「ROS」の基礎と ROS 2プログラミングの実践

6. 発展的な話題

高瀬 英希

(京都大学／JSTさきがけ)

プログラム・スケジュール

6. 発展的な話題 [day2 15:15-16:15]

- mROS: 組込み向けROS 1ノードの軽量実行環境
- ZytleBot: FPGA統合開発プラットフォーム
- 箱庭: IoT時代の仮想シミュレーション環境

組込みデバイス向け ROSノード軽量実行環境

- モチベーションと狙い
- 全体像・ソフトウェア構造とタスク構成
- 分散ロボットシステムへの適用事例

Motivation

ROS

- 豊富なOSS資産
- Linux/Ubuntuの駆動が必要

ROS 2

- マルチプラットフォーム対応
  （組込み環境を含む？）
- OSS資産はまだ潤沢ではない

- 組込み技術は省電力化とリアルタイム性向上に貢献できる
- ただし，ROS 1とROS 2の間に互換性が無い
  - APIレベルで異なる
  - プログラミングスタイルも異なる

Our Strategy

ROS

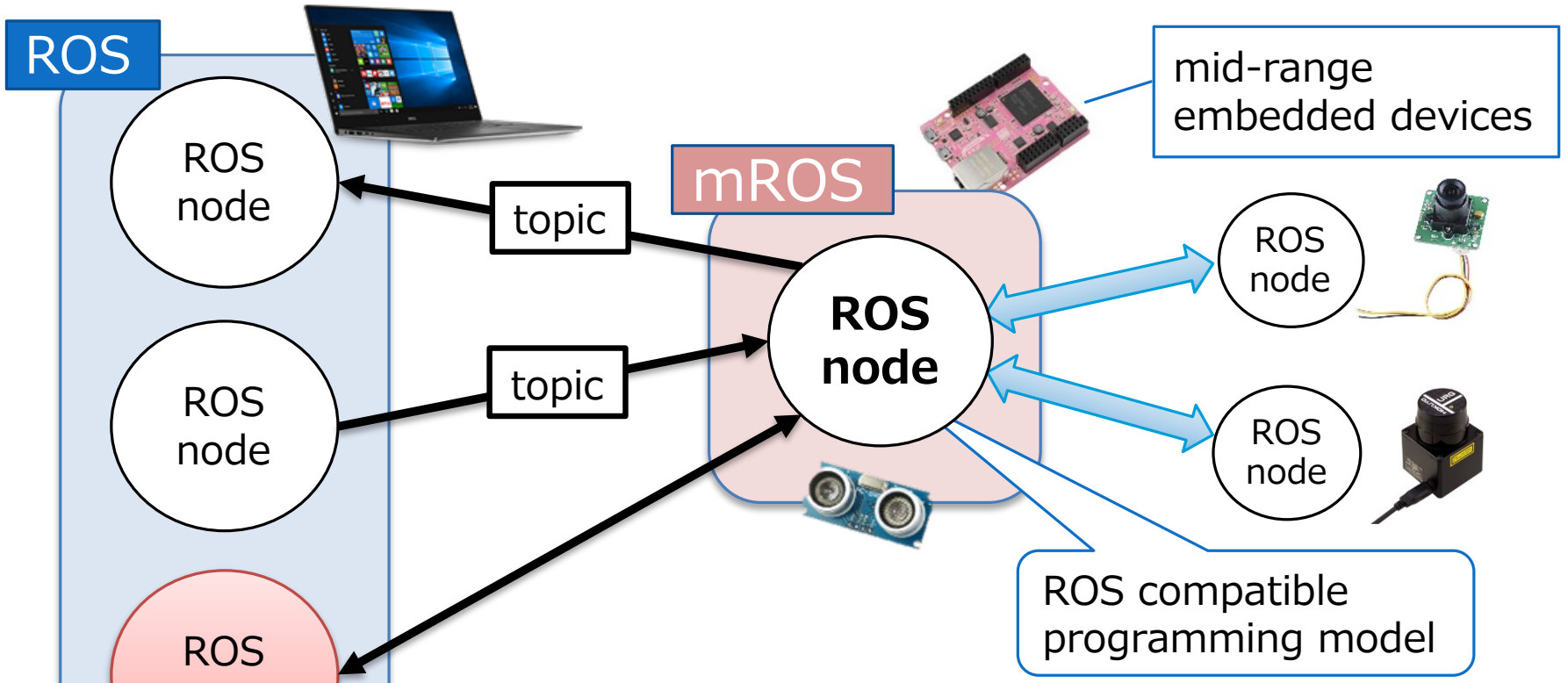
- 豊富なOSS資産
- Linux/Ubuntuの駆動が必要

- マルチプラットフォーム対応
  （組込み環境を含む？）
- OSS資産はまだ潤沢ではない

mROS

ROS 1ノードを組込み環境で
駆動するための
軽量な実行環境

mROSの全体像

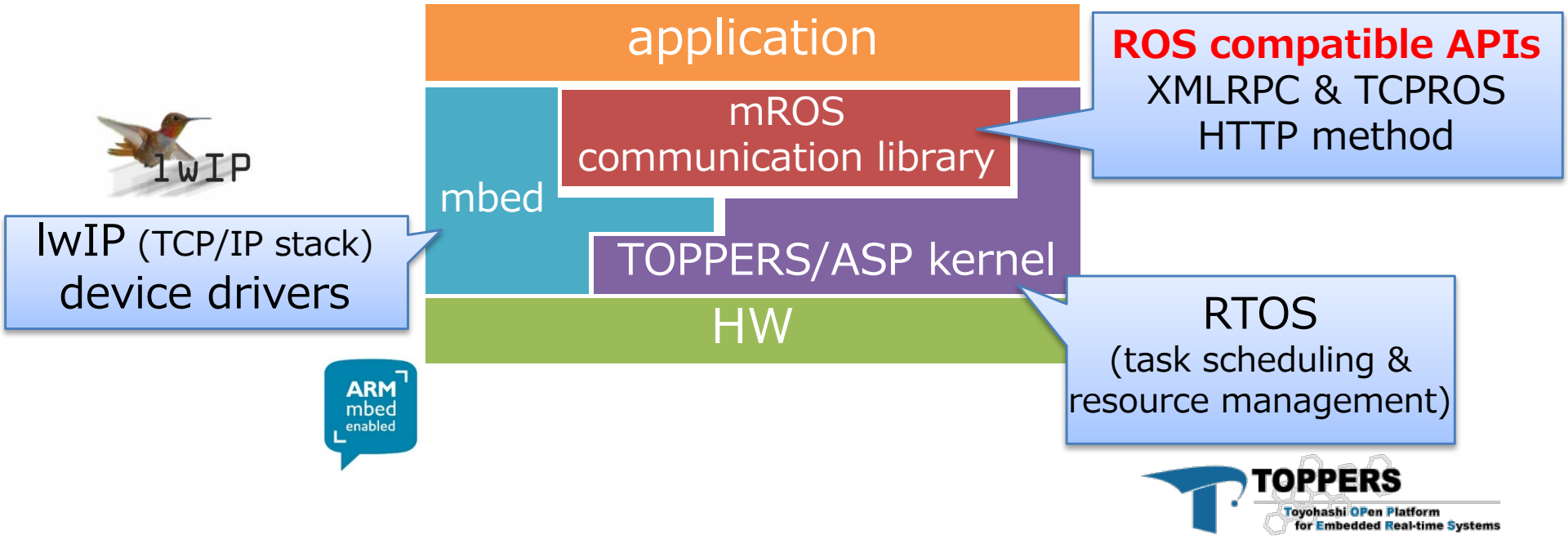


mROSの貢献

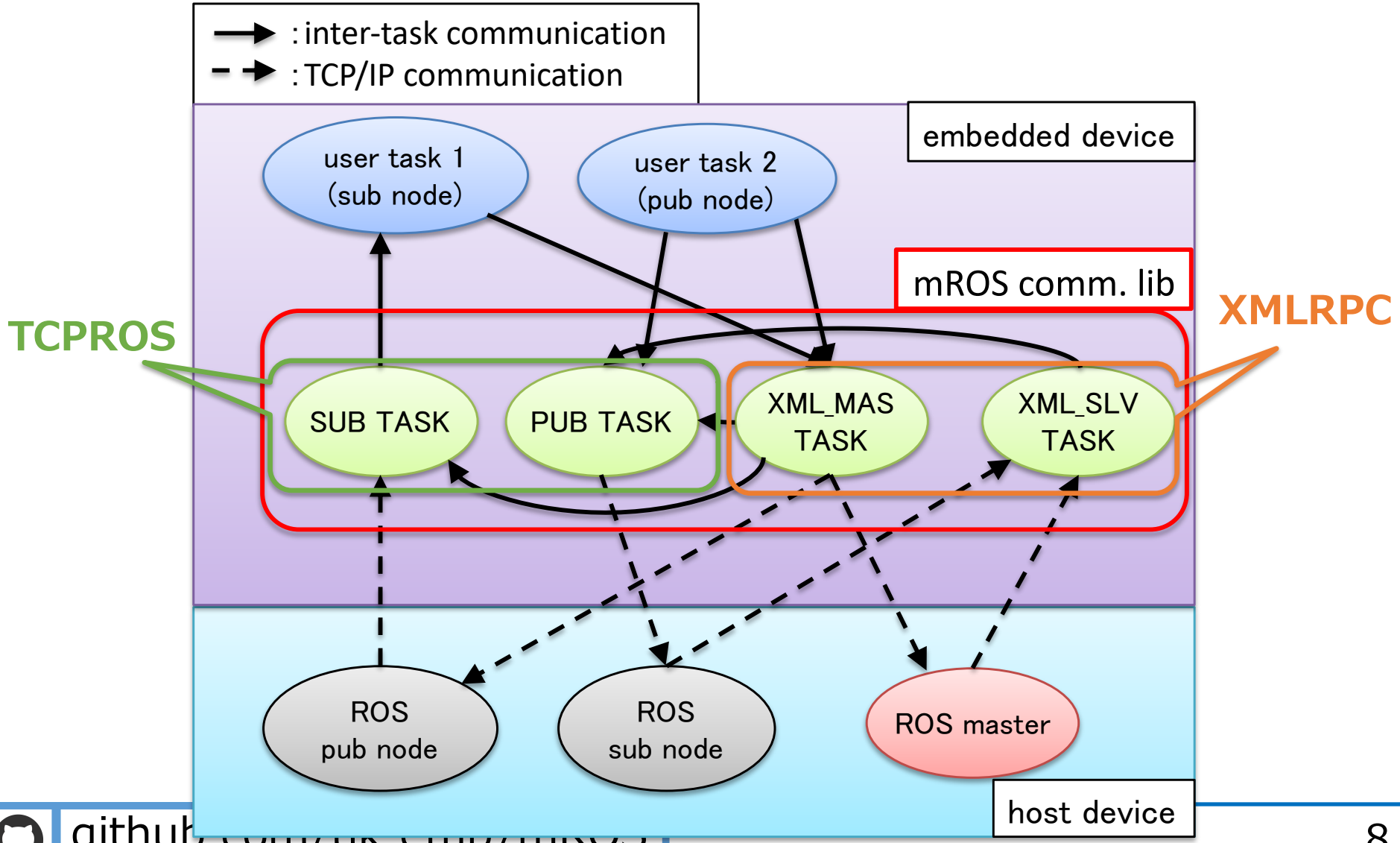
- ✓ ROS 1パッケージの組込み環境への移植性の提供
- ✓ 分散ロボット環境のエッジにおける省電力化とリアルタイム性の向上

ソフトウェア構造

- mROSタスクはROS APIで設計可能
  - mbedによるデバイスプログラミングも実現可能
  - マルチタスク（マルチノード）化は ITRONプログラミングで実現可能

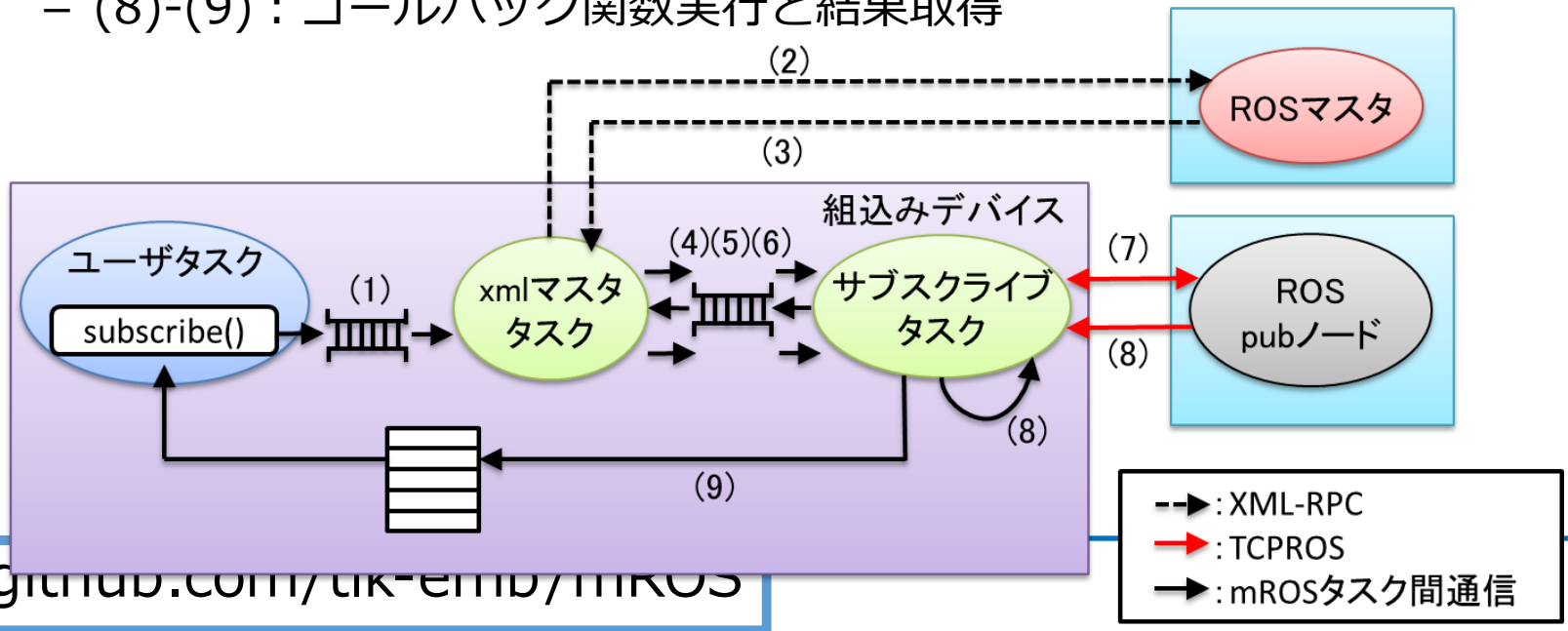


mROSのシステムタスク構成



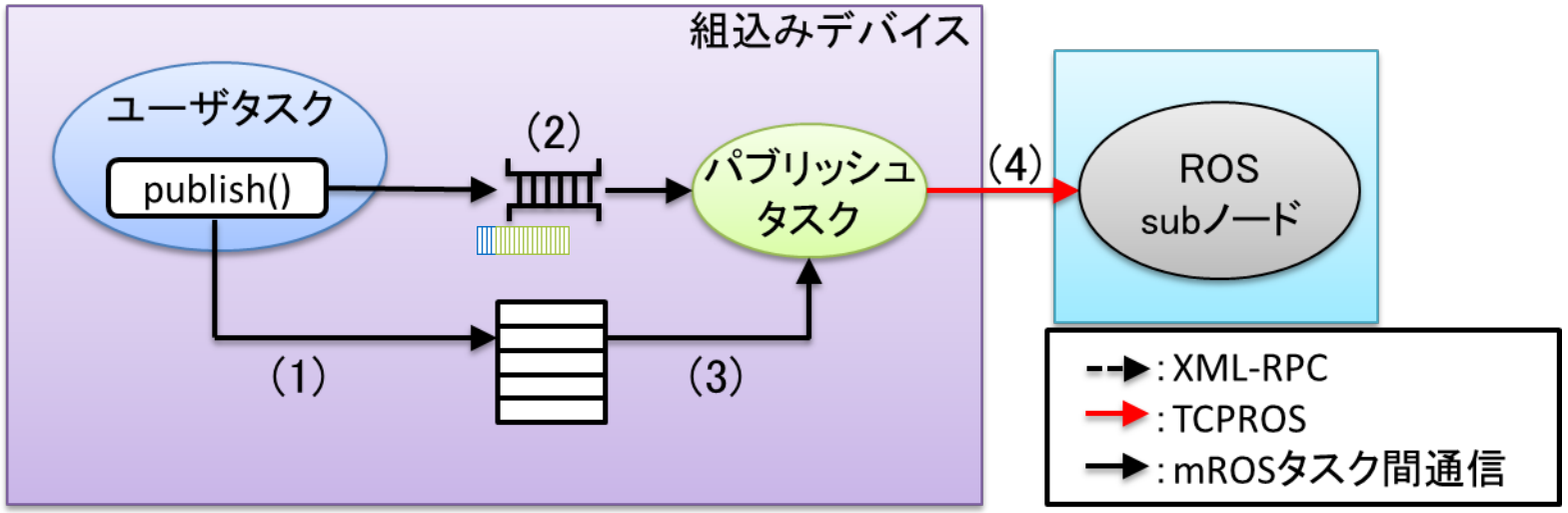
データ購読フロー

- subscribe()呼び出し
  - (1)-(3)：xmlマスタによるROSマスタへのノード登録
  - (4)：サブスクライブタスクによるmROSサブスクライバ初期化
  - (5)-(7)：ROSパブリッシャノードへのトピック購読リクエスト送信
- 周期実行によるデータ購読
  - (8)-(9)：コールバック関数実行と結果取得



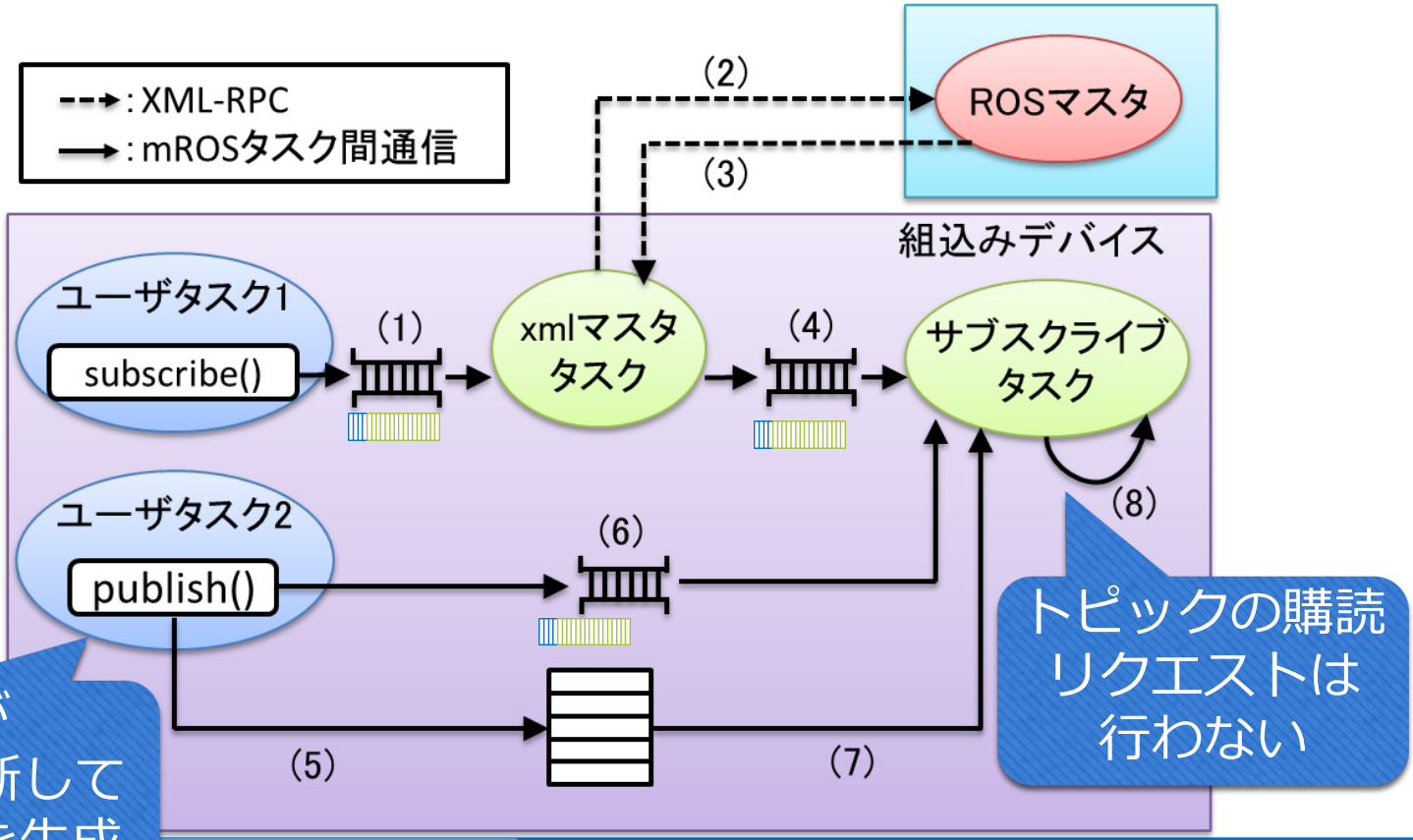
データ出版フロー

- データ出版：advertise()によるROSマスタへの登録後
  - publish()呼び出しによりデータ出版可能
  - (1)：共有メモリに出版データ書き込み
  - (2)：パブリッシュタスクへの出版通知
  - (3)：出版データ読み込み
  - (4)：対応するTCPソケットからデータ出版



デバイス内ノード間通信

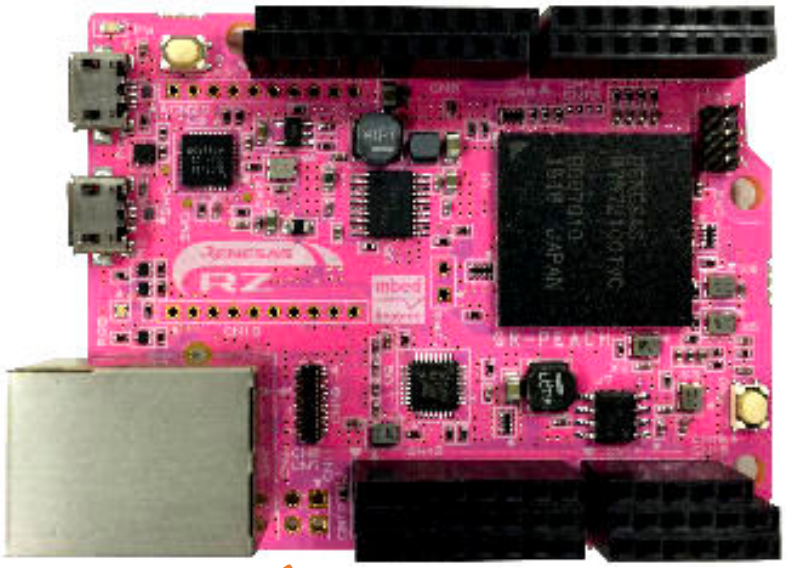
- タスク間の購読通知にはデータキューを使用
- publish()がサブスクライバタスクへと購読通知



現在の開発対象

- Renesas GR-PEACH

搭載マイコン	RZ/A1H
ROM/RAM	外部FLASH 8MB 内蔵10MB
動作周波数	400MHz
動作電圧	3.3V/1.18V



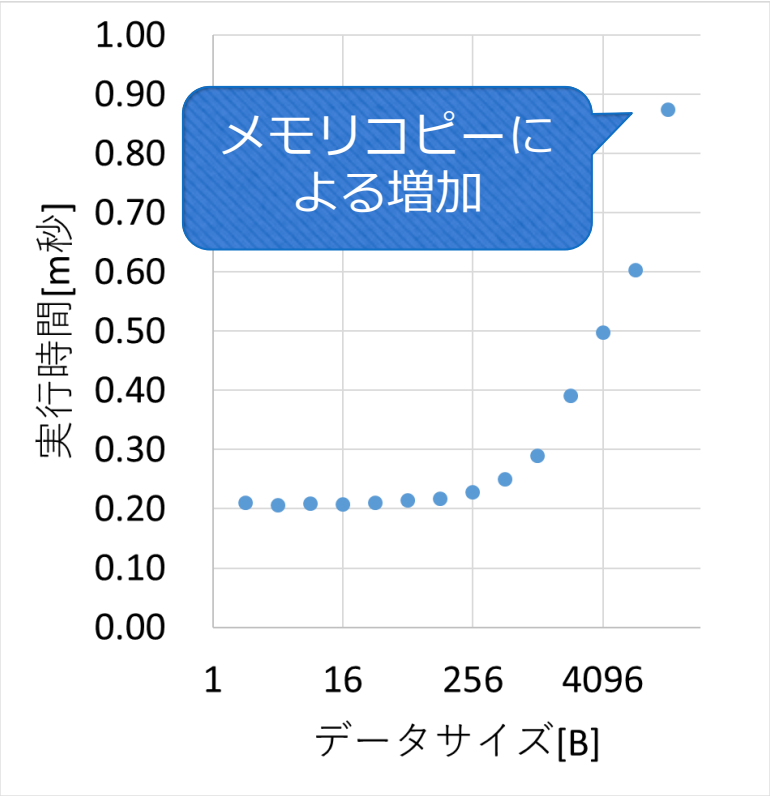
  - mbedライブラリ対応
    - ✓オンラインコンパイラ有り
  - Arduino互換ピン
  - 純正カメラシールドあり
    - ✓OpenCVも使える！？

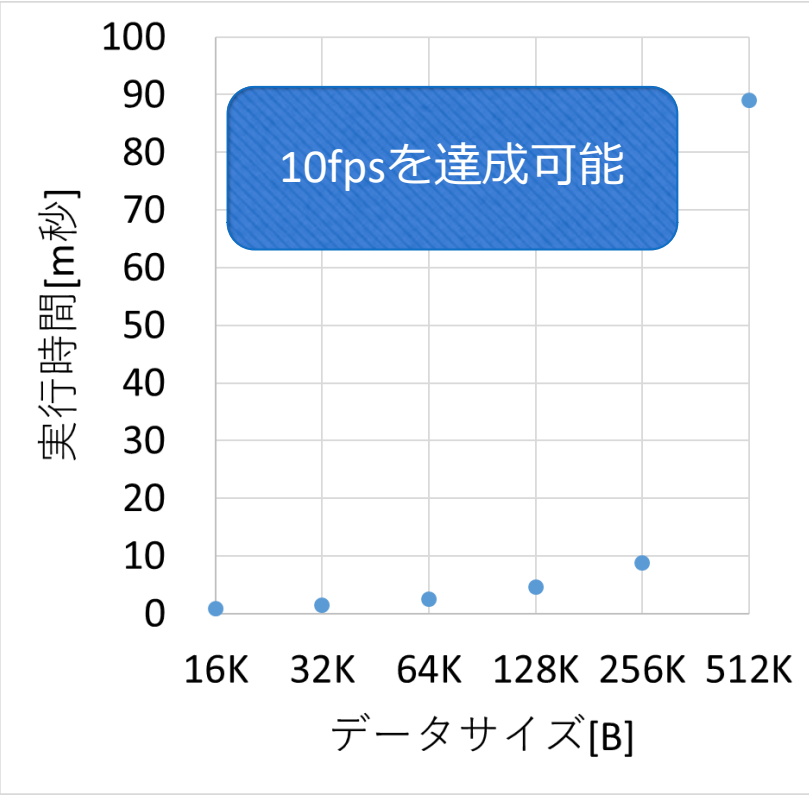
画像認識ROSノードが
きびきび動く，かも！？

http://gadget.renesas.com/ja/product/peach.html

評価結果：データ出版時間

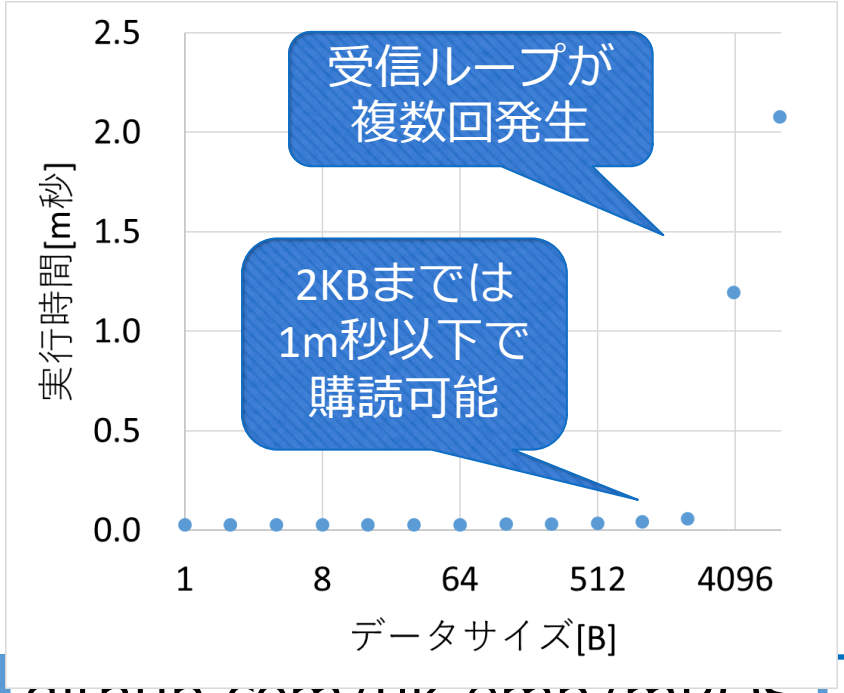
- publish()の呼び出しから完了までを計測
- 512KBのデータ出版は100m秒以下で実行可能

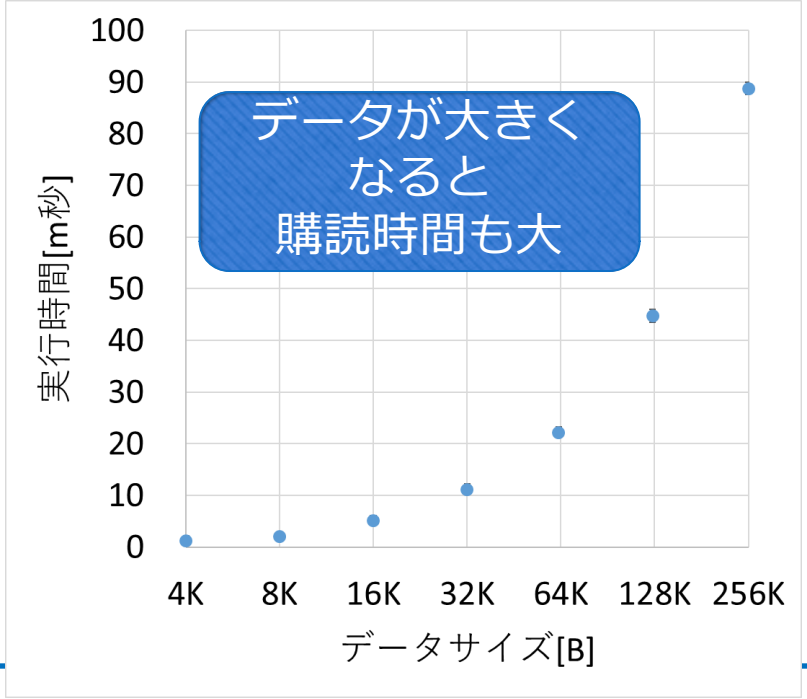




評価結果：データ購読時間

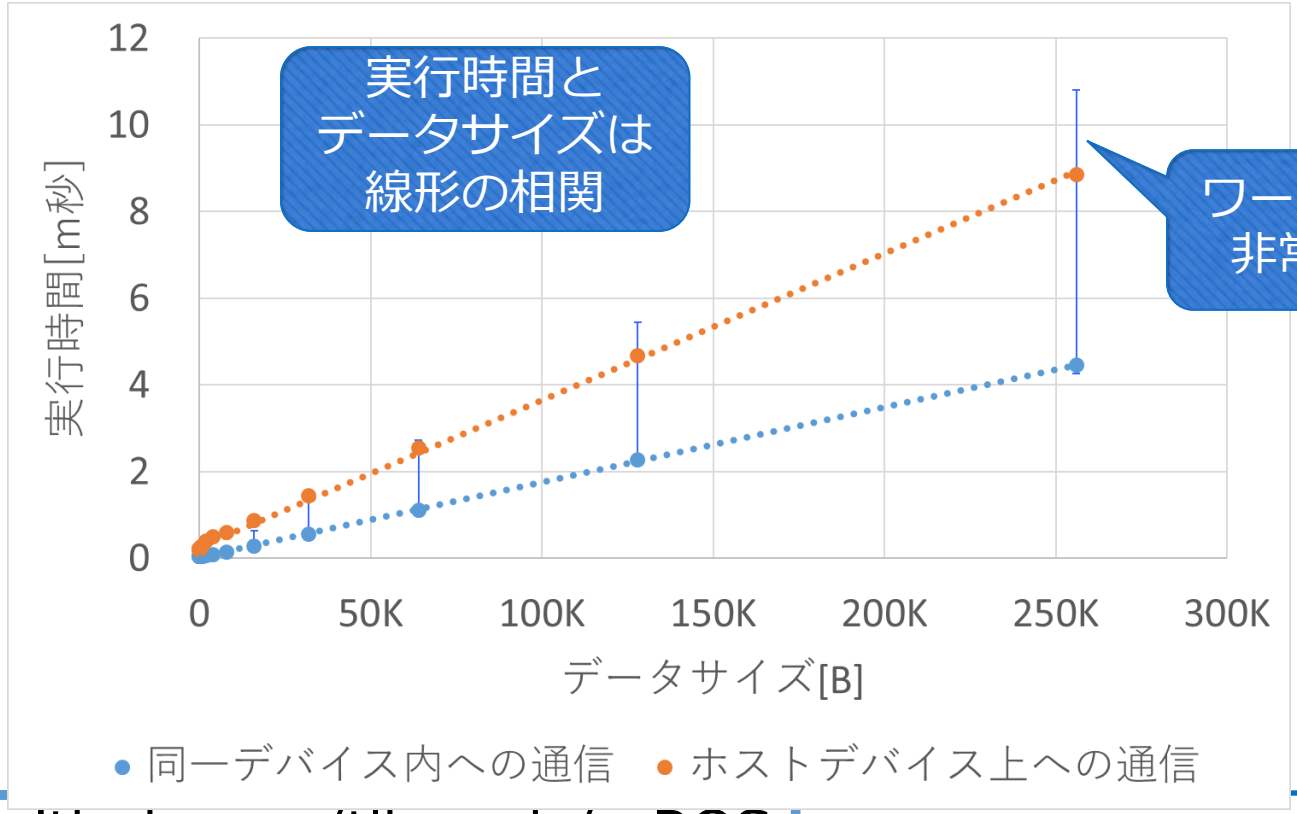
- データ受信からコールバック実行までを計測
- 2KB以下のデータ購読は1m秒以下で実行可能
- データサイズが大きくなるほどループ回数増大
- 大きなデータを購読するシステムは想定しない





評価結果：デバイス内通信

- publish()の呼び出しから完了までを計測
- ホストに対するデータ通信より2倍以上高速
  - データサイズが小さい（<1KB）場合では約5倍高速



評価結果：mROSサイズ

- 合計で2.6MB程度：GR-PEACHにとっては軽量
- さらなる軽量化は可能
  - 使用しないmbedライブラリを除く or そもそも生lwIPを使う
  - タスク間通信用のメモリ領域を削る

ライブラリ	text	data	bss	合計
カーネル	99,676	0	16,408	116,084
mbed	264,477	52,940	45,711	363,128
mROS	57,950	28	2,097,310	2,155,288
合計	422,103	52,968	2,159,429	2,634,500

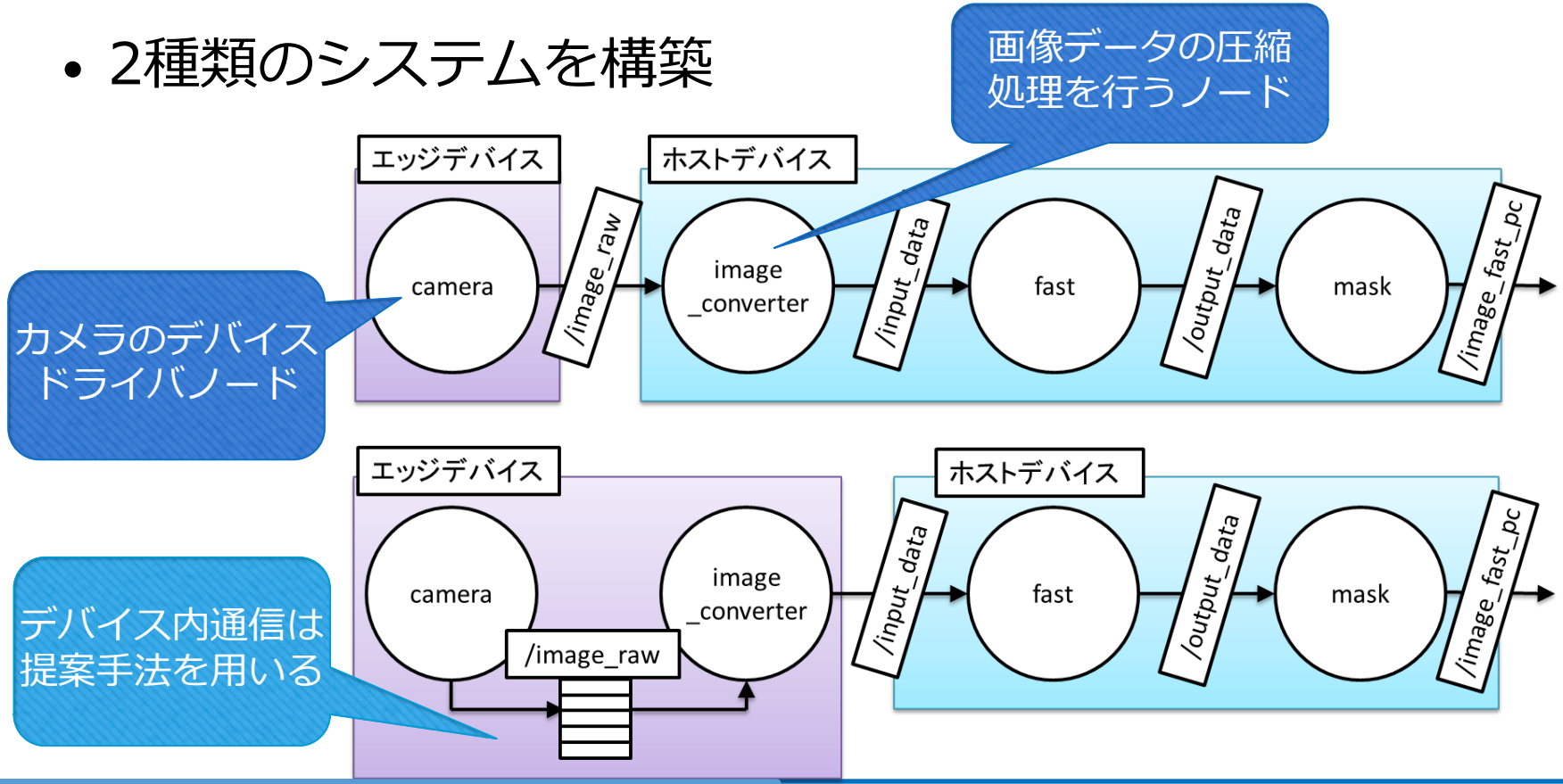
不要な
ライブラリも
含んでいる

共有メモリの容量を
静的に確保しているため

確保領域除いて合計600KB程度
GR-PEACHにとっては十分軽量

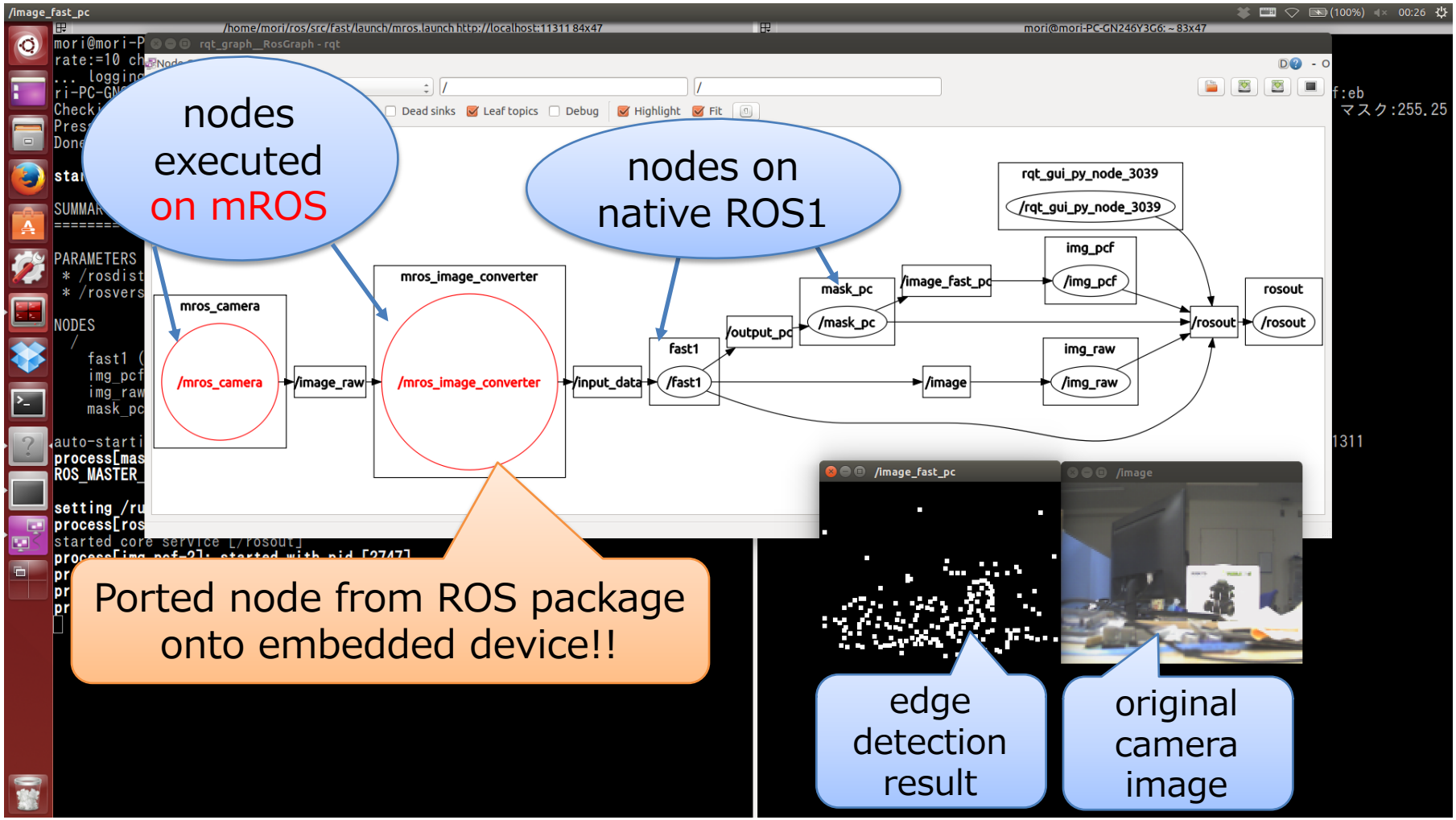
mROSの活用事例

- 特徴点抽出を行うROSパッケージ*1を対象
- エッジデバイス上でシステムの一部を実行
- 2種類のシステムを構築

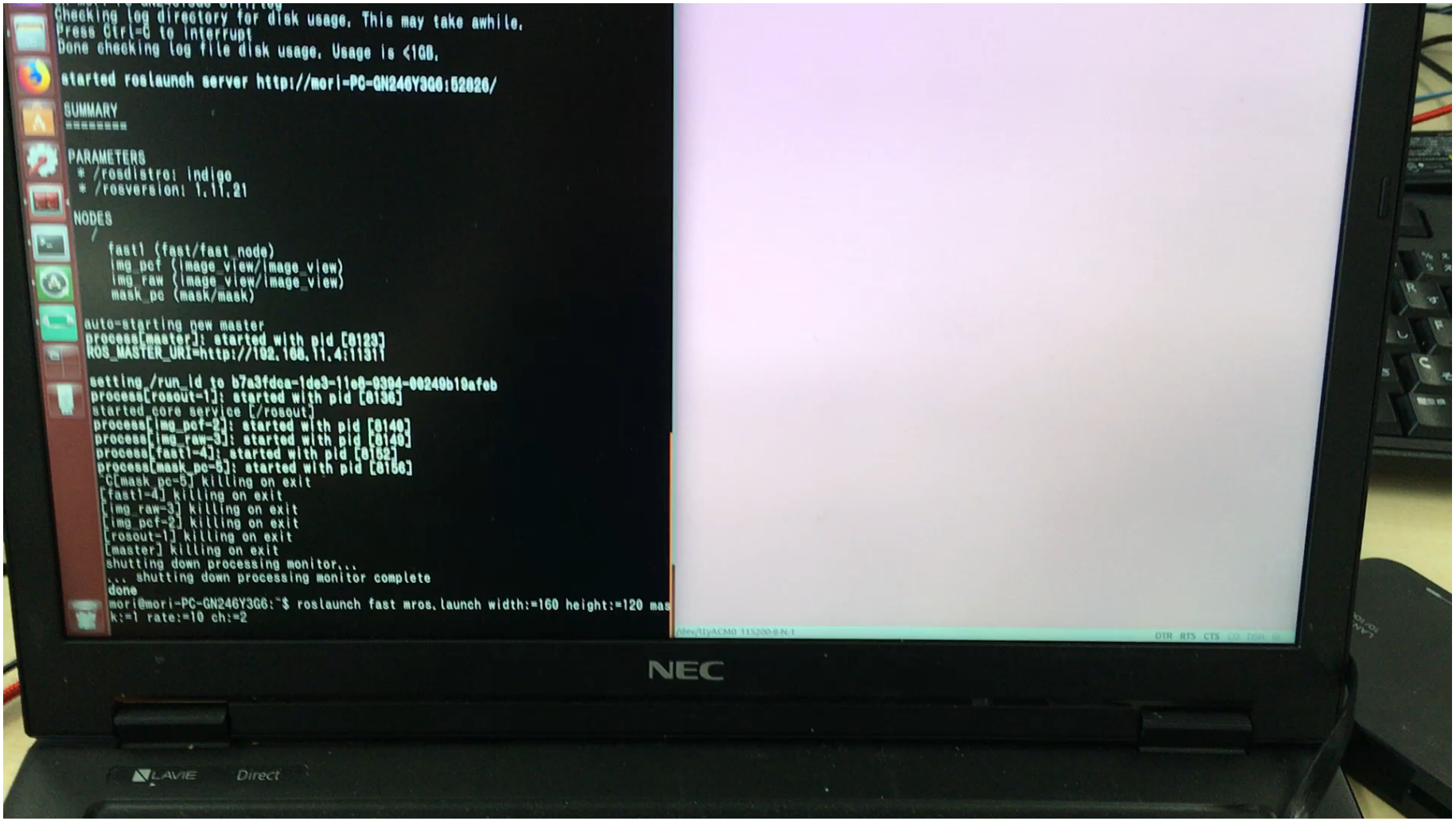


*1:宇都宮大学の大川猛先生からご提供

Case Study:
distributed edge detection system



Case Study:
distributed edge detection system



活用事例の考察

- mROS環境におけるノード設計
  - camera：mbedライブラリを使用して記述
  - image_converter：OpenCVライブラリを使用
    ✓ソースコードの大半はROSパッケージ資産から活用
  - アプリケーションサイズは約3,400KBで軽量

- エッジ処理による性能の違い
  - 75%の通信量削減が可能（300KB→75KB）
  - デバイス内のノード間通信は高速に実現できる
  - 実行周期を2倍高速化が実現（20m秒→10m秒）
  - エッジ処理によってシステム全体の効率化が図れる

ZytleBot
FPGA統合開発プラットフォーム

ZytleBot
ROSxFPGAの統合プラットフォーム

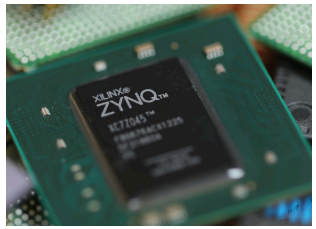
- FPGAとは？取り巻く現状と動向
- ROS対応ロボットへのFPGAの統合
- 得られる利点と効果

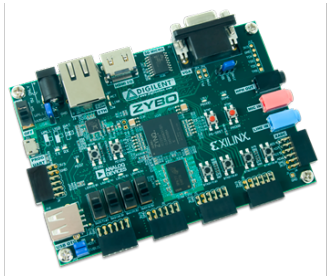
自律移動ロボットの開発に向けて

- 様々な状況に対応する**複雑な制御**
- バッテリ駆動のための**電力制約**
- AI/ML処理を実現する**処理性能**
- **多品種少量生産**の展開への対応

FPGA

ZYNQ™



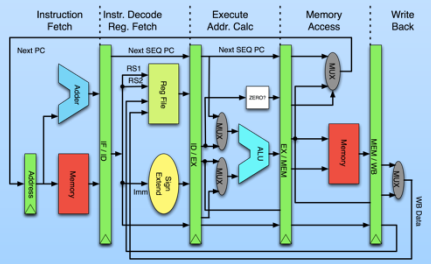


ROS *(Robot Operating System)*

- **高機能・多機能化**
- **省電力化**
- **並列性能**
- **設計柔軟性**

プロセッサ vs. ASIC

プロセッサ
&
ソフトウェア



性能・並列性

設計容易性・柔軟性

省電力性

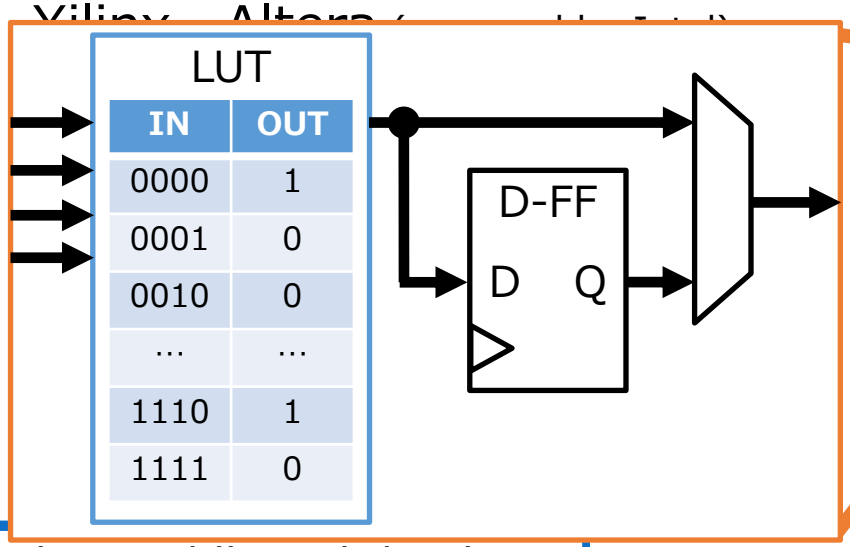
開発製造コスト

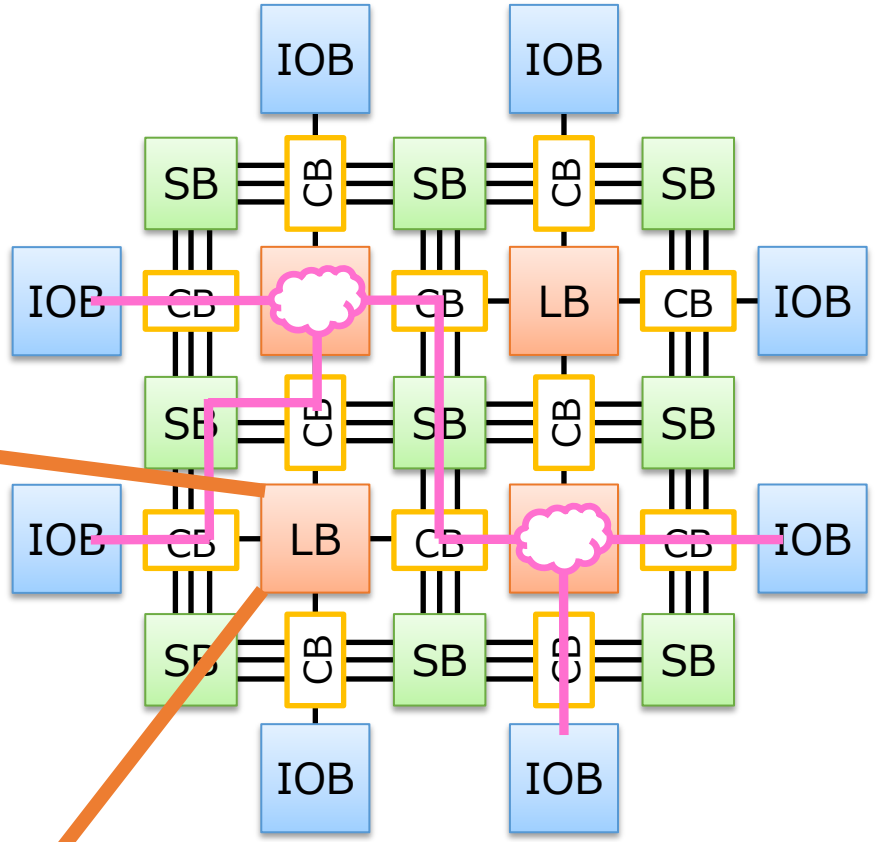
FPGA

ASIC
(専用回路)
ハードウェア

FPGAとは？

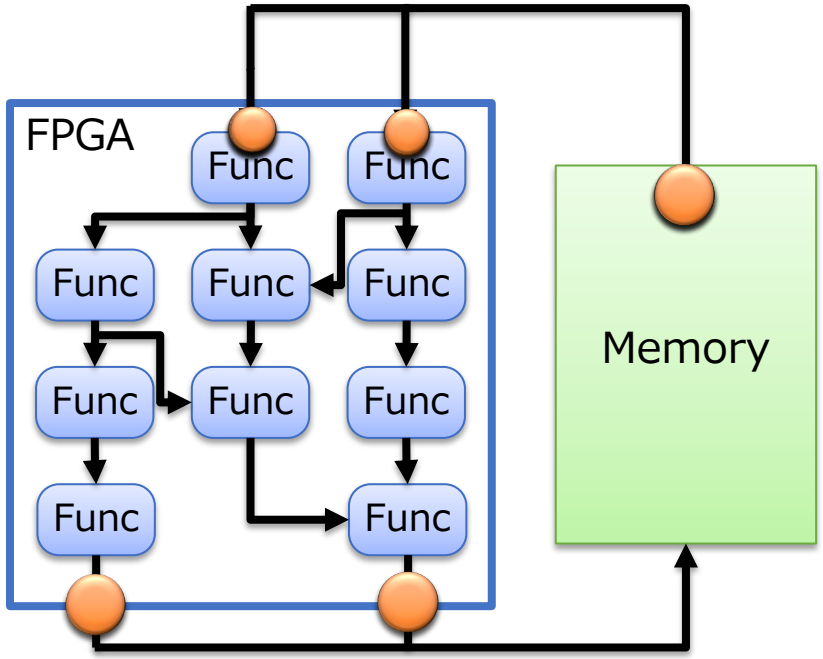
- Field Programmable Gate Array
  - 中身を改変可能なLSI
  - ハードウェアそのものの振る舞いを変えられる
  - 独自のデジタル回路を自由に何回でも形成できる
  - FPGA二大ベンダ：





Advantages of FPGA

- Various systems can be designed onto ***one LSI***
- High performance / low power consumption
- Parallel processing can be realized at task/data level
- Data streaming processing can be realized



Current Technology Trends

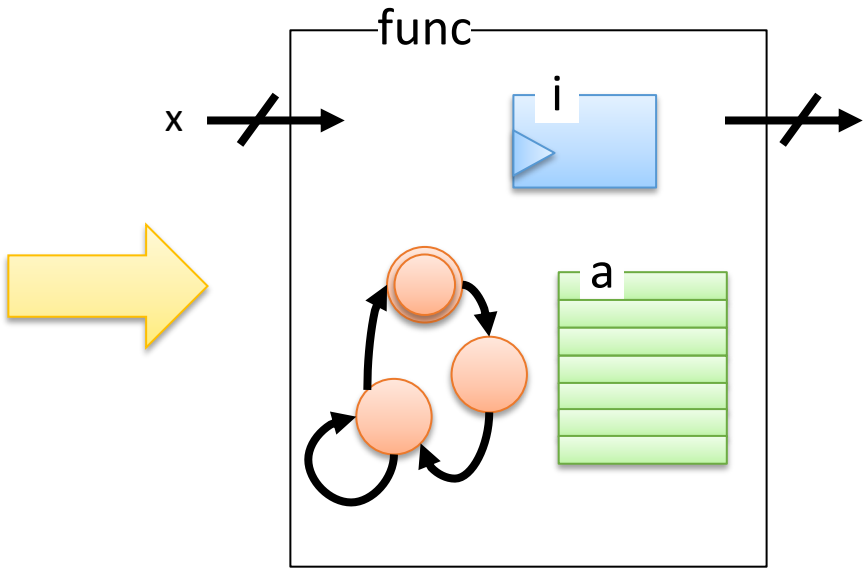
- Increase in circuit scale and amount of LB
  - High performance systems can be realized
  - Further increase will continue by new technology
    - ✓multi-die, 3D stacking,,,
- Tightly coupling with processors
  - General-purpose: Connection via PCIe to processors
  - Embedded: Integration with embedded processors

high-quality system design
in a short time
has become difficult,,,

High Level Synthesis (HLS)

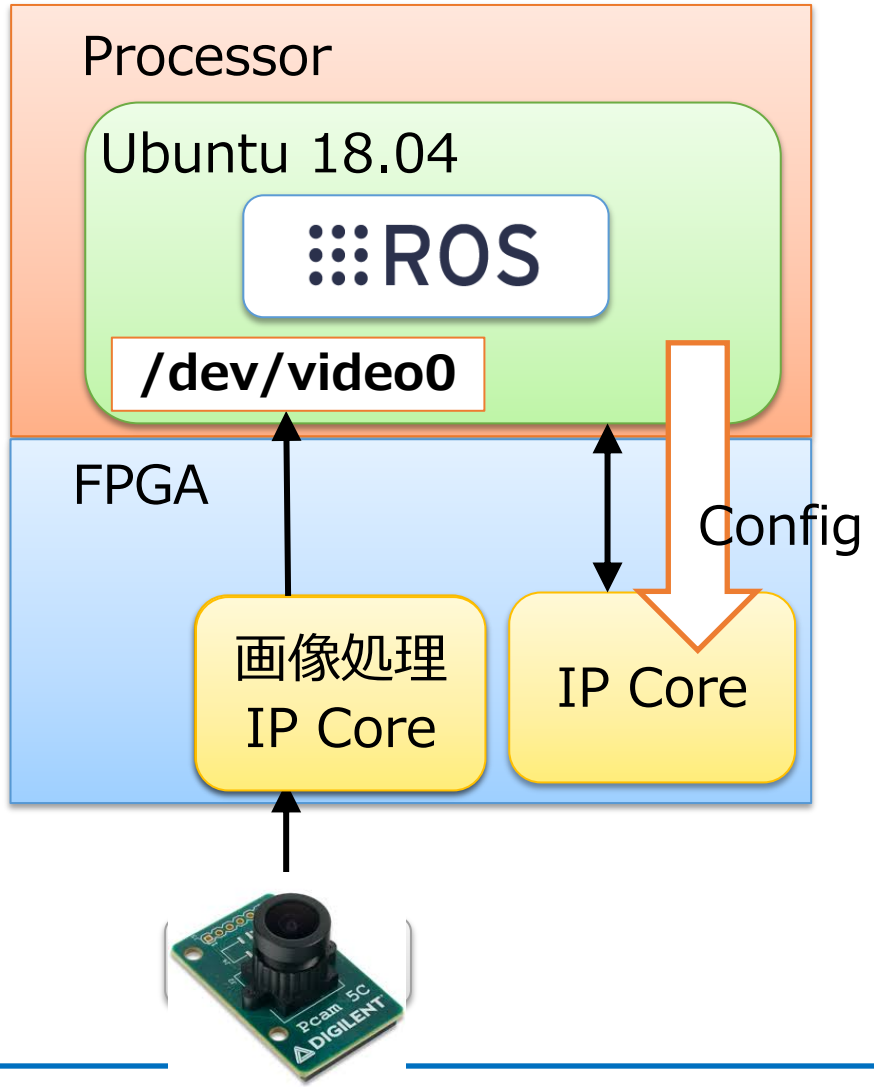
- **Solution to improve design productivity!**
  - Technology for synthesizing HDL from behavioral descriptions with a programming language
    - ✓C/C++ or its extension is commonly used
  - Abstraction level of design becomes higher

```
int func (int x) {
   int a[N];
   int i;
   for(i=0;i<N;i++){
      a[i] = ・・・;
      :
      :
   }
   :
}
```



プログラマブルSoC

- プロセッサとFPGAを1チップに収めたSoC
  - **Xilinx Zynq** / Intel SoC FPGA
- プロセッサ上では**Linux**やリアルタイムOSを稼働できる
- もちろん**ROS**も稼働可能！
- デバイスドライバを介してFPGA上のHW回路をデバイスとして扱える
  - 処理後データを直接受け取れる
  - FPGAの回路の再書き換えをLinuxから行うことも可能！(Linux kernel 4.10以降)



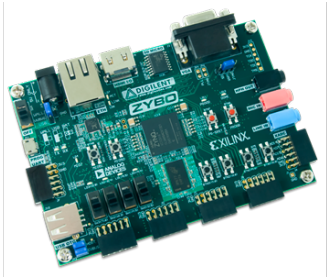
ROS対応ロボットへのFPGAの適用

ZYNQ™

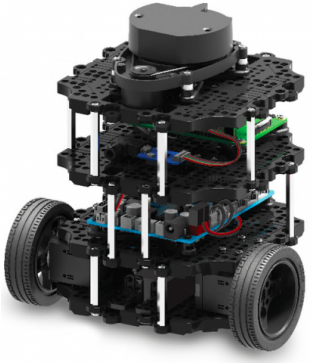
- 書換・再構成可能な回路を設計
- 省電力性・並列性能の向上
- CPUとの密結合による柔軟な構成

ROS

- 開発生産性の向上への期待
- 豊富なOSSパッケージ
- 分散システム・マルチスケール対応



- **ROSとFPGAの双方で高度な開発知識が求められる**
- **ロボット技術者がFPGAを導入する敷居は高い (逆も)**



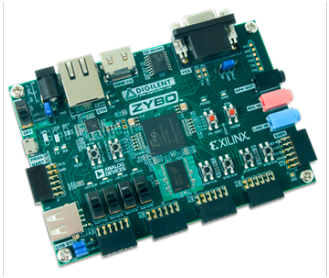
ROS対応ロボットへのFPGAの適用

ZYNQ™

ROS

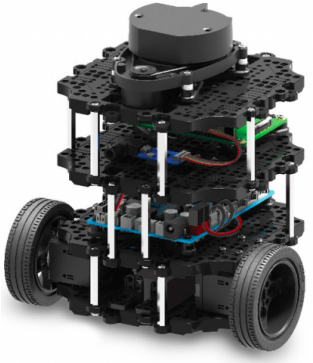
- 書換・再構成可能な回路を設計
- 省電力性・並列性能の向上
- CPUとの密結合による柔軟な設計

- 開発生産性の向上への期待
- 豊富なOSSパッケージ
- 分散システム・マルチスケール対応

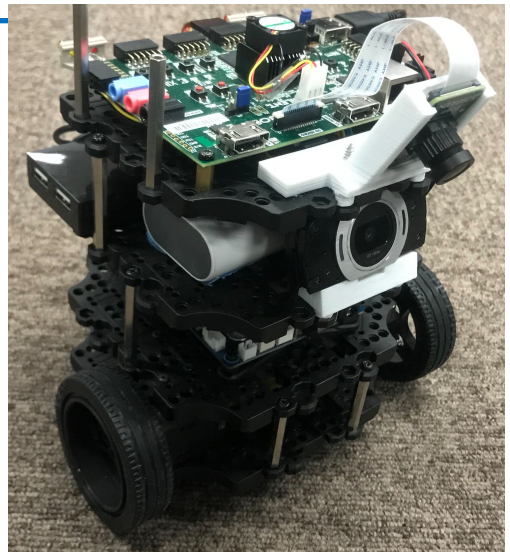


ZytleBot

ROSxFPGAを加速化する
統合開発プラットフォーム



ZytleBot



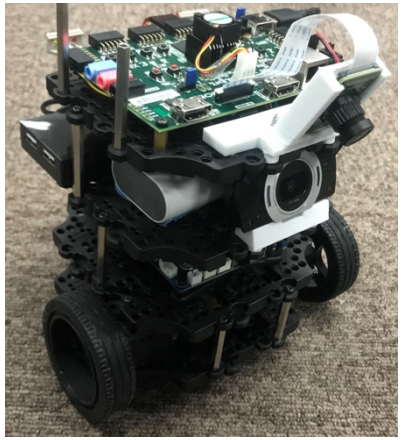
- ROSベースの統合開発プラットフォーム
  - –ROSのメリットを活かしながらFPGAを導入可能
  - –CPUでは難しい処理をFPGAにオフロード
    - ✓Vivado HLSによる回路設計が可能
  - –ROSノードからのFPGAのインターフェースをテンプレートとして提供
    - ✓PetaLinuxイメージ構成も提供

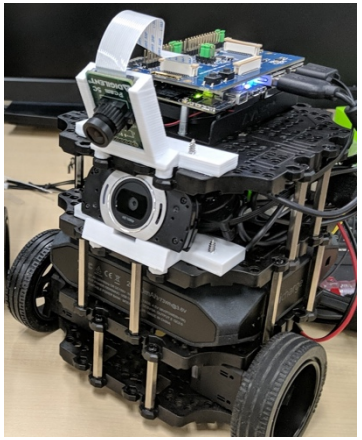
車体ロボット	TurtleBot3
メインボード	Zybo Z7-20
OS	Ubuntu 16.04 LTS
ROS	Kinetic Kame

自律移動ロボット開発における
FPGA開発技術の習得を容易化

github.com/tlk-emb/ZytleBot

ZytleBot





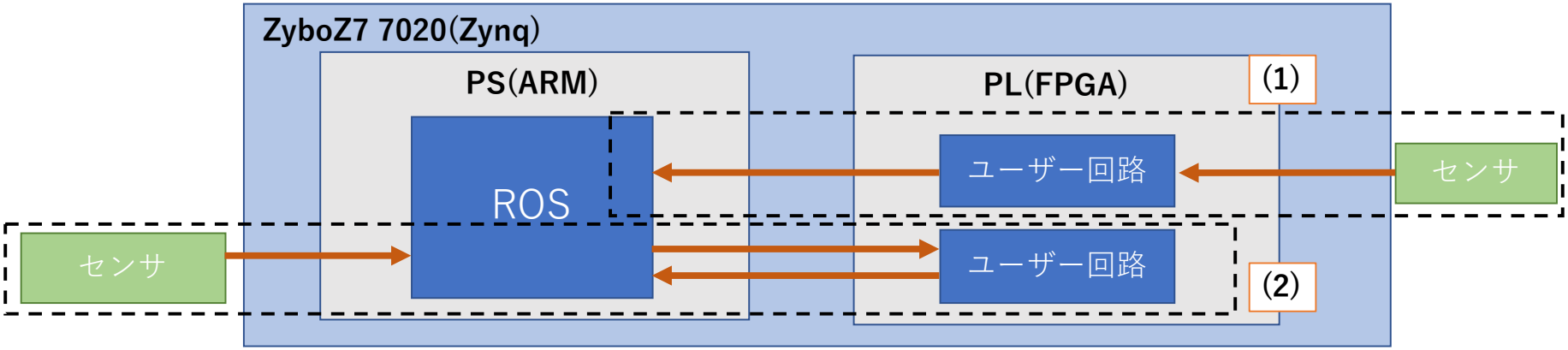
Robot	TurtleBot3
SBC	Zybo Z7-20
OS	Ubuntu 16.04
ROS	Kinetic Kame

Robot	TurtleBot3
SBC	Ultra96
OS	Ubuntu **18**.04
ROS	Crystal Clemmys

現在はDashingに対応

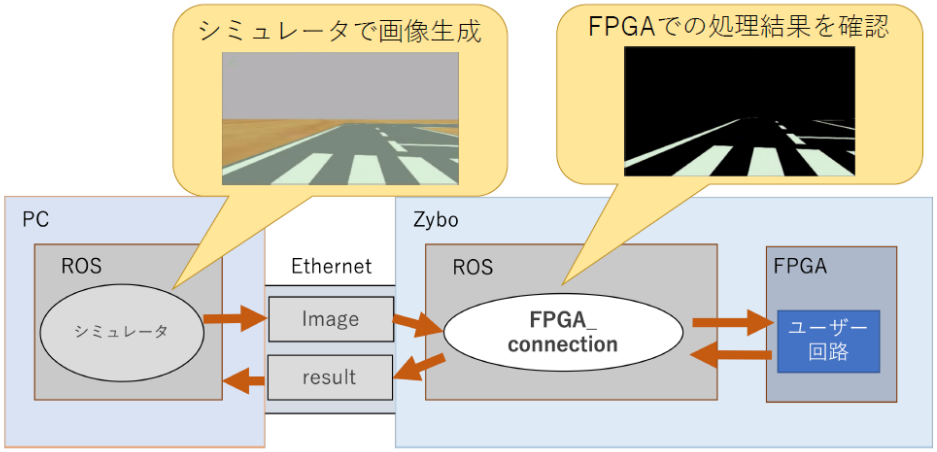
ZytleBot におけるFPGAの活用方式

(1) センサデータをFPGAで直接処理して
　　プロセッサの負荷を軽減
(2) プロセッサからFPGAにデータを送信して
　　処理を高速化

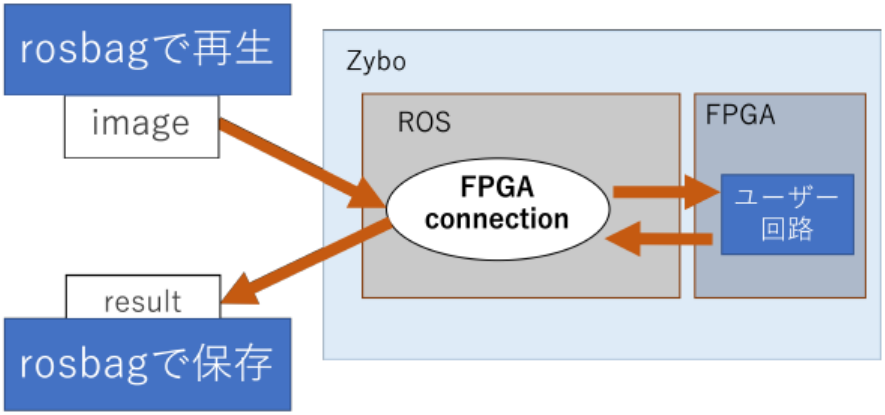


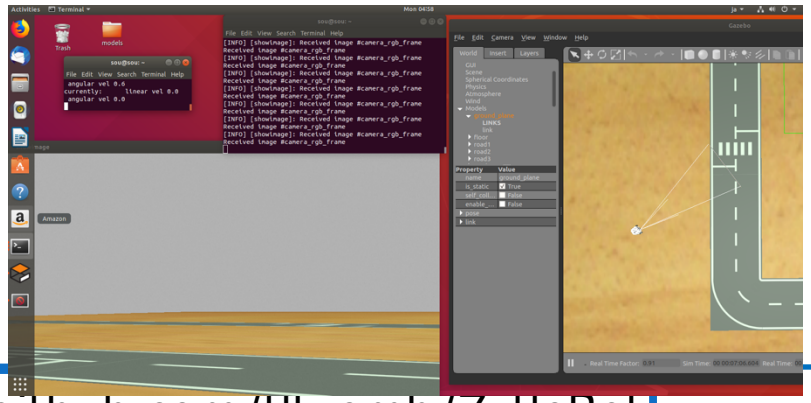
ROSツールの活用

- シミュレーションデータを FPGAに送信して処理を確認



- FPGAモジュールのテストや 結果比較をrosbagで容易化



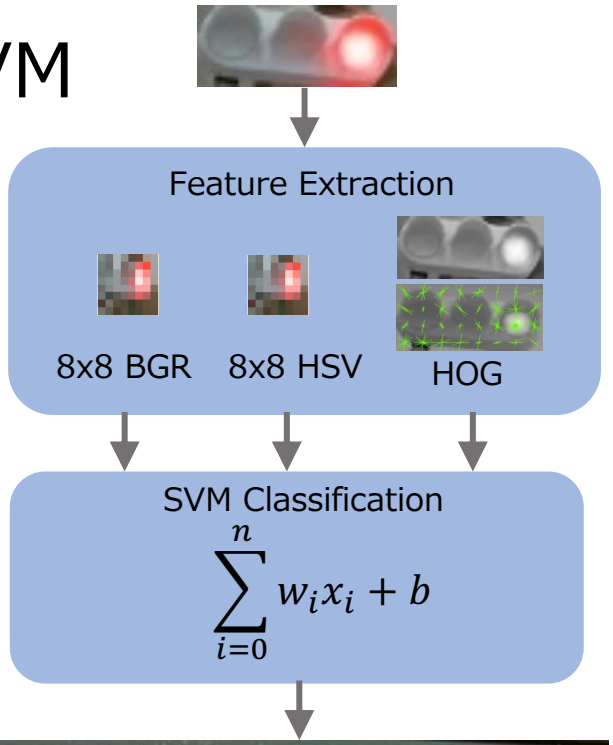


開発・検証コストの 工数削減に貢献

github.com/tlk-emb/ZytleBot

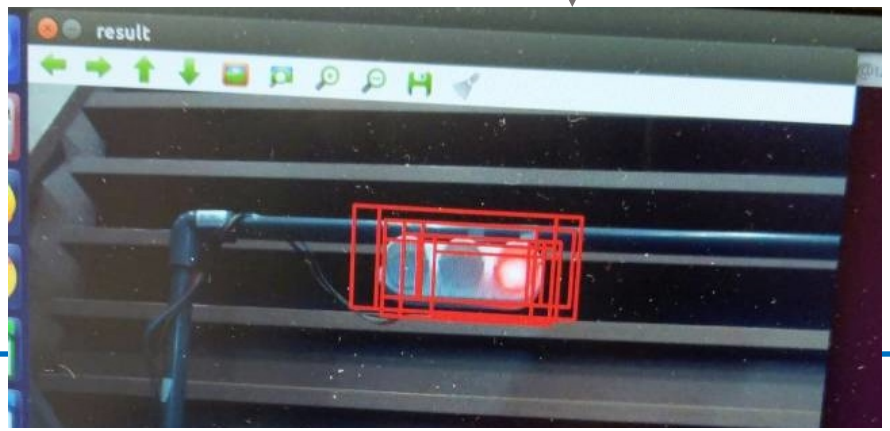
FPGAの活用：赤信号検出

- スライディングウィンドウ法＋SVM
  - –入力:320pix*240pixフレーム画像
  - –出力:891個のウインドウの推定結果



- 実装結果(高位合成による設計)
  - –SWのみ　：1700 ms (0.58 fps)
  - –HW使用時：6.22 ms (160 fps)

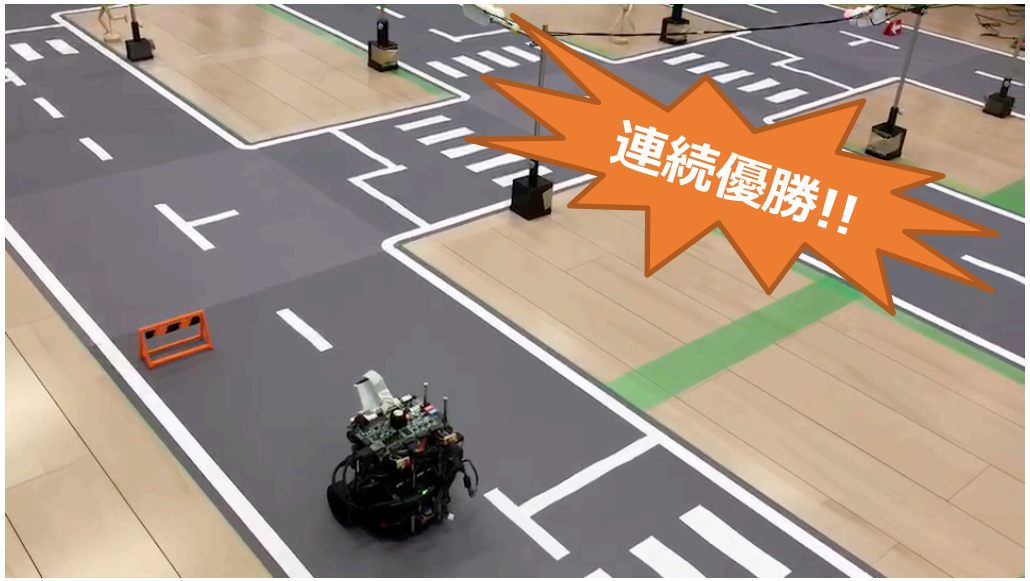
275倍の高速化！

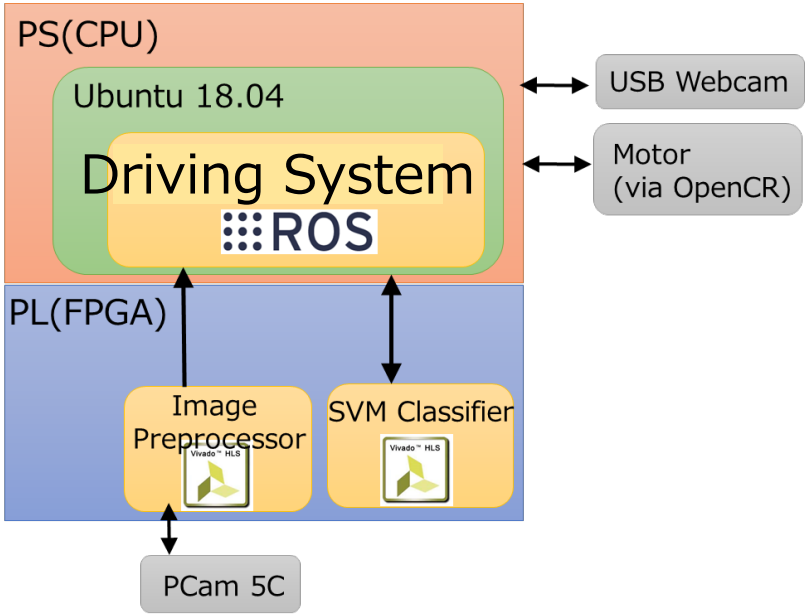


https://github.com/lp6m/ImageDetectionHW2

ZytleBot の活用事例

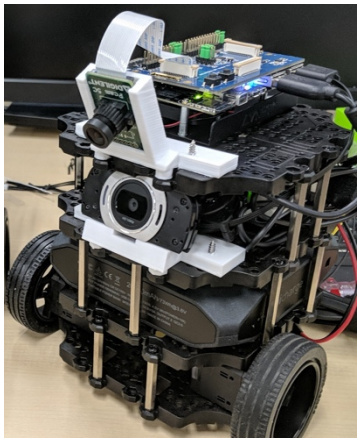
- FPGA関係の国際会議の併設コンテストに参加
  - –FPT2018 FPGA Design Competition
  - –HEART2019 FPGA Design Competition





おわりに

- *ZytleBot* ROSxFPGAを実現する統合開発プラットフォーム
  - –ROSのツールやパッケージを活かして
    自律移動ロボットを開発可能
  - –FPGAを容易に導入可能
  - –省電力性および性能を向上
    - ✓Vivado HLSによる回路設計が可能
    - ✓ROSノードからのFPGA活用方式を提供



- 今後の展開
  - –オープンソースパッケージの拡充
  - –FPGAアクセラレーション回路の拡充

京都大学 KYOTO UNIVERSITY

ROBOTIS

AVNET Reach Further™

IoT／自動運転時代の仮想シミュレーション環境

- IoTシステムサービス構築時のお困り事
- 『箱庭』の狙いとコンセプト・アーキテクチャ
- プロトタイプモデルの開発(進行中)
- 研究開発プロジェクトへのお誘い

IoT/自動運転時代のシステム構築

IoT/自動運転システムのような複雑なシステムでは,
少なくとも以下の２つの課題があると考えています.

1. システム構築の視点
   - 様々な機器/ソフトウェアが絡んでいるため，実証実験の現場では，機器間の不整合が頻発し、トラブル対応の時間・手間・コストがかかることが想定される.
2. サービス構築の視点
   - IoTサービスとして，様々なIoT機器がある中で，それらをどう組み合わせると，斬新で画期的なサービスを創出できるかわからない.
   - 新しいサービスを検討するにしても，実物のロボットでは準備・手間がかかりすぎる.
   - 様々なサービス検討のために，IoT機器の仕様が頻繁に変更され，システム開発者に負担がかかる.

ロボットサービスの構築時の場合

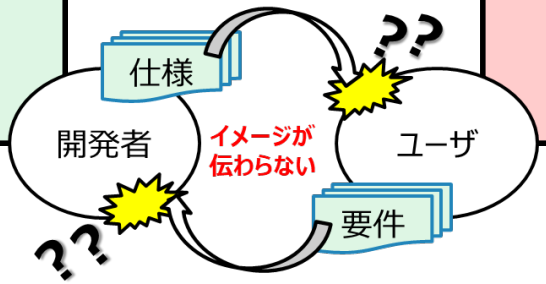
テスト環境構築に手間取る

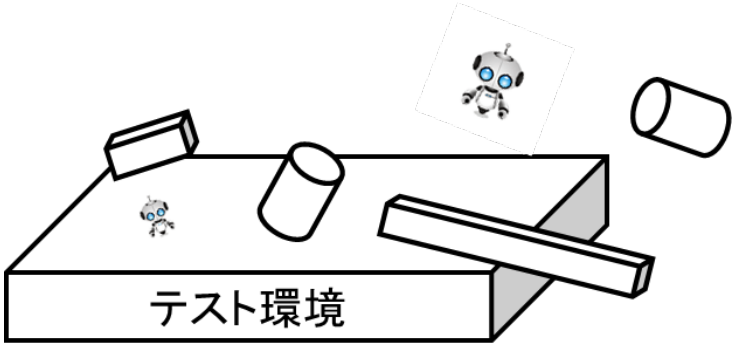
開発者とユーザの間でイメージ共有が難しい

思い付いたアイディアを手軽に試したい

ロボットシステム開発者

ロボットサービス提供者



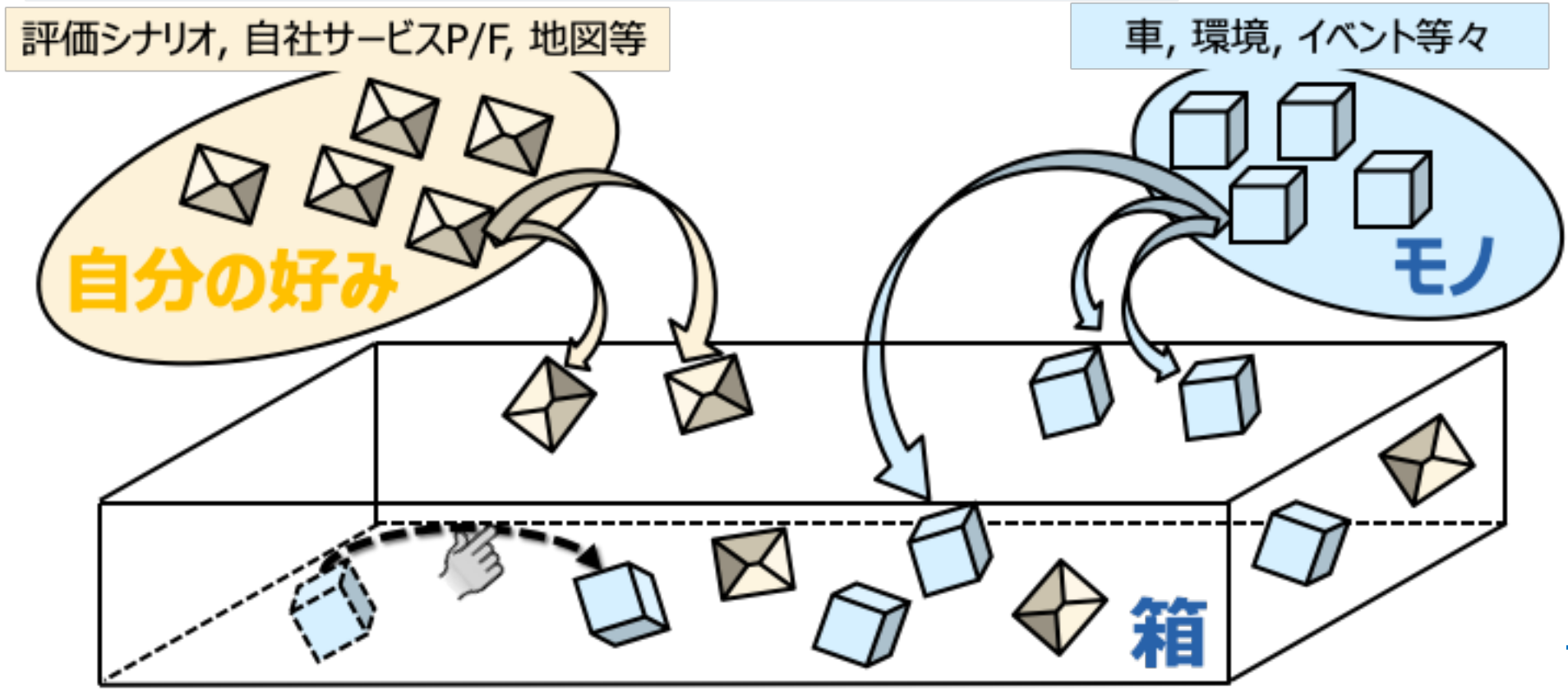


様々な機器を組み合わして色々試したい

IoT機器提供者／開発者

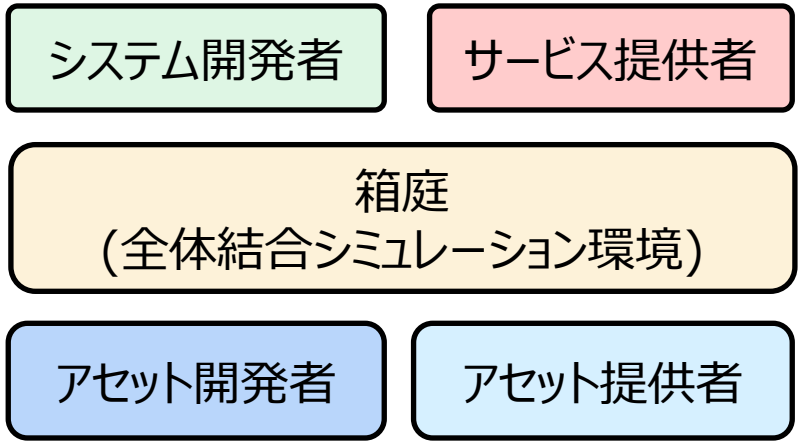
『箱庭』とは？

- **箱**の中に，
- 　いろいろな**モノ**を**自分の好み**で配置して
- 色々試せる！

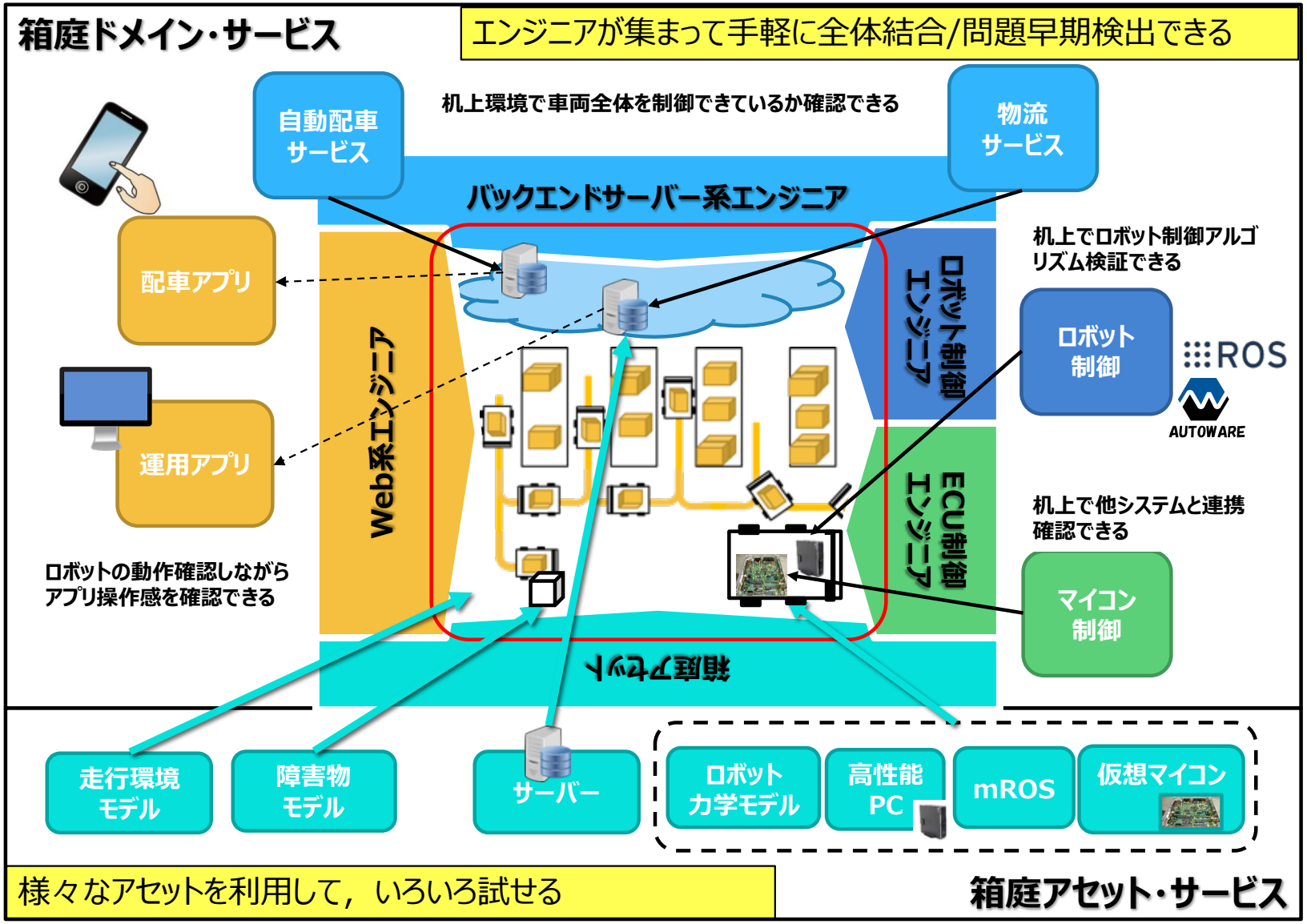


『箱庭』の目指すところ

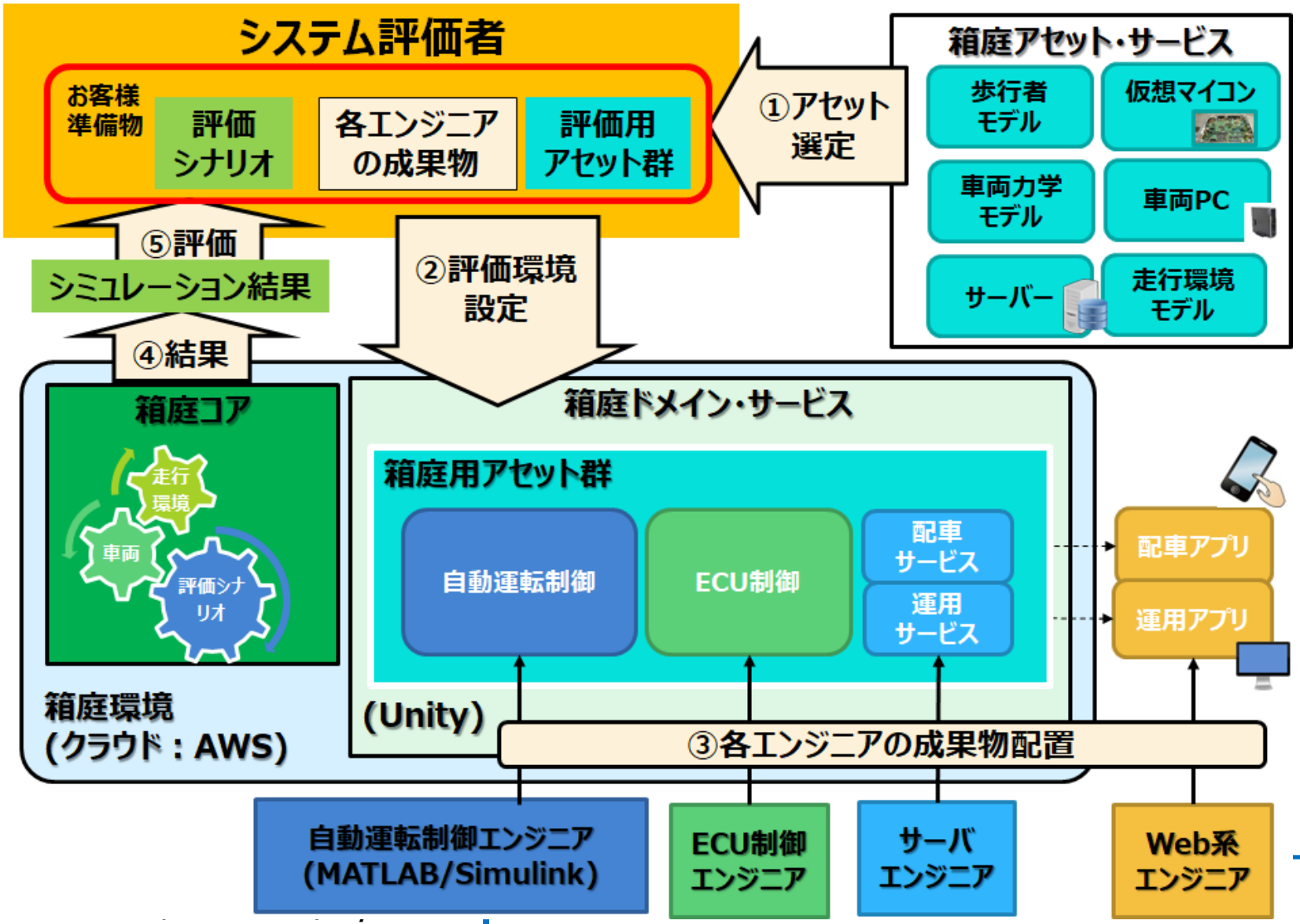
- **IoT／自動運転時代のシステム全体を検証するためのシミュレーション環境**
  - 各機器がネットワーク接続されたアーキテクチャを想定
- **箱庭の利用者**
  - システム開発者
  - サービス提供者
  - 箱庭アセット開発者／提供者
    - ✓＝システム構成要素
- 目指す強みと新しさ
  - IoTの各要素を連携させて**任意の精度で**検証可能
  - **検証**の対象／抽象度／レベルを**任意に変更できる**



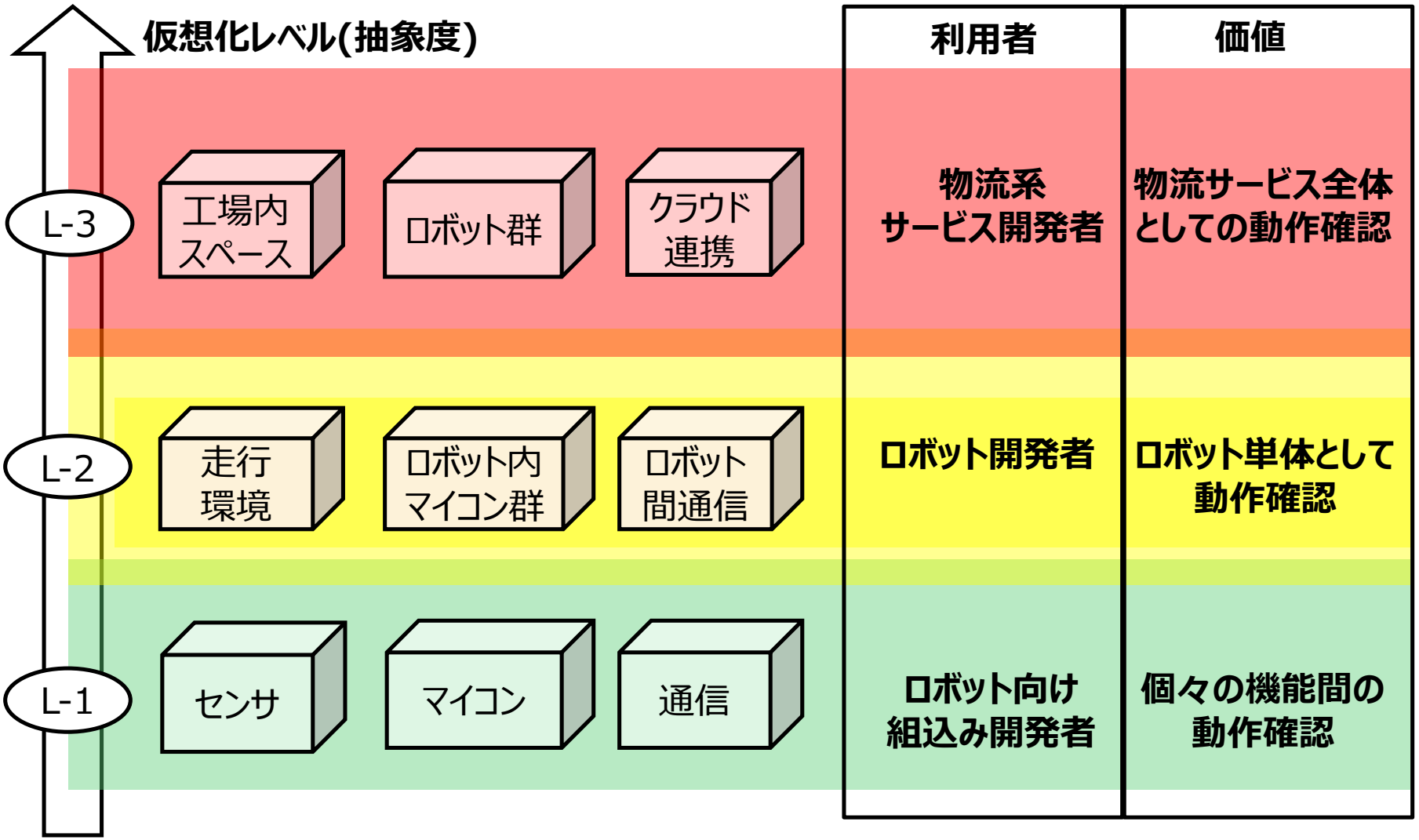
『箱庭』の目指すところ


箱庭ドメイン・サービス
エンジニアが集まって手軽に全体結合/問題早期検出できる
自動配車サービス
机上環境で車両全体を制御できているか確認できる
物流サービス
バックエンドサーバー系エンジニア
配車アプリ
運用アプリ
Web系エンジニア
ロボット制御エンジニア
ECU制御エンジニア
箱庭プラットフォーム
机上でロボット制御アルゴリズム検証できる
ロボット制御
ROS
AUTOWARE
机上で他システムと連携確認できる
マイコン制御
ロボットの動作確認しながらアプリ操作感を確認できる
走行環境モデル
障害物モデル
サーバー
ロボット力学モデル
高性能PC
mROS
仮想マイコン
様々なアセットを利用して，いろいろ試せる
箱庭アセット・サービス

利用イメージ


システム評価者
お客様準備物
評価シナリオ
各エンジニアの成果物
評価用アセット群
①アセット選定
箱庭アセット・サービス
歩行者モデル
仮想マイコン
車両力学モデル
車両PC
サーバー
走行環境モデル
⑤評価
シミュレーション結果
④結果
②評価環境設定
箱庭コア
走行環境
車両
評価シナリオ
箱庭ドメイン・サービス
箱庭用アセット群
自動運転制御
ECU制御
配車サービス
運用サービス
配車アプリ
運用アプリ
箱庭環境
(クラウド：AWS)
(Unity)
③各エンジニアの成果物配置
自動運転制御エンジニア
(MATLAB/Simulink)
ECU制御エンジニア
サーバエンジニア
Web系エンジニア

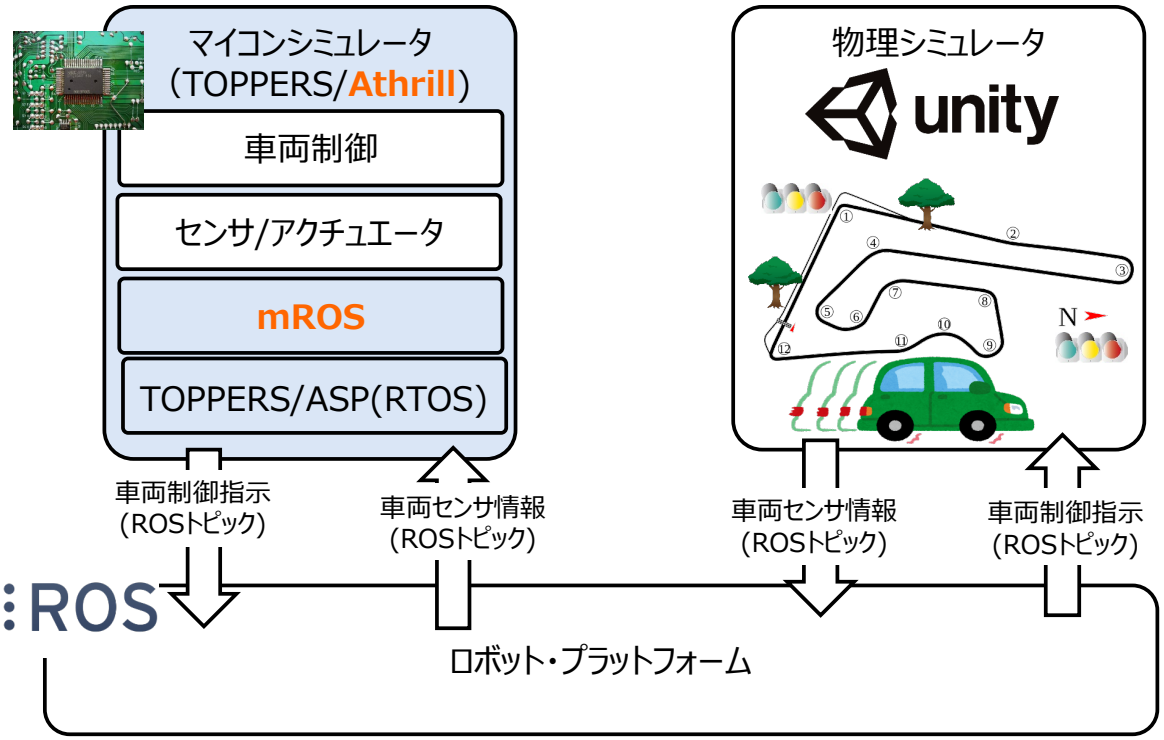
マルチレイヤ・コンセプト
(例：ロボットシステム)



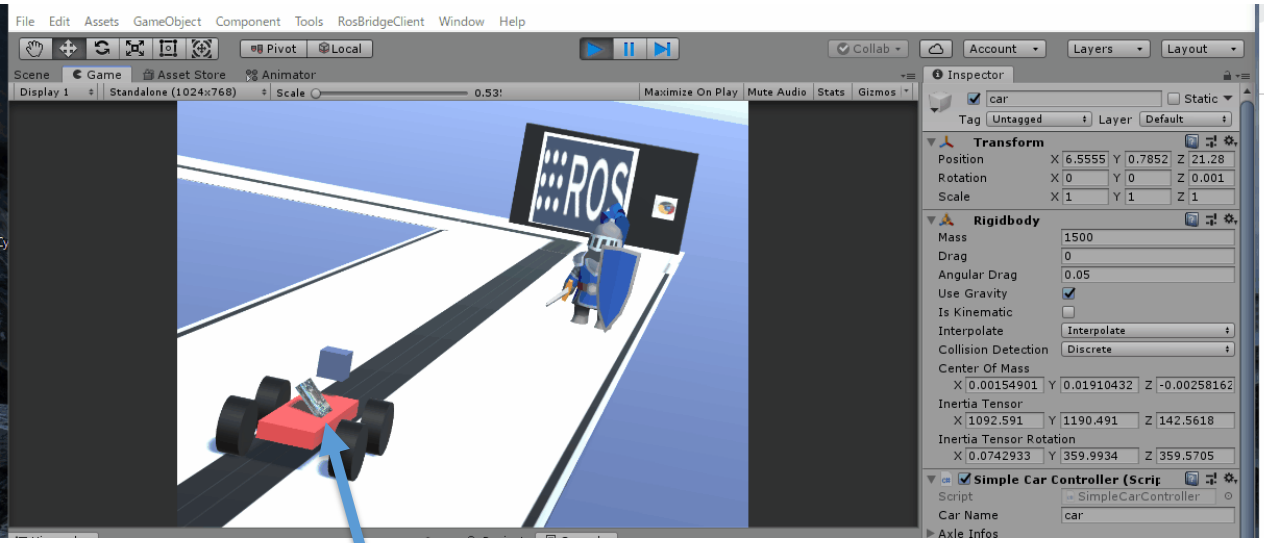
現状のデモのご紹介

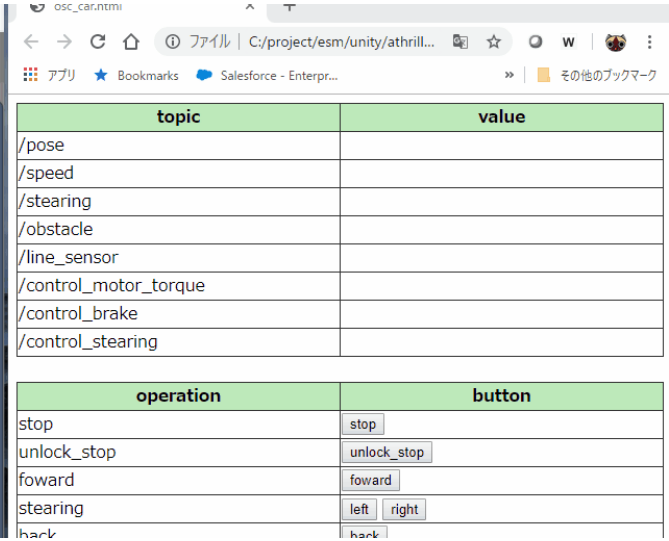
- Athrill/mROSで Unity仮想車両の 制御を机上で デバッグします

- Athrill
  - CPU命令セット シミュレータ
    - ✓V850/RH850
  - リアルタイムOSと ROS/mROSの 対応版も



仮想車両／ROS通信／マイコンの机上デバッグ





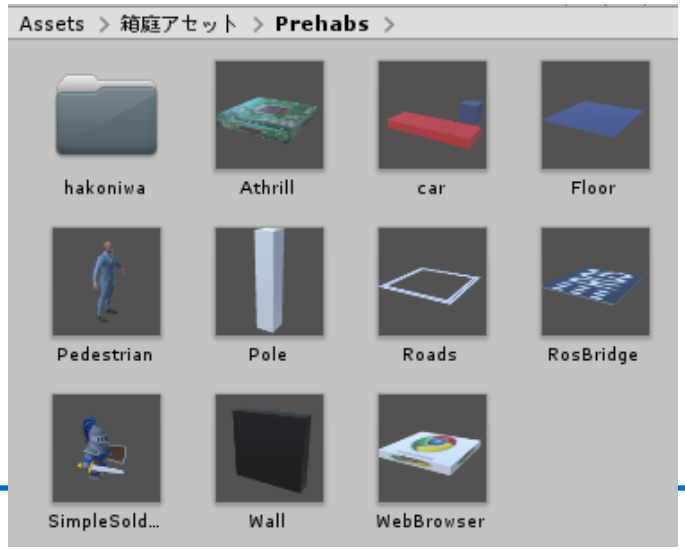
topic	value
/pose	
/speed	
/stearing	
/obstacle	
/line_sensor	
/control_motor_torque	
/control_brake	
/control_stearing	

operation	button
stop	stop
unlock_stop	unlock_stop
foward	foward
stearing	left right
back	back

```
void usr_task1(void)
{
    syslog(LOG_NOTICE,"========Activate user task1========");
    int argc = 0;
    char *argv = NULL;
    int i = 0;
    ros::init(argc,argv,"vehicle_controller");
    ros::NodeHandle n;
    ros::Rate loop_rate(1000);

    car_actuator.motor = n.advertise<std_msgs::String>("control_motor_torque", 1);
    car_actuator.stearing = n.advertise<std_msgs::String>("control_stearing", 1);
    car_actuator.brake = n.advertise<std_msgs::String>("control_brake", 1);
    topic_publish(car_actuator.brake, 1);
```

mROSを使用した
車両制御アプリ

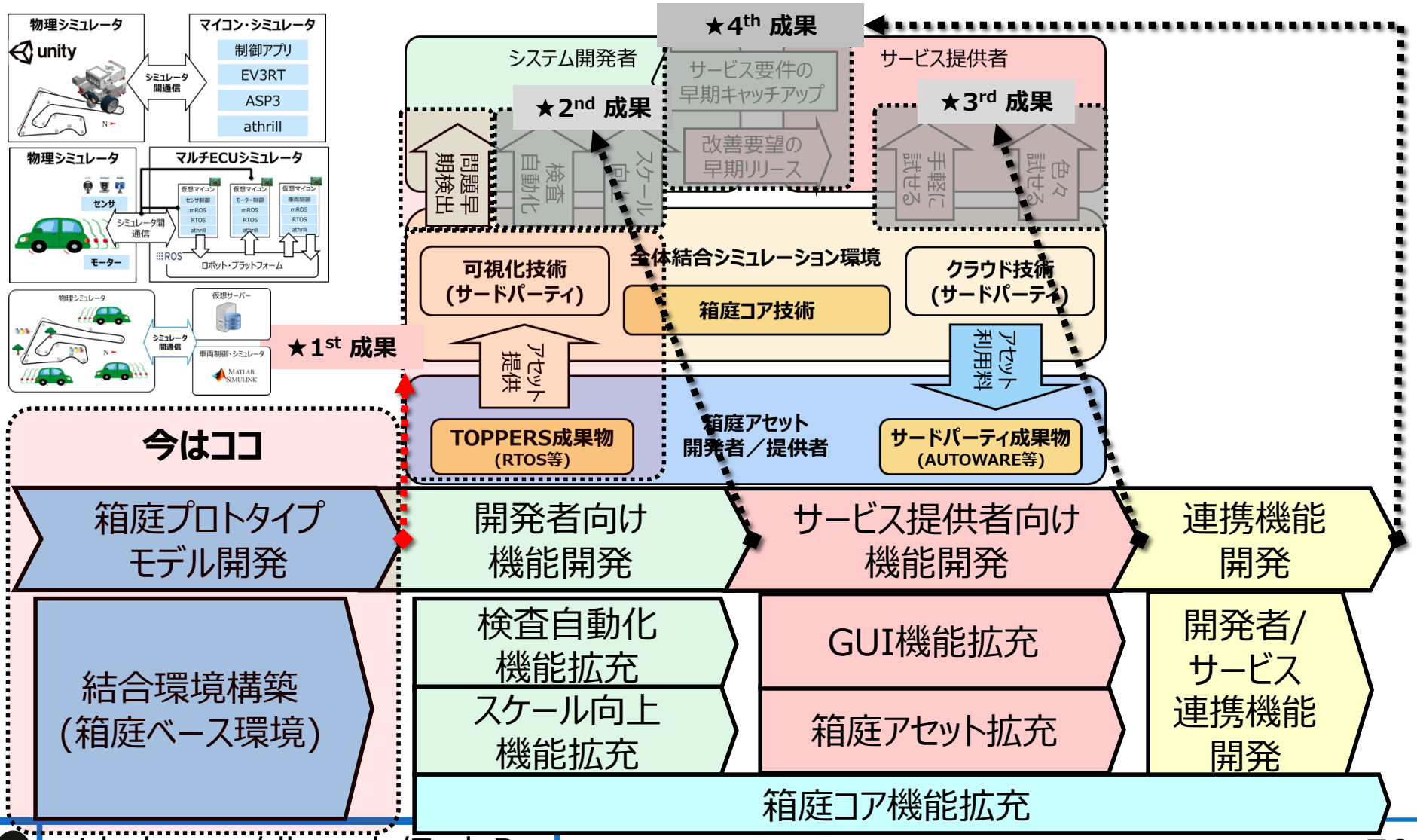


プロトタイプモデル

箱庭コンセプトの実現/技術調査するために,
以下の３つのプロトタイプモデルを構築する予定です.

仮想化レベル	プロトタイプモデル	目的
1,2	A： ETロボコン向けシミュレータ	・技術研鑽 ・広報活動
2	B： ROS・マルチECU向けシミュレータ	・時間同期の仕組み検討 ・mROS/athrill普及
3	C： 車車間協調動作向けシミュレータ	・クラウド連携

全体ロードマップ


物理シミュレータ
unity
シミュレータ間通信
マイコン・シミュレータ
制御アプリ
EV3RT
ASP3
athrill
物理シミュレータ
センサ
モーター
シミュレータ間通信
マルチECUシミュレータ
仮想マイコン
センサ制御
モーター制御
車両制御
mROS
RTOS
athrill
ROS
ロボット・プラットフォーム
物理シミュレータ
仮想サーバー
シミュレータ間通信
車両制御・シミュレータ
MATLAB SIMULINK
★1st 成果
★2nd 成果
★3rd 成果
★4th 成果
システム開発者
サービス提供者
サービス要件の早期キャッチアップ
改善要望の早期リリース
問題早期検出
検査自動化
スケール向上
手軽に試せる
色々試せる
全体結合シミュレーション環境
可視化技術
(サードパーティ)
箱庭コア技術
クラウド技術
(サードパーティ)
アセット提供
アセット利用料
箱庭アセット
開発者／提供者
TOPPERS成果物
(RTOS等)
サードパーティ成果物
(AUTOWARE等)
今はココ
箱庭プロトタイプ
モデル開発
開発者向け
機能開発
サービス提供者向け
機能開発
連携機能
開発
結合環境構築
(箱庭ベース環境)
検査自動化
機能拡充
スケール向上
機能拡充
GUI機能拡充
箱庭アセット拡充
開発者/
サービス
連携機能
開発
箱庭コア機能拡充

協力者募集中！

- **でっかく語って，少しずつ育てております！！**

- 『箱庭』の狙い・趣旨にご賛同いただける方の参画をお待ちしております！！
  - まずはSlackでの議論，活動内容へのご要望，コア技術や各アセットの開発，などに参加したい方
  - 箱庭の活動で期待される技術成果を活用・展開してみたい方

- 現在はTOPPERSプロジェクト傘下のワーキンググループにて活動中
  問合せ先： secretariat@toppers.jp