Ročníková a maturitní práce

Ovladač počítačových ventilátorů

Autor: David Koňařík
Škola: Gymnázium, Praha 6, Arabská 14
Ročník: 4. E 2019/2020

Anotace

Cílem mé práce bylo vytvořit hardware a software pro automatické ovládání počítačových ventilátorů. Výsledný hardware zvládá regulovat oba běžně používané typy ventilátorů, třípinové i čtyřpinové, a poskytuje uživateli mnoho možností pro nastavení.

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

Obsah

Kapitola 1

Hardware

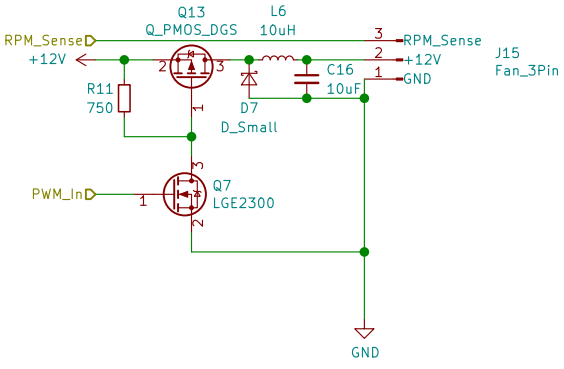
Hardwarová část ročníkové práce má formu zařízení, které je určeno k vložení do počítače a připojení k jeho zdroji, je možné jej ale používat i mimo počítač s externím zdrojem.

Návrh obvodu a tištěného spoje byl proveden v svobodném softwaru KiCad a tištěné spoje vyrobila společnost JiaLiChuang Co., Limited (JLCPCB)[1].

Jako součást ročníkové práce odevzdávám první verzi hardwaru, v1, která je zde dále popsána.

Hardware řídí mikrokontrolér STM32F103C8T6[2] založený na jádře ARM Cortex-M3. Přímo na něj je napojeno pět konektorů pro čtyřpinové ventilátory, které jsou přímo regulovány PWM výstupem mikrokontroléru. Dalších šest třípinových ventilátorů je regulováno obvodem na bázi spínacího zdroje.

Tento obvod lze rozdělit na část spínací a filtrační. Hlavní komponentou spínací části je P-kanálový MOSFET, který je aktivován jednoduchou kombinací N-kanálového MOSFETu a pull-up resistoru. Filtrační část pak sestává z LC low-pass filtru. Využití high-side přepínání umožňuje připojit tachometrický signál přímo k mikrokontroléru, jelikož zem ventilátoru je připojena přímo k



Obrázek 1.1: Regulační obvod třípinových ventilátorů

[1] https://jlcpcb.com/

[2] https://www.st.com/en/microcontrollers-microprocessors/stm32f103c8.html

celkové zemi zařízení.

Součástí zařízení je i USB konektor, který měl sloužit ke komunikaci s počítačem. I přes použití referenčního designu od výrobce mikrokontroléru ale není rozhraní v aktuální verzi USB funkční, je tedy místo něj používán sériový port s dedikovaným převodníkem.

Kapitola 2

Software

Firmware je napsán v jazyce C nad operačním systémem ChibiOS. Dokáže bez připojení k externímu zařízení automaticky regulovat otáčky ventilátorů podle teplotních dat.

Komunikace pro nastavování teplotně-otáčkových křivek a případné aktualizace hodnot virtuálních zdrojů teplotních dat je realizována příkazovou řádkou. Tuto komunikaci lze provádět manuálně s pomocí jakéhokoliv sériového terminálu nebo automatizovat.

2.1 Přehled příkazů

- `fan_list`
  Přehled všech portů na ventilátory a rychlosti k nim aktuálně připojených ventilátorů.
- `fan_set_tempsrc <fan> <tempsrc>`
  Nastaví zdroj teplotních dat, který bude řídit rychlost daného ventilátoru.
- `fan_set_gmap <fan> (<temp> <speed>)+`
  Nastaví křivku závislosti rychlosti ventilátoru na teplotě.
- `fan_get_gmap <fan>`
  Vypíše křivku závislosti rychlosti ventilátoru na teplotě.
- `tempsrc_list`
  Přehled všech zdrojů teplotních dat a jejich hodnot.
- `tempsrc_virt_set <tempsrc> <value>`
  Nastaví hodnotu zdroje teplotních dat.

Kapitola 3

Diskuse

3.1 Hardware

Hardwarová část projektu prošla v rámci vývoje několika iteracemi.

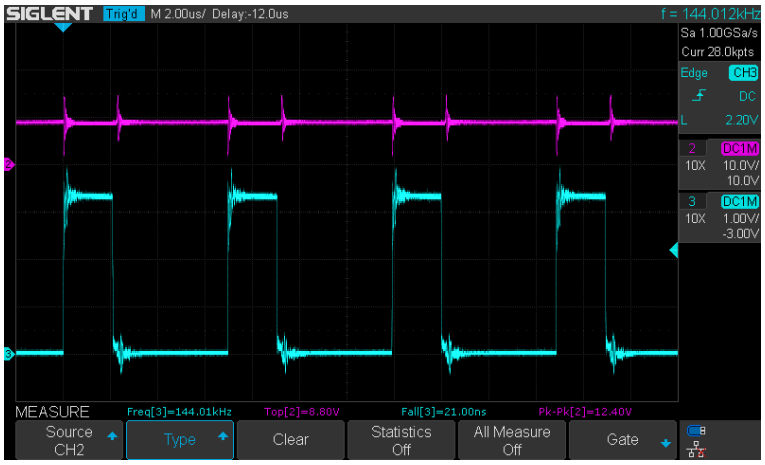
Nejproblematičtější částí byl obvod pro ovládání třípinových ventilátorů. Jeho základní princip je skoro shodný s variabilním zdrojem napětí (buck DC-DC converter), jen bez feedbacku. Kvůli nutnosti regulovat napětí digitálně z mikrokontroléru by nebylo jednoduché použít levné čipy pro dedikované zdroje, jelikož u těch se výstupní napětí definuje rezistivním děličem. Dedikované čipy s digitálním rozhraním jsou vesměs příliš drahé a zároveň i nepotřebně výkonné.

První pokusy s low-side spínáním pomocí N-kanálového MOSFETu byly sice slibné, ale nenašel jsem cestu, jak získat tachometrický signál. Napětí na kladném pinu ventilátoru je v tomto obvodu totiž vždy 12V a mění se jeho záporný pin, který není připojen přímo k zemi. Tachometrický signál ventilátoru je střídavě plovoucí nebo připojený k zemi ventilátoru, která může mít vzhledem k mikrokontroléru jakékoliv napětí mezi 0V a 12V.

Pro v1 tištěného spoje jsem zvolil low-side spínání P-kanálovým MOSFETem. Ventilátor je tak připojen přímo k zemi celého obvodu a tachometrický signál lze připojit přímo k mikrokontroléru. Tato varianta má ale výrazně horší spínací vlastnosti: nezanedbatelná kapacitance gate pinu P-MOSFETu a odpor pull-up rezistoru omezují minimální délku PWM pulzu na 12V straně obvodu, což výrazně zvyšuje minimální otáčky připojeného ventilátoru. Tento problém lze částečně vyřešit použitím pull-up rezistoru o menší hodnotě, což ale způsobí výrazně větší tepelné ztráty na rezistoru.

Pro připravovanou verzi v2 jsem zvolil opět low-side spínání a přidal jsem komparátor na tachometrický signál, který je jím porovnáván vůči referenčnímu 11V napětí. Narazil jsem na fenomén nazývaný "gate ringing", vysokofrekvenční oscilace na gate pinu MOSFETu způsobené rychlým přepínáním. Tyto oscilace by mohly poškodit ventilátor nebo i počítač, ve kterém je zařízení umístěno. Jejich vzniku jsem dostatečně zamezil přidáním rezistoru mezi gate a mikrokontrolér.

Jednou z původně plánovaných vlastností projektu byla možnost funkce bez připojení přes USB. V tomto případě by získával software informace o teplotě chlazených komponent připojením do ventilátorového portu základní desky. V první verzi hardwaru však kvůli limitaci ze strany mikrokontroléru nelze zároveň



Obrázek 3.1: "Gate ringing" (modře gate MOSFETu, červeně napětí na zdroji)

měřit tento vstupní signál a otáčky ventilátoru číslo 9, jelikož oba vstupní piny sdílí linku externího interruptu.

I přes značnou snahu se mi nepodařilo zprovoznit komunikaci přes USB. Původní obvod neobsahoval žádné sériové rezistory na datových linkách, jelikož je nezmiňují oficiální dokumenty od ST[1]. V procesu hledání chyby jsem se pokusil přímo použít obvod ze své vývojové desky, kde USB funguje. Jediný rozdíl byla hodnota pull-up rezistoru pro D+ a 22Ω sériové rezistory na D+ a D-, i po jejich přidání ale zařízení nenaváže úspěšně komunikaci přes USB.

3.2 Software

Software jsem se nejprve rozhodl napsat v programovacím jazyce Rust[2], který se svými vlastnostmi hodí pro programování mikrokontrolérů. Jeho zaměření na compile-time záruky je žádoucí vzhledem k dlouhému procesu nahrávání firmwaru do flash paměti. Bohužel knihovny pro interakci s hardwarem mikrokontrolérů pro Rust[3] jsou aktuálně ve fázi překotného vývoje a jejich použití na složitější programy může jednoduše narazit na nevypilované části designu.

Krom Rustu jsem zkoušel ještě několik jazyků a knihoven:

Platforma Arduino[4] [5] má sice velmi jednoduché rozhraní a není ani náročná na paměť, ale její abstrakce překrývají některé detaily hardwaru, které jsou pro jakékoliv složitější projekty důležité (například časování PWM výstupů nebo limity externích interruptů STM32).

Platforma Mbed[6] by se dala popsat jako evoluce Arduina. Využívá C++ a poskytuje také velmi jednoduché rozhraní, dbá přitom více na zásady organizace kódu a programátor si tak může vybrat úroveň abstrakce vhodnou pro jeho

[1] https://www.st.com/resource/en/application_note/dm00296349-usb-hardware-and-pcb-guidelines-using-stm32-mcus-stmicroelectronics.pdf

[2] https://www.rust-lang.org/

[3] https://github.com/stm32-rs/stm32f1xx-hal

[4] https://www.arduino.cc/

[5] https://github.com/stm32duino/Arduino_Core_STM32

[6] https://os.mbed.com/

projekt. Tato flexibilita ale zvětšuje výsledné soubory natolik, že program se všemi funkcemi by se nemusel vejít do paměti mikrokontroléru. Zároveň jsem narazil na různé problémy s build systémem, ať už při používání oficiálního `mbed-cli` nebo populárního PlatformIO.

Platforma ChibiOS[7] už není tak jednoduchá jako Arduino nebo Mbed, ale zato poskytuje přístup ke všem potřebným detailům hardwaru a má i obsáhlé kontroly konfigurace, které zjednodušují použití jinak na první pohled složitého systému. Je i velmi nenáročná na paměť.

Projekt ChibiOS je rozdělen na několik částí, z nichž jsem použil ChibiOS/RT, takzvaný "real-time operační systém", a ChibiOS/HAL, abstrakční vrstvu nad přímou interakcí s hardwarem.

[7] `http://chibios.org/dokuwiki/doku.php`

Kapitola 4

Závěr

Tento projekt začal jako snaha vyřešit problémy s chlazením vlastního počítače. V rámci práce se mi podařilo vytvořit funkční produkt, který plní svůj účel. Jako každá první verze má i on své nedostatky. Ty však nejsou nepřekonatelné a doufám, že s dalším úsilím a díky nově nabytým znalostem a zkušenostem dokáži i je vyřešit a použít výsledný hardware ve svém počítači.