VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

PROPOJENÍ PROCESORU AURIX S PLATFORMOU NVIDIA JETSON
INTERCONNECTION OF AURIX MICROCONTROLLER WITH NVIDIA JETSON PLATFORM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Michal Smrčka**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Petr Blaha, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Michal Smrčka ***ID:*** 203342
Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Propojení procesoru AURIX s platformou NVIDIA Jetson

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s periferiemi pro sériovou komunikaci procesoru AURIX a platformy NVIDIA Jetson.
2. Vyberte vhodné sériové rozhraní pro vzájemný přenos dat a implementujte ho na obou platformách. Hlediskem pro výběr je rychlost komunikace a obousměrný přenos dat.
3. Spojení otestujte na příkladu, kdy procesor AURIX řídí elektrický motor (případně jeho model) a platforma Jetson si vyčítá měřená data (pevně daný buffer) pro další zpracování.

DOPORUČENÁ LITERATURA:

Frenzel, L. E.: Handbook of Serial Communications Interfaces. A Comprehensive Compendium of Serial Digital Input/Output (I/O) Standards. Elsevier Inc., 2016. ISBN 978-0-12-800629-0.

uživatelské manuály procesoru AURIX2G a platformy NVIDIA Jetson

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 8.6.2020

Vedoucí práce: doc. Ing. Petr Blaha, Ph.D.

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

UPOZORNĚNÍ:
Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

Abstrakt

Část této práce pojednává o základech používání platforem NVIDIA Jetson Nano a AURIX Application Kit, konkrétně typů TC277 TFT a TC224 TFT. Rovněž jsou v ní teoreticky rozebrány některé periferie platforem. Tyto základy jsou nezbytné pro navazující část, která je zaměřena na realizaci sériového komunikačního rozhraní pro přenos dat mezi platformami. V práci je popsána programová implementace 2 rozhraní: SPI a Ethernet MAC. Komunikace přes SPI (respektive QSPI v případě AURIX) je doplněná o využití GPIO pinů, v případě Ethernetu je komunikace realizována na úrovni linkové vrstvy pomocí Ethernetových rámců. Propojení skrze SPI je dále otestováno v konkrétní aplikaci při řízení BLDC motoru pomocí TC224 a AURIX eMotor Drive Kit V2.1, kdy jsou měřená data odesílána na Jetson Nano k real-time zpracování.

Klíčová slova

NVIDIA Jetson Nano, AURIX TC277, AURIX TC224, SPI, QSPI, GPIO, Ethernet

Abstract

Part of this thesis covers the basics of working with NVIDIA Jetson Nano and AURIX Application Kit platforms, namely TC277 TFT and TC224 TFT. It also theoretically analyzes some peripherals of the platforms. These basics are necessary for the following part, which is focused on the implementation of a serial communication interface for data transfer between platforms. The work describes a program implementation of 2 interfaces: SPI and Ethernet MAC. The communication via SPI (or QSPI in the case of AURIX) is supplemented with the use of GPIO pins, in the case of Ethernet the communication is realized at a link layer using Ethernet frames. The connection via SPI is further tested in a specific application when controlling a BLDC motor using TC224 and AURIX eMotor Drive Kit V2.1, where the measured data is sent to the Jetson Nano for real-time processing.

Keywords

NVIDIA Jetson Nano, AURIX TC277, AURIX TC224, SPI, QSPI, GPIO, Ethernet

Bibliografická citace:

SMRČKA, Michal. Propojení procesoru AURIX s platformou NVIDIA Jetson. Brno, 2020. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/126992. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Petr Blaha.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma Propojení procesoru AURIX s platformou NVIDIA Jetson jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **7. června 2020** ……

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce doc. Ing. Petru Blahovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: **7. června 2020**

……
podpis autora

Obsah

Seznam symbolů a zkratek

Zkratky:

AI	…	Artificial Intelligence
API	…	Application Programming Interface
ASCLIN	…	Asynchronous/Synchronous Interface
BLDC	…	Brushless Direct Current
CMOS	…	Complementary Metal–Oxide–Semiconductor
CPU	…	Central Processing Unit
CRC	…	Cyclic Redundancy Check
DHCP	…	Dynamic Host Configuration Protocol
DMA	…	Direct Memory Access
DP	…	DisplayPort
FCS	…	Frame Check Sequence
FOC	…	Field Oriented Control
GPIO	…	General Purpose Input/Output
GPU	…	Graphics Processing Unit
HDMI	…	High-Definition Multimedia Interface
HSCT	…	High Speed Communication Tunnel
HSSL	…	High Speed Serial Link
I/O pin	…	Input/Output pin
I^2C	…	Inter-Integrated Circuit
IDE	…	Integrated Development Environment
IEEE	…	Institute of Electrical and Electronics Engineers
iLLD	…	Infineon Low Level Drivers
L4T	…	Linux for Tegra
LSB	…	Least Significant Bit
LVDS	…	Low-Voltage Differential Signaling
MAC	…	Media Access Controller
MII	…	Media Independent Interface
MPIO	…	Multi-Purpose digital Input /Output
MSB	…	Most Significant Bit
OSI	…	Open Systems Interconnection
PC	…	Personal Computer
PDU	…	Protocol data unit
PWM	…	Pulse Width Modulation
QSPI	…	Queued Serial Peripheral Interface
RAM	…	Random-Access Memory
RMII	…	Reduced Media Independent Interface
RX FIFO	…	Receive First In, First Out

SDK	…	Software Development Kit
SFIO	…	Single Function Input /Output
SoC	…	System on a Chip
SO-DIMM	…	Small Outline Dual In-line Memory Module
SPI	…	Serial Peripheral Interface
TCP/IP	…	Transmission Control Protocol/Internet Protocol
TFT	…	Thin-Film Transistors
TX FIFO	…	Transmit First In, First Out
UDE	…	Universal Debug Engine
USB	…	Universal Serial Bus

Seznam obrázků

Seznam tabulek

1 ÚVOD

Nedílnou součást průmyslové automatizace a měření tvoří číslicové řízení a zpracování dat. Pro uživatele začínající v těchto oblastech se na trhu vyskytuje velké množství různých cenově dostupných vývojářských sad. Mezi takové patří i NVIDIA® Jetson Nano™ Developer Kit a AURIX™ Application Kit, které byly použity pro tuto práci.

Přestože spektrum využití obou platforem je velmi široké, uplatňuje se každá z nich lépe v rozdílných oblastech. V případě procesorů AURIX se jedná zejména o oblasti číslicového řízení, NVIDIA Jetson slouží naopak přednostně pro číslicové zpracování dat pomocí algoritmů s neuronovými sítěmi. Jednou z možností, jak spojit aplikace z obou těchto oblastí do jednoho funkčního celku, pracujícího v reálném čase, je umožnit vzájemný přenos dat mezi platformami.

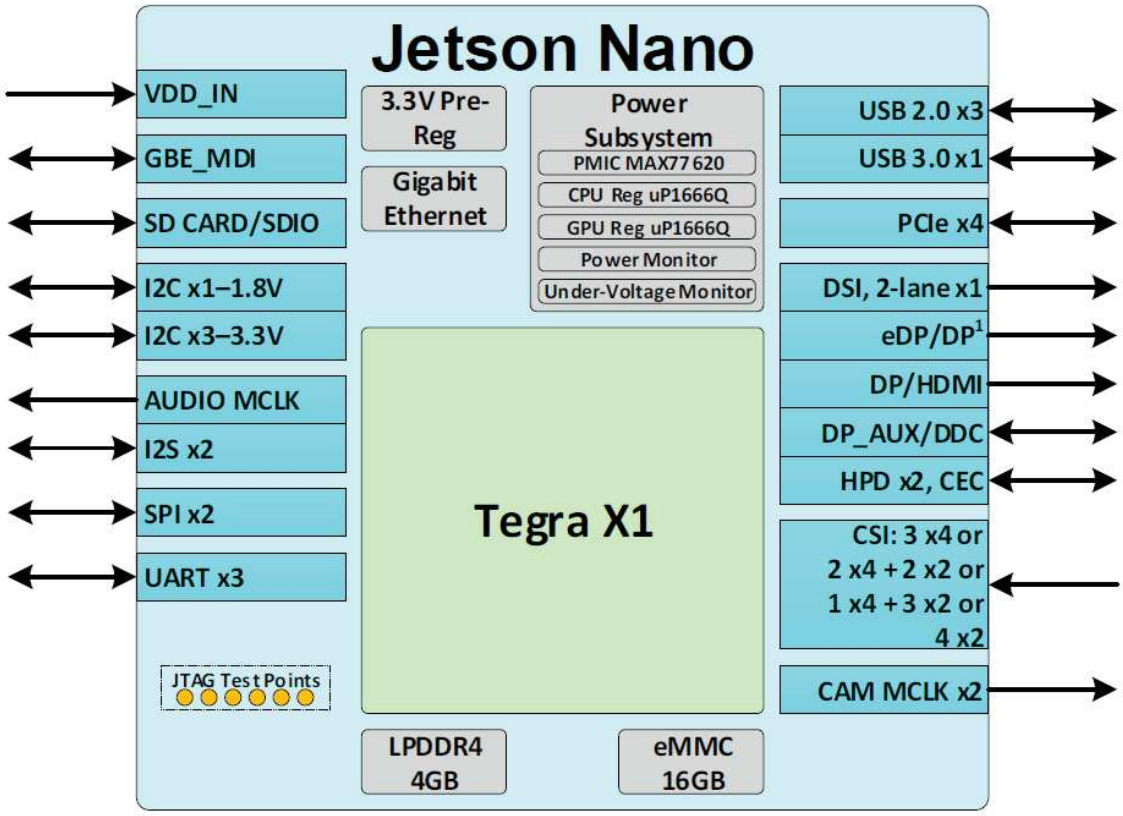
Cílem této bakalářské práce bylo tedy vytvořit komunikační rozhraní mezi mikrokontrolérem AURIX a jednodeskovým počítačem NVIDIA Jetson. První část práce se bude zabývat charakteristikou propojovaných platforem a jejich vybranými periferiemi pro sériovou komunikaci. Z těchto periferií bude následně vybrána nejvhodnější s ohledem na rychlost komunikace a obousměrný přenos dat. Způsob jejich implementace na platformách bude popsán v druhé části. Ta bude obsahovat rovněž popis zprovoznění obou platforem a jejich vývojového prostředí. V závěrečné části bude otestována komunikace ve specifické aplikaci, kdy mikrokontrolér AURIX řídí elektrický motor a data naměřená při tomto řízení posílá k následnému zpracování do platformy NVIDIA Jetson. Tím bude ověřena spolehlivost vytvořeného komunikačního rozhraní.

2 CHARAKTERISTIKA PROPOJOVANÝCH PLATFOREM

2.1 NVIDIA® Jetson Nano™ Developer Kit

Tento vývojářský kit přináší výpočetní výkon a funkce pro aplikaci moderní umělé inteligence (AI) pomocí low-cost, jednoduše použitelné platformy s nízkou spotřebou. Kit je podporován všestrannou sadou vývojářských nástrojů NVIDIA® JetPack™ SDK, která zahrnuje L4T (Linux for Tegra, což je linuxová distribuce Ubuntu Desktop s ovladači NVIDIA), knihovny a rozhraní pro programování aplikací (APIs) využívajících AI a počítačové vidění, vývojářské nástroje, dokumentaci a ukázkové kódy. [1]

NVIDIA® JetPack™ SDK bývá nejčastěji uložen na microSD kartě (minimálně microSD 16 GB UHS-1, která však není standardní součástí balení), která je vložena do microSD slotu na Jetson Nano modulu. [1]

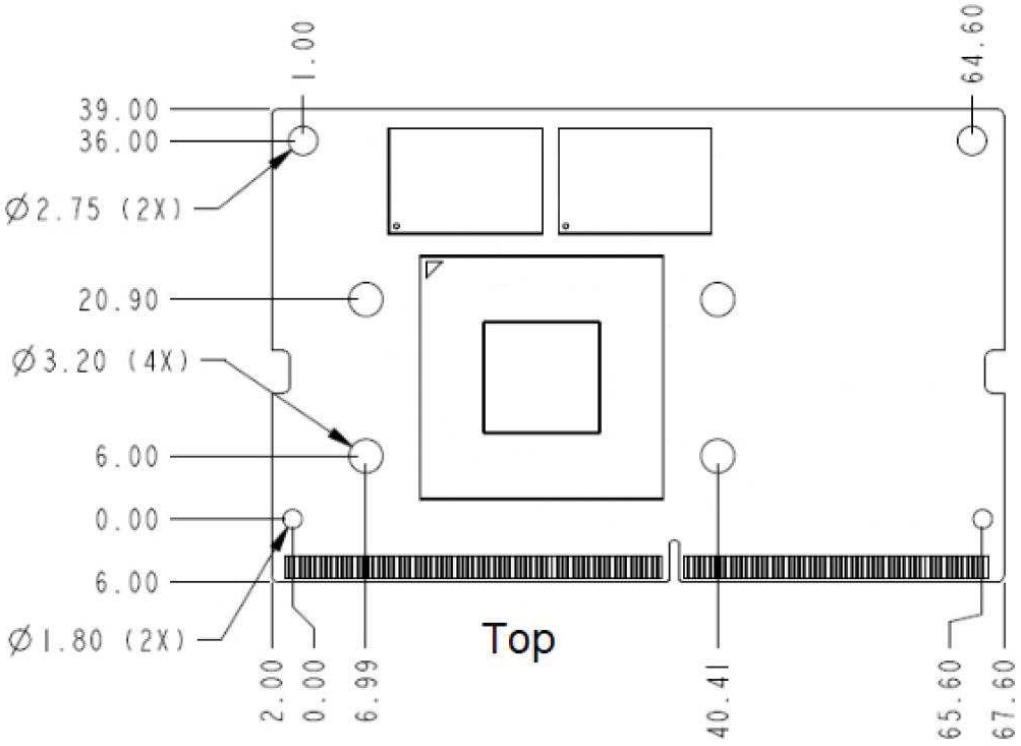


Obr. 2.1 Blokové schéma Jetson Nano modulu [2]

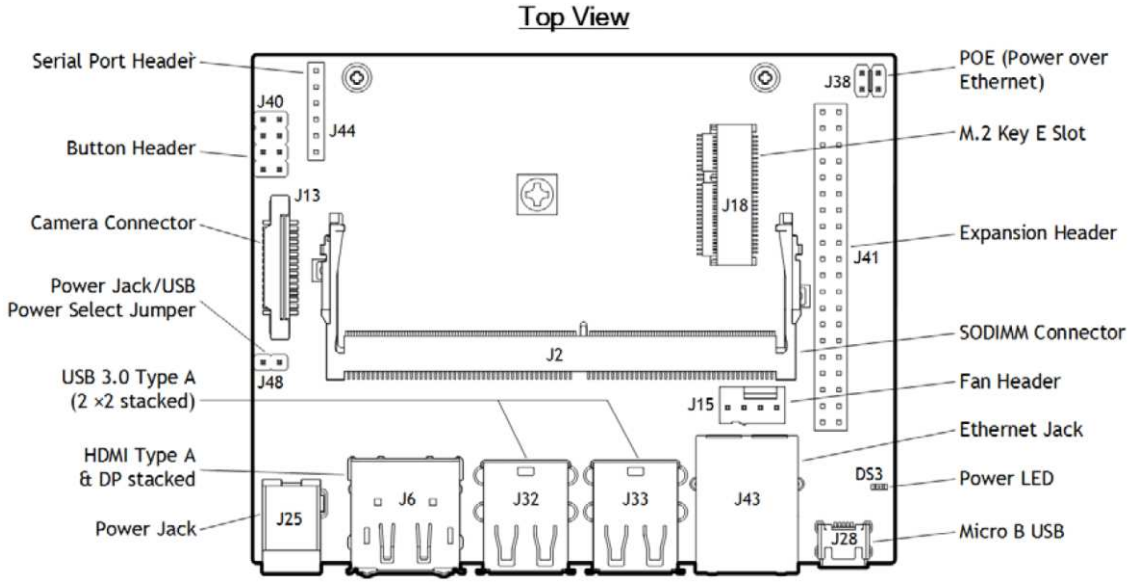
Hlavní součástí kitu je Jetson Nano modul, jehož blokové schéma je znátorněno na (Obr. 2.1) a nákres na (Obr. 2.2). Nejdůležitější částí modulu je

Tegra X1 SoC, který v sobě integruje procesor ARM® Cortex® -A57 MPCore (Quad-Core) a 128-jádrovou GPU Maxwell. [3]

Kompletní kit pak sestává z Jetson Nano modulu opatřeného pasivním chladičem a vloženého do SO-DIMM slotu na nosné desce (carrier board, někdy též base board), která je znázorněna na (Obr. 2.3). Ta obsahuje velké množství konektorů pro připojení periferních zařízení. [1]



Obr. 2.2 Nákres Jetson Nano modulu s rozměry [3]

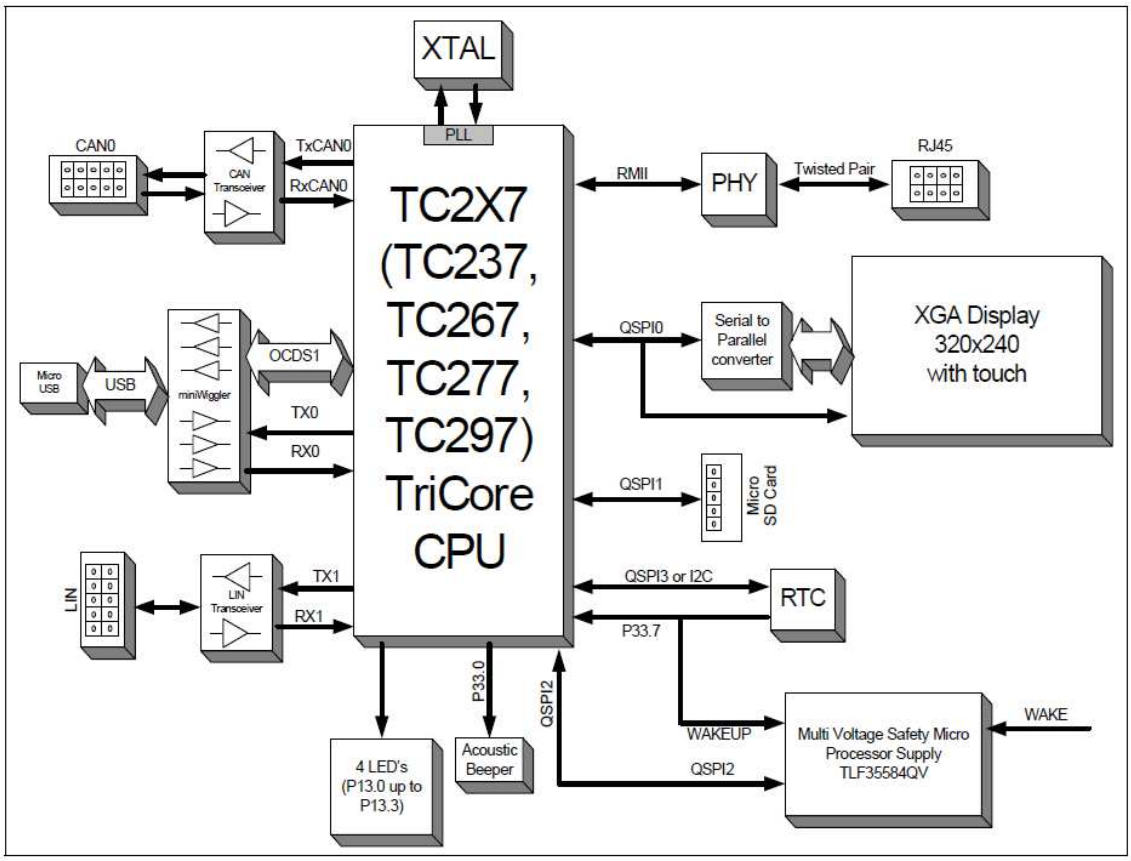


Obr. 2.3 Nosná deska Jetson Nano modulu použitá v kitu [1]

2.2 AURIX™ Application Kit TC277 TFT

AURIX™ TriCore™ spojuje do jediné platformy funkčnost procesoru s architekturou RISC (Reduced Instruction Set Computer), mikrokontroléru a DSP (Digital Signal Processor). Tím se stává ideální pro širokou škálu aplikací, převážně v automobilovém průmyslu. Patří sem například řízení spalovacích motorů, systémů posilovače řízení nebo pokročilých asistenčních systémů pro autonomní vozidla. Především vyniká v optimalizovaných aplikacích řízení elektrických motorů a zpracování signálů. [4]

Tento kit představuje univerzální nástroj, který poskytuje rychlý přístup ke schopnostem výkonné architektury TriCore. Jedná se o low-cost, flexibilní platformu, která nabízí široké spektrum využití díky integraci tříjádrového 32-bitového skalárního procesoru TriCore™ TC277D (pozice U101 na Obr. 2.5). Blokové schéma platformy znázorňuje (Obr. 2.4). Jak je vidět, kit je vybaven celou řadou periferií pro interakci s dalšími zařízeními. [5]



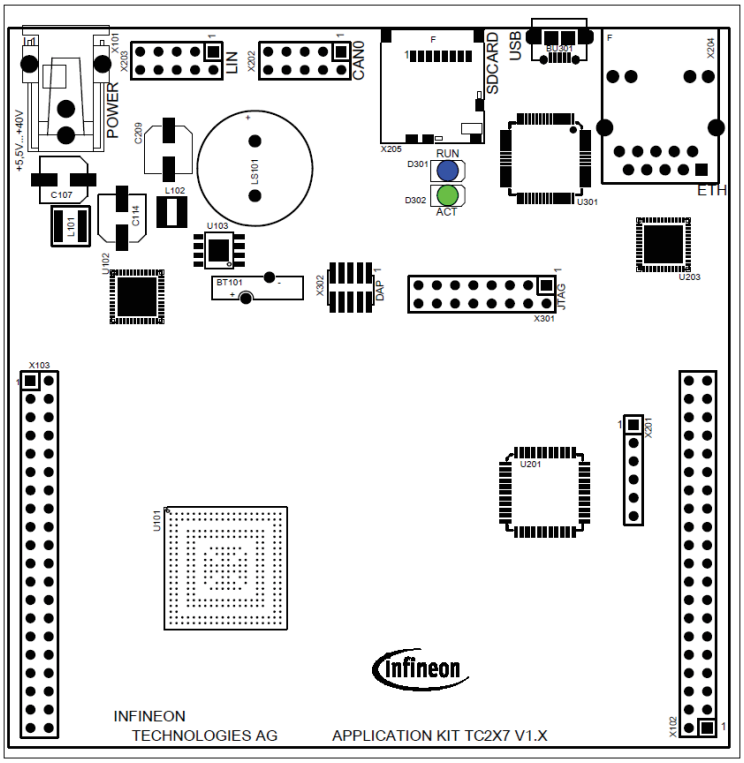
Obr. 2.4 Blokové schéma Application Kit TC2X7 [5]

Vývoj C/C++ aplikací probíhá v PC software HighTec Free TriCore™ Entry Tool Chain. Aplikace vytvořené v tomto prostředí se po zkompilování z PC do platformy nahrávají pomocí Micro USB portu (označen USB na Obr. 2.5).

K následnému testování aplikací je poskytován výkonný debugovací software od Universal Debug Engine UDE and Microcontroller Debugger for AURIX, TriCore, Power Architecture. [5]

2.3 AURIX™ Application Kit TC224 TFT

Od Application Kit TC277 TFT se liší především tím, že integruje méně výkonný jednojádrový procesor TC224A. Dále postrádá některé periferie (např. Ethernet nebo I²C) a odlišuje se také rozložením I/O pinů. Práce s ním je však prakticky stejná jako s TC277, softwarové rozhraní je totožné a tudíž většinou nebývá složité upravit zdrojové kódy aplikací pro jiný typ desky. TC224 byl využíván především z důvodu jednoduššího testování SPI rozhraní, neboť má na pinech pro SPI napětí 3,3 V (TC277 má 5 V).



Obr. 2.5 Nákres rozložení vrchní strany desky Application Kit TC2X7 [5]

3 PERIFERIE PRO SÉRIOVOU KOMUNIKACI

Digitální zařízení spolu mohou komunikovat pomocí sběrnic, které se v základu dělí na paralelní a sériové. V případě paralelní sběrnice se jedná o paralelní komunikaci, kdy se naráz přenáší větší počet bitů, typicky 8, 16, 32 atd. Pro každý bit přenášený v jednom okamžiku je zapotřebí samostatný vodič. Paralelní komunikace je rychlá a používá se především v rámci jednoho plošného spoje, na kterém není problém vytvořit větší množství vodivých cest. Pro komunikaci na větší vzdálenost se však ze zřejmých důvodů využívá sběrnice sériová, která vystačí s jediným vodičem pro data, přenášená po jednotlivých bitech. [6]

Pro splnění zadání této práce by se s ohledem na rychlost komunikace nabízelo i použití sběrnice paralelní, ovšem počet I/O pinů obou platforem je omezený. Je pravděpodobné, že pro další využití platforem v konkrétních aplikacích bude potřeba připojit k nim další periferní zařízení a pro paralelní komunikaci by počet pinů nemusel stačit. Dále se tedy práce zabývá pouze komunikací sériovou.

3.1 General Purpose Input/Output (GPIO)

Nejzákladnější činností mikrokontroléru je nastavování a čtení logických hodnot na jeho vstupech/výstupech. Právě k tomuto účelu slouží GPIO piny. V případě procesorů AURIX se jednotlivé I/O piny pro lepší přehlednost sdružují do Portů, piny se pak značí např. P02.1 (Port 2, pin 1). [7]

GPIO piny se dají nastavit buďto jako výstupní, vstupní nebo jako zdroj přerušení (Interrupt source) od úrovně nebo hrany signálu. Teoreticky se tedy s jejich pomocí dá realizovat celé komunikační rozhraní mezi dvěma mikrokontroléry, ovšem obslužný program by poté byl poměrně složitý.

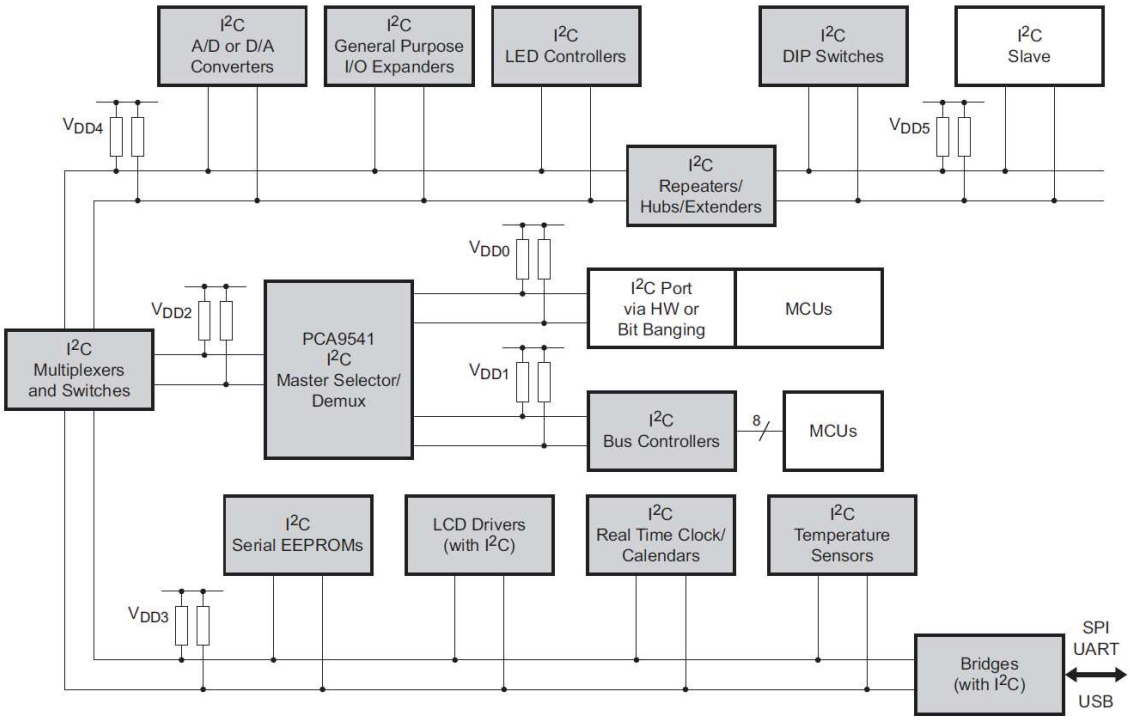
Velké množství I/O pinů jsou tzv. MPIO (multi-purpose digital I/O). To znamená, že kromě konfigurace GPIO je dále lze ještě alternativně konfigurovat jako tzv. SFIO (single function I/O). Právě SFIO konfigurace se využívá pro piny, u kterých požadujeme specifickou úlohu v komunikačním rozhraní (např. piny pro I^2C, SPI, atd.). [3]

3.2 Inter-Integrated Circuit (I^2C)

Protokol I^2C byl vyvinut za účelem zprostředkování jednoduchého a efektivního přenosu dat mezi vícero zařízeními na krátkou vzdálenost. [8]

Při komunikaci pomocí tohoto protokolu se používají dva vodiče – datový (SDA – serial data line) a hodinový (SCL – serial clock line). Tyto vodiče jsou přes pull-up rezistory připojeny na kladný pól napájecího napětí, jak můžeme vidět na

(Obr. 3.1). V klidovém stavu, kdy je sběrnice nepoužívaná, je tedy na obou kanálech úroveň H (logická 1). V průběhu komunikace pak vysílající zařízení připojuje své výstupy v příslušných okamžicích na zem – všechny zařízení připojené na sběrnici tedy musí mít své I/O piny typu otevřený kolektor (v případě bipolárního tranzistoru) nebo open-drain (v případě CMOS tranzistoru). Na sběrnici je tedy realizována tzv. wired-AND funkce (na kanálu je logická 1 pouze v případě, že je logická 1 i na I/O pinech všech zařízení v kanálu). [8]



Obr. 3.1 Různé možnosti použití I²C sběrnice [9]

Komunikující zařízení může být buď I²C Master nebo I²C Slave. I²C Master generuje hodinový signál, iniciuje a ukončuje komunikaci. K jednotlivým I²C Slave zařízením přistupuje pomocí 7-bitových nebo 10-bitových adres. Data mohou putovat v obou směrech, ovšem ne zároveň (half-duplex). [8]

3.2.1 Specifikace rozhraní na platformách

Konkrétní specifikace protokolu I²C pro obě platformy lze získat z příslušné dokumentace a jsou uvedeny v následující (Tab. 3.1):

Tab. 3.1 Srovnání parametrů I²C uvedených v dokumentaci

Platforma	Jetson Nano [3]	AURIX TC27X [8]
Délka adresy	7-bit	7-bit i 10-bit
Podporovaný mód	Master a Slave	Master, Multi-Master a Slave
Typ pinů	Open Drain – 1,8/3,3 V (2,2 kΩ/4,7 kΩ pull-up)	Open Drain – 3,3 V/[1]5,0 V (4,7 kΩ pull-up [10])
Rychlost komunikace	Standard-mode (0-100 kbit/s) Fast-mode (0-400 kbit/s) Fast-mode + (0-1 Mbit/s)	Standard mode (0-100 kbit/s) Fast mode (0-400 kbit/s) High-speed mode (0-3.4 Mbit/s)
Typ komunikace	half-duplex	half-duplex

Zejména nízká rychlost komunikace a podpora pouze half-duplex módu jsou hlavní důvody pro rozhodnutí nepoužívat toto rozhraní. Navíc, iLLD driver pro mikrokontrolér AURIX TC27X je určen primárně pro Master mód [11] a implementace Slave módu by nejspíš vyžadovala větší modifikace kódu. To samé platí pro standardní knihovnu *libi2c* pro platformu Jetson Nano. [12]

3.3 High Speed Serial Link (HSSL)

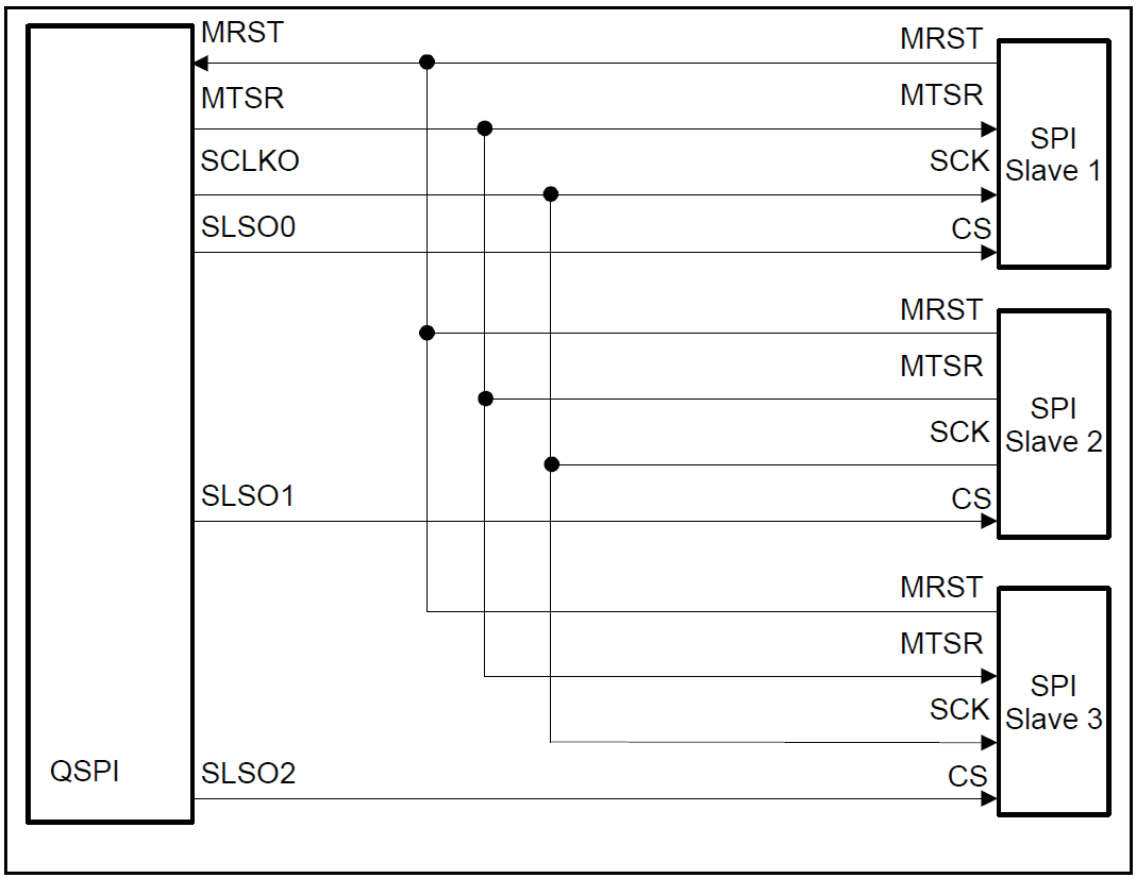
HSSL modul procesoru AURIX TC27X poskytuje point-to-point komunikaci mezi dvěma mikrokontroléry, nebo mezi mikrokontrolérem a jednodušším digitálním zařízením. Podporuje jak přímý zápis dat s velikostí 8/16/32 bitů iniciátorem komunikace do registru cílového zařízení, tak i přímé čtení z tohoto registru. Cílové zařízení se v podstatě chová jako externí paměť. To je umožněno díky nižším komunikačním vrstvám (datová a fyzická), které řídí modul HSCT. Komunikace pomocí tohoto protokolu probíhá podobně jako u Ethernetu pomocí tzv. rámců (frame) – data se umístí do HSSL frame a ten se zapouzdří do HSCT frame. Na úrovni modulu HSCT probíhá IC (inter chip) komunikace mezi master IC a slave IC, a to rychlostí 320 MBaud v high speed módu. [8]

Toto rozhraní by bylo pro propojení platforem ideální, Jetson Nano jím však bohužel nedisponuje. [3]

[1] Záleží na typu použitého kitu – např. Application Kit TC277 TFT má na pinech převážně napětí 5,0 V (až na výjimky – např. tzv. Flexport má napájení $V_{FLEX} = 3{,}3$ V a na této desce ho využívají piny pro Ethernet), zatímco Application Kit TC224 TFT (který ovšem I²C nepodporuje!) má na pinech převážně napětí 3,3 V

3.4 Serial Peripheral Interface (SPI)

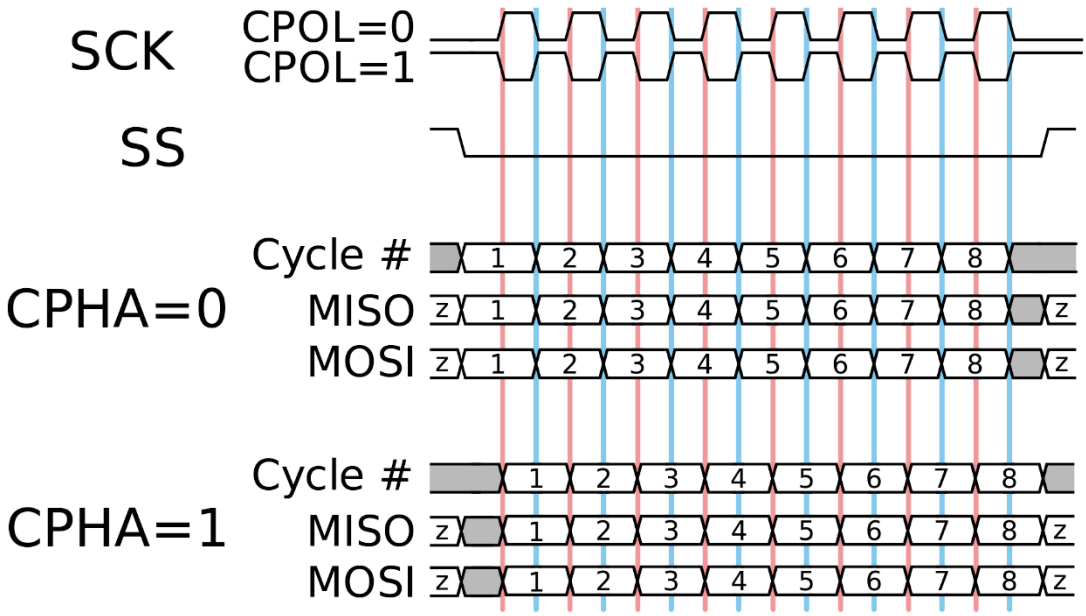
SPI je protokol pro obousměrnou synchronní komunikaci, který je založen na principu master-slave. Použití tohoto protokolu je značně rozšířené, zejména díky vysoké rychlosti a možnosti připojení vícero zařízení na stejnou sběrnici. Fyzická sběrnice je v základu tvořena 3 společnými vodiči s označením SCLK (někdy též SCK – Shift clock, Synchronous clock), MOSI (Mater Out, Slave In; někdy též MTSR – Master Transmitter, Slave Receiver) a MISO (Master In, Slave Out; někdy též MRST – Master Receiver, Slave Transmitter). Každému slave-zařízení dále přísluší samostatný vodič s označením SLS (nebo SS – Slave Select; někdy též CS – Chip Select). Zapojení pak může vypadat jako na (Obr. 3.2). [7]



Obr. 3.2 Vícero SPI slave-zařízení připojených k jedinému SPI modulu [7]

V dokumentaci k procesoru AURIX se používá terminologie „SPI modul“ (module), což je jediné master-zařízení, které může adresovat více než jedno slave-zařízení, a „SPI kanál“ (channel), což je označení pro jednu dvojici master-slave, které přísluší jeden SLS signál. [7] Protože jsou však SPI moduly procesoru AURIX schopny pracovat i ve slave módu, je pro ně v této práci ponecháno označení SPI modul, přestože je to v protikladu s předchozí definicí.

Mezi nejdůležitější součásti nastavení každého SPI kanálu patří parametry CPOL (Clock Polarity) a CPH (někdy též CPHA – Clock Phase). Jak se změna těchto parametrů projeví na časovém diagramu, je patrné na (Obr. 3.3). [8]



Obr. 3.3 Časový diagram SPI pro různé parametry CPOL a CPHA [13]

V případě, že CPHA = 0, znázorňuje oranžová linie na (Obr. 3.3) okamžik čtení dat (data read) a modrá linie okamžik změny vysílaného bitu (data shift). Pro CPHA = 1 je tomu právě naopak. Tento diagram platí pro nastavení, kdy je slave-zařízení aktivováno logickou 0 signálu SLS (většinou označováno jako CS-low, tedy Chip Select active low). Některá SPI zařízení ovšem umožňují i nastavení CS-high, tedy Chip Select active high. Přepnutím příslušného SLS do aktivní úrovně tedy master-zařízení (na Obr. 3.2 označeno QSPI) vybere slave-zařízení, se kterým chce komunikovat.

V této základní konfiguraci se sice jedná o full-duplex mód, neexistuje však způsob, jakým by mohlo master-zařízení zjistit, že slave-zařízení je připraveno na komunikaci (tzn. má data k odeslání). To lze vyřešit následně: zapojení vodičů se ponechá stejné, pouze se nahradí signál SLS signálem ENA, který nevysílá master-zařízení, ale slave-zařízení (v literatuře [14] se tento mód označuje 4-Pin With Enable Option, v [15] zase “ready“- slave pulls low to pause). Jak bude uvedeno dále, pro implementaci v této práci byla zvolena modifikace tohoto řešení, kdy byl signál ENA realizován dalším samostatným vodičem a pro jeho obsluhu byly využity GPIO piny. SPI lze provozovat i v módech s méně než 4 vodiči [8], ovšem poté už buď není tolik robustní (komunikace bez SLS signálu), nebo se

nejedná o full-duplex mód, který ale je vzhledem k požadavku ze druhého bodu zadání žádoucí.

Další důležitou součástí nastavení SPI kanálu je parametr LSB first, tedy že jsou jednotlivé bity vysílány od nejméně významného (tedy bitu s nejnižší hodnotou v binárním zápisu čísla). Pokud LSB = 0, pak je aktivní MSB first. [7]

3.4.1 Specifikace rozhraní na platformách

Konkrétní specifikace protokolu SPI pro obě platformy lze získat z příslušné dokumentace a jsou uvedeny v následující (Tab. 3.2):

Tab. 3.2 Srovnání parametrů SPI uvedených v dokumentaci

Platforma	Jetson Nano [3]	AURIX TC22X [16][2]	
Typ SPI	-	ASCLIN - SPI	QSPI
Šířka slova (bits per word, packet size)	1-bit až 32-bit	až 16-bit	2-bit až 32-bit
Podporovaný mód	Master (Slave mode has not been validated)	pouze Master	Master i Slave
Typ pinů	CMOS – (3,3 V)[3]	CMOS – 3,3 V (push-pull pro výstupní piny, pull-up pro vstupní piny) [17]	CMOS – 3,3 V (push-pull pro výstupní piny, pull-down pro vstupní piny) [17]
Rychlost komunikace	Master mode (až 65 Mbit/s) Slave mode (až 45 Mbit/s)	0,37 - 25 MBbit/s	až 50 Mbit/s (pro šířku slova 4-bit až 32-bit)
Vyrovnávací paměť (buffer)	Nezávislé RX FIFO a TX FIFO	Nezávislé RX FIFO (16 bajtů) a TX FIFO (16 bajtů)	Nezávislé RX FIFO (4 x 32-bit) a TX FIFO (4 x 32-bit)
Typ komunikace	full-duplex	full-duplex	full-duplex

[2] Pro komunikaci s Jetson Nano přes SPI nebylo možné bez přídavných prvků (např. napěťový dělič) použít AURIX™ Application Kit TC277 TFT (viz poznámka [1]), proto byl pro testování komunikace použit Application Kit TC224 TFT

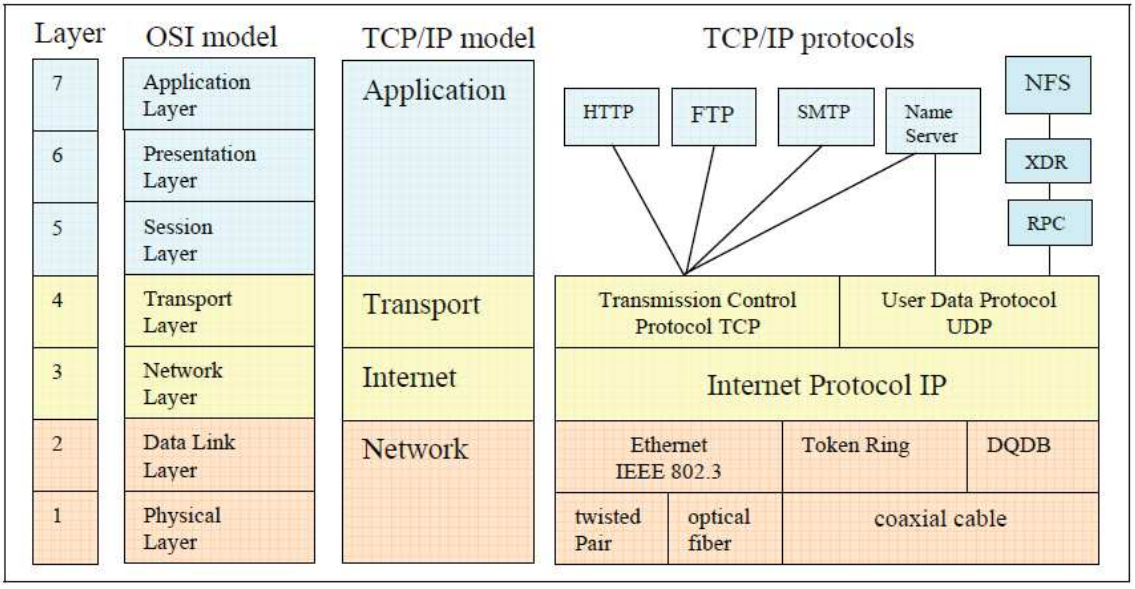
[3] [3] uvádí napětí 1,8 V, ovšem to je napětí přímo na Jetson Nano modulu. Jetson Nano Developer Kit obsahuje level shifter a na všech signálových pinech J-41 (Obr. 2.3) 40-pin expansion headeru (na něm se nachází i piny používané SPI) je pak napětí 3,3 V [1]

Pro aplikaci SPI rozhraní v příkladě ze třetího bodu zadání by se nabízela konfigurace master-procesor AURIX a slave-Jetson Nano. Jak však plyne z (Tab. 3.2), slave mód u platformy Jetson Nano nebyl výrobcem ověřen a rovněž se během psaní práce nepodařilo na internetu nalézt funkční postup, pomocí kterého by se dal tento mód implementovat. Jedinou možnou konfigurací tedy byla tato: master-Jetson Nano (SPI) a slave-procesor AURIX (QSPI).

QSPI (queued SPI) se od klasického SPI modulu liší tím, že konfigurační data pro jednotlivé SPI kanály se spolu se samotnými daty, které se mají na jednotlivé kanály vysílat, vkládají do fronty (queue). To je výhodné především pro případ, že se liší konfigurační parametry jednotlivých SPI kanálů (např. nastavení CPHA). SPI modul pak stačí jednou inicializovat a parametry pro jednotlivé SPI kanály se pak přenastavují bez nutnosti zásahu CPU. [16]

3.5 Ethernet Media Access Controller (MAC)

Způsob, jakým by se měly přenášet data mezi jakýmikoli dvěma body v telekomunikační síti, udává referenční model OSI. Tento model se skládá ze sedmi vrstev (Layers) a každá z nich popisuje úroveň komunikace. Na (Obr. 3.4) jsou znázorněny tyto vrstvy a jejich souvislost s vrstvami modelu TCP/IP, který ke své činnosti využívá Internet. [18]



Obr. 3.4 Porovnání modelů OSI a TCP/IP [18]

V případě této práce však komunikace probíhá pouze mezi dvěma zařízeními, odpadá tedy nutnost použití protokolu IP, určeného ke směrování (proces výběru cesty pro přenos dat v síti). Rovněž není nutné ani použití protokolů vyšších vrstev

a ke komunikaci postačuje použití Ethernetu (přesněji řečeno IEEE 802.3). Z TCP/IP modelu se poté uplatní pouze Network layer, respektive Data Link Layer a Physical Layer modelu OSI.

Ke každé vrstvě OSI modelu se váže tzv. protokolová datová jednotka (PDU). Na úrovni OSI Layer 1 jsou to jednotlivé bity, na úrovni OSI Layer 2 tzv. rámce (frames) a na úrovni OSI Layer 3 pak známé pakety. V praxi se tyto pojmy zaměňují, např. v dokumentaci k iLLD driverům [11] se o rámcích píše jako o paketech. Z toho důvodu je použit na některých místech termín paket i dále v této práci, přestože by toto označení mohlo být zavádějící.

3.5.1 Specifikace rozhraní na platformách

Propojení platforem pomocí kroucené dvojlinky (twisted Pair - Obr. 3.4), zakončené konektrory RJ-45, je umožněno díky Ethernetovému řadiči (Ethernet controller), který bývá označován (viz Obr. 2.4) jako tzv. PHY (zkratka pro *physical layer*), neboť vytváří fyzické rozhraní pro fyzickou vrstvu (Layer 1 OSI modelu) Ethernetu. [5] [3]

U Jetson Nano je PHY tvořeno čipem Realtek RTL81119IHS-CG, který je integrován do Jetson Nano modulu (Gigabit Ethernet na Obr. 2.1). [3] V případě AURIX™ Application Kit TC277 TFT se pak jedná o Lantiq Giagabit PHY PEF7071 (U203 na Obr. 2.5). [5]

Ačkoliv u obou PHY můžeme nalézt označení Gigabit Ethernet, v případě AURIX kitu je spojení mezi procesorem (který zde má roli řadiče MAC) a PHY realizováno pomocí rozhraní RMII, jak je patrné z (Obr. 2.4). Dle standardu IEEE 802.3u je v případě MII počet potřebných vodičů mezi MAC a PHY 16. RMII redukuje tento počet na 7 vodičů. Zároveň však podporuje pouze přenosové rychlosti 10 Mbit/s nebo 100 Mbit/s. [8] Označení Gigabit Ethernet je tedy značně zavádějící.

V dokumentaci pro Jetson Nano modul [3] je uvedeno, že podporuje 10/100/1000 Mbit/s Gigabit Ethernet a IEEE 802.3u MAC.

4 ZPROVOZNĚNÍ PLATFOREM A VÝVOJOVÉHO PROSTŘEDÍ

4.1 NVIDIA® Jetson Nano™ Developer Kit

Jak již bylo zmíněno, pro zprovoznění vývojového kitu je nejprve potřeba obstarat si microSD kartu, na kterou se pomocí nástroje na webové stránce [19] nahraje NVIDIA® JetPack™ SDK. Pro správnou činnost platforma (Obr. 2.3) dále vyžaduje buďto napájení přes Micro USB port (5 V, min. 2 A), přes J25 power jack (5 V, 4 A), nebo přes J41 expansion header (5 V, 5 A). V této práci byl pro napájení platformy použit adaptér MW7H380GTGS – v tomto případě je třeba dbát na správnou volbu konektoru pro J25 jack, který má rozměry: délka 9,5 mm, vnější průměr 5,5 mm a vnitřní průměr 2,1 mm. Konektor musí mít tzv. center positive polarity. Důležité je také nezapomenout umístit jumper na piny J48, což přepne zdroj napájení z Micro USB na J25 jack. [1]

Nakonec se k platformě, podobně jako k běžnému stolnímu PC, připojí klávesnice a myš přes USB 3.0 Typ A a monitor přes HDMI Typ A nebo DP. Pokud nejsou umístěny žádné jumpery na J40 headeru, připojením napájecího kabelu se rozsvítí zelená signalizační LED dioda vedle Micro USB konektoru a Jetson Nano nabootuje do standardního módu. Pokud se jedná o první spuštění, spustí se průvodce počátečním nastavením. [19]

Pro jednoduchou správu aktualizací softwaru je nezbytné zajistit pro Jetson Nano připojení k internetu. Je možné využít kabelové připojení pomocí konektoru RJ-45, to je ovšem již využíváno v jedné z částí této práce pro komunikaci s kitem AURIX. Proto byl pro internetové připojení využit 150 Mbit/s Wireless IEEE 802.11b/g/n nano USB adaptér Edimax EW-7811Un.

Možností pro vývojové prostředí (IDE) je hned několik, v této práci bylo použito Visual Studio Code, které lze v případě připojení k internetu nainstalovat a spustit pomocí několika jednoduchých příkazů, viz [20].

S Jetson Nano lze samozřejmě pracovat i v případě, kdy nemáme k dispozici samostatný monitor a klávesnici, případně jej potřebujeme řídit na dálku. Za tímto účelem jej můžeme ovládat přes SSH (Secure Shell), což je zabezpečený komunikační protokol v počítačových sítích využívajících TCP/IP. Stačí tedy připojit Jetson Nano do stejné lokální sítě, v jaké je připojen PC, kterým potřebujeme Jetson Nano ovládat. Následně je třeba zjistit IP adresu, která byla kitu přidělena pravděpodobně pomocí DHCP. To je možné buďto pomocí příkazu *ifconfig*, který spustíme přímo na Jetson Nano, nebo přes webové rozhraní routeru lokální sítě, kde bývá možnost zobrazení seznamu uživatelů DHCP. V tomto seznamu poté lze Jetson Nano najít podle jména zařízení, v případě kitu použitého

v této práci tedy *jetson-nano* (jméno zařízení se zobrazuje vždy na začátku příkazového řádku ve tvaru *<jméno_uživatele>@<jméno_zařízení>*, jak je vidět např. na Obr. 5.3). Poté už stačí jen spustit na ovládacím PC příkaz ve tvaru *ssh <jméno_uživatele>@<IP_adresa>*. Nyní lze Jetson Nano ovládat přes terminálové rozhraní. Celý postup je popsán v [21], kde jsou zmíněny i způsoby řešení problémů s SSH spojením.

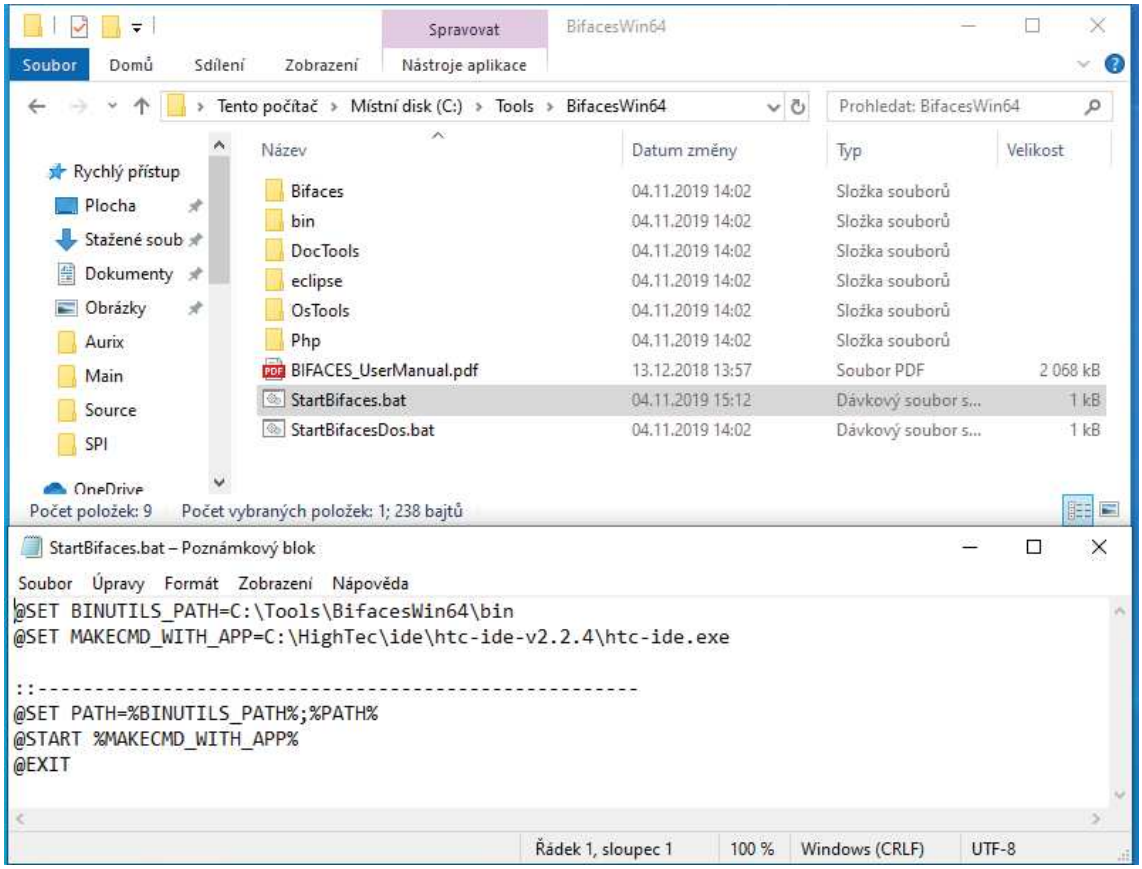
4.2 Procesory AURIX TC2xx

Nejprve je nutné stáhnout [22] a nainstalovat HighTec Free TriCore™ Entry Tool Chain, který v sobě integruje Eclipse IDE. V průběhu instalace se nesmí měnit přednastavená cesta k výchozí složce pro instalaci a rovněž se nesmí měnit její název. Součástí instalace je dále debugger UDE Starterkit 5.0 firmy PLS, který je nutný pro spouštění a debugování aplikací. Při instalaci na některých PC však bylo zaznamenáno, že nedošlo automaticky k instalaci pluginu UDE do prostředí Eclipse IDE. V tom případě je třeba otevřít v hlavní liště Eclipse IDE *Help->Install New Software...->Add...->Archive...* a vybrat soubor *UDEEclipse4Integration.zip*, který se po instalaci Free TriCore™ Entry Tool Chain nachází v instalační složce *...\pls\UDE Starterkit 5.0*. [23]

Dále je třeba stáhnout následující položky, které jsou k dispozici na webu [11] (tyto verze byly použity v práci):

- BIFACES_V1_0_3
- BaseProjects_AURIX1G_V1_0_1_11_0
- iLLD_1_0_1_11_0_TC2xx_Drivers_And_Demos_Release

BIFACES je prostředí poskytující jednotnou platformu pro vývoj softwaru pro AURIX mikrokontroléry. Po instalaci BIFACES je nutné v jeho instalační složce upravit soubor StartBifaces.bat (Obr. 4.1), pomocí kterého se následně spouští IDE. [24]



Obr. 4.1 Změna nastavení v souboru StartBifaces.bat

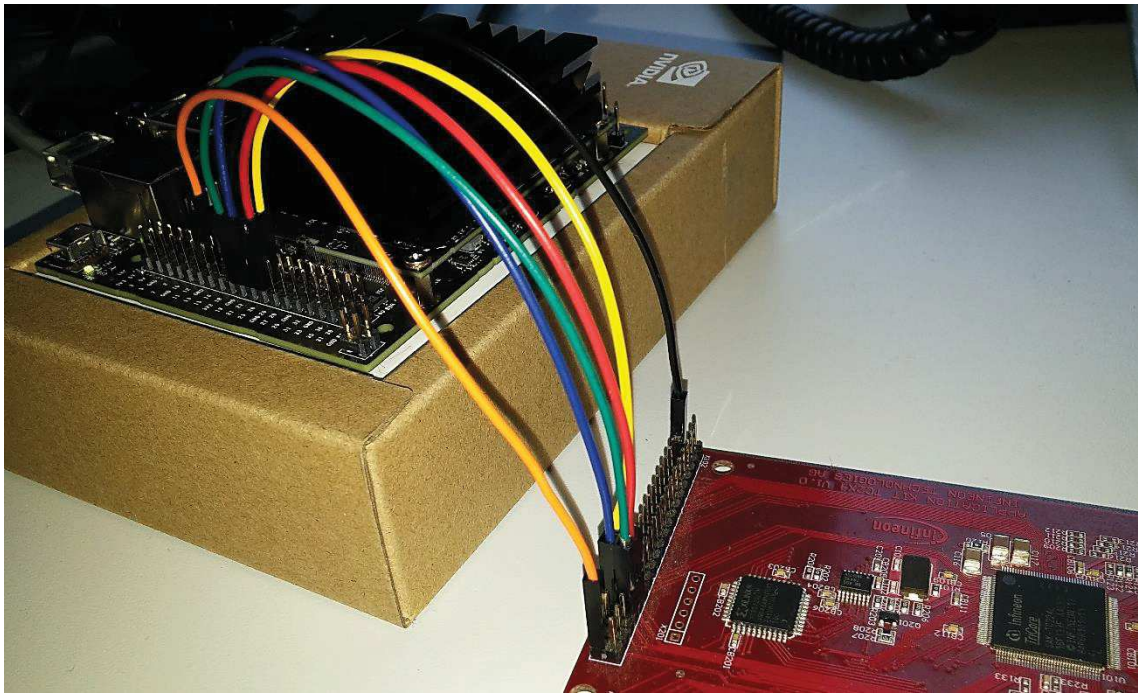
Pro obsluhu periferií procesorů AURIX již jsou vytvořeny tzv. iLLD (Infineon Low Level Drivers) a ukázkové kódy iLLD Demos. Šablony souborů se zdrojovými kódy, které jsou nezbytné pro každý vytvářený program, jsou pak umístěny ve složkách BaseFramework pro příslušný typ procesoru (tyto složky se vytvoří po instalaci BaseProjects AURIX1G). Tvorba programu většinou probíhá tak, že se do složky BaseFramework překopírují potřebné iLLD a iLLD Demo, které se následně modifikuje pro požadovanou aplikaci. Celý postup je podrobně popsán v [24]. Není v něm ovšem zmíněna skutečnost, že pro správnou činnost překladače je nutné mít nainstalovanou poslední verzi Microsoft Visual C++ Redistributable.

Za zmínku pak stojí užitečný debugovací nástroj prostředí UDE, který byl využíván v této práci – tzv. simulovaný IO terminál (nutno přidat soubory se zdrojovým kódem *simio_pls_tc.c* a *simio_pls.h*), který umožňuje použití funkcí jako *printf* a *scanf* pro interakci s uživatelem. [24]

5 IMPLEMENTACE SÉRIOVÉHO ROZHRANÍ PRO KOMUNIKACI MEZI PLATFORMAMI

5.1 Serial Peripheral Interface (SPI)

Jak je patrné z (Obr. 5.1), ke komunikaci přes SPI bylo použito 6 vodičů: MOSI (zelený), MISO (modrý), SCLK (červený), SLS (žlutý), ENA (oranžový) a černý vodič spojující GND piny obou desek, který je nezbytný pro vytvoření společného vztažného potenciálu. Teoreticky by se v tomto případě dalo obejít bez vodiče SLS, ale jak již bylo zmíněno, spojení by bylo méně robustní, protože slave-zařízení by muselo pro určení konce vysílaného rámce počítat přijaté bity, zatímco běžně určuje konec rámce právě SLS signál. [8] Propojení pinů použitých ke komunikaci je patrné z (Tab. 5.1).



Obr. 5.1 Propojení Jetson Nano a Application Kit TC224 přes SPI

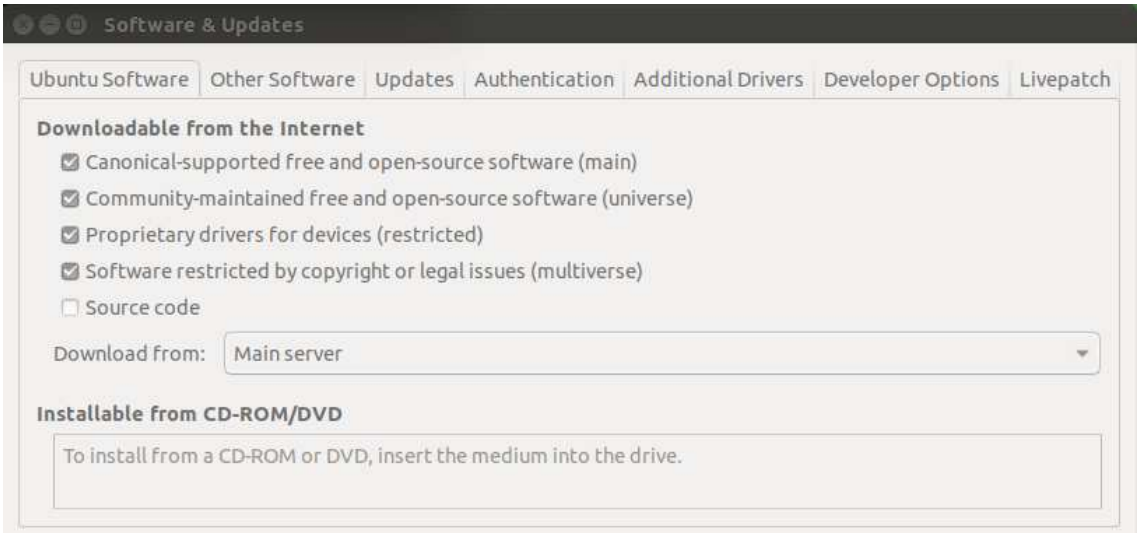
Za účelem testování byl k zobrazení SPI signálů nejprve používán osciloskop Rigol MSO1074Z. Zřejmě z důvodu nevhodného impedančního přizpůsobení však připojení osciloskopu k vodičům způsobilo, že signály byly zašuměné a přijatá data pak neodpovídala vysílaným. Pro testování spojení se tedy ukázalo jako vhodnější zajistit co nejkvalitnější (nejkratší) spojení a přijatá data kontrolovat na výstupu na terminál ze spuštěných programů.

Tab. 5.1 Seznam propojení pinů pro SPI komunikaci

	Jetson Nano (master)		AURIX TC224 (slave)	
Signál	**Název pinu**	**Číslo pinu na headeru J41**	**Název pinu**	**Číslo pinu na headeru X102**
MOSI	gpio16	19	P15.5	34
MISO	gpio17	21	P15.4	33
SCLK	gpio18	23	P15.3	32
SLS	gpio19	24	P15.2	31
ENA	gpio79	12	P14.4	39
GND	GND	20	GND	3

5.1.1 NVIDIA® Jetson Nano™ Developer Kit

Než bylo vůbec možné začít s vytvářením programu pro komunikaci, bylo třeba změnit funkci příslušných pinů na 40-pinovém J41 headeru (Obr. 2.3), protože piny pro SPI jsou defaultně nastaveny jako GPIO vstupy. Tuto změnu funkce pinů popisuje výrobce v manuálu [25]. Jak se však lze dočíst např. v tomto příspěvku na NVIDIA Developer fóru [26], v době, kdy se problém řešil, byl postup popsaný v tomto návodu pro tehdy nejaktuálnější verzi L4T 32.2.1, použitou v této práci, nefunkční. Kvůli tomu byl vytvořen skript na stránce [27], který slouží k aktivování pinů 19, 21, 23 a 24 pro funkci SPI (viz SPI_1 na Obr. 0.2). Pro spuštění tohoto skriptu je zapotřebí samostatné PC s operačním systémem Ubuntu, ke kterému se pomocí kabelu s Micro USB konektorem připojí Jetson Nano, spuštěné ve Force Recovery módu (pomocí jumperu se propojí FRC piny na J40 headeru (Obr. 2.3) vypnutého Jetson Nano a připojí se napájení). Pro tyto účely postačuje free verze Ubuntu, nabootovaného např. z flash disku. Dále je nutné připojení k internetu a povolení stahování softwaru z různých zdrojů, viz (Obr. 5.2).



Obr. 5.2 Nastavení stahování softwaru z internetu v systému Ubuntu

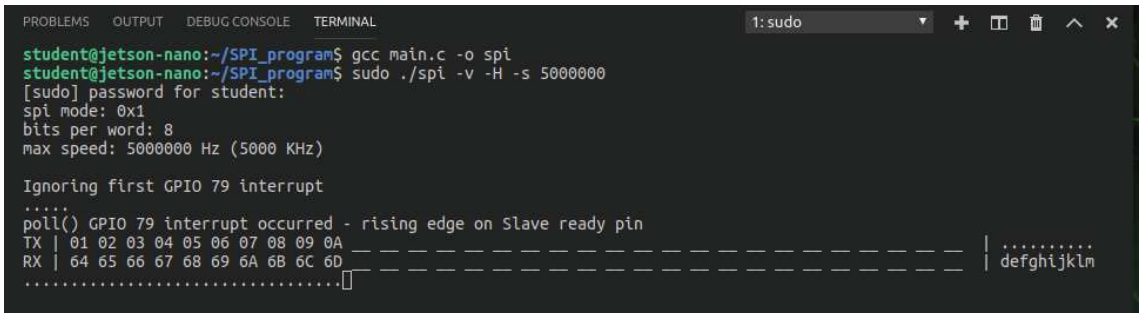
Poté už stačí pouze spustit skript a odsouhlasit flash na Jetson Nano (na rozdíl od postupu popsaného v [25] není potřeba instalovat předem žádný dodatečný software, skript si jej automaticky stáhne, případně aktualizuje, pokud je to zapotřebí). SPI_1 je poté dostupné pomocí adresáře */dev/spidev0.0*.

Pro samotnou obsluhu */dev/spidev0.0* v jazyce C byla použita část kódu z webové stránky [15], která byla následně umístěna do hlavičkového souboru *spi.h*. Pro obsluhu signálu ENA poté byla využita část kódu z [28], který umožňuje práci s GPIO piny. Tento kód je obsažen v hlavičkovém souboru *gpio.h*. Hlavní kód pro obslužný program pak obsahuje *main.c.* Tyto soubory se zdrojovými kódy jsou umístěny v Příloze 2.

Program se překládá pomocí příkazu *gcc*, který lze zavolat z terminálu přímo v prostředí Visual Studio Code (Obr. 5.3). Přeložený program *spi* je zde spuštěn s parametry: *-v* (zobrazení vyslaných bajtů TX), *-H* (nastavení CPHA na 1, defaultně totiž CPHA = 0, CPOL = 0) a *-s 5 000 000* (nastavení rychlosti na 5 MBbit/s). Při každé náběžné hraně signálu ENA (gpio79 na Obr. 0.2) potom proběhne současné vyslání a příjem 10 bajtů dat – Jetson Nano vysílá čísla 0x1 až 0xA (dekadicky 1-10), kit AURIX čísla 0x64 až 0x6D (dekadicky 100-109). Jak je vidět, přijatá data (RX) jsou v pořádku.

Pomocí dalších parametrů funkce main *-i*, *-o* lze dále nastavit vysílání dat z nebo ukládání dat do souboru, což by mohlo být pro další využití užitečné.

Program je pro účely testování realizován tak, že čeká neomezeně dlouho na náběžnou hranu signálu ENA, nebo na standardní vstup z klávesnice STDIN read (stisknutí klávesy Enter). V prvním případě začne přenos dat přes SPI, v případě STDIN read se program korektně ukončí. Z nějakého důvodu nastává první přerušení náběžnou hranou ENA vždy, bez ohledu na logickou úroveň tohoto signálu – proto se první přerušení ignoruje a přenos neprobíhá.



Obr. 5.3 Obslužný program SPI na Jetson Nano

5.1.2 AURIX™ Application Kit TC224 TFT

Pro tvorbu obslužného programu byl modifikován ukázkový kód *QspiCpuDemo* (součást [17]). Navzdory očekávání podporuje QSPI ve slave módu pouze jediné nastavení CPHA = 1 a CPOL = 0. [16] Proto bylo nutné přizpůsobit tomuto nastavení SPI mód na straně Jetson Nano.

Hlavní modifikací bylo přenastavení použitých pinů, neboť u každého AURIX kitu se mnohdy liší jejich rozložení. Např. nosné desky kitů TC277 a TC224 jsou sice velmi podobné, ale rozložení pinů na headerech je rozdílné.[4] Na rozdíl od Jetson Nano mají piny tohoto kitu více než 1 možnou SFIO konfiguraci, a proto bylo také více možností, které piny pro QSPI zvolit – všechny použitelné piny pro QSPI jsou definovány v souboru *IfxQspi_PinMap.c* (viz [17]). Piny zvolené v naší aplikaci jsou patrné z (Tab. 5.1) a jejich umístění na headeru je vidět na (Obr. 0.1).

Tyto piny náleží QSPI2 modulu, který je však již používán i pro Multi Voltage Safety Micro Processor Supply TLF35584 (Obr. 2.4). To způsobuje, že i po inicializaci modulu obslužným programem z *QspiCpuDemo* jsou před začátkem přenosu bajty v RX FIFO (buffer pro příchozí data). V průběhu komunikace jsou z bufferu nejprve vyčteny tyto zbytkové bajty a až poté užitečná data – vznikne tedy nežádoucí posuv. Aby se tomuto zamezilo, musí se při inicializaci QSPI modulu nastavit bity RESETS v registru GLOBALCON na hodnotu 0111_B, čímž se resetuje stavový automat QSPI modulu a vymažou se TX FIFO a RX FIFO. [16]

Pro účely testování je program realizován tak, že před každým uvedením signálu ENA do úrovně logické 1 (tedy před začátkem přenosu stanoveného počtu bajtů) se čeká na stisk libovolné klávesy – tím se ověřuje, že master-zařízení začne komunikaci až v okamžiku, kdy je slave-zařízení připraveno ke komunikaci.

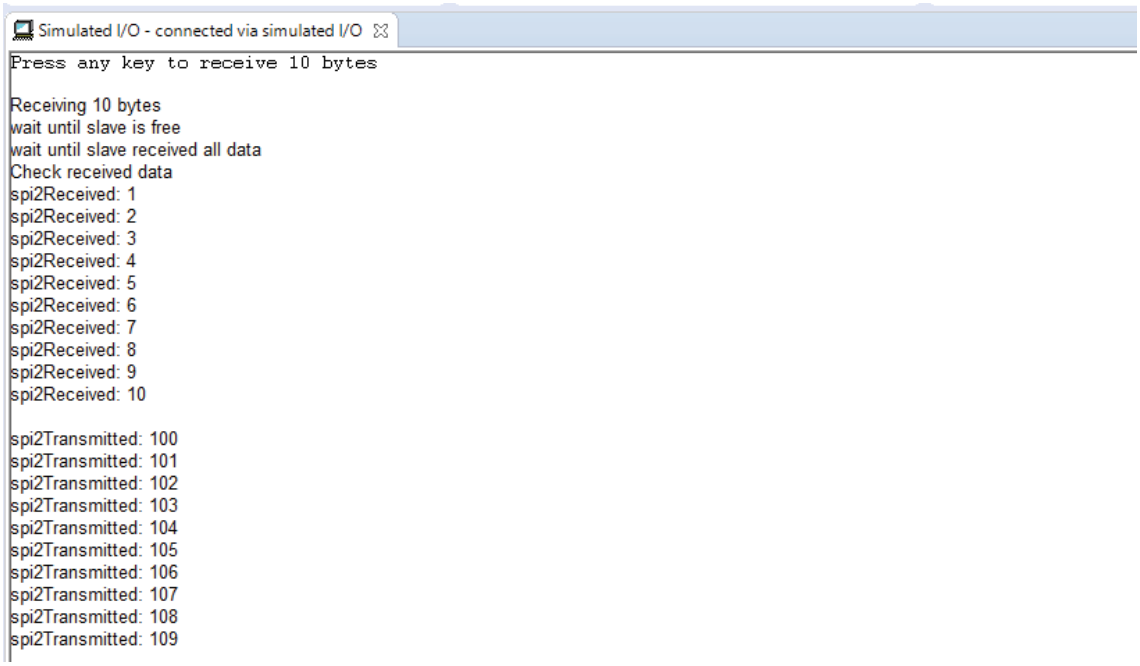
Výstup na simulovaném terminálu při spuštěném programu je vidět na (Obr. 5.4). Je patrné, že přijatá data odpovídají vyslaným z platformy Jetson Nano.

[4] Dále je třeba zjistit z datasheetů i typ použitých pinů. Např. pin P22.0 je sice na obou deskách umístěn na stejné pozici X102 headeru, ale v případě TC277 se jedná o tzv. LVDS pin, který je pro účely SPI v této práci nevyhovující. [10]

V případě *QspiCpuDemo* funguje obousměrný přenos dat bez problému zhruba do rychlosti 10 Mbit/s, což je však výrazně nižší maximální rychlost, než jakou uvádějí výrobci (viz Tab. 3.2). Při této rychlosti se po opakovaném přenosu dat občas stane, že procesor AURIX vyšle místo validního bajtu bajt se samými "1", což je reakce na podtečení TX FIFO (vysílané bajty se nestíhají umístit do výstupního bufferu QSPI modulu [16]).

Ukázalo se, že výrazně vyšší přenosové rychlosti (až 17,5 Mbit/s) lze dosáhnout použitím tzv. DMA (Direct Memory Access) módu, který QSPI modul podporuje. V tomto módu dochází k přímému přenosu dat mezi operační pamětí a vstupně výstupním zařízením. V prvním kroku uživatelský software konfiguruje příslušný DMA kanál – jeden pro vysílání a druhý pro přijímání dat. Uvažujme vysílání dat – v tomto případě nenaplněné TX FIFO použitého QSPI modulu neustále generuje čekající požadavek DMA. Následně časovač spustí softwarovou rutinu, která aktivuje DMA kanál. Ten začne načítat odesílaná data z paměti RAM do TX FIFO. Po dokončení načtení generuje DMA přerušení, které obslouží uživatelský software. Důležité je, že přenos dat probíhá bez nutnosti zásahu CPU, který může mezitím zpracovávat jiné instrukce. Tato posloupnost operací je identická pro DMA kanál zajišťující příjem dat. [16] Pro implementaci DMA poskytuje výrobce ukázkový kód *QspiDmaDemo* (součást [17]), jehož úprava byla obdobná jako v případě *QspiCpuDemo*.

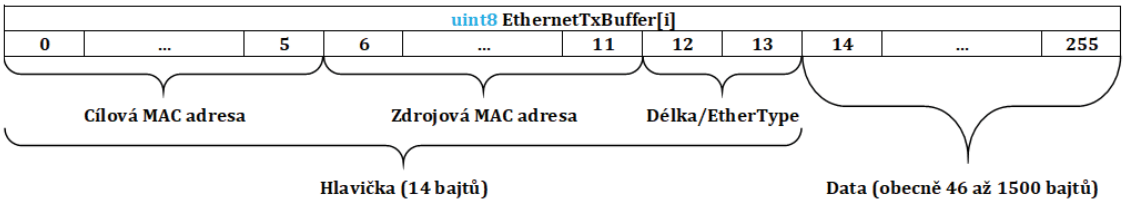
Ukázka adresářové struktury souborů se zdrojovými kódy obslužného programu je na (Obr. 0.3). Modifikované soubory se nacházejí v Příloze 2.



Obr. 5.4 Výpis obslužného programu QSPI v prostředí UDE

5.2 Ethernet Media Access Controller (MAC)

Propojení platforem je jednoduché – byl využit běžně používaný Ethernetový kabel (kroucená dvojlinka), kterým byly propojeny RJ-45 sloty (J43 Ethernet Jack na Obr. 2.3, ETH na Obr. 2.5). Komunikace mezi platformami pak probíhala na úrovni Layer 2 OSI modelu, u hlavičky rámců (viz Obr. 5.5) tedy stačilo nastavit pouze cílovou MAC adresu (Jetson Nano - statická), zdrojovou MAC adresu (AURIX – možno nastavit libovolnou) a 2 bajty představující délku posílaných dat (length of payload).



Obr. 5.5 Struktura Ethernetového rámce na úrovni programového rozhraní[5]

V některých typech rámců linkové vrstvy (OSI Layer 2) udává hodnota těchto 2 bajtů protokol dat zapouzdřených v datové části rámce. To je v případě, kdy je tato hodnota větší nebo rovna 1536 (0x0600) a označuje se jako tzv. EtherType. [29] Pokud je tato hodnota menší nebo rovna 1500, udává právě délku posílaných dat, což je případ komunikace v této práci, protože rámce nezapouzdřují žádný protokol vyšší komunikační vrstvy a datová část rámce obsahuje pouze užitečná data.

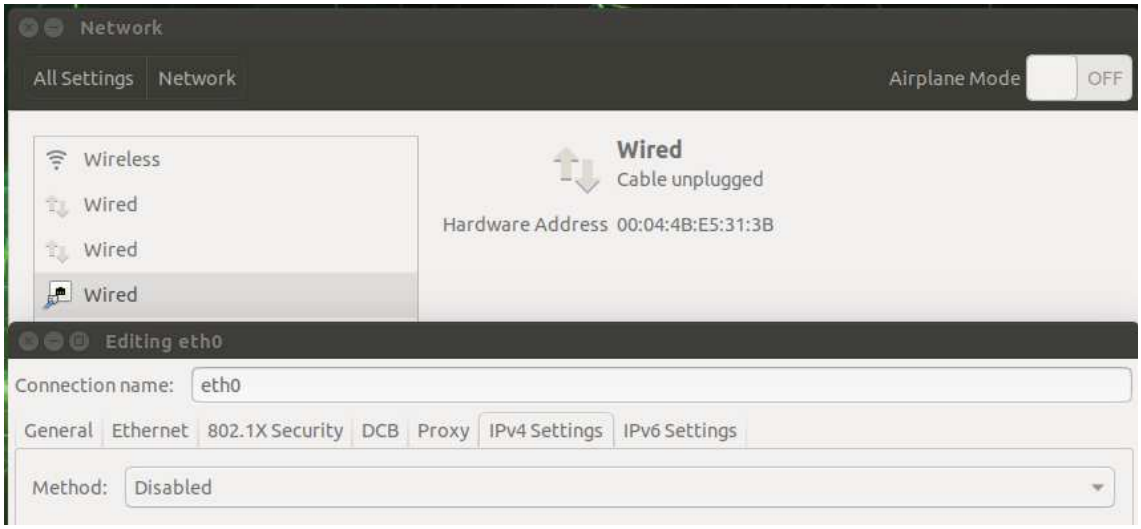
5.2.1 NVIDIA® Jetson Nano™ Developer Kit

Výrobce vydal Ethernet Firmware Update pro všechna zařízení Jetson Nano™ Developer Kit, která byla vyrobena dříve než v 21. týdnu roku 2019. Použitý školní kit byl vyroben ve 20. týdnu roku 2019, takže byl nejprve nutný tento update. [30]

Dále je na rozhraní eth0 (Ethernetový port) ve výchozím nastavení zapnuto dynamické přidělování IP adresy pomocí DHCP. Přidělení IP adresy, která se uplatňuje až na úrovni Layer 3 OSI modelu, v tomto případě není nutné. Ponechání tohoto nastavení má za následek, že Jetson Nano vyšle požadavek na DHCP server, který není dosažitelný. Po nějaké době pak zahlásí Jetson Nano ztrátu spojení a

[5] Ethernetový rámec, jakožto PDU linkové vrstvy modelu OSI (Layer 2), obsahuje kromě hlavičky ještě tzv. trailer na konci rámce, ve kterém je kontrolní posloupnost rámce FCS. FCS využívá cyklický redundantní součet CRC, který slouží k rozpoznání poškozeného rámce. [18] Trailer rámce se však na obou platformách vytváří a zpracovává automaticky díky Ethernetovým driverům, proto není potřeba zpracovávat jej v uživatelském programu. Podobně se nemusí zpracovávat další metadata, které se připojují k rámci na fyzické vrstvě modelu OSI (Layer 1).

požadavek na DHCP server se opakuje, a takto stále dokola. Tato zbytečná komunikace brzdí přenos užitečných dat, je tedy vhodné změnit nastavení IPv4 rozhraní eth0 kabelového připojení k síti, viz (Obr. 5.6).

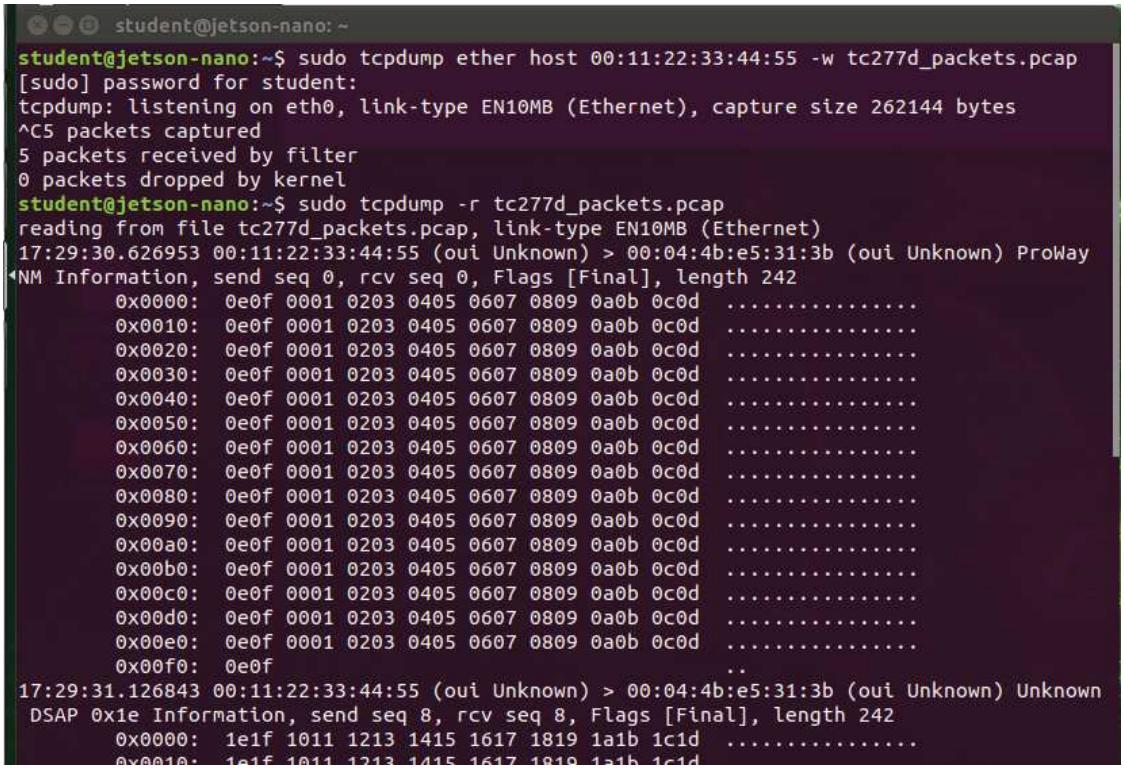


Obr. 5.6 Vypnutí DHCP na platformě Jetson Nano

5.2.1.1 Jednosměrná komunikace

Pokud je jedinou úlohou Jetson Nano analýza dat vyslaných kitem AURIX, bez nutnosti odpovědi na přijatá data, postačuje pro ukládání rámců přicházejících z kitu AURIX např. známý linuxový analyzátor paketů *tcpdump*. Návod na jeho instalaci lze mimo jiné nalézt např. na [31] (zde je popsána zajímavá aplikace Jetson Nano pro odchytávání bezdrátové 802.11ax komunikace).

Poté stačí na Jetson Nano vyfiltrovat rámce s MAC adresou kitu AURIX a uložit je do souboru .pcap, jak je patrné z (Obr. 5.7). Obsah přijatých rámců odpovídá datům vyslaným kitem AURIX.



Obr. 5.7 Ukázka použití *tcpdump* pro uložení rámců z kitu AURIX

5.2.1.2 Obousměrná komunikace

Pokud je požadován přenos dat i v opačném směru, tedy kupříkladu vyslání odpovědi na přijatá data z platformy NVIDIA Jetson, je třeba vytvořit vlastní program. Pro tento účel byl upraven kód pro vysílání/přijímání tzv. raw Ethernetových rámců, který je dostupný z [32].

Stěžejní částí kódu je využití tzv. Socket API. Socket představuje programové komunikační rozhraní uzlu pro příjem a odesílání dat po síti (datagramů, paketů i rámců). Nejdříve je třeba vytvořit/otevřít toto rozhraní. Socket se poté chová jako otevřený soubor, je tedy identifikovaný tzv. souborovým deskriptorem. Programové rozhraní Socket API je stejné pro všechny unixové operační systémy (tedy i L4T). V unixových systémech je Socket API přímo součástí C knihovny, stačí do kódu začlenit příslušné hlavičkové soubory. [33]

```
18    #define ETHER_TYPE ETH_P_ALL /*ETH_P_ALL - pro vsechny Ethernetove linkove protokoly*/
19    #define LENGTH_OF_PAYLOAD 0x00f2 /*242 bajtu*/
20  ⊞ ...
93      sock = socket (AF_PACKET, SOCK_RAW, htons (ETHER_TYPE));
94  ⊞ ...
189     /* Construct ehternet header. */
190     {
191       struct ether_header *eh;
192   ⊞ ...
197       eh->ether_type = htons (LENGTH_OF_PAYLOAD); /*misto EtherType zde zadavam length of payload*/
```

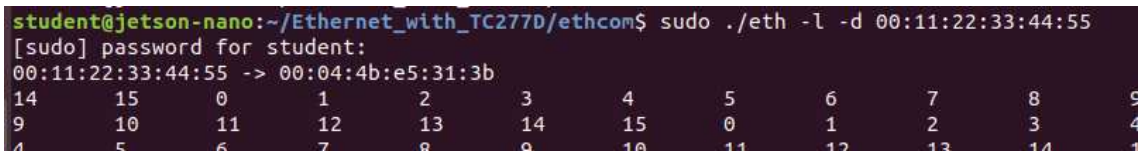
Obr. 5.8 Výňatek z *Zdrojove_kody\Jetson_Nano\Ethernet\ethcom.c*

Způsob použití Socket API je patrný z (Obr. 5.8). Pro Ethernetové rámce se používají sockety z domény AF_PACKET, tzv. paketové sockety. V této doméně lze vytvořit pouze dva typy socketů, SOCK_DGRAM nebo SOCK_RAW. Pro odeslání/příjem celých linkových rámců včetně hlavičky je nutno použít socket typu SOCK_RAW. Poslední parametr představuje požadovaný linkový protokol (EtherType), v našem případě Ethernet, pro který je definováno makro ETH_P_ALL. Linkový protokol označují 2 bajty, kvůli endianitě se tedy musí použít funkce *htons* pro převod z lokálního do síťového tvaru (neboli network byte order, což je big endian). [33]

Dále v programu se takto vytvořený deskriptor *sock* použije nejdříve pro příjem rámců pomocí funkce *recvfrom*, která zkopíruje obsah přijatého rámce do bufferu (pole znaků typu *char*, tj. pole bajtů), se kterým se dále pracuje. Pokud je cílová MAC adresa v přijatém rámci shodná s MAC adresou rozhraní eth0 na Jetson Nano, vytiskne se datová část tohoto rámce na terminál a zároveň se zkopíruje do bufferu pro odesílaný packet (Jetson Nano odešle zpět do AURIXu stejná data, která přijal). Mohlo by se zdát zbytečné filtrovat přijaté rámce podle MAC adresy, protože na rozhraní eth0 je připojeno jediné zařízení, a to AURIX™ Application Kit TC277 TFT. Jak však bude vysvětleno později, je tento krok důležitý.

Po přijetí prvního rámce s odpovídající MAC adresou se pokračuje vytvořením hlavičky rámce s odpovědí. To obnáší vyplnění zdrojové a cílové MAC adresy a hodnoty EtherType. Jak již bylo zmíněno, tato hodnota odpovídá délce posílaných dat. (Obr. 5.8) Rámec se poté vyplní daty a pomocí deskriptoru *sock* a funkce *sendto* se pošle zpět na TC277. Tím se program ukončí.

Program lze spouštět z příkazové řádky, jak je patrné na (Obr. 5.9). Pro příjem rámců slouží parametr *-l* (listen), pomocí parametru *-d* se nastavuje cílová (destination) MAC adresa. Dále je možné pomocí parametru *-i* změnit použité rozhraní, implicitně je zvoleno rozhraní eth0.



Obr. 5.9 Ukázka použití programu pro obousměrnou komunikace přes Ethernet

5.2.2 AURIX™ Application Kit TC277 TFT

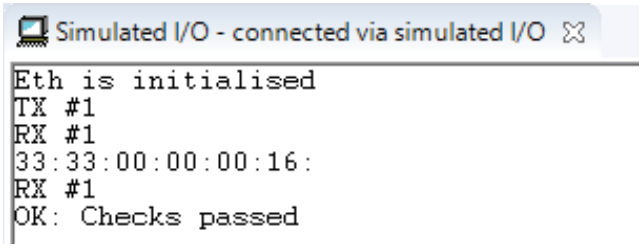
Pro tvorbu obslužného programu byla použita modifikace ukázkového kódu *EthDemo* (součást [17]). Především bylo třeba definovat cílovou MAC adresu platformy Jetson Nano a vypnout tzv. PHY loopback mód. Ze stejných důvodů jako v případě QSPI se musely správně nastavit použité piny pro AURIX™ Application Kit TC277 TFT. Modifikované soubory se nacházejí v Příloze 2.

5.2.2.1 Jednosměrná komunikace

Pro tento případ byla zachována funkčnost podle původní koncepce demo kódu, kdy AURIX vyšle za sebou 5 rámců o velikosti 256 bajtů s testovacími daty délky 242 bajtů (viz Obr. 5.5). Nutné úpravy kódu jsou minimální, zejména pak díky implementaci algoritmu Auto-Negotiation, který slouží k automatickému nastavení Ethernetové komunikace. Pomocí něj dojde po propojení Ethernetových portů k automatickému nastavení rychlosti přenosu a duplex módu na nejrychlejší možnou variantu. [18] V případě použitých platforem se pak jedná o 100 Mbit/s full-duplex.

5.2.2.2 Obousměrná komunikace

Program byl dále upraven tak, aby byl odeslán jeden testovací rámec a následně se čekalo na odpověď od Jetson Nano. Jakmile je přijat rámec, začne jeho kontrola. Nejprve se kontroluje cílová MAC adresa v hlavičce rámce. Pokud se liší od očekávané, vypíše se tato odlišná MAC adresa na terminál a pokračuje se v čekání na příchozí rámec, jak je patrné z (Obr. 5.10). Pokud se cílová MAC adresa shoduje s MAC adresou kitu AURIX, pokračuje se v kontrole rámce. Jestliže jsou všechna data shodná s odeslanými, vypíše se na terminál potvrzení jako na (Obr. 5.10). V opačném případě se vypíše počet chyb.

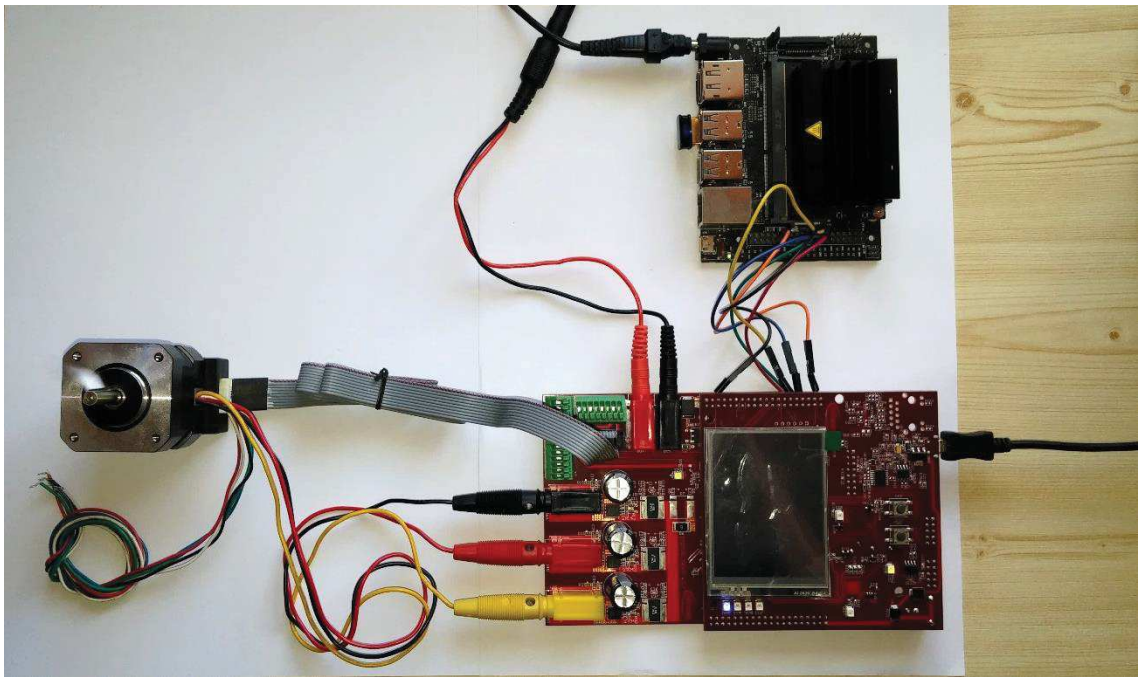


Obr. 5.10 Výpis programu pro Ethernetovou komunikaci na TC277

AURIX tedy před přijetím rámce s očekávanými daty přijme rámec s cílovou MAC adresou 33:33:00:00:00:16. Jedná se o multicastovou adresu specifickou pro protokol MLDv2 (Multicast Listener Discovery version 2). Účelem tohoto protokolu je umožnit IPv6 routerům objevit přímo připojené multicastové posluchače. [34] Je jasné, že pro účely komunikace mezi 2 platformami nemá tento protokol žádný smysl. Byla vyvíjena snaha najít způsob, jak na Jetson Nano vypnout vysílání rámců tohoto protokolu a omezit tak zbytečné zatížení Ethernetové linky, bohužel však nebylo nalezeno žádné funkční řešení. Proto je nutné na obou platformách filtrovat přijaté rámce podle MAC adresy, aby se nezpracovávaly rámce s multicastovými a broadcastovými adresami, které nemají v této konfiguraci žádný význam.

6 PŘENOS MĚŘENÝCH DAT PŘI ŘÍZENÍ MOTORU PROCESOREM AURIX

Dle třetího bodu zadání bylo úkolem otestovat komunikační rozhraní mezi platformami na konkrétním příkladě, kdy procesor AURIX řídí elektrický motor a data naměřená při tomto řízení jsou přenášena na platformu Jetson Nano. Za tímto účelem byl v práci použit AURIX MotorControl Application Kit, jehož součástmi jsou BLDC motor Nanotec o maximálním výkonu 40 W a řídicí deska eMotor Drive Kit V2.1. Tato řídicí deska vytváří rozhraní mezi motorem a procesorem a je konstruována přímo pro použití Application Kit TC2x4 TFT, se kterým se spojuje přes X102 a X103 konektory (Tab. 0.1 a Tab. 0.2). Jedinou možností bylo tedy použití Application Kit TC224 TFT, který byl k dispozici. Z hlediska rychlosti komunikace by pro přenos měřených dat do platformy Jetson Nano bylo lepší použít Ethernet, ten však bohužel TC224 nepodporuje. Tím pádem se volba komunikačního rozhraní omezila na SPI.



Obr. 6.1 Kompletní zapojení pro přenos dat při řízení motoru

Piny použité pro SPI jsou na obou platformách stejné jako při testování SPI komunikace (viz Tab. 5.1). Nastal ovšem menší problém s napojením vodičů na piny kitu AURIX, které se nacházejí na X102 headeru, protože tyto se zasouvají do řídicí desky eMotor Drive Kit V2.1. Nabízela se možnost napájet vodiče přímo na kit AURIX ze strany displeje, což by ovšem znemožňovalo snadnou změnu použitých pinů a pro účely testování se to jevilo jako nepraktické řešení. Proto byly

namísto toho zhotoveny provizorní konektory s úzkou objímkou, které se nasadí na piny před zasunutím do desky eMotor Drive Kit V2.1. Vodiče (wire jumper male to female) se poté mohou k desce připojit z boku přes nasazené konektory, které jsou opatřeny izolační páskou pro eliminaci kontaktu vodičů (použité piny jsou těsně vedle sebe).

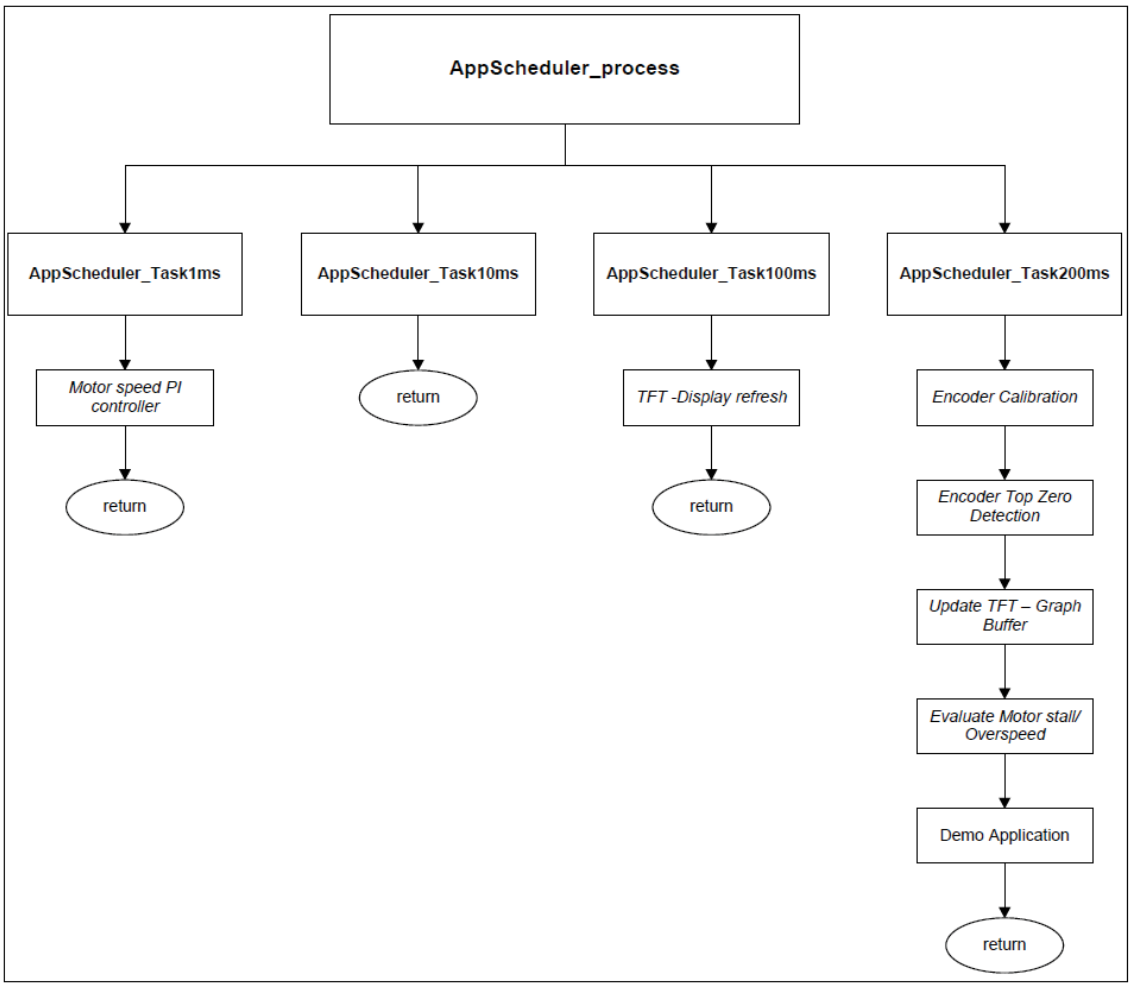
Deska eMotor Drive Kit V2.1 je určena pro řízení třífázových bezkartáčových synchronních motorů s permanentními magnety pomocí tzv. vektorového řízení FOC. Její hlavní součástí je integrovaný obvod TLE9180, který převádí PWM signály generované procesorem na řídicí signály pro střídač. Jeho konfigurace a diagnostika je prováděna pomocí rozhraní QSPI3 kitu AURIX. Pro snímaní otáček, respektive natočení hřídele motoru, jsou na řídicí desce přítomny konektory pro resolver, enkodér a Hallův snímač. Pro zajištění dostatečného výkonu se k desce připojuje samostatný napájecí zdroj, jak je vidět na (Obr. 6.1). Z tohoto obrázku je rovněž patrné, že pro snímání otáček motoru je v tomto zapojení použit pouze enkodér, ostatní snímače jsou odpojeny. [35]

6.1 AURIX™ Application Kit TC224 TFT

Pro řízení motoru Infineon dodává aplikaci, která používá starší verzi iLLD driverů. Pro aplikace v této práci však byla použita novější verze driverů. Nebylo by rozumné a možná ani realizovatelné, spojovat části programů v různých verzích iLLD driverů. Vzhledem k tomu, že cílem této práce nebylo vytvoření programu pro řízení motoru, byl upravený program využívající nové iLLD drivery pro řízení motoru převzat od kolegy z magisterského studia Bc. Davida Buchala. Jeho aplikace pomohla k integraci a otestování částí kódu vytvořených v této práci. Jedná se o projekt pro HighTec Development Platform s názvem *BIFACES_TC224A_eMotor_1v31_HW21_Enc_V3*.

Tento projekt byl původně vytvořen pro Application Kit TC234 TFT, na kterém využíval ovládání pomocí dotykového displeje. Jak se však ukázalo, vykreslování na displej je značně paměťově náročná operace. TC234 má však oproti TC224 více než dvakrát větší paměť RAM.[6] Při kompilaci projektu pro TC224 proto vznikla chyba způsobená nedostatkem paměti. Řešením bylo vzdát se ovládání přes TFT displej a zobrazování měřených dat na něj. To bylo provedeno poměrně jednoduše, a to zakomentováním inicializační funkce pro TFT displej v souboru *Cpu0_Main.c* a částí kódu pro zobrazování dat na displeji v souboru *AppTaskFu.c*.

[6] DSPR (Data Scratch-Pad RAM) SRAM má na TC224 velikost 88 KB, kdežto na TC234 184 KB [16]



Obr. 6.2 Plánovač úloh aplikace pro řízení motoru [35]

Aplikace se po spuštění inicializuje ve funkci *core0_main().* Následuje inicializace plánovače úloh (task scheduler), kdy se nastaví, které operace se budou vykonávat každou milisekundu (*appTaskfu_1ms()*), každých 10 ms (*appTaskfu_10ms()*), každých 100 ms (*appTaskfu_100ms()*) a každých 200 ms (*appTaskfu_200ms()*). V případě potřeby je možno v souboru *Ifx_Cfg_Scheduler.c* nadefinovat další časové sloty. Plánovač úloh poté běží permanentně v nekonečné smyčce. Pro každý časový slot je poté v příslušném okamžiku generováno přerušení – v obslužné rutině se poté vykonávají požadované operace, jak je znázorněno na (Obr. 6.2). [35]

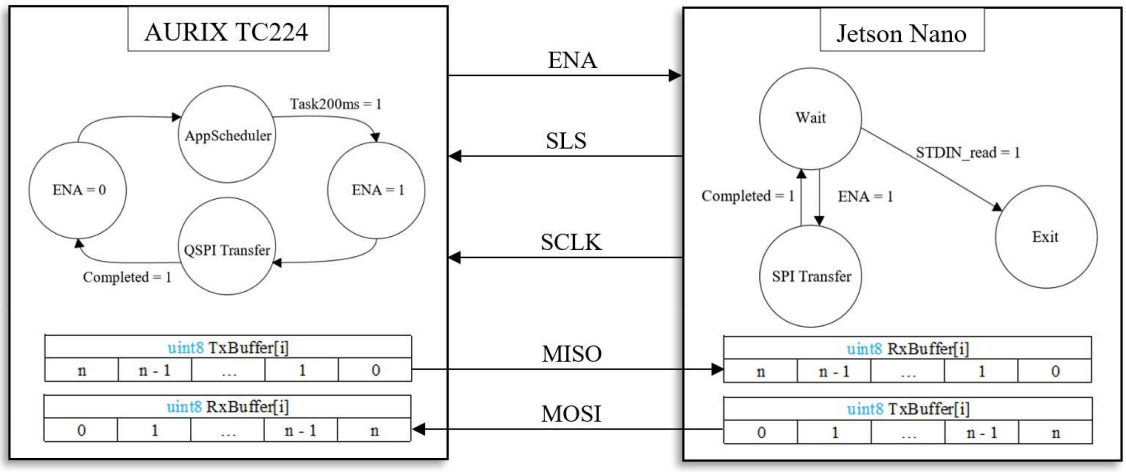
```
60    #include "Ifx_Cfg_FocControl.h"     //for constant "trace_memory = 512"
61  ⊞ ...
64    #define SPI_JETSON_NANO_BUFFER_SIZE (unsigned int)(sizeof(measured_variables))
65  ⊞ ...
73  ⊟ typedef struct __attribute__((packed, aligned(1))) {
74        float32 currenttable[trace_memory];
75        float32 currenttable1[trace_memory];
76        float32 currenttable2[trace_memory];
77        sint16 angletable[trace_memory];
78        float32 velocity;
79    } measured_variables;
80
81  ⊟ typedef struct
82    {
83        measured_variables spi2TxBuffer;
84        uint8 spi2RxBuffer[SPI_JETSON_NANO_BUFFER_SIZE];
85    } AppQspiJetsonNanoBuffer;
```

Obr. 6.3 Výňatek z *Zdrojove_kody\...\QspiDmaJetsonNano.h*

Vzorky dat naměřených při řízení motoru jsou průběžně ukládány do proměnných:

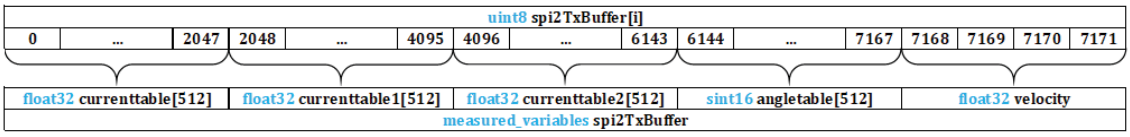
- currenttable – pole 512 prvků typu *float* s hodnotami fázového proudu U
- currenttable1 – pole 512 prvků typu *float* s hodnotami fázového proudu V
- currenttable2 – pole 512 prvků typu *float* s hodnotami fázového proudu W
- angletable – pole 512 prvků typu *short* s hodnotami elektrického úhlu (electrical angle – viz [35])
- velocity – hodnota typu *float* udávající počet otáček motoru za minutu

Obsah těchto proměnných se v rutině *appTaskfu_200ms()* kopíruje do jiných pomocných proměnných, které se v původní verzi aplikace používají pro vykreslování dat na displej (Update TFT – Graph Buffer na Obr. 6.2). Jak již bylo zmíněno, při použití TC224 bylo nutné v aplikaci zobrazování na displej vypnout. Místo toho se v rutině *appTaskfu_200ms()* kopírují měřená data do bufferu pro QSPI (*spi2TxBuffer*), který se v téže rutině odešle do platformy Jetson Nano. Názorněji je tato komunikace zobrazena na (Obr. 6.4).



Obr. 6.4 Blokové schéma SPI komunikace mezi platformami

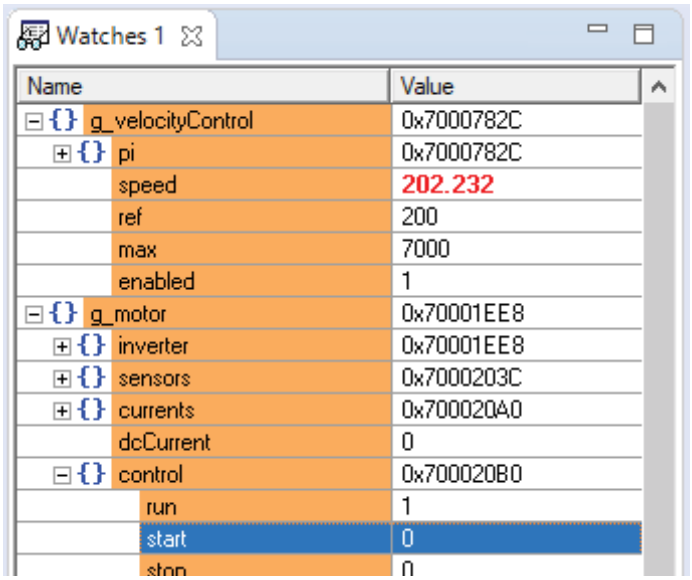
Přijatá nebo odeslaná data přes rozhraní QSPI jsou ve formě pole bajtů (v naší aplikaci se používá délka slova 8 bitů, viz Obr. 5.3). Aby bylo možné jednoduše pracovat s měřenými daty ve stejném tvaru na obou platformách, je vhodné nadefinovat si pro měřená data nový datový typ ve formě struktury (*measured_variables* na Obr. 6.3). Nestačí však definovat standardní strukturu, protože kompilátor by kvůli zefektivnění operací kopírování mohl provést zarovnání struktury v paměti. To je však pro tuto aplikaci nežádoucí, protože by mohlo docházet k přenášení nevýznamných bajtů a komunikace by se zbytečně zpomalovala. Proto se při definici struktury použije klíčové slovo __attribute__, pomocí kterého lze specifikovat speciální atributy struktury. Atribut *packed* zajišťuje, že všechny členy struktury na sebe v paměti navazují, tedy je minimalizována její paměťová náročnost. Atribut *aligned(N)* zajišťuje, že velikost celé struktury bude zarovnána na násobek čísla *N* bajtů. [36] Bylo tedy zvoleno *N = 1* a velikost výsledné struktury poté bude odpovídat součtu velikostí jednotlivých členů struktury. V tomto případě tedy 7172 bajtů = SPI_JETSON_NANO_BUFFER_SIZE (Obr. 6.3). Ukazatel na takto vytvořenou strukturu lze následně po přetypování použít jako ukazatel na pole bajtů ve funkci *IfxQspi_SpiSlave_exchange*, která zajistí odeslání bufferu (viz Obr. 6.5) skrze QSPI.



Obr. 6.5 Struktura měřených dat v SPI zprávách

Jelikož motor nelze ovládat pomocí TFT displeje, bylo nutné zvolit druhou variantu a tou je ovládání skrze UDE debugger. Všechny důležité proměnné a parametry obsahuje struktura *g_motor*, pomocí struktury *g_velocityControl* pak

může být nastavena rychlost otáčení motoru. [35] Proměnné uvnitř těchto struktur je třeba měnit za běhu programu. K tomu slouží nástroj prostředí UDE s názvem Watches, pomocí kterého lze přímo pozorovat a měnit všechny globální a statické proměnné ze zdrojového kódu C/C++. [23]



Obr. 6.6 Ovládání rychlosti motoru pomocí nástroje Watches

Pro roztočení motoru pak stačí spustit vykonávání programu v debuggeru, do proměnné *g_velocityControl.ref* zapsat žádanou hodnotu rychlosti motoru v otáčkách za minutu (rpm) a do proměnné *g_motor.start* zapsat hodnotu 1, čímž se motor spustí. Pokud u struktury *g_velocityControl* nastavíme obnovovací periodu (Refresh Period), lze v reálném čase sledovat aktuální rychlost motoru v proměnné *g_velocityControl.speed*, kterou lze srovnat s hodnotou přijatou na Jetson Nano, a ověřit tak korektní přenos dat (viz Obr. 6.6). [35]

Popis zapojení I/O pinů kitu TC224, které jsou využity v aplikaci *BIFACES_TC224A_eMotor_1v31_HW21_Enc_V3* se nachází v (Tab. 0.1). Z této tabulky se vycházelo při výběru vhodného QSPI modulu. Jako jediná možnost se ukázalo použití modulu QSPI2 (použité piny jsou v tabulce vyznačeny červeně). V původní verzi aplikace je však už modul QSPI2 využíván k obsluze Multi Voltage Safety Micro Processor Supply TLF35584. [35] Tato obsluha se řeší v souborech *QspiTlfDemo.c* a *QspiTlfDemo.h*. Pro TLF35584 na kitu TC224 je však v současné době vytvořen workaround přímo v UDE Target Configuration File, takže v aplikaci není nutné QSPI2 pro obsluhu využívat. [23] Soubory *QspiTlfDemo.c* a *QspiTlfDemo.h* byly tedy přímo nahrazeny vytvořenými soubory *QspiDmaJetsonNano.c* a *QspiDmaJetsonNano.h*, ve kterých se řeší komunikace s Jetson Nano skrze QSPI2. Dále bylo nutné upravit soubory *AppTaskFu.c* a *Cpu0_Main.c.* Všechny tyto soubory se nachází v Příloze 2.

6.2 NVIDIA® Jetson Nano™ Developer Kit

K vyčítání měřených dat z kitu AURIX je používáno rozhraní SPI_1, které je dostupné pomocí adresáře */dev/spidev0.0*. Ve výchozím nastavení je velikost vstupního a výstupního bufferu SPI omezena na 4096 bajtů. Vzhledem k tomu, že kit AURIX vysílá během jednoho přenosu 7172 bajtů, dojde při přenosu k chybě. Je tedy třeba zvětšit velikost SPI bufferu. To lze učinit vytvořením souboru *spidev.conf* v adresáři */etc/modprobe.d/*. Do souboru *spidev.conf* se poté zapíše jeden řádek ve tvaru:

```
options spidev bufsiz=8192
```

Soubor se uloží a následně se Jetson Nano restartuje. Po restartu se velikost SPI bufferu změní na 8192 bajtů. [37]

Změna obslužného programu SPI je minimální. Stačí pouze nadefinovat stejnou strukturu *measured_variables* jako na kitu AURIX a využít funkci *memcpy*, pomocí které se zkopírují bajty v přijatém SPI bufferu na adresu v paměti, kde se nachází struktura *measured_variables*. Poté lze v programu jednoduše pracovat s jednotlivými proměnnými a například jejich obsah vytisknout na terminál, jak je patrné na (Obr. 6.7). Program se spouští z příkazové řádky bez parametru *-v* (zobrazení vyslaných bajtů TX), protože v této aplikaci není nutné vysílat žádná data směrem z Jetson Nano na kit AURIX. Přesněji řečeno - přenos dat probíhá, ale jedná se o samé nulové bajty.

```
student@jetson-nano:~/SPI_program$ sudo ./spi -H -s 17500000
spi mode: 0x1
bits per word: 8
max speed: 17500000 Hz (17500 KHz)

Ignoring first GPIO 79 interrupt
...
poll() GPIO 79 interrupt occurred - rising edge on Slave ready pin

currenttable[0] = 1.000000, currenttable1[0] = 2.000000, currenttable2[0] = 3.000000, angletable[0] = 4, velocity = 5.670000

poll() GPIO 79 interrupt occurred - rising edge on Slave ready pin

currenttable[0] = 3.143311, currenttable1[0] = -2.788544, currenttable2[0] = -0.354767, angletable[0] = 0, velocity = 211.518127
```

Obr. 6.7 Zobrazení měřených dat z řízení motoru na terminálu Jetson Nano

Je vidět, že při prvním SPI přenosu obsahují proměnné inicializační hodnoty, což je první kontrola korektního přenosu dat. Od druhého přenosu již jsou přijímána skutečná měřená data, která lze ověřit srovnáním s hodnotami ve Watches tabulce v debuggeru kitu AURIX. V průběhu řízení motoru lze například jednoduše otestovat, že při jeho zatížení neprodleně vzroste hodnota měřených vzorků fázových proudů do motoru (currenttable), což je v souladu s teoretickým předpokladem.

Měřená data jsou tedy připravena pro další zpracování, čímž je splněn požadavek ze třetího bodu zadání.

7 ZÁVĚR

V první části práce byly teoreticky popsány periferie pro sériovou komunikaci procesoru AURIX a platformy NVIDIA Jetson, které byly uznány za vhodné z hlediska požadavků plynoucích ze zadání. Pro propojení platforem se nakonec ukázaly jako vyhovující 2 z nich: SPI a Ethernet MAC.

Proběhla implementace a testování obou periferií, což zahrnovalo tvorbu obslužných programů na obou platformách. Vzhledem k odlišnosti vývojových prostředí a odlišnému způsobu použití API na platformách bylo k vyřešení zadané úlohy nutné čerpat ze značného množství zdrojů, což je patrné ze seznamu použité literatury. Informace k řešení problému někdy nebylo možné najít v oficiální dokumentaci k platformám a bylo nutné hledat například na internetových fórech, což bylo mnohdy poměrně zdlouhavé. Zvláště v případě platformy Jetson Nano, která je poměrně nová a v průběhu práce docházelo k aktualizacím software a dokumentace. Nakonec se však podařilo rozjet komunikaci přes obě jmenované periferie.

V případě SPI funguje obousměrný přenos dat mezi TC224 a Jetson Nano bez problému do rychlosti 10 Mbit/s s využitím CPU, s využitím DMA lze poté dosáhnout rychlosti až 17,5 Mbit/s. Jedná se o nižší maximální rychlost, než jakou uvádějí výrobci (viz Tab. 3.2) a při dalším vývoji projektu by zřejmě bylo vhodné zabývat se příčinami, které způsobují omezení přenosové rychlosti.

Komunikace přes Ethernet MAC byla rovněž otestována v obou směrech. Teoretické rychlosti 100 Mbit/s ovšem v této implementaci nelze zcela dosáhnout, protože se nepodařilo omezit vysílání nepotřebných rámců s multicastovými a broadcastovými adresami. Tento problém zůstává rovněž námětem pro další vývoj. Přesto je rychlost komunikace oproti SPI výrazně vyšší. Další výhodou je například skutečnost, že použitím Ethernetu se nesnižuje počet využitelných I/O pinů na platformách.

Jak se však ukázalo při řešení třetího bodu zadání, Ethernet nebylo možno použít. Avšak pro objem měřených dat vysílaných na Jetson Nano, které se generují při řízení elektrického motoru procesorem AURIX, v této aplikaci postačila rychlost SPI. Zde byla využita modifikace SPI komunikace použitím 5 vodičů místo běžných 4 (přidání signálu ENA), čímž se komunikace zjednodušila, například nebylo nutné vysílat dotazy skrze SPI pro aktivní kontrolu stavu (polling) ze strany Jetson Nano, případně řešit synchronizaci periody vysílání dat na obou platformách.

Literatura

[1] *Jetson Nano Developer Kit: User Guide* [online]. V1.1. Santa Clara, Kalifornie, USA: NVIDIA Corporation, 2019 [cit. 2019-12-28]. Dostupné z: https://developer.download.nvidia.com/embedded/L4T/r32-2_Release_v1.0/Jetson_Nano_Developer_Kit_User_Guide.pdf?kOcAKuyHJzHA0T45TQDE8C2msG4mvfLo31GkizYhDCyJvnmM9tuair8GPP-acZhbom5hq03uQW88x1xvPKh6wH5AYdBOMrrTWKF__8l223dlL-sL1OV3XjMQMmQd9uWHI0B9x55gluQNmnA4RHeFmc5RramN8eeV7-rKB4QzRGFThg8vj_O6

[2] *NVIDIA Jetson Nano: PRODUCT DESIGN GUIDE* [online]. 20190607| PRELIMINARY –SUBJECT TO CHANGE. Santa Clara, Kalifornie, USA: NVIDIA Corporation, 2019 [cit. 2019-12-18]. Dostupné z: https://developer.nvidia.com/embedded/downloads#?search=Jetson%20Nano

[3] *Jetson Nano Datasheet: NVIDIA Jetson Nano System-on-Module* [online]. V0.8. Santa Clara, Kalifornie, USA: NVIDIA Corporation, 2019 [cit. 2019-12-18]. Dostupné z: https://developer.nvidia.com/embedded/downloads#?search=Jetson%20Nano

[4] 32-bit AURIX™ Microcontroller based on TriCore™. *Semiconductor & System Solutions - Infineon Technologies* [online]. 81726 Munich, Germany: Infineon Technologies AG, 2020 [cit. 2020-05-22]. Dostupné z: https://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-microcontroller/

[5] *Application Kit TC2X7: Hardware Manual: 32-Bit Microcontroller* [online]. V1.0 2015-04. 81726 Munich, Germany: Infineon Technologies AG, 2015 [cit. 2019-12-29]. Dostupné z: https://www.infineon.com/dgdl/Infineon-TC2x7_ApplicationKitManual-UM-v01_00-EN.pdf?fileId=5546d46269bda8df0169ca24e3d124cd

[6] Komunikace pro sběrnici. *Univerzitní informační systém MENDELU* [online]. Brno: Ústav pro informační systém MENDELU [cit. 2019-12-28]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=9947

[7] *TriCore™ A U R I X™ Family 32-bit: Aurix Unleashed: Getting Started With Aurix* [online]. V1.0. University Of Warwick Science Park, Coventry, CV4 7HS, UK: Hitex (U.K.) Limited., 2015 [cit. 2019-12-28]. Dostupné z: https://hitex.co.uk/fileadmin/uk-files/downloads/ShieldBuddy/4274.AurixUnleashed_Htx_v1.0.pdf

[8] *AURIX TC27x-D: User's Manual: 32-Bit Single-Chip Microcontroller* [online]. V2.2 2014-12. 81726 Munich, Germany: Infineon Technologies AG, 2014 [cit. 2019-12-17]. Dostupné z: https://hitex.co.uk/fileadmin/uk-files/downloads/ShieldBuddy/tc27xD_um_v2.2.pdf

[9] *UM10204: I2C-bus specification and user manual* [online]. Rev. 6. Eindhoven, Nizozemsko: NXP Semiconductors N.V., 2014 [cit. 2019-12-17]. Dostupné z: https://www.nxp.com/docs/en/user-guide/UM10204.pdf

[10] *TC27x DB-Step Data Sheet* [online]. V 1.2. 81726 Munich, Germany: Infineon Technologies AG, 2019 [cit. 2019-12-18]. Dostupné z: AURIX Customer Documentation site (po registraci a získání přístupu na https://www.infineon.com/)

[11] *ILLD User Documentation: Device: TC27D* [online]. Version: 1.0.1.11.0. 81726 Munich, Germany: Infineon Technologies AG, 2019 [cit. 2019-12-18]. Dostupné z: AURIX Customer Documentation site (po registraci a získání přístupu na https://www.infineon.com/)

[12] Amaork/libi2c: Linux i2c library, support C/C++/Python. *GitHub* [online]. San Francisco, Kalifornie, USA: GitHub, Inc., 2019 [cit. 2019-12-18]. Dostupné z: https://github.com/amaork/libi2c

[13] Serial Peripheral Interface: SPI timing diagram2. *Wikipedia, the free encyclopedia* [online]. Colin M.L. Burnett, 2010 [cit. 2019-12-28]. Dostupné z: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface#/media/File:SPI_timing_diagram2.svg

[14] *TMS320C672x DSP Serial Peripheral Interface* [online]. LiteratureNumber: SPRU718B. Dallas, Texas, USA: Texas Instruments Incorporated, 2007 [cit. 2019-12-28]. Dostupné z: https://www.ti.com/lit/ug/spru718b/spru718b.pdf

[15] Rm-hull/spidev-test. *GitHub* [online]. UK: Richard Hull, 2017 [cit. 2019-12-28]. Dostupné z: https://github.com/rm-hull/spidev-test

[16] *AURIX TC21x/TC22x/TC23x Family: User's Manual: 32-Bit Single-Chip Microcontroller* [online]. V1.1 2014-12. 81726 Munich, Germany: Infineon Technologies AG, 2014 [cit. 2019-12-28]. Dostupné z: AURIX Customer Documentation site (po registraci a získání přístupu na https://www.infineon.com/)

[17] *ILLD User Documentation: Device: TC22A* [online]. Version: 1.0.1.11.0. 81726 Munich, Germany: Infineon Technologies AG, 2019 [cit. 2019-12-28]. Dostupné z: AURIX Customer Documentation site (po registraci a získání přístupu na https://www.infineon.com/)

[18] A Beginner's Guide to Ethernet 802.3: Application Note (EE-269). *Analog Devices* [online]. Norwood, Massachusetts, USA: Analog Devices, Inc., 2005 [cit. 2020-04-19]. Dostupné z: https://www.analog.com/media/en/technical-documentation/application-notes/EE-269.pdf

[19] Getting Started With Jetson Nano Developer Kit | NVIDIA Developer. *NVIDIA Developer* [online]. Santa Clara, Kalifornie, USA: NVIDIA Corporation, 2019 [cit. 2019-12-30]. Dostupné z: https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#prepare

[20] Jetson Nano - Visual Studio Code + Python. *JetsonHacks* [online]. Pasadena, CA 91101, USA: Copyright @ JetsonHacks 2014-2019, 2019 [cit. 2019-12-30]. Dostupné z: https://www.jetsonhacks.com/2019/10/01/jetson-nano-visual-studio-code-python/

[21] Getting started with the Jetson Nano. *Medium* [online]. Hamburg, Německo: Heldenkombinat Technologies, 2019 [cit. 2020-05-19]. Dostupné z: https://medium.com/@heldenkombinat/getting-started-with-the-jetson-nano-37af65a07aab#f208

[22] *Free TriCore™ Entry Tool Chain - Activation & Download* [online]. 66113 Saarbruecken, Německo: HighTec EDV-Systeme GmbH, 2019 [cit. 2019-12-30]. Dostupné z: https://free-entry-toolchain.hightec-rt.com/

[23] *A Software Guide to Universal Debug Engine* [online]. V 5.00.02.02. Lauta, Germany: PLS Programmierbare Logik & Systeme GmbH, 2019 [cit. 2020-05-21]. Dostupné z: https://free-entry-toolchain.hightec-rt.com/

[24] *Infineon BIFACES: Migrate iLLD demos to managed HighTec projects* [online]. Version 2.4. 66113 Saarbruecken, Německo: HighTec EDV-Systeme GmbH, 2019 [cit. 2019-12-30]. Dostupné z: https://free-entry-toolchain.hightec-rt.com/HighTec_Free_TriCore_Entry_Toolchain_Migration_Guide.pdf

[25] *Jetson Nano Developer Kit 40-Pin Expansion Header Configuration: Application Note* [online]. V1.2. Santa Clara, Kalifornie, USA: NVIDIA Corporation, 2019 [cit. 2019-12-31]. Dostupné z: https://developer.nvidia.com/embedded/downloads#?search=Jetson%20Nano

[26] Updated instructions for SPI on Nano DevelopmentKit with L4T 32.2.1. In: *NVIDIA Developer Forums* [online]. <g.devel@wxy78.net>: George Joseph, 2020 [cit. 2019-12-31]. Dostupné z: https://devtalk.nvidia.com/default/topic/1062646/jetson-nano/updated-instructions-for-spi-on-nano-developmentkit-with-l4t-32-2-1/

[27] Rt-net/JetsonNano_DT_SPI at R32.2.1. *GitHub* [online]. Tokyo,101-0021, Japonsko: RT Corporation, 2019 [cit. 2020-01-01]. Dostupné z: https://github.com/rt-net/JetsonNano_DT_SPI/tree/R32.2.1

[28] Gpio-int-test.c. *RidgeRun Embedded Linux Developer Connection* [online]. Princeton, MN 55371, USA: RidgeRun, 2011 [cit. 2020-01-01]. Dostupné z: https://developer.ridgerun.com/wiki/index.php/Gpio-int-test.c

[29] EtherType. *Wikipedia, the free encyclopedia* [online]. last edited on 6 May 2020, at 13:23 (UTC) [cit. 2020-05-14]. Dostupné z: https://en.wikipedia.org/wiki/EtherType

[30] Jetson Download Center: Jetson Nano Developer Kit Ethernet Firmware Update: 2598410. *NVIDIA Developer* [online]. Santa Clara, Kalifornie, USA: NVIDIA Corporation, 2019 [cit. 2020-01-02]. Dostupné z: https://developer.nvidia.com/embedded/dlc/jetson-nano-developer-kit-ethernet-firmware-2598410

[31] 802.11ax Remote Packet Captures using the Jetson Nano. *SemFio Networks - Wireless Consulting Services in Canada* [online]. Londýn, Ontario, Kanada: SemFio Networks Inc., 2019 [cit. 2020-01-02]. Dostupné z: https://www.semfionetworks.com/blog/80211ax-remote-packet-captures-using-the-jetson-nano

[32] Send / Receive raw Ethernet frames with custom EtherType in Linux. *GitHub* [online]. Seoul: Sunjin Yang, 2015 [cit. 2020-04-20]. Dostupné z: https://gist.github.com/lethean/5fb0f493a1968939f2f7

[33] Socket API. *Jan Outrata – Domovská stránka* [online]. Olomouc: Přírodovědecká fakulta Univerzita Palackého v Olomouci, Katedra informatiky, 2014 [cit. 2020-04-20]. Dostupné z: http://outrata.inf.upol.cz/courses/pos/sockets.html

[34] MLD, Multicast Listener Discovery. *Network Sorcery, Inc.* [online]. San Diego, USA, 2012 [cit. 2020-05-19]. Dostupné z: http://www.networksorcery.com/enp/protocol/mld.htm

[35] *TriCore™ AURIX™ Family 32-Bit: eMotor application example (PMSM with FOC): AP32298 Application Note* [online]. V1.0 2015-06-22. 81726 Munich, Germany: Infineon Technologies AG, 2015 [cit. 2020-05-21]. Dostupné z: AURIX Customer Documentation site (po registraci a získání přístupu na https://www.infineon.com/)

[36] Type Attributes: Using the GNU Compiler Collection. *GCC, the GNU Compiler Collection* [online]. Free Software Foundation, Inc., 2005 [cit. 2020-05-21]. Dostupné z: https://gcc.gnu.org/onlinedocs/gcc-4.0.2/gcc/Type-Attributes.html

[37] Duet 3 and jetson Nano?. In: *Duet3D* [online]. gtj0: George Joseph [cit. 2020-05-22]. Dostupné z: https://forum.duet3d.com/topic/11777/duet-3-and-jetson-nano/62?_=1590151878286

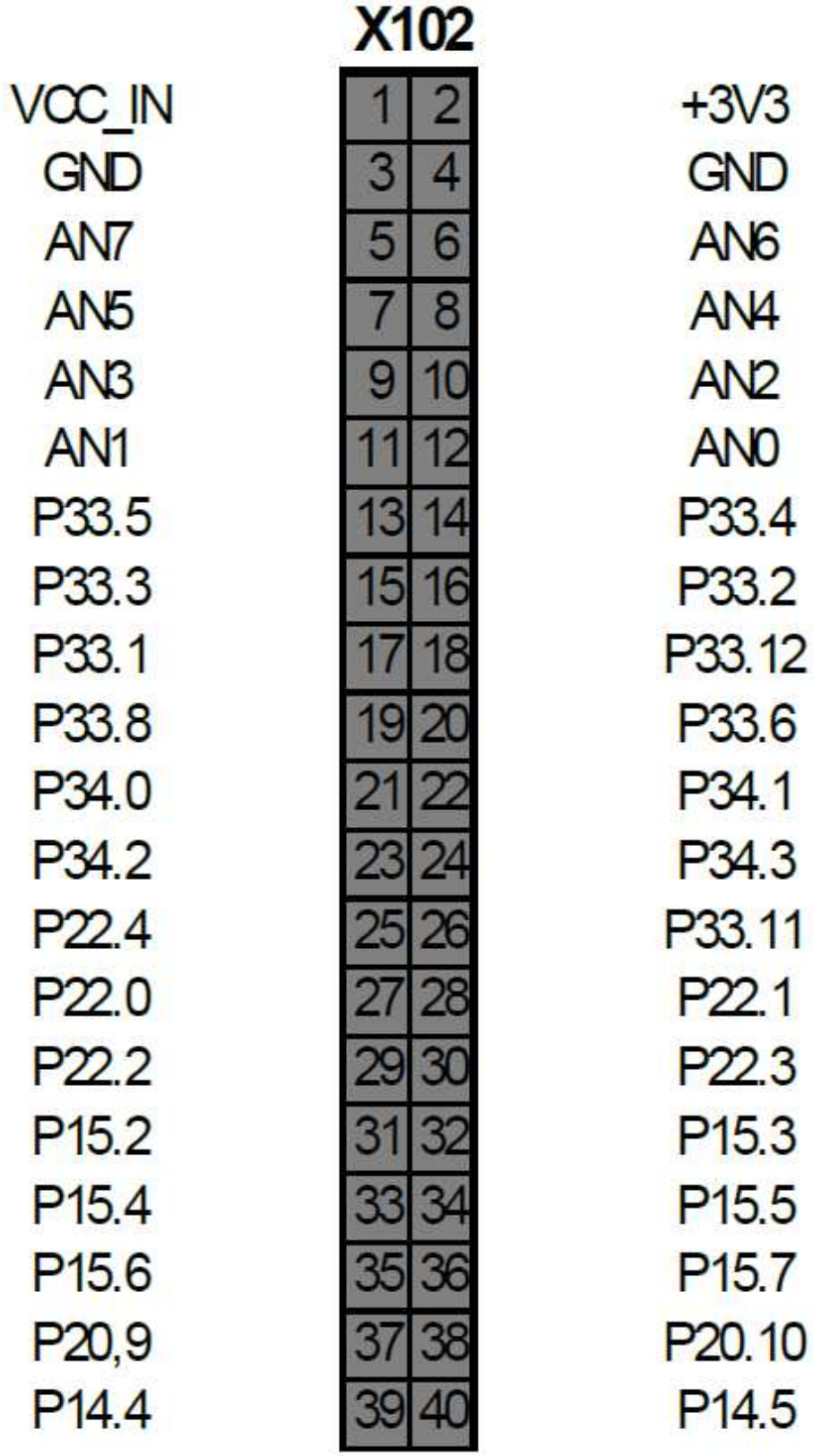
[38] *Application Kit TC2X4: Hardware Manual: 32-Bit Microcontroller* [online]. V1.0 2014-10. 81726 Munich, Germany: Infineon Technologies AG, 2014 [cit. 2019-12-30]. Dostupné z: https://www.infineon.com/dgdl/Infineon-KIT_AURIX_TC224_TFT-UserManual-v01_00-EN.pdf?fileId=5546d4626c1f3dc3016c85c5843a7f4c

[39] NVIDIA Jetson Nano J41 Header Pinout. *JetsonHacks* [online]. Pasadena, CA 91101, USA: Copyright @ JetsonHacks 2014-2019, 2019 [cit. 2020-01-01]. Dostupné z: https://www.jetsonhacks.com/nvidia-jetson-nano-j41-header-pinout/

Seznam příloh

Příloha 1 – Doplňkové obrázky a tabulky většího rozměru
Příloha 2 – Soubory se zdrojovými kódy programů jsou uloženy na přiloženém CD

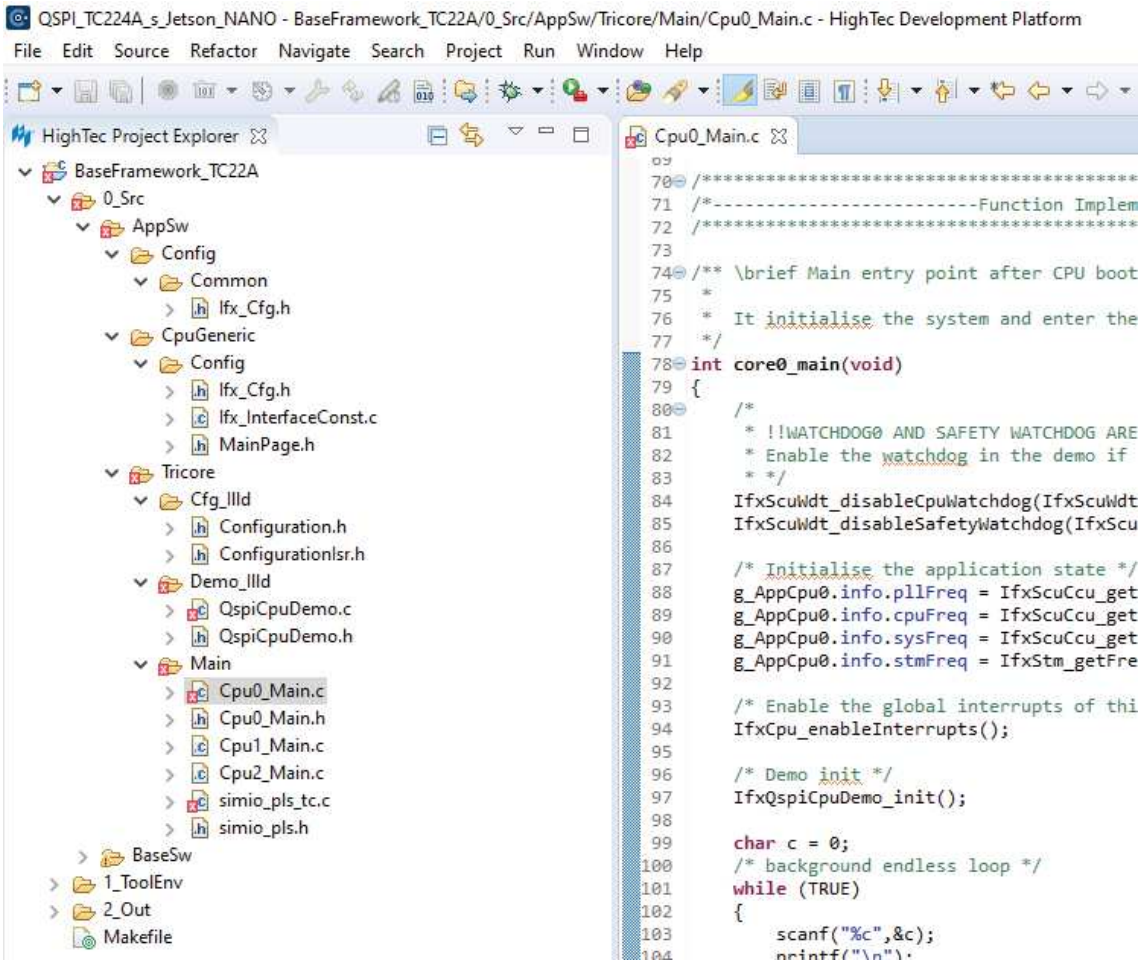
Příloha 1 – Doplňkové obrázky a tabulky



Obr. 0.1 Rozložení pinů X102 headeru na TC224 TFT kitu [38]

Jetson Nano J41 Header					
Sysfs GPIO	Name	Pin	Pin	Name	Sysfs GPIO
	3.3 VDC *Power*	1	2	**5.0 VDC** *Power*	
	I2C_2_SDA *I2C Bus 1*	3	4	**5.0 VDC** *Power*	
	I2C_2_SCL *I2C Bus 1*	5	6	**GND**	
gpio216	**AUDIO_MCLK**	7	8	**UART_2_TX** */dev/ttyTHS1*	
	GND	9	10	**UART_2_RX** */dev/ttyTHS1*	
gpio50	**UART_2_RTS**	11	12	**I2S_4_SCLK**	gpio79
gpio14	**SPI_2_SCK**	13	14	**GND**	
gpio194	**LCD_TE**	15	16	**SPI_2_CS1**	gpio232
	3.3 VDC *Power*	17	18	**SPI_2_CS0**	gpio15
gpio16	**SPI_1_MOSI**	19	20	**GND**	
gpio17	**SPI_1_MISO**	21	22	**SPI_2_MISO**	gpio13
gpio18	**SPI_1_SCK**	23	24	**SPI_1_CS0**	gpio19
	GND	25	26	**SPI_1_CS1**	gpio20
	I2C_1_SDA *I2C Bus 0*	27	28	**I2C_1_SCL** *I2C Bus 0*	
gpio149	**CAM_AF_EN**	29	30	**GND**	
gpio200	**GPIO_PZ0**	31	32	**LCD_BL_PWM**	gpio168
gpio38	**GPIO_PE6**	33	34	**GND**	
gpio76	**I2S_4_LRCK**	35	36	**UART_2_CTS**	gpio51
gpio12	**SPI_2_MOSI**	37	38	**I2S_4_SDIN**	gpio77
	GND	39	40	**I2S_4_SDOUT**	gpio78

Obr. 0.2 Rozložení pinů J41 headeru na Jetson Nano kitu [39]



Obr. 0.3 Adresářová struktura souborů pro obslužný program QSPI

Tab. 0.1 Zapojení pinů headeru X102 na TC224 při řízení motoru

TC224 Application Kit		eMotor Drive		Connector X102/BU6		eMotor Drive		TC224 Application Kit	
Name	***Description***	**Name**	***Description***			**Name**	***Description***	**Name**	***Description***
VCC_IN		**VCC_IN**	*Supply Input TriBoard (5,5 V … 50 V)*	**1**	**2**	**VEXTA**	*n.c.*	**+3V3**	
GND	*QSPI2 GND*	**GND**	*Ground*	**3**	**4**	**GND**	*Ground*	**GND**	
AN7	*VADC G0CH7*	**CH2NA (SIN. Prim. Coil)**	*Analog input / Resolver VCC/2*	**5**	**6**	**CH2PA (SIN. Prim. Coil)**	*Analog input / Resolver SIN*	**AN6**	*VADC G0CH6*
AN5	*VADC G0CH5*	**n.c.**		**7**	**8**	**n.c.**		**AN4**	*VADC G0CH4*
AN3	*VADC G0CH3*	**CH0NA (COS. Prim. Coil)**	*Analog input / Resolver VCC/2*	**9**	**10**	**CH0PA (COS. Prim. Coil)**	*Analog input / Resolver COS*	**AN2**	*VADC G0CH2*
AN1	*VADC G0CH1*	**VRO**	*TLE9180 Output of reference voltage of differential amplifier*	**11**	**12**	**VO1**	*TLE9180 Output of differential 1 amplifier for shunt signal amplification*	**AN0**	*VADC G0CH0*
P33.5		**PFB1**	*TLE9180 Phase feedback of motor connection phase 1*	**13**	**14**	**n.c.**		**P33.4**	
P33.3		**n.c.**		**15**	**16**	**n.c.**		**P33.2**	
P33.1		**PFB1_Enable**	*Enable PFB1*	**17**	**18**	**PFB3**	*TLE9180 Phase feedback of motor connection phase 3*	**P33.12**	
P33.8		**n.c.**		**19**	**20**	**PFB2**	*TLE9180 Phase feedback of motor connection phase 2*	**P33.6**	
P34.0		**n.c.**		**21**	**22**	**n.c.**		**P34.1**	
P34.2		**n.c.**		**23**	**24**	**n.c.**		**P34.3**	
P22.4		**n.c.**		**25**	**26**	**n.c.**		**P33.11**	
P22.0	*QSPI3 MTSR*	**MOSI**	*TLE9180 SPI Master Out, Slave In*	**27**	**28**	**MISO**	*TLE9180 SPI Master In, Slave Out*	**P22.1**	*QSPI3 MRST*
P22.2	*QSPI3 SLSO12*	**CSN**	*TLE9180 Chip Select*	**29**	**30**	**CLK_SPI**	*TLE9180 SPI clock input*	**P22.3**	*QSPI3 SCLK*
P15.2	*QSPI2 SLS*	**n.c.**		**31**	**32**	**n.c.**		**P15.3**	*QSPI2 SCLK*
P15.4	*QSPI2 MRST*	**/ERR**	*TLE9180 Error signal*	**33**	**34**	**n.c.**		**P15.5**	*QSPI2 MTSR*
P15.6		**n.c.**		**35**	**36**	**n.c.**		**P15.7**	
P20.9		**/SOFF**	*TLE9180 Independent safe state switch off*	**37**	**38**	**ENA**	*TLE9180 Enable pin*	**P20.10**	
P14.4	*QSPI2 ENA*	**n.c.**		**39**	**40**	**n.c.**		**P14.5**	

Tab. 0.2 Zapojení pinů headeru X103 na TC224 při řízení motoru

TC224 Application Kit		eMotor Drive		Connector X103/BU7		eMotor Drive		TC224 Application Kit	
Name	***Description***	**Name**	***Description***			**Name**	***Description***	**Name**	***Description***
VCC_IN		**VCC_IN**	*Supply Input TriBoard (5,5 V … 50 V)*	**1**	**2**	**VEXTB**	*n.c.*	**+3V3**	
GND		**GND**	*Ground*	**3**	**4**	**GND**	*Ground*	**GND**	
P33.10		**n.c.**		**5**	**6**	**n.c.**		**P33.9**	
P14.8		**n.c.**		**7**	**8**	**n.c.**		**P14.7**	
P14.6		**n.c.**		**9**	**10**	**/INH**	*TLE9180 Inhibit pin*	**P10.6**	
P20.2		**n.c.**		**11**	**12**	**n.c.**		**P11.8**	
P02.0		**CGPWM_N (Prim. Coil)**	*Primary Coil*	**13**	**14**	**CGPWM_P (Prim. Coil)**	*Primary Coil*	**P02.1**	
P02.2		**n.c.**		**15**	**16**	**HALLA**	*HALL A*	**P02.3**	
P02.4		**HALLB**	*HALL B*	**17**	**18**	**HALLC**	*HALL C*	**P02.5**	
P02.6	*GPT12 T3INA*	**ENC_A**	*Encoder A*	**19**	**20**	**ENC_B**	*Encoder B*	**P02.7**	*GPT12 T3EUDA*
P02.8	*GPT12 T4INA*	**ENC_Z**	*Encoder Top Zero*	**21**	**22**	**n.c.**		**P00.0**	
P00.1		**n.c.**		**23**	**24**	**IL1**	*TLE9180 PWM Low-side switch 1*	**P00.2**	*GTM TOM1_1*
P00.3	*GTM TOM1_2*	**/IH1**	*TLE9180 PWM High-side switch 1*	**25**	**26**	**IL2**	*TLE9180 PWM Low-side switch 2*	**P00.4**	*GTM TOM1_3*
P00.5	*GTM TOM1_4*	**/IH2**	*TLE9180 PWM High-side switch 2*	**27**	**28**	**IL3**	*TLE9180 PWM Low-side switch 3*	**P00.6**	*GTM TOM1_5*
P00.7	*GTM TOM1_6*	**/IH3**	*TLE9180 PWM High-side switch 3*	**29**	**30**	**n.c.**		**P00.8**	
P00.9		**n.c.**		**31**	**32**	**n.c.**		**P21.4**	
P21.5		**n.c.**		**33**	**34**	**n.c.**		**P00.12**	
AN17	*VADC G1CH5*	**n.c.**		**35**	**36**	**n.c.**		**AN16**	*VADC G1CH4*
AN15	*VADC G1CH3*	**n.c.**		**37**	**38**	**VO3**	*TLE9180 Output of differential 3 amplifier for shunt signal amplification*	**AN13**	*VADC G1CH1*
AN14	*VADC G1CH2*	**n.c.**		**39**	**40**	**VO2**	*TLE9180 Output of differential 2 amplifier for shunt signal amplification*	**AN12**	*VADC G1CH0*