Návrh způsobu testování

1. Testované části aplikace

1.1. Grafické rozhraní

Důležitým aspektem testování je grafické rozhraní, u kterého je potřeba zajistit správnost a přehlednost zobrazovaných dat, ošetřit nesprávné uživatelské vstupy a zajistit co největší absenci uživatelsky nepříjemných prvků, jako jsou vyskakovací okna, zadávání číselných hodnot textovým polem atp.

1.2. Emulace

Dalším důležitým aspektem testování je správnost chování emulátoru. Je nutné porovnat chování skutečného mikrokontroléru s chováním námi vytvořené emulace.

2. Manuální testování

Prioritním způsobem testování aplikace bude ruční testování, tedy obyčejné testování uživatelem, který se bude snažit aplikaci rozbít i použít správně. Testování bude během vývoje probíhat spíše individuálně a v malém měřítku. Intenzita testování se bude zvyšovat s blížícím se termínem nasazení aplikace. Veškeré nalezené chyby budou reportovány na projektovém GitLabu.

2.1. Testy grafického rozhraní

Scénáře manuálních testů grafického rozhraní nebudou předepsané. Pro naše potřeby budou stačit dva typy testů:

- Běžné ruční testy, převážně pro otestování *Happy path* scénáře a některých scénářů nesprávného použití.
- *Monkey testing*, kde se s rozhraním bude interagovat dostatečně náhodně, aby se otestovala robustnost aplikace.

U grafického rozhraní je také potřeba hlídat konzistentní chování na různých platformách a systémech, jelikož se během vývoje ukazuje, že se některé prvky mohou zobrazit odlišně, což může vést i k nesprávnému zarovnání textu, špatné viditelnosti prvků atp.

2.2. Testy emulace

Pro otestování správnosti využijeme poskytnutou desku Sipeed Longan Nano v1.1, pro kterou bude vytvořen jednoduchý testovací program, jehož chování porovnáme s chováním emulátoru. Testovat se bude hlavně chování periferií, tedy GPIO a UART. Skutečné hodnoty v paměťové oblasti a v registrech nelze jednoduše zobrazit a bude nutné použít modul UART pro jejich čtení.

3. Automatizované testování

Následující text obsahuje návrh automatizovaných testů, které ale mají nízkou prioritu a jejich případné zhotovení by bylo nad rámec rozsahu projektu.

3.1. Testy grafického rozhraní

Pro automatizované testování Qt aplikací existuje framework Squish for Qt, který by z hlediska použitelnosti i rychlosti testování byl nejideálnější, nicméně se jedná o licencovaný produkt. Jako alternativu lze použít testovací framework Qt Test, který je přímou součástí Qt knihovny.

3.2. Unit testy

Pro unit testy lze opět použít již zmínění framework Qt Test, s případným použitím mockovacího frameworku gMock.