JAK ZWIĘKSZYĆ JAKOŚĆ KODU W PROJEKTACH EMBEDDED?

DARMOWE NARZĘDZIA

ucgosu.pl - Programowanie i Robotyka

FORMATOWANIE KODU

Jednolite formatowanie kodu w całym projekcie zwiększa czytelność i ułatwia znajdowanie błędów. Dlatego projekt powinien posiadać Code Style Guide definiujący takie kwestie jak:

- taby i spacje
- nawiasy
- nazwy symboli
- długości linii

Dobre narzędzie do formatowania powinno umożliwiać konfigurację różnych stylów, wskazywać odstępstwa od stylu i umożliwiać automatyczne poprawianie tam gdzie ma to sens.

Polecane narzędzia:

- clang-format
- astyle
- Uncrustify

ANALIZA STATYCZNA

Kompilatory C i C++ zakładają, że wiemy co robimy i przepuszczają dużo podejrzanego kodu. Narzędzia do statycznej analizy wskazują taki kod, żebyśmy mogli sami ocenić, czy nie zawiera on błędów. Pozwalają wykryć problemy takie jak:

- wyjście poza zakres tablicy
- używanie niezainicjalizowanych zmiennych
- brak returna w funkcji
- błędne użycie funkcji biblioteki standardowej
- dzielenie przez zero

Różne toole specjalizują się w różnych rodzajach błędów. Poza tym jedne skupiają się na zgłoszeniu wszystkich potencjalnych zagrożeń i pozwalają sobie na więcej false-positive, inne z kolei pokazują tylko te których są pewne. Dlatego zawsze warto korzystać z kilku programów uzupełniających się nawzajem.

Polecane narzędzia:

- clang-tidy
- Cppcheck
- cpplint
- OClint
- flint++

METRYKI

Metryki to proste wartości liczbowe przekazujące informacje o jakości kodu. Jedną z popularnych metryk jest złożoność cyklomatyczna (Cyclomatic Complexity) oznaczająca ilość możliwych ścieżek wykonywania pojedynczej funkcji. Im większa złożoność, tym funkcja trudniejsza w utrzymaniu. Inne wykorzystywane metryki to ilość linii kodu w funkcji - wartość maksymalna nie powinna przekraczać pewnego progu. Podobnie z ilością linii w pliku. Dzięki informacji o ilości linii kodu w całym projekcie możemy mieć pojęcie o jego całkowitej złożoności.

Polecane narzędzia:

- Lizard
- CMetrics
- cqmetrics

UNIT TESTY

Test Driven Development to technika, w której przed napisaniem kodu produkcyjnego najpierw piszemy do niego unit test. Poza zmniejszeniem ilości błędów zaletami TDD są lepiej sformułowane wymagania, udokumentowane zachowanie w szczególnych przypadkach, czy lepszy design aplikacji. Stosowanie unit testów w systemach embedded ma jeszcze jedną ogromną zaletę - pozwala rozwijać większość aplikacji bez uruchamiania jej na docelowym hardware. W ten sposób możemy oszczędzić ogromną ilość czasu. Więcej materiałów o TDD dla systemów embedded znajdziesz na mojej stronie.

Frameworki C:

- Unity
- greatest

Frameworki C++:

- CppUTest
- Catch2
- googletest

MOCKI

Mocki to zastępcze implementacje modułów na potrzeby testów. Pozwalają wymusić zadane wartości wyjściowe i sprawdzić podane do nich argumenty wejściowe. Pozwalają dzięki temu sprawdzić w unit testach zależności od sprzętu, czasu i innych trudnych do przetestowania zależności.

Frameworki C:

- CMock
- FFF (Fake Function Framework)

Frameworki C++:

- CppUMock
- Trompeloeil
- googlemock

ANALIZA DYNAMICZNA

Analiza dynamiczna to analiza uruchomionego programu. Za pomocą narzędzi tego typu możemy znaleźć błędy takie jak:

- przekroczenie zakresu tablicy
- brak zwolnienia pamięci
- podwójne zwolnienie pamięci
- wiszące wskaźniki
- błędna kolejność alokacji/dealokacji

Poza szukaniem błędów narzędzia tego typu potrafią również wyznaczyć pokrycie kodu testami, czy wykonać profilowanie. Niestety nie ma dobrych narzędzi tego typu do uruchamiania na mikrokontrolerach. Dlatego aby skorzystać z ich dobrodziejstw musimy mieć działające unit testy na maszynie developerskiej, które poddajemy analizie dynamicznej.

Polecane narzędzia:

- Valgrind
- Address Sanitizer (asan)
- gcov
- gprof