VŠB – Technická universita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Návrh souboru automatických testů software pro kapacitní detekci doteku prstu produktu Freescale Touch

Design of Automatic Software Test Set for Freescale Touch Product

2015 Jakub Tarábek

VŠB – Technická universita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Jakub Tarábek**

Studijní program: B2649 Elektrotechnika

Studijní obor: 2601T004 Měřicí a řídicí technika

Téma: Návrh souboru automatických testů software pro kapacitní detekci doteku prstu produktu Freescale Touch
Design of Automatic Software Test Set for Freescale Touch Product

Zásady pro vypracování:

Náplní práce je návrh a realizace souboru testů ověřující správnou funkčnost softwarových knihoven, které tvoří základ produktu Freescale Touch. Tento produkt spojuje výhody softwarové technologie kapacitního měření a různých detekčních algoritmů s hardwarovými možnostmi mikrokontrolérů firmy Freescale z řady Kinetis L. Výsledkem tohoto spojení je systém umožňující snadno vytvářet koncové aplikace, které poskytují ovládání dotykem prstu, ať už ve formě klávesnice nebo různých posuvných a otočných prvků. Soubor testů, který má být navržen, musí pokrývat všechny softwarové vrstvy a funkce poskytované v rámci Freescale Touch: od měřící vrstvy, přes filtrační a detekční vrstvu až po algoritmy vyhodnocující pohyb prstu po ovládacích prvcích.

Body zadání:
1. Seznámení se a analýza kapacitního měření v produktech Freescale Kinetis L.
2. Návrh testovacích vektorů a metod, jak testovací hodnoty aplikovat na vstup měřícího software.
3. Návrh sady testovacích vektorů a testovacích metod pro ověření správné funkce detekce doteku prstu na elektrodách, detekce pohybu prstu nad sadou elektrod – forma klávesnice, posuvný ovládač a otočný ovládač.
4. Návrh a realizace emulátoru emulujícího fyzickou vrstvu Freescale Touch s využitím FPGA karty PXI-7854R od společnosti National Instruments.
5. Ověření funkčnosti testů a shrnutí výsledků.

Seznam doporučené odborné literatury:

[1] Freescale Kinetis L – KL25 Sub-Family Reference Manual (KL25P80M48SF0RM). Rev. 3, September 2012.
[2] Freescale Touch Sensing Software – User Guide (TSSUG). Rev 6. August/2013.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Petr Bilík, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Tarábek Jakub

Prohlášení zástupce spolupracující právnické nebo fyzické osoby

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských/magisterských programech VŠB-TU Ostrava.

6.5.2015, Rožnov p. R.

Hamerský Václav, vedoucí testovacího týmu

FREESCALE Polovodiče
Česká republika s.r.o.
1. máje 1009
756 61 Rožnov pod Radhoštěm
DIČ: CZ 27116051

Abstrakt

Náplní práce je návrh a realizace souboru testů ověřující správnou funkčnost softwarových knihoven, které tvoří základ produktu Freescale Touch. Tento produkt spojuje výhody softwarové technologie kapacitního měření a různých detekčních algoritmů s hardwarovými možnostmi mikrokontrolérů firmy Freescale z řady Kinetis. Výsledkem tohoto spojení je systém umožňující snadno vytvářet koncové aplikace, které poskytují ovládání dotykem prstu, ať už ve formě klávesnice nebo různých posuvných a otočných prvků. Soubor testů, který má být navržen, musí pokrývat všechny softwarové vrstvy a funkce poskytované v rámci Freescale Touch.

Klíčová slova

Testování SW, Freescale Touch, validace, mikrokontrolér, testovací zpráva

Abstract

The scope of work is the design and implementation of the set of tests to verify the correct functioning of software libraries that form the basis of product Freescale Touch. This product combines the benefits of software technology capacitive measurement and detection algorithms with different hardware options microcontrollers from Freescale Kinetis Series. The result is a system that allows to easily create end-use applications that provide fingertip control, whether in the form of a keyboard or various sliding and revolving elements. A set of tests to be designed to cover all software layers and functions provided within Freescale Touch.

Key Words

SW Testing, Freescale Touch, Validation, Microcontroler, Test Report

Seznam použitých symbolů a zkratek

Keypad	Abstrakce klávesnice.
Slider	Abstrakce tlačítek s vlastností přejetí prstem s diskrétním přechodem.
Rotary	Abstrakce tlačítek s vlastností přejetí prstem v kruhu s diskrétním přechodem.
Analog Slider	Abstrakce tlačítek s vlastností přejetí prstem se spojitým přechodem.
Analog Rotary	Abstrakce tlačítek s vlastností přejetí prstem v kruhu se spojitým přechodem.
System	Hlavní struktura FT.
FT	Freescale Touch
MCU	MicroControler Unit. Mikrokontrolér.
TSI	Periferie MCU Touch Sensing Input
TSS	Touch Sensing software
KSDK	Kinetis Software Development Kit
GPIO	General Purpose Input Output
IIR	Filtr s nekonečnou impulzní odezvou
IDE	Integrated Development Enviroment. Vývojové prostředí.
NI	National Instruments
FPGA	Field Programmable Gate Array. Programovatelná hradlová pole.
BEART	Build Execute Analyze and Report Test.
UART	Universal Asynchronous Receiver and Transmitter. Komunikační periferie.
PXI	Platforma pro měření a automatizaci.

Seznam obrázků

Obsah

1. Úvod

Je známo nepřeberné množství tlačítek, jako ovládací prvek rozhraní mezi člověkem a strojem (HMI, Human Machine Interface), jednou z možností je kapacitní detekce doteku prstu. Firma Freescale nabízí řešení, které ulehčují práci při vývoji aplikací obsahující tlačítka na detekci doteku prstu v hardwarové i softwarové podobě. Hardwarovou podobou jsou myšleny čipy obsahující modul pro měření kapacity a softwarovou podobu ve formě aplikační knihovny, která z naměřených dat vyhodnocuje, zda se tlačítka někdo dotknul.

Obsahem práce je seznámení se základními termíny a způsoby validování softwaru a předmětu práce testování softwarové knihovny Freescale Touch pro vestavěná zařízení.

Tato práce dále představuje produkt Freescale Touch (FT). Freescale Touch je abstrakce kapacitního rozpoznávání doteku prstu pro čipy firmy Freescale. Použitím této knihovny se usnadňuje práce při vývoji zařízení obsahující kapacitní „tlačítka“. Ve skutečnosti je FT zdrojový kód popsaný v jazyce *C* se složitou strukturou a širokou funkcionalitou.

Podle dokumentace knihovny FT a požadavků na testování je popsán automatizovaný testovací systém, který by měl být schopen otestovat softwarovou knihovnu a podpořit její robustnost během vývojového cyklu. Finální podobou diplomové práce by měl vzniknout takový systém, který na jeden povel komplexně otestuje možnosti a funkce FT. Takový systém, který je snadno rozšiřitelný a spolehlivý.

2. Formulace Problému

Předmět této práce se stalo zadání úlohy firmy Freescale. Téma bylo vybráno z důvodu sloučení pracovního a studijního času a prostředků. Cíl práce je kompletní splnění zadání. Finální stav je kompletní testovací systém pro validování softwarové knihovny FT.

Firma Freescale je americký výrobce polovodičových součástek. Hlavní činností společnosti Freescale je výroba součástek pro automobilový průmysl, mikrokontrolérů a mikromechanických senzorů. Jeden z produktů této firmy je snímání doteku prstu elektrody, reprezentující tlačítko, pomocí naměřené kapacity. S těmito tlačítky se lze setkat u různých spotřebičů, dálkových ovladačů, tlačítek displejů, mobilních telefonů jako ovládací prvky.

Obrázek 1 Aplikace kapacitních tlačítek na spotřebiči (1)

Hardwarovým řešení jsou mikrokontroléry a vývojové prostředky, které Freescale poskytuje a softwarovými prostředky je software Freescale Touch, který urychlí vývoj aplikací s kapacitními tlačítky, viz obrázek 2.

Obrázek 2 Vývojový kit s kapacitními tlačítky ve tvaru klávesnice (2)

Knihovna FT lze také označit jako API (Application Programming Interface) nebo SW. Všechna tvrzení jsou pravdivá a znamenají to samé.

Diplomovou prací je validování kapacitní detekce doteku prstu. Smyslem práce je pochopení funkčnosti knihovny FT, dále analýza problému z pohledu testování a následný popis. Cílem je vytvořit automatizovaný soubor testů, které budou schopny odhalit chyby nebo nedostatky softwaru FT. Od takového testovacího systému očekáváme komplexní ohodnocení kvality softwaru.

3. Testování

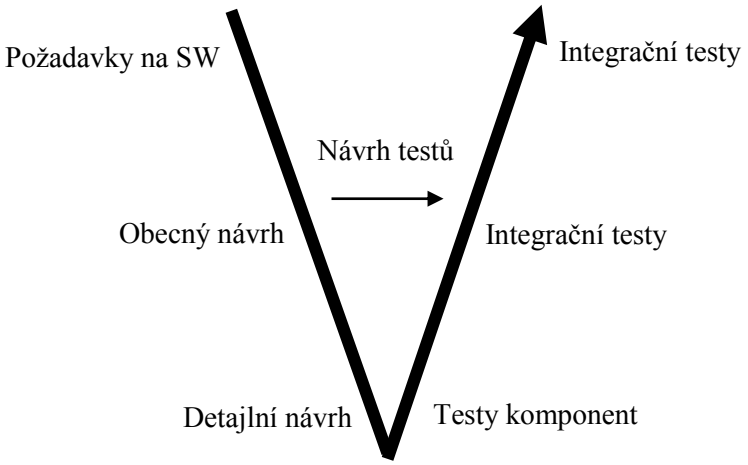
Testování a diagnostika softwaru nabývají s rostoucím počtem počítačových aplikací a intenzivní automatizací technických procesů stále většího významu. Velký podíl vyvíjených systémů tvoří zařízení používaná v kritických místech, ať již jde např. o výrobní linky nebo zařízení pro podporu životně důležitých funkcí lidského organismu. Selhání aplikace může v těchto případech způsobit velké materiální škody, popř. újmu na zdraví.

Zároveň neustále roste složitost i rozsah počítačových programů. Při jejich vývoji již není možné spoléhat na vývojové metody typu pokus-omyl a na ty pracovní postupy, kdy úspěch záleží na individuálním „hrdinství“ a entuziasmu každého pracovníka. Technologie vývoje softwaru směřuje ke zvyšování požadavků na výslednou spolehlivost produktu a na dodržení kvality v průběhu vývoje a testování aplikace. Tyto požadavky a snaha zmenšit možná rizika vedou k růstu podílu testování na celkovém objemu prací při vývoji softwaru. Příkladem je udávaný poměr počtu testovacích pracovníků k počtu vývojářů, který se v současné době u větších projektů pohybuje okolo 1 : 3 až 1 : 4 (společnost Microsoft uvádí poměr 1 : 1). (3)

3.1. Testování v procesu vývoje softwaru

Nejjednodušším způsobem testování softwaru je jeho opakované spouštění a zadávání různých vstupních dat s cílem prověřit co největší počet jejich možných kombinací a uživatelských situací. Tento přístup je však při vývoji rozsáhlejších aplikací zcela nedostačující. Pojem testování je třeba rozšířit a chápat spíše jako specifickou skupinu činností vztažených k jednotlivým etapám procesu vývoje softwaru.

Standardní proces vývoje softwaru postupuje nejčastěji podle tzv. V-modelu, znázorněného na obrázku 3. Základními aktivitami realizovanými v procesu testování softwaru jsou revize a inspekce požadavků a specifikací, identifikace testovacích požadavků a návrh, provedení a vyhodnocení testů. Stručně je popíšeme. (3)



Obrázek 3 V-model

3.2. Revize a inspekce požadavků a specifikací softwaru

Veškerá dokumentace, která popisuje vyvíjený systém, podléhá revizím. Primárním cílem revize je odhalit chyby v návrhu softwaru a doplnit případné nejasnosti. Zároveň je nutné již v počátečních fázích vývoje vnést do návrhu požadavky na budoucí možnost testovat software a rovněž zkontrolovat, zda existující požadavky je možné podrobit testu. Formulace typu např. „hodnota parametru je pravidelně aktualizována“ je příliš vágní a pod pojmem „pravidelná aktualizace“ se může skrývat velké množství možných definic.

Při návrhu je také nutné uvažovat o budoucí automatizaci testování a navrhnout příslušná rozhraní pro testovací nástroje. Přizpůsobení softwaru pro testování je z hlediska vývoje časově náročná práce, se kterou je nutné počítat při plánování celkové doby trvání projektu (3).

3.3. Identifikace testovacích požadavků

Proces identifikace testovacích požadavků spočívá ve výběru všech požadavků, které souvisejí s předmětem testu. Jde především o výběr podmnožin specifikovaných systémových a softwarových požadavků a o definování nových požadavků na základě analýzy dokumentace návrhu (3).

3.4. Návrh testů

Návrh testů zahrnuje specifikaci testovacího zařízení a definici jednotlivých testovacích postupů. Navržené testovací postupy musí ověřit všechny identifikované testovací požadavky (3).

3.5. Provedení testů a regresní testování

Při uskutečňování testů je nutná především pečlivá dokumentace provedení každého jednotlivého kroku každého z postupů tak, aby bylo možné testy v případě potřeby přesně reprodukovat. Regresní testování představuje opakování všech testů, jejichž předmět byl upraven nebo změněn za účelem odstranění chyb detekovaných během předchozího testování. Právě vzhledem k časové náročnosti regresních testů, a to zejména v závěrečných etapách projektu, vyvstává nutnost automatizovat testování softwaru (3).

3.6. Vyhodnocení testů

Důležitou součástí testování je vyhodnocení testů. Testy se vyhodnocují na základě různých měření, která se vždy definují během plánovací fáze projektu. Často se využívají kombinace souhrnných a strukturovaných měření (např. podíl chyb na celkovém počtu vykonaných testů a podíl chyb v rámci jednotlivých komponent). Z hlediska zhodnocení úspěšnosti testů je důležité sledovat poměr chyb nalezených během uskutečňování navržených testů a chyb nalezených náhodnými neplánovanými testy (3).

3.7. Strategie testování

Testovací strategie (technika) je systematická metoda použitá k výběru nebo generování testu, který bude zahrnut do skupiny testů. Testovací strategií jsou definována pravidla, podle kterých je možné objektivně posoudit, zda daný test vyhovuje vybrané strategii. Strategii lze také automatizovat. Testovací strategie se v zásadě dělí na:

- funkční (black-box) testy: návrh testů vychází z definic požadavků na software a popisu jeho funkce; testy na základě této strategie teoreticky mohou být navrženy bez znalosti testovaného objektu, i když v praxi se využívá znalosti návrhu kódu;
- strukturální (glass-box, white-box) testy: jsou odvozeny přímo ze znalosti struktury testovaného objektu; strukturou je zde třeba rozumět architekturu softwaru a způsob jeho implementace;
- hybridní testy: kombinace funkčních a strukturálních testů (3).

3.8. Typy testů

Testy softwaru jsou rozděleny do čtyř základních etap:

Testy komponent, které bývají často zahrnovány do fáze vývoje, protože je uskutečňují přímo vývojoví pracovníci v rámci ladění kódu. Cílem těchto testů je ověřit funkční schopnosti izolovaných modulů programu na základě ověření všech funkcí vstupně-výstupního rozhraní.

Integrační testy, určené k ověřování správnosti spolupráce integrovaných modulů na základě dokumentace návrhu. Obvykle vyžadují simulaci činnosti nehotových modulů, popř. vytváření speciálních ovládačů pro ověřování komunikace mezi moduly. Jednotlivé moduly se při těchto testech považují za uzavřené systémy. Z hlediska modulů tedy jde o funkční testy, zatímco z hlediska celého programu jde spíše o testy hybridní.

Systémové testy, kterými se ověřuje činnost systému jako celku, tedy včetně hardwaru, se kterým je software distribuován. Program je testován jako tzv. černá skříňka. Typické jsou především zátěžové testy, testy správnosti hospodaření s operační pamětí a funkční testy při různých konfiguracích systému.

Kvalifikační testy, zajišťující kompletní ověření všech systémových funkcí garantovaných zákazníkovi. Slouží jako výstupní kontrola výsledného produktu (3).

3.9. Automatizace testů

Proces návrhu a provádění testů softwaru je většinou omezen dobou, která je na testování vyhrazena v rámci projektu. Opakovatelnost testů a preciznost jejich uskutečňování hrají také důležitou roli, ale z hlediska termínů dokončení projektu jsou určující zejména počet a časová náročnost testů. Například pro otestování běžné aplikace pro zdravotnictví je zapotřebí software podrobit dvěma až třem tisícům testů. Průměrná doba trvání jednoho testu při ručním zpracování je v rozsahu řádově minut až desítek minut. Ruční otestování celé aplikace zabere přibližně dva měsíce. Při vývoji aplikace po krocích je nutné testy celé aplikace zopakovat po každém vývojovém kroku, tedy změně kódu. Za těchto podmínek je nutné testování automatizovat.

Automatizací se rozumí použití vhodného testovacího nástroje, který je schopen ovládat testovanou aplikaci tak, jak by to při jejím testování dělal člověk. Nejčastěji se testy píší ve skriptovacím jazyku, podobném např. pomocí Matlab, Ruby nebo Python, které jsou navrženy pro potřeby testovacího nástroje.

Běžným typem funkčních testů jsou testy uskutečňované pomocí standardních rozhraní testované aplikace, nejčastěji pomocí grafického uživatelského rozhraní. Pro tento typ testů existuje velké množství komerčně dostupných nástrojů, jako je např. ATF (Softbridge), WinRunner (Mercury Interactive), Rational Robot (Rational) aj.

Pro méně standardní rozhraní je nutné vytvořit vlastní testovací nástroj, který musí být vyvíjen s testovanou aplikací. Součástí vývoje aplikace se tak stává také vývoj testovacího nástroje. Mnohdy jde o testovací aplikace velmi propracované, jež musí být navrhovány již v počátečních fázích návrhu testovaného softwaru (3).

Testování softwaru je popsána normou ISO/IEC 2911.

3.10. Norma ISO/IEC 29119

Norma je sérií pěti standardů. Definují mezinárodně schválené standardy, které mohou být použity pro jakýkoliv typ SW testování.

- ISO/IEC 29119-1: Concepts and Definitions
- ISO/IEC 29119-2: Test Processes
- ISO/IEC 29119-3: Test Documentation
- ISO/IEC 29119-4: Test techniques
- ISO/IEC 29119-5: Keyword Driven Testing

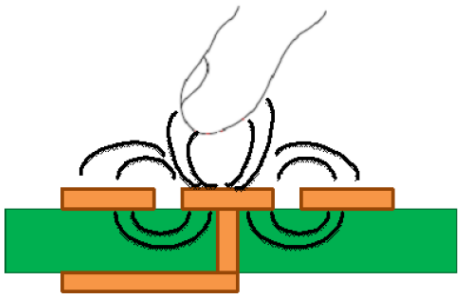
Tato norma shromáždila jiné známé normy (IEEE 829: Test Documentation, IEEE 1008: Unit Testing) pod jeden název.

4. Touch sensing

Hlavní téma se týká validování produktu, který lze použít výhradně u mikrokontrolérů. Mikrokontrolér (MCU) neboli jednočipový počítač, je integrovaný obvod mikroprocesor s podpůrnými periferiemi. Mikrokontroléry se vyznačují spolehlivostí a kompaktností. Jsou určeny pro jednoúčelové aplikace, jako je řízení nebo regulace. Mikrokontrolér si lze představit jako procesor, který je v osobním počítači, nicméně mikrokontroléry zpravidla nejsou tak výkonné a mají v sobě navíc periferie, jako jsou čítače, časovače, analogové převodníky, řadiče pro displeje i klávesnice dále komunikační periferie Ethernet, UART a další.

Kapacitní detekce dotyku prstu je náhradou za mechanické tlačítka nebo membránové tlačítka. Jsou využity v širokém rozsahu uplatnění od průmyslového vybavení po přenosná zařízení. Poskytují rozhraní mezi člověkem a strojem pro ovládání v podobě klasických telefonních tlačítek nebo mohou také sloužit jako ovládací prvky k regulaci hlasitosti nebo jasu.

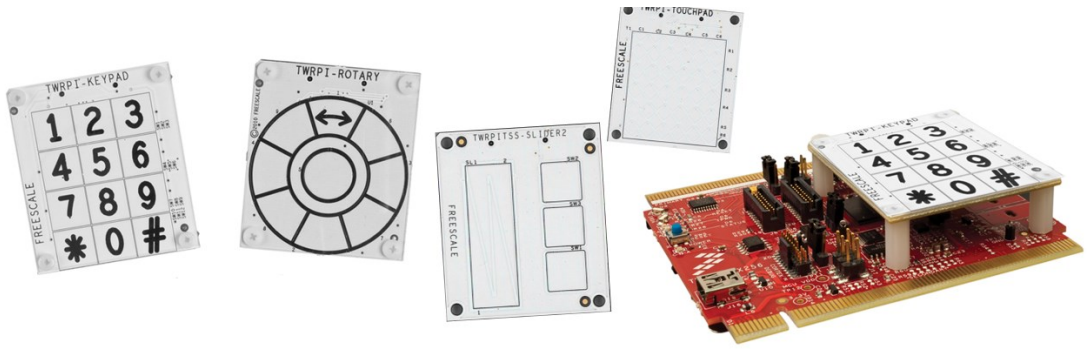
Oproti mechanickým nebo membránovým tlačítkům přináší kapacitní detekce dotyku několik výhod. Jak je již zřejmé touch sensing postrádá mechanické části, takže taková tlačítka jsou odolná vůči mechanickému opotřebení nebo korozi kontaktů. Navrhnuté zařízení mohou být využity ve venkovních podmínkách při dešti a vlhku. Velkou výhodou je i cena, protože tlačítko je fyzicky jen vodivá ploška, viz obrázek 4.



Obrázek 4 Dotykový panel (4)

Malou nevýhodou je, že oproti mechanickým tlačítkům uživateli chybí zpětná vazba, která dodá alespoň pocit, že tlačítko bylo stisknuto.

Na obrázku 5 jsou zobrazeny doplňky, které Freescale nabízí pro snadný vývoj a ladění uživatelských aplikací.

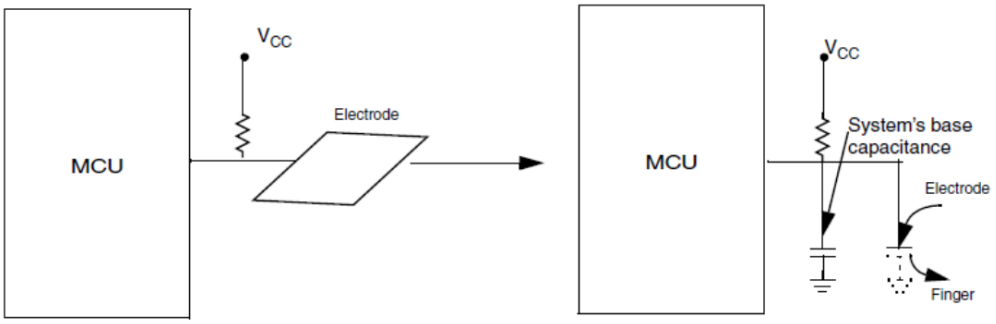


Obrázek 5 Vývojová deska s Kinetis čipem od Freescale s destičkami pro snadný vývoj Touch Sensing aplikaci (2)

4.1. Princip

Dotekem elektrody se v obvodech MCU zpracovávají signály, jejichž změny vyhodnocuje na dotek nebo uvolnění. V těchto obvodech se měří kapacita mezi elektrodou tlačítka a zemí. Pokud nastane dotek, změní se kapacita v obvodu (navýší). Fyzicky existují dva způsoby měření kapacity, GPIO metoda, která je využívá jednoduchý vstupně-výstupní pin s čítačem MCU a TSI metoda, využívající speciální periferii na MCU.

4.1.1 GPIO metoda



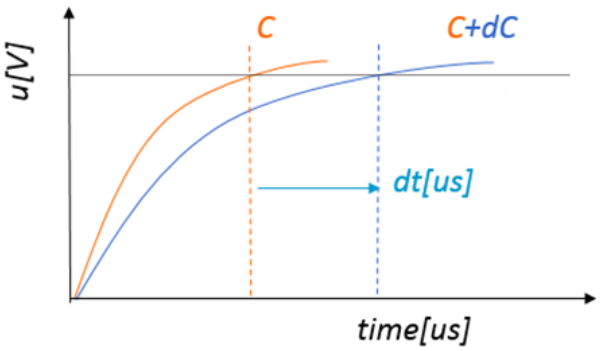
Obrázek 6 Zapojení elektrody a náhradní schéma elektrody (5)

Elektroda připojená k MCU se chová jako kondenzátor, externí pullup rezistor limituje proud k nabíjení elektrody. Na obrázku 6 je zapojení elektrody k pinu MCU a jeho náhradní schéma.

Časová konstanta RC obvodu je definovaná následující rovnicí:

$$\tau = RC \qquad (1)$$

Podle této rovnice je zřejmé, že pokud pullup rezistor je konstantní, tak s rostoucí časovou konstantou roste kapacita. MCU pomocí GPIO (General Purpose Input Output) a časovače měří dobu nabíjení kapacity elektrody. Obrázek 7 zobrazuje rozdíl v nabíjení kapacity samotné elektrody *C* a elektrody s dotekem *C+dC* (5).



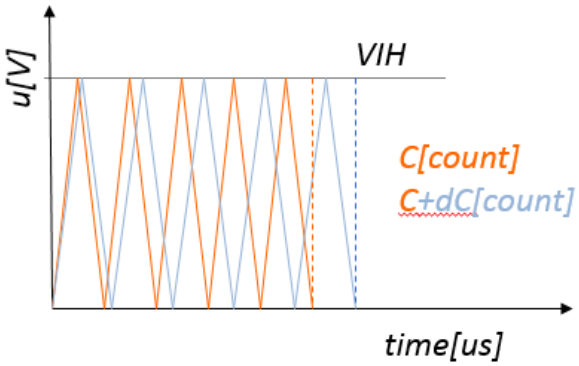
Obrázek 7 GPIO metoda (5)

4.1.2 TSI metoda

TSI metoda je vázaná k periferii TSI. Ne každý čip od Freescalu TSI modul obsahuje, je to speciální periferie obsažená v čipech Kinetis a Coldfire+. Dnes jsou podporovány pouze Kinetis čipy. Kromě mikrokontroléru je možné použít samostatný čip obsahující pouze modul TSI.

Principem této metody jsou dva oscilující signály, jeden je interní a druhý externí. Externí signál má menší frekvenci, to umožní spočítat kolik krát se signál s větší frekvencí opozdí oproti pomalejšímu (6).

Když dotek elektrody změní kapacitu, interní signál se nezmění, ale externí frekvence oscilace se zpomalí. Z tohoto důvodu je zaveden interní oscilátor.



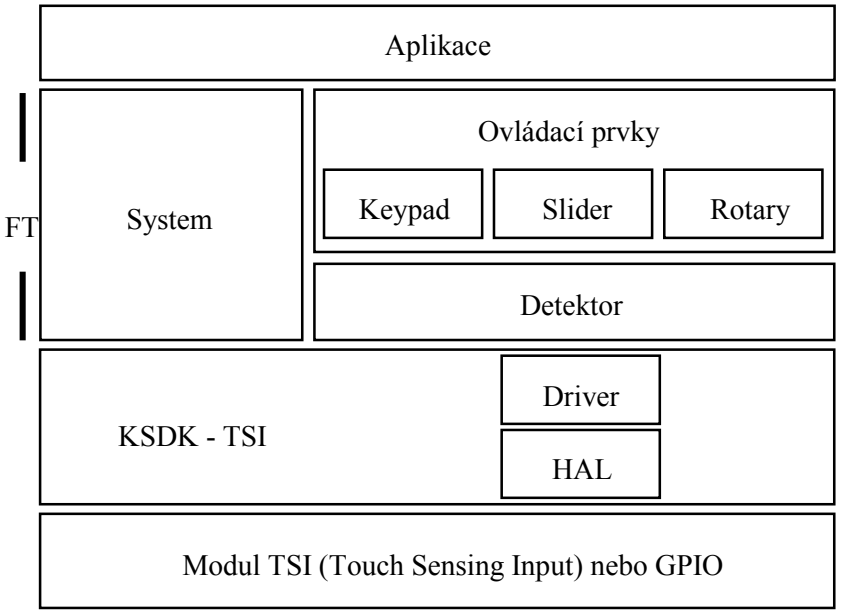
Obrázek 8 TSI metoda (6)

Na obrázku číslo 8 jsou zobrazeny oba oscilační signály, kde *dC* je rozdíl inkrementovaných hodnot interního a externího oscilátoru.

4.2. Freescale Touch

Freescale Touch promění jakýkoliv čip Kinetis na dotykový senzor. FT knihovna podporuje všechny Freescale čipy zakládané na ARM Cortex M4 a ARM Cortex M0 procesorech. Poskytuje běžné dekódovací struktury jako Keypad, Slider, Rotary, Snalog Slider, Analog Rotary. Freescale Touch využívá KSDK knihovnu.

Knihovna Freescale Touch je dodávaná jako soubor zdrojových kódů popsaných v programovacím jazyce *C* a je ke stáhnutí zdarma a uživatel ji může také zadarmo použít na MCU od firmy Freescale.



Obrázek 9 Knihovna FT

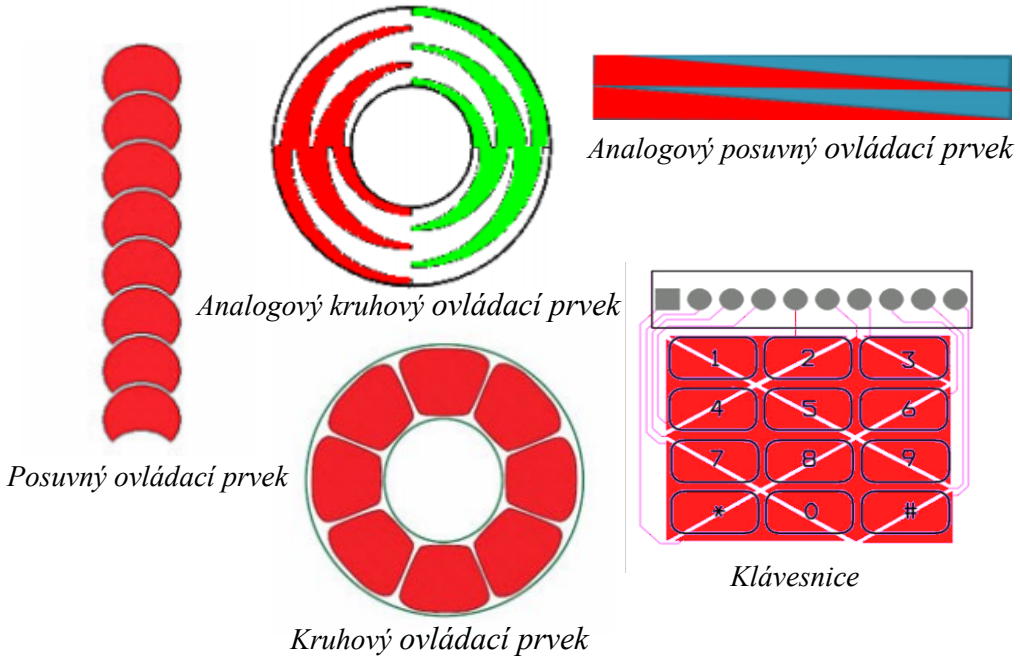
Na obrázku 9 je struktura aplikace s knihovnou FT. Jak je už známo naměřená data pocházejí z mikrokontroléru, buď z TSI modulu nebo z GPIO. Ovladače a hardwarová abstrakce poskytují základní obsluhu hardwarové části struktury. Další vrstvou je knihovna Freescale Touch, detektor pro rozpoznávání doteku, ovládací prvky a System FT. Celý FT zastřešuje Systém. Všechny části budou následně popsány. Více v (2).

Modul detektor zjišťuje, jaká elektroda byla stlačena, nebo uvolněna na základě naměřených hodnot. Modul obsahuje algoritmy pro identifikaci dotčených elektrod porovnáním Baseline hodnoty s kapacitní hodnotou.

Ovládací prvek (Control) je nejvyšší úrovní abstrakce v knihovně. V této vrstvě je informace o doteku nebo uvolnění interpretována specifickým nebo vyvoláním události.

4.3. Ovládací prvky

Ovládací prvky jsou nejvyšší úrovní abstrakce vyhodnocování doteku prstu. Obsahují metody pro registrování událostí, jako jsou například dotek, uvolnění, a pohyb, dále obsahují algoritmy pro detekci pozic dotčených elektrod a rozeznání správného doteku. Výčet podporovaných ovládacích prvků je na obrázku 10.



Obrázek 10 Ovládací prvky (6)

4.3.1 Klávesnice (Keypad)

V aplikaci lze využitím API zaznamenat dotek nebo uvolnění jednotlivých elektrod. Klávesnice poskytuje informace o indexu a neplatném doteku. Umožňuje generovat události, když se prst dotkne, uvolní nebo vyvolá pravidelné události při nepřerušeném doteku tlačítka.

Ovládací prvek klávesnice je schopný konfigurovat skupiny kláves, aby se chovali jako jedno tlačítko. Díky tomu je možné obsáhnout více tlačítek než je počet elektrod.

4.3.2 Posuvný ovládací prvek (Slider)

Posuvný ovládací prvek detekuje lineární pohyb prstem po elektrodách. Využívá diskrétních elektrod k vypočítání pozice v lineárním poli. Algoritmus určuje pozici dotčené elektrody a blízkých elektrod k určení polohy. Posuvný ovládač složený z N elektrod má 2N-1 pozic.

API Posuvného ovládače poskytuje informace o pozici, směru a neplatném doteku. Umožňuje generovat události, když se prst posune, dotkne nebo uvolní.

4.3.3 Kruhový ovládací prvek (Rotary)

Kruhový ovládací prvek detekuje pohyb prstu na diskrétních elektrodách a využívá je k vypočítání pozice v kruhovém poli. Algoritmus určuje pozici dotčené elektrody a blízkých elektrod k určení polohy. Kruhový ovládač složený z N elektrod má N-1 pozic.

API Posuvného ovládače poskytuje informace o pozici, směru a neplatném doteku. Umožňuje generovat události, když se prst posune, dotkne nebo uvolní.

4.3.4 Spojitý posuvný ovládací prvek (Aslider)

Spojitý posuvný ovládací prvek detekuje lineární pohyb prstem po elektrodách. Využívá pár elektrod s největší naměřenou kapacitou k vypočítání pozice v lineárním poli. Algoritmus určuje pozici podle poměru dvou nejvyšších naměřených kapacit. Pozice se přepočítá na rozsah zadaný uživatelem. Například 0-127.

API posuvného ovládacího prvku poskytuje informace o pozici, směru a neplatném doteku. Umožňuje generovat události, když se prst posune, dotkne nebo uvolní (2).

4.3.5 Spojitý kruhový ovládací prvek (Arotary)

Spojitý kruhový ovládací prvek detekuje lineární pohyb prstem po elektrodách. Využívá pár elektrod s největší naměřenou kapacitou k vypočítání pozice v lineárním poli. Algoritmus určuje pozici podle poměru dvou největších naměřených kapacit. Pozice se přepočítá na rozsah zadaný uživatelem.

API kruhového ovládacího prvku poskytuje informace o pozici, směru a neplatném doteku. Umožňuje generovat události, když se prst posune, dotkne nebo uvolní (2).

4.4. Moduly

Tato část API zabezpečuje shromažďování naměřených dat z různých hardwarových zdrojů, jako jsou GPIO piny nebo z periferie TSI. Modul GPIO sbírá data z libovolného General Purpose pinu. Modul TSI sbírá z fyzické elektrody naměřené hodnoty pomocí periferie TSI. Funkce pro čtení a údržbu periferie jsou poskytovány z KSDK. Tento modul také obsahuje jednoduché rozpoznávání doteku prstu pomocí šumu (2).

4.5. Elektrody

Elektrody jsou datové objekty, kde FT ukládá data o naměřených hodnotách a dalších pomocných proměnných. Dále se zde uchovávají i výsledky algoritmů a časové značky. Každá elektroda poskytuje informace o signálu. Časové značky jsou zde ukládány a poskytují informace o posledních změnách doteku. V elektrodě je definováno jaký algoritmus rozpoznávání doteku používá a poskytuje přístup k těmto detekčním algoritmům. To přináší výhodu, že každá elektroda nezávisle na sobě může používat jiné způsoby detekce doteku (2).

4.6. Klíčové rozpoznávání doteku

Algoritmy rozpoznávání doteku určují, kdy nastal dotek a kdy uvolnění na základě obdržených hodnot z vrstvy modulu příslušné elektrody.

4.6.1 AFID

Advanced Filtering and Integrating Detection. Je založen na dvou IIR filtrech s rozdílnou hloubkou a integrují rozdíly mezi dvěma signály vytvořené těmito filtry (2).

4.7. System

System je hlavním strukturou, která zapouzdřuje další klíčové objekty v systému Freescale Touch, jako jsou měřící moduly a ovládací prvky. V jedné aplikaci je může být pouze jedna struktura System. Funkcionalita Systemu v API je pokrýt inicializaci, globální časování a poskytnout jednoduchý přístup k detekčním modulům a dalším strukturám (2).

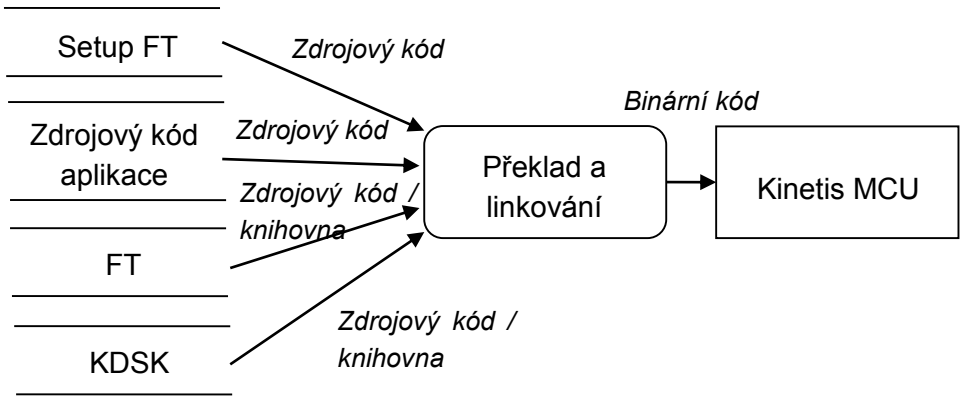
4.8. Kinetis Software Devolopment Kit

Kinetis Software Development Kit, dále jen KSDK, je rozšířená sada ovladačů, frameworku a příkladných aplikací navržené tak, aby vývoj vestavěných aplikací byl jednodušší a rychlejší na mikrokontrolérech Kinetis. S Processor Expert technologií, který je pro podporu čipů a softwaru, poskytuje nespočet možností použití. KSDK je bezplatný a obsahuje open-source licenci pro hardwarovou abstrakci i ovladačů periferií. Více na (7).

4.9. Příklad použití knihovny Freescale Touch

Na obrázku 11 je diagram popisující vývoj aplikace obsahující softwarovou knihovnu FT. Pro usnadnění je dobré si ulehčit práci vhodným vývojovým softwarem neboli IDE. Mezi tyto vývojová prostředí patří Kinetis Development Studio od firmy Freescale, který je ke stažení zdarma. S výhodou se dají použít komerční nástroje pro vývoj, jako jsou KEIL (8) od firmy ARM nebo IAR (9), které nabízí jednodušší prostředí pro vývoj uživatelských aplikací a se spolehlivějším překladem kódu.

Informace ohledně programování v jazyce *C* a vysvětlení pojmů jako jsou překlad, makra nebo linkování najdete zde (8).



Obrázek 11 Zpracování programu do MCU

K vytvoření programu, který má obsahovat knihovnu FT, potřebujeme mimo jiné KSDK kvůli podpoře MCU, který používáme a hlavně kvůli ovladačům periferií. KSDK můžeme použít jako zdrojový kód, ale i jako přeloženou knihovnu. FT a KSDK se společně s uživatelskou aplikací přeloží do strojově srozumitelné podoby a pomocí programátoru nahrají do čipu.

Co se týče aplikace knihovny FT, tak je nutné konfigurovat nastavení formou zdrojových kódů, ve kterých je strukturovaně popsána konfigurace FT. Hlavní strukturou knihovny FT je

System, který ukazuje na další podstruktury popisující vlastnosti ovládacích prvků (Keypad, Slider, Rotary), modulu (TSI, GPIO), detektoru (AFID) a elektrod.

- struktura System
  - struktury elektrod
  - struktury ovládacích prvků
  - struktury modulů
  - struktury detektorů

Popis činnosti běhu programu a využití nakonfigurované knihovny je popsáno v následující kapitole.

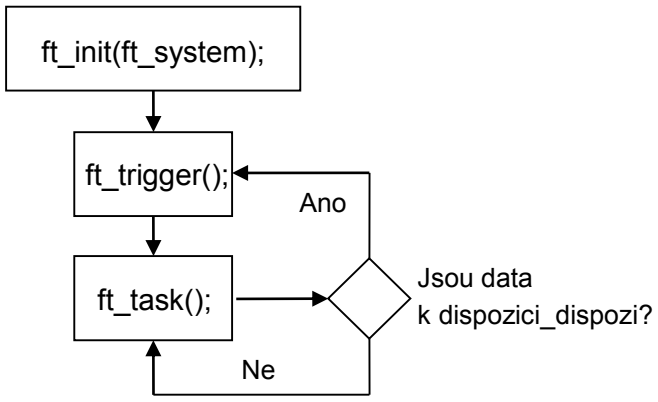
4.10. Životní cyklus programu s knihovnou Freescale Touch

Inicializace, Trigger a Task jsou základní procesy životního cyklu aplikace s FT. V jedné aplikaci na jednom čipu může být nadefinován pouze jeden System FT a ten je parametrem funkce inicializace.

V inicializační části, která je volána funkcí *ft_init,* je systém poskytovatelem konfigurace. *ft_init* kontroluje správnost nastavených konfigurací v systému. Dalším krokem inicializace je nastavení fyzického modulu čipu. Pokud jde o TSI modul je nastavení složité a doporučuji použít nastavení uvedené v příkladu přiloženém v balíčku FT.

Proces Trigger (funkce *ft_trigger*) spouští měření v modulu. Měl by být volán periodicky pomocí časovače. Uživatel při vývoji aplikace by měl myslet na prodlevu při měření kapacity. Jedná se o modifikaci čítače (TSI i GPIO metoda), čili by uživatel měl počítat s určitou rezervou nebo tento případ programově ošetřit.

Výpočty a vyhodnocování naměřených dat probíhá až v procesu pojmenovaném Task. Funkce *ft_task* na svém začátku kontroluje, zda-li má k dispozici naměřená data, poté vyhodnotí, na kterých elektrodách byl detekován dotek. Velkou předností tasku je vyvolání inicializovaných událostí. Událost může zavolat libovolnou uživatelskou rutinu.



Obrázek 12 Životní cyklus aplikace s FT

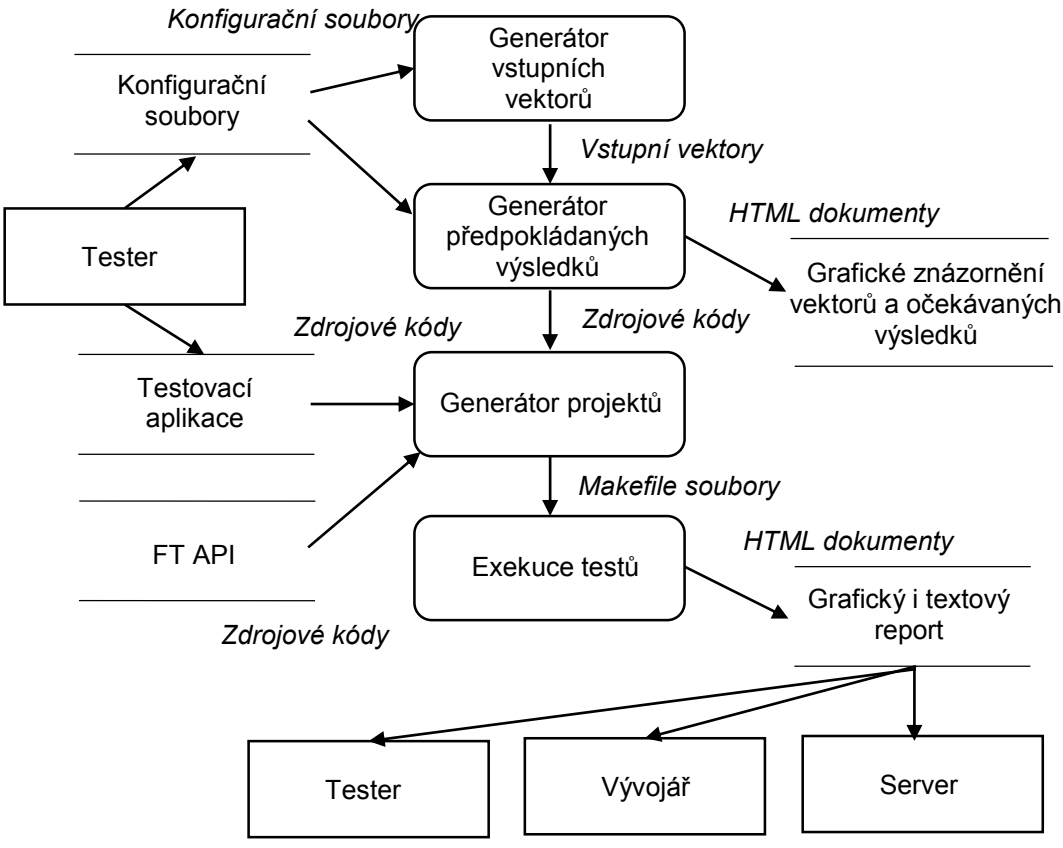
Přehled základních funkcí v životním cyklu je na obrázku 12. Samozřejmě v obsahu knihovny FT je mnoho dalších funkcí třeba pro inicializování událostí, zapínání a vypínání funkcionalit, ovládacích prvků nebo elektrod, vyčítání dat a stavů atd.

5. Testování softwaru Freescale Touch

Dotýkat se ručně elektrody a čekat na zpětnou vazbu, například rozsvícení LED diody, jako testovací rutinu považovat nemůžeme a vůči složitosti FT je to nevhodné. Musíme splnit požadavky na korektnost a přesnost, aby se mohlo prohlásit, že se v čase *t* od knihovní aplikace FT očekávala změna stavu a ten pak nastal. Pokud změna nenastala, je zapotřebí zjistit, jestli se změna stavu nevyvolala se zpožděním nebo vůbec.

V předchozí kapitole bylo základní seznámení s testovanou knihovnou a je zřejmé, že bude obsahovat mnoho nastavení. Z tohoto důvodu se bude vytvářet jedna testovací aplikace k jedné konfiguraci nastavení pro sadu testů. Nyní se text diplomové práce bude zabývat pouze jen testováním softwaru FT.

Na následujícím obrázku 13 je diagram datových toků, který popisuje testovací sekvenci a kompletně nastiňuje funkcionalitu testování FT. Obrázek je velmi důležitý pro pochopení testování knihovny FT.



Obrázek 13 Diagram datových toků testovacího systému

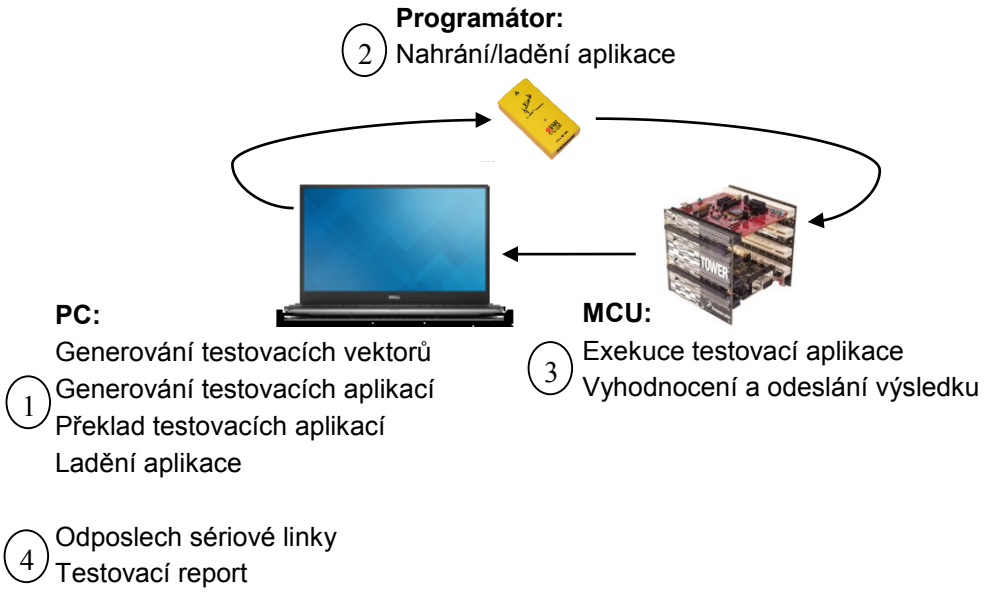
Vstupem testovacího systému jsou konfigurační soubory a testovací algoritmus ve formě zdrojového kódu. Pro jednu testovací aplikaci nebo jednu testovací sadu je zapotřebí jeden konfigurační soubor a samotný kód testovací aplikace. Podle konfiguračního souboru se vygenerují testovací vektory a vyhodnotí se předpokládané výsledky. Tato data jsou pak

uložena jako pole hodnot ve zdrojových souborech. Všechny zdrojové soubory jsou následně zkompilovány a nahrány do cílového čipu, kde se testy spustí a zvalidují. Výsledky testů jsou přijímány na PC po komunikační sběrnici a zpracovány do čitelné formy jako testovací zpráva neboli testovací report.

Bloky generátor vstupních vektorů a generátor předpokládaných výsledků z předchozího obrázku jsou implementovány programovacím jazykem Python (12). Tyto bloky jsou volány ze společného Python skriptu, který se dá spouštět z příkazové řádky. Dále pod těmito bloky jsou schovány i pomocné programy, které umožňují vygenerovat webovou aplikaci, která umí v grafech, textově nebo v tabulkách vizualizovat vygenerovaná data. Tento vizualizační program je naimplementován v jazyce Python a je využit i ke generování testovacího reportu.

V následujících kapitolách 6, 7, 8, 9, 10, 11 jsou jednotlivé části testovacího systému a samotný proces testování a vývoje podrobněji popsán. Bloky *generátor projektů* a *exekuce testů* z obrázku 13 jsou součásti testovacího frameworku vyvinutého ve Freescale, jsou využity pro usnadnění vývoje a automatizaci testů.

Na následujícím obrázku 14 je zobrazená fyzická návaznost zařízení a procesů pro testovací aplikaci.



Obrázek 14 Propojení mezi zařízeními a jejich činnosti

6. Konfigurace generátorů pro Freescale Touch

Předpis je konfigurační soubor, který slouží jako vstup generátoru testovacích vektorů a referenčnímu systému. Konfigurační struktura má podobu JSON.

JSON JavaScript Object Notation (JavaScriptový objektový zápis, JSON) je způsob zápisu dat (datový formát) nezávislý na počítačové platformě, určený pro přenos dat, které mohou být organizována v polích nebo agregována v objektech. Vstupem je libovolná datová struktura (číslo, řetězec, boolean, objekt nebo z nich složené pole), výstupem je vždy řetězec. Složitost hierarchie vstupní proměnné není teoreticky nijak omezena (11). JSON se většinou používá jako strukturovaná databáze dat. Má podporu ve většině programovacích jazycích. JSON databáze je velice přehledná a co se týče editace je velice účinná.

V konfiguračním souboru se definují emulované signály a případné šumy, dále nastavení testovací aplikace, název a stručný popis pro testovací sadu. Jak se konfiguruje testovací sada, se postupně popíše v následujících kapitolách.

Základní struktura:

- **Head** - obsahuje základní popis o testované sadě
- **Samples** - předpis emulovaného signálu, předepisuje tvar signálu, následně se tento signál dá škálovat, přidat bílý šum, libovolnou harmonickou složku nebo offset
- **Setup** - struktura pro vygenerování nastavení pro FT a předpokládaných výsledků.
- **Tolerance**
- **Vizualize** - parametry vizualizace a test reportu
- **Periode** - perioda triggeru

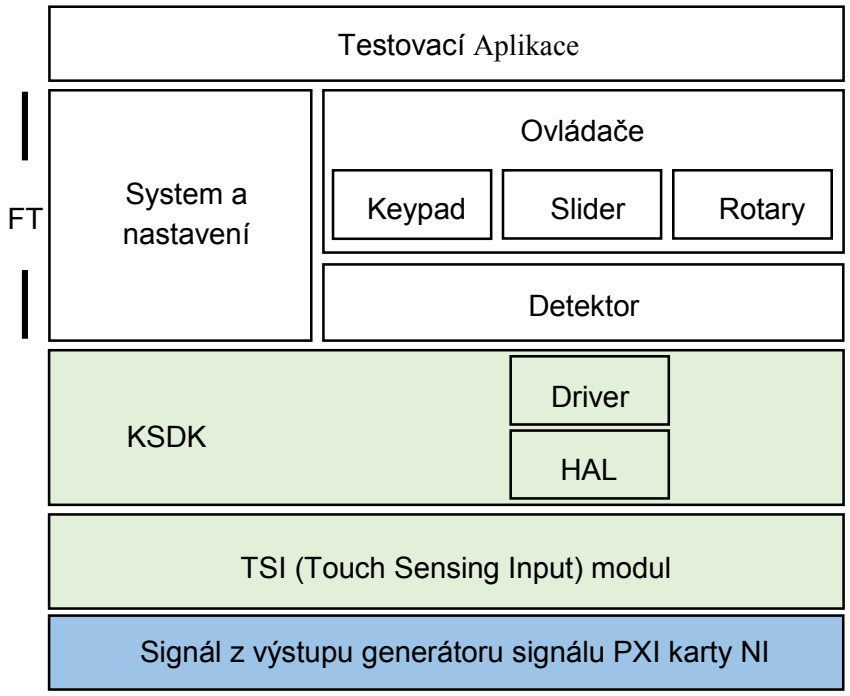
V příloze je ukázka konfiguračního souboru.

7. Generátor test vektorů

V případě testování FT jsou testovacími vektory hodnoty (čísla) simulující čítač, který měří kapacitu, ale nastavení aplikace je také vstupním vektorem. Jedné testovací aplikaci odpovídá sada inicializačního nastavení s předávanými hodnotami. Nejdříve bude zmíněn způsob, kterým budeme tyto hodnoty algoritmu FT podněcovat a poté bude popsán samotný simulovaný signál.

7.1. Implementace emulovaných signálů

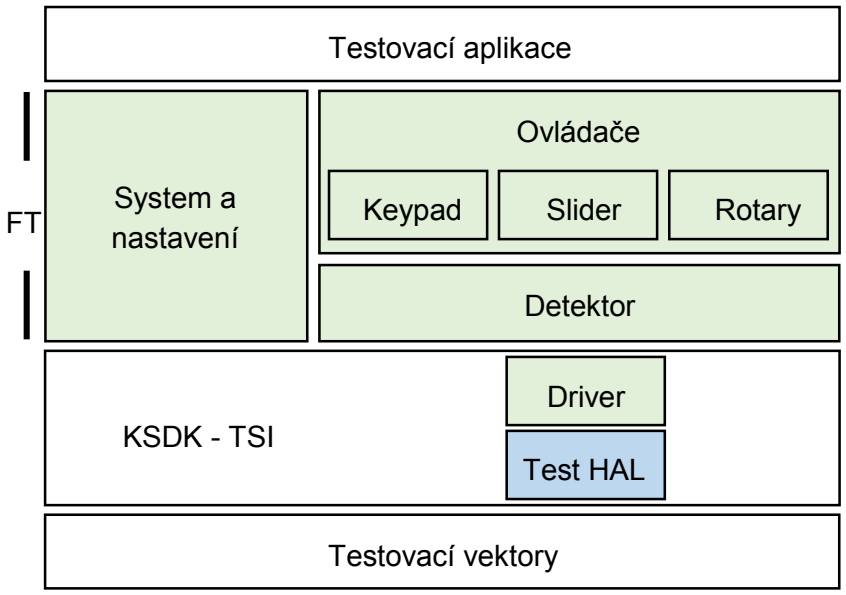
Pokud jde o testování samotného modulu TSI a nejnižších SW vrstev, které jsou na obrázku 15 označeny zelenou barvou, pak na fyzický kanál modulu TSI nebo GPIO se připojí výstup z karty PXI od NI. Tímto způsobem můžeme testovat i zbytek aplikačního rozhraní, nicméně kvůli časové a prostředkové náročnosti stačí, když pomocí PXI karty změříme pouze TSI modul. Jednoduše řečeno pomocí karty PXI je generován elektrický signál, který je zároveň vstupem TSI nebo GPIO modulu.



Obrázek 15 Emulování vstupních dat pomocí PXI

Knihovna FT už v sobě neobsahuje zdrojové kódy, které jsou abstrakcí nad hardwarem a ovladače (driver) pro obsluhu nad TSI modulem. Pro testování samotné knihovny FT není třeba testovat tyto vrstvy, ale lze je využít pro předkládání vlastních simulovaných hodnot.

Dále se práce bude soustředit pouze na testování aplikačního rozhraní FT vyznačeného na obrázku 15. Testování modulu TSI pomocí PXI karty bude popsáno v kapitole Emulace fyzické vrstvy Touch Sensing.



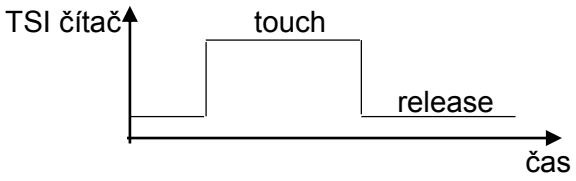
Obrázek 16 Předkládání emoluvaných vstupních dat pomocí vlastní funkce

Na obrázku 16 je zobrazen druhý způsob jak stimulovat simulovaný signál. HAL, což je Hardwarová Abstrakční vrstva, která je rozhraním mezi hardwarem a softwarem obsahuje funkce s jednoduchými příkazy pro HW. „Jaký je stav modulu?“, „Začni měřit!“ nebo „Vrať změřenou hodnotu“. Právě funkce, která čte změřenou hodnotu z paměti modulu TSI je nahrazena funkcí, která vyčítá hodnoty z polí test vektorů reprezentující simulovaný signál.

7.2. Emulovaný signál

TSI modul měří, kolik krát je frekvence externího oscilátoru rychlejší než frekvence jeho referenčního oscilátoru. Pomocí výsledků takto naměřených hodnot se v knihovně vyhodnocuje dotek. Simulovaný signál bude představovat tyto hodnoty. Test vektory pro měřící periferie TSI i GPIO mají stejný charakter, tak v následném textu bude zmiňován pouze TSI metodu.

Tvar základního simulovaného signálu má obdélníkový průběh a význam hodnot je znázorněn na obrázku 17.



Obrázek 17 Ideální průběh naměřeného signálu

Generátor simulovaného signálu je program, který podle předlohy vygeneruje soubory obsahující pole hodnot reprezentující emulovaný signál. Nemusí být pouze jeden, ale několik, takže můžeme emulovat chování více elektrod.

7.3. Generátor emulovaného signálu

Validace FT se řídí podle požadavků na testování, které předkládají vývojáři softwaru. Důvod proč je generátor vstupních vektorů zaveden, je usnadnění práce a ušetření času.

Generátor emulovaného signálu je program, který vytvoří soubory s polí hodnot představující emulovaný signál. Program je popsán skriptovacím jazykem Python. Vstupním parametrem tomuto programu je podstruktura *samples* z předpisu pro generátory. Struktura *samples* má následovný tvar:

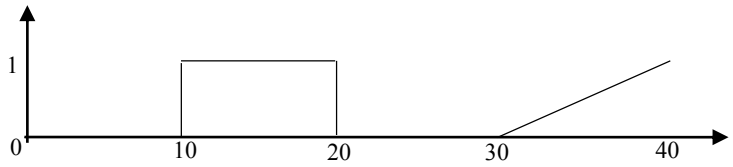
- **Data** - pole následujících podstruktur, které za sebou sekvenčně navazují
  - **Shape** (tvar signálu) - pole textových řetězců
  - **Add** - číslo, kterým se škáluje tvar signálu
  - **Offset** - hodnota, která se přičte k signálu
  - **Macro** - pole textových řetězců
  - **Iteration** - počet iterací vygenerovaných dat
- **Noise** - v této struktuře se definuje pole signálů, které se přičtou k tvaru signálů

Struktura *data* obsahuje informace o základních signálech. Počet signálů, a z toho vyplívá i počet elektrod, závisí na počtu textových řetězců v poli *shape*. Jeden textový řetězec tedy odpovídá emulovanému signálu na jedné elektrodě.

Tvar signálu je definován pomocí krátkých srozumitelných příkazů poskytující rychlý vývoj testovacích vektorů. Jednotlivé příkazy mají následující tvar

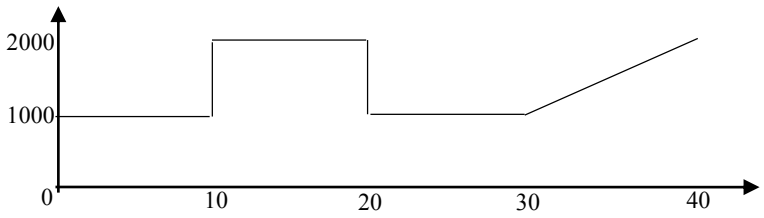
*<hodnota, ke které se směřuje>***x***<počet kroků směřující k hodnotě>*

a jsou odděleny mezerami. Počáteční hodnota je 0. Například „0x1 0x9 1x1 1x9 0x1 0x9 1x10“ generátor převede na pole hodnot, které opisují jednu periodu obdélníkového signálu začínající hodnotami 0, pak10 krát hodnotami jedna a nakonec 10 hodnot 0 a lineární stoupání končící v hodnotě 1. Viz obrázek 18.



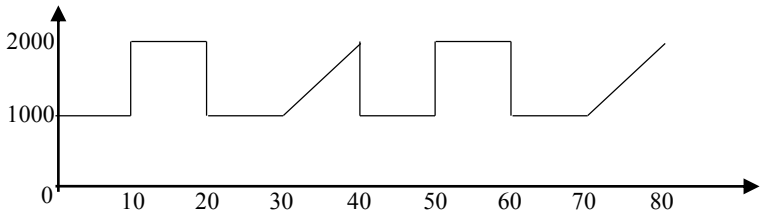
Obrázek 18 Tvar užitečného signálu

Hodnotami struktur *add =1000* a *offset = 1000* pak výsledná data vypadají následně.



Obrázek 19 Normalizovaný tvar užitečného signálu

Iteration udává, kolikrát se tyto data budou opakovat. Nastaví-li se do této podstruktury číslo 2, dvakrát za sebou se vygeneruje pole ze stejných hodnot.



Obrázek 20 Iterace předepsaného signálu

Volitelnou položkou struktury pro generátor je *macro*. *Makro* dělá to samé, jako makro v jazyce *C* a to, že na místo nějakého symbolu se nakopíruje nějaký text. Definice makra vypadá následovně:

<symbol><<<příkaz>

Například pole maker nadefinované *["|<<1x1 1x10", "_<<0x1 0x10"]* zjednoduší modelování obdélníkového signálu. Pokud jsou makra nadefinovaná, mohou se použít v *shape* struktuře.

```
"shape":
[
  "__|_|_|_|_|_",
  "___|_|_|_|_|",
  "__|_|_|_|_|_",
  "___|_|_|_|_|"
]
```

Podle takto nadefinovaného tvaru, generátor emulovaného signálu vygeneruje čtyři pole hodnot reprezentující čtyři signály obdélníkového tvaru.

7.4. Šum

Ke tvaru signálu je v generátoru možné přičíst bílý šum nebo harmonickou složku. Šum je nezbytný při testování FT, protože FT obsahuje filtry pro jejich minimalizaci a jeden z detektorů, který rozpoznává dotek, šum k výpočtu potřebuje.

V běžném slova smyslu znamená šum zvukové nebo hlukové znečištění. V elektronice se termín šum používá k označení elektronického signálu korespondujícího s akustickým šumem (v audio-systémech), nebo signálu korespondujícího s vizuálním šumem, který může být viditelný jako „sníh“ na špatných televizních nebo video snímcích. Ve zpracovávání signálu může šum znamenat data bez významu, tedy data, která nejsou použita pro přenos signálu a jsou jen produkována jako nechtěný vedlejší produkt jiných aktivit. (11)

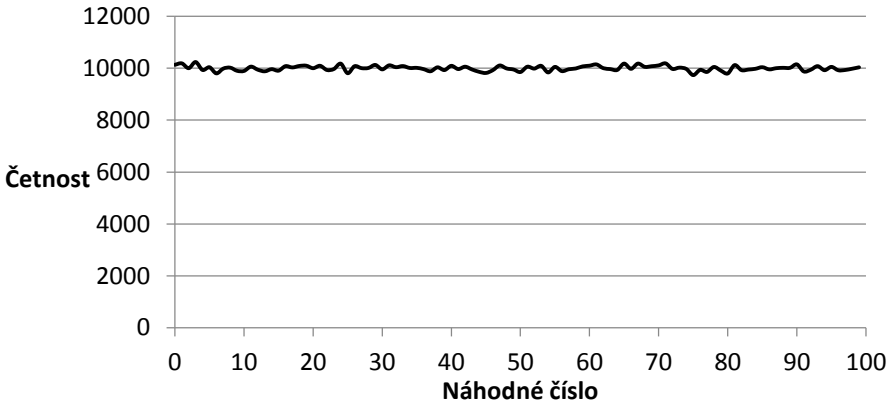
Bílý šum je náhodný signál s rovnoměrnou výkonovou spektrální hustotou. Signál má stejný výkon v jakémkoli pásmu shodné šířky. Například pásmo široké 20 Hz mezi 40 a 60 Hz má stejný výkon jako pásmo mezi 4000 a 4020 Hz. Bílý šum je tak nazýván jako analogie s bílým světlem, které obsahuje všechny frekvence. Nekonečný frekvenční rozsah signálu bílého

šumu je pouze teoretický. Kdyby byl nenulový výkon na všech frekvencích, celkový výkon takového signálu by byl nekonečný. V praxi je signál „bílý“, pokud má ploché spektrum v definovaném rozsahu frekvencí (11).

Důkaz, že funkce *random* programovacího jazyka Python je schopna generovat bílý šum, je demonstrován následujícím příkladem.

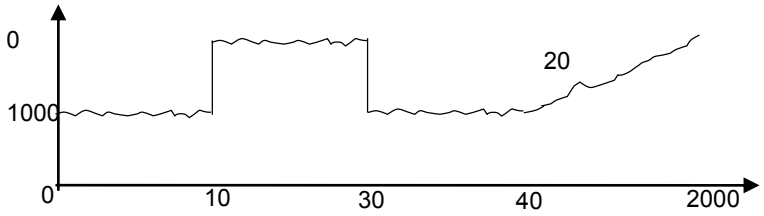
```
random_array = [random.randint(0, 99) for x in range(1000000)]
```

Tento kousek Pythonovského kódu vygeneruje milion náhodných celých čísel v intervalu ‹0,99›. Četnost vygenerovaných čísel, která je zobrazena v grafu, je lineární. To znamená, že funkce pro generování náhodných čísel v Pythonu můžeme považovat za bílý šum.



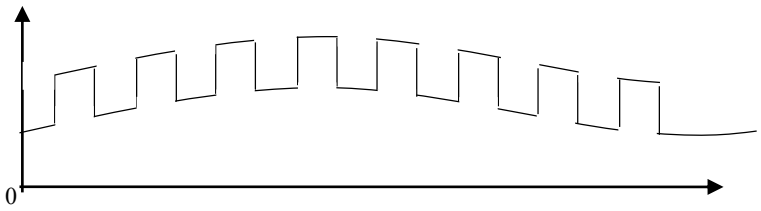
Obrázek 21 Histogram náhodných čísel

Přičtením šumového signálu k tvaru signálu dostaneme signál, který může vypadat takto:



Obrázek 22 Užitečný signál sečtený se šumem

Pomocí harmonické signálu můžeme také vytvářet šum, dokonce pomocí harmonických signálů si můžeme vytvořit signál, jaký chceme. Při testování knihovny FT se aditivní harmonický signál používá při simulování změny prostředí. Změna vlhkosti, teploty má za následek driftování signálu. Vygenerovaný signál je uveden na obrázku 23.



Obrázek 23 Příklad využití harmonického signálu jako emulace změny prostředí

Ke tvaru signálu můžeme, definováním *noise* struktury v předpisu pro testování FT, můžeme přičítat šumové signály. *Noise* struktura je pole signálů do kterých můžeme přidávat:

- **White noise** – bílý šum
  - SNRdB – jednotka dB
- **Harmonic** – harmonický signál
  - SNRdB – jednotka dB
  - FrequencyHz – frekvence sinusového signálu, jednotka Hz

SNR (Signal to Noise Ratio) je poměr výkonu užitečného signálu k signálu šumu.

$$SNR = \frac{P_S}{P_N} \quad (2)$$

Kde P_S je výkon signálu a P_N je výkon signálu šumu. Nicméně takto vypočítaný SNR udává v jakém poměru je signál a šum. Pokud to chceme mít v decibelech, musíme tento poměr navíc přepočítat.

$$SNR, dB = 10log_{10}SNR \quad (3)$$

Decibel je jednotka nejznámější svým použitím pro měření hladiny intenzity zvuku, ale ve skutečnosti se jedná o obecné měřítko podílu dvou hodnot, které se používá v mnoha oborech. Jedná se o fyzikálně bezrozměrnou míru, obdobně jako třeba procento, ovšem na rozdíl od něj je decibel logaritmická jednotka, jejíž definice souvisí s objevením Fechner-Weberova zákona, že totiž lidské tělo vnímá podněty logaritmicky jejich intenzitě (i velké změny velkých podnětů způsobují jen malé změny počitků). Pro představu 20dB je poměr výkonů 100 násobný, 30dB 1000 násobný. (13)

Následnými rovnicemi je popsán algoritmus, kterým se pomocí zadaného SNR v decibelech vytvoří výsledný signál s přidaným šumem. Základní signál je nazván $s(t)$ a šumový signál $n(t)$.

Poměr výkonu užitečného signálu a šumu (SNRdB) se do předpisu testovacího systému udává v decibelech. Nejprve se vypočte SNR, ze SNRdB, kde SNR je poměr mezi výkony signálů. Upravením rovnice 3 dostaneme:

$$SNR = 10^{\frac{SNRdB}{10}} \quad (4)$$

Díky známému SNR a vypočtenému výkonu základního signálu můžeme zjistit výkon signálu šumu.

$$P_N = \frac{P_S}{SNR} \quad (5)$$

Kde obecný vztah pro výpočet výkonu je

$$P = \frac{1}{T}\int_0^T s^2(t)\,dt. \quad (6)$$

Využijeme-li tento vztah pro náš diskrétní signál, vypočtený výkon vypadá takto:

$$P_S = \frac{1}{n}\sum_{i=0}^{n} s^2(i) \quad (7)$$

Výsledný šum vznikne škálováním námi předem vygenerovaného šumu, který pak můžeme přičíst k základnímu signálu.

$$n'(t) = n(t)\sqrt{P_N} \quad (8)$$

Kde $n'(t)$ je již výsledný naškálovaný šum.

Pro kontrolu můžeme porovnat výsledné SNR se zadaným.

$$P_{N'} = \frac{1}{n}\sum_{i=0}^{n} n'^2(i) \quad (9)$$

$$SNR' = \frac{P_S}{P_{N'}} \quad (10)$$

Kontrolní SNR by mělo být stejné jako SNR‘ námi vytvořeným šumem.

7.5. Generátor konfigurací FT

Knihovna FT pro svůj chod vyžaduje konfiguraci. Tato konfigurace popisuje, na jakých elektrodách MCU se má měřit kapacita, jaké detekční algoritmy mají elektrody používat, nastavení ovládacích prvků a měřící modul. Tyto konfigurace se také berou jako testovací vektor, který má využití v inicializaci FT.

Nastavení FT je popsán v předpisu pro testování ve struktuře *setup*.

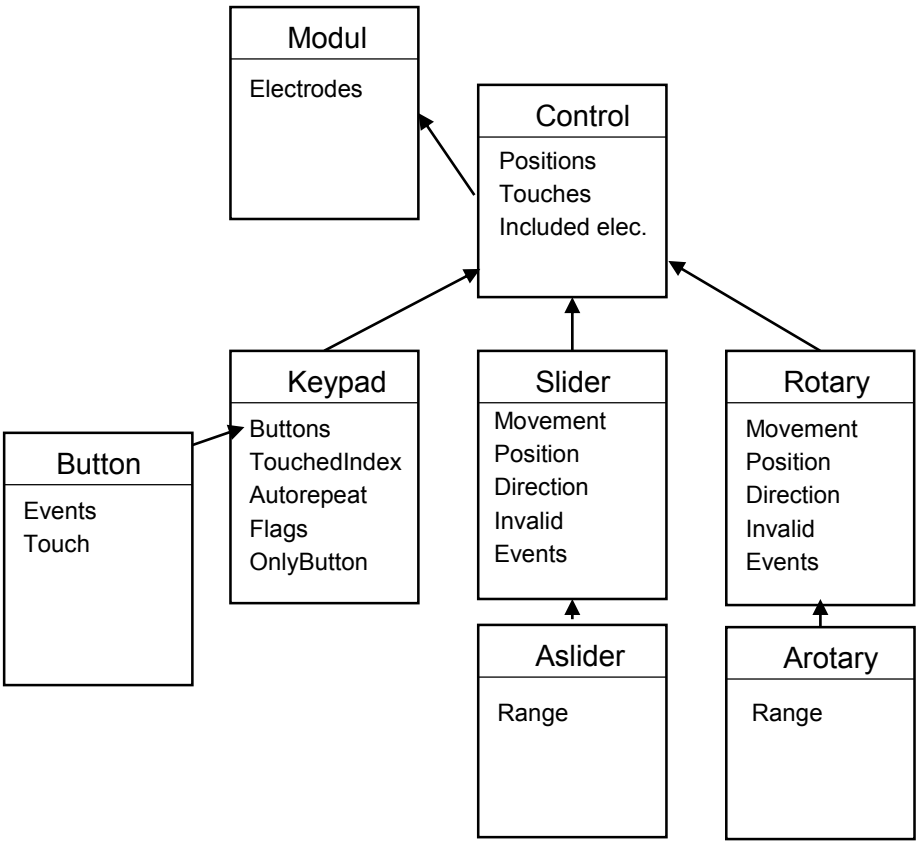
- resolvers – pole ovládacích prvků s jejich nastavením a přiřazenýma elektrodami
- keydetection – pole detektorů doteku s jejich nastavením a přiřazenýma elektrodami
- modules – pole použitých modulů s jejich nastavením a přiřazenýma elektrodami
- periode_ms – doba, jak často je volána funkce *ft_trigger()*
- init_periods – počet period nutných k inicializaci konstant za běhu FT.

Generátor konfigurací podle přepisu vygeneruje soubor *setup.c* a *setup.h*, které se přikládají k testovací aplikaci.

8. Referenční systém

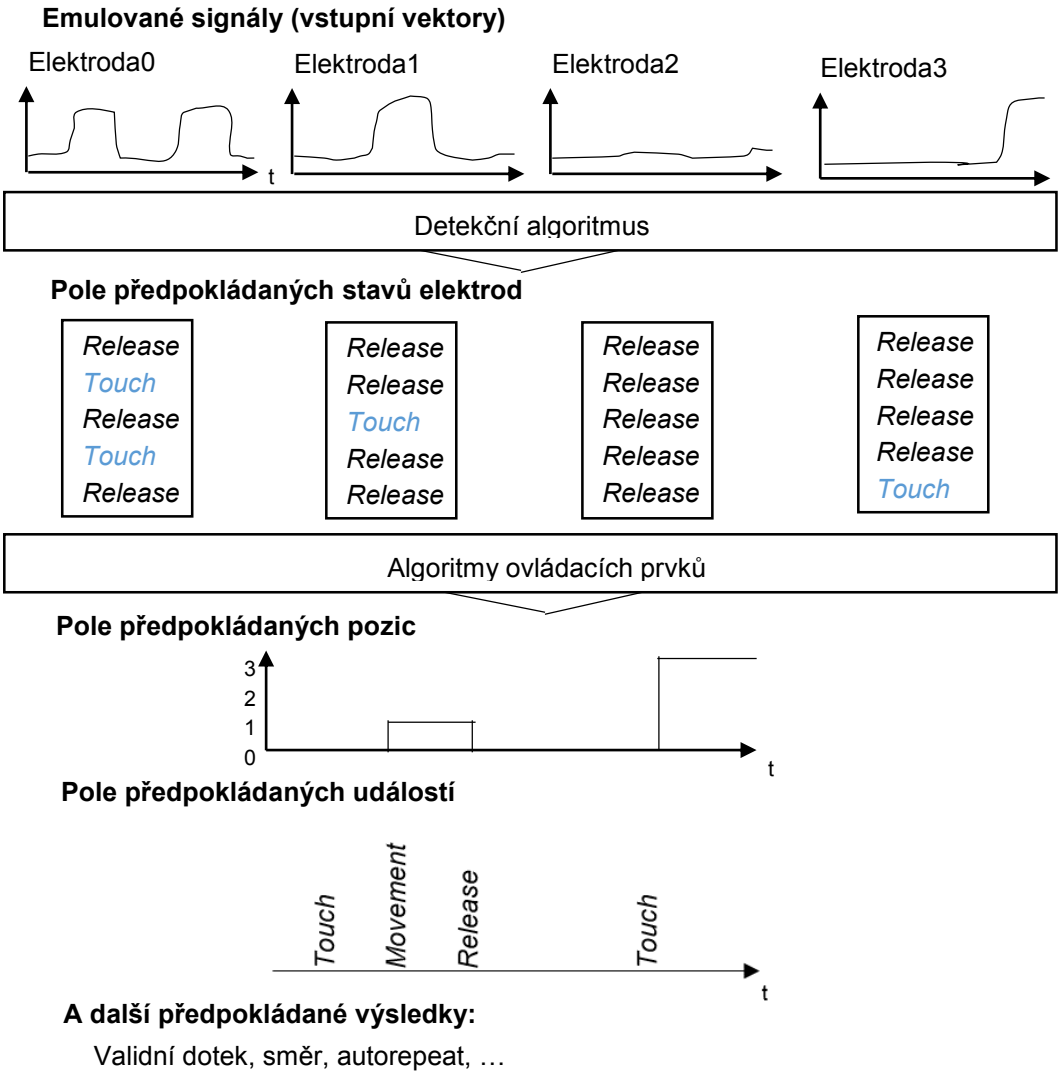
Samotné testovací vektory při testování nestačí. Je zapotřebí, aby hodnoty, které algoritmy v knihovně FT vypočtou, se porovnaly s předpokládanými výsledky. Referenční systém FT je program, který napodobuje chování FT a generuje předpokládané výsledky. Zdroj informací, podle kterého je referenční systém vyvinut, je dokumentace a podklady vývojového týmu. Hodně funkcionalit bylo laděno na pravidelných schůzkách s vývojovým týmem.

Program je naprogramován pomocí jazyku Python. Python je objektově orientovaný jazyk, takže se snadno a rychle udržuje. Další výhodou je snadná rozšiřitelnost. Pokud vznikne požadavek na testování porovnávání hodnot, které již nejsou generovány tímto systémem, dá se snadno rozšířit přidání metod do objektu nebo vytvořit objekt nový.



Obrázek 24 Diagram tříd

Obrázek 24 zobrazuje základní strukturu referenčního systému, kde jsou zobrazeny veřejné parametry tříd, které obsahují nastavení systému a pole předpokládaných výsledků.



Obrázek 25 Diagram chování FT

Na tomto obrázku 25 je diagram chování FT. Vstupem referenčnímu systému jsou testovací vektory, kterými jsou vygenerovaná data emulovaného signálu a nastavení FT. Pro každou hodnotu z emulovaného signálu se vyhodnotí stavy elektrod. A dále podle těchto stavů (*touch*, *release*) se vygenerují pole, která představují předpokládané výsledky pro každý cyklus programu. Vygenerované výsledky se při nahrávání testovací aplikace ukládají do paměti MCU.

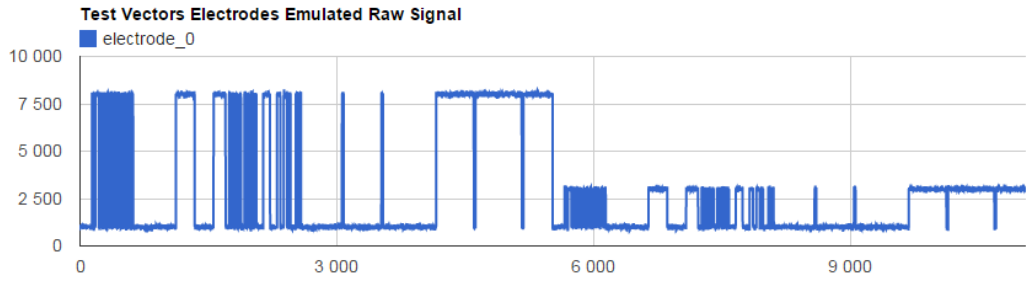
9. Vizualizace

Pro kontrolu vygenerovaných hodnot a pro závěrečnou testovací zprávu byl implementován podpůrný program pro generování vizualizovaných hodnot pomocí grafů, textu a tabulek. Tento program umí ukládat vložená data do databáze a vytvořit webovou aplikaci s interaktivními grafy a tabulkami.

Program, který generuje webovou aplikaci, je implementován v jazyce Python a obsahuje metody snadného vytváření grafů, tabulek, textů s využitím zajímavých webových vlastností.

Webová aplikace využívá online JavaSkriptovou knihovnu od firmy Google nazvanou Google Charts.

9.1. Google Chart API



Obrázek 26 Příklad grafu s podporou Google Chart

Google chart API je online knihovna k vizualizaci dat na webových stránkách. Od jednoduchých grafů ke komplexním větveným diagramům a mapám. Způsob použití Google Charts je pomocí JavaScriptu. Knihovna se načte ze serverů Googlu a pomocí JavaScriptu se popíší data a nastavení grafu. (14)

Grafy jsou vystaveny jako třídy JavaScript, a Google Charts poskytují mnoho typů grafů, které se můžou použít. Výchozí nastavení již obsahuje vše, co je potřeba pro běžné použití, ale vždy je možné si graf přizpůsobit tak, aby pasovali do webových stránek. Grafy jsou vysoce interaktivní a dovolují vystavit události, které umožňují společně s řídicí panely nebo s jinými ovládacími prvky integrované z webové stránky. Grafy jsou vykresleny pomocí technologie HTML5 / SVG k zajištění kompatibility všech prohlížečů a přenositelnosti mezi platformami pro iPhone, iPad a Android. Uživatelé webových stánek nebudou muset nikdy instalovat plugginy nebo jiné podpůrné programy. (14)

Obrázek 27 Vývoj Google Chart API (13)

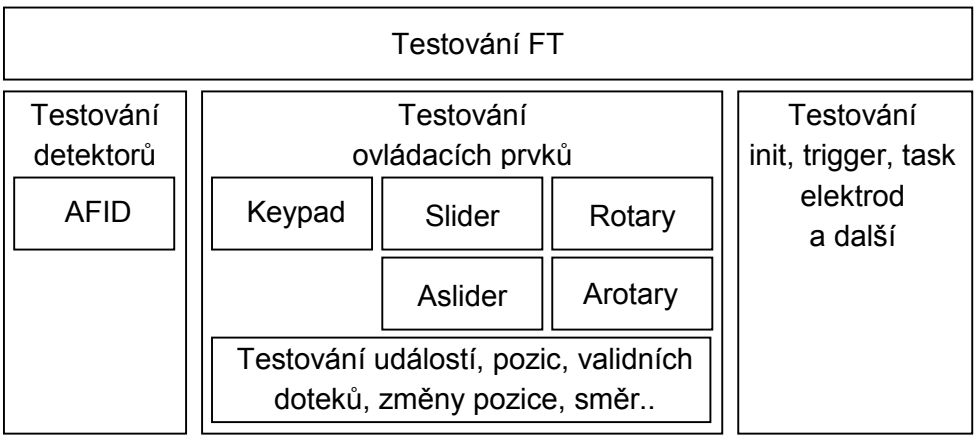
Všechny typy grafů jsou naplněny daty pomocí DataTable třídy (obrázek 27), takže je snadné přepínání mezi různými typy grafů, a tak snadno přizpůsobovat grafy potřebám uživatele. DataTable poskytuje metody pro třídění, úpravu a filtrování dat, a mohou být osazeny přímo z webové stránky, databáze, nebo od jakéhokoli poskytovatele dat podporující protokol. Tento protokol obsahuje dotazovací jazyk SQL a je realizován tabulkách Google. (14)

Nicméně není schopný splnit všechna očekávání, tudíž další vlastnosti si každý musí naimplementovat sám.

10. Testovací aplikace

Testovací aplikace je program pro testování knihovny FT. Tato kapitola popisuje jakým způsobem je aplikace vyvíjena a jaký je její obsah. Testovací aplikace je vyvíjena na PC programovacím jazykem *C*, je to program, který se spouští v MCU. Jsou zde také popsány algoritmy testování vybraných částí knihovny FT. Pro otestování požadovaných stavů algoritmus obsahuje sadu testovacích případů, které v podmínkách kontrolují správnost testovaného výrazu. Sadě testů se říká test suit a testovacímu výrazu nebo chcete-li případu test case.

V jedné testovací sadě se testují vlastnosti FT závisle na požadavcích na testování. Pokud vzešel požadavek na otestování ovládacího prvku Slider, tak testovací sada bude obsahovat pouze případy týkajícího se funkcí Slideru. Na obrázku 28 je obecný přehled testovaných částí knihovny FT. Slovem obecný je míněno, že například ovládací prvek Keypad obsahuje několik nastavení a pro vybraná nastavení je napsána jedna testovací aplikace.



Obrázek 28 Obecný přehled testovacích aplikací

Všechny testovací případy jsou testovány metodou backbox testů, to znamená, že algoritmus nezasahuje, nebo přesněji řečeno nedělá modifikace do testovaného kódu. Tato metoda je z pohledu testování bezpečná. Při testování byly využity pouze základní instrukce a veřejné knihovní funkce.

Testovací podmínky v jednom testovacím případu jsou vyhodnocovány v makrech, která pro jednu podmínku provede více operací. Prvním operací je vyhodnocení testované podmínky, na jejím základě se inkrementuje celkový počet správných nebo nesprávných testovacích výrazů. Pokud je podmínka vyhodnocena jako chybná, pošle se po sériové sběrnici zprávu o chybě. Veškeré zprávy poslané po sériové lince jsou odposlouchávány na PC a zpracovány pro vygenerování testovací zprávy.

Kromě zpráv o chybách, lze z testovací aplikace posílat informační zprávu o libovolném stavu, ke kterému máme v paměti MCU přístup. Tato vlastnost nám umožňuje kromě informace, jaká chyba nastala, také uvést okolnosti této chyby a přináší obrovskou výhodu při vyšetřování chyb a vývojářům to usnadní práci při její opravě. Například pokud

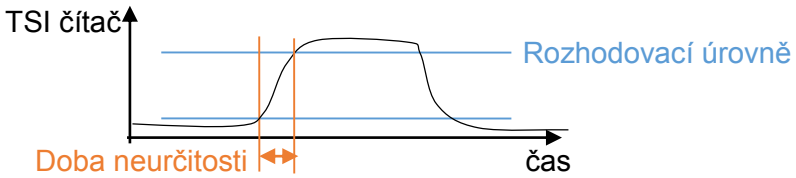
u detektoru, který vyhodnocuje, jestli je na elektrodě dotek, se pravidelně hlásí chyba, tak mohu
do testovací aplikace přidat makra pro hlášení informačních zpráv o stavu signálů, rozhodovacích úrovní a filtrů, které jsou graficky znázorněné v testovacím reportu jako grafy, aby mohly vést k odhalení chyby.

10.1. Vývoj testovací aplikace

Program IAR IDE je vývojové prostředí pro vývoj programů pro mikrokontroléry. Program je popsán v programovacím jazyce *C*. IAR IDE je komerční nástroj pro vývoj vestavěných aplikací. Je složen z textového editoru s vlastnostmi pro snadné navigování ve vytvořeném kódu, kompilátoru, linkeru a ladícími nástroji.

10.2. Tolerance

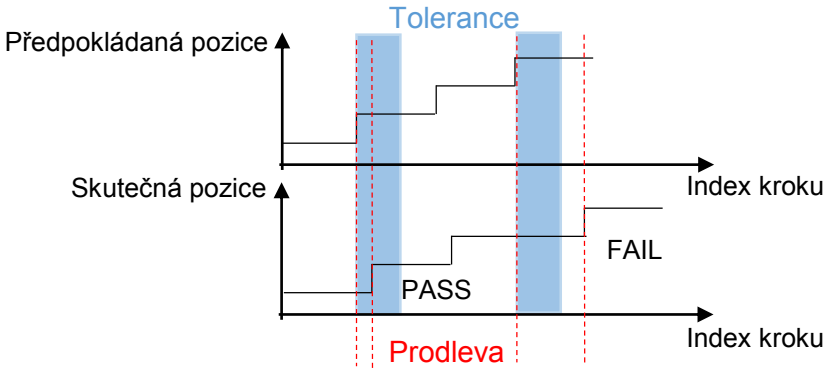
Jednou z věcí, se kterou by se mělo počítat, je doba při změně stavu při doteku elektrody. Knihovna FT používá ke zpracování signálu filtry, to má za následek, že při seberychlejší změně vstupních vektorů to bude mít za následek opoždění vyhodnocení doteku. Na obrázku 29je tento jev vidět.



Obrázek 29 Tvar zfiltrovaného signálu

Řešením by mohlo být použití stejných filtrů v referenčním systému, ale to by mělo za následek napodobování kódu, který chceme testovat. Takové opisování kódu je nepřípustné, protože účelem je najít co nejvíce chyb. Další nevýhodou je špatná údržba, tzn. při vývoji FT filtry v referenčním systému pořád obnovovat.

Chytrým řešení v testování FT je zavedení tolerančního pásma. Pokud změna testovaného stavu se projeví v požadované toleranci, testovací systém nebude považovat opožděné chování za chybu, viz obrázek 30.



Obrázek 30 Tolerance

Příkladem je testování pozice, kterou vypočítal ovládací prvek FT. V každém kroku běhu aplikace se testuje test case, který porovnává předpokládanou pozici se skutečnou hodnotou, kterou vyhodnotila knihovna FT. Pokud prodleva mezi změnami je větší než pásmo tolerance, test case to vyhodnotí jako chybu. Stejný princip platí pro každou jinou vlastnost, která závisí na okamžitých vstupních hodnotách FT (validní dotek, směr, události…).

Tolerance v testovacím systému pro FT je konstanta, která se zadává podle požadavků na testování a po konzultaci s vývojovým týmem FT.

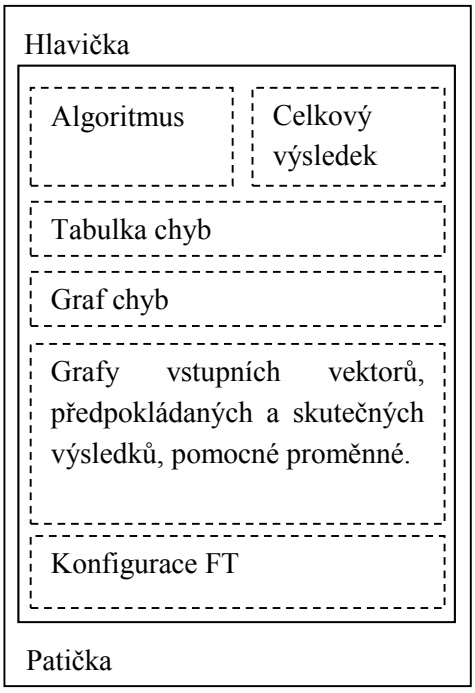
11. Testovací zpráva

Výstupem každé testovací aplikace je testovací zpráva, která je strojově generovaná, žádná jeho část není dopisovaná ručně. Zdrojem informací pro tuto zprávu jsou test vektory, předpokládané výsledky, samotný zdrojový kód testovací aplikace a data nasbíraná ze sériové sběrnice. Testovací zprávu generuje Python skript.

Forma této zprávy je HTML dokument obsahující hlavičku zprávy, způsob testování, výsledky testování a patičku zprávy. Mimo jiné zpráva obsahuje i test vektory s předpokládanými výsledky. Webový prohlížeč obsahuje každý PC nezávisle na operačním systému a z toho pramení jedna z výhod využití HTML dokumentu. Navíc díky JavaScriptu je zvýšena míra interaktivity a prohlížeč může vykreslovat grafy, pěkné tabulky a ovládací prvky.

JavaScript je interpretovaný programovací jazyk pro webové stránky, často vkládaný přímo do HTML kódu stránky. Jsou jím obvykle ovládány různé interaktivní prvky uživatelského grafického rozhraní (tlačítka, textová políčka) nebo tvořeny animace a efekty obrázků.

V obrázku 31 je návrh struktury testovací zprávy. Skutečná testovací zpráva je v příloze.



Obrázek 31 Struktura testovacího reportu

Hlavička obsahuje stručné informace testované aplikace. Nadpis a text uvádí podrobnosti testovaní aplikace. V textu by měli být hesla, která usnadní vyhledání testu. Důležitou informací v hlavičce je identifikační číslo verze, které se test týká.

Pod hlavičkou je obsažen sekvenční popis testovacího algoritmu a procentuální vyjádření všech nalezených chyb v grafické podobě, takže na první pohled je jasné jak testovací sada dopadla. Python skript umí vytáhnout komentáře z testovací aplikace a nakopíruje je

do testovací zprávy jako popis algoritmu testovací sady, takže z testovací zprávy je známo podle čeho testování probíhalo.

Výčet chyb je zobrazen v tabulce, které obsahují informaci o tom, čeho se chyba týká, detail chyby a index kroku, ve kterém se chyba vyskytla. Tyto chyby jsou zobrazeny i v grafu. Jedna řada neboli série je typ chyby a nabývá hodnoty 1 jen na místech, kde se chyba vyskytla.

Testovací zpráva také obsahuje informace o okolnostech při testování. Na požadavek vývojářů může testovací report obsahovat informace o stavech knihovny FT.

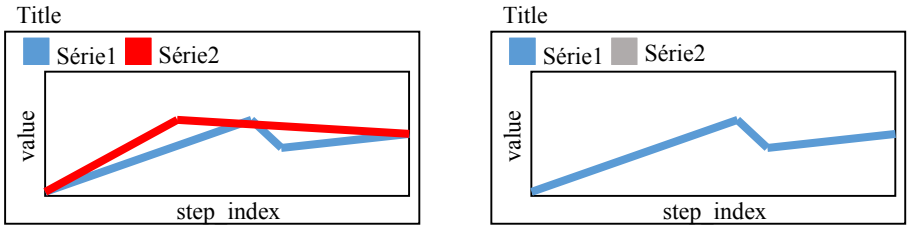
Samozřejmostí testovacího reportu jsou testovací vektory včetně konfigurace nastavení FT, aby byly zřejmé okolnosti testování. Na závěru je jméno zodpovědné osoby testování a datum.

Report je předkládán vývojářům knihovny FT, tak je tedy nutné, aby obsahoval dodatečné informace i o tom, za jakých podmínek a kdy k chybě došlo.

11.1. Interaktivita testovací zprávy

S rostoucí složitostí testování se zvětšoval i testovací report. Velké množství grafů tabulek a textu zhoršuje přehlednost zprávy. Čitelnost se výrazně zlepší a hlavně zpřesní pomocí interaktivity.

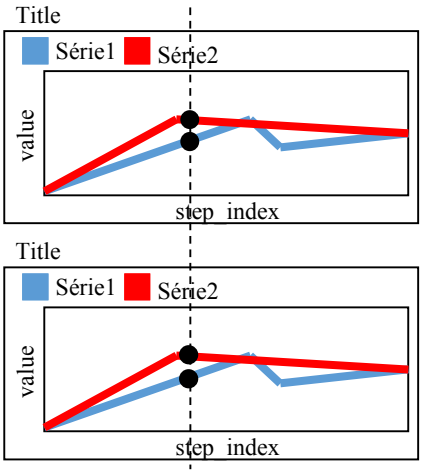
První vlastnost lze využít u grafů obsahující více sérií dat, kde můžeme poklepáním na název série v legendě tuto sérii v grafu schovat, viz obrázek 32. Dalším kliknutím se série znovu objeví. Tato funkce výrazně zpřehlední grafy se třemi a více sériemi jako jsou třeba grafy s nasimulovanými signálem na několika elektrodách, grafy obsahující řadu dat skutečných a předpokládaných výsledků.



Obrázek 32 Skrývání/zobrazování řady

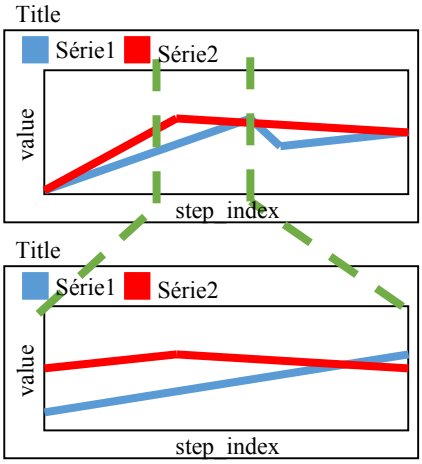
Velké množství hodnot v jedné sérii dat má za následek zhuštění bodů a poté jsou jednotlivé body navzájem nerozlišitelné. Z tohoto důvodu byly naimplementovány další vlastnosti testovací zprávy, týkají se celkového zpřehlednění grafů a zjednodušení jejich procházení. Jde o funkci synchronní horizontální zoom a funkci zvýraznění všech stavů v uživateli daném času.

Kliknutím na libovolný řádek v tabulce chyb nebo kliknutím na bod v grafu chyb se zvýrazní všechny body všech grafů, které se nacházení ve stejném čase, jako označená chyba. Výsledné zvýraznění je zobrazeno na obrázku 33.



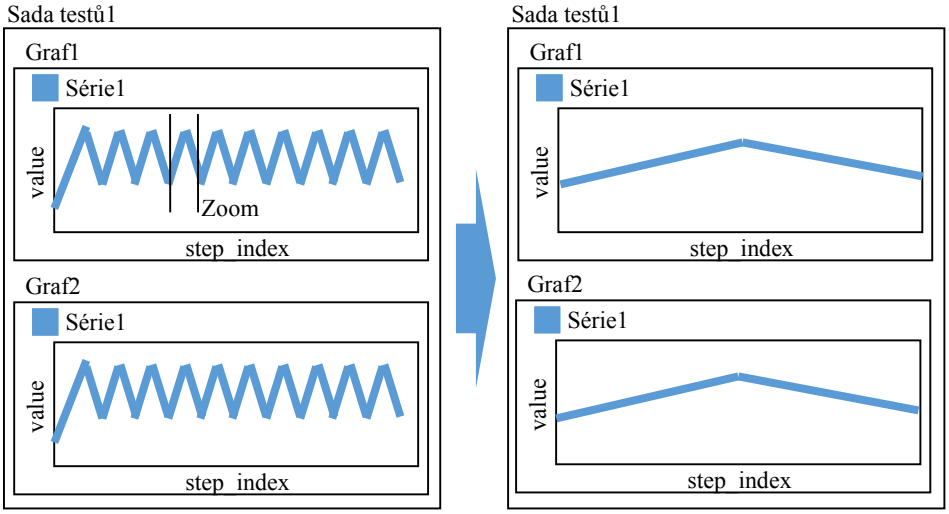
Obrázek 33 Zvýraznění všech bodů v časové relaci s chybou

Označením oblasti se graf ořízne. Kliknutím pravým tlačítkem myši se opět zobrazí všechny data.



Obrázek 34 Funkce zoom

Funkce zoom (viz obrázek 34) je sice pěkná vlastnost a Google Chart ji dokonce nabízí, ale tato schopnost je pro účely testovacího reportu nepoužitelná, protože tento report obsahuje spoustu grafů se stejnou časovou osou. Proto byl naimplementovaný synchronní zoom, který umí přiblížit všechny grafy současně. Pro lepší představu je zde obrázek 35.



Obrázek 35 Synchronní zoom všech grafů

12. Automatizované testování

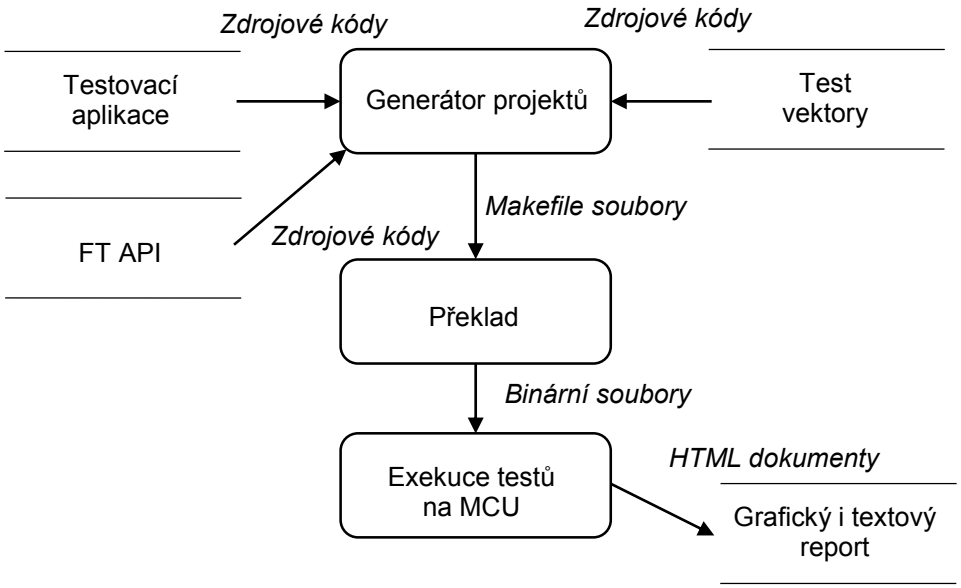
Automatizování vyvinutého testovacího systému se práce testovací testovacího inženýra zredukuje na jediný příkaz, kterým se spustí sada testů, které kompletně otestují FT. Testovací tým ve firmě Freescale používá pro automatizaci testů Framework nazvaný BEART.

12.1. BEART

Beart znamená Build Execute Analyze and Report Test. Je to interní testovací Framework firmy Freescale s administrativou, který si sám umí vygenerovat, zkompilovat a spustit testovací programy.

- BEART standardizuje kód
- Asertuje výsledky testů v testovacím kódě
- Generuje testovací konfiguraci
- Kompiluje testovací aplikaci
- Spouští testy a ukládá výsledky
- Generuje výsledek testů

Testování knihovny Freescale Touch využívá vlastní program pro načítání a ukládání výsledků testů.



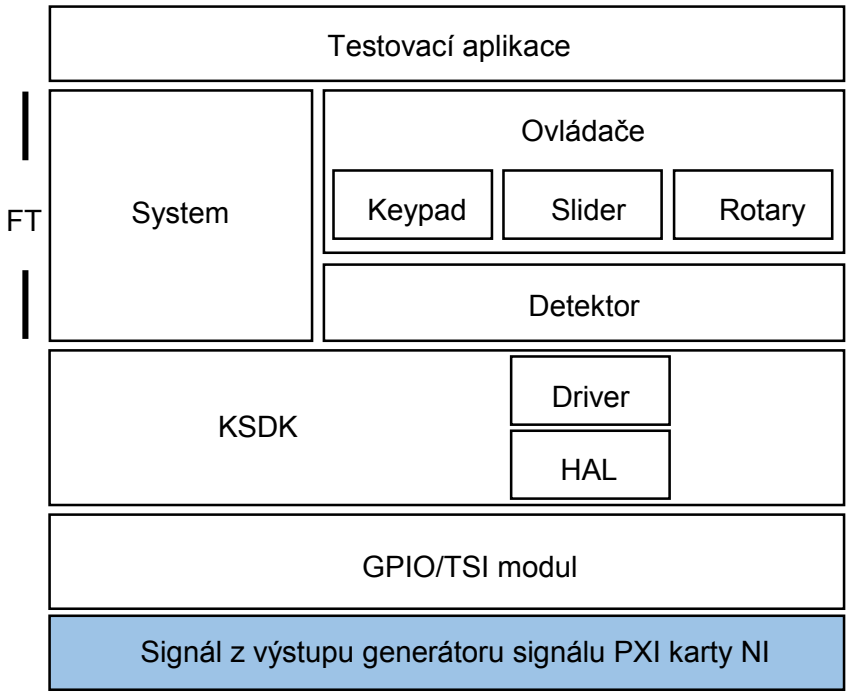
Obrázek 36 Diagram datových toků popisující BEART

Obrázek 36 je výřez z diagramu datových toků z kapitoly Testování softwaru Freescale Touch. Pro spuštění všech vyvinutých testů BEART nejprve podle všech zdrojových kódů vygeneruje projekty ve tvaru makefile. Zdrojovými kódy jsou testovací vektory, API FT a testovací algoritmus. Makefile je předpis pro kompilaci a linkování programu. Pokud jsou makefile soubory vygenerované, spustí se překlad programů do binárních souborů, které se

jeden po druhém nahrávají do MCU a spouští. Během vykonávání testovacích sad PC odposlouchává zprávy z komunikační sběrnice a z těchto zpráv generuje testovací reporty.

13. Emulace fyzické vrstvy Touch Sensing

Pro otestování softwarových vrstev pod knihovnou FT zobrazené na obrázku je nutné podněcovat data elektrodě. Ve firmě Freescale je k dispozici Testovací systém V&V. V&V obsahuje různé nástroje pro validování softwaru, jedním z nástrojů je mimo jiné BEART. Dalším nástrojem, který V&V testovací systém poskytuje je FPGA karta, která generuje signály emulující kapacitní chování při snímání FT.

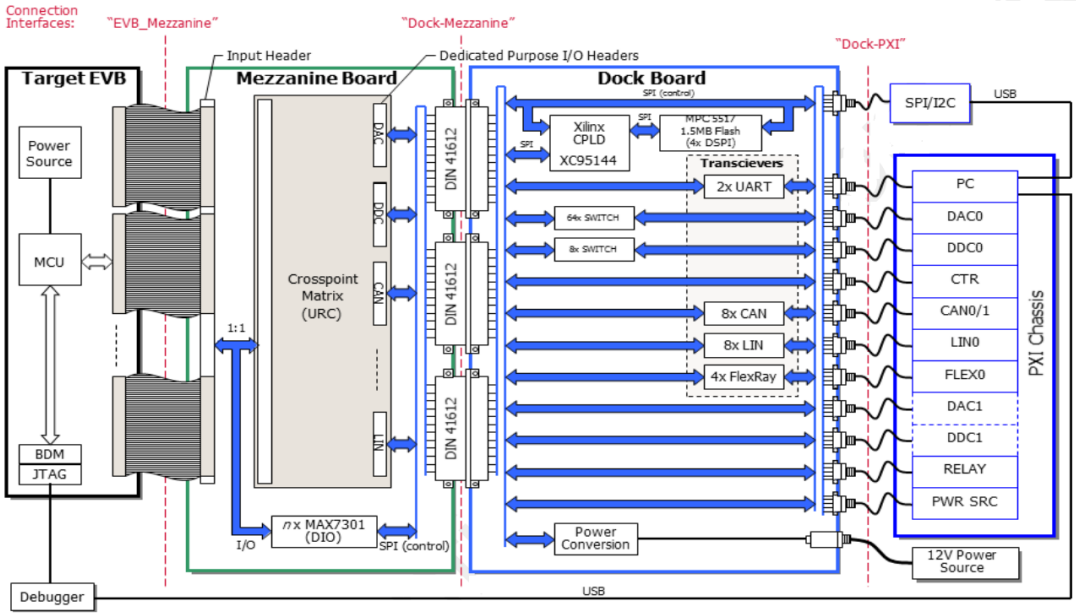


Obrázek 37 Struktura testovací aplikace

Obrázek 37 zobrazuje strukturu testovací aplikace s PXI kartou.

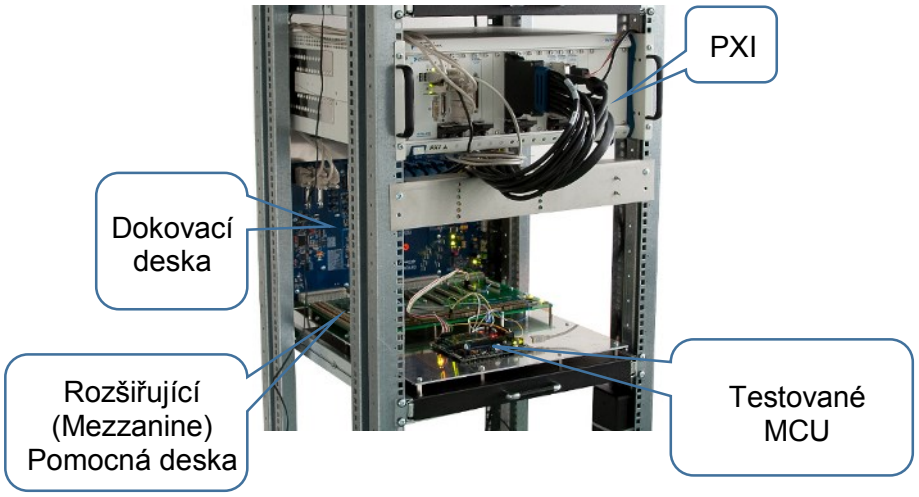
13.1. V&V testovací systém

V&V testovací systém obsahuje PXI šasi s měřícími NI moduly, dokovací desku (Dock Board) a rozšířenou základní deskou (Mezzanine Board), které poskytují jednoduché flexibilní propojovací řešení pro připojení testovací desky s testovaným cílovým MCU do hotového testovacího systému s FPGA kartou NI PXI-7854R (Podrobnosti o kartě jsou zde (13)). Blokový diagram V&V testovacího systému je na obrázku 38.



Obrázek 38 Architektura testovací stanice V&V s PXI, MCU a podpůrnými obvody

Reálné zapojení na obrázku 39.



Obrázek 39 Testovací stanice

13.2. Testovací aplikace s podporou V&V

Testovací stanice V&V je řízena mikrokontrolérem, takže ve zdrojovém kódu testovací aplikace bude obsahovat funkce ovládající V&V systém. Komunikace mezi MCU a stanicí V&V probíhá přes sériovou linku, která slouží k posílání příkazů z MCU do stanice. Těmito funkcemi jsou *IOClient FPGA Touch Emulation* a *IOClient FPGA TSI Touch Emulation*.

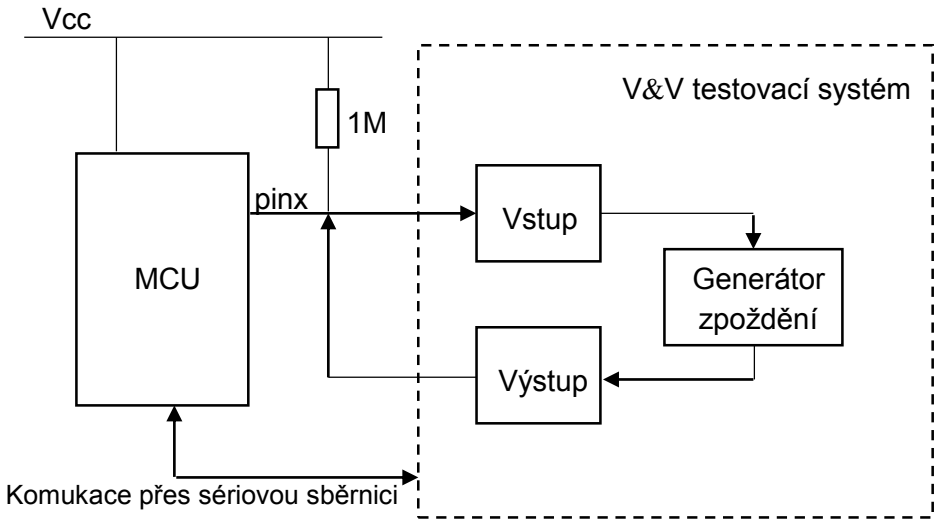
13.3. Podněcování testovaného zařízení

Podněcování elektrod testovacího zařízení je řešeno pomocí NI FPGA karty. NI FPGA karta od firmy National Instruments umí emulovat až 8 kanálů. To znamená, že na testovaném MCU je možné testovat maximálně 8 elektrod. Knihovna FT nabízí nízko úrovňové metody založené na TSI nebo GPIO periferii. Pro každou metodu je použita rozdílné metody podněcování.

- Podněcování GPIO vstupu
- Podněcování TSI vstupu

13.3.1 Podněcování GPIO vstupu

Pokud je v testovací aplikaci FT použit modul GPIO, FT využívá nízko úrovňovou metodu nabíjení RC obvodu, kdy se měřením času nabíjení RC členu měří kapacita. Testovací systém V&V proto musí simulovat „kapacitní“ chování na elektrodě MCU uzemněním tohoto pinu po definovanou dobu. Obrázek 40 je základní blokový diagram.



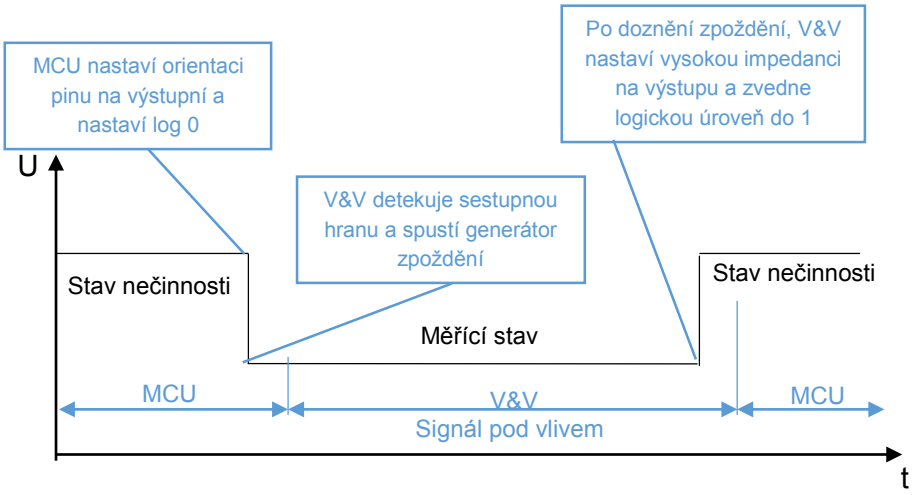
Obrázek 40 Podněcování vstupu MCU testovacím systémem

Podněcování elektrody je obsluhováno pomocí funkce *IO Client FPGA Touch Emulation*. Tato funkce emuluje dotek na elektrodě. Emulovaný signál má tvar jedné periody obdélníkového signálu s definovaným trváním. Signál přechází ve stavu ze stavu logické úrovně 1 do stavu logické úrovně 0 a pak zpátky do stavu log. 1. Minimální doba signálu může být 1 us. Rozeznáváme dva stavy tohoto testovacího zařízení.

- Stav nečinnosti – pin MCU je konfigurován jako vstupní a je vyzdvihnut rezistorem o velikosti 1M. Testovací systém VV drží vysokou impedanci, takže generátor zpoždění je vypnut.
- Měřící stav – Tento stav spouští MCU. MCU nastaví pin jako výstupní a do úrovně log. 1. V&V detekuje sestupnou hranu a spustí generátor zpoždění po definovanou

dobu. Po vypršení doby odezvy V&V systém nastaví výstup do stavu high. Rutina snímání FT bude měřit čas generovaný tímto testovacím systémem V&V.

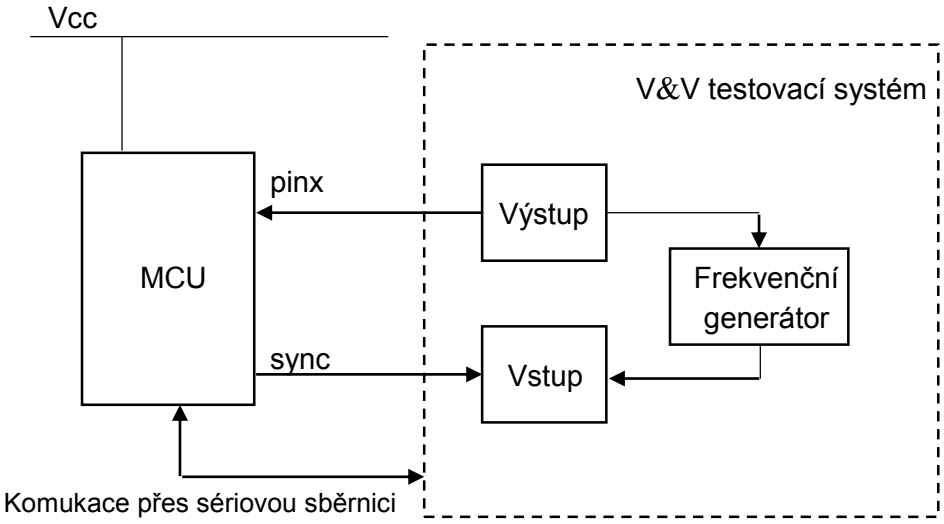
Na obrázku 41 je popsán signál při podněcování GPIO metody.



Obrázek 41 Emulace signálu při testování GPIO modulu

13.3.2 Podněcování TSI vstupu

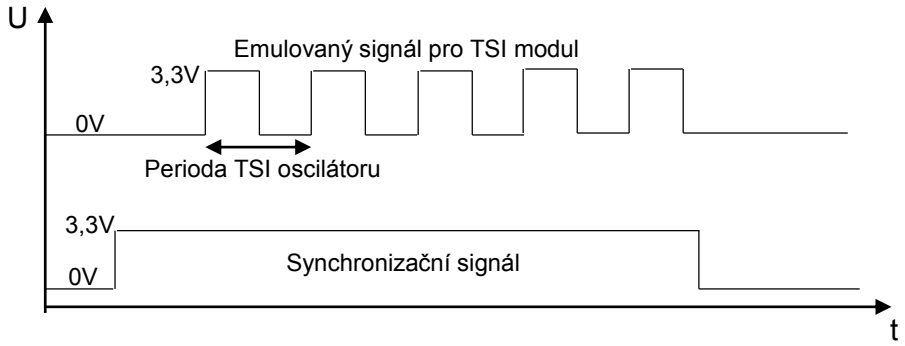
Pokud je v testovací aplikaci použit modul TSI, elektroda TSI modulu mění frekvenci externího oscilátoru TSI modulu podle snímané kapacity. V&V testovací systém simuluje měřenou kapacitu na pinech MCU změnou frekvence na pinu elektrody. Obrázek 42 je obecný blokový diagram.



Obrázek 42 Podněcování vstupu TSI v MCU testovacím systémem

Podněcování elektrody je obsluhováno pomocí funkce IOClient FPGA TSI Touch Emulation. Tato funkce emuluje dotek na elektrodě. Na vzestupnou hranu na synchronizačním kanále

(sync) začne FPGA karta generovat obdélníkový signál se střídou 50%. Sestupná hrana na synchronizačním kanále zastaví generování frekvence, viz obrázek 43. Tato funkce může emulovat signál až na osmi kanálech s minimální periodou 1us. Emulace doteku závisí na periodě signálu.



Obrázek 43 Emulace signálu při testování TSI modulu

14. Závěr

Finální podoba diplomové práce je automatizovaný testovací systém pro validování SW knihovny Freescale Touch. Tento testovací systém jediným příkazem otestuje kompletní funkcionalitu knihovny FT. Testovací systém během vývoje API FT se prokázal svou užitečností v hledání chyb s podporou pro vývojáře knihovny ve formě interaktivních testovacích zpráv.

Testovaný systém pro FT obsahuje programy pro generování test vektorů, referenční program simulující chování FT a další programy pro vytváření testovacích zpráv a usnadnění vývoje dalších testů.

Testovací zpráva, ve formě webové aplikace, zobrazuje testovací algoritmus, testovací vektory a výsledky v grafické podobě pomocí grafů a svou interaktivitou usnadňuje její čtení. Velkou výhodou je zobrazení jakýchkoliv hodnot a stavů vnitřních struktur FT, které by mohly pomoc při hledání chyb v přehledných grafech. Graficky znázorněné test vektory a předpokládané výsledky slouží jako kontrola validnosti samotného testovacího systému.

Testovací systém pokrývá většinu důležitých funkcionalit knihovny FT. Nicméně celý systém je navržen tak, aby se snadno a rychle dalo doimplementovat neotestované požadavky. Obsahuje navíc několik pomocných skriptů a programů pro snadné rozšíření testů, o kterých se nemluvilo.

Celý testovací systém byl pravidelně konzultován s vývojovým týmem. Díky této spolupráci se zdokonalovala knihovna FT i samotný testovací systém. Testovací systém již odhalil spoustu vad, které byly následně hned odstraněny. Některé chyby byly podmětem k novému detekčnímu algoritmu, který si vývojový tým nechal patentovat.

Díky podpoře testovacího týmu byl do testování FT zahrnut testovací Framework V&V, který obsahuje nástroje pro automatizaci testů a testovací systém s PXI kartou. Pomocí PXI karty je testovací systém schopný validovat nižší SW vrstvy, zejména ovladače pro TSI modul a detektor knihovny FT. PXI karta generuje hardwarové signály, které emulují kapacitní chování na měřených elektrodách.

Invencí celého testovacího systému FT by byl generátor, který by sám generoval test vektory bez předpisu a tak dosáhnout 100% pokrytí a umístění testovacího systému na server a spouštět testy z webové aplikace online.

Na projektu testování FT jsem pracoval sám s podporou testovacího a vývojového týmu. Měl jsem velkou volnost při vývoji. Vypracováním této práce jsem získal bohaté znalosti o testování. Zdokonalil jsem si schopnosti v programovacích jazycích C, Python a JavaScript.

Literatura

1. Cypress. [Online] http://www.cypress.com/fckimages/myresources/Home_Apliances _dco.JPG.

2. Freescale. [Online] Freescale Semiconductor, Inc. http://www.freescale.com.

3. Ing. Luboš Král, Ph.D., Ing. Tomáš Hazdra. Testování a diagnostika softwaru. *Automa - časopis pro automatizační techniku.* Testování a diagnostika softwaru, 2000, 08.

4. Oscar Camacho, Eduardo Viramontes a kolektiv. *Designing Touch Sensing Electrodes - Aplication Note.* [PDF] Guadalajara, Mexico : Freescale Semiconductor, Inc., 2010.

5. Gao, Xianhu. *Integrating Touch Sensing Software (TSS 3.0.1) on Kinetis L Using GPIO Method.* [PDF] místo neznámé : Freescale Semiconductor, Inc., 2012.

6. Freescale. *Touch Sensing Software - User Guide.* 08 2013.

7. KINETIS-SDK: Software Development Kit for Kinetis MCUs. *Freescale.* [Online] Freescale Semiconductor, Inc., 2015. http://www.freescale.com/ksdk.

8. KEIL Tools by ARM. *keil.* [Online] ARM. ARM.

9. IAR Systems. *IAR.* [Online] IAR Systems. https://www.iar.com/.

10. Programování v jazyku C/C++. *Sallyx.* [Online] Freescale Semiconductor, Inc..

11. Python. *Python.* [Online] Python Software Fundation. https://www.python.org/.

12. Šum. *Wikipedie, otevřená encyklopedie.* [Online] Wikimedia Project, MediaWIki. [Citace:] http://cs.wikipedia.org/wiki/Šum.

13. NI PXI-7854R. *NI.* [Online] National Instruments. http://sine.ni.com/nips/cds/view/p/lang/cs/nid/206324.

14. Martínez, Daniel. *Writing Touch Sensing Software - Aplication Note.* [PDF] Guadalajara, Mexico : Freescale Semiconductor, Inc., 2011.

15. Freescale. *Touch Sensing software - API Reference Manual.* 08 2013.

16. Google Chart. *Google Developers.* [Online] Google. https://developers.google.com/chart/.

17. JavaScript Object Notation. *Wikipedie, otevřená encyklopedie.* [Online] Wikimedia Project, MediaWiki. http://cs.wikipedia.org/wiki/JavaScript_Object_Notation.

18. Decibel. *Wikipedie, otevřená encyklopedie.* [Online] Wikimedia Project, MediaWiki. http://cs.wikipedia.org/wiki/Decibel.

Seznam elektronických příloh

1. **TS014.html** – Příklad testovací zpráva testující ovládací prvek Keypad
2. **keypad_4.json** – Příklad předpisu pro testování ovládacího prvku Keypad se čtyřmi elektrody