Posudek oponenta diplomové práce

Název práce: Migration of an IoT real-time system and reimplementation of some of its functions

Jméno autora: Bc. Jan Mrázek

Oponent práce: Ing. Michal Sojka, Ph.D.
ČVUT CIIRC, oddělení průmyslové informatiky

Předložená diplomová práce se zabývá migrací systému pro aplikace IoT z mikrokontroléru a malého RTOS na výkonnější systém s OS Linux. Cílem je studie proveditelnosti migrace konkrétního produktu – iQtec PLC firmy Prologue s.r.o.

V práci o rozsahu 63 stran autor nejprve pěkně popisuje historii a aktuální situaci trhu IoT a uvádí, proč dříve vhodná hardwarová řešení postavená na mikrokontrolerech dnes přestávají vyhovovat. Poté představuje hardware stávajícího systému a na základě požadavků definovaných na nástupnický systém analyzuje několik možných kandidátů. Největší část práce se věnuje převodu software ze starého systému na novou platformu.

Práce je psána anglicky, dobře se čte a neobsahuje téměř žádné zásadní pravopisné chyby či překlepy. Grafická forma je pěkná (standardní šablona pro TEX/LATEX), jediné, co se dá vytknout je používání rozdělovníků (-) místo pomlček (–) a umisťování odkazů na literaturu za tečku na konci věty (správně se umisťují před tečku). Text je velmi dobře ozdrojovaný a je jasné, které části jsou převzaté a které reprezentují vlastní práci autora. Měl bych menší výhrady k formě 4. kapitoly (implementace), která místy popisuje přesný sled činnosti prováděných autorem, včetně slepých uliček. Text je pak zbytečně dlouhý. V technických textech bývá zvykem popisovat pouze finální stav a „slepé uličky“ reprezentovat požadavky na finální stav, uvedenými před vlastním popisem. Navíc, např. v sekci 4.6.3 by bylo vhodnější uvést přímo kód, než slovy popisovat co kód dělá.

Co se vlastní technické práce týče, tak postup studenta byl vesměs správný. Součástí práce ale bohužel není žádný kód, takže nelze ohodnotit kvalitu implementace. Chápu, že zveřejnění kódu iQtec PLC asi není žádoucí, ale kód prováděných experimentů by součástí práce být mohl. Občas jsou některé závěry ne úplně přesně formulovány a někdy autor dospěje k řešení, které je sice funkční a pro popisovaný produkt pravděpodobně dostačující, ale existují i vhodnější řešení, která v práci nejsou zmíněna. Dovolím si na některé z těchto nedostatků upozornit, jako námět pro další práci na systému:

- Autor správně analyzuje latenci systému nástrojem `cyclictest`, ale nezatěžuje systém síťovou komunikací. Právě síťová komunikace obvykle způsobuje na ne-preempt-rt Linuxu největší latence.

- Test s vlákny a časovačem pravděpodobné nebyl dobře navržen. Z textu se zdá, ze byly použity POSIXové časovače, které indikují vypršení pomocí signálů a doručení signálů není ani s preempt-rt časově deterministické. Bohužel to není moc dobře zdokumentováno. Dá se o tom dozvědět z přednášek na RT-Linux summitech. Pro časování se doporučuje používat funkci `clock_nanosleep` (případně s příznakem `TIMER_ABSTIME`).

- Mechanismus Device Tree Overlay byl původně navržen, jak správě píšete, pro snadné použití přídavných desek (capes) s BBB. Řekl bych, že použití „overlay“ pro změnu drobných vlastností (rychlost SPI) jedné konkrétní desky , jak navrhujete, je zbytečně komplikované. Většinou se pro každou novou desku zkopíruje základní device tree (dts) a ten se přímo upraví na míru konkrétního HW. Výhoda je, že jeden soubor dts se verzuje snáze, něž několik binárních souborů dtb.

- V kapitole 4.6.2 zmiňujete, že vlákna zpracovávající události od časovače mají nejvyšší možnou prioritu (99). To není vhodné ze dvou důvodů:

    1. Vlákno časovače (to, které volá `sem_post`) má tedy prioritu menší nebo stejnou a pokud se běh zpracovávacího vlákna nečekaně prodlouží, vlákno s časovačem kvůli své nižší prioritě nebude moct obsloužit další vypršení časovače a celé časování se vám může „rozjet". Sice by to šlo nějak ošetřit, ale mnohem vhodnější je nastavit vláknu časovače vyšší prioritu než zpracovávacím vláknům. Z vlákna časovače pak můžete jednoduše detekovat, že se nějaké zpracovávací vlákno zpozdilo či zaseklo a můžete na to nějak reagovat.
    2. Jaderná vlákna (ovladače, síťový stack, ...) mají pří použití preempt-rt patche priority okolo 50. Pokud má aplikace vyšší priority, může se stát, že jádro nebude mít dostatek času pro svou práci a začnou se dít věci jako ztrácení síťových paketů apod.

- Systém periodického spouštění jednotlivých úloh umožní jednoduchou migraci stávajícího kódu, ale pravděpodobně zbytečně plýtvá výkonem systému, když jsou úlohy spuštěny a není pro ně žádná práce. To může hrát velkou roli u bateriemi napájených systémů. V takových případech bývá výhodné použít událostmi řízené spouštění úloh (systémová volání jako `poll`/`epoll` či nadstavbové knihovny jako `libev`, `sd-event` či `Boost.Asio`). Ty umožňují úlohy spouštět jen když je skutečně potřeba (např. při příchodu zprávy ze sítě nebo při změně GPIO) a navíc ihned, bez čekání na další periodu časovače.

Otázky na studenta:

1. V práci několikrát uvádíte, že kód iQtec PLC je poměrně rozsáhlý, ale nikde to není přesně specifikováno. Můžete uvést o kolik (tisíc) řádek kódu se zhruba jedná?

2. Na obrázcích 3.1 a 3.2 jsou uvedeny doby odezvy naměřené při experimentech, ale z textu vyplývá, že byly naměřeny i větší odezvy, které na obrázku nejsou vidět. Mohl byste prezentovat odezvy formou histogramu s logaritmickou svislou osou, aby se daly dobře porovnat výsledky s preempt-rt a bez? Domnívám se, že rozdíl by měl být výrazně větší, než jak se zdá z vašich grafů. Možná to souvisí s použitím signálů, jak uvádím výše.

3. Můžete objasnit, proč je v některých případech potřeba volat třikrát `fopen` pro emulaci původního API? Domnívám se, že i pro vámi uvedený důvod (současné čtení i zápis binárního souboru) stačí pouze jedno zavolání.

I přes několik výše uvedených námětů na zlepšení jsem s předloženou závěrečnou prací spokojen a hodnotím ji klasifikačním stupněm **A – výborně**.

Další, méně důležité, poznámky:

- Odkazy na jiné sekce textu jsou uváděny pouze jako číslo a nemusí být jasné, jestli číslo odkazuje na sekci či obrázek. Vhodnější by bylo použít spojení „described in *section* 3.2".

- Píšete, že SCHED_FIFO and SCHED_RR se dají použít jen s preempt-rt patchem. Není to pravda – bez preempt-rt vše funguje úplně stejně, jen maximální latence jsou většinou větší.

V Praze dne 6. června 2019 Ing. Michal Sojka, Ph.D.