GameModule

ゲームモジュール マニュアル

版	v1.0
発行日	2026-10-22
原本	ゲームモジュールマニュアル.md

目次

1. ゲームモジュールとは ····· 4

全体の構成 ····· 4

ゲームモジュール一覧 ····· 5

2. はじめて動かす ····· 6

必要なもの ····· 6

手順 1: アプリ「ゲームモジュールサーバー」を入れる ····· 6

手順 2: ゲームモジュールをつなぐ ····· 7

手順 3: ゲームモジュールサーバーを立ち上げる ····· 7

手順 4: アプリでゲームモジュールを動かしてみる ····· 10

3. ATTRACTS を動かす ····· 11

4. ロボットやフィールドに組み込む ····· 11

GM コネクト ····· 11

接続の全体像(ロボット搭載時) ····· 13

ユーザ MCU との接続(トランシーバーモジュール) ····· 14

電源接続 ····· 14

5. カメラモジュール ····· 15

5.1 概要 ····· 15

5.2 機能仕様 ····· 15

5.3 外部インターフェース(gRPC) ····· 16

5.4 電気仕様・コネクタ ····· 16

5.5 外形・取付 ····· 17

5.6 注意事項 ····· 19

6. ヒットパネルモジュール ····· 19

6.1 概要 ····· 19

6.2 機能仕様 ····· 19

6.3 外部インターフェース(gRPC) ····· 20

6.4 電気仕様・コネクタ ····· 20

6.5 外形・取付 ····· 20

6.6 注意事項 ····· 21

7. マズルモジュール ····· 22

7.1 概要 ····· 22

7.2 機能仕様 ····· 22

7.3 外部インターフェース(gRPC) ····· 22
7.4 電気仕様・コネクタ ····· 23
7.5 外形・取付 ····· 23
7.6 注意事項 ····· 25

8. トランシーバーモジュール ····· 26
8.1 概要 ····· 26
8.2 ユーザ MCU 側 UART インターフェース ····· 26
8.3 外部インターフェース(gRPC) ····· 28
8.4 電気仕様・コネクタ ····· 28
8.5 外形・取付 ····· 29
8.6 注意事項 ····· 29

9. パワーマネジメントモジュール ····· 30
9.1 概要 ····· 30
9.2 機能仕様 ····· 30
9.3 外部インターフェース(gRPC) ····· 31
9.4 電気仕様・コネクタ ····· 31
9.5 外形・取付 ····· 32
9.6 注意事項 ····· 32

10. gRPC API リファレンス ····· 33
SDK と gRPC の関係 ····· 33
接続情報 ····· 33
サービス一覧 ····· 33
共通の使い方 ····· 35
SDK の使い方 ····· 35

11. SDK で自分の競技を作る ····· 36
サンプル競技「ターゲットストライク」を作る ····· 36

12. トラブルシューティング ····· 39

13. 用語集 ····· 40

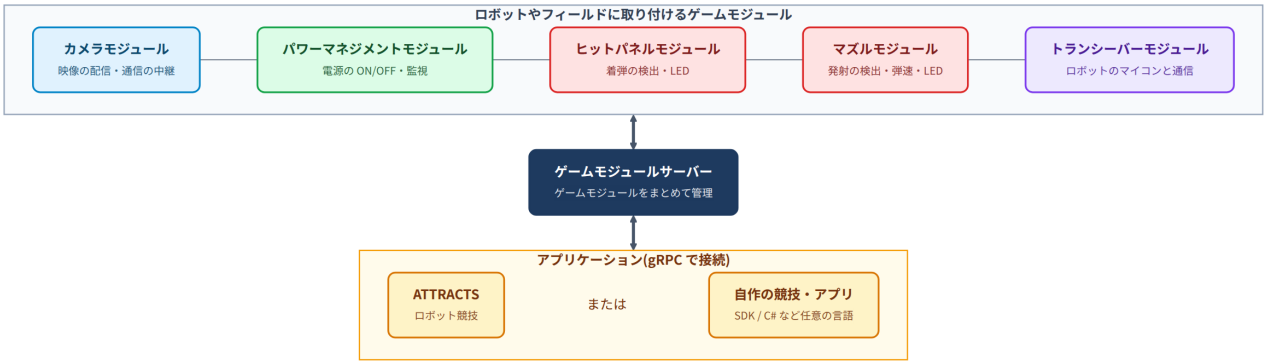
このマニュアルは、ゲームモジュールの全体像、初めて動かすまでの手順、ロボットやフィールドへの組み込み方、各ゲームモジュールの仕様、gRPC API、SDK を使った競技の作り方をまとめたものです。

1. ゲームモジュールとは

ゲームモジュールは、ロボット競技のための**汎用の電子審判・演出プラットフォーム**です。 弾の発射・着弾の検出、電源の遠隔制御といった**審判**の機能と、LED の発光やロボット視点の映像配信といった**演出**の機能を、ロボットやフィールドに取り付ける独立したゲームモジュールとして提供します。ゲームモジュールをロボットやフィールドに取り付けてゲームモジュールサーバーにつなぐと、競技のルールに合わせて判定と演出を自由に組み立てられます。

ゲームモジュールは特定の競技のためのものではありません。ロボット競技「ATTRACTS」はゲームモジュールを使う競技の一つです。**同じゲームモジュールで、自分の競技やアプリケーションも作れます**(11章参照)。

全体の構成



構成要素	役割
ゲームモジュール	ロボットやフィールドに取り付けるハードウェア。センサ・LED・電源制御などの機能を持つ。一覧は下表
ゲームモジュールサーバー	つながっているすべてのゲームモジュールを見つけて管理し、アプリケーションから使えるようにするソフトウェア。カメラモジュールの中で動くほか、Windows 用アプリ「ゲームモジュールサーバー」として Microsoft Store で配布しています
アプリケーション	ゲームモジュールを使って競技を動かすプログラム。ATTRACTS もその一つです

競技とゲームモジュールの役割分担

競技ごとのルール(何回当たったら負けか、当たったロボットを何秒止めるか など)はアプリケーションの側で決めます。ゲームモジュールは「着弾を検出する」「電源を切る」「LED を赤くする」など、ルールの中

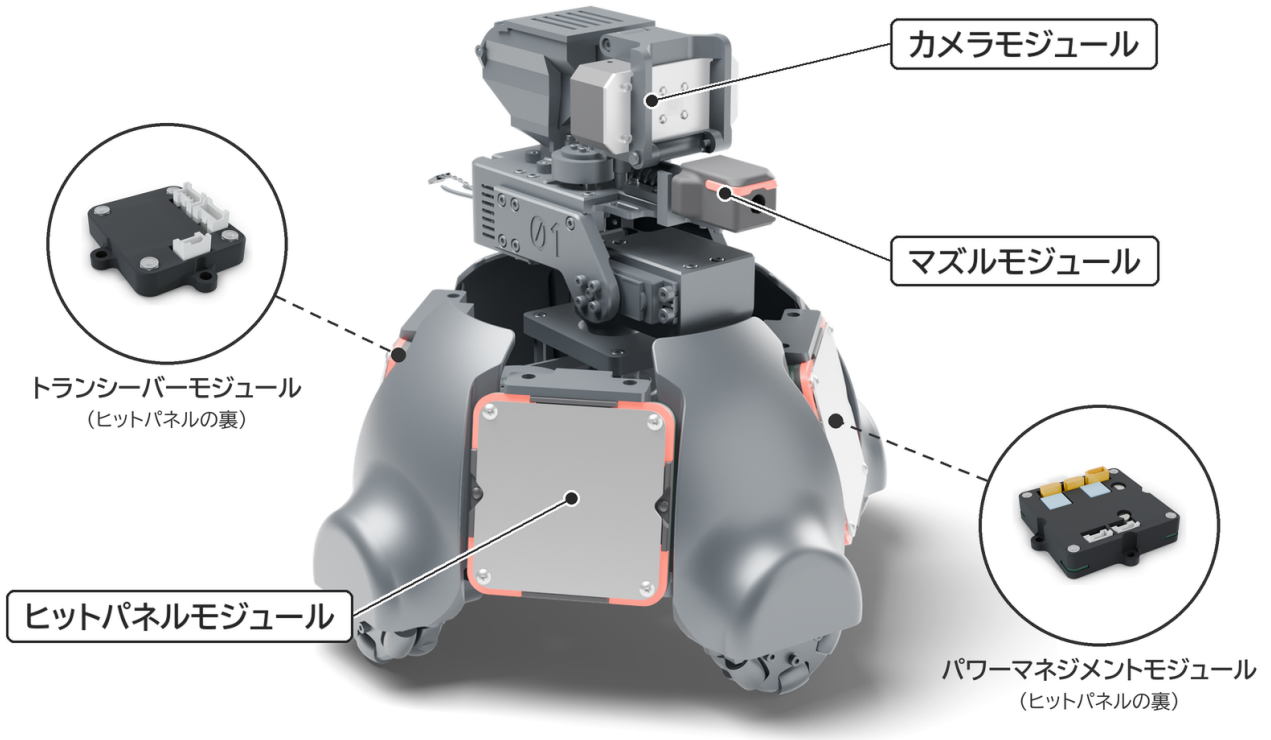
で現実に干渉する部分を受け持ちます。そのため、ルールを変えたり新しい競技を作ったりしても、ゲームモジュールはそのまま使えます。

アプリケーションは、ゲームモジュールサーバーに gRPC(通信の方式の一つ。ポート 50051)でつないで、すべての機能を使えます。公式の SDK を使うと Node.js/TypeScript から簡単に扱えます。SDK を使わず、公開されている proto 定義から C# など任意の言語で作ることもできます。

ゲームモジュール一覧

ゲームモジュール	機能	章
カメラモジュール	ロボット視点のカメラ映像を配信する。ほかのゲームモジュールとゲームモジュールサーバーの通信も中継する	5
ヒットパネルモジュール	BB 弾の着弾を検出する。フルカラー LED で状態を表示する	6
マズルモジュール	発射口に取り付け、弾の発射の検出と弾速の測定をする。フルカラー LED で状態を表示する	7
トランシーバーモジュール	ロボット搭載マイコン(Arduino 等)との通信を中継する	8
パワーマネジメントモジュール	ロボットの電源 ON/OFF 制御と電圧・電流の監視を行う	9

ロボットへの搭載イメージ:



2. はじめて動かす

この章では、ゲームモジュールを初めて使う人が **ゲームモジュールサーバーを立ち上げ、アプリ「ゲームモジュールサーバー」の画面でゲームモジュールを動かしてみる** ところまでを行います。ロボットやプログラミングは不要です。

必要なもの

品目	備考
ゲームモジュール一式	最低限: パワーマネジメントモジュール + カメラモジュール + 試したいゲームモジュール 1 つ
GM コネクトケーブル	2 本。4 章「GM コネクト」の仕様に合わせて作ったもの
バッテリー	パワーマネジメントモジュールにつなぐ 2〜6 セルの LiPo バッテリー
Windows PC	Windows 10/11。アプリ「ゲームモジュールサーバー」を入れます
Wi-Fi アクセスポイント	方法 B のみ。SSID とパスワードを設定できるもの
インターネット接続	アプリのインストール時のみ必要

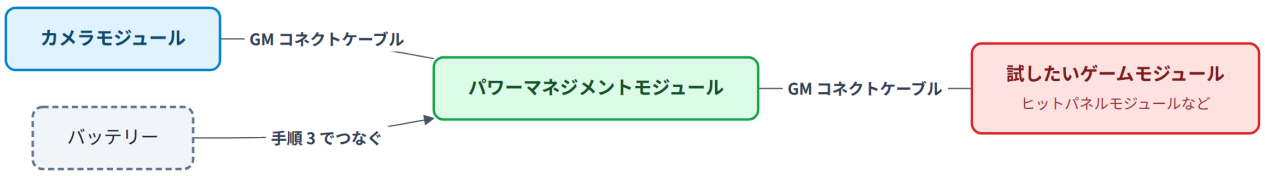
手順 1: アプリ「ゲームモジュールサーバー」を入れる

ゲームモジュールサーバーは、ゲームモジュールとアプリケーションの間をつなぐソフトウェアです。アプリ「ゲームモジュールサーバー」では、PC でゲームモジュールサーバーを動かせるほか、つながっているゲームモジュールを画面で確かめて操作できます。

Microsoft Store の「ゲームモジュールサーバー」からインストールします。



手順 2: ゲームモジュールをつなぐ



1. カメラモジュールとパワーマネジメントモジュールを GM コネクトケーブルでつなぎます
2. パワーマネジメントモジュールのもう一方の GM コネクト端子に、試したいゲームモジュールをつなぎます
3. バッテリーはまだつなぎません

GM コネクトの詳細は 4 章「GM コネクト」を参照してください。

手順 3: ゲームモジュールサーバーを立ち上げる

次のどちらかの方法で立ち上げます。

方法	内容
方法 A	カメラモジュールの中のゲームモジュールサーバーを使う
方法 B	Windows PC でゲームモジュールサーバーを動かす

どちらの方法でも、途中でパワーマネジメントモジュールにバッテリーをつないで電源を入れます。バッテリーのつなぎ方は 9 章「パワーマネジメントモジュール」を確認してください。

方法 A: カメラモジュールの中のゲームモジュールサーバーを使う

SSID が `ATTRACTS_WIFI` の Wi-Fi ネットワークが無いとき、カメラモジュールは自分で Wi-Fi のアクセスポイントになり、カメラモジュールの中のゲームモジュールサーバーが自動的に立ち上がります。

1. パワーマネジメントモジュールにバッテリーをつなぎます
2. しばらく待つと、SSID が `CameraModule-` で始まる 5GHz の Wi-Fi ネットワークが現れます
3. アプリを入れた PC をその Wi-Fi ネットワークに接続します。パスワードは `cameramodule1234` です
4. アプリ「ゲームモジュールサーバー」を起動します。画面上部の「見つかったサーバ」にカメラモジュールの中のゲームモジュールサーバーが表示され、自動で接続されます(表示されたボタンを押しても接続できます)

方法 B: Windows PC でゲームモジュールサーバーを動かす

1. Wi-Fi アクセスポイントを以下の設定にします。カメラモジュールはこの設定のネットワークに自動接続します(設定例は後述の「Wi-Fi ルーターの設定例」)

項目	設定値
周波数帯	5GHz(2.4GHz は無効にします)
SSID	`ATTRACTS_WIFI`
パスワード	`attracts1234`

2. Windows PC をこのアクセスポイントに接続します
3. アプリ「ゲームモジュールサーバー」を起動し、画面上部の「このPCのサーバ」の「起動」を押します



4. 「稼働中」と表示されれば準備完了です。アプリは立ち上げたゲームモジュールサーバーに自動で接続します



5. パワーマネジメントモジュールにバッテリーをつなぎます
6. カメラモジュールの LED が点灯するのを待ちます。しばらくすると、ゲームモジュールサーバーにゲームモジュールが接続されます

> IP アドレスの固定は不要です。カメラモジュールが mDNS(マルチキャスト DNS)でゲームモジュールサーバーを自動で見つけます。
>
> **接続されない場合:** Wi-Fi の SSID・パスワードと、5GHz が有効になっているかを確認します。直したあとはバッテリーをつなぎ直してください(つなぎ直さないと、カメラモジュールは 5 分おきにしか接続をやり直しません)。

Wi-Fi ルーターの設定例

ここでは TP-Link Archer C80 を例に設定手順を示します。SSID とパスワードを設定できる Wi-Fi ルーターであれば、ほかの機種でもかまいません。

1. ルーターの電源を入れ、PC をルーターの LAN ポートに有線で接続するか、ルーター底面のラベルにある初期の SSID とパスワードで Wi-Fi に接続します
2. ブラウザで `http://tplinkwifi.net` を開き、ログインします(初回は管理用パスワードを決める画面が出ます)
3. 画面上部の「ワイヤレス」を開き、2.4GHz の「有効」のチェックを外して 2.4GHz を無効にします。5GHz の「有効」にチェックを入れ、次のように入力します。カメラモジュールが 5GHz でつながるようにするためです

項目	設定値
ネットワーク名 (SSID)	`ATTRACTS_WIFI`
セキュリティ	WPA2-PSK[AES]
パスワード	`attracts1234`

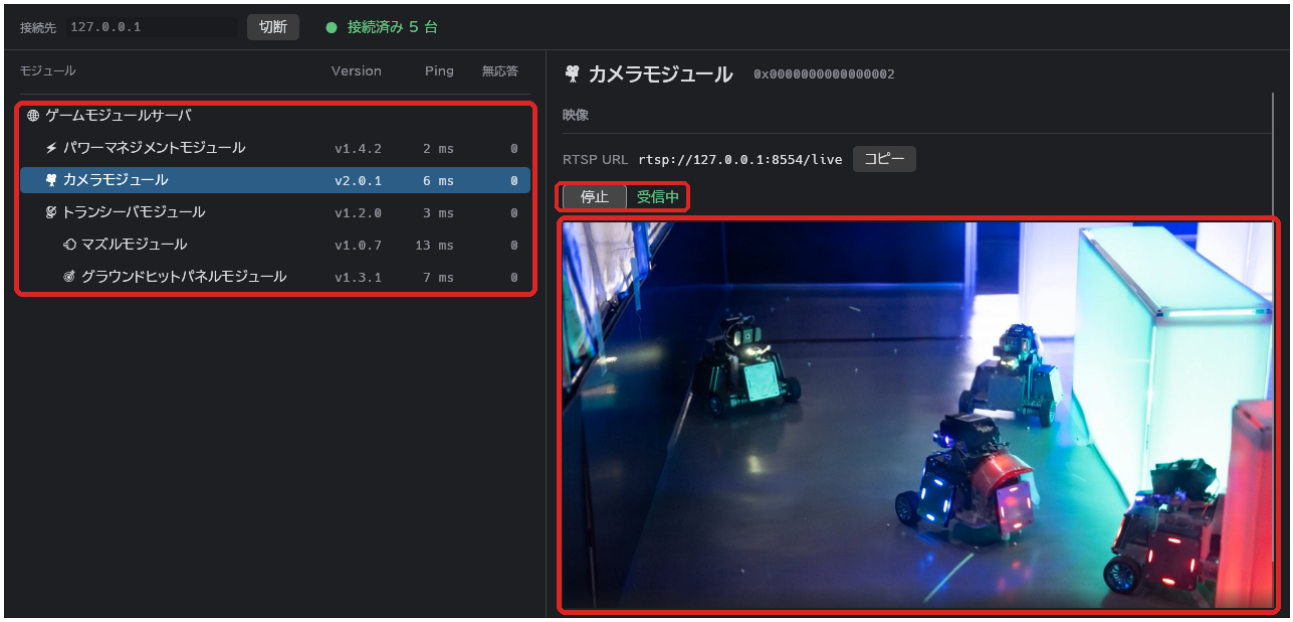


4. 右下の「保存する」を押します

手順 4: アプリでゲームモジュールを動かしてみる

ゲームモジュールサーバーに接続すると、つながっているゲームモジュールの一覧がアプリの左側に表示されます。一覧でゲームモジュールを選ぶと、右側にそのゲームモジュールの画面が表示されます。

たとえばカメラモジュールを選んで「再生」を押すと、カメラモジュールの映像が表示されます。



画面では、ゲームモジュールごとに次のことを試せます。

ゲームモジュール	画面でできること
カメラモジュール	映像の表示(「再生」)、映像の RTSP URL の確認
ヒットパネルモジュール(画面では「グラウンドヒットパネルモジュール」)	着弾の回数・時刻の表示、LED の色の変更
マズルモジュール	発射の回数・弾速の表示、UV LED の ON/OFF、LED の色の変更
パワーマネジメントモジュール	電圧・電流の表示、ドライブ電源の ON/OFF(画面では「給電」の「出力」)
トランシーバーモジュール(画面では「トランシーバモジュール」)	ユーザ MCU とのデータの送受信

3. ATTRACTS を動かす

ロボット競技「ATTRACTS」はゲームモジュールを使う競技の一つです。ATTRACTS を動かすときは、2 章の手順 3 で立ち上げたゲームモジュールサーバーに ATTRACTS のサーバーをつなぎ、ATTRACTS のサーバーがゲームモジュールサーバーを通してロボットを動かします。

1. Microsoft Store の「ATTRACTS アプリ」からインストールします
2. ATTRACTS アプリからゲームクライアントを起動し、接続画面の「ルームを立てる」タブを開きます
3. 「ゲームモジュールサーバー」の欄に、2 章の手順 3 で立ち上げたゲームモジュールサーバーが見つかったことを確かめます
4. 「ルームを立てる」を押します。ATTRACTS のサーバーが立ち上がり、ゲームを始められます(練習で使うときは「遠隔に公開する」を OFF にします)

遊び方と設定は ATTRACTS ドキュメントを参照してください。

4. ロボットやフィールドに組み込む

この章では、ゲームモジュールをロボットやフィールドに組み込むときの、ケーブルの作り方とつなぎ方を説明します。

> GM コネクトでやり取りする通信の中身は公開していません。この章では、ケーブル・コネクタ・電気的な仕様だけを説明します

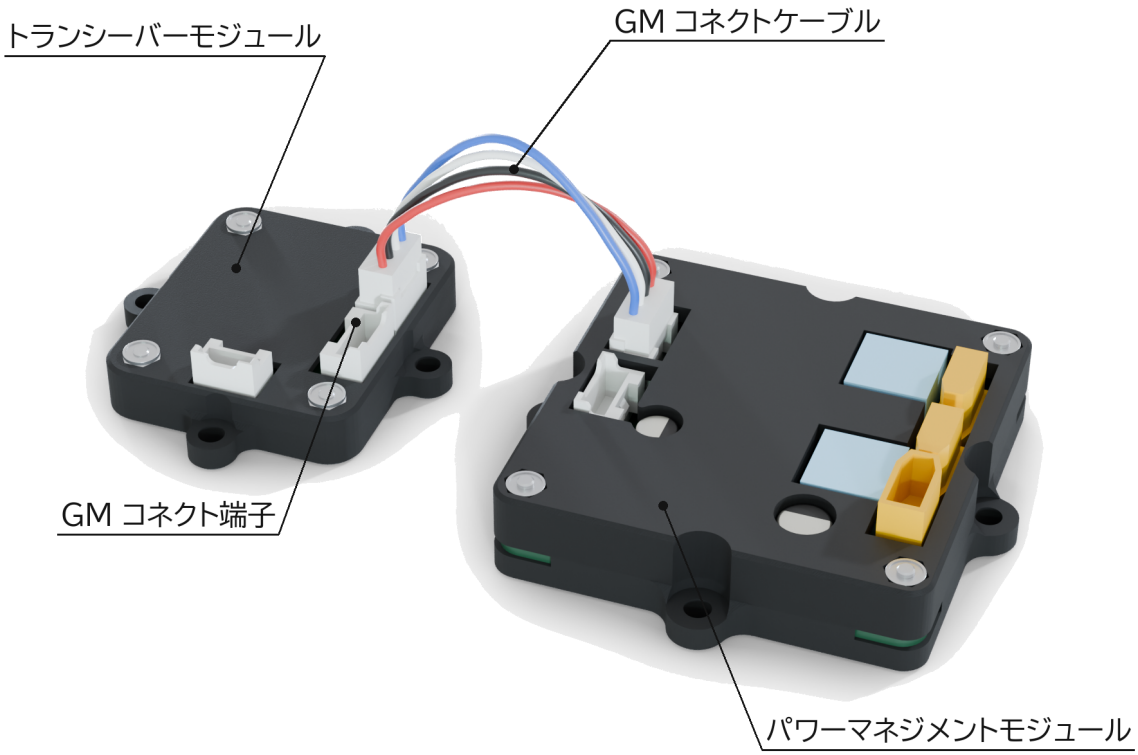
GM コネクト

GM コネクトは、ゲームモジュールどうしをつなぐ接続方式です。ゲームモジュールに付いている PA コネクタ 4 ピンの端子(**GM コネクト端子**)どうしを、**GM コネクトケーブル**でつなぎます。電源と通信は、このケーブル 1 本で送ります。

GM コネクトの仕様

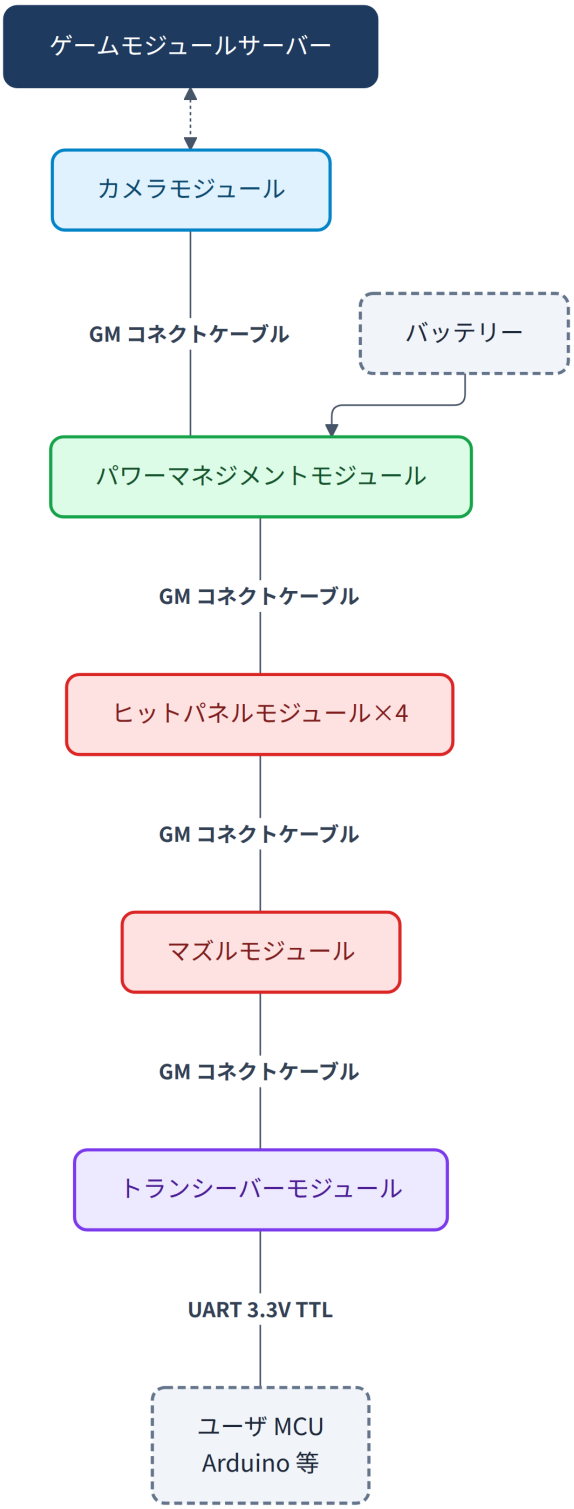
GM コネクト端子の数は、各ゲームモジュールの章の「電気仕様・コネクタ」を参照してください。

項目	仕様
端子	PA コネクタ 4 ピン(2.0mm ピッチ)。全ゲームモジュール共通
ケーブル	GM コネクトケーブル(4 芯)。両端で同じ番号のピンどうしをつなぐ
ピン配置	1: 電源(最大 12V)、2: 通信線 B、3: GND、4: 通信線 A
推奨線材	AWG22
電源を供給するゲームモジュール	パワーマネジメントモジュール



- GM コネクトケーブルは、上の仕様に合わせて作ったものをお使いください。PA コネクタ 4 ピンのケーブルでも、両端のピンのつなぎ方が違うものは使えません。大会には予備のケーブルを持っていくことをおすすめします

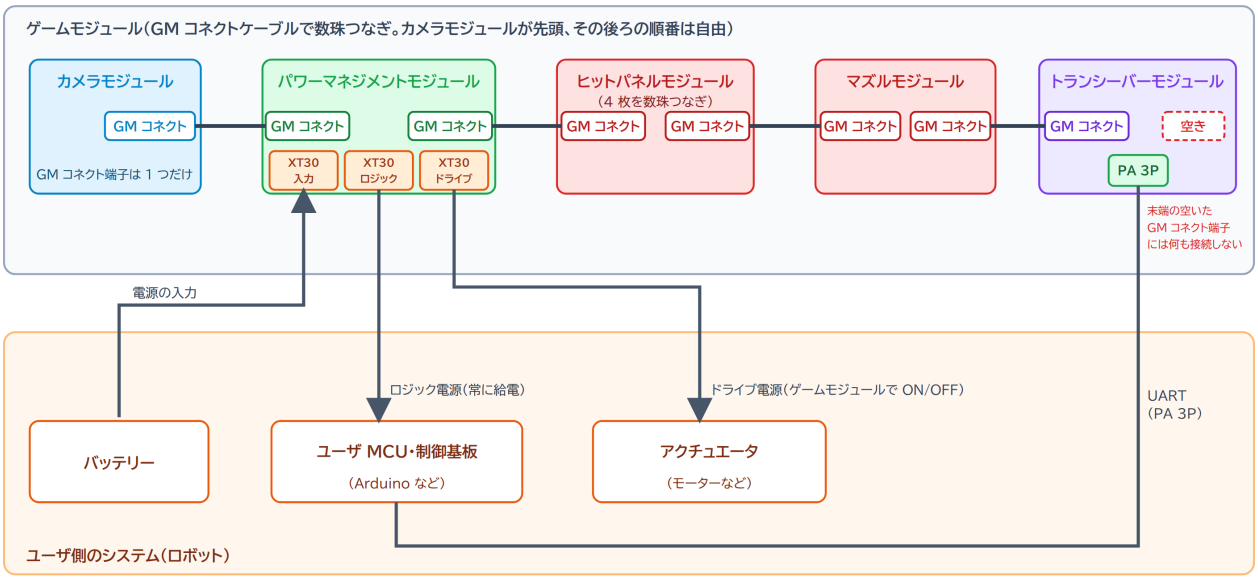
接続の全体像(ロボット搭載時)



- **パワーマネジメントモジュール** がバッテリーからの電源を受け、各ゲームモジュールへ電源を分配します

- **カメラモジュール** がゲームモジュールサーバーと通信します。ゲームモジュールサーバーとロボット上のほかのゲームモジュールの通信は、すべてカメラモジュールを経由します(ロボット 1 台にカメラモジュールが 1 台必要)
- ユーザ MCU とは、**トランシーバーモジュール** のユーザ MCU 接続コネクタでつなぎます

つなぎ方(数珠つなぎ)



- ゲームモジュールは**数珠つなぎ**にします。**つなぐ順番に制限はありません**が、**カメラモジュールは必ず端に置きます**(GM コネクト端子が 1 つだけのため)
- 反対側の端のゲームモジュールには、空いた GM コネクト端子が 1 つ残ります。**この端子には何もつながないでください**

ユーザ MCU との接続(トランシーバーモジュール)

コネクタ・ピン配置・電気仕様は 8 章「トランシーバーモジュール」を参照してください。接続時の注意:

- UART は**クロス接続**です(トランシーバーモジュールの TX → MCU の RX、RX → TX、GND 共通)
- 通信のレベルは 3.3V です。5V 系マイコン(Arduino UNO 等)を使う場合はレベル変換回路を挟んでください
- 3 ピン PA ケーブルで接続します
- ATTRACTS で流れるコマンドの内容は ATTRACTS ドキュメントの「AT-4 ロボット制御仕様書」を参照してください

電源接続

極性・対応電圧・最大電流は 9 章「パワーマネジメントモジュール」を参照してください。接続時の注意:

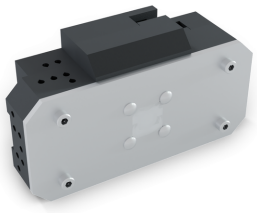
- **パワーマネジメントモジュールは、GM コネクトでつながった中に 1 台だけにしてください。2 台以上つながないでください**

- **逆接続は破損の原因になります。**接続前に必ず極性を確認してください

5. カメラモジュール

5.1 概要

カメラモジュールは、ロボット視点のカメラ映像を RTSP(標準的な映像ストリーミングプロトコル)で配信するゲームモジュールです。 また、ロボットに載せたほかのゲームモジュールとゲームモジュールサーバーの通信を**中継**します。ロボットにゲームモジュールを載せるときは、カメラモジュールが必ず 1 台必要です。



5.2 機能仕様

5.2.1 映像配信(RTSP)

項目	値
プロトコル	RTSP
ポート	8554(固定)
パス	`/live`
解像度	1280×720(720p)
フレームレート	30fps
エンコード	H.264 (High Profile, Level 5.1)
ビットレート	1Mbps(固定)

映像の配信は起動時に自動で始まります。映像の URL はステータス(`rtsp_url`)で通知されるため、受信側で設定する必要はありません。

受信例:

```
# VLC
vlc rtsp://<カメラモジュールのIP>:8554/live

# GStreamer
gst-launch-1.0 rtspsrc location=rtsp://<カメラモジュールのIP>:8554/live latency=100 \
  ! rtph264depay ! h264parse ! avdec_h264 ! videoconvert ! autovideosink
```

5.2.2 ネットワーク接続

- 起動すると、Wi-Fi ネットワーク(SSID `ATTRACTS_WIFI` / パスワード `attracts1234`)に自動で接続します。2.4GHz と 5GHz のどちらにも接続できます
- 同じネットワークにあるゲームモジュールサーバーを mDNS で見つけて接続します。IP アドレスを固定する必要はありません
- 起動後およそ 20 秒たっても `ATTRACTS_WIFI` に接続できないとき、または接続中に切断が 60 秒続いたときは、カメラモジュールが自分で Wi-Fi のアクセスポイントになり、カメラモジュールの中のゲームモジュールサーバーが立ち上がります
- アクセスポイントの SSID は `CameraModule-` のあとに HardwareNumber(16 桁)が続く名前、パスワードは `cameramodule1234` です
- このアクセスポイントは、2.4GHz ではなく **5GHz**(チャネル 36)で立ち上がります
- アクセスポイントになっている間も `ATTRACTS_WIFI` を探し続け、見つかると接続に切り替えます。ただし、アクセスポイントに PC などが接続している間は探しません

5.3 外部インターフェース(gRPC)

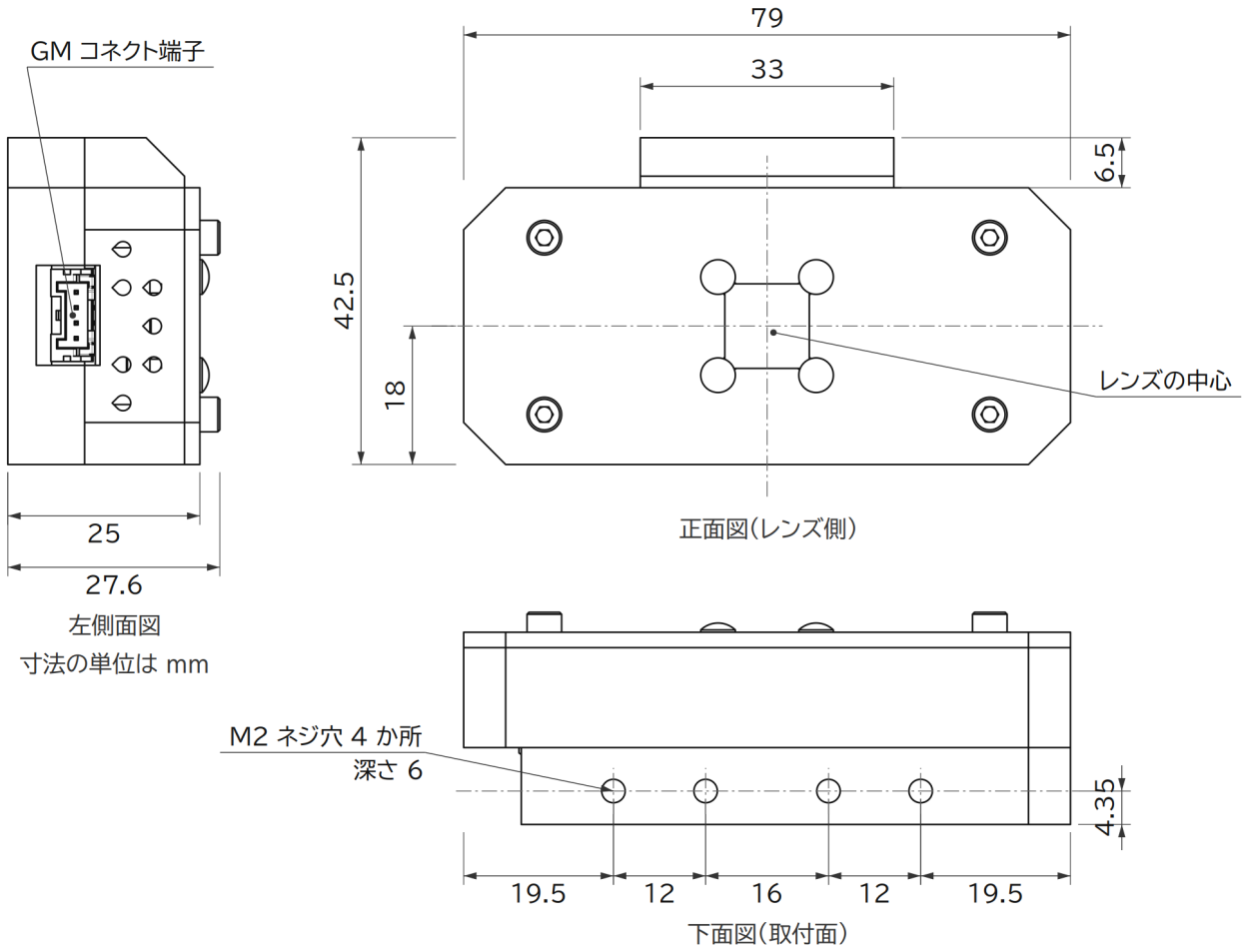
ゲームモジュールサーバー経由で以下の操作ができます。詳細は 10 章「gRPC API リファレンス」を参照してください。

RPC	内容
`StreamCameraStatus`	映像の RTSP URL を取得

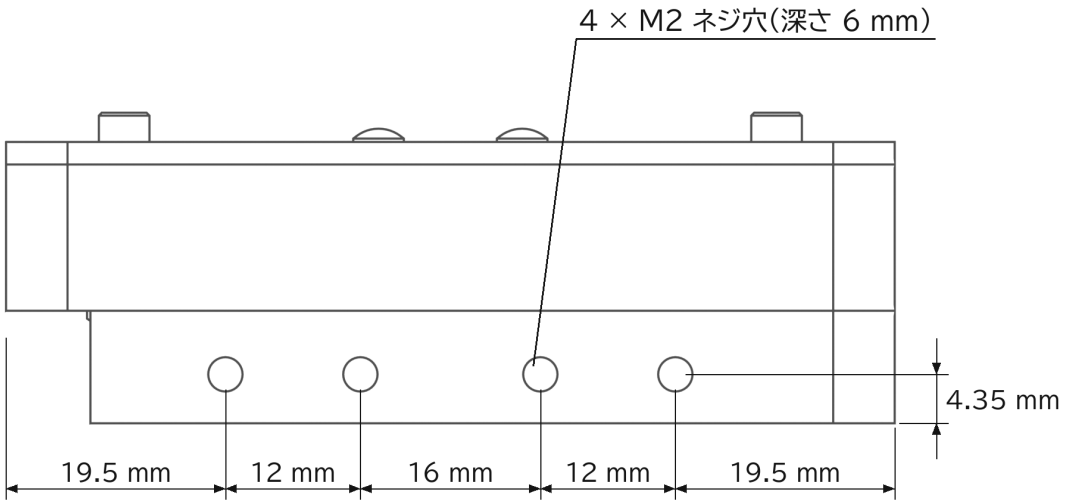
5.4 電気仕様・コネクタ

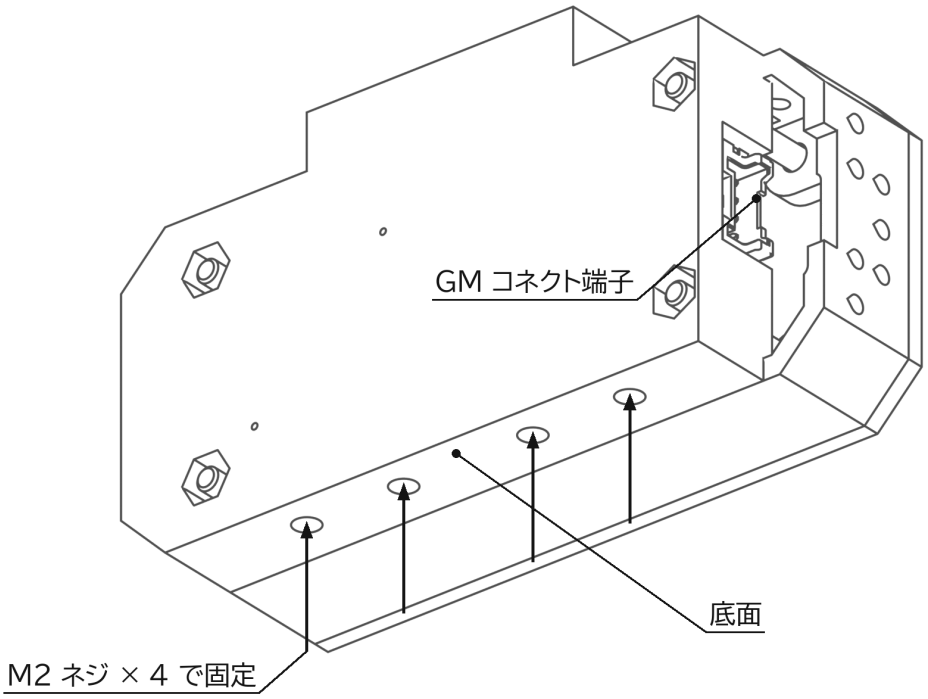
項目	仕様
GM コネクト端子	PA コネクタ 4 ピン × 1
電源	GM コネクトから供給(接続方法は 4 章)

5.5 外形・取付


GM コネクト端子
25
27.6
左側面図
寸法の単位は mm
79
33
6.5
42.5
18
レンズの中心
正面図(レンズ側)
M2 ネジ穴 4 か所
深さ 6
4.35
19.5
12
16
12
19.5
下面図(取付面)

底面





項目	値
外形寸法	79mm × 42.5mm × 25mm(正面のネジ頭を含めると奥行き 27.6mm)
取付ネジ穴	底面 4 × M2 ネジ穴、深さ 6mm

• カメラの視野を遮らないように取り付けてください

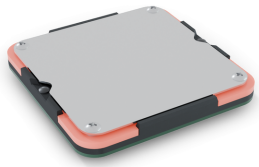
5.6 注意事項

- **金属部品でゲームモジュールを覆うと、電波が遮られて無線がつながりにくくなります。**周りを金属で囲まないように取り付けてください
- 排熱のための吸気口・排気口があります。**それぞれ 30% 以上ふさがないでください**
- `ATTRACTS_WIFI` が見えているのに接続できなかったときは、アクセスポイントになって待ち、5 分おきに接続をやり直します。接続できない場合は、Wi-Fi の SSID とパスワードを確認してください
- 映像の遅延は Wi-Fi の状態で変わります。安定した Wi-Fi 環境で使用してください
- SSID が `ATTRACTS_WIFI` の Wi-Fi ネットワークが複数あると、意図しない接続先に接続することがあります
- **カメラモジュールは屋内で使用してください。** 5GHz の無線を使うためです

6. ヒットパネルモジュール

6.1 概要

ヒットパネルモジュールは、BB 弾の着弾を検出するパネル型ゲームモジュールです。着弾はイベントとして gRPC 経由で通知されます。フルカラー LED を搭載しており、チームカラーや状態の表示に使用できます。



ロボットの被弾判定(ATTRACTS では 1 機あたり 4 枚搭載)のほか、フィールドに設置して「的」として使うこともできます(11 章参照)。

6.2 機能仕様

6.2.1 着弾検出

項目	仕様
検出対象	BB 弾
検出レート	1 秒間に最大 10 回の着弾を検出できる
通知	着弾ごとに 1 イベントを gRPC でストリーミング通知

6.2.2 LED 表示

項目	仕様
LED	フルカラー RGB(パネル面の端の 10 個の LED が発光)
制御	gRPC で RGB 各 0–255 を指定

6.3 外部インターフェース(gRPC)

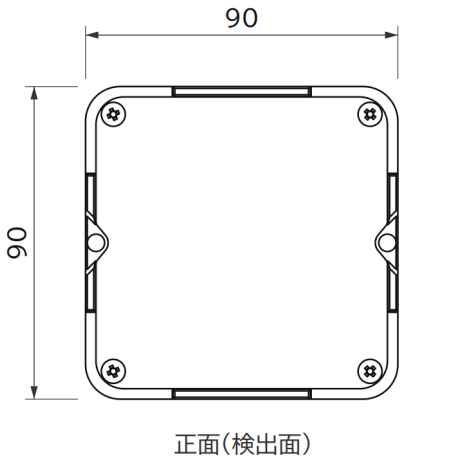
詳細は 10 章「gRPC API リファレンス」を参照してください。

RPC	内容
`StreamHitEvents`	着弾を検出するたびに、通知を受け取る
`SetLEDConfig`	LED 色の設定(RGB 各 0–255)

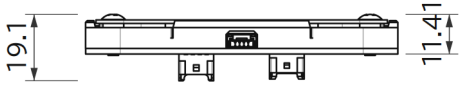
6.4 電気仕様・コネクタ

項目	仕様
GM コネクト端子	PA コネクタ 4 ピン × 2
電源	GM コネクトから供給(接続方法は 4 章)

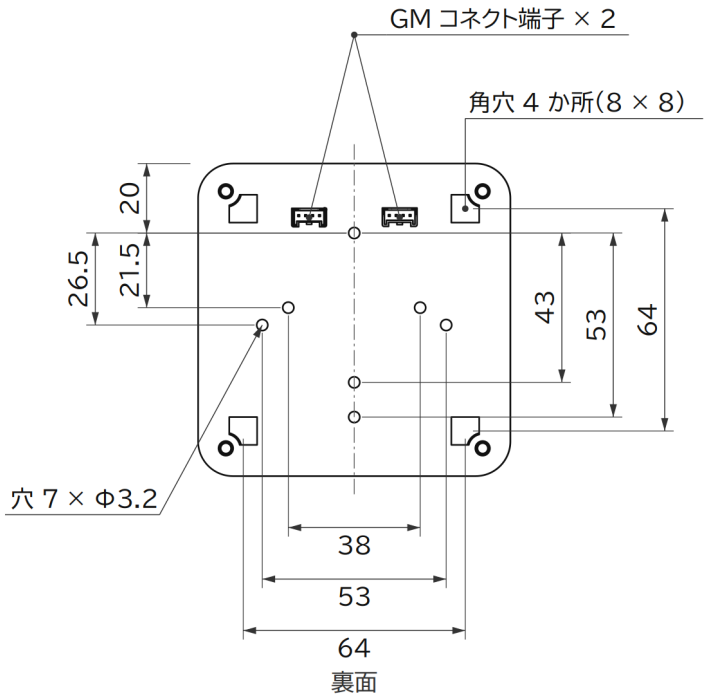
6.5 外形・取付



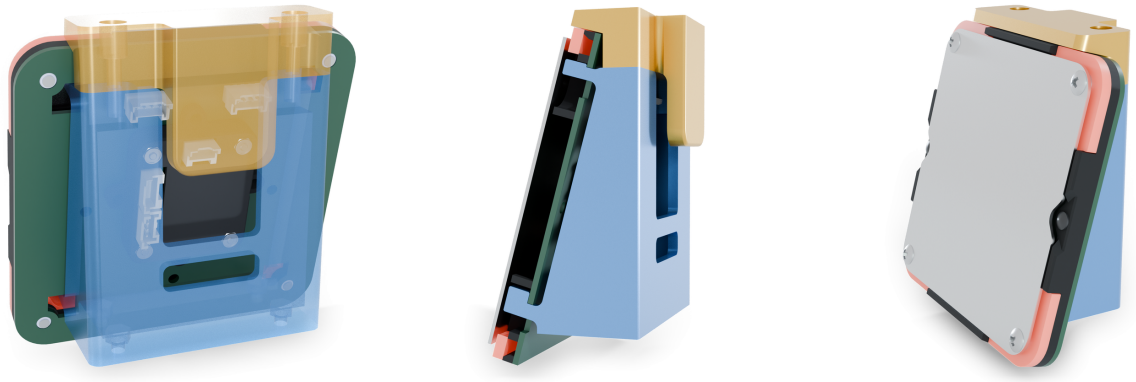
正面(検出面)



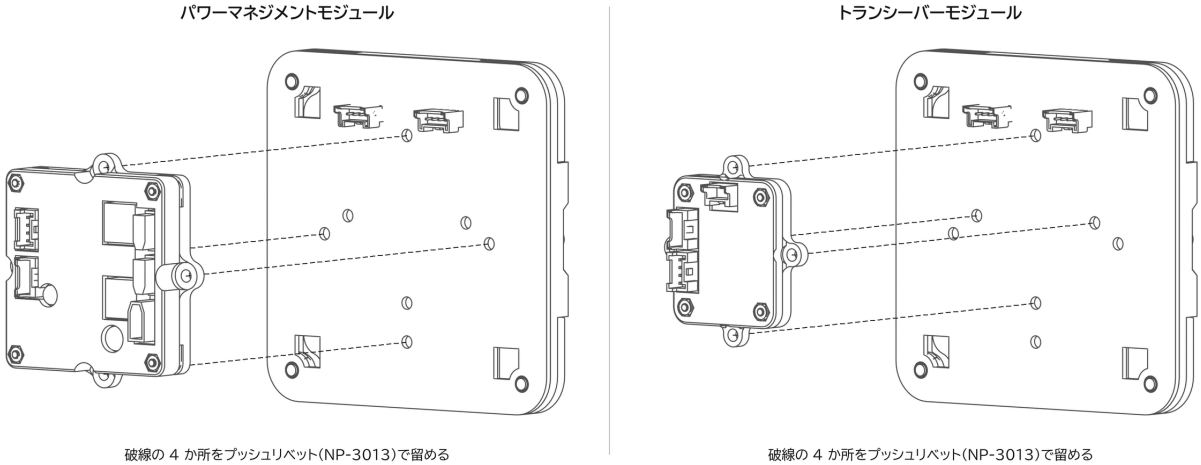
側面図(正面図の下側から見る)



寸法の単位は mm



ロボットへの取り付けイメージ。左から、裏から見た図(受け部品を半透明で表示)、爪の位置で切った断面、表から見た図です。



項目	値
外形寸法	90mm × 90mm × 19.1mm(裏面のコネクタを含む)
取付	四隅の角穴 × 4 に、ロボット側の受け部品の爪を掛けて固定

- 受け部品の形状は、ロボスポ デザインで公開している「ATTRACTSモデルロボット」の CAD データを参照してください。ATTRACTSモデルロボットのページからダウンロードできます
- 受け部品は 1 つでも、上下 2 つに分けてもかまいません
- パネルの検出面に物を触れさせないでください

6.6 注意事項

- **静電気で誤検出することがあります**
- パネルへの打撃(BB 弾以外の物体による衝撃)も着弾として検出される場合があります。競技での扱いは各競技のルールに従ってください
- ATTRACTS における搭載位置・角度の規定は ATTRACTS ドキュメントの「AT-2 ビルドマニュアル」を参照してください

7. マズルモジュール

7.1 概要

マズルモジュールは、発射機構の発射口に取り付けるゲームモジュールです。弾の発射を検出して**弾速を測定**し、イベントとして gRPC 経由で通知します。フルカラー LED を 3 組備えており、状態表示に使用できます。また、紫外線を照射して蓄光式の BB 弾を発光させることができます。



7.2 機能仕様

7.2.1 発射検出・弾速測定

項目	値
測定値	弾速(0.1 m/s 単位、16bit)
通知	発射ごとに 1 イベント(弾速付き)を gRPC でストリーミング通知

7.2.2 LED 表示

項目	仕様
LED	フルカラー RGB 左右ペア × 3(ペアごとに個別制御)
制御	gRPC で 3 つの発光部の RGB 各 0–255 を同時に指定(3 つとも指定必須)
UV LED	有効/無効を gRPC で切り替えられる(蓄光弾を使うとき)。有効にすると、弾を検出したときに自動で照射します

7.3 外部インターフェース(gRPC)

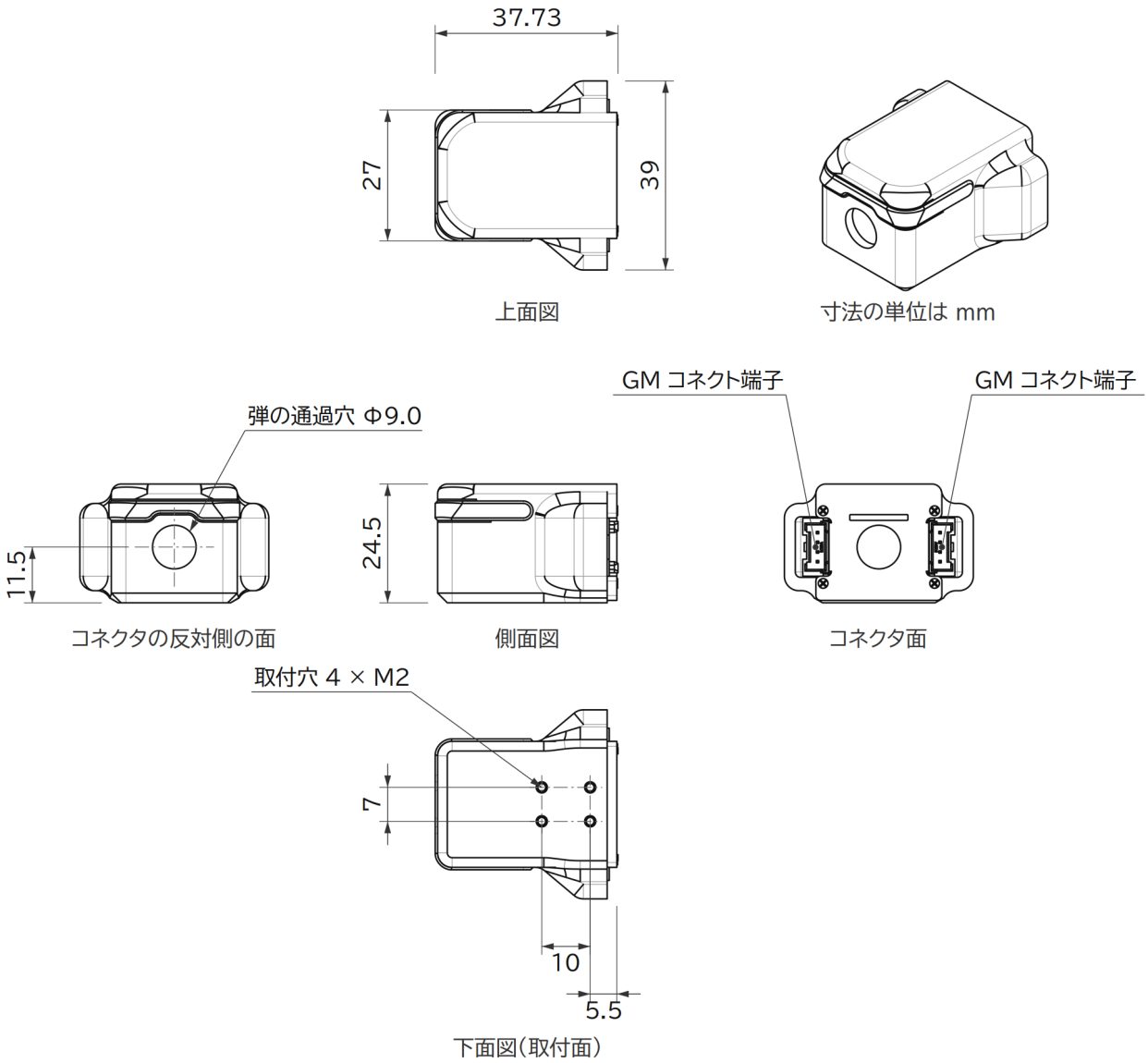
詳細は 10 章「gRPC API リファレンス」を参照してください。

RPC	内容
`StreamShotEvents`	発射を検出するたびに、弾速を受け取る
`SetLEDConfig`	3 つの発光部の色設定
`SetUVLEDEnable`	UV LED の有効/無効切替

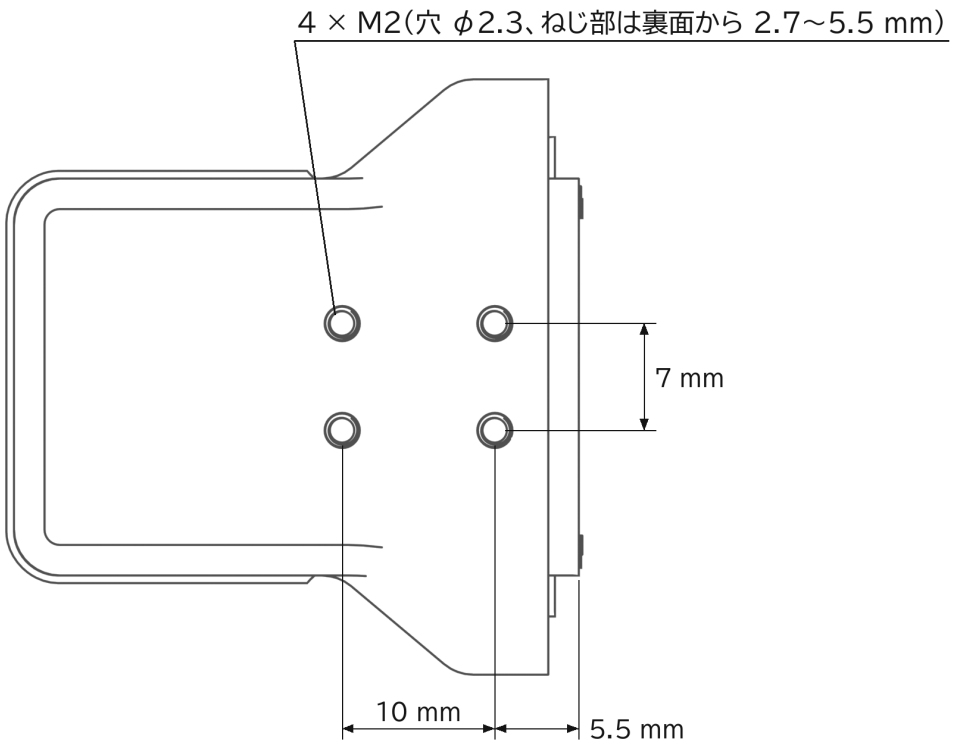
7.4 電気仕様・コネクタ

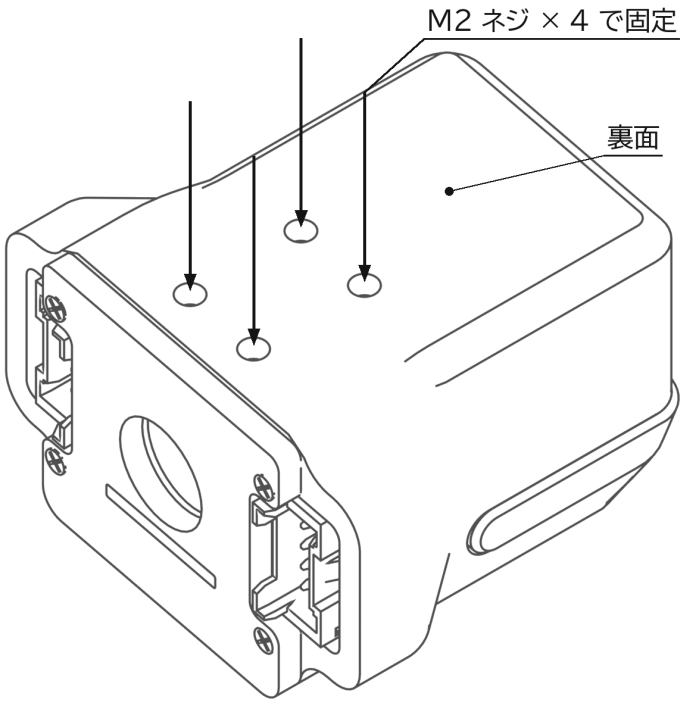
項目	仕様
GM コネクト端子	PA コネクタ 4 ピン × 2
電源	GM コネクトから供給(接続方法は 4 章)

7.5 外形・取付



裏面

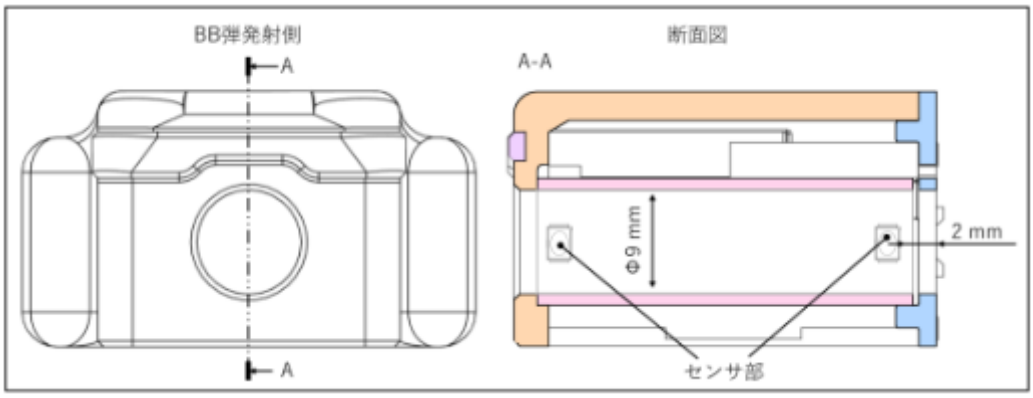


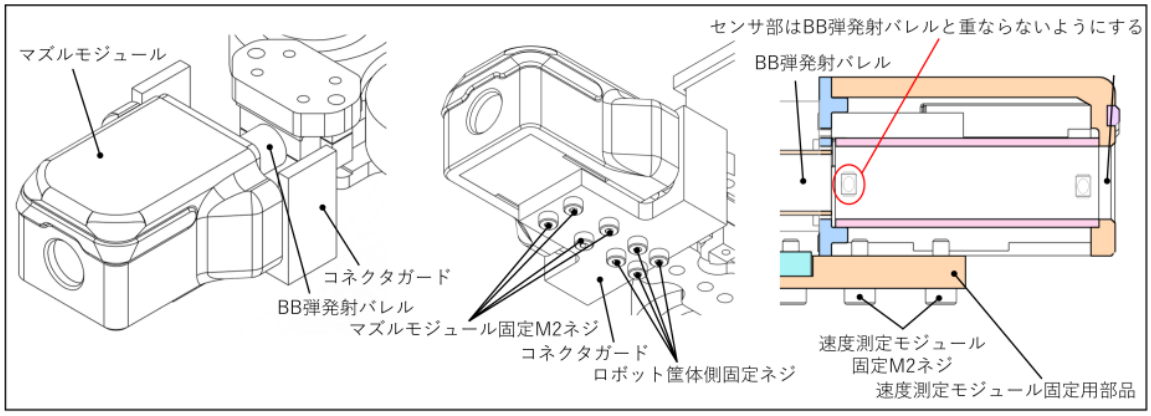


項目	値
外形寸法	長さ 37.73mm × 幅 27mm(後ろの張り出し部 39mm)× 高さ 24.5mm
弾の通過穴	Φ9mm
取付ネジ穴	下面 4 × M2、深さ 3mm(ピッチ 10mm × 7mm)

- 弾が検出ゲートの中心を通過するように、発射機構と同軸に取り付けてください
- 検出ゲート内部への異物・汚れの付着は検出漏れの原因になります

7.6 注意事項

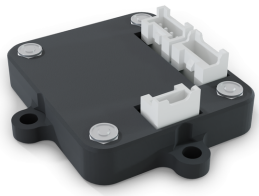




- **センサ部とバレル(発射機構の筒)が重ならないように取り付けてください**。重なるとエラーになります
- BB 弾が、コネクタのある側から LED のある側へ抜ける向きに取り付けてください。逆向きでは正しく動作しません
- **動作中は、バレルの中をのぞかないでください。UV LED の紫外線は強く、目を痛めるおそれがあります**
- 高い弾速で発射する場合は、取り扱いに十分注意してください
- ATTRACTS における搭載ルール(マズルモジュールを介さない射出の禁止等)はATTRACTS ドキュメントの「AT-2 ビルドマニュアル」を参照してください

8. トランシーバーモジュール

8.1 概要



トランシーバーモジュールは、**ユーザが用意したロボット搭載マイコン(Arduino 等。以下「ユーザ MCU」)とゲームモジュールサーバーの間でデータをやり取りする**ゲームモジュールです。 gRPC で送信されたバイナリデータをユーザ MCU へ UART で届け、逆にユーザ MCU から UART で受信したデータを gRPC 側へ届けます。データの中身は変えずにそのまま送るため、サイズの上限以内なら任意のデータを送受信できます。

8.2 ユーザ MCU 側 UART インターフェース

8.2.1 通信仕様

項目	値
通信方式	UART
通信速度	115200 bps
データビット	8 bit
パリティ	なし
ストップビット	1 bit
フロー制御	なし
信号レベル	3.3V。**5V の信号を入力しないでください**(ゲームモジュールが壊れるおそれがあります)

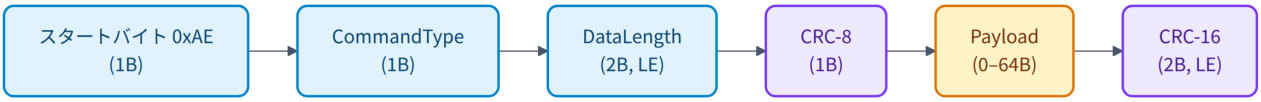
> 5V 系マイコンを接続する場合はレベル変換が必要です。コネクタとピン配置は 8.4 節にあります(つなぎ方は 4 章)。

8.2.2 データ転送

項目	値
1 回の送信データ長(gRPC→MCU / MCU→gRPC)	最大 64 バイト
データ形式	任意のバイナリ(そのまま転送)

8.2.3 フレーミングプロトコル(オプション)

誤り検出の付いたフレームで送りたいときは、gRPC の `SendFramedData` を使います。トランシーバーモジュールが次の形式のフレームを作り、先頭から順にユーザ MCU へ送ります。



フィールド	サイズ	内容
スタートバイト	1	`0xAE` 固定
CommandType	1	用途を表す値(0x00–0xFF。意味はアプリケーションで決める)
DataLength	2	Payload の長さ(0–64、リトルエンディアン)
CRC-8	1	ヘッダ 4 バイト(スタートバイト〜DataLength)のチェックサム
Payload	可変	データ本体
CRC-16	2	CRC-16 自身を除く全バイトのチェックサム(リトルエンディアン)

CRC-8(ヘッダ)

- 多項式: `0x07`、初期値: `0x00`、計算対象: スタートバイト〜DataLength の 4 バイト

```
uint8_t CalculateCRC8(const uint8_t* data, size_t length) {
    uint8_t crc = 0;
    for (size_t i = 0; i < length; i++) {
        crc ^= data[i];
        for (uint8_t j = 0; j < 8; j++)
            crc = (crc & 0x80) ? (crc << 1) ^ 0x07 : (crc << 1);
    }
    return crc;
}
```

CRC-16(全体)

- 多項式: `0x1021`、初期値: `0xFFFF`、計算対象: CRC-16 を除く全バイト

```
uint16_t CalculateCRC16(const uint8_t* data, size_t length) {
    uint16_t crc = 0xFFFF;
    for (size_t i = 0; i < length; i++) {
        crc ^= (uint16_t)data[i] << 8;
        for (uint8_t j = 0; j < 8; j++)
            crc = (crc & 0x8000) ? (crc << 1) ^ 0x1021 : (crc << 1);
    }
    return crc;
}
```

受信側の処理

1. フレームは 7〜71 バイトの可変長。`0xAE` をスタートバイトとして探索する

2. CRC-8 と CRC-16 の両方が一致した場合のみフレームを処理する
3. 不一致の場合はバッファを 1 バイト進めて再同期する

Arduino で受信するプログラムの例は、ATTRACTS ドキュメントの「AT-4 ロボット制御仕様書」の付録にあります(フレームを処理する部分は、ほかの競技でも使えます)。

8.3 外部インターフェース(gRPC)

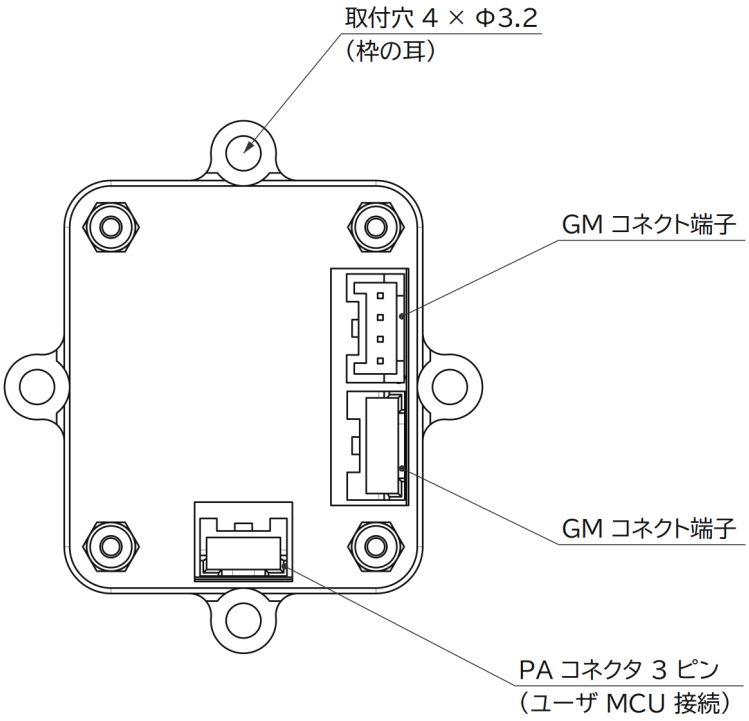
詳細は 10 章「gRPC API リファレンス」を参照してください。

RPC	内容
`SendUARTData`	バイナリデータをユーザ MCU へ送信(最大 64 バイト)
`StreamUARTData`	ユーザ MCU から届いたデータを、届くたびに受け取る
`SendFramedData`	フレーミングプロトコル(8.2.3 節)でペイロードを送信

8.4 電気仕様・コネクタ

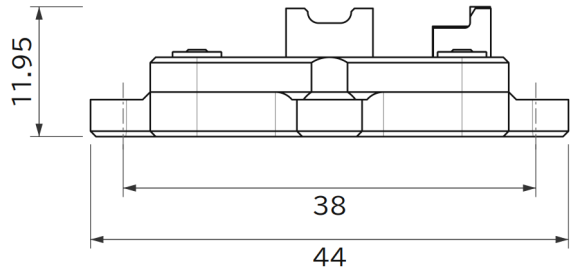
項目	仕様
GM コネクト端子	PA コネクタ 4 ピン × 2
ユーザ MCU 接続コネクタ	PA コネクタ 3 ピン × 1(**1: GND、2: ゲームモジュールの TX、3: ゲームモジュールの RX**)。ピン番号はコネクタのデータシートの番号です
電源	GM コネクトから供給(接続方法は 4 章)

8.5 外形・取付

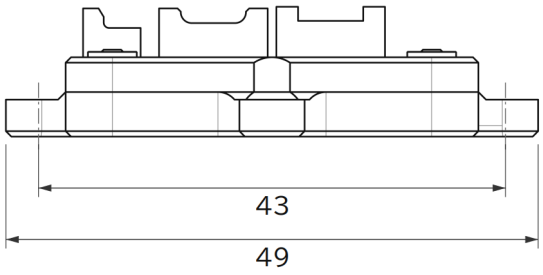


上面図

寸法の単位は mm



正面図



右側面図

項目	値
外形寸法	44mm × 49mm × 11.95mm
取付穴	4 × Φ3.2mm(向かい合う穴の間隔 38mm と 43mm)

- ヒットパネルモジュールの背面にプッシュリベット(NP-3013)で取り付けられます
- プッシュリベットは簡単に取れないので、取り外すにはプッシュリベット取り外し用の器具等を用いてください

8.6 注意事項

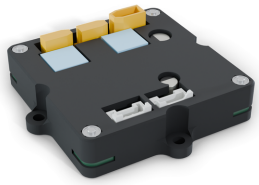
- UART の TX/RX 交差(クロス接続)に注意してください

- 通信が途切れたときの安全のための動作(入力のリセットなど)は、ユーザ MCU 側で作ってください(実装例は ATTRACTS ドキュメントの「AT-4 ロボット制御仕様書」)

9. パワーマネジメントモジュール

9.1 概要

パワーマネジメントモジュールは、ロボットの電源系統を管理するゲームモジュールです。gRPC からの指示で**ドライブ電源の ON/OFF** を切り替えられるほか、**バッテリーの電圧とドライブ電源に流れる電流を監視**できます。 ほかのゲームモジュールへの電源も、このゲームモジュールから送ります。 競技では、負けたロボットを強制的に止めるといった電子審判の役割をします。



9.2 機能仕様

9.2.1 電源制御

項目	仕様
制御対象	ドライブ電源
制御	gRPC で ON/OFF を指定

9.2.2 電源監視

項目	仕様
監視項目	バッテリーの入力電圧 [V]、ドライブ電源に流れる電流 [A]
備考	ロジック電源に流れる電流は測りません
測定範囲	電流 ±30A
通知	ステータスとして gRPC でストリーミング配信

9.2.3 保護機能

保護の対象はドライブ電源だけです。ロジック電源には、ON/OFF も過電圧・低電圧・大電流での遮断もありません。

- 入力電圧が約 29V を超えると、ドライブ電源を自動で OFF にします(部品のばらつきにより 27.4V から働く場合があります)
- 入力電圧が約 7.5V 未満になると、ドライブ電源が OFF になる場合があります

- ドライブ電源に 10A 以上流すと、遮断される場合があります。遮断された場合は OFF を指示してから ON を指示し直すまで復帰しません
- **ドライブ電源に流す電流は、定常 10A、ピーク 30A を超えないでください**

9.3 外部インターフェース(gRPC)

詳細は 10 章「gRPC API リファレンス」を参照してください。

RPC	内容
`SetPowerConfig`	ドライブ電源の ON(1)/OFF(0) 制御
`StreamPowerStatus`	電圧・電流の値を受け取り続ける

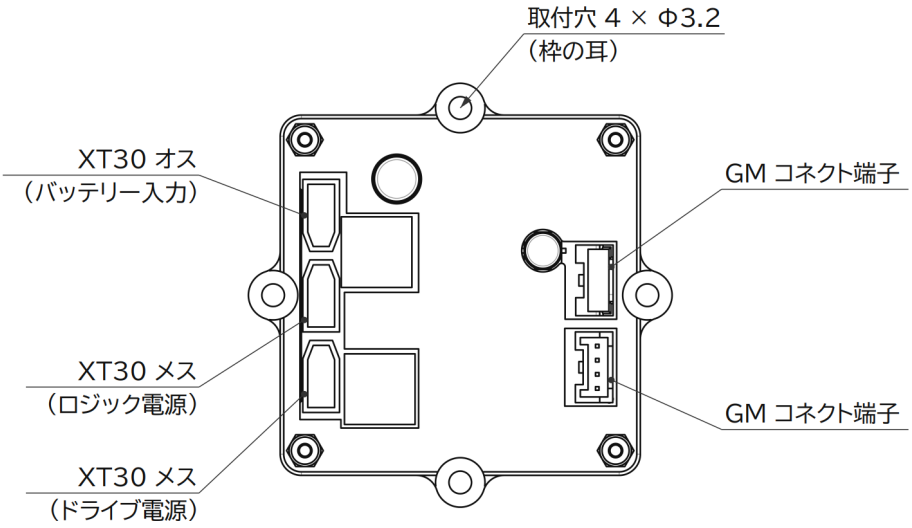
9.4 電気仕様・コネクタ

基板は 1 種類です。

項目	値
対応バッテリー	2〜6 セル LiPo(7.6V〜25.2V)
入力電圧の上限	26V
GM コネクトへの供給電圧	約 8.2V
GM コネクトへの供給電流	最大 2.5A
ドライブ電源に流せる電流	定常 10A、ピーク 30A

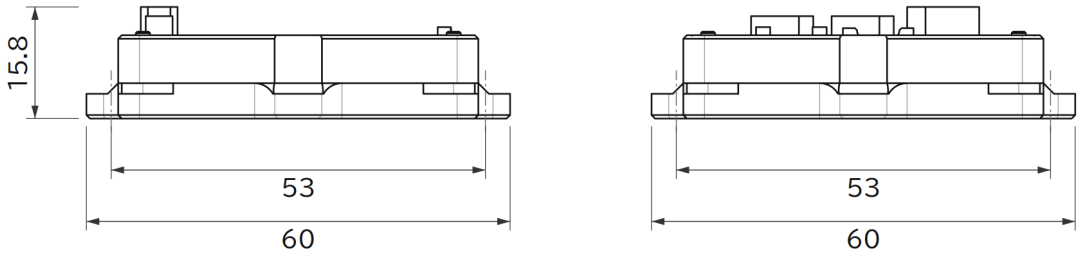
コネクタ	種類・数
バッテリー入力	XT30 オス × 1
ロジック電源	XT30 メス × 1(バッテリー入力に直結。ON/OFF・遮断なし)
ドライブ電源	XT30 メス × 1(ON/OFF・保護機能による遮断あり)
GM コネクト端子	PA コネクタ 4 ピン × 2

9.5 外形・取付



上面図

寸法の単位は mm



正面図　右側面図

項目	値
外形寸法	60mm × 60mm × 15.8mm
取付穴	4 × Φ3.2mm(向かい合う穴の間隔 53mm)

- ヒットパネルモジュールの背面にプッシュリベット(NP-3013)で取り付けられます
- プッシュリベットは簡単に取れないので、取り外すにはプッシュリベット取り外し用の器具等を用いてください
- 放熱を妨げない場所に取り付けてください

9.6 注意事項

- バッテリーは 2〜6 セルの LiPo を使い、入力電圧は 26V 以下にしてください
- **ドライブ電源に流す電流は、定常 10A、ピーク 30A を超えないでください**
- **ロジック電源 / ドライブ電源は、どちらも入力電源の電圧のまま出力します**(電圧変換は行いません)

- ドライブ電源につなぐ機器の範囲(アクチュエータ等)は、競技ルールで決まります。ATTRACTS では ATTRACTS ドキュメントの「AT-2 ビルドマニュアル」を参照してください

10. gRPC API リファレンス

ゲームモジュールサーバーは、接続された全ゲームモジュールを **gRPC**(ポート 50051)で外部に公開します。

> API の定義は GameModuleClientSDK リポジトリの proto ファイルにあります。

SDK と gRPC の関係

公式 SDK(Node.js/TypeScript)は、この gRPC API を使いやすくまとめたライブラリです。SDK にできることはすべて gRPC で直接できます。Python・C#・Go など、gRPC が対応する任意の言語で proto からコードを生成すれば、SDK なしでクライアントを自作できます。

```
# 例: Python でクライアントコードを生成する
python -m grpc_tools.protoc -I proto --python_out=. --grpc_python_out=. \
  proto/game_module.proto proto/modules/*.proto
```

接続情報

項目	値
エンドポイント	`<ゲームモジュールサーバーのIP>:50051` (平文 gRPC)
サービス探索	mDNS(SDK は自動で見つける。自作のクライアントでは IP アドレスを直接指定するのが簡単)

サービス一覧

GameModuleService(ゲームモジュール管理)

`proto/game_module.proto`

RPC	型	内容
`ListModules`	server streaming	接続中のゲームモジュールの一覧。最初に現在の一覧を返し、その後はつながりが変わるたびに送る

`ModuleInfo` の主なフィールド:

フィールド	内容
`hardware_number`	ゲームモジュール 1 台ごとに割り当てられた番号(uint64)。各ゲームモジュールサービスの RPC で対象指定に使う
`parent_hardware_number`	ゲームモジュールサーバー側の隣のゲームモジュールの HardwareNumber(0 はゲームモジュールサーバーに直接つながっている)。ロボットごとにまとめるのに使える
`module_number`	ゲームモジュールの製品番号(下表)
`version`	ゲームモジュールのファームウェアバージョン(v0.0.0 はアップデート待ち)
`ping_time` / `no_response_count`	通信品質の目安

`ModuleNumber` (ゲームモジュールの製品番号):

値	ゲームモジュール
0x0001	カメラモジュール
0x0002	パワーマネジメントモジュール
0x0003	ヒットパネルモジュール
0x0004	トランシーバーモジュール
0x0006	マズルモジュール

カメラモジュール

`proto/modules/camera_module.proto`

RPC	型	内容
`StreamCameraStatus`	server streaming	`CameraStatus` (`rtsp_url`)を配信

ヒットパネルモジュール

`proto/modules/ground_hit_panel_module.proto`

RPC	型	内容
`StreamHitEvents`	server streaming	着弾ごとに `HitEvent` (発生元 `hardware_number`)を配信
`SetLEDConfig`	unary	LED 色設定(`red` / `green` / `blue` : 0–255)

マズルモジュール

`proto/modules/muzzle_module.proto`

RPC	型	内容
`StreamShotEvents`	server streaming	発射ごとに `ShotEvent`（`projectile_speed`：0.1m/s 単位）を配信
`SetLEDConfig`	unary	LED 3 個の色設定（`led1` / `led2` / `led3` すべて必須）
`SetUVLEDEnable`	unary	UV LED の有効/無効

パワーマネジメントモジュール

`proto/modules/power_management_module.proto`

RPC	型	内容
`SetPowerConfig`	unary	電源制御（`power_control`：0=OFF, 1=ON）
`StreamPowerStatus`	server streaming	`PowerStatus`（`voltage` [V], `current` [A]）を配信

トランシーバーモジュール

`proto/modules/transceiver_module.proto`

RPC	型	内容
`SendUARTData`	unary	バイナリデータをユーザ MCU へ送信（最大 64 バイト）
`StreamUARTData`	server streaming	ユーザ MCU からの受信データ（`UARTRecvEvent`）を配信
`SendFramedData`	unary	フレーミングプロトコルで送信（フレーム仕様は 8.2.3 節）

共通の使い方

- **対象指定**: 各 RPC のリクエストには対象ゲームモジュールの `hardware_number` を含めます。まず `ListModules` で一覧を取得し、`module_number` で製品を絞り込みます
- **イベントは server streaming**: 着弾・発射・ステータスは「ストリームを開いて受け続ける」形で受信します
- **ロボットごとにまとめる**: `parent_hardware_number` をたどると「同じロボット（同じカメラモジュール配下）のゲームモジュール」を特定できます

SDK の使い方

SDK の使い方（接続、`getModule`、`pullHitEvents` 等）は GameModuleClientSDK の README を参照してください。

11. SDK で自分の競技を作る

アプリの画面で試した操作は、すべてプログラムからも行えます。GameModuleClientSDK(Node.js/TypeScript 用のライブラリ)を使うと、ゲームモジュールサーバーにつないで自分の競技を作れます。SDK を使わず、公開されている proto 定義から C# など任意の言語で作ることもできます。

2 章の手順 3 で接続した Wi-Fi ネットワーク内の PC で、次を実行します。Node.js 24 以上が必要です。

```
git clone https://github.com/ATTRACTS-Community/GameModuleClientSDK.git
cd GameModuleClientSDK/nodejs
npm install
npx tsx examples/ShowModules.ts   # 接続されているモジュールの一覧を表示
```

ゲームモジュール一覧が表示されれば準備完了です。競技の作り方は、ヒットパネルモジュールを使ったミニゲームを例に次の節で説明します。

サンプル競技「ターゲットストライク」を作る

この節では、**ミニ競技を約 100 行の TypeScript で作ります**。

作るもの: ターゲットストライク

フィールドに置いたヒットパネルモジュールを的にする、1 人用の的当てです。

- フィールドにヒットパネルモジュールを 3 枚置く
- **光っている的(緑)は 1 枚だけ**。それに当てると 10 点、的は別のパネルに移る
- 光っていない的(赤)に当てると -5 点
- 制限時間は 60 秒
- 得点はコンソールに表示し、残り時間はマズルの LED 色で通知する

必要なもの

- ヒットパネルモジュール × 3(1 枚でも動くように作ります)
- マズルモジュール × 1(任意。無くても遊べます)
- カメラモジュール + パワーマネジメントモジュール(ほかのゲームモジュールをつなぐために必要)
- 2 章「はじめて動かす」と、この章のはじめの準備を終えていること

Step 1: ゲームモジュールを列挙する

まず、パネルとマズルを取得できることを確認します。

```
// target-strike.ts
import {
  GameModuleClient, ModuleNumber,
  GroundHitPanelModule, MuzzleModule,
```

```
} from "@gamemodule/client";

const client = await GameModuleClient.connect();

const panels = client.getModules(
  ModuleNumber.MODULE_NUMBER_GROUND_HIT_PANEL_MODULE) as GroundHitPanelModule[];
const muzzle = client.getModule(
  ModuleNumber.MODULE_NUMBER_MUZZLE_MODULE) as MuzzleModule | undefined;

console.log(`パネル ${panels.length} 枚 / マズル ${muzzle ? "あり" : "なし"}`);
if (panels.length === 0) throw new Error("パネルが見つかりません");
```

```
npx tsx target-strike.ts
```

Step 2: ゲームの状態とルールを書く

ゲームの処理は、ゲームモジュールを使わない普通の TypeScript で書きます。ルールは自由に決められます。

```
const GAME_SECONDS = 60;
const HIT_SCORE = 10;
const MISS_PENALTY = -5;

let score = 0;
let targetIndex = 0;                  // いま光っている的
let endTime = Date.now() + GAME_SECONDS * 1000;

function nextTarget() {
  // いまと違うパネルをランダムに選ぶ(1枚しか無ければそのまま)
  if (panels.length === 1) return;
  let next;
  do { next = Math.floor(Math.random() * panels.length); } while (next === targetIndex);
  targetIndex = next;
}
```

Step 3: LED で的を表示する

```
async function renderTargets() {
  for (let i = 0; i < panels.length; i++) {
    if (i === targetIndex) await panels[i].setLed(0, 255, 0);   // 的: 緑
    else await panels[i].setLed(255, 0, 0);                     // それ以外: 赤
  }
}
```

Step 4: 着弾を判定する

イベントは、たまったものを一定の間隔で取り出して処理します(pull 型の API)。

```
async function tick() {
  // 着弾判定
  for (let i = 0; i < panels.length; i++) {
    const hits = panels[i].pullHitEvents();
    if (hits.length === 0) continue;

    if (i === targetIndex) {
      score += HIT_SCORE;
      console.log(`命中! +${HIT_SCORE} 点 (計 ${score} 点)`);
      nextTarget();
      await renderTargets();
    } else {
      score += MISS_PENALTY;
      console.log(`的じゃない! ${MISS_PENALTY} 点 (計 ${score} 点)`);
    }
  }

  // 残り時間をマズルの LED で表示(白→黄→赤)
  const remain = (endTime - Date.now()) / 1000;
  if (muzzle) {
    const c: [number, number, number] =
      remain > 30 ? [255, 255, 255] : remain > 10 ? [255, 160, 0] : [255, 0, 0];
    await muzzle.setLed(c, c, c);
  }
}
```

Step 5: ゲームループで仕上げる

```
console.log(`ターゲットストライク開始! ${GAME_SECONDS} 秒`);
nextTarget();
await renderTargets();

const timer = setInterval(async () => {
  await tick();
  if (Date.now() >= endTime) {
    clearInterval(timer);
    console.log(`\n===== 終了! スコア: ${score} 点 =====`);
    for (const p of panels) await p.setLed(0, 0, 255); // 全パネル青で終了表示
    await client.close();
  }
}, 20);
```

これで完成です。パネルをフィールドに並べて実行してください。

```
npx tsx target-strike.ts
```

発展課題

たとえば次のように広げられます。

アイデア	使う API
2 人対戦にする(パネルを 2 組に分ける)	`getModules` + HardwareNumber での組分け
当たったら相手の電源を 3 秒落とす	パワーマネジメントモジュール `setPower`
ロボット搭載マイコンに得点を送って表示させる	トランシーバーモジュール `sendData`
一人称視点でプレイする	カメラモジュールの RTSP URL + VLC

12. トラブルシューティング

症状	確認すること
方法 A で「見つかったサーバ」に何も表示されない	PC が SSID `CameraModule-` で始まる Wi-Fi ネットワークに接続しているか
ゲームモジュールが一覧に表示されない	パワーマネジメントモジュールにバッテリーがつながっているか / カメラモジュールの LED 点灯 / SSID が `ATTRACTS_WIFI` の Wi-Fi ネットワークが複数存在しないか
一部のゲームモジュールが表示されない	そのゲームモジュールの GM コネクトケーブルを挿し直し、バッテリーもつなぎ直す
SDK のサンプルが接続エラーになる	ゲームモジュールサーバーが立ち上がっているか / PC が同じネットワークにいるか
`npm install` が失敗する	`node --version` が v24 以上か

13. 用語集

用語	意味
gRPC	Google 発のオープンな RPC(遠隔手続き呼び出し)フレームワーク。ゲームモジュールサーバーの公開 API に使用
proto(Protocol Buffers)	gRPC の API を定義するファイル。GameModuleClientSDK リポジトリで公開
SDK	GameModuleClientSDK。gRPC API を Node.js/TypeScript から扱いやすくしたライブラリ
GM コネクト	ゲームモジュールどうしをつなぐ接続方式。PA コネクタ 4 ピンの端子と GM コネクトケーブルで、電源と通信を送る(4 章)
HardwareNumber	ゲームモジュール 1 台ごとに割り当てられた 64bit の番号
ModuleNumber	ゲームモジュールの製品番号(例: `0x0006` = マズルモジュール)