左下に緑のアイコンで示される作業範囲を変更する機能を用いて作業範囲の始点を定める。作業範囲の終点も同様の操作で機体が地面に着地した瞬間に定める。

図 3.24　作業範囲の指定

次に動画上を右クリック→「軌道解析」を選択するとマーカーが現れるのでマーカーの中心を軌道を解析したい物体に合わせる。

図 3.25　軌道解析に必要なマーカーの表示

この状態で動画を再生するとマーカーが自動で対象物の位置変化に合わせて対象物を追従し、軌道が得られる。

2. 軌道が正常に得られない場合

前項では自動でマーカーが対象物を追従し、軌道が得られると説明したが、以下に示すように、周囲の環境などによりマーカーが途中で別の物体を追ってしまい正確な軌道が得られない場合がある。この際の対処法について説明する。

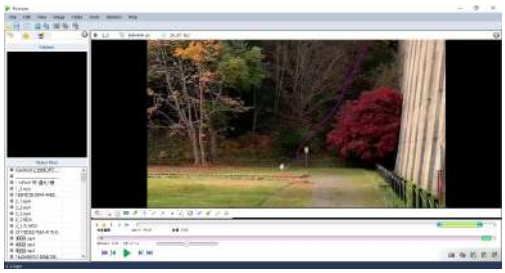

図 3.26　看板によって軌道が正確に得られなかった例

この場合、軌道がそれる直前のフレームの軌道にカーソルを合わせ、右クリック→「このポイント以降の軌道を削除」を選択。

その後 1 フレームずつ動かし終点まで手作業でマーカーの位置を再設定する。なお、フレームの移動は十字キーで行える。

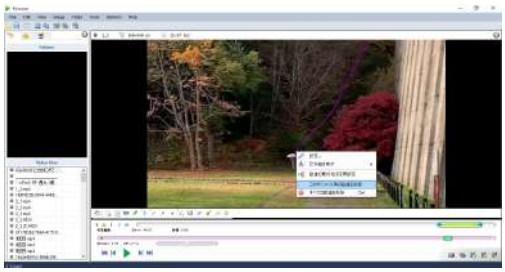

図 3.27

この結果以下のように正常な軌道が得られる。

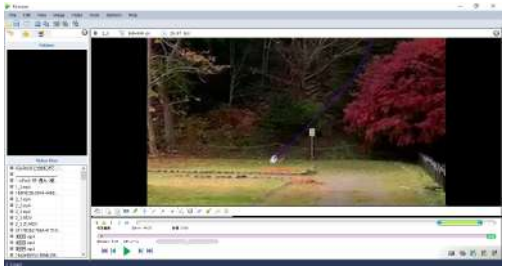

図 3.28

3. 軌道のグラフへの落とし込みと出力

ここまでで対象物の軌道が得られた。ここでは得られた軌道をグラフに落とし込む方法について説明する。

軌道が得られた状態で画面上部の「Tools」から「Linear Kinematics...」を選択。

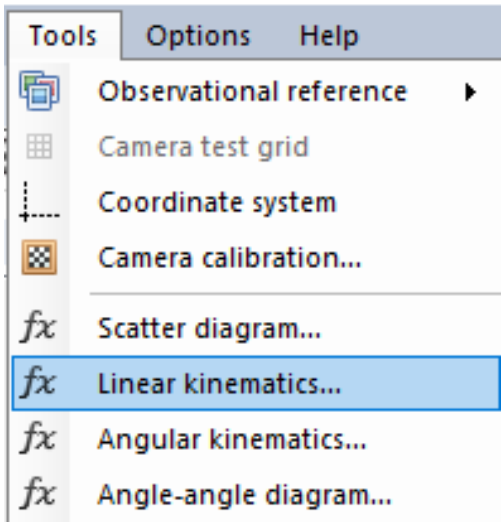

図 3.29

Data のプルダウンリストからから欲しいデータに関してグラフを出力できる。

グラフの保存に関しては「Export data」の「Save to file」からグラフを保存できる。

以下に示すのは垂直方向の速度の変位であり、他にも加速度や高さの変位をグラフとして出力できる。

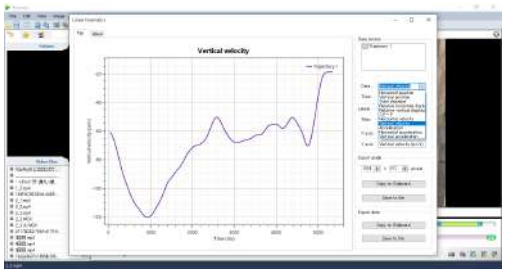

図 3.30

（※文責: 佐藤大地）

3.1.2 制御班

アルゴリズム

スペースプローブコンテストでは空中制御を主として扱ったため、安定した視界を確保できず画像認識等での目標の識別及び誘導を断念せざるを得なかった。そのため、GPS と地磁気等のセンサ系のみを用いた制御アルゴリズムを考案した。特に、GPS は精度の維持のため 1 秒に一回程度の頻度でしかデータを取得することができなかったため、地磁気に重きを置いた制御方法となった。また、今回は 100 m上空からの投下ということもあり、空中にとどまれる時間が短く体勢が崩れた場合立て直すことが難しいと判断し、あえてサーボモータによって引く紐の量を一定にすることによって制御量の安定化を図った。

最初に加速度センサを用いて落下検知を行い、検知に成功したのちに GPS を用いて自身の現在位置を取得する。地磁気を用いて緯度に対する絶対的な機体の正面方向の角度を求める。次に事前に入力されていた目標地点と現在地の X 軸、Y 軸方向の差を算出し、そこから経度に対する tan 角を求める。以上の二つから修正するべき相対的な角度を割り出し、パラフォイルの制御紐の長さをサーボモータによって制御し角度を修正、目標地点へと誘導する。

アルゴリズムの検証のため、仮想環境を用意し実験を行った。仮想環境では事前に取得できると仮定された加速度データ、地磁気データ、ジャイロのデータを準備しておき、入力として与えた。実験の際は制御角の計算に成功しているかの実験を行った。また、その計算された制御角での実機での実験を行い、モーターでの制御量が十分であるかを足しかめる実験と落下検知をのみを目的とした実験を行った。

1. 制御角の計算を目的とした実験

本実験では事前に入力として与えられていた目標地点の仮想の GPS 座標と理想的な機体の軌道の想定の下に導き出された GPS 座標を用いて想定された制御角が導き出されるのかの実験を行った。この実験の結果、制御角を導き出すことには成功したが、仮想データの値の問題か修正角が小さく誤差の範囲内としてとらえられる程度の結果し変えられなかった。そのため、誤差ととらえる閾値を低くすることとして修正を行い、適切な制御が行えるようにした。

2．1 での制御角をもとに実機を用いた実験

本実験では 1 で得られた結果をもとに、サーボモータの制御量を変化させどの程度の制御量でどの程度の時間で角度の修正が行われるかを確かめる実験を行った。この実験の結果、求められている制御角の修正を行う事には成功したが、修正にかかる時間に問題があることが判明した。実験では約 9m からの投下を行い十分な制御が行えているのかを確かめたが、2 回目の制御にかかって修正を行っていたことが分かったため、制御量を増やすことにして 1 回目の制御で求めている量の制御が行われるようにした。

3. 落下検知を目的とした実験

本実験では Z 軸方向の加速度の 5000～0 を閾値としてパラフォイル等の展開なしでの投下実験を行い、正常にシステムの動作が始まるのかを確認する実験を行った。この実験の結果、当初の閾値であっても検知に成功することはあったが、安定した検知の成功とはならず、5000～-5000 の閾値というように修正を行うことした。

アルゴリズムは以下の図のようになっている。

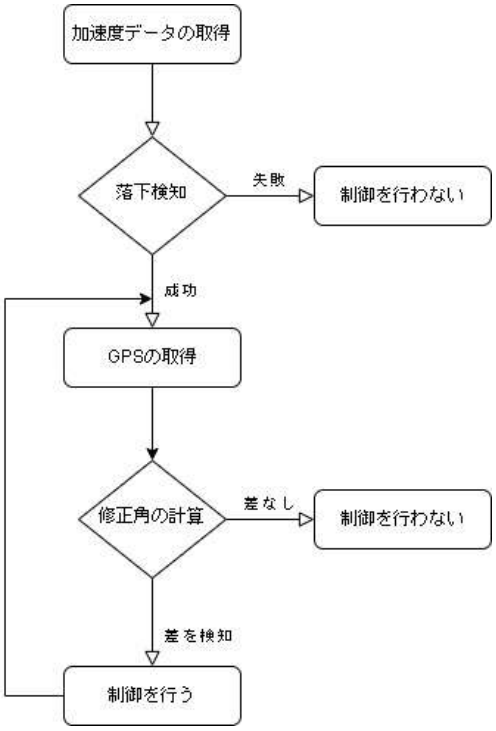

図 3.31　アルゴリズムのフローチャート

（※文責: 西殿大輝）

電装

本グループの参加するスペースプローブコンテストでは目標地点に機体を飛行誘導する必要がある。

メインの制御用マイコンとして Raspberry pi zero を採用し、電源としては 5v のモバイルバッテリーを用意した。Raspberry pi zero が各センサ、モータなどの制御を行う。

そのため、本グループでは機体の飛行誘導を行うためにサーボモータでパラフォイルの制御紐を引っ張り、進行方向を制御することにした。パラフォイルにかかる風圧などの関係からサーボモータのトルクによっては制御紐を引っ張ることができない可能性があったため、トルクが異なる 3 種類のサーボモータを購入しそれぞれについて検証を行った。検証後使用するサーボモータを決定したが、電源電圧がメインシステムとは異なるため、別途サーボ用の 7.4v バッテリーを搭載した。

次に、進行方向の決定のため、GPS を用いて機体位置の特定をすることにした。スペースプローブコンテストでは目標地点の GPS 座標が配布されるため、GPS を用いて機体位置の特定をし、目標地点との差分を求め制御の指針を決定する。

また、位置特定の補助や機体の状態の確認のため、機体の加速度検知や姿勢検知が可能な 9 軸センサを使用することにした。今回使用する 9 軸センサは加速度、ジャイロ、磁気を測定することができるセンサである。今回は加速度センサで機体の加速度を検知して制御データとして使用する予定である。加速度センサで取得した値は落下速度の計算と風による機体への影響を計算することに使用する。

そして、これらのモジュールを搭載した機体の開発を円滑に進めるため、機体の行動履歴を調べることができるよう SD カードを組み込み、各種データを記録した。また、スペースプローブコンテスト当日のデータも SD カードへの書き込みにより記録し後日取得する予定だったが、パラシュートが開かなかったためデータを得ることができなかった。

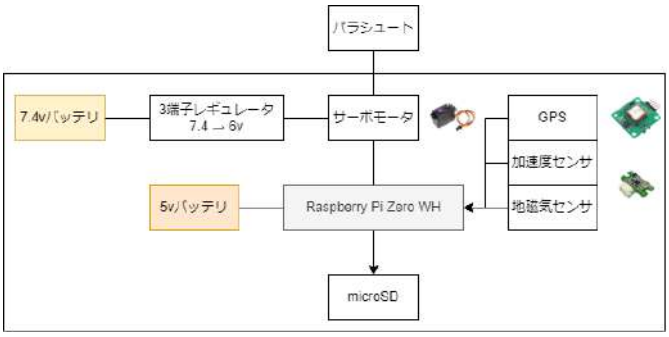

図 3.32　全体構成概要図

（※文責: 鈴木進太郎）

メインプログラム v1

当初 PIC を使用した開発を行ったことがあるメンバが 2 人いたためマイコンには PIC を用いていたが、SD カードの制御や GPS、9 軸センサからの値取得が難しくスペースプローブコンテストに間に合わないと判断した。よって PIC マイコンより実装が容易な RaspberryPi zero を採用し、Python にて開発を行った。

GPS との通信を行うためのプログラムには、データの取得を簡単に行うために microGPS ライブラリを用いた。また、扱いやすいよう GPS クラスを作成してデータ取得の関数群を実装した。

9 軸センサからデータを取得するためのプログラムには lsm9ds1_rjg ライブラリを用いた。また、サンプルコードとして提供されている simple.py を改良し、各データを取得できる関数群を定義した LSM9DS1 クラスを作成した。

制御手順に関してはアルゴリズムの項の通りである。

続いて、使用したライブラリやモジュールと主要なクラス、メソッドの説明を行う。

始めに、ライブラリ、モジュールについて説明する。

外部モジュール

microGPS

Python3 系と MicroPython で動作する GPS データの解析用ライブラリで、GPS から送られてきた NMEA フォーマットの文字列を update 関数に 1 文字ずつ渡すことでデータを整理し、日付や緯度経度などの各値ごとにまとめることができる。

lsm9ds1_rjg

Raspberry Pi から LSM9DS1 へアクセスするためのデバイスドライバで、i2c, spi 通信の両方に対応している。

pyserial

シリアル通信を行うためのライブラリで、デバイス名、ボーレート、タイムアウト値を指定することで簡単に接続することができる。

自作モジュール

main

落下検知を行い、コントローラーによる制御を呼び出すモジュールで、RaspberryPi 起動時に自動的に実行される。

csvWriter

ログファイルの生成と書き込みを行うモジュールで、main から呼び出される。

simple

lsm9ds1_rjg で提供されているサンプルを改良し、本機体で動作するように調整した LSM9DS1 へアクセスするためのモジュール。

GPS

本機体で使用している GYSFDMAXB から整ったデータを取得するためのモジュール。

servo

本機体で使用しているサーボモータ・MG996R および LG20MG を制御するためのモジュール。

Controller

全モジュールを包括し、機体の制御を行うためのモジュール。

続いて、クラス図を以下に示す。

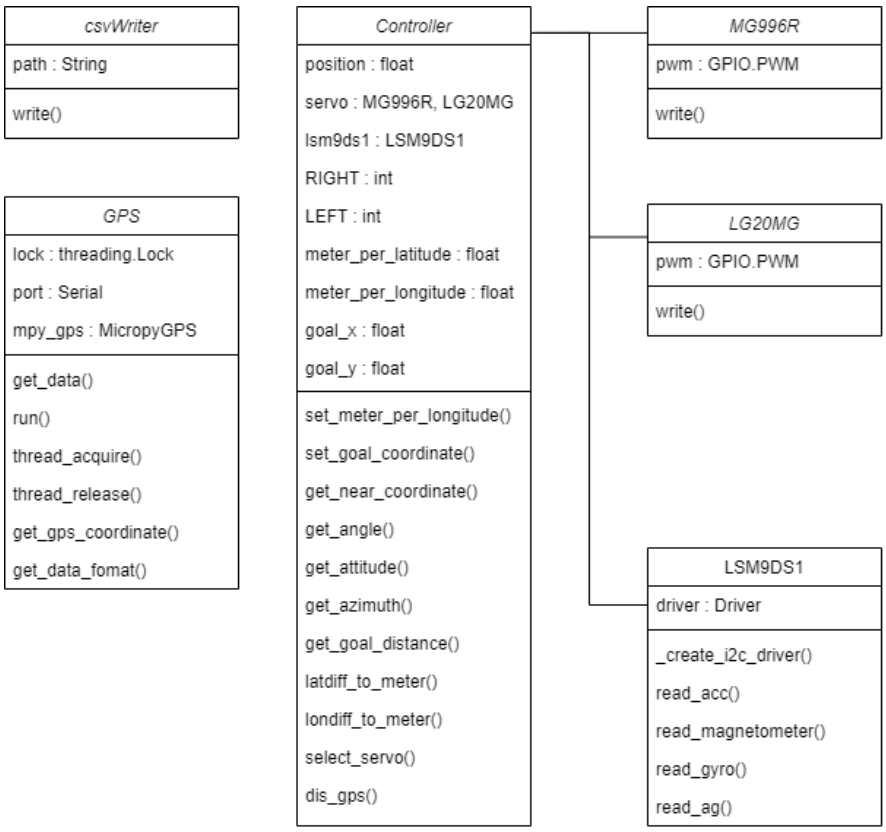

図 3.33　クラス図

各クラスとメソッドについて説明する。

class GPS(threading.Thread)

GPS との通信を開始し、データを取得するためのクラスである。

def get_gps_coordinate(self)

緯度経度をタプルで取得する関数である。

def get_data(self)
日時、時間、緯度経度を辞書として取得する関数である。

class LSM9DS1
9 軸センサとして採用している lsm9ds1 からデータを取得するためのクラスである。

def read_gyro(self)
x,y,z 軸の各ジャイロ値をタプルで取得する関数である。

def read_magnetometer(self)
x,y,z 軸の各地磁気値をタプルで取得する関数である。

def read_acc(self)
x,y,z 軸の各加速度値をタプルで取得する関数である。

class csvWriter
csv ファイルを生成し、データを書き込むためのクラスである。

def write(self, data)
生成した csv ファイルにデータを書き込むための関数で、引数で与えられたデータを 1 行分書き込む。

class MG996R
サーボモータとして採用している MG996R を制御するためのクラスである。コンストラクタで接続ピンに pwm モードで設定している。PWM サイクルが 20ms であるため周波数は 50Hz で設定した。

def write(self, angle)
制御パルスが 0.5ms から 2.4ms である mg996r において、0 から 180 度までの角度を指定し、サーボモータを制御する関数である。

class LG20MG
サーボモータとして採用している LG20MG を制御するためのクラスである。コンストラクタで接続ピンに pwm モードで設定している。PWM サイクルが 20ms であるため周波数は 50Hz で設定した。

def write(self, angle)
制御パルスが 0.5ms から 2.5ms である lgm20 において、0 から 180 度までの角度を指定し、サーボモータを制御する関数である。

class Controller

アルゴリズムに用いる関数群や、サーボモータ・9 軸センサクラスのインスタンスを持ち、機体の制御を行うクラスである。すべてのモジュールの操作はこのクラスを介して行う。

dis_gps(self, gps_x, gps_y)

目標地点までの距離や機体の向きを取得し、サーボモータの制御をおこなう関数である。

（※文責: 杉山宏輝）

メインプログラム v2

スペースプローブコンテストにおいて機体と制御プログラムなどの作成を行ってきたが、そのテストを行う際、ハードウェアを使っていることや、風、温度、落下時の機体の角度などの状態によって様々な結果が出る。その影響でプログラムが正常に動作しているかなどが実験では正確にわからないことがあった。そのため主に後期の活動では、制御に必要な加速度、地軸、GPS などから得られるデータを仮想的に返すモジュールを組み込んだメインプログラム v2 の作成を行った。しかし、まだ製作途中のため、来年度プロジェクトに引き継ぐことにした。

まず、もともとテスト用に CSV ファイルからデータを返すだけの仮想センサモジュールを Python で作成した、動かす前に、そのセンサにあったデータを予め定義する必要があるが、実際にセンサを動かす必要がないため、プロジェクトが始まってすぐのアルゴリズム検証に使用していた。

その後、メインプログラム v2 を作成する段階で元の制御用プログラムの設計を見直した。見直した結果、最適化がされていない処理や命名規則、さらに適切なクラス化がなされていないといった問題があったためその修正を行った。

次にテストを行う際のモジュールの設計を考えた。制御用マイコンには Raspberry Pi Zero を用いており、OS として Linux が動作可能なことから OS の機能を生かした設計となった。サーバーをシステムの根幹として API 経由でセンサからデータの受け取り、モータの制御などを行えるような設計である。プログラムの実行時にテストか本番または落下実験なのかを指定することで、サーバーから帰ってくるデータが仮想センサモジュールか実際のセンサなのかを切り替えるような機能を考えた。

また、実際のセンサを使わなければ Raspberry Pi Zero である必要がない。そのため Docker や Docker-compose などを用いてどの環境でもすぐに起動できるようにすることで、実際のセンサやモータを用意しなくても、仮想的に様々なデバイスの上で動かせるようにした。こうすることで、ハードウェアの制約から開放され、プロジェクトメンバーがそれぞれ自分のコンピュータ上で開発を行えるような設計を考えた。

（※文責: 鈴木進太郎）

3.2 笹流れダム投下実験

スペースプローブコンテスト終了後に笹流れダムにて投下実験を行った。

（※文責: 大岩穂峻）

3.2.1 実験概要

11/6,11/7 の二日間の日程で笹流れダムにて投下実験を行った。実験目的は落下検知及び制御システムの動作検証、パラフォイルの制御の検証であった。

実験内容は機体を笹流れダム頂上 (高度 25.3m) からパラフォイルを開いた状態で投下し、設定した目標地点に対しての飛行時の動作を記録するというものである。その際の投下方向は南西方向であり、目標地点は投下地点から南西方向の直線上の地点に設定した。さらに、比較条件として、マイコンの電源を on にしてシステムを動作させた状態での投下と電源を off にして制御を行わない投下を行った。

また、11/6 は無風、11/7 は北西方向に 4m/s の風が吹いていた。

（※文責: 大岩穂峻）

3.2.2 実験結果

2 日間でマイコンの電源を on にした制御ありの投下を 6 回、off にした制御なしの投下を 2 回の計 9 回の投下を行った。なお、マイコンの電源を on にした投下のうち 1 回は電源ケーブルが抜けた状態で投下を行っていたことが投下後に判明したため、制御なしの投下として扱う。

パラフォイル

2 日間で 9 回の投下を行ったがいずれも安定した飛行を見せ、飛行用の減速機構としては十分な性能であることを確認できた。また、モータで紐を引っ張ることによる機体の旋回も確認できた。しかし、投下直後から徐々に進行方向が右によれていくなど、作製の際に歪みがあったことが発覚した。2 日目の風速 4m/s の条件ではその傾向が顕著であったが、1 日目の無風の条件でも少々その傾向があった。さらに、2 日目の投下の際には風に煽られて進行方向が急激に変わってしまうなど風に対しての弱さも確認できた。

機体本体

2 日間で計 9 回の投下を行ったが実験後に機体本体や内部のマイコンに損傷は見られなかった。2 日目の最後の投下で着地時に地面の岩と接触してしまったがここでも破損することなく実験を終えることが出来た。これにより機体本体に施した工夫や、衝撃吸収機構が機能したことが確認できた。

制御

コンテストでは落下検知を抜けることができなかったためログファイルを生成することができず、その後の制御プログラムも実行されなかった。そのため、本実験では落下検知の妥当性を確かめるとともに、ログファイルの生成および制御が正しく行われるかどうかを検証した。

マイコンの電源を on にして行われた有効な 5 回の投下のうち、4 回はログファイルの生成および書き込みに成功していた。しかし、1 度の投下で複数のファイルが生成されていることがあった。これはシステムが終了したのちに再起動されることが原因であり、そのたびに落下検知が行われ、抜けていると考えられる。よって投下時以外の揺れや衝撃などでも検知を抜けている可能性があり、落下検知の妥当性を確認できなかった。制御においては風やパラフォイルのゆがみ、重心等によって投下直後に目標地点から右に逸れていくことが多かったが、制御プログラムが実行された投下では、サーボモータがひもを引っ張ることで左へ旋回する動作が見られた。このことから、落下検知を抜け、制御プログラムが実行された投下では機体の制御が行えることが確認できた。

図 3.34　笹流ダム

（※文責: 大岩穂峻）

第 4 章　インターワーキング

この章ではプロジェクト全体の進捗やグループの相互間の活動を円滑に進行するために行ったプロジェクト内における活動について述べる。

（※文責: 鈴木進太郎）

4.1　プロジェクト全体のインターワーキング

本プロジェクトでは活動目的によってグループ分けを行い、A グループと B グループの 2 グループで活動を行った。本プロジェクトでは、プロジェクト全体でのオンラインミーティングを週に 1 度行い、適宜大学登校をしながら活動を行った。また、全体ミーティングとは別に、各グループのリーダーが参加するリーダーミーティングを週 1 回の頻度で開催し、プロジェクト全体の運営の円滑化を図った。

プロジェクトの体制として、プロジェクト全体の管理を行うプロジェクトリーダー、各グループの管理を行うグループリーダー、ツールの管理を行う情報システムの役職を設けて活動した。

使用ツールに関しては、昨年度同様ミーティングに Zoom を使用し、メンバー間の意見交流や連絡には Slack を導入した。また、資料は Google ドライブで管理、共有を行い、一部のタスク管理には Notion を活用した。コードの管理は Git, GitHub を使用した。

（※文責: 鈴木進太郎）

4.2　A グループ内のインターワーキング

本グループでは 9 月のスペースプローブコンテストに参加するため、それまでに CanSat システムを完成させる必要があり、他のプロジェクトに比べて取り組む時間が短い。そのため、作業の効率化を意識してグループの進捗管理を行うために、Notion というツールを導入しナレッジマネジメントというマネジメント方式を採用した。また、コードの管理には Git, GitHub を採用した。

（※文責: 鈴木進太郎）

4.2.1　ナレッジマネジメント

ナレッジマネジメントとは「知識を共有して活用することで、新たな知識を想像しながら経営を実践すること。」[11]。

ナレッジマネジメントには「暗黙知」と「形式知」という考え方がある。暗黙知は個人が持っている勘やノウハウを含めた言語化できない知識であり、「形式知」は図や文章などで表せる言語化が可能な知識のことである。 本来ナレッジマネジメントは経営理論であり、知識を共有して新たな価値を創造するといったことを期待するものであるが、本グループでは暗黙知を形式知にしてグループ内で共有、決定事項などの情報の共有などに重点を当てて運用を行った。

（※文責: 鈴木進太郎）

4.2.2 コード管理

本グループでは CanSat システムで使うコードを Git, GitHub で管理を行った。プログラミングを担当したのは 3 名であり、それぞれが別のプログラムを書いているため、Git のブランチという機能を使い効率よく作業を進めた。また、コードの問題点や修正点などは GitHub の issue をベースに管理した。

（※文責: 鈴木進太郎）

第 5 章　成果

この章では、中間と最終成果の二回に分けて行われた発表会の概要と、開発やスペースプローブコンテストの成果について述べる。

（※文責: 大岩穂峻）

5.1　プロジェクトの成果

この項ではプロジェクトの成果について述べる。

（※文責: 大岩穂峻）

5.1.1　スペースプローブコンテスト前の成果

前期のスペースプローブコンテスト参加前の成果として、まず活動を通して得られた知識について説明する。活動を通して、CanSat を設計及び作製する過程で電子部品 (基盤や PIC マイコン等) や道具 (はんだや 3D プリンター等) の取り扱い、ハードウェアを利用するプログラム等についての知識を得ることができた。

次に、スペースプローブコンテスト参加時点での成果物について説明する。機体班では、機体外形及び飛行用パラシュート・展開用パラフォイルの設計、作製を行った。機体外形は下部に MDF 素材の階層構造を用い、衝撃吸収機構を備えたものであった。パラフォイルは安定性向上用の空気セルを用いた構造で作製した。パラフォイルは後部の接続部につながる紐を引くことで進行方向を変更するという制御方式であった。パラシュートはパラフォイルに先行して展開することでパラフォイルの展開補助を行うことを目的としてパラフォイルの上部に接続する形で作製した。制御班では、アルゴリズムの検討と制御システムの作製、制御を行った。アルゴリズムは GPS と地磁気の値によって目標地点へと誘導するというものであった。制御システムは GPS 等各種センサの値からサーボモータの角度制御を行うというものであった。また、加速度センサによる落下検知によってシステムを稼働させる形式とした。

次に、実験結果について説明する。前期のスペースプローブコンテスト参加前の実験結果としては、パラフォイルの減速機構及び旋回制御が可能であることを確認できた。また、制御用のサーボモータのトルクがパラフォイルを引くに耐えうることを確認できた。さらに、落下検知が可能であることが確認できた。

一方で、スペースプローブコンテスト前の懸念点としては、制御システムの飛行テストが行えていないことや、パラシュート及びパラフォイルの展開が不確実であることが挙げられた。

（※文責: 大岩穂峻）

5.1.2　大会結果

スペースプローブコンテストにおいて、本機体はパラシュートの固定器具が不具合を起こし展開用パラシュート・飛行用パラフォイルともに展開せず垂直落下するという結果に終わった。

達成した事項については、衝撃吸収機構の動作確認は成功したことが挙げられる。パラフォイル等の展開に失敗したため垂直落下する結果となったが機体本体に破損はなく、搭載していたモジュールにも損害は見受けられず、正常に動作していた。要因としては、機体下部に取り付けられた衝撃吸収機構がうまく地面と接触したことや、衝撃分散のため機体に角を作らないようにしたこと、草が生い茂っていたこと、連日の雨で地面がぬかるんでいたことなどの要因が考えられる。しかし、ほかの出場チームで垂直落下をしてしまった機体は落下による衝撃で機体そのものやシステムが破損してしまったものがほとんどだった。そのような点で本機体は衝撃吸収機構正常に作動しており、かつ他チームより優れていたと考えられる。

一方で、展開用パラシュート・飛行用パラフォイルの展開は失敗した。原因としてはパラシュートの固定器具の強度が予定よりも強く、想定とは違う箇所が破損、ドローンからの分離には成功したもののパラシュート及びパラフォイルの展開に支障が出たと考えられる。また、落下検知に失敗した。落下時、パラフォイル等の展開に成功しており、機体がある程度水平を保った状態で動作を行い、Z 軸方向の加速度の変化量から判断を行う予定であった。しかし、パラフォイル等の展開に失敗したため機体が回転し、Z 軸方向の加速度の変化が想定とは違った値を示したため、落下検知を行うことができなかった。

また、プログラムの制御機能とログの書き出しは落下検知を行うことで動き出すような設計であったため、機体落下時にプログラムの状態などを確認することができなかった。

スペースプローブコンテストにおける評価（点数）は「着地の正確性」「事前準備」「製作技術」「当日の運用」「プレゼンテーション」「オリジナリティ」の 6 要素を各 0〜100 点で評価、それぞれの合計得点で算出される。

今回のスペースプローブコンテストにおける本チームの評価は以下に示す表とレーダーチャートの通りである。

図 5.1　スペースプローブコンテスト [12]

（※文責: 西殿大輝）

5.1.3　スペースプローブコンテスト後の成果

スペースプローブコンテスト後の成果として、作製した機体とプログラムの両方について説明する。

機体

機体の詳細については節 3.1 で解説しているため省略するが、節 3.2 で記述した笹流れダム投下実験において、パラフォイルの飛行制御による目的地誘導が機能していることが確認できた。これは節 2.2.2 サクセスクライテリアにおいて設定したミニマムサクセス (最低限 CanSat の制御がなされていることが確認できる) を満たすものであった。また、この実験では減速機構が正常に働いた計 9 回の投下を行ったが衝撃吸収機構の働きにより機体に損傷はなかった。これは衝撃吸収機構が機体を保護する役割を満たしていることを確認できるものであった。最終的にはその条件に加え、収納したパラフォイルの展開機構を備えた機体を作製した。展開機構については、パラフォイルの上部にパラシュートを接続する機構であった。展開動作としては畳まれたパラシュートが重力によって先んじて展開することで接続されたパラフォイルが釣られるようにして展開するというものである。また、この動作を検証する実験として未来大体育館にて新規に投下実験を行った。この投下実験では約 9m の高さから機体を投下したが、その間にパラシュート及びパラフォイルの展開を確認することができた。

プログラム

プログラムの詳細については節 3.1 で解説しているため省略するが、スペースプローブコンテスト後はメインプログラム v2 の作成を主に行った。スペースプローブコンテスト前のメインプログラム v1 については実装期間が短かったこともあり、読みやすいコードより動くコードを優先した物となっていた。よって、スペースプローブコンテスト後は、メインプログラム v1 のコードリファクタリングや適切なクラス分け、モジュール化、コメントの追加などを行い極力読みやすいコードの作成を行った。また、メインプログラム v1 で問題となっていた、プログラム自体が Raspberry Pi Zero と実際に使用するモジュールなど、特定のハードウェアを使用しないと動かないという問題解決を目指した。具体的には Raspberry Pi Zero で使用する特定のシステムファイルやモジュールを仮想化し、それらのアクセス時には仮想化されたモジュールを返すことで、特定のハードウェアの上でしかプログラムが動かないという制約から開放されることを目指した。

知識面としてはスペースプローブコンテスト前には取れなかった、技術習得のための学習期間を設けることができた。それにより、Docker, Docker-compose, FastAPI, Git, GitHub, Python を中心とした技術に対する理解度が制御班全体で向上した。さらに、メインプログラム v2 の作成を行っていたこともありそれら技術を使用したことにより技術習得へとつながった。

（※文責: 西殿大輝）

5.1.4　軌道解析結果

節 3.2 で示した、笹流ダムでの実験においてはダムの貯水槽に対して正面方向と横方向から投下の様子を撮影し、投下の軌道、スピード、位置について解析を行った。解析にはフリーソフトの Kinovea を使用。解析における手順については節 3.1.1 を参照のこと。

図 5.2　笹流ダム貯水槽に対し正面方向

図 5.3　笹流ダム貯水槽に対し横方向

解析には実験 2 日目の 3 回目の投下の際に撮影した動画を使用した。

2 日目の 3 回目（全体で 8 回目）

以下に示すのは軌道解析結果のグラフである。図 5.4 は高さの推移のグラフであり、図 5.5 はスピードの推移のグラフである。

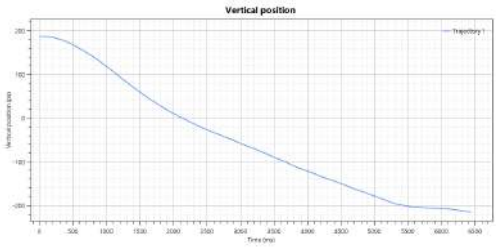

図 5.4　高さの推移

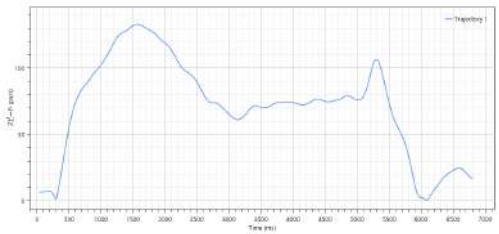

図 5.5　スピードの推移

まず、図 5.4 のグラフについて解説する。縦軸を高さ (px)、横軸を時間 (ms) とした。なお、貯水槽の下から撮影したため、やや角度をつけて撮影せざるを得なかった。これにより縦軸の高さにマイナスの値が現れてしまっている。

グラフの概形から、パラフォイルが正常に機能したことにより緩やかに落下していることが分かる。

次に、図 5.5 のグラフについて解説する。縦軸をスピード (px/s) 、横軸を時間 (ms) とした。投下後から 1500ms にかけて機体のスピードが増加し、1500ms から 2700ms にかけて機体のスピードが減少し、そこからは比較的安定したスピードを継続していることが分かる。また、5300ms 付近で再び機体のスピードが増加しており、着地へと向かっていることが分かる。

このグラフから、投下後 1500ms 付近でパラフォイルが安定したことによりスピードが安定し、5000ms 付近にかけて飛行、5300ms 付近では進行方向を修正する目的でパラフォイルの紐の長さを調整したことでパラフォイルの傾きが変わり安定状態から崩れたため若干の速度上昇がみられたと分析できる。

機体の軌道を解析した結果、グラフからもパラフォイルが正常に機能していたこと、制御が正常になされていたことが確認できた。

（※文責: 佐藤大地）

5.2　発表会

この項では中間発表会および成果発表会の概要や結果とその評価を行う。

（※文責: 佐藤大地）

5.2.1　中間発表

概要

2021 年 7 月 9 日金曜日に行われ、形式としては事前に制作していた Web サイトやポスターを評価者が確認し、その後 ZOOM にて各プロジェクトのブレイクアウトルーム内で発表や質疑応答が行われるという流れであった。

本プロジェクトでは 10 分程度のプレゼンテーションを行い、5 分程度の質疑応答を行った。

結果と評価

当日の質問内容としては、必要知識の学習方法やプロジェクトの進め方、今後の予定など、発表の補足的な内容が多く、技術的な内容については少ない印象を受けた。質疑応答時間が少なかったため、踏み込んだ質問をすることが憚られた可能性がある。

また、有効な評価アンケートの結果は 39 件であった。詳細を以下に示す。

まず、発表技術についての評価グラフを図に示す。評価点の平均は 7.8 であった。また、好意的なコメントとして「発表内容が詳細で充実していた」「スムーズだった」といった意見が多く見受けられた。一方で、批判的なコメントとして「発表時間が長い」「質疑応答の時間が短い」などという意見も目立った。

最終発表会では質疑応答の時間がとれるようバランスを意識して発表を構成していく必要がある。

図 5.6　発表技術についての評価グラフ

次に、発表内容についての評価グラフを図に示す。評価点の平均は 8.3 であった。

また、好意的なコメントとして「わかりやすく説明できていた」「発表の流れがよかった」といった意見が見受けられた。批判的なコメントとしては「スケジュールがざっくりしている」「課題をどのように解決するのか、いつまでに解決するのか」といった意見があった。

これらのコメントから、課題の洗い出しと解決方法を再検討し、スケジュールを再確認する必要があると感じた。

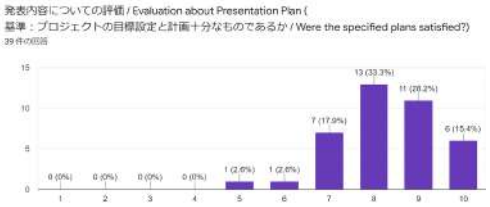

図 5.7　発表内容についての評価グラフ

（※文責: 佐藤大地）

5.2.2　成果発表会

概要

2021 年 12 月 10 日金曜日に行われ、形式としては事前に制作していた Web サイトやポスターを評価者が確認し、その後 ZOOM にて各プロジェクトのブレイクアウトルーム内で発表や質疑応答が行われるという流れだった。

本プロジェクトでは事前に収録した動画を用いて 10 分程度のプレゼンテーションを行い、5 分程度の質疑応答を行った。

結果と評価

当日は A グループへの質問が少なかったように感じられた。発表にて機体や制御方法について説明する内容が多かったため、実際の実験動画を載せられなかったことが影響していると考えられる。質問の一例として衝撃吸収機構についての質問があった。100m からの落下に耐えうる機構ということで評価者の印象に残ったのだろうと考えられる。

また、有効な評価アンケートの結果は 37 件であり、内訳は学生が 83.8%、教員が 16.2% であった。

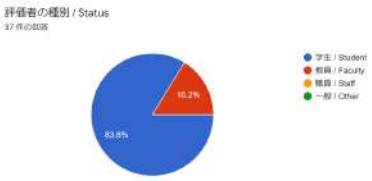

図 5.8　評価者の種別グラフ

以下は発表技術についての評価である。平均点は 7.8 点であった。

発表技術についてのコメントとして「発表時の時間分けがしっかりしていた。機械音声・動画を用いることで時間分けが安定しそうで良い」や、「機械音声は何を言ってるのかは分かりやすく、人間の声は流れが自然に聞こえるのでどちらも良い点があると思う」など、音声へのコメントが多く見られた他、急なトラブルに対する対応についても良い評価を頂いた。一方で「機械音声が不快だった」ことや「発表者の通知音 (discord、Slack?) が入っており少し不快だった」コメントがあり、聴講者への配慮に欠ける場面もあったと推察される。

なお、中間発表時のアンケートで得られた発表技術に対する評価の平均点は 7.8 点だった。

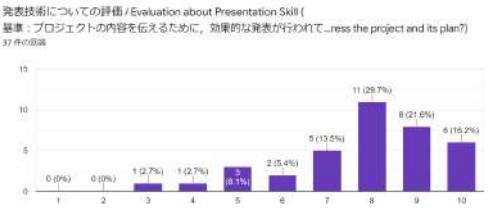

図 5.9　発表技術についての評価グラフ

以下は発表内容についての評価である。平均点は 8.5 点であった。

発表内容についてのコメントとして「ハードウェア系の成果物になるので、動画を用いて実際にどうなったかを説明したり、実際の画像を見ることができて良かった」ことや「動画を再生してからの質疑応答という段取りで質問の前に改めて質問の整理ができたのでよかった」など、動画に関する高評価なコメントが多く見られた他、多くの高評価のコメントもいただくことが出来た。一方で「コンテストの説明に大きな比重を置くよりはプロジェクト実行途上での検討経過や困ったことなども含めてもう少し丁寧に時間をかけても良かった」といったような検討段階のことに対する説明が甘かったことが見て取れるコメントも存在した。

なお、中間発表時のアンケートで得られた発表内容に対する評価の平均点は 8.3 点だった。

中間発表から比べて発表内容に対する評価の平均点が上がっていることが確認できた。中間発表とは違い、成果物を具体的に動画で示せたことが評価の向上につながったと考えられる。

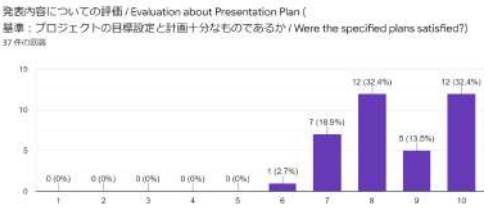

図 5.10　発表内容についての評価グラフ

（※文責: 富樫幹生）

5.3　実験一覧

この項では本グループが実施した実験について述べる。なお、スペースプローブコンテストについても実施した実験に含むものとする。

本グループが実施した実験の一覧を図 5.11 に示す。

日付	実験場所	天気	風	参加人数	目的	結果
2021/6/2	未来大体育館	-	-	2人	パラフォイルとパラシュートの機能比較	パラフォイルを採用することに決定
2021/6/9	未来大体育館	-	-	2人	簡易的なパラフォイルの機能検証	改良が必要だということが判明
2021/6/11	未来大体育館	-	-	2人	パラフォイルの進行方向制御方法の比較	羽の開閉は不向き、紐の長さでの制御が有効
2021/6/23	未来大体育館	-	-	2人	試作パラフォイルの機能検証	パラフォイルとしての機能は確認
2021/7/30	笹流ダム		南南東 5m/s	5人	試作パラフォイル二号機の機能検証	全く機能せず、根本からの調査と改良が必要だということが判明
2021/8/18~2021/8/20	未来大体育館	-	-	2人	複数の試作パラフォイルの機能比較	空気セルを設けたパラフォイルが飛行時の安定性に優れることが判明
2021/9/2~2021/9/6	未来大体育館	-	-	2人	本番用パラフォイルの機能検証	飛行時の安定性と無制御時に前方に進行方向が固定されることを確認
2021/9/7~2021/9/9	未来大体育館	-	-	2人	本番用パラフォイルの旋回動作の検証	紐を引くことによる旋回動作の安定性と旋回の程度を確認
2021/9/10	未来大体育館	-	-	2人	パラシュートによるパラフォイルの展開補助の検証	パラシュートによる展開補助の有意性を確認
2021/10/2	北海道赤平市	曇り	不明	0人（オンラインでの開催）	スペースプローブコンテスト投下審査	パラシュートが展開せず
2021/11/6	笹流ダム	晴れ	無風	6人	制御システムの検証と屋外でのパラフォイルの機能検証	2日目へ
2021/11/7	笹流ダム	晴れ	南西 4m/s	4人	同上	パラフォイルと制御システムが正常に動作していることを確認
2021/11/24	未来大体育館	-	-	2人	パラフォイルの収納状態におけるパラシュートの展開補助の検証	収納状態からのパラシュートの展開補助の有意性を確認

図 5.11　実施実験一覧

（※文責: 富樫幹生）

第 6 章　おわりに

この章ではプロジェクトのまとめについて述べる。

（※文責: 佐藤大地）

6.1　プロジェクトにおける個人の役割

この項では各メンバーが自分に割り当てられた課題についての結果とその評価を記述する。

（※文責: 佐藤大地）

6.1.1　佐藤大地

A グループのリーダーとして活動した。グループを代表して担当の先生やワーキンググループ、国土交通省への航空法の確認の連絡や活動の申請を行ったほか、グループ全体への情報共有、グループのタスクの進捗の管理、期末提出物の管理や作成を行った。

前期の機体班での活動としては、試作段階のパラフォイルの作製、パラフォイルの機能を確認する実験や降下軌道に関する実験、実験に伴う学内利用の申請、機体内部に取り付ける部品の選定や、部品に伴う機構を 3D プリンターで作製を行い、後期の機体班での活動としては機体本体の作製、衝撃吸収機構の性能評価に関する実験、Kinovea での軌道分析と評価を行った。

（※文責: 佐藤大地）

6.1.2　富樫幹生

グループ A で機体班に属し、パラフォイルや展開用パラシュートの設計、作製を主に取り組んだ。具体的には、パラフォイルの構造をインターネットや図書館等の本を借り同メンバーと意見を出しながら考え、その情報を基に設計図を書き、パラフォイルに使用するのに適した素材や紐を変更しながら作製に励んだ。また、パラフォイルのプロトタイプを複数個用意し、体育館や笹流ダムでの投下実験し、その様子を手持ちのカメラ等で撮影した。その結果をもとに試行錯誤を重ね大会用に修正し完成品となる展開用パラシュート付きのパラフォイルを完成させた。

（※文責: 富樫幹生）

6.1.3　鈴木進太郎

プロジェクト全体で利用するツールの管理、導入補助を行う情報システムという役割を担当した。 プロジェクトやグループにおける、Google Drive, Slack, GitHub, notion などの使用方法の共有、使用する際の規則の制定を行った。また上記以外の新たに使用するツールにおいても、テストや導入支援を担当した。A グループでは制御班に属し、電子回路の設計・作製、使用するデバイ

スの選定、制御班におけるタスクの追加や割当、進捗状況の管理などを行った。その他、制御班のメンバーが作業に集中できるように雑務なども担当していた。

（※文責: 鈴木進太郎）

6.1.4 大岩穂峻

前期の活動では A グループの制御班として電装を担当した。PIC マイコンの動作確認やモジュールの実験を行い、機体構成の検討に貢献した。前期終了時に機体班に移動し、パラフォイル・パラシュートの作製を担当した。

パラフォイル・パラシュートの作製では設計、作製から進捗管理までを主な作業として行った。投下実験の際には投下と記録を担当し、パラフォイルの投下と実験資料作成を行った。また、報告書の管理を担当し構成の構築や割り当てを行った。

（※文責: 大岩穂峻）

6.1.5 西殿大輝

A グループの制御班として PIC マイコンやサーボモータ用の基盤作製及びアルゴリズムの考案を担当し、ハードウェア・ソフトウェアのそれぞれからグループに貢献した。回路設計・作製を行い、PIC マイコンの動作確認ができる環境を整えた。また、テスト環境を用いた実験に加えて、簡易的なパラフォイルを作製して実験を行い、アルゴリズムの考案・実装を行った。加えて、ダム実験の結果及び大会結果をもとにアルゴリズムの改良を行った。

（※文責: 西殿大輝）

6.1.6 杉山宏輝

前期では A グループの制御班として PIC マイコンやサーボモータの動作確認を行い、PWM 制御や各モジュールの使用方法などについて調べまとめることで技術的な面から本グループに貢献した。また、大岩と共に中間報告書の作成を担当し、各章の詳細の記述を行った。また、システム全体の実装を行い、コンテストに向けて機体が動作するよう調整した。後期にはコンテストや実験の結果をもとにプログラムの修正を行い、その後コードリファクタリングを行うことで可読性とメンテンナンス性の向上に努めた。

（※文責: 杉山宏輝）

6.2　今後の課題と展望

この項では本グループにおける今後の課題と展望について述べる。

（※文責: 富樫幹生）

6.2.1　機体

パラシュートとパラフォイルの課題について述べる。パラシュート、パラフォイルは展開精度が不安定であり、また、風の影響や紐の本数が多い影響から、パラフォイルの紐が絡まって捻じれる現象が起きてしまい飛行が安定せず、機体本体が垂直に落下してしまうことがあった。原因として、展開用パラシュートの構造が横向きの風に弱い作りであったことや、ドローンなどから切り離す際の展開機構が紐で固定したような簡単な使用になっていたことが問題であった。

今後の展望として、以前まではパラフォイルに簡易的な展開機構を採用していたが、展開の確実性を高めるため、落下検知した際に自動でパラフォイルが発射されるような機構を作製する予定である。そして、パラシュートの構造で採用しているセル型パラフォイルの形状を変えることによって飛行中の安定性を高め、気候変化に対応できるような設計作りや機体が落下した際の吸収機構の完成度も高めていきたいと考えている。

（※文責: 富樫幹生）

6.2.2　制御

次に、制御の課題について述べる。課題として挙げられる項目として、姿勢制御と誘導制御である。姿勢制御では、実験段階の時点では重心等の座標移動の動作は確認できたが、落下検知が上手く作動せず判定できないことが多く見受けれた。また、パラフォイルの展開がうまく行かないことが原因で Z 軸方向の座標取得が予想しているものとかけ離れてしまった。そして、誘導制御では軌道解析を行い目的地点に機体が誘導されているか検証したところ落下検知作動した場合のみ左、右への方向転換が確認できた。

まず姿勢制御での問題は、新たに風圧計を導入することによって風などの気候変化に対応できるようにする。また、機械の傾きを検知するためのジャイロセンサも精度の高いものへ変更する予定である。次に誘導制御では、位置座標の変化推移を調査できるように SD カードへの書き込みを行うほか、都度取り外ししなくても情報を取得できるようなリアルタイムの通信を実現を検討中である。加えて、誘導の際に重要である位置座標のほかに高度を習得するための気圧センサも追加で搭載する予定である。

（※文責: 富樫幹生）

6.2.3 大会

大会についての結果や課題を挙げていく。初めに、大会結果についてだが本チームは銀賞を獲得し全体順位 4 位であった。内容としては、 ドローンから釣り上げられたひもが うまく離れず パラシュートが 展開しない形で垂直落下してしまった。 しかし、 目標地点であるパイロンの近くに落ちたため着地時点でのポイントは高かった。 また、 衝撃吸収機構がうまく作用したおかげで、機体が自由落下したにもかかわらず、機体内部の RaspberryPi や電源、センサ類は壊れることなく無事であった。大会後の実験の際に、動作するかどうかを確認したところ、大会前と同様の動きを示していた。

次に課題であるが、展開用パラシュートが開かないという点が一回きりの本番では勝負を決定づける肝であるということが分かった。ゆえに、展開精度を大幅に向上させる必要がある。そして、本来なら落下した際に加速度センサを用いて、落下検知が行われるはずだったが、大会後の機体の行動履歴を確認したところ収得できておらず検知が正しく行われていなかったことも明らかになった。今後はこの課題をクリアし、来年の同大会に出場するかチームメンバーと検討中である。

（※文責: 富樫幹生）

付録 A　プログラムリスト

ソースコード A.1　main.py

```
1  import csv
2  from datetime import datetime
3  import time
4  import pytz
5  import os
6  import math
7  import sys
8
9  from util.GPS import GPS
10 from util.csvWriter import csvWriter
11 from util.LSM9DS1 import simple
12 from util.Controller import Controller
13
14 def fall_detection(gps, lsm, threshold_acc_range, threshold_count):
15     count = 0
16     while True:
17         #data = gps.get_data()
18
19         acc_x, acc_y, acc_z = lsm.read_acc()
20         # my_csv.write([data['date'][0], data['date'][1], data['date'][2],
21         # data['timestamp'][0], data['timestamp'][1], data['timestamp'][2],
22         # data['latitude'][0], data['latitude'][1], data['longitude'][0],
               data['longitude'][1],
23         # acc_x, acc_y, acc_z, 0, 0])
24
25         if(threshold_acc_range[0] < acc_z and acc_z < threshold_acc_range[1]):
26             count += 1
27         else:
28             count = 0
29
30         # 落下している
31         if(count >= threshold_count):
32             break
33
34         time.sleep(0.05)
35
36 def main():
37     con = Controller.Controller()
38     my_csv = csvWriter.MakeCSV()
39     lsm = simple.LSM9DS1()
40     gps = GPS.GPS(9,'dd')
41     gps.start()
```

```
42
43      my_csv.write(gps.get_data_format())
44      # 落下検知
45      fall_detection(gps, lsm, [-5000, 5000], 15)
46      data = gps.get_data()
47      gps_x, gps_y = gps.get_gps_coordinate()
48
49      my_csv.write(gps.get_data())
50      old_x, old_y = gps_x, gps_y
51
52      #print('while')
53      system_count = 0
54      while True:
55          data = gps.get_data()
56          gps_x, gps_y = gps.get_gps_coordinate()
57          dir_x, dir_y = con.direction(gps_x, gps_y)
58          LR, let = con.dis_gps(old_x, old_y, gps_x, gps_y, dir_x, dir_y)
59          my_csv.write(gps.get_data())
60          old_x, old_y = gps_x, gps_y
61          time.sleep(0.5)
62
63          system_count += 1
64          if(system_count > 60):
65              break
66
67      sys.exit()
68
69
70  if __name__ == '__main__':
71      main()
```

ソースコード A.2　csvWriter.py

```
1     import csv
2
3   class csvWriter:
4       def __init__(self, dirPath, fileName):
5           self.path = dirPath+fileName
6           if not(os.path.isdir(dirPath)):
7               os.makedirs(dirPath)
8           f = open(self.path, 'w')
9           f.close()
10
11      def write(self, data):
12          with open(self.path, 'w') as f:
13              writer = csv.writer(f,lineterminator='\n')
14              writer.writerow(data)
```

ソースコード A.3　GPS.py

```
import serial
import time
import threading

from . import micropyGPS

class GPS(threading.Thread):
    lock = threading.Lock()

    def __init__(self, local_offset=0, location_formatting='ddm'):
        super(GPS, self).__init__(daemon = True)
        self.port = serial.Serial('/dev/serial0', 9600, timeout=0)
        self.mpy_gps = micropyGPS.MicropyGPS(local_offset, location_formatting)

    def run(self):
        while True:
            time.sleep(0.1)
            try:
                sentence = self.port.readline().decode('utf-8')
                if sentence.startswith('$GPRMC'):
                    for x in sentence:
                        self.mpy_gps.update(x)
            except Exception:
                pass

    def thread_acquire(self):
        GPS.lock.acquire()

    def thread_release(self):
        GPS.lock.release()

    def get_gps_coordinate(self):
        GPS.lock.acquire()
        data = self.mpy_gps.latitude, self.mpy_gps.longitude
        GPS.lock.release()
        return data

    def get_data(self) -> list:
        GPS.lock.acquire()
        t = []
        key = ['date', 'timestamp', 'latitude', 'longitude']
        t.append(self.mpy_gps.date)
        t.append(self.mpy_gps.timestamp)
        t.append(self.mpy_gps.latitude)
        t.append(self.mpy_gps.longitude)
        _data = dict(zip(key, t))
        GPS.lock.release()
```

```
48
49          data = [_data['date'][2], _data['date'][1], _data['date'][0],
50                  _data['timestamp'][0], _data['timestamp'][1], _data['timestamp
                        '][2],
51                  _data['latitude'][0], _data['latitude'][1], _data['longitude
                        '][0], _data['longitude'][1]]
52          return data
53
54      def get_data_format(self) -> list:
55          data = ['year', 'month', 'day', 'hours', 'minutes', 'seconds', '
                latitude', 'direction', 'longitude', 'direction']
56          return data
```

ソースコード A.4　simple.py

```
1  import sys, os
2  import time
3
4  #sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))
5  from lsm9ds1_rjg import Driver, I2CTransport, SPITransport
6
7
8  class LSM9DS1:
9      """This example shows how to poll the sensor for new data.
10     It queries the sensor to discover when the accelerometer/gyro
11     has new data and then reads all the sensors."""
12     def __init__(self):
13         #self.driver = self._create_spi_driver()
14         self.driver = self._create_i2c_driver()
15         self.driver.configure()
16
17     @staticmethod
18     def _create_i2c_driver() -> Driver:
19         return Driver(
20             I2CTransport(1, I2CTransport.I2C_AG_ADDRESS),
21             I2CTransport(1, I2CTransport.I2C_MAG_ADDRESS))
22
23     @staticmethod
24     def _create_spi_driver() -> Driver:
25         return Driver(
26             SPITransport(0, False),
27             SPITransport(1, True))
28
29     def main(self):
30         try:
31             count = 0
32             while count < 2000:
33                 ag_data_ready = self.driver.read_ag_status().
                        accelerometer_data_available
```

```
34                 if ag_data_ready:
35                     self.read_ag()
36                     count += 1
37                     time.sleep(0.05)
38         finally:
39             self.driver.close()
40 
41     def read_gyro(self):
42         try:
43             ag_data_ready = self.driver.read_ag_status().
                   accelerometer_data_availabe
44             if ag_data_ready:
45                 temp, acc, gyro = self.driver.read_ag_data()
46                 return gyro
47             else:
48                 return None
49         except Exception:
50             pass
51 
52     def read_acc(self):
53         temp, acc, gyro = self.driver.read_ag_data()
54         return acc
55 
56 
57     def read_ag(self):
58         temp, acc, gyro = self.driver.read_ag_data()
59         print('Temp:{} Acc:{} Gryo:{}'.format(temp, acc, gyro))
60 
61     def read_magnetometer(self):
62         mag = self.driver.read_magnetometer()
63         print('Mag {}'.format(mag))
64 
65 
66 if __name__ == '__main__':
67     LSM9DS1().main()
```

ソースコード A.5　servo.py

```
1  import time
2  import RPi.GPIO as GPIO
3  
4  class MG996R:
5      def __init__(self, pin):
6          #GPIO.setwornings(False)
7          GPIO.setmode(GPIO.BCM)
8          GPIO.setup(pin, GPIO.OUT)
9          self.pwm = GPIO.PWM(pin, 50) #50kHz
10         self.pwm.start(0)
11 
```

```
12      def __del__(self):
13          self.pwm.stop()
14          GPIO.cleanup()
15
16      def write(self, angle): #0 ~ 180
17          if angle < 0:
18              angle = 0
19          elif angle > 180:
20              angle = 180
21          dc = angle * 9.5 / 180.0 + 2.5
22          self.pwm.ChangeDutyCycle(dc)
23
24  class LG20MG:
25      def __init__(self, pin):
26          #GPIO.setwornings(False)
27          GPIO.setmode(GPIO.BCM)
28          GPIO.setup(pin, GPIO.OUT)
29          self.pwm = GPIO.PWM(pin, 50) #50kHz
30          self.pwm.start(0)
31
32      def __del__(self):
33          self.pwm.stop()
34          GPIO.cleanup()
35
36      def write(self, angle): #0 ~ 180
37          if angle < 0:
38              angle = 0
39          elif angle > 180:
40              angle = 180
41          dc = angle * 10 / 180.0 + 2.5
42          self.pwm.ChangeDutyCycle(dc)
43
44  if __name__ == '__main__':
45      l_servo = MG996R(18)
46      r_servo = LG20MG(12)
47      time.sleep(1)
48      i = 0
49      while i < 3:
50          l_servo.write(0)
51          r_servo.write(0)
52          time.sleep(1)
53          l_servo.write(90)
54          r_servo.write(90)
55          time.sleep(1)
56          l_servo.write(180)
57          r_servo.write(180)
58          time.sleep(1)
59          l_servo.write(90)
```

```
60          r_servo.write(90)
61          time.sleep(1)
62          i = i + 1
63      print('fin')
```

ソースコード A.6　Controller.py

```
1  import math
2
3  from . import servo
4  from ..LSM9DS1 import simple
5
6  class Controller():
7      def __init__(self):
8          self.position = [
9              [41.833705, 140.770666]
10             [41.833705, 140.770666]
11             [41.833705, 140.770666]
12             [41.833705, 140.770666]
13         ]
14
15         self.LEFT = 0
16         self.RIGHT = 1
17
18         self.meter_per_latitude = 111.111 * 1000 # 111.111km/degree
19         self.meter_per_longitude = 0
20
21         self.lsm9ds1 = simple.LSM9DS1()
22         self.servo = [servo.MG996R(12),servo.MG996R(18)]
23         self.servo[self.LEFT].write(0)
24         self.servo[self.RIGHT].write(0)
25
26         '''
27         self.l_servo = servo.MG996R(12)
28         self.r_servo = servo.MG996R(18)
29         self.l_servo.write(0)
30         self.r_servo.write(0)
31         '''
32
33         self.goal_x = 0
34         self.goal_y = 0
35
36     # ローカル座標でx 軸方向に進んだとき、ワールド座標で x,
           y 軸のどちらの方向に進むのかを判定
37     # def (self, gps_x, gps_y):
38     # dis_x = gps_x - self.goal_x
39     # dis_y = gps_y - self.goal_y
40     # if(dis_y >= 0 and dis_x >= 0) or (dis_y < 0 and dis_x < 0):
41     # return 0
```

```
42      # else:
43      # return 1
44
45      # 現在地から最も近いターゲットの緯度,経度を返す
46      def get_near_coordinate(self, gps_x, gps_y):
47          distance = float('inf')
48          tmp = 0
49          i = 0
50          selector = 0
51          for position_x, position_y in self.position:
52              tmp = math.pow(abs(position_x - gps_x) + abs(position_y - gps_y))
53              if tmp < distance:
54                  distance = tmp
55                  selector = i
56              i+=1
57
58          return self.position[selector][0], self.position[selector][1]
59
60      # 座標 2つの角度を取る
61      def get_angle(self, old_x, old_y, gps_x, gps_y):
62          return math.degrees(math.atan2(old_y - gps_y, old_x - gps_x))
63
64      # 加速度値から姿勢角 (roll, pitch)を取る
65      # http://watako-lab.com/2019/02/15/3axis_acc/
66      def get_attitude(self):
67          ax, ay, az = self.lsm9ds1.read_acc()
68          roll = math.atan2(ay,az)
69          pitch = math.atan2(-ax, math.sqrt(ay*ay + az*az))
70          return roll, pitch
71
72      # roll, pitch, 地磁気値から方位角を取る
73      # https://myenigma.hatenablog.com/entry/2016/04/10/211919
74      def get_azimuth(self):
75          # read_magnetometer is not return values, so needs fix
76          mx, my, mz = self.lsm9ds1.read_magnetometer()
77          roll, pitch = self.get_attitude()
78          sin_r = math.sin(roll)
79          sin_p = math.sin(pitch)
80          cos_r = math.sin(roll)
81          cos_p = math.sin(pitch)
82          my = mz*sin_r - my*cos_r
83          mx = mx*cos_p + my*sin_p*sin_r + mz*sin_p*cos_r
84          return math.degrees(math.atan2(my, mx))
85
86      def set_meter_per_longitude(self, latitude):
87          self.meter_per_longitude = math.cos(latitude)
88
89      def latdiff_to_meter(self, diff):
```

```
90          return self.meter_per_latitude * diff
91
92      def londiff_to_meter(self, diff):
93          return self.meter_per_longitude * diff
94
95      def get_goal_distance(self, gps_x, gps_y):
96          londiff = self.londiff_to_meter(abs(self.goal_x - gps_x))
97          latdiff = self.latdiff_to_meter(abs(self.goal_y - gps_y))
98          return math.sqrt(math.pow(londiff,2) + math.pow(latdiff,2))
99
100     def select_servo(self, let):
101         if let > 0:
102             return self.LEFT
103         else:
104             return self.RIGHT
105
106     def dis_gps(self, gps_x, gps_y):
107         goal_distance = self.get_goal_distance(gps_x, gps_y)
108         azimuth = self.get_azimuth()
109         # 基準軸を北にする
110         goal_angle = self.get_angle(gps_x, gps_y, self.goal_x, self.goal_y) -
                90
111         let = goal_angle - azimuth
112
113         selector = self.select_servo(let)
114
115         if goal_distance <= 1:
116             self.servo[selector].write(180)
117             fin = True
118         else:
119             self.servo[selector].write(let)
120
121     '''
122     # 進むべき方向に合わせてモータを動かす。戻り値は
            gps_LR の判定結果と修正すべき進路角度
123     def dis_gps(self, old_x, old_y, gps_x, gps_y, dir_x, dir_y):
124         dis_goal_x = abs(dir_x - gps_x)
125         dis_goal_y = abs(dir_y - gps_y)
126         asin_now = self.get_angle(old_x, old_y, gps_x, gps_y)
127         asin_goal = self.get_angle(gps_x, gps_y, dir_y, dir_x)
128         let = asin_goal - asin_now
129         abs_let = abs(let)
130         LR = self.gps_LR(gps_x, gps_y, dir_x, dir_y)
131         if dis_goal_x + dis_goal_y <= 0.002:
132             self.l_servo.write(180)
133             fin = True
134         elif LR == 0 and let > 0:
135             self.l_servo.write(abs_let)
```

```
136             #print('hidari')
137         elif LR == 0 and abs_let < 0.1:
138             pass
139             #print('iji')
140         elif LR == 0 and let < 0:
141             self.r_servo.write(180)
142             #print('migi')
143         elif LR == 1 and abs_let < 0.1:
144             pass
145             #print('iji')
146         elif LR ==1 and let > 0:
147             self.r_servo.write(180)
148         elif LR == 1 and let < 0:
149             self.l_servo.write(abs_let)
150             #print('hidari')
151         return LR, let
152         '''
153 
154     def set_goal_coordinate(self, x, y):
155         self.goal_x = x
156         self.goal_y = y
```

参考文献

[1] 経済産業省 製造産業局宇宙産業室. “宇宙産業の現状と課題について”. https://www8.cao.go.jp/space/comittee/27-sangyou/sangyou-dai3/siryou2.pdf, (参照 2022-01-14)

[2] 中須賀, 松永. “CanSat 計画　ー日米大学による手作り小型衛星への挑戦ー”. https://www.jstage.jst.go.jp/article/kjsass/48/562/48_589/_pdf/-char/ja, (参照 2022-01-14)

[3] 東京大学 中須賀研究室 NASU チーム. “東京大学 ARLISS2012 報告書”. http://www.unisec.jp/history/arliss2012/report/NASU_rep.pdf, (参照 2022-01-14)

[4] 植松電機株式会社. “スペースプローブコンテスト 2021”. https://spc.uematsudenki.com/wp/, (参照 2022-01-14)

[5] 九州工業大学スペースダイナミクス研究室 KINGS 脇田 遼. ”ARLISS2010 報告書”. http://www.unisec.jp/history/arliss2010/report/kyutech-kings_rep.pdf, (参照 2022-01-14)

[6] FN の高校物理. “パラフォイル凧”. http://fnorio.com/0001festival_physics/1parafoil_kite/parafoil_kite.htm, (参照 2022-01-14)

[7] 堂平スカイパークパラグライダースクール. “パラペディア”. https://dd-skypark.com/12_parapedia/, (参照 2022-01-14)

[8] ふうせん宇宙撮影. “宇宙開発で使用できる超本格パラシュートの設計図を公開します！”. http://fusenucyu.com/?p=5611, (参照 2022-01-14)

[9] AUTODESK. “Fusion 360”. https://www.autodesk.co.jp/products/fusion-360/overview, (参照 2022-01-14)

[10] JoanCharmant. “kinovea”. https://www.kinovea.org/, (参照 2022-01-14)

[11] NRI. “ナレッジマネジメント”. https://www.nri.com/jp/knowledge/glossary/lst/na/knowledge, (参照 2022-01-14)

[12] 植松電機株式会社. “スペースプローブコンテスト 2021 結果発表”. https://spc.uematsudenki.com/wp/スペースプローブ 2021　結果発表/, (参照 2022-01-14)