組込機器にて最新情報技術を活用可能とするプラットフォームの実現

Creation of a platform enabling the use of the latest information technology in embedded devices

小森谷　良明　Yoshiaki Komoriya
中林　想太　Souta Nakabayashi
山田　功　Isao Yamada

キーワード
savic-net G5，ビル空調システム，快適性，省エネルギー，制御アプリケーション，人工知能，Python

ビル空調システムでは近年，さらなる快適性と省エネルギーを実現するために，人工知能等の最新情報技術の活用が求められている。ビル空調システムを構成する組込機器に制御アプリケーションを追加する場合，制御専用言語で制御アプリケーションを製作する必要があったため，最新情報技術の活用が困難であった。そこで，汎用プログラミング言語やライブラリで製作した制御アプリケーションを，組込機器に追加可能とするプラットフォームを実現した。これによりPCをシステムに追加しなくても，既にある組込機器で最新情報技術が活用可能となる。

In recent years, in order to realize further comfort and energy savings from building air conditioning systems, the need for artificial intelligence and other advanced information technology has become apparent. However, a new control application for the embedded devices incorporated into a building system usually must be created using the already-implemented control application language, making it difficult to utilize the latest information technology. To solve this problem we created a platform that allows control applications written with general-purpose programming languages and libraries to be added to embedded devices. Existing embedded devices can now take advantage of the latest information technology without the need to add a PC to the system.

一般論文

1. はじめに

ビル内には熱源設備や空調設備など様々な設備がある。ビル空調システムでは一般的なPCより信頼性が高く，省電力でスペース効率にも優れたコントローラと呼ばれる組込機器が，各種設備の機器を連携して制御することで，ビル全体の省エネルギーや快適な空調を実現している。

アズビルではビル空調システムにて，様々な標準制御機能を提供している。また，顧客ごとの要求に合わせた制御アプリケーションを別途製作して，組込機器に追加することも可能としている。

一般的に組込機器に追加する制御アプリケーションは，汎用のプログラミング言語（Java，JavaScript，Python，C#, Go, R等）ではなく，IEC 61131-3という国際規格で定義されているような，組込機器用の制御専用言語で製作される[(1)]。アズビルの従来ビル空調システム（savic-net FX™等）においても，制御アプリケーションの製作は制御専用言語で行う。制御専用言語はPCと比べてCPU性能やメモリ容量の制約が厳しい組込機器向けに設計されているため，組込機器での実行に適した制御アプリケーションの製作が可能である。

一方で近年，様々な分野にて人工知能等の最新情報技術の活用が試みられているが，これらの制御専用言語は汎用プログラミング言語と比べて機能や自由度が低いため，高度な科学計算や複雑なロジックの実装が困難である。

また，利用者が多い汎用プログラミング言語では，最新情報技術を活用するための機能改善やライブラリ開発が絶え間なく行われているが，制御専用言語においては，これらが活発に行われているとは言えない。

このように，制御専用言語は最新情報技術を活用するのが困難であり，別途，汎用プログラミング言語で製作した制御アプリケーションを実行するためのPCを，システムに追加する等の対応が必要であった。

この課題を解決するために，汎用プログラミング言語で製作した制御アプリケーションを組込機器に追加可能とし，かつ，既に動作している制御アプリケーションの機能を損なわずに，最新の言語バージョンや最新ライブラリを活用する新たな制御アプリケーションを追加可能とするプラットフォームを実現した。

これにより，ビル空調システムに新たなPCを追加するコストなしに，最新情報技術を活用する制御アプリケーションを，既にある組込機器に追加することが可能となる。

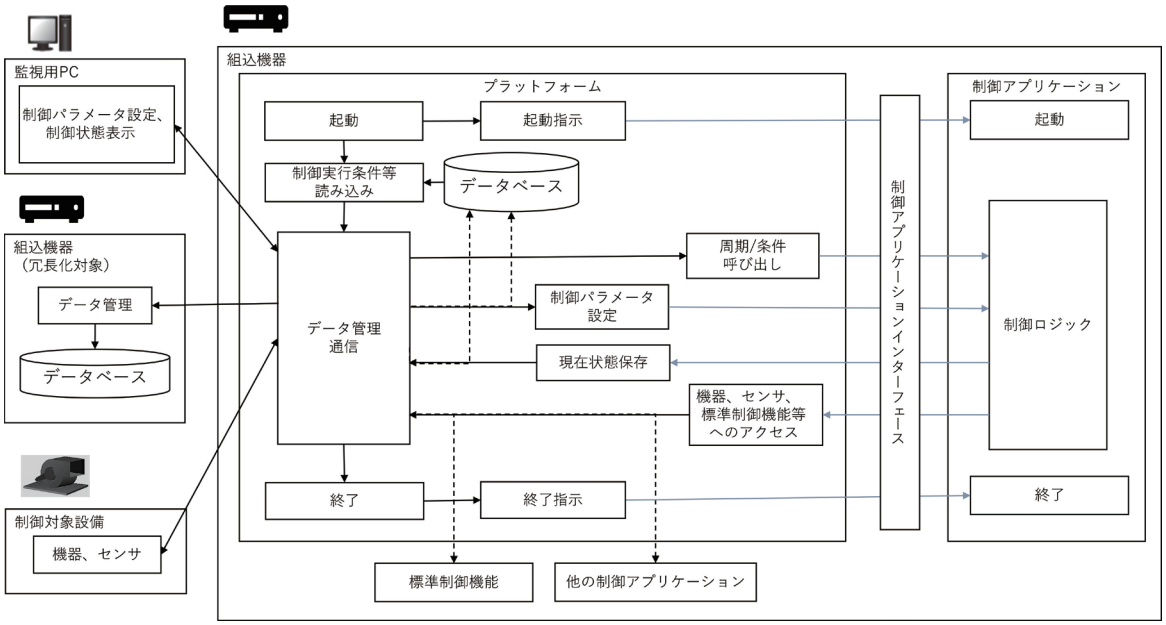

図1　プラットフォーム概要

2.汎用プログラミング言語による制御アプリケーションの実現

2.1　プラットフォームの概要

本プラットフォームの概要を図1に示す。本プラットフォームと制御アプリケーションは互いに独立したプロセスとして動作する。プラットフォームは機器の電源ONに連動して起動し，各制御アプリケーションに起動を指示する。その後，制御実行条件等を読み込んで，指定された周期，または指定された条件に合致したタイミングで，各制御アプリケーションに実装された制御ロジックを呼び出す。機器電源OFF時には，各制御アプリケーションに終了を指示する。

プラットフォームは，監視用PC上の画面から設定された各制御アプリケーションの制御パラメータと，各制御アプリケーションの現在状態をデータベースにて管理する。組込機器が冗長化されている場合には，プラットフォームが冗長化対象の組込機器上のデータベースとリアルタイムで同期をとる。これにより不測の故障が発生して各制御アプリケーションに対する終了指示が行えなかった場合でも，冗長化対象の組込機器の各制御アプリケーションにて故障前の制御パラメータと現在状態を引き継いで，制御を継続することを可能としている。

また，プラットフォームは制御対象設備の機器やセンサと通信して，制御アプリケーションが機器の状態やセンサの計測値にアクセスする機能を提供する。これにより制御アプリケーションは機器やセンサの所在，アクセスに使用する通信プロトコル（BACnet，Modbus等）の違いを意識せずに，機器やセンサの状態や計測値を読み書きすることを可能とした。また，プラットフォームは，制御アプリケーションが組込機器の標準制御機能や，既に動作している他の制御アプリケーションの，制御パラメータや現在状態にアクセスする機能も提供する。

プラットフォームはこれらの機能を，制御アプリケーションインターフェースを経由して各制御アプリケーションに提供する。このインターフェースはプログラミング言語非依存なプロトコル（WebSocketプロトコル）とメッセージ形式（JSON形式）から構成される。これにより特定のプログラミング言語だけでなく，様々な汎用プログラミング言語による制御アプリケーション開発に対応可能としている。

2.2　制御アプリケーション実行環境の仮想化

汎用プログラミング言語の多くは絶え間なく機能改善され，新バージョンの言語やライブラリが次々と登場する。この際，必ずしもバージョン互換性が保たれるとは限らない。このため，新バージョンの言語やライブラリで製作した制御アプリケーションを組込機器に追加する際に機器OSに組み込んだ言語やライブラリのバージョンを上げると，すでに動作している古いバージョンで製作した制御アプリケーションが動作しなくなる場合がある。

そこで，本プラットフォームでは，あとから新しいバージョンの言語やライブラリで製作したアプリケーションを追加しても，既に動作している制御アプリケーションに影響を与えないために，仮想化技術を組込機器に適用した。具体的には，仮想化技術により制御アプリケーションの実行環境を隔離し，制御アプリケーション本体や使用する言語とライブラリを機器OSに組み込むのではなく，機器OSから隔離した制御アプリケーション実行環境に追加する（図2）。この仕組みにより，追加する制御アプリケーションは，常に最新バージョンの言語とライブラリで製作することが可能となる。

2.3　メモリ使用量の削減

組込機器はサーバやPC等と比べて，CPU性能の強化やメモリ容量の拡大が困難である。また，汎用プログラミング言語は制御専用言語と比べて機能や自由度が高いため，汎用プログラミング言語で作成した制御アプリケーションは制御専用言語で作成したそれと比べて，メモリ使用量が多くなる傾向がある。

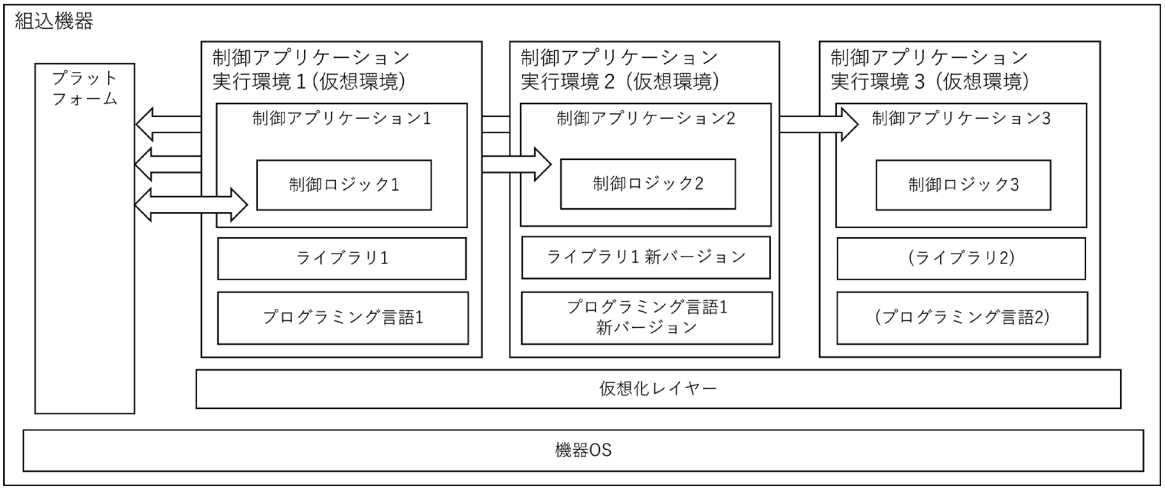

図2　制御アプリケーション実行環境の仮想化

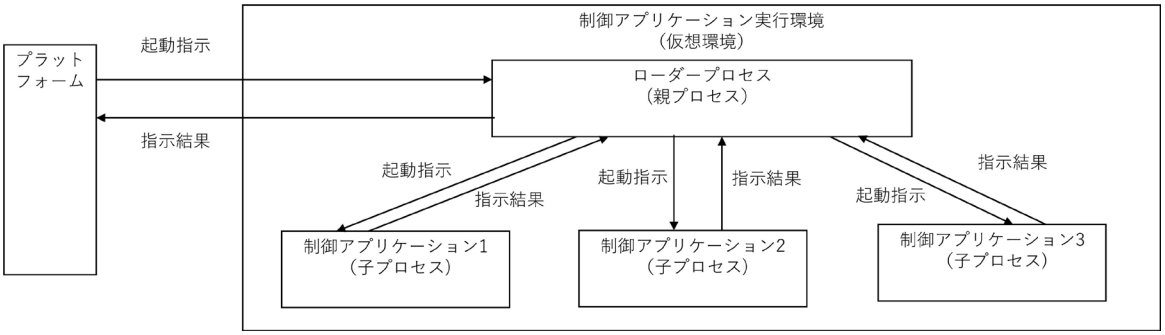

図3　制御アプリケーション起動の仕組み

CPU性能やメモリ容量の制約が厳しい組込機器にて仮想化を実現するために，ホスト型やハイパーバイザー型の仮想化技術と比べて必要なCPU性能やメモリ容量が少ない，コンテナ型仮想化技術[(2)]を採用した。

コンテナ型仮想化技術を採用する場合，保守や管理の容易性確保の観点から，1つのアプリケーション実行環境につき1つのアプリケーションを実行するのが望ましいとされている。しかし，厳密にこの方式を採用すると，各制御アプリケーション実行環境固有のメモリ空間が実行する制御アプリケーションの数だけ必要となるため，多くのメモリが必要になる。そのため，メモリ制約の厳しい組込機器では，動作可能な制御アプリケーションの数が少なくなる。そこで，本プラットフォームでは，同じ言語のバージョンとライブラリバージョンで製作した制御アプリケーションは，同じ制御アプリケーション実行環境にて実行する方式とした。

さらにメモリ使用量を削減するために，同じ実行環境で実行する複数制御アプリケーションを，あらかじめ必要なライブラリをインポートしたローダープロセスから，子プロセスとして起動する仕組みを開発した(図3)。

通常，ライブラリは各制御アプリケーション固有の独立したメモリ空間上に読み込まれる。しかし，今回開発した仕組みを用いることで，ローダーがライブラリを読み込んだメモリ空間を各制御アプリケーションで共有できる。これにより，一例としてPythonで製作した制御アプリケーションを30アプリケーション起動した場合で，メモリ使用量を約50%削減できることを確認した(図4)。

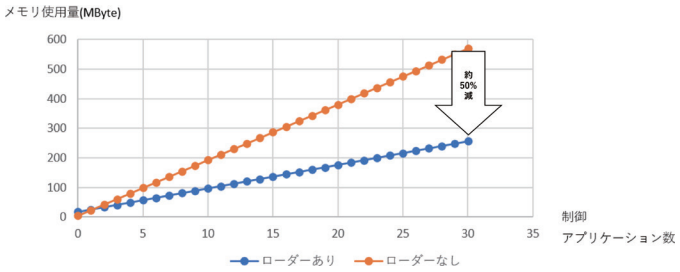

図4　メモリ使用量比較

3. 技術の製品化

3.1　汎用プログラミング言語Pythonの採用

2020年12月より，アズビルの最新ビル空調システムであるsavic-net™ G5にて，本プラットフォームによりPythonで製作した制御アプリケーションを追加可能とした組込機器である，savic-net™ G5 統合コントローラ[(3)]を販売開始した(図5)。

図5　savic-net™ G5 統合コントローラ

Pythonは習得が容易でコーディングの生産性が高く，高度な科学計算や複雑なロジックの実装も可能であることから，人工知能活用分野をはじめとする様々な分野で広く用いられている汎用プログラミング言語である。Pythonでの制御アプリケーション製作を可能としたことにより，複雑な科学計算を効率的に行うためのPythonライブラリや，機械学習を行うためのPythonライブラリを制御アプリケーションにて使用できる。加えて充実したコード編集機能やデバック機能を備える市販のPython用開発環境を，制御アプリケーションの製作に使用することも可能となった。

3.2　制御アプリケーション製作支援ツール

Pythonで制御アプリケーションを製作可能としたことに加えて，その製作をより簡便にするための，製作支援ツールを用意した(図6)。

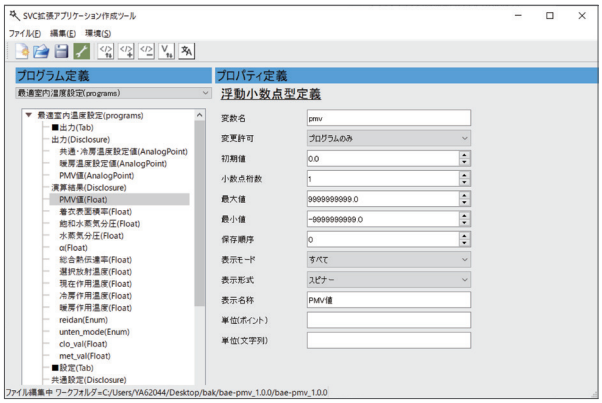

図6　制御アプリケーション製作支援ツール

制御アプリケーション製作者はこのツールに，製作する制御アプリケーションの名称や制御実行条件等と，制御パラメータや現在状態の値を格納するための変数のデータ名やデータ型等を入力する。ツールは入力された内容をもとに，制御実行時等にプラットフォームから呼び出される関数や，データベースのテーブルに格納するデータ構造と変数の定義を，Pythonコードのソースファイル等に自動出力する。加えて，ツールは入力された内容をもとに，制御アプリケーションの利用者が制御パラメータの設定と現在状態の確認を行うための画面[4]を自動生成する。

この制御アプリケーション製作支援ツールにより，アプリケーション製作者がデータ構造の定義や画面等を手作業で作成することを不要とし，ツールが出力したソースファイルの関数に制御ロジックを追記するだけで，制御アプリケーションを製作することが可能となった。

4. 制御アプリケーション開発事例

4.1　最適室温設定アプリケーション

本節では，本プラットフォームにて実現した制御アプリケーションの例として，最適室温設定アプリケーションを紹介する(図7)。

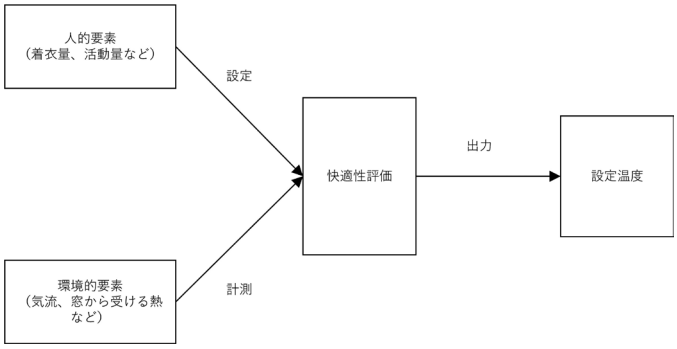

図7　最適室温設定アプリケーション

通常の空調制御では，あらかじめ人が室温の設定値を決めて設定するが，本アプリケーションは，室内の人の着衣量や活動量等の人的要素と室内の気流や窓から受ける熱等の環境的要素を元に，室内の人が快適と感じる設定温度を動的に自動演算する。これにより，室温の冷やし過ぎや暖め過ぎを自動的に防ぎ，快適性と省エネルギーの両立を実現する。

環境的要素として利用可能な計測値は，制御を実施する室内のセンサ設置状況等により異なる。また，快適性を評価し設定温度を決める手法にも，高度な科学計算を使用する等の，無数のバリエーションが考えられる。このような制御であっても，本プラットフォームにてPythonで制御アプリケーションを製作することで，顧客ごとの高度な要求に柔軟に対応可能となった。

4.2　人工知能制御実証実験での利用

人工知能を活用する試みとして，制御対象設備から取得した過去データを使用して学習を行い，得られた学習済みモデルを用いて制御を行うアプリケーションの試作も行っている(図8)。

本プラットフォームと製作支援ツールにより，制御アプリケーション製作者は制御ロジックの検討と実装に注力できるとともに，Pythonのライブラリを活用して効率よく制御アプリケーションを製作し，制御ロジックの効果を検証することが可能となった。

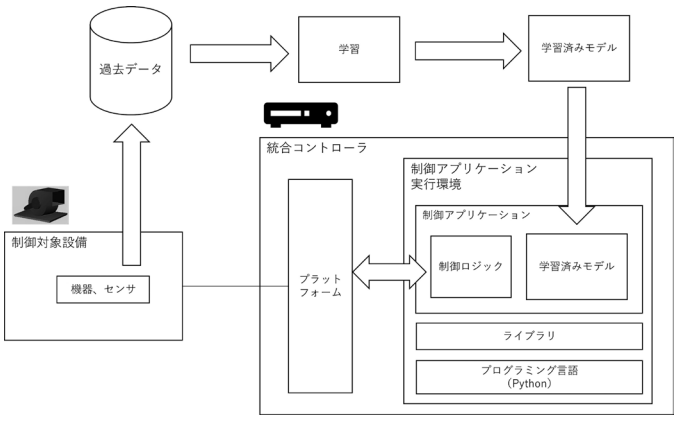

図8　人工知能分野での利用例

5. おわりに

本稿で述べたプラットフォームにより，最新の汎用プログラミング言語や最新ライブラリで製作した制御アプリケーションを，組込機器に追加可能とすることができた。これによりPC等をシステムに追加しなくても，既にある組込機器で最新情報技術を活用可能となった。

加えて，Pythonの採用と制御アプリケーション製作支援ツールにより，簡便に制御アプリケーションが製作可能となった。

今後は，制御アプリケーションを社外の販売協力店等で製作するための体制構築と，本技術による人工知能制御実証実験等を推進する予定である。本技術により，ビル空調システムへの人工知能等の，最新情報技術の活用が加速することを期待している。

＜参考文献＞

(1) 日本電機工業会:
「PLCアプリケーションの開発効率化指針」, https://www.jema-net.or.jp/jema/data/plc_ver1.pdf

(2) 川口 直也 (2020):
「コンテナ型仮想化概論」, カットシステム

(3) 深浦 敦:
「高度なエネルギー管理と快適な空間を提供するsavic-net™ G5システムの統合コントローラ」, azbil TechnicaReview, 2019年4月発行号, pp17-21

(4) 西羅 大貴, 古賀 宏, 山本 博之:
「優れたユーザー体験を提供する新しいビルディングオートメーションシステムの開発」
azbil Technical Review, 2016年4月発行号, pp27-33

＜商標＞

savic-net, savic-net FXはアズビル株式会社の商標です。
Java，JavaScriptはオラクルおよびその関連会社の登録商標です。
PythonはPython Software Foundationの商標または登録商標です。
BACnetはASHRAEの商標です。
Modbus is a trademark and the property of Schneider Electric SE, its subsidiaries and affiliated companies.

＜著者所属＞

小森谷　良明　アズビル株式会社
ビルシステムカンパニー
開発本部開発1部

中林　想太　アズビル株式会社
ビルシステムカンパニー
BAエンジニアリング部

山田　功　アズビル株式会社
AIソリューション推進部