ACTS報告書

提出日：2021年　12月　26日

- チーム情報

CanSatチーム名	電気通信大学　Salamander
CanSatチーム 代表者	松田 尚也 matsuda.naoya@cas.lab.uec.ac.jp, 080-5534-3276
UNISEC団体名	電気通信大学 高玉研究室
UNISEC団体 学生代表	白石 洋輝 hirowhite@cas.lab.uec.ac.jp, 090-1208-0322
責任教員	高玉 圭樹 keiki@inf.uec.ac.jp, 042-443-5808

- メンバー

役割	名前(学年, 経験年数)
リーダー 兼 ソフト班*	松田 尚也(M1, 経験 2年目)
ソフト班	古屋 敬祐(M1, 経験 1年目)
回路班*	河野 航大(M2, 経験 3年目)
回路班	戸板 佳祐(B4, 経験 1年目)
副リーダー	白石 洋輝(M1, 経験 2年目)
ハード班*	加藤 駿(M1, 経験 1年目)
ハード班	金子 颯汰(B4, 経験 3年目)

- CanSatの製作目的・大会参加理由

  弊団体はチーム開発を通して協調性を上げること, 開発に必要な技術力を身につけること, および研究室におけるCanSatに関する技術向上を目的にCanSat開発および大会への参加を実施している.

目次

第1章　　ミッションについて

1. ミッションステートメント(ミッションの意義と目的)

提案:実数値GAに基づくモータの動作不全に対して頑健な進化的自律走行CanSat

本プロジェクトでは，前輪2輪，後輪2輪の計4輪から構成される，モータの動作不全に対して頑健な進化的自律走行CanSatを提案し，その性能を検証する．

惑星探査において，惑星上で探査機に人の手を介入させることは困難であり，探査機が着陸時や地表探査時に故障して走行性能が損なわれると，探査計画に支障をきたす．そこで，探査機にハードウェア故障が生じ，本来期待する走行動作から乖離する場合においても，目的地までの自律走行を達成することが可能なCanSatを提案する．

特に，本ミッションでは自律走行に支障をきたすような故障の中で，発生する可能性が高いと想定されるモータ関連の故障に焦点を当てる．図1.1は提案する4輪のCanSatにおけるモータ部分が故障した際の走行制御の可否を故障の箇所と故障種類で分類したものである．空転はタイヤにモータの出力が伝わらずに電力による動作が不可能な状態(物理的な回転は可能)，固定はタイヤやモータの軸が衝撃などで歪み物理的に回転不可能な状態，ノイズはモータの回転数が安定しない状態，紛失はモータに接続されていたタイヤが機体から外れている状態を指す．

本CanSatでは前輪が走行性能の大部分を担っており，後輪が故障したとしても紛失しない限りは両前輪による走行制御が可能である(図の斜線部に相当)．一方で前輪の片方が故障した際にはあらかじめ設定されていたモータ出力で走行を制御することは困難となるが，タイヤが紛失していなければ残るもう片方のタイヤや後輪部分のモータ出力値を調整することで走行する方向を制御可能な状態に復帰する可能性がある．(ただし，両前輪ともが故障した場合は後輪のモータ出力を調整しても機体が路上で前進するだけの走行は期待しにくい．) そこで，本ミッションでは前輪の片方が故障した状況に対応できるCanSatの実現の第一歩として，故障の中でもモータの「空転」が発生している場合を想定する．(「固定」については片方の前輪を引きずるように走行させる必要があるため，「空転」時よりもダイナミックな出力調整が必要であり，調整可能範囲で走行性能を復帰させることが困難な場合がある．「ノイズ」についてはモータの動作が安定しないために方向制御がモータ出力値の調整だけでは不完全である．) このとき，機体がPID制御による進行方向制御が不可能であると判断し，走行時の情報(例:アクチュエータへの出力や一定時間に走行した直線距離，ゴール方向への距離)から，近似解探索において優れた性能が報告されている進化的アルゴリズムの1つである遺伝的アルゴリズム(Genetic Algorithm: GA)[Katoch, 21]を用いることで，前輪1輪，後輪2輪のモータ回転数と2軸サーボモータの角度を最適化し，再度機体の進行方向を制御可能な状態に復帰させることを目指す．さらに，これにより，目的地へ自律走行を達成できることを確認する．また，本プロジェクトではACTSでの時間制限を考慮し，前輪1輪，後輪2輪のモータ回転数を最適化する際にあらかじめシミュレーション環境で最適化された値をGAの初期値とすることで，最適化に要する計算時間の短縮をねらいとする．

[Katoch, 21] Katoch, S., Chauhan, S.S. and Kumar, V. A review on genetic algorithm: past, present, and future. *Multimed Tools Appl* 80, 8091–8126 (2021).

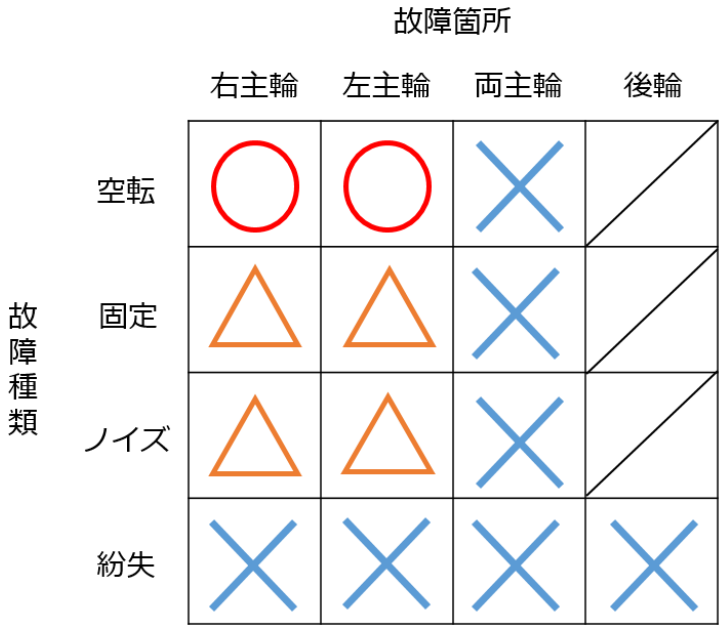


図1.1 本CanSatにおけるモータ故障時の故障していないモータの出力値調整による走行性能の可否
（〇：走行制御が可能，△：走行性能が部分的に復帰，×：走行制御不可，斜線：調整の必要なし）

2. ミッションシーケンス

ミッション全体の流れは以下の通りである．また，下の図1.2はこれらの項目を模式的に表したものである．

I. **[待機シーケンス]**：CanSatはキャリアに収納され，上空で放出されるまで待機状態となる．CanSatはキャリアから放出後，光センサを用いて放出判定を行う．放出後にはパラシュートも開傘．

II. **[落下シーケンス]**：9軸センサ，気圧センサを用いてCanSatが降下中か着地したかの状態判定を開始．

III. **[パラシュート分離シーケンス]**：着地判定後，CanSatはパラシュートを切り離す動作を開始．サーボモータを複数回動かすことによりパラシュートを切り離す(パラシュート切り離し機構については3章を参照)．

IV. **[ナビゲーションシーケンス]**：予め設定されたゴール地点のGPS地点情報を取得し，その地点に向けてPID制御走行する．ただし，片方の前輪モータの故障を判定した場合にはPID制御走行に重要な直進可能なサーボ角度・モータ出力の最適値を求めるための最適化サブシーケンスへ遷移する．そして，片方の前輪モータの故障を再現するためにシーケンス開始時に左の前輪モータの出力を0に固定する処理を一回実行する．(この処理自体は故障判定ではない)

[最適化サブシーケンス]：最適なサーボ角度・モータ出力を以下の手順で求める．
まず，あらかじめ用意した物理シミュレーション環境下で片輪が故障した状況を設定し，GA(解の評価・選択・交叉・突然変異を繰り返すことでより良い解を求める手法)を

用いて故障時の最適なサーボ角度・前輪モータ出力を複数パターン求めておく．実際にCanSatが片輪が故障したと判定した時に，あらかじめ求めていた最適な出力パターンを初期個体として，GAによってサーボ角度・モータ出力の実環境における最適値探索をする．設定された世代数の探索が完了したらもっとも最適であったサーボ角度・モータ出力をPID制御内部に設定し，ナビゲーションシーケンスに復帰する．（最適化サブシーケンスについては4章を参照）．

V. **[ゴール判定シーケンス]**：GPS座標によりゴール5[m]以内に達したら，カメラによるゴール判定を開始する．カメラに映る赤い物体の割合を検知し，旋回・直進を繰り返しながら赤い物体の割合が増える方向へ機体を制御．赤色の割合が予め設定された閾値を超えた場合その位置をゴールと判定しミッションを終了する．

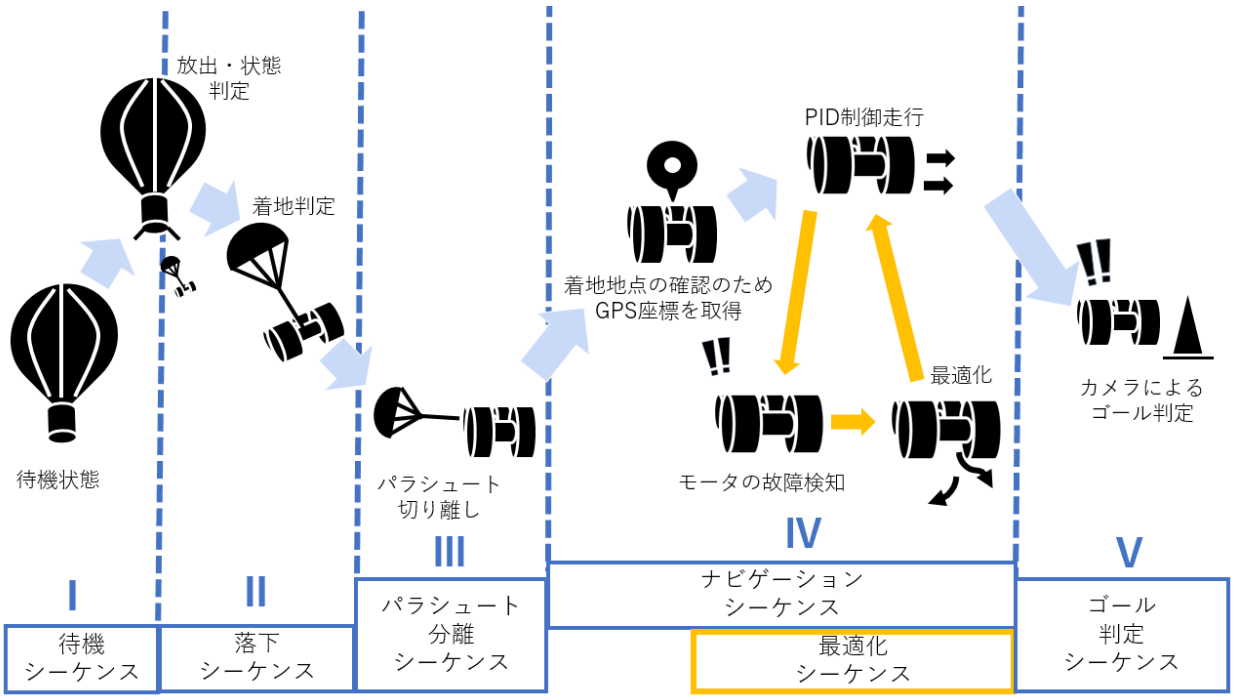


図1.2 ミッションシーケンスの流れ

第2章　　サクセスクライテリア

ミニマムサクセス	意図的に左前輪のモータへの出力を故障が生じたと想定される値にしたときに故障を検知し，故障したモータとそれに対応するタイヤが空転しているか否かが判定できる.
フルサクセス	判定した故障の情報をもとに，駆動系の出力値を最適化アルゴリズム(GA)によって調整し，最適化後の駆動系の出力値において，プログラムでの制御方向と地磁気センサで獲得した実際の進行方向の角度誤差が$3\boldsymbol{\epsilon}_{deg}$ 度以下で走行できる†. †$\boldsymbol{\epsilon}_{deg} = 3.00$ は故障が生じていない状態での進行方向の最大角度誤差を示しており，(V17)最適化実行試験における実験結果に基づいて定めた値である.

また，本ミッションでは以下のエクストラサクセスを設定する.

エクストラ サクセス	意図的に1つの任意のモータへの出力を故障が生じたと想定される値にしたときに，駆動系の出力値を最適化することによって，プログラムでの制御方向と地磁気センサで獲得した実際の進行方向の角度誤差が$3\boldsymbol{\epsilon}_{deg}$ 度以下で走行できる†. †$\boldsymbol{\epsilon}_{deg} = 3.00$ は故障が生じていない状態での進行方向の最大角度誤差を示しており，(V17)最適化実行試験における実験結果に基づいて定めた値である.

なお，フルサクセスおよびエクストラサクセスにおいては，達成レベルを{〇，△，×}の記号表現を用いて定量的に定める. 具体的には，最適化後のCanSatにおける進行方向の角度誤差をθ_{opt}とし，各記号における達成条件を

$〇 : \theta_{opt} < 3\boldsymbol{\epsilon}_{deg}$, $i.e.,\ \theta_{opt} < 9.00°$

$△ : 3\boldsymbol{\epsilon}_{deg} \leq \theta_{opt} < 5\boldsymbol{\epsilon}_{deg}$, $i.e.,\ 9.00° \leq \theta_{opt} < 15.00°$

$× : \theta_{opt} \geq 5\boldsymbol{\epsilon}_{deg}$, $i.e.,\ \theta_{opt} \geq 15.00°$

と定める.

第3章　　要求項目の設定

1. システム要求（安全確保、レギュレーションのための要求）

要求番号	自己審査項目
	ACTS**安全基準**
R1	質量と容積がレギュレーションを満たすことが確認できている
R2	ロスト対策を実施しており，有効性が試験で確認できている （例：地上局にダウンリンクする場合，ACTSで十分な通信距離が実現できるだろうと推測できる根拠が明確に示されていること．）
R3	地表近くで危険な速度で落下させないための減速機構を有し，その性能が試験で確認できている
R4	打ち上げ時の準静的荷重によって，安全基準を充足するための機能が損なわれないことが試験で確認できている
R5	打ち上げ時の振動荷重によって，安全基準を充足するための機能が損なわれないことが試験で確認できている
R6	分離時の衝撃荷重によって，安全基準を充足するための機能が損なわれないことが試験で確認できている
R7	打ち上げ時の無線機の電源OFFの規定を遵守できることが確認できている（FCC認証かつ100mW以下の機器はOFFしなくて良い．また，スマートフォンを用いる場合はFCC認証かつソフトウェアまたはハードウェアスウィッチでoffにできること（2017年追加））
R8	無線のチャンネル調整に応じる意思があり，また実際に調整ができることを確認できている
R9	R1-R8の充足を確認した設計の機体によって，ロケットへの装填から打ち上げ後の回収までを模擬したEnd-to-end試験を実施できており，今後，安全性に関わる大幅な設計変更はない
R10	CanSatの収納・投下準備が5分以内でできている
	カムバックコンペティションルールの充足
R11	ミッション時に人間が介在しない自律制御を実施することが確認できている （注：2014年のレギュレーション改定以降，地上局設備に計算機能を持たせてアップリンクしても良い）
R12	ミッション後，規定された制御履歴レポートを運営者へ提出する準備ができている （以下の根拠の項に制御履歴レポートの例を添付すること．ダミーデータを使用しても良い）

2. ミッション要求（ミッションを実現するにあたり要求される性能）

ミッションの内容から，自己審査項目については以下のように設定した．

要求番号	自己審査項目（ミッション実現要求項目）
M1	着地時の衝撃荷重によって，ミッションを実現するための機能が損なわれていないことが試験で確認できている
M2	環境の悪い地面における走行性能を試験で確認できている
M3	競技時間を十分に走行可能であることを確認できている
M4	前輪の故障が検知可能なことを確認できている
M5	機体が正常に直進できない状態から駆動系の出力値の最適化を開始し，終了後に直進走行ができることを試験で確認できている
M6	CanSatが反転・復帰した時に通常の走行姿勢に復帰できることを試験で確認できている
M7	GPS座標とPID制御によりゴール地点座標へ機体が方向制御をし到達できることが確認できている
M8	ゴール地点に設置されたコーンを検知し，0mゴールすることができることを確認できている

第4章　　システム仕様

3章の要求項目に対応する本CanSatの機構を以下の表4.1にまとめる．また，以降では4.1節にてCanSatの外観について記載し，4.2節にてハードウェア部分における特記事項としてパラシュートの切り離し機構，スタビライザについて記載する．4.3節では，回路を構成するパーツとパーツ同士の関係性についてまとめ，4.4節にて全体的なアルゴリズムフローや最適化アルゴリズムについて記載する．

表4.1 要求項目に対応するCanSatの機構

要求番号		主に寄与するCanSatの機構
安全面	R1	全体的なハードウェア設計
	R2	LoRaモジュールよる長距離通信
	R3	パラシュート，パラシュート切り離し機構
	R4	全体的なハードウェア設計
	R5	
	R6	
	R7	LoRaモジュール，Raspberry PiのWi-Fi
	R8	
	R9	全体的なハードウェア設計
	R10	
カムバックへの参加へのルール充足	R11	全体的なソフトウェア設計
	R12	プログラム実行時のログ保存

ミッション面	M1	スポンジタイヤやカップリングによる衝撃吸収
	M2	タイヤの形状・スパイク
	M3	搭載するバッテリーの容量
	M4	故障検知アルゴリズム
	M5	スタビライザ，最適化アルゴリズム
	M6	横転・反転復帰アルゴリズム
	M7	目的方向へのPID制御アルゴリズム
	M8	カメラを用いたゴール検知アルゴリズム

1. CanSat外観

本ミッションでは，モータの故障に対しても頑健なCanSatを実現させるための工夫として，四輪駆動で走行できるようにしている．それに伴い，垂直・水平方向2軸化されたスタビライザと後輪部分を前輪の内側に納められるようにサーボを繋ぎ合わせた．

CanSatの正面を撮った画像を図4.1，上面から撮った画像を図4.2に示す．図4.1の回路基板下にラズパイを起動するためのリチウムポリマー電池が，胴体下の中央部分に前後輪およびサーボモータ用の単3乾電池6本が装着されている．また，草丈が長い路面に対しても走破できるよう，スパイクをとがった形状の物に変更した．図4.2の中央部にある基板に搭載したセンサ，モジュールについては3章システム図にて後述する．図4.2の下部の小型タイヤは，本ミッションの前輪の故障時にも対応できるように駆動力を持つほか，CanSatのバランスを取るためのスタビライザであり，CanSatの重心を後方に保持し，走行中にCanSat全体が回転するのを抑えるはたらきがある．また，走行性を向上させるために星形の各辺を固めの白いスポンジで補強している．高さの計測のためCanSatの上面にメジャーをあてた画像を図4.3，直径の計測のためCanSatとパラシュートをキャリアに収納した状態でメジャーをあてた画像を図4.4として示す．なお，キャリアの内径は146[mm](直径)である．

また，表4.2にはCanSat全体の大きさを示してある．

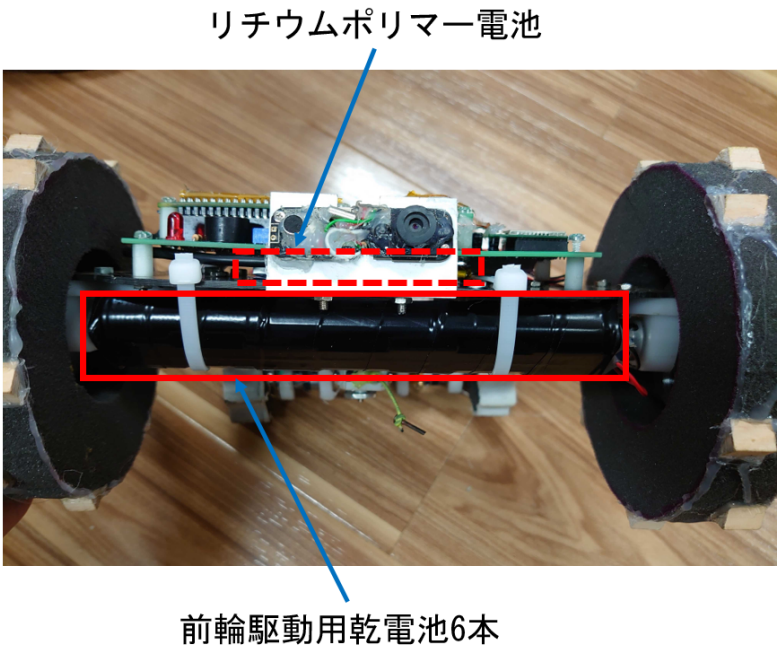


図4.1 CanSatの外観(正面)

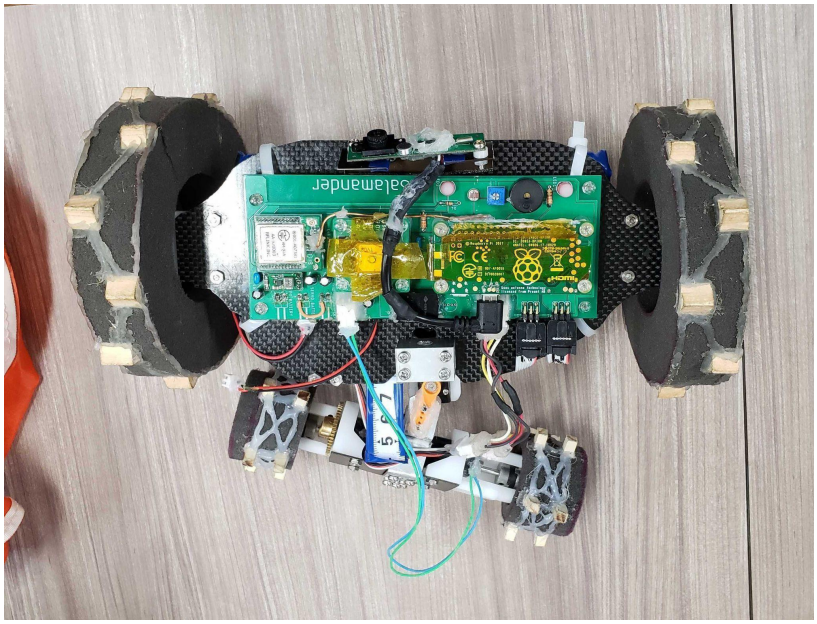


図4.2 CanSatの外観(上部)

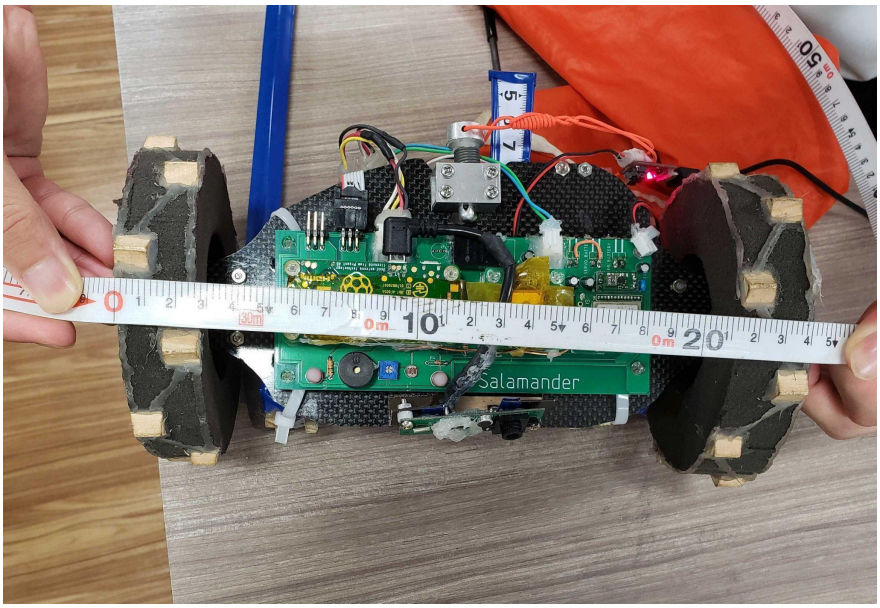


図4.3 CanSatの高さ

図4.4 CanSatの直径

表4.2 CanSatの大きさ

高さ [mm]	238
直径 [mm]	126

2. CanSat内観・機構

1. スタビライザ

図4.5と図4.6にCanSatのスタビライザの画像を示す．また，図4.7と図4.8にCADで作成したスタビライザの画像を示す．本ミッションでは前輪の空転によって生じるCanSatの走行性能の低下を防ぐために，スタビライザとして2つのサーボモータを図4.5の赤線で囲まれた部分のように装着した．これによって，スタビライザの角度は垂直方向のみに限定されず，GAによるCanSatの方向制御に寄与する水平方向に対しても変更可能であるため，前輪のうち1つが空転する状況においてもスタビライザの水平方向を適切に調整することで進行方向の制御が可能となる．また，前輪以外の駆動力を持たせるため，スタビライザの接地部はタイヤ型のものを採用した．タイヤが回転することによって，スタビライザに装備された後輪と地面との間に摩擦による摩擦力が進行方向逆向きに発生する．この摩擦力に対する反作用が進行方向に対する駆動力として獲得できる．なお，前輪と同様に後輪の形状を正円に設計した場合，スタビライザ収納時に後輪が前輪駆動用モータに干渉するという問題が発生する．この問題の回避およびタイヤ直径の拡張を達成するため，後輪タイヤの形状は星型八角形に設計した．図4.9と図4.10に後輪の画像を示す．

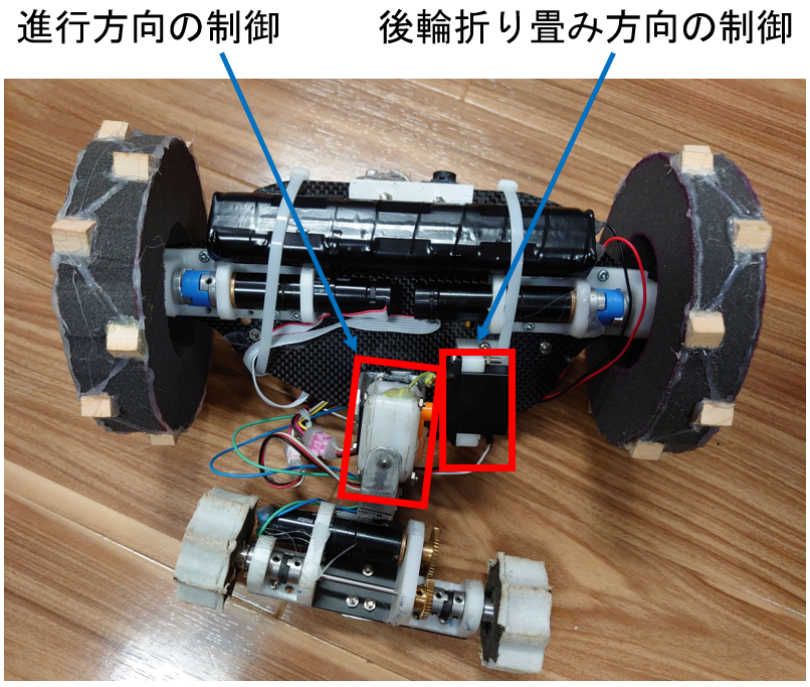


図4.5 スタビライザ（上面）

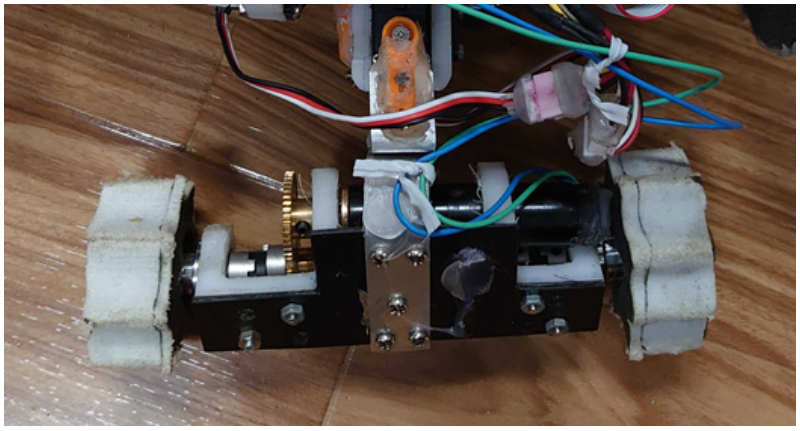


図4.6 スタビライザ（下面）

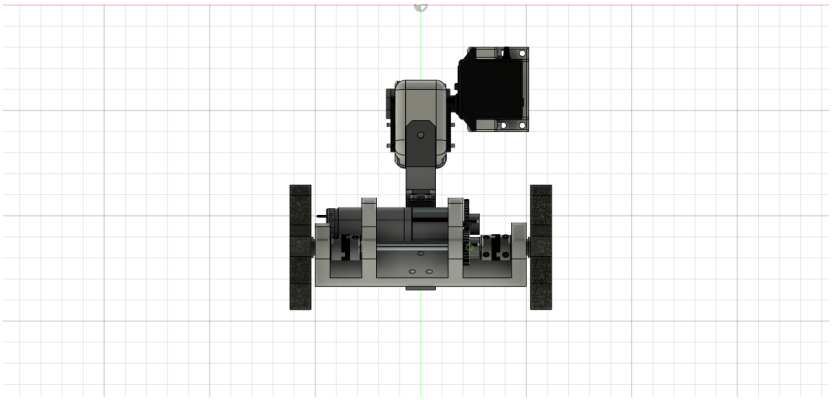


図4.7 スタビライザCAD画像（下面）

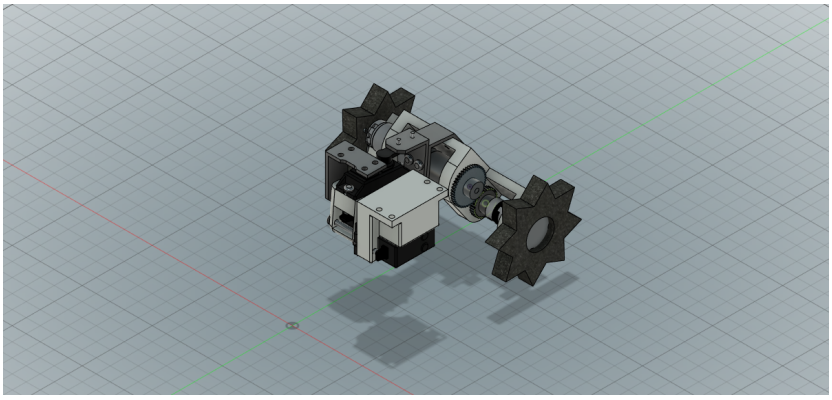


図4.8 スタビライザCAD画像（上面）

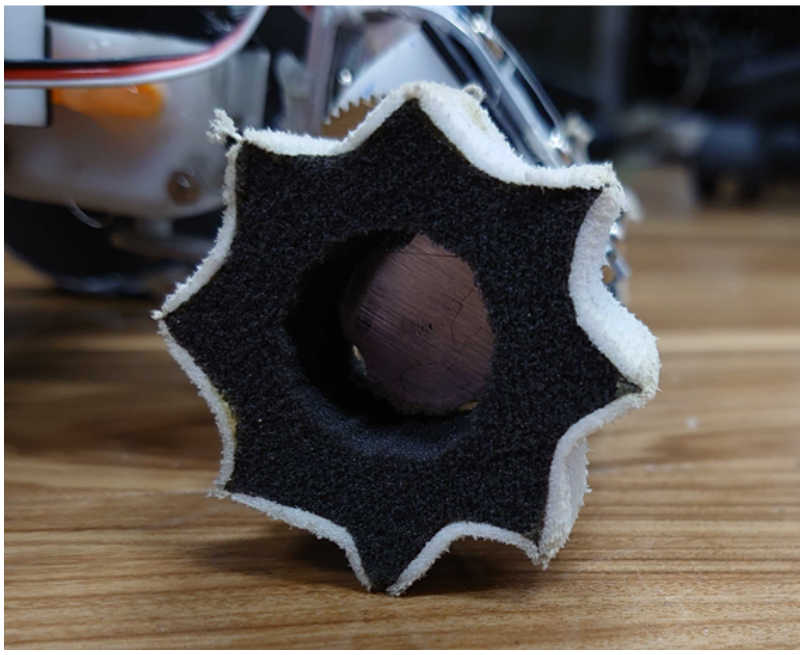


図4.9 後輪タイヤ

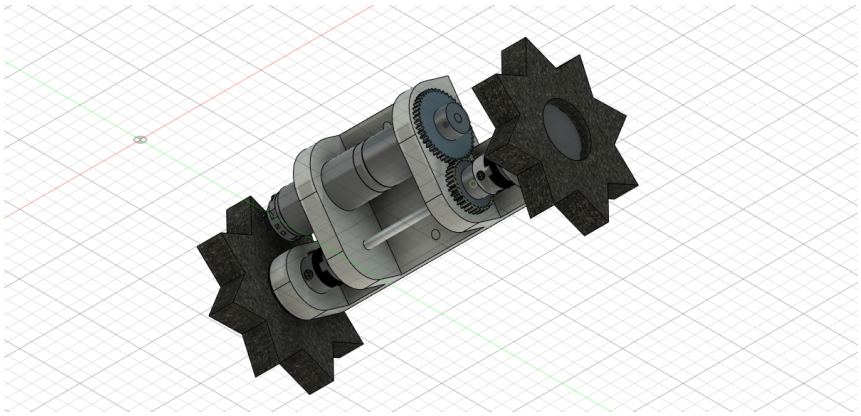


図4.10 後輪タイヤCAD画像

なお，本スタビライザ(以下，旧スタビライザと呼ぶ)を搭載したCanSatは，走行性能において大きな欠陥があることが我々の事前実験によって判明した．具体的には，旧スタビライザの(i)後輪タイヤの接地面における静摩擦・動摩擦係数が著しく小さく，(ii)後輪タイヤの接地面にて発生する垂直抗力が，前輪タイヤのそれよりも小さいという性質により，草地のような不整地で外乱が大きい環境下においては，後輪タイヤのみ頻繁に空転する状況が生じた．この状況に陥ることの回避を目的に，我々は新たなスタビライザ(図4.11，以下，新スタビライザと呼ぶ)を開発した．新スタビライザの機構のほとんどは旧スタビライザを継承しているが，後輪タイヤが星型八角形から，木材スパイクを装着した円形タイヤ(図4.12)に変更されている点のみ大きく異なる．この改良により，後輪タイヤと接地面の間に生じる摩擦力，すなわち，本CanSatの後輪タイヤによる駆動力の大幅向上によるCanSat全体の走行性能の向上が(V14)走行性能確認試験にて示された．また，モーターと車軸カップリングの連結部分を後輪タイヤのスポンジで覆うことで (図4.15)，走行時にフィールド上の草が連結部分に巻

き込み，走行性能を大幅に低下させる状態を回避することができる．以上より，本ミッションでは新スタビライザを搭載したCanSatを使用する．

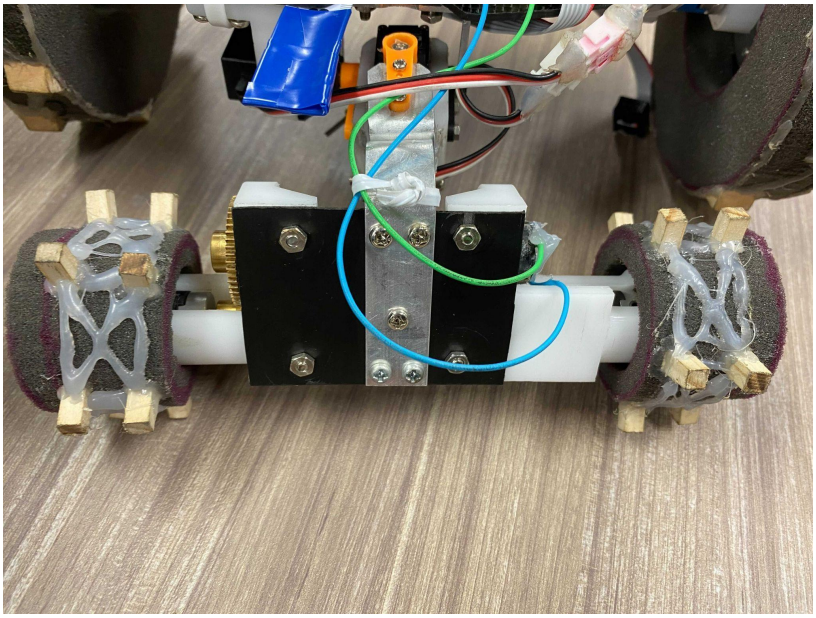


図4.11 新スタビライザ

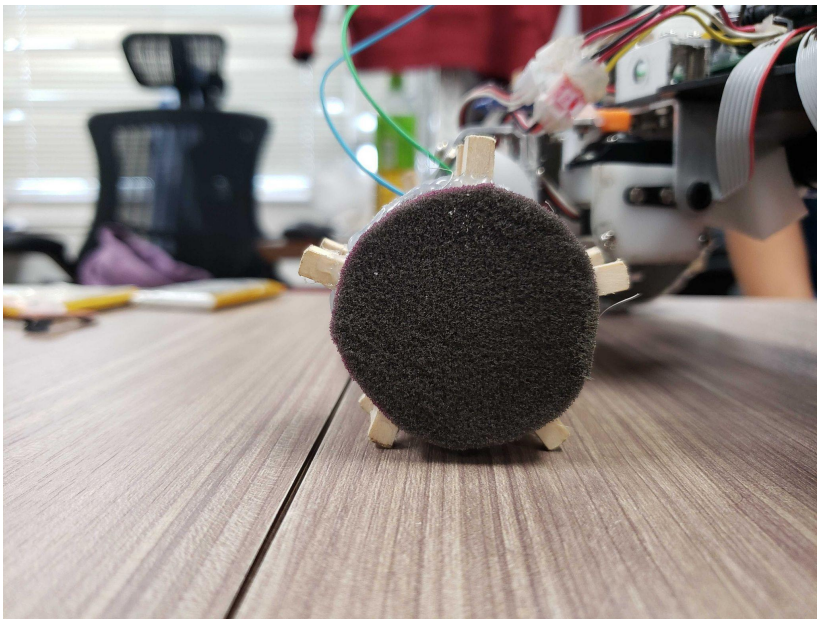


図4.12 木材スパイクを装着した円形タイヤ

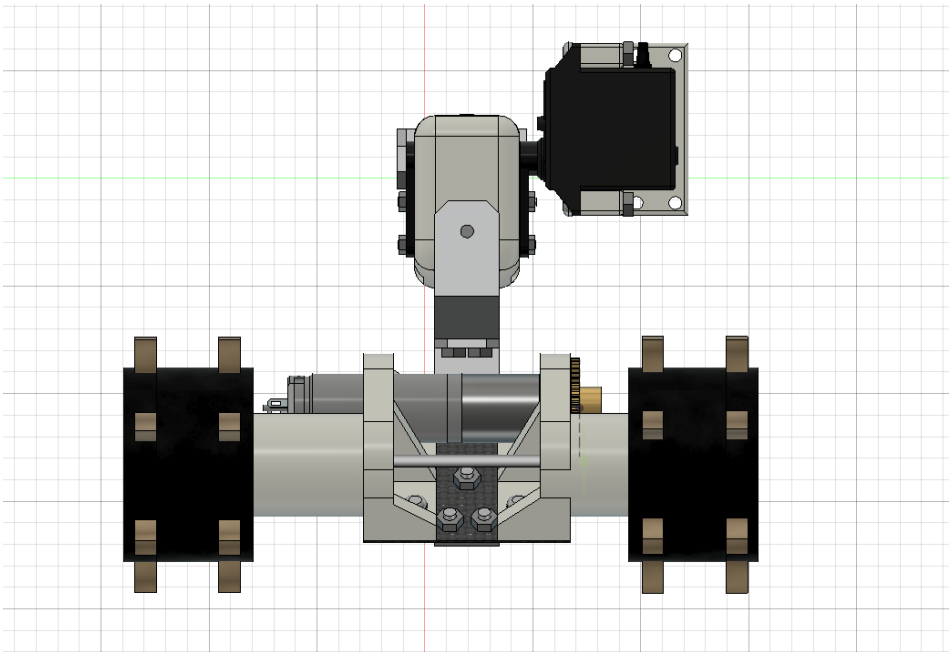


図4.13 新スタビライザCAD画像（下面）

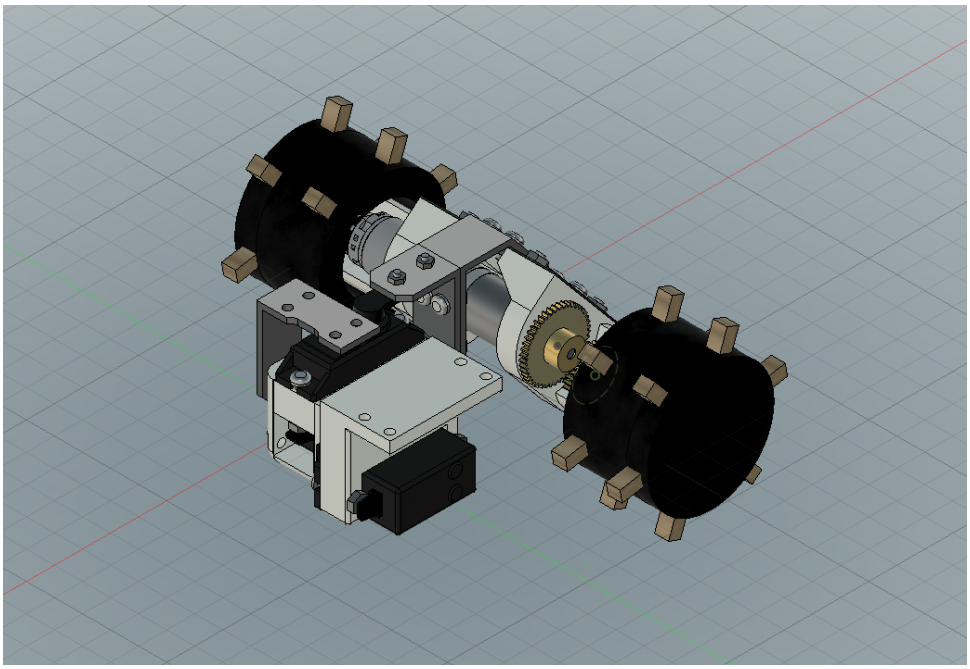


図4.14 新スタビライザCAD画像（上面）

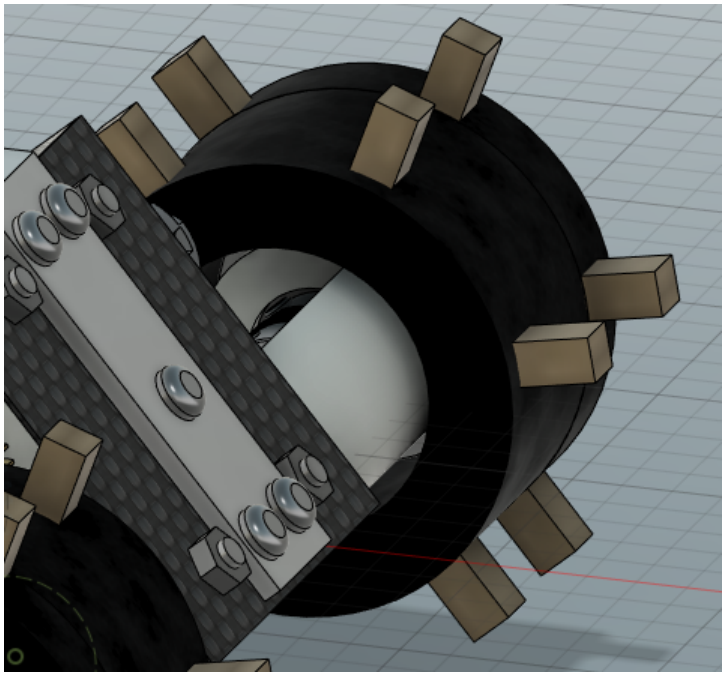


図4.15 後輪タイヤと車軸カップリングの連結部分

2. パラシュート切離し機構

図4.16，図4.17はCanSatのパラシュート切り離し機構の画像である．図4.15はパラシュートがCanSatと接続された状態，図4.17はパラシュートが切り離された状態を示す．なお，パラシュートは図4.16の橙色の紐の先に接続されている．

パラシュートとCanSatを接続するピンには図4.18で示される位置に穴が開いており，ここに釘を差し込むことでピンとCanSatが接続された状態（図4.16）を維持している．CanSat背面に装着されているサーボモータを駆動させると，紐が引っ張られることで釘が走行方向とは反対側に外れ，パラシュートのピンがCanSatから切り離される．ピンには切り離しをスムーズにするためにばねが設置されており，釘が外れた際にはこのばねが伸びることでCanSatからピンが勢いよく切り離される．

なお，この設定は経験上，3秒間サーボモータを上下に駆動し，CanSat前進，という動作を1,2回程度繰り返すことで抜けることが分かっている．フローチャートではより確実にピンが抜けるよう，上記動作を10回繰り返すこととしている．10回という数字は，不確実性のある動作の確実性を上げるために，動作が成功すると期待される試行回数に安全率Sをかけた回数を設定するという機械工学の考えに基づき，鉄などの金属部品の繰り返し動作における安全率S=6をかけた値として設定した．

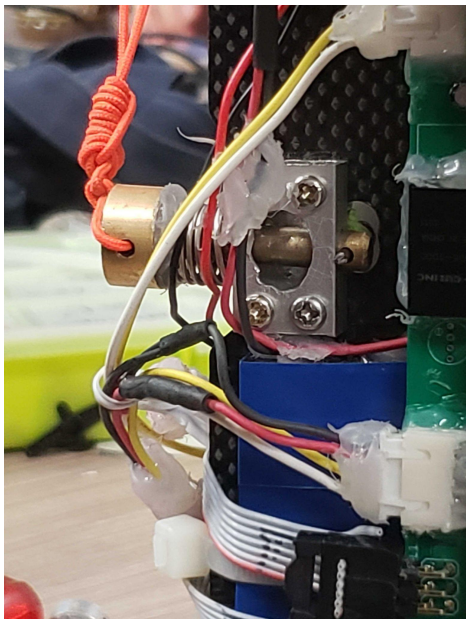


図4.16 パラシュート切り離し機構(装着済み)

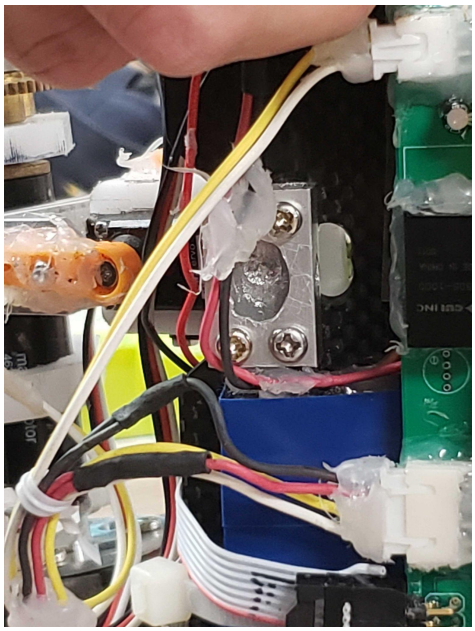


図4.17 パラシュート切り離し機構(切り離し後)

図4.18 ピンに開いている釘穴の位置

ここでパラシュートピンがばねの弾性力によって放出されることを数値計算によって確認する．パラシュートピンがばねの弾性力によって放出される条件は，下式の条件で表される．ここで，kはばね定数，Eは縦弾性係数である．なお今回使用しているピンの材質はばね鋼鋼材であり，JIS B 2704によれば E = 206 × 10^3[N/mm]である．

$$k < E$$

ここからは，今回採用したばねがこの条件を満たすことを確認する．まず，実際にパラシュートピンに使用しているばねのばね定数 k は次の関係式をもとに求めることができる．左辺はばねの弾性エネルギー，右辺はばねの位置エネルギーの差分を示す．

$$\frac{1}{2}\mathrm{k}x^2 = \mathrm{mg}\{h - (l - x)\}$$

ここで，l はばねの自然長[mm]，h はピンの長さ[mm]，x は設置時のねじ先端からのばねの変位[mm] である．図4.19，図4.20 は，これらの寸法を実際の画像上に示したものである．

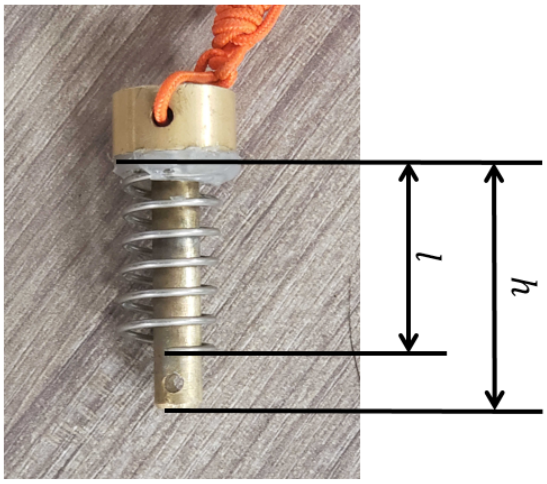


図4.19 ばねとピンの寸法

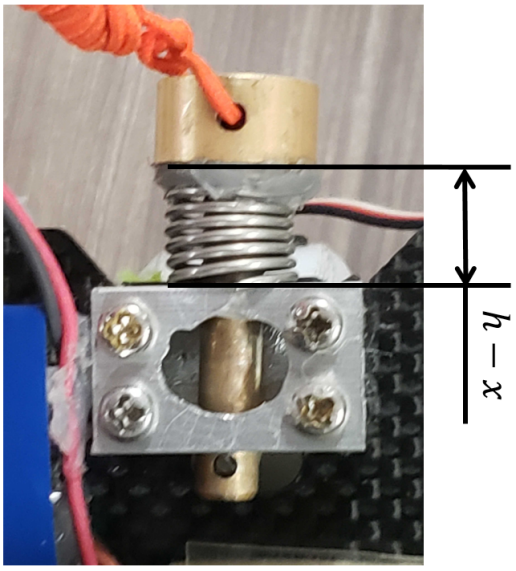


図4.20 縮めた時のばねとピンの寸法

実測より，$h = 30$ [mm] ，$l = 24$ [mm]，$x = 10$ [mm]，$m = 7 \times 10^{-3}$ [kg]であった．なお，重力加速度gは9.8 [m/s^2]とする．これらの値を上式に代入すると，$k = 22 \times 10^{3}$ [N/mm] と求められる．したがってk <E = 206 × 10^3 [N/mm]であり，最初に述べたパラシュートピンがばねの弾性力によって放出される条件$k < E$を満たしていることが確認された．

また，パラシュート切り離し時に釘に加わるトルクはサーボモータから発生する53N(≒5.4kgf)である(サーボモータのデータシートを参照)．また，JIS規格より釘の素材であるSUS304を参照すると釘の強度σは205[N/mm^2]であることが示されている．ここで，サーボモータの回転によって釘が引っ張られる際に力が加わる面積*S*はサーボモータと釘をつなぐ紐の釘に対する接地面の半分(力が加わる方向と真反対の部分)であるので，釘の胴径*2r*が1.25mm，紐の幅*w*が2mmであることを踏まえると以下の式より求められる．

$$S = \frac{2\pi rw}{2} = 3.14 \times 0.625 \times 2 = 3.925mm^2$$

以上よりパラシュート切り離し時に加わることが想定される釘の面の耐力σ'は

$$\sigma' = S\sigma = 3.925 \times 205 = 804.6\mathrm{N}$$

となる．したがって，使用する釘はパラシュート切り離し時に加わるトルクに対して変形することなく耐えることが可能であり，パラシュート切り離し機構に採用するものとして適切であるといえる．

3. システム図(CanSat搭載計器仕様一覧)

図4.21に本CanSatを制御するシステムの構成を示す．赤色の矢印は電力の供給を示し，青色の矢印は制御に必要な情報の通信を示している．CanSatには9軸センサ(加速度，ジャイロ，地磁気)と気圧センサ，光センサ，GPSセンサが搭載されており，これらのセンサとモータに付属のモータエンコーダから得た情報からモータやサーボモータの動作を制御する．また，ロスト対策としてLoRaモジュールを通してCanSatの位置情報を地上局に適宜送信する．使用する各センサやモジュールの名称および使用目的を表4.3にまとめ，その詳細を以下に示す．

使用電源は，下記のようになっている．

- 乾電池(モータ・サーボモータ用電源)
  - 品名: Energizer Ultimate Lithium AA Batteries
  - 型番: 12-2037
  - 備考
    - 6本を直列(約9～10V)
    - サーボモータに対しては，必要な電圧が5Vであるため，スーパー三端子レギュレータを用いて降圧する．
- リチウムイオンポリマー電池(回路用電源)
  - 品名: リチウムイオンポリマー電池3.7V 860mAh
  - 型番: DTP603048(PHR)
  - 備考
    - Raspberry Piに必要な電圧が5Vであるため，昇圧型DC-DCコンバータモジュールを用いて昇圧する．

また，本ミッションでは，環境情報を取得する各センサ(入力)について，以下のように用いる．

- GPSセンサ

    - 自己位置推定
        - 取得したGPS座標を用いて自己位置座標を更新
    - ナビゲーション処理
        - ゴール座標と毎回取得するGPS座標の差分から進行方向を調整
    - スタック判定
        - 毎回取得するGPS座標の差が一定以内ならばスタック状態と判定
- 9軸センサ
    - 着地判定
        - 落下中と着地後の加速度が異なることを利用
    - スタック判定
        - 進行方向の加速度の変化を使用
    - ナビゲーション処理
        - 地磁気を用いた進行方向の調整
    - 横転・反転判定
        - 重力の方向から，CanSatの状態を把握
    - 姿勢制御
        - 重力の方向から，走行時の姿勢を制御
- 気圧センサ
    - 着地判定
        - 落下中に気圧が大きく変化し，着地時後は気圧変化が小さいことを利用
- 光センサ
    - キャリア放出判定
        - ドローンのキャリアから放出されたことを判定
- LoRa
    - ロスト対策
        - GPSセンサが取得した自己位置座標を定期的に発信
- webカメラ
    - 物体検知
        - 周囲の情報を画像データとして取得，演算し物体を検知
- モータエンコーダ
    - 故障判定
        - タイヤの回転数を取得して，モータの故障を検知

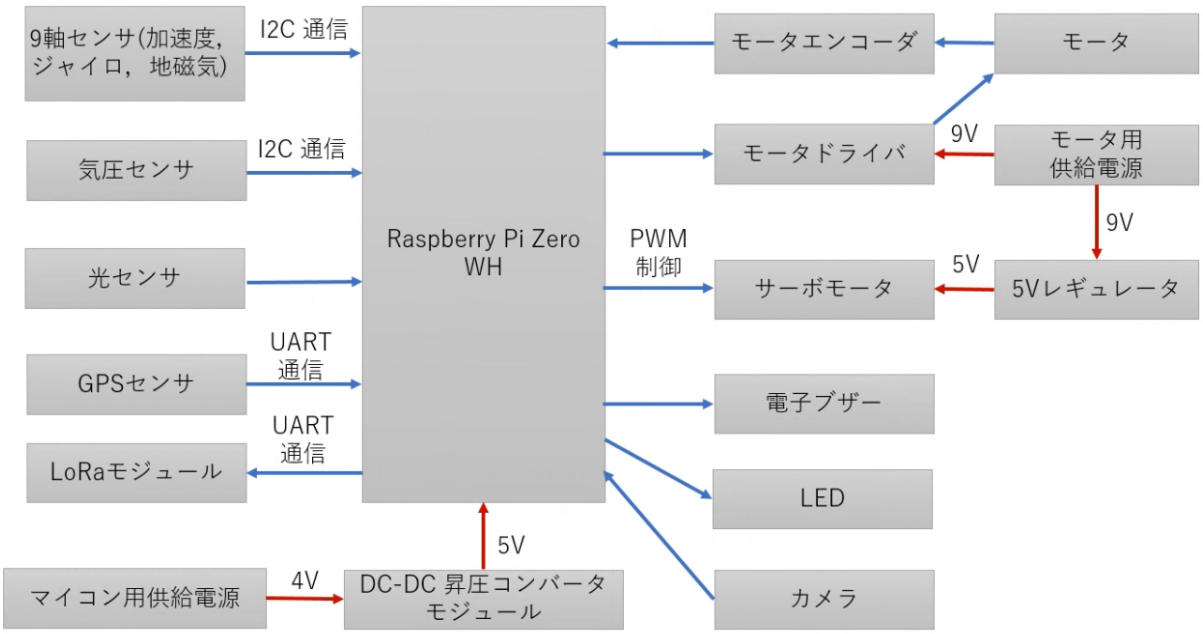


図4.21　CanSatのシステム図

表4.3　搭載計器使用

分類	名称・型番	使用目的	入手先URL・参考情報等
マイコン	Raspberry Pi Zero WH	入出力値の計算，ログデータの記録	https://www.switch-science.com/catalog/3646/
GPSセンサ	AE-GYSFDMAXB	ゴール地点への方向・距離の取得，スタック判定	https://akizukidenshi.com/catalog/g/gK-09991/
モータドライバ	DRV8835	モータの出力制御	http://akizukidenshi.com/catalog/g/gK-09848/
モータ	DC motor RE EBCLL 3.2W SL 2WE	タイヤの回転	https://www.maxongroup.co.jp/maxon/view/product/motor/dcmotor/re/re16/118699
モータエンコーダ	エンコーダ MR, Type M, 32 カウント/回転, 2 チャンネル	タイヤの回転数の測定，故障判定	https://www.maxongroup.co.jp/maxon/view/product/sensor/encoder/Magnetische-Encoder/ENCODERMR/ENCODER-MR-TYPM-32IMP-2-3K

			ANAL/201935
サーボモータ	GWSMIRMGFA S03T/2BBMG/J	パラシュート切り離し，CanSatの姿勢が水平になるように制御，後輪部の向きの制御	http://akizukidenshi.com/catalog/g/gM-01908/ https://akizukidenshi.com/catalog/g/gM-01794/
モータ用供給電源	Energizer L91Ultimate Lithium AA Batteries ×6本	駆動系(サーボ，モータ)への電源供給	https://www.walmart.com/ip/Energizer-Ultimate-Lithium-AA-Batteries-12-Pack-Double-A-Batteries/139060065
マイコン用供給電源	DTP603048(PHR)	マイコン及びセンサ類への電源供給	https://www.marutsu.co.jp/pc/i/836350/
DC-DC昇圧コンバータ	TPS63070	マイコン用の供給電圧を5Vに昇圧	https://strawberry-linux.com/catalog/items?code=16370
スーパー三端子5Vレギュレータ	V7805-1000	サーボモータ用の供給電圧をサーボモータの許容電圧範囲まで降圧する	http://akizukidenshi.com/catalog/g/gM-06350/
9軸センサ	モジュールモデル: GY-9250, チップ: MPU-9250	加速度(横転，反転判定のため)，地磁気(方位推定の制御履歴のため)，ジャイロ(着地判定のため)の値取得	https://www.amazon.co.jp/KKHMF-MPU-9250-9アクシスセンサーモジュール-GY-9250-Arduino用/dp/B019C0MHOU/ref=sxbs_sxwds-stvp
気圧センサ	AE-BME280	着地判定	http://akizukidenshi.com/catalog/g/gK-09421/
電子ブザー	PB04-SE12HPR	光検知中であるかどうかの把握	http://akizukidenshi.com/catalog/g/gP-04497/
光センサ	MI527/MI5527	キャリアからの放出判定	http://akizukidenshi.com/catalog/g/gI-00110/
LED	OSTA5131A-R/PG/B	実行中のシーケンス(4章を参照)の把握	http://akizukidenshi.com/catalog/g/gI-02476/

LoRa通信 モジュール	RM92-AN	ロスト対策としての地上局との長距離通信 自己位置座標を地上局へ送信(ダウンリング)	http://www.rflink.co.jp/body_seihin_musen_920_RM92A.html
カメラ	UCAM-C310FBBK	ゴールコーンの検知	https://www.amazon.co.jp/エレコム-WEBカメラ-500万画素-高精細ガラスレンズ-UCAM-C750FBBK/dp/B078JBBJ73/ref=sr_1_1_sspa

4. アルゴリズム

アルゴリズムのフローチャートを図4.22に示す(後の詳細事項はフローチャート内の番号に対応している).

1. プログラム全体のフローチャート

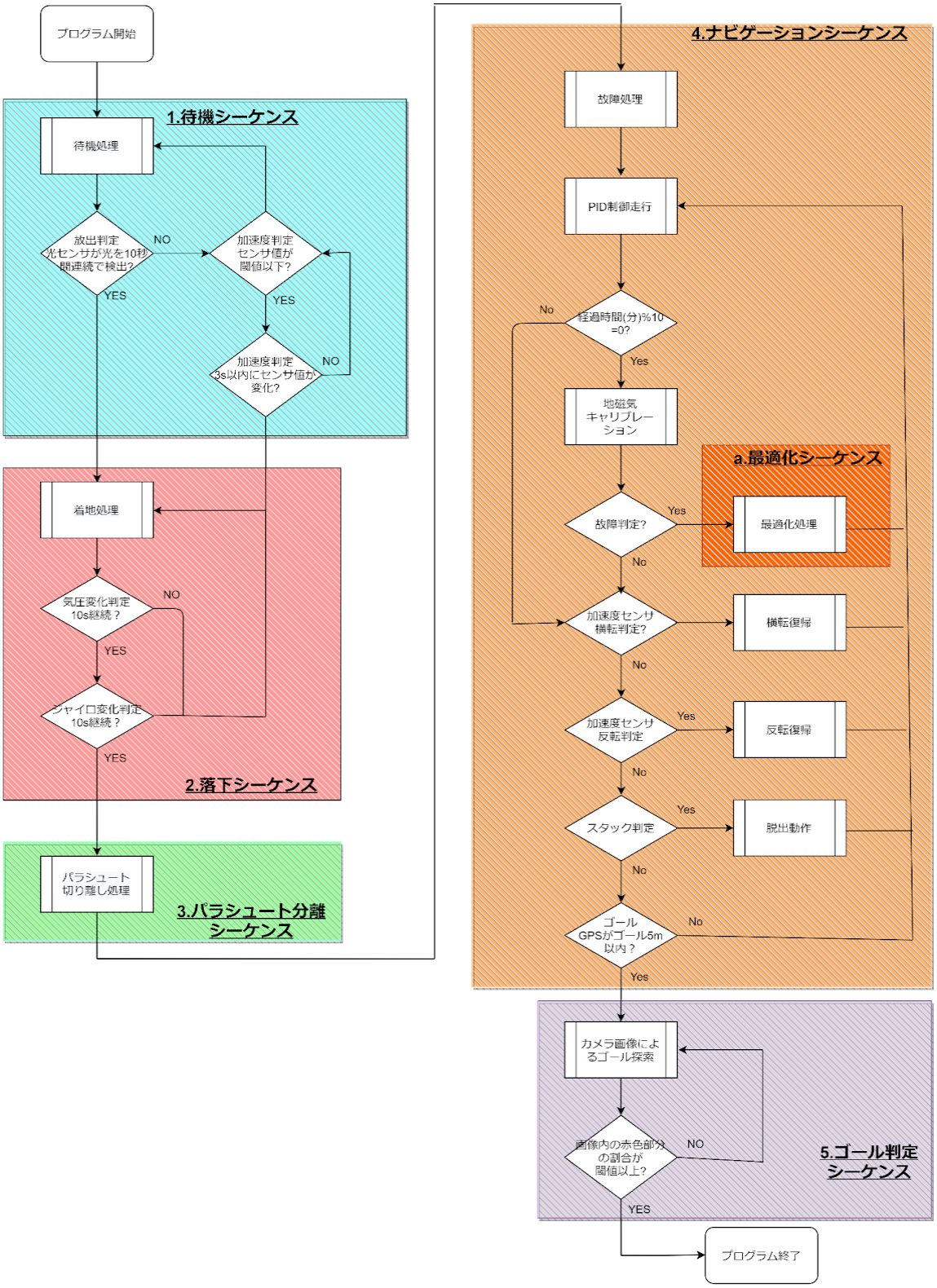


図4.22　アルゴリズムのフローチャート

1. キャリア搭載からCanSat放出までの間はCanSatを待機させる必要があるため，プログラム開始からは待機状態とする．光センサが光を検出している状態を10秒継続した場合はCanSatが放出されたと判断し次のフローに移る．
   光センサの異常により，光を検出しない場合のために，以下の条件を同時に満たした場合はCanSatが放出されたと判断し次のフローに移る．

   条件A: 加速度センサの値のがしきい値以下
   - ｜加速度センサxyz軸のL2ノルム値｜< 0.8 [deg/s]

   条件B: 条件Aを満たしてから3秒以内に加速度センサ値が5.0 [G]を観測する．
   ※単位Gは重力加速度（ ≒ 9.81 [m/s^2] ）

2. 落下中であることの判定のため，以下のすべての条件を同時に満たした状態が一定時間継続するまで待機する．

   - 条件1: 気圧センサの値の変化量がしきい値以下
     - ｜気圧センサの値の変化量｜< 6.0 [hPa] を10秒継続．
   - 条件2: ジャイロセンサの値の絶対値がしきい値以下
     - 以下の3つの条件を同時に10秒継続．
       - ｜ジャイロセンサx軸の値｜< 35.0 [deg/s]
       - ｜ジャイロセンサy軸の値｜< 35.0 [deg/s]
       - ｜ジャイロセンサz軸の値｜< 35.0 [deg/s]

   条件を満たした場合，着地したと判断して次のフローに移る．

3. サーボモータを駆動し，パラシュート接合部の切り離しを行う．
   具体的には，3秒間サーボモータを上下に駆動し，CanSat前進．これを6回繰り返す．
   終了後，次のフローに移る．

4. GPS座標(現在地およびゴール地点)と地磁気センサによるCanSat前方方向の情報から、CanSatの進むべき方向を決定するPID制御によってCanSatの向きを修正しながらゴール方向へと走行する．GPS座標によりCanSatがゴール5m以内へ接近したと判定した場合，次のシーケンスへ遷移する．

   また，このシーケンスを開始した直後および10分経過ごとに，地磁気のキャリブレーションをする．キャリブレーションの後半では，加速度センサから得られる値をもとに故障判定を実行する．具体的な判定条件は以下の通りである．

   - 故障判定:

   この判定は、進行方向が制御可能か否かを識別する役割を担う．故障部分の判定は以下の通りに行う．

- 片前輪のモータ出力をしている状態で左/右輪のモータエンコーダの値が更新されない場合，左/右輪のタイヤの空転・固定やモータの故障が発生していると判定

また，走行中は加速度センサから得られる値をもとに故障判定・横転判定・反転判定を，GPSセンサから得られる値をもとにスタック判定を実行する．それぞれの具体的な判定条件は以下の通りである．

- 横転判定:

x軸(左右方向)の加速度ノルム > 0.65 [G]

- 反転判定:

z軸(基板鉛直方向)の加速度センサ値 < 0.8 [G]

- スタック判定:

GPSによって計測された30秒間の移動距離 < 5[m]

故障判定された場合はミッションを一時中断してPID制御の最適化処理を実行する．
横転，反転判定された場合はミッションを一時中断してタイヤとサーボモータによる復帰動作を実行する．

タイヤの故障には，タイヤの空転，タイヤのノイズ等が考えられ，それぞれ，タイヤの出力を0にすること，各フレームごとにタイヤ出力にランダムなノイズを付与することによって再現することが可能である．
このうち，本ミッションでは，片方の前輪のモータ故障を想定する．
これを再現するために，走行開始(0秒経過)時点で左の前輪のモータ出力を0にする処理を実行する．なお，この処理は故障判定には依存しない．

5. CanSatはカメラに映る赤い物体の割合（赤色領域の割合）R_{red}（後述）を検知し，割合が高くなる方角へ旋回・直進を繰り返しながら赤い物体の割合が増える方向へと進行方向を制御する．R_{red}が閾値θ_{red}を超えた場合，その位置をゴールと判定しミッションを終了する

   カメラに映る赤い物体の割合R_{red}の定義は以下の通りである．

   HSV色空間において、
   - 色相Hが0以上30以下または330以上360以下
   - 彩度Sが90以上255以下
   - 明度Vが75以上295以下

   を満たすピクセルの割合．

   例えば，$\theta_{red} = 20\%$に設定した場合，カメラに映る赤い物体の割合R_{red}が20%を超えていた時に，CanSatはゴールに到達したと識別する．ここで，θ_{red}はCanSat投下前に変更可能なハイパーパラメータであり，CanSatに差し込む外光の光量の大きさに応じて調整する必要があ

る. 具体例としては, 光量が大きい場合は画面が白飛びする可能性が高くなるためθ_{red}は低めの値に設定し, 光量が小さい場合は画面が黒く, 赤色領域を認識しなくなる可能性が高くなるため, θ_{red}は高く設定することが肝要である.

2. 最適化サブシーケンス

● 遺伝的アルゴリズム(GA)の概要

遺伝的アルゴリズム(Genetic Algorithm : GA)は生物の進化の仕組みを模した近似解探索アルゴリズムである．GAではある解の集合に対して環境や問題に対する適応度を導出し，適応度が良い解を次の世代に残していく処理を繰り返すことでより良い解を探索する．GAのアルゴリズムの概要を以下に示す．

1. 初期解をランダムに生成

初期母集団(第0世代の解集団)を生成する

2. 親の選択

母集団内の個体ごとに適応度といわれる指標を導出する．この適応度をもとに，後述する変化を適用する対象の解を選択する．(下図では、最大化を例に説明しており，適応度が最も高い2つの個体を選出している)

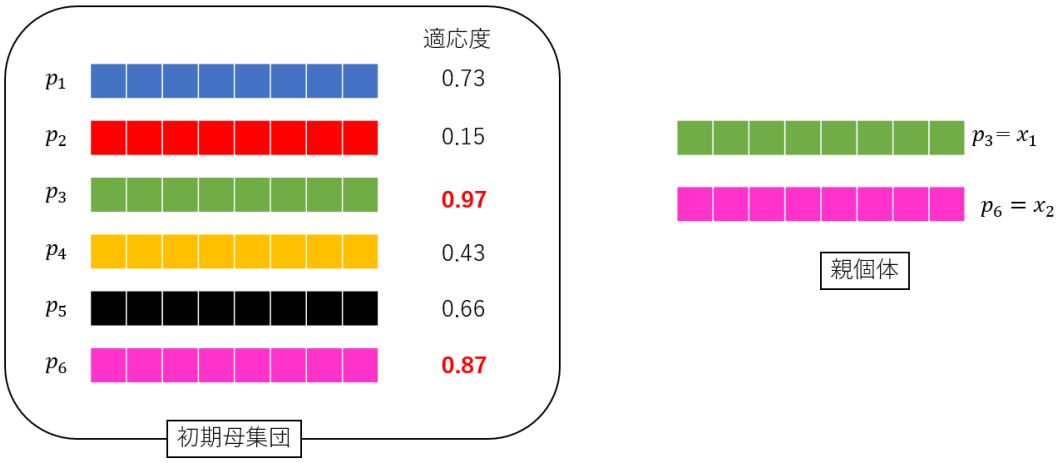


図4.23 親個体の選択

3. 子の生成

前の手順で選択された解を親とし，親に対して後述する処理を適用した異なる解を生成する．ここでは親に対する処理として代表的なものをいくつか紹介する．

変異：親個体を1つ選択し，ランダムに指定された要素に対応する実数値に一様分布に基づく微小な乱数値を加減する．

交叉：親個体を2つ以上選択し，特定のルールに基づいて各要素に対応する実数値をコピーする

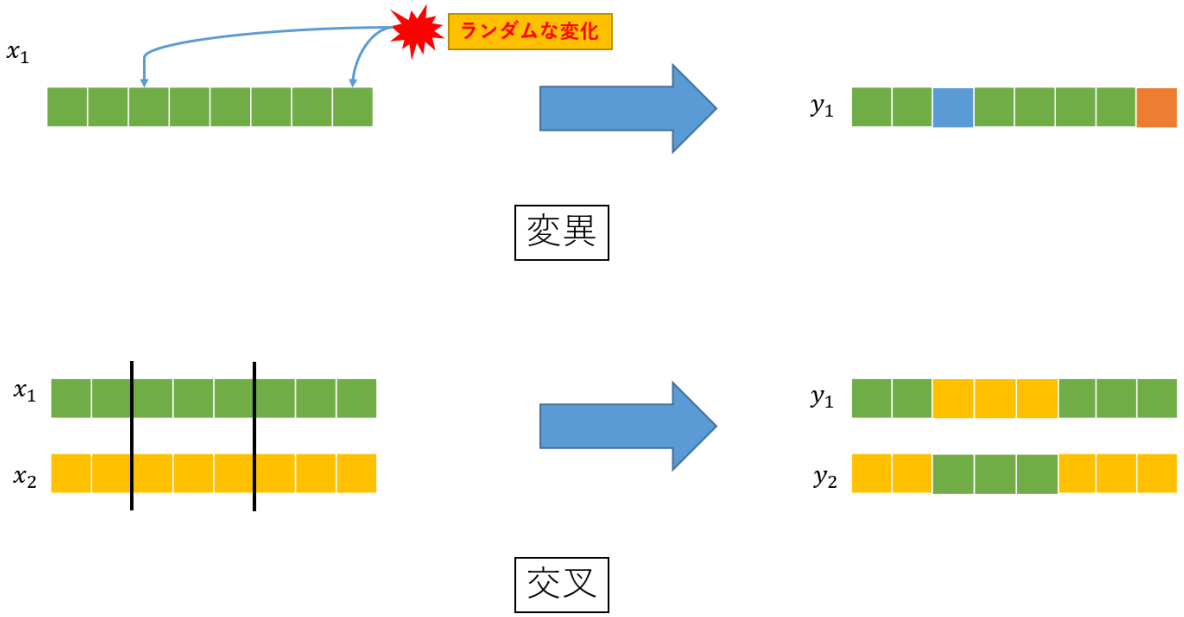


図4.24 変異と交叉

4. 個体選択

母集団と生成された子において，それぞれ適応度を導出し，これらの中からより良い解を選択し，次の世代に継承する．（ここでは，適応度が高い個体を次の世代に継承している．適応度が低い個体はこの世代で淘汰される）

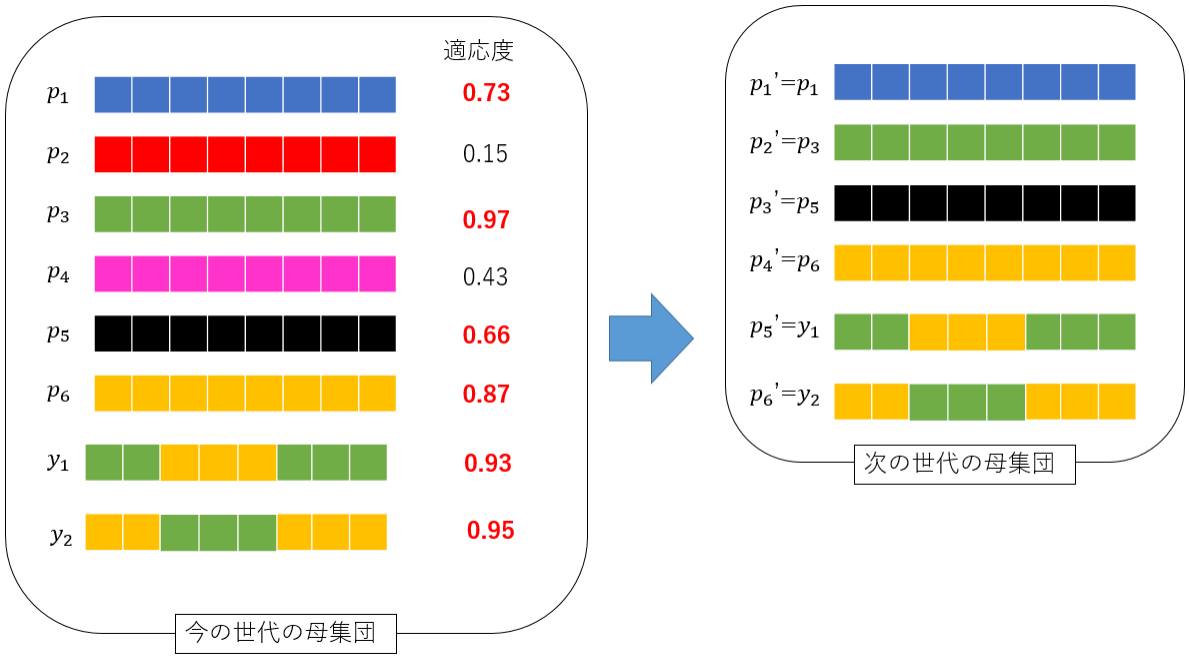


図4.25 個体選択

5. 終了条件判定

あらかじめ決めた終了条件を満たすか判定し，満たされた場合はそこで計算が終了する．満たされない場合は再度手順2に戻って次の世代の計算が開始される．

- PID制御

本ミッションではPID制御を用いて，ゴールまでのナビゲーション走行をする．

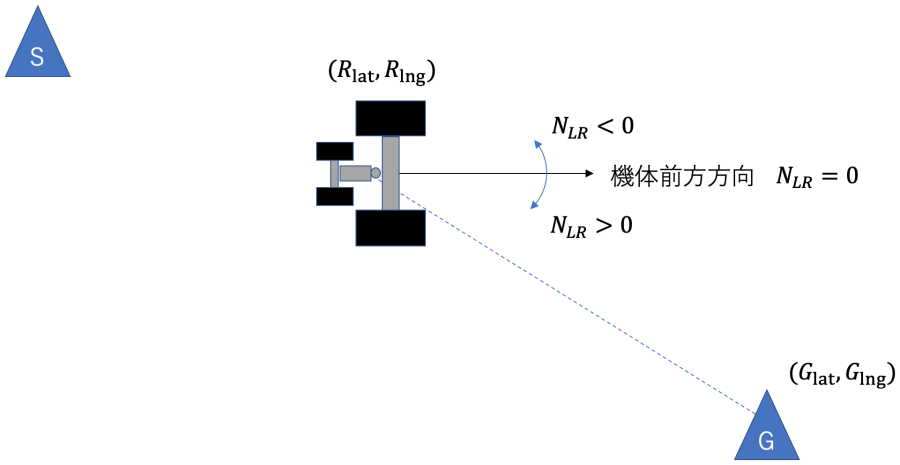


図4.26 スタートとゴールとCanSatの位置によるパラメータ

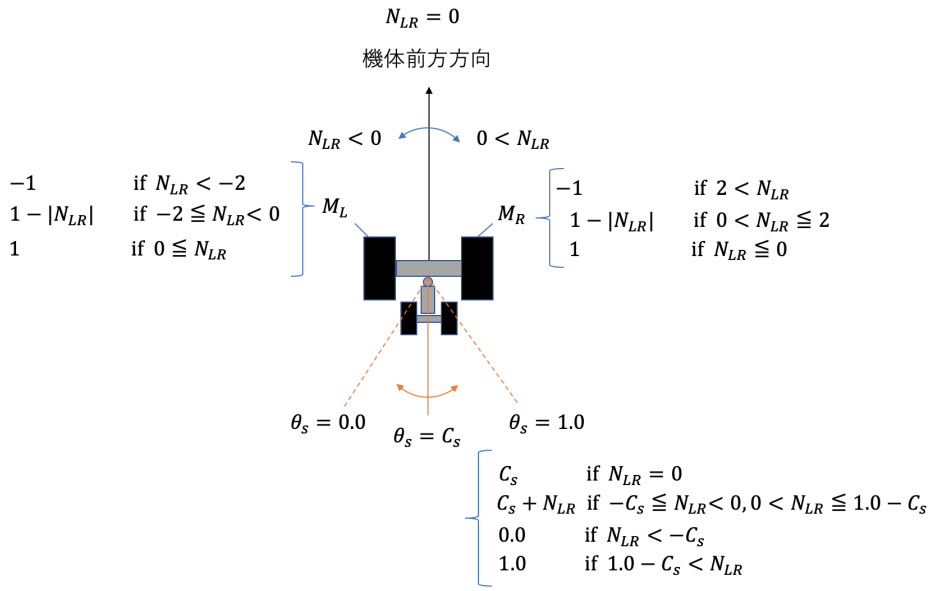


図4.27 CanSat前方と前輪モータ・サーボによるパラメータ

図4.26はスタート地点とゴール地点の間にCanSatがある様子を表している．GPSセンサか

ら取得したR_{lat}, R_{lng}はCanSatのGPS座標を表している. G_{lat}, G_{lng}はゴールのGPS座標を表している. CanSatは(R_{lat}, R_{lng})と(G_{lat}, G_{lng})からゴールへの方位角を算出. 9軸センサからCanSatのCanSat前方の方位角を取得する. ゴールへの方位角とCanSat前方の方位角をPID関数に入力し次にCanSatを制御すべき方向をN_{LR}として得る. $N_{LR} < 0$の時は左方向, $N_{LR} > 0$の時は右方向, $N_{LR} = 0.0$の時は前方へCanSatを制御する. つまり最終的に制御すべき角度が大きくなるほどN_{LR}の絶対値は大きくなり, 制御すべき角度が小さくなる(直進制御に近づく)ほどN_{LR}の絶対値は小さくなる(0.0に近づく). 図4.22は, 得たN_{LR}をCanSat制御に反映する様子を表している. M_L, M_Rはそれぞれ左と右のモータの出力を表しており, 後進状態の−1.0から, 停止状態の0を経て, 前進状態の1まで値を取る. また, M_LやM_RにN_{LR}をそのまま代入するが, N_{LR}の絶対値がM_{LMAX},M_{RMAX}を超えるとM_{LMAX},M_{RMAX}をそれぞれ代入する. 詳細の式は図の通りである. (M_{LMAX}, M_{RMAX}の初期値は1.0)θ_Sは横軸サーボの向きを表し, $\theta_S = 0.0$は左向き, $\theta_S = 1.0$は右向き, $\theta_S = C_S$は中心である. (C_Sの初期値は0.5である.) $N_{LR} = 0.0$をC_Sにずらし, 最大値1.0や最小値0.0としてまるめる.

PID制御補正

PID制御は, 基本的に左右前輪の”前進”出力を調整して方向制御をする. すると, 例えば左旋回する場合, 故障していない場合は右前輪の前進出力と左へ向けられた後輪の前進出力で制御することになる. しかし, この時右前輪が故障している場合は, 左へ向けられた後輪の前進出力でのみ制御されるため, 単純に力不足により制御が不能に陥る可能性があるため, 故障時に上記のような状況になった場合は, 故障していない前輪を後進させることで旋回が不要になる角度まで復帰する.

具体的には, 片前輪が故障検知時している状態のN_{LR}が閾値N_{th}(左前輪の故障時は$N_{th} < 0$, 右前輪の故障時は$N_{th} > 0$)を超えた場合は, 機体を徐々に後退させることで本来の制御すべき方向N_{LR}に戻す措置を取る. なお, この動作が発動している時は, 方向制御に特化し走行距離が著しく減少するため, この動作が発動しないように直進制御ができることが大切である.

- GAによるPID制御の最適化

ここで, 本ミッションで想定している片前輪モータの故障が発生すると$C_S = 0.5$の時にCanSatは前進することはなく, 故障した前輪の方角へ曲がる. そこで, 最適化処理のGAによって片前輪が故障した状態でCanSatが直進する横軸サーボの角度θ_Sと最適値を求め, C_Sと前輪モータの出力の最大値(左輪故障の場合は右輪のM_{RMAX})を, 設定し直すことでPID制御のN_{LR}が0.0に近い時の直進性を確保し, 左右方向への制御が可能になるように調整する.

図4.27は最適化に用いる個体の遺伝子および適応値を表した図である. 遺伝子は左前輪モータ出力, 右前輪モータ出力, 縦軸サーボモータ出力, 横軸サーボモータ出力から構成さ

れ，値の取る範囲はいずれも0.0~1.0である．最適化処理1回目の初期個体は下記で記述するシミュレーションであらかじめ生成した最適値の群を，2回目以降はそれまでの最適化処理で適応度が最小であった順に設定個体数分集め群を，代入し，設定したインターバルで母集団の中から順に個体を切り替えながら適応値を算出する．適応値の算出には，インターバル開始から終了までの機体が回転した角度(累積であり，つまり0度≠360度)を加算し，モータエンコーダから算出した進行した距離を減算する．この値が少ないと回転角度が小さく進んだ距離が大きい，つまり前進したことを意味する．よって，GAは最小化を目指し最適化する．

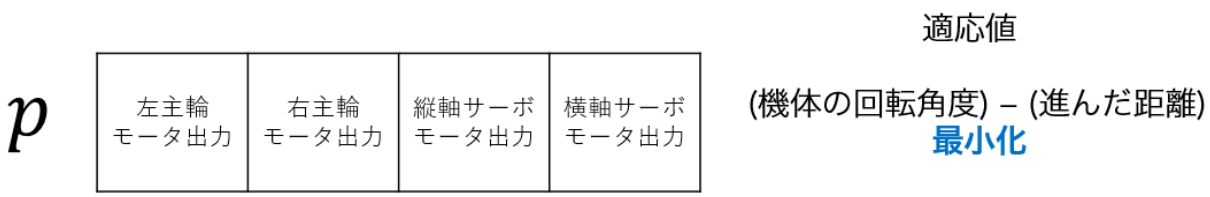


図4.28 本ミッションにおけるGAの個体の構成要素

- GAの初期個体をあらかじめ生成するシミュレータ

実機を用いた最適化は膨大な時間と電力が必要となる問題があるため，シミュレーション上の複数地形下でそれぞれ最適化した個体をまとめて初期個体として扱う．これによって，ミッション下での最適化に要する計算時間の大幅な削減が実現できる．図4.29は今回用いたシミュレーション環境であり，それぞれ異なる図4.23のような個体を持ったCanSatが動作している．シミュレーション環境はUnityで実装されており，CanSatはCADデータをそのまま利用している．地形は図4.30~図4.33で表されるようなパターンで用意している．それぞれの環境で初期個体数を30に設定し，1000世代で算出した個体群のうち，適応度が最も高い4個体を初期個体群として搭載する．また，GAはPython環境で実行されており図4.3のように情報を得る．このGAと同様のものがCanSatプログラム(実機の制御に関するもの)にも搭載されており最適化処理を成す．

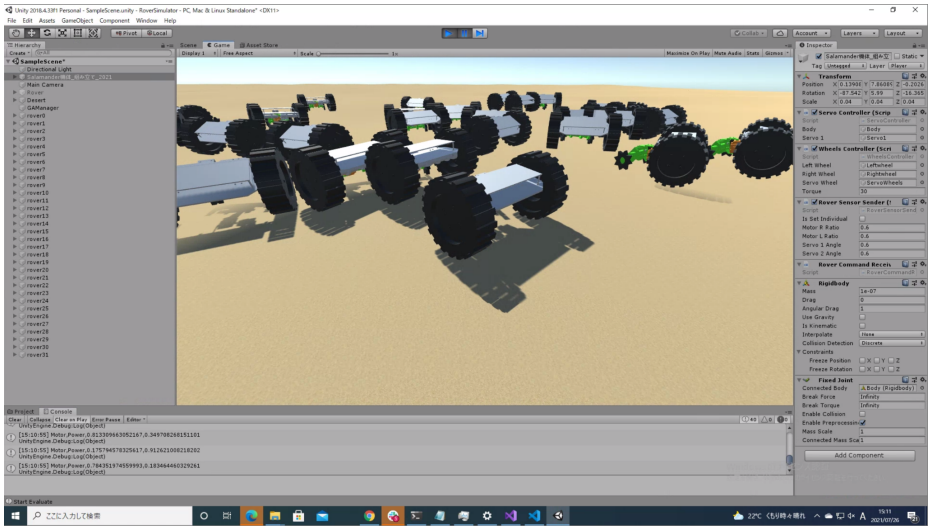


図4.29 シミュレーションの様子

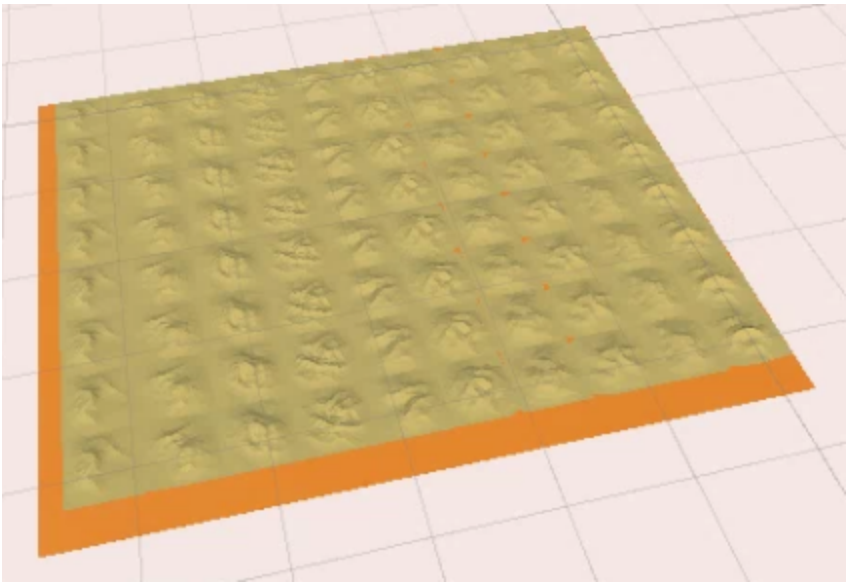


図4.30 シミュレーション環境の地形(砂漠1)

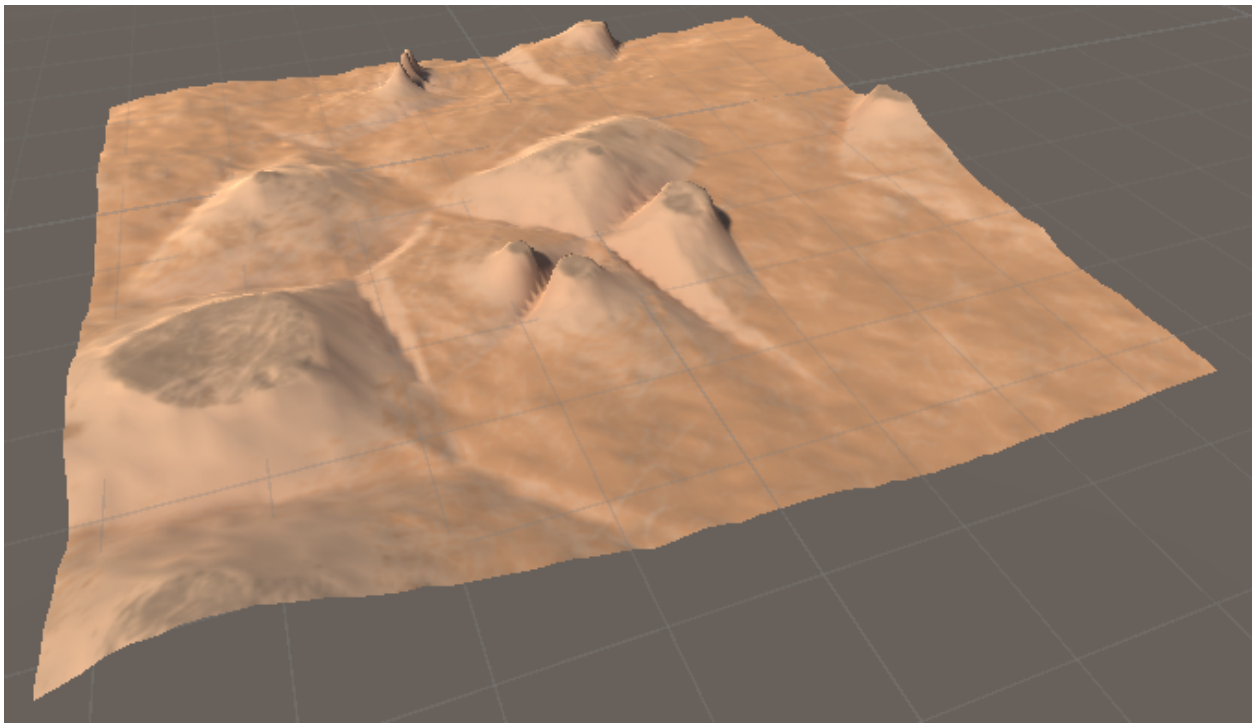


図4.31 シミュレーション環境の地形(砂漠2)

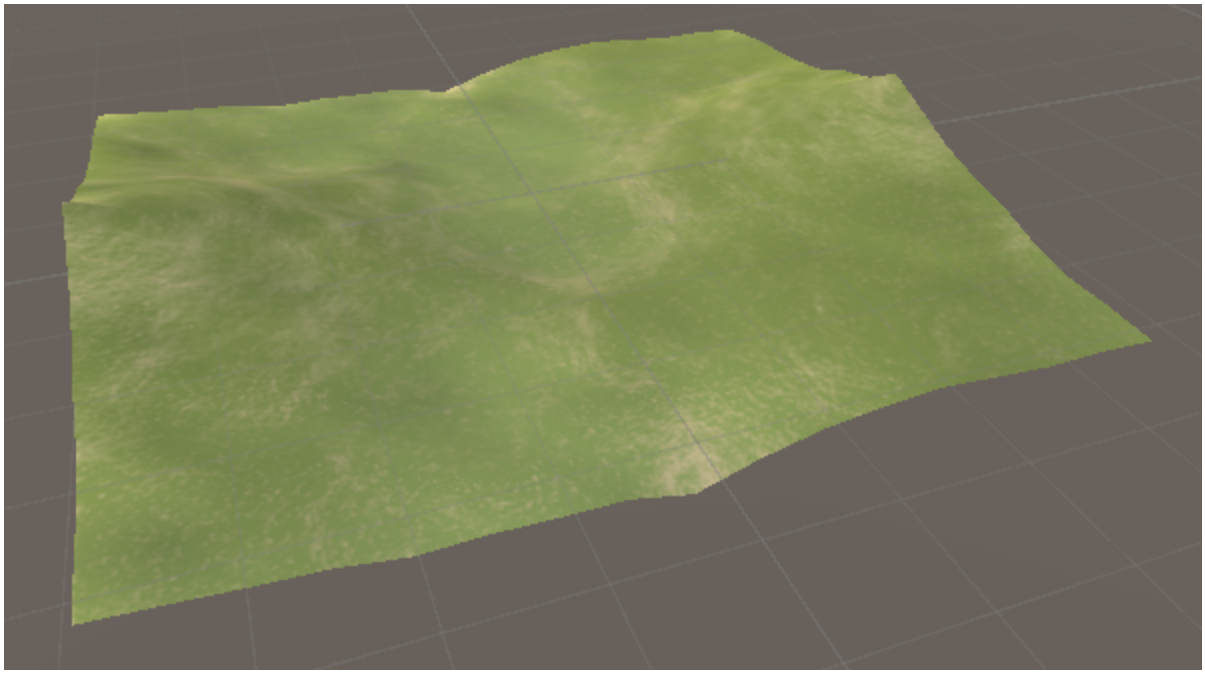


図4.32 シミュレーション環境の地形(ゆるい凹凸のある草原)

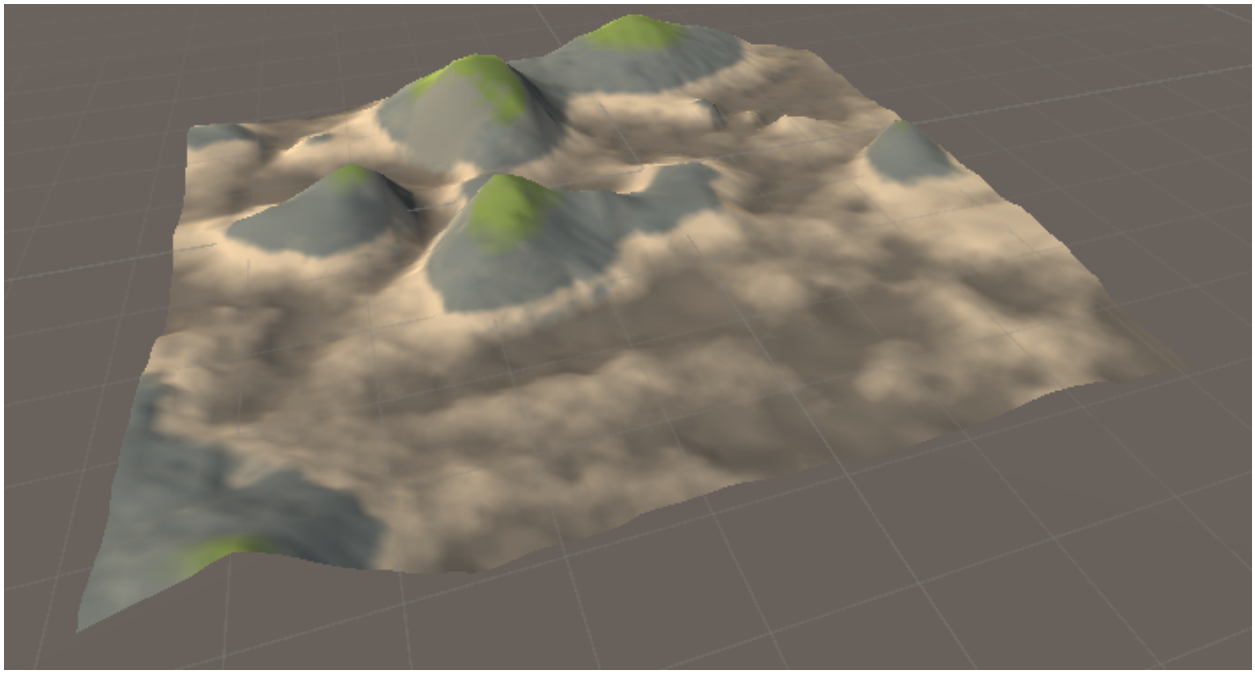


図4.33 シミュレーション環境の地形(激しい凹凸のある海岸)

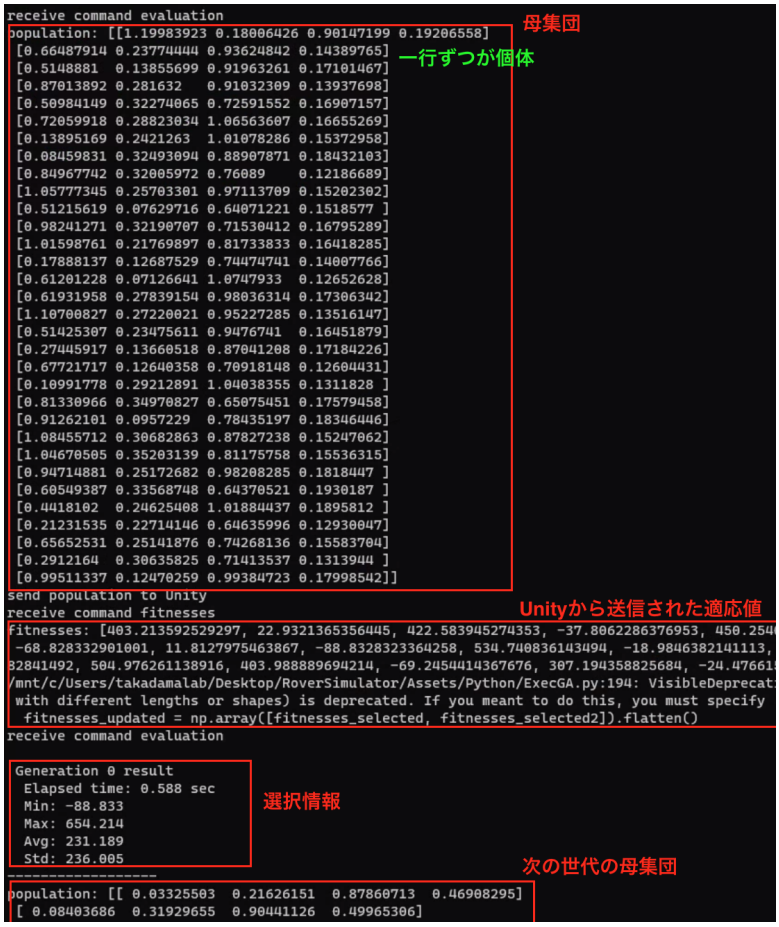

receive command evaluation
population: [[1.19983923 0.18006426 0.90147199 0.19206558]
[0.66487914 0.23774444 0.93624842 0.14389765]
[0.5148881 0.13855699 0.91963261 0.17101467]
[0.87013892 0.281632 0.91032309 0.13937698]
[0.50984149 0.32274065 0.72591552 0.16907157]
[0.72059918 0.28823034 1.06563607 0.16655269]
[0.13895169 0.2421263 1.01078286 0.15372958]
[0.08459831 0.32493094 0.88907871 0.18432103]
[0.84967742 0.32005972 0.76089 0.12186689]
[1.05777345 0.25703301 0.97113709 0.15202302]
[0.51215619 0.07629716 0.64071221 0.1518577]
[0.98241271 0.32190707 0.71530412 0.16795289]
[1.01598761 0.21769897 0.81733833 0.16418285]
[0.17888137 0.12687529 0.74474741 0.14007766]
[0.61201228 0.07126641 1.0747933 0.12652628]
[0.61931958 0.27839154 0.98036314 0.17306342]
[1.10700827 0.27220021 0.95227285 0.13516147]
[0.51425307 0.23475611 0.9476741 0.16451879]
[0.27445917 0.13660518 0.87041208 0.17184226]
[0.67721717 0.12640358 0.70918148 0.12604431]
[0.10991778 0.29212891 1.04038355 0.1311828]
[0.81330966 0.34970827 0.65075451 0.17579458]
[0.91262101 0.0957229 0.78435197 0.18346446]
[1.08455712 0.30682863 0.87827238 0.15247062]
[1.04670505 0.35203139 0.81175758 0.15536315]
[0.94714881 0.25172682 0.98208285 0.1818447]
[0.60549387 0.33568748 0.64370521 0.1930187]
[0.4418102 0.24625408 1.01884437 0.1895812]
[0.21231535 0.22714146 0.64635996 0.12930047]
[0.65652531 0.25141876 0.74268136 0.15583704]
[0.2912164 0.30635825 0.71413537 0.1313944]
[0.99511337 0.12470259 0.99384723 0.17998542]]
send population to Unity
receive command fitnesses
fitnesses: [403.213592529297, 22.9321365356445, 422.583945274353, -37.8062286376953, 450.254
-68.828332901001, 11.8127975463867, -88.8328323364258, 534.740836143494, -18.9846382141113,
32841492, 504.976261138916, 403.988889694214, -69.2454414367676, 307.194358825684, -24.47661
/mnt/c/Users/takadamalab/Desktop/RoverSimulator/Assets/Python/ExecGA.py:194: VisibleDeprecat
with different lengths or shapes) is deprecated. If you meant to do this, you must specify
fitnesses_updated = np.array([fitnesses_selected, fitnesses_selected2]).flatten()
receive command evaluation
Generation 0 result
Elapsed time: 0.588 sec
Min: -88.833
Max: 654.214
Avg: 231.189
Std: 236.005
population: [[0.03325503 0.21626151 0.87860713 0.46908295]
[0.08403686 0.31929655 0.90441126 0.49965306]
母集団
一行ずつが個体
Unityから送信された適応値
選択情報
次の世代の母集団

図4.34 GAの様子

第5章　　試験項目設定

番号	検証項目名	対応する自己審査項目の要求番号(複数可)	実施予定日	実施日	
V1	質量試験	R1	8/14	10/16	完了
V2	収納放出試験	R1, R10	8/14	10/16	完了
V3	長距離通信試験	R2	7/21	7/23	完了
V4	落下試験	R3	6/16	6/23	完了
V5	開傘衝撃試験	R3	8/9	10/15	完了
V6	準静荷重試験	R4	8/13	10/12	完了
V7	振動試験	R5	--	--	実施中止 ※代替シミュレーション実行
V8	分離衝撃試験	R6	--	--	実施中止 ※代替シミュレーション実行
V9	通信機電源ON/OFF試験	R7	8/9	8/9	完了
V10	通信周波数変更試験	R8	6/30	6/30	完了
V11	End-to-End試験	R9, R11	9/20~10/20	10/21	完了
V12	制御履歴レポート作成試験	R12	9/20~10/20	10/21	完了
V13	着地衝撃試験	M1	8/9		完了
V14	走行性能確認試験	M2	8/6	8/27	完了
V15	電力耐久試験	M3	8/5	8/13	完了

V16	故障検知試験	M4	8/13	8/20	完了
V17	最適化実行試験	M5	8/13	8/27	完了
V18	反転・横転復帰試験	M6	8/16	10/7	完了
V19	ナビゲーション走行試験	M7	8/12	8/28	完了
V20	ゴール検知試験	M8	8/19	8/27	完了

第6章　　実施試験内容

以下，
(1)審査基準における安全面/ミッション面を達成するための試験を(※安全面)/(ミッション面)と記述する.
(2)本章では，本ミッションで使用する新スタビライザを搭載したCanSatを使用した試験結果を記載する. なお，一部の試験項目では新スタビライザに加えて，旧スタビライザを搭載したCanSatによる試験結果を各々のサブセクションⅠ・Ⅱに分割したのち，併記している（新スタビライザ，旧スタビライザの詳細については4.1節を参照されたい）.

1. システム要求を満たすための試験内容

（V1） 質量試験 (※安全面)

- 実施日: 10/16
- 実施責任者 : 松田
- 充足要求項目 : R1
- 目的
  CanSatとパラシュートの合計質量がレギュレーションに記載されている質量(1050g)以下であることを確認する.
- 試験内容
  CanSatとパラシュートの合計質量を電子秤で計測し，レギュレーションに記載されている質量(1050g)以下であることを確認する.
- 結果
  CanSatとパラシュートの合計質量はレギュレーションに記載されている1050g以下を満たしていることを確認した.

表6.1　質量試験の結果（新スタビライザ）

CanSatの質量	934 g
パラシュートの質量	110 g
合計	1045 g†

†本試験で使用した電子秤は小数点以下が切り捨てられた質量を表示するため，CanSat，パラシュートそれぞれの質量実測値（図6.2, 6.3）の合計と，Cansatとパラシュートの合計質量実測値（図6.1）がわずかに異なることに留意されたい.

それぞれ図6.1に合計質量，図6.2にパラシュートの質量，図6.3にCanSatの質量を計測した時の様子を示す.

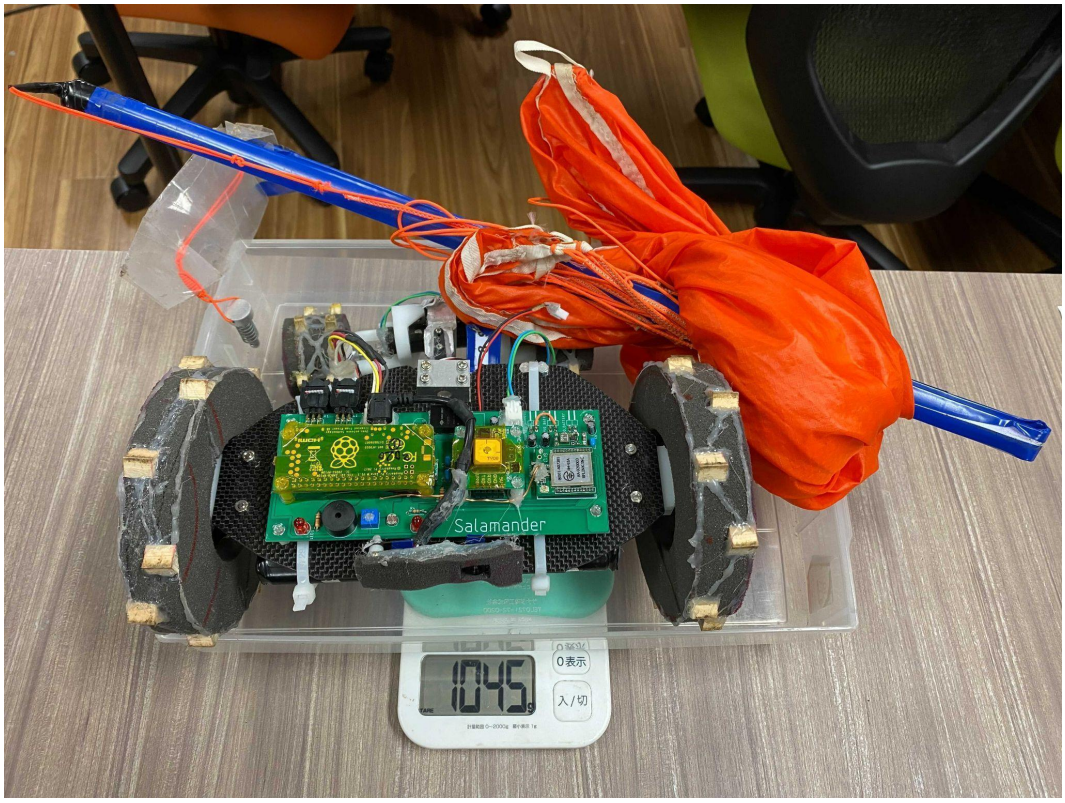


図6.1 CanSat(新スタビライザ)とパラシュートの合計質量

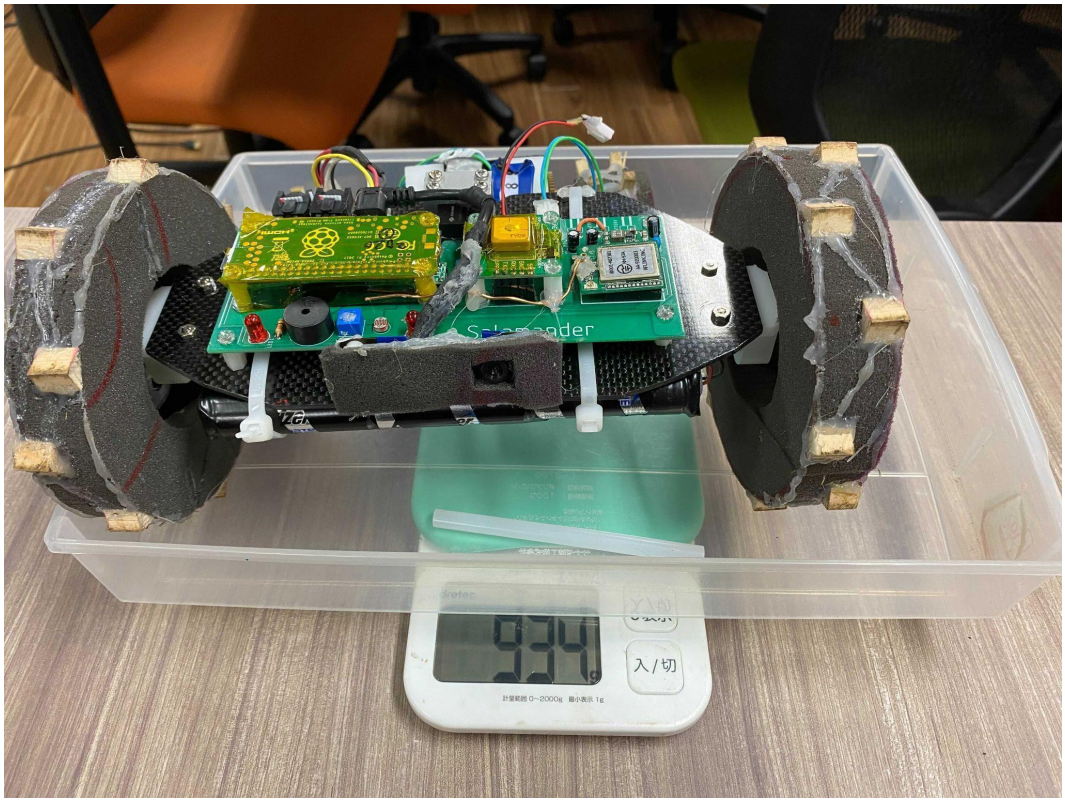


図6.2 CanSat(新スタビライザ)の質量

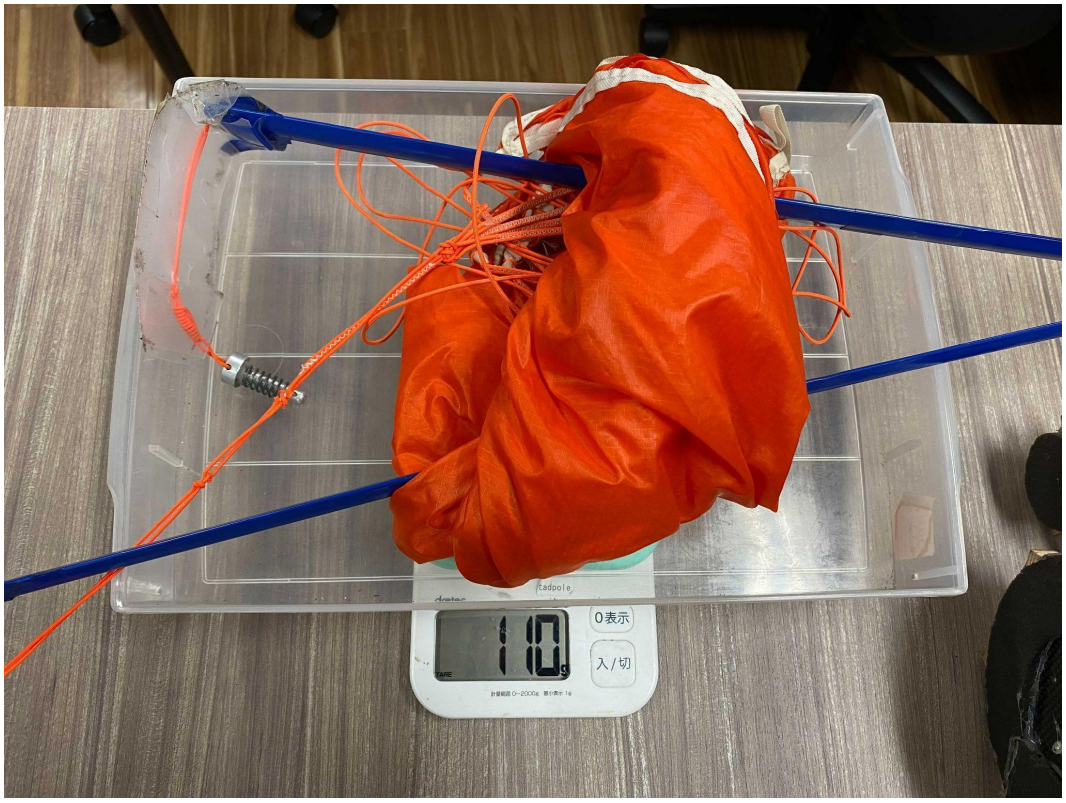


図6.3 パラシュートの質量

- 結論
  以上の結果より，CanSatはレギュレーションである1050gを満たしていることが確認された
  .

(V2) 収納放出試験 (※安全面)

- 実施日: 10/16
- 実施責任者 : 松田
- 充足要求項目 : R1, R10
- 目的

以下の3点を確認することを目的とする.

- CanSatのレギュレーション(直径146mm，高さ240mm)を満たすこと
- キャリアに収納可能であり，円滑に放出可能であること
- CanSatを収納するまでの工程が5分以内に収まっていること

- 試験内容

以下の4つの項目を実施する.

- CanSatを入れるキャリアの内径を計測する.
- CanSatの高さを計測する.
- CanSatがキャリアに収納・放出可能であることを確認する．これを5回繰り返し実施し，

問題ないことを確認する.

- CanSat収納の際に収納までの手順の時間を計測し，5分以内に収納可能であることを確認する.

○ 収納手順

1. CanSatに電源を入れる
2. パラシュートを畳む
3. パラシュートの紐を絡みにくいようにまとめる
4. CanSatの側面に畳んだパラシュートをあてがう
5. キャリアに収納する
6. CanSatとのPCとの通信を確認し，sshで接続する
7. CanSatのプログラムを起動する

○ 結果

Ⅰ. 旧スタビライザを搭載した機体改修前CanSat

図6.4はキャリアの内径を測定した様子を示す. 使用しているキャリアは大会規定のキャリアと同程度の大きさであり，開発したCanSatが問題なくキャリアに収納できていることを確認した.

図6.4 キャリアの内径（旧スタビライザ）

また，CanSatの高さを測定した様子を図6.5に示す．CanSatの高さは236mmであり，レギュレーションに記載された240mm以内の制約を満たしている．

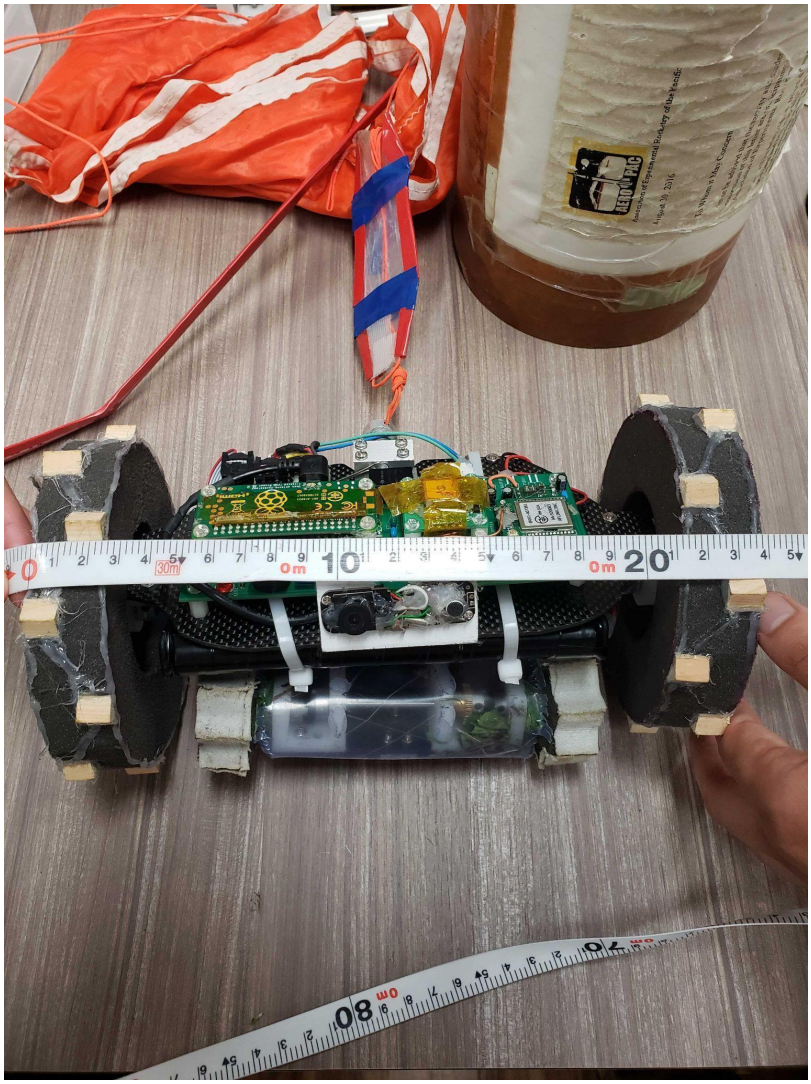


図6.5 CanSatの高さ（旧スタビライザ）

キャリアの収納から放出までの一連の様子を以下の表6.2にまとめる．キャリアからの放出についてはキャリアの自重で放出できるか(キャリアの筒先を下に向けるだけで，CanSatが放出できる)を評価している．

表6.2 収納放出試験の結果（旧スタビライザ）

試行回	収納時間	自重での放出	YouTubeリンク
1回目	1分12秒(成功)	成功	https://youtu.be/tKOtU7gf0_Q
2回目	1分34秒(成功)	成功	https://youtu.be/W1qJto9ws6o
3回目	1分14秒（成功）	成功	https://youtu.be/ROnVT

			AwDntQ
4回目	1分6秒（成功）	成功	https://youtu.be/5QeBz8GhRjU
5回目	1分7秒（成功）	成功	https://youtu.be/m-P9GngJyTw
成功率	100%(5/5)	100% (5/5)	

Ⅱ．新スタビライザを搭載した機体改修後CanSat

図6.6にCanSatの高さを測定した様子を示す．CanSatの高さは238mmであり，レギュレーションに記載された240mm以内の制約を満たしている

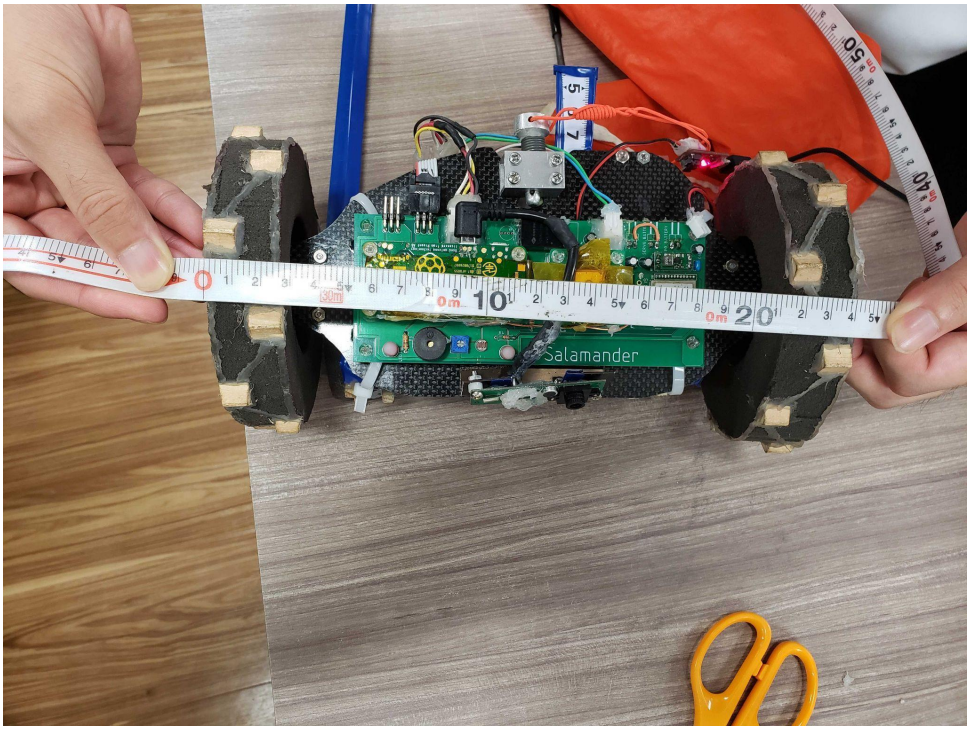


図6.6 CanSatの高さ（新スタビライザ）

新スタビライザを搭載したCanSatを用いて実施した収納放出試験の結果を表6.3に示す．

表6.3 収納放出試験の結果(新スタビライザ)

試行回	収納時間	自重での放出	YouTubeリンク
1回目	1分29秒(成功)	成功	https://www.youtube.com/watch?v=cexiYipoYWU

2回目	1分16秒(成功)	成功	https://www.youtube.com/watch?v=KvDANYLLP-w
3回目	1分10秒(成功)	成功	https://www.youtube.com/watch?v=bcwNcU5wk-s
4回目	58秒(成功)	成功	https://www.youtube.com/watch?v=4RJkrmHM4kU
5回目	1分3秒(成功)	成功	https://www.youtube.com/watch?v=o9UwkHpAa9g
成功率	100%(5/5)	100% (5/5)	

- 結論

以上の結果より，以下の3点が確認された.

- キャリアの内径と高さに関するレギュレーションを満たすこと
- 円滑なキャリア収納・放出が可能であること
- CanSatの収納までの工程が5分以内に収まっていること

(V3) 長距離通信試験 (※安全面)

- 実施日: 7/23
- 実施責任者 : 松田
- 充足要求項目 : R2
- 目的

CanSat投下時にて落下地点を見失ったなどの場合に，CanSatをロストすることを避けるため，CanSatのGPSデータを本営が直接取得できる距離を確認する.

- 試験内容

CanSatからLoRa通信モジュールを通してGPS座標を定期的に送信し続ける. その上でノートPCにLoRa通信モジュールを接続し，CanSatからGPS座標を受信できることを確認する. その後，CanSatをPCから徐々に離し，通信が切断された地点のGPS座標を記録する. そして，この2点の座標から最大通信可能距離を算出する. (なお，本実験は本チームのCanSatを使用し高玉研究室 Carryberチームと合同で実験する.)

- 結果

CanSatとPC(地上局)の通信が切断された時のそれぞれのGPS座標を以下の表6.4に示す. また，図6.6はその時の座標を地図上に示したものである.

表6.4 長距離通信試験の結果

PC(地上局)の座標	緯度: 35.642378 , 経度: 139.523425

CanSatの座標	緯度: 35.6375033，経度: 139.5392733
最大通信可能距離	1.53 km

図6.6 長距離通信試験時の地上局とCanSatの位置

- 結論
  結果より地上局とCanSatの通信可能距離は1.53kmであり，会場の大きさを踏まえるとCanSatの投下後，地上局を介しCanSatのGPS座標を取得することは十分可能であるといえる．したがって，ロスト対策に十分な通信可能距離であるということが確認された．

(V4) 落下試験 (※安全面)

- 実施日: 6/23
- 実施責任者 ： 松田
- 充足要求項目 ： R3
- 目的
  キャリアから放出されたときに減速機構であるパラシュートが問題なく展開されて減速すること，また，落下速度のレギュレーション(5.0m/s)を満たすことを確認する．

- 試験内容
  電気通信大学西6号館7階（地面からの高さ約27.3m地点）から，あらかじめキャリアに収納済みの「疑似CanSat」を放出する．この「疑似CanSat」は，実際に開発中のCanSatと同等の重量，形状，高さ及び重心である．「疑似CanSat」の重心位置をそれぞれ図6.7と図6.8に示す．

本試験はCanSat自体の回路やソフトウェアには関与しない，あくまでパラシュートの性能およびCanSatのハードウェア性能に対する試験であるため，本試験はこの「疑似CanSat」を用いて行っても実際のCanSatを用いた場合と同等の結果が得られるといえる．

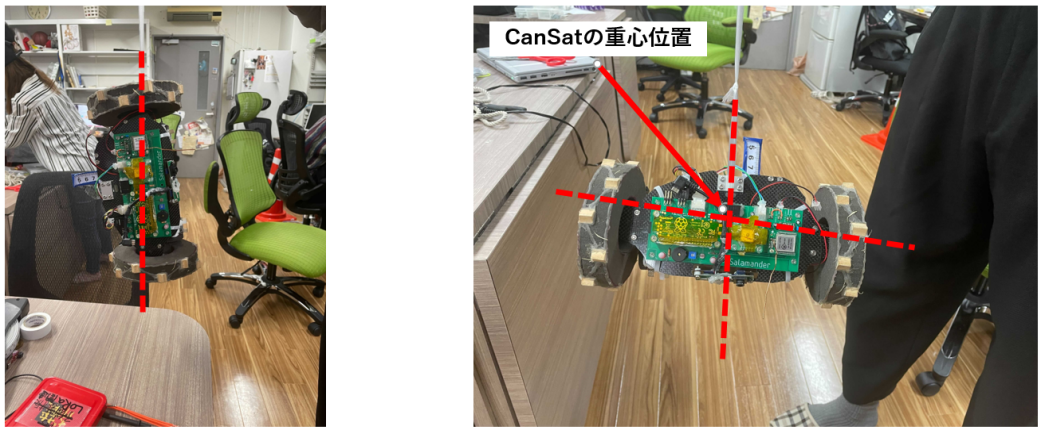


図6.7 CanSatの重心位置

図6.8 疑似CanSatの重心位置

本試験で確認することは以下の通りである．

- パラシュートの正常な開傘
- パラシュートによる減速

表6.5 各階間の通過時間

階	通過時間
7 → 6	1秒0フレーム

6 → 5	0秒9フレーム
5 → 4	0秒17フレーム
4 → 3	0秒18フレーム
3 → 2	0秒18フレーム
2 → 1	0秒18フレーム

これら2項目はビデオカメラで撮影した動画をもとに確認を取る．特に落下時の速度に関しては，図 6.7に示すように，パラシュートをつけた状態でキャリアから放出されて落下し，終端速度に達している高さからの通過時間を計測する． 実験動画から，各階間を通過する時間を動画から算出した．(表6.5) これより，4階以下では，通過時間が同じとなっていることから，終端速度であるといえる，建物3階相当(地上から8.4m)をCanSatが通過してから，着地までにかかった時間を動画から計測し，以下の式を用いて計算する．

$$h = 8.4 \ \ [\mathrm{m}]$$

$$v = \frac{h}{t} \ \ [\mathrm{m/s}]$$

また，自由落下時の終端速度は次の式より，23.1m/sと求められる．

$$v = \sqrt{2gh}$$

$$h = 27.3\ [\mathrm{m}], \qquad g = 9.81\ [\mathrm{m/s^2}]$$

$$\therefore\ v = 23.1\ [\mathrm{m/s}]$$

図6.9 地面から3階までの高さ

○ 結果

試験の結果を表6.6にまとめる．5回全てでパラシュートの開傘が確認された．

表6.6 開傘衝撃試験の結果

試行回	パラシュートの開傘	YouTubeリンク
1回目	成功	https://youtu.be/2KJ_XuNXZ6Q
2回目	成功	https://youtu.be/O6vjGk377tc
3回目	成功	https://youtu.be/-0KGvgfsceg
4回目	成功	https://youtu.be/bE02tp9ydio
5回目	成功	https://youtu.be/HmMTe7tzLOQ

成功率	100% (5/5)	-

また，落下速度の計算結果を表6.7に示す．各落下において動画から落下までにかかる時間を見積もり，この時間から速度を算出した．

表6.7 パラシュートの落下速度

試験回数	算出速度	動画該当箇所
1回目	8.4 m / 1.20 s ≒ 7.0 m/s	00:21:10~00:22:30
2回目	8.4 m / 1.27 s ≒ 6.6 m/s	00:21:46~00:22:73
3回目	8.4 m / 1.40 s ≒ 6.0 m/s	00:16:06~00:17:46
4回目	8.4 m / 1.30 s ≒ 6.5 m/s	00:10:70~00:12:00
5回目	8.4 m / 1.40 s ≒ 6.0 m/s	00:15:20~00:16:60

落下速度の最大値は 7.0 m/s であり，これは自由落下時の速度である 23.1 m/s に比べて小さく，パラシュートによる減速が確認された．

○ 結論

キャリアから放出されたときに減速機構であるパラシュートが問題なく展開し，減速することが確認された．

（V5） 開傘衝撃試験 (※安全面)

○ 実施日: 10/15

○ 実施責任者 : 松田

○ 充足要求項目 : R3

○ 目的

パラシュート開傘時の衝撃にCanSat本体とパラシュートとの結合部分が耐えられることを確認する．

○ 試験内容

CanSatを固定し，パラシュートの紐を付けて自由落下させる．この時加速度センサのログを確認しパラシュート開傘時の衝撃(10 [G])に耐えられたかを確認する．衝撃を加えた後，パラシュート切り離しから走行までのシーケンスを実施し，全てのセンサ・動力系の動作確認，CanSatに破損がないかを確認する．なお，10 [G]という値は過去のACTSでの開傘時の加速度ログをもとに独自に設定したものである．また，本試験におけるセンサ，モータの出力は以下の基準を満たしていれば正常と扱う．

＜センサ類＞
・気圧センサ：実験開始前に計測した気圧±1[hPa]
・9軸センサ：機体の姿勢が静安時の加速度のノルムが0.95[G]~1.05[G](重力加速度1[G]±許容測定誤差5%，および，機体の姿勢によってxyz軸の値が変化している
・光センサ：通常時に光があたっていると判定(センサ値がHIGHを示す)し，手で光センサを覆ったときに光があたっていないと判定(センサ値がLOWを示す)する
・ブザー：音のON/OFFができる

＜モータ類＞
・サーボモータ：上下方向（または左右方向）にスタビライザを制御できる
・モータ：機体が走行可能な程度に回転する

(参考)開傘衝撃試験で与える加速度(10[G])の根拠

開傘衝撃のを決定するにあたって，ACTS2020のSolamilチームのフライトのログ(図6.10および図6.11)を参考にした．この時，加速度が特に大きくなっている箇所(ピーク)が2箇所存在することが確認できる．最初のピークは開傘による衝撃を指したものであり，2度目のピークは着地時の衝撃を指したものである．よって，最初のピークの値を使用する．その値は，それぞれおよそ6 [G]，10 [G]であり，これをもとに10 [G]と設定した．なお，Solamilチームが使用したパラシュートは本ミッションで使用しているパラシュートと同一であり，同程度の開傘衝撃が加わることが想定される．

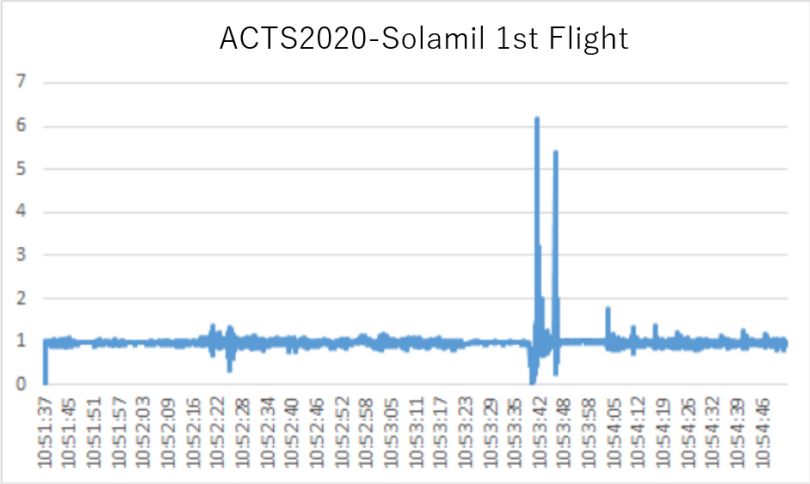


図6.10 開傘衝撃のログ ACTS2020 Solamil 1回目のフライトより

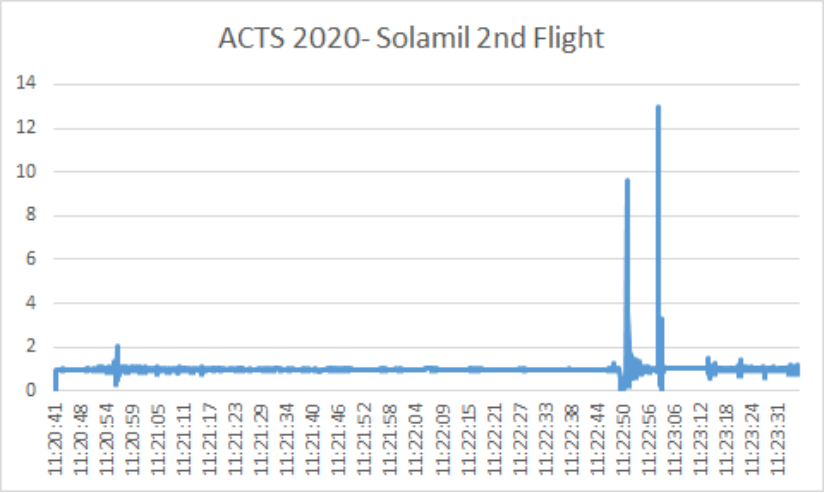


図6.11 開傘衝撃のログ ACTS2020 Solamil 2回目のフライトより

○ 結果

Ⅰ. 旧スタビライザを搭載した機体改修前CanSat

各試験におけるグラフを図6.12~6.16に示す. 横軸は経過時間[s], 縦軸は加速度の大きさ(ノルム)が重力加速度の何倍であるかを示す.

1回目

図6.12 開傘衝撃試験1回目

2回目

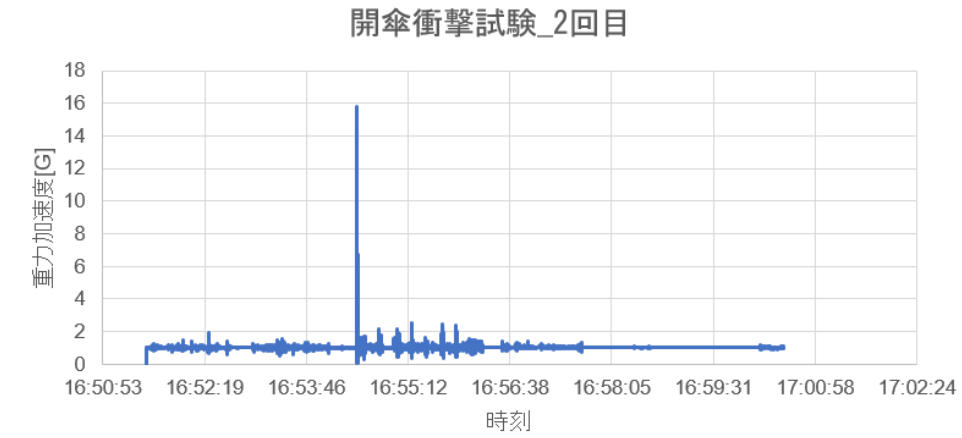


図6.13 開傘衝撃試験2回目

3回目

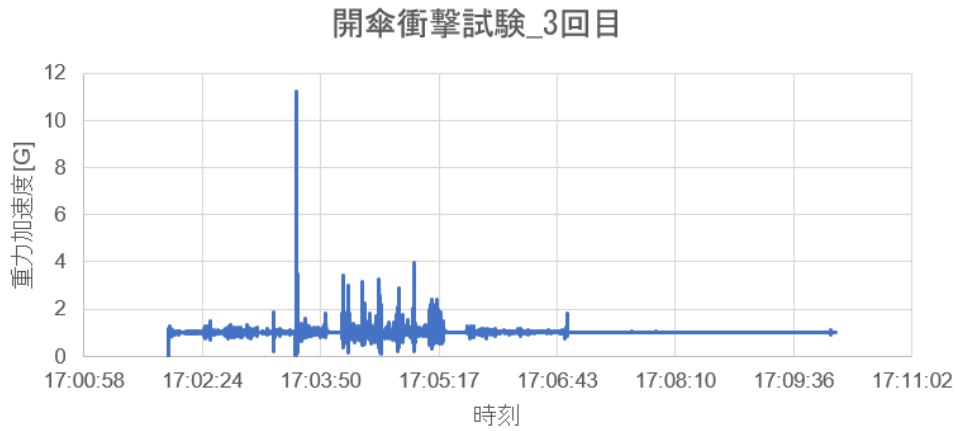


図6.14 開傘衝撃試験3回目

4回目

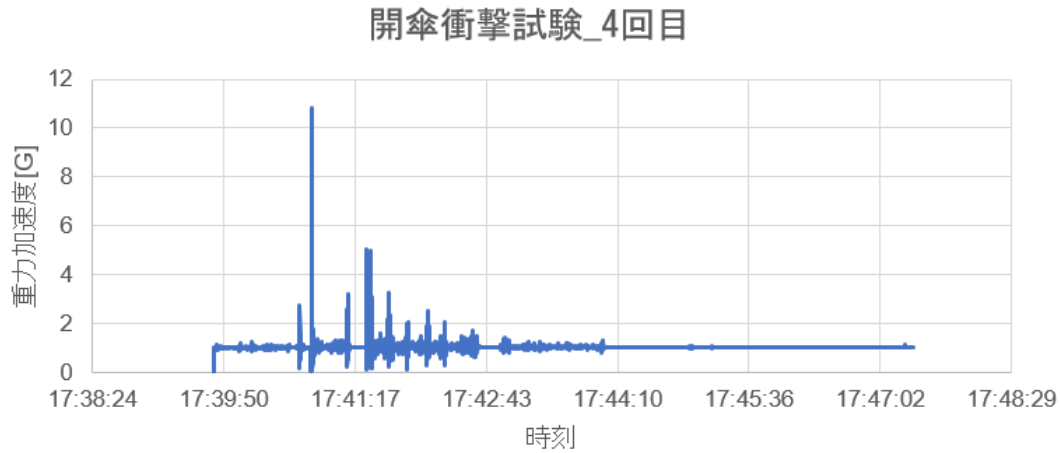


図6.15 開傘衝撃試験4回目

5回目

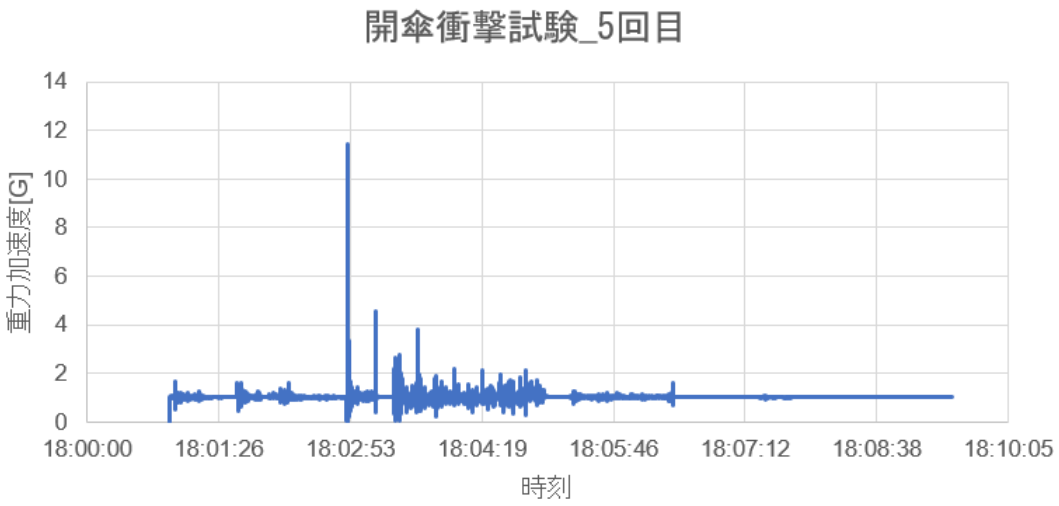


図6.16 開傘衝撃試験5回目

以下の表6.8に試行ごとの結果を示す．全5回の試行において，開傘衝撃CanSat 本体とパラシュートとの結合部分に異常がないことが確認された．

表6.8 開傘衝撃試験の結果

試行回	CanSatの外的損傷	センサ類	モータ類	最大加速度 [G]	YouTubeリンク
1回目	なし	正常	正常	17.9678	https://youtu.be/0BGqvq2Q9TQ
2回目	なし	正常	正常	15.8195	https://youtu.be/AySZmeSZwrw
3回目	なし	正常	正常	11.2325	https://youtu.be/D-JWooFXbqg
4回目	なし	正常	正常	10.8357	https://youtu.be/QaO_i3xQ8oo
5回目	なし	正常	正常	11.4382	https://youtu.be/OpfclTffx6w
成功率	100% (5/5)	100% (5/5)	100% (5/5)	―	

Ⅱ．新スタビライザを搭載した機体改修後CanSat

実験時の加速度グラフを図6.17, 6.18，静荷重を加えた後のCanSatの損傷および各部品の故障の有無を表6.9に示す．

機体改修後1回目

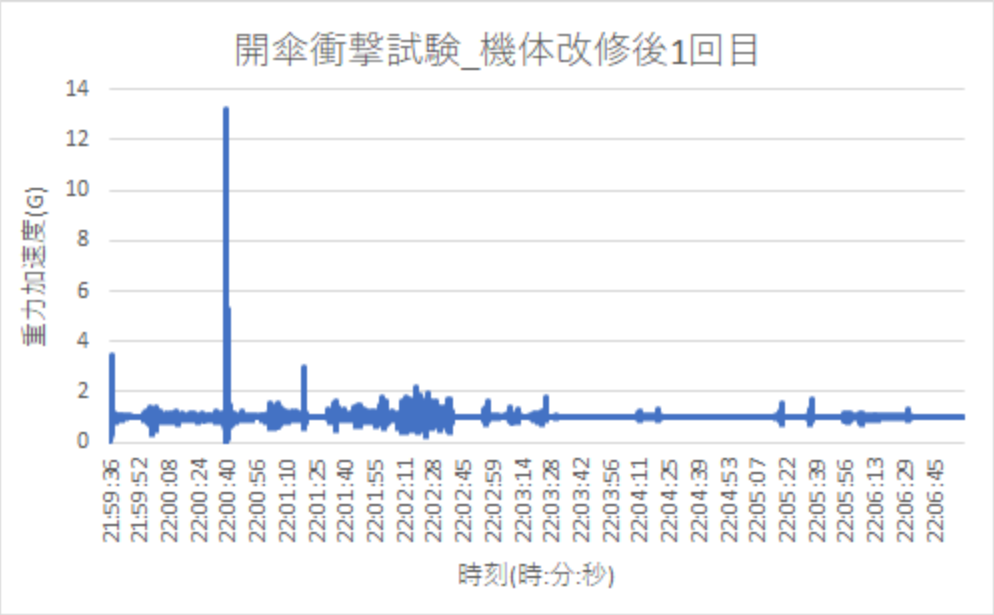


図6.17 開傘衝撃試験機体改修後1回目

機体改修後2回目

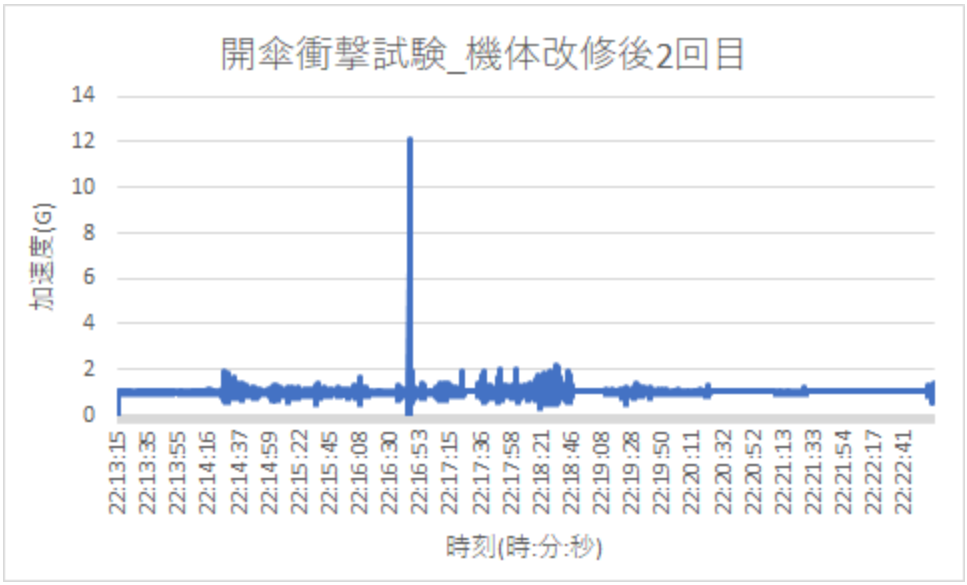


図6.18 開傘衝撃試験機体改修後2回目

表6.9 静荷重試験の結果（新スタビライザ）

試行回	CanSatの外的損傷	センサ類	モータ類	最大加速度[G]	YouTubeリンク

1回目	なし	正常	正常	13.2198	https://youtu.be/gqy9vCMDpEA
2回目	なし	正常	正常	12.1051	https://youtu.be/95GU7qh4Wxc
成功率	100%(2/2)	100%(2/2)	100%(2/2)	-	

- 結論
  以上より, CanSatはパラシュート開傘時にかかる最大予想衝撃である10 [G]の衝撃に耐久可能であることが確認された.

(V6) 準静荷重試験 (※安全面)

- 実施日: 10/12
- 実施責任者 : 松田
- 充足要求項目 : R4
- 目的
  打ち上げ時の準静的荷重(以下, 静荷重と呼ぶ)によってCanSatのハードウェア・ソフトウェア共に問題が発生せず, 正常に動作することを確認する.

- 試験/解析内容
  ロケット搭載状態を想定したCanSatを紐を繋いだ袋に入れハンマー投げの要領で回すことでロケットによる静荷重を再現する. 静荷重はレギュレーションの5.2項に記載されている通り, CanSatの高さ方向に10 [G]を10秒間CanSatに与える. その後, ハードウェアが破損していないことを確認する. また, 放出判定からパラシュート切り離しまでの各シーケンス動作が正常に動作するかを確認することで, モータやサーボモータに損傷がないことを確認する. また, 本試験におけるセンサ, モータの出力は以下の基準を満たしていれば正常と扱う.

  ＜センサ類＞
  ・気圧センサ:実験開始前に計測した気圧±1[hPa]
  ・9軸センサ:機体の姿勢が静安時の加速度のノルムが0.95[G]~1.05[G](重力加速度1[G]±許容測定誤差5%, および, 機体の姿勢によってxyz軸の値が変化している
  ・光センサ:光が照射されている時に光があたっていると判定(センサ値がHIGHを示す)し, 手で光センサを覆ったときに光があたっていないと判定(センサ値がLOWを示す)する
  ・ブザー:音のON/OFFができる
  ・GPS:1つ以上の衛星からGPS情報を受信し, その座標が試験の実施地点を指している.

  ＜モータ類＞
  ・サーボモータ:上下方向(または左右方向)にスタビライザを制御できる

・モータ：機体が走行可能な程度に回転する

○

本試験は全3回の試行を実施し，それぞれについてCanSatにかかっている加速度の大きさを時系列データとして取得することで，10 [G]の静荷重が与えられていることを確認する．また，静荷重の負荷開始時点からシーケンス動作の確認までの一連の動作を映像で記録する．

○ 結果

Ⅰ．旧スタビライザを搭載した機体改修前CanSat

9軸センサから得られた加速度の推移を図6.19～図6.23に示す．横軸は時間[s]，縦軸は加速度の大きさ(ノルム)が重力加速度9.8m/s^{2}の何倍であるか[G]を示す．これらのグラフから，CanSatを振り回すことでおよそ10Gの静荷重が10秒以上与えられていることが確認できる．

1回目

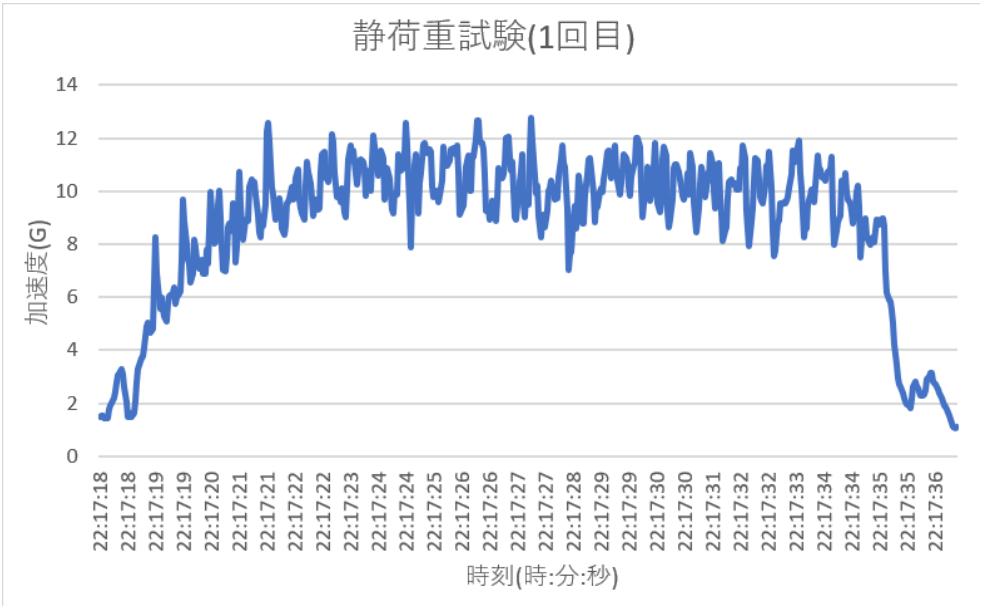


図6.19 準静荷重試験1回目の加速度グラフ

2回目

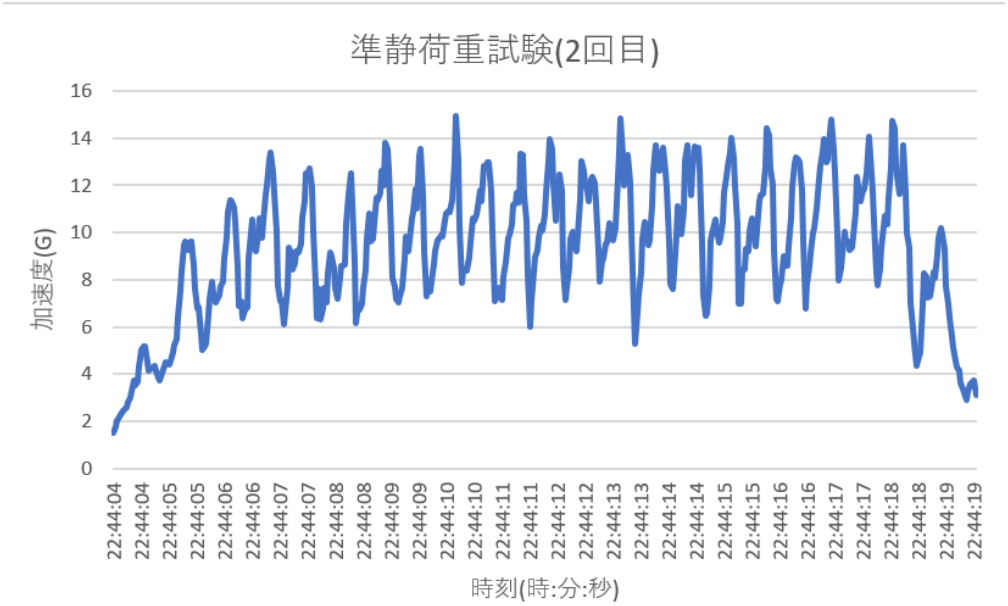


図6.20 準静荷重試験2回目の加速度グラフ

3回目

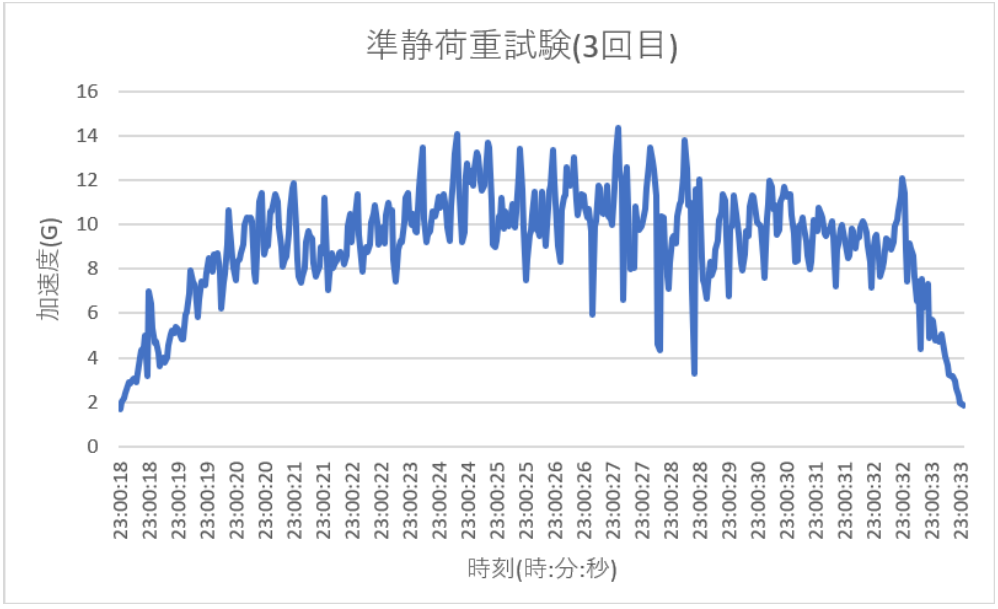


図6.21 準静荷重試験3回目の加速度グラフ

4回目

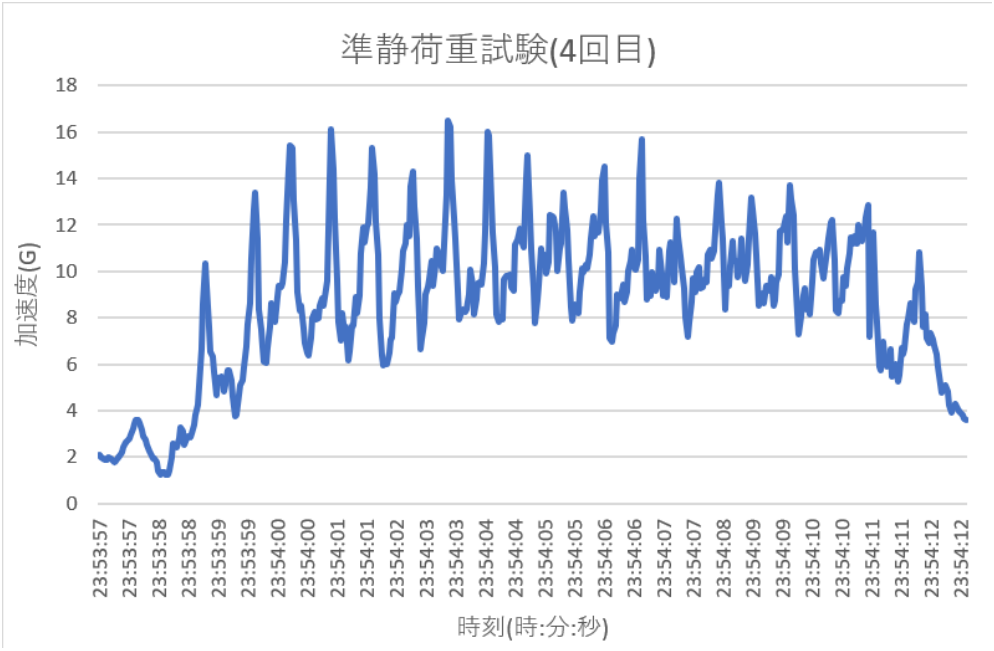


図6.22 準静荷重試験4回目の加速度グラフ

5回目

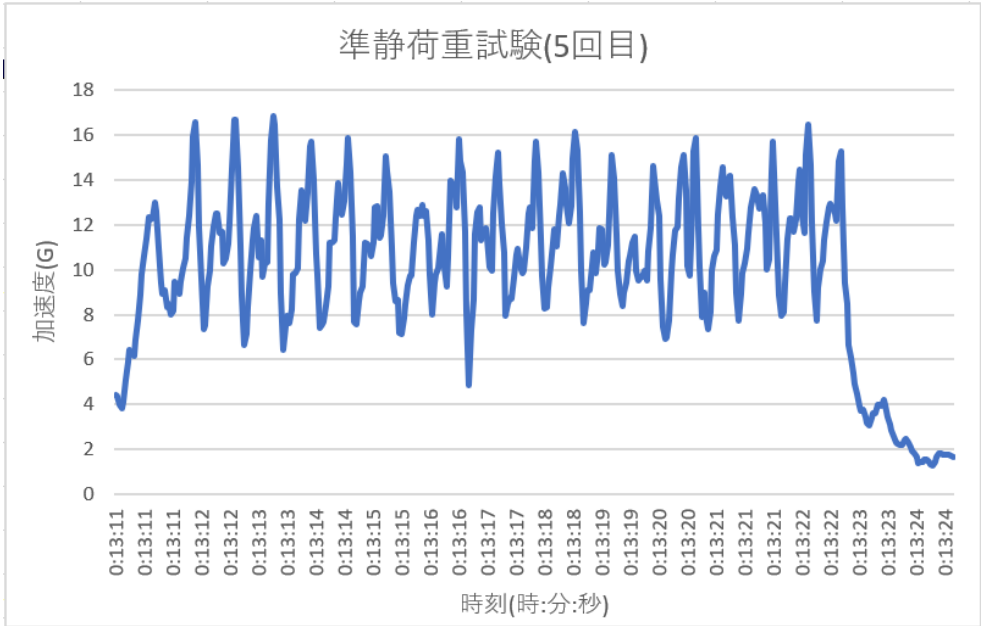


図6.23 準静荷重試験5回目の加速度グラフ

静荷重を加えた後のCanSatの損傷および各部品の故障の有無を表6.10に示す.

表6.10 静荷重試験の結果

試行回	CanSatの外的損傷	センサ類	モータ類	シーケンス動作	YouTubeリンク

1回目	問題なし	正常	正常	正常に遷移	https://youtu.be/8ujiCA92LOY
2回目	問題なし	正常	正常	正常に遷移	https://youtu.be/2jOecqu-GVU
3回目	問題なし	正常	正常	正常に遷移	https://youtu.be/y3gA4b3sSJ8
4回目	問題なし	正常	正常	正常に遷移	https://youtu.be/wi35R_mKLTo
5回目	問題なし	正常	正常	正常に遷移	https://youtu.be/_JXlNbpj_ho
成功率	100%(5/5)	100%(5/5)	100%(5/5)	100%(5/5)	

Ⅱ．新スタビライザを搭載した機体改修後CanSat

実験時の加速度グラフを図6.24, 6.25，静荷重を加えた後のCanSatの損傷および各部品の故障の有無を表6.11に示す．

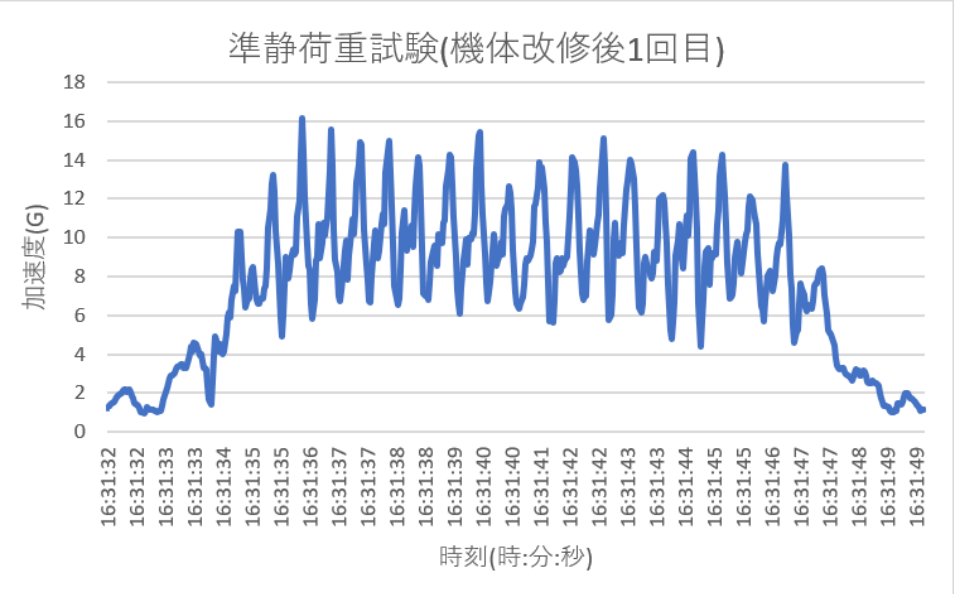


図6.24 準静荷重試験 機体改修後1回目の加速度グラフ

図6.25 準静荷重試験 機体改修後2回目の加速度グラフ

表6.11 静荷重試験の結果（新スタビライザ）

試行回	CanSatの外的損傷	センサ類	モータ類	シーケンス動作	YouTubeリンク
1回目	問題なし	正常	正常	正常に遷移	https://youtu.be/Gom5kBz_8e8
2回目	問題なし	正常	正常	正常に遷移	https://youtu.be/3OdwYjARWyM
成功率	100%(2/2)	100%(2/2)	100%(2/2)	100%(2/2)	

- 結論
  以上の結果より，CanSatは10Gの静荷重に耐久可能であり，打ち上げ時の衝撃によってハードウェア・ソフトウェア共に問題が発生せず，正常に動作することが確認された．

（V7） 振動試験（※実施中止）

- 実施日: 中止
- 実施責任者 ： 加藤
- 充足要求項目 ： R5

- 目的

ロケットによる打ち上げを想定した際，打ち出し時に生じる振動によってCanSatに異常が生じないことを確認する．

○ 試験内容

CanSatをキャリアに収納し，振動試験装置を用いてロケット上昇時に生じるランダムな振動を与える．試験後，パラシュート切り離しから走行までのシーケンスを実施し，全てのセンサ・動力系の動作確認，CanSatに破損がないかを確認する．

○ 結果

東京大学の施設を利用して実施する予定であったが，COVID-19の影響で施設利用の許可が下りなかった．本試験を実施可能な企業等の検討も試みたが，緊急事態宣言下における長距離の移動が必要であったため，感染リスクを考慮し，本試験の実施は見送ることとした．

○ 代替シミュレーション

振動試験装置を用いた振動試験の代替として，CAD上で組み立てた機体にレギュレーションに基づき30～2000[Hz]で15[G]の振動を加えたシミュレーションを行ったシミュレーションに使用したソフトウェアはAutodesk Fusion360である．

図6.26および図6.27にシミュレーション時に与えた荷重を示す．また，与えた荷重の詳細を表6.13および表6.14に示す．また，本CanSatに使用している部材の降伏強度を表6.15に示す．

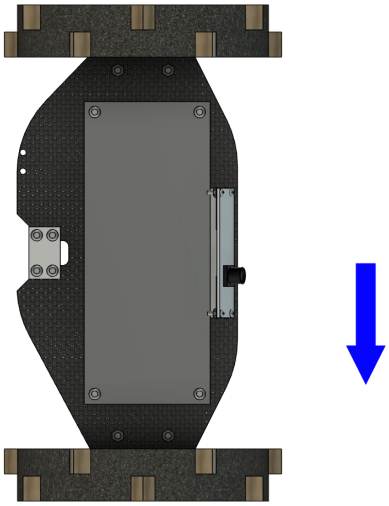


図6.26 重力(1[G])

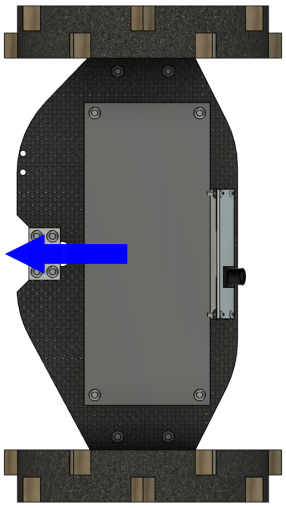


図6.27 グローバルな線形荷重(15[G])

表6.13 荷重(重力)

タイプ	重力
大きさ	9.807 [m / s^2]
X 値	9.807 [m / s^2]
Y 値	0 [m / s^2]
Z 値	2.178E-15 [m / s^2]
X 角度	-90 [deg]
Y 角度	0 [deg]
Z 角度	0 [deg]

表6.14 荷重(グローバルな線形荷重)

タイプ	グローバルな線形荷重
加速度	
大きさ	147.1 [m / s^2]
X 角度	0 [deg]
Y 角度	0 [deg]

Z 角度	0 [deg]

表6.15 部材の降伏強度

名前	降伏強度 [MPa]
CFRP	300
アルミニウム	275
ステンレス鋼	250
鋼	207
ABSプラスチック	20
アセタール樹脂	68.21
ゴム，シリコーン	10.34
プラスチック，不透明(白，黒)	20.67

○ 結果

図6.28および図6.29は解析結果の合計モード変位を示しており，それぞれ最小変位0から最大変位1まで着色されている．また，シミュレーション結果のアニメーションをYouTube リンクとして記載する．204.8 Hzおよび245 Hzの振動に対してCFRPを加工したシャーシに疲労が大きくなっている．しかしながら部品の変形は柔軟性のあるタイヤ部分のみに留まっている．

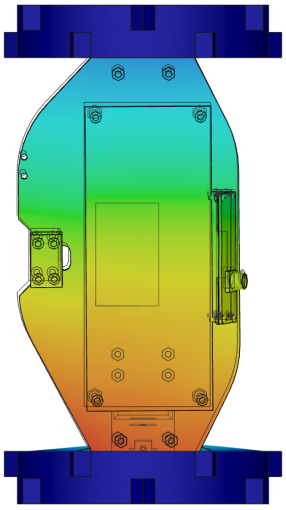


図6.28 合計モード変位(204.8[Hz])

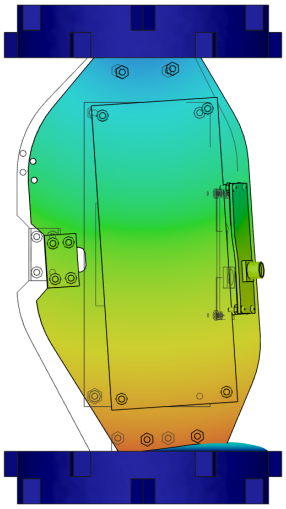


図6.29 合計モード変位(245[Hz])

- 解析結果アニメーション: https://youtu.be/bq8f0xpTpjI

○ 結論

以上の結果より，振動による変形はタイヤ部分に留まっておりロケット打ち出し時に生じる振動によってCanSatに異常が生じないことが確認された.

（V8） 分離衝撃試験（※実施中止）

- 実施日: 中止
- 実施責任者 : 加藤
- 充足要求項目 : R6
- ※他大の施設を借りて実験予定であったが，COVID-19の影響により実施ができなくなったため，別の施設(試験施設のある企業)を検討中である.
- 目的

  ロケットによる打ち上げを想定した際，ロケット分離時の衝撃によってCanSatに異常が生じないことを確認する.

- 試験内容

  振動試験と同様の振動試験装置を用いて40Gのショック振動を与える.試験後，パラシュート切り離しから走行までのシーケンスを実施し，全てのセンサ・動力系の動作確認，CanSatに破損がないかを確認する.

- 結果

  東京大学の施設を利用して実施する予定であったが，COVID-19の影響で施設利用の許可が下りなかった. 本試験を実施可能な企業等の検討も試みたが，緊急事態宣言下における長距離の移動が必要であったため，感染リスクを考慮し，本試験の実施は見送ることとした.

- 代替シミュレーション

  振動試験装置を用いた分離衝撃試験の代替として，CAD上で組み立てた機体にレギュレーションに基づき40[G]のショック振動を加えたシミュレーションを行った. シミュレーションに使用したソフトウェアはAutodesk Fusion360である. 図6.30および図6.31にシミュレーション時に与えた荷重を示す. また，与えた荷重の詳細を表6.16および表6.17示す.

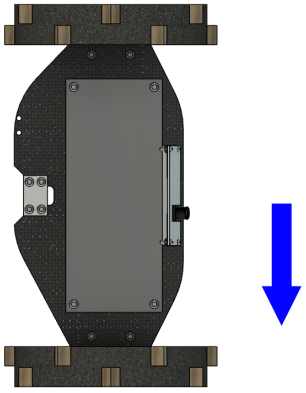


図6.30 重力(1[G])

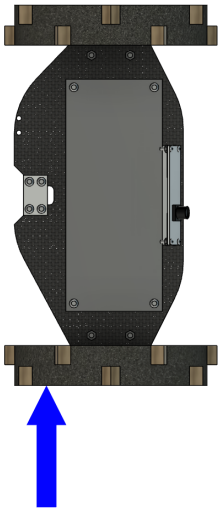


図6.31 グローバルな線形荷重(40[G])

表6.16 重力

タイプ	重力
大きさ	9.807 [m / s^2]
X 値	9.807 [m / s^2]
Y 値	0 [m / s^2]

Z 値	2.178E-15 [m / s^2]
X 角度	-90 [deg]
Y 角度	0 [deg]
Z 角度	0 [deg]

表6.17 グローバルな線形荷重

タイプ	グローバルな線形荷重
加速度	
大きさ	392.3 [m / s^2]
X 角度	0 [deg]
Y 角度	90 [deg]
Z 角度	0 [deg]

また，本CanSatに使用している部材の降伏強度を表6.18に示す．

表6.18 部材の降伏強度

名前	降伏強度 [MPa]
CFRP	300
アルミニウム	275
ステンレス鋼	250
鋼	207
ABSプラスチック	20
アセタール樹脂	68.21
ゴム，シリコーン	10.34
プラスチック，不透明(白，黒)	20.67

○ 結果

表6.19にシミュレーションによって得られた応力の解析結果を示す．

表6.19 解析結果概要

名前	最小値	最大値

応力		
Von Mises	2.345E-04 MPa	120.2 MPa
最大主応力	-23.28 MPa	91.1 MPa
最小主応力	-104 MPa	20.09 MPa

図6.32, 図6.33および図6.34にシミュレーションによって求められた応力分布を示す.

- Von-Mises
  [MPa] 0  120.2

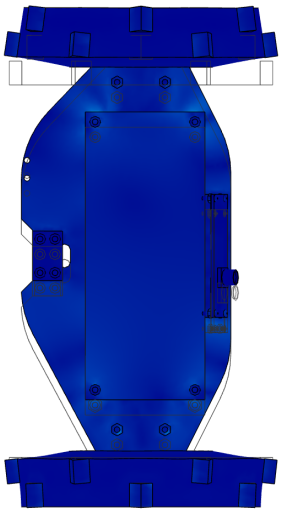


図6.32 Von-Mises応力

- 最大主応力
  [MPa] -23.28  91.1

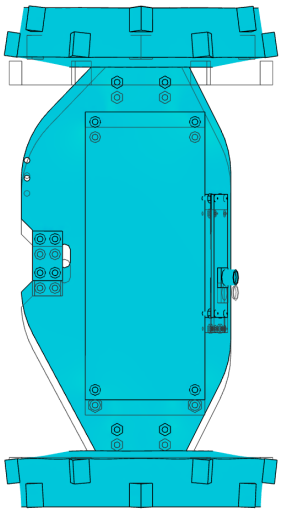


図6.33 最大主応力

■ 最小大主応力

[MPa] −104 20.1

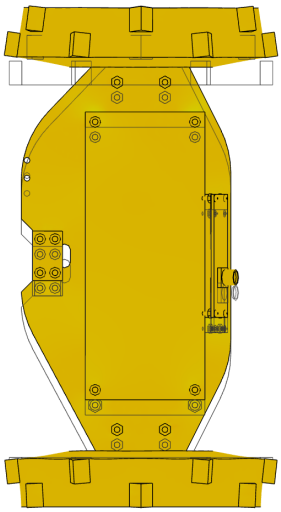


図6.34 最小主応力

シミュレーションにおける解析結果のVon-Mises応力分布および主応力分布が，使用している部材において最も降伏強度の低いゴム,シリコーンよりも大きくなる部分を図6.35に示す．Von-Mises応力分布が10[MPa]以上†として着色されている部分はCFRPおよびステンレス鋼であるため降伏強度は250[MPa]以上である．また，主応力分布が10.34[MPa]†以上として着色されている部分はCFRPおよびステンレス鋼であるためめ降伏強度は250[MPa]以上である．

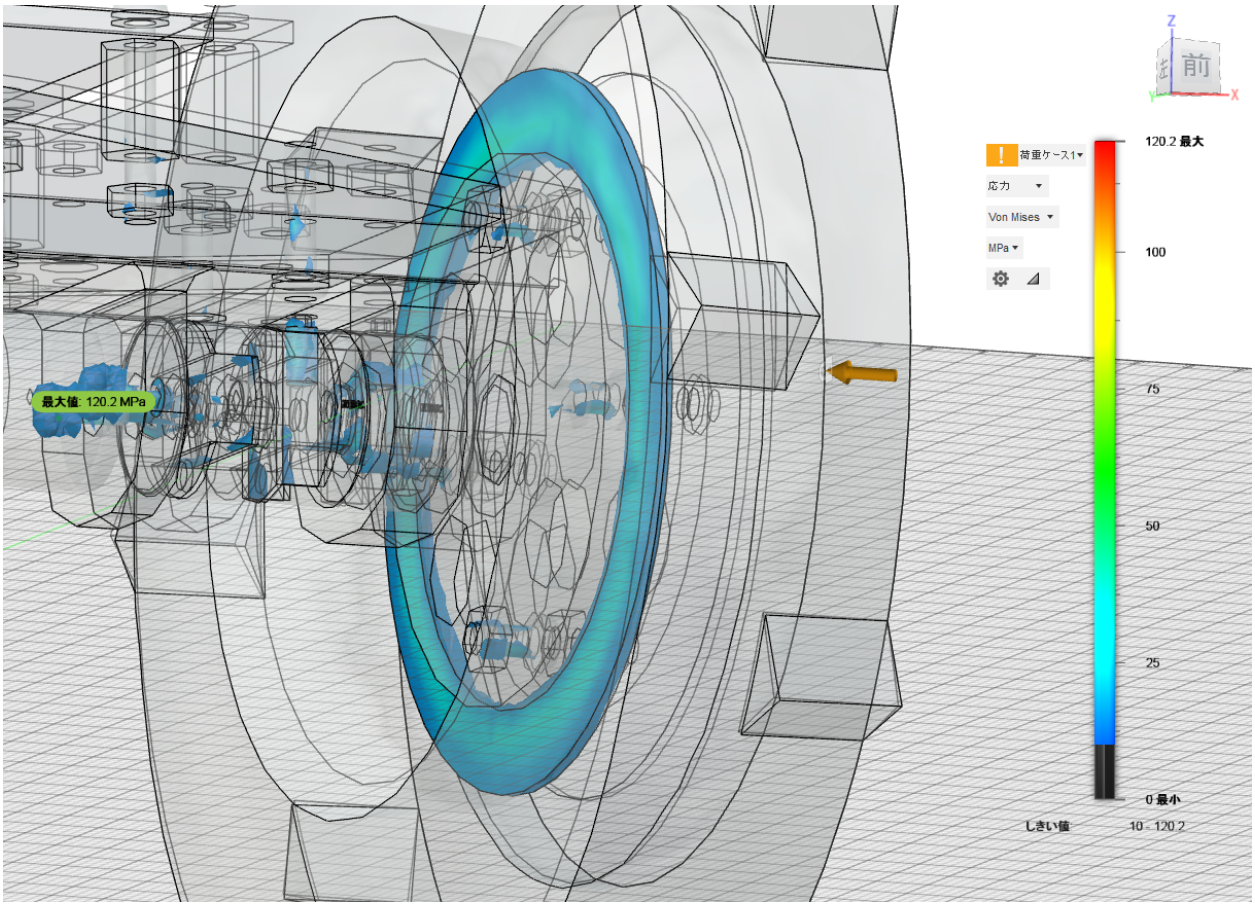


図6.35 Von-Mises応力分布(しきい値: 10[Mpa])

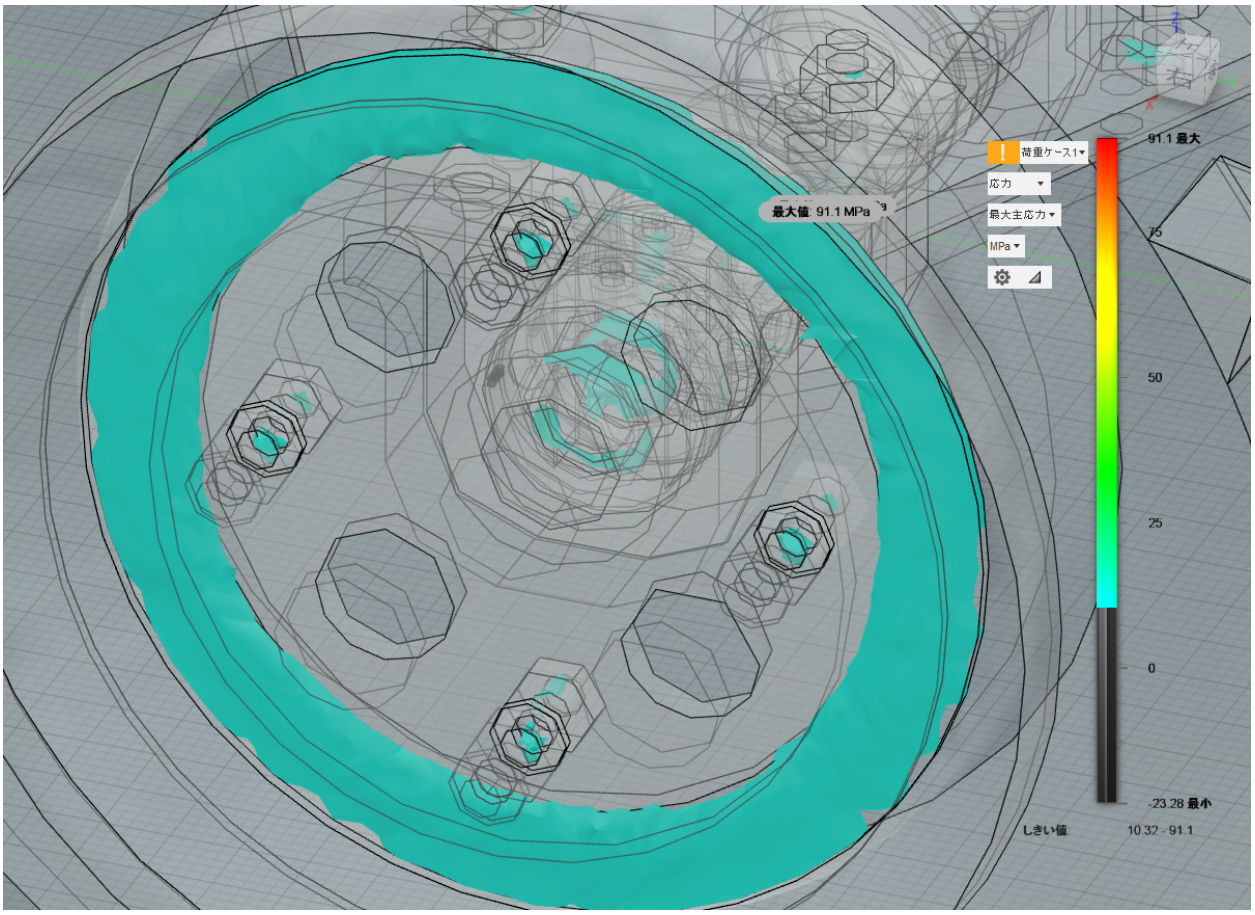


図6.36 主応力分布(しきい値: 10.32[Mpa])

† 構成部材のうち降伏強度が最小であるゴム，シリコーンは降伏強度が10.34[MPa]であるが，シミュレーションに使用したソフトウェアの仕様上しきい値の設定がGUI(グラフィカルユーザインタフェース)のみであり正確に10.34[MPa]にしきい値を設定することが困難であった．そのため構成部材のうち降伏強度が最小である10.34[MPa]よりも小さな値としてしきい値を設定した．最小の降伏強度である値よりも小さな値をしきい値として指定することで最小の降伏強度である部材に対して降伏強度以上の応力が加わっていないことを確認できる．

- 結論
  - 以上の結果より，本CanSatはロケット分離時の衝撃(40 [G])が加わった際において機体に損傷を生じることがなく，分離時の衝撃に耐えうることが確認された．

(V9)　通信機電源ON/OFF試験 (※安全面)

- 実施日: 8/9
- 実施責任者 ： 松田
- 充足要求項目 ： R7

- 目的
  ロケットの通信機器に悪影響を与えないよう，キャリア内でWi-FiおよびLoRaの電源が切れることを確認する．

- 試験内容
  - Wi-Fi
    CanSatに搭載されているRaspberry Pi ZERO WにWi-Fiが内蔵されているため，アクセスポイント化し，PCと通信している状態から，Wi-Fiの通信状態をOnからOffに切り替える．Wi-Fiの通信状態がOffに切り替わることを，ターミナル上で自由にOn/Offできることを確認する．
    まず，ローバプログラムのWi-Fi通信モードを変更．ローバプログラムを起動し待機シーケンス(Waiting)に遷移するとWi-Fiの通信状態がOffに切り替わることを，ターミナル上でログが表示されなくなることにより確認する．次に，CanSatに光を照射し放出判定をさせて，シーケンスをWaitingから移行させる．すると，Wi-Fiの通信状態がOnに切り替わり，CanSatとPCが通信できていることを，センサ値やCanSatのシーケンス状態がターミナル上に表示されることにより確認する．

  - LoRa
    LoRaはRaspberry Pi ZERO WのGPIOピンを制御することで，スリープモード（電波の出力が0）にすることが可能である．そのため，該当するGPIOピンを制御し，LoRaのスリープモードを自由にOn/Offできることを確認する．
    まず，ターミナルにコマンドを入力し，LoRaの通信状態をOnからOffに切り替える．LoRaの通信状態がOffに切り替わることを，地上局側のLoRaがCanSatからの信号を受け取れなくなることにより確認する．次に，Wi-Fiの時と同様にして，地上局側のLoRaがCanSatからの信号を受け取れることにより，LoRaの通信状態がOnに切り替わることを確認する．

- 結果
  - Wi-Fi
    実験動画を以下に示す．Waitingのシーケンス内で本番と同様の手順でマイコンのWifiの通信をOffにした．実際にOffにされたかをスマートフォンアプリのWifiの電波を確認するツール(Wifi Analyzer)を用いて確認した．また，Waitingシーケンスが終了した後，自動的にWifiの通信機能がOnになることも同様に確認した．
    https://www.youtube.com/watch?v=MsUMsjhzD4A
  - LoRa
    実験動画を以下に示す．はじめは，CanSatのLoRaから地上局側のLoRaに向けてGPS座標を送信している状況である．CanSat側のターミナルにLoRaをスリープモードにするコマンドを入力したのち，地上局側がCanSat側からGPS座標を受信しなくなることが確認できた．次に，CanSat側のターミナルにLoRaのスリープモードを解除にするコマンドを入力したのち，地上局側がCanSat側からGPS座標を受信できることが確認できた．
    https://youtu.be/Om8lVJKwJjo

- 結論
  - Wi-Fi
    問題なくWi-Fiの通信状態をOnからOffに切り替えと，シーケンス移行後OffからOnへできる．
  - LoRa
    問題なくLoRaの通信状態をOnからOffに切り替えと，シーケンス移行後OffからOnへできる．

(V10) 通信周波数変更試験 (※安全面)

- 実施日: 10/23
- 実施責任者 ： 松田
- 充足要求項目 ： R8
- 目的
  他の無線通信と通信周波数が干渉する可能性がある場合に，使用する通信周波数が変更できることを確認する．

- Wi-Fiに関する試験内容
  はじめの使用チャンネルを決めておき，ノートPC側からRaspberry Piが出すアクセスポイントへ接続しておく．また，外部からの確認として，他のノートPCでWi-Fiアクセスポイントの情報を表示するソフトウェア(Wi-Fi Analyzer)によってRaspberry Piによるアクセスポイントの通信周波数を確認する．
  その後，Raspberry Pi内の設定ファイル(/etc/create_ap.conf)の内容を変更し，Wi-Fiアクセスポイントのチャンネル番号を変更する．アクセスポイントを作成するためのサービス(create_ap)を再起動し，変更を反映する．その後，他のノートPCからアクセスポイントの使用周波数を確認し，外部から見てもアクセスポイントの使用周波数が変更されていることを確認する．

- LoRaに関する試験内容
  はじめの通信周波数を決めておき，CanSatのLoRa通信モジュールとノートPCに接続した別のLoRa通信モジュール(地上局)間の通信が確立することを確認する．その後，地上局側のLoRaモジュールの設定モードを起動し，通信周波数を変更して通信が確立しなくなることを確認する．そして再び，地上局側の通信周波数をはじめの値に戻し，通信が再確立することを確認する．

- 結果
  Wi-FiおよびLoRaの周波数変更試験の結果を表6.20に示す．また，試験のYoutube動画の解説をそれぞれまとめた．

表6.20 通信周波数変更試験の結果

試験	結果	Youtubeリンク
Wi-Fiの周波数変更	成功	https://youtu.be/Z6LreU

		XRvnM
LoRaの周波数変更	成功	https://youtube.com/watch?v=XdTtu2I8lb0

Wi-Fiに関する試験のYouTube動画の解説

0:00~ 試験の説明
0:23~ CanSatのアクセスポイントの周波数(チャンネル番号)確認

→ チャンネル番号 12 であることを確認

0:48~ Wi-FI周波数(チャンネル番号)の変更操作

→ チャンネル番号 12
から チャンネル番号 4 に変更

1:55~ create_apサービスの再起動
3:35~ CanSatとの接続が復帰
3:43~ Wi-Fi周波数が変更されていることを確認

→ チャンネル番号 4 であることを確認

LoRaに関する試験のYouTube動画の解説

0:00~ 試験の説明
0:11~ LoRaとPCのシリアル通信接続
0:17~ ボーレートの設定
0:43~ LoRaの設定画面を開くためにLoRaを再起動
0:52~ LoRaの設定画面
0:55~ LoRaの現在の設定を確認

→ チャンネル番号24 (周波数 920.6MHz)
であることを確認

1:09~ LoRaの周波数変更操作

→ チャンネル番号を24から35に変更

1:25~ 変更後の周波数の確認

→ チャンネル番号35 (周波数 922.8MHz)
であることを確認

○ 結論

Wi-Fi通信モジュール，LoRa通信モジュールともに周波数を変更できることを確認した．

（V11）End-to-End試験 (※安全面, ミッション面)

○ 実施日: 10/21

○ 実施責任者 : 松田

○ 充足要求項目 : R9, R11

○ 目的

CanSatがミッションを行う上での全シーケンスを一連の流れとして遂行できることを確認する．

○ 試験内容

CanSatがミッションを行う上で質量測定，キャリア収納，放出判定，着地判定，パラシュート切り離し，故障検知や最適化サブシーケンスを含むナビゲーション，ゴール判定までの全シーケンスを一連の流れとして自律的に制御できることを確認する．ゴール検知における赤色領域の割合の閾値θ_{red}は15%とする．なお，本試験では，故障検知試験の試験結果と比較するために，左前輪が空転する故障をした状態で実施する．

○ 結果

各試行における結果を表6.21に示す．ここで，θ_{opt}は駆動系最適化後のCanSatにおける各フレームでの走行前と走行後の方位角のずれの平均値を表しており，(V17)最適化実行試験におけるθ_{opt}と同じ意味合いを持つ．また，フルサクセス判定における達成レベルの詳細に関しては，第2章を参照されたい．

表6.21 End-to-End試験の結果

試行回	質量測定	キャリア収納・放出	放出判定	着地判定	パラシュート切り離	故障判定	故障ナビゲーション	ゴールと機体間の距離[m]	θ_{opt} [deg]	フルサクセス判定	YouTubeリンク

					し						
1回目	成功 (1048g)	円滑に収納し自重で放出	成功	成功	成功	成功	成功	0.0	7.5	○	https://youtu.be/NeeEPgq38L8
2回目	成功 (1048g)	円滑に収納し自重で放出	成功	成功	成功	成功	成功	0.7	16.0	×	https://youtu.be/OkET9ChmFYU
3回目	成功 (1048g)	円滑に収納し自重で放出	成功	成功	成功	成功	成功	0.5	6.47	○	https://youtu.be/tk869jz-JC4
全施行における平均値 (標準偏差)								0.4 (0.36)	9.99 (5.23)	△	

- 結論・考察
  - カムバック賞獲得に向けて

    全ての試行において，ゴール判定までの全シーケエンスを問題なく完遂し，半径1m以内でのゴール到達を達成したことから，本CanSatはゴールまで自律走行可能な性能を十分に有していると結論付けられる．しかし，第1試行のような0mゴールを達成するためには更なる対策が必要である．具体的には，ゴール検知にて使用する，ゴールコーン画像識別における赤色領域の割合の閾値θ_{red}（本稿4.2節も合わせて参照されたい）を外光の光量に応じて適切に調整することで0mゴールの達成が期待できる．しかし，最終的な閾値決定においては十分な試行回数に基づく検証を要するため，これに関しては本番までの最優先課題として取り組んでいく所存である．

  - ミッション賞獲得に向けて

    3試行のうち2試行（第1，第3試行）において，最適化実行後のCanSatがフルサクセスを達成する条件(*i.e.*, $\theta_{opt} < 3\epsilon_{deg} (= 9.00°)$) を満たしていることが確認された．一方で，第2試行においてはその条件を満たせていないことから，最適化サブシーケンスの見直しが今後の課題として急務であると結論付けられる．

    ACTSレギュレーションによると，本ミッションの競技時間はCanSat投下後15分以内であると規定されているが，本試験で要した時間は10分39秒 (SD: 2秒)であった．これはすなわち，最適化サブシーケンスでの最適解探索に要する計算時間を3分近く延長しても規定の時間内に収まることを意味する．現段階においては，GAにおける初期個体

数を4，進化過程における世代数の上限を2に設定しているが，これらの設定値をより大きめの値に見積もることで，より効率的な最適解探索が可能となり，θ_{opt}の減少に寄与することが示唆される．しかし，規定時間を超過しない程度まで初期個体数および世代数を増加させる為には，十分な試行回数に基づく検証が肝要である．これに関しても，本番までの最優先課題として検討する．

（V12）制御履歴レポート作成試験(※安全面，ミッション面)

- 実施日: 10/21
- 実施責任者 ： 松田
- 充足要求項目 ： R12
- 目的

  ミッション後，規定された制御履歴レポートを提出するため，得られたログデータから制御履歴レポートを作成する．

- 試験内容

  End-to-End試験で得られた制御履歴に関するログデータから提出用の制御履歴レポートを実際に作成する．

- 制御履歴

  本試験で作成した制御履歴レポートのうちGPSセンサのログをもとにした軌跡のプロットは図6.37，そこに動作の詳細を記入したものが図6.38である．右上がスタート地点で左下がゴール地点である．赤線がGPSが取得した自己位置座標の軌跡であり，赤線上の白丸とそこから伸びる白線がその時点でのゴール方向へ進行するための制御方向を表している．線が長いほどその方向へ前進するようにPID制御する．初め，スタート地点周辺で徘徊するような制御をしているのは，パラシュート切り離しや最適化実行の制御によるものである．ナビゲーション走行開始後は制御方向にそうようなGPS軌跡となっている．

  また，表6.22はEnd-to-End試験1回目の機体の制御履歴をログデータから一部抜粋したものである．ミッションを進めるためのシーケンス移動やセンサ値をもとにした判定箇所は赤字でコメントを加えた．

図6.37 GPSセンサの軌跡

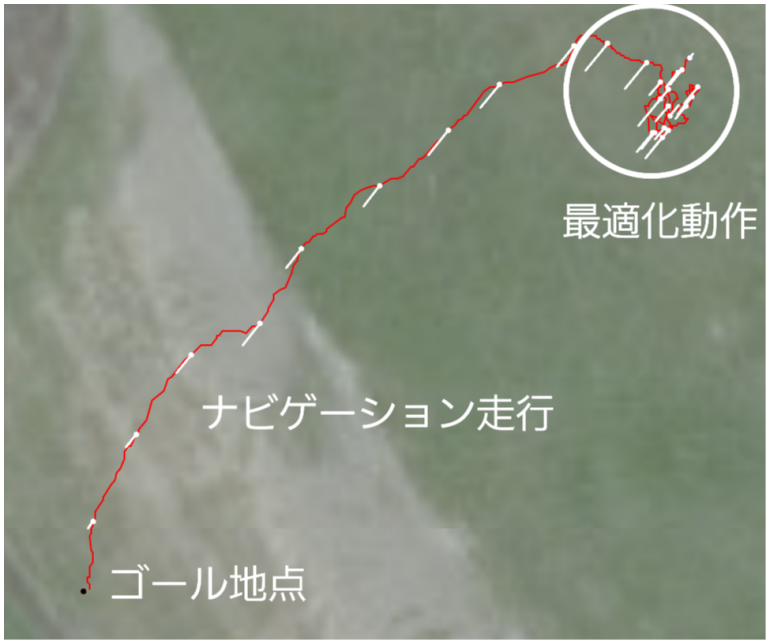


図6.38 GPSセンサ軌跡の動作詳細

表6.22 End-to-End試験1回目 制御履歴レポート

```
GPS Sensor is Ready!
Pressure Sensor is Ready!
Camera is ready
PoseJudgment: ( rollOverThresholdY: 0.65, rollOverThresholdZ: 10, turnOverThresholdX: 9999,
turnOverThresholdZ: 0.5 )

Log file: log/20211011_0825_log_nineaxis1_e2e-dio.csv
Log file: log/20211011_0825_log_gps1_e2e-dio.csv
Log file: log/20211011_0825_log_pressure1_e2e-dio.csv
Log file: log/20211011_0825_log_lora1_e2e-dio.csv

        === Team Salamander ===

      Takadama Lab.  ARLISS 2021

[08:27:32] Sequence: 'waiting' →待機開始
Time: 08:27:33
Light count: 0/10 (LOW)

Time: 08:27:34
Light count: 0/10 (LOW)

<中略>

Time: 08:27:39
Light count: 0/10 (LOW)

Time: 08:27:40
Light count: 0/10 (HIGH)

Time: 08:27:41
Light count: 1/10 (HIGH)

Time: 08:27:42
Light count: 2/10 (HIGH)

<中略>


Time: 08:27:49
Light count: 9/10 (HIGH)

[08:27:50] Sequence: 'falling' →着地待機開始
Time: 08:27:51
Pressure count: 1/10 (1012.18 hPa)
Gyro count:     0/10
GPS position:   35.6442483, 139.5227900

Time: 08:27:52
Pressure count: 2/10 (1012.204 hPa)
Gyro count:     0/10
GPS position:   35.6442483, 139.5227900

Time: 08:27:53
Pressure count: 3/10 (1012.26 hPa)
Gyro count:     0/10
GPS position:   35.6442483, 139.5227900
```

<中略>

Time: 08:27:59
Pressure count: 9/10 (1012.218 hPa)
Gyro count: 0/10
GPS position: 35.6442483, 139.5227850

Time: 08:28:00
Pressure count: 10/10 (1012.188 hPa) →着地判定の条件1つ目(気圧センサによる判定)をクリア
Gyro count: 0/10
GPS position: 35.6442483, 139.5227833

Time: 08:28:01
Pressure count: 11/10 (1012.286 hPa)
Gyro count: 0/10
GPS position: 35.6442467, 139.5227833

Time: 08:28:02
Pressure count: 12/10 (1012.23 hPa)
Gyro count: 0/10 ->CanSatを着地させた
GPS position: 35.6442483, 139.5227833

Time: 08:28:03
Pressure count: 13/10 (1012.228 hPa)
Gyro count: 1/10
GPS position: 35.6442483, 139.5227850

Time: 08:28:04
Pressure count: 14/10 (1012.215 hPa)
Gyro count: 2/10
GPS position: 35.6442500, 139.5227867

Time: 08:28:05
Pressure count: 15/10 (1012.157 hPa)
Gyro count: 3/10
GPS position: 35.6442483, 139.5227867

<中略>

Time: 08:28:11
Pressure count: 21/10 (1012.268 hPa)
Gyro count: 9/10
GPS position: 35.6442500, 139.5227867

Falling completed! (by gyro and pressure)
→ 着地判定の条件2つ目(ジャイロセンサで判定)をクリア
[08:28:12] Sequence: 'para_separating' → パラシュート切り離し動作開始
Para separating... (1/6)
Para separating... (2/6)
Para separating... (3/6)
Para separating start break time [coffee]
Para separating stop break time [ ]
Para separating... (4/6)
Para separating... (5/6)
Para separating... (6/6)
Para separating finished! → パラシュート切り離し動作終了
Log file: log/20211011_0825_log_navigating1_e2e-dio.csv
[08:28:49] Sequence: 'navigating'
0 second past! ** Left Motor was Stopped Programlly!!!! **
[08:28:49] Sequence: 'navigating' > 'magnet_calibrating'

```
[08:28:49] Sequence: 'navigating' > 'magnet_calibrating' > 'waking'
Waking finished!
[08:29:05] Sequence: 'navigating' > 'magnet_calibrating'
MagnetCalibrating: Turning right...
MagnetCalibrating: Turning left...
FailureDetecting: Turning right...
FailureDetecting: Turning left...
MagnetCalibrating: Turning stopped!
min: (34.2, 48, -90.45)
max: (87, 101.55, -75.9)
Command:
nineaxis minmax 34.2 48 -90.45 87 101.55 -75.9
--------------------
filtered min: (36, 49.95, -87)
filtered max: (84.75, 98.25, -77.4)
Command:
nineaxis minmax 36 49.95 -87 84.75 98.25 -77.4
left motor encoder value| before:-511, after:-513, diff:2, threshold:30
right motor encoder value| before:55738, after:64322, diff:8584, threshold:500
** Detected left motor failure!! ** →左モーターの故障検知
[08:29:36] Sequence: 'navigating'
GPS history of stuck_detection_by_gps has been reset because of subsequence interruption!
Initial GPS coordinate is fetched: ( 35.64424, 139.5228 ) →開始gps座標取得

==== MISSION START ==== →ミッションスタート

Time: 08:29:36
Target Azimuth: 217.716 →ゴールへの方角計算
Remaining Distance: 53.236 m (gps) →残りゴールへの距離を計算


Detected Main Left Motor has been Stopped! →左モータの故障により，最適化シーケンスがスタート
python file exist:1
[08:29:45] Sequence: 'navigating' > 'learning_motor_servo_config'

Start Generation ... (gen: 0, population: 5) →進化計算スタート(第0世代)

Start Evaluate. (mlr: 0.666, mrr: 0.841, servo1Angle: 0, servo2Angle: 0.798)
Finish Evaluate. (penalty: 11.9)
Start Evaluate. (mlr: 0.171, mrr: 0.82, servo1Angle: 0, servo2Angle: 0.768)
Finish Evaluate. (penalty: 22.9)
Start Evaluate. (mlr: 0.135, mrr: 1, servo1Angle: 0, servo2Angle: 0.629)
Finish Evaluate. (penalty: 22.4)
Start Evaluate. (mlr: 0.561, mrr: 0.893, servo1Angle: 0, servo2Angle: 0.916)
Finish Evaluate. (penalty: 3.88)
Start Evaluate. (mlr: 0.6, mrr: 0.6, servo1Angle: 0, servo2Angle: 0)
Finish Evaluate. (penalty: 126)
class_obj = ExecGA(
3,3,5,[[0.666122,0.840951,0,0.79844],[0.17114,0.819609,0,0.76823],[0.135327,1,0,0.628871],[0.560961,0.89250
2,0,0.916195],[0.6,0.6,0,0]],[11.9178,22.8537,22.3526,3.8844,125.968])

Start Generation ... (gen: 1, population: 10) →第1世代

Start Evaluate. (mlr: 0.687, mrr: 0.845, servo1Angle: 0, servo2Angle: 0.776)
Finish Evaluate. (penalty: 4.46)
Start Evaluate. (mlr: 0.776, mrr: 0.725, servo1Angle: 0, servo2Angle: 0.75)
Finish Evaluate. (penalty: 4.05)
Start Evaluate. (mlr: 0.725, mrr: 0.796, servo1Angle: 0, servo2Angle: 0.823)
Finish Evaluate. (penalty: 2.44)
Start Evaluate. (mlr: 0.823, mrr: 0.725, servo1Angle: 0, servo2Angle: 0.695)
```

Finish Evaluate. (penalty: 42.9)
Start Evaluate. (mlr: 0.695, mrr: 0.816, servo1Angle: 0, servo2Angle: 0.805)
Finish Evaluate. (penalty: 16.8)
Start Evaluate. (mlr: 0.805, mrr: 0.725, servo1Angle: 0, servo2Angle: 0.75)
Finish Evaluate. (penalty: 2.55)
Start Evaluate. (mlr: 0.725, mrr: 0.795, servo1Angle: 0, servo2Angle: 0.824)
Finish Evaluate. (penalty: 2.53)
Start Evaluate. (mlr: 0.824, mrr: 0.725, servo1Angle: 0, servo2Angle: 0.666)
Finish Evaluate. (penalty: 26.7)
Start Evaluate. (mlr: 0.666, mrr: 0.841, servo1Angle: 0, servo2Angle: 0.798)
Finish Evaluate. (penalty: 3.96)
Start Evaluate. (mlr: 0.798, mrr: 0.725, servo1Angle: 0, servo2Angle: 0.75)
Finish Evaluate. (penalty: 5.91)

Start Generation ... (gen: 2, population: 10)→第2世代(最終世代)

Start Evaluate. (mlr: 0.555, mrr: 0.735, servo1Angle: 0, servo2Angle: 0.75)
Finish Evaluate. (penalty: 4.05)
Start Evaluate. (mlr: 0.725, mrr: 0.566, servo1Angle: 0, servo2Angle: 0.75)
Finish Evaluate. (penalty: 10.2)
Start Evaluate. (mlr: 0.725, mrr: 0.925, servo1Angle: 0, servo2Angle: 0.82)
Finish Evaluate. (penalty: 7.68)
Start Evaluate. (mlr: 0.82, mrr: 0.725, servo1Angle: 0, servo2Angle: 0.608)
Finish Evaluate. (penalty: 43.9)
Start Evaluate. (mlr: 0.608, mrr: 0.589, servo1Angle: 0, servo2Angle: 0.59)
Finish Evaluate. (penalty: 31)
Start Evaluate. (mlr: 0.59, mrr: 0.602, servo1Angle: 0, servo2Angle: 0.75)
Finish Evaluate. (penalty: 13.5)
Start Evaluate. (mlr: 0.725, mrr: 0.725, servo1Angle: 0, servo2Angle: 0.874)
Finish Evaluate. (penalty: 2.3)
Start Evaluate. (mlr: 0.874, mrr: 0.725, servo1Angle: 0, servo2Angle: 0.6)
Finish Evaluate. (penalty: 33.4)
Start Evaluate. (mlr: 0.6, mrr: 0.6, servo1Angle: 0, servo2Angle: 0.75)
Finish Evaluate. (penalty: 0.628)
Start Evaluate. (mlr: 0.725, mrr: 0.561, servo1Angle: 0, servo2Angle: 0.75)
Finish Evaluate. (penalty: 10.1)
Finish Learning. Best is (mlr: 0.6, mrr: 0.6, servo1Angle: 0, servo2Angle: 0.75)

Learning Result Started!!(backSpin: false) 5[sec]→性能評価スタート
SERVO_CENTER: 0.75
MOTOR_L_POWER_OPT_MAGNIFICATION: 0.6
MOTOR_R_POWER_OPT_MAGNIFICATION: 0.6
[08:31:28] Sequence: 'navigating'
GPS history of stuck_detection_by_gps has been reset because of subsequence interruption!
Current Azimuth: 199
Target Azimuth: 199
Max Azimuth Diff: 0
Ave Azimuth Diff: 0
Current Azimuth: 199
Target Azimuth: 199
Max Azimuth Diff: 0
Ave Azimuth Diff: 0

<中略>

Current Azimuth: 196
Target Azimuth: 199
Max Azimuth Diff: 11.6
Ave Azimuth Diff: 7.59
Current Azimuth: 196

Target Azimuth: 199
Max Azimuth Diff: 11.6
Ave Azimuth Diff: 7.53
Finished Check Performance!
Max Azimuth Diff: 11.6 →最大方位角誤差11.6°

Time: 08:31:34 →ナビゲーションスタート
Target Azimuth: 214.415
Remaining Distance: 54.421 m (gps)

Time: 08:31:35
Target Azimuth: 214.506
Remaining Distance: 54.029 m (gps)

Time: 08:31:36
Target Azimuth: 214.617
Remaining Distance: 53.877 m (gps)

<中略>

Time: 08:34:26
Target Azimuth: 196.095
Remaining Distance: 5.982 m (gps)

Time: 08:34:27
Target Azimuth: 196.603
Remaining Distance: 5.805 m (gps)

Time: 08:34:28
Target Azimuth: 195.665
Remaining Distance: 5.584 m (gps)

stuck detection value: 17.8
stuck detection threshold: 5
stuck detection: 0
Time: 08:34:29
Target Azimuth: 194.649
Remaining Distance: 5.366 m (gps)

Time: 08:34:30
Target Azimuth: 193.548
Remaining Distance: 5.149 m (gps)

Close to Goal!

remaining distance: 4.75
[08:34:31] Sequence: 'goal_detecting' →ゴール検知シーケンス
Goal not found! (rate: 0)
Turning to left [deg] ... →ゴールを検出するまで左旋回
Goal not found! (rate: 0)
Turning to left [deg] ...
Goal not found! (rate: 0)

<中略>

Turning to left [deg] ...
Goal not found! (rate: 0)
Turning to left [deg] ...
Goal not found! (rate: 0)
Turning to left [deg] ...
Goal found! (rate: 0.000677) →ゴール発見
Goal azimuth = 207 (current) + 17.6 (red area angle) = 225 →ゴール方角計算
Goal found! (rate: 0.00208)
Goal azimuth = 211 (current) + 17.6 (red area angle) = 229
Goal found! (rate: 0.0215)
Goal azimuth = 224 (current) + 3.22 (red area angle) = 227
Goal found! (rate: 0.721)
Goal! →ゴール

- 結論
  ミッション後，規定された制御履歴レポートを提出するため，得られたログデータから制御履歴レポートを作成できることが確認された．

---

2. ミッション要求を満たすための試験内容

(V13) 着地衝撃試験 (※ミッション面)

- 実施日:
- 実施責任者 ： 松田
- 充足要求項目 ： M1
- 目的
  着地衝撃の付与後であっても，CanSatに損傷が無くミッション継続可能であることを確認する．

- 試験/解析内容
  本試験ではACTS本番で予想される終端速度を再現できる高さから自由落下させることにより，現地の環境と同等の着地衝撃をCanSatに与える．自由落下させる高さ h は終端速度を v，重力加速度を g としたときエネルギー保存則から次式によって導出される．ただし，g =9.81 m/s^2 である．

$$h = \frac{v^2}{2g}$$

想定される終端速度 v を(V4)落下試験で得られた終端速度の最大値の7.0m/sとすると，自由落下させる高さは次式から2.49 mとなる．

$$h = \frac{(7.0)^2}{2 \times 9.81} \fallingdotseq 2.49 \text{ [m]}$$

以上の結果に基づき，本試験では2.5m以上の高さからCanSatを落下させる．以下に実際に落下させる高さの画像を掲載する．図6.39の机の上に乗っている人物の手の高さから落下させる．また，図6.40は図6.39の机の横でメジャーで計測している人物の手元を写しており，メジャーは2.5mを指していることが分かる．

図6.39 投下位置の計測の様子

図6.40 地面から投下位置までの距離

また，衝撃付与後にパラシュート切り離しから走行までのシーケンスを実施し，全てのセンサ・動力系の動作確認，CanSatに破損がないかを確認する．本試験におけるセンサ，モータの出力は以下の基準を満たしていれば正常と扱う．

＜センサ類＞
・気圧センサ：実験開始前に計測した気圧±1[hPa]
・9軸センサ：機体の姿勢が静安時の加速度のノルムが0.95[G]~1.05[G](重力加速度1[G]±許容測定誤差5%，および，機体の姿勢によってxyz軸の値が変化している
・光センサ：通常時に光があたっていると判定(センサ値がHIGHを示す)し，手で光センサを覆ったときに光があたっていないと判定(センサ値がLOWを示す)する
・ブザー：音のON/OFFができる
・GPS：1つ以上の衛星からGPS情報を受信し，その座標が試験の実施地点を指している．

＜モータ類＞
・サーボモータ：上下方向（または左右方向）にスタビライザを制御できる
・モータ：機体が走行可能な程度に回転する

○ 結果

Ⅰ．旧スタビライザを搭載した機体改修前CanSat

以下の表6.23に試行ごとの結果を示す．

表6.23 着地衝撃試験の結果

試行回	CanSatの外的損傷	センサ類	モータ類	シーケンス動作	YouTubeリンク
1回目	なし	正常	正常	正常	https://www.youtube.com/watch?v=ZmiVE6qW6XQ
2回目	なし	正常	正常	正常	https://www.youtube.com/watch?v=ju2WTBGMlQs
3回目	なし	正常	正常	正常	https://www.youtube.com/watch?v=ERizn_ZAXuw
4回目	なし	正常	正常	正常	https://www.youtube.com/watch?v=9T0mfi3n_eQ
5回目	なし	正常	正常	正常	https://www.youtube.com/watch?v=Zqhr1BDoqO0
成功率	100% (5/5)	100% (5/5)	100% (5/5)	100% (5/5)	

Ⅱ. 新スタビライザを搭載した機体改修後CanSat

以下の表6.24に試行ごとの結果を示す.

表6.24 着地衝撃試験の結果(新スタビライザ)

試行回	CanSatの外的損傷	センサ類	モータ類	シーケンス動作	YouTubeリンク
1回目	なし	正常(※)	正常	正常	https://youtu.be/ymN4FgBUPgY
2回目	なし	正常(※)	正常	正常	https://youtu.be/pJ5SWCXXfS4
成功率	100% (2/2)	100% (2/2)	100% (2/2)	100% (2/2)	

※本試験は, 量産した2機体目を用いて試験を実施している. この機体に搭載されている9軸センサはz軸の正方向(機体下方)が弱め(90%程度)に出る個体であるため, 機体を走行姿

勢に置いて9軸センサの出力を確認する際(動画内の9軸センサの確認1回目)に，実験開始前に取った9軸センサの値と比較し，許容測定誤差が5%以内に収まっていることを確認している．なお，2機目の9軸センサの特性については以下の動画を参照されたい．

2機体目の9軸センサの特性についての解説: https://youtu.be/G1uUILkoOwo

- 結論
  パラシュート使用時の終端速度を想定した着地衝撃をCanSatに与えた場合でも，CanSatに損傷がなくミッション継続可能であることが確認された．

(V14) 走行性能確認試験 (※ミッション面)

- 実施日: 8/27
- 実施責任者 : 松田
- 充足要求項目 : M2
- 目的
  CanSatが目的地に到達するために必要な走行性能を有することを確認する．

- 試験内容
  芝生の上で走行させ，途中で停止せずに走行できることを確認する．全車輪が動作する状態と片前輪故障に対する最適化後の2通りで実施する．

- 結果
  走行性能試験の結果を表6.25, 6.26に示す．

表6.25 走行性能試験(全車輪駆動)の結果

試行回	走破性を測る状況	地面の走破	YouTubeリンク
1回目	整地された草地	成功	https://www.youtube.com/watch?v=8wRza4MXRpo
2回目	草丈が高い草地	成功	https://www.youtube.com/watch?v=Hzqj4qpRQfQ
3回目	人の足を障害物とした場合	成功	https://www.youtube.com/watch?v=HSsVyatBDBw
成功率		100% (3/3)	

表6.26 走行性能試験(片車輪故障&最適化後)の結果

試行回	走破性を測る状況	地面の走破	YouTubeリンク
1回目	整地された草地	成功	https://youtu.be/zs7qNFVzx70

2回目	草丈が高い草地	成功	https://www.youtube.com/watch?v=73qr_jrVbE0
3回目	人の足を障害物とした場合	成功	https://www.youtube.com/watch?v=oBGmxIjGWjY
成功率		100% (3/3)	

- 結論
  結果より，様々な状況の草地を走破可能であり，CanSatが目的地に到達するために必要な走行性能を有することを確認した．

(V15) 電力耐久試験 (※ミッション面)

- 実施日: 8/13
- 実施責任者 ： 松田
- 充足要求項目 ： M3
- 目的
  CanSatにミッションを十分に遂行できるバッテリが搭載されている事を確認する．

- 試験内容
  草地でCanSatを約30分（※）の間走行させ，CanSatが止まることなく走行を続けることができることを確認する．CanSatに搭載する電源は，マイコンを含めたセンサ類に供給する電源（マイコン用バッテリー）と駆動系に供給する電源（モータ用バッテリー）の2つを使用し，試験前に十分に充電したものを使用する．
  また，試験を実施した場所を以下の図6.41の写真に示す．

図6.41 電力耐久試験実施場所

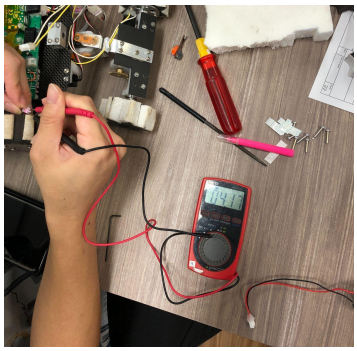
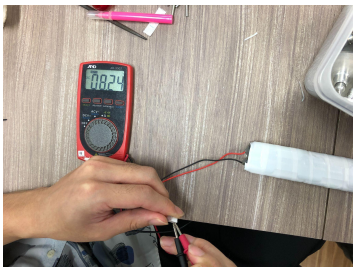
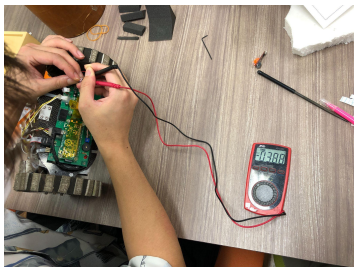
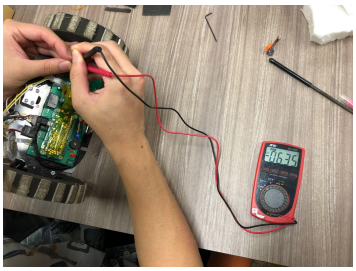
※本大会の競技時間である15分を超えて走行できるよう，多めに見積って本試験は30分実施する.

○ 結果

実験動画：https://youtube.com/watch?v=PPQ_kx8CekY&feature=share

以下の表6.27に耐久試験前後の各バッテリーの電力を示す.

表6.27 耐久試験前後の各バッテリーの電力値

	マイコン用バッテリー	モータ用バッテリー
走行前	4.17V 	8.24V 
走行後	3.88V 	6.35V 

試験の間，Raspberry PiとノートPCの接続が切れることはなく30分以上走行し続けた．また，通常の走行時では使用しないがCanSatがスタックしたときやパラシュートを切り離すときに必要となるスタビライザーの上下方向の動作についても試験の間定期的に実施し，十分に動作することを確認した.

○ 結論

結果より、本CanSatは十分に競技時間内の走行を可能とする電力を有することがわかった。

(V16) 故障検知試験 (※ミッション面)

- 実施日: 8/20
- 実施責任者 : 松田
- 充足要求項目 : M4
- 目的
  (V17)最適化実行試験の目的を満たすための事前実験として，CanSatの片前輪が空転する現象を故障検知によって検知できるか確認する.

- 試験/解析内容
  CanSatの前輪について，
  Case 1: 両前輪が稼働している状態
  Case 2: 左前輪が停止している状態(空転)
  Case 3: 右前輪が停止している状態(空転)
  のそれぞれの状態で故障検知を実行し
  Case 1: 故障と判定しない
  Case 2: 左前輪が故障と検知する
  Case 3: 右前輪が故障と検知する
  ことをそれぞれ確認する.

- 結果
  故障検知試験の結果を表6.28に示す.

表6.28 故障検知試験の結果

試行回	Case 1	Case 2	Case 3	YouTubeリンク
1回目	成功	成功	成功	https://youtu.be/0tdxOC07DuY
2回目	成功	成功	成功	https://youtu.be/n0h7tHOsFiQ
3回目	成功	成功	成功	https://youtu.be/O91pULVosf4
成功率	100%(3/3)	100%(3/3)	100%(3/3)	

- 結論
  本ミッションで想定する全ての故障パターンを，本CanSatの故障検知機構にて識別できることが示された.

(V17) 最適化実行試験 (※ミッション面)

- 実施日: 8/27
- 実施責任者 : 松田

- 充足要求項目 : M5
- 目的

  Cansatの片前輪が空転する状態に陥った時に，GAの最適化によってPID制御に基づく直進走行が可能な状態に移行できるかどうかを確認する．

- 試験/解析内容

  直進停止制御プログラムによってCanSatの片前輪が空転していることを確認した上でGAによるモータ制御最適化プログラムを実行し，CanSatの直進走行の性能を確認する．具体的には，

  Case 1: 両前輪が正常に作動する状態
  Case 2: 左前輪を停止させている状態（最適化前）
  Case 3: 左前輪を停止させている状態（最適化後）

  で機体をセンサなどに頼らず直進させる動作を5秒間ずつ実行し，Case1/Case2/Case3における各フレームでの走行前と走行後の方位角のずれの平均値$\theta_0/\theta_I/\theta_{opt}$を記録する．

  最後に，フルサクセス及びエクストラサクセス達成において必要となる，「故障が生じていない状態における進行方向の最大角度誤差$\boldsymbol{\epsilon}_{deg}$」を，$\boldsymbol{\epsilon}_{deg} = \max \theta_0$として定める．ここで，$\max \theta_0$は全試行にて記録した$\theta_0$の最大値を意味する．

- 結果

  最適化実行試験の結果を表6.29に示す．なお，$\boldsymbol{\epsilon}_{deg} = \max \theta_0 = 3.00$と定めた際のフルサクセスクライテリア達成レベルも合わせて記載する（各レベルにおける達成条件は，第2章を参照されたい）．

表6.29 最適化実行試験の結果

試行回	θ_0[deg]	θ_I[deg]	θ_{opt}[deg]	フルサクセス達成レベル	YouTubeリンク
1回目	2.73	32.34	6.67	○	https://youtu.be/iGvNARV8_wQ
2回目	3.00	24.64	13.39	△	https://youtu.be/T_R1JwnGfi8
3回目	2.20	25.27	6.45	○	https://youtu.be/ZbbXxbGAaXM
4回目	2.66	33.52	26.31	×	https://youtu.be/Lj5x-DKZppA
5回目	2.94	42.98	16.07	×	https://youtu.be/fhhRPXCpKnU
全試行における	2.71	31.75	13.78	△	

平均値 （標準偏差）	(0.32)	(7.45)	(8.17)		

○ 結論・考察

全試行において，GAに基づく最適化が，左前輪が停止している本CanSatにおける直進走行性能の改善（*i.e.,* $\theta_I > \theta_{opt}$）に寄与していることが表6.20より示された．一方で，フルサクセス達成レベルが各試行において大きく異なることから，最適化実行後のCanSatがフルサクセスを達成する(*i.e.*, $\theta_{opt} < 3\epsilon_{deg} (= 9.00°)$) 直進走行性能を十分に担保しているとはいえず，本CanSatのさらなる改良が必要であると結論付けられる．この原因として，全試行におけるθ_{opt}の標準偏差が大きいことから，GAオペレータの中でも特に強いランダム性が内在する突然変異が過度に影響を及ぼしていると現段階では考えられる．シュミレータ上では初期個体30個体を，1000世代いう長い期間を経て進化させるため，個体群の多様性を生み出すという意味合いでは突然変異は有効であるものの，実機における初期個体は4個体かつ2世代のみの短い進化過程を経るため，突然変異はかえって初期優良個体の性能を著しく低下させる可能性がある．この問題を抑制するために，(i)実機における最適化シーケンスでは突然変異における最大変化量および突然変異確率をより低い値に設定することを検討するとともに，(ii)シュミレーター上での初期優良個体の選定の見直しを図ることで，直進走行性能の改善に寄与することが示唆される．

（V18）反転・横転復帰試験 (※ミッション面)

○ 実施日: 10/7

○ 実施責任者 : 松田

○ 充足要求項目 : M6

○ 目的

CanSatが走行中に反転・横転をした時に,走行可能な姿勢に復帰できることを確認する.

○ 試験/解析内容

■ 反転復帰

CanSatを反転させた状態から反転復帰シーケンスのプログラムを実行し,走行可能な姿勢に復帰させる.

■ 横転復帰

CanSatを横転させた状態から横転復帰シーケンスのプログラムを実行し.走行可能な姿勢に復帰させる.

○ 結果

■ 反転復帰

左前輪及び右前輪が空転している2通りの場合における試験を，各3回実施した．それらの結果を表6.30に示す．

表6.30 反転・横転復帰試験の結果

試行回	反転復帰	YouTubeリンク	反転復帰	YouTubeリンク

	（両前輪稼働）		（片前輪故障）	
1回目	成功	https://youtu.be/KCeEqHKfUMo	成功	https://youtu.be/bczng0NAJ4E
2回目	成功	https://youtu.be/b8Fea6lqN-8	成功	https://youtu.be/Fg0KVfeXTB4
3回目	成功	https://youtu.be/f1PV2CLoM4o	成功	https://youtu.be/J6yP_DcIpLY
成功率	100%(3/3)		100%(3/3)	

■ 横転復帰

試験は全5回実施した．それらの結果を表6.31に示す．

表6.31 横転復帰試験の結果

試行回	横転復帰	YouTubeリンク
1回目	成功	https://youtu.be/TL7mPkOJoVw
2回目	成功	https://youtu.be/9nrB-zv3HN0
3回目	成功	https://youtu.be/c3UdxyGnvFE
4回目	成功	https://youtu.be/Hq_SDfBCziE
5回目	成功	https://youtu.be/SGfXkZReP0E
成功率	100%(5/5)	

○ 結論

全ての試行において，CanSatが走行中に反転・横転をした際，無事走行可能な姿勢に復帰できることが確認された．

(V19) ナビゲーション試験 (※ミッション面)

○ 実施予定日: 8/28

○ 実施責任者 ： 松田

○ 充足要求項目 : M7

○ 目的

GPSセンサと地磁気センサの情報をもとに走行して，ゴール地点付近へ到達し，走行制御が終了できることを示す．ただし，本試験では故障検知，最適化のシーケンスおよびゴール検知は除く．

○ 試験内容

多摩川河川敷の芝生上にゴール地点を設定し，そこから50m程度離れた地点(スタート地点とする)で制御プログラムを開始し，PID制御によるゴール地点へのナビゲーション制御を開始する．ナビゲーション制御が終了したら，ゴール地点との距離をメジャーを使用して測定・記録する．上記の流れを，

Case 1: 両前輪が稼働している場合

Case 2: 片前輪が故障している(稼働していない)場合

の，それぞれの状態で3回ずつ実施する．

○ 結果

Case1/Case2における本試験の結果を，表6.32/表6.33に示す．

表6.32 ナビゲーション試験の結果(Case1: 両前輪が稼働している場合)

	制御終了後のゴール距離(メジャーによる実測値)	Youtubeリンク
1回目	5.20m	https://www.youtube.com/watch?v=8UabpxHQM0E
2回目	4.40m	https://www.youtube.com/watch?v=ixGcSISd3EA
3回目	4.05m	https://www.youtube.com/watch?v=LgiN8P7mmcI

表6.33 ナビゲーション試験の結果(Case2: 片前輪が故障している場合)

	制御終了後のゴール距離(メジャーによる実測値)	Youtubeリンク
1回目	3.10m	https://youtu.be/4bG5VS1ohfo

2回目	0.10m	https://youtu.be/hL6zLJ7twFw
3回目	5.10m	https://youtu.be/EbHNcDsgmKk

- 結論
  走行時にGPSと地磁気センサを用いることなくゴール地点付近へ到達し，走行制御が正常に終了できることが確認された．

(V20) ゴール検知試験 (※ミッション面)

- 実施予定日: 8/27
- 実施責任者 : 松田
- 充足要求項目 : M8
- 目的
  CanSatが目的地に設置されたゴールコーンを検知し0mゴールが可能であることを確認する．

- 試験内容
  まず，ゴールのGPS座標を設定し，その地点にゴールコーンを設置する．次に，図6.42，図6.43のように，CanSatをゴールのGPS座標から半径10m以内に設置し，精密ゴール用の自律プログラムを実行する．CanSatがゴールコーンに対して，0mゴール可能であることを確認する．上記の流れを，
  Case 1: 両前輪が稼働している場合
  Case 2: 片前輪が故障している(稼働していない)場合
  の，それぞれの状態で3回ずつ実施する．ゴール検知における赤色領域の割合の閾値θ_{red}は35%に設定する．

図6.42 ゴール検知試験の外観

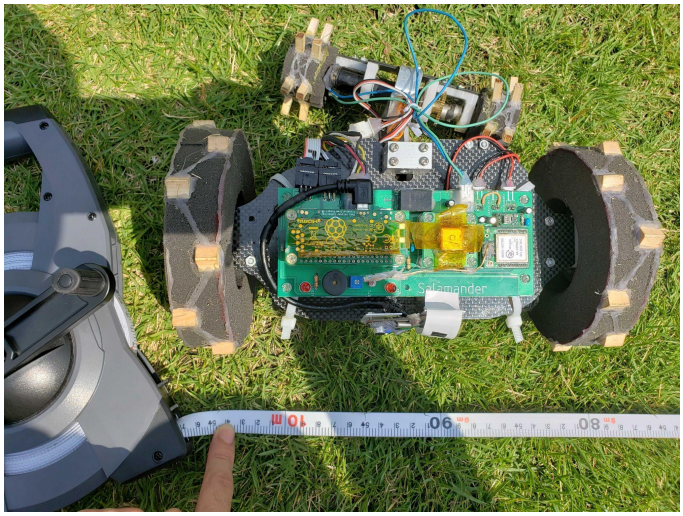


図6.43 CanSat – ゴール間の距離(10m)

○ 結果

Case1/Case2における本試験の結果を，表6.34/表6.35に示す．

表6.34 ゴール検知試験の結果（Case1: 両前輪が稼働している場合）

回数	実験動画	0mゴール検知	機体とゴール間の距離
1回目	https://youtu.be/tqJC7Yu3hLE	成功	0.30m
2回目	https://youtu.be/ShcWt_p0WeU	成功	0.30m
3回目	https://youtu.be/uxSm9fiiSJo	成功	0.00m

表6.35 ゴール検知試験の結果（Case2: 片前輪が故障している場合）

回数	実験動画	0mゴール検知	機体とゴール間の距離
1回目	https://youtu.be/tkEhRVvBByQ	成功	0.19m
2回目	https://youtu.be/An7oxXJaTqY	成功	0.13m
3回目	https://youtu.be/pT3M99I9rgs	成功	0.00m

○ 結論

上記の結果より，本CanSatはゴールを適切に検知し，最大限接近できる性能を持つことが示された．

第7章　　工程管理、ガントチャート(スプレッドシートを推奨)

1. 各担当(ハード・ソフト・全体などの進行状況・予定を記入)

担当	工程	予定日	完了日(実績)
チーム全体	定例MTG	毎週 水曜日	---
	BBM完成	5/21	5/21
	PM完成	7/12	8/22
	準静荷重試験	8/13	10/12
	予備審査書提出	~7/25	7/25
	End-to-Ends試験 (本審査書提出後も試験を繰り返す)	9/20~11/20	10/21
	本審査書提出	10/24	10/24
	FM完成	9/10	10/1
ソフトウェア	制御系完成	6/14	6/21
	統合テスト1完了	6/17	6/24
	通信系完成	6月下旬	6/30
	統合テスト2完了	7/7	8/22
	制御履歴レポート試験	9/20~10/20	10/21
	故障検知試験	8/13	8/20
	最適化実行試験	8/13	8/27
	反転・横転復帰試験	8/16	10/7
	通信機電源ON/OFF試験	8/9	8/9
	通信周波数変更試験	6/30	10/23
	長距離通信試験	7/21	7/23
	ゴール検知試験	8/19	8/27
	ナビゲーション走行試験	8/12	8/28
ハードウェア	概形決定	~6/11	6/30
	BBM用機体作成	~5/21	5/21

	PM用機体作成	6/2~7/12	8/22
	落下試験	6/16	6/23
	走行性能確認試験	8/6	8/27
	開傘衝撃試験	8/9	10/15
	着地衝撃試験	8/9	
	質量試験	8/14	10/16
	キャリア収納試験	8/14	10/16
	量産	8/2~10/30	10/22
	電力耐久試験	8/5	8/13
回路	回路図設計	6/14	6/25
	回路作成	7/15	7/21
	回路量産	7/26~	8/2(2枚目作成)

2. ガントチャート

以下のリンクにSalamanderの使用するガントチャートのリンクを掲載する．また，図7.1はガントチャートの例である．画像中央から左側はタスクの一覧であり，項目は左からタスクNo.・終了目標までの残り日数・優先度・タスク項目・タスク内容・進捗状況・担当者・着手/完了日・予想工数・開始/終了目標の日付が記載されている．特に進捗状況はBacklog(未着手)・InProgress(作業中)・Test&Confirming(テスト中)・Done(完了)からなる．Backlog・InProgress・Test&Confirmingの場合は終了目標までの残り日数が多いほど緑色であり，日数が減ってくると黄色からピンク色に変わり，終了目標期日を超えると赤色になる．また，右側は開始/終了目標を記入したカレンダーである．

https://docs.google.com/spreadheets/d/1o2gak40u5sCfG0BUOJCIZhGw4hmBOPNFf8TklnuEmRc/edit#gid=0

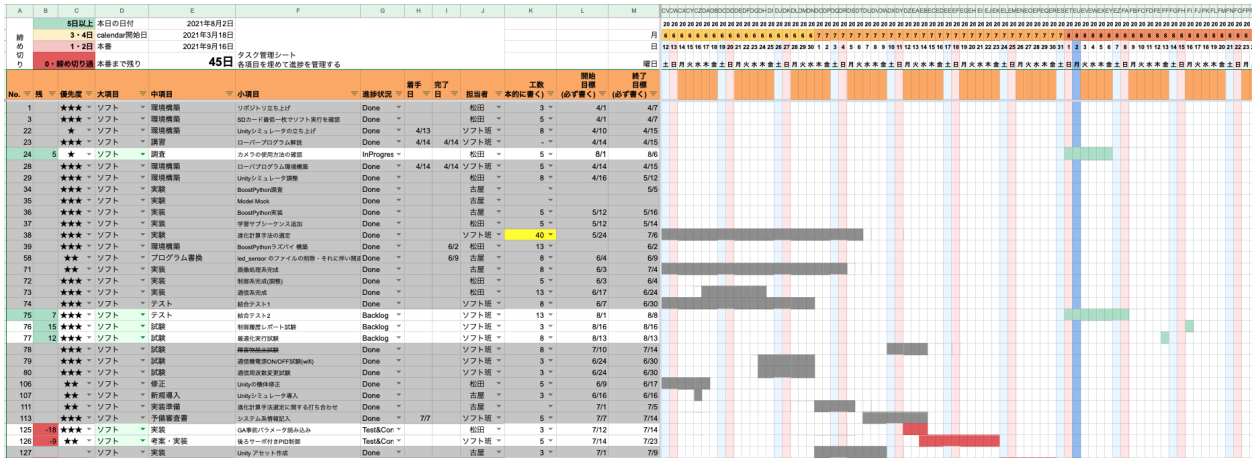


図7.1 ガントチャートの例

第8章　　大会結果報告

1. 目的

本CanSatがキャリア放出判定からゴール判定およびミッション達成までのシーケンスを自律して制御できるかを確認する．それとともに，まずはモータの故障検知をする．そして，故障状態での走行姿勢を進化計算による最適化によって修正し故障前と同様の走行性能を獲得できるか検証する．

2. 結果

ACTSにおける大会結果をまとめる．表8.1はサクセスクライテリアの達成度を示している．

表8.1　サクセスクライテリアの達成度

	ミニマムサクセス	フルサクセス	エクストラサクセス
1回目	○	-	×
2回目	○	○	×

- 投下1回目

<サクセスクライテリア達成状況>

ミニマムサクセス：　　○

フルサクセス：　　-

エクストラサクセス：　　X

<制御概要>

2021年11月20日

（CanSat投下時刻は13:42であったが，RaspberryPiにおける時刻設定と実際の時刻に差があり，RaspberryPi内での投下時刻は21:20となってしまっている．以下，RaspberryPiにおけるタイムスタンプを記載する）

21:29:19 プログラム起動

21:33:47 放出判定

21:33:52 着地判定

21:34:09 パラシュート切り離し完了

21:34:39 右主輪モータを意図的に停止

21:35:32 故障検知開始->成功

21:36:03 現在位置のGPS座標取得（35.41268，138.5933）

21:36:13 最適化実行

21:38:04 ナビゲーション開始

21:39:24 スタック状態に陥り，CanSat前進せず

21:49:04 駆動用電池の電圧降下によってCanSat駆動系が制御不能となり，完全停止．競技時間が15分経過したため終了．

<ナビゲーション軌跡>

図8.1 投下1回目の制御開始・終了地点

図8.2 投下1回目の制御方角詳細

<制御履歴ログ>

```
         === Team Salamander ===
Waiting Mode :light, 1 ->光センサを使用
Wi-Fi has been stopped!->Wi-Fi off
[1m[21:29:19] [36mSequence: [32m'waiting'->待機
[0mTime: 21:29:19
Light count: 0/10 (LOW)

(略)

Time: 21:33:47
Light count: 9/10 (HIGH)->放出判定の条件(光検出). 10カウント

Wi-Fi has been restarted!->Wi-Fi on
[1m[21:33:51] [36mSequence: [32m'falling'
[0mTime: 21:33:52
Pressure count: 1/10 (915.314 hPa)->着地判定の条件1つ目(気圧センサによる判定). 10カウント
Gyro count:     1/10->着地判定の条件2つ目(ジャイロセンサによる判定). 10カウント
GPS position:   35.4127167, 138.5931817

(略)

Falling completed! (by gyro and pressure)->着地判定完了
[1m[21:34:01] [36mSequence: [32m'para_separating'-> パラシュート切り離し動作開始
[0mPara separating... (1/6)
Para separating... (2/6)
```

```
Para separating... (3/6)
Para separating start break time [coffee]
Para separating stop break time [ ]
Para separating... (4/6)
Para separating... (5/6)
Para separating... (6/6)
Para separating finished!-> パラシュート切り離し動作終了
Log file: log/20211118_2129_log_navigating1.csv
[1m[21:34:39] [36mSequence: [32m'navigating'
[0m0 second past! ** Right Motor was Stopped Programlly!!!! ** -> 右主輪モータを意図的に停止(空転)
[1m[21:34:39] [36mSequence: [32m'navigating' > 'magnet_calibrating'-> 地磁気のキャリブレーション
[0m[1m[21:34:39] [36mSequence: [32m'navigating' > 'magnet_calibrating' > 'waking'-> 反転判定
[0mHOLD_SERVO Waking failed! -> 反転失敗
Retrying...
Waking finished!-> 反転復帰成功
[1m[21:35:32] [36mSequence: [32m'navigating' > 'magnet_calibrating'
[0mMagnetCalibrating: Turning right...-> 地磁気のキャリブレーション
MagnetCalibrating: Turning left...
FailureDetecting: Turning right...-> 故障検知
FailureDetecting: Turning left...
MagnetCalibrating: Turning stopped!
min: (-4.65, -89.85, -21.9)
max: (25.2, -57.15, -14.55)
Command:
nineaxis minmax -4.65 -89.85 -21.9 25.2 -57.15 -14.55
--------------------
filtered min: (-2.85, -86.4, -19.95)
filtered max: (25.2, -59.25, -16.2)
Command:
nineaxis minmax -2.85 -86.4 -19.95 25.2 -59.25 -16.2
left motor encoder value| before:2809, after:1787, diff:1022, threshold:10-> モータのエンコード値を取得
right motor encoder value| before:-2484, after:-2484, diff:0, threshold:100
** Detected right motor failure!! **-> 故障検知成功(右側モータの故障を確認)
[1m[21:36:03] [36mSequence: [32m'navigating' -> ナビゲーション開始
[0mGPS history of stuck_detection_by_gps has been reset because of subsequence interruption!
Initial GPS coordinate is fetched: ( 35.41268, 138.5933 ) -> 現在位置のGPS座標取得

==== MISSION START ====

Time: 21:36:03
Target Azimuth: 300.697 -> 制御方向
Remaining Distance: 59.610 m (gps) -> GPSによるゴールまでの残り距離


Detected Main Right Motor has been Stopped!
python file exist:1
[1m[21:36:13] [36mSequence: [32m'navigating' > 'learning_motor_servo_config'
```

```
[0mwheel malfuction: right

Start Generation ... (gen: 0, population: 5)  -> GAによる最適化（第0世代，5個体）

Start Evaluate. (mlr: 0.841, mrr: 0.666, servo1Angle: 0, servo2Angle: 0.142) -> GAの個体（右主輪の出力・左主輪の出力，サーボ1の角度，サーボ2の角度）
Finish Evaluate. (penalty: 16.7) -> 評価値（角度誤差）
Start Evaluate. (mlr: 0.5, mrr: 0.5, servo1Angle: 0, servo2Angle: 0.234)
Finish Evaluate. (penalty: 12.6)
Start Evaluate. (mlr: 0.67, mrr: 0.67, servo1Angle: 0, servo2Angle: 0.16)
Finish Evaluate. (penalty: 21.8)
Start Evaluate. (mlr: 0.893, mrr: 0.561, servo1Angle: 0, servo2Angle: 0.179)
Finish Evaluate. (penalty: 11)
Start Evaluate. (mlr: 0.5, mrr: 0.5, servo1Angle: 0, servo2Angle: 0)
Finish Evaluate. (penalty: 18.1)
class_obj = ExecGA(
3,3,5,[[0.840951,0.666122,0,0.142437],[0.5,0.5,0,0.23428],[0.669975,0.669975,0,0.1601],[0.892502,0.560961,0,0.179126],[0.5,0.5,0,0]],[16.6518,12.5771,21.7842,11.0467,18.1064])
-> 最適化後の個体群

Start Generation ... (gen: 1, population: 10) -> GAによる最適化（第1世代，10個体）
(略)
Start Evaluate. (mlr: 0.559, mrr: 0.5, servo1Angle: 0, servo2Angle: 0)
Finish Evaluate. (penalty: 22.3)

Start Generation ... (gen: 2, population: 10) -> GAによる最適化（第2世代，10個体）
(略)
Start Evaluate. (mlr: 0.57, mrr: 0.893, servo1Angle: 0, servo2Angle: 0)
Finish Evaluate. (penalty: 24.9)
Finish Learning. Best is (mlr: 0.565, mrr: 0.559, servo1Angle: 0, servo2Angle: 0.346)-> GA終了
Learning Result Started!!(backSpin: false) 5[sec]
SERVO_CENTER: 0.346 -> サーボ中央の最適位置
MOTOR_L_POWER_OPT_MAGNIFICATION: 0.565-> 左モータの最適出力
MOTOR_R_POWER_OPT_MAGNIFICATION: 0.559-> 右モータの最適出力
[1m[21:38:46] [36mSequence: [32m'navigating'-> ナビゲーション開始
[0mGPS history of stuck_detection_by_gps has been reset because of subsequence interruption!
Current Azimuth: 108-> サクセスクライテリア用の性能評価開始
Target Azimuth: 108
Max Azimuth Diff: 0.296
Ave Azimuth Diff: 0.296
(略)
Current Azimuth: 110
Target Azimuth: 108
Max Azimuth Diff: 4.26
Ave Azimuth Diff: 1.64
Current Azimuth: 110
Target Azimuth: 108
Max Azimuth Diff: 4.26
```

```
Ave Azimuth Diff: 1.65
Finished Check Performance!
Max Azimuth Diff: 4.26-> 最大角度誤差

Time: 21:38:52
Target Azimuth: 304.394
Remaining Distance: 71.279 m (gps)


Time: 21:38:53
Target Azimuth: 304.394
Remaining Distance: 71.279 m (gps)
(略)

Time: 21:39:25
Target Azimuth: 303.986
Remaining Distance: 72.029 m (gps)


stuck detection value: 1.3
stuck detection threshold: 2.5
stuck detection: 1-> スタック判定
[1m[21:39:26] [36mSequence: [32m'navigating' > 'escaping'-> エスケーピング（轍脱出）開始
[0mRetrying escaping... (1/4)
Retrying escaping... (2/4)
Retrying escaping... (3/4)
Retrying escaping... (4/4)
[1m[21:39:50] [36mSequence: [32m'navigating'
[0mGPS history of stuck_detection_by_gps has been reset because of subsequence interruption!
Time: 21:39:50
Target Azimuth: 303.196
Remaining Distance: 70.830 m (gps)

(略)
stuck detection value: 1.51
stuck detection threshold: 2.5
stuck detection: 1
SandMountain escaping.'-> エスケーピング（モグラ山脱出）開始
[1m[21:40:30] [36mSequence: [32m'navigating' > 'turning_and_escaping_sand_mountain'
[0m[1m[21:40:46] [36mSequence: [32m'navigating'
[0mGPS history of stuck_detection_by_gps has been reset because of subsequence interruption!
Time: 21:40:46
Target Azimuth: 301.019
Remaining Distance: 70.573 m (gps)

(略)
Time: 21:49:04
Target Azimuth: 298.466
Remaining Distance: 57.618 m (gps)
```

```
stuck detection value: 1.75
stuck detection threshold: 2.5
stuck detection: 1-> スタック判定
[1m[21:49:04] [36mSequence: [32m'navigating' > 'escaping'-> エスケーピング（轍脱出）開始
[0mRetrying escaping... (1/4)
Retrying escaping... (2/4)
Retrying escaping... (3/4)
駆動用電池の電圧降下によってCanSat駆動系が制御不能となり，完全停止．
```

- 投下2回目
  <サクセスクライテリア達成状況>
  ミニマムサクセス：　　○
  フルサクセス：　　-
  エクストラサクセス：　　X
  <制御概要>
  2021年11月21日
  （CanSat投下時刻は12:56であったが，RaspberryPiにおける時刻設定と実際の時刻に差があり，RaspberryPi内での投下時刻は11:23となってしまっている．以下，RaspberryPiにおけるタイムスタンプを記載する）

  11:23:38 プログラム起動

  11:27:18 放出判定

  11:27:32 着地判定

  11:28:10 パラシュート切り離し完了

  11:28:26 故障検知開始->成功(故障していないことを検知)

  11:29:06 現在位置のGPS座標取得（35.41268，138.5933）

  11:29:06 ナビゲーション開始

  11:29:31 ゴール20m以内に接近->右主輪モータを意図的に停止

  11:29:35 故障検知開始->成功(右主輪の故障を検知)

  11:30:56 最適化実行

  11:35:24 ナビゲーション再開

11:42:08 競技時間15分が経過したため，ゴールまでの距離を計測．実測10.80m（競技終了）

11:47:01 スタック検知->スタック脱出を繰り返し，スタックに関する動作の検証は十分と判断したため，機体をゴール5m付近まで直接持ち運び移動

11:47:02 ゴール検知開始

11:52:06 ゴールコーンから数センチ地点でゴール判定しプログラムを終了

<ナビゲーション軌跡>

図8.3 投下2回目の制御開始・終了地点

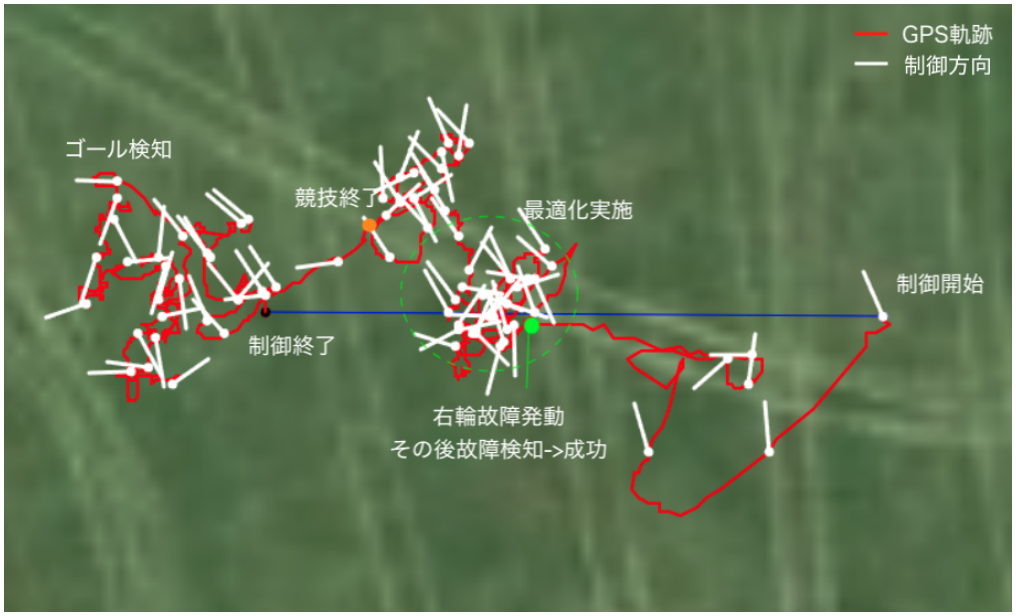


図8.4 投下2回目の制御方角詳細

<制御履歴ログ>

```
       === Team Salamander ===
Waiting Mode :light, 1 ->光センサを使用
Wi-Fi has been stopped!->Wi-Fi off
[1m[11:23:38] [36mSequence: [32m'waiting'->待機
[0mTime: 11:23:38
Light count: 0/10 (LOW)

(略)

Time: 11:27:18
Light count: 9/10 (HIGH)->放出判定の条件(光検出). 10カウント

Wi-Fi has been restarted!->Wi-Fi on
[1m[11:27:23] [36mSequence: [32m'falling'
[0mTime: 11:27:24
Pressure count: 1/10 (918.282 hPa)->着地判定の条件1つ目(気圧センサによる判定). 10カウント
Gyro count:     1/10->着地判定の条件2つ目(ジャイロセンサによる判定). 10カウント
GPS position:   35.4129600, 138.5930967

(略)

Time: 11:27:32
Pressure count: 9/10 (918.3987 hPa)
Gyro count:     9/10
GPS position:   35.4129550, 138.5930933

Falling completed! (by gyro and pressure)->着地判定完了
[1m[11:27:33] [36mSequence: [32m'para_separating'-> パラシュート切り離し動作開始
```

```
[0mPara separating... (1/6)
Para separating... (2/6)
Para separating... (3/6)
Para separating start break time [coffee]
Para separating stop break time [ ]
Para separating... (4/6)
Para separating... (5/6)
Para separating... (6/6)
Para separating finished!-> パラシュート切り離し動作終了
Log file: log/20211121_1123_log_navigating1.csv
[1m[11:28:10] [36mSequence: [32m'navigating'
[0m[1m[11:28:10] [36mSequence: [32m'navigating' > 'magnet_calibrating'-> 地磁気のキャリブレーション
[0m[1m[11:28:10] [36mSequence: [32m'navigating' > 'magnet_calibrating' > 'waking'-> 反転判定
[0mWaking finished!-> 反転復帰成功
[1m[11:28:26] [36mSequence: [32m'navigating' > 'magnet_calibrating'
[0mMagnetCalibrating: Turning right...-> 地磁気のキャリブレーション
MagnetCalibrating: Turning left...
MagnetCalibrating: Turning right...
MagnetCalibrating: Turning left...
FailureDetecting: Turning right...-> 故障検知
FailureDetecting: Turning left...
MagnetCalibrating: Turning stopped!
min: (-6.6, -97.5, -28.2)
max: (46.2, -35.85, -8.1)
Command:
nineaxis minmax -6.6 -97.5 -28.2 46.2 -35.85 -8.1
--------------------
filtered min: (-4.5, -94.8, -20.85)
filtered max: (43.5, -43.95, -9.9)
Command:
nineaxis minmax -4.5 -94.8 -20.85 43.5 -43.95 -9.9
left motor encoder value| before:-676, after:-2012, diff:1336, threshold:10-> モータのエンコード値を取得
right motor encoder value| before:628, after:1669, diff:1041, threshold:100
MagnetCalibrating has been finished!-> 故障検知成功(モータが故障してないことを確認)
[1m[11:29:06] [36mSequence: [32m'navigating'-> ナビゲーション開始
[0mGPS history of stuck_detection_by_gps has been reset because of subsequence interruption!
Initial GPS coordinate is fetched: ( 35.41294, 138.593 )

==== MISSION START ====

Time: 11:29:06
Target Azimuth: 274.156 -> 制御方向
Remaining Distance: 30.773 m (gps)-> GPSによるゴールまでの残り距離

(略)

Time: 11:29:31
```

```
Target Azimuth: 272.093
Remaining Distance: 20.424 m (gps)


20 [m] close to goal ** Right Motor was Stopped Programlly!!!! ** -> ゴール20m以内に達したため右主輪モータを意図的に停止(空転)

Time: 11:29:32
Target Azimuth: 272.672
Remaining Distance: 19.978 m (gps)


Time: 11:29:33
Target Azimuth: 272.756
Remaining Distance: 19.374 m (gps)


Time: 11:29:34
Target Azimuth: 272.822
Remaining Distance: 18.921 m (gps)


[1m[11:29:35] [36mSequence: [32m'navigating' > 'magnet_calibrating'
[0mMagnetCalibrating: Turning right...-> 地磁気のキャリブレーション(定期実行)
MagnetCalibrating: Turning left...
MagnetCalibrating: Turning right...
MagnetCalibrating: Turning left...
FailureDetecting: Turning right...-> 故障検知
FailureDetecting: Turning left...
MagnetCalibrating: Turning stopped!
min: (-4.35, -94.3, -23.6)
max: (47.1, -43.9, -5.25)
Command:
nineaxis minmax -4.35 -94.3 -23.6 47.1 -43.9 -5.25
--------------------
filtered min: (-1.5, -91.8, -19.3)
filtered max: (45.1, -46.8, -7.2)
Command:
nineaxis minmax -1.5 -91.8 -19.3 45.1 -46.8 -7.2
left motor encoder value| before:-13176, after:-14450, diff:1274, threshold:10-> モータのエンコード値を取得
right motor encoder value| before:15301, after:15302, diff:1, threshold:100
** Detected right motor failure!! **-> 故障検知成功(右側モータの故障を確認)
[1m[11:30:09] [36mSequence: [32m'navigating'-> ナビゲーション再開
[0mGPS history of stuck_detection_by_gps has been reset because of subsequence interruption!
Time: 11:30:09
Target Azimuth: 276.707
Remaining Distance: 17.506 m (gps)
```

```
Detected Main Right Motor has been Stopped!
python file exist:1
[1m[11:30:56] [36mSequence: [32m'navigating' > 'learning_motor_servo_config'
[0mwheel malfuction: right

Start Generation ... (gen: 0, population: 5)-> GAによる最適化（第0世代，5個体）

Start Evaluate. (mlr: 0.841, mrr: 0.666, servo1Angle: 0, servo2Angle: 0.142)-> GAの個体（右主輪の出力・左主輪の出力，サーボ1の角度，サーボ2の角度）
Finish Evaluate. (penalty: 109)-> 評価値（角度誤差）
azimuthDiffThreshold:30 encoderDiff:551 encoderThreshold:100-> GA用スタック検知(角度誤差が少ないかつ故障輪のエンコーダの増加値が極端に小さい時はスタック判定->スタック脱出動作->その個体評価をリセットし再評価)
Start Evaluate. (mlr: 0.5, mrr: 0.5, servo1Angle: 0, servo2Angle: 0.234)
Finish Evaluate. (penalty: 7)
azimuthDiffThreshold:30 encoderDiff:5 encoderThreshold:100
stack detected!!-> スタックを検知
[1m[11:31:08] [36mSequence: [32m'navigating' > 'learning_motor_servo_config' > 'escaping'
[0mRetrying escaping... (1/6)-> エスケーピング（轍脱出）開始
Retrying escaping... (2/6)
Retrying escaping... (3/6)
Retrying escaping... (4/6)
Retrying escaping... (5/6)
Retrying escaping... (6/6)
[1m[11:31:29] [36mSequence: [32m'navigating' > 'learning_motor_servo_config'-> 最適化再開
[0mStart Evaluate. (mlr: 0.5, mrr: 0.5, servo1Angle: 0, servo2Angle: 0.234)-> スタック検知前の個体の評価をリセットし再評価
Finish Evaluate. (penalty: 101)
azimuthDiffThreshold:30 encoderDiff:257 encoderThreshold:100
Start Evaluate. (mlr: 0.67, mrr: 0.67, servo1Angle: 0, servo2Angle: 0.16)
Finish Evaluate. (penalty: 33.5)
azimuthDiffThreshold:30 encoderDiff:8 encoderThreshold:100
Start Evaluate. (mlr: 0.893, mrr: 0.561, servo1Angle: 0, servo2Angle: 0.179)
Finish Evaluate. (penalty: 55.3)
azimuthDiffThreshold:30 encoderDiff:80 encoderThreshold:100
Start Evaluate. (mlr: 0.5, mrr: 0.5, servo1Angle: 0, servo2Angle: 0)
Finish Evaluate. (penalty: 7.91)
azimuthDiffThreshold:30 encoderDiff:147 encoderThreshold:100
class_obj = ExecGA(
3,3,5,[[0.840951,0.666122,0,0.142437],[0.5,0.5,0,0.23428],[0.669975,0.669975,0,0.1601],[0.892502,0.560961,0,0.179126],[0.5,0.5,0,0]],[109.159,100.51,33.5208,55.2647,7.91058])-> 最適化後の個体群

Start Generation ... (gen: 1, population: 10) -> GAによる最適化（第1世代，10個体）
(略)
Start Evaluate. (mlr: 0.519, mrr: 0.561, servo1Angle: 0, servo2Angle: 0.214)
Finish Evaluate. (penalty: 134)
azimuthDiffThreshold:30 encoderDiff:387 encoderThreshold:100
```

```
Start Generation ... (gen: 2, population: 10)-> GAによる最適化（第2世代，10個体）
(略)
Start Evaluate. (mlr: 0.915, mrr: 0.546, servo1Angle: 0, servo2Angle: 0.0571)
Finish Evaluate. (penalty: 120)
azimuthDiffThreshold:30 encoderDiff:250 encoderThreshold:100

Finish Learning. Best is (mlr: 0.514, mrr: 0.819, servo1Angle: 0, servo2Angle: 0)-> GA終了
Learning Result Started!!(backSpin: false) 5[sec]
SERVO_CENTER: 0-> サーボ中央の最適位置

MOTOR_L_POWER_OPT_MAGNIFICATION: 0.514-> 左モータの最適出力
MOTOR_R_POWER_OPT_MAGNIFICATION: 0.819-> 右モータの最適出力
[1m[11:35:24] [36mSequence: [32m'navigating'-> ナビゲーション再開

[0mGPS history of stuck_detection_by_gps has been reset because of subsequence interruption!
Current Azimuth: 118-> サクセスクライテリア用の性能評価開始
Target Azimuth: 118
Max Azimuth Diff: 0
Ave Azimuth Diff: 0
(略)
Current Azimuth: 123
Target Azimuth: 118
Max Azimuth Diff: 6.21
Ave Azimuth Diff: 4.53
Finished Check Performance!
Max Azimuth Diff: 6.21-> 最大角度誤差
Time: 11:35:30
Target Azimuth: 265.079
Remaining Distance: 19.423 m (gps)


Time: 11:35:31
Target Azimuth: 265.079
Remaining Distance: 19.423 m (gps)
(略)

Time: 11:36:04
Target Azimuth: 270.011
Remaining Distance: 19.200 m (gps)


stuck detection value: 1.68
stuck detection threshold: 2.5
stuck detection: 1-> スタック判定
[1m[11:36:05] [36mSequence: [32m'navigating' > 'escaping'-> エスケーピング（轍脱出）開始
[0mRetrying escaping... (1/6)
Retrying escaping... (2/6)
Retrying escaping... (3/6)
```

```
Retrying escaping... (4/6)
Retrying escaping... (5/6)
Retrying escaping... (6/6)
[1m[11:36:28] [36mSequence: [32m'navigating'
[0mGPS history of stuck_detection_by_gps has been reset because of subsequence interruption!
[1m[11:36:28] [36mSequence: [32m'navigating' > 'waking'
[0mWaking finished!
[1m[11:36:45] [36mSequence: [32m'navigating'-> ナビゲーション再開
[0mGPS history of stuck_detection_by_gps has been reset because of subsequence interruption!
Time: 11:36:45
Target Azimuth: 268.514
Remaining Distance: 21.325 m (gps)
(略)

Time: 11:37:25
Target Azimuth: 272.734
Remaining Distance: 19.525 m (gps)


stuck detection value: 0.871
stuck detection threshold: 2.5
stuck detection: 1-> スタック判定
[1m[11:37:25] [36mSequence: [32m'navigating' > 'escaping'-> エスケーピング（轍脱出）開始
[0mRetrying escaping... (1/6)
Retrying escaping... (2/6)
Retrying escaping... (3/6)
Retrying escaping... (4/6)
Retrying escaping... (5/6)
Retrying escaping... (6/6)
[1m[11:37:48] [36mSequence: [32m'navigating'-> ナビゲーション再開
[0mGPS history of stuck_detection_by_gps has been reset because of subsequence interruption!
Time: 11:37:48
Target Azimuth: 270.603
Remaining Distance: 17.992 m (gps)
(略)

Time: 11:42:07
Target Azimuth: 259.323
Remaining Distance: 13.999 m (gps)


Time: 11:42:08 -> 競技時間15分経過し，ここでゴールまでの距離を計測（競技終了）
Target Azimuth: 258.455
Remaining Distance: 13.887 m (gps)

Time: 11:42:09
Target Azimuth: 257.707
Remaining Distance: 13.925 m (gps)
(略)
```

```
Time: 11:47:00-> 計測終了(競技終了)後，ゴール5m付近まで機体を直接移動
Target Azimuth: 206.624
Remaining Distance: 5.391 m (gps)


Time: 11:47:01
Target Azimuth: 208.506
Remaining Distance: 5.063 m (gps)


Close to Goal!

remaining distance: 4.98
[1m[11:47:02] [36mSequence: [32m'goal_detecting'-> ゴール検知開始
[0mGoal not found! (rate: 0)
Turning to right [deg] ...
Goal not found! (rate: 0)
Turning to right [deg] ...
(略)
Goal not found! (rate: 0)
Turning to right [deg] ...
Goal found! (rate: 0.0582)
Goal azimuth = 276 (current) + -6.1 (red area angle) = 270-> ゴールを検知し移動開始
Goal not found! (rate: 0)
Turning to right [deg] ...
(略)
Goal found! (rate: 0.0035)
Goal azimuth = 322 (current) + 5.56 (red area angle) = 328
Goal found! (rate: 0.0742)
Goal azimuth = 317 (current) + 6.28 (red area angle) = 323
Goal found! (rate: 0.286)
Goal!-> 赤色割合が25%を超えたためゴールに十分近づいたと判定->プログラムを終了
[1m[11:52:06] [36mSequence: [32m'testing'
```

＜受賞した賞＞

・Accuracy Award 3位

3. 考察

- 投下1回目

  CanSatが着地後に無事にパラシュートの切り離しに成功した．しかし，着地地点付近の草にスタックした．その後スタックを検知し脱出動作によってスタックした草から脱出し移動を始

めたが，以後もスタックと脱出を繰り返した．ミッション開始から約10分後に駆動系が動作しなくなり，制限時間である15分を迎え制御を終了した．最適化の処理（GA）はCanSatがスタックした（進行方向が直進に限定されている）状況下でのみ実行された．そのため，フルサクセス達成条件である最適化後の角度誤差9°未満に対して，実際に記録した角度誤差は4°となっているものの，「走行できる」という条件を満たしていない．

- 投下2回目

  まず，投下1回目にスタックが頻発しミッションを実施するような状況にならなかったことを踏まえ，今回は故障発生のタイミングを投下1回目の「ナビゲーション動作開始直後」から「ローバがゴール20m以内に接近したと判定した時」に変更している．CanSatが着地後に無事にパラシュートの切り離しに成功し，順調に（故障検知を定常的に発動しながら）走行を続けた．ゴール20m以内に到達した直後，右主輪を意図的に停止させ，最適化の処理（GA)を実行した．実行後，スタック脱出を繰り返しながらゴールに向かって走行を続けたが，ゴール10m地点で時間切れとなった．CanSatの最適化実行後の角度誤差は4°を記録しており，フルサクセス達成条件である「最適化後の進行方向における角度誤差が9°以下で走行できる」を満たすことができている．

第9章　　まとめ

1. 工夫・努力した点（ハード、ソフト、マネジメント面すべて）

(1) ハード・回路

- 故障対応のためにスタビライザの駆動を2軸にする必要があり，1軸サーボを二つ用いて縦軸と横軸にすることで技術導入コストおよびパーツの費用を削減した．また，二つのサーボ制御を一つの出力端子から行うように出力振り分け用の三つ枝コネクタを制作することで回路の修正コストを削減した．
- 後輪のスタビライザにモータを設置しているため機体全体の重心がスタビライザに寄ってしまい反転復帰が困難である問題(反転時に機体が回転しすぎて再び反転状態に戻るなど)があった．これに対して，反転復帰の回転方向を一方向に限定するための補助パーツをメジャーを用いて作成．反転復帰成功率が飛躍的に向上した．
- パラシュートと機体を接続するためのピンを昨年から90度回転した方向に接続するように変更．昨年以前で頻発していた着地時に機体が反転した際にピンが抜けきらない問題を解決した．
- 車軸とタイヤの接続部に草が絡んで走行性能が低下する問題に対し，接続部をタイヤ内部に埋め込むことにより解決した．(スパイクに絡むものとは別問題)

(2) ソフト

- ゴール検知に用いる色判定の閾値を効率的に設定するために，入力画像と範囲選択によって閾値を出力する補助ソフトを開発．ゴール検知の精度向上のための調整コストが減少した．
- シミュレーション環境を用意するにあたってハード班と連携して3Dモデルを作成．あらかじめ決定していたことにより実環境(実機)・シミュレーション環境の齟齬を産まないように開発を進めた．
- 制御プログラム(c++)と進化計算による最適化プログラム(python)をboost pythonで接続したことにより，他の進化計算手法の導入のためのコストを削減し再利用性を高めた．

2. 課題点

(1) 審査書

- スケジュール進行の遅さが目立った. E2Eの回数を増やすために前の実験を早めるということは勿論, 前半がコロナ禍の影響を受けていたが, 試験の優先順位を設け, 低いものはカットすることなども考慮するべきであった.

(2) 機体全般

- FMの仕様確定に時間がかかり, また機体改修の度の追加試験に時間を掛けすぎてしまった.
- ある一定の深さを超える草に対してはモータの力が不足している問題があった. それに対してのスタック脱出動作の準備が不足していた. 1日目の投下後に調査と実装をしたが前もって準備することで他の実験に余力を割けたと考える.

(3) 発表

- 事前発表会にて制御系における「最適化」と進化計算による「最適化」の区別が曖昧であったため, ミッションの要が伝わりにくいスライドとなってしまった.
- 大会報告において, 動画を埋め込んでいたが上手く再生できなかった. 発表練習では問題なく再生できていたこと, 動画サイズがもっとも軽くなるように圧縮して出力していたため, 慢心し確認を怠っていたので, gif化を含め別の形式を用いるように検討.

3. 今後の展望

(1) ミッション面

制御系の最適化は環境依存性が高く, 適応できない環境には対処できないという問題がある. それに対し進化計算の最適化を併用することで制御系の最適化が適用可能になるように補正できること確認できた. 今回は4輪ローバの制御走行に対する問題として扱ったが, 補正を用いる箇所など組み合わせはさまざま考えられ, 制御系の最適化の汎用性や進化計算の実用性を高めることに繋がる.

(2) 開発面

機体の仕様に関しては, パラシュートの切り離し機構の改良により反転時などでも切り離しが安定するようになった. また, 後輪へ重心が偏ったローバでも反転復帰が安定する予備パーツの作成に成功した. こうした基本動作が安定するとミッションへの集中力が飛躍的に向上するため来年以降の挑戦を効率的にすすめることが出来ると考える.