The Qt Company

임베디드 제품 개발에 필요한 요구사항과 계획 수립을 위한 가이드

Embedded product planning and requirements guide

개요

임베디드 제품을 개발하기에 더할 나위 없이 좋은 시기입니다. 풍부한 오픈소스 소프트웨어 스택, 커뮤니티 리소스, 실리콘 부품, 전자 부속품 덕분에 세련되고 풍부한 기능을 갖춘 제품을 단시간에 개발할 수 있게 되었습니다.

그러나 역설적이게도 명확하고 확고한 선택 기준이 없다면 이러한 많은 선택지가 오히려 의사결정에 방해될 수 있고, 제품 개발이 소모적인 연구개발 프로젝트로 끝날 수 있습니다. 설상가상으로 선택한 솔루션이 시장에서 사장될 수도 있습니다. 개발하던 제품의 출시를 강행하면 금세 사용자에게 외면받을 것이며, 출시 지연을 결정할 경우 전면 재작업에 들어가야 합니다.

정보의 과부하 속에서 성공적인 사물 인터넷(IoT, Internet of Things)과 임베디드 프로젝트를 위해 고려해야 할 가장 중요한 기준들과, 각 항목별로 가장 널리 사용되는 기술들을 정리했습니다. 그리고 각 기술 간 쉬운 비교를 위해 카테고리별로 등급을 매겼습니다.

본 가이드가 여러분이 제품 개발 프로젝트에서 노력과 리스크를 줄이는 데 도움이 되기를 바랍니다.

목차

적용 방법론

새로운 임베디드 및 IoT 디바이스를 개발할 때 어떻게 접근해야 할까요? 전통적인 방법은 비교적 명확합니다. 요구사항을 정의한 후 하드웨어 보드를 설계하고 이를 실행할 소프트웨어를 개발하는 것입니다.

그러나 오늘날 이러한 방식은 더 이상 통하지 않습니다. 시장은 빠르게 변하고 경쟁은 치열하며 고객의 기대치는 높습니다. 이러한 압박은 두 가지 직접적인 변화를 불러오고 있습니다. 첫째는 제품 디자인을 결정하는 것은 이제 하드웨어가 아닌 소프트웨어라는 것입니다. 이는 소프트웨어 개발에 상당한 시간과 인력이 필요하고 제품 사용 전반에 대한 성공적인 고객 경험이 소프트웨어에 의해 좌우되기 때문입니다. 다른 하나는 그 어느 때보다 기업에게 민첩성이 요구된다는 것입니다. 끊임없는 변화를 위해 계획을 세우고 움직여야 합니다.

최신 임베디드 제품을 개발하려면 네 가지 핵심적인 단계를 수행해야 합니다.

최신 임베디드 제품 개발의 네 가지 핵심 단계

1

범위를 정하라.

시장이 필요로 하는 제품의 기능 목록과 사용성에 대해 설명하는 **요구사항 정의 문서**를 작성합니다. 제품에 탑재되는 소프트웨어가 어느 정도 규모에서 얼마나 많이 사용될 것인지에 대해 경영진, 마케팅, 기술 지원, 개발팀 간 진지한 논의가 이루어져야 합니다. 또한, 고객과 만나 직접 이야기를 들어보고 경쟁 제품에 대해 조사해야 합니다. 이러한 과정을 통해 향후 필요할 것으로 예상되는 요소를 찾아내고 요구사항에 반영할 수 있습니다.

2

프로세스를 결정하기.

소프트웨어 개발자와 UX 디자이너 및 하드웨어 엔지니어 간의 상호 작용을 이해해야 합니다. 사용하는 도구와 워크플로우가 효율적이고 안정적인 제품 개발을 지원하며, 팀이 이러한 프로세스를 빠르게 받아들이고 효율적으로 일할 수 있어야 합니다.

3

소프트웨어 선택하기.

제품에 사용되는 소프트웨어 스택에 따라 무엇을 할 수 있는지, 그리고 어디서 동작할 것인지가 결정됩니다. 신뢰할 수 있는 기술 지원 및 유지보수와 함께 가장 다양한 하드웨어를 지원하는 소프트웨어를 선택하십시오. 외부 라이브러리 또는 컴포넌트에 대한 종속성이 가장 적은 소프트웨어를 선택하는 것이 좋습니다. 또한, 업계 표준을 준수하고 안정적인 API를 제공하며 재사용 가능한 소프트웨어 컴포넌트를 갖춘 소프트웨어를 선택해야 합니다.

4

하드웨어 선택하기.

제품의 소프트웨어의 요구사항을 만족하는 하드웨어 플랫폼을 선택하십시오. 전통적으로 고려되는 비용, 안정성, 신뢰성뿐만 아니라 확장(Scale-up)과 축소(Scale-down)가 유연한 하드웨어 플랫폼이라면 금상첨화입니다.

범위 산정

임베디드 프로젝트에 사용할 소프트웨어 스택의 컴포넌트를 선택하거나 개발을 처음 시작하기 전 많은 고민이 필요하지만 결국 두 개의 질문으로 정리할 수 있습니다.

여러분의 프로젝트가 포함해야 하는 범위는 어디까지인가요? 그리고 여러분의 선택이 제품의 향후 로드맵 실현을 위한 올바른 결정인가요? 프로젝트 초기에 이를 충분히 고려하지 않는다면 결국 프로그램 재개발과 수정에 많은 시간을 낭비하게 될 수 있습니다.

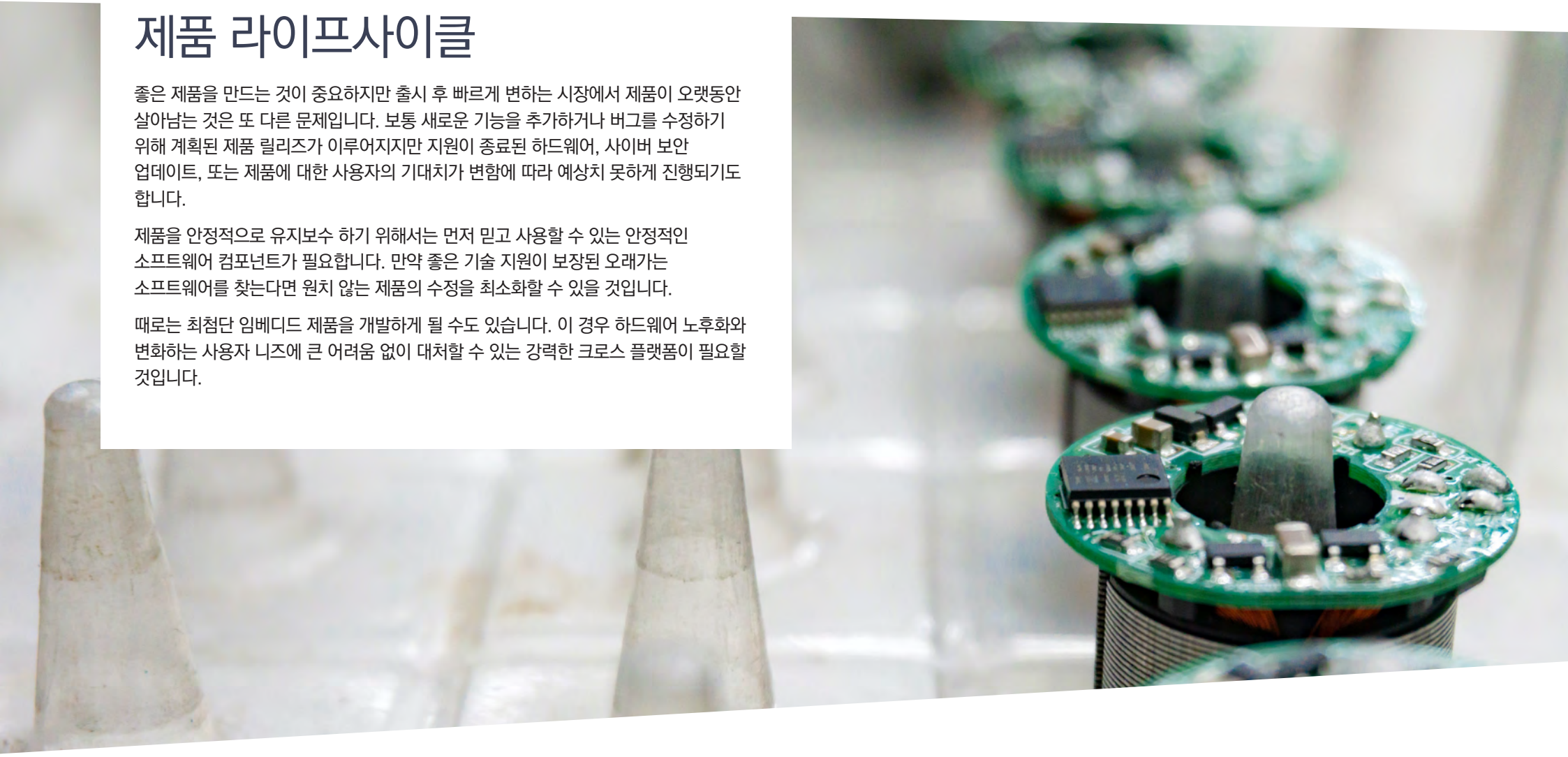
본 챕터에서는 사전 계획 단계에서 고려해야 하는 중요한 선택사항에는 무엇이 있는지 알아볼 것입니다. 여기에는 여러분들이 미처 고려하지 못했던 항목들도 포함되어 있을 것입니다.

제품 라이프사이클

좋은 제품을 만드는 것이 중요하지만 출시 후 빠르게 변하는 시장에서 제품이 오랫동안 살아남는 것은 또 다른 문제입니다. 보통 새로운 기능을 추가하거나 버그를 수정하기 위해 계획된 제품 릴리즈가 이루어지지만 지원이 종료된 하드웨어, 사이버 보안 업데이트, 또는 제품에 대한 사용자의 기대치가 변함에 따라 예상치 못하게 진행되기도 합니다.

제품을 안정적으로 유지보수 하기 위해서는 먼저 믿고 사용할 수 있는 안정적인 소프트웨어 컴포넌트가 필요합니다. 만약 좋은 기술 지원이 보장된 오래가는 소프트웨어를 찾는다면 원치 않는 제품의 수정을 최소화할 수 있을 것입니다.

때로는 최첨단 임베디드 제품을 개발하게 될 수도 있습니다. 이 경우 하드웨어 노후화와 변화하는 사용자 니즈에 큰 어려움 없이 대처할 수 있는 강력한 크로스 플랫폼이 필요할 것입니다.



재사용성 및 크로스 플랫폼

좋은 코드를 작성하는 것은 쉬운 일이 아니므로 가능한 한 여러 플랫폼에서 최대한 재사용하는 것이 좋습니다. 소프트웨어 스택을 실행할 프로세서 아키텍처와 플랫폼 및 운영 체제를 결정해야 합니다. 소프트웨어가 너무 많은 외부 종속성을 가지면 재사용과 관리가 힘들어지므로, 관리가 가능한 수준에서 몇 가지 핵심 모듈에 집중하는 것이 좋습니다.

모든 소프트웨어 컴포넌트 하나하나에 대한 조합을 고려할 필요는 없지만, 문제가 발생했을 때 여러분 스스로 해결책을 찾는다는 것은 쉬운 일이 아닙니다. 오픈 소스에 생태계에 기여하는 것은 훌륭한 일이지만, 단지 여러분이 찾던 하드웨어와 소프트웨어의 조합을 지원한다고 해서 소스의 공개의무를 지닌 오픈소스를 선택할 필요는 없습니다. 여러분에게 필요한 것을 이미 지원하는 소프트웨어를 선택하십시오.

소프트웨어 추상화

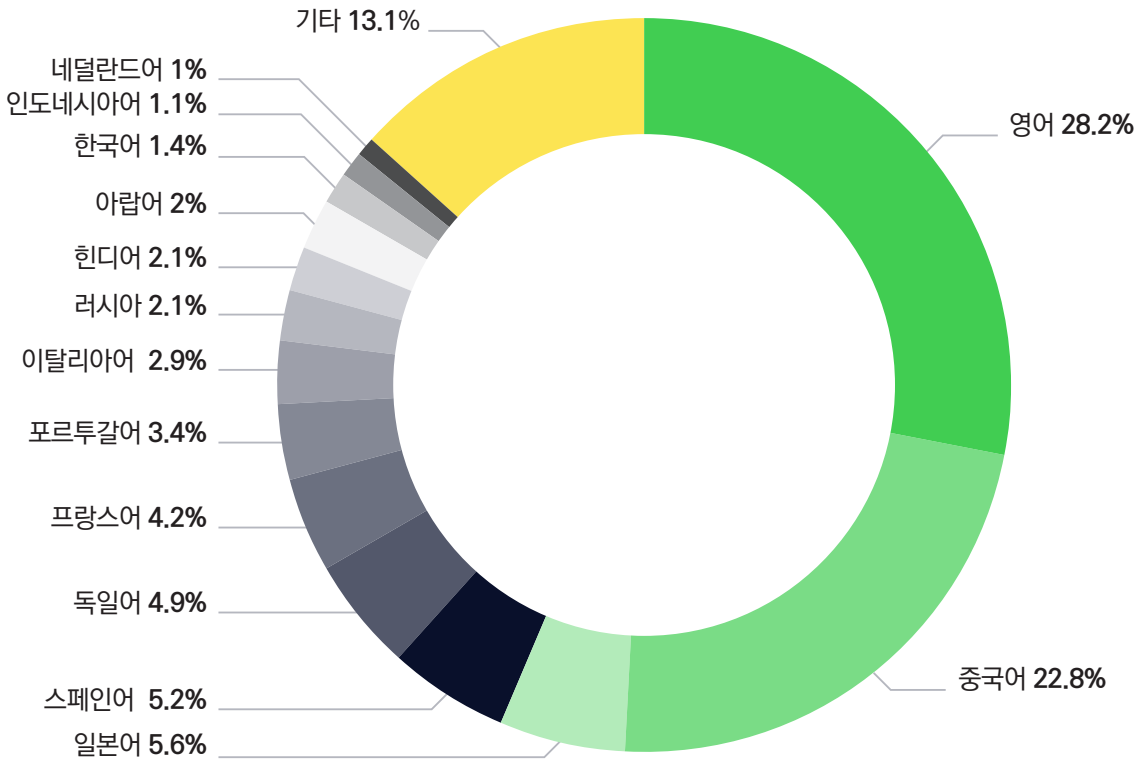
만약 여러분의 소프트웨어가 처음부터 크로스 플랫폼을 지원하지 않는다면 외부 종속성의 변경에 대한 소프트웨어의 영향을 없애기 위해 추상화 계층을 만들어야 할 수도 있습니다. 물론 이를 위한 선행 작업이 추가로 필요합니다. 그러나 개발 초기에 이러한 추상화를 잘 구현해 놓는다면, 향후 코드베이스 전체를 뜯어고치는 것에 비해 엄청난 시간을 절약할 수 있게 됩니다.

모든 것을 추상화 시킬 필요는 없습니다. 하드웨어, OS, 외부 소프트웨어 컴포넌트의 교체에 맞춰 수정이 필요할 것이라고 예상되는 부분에 집중하십시오. 처음부터 너무 완벽하게 구현할 필요는 없습니다. 아무리 고민하고 설계한다 해도 추상화가 미래의 모든 변경 사항에 대응할 수는 없기 때문입니다. 변화하는 방향에 따라 어느 정도 조정해 나갈 수 있습니다.

국제화

여러분의 제품이 하나의 언어 또는 하나의 국가만을 지원하기를 원하시나요? 아마도 아닐 것입니다. 전 세계 고객을 대상으로 시장 점유율을 높일 방법이 있다면 누구나 그렇게 할 것입니다. 비록 제품을 처음 출시할 때는 영어만 지원한다 해도, 향후 국제화(i18n, internationalization)를 쉽게 지원할 수 있는 개발 언어, 도구, UI 프레임워크를 사용하여 소프트웨어를 개발해야 합니다. 이를 위해서는 화면에 표시되는 문자를 한 곳에서 별도로 관리하여 개발자와 번역가의 편의를 높여야 합니다. 또한, UI를 쉽게 변경할 수 있도록 비즈니스 로직으로부터 분리해야 합니다.

언어별 국내 총생산 (GDP)



최신 추정 데이터: Unicode technical note #13.

연결성, 지속적인 업데이트, 그리고 보안

오늘날 대부분의 임베디드 장치는 OTA(Over-the-Air)를 통해 보안 패치, 버그 수정 및 기능 업데이트를 지원하기 위한 연결성을 제공해야 합니다. 업데이트는 기기 자체, 기업의 백엔드 서버, 사용자, 또는 이 세 가지 조합에 의해 실행될 수 있으며 이 중 가장 적합한 방법을 선택해야 합니다. 많은 개발자가 자신만의 OTA 업데이트 환경을 만들려 하는 경향이 있지만, 이미 구축된 기존의 상용 솔루션을 사용하는 것이 훨씬 좋은 선택이 될 수 있습니다. 오픈소스를 선택하는 경우 하드웨어에 대한 지원과 더불어, 지금뿐만 아니라 앞으로도 코드에 대한 활발한 유지 관리를 제공할 프로젝트를 선택해야 합니다. 또한, OTA 솔루션에 필요한 추가 스토리지를 어떻게 도입할 것인지 미리 계획해야 합니다. 이 중 일부는 두 배 이상의 플래시 용량이 필요할 수도 있습니다.

이러한 연결에는 소프트웨어 보안이 필요하며 이는 개발 완료 후 끼워 넣는 것이 아닌 처음부터 제품의 일부로 포함되어야 합니다. 업로드된 파일이나 사용자 입력 값 탈취, 또는 네트워크 해킹과 같은 최악의 상황에 노출되지 않도록 소프트웨어를 보호해야 합니다. 소프트웨어 엔지니어가 사이버 보안 모범 사례에 대한 교육을 받았는지 확인하고 필요하면 사이버 보안 전문가에게 자문을 구하십시오. 하드웨어에 대해서도 철저한 보안 조치가 필요합니다. 운영 시스템에서 개발, 디버깅, 테스트 목적을 위해 추가된 포트 또는 컴포넌트를 노출해서는 안됩니다. 또한, Secure Core 또는 TrustZone과 같은 하드웨어 잠금 및 보안 기능을 적용하여 장치를 안전하게 보호할 수 있습니다.

OTA 기능 검토 항목

- 잘못된 OTA 사용을 방지하기 위한 인증 및 암호화
- 안정적인 업데이트 전달을 위한 패치 검증, 이미지 확인 및 재시도 메커니즘
- 자동 패치 생성을 위한 통합 빌드 도구
- 고객이 사용 중인 소프트웨어 버전과 업데이트 현황에 대한 분석 및 보고
- 패키지 크기 최소화를 위한 버전 간 비교 및 압축
- 전송량 제한 및 변형 분할을 통한 서버 부하 감소
- 배포된 업데이트에 문제 발생 시 이를 회수할 수 있는 긴급 롤백 기능

모바일과 병행사용 가능한 앱

사용자의 전화기는 오늘날 임베디드 기기의 연결성을 극대화시킬 수 있는 최고의 도구입니다. 사용자와 한시도 떨어져 있지 않고 온라인에 연결하기 위한 중계기 역할을 하며 기업과 고객을 연결해 줄 수 있기 때문입니다. 하지만 서로 다른 플랫폼에서 두 개의 앱을 따로따로 만들게 되면 할 일이 두 배로 늘어나게 됩니다. 즉, 개발자가 아무리 모바일 앱 개발을 위해 Swift 또는 Kotlin을 사용하고 싶다 하더라도 애플, 안드로이드, 임베디드 플랫폼에서 모두 동작하는 단일한 개발 언어와 관련 도구를 선택하는 것이 가장 현명합니다. 이렇게 하면 모바일 앱을 하나만 만들어도 모든 장치에 배포할 수 있으며 단일한 개발 리소스, 그래픽 자료, 사용성 디자인, 코드베이스를 유지할 수 있게 됩니다.

클라우드와 그 이상

앞서 설명했던 분야 외에도 많은 영역에 IoT와 임베디드 제품을 적용할 수 있습니다.

클라우드 연동.

장치의 기능 구현을 엣지(Edge)와 클라우드로 나누어 현장에 배포되는 모든 장치에 기본적인 기능을 탑재하고 기능 수행을 위한 알고리즘과 리소스의 일부를 확장 가능한 클라우드에서 대신 수행할 수 있습니다.

통합 인공 지능(AI).

하드웨어 결함, 로그 정보, 사용 패턴에 대한 데이터를 수집하고 머신러닝을 통해 더 나은 UI와 더 스마트한 동작을 수행하는 제품을 만들 수 있습니다.

디지털 트윈.

장치의 상태나 작업 진행 상황에 대한 데이터를 수집하고 **물리적 장치를 가상화**를 통해 데스크톱이나 모바일에서 동일하게 실행할 수 있습니다.

블록체인.

시스템 전체에 분산된 트랜잭션을 검증 가능하고 변경이 불가한 신뢰할 수 있는 레코드로 보호합니다.

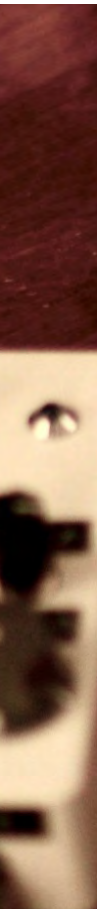
알고 계셨나요?

블록체인은 일반적으로 통화를 지칭하는 데 사용되지만, 아래와 같이 신뢰할 수 있는 데이터 저장이 필요한 임베디드 장치에도 유용하게 사용될 수 있습니다.

- 규정 준수 및 규제 검토를 위한 센서 데이터 기록
- 제삼자의 인증을 위한 메시지 유효성 검증
- 프로세스 감독을 위해 변조 방지 데이터 로깅
- 인증을 위한 장치 제조 정보 저장
- 측정 정확도 향상을 위한 교정 기록 공유

개발 프로세스 PROCESS

프로젝트를 시작하며 여러분은 품질 저하 없이 혹은 품질을 개선하는 동시에 개발 시간을 앞당기는 워크플로우를 수립한다는 목표를 세우게 될 것입니다. 이는 UX 디자인, 소프트웨어 개발, 하드웨어 엔지니어링 및 품질 보증(QA)의 네 가지 핵심 영역 간 워크플로우 간소화를 포함합니다. 더 좋은 개발 툴, 애자일 방법론과 더불어 어떻게 하면 효율적인 협업이 가능할지 고민해야 합니다.

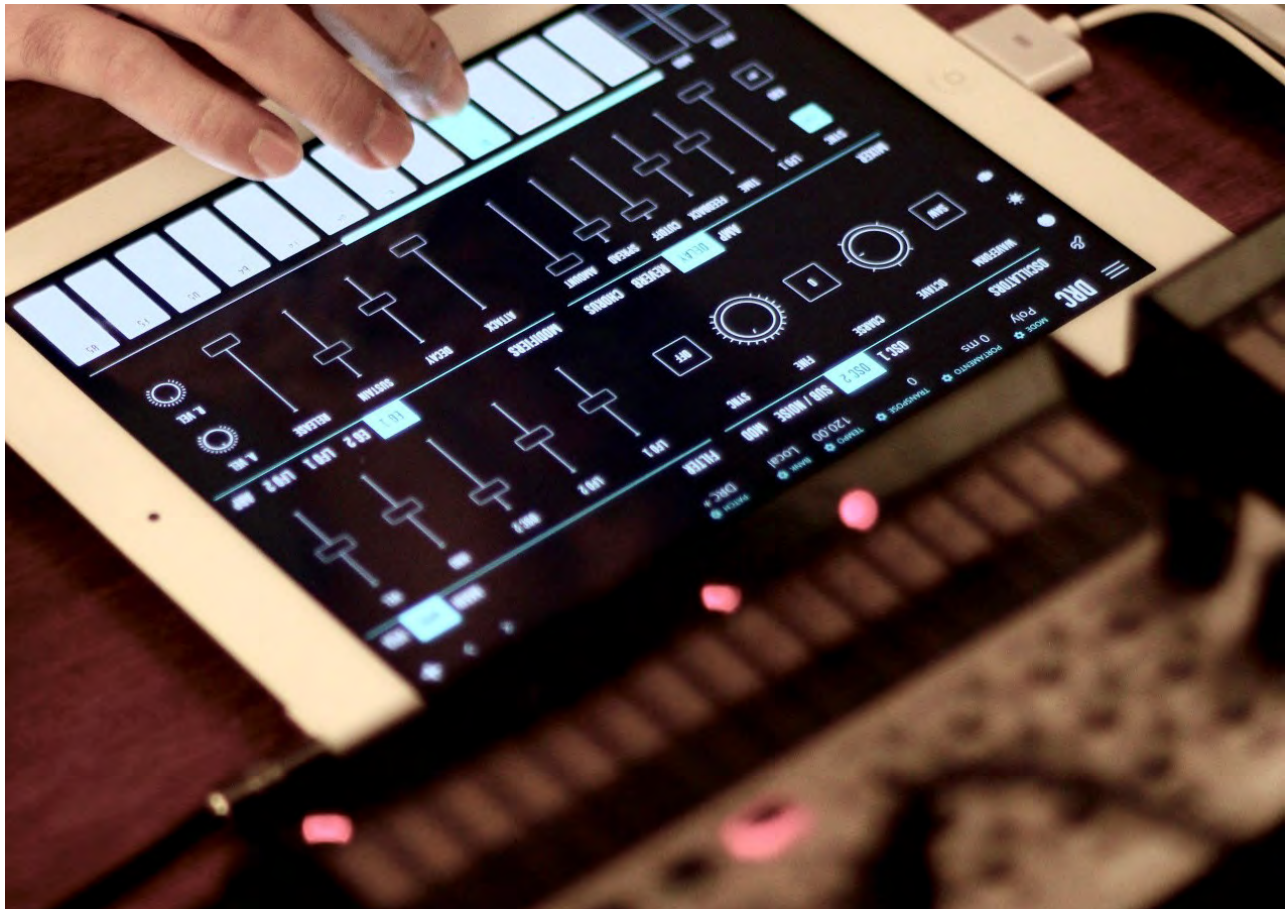


소프트웨어 개발

선택한 소프트웨어 개발 도구 모음은 새로운 개발자가 쉽고 빠르게 습득할 수 있으면서도, 동시에 경험이 풍부한 개발자가 좋은 품질 코드를 빠르게 작성할 수 있도록 효율적이어야 합니다. 이러한 도구를 통해 가벼운 바이너리, 빠른 실행 파일, 리소스가 소모가 적은 런타임 동작 등 최적의 결과물을 얻을 수 있어야 합니다. 훌륭한 오픈소스 선택지가 많이 있지만, 개발을 속도와 품질을 높이는데 도움이 되는 상용 툴이 있다면 적극적으로 고려해 봐야 합니다. 그만한 투자 가치가 있기 때문 입니다.

빠르고 안정적인 도구

여러분이 선택하는 도구가 활발하게 개발이 이루어지고 있다는 것은 매우 중요한 부분입니다. 왜냐하면, 개발팀이 새로운 프로그래밍 패러다임, 보다 효율적인 방법론, 새로운 하드웨어, 그리고 무엇보다 중요한 빠른 버그 수정이라는 이점을 누릴 수 있기 때문입니다. 이렇듯 활발한 개발이 이루어지는 툴을 선택해야 하지만, 반면 진행하는 프로젝트에 문제가 될 만큼 너무 빠르게 변경되거나 예측할 수 없는 방향으로 업데이트될 수 있다면 신중히 고려해야 합니다. 마지막으로 팀 내에서 배포와 호환성 문제를 최소화하려면 모두가 항상 같은 곳에서 동일한 개발 도구를 구해서 사용하는 것이 좋습니다.

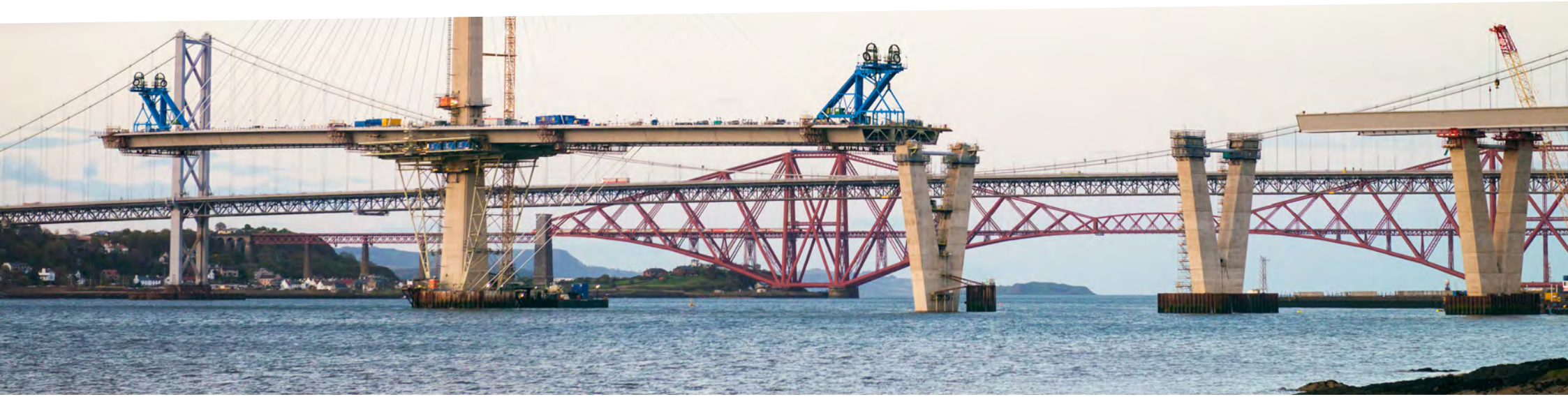


빌드하기

빌드 프로세스는 소프트웨어 개발에서 보기보다 중요합니다. 빌드를 통해 일관되고 안정적이며 추적 가능한 소프트웨어 배포가 가능합니다. 거의 모든 빌드 시스템에서 사용자가 직접 지정할 수 있는 컴포넌트를 지원하지만, 최대한 빌드 도구와 스크립트를 널리 사용되는 업계 표준에 맞추도록 노력하십시오. 여러분이 겪는 대부분의 문제는 이미 다른 사람들이 해결한 문제입니다. 이를 적극적으로 활용하세요.

테스트하기

프로젝트에 적절한 테스트 프레임워크가 없다는 것은 치명적인 리스크입니다. 테스트 스캐폴딩과 지속적 통합(CI, Continuous Integration)을 적극적으로 사용할수록 테스트를 일일이 생성할 필요가 줄어듭니다. 기존의 상용 또는 오픈소스 도구를 사용하는 또 다른 중요한 이점은 이러한 도구가 직접 개발하는 솔루션에 비해 풍부한 기능을 제공하고 잘 유지보수 된다는 점입니다. 테스트는 시스템을 통합하거나 고객이 발견하기 전에 문제를 찾아 해결할 수 있는 중요한 과정이므로 절대 소홀히 해서는 안 됩니다.



UX 디자인과 UX 워크플로우

사용자 경험(UX, User Experience)은 오늘날 제품을 차별화할 수 있는 가장 확실한 방법입니다. 사람들은 긍정적인 경험을 통해 브랜드에 충성고객이 됩니다. 훌륭한 디자이너와 개발자의 재능이 조화를 이룰 때 최고의 사용자 인터페이스가 탄생합니다. 프로젝트에서는 서로 다른 스킬셋을 보유한 이 두 종류의 전문가가 유기적으로 협업할 수 있는 프레임워크가 필요합니다.

디자인과 개발 사이클을 반복적으로 수행하는 워크플로우는 다음과 같이 시간 낭비를 줄이고 더 나은 제품을 더 빠르게 만드는데 도움을 줍니다.

- **설계대로 구현되지 않은 화면의 재작업**
- **제품 사양에 대한 잘못된 이해 또는 불완전한 산출물에 기반한 설계 오류**
- **기존의 경직된 UX 워크플로우로 인한 사용성 저하**

디자이너는 선호하는 도구를 사용하여 제품이 전하고자 하는 중요한 사용자 경험을 만들고 시각화하며 이를 프로토타입으로 제시할 수 있어야 합니다. 이러한 디자인 도구는 사용자 인터페이스(UI)와 비즈니스 로직의 명확한 분리를 지원하고 디자이너가 소프트웨어 개발 프로세스 전체에서 개선된 UI 디자인을 테스트하고 통합할 수 있어야 합니다.

회사 내에서 원활한 UX 워크플로우를 구현하려면 올바른 기업 문화가 필요합니다.

이 **백서에서는 UX**가 프로젝트의 성공에 얼마나 중요한지, 그리고 조직 내에서 어떻게 UX를 지원하는 문화를 개발할 수 있는지 자세히 소개하고 있습니다.

초기 단계부터 실제 출시할 하드웨어에서 동작하는 디자인 프로토타입을 만들어 공유하면 부서 간 불필요한 피드백 루프를 제거할 수 있습니다. 실제 임베디드 하드웨어에서 UI 디자인을 실행하면 디자이너가 화면 크기, 종횡비, 색 농도 등 실제 디자인의 품질을 시각적으로 확인할 수 있습니다. 더 중요한 것은 성능에 대한 즉각적인 피드백을 제공할 수 있다는 것입니다. UI 디자인은 임베디드 기기보다 성능이 더 높은 스마트폰을 기준으로 만들어지므로 디자이너는 설계 과정에서 UI가 과하거나 성능이 떨어지는지 미리 파악하고 수정할 수 있습니다.

통합 개발 환경

통합 개발 환경(IDE, Integrated Development Environment)은 프로그래머가 코딩, 테스트, 디버깅 등 대부분의 개발 관련 작업을 수행하는 환경입니다. 그러나 좋은 IDE는 여기서 한 걸음 더 나가야 합니다.

통합

설계, 개발, 빌드, 디버깅, 프로파일링, 테스트, 현지화, 배포 등 소프트웨어 개발 프로세스 전체를 지원할 수 있는 완전히 통합된 도구를 찾아야 합니다. 실제 하드웨어와 연동되거나 대상 하드웨어를 특정한 형태로 에뮬레이션하는 IDE를 사용하면 프로토타입을 위한 하드웨어가 충분하지 않은 경우에도 개발을 계속 이어나갈 수 있습니다.

GUI 디자인

디자이너는 위즈윅(WYSIWYG, What You See Is What You Get) 도구를 사용하여 제품의 그래픽 사용자 인터페이스(GUI, Graphical User Interface)를 만들고 시각적 요소와 동작을 정의하여 화면에 추가합니다. IDE는 이러한 GUI 디자인을 워크스테이션과 대상 하드웨어에 모두에서 직접 실행할 수 있도록 지원해야 합니다. 이는 특히 프로토타이핑과 사용성 최적화 과정에 소요되는 시간을 크게 절약할 수 있습니다. 디자이너가 특별한 교육 없이 UI를 만들 수 있는 툴을 찾는 것도 중요한 부분입니다.

에셋(Asset) 불러오기

디자이너 도구(예: Adobe Photoshop, Autodesk 3ds Max, Blender, Figma, Maya, Modo 또는 Sketch)와 통합할 수 있는 IDE를 사용하면 디자이너가 해당 Asset을 프로젝트로 바로 가져올 수 있습니다. 만약 도구가 UI 작업, 애니메이션 및 상호 작용도 불러올 수 있는 경우, 이러한 UI 요소를 개발자가 다시 구현하는 대신 디자인 팀에서 만든 그대로 사용할 수 있습니다

프로젝트 관리

버전 제어 및 빌드 시스템은 화려하지는 않지만, 안정적인 소프트웨어와 생산적인 개발 환경을 위해서는 절대적으로 필요합니다. 개발팀이 선호하는 버전 제어 시스템 및 빌드 도구와 바로 통합되는 IDE를 통해 엔지니어는 더욱 쉽고 효율적으로 작업할 수 있습니다.

확장

IDE가 풍부한 플러그인과 에코 시스템을 갖추고 있다면 회사의 워크플로우와 기존 도구를 지원할 수 있으며, 개발자는 자신에게 최적화된 환경을 구성하여 작업 효율을 높일 수 있습니다. 정적 코드 분석과 성능 메트릭스를 제공하는 플러그인은 개발자가 소프트웨어를 수정하고 최적화할 수 있도록 중요한 인사이트를 제공합니다.

유지보수

제품의 유지보수를 위한 릴리즈는 어떻게 관리할 수 있을까요? 여기서 말하는 변경 사항은 여러분이 개발한 앱 자체의 버그나 기능 업데이트가 아닌 연동된 외부 소프트웨어의 변경 사항에 관한 것입니다. 제품에 존재하는 모든 종속성(운영 체제, 드라이버, 라이브러리, 프레임워크 및 도구) 각각에서 계속 변경사항이 발생하며 이는 여러분이 제어할 수 있는 부분이 아닙니다. 이로 인해 API(Application Programming Interface)가 더 이상 동작하지 않거나 ABI(Application Binary Interface)에서 충돌이 발생하고, 개발 도구 모음에서 호환성 문제가 생기며, 소프트웨어에 대한 지원이 중단될 수 있습니다.

기존 버전에 대해서도 장기적인 지원을 제공하고 버그 수정과 보안 패치를 제공할 수 있도록 애플리케이션에 소프트웨어 빌딩 블록(Building Block)을 도입할 것을 고려해야 합니다. 이 때문에 강력하고 번창하는 개발자 커뮤니티를 보유한 외부 컴포넌트를 선택하는 것이 중요합니다.

종속성을 가진 소프트웨어를 최신 상태로 유지해야 하는 7가지 이유

1. 버그 감소
2. 더 나은 사이버보안
3. 성능 향상
4. 더 쉽고 빠른 업그레이드
5. 추가적인 기능
6. 보다 용이한 시스템 간 호환성
7. 더 나은 기술 지원

커뮤니티와 문서화

마지막으로 강조할 부분은 큰 생태계를 찾아야 한다는 것입니다. 소프트웨어 도구가 널리 사용되고 큰 생태계를 가지고 있다면 어려운 문제를 빨리 해결할 수 있습니다. 또한, 호환되는 라이브러리를 찾아 바로 사용할 수 있으며, 개발자를 더 쉽게 채용할 수 있습니다.

온라인 전용 상담창구, 스택 오버플로우(Stack Overflow) 키워드, git 샘플 코드 저장소, 구직 카테고리 등 여러 영역에서 심도 있는 기술 지원을 제공하는 솔루션을 찾으십시오. 여러분이 선택한 도구에 대한 플러그인, 애플리케이션, 커맨드라인 도구, 사용 개발자가 많다는 것은 여러분에게 필요한 솔루션을 누군가가 이미 만들어 놓았을 가능성이 크다는 것을 의미합니다. 또한, 상세한 문서를 포함한 도움말, API 예제, 온라인 팁 등을 빠르게 찾을 수 있고 좋은 개발자를 찾는 일이 더 쉬워집니다.

소프트웨어 스택 컴포넌트의 필수 요소

- 소스 코드의 공개 여부
- 공급 업체에 대한 종속성이 없어야 함
- 낮은 총 소유 비용
- 빠른 제품 출시
- 번창하는 개발자 커뮤니티
- 잘 갖춰진 생태계

소프트웨어 스택

제품의 핵심은 소프트웨어입니다. 사용하는 소프트웨어 빌딩 블록은 제품 기능의 범위와 한계를 결정합니다.

개발 환경을 구성하는 소프트웨어 스택에서 가장 많이 사용되는 기술들을 살펴보도록 하겠습니다.

OS

가장 영향력 있는 소프트웨어 스택의 컴포넌트는 운영 체제일 것입니다. 여러분이 선택한 OS에 따라 시스템에 통합할 수 있는 소프트웨어가 결정됩니다. 또한, 이로 인해 어떤 작업은 쉬워지는 반면, 다른 작업은 매우 어려워질 수도 있습니다. (물론 아무것도 없는 베어메탈에서 시작하는 것보다는 낫습니다!) 임베디드 시장에서 가장 인기 있는 8가지 운영 체제를 아래와 같이 비교해 봤습니다.

	임베디드 리눅스	안드로이드	QNX Neutrino RTOS	GreenHills INTEGRITY	Wind River VxWorks	아마존 FreeRTOS	webOS	Windows for IoT
개발 편의성	★★★★	★★★★★	★★★	★★	★★★	★★	★★★★	★★★★
효율성	★★★	★★	★★★★	★★★★	★★★★	★★★★★	★	★★★
결정론적 행동 (실시간)	★★★	★	★★★★★	★★★★★	★★★★★	★★★★	★	★★
API	POSIX	POSIX	POSIX	POSIX	POSIX	FreeRTOS	POSIX	UWP
연결성	★★★★★	★★★	★★★	★★★★	★★★	★★	★★★★★	★★★★★
그래픽	★★★★	★★★★★	★★★★	★★	★★	★	★★★★	★★★★★
하드웨어 지원	★★★★★	★	★★★	★★	★★★★	★	★★	★★★★
오픈소스	네	네	아니요	아니요	아니요	네	네	아니요
커뮤니티	★★★★★	★★★★	★★★	★★	★★	★★	★	★★★
라이센스 비용	$	$	$$$$	$$$$	$$$	$	$	$$$$$
수정 및 기능 강화 비용	$$$	$$$$	$$	$$	$$	$$$	$$$	$

★가 가장 낮고 ★★★★★가 가장 높은 평점입니다. 약어는 부록 A를 참조해 주세요.

본 문서에서는 시장의 변화에 따른 상대적인 위치 보다는 각 기술 본연의 특징에 집중하여 각 소프트웨어 및 하드웨어가 가진 강점을 쉽게 비교할 수 있도록 평가했습니다. (각 소프트웨어는 빠르게 개발되고 있으므로 이를 최신 상태로 유지하며 하나로 정리할 수 있는 방법은 없기 때문입니다). 또한, 위 표는 대형(산업, 의료, 자동차), 중형(인포테인먼트, 백색 가전, 키오스크) 및 소형(휴대용 장치, IoT, 웨어러블) 장치에 일반적으로 적용할 수 있습니다. 하나의 컴포넌트에 여러 버전이 존재하는 경우 가장 널리 사용되는 버전에 대한 평점을 보여줍니다. 여기에는 가장 인기있는 하드웨어 위주로 선정했지만, 공급 업체로부터 지원되는 제품 전체에 대해 확인하십시오.

컨테이너 및 하이퍼바이저

제품의 소프트웨어 아키텍처를 지원하는 두 가지 강력한 기법인 컨테이너와 하이퍼바이저는 데스크탑과 클라우드의 전유물이었지만, 오늘날 강력한 프로세서에 힘입어 임베디드 환경에서도 이를 사용할 수 있게 되었습니다.

컨테이너는 **임베디드 프로젝트 개발을 가속화**할 수 있습니다. 매번 일일이 설정할 필요 없이 일관되고 즉각적인 툴 사용이 가능하고, 테스트 프레임워크를 표준화하며 여러 개의 독립적인 도구/라이브러리 구성을 유지할 수 있습니다. 또한, 직관적인 장치 프로비저닝과 하드웨어 공급 업체 업데이트를 손쉽게 적용할 수 있으며, 동일한 코드베이스에서 우수한 테스트를 수행함으로써 **타겟 임베디드 장치에서 유용하게 사용**될 수 있습니다.

하이퍼바이저는 리소스 보호, 샌드박스 제공, 독립적인 작업을 제공하는 데 중점을 두고 있습니다. 이를 통해 **하나의 칩에서 동시에 여러 운영 체제(OS)를 실행**할 수 있습니다. 예를 들어 시스템의 핵심 기능이나 보안 기능은 QNX Neutrino RTOS 또는 GreenHills Integrity RTOS와 같은 OS에서 실행하고, 임베디드 리눅스와 같은 두 번째 OS에서 제품의 기본 UI를 실행하며, 안드로이드 또는 webOS와 같은 OS에서 다운로드 가능한 써드파티 앱을 제공할 수 있습니다. 하이퍼바이저는 안전 기능에 대한 요구 사항이 있는 장치를 개발할 때 핵심적인 역할을 할 수 있습니다.

프로그래밍 언어

OS상에 동작하는 애플리케이션을 개발하려면 프로그래밍 언어가 필요합니다. 모든 프로그래밍 언어는 개발 프로세스 측면에서 장단점을 가지고 있고, 언어에 따라 임베디드 애플리케이션 개발에 큰 영향을 미치게 됩니다. 제약에는 각 언어가 지원할 수 있는 GUI에도 제약이 있기 때문에 신중하게 살펴봐야 합니다. 여기에서는 임베디드 업계에서 가장 많이 사용되는 언어 위주로 살펴보겠습니다.

	C	C++	Java	Python / MicroPython	JavaScript/ HTML5/CSS	Rust
강점	임베디드, 베어메탈, IoT	임베디드, 베어메탈, 독립형 앱, IoT	웹, 클라우드	클라우드, 데이터 사이언스	웹	임베디드 베어메탈
개발 편의성	★★★	★★	★★★	★★★★★	★★	★★★★
표현력	★	★★★★	★★	★★★★★	★★★	★★★★
쉬운 유지보수	★★★★	★★★★	★★★★	★★★	★	★★★★
언어의 생명력	★★★★★	★★★★	★★★	★★★	★	★
런타임 효율성	★★★★★	★★★★★	★★★	★★	★	★★★★
라이브러리/모듈 가용성	★★★★	★★★★	★★★	★★★★	★★★★	★★
저수준 인터페이스	★★★★★	★★★★★	★★	★★★	★	★★★★
연결 지원	★★	★★★ / ★★★★★[1]	★★★★	★★★★★	★★★★★	★★
그래픽 지원	★★★★	★★★★★	★★★	★★★	★★★★★	★
개발자 커뮤니티	★★★	★★★	★ / ★★★★[2]	★★★★	★★★★★	★★

★가 가장 낮고 ★★★★★가 가장 높은 평점입니다. 약어는 부록 A를 참조해 주세요.

1 후자: 플랫폼 라이브러리가 포함된 경우로, 높은 수준의 연결과 네트워킹을 지원함
2 전자: 안드로이드 전용 커뮤니티 – 낮음. 후자: 안드로이드를 포함한 전체 커뮤니티 – 높음.

GUI 프레임워크

기술 스택에서 가장 중요한 부분 중 하나는 그래픽 사용자 인터페이스(GUI)입니다. 애플리케이션에서 사용하는 GUI 프레임워크는 애플리케이션의 외관뿐만 아니라 어떠한 기능을 구현할 수 있을지, 얼마나 빨리 기능과 제품을 시장에 출시할 수 있을지, 어떤 프레임워크를 지원할 수 있을지에 영향을 줍니다. 이미 제작된 테스트를 마친 코드는 직접 디자인, 코딩, 테스트를 거치지 않고 그대로 가져다 쓸 수 있습니다.

크로스 플랫폼

GUI 프레임워크의 주요 특징은 크로스 플랫폼으로 요약될 수 있습니다. 얼마나 많은 플랫폼을 지원하며 모두 일관되게 동작하는지가 매우 중요합니다. 하드웨어를 바꾸거나 다양한 OS를 지원할 때, 그리고 임베디드 제품과 모바일 및 데스크톱 컴패니언 앱 간에 소스 공유가 필요한 경우 크로스 플랫폼에 대한 지원을 필수입니다.

오늘날 성장과 변화는 선택이 아닌 필수입니다. 지금 당장의 요구사항만을 기준으로 플랫폼을 선택하는 실수를 범하지 말아야 합니다. 다양한 데스크톱 OS(리눅스, MacOS, 윈도우), 모바일 OS(안드로이드, iOS), 임베디드 OS(QNX, INTEGRITY, Linux/Wayland, Linux/X11, UWP, VxWorks) 뿐만 아니라 수많은 하드웨어 아키텍처와 플랫폼을 폭넓게 지원하는 GUI 프레임워크를 찾으십시오.

강력한 크로스 플랫폼 GUI는 다양한 방식으로 제품을 지원합니다

빠른 제품 출시.

사전에 패키지된 광범위한 기능, 프로그래머를 위한 다양한 기능, 적극적인 기술 지원, 활발한 커뮤니티를 피드백 등을 갖춘 도구는 개발자의 생산성을 크게 높이고 제품의 출시를 앞당길 수 있습니다.

일관된 경험.

좋은 도구는 사용하는 플랫폼과 관계없이 사용자에게 동일한 경험을 제공하므로 모든 플랫폼에서 일관된 방식으로 사용할 수 있습니다

코드 재사용.

플랫폼 간의 기능, API, 빌드 환경 및 에셋 관리의 일관성을 통해 개발자는 임베디드, 모바일, 데스크탑 솔루션 간 코드를 쉽게 이동할 수 있습니다.

개발 용이성.

크로스 플랫폼에서 동작하는 강력한 도구 모음을 통해, 개발자는 하드웨어마다 따로 개발하거나 빌드할 필요가 없어지고 자신이 선호하는 환경에서 개발하며 생산성을 높일 수 있습니다.

컴포넌트

모든 GUI 프레임워크는 위젯뿐만 아니라 다양한 기능을 지원합니다. 2D/3D 디자인 및 시각화, 임베디드 브라우저, 주변기기 지원, 공통 네트워크 프로토콜, 국제화, 다양한 멀티미디어 지원, 데이터베이스 통합, 센서 접근, 차트 작성 등 다양한 컴포넌트를 제공하는 프레임워크를 찾아보십시오. 프레임워크에서 기본적으로 제공하는 요소들은 디자이너와 개발자의 업무 부담을 줄여줄 수 있습니다.

프로그래밍 언어

프레임워크는 다양한 언어에 대한 API를 제공하므로 개발팀은 가장 생산성 높은 언어를 선택하여 개발할 수 있습니다. 또한, 개발자가 작업에 가장 적합한 언어를 선택할 수 있는 유연성을 제공합니다.

예를 들어, 프레임워크가 C ++와 Python을 모두 지원하는 경우 개발팀은 성능이 가장 중요할 때는 C++를 사용하고 개발 속도가 우선시 될 때는 Python을 선택할 수 있습니다.

개발 지원

자체적인 IDE가 있거나 VSCode, Visual Studio 또는 Eclipse와 같은 널리 사용되는 IDE용 플러그인을 제공하는 GUI 프레임워크를 찾으십시오. 선택한 UI는 쉽게 디자인, 빌드, 테스트 및 디버깅할 수 있어야 합니다. 상황에 따라 특화된 개발 환경이 필요하다면 이 역시 중요한 요소가 될 수 있습니다.

룩앤필:
운영체제, 프레임워크, 혹은 자체 제작

제품의 룩앤필(Look and Feel)이 가지는 장단점에 대해 항상 논쟁이 있을 것입니다. OS에 특화된 위젯과 프레임워크에서 제공하는 기본 UI 요소 중 무엇을 사용할지, 아니면 직접 디자인 할지 등 많은 선택사항이 있습니다. 안타깝게도 플랫폼에서 제공하는 UI를 항상 사용할 수 있는 것은 아닙니다. 많은 임베디드 제품은 기본 UI가 없는 Linux를 사용합니다. 당연하게도 제품이 여러 OS에서 실행되는 경우 각 플랫폼에서 서로 다르게 보일 수밖에 없습니다. 이러한 UI는 프레임워크가 제공하는 것일 수도 있고 직접 디자인한 것 일 수도 있겠지만, 누가 보더라도 쉽게 이해할 수 있는 보편적이고 깔끔하며 매력적인 UI를 선택하는 것이 최선의 선택일 것입니다. 이를 통해 전체 제품 라인에서 브랜드의 일관된 룩앤필을 전달할 수 있습니다.

헤드리스 UI

임베디드 장치에 화면이 없다면 어떻게 될까요? 많은 IoT 제품이 이에 해당됩니다. 헤드리스 UI 제품 자체가 아닌 사용자의 브라우저에 GUI를 제공함으로써 훨씬 저렴한 비용으로 뛰어난 사용자 경험을 제공합니다.

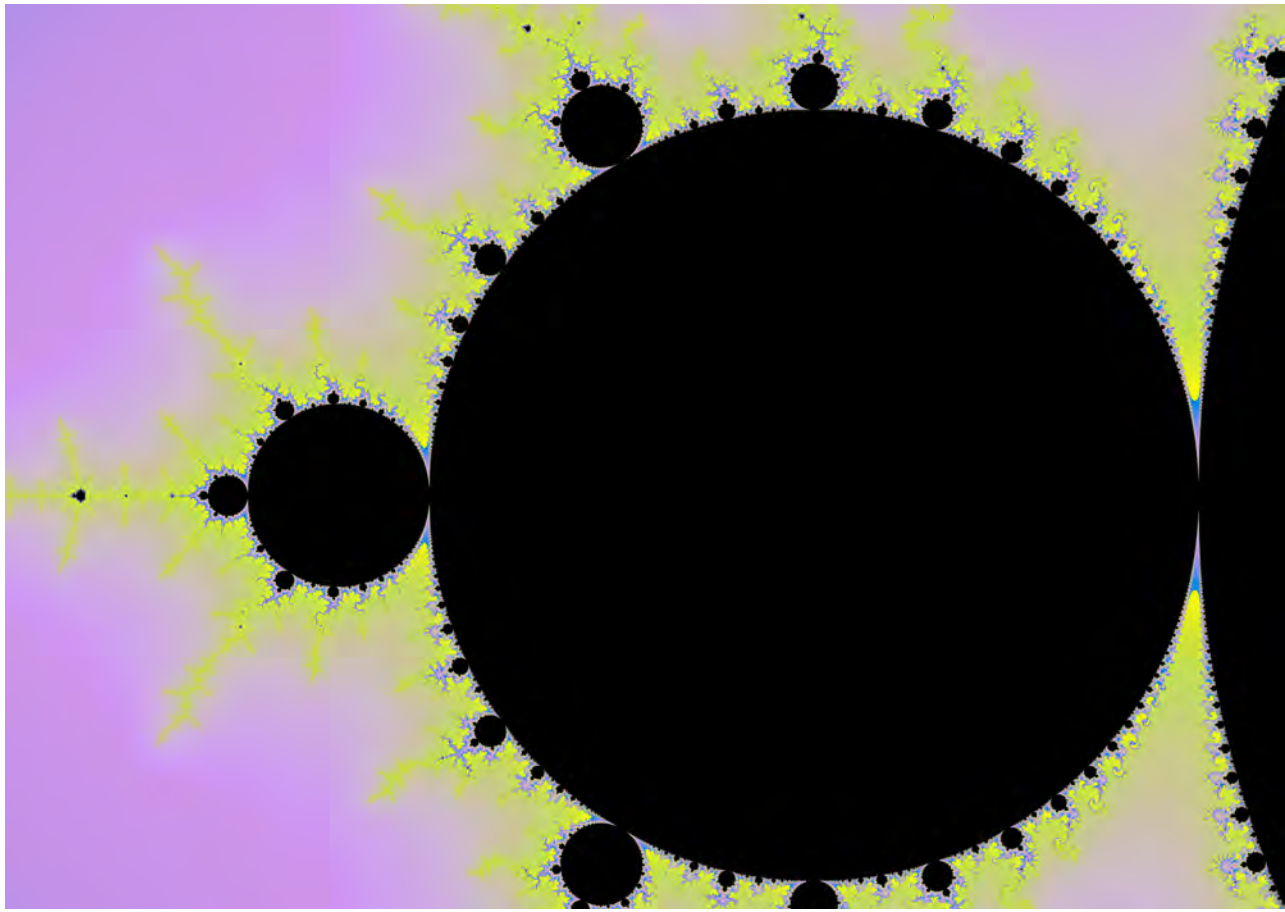
이러한 UI를 구축하기 위해 HTML, CSS, JavaScript를 직접 코딩할 필요가 없으며 이제는 그렇게 해서도 안 됩니다. 매력적인 브라우저 기반 UI를 직접 개발하는 노력을 덜어줄 많은 자바스크립트 프레임워크가 있으므로 이 중에서 선택하기만 하면 됩니다. 일부 GUI 프레임워크는 WebGL 또는 WebAssembly를 사용하여 UI 코드를 내보낼 수도 있으므로 웹 기반이 아닌 GUI 환경에서도 브라우저 기반의 헤드리스 UI를 제공할 수 있습니다.

라이브러리

애플리케이션 소프트웨어에는 GUI 그 이상의 것을 필요로 합니다. 제품을 개발하기 위해서는 다음과 같이 이미 개발과 테스트가 완료된 라이브러리가 필요할 수도 있습니다: 네트워크 프로토콜, JSON 및 XML 구문 분석, 머신러닝, 행렬 계산, 고속 푸리에 변환(FFT, Fast Fourier Transforms), 난수 생성기, 암호화, 다중 스레딩 지원, 이미지 처리, 정규식, 데이터베이스 지원 등.

대부분은 직접 구현하기 쉬운 기능이 아니며, 코드가 복잡할수록 직접 개발하기보다 오랜 시간 검증을 거친 소프트웨어에 더 많이 의존하는 것이 좋습니다. 여기에 더해 GUI 프레임워크에 기본적으로 포함된 많은 라이브러리를 사용할 수 있습니다. 또한, 구현 언어에 따라 Boost, Apache Portable Runtime(APR), NumPy, TensorFlow 등 훌륭하게 지원되고 있는 여러 외부 라이브러리를 조합해서 사용해야 할 수도 있습니다.

필요한 기능이 무엇이든지 간에 좋은 요구사항을 만족할 수 있는 가장 적은 숫자의 라이브러리를 사용하는 것이 좋으며 높은 호환성, 쉽게 이해할 수 있는 API, 훌륭한 문서화를 갖춘 라이브러리를 선택해야 합니다.



하드웨어

지금까지 소프트웨어에 필요한 것이 무엇인지 알아봤으니 이제 하드웨어에 대해 알아볼 차례입니다. 제품이 요구하는 기능과 향후 로드맵을 고려하면 좀 더 쉽게 하드웨어를 선택할 수 있습니다. 예를 들어, 제품에 유려한 3D 애니메이션이 필요한 경우 하드웨어 가속 그래픽이 탑재된 장치를 사용하는 것이 좋습니다. 빠른 모바일 연결이 필요한 경우 WiFi ac 또는 USB, 혹은 둘 다 필요합니다. 제스처, 터치 또는 음성을 포함하는 상호 작용을 지원해야 하는 경우 이에 필요한 하드웨어와 센서가 필요합니다.

지금부터 어떤 하드웨어를 선택할 수 있는지 살펴보겠습니다.

제품 라인 및 확장성

임베디드 시스템을 구동할 수 있는 장치는 3GHz 64비트 16코어 프로세서에 멀티 디스플레이 4K GPU를 갖춘 고성능 기기에서, 4개의 입출력 핀을 갖춘 작은 8비트 마이크로칩에 이르기까지 광범위합니다.

지금쯤이면, 제품의 현재와 미래 요구사항은 무엇이며, 어디까지 확장할 수 있을지 명확히 이해하고 있어야 합니다. 호환 가능한 기기의 제품군 내에서 요구사항의 범주를 포괄할 수 있는 하드웨어 선택하십시오. 하이엔드와 로우엔드 선택지를 모두 만족시킬 수 있는 시스템과 아키텍처를 고려하는 것이 가장 좋습니다.

제품 라인의 다양한 제품에서 재사용 가능한 컴포넌트를 선택하면 제품의 파편화를 피할 수 있습니다. 또한, 수명이 긴 제품이 필요한 시장인 경우 선택한 부품이 향후 수년간 사용하는 데 문제가 없는지 확인하십시오.

즉, 소프트웨어 스택이 필요로하는 간결한 처리, 속도, 메모리 요구 사항 등 하드웨어 요구사항을 염두에 두어야 합니다. 개발 사이클 후반에 어쩔 수 없이 하드웨어를 교체하는 경우 더 많은 비용이 들어가게 되므로 비용 절감 부담으로 인해 무조건 저사양 하드웨어를 선택하는 실수를 범하지 않도록 주의하십시오

SoC/CPU

시스템 온 칩(SoC)의 중앙 처리 장치(CPU)는 임베디드 시스템에서 가장 중요하고 영향력이 크며 가장 비싼 하드웨어 컴포넌트입니다. 또한, 한 번 선택하면 나머지 제품군에 전체가 이를 따라야 할 가능성이 큽니다. 다음은 가장 좋은 성능을 제공하는 인기있는 기성 SoC 제품의 평가 비교표입니다.

	AMD	Intel	NXP	Renesas	TI
평가한 보드	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
메모리	4 DDR4, 2 DIMM (최대 32GB), ECC 옵션	2GB 온보드 DDR (최대 8GB), 32GB eMMC	LPDDR4 (x64), 32GB eMMC	384K sys RAM, DDR4, 80MB 온보드 플래시, 8GB eMMC, 마이크로 SD	2GB DDR3L, 4GB eMMC, 마이크로 SD
주변기기	HDMI, 4 x DP++, 4K, Vega GPU, PCIe, SATA, 오디오, GigE, USB 3.1+2.0	HDMI, Intel HD 500 GPU, 오디오, GigE, PCIe, SATA, USB 3.0 + 2.0, WiFi ac, BT	MIPI, LVDS, 4K, PMIC, GPU,PCIe, 오디오, GigE, USB 3.0, CAN, 최대 4개 HDMI 디스플레이	HDMI, PowerVR GPU, LVDS,WiFi, BT, 오디오, Eth 10/100, USB 2.0	7인치 정전식 터치 스크린, HDMI, 오디오, PowerVR GPU, GigE, SATA, MiniPCIe, USB 3.0
아케텍처	x86-64	x86-64	ARM-64	ARM-64	ARM-32
연산능력	V1807B x 4 core @ 3.8GHz	J3455 x 4 core @ 1.5GHz	A72 x 2 core @ 1.6 GHz; A53 x 4 core @ 1.2 GHz; M4 x 2 core @ 266 MHz	A57 x 4 core @ 1.5 GHz; A53 x 4 core @ 1.2 GHz	A15 x 2 core @ 1.5GHz, M4 x 2 core
사용처	대형	대형	대형	대형	대형
전력 소모	★	★	★★	★★	★★★
수명	★★★	★★★	★★★★★	★★★★★	★★★★★
특화 기능	최대 4개의 4K 디스플레이	NUC는 소형 PC지만 SoC는 임베디드임 – 호환가능	가속, 자이로, 압력, 광 센서	자동차, EAVB, 440 핀 확장	카메라 옵션, 2 x C66x DSP

★가 가장 낮고 ★★★★★가 가장 높은 평점입니다. 약어는 부록 A를 참조해 주세요

* SoC 제품은 위 평가표에 없는 추가적인 기능을 제공할 수 있습니다.

단일보드컴퓨터(SBC, Single-Board Computer)

만약 맞춤형 서킷 보드 없이도 만들 수 있는 제품이라면 아래에 있는 다양한 단일 보드 컴퓨터(SBC, Single-board Computer) 중에서도 선택할 수 있습니다

	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
메모리	8GB RAM, 마이크로 SD	4GB RAM, 32GB flash, 마이크로 SD	8GB RAM, 32GB eMMC	LPDDR4 (최대 4GB), eMMC, 마이크로 SD	2GB RAM, 4GB 플래시, 마이크로 SD	64MB RAM, 16MB 플래시, 마이크로 SD
주변기기	µHDMI, 4K, MIPI, VideoCore VI GPU, WiFi ac, GigE, USB 3.0+2.0, BT 5.0	Intel Gen8 GPU, HDMI, 오디오, WiFi ac, GigE, USB 3.0+2.0, BT 4.2	HDMI 2.0, 4K, GP10B GPU, GigE, MIPI, 6 개의 카메라, USB 3.0+2.0	4K, MIPI, Mali GPU, 오디오, GigE, USB 3.0+2.0,	HDMI, PowerVR GPU, 오디오, GigE, SATA, USB 3.0+2.0	오디오, 100Eth, WiFi b/g/n, USB 2.0
아케텍처	ARM-32	x86-64	ARM-64	ARM-64	ARM-32	MIPS32
연산능력	A72 x 4 core @ 1.5GHz	Atom x 4 core @ 1.44GHz	Denver 2 x 2 core @ 2.0GHz, A57 x 4 core @ 2.0 GHz	A72 x 2 core @ 2.0GHz, A53 x 4 core @ 2.0GHz	A15 x 2 core @ 1.5GHz; M4 x 2 core @ 212 MHz	24K @ 400MHz
사용처	중형	대형	대형	중형	중형	소형
전력 소모	★★★	★	★★★	★★★	★★★	★★★★★
수명	★★★★	★★	★★	★★★★	★★★	★★
특화 기능	광범위한 생태계	윈도우 호환	256 AI 또는 컴퓨터 비전, 비디오 인코더/ 디코더를 위한 CUDA 코어	Wifi ac + BT 및 터치 패널 옵션 모듈, 저전력을 위한 big.LITTLE 아키텍처	2개의 TMS DSP, 지연 시간이 짧은 I/O 제어를 위한 4개의 PRU	ATmega32U4 마이크로 온보드

★가 가장 낮고 ★★★★★가 가장 높은 평점입니다. 약어는 부록 A를 참조해 주세요.

* SoC 제품은 위 평가표에 없는 추가적인 기능이 제공할 수 있습니다.

램(RAM)

보통 임베디드 보드에서 CPU 다음으로 비싼 컴포넌트가 RAM입니다. 임베디드 장치에 램이(RAM, Random-Access Memory) 충분하지 않으면 애플리케이션이 메모리 부족으로 인해 문제를 발생시킬 수 있습니다. 이러한 치명적인 상황을 조기에 발견하려면 시스템에서 메모리 소비가 가장 큰 부분(OS, 그래픽 UI, 데이터베이스 등)을 대상 하드웨어에서 실행하고 실제 운영 환경을 최대한 가깝게 재현하는 테스트 환경에서 독립적으로 실행해야 합니다. 최대 메모리 사용량을 주의깊게 관찰하고 실제 사용량뿐만 아니라 낭비되는 공간까지 모두 합산합니다.

가장 큰 메모리를 소모하는 컴포넌트를 정확히 찾아낼 수 있도록 애플리케이션 라이프사이클 동안 메모리 사용량을 정확히 모니터링 할 수 있는 도구가 필요합니다. 사용 패턴에 따른 메모리 사용량을 분석을 통해, 메모리를 많이 사용하는 애플리케이션의 설정을 바꿔가며 최적화를 수행합니다. 이를 통해 실제 운영에서 필요한 값을 찾을 수 있습니다. 여기에는 그래픽 디스플레이 버퍼, 오디오 버퍼 또는 디스크 캐시와 같이 별도로 할당된 램의 사용량도 포함됩니다.

메모리 도구

여러분이 개발한 소프트웨어가 메모리 문제를 일으키지 않는지 테스트를 통해 반드시 검증해 보아야 합니다. Cppcheck, heob, GammaRay 또는 valgrind와 같은 도구를 통해 코드의 정적/런타임 분석을 수행하고 과도한 메모리 할당과 누수를 찾는 데 매우 유용하게 사용할 수 있습니다.

플래시

대부분의 보드는 마이크로 SD 카드 또는 다른 외부 플래시 인터페이스를 지원합니다. 개발자가 디스크 공간에 충분한 여유를 두고 개발하고, 개발이 끝났을 때 실제로 필요한 용량에 맞춰 최적화하는 것이 가능합니다. 플래시의 더 큰 문제는 보통 속도입니다. 플래시가 충분히 빠르지 않다면 빠른 디스크 응답을 요구하는 애플리케이션에서 문제가 발생합니다.

SD 카드(표준, 미니 또는 마이크로)에서 클래스(Class) 혹은 비디오 클래스(Video Class)의 숫자는 쓰기 속도, 즉 MB/초를 의미합니다. 클래스 10카드는 10MB/초를 의미하고 V90 카드는 90MB/초의 쓰기 능력을 갖추고 있음을 나타냅니다. 다양한 등급의 마이크로 SD 카드를 여러 개 사용해 보면 쉽게 비교할 수 있습니다. 서로 다른 메모리 카드를 교체해가며 플래시 속도의 영향을 간단하게 측정함으로써 비용 대비 최적의 속도 값을 도출해 낼 수 있습니다.

만약 여러분이 평가하는 보드에 온보드 플래시가 있는 경우 이를 최대한 활용해 볼 필요가 있습니다. 대부분 SD 카드보다 더 나은 안정성과 빠른 성능을 제공하기 때문입니다.

플래시 포맷 방식

플래시 파일 시스템은 장치 웨어 레벨링(Wear Leveling), 오류 감지/수정, 오류 복원력을 관리하기 위해 각기 다른 전략을 사용하며 장단점을 가지고 있습니다. 특히, 개발하는 앱이 다음 중 하나에 특화되어야 하는 경우 여러 파일 시스템에서 앱을 테스트하며 무엇이 가장 우수한 성능을 보이는지 확인해보세요.

- 빠른 부팅 속도
- 고속 대량 읽기
- 매우 빠르고 연속적인 쓰기
- 작은 파일이 무수히 많이 존재
- 고성능 디스크 압축
- 신속한 디렉토리 스캔

마이크로 컨트롤러(MCU, Microcontroller Unit)

오늘날 마이크로 컨트롤러(MCU, Microcontroller Unit)는 매우 뛰어난 성능과 더 많은 기능을 제공합니다. 제품이 초고성능을 요구하지 않는다면 최고의 선택이 될 수 있습니다. 이러한 칩은 일반적으로 MMU(메모리 관리 장치)가 없으므로 애플리케이션을 베어메탈에서 바로 실행하거나, Azure RTOS ThreadX 또는 Amazon FreeRTOS와 같은 비 MMU 칩을 지원하는 OS를 선택해야 합니다. MCU를 사용하려면 램과 플래시 사용에 대해 특별히 더 주의를 기울여야 합니다. 여기에서는 다양한 기능과 아키텍처를 가진 마이크로 컨트롤러를 살펴 보겠습니다.

	Microchip	NXP	Renesas	ST	TI
평가한 보드	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
메모리	32MB SDRAM, 4MB 플래시, 마이크로 SD	64MB RAM, 336MB flash,SD 카드	4MB RAM, 5MB flash, DDR2, ECC 옵션	532KB RAM, 2MB 플래시, 마이크로 SD	04KB RAM, 1MB 플래시
주변기기	5.0” WVGA, 2D GPU, 오디오, Eth 10/100, USB 2.0, CAN, 6 x SPI, ADC	5.5” 720p 디스플레이, OpenVG GPU, MIPI, 오디오, USB, CAN, GigE, EAVB, SIM	2D GPU, sprites, LVTTL 비디오 입력, 오디오, 2 x I2C, 6 x CSI, 3 x CAN, Eth 10/100, EAVB	4” 터치 디스플레이, 2D GPU, MIPI, 오디오, USB, 4 x I2C, 6 x SPI, 3 x CAN, Eth 10/100	USB, ADC/DACs, PWMs, CAN
아키텍처	MIPS32	Cortex-M7 + M4	RH850	Cortex-M7	C28X
프로세서 속도	252MHz	M7 @ 1GHz, M4 @ 400MHz	240MHZ	216MHz	200MHz x 2
동작 온도	-40 to 125°C	-40 to 125°C	-40 to +150°C	-40 to +105°C	-40 to 125°C
전력 소모	★★★★	★★★	★★★★	★★★★	★★★★
수명	★★★★	★★★	★★★	★★★	★★★★
특화 기능	MEB 확장 모듈을 통한 디스플레이, 암호화 옵션 제공, Raspberry Pi 헤더	Mag + 가속 센서, Arduino 헤더	자동차	WiFi 모듈 옵션, Arduino 헤	CLA FPU 2 개, Trig + Complex math units, BoosterPack 헤더, 그래픽 지원 없음

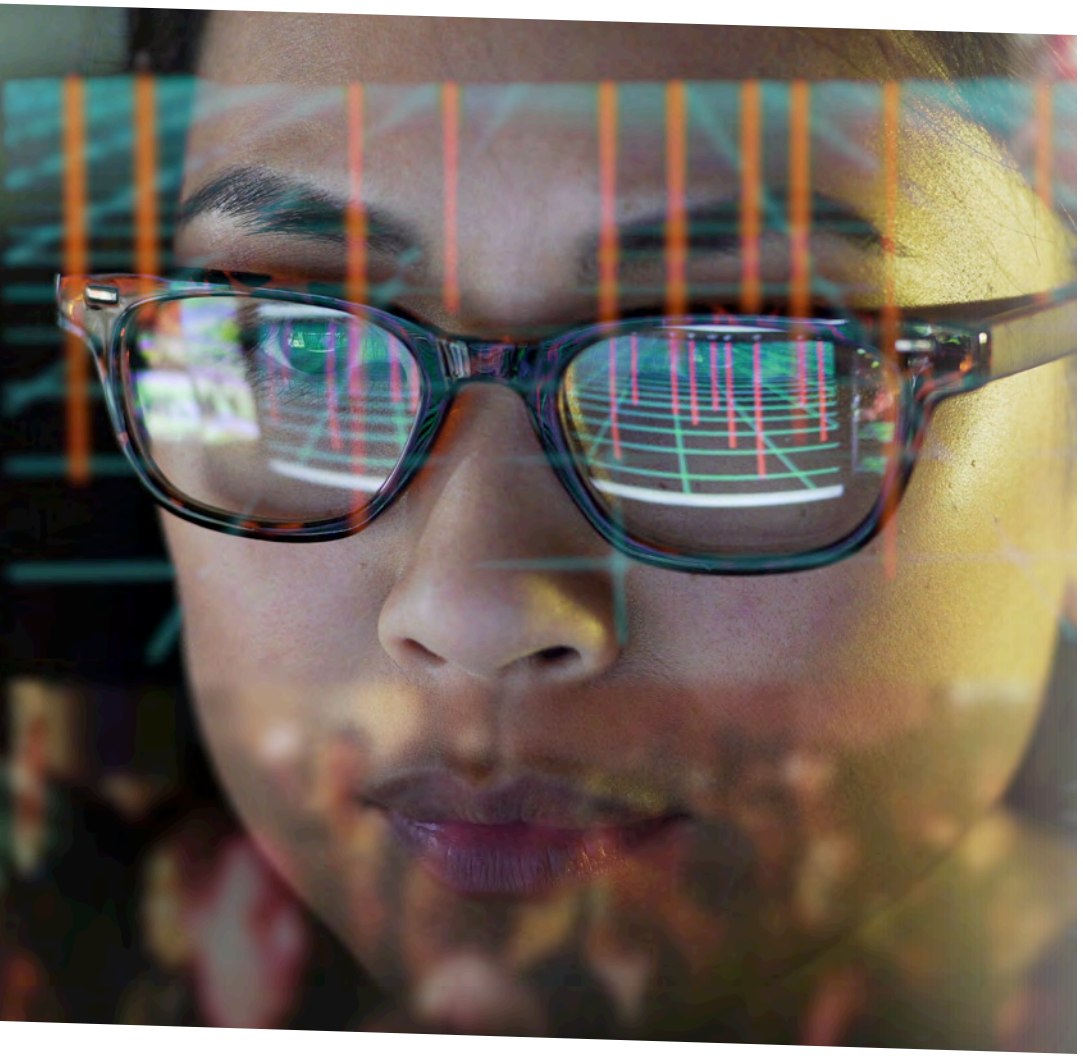
★가 가장 낮고 ★★★★★가 가장 높은 평점입니다. 약어는 부록 A를 참조해 주세요

사용자 I/O

사용자 인터페이스와 입출력(I/O) 메커니즘은 가장 큰 제품 차별화 요소 중 하나지만 제품의 비용과 복잡성으로 이어질 수 있습니다. 다음은 장치가 사용자와 상호작용하는 방식을 설계할 때 고려해야 할 요소입니다.

	터치스크린 2D	터치스크린 3D	오디오	음성	물리적 조작	햅틱 조작	제스처
하드웨어	스크린, 정전식 패널	화면, 정전식 패널, 그래픽 처리 장치 (GPU)	스피커, 디지털 아날로그 컨버터 (DAC)	마이크, 아날로그 디지털 컨버터 (ADC), 스피커, DAC, 음성 지원 인터페이스	회전식 다이얼, 스위치, 버튼, 슬라이더, 발광 다이오드(LED) 상태 램프	회전식 다이얼이, 스위치, 모터	카메라, 근접/압력 센서, IR 카메라, 라이더 센서
가격	$$$$$	$$$$$	$	$$$$	$	$$	$$$$
풍부한 상호작용	★★★★	★★★★★	★★	★★★★	★	★½	★
사용자가 인터페이스 의도를 이해하는 속도	★★★★★	★★★★★	★★	★★	★★★	★★★	★★
직관적인 사용성	★★★★	★★★★	★★★	★★	★★★	★★★	★
쉬운 사용법	★★	★	★★★★★	★★	★★★★★	★★★★	★★
쉬운 국제화	★★★★★	★★★★★	★★	★★	★★★★	★★★★	★★★
여러 명이 사용 시 위생도	★	★	★★★★★	★★★★★	★	★	★★★★★

★가 가장 낮고 ★★★★★가 가장 높은 평점입니다. 약어는 부록 A를 참조해 주세요.



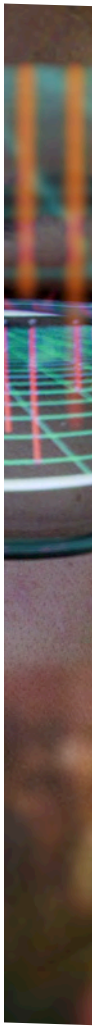
그래픽

대부분의 SoC에 GPU가 내장되어 있으므로 여러분이 직접 하드웨어를 설계하지 않는 한 그래픽 성능은 일반적으로 그래픽 프레임워크와 디스플레이 파이프라인의 품질에 달려 있습니다. 이 경우 기존 OS와 드라이버를 충실히 지원하는 CPU와 GPU의 조합을 선택하도록 주의해야 합니다. 직접 그래픽 드라이버를 만드는 것은 전혀 권장되지 않으며, 한다고 해도 불가능에 가깝습니다.

각각의 애플리케이션은 서로 다른 방식으로 GPU를 사용합니다. 따라서 흔히 보는 벤치마크는 테스트 결과는 특정 성능 지표에 대해서만 테스트한 결과라고 볼 수 있습니다. 특별한 테스트 결과가 없는 경우 우리가 선호하는 그래픽 API 표준은 Vulkan, DirectX, OpenGL, OpenGL ES 순입니다.

프레임 속도, GPU 및 CPU 부하, 배터리 수명, 온도 측면에서 그래픽 API 표준 간에는 눈에 띄는 성능 차이가 있을 수 있습니다. GUI 프레임워크가 렌더링에 사용할 API 표준을 선택할 수 있다면 가장 좋습니다. 이를 통해 GUI 코드를 그래픽 표준과 독립적으로 유지할 수 있습니다. 필요에 따라 특정 애플리케이션을 대상으로 지표를 측정하고 성능 향상을 위해 활용할 수 있습니다.

3D GPU가 가장 널리 쓰이지만, 2D로 표준 UI를 구현하는 경우 2D 또는 2.5D GPU를 통해 빠른 그래픽 처리를 위한 하드웨어 가속을 제공할 수 있습니다. 비트 블리트(BitBlit) 엔진, 레이어링(Layering)과 컴포지터(Compositor), 앤티 앨리어싱(Anti-aliasing), 폰트 글리프(Glyph) 렌더링 등을 통해 GPU는 애플리케이션의 렌더링 명령을 사용자가 볼 수 있는 픽셀로 변환하는 프로세스를 훨씬 빠르게 처리할 수 있습니다.



디스플레이

돋보이는 디스플레이로 고객의 시선을 사로잡고 싶으신가요? 그렇다면 아래 소개하는 다양한 디스플레이를 비교해보세요.

	TFT LCD	PMOLED	AMOLED	QLED	E-페이퍼
설명	박막 트랜지스터 액정 디스플레이	수동형 유기 발광 다이오드	능동형 유기 발광 다이오드	양자 발광 다이오드	전자 종이
밝은 빛 아래서의 가독성	★★★	★★★★	★★★★	★★★★	★★★★★
어둠 속에서 가독성	★★★★	★★★★★	★★★★★	★★★★★	★
업데이트 속도	★★★★★	★★	★★★★★	★★★★★	★
색 재현	★★★★	★★★	★★★★★	★★★★★	★
명암 대비	★★★	★★★★★	★★★★★	★★★★★	★★★★
시야각	★★★	★★★★	★★★	★★★★	★★★★★
전력 소비	★	★★	★★★★	★★★	★★★★★
내구성	★★★★★	★★	★★★	★★★★	★★★★★
크기	1" – 100"	0.5" – 6"	1" – 18"	TV 크기	1" – 10"
가격	$$	$$$	$$$$$	?	$
특징	현재 디스플레이의 지배적인 기술. 대부분 환경에서 볼 수 있도록 백라이트에 의존함 (백라이트가 매우 밝으면 햇빛 아래서도 사용 가능함).	AMOLED보다 제조 비용이 저렴하지만 전력 소모가 높고 화면 리프레쉬가 느림. 높은 전류로 인해 성능 저하가 빨리 옴.	RGB 구성 요소가 고르지 않게 저하되고 번인(burn-in)이 발생할 수 있지만 뛰어난 색 재현율과 가독성을 보여줌.	뛰어난 색 안정성을 제공함. 삼성이 계획하고 있지만, 아직 임베디드 애플리케이션에서 사용되고 있지 않음.	일반적으로 단색이지만 일부 색상 변형이 가능합니다. 이미지가 변경될 때만 전력을 소비함.

★가 가장 낮고 ★★★★★가 가장 높은 평점입니다. 약어는 부록 A를 참조해 주세요

정리하기

본 가이드가 여러분의 임베디드 및 IoT 장치에 선정 과정에서 유용하게 사용되기를 바랍니다. 기본적인 내용을 최대한 많이 다루려 노력했지만, 하루가 다르게 새로운 제품이 출시되고 있으며 하나의 문서에 이 모든 선택 사항을 담는다는 것은 불가능합니다. 당사의 컨설턴트와 엔지니어는 자동차, 항공 전자 시스템, 의료, 산업, 가전제품, 모바일 앱, 게임을 비롯한 많은 분야에서 기업이 성공적인 시스템을 구축하도록 협업해 왔습니다. 프로젝트에 무엇이 필요한지 고민 중이시라면 기꺼이 도와드릴 준비가 되어있습니다.

Qt EXPERIENCE

본 가이드 문서의 목적은 특정 컴포넌트를 추천하는 것이 아닌 최대한 공정하게 정보를 제공하는 것입니다. 실제 Qt 소프트웨어 제품군 개발에는 여러 회사의 제품 기획을 도우면서 배우고 경험한 수많은 노하우가 반영되어 있습니다. 이에 대해 궁금하다면 아래 링크에서 자세한 정보를 찾으실 수 있습니다.

Qt 다운로드 | **시작하기** | **제품 개요** | **리소스 센터** | **블로그**

부록 A – 줄임말과 약어

2D – 2차원
3D – 3차원
4K – 4K 해상도
ABI – 애플리케이션 바이너리 인터페이스 (Application binary interface)
ADC – 아날로그 디지털 컨버터 (Analog digital converter)
AI – 인공 지능 (Artificial intelligence)
AMOLED – 능동형 유기 발광 다이오드 (Active-matrix organic light-emitting diode)
API – 애플리케이션 프로그래밍 인터페이스 (Application programming interface)
APR – Apache 포터블 런타임 (Apache portable runtime)
BT – 블루투스
CAN – 컨트롤러 영역 네트워크 (Controller area network)
CI – 지속적인 통합 (Continuous integration)
CLA – 자리올림수 예측 가산기 (Carry-lookahead adder)
CPU – 중앙 처리 장치 (Central processing unit)
CSI – 카메라 직렬 인터페이스 (Camera serial interface)
CSS – Cascading style sheets
CUDA – 컴퓨팅 통합 디바이스 아키텍처 (Compute unified device architecture – Nvidia)
DAC – 디지털 아날로그 컨버터 (Digital analog converter)
DDR – Double data rate
DDR2 – Double data rate 2
DDR3L – Double data rate 3 low voltage
DDR4 – Double data rate 4
DIMM – 듀얼 인라인 메모리 모듈 (Dual in-line memory module)
DP++ – 듀얼 모드 디스플레이 포트 (DisplayPort dual-mode)
DSP – 디지털 신호 프로세서 (Digital signal processor)
E-페이퍼 – 전자 종이 (Electronic paper)
EAVB – 이더넷 오디오 비쥬얼 브리징 (Ethernet audio visual bridging)
ECC – 오류 수정 코드 (Error correcting code)
eMMC – 임베디드 멀티미디어 카드 (eMMC)
Eth – 이더넷 (Ethernet)
Eval – 평가 (Evaluation)
FFT – 고속 푸리에 변환 (Fast Fourier transform)
FPU – 부동소수점 처리장치 (Floating-point processing unit)
GB – 기가바이트 (10억 바이트)
GDP – 국내 총생산 (Gross domestic product)
GHz – 기가헤르츠 (Gigahertz)
GigE – 기가바이트 이더넷 (Gigabyte Ethernet)
GPU – 그래픽 처리 장치 (Graphics processing unit)
GUI – 그래픽 사용자 인터페이스 (Graphical user interface)
HDMI – 고화질 멀티미디어 인터페이스 (High-definition multimedia interface)
HTML5 – 하이퍼텍스트 마크업 언어 5 (Hypertext markup language 5)
IDE – 통합 개발 환경 (Integrated development environment)
LCD – 액정 디스플레이 (Liquid crystal display)
LED – 발광 다이오드 (Light emitting diode)
LPDDR4 – Low-power double data rate 4
LVDS – 저전압 차동 신호 (Low-voltage differential signaling)
LVTTL – 저전압 트랜지스터-트랜지스터 로직 (Low-voltage transistor-transistor logic)
MB – 메가바이트 (백만 바이트)

MCU – 마이크로 컨트롤러 (Microcontroller unit)
MHz – 메가헤르츠
Mic – 마이크 (Microphone)
microSD – 마이크로 SD (Micro secure digital)
MiniPCIe – 미니 주변 장치 구성 요소 상호 연결 익스프레스 (Mini peripheral component interconnect express)
MIPI – 모바일 산업 프로세서 인터페이스 (Mobile industry processor interface)
MMU – 메모리 관리 장치 (Memory management unit)
NUC – 차세대 컴퓨팅 장치 (Next unit of computing – Intel)
OpenGL – 개방형 그래픽 라이브러리 (Open graphics library)
OpenGL ES – 개방형 그래픽 라이브러리 임베디드 시스템 (Open graphics library embedded systems)
OS – 운영 체제 (Operating system)
OTA – 오버-디-에어 (Over the air)
PCIe – 주변 장치 구성 요소 상호 연결 익스프레스 (Peripheral component interconnect express)
PMIC – 전력 관리 집적 회로 (Power management integrated circuit)
PMOLED – 수동형 유기 발광 다이오드 (Passive-matrix organic light-emitting diode)
POSIX – 휴대용 운영 체제 인터페이스 (Portable operating system interface)
PRU – 프로그래머블 실시간 유닛 (Programmable real-time unit)
PWM – 펄스 폭 변조기 (Pulse width modulator)
QA – 품질 보증 (Quality assurance)
QLED – 양자 발광 다이오드 (Quantum light emitting diode)
QML – Qt 모델링 언어 (Qt modeling language)
RAM – 랜덤 액세스 메모리 (Random access memory)
RGB – 빨간색 녹색 파란색 (Red green blue)
RTOS – 실시간 운영 체제 (Real-time operating system)
SATA – 직렬 ATA (Serial AT attachment)
SBC – 단일 보드 컴퓨터 (Single board computer)
SD – Secure digital
SDRAM – 동기식 동적 랜덤 액세스 메모리 (Synchronous dynamic random-access memory)
SoC – 시스템 온 칩 (System on a chip)
SPI – 직렬 주변기기 인터페이스 (Serial peripheral interface)
TFT – 박막 트랜지스터 (Thin film transistor)
TI – 텍사스 인스트루먼트 (Texas Instruments)
TMS – TMS320 series DSP (TI)
TV – 텔레비전 (Television)
UI – 사용자 인터페이스 (User interface)
USB – 범용 직렬 버스 (Universal serial bus)
UWP – 유니버설 윈도우 플랫폼 (Universal Windows platform – Microsoft)
UX – 사용자 경험 (User experience)
V90 – SD 연합 비디오 스피드 클래스 90 (SD association video speed class 90 – 90MB/sec)
WebGL – 웹 그래픽 라이브러리 (Web graphics library)
WiFi – IEEE 802.11 무선 프로토콜 (Wireless protocol)
WiFi ac – IEEE 802.11ac-2013 또는 WiFi 5
WYSIWYG – 위즈윅 (What you see is what you get)
µHDMI – 마이크로 고화질 멀티미디어 인터페이스 (Micro high-definition multimedia interface)

ABOUT THE Qt Company

Qt Company는 오픈 거버넌스 모델에 따라 Qt 프로젝트와 함께 제품 개발, 상용 및 오픈소스 라이센스를 포함, Qt와 관련된 모든 활동을 담당하고 있습니다. 라이센싱, 기술 지원 및 서비스 제공과 함께, Qt 프로젝트가 적시에, 예산 내에서, 경쟁 우위를 확보하며 완료될 수 있도록 개발자와 긴밀히 협력하고 있습니다.

Qt Company의 목표는 데스크톱과 임베디드 및 모바일 개발자와 회사에 가장 강력한 크로스 플랫폼 UI와 애플리케이션 프레임워크를 제공하는 것입니다.

www.qt.io

The Qt Company