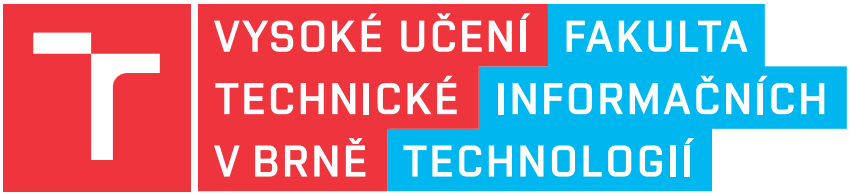


Dokumentace k projektu
Hra HAD na platformě FITkit 3
Mikroprocesorové a vestavěné systémy

2. prosince 2021 Michal Pyšík (xpysik00)

1 Úvod a použité prostředky

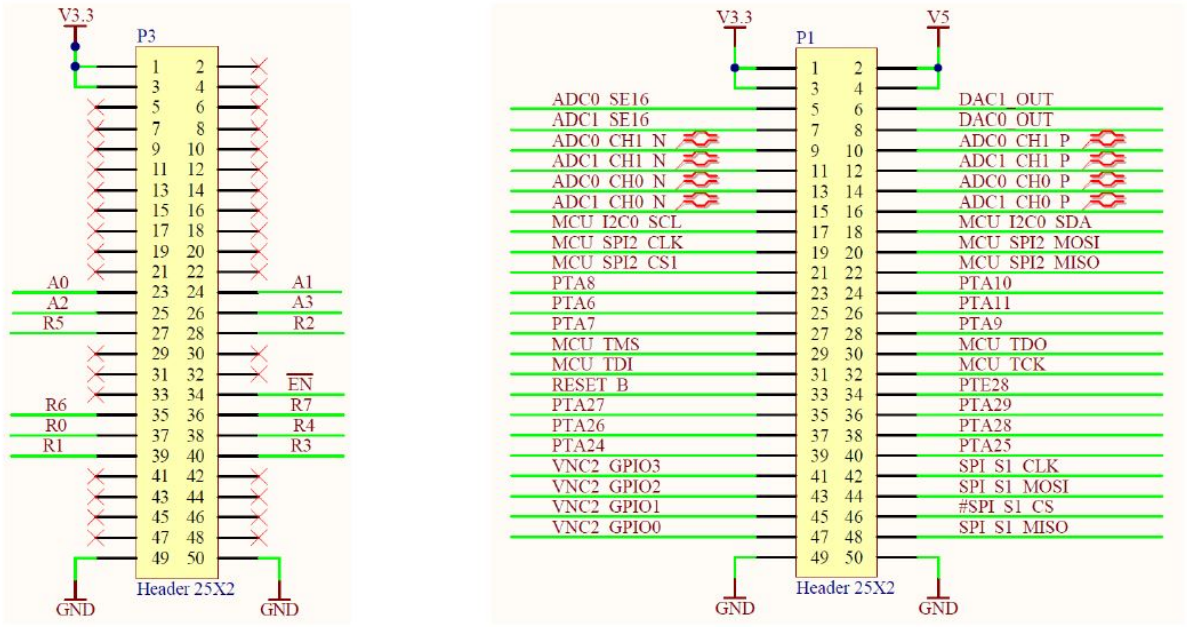
Cílem projektu bylo implementovat zjednodušenou verzi hry had na platformě **FITkit 3**. V této verzi hry má had pouze fixní délku a nesbírá žádné prvky, které by ho postupně zvětšovaly. Cílem hry je tedy pouze co nejdéle přežít. Samotná hra končí, když had narazí do hranice herního pole, nebo do svého vlastního těla.

1.1 Použitá platforma a externí prvky

Hlavním mikrokontrolérem kitu **FITkit 3** je Freescale Kinetis **MK60DN512ZVMD10**, obsahující jádro **ARM Cortex-M4**, jehož programováním se tento projekt zabývá. Pro vývoj bylo využito prostředí **Kinetis Design Studio**. Kromě samotného kitu byl využit externí modul s maticovým displayem (spojen ze dvou 8x8 displayů), obsahující celkem 16 sloupců a 8 řádků adresovatelných buněk. Maticový display je ovládán pomocí multiplexingu, tedy v daný moment je vybrán pouze jeden sloupec, na kterém svítí buňky odpovídající vybraným řádkům.

1.2 Schéma zapojení a ovládání displaye

Jak lze vidět na obrázku, konektor **P3** modulu obsahující maticový display je připojen na konektor **P1** FITkitu.



Obrázek 1: Schéma zapojení konektoru P3 externího modulu na konektor P1 FITkitu 3[1]

GPIO vývody portu A `PTA8`, `PTA10`, `PTA6`, `PTA11` jsou napojeny na řídící piny `A0`–`A3`, jejichž binární kombinace udává vybraný aktivní sloupec displaye. Další vývody tohoto portu `PTA7`, `PTA9` a `PTA24`–`PTA29` jsou připojeny k řádkovým anodovým vodičům `R0`–`R7`, které adresují jednotlivé řádky vybraného sloupce (log. 1 = svítí, log. 0 = nesvítí). Vývod `PTE28` portu E je napojen na pin $\overline{\texttt{EN}}$ značící negaci povolovacího signálu, tedy aby display svítil, musí zde být přivedena logická 0.

[1] Převzato z prezentace **IMP – projekt „HAD“**, Ing. Vojtěch Mrázek, Ph.D., mrazek@fit.vutbr.cz, 22.10.2020, Vysoké učení technické v Brně, Fakulta informačních technologií

1.3 Ostatní použité prvky

Kromě displaye použitého jako výstup je samozřejmě zapotřebí i jiných prvků, především tlačítek pro ovládání samotné hry uživatelem. Použil jsem tlačítka `BTN_SW2`–`BTN_SW6`, namapované na piny portu E `PTE10`, `PTE12`, `PTE27`, `PTE26` a `PTE11`. Pro periodickou obsluhu displaye bylo samozřejmě zapotřebí použít nějaký časovač, rozhodl jsem se pro Low-power timer (LPTMR).

2 Programová realizace

Samotný program pro MCU je napsán v jazyce C, a nachází se v souboru `main.c`. Tato kapitola je zaměřena na hlavní konstrukce a mechanismy použité v programu. Pro řešení byl jako kostra použit zdrojový soubor z ukázky `FITkit3-demo.zip` [2]. Všechny proměnné v rámci programu jsou globální, a všechny funkce (kromě `int main()`) jsou typu `void`.

2.1 Inicializace portů a časovače

Na začátku samotného programu dochází hned po inicializaci MCU funkcí `MCUInit()`, kterou jsem nijak neupravoval. Dále jsou inicializovány potřebné porty funkcí `PortsInit()`. Zde jsou zapnuty hodiny pro oba porty využívané programem, tedy port A a port E. Vybrané bity portu A řídí ovládání maticového displaye (výstup), zatímco port E má na starost vstup z pěti tlačítek, a navíc výstup v podobě $\overline{\text{EN}}$ signálu displaye. Pro oba porty jsou nastaveny odpovídající bity v registrech `PTx->PCR` na hodnotu 1, jelikož jsou použity jako `GPIO`. Nakonec jsou nastaveny piny v registrech `PTx->PDDR`, aby bylo zřejmé které piny jsou vstupní nebo výstupní, a všechny použité výstupní piny jsou v registrech `PTx->PDOR` pro začátek vynulovány. Časovač `LPTMR0` je inicializován funkcí `LPTMR0Init(int count)` převzaté ze zmíněného dema, zde jsem experimentálně dospěl k nastavení compare value na hodnotu 0, jelikož jakákoli vyšší hodnota vedla k nedostatečné obnovovací frekvenci displaye. Také jsem zkoušel experimentovat s odlišnými časovači.

2.2 Herní pole

Stav herního pole je uchováván v 2D poli `bool smatrix[16][8]`, kde hodnota `true` na daném indexu značí, že se zde nachází článek hada (dioda bude svítit), a velikost pole odpovídá velikosti displaye. Pomocná funkce `clearGameField(bool fill)` vyplní celé pole hodnotou `fill`, což je užitečné pro inicializaci začátku hry, ale také rozsvícení celého displaye poté, co hráč prohrál.

2.3 Had a jeho pohyb

Pozice hada v poli je uložena v instanci `snake` struktury `Snake`, která se skládá z polí `x[]` a `y[]` o délce `SNAKE_LEN` (délka hada, výchozí hodnota 8), sloužící pro uchování horizontálních/vertikálních souřadnic jednotlivých článků hada. Funkce `setDefaultSnake()` volána vždy po startu/resetu hry nastaví hada do jeho výchozí pozice v herním poli. Jeho pohyb je prováděn funkcí `moveSnake()`, která podle hodnoty proměnné `snake_dir` typu `DIRECTION_T` udávající směr pohybu zkontroluje, zda je odpovídající buňka pole volná. Pokud ano, celý had se posune, ale pokud se buňka nachází za hranicí pole nebo se v ní nachází článek hada, nastaví se hodnota proměnné `bool gameOver` na `true`, což způsobí konec hry a rozsvícení celého displaye.

2.4 Obsluha přerušení od časovače

Obsluha přerušení časovače funkcí `LPTMR0_IRQHandler()` ovládá samotný display. Nejprve jsou všechny výstupní piny pro ovládání displaye pomocí registru `PTA->PDOR` vynulovány. Proměnná `int col_select` slouží k výberu aktivního sloupce a její hodnota v rozmezí 0 a 15 je inkrementována při každé obsluze přerušení.

[2] **FITkit3-demo.zip**, Ing. Michal Bidlo Ph.D., bidlom@fit.vutbr.cz, 2019, Vysoké učení technické v Brně, Fakulta informačních technologií

Na 4 spodní bity této proměnné jsou namapovány bity registru `PTA->PDOR` řídící výber sloupce na displayi pomocí multiplexingu. Zbývající bity řídící zda svítí jednotlivé řádky displaye jsou poté nastaveny na základě hodnot v herním poli namapované na jejich pozici. Kromě blikání displaye musí časovač obsluhovat také časování pohybu hada. Proměnná `int period_cnt` je při každém přerušení časovače inkrementována, a když dosáhne hodnoty `PERIODS_PER_MOVE` (výchozí hodnota 384), vynuluje se a zavolá funkci `moveSnake()`.

2.5 Smyčka obsluhy vstupů

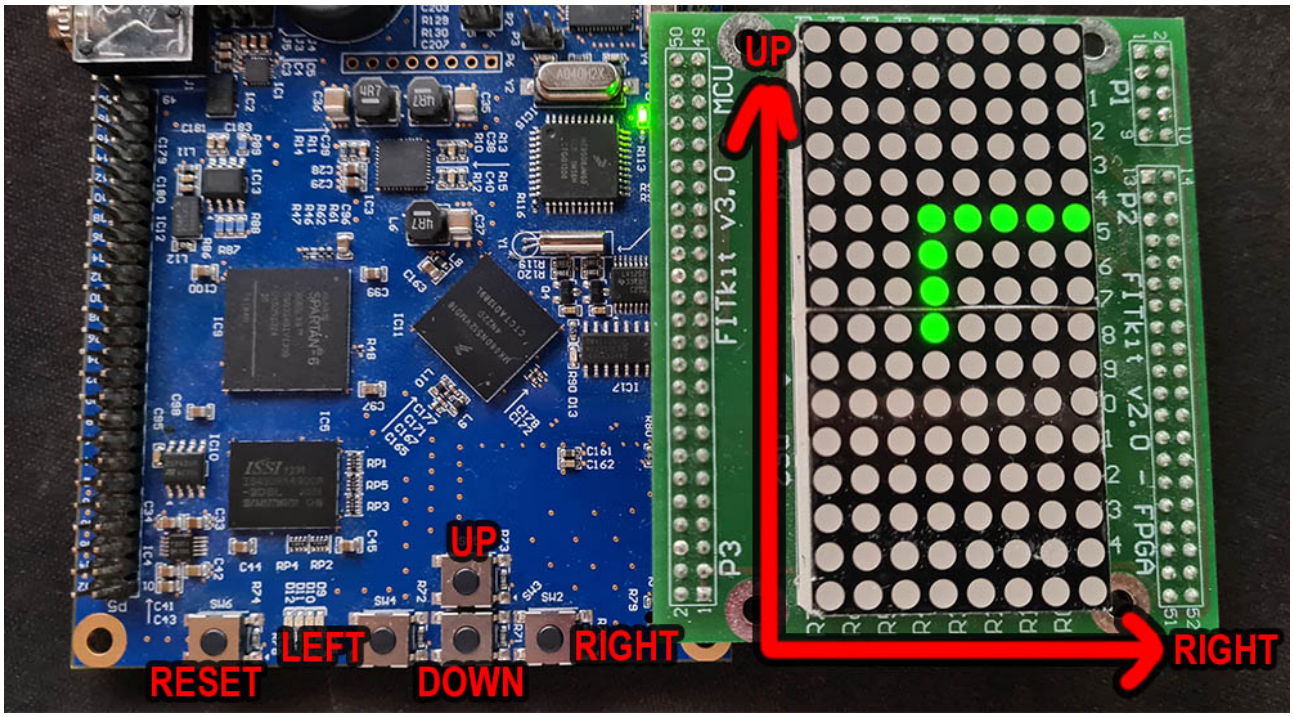
Při spuštění programu je kromě inicializace MCU, portů a časovače inicializováno herní pole, nastavena výchozí poloha hada a příznak `gameOver` nastaven na hodnotu `false`. S výjimkou obsluhy časovače dále celý program běží v nekonečně smyčce, monitorující stisk jednotlivých tlačítek pomocí vstupního registru portu E `GPIOE_PDIR`. Při stisku jednoho ze 4 tlačítek řídící pohyb hada je změněn směr v proměnné `snake_dir`. Při stisku tlačítka pro resetování hry je znovu provedena inicializace herního pole, polohy hada, a příznaku konce hry.

3 Výsledné řešení

V této kapitole je výsledný program popsán spíše z uživatelského hlediska, bez podrobných technických detailů.

3.1 Ovládání hry

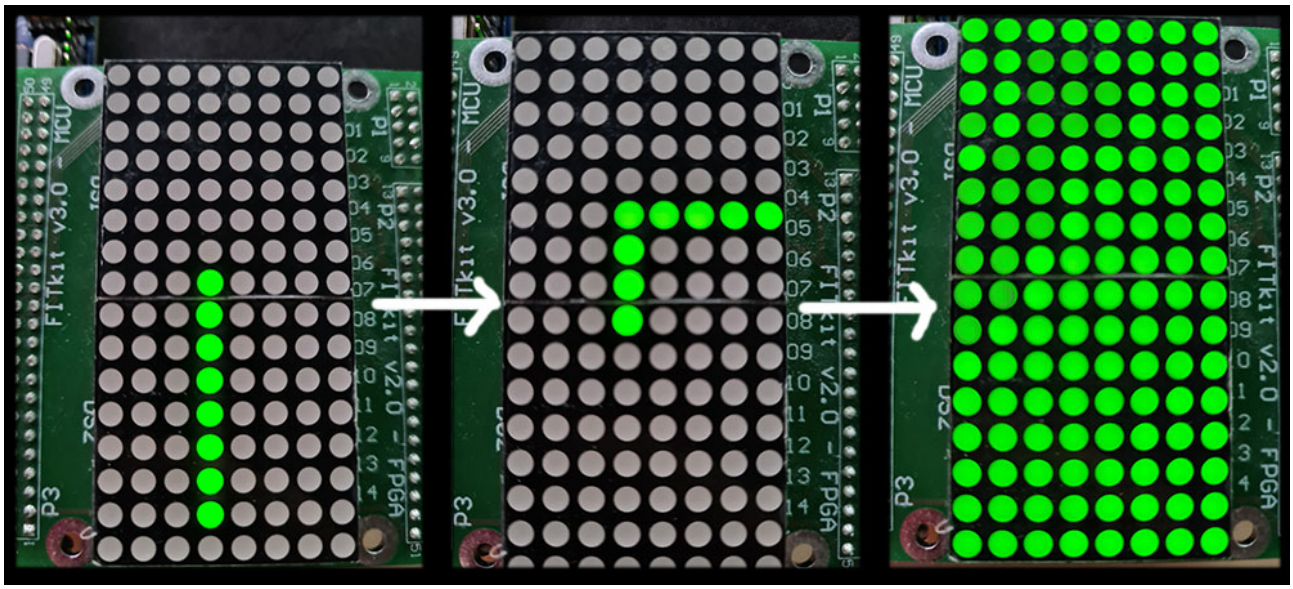
Hra je ovládána celkově pěti tlačítky. Tlačítka `SW2` až `SW5` slouží ke změně směru pohybu hada, zde je důležité podotknout, že je display postaven na výšku, tudíž jsou vlastně sloupce displaye použité jako řádky, a řádky jako sloupce. Obsluha tlačítek je tomu přizpůsobena, tedy had se pohybuje stejným směrem, jako je směr tlačítka při pohledu na samotné zařízení. Tlačítko `SW6` slouží k resetu celé hry, hru lze resetovat kdykoli nehledě na její stav (není třeba čekat až uživatel prohraje).



Obrázek 2: Ovládání hry pomocí tlačítek

3.2 Funkčnost a vlastnosti řešení

Výsledná hra umožňuje uživateli volný pohyb po herní ploše, přičemž jsou detekovány kolize s hranicemi a také se samotným tělem hada, které hru ukončí a informují uživatele o prohře rozsvícením celého maticového displaye. Hráč může kromě změny směru pohybu hru kdykoli resetovat, čímž navrátí hada do výchozí pozice a směr pohybu nastaví dopředu. Výsledná obnovovací frekvence displaye je vhodná pro dobrou viditelnost aktivity na herní ploše, a had provádí přibližně 3 pohyby za sekundu. Kód hry je napsán flexibilním způsobem, změnou hodnot konstant `SNAKE_LENGHT` a `PERIODS_PER_MOVE` je snadné pozměnit velikost hada a frekvenci jeho pohybů, pro zaručení správné funkcionality je ale vhodné se příliš nevzdalovat od výchozích hodnot.



Obrázek 3: Ukázka průběhu hry, včetně prohry kolizí s pravou hranicí herního pole

4 Závěr

Projekt jsem začal řešit už v polovině října a musím přiznat, že jsem byl nejprve poměrně zmatený z vývojového prostředí **KDS** a také ze samotného **FITkitu 3**, který se dosti liší od jeho předchůdců. Po několika experimentech s materiály poskytnutými v rámci předmětu jsem se však začal v problematice lépe orientovat. Hlavní pokrok přišel po nalezení dokumentu obsahující schéma zapojení maticového displaye na FITkit, viz 1.2. S ovládáním maticového displaye pomocí multiplexingu jsem se již dříve setkal v semináři VHDL (předmět **IVH**), takže vytvořit mechanismus obsluhy displaye nebylo příliš problematické. Při řešení jsem se trochu bál, zda není nastavení compare hodnoty časovače Low-power timer na 0 nevhodné, s tímto nastavením se mi však podařilo dosáhnout nejlepších výsledků co se týče obnovovací frekvence displaye. S výsledkem jsem spokojen a samotné programování mě celkem bavilo. Dosti se lišilo od programování FPGA se kterým jsem se setkal v předešlých kurzech, jedná se tedy o kompletně novou zkušenost, což považuji za přínosné.

- Původní (nekomentované) video prezentující funkčnost řešení: https://www.youtube.com/watch?v=RqtJvzvbgGs
- Obhajoba projektu formou videa: https://drive.google.com/file/d/1J7mCmarkUhP2P4I92j_V0exUolZ5oRXE/view?usp=sharing
- Výsledek autoevaluace pomocí hodnotícího klíče: 12,6 b.