VEROLT

The Qt Company

ホワイトペーパー

Qt Quickアプリケーションを Qt for MCUsへポーティング

Qt Quickユニファイドアーキテクチャを基盤としたアプローチ

本ホワイトペーパーでは、既存のQt QuickアプリケーションをQt for MCUsを使ってマイクロコントローラ（MCU）にポートした具体的な事例をご紹介します。

今回のプロジェクトでは、デスクトップ、モバイル、および組み込みシステムで展開する既存の「HomeChef」アプリケーションのポートを行いました。すべてのデプロイメントでユニファイドアーキテクチャを用いていますが、ユースケースとOSは異なります。また、MCUポートを含め、コードベースとツールチェーンを共有することで、ブランドアイデンティティの確立を図っています。

本ホワイトペーパーでは、Qt for MCUs 1.0を使ってQt Quickアプリケーションを480kbまで縮小させ、STM 32F769にポートするプロセスの詳細をご紹介します。また、ごく短期間でいかにしてさまざまな課題を解決し、通常であればよりパワフルなハードウェアを必要とする高いパフォーマンスを実現したか、その詳細をご説明します。

目次

HomeChefのエコシステム：Qt Quickユニファイドアーキテクチャ

マルチデプロイメントの開発プロセスの課題
ユーザーがお気に入りのソフトウェアを複数のデバイスで使う場合、どのデバイスでも一貫性のあるエクスペリエンスを求めるのが当然でしょう。一方、開発者サイドでは、ソフトウェアをポートするプラットフォームの種類が増えるに従って、統一感のあるユニファイドアーキテクチャを開発するという課題の難易度が増すことになります。また、反復型開発プロセスへのシフトに伴い、製品開発が一層細分化する傾向も高まっています。

ユニファイドアーキテクチャメソッド（UAM）などのアプローチであれば、個別のユーザーインターフェース（UI）と再利用可能なコンポーネントをクロスプラットフォーム環境で利用することにより、複数のプラットフォームでの製品デリバリーを容易にできます。

このように製品デリバリーが細分化した環境では、「統合」の実現が必須課題となってきます。この課題を達成するに当たり、ユニファイドアーキテクチャメソッド（UAM）などのアプローチであれば、個別のユーザーインターフェース（UI）と再利用可能なコンポーネントをクロスプラットフォーム環境で利用することにより、複数のプラットフォームでの製品デリバリーを容易にできるとThe Qt Companyは考えています。

技術的な要件の増加に伴い、革新的な家電デザインの実現はますます複雑化し、コストも拡大傾向にあります。MCUは扱いやすい一面、高機能の新製品には性能面で不十分な場合があります。しかしながら、ハードウェアもソフトウェアも複雑化したハイエンドな産業用マイクロプロセシングユニット（MPU）では、追加コストが増大するという問題があります。

Qtテクノロジーがもたらすメリット
Qtでこれらの課題に対処する方法を概説するため、私たちは家電向けのデモ製品「HomeChef」を開発しました。1つのエコシステムでマルチデプロイメントする、現実的なユースケースのプロジェクトです。将来のサステナビリティを考えた場合、一般家庭ではこれから調理の最適化（より短時間で、安く、無駄なく、健康的に）が重要であると言えます。HomeChefは、ユーザーのキッチンに今何があるか、スーパーマーケットで何を買い足せばいいか、常に最新の情報を提供することで、調理の最適化を支援するアプリケーションです。同様のシナリオは他の業界やユースケースでも考えられるでしょう。このHomeChefは3種類のアプリケーションとデバイスで展開します：

デスクトップバージョンは、一週間の料理をサポートします。ユーザーはレシピブックやお気に入りからメニューを選ぶか、おススメ機能を使い、買い物リストに必要な食材を追加できます。また、栄養情報の確認や、生活の健康レベルに関する統計値を確認することもできます。本プロジェクトではネイティブなデスクトップアプリケーションを開発するのではなく、デスクトップアプリケーションをWebAssemblyにデプロイすることで、ユーザーがアプリをインストールしなくても実行できるようにしました。つまりHomeChefは、Qtを使ったソフトウェアアズアサービス（SaaS）の一例と言えます。

2つめはモバイルバージョンで、機能はデスクトップバージョンと同じですが、小型のフォームファクタ向けに最適化しており、ネイティブなiOS／Androidアプリとして開発しました。

3つめは組み込みクッキングマシンで、ユーザーはマシンの指示に従って手順を追い、料理を作ることができます。クッキングマシンには、i.MX6Dualベースのマンマシンインターフェース（HMI）パネルであるGarz&Fricke Santaroを採用し、組み込みLinuxおよびビルドシステムとしてYoctoを用いました。また技術的ハイライトとしては、WebGLストリーミングをカメラ出力のリモート表示用に、WebEngineをYouTube動画の表示用にそれぞれ追加しました。

いずれのバージョンも、通信はクラウドでMQTT[1]ブローカーを用いています（またはデモンストレーション用にローカルで実行）。

図1：Qt Quickユニファイドアーキテクチャ

Qt Design StudioとWebAssemblyを用いたHomeChefアプリケーション開発の事例

Qt Design studio[2] を使うのは初めてでしたが、トレーニングを行った結果、開発プロセスに容易に導入できることが分かりました。最初はQt Design Studioをデザイン案のプロトタイピングに使用しましたが、すぐに欠かすことのできない開発ツールになりました。

1 MQTTはパブリッシュ／サブスクライブ方式のマシンツーマシン（M2M）プロトコルです。最小限の通信オーバーヘッドでチャネルを提供します。

2 Qt Design StudioはUIデザイン&開発環境で、アニメーションUIを作成し、デスクトップやAndroid、組み込みLinuxデバイスでプレビューを行うことができます。

Qt UltraLiteへの拡張

MCUでHMIを開発した理由
Verolt社は組み込みHMI開発会社として常に、NXPなどの組み込みプラットフォーム向けのアプリケーション開発に関心を注いでいます。しかしながらそうしたアプリケーションは大抵の場合、Linuxベースのプロセッサ とGPU、その他のハードウェアサポートが必要になります。

私たちは、MCUベースのHMI製品をポートフォリオに加えたいと常々考えていました。しかし、私たちが本当に求めるものが業界にないのが実情でした。
そこへThe Qt Companyから、MCU向けに最適化されたグラフィックフレームワーク&ツールキットであるQt for MCUsのリリースが発表されたので、このテクノロジーへの投資を決定したというのが本プロジェクトの経緯です。

以下のセクションでは、私たちのアプローチの概要と共に、Qt for MCUsでのHMI開発実現に向けて行った分析、分析により明らかになったギャップ、これらのギャップ克服のために行った措置を見ていきます。

HomeChefのトップダウンアプローチ
LinuxでのHomeChefアプリケーションのデザインと開発はほぼ完了していたので、MCUでのHomeChef開発はトップダウンアプローチを取ることに決めました。

ユニファイドアーキテクチャの利点として、1つのプロジェクトを管理しやすい複数の小さなプロセスに簡単に分割できる点が挙げられます。

分割して克服：ユニファイドアーキテクチャの利点として、1つのプロジェクトを管理しやすい複数の小さなプロセスに簡単に分割できる点が挙げられます
HomeChefの特徴とMCUとの互換性について分析を行い、実現可能性に関するブレインストーミング、および定期的なライブコーディングと妥当性確認を繰り返しました。これにより開発チームは未知の課題を克服し、最小限のデザイン変更で機能の完成目標を達成できました。

このようなトップダウンアプローチにより、以下を実現しました：

- 顧客に最適な機能の開発に焦点を当てることができた。
- 各機能を完成後、直ちに個別にテストできた。プロジェクトの途中で統合テストやユーザーテストを頻繁に実施することで、結果の予測が容易になった。
- 新たに実装した機能を加えた継続的ビルドを顧客に定期的に送ることができた。

ギャップ分析（Qt for MCUsでできないこと）
マイクロプロセッサは通常、マルチプロセスとマルチスレッドが可能なOSを搭載します。マイクロコントローラはそうしたOSを搭載しないため、アプリケーションデザインにも影響が及ぶことになります。以下の表は、Qt for Embedded LinuxとQt for MCUsの機能比較をまとめたものです。

組み込みLinuxのQtモジュール	Qt for MCUsでの利用の可否
Qt Network	利用できない
Qt Quick Dialogs	利用できない
Qt Quick Controls	制約あり
Qt Quick Layouts	利用できない
その他のモジュール（例：Qt MQTT、Qt Serialport など）	利用できない

開発者は、Qt for MCUsがQt C++モジュール対応でない点に留意する必要があります。ただし、既存のコードは利用可能。

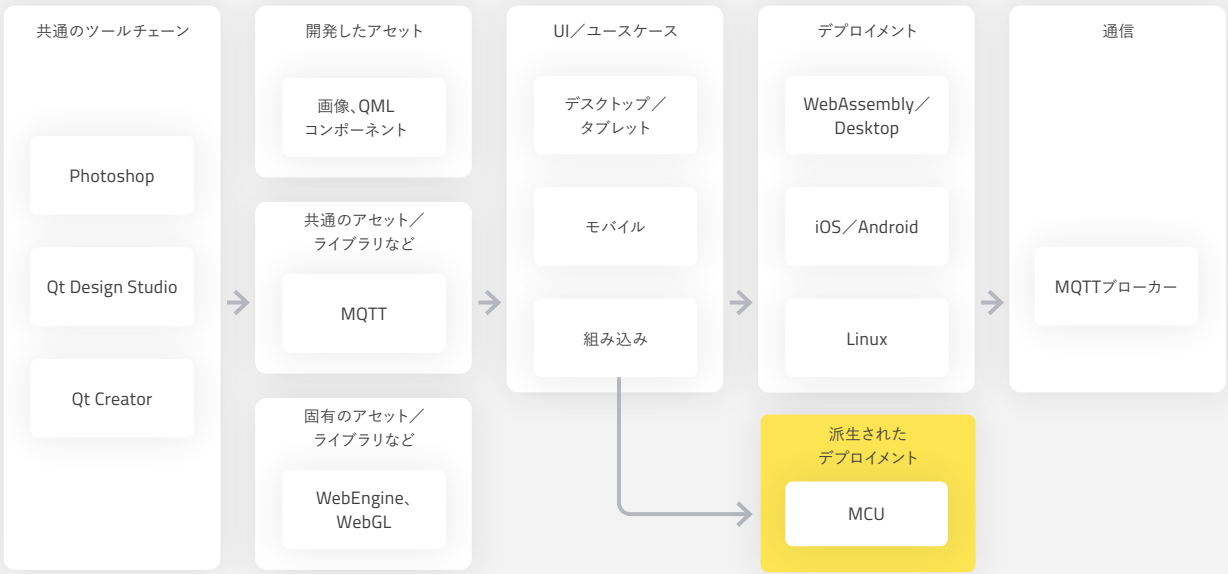


図2：HomeChefのMCUデプロイメント

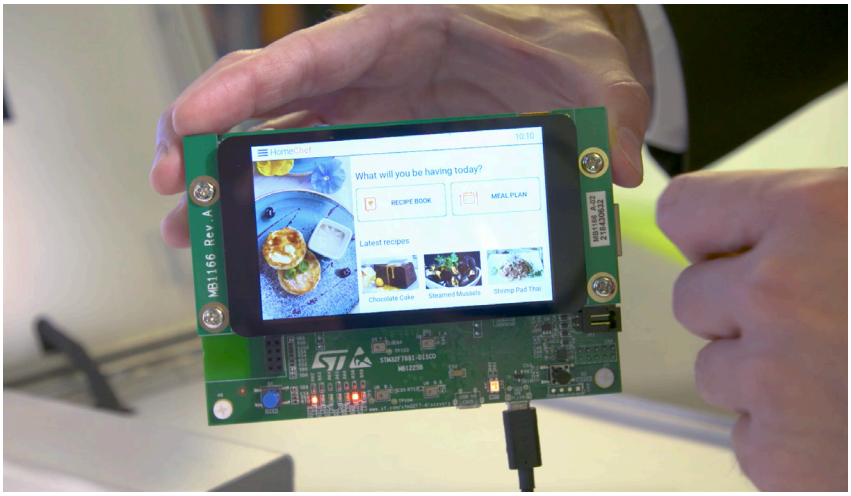
バックエンドはMQTTとWiFiモジュールの統合で労力がかかる一方、まずはフロントエンドのポートから着手することを決定しました

Qt for MCUsは、Qt Quickのサブセットであり、MCUのリソース上の制約に対応したQt Quick UltraLite[3] (QUL)を提供します。QULはメモリフットプリントが少ないという利点がありますが、反面、Qt Quickに比べて能力に限界があります。そのため、QML[4]のデザインエレメントの一部を簡素化する必要が生じました。この開発フェーズでは、以下のような教訓を得ました。

Qt Design Studio

フロントエンドはQt Design Studioを使って開発しました。開発段階でQULに未対応だったため、Qt Design StudioコンポーネントやQt Quickに固有の一部のコンポーネントは手直しが必要になりました。

Qt Design Studio コンポーネント	Qt for MCUs用の手直し	手直しを行った理由
円形のプログレスバー	タイムラインベースからタイマーベースのアニメーションに変更	Qt Design StudioコンポーネントのTimeline がQt for MCUsで使えないため
チャート	QMLでコンポーネントを一からデザインし直し	QChartモジュールが Qt for MCUsにないため
調理手順	細かな修正	QMLタイプのサポートがないため



HomeChefアプリケーションを MCUで実行

3 グラフィックランタイムを最適化し、リソースに制約のあるデバイスでハイパフォーマンスと低メモリ消費を実現します。

4 Qt Modeling LanguageはUIアプリケーションをデザインするための宣言型のマークアップ言語です。

Qt Quick UltraLite（QUL）移行時の課題

移行プロセスでは、QULとQt Quickのさまざまな違いに直面しました。HomeChefで手直しが必要になったエレメントは以下の通りです

Qt Quick	QUL 1.0	制約	次善策・回避策
QTimer	利用できるが制約あり	Triggeronstartの機能が使えない	明示的に開始
Rectangle border	利用できない		長方形を重ねてボーダーを表示
ListView	利用できるが制約あり	水平スクロールができない	フリックで水平スクロール
TextInput	利用できない		カスタムタイプ
TextEdit	利用できない	–	カスタムタイプ
Property Animation	利用できるが制約あり	アンカーのプロパティアニメーションが利用できない	タイマーとトランスフォームでオブジェクトを回転 Photoshopでアニメーションをプリベイク
Java scriptの各種機能	利用できない		
Popup	利用できない		高いZオーダーを用いたカスタムタイプ
フォント	利用できるが制約あり	フォントウェイトによりメモリフットプリントが著しく増加 フォントピクセルサイズを固定値にする必要あり	フォントのダイナミックリサイズを避ける必要あり
Loader	利用できない		ビジビリティを使ってページローディング
ScrollView	利用できない		フリックで垂直スクロール
Drawer	利用できない		アニメーションとビジビリティを用いたカスタムタイプ
StackView	利用できない		ビジビリティフラグを用いたカスタムタイプ
Model	利用できるが制約あり	別のファイルで宣言できない	モデルとデリゲートを同じファイルで宣言する必要あり
ComboBox	利用できない		カスタムタイプ
Application Window	利用できない		代替策なし。新しいバージョンで追加する計画あり
ItemDelegate	利用できない		代替策なし。新しいバージョンで追加する計画あり
Page	利用できない		代替策なし。新しいバージョンで追加する計画あり

QULのパフォーマンスとMCUの制約
個々のアニメーションを確認してみると、パフォーマンス低下の兆候が見られました。この問題を克服するためにいくつかのトライアルを実施し、最終的にコンピューテーションを減らして、複数のマイクロアニメーションに単一のタイマーを用いるという方法を選択しました。

たとえばあるユースケース（材料のこね具合を示す円形のプログレスバー）は、Qt Design Studioのエレメントで開発していました。具体的には「Arc」というエレメントを、「Timeline」というエレメントでアニメーション化しています。これらはQt Design Studioのエレメントなので、円形プログレスバーはデザインし直す必要が生じました。

- プログレスバーのアニメーションにメモリとパワーが足りなかったので、タイマーベースで回転させる必要があった。このアプローチにしたところ、メモリフットプリントを著しく減らし、滑らかなアニメーションを実現できた。
- 最終的なアプリケーションサイズは480kbとなり、参照用の組み込みUIとほぼ同じルック＆フィールを実現できた。

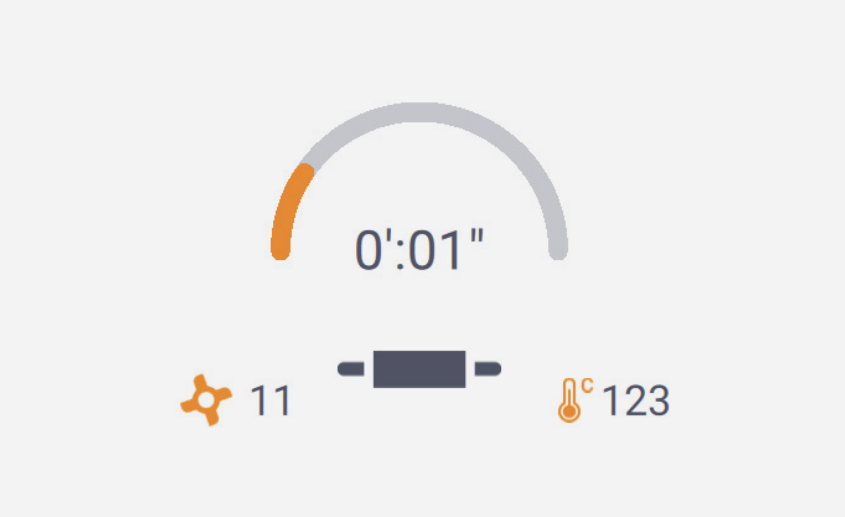


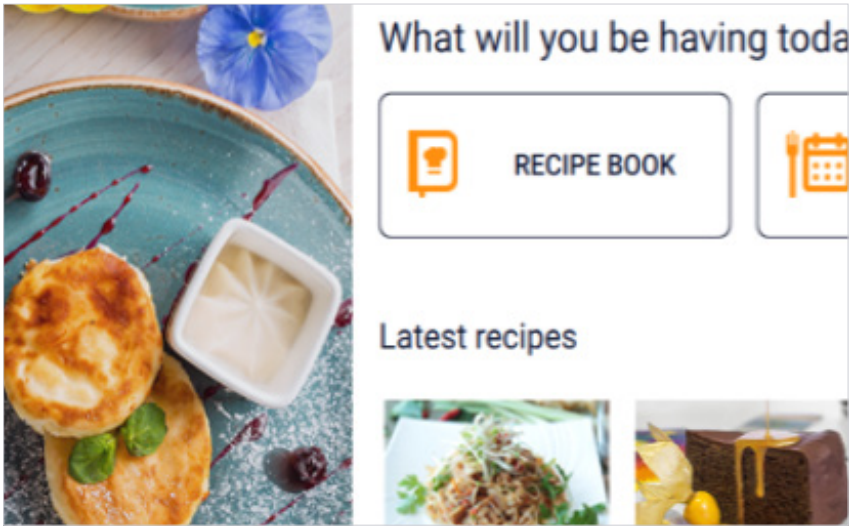
図3：スクリーンのアニメーションエレメント（温度、ファンのスピード、プログレスバー）は同じタイマーを再利用することで最適化

全体的な労力と時間

ソフトウェアのマイクロコントローラへの移行に際しては、開発コストの削減と市場投入時間の短縮化を実現しながら、実用的なソフトウェアアーキテクチャの組み込みシステムを提供しなければなりません。

HomeChefのケースでは、ユニファイドアーキテクチャを選択し、Qt製品の強みを複数のプラットフォームに適用したことで、開発者が最適な学習曲線でスピーディにMCUへの展開を実現しました。組み込みソフトウェアのリソースと開発時間が足りなかったため、ソフトウェアコンポーネントは再利用が不可欠でした。
Qt QuickからQULへの移行にかかる時間と労力について、MCU版HomeChefでとりわけ複雑な4つのスクリーンをベースに考えてみましょう。ここでは、中くらいのスキルとQMLの基本知識を持った開発者を想定しています。

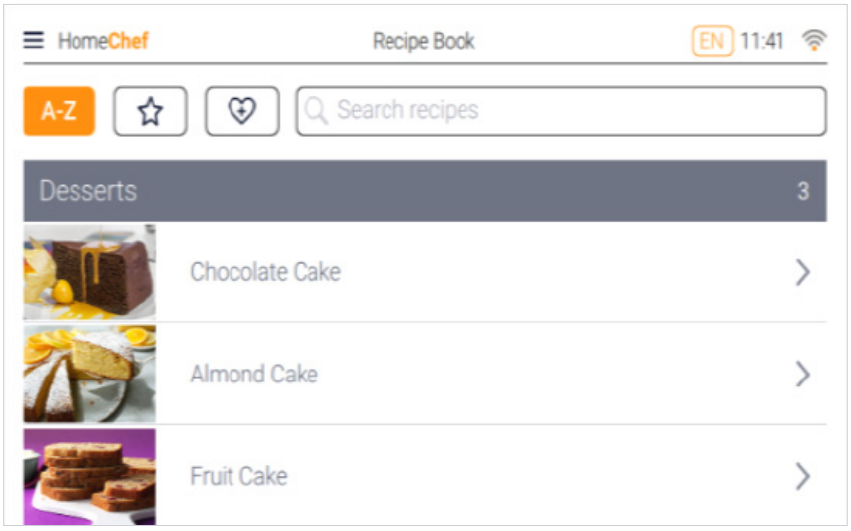
スクリーン



複雑度：中くらい

- このスクリーンは、水平スクロールリストやその他のGUIエレメントなど、中くらいのレベルの変更が必要でした。
- **所要時間：**1日

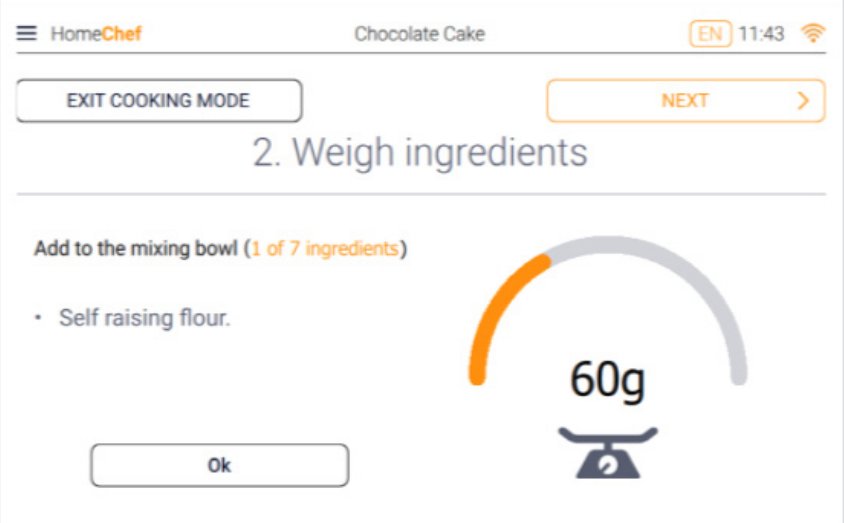
スクリーン



複雑度：高い

- ネストされたアコーディオンリスト：このコンポーネントはMCU用にGUIの変更が必要でした。
- 小さいスクリーンサイズに対応するため、UIを大きめのフォントとマウスエリアに変更する必要がありました。
- **所要時間：**3日

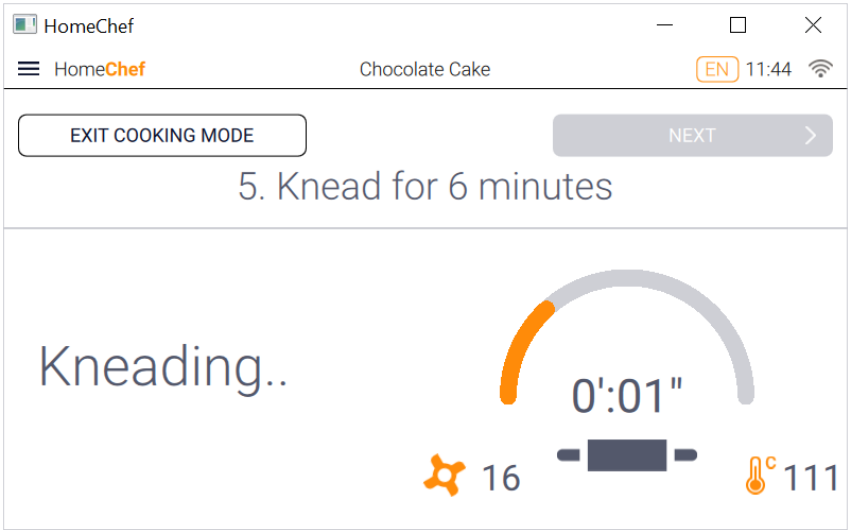
スクリーン



複雑度：中くらい

- 円形プログレスバーのアニメーション
- Qt Design Studioエレメントをタイマーベースのアニメーションに変更
- 複数画像を重ねてデータを表示し、不要な画像データを削除
- **所要時間：**1日

スクリーン



複雑度：中くらい

- 円形プログレスバーのアニメーション
- 回転アニメーション
- ファンや温度など複数のマイクロアニメーションに同じタイマーを使用しました。
- **所要時間：**2日

Qt for MCUsのリリースにより、The Qt Companyは低コストで大量生産される製品分野の基本的なギャップを克服しました。

まとめ／教訓

私たちはお客様のご意見を基に、消費者／顧客とのすべてのタッチポイントで共通のブランドアイデンティティを打ち出すことがビジネスの成長に不可欠であることを学んできました。このことは、HomeChefのような家電製品業界だけではなく、他の多くの業界に当てはまります。

このデモにより、共通のデザインとコードベースをあらゆるデプロイメントシナリオに適用できることを証明できたと考えています。結果的に、高度なエンジニアリングスキルが不要、デザインチームと開発チームの協働に摩擦が生じにくい、プロトタイピングから製品開発への移行がスムーズになる、製品の反復開発が可能、複数のUIやバージョン、プラットフォームで各種コンポーネントを再利用可能といったメリットを得ることができました。

Qt for MCUsのリリースにより、The Qt Companyは低コストで大量生産される製品分野の基本的なギャップを克服しました。上述したように、1.0バージョンにはすでに包括的な製品ポートフォリオが用意されています。QMLタイプとSCXML[5]の改善、Qt Design Studioとの完全統合、画像処理機能（2.5Dアニメーションなど）なども大きな強みですが、私たちにとっての決定打はそこではありません。

総合的に言って、今回のHomeChefの概念実証を通じ、Qt for MCUsであればエンジニアリングコストを削減しながら、同時に顧客にとってのBOMも削減できることを証明できたと考えています。

レガシーコードの縛りがなければ、MCUを含むユニファイドアーキテクチャはQtであれば可能です。また賢明なアプローチを取ることができれば、エンジニアリングチームに将来的なイノベーションの実現も約束できるでしょう。私たちはこの分野のサービスプロバイダーとして、Qt for MCUsに勝る代替案はないと結論付けています。

筆者

ラース・ケーニッヒ（Lars König）
ソフトウェアサービス担当ディレクター、
ドイツ・ウルム
+49 151 612 575 36
Lars.Koenig@verolt.com
www.verolt.com

ラヴィ・ダッタトラヤ（Ravi Dattatraya）
エンジニアリング・ヘッド、インド・バンガロール
+ 91 90359 87074
Ravi.Dattatraya@verolt.com
www.verolt.com

5 Qt SCXMLモジュールは、SCXMLファイルからステートマシンを作成するための機能を提供します

お問い合わせ：
The Qt Company　日本オフィス
〒100-0005
東京都千代田区丸の内3-3-1新東京ビル2F

Web: https://www.qt.io/jp/
Email: japan@qt.io
TEL: 03-6264-4500