# The Embedded GUI Playbook:

Developing Touchscreen for Consumer-Centric Products

A practical guide to creating high-volume, high-margin goods on embedded hardware with sophisticated GUIs.

## What to expect from this e-book

### Four critical consumer expectations that open wallets

Create successful products by taking these consumer preferences into account.

PART 1

### Two ways to meet the relentless pressure to deliver faster

Understand the importance of streamlining the development process and ways to do it.

PART 2

### Selecting silicon and software to differentiate your product

Capitalize on consumer preferences and overcome time-to-market pressures through component selection.

PART 3

### Two sure-fire ways to break into the product sweet spot

Adapt existing designs to create products with mass-market appeal and high-margin pricing.

PART 4

### How to determine if your product is a good candidate

Decide for yourself if these strategies make sense for your product.

PART 5

### Key takeaway

Learn what it takes to turn great ideas into polished, consumer-ready products.

PART 6

## INTRODUCTION

The face of consumer electronics is evolving fast. From smart thermostats to connected kitchen appliances, today's products are expected to deliver the same smooth, intuitive interfaces that users experience on their phones and tablets. Consumers no longer settle for clunky screens or confusing controls—they expect touch-friendly, visually polished interactions everywhere.

But for product teams and GUI developers, meeting these expectations isn't easy. The challenge? Delivering sleek, responsive interfaces on cost-sensitive hardware without sacrificing performance or extending timelines.

This ebook explores the next wave of consumer products, the hurdles developers face—like optimizing for resource-constrained devices, integrating with modern design workflows, and accelerating time to market—and practical ways to build GUIs that feel high-end while staying efficient and affordable.

Whether you're designing your first connected product or scaling up your product line, you'll find strategies here to help you turn tough constraints into competitive advantages.

**PART 1**

## Four critical consumer expectations that open wallets

UX leaders Apple and Google have forever changed consumer expectations. Products must now be simple and user-friendly regardless of device or platform. While these two companies have loyal followers, in general people are beginning to care less about the brand and more about the experience. Data from Forrester shows that improving customer experience improves profitability; in fact, the revenue growth of customer experience leaders is 5.1 times that of laggards. A case in point is Uber: the company has been through one scandal after another yet they remain successful because people love the overall experience. Let's examine four of today's biggest consumer expectations and develop some guidelines to follow in order to make your next consumer electronics product wildly successful.

### 1 Create products with both functionality and design in mind

In the good old days when function trumped form, consumers had to live with difficult-to-use products like the much-maligned VHS recorder. The reason TiVo killed the VHS was it was well-designed and dead simple to use. Today, consumer preference for a product is increasingly determined by the quality of the user experience. Designers and developers who work closely together to create products that are at once attractive, functional, and easy to use are behind some of the world's most successful products.

Apple was not the first MP3 player to hit the market but, with its sleek design and intuitive interface, it quickly dominated all other rivals.

In order to strike that perfect balance between form and function, you need to have designers engaged from the very beginning of the product development process. Since critical design choices may significantly morph your product, you shouldn't wait until the product requirements are done before you start working with the design team or you risk wasting time or building suboptimal products. We also recommend facilitating a smooth and efficient workflow between designers and development teams with both people-focused activities like team building and technological solutions like designer-friendly tools.

*Quite simply, design can no longer be an afterthought.*

## 2 Include UI screens that are attractive and intuitive

Now that consumers have continual access to an amazing UI in their pockets, they're less tolerant of lukewarm experiences. They may not always be able to articulate their preferences but know how to vote with their wallets.

A good example is the growing popularity of the single-serve coffee machine. Price comparison shows a unit of K-cups (a single-serve of coffee grounds in a small filter) commands nearly five times more than a unit of traditional ground coffee.

One of the reasons these coffee machines rank in the fastest-growing private-label food and beverage category despite the hefty price tag is their full-color LCD. Consumers see these displays as a valuable feature and are increasingly making purchasing decisions based on their availability.

Millennials were among the first to embrace digital technology. They grew up with screens and were key drivers behind the rise of smartphones, apps, and connected devices. But now, things are changing, according to a recent Social Values research, they're second only to Gen Z in their enthusiasm for new tech.

As technology keeps evolving, many Millennials feel overwhelmed with- issues like digital burnout, data privacy, and online security. While they still value innovation, they're becoming more cautious and slower to adopt new tech products.

Consumers see full-color displays as a valuable feature and are increasingly making purchasing decisions based on their availability.

In an ever-digitized world, brands and companies need to recognize this shift. It's important for them to balance exciting new features with thoughtful design that respects users' time and privacy. To build trust and long-term relationships with Millennial consumers, businesses must understand both the benefits and the downsides of connectivity.

Consumers love the tactile experience of interacting with sophisticated touch-screen displays.

### Reasons consumers are attracted to products with color LCDs

- **Modernity**: Full-color dot-addressable screens are fresh, while custom-designed LCDs are reminiscent of old technology like clock radios
- **Capacity**: The flexibility of a product with an intuitive full-color screen makes the product appear to do more than its fixed-function LCD counterpart
- **Simplicity**: Features can be revealed as needed (including usage guides or servicing instructions) while products that have many buttons or LEDs can confuse a user by presenting all options at once
- **Elegance**: End users have increasingly sophisticated design ascetics, making them prefer products with appealing, attractive, and artful screens

## 3 Plan for multi-modal input

Of course, there are other ways to interact with a product than through a screen. companions in our daily lives. Thanks to big leaps in AI and language understanding, in near future these assistants can handle complex tasks, have natural conversations, and even predict what you might need based on your habits. For example, Google's Gemini Live, offers capabilities such as real-time voice translation and sophisticated question answering.

Moreover, the integration of voice assistants with Internet of Things (IoT) devices is transforming homes. Voice assistants like Siri and Apple intelligence lets you control lights, locks, and appliances just by speaking. In Automotive segment Volkswagen's integration with ChatGPT through its IDA voice assistant allows drivers to interact with their vehicles using natural language, making the driving experience more enjoyable.

The future of voice assistant has immense potential with Integration with VR, MR, AR technologies, and AI algorithms. As technology continues to evolve, voice assistants are becoming part of our day to day lives—making it easier, more efficient, and more connected.

**Use a tool that can bridge to C/C++ to maximize your ability to expand in the future.**

We recommend designing with multi-modal input in mind—even if your product doesn't immediately require voice or device connectivity, it should be able to accommodate it. This means designing your product for expansion, exposing APIs for remote functionality, and planning to provide over-the-air updates.

## 4 Plan for scalability

If the best products provide an excellent user experience, the best selling products also provide excellent value for the money. With so many forces attempting to get a share of the consumer wallet, you need to build products that are "worth it", and that usually means a bill of materials that's as low as you can manage.

While it may be accepted practice to create an initial line of products with high prices (and high margins) to test out market acceptability and build up cash reserves, this is really a short term strategy. Tesla is a perfect example of this approach: the company started with a pricey Model S before releasing the Model 3 for middle-class buyers. Companies that don't make the transition to commodity pricing are often stuck creating a niche product for a limited market instead of breaking into the mass market.

**Use a tool set that works with microprocessors and microcontrollers to target products at all price points.**

Build your niche-market product with scalability in mind so you can move from a high-end microprocessor to a lower-priced microcontroller with ease.

The solution is to plan for scalability. From the electronics point-of-view, you should choose a microprocessor line for your initial product that's part of a product family with lower price-point cousins. Use a tool set that isn't inherently tied to beefier processors and that can scale into smaller—yet still capable—microcontrollers. Finally, plan your RAM and Flash consumption carefully so that you're not trapped into an architecture that is resource heavy and can't be scaled down. These considerations will help ensure you're building a product that could run on a lightweight microcontroller even if your initial design hasn't yet taken the necessary steps.

PART 2

## Two ways to meet the relentless pressure to deliver faster

Everyone knows that the innovation treadmill is leading to ever-shortening development cycles and increased competition. The big question is: how can manufacturers compete under that relentless pressure to deliver faster and faster? Just as critical is maintaining quality at this break-neck speed.

Since the software development timeline is the single biggest factor that inhibits speed to market, let's take a closer look at ways to reduce it.

### 1 Streamline design and development

As discussed, design is a key part of today's product success. But the normal design-developer cycle is often inefficient.

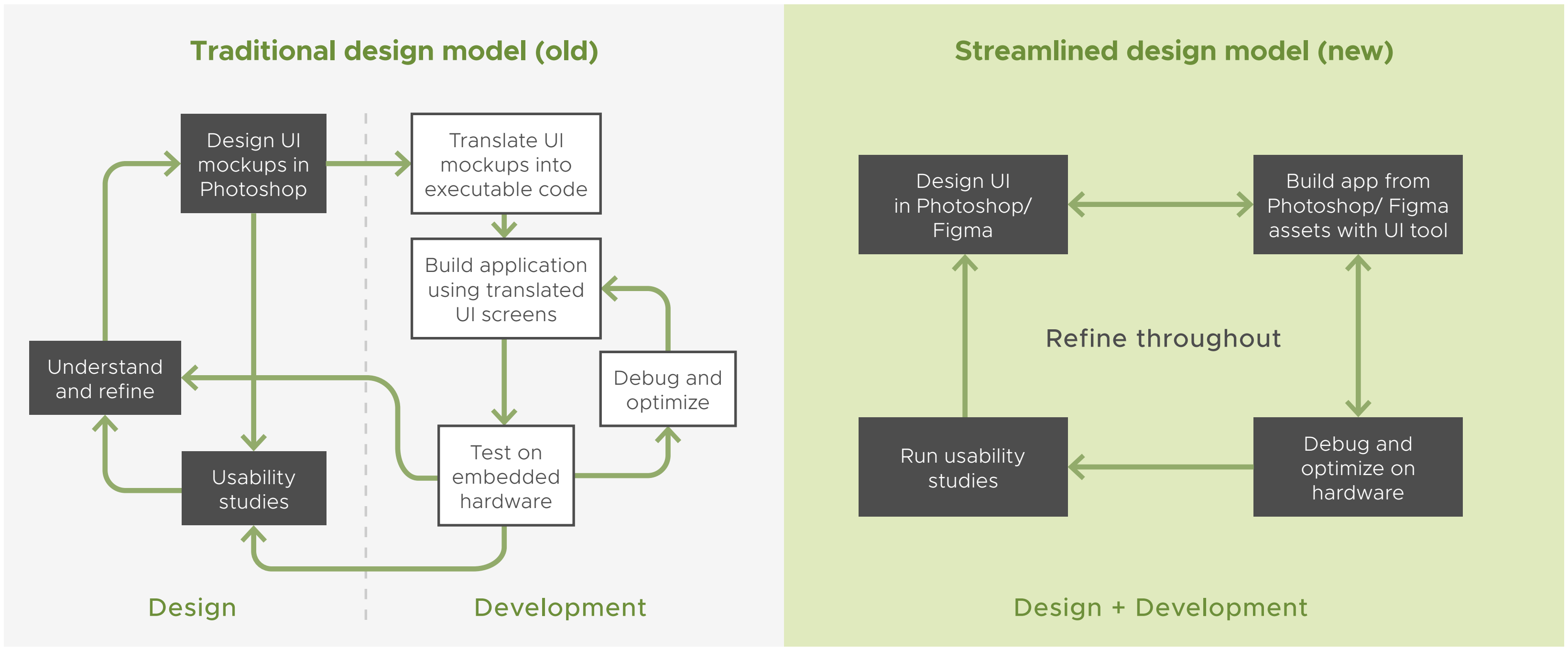
Designers often use one set of tools (such as Adobe Photoshop) to craft the look and feel of a UI and another set for prototyping (like Adobe Experience Design). Developers then use a third toolset (often C/C++) to implement the final product. All of these changes in tools, process, and workflow suck up time and introduce errors. Thankfully, there are tools on the market that can help create a streamlined design-development process.

The best development tools are ones that are simple enough for the designer to create a prototype UI but deep enough for the developer to produce complex logic, include pre-existing libraries, and modularize components.

As always, your mileage may vary—depending on the skills and breadth of your team—but keep in mind that your team's current skill level in various tools often changes when principle members move on or you need to staff up.

If your team doesn't already use a tool to streamline the design-development process, you'll want to seriously consider one.

**Use a tool that streamlines the design-development workflow to minimize wasted effort.**


Traditional design model (old)
Design UI mockups in Photoshop
Translate UI mockups into executable code
Build application using translated UI screens
Understand and refine
Debug and optimize
Usability studies
Test on embedded hardware
Design
Development
Streamlined design model (new)
Design UI in Photoshop/ Figma
Build app from Photoshop/ Figma assets with UI tool
Refine throughout
Run usability studies
Debug and optimize on hardware
Design + Development

## 2 Architect code for reuse

If you're building a one-off product, code reuse doesn't matter. That being said, you almost never use good code only once. Initial releases invariably get updated with new features and bug fixes, one product can lead to a family of products, and reliable code from older projects is often implemented into new incarnations. So if you're writing code with a plan to throw it away, you're doing it wrong.

A rough guideline is that writing reusable code takes at least three times longer than writing single-purpose code. Not everything however needs to be built for reuse or you may end up over-engineering your code and adding unneeded complexity. If code is written too generically but only used once, it's a waste of development time and code overhead.

**Ensure your UI tool can cleanly separate UI logic from business logic in order to leverage visual elements and themes across projects.**

But somewhat paradoxically, code reuse is very important to rapidly producing reliable products, even with the extra time required to make code reusable. You can always get one product to market quickly. With reusable code you can get subsequent products to market even quicker.

Experienced developers constantly make subconscious trade-offs about the architecture of the software they're writing—which parts are likely reusable and deserve extra attention. It pays to discuss, and incorporate reuse guidelines into your team workflow.

Finally, use a methodology that separates the UI logic from the business logic. Doing this will allow common visual elements and themes to be leveraged across products and platforms, making it easier for you to build derivative products. Ensure your UI tool of choice makes this simple. Your development staff shouldn't constantly fight against the tool to implement a clean MVC (model view controller) paradigm.

Allowing designers and developers to use the same tool eliminates disconnects and rework.

### Top things to consider when choosing a design-friendly UI tool

- Scripting for simple understanding by designers and speed of creation by developers
- Ability to invoke C/C++ code and libraries from within the UI environment
- Flat tool hierarchy with easy learning curve
- Easy integration with existing design tools
- Ability to manage round-tripping design assets
- Comprehensive Component Library (ready-to-deploy) for developers
- Reusable Component Library to implement design changes faster
- Effortless UI updates with static data control

PART 3

## Selecting silicon and software to differentiate your product

If your component evaluation process is like the majority, it can be overwhelming with all the comparative options, and rarely is there a single right choice. Rather than building a comparative chart, you should be laser focused on the factors that will allow you to capitalize on consumer preferences and overcome competitive pressures. Let's start with a look at hardware.

### Hardware

While full-featured microprocessors have traditionally been the solution for complex designs, microcontrollers have been steadily gaining power and capability and have spawned a new category of crossover microcontrollers (see sidebar).

These crossovers are excellent choices for many consumer electronics products, because they provide a high level of sophistication at a much lower price point.

In fact, they sport many of the same features. Yet they have few of the frills that are standard fare on application processors like memory management units, multiple cores, on-board GPUs, floating point units, and so on. But make no mistake, they can still pack a lot of power into a small chip—enough to handle all but the most demanding designs.

### Crossover Microcontrollers

MCUs are increasingly pushing into low-end MPU territory, offering more performance, memory, and connectivity—while still retaining their low-power efficiency. With higher clock speeds, multi-core designs, and built-in wireless options, modern MCUs can now handle tasks like real-time connectivity, advanced peripherals, and even lightweight machine learning. This evolution lets developers deliver smarter, more responsive products without sacrificing battery life or driving up costs, making MCUs the go-to choice for today's connected consumer devices.

The really good news is the price of a crossover microcontroller is usually significantly less than a low-end application processor. With RAM and Flash on board the microcontroller, crossovers are often able to remove the cost of additional chips from a design. Depending on which RAM and Flash chips are eliminated, a crossover micro can reduce cost from $4 to $40 (sometimes more), which is significant even on the lower end.

This substantial price advantage is why we're looking at crossover microcontrollers in this e-book.

## How do I pick the right hardware?

We've taken the consumer and competitive factors, translated them into the most important hardware features, and created the chart at right to help you decide which hardware is best for your project. This list isn't exhaustive but should get you thinking. For instance: if you want your product to appeal to millennials and intuitively communicate the functionality it houses, your hardware better have fantastic graphics support.

If you want to create a product with mass-market appeal, keep on the lookout for chips that minimize your BOM cost—such as ways to eliminate external SDRAM.

We suggest using this chart to review the features of each microcontroller. We've started you off by doing this for a few of today's popular ones in the table on the following page.

**Prioritizing hardware selection criteria based on product requirements**

| To support these features... | Look at these criteria... |
|---|---|
| Functionality and design | Graphics support |
| Attractive screens and UI | Graphics support |
| | CPU horsepower |
| Multi-modal inputs | CPU horsepower |
| | Wide array of interfaces |
| Excellent price point | BOM cost |
| | Size of on-board memory |
| Streamlined design and development | Standardized instruction set |
| Reusable software | Standardized instruction set |
| | Scalable product family |

## Microcontroller comparison

| | Processor attribute[1] | NXP iMXRT595 | ST Microelectronics STM32H7XX | Renesas Electronics RA8D1 | Espressif ESP32-S3 |
|---|---|---|---|---|---|
| Graphics support | Acceleration | Cadence Fusion F1 DSP, Vivante GCNanoLite-V GPU | DSP instructions, double-precision FPU | Integrated Graphics LCD Controller, DRW | Single-precision FPU |
| | Screen | 1024 × 480 | 1024×768 | 1280×768 | Up to 800×600[2] |
| CPU horsepower | CPU frequency | 300MHz | Cortex M7 up to 600 MHz | 480 MHz | 240 MHz |
| | CoreMark | 3020 | 3224 | 3000 | 1181 |
| Wide array of interfaces | Peripherals | USB, I2C, UART, SPI, CAN, SDIO, FlexIO, MIPI-DSI, I3C | USB (FS/HS), Ethernet, CAN FD, I2C, UART, SPI | Ethernet MAC, USB 2.0 High-Speed, CAN FD, SDHI, I3C | USB OTG, SPI, I2C, UART, I2S, CAN |
| | Audio | I2S, dual DMICs, stereo codec, audio PLL | I2S, SAI, DFSDM | 12-bit DACs | I2S, PDM, support for external audio codecs |
| Size of onboard memory | RAM | 5MB SRAM, 6 GB eMMC module | 1MB SRAM | 1 MB SRAM with ECC | 512 KB SRAM + 16 KB RTC SRAM |
| | Flash | 64 MB | 2MB | 2 MB | 16 MB |
| Standardized instruction set | Core architecture | ARM Cortex M33 + Cadence Fusion F1 DSP | Dual-core configurations with Cortex®-M4 at 240 MHz | Arm Cortex-M85 with Helium and TrustZone | 32-bit Xtensa LX7 dual-core |
| Scalable product family | Substitutable chips | Scales up to i.MX RT700 series and scales down to i.MX RT1060 | Scales up to STM32MP1 | RA6M3, RA6M5, RA8T1 | Scales up to ESP32-C6 (Wi-Fi 6 + RISC-V) |

[1] Processors being compared are not intended to be an exhaustive list of all compatible options
[2] Supports external displays via LCD interface (up to 800×600)

Compare hardware

## Software

What about UI  tools, the other critical ingredient for building a beautifully designed product? Many companies still build their UI code in-house. With all the excellent options out there that are extremely capable, already written, and pre-debugged, this is a complete waste of energy. You're not going to differentiate your product based on custom UI tools—you're going to waste a lot of time reinventing the wheel. Save that energy for building things that will make your product stand out and leverage someone else's hard work.

That's not to say that UI choice is any easy one; rather it's a fundamental one. Your UI tool can make a development job easy or hard, shape the look and feel of your product's screen, determine how easily designers can be involved, dictate minimal hardware requirements, set the development language and toolchain, and either constrain or enable the inclusion of additional features.

## How do I pick the right software?

If multi-modal inputs are critical, you're probably going to need to be calling code in C libraries. What if getting to market in record speed is at the top of your list? Make sure your tool breaks down the barriers between designers and developers.

Use the chart at right to guide you when evaluating software tools. We've started the process for a selection of UI tooling products as shown in the table on the following page.

**Prioritizing software tool criteria based on product requirements**

| To support these features... | Look at these criteria... |
|---|---|
| Functionality and design | Designer-friendly |
| Attractive screens and UI | Designer-friendly |
| | CPU horsepower |
| Multi-modal inputs | External code bridge |
| | HTML support |
| Excellent price point | Microcontroller support |
| | Runtime size |
| Streamlined design and development | Designer-friendly |
| | Easy learning curve |
| | Powerful scripting |
| Reusable software | Easy learning curve |
| | Powerful scripting |

Choosing the right GUI development software is about selecting a solution that accelerates your workflow, supports real-time collaboration between designers and developers, and helps you deliver a high-performing product on schedule.

**Below are four key features to prioritize when evaluating your options:**

### 1. Fast Prototyping with Widget and Component Libraries

Having a robust widget or component library is a productivity accelerator. The ability to drag & drop pre-built UI elements like buttons, sliders, menus, and containers, teams can quickly assemble a working prototype.

### 2. Seamless Integration with Design Tools

The handoff between design and development is often where projects slow down. The ability to import layouts, assets, fonts, and styling directly from the design tools like Figma, Sketch, Photoshop etc, into the development workspace helps preserve the original design intent while saving hours of manual rework.

### 3. Support for Round-Tripping Design Files

Project needs evolve, feedback rolls in, and UI elements often require last-minute tweaks. The right GUI tool will support round-tripping—the ability to update design files and re-import them into the development environment without breaking the build.

### 4. Built-in Optimization Tools for Embedded Performance

As important as aesthetics and usability are, your GUI also needs to run efficiently on the target hardware. Whether you're working with a high-performance MPU or a resource-constrained MCU, the best GUI development tools include optimization features that identify unused assets, heavy rendering paths, or unnecessary complexity.

PART

## Two sure-fire ways to break into the product sweet spot

If you're starting a product from scratch, you have the freedom of creating something that accommodates this advice from the start. But what if you've already got products that don't have attention-getting interfaces with mass-market prices? Thankfully, we've thought of that. We've got two ways for you to consider adapting your existing designs.

### 1 Downward migration

It often makes sense to introduce new features—especially screens—into high-end products first. With larger margins, premium products are better able to absorb the cost of extra hardware and early adopters are usually willing to spend a little more coin to get the latest tech. However, as these features catch on with the general public, it makes sense to migrate them into mass-market products.

Automotive instrument clusters are a prime example of this downward feature migration. Replacing physical gauges with digital screens started in premium level cars but has now trickled down to mid- and entry-level vehicles. Moving screens from high-end products into lower-end ones may be a successful strategy for you as well. If you've already built a great UI on your top-end device, bringing it into a mass-market product is a sure-fire way to differentiate it from the screen-less competition.

To succeed at this downward migration however, requires dramatically reducing costs and simplifying features. Thankfully the transition to a microcontroller provides enough cost savings to allow you to move premium features into mid-level products, even though that journey often requires simplification to meet the tighter constraints of the smaller

A premium code base can support a smaller screen with fewer pixels by using lower resolution images, excluding textures, and removing low-priority elements.

hardware. The process of extending your UI downwards is dramatically simplified if you plan in advance to build your UI for reuse and testability, and make UI reuse and testing part of your completion criteria.

The UI feature simplification required to reach lower-level screens has an added benefit. Besides smaller bill of materials costs, it also better segments your product family so that the newly enabled mid-market devices won't accidentally cannibalize your upper-end products.

## 2 Upward migration

The other direction for product migration is upwards, by adding screens to products that are already mass-market to help them attract more customers and increase their market penetration. The nearly ubiquitous smart device is a case in point. While not every consumer product has a smart variant, anybody who's been to CES or recently visited an appliance store can tell you which way the wind is blowing—screens are quickly becoming ubiquitous.

Adding a screen and smarts to an existing product lets you get more dollars and longevity out of an existing product line.

While screen-less products clearly get the job done, products with intuitive interfaces offer busy consumers additional convenience and efficiency. Take the white goods market as an example. Consumers can now set up laundry cycles so that they run in off-peak hours. They can also receive alerts from their fridge instead of wondering which condiments are running low. You need a screen to make features like this shine; it's nearly impossible to imagine these advancements implemented through old-fashioned dials and buzzers.

The public's openness to smart screens in all manner of devices is thanks (again) to smartphones and apps, the continual fast pace of technology breakthroughs, and the increasingly larger segment of society that embraces technological advances. An additional consideration in the white goods market is that governments and regulatory bodies introducing energy reduction incentives often coincidentally drive sales of the more energy-efficient smart appliances.

While these reasons for scaling your mass-market products upwards might be sufficient on their own, another major factor is the bottom line. By shifting focus from the lower-margin segment of the consumer electronics market towards the higher-margin and higher-growth segment, you stand to improve revenue and shareholder value.

Full color screens allow you to target the higher-growth and higher-margin segments of your market.

PART 5

## How to determine if your product is a good candidate

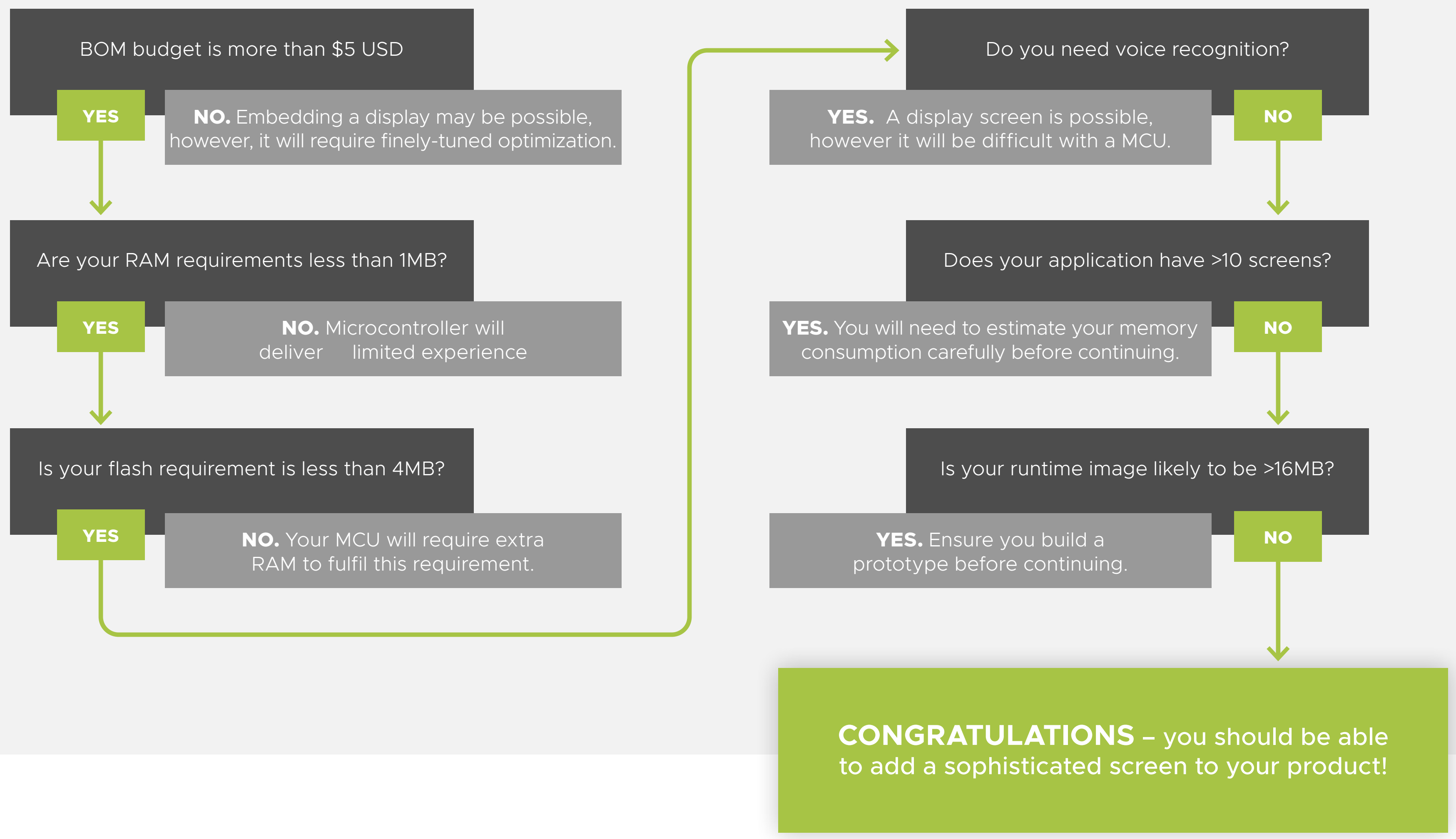
You're now convinced that adding a small screen and amazing UI to an existing product is a great idea. The next step is deciding if this approach makes sense for your product regardless of whether it's already on the shelves or just on the drawing board.

The simple checklist on the following page will help you quickly eliminate poor candidates for adding microcontroller-driven small screens to a product, potentially saving you some time for an evaluation that likely wouldn't pan out.

Even if you pass the checklist, you need to create a prototype to test out the concept. A recommended approach is to consult with your chosen UI framework company. They'll be able to give you their experience-based opinion if your product is feasible and if their platform is appropriate. Many also offer consulting services that can help you prototype, migrate, train, and develop your product.

Don't forget all parts of your software stack when calculating your RAM and Flash budget.

## Should you incorporate a display screen on your device?



BOM budget is more than $5 USD

**YES**

**NO.** Embedding a display may be possible, however, it will require finely-tuned optimization.

Are your RAM requirements less than 1MB?

**YES**

**NO.** Microcontroller will deliver limited experience

Is your flash requirement is less than 4MB?

**YES**

**NO.** Your MCU will require extra RAM to fulfil this requirement.

Do you need voice recognition?

**YES.** A display screen is possible, however it will be difficult with a MCU.

**NO**

Does your application have >10 screens?

**YES.** You will need to estimate your memory consumption carefully before continuing.

**NO**

Is your runtime image likely to be >16MB?

**YES.** Ensure you build a prototype before continuing.

**NO**

**CONGRATULATIONS** – you should be able to add a sophisticated screen to your product!

PART 6

# Key takeaway

Storyboard is a next-gen UI design & development tool, that can accelerate product delivery and enhance user experience. It empowers developers to create rich, scalable interfaces with minimal overhead.

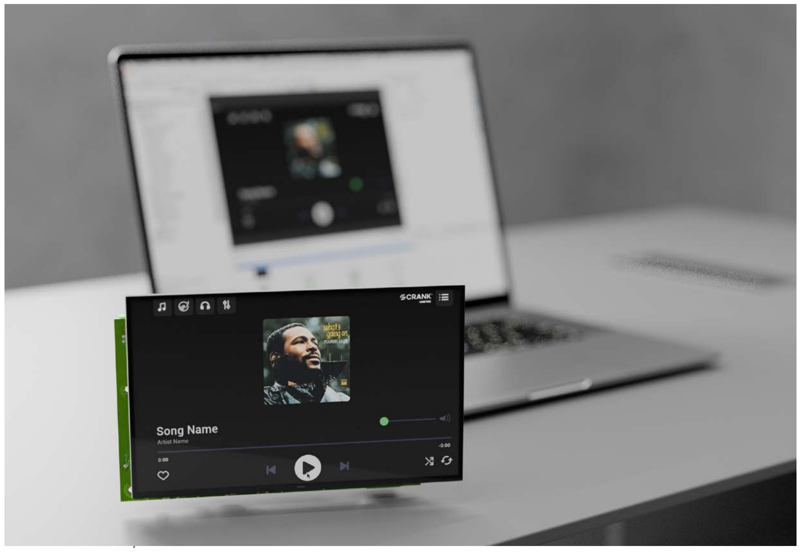
One can create visually appealing yet intuitive UI screens through a drag-and-drop interface and reusable component design libraries. It supports custom styling, animations, and responsive layouts. One can design in their favourite design tool like- Photoshop, Sketch, Illustrator or Figma and import it directly helping ensure that products not only function effectively but also provide delightful user experiences.

It bridges the gap between design and development by offering a unified environment where UI/UX designers and developers collaborate seamlessly. Features like real-time preview not only minimize miscommunication but also reduce iteration cycles, keeping development timelines tight and efficient.

Built with scalability in mind, Storyboard supports direct integration with C/C++ codebases, enabling the reuse of existing business logic and system-level libraries.

It also supports a modular design approach and a scalable UI architecture making it easy to scale from simple interfaces on low-cost MCUs to more complex applications on high-performance MPUs.

It has Reuse of UI components across multiple projects and with component-based UI building, real-time collaboration tools, and compatibility with both design and development workflows Storyboard reduces development time.



A supplier with experience in creating rich UIs on a wide range of silicon can help you diversify your portfolio.

pleasure for developers to work with. Developers can focus on logic without getting bogged down by UI intricacies, while still maintaining full control over performance-critical aspects of the product.

Storyboard features empowering developers to develop faster, meet demanding UX expectations on tight hardware budgets, and reduce costly iteration cycles.

- Storyboard allows embedded render extensions and components at any level of the layout tree, reducing rework during design updates or late-stage changes.
- Build responsive UIs that adapt to different screen sizes using relative sizing and anchor points. development time and reducing hardware-specific customization.
- Dynamic UI Creation with features like direct action triggering, variable listeners, and lightweight attribute updates via Lua
- Import interactive and data-driven elements directly from Figma and other design tools including variables, transitions, and strokes as lightweight fills.
- A suite of widget-style components making it easier to assemble interfaces quickly and consistently.
- Inbuilt testing features improve the accuracy of UI test automation by ensuring that test execution waits for screen transitions to complete before continuing.

Try Storyboard today

**"Storyboard made something possible that none of our competitors can come even close to reproducing."**

— Hank Bezuidenhout, Embedded IQ

**Crank Software**
1000 Innovation Drive, Kanata, ON, Canada K2K 3E7
+1 (613) 595-1999 | info@cranksoftware.com | cranksoftware.com