ANALOG DEVICES
想像を超える可能性を
AHEAD OF WHAT'S POSSIBLE™

オープンソースの再利用可能ソフトウェア・スタックがCbMのリアルタイム処理とアルゴリズム開発を実現

著者：Travis Collins、シニア・アルゴリズム・エンジニア

回路ノートCN0549の概要

この記事では、CN0549の様々なコンポーネントで使用できるソフトウェア・エコシステム、データ分析ツール、およびソフトウェア統合機能に焦点を当て、エンジニアとデータ・サイエンティストが、アプリケーション開発においてそれらをどのように利用できるのかについて述べます。これは、CN0549開発プラットフォームを使用する状態基準保全（CbM）アプリケーションと予防メンテナンス（PbM）アプリケーションに関する2つの記事で構成されるシリーズ記事の2番目にあたります。この新しいプラットフォームは、プロトタイプから生産へと続くカスタムCbMソリューションの開発フローを加速できるように設計されています。パート1は、MEMSを利用した振動モニタリング技術と、CbMアプリケーションのための高品質振動データの収集に焦点を当てています。

生産までの道のりとその短縮方法

状態監視ソリューションを実現するには、センサー、ローカル処理、ネットワーク接続に加えて、これらすべてを機能させる何らかのソフトウェアやファームウェアが必要です。CN0549は、ハードウェアとソフトウェア両方の側面に対しカスタマイズ可能なオプションを提供することによってこれらすべての課題を解決できるため、エンジニアやソフトウェア開発者は、共通のツールやインフラストラクチャを使いながら、アプリケーション設計のためのトレードオフを行うことができます。例えば、処理に特定マイクロコントローラやFPGAを使いたい場合や、Pythonコーディングを使いたい場合、あるいは好みのセンサーがあって、それを再利用したい場合などです。この特長により、処理、電力、性能、ソフトウェア、データ分析を必要に応じてカスタマイズできる最適化されたCbMソリューションの構築を目指す技術者にとって、CN0549は強力なプラットフォームとなります。

組込みシステムの開発プロセス

概念段階から生産段階まで、組込みシステムの一般的な開発フローを考えてみましょう。フローを抽象化した全体的な概要を図1に示します。

図1に示す設計プロセスの最初のステップは、データ調査フェーズです。このフェーズでは、アプリケーションに求められる様々なハードウェア条件とソフトウェア条件にユーザの要求事項を割り振ります。ハードウェア的な視点からすると、これらには衝撃許容値、アナログ信号帯域幅、あるいは測定範囲などのパラメータがあります。ソフトウェアの条件を考えた場合は、サンプル数、サンプル・レート、周波数スペクトラム、オーバーサンプリング、およびデジタル・フィルタリングなどが、CbMアプリケーションにとって重要なパラメータとなります。このプラットフォームは非常に便利かつ柔軟なもので、データ調査を行う技術者は、様々なセンサーを組み合わせることにより、独自のアプリケーション・ニーズに合わせてデータ・アクイジション・パラメータを調整することができます。

データ調査フェーズに続くのがアルゴリズム開発フェーズです。ここでは、そのシステムを適用あるいは使用することが可能であることを確認します。通常、これには高水準ツールを使ったモデルの作成やアルゴリズムの開発が含まれ、最終的にはこれらのモデルやアルゴリズムが組込みシステムに移植されます。設

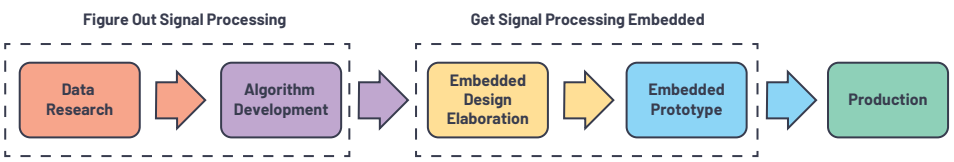

図1. 組込みシステムの開発フロー

計を最適化する前に、実際のデータとハードウェア・イン・ザ・ループを使ってその妥当性を確認する必要がありますが、ここでCN0549が非常に優れた機能を発揮します。これは、CN0549が一般的な各種の高水準分析ツールを直接統合化できるだけでなく、ハードウェア・イン・ザ・ループによる妥当性確認を行うこともできるからです。

この設計確認が完了すると、必要なソフトウェア・コンポーネントを最適化して組み込むための作業が開始されます。組込み設計の精密調整フェーズでは、FPGAやリソースに制限のあるマイクロコントローラ上で機能させるために、特定のアルゴリズムやソフトウェア層を再実装しなければならないことがあります。この段階の設計は最終検証用のプロトタイプや量産品に近いハードウェアに組み込まれるものなので、細心の注意を払って継続的に確認する必要があります。

最後に来るのが生産フェーズで、この段階では設計開始当初に使用した元々の開発環境と共通する点はほとんどなくなりますが、やはり同じ要求事項を満たす必要があります。最終的なシステムは当初の調査システムとは全く異なるものになっている可能性が高いので、多くの場合、同じコードやテストを実行することは不可能か、あるいは極めて困難です。そのために生産テストに関わる問題や装置の不具合が発生して、その是正に追加的な時間や予算が必要になることもあります。

再利用を増やすことでリスクを軽減

設計プロセスのリスクを軽減する最も簡単な方法の1つは、各段階を通じ、できるだけ多くのハードウェア・コンポーネントとソフトウェア・コンポーネントを再利用することです。CN0549には、このような場合にそのまますぐに利用できるリソースが数多く含まれており、開発者は開発フローのすべての段階でそれらを直接利用することができます。CN0549のリファレンス設計は、ボードを適切に設計するための回路図／部品表／レイアウト・ファイルと、最適化されたフル機能環境のためのオープンソース・ソフトウェア・スタックを備えており、MATLAB®やPythonなどの高水準ツールとの統合化が可能です。エンド・ユーザは、妥当性を確認済みのアナログ・デバイセズ製コンポーネントを利用し、調査段階から生産段階へと移行していくのに合わせて、維持あるいは変更したいコンポーネントを選ぶことができます。また、これによりエンド・ユーザは、回路図にアナログ・デバイセズ製部品を組み込んだり、ソフトウェアをゼロから開発したりといった作業ではなく、アルゴリズム開発とシステム・インテグレーションに集中することができます。ハードウェア・モジュールの利用や、アナログ・デバイセズの提供するデバイス・ドライバやHDL、あるいはアプリケーション・ファームウェアといったソフトウェア層の再利用は、システムの構築に必要な開発時間を短縮し、製品の市場投入を劇的に加速します。

ソフトウェア開発のフローとプロセス

CN0549は開発時に無数とも言える選択肢をエンジニアに提供して、MATLABやPythonなどの使い慣れたデータ分析ツールを使用しながら、CやC++などの共通言語を使って作業することを可能にします。これは主に、オープン標準や、様々なメーカーが提供する複数の組込みプラットフォームをサポートする既存ソリューションを利用し、それらに基づいてシステムを構築することによって実現されます。

CN0549のシステム・スタック

図2に示すシステム・スタックは、CN0549システムを構成する様々なコンポーネントの基本的な概要を示しています。左上にあるダーク・ブルーのボックスはセンサーとデータ・アクイジション（DAQ）ボード、ライト・ブルーとパープルのボックスはデータ処理に使うFPGAパーティションの概要です。このプラットフォームはIntel DE10-NanoとXilinx® CoraZ7-07sを直接サポートしており、2つの主要FPGAベンダーのどちらにも対応しています。グリーンのボックスはホストPCとの接続を表しています。これにより、ハードウェアからアルゴリズム開発用高水準データ分析ツールへの、直接データ・アクセスが可能になります。

すべてのハードウェア記述言語（HDL）コードはオープンソースで、開発者はこれに変更を加えて、図2に示すように、プログラマブル・ロジック（PL）内のデータ・ストリームにデジタル信号処理（DSP）を挿入することができます。挿入できるものには、フィルタからステート・マシン、更には機械学習までのあらゆるものが含まれます。また、実装するシステムのパーティショニングに応じて、ユーザ空間内やアプリケーション層内でこのステップを行うことも可能です。このコードはオープンで使用できるので、エンド・アプリケーションのニーズに応じて、様々なメーカーが提供する別のFPGAや異なるプロセッサ・ファミリへの移植が可能です。

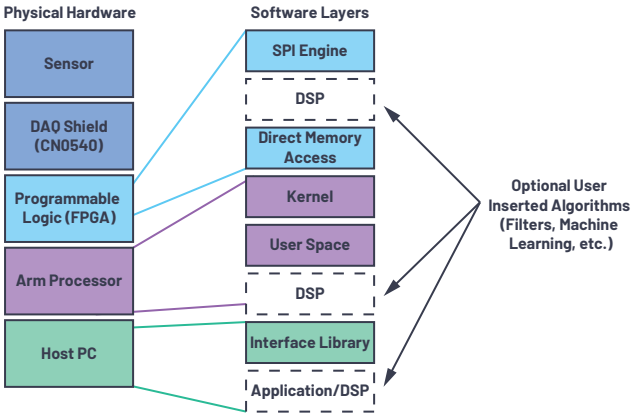

図2. CN0549プラットフォームのシステム・スタック

ARM®プロセッサ内部には2つのソフトウェア・オプションがあります。どちらを使用するかはユースケースによって異なりますが、どちらのオプションも多くの開発者に使われています。

- Linux：カーネルの産業用入出力（IIO）フレームワーク内に構築されたDAQシールド用に、カーネル内ドライバを備えています。これは、Kuiper Linux®と呼ばれる完全組込み型Linuxディストリビューションと組み合わせて使われます。Kuiper LinuxはARMコア内で実行され、Raspberry Pi OSを基本としています。
- No-OS：Linuxカーネルで使われるものと同じドライバを持つベアメタル・プロジェクトが用意されており、これはXilinxまたはIntelのSDKと共に使用することができます。これは、もう1つの実装としてリアルタイム・オペレーティング・システム（RTOS）環境内に実装することもできます。

開発者には、Linuxを学習してからシステム開発を始めることが推奨されます。これにより、ツールを最大限に利用することが可能になるからです。また、Linuxでは膨大な数のパッケージとドライバを使用できるので、望みの開発環境を構築することができます。システム設計が安定して最適化できる状態になったら、No-OSに切り替えて必要なソフトウェアだけを出荷するのが一般的です。しかし、これはアプリケーションに依存する部分が極めて大きく、多くのメーカーは、その柔軟性の高さからフルLinuxシステムを出荷しています。

プログラマブル・ロジック用のHDL同様、すべてのカーネル・ソース、Kuiper Linuxイメージ、およびNo-OSプロジェクトは完全なオープンソースであり、あらゆるコンポーネントはエンド・ユーザが望むように変更できます。これらのコード・ベースは、必要であれば様々なプロセッサ・システムやランタイム環境へ移植することも可能です。

図2の最後のコンポーネントはホストPCへの接続で、グリーンのボックスで示されています。システムの実行時は、デバイスを設定して、分析のためデータ・ストリームをホスト・システムへ戻すことができます。この場合、開発者はMATLABやTensorFlowなどの標準ツールを利用して、ホスト・マシン上で各種のアルゴリズムを作成します。その後、最終的にこれらのアルゴリズムを組込みターゲットに移し、ローカル処理能力を使って、アルゴリズム開発の反復をより短時間で行えるようにします。

CbMデータへのアクセス – 第1段階

一般に、ARMプロセッサとPLは、設計フローに沿って開発がかなり進んだ段階、つまり展開のためにシステムを最適化する時点で使われます。したがって、一般的に開発者が初めてこれを使用する場合は、最初にワークステーションから組込みシステムへのリモート接続が必要になります。組込みシステム上でLinuxを実行する場合、リモートやローカルによるワークステーション上でのコード実行は、インフラストラクチャがどのように設計されたのかに応じ、比較的透過的なプロセスとして行われます。これは主に、libIIOと呼ばれるオープン・ライブラリによるものです。libIIOは、カーネルのLinux IIOフレームワーク内に構成された様々なデバイス・ドライバに対して、シンプルかつ一貫したアクセス・モデルを実現するインターフェース・ライブラリです。このライブラリは、CbMプラットフォームの使用を柔軟なものとする中心的な存在であり、データ・ストリーミングとデバイス制御のための機能を提供します。

libIIO自体は2つの主要コンポーネントに分かれています。

- libIIOライブラリ。これは、IIOドライバの様々なプロパティや機能にアクセスするためのCライブラリです。これには、ADC、DAC、センサーなどの各種デバイスとの間でやり取りするストリーミング・データが含まれます。
- iiodと呼ばれるIIOデーモン。これは、libIIOライブラリ、またはライブラリを使用するクライアントと、実際のドライバへのカーネル・インターフェース間のアクセスを管理します。

libIIOとiiod自体への書込みは、バック・エンドと呼ばれる部分にあるドライバへの各種のアクセス方法を可能にする様々なコンポーネントから行われます。バック・エンドは、ローカルおよびリモート・ユーザからのlibIIOの制御とデータフローを可能にします。また、コンポーネント化されているので、システムには新しいバック・エンドを追加することができます。現在、libIIOがサポートしているバック・エンドは4種類です。

- ローカル：同じ装置に接続されたハードウェア用に、ローカルでアクセス可能なドライバのアクセスを可能にします。
- USB：libusbを利用します。このバック・エンドは、USBリンクを介したドライバのリモート制御を可能にします。
- シリアル：シリアル接続されたボード用に、より一般的なインターフェースを提供します。UARTが最も広く使われます。
- ネットワーク：最も多用されるIPベースのリモート・バック・エンドで、ネットワークを介したドライバへのアクセスに使われます。

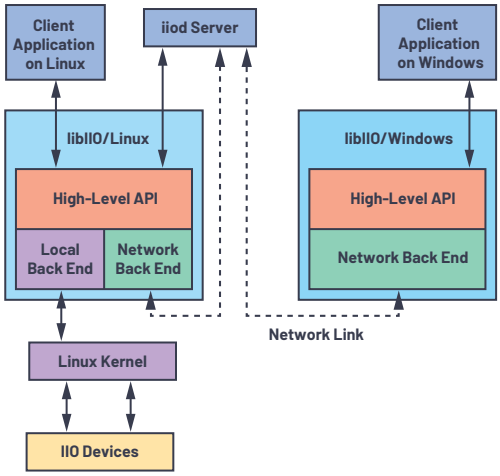

図3. ネットワーク・バック・エンドを使用するlibIIOシステムの概要

図3は、libIIOのコンポーネントがどのように使われ、システム全体にどのように組み込まれているのかについて、その概要をシステムレベルで示したものです。図の左側が組込みシステムです。この部分にはlibIIOライブラリがインストールされており、iiodデーモンを実行します。ユーザは、この組込みシステムからローカル・バック・エンドにアクセスできる他、ネットワーク・バック・エンドにもアクセスできます。そのコード内では、1つの行を変更するだけでバック・エンドを切り替え、どちらかを指定することができます。ターゲット・コードに他の変更を加える必要はありません。

図3の右側はリモート・ホストを表しており、任意のオペレーティング・システムを実行できます。Wondows、macOS、Linux、BSD用のオフィシャル・パッケージがあります。この図ではネットワークベース（IPベース）のバック・エンドが使われていますが、これはシリアル接続、USB接続、またはPCIe接続とすることもできます。ユーザの視点からすると、libIIOは、Cライブラリ自体から、あるいはPython、C#、Rust、MATLAB、Node.jsなどの他言語に使用可能な数多くのバインディングから利用することができ、アプリケーションから様々なドライバへのインターフェースを必要とするユーザに非常に豊富な選択肢を提供します。

```
# Connect To IP context
ctx = iio.Context("ip:192.168.2.1")
# Get ADC
adc = ctx.find_device("ad7768-1")
# Set sample rate
adc.attr['sampling_frequency'].value = '128000'
```

```
# Connect To local context
ctx = iio.Context("local:")
# Get ADC
adc = ctx.find_device("ad7768-1")
# Set sample rate
adc.attr['sampling_frequency'].value = '128000'
```

図4. リモートlibIIOとローカルlibIIOの例

アプリケーションとツール

新しいデバイスで開発を始める場合、一般的に、libIIOを直接使用することは推奨できません。したがって、libIIOをベースに作成された高水準アプリケーションが数多く存在します。これらのアプリケーションを使用すれば、コマンド・ラインまたはGUIフォーマットを使い、あらゆるIIOデバイスの基本的な設定を行うことができます。これらは、それぞれIIO Tools（IIOツール）とIIO Oscilloscope（IIOオシロスコープ）です。

IIO ToolsはlibIIOに付随するコマンド・ライン・ツールのセットで、低水準デバッギングとスクリプト使用を通じた自動タスクに便利です。例えば、実験室でのテスト時には、プラットフォームを異なるサンプル・レートのモードでセットアップして、いくつかのデータを収集するのに役立ちます。これは数行のBashを使って、あるいはIIO Toolsを利用したバッチ・スクリプトを通じて、簡単に行うことができます。ADCのサンプル・レートと入力コモンモードを変更するためにローカルまたはリモートで実行できる簡単な例を、図5に示します。この例では、iio_attrというIIOツールを利用しています。このツールを使用すると、デバイスの構成設定を簡単に更新することができます。

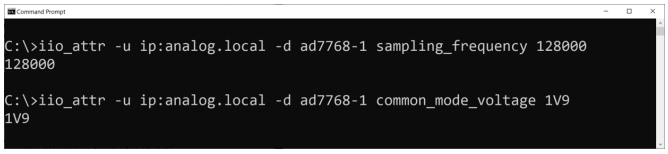

図5. IIO Toolsのiio_attr部分の使用例

しかし、ユーザにとって最も一般的な入口は、GUIアプリケーションであるIIO Oscilloscopeです。これは通常、OSCと呼ばれます。OSCは、IIO Tools同様、あらゆるIIOドライバを制御できるように汎用性を持たせて設計されています。また、libIIOをベースにしているので、リモートで実行したりボード上で直接実行したりすることが可能です。しかし、プラグイン・システムも含まれており、特定のドライバやドライバの組み合わせに合わせて特別なタブを追加することができます。CN0540ベースのボード用に自動的にロードされるプラグイン・タブを図6に示します。これには制御タブやモニタリング・タブも含まれています。これらのタブは、CN0540のADC、DAC、制御ピンの低水準機能にアクセスするための使いやすいインターフェースと、データ・アクイジション・ボードおよびテスト・ポイント・モニタリングの基本的なブロック図を提供します。その他のデフォルト・タブや使用可能なプラグインについては、Analog Devices Wikiで他のOSC関連ドキュメントを参照してください。

OSCの重要な側面の最後はキャプチャ・ウィンドウです。キャプチャ・ウィンドウは、ADCやlibIIOベースのバッファから収集したデータをプロットする機能を提供します。図7は周波数領域モードで使用した場合のキャプチャ・ウィンドウで、データのスペクトラム情報がプロットされています。時間領域、相関、コンステレーションなどを含む他のプロットを表示することもできます。これは、デバイスのスポット・チェック、デバッグ、または評価プロセスに有効です。プロットには、マーカー、ピーク検出、高調波検出、均等位相予測などの一般的なユーティリティが含まれています。OSCもオープンソースなので誰でも拡張でき、より多くのプラグインやプロットを追加したり、既存の機能に変更を加えたりすることができます。

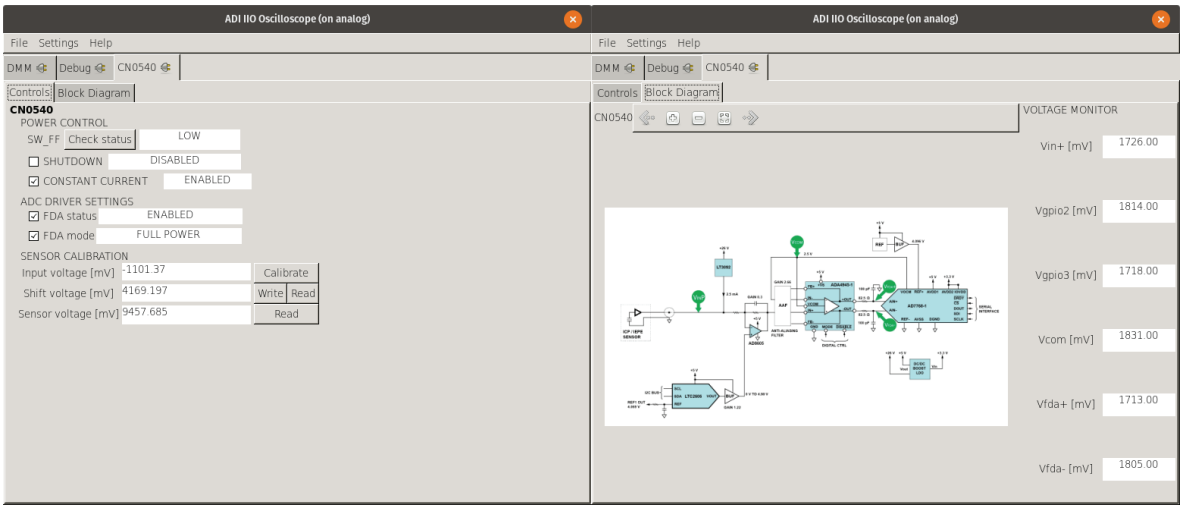

図6. CN0540 IIO Oscilloscopeのプラグイン・タブ

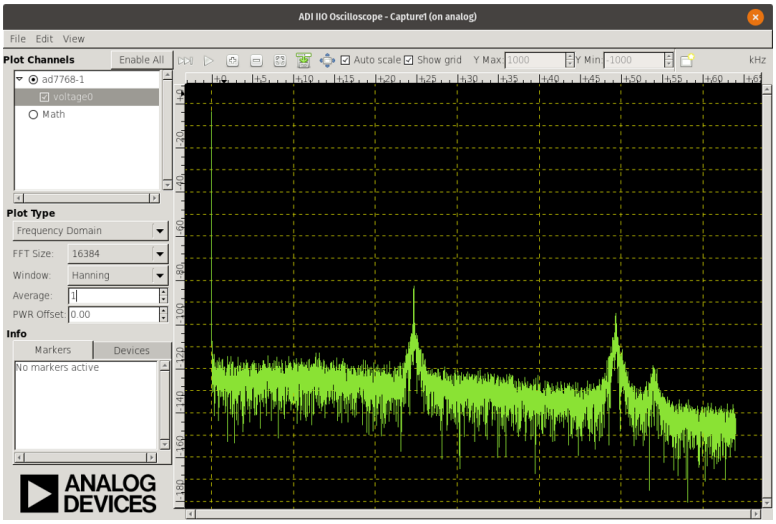

図7. IIO Oscilloscopeのキャプチャ・ウィンドウ（周波数領域モード）

アルゴリズム開発環境の統合化

ここまでは、ほとんどのエンジニアが初めてCN0549を使用する際の開始点となる、中心部分の低水準ツールについて見てきました。これらを最初に理解しておくことは重要です。それによって開発者は、システムの柔軟性と、利用できる様々な選択肢やインターフェースを理解することができます。しかし、開発者からすると、ベースライン・システムを起動して実行した後に、MATLABやPythonなどのツールを使ってデータをすぐにアルゴリズム開発へ移動させたい場合もあります。それらのプログラムはハードウェアからデータをインポートでき、必要な場合は追加的な制御ロジックを設計することもできます。

機械学習開発サイクルの場合は、通常、データを扱うために求められるソフトウェア環境とは関係なく、開発者が従う一般的なフローが存在します。このプロセスの一例の概要を図8に示します。この図では、データが収集され、そのデータがテスト用とトレーニング用に分割されて、更にモデルやアルゴリズムが開発され、最後に現場での推論のためにそのモデルが展開されます。実際のサービスでは、新しい学習成果を生産モデルに導入するために、このプロセス全体が継続的に実行されます。TensorFlow、PyTorch、あるいはMATLAB Machine Learning Toolboxなどのツールは、このプロセスを考慮した上で機能します。このプロセスは意味のあるものですが、通常、データの収集と整理のための努力やデータ管理の複雑なタスクは、軽視されたり、完全に無視されたりしがちです。このタスクを単純化するために、これらのツールやパッケージを念頭に置いて、関係するソフトウェア・エコシステムが設計されました。

Pythonの統合 – Python分析ツールへの接続

まずPythonから説明します。CN0549のデバイス固有クラスはPyADI-IIOモジュールを通じて使用できます。デバイスのサンプル・レートを設定し、Ehernetを介してバッファの内容を取り込む簡単な例を図9に示します。複雑なレジスタ・シーケンス、分かりにくいメモリ制御コール、あるいはメモリに格納するランダム・ビットなどはありません。これは、ドライバ、libIIO、PyADI-IIOによって自動的に管理されます。これらのドライバ、libIIO、PyADI-IIOは、ボード上で直接実行されるか、ワークステーション上でリモート実行されます。また、クラウド上で実行することも可能です。

PyADI-IIOはpipおよびcondaでインストールでき、ドキュメントに示すプロパティに従い、使いやすいようにコントロール・ノブを露出させます。また、NumPyアレイやネイティブ・タイプのように一般的に理解しやすいタイプのデータも提供し、データ・ストリームを使用できる場合はその単位変換を扱います。こ

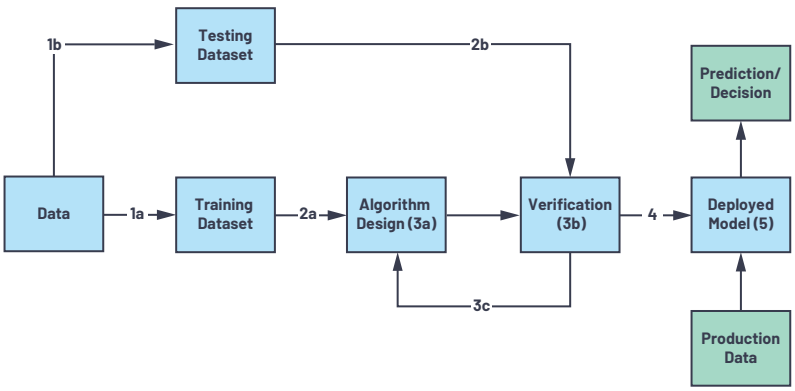

図8. 機械学習モデルの開発フロー

れにより、Jupyter Notebookのような環境へのPyADI-IIOの追加が容易になり、様々なツールや複雑なデータ変換を使わなくても機械学習パイプラインに容易にデータを入力できるようになるため、開発者は面倒なAPI変換やデータ変換ではなく、アルゴリズム自体に集中できます。

```
import adi # Import module
xl = adi.cn0532("ip:192.168.2.1") # Create object
xl.sample_rate = 12800 # Update sample rate
data = xl.rx() # Collect data
```

図9. PyADI-IIOの例

MATLABの統合 – MATLABへの接続

MATLAB側では、CN0549およびそのコンポーネントのサポートはAnalog Devices Sensor Toolboxを通じて行われます。このツールボックスは、PyADI-IIOと同様に様々な部品用のデバイス固有クラスを備えており、MATLABシステム・オブジェクト（MSO）としてそれらを実装します。MSOは、MathWorksのコード作成者がハードウェアや様々なソフトウェア・コンポーネントを処理できるようにするための標準化された方法で、コード生成、Simulinkのサポート、および一般的な状態管理を支援する高度な機能を備えています。多くのMATLABユーザは、スコープやシグナル・ジェネレータなどのMSOとして実装されたMATLAB機能を、それと知らずに利用しています。図10ではCN0532インターフェースとDSPスペクトラム・アナライザ・スコープを使用していますが、これらは共にMSOとして実装されています。この場合も、PyADI-IIO同様、従来からのMATLABユーザにとって使いやすいインターフェースがあります。

ハードウェアの接続に加えて、Sensor ToolboxはHDLやC/C++用のコード生成ツールとの統合も可能です。これらはIPの開発、シミュレーション、および展開用の強力なツールで、HDLの設計やツーリングに慣れていないエンジニアでも、MATLABやSimulinkを理解していれば使用することができます。

```
%% Configure object interface
xl = adi.CN0532('uri','ip:192.168.2.1');
xl.SampleRate = '1024';

%% Create scope
sa = dsp.SpectrumAnalyzer('PlotAsTwoSidedSpectrum',false);
for k=1:10
    sa(xl());
end
```

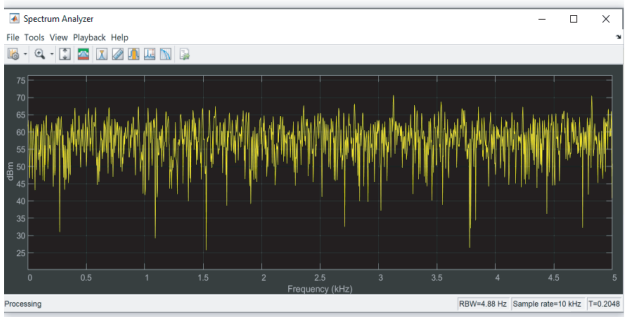

図10. スコープを使ったSensor Toolboxのストリーミング例

TensorFlowを使用する分類の例

CN0549キットには、基本的なデータ・ストリーミングから機械学習分類まで、いくつかの例が含まれています。CN0532の振動データのような時系列データの機械学習には、いくつかの異なる視点からアプローチすることができます。これにはサポート・ベクター・マシン（SVM）や長・短期記憶（LSTM）を含むことができる他、データを時系列として直接捉えることができる場合はオートエンコーダを含めることも可能です。しかし多くの場合は、時系列の問題をイメージ処理の問題に変換して、そのアプリケーション空間で得られた豊富な知識とツールを利用する方が、より有効な方法となり得ます。

Pythonにおけるこのアプローチを見てみましょう。PyADI-IIOと共に提供されている例の1つにおいては、振動のあるファンにCN0532を取り付けることによって、いくつかの測定が行われます。これはファンをいくつかの異なる設定（スリープ、通常、アレルゲン）に変えて行われ、各モードでは409,600サンプルが

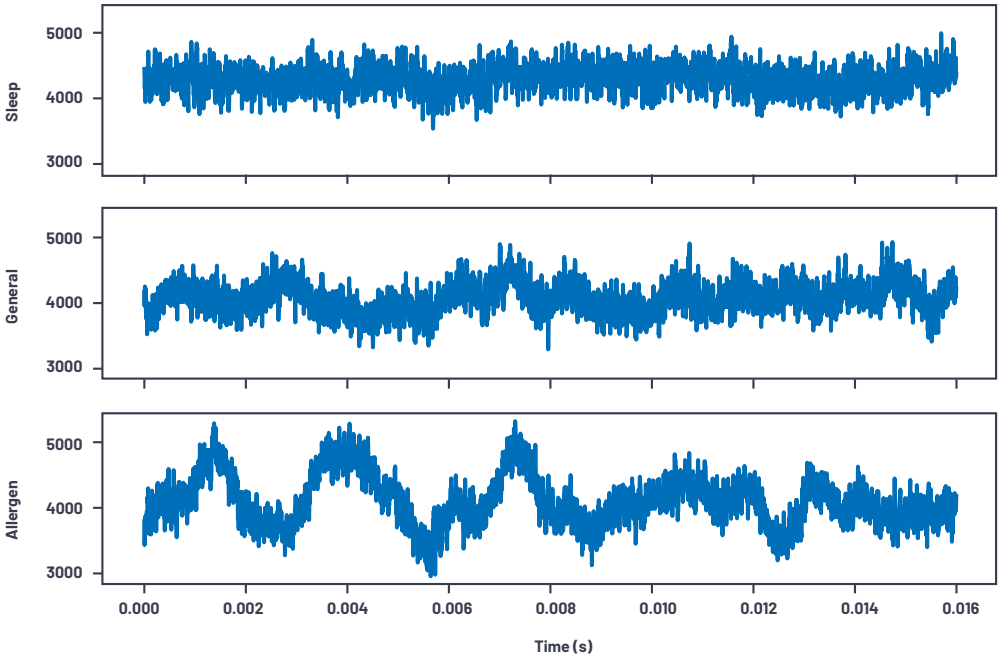

図11. 時系列で表したファン振動データ

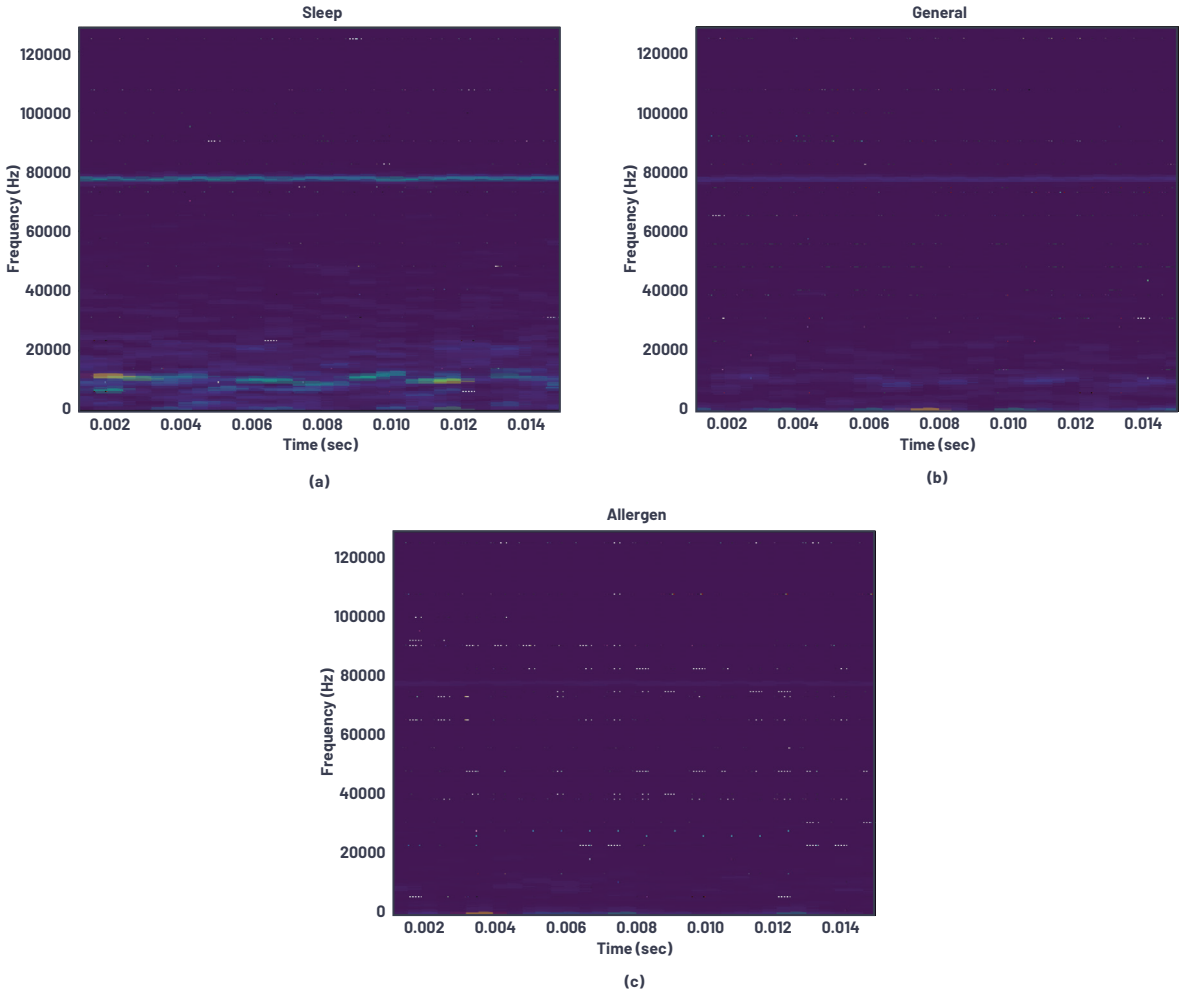

図12. キャプチャした振動データのスペクトログラム

収集されます。図11に示すこのデータを検討すると、アレルゲンのケースの時間領域は識別が容易ですが、他の2つのケースは区別しにくくなっています。これらを検査によって識別することは可能ですが、これらのケースを識別するアルゴリズムを使用する場合、時間領域ではエラーが発生しやすくなります。

これらのユースケースを区別しやすくするために、データは周波数領域に変換され、スペクトログラムを使って異なる周波数の密集状態が時間変化としてプロットされています。図12に示すスペクトログラムにはデータに明確な違いが現れており、時間方向の傾向は図11と一致しています。これらのスペクトログラムは実質的にイメージであり、これは従来からのイメージ分類手法を使って処理することができます。

データセットはトレーニング用セットとテスト用セットに分割され、そのスペクトログラムが、3つの全結合層を持つニューラル・ネットワーク（NN）のみのモデルと、それより小さい畳み込みニューラル・ネットワーク（CNN）モデルの両方に送られます。これらは共にTensorFlowで実装され、エポック数100でほぼ100%のテスト評価に容易に収束させることができます。CNNは、おおむね1%の調整可能パラメータを使い約半分の時間で収束するので、はるかに効率的な設計になります。短時間で収束するCNNの概要を、精度とエポック数で表したトレーニング収束プロットとして図13に示します。

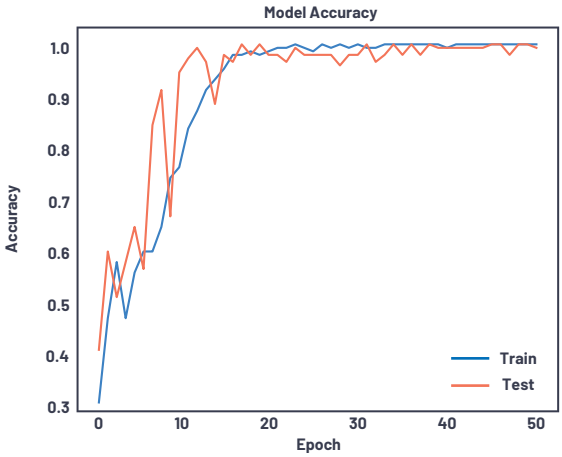

図13. 振動スペクトログラムに関するCNNトレーニング精度の時間変化

PyADI-IIOソース・ツリーにおけるこのGitHubの例では、Pythonスクリプト、ノートブック、データセットをすべて使用できます。データセットを使用できるので、Tensorflowによるこのサンプル・デモはCN0549のハードウェアがなくても使用できます。ただし、ハードウェアがあれば、トレーニング完了モデルを使用したリアルタイムの推論が可能です。

エッジからクラウドへ：組込みソリューションへの移行

モデルを作成したら、推論や意思決定のためにそのモデルを展開することができます。CN0549を使用すれば、このモデルをリモートPC上に置いてCN0540からデータをストリーム送信したり、組込みプロセッサ上で直接実行したりすることができます。実装によっては、このモデルをプロセッサ内に置くのにより多くの技術的努力が必要とされますが、電力効率が1桁向上し、リアルタイムでの動作も可能になります。幸いなことに、機械学習モデルを展開するためのツールとソフトウェアは、過去数年間で大きな発展を遂げました。

FPGAの利用

XilinxもIntelも、高水準言語をHDLコードに変換してFPGAで実行するための高水準合成（HLS）ツールを提供しています。これらは通常、IPコアへのモデル変換を容易にするために、TensorFlow、PyTorch、あるいはCafeといったPythonフレームワークに統合化されて、エンジニアがDE10-NanoやCora Z7-07S、あるいはカスタム・システムにIPを展開できるようにします。更にこれらのIPコアは、アナログ・デバイセズが提供するオープンHDLリファレンス設計に組み込まれます。図14は、VivaoによるCora Z7-07S CN0540のスクリーンショットに注釈を入れたもので、データパスに焦点が当てられています。この設計では、CN0540からのデータがSPIピンを通じて読み込まれ、SPIエンジンによって24ビットのサンプルが変換されて、DMAコントローラからメモリに渡されます。DSPや機械学習モデルは、すべてこのパイプラインのデータパスに直接挿入することができます。

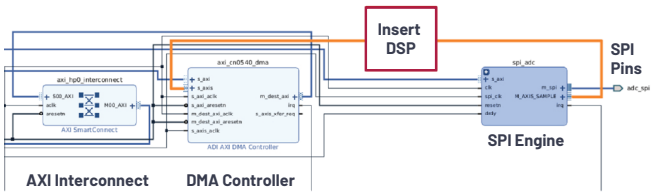

図14. Vivado 2019.1に表示された
Cora Z7-07S HDLリファレンス設計のデータパス

マイクロプロセッサの利用

アルゴリズムは、HDL層に変換する代わりに、ARMコア内で直接実行することができます。データ・レートとアルゴリズムの複雑さによっては、これは妥当な開発パスであり、通常はもっと単純なものになります。ARMコア用のCコードやPythonの開発に要するリソースや時間はHDLよりもはるかに少なくて済み、通常は維持も容易になります。

MATLAB Embedded Coderなどのツールはこのプロセスを更に合理化することができ、MATLABを、組込み可能で最適化されたARMコア用のCコードに自動的に変換します。あるいは、TensorFlowにはTensorFlow Liteのようなツールがあります。これはPythonライブラリの組込み可能なCバージョンで、より容易に組込みターゲットへ移行することができます。

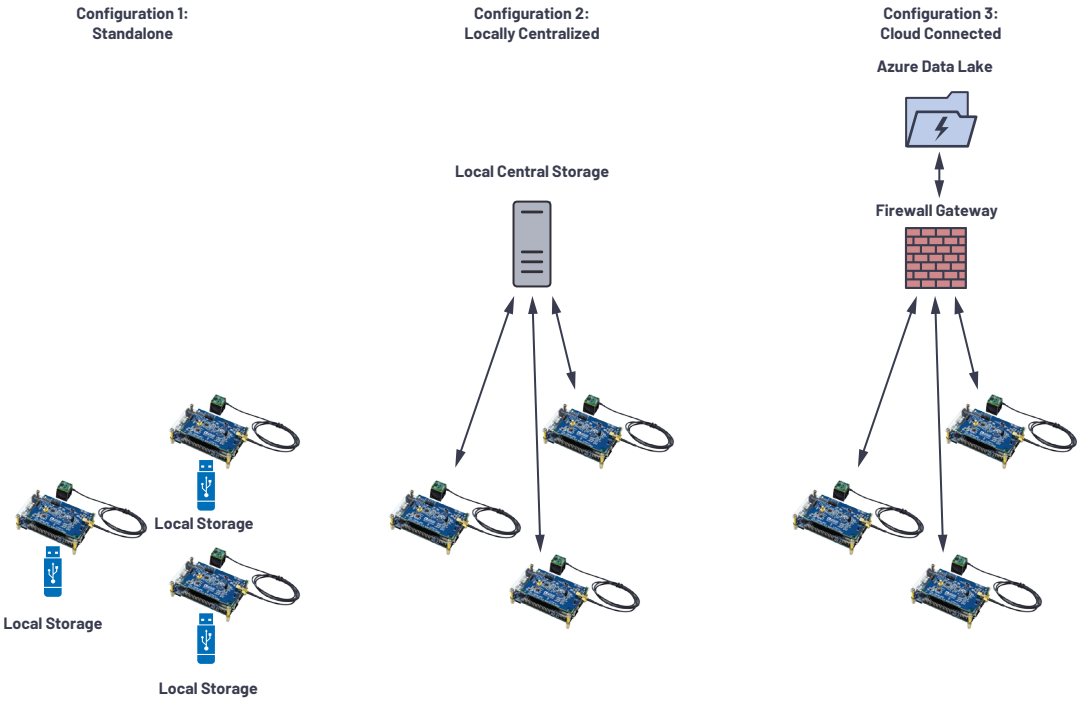

図15. CbMネットワークのトポロジ

スマート意思決定トポロジ

状態基準保全は、ハードウェア的にもソフトウェア的にも、1つのサイズであらゆるスペースに対応できるものではありません。CN0549が柔軟性を備えた設計となっている理由は、ここにあります。CbMのための異常検出といった問題について考える場合は、通常、2つの時間スケールからのアプローチが可能です。1つは、安全関連のシナリオのように直ちに対応する必要がある場合です。もう1つは長期的な時間スケールに関するもので、この場合はメンテナンスや装置の交換との関連性が強くなります。いずれの場合も、異なるアルゴリズム、処理能力、アプローチが必要です。

理想的ケースの機械オペレータとしては、モデルのトレーニング用に大きなデータ・レイクが必要であり、誤検出のない短期的な検出と、将来的なメンテナンス予測のために運転中の機器類から継続的に送られてくるストリーム・データの両方を扱うことになります。しかし、ほとんどのオペレータにとってこれは発生する可能性の低いケースであり、データ・レイクが干上がってしまう可能性もあります。また、一部のオフザシェルフ・ソリューションでは、セキュリティ上の懸念、物理的位置、ネットワーキング、あるいはトポロジに関する条件によって、データの収集が難しくなる可能性もあります。これらの問題点から、より特別なソリューションの必要性が生じます。

CN0549は複数の接続オプションを持つスタンドアロン・システムです。標準のLinuxを実行するので、そのまますぐにイーサネットやWi-Fiのような従来型のネットワーキング・スタックを使用することができ、必要であればセルラー・モデムに接続することも可能です。図15に示すように、実際のアプリケーションでは優れた代表的トポロジがいくつかあります。

図15の左端に示されているのがオフライン収集のケースで、遠隔地やインターネットへの接続ができない場合に使用できます。この場合は大容量のストレージ媒体がプラットフォームに併設され、手動により一定のスケジュールでデータが収集されます。これに対し、他の2つのオプションは、共通のエンドポイントにデータをストリーム送信します。図15の中央に示す構成は、特定組織内のみで使われる外部から隔離されたネットワークか、他から離れた場所にあるプラットフォームのクラスタで、一元的にデータを収集します。セキュリティ上の懸念がある場合や、単純にネットワークへの接続ができない場合は、この形態が必要になります。これらいずれの構成においてもCN0549のセットアップは容易で、最終展開時の具体的なニーズに合わせてカスタマイズすることができます。

3番目の構成がダイレクト・クラウド・オプションで、各プラットフォームがインターネットに直接アクセスして、クラウドへ測定値を送ります。CN0549はLinuxで動作するので、プラットフォームは、Pythonのような言語を使って、Microsoft Azure IoTやAmazon IoT Greengrassなどの異なるクラウド・ベンダーのAPIを容易に利用することができます。これは、新たに接続された装置のデータ・レイク作成を開始するための方法を簡単に作り出せることを意味します。

クラウドとローカル・プロセスの間に安定した接続が存在する場合は、既に述べたように、ローカルで必要とされるものあるいは実行できるものと、クラウドで実行できるものとの間で、異なるアルゴリズムを使い分けることができます。これには、アルゴリズムの複雑さに応じた処理能力やイベントに対する遅延に関わる条件と、クラウドへ送ることができるものに関する帯域幅上の制約の間で、自然なトレードオフの必要が生じます。しかし、CN0549は柔軟性が高いので、これらの要素に関する解決策を容易に見出すことができます。

まとめ

CN0549 CbMプラットフォームは、柔軟なシステムを実現し、アプリケーション開発を行う設計者に豊富なソフトウェア・リソースを提供します。ここでは、CbMおよび予防メンテナンス（PdM）の開発に様々なコンポーネントをどのように利用できるかという議論を通じて、ソフトウェア・スタックについて深く掘り下げた検討を行ってきました。ソフトウェアのオープン性、HDL、回路図、およびデータ・サイエンス・ツールとの統合化により、設計者はスタック全体を通じて、エンド・システムに必要なコンポーネントを利用することができます。要約すると、オープンソースのソフトウェアとハードウェアを備えたこの状態監視設計は、そのまますぐに使用できる使いやすいソリューションを提供して高い柔軟性を提供し、設計者がより短い時間で、より良い、要求に合った結果を実現することを可能にします。

著者について

Travis Collins

ウースター工科大学で電気工学およびコンピュータ工学の博士号と修士号を取得。主な研究分野はスモール・セル干渉モデリング、フェーズド・アレイ方向探知、およびソフトウェア無線用の高性能計算。現在はアナログ・デバイセズのシステム開発グループに勤務。通信、レーダー、一般的なシグナル・プロセシングなどのプリケーションが専門。

連絡先：travis.collins@analog.com

EngineerZone®
オンライン・サポート・コミュニティ

アナログ・デバイセズのオンライン・サポート・コミュニティに参加すれば、各種の分野を専門とする技術者との連携を図ることができます。難易度の高い設計上の問題について問い合わせを行ったり、FAQを参照したり、ディスカッションに参加したりすることが可能です。

ADI EngineerZone™
SUPPORT COMMUNITY

Visit *ez.analog.com*

＊英語版技術記事はこちらよりご覧いただけます。

TA23098-8/21