KATEDRA ELEKTRICKÝCH POHONŮ A TRAKCE

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

NÁVRH ZAŘÍZENÍ PRO PODPORU AUTOMATIZOVANÉHO TESTOVÁNÍ DISTRIBUTED IO TECHNOLOGIÍ

DIPLOMOVÁ PRÁCE

KVĚTEN 2022

ONDŘEJ BĚLOVSKÝ

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bělovský** Jméno: **Ondřej** Osobní číslo: **474389**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra elektrických pohonů a trakce**

Studijní program: **Elektrotechnika, energetika a management**

Specializace: **Elektroenergetika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Návrh zařízení pro podporu automatizovaného testování Distributed IO technologií

Název diplomové práce anglicky:

Design of Automated Device for Distributed IO Technology Testing

Pokyny pro vypracování:

1. Teoretický úvod do distributed IO technologie a jejího testování.
2. Navrhněte zařízení pro podporu testování DI, DQ a IO-Link portů podle požadavků integračního testu.
3. Navrhněte a otestujte hardwarový prototyp navrženého zařízení.
4. Implementujte softwarovou knihovnu pro ovládání zařízení pomocí připojeného Raspberry Pi.
5. Vytvořte demo aplikaci pro vzdálené ovládání testovacího zařízení.

Seznam doporučené literatury:

[1] https://datasheets.raspberrypi.org/rpi4/raspberry-pi-4-reduced-schematics.pdf
[2] https://datasheets.raspberrypi.org/rpi4/raspberry-pi-4-datasheet.pdf
[3] https://datasheets.raspberrypi.org/bcm2711/bcm2711-peripherals.pdf
[4] https://io-link.com/share/Downloads/Package-2020/IOL-Interface-Spec_10002_V113_Jun19.pdf
[5] https://io-link.com/share/Downloads/Package-2020/IOL-Test-Spec_10032_V113_Jan21.pdf

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Martin Hunčovský Siemens s.r.o.

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **29.11.2021** Termín odevzdání diplomové práce: **20.05.2022**

Platnost zadání diplomové práce: **30.09.2023**

Ing. Martin Hunčovský podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání	Podpis studenta

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat svému vedoucímu Ing. Martinu Hunčovskému za cenné rady a za vedení mé diplomové práce. Dále bych chtěl poděkovat Ing. Filipovi Vodrážkovi za pomoc a konzultace při návrhu hardwarové řešení diplomové práce. Také děkuji firmě Siemens za možnost psát a tvořit toto téma.

PROHLÁŠENÍ

Prohlašuji, že jsem předloženou práci vypracoval/a samostatně a že jsem uvedl/a veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 17. května 2022

.....

ABSTRAKT

Cílem této diplomové práce je vytvoření HW návrhu a návrhu DPS prototypu pro zařízení podporující automatizované testování distributed IO technologií ve společnosti Siemens, s.r.o. Součástí práce je také implementace SW knihovny pro ovládání prototypu včetně webového GUI. V teoretické části se potom budu zabývat obecně problematikou testování distributed IO technologií a také technologií IO-Link, pro kterou je tento výrobek primárně určen. V praktické části bude popsán postup návrhu prototypu včetně vyhodnocení výsledků.

Klíčová slova: IO-Link, Distributed IO technologie, testování, automatizace, Siemens,

ABSTRACT

The aim of this diploma thesis is to create a HW design and PCB prototype design for devices supporting automated testing of distributed IO technologies in the Siemens company. Part of the work is also implementation of a software library to control the prototype, including a web GUI. In the theoretical part, I will deal with the general issue of testing distributed IO technology and IO-Link technology, for which this product is primarily intended. The practical part will describe the prototype design procedure, including the evaluation of results.

Keywords: IO-Link, Distributed IO technology, testing, automation, Siemens

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V dnešní době, kdy se spousta věcí automatizuje a dochází k obrovskému pokroku v řízení procesů, zejména pak v oblasti průmyslové automatizace, je také tlak na automatizaci dalších činností. Z tohoto důvodu mi bylo ve společnosti Siemens nabídnuto toto téma jako téma diplomové práce.

Cílem této diplomové práce je navržení prototypu pro podporu automatizovaného testování, a to jak z pohledu elektrického schéma, návrhu DPS, tak i z pohledu SW části ovládání tohoto prototypu, který jsme pracovně nazvali *IO-Link Port Tester*. Ovšem nebude fungovat pouze pro IO-Link, ale také pro klasické DI/DQ systémy.

Tento port tester by měl postupně nahradit část test racku, na kterém probíhá testování IO-link technologie v distributed IO zařízeních. Aktuální zapojení zabírá hodně místa na racku a vyžaduje to hodně manuální práce takový rack sestavit. Zatímco při využití port testeru by mělo dojít ke značnému zjednodušení sestavení racku a zároveň by také měl pomoci při automatizaci testování. Popis současného řešení test racku bude uveden v teoretické části této práce.

V teoretické části se budu zabývat obecně komunikací IO-Link, jejímu popisu, výhodám a charakteristickým vlastnostem. Dále bude také popsán přístup k testování distributed IO technologií a její specifika. Částečně také popíšu některé základní testy, které by se pomocí tohoto port testeru měly testovat. Dále popíšu možnosti konfigurace a datovou strukturu distributed IO modulů od firmy Siemens, z řady SIMATIC ET 200eco PN.

Součástí teoretické části této diplomové práce bude také popis funkce kritických součástí návrhu. V této části se budu zabývat obecným popisem funkce použitých součástek a popíšu také základní funkčnost použitého typu komunikace.

V praktické části bude popisovat a vysvětlovat postup při řešení některých částí návrhu elektrického schématu, DPS a SW knihovny. Také popíšu provedené HW testy včetně výsledků. Na konci této práce musím určitě provést zhodnocení dosažených výsledků a porovnání s požadavky na port tester.

Dalším dílčím úkolem je vytvoření SW knihovny pro ovládání navrženého zařízení pomocí připojeného Raspberry Pi. Tato knihovna by měla být v budoucnu využívána pro automatické ovládání port testeru v integračním testu. Součástí SW části by měla být i demo aplikace ve formě GUI. Jedná se o základní požadavek na port tester, protože je nezbytné mít možnost ovládat port tester i jinak než pouze programově. Pro odladění, manuální testy a drobné úpravy port testeru je určitě vhodné použít webové GUI, ale pro automatizované testování je výhodné využívat přímo SW knihovnu.

Jak jsem již zmiňoval, tak by tato diplomová práce měla mít velký přínos v oblasti automatizace testování distributed IO technologií, o které je v dnešní době enormní zájem. Tato práce by měla sloužit jako základ pro možnost implementování dalších vlastností.

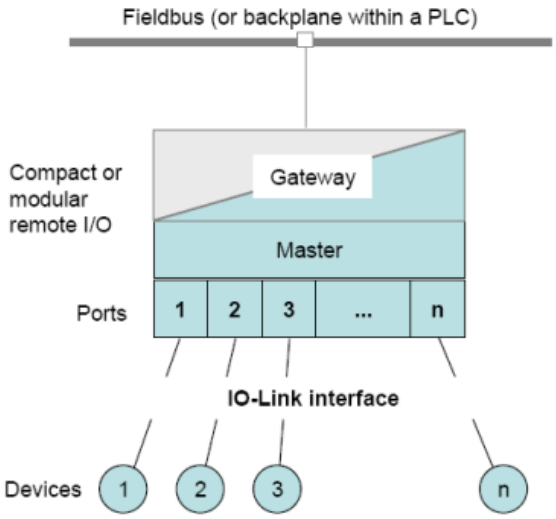
V Praze dne 11.02.2022

Bc. Ondřej Bělovský, autor diplomové práce

KAPITOLA 1: IO-LINK

Jedná se o speciální průmyslový standard pro komunikaci se senzory a akčními členy. IO-Link je komunikace typu point-to-point, specializující se na inteligentní senzory a jejich využití v průmyslové automatizaci, která vznikla jako digitální náhrada analogových vstupů/výstupů. IO-Link je navržen a standardizován podle normy ČSN EN 61131-9. Tato norma definuje elektrická připojení, požadavky a omezení pro vytvoření spolehlivého a bezpečného protokolu [1], [2].

Hlavní výhody IO-Link jsou např. snadné napojení na průmyslové sběrnice (PROFINET, Ethernet/IP, MODBUS TCP apod.), rychlý přenos naměřených hodnot včetně diagnostiky senzoru. Další velkou výhodou je kompatibilita se senzory s klasickým spínaným výstupem, což může být velká výhoda při postupné modernizaci výrobního procesu. Co se týká hardwarového zapojení, tak IO-Link komunikace je omezena na použití kabelů dlouhých maximálně 20 metrů. To je z důvodu toho, že fyzická vrstva je založena na standardním 24 V UARTu. Pro přenos se používají rychlosti 4,8, 38,4 nebo 230,4 kb/s [1], [3].

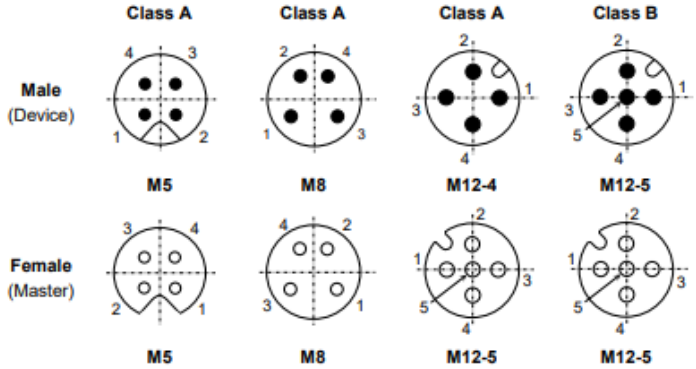


Obr. 1-1 Topologie IO-Link komunikace [3] (upraveno)

Klasická aplikace systémů s komunikací IO-Link připojuje do průmyslových sběrnic prostřednictvím IO modulů, tzv, IO-Link masterů. Této topologii se říká distributed IO a bude popsána později v kapitole 3. IO-Link master mapuje připojená IO-Link zařízení a spojuje IO-Link s nadřazenou průmyslovou sběrnicí. Topologie IO-Link komunikace je znázorněna na Obr. 1-1.

Jednotlivé senzory mohou být plně ovladatelné vzdáleně, a to díky obousměrné IO-Link komunikaci, která může probíhat jak ve směru ze senzoru do masteru (naměřená a diagnostická data), tak i naopak (konfigurační data pro senzor).

Fyzicky se IO-Link připojuje nejčastěji přes konektory M12 a někdy také přes konektory M8. Konektory M5 zobrazené na Obr. 1-2 se téměř nevyužívají [3].



Obr. 1-2 Provedení konektorů pro IO-Link komunikaci [4]

Na obrázku 1.2 je vidět provedení konektorů. Horní řada je provedení konektoru pro připojení k IO-Link zařízení, tzv. *male* konektor a spodní řada je připojení k masteru (*female*). V současné době se nejvíce využívají konektory M12-4 a M12-5, tedy konektor o velikosti M12 ve 4vodičovém, resp. 5vodičovém provedení. 5vodičové provedení nachází využití zejména při využití IO-Link zařízení typu B (*Port Class B*) a 4vodičové zase pro *Port Class A [4]*.

1.1 Připojení IO-Link zařízení

Jak je psáno výše, tak IO-Link zařízení můžeme rozdělit do dvou skupin, podle potřeby přídavného napájení. Zatímco zařízení podporující *Port Class A* přídavné napájení nepotřebují, tak v případě zařízení *Port Class B* už je externí napájení potřeba.

Tab. 1-1 Význam jednotlivých pinů při IO-Link komunikaci [4] (upraveno)

Pin	Signál	Význam	Mnou používané označení v DP
1	L+	Napájení +	1L+
2	I/Q	Nezapojeno/DI/DO (port class A)	2L+ / pin 2
	P24	Přídavné napájení + (port class B)	
3	L-	Napájení -	1M
4	C/Q	SIO/IO-Link	C/Q
5	NC	Nezapojeno (port class A)	2M
	N24	Přídavné napájení – (port class B)	

1.1.1 Port Class A

Zařízení, které je připojeno pomocí Port Class A nepotřebuje přídavné izolované napájení. Toto propojení je provedeno pomocí 2vodičového, resp. 3vodičového kabelu. Protože stav logického signálu se posuzuje na základě rozdílu napětí mezi L+ a C/Q, tak může být signál L- nezapojený. Mimo režimu IO-Link, může být podporován také klasický spínaný mód (SIO). Proto je tedy možné zapojit do IO-Link Masteru i starší senzory, které IO-Link nepodporují [3].

Pin 5 zůstane v tomto případě nezapojený a pin 2 může být buď také nezapojený nebo může sloužit jako klasický spínaný vstup, resp. výstup [3].

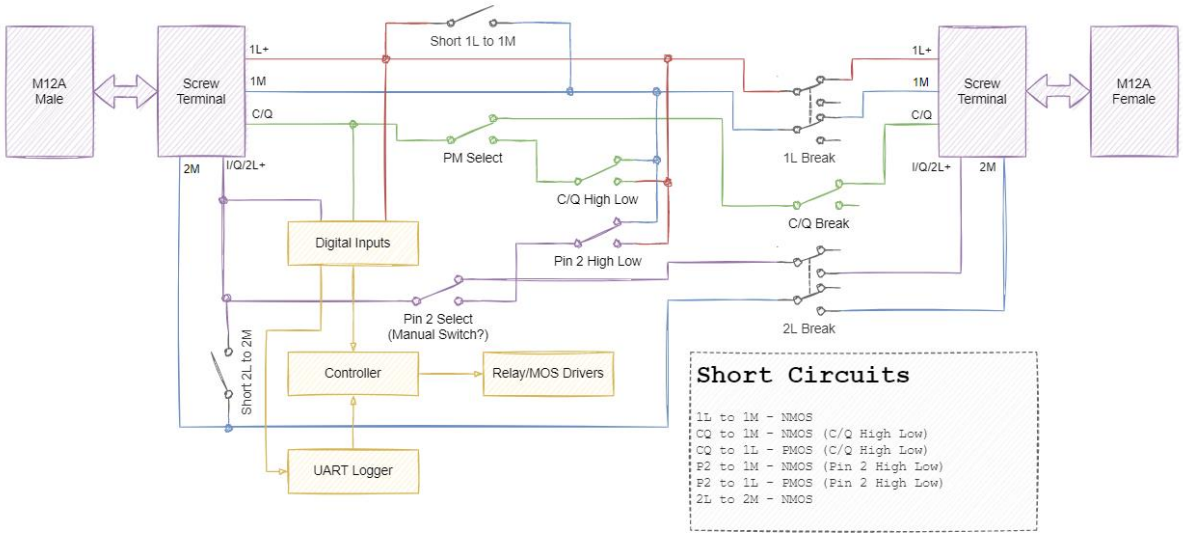
1.1.2 Port Class B

V případě IO-Link zařízení typu *Port Class B* je zapotřebí přídavné napájení. Jedná se především o výkonové výstupy. V tomto případě je přiřazení pinů následující. Piny 1,3 a 4 jsou stejné jako v případě *Port Class A*, ale na pinu 2 bude potenciál 24 V, který je izolovaný od pinu 1. Pin 5 bude v tomto případě již zapojený a jedná se o referenci pro napětí na pinu 2. Komunikace probíhá vždy na pinu 4 [3].

KAPITOLA 2: POŽADAVKY PRO IO-LINK PORT TESTER

Toto téma vzniklo na základě požadavku na zařízení, které pomůže v automatizaci testování distributed IO technologií, a to nejen IO-Link masteru, ale také klasických DI/DQ modulů. Schéma se základními požadavky, které jsem dostal k dispozici je na Obr. 2-1. Mělo by se v podstatě jednat o programově řízené spínače, které budou vyvolávat danou diagnostiku. Toto schéma nemusí být finální, slouží pouze jako návrh a seznam požadavků.

Jedním z požadavků je vyvedení znegovaného signálu C/Q na pin header. To je potřeba z důvodu sledování tohoto signálu přes logický analyzátor. Logický analyzátor je zařízení podobné osciloskopu, ale umožňuje zobrazovat logické signály v digitálním zařízení. Námi běžně používaný logický analyzátor pracuje s napětím do 5 V, proto bude potřeba převést 24 V signály na nižší napětí. Na Obr. 2-1 je tento pin header znázorněn blokem *UART looger*.



Obr. 2-1 Blokové schéma s požadavky na port tester

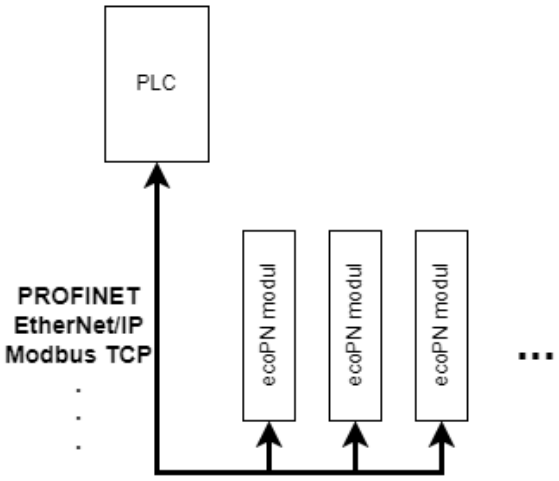
KAPITOLA 3: DISTRIBUTED IO TECHNOLOGIE

3.1 Obecné informace

Jedná se o způsob konfigurace sítě senzorů a nadřazených řídicích systémů, ve které senzory komunikují s řídícím systémem přes IO modul, který není na stejné desce jako řídící systém (např. PLC) [5].

IO modul komunikuje s řídicím systémem pomocí komunikačního modulu, který bývá součástí samotného IO modulu a komunikace probíhá přes některý s průmyslových komunikačních protokolů (PROFINET, Ethernet/IP, MODBUS TCP apod.).

Na obrázku Obr. 3-1 je znázorněna typická topologie distributed IO sítě. Úplně nahoře se nachází řídicí systém, nejčastěji se jedná o PLC. Řídicí systém je potom propojen s jednotlivými moduly pomocí jednoho, popř. i více podporovaných komunikačních protokolů. Do jednotlivých modulů jsou potom připojeny senzory a akční členy.



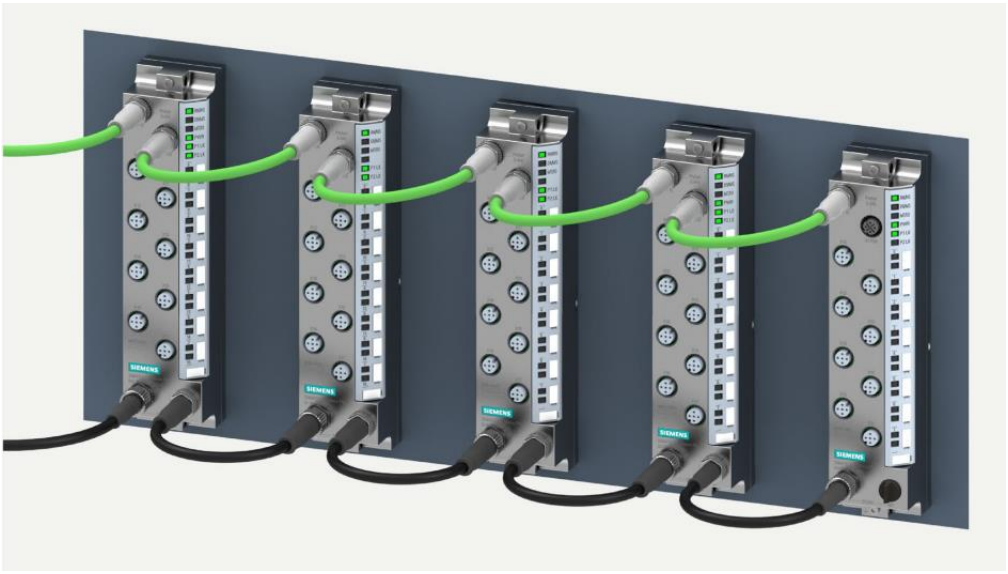
Obr. 3-1 Topologie distributed IO systémů

Na obrázku Obr. 3-1 jsou přidané moduly z řady SIMACTIC ET 200eco PN. Používám je, protože v první fázi by měl navržený port tester sloužit především k testování této technologie. Do této řady systémů patří několik zařízení: IO Digital Inputs, IO Digital Outputs, IO Digital Inputs/Outputs a IO-Link Master. Moduly z této řady jsou na Obr. 3-2, mohou být mezi sebou zapojeny jak v kruhové (tzv. ring), tak v přímé (line) topologii.

3.2 Produkty z řady SIMATIC ET 200eco PN

Řada distributed IO modulů Siemens ET 200eco PN je založená na připojení přes M12 konektory a na komunikaci přes 3 různé protokoly – PROFINET, EtherNet/IP a Modbus TCP [6], [7]. Vizuální podoba modulů této řady je na Obr. 3-2, konkrétně tato řada obsahuje několik zařízení:

- *DI 8x24V DC M12-L* – Jedná se o modul 8 digitálních vstupů na napěťové hladině 24 V.
- *DI 16x24V DC M12-L* – V tomto případě se jedná také o modul digitálních vstupů, který ovšem má na každém portu 2 vstupy (pin 2 a pin 4).
- *DQ 8x24V DC/0.5A, M12-L* – Modul digitálních výstupů, přičemž jednotlivé výstupy jsou napájeny maximálně 0,5 A.
- *DQ 8x24V DC/2A, M12-L* – Opět se jedná o modul digitálních výstupů, které ale v tomto případě mohou být napájeny až 2 A. Musí být připojeno přídavné napájení 2L+.
- *DIQ 16x24V DC/0.5A/2A, M12-L* – Modul, jehož porty mohou být konfigurovány buď jako DI nebo také jako DQ.
- *CM 8xIO-Link + DI 4x24V DC M12-L* – Posledním modulem z této řady je IO-Link master, který podporuje IO-Link komunikaci se senzory a akčními členy. Také může být nakonfigurován jako standardní DI, resp. DQ.



Obr. 3-2 Moduly řady SIMATIC ET 200eco PN [7]

KAPITOLA 4: TESTOVÁNÍ DISTRIBUTED IO TECHNOLOGIÍ

Při testování distributed IO produktů se klade důraz především na kontrolu cyklické komunikace, správnost alarmů při konkrétní diagnostice apod. Port tester by měl zjednodušit HW konfiguraci testovacího racku zejména pro kontrolu zkratů, wirebreaků apod.

4.1 Konfigurace a struktura dat

Port tester je v první fázi určen zejména k testování IO-Link Masteru řady SIMATIC ET 200ecoPN, proto budu popisovat testování na této technologii.

Slot	Name	Type	Article number	Ident number		
0	CM 8x IO-Link + DI 4x24VDC M12-L 8xM12 V5.1	IoLinkMaster	6ES7 148-6JG00-0BB0	0x0002C715		
0.1	CM 8x IO-Link + DI 4x24VDC M12-L 8xM12 V5.1			0x00000001		
0.2						
0.3						
0.4						
1	CM 8x IO-Link + DI 4x24VDC M12-L 8xM12 V1.0	Digital	6ES7 148-6JG00-0BB0	0x0000CC01		
1.1	IO-Link Master + DI 4 (Pin 2)			0x00010001		
1.2	IO-Link 2 I/ 2 O + PQI			0x00000203		
1.3	IO-Link 2 I/ 2 O + PQI			0x00000203		
1.4	IO-Link 2 I/ 2 O + PQI			0x00000203		
1.5	IO-Link 2 I/ 2 O + PQI			0x00000203		
1.6	Digital input			0x00000081		
1.7	Digital input			0x00000081		
1.8	Digital output			0x00008100		
1.9	Digital output			0x00008100		

Obr. 4-1 Ukázková konfigurace IO-Link Masteru v programu MFCT

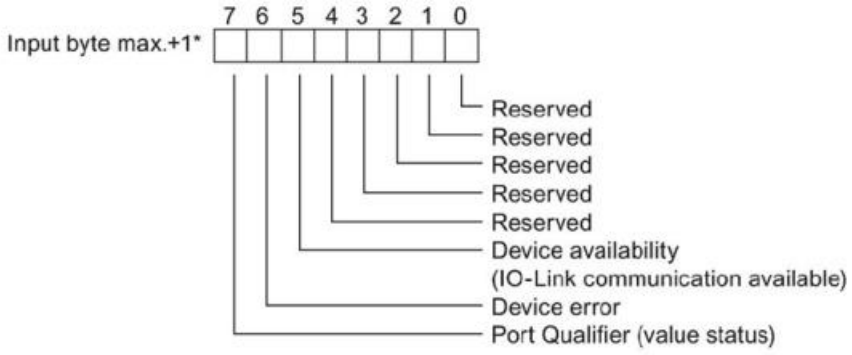
Existují různé možnosti konfigurace IO-Link masteru. Například konfigurace při komunikaci přes PROFINET může probíhat pouze přes TIA Portal nebo v kombinaci s programem S7-PCT (Port Configuration Tool), který se používá pouze pro konfiguraci jednotlivých portů IO-Link masteru a k nim připojených zařízení. Při komunikaci přes EtherNet/IP nebo Modbus TCP může konfigurace probíhat přes MFCT nebo také v kombinaci s S7-PCT. Na Obr. 4-1 je vidět příklad konfigurace IO-Link Masteru. První 4 porty jsou nastaveny pro komunikaci přes IO-Link, obsahující vstupní i výstupní procesní data o délce 2 bajty. Každý port obsahuje navíc diagnostické informace o daném portu přímo v procesních datech, tzv. PQI a, u EtherNet/IP, také informace o validitě vstupních, resp. výstupních dat a diagnostikách označené jako IDS, resp. ODS. Porty 5 a 6 jsou nastaveny jako digitální vstupy. Funkčnost tohoto módu je stejná jako pro klasický digitální vstup, rozlišuje pouze dva stavy – logickou 0 a 1. Poslední dva porty jsou konfigurovány jako digitální výstupy, tyto porty jsou schopny nastavit výstup do logické 0, resp. 1 (0 V, resp. 24 V).

Obrázek Obr. 4-2 obsahuje informace o adresaci dat pro danou konfiguraci. V prvním sloupci je vidět aktuální komunikační protokol, v mém případě je nastavený PROFINET, ale jsou podporované další 2 protokoly – EtherNet/IP a Modbus TCP. Další sloupce obsahují základní informace o datech (vstup/výstup, adresu atp.). Ve sloupci H je už vidět samotná struktura dat. V tomto případě každý port nakonfigurovaný jako *IO-Link 2 I/2 O + PQI* obsahuje 4 bajty vstupních a 2 bajty výstupních dat. Celková struktura dat je zobrazena na Obr. 4-2 a data pro port 1 jsou zvýrazněná černými obdélníky.

	A	B	C	D	E	F	G	H	I	J
1	FieldbusConnectio	Direction	RegAddr	Lo/Hi	ByteAddr	Slot	Subslot	DataItem	Submodule	BitAddr
2	PROFINET	INPUT	-	MSB	0	0	1	IDS	CM 8x IO-Link + DI 4x24VDC M12-L 8xM12 V5.1 / Input Data State	-
3	PROFINET	INPUT	-	LSB	1	1	1	Inputs	IO-Link Master + DI 4 (Pin 2)	-
4	PROFINET	INPUT	-	MSB	2	1	1	IDS	IO-Link Master + DI 4 (Pin 2) / Input Data State	-
5	PROFINET	INPUT	-	LSB	3	1	2	Input_data_2_Bytes_0	IO-Link 2 I/ 2 O + PQI	-
6	PROFINET	INPUT	-	MSB	4	1	2	Input_data_2_Bytes_1	IO-Link 2 I/ 2 O + PQI	-
7	PROFINET	INPUT	-	LSB	5	1	2	Port_status	IO-Link 2 I/ 2 O + PQI	-
8	PROFINET	INPUT	-	MSB	6	1	2	IDS	IO-Link 2 I/ 2 O + PQI / Input Data State	-
9	PROFINET	INPUT	-	LSB	7	1	3	Input_data_2_Bytes_0	IO-Link 2 I/ 2 O + PQI	-
10	PROFINET	INPUT	-	MSB	8	1	3	Input_data_2_Bytes_1	IO-Link 2 I/ 2 O + PQI	-
11	PROFINET	INPUT	-	LSB	9	1	3	Port_status	IO-Link 2 I/ 2 O + PQI	-
12	PROFINET	INPUT	-	MSB	10	1	3	IDS	IO-Link 2 I/ 2 O + PQI / Input Data State	-
13	PROFINET	INPUT	-	LSB	11	1	4	Input_data_2_Bytes_0	IO-Link 2 I/ 2 O + PQI	-
14	PROFINET	INPUT	-	MSB	12	1	4	Input_data_2_Bytes_1	IO-Link 2 I/ 2 O + PQI	-
15	PROFINET	INPUT	-	LSB	13	1	4	Port_status	IO-Link 2 I/ 2 O + PQI	-
16	PROFINET	INPUT	-	MSB	14	1	4	IDS	IO-Link 2 I/ 2 O + PQI / Input Data State	-
17	PROFINET	INPUT	-	LSB	15	1	5	Input_data_2_Bytes_0	IO-Link 2 I/ 2 O + PQI	-
18	PROFINET	INPUT	-	MSB	16	1	5	Input_data_2_Bytes_1	IO-Link 2 I/ 2 O + PQI	-
19	PROFINET	INPUT	-	LSB	17	1	5	Port_status	IO-Link 2 I/ 2 O + PQI	-
20	PROFINET	INPUT	-	MSB	18	1	5	IDS	IO-Link 2 I/ 2 O + PQI / Input Data State	-
21	PROFINET	INPUT	-	LSB	19	1	6	Input	Digital input	-
22	PROFINET	INPUT	-	MSB	20	1	6	IDS	Digital input / Input Data State	-
23	PROFINET	INPUT	-	LSB	21	1	7	Input	Digital input	-
24	PROFINET	INPUT	-	MSB	22	1	7	IDS	Digital input / Input Data State	-
25	PROFINET	INPUT	-	LSB	23	1	2	ODS	IO-Link 2 I/ 2 O + PQI / Output Data State	-
26	PROFINET	INPUT	-	MSB	24	1	3	ODS	IO-Link 2 I/ 2 O + PQI / Output Data State	-
27	PROFINET	INPUT	-	LSB	25	1	4	ODS	IO-Link 2 I/ 2 O + PQI / Output Data State	-
28	PROFINET	INPUT	-	MSB	26	1	5	ODS	IO-Link 2 I/ 2 O + PQI / Output Data State	-
29	PROFINET	INPUT	-	LSB	27	1	8	ODS	Digital output / Output Data State	-
30	PROFINET	INPUT	-	MSB	28	1	9	ODS	Digital output / Output Data State	-
31	PROFINET	INPUT	-	LSB	29	-	-	-	Reserved	-
32	PROFINET	OUTPUT	-	MSB	0	1	2	Output_data_2_Bytes_0	IO-Link 2 I/ 2 O + PQI	-
33	PROFINET	OUTPUT	-	LSB	1	1	2	Output_data_2_Bytes_1	IO-Link 2 I/ 2 O + PQI	-
34	PROFINET	OUTPUT	-	MSB	2	1	3	Output_data_2_Bytes_0	IO-Link 2 I/ 2 O + PQI	-
35	PROFINET	OUTPUT	-	LSB	3	1	3	Output_data_2_Bytes_1	IO-Link 2 I/ 2 O + PQI	-
36	PROFINET	OUTPUT	-	MSB	4	1	4	Output_data_2_Bytes_0	IO-Link 2 I/ 2 O + PQI	-
37	PROFINET	OUTPUT	-	LSB	5	1	4	Output_data_2_Bytes_1	IO-Link 2 I/ 2 O + PQI	-
38	PROFINET	OUTPUT	-	MSB	6	1	5	Output_data_2_Bytes_0	IO-Link 2 I/ 2 O + PQI	-
39	PROFINET	OUTPUT	-	LSB	7	1	5	Output_data_2_Bytes_1	IO-Link 2 I/ 2 O + PQI	-
40	PROFINET	OUTPUT	-	MSB	8	1	8	Output	Digital output	-
41	PROFINET	OUTPUT	-	LSB	9	1	9	Output	Digital output	-

Obr. 4-2 Struktura dat IO-Link Masteru

Význam jednotlivých bitů v bajtu, který obsahuje informace o stavu jednotlivých portů je na Obr. 4-3. První 2 bity jsou rezervované, to znamená, že momentálně nemají žádné využití a jejich hodnota je logická 0. Důležitý je např. bit 2, který dává informace o IO-Link zařízení, především o stavu jeho parametrů. Je-li tento bit 1, tak parametry IO-Link zařízení a masteru nejsou totožné. Dalším podstatným bitem je bit *Device availability*, který informuje o stavu komunikace mezi zařízením a masterem. V případě že na zařízení dojde k nějaké chybě, tak se tato chyba projeví tím, že *Device error* bude mít hodnotu logické 1. Pokud bych chtěl kontrolovat správnost dat, přenášených z IO-Link zařízení, tak musím kontrolovat bit *Port Qualifier* [8].



Obr. 4-3 Struktura dat PQI [8]

4.2 Základní testy

4.2.1 Přerušení komunikačního signálu

Princip tohoto testu spočívá v odpojení a opětovném připojení IO-Link zařízení, kterým může být např. digitální vstupní nebo výstupní modul SIMATIC ET 200AL. Při přerušení je zapotřebí vyčíst diagnostická data, která se ovšem pro jednotlivé protokoly vyčítají různě. Například v případě PROFINETu je možné tyto informace vyčíst přímo z PLC, kam se při přerušení komunikace pošle alarm, jehož součástí jsou i veškerá diagnostická data. Po zkontrolování dat znovu připojím komunikační signál a je potřeba zkontrolovat, jestli se komunikace znovu správně navázala. To se dělá pomocí zrcadlení výstupů na vstupy, tzn. měním výstupy, čtu vstupy a kontroluji, zda se správně mění.

Tato diagnostika jde také kontrolovat u klasického DI modulu. V tomto případě se jedná o signál, po kterém se posílají procesní data.

4.2.2 Zkraty

Dalším typem testů je test správné diagnostiky zkratů. Zkraty na IO modulech můžou být různé např. 1L+ - 1M, CQ – 1L+, CQ – 1M apod. Princip testování zkratů je hodně podobný jako u přerušení komunikačního signálu. Modul je ve stavu, kdy na něm není žádný error nebo diagnostika. Potom vyvolám příslušný zkrat a zkontroluji stav modulu a diagnostiku. Odstraním zkrat a zkontroluji komunikaci a stav modulu po zkratu.

4.2.3 Přerušení přídavného napájení 2L+

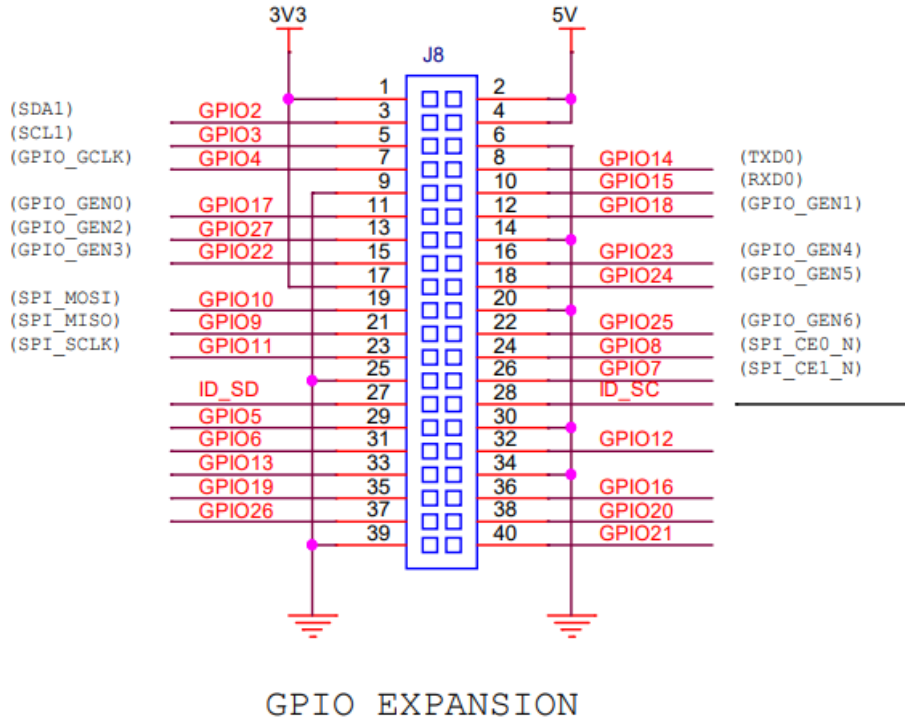
Tato kontrola se nechá provádět pouze u výstupních modulů a IO-link masterů při konfigurace na port class B. Pokud je diagnostika povolená a na portu chybí napájení 2L+, tak bych měl obdržet diagnostiku se správnými daty. Potom buď můžu připojit 2L+, pokud toto napájení chybí, tak stačí zakázat zobrazování této diagnostiky. Až diagnostika zmizí, tak zkontroluji cyklická data.

KAPITOLA 5: POPIS HLAVNÍCH HW SOUČÁSTÍ NAVRHOVANÉHO ŘEŠENÍ

5.1 Raspberry Pi

Jedná se o jednodeskový počítač, který může být připojen ke vstupním a výstupní periferiím pomocí pin headeru 2x20. Přiřazení jednotlivých pinů je na Obr. 5-1. GPIO piny mohou sloužit jako vstupy tak i výstupy. Konfigurace jednotlivých pinů se dělá programově. Raspberry Pi se napájí napětím o velikosti 5 V a odpovídají tomu piny 2 a 4. Raspberry Pi může sloužit také jako zdroj 3,3 V, které slouží k napájení ovládaných periferií [9].

Obsahuje jeden konektor na napájení, 4 USB porty, 1 komunikační port RJ45 a mikro HDMI port pro připojení externího monitoru [9].



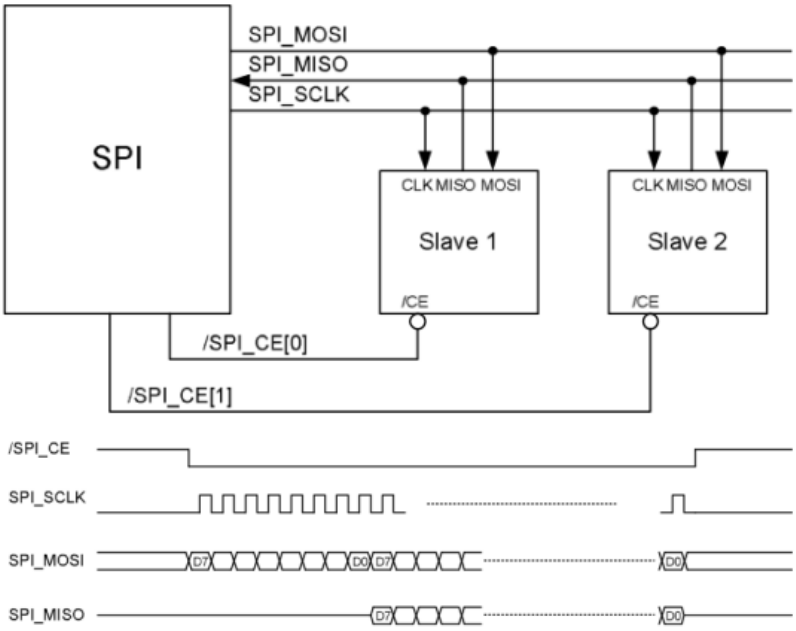
Obr. 5-1 Popis přiřazení jednotlivých pinů na Raspberry Pi [10]

5.1.1 Použité vstupy a výstupy

5.1.1.1 SPI

SPI sériová komunikační sběrnice používaná ke komunikaci mezi mikrokontrolery nebo mezi mikrokontrolerem a periferiemi (např. displeje, teplotní senzory apod.). Umožňuje komunikaci mezi větším počtem uzlů, což je podstatný rozdíl např. oproti sériovému portu RS-232 C, který je pouze pro 2 uzly. SPI obsahuje hodinový signál, jehož frekvence může dosahovat až 70 MHz, běžně se, ale používá frekvence 10 MHz. Jeden uzel obvykle pracuje v roli *masteru* a ostatní uzly pracují v režimu *slave* [11], [12].

Hodinový signál je rozveden z *master* uzlu do všech *slave* uzlů pomocí vodiče, jenž se většinou označuje *SCK*. Uzly jsou dále propojeny i dalšími signály. Pro obousměrný přenos dat se používají signály *MOSI* a *MISO*. Každý komunikační kanál obsahuje také jeden signál *CS*, kterým se povoluje/zakazuje zápis dat do daného uzlu [11], [12].



Obr. 5-2 Průběh SPI komunikace [10]

Na Obr. 5-2 je znázorněn časový průběh SPI komunikace. Komunikace je navázána, jakmile má chip select (na Obr. 5-2 tomu odpovídá */SPI_CE[0]*, resp. */SPI_CE[1]*) hodnotu logické 0. Poté se tam s nějakým zpožděním aktivuje hodinový signál a podle *MOSI* se zapisují hodnoty do zařízení. Také přes signál *MISO* můžu využívat SPI komunikaci ke čtení dat z periferního zařízení. Po skončení komunikace se nastaví signál chip select do 1 a posílání dat je zablokováno [11], [12].

5.1.1.2 GPIO

Tento typ signálu funguje jako klasický digitální vstup, resp. výstup. To znamená, že pokud je signál nastaven jako výstup, tak ho musím programově nakonfigurovat jako GPIO výstup a potom v nějakém místě v programu měnit manuálně jeho hodnotu mezi 0 a 1. Pokud chci daný signál GPIO použít jako vstup, tak ho musím opět nakonfigurovat jako vstupní signál a dále už ho můžu v kterémkoliv místě programu libovolně číst [13].

5.2 Shift registry

5.2.1 Vstupní shift registry

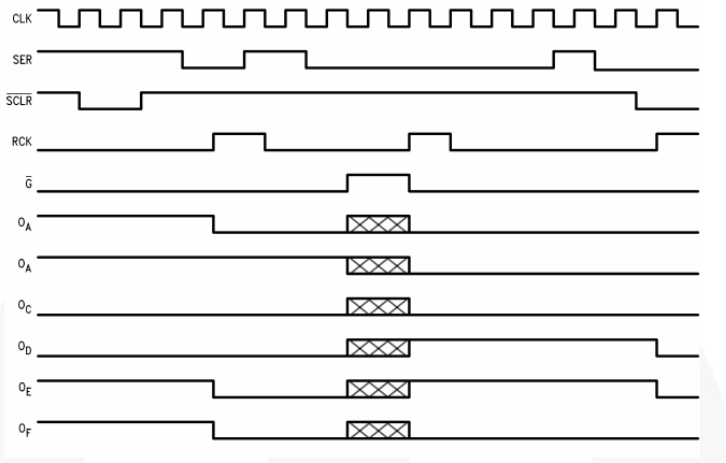
Vstupní shift registry sbírají paralelní signály a převádí je do jednoho sériového signálu, který se v programu nechá rozdělit na jednotlivé signály. Na pin 2 vstupního shift registru je připojen hodinový signál z Raspberry Pi a s každou náběžnou hranou hodinového signálu se přečte jeden paralelní vstup na pinech označených D0 až D7. Tím vznikne jeden signál a v programu se nechá rozdělit podle hodinového signálu do jednotlivých proměnných.

5.2.2 Výstupní shift registry

Výstupní shift registr ovládá paralelními signály na svém výstupu např. tranzistory. Funguje na opačném principu než vstupní shift registr, převádí signál vytvoření v Raspberry Pi na paralelní výstupy. Opět s každou náběžnou hranou hodinového signálu se nahrají jednotlivé bity do shift registru. Obsahuje také ovládací piny, které slouží k tomu, aby bylo řízení plně pod kontrolou vývojáře. Klasické ovládání výstupního shift registru by mohlo vypadat např. takto: výstupy jsou pomocí ovládacího pinu zablokované, nahraje se signál do shift registru, signál se rozdělí do jednotlivých výstupů, a nakonec se tyto výstupy aktivují. Důležitou vlastností shift registrů je, že se signál „posouvá“. Pokud je k dispozici 8bitový SR, tak pokud se pošle např. 2bitový signál, tak se signál dostane pouze k prvním 2 výstupům. Výstupy, které se nemají aktivovat se musí vyplnit

logickými nulami. Na Obr. 5-3 je vidět časový diagram zpracování sériového signálu shift registrem.

Pokud je signál $\overline{G}$ (*Output Enable*) v hodnotě logické 0, tak shift registr pracuje standardně, ale pokud ho nastavím do logické 1, tak výstupy budou ve stavu vysoké impedance, tedy bude se chovat jako by byly výstupy odpojeny.

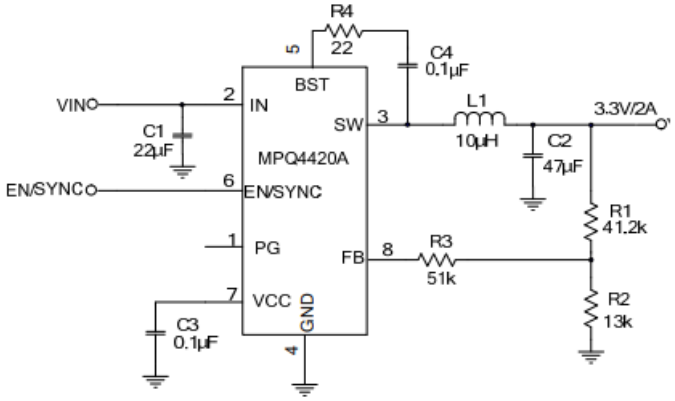


Obr. 5-3 Časový diagram zpracování signálu výstupním shift registrem [14]

5.3 Buck konvertor

Pro převod stejnosměrného napětí z 24 V na 3,3 V bude využit DC-DC buck konvertor, konkrétně typ MPQ4420A. Na vstupu tohoto konvertoru jsou doporučené hodnoty napětí 4–36 V. Tento regulátor má regulaci 0,8x – 0,9x VIN, to znamená, že pokud chci napětí 3,3 V, tak ho musím na výstupu vhodně upravit, pomocí napěťového děliče. Tento dělič je zobrazen na obrázku Obr. 5-4 a je tvořen rezistory R1 a R2. Rovnice (5-1) uvádí vzorec pro výpočet spodní větve odporového děliče ze známého výstupního napětí a horní větve děliče [15].

Na obrázku Obr. 5-4, kde je zobrazená typická aplikace podle datasheetu použitého konvertoru, je vidět, že na vstupu i na výstupu jsou paralelně řazeny kondenzátory. Kondenzátory jsou tam kvůli stabilizaci napětí, protože chci mít na vstupu do regulátoru i na jeho výstupu co možná nejvíc vyhlazený průběh napětí.

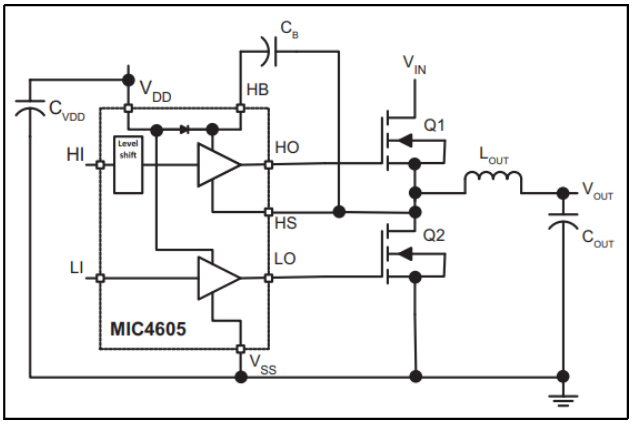


Obr. 5-4 Typická aplikace buck konvertoru [15]

$$R2 = \frac{R1}{\frac{V_{OUT}}{0.792V} - 1} \tag{5-1}$$

5.4 Half-bridge driver

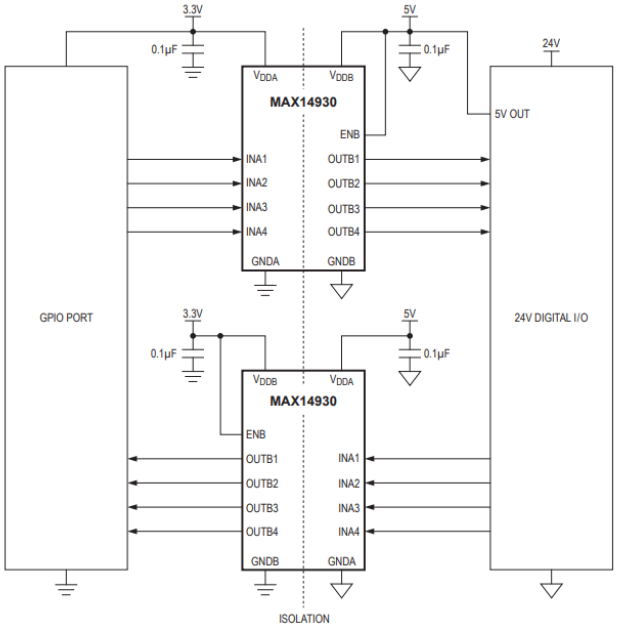
Half-bridge driver je integrovaný budící obvod v půl můstkovém zapojení. Half-bridge drivery MIC4605-1YM budu ve svém návrhu používat pro buzení 2 tranzistorů. Na Obr. 5-5 je ukázka typické aplikace tohoto budiče. Na tomto obrázku je také vidět, že je potřeba přidat 2 kondenzátory. Bypass kondenzátor C_{VDD}, který je mezi napájecími svorkami a kondenzátor C_B, který je zde pro spínání tranzistoru Q1 [16].



Obr. 5-5 Typická aplikace budiče tranzistorů MIC4605-1YM [16]

5.5 Digitální izolátor

Další velmi důležitou součástkou využitou v port testeru je digitální izolátor, konkrétně využívám ve svém návrhu součástku MAX14930BASE+T. Tato součástka je pro můj návrh velice důležitá, protože všechny signály, které se využívají pro komunikaci s Raspberry Pi, musí být izolovány. Toto oddělení 1M a 3M pomocí izolátorů bude využito z důvodu ochrany Raspberry Pi. Jedna jeho strana je napájena z jednoho potenciálu a druhá z jiného. Model MAX14930BASE+T může být napájen napětím 1,71 - 5,5 V. Na obr. 6-5 je vidět obecná typická aplikace tohoto izolátoru. Na každé napájecí větvi musí být kondenzátor, musí tam být kvůli potlačení rušení. Blok „24 V DIGITAL I/O“ pro mojí aplikaci představuje většinu DPS a blok „GPIO PORT“ zase představuje Raspberry Pi a jeho vstupy/výstupy. Tento izolátor má podle svého datasheetu izolační hladinu až 3,75 kV [17].



Obr. 5-6 Typická aplikace digitálního izolátoru MAX14930BASE+T [17]

5.6 Optočlen s tranzistorovým výstupem

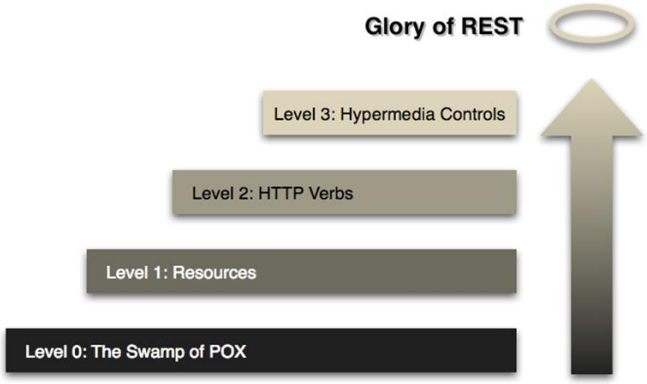
Optočlen s tranzistorovým výstupem je založen na principu otevření tranzistoru pomocí LED diody na vstupu tohoto optočlenu. Ve chvíli, kdy bude na vstupu napětí, tak se rozsvítí dioda a tranzistor se otevře. A naopak, pokud na vstupu nebude napětí dioda nebude svítit a výstupní tranzistor zůstane zavřený. Optočlen je přidán do návrhu z důvodu izolace. Bude využit pouze u signálů, u kterých není zapotřebí dosáhnout vysoké rychlosti spínání, tzn. u signálů ovládajících obvody pro zkraty na potenciálu 2M. Pro komunikační signály, které jsou velmi rychlé budu využívat digitální izolátor.

KAPITOLA 6: REST API

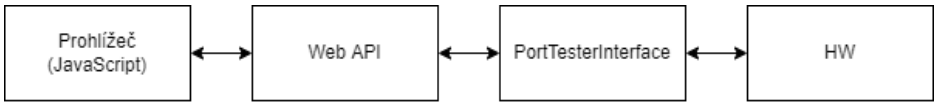
Termínem API se označuje rozhraní pro programování aplikací. Jedná se o sbírku procedur, funkcí, tříd či protokolů určité knihovny, které může vývojář využívat. API určuje, jakým způsobem jsou funkce knihovny volány z programu. [18]. V praxi se to využívá např. při propojení mobilní a webové aplikace. V mém případě budu API využívat pro přenos dat mezi ovládací knihovnou Raspberry Pi a webovým rozhraním. Data většinou bývají přenášena ve formě JSON.

REST API je specifický druh API. Jedná se o rozhraní orientované na data. REST je možné pro přehlednost rozdělit do 4 úrovní, tyto úrovně jsou na Obr. 6-1 [19].

Na úrovni 0 je uskutečňován samotný přenos dat. Úroveň 1 obsahuje zdroje, každý zdroj má jeden koncový bod, kde ho lze dosáhnout. Další úroveň obsahuje klíčová slova http protokolu, z nichž je patrně nejpoužívanější GET, pomocí něhož se získávají data. Dále potom obsahuje také např. POST (vytvoření dat), DELETE (smazání), atp. Součástí HTTP protokolu jsou také stavové kódy, které získáme jako odpověď při volání konkrétní HTTP metody. Poslední úroveň je nazvaná *„Hypermedia Controls“* a jde o to, že nám stačí znát pouze základní koncový bod, ostatní koncové body získáme jako odpověď společně s daty [19].



Obr. 6-1 Jednotlivé úrovně RESTu [19]

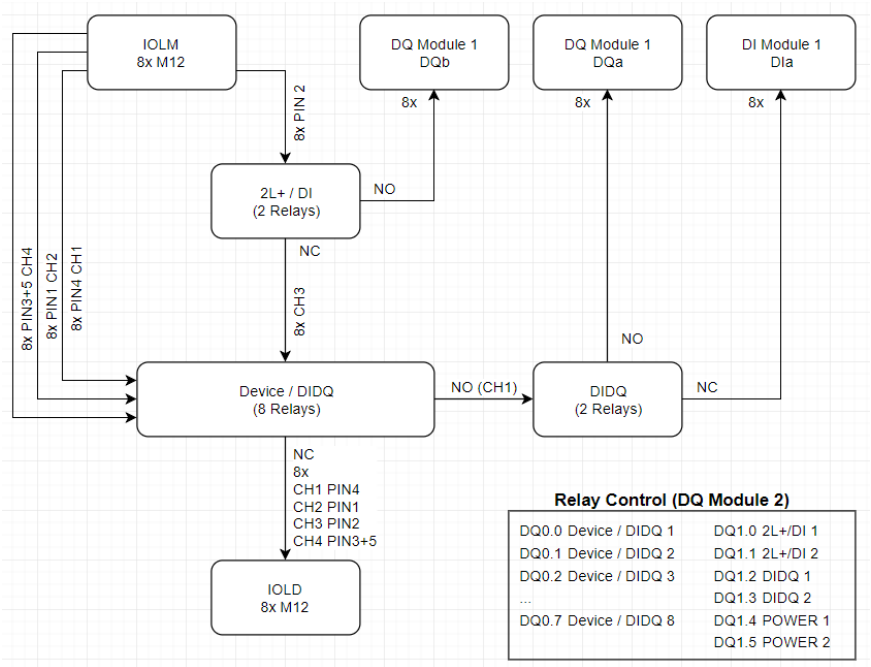


Obr. 6-2 Blokové schéma komunikace

KAPITOLA 7: NAVRHOVANÉ ŘEŠENÍ

7.1 Původní koncept testovacího racku

Port tester by měl nahradit aktuální HW konfiguraci, která umí simulovat přerušení vodiče, přepínat mezi DI a DQ, ale neumí dělat zkraty, které se stále musí dělat manuálně, a navíc jak je vidět v blokovém schématu na Obr. 7-1, má spojené oba potenciály 1M i 2M.

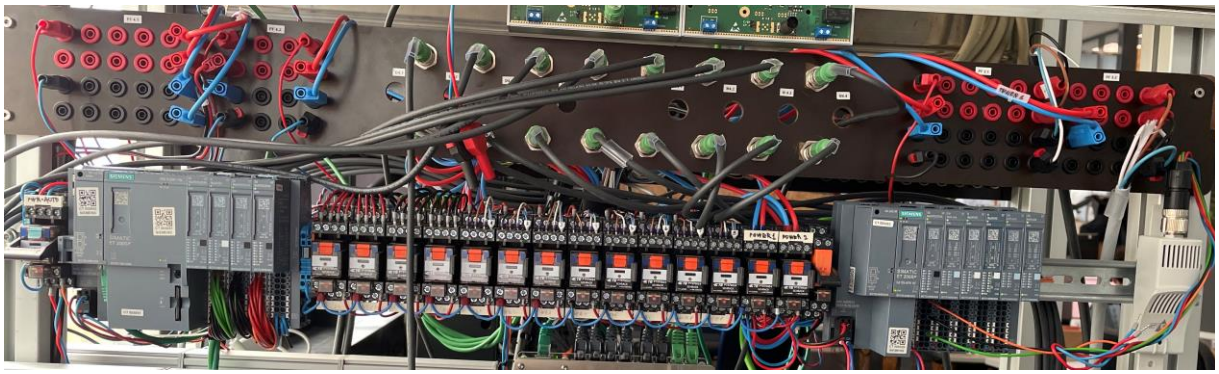


Obr. 7-1 Blokové schéma stávající HW konfigurace pro testování IO-Link

Každý IO-Link master má vyvedené všechny signály na relé. Signál, označený v blokovém schématu jako „PIN 2“, je signál, který může sloužit buď jako DI, DQ nebo přídavné napájení 2L+. Tento signál je vyvedený na relé, které přepíná signál podle toho, jestli je tento signál nakonfigurovaný jako 2L+ nebo DI. Pokud je v režimu DI, tak tento signál vede do karty digitálních výstupů a z řídicího systému můžu tento signál měnit a pomocí IO-Link Masteru číst aktuální hodnotu. Ovšem pokud je tento signál v režimu 2L+, tak je vyveden do dalších relé.

Všechny ostatní signály jsou vedeny přímo do hlavní skupiny relé, která přepíná tyto signály podle toho, jestli jsou nastaveny pro IO-Link zařízení (senzory, akční členy komunikující po IO-Linku) nebo pro digitální vstupy/výstupy. Pokud je daný port nastaven do režimu DI/DQ, tak jsou zde další 2 relé, které přepínají na vstupní, resp. výstupní kartu podle nastavení každého portu.

Na Obr. 7-2 je fotografie aktuální konfigurace racku pro testování technologie IO-Link. Je tam vidět již zmíněná celá řada relé a 2 moduly z řady ET 200SP. Do těchto modulů jsou vyvedeny jednotlivé signály z relé, které odpovídají tomu, že daný port je přepnut do režimu DI, resp. DQ. Lišta nad relé obsahuje napájecí vodiče a konektory M12, které vedou z příslušných IO-Link masterů.



Obr. 7-2 Aktuální HW konfigurace

7.2 Návrh nového řešení

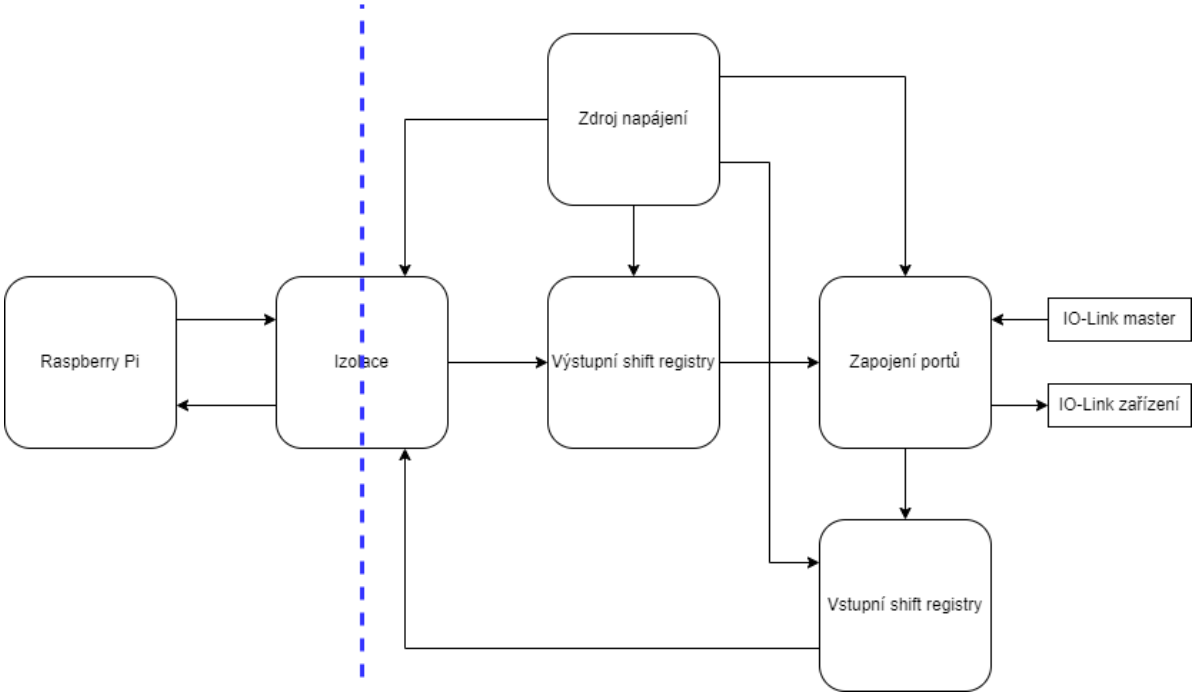
Nové řešení bude vycházet z blokového schématu na Obr. 7-3. Raspberry Pi jsem navrhl jako hlavní řídící jednotku. Použil jsem ji, protože je na Raspberry Pi velmi jednoduchá implementace web severu a nechá se jednoduše programovat (např. v Pythonu). Pro ovládání výstupů použiji výstupní shift registry. Zápis do shift registrů bude zajištěn přes SPI rozhraní Raspberry Pi. Překlopení shift registrů do výstupních registrů bude zajištěno pomocí GPIO signálů. Pro čtení vstupů budu používat také shift registry, ale vstupní, které převádí paralelní signály do jednoho sériového signálu.

Velkou výhodou SPI rozhraní je, že je hardwarově naimplementováno v Raspberry Pi a díky knihovně *spidev* je jednoduše dostupné pro program psaný v jazyce Python. Pomocí příkazů v této knihovně jednoduše navážu komunikaci a odešlu data [11], [12]. Pokud bych vytvářel tuto komunikaci pomocí GPIO signálů, tak komunikace bude výrazně pomalejší a implementace pracnější. Nemohl bych použít knihovnu *spidev* a celou její funkčnost bych musel naimplementovat ve vlastním programu.

Bude nutné poslat signál do shift registrů, kdy se mají výstupy nahrané v shift registrech překlopit na fyzické výstupy nebo kdy se mají resetovat apod. Tyto signály budou typu GPIO. Pro práci s GPIO signály lze využít Python knihovnu *RPi.GPIO*. Tento typ signálů se musí ovládat manuálně, to znamená, že každá změna musí být napsaná v programu [13]. Je to jednoduchý nástroj, jak ovládat jednotlivé signály. GPIO signály budu používat pro překlápění vstupních i výstupních shift registrů.

Pro spínání jednotlivých zkratů a wirebreaků budou použity MOSFET tranzistory buzené buď samostatným budícím obvodem nebo integrovaným budícím obvodem ve formě half-bridge driveru. U wirebreaků se nabízelo použití dvou kanálových relé, ale tento způsob nebude využit, protože relé jsou mnohonásobně větší a pomalejší.

Softwarová část port testeru bude ovládána pomocí aplikace (GUI, test skript), která bude komunikovat s web serverem pomocí HTTP (REST API). Pro implementaci REST API jsem vybral knihovnu *flask,* a to zejména pro její jednoduchost.

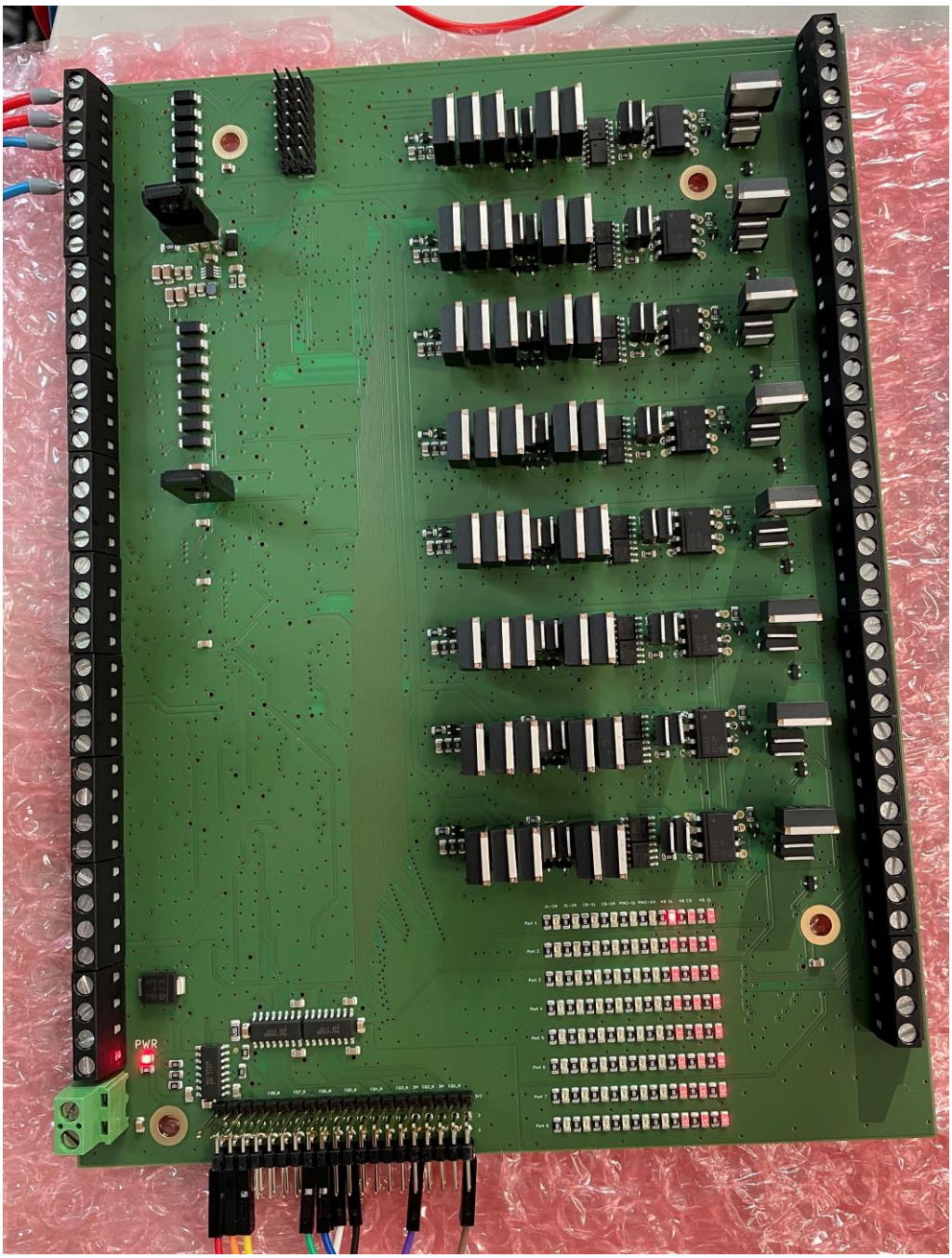


Obr. 7-3 Blokové schéma port testeru

KAPITOLA 8: ELEKTRICKÉ SCHÉMA HW

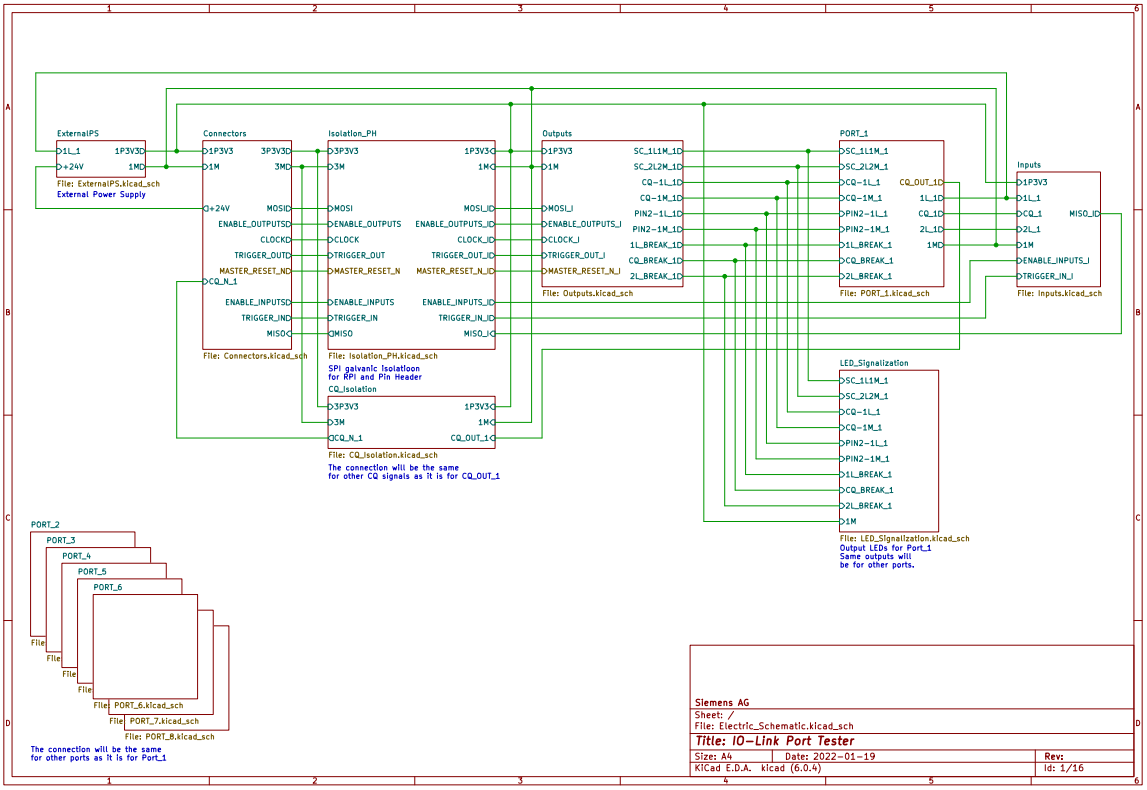
8.1 HW konfigurace při využití IO-Link port testeru

Pokud využiji místo stávající HW konfigurace port tester, tak celá soustava relé, popsaná v kapitole 7.1, bude nahrazena jednou deskou plošných spojů, ke které se připojí všechny porty IO-Link Masteru. Funkčnost stávajícího řešení s relé bude kompletně pokryta port testerem, který má další velké výhody proti staré konfiguraci. Na port testeru jsou vzájemně izolované oba potenciály (1L i 2L) a umožňuje uživateli dělat také zkraty. Dalším požadavkem pro efektivnější využití port testeru byla možnost ovládání port testeru pomocí programu. Já jsem zvolil komunikaci aplikace s web serverem přes REST API, které je možné velmi dobře využít při automatizaci testování.



Obr. 8-1 Fotografie vyrobeného port testeru

Na dalším obrázku je znázorněno blokové schéma port testeru. Na vstupu je blok napájení, ve kterém jsou zdroje napětí 3,3 V a dvakrát 15 V. Dva zdroje napětí 15 V jsou zde z důvodu napájení half-bridge driverů. Některé tyto drivery budí tranzistory na referenčním potenciálu 1M a některé zase tranzistory na 2M, proto je zapotřebí mít i napájení driverů zajištěno z obou potenciálů. Následuje blok s konektory. V tomto bloku jsou rozkresleny všechny fyzické konektory, které jsou vstupem či výstupem desky plošného spoje. Za tímto blokem musí následovat blok izolací. Tento blok tam musí být kvůli ochraně připojeného Raspberry Pi. Poté je už blok se všemi výstupy. Vstupem tohoto bloku jsou SPI a GPIO signály z Raspberry Pi a výstupem jsou už signály, které jdou z shift registrů do jednotlivých budících obvodů portů. Dalším blokem je zapojení každého portu, ze kterého se ještě vyvedou signály, které je potřeba sledovat v Raspberry Pi (1L+, 2L+ a CQ). Posledním blokem na tomto schématu je zapojení signalizačních LED diod.



Obr. 8-2 Blokové schéma HW návrhu

8.2 Konektory

Na DPS je hned několik konektorů. Nachází se tam 2 konektory, které definují velikost desky. Jsou to vstupní a výstupní šroubovací konektory pro připojení portů IO-Link masteru a zařízení. Dalším výrazným konektorem je pin header se 40 piny ve dvou řadách, kterým je připojeno Raspberry Pi k desce.

Důležitým konektorem je také pin header ve 3 řadách po 8 pinech. Tento pin header slouží k manuální izolaci pomocného zdroje napětí na pinu 2. Tato izolace je velmi důležitá, protože bez ní bych pro izolaci 2L+ musel použít tranzistory s velkou izolační hladinou.

Napájení desky může být také zprostředkováno dvěma způsoby – přímo z IO-Link portů nebo z externího zdroje napětí. Externí zdroj napětí musí být na stejném potenciálu jako napájení IO-Link portů.

Posledním konektorem je ten, na kterém jsou vyvedeny invertované signály C/Q a referenční potenciál Raspberry Pi. Je důležitý kvůli analýze IO-Link signálu na logickém analyzátoru.

8.3 Napájení DPS

Jak jsem již zmiňoval, tak napájení DPS je uskutečňováno buď přímo z IO-Link portů nebo z externího zdroje napětí. Deska by měla být napájena především z externího zdroje, proto jsou na tomto vstupu 2 Schottkyho usměrňovací diody – jedna pro 1L+ a druhá pro 2L+. Tyto diody jsou uvedeny ve schématu v příloze *B.1, strana 2, součástky D209 a D218.* Na portech IO-Link masteru je klasická dioda. Schottkyho dioda má menší úbytek napětí. To znamená, že pokud bude externí napájení připojeno, tak bude proud téct právě přes tuto diodu.

Součástky na desce nebudou napájeny 24 V, ale napájení bude uskutečňováno na dvou napěťových úrovních – 3,3 V a 15 V.

Pro převod napětí z 24 V na 3,3 V jsem použil buck konvertor. Na vstupu je soustava kondenzátoru především kvůli stabilizaci napětí. Konvertor bude pracovat se vstupním napětím 18,6 V – 36 V. Na výstupu je napěťový dělič, který převádí napětí na požadovanou hodnotu. Pro moje konkrétní řešení jsem použil dělič s rezistory o hodnotách 47 kΩ pro horní větev a 15 kΩ pro spodní větev spojenou s referenční zemí. Pro tento dělič dostanu výsledné napětí podle rovnice 6-1, rovné 3,27 V. Samozřejmě musím uvést nějaké rozmezí, ve kterém se tato hodnota může pohybovat. Toto rozmezí je dáno nepřesností rezistorů a tolerancí napěťové reference pinu FB buck konvertoru. Tolerance rezistorů je v tomto případě 1 % z celkové hodnoty odporu a hodnota napěťové reference buck konvertoru je v rozmezí 780 V - 804 mV. Za děličem se dále nachází opět soustava kondenzátorů, která stabilizuje výstupní napětí.

Na desce se objevují 3 zdroje napájení. Jedna 3,3V hladina, která byla popsána v odstavci výše a další 2, obě o hodnotách 15 V, lišící se referenční zemí. Jedna hladina je vztažena k potenciálu 1M, tedy hlavního napájení portů, a druhá je vztažená ke 2M. Tento signál reprezentuje referenční zem pro přídavné napájení IO-Link portů. Pro realizaci těchto napětí jsem využil lineární napěťový regulátor. Vybral jsem regulátor typu BA17815T, který pracuje se vstupním napětím v rozmezí 17,5 V – 30 V a na výstupu má pevné napětí 15 V a jeho výstupní proud může být až 1 A. Na vstupu a výstupu jsou opět soustavy kondenzátorů pro stabilizaci napětí.

8.4 Návrh jednotlivých portů

8.4.1 Parametry vybraných součástek

Při návrhu portů jsem začal s výběrem vhodných tranzistorů. Tranzistory pro zkraty jsem vybíral především na základě průrazného napětí mezi kolektorem a emitorem, které je 60 V u mnou použitých tranzistorů BUK7E3R5-60E. Jsou to unipolární N-kanálové tranzistory. Při překročení tohoto napětí může dojít ke zničení tranzistoru. Hodnota tohoto parametru je zvolena s rezervou, protože pracovní hodnota napětí bude v rozmezí 18 V-30 V. Rezervu jsem volil pro případ, že se na tomto zařízení budou dělat testy EMC, zejména potom burst test. Tento test testuje zařízení, jak moc je odolné vůči rušení vyzářeném při přechodných dějích a je popsaný normou ČSN EN 61000-4-4 [20]. Proud v sepnutém stavu tranzistoru je v tomto případě irelevantní, protože v datasheetu součástky je uveden proud 120 A, zatímco zkratový proud IO-Link masteru je řádově v jednotkách A. Také mě u těchto tranzistorů bude zajímat odpor v sepnutém stavu, který je 2,6 mΩ [21]. Tento odpor definuje impedanci zkratové smyčky, kterou chci mít co možná nejmenší.

Pro tranzistory použité pro přerušování vodičů je opět důležitá hodnota průrazného napětí mezi kolektorem a emitorem, která je také 60 V, ale je zde důležitý také odpor v sepnutém stavu [21]. Odpor v sepnutém stavu je důležitý, protože používám dva tranzistory v sérii a pokud by byl tento odpor příliš velký, tak by docházelo k velkým ztrátám, tím také k nadměrnému zahřívání součástek. Pro přerušování vodičů jsem zvolil tranzistory FQU17P06TU, které mají maximální odpor v sepnutém stavu 135 mΩ a jedná se o unipolární P-kanálové tranzistory [22].

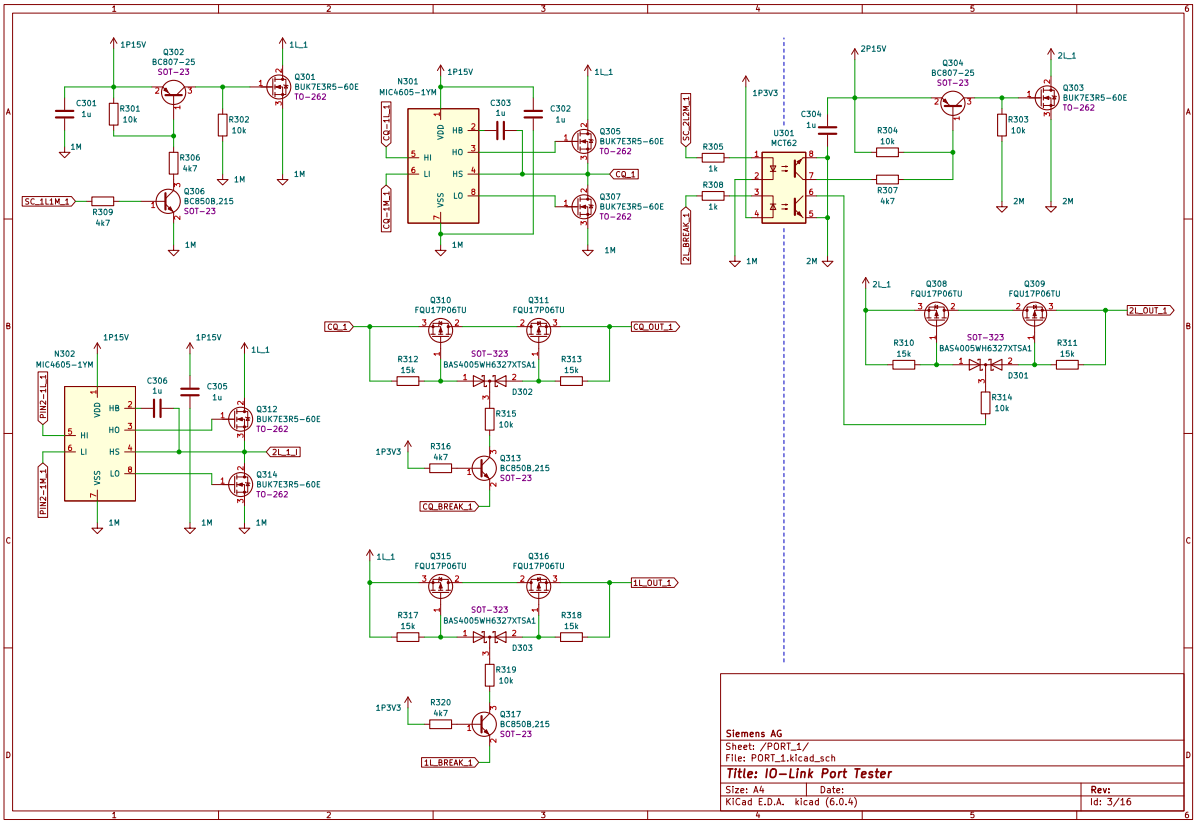
Další důležitou součástkou v návrhu portů jsou half-bridge drivery tranzistorů. Tyto drivery zajišťují rychlé a spolehlivé sepnutí tranzistorů. Já jsem vybral drivery typu MIC4605-1YM, které se napájí napětím 5,5 V-16 V. To je hlavní důvod, proč musím mít také napěťovou hladinu 15 V.

Tento budič může najednou ovládat 2 tranzistory, přičemž každý tranzistor se nechá ovládat jiným signálem. Jeho maximální výstupní proud je až 1 A. Doba nárůstu a poklesu výstupního signálu pro tranzistor je 20 ns a maximální zpoždění u této součástky je 75 ns. Tyto hodnoty jsou dostatečné pro rychlé sepnutí zkratů [16]. Rychlost vypínání není úplně podstatná hodnota, protože zkratový proud je vypnutý ochranou masteru.

Poslední kritickou součástkou v návrhu portů je 2 kanálový optický člen s tranzistorovými výstupy. Pro tuto součástku jsou kritické především hodnoty časů spínání a maximální pracovní izolační napětí. Pro izolátor MCT62 je tato izolační hladina 890 V, ovšem jedná se o špičkovou hodnotu. Časy spínání a vypínání jsou stejné – 3 µs [23].

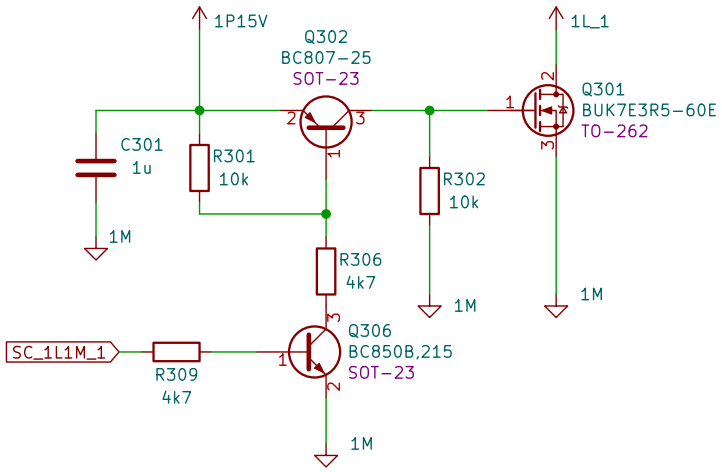
8.4.2 Popis funkčního principu návrhu

Celý návrh každého portu je na samostatné stránce elektrického schématu uvedeného v příloze *B.1*. Popis bude uveden pouze pro port 1, který je znázorněn na Obr. 8-3, ostatní porty jsou naprosto totožné. Stránka s návrhem portu je rozdělena přerušovanou čarou na dvě vzájemně izolované části. V levé části stránky se nachází vše vztažené k referenčnímu potenciálu 1M, v pravé části je zase vše s referenčním potenciálem 2M. Všechny řídící signály z výstupních shift registrů jsou vztaženy k potenciálu 1M, a proto pravá část návrhu, která je vztažena ke 2M, musí být oddělena od řídicích signálů optočlenem. Ten je na návrhu označen jako U301.



Obr. 8-3 Návrh jednoho portu Port Testeru

Prvním dílčím obvodem na schématu je obvod pro vytvoření zkratu mezi hlavním napájením IO-Link masteru, tedy signály 1L+ a 1M. Schéma tohoto obvodu je na Obr. 8-4. Budící obvod je napájen napětím o hodnotě 15 V a řídicím vstupem tohoto obvodu je signál z výstupního shift registru s názvem *SC_1L1M_1*. Tento signál řídí bázi NPN tranzistoru Q306, jehož emitor je sveden do potenciálu 1M a kolektor řídí další, tentokrát již PNP tranzistor. Tento tranzistor, označený jako Q302, je již schopný s dostatečným proudem spínat výkonový N-kanálový MOSFET tranzistor Q301. Sepnutím tranzistoru Q301 dojde již k požadované akci, ke zkratu na hlavním napájení 1L+ a 1M. Odpor R302 funguje jako pull-down rezistor. Stav, kdy není Q302 sepnutý, neznamená automaticky, že tam neteče žádný proud, ale mohou tam vznikat malé proudy, které jsou svedeny do země právě přes pull-down rezistor [24]. Primární funkcí tohoto odporu je však vybití kapacity gate tranzistoru Q302 při vypnutí. Ostatní rezistory jsou v obvodu zejména pro omezení proudu tekoucího obvodem. Bypass kondenzátor C301 snižuje impedanci napájení 1P15V a tím umožňuje rychlé sepnutí tranzistoru Q301. Další funkcí tohoto bypass kondenzátoru je, že snižuje šum napájecího napětí. Zjednodušeně by se dalo říct, že všechny střídavé složky tento kondenzátor zkratuje na zem [25].



Obr. 8-4 Návrh obvodu pro zkrat mezi 1L+ a 1M

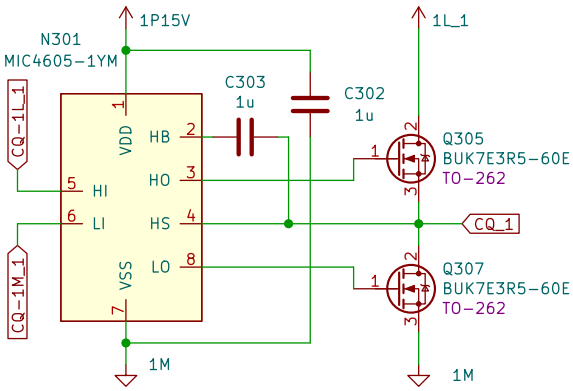
Dalším požadavkem bylo zkratování komunikačního signálu C/Q na 1L+ a 1M. Toto schéma je uvedeno na Obr. 8-5. Opět je to řešeno přes N-kanálové MOSFET tranzistory. Tentokrát jsou zde však použity 2 tranzistory. Jedním se zkratuje C/Q signál na 1L+ a druhým zase na 1M. Pro tyto 2 tranzistory mi již přišlo vhodné použít half-bridge driver. Popis funkce těchto součástek je uveden v části **Chyba! Nenalezen zdroj odkazů.**. Vstupními signály tohoto budiče a zároveň c elého obvodu jsou signály z výstupních shift registrů, které jsou na hladině 3,3 V. Mezi piny 1 a 7 integrovaného obvodu budiče je přivedeno napětí 15 V signály nazvanými 1P15V a 1M. Mezi těmito piny je kondenzátor C302, který zde má podobnou funkci jako kondenzátor C301 na Obr. 8-4, tedy vyhlazení vstupního napájení integrovaného obvodu. Kondenzátor C303 je zde přidán na základě datasheetu budiče. Je zapojen za diodou, která je již součástí integrovaného obvodu a jedná se o tzv. boost kondenzátor, na kterém je udržováno napětí potřebné pro sepnutí tranzistoru Q305. V případě že se tento kondenzátor vybije, tak se tranzistor Q305 zavře [16].

Pokud je sepnutý tranzistor Q307, pak je signál C/Q, zkratován na 1M. V tuto chvíli umožňuje interní dioda nabití kondenzátoru C303 na vstupní napětí snížené o hodnotu prahového napětí diody. Když tranzistor Q307 vypne a pin HO sepne, tak je napětí z kondenzátoru aplikováno na gate horního tranzistoru Q305. Napětí na pinu HS roste až do hodnoty napětí signálu 1L_1, které je v tomto případě 24 V. S rostoucím napětím na pinech HS a HB je vnitřní dioda v závěrném směru

a chrání kondenzátor před vybíjením. Signál na pinu HI, pojmenovaný jako *CQ-1L_1* ovládá tranzistor Q305 a signál s názvem *CQ-1M_1* zase ovládá tranzistor Q307 [16].

Tento obvod může fungovat ve 4 stavech. První stav je HI ve stavu logické 0 a LI ve stavu logické 1. V tomto případě je tranzistor Q307 sepnutý a Q305 není sepnutý. To znamená, že komunikační signál C/Q bude zkratovaný na 1M. Druhým stavem je stav opačný. Tedy HI je ve stavu logické 1, tento případ znamená, že je C/Q signál zkratovaný na 1L+. Dalším případem, ve kterém bude tento obvod fungovat je stav, kdy jsou oba vstupní piny ve stavu logické 0. V tomto případě není signál C/Q zkratován nikam a pokračuje do obvodu, ve kterém se tento signál přerušuje. A posledním stavem, ve kterém se může tento obvod nacházet je, že oba vstupní signály budou v hodnotě logické 1. Tento stav odpovídá zkratování 1L+ s 1M, ale zároveň také C/Q. Tento stav není žádoucí.

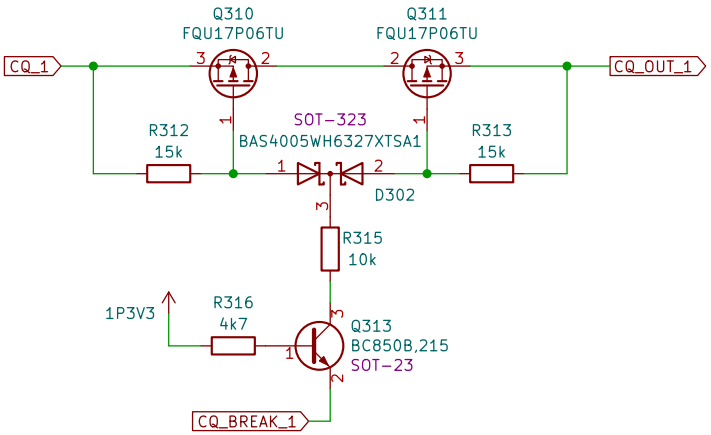
Obvod pro zkratování komunikačního signálu na 1L+ resp. 1M je naprosto stejný jako obvod pro zkratování pinu 2. Jediný rozdíl je, že místo komunikačního signálu je do tohoto obvodu přiveden pin 2 IO-Link masteru. Pin 2 musí být v tomto případě nakonfigurován jako DI nebo DQ (port class A).



Obr. 8-5 Návrh obvodu pro zkratování C/Q na 1L+ a 1M

Základem obvodu pro přerušování komunikace (wirebreak) jsou dva anti sériově zapojené tranzistory. Další součástí tohoto obvodu je dále budící obvod pro spínání těchto tranzistorů. Celé schéma tohoto obvodu je na Obr. 8-6. Konkrétně se jedná o přerušování komunikačního C/Q kanálu. Toto přerušení se vyvolává tranzistory Q310 a Q311. Celý obvod je řízen signálem *CQ_Break_1* a wirebreak se v tomto případě aktivuje vypínáním tranzistorů, nikoli jejich spínáním. Toto jsem provedl kvůli logice signálu, když je aktivovaná diagnostika, tak by měl být signál potřebný k její aktivaci rovný 1. Takže pokud nebude wirebreak aktivovaný, tak je *CQ_Break_1* v logické nule, a tudíž tranzistor Q313 je sepnutý a budí tranzistory Q310 a Q311. Odpory R312 a R313 spolu s R315 tvoří napěťový dělič, aby nebylo překročeno maximální napětí pro gate tranzistoru. Skrze R312 a R313 se vybíjí gate tranzistorů, aby mohlo dojít k jejich zavření. Také fungují jako proudové omezení.

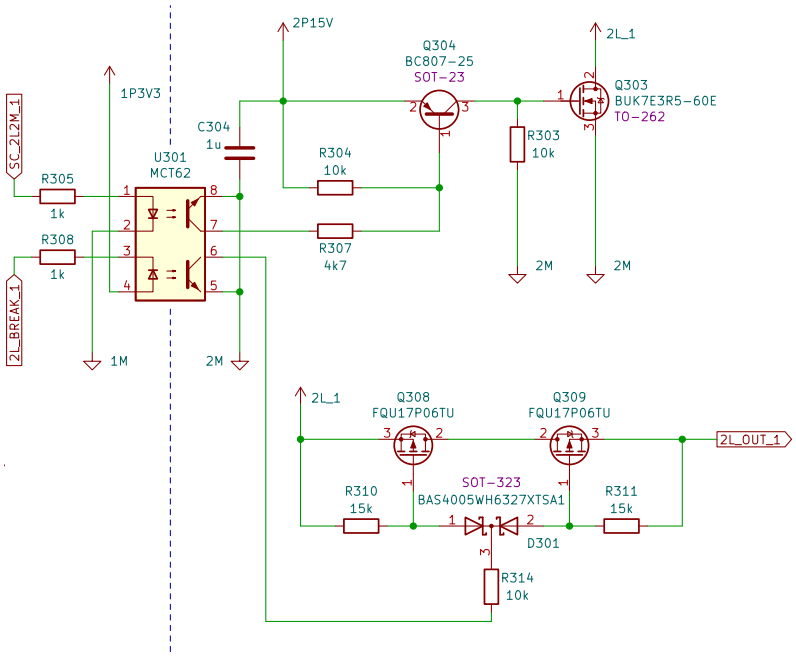
V případě, že aktivuji wirebreak, tak vypne tranzistor Q313, protože mezi bází a emitorem tranzistoru bude nulové napětí. Tím ztratím napětí i mezi elektrodami gate a source MOSFET tranzistorů, proto se zavřou.



Obr. 8-6 Návrh obvodu pro zkratování přerušování C/Q (1L+)

Poslední 2 obvody se týkají pinu 2, který je referován k potenciálu 2M. Proto musí být mezi hlavním obvodem a vstupními signály optočlen, který má tranzistorový výstup. Jeden obvod pro zkratování přídavného napájení (2L+ a 2M) a druhý obvod pro přerušování signálu na pinu 2. Tyto obvody jsou stejné jako pro pin 1 (1L+), pouze je zde již zmíněný optočlen. Obvody pro pin 2 jsou znázorněny na Obr. 8-7.

Zkrat pinu 2 na 2M se provádí tranzistorem Q303. Tento obvod má opět na vstupu řídicí signál z shift registru, který ovšem v tomto případě musí jít přes optočlen U301. Použitý optočlen má tranzistorový výstup, který v tomto případě nahrazuje tranzistor Q306 na Obr. 8-4. Dále už je tento obvod stejný. To samé platí pro obvod přerušování pinu 2, které se provádí anti sériově zapojenými tranzistory Q308 a Q309. Tranzistor na výstupu optočlenu nahrazuje tranzistor Q313 uvedený na Obr. 8-6.



Obr. 8-7 Návrh obvodu zkratu mezi 2L+ - 2M a přerušení 2L+

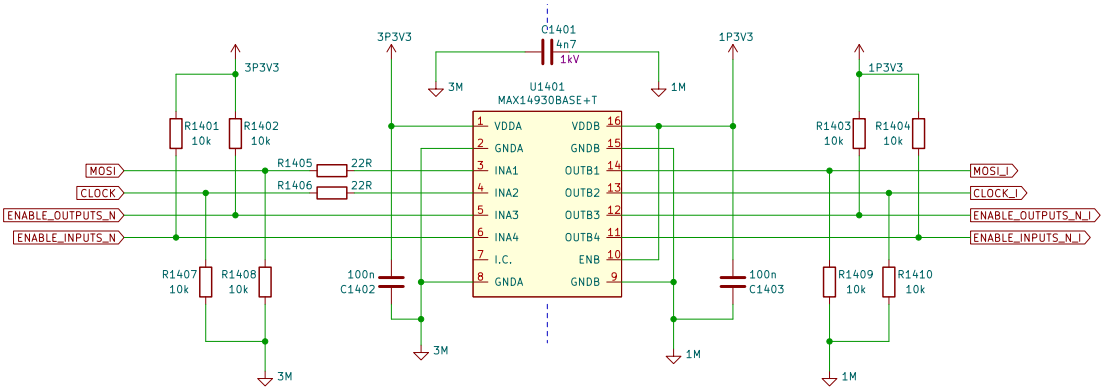
8.5 Izolace

Velice důležitou součástí návrhu je také návrh izolace signálů, které jsou nějakým způsobem spojené s Raspberry Pi, protože Raspberry Pi má odlišný referenční potenciál, ve schématu označený jako 3M.

Na Obr. 8-8 je vidět zapojení digitálního izolátoru U1401 pro izolaci SPI signálů (*MOSI*, *CLOCK*) a 2 GPIO signálů (*ENABLE_OUTPUTS_N, ENABLE_INPUTS_N*). Signály vztažené k potenciálu 3M jsou používané Raspberry Pi a jsou ve schématu nazvány pouze svým názvem podle funkce (např. *MOSI*). Signály rozvedené po DPS jsou vztaženy k potenciálu 1M a jsou pojmenovány také svým názvem, za kterým je přidané I, značící izolovaný signál (např. *MOSI_I*). Obvod na Obr. 8-8 je rozdělen na dvě části modrou přerušovanou čarou, značící izolaci. Část obvodu vpravo od izolátoru je výstupní stranou a je celá vztažená k potenciálu 1M. Tyto signály tedy mohou být rozvedeny po desce plošných spojů. Levá část obvodu od izolátoru je napájena stejně jako pravá část obvodu 3,3 V, ale tentokrát vztaženými ke 3M. Toto napájení je vyvedeno z Raspberry Pi. Signály na této straně izolátoru jsou generovány Raspberry Pi a izolátor je zpracovává.

Jak je vidět z Obr. 8-8Obr. 8-8, tak SPI signály jsou specifické tím, že se k nim přidává sériový odpor R1405, resp. R1406. Tento odpor je zde z důvodu potlačení odrazů na vedeních při vysokých frekvencích. Obvod také obsahuje 3 kondenzátory - 2 bypass kondenzátory a jeden vysokonapěťový. Vysokonapěťový kondenzátor je zde zapojen z důvodu potlačení rušení mezi

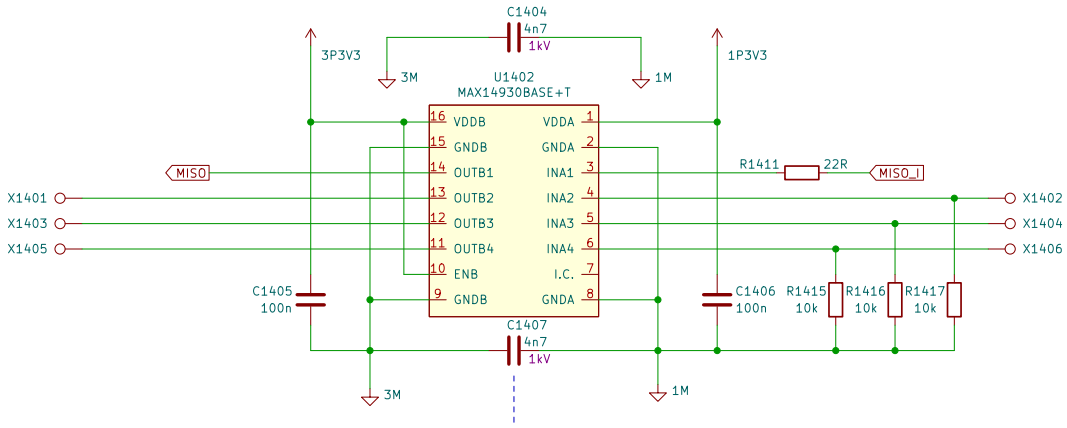
jednotlivými izolátory. Obecně jsem v návrhu postupoval tak, aby mezi každými 2 izolátory byl jeden vysokonapěťový kondenzátor. Důležitou součástí návrhu izolace jsou také pull-up resp. pull-down rezistory. Pull-up rezistory jsem přidal k signálům, které budu pro jejich aktivaci spínat do hodnoty logické 0, tzn. většinu času budou mít hodnotu logické 1. Pokud si představím GPIO periferii mikrokontroléru jako spínač, tak to vlastně znamená, že tento signál připínám k referenční zemi napájení. Pokud je tento signál rozpojen tak na vstupu izolátoru není automaticky logická 1, ale náhodná hodnota. Přidáním pull-up rezistoru do obvodu docílím toho, že na vstupu izolátoru bude skutečně logická 1, nikoliv náhodná hodnota [24]. To samé platí pro pull-down rezistory, které jsem ale přidal k signálům, které se aktivují přepnutím do logické 1. Jejich funkce je opačná než funkce pull-up rezistorů – definují stav v hodnotě logické 0.



Obr. 8-8 Návrh izolace SPI signálů

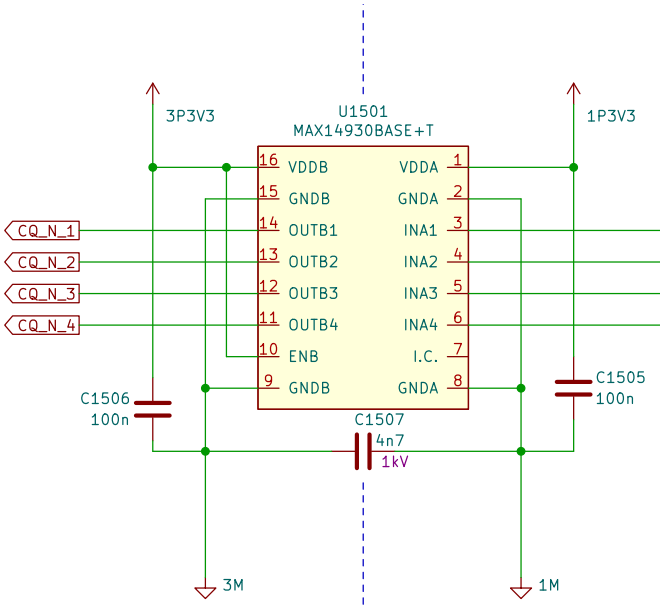
Stejným způsobem jsou řešeny i další signály použité pro řízení tohoto port testeru, které jsou vedeny přes izolátory U1402 a U1403. Digitální izolátor U1403 je orientován ve stejném směru jako U1401 a jsou k němu připojeny 3 signály (*MASTER_RESET_N, TRIGGER_IN_N, TRIGGER_OUT*) a jeden tzv. test point – nejedná se o klasickou součástku, ale značí pouze to, že je na desce odmaskovaná vodivá ploška, na které může být později připájen další signál nebo součástka, případně může sloužit také k testování.

Drobná změna je u izolátoru U1402, jehož zapojení je znázorněno na Obr. 8-9. Tento izolátor je orientován v opačném směru – výstup je orientovaný ve směru do Raspberry Pi a jedná se o signál MISO. Další 3 kanály jsou obsazeny test pointy.

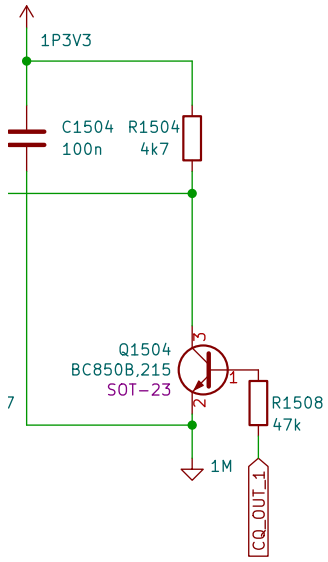


Obr. 8-9 Návrh izolace MISO signálu

Dalším požadavkem bylo vyvedení znegovaných C/Q signálů na pin header. Tento pin header má ovšem referenci 3M stejně jako Raspberry Pi. Proto se tyto signály také musí izolovat. Návrh řešení negace těchto signálu je vidět na Obr. 8-11. Pokud se podívám na funkci tohoto obvodu, tak je vidět, že tranzistor Q1504 bude sepnutý, pokud bude mezi bází a emitorem rozdíl potenciálů. Tento rozdíl potenciálů vznikne, pokud hodnota signálu *CQ_OUT_1* bude v logické 1. Tranzistor sepne a připojí potenciál 1M ke vstupu izolátoru. To znamená logickou 0. Naopak, pokud bude sledovaný signál v logické 0, tak bude tranzistor Q1504 rozepnut a přes odpor R1504 bude na vstup izolátoru přivedeno napájecí napětí – logická 1. Tím je hotová negace signálu. Součástí obvodu pro negování signálu je také bypass kondenzátor. Stejný způsobem se znegují a odizolují i ostatní komunikační signály.



Obr. 8-10 Návrh izolace C/Q signálů



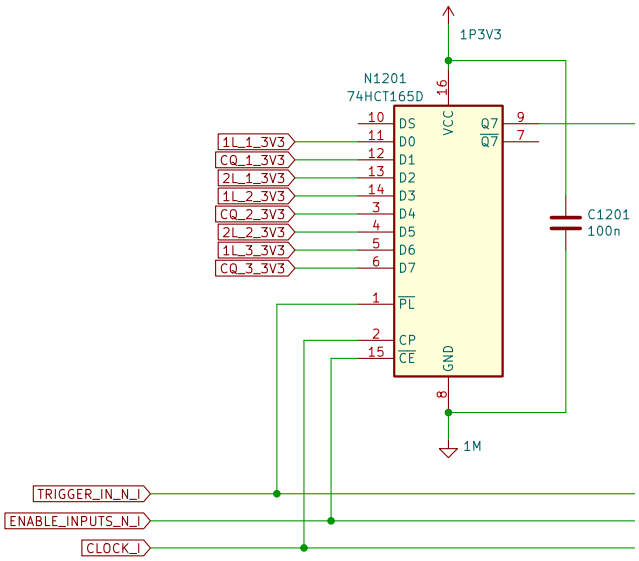
Obr. 8-11 Návrh negování C/Q signálu

8.6 Realizace vstupů a výstupů

Jedním z požadavků na návrh tohoto port testeru byla i možnost sledování vybraných signálů v Raspberry Pi. Tyto sledované signály budou – hlavní napájení 1L+, pin 2 a komunikační vodič C/Q. Abych je mohl posílat do Raspberry Pi, tak potřebuji tyto signály převést z paralelních signálů na sériové. Tuto změnu je možné udělat pomocí vstupních shift registrů, které vstupní – paralelní signály převádí na jeden sériový signál. Pro mou aplikaci jsem využil vstupní shift registry 74HCT165D, které mají možnost 8 vstupních signálů. Protože potřebuji sledovat 3 signály z každého portu, tak budu potřebovat dohromady pro 8 portů 3 shift registry. Ukázka zapojení prvního shift registru je na Obr. 8-12.

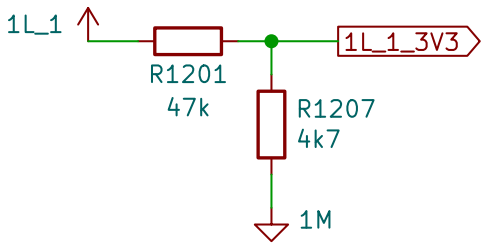
8.6.1 Realizace vstupních signálů

Vstupní signály se přivádí na piny označené D0 až D7. Shift registry se napájí z 3,3 V, které se přivádí na pin VCC a referenční potenciál je zase přiveden na GND. Mezi těmito piny se nachází bypass kondenzátor C1201. Signál přivedený na pin $\overline{PL}$ je pro přečtení paralelních vstupů. V momentě, kdy je tento signál přepnutý do logické nuly, se přečtou aktuální stavy paralelních signálů na vstupu. Tento pin se řídí signálem *TRIGGER_IN_N_I*. Dalšími důležitými signály pro SPI komunikaci jsou signály *ENABLE_INPUTS_N_I* přivedený na pin $\overline{CE}$ a *CLOCK_I* přivedený na pin CP. Přepnutím signálu na pinu $\overline{CE}$ do hodnoty logické 0 se povolí zapsání hodnot do sériového výstupu. Zápis paralelních vstupů do sériového výstupu probíhá v synchronizaci se signálem *CLOCK_I*. S každou náběžnou hranou tohoto signálu se přečte další vstup. A protože používám 3 shift registry, tak je musím spojit za sebe. Toto spojení vypadá tak, že signál, který vede z pinu Q7 a představuje sériový výstup, přivedu do pinu DS následujícího shift registru. Pin DS označuje sériový vstup a spojuje sériové signály všech shift registrů. Signál z posledního shift registru z pinu Q7 se je označen jako *MISO_I*. Tento signál přivádí přes izolátor do Raspberry Pi informace o vstupech. Pin $\overline{Q7}$ jsem nechal nezapojený, protože nebude využit. Tento signál je pouze znegovaný signál Q7 [26].



Obr. 8-12 Návrh aplikace vstupních shift registrů

Důležitá věc, kterou jsem musel zohlednit, je fakt, že všechny sledované signály jsou signály na hladině 24 V, zatímco shift registr je pouze na 3,3 V. Toto mě vedlo k použití napěťového děliče, který je znázorněn na Obr. 8-13. Dále je také uvedena rovnice (8-1) pro výpočet výstupního napětí odporového děliče. Výstupní napětí je rovné hodnotě 2,18 V. Tato hodnota je dostatečná pro shift registr [26]. Při návrhu děliče jsem musel zohlednit, že i při minimální povolené hodnotě napětí (18 V) musí mít signál dostatečnou velikost, aby ho shift registr rozpoznal. Naopak pokud bude maximální napájení na masteru (30 V), tak hodnota výstupního napětí děliče nesmí přesáhnout 3,3 V. Při překročení této hodnoty může dojít ke zničení shift registru.

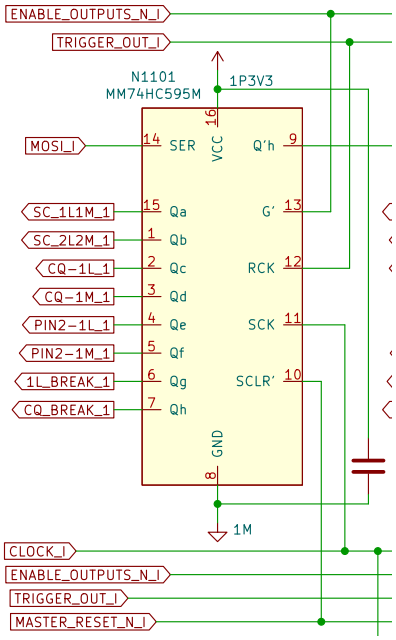


Obr. 8-13 Návrh napěťového děliče pro snížení napětí sledovaných signálů

$$U_2 = U * \frac{R_2}{R_1 + R_2} = 24 * \frac{4700}{47000 + 4700} = \mathbf{2,18\ V} \tag{8-1}$$

8.6.2 Realizace výstupních signálů

Abych mohl použít výstupní signály z Raspberry Pi, tak musím nějakým způsobem převést signál ze sériového na paralelní signály. Raspberry Pi posílá signál *MOSI*, který obsahuje data pro ovládání port testeru. Tento signál je sériový a prochází přes izolátor. K převodu na paralelní signály použiji opět shift registry, ale tentokrát výstupní. Tyto shift registry převádí sériový signál na několik paralelních signálů. Sériový signál se dělí podle signálu *CLOCK*. S každou náběžnou hranou se zapíše další signál. Izolovaný signál *MOSI_I* vstupuje do prvního shift registru na pin SER, který funguje jako sériový vstup. Napájecí piny jsou označeny stejně jako v případě vstupních shift registrů – VCC a GND. Opět je mezi nimi bypass kondenzátor. Tyto součástky jsou také napájeny z hladiny napětí 3,3 V. Piny s paralelními výstupy jsou označeny jako Qa až Qh, tyto signály již přímo ovládají obvody na desce plošných spojů, jejichž funkčnost byla popsána v kapitole 8.4.2. Signál MOSI_I bude mít velikost 9 bajtů. V prvním shift registru se zpracuje pouze jeden bajt informace a zbytek informací se posune do dalších shift registrů přes pin sériového výstupu, označeného jako Q'h. Tento shift registr má také 3 ovládací piny a pin pro přivedení signálu *CLOCK_I*. Pin s názvem $\overline{G}$ je velice důležitý a pokud tento bit bude mít hodnotu logické 1, tak shift registr bude ve stavu tzv. vysoké impedance, tento stav je popsán v kapitole 5.2.2. Pokud bude mít pin $\overline{SCLR}$ hodnotu logické 1, tak se celý shift registr vymaže. Na pin SCK je připojen signál *CLOCK_I* a konečně s náběžnou hranou pinu RCK se propisuje paměť shift registru na fyzické výstupy. Na Obr. 8-14 jsou vidět názvy signálů, které jsou připojeny na jednotlivé piny [14].



Obr. 8-14 Signálové provedení výstupního shift registru MM74HC595

KAPITOLA 9: NÁVRH DESKY PLOŠNÉHO SPOJE

9.1 Deska plošného spoje

Z důvodu velkého množství součástek v mnou vytvořeném elektrickém schématu, jsme po dohodě s vedoucím zvolili variantu 4vrtstvé DPS. Tato deska má kromě standardních 2 krajních vrstev, také 2 vrstvy, které se nachází uvnitř. Po těchto vnitřních vrstvách lze rozvádět pouze signály, nelze na ně umisťovat součástky.

Návrh jsem nejprve zkoušel vyřešit s 2vrstvou deskou, ale v momentě, kdy jsem začal rozvádět 72 signálů ze signalizačních LED diod, tak rozměry desky začaly výrazně růst. Z tohoto důvodu jsem zvolil 4vrstvou desku. Na této desce jsem mohl dlouhé signály, které vedou přes celou desku, vést ve vnitřních vrstvách a rozměr desky tolik nerostl.

Celý návrh DPS jsem se snažil koncipovat tak, aby byly všechny hlavní logické celky u sebe, především zapojení jednotlivých portů, napájení, izolace a signalizační LED. Ostatní části (např. jednotlivé odporové děliče) jsem umístil do volných míst, abych co nejvíce zmenšil rozměr desky.

Na začátku jsem si nastavil pravidla pro spoje a prokovy. Zvolil jsem 2 tloušťky spojů - 0,25 mm, resp. 1 mm, k nim odpovídající prokovy – 0,8 mm (z toho 0,4 mm díra) resp. 1,4 mm (0,8 mm). Dále jsem si musel definovat potřebné izolační vzdálenosti mezi signály. Izolační mezeru mezi signály vztažené ke stejnému referenčnímu potenciálu jsem volil 0,2 mm a izolační mezeru mezi signály odlišných potenciálů jsem volil 0,5 mm.

Při návrhu konkrétních částí jsem postupoval tak, aby byly, pokud možno co nejvíce zachovány logické cesty signálů. To znamená, že u vstupních svorek je hned napájení, vedle napájení zase vstupní a výstupní shift registry, které následně ovládají jednotlivé porty. Za zapojením portů se už nachází výstupní svorky. Napěťové děliče používané v obvodu jsem se snažil umisťovat vždy do volných míst, hned vedle daného signálu.

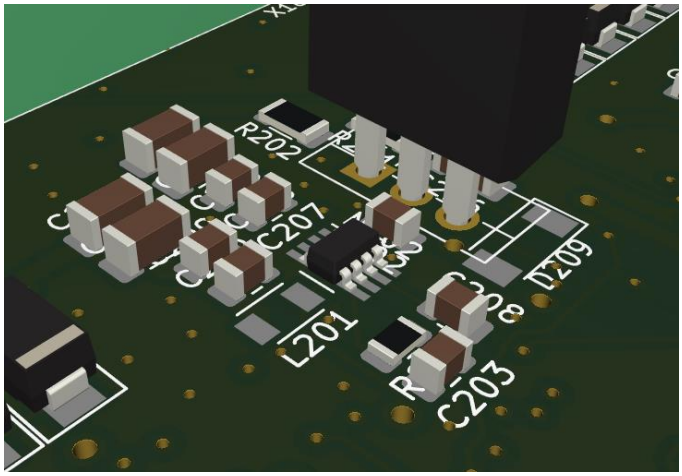
V místech desky, kde nejsou součástky nebo signály je vrstva mědi s potenciálem 1M. Toto se běžně dělá z důvodu lepší cesty signálů a snazšího uzavírání signálových cest. Jsou tam 2 výjimky. V místě mezi pin headerem pro připojení Raspberry Pi a izolátory se nachází zóna s potenciálem 3M, protože Raspberry Pi má tento referenční potenciál. A další zónou je 2M, která se nachází v části portů napravo od izolátoru, protože jsem port navrhl tak, aby vše, co je na potenciálu 2M bylo vpravo od izolátoru.

9.2 Řešení regulátoru pro napájení

Při návrhu regulátoru pro napájení desky jsem postupoval podle datasheetu, protože u tohoto návrhu je velmi důležitá vzdálenost určitých signálů a součástek. Pokud bych se tohoto postupu nedržel, tak by pravděpodobně regulátor fungoval také, ale měl by výrazně horší vlastnosti. Výstupní napěťový dělič jsem umístil blízko konvertoru, protože tato signálová cesta nesmí být rušena. Plocha tohoto děliče musí být co nejmenší a signály co nejkratší.

Lineární napěťové regulátory nevyžadují žádné speciální zapojení na desce plošných spojů, takže jsem je umístil poblíž buck konvertoru, abych měl celé napájení desky v jednom místě. Pouze jsem se při návrhu držel toho, abych měl kondenzátory blízko u konvertoru.

Na Obr. 9-1 je vidět snímek z 3D náhledu port testeru, na tomto snímku je buck konvertor, konkrétně se jedná o integrovaný 8pinový obvod v pouzdře SOT-23 a k němu zapojené pasivní součástky. Největší součástka na tomto obrázku je lineární regulátor, jedná se THT součástku v pouzdře TO-220F.

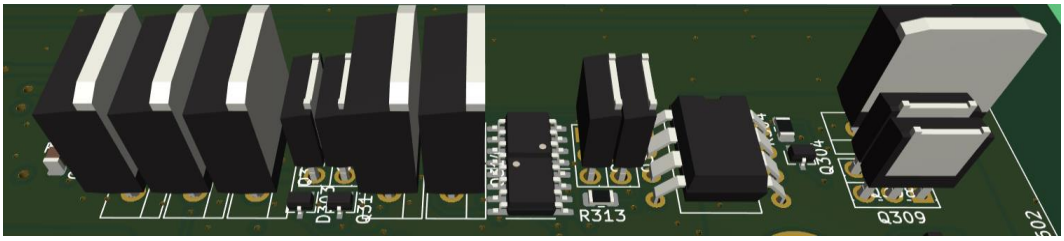


Obr. 9-1 Detail řešení buck konvertoru

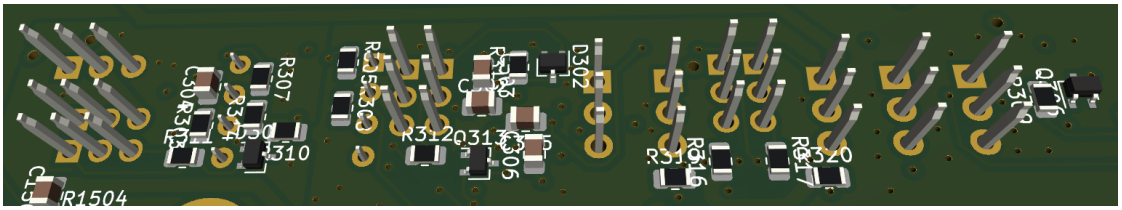
9.3 Layout portů

V tomto případě nebylo potřeba se držet nějakých speciálních pravidel nebo datasheetů. Ale zároveň byl návrh portů nejvíce kritickou částí celé DPS, protože ve výsledku se jedná o celkem velký obvod a celkově jich je na jednom port testeru 8. Takže bylo na místě si dát záležet na návrhu portů, aby byly co nejvíce kompaktní. Důležitou věcí bylo si nechat místo v návrhu pro přivedení řídicích a napájecích signálů k jednotlivým portům, pro každý port se jednalo o 9 řídicích, 5 vstupních a 5 výstupních signálů.

Prvním krokem bylo umístění THT součástek, v tomto případě jde pouze o tranzistory. Na Obr. 9-2 je vidět náhled zapojení vrchní vrstvy a na Obr. 9-3 zase zapojení spodní vrstvy jednoho portu. Z Obr. 9-2 je vidět, že THT součástky zabírají nejvíce místa. Důležitým místem je to, kde se nachází jeden 8pinový integrovaný obvod – optočlen v THT pouzdře DIP8. Celá část nalevo od optočlenu je totiž vztažená k potenciálu 1M, a naopak část vpravo je vztažena k potenciálu 2M. Toto oddělení je důležité opět z pohledu rušení a dalších nežádoucích jevů. V zapojení jsou 2 typy THT tranzistorů. Jedná se o tranzistory, které zajišťují zkraty – BUK7E3R5, v pouzdře TO-262 a tranzistory přes které se dělá wirebreak – FQU17P06TU, v pouzdře TO-251. Na Obr. 9-3, který ukazuje spodní stranu desky, je vidět, že SMD součástkami vyplňuji prázdná místa mezi velkými THT součástkami.



Obr. 9-2 3D náhled zapojení jednoho portu – přední strana DPS



Obr. 9-3 3D náhled zapojení jednoho portu – zadní strana DPS

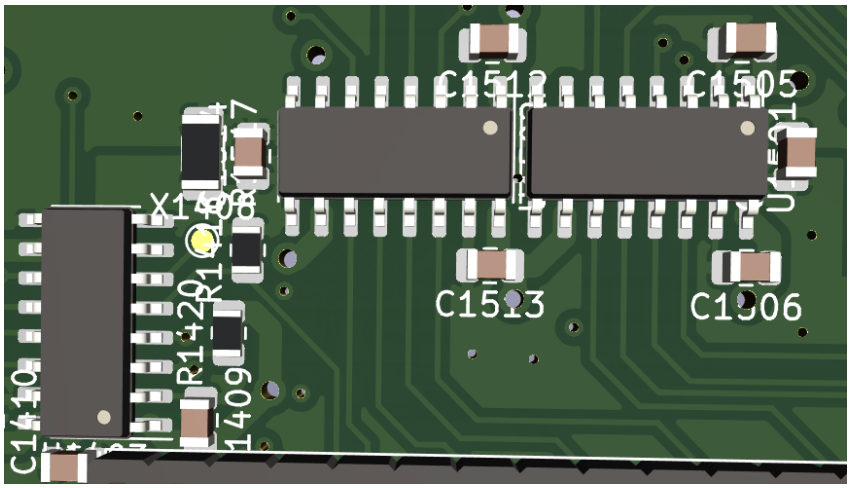
9.4 Návrh signalizačních LED diod

Tato část DPS zabírá velký prostor, protože se jedná o 72 LED diod a ke každé je navíc připojen odpor. Obě tyto součástky jsou v pouzdře 0805. Dalším problémem je, že jsem musel přivést signály z jednotlivých shift registrů na všechny LED. Tyto signály jsem vyvedl převážně z prostoru mezi shift registry a jednotlivými porty. Rovnice (9-1) vyjadřuje postup při návrhu odporu, který je zapotřebí pro omezení proudu tekoucího LED diodou. Podle datasheetu LED diody musí být proud v propustném směru maximálně 2 mA [27]. Proto jsem zvolil rezistor o velikosti odporu 1 kΩ. Pro tuto velikost odporu podle rovnice (9-1) vychází proud v propustném směru LED diodou 1,45 mA.

$$I_{LED} = \frac{U_{sig} - U_{LED}}{R} = \frac{3{,}3 - 1{,}85}{1000} = 1{,}45\ mA \qquad (9\text{-}1)$$

9.5 Návrh izolace

Všechny použité izolátory se nacházejí v prostoru u pin headeru pro připojení Raspberry Pi. Zvolil jsem toto umístění, protože všechny signály přivedené na pin header musí projít izolátorem, než budou použity na DPS. Nejlepší by bylo, kdyby byly v jedné řadě. Bohužel jsem už pro tuto variantu nenašel na desce místo. Proto jsem zvolil alternativní variantu 2 izolátory z přední strany a 2 přesně pod nimi ze spodní strany. Poslední izolátor se nachází otočený o 90° vedle nich. Všechny izolátory jsou orientovány tak, aby v prostoru mezi izolátory a pin headerem byly „neizolované“ signály vztažené k potenciálu 3M. Bypass kondenzátory jsou umístěné co nejblíže k napájecím svorkám izolátoru a vysokonapěťové kondenzátory, použité proti rušení, jsou zase nad, resp. pod izolátorem.



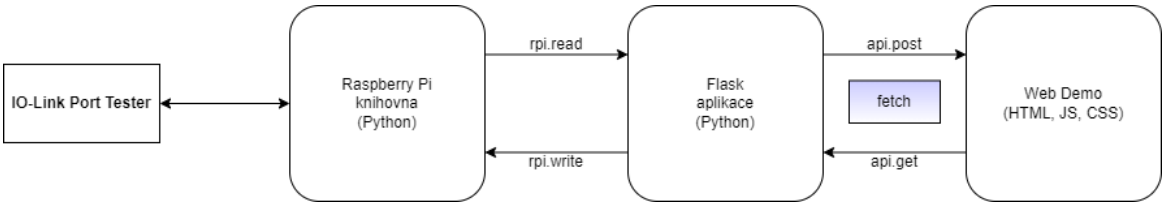
Obr. 9-4 Detail provedení izolace – přední strana

KAPITOLA 10: SOTFWAROVÁ ČÁST NÁVRHU IO-LINK PORT TESTERU

10.1 Odladění programu pomocí logického analyzátoru bez prototypu

V této části návrhu jsem se zaměřoval především na propsání potřebných dat z webového GUI až do knihovny pro ovládání Raspberry Pi. Dále také samozřejmě na tvorbu webového GUI.

Na Obr. 10-1 je vidět struktura provedení softwarové části port testeru. Celé ovládání bude z webového GUI, odkud bude přes příkaz fetch probíhat komunikace se softwarovou knihovnou ovládající Raspberry Pi.



Obr. 10-1 Struktura softwarové části port testeru

10.1.1 Webové GUI

Grafický náhled rozhraní je na Obr. 10-2. Jedná se o matici talčítek, ve které jednotlivé sloupce vyjadřují signály a řádky zase porty. V dolní části obrazovky jsou umístěna 2 žlutá tlačítka, která jsou zde pro odesílání dat do Raspberry Pi (*SUBMIT*) a pro vyčtení aktuálně nahraných hodnot (*GET ACTUAL STATUS*). Pod každým tlačítkem je ještě malý obdélník, který funguje jako kontrola signalizace zapnutých signálů. Pokud je tento obdélník žlutý, pak je signál aktivní a pokud šedý, tak není aktivní. Zelená tlačítka zase znamenají aktuálně vybrané signály, které ještě nejsou nahrány do Raspberry Pi. Pokud bych klikl na tlačítko *SUBMIT*, tak by se na port testeru deaktivovaly všechny aktuálně aktivní signály a aktivovaly by se pouze 3 signály, které jsou vybrané – *SC 1L-1M* pro porty 2, 3 a 4. Webové rozhraní jsem psal pomocí jazyka HTML a soubor s tímto GUI se jmenuje *GUI.html*. Tento soubor musí být umístěn ve složce templates, protože při vytváření API (v programu *flask_app.py*) volám toto grafické rozhraní pomocí příkazu `render_template('GUI.html')` a ten odkazuje do složky templates. Funkce používané v GUI jsou v souboru command.js, který je psán v jazyce JavaScript, dalším volaným souborem v GUI je soubor s názvem style.css, který definuje vzhled jednotlivý objektů v GUI na základě tzv. class, kterou můžu přiřadit každému objektu.

Obr. 10-2 Náhled webového GUI použitého pro ovládání port testeru

V následující ukázce kódu je ukázán vzor pro vytvoření tlačítek ze souboru *style.css*. Tato část kódu definuje, jak mají vypadat tlačítka, které mají *class="btn-group"*. Jsou zde nadefinovány např. barva a velikost písma apod.

```
.btn-group button {
  border: 1px solid green;
  color: white; /* White text */
  padding: 10px 24px; /* Some padding */
  cursor: pointer; /* Pointer/hand icon */
  float: left; /* Float the buttons side by side */
  margin-right: 20px;
  font-size: 18 px;
  border-radius: 100 px;
}
```

V další ukázce je nadefinována určitá oblast stránky, která má *class="btn-group"* a tudíž všechny tlačítka vytvořené v této oblasti budou mít vzhled definovaný výše. Oproti základním vlastnostem je zde navíc uvedena velikost tlačítka a přesné umístění na stránce. Dále jsem každému tlačítku přiřadil parametr s názvem id, který musí být unikátní pro každý objekt na stránce. Další vlastností, co bude mít toto tlačítko je, že při kliknutí se spustí funkce `changeValue(btn)`, jejíž parametr je id tlačítka. Tato funkce bude pospána dále. Tímto

způsobem je vytvořeno všech 72 tlačítek. A podobným způsobem jsou vytvořeny i ostatní objekty a styly na stránce.

```
<div class="btn-group">
    <button style="width:150px; height:40px; position:absolute; top:50px;
left:70px;" id="0" class="btn_off" onclick="changeValue('0')">SC 1L-
1M</button>
```

V této části popíšu podrobněji soubor command.js, ve kterém jsou nadefinovány 3 funkce – `changeValue(btn)`, `submit()` a `get_status()`.

Funkce `changeValue(btn)`, jejíž parametr je id tlačítka mění hodnotu parametru class mezi "*btn_off*" a "*btn_on*". Na začátku si do proměnné *btn_prop* uložím informace o tlačítku. Potom už následují samotné změny hodnot. Návratovou hodnotou této funkce je zapsaný parametr class.

```
function changeValue(btn){
    btn_prop = document.getElementById(btn);
    if (btn_prop.className == "btn_off"){
        btn_prop.className="btn_on";
    } else if(btn_prop.className == "btn_on"){
        btn_prop.className="btn_off";
    };
    return btn_prop.className;
}
```

Další funkcí je funkce submit, pomocí které se odesílají data do Raspberry Pi přes API. Na začátku vytvořím proměnnou object, tato proměnná obsahuje list dvojic – id a hodnota. Při volání funkce submit se nejdříve pomocí for smyčky se zkontrolují parametry class u všech tlačítek a na základě těchto parametrů se do listu a příslušného id zapíše hodnota 1 nebo 0.

Následuje již odeslání hodnot přes metodu *POST* do Raspberry Pi. Funkce fetch definuje strukturu odesílaných dat. Hlavní částí budou samozřejmě samotná data, která se pomocí příkazu `JSON.stringify(object)` změní z proměnné typu JavaScript object na JSON string. Tento formát je běžně používán k výměně dat přes API a je snadno čitelný. Návratovou hodnotou jsou odeslaná data.

```
var object = {"0":0,"1":0,"2":0,"3":0, …, "71":0}
async function submit(){
    for (let i = 0; i < 72; i++){
    btn_class = document.getElementById(i).className;
    if (btn_class == "btn_off"){
        object[i] = 0;
    } else if (btn_class == "btn_on"){
        object[i] = 1;
    };
    };
    const response = await fetch('/api', {
        method: 'POST',
        body: JSON.stringify(object),
        headers: {
        'Content-type': 'application/json'
        }
    });
    return object;
}
```

Poslední funkcí napsanou v programu *command.js* je funkce `get_status()`, která čte zapsaná data do Raspberry Pi. Opět používám příkaz *fetch*, ale tentokrát pouze s parametrem cesty k API, odkud se mají data číst. Ve druhé části této funkce se už jen projde celý seznam hodnot a na základě hodnoty se nastaví parametr class na ACTIVE nebo INACTIVE. A na základě tohoto příznaku se obdélník pod tlačítkem zbarví žlutě, resp. šedě.

```
async function get_status(){
    const response = await fetch('/api');
    const actual_data = await response.json();
    console.log(actual_data);
    x = JSON.stringify(actual_data);
        for(let j = 0; j < 72; j++){
        btn_val = actual_data[j];
        elem_prop = document.getElementById(j+100);
        if(btn_val == 1){
            elem_prop.className="ACTIVE"
        } else{
            elem_prop.className="INACTIVE"
        };
    };
}
```

10.1.2 flask_app.py

Hlavní částí mého programu je soubor *flask_app.py*. Tento soubor propojuje knihovnu Raspberry Pi s webovým GUI a využívá knihovnu *flask*. Obsahuje dvě funkce – `get(self)` a `post(self)`, které jsou definované v třídě *GetPost* odkazující na webové GUI.

Pokud volám funkci submit v GUI, tak tato funkce volá funkci `post(self)`, která je definována v následující části kódu. Tato funkce nejdříve načte data z GUI (`request.json`) a potom zavolá funkci `write(write_data)` z knihovny *PortTester_Interface.py*, tím se hodnoty zapíšou do Raspberry Pi.

Druhou funkcí je funkce `get(self)`, tato funkce je volána při stisknutí tlačítka *GET_ACTUAL_STATUS*. Na začátku GUI odešle příkaz GET na adresu definovanou příkazem `api.add_resource(GetPost,'/api')`. Odpovědí na GET jsou *read_data*, které jsou následně zpracovány a zobrazeny.

Poslední částí programu je soubor část main. Tato část obsahuje pouze jeden příkaz – `api.run(debug=True)`. Tento příkaz spouští flask development server a umožňuje komunikaci.

```
class GetPost(Resource):
    def get(self):
        read_data = PTI.read(PTI.actual_data)
        print(read_data)
        return jsonify(read_data)

    def post(self):
        write_data = request.json
        print(write_data)
        PTI.write(write_data)
        return jsonify(write_data)

api.add_resource(GetPost, '/api')

if __name__ == '__main__':
    app.run(debug=True)
```

10.1.3 Softwarová knihovna pro ovládání Raspberry Pi

Knihovna pro ovládání Raspberry Pi je nazvána *PortTester_Interface.py*, je psána v jazyce Python a obsahuje 4 funkce. První dvě funkce jsou použity pouze pro převod mezi datovými typy z dictionary na bajt a naopak. Další dvě funkce jsou již použity k ovládání Raspberry Pi.

Při psaní programu jsem nejprve vůbec nevyužíval Raspberry Pi a pouze jsem se soustředil na nastavení komunikace, proto jsem v této části pouze využíval tuto knihovnu jako testovací. Potom, co jsem měl funkční API, jsem začal používat Raspberry Pi a proto jsem mohl začít používat 2 knihovny pro ovládání GPIO a SPI signálů na Raspberry Pi – *RPi.GPIO* a *spidev*. Tyto knihovny umožňují jednoduše definovat a ovládat signály na Raspberry Pi.

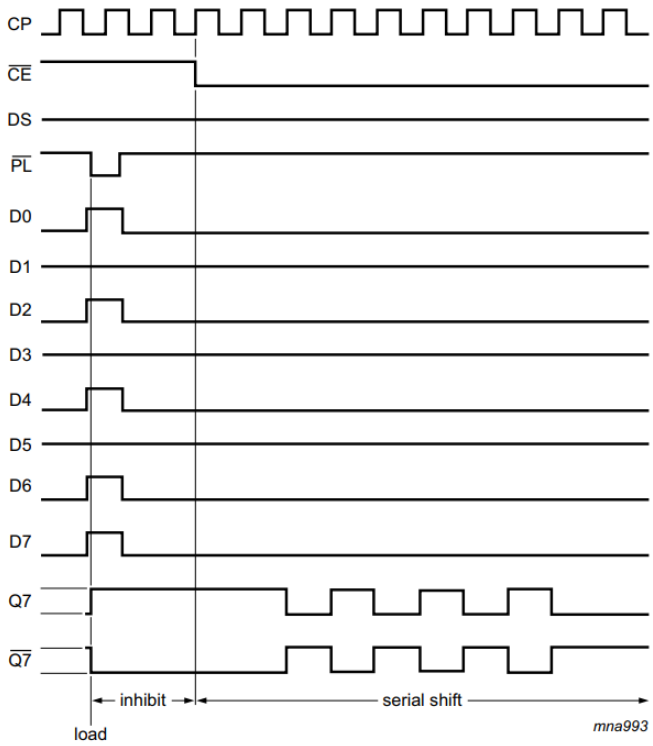
10.1.3.1 read

Tato funkce vyčítá hodnoty ze vstupních shift registrů a aktualizuje na webovém GUI aktuálně aktivované signály. Funkce se volá při stisknutí tlačítka *GET ACTUAL STATUS*. Data se nejsou rozparsovaná a jsou ve struktuře 3 bajtů.

V následující ukázce kódu ukázáno nastavení GPIO a SPI rozhraní společně se samotným čtením z shift registrů. Na začátku jsem nastavil, na jakých pinech budou GPIO signály a nastavil jsem je jako výstupy. V další části už probíhá konfigurace samotné SPI komunikace, je nastavená tak, že se povoluje GPIO signálem *CE*.

Celý proces probíhá následujícím způsobem. Na začátku čtení vytvořím signálem *PL* krátký pulz do hodnoty logické 0, kterým se načtou aktuální hodnoty signálů do shift registrů. Dále přepnutím signálu *CE* do 0 povolím čtení, které načte aktuální hodnoty do proměnné *MOSI*. Na konci čtení nastavím opět signál *CE* do 1, čímž zakážu čtení dat. Tato sekvence se opakuje při každém volání této funkce. Data mohou být dále použita, protože jsou uložena v proměnné *MOSI*. Celý proces zpracování signálů je zobrazen na Obr. 10-3.

Aktualizování hodnot na web serveru jsem udělal jen jednoduchým zkopírováním výstupních dat do nové proměnné, podle které se různě zbarvují obdélníky pod tlačítky a tím signalizují aktuální stav daných signálů.



Obr. 10-3 Časový diagram zpracování signálů vstupním shift registrem [26]

```
# Setting up GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

# Defining GPIO signals
PL = 16
CE = 7

# Setting GPIO outputs
GPIO.setup(PL, GPIO.OUT)
GPIO.setup(CE, GPIO.OUT)
GPIO.output(PL, GPIO.LOW)
GPIO.output(PL, GPIO.HIGH)

# Setting up the SPI communication
SPI_BUS = 0
INPUTS = 1
SPI_IN = spidev.SpiDev()
SPI_IN.open(SPI_BUS, INPUTS)
SPI_IN.max_speed_hz = 500000
SPI_IN.mode = 0b00
SPI_IN.no_cs = True

# Enable reading data from SR
GPIO.output(CE, GPIO.LOW)

# Reading from SR
MISO = SPI_IN.readbytes(3)
print(MISO)

# Disable reading from SR
GPIO.output(CE, GPIO.HIGH)
```

10.1.3.2 write

Funkce *write* je volána při stisknutí tlačítka *SUBMIT* na web serveru. Tato funkce zapisuje potřebné hodnoty do výstupních shift registrů, odkud se dále posílají do příslušných budících obvodů tranzistorů. Začátek funkce je stejný jako pro funkci `read(data_read)` a spočívá v nastavení GPIO módu a konkrétních signálů. V této funkci používám 3 GPIO signály – *MASTER_RESET_N, TRIGGER_OUT* a *ENABLE_SPI.* První signál funguje, jak již jeho název napovídá, pro resetování shift registrů, přesněji řečeno, vymaže z nich všechna zapsaná data. Náběžná hrana signálu *TRIGGER_OUT* zapisuje hodnoty z shift registru na jeho fyzické paralelní výstupy. A poslední signál povoluje samotnou SPI komunikaci.

Na začátku zápisu vymažu z shift registru veškerá data a povolím komunikaci přepnutím signálu *ENABLE_SPI* do 0. V dalším kroku navážu SPI komunikaci a převedu zapisovaná data do správné podoby – funkce `dct2byte(data_dct2byte)`. Následuje samotný zápis dat a jejich překlopení na fyzické výstupy.

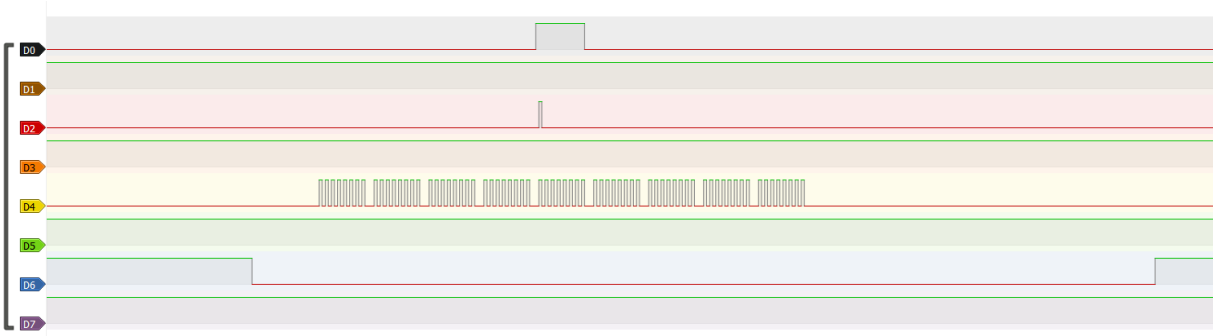
```
# Setting up GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

# Defining GPIO signals
MASTER_RESET_N = 24
TRIGGER_OUT = 25
ENABLE_SPI= 8

# Setting GPIO outputs
GPIO.setup(ENABLE_SPI, GPIO.OUT)
GPIO.setup(MASTER_RESET_N, GPIO.OUT)
GPIO.setup(TRIGGER_OUT, GPIO.OUT)
GPIO.output(ENABLE_SPI, GPIO.LOW)
GPIO.output(MASTER_RESET_N, GPIO.LOW)
GPIO.output(TRIGGER_OUT, GPIO.LOW)
GPIO.output(MASTER_RESET_N, GPIO.HIGH)

# Setting up the SPI communication
SPI_BUS = 0
OUTPUTS = 0
SPI_OUT = spidev.SpiDev()
SPI_OUT.open(SPI_BUS, OUTPUTS)
SPI_OUT.max_speed_hz = 500000
SPI_OUT.mode = 0b00
SPI_OUT.no_cs = True

byte_data = dct2byte(data_write)
global actual_data

# Writing to SR
SPI_OUT.writebytes2(byte_data)

# Write data to the physical outputs
GPIO.output(TRIGGER_OUT, GPIO.HIGH)
```

10.1.4 Výstup na logickém analyzátoru

V této kapitole je popsán test na logickém analyzátoru. Data jsem zadal přes webové GUI. Jedná se o 9 bajtů, v tomto případě jsem nastavil všechny signály do 0, pouze prostřední – 5. bajt obsahuje 1. Zakliknul jsem příslušná tlačítka a odeslal jsem je pomocí tlačítka *SUBMIT* do Raspberry PI, ke kterému byl připojen logický analyzátor.

Na Obr. 10-4 jsou data, která zachytil analyzátor. Signál D0 odpovídá signálu *MOSI*, D4 signálu *CLOCK* a D6 zase signálu *CS* (chip select), který jsem nakonec pro ovládání port testeru nepoužil. Jak je vidět z obrázku, tak na začátku se otevře spojení, to odpovídá místu, kde se signál *CS* změní na logickou 0. V programu tato akce odpovídá příkazu `SPI_OUT.open(SPI_BUS, OUTPUTS)`. Po nějakém zpoždění začne fungovat hodinový signál *CLOCK*, který svou strukturou odpovídá devíti po sobě jdoucím bajtům. A nakonec zkontroluji data *MOSI*, která odpovídají odeslaným datům – prostřední bajt jsou jedničky, zbytek 0. V místě, kde CS nabude hodnoty 1 je uzavřena SPI komunikace a v programu to odpovídá příkazu `SPI_OUT.close()`.

Tímto postupem jsem si ověřil, že správná data dostávám až na výstup Raspberry Pi. Proto se při oživování port testeru budu moci soustředit pouze na správnou funkci shift registrů.



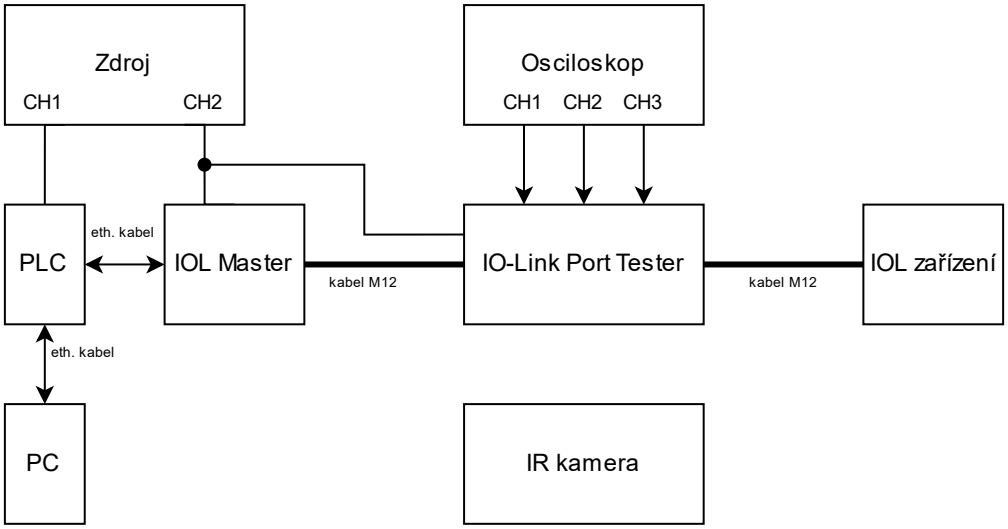
Obr. 10-4 Příklad SPI komunikace zachycené na logickém analyzátoru

KAPITOLA 11: HW TESTY

V této části budou popsány jednotlivé hardwarové testy, kterými budu testovat funkčnost navrženého prototypu, zejména z hlediska napájení, kontroly jednotlivých typů přerušení a emise rušení.

11.1 Testovací sestava

Základní testovací sestava, na které jsem prováděl testy, je zobrazena na Obr. 11-1. Sestava se skládá z několika částí. Ze zdroje DC napětí napájím IO-Link master a řídící PLC, tyto dvě zařízení spolu komunikují po ethernetovém kabelu, který z PLC vede také do mého PC, ze kterého probíhá celá konfigurace a ovládání systému. Pro testovací účely budu využívat pouze jedno IO-Link zařízení, které je připojeno přes port tester do jednoho portu IO-Link masteru. Seznam použitých zařízení je v potom v Tab. 11-1



Obr. 11-1 Blokové schéma testovacího pracoviště

Tab. 11-1 Seznam použitých zařízení

Zařízení
PLC S7-1511PN
SIMATIC ET 200eco PN CM 8xIO-Link + DI 4x24V DC M12-L
IO-Link Port Tester
SIMATIC ET 200AL DIQ16
Osciloskop
Zdroj
IR kamera

11.2 Kontrola externího napájení

Tab. 11-2 Tabulka výsledků testu externího napájení

Označení prototypu	IOL_PT_#1
Použité zařízení	Sestava z Obr. 11-1
Tester	Ondřej Bělovský
Datum testu	9.5.2022
Očekávané výsledky	Očekávané výsledky jsou uvedené v kapitole 11.2.1
VÝSLEDEK:	**ODPOVÍDÁ PŘEDPOKLADŮM**

11.2.1 Přepokládané výsledky

Zde otestuji správnost návrhu zdrojů napájení jednotlivých hladin napětí. Buck konvertor by podle mého návrhu (*B.1, strana 2*) začít pracovat při 16,5 V. To znamená, že do této hodnoty by mělo být napětí na kondenzátoru C216 (výstupní kondenzátor lineárního regulátoru pro 1P15V) rovné 0, při dalším zvýšení napětí už by se na C212 (výstupní kondenzátor buck konvertoru) měla objevit hodnota 3,29 V (*B.1, strana 2*). U lineárních regulátorů je to trochu jiné, ty mají přesně danou hladinu a od této hladiny výše je napětí rovné 15 V. Při snížení pod kritickou hladinu se napětí regulátoru přímo úměrně snižuje s hodnotou napájecího napětí.

11.2.2 Postup

1. Odpojte IO-Link porty a připojte externí napájení.
2. Nastavte napětí na regulovatelném zdroji na hodnotu 10 V a proudové omezení zdroje na 100 mA.
3. Změřte napětí na kondenzátorech C212, C216 a C218.
4. Zvyšte napětí o 1 V a opakujte kroky 1-2. (Pokračujte až do 17 V.)
5. Nastavte hodnotu napětí na 16,4 V.
6. Změřte napětí na kondenzátorech C212, C216 a C218.
7. Opakujte kroky 3-4. Pokračujte až do 30 V.

11.2.3 Naměřené hodnoty

Tab. 11-3 Tabulka naměřených hodnot pro test externího napájení

Napájecí napětí [V]	**Napětí na C212 [V]**	**Napětí na C216 [V]**	**Napětí na C218 [V]**
10	0	8,65	-
15	0	13,75	-
16,4	3,29	15,00	-
17	3,29	15,00	-
18	3,29	15,00	-
30	3,29	15,00	-

V Tab. 11-3 chybí hodnoty pro napětí na kondenzátoru C218. To je způsobeno odpájením Schottkyho diody D218. Z toho důvodu nebude regulátor N203 napájen. Důvody odstranění této diody budou popsány v kapitole 12.

11.3 Kontrola napájení z jednotlivých IO-Link portů

Tab. 11-4 Tabulka výsledků testu napájení z portů

Označení prototypu	IOL_PT_#1
Použité zařízení	Sestava z Obr. 11-1
Tester	Ondřej Bělovský
Datum testu	9.5.2022
Očekávané výsledky	Očekávané výsledky jsou uvedené v kapitole 11.3.1
VÝSLEDEK:	**ODPOVÍDÁ PŘEDPOKLADŮM**

11.3.1 Předpokládané výsledky

Hlavní myšlenka tohoto testu je, ověřit napájení port testeru z jednotlivých IO-Link portů. Po připojení portů a zapnutí napájení by se na všech kondenzátorech mělo objevit příslušné napájecí napětí.

11.3.2 Postup

1. Odpojte všechny IO-Link kanály a externí napájení.
2. Zapojte port 1 (1L+ a 2L+ budou napájeny ze stejného kanálu zdroje).
3. Nastavte napětí na zdroji na 24 V.
4. Zkontrolujte napětí na kondenzátorech C212, C216 a C218.
5. Kroky 1-2 proveďte pro porty 2-8.

11.3.3 Naměřené hodnoty

Tab. 11-5 Tabulka naměřených hodnot pro test napájení z portů

Port	Napětí na C216 [V]	Napětí na C212 [V]	Napětí na C218 [V]
Port 1	15,00	3,29	15,00
Port 2	15,01	3,29	15,00
Port 3	15,00	3,28	15,00
Port 4	15,00	3,28	15,01
Port 5	15,01	3,29	15,01
Port 6	15,01	3,29	15,00
Port 7	15,00	3,29	15,01
Port 8	15,00	3,29	15,00

11.4 Kontrola prototypu IR kamerou

Tab. 11-6 Tabulka výsledků testu externího napájení

Označení prototypu	IOL_PT_#1
Použité zařízení	Sestava z Obr. 11-1
Tester	Ondřej Bělovský
Datum testu	9.5.2022
Očekávané výsledky	Očekávané výsledky jsou uvedené v kapitole 11.4.1
VÝSLEDEK:	**NEODPOVÍDÁ PŘEDPOKLADŮM**

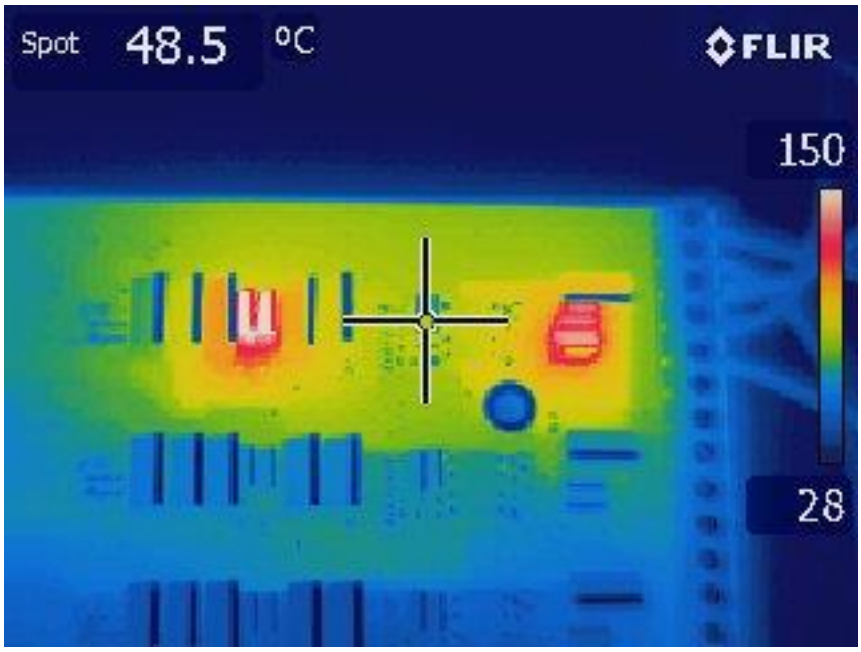
11.4.1 Přepokládané výsledky

Při maximálním zatížení jednoho portu port testeru by nemělo docházet k nadměrnému přehřívání součástek ani celé DPS.

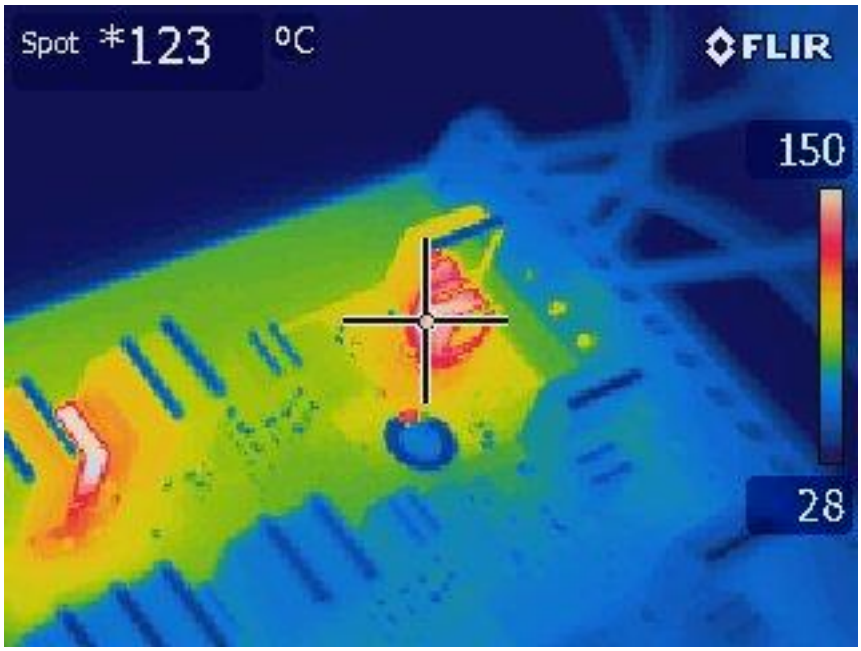
11.4.2 Postup

1. Na svorky 1L+, C/Q a 2L+ připojte kladnou svorku zdroje, na svorky 1M a 2M zase zápornou.
2. Na výstupní stranu zapojte rezistory mezi signály 1L+ - 1M, C/Q – 1M a 2L+ - 2M.
3. Nastavte hodnotu odporů tak, aby obvody 1L+ a 2L+ tekl proud 2 A a obvodem C/Q proud 0,5 A.
4. Nechte běžet v tomto stavu 5 minut a proveďte kontrolní měření IR kamerou.
5. Nechte běžet dalších 20 minut a proveďte výstupní kontrolu IR kamerou.

11.4.3 Naměřené hodnoty



Obr. 11-2 Detail zapojení portu pod IR kamerou v čase 5 min od začátku testu



Obr. 11-3 Detail tranzistorů pod IR kamerou v čase 5 minut od začátku testu

V tomto testu docházelo k nadměrnému zahřívání desky v okolí zapojeného portu, především potom tranzistorů Q308, Q309 a Q3151, Q316. Podrobněji budou výsledky testů rozebrány v kapitole 12.

11.5 Wirebreak

Tab. 11-7 Tabulka výsledků pro testy wirebreaků

Označení prototypu	IOL_PT_#1
Použité zařízení	Sestava z Obr. 11-1
Tester	Ondřej Bělovský
Datum testu	9.5.2022
Očekávané výsledky	Očekávané výsledky jsou uvedené v kapitole 11.5.1
VÝSLEDEK:	**ODPOVÍDÁ PŘEDPOKLADŮM**

11.5.1 Předpokládané výsledky

V tomto testu již testuji samotnou funkci port testeru – wirebreak signálů 1L+, C/Q a 2L+. Jeden kanál osciloskopu bude umístěn na daném signálu na vstupní straně a druhý na výstupní straně port testeru. Na vstupní straně bude také proudová sonda osciloskopu. Očekávám, že při neaktivním wirebreaku bude napětí jak na vstupu, tak i na výstupu rovné 24 V. Při aktivovaném wirebreaku by mělo napětí na výstupu poklesnou k nule. Z časových důvodů bude tento test proveden pouze na prvním a posledním portu, stejně tomu bude i u zkratů. V tabulce naměřených hodnot nebudu uvádět hodnoty napětí, ale pouze časové údaje, které odpovídají času sestupné hrany. Tyto hodnoty by se měly pohybovat maximálně v desítkách µs.

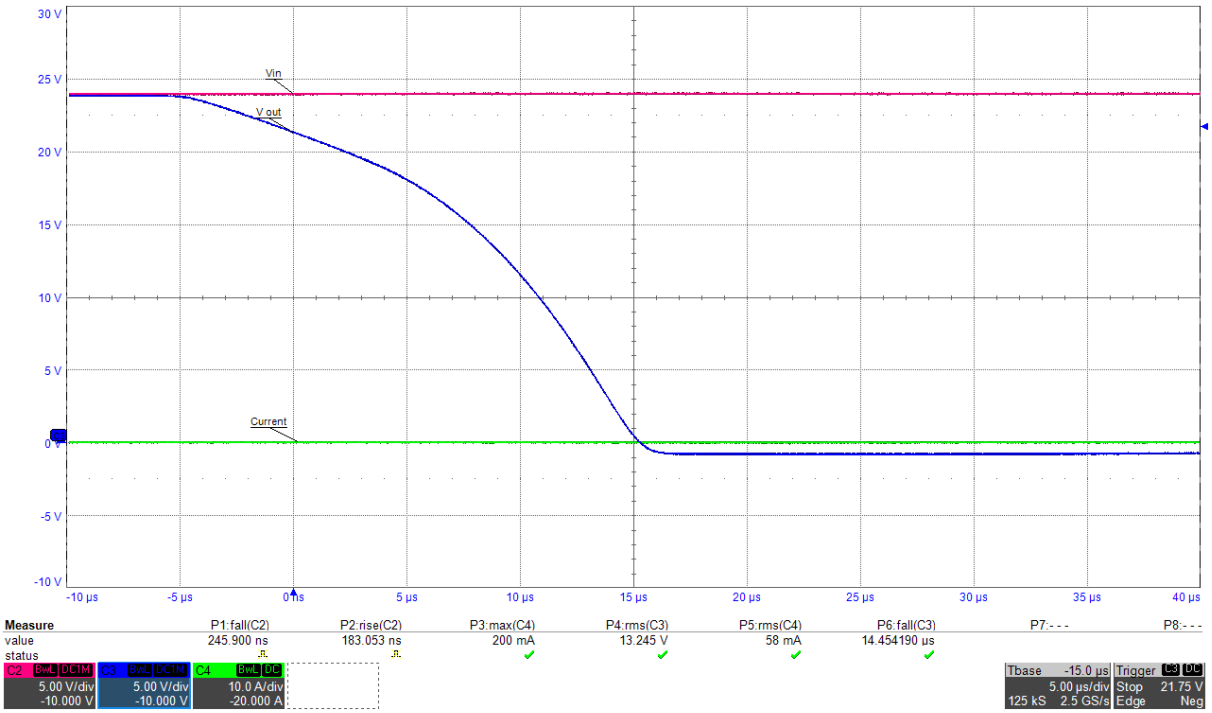
11.5.2 Postup

1. Zapojte port 1, a to jak na straně vstupní, tak i výstupní.
2. Aktivujte wirebreak na 1L+.
3. Zachyťte průběh wirebreaku na osciloskopu.
4. Zrušte wirebreak na 1L+.
5. Aktivujte wirebreak na CQ.
6. Zachyťte průběh wirebreaku na osciloskopu.
7. Zrušte wirebreak na CQ.
8. Aktivujte wirebreak na 2L+.
9. Zachyťte průběh wirebreaku na osciloskopu.
10. Zrušte wirebreak na 2L+.
11. Opakujte kroky 1-10 pro porty 2-8.

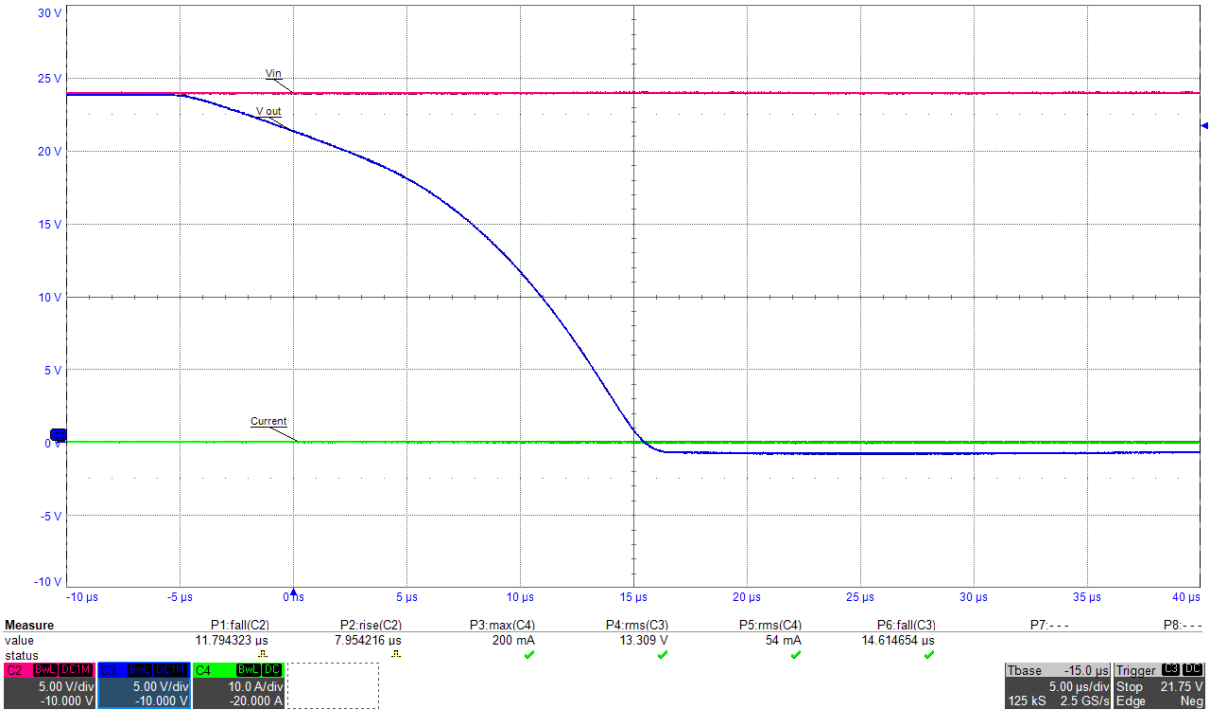
11.5.3 Naměřené hodnoty

Tab. 11-8 Tabulka naměřených hodnot pro wirebreaky

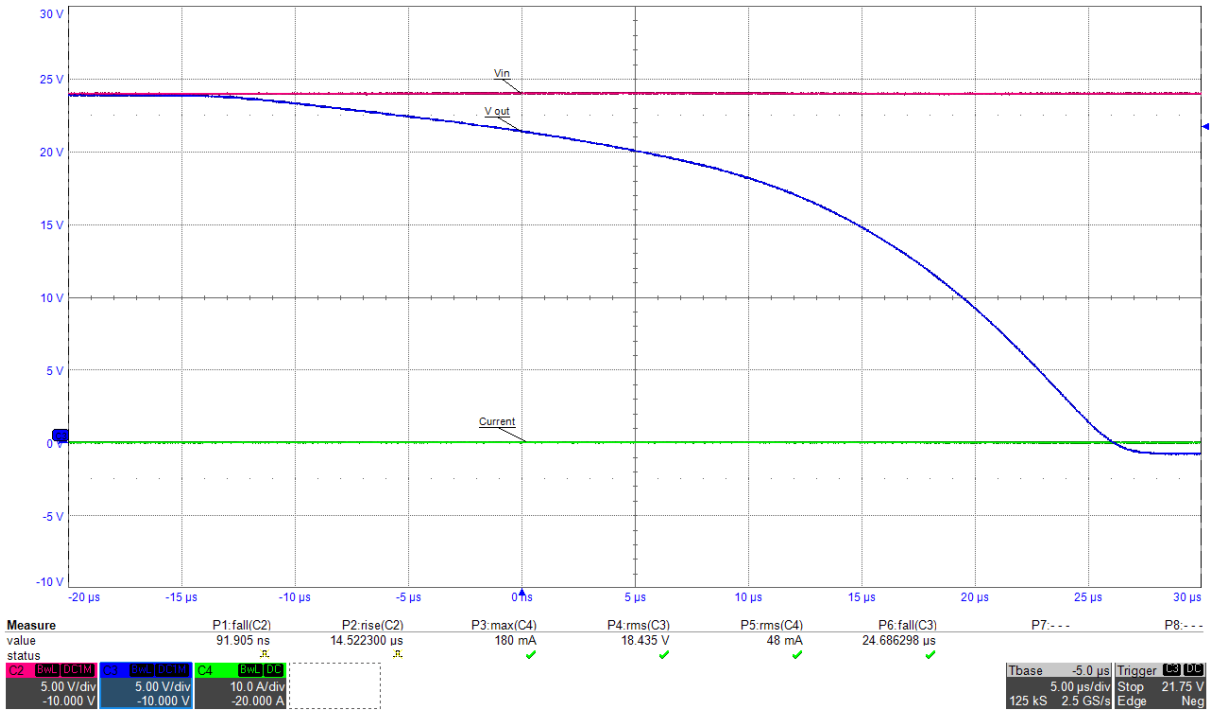
		Čas sestupné hrany* [µs]
Port 1	**WB 1L+**	**14,45**
	WB C/Q	**14,61**
	WB 2L+	**24,69**
Port 8	**WB 1L+**	**14,64**
	WB C/Q	**14,42**
	WB 2L+	**20,66**
* čas poklesu napětí z 90 % na 10 % jmenovité hodnoty		



Obr. 11-4 Detail wirebreaku na 1L+ na portu 1 s rezistorem (1 kΩ) zapojeným na výstupní straně



Obr. 11-5 Detail wirebreaku na C/Q na portu 1 s rezistorem (1 kΩ) zapojeným na výstupní straně



Obr. 11-6 Detail wirebreaku na 2L+ na portu 1 s rezistorem (1 kΩ) zapojeným na výstupní straně

11.6 Zkrat

Tab. 11-9 Tabulka výsledků pro testy zkratů

Označení prototypu	IOL_PT_#1
Použité zařízení	Sestava z Obr. 11-1, IO-Link zařízení nepřipojeno
Tester	Ondřej Bělovský
Datum testu	9.5.2022
Očekávané výsledky	Očekávané výsledky jsou uvedené v kapitole 11.6.1
VÝSLEDEK:	**NEODPOVÍDÁ PŘEDPOKLADŮM**

11.6.1 Předpokládané výsledky

Tento test by měl otestovat funkčnost zkratů. V tomto testu nebude na výstupní straně nic připojeno. Jeden kanál osciloskopu bude umístěn na příslušných vstupních signálech. Při aktivaci zkratu očekávám pokles napětí mezi měřenými signály k nule. Tabulka naměřených hodnot bude obsahovat mimo času sestupné hrany také špičkový proud I_{max}, napětí v okamžiku špičkového proudu U_{Imax} a sériový odpor zkratu R_{ser}. Tento sériový odpor se vypočítá z rovnice (11-1). Sériový odpor je velmi důležitý parametr zkratů a tento odpor by měl být co možná nejmenší. Obecně platí, že čím menší sériový odpor, tím kvalitnější zkrat.

$$R_{ser} = \frac{U_{Imax}}{I_{max}} \tag{11-1}$$

11.6.2 Postup

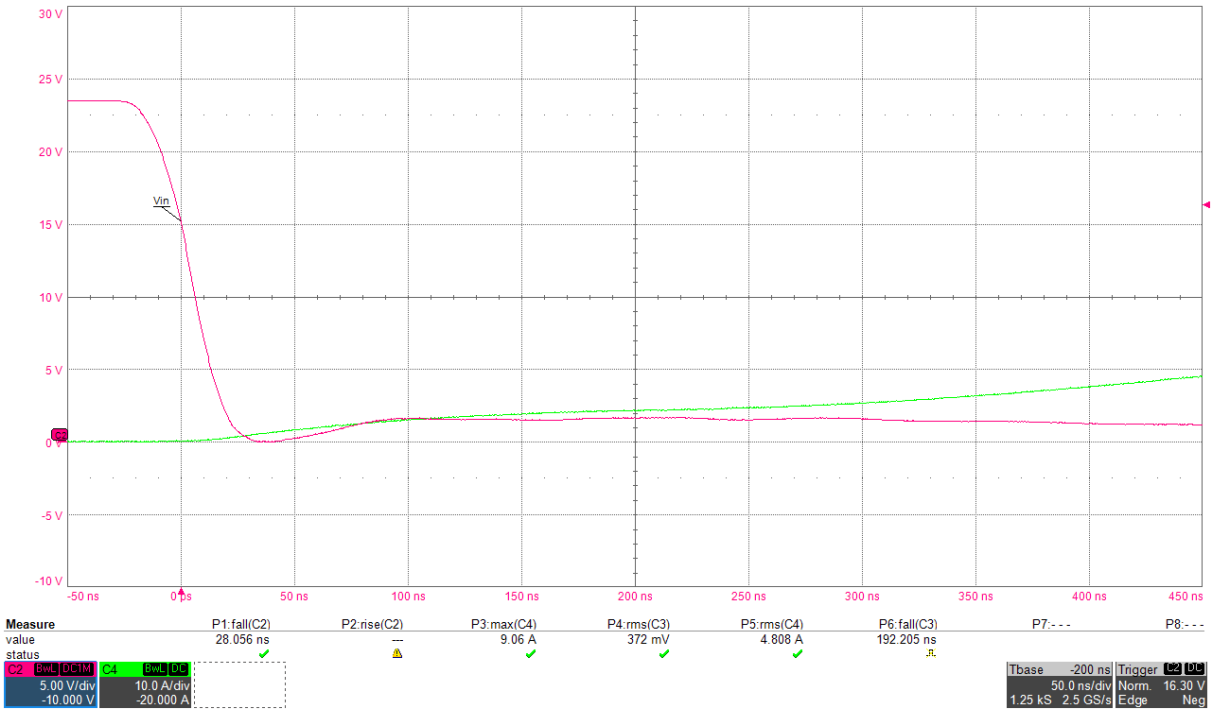
1. Zapojte port 1.
2. Aktivujte zkrat mezi 1L+ a 1M.
3. Zachyťte průběh zkratu na osciloskopu.
4. Zrušte aktivní zkrat.
5. Změňte konfiguraci IO-Link masteru na port class B
6. Aktivujte zkrat mezi 2L+ a 2M.

7. Zachyťte průběh zkratu na osciloskopu.
8. Zrušte aktivní zkrat.
9. Aktivujte zkrat mezi CQ a 1L+.
10. Zachyťte průběh zkratu na osciloskopu.
11. Zrušte aktivní zkrat.
12. Aktivujte zkrat mezi CQ a 1M.
13. Zachyťte průběh zkratu na osciloskopu.
14. Zrušte aktivní zkrat.
15. Změňte konfiguraci IO-Link masteru na port class A
16. Aktivujte zkrat mezi 2L+ a 1L+.
17. Zachyťte průběh zkratu na osciloskopu.
18. Zrušte aktivní zkrat.
19. Aktivujte zkrat mezi 2L+ a 1M.
20. Zachyťte průběh zkratu na osciloskopu.
21. Zrušte aktivní zkrat.
22. Kroky 1-21 opakujte pro porty 2-8.

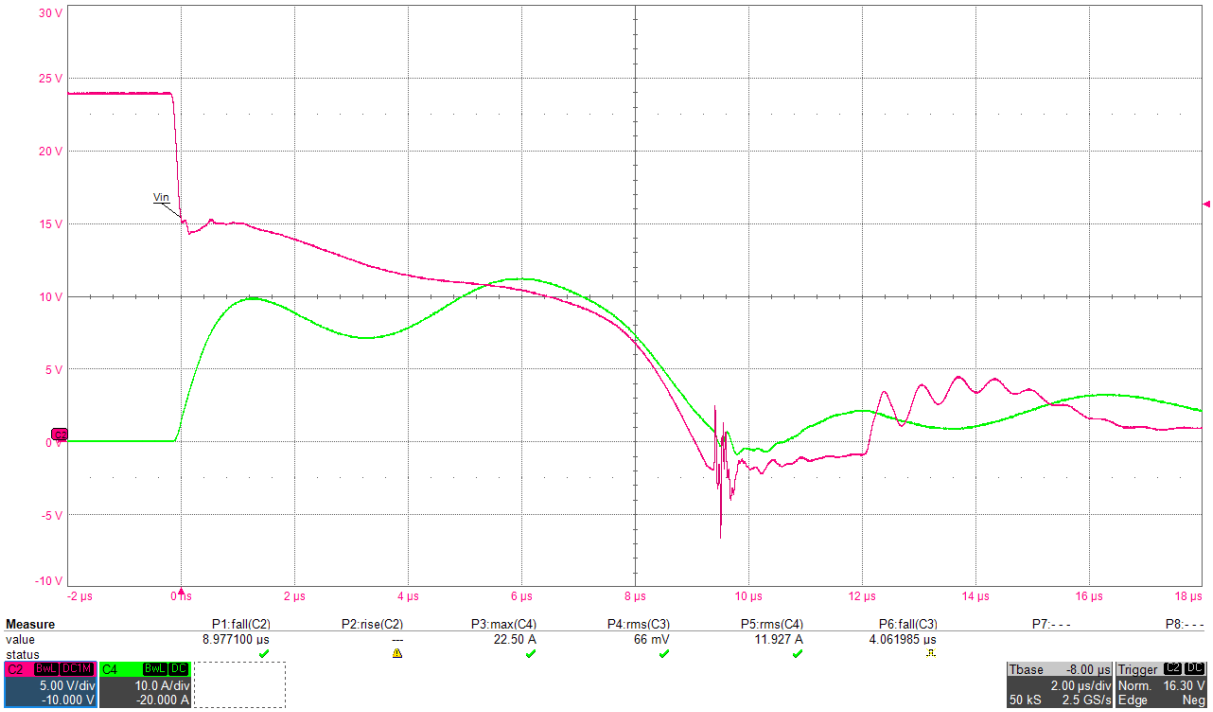
11.6.3 Naměřené hodnoty

Tab. 11-10 Tabulka naměřených hodnot pro test zkratů

		Čas sestupné hrany [ns]	I_{max} [A]	R_{ser} [Ω]
Port 1	**1L+ - 1M**	**28,06**	**34,82**	**0,01**
	2L+ - 2M	**8977,00**	**22,50**	**0,44**
	C/Q - 1L+	**24,12**	**4,48**	**0,06**
	C/1 - 1M	**282,64**	**7,96**	**0,06**
	Pin 2 - 1L+	**13,72**	**21,94**	**0,11**
	Pin 2 - 1M	**15,09**	**34,44**	**0,07**
Port 8	**1L+ - 1M**	**268,19**	**26,48**	**0,07**
	2L+ - 2M	**8250,00**	**19,86**	**0,55**
	C/Q - 1L+	**18,97**	**4,54**	**0,09**
	C/1 - 1M	**13,98**	**8,00**	**0,07**
	Pin 2 - 1L+	**13,68**	**21,96**	**0,18**
	Pin 2 - 1M	**15,05**	**34,60**	**0,09**

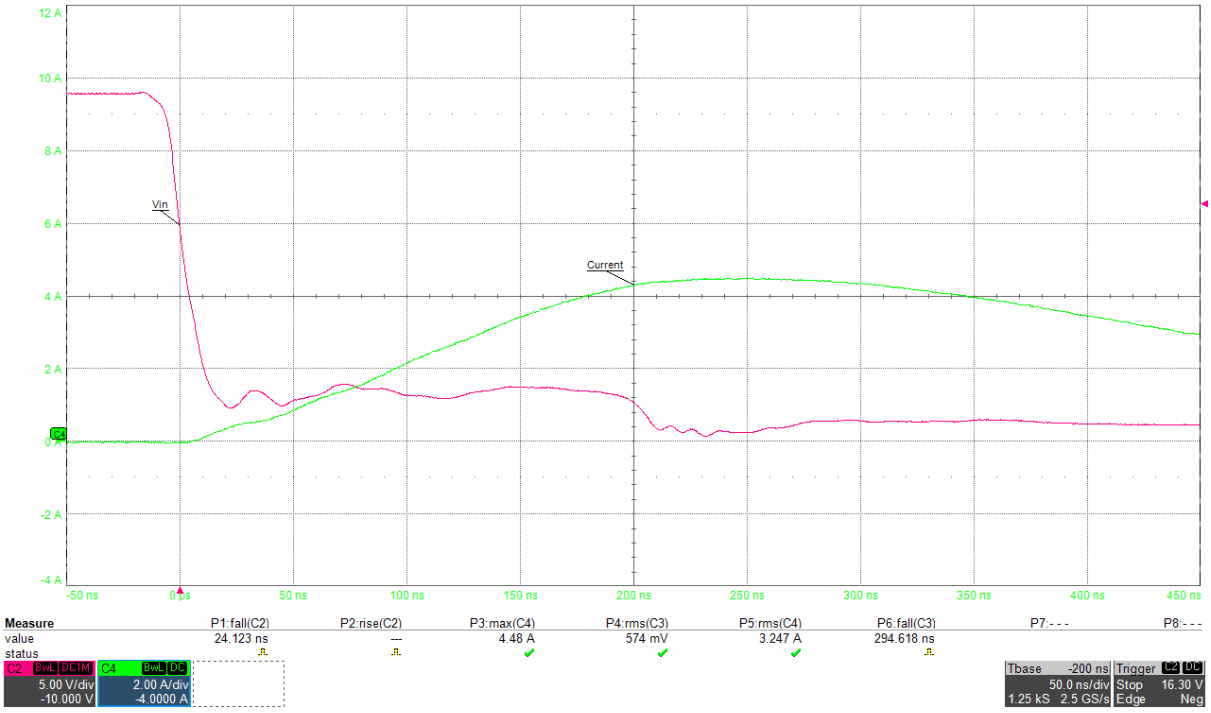


Obr. 11-7 Detail zkratu na hlavním napájení (1L+ - 1M) na portu 1

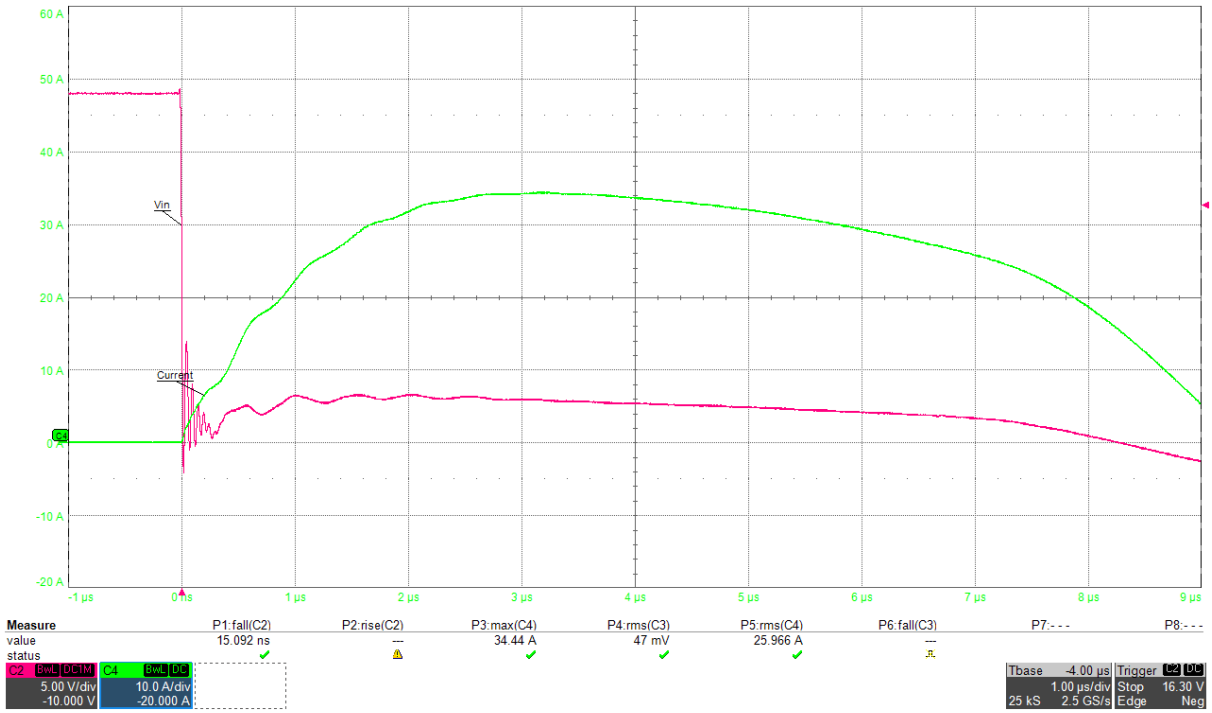


Obr. 11-8 Detail zkratu na přídavném napájení (2L+ - 2M) na portu 1

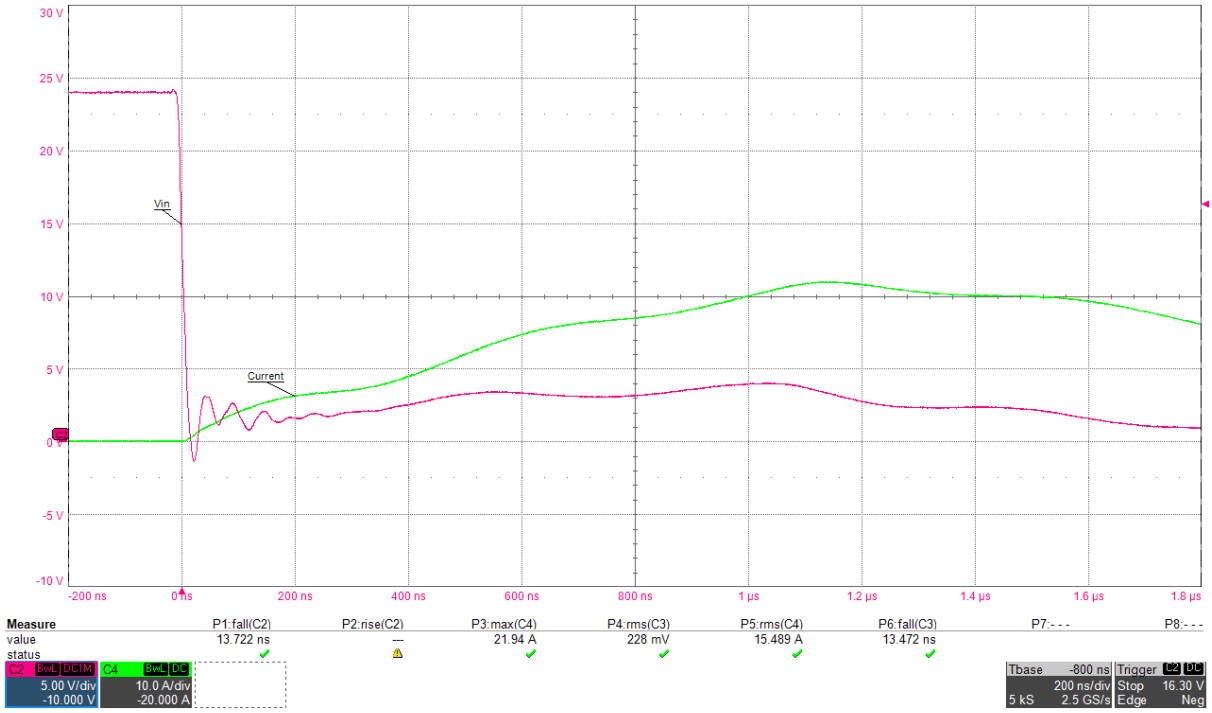
Obr. 11-9 Detail zkratu mezi C/Q – 1M na portu 1



Obr. 11-10 Detail zkratu mezi C/Q – 1L+ na portu 1



Obr. 11-11 Detail zkratu mezi Pinem 2 – 1M na portu 1



Obr. 11-12 Detail zkratu mezi Pinem 2 – 1L+ na portu 1

Test jsem vyhodnotil jako neúspěšný zejména, protože hodnoty odporu zkratů 2L+ - 2M a pin 2 – 1L+jsou příliš vysoké a neodpovídají předpokladům uvedeným v 11.6.1. Podrobnější rozbor včetně možných důvodů bude uveden v kapitole 12.

KAPITOLA 12: VYHODNOCENÍ VÝSLEDKŮ

Toto vyhodnocení začnu kontrolou požadavků na port tester, které jsou shrnuty v kapitole 2. Nejdůležitější byl požadavek na všechny typy diagnostik – zkraty a wirebreaky. Z tohoto pohledu port tester splňuje všechny požadavky, které jsou uvedeny na Obr. 2-1. Zkraty splňují předpoklady pouze z pohledu integračního testu vyvolávání diagnostik, problém se zkraty bude popsán později v této kapitole. Spínače jsem vyřešil pomocí signálově řízených tranzistorů, přičemž některé spínače z Obr. 2-1 jsem vynechal, protože tam byly zbytečně. Vynechal jsem spínače označené jako *PM Select* a *Pin 2 Select*, které byly v blokovém schématu pouze kvůli přepínání signálu do dalšího spínače. Spínače *PM Select* a *C/Q High Low* jsem vyřešil pomocí dvou navzájem propojených tranzistorů, které jsou řízené signály a spínané pomocí half-bridge driveru. To znamená, že jsem schopen získat všechny 3 stavy, kterých bych dosáhl při použití spínačů v blokovém schématu. Toto zapojení je popsáno v kapitole 8.4.2. Stejným způsobem jsem to vyřešil i pro pin 2.

Dalším z požadavků na port tester bylo vyčítání hodnot signálů 1L+, C/Q a 2L+. Tato funkčnost je zajištěna z HW hlediska vstupními shift registry, před kterými ještě musím snížit napětí z 24 V na maximálně 3,3 V. Protože shift registry jsou napájené 3,3 V a napětí vstupů nemůže být vyšší než napájecí napětí. Napěťový dělič, který jsem zvolil převádí napětí podle rovnice (8-1) z 24 V na 2,18 V. Ze softwarového hlediska tyto hodnoty vyčítám pomocí SPI rozhraní Raspberry Pi přímo z shift registrů do Raspberry Pi, a to konkrétně funkcí *read.* Signály C/Q jsou dále znegovány a vyvedeny na pin header, kde je možnost připojení logického analyzátoru nebo USB/UART konvertoru.

Patrně posledním, ale neméně důležitým požadavkem byla také možnost snadné implementace ovládání port testeru přímo v programech pro jednotlivé testy. Použil jsem řešení REST API, které je velice snadno implementovatelné v dalších aplikacích. Jako programovací jazyk jsem zvolil Python, protože je to hlavní jazyk používaný pro integrační test a díky tomu půjde ovládání port testeru jednoduše automatizovat. GUI bylo také jedním z požadavků, protože se jedná o velice jednoduché řešení, jak ovládat port tester manuálně. Zvolil jsem řešení pomocí webového GUI, které se opět jednoduše implementovalo v jazyce HTML a JavaScript.

Celý návrh není úplně bezchybný. Po odeslání port testeru do výroby jsem si všiml chyby ve schématu, konkrétně se jedná o Schottkyho diodu D218 (*příloha B.1, strana 2*), která v tomto případě spojuje 2 různé referenční potenciály. Připojuje signál +24 (referovaný k 1M) na lineární regulátor, který vytváří napětí referované ke 2M. Toto by se rozhodně nemělo stávat, takže jsem byl nucen po obdržení vyrobeného port testeru odpájet tuto součástku. Není to nijak zásadní problém, protože z hladiny 2P15V napájím pouze budící obvody tranzistorů, přičemž tyto obvody vyvolávají diagnostiky na pinu 2, takže je logické, že pokud chci dělat tato přerušení, tak budu mít zapojen IO-Link master. Větší problém by to byl pro potenciál 1M, ke kterému je vztažena převážná část DPS.

Pokud bych měl nějak podrobněji vyhodnotit HW testy, tak základní testy napájení proběhly v pořádku. Pouze jsem nebyl schopný měřit napětí lineárního regulátoru v testu *Kontrola externího napájení*, protože Schottkyho diody D218 jsem odpájel z důvodu, které jsou popsány výše.

Pokud se podívám na rozdíl mezi wirebreaky a zkraty, tak je tam na první pohled viditelný rozdíl v délce sestupné hrany, která je u wirebreaku několikrát delší než u zkratů. Tato skutečnost je způsobena tím, že u zkratu můžu nabíjet kapacitu gate pro sepnutí tranzistoru výrazně větším proudem (až okolo 1 A). U wirebreaků se gate vybíjí přes rezistor 15 kΩ, proto je průběh výrazně pomalejší. Průběh zkratu na hlavním napájení je velmi hladký a špičkový proud dosahuje zhruba průměrně hodnoty 22 A. Na Obr. 11-4 je průběh napětí a proudu při wirebreaku na 1L+. Je vidět, že wirebreak funguje správně a průběh sestupné hrany je plynulý, i když výrazně pomalejší než u zkratů. Průběhy všech wirebreaků vypadají velice podobně.

U zkratů jsem také odečítal špičkový proud a napětí při špičkovém proudu. Z těchto hodnot jsem byl podle Ohmova zákona schopen dopočítat odpor zkratové smyčky, který by se měl blížit nule. Problém nastal při zkratu na přídavném napájení 2L+ - 2M, pro který byl odpor 0,44 Ω. U zkratu mezi pinem 2 a 1L+ byl odpor 0,11 Ω. To už jsou na zkrat velmi vysoké hodnoty. Z tohoto důvodu byl test na zkraty neúspěšný. Vysoká hodnota odporu zkratové smyčky by mohla být způsobena špatným spínáním tranzistorů, příliš tenkými signály na DPS, případně také chybou měření. Bude zapotřebí další investigace a objasnění tohoto problém. V dalším prototypu tato chyba musí být opravena.

U hardwarových testů jsem narazil také na problém u testu s IR kamerou, který je uveden v kapitole 11.4. Během tohoto testu jsem zjistil, že při maximálním možném zatížení se port tester zahřívá více než bych čekal. Jedná se především o anti sériově zapojené tranzistory pro vyvolání wirebreaku na 1L+ a 2L+ (pro port 1 tomu odpovídají součástky Q308, Q309 a Q315, Q316). Během tohoto testu teče těmito tranzistory maximální možný proud – 2 A. Na těchto tranzistorech jsem naměřil teplotu okolo 140 °C po 5 minutách testu. Podle mého názoru je to dané vyšším sériovým odporem, který je pro tyto součástky dost kritický, protože se tyto součástky nacházejí 2 za sebou. Toto přehřívání by mohlo být způsobeno například nedostatkem místa pro chlazení v okolí tranzistorů. Závěr tohoto testu je stejný jako v případě zkratů. Musím zjistit kde je problém a opravit ho v dalším prototypu.

Celkově je ve funkčnosti port testeru pár problémů. Zkraty mají vysoký odpor a port tester se při maximální zatížení přehřívá. Ovšem během testů v kapitole 11 jsem otestoval, že funkčnost port testeru je v pořádku z hlediska integračního testu (diagnostiky se vyvolají). Ovládání přes webové GUI je intuitivní, pouze není otestované, jak se chová port tester při automatizaci. Věřím ale, že pokud funguje ovládání přes web, tak programové ovládání je už jen otázka implementace knihovny.

ZÁVĚR

Cílem této diplomové práce bylo navrhnout a otestovat hardwarový prototyp zařízení, které jsme nazvali *IO-Link Port Tester*, včetně implementace softwarové knihovny pro ovládání port testeru pomocí připojeného Raspberry Pi.

V teoretické části jsem se zabýval dnes čím dál více využívaným komunikačním standardem IO-Link. V této části, která je popsána na straně 2 jsem popsal základy komunikace přes IO-Link včetně zařízení, které takto mohou komunikovat. Jedná se především o senzory a akční členy. Jsou zde popsány i výhody IO-Linku.

Další teoretická část, konkrétně pak kapitoly 3 a 4 se zabývají dnes také velmi žádanými technologiemi, distributed IO technologiemi. V těchto kapitolách se zabývám opět nejdříve obecným popisem těchto technologií, poté se věnuji popisu řady distributed IO zařízení *SIMATIC ET 200eco PN* vyvíjené společností Siemens. Tuto řadu jsem zvolil, protože primárně pro tuto řadu je určen port tester, kterým jsem se v této diplomové práci zabýval. V kapitole 4 jsem popisoval problematiku konfigurace, struktury dat a testování IO-Link masteru z řady *SIMATIC ET 200eco PN*. Jsou zde popsány principy testů, se kterými by měl port tester pomáhat. Nejsou zde uvedeny všechny testy, pouze základní diagnostické testy. Součástí této kapitoly je také strukturování dat, jehož znalost je nezbytná pro testování, protože je potřeba vědět jaká diagnostická data je nutné při spuštění konkrétní diagnostiky vyčítat.

Na straně 4 jsou uvedeny požadavky, které by měl port tester splňovat. Tyto požadavky posloužily jako vstupní data pro návrh port testeru. Zhodnocením implementovaných požadavků na reálném zařízení jsem se zabýval v kapitole 12. A jak jsem již zmiňoval port tester splňuje požadavky integračního testu. V kapitole 7.1 jsem popisoval současný stav testovacího racku před využitím port testeru. Port tester výrazně sníží potřebu manuální práce pro zapojení jednotlivých portů. Při použití řešení s port testerem odpadá potřeba použití většiny relé.

V teoretické části jsem nakonec samozřejmě popsal obecně i funkci použitých HW součástek a je zde popsáno také REST API, jež jsem použil pro komunikaci mezi ovládací aplikací (GUI, test skript) a web serverem.

V praktické části se převážně zabývám návrhem elektrického schématu port testeru a jeho layoutu neboli návrhu DPS. Tyto části jsou popsány v kapitolách 8 a 9. Během návrhu těchto částí jsem nenarazil na vážnější problém. Srdcem celého návrhu je návrh zapojení portů, které je tvořené převážně tranzistory spínajícími zkraty, či wirebreaky a jejich budícími obvody. Při návrhu DPS jsem měl za cíl mít co nejmenší DPS, proto bylo zapotřebí zaměřit velké úsilí na vytvoření jednoho portu. Zapojení portů tvoří velmi podstatnou část DPS. Další velkou částí zde byly LED diody s příslušnými odpory, ale také signály, které vedly často téměř přes celou desku. Byl požadavek na umístění všech LED diod v jednom místě, a proto jsem musel vést tyto dlouhé signály. Zbytek místa na desce jsem vyplnil ostatními součástkami. Schéma port testeru je uvedeno v příloze *B.1 Elektrické schéma*.

Dalším důležitým praktickým úkolem bylo vytvoření a implementace softwarové knihovny a webového GUI. Tato softwarová knihovna ovládá vstupy a výstupy Raspberry Pi, které jsou přes pin header připojené na port tester. Přiřazení jednotlivých pinů jsem musel trochu pozměnit, protože jsem musel upravit piny pro odladění na prototypu. Proto nelze využít jeden kabel, kterým by se mohly propojit všechny piny. Vyčtená a zapsaná data se vyměňují přes SPI rozhraní Raspberry Pi a jejich HW zpracování na port testeru zajišťují shift registry. Celý program je součástí přílohy B.2.

Při testování a uvádění do provozu jsem narazil na několik nedostatků, které jsou uvedeny v kapitole 12, kde jsem provedl vyhodnocení funkčnosti port testeru. Jedním z problémů byla Schottkyho dioda D218, která spojuje signály se dvěma různými referencemi. Tento problém bude opraven pro druhý prototyp.

Dalším, větším problémem, bylo přehřívání port testeru při testu na maximální možný výkon. Řešením by mohlo být zlepšení chlazení, např. přidáním chladiče. Dalším řešením by mohlo být i umístění součástek, protože v tuto chvíli jsou všechny tranzistory velmi blízko u sebe, a tudíž se hůře chladí. Také by šly použít kvalitnější tranzistory s menším sériovým odporem. Při ovládání port testeru jsem nevypozoroval žádný problém. Zápis dat a čtení dat z Raspberry Pi funguje správně.

Dalším problémem byly zkraty. Některé ze zkratů měly příliš velký odpor zkratové smyčky. Tato skutečnost je rozepsána v kapitole 12. Řešení bude jasnější až po další investigaci.

V závěru bych chtěl dodat, že se jedná pouze o první prototyp a do budoucna bude opraven o všechny nedostatky a případně i vylepšen o další funkce. V současné době je prototyp použitelný pro potřeby integračního testu.

LITERATURA

[1] VOLF, Ondřej. IO-Link Device for testing of IO-Link Masters. 2017. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology.

[2] ČSN EN 61131-9. Programovatelné řídicí jednotky - Část 9: Drobné digitální komunikační rozhraní pro malé snímače a ovládací členy. 2014.

[3] IO-Link komunikace. In: Automatizace.hw.cz [online]. [cit. 2022-05-08]. Dostupné z: https://automatizace.hw.cz/iolink-popis-digitalni-komunikace-pro-senzory

[4] IO-Link Interface and System Specification. In: . 2019, Version 1.1.3. Dostupné také z: https://io-link.com/share/Downloads/Package-2020/IOL-Interface-Spec_10002_V113_Jun19.pdf

[5] MORTENSON, Ted. Distributed IO systémy. Real Pars [online]. 22.06.2020 [cit. 2022-05-08]. Dostupné z: https://realpars.com/distributed-io/

[6] Siemens: SIMATIC ET 200eco PN. SIEMENS [online]. [cit. 2022-05-08]. Dostupné z: https://new.siemens.com/global/en/products/automation/systems/industrial/io-systems/simatic-et-200eco-pn.html

[7] System Manual: SIMATIC ET 200eco PN M12-L. Support Industry Siemens [online]. 02/2022 [cit. 2022-05-08]. Dostupné z: https://cache.industry.siemens.com/dl/files/292/109778292/att_1027586/v1/et200ecopn_m12l_system_manual_en-US_en-US.pdf

[8] Struktura PQI bajtu. In: Siemens [online]. 11/2019 [cit. 2022-05-08]. Dostupné z: https://cache.industry.siemens.com/dl/files/813/109772813/att_1007238/v1/109772813_IO-Link_Diagnose_en.pdf

[9] Raspberry Pi 4 Model B: Datasheet. Raspberry Pi [online]. 06/2019 [cit. 2022-05-08]. Dostupné z: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf

[10] Raspberry Pi: Pin Header. In: Raspberry Pi: Datasheets [online]. 18/06/2019 [cit. 2022-05-08]. Dostupné z: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-reduced-schematics.pdf

[11] Raspberry Pi 4 Model B: Periferie. Raspberry Pi: Datasheets [online]. 2022-01-18 [cit. 2022-05-08]. Dostupné z: https://datasheets.raspberrypi.com/bcm2711/bcm2711-peripherals.pdf

[12] DHAKER, Piyu. Introduction to SPI interface. Analog Dialogue [online]. 09/2018 [cit. 2022-05-08]. Dostupné z: https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html

[13] Raspberry GPIO. SparkFun [online]. [cit. 2022-05-08]. Dostupné z: https://learn.sparkfun.com/tutorials/raspberry-gpio/all

[14] MM74HC595M: Datasheet. Farnell [online]. 06/2009 [cit. 2022-05-10]. Dostupné z: https://www.farnell.com/datasheets/2285931.pdf

[15] MPQ4420AGJ-P: Datasheet. Farnell [online]. 18/12/2015 [cit. 2022-05-08]. Dostupné z: https://www.farnell.com/datasheets/2914758.pdf

[16] MIC4605-1YM: Datasheet. Farnell [online]. 14/05/2019 [cit. 2022-05-08]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/MIC4605-Data-Sheet-DS20005853E.pdf

[17] MAX14930BASE+T: Datasheet. Farnell [online]. 01/2017 [cit. 2022-05-08]. Dostupné z: https://www.farnell.com/datasheets/2369796.pdf

[18] Wiki: API. Wikipedie [online]. poslední aktualizace 4/10/2021 [cit. 2022-05-15]. Dostupné z: https://cs.wikipedia.org/wiki/API

[19] REST API. IT Network [online]. [cit. 2022-05-08]. Dostupné z: https://www.itnetwork.cz/programovani/nezarazene/stoparuv-pruvodce-rest-api/

[20] ČSN EN 61000-4-4. Elektromagnetická kompatibilita (EMC): Část 4-4: Zkušební a měřicí technika - Rychlé elektrické přechodné jevy/skupiny impulzů - Zkouška odolnosti. Ed.3. 2013.

[21] BUK7E3R5-60E: Datasheet. Farnell [online]. 11/09/2002 [cit. 2022-05-10]. Dostupné z: https://4donline.ihs.com/images/VipMasterIC/IC/NEXP/NEXP-S-A0003060256/NEXP-S-A0003060158-1.pdf?hkey=6D3A4C79FDBF58556ACFDE234799DDF0

[22] FQU17P06TU: Datasheet. Farnell [online]. 04/2014 [cit. 2022-05-10]. Dostupné z: https://www.farnell.com/datasheets/2304440.pdf

[23] MCT62: Datasheet. Farnell [online]. 08/2016 [cit. 2022-05-10]. Dostupné z: https://www.farnell.com/datasheets/2299518.pdf

[24] Pull-up rezistory. Electrnics Tutorials [online]. [cit. 2022-05-10]. Dostupné z: https://www.electronics-tutorials.ws/logic/pull-up-resistor.html

[25] Bypass kondenzátor. CS.jf Parede [online]. [cit. 2022-05-10]. Dostupné z: https://cs.jf-parede.pt/basics-bypass-capacitor

[26] 74HC165D,653: Datasheet. Farnell [online]. 28/12/2015 [cit. 2022-05-10]. Dostupné z: https://4donline.ihs.com/images/VipMasterIC/IC/PHGL/PHGL-S-A0001367765/PHGL-S-A0001367765-1.pdf?hkey=6D3A4C79FDBF58556ACFDE234799DDF0

[27] KPT-2012SRC-PRV: Datasheet. Farnell [online]. 18/12/2011 [cit. 2022-05-14]. Dostupné z: https://www.farnell.com/datasheets/1854096.pdf

PŘÍLOHA A: PŘÍKLAD SEZNAMU SYMBOLŮ A ZKRATEK

A.1 Seznam zkratek

I/O	Inputs/Outputs
UART	Universal asynchronous reciever-transmitter
PLC	Programmable Logic Controller
DI	Digital Input
DQ	Digital Output
PQI	Port Quality Information
GPIO	General Purpose Input Output
SPI	Serial Peripheral Interface
MISO	Master In Slave Out
MOSI	Master Out Slave In
CS	Chip Select
SCK	Serial Clock
SR	Shift registr
REST API	Representational State Transfer Application Programming Interface
JSON	Java Script Object Notation
DPS	Deska plošných spojů
THT	Through-Hole Technology
SMD	Surface-mount Device
GUI	Graphical User Interface
IR	infračervený

A.2 Seznam použitých výrazů

Wirebreak	přerušení vodiče
Pin header	typ elektrického konektoru
buck konvertor	DC/DC snižovací měnič
half-bridge driver	budič tranzistorů v půl můstkovém zapojení

PŘÍLOHA B: SEZNAM PŘILOŽENÝCH ELEKTRONICKÝCH PŘÍLOH

B.1 Elektrické schéma

PortTester_ElectricSchematic.zip

- 1_PortTester_BlockScheme.pdf
- 2_PortTester_PowerSupply.pdf
- 3_PortTester_Port1.pdf
- 4_PortTester_Port2.pdf
- 5_PortTester_Port3.pdf
- 6_PortTester_Port4.pdf
- 7_PortTester_Port5.pdf
- 8_PortTester_Port6.pdf
- 9_PortTester_Port7.pdf
- 10_PortTester_Port8.pdf
- 11_PortTester_Outputs.pdf
- 12_PortTester_Inputs.pdf
- 13_PortTester_LED.pdf
- 14_PortTester_SignalIsolation.pdf
- 15_PortTester_CQIsolation.pdf
- 16_PortTester_Connectors.pdf

B.2 Program

PortTester_Program.zip

- static
    - commands.js
    - style.css
- templates
    - GUI.html
- flask_app.py
- PortTester_Interface.py