TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Výukový model manipulační plošiny

Bakalářská práce

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Autor práce: **Martin Kastner**
Vedoucí práce: Ing. Petr Školník, Ph.D.

Liberec 2016

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Realizatoin of a lift lab station

Bachelor thesis

Study programme: B2612 – Electrical Engineering and Informatics
Study branch: 2612R011 – Electronic Information and Control Systems

Author: **Martin Kastner**
Supervisor: Ing. Petr Školník, Ph.D.

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin Kastner**
Osobní číslo:	**M12000053**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronické informační a řídicí systémy**
Název tématu:	**Výukový model manipulační plošiny**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s požadavky na realizaci a využití výukového modelu.
2. Navrhněte mechanické a elektrické řešení/zapojení, respektuje požadavky na vstupně-výstupní komunikaci s řídicím systémem.
3. Vyberte vhodné součástky pro realizaci (akční členy, sensory, atd.).
4. Zkonstruujte a oživte laboratorní úlohu.
5. Ověřte její funkčnost.
6. Navrhněte a vytvořte zadání laboratorních cvičení pro tuto úlohu.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **cca 30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **HW.CZ. Vše o elektronice a programování [online]. 2013 [cit. 2013-09-20]. Dostupné z: http://www.hw.cz/**

[2] **DIEDRICH, Kurt. Elektronika tajemství zbavená: objevovat, experimentovat, porozumět. 1. čes. vyd. Překlad Miroslav Hrdina. Ostrava: HEL, 2004, 207 s. ISBN 80-861-6724-0.**

[3] **Senzory a převodníky. 1. vyd. Praha: Vydavatelství ČVUT, 2005, 136 s. ISBN 80-010-3123-3.**

Vedoucí bakalářské práce: **Ing. Petr Školník, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2015**

Termín odevzdání bakalářské práce: **16. května 2016**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2015

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 15.5.2016

Podpis:

Poděkování

Rád bych na tomto místě poděkoval všem, kteří mě při mém úsilí podporovali, ať už se jednalo o morální podporu, která přicházela především od rodiny, tak i o podporu materiální nebo konzultačního charakteru. Za rady i za poskytnuté zařízení bych poděkoval mým spolupracovníkům z firmy ConTeK spol. s r. o. A v neposlední řadě patří mé díky vedoucímu Panu inženýru Petru Školníkovi za shovívavost při ne vždy časté a jasné komunikaci, jakou si vedení bakalářské práce vyžaduje. Nakonec bych rád poděkoval nejmenovanému studentovi, který přede mnou vytvářel model, potřebný k dokončení mé práce.

Abstrakt

Tato bakalářská práce shrnuje popis mého úsilí z více oborů činností, které jsem použil pro vytvoření laboratorní úlohy manipulační plošiny. Jako první úsek je zde mechanická práce, potřebná pro dodělání modelu. Následuje vysokoškolsky zajímavější činnosti a to konkrétně návrh a elektrické řešení daného problému. Po úspěšném návrhu následovala výroba nového řídicího systému a jeho oživení. Jedním z problému při návrhu, bylo rozmyšlení, jaké součástky použiji (jako akční členy, senzory atd.), měl jsem zde ovšem volnou ruku a tak problém spočíval pouze v návrhu měřících postupů, ať už se jednalo o měření polohy nebo rychlosti kabiny. Je zde použita zpětnovazební regulace otáček, což vyžadovalo měření rychlosti. Jako další podstatná část této práce byla tvorba samostatného softwaru pro ovládání modelu. Jak jsem již naznačil, budeme zde rychlost řídit zpětnovazebně a k tomu nám pomůže PID regulátor. V závěru této práce najdeme mnou naměřená data plus komentář co by se dalo zlepšit a co naopak považuji za dostatečně prověřené s ohledem na budoucnost modelu.

Klíčová slova

model výtahu, měření proudu a otáček motoru, řízení stejnosměrného motoru, PID regulace, tvorba nového řídícího systému

Abstract

This bachelor work contains a description of my efforts from multiple lines of business, which I used to create a laboratory assigment handling platforms. The first section of my activity is a description of the mechanical work needed for completion of the model. This is followed by academicaly more interesting activities, specifically the design and electrical solutions of the given problem. After a successful design came production of a new control system and its recovery. One of the problems in the design was, to decide which components I use (as actuators, sensort etc.), I had a free hand here so the problem was only in the design of measuring procedures, whether it was the measurement of position or speed of the cab. I used feedback regulation of revolutions which required speed measuring. Another substantial part of this work was to create a separate software to control the model. As I have indicated, we will control the speed feedback and for that we will use help of a PID controller. At the end of the assigment we can find data measured by myself plus commentary of what could be improved, and on the other hand, what I consider to be properly tested with a regard to the future of the model.

Keywords

elevator model, measurement of a current and revolutions of an engine, DC motor control, PID regulation, creation of new control system

Obsah

Seznam symbolů, zkratek a termínů

Offpage konektor – propojka mezi listy
ADC, AD – (analog to digital convertor) analogově-digitální převodník
RTC – (real time clock) hodiny reálného času
PWM – (pulse-width modulation) pulzně šířková modulace
MOSFET - (Metal Oxide Semiconductor Field Effect Transistor) je polem řízený tranzistor
PID – regulátor s proporcionální, integrační a derivační části
GND – zem, nulový potenciál
VCC – napájecí napětí
Vref – referenční napětí
Enable – povolení činnosti
*.h, *.c, *_CTX – (*) znázorňuje, že nezáleží na uvedených písmenech, př: fronta.h
Tick – jeden přístup vykonání automatu
timeTick – časový hodinový puls vykonání automatu
fastTick, mainTick, realTick – názvy hodinových pulzů automatů s různými periodami volání
observer – pozorovatel
request, req – požadavek, žádost
state – stav
výčtový typ – typ tvořený konečnou množinou pojmenovaných hodnot
struktura – vnitřní uspořádání objektu, můžou zde být proměnné, konstanty, struktury
konstruktor – speciální metoda pro vytvoření instance objektu
booleovská proměnná – proměnná se dvěmi stavy (TRUE, FALSE)
dir, direciton – směr pohybu (nahoru, dolu, někdy i stop)
IO – (input, outpu) vstupy výstupy
DPS – deska plošných spojů
pad – ploška na DPS pro připájení vývodů součástky
SMD - Surface Mount Device- součástka pro povrchovou montáž
THT - Through-hole technology

1 Úvod

Toto téma bakalářské práce jsem si vybral, protože mě hned zaujala možnost vytvořit něco užitečného pro měřící laboratoře. Již v minulosti jsem slyšel, že některé z laboratorních úloh tvořili studenti a už tehdy jsem to chtěl také zkusit. Druhý důvod, proč jsem si toto téma vybral, byla možnost volby různého způsobů řešení, použitých komponent a součástek, téměř bez omezení. Protože ve svém volném čase pracuji ve firmě zabývající se automatizací, regulací a řízením, měl jsem možnosti, prostředky i znalosti na výrobu vlastní řídicí jednotky pro danou bakalářskou práci. Prvním úkolem bakalářské práce bylo seznámit se stávajícím stavem modelu výtahu, který přede mnou již někdo řešil a měl již rozpracovanou mechanickou část konstrukce. Tento model jsem doplnil o novou pohonovou část tvořenou stejnosměrným motorem a převodovkou. Dále jsem pracoval na návrhu elektronické části modelu. V zadání práce byla podmínka oboustranné komunikace modelu výtahu s počítačem PC. Protože jsem navrhoval vlastní řídicí jednotku, nebyl žádný problém tomuto požadavku vyhovět. Pomocí externích vstupů a výstupů řídící jednotky je možné celý model ovládat a zároveň na něm provádět potřebná měření rychlosti pohybu a proudu motoru. Třetí bod zadání byl zvolit správné akční členy, senzory a ostatní prvky systému. Nejvíce času jsem strávil s vlastním návrhem, realizací, oživením a odzkoušením řídící jednotky a jejím programovým vybavením. Oživení a odzkoušení funkčnosti modelu výtahu i jednotlivých jeho částí bylo opakovaně prováděno, protože se vyvíjela nová řídící jednotka. Nakonec jsem vymýšlel zadání výukových laboratorních úloh.

2 Dokončení mechanické části modelu

2.1 Počáteční stav modelu

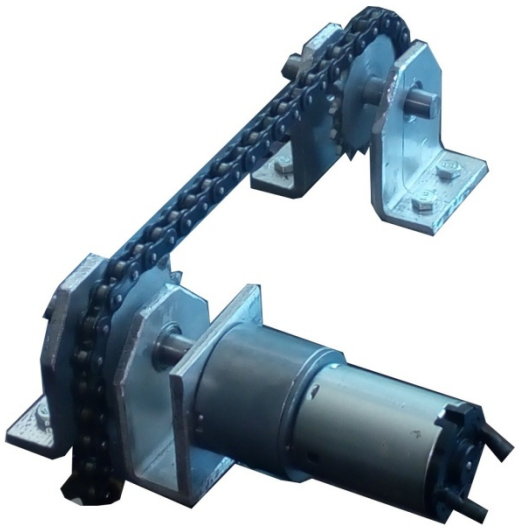
Model jsem dostal již v rozpracované podobě, byla svařená kovová konstrukce a vymyšleny vodící tyče kabiny a protizávaží. V horní části byla bakelitová deska s vytvořenými otvory pro řetěz a připevnění motoru. Model kabiny byl již také hotov. Magnetický pás snímače polohy již byl rovněž nalepen na kovovém profilu a připevněn ke konstrukci.

Obrázek 1: model ve stavu, ve kterém jsem ho dostal

2.2 Práce na modelu

Protože byla kovová konstrukce odřená a na některých místech začínala korodovat, jsem celý stávající model rozšrouboval, kovové části obrousil a nastříkal černou barvou. Poté jsem opět model složil dohromady. Mechaniku modelu jsem dořešil tak, že pohon výtahu i ozubená kola jsem umístil nad horní desku. Od ozubených kol vede dolů řetěz, na kterém je připevněna kabina a protizávaží. Potřeboval jsem vymyslet, jak připevním kabinu k vodícím tyčím, aby měla co nejmenší třecí odpor. Pomohl jsem si zbytkem plechového žlabu, na kterém byl již z výroby vytvořený půlkruhový ohyb. Z tohoto žlabu jsem si vyrobil čtyři vodiče, které jsem připevnil na kabinu, aby jezdila pod stejným sklonem a ve stejné vzdálenosti od stacionární části magnetického senzoru polohy. Na model kabiny jsem pouze upevnil snímač polohy a konektor připojení plochého kabelu k tomuto senzoru. Dalším problémem, který jsem řešil, bylo vedení protizávaží. Pomohl jsem si opět plechovým žlabem, na který jsem protizávaží upevnil. Hmotnost protizávaží

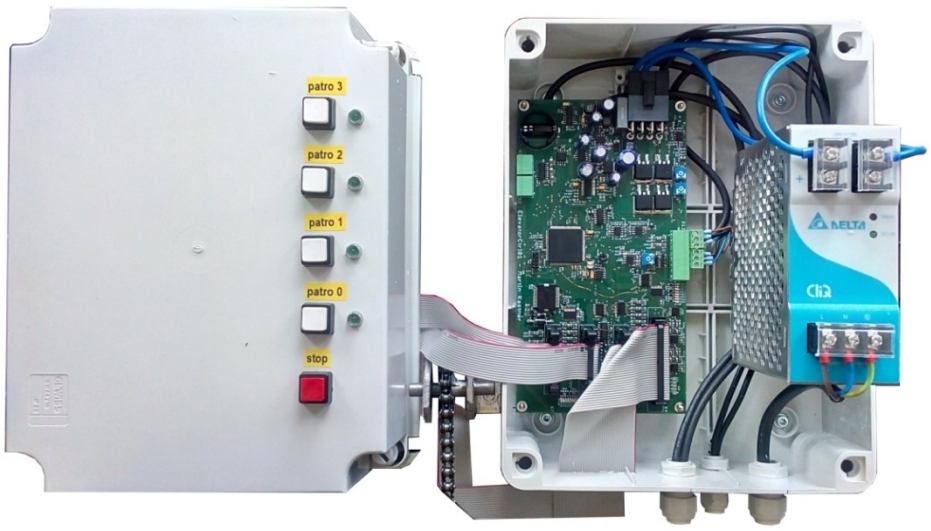
je rovná váze kabiny. V horní bakelitové desce jsem provrtal otvory, aby skrz ně mohl procházet řetěz. Na vrchní stranu desky jsem připevnil ozubená kola a motor s převodovkou. Pro vytvoření domečku pro ložiska a držáku pro motor jsem použil široký hliníkový hliníkový plech, který jsem uřízl, naohýbal a provrtal, tak aby vyhovoval požadavkům modelu.



Obrázek 2: domečky pro ložiska s ozubenými koly, řetězem a motorem

Osičky ozubených kol jsem vybavil ložisky, aby třecí odpor soustavy byl co nejmenší a model se co nejvíce přiblížil reálnému výtahu.

Dále jsem zajistil plastový elektrický rozvaděč, do něž jsem vyvrtal otvory pro průchodky na kabely a otvory pro tlačítka pro ovládání a led diody pro signalizaci. Dovnitř rozvaděče jsem přinýtoval DIN lištu pro připevnění spínaného zdroje napájení a sloupky na připevnění řídicího systému. Elektrický rozvaděč jsem upevnil ke konstrukci výtahu.



Obrázek 3: foto rozvaděče

3 Výběr komponent

3.1 Senzory a akční členy

Důležitou stránkou celé bakalářské práce byl výběr správného motoru. Potřeboval jsem stejnosměrný 12V motor, ideálně i s převodovkou. Vedoucí práce mi doporučil, abych použil dvanáctivoltový stejnosměrný motor, podobný motoru, který se používal v autíčkách na elektrickou autodráhu. Začal jsem hledat, jak jsou tyto motory výkonné a jaký mají krouticí moment. Zjistil jsem, že jejich krouticí moment je malý, a proto jsem musel začít uvažovat o menší velikosti ozubeného kola. Rozhodl jsem se pro průměr ozubeného kola 41mm. Samozřejmě jsem dbal na rozteč a rozměry řetězu. Dále pak jsem potřeboval určit, jak výkonný motor budu potřebovat, aby nebyl zbytečně předimenzovaný. K tomuto výpočtu jsem potřeboval vědět, jak velkou hmotnost závaží bude výtah zvedat jako zátěž. Bylo stanoveno, že jako maximální zátěž bude stačit hmotnost jeden kilogram. Pomocí vzorce pro krouticí moment (1) a ze znalosti vzorce (2) a (3), jsem spočítal, že minimální krouticí moment pro toto zadání musí být minimálně 205mNm. Našel jsem stejnosměrný motor s převodovkou o krouticím momentu 300mNm, což umožní zvedat závaží o hmotnosti až 1,5kg.

$$M = F * r \tag{1}$$

$$r = d_z/2 \tag{2}$$

$$F = m * g \tag{3}$$

$$v = l/t \tag{4}$$

$$l = \pi * d_z * ot \tag{5}$$

Díky stejné váze kabiny a protizávaží je možné využít všechen krouticí moment pohonu na zvedání zátěže. Tento pohon má při chodu na prázdno 18 otáček za minutu a odběr 0,1A při napájecím napětí 12V. Průměr hřídele převodovky je pouze 5mm, zatímco průměr otvoru v ozubeném kole je 10mm. Potřeboval jsem tedy motorovou hřídel zvětšit. Vybral jsem osičku o průměru 10mm, nechal do ní vysoustružit otvor pro hřídel převodovky. Jelikož motorová hřídel byla z jedné strany plochá, udělal jsem do hřídele ozubeného kola otvor se závitem a zajistil spojení převodovky a hřídele šroubem. Podle vzorce (4) a (5) jsem spočítal, že maximální rychlost výtahu bude 2,2 metru za minutu, což by mělo být pro model o výšce 1,5 metru dostačující.

Dalším úkolem bylo vybrat vhodné snímače. Začal jsem s koncovými snímači, které hlídají pohyb kabiny ve vymezeném prostoru, aby výtah nepřejel hranice, za které se nesmí dostat. Pokud by tyto hranice kabina přejela, mohlo by dojít ke zničení motoru nebo jeho ovládání, protože by se kabina nebo závaží zasekly o rám modelu. Z tohoto důvodu jsem zvolil snímače typu normal close (dále jen NC), aby byla detekována chyba i v případě přerušeného přívodního kabelu. V programovém vybavení řídící jednotky je možné provést inverzi daného digitálního vstupů. Pro správnou bezpečnostní funkci je nutné zabezpečit, aby když výtah najede na snímač, byl vstup na úrovni logické nuly. Dále pak bylo potřeba vyřešit otázku měření vzdálenosti. Toto již vyřešil můj předchůdce, který na modelu výtahu pracoval. Proto jsem dostal s konstrukcí výtahu i lineární magnetický inkrementální snímač. Tento snímač funguje na principu třech signálů.

Signály A a B jsou vůči sobě posunuté a z jejich kombinace lze určit polohu, rychlost a směr pohybu. Třetí signál Z je pro určení nulového bodu. V modelu výtahu jsou využívány pouze signály A a B.
Ovládací tlačítka a signalizační LED diody byly použity běžně dostupné. Za zmínku stojí pouze využití NC spínače pro tlačítko Stop, a to ze stejných bezpečnostních důvodů, jako u použití NC snímačů krajních polohy.
Pro napájení elektroniky modelu výtahu byl použit spínaný zdroj napětí firmy Delta Elektronic, který z 230V střídavých vytvoří 24V stejnosměrných o maximálním výstupním proudu 5A. Tento zdroj by měl vyhovět pro napájení všech obvodů včetně stejnosměrného motoru.
Při výběru řídicí jednotky modelu výtahu jsem dostal od vedoucího práce volnou ruku. Pokud bych použil nějakou předem vytvořenou řídicí jednotku, musel bych také dodělávat obvody řízení motoru, měření otáček i rychlosti. Proto jsem se domluvil s vedoucím práce, že ideálním řešením bude vývoj vlastní řídicí jednotky modelu.
Jako poslední komponenta, kterou jsem potřeboval vyřešit, byla brzda. Vedle vodících tyčí je umístěna ještě jedna vzpěrná tyč, která by se dala využít k řešení tohoto problému. Také by se daly použít vodící tyče, to by ovšem bylo méně elegantní. Nejjednodušší způsob jak udělat brzdu by byl, obyčejný elektromagnet přidělaný ke kabině výtahu. Ten by zamezil jejímu pohybu dolů, pouhým přitažením k tyči. Ovšem daleko lepší způsob jak tuto problematiku vyřešit, se nabízí možnost udělat obrácený elektromagnet, jenž by nepřitahoval kabinu k tyči, ale odtahoval by nějakou brzdu směrem od tyče. Nejsložitější by na tomto řešení bylo vymyslet samotnou brzdu. Mohla by se využit pružinou vysouvaná kovová tyč s gumovou koncovkou pro větší třecí odpor. Silu pružiny by v případě požadavku odbrzdění přemáhala síla elektromagnetu. Opět, kvůli bezpečnosti, je toto řešení vhodnější jako klidová brzda.

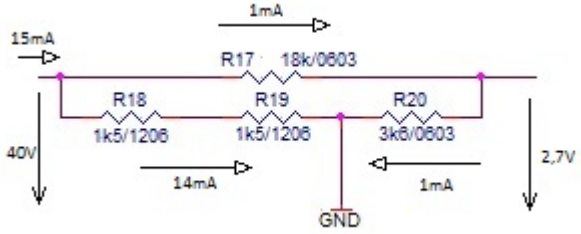
3.2 Důležité komponenty řídicí jednotky

Pro řízení byl použit 32-bitový RISC mikroprocesor firmy Atmer s jádrem typu ARM® Cortex®-M4 s pracovní frekvencí 120 MHz, obsahující FPU a 2 kByte vyrovnávací pamětí cache. Disponuje vestavěnou Flash pamětí o velikosti 1 MByte, 128kB SRAM paměti a obsahuje velké množství periferií, jako například SPI, I2C, USB nebo Ethernet. Dále obsahuje čtyři kanály PWM, tříkrálové 32bitové čítače/časovače a 12ti bitové AD a DA převodníky. Volba tohoto procesoru byla jednoduchá, protože již dříve jsem jej využíval v zaměstnání pro jiné aplikace.
Pro napájení obvodů na řídicí jednotce byl použit pulzní napájecí zdroj MC34063 v doporučeném zapojení výrobce, s ověřenými hodnotami součástek.
Vzhledem k malé zátěži napájecího napětí 3,3V byl použit lineární stabilizátor LP2951CD v doporučeném zapojení.
Pro napájení budiče tranzistorů můstku stejnosměrného motoru bylo třeba vytvořit napájecí napětí 12V. Protože není třeba velkého proudu, byl použit lineární stabilizátor napětí LM8712. Pro dobrou funkci stabilizátoru je třeba vstupní napětí od 19V do 35V. Tato podmínka je splněna při napájení jednotky z napětí 24V. Výstupní napětí stabilizátoru má toleranci ±0,5V, což je pro dané potřeby dostačující.
Pro spolehlivý chod procesoru byly ochráněny všechny vstupy a výstupy systému.

Digitální vstupy jsou zatíženy rezistorem 3kΩ a přes sériový rezistor jsou přivedeny do vstupního oddělovače 74VHC541. Oddělovače obsahují CMOS technologií vytvořené diody, které při přepětí nad napájecí napětí svedou přebytek energie do zdroje, který napájí tento čip. Naopak při záporném napětí na vstupu je energie svedena do GND. Ukážeme si na příkladu, jak ochrana funguje. Na vstup, který je dimenzovaný na 24V přivedeme napětí o hodnotě 40V. Z obrázku je patrné, že nejprve je zde zatěžovací odpor a dělič napětí. Vstup je zatížený zatěžovacím odporem a podle ohmova zákonu (6) jím proteče 14mA. Dále si spočítáme odpor celého vstupu, tedy obou děličů dohromady (7), což nám dá odpor 2,6kΩ. Ze vzorce (6) vyplývá, že při napájení 40V, teče do vstupu proud 15mA, ze kterých 14mA teče zatěžovacím odporem. Zbude nám 1mA, která vytvoří úbytek na druhém odporu napěťového děliče napětí o hodnotě 2,7V. Toto napětí je zcela v toleranci vstupního napětí oddělovače. Problém by mohl nastat, pokud by toto napětí bylo větší než napájecí napětí bufferu. V tom případě by se svedl přebytek energie do zdroje napájení. Tato přebytečná energie se dokáže využít v připojených součástkách. Pokud bychom však dodali moc velkou energii, zničil by se zdroj tohoto napětí a pravděpodobně i součástky z něho napájené.

$$U = R * I \tag{6}$$

$$R_c = \frac{(R_{18}+R_{19})*(R_{17}+R_{20})}{(R_{18}+R_{19})+(R_{17}+R_{20})} \tag{7}$$

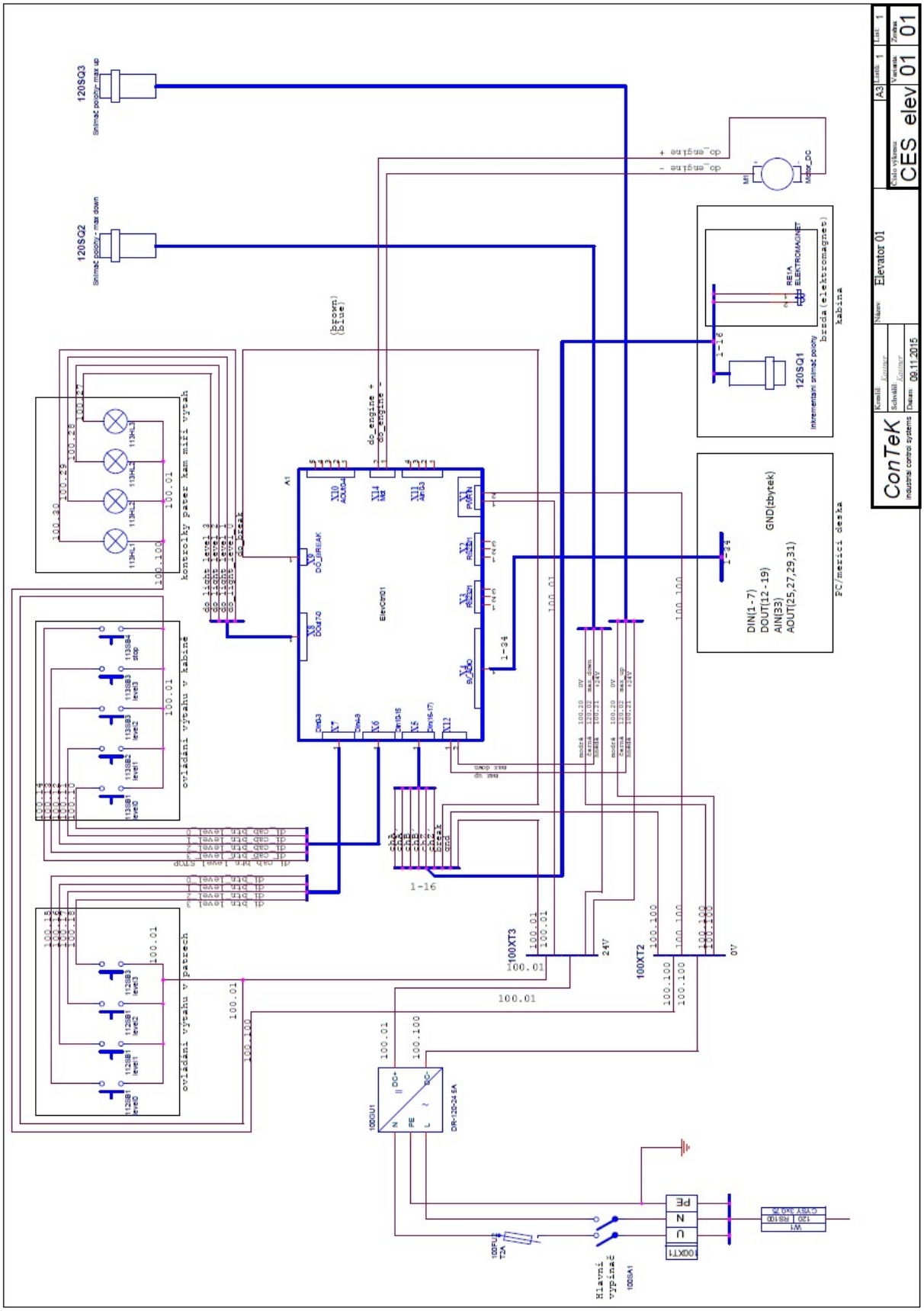


Obrázek 4: počítání na děliči

4 Návrh elektrická část modelu

4.1 Popis blokového schématu

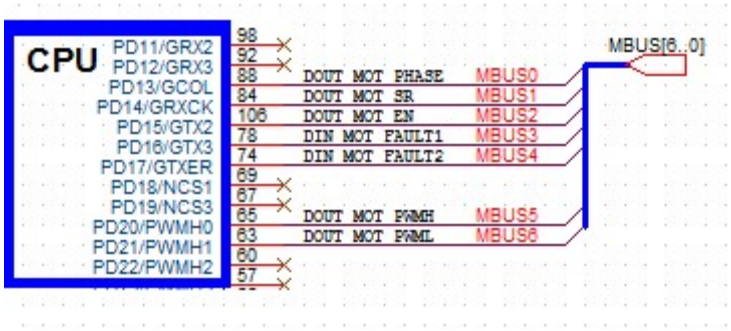
Návrh schémat probíhal v programu Orcad. Na blokovém schématu vlevo dole je naznačený přívod elektrické energie o napětí 230V střídavých.Přívodní kabel musí být třívodičový s ochranným vodičem PE, podle požadavků elektrických norem. Spínaný zdroj vytváří na svojí svorkovnici 24V a 0V pro připojení řídící jednotky. Řídicí jednotka zabezpečuje napájení ostatních elektrických komponent modelu výtahu. Na schématu nad řídicím systémem jsou umístěna tlačítka ovládání z jednotlivých pater i z kabiny výtahu. V horní části schémat jsou umístěny led diody znázorňující, kam výtah míří. Vpravo nahoře jsou umístěny dva indukční snímače plnící funkci koncových spínačů. Pod nimi je znázorněn motor, inkrementální magnetický snímač polohy a elektromagnetická brzda.



Obrázek 5:blokové schéma elektrické časti modelu

4.2 Popis schématu řídicí jednotky

Návrh schématu řídicí desky probíhal v programu Orcad. Toto schéma je již složitější, a proto je umístěno v přílohách bakalářské práce. Je rozdělené do sedmi listů, na nichž jsou jednotlivé části celé řídicí jednotky. Najdeme zde listy obsahující napájecí zdroje, CPU, digitální vstupy, digitální výstupy, analogové vstupy a výstupy, list s obvody řízení DC motoru a list s připojením měřící počítačové karty v 5V logice, s digitálními i analogovými vstupy a výstupy. Signály mezi listy jsou propojeny takzvanými offpage konektory. Vede-li více signálů pospolu, je využito sběrnicového připojení, pro zjednodušení schématu. Toto využití sběrnice není ovšem obvyklé (obvyklé využití je více zařízení připojených na jenom datovém a adresovém kanálu), jedná se pouze sdružení signálů. Všechna jména signálu jsou vztažena k řídicí jednotce, jejímu procesoru.



Obrázek 6: ukázka sběrnice a offpage konektoru

4.2.1 List: 01. Power

Na tomto listu jsou znázorněny všechny interní zdroje napájení. Je zde přívodní napájecí konektor s 24V a připojení stejnosměrného motoru výtahu. Na vstupu je nejprve vyfiltr a poté je toto napětí přímo využito jako zdroj napětí pro silovou část jednotky. Dále se z tohoto napětí pomocí regulátorů udělá 12V, 5V a 3,3V. Na tomto listu je i referenční napětí 3,3V pro AD převodník procesoru jako referenční napětí.

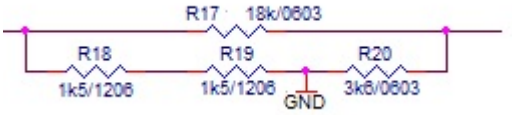
4.2.2 List: 02. CPU

Zde je zapojení procesoru a nezbytných součástek, které s ním souvisí. V pravém horním rohu se nachází obvody rozrhraní RS232, realizovaného pomocí obvoduMAX3232. Jsou k dispozici dva kanály sériové komunikace COM s konektory. Je zde připojení 32Mb FLASH paměti určené pro data aplikace. Vlevo od procesoru se nachází obvod RTC a paměť FRAM. Použitý procesor má pouzdro s 117 IO nožiček s různými funkcemi. Digitální vstupy a výstupy jsou nejjednodušší, protože se dají připojit k jakémukoliv V/V pinu. Analogové výstupy se dají připojit pouze k pinům umožňujícím PWM, čítače nebo DAC. Analogové vstupy se musí připojit k pinům AD převodníku. Jsou zde vývody pro timer/couter, na nichž je připojen například inkrementální snímač polohy. Snímač polohy je zapojen podle datasheetu tohoto procesoru [1] pro připojení na Quadrature Decoder. Externí zdroj kmitočtu je uveden vpravo vedle procesoru. Pod krystalem je konektor, který umožňuje smazat veškerá data z celé vnitřní paměti Flash. Tento konektor není z důvodu bezpečnosti osazen.

Ještě níž pod krystalem se nacházejí signalizační LED diody, které jsou využity pro znázornění stavu činnosti procesoru, abychom rozeznali správnou funkci procesoru. Různé programy můžou s těmito dvěma LED diodami blikat rozdílně. Například program BoatLoader střídá blikání obou led diod, program IOtest, se kterým jsem testoval vstupy a výstupy, bliká s jednou led diodou a můj program pro řízení výtahu bliká pouze s druhou diodou. Pod těmito led diodami se nachází čtyřpinový konektor. Dva jeho piny slouží BoatLoaderu k rozhodnutí zda má spustit aplikační program po zapnutí napájení. Druhé dva piny složí k rozhodnutí, zda se mají aplikační data vymazat z nevolatilní paměti. Na konektoru X13 je vyvedena komunikace JTAG, která slouží mimo jiné pro nahrání BoatLoaderu.
Z procesoru nám jdou tedy svazky do ostatních listů jako například DINEXT[5..0]. Kódování těchto sběrnic je jednoduché JMENO[x..y]. Signály jdoucí tímto svazkem se musí jmenovat JMENOx až JMENOy. Máme tu sběrnice jdoucí k digitálním a analogovým vstupům, výstupům, k 5V konektoru nebo k čipu PWM pro řízení motoru.

4.2.3 List: 03. DIN

Následuje zapojení a úprava digitálních vstupů. Protože procesor pracuje na 3,3V logice, musíme 24V nebo 5V vstupní signály snížit pomocí děliče napětí a proudu.



Obrázek 7: Ukázka děličů

Napěťový dělič spočítáme podle vzorce

$$U_{vyst} = \frac{U_{vst} * R20}{R17 + R20}$$

a proudový podle

$$I_{vyst} = \frac{I_{vst} * R17}{R17 + R18 + R19}$$

Na pravé části listu vidíme čtveřici konektorů X7,X6,X5 a X12. X6 ta má za úkol připojení ovládání v kabině, kde je na rozdíl od jednotlivých pater i tlačítko pro zastavení. X7 reprezentuje připojení ovládání v jednotlivých patrech a diodová signalizace ukazuje, kam míří kabina. Na konektor X5 se připojuje magnetický inkrementální snímač polohy kabiny. Každý ze tři výstupů snímače má signál negace. Všechny signály musí být upraveny odporem a ochrannými diodami. Upravené signály vedou do lineárního diferenciálního přijímače [2], který porovná signál a jeho negaci. Který ze signálu bude větší, taková hodnota bude na výstupu přijímače. Negovaný signál je tu z důvodu odstranění rušení. Ochranné diody pracují na principu předání přebytečné energie do napájecího napětí. Pokud na vstup připojíme například 10V, sériový rezistor a dioda sníží toto napětí na napětí zdroje. Snímač polohy je 5V logiky, a proto i jeho výstupy jsou v této logice. Na napěťovou logiku procesoru nepřecházíme pomocí děliče ale pomocí oddělovače [3]. Konektor X12 slouží pro připojení koncových snímačů. Z konektorů vedou všechny signály do děliče napětí a do vstupního oddělovače. Zde jsou tyto signály upraveny a vedou přímo ke vstupům procesoru. Předtím ještě rozsvítí signalizační LED diody.

4.2.4 List: 04. DOUT

Zde jsou obsaženy digitální výstupy, předtím než se signály dostanou do konektorů. Cesta signálu začíná u procesoru, poté jde na list DOUT, kde pokračuje do výstupního třístavového budiče. Následují signalizační diody a konektor 5V. Zároveň postupuje přes rezistor do nízkoprahového spínače [4]. Zjednodušeně se jedná o CMOS spínač, který připne napájecí napětí na výstup. Poté už vede přes dva rezistory pro signalizační LED diody a do konektoru. Rezistory před spínačem chrání další součástky před zničením v případě poruchy výkonového spínače. Na výstupní signál DO_BRAKE se připojí indukční zátěž, a proto je vybaven antiparalelní diodou a transilem, aby se zmařila energie indukovaná cívkou brzdy. Výstupy budičů nejsou aktivní, dokud signál nOUT_EN není procesorem nastaven do log. 0 a zároveň není aktivní signál nRESET. Tuto bezpečnostní funkci zajišťuje N-MOS tranzistor. Toto zapojení se používá pro zajištění správného chodu výstupů při startu jednotky. Není tedy nijak možné, aby se na výstupu objevila log 1, není-li procesor nastartovaný a sám o to nepožádal.

4.2.5 List: 05. AIO

Protože analogových výstupů a vstupů je málo, celkem devět, z toho jsou čtyři na jiném listu schématu, dovolil jsem si dát vstupy a výstupy na jeden list. Začněme analogovým vstupem, ten je pouze jeden. Je zde jako rezervní - pro vnější použití. Je takzvaný diferenciální, což znamená, že v konektoru jsou dva diferenciální vstupy, jejichž rozdíl se zesílí. V konektoru jsou také Gnd a VCC5V z důvodu možného využití pro potřeby napájení dané periferie. Oba signály se mohou nejprve upravit, aby mohly do vstupu diferenciálního operačního zesilovače napájeného pouze 5V. Výhodou tohoto diferenciálního zesilovače je, že vstupy nejsou ničím zatíženy. Je zde propojka PJ3, která slouží ke kalibraci. Rezistory R94 a R95 jsou zde pro změnu zesílení zesilovače. Ostatní analogové vstupy jsou popsány na listu motoru, protože se týkají jeho měření. Poslední analogový vstup se nachází na listě vstupně výstupní komunikace s počítačem PC.

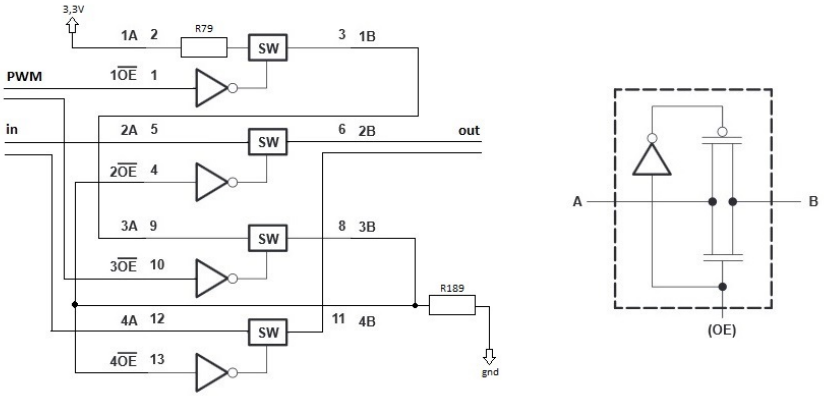
Analogové výstupy jsou čtyři a jsou vytvářeny pomocí PWM výstupů a Timer/Couter výstupů. Signály nejprve vedou od procesoru k výstupnímu budiči. Rezistory za budičem slouží k vytvoření nulového napětí v případě nepovolení výstupů. Za nimi následují dvojité RC články, které vytváří analogové hodnoty v rozsahu 0-5V. Výstup z RC článků je přiveden na neinvertující vysokoimpedanční vstupy operačních zesilovačů v zapojení sledovačů vstupního napětí.

4.2.6 List: 06.DCmotor

Na tomto listu probereme ovládání a řízení stejnosměrného motoru. Pro řízení motoru nám poslouží budič řízení úplného MOSFET můstku. Ten nám bude ovládat otevírání čtyř tranzistorů můstku, přes které teče proud do stejnosměrného motoru. Při otevření pravého horního a levého dolního tranzistoru, poteče proud motorem a motor se roztočí na jednu stranu. Sepnutím opačných tranzistorů, poteče proud opačným směrem a i motor se bude točit opačně. Tím je vyřešena změna směru otáčení motoru. Otáčky motoru se řídí pomocí PWM modulace, která je přivedena na odpovídající

vstup budiče můstku.
Tento list by se dal rozdělit na dvě části, motorovou a měřící. Motorová část je jednodušší a proto s popisem začneme u ní. Z CPU jsou vyvedeny dva digitální signály pro řízení PWM, jeden digitální pro směr, signál Enable a nakonec signál SR, který dokáže synchronizovat PWM vstupy. Pro mé potřeby tento signál není potřeba, proto jej nebudu dále popisovat. Tyto signály vedou do budiče, který nám zajistí otevíraní tranzistorového můstku. Podle [5] má čip více typů řízení. Je zde řízení pomocí pouze digitálních vstupů, což není vhodné, protože to je pouze řízení směru a rychlost je řízena také digitálním signálem. V dalším typu řízení se uplatňují PWM místo pouhého digitálního řízení. Nastavují se zde PWM na vstupech pwmh a pwml (tímto způsobem jsem řídil tento budič). Jako další možnost řízení by bylo vhodné dát PWM signál na vstup phase a na pwml a pwmh log 1, což by ovšem znamenalo mnohem hrubší řízení. Pro otestování tohoto způsobu řízení, ovšem za potřebí předělání mapování vstupů a výstupů. Musel by být signál phase přidělán na IO, který zvládne funkci PWM (na jiný pin), kterých je málo. Jako poslední možnost řízení je s využitím vstupu SR, jeden vstup pwmh nebo pwml dát tvrdě na log 1 a druhý ovládat pomocí PWM, což by mělo za následek při nulovém PWM aktivní brzdění, v ostatních případech se pouze rozepnou tranzistory. Tento způsob není tak náročný na předělání IO a tak bych ho rád otestoval v závěru práce a porovnal. Popisovat zapojení čipu nebudu, jde o doporučené zapojení podle výrobce [5]. Za zmínku by stály digitální výstupy fault1 a fault2, což jsou příznaky chyb. Co přesně představují jednotlivé chyby je uvedeno v [5]. Vstup RDEAD hlídá rychlost spínání tranzistorů, aby tranzistory měly delší čas pro sepnutí a rozepnutí. Potřebuje li tranzistor větší časový úsek pro sepnutí, tento vstup musí být opatřen rezistorem o menším odporu.
V měřící části tohoto listu budou popsány dvě věci, a to měření otáček a měření proudu motorem. Na každé větvi mám měřící odpor, R74 a R192. V jednom okamžiku v závislosti na směru otáčení bude jeden odpor připojen na zem a druhý na +24V. Měření bude probíhat na tom odporu, který je připojený k zemi (znám směr tudíž, i na kterém odporu budu měřit). Nebo se dá poznat, na kterém rezistoru měřit, tak že na výstupu z jednoho AD převodníku bude maximální hodnota a na druhém nějaká blízká 0 (měření probíhá na ADC, kde nebude maximální hodnota). Toto řešení je zde kvůli nemožnosti najít operační zesilovač, který by fungoval od 0–24V Rail to Rail. Na rezistoru o malé hodnotě odporu budu měřit napětí. Ze znalosti Ohmova zákonu a hodnoty odporu se dá dopočítat proud. Tento proud motorem je úměrný momentu. Řekněme, že motorem poteče proud 1A, to na rezistoru o odporu 100mΩ vytvoří napětí 0,1V. Toto napětí je přivedeno na vstup diferencíálního zesilovače. Na výstupu se objeví analogová hodnotu napětí na odporu. Je-li přivedeno referenční napětí v půli chtěného rozsahu, posune se do této hodnoty nula. Tímto způsobem se sice zmenší rozsah, ale výsledek bude od nuly do dvojnásobku referenčního napětí. Před vstupem do procesoru signál vede do RC článku a ochranných diod.
Ve chvíli, kdy motor není saturován, tedy v době nečinnosti PWM, se chová jako generátor, který generuje napětí úměrné otáčkám motoru. Této technologii měření se říká Back Electro Magnetic Flux (BEMF) [6]. Takto se dají elegantně změřit otáčku motoru. Ze znalostí možnosti takto měřit rychlost otáčení můžu řídit PWM pomocí

například zpětnovazebného PID regulátoru. Napětí na motoru přivedu do děliče poté do spínače, který propustí analogový signál dál, pouze tehdy, když PWM signály jsou v nečinnosti. Střídu PWM ovládám digitálními výstupy z procesoru, je tedy jednoduché podle nich spínat spínače. Signál pokračuje dál do integračního článku.



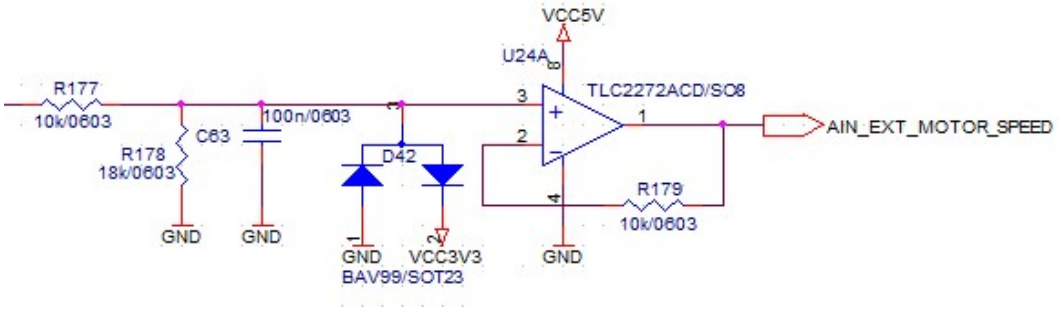
Obrázek 8: zapojení spínače tak, aby propouštěl analogovou hodnotu při nečinnosti PWM a vnitřní zapojení spínače

Poté jsou signály zavedeny do rozdílového zesilovače. Ten na výstup dá hodnoty od -3,3V do +3,3V což je pro nevýhodné, protože procesor nezvládne změřit zápornou hodnotu na analogovém signálu. Muselo byt vymyšleno, jak zapojit referenční napětí, aby výstupní napětí z rozdílového zesilovače bylo v rozsahu 0-3,3V. Toto je umístěno ve schématech pod diferenciálním zesilovačem. Aby -12V odpovídalo 0 a +12V odpovídalo +3,3V, musí se posunout nula do půlky rozsahu. Přesné a stabilizované napětí 3,3V z VREF se vydělí dvěma a zapojí se do reference diferenciálního zesilovače. Výstupní napětí je přivedeno do obvodu sample and hold a před vstup procesoru, je vložena napěťová ochrana. Do procesoru vede analogová hodnota ze zesilovače zajišťujícího referenci.

4.2.7 List: 07.5VCON

Na posledním listu je znázorněno zapojení konektoru pro připojení k měřící desce počítače. Kvůli tomuto připojení musely byt některé signály duplikovány a předělány na 5V logiku. Kabel k připojení jsem zvolil 34 žilový řezný, který je vhodný pro toto zapojení, protože je nejjednodušší na výrobu a snadno se dá dopočítat, co na dané žíle má být za signál. Další výhodu tohoto kabelu je obvyklé zapojení, které je vytvořeno tím, že na lichých vodičích je zem a na sudých je signál. Toto se nejenom dobře zkouší, ale je to i vhodná ochrana proti rušení od ostatních signálů. V mé práci je bohužel potřeba více než 17 signálů pro striktní rozložení na sudé a liché. Byla zde snaha o co největší zmenšení těchto negativních vlivu. Digitální vstupy se rušit navzájem nebudou, a tak mohli být vedeny vedle sebe. Od digitálních výstupů jsou odděleny vodiči se zemí. Nakonec u analogové signály, u kterých je riziko ovlivňování úrovní znatelnější, střídání signálu a země proběhlo. Na tomto listu najdeme sedm digitálních vstupů, které jsou děliči upraveny tak, aby mohly jít do vstupního oddělovače, nacházejícího se na listu digitálních vstupů (vstupy jsou upraveny pouhými děliči). Pak tu je osm digitálních výstupů, čtyři analogové výstupy a jeden analogový vstup, který je jednoduše upraven do logiky vhodné pro procesor. Každý signál, který jde na

pin procesoru, se musí upravit a zajistit, aby se do procesoru nedostala nevhodná hodnota, například větší napětí než 3,3V. Digitálními vstupy je možno docílit posunu výtahu do jednotlivých pater nebo pomocí DI Auto/man, směru a AI ExtMotorSpeed zůe řídit pohyby motoru úplně externě. To se bude hodit zvláště pro potřeby laboratorních úloh spojených s měřením přechodových charakteristik motoru.



Obrázek 9: Analogový vstup externího zadání rychlosti motoru

5 Popis programu pro řízení výtahu

Program je psaný v jazyce C. Program obsahuje vždy hlavičkový soubor *.h a soubor konkrétní knihovny *.c. V hlavičkovém souboru je popis funkcí co budou provádět automaty a deklarace funkcí, v *.c souboru jsou přesně definované funkce. Pro automatickou funkci je program dělaný jako nekonečná smyčka volání jednotlivých stavových automatů, které jsou navzájem propojené. Je tu jeden hlavní automat obstarávající ostatní automaty související s časem a jmenuje se timeticks. Obstarává hlavě automaty FastTick, SlowTick, maintick a realtick.

5.1 TimeTick

Hlavním účelem tohoto automatu je funkce pro práci s časovačem. Tady v téhle knihovně se s každým taktem procesoru připočítá k čítačům jednička. A podle děliče se periodicky tikne s knihovnami ostatních ticku, jako jsou realtick, fasttick nebo slowtick. Real tick je volán se zakázaným přerušením a fast, slow jsou volány s povoleným přerušením.

5.2 FastTick

Automat fast tick se stará a volá automaty, které je potřeba volat rychle jako například tik automatem hlídající zmáčknutí tlačítka, měření nebo automat na obsluhu motoru. Pro simulaci na počítači ještě tiká s automatem simulátoru. Pak se ještě stará o tik automatem pro zápis a čtení vstupů a výstupů.

5.2.1 button_observer

Automat starající se o zmačknuta tlačítka.

V button_observe.h najdeme výčtový typ všech requestů, tedy požadavku na konání pohybu výtahu. Jsou zde požadavky na pohyb do všech pater, požadavek na externí řízení, externí zadaní směru a požadavek zastavení pomocí tlačítka stop. Druhý výčtový typ je naplněný stavy automatu. Následuje struktura kontextu automatu. A jako obvykle konstruktory funkcí. Máme tu funkci, která nám vrátí požadavek z fronty. Nebo funkce, na nastavení příznaku requestDone.

V knihovně buton_observer.c nalezneme nejprve funkci `ButtonObserverGetRequest`, která vrátí requset z fronty, který se bude vykonávat. Je to udělané tak že se zavola funkce z knihovny fronta a to konkrétně `GetNextReq`, do parametru této funkce musíme dát aktuální patro a směr, kterým se chceme pohybovat, to bude potřeba ke správnému určení dalšího patra, kde chceme zastavit.

Funkce `ButtonObserverSetRequestDone` a `ButtonObserverSetRequestNotDone` pouze nastaví do proměnné req_done buď true nebo false.

V této knihovně jsem napsal ještě funkci pro měnění určitých parametrů výtahu pomocí vstupně-výstupní komunikace s PC. Konkrétně rozjížděcích a brzdných ramp, nebo také základní parametry PID regulátoru. Pro zavolání této funkce musí být na digitálních vstupech auto/man a stop logická jedna. Poté už mají ostatní digitální vstupy jinou funkci. Vstup pro jetí do patra nula slouží k nastavení hodnoty analogového vstupu pro zadání rychlosti uložen do parametru P. Samozřejmě analogová hodnota musí být

vydělena, protože při nastavení 5V na analogový vstup uložilo by se do parametrů 65536. U parametrů PID regulátoru jsou hodnoty děleny tak aby rozsah byl 0-10, u nastavování ramp je rozsah 0-2. Digitální vstup pro pohyb do patra jedna slouží k nastavení I, level2 pro nastavení parametru D, level3 pro nastavení rozjížděcí rampy a digitální vstup dir pro nastavení brzdné rampy. V celém odstavci se jedná pouze o externí analogové nebo digitální vstupy.

Následuje automat. Zde jsou stavy finished, ve kterém se nic nedělá, stav restart, který nás posune do stavu bo_state_no_request, tyto stavy nejsou nijak zajímavé. Na rozdíl do stavu no_requset, ve kterém se hlídají všechna tlačítka. Je to tu uděláno jednoduchými podmínkami na zjištění do jakého patra chceme jet. Pouze se zavolají konkrétní vstupy, například chceme jet do nultého patra, zeptáme se pouze, jestli je na digitálním vstupu `DI_BTN_LEVEL_0`, `DI_CAB_BTN_LEVEL_0` nebo `DI_EXT_GO_LEVEL_0` není logická jedna. Je-li, zavolá se funkce pro zapsání requestu do fronty a smaže se `BTN_STOP`. Toto mazání je zde, kvůli tomu, že když jedou zmačkneme tlačítko stop, zůstane ve frontě dokud nezmačkneme jiné tlačítko, tudíž výtah do té doby se nerozjede. To by bylo popsáno ovládání výtahu pomocí tlačítek. V tomto stavu dále kontrolujeme zmačknutí tlačítka stop, auto/man, extdir. Poslední podmínka v tomto stavu je zjištění jestli v proměnné request není `BTN_NO_REQ`. Není-li to tam, tak se skočí na stav bo_state_request. Ovšem toto je tu připravené na případ, kdy se vyhodnotí jeden požadavek a dokud není hodový, tak se neděje nic jiného. Do tohoto stavu se nemůžu nikdy dostat pro tento program.

5.2.2 DCmotor

Ovládání stejnosměrného motoru. V podstatě se jedná o knihovnu, která se stará o obsluhu driveru a ten ovládá H můstek, čímž řídí motor, jeho otáčky, směr a ostatní věci spojené s motorem.

Je zde tu vyčtené stavy automatu starajícího se o řízení driveru. Je tu také kromě defaultních stavů pouze další dva stavy a to stav stop a enable. Dále tu je strukturu RampDcStruc, ve které najdeme proměnné potřebné k rampování rychlosti. A strukturu kontextu automatu. Následují konstruktory funkcí potřebných k ovládání motoru jako například DcMotorStoped, která vrátí hodnotu typu bool jestli motor stoji. Dále pak funkce, která zjistí, jestli se už má začít brzdit. Mimo jiné jsou zde i funkce spojené se samotným automatem.

Zde jsou popsány jednotlivé funkce a to například funkci DCRampSpeed ta má za úkol dostat postupně po rampě aktuální rychlost na chtěnou a poté se pošlou parametry do PID regulátoru, aby nebylo zrychlování nebo brzděni skokové. K účelu postupného zvyšování rychlosti nám pomůže jednoduchá podmínka a to jestli je aktuální rychlost stále menší než požadovaná. Jeli tomu tak, připočteme v každém zavolání této funkce k aktuální rychlosti konstantu, toto platí pro zrychlování, pro zpomalování by to platilo obráceně. Po dokončení přičtení je tato rychlost oříznuta, a protože pracujeme s rychlostí, jako s procentuální hodnotou ořízneme ji v rozsahu 0 až 100. Pak už se jenom tato hodnota do PID regulátoru nahráním do parametru W. Do parametru Y nahrajeme aktuální rychlost, kterou poté přepíšeme parametrem U.

Další funkce se nazývá DCMotorStoped, jak už bylo tvrzeno v hlavičkovém souboru,

má za úkol vrátit jestli motor stojí. Toho lze dosáhnout různými možnostmi, já zde mám dvě možnosti a to přes zjišťování aktuální rychlosti nebo vyhodnocení stavu automatu.
Důležitá funkce je DCMotorBreakingPos. Ta je volána z automatu elevatorMain, ale je umístěná zde, protože souvisí spíše s motorem. Tady se podle aktuální rychlosti a brzdícího koeficientu vypočte vzdálenost, za jakou výtah zabrzdí. Koeficient brzdění je opět pouze proměnná, která se odečítá od rychlosti. Tato funkce vrátí hodnotu, v jakém místě zastaví výtah, pokud začneme okamžitě brzdit.
A teď již k popisu samotného automatu. Ve stavu restart se zjistí, kde se nachází výtah a protože se do tohoto stavu dostaneme obvykle po startu, musíme sjet pomalu k senzoru spodní maximální hranice, tím program zjistí, kde je. Poté se automat může dostat dál a to do stavu stop. Zde se pouze kontroluje, jestli jsme nepřekročili spodní hranici a nechceme pokračovat níž, to samé platí pro horní mez. Do dalšího stavu se můžeme posunout pouze na pokyn automatu elevatorMain. A poslední stav enable, tady se povolí výstup, který povolí driver. Následuje rozhodnutí o směru a tedy o dalším výstupu pro driver, opět je to řízeno automatem elevatorMain. Poslední co se udělá, je, že se zavolá funkce pro rampování rychlosti a zapíše se na výstupní PWM piny hodnota přepočítaná na 16b. PWM.

5.3 SlowTick

V knihovně slowtick je k nalezení pouze funkci, která bliká LED diodou na řídící desce a automat volající funkce ElevatorTick, LEDObserverTick, ParamStoreTick, volání té jediné funkce a v neposlední řadě nám zavolá funkci restartující watchdog. ElevatorTick přenese vykonávání programu do knihoven s názvem elevator_main, LEDObserverTick do knihoven led_observer a ParamStoreTick do knihoven které jsem opět netvořil a to do paramstore. A jako poslední je zde volání funkce starající se o PID regulátor. SlowTick se provede jednou za 10 taktů TimeTicks.

5.3.1 Elevator main

Zde se ukrývá automat starající se o chod výtahu jako celku a s tím spojené funkce.
V hlavičkovém souboru elevátor_main najdeme výčtové typy em_dir, kde jsou všechny směry, kam může výtah jet, a to konkrétně nahoru, dolu a je zde přidán směr stop, ten nám slouží jako příznak, že výtah nikam nejede. Dále pak výčtový typ em_level, v němž jsou vyčteny všechny patra. Jako poslední výčtový typ je ten, ve kterém jsou vypsány stavy automatu. Následuje struktura kontextu automatu pro řízení výtahu, zde najdeme stavy automatu a různé potřebné proměnné, potřebné k předávání do jiných knihoven.
Dále pak tu najdeme konstruktory funkci spojených s ovládáním, jako například ActualElevPos, která navrátí aktuální polohu výtahu. Dále funkce, která navrátí ano nebo ne a to podle toho jestli výtah stojí. Funkce na přepočtení z typu em_level na pozici je nutnosti, stejně jako funkce, která zjistí, jestli jsem poblíž nějakého patra. Nakonec tu jsou funkce spojené s automatem.
Na začátku knihovny elevator_main.c jsou definovány konstanty pozic jednotlivých pater. Jako první funkci zde mám ElevatorStopped ta má návratovou hodnotu typu bool a zjistí, jestli jsme zastavili výtah. Tady je to řešené zjištěním stavu automatu

elevator_main, konkrétně se zeptá, jestli je ve stavu STOP, do tohoto stavu se dostane, pouze pokud motor stojí, ale blíže se tomu budu věnovat v popisu automatu.
Další funkce se nazývá actualElevPos, která nám vrátí aktuální polohu výtahu. Zde je rozdíl v tom jestli program pouštíme na počítači, tedy jako simulaci nebo přímo na desce. Proto je tu úsek ifdef MSVC, což umožní rozlišit, kde program běží, pokud běží na PC, tato funkce není potřeba, protože aktuální pozici obstarává simulátor. Ale běží-li na řídicí jednotce, aktuální pozice se vyčítá z registru timer/counter. Který je nastavený na counter pro inkrementální čidlo.
Funkce ElevatorOnLevel zjistí, jestli jsme už v těsné blízkosti nějakého patra, tato blízkost je daná proměnou k1, jež je definovaná na začátku této knihovny. Jednoduše se porovná aktuální pozice s pozicemi všech pater. Je-li tomu tak, platí to pouze pro jedno patro, můžeme tedy do ukazatele, který je v parametru funkce zapsat o jaké patro se jedná. Funkce je typu bool tudíž nám vrátí ano nebo ne. Konstantu k1 mám nastavenou na 5, což je dost přesné, jsou-li rozestupy mezi patry minimálně 10000. S touto funkcí je spojená další, která má zjistit, jeslí je v okolí nějaké patro. Tato funkce se nazývá ElevatorNearbyLevel a jediný rozdíl od předešlé je, že tato funkce má konstantu podstatně větší a je to z důvodu, abychom měli prostor na brzdění.
Poslední funkce levelToPosv jednoduše vrátí pozici patra, které jsme zadali do parametru. Akorát předělá typ em_level na pozici.
A teď už samotný automat. Popis začíná zrovna od stavu em_state_stop, protože ve stavu finished se nic neprovádí a ve stavu restart se pouze skočí do stavu stop. Tady začínáme podmínkou, pokud nemůžeme začít, automat skočí znovu na začátek vykonávání stavu stop. To je zajištěno podmínkou tázající se na směr výtahu (defaultně je nastaven na pohyb nahoru) a zároveň na to že jsme nepřejeli maximální hranice pro daný směr. Jestli můžeme začít, pak se stav automatu DCmotor posune na stav stop. A do proměnné lTimer se uloží aktuální hodnota čítače tiků zvýšená o konstantu. Toto je tu proto, aby výtah zastavil v patře a chvíli tam zůstal, pokud je další požadavek na vykonávání pohybu. Zůstane stát, dokud nepřeteče hodnota čítače hodnotu uloženou v lTimer. Pokud je ve frontě uložený požadavek na řízení automatu pomocí externích vstupů, skočíme s automatem do stavu em_state_automan. Následuje kontrola sepnutí tlačítka Stop, a pokud se vrátí false, tak vybereme z fronty raquest. Tento request vyhodnotíme a převedeme do typu patra, tím je zjištěno, do kterého patra máme jet. Následuje rozhodování o směru. A skočíme do stavu em_state_go. Hned na začátku se nastává chtěná hodnota rychlosti na 100. Následuje podmínka pro zjištění, jestli nepřišel request Stop, nebo jestli výtah nepřejel hranice. Pokud je jedna z těchto podmínek platná tak se nuluje požadovaná rychlost a skočí se do stavu em_state_stopping. Není-li tomu tak, program může testovat, jestli je kabina poblíž nějakého patra, je-li a patro je ve frontě příkazu, tak i když ze začátku pohybu nebyla jako koncová poloha toto patro, zastavíme v něm. Tedy je naleznut breakPosition funkcí z knihovny DCmotor a začne brzdit, pokud tuto hranici přejedeme, nastavíme chtěnou rychlost na 0, do proměnné aktuálního patra se nastaví toto patro, smaže toto patro z fronty, a skočíme do stavu em_steate_stopping. Ve stavu stopping se pouze čeká, než výtah zastaví. Dále tu je stav em_state_automan, do kterého lze skočit pouze ze stavu stop, to neznamená, že signál pro řízení pomocí externích vstupů musí přijít právě tehdy, když je automat ve stavu

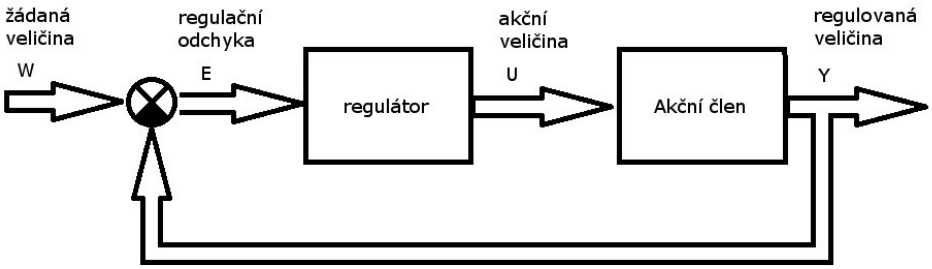
stop, ale nejprve se vykoná předchozí operace, tudíž dojedeme do nejbližšího chtěného patra, tím se automat dostane do stavu stop a z něj už do stavu automan. Vraťme se do stavu auto/man, tady se rozhodne o směru jízdy pomocí dalšího externího vstupu a to ext_dir. Podle směru se skočí do automatu em_state_automan_up nebo em_state_automan_down. V těchto stavech se pouze nastaví chtěná rychlost podle externího analogového vstupu mot_speed. A kontrolují se hranice a zmačknutí tlačítka stop.

5.3.2 Led observer

Tato část programu má za úkol starat se o rozsvěcení ledek.
V hlavičkovém souboru je pouze deklarace automatu a inicializace a finalizace. A v knihovně *.c najdeme Automat starající se o led znázornění akorát nastavuje digitální výstupy podle toho, ve kterém patře je kabina, nastaví na příslušný index digitálního výstupu true.

5.3.3 PID regulátor

V této podkapitole je nejprve popsaná funkce diskrétního PID regulátoru a následně jak byla řešena programově.
PID regulátor je složen z proporcionální, integrační a derivační složky. Proporcionální složka je potřebná k hrubé a rychlé regulaci, tedy pro přiblížení se chtěné hodnoty. Integrační složka je tu pro přesné doregulování a derivační složka kvůli překmitům, protože jak víme, derivace působí proti změně regulační odchylky. Toto se týká samostatného regulátoru.



Obrázek 10: soustav se zpětnovazebnou regulací

Z obrázku je jasné, že regulační odchylka (11), ta je vstupem do regulátoru. Výstupní veličinou regulátoru je akční veličina, která pokračuje do akčního členu. Něco se vykoná a vznikne regulovaná veličina. A ta se zpět zase odečte od žádané veličiny, což nám opět dá regulační odchylku.
Základní PID regulátor má tři veličiny, které reprezentují jednotlivé složky. Nejjednodušší matematický model je vyjádřen (12).

$$E = W - Y \tag{11}$$

$$\boldsymbol{U_{PID} = P * e + SumI + D * \Delta e} \tag{12}$$

Kde:

$$\boldsymbol{SumI = SumI + I * e} \tag{13}$$

Takto už by to šlo řešit, ovšem je zbytečné zapojovat integrační a derivační složku, když ze začátku regulace hrubé přiblížení k požadované hodnotě zajišťuje proporcionální složka. Můžou sem byt vloženy další dvě proměnné, které, když absolutní hodnota

regulační odchylky klesne pod tyto hodnoty, způsobí připnutí dané složky. Tímto způsobem jsou vytvořena pásma, kde budou jednotlivé složky aktivní. Tyto pásma se volí obvykle tak, že derivační složka je aktivní pouze tehdy, je-li odchylka hodně malá, aby nedocházelo ke zbytečným kmitům ke konci regulace, naopak integrační složka musí být aktivní ve větší části rozsahu, kvůli nedokonalé regulaci pouze pomocí proporcionální složky.

Programová část této problematiky je velice jednoduchá, je zde jedna funkce a jeden automat. Funkce má za úkol spočítat odchylku a Δe, tato funkce je volaná v automatu. Automat je zde z důvodu časovému zpoždění, je zde stav wait, kde se pouze čeká. Pak stav auto, zde se jako první zavolá funkce pro update regulační odchylky, zjistí se, jestli už jsou potřeba ostatní složky regulátoru, vypočte se SumI a provede se výpočet podle vzorce (13).

5.4 Knihovny bez automatu

5.4.1 Fronta

V této knihovně se program řízení stará o frontu příkazů. Má k dispozici pole o deseti prvcích, kam zapisujeme požadavky na volání výtahu. Jsou jenom čtyři patra, tlačítko stop, auto/man, externí dir, tudíž ve frontě by mohlo být jenom sedm prvků. V automatu starající se o zmačknuté tlačítka, nám volá funkci pro zápis do fronty. Zapíše pouze tehdy, pokud již není ten samí request již ve frontě. Dále tu máme funkce konvertující požadavek z formátu request na formát level a obráceně. Dotaz na zjištění je-li request v parametru ve frontě, můžeme provést přes funkci IsInFrontaAktualReq. Důležitá funkce je SortArray, která potřebuje směr pohybu a podle toho srovná pole tak, aby první prvek byl, pokud jedeme nahoru, ten nenižší následující a pokud jedeme dolu, ten největší následující. Další funkce na výběr toho nejvyššího nebo nejnižšího položeného requestu nemohou chybět. Nakonec je funkce, která odebere požadavek z fronty.

5.4.2 Gvariables

Zde jsou všechny globální proměnné celé aplikace, zde se volají všechny důležité knihovny, ve kterých je nějaká struktura, která je použita i někde jinde než je definovaná. Deklaruji se zde struktury automatů od komunikace přes IO, paměť až po deklaraci struktury z elevátor main. Prostě všechny struktury co se nazivají *_CTX. Dělá se to v jedné knihovně proto, abychom nemuseli volat v každé knihovně všechny ostatní. Jednoduše se zavolá pouze tato.

5.4.3 Inicializace

Tato knihovna se volá jako úplně první a zavolá funkce inicializace od všech knihoven, které tuto funkci mají, měli by ji mít všechny. Například se volá inicializace od knihovny buton observer, v níž se inicializuje automat. To nás odkáže na knihovnu statemachine, kde je přesně definováno, jak má stavový automat vypadat. Je tu také funkce na úvodní restartování automatu, aby nezačal v nějakém nechtěném stavu.

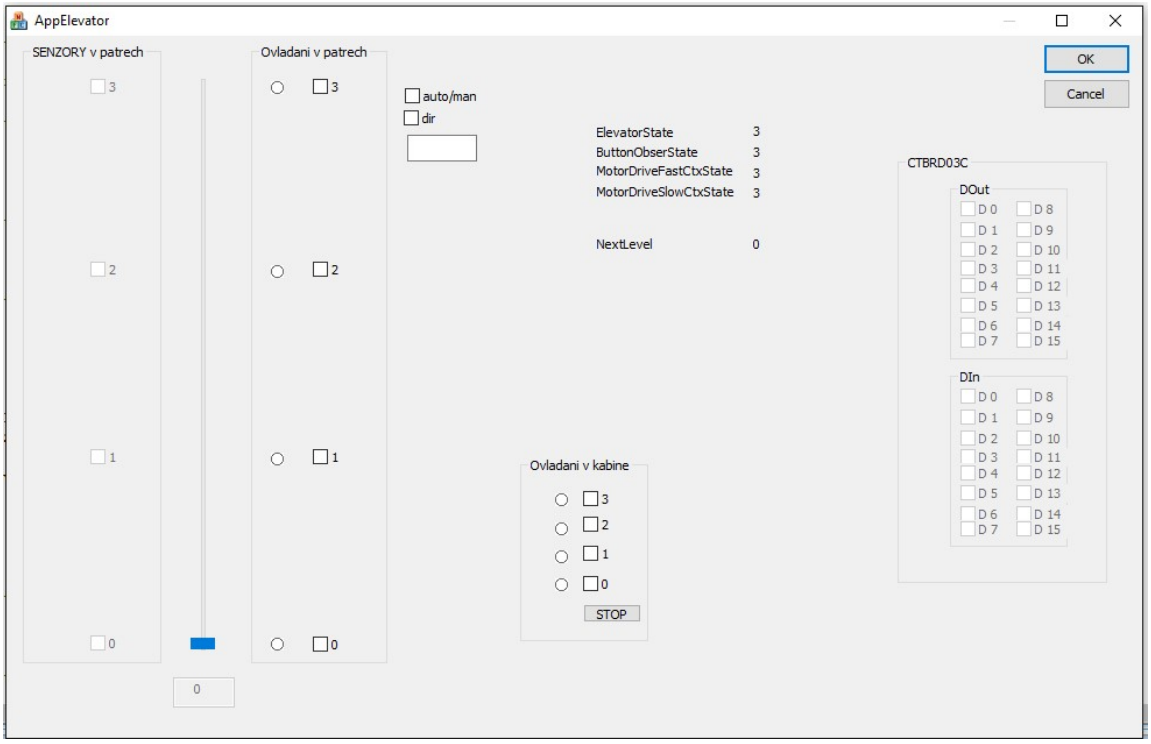
5.4.4 Finalizace

Tato funkce se volá při ukončení programu, zavolají se funkce finalizace od všech knihoven, podobně jak je to u knihovny inicializace. V jednotlivých funkcích, které se volají touto knihovnou, se ukončují jednotlivé automaty.

6 Simulace

Simulace je zde pouze pro ověření funkčnosti programu na PC, abychom nemuseli každou změnu nahrávat do systému a zkoušet ji v provozu. Pracuje s celým programem kromě HW knihoven a funkcí, které HW nezbytně potřebují.

Nejprve je zde popsáno, jak vypadá simulační okno. Najdeme zde vysoký slider, znázorňující kabinu výtahu, je ovládán knihovnou promněnou actualposition. Nalevo od něj jsou checkBoxy znázorňující digitální vstupy senzoru patra. Jsou nastaveny jako disable, aby nešli uživatelem zmáčknout, mění se pouze na základě programu. Napravo od slideru jsou v každém patře pictureControl, ty znázorňují, kam míří výtah. A za nimi je ovládání výtahu v jednotlivých patrech. Je zde ještě ovládání v kabině, kde kromě tlačítek jednotlivých pater je ještě jedno a to tlačítko Stop. V pravé části okna najdeme všechny vstupy a výstupy a stavy všech automatů.



Obrázek 11: Ukázka simulačního okna

Na začátku hlavičkové knihovny jsou definovány konstanty na jaké pozici jsou patra, kolik je pater, vzdálenost mezi patry a jak kolik je maximum slideru znázorňující kabinu výtahu, což je výpočet počtu pater krát vzdálenost mezi patry. Máme zde jednu strukturu SIM_ELEVATOR_CTX a v ní pouze proměnnou ElevatorPos typu integer. Musí být ve struktuře, protože později budu předávat odkaz na strukturu, tomuto programování se říká objektové programování a jazyk C, ve kterém pracuji, objektově orientovaný není. Opět jsou zde názvy funkci, definované v knihovně *.c

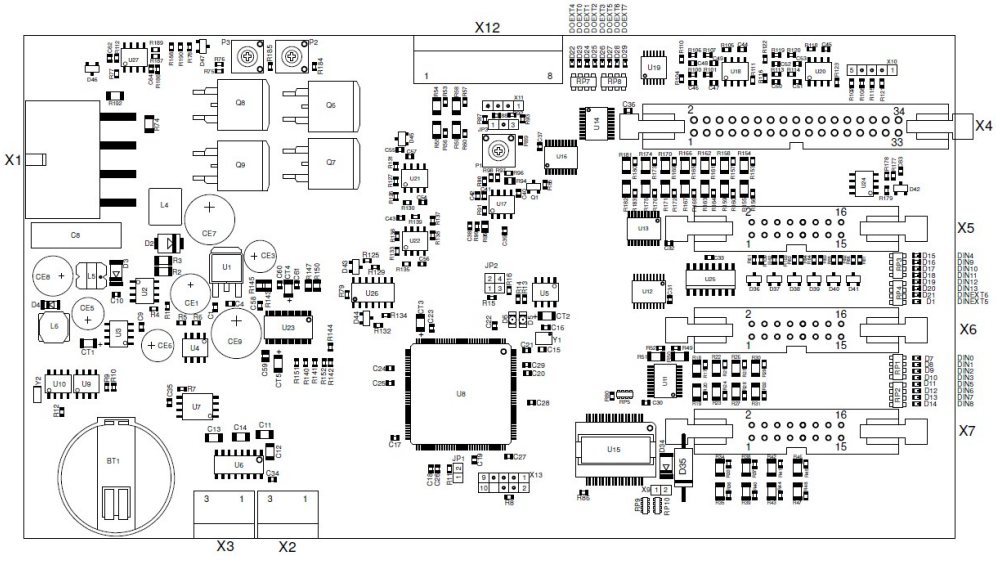
V tomto automatu se nejprve zjistí, kde se nachází kabina a podle toho se nastaví maska vstupů. Nejprve se zjisti index bytu podle patra a je-li kabina v patře tak se nastaví příslušný vstup na true. Pro zjišťování indexu a hodnot vstupů a výstupů se volají funkce z knihovny iosomulator.cpp Toto se provede pro všechny patra a tím získáme masku vstupů. Poté se simulují analogové vstupy, nastaví se relativní rychlost výtahu.

Dále pak vypočet nové a aktuální pozice. Nakonec tohoto automatu je zadání nové pozice pro automat ovládající motor. Ve funkcích inicializace a finalizace neprovádíme nic.
Je tu ještě jedna knihovna související se simulátorem, jmenuje se iosimulator. V této knihovně jsou funkce a makra volaná, při potřebě simulovaní vstupů a výstupů a to jak analogový tak digitálních. Tuto knihovnu jsem také netvořil já, ale pouze jsem využil již hotové simulační prostředky.

7 Návrh a osazení plošného spoje

Návrh plošného spoje probíhal v prostředí Eagle. Protože jsem měl přístup k firemní licenci, mohl jsem navrhnout plošný spoj o rozměrech větších než 100mm na 100mm, které jsou povoleny pro freeware verzi. plošný spoj řídicí jednotky má rozměry 100mm na 200mm.

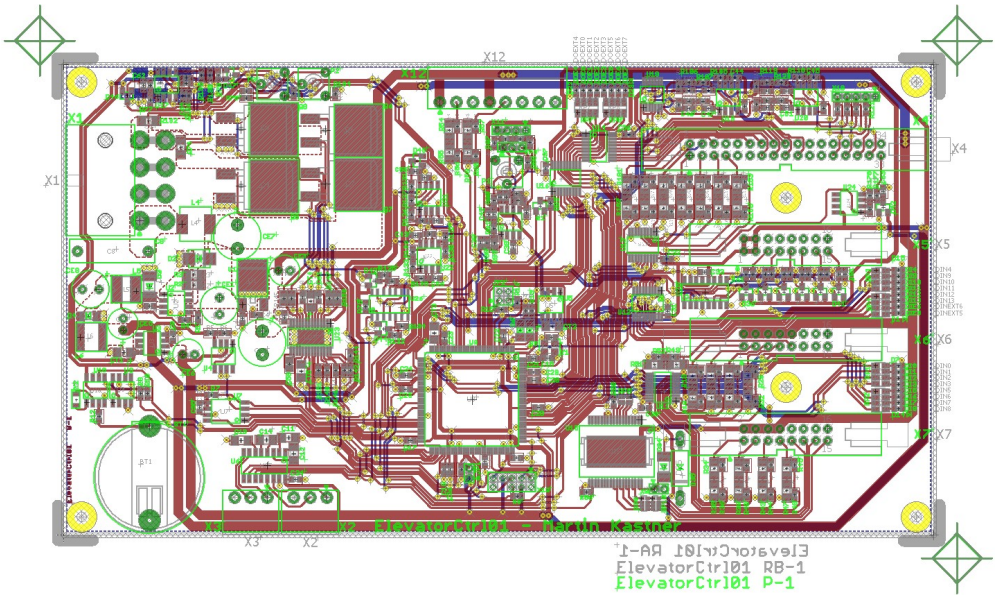
Prvním krokem bylo použití firemního skriptu na převedení netlistu z programu Orcad, kde probíhal návrh schémat do programu Eagle. Jena z nevýhod tvorby schémat v jiném programu než návrh DPS bylo, že se nenakopírovaly součástky ze schématu přímo do návrhu DPS, z čehož plyne, že museli být vloženy všechny součástky ručně. Naštěstí v programu Eagle jsou prostředky na určité kontroly, takže jsem věděl, že byly všechny součástky vloženy. Po vložení všech součástek, mohli být rozdzděleny podle jejich účelu. Například pro digitální výstupy byly shromážděny všechny odpory pro děliče proudů i napětí, všechny vstupní oddělovače, diody a všechny ostatní komponenty, které jsou potřeba pro tuto část, do jednoho místa. Jednotlivé úseky byly rozděleny zhruba podle listů ve schématu. Poté mohlo přijít na řady samotné rozmísťování. Jako byly umístěny konektory. K přívodnímu konektoru jsou umístěny součástky související s napájením, jako například vstupní filtr nebo zdroje různých napětí. U konektoru digitálního vstupu jsou umístěny děliče a oddělovače, které upraví vstupní signál na 3,3V digitální logiku. Potom byl umístěn zhruba doprostřed procesor, aby trasy jednotlivých signálů nebyly moc složité.



Obrázek 12: rozložení součástek na DPS řídicího systému

Když byly součástky rozmístěny, pozornost mohla být věnována natahování jednotlivých spojů. Opět bylo takticky začato s natahováním napájecích cest, které jsou většinou širší než ostatní. Po natažení tras pro 24V, 12V, 5V, 3,3V i 1,2V, se mohlo začít s tahání signálů, které nepotřebují být tak silné. Začalo se od procesoru, protože tam vedou skoro všechny trasy, a vybrána bylo a součástka, ze které jde co nejvíce signálů do procesoru, aby se trasy co nejméně křížily. Například jedny z prvních byly připojeny oddělovače vstupních a budiče výstupních signálů. Tímto způsobem byly vytvořeny všechny cesty. Na druhé straně desky je rozlitá zem, ovšem jenom tam kde, je to možné. Nakonec je

třeba říct, že s postupem i rozvržením mi bylo porazeno od firemního odborníka na tvorbu DPS.



Obrázek 13: konečný návrh DPS řídicího systému

Osazení proběhlo mojí rukou, protože se jednalo o jednotlivou výrobu a ne o sériovou výrobu, na kterou je automatizované osazování vhodné. Jako první byly osazeny SMD součástky a to nejprve čipy a poté pasivních součástky. U osazování vícenožičkových čipů nastal problém a to zvláště u procesoru, kde je 117 nožiček. 'Tento problém byl vyřešen problém kolegou, který ukázal, jak se osazuje pomocí horkého vzduchu a pájecí pasty, což je cín s kalafunou v pastě. Normálním způsobem byl čip přichycen jednou či více nožičkami k DPS, dál byla nanesena tenkou vrstva pájecí pasty na ostatní nožičky. Jednotlivé nožičky nebyly řešeny, klidně se mohla nanést jednolitá vrstva přes všechny pady, poté se na nožičky fouknul žhavý vzduch, pod kterým se cín roztavil a díky kalafuně zalezl mezi pad a nožičku a nikam jinam, takže pokud byla vrstva pasty správně tenká, tak se nespojily sousední pady. Po osazení všech SMD čipů, byly osazeny pasivní SMD součástky, jako hlavně odpory a kondenzátory ale i indukčnosti. Následovali osadit THT součástky, většinou to byly elektrolytické kondenzátory. No a nakonec byly osazuny konektory.

Obrázek 14: kompletně osazená deska

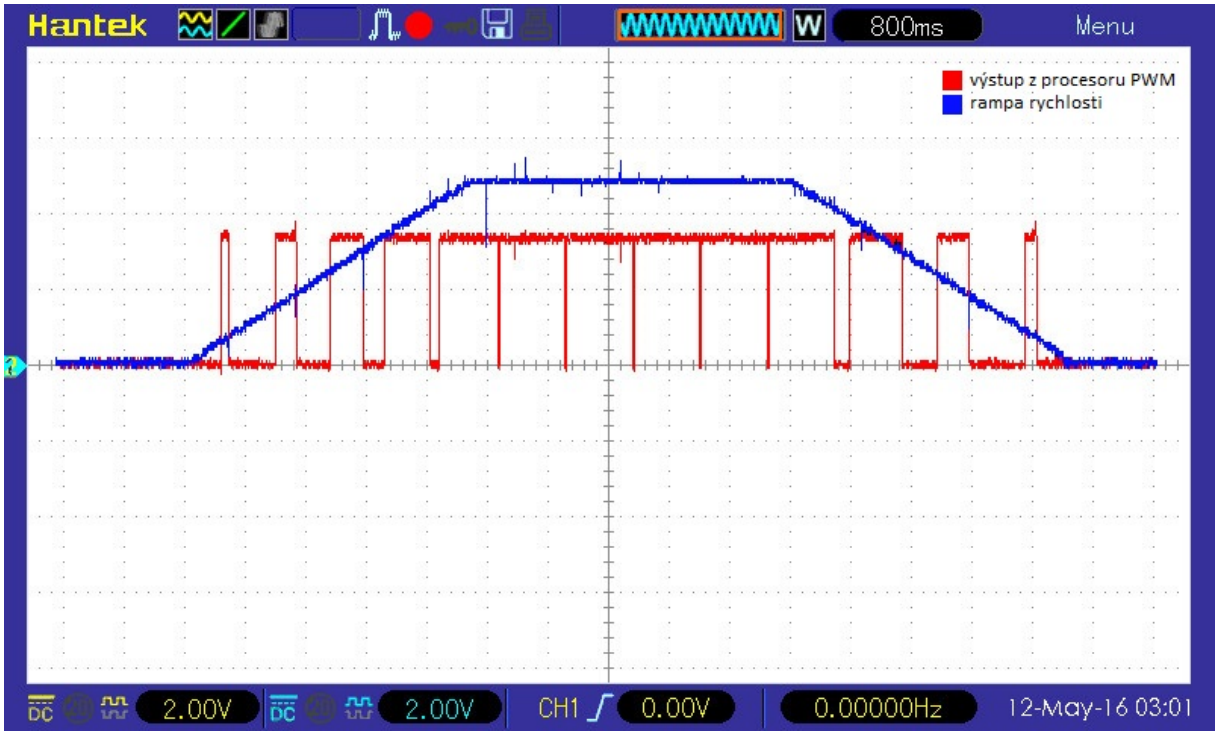
8 Oživení desky

S touto aktivitou mi pomohl ten sami kolega, ne snad že by oživoval on, ale dělal mi odborný dohled. Nejprve bylo proměřeno, jestli náhodou nebyl vytvořen zkrat mezi jednotlivými napájeními nebo mezi napájením a zemí. Po zjištění, že tady v tomto problém nenastal, deska byla připojena ke zdroji a postupně bylo zvedáno napětí, přitom byl kontrolován vstupní proud. Tato činnost probíhala samozřejmě se zapnutým omezením proudu. Po úspěšném spuštění byl nahrát do systému BoatLoader, což je v podstatě zavaděč pro další program. Tento boatloader byl předem připravený, je to opět software s firmy ConTeK, pouze upraveny pro tuto konkrétní desku. Slouží nejenom jako zavaděč programu, ale i jako ochrana před nahráním programu co na tuto desku nepatří. Problém by mohlo dělat nahrání programu, kde je jiné mapování vstupů a výstupů, a to zejména těch ochranných, například enable výstupních budiču. Pomocí tohoto programu mohl byt nahrán program IOtest, čímž mohli být vyzkoušeny vstupy a výstupy analogové i digitální. Po ukončení zkoušení, jsem si omylem spojil +24V s 5V vstupem, což mělo za následek vypálení vstupního oddělovač, který nesplnil svoji funkci a poslal dál k procesoru větší napětí než 3,3V, oddělovač dokáže omezit napětí od měkkého zdroje, který, když se zatíží, zmenší své napětí, to ovšem nebyl bohužel můj případ. Protože se do procesoru dostalo mnohem větší napětí, než bylo schopno akceptovat, musel jsem procesor vyměnit. Tato nehoda mě nestála nejenom další procesor, ale i další tři dny práce se zpravováním škod a zjišťováním co jsem dále zničil. Po této opravě škod muselo být pokračováno do začátku této kapitoly. Při oživování analogových vstupů, konkrétně měření rychlosti, bylo zjištěno, že deska byla osazena špatným čipem spínače. Čip U26 měl být analogový spínač spínaný digitálním vstupem, ovšem osazena součastka byla digitální spínač, což mělo za následek úpravu analogové hodnoty na digitální. Projevilo se to tak, že do přejetí hranice 0,8V čip se choval jako, že na vstupu byla logická 0, mezi hranicemi 0,8V až 2,4V byl v nedefinovaném stavu a nad 2,4V se choval jako by na vstupu byla log. 1. Po hardwarovém oživení vstupů a výstupů se mohlo pokročit k oživování programu. Tady opět muselo být upraveno mapování vstupů a výstupů, aby seděly vstupy a výstupy v hardware se vstupy a vystupy v software. Nakonec muselo být oživeno inkrementální čidlo pozice kabiny. To je připojené k Timer/counter o funkci Quadrature Decoder [1]. Je to nastavení, které přesně odpovídá čítání impulzů z inkrementálního čítače, do registru se tedy přičte nebo odečte jednička, právě tehdy když přijdou na vstupy TIOA a TIOB impulzy, přičtení nebo odečtení závisí na pořadí impulzů. Při oživování vstupů a výstupů nastal na problém, s počtem PWM výstupů procesoru. Špatně byl mnou pochopen manuál, bylo předpokládáno, že procesor má šest samostatných PWM kanálů, má jich pouze čtyři, ale jeden kanál je na více pinech. Je to z důvodu častého využití PWM a protože každý pin může mít různé funkce, tak by se mohlo stát, že si zabereme pin pro jinou funkci. Proto má čtyři kanály a každý kanál je na čtyřech pinech. Potřeboval jsem čtyři výstupy pro komunikaci s měřící kartou počítače a další dva na ovládání čipu motoru. Vymyšlena byla možnost udělat PWM kanál z timer/counter. To se dá udělat jednoduše, protože PWM kanál je digitální výstup řízený časovačem, tudíž je v logické 1 po dobu, než registr přeteče určitou hodnotu. Protože má rychlou periodu, po změření střední hodnoty signálu, získáme analogovou hodnotu.

9 Naměřené výsledky

Prvním pozitivním výsledkem je funkčnost ovládání výtahu pomocí vstupů a výstupů. Model vykonává pohyby dle vstupních požadavků do správného patra. Protože je možné měnit rozjezdové a dojezdové rampy, docílíme i změny dynamických vlastností celého modelu. Konkrétně zejména změny proudu při rozjezdu a zastavení výtahu se zátěží. Zajímavé bude nastavovat tyto vlastnosti v laboratorních úlohách. Proto je možné nastavení těchto i jiných parametrů řídící jednotky provádět připojeným počítačem PC přes vstupně výstupní komunikace, bez nutnosti přehrání celého aplikačního programu výtahu.

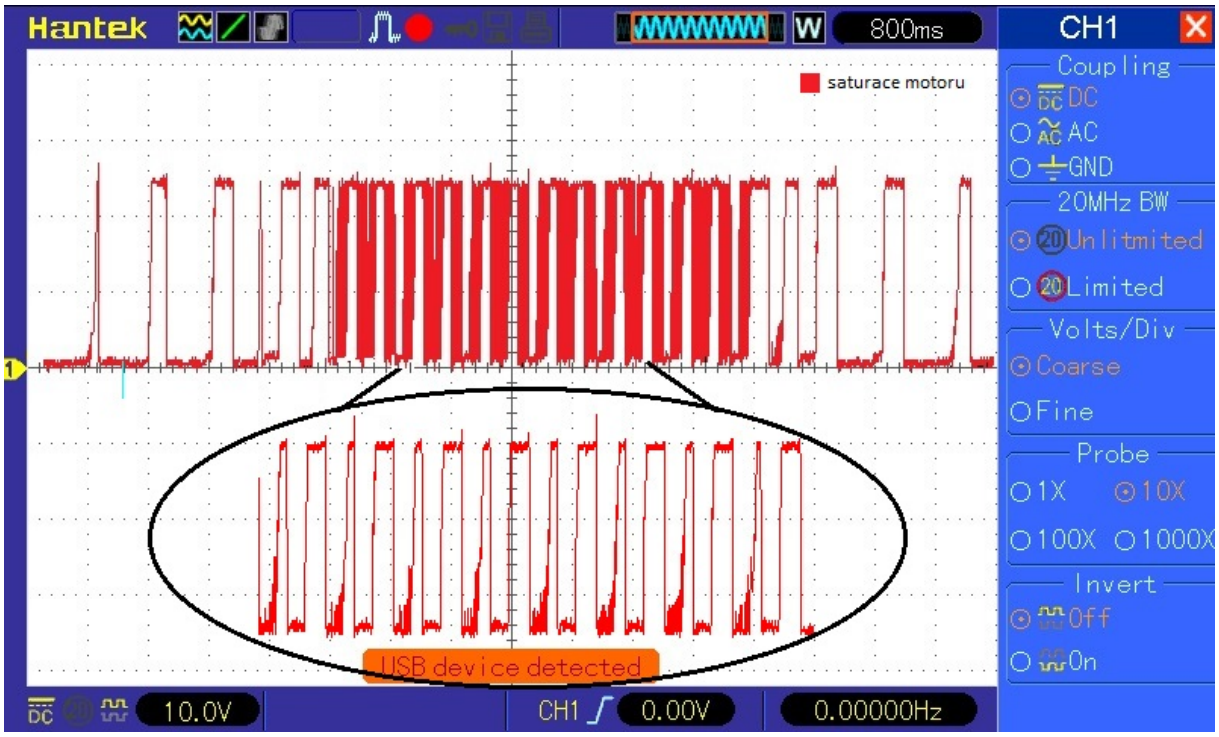
Pro další výsledky jsem potřeboval osciloskop.



Obrázek 15: ukázka PWM výstupu a rampa rychlosti

Na tomto obrázku je vyobrazen červeně signál z PWM výstupu od procesoru do čipu pro ovládání motoru. A také je zde vidět modrá rampa zrychlování i zpomalování požadované rychlosti motoru. Zrychlovací rampa je jasně vidět na levé půlce grafu. Rozšiřující pulzy červeného grafy jsou známkou zrychlování. Bohužel červený signál PWM je zkreslený, protože PWM perioda je mnohonásobně větší než perioda vzorkování osciloskopu. Pěkný průběh je však dán skutečností, že osciloskop při nižším PWM spíše zachytává logickou 0, naopak s vysokým PWM zachytává spíše logickou 1. Modrý graf jde do maxima 5V, protože je měřen na analogovém výstupu jednotky za zesilovačem.

Na dalším obrázku je vyobrazen signál jsoucí z tranzistorového můstku do stejnosměrného motoru. Protože je H můstek napájen z 24V budou také pulzy této hodnoty.

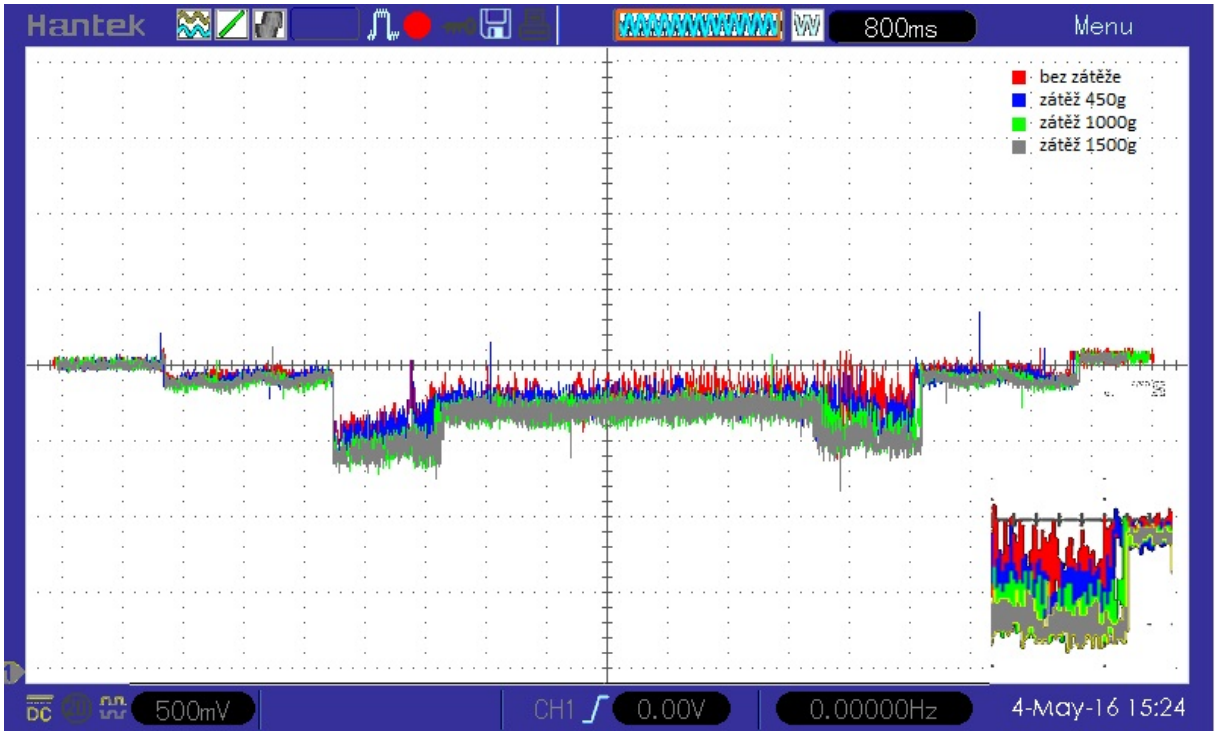


Obrázek 16: saturace motoru

Takto jsem změřil přejezd z nultého patra do prvního. I přesto že napájíme můstek z 24V, střední hodnota napětí na motoru bude odpovídat maximálně 12V, protože maximální hodnotu PWM nastavujeme na 50%.

Zvětšený průběh je zde uveden, protože je zde stejná chyba při vzorkování osciloskopu jako na předchozím grafu. Stejný průběh by poté měřen s větší vzorkovací frekvencí osciloskopu.

Jako další měření jsem zvolil měření proudu motoru při různém zatížení výtahu. Toto měření bylo prováděno pomocí analogového vstupu AIN2, který měření proud motoru na rezistoru ve vodiči B. Měření byla prováděna při pohybu vzhůru. Kabina výtahu se pohybovala mezi patry 1 a 2, abychom vlivem brzdných ramp nenajeli horní koncový snímač. Při najetí na koncový snímač, by výtah zastavil okamžitě a měření by bylo zkresleno. Toto měření bylo provedeno pokaždé s jinou zátěží. Červené grafy znázorňují jízdu s nulovou zátěží, modré křivky se zátěží 450g, zelené se zátěží 1000g a jako poslední se zátěží 1500g jsou grafy šedivé barvy.



Obrázek 17: grafy zatěžovacích charakteristik při stoupání

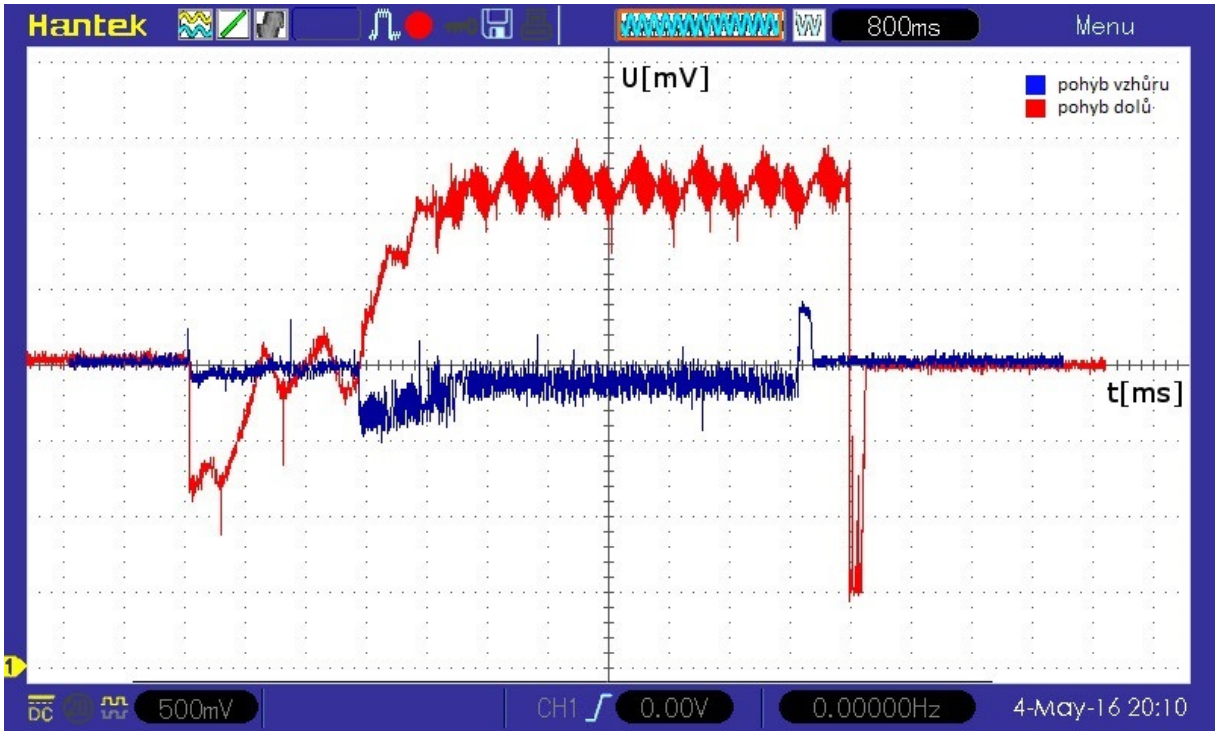
Výsledný graf je složen ze čtyř po sobě jdoucích samostatných měření.

Hned ze začátku musím vysvětlit, proč jsou hodnoty znázorněny inverzně. Je to z důvodu měření napětí na rezistoru, kterým prochází proud opačné polarity.

Z grafu je jasně vidět, že čím větší zátěž, tím větší teče proud do motoru a to po celou dobu pohybu. V pravém dolním rohu je zvětšenina všech grafů, pro jednoznačnější určení rozdílu proudu při zátěži.

Měření je zatíženo chybou způsobenou zpomalením otáček motoru při vetším zatížení motoru. Pokud by regulátor otáček držel požadované otáčky při všech zatížení, rozdíly v naměřených proudech při různých zátěžích by měly být znatelně větší.

Další graf tohoto měřícího bloku, bych rád ukázal, jak se výtah chová, lépe řečeno brzdí, při najetí na maximální hranice, které hlídají koncové spínače horní nebo dolní polohy.



Obrázek 18: ukázka brzdění po přejetí hranice

Opět se jedná o složený graf ze dvou samostatných měření pro pohyb vzhůru a dolů.

Modrý graf znázorňuje jízdu vzhůru a červený naopak klesání. Nepodařilo se mi nastavit polohu přesně tak, aby výtah najel na senzor patra v ten samý časový okamžik. Důležité je, že program řídicí jednotky hlídá tyto snímače a pokud se od nich objeví signál okamžitě vypne PWM vystup pro točení motoru. Je zde vidět jak proud překmitne do opačné polarity proudu při zastavení motoru. Motor se při rychlém zastavování začne chovat jako generátor energie.

Bohužel jsem v bakalářské práci nestihl vytvořit program pro měření pomocí měřící karty v počítači PC, nicméně vstupně výstupní komunikace s PC je oživená a zkontrolovaná její funkčnost. Vstupy jsou otestovány pomocí zdroje napětí a výstupy jsou změřeny multimetrem a osciloskopem. Určitě bude zajímavé vytvořit automatické testovací úlohy, kde se budou postupně přidávat otáčky a v ten samý okamžik měřit aktuální proud motoru.

Dále je možné doplnit regulátor otáček do řídící jednotky výtahu.

Motor je vybaven převodovkou s převodovým poměrem 1:312. Mechanický odpor motoru je na výstupu převodovky 312 krát zesílený. Při zatížení výtahu hmotností 1,5 kg, se motor vlivem tohoto mechanického odporu i bez přítomnosti mechanické brzdy neroztočí. Klidová brzda je z tohoto pohledu zbytečná.

Jako další bod zadání jsem se měl zamyslet a vytvořit zadání laboratorních cvičení pro úlohy s modelem výtahu. Jelikož nevím pro jaké studeny bude tato úloha určena, uvedu zde pouze náznaky, toho co by se dalo měřit, popřípadě způsob měření. Kdyby byla k dispozici měřící karta pro počítač PC, určitě by bylo dobré vytvoření programu pro měření, o němž jsem se zmiňoval v předcházejícím odstavci. Dále by se úloha dala rozšířit o měření zatěžovacích charakteristik při různém zatížení kabiny, a proměřovat rozjezdové a brzdné charakteristiky proudu motoru. Pokud by nebyla k dispozici vstupně výstupní komunikace s PC, směřoval bych zadání spíše k manuálnímu měření na modelu. Naměřit požadované průběhy osciloskopem a správně je okomentovat.

10 Závěr

V této práci se mi povedlo vytvořit funkční model manipulační plošiny pro laboratorní využití. Vyhověl jsem zde všem požadavkům zadání i požadavkům vedoucího práce, zejména pak byla důležitá vstupně výstupní komunikace v měřící kartou počítače PC, jež jsem samozřejmě vytvořil. Tato bakalářská práce ověřila mé znalosti i dovednosti v různých oborech činnosti. Jednalo se především o výběr vhodných komponent, vytvoření programového vybavení pro řízení modelu nebo vytvoření nové řídicí jednotky, což zahrnovalo návrh, tvorba i osazení DPS, zde jsem udělal pár chyb, ovšem žádné fatální, které bych nedokázal vyřešit. Model jsem úspěšně zkonstruoval, oživil a otestoval jeho funkčnost. Bohužel jsem nevytvořil úplné zadání pro laboratorní úlohu, nýbrž jsem jenom nastínil možnosti, jak by se moje práce dala využít v budoucnosti, ať už by se jednalo laboratorní využití, nebo pokračování a rozšíření mé práce, pomocí další bakalářské, diplomové nebo jakékoliv jiné práce.

Seznam použité literatury

[1] *11157F-ATARM-SAM4E16-SAM4E8* [online]. San Jose, California, USA: Atmel, 2015 [cit. 2016-05-16]. Dostupné z: http://www.atmel.com/images/atmel-11157-32-bit-cortex-m4-microcontroller-sam4e16-sam4e8_datasheet.pdf

[2] *SN75173 QUADRUPLE DIFFERENTIAL LINE RECEIVERS* [online]. Dallas, Texas, USA: Texas Instruments, 2015 [cit. 2016-05-16]. Dostupné z: http://www.ti.com.cn/cn/lit/ds/symlink/sn75173.pdf

[3] *74VHC541 Octal Buffer/Line Driver* [online]. San Jose, Kalifornie, USA: Fairchild Semiconductor, 2005 [cit. 2016-05-16]. Dostupné z: https://www.fairchildsemi.com/datasheets/74/74VHC541.pdf

[4] *VN808CM-E Octal channel high-side driver* [online]. Geneva, Switzerland: STMicroelectronics, 2013 [cit. 2016-05-16]. Dostupné z: http://www.farnell.com/datasheets/1815522.pdf

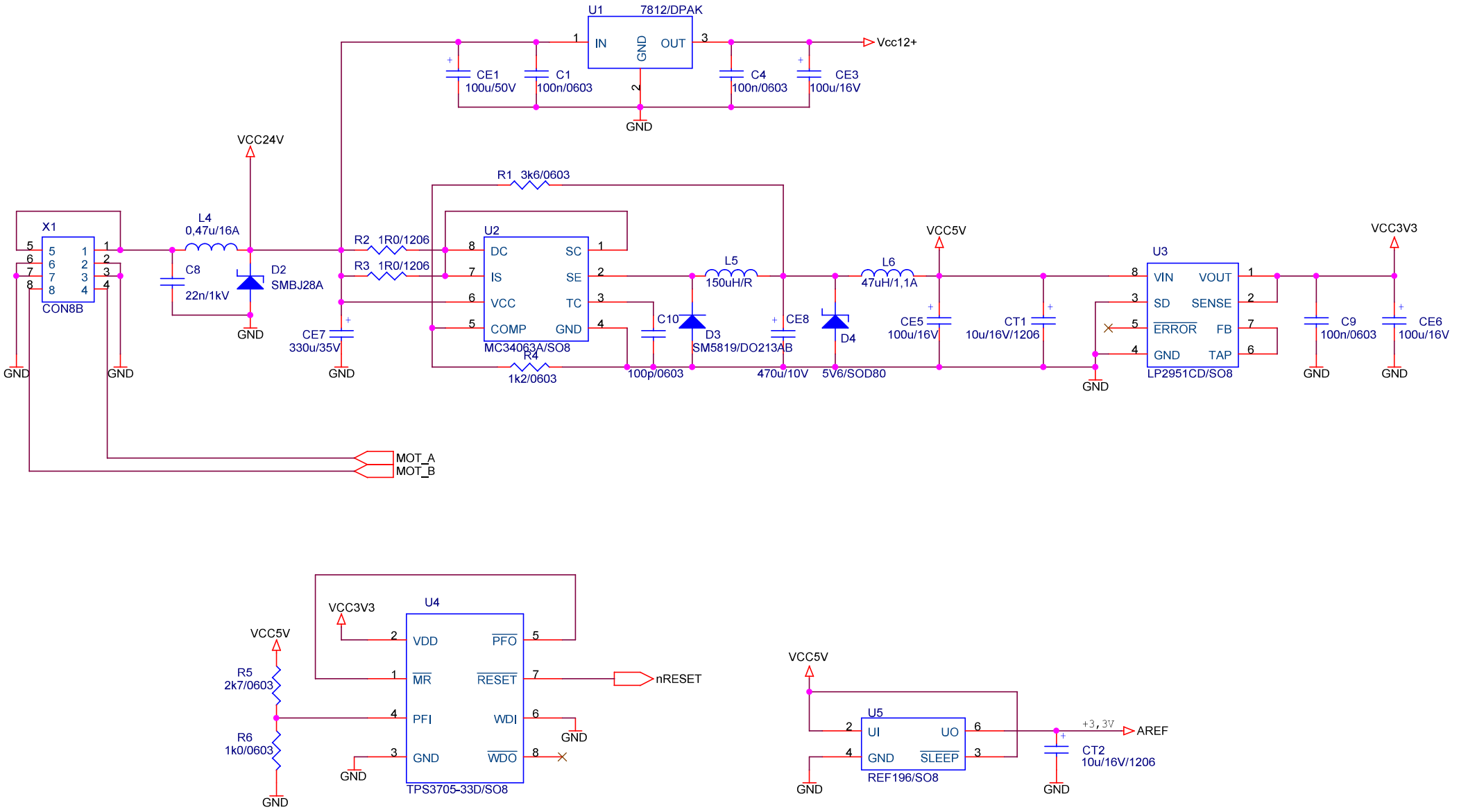
[5] *A3941Automotive Full Bridge MOSFET Driver* [online]. Worcester, Massachusetts, USA: Allegro, 2011 [cit. 2016-05-16]. Dostupné z: http://www.datasheetlib.com/datasheet/164230/a3941_allegro-microsystems.html

[6] *AN905 Brushed DC Motor Fundamentals* [online]. Worcester, Massachusetts, USA: Microchip Technology, 2004 [cit. 2016-05-16]. Dostupné z: http://ww1.microchip.com/downloads/en/AppNotes/00905B.pdf

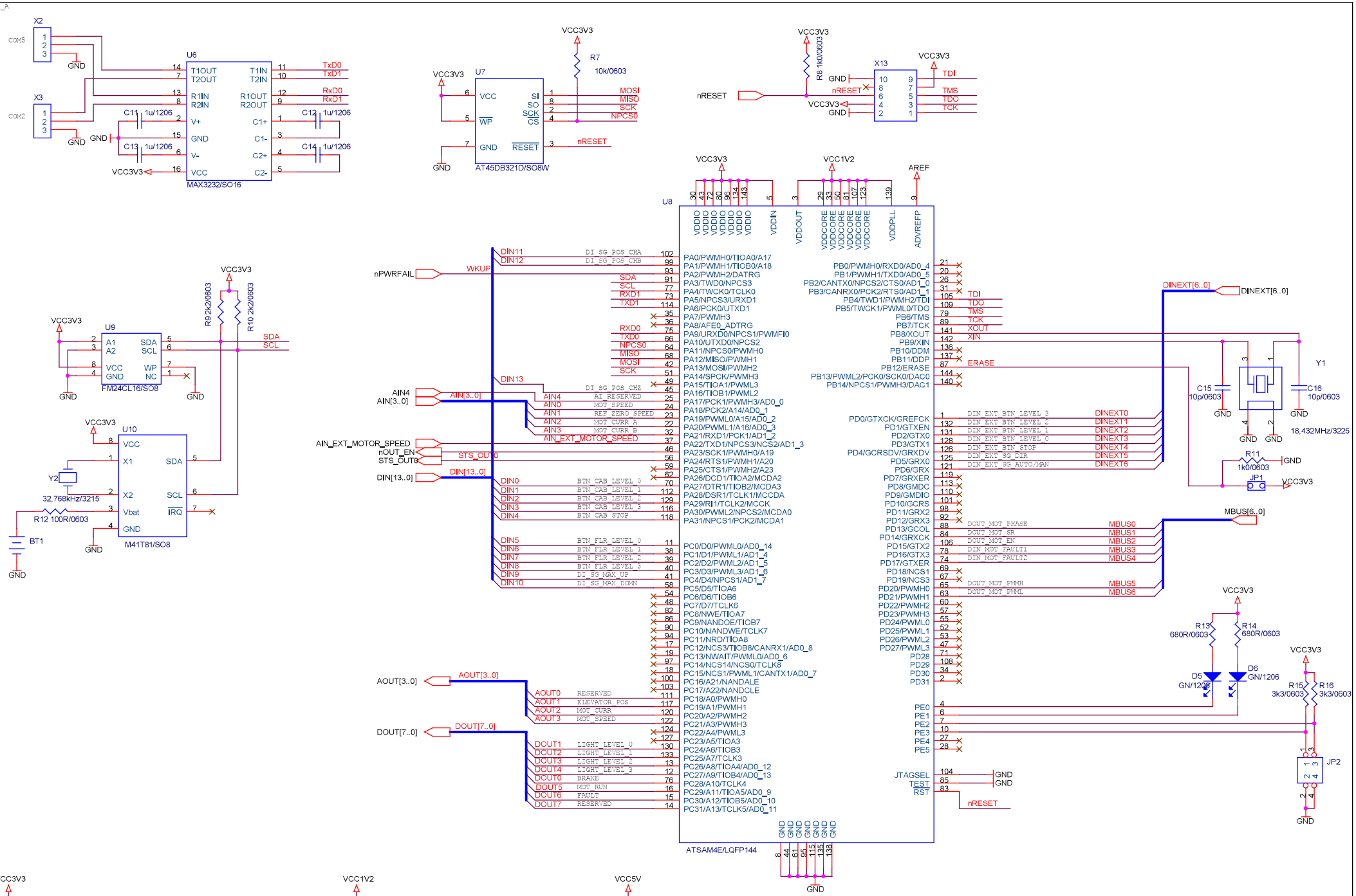
Přílohy

Obsah na CD

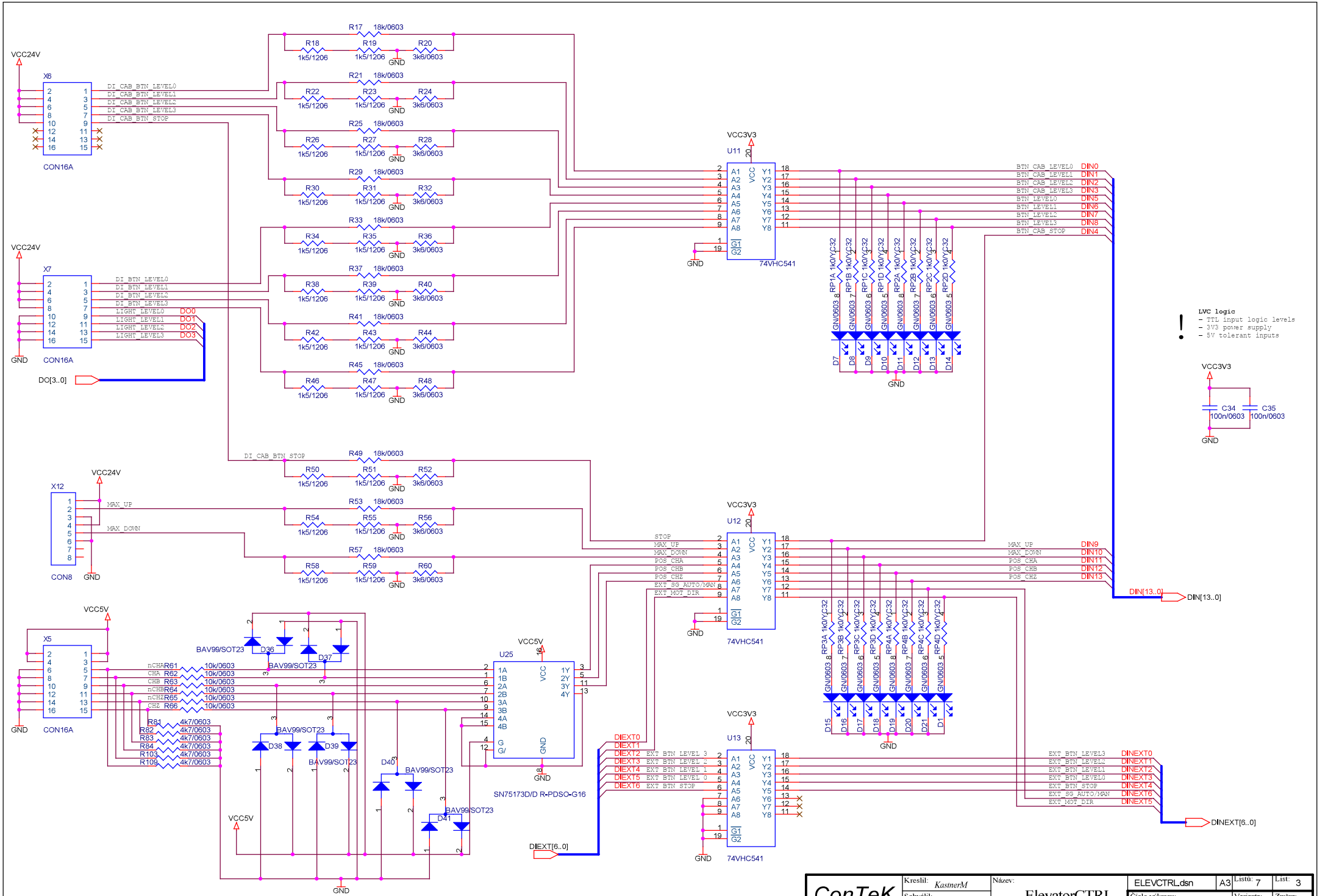
1. Schéma zapojení řídicí desky
2. Uživatelská příručka
3. Program simulace výtahu



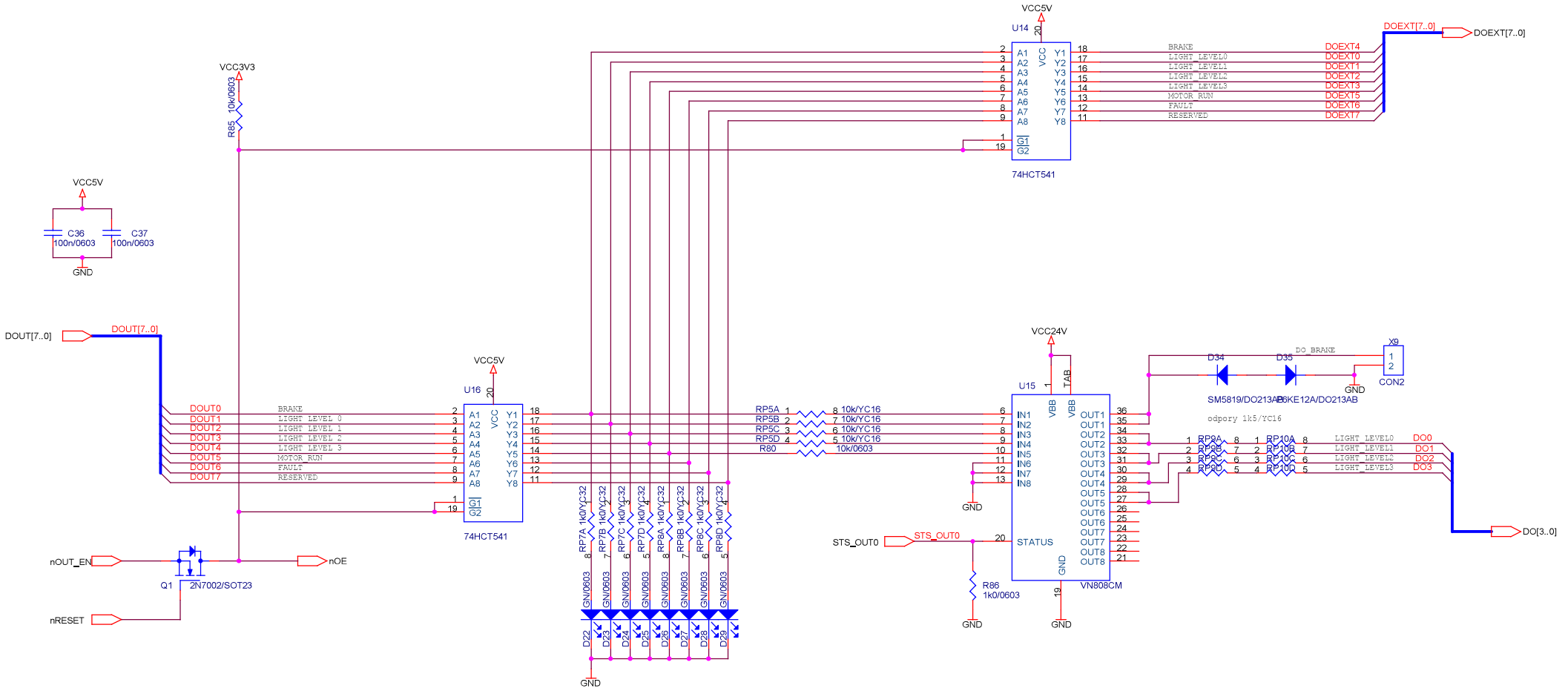
ConTeK industrial control systems	Kreslil: *KastnerM*	Název: ElevatorCTRL Power supply	ELEVCTRL.dsn	A4	Listů: 7	List: 1
	Schválil:		Číslo výkresu: CTS083		Varianta: 01	Změna: 01
	Datum: 25.11.2015					



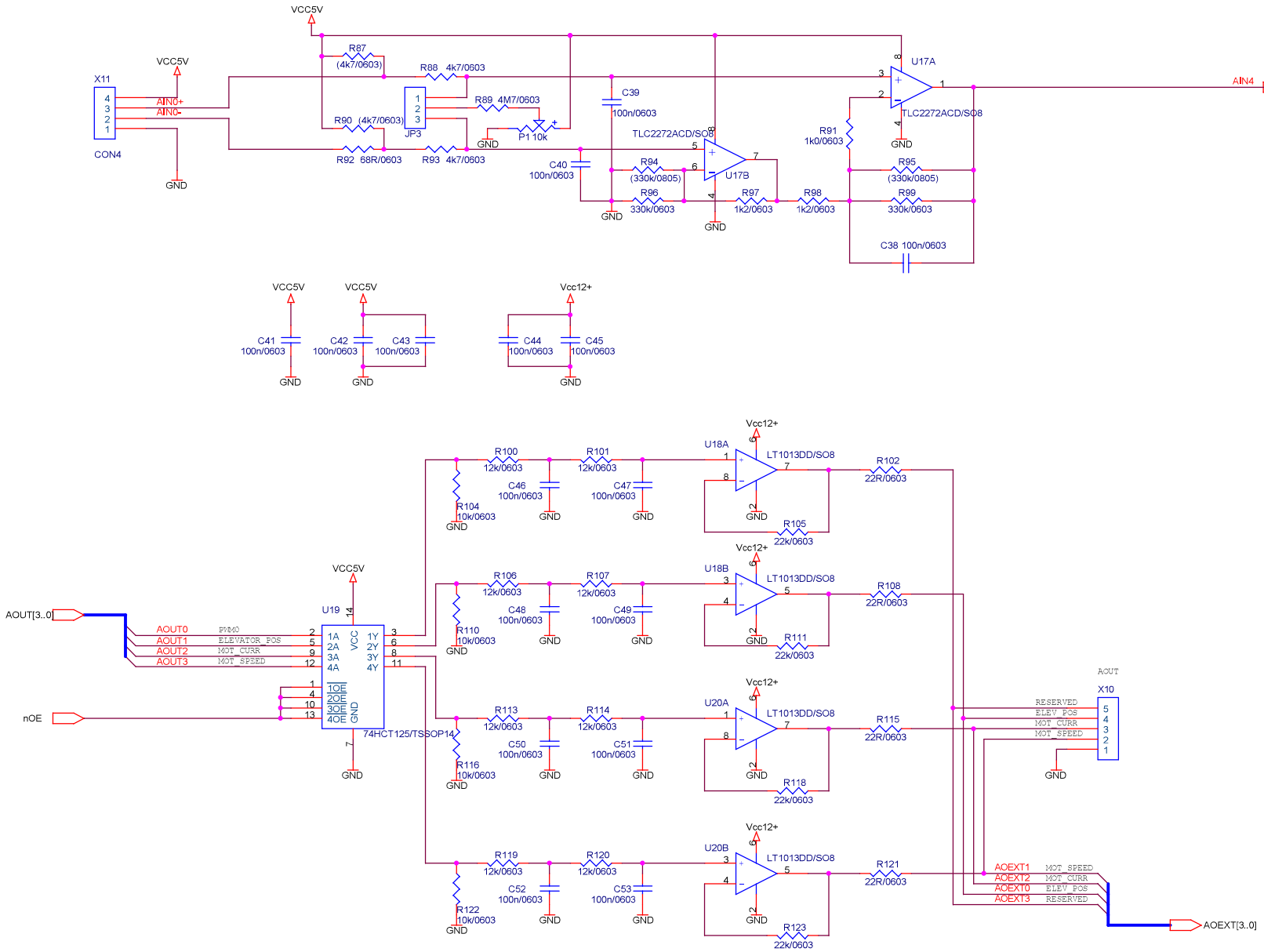
ConTeK industrial control systems	Kreslil: KastnerM	Název: ElevatorCTRL CPU conntection	ELEVCTRL.dsn	A3	Listů: 7	List: 2
	Schválil:		Číslo výkresu: CTS083		Varianta: 01	Změna: 01
	Datum: 25.11.2015					



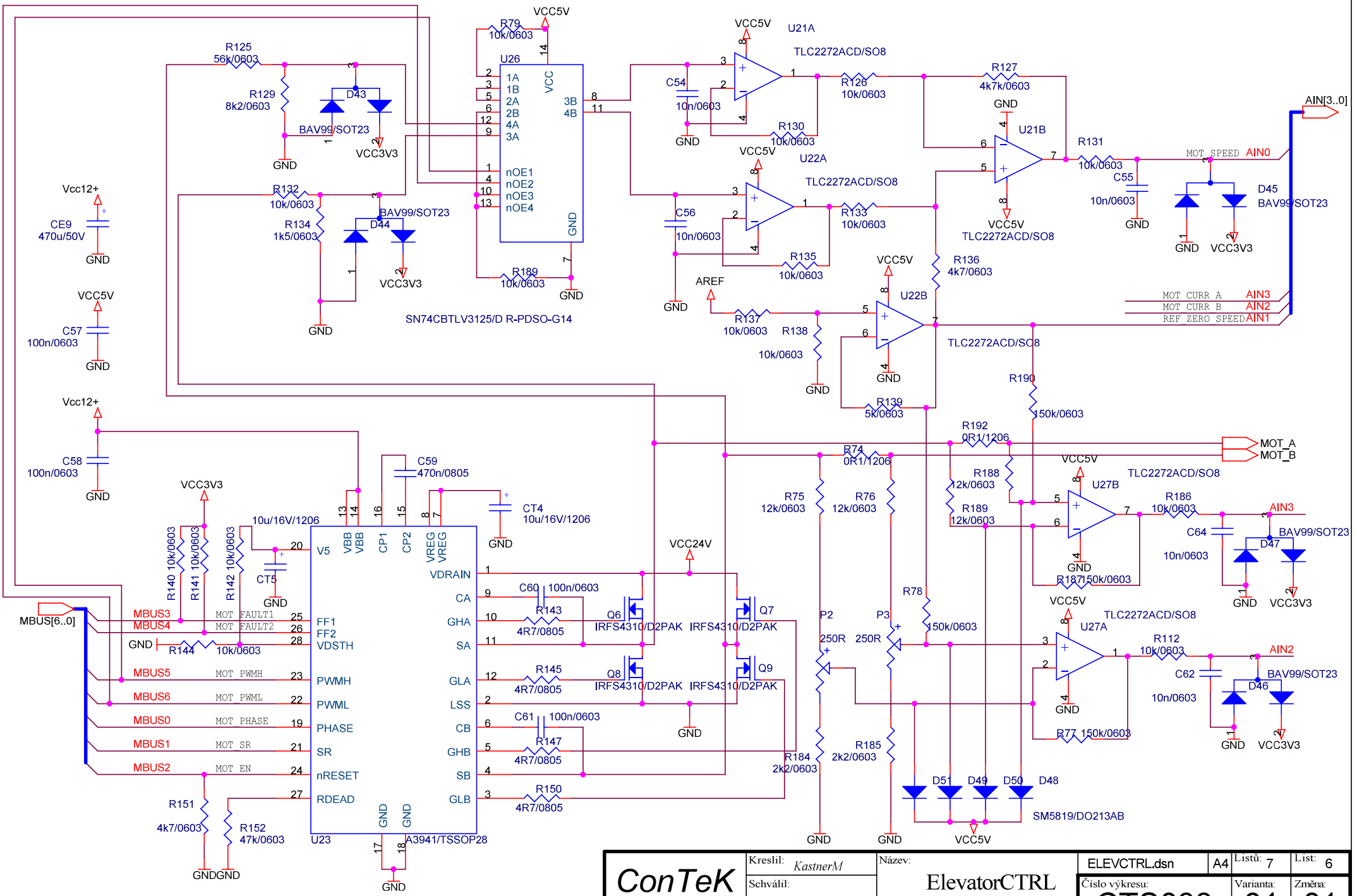
ConTeK industrial control systems	Kreslil: KastnerM	Název: ElevatorCTRL Digital Input	ELEVCTRL.dsn	A3	Listů: 7 / List: 3
	Schválil:		Číslo výkresu: CTS083	Varianta: 01	Změna: 01
	Datum: 25.11.2015				



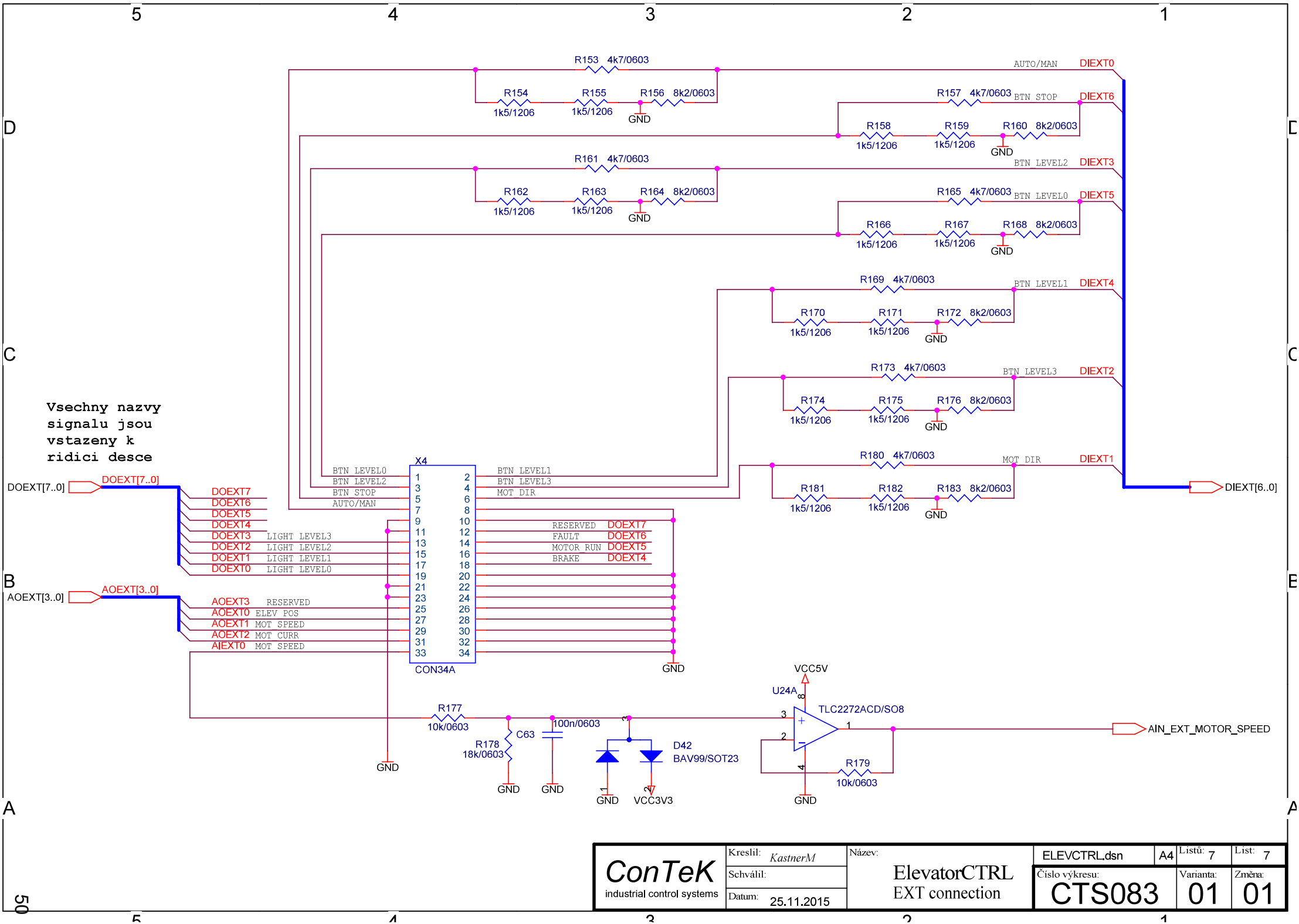
ConTeK industrial control systems	Kreslil: KastnerM	Název: ElevatorCTRL Digital Output	ELEVCTRL.dsn	A3	Listů: 7	List: 4
	Schválil:		Číslo výkresu: CTS083		Varianta: 01	Změna: 01
	Datum: 25.11.2015					



ConTeK industrial control systems	Kreslil: KastnerM	Název: ElevatorCTRL Analog IO	ELEVCTRL.dsn	A3	Listů: 7	List: 5
	Schválil:		Číslo výkresu: CTS083		Varianta: 01	Změna: 01
	Datum: 25.11.2015					



ConTeK industrial control systems	Kreslil: KastnerM	Název: ElevatorCTRL DC motor connection	ELEVCTRL.dsn	A4	Listů: 7 / List: 6
	Schválil:		Číslo výkresu: CTS083	Varianta: 01	Změna: 01
	Datum: 25.11.2015				



ConTeK industrial control systems	Kreslil: KastnerM	Název: ElevatorCTRL EXT connection	ELEVCTRL.dsn	A4	Listů: 7 / List: 7
	Schválil:		Číslo výkresu: CTS083	Varianta: 01	Změna: 01
	Datum: 25.11.2015				