VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

TESTOVACÍ NÁSTROJ PRO PLATFORMU FITKIT3

THE FITKIT3 TESTER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **PETR STEHLÍK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MICHAL BIDLO, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Stehlík Petr**
Program: Informační technologie
Název: **Testovací nástroj pro platformu FITkit3**
The FITkit3 Tester
Kategorie: Vestavěné systémy

Zadání:

1. Seznamte se s výukovou platformou FITkit3, jejím HW rozhraním a s principy diagnostiky a testování této kategorie systémů.
2. Po dohodě s vedoucím práce zvolte množinu prvků a periferií platformy FITkit3 a pro tyto navrhněte vhodné sady testovacích postupů (včetně potřebné HW podpory).
3. Sady testů, navržené v bodu 2, implementujte a integrujte je do systému s důrazem na uživatelskou přívětivost a efektivitu.
4. Formou technické zprávy pečlivě zdokumentujte navržené řešení.
5. Zhodnoťte dosažené výsledky a diskutujte možnosti pokračování projektu.

Literatura:

- Dle pokynů vedoucího projektu.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání, demonstrace prototypu systému z bodu 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bidlo Michal, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 25. října 2019

Abstrakt

Tato práce řeší problém testování určité množiny modulů mikrokontroléru ARM Kinetis Cortex-M4 (MK60DN512ZVMD10) na výukové platformě FITkit3 (známé také jako Minerva). Konkrétně se zabývá návrhem testovacího firmwaru v jazyce C pro automatizované testování, návrhem hardwarového modulu pro testování vstupně/výstupních rozhraní platformy s využitím integrovaného obvodu MCP23S17 a návrhem obslužného softwaru s textovým uživatelským rozhraním ve skriptovacím jazyce Python3 pro ovládání testování z PC. Vybranou testovací množinou modulů jsou časovače periodického přerušení (PIT), časovač s nízkou spotřebou (LPTMR) a hodiny reálného času (RTC), dále moduly pro sériový přenos dat skrze UART a SPI rozhraní, reproduktor a GPIO porty. Vytvořené řešení poskytuje komplexní nástroj pro analýzu funkčnosti často využívaných modulů při programování na platformě FITkit3, který samotný proces analýzy téměř celý automatizuje. Hlavním přínosem této práce je zejména usnadnění testování a rychlé odhalení chyb na velkém počtu výukových kitů.

Abstract

This work solves the problem of testing a certain set of ARM Kinetis Cortex-M4 (MK60DN512ZVMD10) microcontroller modules on the FITkit3 learning platform (also known as Minerva). Specifically, it deals with the design of test firmware in the C programming language for automated testing, the design of a hardware module for testing platform input/output interfaces using the MCP23S17 integrated circuit and the design of a software with text-based user interface in Python3 for the testing control from a PC. The selected test set of modules are Periodic interrupt timers (PIT), Low-power timer (LPTMR) and Real time clock (RTC), further modules for serial data transmission via UART and SPI interfaces, speaker and GPIO ports. The created solution provides a comprehensive tool for analysing the functionality of frequently used modules on FITkit3, which automates the analysis process almost completely. The main benefit of this work is the facilitation of testing and a rapid detection of errors in a large number of kits.

Klíčová slova

automatizované testovaní, FITkit3, testování, mikrokontrolér, vestavěný systém, ARM Cortex-M4, Kinetis K60

Keywords

automated testing, FITkit3, testing, microcontroller, embedded system, ARM Cortex-M4, Kinetis K60

Citace

STEHLÍK, Petr. *Testovací nástroj pro platformu FITkit3*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Bidlo, Ph.D.

Testovací nástroj pro platformu FITkit3

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Bidla, Ph.D a uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Petr Stehlík
4. června 2020

Poděkování

Děkuji vedoucímu práce, že mi umožnil psát bakalářskou práci pod svým vedením a vždy mi s velmi rychlou odezvou poskytoval kvalitní odbornou pomoc i užitečné rady. Také děkuji tatínkovi za předání praktických zkušeností a odborných rad během práce.
Děkuji moc i mamince, dědovi, babičkám a všem mým blízkým za veškerou pomoc a podporu při studiu, protože bez Vás by tato práce nikdy vzniknout nemohla.

Obsah

Kapitola 1

Úvod

Elektronická zařízení nás v dnešním světě provázejí téměř na každém kroku. Uvnitř většiny takových zařízení se zpravidla nachází mikrokontrolér – malá, avšak velmi podstatná univerzální součástka, která řídí jejich činnost. Kvůli rapidně narůstajícímu množství vyráběných kusů je téměř nemožné dodržet 100% přesnost a bezchybovost, zvláště pokud je výroba nějak časově nebo finančně omezena.

Jedním ze způsobů, jak redukovat tento nepříznivý jev, je použít různé techniky zajištění odolnosti systému vůči chybám již ve fázi návrhu. V řadě případů však chyby plynou přímo z nedokonalostí výrobních technologií, přičemž tento typ chyb nelze odhalit dříve než po dokončení výrobku. Je tak nutné zařadit další fázi před uvedením na trh – výstupní kontrolu. Snahou je pak snížení celkových nákladů na výrobu a zajištění (pokud možno) co nejefektivnějšího procesu odhalování chyb v rámci výstupní kontroly. Typickým způsobem může být využití prostředků automatizace.

Motivace k této práci vznikla na základě odhalení nedokonalého provedení výstupní kontroly u výukové platformy FITkit3. Proto cílem této bakalářské práce je vytvořit univerzální testovací nástroj pro testování výukových kitů, který bude snadno rozšiřitelný a použitelný. Na základě seznámení se s výukovou platformou FITkit3 jsou navrženy optimální postupy a principy, které tvoří další úroveň výstupní kontroly pro odhalení chybných modulů či jiných periferií.

Představením použité platformy se zabývá kapitola 2. V jejím rámci je vysvětlena základní výbava včetně popisu, detailně rozebrán osazený hlavní mikrokontrolér, jenž je předmětem testování, a stručně shrnuta výbava ostatní.

O pojetí zadání a návrhu řešení pojednává kapitola 3. Z hlediska lepší orientace v textu je rozčleněna do 3 sekcí, které se vztahují k návrhu řešení pro testovací firmware, dále pro podpůrný hardware a nakonec pro obslužný software. Každá sekce se věnuje zvlášť samostatné komponentě, jež dohromady tvoří ucelenou testovací sadu.

Hlavní částí této práce je kapitola 4 pojednávající o implementačních detailech. Taktéž je rozčleněna do 3 základních sekcí jako kapitola související s návrhem řešení. V rámci obsahu jsou objasněny složité, problémové i zajímavé programové části doplněné o ukázky ze zdrojových kódů.

K závěru bakalářské práce je umístěna kapitola 5 zabývající se zhodnocením vypracovaného řešení. Na začátku kapitoly se nachází obrázek sestaveného výsledného prototypu podpůrného hardwarového modulu. Kromě závěrečné diskuze je součástí obsahu také ukázka průběhu kompletních testů včetně jejich popisu.

Kapitola 2

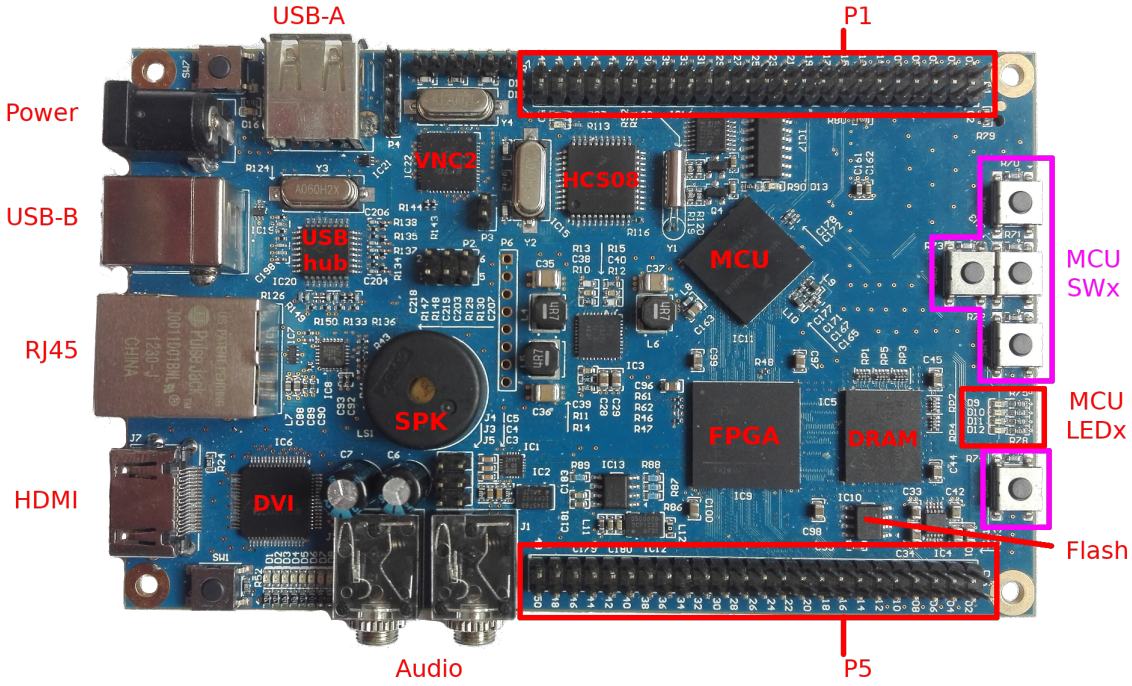
Popis platformy FITkit3

Není-li uvedeno jinak, následující popis platformy vychází z dokumentace a informací dostupných na oficiálním webu projektu[3]. FITkit3 vznikl jako výuková a experimentální platforma na Fakultě informačních technologií Vysokého učení technického v Brně. Jedná se o samostatný hardware s výkonným mikrokontrolérem (dále označovaným zkratkou MCU) a programovatelným hradlovým polem. Obsahuje celou řadu dalších modulů a periferií, které jsou v rámci této kapitoly detailněji popsány níže. Díky vhodné kombinaci hardwaru lze v praxi tento produkt využít v širokém spektru oblastí, od primitivního zaznamenávání naměřených hodnot z různých čidel přes samočinné řízení systémů až například po klasifikaci a rozpoznávání zvuku či obrazu z kamery v reálném čase[1].

2.1 Základní hardwarové vybavení výukového kitu

V odrážkovém seznamu níže jsou uvedeny základní komponenty výukového kitu včetně jejich charakteristiky. Jedná se buď o vybavení, kterým je FITkit3 výrazně specifický, anebo o vybavení, které bylo využito při tvorbě této práce. Pro ilustraci je na následující stránce přiložen obrázek 2.1 s vyznačením většiny popisovaných komponent (nejen) v této sekci. Seznam součástek včetně jejich schémat zapojení je taktéž dostupný na oficiálním webu kitu[3], odkud bylo při vytváření popisu čerpáno.

[1] Mnoho příkladových aplikací je uvedeno také na oficiálním webu[3].



Obrázek 2.1: **Základní deska FITkitu3 s vyznačenými komponentami.** Podstatnými prvky na této desce jsou FPGA, MCU hlavní a vedlejší (HCS08), paměti typu DRAM a Flash, USB hub, USB převodník VNC2 a DVI řadič zobrazení. Na obrázku lze nalézt vyznačená tlačítka (SWx), indikační LED (LEDx), reproduktor (SPK) a vstupně/výstupní porty pod označením P1 a P5. Po stranách desky jsou přítomné konektory pro externí napájení (Power), Ethernet (RJ45), HDMI, 2 audio konektory typu Jack a USB typu A i B, přičemž typ B slouží pro připojení k PC.

Seznam základních komponent a jejich charakteristika

- **FPGA**[2] – Společnost Xilinx, která se zabývá výrobou FPGA, jej popisuje [17] jako polovodičovou součástku založenou na matici konfigurovatelných logických bloků, jejichž propojení mezi sebou lze přeprogramovat k žádoucímu účelu či funkčním požadavkům až po výrobním procesu. Díky této vlastnosti se FPGA liší od aplikačně závislých obvodů, které jsou běžně vyráběny jen pro konkrétní účely. Ačkoliv existují i jednorázově programovatelné hradlové pole, častějším typem jsou opakovaně programovatelné hradlové pole založené na paměti SRAM. Jedním z nich je i FPGA Xilinx Spartan 6 (XC6SLX9), jež je osazené na desce platformy FITkit3. Hlavní výhodou v praxi je zejména snížení nákladů a urychlení procesu vývoje složitých číslicových obvodů, přičemž jedním z jazyků pro popis hardwaru, který se používá pro programovatelné hradlové pole, je jazyk VHDL[3].

- **Hlavní MCU** – Součástí výbavy je také výkonný mikrokontrolér od společnosti Freescale Semiconductor s označením Kinetis K60 (MK60DN512ZVMD10). O detailnějším popisu pojednává sekce 2.2.

[2] FPGA – programovatelné hradlové pole (anglicky Field-Programmable Gate Array)
[3] https://cs.wikipedia.org/wiki/VHDL

- **Paměť DDR2 SDRAM**[4] – Na platformě FITkit3 je k FPGA připojen mimo jiné paměťový modul sloužící k dočasnému uchování dat za běhu. Sestaven je z paměti DDR2 SDRAM (IS43DR16320B) od firmy ISSI o kapacitě 512Mb (tj. 64MB) a pracující na frekvenci 333MHz. Tento typ paměti je specifický především dvojnásobně vyšším výkonem při přenosu dat než její předchůdce – paměť SDRAM, jelikož k přenosu dochází jak na vzestupné, tak na sestupné hraně řídicího hodinového signálu.

- **Paměť Flash** – Stálá elektricky programovatelná (zapisovatelná) paměť uchovávající informace i bez přítomnosti napájení (tzv. nevolatilní). Je vnitřně organizována po blocích, přičemž každý blok lze programovat samostatně (obsah ostatních bloků je zachován). Na výukovém kitu se používá jako paměť typu ROM[5] pro uložení konfigurace FPGA. Výhodou této paměti je, že ji lze snadno znovu naprogramovat bez vyjmutí ze zařízení a s použitím minimálního počtu pomocných obvodů. Osazená paměť typu NAND Flash 25MHz (M25P40) od výrobce STMicroelectronics s kapacitou 4Mb (512KB) komunikuje skrze 8 bitovou sběrnici přes rozhraní SPI [14].

- **Převodník rozhraní USB/COM** – Zpřístupnění některých komponent platformy FITkit3 na rozhraní USB vyžaduje použití podpůrného převodníku. Jedná se o produkt FTDI Vinculum-II (VNC2-48L1B) společnosti Future Technology Devices International Ltd. (FTDI) založený na vlastní čipové sadě. Použitý mikrokontrolér navržený podle Harvardskou architekturou je 16 bitový. Kapacita jeho vnitřní flash paměti je 256KB a paměti typu RAM je 16KB. Podporuje převod mezi rozhraním SPI či UART a rozhraním USB 2.0, díky čemuž je například zpřístupněn sériový port hlavního MCU přes rozhraní USB v počítači. Výhodou jsou dostupné ovladače pro všechny základní operační systémy.

- **Vedlejší MCU** – Pro ladění ostatních komponent je na platformě FITkit3 osazen druhý mikrokontrolér MC9S08 (MC9S08JM60) taktéž od společnosti Freescale Semiconductor [8]. Vyznačuje se 8 bitovou sběrnicí, pamětí flash o kapacitě 60KB pro program a pamětí RAM s kapacitou 4KB. Podporuje mimo jiné rozhraní I^2C, SPI, UART a USB. Jádro mikrokontroléru pohání procesor z rodiny HCS08 s taktem o frekvenci 48MHz.

- **USB hub se 4 porty** – Jedná se o rozbočovač, který umožňuje připojení více zařízení s rozhraním USB na jediný USB port. Na výukovém kitu se nachází USB hub od firmy Texas Instruments (TUSB2046B) se 4 rozšiřujícími porty. Podle schématu zapojení (dostupného na oficiálním webu [3]) jsou rozšiřující porty využity pouze 3, a to k připojení ladění a vstupně/výstupní komunikace MCU a také k připojení převodníku rozhraní USB/COM. Zmíněný USB hub podporuje rozhraní USB 2.0 s přenosovou rychlostí až 12Mbit/s.

- **Řadič napájení** – Jelikož je na desce výukového kitu spousta součástek, z nichž každá má různou toleranci vstupního napájení, je nutný také řadič napájení pro snížení a stabilizaci několika předvolených napěťových úrovní. Na FITkitu3 je osazen pulzní měnič od společnosti Texas Instruments (TPS65251RHA) [16].

- **Konektor USB typ B** – Využívá se k propojení rozhraní USB s PC. Slouží také jako zdroj napájení pro výukový kit, pokud není připojeno jiné externí napájení.

[4] DDR SDRAM – anglicky Double Data Rate Synchronous Dynamic Random Access Memory
Více informací na https://cs.wikipedia.org/wiki/DDR_SDRAM .

[5] ROM – Paměť určená pouze ke čtení (anglicky Read Only Memory).

- **Rozšiřující konektory (anglicky pinheaders)** – Po stranách desky lze najít 2 sady dvouřadých konektorů, z nichž každý obsahuje celkem 50 pinů. Jsou označeny jako `P5` a `P1` a jejich detailnější popis lze nalézt v sekci 2.2.
- **Reproduktor** – Na současných deskách je osazen reproduktor od firmy TDK. Jeho vstup je přiveden na vstupně/výstupní rozhraní MCU k pinu 4 na portu A. Pro vydání tónu jej lze budit například obdélníkovým signálem o potřebné frekvenci.
- **Sada ovládacích tlačítek** – Z celkových 7 tlačítek na desce je 5 z nich přivedených na vstupně/výstupní rozhraní mikrokontroléru. Ve schématu k FITkitu3 se nachází pod označením `SW2-6` a jsou přivedeny na piny portu E. Při stisku generují logickou úroveň 0, jelikož se nacházejí v zapojení s pull-up[6] rezistorem.
- **Sada indikačních LED** – Celkový počet indikačních LED pro MCU je 4 a na desce je lze najít pod označením `D9-12`. Jejich napájení lze ovládat skrze vstupně/výstupní piny na portu B. Aktivují se logickou úrovní 0.

2.2 Mikrokontrolér Freescale Kinetis K60

Jedná se o procesor založený na architektuře ARM Cortex-M4 pod konkrétním označením `MK60DN512ZVMD10`. Výrobcem osazeného mikrokontroléru řady Kinetis K60 byla společnost Freescale Semiconductor, která se na konci roku 2015 sloučila se společností NXP Semiconductors [12].

Následující informace a popis vychází z oficiálního datasheetu [7] a manuálu [9].

Architektura ARM Cortex-M4

Procesor s jádrem ARM Cortex-M4 je postaven na Harvardské architektuře ARMv7 a Thumb®-2 ISA. Je také dopředně kompatibilní s architekturami Cortex M3, Cortex M1 i Cortex M0. Jedná se o speciální procesor s přidanou podporou DSP[7] 32 bitových instrukcí, které se slouží ke zpracování digitálních signálů. Součástí je také rozsáhlá podpora ladicích rozhraní jako JTAG[8], SWD[9] a také cJTAG.

Provozovat procesor je podle manuálu možné na maximální frekvenci 100MHz, přičemž sběrnice s ostatními komponentami pracuje na poloviční rychlosti. Součástí čipu je dále paměť typu flash pro program o kapacitě 512KB a také paměť typu RAM o kapacitě 128KB. Napájecí napětí by se mělo pohybovat v rozmezí od 1.71V do 3.6V.

Mikrokontrolér disponuje také nastavitelnými 16 úrovněmi priorit přerušení (modul NVIC[10]), ochranou proti zacyklení běhu programu (modul Watchdog) a modulem chránícím nestabilitu napájecího napětí (LLWU[11]).

K dalším výhodám patří podpora paralelního rozhraní FlexBus a sériového rozhraní EzPort, které se využívá pro programování pamětí typu flash.

[6] Zapojení s pull-up rezistorem slouží k pevně definované logické úrovni 1 na vstupním pinu MCU za předpokladu, že zdrojový přívod má nekonečný odpor (tlačítko není stlačené apod.).

[7] DSP – Digital signal processor (viz https://en.wikipedia.org/wiki/Digital_signal_processor).

[8] JTAG – Joint Test Action Group (viz https://en.wikipedia.org/wiki/JTAG).

[9] SWD – Serial Wire Debug (viz https://wiki.segger.com/SWD).

[10] NVIC – Nested Vectored Interrupt Controller. Jedná se o řadič přerušení.

[11] LLWU – Low-leakage wakeup unit. Jedná se o speciální modul, který umí probudit procesor z režimu spánku při nestabilitě napájení.

Periferní výbava mikrokontroléru

Mezi periferní výbavu se řadí mimo jiné analogově/digitálních moduly, kterými jsou:

- 2x 16 bitové SAR analogově/digitální převodníky,
- programovatelný zesilovač (PGA) (až 64 krát), který je součástí každého analogově/-digitálního převodníku,
- 2x 12 bitové digitálně/analogové převodníky,
- 3x analogové komparátory (CMP) obsahující 6 bitové digitálně/analogové převodníky a programovatelný referenční vstup.

Dalšími užitečnými moduly jsou časovače. Následuje jejich výčet:

- 2-8 kanálový časovač FlexTimer (FTM) s podporou generování pulzní šířkové modulace signálu,
- Programovatelný zpožďovací blok (PDB),
- podpora pro časovou synchronizaci podle standardu IEEE 1588,
- 4x časovače s generátorem periodického přerušení (PIT),
- 16 bitový časovač s nízkou spotřebou (LPTMR),
- vysílač s nastavitelným časováním přenosu (CMT),
- hodiny reálného času (RTC).

Rozhraní mikrokontroléru

Osazený mikrokontrolér podporuje především dále zmíněné rozhraní:

- 2x modul s vestavěnou podporou CAN (Controller Area Network) protokolu,
- 2x sběrnici I^2C (Inter-Integrated Circuit),
- 3x rozhraní SPI (Serial Peripheral Interface),
- 6x modul UART k asynchronnímu sériovému přenosu dat,
- USB (Universal Serial Bus) včetně podpory OTG specifikace,
- Ethernet s hardwarovou podporou standardu IEEE 1588,
- I^2S pro přenos digitálního zvuku,
- SDHC (Secure Digital host controller) pro možnost připojení SD karty.

Rozhraní pro připojení externích modulů

Kromě již zmíněných rozhraní má osazený mikrokontrolér také řadu pinů, které lze programově přepnout do módu univerzálních vstupně/výstupních digitálních pinů (GPIO) a následně obsluhovat. Podle schématu zapojení (dostupného na oficiálních stránkách kitu) jsou tyto piny vyvedeny na konektor s označením P1. Společně s GPIO piny jsou vyvedeny na konektor i rozhraní SPI (konkrétně modul pod označením `SPI2`), I^2C (modul označený jako `I2C0`) a také vstupy/výstupy analogově/digitálních převodníků.

Mezi piny, které lze používat v režimu GPIO, patří piny s označením PTA6-11, PTA24-29 a PTE28. Mikrokontrolér má své vývody rozdělené do sekcí, které jsou nazvány jako port A až E. K řadě těmto pinům je přivedeno také rozhraní Flexbus.

Součástí desky výukového kitu je dále konektor označený jako P5. Zde jsou vyvedeny zase všechny vnější piny programovatelného hradlového pole (FPGA). Dohromady tak oba konektory (P1 a P5) shlukují všechny dostupné rozhraní na FITkitu3, které lze použít k připojení externích modulů.

2.3 Další hardwarová výbava na platformě FITkit3

Aby byla uvedená výbava platformy kompletní, jsou zde uvedeny zbývající rozhraní, řadiče a jiné součástky, které jsou taktéž součástí kitu, avšak tato bakalářská práce se jimi nezabývá.

Seznam dalších integrovaných obvodů a rozhraní

- Stereo audio kodek od společnosti Freescale Semiconductor (model SGTL5000)
- Rozhraní pro zobrazení DVI od společnosti Texas Instruments (model TFP410PAP)
- LAN řadič pro Ethernet od společnosti SMSC (model LAN8720A)

Seznam ostatních součástek

- Konektor pro externí napájení (Power)
- HDMI konektor k DVI řadiči
- Konektor RJ45 pro Ethernet
- Konektor USB typu A pro připojení externího zařízení
- Slot pro SD kartu (nachází se na spodní straně kitu)
- Vstupní a výstupní konektor pro audio
- Oddělovače a linkové budiče (model SN74HCT125D)
- Sada indikačních LED pro FPGA nacházející se mezi audio konektory a DVI řadičem
- Ovládací tlačítko pro FPGA pod označením SW7 na plošném spoji (umístěno vedle indikačních LED pro FPGA)

Kapitola 3

Návrh řešení pro testovaní platformy FITkit3

Jak již bylo naznačeno v úvodu práce, v současné době neexistuje pro platformu FITkit3 komplexní testovací nástroj. V rámci této bakalářské práce je návrh zaměřen na vytvoření základních stavebních bloků pro testování osazeného mikrokontroléru a jeho periferií. Následně jsou navrženy (viz sekce 3.1) základní testovací sady vybraných modulů dle požadavků zadání. K testování vstupně/výstupních portů a komunikačního modulu pro SPI rozhraní musel být navržen přídavný hardware. O jeho návrhu a principu funkcionality pojednává sekce 3.2. Aby nebyl proces testování příliš složitý a pokud možno při velkém počtu testovaných zařízení i pomalý, byl navržen obslužný software detailněji popsaný v sekci 3.3.

V diplomové práci [2] zaměřené nejen na testování hardwaru bylo vysvětleno, že postupným neustálým zmenšováním výrobní velikosti součástek jsou do procesu výroby zanášeny defekty. Stačí, aby byl v mikrokontroléru vadný jen jediný tranzistor či vodič (například v důsledku použitého vadného materiálu nebo poškozené masky při výrobě) a výsledná součástka nemusí fungovat podle očekávání. Jedná se zároveň o chyby, které je možné odhalit až po výrobě v rámci výstupní kontroly mimo jiné také softwarovými testy. Navržené řešení se zabývá principy diagnostiky této kategorie testování.

Všechny grafy obsažené v kapitole byly nakresleny pomocí webového nástroje draw.io.

3.1 Testovaní modulů a funkcí mikrokontroléru

Na základě zadání práce byla po dohodě s vedoucím zvolena následující množina prvků:

- Hodiny reálného času (modul RTC),
- časovač s nízkou spotřebou (modul LPTMR),
- časovače periodicky generující přerušení (modul PIT),
- vstupně/výstupní GPIO porty (viz 3.2),
- komunikační moduly skrze rozhraní sériového portu (UART) a rozhraní SPI,
- osazené tlačítka a sada indikačních LED a
- reproduktor.

Při volbě byl kladen důraz na periferie, které lze obsluhovat pomocí mikrokontroléru.

Testování časovačů

V rámci testování jsou pro každý časovač vybrány jeho specifické vlastnosti podle dokumentace [9].

Hodiny reálného času se vyznačují každou sekundu se zvyšující hodnotou v čítači sekund. Jestliže by byl modul poškozen (například nefunkční 32KHz oscilátor či samotný čítač), musí v rámci testování dojít k detekci takové vady. Zvolenou možností detekce bude algoritmus pravidelně kontrolující hodnotu čítače a jeho rychlost inkrementace.

Časovač s nízkou spotřebou je specifický svou schopností detekovat náběžnou či sestupnou hranu vstupního signálu a při její detekci zvýšit hodnotu vnitřního čítače. Následně v porovnávacím režimu, který bude otestován, kontroluje pravidelně tuto hodnotu s programově přednastavenou hodnotou v řídicím registru. Pokud dojde k překročení meze, je vyvoláno přerušení a vnitřní čítač vynulován. Dochází tak k pravidelné detekci určitého počtu kmitů. Tato vlastnost lze využít například pro generování signálu s pulzní šířkovou modulací. Způsob testování bude obdobný jako u hodin reálného času – po spuštění časovače bude hlídací algoritmus pravidelně kontrolovat předpokládanou zvyšující se hodnotu vnitřního čítače a očekávat vyvolání přerušení. V případě funkčního modulu bude vnitřní časovač pravidelně automaticky inkrementován.

Poslední zmíněný časovač má tu vlastnost, že periodicky generuje přerušení v pravidelně zvoleném rytmu. Součástí výukového kitu jsou podle dokumentace celkem 4 tyto moduly, které navíc podporují zřetězený režim (anglicky chain mode). V rámci jejich testování bude nejprve otestována vlastnost generování pravidelného přerušení pro každý časovač zvlášť a následně dojde k propojení vždy dvou sousedních čítačů do párového režimu, což znamená, že výše postavený čítač sníží svou hodnotu až na základě vyvolaného přerušení nižším čítačem. Výjimka těchto časovačů oproti ostatním je v tom, že na počátku je do jejich vnitřního čítače programově nastavena finální hodnota a poté je tato hodnota postupně snižovaná až na 0. V tu chvíli je vyvoláno přerušení a počáteční hodnota v registru čítače je obnovena.

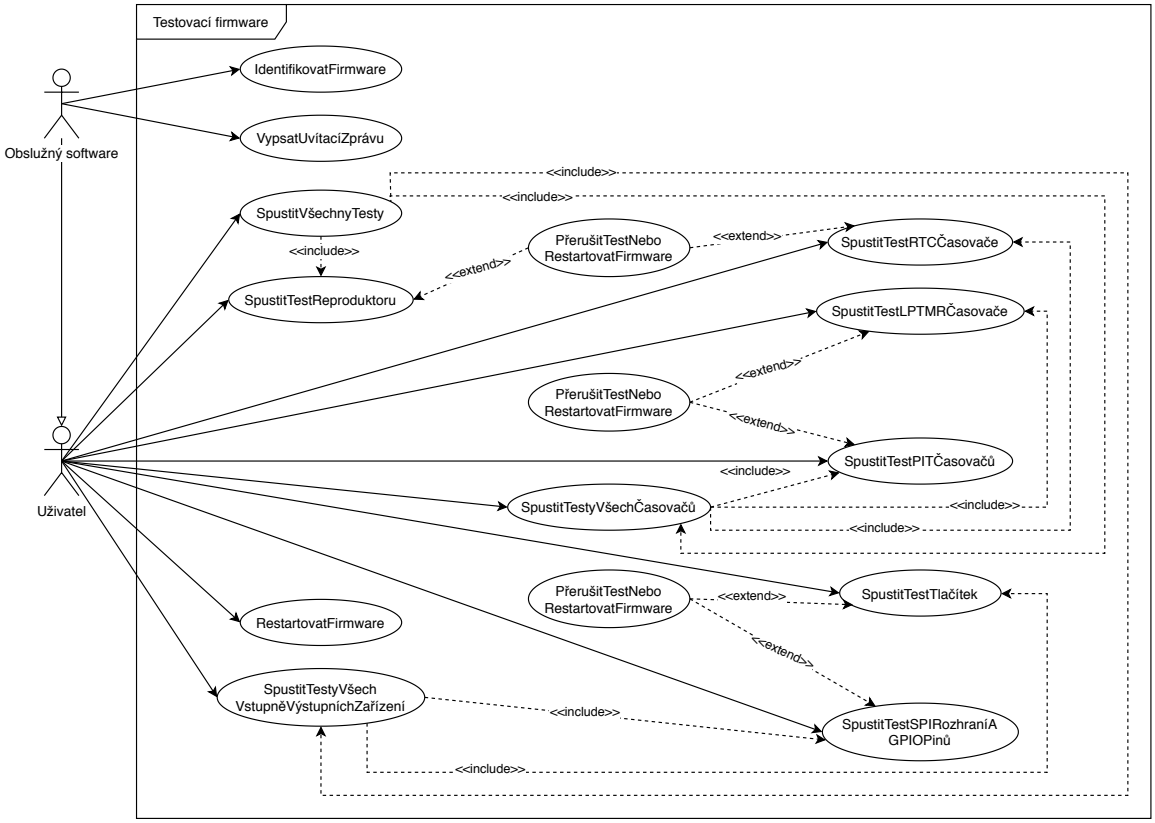
Testování komunikačních modulů

Samotné testování komunikačních modulů je zahrnuto již v rámci jejich využití. Jistě by bylo možné testovat sadu konkrétních nastavení z hlediska různých rychlostí, detekce paritních bitů a podobně. V rámci této práce však postačí otestovat základní funkcionalitu modulu, tedy zdali úspěšně komunikuje s externě připojeným zařízením.

Avšak součástí testování musí být určitě zahrnuta aktivace modulu včetně povolení generování přerušení, přednastavení komunikačních parametrů a následná kontrola, že byl modul aktivován a funkční je jak přijímač, tak i vysílač jednotlivých datových slov.

Ovládání a uživatelské rozhraní

Komunikaci s uživatelským obslužným terminálem bude zajišťovat modul UART, přičemž použitým rozhraním tak bude sériový port a klasické textové prostředí. Pro ovládání testovacího firmwaru se jeví nejsnazší možnost provedení skrze jednoduché textové menu, kde jednotlivé volby budou souviset se spouštěním určité sady testů. Následující obrázek 3.1 ukazuje typické případy užití testovacího firmwaru, které bude vhodné implementovat v rámci obsluhy komunikace. Restartování firmwaru či přerušení testů by mělo být umožněno v jakékoliv fázi.



Obrázek 3.1: **Diagram případů užití testovacího firmwaru.** Uživatel má možnost spustit 1 z 6 různých testovacích sad, případně některé jejich kombinace, které slouží k postupnému samočinnému spuštění dílčích testů. Průběh kteréhokoli testu může v libovolnou chvíli přerušit. Obdobně může kdykoliv restartovat zařízení, aby vynuloval výsledky a nastavení již proběhlých testů. Testy lze spouštět v libovolném pořadí a opakovat donekonečna. Obslužný software má stejné schopnosti, ale navíc může požádat testovací firmware o identifikaci či vypsání uvítací zprávy.

3.2 Testovaní fyzického rozhraní výukového kitu

Součástí testování fyzického rozhraní výukového kitu je otestovat funkci vstupně/výstupní GPIO pinů mikrokontroléru. Podle diplomové práce [2] se tato problematika nazývá kontrola důvěryhodnosti. V principu se jedná o kontrolu nepřípustných stavů vstupů a výstupů, tudíž postupnou kontrolu každého pinu na jeho očekávanou hodnotu. Přičemž toto lze provést na dvou různých místech – testovat přímo hodnotu pinu mikrokontrolérem, anebo použít externí součástku, která otestuje pin zvnějšku.

Jiným způsobem testování je využít pull-up či pull-down rezistory. Diplomová práce uvádí, že by při výrobě mohlo dojít ke zkratu k zemi nebo naopak ke zkratu k napájení. Pro pin s povoleným pull-down rezistorem je očekávaná logická hodnota 0. Oproti tomu pro pin s povoleným pull-up rezistorem se očekává detekce logické hodnoty 1. Pokud by došlo k poruše při výrobě, popsaným způsobem testování by ji bylo možné odhalit.

V rámci bakalářské práce je testování fyzického rozhraní zaměřeno na použití podpůrného externího modulu, díky kterému bude možné otestovat očekávané hodnoty pinů z obou stran (jak z mikrokontroléru, tak z externího modulu). K tomuto účelu byl vybrán integro-

vaný obvod `MCP23S17`, přičemž v jeho dokumentaci [11] je uvedeno, že komunikuje skrze rozhraní SPI a podporuje až 16 GPIO pinů. Jedná se o rozšiřující modul pro případ, kdy zařízení nemá dostatek dostupných GPIO pinů. Zde bude jeho účel upraven.

Součástí implementace je tedy potřeba navrhnout zapojení s vybraných integrovaným obvodem tak, aby testovací sada byla schopna pokrýt obousměrný cyklický způsob zjišťování hodnot všech 13 GPIO pinů mikrokontroléru. Ideálním způsobem je propojit GPIO piny mikrokontroléru s GPIO piny rozšiřujícího integrovaného obvodu. Pro testování vstupního režimu pinů mikrokontroléru pak bude skrze externí modul nastavena cyklicky logická úroveň 1 vždy pouze na jediném pinu. Obdobně bude probíhat testování výstupního režimu pinů mikrokontroléru s tím rozdílem, že logickou úroveň jedna bude tentokrát nastavovat mikrokontrolér a čtena bude externím modulem. Na všech ostatních pinech přitom po celou dobu testování musí být logická hodnota 0.

Při komunikaci mikrokontroléru s externím modulem bude zároveň otestován modul pro rozhraní SPI v mikrokontroléru. Jedinou nevýhodou je, že bez dalších součástek nelze určit, zdali v případě poruchy rozhraní SPI je nefunkční komunikační modul, nebo podpůrná hardwarová deska.

Současně z hlediska samočinného testování by bylo vhodné zařadit snímač svítivosti sady indikačních LED osazených na výukovém kitu. Zvolenou součástkou postačí fotorezistor, který bude vhodné umístit do zapojení odporového děliče mezi kontakt napájení a vstupní GPIO pin podpůrného integrovaného obvodu. K odstínění okolních světelných rušivých vlivů by bylo vhodné vytvořit také malou rouru, který by zatemnila fotorezistor a zvýšila tak jeho odpor. V rámci hardwarového modulu by byla použita k přiklopení sady indikačních LED na desce FITkitu3.

3.3 Automatizace průběhu testovaní

Vzhledem k zadání a cíli práce, které jsou vymyšleny pro testování velkého množství výukových kitů, by bylo vhodné, kdyby průběh celého testování, od připojení FITkitu3 k obslužnému počítači přes jeho otestování až po následné odpojení, byl nějakým způsobem automatizován. Nebo alespoň co nejvíce zjednodušen. Na místě je tedy navrhnout obslužný software.

Shrnutí požadavků

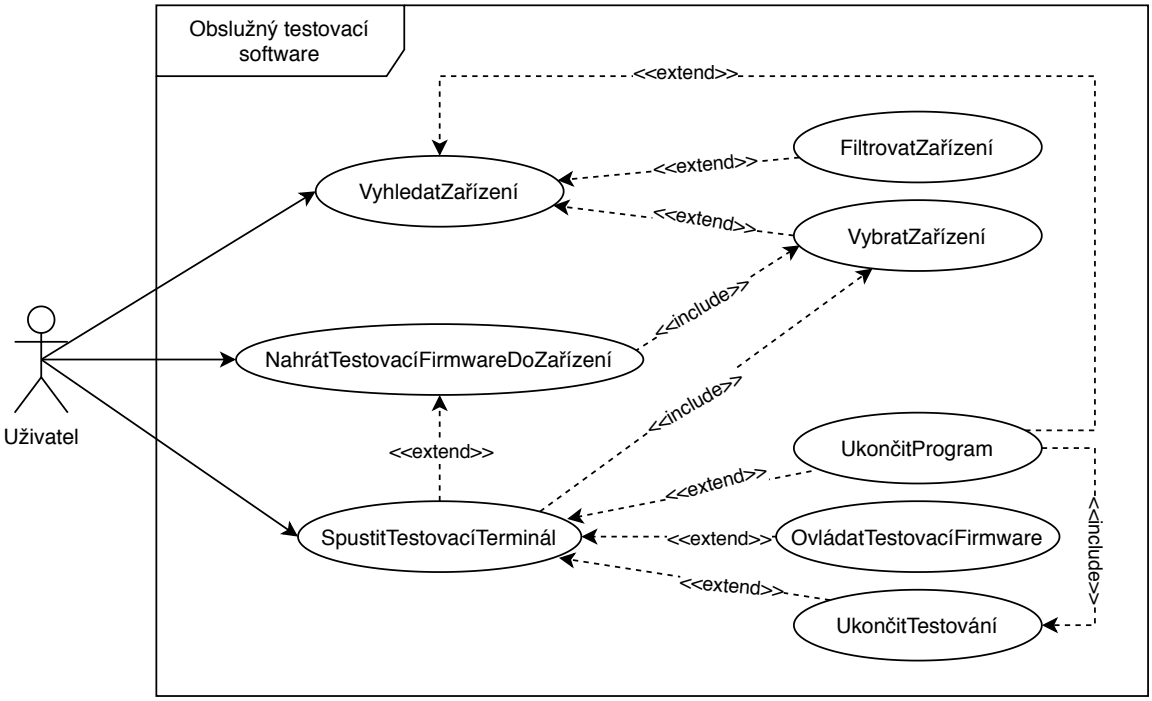
Mezi základní požadavky bude patřit přenositelnost (nezávislost na platformě či operačním systému), snadná ovladatelnost (výstižné uživatelské rozhraní obsahující jen nezbytné prvky) a rychlost (čistý kód s minimálním počtem podpůrných knihoven). Jelikož je pro testování nutná dostupnost testovacího firmware ve výukovém kitu, součástí automatizace procesu bude také požadavek na jeho samočinné nahrávání do zařízení. Z hlediska většího množství potřebných podpůrných programů bude nejvhodnější tuto část vyčlenit do externího skriptu.

Obrázek 3.2 dále ukazuje typické případy užití obslužného softwaru. Lze vyvodit, že pro pokrytí všech případů užití budou stačit 3 základní obrazovky:

1. **Výběr zařízení** – například jednoduchý formulář obsahující seznam dostupných zařízení s možností filtrace.

2. **Nahrávání testovacího firmwaru** – okno pro obsluhu výstupu spuštěného vnějšího skriptu, které bude následovat po výběru zařízení v případě, že v něm nebude testovací firmware dostupný.

3. **Obslužný testovací terminál** – jiné okno s textovým terminálem, jenž bude zprostředkovávat vstupně/výstupní operace mezi uživatelem a testovaným zařízením. Součástí okna bude tlačítko pro ukončení testování či celého programu.



Obrázek 3.2: **Diagram případů užití obslužného testovacího softwaru.** Uživatel má na začátku možnost vyhledat dostupná zařízení a některé vybrat. Následně je detekována přítomnost testovacího firmwaru a v případě úspěchu spuštěn testovací terminál, který slouží k jeho obsluze. Naopak při neúspěchu je uživateli nabídnuta možnost automatického nahrání. Průběh testování lze kdykoliv ukončit.

Vhodné bude také uvažovat pár vyskakovacích oken s upozorněním nebo jinou zprávou pro uživatele v situacích kdy:

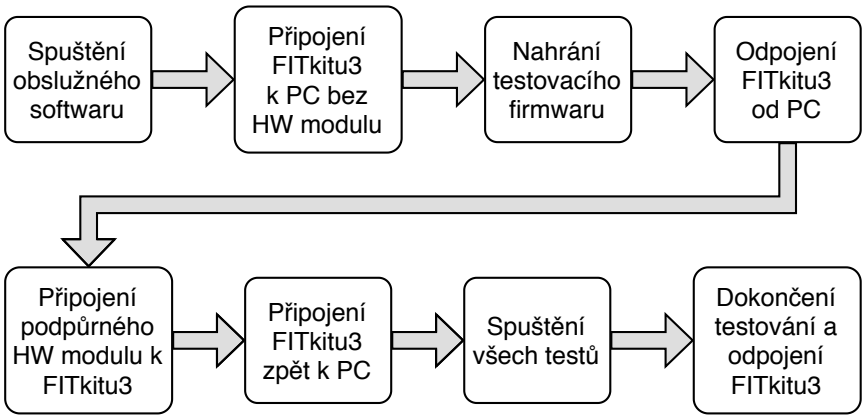
- v zařízení není dostupný testovací firmware,
- zařízení má nefunkční komunikační modul,
- nezdaří se navázat spojení vlivem operačního systému nebo jiného programu,
- během testování dojde k přerušení spojení,
- je přítomen automatický nahrávací skript a obslužný software má důvod jej použít.

Program dále musí umět samočinně spustit všechny testy a zobrazit uživateli jejich výsledky, vypořádat se s větším množstvím současně připojených výukových kitů, automaticky detekovat přítomnost testovacího firmwaru v zařízení a případně spustit externí skript pro jeho nahrání.

Výsledný návrh

Podle zmíněných požadavků bude vhodné použít skriptovací jazyk Python3[1], který usnadní přenositelnost programu. Z hlediska jednoduchého návrhu uživatelského rozhraní a rychlosti bude vhodné zůstat u textového prostředí. Pro tento účel existuje v jazyce Python3 knihovna `npyscreen`[2] využívající modul `curses`[3]. Výhodou jsou velmi nízké výpočetní i grafické nároky, protože obsah komunikace není nutné příliš upravovat (stačí jej jen zobrazit). Ke komunikaci skrze sériový port (s UART modulem na FITkitu3) bude potřeba knihovna `pySerial`[4].

Průběh celého automatizovaného procesu testování by pak ve finále mohl vypadat například tak, jak naznačuje obrázek 3.3.



Obrázek 3.3: **Blokový diagram prvního průběhu automatizovaného testovacího procesu.** Každý další průběh na stejném zařízení již nebude využívat fázi nahrávání testovacího firmwaru a s tím související připojení a odpojení bez rozšiřujícího hardwarového modulu.

Závěrečné finální shrnutí zvoleného prostředí a knihoven (včetně verzí) je následující:

- **Python** 3.6.9
- **npyscreen** 4.10.5
- **pySerial** 3.4

[1] https://www.python.org/
[2] http://www.npcole.com/npyscreen/
[3] https://docs.python.org/3/howto/curses.html
[4] https://pypi.org/project/pyserial/

Kapitola 4

Popis implementace a provedení navrženého řešení

Při popisu implementace a provedeného řešení na základě návrhu byl kladen důraz i na objasnění problémů, které nastaly nejen během programování. Jelikož se návrh zabývá 3 podstatnými komponentami tvořícími výsledný celek, implementace každé z těchto komponent byla popsána v samostatné sekci, přičemž sekce 4.1 popisuje tvorbu testovacího firmwaru pro MCU[1], sekce 4.2 zase pojednává o provedení podpůrného hardwarového modulu a poslední sekce 4.3 vysvětluje tvorbu obslužného softwaru.

Všechny zdrojové kódy programů i skriptů jsou zdokumentované pomocí speciální komentářové notace, jež rozumí program Doxygen[2] a umí tak vygenerovat programovou dokumentaci ve formátu HTML. Přednastavený konfigurační soubor `Doxyfile` se nachází ve složce se zdrojovými kódy testovacího firmwaru.

Adresářová struktura celého projektu je rozčleněna podle již zmíněných sekcí. Ve složce `Software` jsou umístěny zdrojové kódy pro obslužný program. Oproti tomu adresář `Firmware` obsahuje více podadresářů, z nichž podstatný je v tuto chvíli adresář `src`, ve kterém se nachází zdrojové kódy testovacího firmwaru. Schéma, návrh plošného spoje a soubory týkající se vytvořeného 3D modelu podpůrného hardwaru jsou obsaženy ve složce `Hardware`.

Veškeré zdrojové kódy a s nimi související přiložené soubory vytvořené v rámci této bakalářské práce (pokud není uvedeno jinak) jsou licencovány pod GNU General Public License verze 3[3]. V příslušných adresářích se vždy nachází soubor `COPYING` a vyjadřuje, že všechny obsažené soubory v konkrétní složce na stejné adresářové úrovni jsou pokryty licencí v něm popsanou. Zároveň každý takto pokrytý soubor obsahuje zpravidla úvodní licenční hlavičku. Díky tomu je zaručena ochrana svobody díla.

4.1 Testovací firmware pro mikrokontrolér

Testovací firmware byl převážně implementován v programovacím jazyce C, přičemž v minimu nutných případů musel být použit jazyk symbolických adres (assembler). Sekce popisuje principy použité pro testování jednotlivých periferií a také tvorbu skriptu `Makefile`

[1] MCU – Jedná se o zkratku pro mikrokontrolér.
[2] https://www.doxygen.nl/
[3] https://www.gnu.org/licenses/gpl-3.0.html

pro program `make`[4], který slouží k osvobození se od integrovaného vývojového prostředí KDS[5]. Během implementace byl využit manuál k testovanému mikrokontroléru [9].

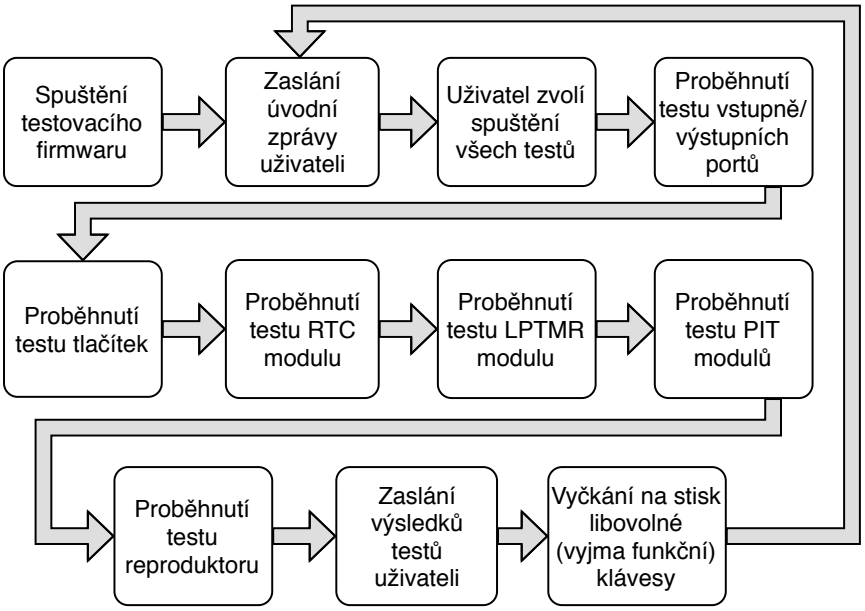
Vytvořené zdrojové kódy sice nepoužívají pro svůj chod prostředí KDS, ale využívají část zdrojových kódů (hlavičkové soubory pro definice registrů mikrokontroléru a kód zahajující jeho spuštění) od společnosti Freescale Semiconductors Incorporated. Na tento podpůrný obsah se vztahuje jiná licence, jež je popsána například v záhlaví týkajících se souborů. Zmíněné pomocné soubory byly na začátku vygenerovány vývojovým prostředím KDS a pro účely osvobození z něj byly vyjmuty.

Zobrazení výsledků testů i veškerá interakce s obslužným softwarem je provedena pomocí modulu UART, který zprostředkovává komunikaci skrze sériový port. Z toho důvodu byly implementovány funkce v souboru `UART.c` pro obsluhu vstupu a výstupu. Všechny ostatní testovací moduly využívají tyto funkce pro sdělování průběhu testů uživateli.

Implementace jednoduchého testu pro ověření funkčnosti osazeného reproduktoru na výukové platformě se nachází v souboru `speaker.c`. Výsledky tohoto druhu testu musí uživatel posoudit vždy sám, protože součástí řešení není vytvořená automatická detekce zvuku.

Hlavním jádrem testovacího firmwaru je nekonečný cyklus obsluhující vstup komunikace s uživatelem. Funkce `startTests()` v souboru `tests.c` je tak výchozím bodem pro spuštění libovolné testovací sady a stará se také o následné zaslání výsledků testů uživateli. V souboru `main.c` se nachází pouze funkce pro prvotní nastavení mikrokontroléru a rychlostí sběrnic.

Na obrázku 4.1 je možné shlédnout ilustraci toku programu při spuštění všech vytvořených testovacích sad. Obdobně vypadají i ostatní případy provedení jen určité skupiny testů. Za obrázkem následují podsekce s popisem implementace dílčích testovacích sad.



Obrázek 4.1: **Blokový diagram s názorným příkladem průběhu testování.** Znázorněn je jeden z nejpravděpodobnějších možných scénářů, který na základě volby uživatele spouští postupně za sebou všechny dostupné testovací sady.

[4] https://www.gnu.org/software/make/

[5] KDS – Kinetis Design Studio od společnosti NXP (viz https://www.nxp.com/design/designs/design-studio-integrated-development-environment-ide:KDS_IDE).

Průběh každého testování lze zvlášť přerušit zasláním znaku velkého písmene `C`. V případě jakékoliv potřeby je možné zasláním znaku velkého písmene `R` restartovat testovací firmware včetně celého mikrokontroléru, čímž se nastavení modulů vrátí do původního stavu. Tyto znaky jsou na obrázku 4.1 míněny jako funkční klávesy.

Všechny proměnné nebo funkce, které mají být použity pouze v rámci jednoho zdrojového souboru, jsou deklarovány jako statické klíčovým slovem `static`. Dále některé často používané proměnné například v rámci cyklů nebo `inline` funkcí jsou označeny klíčových slovem `register`, aby je mohl překladač lépe optimalizovat. Obdobně funkce, které jsou volány jen z minimálního počtu míst v programu, jsou deklarovány s použitím klíčového slova `inline` taktéž pro efektivnější výsledný program.

Součástí každé testovací sady musí být funkce sloužící k provedení daných testů a funkce pro zaslání jejich přehledně formátovaných výsledků uživateli. Přidávat další testovací sady je možné pouze při dodržení zmíněných podmínek. K začlenění nové testovacích sady stačí obě funkce provázat s hlavní obslužnou rutinou programu v souboru `tests.c`.

Testování časovačů RTC, LPTMR a PIT

Struktura testování časovačů je velmi podobná, proto byl jejich popis sloučen do jedné sekce. Dílčí zdrojové kódy pro vyjmenované časovače se nacházejí v souborech `RTC.c`, `LPTMR.c` a `PIT.c`.

Pro účely přednastavení řídicích registrů je u každého modulu implementována funkce začínající názvem `_timersInit`. Z hlediska časovačů se většinou jedná o jejich aktivaci povolením příslušného rozvodu hodin, povolení vzniku přerušení a případnou volbu režimu, ve kterém bude časovač pracovat. Přednastavení proběhne vždy jen jednou na začátku spuštění testovacího firmwaru.

Spuštění samotného testování časovače je provedeno zavoláním funkce s názvem začínajícím na `_timersRun`, která mimo jiné zajišťuje vynulování čítacích registrů, povolení obsluhy přerušení mikrokontrolérem a přednastavení nových řídicích hodnot. V rámci každého zdrojového souboru jsou na jeho začátku uvedeny makra překladového preprocesoru definující tyto řídicí hodnoty. Konkrétní význam hodnot je pak možné vyhledat v přiloženém komentáři.

Oproti tomu funkce s pojmenováním začínajícím na `_timersStop` způsobí zastavení časovače a zakáže obsluhu přerušení. Volána je v případě funkčního modulu časovače automaticky po uplynutí přednastavené doby (již zmíněnými konfiguračními makry).

U každého časovače je testována jak funkce vnitřního čítače, tak i generování přerušení v rámci specifické události. Hodiny reálného času (RTC) jsou například testovány na pravidelnou inkrementaci čítače v rozmezí přibližně 1 sekundy a také na spuštění alarmu při překročení určité prodlevy.

Časovač s nízkým příkonem (LPTMR) disponuje funkcí porovnávání hodnoty čítače a přednastaveného registru. Při překročení hranice generuje přerušení a nuluje čítač. Kontrolován je jak vznik přerušení ve správnou dobu, tak inkrementace čítače v závislosti na aktivovaném vstupním signálu z modulu LPO[6] o frekvenci 1KHz.

Poslední sadou testovaných časovačů jsou celkem 4 časovače s periodicky generovaným přerušením (PIT). Implementované řešení prověřuje možnost vzniku přerušení a jeho skutečnou pravidelnost. Sdružený režim (anglicky chain mode) je také postupně otestován vždy

[6] LPO – Low Power Clock (hodiny s nízkých příkonem)

v kombinaci dvou sousedních časovačů, přičemž není testováno přetečení čítačů z důvodu urychlení testovacího procesu.

Aby nedocházelo k zaseknutí testovacího firmwaru vlivem čekání na nefunkční časovač, byl čekající cyklus vybaven „manuálním watchdogem“. Jedná se o jednoduchý algoritmus, který si interně počítá provedený počet cyklů a při překročení zvolené hranice zkontroluje stav podstatných registrů časovače. V některých případech si zároveň pamatuje jejich předchozí hodnotu a sleduje, zdali byla hodnota změněna dle očekávání. V případě, že časovač nefunguje tak, jak je uvedeno v dokumentaci, hlídací algoritmus testování přeruší a patřičně o situaci informuje uživatele.

Testování GPIO pinů

Všechny testy týkající se vstupně/výstupních pinů jsou implementovány v souboru `ports.c`. Jejich testování však vyžaduje navíc připojení podpůrného hardwarového modulu (viz sekce 3.2). V případě, že není dostupný, testy jsou automaticky vynechány.

Testování se neobejde bez implementace funkcí ke komunikaci skrze modul pro rozhraní SPI, které je využito při obsluze přídavného hardwarového modulu. V souboru `SPImaster.c` se nachází funkce pro zahájení činnosti modulu SPI v mikrokontroléru a nastavení do režimu `master`. Další funkcí je `SPIMasterWriteReadData()`, jež slouží pro výměnu dat mezi mikrokontrolérem a podpůrným testovacím hardwarovým modulem.

Soubor `MCP23S17.c` obsahuje funkce pro nastavení a komunikaci s integrovaným obvodem MCP23S17 [11]. Implementace jednotlivých funkcí vychází ze zmíněné dokumentace a využívá pro svou činnost také obslužné funkce k modulu SPI. Důležitou funkcí je `MCP23S17_Init()` sloužící k otestování přítomnosti a funkčnosti hardwarového modulu a následnému nastavení výchozích hodnot – všechny GPIO piny integrovaného obvodu jsou nastaveny jako vstupní s povolenými pull-up rezistory a odráží skutečnou logickou hodnotu. Pomocné funkce `MCP23S17_ReadRegister()` a `MCP23S17_WriteRegister()` slouží zpravidla ve vnitřní implementaci funkce `MCP23S17_ReadRegisterAB()` pro čtení dat z obou sad (A i B) registru integrovaného obvodu a funkce `MCP23S17_WriteRegisterAB()` určené k jejich zápisu. Pro větší srozumitelnost je doporučeno nahlédnout do hlavičkového souboru `MCP23S17.h`. Speciálním rozšířením jsou pak funkce `MCP23S17_ReadGPIOAB()`, `MCP23S17_WriteGPIOAB()` a `MCP23S17_SetGPIOABDirection()` umožňující snadnější čtení/zápis logických hodnot GPIO pinů či nastavení jejich směru toku dat. Vnitřní funkce `_MCP23S17_SPIcontrolByte()` pak vytváří řídicí byte pro integrovaný obvod při komunikaci skrze rozhraní SPI.

Před zahájením testování je provedeno funkcí `_portsSetup()` potřebné přednastavení pinů na vstupně/výstupních portech mikrokontroléru. Jedná se především o nastavení směru toku dat, povolení hodinového signálu do periferních modulů pro jejich aktivaci, nastavení výchozích hodnot na pinech a povolení vnitřních pull-down rezistorů. Z hlediska implementace jsou použity 2 možné varianty – nastavení každého pinu zvlášť skrze vlastní registr a použití globálního řídicího registru pro hromadné nastavení více pinů současně.

Spustit testování GPIO pinů mikrokontroléru je možné funkcí `portsGPIOTest()`. Nejprve je provedena detekce funkčnosti a aktivace podpůrného hardwarového modulu. Následně jsou navrženou technikou (viz 3.2) otestovány všechny GPIO piny portu A i pin číslo 28 na portu E. V závěru je využit fotorezistor osazený na hardwarovém modulu k samočinnému testování svítivosti indikačních LED platformy FITkit3. Detekce logické úrovně 1 na pinu, kde je připojen fotorezistor, označuje funkční indikační LED, přičemž pro ustálení

analogové hodnoty napětí na fotorezistoru je do procesu mezi aktivaci a detekci jednotlivých indikačních LED vložena krátká prodleva.

Testování tlačítek a indikačních LED

Spuštění testů tlačítek provádí funkce `portsButtonsTest()` implementovaná taktéž v souboru `ports.c`. Samotné testování představuje nekonečný cyklus, díky kterému se opakovaně detekuje hodnota na každém vstupním pinu, kde jsou připojeny tlačítka. Stisknuté tlačítko značí logická úroveň 0. Při její detekci je zavolána funkce `_btnPressAction()`, jež zašle zprávu o stisku konkrétního tlačítka uživateli. Navíc zde dochází k cyklické aktivaci a deaktivaci indikačních LED na kitu, aby je bylo možné testovat i manuálně v případě nepřipojeného rozšiřujícího hardwarového modulu. Zaznamenání stisku probíhá logickým přičtením 1 do proměnné `_testedBtns` na bitovou pozici určenou pořadím tlačítka. K prevenci před mnohonásobnou detekcí (vlivem nedokonalosti spoje v tlačítku a generování rušení při stisku) je použita pomocná proměnná `activated`, jež obdobně jako `_testedBtns` udržuje logickou 1 na konkrétní bitové pozici od stisku tlačítka až po jeho uvolnění.

Nekonečný cyklus testování končí, pokud vnitřní proměnná `_testedBtns` obsahuje bity s logickou hodnotou 1 na všech bitových pozicích dle pořadí tlačítek. V případě, že je některé tlačítko nefunkční a není tak možné automaticky ukončit cyklus, je doporučeno použít funkční klávesu pro přerušení testování.

Využití jazyku symbolických adres

Využití assembleru umožňuje vytvořit kus kódu s přesně definovanými instrukcemi. Takového jevu lze využít například při vytváření funkce s parametrickou prodlevou čekání, která nevykonává žádnou podstatnou činnost. Obyčejně by stačil do kódu umístit cyklus s prázdným tělem čítající do potřebného počtu iterací. Problém však nastane, pokud je zapnuta optimalizace při překladu. Jednou z možností, jak předejít odebrání prázdných smyček překladačem, je obalit chtěný kus kódu speciálními `pragma` makry preprocesoru[7]. Dokonalejším principem může být umístění nějaké instrukce v assembleru[8] do těla cyklu, kterou překladač nesmí odebrat. V kombinaci s vložením pouhé pseudoinstrukce pak cyklus i počet instrukcí zůstane zachován. Následující funkce `delayUS` slouží k provedení prodlevy v řádu mikrosekund a nachází se v hlavičkovém souboru `main.h`. Cyklus byl během implementace speciálně odladěn programem `objdump`[9], aby se na úrovni assembleru skládal vždy ze 4 instrukcí. Pokud je předdefinován správný takt procesoru v MHz makrem `MCU_CLOCKS`, je možné vypočítat přibližný nutný počet iterací zpožďujícího cyklu vynásobením požadované prodlevy v mikrosekundách s $\frac{1}{4}$ hodnoty taktu v MHz.

```
#define MCU_CLOCKS 48 /**< MHz */

inline void delayUS(register uint32_t us) {
    for(us *= MCU_CLOCKS/4; us > 1; us--)
        __asm__ __volatile__ ("" : : "r" (us));
}

#define delayMS(ms) delayUS((ms)*1000)
```

[7] https://gcc.gnu.org/onlinedocs/gcc/Function-Specific-Option-Pragmas.html
[8] https://gcc.gnu.org/onlinedocs/gcc/Extended-Asm.html
[9] https://en.wikipedia.org/wiki/Objdump

Výsledné řešení není na mikrosekundy přesné, avšak pro využití zpoždění u testování časovačů a reproduktoru je dostačující. Vhodné může být také makro `delayMS` čekající s prodlevou v řádu milisekund.

Tvorba překladového skriptu Makefile

Aby bylo možné zachovat samostatnou činnost překladu zdrojových kódů a přitom se osvobodit od vývojového prostředí KDS, byl vytvořen speciální skript pro program `make`. Současně je však stále potřeba překladové prostředí včetně programů pro nahrání testovacího firmwaru do FITkitu3. Více je o této problematice uvedeno v příloze B.

Při vytváření skriptu byly čerpány informace z manuálu programů `make` [4], `GNU/GDB` [6], `GNU/GCC` [5] a článku o využití kombinace programů `GNU/GDB` a `PEMicro GDB server` z příkazové řádky [15]. Inspirací dále byly původní překladové skripty vygenerované vývojovým prostředím KDS.

Na začátku skriptu se nachází sekce s přednastavením překladových a jiných parametrů. Následují automatické návěští, které přeloží všechny nalezené `*.c` soubory ve složce se zdrojovými kódy a slinkují jejich přeložený obsah do výsledného binárního souboru. Skript také dále podporuje následující pomocné návěští:

all	Spustí překlad zdrojových kódu.
flash	Nahraje testovací firmware do připojeného výukového kitu, přičemž pomocí proměnné `DEVICEPORT` lze zvolit port, ke kterému je připojeno zařízení.
run	Spustí testovací firmware v připojeném FITkitu3. Platí zde stejné pravidlo pro použití proměnné `DEVICEPORT` jako u návěští `flash`.
doc	Spustí generování programové dokumentace.
clean-doc	Smaže vygenerovanou programovou dokumentaci.
clean-all	Smaže stejné soubory jako návěští `clean` a navíc i přeložený binární soubor `FITkit3Tests.elf`.
clean	Smaže všechny pomocné soubory vytvořené při překladu.

4.2 Podpůrný testovací hardware

Tvorba podpůrného hardwarového modulu pro testování vstupně/výstupních periferií mikrokontroléru na platformě FITkit3 by se dala rozdělit na 3 části, které jsou rozděleny do podsekcí níže. **Hardwarový modul je doporučeno připojovat vždy k vypnutému FITkitu3 až po nahrání testovacího firmwaru**, aby nedošlo k poškození modulu vlivem neznámého programu. Výsledný vytvořený prototyp hardwarového modulu ukazuje obrázek 5.1.

Tvorba elektrického schématu zapojení

Elektronické schéma bylo vytvořeno pomocí webové aplikace EasyEDA[10]. Podle návrhu řešení musejí být GPIO piny zvoleného integrovaného obvodu `MCP23S17` propojeny s GPIO

[10] https://easyeda.com/

piny mikrokontroléru, aby mohlo dojít k jejich otestování. Obdobně je potřeba propojit také komunikační rozhraní SPI. Schéma výukového kitu dostupné z oficiálních stránek [3] ukazuje, že všechny potřebné kontakty jsou na desce vyvedeny do rozšiřujícího konektoru s označením P1. Pro intuitivní zapojení byly GPIO piny mikrokontroléru seřazeny vzestupně – tedy od pinu s číslem 6 po pin číslo 29 portu A a nakonec pin číslo 28 portu E. K napájení celého obvodu bylo zvoleno 3.3V, aby korespondovalo s napájením GPIO pinů mikrokontroléru.

Samočinné testování sady indikačních LED vyžaduje dalé použití fotorezistoru. Nejvhodnější by bylo k jeho připojení použít analogově/digitální převodník. Avšak protože má použitý integrovaný obvod pouze digitální vstupy, bylo zvoleno řešení, kdy je fotorezistor umístěn v zapojení odporového děliče společně s rezistorem o odporu 220KΩ. Při posvícení na fotorezistor klesne jeho odpor přibližně o jeden řád (záleží na provedení) a vlivem toho stoupne napětí na rezistoru. Digitální GPIO pin integrovaného obvodu je připojen mezi fotorezistor a rezistor, čímž vůči zemi naměří napětí na rezistoru. V zapojení se využívá jevu detekce digitální logické úrovně v analogovém signálu, přičemž u obvodu `MCP23S17` je podle dokumentace [11] analogové napětí do hodnoty 0.2 násobku napájecího napětí (v tomto případě 0.66V) považováno za logickou úroveň 0, naopak analogové napětí od 0.8 násobku napájecího napětí (v tomto případě 2.64V) je považováno za logickou úroveň 1. Ostatní rozmezí se považuje za zakázanou zónu. Použitý fotorezistor má při posvícení přibližný odpor 30KΩ.

Podle zapojení vývodů integrovaného obvodu dostupného v jeho dokumentaci [11] bylo navrženo elektrické schéma na obrázku 4.2.

Návrh plošného spoje

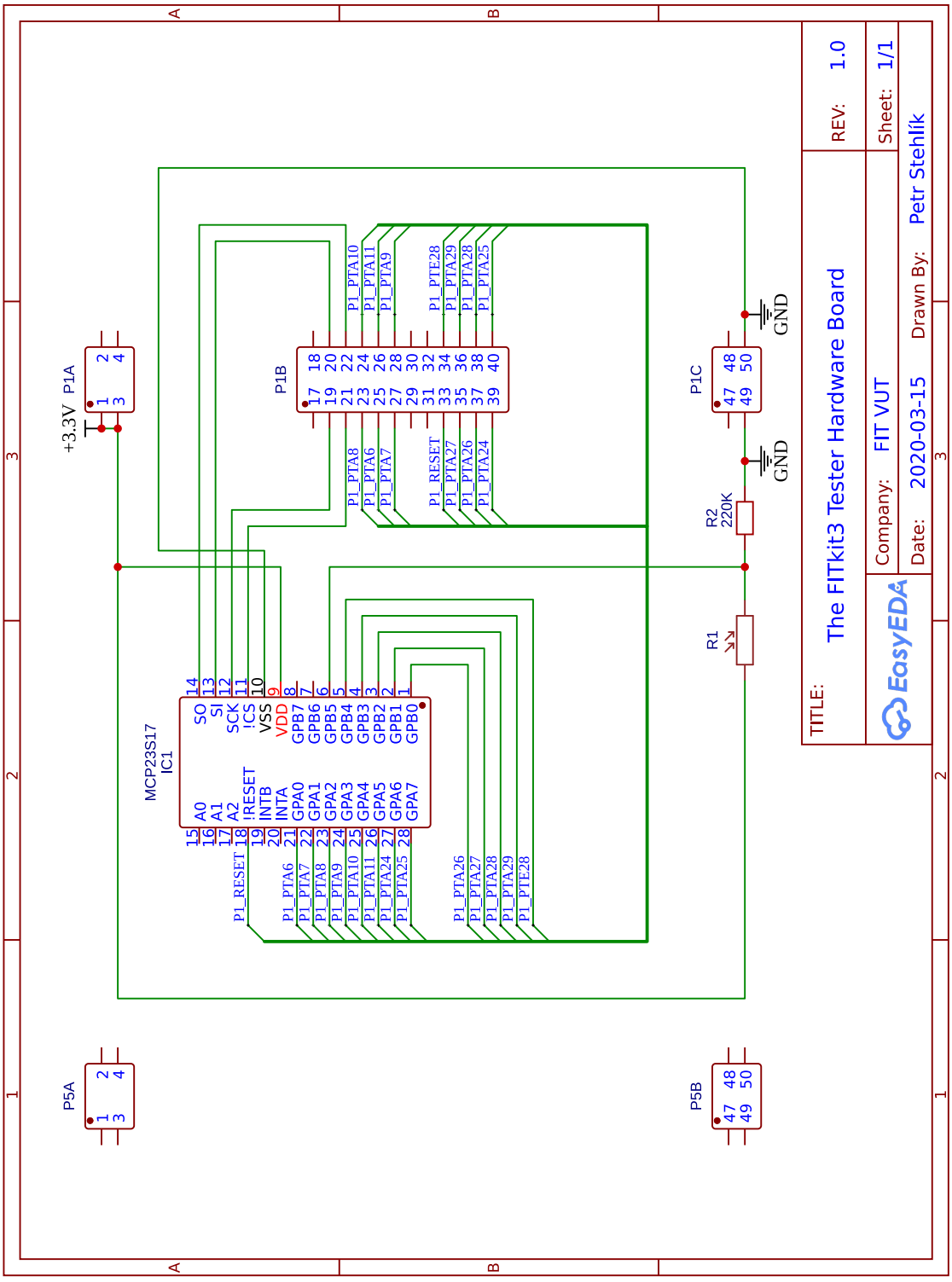
Tvorba plošného spoje byla provedena (stejně jako elektrické schéma) ve webové aplikaci EasyEDA. Návrh vedení cest probíhal ručně, poněvadž byl kladen důraz na jednostranný plošný spoj. Oboustranné prokovení děr je však nutností, jelikož patice s integrovaným obvodem a rezistor se osazují seshora. Všechny ostatní součástky jsou pak určeny k osazení zespoda.

Výsledný návržený plošný spoj s řádně okótovanými rozměry je možné vidět na obrázku 4.3.

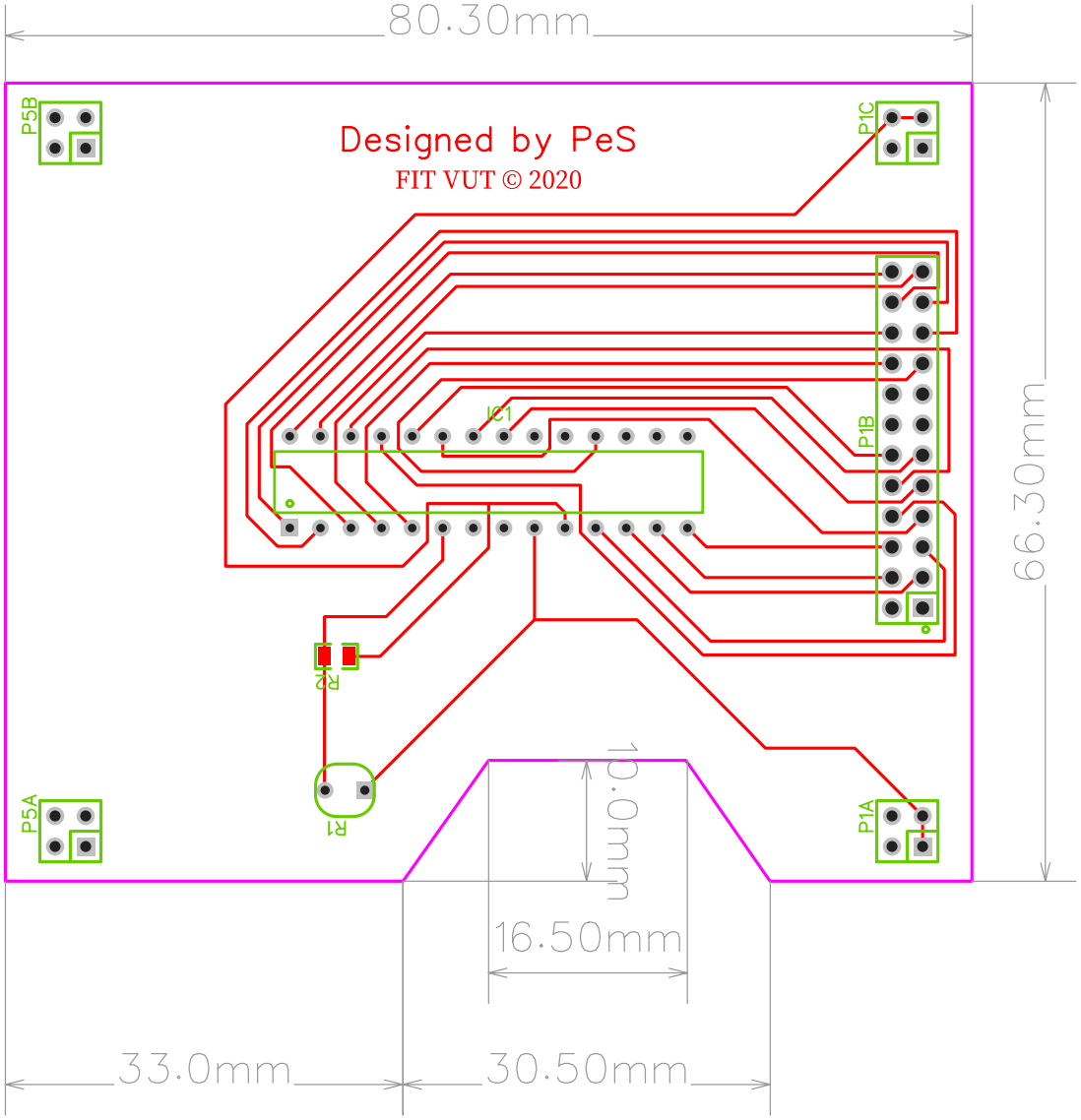
Modelování světlovodu

V procesu modelování bylo použito modelovací prostředí OpenSCAD[11]. Výhodou tohoto programu je možnost parametrizovat navržené modely. Díky tomu lze vlastnosti či rozměry vytvořeného světlovodu jednoduše měnit. Výsledná tvorba probíhala metodou pokus-omyl, protože některé rozměry nemohly být přesně změřeny, ale pouze odhadnuty. Celkem byly provedeny 3 iterace návrhu. Na obrázku 4.4 je možné vidět vykreslený finální model světlovodu v již zmíněném modelovacím prostředí.

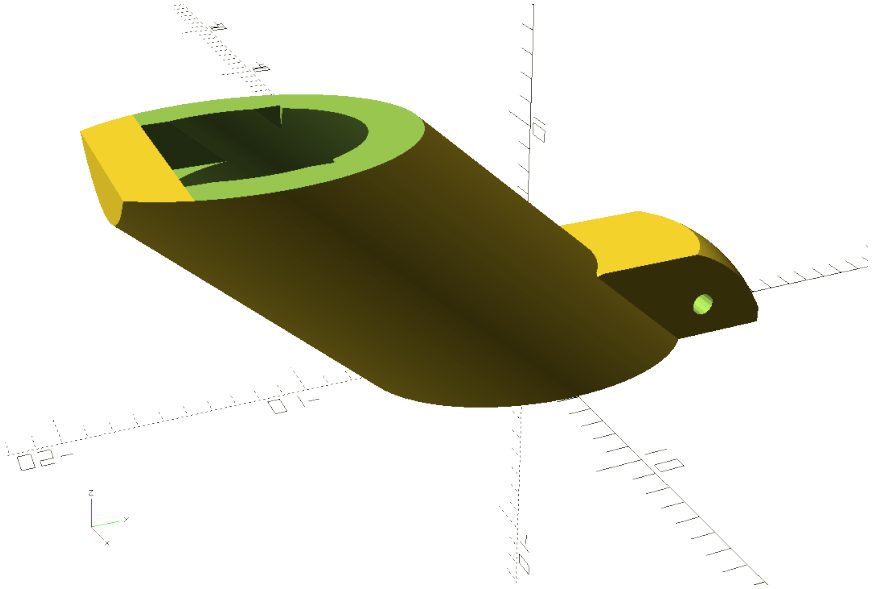
[11] http://www.openscad.org/



Obrázek 4.2: **Elektrické schéma zapojení podpůrného hardwarového modulu.** Napájení celého obvodu je 3.3V. Mezi obsažené součástky patří integrovaný obvod `MCP23S17`, fotorezistor (`R1`) a obyčejný rezistor s odporem 220KΩ, přičemž rezistory v daném zapojení tvoří odporový dělič. Konektory `P5` nejsou nijak využity, slouží jen pro uchycení modulu na výukovém kitu. Komunikace s modulem probíhá skrze rozhraní SPI a připojeny jsou pro testování všechny GPIO piny obsažené na testovaném MCU.



Obrázek 4.3: **Deska plošného spoje podpůrného hardwarového modulu.** Jedná se o pohled shora. Během tvorby byl kladen důraz na jednostranné propojení všech prvků, aby výsledná deska byla snadnější a levnější na výrobu. Osazení součástek (kromě fotorezistoru a konektorů) je nutné provést seshora. Naopak konektory a fotorezistor lze osadit zespoda. V dolní části obrázku je možné vidět speciální výřez na tlačítka, která se nachází v tomto prostoru na FITkitu3. Skutečné rozměry desky včetně výřezu jsou řádně okótovány a šířka cestiček je výchozích 0.254mm.



Obrázek 4.4: **Vykreslený 3D parametrický model navrženého světlovodu.** Zahrnuje packu s otvorem pro uchycení k desce třeba pomocí drátku, seříznutou hranu na horním konci pro zarovnání s deskou výukového kitu, podpěrnou plošinu pro čidlo i s otvory pro jeho vývody (nejde z daného úhlu na obrázku vidět) a speciální výřez na konci trubice přesně pasující na sadu indikačních LED výukového kitu určenou k testování. Mezi vypočítané parametry patří především tloušťka stěny (1.5mm), výška objektu (11.2mm) a vnitřní průměr roury pro čidlo včetně rezervy (6.9mm). Model je určen k plastovému 3D tisku.

4.3 Software pro obsluhu testování z PC

Implementace softwaru pro obsluhu testování nebyla tak jednoduchá, jak by se na první pohled mohlo zdát. Vývoj byl rozdělen hned zpočátku do 4 základních modulů, z nichž posléze vznikal výsledný skript `FITkit3Tester.py`. Následující podsekce popisují tvorbu každého modulu zvlášť, přičemž v poslední je popsáno jejich sjednocení. Dílčí skripty nebyly zařazeny mezi finální přílohy, protože podstatný je až výsledný skript.

Zpracování programových argumentů – args.py

Jedním ze způsobů, jak programu před spuštěním předat potřebné parametry, je pomocí programových argumentů (zadaným například z příkazové řádky). V jazyce Python3 se nabízí 2 rozumné řešení – novější možností je použít standardní modul `argparse`[12], který je podle dokumentace v jazyce dostupný od verze 3.2. Starší způsob představuje použití standardního modulu `getopt`[13], jenž se svým stylem zpracování podobá řešení v jazyce C. K implementaci ve vlastní funkci `programArguments()` byl nakonec zvolen modul `getopt`.

Jestliže některý z programových parametrů přijímá i specifikovanou hodnotu, je její formát nejprve zkontrolován a v případě nevhodného zápisu je skript ukončen s patřičným hlášením. Dobrým příkladem může být hodnota parametru `--flashscript`, která určuje cestu k externě spustitelnému programu včetně jeho programových argumentů. Tato hodnota

[12] https://docs.python.org/3/library/argparse.html
[13] https://docs.python.org/3/library/getopt.html

je nejprve rozdělena podle znaků prázdné mezery na samostatné části. Skript poté předpokládá a ověří, že v 1. části se nachází pouze platná cesta k externě spustitelnému souboru a případné ostatní části jsou jeho argumenty.

Běžnými podporovanými parametry jsou `--help` a `--version`, přičemž více o možnostech použití programových argumentů pojednává sekce C.2 v přiložené uživatelské příručce.

Komunikace skrze sériový port – serial.py

Během implementace komunikace bylo čerpáno z dokumentace knihovny `pySerial` [10], která je použita jednak k vyhledávání dostupných zařízení a dále také k navázání samotné komunikace a obsluze vstupu/výstupu.

Pro vyhledávání zařízení knihovna poskytuje 2 funkce – `comports()` a `grep()`. Na následujícím výňatku z vlastní funkce `serialListDevices()` je možné předvést jejich rozdíly:

```
devices = list(grep('(15A2)') if canFilter else comports())
for d in devices:
  if (canFilter and ( (d.manufacturer is not None and
                       'Freescale' not in d.manufacturer) or
                       (d.vid is not None and d.vid != 5538) )):
    devices.remove(d)
```

Proměnná `canFilter` slouží k aktivaci filtru. Z toho plyne, že funkce `comports()` vrací všechny právě dostupné sériové porty. Naopak funkce `grep()` umožňuje seznam portů před návratem vyfiltrovat například podle čísla prodejce. Typickými identifikačními údaji pro FITkit3 připojeným skrze rozhraní USB je číslo 5538 (v hexadecimální soustavě se jedná o hodnotu 15A2) jakožto číslo prodejce a slovo „Freescale“ obsažené v případném textovém řetězci s názvem výrobce. Následný cyklus a podmínka v kódu obslouží pro jistotu filtrování podle názvu výrobce a výsledkem je seznam zařízení odpovídajících platformě FITkit3.

Začátek komunikace je umístěn do vlastní funkce `serialCommunicationInit()`. Nejprve je nutné před navázáním spojení nastavit správně komunikační parametry, které jsou přehledně shrnuty v sekci B.2 přiloženého uživatelského manuálu. Poté je na řadě pokus o navázání spojení pomocí funkce `open()`. Jestliže se vydaří, jsou vyprázdněny dočasné vstupně/-výstupní paměti funkcemi `reset_input_buffer()` a `reset_output_buffer()`. Dále je zaslán znak `R` funkcí `write()` znamenající pokyn k restartu testovacího firmwaru uvnitř FITkitu3. Následuje totéž se znakem `I`, který slouží pro opětovné zaslání identifikačních údajů (obsahujících textový řetězec „FITkit3Tests“) z výukového kitu do obslužného softwaru. Tyto údaje mají za úkol potvrdit přítomnost testovacího firmwaru ve FITkitu3 a zároveň funkčnost modulu UART.

Periodicky volaná vlastní funkce `serialCommunicationIO()` obsluhuje vstup a výstup. Čtení vstupu je zde vylepšeno o „ruční“ zpracování speciálních netisknutelných řídicích znaků. Jedná se o znak `\r` s hodnotou 13 ve standardní ASCII[14] tabulce sloužící pro návrat kurzoru na začátek řádku (anglicky Carriage return) a znak `\b` s hodnotou 8 (podle ASCII tabulky) používající se pro smazání předcházejícího znaku (anglicky Backspace). Vylepšení může připomínat simulaci pohybu kurzoru na obrazovce. Samotné zpracování však obsahuje jednu výjimku, jelikož řídicí znak `\b` má umožněno smazat také řídicí znaky

[14] ASCII – Kódová tabulka pro znaky anglické abecedy a jiné speciální znaky využívané v informatice (viz https://cs.wikipedia.org/wiki/ASCII).

konce řádku a tím pomyslně „přetéct“ na řádek předchozí. Využití této vlastnosti je popsáno v sekci B.2 přiložené uživatelské příručky. Aby výjimka fungovala správně, je nutné před čtením vstupu získat předcházející nekompletní obsah až po poslední zakončený řádek včetně. K tomuto obsahu je poté připojen nový obsah vstupu. Následující výňatek ze zdrojového kódu představuje zmíněný algoritmus, přičemž aktuálním obsah řádků je uložen jako pole v proměnné `form.console.values`:

```
for i in range(0,2):
  if len(form.console.values) > 0:
    lineLength = len(form.console.values[-1])
    out = form.console.values[-1] + out
    del form.console.values[-1]
    if lineLength > 0 and out[lineLength-1] == '\n':
      break
  else:
    break
while connection.in_waiting > 0:
  out += connection.read(connection.in_waiting).decode()
```

V novém obsahu jsou od sebe odděleny kompletně a nekompletně přijaté řádky. Kompletní řádek musí být zakončen znakem nového řádku `\n` či znakem `DEL` (s hodnotou 127 ve standardní ASCII tabulce), přičemž znak `DEL` je umístěn testovacím firmwarem speciálně na konec informativní zprávy, aby ji bylo možné identifikovat. Poté je na kompletní řádky spuštěn algoritmus, který zpracuje netisknutelné znaky a finálně jsou předchozí nekompletní řádky zase připojeny na konec nového obsahu. Pro lepší pochopení je vhodné nahlédnout do zdrojového kódu.

Ukončení komunikace (například v důsledku zavírání obslužného programu) lze provést voláním vlastní funkce `serialCommunicationClose()`, která před ukončením ještě zašle znak `R` jako pokyn k restartu FITkitu3.

Textové uživatelské rozhraní – tui.py

Knihovna `npyscreen`, jež je použitá pro implementaci textového uživatelského rozhraní, vypadala na první pohled velmi podařeně. Avšak opak může být pravdou. Dokumentace [1] není příliš podrobná, a proto bylo nutné během implementace čerpat i ze zdrojových kódů[15]. Některé části musely být vylepšeny a jiné zcela přepsány, aby bylo docíleno potřebného efektu. Například vytvořené vlastní třídy `PopupCentered` i `YesNoPopupCentered` slouží jako rozšíření knihovní vestavěné třídy `npyscreen.ActionPopup` s tím rozdílem, že mají opravené automatické centrování na střed okna při změně jeho velikosti.

Podobně špatně fungující byly i vestavěné knihovní funkce pro vyskakovací okna s upozorněním. Byly proto přepsány a ve zdrojovém kódu je lze nalézt pod vlastními názvy `notifyOKDialog` a `notifyYesNoDialog`. Využívají také již zmíněné vytvořené vlastní třídy. Upravené nebo přidané části jsou označeny komentářem a zbytek kódu je převzat z původních funkcí knihovny `npyscreen`.

Přidaná musela být i vlastní třída `selectorWidget` vycházející z vlastností knihovní třídy `npyscreen.MultiLine`. Upravuje své chování tak, aby vyhovovalo principu výběru zařízení ze seznamu.

[15] https://bitbucket.org/npcole/npyscreen/src/default/npyscreen/

Poslední úprava se týká přidané třídy `pagerConsole`, která dědí své vlastnosti z knihovní třídy `npyscreen.Pager`. Využita je jako konzole pro zobrazení jak vstupu ze sériového portu, tak přesměrovaného výstupu z externě spuštěného skriptu. Rozdíl oproti původní implementaci je ve vylepšeném zobrazení speciálního nečitelného znaku `\t` (horizontální tabulátor má ve standardní ASCII tabulce hodnotu 9), který není zobrazen jen jako 1 mezera, ale chová se jako klasický tabulátor s přednastavenou šířkou 8 znaků.

Samotné textové uživatelské rozhraní je vytvořeno třídami `consoleForm`, `flashForm` a `selectorForm`, z nichž každá určuje samostatné okno s rozložením prvků a jejich funkcionalitou. V okně třídy `consoleForm` se zobrazuje obslužný terminál pro účely procesu testování s 2 tlačítky – slouží k přerušení komunikace s výukovým kitem a k ukončení testování, nebo pro zavření celého programu. Zde se projevuje skutečná výhoda použitého textového rozhraní, protože výstup z FITkitu3 tak není potřeba příliš upravovat a může být rovnou zobrazen uživateli.

Stejné rozvržení prvků obsahuje také okno implementované ve třídě `flashForm` s tím rozdílem, že terminál zobrazuje pouze výstup z externě spuštěného procesu, ale nepřijímá žádný vstup. Dokud běží externí skript, není možné okno uzavřít, aby nedošlo k porušení nahrávání firmwaru. V případě výskytu chyby se zobrazí vyskakovací okno s příslušnou zprávou a proces nahrávání je ukončen. Po úspěšném dokončení je možné pokračovat tlačítkem „Continue“ ke spuštění procesu testování, anebo se vrátit zpět na výběr zařízení tlačítkem „Close“.

Poslední okno vytvořené v rámci třídy `selectorForm` zobrazuje formulář pro výběr zařízení. Součástí jsou také 2 tlačítka – pro ukončení programu a pro přepnutí aktivačního stavu filtru zařízení. V seznamu se může zobrazit zpráva informující o nenalezení žádných zařízení. V případě, že je kurzorem pomocí kurzorových šipek vybráno nějaké zařízení a výběr potvrzený stiskem klávesy `Enter`, dojde k vyvolání metody `runTests()`, jež je implementována uvnitř třídy. Jejím úkolem je spustit detekci přítomnosti testovacího firmwaru v zařízení a následně rozhodnout, zdali bude spuštěn skript pro samočinné nahrávání firmwaru, anebo bude zobrazeno okno pro obsluhu testování. Vznik jakékoliv chyby je okamžitě zobrazen ve vyskakovacím okně s příslušnou zprávou pro uživatele.

Vícevláknovost a spouštění externích procesů – process.py

Pro podporu spuštění externího procesu ve skriptovacím jazyce Python3 existuje modul `subprocess`. Implementace je rozdělena do 3 základních funkcí, přičemž byla použita dokumentace modulu [13] jako zdroj informací ohledně použití.

Spuštění externího procesu zajišťuje funkce `runFlashScript()`. Jako parametry přijímá příkaz ke spuštění (platnou cestu k externímu skriptu s případnými argumenty oddělenými mezerou) a port, ke kterému je připojeno zvolené zařízení. V případě, že je součástí příkazu slovo „FLASHDEVICE“, funkce všechny shody v textovém řetězci nahradí názvem portu zařízení před spuštěním samotného skriptu. Využití tak může být například k předání informace o zvoleném zařízení.

Protože se během implementace ukázalo, že čtení výstupu spuštěného procesu vždy čeká na vyrovnávací paměti a zasekne průběh celého programu, musela být obsluha čtení výstupu vyčleněna do samostatného vlákna. Součástí jazyku Python3 je modul `threading`[16] umožňující běh vlastní funkce v jiném vlákně. Z toho důvodu je implementována funkce

[16] https://docs.python.org/3.8/library/threading.html

`handleFlashScriptOutput()`, jež vždy čeká na výstup spuštěného externího skriptu a následně jej uloží do sdílené proměnné `buffer`.

Vlastní funkce `handleFlashScript()` volaná z prostředí textového rozhraní pak aktualizuje obsah terminálu příslušnými daty ze sdílené proměnné `buffer` a zároveň hlídá ukončení včetně návratové hodnoty běžícího procesu.

Vytvořené řešení se v obslužném testovacím softwaru používá pro spuštění externího skriptu k bezpečnému a přenositelnému nahrání testovacího firmwaru do výukového kitu. Přiložený skript `flash.sh`[17] pro interpretr Bourne shell[18] využívá implementovaný `Makefile` (viz sekce 4.1), který je součástí testovacího firmwaru. O vytvoření vlastního nahrávacího skriptu více pojednává příloha C.4.

Sjednocení a vznik výsledného skriptu – FITkit3Tester.py

Finální skript je vytvořen sloučením 4 zmíněných předchozích modulů. Částečně je použito objektově orientované programování, protože jej vyžaduje knihovna pro textové uživatelské rozhraní. Vše ostatní je rozčleněno pouze do funkcí dle potřeby.

Jádro běžící aplikace řídí algoritmus v knihovní třídě `npyscreen.NPSAppManaged`, který se stará o vykreslování formulářů a obsluhu interakce uživatele s uživatelským rozhraním. V případě potřeby poskytuje běhový čas ostatním funkcím zajišťujícím komunikaci skrze sériový port nebo obsluhu spuštěného externího procesu.

[17] Skript `flash.sh` je součástí adresáře `/Software/`.
[18] https://en.wikipedia.org/wiki/Bourne_shell

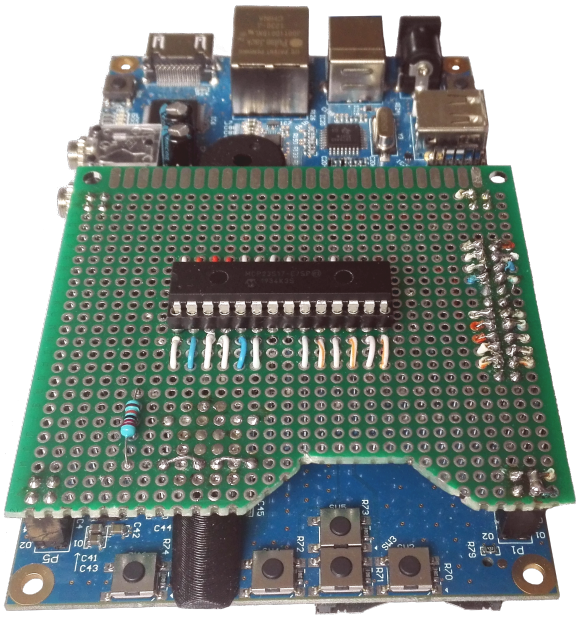
Kapitola 5

Zhodnocení navrženého řešení

Podle zadání práce byla úspěšně vytvořena funkční testovací sada pro vybranou množinu prvků platformy FITkit3. Výsledné řešení se snaží být svým konceptem snadno modifikovatelné, ve zdrojových kódech přehledné i řádně okomentované a také rozšiřitelné pro nové soubory testů dalších periferií a modulů. Součástí kapitoly je ukázka vyrobeného hardwarového modulu, dále popis výstupu z kompletního průběhu všech testů a následná diskuze.

5.1 Vyrobený prototyp podpůrného hardwarového modulu

Na obrázku 5.1 je možné vidět vyrobený prototyp hardwarového modulu s popisem.



Obrázek 5.1: **Prototyp hardwarového modulu nasazený na FITkitu3.** Připojení je doporučeno provádět k vypnutému výukovému kitu až po nahrání testovacího firmwaru (nikoliv dříve, aby nedošlo ke zkratu vlivem spuštění předchozího neznámého programu). Pro spojení i komunikaci jsou využity dvouřadé konektory po stranách desky. Ve spodní části obrázku mezi tlačítky je možné vidět speciálně navržený světlovod vytištěný z černého plastu na 3D tiskárně Original Prusa i3 MK3S. Slouží ke směrování světla z indikačních LED do fotorezistoru, jenž se využívá při automatické detekci záření během jejich testování.

5.2 Ukázka průběhu celého testu s popisem výstupu

Podrobný popis obsluhy testovacího firmwaru lze nalézt v příloze B a testovacího softwaru v příloze C. Následuje ukázkový výstup průběhu kompletních testů s popisem:

```
Welcome to Routines for testing modules of MCU (MK60DN512ZVMD10)
================================================================
          The FITkit3 Tester by Petr Stehlik (c) 2020

Option menu:
[0] - run All tests
[1] - run only Speaker test
[2] - run only RTC timer test
[3] - run only LPTMR timer test
[4] - run only PIT timers test
[5] - run all Timers tests
[6] - run only Buttons test
[7] - run only SPI & GPIO test
[8] - run all I/O tests
```

Na začátku je vypsána uvítací zpráva s nabídkou možností testování. Konkrétní soubor testů se volí odpovídající číslicí. V tomto případě bylo zvoleno provedení všech testovacích sad pod číslem 0. Pokračuje výpis na základě jejich spuštění:

```
Testing I/O...
Testing GPIO ports...
HWTestBoard module initialization...
Testing SPI in master mode...
Testing SPI in master mode... Done
```

Nejprve jsou na řadě testy vstupně/výstupních portů. K automatickým testům je ovšem potřeba patřičně nastavit přídavný hardwarový modul. Proběhne tedy pokus o zahájení komunikace s hardwarovým modulem přes rozhraní SPI, přičemž se zároveň otestuje i modul zodpovědný za SPI.

```
Initializing MCP23S17 module...
Initializing MCP23S17 module... Done
HWTestBoard module initialization... Done
```

Rozhraní SPI funguje, tudíž následuje přednastavení a aktivace hardwarového modulu. Nyní již přišlo na řadu samotné testování vstupně/výstupních portů a přidružených pinů:

```
Testing GPIO pins as input...
        PTA_6... Done
        PTA_7... Done
        PTA_8... Done
        PTA_9... Done
        PTA_10... Done
        PTA_11... Done
```

```
        PTA_24... Done
        PTA_25... Done
        PTA_26... Done
        PTA_27... Done
        PTA_28... Done
        PTA_29... Done
        PTE_28... Done
        All input pins... Done
Testing GPIO pins as input... Done
Testing GPIO pins as output...
        PTA_6... Done
        PTA_7... Done
        PTA_8... Done
        PTA_9... Done
        PTA_10... Done
        PTA_11... Done
        PTA_24... Done
        PTA_25... Done
        PTA_26... Done
        PTA_27... Done
        PTA_28... Done
        PTA_29... Done
        PTE_28... Done
        All output pins... Done
Testing LEDs on board...
        All LEDs off... Done
        LED_D9... Done
        LED_D10... Done
        LED_D11... Done
        LED_D12... Done
Testing LEDs on board... Done
Testing GPIO pins as output... Done
Testing GPIO ports... Done
```

Podle výpisu lze soudit, že vstupně/výstupní piny fungují správně. Automaticky byly úspěšně otestovány také dostupné indikační LED. Pokračuje test osazených tlačítek na výukovém kitu. Jedná se o jedinou manuální část testování. Zpráva na druhém řádku vyzývá uživatele, aby postupně stiskl všechny odpovídající tlačítka, přičemž na pořadí nezáleží:

```
Testing buttons...
Try to press the switch buttons on the board one after the other.
Please cancel this test if any button does not work.
[1/5]  Button SW2 was pressed.
[2/5]  Button SW3 was pressed.
[3/5]  Button SW4 was pressed.
[4/5]  Button SW5 was pressed.
[5/5]  Button SW6 was pressed.
Testing buttons... Done
Testing I/O... Done
```

Předchozí zprávy naznačují, že testy tlačítek proběhly v pořádku. Další na řadě jsou testy všech časovačů:

```
Testing timers...
Testing Real Time Clock (RTC)...
If there is some error with RTC module wait a maximum of 7 seconds
to detect it and then testing will continue automatically!
Testing Real Time Clock (RTC)... Done
```

Na začátek byl otestován modul s hodinami reálného času. Minimální prodleva o délce 7 sekund se zde nachází pro ustálení kmitů nově aktivovaného oscilátoru. Jestliže se během této doby nezačne zvyšovat počítadlo sekund, modul je prohlášen za nefunkční. V tomto ilustračním příkladě však modul funguje správně. Následuje test časovače s nízkou spotřebou:

```
Testing Low-power Timer (LPTMR)...
Testing LPTMR in timer mode with 1kHz LPO signal input.
Testing Low-power Timer (LPTMR)... Done
```

Test proběhl taktéž úspěšně. I zde existuje automatická pojistka chránící testování před zaseknutím. Posledním modulem na řadě je sada časovačů s periodicky generujícím se přerušením:

```
Testing Periodic Interrupt Timers (PITs)...
Testing Periodic Interrupt Timers (PITs).. Special chain mode test
launched
Testing Periodic Interrupt Timers (PITs)... Done
Testing timers... Done
```

Časovače byly úspěšně otestovány ve dvou režimech – v klasickém (každý sám za sebe) a v párovém. Pořadí časovačů je pevně dané, přičemž druhý zmíněný režim se týká vždy dvou sousedních. Časovač s vyšším pořadovým číslem sníží svou hodnotu až na základě vypršení časovače s nižším pořadovým číslem. Na závěr je připraveno malé překvapení při testování reproduktoru:

```
Testing speaker... Listen!
Testing speaker... Done
```

Na konci každého testování zbývá už jen vypsat výsledky všech provedených testů přehledně do pomyslné tabulky:

```
              *** Tests results ***
==================================================
Speaker tests                                  Done
Low-power Timer (LPTMR) test                   Done
        Compare counter IRQ             OK
        Compare counter                 OK
Periodic Interrupt Timers (PITs) test          Done
        PIT0 counter IRQ                OK
        PIT1 counter IRQ                OK
```

```
        PIT2 counter IRQ                OK
        PIT3 counter IRQ                OK
        PIT0 counter                    OK
        PIT1 counter                    OK
        PIT2 counter                    OK
        PIT3 counter                    OK
        PIT1 counter chain mode         OK
        PIT2 counter chain mode         OK
        PIT3 counter chain mode         OK
Real Time Clock (RTC) test                      Done
        Time seconds counter IRQ        OK
        Time seconds counter            OK
        Time seconds alarm IRQ          OK
Buttons test                                    Done
        Button SW2                      OK
        Button SW3                      OK
        Button SW4                      OK
        Button SW5                      OK
        Button SW6                      OK
SPI & GPIO ports test                           Done
    ** Results for GPIO pins in input mode **
        PTA_6                           OK
        PTA_7                           OK
        PTA_8                           OK
        PTA_9                           OK
        PTA_10                          OK
        PTA_11                          OK
        PTA_24                          OK
        PTA_25                          OK
        PTA_26                          OK
        PTA_27                          OK
        PTA_28                          OK
        PTA_29                          OK
        PTE_28                          OK
    ** Results for GPIO pins in output mode **
        PTA_6                           OK
        PTA_7                           OK
        PTA_8                           OK
        PTA_9                           OK
        PTA_10                          OK
        PTA_11                          OK
        PTA_24                          OK
        PTA_25                          OK
        PTA_26                          OK
        PTA_27                          OK
        PTA_28                          OK
        PTA_29                          OK
        PTE_28                          OK
```

```
      ** Results for automatic test of LEDs **
         LED_D9                          OK
         LED_D10                         OK
         LED_D11                         OK
         LED_D12                         OK
================================================
*** Press any key (except C and R) to continue ***
```

Po výpisu výsledků testů čeká testovací firmware na zaslání libovolného znaku (vyjma velkých písmen C a R), čímž se pomyslný testovací okruh uzavře, proces je vrácen na začátek a menu s možností volby testů je opět vypsáno na výstup. Lze pokračovat výběrem dalšího testu.

Vytvořené testovací prostředí není složité na obsluhu a otázkou k zamyšlení může být, do jaké míry je uživatelsky přívětivé. Určitě se najdou uživatelé, kteří preferují propracované grafické rozhraní. Výsledné textové uživatelské rozhraní jim tak může připadat poněkud zastaralé. Avšak zvolené TUI[1] má velkou výhodu v minimálních požadavcích na běh programu (viz 3.3), což v závěru může vést k jednoduché přenositelnosti na různé další platformy a operační systémy. Vzhledem k rozsahu a minimálnímu nutnému množství vstupních operací během testování by složitější grafické uživatelské rozhraní ani nedávalo smysl.

Menší komplikace se během řešení vyskytly při ověřování správnosti implementovaných testů. Ukázalo se, že z náhodně vybraných 4 zapůjčených výukových kitů nebyl ani jeden plně funkční. Naštěstí se však jejich chyby vzájemně vylučovaly, tudíž řešení bylo možné dokončit.

Práce splnila svým obsahem cíle zadání a dá se říct, že i svůj účel, jelikož se již během tvorby podařilo odhalit několik vadných FITkitů3.

[1] TUI – Textové uživatelské rozhraní.

Kapitola 6

Závěr

V tomto textu byla uvedena motivace a smysl práce v rámci úvodní kapitoly. Následoval popis výukové platformy FITkit3 včetně jeho hardwarového vybavení a detailnějšího popisu osazeného mikrokontroléru. V rámci další kapitoly došlo k vysvětlení návrhu řešení podle zadání této práce. Dále byl na řadě popis implementace jednotlivých částí řešení, od tvorby testovacího firmwaru přes realizaci podpůrného hardwaru až po implementaci pomocného softwaru. Na závěr práce byla uvedena ukázka průběhu jednoho celého souboru testů s jejich popisem a zhodnocením.

Výsledkem práce je vznik kompletní sady testovacích nástrojů, která umožňuje snadno odhalit chyby v hardwaru výukového kitu. Jelikož jsou testy navrženy tak, aby šly jednoduše vylepšit o další funkcionalitu, určitě by se v budoucnu práce dala rozšířit například přidáním testovací sady pro FlexTimer a jiné nezahrnuté časovače, analogově/digitální i digitálně/analogové převodníky, audio či Ethernet moduly, anebo by mohla zaměřit svou pozornost třeba k testování samotného FPGA.

Rámec tohoto díla zasahuje do velké škály různých odvětví informatiky, od návrhu elektrického schématu a plošného spoje přes nízkoúrovňové programování v jazyce symbolických adres až po tvorbu uživatelského rozhraní na stolních počítačích. Nejen díky tomu mi tvorba bakalářské práce přinesla spoustu nových užitečných znalostí. Protože byl splněn důležitý požadavek na vytvoření sady testovacích nástrojů pro výukový kit, mohla by tato práce najít skutečné uplatnění v praxi, například pomoct s hledáním funkčních kusů k zařazení do výuky.

Literatura

[1] Cole, N. *Npyscreen's documentation*. 2014 [cit. 20.5.2020]. Revision d9e2fee986f3. Dostupné z: https://npyscreen.readthedocs.io/.

[2] Denk, F. *Samočinné testování mikrokontrolerů*. Brno, CZ, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/21576/.

[3] Fakulta informačních technologií, VUT v Brně. *MINERVA KIT - výuková a experimetální platforma s MCU a FPGA*. 2020 [cit. 20.5.2020]. Dostupné z: https://minerva.php5.cz/.

[4] Free Software Foundation, Inc. *GNU make manual*. 0.75. 17.1.2020 [cit. 20.5.2020]. Dostupné z: https://www.gnu.org/software/make/manual/make.html.

[5] Free Software Foundation, Inc. *Using the GNU Compiler Collection (GCC)*. 2017 [cit. 20.5.2020]. Dostupné z: https://gcc.gnu.org/onlinedocs/gcc-6.5.0/gcc/.

[6] Free Software Foundation, Inc. *Debugging with GDB*. 10. vyd. 2020 [cit. 20.5.2020]. Dostupné z: https://sourceware.org/gdb/current/onlinedocs/gdb/.

[7] Freescale Semiconductor Inc.. K60 Sub-Family Data Sheet. In:. 7. vyd. únor 2013. Dostupné z: https://www.nxp.com/docs/en/data-sheet/K60P144M100SF2.pdf.

[8] Freescale Semiconductor Inc.. MC9S08JM60 Series Data Sheet. In:. 5. vyd. říjen 2014. Dostupné z: https://www.nxp.com/docs/en/data-sheet/MC9S08JM60.pdf.

[9] Freescale Semiconductor Inc.. K60 Sub-Family Reference Manual. In:. 2.6.2012. Dostupné z: https://www.nxp.com/docs/en/reference-manual/K60P144M100SF2V2RM.pdf.

[10] Liechti, C. *PySerial's documentation*. 2017 [cit. 20.5.2020]. Revision a27715f3. Dostupné z: https://pyserial.readthedocs.io/en/latest/.

[11] Microchip Technology Inc.. 16-Bit I/O Expander with Serial Interface. In:. 2016. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/20001952C.pdf. ISBN 978-1-5224-0755-3.

[12] NXP Semiconductors. *NXP and Freescale Announce Completion of Merger*. 2015-12-07 [cit. 20.5.2020]. Dostupné z: https://investors.nxp.com/news-releases/news-release-details/nxp-and-freescale-announce-completion-merger/.

[13] Python Software Foundation. *Subprocess's documentation*. 2020 [cit. 20.5.2020]. Dostupné z: https://docs.python.org/3/library/subprocess.html.

[14] STMicroelectronics. M25P40 Data Sheet. In:. 16.1.2002. Dostupné z: https://minerva.php5.cz/doc/datasheet/FLASHM25P40.pdf.

[15] Styger, E. *Command Line Programming and Debugging with GDB*. 25.3.2015 [cit. 20.5.2020]. Dostupné z: https://mcuoneclipse.com/2015/03/25/command-line-programming-and-debugging-with-gdb/.

[16] Texas Instruments Inc.. TPS65251 Synchronous Step-Down Three Buck Switcher With Integrated FET Data Sheet. In:. únor 2018. Dostupné z: https://www.ti.com/lit/ds/symlink/tps65251.pdf.

[17] Xilinx. *What is an FPGA? Field Programmable Gate Array*. [cit. 20.5.2020]. Dostupné z: https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html.

Příloha A

Obsah přiloženého paměťového média

Základní adresářová struktura na přiloženém paměťovém médiu obsahuje:

/Software/	Obslužný software `FITkit3Tester.py` pro obsluhu testování z PC.
/Firmware/	Testovací firmware (viz níže).
/Hardware/	Data přídavného testovacího modulu (viz níže).
/Text/	Zdrojové soubory textu práce pro LaTeX a vygenerovanou barevnou i černobílou PDF verzi pro tisk (v názvu obsahuje „print“).

Veškerý obsah vytvořený za účelem této práce podléhá licenci, která je v daných adresářích umístěna v souboru `COPYING`.

Obsah adresáře Firmware

Podadresáře zde umístěné obsahují:

src/	Vytvořené zdrojové soubory v jazyce C.
includes/	Přiložené hlavičkové soubory jazyka C vygenerované Kinetis Design Studiem (KDS)[1].
tools/	Vyjmutý doplněk z KDS – PE Micro GDB server[2] pro vlastní použití.
obj/	Objektové soubory vytvořené překladačem.
doc/	Dokumentaci zdrojových kódů vygenerovaná Doxygenem[3].
bin/	Přeložený výsledný binární soubor `FITkit3Tests.elf` obsahující testovací firmware pro FITkit3.

[1] https://www.nxp.com/design/designs/design-studio-integrated-development-environment-ide:KDS_IDE

[2] http://www.pemicro.com/products/product_viewDetails.cfm?product_id=15320151&productTab=1

[3] http://www.doxygen.nl/

Obsah adresáře Hardware

Významy souborů zde umístěných jsou následující:

LightPipe.png	Snímek náhledu 3D modelu Světlovodu.
LightPipe.scad	3D model Světlovodu ve formátu SCAD[4].
LightPipe.stl	3D model Světlovodu ve formátu STL připravený pro tisk.
Schema.pdf	Schéma zapojení přídavného testovacího modulu.
PCB.pdf	Deska plošného spoje podpůrného testovacího modulu.
PCBGerber.pdf	Deska plošného spoje ve formátu Gerber[5] pro možnost výroby.
PCB.json	Deska plošného spoje ve formátu aplikace EasyEDA[6].

[4] http://www.openscad.org/
[5] https://en.wikipedia.org/wiki/Gerber_format
[6] https://easyeda.com/

Příloha B

Uživatelský manuál k testovacímu firmwaru pro FITkit3

Jak již bylo zmíněno v kapitole 3.1, testovací firmware je vytvořený převážně v programovacím jazyce C. Z toho důvodu je zapotřebí doplňující software[1] pro mikrokontroléry řady ARM Cortex-M4 k překladu firmwaru ze zdrojových kódů.

V rámci příloh (viz příloha A) je již spustitelný soubor s přeloženým firmwarem dostupný. Stačí jej tedy nahrát do FITkitu3 pomocí ladicího softwaru. K tomu lze využít například program `GNU/GDB`[2], který je zahrnutý v rámci již zmíněné překladové sady programů od společnosti ARM. Program `GNU/GDB` je v tomto případě využit jen jako klientská aplikace a k připojení na FITkit3 potřebuje ještě zprostředkovatele – server, jenž vytvoří pomyslný komunikační most mezi rozhraním programu `GNU/GDB` a rozhraním ladicího MCU na kitu.

Jako jedna z možností pro zařízení ARM se nabízí program `PEmicro GDB server`[3]. Jde sice o doplněk do vývojového prostředí Eclipse[4], ale lze jej provozovat i samostatně. Stažení je zdarma, avšak pro umožnění je nejprve potřeba se na oficiálních stránkách zaregistrovat. Pro usnadnění byla mezi přílohy k této bakalářské práci přidána jeho omezená verze[5].

Aby byla obsluha testovacího firmwaru co možná nejjednodušší, byl v rámci práce vyvinut obslužný software (viz příloha C.3), který stačí jen spustit a o veškeré nastavení komunikace se již postará za uživatele. Jeho použití je doporučeno. Avšak dříve zmíněný doplňující ladicí software je nutný pro nahrávání testovacího firmware do výukového kitu a výjimku tak netvoří ani použití obslužného softwaru!

B.1 Překlad testovacího firmwaru

K překladu je mezi přiloženými zdrojovými soubory dostupný také skript `Makefile` pro program `make`[6]. Proces vzniku skriptu je detailněji popsán v kapitole 4.1. Lze jej využít k au-

[1] https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads

[2] https://www.gnu.org/software/gdb/

[3] http://www.pemicro.com/products/product_viewDetails.cfm?product_id=15320151&productTab=3

[4] https://www.eclipse.org/

[5] `PEmicro GDB server` je umístěn v adresáři `/Firmware/tools/` a podporuje pouze jediný typ mikrokontroléru, který je osazen na platformě FITkit3.

[6] https://www.gnu.org/software/make/

tomatickému překladu všech dílčích zdrojových souborů a k sestavení výsledného spustitelného souboru s firmwarem. Vše je dostupné pod specifickými návěštími[7]. Součástí požadavků pro překlad je tedy nutné mít funkční program `make` a překladovou sadu programů pro ARM zmíněnou v úvodním odstavci této kapitoly (viz příloha B).

V prvním kroku je potřeba otevřít soubor `Makefile` a přenastavit výchozí umístění pro překladové programy. V následujícím úryvku je zkráceně vyjmuta pasáž ze začátku souboru určená k úpravě:

```
#
# == Settings of Makefile...
# TODO !!! Setup these paths according to your installation...
#
ARM_GPP := arm-none-eabi-g++
ARM_GCC := arm-none-eabi-gcc
ARM_GDB := arm-none-eabi-gdb
PE_GDB_SERVER := ../tools/pemicro*/lin/pegdbserver_console
#SEGGER_GDB_SERVER := JLinkGDBServer
```

Proměnné začínající `ARM_*` musí obsahovat cesty ke spustitelným programům (`g++`, `gcc`, `gdb`) z doplňující překladové sady. Dále je potřeba vyplnit platnou cestu ke spustitelnému programu `PEmicro GDB server` do odpovídající proměnné. Volitelně lze tento řádek zakomentovat vložením znaku `#` na začátek řádku a nastavit alternativní serverový program pro `GNU/GDB`.

Přizpůsobit si lze také vlastnosti samotného překladu, jak ukazuje následující vyjmutý úryvek:

```
# You can change the optimization level of program
OPTIMIZATION_LEVEL := 0 # 0-3
ELF_NAME := ../bin/FITKit3Tests.elf
FREESCALE_DEVICE := Freescale_K6x_K60DN512M10
PE_GDB_SERVER_INTERFACE := USBMULTILINK
```

Proměnná `OPTIMALIZATION_LEVEL` může nabývat hodnot celých čísel v rozmezí 0-3. Jedná se o optimalizační úroveň při překladu[8]. Testovací firmware je naprogramován tak, aby byl schopen fungovat ve všech úrovních optimalizace. Proměnná `ELF_NAME` obsahuje cestu a název souboru s výsledným spustitelným firmwarem. Pokud k tomu není opodstatněný důvod, není potřeba do její hodnoty zasahovat. Poslední dvě uvedené proměnné se vztahují k programu `PEmicro GDB server` a slouží k určení typu zařízení (FITkitu3) a rozhraní, ke kterému je výukový kit připojován při nahrávání či spouštění firmwaru. Taktéž by neměl být důvod tyto hodnoty nijak měnit.

[7] Návěští u vstupního skriptu programu `make` slouží k oddělení určitého bloku příkazů do podskupiny, kterou lze mimo jiné samostatně spouštět.

[8] https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

Finální vyjmutý úryvek z přiloženého souboru `Makefile` obsahuje výchozí příkazy:

```
# Other commands, you may adjust them too
RM := rm -rf
ifdef PE_GDB_SERVER
KILL_GDB_SERVER_CMD := pkill -f pegdbserver
else
KILL_GDB_SERVER_CMD := pkill -f JLink
endif
```

Volitelně lze přizpůsobit, například podle operačního systému nebo platformy, na které je překlad prováděn. Určitě je dobré pohlídat, aby příkaz pro ukončení `PEmicro GDB serveru` (případně jiného serverového programu) byl funkční a ve správném tvaru!

Po nastavení všech potřebných proměnných v souboru `Makefile` je možné použít příkaz `make` bez dodatečných parametrů. Program sám zajistí překlad všech nutných zdrojových souborů a pokud vše dopadne bez chyby, vytvoří spustitelný soubor s firmwarem, který jen možné nahrát do výukového kitu (viz příloha C.4). Pro úplnost existuje také návěští `clean` starající se o úklid všech překladových souborů.

B.2 Provoz testovacího firmwaru bez obslužného softwaru

Testovací firmware lze provozovat i bez použití obslužného softwaru. Jedinou nevýhodou bude mírně zhoršená přehlednost výstupu, jelikož testovací firmware využívá jednu nestandardní funkcionalitu pro odřádkování přičemž obslužný software ji umí zpracovat. Jedná se o znak „backspace“ (`\b`) s hodnotou 8 ve standardní ASCII[9] tabulce. Tento znak je testovacím firmwarem používán pro smazání vždy posledního řádku výstupu, kde se nachází informativní zpráva pojednávající o možnosti restartovat MCU či ukončit právě probíhající test (zasláním znaku velkého písmene `R` či `C`). Vlastnost automatického mazání posledního řádku by měla fungovat při interpretaci výstupu firmwaru v libovolném programu. Obslužný software však navíc umožňuje touto sekvencí mazat i předešlé odřádkování (znaky konce řádku), díky čemuž může kurzor na výstupu „cestovat do historie“. Důvod je prostý – vždy před zasláním informativní zprávy je nejprve zaslán znak pro odřádkování, aby byl celkový výstup přehlednější. Díky speciální funkcionalitě obslužného softwaru je tento prázdný řádek automaticky smazán při následujícím tisku nového výstupu. Důvodem jsou zaslané úvodní mazací sekvence. Při obsluze testů tak nedochází k prokládání výstupních informací prázdnými řádky.

K provozu testovacího firmwaru bez obslužného softwaru je nutno nastavit parametry komunikace a zvolit příslušný terminálový program. Návod je popsán v následujících odstavcích.

Parametry sériové komunikace

- Rychlost přenosu 115 200 baudů,
- 1 start bit a 1 stop bit s délkou slova 8 bitů,
- bez kontroly parity,
- bez softwarového či hardwarového řízení toku (anglicky flow control).

[9] ASCII – Kódová tabulka pro znaky anglické abecedy a jiné speciální znaky využívané v informatice (viz https://cs.wikipedia.org/wiki/ASCII).

Alternativní program pro komunikaci

Vhodnou alternativou pro ovládání testů a asynchronní sériovou komunikaci s FITkitem3 může být například program `picocom`[10]. Z výše uvedených parametrů komunikace vyplývá, že navázání spojení pomocí tohoto programu může být následující:

```
~$ picocom -b 115200 <zařízení>
```

Za „zařízení“ je nutné dosadit cestu nebo označení sériového portu, kam je připojen FITkit3. Cesta k zařízení se bude lišit podle platformy a operačního systému.

Nahrávání testovacího firmware do FITkitu3

Pro automatické nahrávání testovacího firmware do výukového kitu lze využít předem připravený `Makefile`, který je součástí příloh umístěný mezi zdrojovými soubory. Je nutné mít však dostupný program `make` a připravený skript správně nastavit (viz sekce B.1). Zmíněný `Makefile` podporuje dva druhy návěští – `flash` a `run`, přičemž první (`flash`) se postará o nahrání testovacího firmwaru do FITkitu3 a druhé (`run`) o jeho spuštění. Po nahrání je výukový kit restartován a provádění programu je zastaveno, proto musí dojít k jeho opětovnému spuštění již zmíněným druhým návěštím. Druhou možností je FITkit3 odpojit a připojit k PC, čímž dojde k automatickému spuštění programu. V obou zmíněných případech využívá program `make` pro svou činnost programy `GNU/GDB` a `PEMicro GDB server`, které je nutné mít také dostupné.

Existuje i manuální možnost, jak nahrát testovací firmware do FITkitu3. Z principu by se však jednalo jen o popis jednotlivých kroků, které jinak udělá program `make` podle předepsaného skriptu `Makefile`. Lze tedy buď nahlédnout do tohoto souboru, anebo navštívit kapitolu 4.1 pojednávající o jeho vzniku.

B.3 Programová dokumentace a její generování

Součástí přiložených zdrojových kódů je také soubor `Doxyfile`, který slouží pro konfiguraci programu `Doxygen`[11]. Programové komentáře jsou v rámci všech zdrojových kódů formátovány v notaci tohoto programu, což umožňuje automaticky vygenerovat přehlednou dokumentaci ve formátu HTML. Pokud je dostupný program `make` a správně nastavený přiložený `Makefile` (viz 4.1), lze využít návěští `doc` pro spuštění `Doxygen`u. Výsledná dokumentace v HTML formátu se pak nachází v adresáři `/Firmware/doc/`.

B.4 Popis obsluhy testovacího firmwaru

Základním stavebním kamenem je úvodní menu s možností volby testu. Výběr se provádí zasláním odpovídajícího číselného znaku skrze sériovou komunikaci – při použití obslužného softwaru postačí stisk klávesy s daným číslem. Volba čísla `0` způsobí spuštění všech dostupných testů postupně v řadě za sebou. Čísla `5` a `8` spouští zase odpovídající sadu testů pro všechny časovače respektive vstupně/výstupní porty. V průběhu testování lze v libovolnou chvíli přerušit aktuálně běžící test zasláním znaku velkého písmene `C`, anebo zcela restartovat všechny testy posláním znaku velkého písmene `R`. Při restartu dojde k vymazání výsledků předchozích testů a k obnovení veškerého nastavení v rámci testování na kitu.

[10] https://github.com/npat-efault/picocom
[11] http://www.doxygen.nl/

Ilustrace spuštění testu reproduktoru krok za krokem

1. FITkit3 byl připojen do USB obslužného PC.
2. Spuštěn přístupový terminál `picocom` a navázáno spojení (viz B.2).
3. Spuštěn nahrávací skript příkazem `make flash` (viz B.2).
4. Nahrávání proběhlo úspěšně. Spuštěn nahraný firmware na kitu příkazem `make run`.
5. Úvodní výstup v přístupovém terminálu naznačuje, že vše funguje a bylo nastaveno správně:

```
Welcome to Routines for testing modules of MCU (MK60DN512ZVMD10)
================================================================
          The FITkit3 Tester by Petr Stehlik (c) 2020

Option menu:
[0] - run All tests
[1] - run only Speaker test
[2] - run only RTC timer test
[3] - run only LPTMR timer test
[4] - run only PIT timers test
[5] - run all Timers tests
[6] - run only Buttons test
[7] - run only SPI & GPIO test
[8] - run all I/O tests

== Press C to cancel the test or R to restart the MCU ==
```

6. Zaslán znak čísla `1` stiskem odpovídající klávesy na klávesnici.
7. Testovací firmware spustil test reproduktoru a postupně posílá výstup:

```
Testing speaker... Listen!
Testing speaker... Done

              *** Tests results ***
================================================
Speaker tests                                Done
================================================
*** Press any key (except C and R) to continue ***
```

8. Na výstupu je zřejmé, že test proběhl úspěšně a čeká na stisk libovolné klávesy (s výjimkou tiskacích písmen `R` a `C`, které slouží pro restart kitu či přerušení právě probíhajícího testu). Po stisku libovolné klávesy se opět vypíše menu s možností volby spuštění dalšího testu.
9. Zaslán znak `R` pro restart, což při spuštění příkazem `make run` v kroku 4 způsobí zastavení provádění programu na FITkitu3 a jeho odpojení od `GNU/GDB`. Výukový kit tak může být bezpečně odpojen od PC.

Příloha C

Uživatelský manuál k obslužnému softwaru pro PC

Jak již bylo zmíněno v kapitole 3.3, testovací software `FITkit3Tester.py` je napsaný ve skriptovacím jazyce Python3[1] a vyžaduje ke svému běhu rozšiřující knihovny, které nejsou součástí standardní distribuce `CPython` implementace. Těmito nestandardními knihovnami jsou `npyscreen`[2] a `pySerial`[3], jejichž instalaci lze provést například pomocí repozitáře `PyPi` podle návodu na webových stránkách uvedených pro každou knihovnu v poznámce pod čarou. Dále pro svůj běh vyžaduje také moduly `sys`[4], `os`[5], `getopt`[6], `subprocess`[7], `threading`[8] a `curses`[9].

Vytvořený software byl otestován ve verzi interpreteru `Python 3.6.9` na operačním systému GNU/Linux s verzí kernelu 4.15.0-99-generic. Testované rozšiřující knihovny byly ve verzích `pySerial 3.4` a `npyscreen 4.10.5`.

C.1 Spuštění obslužného testovacího softwaru

Před prvním spuštěním softwaru je nutné provést následující kroky:

1. Pokud není již dostupný, je nutné provést instalaci prostředí jazyka Python3. Je vhodné zvolit implementaci CPython, která je ke stažení na oficiálních stránkách jazyka.

2. Je potřeba stáhnout a nainstalovat rozšiřující knihovny, které testovací software využívá (jsou zmíněny v úvodu této kapitoly).

3. Je doporučeno zkontrolovat, zdali jsou dostupné všechny ostatní požadované moduly ke spuštění. Zde může nastat problém s modulem `curses`, jenž nemusí být ve výchozím stavu instalace prostředí jazyka Python dostupný na operačních systémech

[1] Oficiální stránky jazyka Python jsou https://www.python.org/.
[2] https://pypi.org/project/npyscreen/
[3] https://pypi.org/project/pyserial/
[4] https://docs.python.org/3/library/sys.html
[5] https://docs.python.org/3/library/os.html
[6] https://docs.python.org/3/library/getopt.html
[7] https://docs.python.org/3/library/subprocess.html
[8] https://docs.python.org/3/library/threading.html
[9] https://docs.python.org/3/howto/curses.html

Microsoft Windows. Existuje proto projekt `windows-curses`[10] s cílem zprostředkovat základní podporu tohoto modulu.

4. Je vhodné si také připravit skript pro nahrávání testovacího firmwaru do FITkitu3. Potřebné informace včetně postupu jsou popsány v sekci C.4.

Pokud jsou splněny všechny požadavky před prvním spuštěním, měl by již být obslužný testovací software schopný běhu v prostředí jazyka Python3. V sekci C.3 je popsáno základní použití a obsluha softwaru.

C.2 Vstupní parametry testovacího softwaru

Seznam vstupních parametrů programu je následující:

--help	Vypíše nápovědu programu.
--version	Vypíše informace o verzi a licenci programu.
--autotests	Povolí automatizaci testovacího procesu.
--flashscript="S"	Určí externí skript, který při spuštění nahraje testovací firmware do FITkitu3. Jak vytvořit takový skript je popsáno v sekci C.4. Cestu ke spustitelnému souboru se skriptem (včetně jeho parametrů oddělených mezerami) lze dosadit za řetězec `S`, přičemž každý výskyt podřetězce „FLASHDEVICE“ v řetězci `S` je před spuštěním skriptu nahrazen názvem portu s připojeným právě testovaným zařízením.
--forceflash	Před každým spuštěním testovacího procesu nahraje testovací firmware do FITkitu3 bez ohledu na jeho předchozí přítomnost. Použití má smysl jen v kombinaci s parametrem `--flashscript`.

Program podporuje také zkrácený zápis parametrů, avšak ty zde pro přehlednost nejsou uvedeny. Všechny parametry lze libovolně kombinovat. Parametry `--help` a `--version` po výpisu daných informací program ihned ukončí. Pro základní funkčnost programu není nutné používat žádný parametr.

C.3 Obsluha testovacího softwaru

Je důležité mít na paměti, že zvolené textové uživatelské rozhraní má jednu zásadní vlastnost – přepínání mezi jednotlivými prvky (tlačítky, seznamem či textovým terminálem) se provádí klávesou `Tab`. V některých případech se může stát, že přepínání mezi prvky rozhraní jde uskutečnit například i kurzorovými šipkami. Nicméně použití klávesy `Tab` bude vždy funkční jistotou.

Po spuštění programu je uživatel přiveden na výchozí dialog (viz obrázek C.1), který obsahuje seznam nalezených zařízení. Při potvrzení výběru software zkontroluje (viz sekce C.5) dostupnost testovacího firmwaru v zařízení. Pokud není dostupný a uživatel zadal při spuštění programu prostřednictvím parametru `--flashscript` platný spustitelný skript, je skript

[10] https://pypi.org/project/windows-curses/

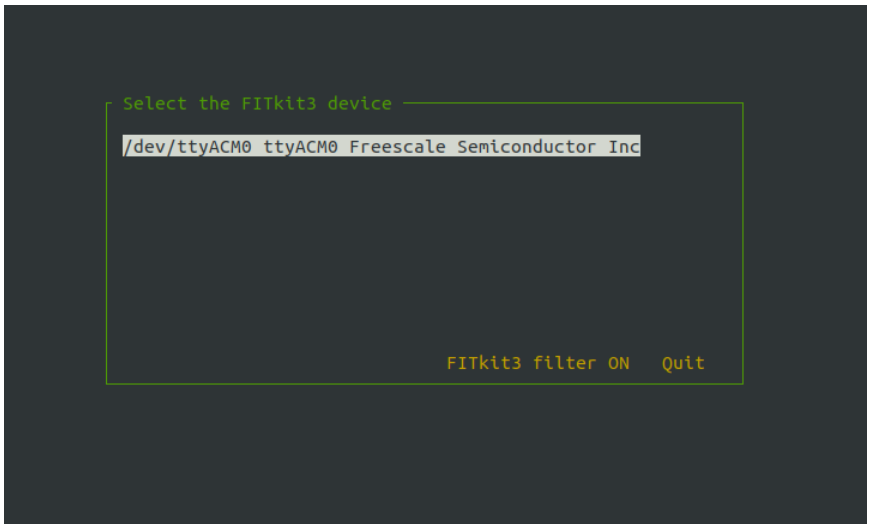
spuštěn a testovací firmware jeho prostřednictvím nahrán do FITkitu3. Následuje navázání komunikace s výukovým kitem a zobrazení přístupového terminálu k obsluze testů za předpokladu, že všechny předchozí kroky proběhly úspěšně a FITkit3 má funkční komunikační modul. Různé typy chybových scénářů ilustrují nadcházející podsekce.

Výběr zařízení

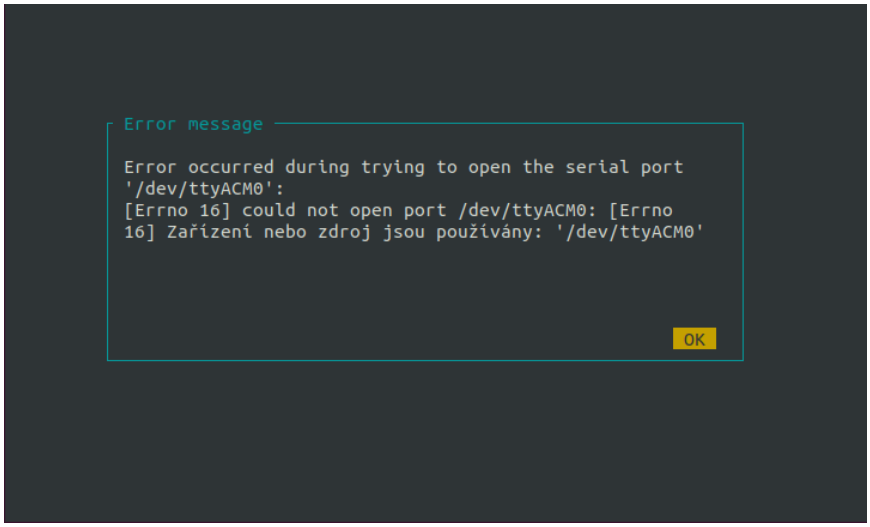
Ke standardnímu vyhledávání dostupných zařízení byl přidán navíc speciální filtr s přepínatelným (ne)aktivním stavem. Ve vypnutém stavu se do výběrového seznamu dostanou všechna detekovaná zařízení na sériových portech. Avšak při zapnuté činnosti filtru se v seznamu zobrazí pouze zařízení splňující dvě podmínky. Pro první z nich platí, že hodnota identifikačního čísla prodejce (anglicky VendorID) musí být číslo 5538 (v hexadecimální soustavě se jedná o hodnotu 15A2). Druhou podmínkou je nalezení slova „Freescale“ v obsahu případného textového řetězce s názvem výrobce. Zmíněné údaje odpovídají typickým identifikačním údajům pro FITkit3 a jedná se o informační údaje dostupné pro rozhraní USB, kterým je výukový kit připojen k počítači.

Některé operační systémy mohou při připojení nového zařízení na rozhraní USB spustit automatickou detekci a vyhledávání ovladačů. Proces detekce trvá často i několik sekund přičemž po tuto dobu není zařízení pro obslužný software dostupné. Proto je žádoucí chvíli vyčkat a poté pokus o připojení opakovat. V případě neúspěšného spojení je ihned uživateli zobrazena na obrazovce příslušná informující zpráva (viz obrázek C.2).

Výběr zařízení se provede pohybem kurzoru v seznamu pomocí kurzorových šipek a potvrdí stiskem klávesy `Enter`. Dále následují ukázkové snímky obrazovek obslužného testovacího softwaru s popisem.



Obrázek C.1: **Úvodní okno s možností výběru zařízení.** Kurzorové šipky na klávesnici umožňují pohyb nahoru a dolů v seznamu zařízení. Pokud program nedetekuje žádné právě připojené zařízení, místo seznamu je zobrazena pouze odpovídající hláška. Na pravém dolním okraji okna se nachází dvě tlačítka. Levé slouží k přepnutí stavu filtru dostupných zařízení a pravé k ukončení programu. Pro přechod mezi tlačítky a seznamem zařízení se využívá klávesa `Tab`.



Obrázek C.2: **Chybová zpráva oznamující neúspěšné spojení.** Ve zprávě je vždy vypsán port zařízení, kde došlo k chybě a následuje systémová zpráva specifikující daný problém. Jedinou výchozí možností je potvrdit hlášení stiskem klávesy `Enter`.

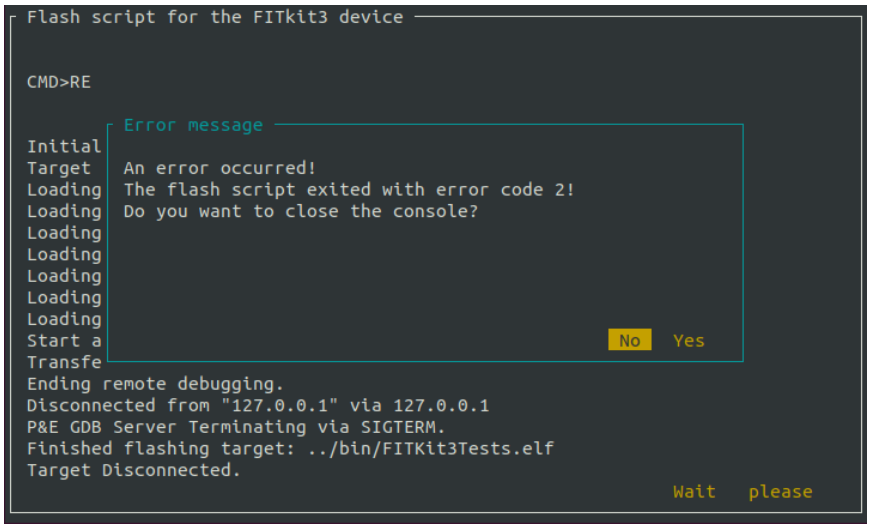
Automatické nahrávání testovacího firmwaru

Po výběru zařízení se obslužný testovací software pokusí detekovat přítomnost testovacího firmwaru v zařízení. Jestliže byl program spuštěn s platným parametrem `--flashscript` (viz sekce C.2) a testovací firmware není v zařízení dostupný, následuje potvrzení spuštění vybraného nahrávacího skriptu (viz obrázek C.3). Jeho výstup je poté v reálném čase zobrazován v novém okně.

Jestliže při nahrávání dojde k nějaké chybě ve skriptu, uživatel je informován příslušným hlášením včetně návratového chybového kódu (viz obrázek C.4).

Skript za běhu nelze přerušit ani ukončit, aby nedošlo k poškození výukového kitu. Teprve po úspěšném skončení nahrávání je uživateli umožněno pokračovat ke spuštění přístupového terminálu a obsluze testů skrze tlačítko v pravém dolním okraji okna.

Obrázek C.3: **Dotaz na spuštění externího skriptu pro automatické nahrávání testovacího firmwaru.** Při potvrzení dojde k otevření nového okna, kde je zobrazován výstup skriptu v reálném čase. U negativní reakce je uživatel vrácen na okno s výběrem zařízení.

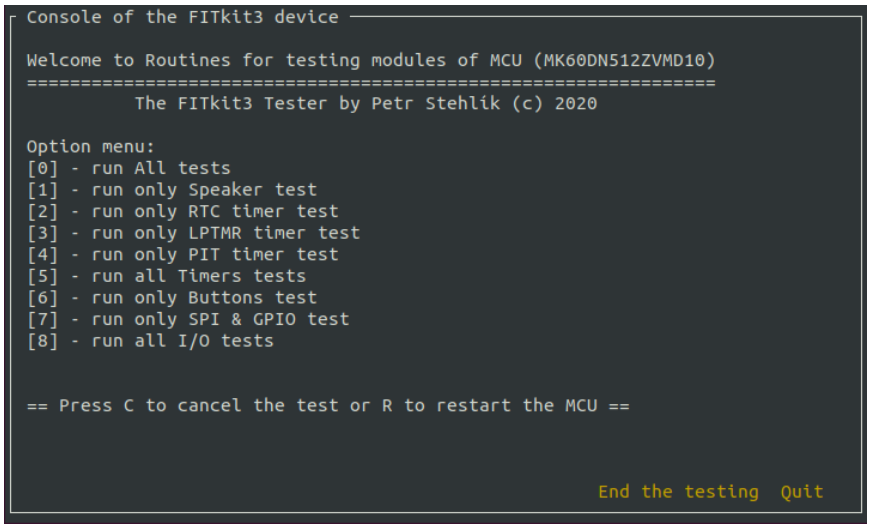


Obrázek C.4: **Zpráva informující o vzniku chyby při běhu externího skriptu.** Součástí obsahu je také chybový návratový kód. Celé okno lze buď ihned ukončit a vrátit se na výběr zařízení pozitivní odpovědí, anebo jen zavřít okno s hlášením, což se může hodit pro náhled výstupu skriptu a k nalezení vzniklé chyby.

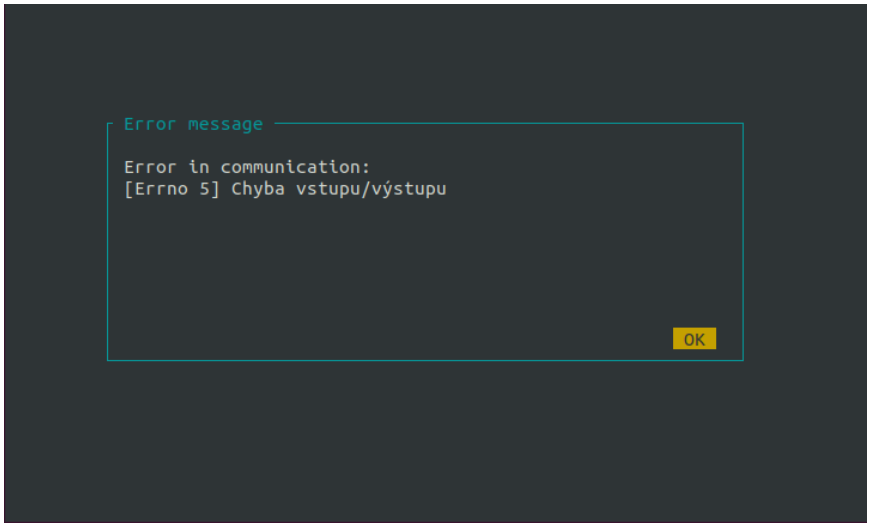
Spuštění testů na FITkitu3

Pokud je vybráno zařízení s dostupným testovacím firmwarem, je na řadě pokus o navázání komunikace a spuštění přístupového terminálu v novém okně. V opačném případě se zobrazí výzva k doplnění firmwaru (viz obrázky C.3 a C.5) a následuje možnost zvolit jiné zařízení. Popis obsluhy testování je shrnut u obrázku C.6, přičemž detailnější informace je možné nalézt v sekci B.4. Když během testování nastane chyba v komunikaci (viz obrázek C.7), zobrazí se příslušná zpráva a proces testování je ukončen s návratem na seznam výběru zařízení.

Obrázek C.5: **Zpráva informující o nepřítomnosti testovacího firmware na FITkitu3.** Výzva se zobrazí uživateli v případě, že programu nebyl zadán skript pro automatické nahrávání firmwaru a naznačuje, že by měl uživatel nahrát testovací firmware vlastním způsobem. Druhým případem zobrazení výzvy je úspěšné dokončení automatického nahrávacího skriptu a opakovaná chyba v detekci testovacího firmwaru. V nejčastějším případě souvisí problém s nefunkčním komunikačním modulem ve výukovém kitu.



Obrázek C.6: **Výchozí okno terminálu s úvodním výstupem z funkčního testovacího firmwaru.** V tuto chvíli program čeká na zvolení sady testů ke spuštění. Výběr lze provést stiskem klávesy s odpovídajícím číslem. Pod volbou čísla `0` se skrývá schopnost spustit všechny dostupné testy postupně za sebou. Obdobně číslo `5` a `8` spouští odpovídající sadu testů pro všechny časovače respektive vstupně/výstupní porty. Za zvláštní pozornost stojí znak velkého písmene `C`, jehož zasláním je možné přerušit aktuálně probíhající test. Následuje tak spuštění dalšího testu v pořadí. Přerušení se může hodit například pro zaseknutý test v důsledku nefunkční periferie či modulu. Všechny testy včetně jejich výsledků a nastavení je také možné jednorázově restartovat zasláním velkého písmene `R`.



Obrázek C.7: **Zpráva informující o vzniku chyby v komunikaci.** Ve většině případů se jedná o přerušení spojení, například v důsledku odpojení FITkitu3 z rozhraní USB. Potvrzení zprávy stiskem klávesy `Enter` vede k ukončení testování a k návratu na seznam s výběrem zařízení.

C.4 Tvorba skriptu pro automatické nahrávání testovacího firmwaru do FITkitu3

Jedná se o pomocný skript, který může být vytvořený v libovolném skriptovacím jazyce. Důležitou podmínkou je, že musí být jednoduše spustitelný (mimo jiné mít nastavený souborový příznak spustitelnosti) v daném operačním systému, například z příkazové řádky. Jelikož jej v případě potřeby spouští testovací software s využitím modulu `subprocess`, jenž je součástí jazyka Python3.

Hlavním úkolem externího skriptu je zprostředkovat nahrání chybějícího testovacího firmwaru do FITkitu3, aby obslužný software mohl po provedení zahájit ihned testování. Detailně jsou jednotlivé kroky nahrávání (včetně potřebného doplňkového softwaru) uvedeny v úvodu kapitoly B a dále v její sekci B.1. Vyčlenění procesu nahrávání z testovacího softwaru umožňuje jeho jednoduchou úpravu, rozšiřitelnost a přizpůsobení podle daného operačního systému i dostupných prostředků.

Součástí příloh je také experimentální skript `flash.sh`[11] pro interpretr Bourne shell[12]. Avšak jeho použití v přiložené formě je na vlastní riziko. V každém případě může posloužit jako inspirace při tvorbě vlastního skriptu na míru. Přiložený skript totiž předpokládá dostupné překladové prostředí pro mikrokontroléry typu ARM a správně nastavený `Makefile` s funkčním programem `make` (viz sekce B.1).

Podmínky pro skript k automatickému nahrávání testovacího firmwaru

- Skript **nesmí** očekávat žádná data na standardním vstupu, protože tento vstup není za běhu podporován. Naproti tomu veškerý standardní výstup je přesměrován do okna

[11] Umístění skriptu je v adresáři `/Software/`.

[12] https://en.wikipedia.org/wiki/Bourne_shell

testovacího softwaru, tudíž je vhodné prostřednictvím výstupu informovat uživatele o průběhu.

- V případě, že nastane libovolná chyba, **musí** skript vrátit chybový návratový kód (jiný než 0), aby došlo k zastavení testovacího procesu.
- Soubor se skriptem **musí** mít nastavené spustitelné příznaky.
- V rámci hodnoty parametru `--flashscript` obslužného softwaru lze zadat pouze jedna cesta ke spustitelnému souboru s případnými programovými parametry oddělenými mezerami. Z toho plyne, že je vhodné připravit pomocný skript v rozsahu jednoho souboru.
- Pro upřesnění zvoleného testovaného zařízení je možné využít vstupní parametry skriptu. Například přiložený ukázkový skript `flash.sh` očekává název portu se zvoleným zařízením jako první vstupní parametr. Do hodnoty parametru `--flashscript` může být vhodné vložit řetězec „FLASHDEVICE“ jako obecnou hodnotu vstupního parametru skriptu, protože tento řetězec je před jeho spuštěním nahrazen názvem portu právě zvoleného zařízení.
- Skript musí splňovat svůj účel – nahrát testovací firmware do zařízení.
- Před úspěšným skončením skriptu je doporučeno zaručit (například restartem zařízení a následnou vhodnou časovou prodlevou), že testovací firmware na zařízení již běží a je opět připojitelné, aby po skončení skriptu bylo možné ihned zahájit komunikaci.

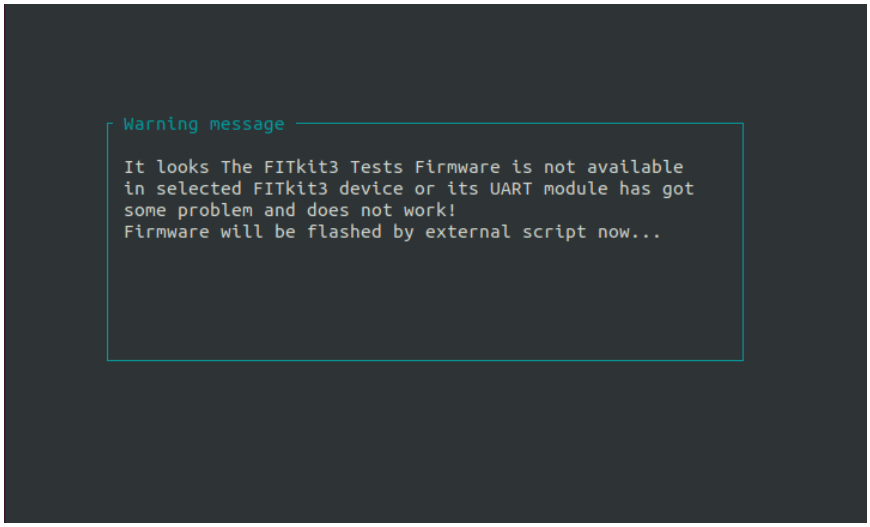
C.5 Automatizace testování

Pro spuštění testovacího softwaru v automatizovaném režimu je zapotřebí zadat vstupní parametr programu `--autotests`. Přepnutí režimu způsobí zrychlení testovacího procesu, obzvláště pokud je k obslužnému počítači připojeno současně vždy jen jedno testované zařízení.

Testovací software čeká po spuštění na připojení prvního zařízení a dále následuje okamžité zjištění dostupnosti testovacího firmwaru. Kdyby došlo na zablokování nově připojeného zařízení operačním systémem (viz obrázek C.2), je vhodné chvíli vyčkat a poté znovu ručně zvolit zařízení v seznamu. Jestliže nebyl testovací firmware detekován a obslužný software má k dispozici externí nahrávací skript, zobrazí se okno s informační zprávou (viz obrázek C.8) a po uplynutí patřičné prodlevy dojde ke spuštění automatického nahrávacího skriptu. I z tohoto důvodu je vhodné při použití automatizovaného režimu zadat platný programový parametr `--flashscript`.

Pokud již byl testovací firmware dostupný, anebo byl úspěšně nahrán, obslužný software ihned pokračuje k navázání komunikace se zařízením a automaticky zasílá znak číslice `0`, což vede ke spuštění všech testů postupně v řadě za sebou. Jediným nutným úkonem uživatele je uzavřít terminál s výstupem testů po jejich skončení. V případě výskytu chyby během procesu je zobrazena příslušná zpráva a testování ukončeno.

Jestliže je v seznamu dostupné současně více než jedno zařízení, veškerá automatizace funguje stejným způsobem s výjimkou toho, že si uživatel nejprve musí ručně zvolit, na kterém zařízení ze seznamu bude testování probíhat.



Obrázek C.8: **Časově omezená zpráva se 4 sekundovou prodlevou.** Její zobrazení nastane při výběru jednoho z více dostupných, anebo při detekci nově připojeného a dostupného právě jednoho zařízení. Informuje uživatele, že po jejím zmizení dojde ke spuštění automatického externího skriptu, který se postará o nahrání chybějícího testovacího firmwaru do výukového kitu. Podmínkou pro tento automatizovaný postup je spuštění testovacího softwaru s platnými parametry `--autotests` a `--flashscript`.

Díky automatizovanému režimu může být výsledný průběh testování velmi hladký. Stačí jen spustit obslužný testovací software s patřičnými parametry, připojit FITkit3 k počítači (pozor na zablokování nově připojeného zařízení operačním systémem), vyčkat na nahrání testovacího firmwaru, dále splnit případné požadavky během testování (například stiskem otestovat tlačítka na výukovém kitu) a na závěr si projít výsledky všech testů. Zbývá už jen potvrdit ukončení testovacího procesu a odpojit FITkit3. Celý scénář se tak může opakovat s dalším zařízením.