> **Important**
>
> 펌웨어를 수없이 양산하는 일을 멈추고, 코드에게 유효 수명을 길게 늘릴 수 있는 기회를 주어라

앱-티튜드 테스트

- 임베디드 소프트웨어가 왜 많이 펌웨어로 변하는가?
  - 동작하게 만드는 것과 빠르게 만들어라는 목표에 집착하기 때문
- 앱티듀드 테스트
  - 앱이 동작하도록 만드는 것
  - 프로그래머가 오직 앱이 동작하도록 만드는 일만 신경쓴다면 자신의 제품과 고용주에게 몹쓸 짓을 하는 것이다

타깃-하드웨어 병목현상(target-hardware bottleneck)

- 임베디드 개발자들은 임베디드가 아니었다면 다루지 않아도 될 특수한 관심사를 많이 가지고 있다
  - 제한된 메모리 공간, 실시간성 제약과 처리완료 시간, 제한된 입출력, 특이한 사용자 인터페이스, 여러 센서와 실제 세상과의 상호 작용 등
- 거의 모든 경우 하드웨어, 소프트웨어, 펌웨어가 동시에 만들어진다

클린 임베디드 아키텍처는 테스트하기 쉬운 임베디드 아키텍처다

계층

- 하드웨어는 시스템의 나머지(소프트웨어, 펌웨어) 부분으로부터 반드시 분리되어야 한다
- 소프트웨어와 펌웨어가 서로 섞이는 일은 anti-pattern
  - 이 안티 패턴을 보이는 코드는 변화에 저항하게 만들며, 변경하는 일 자체가 위험을 수반해, 때로는 의도치 않은 결과를 가져온다

하드웨어는 세부사항이다

- 하드웨어 추상화 계층(Hardware Abstraction Layer)
  - 소프트웨어와 펌웨어 사이의 경계
  - HAL의 API는 소프트웨어의 필요에 맞게 만들어져야 한다

HAL 사용자에게 하드웨어 세부사항을 드러내지 말라

- 클린 임베디드 아키텍처로 설계된 소프트웨어는 타깃 하드웨어에 관계없이 테스트가 가능하다

- HAL을 제대로 만들었다면, HAL은 타깃에 상관없이 테스트할 수 있는 경계층 또는 일련의 대체 지점을 제공한다

프로세서는 세부사항이다

- 클린 임베디드 아키텍처라면 장치 접근 레지스터를 직접 사용하는 코드는 소소의, 순전히 펌웨어로만 한정시켜야 한다
- 프로세서 추상화 계층(Processor Abstraction Layer)

운영체제는 세부사항이다

- OS는 소프트웨어를 펌웨어로부터 분리하는 계층
- OS를 직접 사용하면 문제가 된다
- 운영체제 추상화 계층(Operating System Abstraction Layer)
  - 운영체제와 소프트웨어를 격리 시키는 것
- 소프트웨어가 OS에 직접 의존하는 대신 OSAL 에 의존하면, 이식 작업의 대부분은 기존 OSAL과 호환되도록 새로운 OSAL을 작성하는 데 시간을 소요
- OSAL은 테스트 지점을 만드는 데 도움이 되고, 소프트웨어 계층의 귀중한 애플리케이션 코드를 타깃이나 OS에 관계없이 테스트할 수 있게 된다

인터페이스를 통하고 대체 가능성을 높이는 방향으로 프로그래밍하라

- 관심사를 분리시키고, 인터페이스를 활용하며, 대체 가능성을 높이는 방향으로 프로그래밍하도록 유도해야한다
- 계층형 아키텍처
  - 인터페이스를 통한 프로그래밍
  - 모듈들이 서로 인터페이스를 통해 상호작용한다면 특정 서비스 제공자를 다른 제공자로 대체할 수 있다
  - 인터페이스 정의는 헤더 파일에 해야한다

DRY 원칙: 조건부 컴파일 지시자를 반복하지 말라

-