The Qt Company

[ガイドブック]
組み込み製品の計画と要件

クリックして開始

はじめに

組み込み製品を構築するのに、これほど恵まれた時代はありませんでした。豊富なオープンソースソフトウェアスタック、活発で利用しやすいコミュニティー、シリコンコンポーネントEMS（電子機器の受託生産を行うサービス）を利用して、洗練された多機能な製品を最小限の時間で開発できるようになりました。

しかし裏を返せば、テクノロジーの選択肢がふんだんにあっても明確な選択基準がなければ、開発プロジェクトが本来の目的から逸れて非生産的な調査プロジェクトとなり、分析のしすぎで身動きがとれないといった状況に陥りかねません。しかも、いわゆるテクノロジーの負け組に期待をかけすぎると、エンジニアリングチームによる再作業と市場投入の延期を余儀なくされ、早期のうちの製品の陳腐化を招いてしまうリスクがあります。

モノのインターネット（IoT）または組み込みプロジェクトの開始当初に陥りがちな情報過多を軽減するため、The Qt Companyでは考慮すべき特に重要な基準を調査し、最もよく使用されるテクノロジーの一覧をまとめてそれらを比較しやすいカテゴリーに分けて評価しました。

読者の皆様が次の製品を構築する際にこのガイドを活かして労力とリスクを軽減できることを願っています。

目次

The Qt Companyの手法

新しい組み込みデバイスまたはIoTデバイスを構築する際、どのようにアプローチしていますか?従来の手法では要件をまとめ、ハードウェアボードを設計し、その上で実行するソフトウェアを開発するという比較的わかりやすいものでした。

ただしこの手法は、現在はうまく機能しなくなっています。競争が熾烈になり、市場投入までの期間が短くなって、顧客の期待値が高まっています。このようなプレッシャーは2つの直接的な影響をもたらします。1つは、ハードウェアではなくソフトウェアが製品設計を決定付けるようになったということです。これは、ソフトウェア開発が多くの時間とリソースを必要とし、デバイスの全体的なカスタマーエクスペリエンスの成否を左右するからです。もう1つは、企業の俊敏性がかつてないほど求められているということです。企業は絶え間のない変化に備えなければなりません。

最新の組み込み製品を開発するには、4つの非常に重要なステップを実行する必要があります。

最新の組み込み製品を開発するための4つの重要なステップ

1 スコープを設定

通常の製品開発と同様に、製品の機能セットとユーザビリティを説明する**市場要件をまとめる**。製品のソフトウェアが対応する範囲について経営陣、マーケティング、サポート、エンジニアリング部門と綿密に協議する。顧客と対話して競合製品を把握。これらすべてを踏まえ、要件への追加を検討すべき「将来性(“future-proof”)」要素を策定。

2 プロセスを判定

ソフトウェア開発者、UXデザイナー、ハードウェアエンジニア間のやり取りを把握。開発を行うワークフローとツールにおいて、一貫して効率的で信頼性の高い製品開発がサポートされているか、チームとしての迅速な行動とすばやい順応能力が促進できるかを確認。

3 ソフトウェアを選択

製品が実行できることと、その実行環境は、製品のソフトウェアスタックによって決まるため、サポートするハードウェアの範囲が最も広く、信頼できるサポートとメンテナンスプランがあるソフトウェアを選択。ライブラリやコンポーネントへの外部依存関係が最も少ないソフトウェアを選択。業界標準、安定したAPI、再利用可能なソフトウェアコンポーネントを備えたソフトウェアを選択。

4 ハードウェアを選択

要件とソフトウェアに適合するハードウェアプラットフォームを選択。従来の基準はコスト、信頼性、確実なデリバリーです。拡張と縮小の両方が可能なハードウェアプラットフォームが見つかればさらに大きなメリットが得られます。

スコープの設定

組み込みプロジェクトに関して、コードを書きはじめたり、最初のコンポーネントを選択したりする前に答えを見つけるべき多くの問いがあります。

これらの問いはすべて、プロジェクトの包括的なスコープは何か、現在の選択に基づいて将来の製品ロードマップも構築できるか、という問いに帰着します。プロジェクトに関連するあらゆることを理解していなければ、再設計と改良に膨大な時間を浪費しかねません。

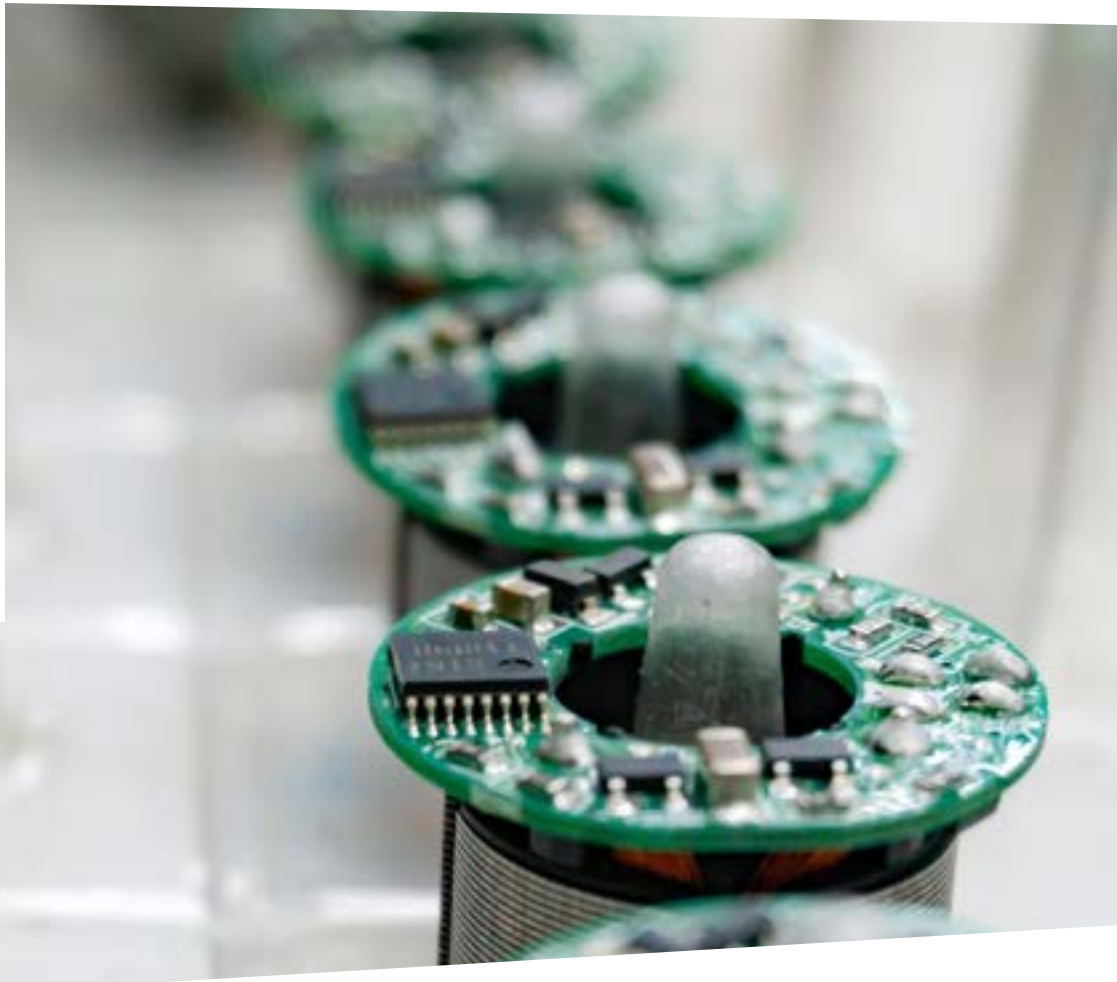
このセクションでは、見逃されがちなオプションも含めて、予備計画段階で詳細を詰める必要がある重要な選択肢を取り上げます。

製品のライフサイクル

優秀な製品を構築することと、初期デプロイの後、変化の激しい市場で長期にわたって製品の妥当性を維持することは分けて考える必要があります。新機能やバグ修正を追加するためのメンテナンスリリースは必ずしも計画できるというわけでもなく、ハードウェアの廃止、サイバーセキュリティーの更新、またはユーザーの期待の変化によってメンテナンスリリースを余儀なくされる場合があります。

製品のメンテナンスには、信頼できる安定したソフトウェアコンポーネントが必要です。アップデートが必要になるまで強制的アップデートを回避できるように、サポート期間が非常に長いソフトウェアを探すことをお勧めします。

最先端の組み込み製品を開発する場合には、ハードウェアの陳腐化やユーザーの好みの変化にも対応しやすく、安心感を抱ける強力なクロスプラットフォームサポートを探しましょう。



再利用可能性とクロスプラットフォーム

良いコードを書くのは簡単ではありません。そのため、できる限り多くのプラットフォームで、できる限り頻繁にコードを再利用しましょう。ソフトウェアスタックの実行基盤となるプロセッサーアーキテクチャ、プラットフォーム、オペレーティングシステムを決定します。ソフトウェアの外部依存関係を少数の適切に管理されたモジュールに集中させます。そうすれば、再利用がさらに簡単になります。

すべてのソフトウェアコンポーネントがすべての組み合わせで動作するとは限りませんが、カスタムして無理矢理動かす方法を考案することはできる限り回避するのが得策です。また、オープンソースを支援するのは素晴らしいことですが、必要なハードウェア/ソフトウェアの組み合わせに対応できるようにするという目的だけのためにプロジェクトで使用するオープンソースモジュールに無理に貢献しようとすることは避けましょう。必要なものをすでにサポートしているソフトウェアを選択することが重要です。

抽象化が必要なケース

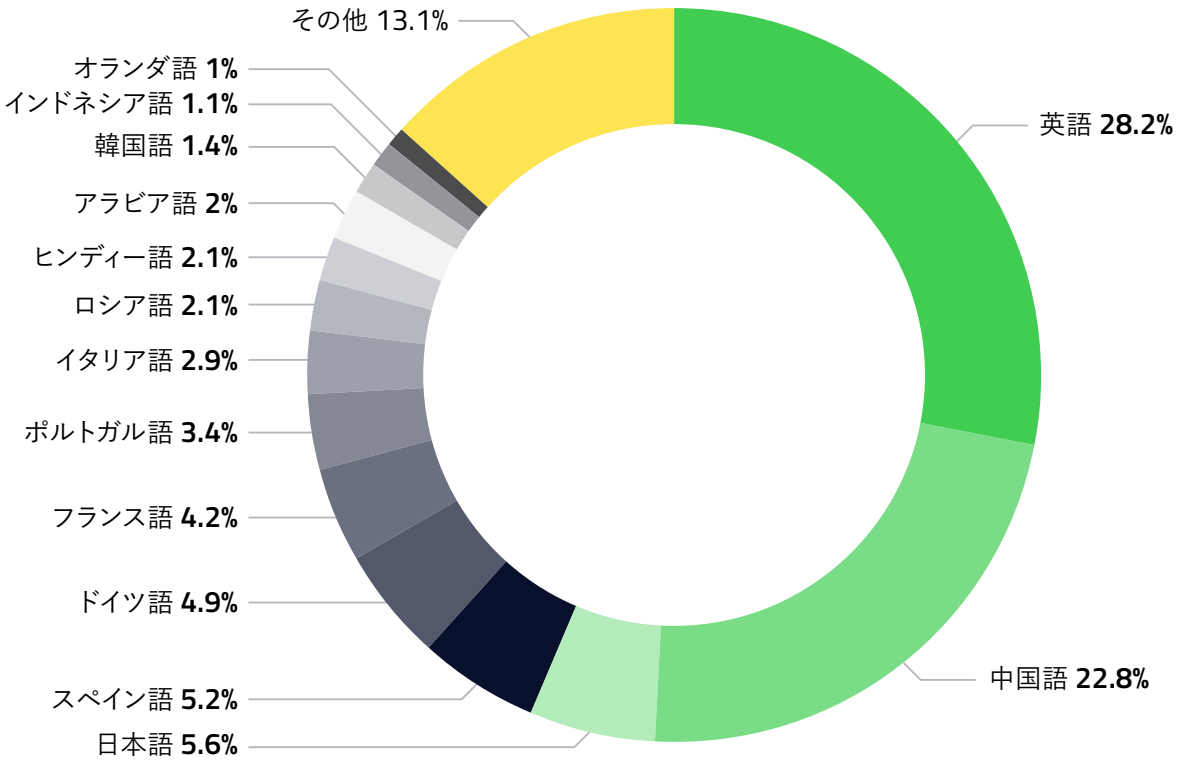
通常は、クロスプラットフォームに対応しているソフトウェアを使用しない場合に、ソフトウェアを変化する依存関係から隔離するため抽象化レイヤーを作成する必要があります。もちろん、これには追加の事前作業が必要になります。ただし、適切な抽象化を早期に実装すれば、事後にコードベース全体を改良する場合と比べて大幅に時間を節約することができます。

また、すべてのものに抽象化が必要なわけではありません。ハードウェア、OS、またはサードパーティー製ソフトウェアコンポーネントを交換する際に最も変更される可能性が高い領域に集中してください。また完璧である必要もありません。どれだけ念入りに設計しても、抽象化が将来のすべての変更に対応できるとは限らないからです。方向転換するときに微調整が必要になっても問題ありません。

国際化

製品を単一の言語または地域に限定する予定ですか?グローバルなマーケットと市場シェアを拡大するチャンスを放棄するという選択肢は、考えにくいことでしょう。たとえ初期リリースが日本語対応のみであっても、必ず国際化(i18n)への移行を簡単にする言語、ツール、およびUIフレームワークを使用してコードを開発することを強くお勧めします。この際、開発者と翻訳者が簡単にリソースを利用できる文字列定数と、ビジネスロジックから切り離されたUIを使用することをお勧めします。

言語別の国内総生産(GDP)



利用可能な最新の推定値:**Unicode Technical Note #13**

接続性、アップデートのしやすさ、セキュリティー

当然のことながら、現在のほとんどの組み込みデバイスは、特にセキュリティーパッチ、バグ修正、機能アップデートに必要な無線（OTA）アップデートをサポートするための接続性を必要としています。更新がデバイス、バックエンド、ユーザー、またはこれら3つの組み合わせのどれによって実行されるかを決定してください。多くの開発者は独自のOTA更新スキームを作成しようとしますが、非常に複雑で、かつ重要性が求められるため、代わりに商用のすぐに使用可能なソリューションを使用することを強く検討すべきです。オープンソースを利用する場合は、ハードウェアをサポートするだけでなく、そのロードマップにおいてアクティブなメンテナンスも担えるプロジェクトを選択しましょう。OTAソリューションに必要な追加のストレージを計画することも忘れないでください。一部のソリューションでは、フラッシュのフットプリントが2倍以上になるので注意が必要です。

接続にはソフトウェアセキュリティーが必要であり、このセキュリティーは後付けではなくビルトインのものでなければなりません。ファイルのアップロード、ユーザー入力、ネットワーク接続に関しては、ソフトウェアを最悪の事態から保護する必要があります。ソフトウェアエンジニアがサイバーセキュリティーのトレーニングを受けていることを確認し、必要に応じてサイバーセキュリティーのエキスパートと連携しましょう。ハードウェアにもセキュリティーが必要です。開発、デバッグ、またはテストのために特別に追加されたポートまたはコンポーネントを本番システムに設定しないでください。また、ハードウェアロックとSecure CoreやTrustZoneなどのセキュリティー機能を活用してください。

これらのOTA機能を探す

- OTAの誤用を防止するための認証および暗号化手段
- 信頼性の高い配信のためのパッチ検証、画像確認、および再試行メカニズム
- 自動パッチ生成のための統合ビルドツール
- フィールド内のソフトウェアバージョンと更新試行を追跡するための診断レポート
- パッケージサイズを最小限に抑えるための差分計算と圧縮
- サーバー負荷を軽減するための配信抑制とバリアントのセグメント化
- デプロイ失敗の緊急事態に対処するためのロールバック機能

モバイルコンパニオンアプリ

いつでもユーザーのモバイルから接続することができ、リモートアクセスを提供し、企業が顧客とつながる新しい方法を構築することができるという理由で、現在の組み込みデバイスは、モバイルアプリで接続できるケースが一般的です。ただし、2つの独立したアプリを異なるプラットフォームでコーディングすると、作業負荷が2倍になります。つまり、開発者がモバイル開発を行ううえでSwiftやKotlinにどれだけ魅力を感じていようと、Apple、Android、組み込みプラットフォームで使用できる単一の言語とツールチェーンを選ぶのが最も簡単です。そうすることで、モバイルアプリを一度だけ作成すれば、あらゆるデバイスで開発リソース、グラフィカル資産、ユーザビリティ設計、コードベースを共有することができます。

クラウドおよびその存在を越えて

言うまでもなく、IoTまたは組み込み製品のために検討できる機能は他にも多数あります。ここでは、そのいくつかをご紹介します。

クラウド活用

デバイスの機能実装をエッジとクラウドで分割し、フィールドですべての製品にインスタント機能を追加できるようにするとともに、製品のロジックとリソースの一部を拡張性のある代替デバイスにオフロードできるようにします。

統合人工知能（AI）

ハードウェアの欠陥、ログ情報、使用パターンに関するデータを収集し、機械学習を使用してより良いUIとスマートな動作を兼ね備えた製品を構築します。

デジタルツイン

デバイスの状態とステータスに関するデータを収集し、デスクトップまたはモバイルで**物理デバイスを仮想化**できるようにします。

ブロックチェーン

検証可能かつ変更不能の信頼できるトランザクションレコードを複数のシステムに分散することで保護を強化します。

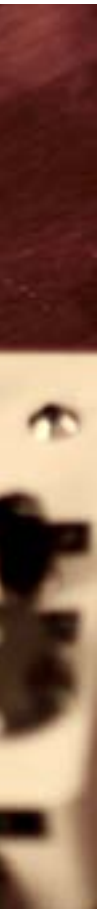
ご存じでしたか？

ブロックチェーンは通常は通貨に使用されますが、信頼性の高いデータストレージが重要な場合にはあらゆるケースで組み込みデバイスにも役立ちます。

- コンプライアンスおよび規制確認のためのセンサーデータの記録
- サードパーティー認証のためのメッセージの検証
- プロセスを監視するための改ざん防止データのログ記録
- 証明のための製造情報の保存
- 測定の確実性を確保するための校正記録の共有

開発プロセス

プロジェクト開始時の目標は、品質を損なわずに（あるいは向上させつつ）開発時間を短縮するワークフローを開発することです。これには、UX設計、ソフトウェア開発、ハードウェアエンジニアリング、品質保証（QA）の4つの主要機能の間でワークフローを合理化することが含まれます。また、優れたツール、共同設計、アジャイル手法についても検討してください。

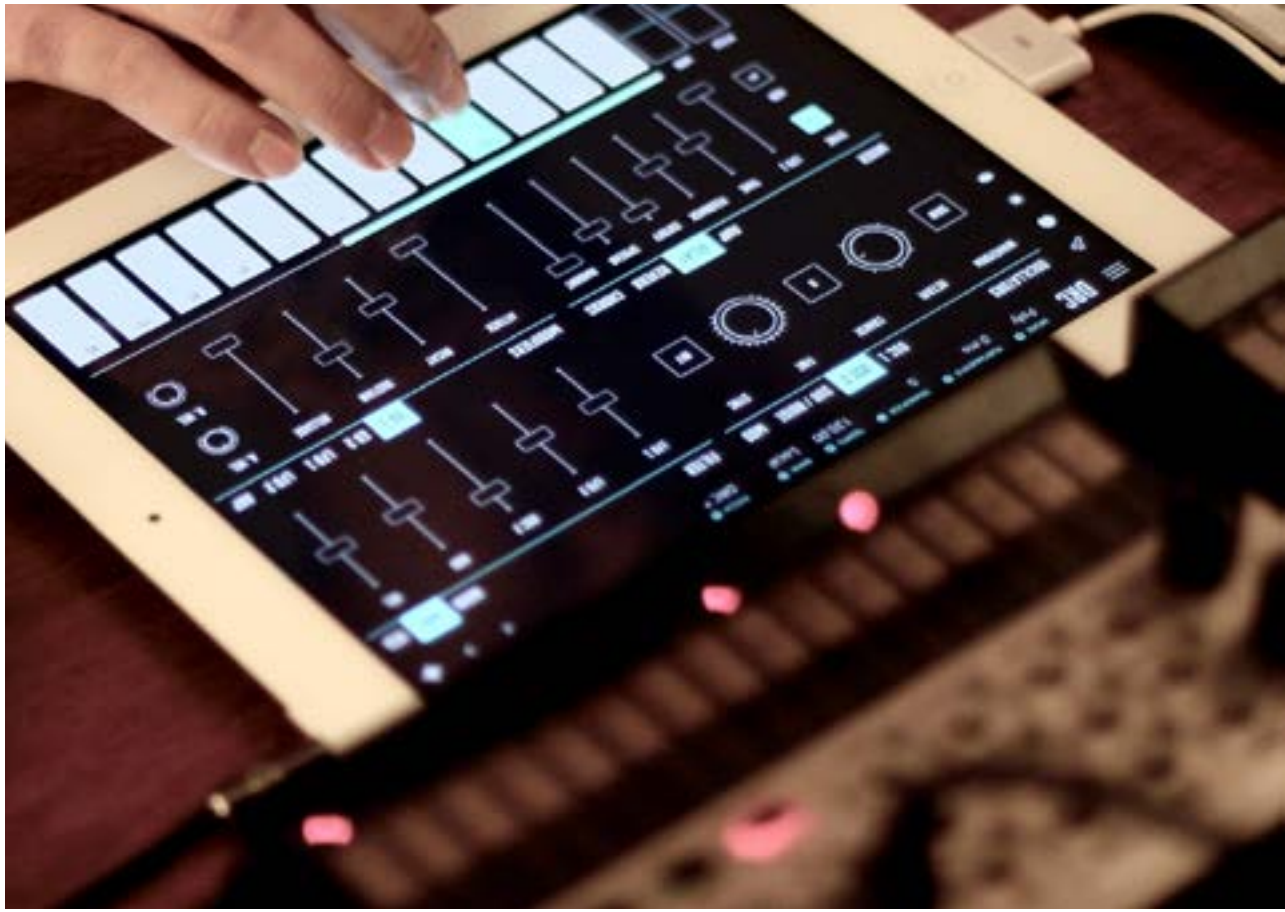


ソフトウェア開発

ソフトウェア開発ツールチェーンは、新しい開発者が素早く作業を習得し、経験豊富な開発者が質の高いコードを大量に作成できるように、理解しやすく、効率的に使用できるものでなければなりません。ツールは、小さなバイナリー、高速な実行可能ファイル、リソースを多用しないランタイムなどの最適な結果を生み出すものでなければなりません。多くの優れたオープンソースオプションが利用可能となっていますが、開発の合理化に役立つツールが見つかったらそのツールに費用をかけることを惜しまないでください。投資に見合う価値があります。

迅速でありながらも安定したツール

チームが新しいプログラミングパラダイム、より効率的な手法、新しいハードウェアのサポート、非常に重要なバグ修正にアクセスできるように、アクティブな開発に使用できるツールをお選びください。アクティブなサードパーティー製ツールを必要とする一方で、使用するソフトウェアツールがあまりにも早く、または予期せず変更され、開発に余分な摩擦を生じさせることがないことを確認してください。ほとんどのツールを同じ場所から調達すれば、チーム内での配布と適合性の問題を最小限に抑えることができます。



ビルド

ビルドプロセスはソフトウェア開発における影の主役といえます。一貫性があり、高い信頼性をもち、追跡されたソフトウェア構築を行えるようにしましょう。ほぼすべてのビルドシステムにカスタムコンポーネントがありますが、使用するビルドツールとスクリプトはできる限り業界標準で、よく使用されているツールものを選ぶのが良いでしょう。これによりチームが抱えている問題は他のユーザーによってすでに解決されている可能性が高くなり、既知のソリューションを最大限に活用することができます。

テスト

適切なテストフレームワークがないと、プロジェクトが頓挫する可能性があります。選択したツールスイートがテストの枠組みを確立して継続的インテグレーション（CI）のニーズを満たすのに役立つという信頼感が増せば増すほど、手動で作成する必要性が軽減されます。既存の商用またはオープンソースツールを使用することのもう1つの重要な利点は、これらのツールが内製したソリューションよりも機能が豊富になり適切にメンテナンスされるということです。統合前に、ひいてはユーザーに指摘される前に問題を見つけられるよう、テストは割愛しないようにしましょう。

UX設計とUXワークフロー

現在、ユーザーエクスペリエンス（UX）は、製品を差別化するための最も確実な方法となっています。ユーザーは、有益なエクスペリエンスが得られたときにブランドの熱心なファンになります。最適なユーザーインターフェースは、デザイナーと開発者の両方の才能を組み合わせることで生み出されます。また、プロジェクトで同時に作業するために両者のスキルセットをサポートするフレームワークの活用も非常に有効です。

設計と開発の間の反復ワークフローは、以下のような工程におけるタイムロスを回避することで、良い製品をより迅速に構築することが可能になります。

- **デザイン画面の再実装**
- **仕様の誤解や不完全な引継ぎから生じるデザインエラー**
- **従来のUXワークフローにおける柔軟性に欠けるプロセスによるユーザビリティの阻害**

デザイナーは、つかいなれたデザインツールで開発したアートワークを用いて、すべての重要なユーザーエクスペリエンスを作成、可視化、プロトタイピングできなければなりません。これらのツールでは、デザイナーがソフトウェア開発プロセス全体で改善を施したUIデザインをテストおよび統合できるように、ユーザーインターフェース（UI）とビジネスロジックを明確に分離してサポートしている必要があります。

社内でスムーズなUXワークフローを成功させるには、適切な企業文化が必要です。UXがプロジェクトの成功にどれだけ重要か、組織内でUXをサポートする文化を育てるにはどうすればよいかに関する詳細は**こちらのホワイトペーパー**をご確認ください。

早い段階から実際のターゲットハードウェアで実行できるデザインプロトタイプを作成することができると、部門間での不要なフィードバックループを排除することができます。デザイナーは、実際の組み込みハードウェア上でUIのデザインをすることで、画面サイズ、アスペクト比、色深度などの実際の見映えを確認できます。さらに重要なのは、これによってパフォーマンスに関するフィードバックが即座に得られることです。UIデザインは、多くの場合、対応する組み込み機器よりもパフォーマンスが高い携帯電話によって駆動されるため、デザイナーはいつUIが過剰になり、いつパフォーマンス不足になるかを把握し、デザインを完了する前にUIを調整できます。

統合開発環境

統合開発環境（IDE）は、プログラマーがコーディング、テスト、デバッグに要する時間の大半を費やす場所です。ただし、優れたIDEではそれ以上のことを行うことが可能です。

統合

ソフトウェア開発プロセス全体（設計、記述、構築、デバッグ、プロファイリング、テスト、ローカライズ、デプロイ）をサポートできる十分に統合されたツールをお探しください。実際のターゲットや何らかの形態のターゲットエミュレーションと連携するIDEを使用すれば、調査対象のプロトタイプハードウェアが十分にない場合でも開発を継続できます。

GUI設計

デザイナーは、WYSIWYG（What You See Is What You Get、見たままを得られる）ツールを使用して製品のグラフィカルユーザーインターフェース（GUI）を作成し、コントロールと視覚的要素を備えた画面を構築します。GUIデザインは、ワークステーションとターゲットの両方でIDEが直接サポートする必要があります。これにより、プロトタイプの作成とユーザビリティの微調整の際に大幅に時間を節約できます。デザイナーが特別なトレーニングなしにUIを構築できる環境を見つけることも必要です。

アセットのインポート

アーティストがアセットをプロジェクトに直接組み込めるように、デザイナー向けツール（Adobe Photoshop、Autodesk 3ds Max、Blender、Figma、Maya、Modo、Sketchなど）と統合できるIDEをお探しください。UIアクション、アニメーション、インタラクションもインポートする機能がツールにあれば、これらのUI要素を開発者が再実装するのではなく、デザインチームが定義できます。

プロジェクト管理

バージョン管理およびビルドシステムは、華やかではないかもしれませんが、信頼できるソフトウェアと生産的な開発環境においては不可欠です。リビジョン管理システムやビルドツールとすぐに統合できるIDEがあれば、エンジニアの作業負荷が減り、作業効率が向上します。

拡張

優れたプラグインエコシステムは、IDEがお客様の社内ワークフローとツールをサポートするのに役立つとともに、開発者がIDEを自分の好みに合わせてカスタマイズし、開発時間を短縮できるようにします。静的コード分析とパフォーマンス指標用のプラグインは、ソフトウェアの修正および最適化に欠かせない重要な知見を開発者にもたらします。

メンテナンス

製品メンテナンスリリースをどのように管理するべきでしょうか？お客様固有のバグや機能アップデートについてはお話しませんが、使用する外部ソフトウェアの変更について触れておきたいと思います。製品に含める依存関係（オペレーティングシステム、ドライバー、ライブラリ、フレームワーク、ツール）はすべて、お客様のコントロールの及ばないペースで変更されます。これにより、アプリケーションプログラミングインターフェース（API）の破損、アプリケーションバイナリーインターフェース（ABI）の競合、互換性のないツールチェーン、ソフトウェアの廃止などの問題が生じる可能性があります。

古いリリースに対しても長期のサポート、バグ修正、サイバーセキュリティーパッチを提供するソフトウェア構成要素をアプリケーションに組み込むことを検討しましょう。この点を考慮に入れると、サードパーティーの組み込みでは、活発な開発者コミュニティーの存在が重要になります。

依存ソフトウェアを最新の状態に保つべき7つの理由

1. バグの減少
2. サイバーセキュリティーの改善
3. パフォーマンスの向上
4. アップグレードが迅速かつ簡単
5. 追加の機能
6. システム間の互換性の確保が容易
7. より良いサードパーティーサポート

コミュニティーとドキュメント

最後に、大きなエコシステムを見てみましょう。ソフトウェアツールがどれだけ広くサポートされているかにより、難しい問題を解決したり、すでに利用可能な互換性のあるライブラリを見つけたり、新しい人材を採用したりする際に大きな違いが生じる可能性があります。

オンラインでなされる会話、stack overflowで設置されているキーワード、Gitサンプルコードリポジトリ、求職カテゴリーなど、複数の分野で十分なサポートがあるソリューションを探しましょう。お気に入りのツールチェーンに使用するプラグイン、アプリケーション、コマンドラインツール、サードパーティー開発者の数が多いほど、必要なソリューションを誰かがすでに作成している可能性が高まります。したがって、詳細な文書、API例、オンラインヒント、さらには請負業者の採用などに関して役立つ情報を素早く見つけることができます。

ソフトウェアスタックコンポーネントの必須条件

- ソースコードの入手可能性
- ベンダーロックインがない
- 総所有コストが低い
- 製品化期間が短い
- 力強い開発者コミュニティー
- 十分にサポートされたツールエコシステム

ソフトウェアスタック

製品の核となるのはそのソフトウェアです。使用するソフトウェアビルディングブロックは、製品の機能のスコープと制限を定義します。

開発環境を定義するソフトウェアスタックで最も人気のある選択肢をいくつか見てみます。

OS

最も影響力が強い可能性があるソフトウェアスタックのコンポーネントはオペレーティングシステムです。選択したOSにより、システムに組み込むことができるソフトウェアが決まります。OSは、一部のタスクを容易にすると同時にそれ以外のタスクを非常に難しくすることもあります（ただし、ベアメタルを採用し、自分ですべて行うことと比べれば格段に簡単です）。ここでは、組み込み市場で最も一般的な8つのオペレーティングシステムをご紹介します。

	Embedded Linux	Android	QNX Neutrino RTOS	GreenHills INTEGRITY	Wind River VxWorks	Amazon FreeRTOS	webOS	Windows for IoT
デプロイのしやすさ	★★★★	★★★★★	★★★	★★	★★★	★★	★★★★	★★★★
効率性	★★★	★★	★★★★	★★★★	★★★★	★★★★★	★	★★★
決定的動作	★★★	★	★★★★★	★★★★★	★★★★★	★★★★	★	★★
API	POSIX	POSIX	POSIX	POSIX	POSIX	FreeRTOS	POSIX	UWP
接続性	★★★★★	★★★	★★★	★★★★	★★★	★★	★★★★★	★★★★★
グラフィックス	★★★★	★★★★★	★★★★	★★	★★	★	★★★★	★★★★★
ハードウェアサポート	★★★★★	★	★★★	★★	★★★★	★	★★	★★★★
オープンソース	はい	はい	いいえ	いいえ	いいえ	はい	はい	いいえ
コミュニティー	★★★★★	★★★★	★★★	★★	★★	★★	★	★★★
ライセンス価格	$	$	$$$$	$$$$	$$$	$	$	$$$$$
カスタマイズおよびハードニングコスト	$$$	$$$$	$$	$$	$$	$$$	$$$	$

★が最低ランクで、★★★★★が最高ランクです。「付録A」でテクノロジーの頭字語を参照してください。

本書のチャートは、ソフトウェアとハードウェアの長所を市場力学に影響されにくい方法で記載し、一目で各コンポーネントの本質をとらえるのに簡単な評価を表しています（開発のペースの速さを考慮すると、包括的なガイドを完全に最新の状態に保つことはできません）。これらのチャートの代表的なユースケースには、大規模（産業、医療、自律システム）、中規模（インフォテイメント、白物家電、キオスク）、および小規模アプリケーション（携帯デバイス、IoT、ウェアラブル）が含まれます。あるコンポーネントで複数のオプションが利用可能な場合は、最も高性能なバリアントのデータを記載しています。本書では、最も一般的なハードウェアの選択肢をいくつか選びましたが、完全な製品リストについては各サプライヤーにご確認ください。

コンテナーとハイパーバイザー

近年の強力なプロセッサーにより、デスクトップとクラウド向けのソリューションを組み込むことができるようになりました。これには、製品のソフトウェアアーキテクチャを分割する2つの強力な手法であるコンテナーとハイパーバイザーが含まれます。

コンテナーを使用すると、ツール設定を即時かつ一貫したものにしたり、テストフレームワークを標準化したり、複数の独立したツール/ライブラリ構成を保持したりできます。これにより、**組み込みプロジェクトの開発を加速**できます。コンテナーは、わかりやすいデバイスプロビジョニング、簡単なハードウェアベンダーのアップデート、同じコードベースからの優れたテストに役立つため、**ターゲットで使用する**際に有用な構成要素にもなりえます。

ハイパーバイザーは主に、保護、サンドボックス化、独立した操作を提供することでターゲットを支援します。ハイパーバイザーを使用すると、1つのチップで**複数のオペレーティングシステム（OS）を同時に実行**できます。たとえば、システムクリティカルまたはセーフティークリティカルな機能をQNX Neutrino RTOSやGreenHills INTEGRITY RTOSなどのOSで、製品のメインUIをEmbedded Linuxなどの2番目のOSで、ダウンロード可能なサードパーティー製アプリをAndroidやWebOSなどの3番目のOSで実行できます。ハイパーバイザーは、機能的安全性要件があるデバイスを構築する際にも決定的な違いを生み出します。

言語

OS上にアプリケーションを構築するには、プログラミング言語が必要です。どのプログラミング言語にも、開発プロセスに反映されるにしたがって、組み込みアプリケーションの開発に影響する長所と短所があります。このような制約には、各言語がサポートできるGUIのタイプが含まれます。ここでは、組み込み業界の主要な候補をいくつか検討します。

	C	C++	Java	Python / MicroPython	JavaScript/ HTML5/CSS	Rust
Strengths	組み込み、ベアメタル、IoT	組み込み、ベアメタル、スタンドアロンアプリ、IoT	Web、クラウド	クラウド、データサイエンス	Web	組み込み、ベアメタル
デプロイのしやすさ	★★★	★★	★★★	★★★★★	★★	★★★★
表現力	★	★★★★	★★	★★★★★	★★★	★★★★
メンテナンスのしやすさ	★★★★	★★★★	★★★★	★★★	★	★★★★
寿命	★★★★★	★★★★	★★★	★★★	★	★
実行時の効率性	★★★★★	★★★★★	★★★	★★	★	★★★★
ライブラリ/モジュールが利用可能	★★★★	★★★★	★★★	★★★★	★★★★	★★
低レベルインターフェース	★★★★★	★★★★★	★★	★★★	★	★★★★
接続サポート	★★	★★★ / ★★★★★[1]	★★★★	★★★★★	★★★★★	★★
グラフィックスサポート	★★★★	★★★★★	★★★	★★★	★★★★★	★
開発者コミュニティー	★★★	★★★	★ / ★★★★[2]	★★★★	★★★★★	★★

★が最低ランクで、★★★★★が最高ランクです。「付録A」でテクノロジーの頭字語を参照してください。

1 プラットフォームライブラリが含まれている場合に高レベルの接続性およびネットワーキングサポートを提供
2 Android（低バージョン）を除き、Android（高バージョン）を含むJava開発者コミュニティー

GUIフレームワーク

テクノロジースタックで最重要となる要素の1つはグラフィカルユーザーインターフェース（GUI）です。アプリケーションで使用するGUIフレームワークは、アプリケーションの外観だけでなく、実装可能な機能、製品を市場に投入するまでの時間、サポート可能なプラットフォームにも影響します。事前に構築され、テストされた既存のコードを活用すると、自分で設計、コーディング、テストを行う必要がなくなります。

クロスプラットフォーム

GUIフレームワークの主な特性の1つは、そのクロスプラットフォーム機能です。GUIフレームワークがサポートするプラットフォームの数はいくつでしょうか。GUIフレームワークはそれらすべてのプラットフォームを一貫してサポートするでしょうか。GUIフレームワークは、ハードウェアを交換する場合、複数のOSをサポートする必要がある場合、または組み込み製品とモバイルまたはデスクトップアプリの間でソースを共有する場合に必要になります。

成長と変化は避けられないため、現在のプラットフォーム要件だけに基づいて選択することは避けましょう。さまざまなデスクトップOS（Linux、MacOS、Windows）、モバイルOS（Android、iOS）、組み込みOS（QNX、INTEGRITY、Linux/Wayland、Linux/X11、UWP、VxWorks）のほか、多数のハードウェアアーキテクチャとプラットフォームをサポートするGUIフレームワークをお探しください。

製品開発に役立つクロスプラットフォームGUIの強み

製品化期間

事前にパッケージ化された豊富な機能、多数のプログラマー指向機能、アクティブなサポートコミュニティーを備えたツールがあると、開発者は生産性を劇的に向上させ、製品をいち早く市場に投入できるようになります。

統合されたエクスペリエンス

優れたツールは、ユーザーが使用するプラットフォームに関係なくユーザーに同じエクスペリエンスをもたらし、ユーザーがすべてのプラットフォームを一貫して使用できるようにします。

コードの再利用

プラットフォーム間で機能、API、ビルド環境、資産管理が一貫していれば、開発者は組み込み、モバイル、デスクトップの各ソリューション間でコードを容易に再利用できます。

開発のしやすさ

強力なクロスプラットフォームツールチェーンがあれば、開発者はサポート対象のターゲットごとの開発または構築を強いられることなく、自分が高い生産性を発揮できる使い慣れた環境を使用できます。

コンポーネント

どのGUIフレームワークでもサポートされるのはウィジェットだけではありません。サポートされる各種のコンポーネントもここでご確認ください。2Dおよび3Dのデザインと可視化、組み込みのブラウザー、周辺機器のサポート、一般的なネットワークプロトコル、国際化、完全なマルチメディアサポート、データベースの統合、センサーアクセス、チャート作成などの豊富なコンポーネントオプションを備えたフレームワークを見つけましょう。どのような構築済みのオプションでも、チームの開発負荷を軽減するのに役立つ可能性があります。

言語

フレームワークは、さまざまな言語向けのAPIを提供します。これにより、開発者チームは自分たちが最も生産性を発揮できる言語で記述できます。また、開発者は作業に最適な言語を選択できる柔軟性も得られます。たとえば、フレームワークでC++とPythonの両方がサポートされる場合、チームはパフォーマンスが最優先のときはC++、開発速度が優先されるときはPythonを選択するといったことができます。

開発サポート

独自のIDEを持っているか、一般的なIDE用のプラグインを備えているGUIフレームワークを探します。UIの設計、構築、テスト、デバッグが容易であることが不可欠です。専用の開発環境も鍵となります。

ルックアンドフィール：
OS、フレームワーク、またはカスタム

製品のルックアンドフィールのメリットについては、常に議論が分かれます。OS固有のウィジェット、フレームワークによって提供されるコントロール、カスタムルックのどれを使用すればよいのでしょうか。残念ながら、プラットフォーム提供のUIは常に使用できるとは限りません。多くの組み込み製品でLinuxが使用されていますが、LinuxにはデフォルトのUIはありません。言うまでもありませんが、複数のOSで製品を実行する場合、プラットフォームはそれぞれ異なった外観になります。当社では、お客様によるカスタム構築やフレームワークを通じた提供などの区別を問わず、広く理解されているクリーンで魅力的なUIを選ぶことが、通常は総合的な最適妥協点になると考えています。このようなUIは、製品ライン全体にわたって一貫性のあるルックアンドフィールを維持するのに役立ちます。

ヘッドレスUI

組み込みデバイスが画面を持たないほうが良い場合はあるのでしょうか。特にIoT製品の場合、一般的な答えは「イエス」です。ヘッドレス製品は、そのUIをユーザーのブラウザーに配信し、製品が素晴らしいユーザーエクスペリエンスをはるかに低コストで提供できるようにします。

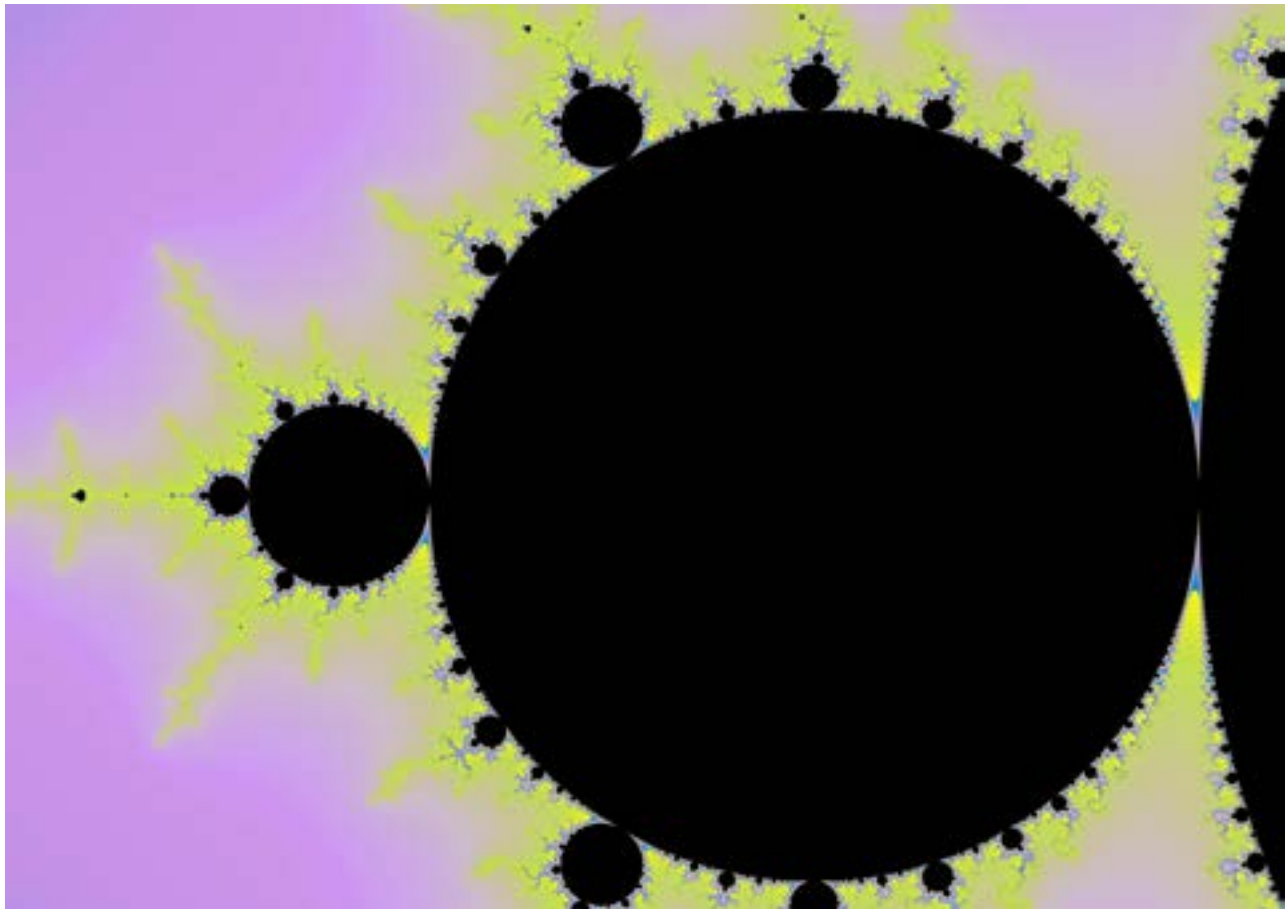
未加工のHTML、CSS、JavaScriptからリモートUIを構築する必要はありません。また、そうすべきでもありません。魅力的なブラウザーベースのUIを開発する際の労力を軽減するいくつかのJavaScriptフレームワークの利用をおすすめします。一部のGUIフレームワークでは、WebGLまたはWebAssemblyのいずれかを使用してUIコードをエクスポートし、Webベースではないツールを使用してヘッドレスのブラウザーベースのエクスペリエンスを提供することも可能です。

ライブラリ

アプリケーションソフトウェアにはGUI以外にも多くのものが必要です。製品には、事前に構築およびテストされた多数のライブラリが必要になる場合があります。たとえば、ネットワークプロトコル、JSONおよび/またはXML解析、機械学習、行列演算、高速フーリエ変換（FFT）、乱数ジェネレーター、暗号化、マルチスレッドのサポート、画像処理、正規表現、データベースのサポートなどです。

これらのほとんどは単純ではありません。そしてコードが複雑になるほど、コードを自分で記述する代わりに長年の実績をもつソフトウェアを活用するべきです。GUIフレームワークにすでに組み込まれている多数の、事前にパッケージ化されたライブラリを使用できる場合があります。実装言語によっては、Boost、Apache Portable Runtime（APR）、NumPy、TensorFlowなどの十分に信頼できる外部ライブラリを多数組み込むことも必要になる場合があります。

必要な機能が何であれ、互換性があってわかりやすいAPIと優れたドキュメントを備え、ニーズを総合的に満たす最小限のライブラリセットをお探しください。



ハードウェア

ソフトウェア要件を調べ終えた後は、ハードウェアについて検討しましょう。多くの製品機能と今後のロードマップにより、ハードウェアの選択肢が決定されます。たとえば、3Dアニメーションが必要な場合は、ハードウェアアクセラレーション方式のグラフィックスを備えたものを選ぶと良いでしょう。高速モバイル接続が必要な場合は、WiFi ac、USB、またはその両方が必要になります。ジェスチャー、タッチ、または音声を含む対話モデルがある場合は、必要なハードウェアとセンサーを含める必要があります。

どのようなハードウェア選択を行う必要があるか見てみましょう。

製品ラインと拡張性

組み込みシステムの背後にあるチップの機能は、16個のコアを搭載した3 GHz 64ビットプロセッサーやハードウェアアクセラレーション方式のマルチディスプレイ4Kグラフィックス処理ユニット（GPU）から4つの入力/出力（I/O）ピンを備えた8ビットマイクロに至るまで多岐にわたります。

この時点では、製品の今後の要件の範囲について十分な理解をお持ちのはずです。広範な製品スコープを互換性のあるパーツファミリー内でカバーできるハードウェアを最初にお選びください。ハイエンドオプションとローエンドオプションの両方を備えたシステムとアーキテクチャを検討するのが最善です。製品ラインの複数の製品にわたって再利用できるコンポーネントをお選びください。そうすることで、ボリュームの増大というメリットが得られます。寿命の長い製品を構築する場合は、選んだパーツがこの先何年もの間利用できることを確認してください。

ただし、ソフトウェアスタックがハードウェアに求める最小限の処理、速度、およびメモリー要件を忘れないようにしてください。コストを抑えようとするあまり、性能の劣るハードウェアを選ぶことがないように注意してください。開発サイクルの終わりにハードウェアの移行を余儀なくされ、かえってコストがかさむことになります。

SoC/CPU

システムオンチップ（SoC）の中央処理装置（CPU）は、組み込みシステムでは最も重要で、最も影響が大きく、最も高価となる可能性の高いハードウェアコンポーネントです。また、製品ファミリー全体にとっての決め手になる可能性もあります。以下の表に、最も高性能で一般的ないくつかの市販SoC評価ボードを示します。

	AMD	Intel	NXP	Renesas	TI
評価ボード	Kontron D3713-V4 mITX	Avnet BOX NUC6CAYS AJL	NXP i.MX 8 QuadMax MEK	R-Car H3 Starter Kit	TI AM572x EVM
SoC*	AMD Ryzen V1807B	Intel Apollo Lake (Celeron J3455)	NXP i.MX8	R-CAR H3	AM5728
メモリー	4 x DDR4、2 x DIMM（最大32GB）、ECCオプション	2GBオンボードDDR（最大8GB）、32GB eMMC	LPDDR4 (x64), 32GB eMMC	384Kシステム RAM、DDR4、80MBオンボードフラッシュ、8GB eMMC、microSD	2GB DDR3L、4GB eMMC、microSD
周辺機器	HDMI、4 x DP++、4K、Vega GPU、PCIe、SATA、オーディオ、GigE、USB 3.1+2.0	HDMI、Intel HD 500 GPU、オーディオ、GigE、PCIe、SATA、USB 3.0 + 2.0、WiFi ac、BT	MIPI、LVDS、4K、PMIC、GPU、PCIe、オーディオ、GigE、USB 3.0、CAN、最大4台のHDMIディスプレイ	HDMI、PowerVR GPU、LVDS、WiFi、BT、オーディオ、Eth 10/100、USB 2.0	7インチ静電容量式タッチスクリーン、HDMI、オーディオ、PowerVR GPU、GigE、SATA、MiniPCIe、USB 3.0
アーキテクチャ	x86-64	x86-64	ARM-64	ARM-64	ARM-32
処理能力	V1807B x 4コア@ 3.8GHz	J3455 x 4コア@ 1.5GHz	A72 x 2コア@ 1.6 GHz、A53 x 4コア@ 1.2 GHz、M4 x 2コア@ 266 MHz	A57 x 4コア@ 1.5 GHz、A53 x 4コア@ 1.2 GHz	A15 x 2コア@ 1.5GHz、M4 x 2コア
ユースケース	大規模	大規模	大規模	大規模	中規模
消費電力	★	★	★★	★★	★★★
寿命	★★★	★★★	★★★★★	★★★★★	★★★★★
特徴	最大4台のディスプレイ@4K	NUCはミニPCだが、SoCは組み込み対応	加速度、ジャイロスコープ、圧力、光センサー	車載用、EAVB、440ピン拡張	カメラオプション、2 x C66x DSP

★が最低ランクで、★★★★★が最高ランクです。「付録A」でテクノロジーの頭字語を参照してください。

* SoCには、評価ボードで公開されていない機能が含まれている場合があります。

SBC

製品は、カスタム回路基板なしで製造される場合があります。その場合は、以下に示すさまざまなシングルボードコンピューター（SBC）を利用できます。

	Broadcom	Intel	Nvidia	Rockchip	TI	Qualcomm
SBC	Raspberry Pi 4 Model B	Radxa Rock Pi X	Jetson TX2	Pine64 ROCKPro64	BeagleBoard X-15	Arduino Yún 2
SoC*		Intel Cherry Trail	Tegra X2	Rockchip RK3399 SOC	Sitara AM5728	QC Atheros AR9331
メモリー	8GB RAM、microSD	GB RAM、32GBフラッシュ、microSD	8GB RAM、32GB eMMC	LPDDR4（最大4GB）、eMMC、microSD	2GB RAM、4GBフラッシュ、microSD	64MB RAM、16GBフラッシュ、microSD
周辺機器	µHDMI、4K、MIPI、VideoCore VI GPU、WiFi ac、GigE、USB 3.0+2.0、BT 5.0	Intel Gen8 GPU、HDMI、オーディオ、WiFi ac、GigE、USB 3.0+2.0、BT 4.2	HDMI 2.0、4K、GP10B GPU、GigE、MIPI、6台のカメラ、USB 3.0+2.0	4K、MIPI、Mali GPU、オーディオ、GigE、USB 3.0+2.0、	HDMI、PowerVR GPU、オーディオ、GigE、SATA、USB 3.0+2.0	オーディオ、100Eth、WiFi b/g/n、USB 2.0
アーキテクチャ	ARM-32	x86-64	ARM-64	ARM-64	ARM-32	MIPS32
処理能力	A72 x 4コア@ 1.5GHz	Atom x 4コア@ 1.44GHz	Denver x 2コア@ 2.0GHz、A57 x 4コア@ 2.0 GHz	A72 x 2コア@ 2.0GHz、A53 x 4コア@ 2.0GHz	A15 x 2コア@ 1.5GHz、M4 x 2コア@ 212 MHz	24K @ 400MHz
ユースケース	中規模	大規模	大規模	中規模	中規模	小規模
消費電力	★★★	★	★★★	★★★	★★★	★★★★★
寿命	★★★★	★★	★★	★★★★	★★★	★★
特徴	拡張エコシステム	Windows互換	AIおよびコンピュータービジョン用の256 CUDAコア、ビデオエンコーダー/デコーダー	Wifi ac + BTおよびオプションのタッチパネルモジュール、big.LITTLE省電力アーキテクチャ	2 x TMS DSP、4 x 低レイテンシーI/O制御のためのPRU	オンボード ATmega32U4マイクロ

★が最低ランクで、★★★★★が最高ランクです。「付録A」でテクノロジーの頭字語を参照してください。

* SoCには、評価ボードで公開されていない機能が含まれている場合があります。

RAM

RAMは一般に、組み込みボードのコンポーネント中でCPUに次いで2番目に高価なコンポーネントです。組み込みデバイスに十分なランダムアクセスメモリー（RAM）が搭載されていない場合は、アプリケーションが最終的に利用可能なメモリーに適合しなくなるリスクがあります。このように重大な状態を早期に検出するには、ターゲットハードウェアシステムでメモリー消費量が最も大きい各コンポーネント(OS、グラフィカルUI、データベースなど）を実行します。さらに、実際の実行環境にできる限り近いテスト環境でも個別に実行します。これらのコンポーネントの最大メモリー消費量を注意深く追跡し、それらの消費量全部とマージンを足し合わせます。

アプリケーションのライフサイクル中のメモリー使用状況を理解し、注意が必要なメモリー消費量が非常に大きいコンポーネントを特定するには、優れたツールが必要です。使用パターンに一致する予備データがある場合は、それを参考にしてメモリーに影響を与えるアプリケーション構成を、最終的に本番環境向けに調整が必要になるように設定することもできます。これには、グラフィックス表示バッファー、オーディオバッファー、ディスクキャッシュなどのために確保されているメモリーが含まれる場合があります。

メモリーツール

他者のソフトウェアだけではありません。自分のアプリをテストし、アプリがメモリーの問題を引き起こさないことを忘れずに確認してください。Cppcheck、heob、GammaRay、valgrindなどのツールは、コードの静的および実行時分析を実行し、過剰なメモリー割り当てまたはメモリーリークを見つけるのに非常に役立ちます。

Flash

ほとんどのボードでは、microSDまたは別の外部フラッシュインターフェースを使用して、エンジニアリングスタッフが十分なスペアディスク領域を使って開発しつつも、開発終了時の必要な容量確保のために最適化を行えるようにします。一般にフラッシュの大きな課題はその速度です。ディスクの制約を受けるアプリケーションは、かなり高速なフラッシュを使用していない限りその点が課題になります。

セキュアデジタル（SD）カードの専門用語では（スタンダード、ミニ、マイクロを問わず）、クラスまたはビデオクラスの指定は書き込み可能なMB/秒を示します。つまり、クラス10カードは10MB/秒で、V90カードは90MB/秒で書き込むことができます。異なるクラスのmicroSDカードを複数用意しておくと役立つ場合があります。これにより、メモリーカードを交換することでフラッシュ速度の影響を簡単に測定し、コストと速度の最適なトレードオフを判断できます。

評価ボードにオンボードフラッシュが搭載されている場合は、そのフラッシュを最大限活用することをお勧めします。ほとんどの場合、フラッシュはSDカードよりも冗長性とパフォーマンスに優れているからです。

フラッシュのフォーマット

フラッシュファイルシステムでは、まったく異なる戦略とトレードオフを採用し、デバイスウェアレベリング、エラー検出/修正、クラッシュレジリエンスを管理できます。特にアプリが以下のいずれかの動作に依存している場合は、さまざまなファイルシステムでアプリをテストし、パフォーマンスが最も良いファイルシステムを確認してください。

- 高速起動
- 高速一括読み取り
- 高頻度の継続的書き込み
- 多数の小さなファイル
- 効率的なディスク圧縮
- 高速ディレクトリースキャン

MCU

現在のマイクロコントローラー（MCU）は非常に高性能でさらに高度になりつつあります。最大級の処理能力を必要としない場合は最適な選択肢になります。これらのチップには一般にメモリー管理ユニット（MMU）が搭載されていないため、アプリケーションベアメタルを実行するか、Azure RTOS ThreadXやAmazon FreeRTOSなどの非MMUチップをサポートできるOSを選択する必要があります。また、RAMとフラッシュの使用法にも特別な注意を払う必要があります。ここでは、さまざまな機能とアーキテクチャを備えたマイクロコントローラーを見てみます。

	Microchip	NXP	Renesas	ST	TI
評価ボード	PIC32MZ Embedded Graphics with Stacked DRAM Starter Kit	i.MX RT 1170 EVK	RH850/ D1M1A	32F769I-DISC0 discovery kit	C2000 Delfino MCU F28379D LaunchPad
MCU	PIC32MZ DA	i.MX RT 1176 DVMAA	RH850 G3M	STM32F7	TMS320-F28379D
メモリー	32MB SDRAM、4MBフラッシュ、microSD	64MB RAM、336MBフラッシュ、SDカード	4MB RAM、5MBフラッシュ、DDR2、ECCオプション	532KB RAM、2MBフラッシュ、microSD	204KB RAM、1MBフラッシュ
周辺機器	5.0インチWVGA、2D GPU、オーディオ、Eth 10/100、USB 2.0、CAN、6 x SPI、ADC	5.5インチ720pディスプレイ、OpenVG GPU、MIPI、オーディオ、USB、CAN、GigE、EAVB、SIM	2D GPU、スプライト、LVTTLビデオ入力、オーディオ、2 x I2C、6 x CSI、3 x CAN、Eth 10/100、EAVB	4インチタッチディスプレイ、2D GPU、MIPI、オーディオ、USB、4 x I2C、6 x SPI、3 x CAN、Eth 10/100	USB、ADC/DAC、PWM、CAN
アーキテクチャ	MIPS32	Cortex-M7 + M4	RH850	Cortex-M7	C28X
プロセッサー速度	252MHz	M7 @ 1GHz、M4 @ 400MHz	240MHZ	216MHz	200MHz x 2
温度定格	–40～125°C	–40～125°C	-40～+150 °C	-40～+105 °C	–40～125°C
消費電力	★★★★	★★★	★★★★	★★★★	★★★★
寿命	★★★★	★★★	★★★	★★★	★★★★
特徴	MEB拡張モジュールによる表示、暗号化オプションが利用可能、Raspberry Piヘッダー	Mag +加速度センサー、Arduinoヘッダー	車載用	オプションのWiFiモジュール、Arduinoヘッダー	2 x CLA FPU、三角関数および複素数計算ユニット、BoosterPackヘッダー、グラフィックスサポートなし

★が最低ランクで、★★★★★が最高ランクです。「付録A」でテクノロジーの頭字語を参照してください。

ユーザーI/O

ユーザーインターフェースと入出力(I/O)メカニズムは、製品差別化の最も大きな領域の1つですが、製品のコストと複雑さを増大させる要因にもなりえます。
以下の表に、デバイスとユーザーのインタラクション方法を計画する際に考慮すべき要因をまとめます。

	タッチスクリーン2D	タッチスクリーン3D	オーディオ	音声	物理コントロール	触覚コントロール	ジェスチャ
ハードウェア	画面、容量式プレート	画面、容量式プレート、グラフィックス処理ユニット(GPU)	スピーカー、デジタルアナログコンバーター(DAC)	マイク、アナログデジタルコンバーター(ADC)、スピーカー、DAC、音声アシスタントインターフェース	回転つまみ、スイッチ、ボタン、スライダー、発光ダイオード(LED)インジケーター	回転つまみ、スイッチ、モーター	カメラ、近接/圧力センサー、IRカメラ、ライダー
価格	$$$$$	$$$$$	$	$$$$	$	$$	$$$$
コミュニケーションの豊富さ	★★★★	★★★★★	★★	★★★★	★	★½	★
ユーザーがインターフェースの意図を理解する速度	★★★★★	★★★★★	★★	★★	★★★	★★★	★★
説明なしで使用可能	★★★★	★★★★	★★★	★★	★★★	★★★	★
トレーニングなしで使用可能	★★	★	★★★★★	★★	★★★★★	★★★★	★★
国際化の容易さ	★★★★★	★★★★★	★★	★★	★★★★	★★★★	★★★
多人数用途向けの衛生対策容易さ	★	★	★★★★★	★★★★★	★	★	★★★★★

★が最低ランクで、★★★★★が最高ランクです。「付録A」でテクノロジーの頭字語を参照してください。

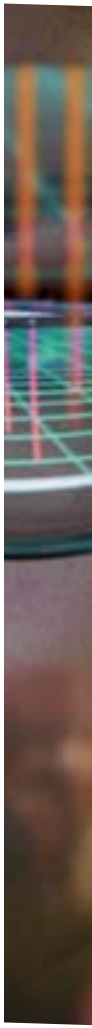
グラフィックス

選択したグラフィックス処理ユニット（GPU）は、ほとんどの場合、SoCに組み込まれます。そのため、グラフィックスパフォーマンスは、グラフィックスフレームワークとディスプレイパイプラインの品質に左右されます。もちろん、独自のハードウェアを設計する場合を除きます。独自のハードウェアを設計する場合でも、既存のOSとドライバーを適切にサポートできるCPUとGPUの組み合わせを慎重に選択する必要があります。独自のグラフィックスドライバーを作成することはまったくお勧めできません（技術的に不可能な場合もあります）。

GPUに対する要求はアプリケーションごとに異なり、現実世界のベンチマークは特定の指標が満たされることを保証するための唯一の手段です。とくに並列的な比較はありませんが、当社がお勧めするグラフィカルAPI標準はVulkan、DirectX、OpenGL、OpenGL ES（順不同）です。

グラフィカルAPI標準は、フレームレート、GPUおよびCPU負荷、バッテリー持続時間、過熱という点でパフォーマンスに大きな違いがある場合があります。レンダリング時に基盤としての標準を選択できるオプションがGUIフレームワークにあれば非常に効果的です。そうすることで、GUIコードをグラフィカル標準から独立させておくことができます。特定のアプリケーションの測定を行い、必要に応じてパフォーマンスを高めることができます。

3D GPUが最も一般的ですが、2D要素を含む標準UIを作成する場合は、2Dまたは2.5D GPUで必要な高速化がすべて得られる可能性があります。これらのGPUは、ビットブリットエンジン、レイヤリングとコンポジター、アンチエイリアシング、フォントグリフレンダリングの各機能を備えており、アプリケーションのレンダリング命令をユーザーに見えるピクセルに変換するプロセスを高速化できます。



ディスプレイ

製品を専用のディスプレイで際立たせることは可能でしょうか。その可能性は大いにあるでしょう。ここでは、顧客をあっと言わせるさまざまな方法をご紹介します。

	TFT LCD	PMOLED	AMOLED	QLED	E-paper
説明	シンフィルムトランジスター液晶ディスプレイ	パッシブマトリックス式有機発光ダイオード	アクティブマトリックス式有機発光ダイオード	量子発光ダイオード	電子ペーパー
明るい場所での可読性	★★★	★★★★	★★★★	★★★★	★★★★★
暗い場所での可読性	★★★★	★★★★★	★★★★★	★★★★★	★
更新速度	★★★★★	★★	★★★★★	★★★★★	★
色再現	★★★★	★★★	★★★★★	★★★★★	★
コントラスト	★★★	★★★★★	★★★★★	★★★★★	★★★★
視野角	★★★	★★★★	★★★	★★★★	★★★★★
電力消費量	★	★★	★★★★	★★★	★★★★★
耐久性	★★★★★	★★	★★★	★★★★	★★★★★
サイズ	1インチ～100インチ	0.5インチ～6インチ	1インチ～18インチ	TVサイズ	1インチ～10インチ
価格	$$	$$$	$$$$$	?	$
注記	現在の支配的なディスプレイテクノロジー。ほとんどの条件でバックライトを使用して視聴可能（太陽光の下で非常に明るいバックライトを使用可能）。	製造コストはAMOLEDより低いものの、消費電力が大きく、更新が低速。高電流のため、劣化が早い。	色域と可読性に優れているが、RGBコンポーネントが不均等に劣化し、焼き付きが起こりやすい。	AMOLEDの利点を活用しながらも、色の不安定さを排除。組み込みアプリケーションではまだ利用できないが、Samsungによって計画されている。	通常はモノクロだが、いくつかのカラーバリアントが使用可能。電力は画像変更時にのみ消費される。

★が最低ランクで、★★★★★が最高ランクです。「付録A」でテクノロジーの頭字語を参照してください。

まとめ

このガイドがお客様の組み込みデバイスまたはIoTデバイスの計画プロセスでお役に立つことを願っています。本書ではほとんどの基本事項を網羅するよう配慮しましたが、新しい製品は常に登場しており、可能な選択のすべてを1つのガイドで網羅することはできません。The Qt Companyのコンサルタントとエンジニアは、自動車、航空電子機器、医療用電子機器、産業用電子機器、家電、モバイルアプリ、ゲームなど、様々な業界の企業をサポートした実績があります。貴社の次のプロジェクトで取り組むべき内容についてサポートを必要となさる場合は、The Qt Companyが喜んでお手伝いいたします。

Qtエクスペリエンス

これはコンポーネントの選択を問わずに役立つ公平なガイドを意図していますが、Qtソフトウェアポートフォリオの開発には、当社が企業の製品計画を支援してきた経験から学んだ多くの教訓が組み込まれています。詳細については、以下のリンクをクリックしてください。

Qtをダウンロードする | **今すぐはじめる** | **製品概要** | **リソースセンター** | **ブログ**

付録A – 頭字語と略語

2D – 2次元
3D – 3次元
4K – 4K解像度
ABI – アプリケーションバイナリーインターフェース
ADC – アナログデジタルコンバーター
AI – 人工知能
AMOLED – アクティブマトリックス式有機発光ダイオード
API – アプリケーションプログラミングインターフェース
APR – Apacheポータブルランタイム
BT - Bluetooth
CAN – コントローラーエリアネットワーク
CI – 継続的インテグレーション
CLA – 桁上げ先見加算器
CPU – 中央処理装置
CSI – カメラシリアルインターフェース
CSS – カスケーディングスタイルシート
CUDA – コンピュートユニファイドデバイスアーキテクチャ(Nvidia)
DAC – デジタルアナログコンバーター
DDR – ダブルデータレート
DDR2 – ダブルデータレート2
DDR3L – ダブルデータレート3低電圧
DDR4 – ダブルデータレート4
DIMM – デュアルインラインメモリーモジュール
DP++ – ディスプレイポートデュアルモード
DSP – デジタル信号プロセッサー
E-paper – 電子ペーパー
EAVB – イーサネットオーディオビジュアルブリッジング
ECC – エラー修正コード
eMMC – 組み込み型マルチメディアカード
Eth – イーサネット
Eval – 評価
FFT - 高速フーリエ変換
FPU – 浮動小数点演算処理装置
GB – ギガバイト(10億バイト)
GDP – 国内総生産
GHz – ギガヘルツ
GigE – ギガバイトイーサネット
GPU – グラフィックスプロセッシングユニット
GUI – グラフィカルユーザーインターフェース
HDMI – 高解像度マルチメディアインターフェース
HTML5 – ハイパーテキストマークアップ言語5
IDE – 統合開発環境
LCD – 液晶ディスプレイ
LED – 発光ダイオード
LPDDR4 – 低電力ダブルデータレート4
LVDS – 低電圧差動信号
LVTTL – 低電圧トランジスター-トランジスターロジック
MB – メガバイト(100万バイト)
MCU – マイクロコントローラーユニット
MHz - メガヘルツ
Mic – マイクロフォン
microSD – マイクロセキュアデジタル
MiniPCIe – ミニペリフェラルコンポーネントインターコネクトエクスプレス
MIPI – モバイルインダストリープロセッサーインターフェース
MMU – メモリー管理ユニット
NUC – ネクストユニットオブコンピューティング(Intel)

OpenGL – オープングラフィックスライブラリ
OpenGL ES – オープングラフィックスライブラリ組み込みシステム
OS – オペレーティングシステム
OTA – 無線アップデート
PCIe – ペリフェラルコンポーネントインターコネクトエクスプレス
PMIC – 電源管理用集積回路
PMOLED – パッシブマトリックス式有機発光ダイオード
POSIX – ポータブルオペレーティングシステムインターフェース
PRU – プログラマブルリアルタイムユニット
PWM – パルス幅変調器
QA – 品質保証
QLED – 量子発光ダイオード
QML – Qtモデリング言語
RAM – ランダムアクセスメモリー
RGB – 赤緑青
RTOS – リアルタイムオペレーティングシステム
SATA – シリアルATアタッチメント
SBC – シングルボードコンピューター
SD – セキュアデジタル
SDRAM – 同期ダイナミックランダムアクセスメモリー
SoC – システムオンチップ
SPI – シリアルペリフェラルインターフェース
TFT – シンフィルムトランジスター
TI – テキサスインスツルメンツ
TMS – TMS320シリーズDSP（TI）
TV – テレビジョン
UI – ユーザーインターフェース
USB – ユニバーサルシリアルバス
UWP – ユニバーサルWindowsプラットフォーム（Microsoft）
UX – ユーザーエクスペリエンス
V90 – SDアソシエーションビデオスピードクラス90（90MB/秒）
WebGL – Webグラフィックスライブラリ
WiFi – IEEE 802.11ワイヤレスプロトコル
WiFi ac – IEEE 802.11ac-2013またはWiFi 5
WYSIWYG - 見たままを得られる
μHDMI – マイクロ高解像度マルチメディアインターフェース

THE QT COMPANYについて

The Qt Companyは、製品開発、商用およびオープンソースライセンシング、オープンガバナンスモデルの下でのQtプロジェクトなど、あらゆるQtアクティビティーを担っています。当社は、ライセンシング、サポートおよびサービス機能を提供するとともに、開発者と緊密に連携して、Qtプロジェクトがスケジュールどおりに予算内で競争優位性を維持しながらデプロイされるようにするという使命感のもとに活動しています。

The Qt Companyのゴールは、デスクトップ、組み込み、およびモバイルの開発者と企業に最も強力なクロスプラットフォームのUIおよびアプリケーションフレームワークを提供することです。The Qt Companyは、ライセンシング、サポートおよびサービス機能を提供するとともに、開発者と緊密に連携することを常に重視しています。

www.qt.io/ja-jp

The Qt Company

〒100-0005 東京都千代田区丸の内3-3-1新東京ビル4F
Web: https://www.qt.io/ja-jp/
Email: japan@qt.io | TEL: 03-6264-4500