公立はこだて未来大学 2020 年度 システム情報科学実習
グループ報告書

Future University-Hakodate 2020 Systems 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

Space Development - Autonomous movement robot Flight Project

グループ名

A グループ

Group Name

A Group

プロジェクト番号/Project No.

17-A

プロジェクトリーダ/Project Leader

比留川満洋　Mitsuhiro Hirukawa

グループリーダ/Group Leader

森宗誠太　Seita Morisou

グループメンバ/Group Member

森宗誠太　Seita Morisou
松村海斗　Kaito Matsumura
長谷川沙織　Saori Hasegawa
松尾威斗　Taketo Matsuo
山崎颯太　Souta Yamazaki

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

2021 年 1 月 14 日

Date of Submission

January 14, 2021

概要

近年、ある期間内で制作物やサービスを開発し、実験を行って結果を評価するというプロジェクト学習形式は工学系の学部などでよく行われている。そのようなプロジェクト型学習に利用できるもののひとつとして CanSat がある。CanSat とは、Can(缶) Satellite(人工衛星)の省略形であり、缶の形をした超小型模擬人工衛星のことを指す。CanSat では実際の衛星開発 でも直面することの多い問題を解決する必要が出てくる。

本グループでは、一般的な 350ml 缶サイズの CanSat を運用してきた。本グループでは、CanSat の着地後、走行の軌道によって図形を描くというミッションを掲げた。5 月から 8 月にかけて、リスクマネジメント、サクセスクライテリア、材料、およびセンサ類などについて議論した。また、10 月の活動から機体、基板、そしてパラシュート制作の実施を開始した。11 月は、笹流ダムでの実験を計 3 回行った。1 回目と 2 回目は、パラシュートのみの実験であったが、重量の調整をすることで想定通りの速度で落下に成功した。3 回目は、落下させた後、自動走行まで想定していたが、落下場所の精度が良くなく、まっすぐ走行することが困難であったためミッションは達成できなかった。12 月は、学校で走行のみの実験を行った。結果は、加速度センサーが動かなかったため、目的地に向かうことができなかった。ゴール付近で走行実験を行うと、想定していたゴール範囲には入り、停止したため部分的に成功した。

新型コロナウイルス感染症 (COVID-19) の影響により、大学へ登校が制限されていたためプロジェクト学習の主な活動をオンラインで行った。本グループの成果として、CanSat のミッションは、成功といえる物ではなかった。しかし、プログラム、構造系、電装系、そして飛行物系の担当者はそれぞれの運用方法や知識を得ることができた。

キーワード　CanSat, ミッション, サクセスクライテリア, i-CanSat

（※文責: 松村海斗）

Abstract

In recent years, project-based learning, in which students develop products and services within a certain period of time, conduct experiments, and evaluate the results, has become a common practice in engineering departments. CanSat is an abbreviation of "Can (can) Satellite" and refers to a micro-simulated satellite in the shape of a can. In CanSat, it is necessary to solve problems that are often faced in actual satellite development.

In this group, we have been operating a CanSat that is the size of a common 350ml can. From May to August, we discussed about risk management, success criteria, materials, and sensors. From May to August, we discussed risk management, success criteria, materials, sensors, etc. In October, we started to fabricate the airframe, substrate, and parachute, and in November, we conducted three experiments at Sasaryu Dam. The first and second experiments were only with the parachute, but we succeeded in dropping it at the expected speed by adjusting the weight. In December, we conducted a running experiment only at school. In December, we conducted a run-only experiment at the school. The result was that the accelerometer did not work, so we could not go to the destination. In December, we conducted a driving experiment at the school.

Due to the effects of the new coronavirus infection (COVID-19), posting to the university was restricted, so the main activities of the project study were done online. As an outcome of this group, the CanSat mission was not a success. However, the people in charge of the program, structure, electronics, and flight systems were able to gain knowledge and methods of operation.

Keyword CanSat, Missions, Success Criteria, i-CanSat

（※文責: 松村海斗）

目次

第 1 章　はじめに

1.1　背景

CanSat は、1998 年スタンフォード大学宇宙開発研究所の Bob Twiggs 教授が缶で衛星を作ろうと提唱したことが始まりである [1]。現在では、ARLISS（A Rocket Launch for International Student Satellites）のようなイベントが実施されていることにより積極的にロケットの打ち上げや小型衛星の制作が行われている。ARLISS とは、アメリカネバダ州のブラックロック砂漠にてアマチュアロケットグループ (AeroPac) 協力のもと大気圏内への CanSat の打ち上げ実験のことである。学生が宇宙開発技術の基礎研究を競う競技でもある [2]。

近年、ある期間内で何かを開発し、実験を行って結果を評価するという学習形式は工学系の学部などでよく行われている。CanSat は、プロジェクト型学習ツールに利用することができ、より実際の衛星開発でも直面することの多い問題を解決する必要が出てくる。こういった特徴があるため多くの大学では、CanSat の開発を行っている。

（※文責: 松村海斗）

1.2　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、航空宇宙工学に関連した、回路設計制作技術、飛行制御技術、無線通信技術、プログラミング、設計のための理論、構築に必要な技術、運用の経験、そしてプロジェクトそのものの進め方などを学習することである。また、プロジェクト全体の目標として、あらかじめ決めてある目標地点へ到達できる、姿勢制御と自律移動アルゴリズムを取り入れた機体の制作としている。

（※文責: 松村海斗）

1.3　課題

本プロジェクトの課題として、新型コロナウイルス感染症 (COVID-19) によって対面での作業を行う機会が制限されているため、遅れを取り戻すように計画する必要がある。また、他大学のレポートで落下時に内蔵物が壊れる事例が多かったため、衝撃対策を重点的にする必要がある。

（※文責: 松村海斗）

第 2 章　プロジェクト学習の概要

2.1　問題の設定

本グループでは、過去の資料をもとにグループで設計・制作・運用を行う際に、成功を妨げる問題を以下のように設定した。

1. パラシュート展開時にうまく開かず、ほとんど自由落下状態に近くなること
2. パラシュートの展開に成功しても、基板等が壊れること
3. 走行時小さい障害物に引っかかること
4. CanSat 本体が木の上に引っかかること
5. 基板のようなハード面が衝撃により壊れること

（※文責: 松村海斗）

2.2　具体的な手順・課題設定

A グループでは、第 2 章第 1 節で述べた問題を解決するための課題を以下のように設定した。

1. パラシュート展開実験を素材や畳み方など変えるなど試作品を多く作ること
2. 衝撃緩和材を変えて落下実験をすること
3. タイヤ展開させ半径を大きくすることにより障害物を乗り越えやすくすること
4. 9 軸センサーを使い最低限の空中制御をすること
5. クラッシャブルゾーンを設けることにより基板やセンサーを保護すること

クラッシャブルゾーンとは、電車や車などに使用されている技術であえて壊れやすい部分を作ることにより衝撃を吸収し、壊れてほしくないを守る部分のことである。

（※文責: 松村海斗）

第 3 章　活動内容

この章では、A グループの活動の概要、目標、課題およびその解決方法についての説明をしていく。

（※文責: 山崎颯太）

3.1　概要

A グループのミッションは、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。今回のミッションの目標や課題およびその解決方法に関する内容は、以下の通りである。

（※文責: 山崎颯太）

3.1.1　到達レベル

到達レベルとは、結果に対してどれくらいの成功であったかを段階に分けて評価するものである。A グループの到達レベルは 4 段階に分けられており、成功基準の低いほうからミニマムサクセス (成功基準の 60 %)、ミドルサクセス (成功基準の 80 %)、フルサクセス (成功基準の 100 %)、アドバンスドサクセス (成功基準の 120 %) となっており、前期の段階では次の通りである。

ミニマムサクセス

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とても基礎的な目標である。A グループでは、「構造物を破損させずに着地させる」という目標を設定した。

ミドルサクセス

CanSat がミッションを達成できる状態で着地できた場合の最低ラインの目標である。A グループでは、「パラシュートを分離した後に走行し、ゴール地点に到達する」という目標を設定した。

フルサクセス

CanSat を用いてミッションを達成することを目指す本来の目標である。A グループでは、「簡単な図形を描いた後、ゴール地点に到達する」という目標を設定した。

アドバンスドサクセス

目標が達成可能であると判断した場合など、より高度な達成目標である。A グループでは、「複雑な図形を描いた後、ゴール地点に到達する」という目標を設定した。

後期からは、作業時間及び基準が不明瞭であるという理由から目標の簡易化・詳細化を行い、最終的な到達レベルを次の通りに設定した。

ミニマムサクセス

前期と大きく変わらず、「構造物を破損させずに着地させる」という目標を設定した。

ミドルサクセス

「パラシュートを分離した後に走行し、ゴール地点に到達する」というのは変わらず、ゴールしたかどうかの判断基準を設定し、「ゴールに指定した地点から半径 10m」をゴール条件に設定した。なお、この目標については、最終実験直前に範囲を狭くできると判断したため「ゴールに設定した地点から半径 5m」に変更を行った。

フルサクセス

「描いた軌跡を画像認識で線として出力」という絵の前段階でもある図形を描くことを主な目標として設定した。

アドバンスドサクセス

「描いた軌跡と元絵の一致率が 70 %」と設定し、図形を描くという工程において、それが成功であるか失敗であるかの判断基準を明確なものにするための目標設定とした。

（※文責: 山崎颯太）

3.1.2　課題の割り当て

A グループの目標であるフルサクセスの簡単な図形を描いた後に、ゴール地点に到達するためにはいくつかの課題がある。

最初の課題が、着地時の衝撃である。着地したときの衝撃によって、機体が破損した場合はその後のミッション進行が困難になるだけではなく最悪の場合、ミッションの進行をすること自体が不可能になってしまうためである。この課題をクリアするための対策としてあげられるものは、機体を入れるコンテナを 2 重構造にする、クラッシャブルゾーンを設けることや緩衝材を使用するという方法があげられる。

次の課題として走行性がある。走行性が悪い場合、着陸後のミッションである図形を描く工程に支障が出るだけでなく、そもそも障害物などによって走行自体ができなくなる可能性がある。その対策として、タイヤを大きくするなどの障害物を越えやすくする工夫が求められる。

最後の課題として、飛行時の姿勢に関する課題がある。空中での姿勢が安定していない場合、飛行時の起動が安定しないだけでなくパラシュートのラインが絡まるなど飛行自体が困難になる可能性があるためである。ラインについては、第 3 章第 3 節第 1 項で説明する。この課題に対処するために、機体を正方形にすることやパラシュートの紐を長くするなどの姿勢を安定させる方法や、紐をチェーンのような絡まりにくい構造のものを利用することなどがあげられる。

（※文責: 山崎颯太）

3.2　課題解決のプロセス

この節では、A グループの課題解決を行うための活動内容を説明する。

（※文責: 松尾威斗）

3.2.1　前期スケジュール (作業内容)

これまで行ってきた活動、及び後期の作業予定について説明する。なお、5 月の 20 日、22 日、27 日はグループに分かれずに作業を行った。

5 月

グループ内の役職をきめ、グループのメインミッションについて議論を行った。それに伴うサブミッションを、センサや実現性などの観点からいくつか仮案として決定した。

6 月上旬

メインミッションの達成する方法について 3 つほど案を出した。それらについて、センサや実現性などから実験を行いつつ決めていくことに決定した。また制作にあたり、リスクマネジメントについて現状で考えられる範囲で検討、共有を行った。特に衝撃対策について深く議論を行い、いくつか実験を行っていくことで決定した。 制作にあたって過去の CanSat の大会のルールに則るために、缶サイズの決定や大まかなミッション実施の流れを決定した。

6 月中旬

CanSat に関することでそれぞれが興味のある分野について勉強し、それを共有した。CanSat は空中から地上へ着陸し活動するという流れがあり、その中の空中に関することについて議論し、CanSat を空中へ運ぶ方法としてドローンで降下させることに決定した。また、降下の方法としてパラシュートを用いるのか、またはグライダーを用いるのかなどを議論した。中間発表会に向け、ポスター制作と Web 制作の役割分担を決定した。

サクセスクライテリアとして、今まで議論してきたサブミッションとメインミッションをミニマムサクセス、ミドルサクセス、フルサクセス、アドバンスドサクセスに分類した。先月行った目標に対する課題の解決策を具体的に議論を行い、それをもとにそれぞれのメンバーが構想案を提案し共有した。また、それらの構想案から良い部分を取り出したひとつの構想案を決定した。それに伴った材料の検討も行い始めた。

6 月下旬

材料の検討とポスター、Web 制作を主に行った。材料の検討では、どの程度の距離感でどのようなデータをやり取りするかをもとにセンサ類を決め、電源やモータなどもこの段階で決定した。ポスター、Web 制作については、B グループとも情報を共有し、前例を参考にしながら制作を行った。

7 月上旬

先月決定した構想案をより深めるために、設計図の制作を行った。これに伴い、A グループの CanSat の特徴を伝えるために、サクセスクライテリアと問題対策の一部分をピックアップし、ポスターと Web に掲載した。中間発表が始まるにあたり、必要な役割分担を行った。また、グループ名を Art CanSat(ACS) と決定した。中間発表会で行われる質疑応答に対し、予めいくつか予想できる質問に対し返答の準備を行った。これらをもとにリハーサルを行い、中間発表会を行った。

7 月中旬

中間発表会で得られた様々な評価や意見をもとに、改善点を見出した。なかでも、地面に絵を描けたという成功基準をどう定めるのかという意見を受けて、今後、描く絵についての検討と、その成功基準を定めるとした。また、時間的な厳しさなど物理的な問題点も指摘があったため、スケジュールの見直しを行った。

7 月下旬

少人数でホームセンターに行き、材料の検討を行った。また、その情報をグループ内で共有し、現状で必要な材料を、ホームセンターで購入するものとインターネット上で購入するものに分けて決定した。その後、B グループと同日にホームセンターに行き、決定した材料を購入した。夏期休暇に向けて、メンバーそれぞれに担当を設けて作業に取り組み、随時進捗状況を共有するよう決定した。また、メンバー間でのタスクの偏りが生じた場合や、タスクの取り組みが困難である場合は、その時折に臨機応変に対応していくこととした。

（※文責: 松尾威斗）

3.2.2　後期スケジュール (作業内容)

中間発表会以後におこなった作業について説明する。

8 月

8 月の活動では、主に中間報告書の作成にあたった。TeX を用いて作成し、週をまたぎながら確認と修正を繰り返した。これらの作業と並行し、物理的に機体の制作に取り掛かる準備を始めた。機体部分としては 3D モデリングソフトウェアの Fusion360 を実際に入れ、学習するとともに 3D モデルの設計を行った。電子部品と基板については、機体のサイズとの兼ね合いを考えつつ回路図を設計していった。また、適時ホームセンターやネットショッピングなどで必要なパーツを入手していった。

9 月

9 月の活動では、夏期休業もあり活動日自体も少なく、オンライン上での会議が数回のみであった。その中で、実験を行える日数が少ないということが判明し、スケジュールの見直しを行った。

10 月

10 月の活動では、工房での作業が主であった。機体としては、3D モデルの印刷に想定より時間がかかったため、日を分けて印刷を行った。基板については、はんだ付けを行っていく中で、物理的に隙間に余裕のない部分や、電力的に電池が足りないなどから設計の変更を行った。また、基板の設計変更から機体側もデータから設計を作り直し、印刷も 10 月中に再び行った。これらと並行してパラシュートの作成も行った。

11 月

11 月の活動では、実験を主に行った。実験は笹流ダムで行った。1 回目の実験では、パラシュートの落下実験を行った。実験の結果としては、想定通りの速度で落下し成功であった。2 回目も落下実験のみであったが、落下物の重量を調整し実験を行い、これも想定通りの速度で落下し成功であった。3 回目の実験は最終実験であり、ここでは落下とそこから CanSat が自動走行するという内容であった。結果は、落下は成功であったが落下場所の精度が良くなく、走行できる場所に降りられなかった。また、走行自体も砂利道の上ではまっすぐ走行するのが困難であり、設定していたミッションは達成できなかった。

12 月

12 月の活動では、まず最終実験でうまくできなかった走行のみの実験を大学で行った。結果は、加速度センサがうまく働かなかったため、目的地へ向かうことができなかった。しかし、ゴール付近から走行実験を行うと、想定していたゴール地点から半径 5m 以内の範囲に入ったときに停止し、部分的には成功した。これらの結果をもとにプロジェクト学習成果発表会の準備として Web 制作を行い、発表会を行った。発表会終了後は、報告書と学習ポートフォリオ、学習フィードバックの作成にあたった。

1 月

1 月の活動では、報告書の作成と最終確認、提出を行った。

（※文責: 松尾威斗）

3.3 作業詳細

機体の制作を飛行物、構造系、電装系、プログラム系、通信系に分けて作業を行った。この節では各分野で行った作業について説明をしていく。

（※文責: 松村海斗）

3.3.1 飛行物

本項では、CanSat のパラシュートについて説明する。

パラシュートに要求される条件として、以下が挙げられる。

1. 落下速度が速すぎるために、CanSat の機体本体と基板が損傷しないようにすること
2. 落下速度が遅すぎるために、風に流されないようにすること
3. 木やダムの中段に乗らないように左右揺れを小さくすること
4. 落下開始直後パラシュートが開ききること
5. 落下後、機体がうまく走り出せるようにすること

これらの条件を満たすためにパラシュートは以下のようになった (図 3.1)。

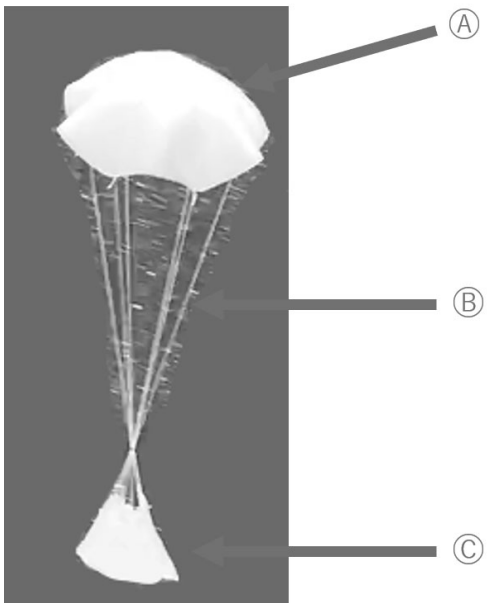

図 3.1　パラシュート

A. キャノピーについて説明する。キャノピーとは、パラシュートの傘のような分である。この部分は、様々な先行論文などを確認した結果、六角形が一番向いていると考えたため六角形にした。また、材料については、強度などの観点からナイロン生地を使用した。

B. 次に、ラインについて説明する。ラインとは、パラシュートのキャノピーと機体を収納する部分をつなぐものである。この部分の材質は、当初釣り糸のようなものを使用する予定であったが、絡まった時の対処などのメンテナンス面や視認性の高さなどを再検討した結果、縫合糸を使用することになった。しかし、実験中、糸が細く視認しづらく実験をスムーズに行うことができなかったため、最終的にタコ糸でラインを制作した。しかし、タコ糸を使用しても絡まったため、今後実験を行うことがあればプラスチックチェーンのような絡まりにくい素材に変更する予定である。

C. 機体を収納する場所について説明する。元々、機体にテグス糸をエナメル線の抵抗熱で切る予定であった。しかし、制作段階でテグス糸を切断できるほど抵抗熱まで上昇しきらなかったため、機体を入れる袋を作り落下させるように変更した。こうすることにより糸を切り離す必要がなくなった。

ここからキャノピー、ラインのサイズ決定の仕方を説明する。

キャノピーの面積は、以下の運動方程式に当てはめて計算を行った。質量 m、重力加速度 g、空気密度 ρ、空気抵抗係数 Cd、面積 S とする。

$$v = \sqrt{\frac{2mg}{\rho \times Cd \times S}}$$

この式に $m = 0.50(kg)$、$g = 9.80665 \approx 9.81(m/s^2)$、$\rho = 1.225 \approx 1.22(kg/m^3)$、$v = 4.00(m/s)$、$Cd = 0.60$ を代入すると、$S = 0.8376(m^2)$ となった。速度 v は、グループメンバーと相談して決定した。
求めた面積 $S = 0,8376(m^2)$ を正六角形の 1 辺の長さ r を求めるため以下の式に代入した。

$$S = 6 \times sin60 \times r^2$$

この式より 1 辺の長さ r = 40.5 (cm) となる。またラインの長さ L は、直径 R = 81(cm) の 1.4 倍にした。L = 80 × 1.4 =112 (cm) となった。

ここからは実験の解析データについて説明する。

実験後、Kinovea[3] という無料ソフトウェアを使い、CanSat の落下軌道から垂直位置を割り出しグラフに表した。図 3.2 を以下に示す。図 3.2 の縦軸は、垂直位置 m、横軸は経過時間 ms を表す。落下実験を実施したダムの高さは 25.3m であったが、解析する上で CanSat 機体を動画上でうまく視認できるとまで動画を経過させなければならなかったため、縦軸の最大値は 22.5m となった。また、横軸の最大値は 5400ms となった。

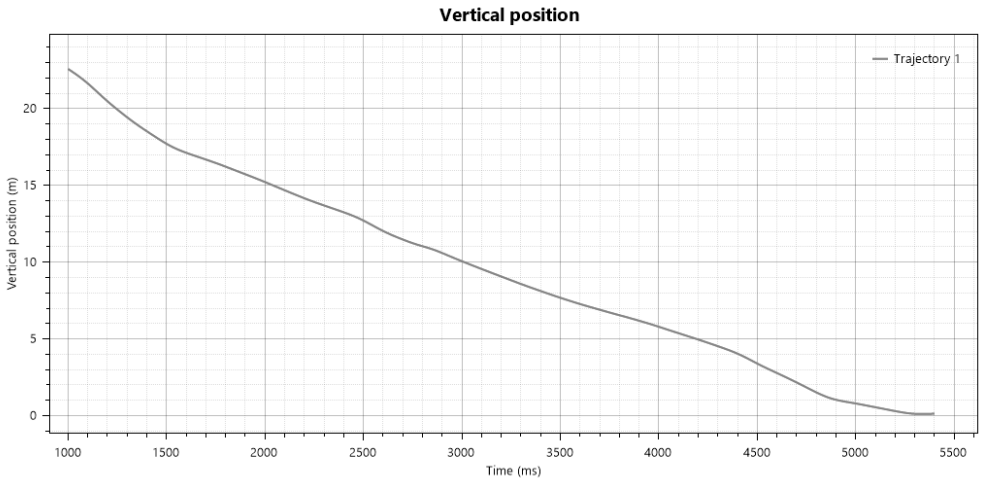

図 3.2　垂直位置

CanSat の落下軌道から落下速度を割り出しグラフに表した。図 3.3 を以下に示す。

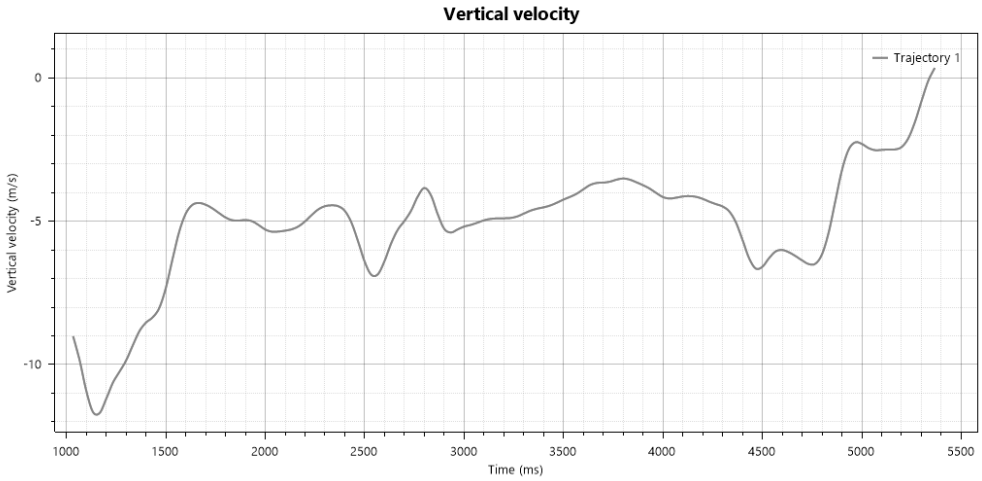

図 3.3　落下速度

図 3.3 の縦軸は、落下速度 m/s、縦軸は経過時間 ms を表す。グラフから横軸がおよそ 1700ms から落下速度が横ばいになっているので 1700ms あたりでパラシュートが開いたと推測する。また、キャノピー作成時に落下速度を v=4.0m/s に設定し、解析後の落下速度のグラフも 4.0m/s から 5.0m/s の間に収まっていた。

ここから無料ソフトウェア Kinovea を使用して、今回上記で行った解析方法を解説していく。https://www.kinovea.org/download.html から Kinovea 0.8.27 をダウンロードし、使用した。Kinovea を起動すると図 3.4 の画面が表示される。

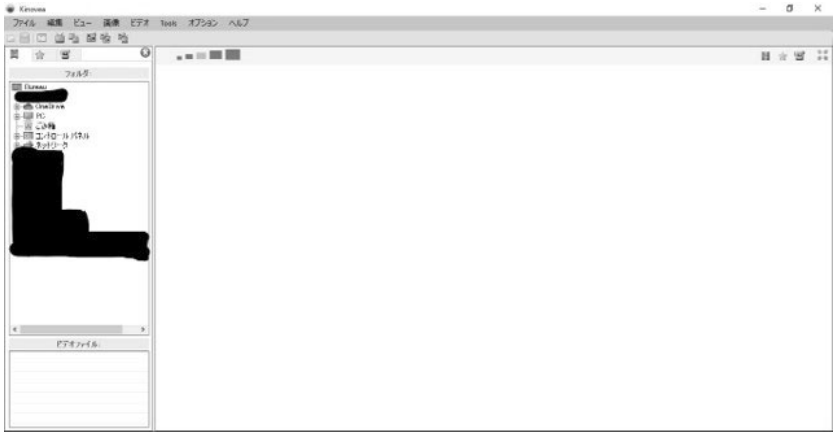

図 3.4　起動後画面

まず、図 3.4 の左側にあるフォルダーから解析したい動画を選択する。図に使用した動画は、実際に解析した実験動画である。動画選択後の画面は、図 3.5 のようになる。

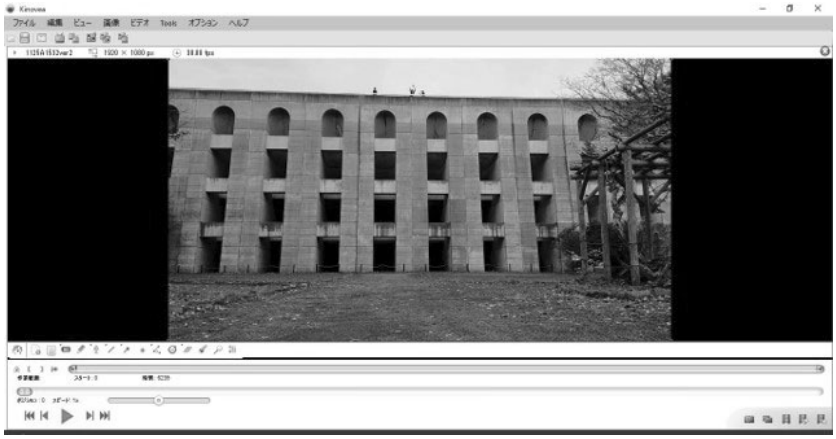

図 3.5　動画選択後画面

図 3.6 を以下に示す。

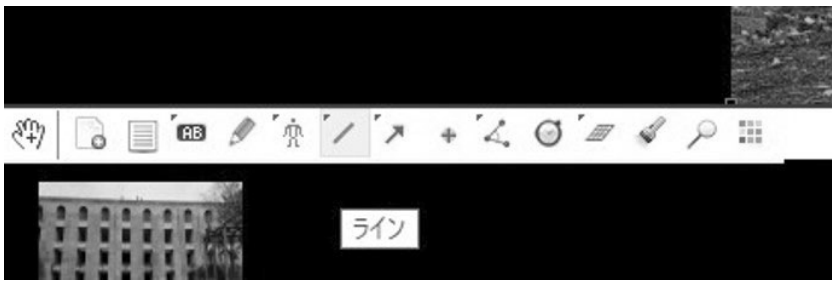

図 3.6　動画選択画面

図 3.6 にある通りのラインを選択する。選択後、基準とするものの端から端まで線を引く。今回は、ダムの上から下まで線を引いた。線を引いた後、線上で右クリックし、較正を選択する。較正選択後の画面を図 3.7 を以下に示す。また、選択した物の長さを入力する。今回は笹流ダムの高さは 25.3m なので、25.3m を入力した。

図 3.7　較正選択後画面

次に、対象物を設定する。まず、先ほどの動画の上で右クリックし、軌道追跡を選択する。軌道追跡を選択すると図 3.8 のようになる。図 3.8 の線の上で右クリックし、設定を選択する。設定を

図 3.8　軌道追跡選択後画面

選択後の画面、図 3.9 を以下に示す。

図 3.9　設定選択後画面

図 3.9 の時に Object Window と Search Window のサイズを適当に入力し、適用を左クリックする。解析する対象物を決定するため Object Window、対象物からの解析範囲を限定するために Search Window のサイズを決める。これらの設定をすることにより解析がよりスムーズになる。以上で、解析する前に必要な設定を終了した。Object Window と Search Window に入力後の画面、図 3.10 を以下に示す。

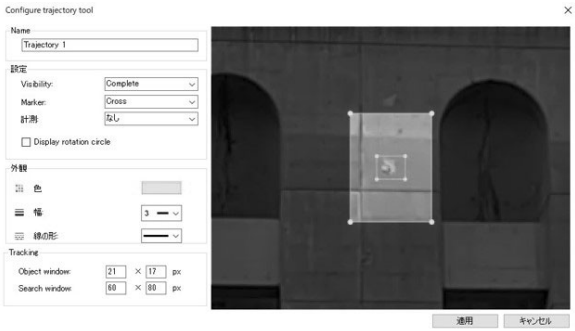

図 3.10　Window サイズ入力後画面

画面左下の再生ボタンを押すと、自動で解析される。再生をしている最中に図 3.11 のようにずれることがある。これを図 3.12 のようにカーソルで元に戻す必要がある。対象物が動かなくなると、マーカの上で右クリックし、軌道の最終地点を確定を選択する。ここで解析は終了である。Tools をクリックし、Linear Kinematic を選択すると、解析結果をグラフで見ることができる。また、選択後データの種類を変えることができる。Kinovea での解析方法の説明は以上である。

図 3.11　ずれの例

図 3.12　ずれ修正例

（※文責: 松村海斗）

3.3.2　構造系

A グループの CanSat の構造系について詳しく記述する。最終的になぜこの形になったのか、それまでに至った理由やプロセスなどを時系列に沿って解説する。

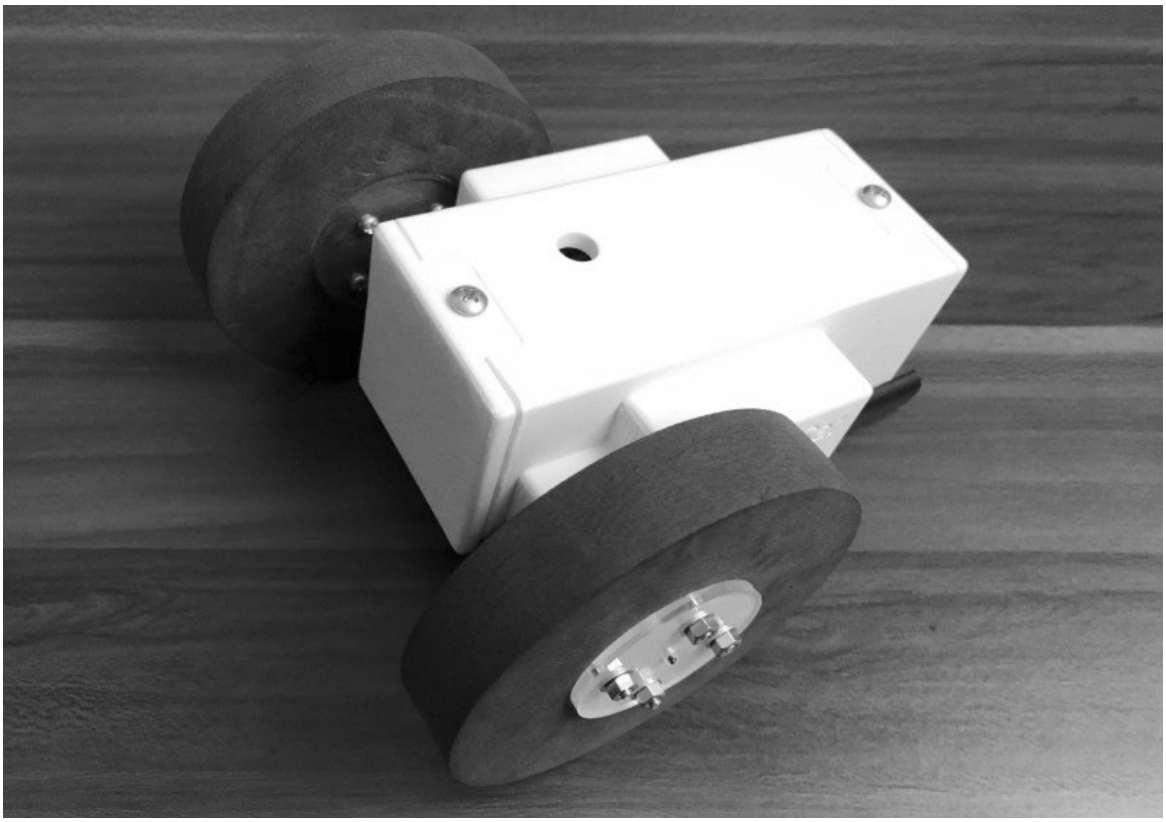

図 3.13　地上走行機体完成版

初期構想

今回のプロジェクト学習は新型コロナウイルス感染症 (COVID-19) の影響によって対面で会議をする時間や試行錯誤しながら制作する時間、実際に大学で作業を行う時間などが大きく制限され、更には作業に入る段階までもが遅い中での取り組みとなった。そのため、A グループでは限られた中で従来の CanSat から大きく離れた形の構造に挑戦していくということはせず、シンプルにしっかりとミッションをこなせる構造にするという結論になった。そこで、過去の CanSat の制作物をいくつか見る中で、主流であった二輪走行で進めていくこととなった。

CanSat に与える機能

この段階では、まだ実際に材料やどのように制作していくかは考慮せず、どのような機能を設ければミッションの成功により近づきやすくなるのかを話し合った。その中で、提案されたものの中でいくつかピックアップしたものを CanSat の構想図とした。それが、次の画像である (図 3.14)。

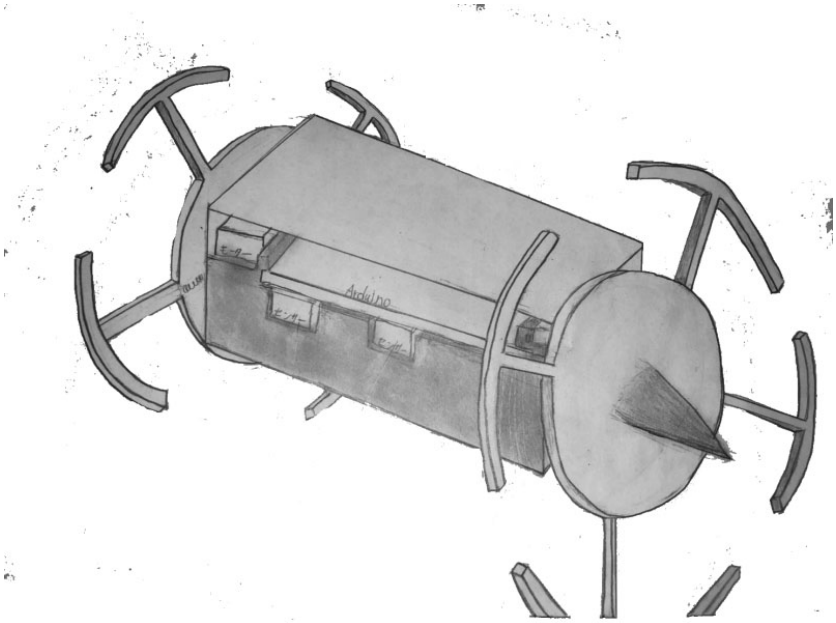

図 3.14　構想図

この構想図について解説する。

はじめに、タイヤについて解説する。タイヤは機体と地面との距離がある程度開いたほうが、小石や砂利などで引っかかることもなくなり走行性が上がるのではないかと考え、少し大きめのものをつけようと考えた。しかし、CanSat にはサイズの規定があり、A グループが想定していた 350ml サイズのルールではタイヤを大きくすることは困難であった。そこで、タイヤを展開式にすることで打ち上げから、パラシュート展開、地面に着陸までの間でそのルールに反することなく走行性を上げられると考えた。

次に機体について解説する。過去の CanSat の例をいくつか見てみると基板が露出した状態で走行しているものが多く見られた。しかし A グループでは、耐久性や基板をしっかりと守りたいという意見から、基板を露出することはせず基板全体を覆う箱状のものにしようという意見にまとまった。ここで、出てくる問題が基板のメンテナンスについてである。基板を箱の中に固定しようとすると、何らかのエラーで CanSat が動かなくなった場合、原因を特定できず 1 から作り直しというようになりかねない。そこで、基板をスライドしながら入れられるようにすればよいという提案があり、これは CanSat のサイズの関係上、基板が 2 枚になってしまうという点とも都合がよいということもあり構想図で取り入れた。

最後に左右の突起について解説する。これは、二輪走行の CanSat が空中からの落下時や走行中に坂道などで真横に直立してしまうという状況を考えたときの対策である。先ほどで述べたようにタイヤを大きくするという構造にしたため、真横に直立してしまう確率も高くなる。そこで横に突起状のものをつけることで、そもそも直立するという状況が起こり得ないようにし、この問題を解決しようと考えた。以上が構想図時点での、構造の特徴である。

モデリング 1

実際に設計を始めるにあたり、必要なサイズの情報が存在する。サイズと言っても様々で、モータやバッテリー、基板の縦横高さなど設計をする上でとても重要である。しかし、基板を制作で機器の検討に時間がかかり、実際に基板と合わせたときの高さなどが出ない状況が続いた。そのため、設計を始めるのが遅めとなった。

A グループでは、ABS 樹脂を素材に 3D プリンターで作成すると決めた。3D プリンターで物を印刷するためのデータを作成するために、3DCAD ソフトウェアである Fusion360 を使用した。そのはじめの、設計データが次の画像である (図 3.15)。

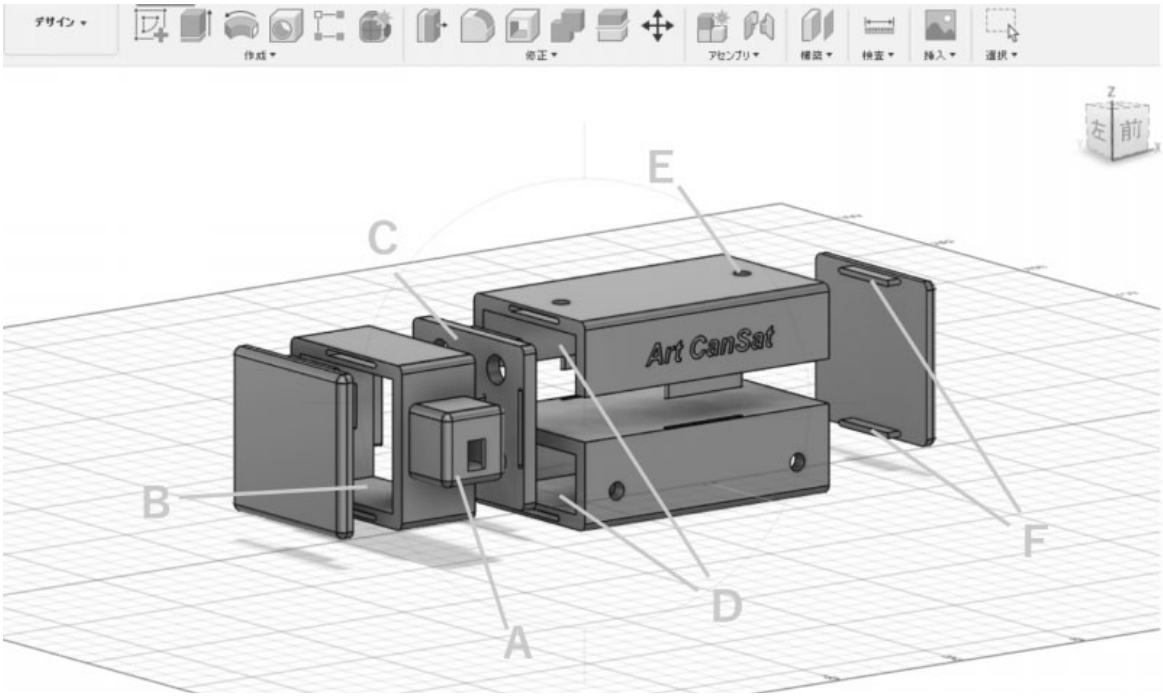

図 3.15　初号機設計データ

まず、構想図と大きく異なる点は機体が横長ではなく縦長になったことである。なぜ縦長になったかの経緯を説明する。二輪で走行するためにはもう一点地面と接触している部分が必要である。それがないと、何らかのセンサで地面と機体とを並行に保たない限りタイヤが空回りする、あるいは機体が回ってしまうという現象が起きる。しかし、それに挑戦するには、あまりに時間が足りないと考えたためタイヤ以外のもう一点を地面に接触させることを選択した。その上でもう一つの地面との接触部分を ABS 樹脂だけで作成してしまうと軽すぎるため、機体を支える点として十分な働きができないと考えた。しかし、ABS 樹脂以外におもりを乗せようとすると CanSat 全体としての重量が上がってしまい、ルールの規定内に収まらないという問題が発生するためこの案も適当とは言えなかった。そこで、機体を縦長にすることで重心をタイヤより後方にずらし機体の尾を地面との接触点として機能するようにした。しかしこれにも衝撃の問題が発生する。そこで地面との接触部分にクッション材を用いた緩衝パーツを取り付けることで、衝撃の問題を軽減した。

次にアルファベットに沿って解説していく。

A. ここはモータを入れる部分になりタイヤと機体との距離を少し離すため、突出した形状になっている。タイヤと機体を離す理由は、ホイール部分をねじで止めるため、そのネジが機体と擦れてしまうことを避けるためである。

B. ここは、バッテリーを収納する場所である。この段階では、バッテリー 1 個で CanSat を動かす予定であったため、モータの設置に当たり空いた隙間をバッテリーサイズに調整した。

C. バッテリー部屋と D の部屋を仕切るセパレータの役割を持つ壁である。ただの壁だけだと各種接続ができないため、線を通す穴を開けている。

D. 基板を収納する部屋である。上下 2 つに分かれており、2 枚の基板を収納する。また、構造図の段階であったスライド式の収納方式をここで取り入れている。

E. 外部と通信を行うためのアンテナを出すための穴である。

F. 各パーツを接続するためのジョイント部分である。

以上が初号機のデータの解説である。

初号機制作

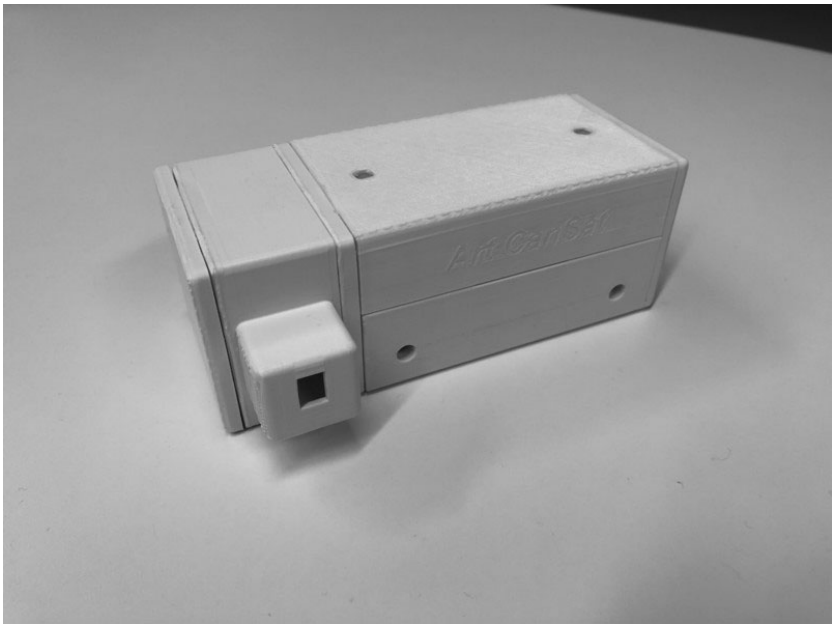

図 3.16　初号機

時間はかかったが 3D モデルを実際に印刷した。しかし、様々な問題が生まれてきた。まず、ジョイント部分である。データではピッタリのサイズで作成したのだが、積層型の 3D プリンターであったこともあり穴のサイズがデータよりも縮んでしまった。そのため、凸の方のパーツを紙やすりで削りぴったりになるように調整したが、1 個の凸パーツが折れてしまった。さらにこの段階で、基板の高さが足りなくなり 2 枚構造のところを薄く重ねて一部屋で管理することとなった。その結果、先述の D 部分の上下を仕切る壁を取り除き一部屋にした。更にここで問題が発生し、積層型の 3D プリンターはモデルの中身が詰まっているわけではないため、切り口が壁にならずパラパラと中身が出てしまうという問題が発生した。さらには、バッテリーがこの段階で 2 つ必要であると要請があり、2 つ目のバッテリーを乗せるスペースは存在しなかった。これらの問題が重なり初号機という名前の通り、この機体は没となった。なお、問題が発生した箇所の画像を下に載せた。(図 3.17)(図 3.18)

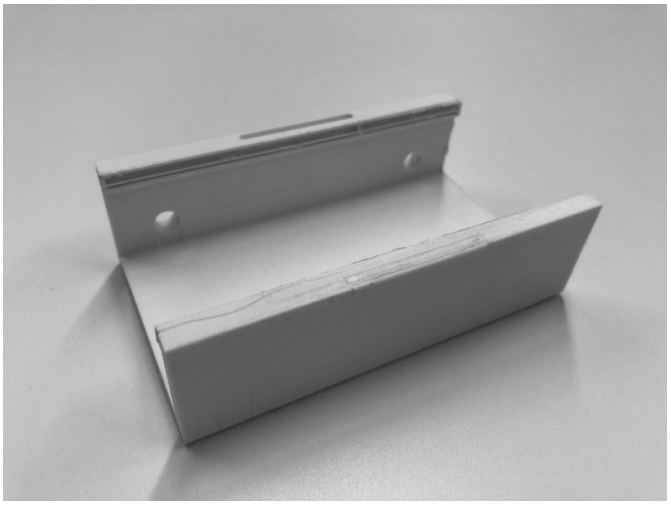

図 3.17　初号機問題箇所 1

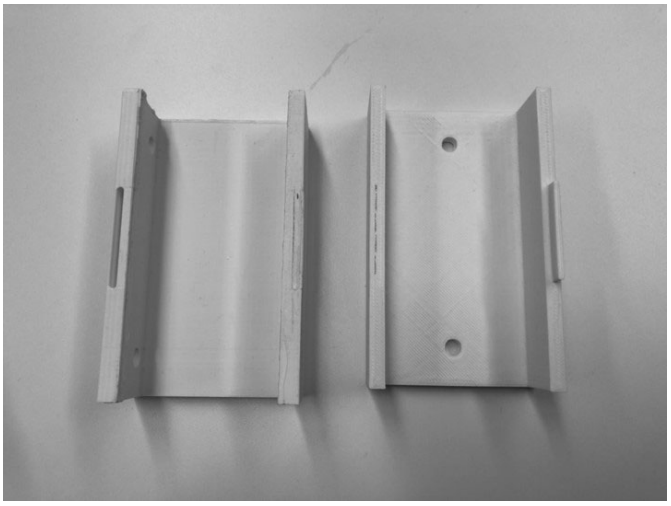

図 3.18　初号機問題箇所 2

初号機で発生した問題や追加項目なども踏まえて再度データの作成を行った。このデータが最終制作物のデータであるため、各種機能解説は後述の構造解説で行う。

モデリング 2

初号機で発生した問題や追加項目なども踏まえて再度データの作成を行った。このデータが最終制作物のデータであるため、各種機能解説は後述の構造解説で行う。

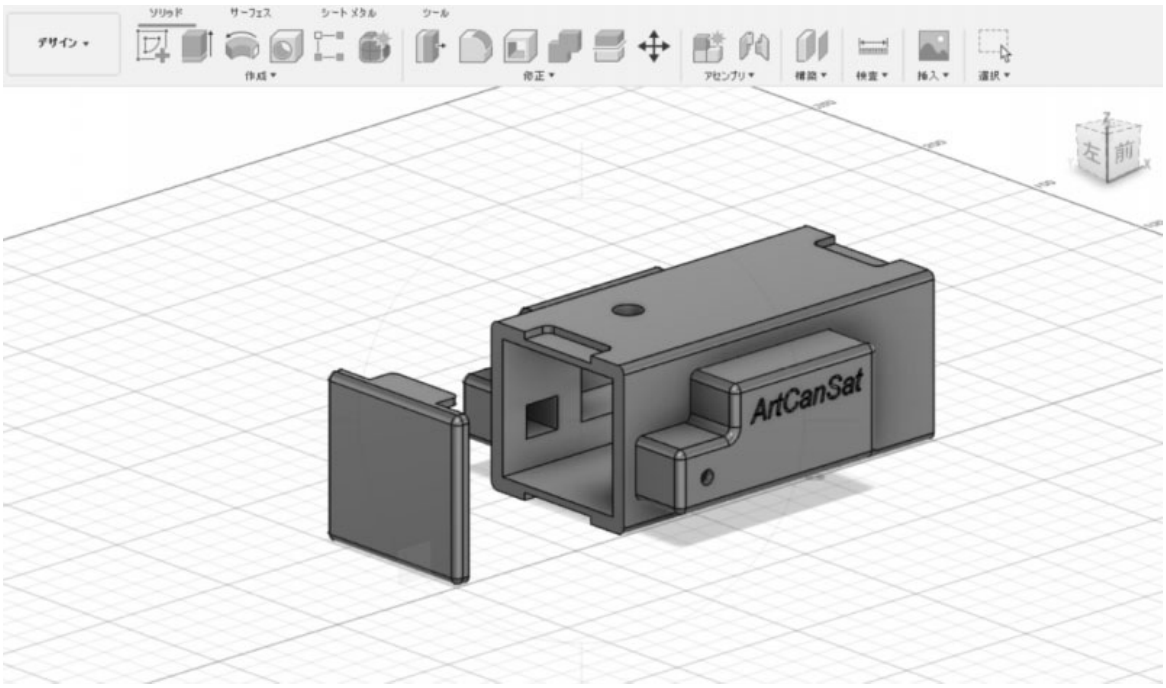

図 3.19　最終機体設計データ

前回と比べての改善点としては、まずジョイント部分を凸と穴にするのではなく、壁の半分のサイズの板を上下に合わせる形にした。このようにすることで、印刷後の紙やすりでの調整がしやすくなる。加えてジョイントの両方の状況がわかるのでネジ止めが容易となった。

次に「モデリング 1」での C 部分である。バッテリーを壁際に埋め込む形となり、さらには基板が一枚構成になったため、セパレータの必要がなくなり削除した。これにより、中心部での接合部分がなくなり、落下時にパーツ同士が外れてしまうという危険性もなくなった。また、サイズ自体も縦に 5mm 短くなった。

2 号機制作

2 号機はパーツがほとんど分かれていないため、1 回の印刷に 9 時間も必要とした。さらに、モータ・バッテリー部分が浮いているため、積層型 3D プリンターの支柱も多く作られてしまったため、その跡を取り除き消すという作業も行った。

構造の制作と並行してタイヤの制作も行っていた。タイヤは高層図の段階では、展開式にするという予定であったが、展開するための信号を送るためにタイヤ部分に電線を伸ばさなくてはいけないことや、金属パーツを扱うことになるため、正確に加工ができないことなどからこの案は没となった。

構造解説

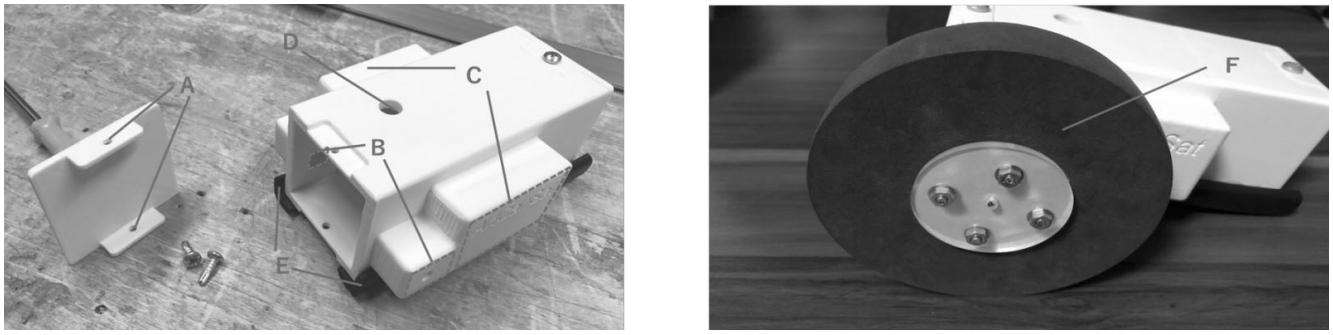

図 3.20　最終機体詳細図

A. 機体は主に、内蔵物を乗せる筒状のパーツ 1 つとそれを塞ぐ蓋状のパーツ 2 つでできている。これらのパーツを接続する際にネジを用いて固定する。ネジを用いることのメリットは、頑丈にパーツとパーツを固定できることと、着脱が容易であることである。着脱が容易であることにより、内蔵物に何らかのエラーが発生した場合に、蓋を取り外し内容物の確認を行う、あるいは修理も行うことができる。

B. モータを内蔵する部分である。この箱状の部分の機体の外側にはモータの回転軸のみが外に飛び出る形になっており、その飛び出した軸とタイヤを接続することでタイヤを回転させる。

C. バッテリーを内蔵する部分である。バッテリーはこの機体において重さの大きな部分を占めている。この機体は、B 部分が前方になり、B 部分の外側にタイヤが付くため重心が前方に偏ってしまう。軽すぎる尾部分を引きずりながら走行してしまうと、後方が跳ねて内蔵物にダメージが入ることが考えられる。そのため、バッテリーをタイヤよりも後方に内蔵することで、重心を少し後ろへずらしている。

D. XBee という機体と PC とで、無線通信を行うための機器のアンテナ部分を出す穴である。

E. タンスの角などを守るクッション材で、機体の緩衝材として作用する。この CanSat の構造上、機体の尾部分を引きずって走行する。ABS 樹脂の状態で走行してしまうと地面との接触で尾が跳ねたりするほか、内部の基板にダメージが入ってしまう。そこで、クッション材で保護してあげることで尾の跳ねを軽減し、中の基板へのダメージも軽減している。同様に落下時の衝撃を和らげる役割を持っている。

F. タイヤパーツである。材料はスポンジでできており、衝撃吸収や柔軟な走行を行うための働きがある。また、タイヤは左右に動かず縦方向のみの回転で、スピードに差をつけることで車体の転回を行う。

（※文責: 松尾威斗）

3.3.3 電装系

この章では、電装系に関わる部品や構造についての説明をする。
使用した部品は、マイコン、モータ、モータドライバ、9 軸 IMU、GPS センサ、EEPROM、プルアップ抵抗、通信モジュール、シールドであり、その特徴と採用理由について説明していく。

マイコンは「Arduino Nano」を使用した。この製品はイタリアの Arduino SRL 社のもので、大きさが 43.2 × 17.8mm で ATmega328P 搭載、動作電圧 5V、推奨入力電源電圧 7〜12V、デジタル入出力ピン 14 本、PWM チャンネル 6 本、アナログ入力チャンネル 8 本、直流電流 1 ピン当り最大 40mA で 3.3V ピンの 1 ピン当り最大は 50mA、Flash メモリ 32KB であり、そのうち 2KB はブートローダーで使用している。採用理由については、Arduino シリーズの中で十分なピン数があり、その中で最も小さいためである。また、班員の私物があったため試験的に用いたことがきっかけである。

モータは「POLOLU-3076」を使用した。購入先は SWITCH SCIENCE である。この製品は大きさが 10 × 12 × 26mm、6V 時の速度が無負荷時 220rpm、最大効率時 170rpm で電流が無負荷時 0.10A、ストール時 15A、最大効率時 0.39A で、トルクがストール時 2.0kg・cm、最大効率時 0.41kg・cm となっている。採用理由については、小型であり、タイヤのスペックに合う性能であるため採用を決定した。また、このモータは担当教員と相談をして決定した。

モータドライバは「DRV8835」をモータ 1 個に対して 1 個使用した。購入先は秋月電子通商である。この製品は、大きさ 10 × 15mm で、2 個 (並列接続時は 1 個) の DC モータもしくは 1 個の 2 相バイポーラステッピングモータを駆動でき、1 回路ごとに 1.5A のドライブ能力、モータ電源 0〜11V、ロジック電源 2〜7V、IN/IN・PHASE/ENABLE の 2 種類の信号付与方式が選択可能になっている。採用理由については、先に決めたモータに合う性能であり、安価・小型であるため採用を決定した。

9 軸 IMU は「MPU9250」を使用した。購入先は Amazon である。9 軸 IMU とは、加速度 3 軸、ジャイロ 3 軸、磁気 3 軸をそれぞれ測定することができるセンサである。本 CanSat 初期プログラムでは、加速度 z 軸と、補正で磁気を用いて機体の向きを測定している。この製品は大きさ 68 × 59mm で電源が 3〜5V で、通信が標準 IIC/SPI 通信プロトコル内蔵の 16 ビット AD コンバーターチップ、16 ビットデータアウトプット、ジャイロスコープ範囲が± 250、500、1000、2000 ° /s で、加速範囲が± 2、4、8、16g で、地場範囲が± 4800 μ T となっている。採用理由は安価であることと、入手が容易であることから採用を決定した。

GPS センサは「GYSFDMAXB」を使用した。購入先は秋月電子通商である。この製品は日本の準天頂衛星システム (QZSS)「みちびき」3 機受信 (衛星番号 193、194、195) に対応しており、NMEA0183 に準拠した緯度・経度・高度・時刻などの各種ナビゲーション情報をシリアル信号で出力できる。みちびき (準天頂の衛星を主とする衛星システム) とは、準天頂軌道の衛星が主体となって構成されている日本の衛星測位システムのことで、英語で QZSS(Quasi-Zenith Sarellite System) と呼ばれている。準天頂軌道の衛星と静止軌道の衛星両方を合わせて呼ぶため準天頂

軌道の衛星と区別する場合準天頂軌道衛星と呼ぶ。大きさ 30 × 30 × 13.5mm(電池ボックス実装時) で搭載 GPS 受信チップが MT3339(MediaTek)、受信周波数が 1575.42MHz(L1,C/A コード)、受信チャネル数が 66(アクイジション) と 22(トラッキング)、対応即位衛生システムが GPS と QZSS、受信 (トラッキング) 感度が-164dBm(typ.)、出力データ形式は NMEA0183V3.01 準拠、電圧が DC5V(3.8〜12V)、電流 40mA、UART 通信速度 9600bps(デフォルト)4800 115200bps、出力データ更新レートが毎秒 1 回 (デフォルト) 毎秒 1〜10 回出力可能、出力データ形式が NMEA0183V3.01 準拠となっている。採用理由は、みちびきには、補完・補強機能があり、山間部でも比較的安定して位置情報が得られること、スマートフォンなどにも使われていることを、GPS について学習する中で知ったためである。

EEPROM は「24LC256」を使用した。購入先は秋月電子通商である。この製品はマイクロチップ製の EEPROM で、256k(32k × 8) ビット EEPROM、2 線式 I2C インターフェース、バススピード 400kHz、動作電圧 2.5〜5.5V で、パッケージは 8 ピン DIP となっている。採用理由は安価であるため採用を決定した。また、CanSat 大会規定では、「機体動作中に得られるデータを記録する」必要があり、そのために実装した。しかし、最終的には、コンソールのログを記録することで、その条件を達成することに変更したため使用していない。また、プルアップ抵抗は、SDA・SCL 間に、$1k\ \Omega$ を 2 個使用した。

通信モジュールは XBee S2C を採用した。購入先は SWITCH SCIENCE である。採用理由としては、構造体が箱型であるため、ワイヤ型のほうが外にアンテナを出しやすく、通信を妨げられることがないと考えたためである。シールドは XBee USB アダプターと XBee-4NANO を使用した。XBee USB は PC 側親機に、XBee-4Nano は機体側子機に使用した。シールドを用いることで、回路設計の手間を省くことができた。購入先は、それぞれ SWITCH SCIENCE、MOUSER である。当初は、これら以外に小型カメラモジュールをつける予定をしていたが、基板を入れるスペースや時間の都合上、今回は使用を断念した。

設計した回路は以下の通りである。

まず、機体内部に基板を入れるスペースが限られていたため、ジャンパー線をできる限り使わないで作成する必要があった。そのため、基板裏ではんだを用いて接続するようにすることが多くなるので、片面基板を用いて作成している。9 軸 IMU と EEPROM は、どちらも I2C 通信を用いるため、並列につなぐことができる。また、それらを並列につないだため、EEPROM はプログラムでプルアップにすることもできたが、9 軸 IMU はプルアップの必要がないため、プルアップ抵抗をつけている。モータは、速度を変える必要があったため、モータドライバのピンはすべて PWM 出力のできるピンに接続した (アナログピンでピン数が足りない)。モータドライバの付属説明書に、可変抵抗を使用する回路が書かれているが、それを採用しなかった理由は、可変抵抗をつけるスペースがないことと、PWM 出力を用いて、プログラムで速度を決めることができるからである。Arduino の VIN ピンは、接続できる電源の電圧が 7〜12V でなければいけないため、9V 乾電池を 2 つ並列に繋げて使用している。

（※文責: 山崎楓太）

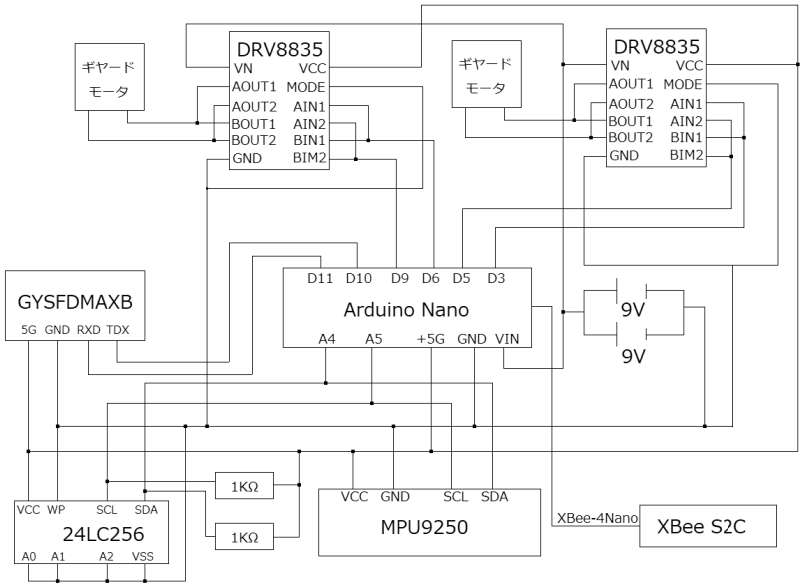

図 3.21　回路図

3.3.4　通信系

CanSat の大会規定において、CanSat に無線通信モジュールを搭載し、PC との通信をするように定められているため、ACS では、XBee S2C を用いて通信を行った。「センサの動作確認を行う」、「機体の現在位置と目標地点の距離を測定する」という目的で、リアルタイムで各種センサ (GPS、9 軸 IMU) のデータを送受信した。

担当者は 1 人で、主に 2 つの作業内容で活動を行った。

1 つ目の活動内容は、XBee の通信設定である。

XBee は、送信側 (機体側) と受信側 (PC 側) の 2 つを必要とし、それぞれに送信、受信の役割を設定する必要がある。そのため、XCTU[4] というソフトウェアを使って、「XBee の ID 登録」「通信モードの設定」を、インターネットで調べて行った [5]。

まずは、PC 側の親機 XBee の設定を行った。PAN ID を任意の値で設定、Coordinator Enable を Coordinator に設定、Serial Number High を 13A200 に設定、Serial Number LOW は Arduino 側の XBee の設定を行うために記憶、Destination Address High を 13A200 に設定、Destination Address Low に Arduino 側の XBee のシリアルナンバーを入力、Baud Rate は 9600 に設定する。

次に、Arduino 側の子機 XBee の設定を行った。PAN ID を PC 側で設定した任意の値を入力、Destination Address High は 13A200 で設定、Destination Address Low は PC 側の XBee のシリアルナンバーを入力、Baud Rate は 9600 に設定する。

通信モードは、AT モードで行った。AT モードは「Arduino でシリアル通信を用いてシリアルモニタに表示されるデータ」と同様のものが無線で PC に送信される、つまりシリアル通信と同様のことを行うため、結果を比較することで動作確認が容易に行えるというメリットがある。

2 つ目の活動内容は、プログラム動作時に PC 側コンソールに表示されるデータログの監視・記録である。

初期プログラム内にある XBee 関数は、プログラム起動からの経過時間・経度・緯度・高度・速度・方向・変数 switch_count を表示するもので、それらの値に異常がないかを監視する役割があった。監視を行うことで、センサやプログラムに異常がないかを確認することが可能である。しかし、初期プログラムを用いた最終実験では、そもそもシリアル通信が正常に行えなかったため、監視も記録もできなかった。その後行った走行実験では、コンソールに表示させる内容を、緯度・経度・方向・変数 switch_count、int 型変数 val に絞って、シリアル通信が詰まらないように通信を行った。その結果、setup 内の表示には多少の文字化けは存在したが、loop 内では、それらのデータを正常にコンソールに表示させることができた。なお、2 つの変数は、プログラムの動作確認のため表示させたため、監視が行えれば十分である。

また、データログの記録についてであるが、XCTU にはコンソールに表示できる量に上限がある。そのため、Tera Term を用いてコンソールに表示させる方法を検討していたが、接続することができなかった。単純にシリアル通信に設定し、接続しているポート番号を設定するだけであるが、それで繋がらなかったため、原因の解明はできていない。よって、監視を有益に行うことはできたが、記録は行うことができなかった。

（※文責: 長谷川沙織）

3.3.5　プログラム系

担当者は 1 人で、主に 3 つの作業内容で活動を行った。

1 つ目の活動内容は、センサなどの部品の動作確認用プログラムを作成、あるいはサンプルプログラムを用意し、基板の制作完了後、そのプログラムを通して動作確認をすることである。

モータドライバ確認用プログラムは「指定したピンに PWM 出力し (値は-255〜255 をとる)、キーボードの入力に従って、前方・左回転・右回転・後転する」という動作を行うものを作成した。GPS センサ確認用プログラムは「得られた緯度・経度・高度のそれぞれの値をシリアルポートに出力する」という動作、9 軸 IMU 確認用プログラムは「9 軸から得られたそれぞれの x, y, z 成分の値をシリアルポートに出力する」という動作を行うもので、それぞれインターネットで調べて用意したものである [6] [7]。

動作確認は、作り替える前の基板とその後の基板で、2 回行った。結果、GPS センサと 9 軸 IMU は正常に動作した。モータドライバも正常に動作はしたが、タイヤの構造上の問題でまっすぐ進まなかったため、「両輪の値を少しずつずらして走らせる」という原始的な試行を何度か行うことでプログラムでの微調整を図った。最終的に、まっすぐ走ることはできなかったが、微調整のための両輪の回転数の差を一意に定めることができた。なお、その値は全体プログラムに適用されている。

2 つ目の活動内容は、すべての動作確認が終了したのちに、実験に用いる全体プログラムを作成することである。

まず、ミッションプロセスに対して、プログラムはどのような動きをするべきであるかを考え、フローチャートを作成した。そのフローチャートは以下の通りであるが、最終発表会に用いるために、最終実験終了後に新たに加筆・修正したものである。

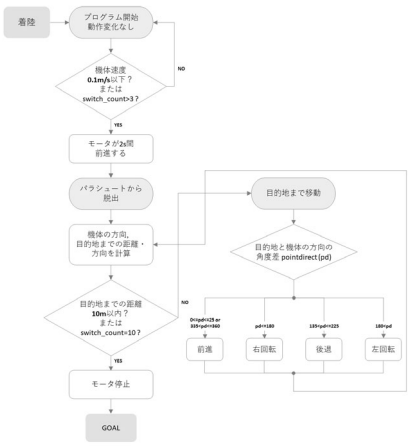

図 3.22　初期プログラムのフローチャート

次に、プログラムの作成に移った。しかし、この時点で、最終実験までに 2 週間程度の時間しかなかったので、B グループのプログラムの雛型などを参考にしながら作業をした。プログラムの構成は以下の図のとおりである。なお、この元となる初期プログラムは、最終発表会で用いた Web

サイトに示したリンク先に存在し、本報告書の付録 A にも載せてある。

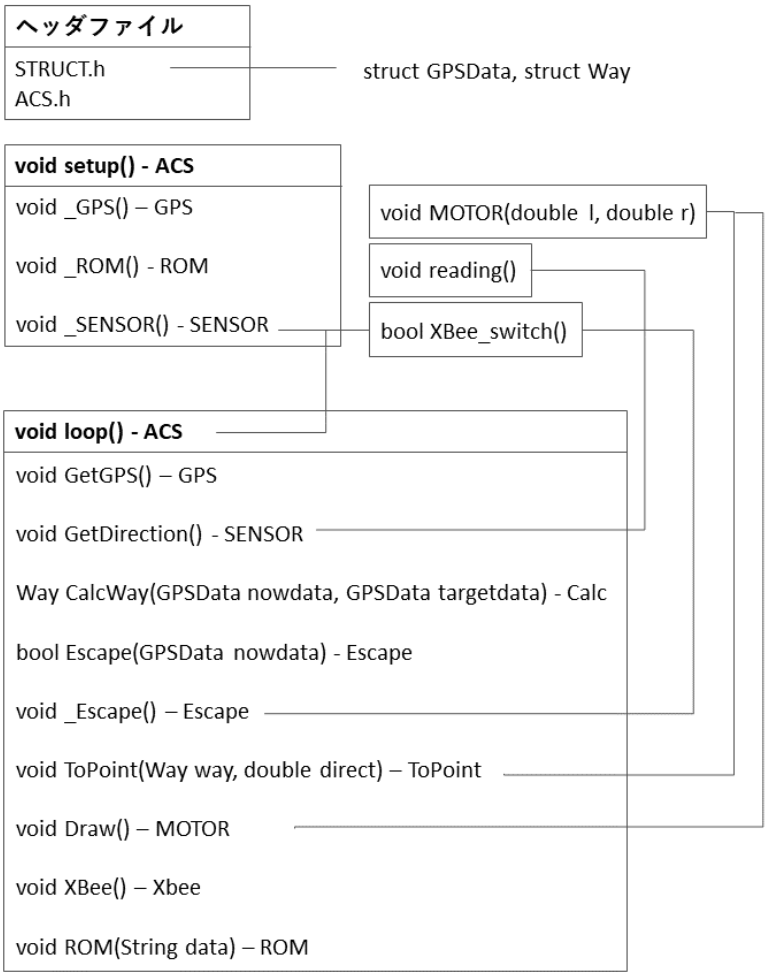

図 3.23　初期プログラムの構成図

ここで、2 つのヘッダファイルと、ACS.ino 内に記述してある関数、緊急用関数、使用したライブラリについて簡単に説明する。

「ACS.h」

メイン関数をまとめたヘッダファイルである。

「STRUCT.h」

構造体 GPSData と構造体 Way を定義するヘッダファイルである。

次に、ACS.ino 内の関数について説明する。まずは、setup 関数内の関数について説明する。

「_GPS()」「_ROM()」「_SENSOR()」

GPS、EEPROM、9 軸 IMU の動作確認・初期設定を行う関数である。

次に、loop 関数内の関数について説明する。

「GetGPS()」

GPS センサから、緯度・経度・高度・速度を取得する関数である。

「GetDirection()」

現在の機体の方向を取得する関数である。本プログラムでは、9 軸 IMU の z 軸の加速度の値を主に用いて、誤差補正に磁気センサの値を用いた。

「Escape(nowdata)」

GPS から得られた現在の高度・速度の情報を用いて、パラシュートから脱出するための関数である。

「_Escape()」

Escape 関数による脱出が何らかの原因でうまくいかなかった際に、緊急用関数を用いて脱出する関数である。

「ToPoint(way, direct)」

道のりの情報と機体の向きの情報を計算して、進むべき方向に機体の向きを変えて進む関数である。

「Draw()」

絵を描くための動きをする関数である。本プログラムでは、ハート形のような形を描くように記述した。

「XBee()」

緯度・経度・高度・速度・方向および変数 switch_count を、XBee による通信を通して、PC 側コンソールに表示させる関数である。XBee が AT モードで設定されているため、単にシリアル通信を用いてコンソールに表示させるプログラムと同様のものとなっている。

「ROM(mpsdata)」

EEPROM にデータを書き込む関数であり、今回は GPS から得られた速度データを対象とした。しかし、最終実験後は、「コンソール上のログを保存する」ことで大会規定の条件をクリアすることに変更したため、最終的には EEPROM を用いていない。

ここで、loop 関数内に直接出ていないが、重要な関数について説明する。

「MOTOR()」

ギヤードモータに対して PWM 出力できるピンを用いることで、疑似的にアナログ出力ができる。本関数では、-255〜255 の値を用いてアナログ出力することで、機体の速度を調整した。

次に、緊急用関数について説明する。

「XBee_switch()」

センサ類の動作不良、バグなど、何らかの原因でプログラムが動作しない際に、親機 XBee からコマンドを送信することで強制的に動かす緊急用関数である。この関数では PC から任意の文字を 1 つ入力し、子機が受信することで true を返すので、それを別条件として使用する、あるいは、緊急用関数を使用するたびに値が増加する int 型変数 switch_count を条件として用いることで、プログラムを強制的に動かすことを可能とした。

最後に、ライブラリについて説明する。

「TinyGPS++.h」 [8]

GPS のデータを、エンコード関数を用いてエンコードすることで、扱いやすい形に変換するという用途で用いた。

「SoftwareSerial.h」

XBee で通信をする際に必要とした。

「Wire.h」

9 軸 IMU が I2C 通信をするため必要とした。

「EEPROM.h」

EEPROM に書き込み・読み込みをする際に用いた。

「math.h」

三角関数など、数学で用いる関数を使用する際に必要とした。

「string.h」

文字列操作に用いた。

「stdlib.h」

汎用的な関数を使用する際に必要とした。

3 つ目の活動内容は、プログラムの実行およびバグチェック、プログラムの修正を行うことである。

初期プログラムは結果として動作せず、その際に問題個所を摘出するため、「要所に Serial.print() を用いる」「コメントアウトする」などを用いて、デバッグを行った。

最終実験後、限られた短い時間のなかで、動かなかった原因を追究し、できる限りシリアル通信を用いないプログラムを考えた。結果的に、millis() を用いて時間で動くプログラムに簡易的に変更し [9]、走行実験において実行したが、ToPoint() がうまく動作しなかった。コンソールに示された direct の値が常に 0.1 を示していたため、後に 9 軸 IMU の動作確認をいろんな手段を用いて行った結果、加速度センサから得られる値が、変動してはいるもののほぼ 0 に近い値を示していた

ため、センサの不良であると考えた。その後、磁気センサは動いていたため、GetDirection() を、磁気センサの x, y 軸の値を用いて方向を取得するものに変更し、全体的にバグを取り除いたプログラムを最終的に作成した付録 B のプログラムが、その改良版プログラムである。なお、改良した部分のみを載せてある。

（※文責: 長谷川沙織）

第 4 章　結果

この章では、成果として活動で得られた知識、成果物、及び経験についての説明を行う。また、解決手段と評価として、手段の妥当性や成果についての評価を行う。

（※文責: 長谷川沙織）

4.1　全体実験について

本グループは笹流ダムと公立はこだて未来大学の敷地内で異なる実験を行った。
新型コロナウイルス感染症 (COVID-19) の影響により制作作業時間が短く機体の完成に時間がかかり実験回数が少なく、参加可能人数が制限されている中、笹流ダムの利用可能時間中で実験をすべて終わらせる必要があった。

笹流ダムの実験の手順は以下の通りである。

まず初めにダムから CanSat を落下させ、パラシュートを展開させる。その後、CanSat が地面に着地後地面を走行し、目標地点を目指し、到達時停止するというものである。

実験可能日の前半では、機体が完成しておらずパラシュートの落下実験のみを行った。パラシュートの実験では、機体の想定重量と同程度の重量に調整した水の入ったペットボトルを乗せ、ダムの上から落下させパラシュートを予定通り展開できるかどうかや、想定した速度で落下するかの計測などを行った。

中盤では改良の過程で制作され使用する予定のない初号機を乗せダムの上から落下させパラシュートを予定通り展開できるかどうかや、想定した速度で落下するかの計測などのほかにペットボトルでの実験ではできなかった機体に対する衝撃の確認などの実験を行った。

後半の実験では、実際に実験に使用する機体をパラシュートに乗せ実験を行った。この実験では、予定通りの手順で実験を行った。この実験では、落下場所が悪く CanSat が走行できる場所に着地できなかったためタイヤは動いているが、CanSat が進むことはなかったために走行実験を笹流ダムで実施することを断念するに至った。そのため公立はこだて未来大学で着地以降の走行実験を行った。

こちらの実験では機体が向いている方向と GPS によって与えられた目標地点の方向の差をもとに機体を回転させ、目標地点に向かうというシステムのうち機体が向いている方向を割り出すシステムがうまく機能せず、想定した動作で目標地点に向かうことはなく円形移動を取りながら一定方向に移動するにとどまり、目標地点に到達できず失敗に終わった。しかし、その機体の円形移動範囲内に目標地点が入る位置で起動し、走行させたとき目標地点に到達時に CanSat が停止したので停止条件や GPS 自体に問題がないことがわかった。

（※文責: 山崎楓太）

4.2 成果

前期

前期には、電子部品の組み立てや、i-CanSat を参考にした勉強を行った。設定した問題・課題に対する直接的な成果は得られていないが、設計に関わる知識を得ることができた。具体的には、設計に用いるソフトウェアや機械の使い方、設計に利用するものの材質の特性、電子部品の性質、回路設計の基礎などを学んだ。また、それらの勉強を通して、成果物の基本設計を考案した。

後期

後期は、飛行物系・地上構造系・電装系・プログラム系 (通信含む) に分かれて、実際に制作に取り組んだ。その後、各々完成次第、パーツごとに実験を行い、全体を通した実験も行った。実験は、「飛行単体実験」を 2 回、「最終実験」「走行単体実験」をそれぞれ 1 回ずつ行った。

飛行単体実験 1 回目では、500g の重さに耐えられるパラシュートに対して、ペットボトルで作った重りを 350g に設定して、高さおよそ 30m 程度のダムから、畳んだ状態のパラシュートと重りを投下した。結果、想定通りの速度 (約 3m/s) で落下したため、成功とした。

飛行単体実験 2 回目では、同様のパラシュートに重りを入れる用の袋を付けた。また、重りの重量を約 500g に増量し、1 回目同様に実験を行った。結果、想定通りの速度で落下したため、成功とした。

最終実験では、飛行単体実験で用いたパラシュートに CanSat を入れて、落下〜目標地点まで走行・ゴール到達の 1 連の流れを計画し、実験を行った。その前段階として、同日、先に走行のみ実験を行ったが、ハードおよびプログラムの問題で走行ができなかった。しかし、1 度通して実験を行いたいという気持ちもあって、プログラムを、単に走行するだけのモータドライバ動作確認用に変えて最終実験に臨んだ。

結果として、CanSat を破損させることなくパラシュートが落下することは達成したが、落下場所が悪かったため、その後の走行は不可能であった。サクセスクライテリアで考えると、ミッションの 60 %を成功したことになる。

その後行った走行単位実験では、コンクリートの上であれば、タイヤはまともに動くのではないかと考え、大学敷地内で走行のみをテストした。結果、CanSat は自分の方向が取得できず、目標地点に向かって走行することはできなかった。原因をその場で考えた結果、コンソールに表示されていた値に異常を発見したので、プログラムの問題ではないと判断した。そのため、目標地点に向かって走行したとして、最後に問題となる「ゴール地点での停止」を実験することにした。

「ゴール地点での停止」は、B グループの最終実験までの結果を踏まえて、「目標地点から半径 5m 以内の範囲に入った場合停止する」という条件のプログラムで行われる (なお、目標地点ちょうどをゴールとしない理由は、GPS から得られるデータには誤差が生じるから

である)。目標地点に向かおうとしてタイヤが回転したままの機体を、ゴール地点付近まで持っていき、地面に下して走らせた結果、半径 5m 以内の範囲に入ったときに停止した。

（※文責: 長谷川沙織）

4.3 解決手段と評価

前期

前期は、雑誌、論文、インターネットなどから、CanSat に関わる基礎知識を勉強した後、ミッション中に発生し得る問題点を事前に想定し、「衝撃」「走行性」「姿勢制御」「パラシュート」「メンテナンス」についてそれぞれ解決手段・対策方法を考案した。また、7 月に行われた中間発表のアンケート結果から得られた意見や疑問点をグループ内で確認し、考案した手法について再度検討を行った。8 月には、必要となる材料の調達および発注を行い、その間に、ミッションに適した外殻の構造の考案や、入手した電子部品の動作確認などを行った。

後期

後期は、それぞれ専門に分けて作業を行った結果、前期に対して進捗具合が良かったように感じられた。ただし、それぞれの担当の間での情報共有や連携が重要となり、その大切さを実感した。最終実験に対しての解決手段と、その評価を以下に行う。

まず、パラシュートに関して評価を行う。最終実験での落下速度は想定通りのものであり、機体の破損はなかったため、計算にもとづいて作成したパラシュートは成功していると評価できる。また、最終実験以前に、2 回シミュレーション実験を行っていたことも成功を保証していたと考える。しかし、パラシュートでは、落下地点を制御できるわけではないので、今後このプロジェクトを再度行う際には、落下地点の重要性も考慮して飛行物の制作にあたる必要があると考える。

次に、構造物に関して評価を行う。落下時に生じる衝撃に対して破損がなかったため、構造の出来がよかったと評価できる。破損リスクを抑えるための設計は成功であったと考える。反面、そのために生じた問題もあった。パーツを少なくすることで、衝撃による分解のリスクを抑えることはできたが、実験時に必要となる基板のセッティングが、毎度大変で時間がかかるものとなった。また、まっすぐ走行できないという問題があった。原因は解明していないが、タイヤのホイールのずれ、基板を固定していないため機体の重心が毎度異なるなどの原因が考えられる。特に、砂利道の上では走行が難しかったので、障害物による走行不能のリスクも想定して設計を行う必要があると考える。

続いて、回路設計に関して評価を行う。モータの選定の際に、わからないところは担当教員に質問するなどして、適切なものを選べた点はとても良かったと考える。その他部品に関しても、用途やマイコンのことを考慮し、適切なものを選べたと考える。結果的に、モータのスペックは期待通りのもので、部品も満足できる働きをした。また、ハードの限られた

スペースに基板を入れる必要があったため、最小限のジャンパー線を用いてはんだづけを行う必要があった。ジャンパーを使わずはんだを用いて基板裏で接続するなどの工夫をし、ショートなどを起こすことないように、定期的にテスターを用いて回路をチェックしながら作成した結果、厚さ 3cm 程度におさえることができたので、成功であると評価できる。しかし、回路図を書かずに基板を制作した、構造担当との情報共有があまりできていなかったなどの原因で、1 度基板制作に失敗しており、回路設計に時間を割きすぎたと考えられるため、今後はそれらの点に注意するべきであると考える。

最後に、プログラムおよび通信に関して評価を行う。最終発表で、センサの誤差や、緊急時を考えたプログラムの設計が良いと思ったという外部評価を得られた。しかし、初期プログラムに関しては、最後まで動作しないという問題があった。また、その後の走行実験に関しては、プログラムでなくセンサが原因であった。最終工程であるため、全体的に時間がなかったとはいえ、走行単体実験を 1、2 回設けることができれば、結果はもう少し良かったのではないかと考える。そして、定期的に部品の動作確認を行うべきであると考える。通信に関しては、シリアル通信を行った際に生じる文字化けに対して、いろんな解決方法を試した点は良かったが、結果的に原因は解明していない。また、Tera Term との接続ができなかった点に関しても原因は解明していない。しかし、機体走行時に各種センサから得られるデータや変数を、シリアル通信の問題を考慮して、すべて得るのではなく取捨選択を行い、適切に監視を行えた点に関しては良かったと考える。

（※文責: 長谷川沙織）

第 5 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やグループ内における活動について説明する。

（※文責: 森宗誠太）

5.1　プロジェクト全体のインターワーキング

プロジェクト全体のインターワーキング前期のプロジェクトでは、新型コロナウイルス感染症（COVID-19）の影響によって、プロジェクト学習全体でオンラインによる会議が主であった。そのため、プロジェクトメンバー間での会議は、Zoom を使用して行い、資料の共有などは Slack を使用して行った。また、参考資料やプロジェクトメンバー個々が調べてきた資料などの分類を明確にするために、Slack のチャンネル機能を使用した。後期のプロジェクトでは、徐々に大学での作業をできる機会が増えた。また、後期の作業は主にグループ単位での活動となった。グループ同士の情報交換は、前期同様、主に Slack と Zoom を使用して行った。また、大学での作業時には、ソーシャルディスタンスを保ちつつ情報交換を行った。なお、グループ内でのインターワーキングは、第 5 章第 2 節で説明する。

（※文責: 森宗誠太）

5.2　A グループのインターワーキング

本グループの作業の前期進捗管理は、毎週 2 回のオンライン会議の時に行った。しかし、この方法では、正確な進捗管理ができたかについて不安が残った。これらを踏まえ、後期では Google スプレッドシートなどを利用した方法で進捗管理を行うことを改善策として挙げた。また、前期の作業分担は、少し偏りがあったので後期はできるだけ負担の大きさが均等になるように努めようと考えた。本グループは、少人数での作業のため、グループメンバーだけでは解決できないような問題点は、B グループメンバーの意見も参考にしつつ、様々なコミュニケーションツールを利用してメンバー同士での情報共有・解決策の模索を行った。全体的にグループ内の雰囲気を大切に、プロジェクト全体の工程を楽しく活動ができたと考える。

後期の進捗管理は、前期同様、3 回の内 2 回のオンラインでの会議の時に行った。しかし後期の作業については、主に機体の制作を行ったため、このオンライン時での進捗確認は、軽く行う程度で、正確な進捗管理は、大学での作業時に行った。また、後期の作業については「グループ」というものに固執せず、B グループメンバーの意見も参考にしつつ、機体の制作を行った。しかし、作業分担については若干偏りがあったことに関しては、問題点として残った。しかし、全体を通して、グループ内の雰囲気を大切に、プロジェクト学習全体の工程を楽しく活動ができたと考えている。

（※文責: 森宗誠太）

第 6 章　まとめ

この章では、中間発表会と最終発表会の成果とグループメンバー個々の役割などの評価を行い、プロジェクト学習のまとめを行う。また新型コロナウイルス感染症下でのプロジェクト運営についてもここで説明を行う。

（※文責: 森宗誠太）

6.1　プロジェクトの成果

この節では、中間発表会と最終発表会の結果を利用して、プロジェクトの結果の考察・評価を行う。

（※文責: 森宗誠太）

6.1.1　中間発表会

この項では、中間発表会の概要や結果と結果に対する評価を行う。また 2020 年度中間発表会については、新型コロナウイルス感染症（COVID-19）予防の観点からオンラインでの実施となった。

日時と場所

2020 年 7 月 17 日金曜日に、制作していた Web サイトを評価者が確認、その後 Zoom を利用してオンライン上で簡潔にグループ概要を述べたのち、質疑応答という形の中間発表会を行った。

展示

中間発表会では、プロジェクト全体で 1 枚のメインポスターを制作したのち PDF 化した。また、例年通りの対面でのプレゼンテーションをすることができないため、Web サイトの制作を行った。

Zoom での質疑応答

先に制作を行った Web サイトだけでは、十分にグループ活動の内容が伝わらない可能性があったのでプレゼンテーションの資料を作成してプレゼンテーションを行った。

結果と評価

今回の中間発表の評価は、例年通りに紙媒体を作成・配布して評価をしてもらうことができなかったので、Google Forms を利用して発表技術と発表内容について 10 段階評価とコメントを行ってもらった。また、今回は評価場所の構成がわかりにくかった面があったた

め、発表技術と発表内容の両方において無回答が多くなっている。
発表技術についての評価点の平均は、7.7 点でおおむね良好だったといえる。「Web サイトが見やすかったです」や「サイトに絵があるためわかりやすかった」などのコメントから Web サイトに関する評価は良好であった。その反面、「発表が棒読みだった」「質問対応が少し塩対応かなと思う」など、発言に対する評価はやや悪かった。
発表内容に関する評価は、こちらも平均が 7.9 点でおおむね良好だったといえる。「目標が明確でわかりやすかった」などの意見から目標に関する評価は良好であった。その反面、「もう少し詳しく説明してほしかった」など発表の内容全般に関する評価は少し悪かった。しかし、準備期間が少なかったがこのような評点が出ていることは誇りに思っている。なお、評価点数をまとめたグラフを以下に示した。(図 6.1) (図 6.2)
　また中間発表用に作成した Web サイトの URL は、付録 C に掲載している。

（※文責: 森宗誠太）

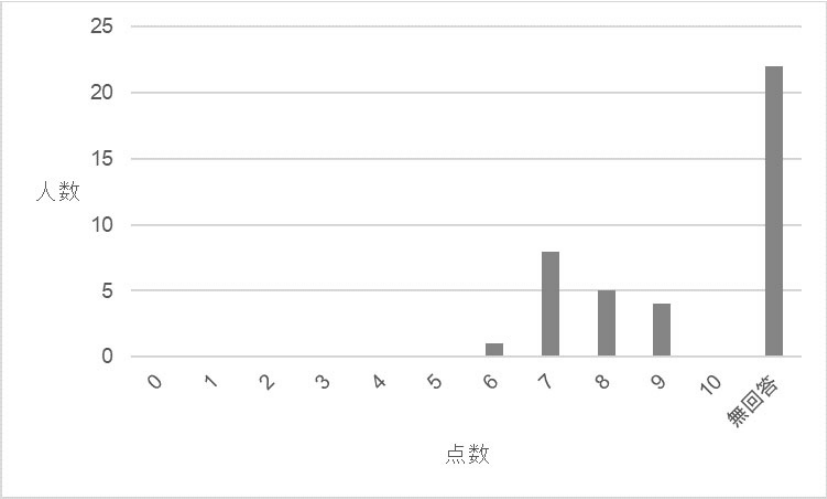

図 6.1　中間発表技術評価点数

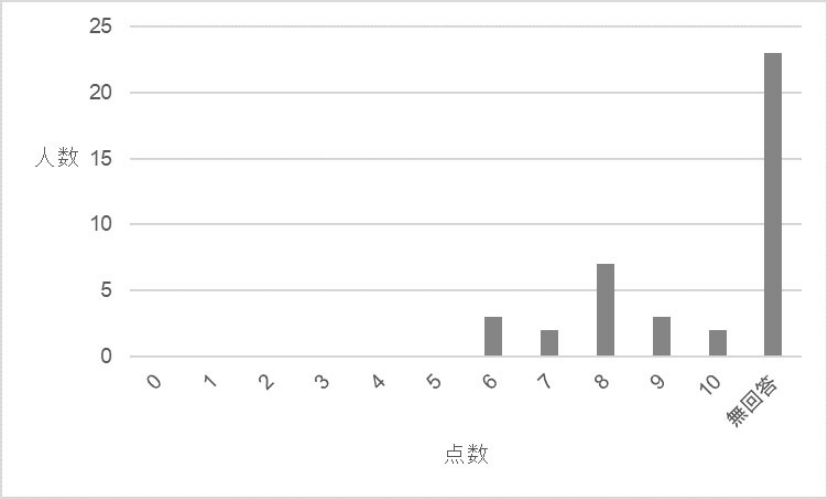

図 6.2　中間発表内容評価点数

6.1.2 最終発表会

この項では、最終発表会の概要や結果と結果に対する評価を行う。また、2020 年度の最終発表会については、中間発表会同様、新型コロナウイルス感染症 (COVID-19) 予防の観点からオンラインでの実施となった。

日時と場所

2020 年 12 月 4 日金曜日に、制作していた Web サイトを評価者が確認、その後 Zoom を利用してオンライン上での質疑応答という形の最終発表会を行った。

展示

最終発表会では、プロジェクト全体で 1 枚のメインポスターを制作したのち PDF 化した。また、例年通りの対面でのプレゼンテーションをすることができないため、動画作成を行うまたは、Web サイト制作を行うかの 2 択の選択肢があったが、作成の都合などから Web サイトの制作を行った。

Zoom での質疑応答

Zoom での質疑応答の際、中間発表会の時のように質疑応答前の軽いプレゼンテーションを省いたため、質疑応答が出にくい場合には、グループでの活動の概要については説明を行った。また同じプロジェクト同士で質問を出し合い、質問者が質問しやすい環境の構築を行った。

結果と評価

今回の最終発表の評価は、中間発表同様に、紙媒体を作成・配布して評価をしてもらうことができなかったので、Google Forms を利用して発表技術と発表内容について 10 段階評価とコメントを行ってもらった。

発表技術についての評価点の平均は、7.6 点でおおむね良好だったといえる。コメントの内容は、「概要がわかりやすかった」などの肯定的な意見もあったが、その反面、「情報が散逸的でわかりにくかった」などの否定的な意見も見られた。また、「プレゼン資料などの補助的な資料があってもよかった」等の「質疑応答時に、もう少し説明があってもよかった」のような意見があった。しかし、この意見に対しては、発表練習の時にメンバーで話し合ってあくまでも質疑応答の時間であるため、最初の説明は省くという結論に達した。そのために今回は省いたが、説明は省くということを質疑応答の最初に伝えていればよかった。

発表内容に関する評価は、こちらも平均が 7.8 点でおおむね良好だったといえる。肯定的な意見としては、「コロナ渦の中でここまでの成果物をできたことがすごいと思う」や「オンラインでのグループでこのような成果物ができたことはすごいと思う」などの私たちの成果物の作業に対しての意見が多かった。しかし、「目標が明確ではなかった」等の目標に関することについては、否定的な意見があった。このことに関しては、もう少し初期の段階で目標を明確にするための議論を行っていればよかった。また、「プロジェクト学習としてどのような学びがあったかどうかを発表してもらえるとよかった」等の意見があった。このことは、発表内容に大々的には含まれていなかったので、含めるべきだと思った。なお、評価点数をまとめたグラフを以下に示した。(図 6.3) (図 6.4)

また最終発表用に制作した Web サイトは、付録 C に掲載している。

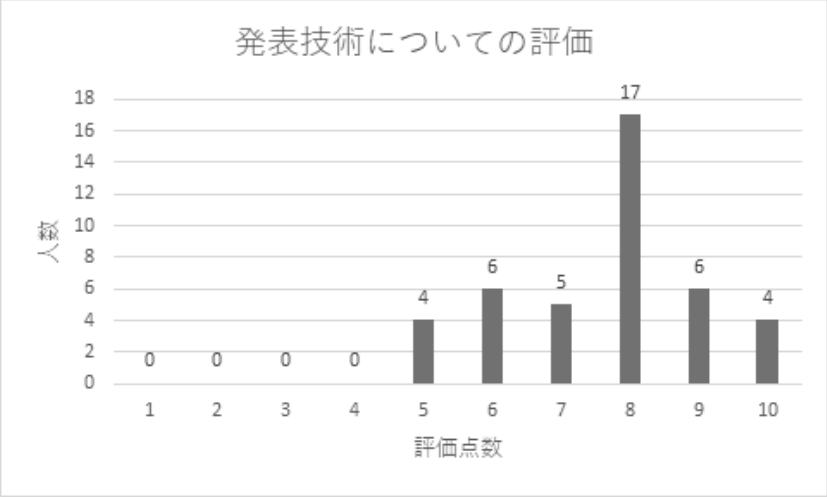

図 6.3　最終発表技術評価点数

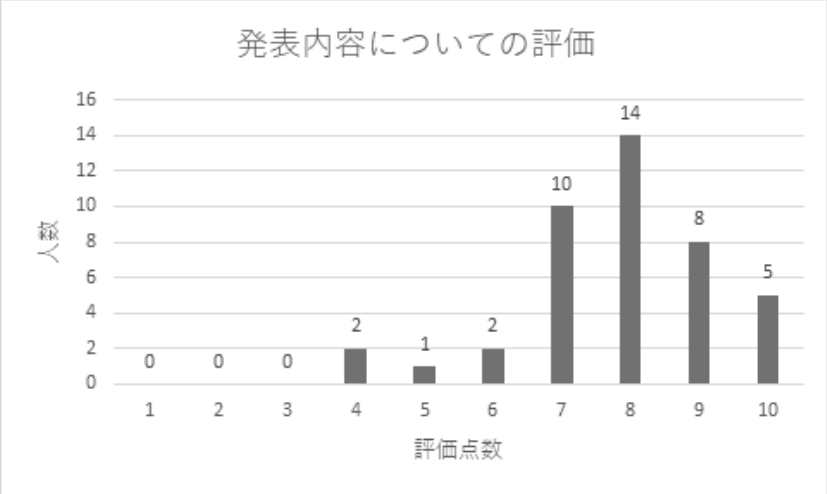

図 6.4　最終発表内容評価点数

（※文責: 森宗誠太）

6.2 プロジェクト内での自分の役割

この節では、グループメンバー個々が自分の役割とそれに対する評価を行う。

（※文責: 森宗誠太）

6.2.1 森宗誠太

グループリーダー兼ハードウェア設計を担当した。まず、グループリーダーとして気を付けてたことは、グループの雰囲気を壊さないことだ。グループ学習である以上これは一番大切にしないといけないことだと考えている。このことについては、しっかり守れた。しかし、グループリーダーとしてグループメンバーを正確にまとめる点においては、進捗管理などの面においてしっかりできた自信がない。そのため、後期には、しっかり雰囲気は維持しつつ、しっかりグループのメンバーをまとめたいと考えている。ハードウェア設計においては、ホーマックなどに素材を確認しに行った。しかし、設計の面ではほかの人に任せきりであった部分もあったと感じているので、後期には、しっかり設計部分や制作部分において、積極的に作業していきたいと考えている。

後期は、前期同様、グループリーダーとハードウェアの中でも飛行物系に関して担当した。飛行物系に関しては、グループメンバーと共同で作業を行った。私は、主にプロトタイプの制作を行い、プロトタイプ制作の段階で生じた問題点や、作り方の工夫した点などの情報を、本制作を行うメンバーに提供した。後期も前期同様に、グループ内の雰囲気を壊さないようにプロジェクト学習の進行を行った。このことについては、前期同様守れたと思っている。グループリーダーの活動については、前期よりはしっかりグループメンバーをまとめられていたと考えている。グループメンバーの提出物に関しては、チェックシートを作成し、確認を行った。しかし、作業進捗状況の確認の面においては、グループメンバー全員の状況を確実に把握できていたかといわれると不安な点が残った。そのために、将来このような作業を指揮するときには、常に進捗管理を行い、先をよんでグループメンバーなどに助言をできるように頑張りたい。最終的に、最終発表会ぎりぎりまで実験をできる環境を構築できたことで、メンバーの努力が無駄にならなかったことは、うれしく思っている。

（※文責: 森宗誠太）

6.2.2 松村海斗

前期は会議で意見を出したり、このプロジェクトでセンサ類のはんだ付けや i-CanSat の作成に携わった。また、後期の役割は、パラシュートの作成、改良、落下実験そして実験後の動画から速度と位置の解析を行った。

また、後期の役割は、パラシュートの作成、改良、落下実験そして実験後の動画から速度と位置の解析を行った。A グループの CanSat の電装系、飛行物系を担当した。私が電装系を担当したのは、制作に本腰を入れる前の後期前半であったので、電装系としてはんだ付けと i-CanSat の制作に携わった。はんだ付けに関して、初めてではなかったが数年ぶりであったので初めは少し難しく感じた。飛行物系は、今まで 1 度も触った事がなかったため 1 から CanSat のパラシュートに求められる要件を調べた。その際に他の CanSat プロジェクトの失敗した原因を調べ、起こりうる原因

をあげた。作り方に関しても、他の CanSat プロジェクトの制作方法を参照して制作した。また、パラシュートの落下速度に関してもグループメンバーと相談して決定した。パラシュート落下実験の後、パラシュートの改良を行った。また、苦戦したがプロジェクトメンバーに聞くことで実験の動画をからパラシュートの落下速度と垂直位置を解析した。

グループメンバーのタスクが重かったためタスクをうまく分けたり、進捗具合を確認することが意外と難しいことに気づかされた。プロジェクト学習の会議では、全体会議で発言することは少なかったが、グループ活動では、自分の意見を述べることができたと思う。また、他の人の意見に対しても意見を出したこともあった。

（※文責: 松村海斗）

6.2.3 長谷川沙織

A グループの CanSat のプログラム・通信系を担当した。また、電装系において、使用する部品の選定、回路設計も担当した。グループメンバーで考えた全体のミッションの流れに沿って、機体をどのように動かすかを考え、フローチャートを作成し、全体のプログラムを作成した。今までに、Arduino は何回か使用しているため、まったくの初心者ではなかったが、初めてヘッダファイルを作成したり、初めて使うライブラリや関数があったりなど、学べたことも多かった。また、自律移動アルゴリズムなどを学ぶことができた。通信系は、扱うことに関しては初心者であったが、あまり難しいものでもなかったので扱うことはできた。今回 XBee について勉強する際に、情報ネットワーク (講義) や基本情報技術者試験で得た知識が役に立ったと思う。しかし、知識不足も実感したため、良い経験になったと考える。電装系は、部品の選定がとても大変であったが、論文やインターネットで調べたり、わからないところは先生に質問したりして、適切なものを選ぶことができた。また、回路設計は、今までに何度か経験していたため、その経験と知識を生かして行うことができた。また、「VIN ピンに外部電源を接続できる (電源電圧に制限はある)」など、新しく勉強できたこともあり、良い経験になったと思う。また、担当したものすべてが内部の設計に関わるものであったこと、タスクが大きめであったため、外部の設計についての情報があまり頭にないなどの問題が発生し、ハードとソフトの連携がとても難しかった。協力することの難しさを改めて実感した。全体ミーティングでは、積極的に提案したり、意見を述べることができたと思う。

（※文責: 長谷川沙織）

6.2.4 松尾威斗

A グループのハードウェア設計・制作を担当した。ハードウェア設計・制作の上で気をつけたことは、メンバーの要望を取り入れていくことと同時に作業時間との兼ね合いを見て現実的に制作できるのかという点を考えながら設計したことである。今回のプロジェクト学習は新型コロナウイルス感染症 (COVID-19) の影響によって作業時間の多くが削られたことは何度も述べているが、一番ダメージを受けたのはこのハードウェアの制作と言っても過言ではないくらいに時間が足りなかった。大学での作業日が少なくなった分、試行回数を重ねることが困難であった。そのため、一度の試行回数により多くの意見や改善案などを考えてから設計・制作しなければならなかった。初期構想の段階では、ミッションを成功させるために様々な機能を機体に与える構想図を制作したが、この中の多くの機能は没となり、よりシンプルなものへという判断にせざるを得なかった。し

かし、いくら話し合いを重ねた設計図で制作しても実際に物理的に制作したときに問題はいくつも生まれてくる。こうして生まれた問題をいかに解決するかというのも非常に困難であった。ハードウェアの視点だけで言えば、もう 2、3 回は試作機を作りたかったというのが本音である。しかしながら、実質的に作業できたのが後期のみであったという状況の中では十分な制作物であったと思う。

ハードウェア関連以外で私が貢献したことは、ミーティングの中で積極的に意見を出すのはもちろん他者の意見にもコメントすることに気をつけた。また、ハードウェアは他の基板や飛行物とも関連することが多かったため、他メンバーの進捗にも気を張りながら、時には意見を交換しながら取り組んだ。

今回のプロジェクトを踏まえて今後に活かしたいことは、実際に行動するタイミングを早めにするという点である。今回は大学での作業こそ後期のみしかできなかったものの、部品ごとのテストや材料のテストなどまだ詰めれるところはあったのではないかと考える。また、コミュニケーションはやはり重要であり、今回のプロジェクトでは一つのことを決定するのに時間をかけすぎていると感じる部分が多くあったので、効率的なコミュニケーションというのを心がけたいと考える。

（※文責: 松尾威斗）

6.2.5 山崎楓太

A グループの電子機器関連の作業を担当した。その他にも i-CanSat の制作や、制作に関する話し合いでの改善点や問題点の発見やアドバイスなどを行った。後期には制作作業が本格的に始まり、新型コロナウイルス感染症 (COVID-19) の影響で登校が減少した結果、作業時間があまり取れないため、CanSat を完成させなければならないため作業を素早く進めていく必要があった。その中で基盤制作を 2 機分の制作を行った。最初の基盤は作業のしやすさや、作業速度を優先するために配線を長めにとっていたが基盤が機体の外装に入れにくく邪魔となり実験に問題が出ることがわかったため実験に支障がきたさないようにするために 2 機目の制作を行った。こちらは配線をできるだけ短くし、制作作業の速度が下がり配線が破損したときなどの整備の難易度が上がる代わりに機体の外装に入れやすくし、実験を行いやすいものを制作した。また実験の序盤でうまく CanSat が起動せず、配線が断線していた。断線しているのが原因であることも多く、そこの対策や改良を行いたかったが制作時間の関係から基盤の修繕作業だけにとどまった。

（※文責: 山崎楓太）

6.3 新型コロナウイルス感染症下におけるプロジェクト学習

本年度は、新型コロナウイルス感染症が世界各国で猛威を振るっている。そのために、本年度のプロジェクト学習は、新型コロナウイルス感染症予防の観点から色々と作業が制限されながらの活動となった。この節では、本グループのプロジェクト学習の流れとその反省点を説明したあと、今後このような状況になった場合の進め方について提案を行う。

（※文責: 森宗誠太）

6.3.1 プロジェクト学習の流れ

本年度のプロジェクトの流れについて以下に示す。

4 月下旬〜6 月 29 日頃

オンラインでの実施した。

6 月 29 日頃〜後期開始

原則オンラインでの実施した。指導教員の届け出によって、大学の決めた制限ルールを達成した人でなおかつ、少人数の登校が許可された。この期間の中で、オンライン (Zoom) を使用した中間発表会があった。

後期開始〜10 月中旬頃

原則オンラインでの実施した。大学内にいる人数が全体の 3 分の 1 以下になるように、3 回に 1 回大学で作業するもしくは毎回プロジェクトメンバーの 3 分の 1 だけが作業することが許可された。

10 月中旬頃〜最終発表会

原則オンラインでの実施した。大学内にいる人数が全体の 3 分の 1 以下になるように、3 回に 1 回大学で作業することの許可＋ 3D プリンタなどの工房の機械を使用する場合に限り 1 日 2 時間 (プロジェクトメンバーの 3 分の 1) だけ登校許可された。

6.3.2 反省点と解決策

今年度のプロジェクト学習では、ほとんどの期間がオンラインでの作業となったために意思疎通が対面に比べるとできない問題点が発生した。そのために今後は、まず初めに、どのような方法でいつコミュニケーションをとるかのルールを決めておくことをお勧めする。

今年度のプロジェクト学習においては、プロジェクト全体に対する情報の伝達が学生に伝わるのが遅かった。また、情報が大雑把すぎて、学生によって情報の交錯が起こったため、このような状況下こそ、情報は詳細にかつ迅速にお願いしたい。

（※文責: 森宗誠太）

第 7 章　謝辞

2020 年度のプロジェクト学習において、新型コロナウイルス感染症が猛威を振るっている中、公立はこだて未来大学工房職員、函館市企業局上下水道部浄水課の担当者他協力していただいた皆様に心から感謝の気持ちと御礼申し上げます。

（※文責: A グループメンバー一同）

※函館市企業局上下水道部浄水課　笹流ダム
URL:https://www.city.hakodate.hokkaido.jp/docs/2014022500484/

付録 A　初期プログラム

「ACS.h」

```
#ifndef ACS_H
#define ACS_H

#include "STRUCT.h"

Way CalcWay(GPSData nowdata, GPSData targetdata); //ok
bool Escape(GPSData nowdata); //ok
bool _Escape(); //ok
void GetDirection(); //ok
void _SENSOR(); //ok
void ToPoint(Way way, double direct); //ok
void GetGPS(); //ok
void XBee(); //ok
bool XBee_switch(); //ok
void ROM(String data); //ok
void MOTOR(double l, double r); //ok
void Draw();

#endif
```

「ACS」

```
#include "ACS.h"
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

#define DELAY 1000

TinyGPSPlus tgp;
SoftwareSerial mySerial(10,11);
GPSData targetdata={0.0,0.0,0.0,0.0};
GPSData nowdata={0.0,0.0,0.0,0.0};
double direct=0.0;

bool task=false; //ゴールに着くまでやること
unsigned long lastupdate=0;
int switch_count=0; //XBee_switchを使ったときに増えるカウンター
int val=0; //カウンター
int drawing=0; //カウンター
int subtaskend=0; //カウンター


void setup(){
  //Serial.begin(9600);
  _GPS();
  _ROM();
  _SENSOR();
}


void loop(){
  if(!task){

    GetGPS();

    if(lastupdate+DELAY<millis()){
      GetDirection();
      Way way=CalcWay(nowdata,targetdata);

      if(Escape(nowdata) || _Escape()) val++;
```

```
        if(val>0 || subtaskend==1 || switch_count==7){
          ToPoint(way,direct); drawing++;
        }
        if(drawing>2 && subtaskend==0){
          Draw();
          subtaskend++;
        }
        if(way.distance<10.0 || switch_count==10) task=true;
      }

      XBee();

      String mpsdata=String(nowdata.mps,10);
      ROM(mpsdata);

      lastupdate=millis();

    }
```

「Calc」

```
#include "ACS.h"
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

#define GOAL_LAT
#define GOAL_LON


Way CalcWay(GPSData nowdata, GPSData targetdata){
  Way way={0.0,0.0};

  way.distance=
    TinyGPSPlus::distanceBetween(
      nowdata.lat,
      nowdata.lon,
      targetdata.lat,
      targetdata.lon) / 1000.0;

  way.direct=
    TinyGPSPlus::courseTo(
      nowdata.lat,
      nowdata.lon,
      targetdata.lat,
      targetdata.lon);

  return way;
}
```

「Escape」

```
#include "ACS.h"
#include <math.h>

#define stopalt 1.0
#define stopmps 0.1
#define starttime 1500
#define FORCED 18000

int t=0;
unsigned long s=0;

bool Escape(GPSData nowdata){
  double mps=nowdata.mps;

  if(s==0){

    if(abs(targetdata.alt-nowdata.alt)<=stopalt && fabs(mps)<=stopmps){
      t+=DELAY;
      if(t>=3500) s=millis();
    }else t=0;

  }else{
    t=0;
    if(s+starttime>=millis()){
     s=millis();
     while(s+2000>millis()) MOTOR(255,255);
     return true;

    }else{
     MOTOR(0,0);
    }
  }
  return false;
}

//緊急脱出用
bool _Escape(){
  t=0;
```

```
    if(XBee_switch()==true && switch_count>3){
      s=millis();
      while(s+2000>millis()) MOTOR(255,255);
      return true;
    }
    return false;
  }
```

「GPS」

```
#include "ACS.h"
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

void _GPS(){
  while(!Serial);
  mySerial.begin(9600);
  Serial.println("GPS...ok");
}

void GetGPS(){
  while(mySerial.available()>0){
    char c=mySerial.read();
    tgp.encode(c);

    if(tgp.altitude.isUpdated()){
      nowdata.lat=tgp.location.lat();
      nowdata.lon=tgp.location.lng();
      nowdata.alt=tgp.altitude.meters();
      nowdata.mps=tgp.speed.mps();
    }
  }
}
```

「MOTOR」

```
#include "ACS.h"

int L1=3; //左モータにつながってる
int L2=5;
int R1=6; //右モータにつながってる
int R2=9;

void MOTOR(double l, double r){
  l=(int)l; r=(int)r;

  //forward
  if(l>0 && r>0){
    l-=12;
    analogWrite(L1,l); analogWrite(L2,0);
    analogWrite(R1,r); analogWrite(R2,0);

  //reverse
  }else if(l<0 && r<0){
    l-=16;
    analogWrite(L1,0); analogWrite(L2,(abs(l)));
    analogWrite(R1,0); analogWrite(R2,(abs(r)));

  //brake or turn
  }else{
    analogWrite(L1,l); analogWrite(L2,0);
    analogWrite(R1,r); analogWrite(R2,0);
  }
}


unsigned long timer=0;

void Draw(){
  while(direct>235 || direct<55) MOTOR(200,-200);
  timer=millis();
  while(timer+6000>timer) MOTOR(200,200);

  while(direct>275 || direct<85) MOTOR(200,-200);
```

```
    timer=millis();
    while(timer+2000>timer) MOTOR(200,200);

    while(direct>330 || direct<160) MOTOR(200,-200);
    timer=millis();
    while(timer+2000>timer) MOTOR(200,200);

    while(direct>240 || direct<100) MOTOR(200,-200);
    timer=millis();
    while(timer+2000>timer) MOTOR(200,200);

    while(direct>330 || direct<160) MOTOR(200,-200);
    timer=millis();
    while(timer+2000>timer) MOTOR(200,200);

    while(direct>275 || direct<85) MOTOR(200,-200);
    timer=millis();
    while(timer+6000>timer) MOTOR(200,200);
}
```

「ROM」

```
#include "ACS.h"
#include <EEPROM.h>
#include <string.h>
#include <stdlib.h>


void _ROM(){
  Serial.println(EEPROM.length());
  Serial.println("EEPROM...ok");
}

int next=0;

void ROM(String data){
  byte buf[8192];
  int len=data.length();
  data.getBytes(buf,len); //copy to buf[]

  //Write
  for(int i=0;i<len;i++) EEPROM.write(i+next, buf[i]);

/*
  //Read
  for(int i=0;i<len;i++){
    EEPROM.read(buf[i]);
     Serial.println(buf[i]);
  }
*/

  next+=len;

}
```

「SENSOR」

```
#include <Wire.h>

#define FIL 0.9          //フィルタ
#define AVE 100          //測定回数
#define GYRO 30          //ドリフト許容値

int emp[6]={};
int data=0;              //生データ格納 1から順に加速度xyz角速度xyz
long datasum=0;          //合計データ格納　平均算出用
int th=0;                //閾値
long integral=0;         //積分値
int rock;                  //ロック機構により導かれた初期値
char input;

void _SENSOR(){
  Wire.begin();  Serial.begin(9600);  Wire.begin(0x68);
  Wire.beginTransmission(0x68);  Wire.write(0x6B);  Wire.write(0x00);
Wire.endTransmission();
  Wire.beginTransmission(0x68);  Wire.write(0x1C);  Wire.write(0x10);
Wire.endTransmission();
  Wire.beginTransmission(0x68);  Wire.write(0x1B);  Wire.write(0x08);
Wire.endTransmission();
  Wire.beginTransmission(0x68);  Wire.write(0x1A);  Wire.write(0x05);
Wire.endTransmission();

  Serial.println("Press[1].");
  while(1){
    if(XBee_switch()==true && switch_count==1){
      reading();
      datasum=0;
      Serial.println("Turn 360 degrees and Press[2].");

      if(XBee_switch()==true && switch_count==2){
        rock=integral/360;
        Serial.println("SENSOR...ok");
        integral=0;
        delay(500);
        break;
```

```
        }

      }
    }

}



void reading(){
  for(int i=0;i<AVE;i++) {
    Wire.beginTransmission(0x68); Wire.write(0x3B); Wire.endTransmission();
    while(Wire.available()<14);
    emp[0]=Wire.read() << 8 | Wire.read();
    emp[1]=Wire.read() << 8 | Wire.read();
    emp[2]=Wire.read() << 8 | Wire.read();
    emp[3]=Wire.read() << 8 | Wire.read();
    emp[4]=Wire.read() << 8 | Wire.read();
    emp[5]=Wire.read() << 8 | Wire.read();
    data=Wire.read() << 8 | Wire.read();
    datasum+=data;
  }
  datasum/=AVE;

  if(th==0) th=datasum;

  if(abs(datasum-th)>GYRO){ //誤差が小さい場合はスルー ドリフト対策
    integral+=datasum-th;
  }
}



void GetDirection(){
  reading();
  direct=abs(integral/rock);
  Serial.println(direct);
  datasum=0;
}
```

「STRUCT.h」

```
#ifndef STRUCT_H
#define STRUCT_H

struct GPSData{
  double lat;
  double lon;
  double alt;
  double mps;
};

struct Way{
  double distance;
  double direct;
};

#endif
```

「ToPoint」

```
#include "ACS.h"
#include <math.h>

#define TPI 360.0

void ToPoint(Way way, double direct){
  double pointdirect=fmod(way.direct-direct+TPI,360);

  if((pointdirect<=25 && pointdirect>=0)||(pointdirect<=360 &&
      pointdirect>335)) MOTOR(200,200);

  else if(pointdirect<=180) MOTOR(150,-150);

  else if(pointdirect<=225 && pointdirect>135) MOTOR
      (-150,-150);

  else if(pointdirect>180) MOTOR(-200,200);
}
```

「XBee」

```
#include "ACS.h"

void XBee(){
  Serial.print("");
  Serial.print("TIME = "); Serial.println(millis()); //プログラム起動時間
  Serial.print("LAT = "); Serial.println(nowdata.lat, 6); //緯度
  Serial.print("LONG = "); Serial.println(nowdata.lon, 6); //経度
  Serial.print("ALT = "); Serial.println(nowdata.alt); //高度
  Serial.print("M/S = "); Serial.println(nowdata.mps); //速度
  Serial.print("DIRECTION = "); Serial.println(direct); //方向
  Serial.print("");
  Serial.print("SWITCH COUNT: "); Serial.println(switch_count);
}

bool XBee_switch(){
  char receivedata[1];
  if(Serial.available()>1){
    Serial.readBytes(receivedata,1);
    Serial.print("receive: ");
    Serial.println(receivedata);
    switch_count++;
    Serial.print("count [");
    Serial.print(switch_count);
    Serial.println("]");
    return true;
  }else return false;
}
```

付録 B　改良版プログラム

「ACS」

```
#include <MPU9250_asukiaaa.h>

#include "ACS.h"
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

#define DELAY 100

TinyGPSPlus tgp;
SoftwareSerial mySerial(10,11);
GPSData targetdata={41.839778,140.766281,0.0,0.0}; //目標地点
GPSData nowdata={0.0,0.0,0.0,0.0};  //現在位置
double direct=0.0; //機体の向き

bool task=false;
unsigned long lastupdate=0;
unsigned long counttime=0;
int val=0;
Way way;

void setup(){
  Serial.begin(9600);
  _GPS();
  _SENSOR();
}

//最初にゴール地点の位置情報取得する
/*
void loop(){
  GetGPS();
  XBee();
 // GetDirection();
  delay(500);
}
```

```
*/



//改良版プログラム(走行のみ)

void setup(){
  Serial.begin(9600);
  _GPS();
  _SENSOR();

  //push (1)
  //方位磁石で機体向き北に合わせる

}


//3s run
//draw
//if(draw=false) push (2)
//if(end=false) push (3)


void loop(){

  GetGPS();

  if(lastupdate+DELAY<millis()){


      GetDirection();
      way=CalcWay(nowdata,targetdata);

      if(!task){
        if(val>0) ToPoint(way,direct);

        if(3000<millis()) val=1;

        if(val==1 && (way.distance<35.0 || 8000<millis())) Draw();
```

```
            }

            if(way.distance<5.0 || XBee_switch()==true) {
              task=true; MOTOR(0,0);
          }

          XBee();
          delay(100);

          lastupdate=millis();

        }

      }
```

「SENSOR」

```
#include <MPU9250_asukiaaa.h>

#ifdef _ESP32_HAL_I2C_H_
#define SDA_PIN 21
#define SCL_PIN 22
#endif

#define CALIB_SEC 10

MPU9250_asukiaaa mySensor;

uint8_t sensorId;
float mX, mY, mZ;
char input;

void _SENSOR(){

  while(!Serial);
  Serial.println("started");

#ifdef _ESP32_HAL_I2C_H_ // For ESP32
  Wire.begin(SDA_PIN, SCL_PIN); // SDA, SCL
#else
  Wire.begin();
#endif

  mySensor.setWire(&Wire);
  while (mySensor.readId(&sensorId) != 0) {
    Serial.println("Cannot find device to read sensorId");
    delay(2000);
  }
  mySensor.beginMag();

  float magXMin, magXMax, magYMin, magYMax, magZ, magZMin, magZMax;

  Serial.println("ok? --> Press.");

  while(1){
```

```
        if(XBee_switch()==true){
        Serial.println
            ("Start scanning values of magnetometer to get offset values.");
        Serial.println
            ("Rotate your device for " + String(CALIB_SEC) + " seconds.");
        setMagMinMaxAndSetOffset(&mySensor, CALIB_SEC);
        Serial.println("Finished setting offset values.");
        break;
        }
      }

    }

    void setMagMinMaxAndSetOffset(MPU9250_asukiaaa* sensor, int seconds){
      unsigned long calibStartAt=millis();
      float magX, magXMin, magXMax,
            magY, magYMin, magYMax,
            magZ, magZMin, magZMax;

      sensor->magUpdate();
      magXMin = magXMax = sensor->magX();
      magYMin = magYMax = sensor->magY();
      magZMin = magZMax = sensor->magZ();

      while(millis() - calibStartAt < (unsigned long) seconds * 1000){
        delay(100);
        sensor->magUpdate();
        magX = sensor->magX();
        magY = sensor->magY();
        magZ = sensor->magZ();
        if (magX > magXMax) magXMax = magX;
        if (magY > magYMax) magYMax = magY;
        if (magZ > magZMax) magZMax = magZ;
        if (magX < magXMin) magXMin = magX;
        if (magY < magYMin) magYMin = magY;
        if (magZ < magZMin) magZMin = magZ;
      }

      sensor->magXOffset = - (magXMax + magXMin) / 2;
```

```
    sensor->magYOffset = -(magYMax + magYMin) / 2;
    sensor->magZOffset = -(magZMax + magZMin) / 2;
  }

  void GetDirection() {
    //Serial.println("sensorId: " + String(sensorId));

    mySensor.magUpdate();
    mX = mySensor.magX();
    mY = mySensor.magY();
    mZ = mySensor.magZ();
    direct = mySensor.magHorizDirection()+180;

    Serial.println("horizontal direction: " + String(direct));

    /*
    Serial.println("at " + String(millis()) + "ms");
    Serial.println(""); // Add an empty line
    delay(500);
    */
  }
```

付録 C　発表会 Web サイト (発表会資料)

C.1　中間発表会

Project 17　めざせ宇宙開発 - 自律移動ロボット飛行プロジェクト
URL: https://fun-cansat.amebaownd.com/

C.2　最終発表会

Project17　めざせ宇宙開発 - 自律移動ロボット飛行プロジェクト
URL: https://fun-cansat2.amebaownd.com/

参考文献

[1] 大阪府立大学　工学研究科　小型宇宙機システムセンター　CanSat Project.
http://www.sssrc.aero.osakafu-u.ac.jp/activity/cansat/ (最終閲覧日:2020/8/19)

[2] UNISEC(大学宇宙工学コンソーシアム)　ARLISS.
http://unisec.jp/activities/arliss.html (最終閲覧日:2020/8/19)

[3] kinovea
https://www.kinovea.org/ (最終閲覧日:2020/12/23)

[4] DIGI / XCTU.
https://www.digi.com/products/embedded-systems/digi-xbee/digi-xbee-tools/xctu.
(最終閲覧日:2020/12/18)

[5] garchiving.com / XBee ZB(S2C) で Arduino とパソコンの無線通信｜簡単な接続通信テスト.
https://garchiving.com/xbee-arduino-pc-test/
(最終閲覧日:2020/12/18)

[6] DEVICE PLUS / 第 60 回 Arduino で GPS 情報を取得〜Arduino でパーツやセンサーを使ってみよう.
https://deviceplus.jp/hobby/entry060/ (最終閲覧日:2020/12/18)

[7] AKIRACING.COM / 【9 軸センサ】MPU9250 の使い方と ARDUINO プログラム.
http://akiracing.com/2018/02/04/how_to_use_mpu9250/ (最終閲覧日:2020/12/18)

[8] Arduiniana / TinyGPS++.
http://arduiniana.org/libraries/tinygpsplus/ (最終閲覧日:2020/12/18)

[9] asukiaaa.blogspot.com / Arduino で MPU9250（加速度センサ、磁気センサ）を使う方法.
https://asukiaaa.blogspot.com/2017/07/arduinompu9250.html?m=0
(最終閲覧日:2020/12/18)