Vol.2020-EMB-55 No.3
2020/12/11

# IoT時代の仮想シミュレーション環境「箱庭」の実現に向けた検討および初期実装

高瀬 英希[1,2,a)]　細合 晋太郎[3]　高田 光隆[4]　庭野 正義[5]　辻 悠斗[6]　森 崇[6]

**概要**：IoT の新たな時代を迎え，開発現場では様々な機器が絡んだ複雑なシステムと向き合う必要がある．我々は，このようなシステムを効率的にシミュレーションできる環境である「箱庭」の研究開発の構想に取り組んでいる．本環境では，IoT システムの全体像を机上で仮想化できるようにして，多様な領域からの技術者が集まって構築される IoT システムの振る舞いや問題経路を，各自の開発対象や興味から多様に観察できるようにする．本稿では，これまで我々が検討を進めてきた「箱庭」の狙いや基本構想を紹介する．その後，これらを実証する応用事例として開発を進めているプロトタイプについて共有する．最後に，単体ロボットを題材としたプロトタイプの開発状況を紹介し，その技術的有意点を議論する．

# A study of virtual simulation environment in the novel IoT era “Hakoniwa” and its initial implementation

## 1. はじめに

IoT（Internet of Things）は，情報技術の総合格闘技である．多様かつ大量の計算機器がネットワークを介して密接に絡み合い，大規模かつ複雑な IoT システムが構築される．高品質な IoT システムを実現するためには，組込み系のみならず IT 系や通信インフラ系，ICT サービス系にクラウドサーバ系など，様々な分野の技術領域を結集させることが不可欠となる．

このような大規模な IoT システム開発について，自動運転車を活用した IoT 交通サービスを例として示す．**図 1** のように，本例では，メカやエレキ・ECU の自動車制御系はもちろんのこと，位置情報から配車管理を担うクラウド交通サービス系，さらには Web サービスのためのネットワーク系など，様々な分野の技術領域を横断していることがわかる．このような IoT システムを構築する際の課題としては，まず，各技術領域の成果物を結合する際のテストや検証が困難であることが挙げられる．全体結合しないと見えない問題が多数潜んでおり，様々な機器間の整合性を取る必要がある．さらに，問題発生時には，そもそもデバッグが困難であり，その経路や原因調査が著しく複雑になる．加えて，IoT システムの実用化には実証実験が不可欠であるが，その手間や時間，費用コストも非常に大きくなる．IoT サービスの構築にあたっても，どのモジュールをどう組み合わせて効果的な新しいサービスを創出できるかを評価することは，準備と手間の大きくなる工程となる．

我々は，このような IoT システム開発時および IoT サービス構築時の課題に対するアプローチとして，「箱庭」の研究開発に取り組んでいる．多様な分野からの技術者が集まって IoT システムを開発する際に，仮想環境上にソフトウェアやサービスを持ち寄って，机上で実証実験できる場を提供できるようにする．本取り組みは，組込みシステム構築の基盤となるソフトウェアやプラットフォームを良質なオープンソースソフトウェアとして開発・公開することで，組込みシステム技術と産業の振興を図ることを目的とした特定非営利活動法人である TOPPERS プロジェクト [1] のワーキンググループとして活動を推進している．

[1] 京都大学
Kyoto University
[2] JST さきがけ
JST PRESTO
[3] チェンジビジョン
Change Vision, Inc.
[4] 名古屋大学
Nagoya University
[5] アイコムシステック
ICOMSYSTECH Co.,Ltd.
[6] 永和システムマネジメント
ESM, Inc.
[a)] takase@i.kyoto-u.ac.jp

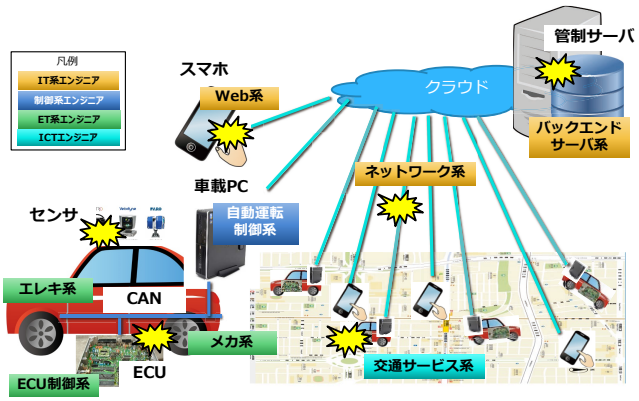


図 1　自動運転を活用した IoT 交通サービスの開発例とその課題

本稿では，大規模かつ複雑な IoT システムを効率的にシミュレーションできる環境である「箱庭」について，我々のこれまでの検討状況を報告する．まず 2 章では，箱庭の基本構想としてコンセプトおよび我々の狙いを概説する．次に 3 章では，箱庭のアーキテクチャを示し，これを形成する技術要素およびコア技術を紹介する．その後 4 章では，我々の構想を実証することを目的とした応用事例として開発を進めている 3 種類のプロトタイプモデルについて詳述する．さらに 5 章では，単体ロボット向けシミュレータについて詳説し，その核となる異種レイヤ間の時間管理機構を提案する．6 章にて関係する既存の取り組みに触れたのち，最後に 7 章にて本稿のまとめと今後の展望を示す．

## 2.　「箱庭」のコンセプトと狙い

箱庭[*1]のコンセプトは，「箱の中に，様々なモノを自分の好みに配置して，いろいろ試せる」ことである．IoT システムの開発や IoT サービスの構築に携わる全ての技術者のために，仮想的なシミュレーション環境とエコシステムを構築することを目指している．多様な分野からの技術者が集まって IoT システムを開発する際に，仮想環境である箱庭上にソフトウェアやサービスを持ち寄って，机上で実証実験できる場を提供する．

図 2 に，箱庭のコンセプトと想定する利用シーンを示す．本環境では，IoT システムの全体像を机上で仮想化できるようにして，多様な領域からの技術者が集まって構築される IoT システムの振る舞いや問題経路を，各自の開発対象や興味から多様に観察できるようにする．目指す強みと新しさとしては，IoT の各要素を連携させた複雑なシステムの事象や状態を，任意の精度で検証可能とすることである．箱の中では同じ挙動が再現されるシステム環境を，各技術者の分野や立場に応じた視点や抽象度で評価できるようにする．箱庭では，組込みマイコンやサーバに車両力学モデルや走行環境モデルなど，IoT システムを構成する

[*1] 「箱庭」の名称の語源は，名園や山水を模したミニチュアの庭園に由来している．盆景や盆栽に類するもので，江戸時代後半から明治時代にかけて流行したといわれている．

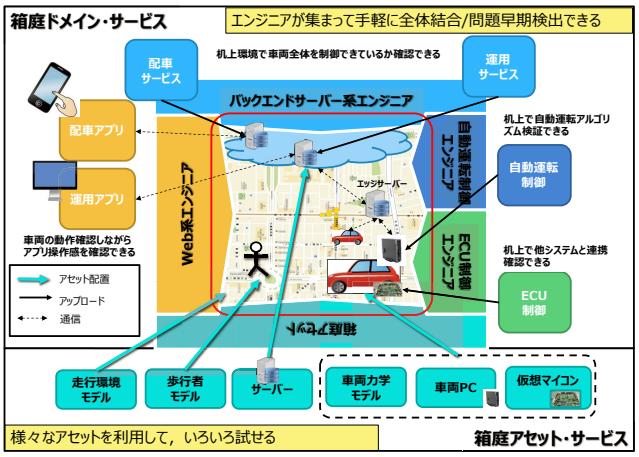


図 2　箱庭のコンセプトと想定する利用シーン

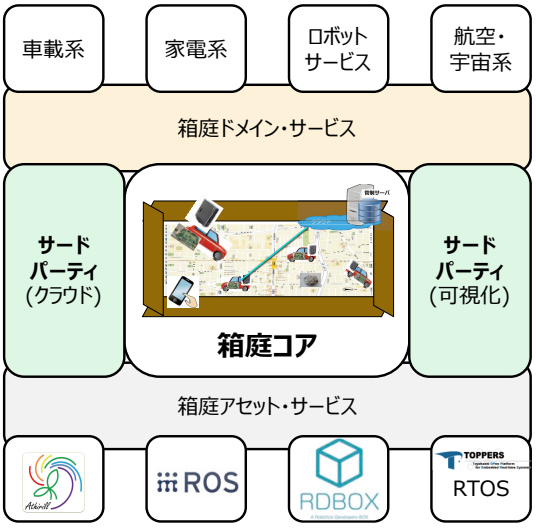


図 3　箱庭のアーキテクチャ

要素の名称を「アセット」と定義している．このアセットを差し替えることで，検証の対象や抽象度などを任意に設定できるようにする．

箱庭の利用者としては，IoT システム全体の開発者やそれを利用してサービスを展開する提供者だけでなく，IoT システムを構築するための構成要素の開発者および提供者を想定する．すなわち，構成要素を開発する技術者は，それに対応するアセットを提供することで，この自身の成果物を箱庭環境上で試行できるようになる．

## 3.　アーキテクチャ

図 3 に，現時点で検討している箱庭のアーキテクチャを示す．箱庭が対応するドメインサービスとしては，車載系や家電系，ロボットサービスにさらには航空・宇宙系など，様々な分野へ適応できることを目指している．アセット・サービスとして利用されるシミュレーション内の登場物は，3.1 節にて述べる既存のソフトウェアフレームワークを技術要素として活用している．アセットの拡充ならびに箱庭への活用に向けた機能拡張にも取り組んでいる．な

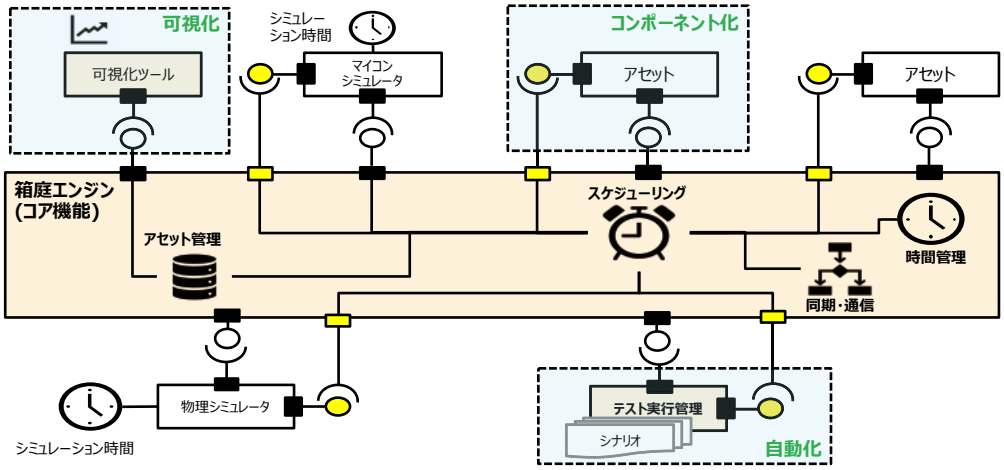


図 4　複雑な IoT システムを仮想環境で動作させる仕組み

お，特にクラウドサービスや可視化技術については，活用できる既存のサードパーティのソフトウェアフレームワークを積極的に利用する方針としている．3.2 節にて述べるように，仮想シミュレーションを実現するための共通かつ重要なコア機能の技術確立に重点的に取り組んでいる．

### 3.1　技術要素

箱庭のシミュレーション内の登場物として活用するアセット・サービスである技術要素について概説する．

**athrill**　主に車載向けの ECU やカーネルを対象とした命令レベルの CPU エミュレータである [2], [3]．仮想マイコン上で組込みプログラムの実行およびデバッグが試行でき，ベアメタルプログラムやリアルタイム OS 上で動作する組込み制御プログラムを評価できる．

**mROS**　箱庭では，デバイス間の通信機構として，ロボット向け開発プラットフォームである ROS（Robot Operating System）[4] に着目している．mROS は，ROS ノードおよびその出版購読通信機能を組込みデバイス上で実行可能とする軽量実行環境である．

**TOPPERS カーネル**　TOPPERS プロジェクトで開発が進んでいるオープンソースのリアルタイムカーネルである．高い信頼性・安全性・リアルタイム性を要求される組込みシステムを対象として，$\mu$ITRON 仕様の拡張・改良である ASP カーネルや保護機能を持つ HRP カーネル，車載制御用の AUTOSAR OS 仕様に準拠した ATK カーネルなどが公開されている．

**RDBOX**　ROS ロボットや IoT に最適化された，Kubernetes クラスタとセキュアで拡張性の高い Wi-Fi ネットワークを自動構築できるネットワークフレームワークである [7]．シミュレーション環境と現実の作業環境をブリッジすることを目的としている．

**Unity**　リアルタイム 3D 開発プラットフォームである．箱庭では，物理演算エンジンと空間可視化のためのサードパーティツールとして活用している．

これらの他にも，IoT 時代の仮想シミュレーション技術の確立に向けて，箱庭を形成するために統合すべき技術要素を模索しているところである．

### 3.2　箱庭コア機能

IoT システムのような大規模かつ複雑なシステムを仮想環境で動作させる仕組みを体系化するために，箱庭エンジンと名付けた共通かつ重要なコア機能の検討を進めている．箱庭の核となる機能として，次の 4 種類のコア機能を提案する．

**時間管理**　箱庭にプラグインされたアセットのシミュレーション時間を管理し，アセット間のシミュレーション時間を調歩する機能

**同期・通信**　アセット間でのデータの交換，および，イベント通知や処理待ちを実現

**スケジューリング**　アセットのそれぞれの実行タイミングを管理する機能

**アセット管理**　アセットのライフサイクルや情報出力，および，内部情報や入出力データを管理する機能

図 4 に示すように，箱庭コア機能が備えるべき重要な機能特性としては，コンポーネント化，可視化，イベント駆動化および自動化が挙げられる．

## 4.　プロトタイプモデルの開発

ここまで述べてきた箱庭のコンセプトと狙いを実証するための応用事例を示すことを目的として，現在，3 種類のプロトタイプモデルの開発に取り組んでいる．本取り組みを通して，箱庭エンジンとしてのコア機能の機能詳細検討を同時に進めている．

1 つめのプロトタイプモデルは，図 5 に示す単体ロボット向けシミュレータである．ET ソフトウェアデザインロボットコンテスト（ET ロボコン）[8] を題材としており，1

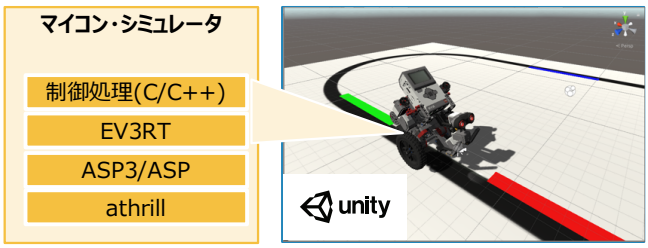


図 5　単体ロボット向けシミュレータ

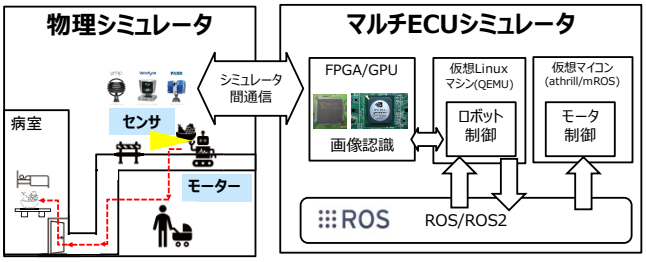


図 6　ROS・マルチ ECU 向けシミュレータ

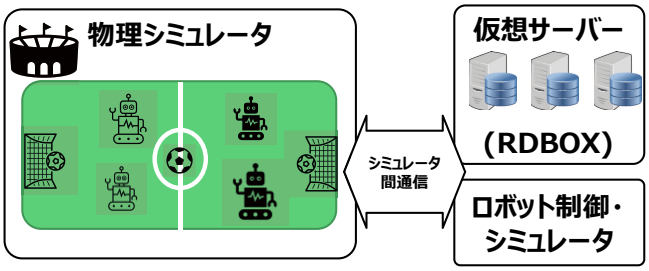


図 7　協調動作向けシミュレータ

つのマイコンで 1 台のロボット全体を制御するシステムのシミュレーションを実現する．本シミュレータの開発を通して．物理シミュレータとマイコンシミュレータ間の連携方法や，異なるシミュレータ間の時間管理の仕組みを確立する．詳細は 5 章で述べる．

2 つめは，図 **6** に示す ROS・マルチ ECU 向けシミュレータである．複数のマイコンが ROS 通信を介して 1 台のロボットを制御するシステムのシミュレーションを実現する．マイコンだけでなく FPGA や GPU 等の異種混載型のシステムへの適応も目指しており，マルチ ECU や FPGA ／ GPU 間の分散的な連携方法と時間管理の仕組み，箱庭アセット間のスケジューリング方式および同期・通信機能を検討している．また，アセット間における ROS 通信のデータや状態の可視化を実現することを目指している．

3 つめは，図 **7** に示す協調動作向けシミュレータである．ロボットなどの複数のシステムが協調的に動作するシステムのシミュレーションを実現する．本シミュレータの開発を通して，クラウドサービスとの連携による包括的な IoT システムの仮想化の実現や，システム間の協調動作の連携方法を検討する．これと合わせて，箱庭アセットを充実させるための仕組みを構築していく．

## 5.　単体ロボット向けシミュレータ

本章では，ET ロボコンを題材として開発しているプロトタイプモデルである単体ロボット向けシミュレータについて詳説する．本シミュレータの構成と狙いを示し，この核をなす箱庭コア技術である異種レイヤ間の時間管理機構を提案する．その後，本シミュレータの想定される展開事例を提示し，提案手法の技術的有意点を議論する．

### 5.1　プロトタイプモデルの構成

単体ロボット向けシミュレータは，図 5 に示すように，ET ロボコン大会の規定に基づく HackEV を仮想化の対象とした．教育版 EV3 インテリジェントブロック内の組込みマイコンの仮想化については athrill を，ロボット全体と外部環境の可視化および物理演算については Unity をそれぞれ用いている．ソフトウェアプラットフォームについては TOPPERS/EV3RT（Real-Time platform for EV3）[9] を採用している．ただし，athrill が現状で主に対応するマイコンが V850 の命令セットであるため，TOPPERS/EV3RT が標準サポートする HRP3 カーネルではなく ASP3 カーネルに代替して実現している．ロボットの制御プログラムは C/C++で記述することができる．

本プロトタイプによって，Unity 上の 1 台のロボットおよび athrill によって仮想化された組込みマイコンとその制御プログラムを連携させたシミュレーションが実現可能となる．構成要素としては，ロボット本体についてはモータやセンサ，制御用のマイコンについてはリアルタイムカーネルおよびロボット制御用のプログラム，走行コースや障害物などの外部環境からなる．本体のセンサについては，HackEV に取り付けられているカラーセンサ，超音波センサおよびジャイロセンサの値を取得できる．また，制御用プログラムのデバッグ容易化のために，インテリジェントブロックの LCD 上におけるシリアルメッセージ表示の仮想化も実現している．なお，Unity エディタ上での操作によって，走行コースや障害物を任意に設定したり，さらにはロボット本体を変更することも可能である．

### 5.2　時間管理機構

箱庭は，構成要素として様々なシミュレータが混在する環境となる．各シミュレータはそれぞれ固有のシミュレーション時間を持つため，独立して動作するとそれぞれの実行タイミングにずれが生じる．実際に単体ロボット向けシミュレータでは，組込みマイコンシミュレータの athrill とロボット本体および外部環境を仮想化する物理シミュレータの Unity が同時に稼働されることとなる．それぞれのデータのセンシング周期およびシミュレーションのための計算時間は同一ではない．また，同種のシミュレータで

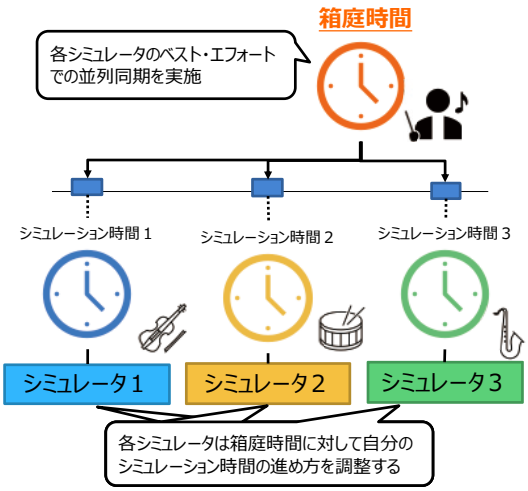


図 8　異種レイヤ間の時間管理機構

あっても，実行する計算機の負荷や環境によってはシミュレーションのための計算時間に変動が生じうる．そこで本稿では，異種レイヤ間での時間を同期させる方式を提案する．

提案する時間管理機構の方式を図 8 に示す．仕組みとしては，箱庭時間と呼ぶ時刻をコア機能として用意する．各シミュレータは，この箱庭時間を適切なタイミングで参照し，自身のシミュレーション時間の進め方を調整するようにする．つまり，箱庭時間より早い場合はシミュレーション時間を遅くし，箱庭時間より遅い場合は，シミュレーション時間を早く進めるようにする．加えて，いずれかのシミュレータを一時停止する場合や，処理負荷が高くなりシミュレータの時間が遅れる場合には，箱庭時間の進度を調整する仕組みも用意している．この時間管理機構の同期の仕組みによって，あるシミュレータ上でのプログラムのデバッグのためにブレークポイントに到達する時には，連携する他のシミュレータも一時停止してその時点の挙動を観察することができる．

箱庭時間を各シミュレータで共有する通信方式としては，UDP および MMAP による 2 種類の方式がある．UDP パケットで箱庭時間を共有する方式では，各シミュレータを別々の計算機に配置することができ，箱庭の処理全体を負荷分散することが可能となる．ただし，ネットワーク通信を伴うため，箱庭時間を共有するための処理にかかる遅延が発生しうる．MMAP ファイルで箱庭時間を共有する方式では，遅延は発生しないが，時間管理機構および各シミュレータを同一の計算機に配置する必要がある．

表 1 および表 2 に，単体ロボット向けシミュレータにおける athrill と Unity の通信データ仕様の一部を示す．表 1 は athrill から Unity への制御データを，表 2 は Unity から athrill へのセンシングデータの情報をそれぞれ表している．それぞれのデータは 1,024 バイトのサイズ長で規定している．先頭から 32 バイトまでのフィールドにヘッダ情報やバージョン番号，拡張領域のオフセットおよびサイズ，さらに時間同期のための情報が含まれている．続く 480 バイトのフィールドには，カラーセンサやモータなど HackEV および EV3RT における基本機能のデータが保持される．後半の 512 バイトは，仮想化の対象や機能に応じて構成可能なフィールドとして確保している．

時間管理のための情報は，計 64 ビットのマイクロ秒単位で管理している．本プロトタイプモデルでは 2 種類のシミュレータのみで構成されるため，短周期でのセンシングが求められより処理負荷が重くなる Unity 側に箱庭時間を配置し，athrill にセンシングデータとともに通知してそれぞれのシミュレーション時間を同期するようにしている．

本手法は中央制御的ではなく分散制御方式となるため，複数シミュレータから構成される箱庭の仮想化処理にかかる計算機上での並列化が容易となる．また，各シミュレータ時間を可視化して時間同期の程度を定量化することができるため，シミュレータの動作環境やスペックの妥当性を評価して調整することも可能となる．

### 5.3　展開事例

単体ロボット向けシミュレータの展開事例としては，オンラインでのロボット教育での採用を想定している．図 9 に，展開事例として想定する演習課題のデモを示す．このデモでは，フロアの色や障害物までの距離を各種センサで検出し，ゴールまでロボットの動作を制御することを演習課題としている．また，ロボットの内部状態をインテ

表 1　athrill から Unity への通信データの仕様（一部）

| 名称 | オフセット | サイズ | 値の単位表現 |
|---|---|---|---|
| ヘッダ情報 | 0 | 4 | ASCII コード |
| バージョン番号 | 4 | 4 | 整数値 |
| 時間同期用情報 [usec] | 8 | 4 | 下位 32 ビット |
| (athrill 側の時間) | 12 | 4 | 上位 32 ビット |
| 時間同期用情報 [usec] | 16 | 4 | 下位 32 ビット |
| (Unity 側の時間) | 20 | 4 | 上位 32 ビット |
| 拡張領域オフセット | 24 | 4 | 整数値 |
| 拡張領域サイズ | 28 | 4 | 整数値 |
| (HackEV 基本機能) | 32 | 480 | — |
| (HackEV 拡張機能) | 512 | 512 | — |

表 2　Unity から athrill への通信データの仕様（一部）

| 名称 | オフセット | サイズ | 値の単位表現 |
|---|---|---|---|
| ヘッダ情報 | 0 | 4 | ASCII コード |
| バージョン番号 | 4 | 4 | 整数値 |
| (予約領域) | 8 | 8 | — |
| 時間同期用情報 [usec] | 16 | 4 | 下位 32 ビット |
| (Unity 側の時間) | 20 | 4 | 上位 32 ビット |
| 拡張領域オフセット | 24 | 4 | 整数値 |
| 拡張領域サイズ | 28 | 4 | 整数値 |
| (HackEV 基本機能) | 32 | 480 | — |
| (HackEV 拡張機能) | 512 | 512 | — |

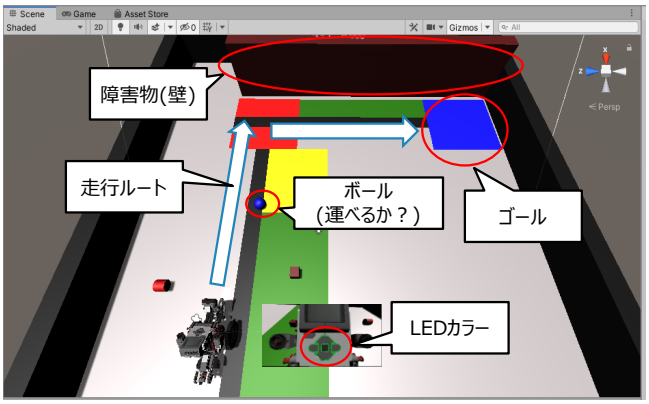


図 9　オンラインでのロボット教育に想定される展開事例

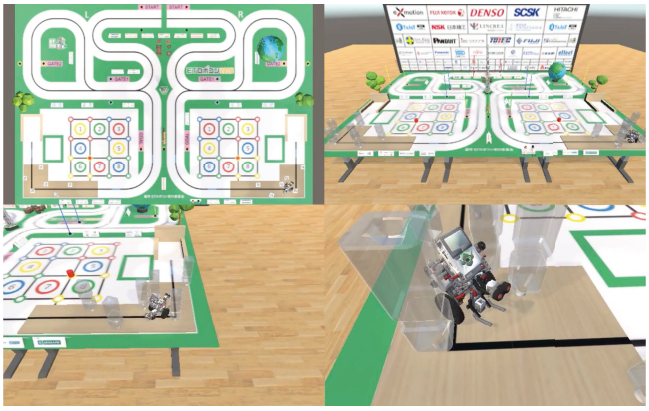

図 10　ET ロボコン 2020 大会での展開事例 [8]
（提供：ET ロボコン 2020 実行委員会）

リジェントブロック上の LED の発光色で変化させたり，HackEV 前方のアームを用いて地面に落ちているボールを運ぶといった発展課題を用意することができる．

昨今の新型コロナウイルス感染症を取り巻く情勢により，組込みシステムやロボット制御に関する教育実習のオンライン化の模索が続いている．我々が開発した成果の一部は，図 **10** のように 2020 年度の ET ロボコン大会において実際に採用されることとなった．また，大学における演習科目でも，箱庭の技術によって構築された仮想シミュレーション環境を採用する動きがあり，いくつか進行中である．今後，このような取り組みでの教育効果および箱庭の有用性について評価ならびに議論を進めていく予定である．

## 6. 既存の取り組み

箱庭に関係する既存の取り組みについて触れる．

欧州の Modelica Association が中心となって策定を進めている規格として，FMI（Functional Mockup Interface）[10], [11] がある．本規格は，様々なベンダが提供するツールの接続や統合をするためのオープンな標準インタフェース仕様である．FMU（Functional Mockup Unit）と呼ばれるプラントモデルを単位として，シミュレータ等の異種ツール間で交換したり接続することができる．これは，アセット管理の仕組みとして，箱庭と目指す方向性が近い．相互接続モデルについては，FMU 内に数値計算ソルバを内包しない Model Exchange および Co-Simulation の 2 種類の方法が提供されている．シミュレーション時間の同期方法についても検討されており，前者については完全な時間同期が可能であり，後者の場合はシミュレータ間の許容される遅延時間をユーザが設定する．ただし，いずれも中央制御方式であるため，システムの構成要素が増えると時間管理の処理オーバヘッドが大きくなる可能性が考えられる．

The Eclipse Foundation のワーキンググループとして活動が推進されているプロジェクトとして，自動運転システムの開発を加速することを目的とした開発環境である OpenADx[12] がある．オープンソースツールを用いて，アーキテクチャ設計からインテグレーションの製造，試験などの開発プロセス全般をカバーしており，各開発プロセスにおいてシミュレーションができるようになっている．このため，技術者の任意の視点や抽象度でシステム環境を評価できることを目指している箱庭の参考になるものと考えている．

3DCoAutoSim[13] は，Unity をベースとした高品質の 3D 視覚化を備えた車両シミュレータであり，様々な運転環境をエミュレートすることができる．交通モデルに基づいて他の車両の移動行動を視覚化し，Unity と交通流シミュレータである SUMO との通信を実現している．すなわち，IoT サービスとして交通流のシミュレーションに特化した仮想化環境であるといえる．

## 7. おわりに

本稿では，IoT 時代の新たな仮想シミュレーション環境である「箱庭」について紹介し，そのコア機能のひとつである時間管理機構について提案した．本活動を通して得られる我々の研究開発の成果は，すべてオープンソースとして公開予定である．また，本活動の Web サイト（`https://toppers.github.io/hakoniwa`）を立ち上げ，最新動向を積極的に情報発信している．今後は，4 章で示した箱庭プロトタイプモデルの開発を推進していき，さらに 3.2 節で示した箱庭エンジンとしてのコア機能の成熟および様々な機能拡充を進めていく予定である．

**謝辞**　本稿で取り上げた箱庭プロトタイプモデルの 1 つである単体ロボット向けシミュレータにおいて，走行体ロボットである HackEV の Unity アセットは，ET ロボコン実行委員会より提供いただいたデータを基に作成しています．また，athrill と Unity の時間管理機構に関しては，その仕様について実行委員会の皆さまより多くの助言や技術的な協力をいただきました．ET ロボコン実行委員会の皆さまに深く感謝いたします．なお，TOPPERS プロジェクト箱庭 WG より公開している HackEV のアセットは，ET

ロボコン実行委員会より提供されている本番環境とは異なりますのでご注意ください．また，本アセットは，個人利用または教育利用に限定してご利用ください．

Unity パッケージの設計と作成にあたっては，宝塚大学東京メディア芸術学部の吉岡章夫准教授および学部生の杉﨑涼志さん，木村明美さん，千葉純平さんにご協力いただきました．ここに深く感謝いたします．

## 参考文献

[1] TOPPERS プロジェクト (online), `https://www.toppers.jp/` (2020.11.20).
[2] TOPPERS プロジェクト：Athrill (online), `https://www.toppers.jp/athrill.html` (2020.11.20).
[3] TOPPERS プロジェクト：第 7 回 TOPPERS 活用アイデア・アプリケーション開発コンテスト 活用アイデア部門金賞「athrill(アスリル)」 (online), 入手先 ⟨https://www.toppers.jp/docs/contest/2017/athrill.pdf⟩ (2020.11.20).
[4] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A. Y.: ROS: an open-source Robot Operating System, *Proc. of the IEEE Int'l Conf. on Robotics and Automation Workshop on Open Source Robotics*, Vol. 3, No. 3.2 (2009).
[5] Takase, H., Mori, T., Takagi, K. and Takagi, N.: mROS: A Lightweight Runtime Environment of ROS 1 nodes for Embedded Devices, *Journal of Information Processing*, Vol. 28，pp. 150–160 (2020).
[6] Yugen, H., Takase, H., Mori, T., Takagi, K. and Takagi, N.: A functionality expansion of the lightweight runtime environment mROS for the user defined message types, *Proc. of Asia Pacific Conference on Robot IoT System Development and Platform (APRIS)*, pp. 1-7 (2019).
[7] 株式会社インテック：<Open Source> RDBOX –アールディーボックス– | コンテナ型仮想化で ROS ロボット開発をサポート (online), `https://rdbox-intec.github.io/homepage_jp/` (2020.11.20).
[8] ET ロボコン実行委員会：ET ロボコン | 組込みソフトウェア技術教育をテーマとした「ET ロボコン」 (online), `https://www.etrobo.jp/` (2020.11.20).
[9] Li, Y., Matsubara, Y. and Takada, H.: EV3RT: A Real-time Software Platform for LEGO Mindstorms EV3, *Computer Software*, Vol. 34, Issue 4, pp. 4_91–4_115 (2017).
[10] Modelica Association: Functional Mock-up Interface for Model Exchange and Co-Simulation (Version 2.0.1) (online), 入手先 ⟨https://fmi-standard.org/⟩ (2020.11.20).
[11] 自動車技術会 自動車制御とモデル部門委員会 FMI 活用・展開検討 WG：FMI 活用ガイド Ver.1.0.1 (online), 入手先 ⟨https://www.jsae.or.jp/katsudou/docu/1035/fmi_guide101.pdf⟩ (2020.11.20).
[12] The Eclipse Foundation: OpenADx |Eclipse openADx Working Group (online) `https://openadx.eclipse.org` (2020.11.20).
[13] Hussein, A., Alvarez, A., Armingol, M. J. and Olaverri-Monreal, C.: 3DCoAutoSim: Simulator for Cooperative ADAS and Automated Vehicles, *Proc. of 21st Int'l Conf. on Intelligent Transportation Systems* (2018).