Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Software pro ovládání dveří dopravních prostředků s elektromechanickým otevíráním dveří

2020

Marek Schiller

Vedoucí práce: RNDr. Arnošt Večerka

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Marek Schiller
Název práce:	Software pro ovládání dveří dopravních prostředků s elektromechanickým otevíráním dveří
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2020
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	RNDr. Arnošt Večerka
Počet stran:	25
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Marek Schiller
Title:	The software designer for door opening purposes of vehicles with electromechanical door opening
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2020
Study field:	Applied Computer Science, full-time form
Supervisor:	RNDr. Arnošt Večerka
Page count:	25
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Tato bakalářská práce se zabývá návrhem firmwaru pro řídící mechanickou jednotku otevírání a zavírání dveří dopravních prostředků. Při návrhu jsem musel dodržet veškeré bezpečnostní prvky a požadavky dány normou a firmou ASN Plus.

Synopsis

This bachelor thesis deals with the firmware model for mechanical control unit opening and closing vehicle doors. For this model I had to heed with all safety regulations and requiements given by the standard ASN Plus.

Klíčová slova: PWM; PlatformIO; mikroprocesor; firmware

Keywords: PWM; PlatformIO; microcontroller; firmware

Především bych chtěl poděkovat panu RNDr. Arnoštu Večerkovi. Dále bych chtěl poděkovat firmě ASN Plus s.r.o za možnost vyvíjet tuto jednotku. A nakonec chci poděkovat rodičům a všem, kteří mě po celou dobu studia podporovali.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam vět

Seznam zdrojových kódů

1 Úvod

Vývoj firmware je jedním z nejnáročnějších druhů vývoje software. Rád bych zde vysvětlil pojmy firmware a software, protože v laické (někdy i odborné) společnosti bývají špatně chápány. Software obecně zahrnuje všechny druhy softwaru: firmware, ovladače, middlewere a software (aplikace). Hardware je již společností chápán správně. Obecně to je to, na co si můžeme šáhnout rukou, například CPU, MCU, myš, monitor atd. Jsou to fyzické komponenty. Firmware je software, který je nejblíže hardwaru. Je psaný přesně pro daný hardware. Firmware nejčastěji řeší práci s pamětí a periferiemi. Počítačová myš v sobě má mikrokontrolér (MCU), který má firmware. Všeobecně platí, že MCU je v embedded (jednočipovém) zařízení.

Naskytla se mi možnost vypracovat návrh firmware, který bude používán v reálném životě. Tato práce mě velmi nadchla, ale zároveň i trochu znepokojila. Nikdy předtím jsem nic takového ještě neprogramoval a navíc se tyto programy velmi špatně testují. Vývojové prostředí, dále také jako IDE, mi pohlídá jenom základní syntaxi a překlepy. Neexistuje žádná možnost debug režimu. Napsaný program se nahraje do mikrokontroleru a už vykonává svojí činnost.

2 Specifikace požadavků

Požadavků, které taková jednotka musí splňovat je spousta. Ať už jsou dány normou nebo konkurencí mezi firmami, které takové zboží nabízí. Zde bych rád uvedl nejzákladnější požadavky na jednotku.

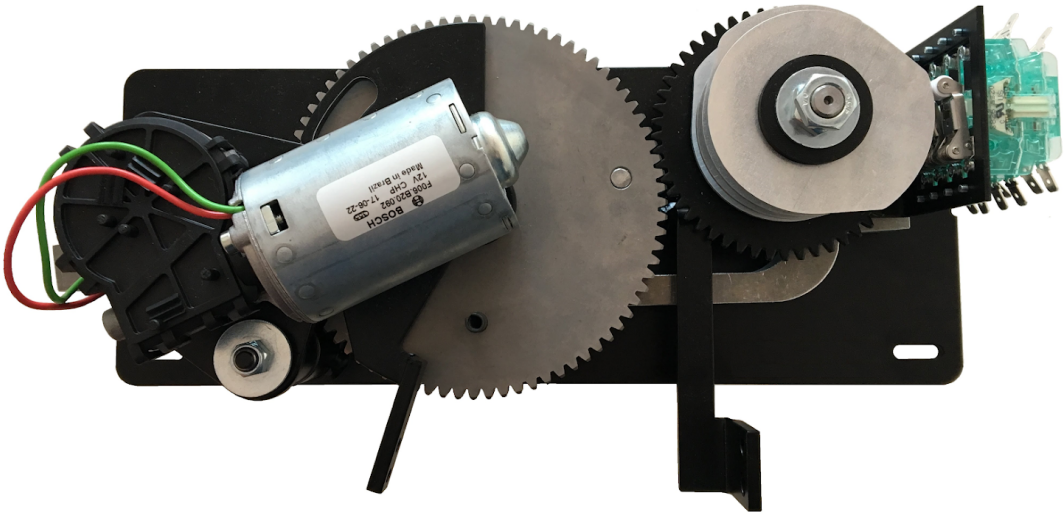
- Při zapojení jednotky zkontrolovat polohu dveří. Pokud je poloha známá ⇒ žádná reakce. Pokud jsou dveře v nedefinované poloze ⇒ musí se zavřít.
- Nulová rychlost znamená, že dopravní prostředek stojí na místě ⇒ není v pohybu. Nulová rychlost má nejvyšší prioritu. Pokud jsou dveře otevřené a dopravní prostředek se rozjede ⇒ nouzové zavírání dveří.
- Při každém otáčení motoru pohonu měřit čas otáčení. Pokud měřený čas přesáhne nastavenou hodnotu ⇒ zastavit motor a zahlásit chybu, která rozsvítí červenou led diodu na jednotce.
- Zkratová ochrana je neustále kontrolována. Když se objeví ⇒ zastavit motor a zahlásit chybu, která rozsvítí červenou led diodu na jednotce.
- Poptávka nastane, pokud pasažér zmáčkne tlačítko s požadavkem na otevření dveří ⇒ poptávka se uloží a čeká na uvolnění, které provede řidič tlačítkem na řídící desce.
- Plynulá změna otáček motoru, které zamezují proudovým špičkám a výkyvům.
- Při každém pohybu dveří se měří proud motoru, pokud je větší než nastavená hodnota (reakce se liší podle předchozího povelu) můžou nastat tři akce:
  - pokračování v otevírání
  - zavírání
  - reverzování tzn. opakování

  K tomu zde máme nastavitelnou hodnotu - kolikrát se má pokus opakovat.
  - Povel otevřít při naměření kritického proudu ⇒ začne běžet čekací doba 1s (nastavitelná parametrem). Až doba vyprší pokračuje se v otevírání. Tím se zmenší počet pokusů otevření.
  - Povel zavřít při naměření kritického proudu ⇒ začne běžet čekací doba 1s (nastavitelná parametrem), po vypršení času ⇒ se nastaví akce - reverzování (dveře se začnou otevírat).
- Světelná a zvuková výstraha se ovládá přes jeden společný pin. To je výhodné, nemusím ovládat každou výstrahu zvlášť. Pokud by byl požadavek vypnout například zvukovou výstrahu, musela by se odpojit mechanicky. Výstraha je zapnutá při zavíráni, reverzování a při nouzovém zavírání.

3 Základní pojmy

3.1 Pohon

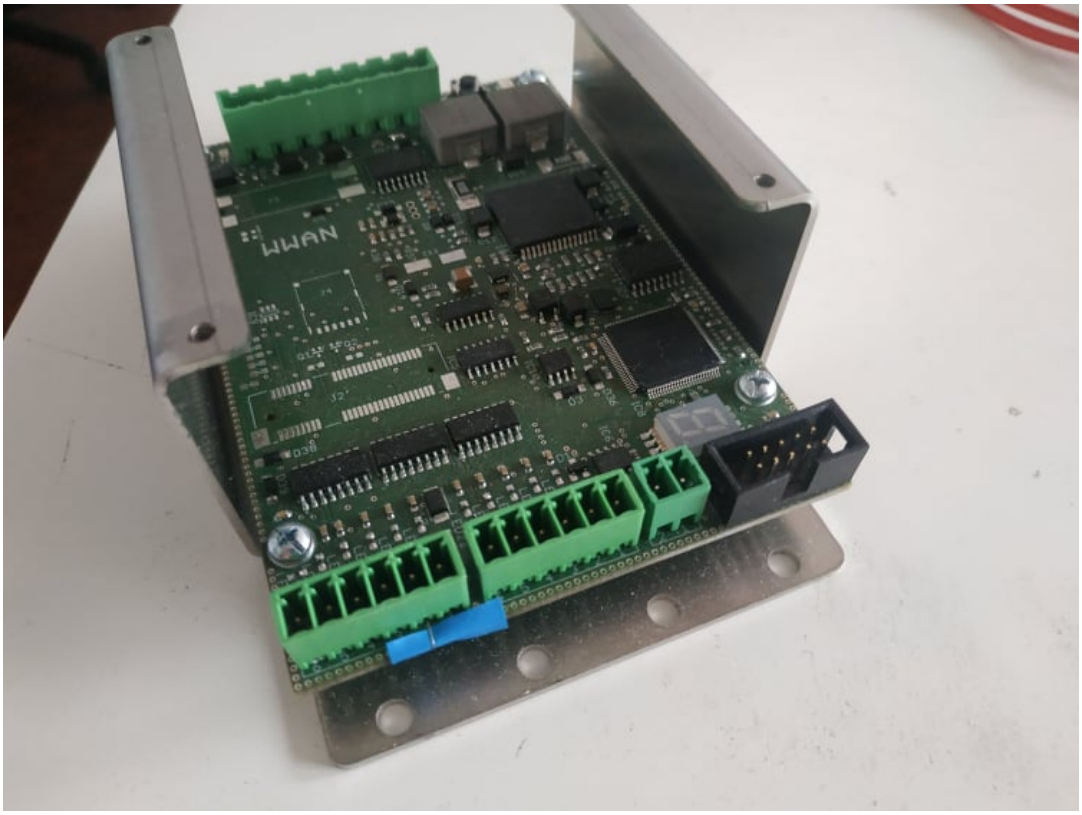
Pohon chápejme jako sestavu mechanických součástek a elektronických čidel a snímačů. Takový pohon lze vidět na obrázku 1. Tento pohon je dvoukřídlý, to znamená, že ovládá obě křídla dveří současně. Tento systém vídáme velmi často u zadních autobusových dveří. U předních autobusových dvěří mohou být také dvě křídla ale u menších autobusů a mikrobusů jsou jednokřídlé dveře. To znamená výměnu pohonu ale jednotka zůstává stejná.



Obrázek 1: Ukázka mechanického pohonu pro mikrobus

3.2 Embedded systém

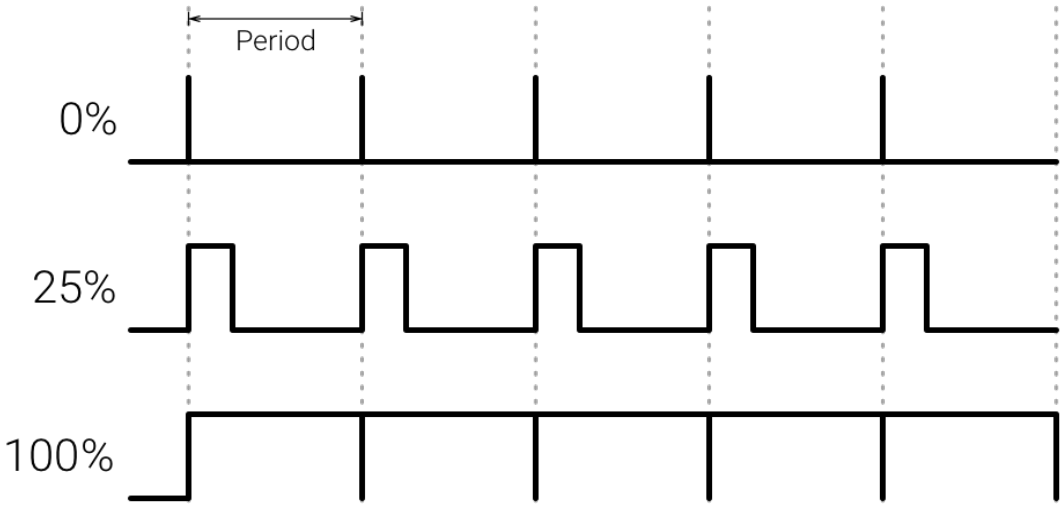
Embedded systém je jednoúčelový počítač. Dále budu označovat za jednotku nebo DCU (door control unit). DCU je zobrazeno na obrázku 2. Základem je integrovaný obvod na kterém jsou elektronické součástky. Hlavní součástkou DCU je mikrokontrolér (MCU), který řídí celou jednotku. Každý MCU má několik pinů na kterých je čtena hodnota, buď logická jednička nebo logická nula. A na tyto hodnoty reaguje program, který je nahraný v MCU. V každém takovém programu musí být realizována nějakým způsobem nekonečná smyčka. Program, který kontroluje vstupy a nastavuje výstupy podle toho co je v instrukcích programu.



Obrázek 2: Ukázka DCU

3.3 Pulzně široká modulace PWM

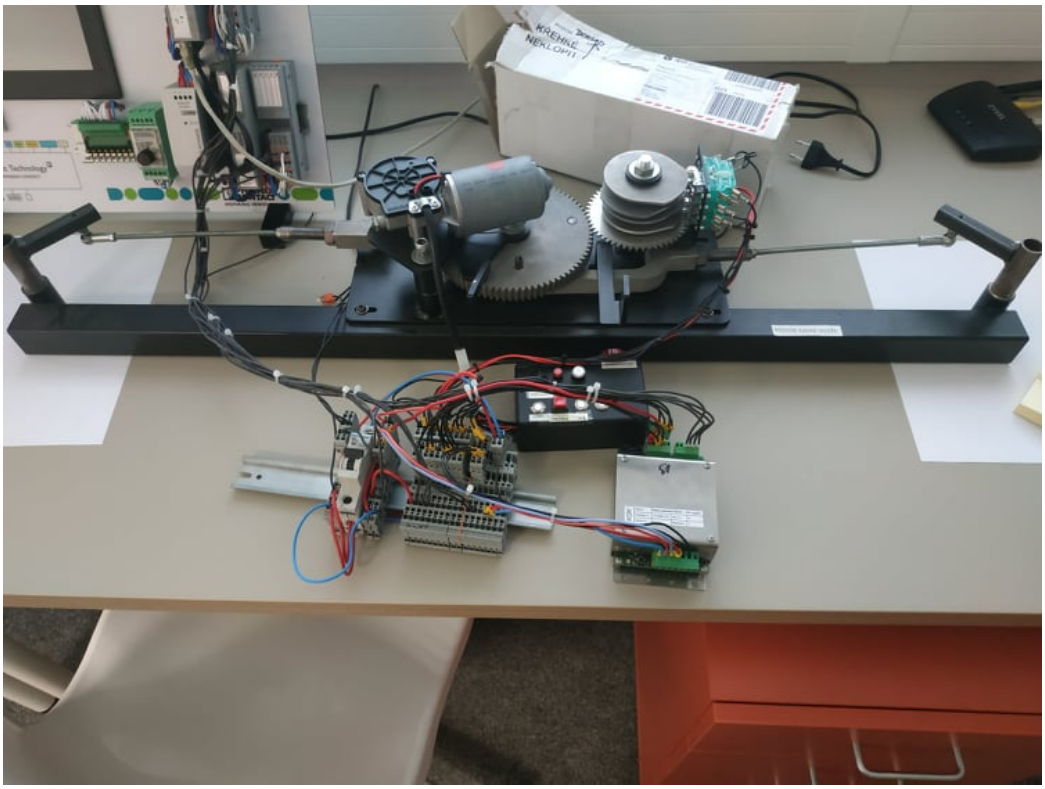
Pulzně široká modulace je dost složitý pojem a ještě více složitější je s ním správně pracovat. Nejlepší to bude popsat na obrázku 3. Na obrázku jsou tři průběhy PWM. Horní a spodní křivky jsou poměrně jednoduché. Horní ukazuje, že signál je stále v logické nule po celou dobu. Spodní ukazuje, že je od zapnutí v logické jedničce. Na těchto dvou křivkách není zatím nic zajímavého. Podíváme se proto podrobněji na prostřední křivku. Všimněme si periody jak se křivka opakuje. Je naznačeno, že po dobu jedné periody je signál 25% času v logické jedničce a zbylý čas v logické nule [2]. Pro nastavení periody je důležitá frekvence (též kmitočet) MCU. Například frekvence MCU je 1kHz, což je 1000 tiků za sekundu. MCU mají takzvaný prescaler, který umožnuje změnit frekvenci na určitém timeru, na kterém můžeme mít PWM. Pro přesnější vysvětlení [1] a [2].



Obrázek 3: PWM s životním cyklem

4 Testování

Testování probíhalo na tzv. testovací stolici, která je na obrázku 4. Testovat funkčnost celého pohonu přímo na dopravním prostředku je nereálné, protože bych mohl něco poškodit a dopravní společnosti nemají vozidlo navíc, které by mi dali volně k dispozici na testovaní. Detailněji k testování v kapitole 6.7.



Obrázek 4: Testovací stolice

5 Použité technologie

V této kapitole bych rád příblížil a popsal jednotlivé technologie, programy a součástky, se kterými jsem se setkal při vývoji FW.

5.1 FreeRTOS

Real time operation system (FreeRTOS) [5] je vyvíjen ve spolupráci s předními společnostmi pro mikrokontrolery. V posledních letech nabírá FreeRTOS na velké oblibě. FreeRTOS nabízí kooperativní i preemptivní multitasking. Já jsem použil kooperativní, protože to byl požadavek firmy.

5.2 PlatformIO

PlatformIO [3] je rozšíření do VS Code. PlatformIO umožňuje vyvíjet FW na mnoho různých MCU. Při vytváření nového FW se musíme podívat zda PlatformIO má daný MCU v databázi a danou platformu.

Dále umí PlatformIO základní funkce jako jsou build a upload FW do zařízení. Navíc obsahuje platformu STM32, která v sobě obsahuje LL i HAL knihovny [9]. Čehož využívá firma ASN Plus, která na LL knihovně vybudovala vlastní knihovnu, kterou nazývá LOOL. Firma ASN Plus mi umožnila její použití k vývoji FW.

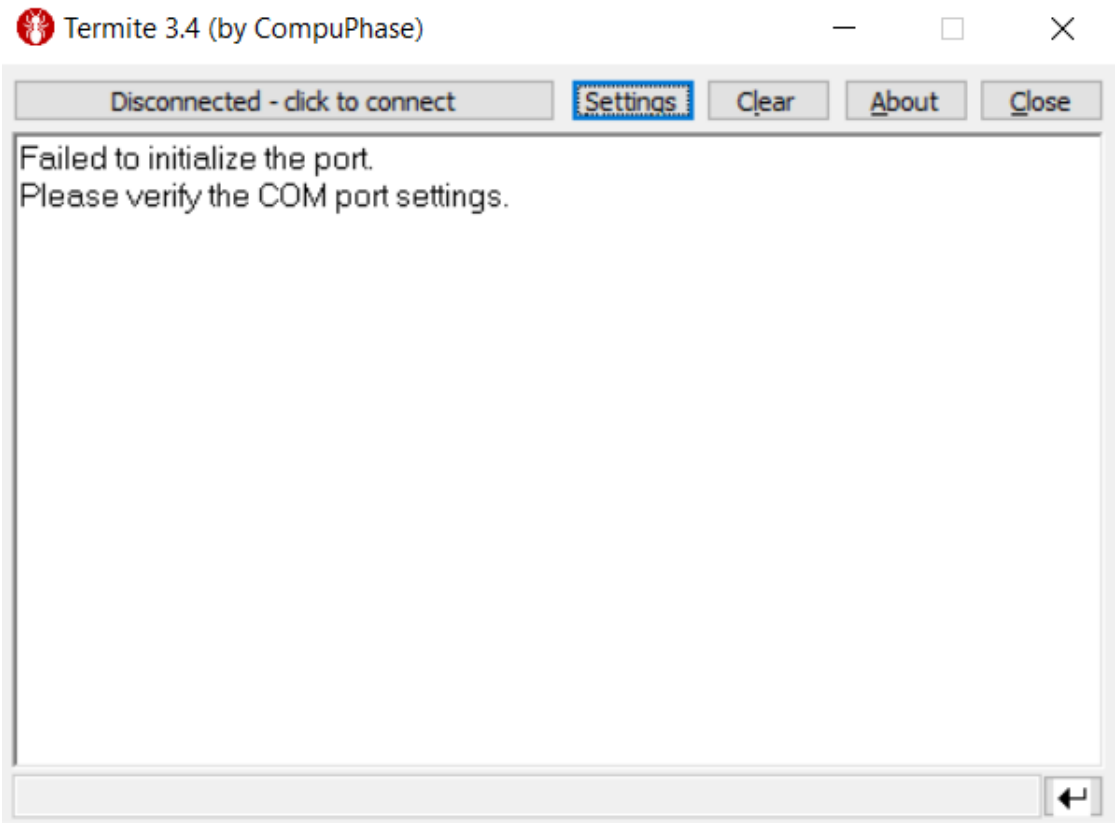
Kombinace VS Code a PlatformIO je vhodná pro firmy, které vyvíjejí FW na více MCU. Nemusejí tak mít pro různé rodiny MCU jiné IDE. Tímto se často vývoj může zrychlit protože vývojář se naučí například klávesové zkratky, které značně usnadňují práci. PlatformIO nabízí například podporu u Atmel, CHIPS Alliance, Microchip a ST STM32.

5.3 STM32CubeProgrammer

PlatformIO má své omezení. Například nemůže mazat a číst paměť z připojené jednotky. Pro tyto účely je vhodný program STM32CubeProgrammer [10]. Pomocí kterého propojíme jednotku s PC, což nám umožňuje nastavit parametry připojení k MCU a aktualizovat samotný FW programátoru [11].

5.4 Termite

Termite je velmi jednoduchý konfigurovatelný terminál, který jsem používal na výpisy USART. Rozhraní je vidět na obrázku 5. K propojení bylo zapotřebí USB TTL převodník [13].



Obrázek 5: Rozhraní programu Termite

6 Implementace

V této kapitole se budu věnovat jednotlivým problémům a algoritmům řešících tyto problémy. V popisech tříd nebudou rozebírány jednotlivé atributy nebo metody.

Jako vývojové prostředí (IDE) jsem použil Visual Studio Code s rozšířením na jazyk C++, ve kterém je napsána většina FW. Jako další rozšíření jsem použil PlatformIO, kterému se věnuji v kapitole 5.2. V jednotce je použit MCU od firmy STMicroelectronics s označením STM32F091VC [6]. U vývoje FW na konkrétní MCU jsem musel neustále nahlížet do manuálu [7] a datasheetu [8] od MCU.

Program začíná funkcí main v souboru main.cpp, kde z třídy System zavolám metodu run. Třídě System se věnuji více v kapitole 6.1.

6.1 Třída System

Ve třídě systém je definice tasků pro FreeRTOS. Zde mám dva tasky, jeden pro testy a hlavní task (applicationTask1). Zde se zdá používat FreeRTOS zbytečné, ale bylo to v požadavcích firmy. Použití tohoto systému bude využito pro další vývoj FW.

```
1       static void applicationTask1( void * parameter )
2       {
3          ((Application *) parameter) -> doorTask();
4       }
5
6       static void testTask1( void * parameter )
7       {
8          ((Test *) parameter) -> run();
9       }
```

Zdrojový kód 1: Definice tasků FreeRTOS

6.2 Třída Display

Třída Display mi reprezentuje elektronickou součástku - segmentový display [14]. Každý segment displeje reprezentuji jedním bitem. Pro zobrazení čísel jsem si vytvořil enum. Samozřejmě by šlo zobrazovat nějaké speciální symboly. Čísla jsou zatím dostačující.

Pro zapnutí jednotlivých segmentů slouží metoda showSymbol.

Tohoto principu jsem používal velmi často, proto jsem to vysvětloval na nejjednodušším příkladě použití.

Display neměl při vývoji FW jednoznačně danou funkci. Proto jsem zvolil zobrazování na displeji číslo akce kterou právě provádí motor. Toto řešení se mi velmi osvědčilo při ladění FW.

```
1     enum symbol_t
2     {
3        symbol_none = 0,
4        symbol_0 = 0b0111111,
5        symbol_1 = 0b110,
6        symbol_2 = 0b1011011,
7        symbol_3 = 0b1001111,
8        symbol_4 = 0b1100110,
9        symbol_5 = 0b1101101,
10       symbol_6 = 0b11111101,
11       symbol_7 = 0b111,
12       symbol_8 = 0b1111111,
13       symbol_9 = 0b11101111,
14    };
```

Zdrojový kód 2: enum symbol_t třídy Display

```
1   void showSymbol( symbol_t symbol )
2       {
3          _segmentAPin.write( ! (symbol & 0b00000001) );
4          _segmentBPin.write( ! (symbol & 0b00000010) );
5          _segmentCPin.write( ! (symbol & 0b00000100) );
6          _segmentDPin.write( ! (symbol & 0b00001000) );
7          _segmentEPin.write( ! (symbol & 0b00010000) );
8          _segmentFPin.write( ! (symbol & 0b00100000) );
9          _segmentGPin.write( ! (symbol & 0b01000000) );
10         _segmentDPPin.write( ! (symbol & 0b10000000) );
11      }
```

Zdrojový kód 3: metoda showNumber třídy Display

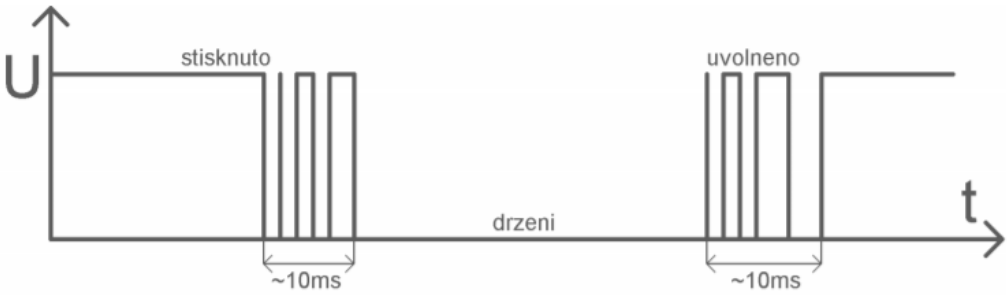
6.3 Třída Board

Ve třídě Board jsou definované jednotlivé piny, vstupy, výstupy a inicializace sběrnic, timerů.

6.4 Třída DigitalSwitch

Třida DigitalSwitch reprezentuje tlačítka. V této třídě řeším problém zvaný zákmit. Jde o to, že po stisknutí tlačítka dojde ke změnám signálu, lépe řečeno zákmitu signálu nahoru a dolů. Dobře je to vidět na obrázku 6 [15].

Rozhodnul jsem se pro inkrementování hodnoty. Toto řešení jsem mohl udělat za určitých podmínek. To znamená, že po dobu 10ms se signál ustálil a neměnil svojí hodnotu. Zde je důležité si uvědomit frekvenci MCU, ta je 8MHz. Tuto hodnotu musím mít nastavevou i v souboru FreeRTOSConfig.h a zároveň v tomto souboru je nastavené časové rozlišení se kterým pracuje FreeRTOS a to je 1kHz. To znamená, že jednou za jednu milisekundu nastane přerušení, FreeRTOS inkre-



Obrázek 6: Graf se zákmitem

mentuje si aktuální čas a rozhodne co udělá dál. Protože mám jenom jeden task, který běží nepřetržitě, mám nastaveno, že tento jeden task může stihnout vše co potřebuje rychleji než za jednu milisekundu. Horší je, že nevím jak dlouho mu to trvá a proto jsem tam přidal, že nebude nic dělat, dokut se nezmění systémový čas.

```
1 while( systemTime == updateTime );
```

Zdrojový kód 4: čekání na začátek nové milisekundy

Nastavil jsem, že chci signál stabilní po dobu deseti milisekund.

```
1           if( valueNow )
2           {
3              if( _duration < 10 )
4              {
5                 _duration ++;
6              }
7              else
8              {
9                 _result = valueNow == _activeLevel;
10             }
11          }
```

Zdrojový kód 5: inkrementování kladného signálu

Hlavní důvod byl ten, že jsem nemusel ve třídě řešit vlastní timer, což by vedlo k zpomalení programu. Uvědomuji si, že toto řešení není nejlepší a s přidáním dalších tasků bude nutno předělat toto řešení.

6.5 Třída MotorDriver a Motor

V těchto třídách řeším problematiku ovládání motoru. Třídá MotorDriver posílá akce přímo motoru. Třídá Motor umožnuje nastavit akce z následujícího

seznamu:

- neutrál
- otevírání
- zavíraní
- brzda

V této třídě jsem musel vyřešit plynulé zvyšování a snižování otáček motoru a to proto, že kdybych motoru dal plný výkon, mohla by vznikat napěťová špička. Což rozhodně nechci. Jednak by mohl FW vyhodnotit napěťovou špičku chybně jako sevření pasažéra nebo závadu na zavíracím systému dveří. A navíc toto plynulé řízení otáček motoru šetří celé zařízení, čímž se zvýší životnost celého pohonu.

K řízení výkonu motoru používám PWM. Protože k motoru má jít frekvence 1,6MHz a jednotka má frekvenci 8Mhz, musím použít prescaler. Všeobecný vzorec na výpočet frekvence:

$$požadovánáfrekvence = \frac{aktuálnífrekvence}{prescaler}$$

Po úpravě vzorečku dostáváme:

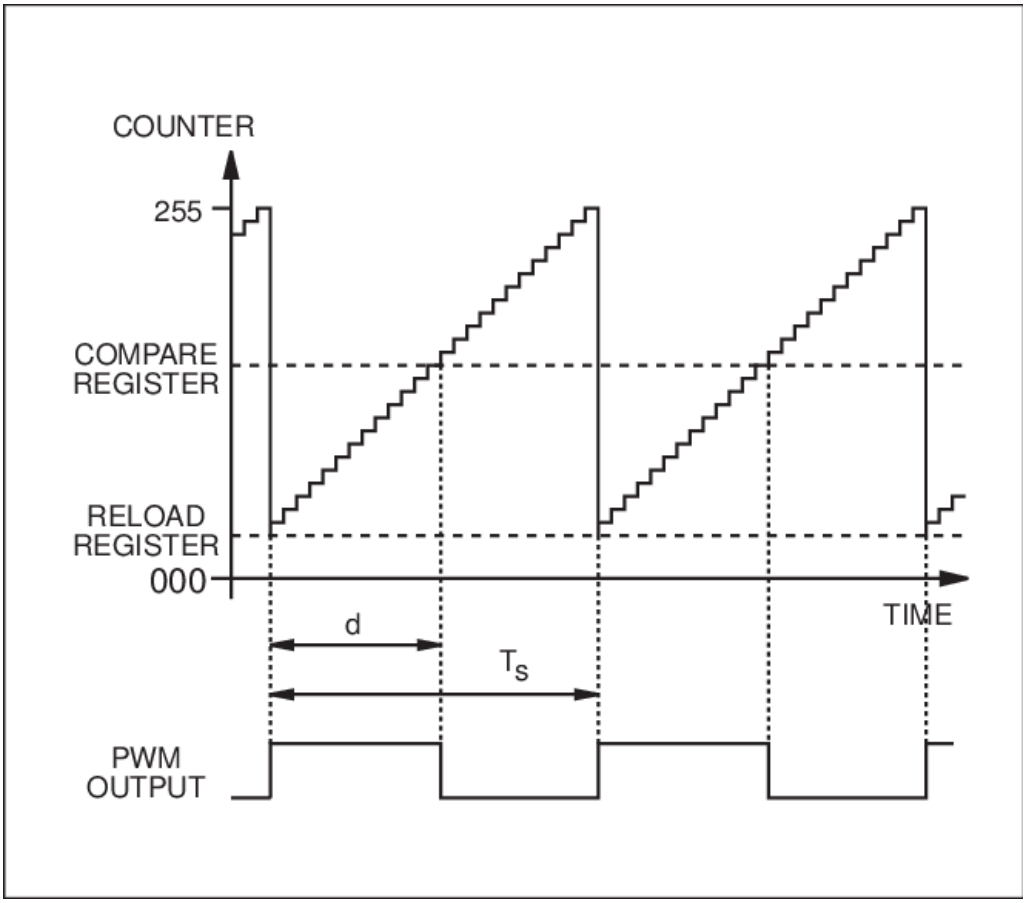
$$prescaler = \frac{aktuálnífrekvence}{požadovánáfrekvence}$$

Po výpočtu mi vyšla hodnota prescaleru 5. Zde se musí dát velký pozor. Na prescaleru ale musím nastavit hodnotu 4, protože se v manuálu od MCU uvádí, že si MCU k nastavenému prescaleru vždy samo přičítá hodnotu 1.

Další parametr, který musím nastavit je autoreload, který nastavím na 99. Autoreload vysvětlím na obrázku 7 [17]. Na obrázku vidíme graf s průběhem PWM (ve spodní části obrázku) a v horní části obrázku vyznačen průběh autoreload. Autoreload mi určí periodu opakování, na grafu označeno T_s, d označuje délku aktivního signálu, respektivě počet tiků aktivního signálu. Dalším důležitým parametrem je PWM mód. Jsou dva typy módu PWM:

- mód 1 $\Rightarrow$ signál je aktivní dokud platí $počítání < nastavenáHodnota$
- mód 2 $\Rightarrow$ signál je neaktivní dokud platí $počítání < nastavenáHodnota$

Zvolil jsem mód 1.



Obrázek 7: PWM s autoreload

6.6 Třída Application

Ve třídě Application v metodě doorTask je celá logika otevírání a zavírání dveří. Kde reaguji na akce podle kterých se orientuji co dveře dělají, případně proč čekají. Akce jsou: žádná, otevírání, zavíraní, reverzování, výstražné zpoždění, otevírací zpoždění, zavírací zpoždění a reverzovací zpoždění.

6.7 Třída Test

Třída Test, která je využívána taskem test. Tato třída byla přepisována v průběhu vývoje neustále. Na začátku vývoje jsem zkoušel jenom nejjednodušší komunikaci s hardware a to: vypínání, zapínání led diody, na display komunikaci ze všemi segmenty displeje, výpisy na UART, zachycení změny a stisknutí tlačítka.

Na první pohled se tento postup může zdát zbytečný nebo až příliš detailní, ale není tomu tak. Uvedl bych hlavní dva důvody. Jednak při mé definici pinů jsem mohl udělat chybu při přepisování čísla pinu nebo portu (této chyby jsem se párkrát dopustil). Další důvod byl výroba této jednotky. Firma ASN Plus si tuto jednotku celou navrhla a vyrobila a při této výrobě mohlo dojít k chybě návrhu

či někde mohla být chyba plošného spoje. K odhalení takovéhoto problému bylo vhodné použít osciloskop [16]. Osciloskop mi ušetřil spoustu práce a bylo velmi zajímavé vidět s takovou přesností naměřené hodnoty. Přesto to někdy nebylo vůbec jednoduché měřit. Při měření jsem dokonce spálil jeden MCU.

Po tomto postupu se třída dostala do podoby před smyčkou, vypíši na UART „test start“, dám povel otevřít a na displeji zobrazím číslo. Potom jsem si pro každou jednotlivou část jako jsou input a output, motor, ADC, shorting a UART udělal metodu, tyto metody potom volám ve smyčce.

V souboru platformio.ini jsou definované prostředí Main a Test, takže stačí přepsat jenom výchozí prostředí a udělat build bez jakéhokoliv dalšího zásahu.

Závěr

Podařilo se mi vytvořit funkční návrh firmwaru, který má ještě nedostatky. Chybí mu některé funkce aby se mohl bezpečně a komfortně uvést na trh. Na tyto funkce mi při vytváření nezbylo mnoho času, protože už vyvinout ten základní návrh firmware bylo velmi časově náročné.

Další funkcionalitu, která by byla potřeba implementovat je čtení a ukládání parametrů do paměti. V mém řešení používám konstanty a samozřejmě tyto konstanty jsou při testování a nasazení do provozu jiné. A pokud jsem je chtěl změnit musel jsem to přepsat v programu a ten znovu nahrát do jednotky.

Při této práci jsem si měl možnost vyzkoušet mnoho nových a zajímavých věcí. Pro pochopení problematiky vývoje tohoto firmwaru jsem musel prostudovat mnoho literatury a doufám, že se mi to bude hodit při dalším studiu na magisterském oboru. Spousta věcí byla pro mě úplně nových a jejich pochopení mi zabralo spoustu času. K jejich porozumění mi velice pomohlo, že jsem si vše mohl vyzkoušet v praxi na zkušební stolici.

Conclusions

I created funkcional firmware model which has unfortunately a few defects. This firmware miss some functions to be great for sale. I didn´t have much time to creat more functions because just the main firmware model was very hard to make.

Reading and saving parameters to memory would be another good function. I applied constats. These constats are obviously in testing and using in real life different. If I wanted to change them I had to rewrite them in the program and reload them to the control unit again.

I had chance to try a lot of new and very interesting things in this case. I had to read many books about firmware development and I hope that this knowledge will be useful during my further study. A lot of things were completely new to me. Their understanding took me a lot of time but it was great to try everything in practice.

A Obsah přiloženého CD/DVD

Na samotném konci textu práce je uveden stručný popis obsahu přiloženého CD/DVD.

`bin/`
: Firmware na hratelný přímo do řídící jednotky (DCU).

`doc/`
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

`src/`
: Kompletní zdrojové texty programu DCU. Knihovna LOOL je součastí.

`readme.txt`
: Instrukce pro instalaci a nahrání programu DCU, včetně všech požadavků pro jeho bezproblémový provoz.

Literatura

[1] Wikipedie. PWM [online]. 2018 [cit. 2020-08-13]. Dostupné z: https://cs.wikipedia.org/wiki/PWM

[2] Developer android.[online]. 2019 [cit. 2020-08-13]. Dostupné z: https://developer.android.com/things/sdk/pio/pwm?hl=pt-br

[3] Platform IO [online]. Dostupné z: https://platformio.org/platformio-ide

[4] STMicroelectronics datasheet [online]. Dostupné z: https://www.st.com/resource/en/datasheet/stm32f091vc.pdf

[5] FreeRTOS Dostupné z: https://www.freertos.org/

[6] STMicroelectronics MCU [online]. Dostupné z: https://www.st.com/en/microcontrollers-microprocessors/stm32f091vc.html

[7] STMicroelectronics MCU Reference manual [online]. Dostupné z: https://www.st.com/resource/en/reference_manual/dm00031936-stm32f0x1stm32f0x2stm32f0x8-advanced-armbased-32bit-mcus-stmicroelectronics.pdf

[8] STMicroelectronics MCU datasheet [online]. Dostupné z: https://www.st.com/resource/en/datasheet/stm32f091vc.pdf

[9] LL, HAL libraly [online]. Dostupné z: https://www.st.com/resource/en/user_manual/dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroelectronics.pdf

[10] STM32CubeProgrammer [online]. Dostupné z: https://www.st.com/en/development-tools/stm32cubeprog.html#get-software

[11] STMicroeletronics programátor [online]. Dostupné z: https://estore.st.com/en/stlink-v3set-cpn.html

[12] Termite [online]. Dostupné z: https://www.compuphase.com/software_termite.htm

[13] Arduino shop USB TTL převodník [online]. Dostupné z: https://arduino-shop.cz/arduino/1158-eses-cp2102-usb-ttl-prevodnik.html?fbclid=IwAR11rqB-0KIzC45vbEz5tJf_yG9iD6tngLG-9UaxhyJhxe2VlrT0Qk2qtPs

[14] Segmentový display [online]. Dostupné z: https://asset.conrad.com/media10/add/160267/c1/-/en/001050489DS01/001050489DS01.pdf

[15] Vladimír Mejzlík kapitola4.1.1 [online]. 2008 Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=9497

[16] Wikipedie. osciloskop [online]. 2020 [cit. 2020-08-13]. Dostupné z: https://cs.wikipedia.org/wiki/Osciloskop

[17] ResearchGate. osciloskop [online]. 2020 Dostupné z: https://www.researchgate.net/figure/Operation-of-the-autoreload-PWM-timer-CPU-controls-the-period-Ts-with-the-Reload_fig3_3169782