EPSON

VC 判定 振動計測システム

ユーザーズガイド

1. 本評価ボード・キット、開発ツールは、お客様での技術的評価、動作の確認および開発のみに用いられることを想定し設計されています。それらの技術評価・開発等の目的以外には使用しないでください。本品は、完成品に対する設計品質に適合していません。
2. 本評価ボード・キット、開発ツールは、電子エンジニア向けであり、消費者向け製品ではありません。お客様において、適切な使用と安全に配慮願います。弊社は、本品を用いることで発生する損害や火災に対し、いかなる責も負いかねます。通常の使用においても、異常がある場合は使用を中止してください。
3. 本評価ボード・キット、開発ツールに用いられる部品は、予告なく変更されることがあります。

● 本資料のご使用につきましては、次の点にご留意願います。

本資料の内容については、予告なく変更することがあります。

1. 本資料の一部、または全部を弊社に無断で転載、または、複製など他の目的に使用することは堅くお断りします。
2. 弊社製品のご購入およびご使用にあたりましては、事前に弊社営業窓口で最新の情報をご確認いただきますとともに、弊社ホームページなどを通じて公開される最新情報に常にご注意ください。
3. 本資料に掲載されている応用回路、プログラム、使用方法などはあくまでも参考情報です。お客様の機器・システムの設計において、応用回路、プログラム、使用方法などを使用する場合には、お客様の責任において行ってください。これらに起因する第三者の知的財産権およびその他の権利侵害ならびに損害の発生に対し、弊社はいかなる保証を行うものではありません。また、本資料によって第三者または弊社の知的財産権およびその他の権利の実施権の許諾を行うものではありません。
4. 弊社は常に品質、信頼性の向上に努めていますが、一般的に半導体製品は誤作動または故障する場合があります。弊社製品のご使用にあたりましては、弊社製品の誤作動や故障により生命・身体に危害を及ぼすこと又は財産が侵害されることのないように、お客様の責任において、お客様のハードウェア、ソフトウェア、システムに必要な安全設計を行うようお願いします。なお、設計および使用に際しては、弊社製品に関する最新の情報(本資料、仕様書、データシート、マニュアル、弊社ホームページなど)をご確認いただき、それに従ってください。また、上記資料などに掲載されている製品データ、図、表などに示す技術的な内容、プログラム、アルゴリズムその他応用回路例などの情報を使用する場合は、お客様の製品単独およびシステム全体で十分に評価を行い、お客様の責任において適用可否の判断をお願いします。
5. 弊社は、正確さを期すために慎重に本資料およびプログラムを作成しておりますが、本資料およびプログラムに掲載されている情報に誤りがないことを保証するものではありません。万一、本資料およびプログラムに掲載されている情報の誤りによってお客様に損害が生じた場合においても、弊社は一切その責任を負いかねます。
6. 弊社製品の分解、解析、リバースエンジニアリング、改造、改変、翻案、複製などは堅くお断りします。
7. 弊社製品は、一般的な電子機器(事務機器、通信機器、計測機器、家電製品など)および本資料に個別に掲載されている用途に使用されることを意図して設計、開発、製造されています（一般用途）。特別な品質、信頼性が要求され、その誤動作や故障により生命・身体に危害を及ぼす恐れ、膨大な財産侵害を引き起こす恐れ、もしくは社会に深刻な影響を及ぼす恐れのある以下の特定用途に使用されることを意図していません。お客様に置かれましては、弊社製品を一般用途に使用されることを推奨いたします。もし一般用途以外の用途で弊社製品のご使用およびご購入を希望される場合、弊社はお客様の特定用途に弊社製品を使用されることへの商品性、適合性、安全性について、明示的・黙示的に関わらずいかなる保証を行うものではありません。

   【特定用途】

   宇宙機器(人工衛星・ロケットなど) / 輸送車両並びにその制御機器(自動車・航空機・列車・船舶など)
   医療機器（本資料に個別に掲載されている用途を除く） / 海底中継機器 / 発電所制御機器 / 防災・防犯装置
   交通用機器 / 金融関連機器
   上記と同等の信頼性を必要とする用途
8. 本資料に掲載されている弊社製品および当該技術を国内外の法令および規制により製造・使用・販売が禁止されている機器・システムに使用することはできません。また、弊社製品および当該技術を大量破壊兵器等の開発および軍事利用の目的その他軍事用途等に使用しないでください。弊社製品または当該技術を輸出または海外に提供する場合は、「外国為替及び外国為替法」、「米国輸出管理規則(EAR)」、その他輸出関連法令を遵守し、係る法令の定めるところにより必要な手続きを行ってください。
9. お客様が本資料に掲載されている諸条件に反したことに起因して生じたいかなる損害（直接・間接を問わず）に関して、弊社は一切その責任を負いかねます。
10. お客様が弊社製品を第三者に譲渡、貸与などをしたことにより、損害が発生した場合、弊社は一切その責任を負いかねます。
11. 本資料についての詳細に関するお問合せ、その他お気付きの点などがありましたら、弊社営業窓口までご連絡ください。
12. 本資料に掲載されている会社名、商品名は、各社の商標または登録商標です。

2022.08

商標

- Raspberry Piis a trademark of Raspberry PiLtd.
- マイクロソフト、Windows は、マイクロソフト グループの企業の商標です。
- EPSON はセイコーエプソン株式会社の登録商標です。
- その他の製品名は各社の商標または登録商標です。

目次

改訂履歴

Rev. No.	改訂日	Page	改訂内容
20250331	2025/3/31	ALL	初版制定 MSG006-001a_v1.0.0 リリースに対応

1. 関連文書

1. 『Raspberry Pi 社製品を利用した振動計測システム セットアップマニュアル』Rev.20250331
2. 『Raspberry Pi 社製品を利用した振動計測システム オペレーションマニュアル』Rev.20250331
3. 『VC 判定ライブラリ リファレンスマニュアル』Rev.20250221
4. 『振動計測システム モニタリングアプリ ユーザーズガイド』Rev.20250331
5. 『BitTradeOne 社製 リレー制御拡張基板 ADRSRU4』(https://bit-trade-one.co.jp/product/module/adrsru/)

2. はじめに

本ユーザーズガイドは、「VC 判定 振動計測システム」について、以下の項目を説明するものです。

- 仕様
- セットアップ方法
- 実行方法
- 開発者向けガイド

「VC 判定 振動計測システム」（以下「本アプリ」と記載します）は、「Raspberry Pi 社製品を利用した振動計測システム」（以下「ラズパイロガー」と記載します）と「VC 判定ライブラリ」を組み合わせたアプリケーションです。

本アプリは、ラズパイロガーによる加速度センサーからの計測データを、VC 判定ライブラリにより計算し、その結果を出力します。

また、Raspberry Pi に「BitTradeOne 社製 リレー制御拡張基板 ADRSRU4」（以下「リレー制御基板」と記載します）を実装することで、振動レベルに応じたリレー制御を行うことが可能です。

本アプリはラズパイロガーによる「MQTT ブローカー」を通じた状態メッセージの送信に加えて、VC 判定ライブラリの計算結果を MQTT メッセージとして送信する機能を備えます。送信されたメッセージは「振動計測システム モニタリングアプリ」（以下「モニタリングアプリ」と記載します）を使用して、PC から Web ブラウザで確認することができます。

なお、本アプリは、基本的な実行メカニズムをラズパイロガーにより実現しているため、実行方法や計測方法はラズパイロガーに準じます。

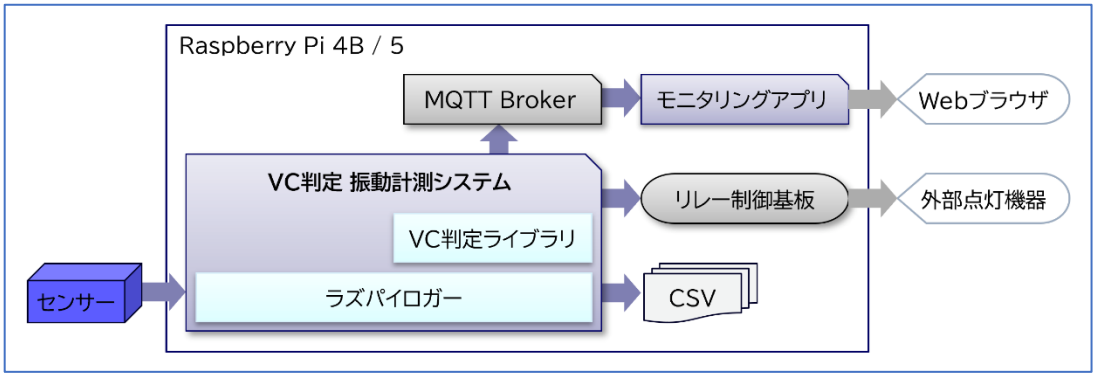

図 2-1　全体構成図

3. 仕様

3.1. 動作確認環境

- Raspberry Pi 4B
- Raspberry Pi 5

3.2. 対応センサー

- M-A352AD
- M-A552AR

※ 注意： M-A342VD, M-A542VR には対応していません。

3.3. アプリケーション設定項目

本アプリはラズパイロガーをベースにしているため、ラズパイロガーの設定項目はそのまま適用されます。詳細は「関連文書 2」を参照してください。

本アプリに特有の設定項目を以下に記載します。

表 3-1

設定項目名	説明	設定可能な値	補足
VC_FFT_SIZE	VC 判定 FFT 演算ポイント数	2000～20000 の偶数	*1
VC_EXE_SIZE	VC 判定 実行データ数	VC_FFT_SIZE を割り切れる整数	*1
VC_AVG_SIZE	VC 判定 VC 平均出力での平均数	10～50 の整数	*1
VC_LOG_SINGLE	VC 判定の SINGLE 判定結果をログ出力するか	True, False	
VC_LOG_AVERAGE	VC 判定の AVERAGE 判定結果をログ出力するか	True, False	
VC_OUTPUT_RAW	計測データの CSV を出力するか	True, False	
VC_CONTROL_GPIO	GPIO 制御を行うか	True, False	

- *1：詳細は「関連文書 3」を参照してください

3.4. 実行仕様

本アプリの実行方法は、ラズパイロガーに準じます。詳細は「関連文書 2」を参照してください。

3.5. 入力仕様

本アプリは、USB で接続された複数台の M-A352AD/M-A552AR センサーによる計測が可能です。また、サンプリングレートは 1000 SPS のみ対応しています。

※ 注意： M-A342VD/M-542VR センサーには対応していないため、接続した場合はエラーが発生し終了します。

※ 注意： 設定項目 A352_SPS でサンプリングレートを 1000 SPS 以外に設定した場合、エラーが発生し終了します。

3.6. 出力仕様

本アプリは、ラズパイロガーの計測機能に加えて VC 判定計算を行うため、以下の種類の情報を出力します。

1. 計測データファイル

2. 計測情報ファイル
3. VC 判定データファイル
4. VC 判定レベル（ログファイルに出力）
5. VC 判定レベル（GPIO に出力）
6. 状態メッセージ
7. ログファイル

3.6.2. 計測データファイル

計測データが記録された計測データファイルについて、ラズパイロガーの仕様に準じて出力します。

設定項目 `VC_OUTPUT_RAW` が `False` に設定されている場合、このファイルは出力されません。

3.6.3. 計測情報ファイル

計測に関する情報を記録した計測情報ファイルについて、本アプリはラズパイロガーの出力内容に加えて、本アプリで追加した設定項目を出力します。

3.6.4. VC 判定データファイル

本アプリは計測データに VC 判定計算を行い、計算が完了した段階のデータを CSV ファイルに出力します。VC 判定ライブラリでは「SINGLE」と「AVERAGE」の二種類の計算を行うため、これを別々のファイルに出力します。

VC 判定データファイルのファイル名は、以下のとおりです：

- SINGLE： 「センサーモデル_センサーシリアル_vccalc_sgl.csv」
- AVERAGE： 「センサーモデル_センサーシリアル_vccalc_avg.csv」

VC 判定データファイルのファイルレイアウト（SINGLE,AVERAGE 共通）は、以下のとおりです：

- 出力データ項目（119 項目/行）：
  - 連番
  - 出力時分秒
  - VC 判定レベル（OA, A, B, C, D, E, F, G）
  - 以降「Composite」「X Axis」「Y Axis」「Z Axis」それぞれについて以下のデータ（116 項目）：
    - Peak Velocity (mm/s)
    - Peak Frequency (Hz)
    - VC 判定の 1/3 オクターブバンド周波数毎の速度演算結果（27 項目）
- 見出し行： 2 行構成
  - 1 行目： No, Time, VC-Level, 「Composite」「X Axis」「Y Axis」「Z Axis」の見出し
  - 2 行目： Peak Velocity, Peak Frequency, 1/3 オクターブバンド周波数毎の見出し
- データ行： 3 行目以降

No	Time	VC-Level	Composite				
			Peak Velocity (mm/s)	Peak Frequency (Hz)	1.00 (Hz)	1.26 (Hz)	…
1	2025/03/01 10:00:00	C	0.011457	10.08	0.000321	0.000221	…
2	2025/03/01 10:00:01	C	0.011258	10.08	0.000307	0.000225	…
…	…	…	…	…	…	…	…

図 3-1 VC 判定データファイル イメージ

3.6.5. VC 判定レベル（ログファイルに出力）

VC 判定計算結果のうち、VC 判定レベルの値をログに出力します。

設定項目 VC_LOG_SINGLE, VC_LOG_AVERAGE が False に設定されている場合、該当する結果は出力されません。

3.6.6. VC 判定レベル（GPIO に出力）

VC 判定計算結果のうち、VC 判定レベルの値に応じて GPIO を制御し、リレー制御基板に出力します。

設定項目 VC_CONTROL_GPIO が False に設定されている場合、この制御を行いません。

GPIO の制御はリレー制御基板の仕様に準じます。

表 3-2　ADRSRU4　GPIO 仕様

リレー番号	GPIO 番号
1	4
2	17
3	27
4	22

これに対し、各 VC 判定レベルについて、以下のようにリレー基盤を制御します。

表 3-3　VC 判定レベル別 リレー制御

	VC 判定レベル							
リレー番号	OA	A	B	C	D	E	F	G
1	ON							
2		ON						
3			ON					
4				ON	ON	ON	ON	ON

3.6.7. 状態メッセージ

本アプリは、VC 判定により計算される VC 判定レベルと FFT の結果を MQTT メッセージとして送信します。
送信されたメッセージは、モニタリングアプリを使って PC の Web ブラウザから確認できます。詳細は「関連文書 4」を参照ください。

本アプリの状態メッセージの仕様を以下に記載します。

表 3-4　状態メッセージ仕様

メッセージ種類	トピック名	メッセージ内容
センサー VC 判定レベル	logger/$LOG/sensor/$MDL/$SNO/vc	level: “OA”|”A”|”B”|”C”|”D”|”E”|”F”|”G”
センサー FFT	logger/$LOG/sensor/$MDL/$SNO/fft	value: [0.000282,0.000124,0.000162,…]

- トピック名の中の $ 要素は、それぞれ以下の値が設定されます：
  - $LOG：設定ファイルの LOGGER_ID 設定値、$MDL：センサーモデル名、$SNO：センサーシリアル NO
- FFT メッセージの value 項目は、「Composite」として計算された 27 個の数値を含んだ配列です。
  - 各々の周波数はメッセージ項目に含みません。
- メッセージ内容は、以下の詳細事項が適用されます：
  - メッセージ形式はいずれも JSON 文字列です。
  - すべてのメッセージに「timestamp: “yyyy/mm/dd hh:mm:ss.nnnnnn”」項目を含みます。

3.6.8. ログファイル

本アプリはラズパイロガーの仕様に準じてログファイルを出力します。

本アプリから出力する内容は、以下のとおりです：

- VC 判定データファイル名
- VC 判定レベル

3.6.9. 出力フォルダ

本アプリから出力するファイルは、いずれもラズパイロガーの仕様に準じて「計測年月日時分秒」フォルダに出力します。

4. Raspberry Pi へのセットアップ

本アプリを Raspberry Pi にセットアップする手順を説明します。

4.1. ファイル・フォルダ構成

ダウンロードした ZIP ファイルを展開したファイル・フォルダ構成は以下のとおりです。

```
展開フォルダ
├──── bin                                    実行環境の構築に必要な設定ファイル
├──── pyproject.toml                         Python プロジェクト設定ファイル
├──── src                                    本アプリの Python ソースコード
├──── sub                                    本アプリが参照するライブラリのインストール用
│     ├──── raspi_logger-1.2.0-py3-none-any.whl    ラズパイロガーのインストールファイル
│     └──── vc_calculation-1.0.0-py3-none-any.whl  VC 判定ライブラリのインストールファイル
└──── .env.default                           本アプリの設定ファイルのテンプレート
```

図 4-1　ファイル・フォルダ構成

4.2. 事前準備

本アプリのセットアップを開始する前に、「関連文書 1」を参照し、以下の準備を行ってください。

- Raspberry Pi と PC をネットワークで接続する
- Raspberry Pi からインターネットに接続する
- Raspberry Pi に MQTT ブローカー（mosquitto）をインストールする

本ガイドの以下の手順では、以下の想定で記載しています：

- Raspberry Pi に作成したユーザー名：　`pi`
- Raspberry Pi に設定した固定 IP アドレス：　`192.168.1.52`

ユーザー名・固定 IP アドレスに異なるものを設定した場合には、適宜読み替えて実行してください。

4.3. Raspberry Pi へ ZIP ファイルを転送

以下のコマンドを実行して、ダウンロードした ZIP ファイルを Raspberry Pi に転送します。

- `scp MSG006-001a.zip pi@192.168.1.52:.`

`pi` ユーザーのホームディレクトリにファイルが転送されます。

4.4. Raspberry Pi にアプリケーションをインストール

以下の手順に従い、Raspberry Pi にアプリケーションをインストールします。

1. Raspberry Pi にログインします
   - `ssh pi@192.168.1.52`
2. アプリケーションをインストールするディレクトリを作成します
   - `sudo mkdir /app`　　（※ /app フォルダがない場合）
   - `sudo chown pi:pi /app`　　（※ /app フォルダがない場合）
   - `mkdir -p /app/MSG006-001a`

3. アプリケーションの ZIP ファイルを展開します
   - `cd /app/MSG006-001a`
   - `unzip ~/MSG006-001a.zip`
4. アプリケーション実行用に Python の仮想環境を作成します
   （仮想環境の作成では、OS にインストールされている GPIO 関連モジュールを使用するため、追加オプションを指定します）
   - `python -m venv --system-site-packages venv`
   - `source venv/bin/activate`
5. `pip` を最新のバージョンにアップグレードします
   - `pip install -U pip`
6. 本アプリが参照するライブラリをインストールします
   - `pip install . sub/*.whl`
7. 本アプリを含み、各アプリを実行するサービス登録ファイルを必要に応じて OS にインストールします
   - 本アプリ
     `sudo cp bin/vc_calc_app@.service /etc/systemd/system`
   - ラズパイロガー
     `sudo cp bin/logger@.service /etc/systemd/system`
   - ラズパイロガー付属 ハードウェア状態モニター
     `sudo cp bin/hwmonitor.service /etc/systemd/system`
8. サービス登録ファイルを OS に読み込ませます
   - `sudo systemctl daemon-reload`

※ 注意： 各サービス登録ファイルは、Python 仮想環境が `/app/MSG006-001a` にあることを前提としています。
異なるパスに仮想環境を作成した場合は、各ファイルの以下の行を適切なパスに修正してください。
`ExecStart=/app/MSG006-001a/venv/bin/python -m` （以降はアプリにより異なります）

4.5. アプリケーション設定

以下のコマンドで設定ファイルのテンプレート `.env.default` をコピー、設定ファイルを作成します。

- `cp .env.default .env`

必要に応じてテキストエディタで設定を変更します。

設定項目は「3.3 アプリケーション設定項目」を参照ください。

5. アプリケーション実行

本アプリの実行は、ラズパイロガーの実行方法に準じます。

- OS 起動時に自動的に計測開始するよう本アプリを登録する
  - `sudo systemctl enable vc_calc_app@auto.service`
- 設定ファイルに設定した計測時間で本アプリを起動する
  - `sudo systemctl start vc_calc_app@manual.service`
- 計測時間（秒数）を指定して本アプリを起動する
  - `sudo systemctl start vc_calc_app@計測時間.service`

または、以下のように `python` コマンドを指定した実行も可能です。

- `python -m vc_calc_app 計測秒数`

詳細は「関連文書 2」を参照ください。

6. Appendix： 開発者向けガイド

この章は開発者向けガイドとして、本アプリの機能追加や改修だけでなく、ラズパイロガーをベースとした応用アプリケーションを開発するのに必要となる情報を記載します。

- 開発環境構築
- ラズパイロガー カスタマイズ ガイド
- 本アプリ カスタマイズ ポイント

6.1. 開発環境構築

ダウンロードした ZIP ファイルを、ローカル PC の任意のフォルダに展開した前提で説明します。

6.1.1. Python 仮想環境作成

ターミナルを起動し、ZIP ファイルを展開したフォルダに Python 仮想環境を作成します。

Windows の場合：

- PowerShell を起動し、以下のコマンドを実行します。
  - `py -3.11 -m venv venv`
  - `venv\scripts\activate.ps1`

Mac や Linux の場合：

- ターミナルを起動し、以下のコマンドを実行します。
  - `python3.11 -m venv venv`
  - `source venv/bin/activate`

6.1.2. パッケージ インストール

本アプリが使用するパッケージをインストールします。

Windows の場合：

- `python -m pip install -U pip`
- `pip install -e .`

Mac や Linux の場合：

- `pip install -U pip`
- `pip install -e .`

本アプリのソースコードは「editable mode」で仮想環境にインストールします。

また、開発やテストでのみ使用するパッケージをインストールするには、以下を実行します。

Windows の場合：

- `pip install -e .[dev,test]`

Mac や Linux の場合：

- `pip install -e ".[dev,test]"`

6.2.　ラズパイロガー　カスタマイズ ガイド

6.2.1.　ラズパイロガー プログラム概要

ラズパイロガーは、Raspberry Pi に USB 接続された振動センサーを使って振動データを計測し、CSV ファイルに計測データを保存するプログラムです。この一連の処理は、大まかに以下の流れで行われています。

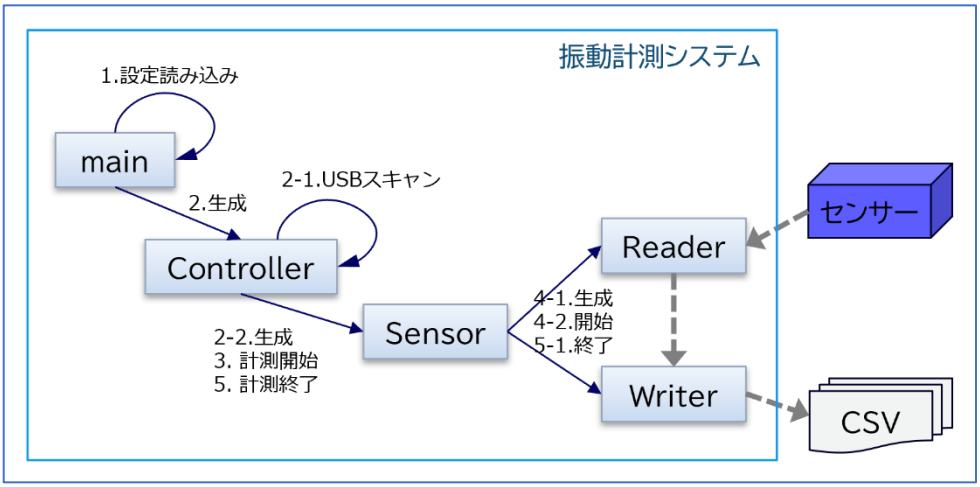

図 6-1　ラズパイロガー シーケンス

1. プログラムが起動されると、プログラムは設定ファイルから設定値を読み取る
2. プログラムは Raspberry Pi に接続された USB をスキャンし、見つかったセンサーのモデル名に応じて、センサーを制御する Sensor オブジェクトを生成する
3. プログラムは Sensor オブジェクトに計測開始を指示する
4. Sensor オブジェクトは、センサーからデータを読み取る Reader プロセスと、読み取ったデータを出力する Writer プロセスを生成し、計測処理を行う
5. あらかじめ設定された時間や外部からのシグナルに応じて、プログラムは計測を停止する

USB に複数のセンサーが接続されている場合、Sensor・Reader・Writer のセットが各センサーに応じて生成されます。例えば 4 つのセンサーが接続されている場合、次の図のようにオブジェクトが構成されます。

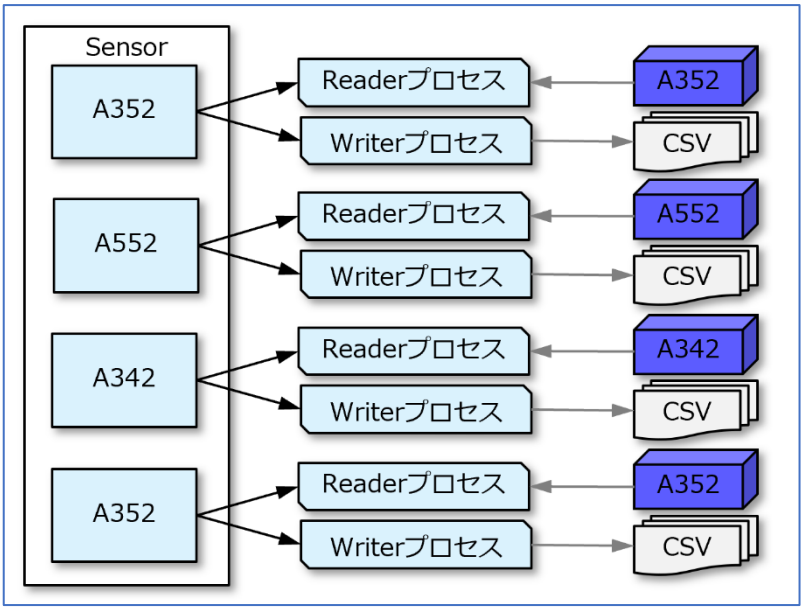

図 6-2　ラズパイロガー オブジェクト構成

6.2.2. ラズパイロガー カスタマイズ 概要

ラズパイロガーはマルチプロセスで動作する Python プログラムですが、拡張性を持たせるため、主要なオブジェクトを置き換えることを可能にする仕掛けが備わっています。

主要なオブジェクトは単一の「Factory オブジェクト」から生成されるように作られており、この Factory オブジェクトを置き換えることで主要なオブジェクトの拡張バージョンに入れ替えることが可能になります。

これはデザインパターンの Abstract Factory パターンに該当します。

Factory オブジェクトは `main` 関数に渡すことができるため、`main` 関数を呼び出す前に拡張した Factory オブジェクトを生成し `main` 関数に渡すことで、Factory オブジェクトを置き換えることが可能です。

6.2.3. `logger.LoggerFactory` クラス

ラズパイロガーのプログラム内で使われる主要なオブジェクトを生成する Factory オブジェクトのインタフェースクラスです。

このクラスの実体は `logger.core.LoggerFactoryImpl` にあります。

拡張版の Factory を定義する場合は、`LoggerFactoryImpl` を継承し、必要なメソッドを拡張した Factory クラスを定義します。

`LoggerFactory` は以下の Factory メソッドを備えます。

6.2.3.1. `create_configurator()` メソッド

ラズパイロガーの設定ファイルをチェックし、設定値を保持した `logger.core.Config` オブジェクトを生成する `logger.core.Configurator` オブジェクトを生成します。

設定値を追加したい場合に、拡張版の `Configurator` クラスを定義し、そのオブジェクトを生成して返すようにこのメソッドを定義します。

6.2.3.2. `create_A342()` メソッド

ラズパイに A342 センサーが接続されている場合、A342 センサーを制御する `logger.measure.A342` オブジェクトを生成します。

A342 センサーに対するデータ読み取り・書き出しの振る舞いを変更したい場合に、拡張版の `A342` クラスを定義し、そのオブジェクトを生成して返すようにこのメソッドを定義します。

6.2.3.3. `create_A352()` メソッド

ラズパイに A352 センサーが接続されている場合、A352 センサーを制御する `logger.measure.A352` オブジェクトを生成します。

A352 センサーに対するデータ読み取り・書き出しの振る舞いを変更したい場合に、拡張版の `A352` クラスを定義し、そのオブジェクトを生成して返すようにこのメソッドを定義します。

6.2.3.4. `create_reader_process()` メソッド

Sensor オブジェクトの計測開始に伴い使用される Reader プロセスを生成します。
具体的には `logger.core.LoggerProcess` に `logger.measure.reader_job` 関数を指定して生成します。

Reader プロセス上で実行される関数を拡張したい場合に、拡張版のジョブ関数を定義し、その関数を指定して Reader プロセスを生成して返すようにこのメソッドを定義します。

6.2.3.5. create_writer_process() メソッド

Sensor オブジェクトの計測開始に伴い使用される Writer プロセスを生成します。
具体的には logger.core.LoggerProcess に logger.measure.writer_job 関数を指定して生成します。

Writer プロセス上で実行される関数を拡張したい場合に、拡張版のジョブ関数を定義し、その関数を指定して Writer プロセスを生成して返すようにこのメソッドを定義します。

6.3. VC 判定 振動計測システム　カスタマイズポイント

本アプリでは、上述のようなラズパイロガーの拡張メカニズムを活用し、以下の拡張を行うことで VC 判定ライブラリを使用した計測を実現しています。

6.3.1. vc_calc_app.VcConfig クラス

logger.core.Config クラスを継承し、本アプリで追加した設定項目を保持するデータクラスです。

6.3.2. vc_calc_app.VcConfigurator クラス

logger.core.Configurator クラスを継承し、本アプリで追加した設定項目を設定ファイルから読み取り、VcConfig オブジェクトを生成して返します。

6.3.3. vc_calc_app.VcWriterArgs クラス

logger.measure.WriterArgs クラスを継承し、計測データに対する出力処理の一環として VC 判定ライブラリを使った演算を行うよう拡張しています。

WriterArgs オブジェクトは Writer プロセスに対して Sensor が生成するオブジェクトで、Sensor ごとの固有パラメータを保持し、出力処理のサポートを行う役目を果たします。

WriterArgs オブジェクトを拡張することで、出力処理の一環として VC 判定演算を実現しています。

6.3.4. vc_calc_app.VcGPIO クラス

VC 判定結果に対する GPIO 制御を行う目的で作成されたクラスです。

VcWriterArgs オブジェクトがその処理の過程で使用します。

6.3.5. vc_calc_app.VcA352 クラス

logger.measure.A352 クラスを継承し、出力処理の一環として VC 判定演算を行うよう VcWriterArgs オブジェクトを生成しています。

それを実現するにあたり、以下のメソッドを拡張しています。

6.3.5.1. to_writer_args() メソッド

logger.measure.Sensor クラスに定義されたメソッドで、WriterArgs オブジェクトを生成します。

VcA352 では VcWriterArgs を生成して返します。

6.3.6. vc_calc_app.VcLoggerFactory クラス

logger.core.LoggerFactoryImpl クラスを継承し、以下のメソッドを定義しています。

6.3.6.1. create_configurator() メソッド

VcConfigurator オブジェクトを生成して返します。

6.3.6.2. create_A352() メソッド

VcA352 オブジェクトを生成して返します。

6.3.6.3. create_A342() メソッド

本アプリでは A342 系センサーは対象外のため、A342 センサーが見つかった場合に呼び出されるこのメソッドで、対象外であることを示す例外を発生させています。

6.3.7. vc_calc_app.__main__ モジュール

src/vc_calc_app/__main__.py ファイルでは vc_calc_app パッケージの __main__ モジュールを定義しています。

ここでは VcLoggerFactory を生成し、logger.main 関数を呼び出しています。

```
import sys
from logger import main
from .factory import VcLoggerFactory

if __name__ == "__main__":
    code = main(factory=VcLoggerFactory())
    sys.exit(code)
```

図 6-3 __main__ モジュール

7. お問い合わせ

セイコーエプソン株式会社

営業本部 MD 営業部

インターネットによるお問い合わせ先

https://www.epson.jp/prod/sensing_system/contact/