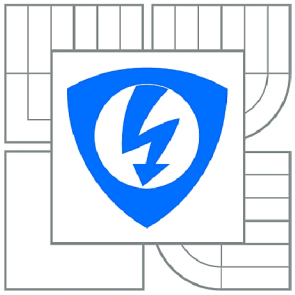
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

NÁVRH A REALIZACE OVLADAČE LCD DISPLAYE PRO RASPBERRY PI

DESIGN AND IMPLEMENTATION OF THE LCD DISPLAY DRIVERS FOR RASPBERRY PI

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

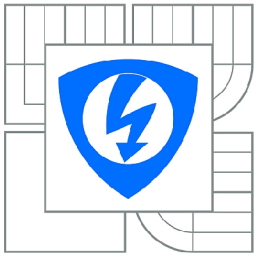
AUTOR PRÁCE — Ing. MIROSLAV CEPL

AUTHOR

VEDOUCÍ PRÁCE — Ing. ONDŘEJ KRAJSA, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Ing. Miroslav Cepl	***ID:***	10525
Ročník:	2	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Návrh a realizace ovladače LCD displaye pro Raspberry Pi

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte ovladač pro dotykový LCD display. Předpokladem je připojení displaye k mikrokontroleru pomocí sběrnice I2C a využití obvodů MCP23017.

DOPORUČENÁ LITERATURA:

[1] RICHARDSON, By Matt. Beginning BeagleBone: creating Linux-powered electronics projects. Farnham: O'Reilly. ISBN 978-144-9345-372.
[2] VENKATESWARAN, Sreekrishnan. Essential Linux device drivers. Upper Saddle River: Prentice Hall, c2008, xxx, 714 s. ISBN 978-0-132-39655-4
[3] KHAN, Ashfaq A. Practical Linux programming: device drivers, embedded systems and the Internet. Hingham: Charles River Media, c2002, xv, 420 s. ISBN 1-58450-096-4.

Termín zadání:	9.2.2015	***Termín odevzdání:***	26.5.2015

Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Cílem diplomové práce je navrhnout a realizovat ovladač pro dotykový LCD displej. Předpokladem je připojení displeje k mikrokontroleru pomocí sběrnice I2C a využití expandérů MCP23017.

Pro komunikaci bude použito Raspberry Pi B+, které bude komunikovat s obvody MCP23017 po sběrnici I2C. Display TFT_320QVT bude připojen k rozšiřujícím obvodům MCP23017. Součástí displaye je i dotykový display s řadičem XPT2046.

Na Raspberry Pi bude nainstalován operační systém Raspbian, který je založen na linuxové distribuci Debian. Ovladač bude sloužit ke komunikaci a zobrazení informací na displeji. Při kombinací s dotykovým displejem umožní aplikace dotykem ovládat například výběr nastavení zobrazených na displeji.

KLÍČOVÁ SLOVA

Raspberry Pi, MCP23017, I2C, LCD displej, TFT_320QVT.

ABSTRACT

The aim of diploma thesis is to design and implement a driver for touchscreen LCD display. The premise of the connection the display to the microcontroller using the I2C bus and use MCP23017 circuits.

Raspberry Pi B + will be used for communication with circuits MCP23017 via I2C bus. Display TFT_320QVT will be connected to the outputs of extension circuits MCP23017. Part of the display complete is also a touchscreen with XPT2046 controller.

Raspberry Pi runs on Raspbian system based on the Debian Linux distribution. The driver will serve to communicate and display information on the screen. The complete will be useable for applications such as selecting settings and show main information on the display.

KEYWORDS

Raspberry Pi, MCP23017, I2C, LCD display, TFT_320QVT

CEPL, M. *Návrh a realizace ovladače LCD displaye pro Raspberry Pi.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2015. 51 s. Vedoucí diplomové práce Ing. Ondřej Krajsa, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma Návrh a realizace ovladače LCD displaye pro Raspberry Pi jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Ondřeji Krajsovi , Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne ....

(podpis autora)

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Purkynova 118, CZ-61200 Brno, Czechia

http://www.six.feec.vutbr.cz

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Návrh a realizace ovladače LCD displeje pro Raspberry Pi. Raspberry Pi je mini počítač, který je založen na jednojádrovém procesoru ARM Broudcom. Mini počítač obsahuje 40 GPIO pinů, které umožňují komunikaci s okolními periferiemi. Ke komunikaci lze použít sériové sběrnice SPI a I2C.

Raspberry Pi umožňuje nainstalovat operační systém Android nebo operační systém založený na linuxu. Oba druhy systémů jsou poskytovány zdarma a dle svého sestavení a možnosti doinstalování softwarových rozšíření poskytují obrovské možnosti k využití. Výběrem vhod-ného operačního systému lze z Raspberry Pi udělat během několika minut HTPC, které umožňuje přehrávat filmy ve vysokém rozlišení (HDTV). Použitím operačního systému Ra-psbian, který je založen na operačním systému Debian, lze využít Raspberry Pi například pro ovládání domácnosti, jako zabezpečovací systém, aj.

Rozšířením Raspberry Pi o dotykový display umožňuje tato kombinace využití například pro ovládání systému řízení, zobrazení informací z připojených čidel, či zadávání vstupních údajů.

1 POPIS POUŽITÝCH SOUČÁSTÍ

1.1 Raspberry Pi

Raspberry Pi [1] je jednodeskový „mini“ počítač, který je postaven na architektuře ARM. Po-čítač je na trhu dostupný od roku 2011 [2]. Od té doby vzniklo několik variant, které se odlišují pracovní frekvencí procesoru, velikostí operační paměti, počtem USB portů, typem slotu pro přídavnou paměť, rozšiřujícími GPIO piny, aj.

Pro diplomovou práci bude použita varianta B+, která je dostupná od července 2014. Popis vlastností je uveden v tab. 1.

Tab. 1.1: Vlastnosti Raspberry Pi B+

Hardware	Popis
základní čip	Broadcom BCM2835
CPU model	ARM1176JZF-S core
Frekvence	700MHz
GPU	Broadcom VideoCore IV
	OpenGl ES 2.0
	MPEG-2, MPEG-4
	1080p30 H.264
	1 Gpixel/s, 1,5Gtexels/s
Video výstup	HDMI
	DSI displej konektor
Audio výstup	3.5 mm jack
	HDMI
Úložiště	micro SD
Periférie	40 GPIO pinů
	UART
	I2C sběrnice
	SPI sběrnice
	4 USB
	CSI konektor pro kameru
	LAN RJ45 10/100Mb/s
Napájení	5V micro-USB

GPIO port obsahuje:

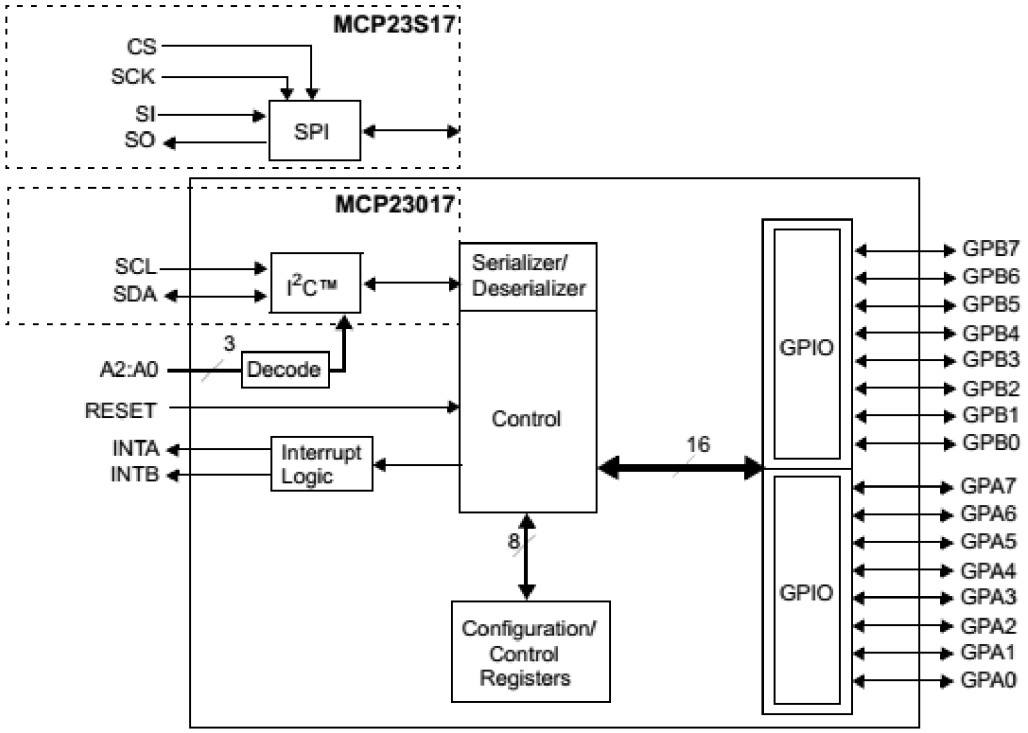
- Výstupní napětí na GPIO portu 3,3V a 5V.
- Sériový port I2C
- Jeden sériové port SPI s výběrem dvou čipů
- Sériový port RS232
- Možnost připojit paměť EEPROM na piny č. 27 a č. 28

Raspbery Pi má dostatečný výkon nejen pro řízení, ale i pro běh multimediálních aplikací, umožňující přehrávání multimédií v HDTV. Pro mini počítač jsou dostupné různé typy operačních systémů, které jsou převážně založeny na nějaké verzi linuxové distribuce. Výběrem operačního systému je možné Raspberry Pi používat v různých odvětvích, např. aplikace určené pro řízení nebo jako multimediální domácí centru. Přehled některých operačních systémů je uveden níže:

- Raspbian, který je založen na Debian Wheezy
- Pidora – Fedora Remix
- OpenELEC – Open Embedded Linux Entertainment Center
- Raspbmc – .Minimální linuxová distribuce založená na Debianu s předinstalovaným XBMC multimediálním přehrávačem.

1.2 Expandér MCP23017

MCP23017 obvod od firmy Microchip Technology Inc. poskytuje 16bitové rozšíření vstupně/výstupních portů, který komunikuje po sériové sběrnici. MCP23017 je připojeno po sběrnici I2C (obvod MCP23S17 komunikuje po sběrnici SPI). Expandér se skládá ze dvou 8bitových konfiguračních registrů. Každý z registrů může být nastaven jako výstup nebo vstup s možností výběru polarity. Rozlišení jestli se jedná o vstup nebo o výstup je zajištěno nastavením konfiguračních bitů I/O (IODIRA/B). Data pro každý vstup nebo výstup jsou uložena v odpovídajícím vstupním nebo výstupním registru. Polarita vstupního registru může být invertována s „Polarity Inversion“ registrem. Expandér MCP23017 obsahuje dva 8bitové porty (PORTA a PORTB), které lze provozovat v 16ti bitovém módu.



Obr. 1.1: Blokové schéma obvodu MCP23x17 (převzato z [5])

1.2.1 Sériové rozhraní

Pro komunikaci s okolím používá obvod MCP23017 sériovou sběrnici I2C a obvod MCP23S017 sběrnici SPI. Obvody obsahují 22 individuálních registrů (11 párů), které lze adresovat přes sériové rozhraní. Adresy a popis jednotlivých registrů popisuje Tab. 1.2.

Tab. 1.2: Adresy registrů MCP23017

Adresa IOCON.BANK = 1	**Adresa IOCON.BANK = 0**	**Přístup do**
00h	00h	IODIRA
10h	01h	IODIRB
01h	02h	IPOLA
11h	03h	IPOLB
02h	04h	GPINTENA
12h	05h	GPINTENB
03h	06h	DEFVALA
13h	07h	DEFVALB

04h	08h	INTCONA
14h	09h	INTCONB
05h	0Ah	IOCON
15h	0Bh	IOCON
06h	0Ch	GPPUA
16h	0Dh	GPPUB
07h	0Eh	INTFA
17h	0Fh	INTFB
08h	10h	INTCAPA
18h	11h	INTCAPB
09h	12h	GPIOA
19h	13h	GPIOB
0Ah	14h	OLATA
1Ah	15h	OLATB

Pro diplomovou práci bude využíván převážně registr IODIR a OLAT. Registrem IODIR se nastavuje směr komunikace. Pro konfiguraci pinu jako výstupní odpovídá úroveň logická 0 a pro konfiguraci pinu jako vstupní odpovídá úroveň logická 1. Zápisem do registru OLAT se modifikuji hodnoty pinů, které jsou konfigurovány jako výstup.

Obvody MCP23x17 mají dva režimy komunikace po sériové sběrnici:

- Byte mód
- Sekvenční mód

Byte mód neinkrementuje ukazatel na adresu, z které se má číst nebo do které zapisovat. Pokud probíhá komunikace v byte módu, MCP23x17 v průběhu datového přenosu neinkrementuje po každém bytu interní adresové počítadlo. Tímto je možné neustále přistupovat ke stejné adrese, bez dalších kontrolních bytů. Tento přístup je užitečný pokud je třeba zjišťovat změny dat v GPIO registrech nebo pro kontinuální ukládání na výstupní zařízení.

Speciální mód (Byte mód s nastaveným IOCON.BANK = 0) přepíná adresní ukazatel mezi asociovanými A/B registrovými páry. Například pokud je BANK bit vynulován a adresní ukazatel je nastaven na adresu 12h (GPIOA) nebo 13h (GPIOB), adresový ukazatel se bude postupně měnit mezi GPIOA a GPIOB. Adresní ukazatel může inicializovat obě adresy z daného registrového páru.

Sekvenční mód automaticky inkrementuje adresní ukazatel. V sekvenčním módu dochází v průběhu komunikace k inkrementaci adresního ukazatele automaticky po každém

přijatém nebo vyslaném bytu. Adresní ukazatel je automaticky nulován po přístupu k poslednímu registru.

Použití byte módu nebo sekvenčního módu záleží na požadovaném chování aplikace. Například pokud je MCP23x17 nakonfigurován pro byte mód, Master může sekvenčně číst. V tomto případě nedojde k inkrementaci adresního ukazatele a z MCP23x17 bude stále číst data ze stejné adresy.

1.2.2 I2C rozhraní

Procedura zápisu po I2C obsahuje:

Kontrolní byte, Opcode zařízení, potvrzovací signál (ACK) z MCP23017, adresu registru pro zápis, potvrzovací signál (ACK) z MCP23017 a 8 bitů dat. Operace zápisu je ukončena Stop nebo Restart, která je generována Masterem. Data do MCP23017 jsou zapsána vždy po přenosu celého bytu. Pokud v průběhu přenosu vygeneroval Master Stop nebo Reset, data nebudou do MCP23017 zapsána. Zápis může být po jednom nebo více bytech. Pokud jde o sekvenční zápis, který je v základním nastavení povolen (IOCON a SEQOP jsou nastaveny na 0), MCP23017 inkrementuje čítač adres po každém potvrzovacím signálu ACK.

Procedura čtení obsahuje:

Kontrolní byte, obsahující Start bit a potvrzovací signál (ACK) s nastaveným R/W bitem (R/W = 1). V dalším kroku MCP23017 pošle data uložená v adresním registru. Sekvence čtení je zakončena vygenerovaným signálem Stop nebo Reset od mastera.

Popis vlastností jednotlivých pinů je zobrazen v tab. 1.3.

Tab. 1.3: Popis pinů MCP23017

Název pinu	Číslo pinu	Typ pinu	Funkce
GPB0	1	I/O	Obousměrný I/O pin. Možnost nastavit na přerušení při změně a/nebo jako interní pull-up odpor
GPB1	2	I/O	Obousměrný I/O pin. Možnost nastavit na přerušení při změně a/nebo jako interní pull-up odpor
GPB2	3	I/O	Obousměrný I/O pin. Možnost nastavit na přerušení při změně a/nebo jako interní pull-up odpor
GPB3	4	I/O	Obousměrný I/O pin. Možnost nastavit na přerušení při změně a/nebo jako interní pull-up odpor
GPB4	5	I/O	Obousměrný I/O pin. Možnost nastavit na přerušení při změně a/nebo jako interní pull-up odpor
GPB5	6	I/O	Obousměrný I/O pin. Možnost nastavit na přerušení při změně a/nebo jako interní pull-up odpor
GPB6	7	I/O	Obousměrný I/O pin. Možnost nastavit na přerušení při změně a/nebo jako interní pull-up odpor
GPB7	8	I/O	Obousměrný I/O pin. Možnost nastavit na přerušení při změně a/nebo jako interní pull-up odpor
VDD	9	P	Napájení - Power
Vss	10	P	Zem - Ground
NC/CS	11	I	NC (MCP23017), Chip Select (MCP23S17)
SCL/SCK	12	I	Vstup pro hodinový signál - Serial clock input
SDA/SI	13	I/O	Vstup pro hodinový signál - Serial clock input
NC/SO	14	O	NC (MCP23017), Serial data out (MCP23S17)
A0	15	I	Pin pro nastavení hardwarové adresy
A1	16	I	Pin pro nastavení hardwarové adresy
A2	17	I	Pin pro nastavení hardwarové adresy
RESET	18	I	Hardware reset. Must be externally biased.
INTB	19	O	Výstup přerušení pro PORTB.
INTA	20	O	Výstup přerušení pro PORTA.
GPA0	21	I/O	Obousměrný I/O pin. Možnost nastavit na přerušení při změně a/nebo jako interní pull-up odpor
GPA1	22	I/O	Obousměrný I/O pin. Možnost nastavit na přerušení při změně a/nebo jako interní pull-up odpor
GPA2	23	I/O	Obousměrný I/O pin. Možnost nastavit na přerušení při změně a/nebo jako interní pull-up odpor
GPA3	24	I/O	Obousměrný I/O pin. Možnost nastavit na přerušení při změně a/nebo jako interní pull-up odpor
GPA4	25	I/O	Obousměrný I/O pin. Možnost nastavit na přerušení při změně a/nebo jako interní pull-up odpor
GPA5	26	I/O	Obousměrný I/O pin. Možnost nastavit na přerušení při změně a/nebo jako interní pull-up odpor
GPA6	27	I/O	Obousměrný I/O pin. Možnost nastavit na přerušení při změně a/nebo jako interní pull-up odpor
GPA7	28	I/O	Obousměrný I/O pin.

1.2.3 Adresace MCP28017

Obvod MCP23017 komunikuje po sběrnici I2C jako Slave. Pro hardwarovou adresaci obvodu je v kontrolním bytu vyhrazeno 7bitů. Adresa zařízení obsahuje 4 bity pevně dané a 3 bity, které jsou uživatelsky nastavitelné. Jedná se o hardwarové bity – piny A2, A1 a A0. Nastavení těchto bytů je dáno přivedením jednotlivých pinů na zem (logická 0) nebo k napájení obvodu (logická 1).

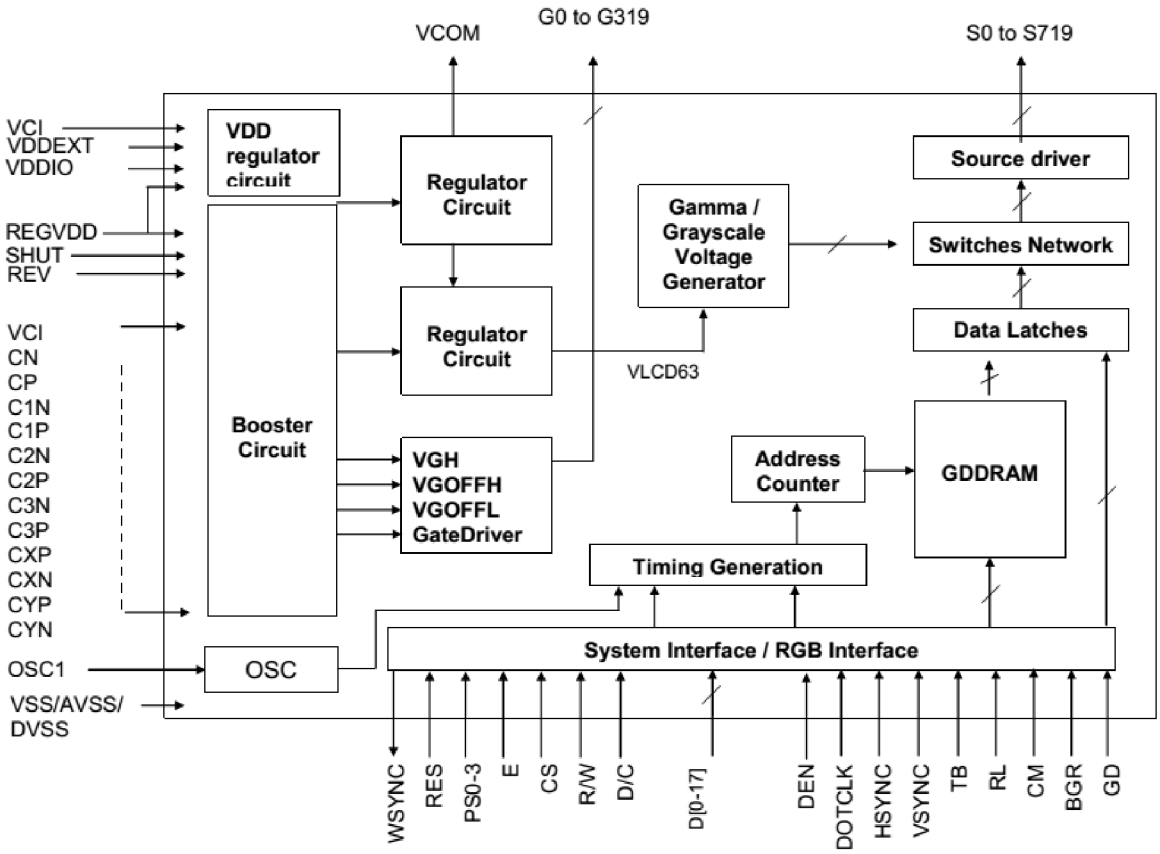
Obvod může komunikovat na adrese 20h (připojením všech tří pinů na zem) až 27h (připojením všech tří pinů ke zdroji napájení obvodu). Tímto způsobem lze adresovat až osm zařízení. Každé z osmi zařízení rozšíří počet I/O pinů o 16. Celkově je možné obslu-hovat až 128 rozšiřujících pinů.

Tab. 1.4: Nastavení adresy expandéru MCP23017

Adresa	**Piny**			**i2c adresa**
	A2	**A1**	**A0**	
000	zem	zem	zem	0x20
001	zem	zem	3,3 V	0x21
010	zem	3,3 V	zem	0x22
011	zem	3,3 V	3,3 V	0x23
100	3,3 V	zem	zem	0x24
101	3,3 V	zem	3,3 V	0x25
110	3,3 V	3,3 V	Zem	0x26
111	3,3 V	3,3 V	3,3 V	0x27

1.3 SSD1289 TFT řadič

SSD1289 TFT řadič displeje umožňuje připojení displeje o rozlišení 240x320 bodů s 262k barev. Řadič integruje napájecí obvody, řídící obvody, zobrazovací obvody i paměť GDDRAM. SSD1289 obsahuje DC-DC měnič a napěťový generátor, které zabezpečují nutná provozní napětí. Kompletní blokové schéma je znázorněno na Obr. 1.2: Blokové schéma TFT řadiče SSD1289 (převzato z [6])Obr. 1.2.

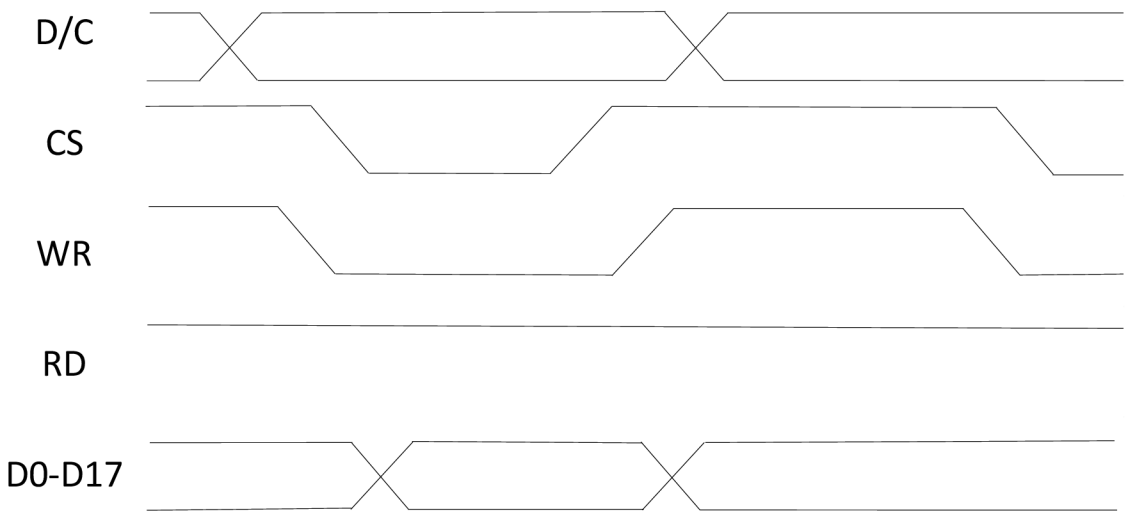


Obr. 1.2: Blokové schéma TFT řadiče SSD1289 (převzato z [6])

Systémový řadič sestává ze tří funkčních bloků pro řízení paralelního rozhraní řady 6800 (Motorola), 8080 (Intel), 3vodičového sériového rozhraní a 4vodičového sériového rozhraní.

Pro potřeby této práce bude využita komunikace paralelního rozhraní řady 8080.

Paralelní rozhraní obsahuje následující piny: 18 obousměrných datových, RD (čtení), WR (zápis), D/C (data/příkaz) a CS. Časový průběh zápisu zobrazuje obr. 1.4.



Obr. 1.3: Časový průběh zápisu do SSD1289

Zápis obrazových dat do GDDRAM paměti řadiče je po jednom obrazovém bodu (pixelu). Řadič umožňuje adresovat až 262 tisíc nebo 65 tisíc barevnou hloubku. Barevná hloubka se nastavuje v registru 0x11 bit DFM0 a DFM1. Pro poslání obrazových dat lze vybrat ze čtyřech módů posílaných sekvencí bitů: 18ti, 16ti, 9ti a 8mi bitový.

Pro diplomovou práci použijeme dotykový displej TFT320QVT s rozlišením 320x240 bodů a 65 tisíci barvami. Vzhledem k použitému typu dotykového displeje budeme využívat 16bitové rozhraní s 65536 barvami. Data posíláme ve formátu RGB565 (5bitů pro červenou, 6 bitů pro zelenou a 5 bitů pro modrou barvu). Ukázka formátu barev je v Tab. 1.7. Adresace rozhraní zadávání instrukce při 16bitovém módu je zobrazena v Tab. 1.5 Zápis každého obrazového bodu je 16bitový. Adresace je v tab. 1.6.

Tab. 1.5: Adresace rozhraní – zápis obrazových instrukcí – 16ti bitový mód

Hardwarové piny								
D17	D16	D15	D14	D13	D12	D11	D10	D9
IB15	IB14	IB13	IB12	IB11	IB10	IB9	IB8	
D8	D7	D6	D5	D4	D3	D2	D1	D0
IB7	IB6	IB5	IB4	IB3	IB2	IB1	IB0	

Tab. 1.6: Adresace rozhraní – zápis pixel dat 16ti bitový mód, 65 tisíc barev

Hardwarové piny								
D17	D16	D15	D14	D13	D12	D11	D10	D9
R4	R3	R2	R1	R0	G5	G4	G3	
D8	D7	D6	D5	D4	D3	D2	D1	D0
G2	G1	G0	B4	B3	B2	B1	B0	

Tab. 1.7: Tabulka barev RGB565

Barva	Název	Hex hodnota	Hex hodnota
		RRGGBB	RGB565
	Black	0x000000	0x0000
	White	0xFFFFFF	0xFFFF
	Red	0xFF0000	0xF800
	Lime	0x00FF00	0x07E0
	Blue	0x0000FF	0x001F
	Yellow	0xFFFF00	0xFFE0
	Cyan	0x00FFFF	0x07FF
	Magenta	0xFF00FF	0xF81F
	Silver	0xC0C0C0	0xC618
	Gray	0x808080	0x8410
	Maroon	0x800000	0x8000
	Olive	0x808000	0x8400
	Green	0x008000	0x0400
	Purple	0x800080	0x8010
	Teal	0x008080	0x0410
	Navy	0x000080	0x0010

1.4 Displej TFT_320QVT

Displej TFT_320QVT je 3,2“ TFT dotykový displej, který je připojen k řadiči SSD1289. Barevná hloubka 65 tisíc barev, napájení 3,3V. Displej se připojuje konektorem typu TFT40. Jednotlivé piny jsou zobrazeny v Tab. 1.8.

Tab. 1.8: Konektor displeje TFT_320QVT

Název a číslo pinu		Název a číslo pinu	
GND	1	21	DB0
VCC	2	22	DB1
NC	3	23	DB2
DC	4	24	DB3
WR	5	25	DB4
RD	6	26	DB5
DB8	7	27	DB6
DB9	8	28	DB7
DB10	9	29	T_CLK
DB11	10	30	T_CS
DB12	11	31	T_DIN
DB13	12	32	NC
DB14	13	33	T_DO
DB15	14	34	T_IRQ
CS	15	35	SD_DO
F_CS	16	36	SD_CLK
REST	17	37	SD_DIN
NC	18	38	SD_CS
LED_A	19	39	NC
NC	20	40	NC

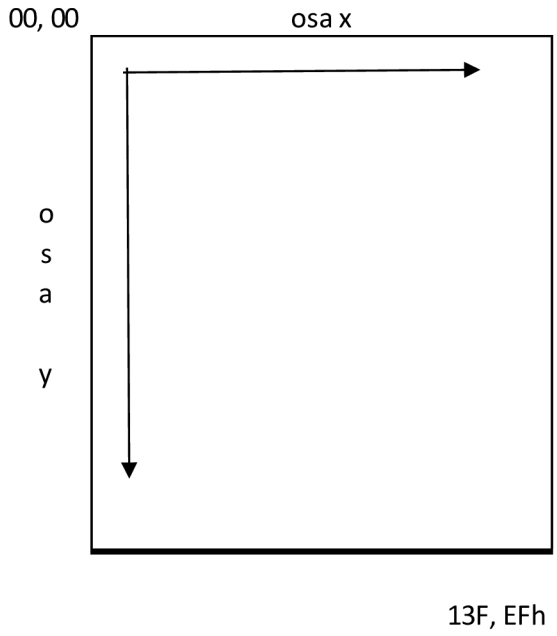
Konektor TFT40 obsahuje následující piny:

- DB0 – DB15 datové piny
- RS (DC) rozlišení data nebo příkaz
- WR zápis
- RD čtení
- CS chip select
- REST reset

- T_ piny pro správu dotykového displeje
- SD_ piny pro práci s paměťovou kartou

DB0 až DB15 slouží na datovou komunikaci, RS (DC) se rozlišuje, jestli se bude přenášet data nebo příkaz (v logické 0 se jedná o zápis příkazu, v logické 1 o zápis nebo čtení dat), WR zápis dat do paměti řadiče (aktivní v logické 0), RD čtení dat z paměti řadiče (aktivní v logické 0), CS výběr čipu (aktivní v logické 0) a RESET slouží k reset displeje (aktivní v logické 0).

Souřadnice displeje jsou zobrazeny na obrázku Obr. 1.4. Pro aplikaci budeme používat stejný souřadnicový systém.



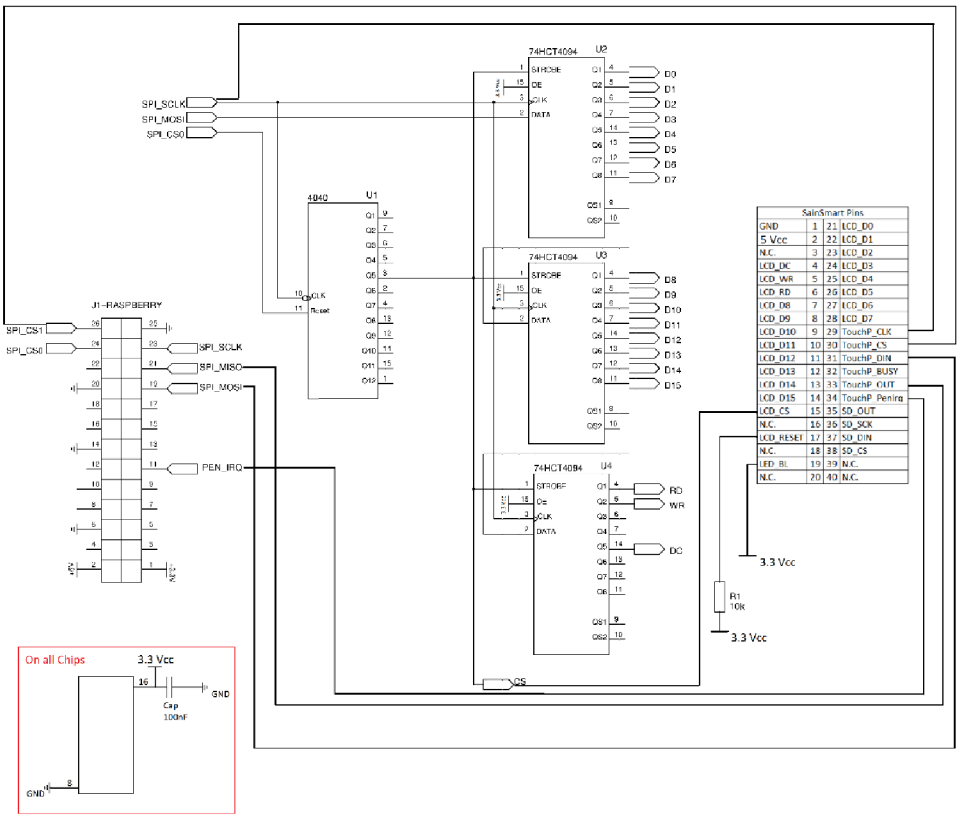
Obr. 1.4: Souřadnicový systém displeje

2 SOUČASNÁ ŘEŠENÍ

Ke komunikaci s displejem existuje několik řešení. Omezujícím faktorem je počet pinů, které jsou potřeba pro připojení displeje. Pro zobrazovací část displeje je třeba 16 datových pinů (DB0 – DB15) a 5 pinů řídících (RS pro rozlišení jestli se jedná o data nebo příkaz, WR pin pro zápis, RD pin pro čtení, CS výběr čipu a REST reset). Pro připojení dotykové části displeje je třeba 5 pinů (T_CLK, T_CS, T_DIN, T_DO a T_IRQ. Celkově je tedy třeba 27 pinů.

Současná řešení využívají pro komunikaci mezi Raspberry Pi a displejem přídavný modul Arduino MEGA [7]. Arduino je založeno na mikropočítači ATMega. Pro dostatečný počet vývodů je displej propojen s Arduinem. Komunikace mezi Raspberry Pi a displejem je po sběrnici SPI.

Dalším řešením, které je možné na internetu najít je využít integrované obvody posuvných registrů [8]. K rozšíření počtu výstupních pinů jsou použity tři posuvné registry 74HCT4094.



Obr. 2.1: Schéma zapojení s využitím posuvných registrů (převzato z [8])

Frekvence komunikace po sériové sběrnici I2C lze nastavit na 100kHz, 400kHz nebo 1,7MHz. Na jeden expandér MCP23017 budou připojeny datové signály displeje DB0 – DB15. Protože lze adresovat pouze poloviny expandéru (8 bitů), budou řídící signály displeje připojeny na PORTA. Při vykreslování na displej není třeba překreslovat celý displej, ale lze vybrat pouze určitou část zadáním adresy ve formátu xx a yy. Tím lze minimalizovat velikost přenášených dat do displeje.

Pro komunikaci s displejem bude třeba nadefinovat:

- Řídicí signály displeje a k nim odpovídající řídící pin
- DC – data nebo řídicí (v některých zapojeních označen jako RS registr select)
- WR – zápis
- CS – chip select
- REST – reset displeje
- Funkce pro výběr řídícího registru
- Funkce pro zápis dat do registru
- Inicializaci registrů řadiče displeje
- Pro zápis obrazových dat definovat výstupní okno, do kterého se bude zobrazovat
- Funkce pro vymazání části nebo celého displeje

3 NÁVRH VLASTNÍHO ŘEŠENÍ

Vlastní řešení vychází ze zadání diplomové práce. Navrhnout a realizovat ovladač pro dotykový LCD displej TFT_320QVT. Displej ovládáme počítačem Raspberry Pi B+ přes dva expandéry MCP23017 připojené na sběrnici I2C. Dotyková součást LCD displeje s řadičem XPT2046 je připojena přímo k Raspberry Pi přes SPI sběrnici.

Vlastní řešení sestává z následujících komponent:

- Raspberry Pi B+
- 2 obvody MCP23017
- TFT displej s řadičem SSD1289
- Dotykový displej s řadičem XPT2046

Expandéry připojíme k Raspberry Pi sběrnicí I2C – piny I2C1_SDA a I2C1_SCL. Výstup z jednoho expandéru MCP23017 připojíme na datové konektory LCD displeje DB0 – DB15. Expandér nazveme datovým a pro další pojmenování budeme používat adresový MCP1. Druhý expandér MCP23017 pojmenujme řídícím a dále ho označujme jako řídící MCP2. Na tento expandér připojíme následující řídící piny displeje LCD:

- DC – data nebo příkaz (v některých zapojeních označen jako RS registr select)
- CS – chip select
- WR – signál pro zápis
- REST – pin pro reset displeje

Dotyková část displeje je připojena sériovou sběrnicí SPI s Raspberry Pi:

- T_CLK – hodinový signál (Raspberry Pi pin 23 SPI_CLK)
- T_CS – chip select dotykového displeje (Raspberry Pi pin 26 SPI_CE1)
- T_DIN – vstup dat (data in – Raspberry Pi pin 19 SPI_MOSI)
- T_DO – výstup dat (data out –Raspberry Pi pin 21 SPI_MISO)
- T_IRQ – detekce dotyku (Raspberry Pi pin 25 GPIO25)

Propojením využijeme 7 GPIO pinů u Raspberry Pi. Pro celkové zapojení potřebujeme následující počet propojení:

- 2 piny pro sběrnici I2C (propojeni Raspberry Pi a expandérů MCP23017)
- 5 pinů pro sběrnici SPI (dotykový displej)

- 16 datových pinů – propojení expandérů a řadiče LCD displeje
- 5V z kterého získáme usměrňovačem 3,3V pro napájení expandérů MCP23017 a LCD displeje

Hlavní výhodou tohoto řešení je, že získáme volný jeden kanál rychlé sériové sběrnice SPI, na kterou můžeme připojit další libovolné zařízení. Připojení přes sériovou sběrnici I2C je násobně pomalejší, než připojení přes sériovou sběrnici SPI. Navržené řešení se hodí pro aplikace, které nejsou náročné na rychlost překreslování displeje. Displej připojený přes sériovou sběrnici I2C můžeme využít například v aplikacích pro ovládání, které nahradí například numerickou klávesnici. Na displeji si můžeme nechat vypisovat libovolné informační údaje.

Raspberry Pi s nainstalovaným operační systém Raspbian, který je založený na linuxové distribuci Debian. Komunikace s obvody MCP23017 je po sériové sběrnici I2C. Na výstupní porty expandéru MCP23017 je připojen displej. Ovladač je naprogramován v jazyce C.

3.1 Návrh zapojení

Raspberry Pi je napájen 5V přes konektor micro USB. Komunikaci s okolím zajišťují GPIO porty, které mohou být připojeny maximálně na 3,3 V. Napájecí napětí displeje je též pouze 3,3V. Přivedení vyššího napětí než 3,3V na GPIO porty nebo porty LCD displeje může dojít k trvalému poškození vstupně/výstupních pinů těchto zařízení. Protože komunikace s Raspberry Pi a dotykovým LCD displejem musí být při napětí 3,3V, jsou i oba expandéry MCP23017 napájeny napětím 3,3V. Tím je na všech I/O portech zaručená hodnota napětí 3,3V a nemůže dojít ke zničení I/O portů.

Napájení 3,3V řeším připojením externího 5V adaptéru, který je připojen přes usměrňovač LF33CV. Tím nezatěžuji výstupní svorky napětí 3,3V u Raspberry Pi, které by výkonovým nárokům všech zapojených obvodů nestačily.

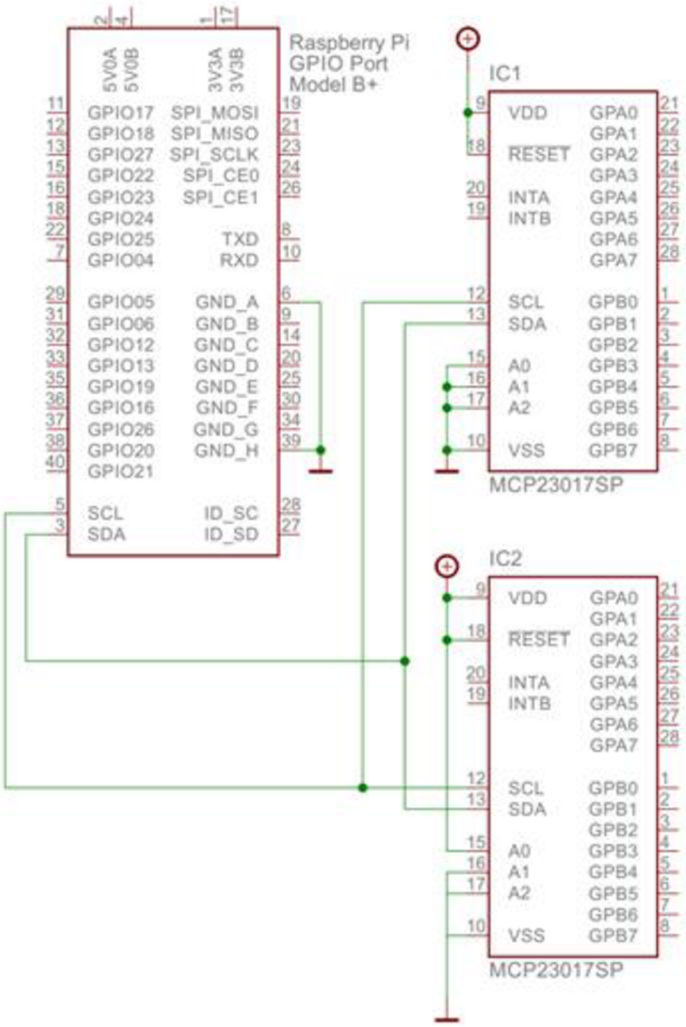
Na port A a port B expandéru MCP1 (GPA0 – GPA7 a GPB0 – GPB7) připojíme displej (piny DB0 – DB15). Dolní bity na PORTA a horní bity na PORTB. Řídicí signály displeje (DC, WR, CS a REST) připojíme na výstupy portu A obvodu MCP2 (GPA0 – GPA3).

Dotykový displej (touchscreen) připojíme k Raspberry Pi sběrnicí SPI. Jednotlivé piny zapojení jsou uvedeny v tabulce Tab. 3.1.

Tab. 3.1: Popis propojení touchscreenu s Raspberry Pi

Touchscreen	Raspberry Pi	Pin wPi
T_CLK	SCLK – 11	14
T_CS	CE1 – 7	11
T_DIN	MOSI – 10	12
T_DO	MISO – 9	13
T_IRQ	GPIO6 - 25	6

Schéma návrhu zapojení:



Obr. 3.1: Návrh zapojení

3.2 Komunikace Rapberry Pi a expandéru MCP23017

Pro komunikaci a ovládání GPIO Raspberry Pi použijeme knihovnu WiringPi **Chyba! Nenalezen zdroj odkazů.**. Knihovna je psaná i v jazyce C pod licencí GNU LGPLv3 a obsahuje příkazy pro ovládání GPIO portů Raspberry Pi. Obsahuje i utilitu příkazového řádku `gpio`, kterou můžeme použít pro nastavení a programování GPIO pinů. Knihovnu můžeme dále rozšířit o podporu pro ovládání námi použitých expandérů MCP23017.

3.2.1 Knihovna WiringPi

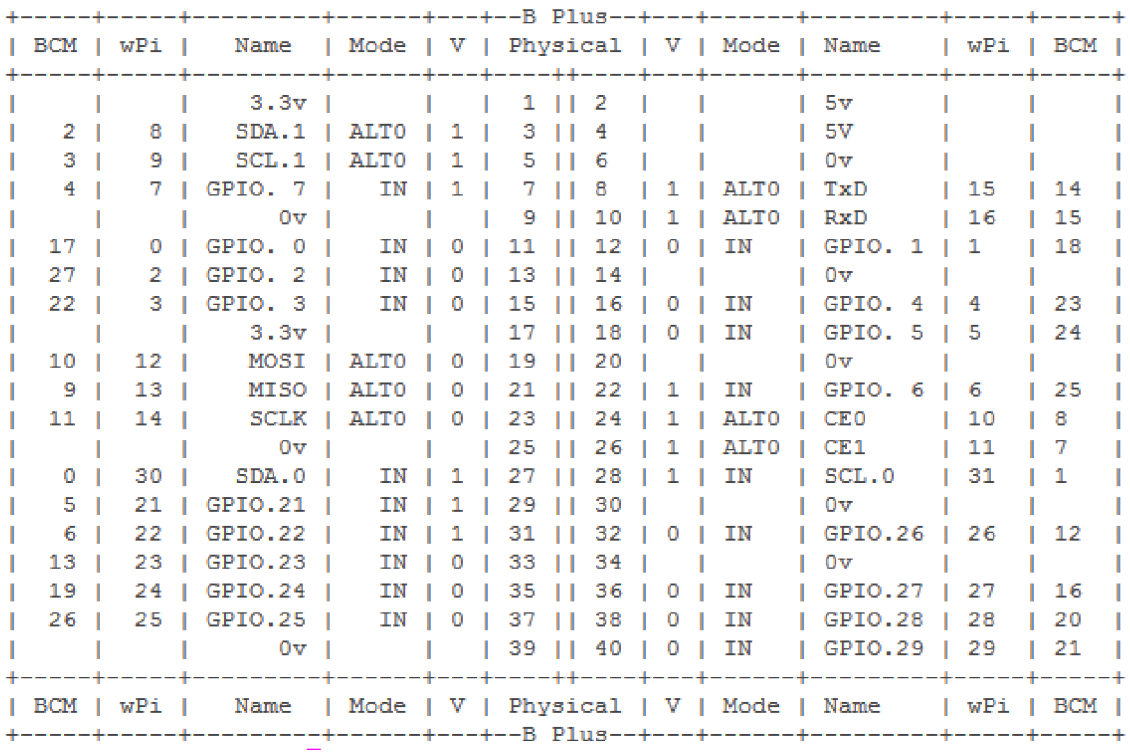
Knihovnu WiringPi naistalujeme přímo z příkazové řádky v aktuální verzi Raspbianu příkazem

```
sudo apt-get install git-core
```

Po úspěšné instalaci si můžeme příkazem `gpio -v` nechat vypsat základní informace o nainstalované verzi WiringPi a jakou verzi Raspberry Pi používáme.

```
pi@raspberrypi ~ $ gpio -v
gpio version: 2.26
Copyright (c) 2012-2015 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
  Type: Model B+, Revision: 1.2, Memory: 512MB, Maker: Sony
```

Seznam a bližší informace o jednotlivých GPIO pinech Raspberry Pi získáme příkazem `gpio readall`, který nám poskytne seznam viz Obr. 3.2. Pro využití funkcí knihovny WiringPi nás bude dále zajímat sloupec wPi, kde jsou číselné hodnoty jednotlivých pinů, ke kterým lze pomocí knihovny přistupovat.



```
+-----+-----+---------+------+---+--B Plus--+---+------+---------+-----+-----+
| BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
+-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
|     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
|   2 |   8 |   SDA.1 | ALT0 | 1 |  3 || 4  |   |      | 5V      |     |     |
|   3 |   9 |   SCL.1 | ALT0 | 1 |  5 || 6  |   |      | 0v      |     |     |
|   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT0 | TxD     | 15  | 14  |
|     |     |      0v |      |   |  9 || 10 | 1 | ALT0 | RxD     | 16  | 15  |
|  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
|  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
|  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
|     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
|  10 |  12 |    MOSI | ALT0 | 0 | 19 || 20 |   |      | 0v      |     |     |
|   9 |  13 |    MISO | ALT0 | 0 | 21 || 22 | 1 | IN   | GPIO. 6 | 6   | 25  |
|  11 |  14 |    SCLK | ALT0 | 0 | 23 || 24 | 1 | ALT0 | CE0     | 10  | 8   |
|     |     |      0v |      |   | 25 || 26 | 1 | ALT0 | CE1     | 11  | 7   |
|   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
|   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
|   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
|  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
|  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
|  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
|     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
+-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
| BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
+-----+-----+---------+------+---+--B Plus--+---+------+---------+-----+-----+
```

Obr. 3.2: Výpis GPIO pinů Raspberry Pi příkazem gpio readall

3.2.2 Komunikace s expandéry MCP23017

Každému z expandérů můžeme hardwarově nastavit adresu v hexadecimálním rozsahu 0x20 – 0x27. Adresu nastavujeme vhodným připojením pinů A2, A1 a A0. Celkově můžeme připojit až osm expandérů MCP23017, což nám umožňuje ve výsledku rozšíření až o 128 GPIO pinů.

Pro naše zapojení využijeme adresy 0x20 a 0x21. Seznam adres aktuálně připojených zařízení ke sběrnici i2c si ověříme příkazem `gpio i2cdetect`. Na obrázku Obr. 3.3 vidíme výpis připojených expandérů MCP23017, kterým jsme nastavili hexadecimální adresy 0x20 a 0x21.

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: 20 21 -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Obr. 3.3: Výpis připojených zařízení na sběrnici i2c

Po zjištění adres jednotlivých připojených expandérů s nimi začneme komunikovat v programovacím jazyce C. Pro kompilaci programu, např. gcc, nesmíme zapomenout přidat parametr -lwiringPi. Následuje ukázka a popis komunikace.

```
#include <wiringPi.h>
#include <mcp23017.h>
#include <wiringPiI2C.h>

#define PIN_BASE        100     // vychozi adresa pinu
#define PIN_DC          100     // DC (Data/Command) pin
#define PIN_WR          101     // WR (Write) pin
#define PIN_CS          102     // CS (Chip Select) pin
#define PIN_RESET       103     // RESET pin

wiringPiSetup();            // pouziti digitalRead, ...
mcp1 = wiringPiI2CSetup(MCP1);    // zadava se adresa i2c zarizeni
mcp2 = wiringPiI2CSetup(MCP2);
mcp23017Setup (PIN_BASE, MCP2);   // pro 2. MCP23017 piny 100 - 115
wiringPiI2CWriteReg16(mcp1, IODIRA, 0); // nastaveni MCP na vystup
wiringPiI2CWriteReg16(mcp2, IODIRA, 0); // nastaveni MCP na vystup

void spin(char pin) {
        digitalWrite(pin, 1);   // zapise na pin log. 1
}
void cpin(char pin) {
        digitalWrite(pin, 0);   // zapise na pin log. 0
}
```

V prvním kroku přidáme knihovny wiringPi.h, mcp23017.h a wiringPiI2C.h, které jsou nezbytné v našem projektu. Tímto získáme možnost využít připravené funkce pro jednodušší komunikaci s expandéry MCP23017.

Funkcí `wiringPiSetup();` inicializujeme využití wiringPi knihovny a získáme přístup ke GPIO pinům. Tato funkce musí být volána s rootovským oprávněním. Funkcí `wiringPiI2CSetup()` inicializujeme oba expandéry. MCP1 obsahuje adresu datového expandéru 0x20 a MCP2 adresu řídícího expandéru 0x21. Tím máme inicializované oba expandéry a můžeme s nimi dále komunikovat. Funkcí `mcp23017Setup()` získáme možnost komunikovat s jednotlivými piny řídícího expandéru od virtuálního čísla portu 100, který nastavíme v PIN_BASE. Funkcí `wiringPiI2CWriteReg16(, IODIRA, 0)` zapíšeme do registru IODIRA a IODIRB expandéru MCP1 a MCP2 hodnotu 0. Tento příkaz nastaví všechny porty expandéru jako výstupy. Tím získáme možnost posílat do řadiče LCD displeje data a řídícími signály ho ovládat.

Jednotlivé piny ovládáme funkcí `spin` a `cpin`. Funkcí `spin` nastavíme zadaný pin na úroveň logická 1 a funkcí `cpin` tento pin nastavíme na úroveň logické 0. Například pro nastavení řídícího signálu DC pro data použijeme `spin(PIN_DC)`.

3.3 Komunikace s řadičem LCD displeje

V předešlé kapitole jsme nastavili komunikaci mezi Raspberry Pi a expandéry MCP23017. Tato komunikace nám poslouží i k ovládání řadiče LCD displeje. S řadičem komunikujeme následovně:

- Nastavíme řídící signály
- Na první (datový) expandér pošleme adresu registru nebo data pro LCD řadič
- Na druhém expandéru aktivujeme zápis do řadiče LCD

Funkce pro zápis dat do řadiče provedeme následujícím příkazem:

```
void LCD_WRITE_COM (word com) {
        cpin(PIN_DC);  // nastavi DC = 0 => prikaz (Command)
        cpin(PIN_CS);           // aktivuje CS SSD1289
        wiringPiI2CWriteReg16(mcp1, OLATA, com); // zapis data
        puls_high(PIN_WR); // nabeznou hranou zapis do radice LCD
        spin(PIN_CS);           // deaktivace CS
```

Zápis dat provedeme nastavením pinu DC (PIN_DC) do logické hodnoty 0, čímž říkáme řadiči, že další informace na datových pinech bude příkaz. Nastavením pinu CS (PIN_CS) aktivujeme řadič LCD. Příkazem wiringPiI2CWriteReg16(mcp1, OLATA, com) zapíšeme na výstupy expandéru 1 adresu příkazu pro řadič. Následuje načtení těchto dat do řadiče. Zápis dat/příkazu do řadiče probíhá při CS = 0 s náběžnou hranou signálu WR (PIN_WR). Tím jsme do řadiče zaslali příkaz, po kterém následuje zaslání dat. Nakonec deaktivujeme CS (nastavíme hodnotu na logickou úroveň 1).

3.4 Základní funkce ovladače

Základní funkce pro ovládání LCD displeje můžeme rozdělit do následujících kategorií.
Funkce pro ovládání řídících signálů

```
void spin(char pin);         // vystupni pin na "1" - set PIN
void cpin(char pin);         // vystupni pin na "0" - clear PIN
void puls_low(char pin);     // sestupna hrana
void puls_high(char pin);    // nabezna hrana
```

Funkce pro zápis do řadiče displeje SSD1289

```
void LCD_WRITE_COM (word com);          // zapise prikaz do LCD
void LCD_WRITE_DATA (word data);        // zapise data do LCD
void LCD_WRITE_COM_DATA (word com, word data); // zapis prik a dat
```

Funkce pro nastavení displeje

```
void initLCD(void);
void setxy(word ax1, word ay1, word ax2, word ay2);
void clrscr(word color);
```

Funkce pro vykreslení základních prvků

```
void setPixel(word color);
void drawPixel(word x, word y, word color);
void drawHLine(word ax1, word ax2, word ay1, word color, byte th);
void drawVLine(word ay1, word ay2, word ax1, word color, byte th);
```

Funkce pro vykreslení rovnoběžníků

```
void drawRect(word ax1, word ay1, word ax2, word ay2, word color);
void drawRectLine(word ax1, word ay1, word ax2, word ay2, word color,
byte th);
void drawRectFill(word ax1, word ay1, word ax2, word ay2, word
bcolor, word lcolor, byte th);
void drawRectText(char *string, word ax1, word ay1, word ax2, word
ay2, word bcolor, word lcolor, byte orient);
```

Funkce pro vykreslení textu

```
void printChar(char ch, word x, word y, word chcolor, word bgcolor,
byte orient);
void printText(char *str, word x, word y, word chcolor, word bgcolor,
byte orient);
```

Funkce pro vykreslení obrázku

```
void printPix (word x, word y, word xr, word yr, long *pictures,
byte orient);
```

Funkce pro dotykový displej

```
void t_setup(unsigned char t_cs);       // inicializace t_displeje
int t_touch(void);                      // zjisteni dotyku
void t_read (unsigned char precission); // nacteni souradnic dotyku
```

3.5 Popis funkcí

V této kapitole popíšeme jednotlivé funkce, jejich použití a parametry

Funkce pro ovládání řídících signálů

```
void spin(char pin);        // výstupní pin na "1" - set PIN
void cpin(char pin);        // výstupní pin na "0" - clear PIN
void puls_low(char pin);    // sestupná hrana
void puls_high(char pin);   // náběžná hrana
```

Parametr: pin: číslo nebo označení pinu

Použití: provede funkci na zadaném pinu

Funkce pro zápis do řadiče displeje SSD1289

```
void LCD_WRITE_COM (word com);
```

Parametr: com: adresa příkazu

funkce zapíše příkaz do LCD řadiče

```
void LCD_WRITE_DATA (word data);
```

Parametr: data: hodnota dat

Použití: funkce zapíše data do LCD řadiče

```
void LCD_WRITE_COM_DATA (word com, word data);
```

Parametr: com: adresa příkazu

data: data

Použití: funkce zapíše data a následně data do LCD řadiče

Funkce pro nastavení displeje

```
void initLCD(void);
```

Použití: inicializace dipsleje

```
void setxy(word ax1, word ay1, word ax2, word ay2);
```

Parametry: ax_1, ay_1: počáteční souřadnice x_1, y_1

ax_2,ay_2: cílové souřadnice x_2, y_2

Použití: v paměti řadiče nastaví souřadnice od x_1, y_1 do x_2, y_2 pro vykreslení dat

```
void clrscr(word color);
```

Parametr: color: barva výplně

Použití: celý displej vyplní barvou color

Funkce pro vykreslení základních prvků

```
void setPixel(word color);
```

Parametr: color: barva bodu

Použití: vykreslí bod o barvě color, využívá se při zobrazení textu

```
void drawPixel(word x, word y, word color);
```

Parametry: x, y: souřadnice

color: barva bodu

Použití: vykreslí bod na zadaných souřadnicích x, y a barvě color

```
void drawHLine(word ax1, word ax2, word ay1, word color, byte th);
```

Parametry: ax_1, ay_1: počáteční souřadnice x_1, y_1

ax_2: cílová souřadnice x_2

color: barva linky

th: tloušťka linky

Použití: vykreslí horizontální linku od x_1, y_1 do vzdálenosti x_2, barvě color a tloušce th

```
void drawVLine(word ay1, word ay2, word ax1, word color, byte th);
```

Použití: vykreslí vertikální linku od y_1, x_1 do vzdálenosti y_2, barvě color a tloušce th

Funkce pro vykreslení rovnoběžníků

```
void drawRect(word ax1, word ay1, word ax2, word ay2, word color);
```

Parametry: ax_1, ay_1: počáteční souřadnice x_1, y_1

ax_2,ay_2: cílové souřadnice x_2, y_2

color: barva čtverce

Použití: vykreslí rovnoběžník o souřadnicích x_1, y_1 a x_2, y_2 v barvě color

```
void drawRectLine(word ax1, word ay1, word ax2, word ay2, word color,
byte th);
```

Parametry: color: barva rámečku

th: tloušťka orámování

Použití: vykreslí rámeček o zadané tloušťce v bodech

```
void drawRectFill(word ax1, word ay1, word ax2, word ay2, word
bcolor, word lcolor, byte th);
```

Parametry: bcolor: barva výplně

lcolor: barva orámování

Použití: vykreslí vyplněný rámeček o zadané tloušťce v bodech

```
void drawRectText(char *string, word ax1, word ay1, word ax2, word
ay2, word bcolor, word lcolor, byte orient);
```

Parametry: *string: ukazatel na řetězec znaků

orient: orientace textu, 1 – na výšku (portrait), 0 – na šířku (landscape)

Použití: vykreslí vyplněný rámeček s textem

Funkce pro vykreslení textu

```
void printChar(char ch, word x, word y, word chcolor, word bgcolor,
byte orient);
```

Parametry: ch: znak

x, y: souřadnice odkud se vykreslí znak

chcolor: barva znaku

bgcolor: barva podkladu

orient: orientace znaku, 1 – na výšku (portrait), 0 – na šířku (landscape)

Použití: vykreslí znak od zadaných souřadnic a orientaci

```
void printText(char *string, word x, word y, word chcolor, word bgcolor, byte orient);
```

Parametry: *string: ukazatel na řetězec znaků

x, y: souřadnice odkud se řetězec znaků vykreslí

chcolor: barva znaku

bgcolor: barva podkladu

orient: orientace znaku, 1 – na výšku (portrait), 0 – na šířku (landscape)

Použití: vykreslí řetězec znaků od zadaných souřadnic a orientaci

Funkce pro vykreslení obrázku

```
void printPix (word x, word y, word xr, word yr, long *pictures, byte orient);
```

Parametry: x, y: souřadnice odkud se obrázek vykreslí

x_r, y_r: rozlišení obrázku

*pictures: ukazatel na obrázková data

orient: orientace znaku, 1 – na výšku (portrait), 0 – na šířku (landscape)

Použití: vykreslí obrázek od zadané souřadnice

Funkce pro dotykový displej

```
void t_setup(unsigned char t_cs);
```

Parametry: t_cs: označuje na který chip select u Raspberry Pi je dotykový dispej připojen

Použití: inicializace dotykového displeje

```
int t_touch(void);
```

Parametr: vrací parametry jestli byl stisknut dotykový displej, 1 – ano, 0 – ne

Použití: funkce testuje, jestli byl stisknut dotykový dipslej

```
void t_read (unsigned char precission);
```

Parametr: precission: počet opakování určení souřadnic dotyku

TS_X, TS_Y: souřadnice místa dotyku x, y

Použití: funkce čte pozici dotyku displeje a vrací hodnoty souřadnic TS_X, TS_Y

3.6 Příklady řešení funkcí

V této kapitole si ukážeme softwarové řešení některých funkcí.

3.6.1 Funkce pro nastavení logických úrovní

Základní funkce pro nastavení logické úrovně na zadaném pinu

```
void spin(char pin) {        // funkce nastavi pin na log.1 (set pin)
        digitalWrite(pin, 1);   // nastavi na pin log. 1
}
```

3.6.2 Funkce pro zápis dat do řadiče LCD displeje

Funkce digitalWrite() je z knihovny wiringPi a zapisuje logickou hodnotu na zadaný pin. V našem případě se jedná o nastavení pinu na logickou úroveň 1. Funkce patří mezi základní funkce, které přímo ovládají řídící signály pro komunikaci s displejem.

Funkce pro zápis dat do řadiče LCD displeje

```
void LCD_WRITE_DATA (word data) {
      spin(PIN_DC);           // nastavi DC = 1 => data
      cpin(PIN_CS);           // aktivuje CS SSD1289
      wiringPiI2CWriteReg16(mcp1, OLATA, data); // zapis dat na
                                                // vystupy expanderu
      puls_high(PIN_WR);      // nabeznou hranou zapis do radice LCD
      spin(PIN_CS);           // deaktivace CS
}
```

Zápis dat provedeme nastavením pinu DC (PIN_DC) do logické hodnoty 0, čímž říkáme řadiči, že další informace na datových pinech bude příkaz. Nastavením pinu CS (PIN_CS) aktivujeme řadič LCD. Příkazem wiringPiI2CWriteReg16(mcp1, OLATA, com) zapíšeme na výstupy expandéru 1 adresu příkazu pro řadič. Následuje načtení těchto dat do řadiče. Zápis dat/příkazu do řadiče probíhá při CS = 0 s náběžnou hranou signálu WR (PIN_WR). Tím jsme do řadiče zaslali příkaz, po kterém následuje zaslání dat. Nakonec deaktivujeme CS (nastavíme hodnotu na logickou úroveň 1).

3.6.3 Funkce pro nastavení vykreslovacího okna

Funkcí `setxy()` nastavíme rozsah vykreslovacího okna v paměti LCD řadiče. Na adresu řadiče 0x44 zapisujeme 16bitovou hodnotu souřadnic x. Horních 8 bitů označuje

cílovou souřadnici x a dolních 8 bitů označuje počáteční souřadnici x. Na adresu 0x45 posíláme počáteční adresu y souřadnice a na adresu 0x46 cílovou y souřadnici.

```
void setxy(word ax1, word ay1, word ax2, word ay2)
   LCD_WRITE_COM_DATA(0x44,(ax2<<8) + ax1);
   LCD_WRITE_COM_DATA(0x45,ay1); // pocatecni y-ova souradnice
   LCD_WRITE_COM_DATA(0x46,ay2); // cilova y-ova souradnice
   LCD_WRITE_COM_DATA(0x4e,ax1); // x - odkud se zacne vykreslovat
   LCD_WRITE_COM_DATA(0x4f,ay1); // y - odkud se zacne vykreslovat
   LCD_WRITE_COM(0x22);          // následuje zapis obrazových dat
   }
```

3.6.4 Funkce pro vykreslení bodu

Následující funkcí `drawPixel()` vykreslíme bod na zadaných souřadnicích x, y a barvě color. Funkcí setxy nastavíme požadované souřadnice pro vykreslení bodu a následně je funkcí `LCD_WRITE_DATA(color);` zapíšeme do LCD řadiče, čímž dojde k vykreslení bodu.

```
void drawPixel(word x, word y, word color) {
      setxy(x, y, x, y);      // nastaveni souradnic bodu
      LCD_WRITE_DATA(color);  // zapis bodu do radice LCD
```

3.6.5 Funkce pro vykreslení rovnoběžníku

Funkcí drawRect() vykreslíme rovnoběžník o souřadnicích x_1, y_1 a x_2, y_2 a barvě zadané parametrem color. V tomto příkladu použijeme rychlejší možnost vykreslení bodů na LCD displeji. Nejprve nastavíme souřadnice pro vykreslení. Příkazem `LCD_WRITE_DATA()` vykreslíme jeden pixel o požadované barvě color. Následně pro komunikaci vybereme řadič LCD nastavením pinu CS na úroveň logické 0. Teď máme v paměti řadiče nastavené souřadnice pro vykreslení rovnoběžníku a na datových pinech LCD displeje (D0 – D16) máme nastavenou hodnotu barvy. Teď nám stačí pouze signálem Write vytvářet náběžné hrany, čímž dojde k uložení dat z datových pinů do řadiče displeje a k jejich vykresleni. Cyklusem `for (i=(ax2-ax1+1)*(ay2-ay1+1)` nastavujeme celkový počet bodů rovnoběžníku. Ke každé výsledné souřadnici nesmíme zapomenout připočítat hodnotu 1, protože souřadnice začínáme počítat od hodnoty 0.

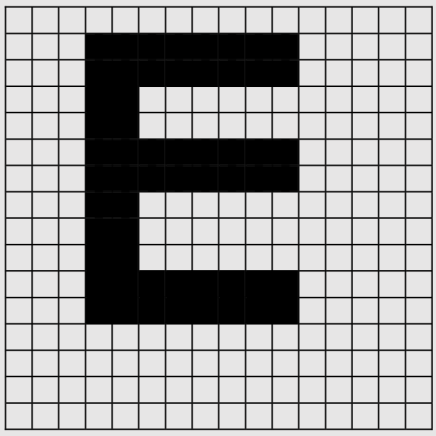
```
void drawRect(word ax1, word ay1, word ax2, word ay2, word color) {
      long i;                   // pocitadlo vykreslovanych bodu

      setxy(ax1, ay1, ax2, ay2); // nastaveni souradnic pro vykresleni
      LCD_WRITE_DATA(color);    // vykresleni bodu v barve color
      cpin(PIN_CS);             // vyber radice LCD

      for (i=(ax2-ax1+1)*(ay2-ay1+1); i > 1; i--) {
      // pripocitat 1 k souradnici x a y (0. bod)
            puls_high(PIN_WR);      // nabeznou hranou zapis do radice
      }
      spin(PIN_CS);           // deaktivace CS
}
```

3.6.6 Funkce pro vykreslení znaku

Pro vykreslování znaku musíme mít nejprve připravené pole znaků, které bude možné vykreslovat. Jedná se o znaky z ASCII tabulky od kódu 32, která je pro znak mezery až po kód 126, který je pro znak ~. V rozsahu kódů 30 – 39 máme číslice, v rozsahu 65 – 90 velká písmena a v rozsahu 97 – 122 malá písmena.

Pole znaků bude obsahovat rozpixelované jednotlivé znaky. Šířka znaku v bodech musí být dělitelná 8, aby nedocházelo k chybnému vykreslení. Jeden znak např. E vypadá v našem poli následovně:

```
0x00, 0x00, 0x1F, 0xE0, 0x1F, 0xE0, 0x18, 0x00, 0x18, 0x00,
0x1F, 0xE0, 0x1F, 0xE0, 0x18, 0x00, 0x18, 0x00, 0x18, 0x00,
0x1F, 0xE0, 0x1F, 0xE0, 0x00, 0x00, 0x00, 0x00,
```



Použitý font má matici 16x16 bodů a vykreslení odpovídá hodnotám v poli znaků. Jeden řádek znaku odpovídá dvojici hexadecimálních čísel (16 bitů). Na místě kde je jednička se nám vyznačí bod, kde je 0, nastaví se barva pozadí. Tabulku znaků vytvoříme

například pomocí programu The Dot Factory [11]. V programu si můžeme vygenerovat font pro LCD displej z existujícího systémového fontu. Musíme akorát udržovat šířku fontu dělitelnou 8. To je podmínka pro správné vykreslení znaků.

Znak vykreslíme funkcí printChar(). Nejprve otestujeme, jestli je znak vykreslován v režimu portrét. Pokud ano, následuje podmínka, kterou zjišťujeme, jestli dle zadaných souřadnic a velikosti lze znak na displej vykreslit. Pokud ne, funkci ukončíme příkazem return. Pokud se znak na displej vejde, nastavíme způsob vykreslení a pokračujeme ve vykreslení.

```
void printChar(char ch, word x, word y, word chcolor, word bgcolor,
byte orient)
{
   word  i, j, temp; // pacitadla i, j, ukazatel na pocatek znaku
   if (orient==PORTRAIT) { // pokud je nastaven portrait
     if (x+mFONT.f_x_size-1 <= x_max && y+mFONT.f_y_size-1 <=
y_max){
       setxy(x, y, x + mFONT.f_x_size-1, y+mFONT.f_y_size-1);
       LCD_WRITE_COM_DATA(0x11,0x6070); způsob vykreslovani
       LCD_WRITE_COM(0x22);  // priprava pro zadavani bodu
                } else {
                        return;
                }
     } else {
   if (x+mFONT.f_y_size-1 <= x_max && y+mFONT.f_x_size-1 <= y_max) {
     setxy(x, y, x+mFONT.f_y_size-1, y+mFONT.f_x_size-1);
     LCD_WRITE_COM_DATA(0x11,0x6068);
     LCD_WRITE_COM(0x22);
     } else {
        return;
     }
}
```

Proměnnou temp použijeme na ukazatel v poli fontu. Od ASCII kódu znaku odečteme offset prvního znaku, který máme v našem poli. Získáme hodnotu kolikátý znak v poli budeme vykreslovat. Vypočteme kolik bajtů jednotlivý znak v poli fontů zabere. Vypočteme `((mFONT.f_x_size/8)*mFONT.f_y_size)`. Šířku fontu x_size vydělíme 8, což odpovídá 1B. Vynásobením této hodnoty šířkou fontu y_size získáme počet bajtů, které jednotlivý znak v poli fontů zabere. Vynásobením hodnoty velikosti znaku s hodnotou kolikátý znak chceme vykreslit nám dá kolikátý B v poli fontu chceme vykreslit. Nesmíme zapomenout připočítat první 4B, které nám na začátku pole označují šířku a výšku fontu, posunutí v ASCII tabulce a celkový počet znaků v poli. Známe odkud

z pole máme začít načítat znaky, známe délku jednoho znaku, můžeme začít vykreslovat. Každý načtený bajt rotujeme a pokud je na nejvyšší pozici jednička, vykreslíme bod v barvě znaku. V opačném případě se jedná o pozadí a vykreslíme bod v barvě pozadí.

```
cpin(PIN_CS);   // aktivace CS
temp = ((ch-mFONT.f_offset)*((mFONT.f_x_size/8)*mFONT.f_y_size))+4;
for(j=0; j <((mFONT.f_x_size/8) * mFONT.f_y_size); j++) {
  ch = mFONT.tfont[temp];
  for(i=0;i<8;i++) { // vykresleni radku znaku
    if((ch&(1<<(7-i)))!=0)   {
      setPixel(chcolor); // barva znaku
    }
    else
    {
      setPixel(bgcolor); // barva pozadi
    }
  }
  temp++; // posun na dalsi radek znaku
  }
  spin(PIN_CS);   // deaktivace CS
}
```

3.6.7 Vykreslení obrázku

Displej pracuje v režimu RGB565. Obrázek, který chceme vykreslit musíme do tohoto režimu převést. Jedním z konvertorů je Image Converters, který je součástí UTFT nástrojů [12]. Image converter je pod licencí CC BY-NC-SA 3.0, lze tedy pro nekomerční účely použít. Převedený obrázek je pole hodnot. Poli nastavíme static void, dále vložíme příkazem include do zdrojového kódu a můžeme vykreslit funkcí printPix(). Zadáme počáteční souřadnice pro vykreslení, rozlišení pro osy x a y, název obrázku a orientaci. Nejprve provedeme otestování, jestli lze dle zadaných parametrů obrázek vykreslit. Pokud ano, procházíme postupně pole hodnot a jednotlivé body obrázku vykreslujeme.

```
void printPix (word x, word y, word xr, word yr, long *pictures,
byte orient) {
 long pix_c = 0;  // pocitadlo bodu obrazku pix color
 if ((orient==PORTRAIT) && ((x + xr <= x_max) && (y + yr  <= y_max))){
   setxy(x, y, x+xr-1, y+yr-1); // souradnice pro vykresleni
   LCD_WRITE_COM_DATA(0x11,0x6038); // zpusob vykresleni
   LCD_WRITE_COM(0x22); // cekani na data bodu
   cpin(PIN_CS);
   for (pix_c = 0; pix_c < (xr*yr); pix_c++) {
       LCD_WRITE_DATA(pictures[pix_c]);
   }
   spin(PIN_CS);
   clrxy();
```

3.6.8 Funkce pro získání souřadnic z dotykového displeje

Řadič dotykového displeje vrací 12bitové hodnoty, které musíme převést na souřadnice x a y. Aby bylo možné získat rozsahy hodnot pro osy, musíme vyzkoušet dotyky ve všech rozích displeje hodnoty, které řadič vrací. Dle těchto parametrů jsme schopni převést hodnoty získané z řadiče dotykového displeje na hodnoty souřadnic x a y, které využijeme pro ovládání LCD displeje.

Souřadnice dotyku na dotykový displej získáme funkcí tread () s parametrem precission. Tento parametr nám říká, kolikrát proběhne načtení dat z dotykového displeje, než získáme konečné souřadnice. Připravíme si tří prvkové pole pro hodnoty x a y. Pro čtení dat z dotykového displeje použijeme funkci wiringPiSPIDataRW() z knihovny wiringPiSPI. Posíláme jeden bajt a chceme přijmout dva bajty. Přenos po sběrnici SPI běží duplexně, tedy když vysíláme, zároveň přijímáme. Potřebujeme tedy poslat tři bajty, z nichž první je příkaz pro řadič dotykového displeje a další dva jsou libovolné. Do těchto dvou bajtů získáme převedenou 12bitovou hodnotu z řadiče o poloze dotyku v dané ose. Máme tedy 16bitů dat a na MSB je bit, který je dán zpožděním převodníku, tedy ho nepotřebujeme. Následuje 12bitová hodnota. Rotací bitů získáme do proměnných x_poz a y_poz hodnoty o pozici dotyku. 12bitů odpovídá maximální hodnotě 4095, pokud hodnota v pomocné proměnné je v rozsahu <0, 4095> připočítáme ji do pomocné proměnné dle os x_temp nebo y_temp a zvýšíme počítadlo dobrých průchodů. Počtem precission jsme si nastavili, kolikrát proběhne zjištění polohy doteku. Po tomto počtu průchodů máme v pomocných proměnných nasčítané hodnoty poloh x a y, které dělíme počtem úspěšných průchodů. Tím získáme průměrnou hodnotu. Pokud proběhlo z nastavení precission úspěšně alespoň 4/5 průchodů, převedeme získané hodnoty na souřadnice x a y.

```
void t_read (unsigned char precission) {

for (i=0; i<precission; i++) {
  unsigned char datay[] = {0x90, 0x00,0x00};
  rc =  wiringPiSPIDataRW (1, datay, 3) ;
  unsigned char datax[] = {0xd0, 0x00,0x00};
  rc =  wiringPiSPIDataRW (1, datax, 3) ;
  x_poz = ((datax[1])<< 1)<< 4 | datax[2] >> 3;
  printf("x_poz = %04X = %03d\n", x_poz, x_poz);

  y_poz = ((datay[1])<< 1)<< 4 | datay[2] >> 3;
     // 12bit prevedena hodnota (busy bit ven, posun o 4 a | 5bit)

  if (x_poz > 0 && y_poz <4095) {
          x_temp += x_poz;
          y_temp += y_poz;
          count++;
  printf("xtemp: %d, ytemp %d\n", x_temp, y_temp);
  }
  else {
    printf("nedotek c %d\n", i);
  }
}
   if ((count > 0) && (count > precission*0.8)) {
   x_poz = x_temp/count;
   y_poz = y_temp/count;


   if (x_poz > 0 && y_poz <4095) {


   TS_X = ((x_poz - t_x_bot) * x_size)/(t_x_top - t_x_bot);
   TS_Y = ((t_y_left - y_poz) * y_size)/(t_y_left - t_y_right);
}}}
```

4 TESTOVÁNÍ

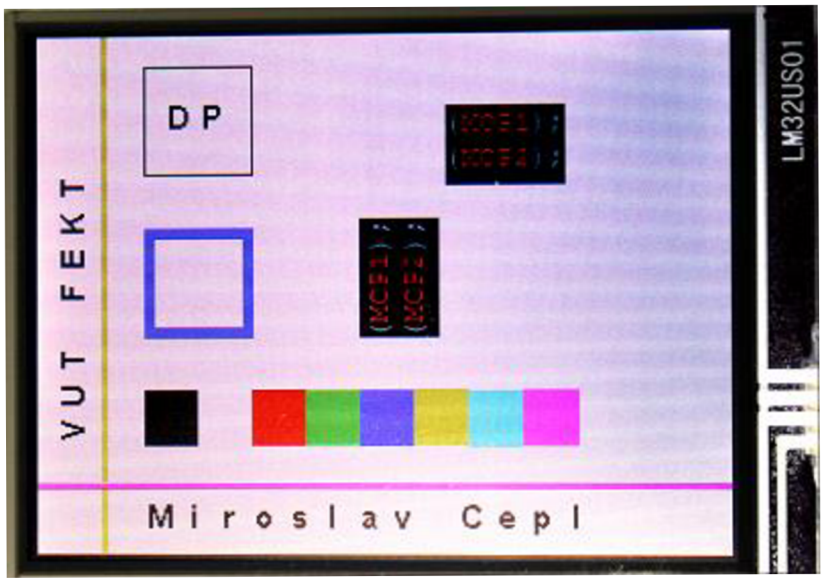
Po úspěšném zapojení, naprogramování a odladění můžeme přejít k testování.

Displej má 76800 bodů a jeho překreslení proběhne za 12s, při kmitočtu sběrnice I2C 750kHz. Testoval jsem i základní kmitočet 100kHz, při kterém překreslení celého displeje trvá přibližně 50s. Sběrnice i2c může dosahovat kmitočtu až 1,7 MHz. Stejnou frekvenci mají uvedenou i expandéry MCP23017 v datasheetu. Při nastavení frekvence 800kHz přestalo navržené zapojení pracovat. Sběrnice nebyla schopná správně rozeznat připojené expandéry MCP23017. Je to dáno tím, že expandéry pro frekvenci 400kHz mají být napájený 2,7 – 5,5V a pro frekvenci 1,7MHz 4,5 – 5,5V. Displej i GPIO piny Raspberry Pi může být napájen pouze 3,3V, při vyšším napětí by mohlo dojít k jejich zničení.

Pro odzkoušení dotykového displeje jsem připravil aplikaci, která sestává ze čtyřech tlačítek a jednoho stavového okna. Při stisku tlačítka „off“ dojde k červenému prokreslení stavového okna s nápisem off a ukončí se aplikace. Při stisku ostatních tlačítek dojde vždy k překreslení stavového okna a výpisu textu v barvách odpovídajících stisknutému tlačítku. Obdobnou aplikaci můžeme vytvořit pro řízení zařízení, kde potřebujeme nějakou odezvu. Lze zařízení nejen řídit, ale i nechat si vypisovat stavové informace.

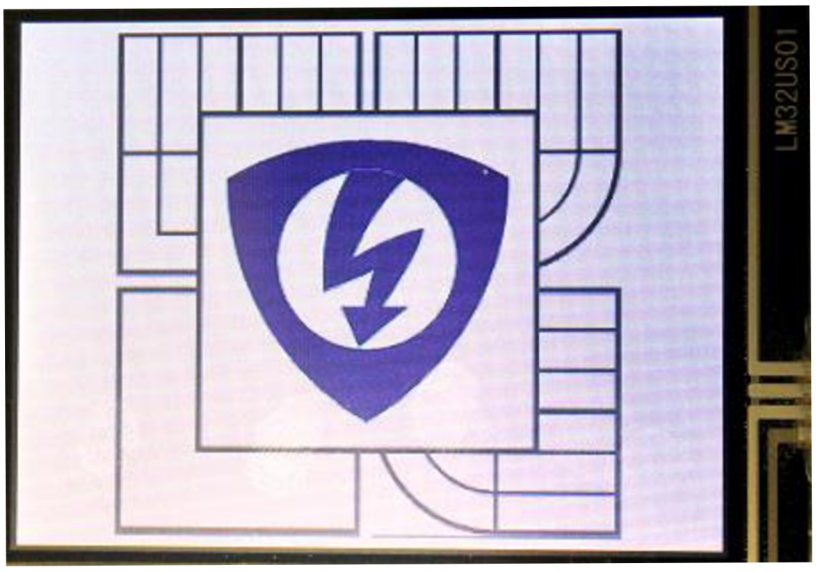
Na Obr. 4.1: Ukázka vykreslení 1Obr. 4.1 je ukázka aplikace vybraných funkcí.

```
clrscr(WHITE);
printPix(170, 190, 55, 37, pokus, 1);
printPix(100, 150, 55, 37, pokus, 0);
drawRectText("DP", 175, 50, 225, 100, WHITE, BLACK, 1);
drawRect(50,50, 75, 75, BLACK);
drawRect(50,75, 75, 100, WHITE);
drawRect(50,100, 75, 125, RED);
drawRect(50,125, 75, 150, LIME);
drawRect(50,150, 75, 175, BLUE);
drawRect(50,175, 75, 200, YELLOW);
drawRect(50,200, 75, 225, CYAN);
drawRect(50,225, 75, 250, MAGENTA);
```

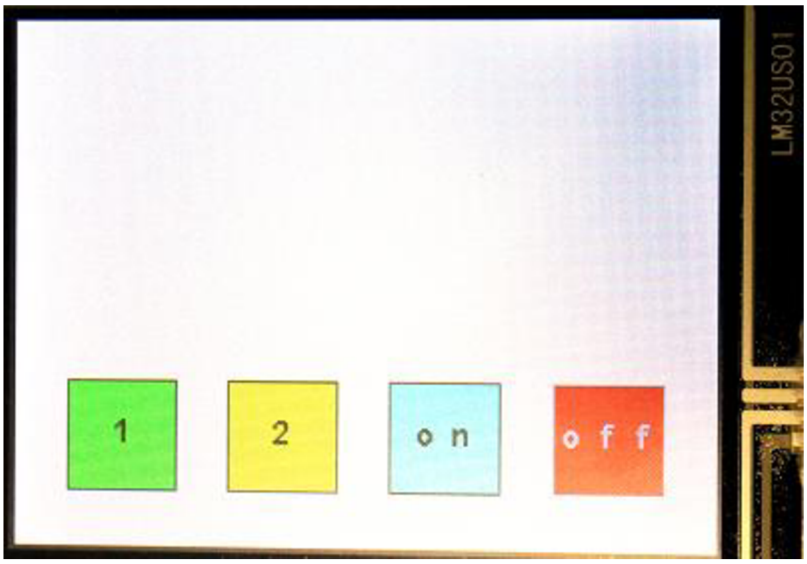


Obr. 4.1: Ukázka vykreslení 1

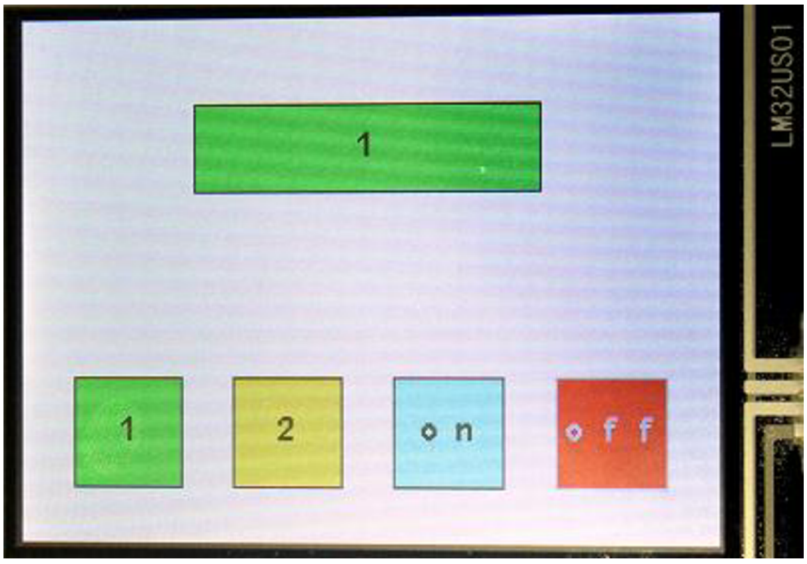
Na Obr. 4.2 je použitá funkce printPix().



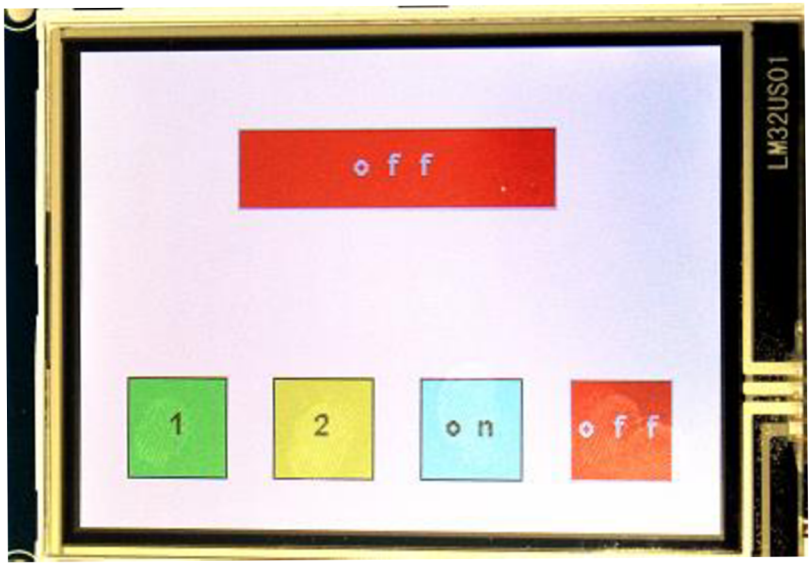
Obr. 4.2: Ukázka vykreslení 2



Obr. 4.3: Ukázka vykreslení 3



Obr. 4.4: Ukázka vykreslení 4



Obr. 4.5: Ukázka vykreslení 5

5 ZÁVĚR

Raspberry Pi s připojením expandérů MCP23017 rozšiřujících počet vstupně/výstupních pinů umožňuje rozšířit svoji flexibilitu při nasazení v náročných aplikacích. Po sériové sběrnici I2C lze komunikovat s použitými expandéry MCP23017 na frekvenci 100 kHz až 1,7 MHz. Při porovnání se verzí MCP23S17 pro sériovou sběrnici SPI, která může komunikovat až frekvencí 10 MHz, se jedná o velmi omezující frekvenci. V případě, že není třeba neustálé překreslování celého displeje, lze omezení dané malou frekvencí komunikace ovlivnit překreslením pouze parciálních částí na displeji. Tím dojde k omezení přenosu dat.

V současné době při řešení komunikace Rapsberry Pi s displejem se používá komunikace s rozšiřujícím kytem Arduiono, který je osazen mikropočítačem ATMega. Cena s využitím expandéru MCP23017 je zhruba 15krát menší.

Použití rozšiřujících obvodů MCP23017 k ovládání a vykreslování dotykového displeje TFT_320QVT po sériové sběrnici I2C ušetří jeden kanál rychlejší sériové sběrnice SPI u Raspberry Pi. Použitím displeje pro informační účely nebo možnost výběru volby dotykem na dotykovou část displeje lze tuto sběrnici plně využít.

Navržený ovladač splňuje zadání a lze ho plně využít pro ovládání dotykového displeje TFT_320QVT.

LITERATURA

[1] MONK, Simon. *Raspberry Pi cookbook*. xiv, 393 pages. ISBN 978-144-9365-226.

[2] GOODWIN, Steven. *Smart home automation with Linux*. New York: Apress, 2010, xvi, 289 p. ISBN 14-302-2778-8.

[3] RICHARDSON, Matt. *Getting started with BeagleBone*. New York: Apress, 2010, xiii, 126 pages. ISBN 14-493-4537-9.

[4] Pin Numbering - Raspberry Pi Model B+. [online]. [cit. 2014-10-16]. Dostupné na www: <http://pi4j.com/pins/model-b-plus.html>

[5] MICROCHIP. *Datasheet MCP23017* [online]. [cit. 2014-11-15]. Dostupné na www:<http://ww1.microchip.com/downloads/en/DeviceDoc/21952b.pdf>.

[6] SOLOMON SYSTECH. *Datasheet SSD1289* [online]. [cit. 2014-11-18]. Dostupné na www: < http://www.datasheet-pdf.com/datasheetdownload.php?id=787786>.

[7] ARDUINO8.WEBNODE.CZ. [online]. [cit. 2014-10-3]. Dostupné na www: <http://arduino8.webnode.cz/news/lekce-28-arduino-a-tft-dotykovy-displej/>.

[8] OZZMAKER.COM. [online]. [cit. 2014-10-4]. Dostupné na www: <http://ozzmaker.com/2013/05/23/raspberry-pi-with-a-3-2-tft-with-touch-control

[9] FORUM.ARDUINO.CC. *TFT_320QVT* [online]. [cit. 2014-10-12]. Dostupné na www: http://forum.arduino.cc/index.php?PHPSESSID=fb1cgpem3jo44is-aifgq9juaf3&action=dlattach;topic=183302.0;attach=104311

[10] WiringPi. HENDERSON, Gordon. *Raspberry Pi | Wiring | Gordons Projects* [online]. 2014 [cit. 2015-05-26]. Dostupné z: https://projects.drogon.net/raspberry-pi/wiringpi/

[11] DUCHAN, Eran. The Dot Factory: An LCD Font and Image Generator. ,. *The Dot Factory: An LCD Font and Image Generator* [online]. 2009 [cit. 2015-05-26]. Dostupné z: http://www.eran.io/the-dot-factory-an-lcd-font-and-image-generator/

[12] LOOKING FOR A COST-EFFECTIVE TFT HMI? In: *Rinky-Dink Electronics* [online]. 2015 [cit. 2015-05-26]. Dostupné z: http://www.rinkydinkelectronics.com/

A PŘÍLOHA

Obsahem přiloženého CD je elektronická verze diplomové práce a použité zdrojové texty s ukázkami.