第22回TOPPERS開発者会議開催レポート

2021年10月24日（日）、25（月）に、オンラインにて、第22回TOPPERS開発者会議を開催しました。TOPPERS開発者会議は、TOPPERSプロジェクトの開発成果物の開発者・利用者が集まり、合宿形式で集中的に議論、開発する会議です。非会員も参加可能です。今年も昨年度に続き、コロナ禍のためオンライン開催となりましたが、参加者数過去最高となり、活発な議論が展開されました。本レポートでは、その魅力をたっぷりお伝えします。

ゲストトーク1

今年は二本立て構成となったゲストトークの一件目として、マイクロソフトコーポレーションの太田寛様に、「IoTを支える技術の'今'と'これから'」と題して講演頂きました。

太田様は、マイクロソフトコーポレーションでIoT関連の技術導入支援や技術普及啓発などの業務に従事されると共に、IoT ALGYAN などのコミュニティ活動でも活躍されています。

講演の前半では、近年のIoT ソリューションの進化の動向を「デジタルツインを活用したコネクテッド環境」実現に向けた流れとして説明して頂きました。また、それを実現するためのマイクロソフトの統合的なIoTプロダクト群とそのアーキテクチャに関して、一貫したアーキテクチャに基づき提供されるサービス部品、OSS、技術群としての「テクノロジーセット」という概念に基づき紹介頂きました。

組込み機器をテクノロジーセットにアラインし連携させることで、各種サービスとの接続、IoT活用サービスでのデータ活用、セキュリティの確保などが容易に実現できます。マイクロソフトでは、組込み機器ハードウェア規模の大小に応じてAzure IoT Edge (Dockerベース)、Azure IoT Device SDK、Azure Embedded C SDK などの様々な接続手段を、オープンソース技術として提供しています。現状、これらの中にTOPPERS成果物を活用したものは含まれていませんが、太田様からは「Azure Sphere のRT Core 部分にTOPPERSのリアルタイムカーネルを搭載してはどうか」という技術提案を頂きました。

講演の後半では、IoT システム開発におけるモデリングの重要性について説明を頂きました。デジタルツイン構築とは、ビジネスの世界と機械の世界のそれぞれのエンティティ間の対応付けを行うことであり、その対象を明確化するためのデータモデリングが重要となります。モデリングを行うには、モデル記述のためのシンタックスやセマンティクス、それを支援するツールなどが必要ですが、ここで太田様が以前から取り組まれていたモデリング技法 Shlaer Mellor 法の知見が大変役に立ったとのことでした。また、Azure Digital Twins プラットフォーム上でのDTDL(デジタルツイン記述言語)による概念モデル記述法やモデリング事例についても分かりやすく解説して頂きました。

講演後の質疑では、関連する話題について活発な議論が行われました。主なものをいくつか紹介します。

・Shlaer Mellor 法の関連情報

→ 日本語での資料は少ないが’80年代の英文文献がamazon 等で購入可能。モデリングツールとしては BridgePoint がある。

・Shlaer Mellor 法とモデルベース開発の関係について
→ Shlaer Mellor 法はモデルベース開発のさきがけの一つ。最近のモデルベース開発は振る舞い記述中心になりがちだが、基盤となる概念モデリングやオントロジー等も重要である。

・DTDL の（ビジュアルな）モデリングツールについて
→ ツインをグラフ表示するexplorerツールは既にある。特定ドメインに特化したツール（例えば地図表示など）はMSでは開発せずサードパーティに委ねる方針。DTDLはオープン規格でありコミュニティベースで開発しているので、そこにツールをコントリビュートするのもよい。

・これまで対応してきたIoT導入支援案件では、IT部署主導、ハードウェア部署主導、どちらが多いか
→ IT部署主導が多く7割くらいを占めるが、現場がIT部署の言うことを聞いてくれず、社内での齟齬によりシステムが不必要に複雑になってしまうケースもあった。広い視野の上の専門性が重要。

なお、本セッションの講演資料は以下のURL で公開されておりますので、ぜひ参照下さい。
「IoTを支える技術の’今’と’これから’」 https://www.slideshare.net/HiroshiOta4/iot-250516781

ゲストトーク2

今年のゲストトークの2件目は、京都マイクロコンピュータ株式会社の河田智明様に、「RustでITRONプログラミング: 今できることと、これからの進化」と題して講演を頂きました。
河田様は、京都マイクロコンピュータ株式会社でRustに対応した組込み開発プラットフォームや動的解析ツールの開発などに携わっておられます。

ソフトウェア開発の永遠の課題として、いかに信頼できるプログラムを経済的に書くことができるか、組込みシステムにおいてもその要求が高まっている中で、最近注目されているRustという言語がこの問題に対するアプローチを提供しており、本講演では「ベアメタルRust」とTOPPERSカーネルを組み合わせ、それぞれの良さを活かすアプローチについての議論、この最初の一歩であるバインディングの設計について検討、さらに、「ベアメタルRust」の壁を超え、オープンソースエコシステムを最大限に活用できるようになるまでの展望が語られました。

本講演は3つのサブセッションから構成されています。
まず最初のサブセッションでは、Rustの言語機能と、APIへの第一歩として「安全でない」ラッパーを示していただきました。

簡単な機能からRustを概観し、メモリ安全に重要な役割を果たすenum型、モジュールやクレート、ビルドプロセスを管理するパッケージマネージャCargoなどについて、続いてRustの独特な機能として、所有権モデル、強力なマクロ、メモリ安全性を保障するためのスライス型や寿命付き参照型などについて紹介されました。

さらにRustにおける標準ライブラリについて、対象環境のレベルに応じて3つの部分（std, alloc, core）の3つの部分に分割されている点や、coreだけを使用したRustはベアメタルRustと呼ばれることが説明されました。

そして、最初のアプリケーションとして、このcore上で動作するアプリケーション、RustからカーネルAPIを呼び出すラッパーの具体的な形を示し、この関数（ラッパー）がメモリ安全でなく、この関数を呼びだすにはunsafeブロック内からでないといけないことが説明されました。

次のサブセッションは、Rustの「安全」の意味についての話でした。

安全とは、究極的にはプログラムで定義された動作のみが行われることを意味する。そのためにはRustのような低レベル言語ではメモリ安全が必須であり、Rustでは、安全でない動作を分離し、コンポーネントに封じ込め、コンポーネントを信頼することによってメモリ安全な”世界”を拡張できる、ということでした。

そして、安全な高レベル抽象化の例としてスライス参照が示され、前のサブセッションで紹介された安全でないラッパーを安全なラッパーにするやり方も具体的に示されました。もう一つの有用なアイデアであるI/O Safety、これはRustの所有権モデルをカーネルのオブジェクトに適用する考え方についても具体的なお話がありました。

RTOSカーネルをRustで書くもしくは書き直すことについて、簡単な話ではないが、一番低レベルな操作だけをunsafe Rustで書いて、残りをsafe Rustで書くことが出来る、と述べられました。カーネルのユーザーから見た利点や不利な点についても語られ、ソフトウエアを新しい言語で書くという議論は、他の言語との協調や新しく入ってきた開発者のdeveloper experienceなど長期的な目線や幅広い視野を見失わないことが重要であると語られました。

最後のサブセッションは、ソフトウェアの再利用性に関するお話でした。

コードの増大によって、ソフトウェアの再利用性の重要度が増していますが、活用されているオープンソースライブラリとの統合は手間がかかり問題となっています。この問題を回避するパッケージマネージャーと協調するように、最初からRustは設計されていた、ということです。

しかし、残念ながらベアメタルRustは、標準ライブラリstdを使うことは出来ず、stdに依存しないパッケージは全体の4.39%しかない。この分断は、同じ機能を持つパッケージが複数存在する「エコシステムの断片化」につながる。この断片化のメリットとデメリットを踏まえて、ITRONアプリケーションのためのRustのエコシステムを構築するためにはどうするか、3つの選択肢が示されました。その3つとは、「何もしない」「カーネルAPIのラッパーパッケージをリリースし、これをベースとしたエコシステムを育てる」「Rustのstdライブラリにサポートを追加しエコシステムを統一する」でした。

2番目については、最近、itronパッケージがリリースされ、このアプローチの実証を行っています。3番目については、SOLIDプラットフォーム上にある程度実装したコードがRust標準ライブラリに入っているということでした。また、この道を選択するには、カーネルだけでなくプラットフォームという目線で考える必要があり、これを効率よく実

装するためにはカーネルに今存在する機能に加えて他のOSスタンダードの互換性を配慮した機能が提供されることが必要とのお話でした。

最後に、Rustの改善が望まれる点として標準ライブラリの肥大化を防ぐためにRustのソースリポジトリの外側で新しいターゲットのサポートを追加する仕組みが望まれているというお話でした。

Rustについて、より詳しく学びたい人のために、以下が紹介されていました。

・The Rust Programming Language("The Book") : https://doc.rust-lang.org/stable/book/ (en)
https://doc.rust-jp.rs/book-ja/ (ja)

・The Rust Reference : https://doc.rust-lang.org/stable/reference

・This Week in Rust : https://this-week-in-rust.org/

ともかく、Rustの特徴に始まり、ベアメタルRust、Rustの「安全」の意味とRTOSのAPIラッパーにおける安全性の実装、スケーラブルなソフトウェア再利用の実現などについて、高度な内容を凝縮して講演していただきました。ありがとうございました。

Rustから使う！TOPPERS/ASP

準会員の杉本様からRustで書いたアプリケーションをASPカーネルと組み合わせて動かすための手法について解説していただきました。

Rustで記述したアプリからASPカーネルのAPIを呼び出せるようにするためのRust用ラッパーの開発を行い、sample1プログラムを動作確認したとのことです。

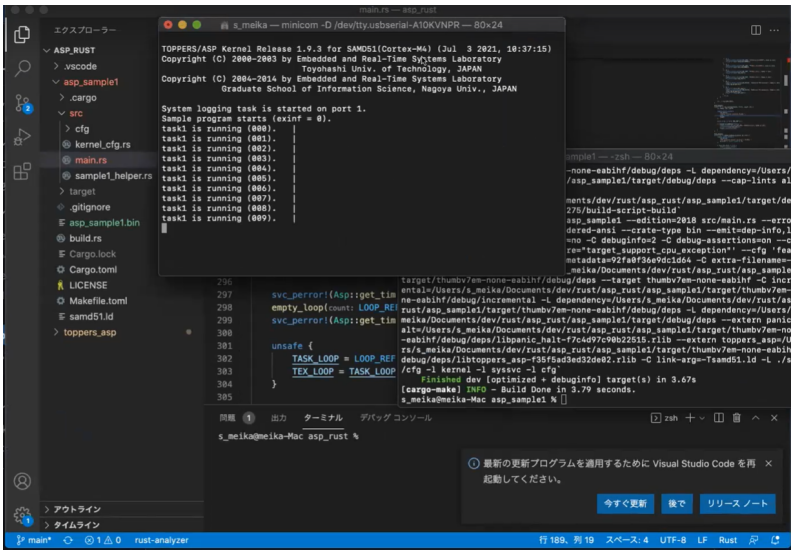

図1: sample1の実行

このラッパーを使うことで、アプリケーションはモダンなRustで書きつつ、OS側はTOPPERSをそのまま使え信頼性を確保できるとのことでした。
ターゲットはWioTerminalをこのために購入されたそうで、既にTOPPERSがターゲットで動いていたおかげで3週間ほどでラッパー開発とRustアプリでの動作確認ができたそうです。
　いくつか工夫された点の報告もありました。OSのAPIの引数にポインタがある場合でポインタを使わなくても良いケース（例えばget_priのタスクのベース優先度）ではタプルを用いてポインタを隠蔽することでAPIの引数を安全に利用といった紹介がありました。
Rustを使ってみて、volatileをRustで記述するのが大変とか、char*型が扱いづらいなどと言ったC言語プログラマーあるあるも披露していただき、体験を踏まえた感想を交えて分かりやすく説明をしてもらいました。
今回開発されたラッパーコードはGitHubにて公開されておりQiitaにて解説記事もありますので、TOPPERSカーネル上でRustを手軽に試してみようかなと思う方は試してみるのはいかがでしょうか。

https://github.com/s-meika/toppers_asp
https://qiita.com/s_meika/items/a60f7be7f3c4044dc9b4

Zig言語によるASP3カーネルの実装

高田会長から、Zig言語(https://ziglang.org/)によりASP3カーネルを実装したことについて発表して頂きました。
昨年、コロナ禍の影響で出張がなくなり、出張に使っていた時間を新しいことへの挑戦に使い、C言語以外の言語によるASP3カーネルの実装を試みたとのことでした。

・Zigを選んだ理由
1.C言語を現代風に改良・拡張したものであること
2.コンパイル時のコード実行機能(comptime)により静的APIを置き換えられる可能性を感じた
3.すべての資源を明示的に管理できることがRTOSの記述に向いている
Rustを選択しなかったのは、Rustをよく知らなかったため、習得するまでの学習コストが高いと判断したため、またヒープが勝手に使われてしまうようだとの懸念があったためだそうです。
Zigは比較的シンプルな文法(C言語よりは複雑だが、C++と比べるとはるかにシンプル)であり、C言語との共存が容易になる機能もあるとのことでした。

・OSにおけるメモリ管理
手動でメモリ管理ができる(言語が裏でこっそりメモリを使わない)ため、安心感がある。
Rustでヒープが使われるという懸念に関しては、「標準のcrate(C言語での標準ライブラリに相当)の実装がヒープを使うものが多いためで、heaplessのcrateを指定すれば回避できるのではないか」という参加者からの指摘がありました。

・コンパイル時のコード実行(comptime)と最適化による静的APIの置き換え
TOPPERSのOSは静的OS(設計時に決まるものはランタイムではやらない)である。例えば何個の、どういう属性のタスクを用いるか設計し、その情報を静的APIで記述し、ツールを用いて*.c、*.hに変換後、その他のCソースファイルと一緒にコンパイル、リンクしていた。
Zig言語ではZigの文法で記述でき、最適化とcomptimeにより、コンパイルだけで実現できた。

・現在の実装状況
1.TOPPERS/ASP3カーネル(ARM向け)のカーネル本体をZigで実装。libkernel.aの全体をZig+インラインアセンブラで記述(*.c,*.Sファイルはなくなった)
2.システムコンフィギュレーションファイル(静的API)とコンフィギュレータ(Ruby)もZigで実装(パス3は除く)
3.アプリケーションとシステムサービス(C言語+TECS)は修正なし(Zigで記述することも可能)
4.ビルドシステム(Makefile)には修正を加えた(ここもZigにしたいが、出来ていない)

・Zigの課題
未完成の言語である(未実装の機能、制限事項が多数、言語処理系の不具合も多い、急な言語仕様の変更により、コードが動かなくなる)、ドキュメントの不足、エラーメッセージが不親切、ターゲットプロセッサが限られる(LLVMが対応していることが必要条件)などがある。
現在も開発版のRelease 0.9.0で@importの仕様が変更(パラメータが単純文字列に限定された)、インラインアセンブラの仕様変更(単純文字列に限定された？)などが存在する。対応方法のトレードオフの検討にとどまらず、解決方法の模索が必要なものもある。

ソースコードは、GithubのTOPPERSオーガナイゼーションで公開されています。
https://github.com/toppers/asp3_in_zig
Zigを用いたASP3カーネル実装のより詳しい話はSWEST22の「Zxx言語によるリアルタイムOSの実現」を参照して下さい。
https://swest.toppers.jp/SWEST22/program/s3a.html#s3

パネルディスカッション＆LT

Rust と RTOS に関連するテーマで、6名の方々にライトニングトーク形式で話題提供を頂きました。その後、午後のRust関連講演の質疑も兼ねて全体議論を行いました。

Rust の **HAL**（**avr-hal**）で **Leafony** を動かした件（阿部耕二様）
Leafony という小型マイコン基板上で、C言語のサンプルコードをRustに移植した事例について紹介を頂きました。Leafony は、Arduino UNOなどと同じAVRマイコンを搭載したマイコンで、avr-hal というArduino向けクレートを

使い、Leafony向けに若干の修正（クロック周波数の変更など）を加えることで、Rustによる開発が実現できたところです。

Rust で **OS** を実装する（ガラスボー様）

実際にRust による組込みOS開発を試みた経験や他グループの研究事例に基づき、Rust によるOS実装について紹介を頂きました。

Unsafe を多用するとOS記述は可能だがRustの利点が失われるため、unsafe操作箇所を限定しそれ以外との部分のインターフェースを適切に設計する事が重要、との事でした。また、TOPPERS で Rust を採用する場合に取り得る選択肢として、OS全体を（unsafeも使い）Rust で実装する、カーネル内モジュールの一部をRustで実装する、アプリ側をRustで記述する、Rustを前提として設計を全面的に見直す、などの案を提示頂きました。

ベアメタル向け **Rust std** クレート実装調査（井田健太様）

ベアメタル環境向けの std クレート実装の現状について、情報提供を頂きました。Rust では、stdクレートに依存しない環境をベアメタルと呼んでいますが、ネットワーク、ファイル操作、スレッドなどの主要機能が軒並みstdクレートに依存しているため大きな制約が生じます。

そこで、既にstdクレート実装が提供されているESP32マイコンの事例に基づき、ベアメタル環境でのstdクレート実装の実現方法について紹介を頂きました。Std クレートの内部構造を分析したところ機種依存部分はlibc, socket, pthread など少数のクレートにまとまっており、これらの実装を動作環境向けに用意することで比較的小さな実装量でstdクレート実装が可能になるのではないか、とのことでした。

Rust で **RTOS** を考える（渕上竜司様）

自作のITRON実装（HOS）をZynq MP マイコンボードに移植し、その上でRustを使用した事例について紹介を頂きました。また、Cを使わずにRust でRTOSを実装する場合の設計方針の考察について紹介頂きました。ID ベースのオブジェクト管理ではRustの所有権システムを活用できないためRustらしい代替方法が必要であること、など幾つかの課題が提起されました。また、Rust活用の方向性の一つとして、ITRONの設計思想のみを受け継いだRust前提のRTOSを開発する、という提案を頂きました。

組込み**Rust**について思うこと ～それでも**Rust**で**RTOS**を開発しますか～（加藤丈治様）

Rust によるOS開発を具体的に検討した経験から、RustでOSを書くことの利点と欠点、および利点を活用する方策について提言を頂きました。

x64 Linux 環境で Rust によるタスクディスパッチャを実装した際の感想としては、アセンブラによるスタック操作が主で、Rustで書くメリットを出しにくかったそうです。また、Linux カーネル開発におけるRust導入検討の議論を踏まえて検討すると、

・ファイルシステムなど、品質確保が難しく複雑なモジュール
・ドライバやHALなど、変化が早く迅速な対応が必要な部分

などがRustの適用に向いている、とのことでした。また、Rust利用に適したOSアーキテクチャとして、マイクロカーネル（サーバ部分にRustを使う）やナノカーネル（資源管理をアプリ層に委譲）が適しているのではないか、との提言を頂きました。

SOLID Rust対応機能のデモ（京都マイクロコンピュータ 辻　邦彦様）

京都マイクロコンピュータ㈱の組込みソフトウェア開発プラットフォームSOLIDで新たにサポートされたRust対応機能について、実機デモを交えて解説して頂きました。Rustコードのコンパイルはもちろん、IDE上でのデバッグ機能もRust対応となっており、CとRustコードにまたがる部分のトレース実行などを実演して頂きました。

プレゼンの後に、これまでのRust関連の講演全体に対する質疑、および自由討論を行いました。TOPPERSとしての今後のRust対応の取組みについては、itronクレートのようなものを公式に保守していくことが必要であり、TOPPERS のWGとしての活動を立ち上げてはどうか、との議論がありました。また、何人かの講演者からも言及があった、既存のITRON C言語APIにとらわれないRust 前提の新しいRTOSの開発についても、技術的な実装方法まで踏み込んだ活発な議論が展開されました。最後に、本日の議論の総括として河田様、杉本様から「組込み開発の生産性向上に向けて、Rustなどの新しい技術の導入をTOPPERSが中心となって進めていくべきではないか」との提言を頂きました。

テスト手法Fuzzingの研究動向と実開発への適用

名古屋大学の伊藤弘將様からFuzzingについてご説明いただきました。
Fuzzingはソフトウェアテストの手法の一つで、最近注目が集まっていますが古くから存在する手法ということや、注目されることとなった出来事など、Fuzzingの歴史について紹介がありました。
Fuzzingはランダムデータをプログラムに入力して脆弱性を見つけ出すテスト手法ですが、単純なランダムデータを入力するものから、実行時の情報を使ってランダムデータを変化させるもの、ソースコードを解析して入力データを作成するものなど、いくつかの種類があり、それぞれの特徴についての説明がありました。
Fuzzingは実際に使われている数多くのソフトウェアに対して行われ、多くの脆弱性が実際に見つかっているようです。
伊藤様の研究では、すでにFuzzingが行われているlwIPについて、ご自身が考案した手法でFuzzingを行い、新たな脆弱性を見つける成果が得られたようです。
組み込みソフト開発への適用についても説明があり、開発工程に組み込むためのCI/CDツールの紹介や、開発工程への適用に対する課題についても説明していただきました。
このように議論の前半ではFuzzingに関する全体像が把握できるまとまった内容で、有用性が感じられる講演をして頂きました。今後の組込みソフト開発にFuzzingを適用することにより、品質向上が期待できると思いました。
議論の後半では、ホームネットワーキングWGの取り組みとしてTOPPERSで提供しているTCP/IPプロトコルスタックのTINETに対して、Fuzzingの成果として、テスト環境や見つかった脆弱性の種類の紹介がありました。

TOPPERSでROS 2ノードを稼働するための一提案

当初この時間枠では「時間パーティショニング」に関するセッションを実施予定で、その次に「TOPPERSでROS2ノードを稼働するための一提案」というタイトルで、東京大学の高瀬先生より30分の枠で本内容を紹介いただく予定でした。しかし、後者の内容から派生した議論が開発者会議向けに設置されたSlackで盛り上がったため、「TOPPERSカーネルの上周りと下周りについて」という内容を加えて、90分の拡大セッションでの議論となりました。前者の「時間パーティショニング」に関する内容は、別の機会で実施する予定です。

最初の話題として高瀬先生よりmROS2を作成するにあたって既存のライブラリやミドルウェアをTOPPERSカーネルとどうやって共存させるかという話でROSの通信方式の説明とROS2の階層構造、mROSの最新動向について説明がありました。

組込み向けのROSにはmicro-ROSがありますが、エージェントを経由するためmROS2ではエージェントレスの実装としてembeddedRTPSというDDSライブラリを採用することで高速化、高リアルタイム性を実現できたとのことでした。

ただembeddedRTPSはlwIPやFreeRTOS(CMSIS OS API)に依存しているため、ASP3カーネルから利用するための実装コストがかなりかかってしまったとのことでした。

mROS2の発表(ROS 2 Client Library for E^2)の詳細ついては箱庭WGのサイトから参照ください。

https://toppers.github.io/hakoniwa/technical-links/#roscon-jp-2021

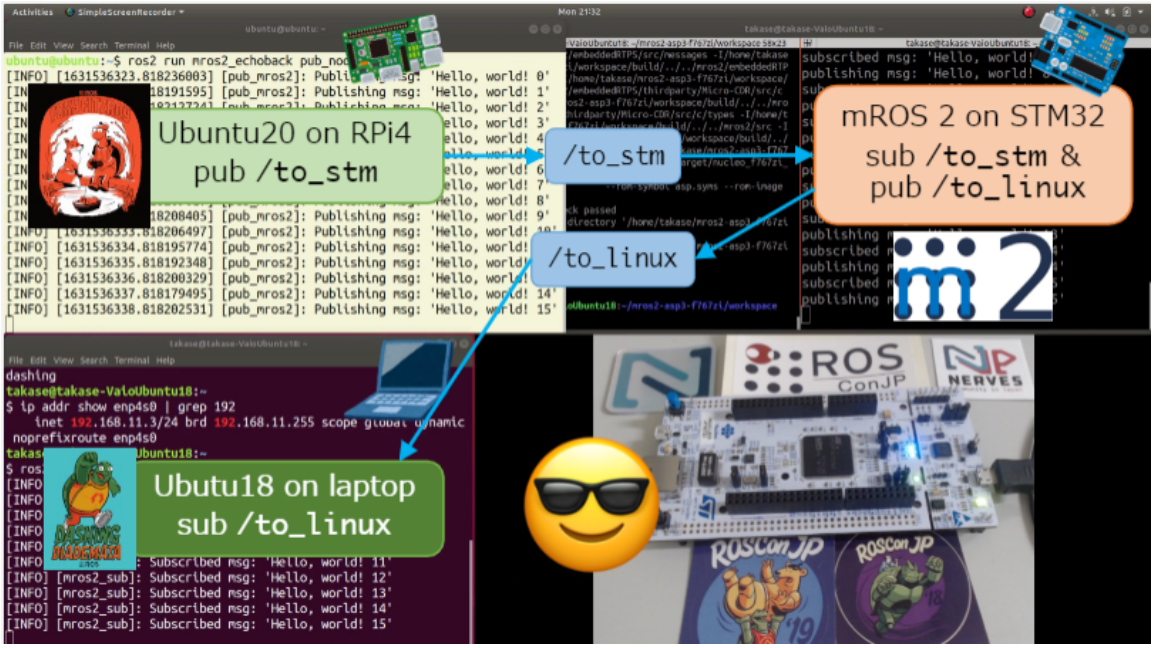

図2:STM32にmROS2を実装しPub/Sub 通信を行っている

そこでベンダーツール(例えばSTM社のCubeMX)へのロックインは許容するとして、TOPPERS側がベンダーツールの出力コードを利用できると

1. ベンダー対応の任意のボードを決めて
2. ベンダーツールでプロジェクトを作成し、pin I/Oを設定し
3. TOPPERSの対応コードを展開し、コード修正などは行わずに
4. ビルド、Flash書き込み、デバッグがベンダーツールで行う

というツールチェーン環境ができるようになってほしいとのことでした。

後半からはTOPPERS/ASPのCortex-M3ポーティングをいち早く行われた堀江様も加わり、「TOPEPRS/ASPへの提言」ということで最近のマイコン・ユーザ視点でのTOPEPRSカーネルについての発表がありました。
最近のワンチップマイコンのエコシステムではピンが多くかなり多重化されていたり、クロックツリーと呼ばれるデバイスへのクロック供給設定を行う必要が多数あるため、メーカ提供の設定ツールを使わないとソフトウェア開発ができないのが実情とのことです。
また設定ツールでは、そのほかの設定としてCPUのパラメータ、ペリフェラルや各種ミドルウェアの設定もできるようになっており、RTOSの利用もツールで設定できるようになっているため、ポーティングコストなどを考えると現在のTOPPERSカーネルを利用するメリットが感じられない人が多いのではないかとのことでした。
ベンダー設定ツールからの生成コードをTOPPERS/ASPで使えるように手動で変更するのは割込みハンドラの設定をすべて書き換える必要があり、コストと開発効率・信頼性の低下の面からもお勧めでなくディスパッチャなどの性能が落ちることを許容しても、ベンダー設定ツールの生成コードにTOPPERS側が対応したターゲット依存部を新たに作成したほうがよいという内容でした。
具体的な提言として、

- main関数からkernel_start()を呼び出す
- 生成された割り込みハンドラコードを使用する
- ディスパッチの中身を割込みとして実装する
- 遅延割込み機能が無いCPU向けにret_int()を提供する

といった対応をTOPPERSカーネル側が行うことで、ベンダーの設定ツールを含むエコシステムの中でTOPPERSカーネルがより使いやすくなるというご意見でした。
その後も活発な議論が飛び交いながら時間となりましたが、開発者会議用のチャットルームでも引き続き具体的な対応についての議論もあり、何かしらの成果がでてくれるのではと期待をする議論となりました。

TECS WG の最新研究

「ECHONET Liteの通信部とデバイス部をつなぐTECSコンポーネントおよびインタフェースコードの自動生成」斉峰様（埼玉大学）

IoTデバイスソフトウェアは、機器種類と機能が複雑化し、ソフトウェアのメンテナンス性が低くなっているので
TECS(TOPPERS Embedded Component System)を用い、メンテナンス性、再利用性の向上を目指したそうです。

HEMS(Home Energy Management System)向けの通信プロトコルであるECHONET Liteを対象とし、ECHONET Liteデバイス記述(*.json)を入力、以下の(1),(2)を出力とするECNLコードジェネレータ、ECHONET Liteミドルウェア・コンポーネントよりなるフレームワークを提案されました。
(1)デバイスをコンポーネント化するコンポーネント記述(*.cdl)
(2)ECHONET Liteミドルウェア・コンポーネントを呼び出すスマートホームソフトウェアのテンプレート
ECNLコードジェネレータに対し、必要なデバイス情報を選択し、アクセスルールによってプロパティを指定して、自動生成するそうです。
さらに、クライアント側でのスマートホームソフトウェアのプログラミングをビジュアルプログラミングで実現し、フレームワークを拡張されました。Googleが開発したBlockly(Block-based visual programming tool)を採用し、ECNLコードジェネレータに、Blockly上で利用できるブロックのインタフェース(*.js)を生成させたとのことでした。

「TECSコンポーネントからmrubyスクリプトを呼び出すフレームワーク」下村遼太様（埼玉大学）

組込みシステムの規模・複雑性の増加に対し、機能を部品(コンポーネント)として開発し、コンポーネントの追加/削除を容易にして、高い生産性、再利用性が実現できるコンポーネントベース開発(CBD)に取り組んでおり、TECSは組込みシステムに適したC言語ベースのコンポーネントシステムだとのことでした。
言語の特性を考慮し、アプリケーションの核をC言語、リアルタイム性が要求されないコンポーネントをスクリプト言語で記述する場合、TECSにおいてmruby(https://mruby.org/)スクリプトでTECSコポーネントを操作することは実現しているが、C言語からスクリプト言語を呼び出すことはまだ実現していないため、事前準備したControlコンポーネントとmruby VMコンポーネント、TECSジェネレータのTECS/mruby連携プラグインで生成したBridgeコンポーネントを利用して、C言語からmrubyスクリプトで定義されたメソッドを呼び出すフレームワークを提案されました。
Controlコンポーネントがmruby VMコンポーネントにmruby VMを起動させ(この時mruby VMはCDL上で指定されたmrubyスクリプトを読み込む)、Bridgeコンポーネントを経由して、mruby VMコンポーネントにmrubyのメソッド呼び出しを要求し、mruby VMコンポーネントが自身の管理するmruby VMに対してmrubyのメソッド呼び出しを行うそうです。
LEGO Mindstorms EV3に対して動作検証を行い、C言語からmrubyスクリプトで定義されたメソッドの呼び出しができることを確認されたとのことでした。
今後はコンポーネントの振る舞いの動的な変更を実現したいと言われていました。

TOPPERS基礎実装通信講座開始

教育WGで作成している、新しい教材についての説明がありました。
教育WGでは例年セミナーを開催し、教材の基板や電子部品などを使って、組込み開発の基礎を学べるコンテンツを用意していましたが、昨年や今年は新型コロナの影響で開催を見送ってきました。

そこで、セミナーの内容を動画として提供する方針に変え、通信講座として提供出来るようなコンテンツを作ることにしました。
受講は基本的に自己学習で行います。まず受講のためのボードセットを購入してもらい、自宅のパソコンに開発環境を構築し、受講用のプロトタイピングシールドを実装します。(半田付け不要)資料やソースコードをダウンロードし動画を見ながら自分で学習します。
その後、学習の結果を事務局に送ってもらい、教育WGで認定出来たら、修了証を発行する流れになると説明がありました。
また、受講開始3ヶ月間は質問を受け付けるようにし、それらが蓄積されてFAQサイトとして利用が可能となる予定です。
受講のためのボードセットは、STM32F401 nucleo-64とプロトタイピングシールド(TEB001)、LCDとジョイスティックが付いたシールド(TEB002)、その他部品セットで構成され、受講費込みの2万円を想定しています。
TOPPERS BASE_PLATFORMには、実際に使えるアプリの例としてリファレンスアプリが用意され、一つはMedia Playerで、MP3の演奏やJPEGの表示、USBのデバイスとホストが学べる内容で、もう一つは、今年開発したBLE REMOTEで、プラモデルの戦車をBLEでAndroidから操作するもので、BLE通信やモータ制御が学べる内容となっています。
いずれも、最新のTOPPERS BASE PLATFORMリリースからソース等をダウンロード可能です。
以上の通信講座用コンテンツについてのデモが行われ、戦車をAndroidから操作したり、動画コンテンツの一部を紹介して頂きました。
通信講座は作成中で2021年12月中の開設を目指しているとのことです。

ハンズオン1：Raspberry Pi Pico でのTOPPERS第3世代RTOSの実行

名古屋大学の小森工様から、Raspberry Pi Picoで、TOPPERS/ASP3またはTOPPERS/FMP3のサンプルプログラムを動かすハンズオンを行って頂きました。
下記のQiitaの記事の手順で、2台持っている人はSWDを使ったデバッグ環境で実行する手順、1台の人は単独で動作を行うための手順を、参加者の状況に合わせて進めていきました。
参加者の持つRaspberry Pi Picoの個体差によって準備した手順で動かないことがありましたが、その場で解決して動かすことが出来ました。
https://qiita.com/komori-t/items/784dcfdfeaa1d9732169

ハンズオン2：mruby+TECS on TOPPERS BASE PLATFORM

オークマの大山 博司様からTECS を使って mruby を TOPPERS BASE PLATFORM に移植するハンズオンを行っていただきました。
対象はTOPPERS BASE PLATFORM が動作するボード(SMT32F746 Discovery Kitなど)です。
TOPPERS/ASP をビルドしボードに書き込む手段(例：PCとUSBケーブル、ST-Link Utility(ソフトウェア))が必要です。

具体的な手順、ソースコードは以下のGithub上のリポジトリに公開されています。
https://github.com/hiro22022/mruby-TECS_on_TOPPERS_BASE_PLATFORM
TECSが組み込まれていないTOPPERS/ASPカーネルに対して、TECSを組み込むための手順(ソースファイルの追加、Makefileの修正、CDLファイルの修正など)を、一つ一つ進めていきました。
CDLファイルの修正の際に異なる文字コードの混入によるトラブルが発生し、TECSジェネレータによるデバッグを見ていただく事態にもなりました。最後は駆け足になってしまいましたが、予定の内容を一通り終えることが出来ました。

2021年度開発者会議を振り返って

オンライン開催も2回目となり、例年の3倍近い参加者による活発な議論が行われました。オンラインイベントに慣れている参加者の方も多く、そのような方たちを中心にSlackやZoomのチャットでの議論も活発にできたように思います。今年も遠方に在住の方や時間に制約のある方にも柔軟に参加いただくことができ、オンラインならではの良さも認識できました。

一方で、合宿ならでの良さもあり、来年こそは合宿を再開できることを期待しています。

開発者会議2021実行委員会

山根ゆりえ（特別会員/実行委員長）伊与田健敏（創価大学）岡山直樹（アイシン・ソフトウェア(株)）
小川清（個人会員）小南靖雄（個人会員）高田光隆（名古屋大学）長島宏明（コアーズ(株)）
堀武司（個人会員）松原豊（名古屋大学）

NPO法人 **TOPPERS**プロジェクト
〒103-0011 東京都中央区日本橋大伝馬町6-7　住長第2ビル3F
（一社）組込みシステム技術協会内
TEL & FAX: 03-5643-5166　Email: devconf@toppers.jp
Facebook：http://www.facebook.com/toppersproject