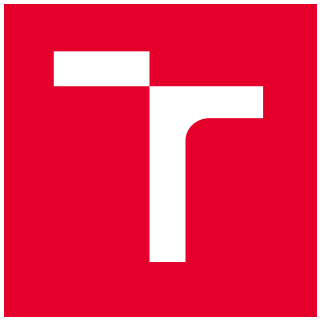


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

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

INOVACE LABORATORNÍCH ÚLOH KURZU VESTAVNÉ SYSTÉMY

INNOVATION OF THE LABORATORY EXERCISES FOR COURSE EMBEDDED SYSTEMS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. Matej Pončák

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Petr Petyovský, Ph.D.

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Matej Pončák ***ID:*** 203325

Ročník: 2 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Inovace laboratorních úloh kurzu Vestavné systémy

POKYNY PRO VYPRACOVÁNÍ:

1. Prostudujte dokumentaci a HW možnosti vývojového kitu Raspberry Pi Pico. Nastudujte vlastnosti vývojových nástrojů a knihoven pro implementaci programů v jazyce C/C++ na tomto vývojovém kitu.
2. Nastudujte zadání současných laboratorních úloh kurzu Vestavné systémy. Zvolte alespoň 4 úlohy, které by bylo vhodné zmodernizovat a přenést na vývojový kit Raspberry Pi Pico.
3. Navrhněte koncept HW platformy, která umožní modulární připojení vývojového kitu Raspberry Pi Pico a bude obsahovat vhodné komponenty pro výukové úlohy, tj.: ovládací a vizualizační prvky, komponenty na základních embedded sběrnicích (např. I2C, SPI, UART) a další konektory (např. VGA, zvukový výstup).
4. Vytvořte a otestujte koncept této HW platformy pro alespoň 4 zvolené úlohy kurzu.
5. Zrealizujte a osaďte finální revizi HW platformy. Vytvořte zadávací a výrobní dokumentaci za účelem jednoduché a opakované výroby HW platformy pro potřeby kurzu.
6. Implementujte sadu alespoň 4 laboratorních úloh pro vybrané periferie HW platformy. Ke každé laboratorní úloze vypracujte dokumentaci pro studenty a vzorové řešení úlohy včetně její dokumentace pro vyučující.
7. Zapracujte připomínky a návrhy vyučujících do konečné verze výukových úloh.
8. Zhodnoťte dosažené výsledky, uveďte výhody a nevýhody řešení a navrhněte další možná rozšíření.

DOPORUČENÁ LITERATURA:

[1] VIRIUS, Miroslav. Jazyky C a C++: kompletní průvodce. 2., aktualiz. vyd. Praha: Grada, 2011. ISBN 9788024739175.

[2] Raspberry Pi Pico: Technical Specification. Raspberry Pi Foundation. [online]. Feb 2. 2022. Dostupné na WWW: <https://www.raspberrypi.org/documentation/microcontrollers/raspberry-pi-pico.html>.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 18.5.2022

Vedoucí práce: Ing. Petr Petyovský, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

ABSTRAKT

Neustále napredovanie trhu v oblasti embedded systémov si vyžaduje prispôsobovanie sa tejto skutočnosti. Spoločnosť Raspberry Pi vydala vývojovú dosku Raspberry Pi Pico s 32-bitovým mikrokontrolérom s architektúrou ARM Cortex-M0+, čo predstavuje potenciál pre využitie dosky ako inováciu vo výuke embedded systémov. Práca popisuje túto dosku a jej vlastnosti, predmet Vstavané systémy a mikroprocesory a možnosti jeho inovácie a následne popisuje narhnutú HW platformu a demonštruje jej využitie. Súčasťou práce je aj vytvorenie nových zadaní laboratórnych úloh.

KĽÚČOVÉ SLOVÁ

Raspberry Pi Pico, RP2040, embedded systémy, inovácia, výukový kit RPi Pico Kit, laboratórne úlohy

ABSTRACT

The constant market development of the embedded systems requires an adaptation to this fact. Raspberry Pi has released the Raspberry Pi Pico development board with a 32-bit microcontroller ARM Cortex-M0+, which has the potential to use the board as an innovation in teaching embedded systems. The thesis describes this board and its properties, the course Embedded Systems and the possibilities of its innovation, and then describes the proposed HW platform and demonstrates. The creation of new laboratory assignments is also part of this thesis.

KEYWORDS

Raspberry Pi Pico, RP2040, embedded systems, inovation, education kit RPi Pico Kit, laboratory exercises

Vysadené pomocou balíčku `thesis` verzie 4.07; `http://latex.feec.vutbr.cz`

PONČÁK, Matej. *Inovace laboratorních úloh kurzu Vestavné systémy*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2022, 122 s. Diplomová práca. Vedúci práce: Ing. Petr Petyovský, Ph.D.

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Bc. Matej Pončák
VUT ID autora:	203325
Typ práce:	Diplomová práca
Akademický rok:	2021/22
Téma závěrečnej práce:	Inovace laboratorních úloh kurzu Vestavné systémy

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno 18.05.2022 .....

podpis autora

POĎAKOVANIE

Rád by som sa poďakoval vedúcemu diplomovej práce pánovi Ing. Petrovi Petyovskému, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podporu pri tvorbe práce.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov

Úvod

V posledných rokoch sa mikrokontroléry dostávajú čím ďalej, tým viac do bežného života človeka a vytláčajú tak doteraz bežne používané logické obvody, ktorých použitie je v dnešnej dobe omnoho zložitejšie a neraz aj drahšie ako použitie mikrokontroléra, ktorý so sebou prináša značné množstvo flexibility. Nájsť ich môžeme v nejednom zariadení v každej domácnosti, ako napríklad v rádiu, televízore, hračkách, mobilných telefónoch a tiež v čoraz viac používaných zariadeniach pre internet vecí. Rôznorodosť čipov na trhu umožňuje vybrať taký, ktorý najviac vyhovuje danej aplikácii.

Na tento trend sa snaží reagovať spoločnosť Raspberry Pi, ktorá je známa výrobou rovnomenných jednodoskových počítačov. Začiatkom roka 2021 spoločnosť predstavila svoj prvý mikrokontrolér RP2040 spolu s vývojovou doskou Raspberry Pi Pico. Jeho 32-bitová architektúra umožňuje využitie v najjednoduchších aplikáciách až po strojové učenie či obrazový výstup na počítačový monitor.

Na VUT FEKT v Brne v bakalárskom študijnom programe Automatizačná a meracia technika sa vyučuje predmet Vstavané systémy a mikroprocesory, ktorého náplňou je oboznámiť sa so základnými princípmi mikroprocesorových a embedded systémov. Na laboratórnych cvičeniach tohto predmetu sa pracuje s 8-bitovým mikrokontrolérom z rodiny HCS08 od firmy NXP. Aby boli študenti po absolvovaní tohto programu, čo najviac zoznámení s komponentami dostupnými na trhu, ktorý napreduje veľmi veľkým tempom, predmet si vyžaduje inováciu učebných pomôcok, čím vzniká potenciál pre využitie vývojovej dosky Raspberry Pi Pico.

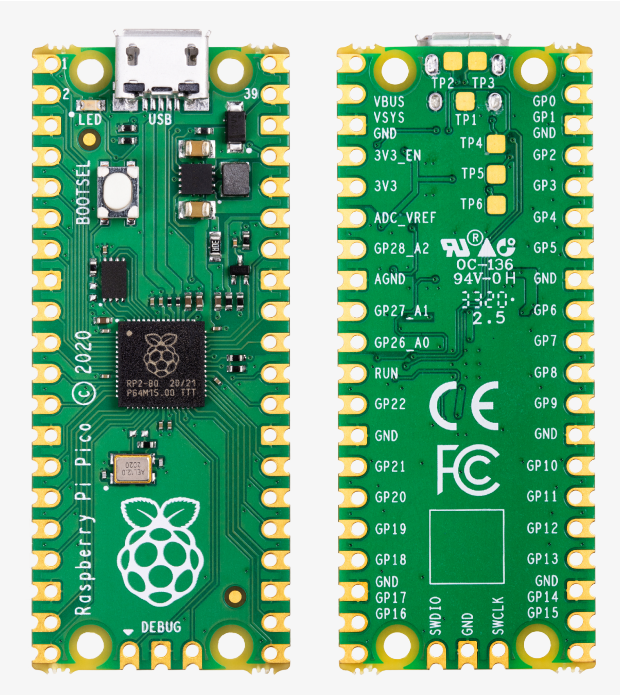
Na začiatok sa čitateľ oboznámi s vývojovou doskou, s mikrokontrolérom, ktorý je na nej osadený, a spôsobom využitia dostupných vývojových prostriedkov. Ďalej bude vysvetlená náplň predmetu Vstavané systémy a mikroprocesory a bude navrhnutá sada úloh pre laboratórne cvičenia tak, aby svojím edukatívnym charakterom previedli študenta základnými princípmi embedded systémov a vysvetlili prácu s vývojovou doskou.

Aby študenti predmetu mohli vývojovú dosku využívať spolu s externými periférіami, je potrebné navrhnúť koncept HW platformy, ktorý to umožní. V ďalšej časti práce je navrhnutý koncept takejto platformy a ďalej je otestovaný na sade vytvorených testovacích programoch. Výsledky z testovania sú zapracované do návrhu HW platformy, ktorá je následne zrealizovaná. V práci sú tiež navrhnuté zadania laboratórnych úloh pre študentov ako aj ich vzorové riešenia vrátane dokumentácie pre vyučujúcich predmetu.

V závere je diskutovaný prínos práce a tiež jej nedostatky, či možné vylepšenia.

1 Vývojová doska Raspberry Pi Pico

Vývojová doska Raspberry Pi Pico (obr. 1.1) je debutová doska v oblasti mikrokontrolérov od nadácie Raspberry Pi, ktorá bola uvedená na trh 21. januára 2021. Táto nadácia od svojho vzniku v roku 2009 podporuje štúdium základov počítačovej vedy na školách a je zodpovedná za vývoj jednodoskových počítačov Raspberry Pi. Počítač Raspberry Pi však nemá možnosť analógového vstupu bez prídavných obvodov a softvér bežiaci pod operačným systémom ako je napríklad Linux, nie je vhodný na riadenie vstupno-výstupných pinov (GPIO) s nízkou latenciou. To spolu s inými dôvodmi, popísanými nižšie viedlo k vytvoreniu dosky Raspberry Pi Pico. Produkty tejto nadácie sa vyznačujú nízkou cenou a vďaka tomu sú ľahko dostupné pre študentov. Cena vývojovej dosky Raspberry Pi Pico je 4 doláre [19].



Obr. 1.1: Vývojová doska Raspberry Pi Pico [9]

Jadrom dosky je ich vlastný 32-bitový mikrokontrolér RP2040, ktorý si sami ako spoločnosť navrhli. Čip je prekvapivo výkonný, ale zároveň veľmi lacný s dvojjadrovým procesorom architektúry ARM Cortex-M0+, čo je energeticky najefektívnejší ARM procesor. To zaručuje maximálny výkon pri minimálnej spotrebe energie. Pri nečinnosti je jeho príkon menej ako 1 mW, čo umožňuje dlhodobú prevádzku za použitia batérie s malou kapacitou. Pre porovnanie, najmenší počítač Raspberry Pi Zero v režime s najnižšou spotrebou energie spotrebuje rádovo 100 mW.

Hlavnými prednosťami dosky Raspberry Pi Pico je veľká vstavaná pamäť, deterministická zbernica a relatívne veľké množstvo periférií s unikátnym programovateľným vstupno-výstupným rozhraním PIO, ktoré si spoločnosť dala patentovať. Svojou jednoduchosťou si ľahko dokáže získať mnohých začiatočníkov a hobby používateľov v oblasti embedded systémov [7].

Keďže mikrokontrolér beží na relatívne vysokej frekvencii, svoje využitie nájde aj v oblasti umelej inteligencie. Vďaka svojej nízkej cene nie je problém dosku zakomponovať do zariadení v domácnosti, ako napríklad do vypínačov pre svetlá alebo do elektrických zásuviek. Ďalšie využitie môže nájsť v zariadeniach pre internet vecí či v rôznych vychytávkach pre každodenný život [20].

Spolu s vývojovou doskou spoločnosť Raspberry Pi Pico poskytuje aj sadu ukážkových príkladov, SDK (Software Development Kit), čiže súbor nástrojov umožňujúci vývoj softvéru a taktiež kvalitne spracovanú dokumentáciu pre toto SDK, vývojovú dosku a samotný mikrokontrolér RP2040.

Na trhu už existujú aj ďalšie rôzne dosky, ktorých jadrom je mikrokontrolér RP2040. Tieto dosky vyrábajú napríklad spoločnosti Arduino, Adafruit, Pimoroni a SparkFun [19].

Okrem toho je potrebné spomenúť, že Raspberry Pi má veľmi veľkú komunitu, ktorá publikuje množstvo vytvorených projektov a v prípade akýchkoľvek problémov je ochotná pomôcť. To všetko túto dosku predurčuje ako veľmi úspešný projekt vo svojej oblasti.

1.1 Mikrokontrolér RP2040

Ako bolo v predošlej časti spomenuté, jadrom vývojovej dosky Raspberry Pi Pico je 32-bitový mikrokontrolér RP2040 od spoločnosti Raspberry Pi. Ten so sebou prináša vysoký výkon a jednoduché použitie za nízke náklady. RP2040 je bezstavové zariadenie s podporou priameho vykonávania kódu (XIP) z QSPI rozhrania za pomoci vyrovnávacej pamäte, vďaka čomu si používateľ môže zvoliť potrebnú veľkosť flash pamäte.

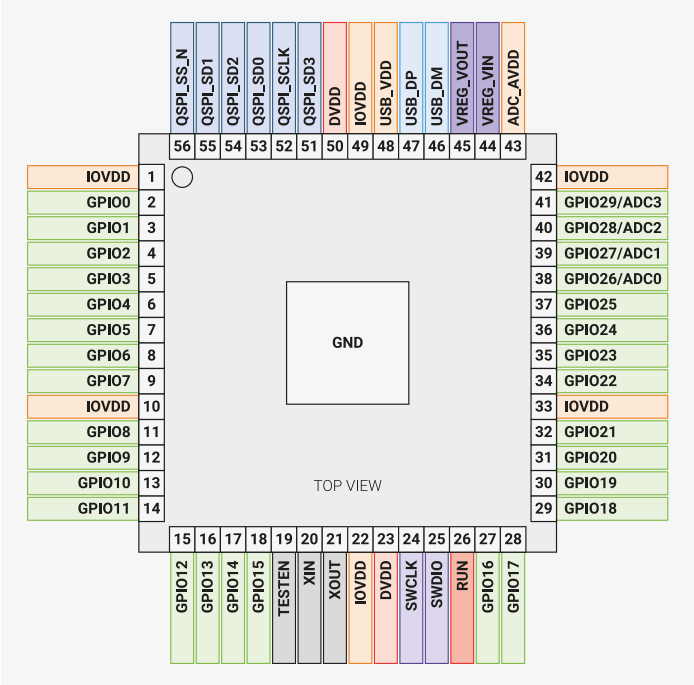
Čip je vyrobený za použitia 40 nm technológie a osadený je v puzdre QFN-56 s rozmermi 7×7 mm. To umožňuje dosiahnuť vysoký výkon, nízku dynamickú spotrebu energie, nízky vyžarovací výkon a podporu viacerých režimov nízkej spotreby energie pre dosiahnutie čo najdlhšej prevádzky zariadení napájaných batériou.

Medzi kľúčové vlastnosti mikrokontroléra RP2040 patrí:

- dvojjadrový mikroprocesor s architektúrou ARM Cortex-M0+ s maximálnou frekvenciou 133 MHz
- SRAM pamäť o veľkosti 264 kB rozdelená do 6 nezávislých buniek

- možnosť pripojenia až 16 MB externej flash pamäte cez rozhranie zbernice QSPI
- DMA radič
- interpolátor a celočíselná delička
- vstavaný programovateľný lineárny regulátor napätia pre generovanie napätia pre procesor
- 2x vstavaný fázový záves na generovanie hodinového signálu pre USB a procesor
- 30 univerzálnych vstupných/výstupných pinov GPIO, z nich 4 môžu byť použité ako analógové vstupy
- periférie:
  - 2x UART rozhranie
  - 2x SPI radič
  - 2x I²C radič
  - 16 PWM kanálov
  - USB fyzická vrstva verzie 1.1 s radičom
  - 8 PIO stavových automatov

Vzhľadom na periférie, ktoré čip obsahuje, má mikrokontrolér dostatočný výkon a funkcie pre použitie na strojové učenie, riadenie motorov či zvukový a obrazový výstup.

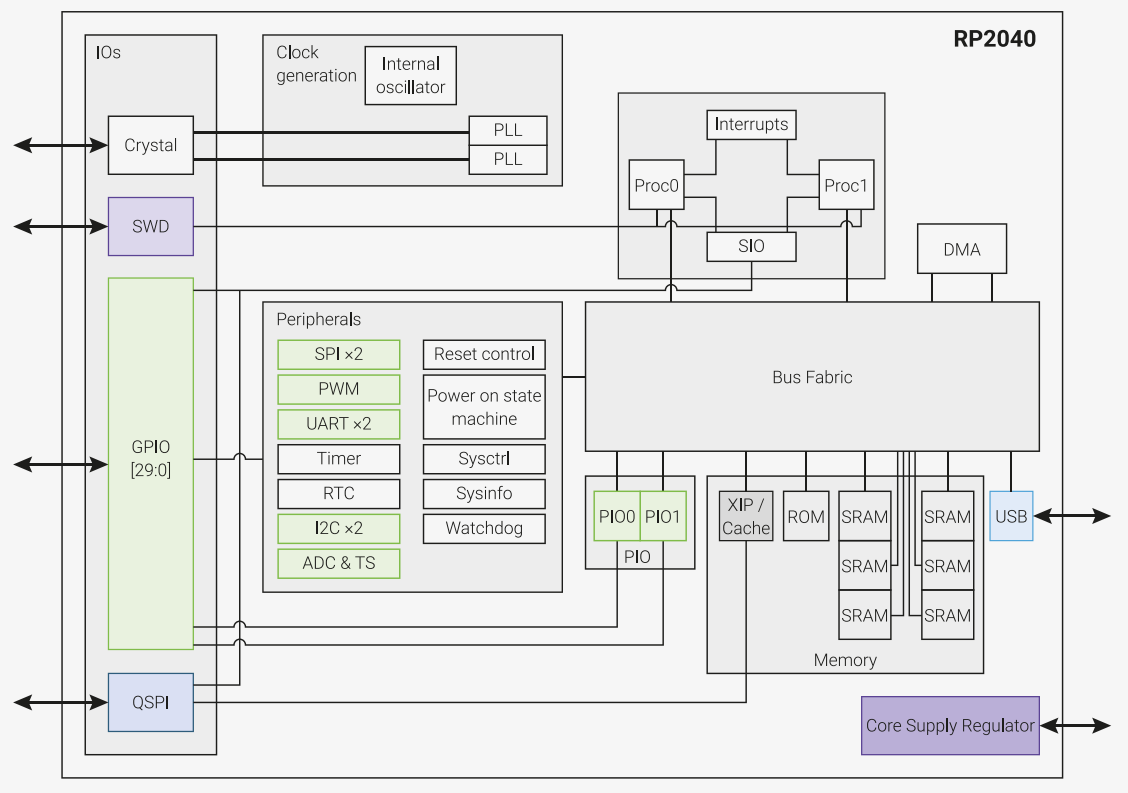


Obr. 1.2: Rozloženie vývodov mikrokontroléra RP2040 [8]

Popis vývodov mikrokontroléra je na obrázku 1.2. Väčšinu z nich tvoria vývody GPIO. Ďalej je vyvedené QSPI rozhranie, vývody pre USB, rozhranie SWD pre ladenie, resetovací pin RUN a vstup hodinového signálu XIN a XOUT pre pripojenie kryštálového oscilátora, s frekvenciou 12 MHz. Vývody pre napájanie budú popísané v ďalšom texte.

1.2 Popis mikrokontroléra

Na obrázku 1.3 je zobrazená vnútorná architektúra mikrokontroléra RP2040. Jeho jednotlivé bloky budú popísané v nasledujúcom texte. Informácie uvedené v tejto podkapitole sú čerpané z [8].



Obr. 1.3: Vnútorná architektúra mikrokontroléra RP2040 [8]

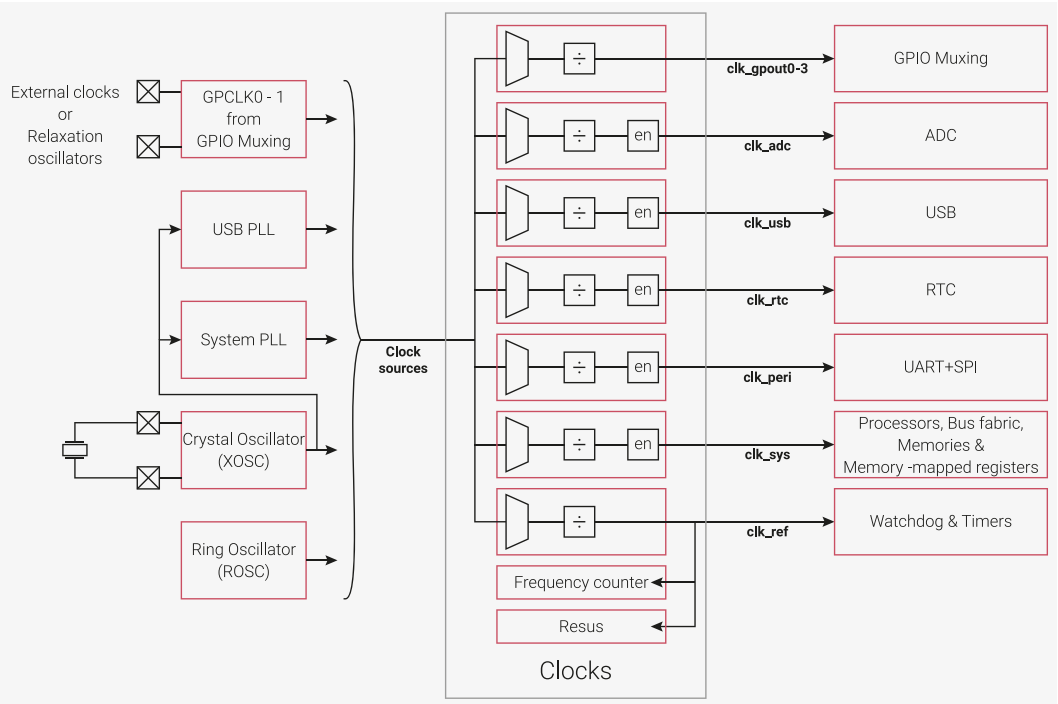
1.2.1 Napájanie mikrokontroléra

Mikrokontrolér RP2040 má päť rôznych vývodov pre napájanie, ale vo väčšine aplikácií je mikrokontrolér pripájaný na jediný napájací zdroj, typicky s napätím 3,3 V. Napájanie digitálnych vstupov a výstupov je realizované cez pin IOVDD. Napäťový

vstup VREG_VIN, slúži na napájanie vnútorného napäťového regulátora. Nominálna hodnota tohto napätia ako aj napätia na vstupe IOVDD je 1,8 až 3,3 V. Výstupom vnútorného regulátora je vývod VREG_VOUT s napätím 1,1 V a maximálnym prúdovým zaťažením 100 mA. Toto napätie sa pripája na vstup DVDD a slúži ako napájanie pre digitálne obvody procesora. Zvlášť sa napája A/D prevodník a fyzická vrstva USB napätím 3,3 V a to cez piny ADC_AVDD a USB_VDD.

K dispozícii je niekoľko možností ako znížiť spotrebu čipu. Ide napríklad o spínanie napájania a hodinového signálu pre jednotlivé periférie, vypnutie pamäte so zachovaním jej dát, či rôzne režimy spánku.

1.2.2 Generovanie hodinového signálu



Obr. 1.4: Prehľad rôznych zdrojov hodinového signálu [8]

Blok hodín (obr. 1.4) poskytuje nezávislé hodinové signály pre čip a externé komponenty. Programátorovi to dáva priestor na výber vhodného zdroja hodín podľa jeho presnosti a spotreby energie. Z týchto zdrojov sú vygenerované potrebné hodinové signály pre ostatné komponenty. Architektúra tohto bloku umožňuje nezávislé spínanie hodín a úpravu generovanej frekvencie vďaka deličke.

V prípade, že na konečnú aplikáciu sú kladené požiadavky, aby mala čo najmenšiu spotrebu energie a najnižšiu výrobnú cenu, pričom presnosť hodín nie je tak

podstatná, je možné využiť interný kruhový oscilátor ROSC. Ak je potreba presného časovania, použije sa externý kryštálový oscilátor pripojený na vývody XIN a XOUT.

PLL

Fázový záves PLL (Phase-locked loop) slúži na násobenie frekvencie hodinového signálu pomocou napäťovo riadeného oscilátora VCO. Na čipe sú dva bloky PLL, jeden s pevne stanovenou frekvenciou 48 MHz pre USB a A/D prevodník, druhý s voliteľnou frekvenciou pre taktovanie systému a to až do 133 MHz. Výpočet parametrov PLL zjednodušuje výpočtový program `vcocalc.py`, ktorý na základe požadovanej frekvencie hľadá najlepšie parametre pre nastavenie PLL tak, aby výsledná frekvencia sa čo najviac blížila požadovanej.

1.2.3 Pamäť mikrokontroléra

RAM pamäť

Dohromady má čip 264 kB SRAM pamäte, ktorá je fyzicky rozdelená do šiestich buniek – štyri bunky o veľkosti 64 kB a dva bunky o veľkosti 4 kB. Členenie pamäte do buniek je iba na fyzickej úrovni, logicky je pamäť braná ako jeden 264 kB celok. Neexistujú žiadne obmedzenia, ktoré by vraveli do ktorej bunky sa majú ukladať dáta alebo kód programu. Každá bunka pamäte je pripojená k zbernici zvlášť, čo znamená, že viacero master zariadení na zbernici môže pristupovať do buniek pamäte paralelne, takže na jeden hodinový takt sa môžu uskutočniť až štyri 32-bitové prístupy k pamäti.

Okrem hlavnej 264 kB pamäte sú k dispozícii aj ďalšie dva bloky RAM pamäte, ktoré môžu byť použité za istých podmienok. Ak sa vyrovnávacia pamäť pre vykonávanie programu z flash pamäte o veľkosti 16 kB nevyužíva, môže sa využiť pre iný účel. Rovnako ak sa nepoužíva USB rozhranie, jeho dedikovaná RAM pamäť o veľkosti 4 kB sa môže tiež využiť. Dohromady to dáva 284 kB pamäte, ktorá je k dispozícii na akýkoľvek účel.

Flash pamäť

K externej flash pamäti pristupuje mikrokontrolér cez QSPI rozhranie, takým spôsobom akoby išlo o internú pamäť. Kód sa môže vykonávať priamo z tejto pamäte, bez predošlého kopírovania do internej pamäte. Z toho plynie názov execute-in-place (XIP), čiže priame vykonávanie programu. Pomocná vyrovnávacia pamäť zrýchľuje prístup k dátam vo flash pamäti a znižuje latenciu a to tak, že si pamätá obsah pamäťových oblastí, ku ktorým nedávno pristupovala.

ROM pamäť

Obsah ROM pamäte o veľkosti 16 kB je pevne daný pri výrobe čipu. Nachádza sa v nej napríklad štartovacia sekvencia pre jadrá procesora, bootloader pre nahrávanie programu alebo dát cez USB do flash alebo RAM pamäte, rutiny pre manipuláciu s flash pamäťou alebo funkcie bežne používané v jazyku C, ako napríklad funkcie pre bitové operácie či kopírovanie pamäte. Hlavnou prednosťou tejto pamäte je, že obsahuje funkcie pre prácu s dátovými typmi s pohyblivou desatinnou čiarkou o veľkosti 32 alebo 64 bitov. Ide napríklad o základné aritmetické funkcie až po zložité goniometrické funkcie a výpočet logaritmu. Podľa [11] sú tieto funkcie niekoľkonásobne rýchlejšie oproti štandardne využívaným funkciám prekladača GCC.

Obsah pamäte ROM sa postupne vyvíja a keďže ho nie je možné meniť, je v prípade potreby nutné siahnuť po novom čipe. V čase písania práce existuje najnovšia verzia tejto pamäte označená ako B2. Od vydania čipu ide doposiaľ o tretiu verziu.

1.2.4 Periférie mikrokontroléra

USB rozhranie

Mikrokontrolér obsahuje fyzickú vrstvu USB verzie 1.1 a USB radič verzie 2.0, ktorý sa stará o nízkoúrovňový USB protokol na hardvérovej úrovni, čo znamená, že hlavnou prácou programátora je nakonfigurovať radič a vyhradiť miesto pre dátové buffre, k čomu sa používa vlastná vyhradená pamäť o veľkosti 4 kB. V prípade potreby interagovania s procesorom, radič vygeneruje prerušenie.

Radič môže pracovať v dvoch režimoch. V prvom režime sa radič správa ako USB zariadenie, ktoré sa pripojí k vyššiemu celku s rýchlosťou 12 Mbps. V druhom režime môže hostiť iné USB zariadenia, pričom pri použití USB rozbočovača vie komunikovať až s 15 koncovými zariadeniami pri plnej rýchlosti 12 Mbps alebo s pomalými zariadeniami pri rýchlosti 1,5 Mbps.

Modul časovača

Časovač poskytuje pre systém globálnu časovú základňu s rozlíšením 1µs a na základe nej generuje prerušenia. Zdrojom hodín je signál *clk_ref*. Hlavnou časťou časovača je 64-bitový čítač, ktorý sa inkrementuje každú mikrosekundu. Čítač prakticky nemôže pretiecť, pretože pri danej časovej základni by to trvalo tisícky rokov. Keďže mikrokontrolér je 32-bitový, čítanie zo 64-bitového čítača, čiže dvoch 32-bitových registrov, by mohlo spôsobiť súbeh operácií na týchto registroch (po anglicky race condition). Aby sa tejto chybe predišlo, k dispozícii je pár 32-bitových latch registrov pre zápis aj pre čítanie. Zapisovanie do čítača je síce technicky možné, ale nie

je to odporúčané, pretože SDK využíva časovač pre funkcie ako je napríklad počítanie uplynutého času, a pre svoju správnu činnosť potrebuje, aby hodnota čítača monotónne narastala.

Časovač má k dispozícii 4 tzv. alarmy, ktoré pri zhode na spodných 32 bitoch čítača vygenerujú prerušenie. To znamená, že uplynúť môžu najviac po 2^{32} mikrosekundách, čo je približne 72 minút.

Hodiny reálneho času RTC

Zatiaľ čo časovač je určený na kratší čas a čas je vyjadrený inkrementujúcim sa čítačom, pre dlhší čas je možné využiť hodiny reálneho času RTC (Real Time Clock) a výstupom z nich je pre človeka ľahko čitateľný formát (tab. 1.1). Zdrojom hodín pre modul je signál *clk_rtc*. Tieto hodiny je taktiež možné využiť na generovanie prerušení v daný čas.

Tab. 1.1: Formát hodín reálneho času

Parameter	**Veľkosť**	**Rozmedzie hodnôt**
Rok	12 bitov	0 – 4095
Mesiac	4 bity	1 – 12
Deň	5 bitov	1 – 31 (podľa mesiaca)
Deň v týždni	3 bity	0 – 6 (0 = nedeľa)
Hodiny	5 bitov	0 – 23
Minúty	6 bitov	0 – 59
Sekundy	6 bitov	0 – 59

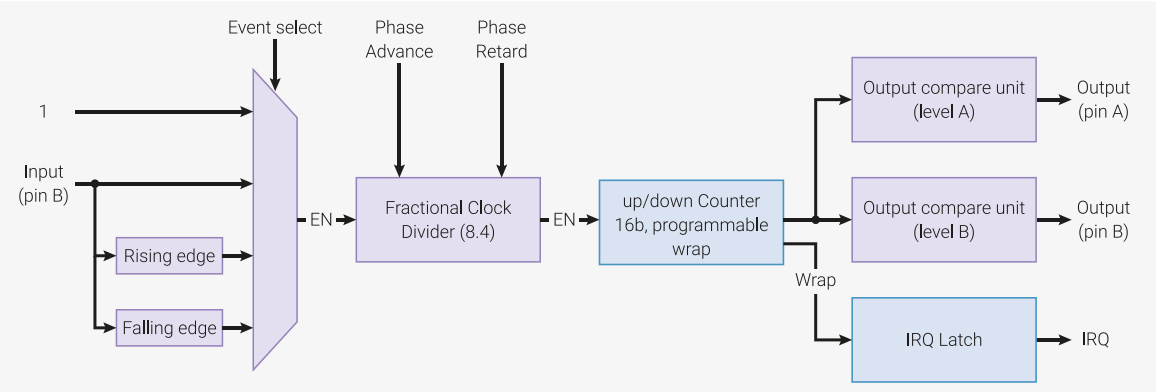
Zapísanie hodnôt času mimo povolené rozmedzie môže viesť k neočakávanému správaniu. Modul RTC nevypočítava správny deň v týždni z daných časových údajov, iba ho inkrementuje.

PWM generátor

Mikrokontrolér RP2040 má 8 identických PWM blokov. Každý blok môže ovládať dva PWM výstupy alebo merať frekvenciu alebo striedu vstupného signálu. Obrázok 1.5 zobrazuje zloženie jedného bloku PWM.

Každý PWM blok obsahuje:

- 16-bitový čítač
- zlomkovú delička hodinového signálu
- 2 nezávislé výstupné kanály



Obr. 1.5: Zloženie periférie PWM [8]

- vstup citlivý na zmenu hrany pre meranie frekvencie
- vstup citlivý na úroveň vstupu pre meranie striedy signálu

PWM výstup môže byť pripojený na každý jeden GPIO výstup. Tých je však 30, zatiaľ čo PWM výstupov je 16. To je vyriešené tak, že PWM výstupy sú v poradí pripojené na GPIO piny 0 až 15 a tie isté PWM výstupy sú pripojené aj na GPIO piny 16 až 29. Z toho vyplýva, že to čo je na výstupe GPIO0, bude aj na výstupe GPIO16 a podobne.

A/D prevodník

Vstavaný A/D prevodník s postupnou aproximáciou (SAR) má rozlíšenie 12 bitov a využíva hodinový signál *clk_adc* generovaný z fázového závesu pre USB. Prečítanie jednej hodnoty z prevodníka trvá 96 cyklov, čiže pri frekvencii 48 MHz to je 2 µs, z čoho vyplýva maximálna vzorkovacia frekvencia 500 kS/s. Efektívna hodnota vstupného odporu je viac ako 100 kΩ. Vstupný multiplexor prevodníka má päť vstupov. Štyri z nich sú vyvedené na vývody GPIO a na jeden je privedený vnútorný snímač teploty. Blok A/D prevodníka vie generovať prerušenia a na prenos dát využívať DMA.

Modul DMA

DMA čiže Direct memory access, čo v preklade znamená priamy prístup do pamäte, je funkcia, ktorá umožňuje subsystémom procesora používať systémovú pamäť typicky na prenos veľkých objemov dát nezávisle od procesora. To výrazne znižuje zaťaženie procesora, ktorý takto môže vykonávať iné úlohy alebo byť v režime spánku kvôli nízkej spotrebe energie. DMA radič mikrokontroléra RP2040 je k zbernici pripojený zvlášť pre čítanie a zvlášť pre zápis, vďaka čomu môže preniesť, teda prečítať

a zapísať až 32 bitov každý hodinový takt. O správne spárované adresy čítania a zápisu sa stará generátor adries. Súčasne môže prebiehať až 12 prenosov.

Existujú 3 typy prenosu dát:

- Z pamäte do periférie – periféria signalizuje DMA radiču, keď potrebuje ďalšie dáta na prenos. Radič prečíta dáta z RAM alebo flash pamäte a zapíše ich do periférie.
- Z periférie do pamäte – periféria signalizuje DMA radiču, keď prijala dáta. Radič prečíta dáta z periférie a zapíše ich do pamäte RAM.
- Z pamäte do pamäte – DMA radič prenáša dáta medzi dvoma buffermi v pamäti RAM tak rýchlo, ako je to možné.

Každý prenosový kanál riadia stavové a riadiace registre, ktoré sa konfigurujú programovo. Veľkosť prenosu môže byť 8, 16 alebo 32 bitov a nastavuje sa zvlášť pre každý kanál.

Periféria PIO

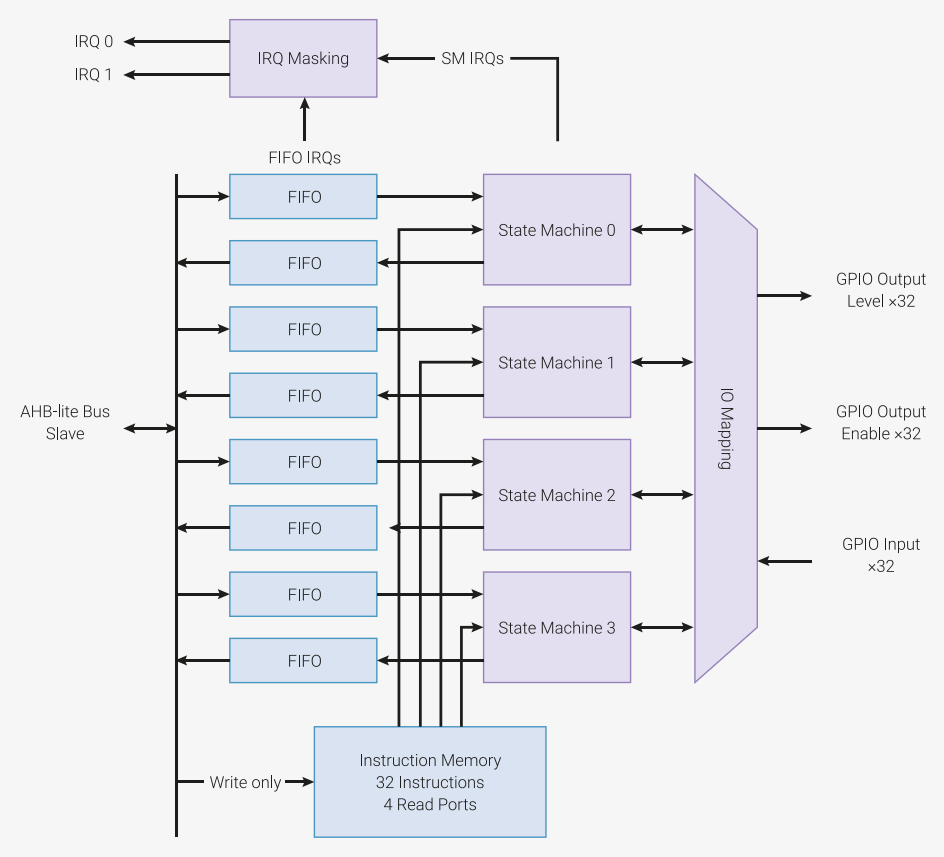
Mikrokontrolér obsahuje dva bloky programovateľného vstupno-výstupného rozhrania PIO (Programmable I/O). Ide o blok, ktorý poskytuje univerzálne hardvérové rozhranie, pomocou ktorého sa dá naprogramovať napríklad rozhranie I²C, I²S, SPI, UART alebo dokonca aj rozhranie VGA bez toho, aby to akokoľvek využívalo procesor.

Obrázok 1.6 zobrazuje blokovú schému PIO bloku na základe, ktorej je možné povedať, že jeden blok PIO obsahuje štyri stavové automaty, ktoré môžu súčasne vykonávať programy z vlastnej pamäte vyhradenej pre inštrukcie. Každý z blokov má dedikované pripojenie na zbernicu a vie generovať prerušenie. Každý stavový automat má k dispozícii dva dátové fronty FIFO slúžiace na prenos dát medzi PIO blokom a nadradeným systémom a môže ovládať všetky GPIO.

Každý stavový automat zaberá na čipe približne rovnaké miesto ako štandardná periféria ako je SPI alebo I²C. Stavové automaty je však možné dynamicky konfigurovať, vďaka čomu je možné implementovať množstvo rôznych rozhraní, ktoré nie sú na čipe vstavané alebo je ich potrebné využiť väčšie množstvo ako je k dispozícii.

Blok PIO je programovateľný v rovnakom zmysle ako procesor. Využíva však svoje vlastné inštrukcie:

- `JMP` – inštrukcia podmieneného skoku
- `WAIT` – čakanie za splnením určitej podmienky
- `IN` – načítanie binárnych vstupov alebo pomocného registra
- `OUT` – zapísanie na binárne vstupy alebo do pomocného registra
- `PUSH` – zapísanie 32 bitov do dátovej fronty FIFO



Obr. 1.6: Programovateľné vstupno-výstupné rozhranie PIO [8]

- `PULL` – načítanie 32 bitov z dátovej fronty FIFO
- `MOV` – presun dát medzi vstupmi a výstupmi alebo pomocnými registrami
- `IRQ` – vygenerovanie alebo resetovanie prerušenia
- `SET` – zapísanie konštanty na binárny výstup alebo do pomocného registra

1.3 Popis vývojovej dosky Raspberry Pi Pico

Vývojová doska Raspberry Pi Pico je vyrobená ako plošný spoj o rozmeroch 21×51 mm so 40 vývodmi s rozostupom 2,54 mm. Navonok má teda doska rovnaké rozloženie vývodov ako púzdro DIP40, ktoré sa používa pre vývodové THT integrované obvody. Vývody na plošnom spoji (obr. 1.1) sú špeciálne vyrobené tak, aby bolo možné dosku prispájkovať k inému plošnému spoju ako modul pre povrchovú montáž SMD. Výrobca dodáva dosku bez osadených pinových líšt pre vývody, čím používateľovi necháva možnosť vybrať si ako dosku použije. Okrem týchto vývodov obsahuje ďalšie 3 vývody pre rozhranie SWD, ktoré sa využíva na ladenie. Pre spoľahlivé upevnenie dosky je možné využiť 4 montážne otvory s priemerom 2,1 mm.

Doska môže pracovať v teplotách od -20 °C do +85 °C. Ide o rozšírený teplotný rozsah pre bežné komerčné využitie. Z toho vyplýva, že dosku nie je možné využiť pre automobilový priemysel alebo pre armádu, kde sú vyžadované väčšie teplotné rozsahy [46].

Jadrom Raspberry Pi Pico je mikrokontrolér RP2040, ktorý bol podrobne popísaný v predošlom texte tejto práce. Na vývojovej doske sú osadené základné súčiastky, ktoré mikrokontrolér požaduje pre svoju funkcionalitu, a to flash pamäť o veľkosti 2 MB, kryštál s frekvenciou 12 MHz, napájací obvod, blokovacie kondenzátory a micro-USB konektor typu B pre napájanie a prenos dát ako aj programovanie flash pamäte. Navyše je na doske osadená programovateľná nízkosvietivá zelená LED dióda a tlačidlo BOOTSEL, ktorého funkcia bude popísaná v ďalšom texte.

Napájacie napätie VBUS je 5 V z USB portu alebo externé napájanie spínaného zdroja VSYS v širokom rozsahu 1,8 až 5,5 V. To umožňuje značnú flexibilitu v napájaní dosky z rôznych zdrojov ako je napríklad jeden článok Li-ion batérie alebo 3 články AA alebo AAA batérií v sérii. Takisto nie je problém integrácie nabíjačky batérie do napájacieho obvodu dosky. Univerzálne vstupné/výstupné piny GPIO pracujú fixne na 3,3 V logickej úrovni.

Výrobca zaručuje výrobu Raspberry Pi Pico minimálne do januára 2028. Zhrnutie základných parametrov vývojovej dosky je uvedené v tabuľke 1.2. Informácie uvedené v tejto podkapitole sú čerpané z [9].

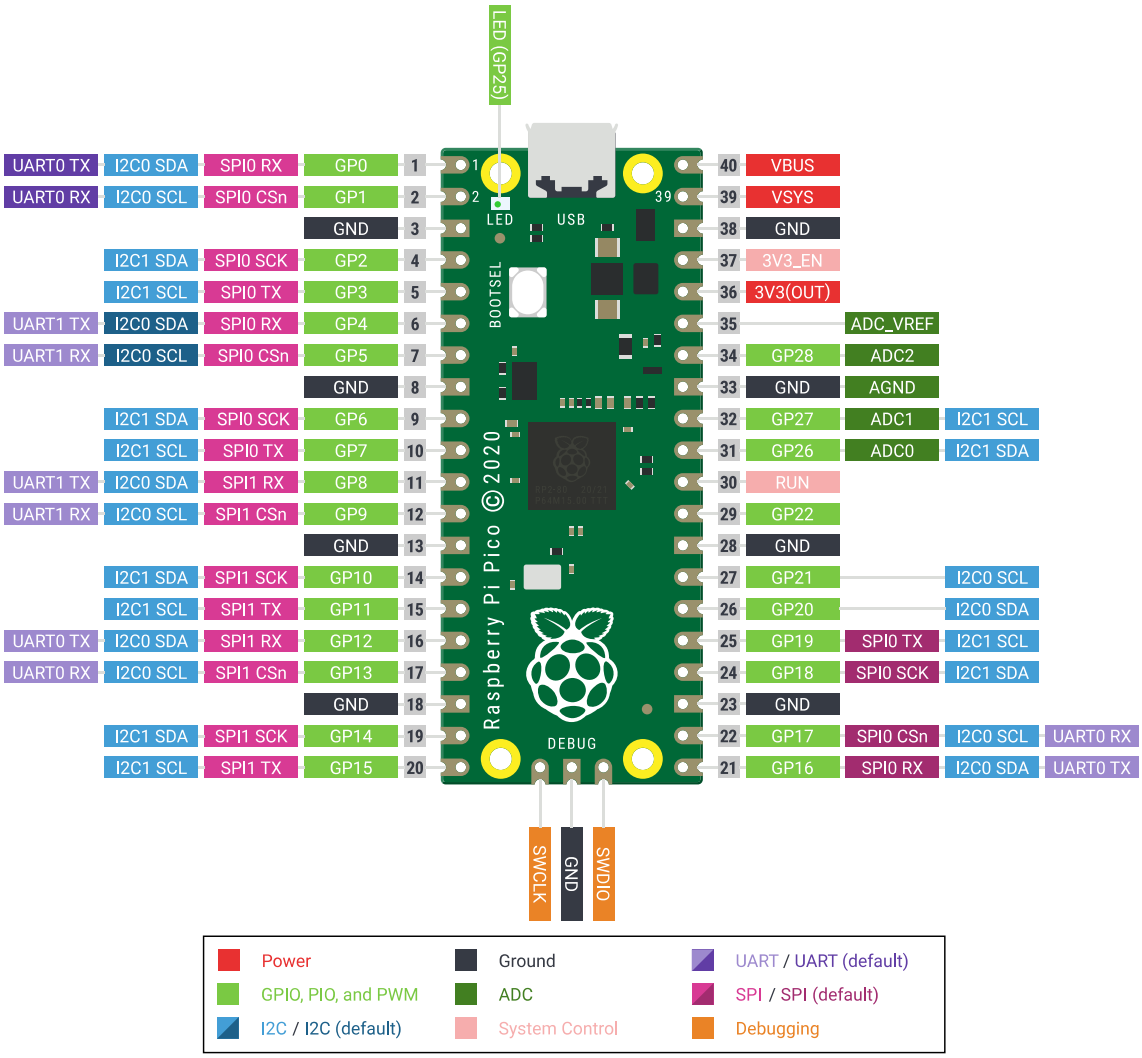
Tab. 1.2: Technické parametre vývojovej dosky Raspberry Pi Pico

Rozmery	21×51 mm
Pracovný rozsah teplôt	-20 až +85 °C
Napätie USB zbernice	5 V ±10 %
Externé napájanie	1,8 až 5,5 V DC
Logická úroveň GPIO	3,3 V

1.3.1 Popis vývodov

Vývojová doska bola navrhnutá tak, aby poskytla čo najlepšie využitie GPIO mikrokontroléra RP2040 a jeho vnútorných funkcií. Preto väčšina vývodov mikrokontroléra (obr. 1.2) je vyvedených aj na vývody vývojovej dosky. Ohľad bol braný aj na poskytnutie vhodného množstva zemniacich vývodov pre dosiahnutie čo najnižšej elektromagnetickej interferencie a presluchov medzi signálmi.

Rozloženie vývodov dosky a ich možné využitie je zobrazené na obrázku 1.7. Dohromady je vyvedených 26 z celkovo 30 GPIO pinov. Piny GPIO0 až GPIO22



Obr. 1.7: Popis vývodov vývojovej dosky Raspberry Pi Pico [9]

sú digitálne a piny GPIO26 až GPIO28 je možné využiť ako digitálne alebo analógové vstupy. Každý z týchto pinov disponuje viacerými funkciami, ako je napríklad UART, SPI, I²C. To ako bude pin využitý sa nastaví v programe. Piny s označením SWCLK a SWDIO slúžia na ladenie programu cez SWD rozhranie. Niekoľko GPIO vývodov mikrokontroléra RP2040 je použitých pre interné funkcie dosky:

- GPIO23 – výstup – ovládanie napájacieho obvodu (viac v časti 1.3.2)
- GPIO24 – vstup – snímanie napätia VBUS (log. 1 v prípade prítomnosti napätia, inak log. 0)
- GPIO25 – výstup – programovateľná LED dióda
- GPIO29 – vstup – analógové meranie napätia VSYS/3

Okrem GPIO a zemniacich pinov je z vývojovej dosky vyvedené aj niekoľko

ďalších pomocných pinov:

- VBUS – napájacie napätie z USB zbernice, ktorého nominálna hodnota je 5 V
- VSYS – systémové napájacie napätie v rozsahu 1,8 až 5,5 V, z ktorého sa generuje napájacie napätie mikrokontroléra 3,3 V
- 3V3_EN – pripojenie na povoľovací pin spínaného zdroja
- 3V3 – napájacie napätie 3,3 V mikrokontroléra generované spínaným zdrojom
- ADC_REF – referenčné napätie A/D prevodníka
- AGND – referenčný zemniaci vývod pre analógové vstupy
- RUN – povoľovací pin pre mikrokontrolér, vnútorne pripojený cez pull-up rezistor na napätie 3,3 V

Na tomto mieste je nutné podotknúť, že vývojová doska Raspberry Pi Pico nemá žiadne RESET tlačidlo. V prípade potreby je možné si toto tlačidlo externe doplniť a to tak, aby sa jeho stlačením prepojil povoľovací pin mikrokontroléra RUN so zemou.

GPIO piny, ktoré je možné využiť ako analógové vstupy, majú vnútorne pripojenú diódu v závernom smere na napätie VDDIO. Preto napätie pripojené na nich nesmie prekročiť napätie VDDIO o viac ako 300 mV, čo by spôsobilo otvorenie diódy. To isté by spôsobilo priloženie napätia na tieto piny v čase vypnutého mikrokontroléra.

Napäťová referencia A/D prevodníka

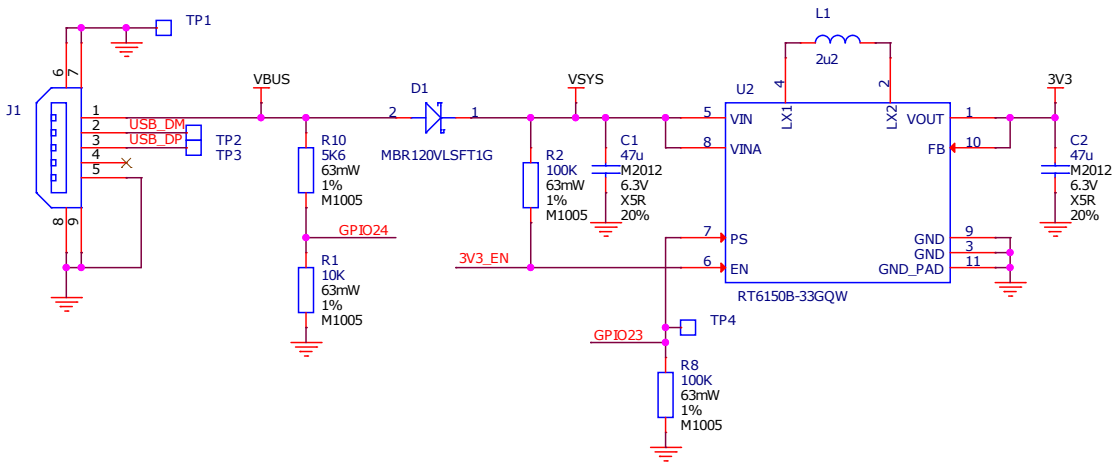
Napäťová referencia pre A/D prevodník mikrokontroléra je na vývojovej doske generovaná z napätia 3,3 V spínaného zdroja za použitia dolnopriepustného RC filtra s medznou frekvenciou 360 Hz. Ide o jednoduché riešenie, ktoré však vnáša do obvodu určitú chybu, z dôvodu nepresnosti spínaného zdroja.

V prípade potreby presnejšieho merania je odporúčané použiť externú napäťovú referenciu pripojenú na pin ADC_VREF.

1.3.2 Napájanie dosky

Raspberry Pi Pico vďaka svojmu flexibilnému napájaciemu obvodu umožňuje jednoduché napájanie nielen z USB portu, ale aj z iných zdrojov ako sú napríklad batérie alebo externé zdroje. Rovnako jednoduché je aj integrovanie externého nabíjacieho obvodu pre napájací akumulátor. Obrázok 1.8 zobrazuje elektrickú schému zapojenia napájacej časti.

Napätie VBUS je 5 V vstup z micro-USB portu. Toto napätie je pripojené na vstup spínaného zdroja VSYS cez Schottkyho diódu, ktorá zabraňuje spätnému prúdu do zdroja, v prípade použitia viacerých zdrojov. Rovnakým spôsobom je možné na vstup VSYS pripojiť aj iné napájacie zdroje. Napätie VSYS je cez RC



Obr. 1.8: Schéma zapojenia napájacej časti vývojovej dosky Raspberry Pi Pico [9]

filter a napäťový delič s prenosom $K_u = 1/3$ privedené na 3. kanál A/D prevodníka, vďaka čomu je možné monitorovať hodnotu vstupného napätia.

Ako spínaný zdroj je použitý blokujúci DC/DC menič, ktorý vie z akéhokoľvek napätia v jeho pracovnom rozsahu, konkrétne 1,8 až 5,5 V, vygenerovať požadované napätie 3,3 V pre napájanie mikrokontroléra RP2040. Obvod vie pracovať v dvoch režimoch. Pracovný režim je daný logickou hodnotou na vstupnom pine meniča PS. Keď je na pin pripojená log. 0, menič beží v režime impulzovej frekvenčnej modulácie PFM, čo značne šetrí energiu. V opačnom prípade, kedy je na pine log. 1, menič prejde do režimu impulzovej šírkovej modulácie PWM. Tento režim znižuje zvlnenie výstupného napätia pri malej záťaži, avšak za cenu oveľa horšej účinnosti.

Pin PS je pripojený cez pull-down rezistor k zemi, čím je stanovené, že predvoleným režimom meniča je PFM mód. Programovo sa tento režim dá ovládať pomocou pinu GPIO23. V prípade veľkej záťaže, menič bude pracovať v PWM móde bez ohľadu na stav pinu PS.

Povoľovací pin DC/DC meniča je pripojený cez pull-up rezistor na napätie VSYS. Taktiež je vyvedený z dosky Raspberry Pi Pico na pin č. 37. Jeho pripojením na zem sa menič vypne.

Vývod na doske s označením 3V3 je výstup DC/DC meniča a je možné ním napájať externé obvody. Celkový prúd tečúci z tohto vývodu by nemal presiahnuť 300 mA.

1.4 Základy práce s vývojovou doskou

Pre prácu s vývojovou doskou je potrebné stiahnuť a nainštalovať niekoľko programovacích nástrojov ako je napríklad CMake alebo prekladač pre architektúru ARM.

Dosku je možné programovať cez operačné systémy Linux, MacOS, Windows na rôznych architektúrach, a preto postup pre inštaláciu je pre každý systém odlišný a v tejto práci nie je uvedený. Podrobný návod na inštaláciu je možné nájsť v [10], odkiaľ sú čerpané aj informácie uvedené v tejto časti.

1.4.1 Programovacie jazyky

Vývojovú dosku Raspberry Pi Pico je možné programovať v jazyku symbolických adries a v jazyku C/C++. Dosku je taktiež možné programovať aj v jazyku Python vďaka softvérovej implementácií interpretačného prekladača v jazyku C s názvom MicroPython, ktorá je optimalizovaná pre použitie v mikrokontroléroch. Jazyk Python v tejto práci nebude využitý, avšak ďalšie informácie o jeho použití je možné nájsť v [12].

Výpis 1.1: Program v jazyku symbolických adries [2]

```
.thumb_func     @ necessary because sdk uses BLX
.global main    @ provide program starting address to linker

main:
    MOV R7, #0          @ initialize counter to 0
    BL  stdio_init_all  @ initialize uart or usb
loop:
    LDR R0, =helloworld @ load address of helloworld string
    ADD R7, #1  @ increment counter
    MOV R1, R7  @ move the counter to second parameter
    BL  printf  @ call printf
    B loop      @ loop forever

.data
.align 4        @ necessary alignment
helloworld: .asciz   "Hello␣world␣%d\n"
```

Na výpisoch 1.1 a 1.2 je uvedený rôzny spôsob zápisu toho istého programu v jazyku symbolických adries a v jazyku C, ktorého úlohou je vypísanie textu `Hello World` s poradovým číslom aktuálnej iterácie na UART port.

1.4.2 C/C++ SDK

SDK (Software development kit), čiže súbor nástrojov umožňujúci vývoj softvéru, poskytuje hlavičkové súbory, knižnice a zostavovací systém nevyhnutné na vývoj

Výpis 1.2: Program v jazyku C

```
#include <stdio.h>
#include "pico/stdlib.h"

int main()
{
    int32_t i = 0;       // initialize counter to 0
    stdio_init_all();    // initialize uart or usb
    for (;;)             // loop forever
    {
        ++i;                                  // increment counter
        printf("Hello␣world␣%d\n", i);  // call printf
    }
}
```

programov pre zariadenia založené na mikrokontroléri RP2040 ako je aj doska Raspberry Pi Pico v jazyku C, C++ a jazyku symbolických adries. Raspberry Pi poskytuje SDK aj pre jazyk Python [12]. SDK je navrhnuté tak, aby poskytovalo API[1], s ktorým sa jednoducho pracuje. Taktiež toto SDK prináša podporu pre nízkoúrovňové funkcie pre prístup k perifériám mikrokontroléra vrátane prerušení, DMA alebo PIO. Okrem toho SDK poskytuje aj funkcie na vyššej úrovni pre prácu s USB, viacjadrové programovanie alebo funkcie na generovanie zvuku za použitia periférie PIO. Všetky funkcie sú veľmi dobré zdokumentované, a preto je možné si v prípade potreby naprogramovať vlastné funkcie pracujúce s registrami mikrokontroléra [11].

SDK je možné stiahnuť zo stránky [36], pričom v čase písania tejto práce bola najaktuálnejšou verziou SDK verzia 1.3.0, ktorá sa ďalej v tejto práci aj používa.

Výpis 1.3 ukazuje príklad použitia C/C++ SDK. Vývod pre LED je inicializovaný a nastavený ako výstup pomocou funkcií `gpio_init` a `gpio_set_dir`. Následne v nekonečnej slučke je funkciou `gpio_put` zapisovaná logická hodnota na tento výstup. Oneskorenie pre vytvorenie efektu blikania zabezpečuje funkcia `sleep_ms`. SDK bolo použité aj v programoch 1.1 a 1.2, v ktorých funkcia `stdio_init_all`, umožnila používať funkciu zo štandardnej knižnice jazyka C `printf`.

V uvedených príkladov neboli priamo nastavované žiadne registre, ako je to zvykom pri programovaní mikrokontrolérov, pretože ich nastavenie zabezpečilo volanie funkcií SDK, čo výrazne zjednodušuje prácu programátora.

[1]API je skratka pre aplikačné programovacie rozhranie. Ide o zbierku procedúr, funkcií, tried, či protokolov nejakej knižnice, ktoré môže programátor používať [47].

Výpis 1.3: Program na blikanie LED diódy za použitia C/C++ SDK [37]

```
#include "pico/stdlib.h"

int main()
{
    const uint LED_PIN = PICO_DEFAULT_LED_PIN;
    gpio_init(LED_PIN);
    gpio_set_dir(LED_PIN, GPIO_OUT);

    for (;;)
    {
        gpio_put(LED_PIN, 1);
        sleep_ms(250);
        gpio_put(LED_PIN, 0);
        sleep_ms(250);
    }
}
```

1.4.3 Zostavovací systém

Pre riadenie procesu kompilácie programov sa používa program CMake. Ide o súbor nástrojov na vytváranie a testovanie softvéru. Na riadenie kompilácie využíva jednoduché konfiguračné súbory, ktoré sú nezávislé na platforme a prekladači. Tieto nástroje je možné využiť naprieč rôznymi vývojovými prostrediami.

CMake sa dá využiť aj na nakonfigurovanie definícií pre preprocesor prekladača pre konkrétny program, alebo v závislosti na tom pre akú dosku s mikrokontrolérom RP2040 je program vytvorený sa nadefinuje požadované mapovanie vstupov a výstupov [11].

Základom programu CMake sú súbory s názvom `CMakeLists.txt`. Ukážku takého súboru zobrazuje výpis 1.4. Na začiatku súboru je zadefinovaná minimálna požadovaná verzia CMake, ďalej je priložený CMake súbor, ktorý zaručí správne importovanie SDK do projektu. Ďalej je zadefinovaný samotný názov projektu, zoznam použitých programovacích jazykov a ich štandardy. Nasleduje inicializácia SDK, voliteľné pridanie parametrov pre prekladač a vytvorenie spustiteľného súboru `.elf` zo zdrojových súborov. Nakoniec sa k programu nalinkujú potrebné knižnice a vyprodukujú sa ďalšie súbory s týmito príponami:

- `.bin` – binárne zapísaný preložený program
- `.hex` – hexadecimálne zapísaný preložený program
- `.uf2` – umožňuje jednoduché nahratie programu cez USB rozhranie

Výpis 1.4: Príklad zápisu súboru `CMakeLists.txt`

```
cmake_minimum_required(VERSION 3.13)

# include pico sdk
include(pico_sdk_import.cmake)

# provide project name
# and list of used languages and their standards
project(HelloWorld C CXX ASM)
set(CMAKE_C_STANDARD 11)
set(CMAKE_CXX_STANDARD 17)

# initialize the SDK
pico_sdk_init()

# additional options for compiler
add_compile_options(-Wall
                    -Wno-format)

# add executable called "HelloWorld" that is built
# from the source file "HelloWorld.S"
add_executable(HelloWorld
  HelloWorld.S
)

# link the "pico_stdlib" library to executable
target_link_libraries(HelloWorld pico_stdlib)

# create map/bin/hex file etc.
pico_add_extra_outputs(HelloWorld)
```

- `.map` – detailné informácie o využití pamäte
- `.dis` – program zapísaný v jazyku symbolických adries

Podrobné informácie o zostavení programu je možné nájsť v dokumentácii [10] a [11], všeobecne však je potrebné vykonať tieto kroky:

1. Pre úspešný preklad je potrebné do systémovej premennej `PICO_SDK_PATH` uložiť cestu ku C/C++ SDK.
2. Vytvorením priečinka `build` vznikne pracovný priestor pre uloženie pomoc-

ných súborov a zostavených programov.

3. Vykonanie príkazu `cmake ..` v priečinku `build` zabezpečí vygenerovanie všetkých potrebných súborov pre preklad na danej platforme.
4. Posledným krokom je samotné zostavenie príkazom `make -j N`, kde N vyjadruje počet jadier procesora, ktoré sa majú na preklad využiť, čím môže dôjsť k výraznemu zrýchleniu prekladu.

1.4.4 Nahrávanie programu

Najrýchlejším spôsobom ako nahrať firmvér do vývojovej dosky je pripojiť ju ako veľkokapacitné pamäťové USB zariadenie. Takto je možné presunúť súbor s príponou `.uf2` z počítača na toto USB zariadenie, čím sa program zapíše do flash pamäte osadenej na doske. Následne sa mikroprocesor reštartuje, odpojí sa ako pamäťové USB zariadenie a začne vykonávať nahraný program.

Pripojenie ako veľkokapacitné pamäťové USB zariadenie sa docieli stlačením tlačidla BOOTSEL a súčasným resetovaním mikroprocesora, čiže napríklad odpojením a pripojením k počítaču pri stlačenom tlačidle BOOTSEL.

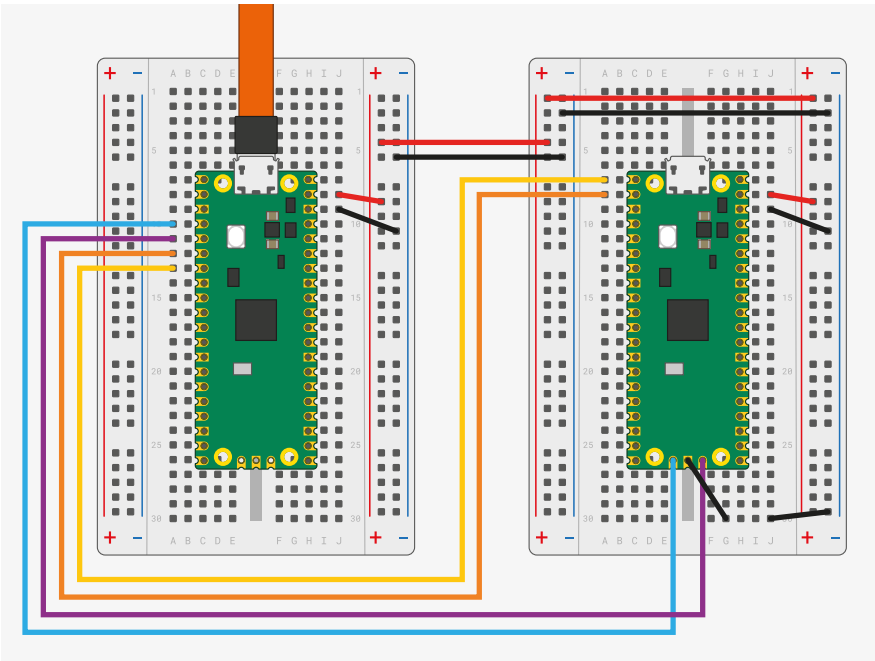
Tento spôsob nahrávania firmvéru do vývojovej dosky nie je najlepší, pretože časté pripájanie a odpájanie pri vývoji a neustálom testovaní firmvéru môže rýchlo viesť k opotrebovaniu USB konektora jednak na doske a jednak v počítači. Ďalší spôsob ako program nahrať je použitie SWD rozhrania, čo bude bližšie popísané v nasledujúcej časti.

1.4.5 Programovanie a ladenie cez SWD rozhranie

Pre komunikáciu s vývojovou doskou prostredníctvom SWD rozhrania je potrebné stiahnuť a nainštalovať program OpenOCD. Návod na jeho inštaláciu sa nachádza v [10]. Ako prevodník medzi počítačom a SWD rozhraním sa použije ďalšia vývojová doska Raspberry Pi Pico s nahraným programom `picoprobe.uf2`. Vzájomné prepojenie dosiek je zobrazené na obrázku 1.9. Doska naľavo slúži ako debugger a doska napravo je doska, ktorá sa programuje. Komunikácia cez UART port z pôvodnej dosky je premostená do druhej, aby bolo možné komunikovať s programovanou doskou cez jeden USB port.

Výpis 1.5: Príkaz pre nahratie programu cez SWD rozhranie

```
openocd -f interface/raspberrypi-swd.cfg -f target/rp2040.cfg
-c "program␣blink/blink.elf␣verify␣reset␣exit"
```



Obr. 1.9: Raspberry Pi Pico vo funkcii debuggera [10]

Príkaz 1.5 slúži pre nahratie programu `blink.elf` do vývojovej dosky cez SWD rozhranie. Pozorný čitateľ si iste povšimol, že cez SWD sa nahráva súbor s príponou `.elf` namiesto súboru `.uf2`.

Vývojová doska sa dá odladzovať cez SWD rozhranie pomocou programu GDB. Programom OpenOCD sa spustí server, ku ktorému sa následne pripojí program GDB, cez ktorý ladenie prebieha. Medzi hlavné nástroje patrí nastavenie miest pre zastavenie behu programu (breakpoint), krokovanie vykonávania programu po jednotlivých riadkoch, čítanie hodnôt premenných a registrov.

1.4.6 Vývojové prostredie

Aby bol uľahčený vývoj programu a jeho odladzovanie, používa sa vývojové prostredie Visual Studio Code. Ide o voľne dostupný program od spoločnosti Microsoft použiteľný na viacerých architektúrach a vo všetkých bežne používaných operačných systémoch.

Pre prácu s Raspberry Pico je potrebné do neho nainštalovať doplnky CMake, CMake Tools, C/C++ a Cortex-Debug a v nastaveniach rozšírenia CMake Tools nastaviť premennú `Cmake: Generator` na hodnotu `Unix Makefiles`. Pre správne fungovanie je ešte potrebné nastaviť vývojové prostredie použitím konfiguračného súboru `settings.json` a debugger súborom `launch.json`. Vzorové súbory sú dostupné v elektronickej prílohe F v priečinku `.vscode`. Pre každý počítač v nich však

treba nastaviť správnu cestu k programom OpenOCD, GDB a tiež cestu ku C/C++ SDK.

Ďalšie vývojové prostredia, ktoré je možné využiť sú CLion a Eclipse. Microsoft Visual Studio sa dá taktiež využiť, ale je k tomu potrebné zakúpiť rozšírenie programu VisualGDB [21].

2 Modernizácia laboratórnych úloh predmetu Vstavané systémy a mikroprocesory

Hlavným cieľom tejto práce je modernizácia laboratórnych úloh predmetu Vstavané systémy a mikroprocesory. V tejto kapitole bude najprv uvedená stručná charakteristika predmetu a jeho náplň. Ďalej bude uvedený rozbor súčasných laboratórnych úloh. Výsledkom tejto kapitoly bude návrh nových úloh pre vývojový kit Raspberry Pi Pico.

2.1 Zaradenie predmetu do študijného programu

Predmet Vstavané systémy a mikroprocesory, so skratkou BPC-MIC, je povinný predmet na VUT FEKT v Brne. Vyučuje sa v 2. ročníku v letnom semestri v bakalárskom študijnom programe Automatizačná a meracia technika. Výuku zaisťuje Ústav automatizácie a meracej techniky (UAMT). Týždenná časová dotácia pre predmet je 2 hodiny prednášok a 3 hodiny cvičení po dobu 13 týždňov.

Povinnou prerekvizitou pre tento predmet je absolvovanie predmetu Úvod do programovania. Na tomto predmete sa študenti naučia základné princípy fungovania počítača a základy programovania v jazyku C. Rovnako sa vychádza aj z predpokladu, že študenti absolvovali povinné predmety Návrh elektronických obvodov a tiež Logické obvody a systémy. V prvom z nich sa naučili vlastnosti a princíp fungovania základných pasívnych a aktívnych súčiastok v elektronických obvodoch. V druhom z nich sa zoznámili so základným fungovaním logických obvodov ako napríklad kombinačné a sekvenčné obvody, čítač, časovač, konečný stavový automat a pritom sa tiež naučili základy programovania v jazyku VHDL. Študent by mal byť teda schopný po absolvovaní týchto predmetov vytvoriť jednoduchý program v jazyku C a rozumieť základom HW dizajnu.

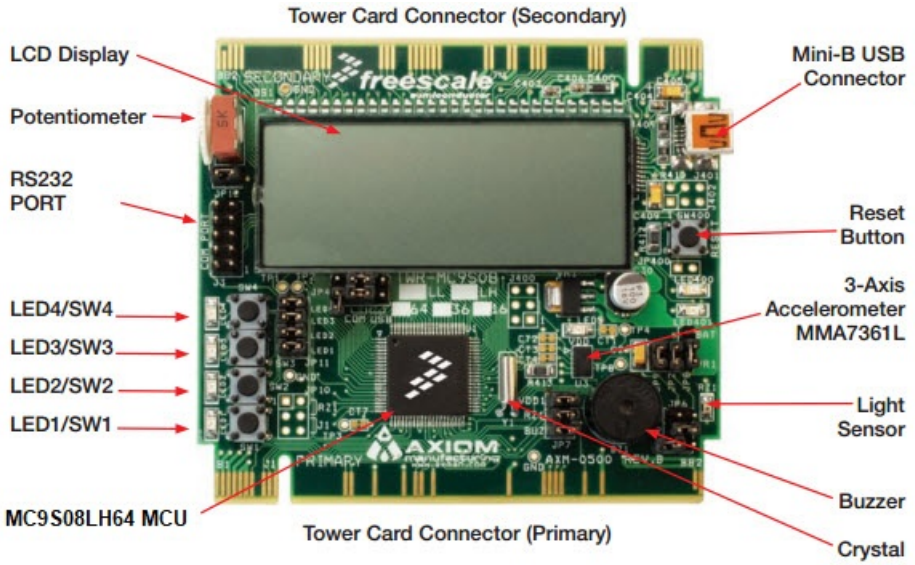
Predmet Vstavané systémy a mikroprocesory je zameraný na základné princípy mikroprocesorových a embedded systémov. V rámci predmetu sa študenti zoznámia s princípom činnosti mikroprocesorov, perifériami integrovanými na čipoch mikroprocesorov, použitím periférnych subsystémov mikrokontrolérov, pamäťovými systémami a správou pamäte[22].

2.2 Laboratórne cvičenia predmetu

Nadobudnuté znalosti z prednášok sa následne aplikujú na cvičeniach, čím študent získa praktické skúsenosti s obsluhou vstupno-výstupných periférií a s programovaním embedded systémov v jazyku symbolických adries a jazyku C.

2.2.1 Súčasná vývojová doska

V súčasnosti sa na laboratórnych cvičeniach predmetu BPC-MIC používa vývojová doska TWR-S08LH64 (obr. 2.1). Jej jadrom je 8-bitový mikrokontrolér z rodiny HCS08 od výrobcu NXP, konkrétne model MC9S08LH64. Mikrokontroléry z tejto rodiny sa vyznačujú rýchlym prístupom do pamäte, malým počtom registrov a komplexnou inštrukčnou sadou CISC. Nechýba ani podpora 32 možných zdrojov prerušenia, dekódovanie adresy na úrovni čipu, či hardvérová násobička a delička. Maximálna frekvencia je 40 MHz a rýchlosť zbernice najviac 20 MHz. Medzi jeho hlavné moduly patrí rozhranie pre LCD displej, 16-bitový A/D prevodník, modul komunikácie I²C, UART a SPI, analógový komparátor, časovač, modul času, modul prerušenia klávesnice a celkovo až 39 vstupno-výstupných pinov [5].



Obr. 2.1: Vývojová doska TWR-S08LH64 a jej periférie [23]

Vývojová doska komunikuje s PC cez USB rozhranie. Základnú interakciu s používateľom zabezpečujú 4 tlačidlá, 4 LED diódy, piezoelektrický menič a potenciometer. Pre snímanie okolia je na doske osadený snímač osvetlenia v podobe fototranzistora a trojosý akcelerometer s analógovými výstupmi. Ako zobrazovacia jednotka je použitý LCD displej, ktorý je priamo pripojený s mikrokontrolérom. Ten je určený najmä na zobrazovanie časových údajov. Z dosky je vyvedené RS-232 rozhranie a rozhranie Tower System pre prepojenie viacerých vývojových dosiek dohromady. Mikrokontrolér na vývojovej doske je možné resetovať tlačidlom RESET.

Výrobca NXP poskytuje pre vývoj softvéru program CodeWarrior s vlastným prekladačom pre jazyk symbolických a adries a jazyky C a C++. Vývoj aplikácií

a ich ladenie je možné cez BDM[1] rozhranie, ktorým doska disponuje [6].

2.2.2 Súčasné laboratórne úlohy

Výuka laboratórnych cvičení je rozdelená do troch častí. Najprv si študenti vyskúšajú prácu so strojovým kódom, následne s jazykom symbolických adries a predmet je zavŕšený prácou v jazyku C.

Na prvých cvičeniach sa zopakujú základné informácie z výpočtovej techniky ako napríklad vyjadrenie čísel v dvojkovej a šestnástkovej (hexadecimálnej) sústave a výpočet druhého doplnku. Aby mohli pracovať s vývojovou doskou, najprv sa musia zoznámiť s vývojovým prostredím NXP CodeWarrior a hneď si vyskúšajú písanie programu na sčítanie dvoch čísel v strojovom kóde.

V ďalších cvičeniach sa prechádza k jazyku symbolických adries, do ktorého sa prepíše program v strojovom kóde z predošlých cvičení. Študenti si vyskúšajú rôzne adresovacie módy použitého mikrokontroléra a začnú využívať testovanie podmienok a inštrukcie vetvenia. S týmito znalosťami si tak môžu vyskúšať programovanie cyklov a prácu s poľom čísel. Ďalej sa naučia používať podprogramy a predávanie parametrov týmto podprogramom. Doposiaľ všetku prácu vykonávajú za použitia emulátora mikrokontroléra vo vývojovom prostredí a výsledky si kontrolujú pomocou odladzovacích nástrojov. V ďalších cvičeniach si však odskúšajú už prácu s vývojovou doskou a vyskúšajú si naprogramovať binárne vstupy a výstupy a taktiež definovanie a obsluhu prerušení. Táto časť cvičení končí prvým kontrolným testom, v ktorom si žiaci overia svoje získané znalosti v oblasti programovania mikrokontrolérov v jazyku symbolických adries.

V poslednej časti laboratórnych cvičení študenti pokračujú prácou v jazyku C, do ktorého si najskôr prepíšu programy na obsluhu binárnych vstupov a výstupov a prácu s prerušeniami z jazyka symbolických adries. Následne sa naučia využívať niektoré periférie použitého mikrokontroléra, ako sú napríklad hodiny reálneho času, funkcie časovača output compare pre generovanie časového intervalu a input capture pre meranie časového intervalu. Na posledných cvičeniach si ešte vyskúšajú generovanie PWM signálu pre ovládanie jasu LED diódy. Cvičenia končia druhým kontrolným testom, na ktorom si študenti vyskúšajú naprogramovať komplexnejšiu úlohu využívajúcu periférie, s ktorými sa počas semestra naučili pracovať [22].

[1]Background Debug Mode (BDM) je elektronické rozhranie používané v mnohých NXP produktoch, ktoré umožňuje ladenie embedded systémov [45].

2.3 Inovácia laboratórnych cvičení

Aby boli študenti po absolvovaní študijného programu, čo najviac prispôsobení trhu, ktorý napreduje veľmi veľkým tempom, predmet si vyžaduje inováciu učebných pomôcok a príslušných laboratórnych úloh. Mikrokontroléry z rodiny HCS08 majú 8-bitovú architektúru, pričom v praxi sa už bežne využívajú viacjadrové mikrokontroléry s 32-bitovou architektúrou. Tie navyše disponujú viacerými perifériami ako je napríklad DMA, USB alebo PLL. Taktiež sa čím ďalej, tým viac využíva architektúra ARM, ktorá má výrazný podiel na trhu. Mikrokontroléry HCS08 nájdu svoje využitie v mnohých oblastiach, avšak kvôli spomenutým skutočnostiam tieto mikrokontroléry nereprezentujú súčasný vývoj vo svojej oblasti, čím vzniká potenciál pre využitie vývojovej dosky Raspberry Pi Pico.

2.3.1 Návrh úloh pre vývojový kit Raspberry Pi Pico

S využitím vývojovej dosky vo výuke predmetu Vstavané systémy a mikroprocesory prichádza príležitosť na zmenu zadaní laboratórnych úloh, tak aby čo najviac využili periférie vývojovej dosky. V tejto časti budú zvolené úlohy, ktoré je možné realizovať na vývojovej doske. Tieto úlohy však musia byť v súlade s náplňou predmetu a aby boli realizovateľné na laboratórnych cvičeniach, musia sa dať vypracovať v rámci jedného laboratórneho cvičenia, ktoré trvá tri hodiny.

Úloha č. 1: Zoznámenie sa s inštrukčnou sadou

Mikrokontrolér RP2040 využíva inštrukčnú sadu ARMv6-M, a preto prvá úloha je venovaná zoznámeniu sa s týmito inštrukciami. Študenti si naprogramujú v jazyku symbolických adries program, ktorý bude pomocou naprogramovaných podprogramov vyhľadávať minimálne a maximálne číslo v poli. Volanie funkcií a získavanie návratových hodnôt bude realizované tak, aby boli dodržané volacie konvencie jazyka C.

Úloha č. 2: Práca s binárnymi vstupmi a výstupmi

Medzi prvotné úlohy, ktoré skúša každý programátor pri zoznamovaní sa s novým mikrokontrolérom je práca s binárnymi vstupmi a výstupmi. Táto úloha slúži na oboznámenie sa so spôsobom inicializácie GPIO, kde patrí napríklad zvolenie, či daný vývod bude vstup alebo výstup, či sa má použiť vnútorný pull-up/pull-down rezistor pre vstup, alebo aké prúdové obmedzenie má mať výstup. Samozrejmosťou bude prečítanie hodnoty zo vstupu a zapísanie na výstup.

Úloha č. 3: Obsluha časovača a práca s prerušeniami

Práca s časom a oneskorením je taktiež veľmi podstatnou súčasťou programovania mikrokontrolérov. Vytváranie oneskorení umožňuje systémový zdroj hodín a časovače v podobe alarmov, ktoré vedia spustiť určitú akciu pri dosiahnutí daného času. Aby mohol mikrokontrolér vykonávať viac úloh súčasne a nemusel byť neustále zaneprázdnený dopytovaním sa na určitý stav, je nevyhnutné použiť prerušenia. Ich fungovanie bude demonštrované na periférii alarmu, ktorý vie pravidelne s istou časovou periódou generovať prerušenie. Študenti si vyskúšajú rôzne spôsoby vytvárania oneskorenia v programe, vďaka čomu môžu blikať LED diódou.

Úloha č. 4: Práca s perifériou - A/D prevodník

Jednou zo základných periférií, ktoré sa veľmi často používajú vo svete mikrokontrolérov je A/D prevodník. Ten umožňuje získavať analógové hodnoty z rôznych snímačov, čo je základnou požiadavkou na vznik regulačného obvodu. Študenti sa v tejto úlohe oboznámia s inicializáciou A/D prevodníka a následne čítaním analógovej hodnoty zo vstupu, na ktorom je pripojený potenciometer. Podľa veľkosti vstupného napätia sa rozsvietia postupne pripojené štyri LED diódy.

Úloha č. 5: Komunikácia mikrokontroléra s okolím pomocou UART

Pri programovaní mikrokontrolérov je často potrebné vypisovať na sériový port správy, a tak odladzovať vyvíjaný program, pretože debugger nie je vždy k dispozícii. Rovnako sa sériový port využíva na komunikáciu s inými zariadeniami, napríklad čítačkami RFID kariet a mnohými ďalšími. Ide teda o jednu zo základných periférií, ktoré potrebuje programátor využívať. V tejto úlohe je demonštrovaná práca so sériovým UART portom. Najprv sa port vhodne nakonfiguruje a následne prijíma textové správy. Všetky prijaté znaky pošle naspäť na sériový port s tým, že malé písmená program premení na veľké a veľké na malé.

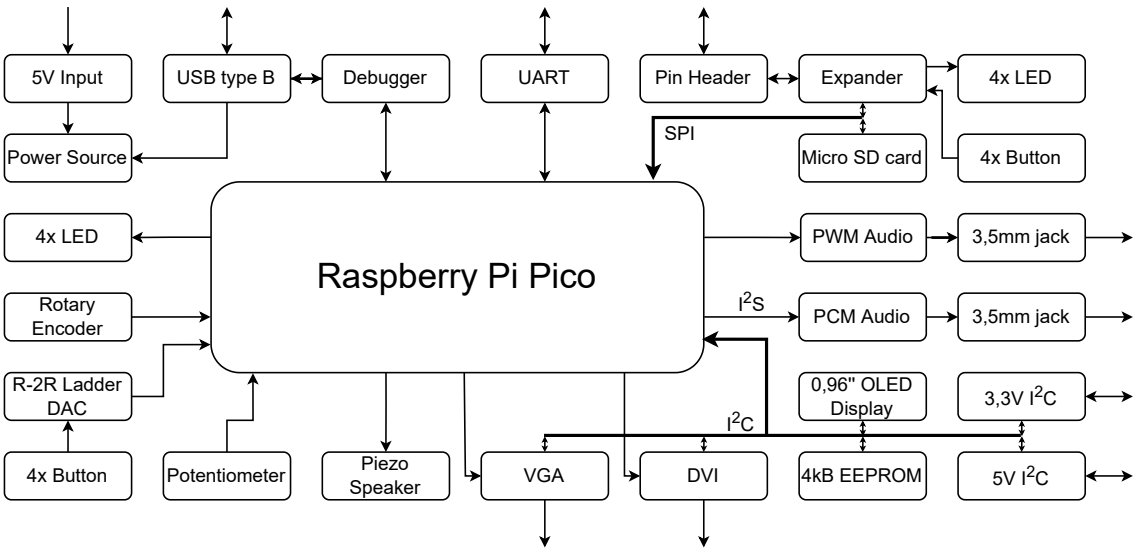
3 Návrh konceptu HW platformy

Aby boli navrhnuté laboratórne úlohy realizovateľné, k vývojovej doske Raspberry Pi Pico je potrebné pripojiť komponenty, ktoré sa v jednotlivých úlohách majú využívať. Ide teda o aspoň štyri LED diódy, jedno tlačidlo, otočný potenciometer a UART výstup. Navyše implementovaný má byť aj zvukový a obrazový výstup. Pre jednoduché odladzovanie programov musí byť k dispozícii debugger. Všetky vyžadované komponenty sa majú nachádzať na jednej HW platforme a doska Raspberry Pi Pico sa má pripájať modulárne. Elektrické pripojenie komponentov k doske Raspberry Pi Pico musí byť vhodne ošetrené, aby sa predišlo potenciálne vzniknutým skratom, ktoré by HW platformu znefunkčnili.

Na trhu existuje viacero rozširujúcich vývojových kitov pre dosku Raspberry Pi Pico od rôznych výrobcov ako je Arduino, Adafruit, Pimoroni či Waveshare. Väčšinou sú to však jednoúčelové dosky, ktoré obsahujú iba niekoľko špecifických modulov ako napríklad OLED displej, Bluetooth a Wi-Fi modul, inerciálnu meraciu jednotku, vývody pre zapojenie snímačov a akčných členov alebo obrazový VGA výstup. Žiadna z týchto dosiek neobsahuje debugger [24].

Uvedeným požiadavkám pre realizáciu laboratórnych úloh nevyhovuje žiaden existujúci vývojový kit. V nasledujúcej časti práce je preto kompletne vytvorený vývojový kit, ktorý bude dané požiadavky spĺňať.

3.1 Bloková schéma zapojenia



Obr. 3.1: Bloková schéma zapojenia HW platformy

Celé zapojenie HW platformy je rozdelené do viacerých blokov, ktoré sú zobrazené na obrázku 3.1. Najpodstatnejšou časťou návrhu je samotná vývojová doska Raspberry Pi Pico s mikrokontrolérom RP2040, ktorá sa ku kitu pripojí modulárne cez dutinkové lišty. Debugger slúži na nahrávanie programu do vývojovej dosky Raspberry Pi Pico a taktiež na jeho odladzovanie. Pre základnú interakciu s užívateľom je využitých dohromady osem LED diód a osem tlačidiel. Ďalej sa využíva otočný potenciometer a rotačný enkóder. Zvukový výstup je realizovaný troma rôznymi spôsobmi. Ide o piezoelektrický menič a generátory zvukového výstupu využívajúce PWM a PCM moduláciu. Najjednoduchší obrazový výstup poskytuje 0,96" OLED displej. Ďalšie obrazové výstupy sú vyvedené cez VGA a DVI rozhranie. Potrebné údaje je možné zaznamenávať do pamäte EEPROM s veľkosťou 4 kB, alebo na externú micro-SD kartu.

V prípade potreby ušetrenia finančných prostriedkov pri prípadnej výrobe viacerých kusov vývojového kitu, nie je nutné osadiť všetky súčasti HW platformy.

3.2 Popis súčastí HW platformy

Táto časť textu prevedie čitateľa návrhom jednotlivých častí vývojového kitu s názvom RPi Pico Kit. Celkovú schému zapojenia, na ktorú sa text odvoláva je možné nájsť v prílohe A. Schéma je vypracovaná v programe KiCad verzie 6.99 a rovnako aj projekt v tomto programe je dostupný v prílohe D.

3.2.1 Napájanie

Vývojový kit si vyžaduje napájanie 5 V, ktoré môže byť realizované tromi spôsobmi. Najčastejšie bude doska napájaná pomocou USB konektora typu B, ktorým sa doska prepojí s počítačom a cez tento port bude aj naprogramovaná za využitia SWD rozhrania. Týmto spôsobom môže byť doska aj odladzovaná. Ďalšou možnosťou ako napájať dosku je pripojiť ju ku počítaču cez micro-USB konektor typu B, ktorý je osadený na vývojovej doske Raspberry Pi Pico. Takto je možné dosku naprogramovať použitím `.UF2` súboru. V tomto prípade sa však doska nedá odladzovať. Tretia možnosť napájania je cez napájací DC jack rovnako napätím 5 V, čo sa využije v prípade použitia vývojového kitu v konečnej aplikácii.

Ochranu proti nadprúdu pri napájaní z USB zbernice alebo z externého zdroja pripojeného cez DC jack zabezpečujú polymérové PTC poistky F1 a F2 s menovitou hodnotou 500 mA.

Všetky tieto napájania sú v schéme privedené na bod +5 V cez Schottkyho diódu, čo zabezpečí ochranu v prípade súčasného použitia viacerých zdrojov napájania pred tečením spätného prúdu do daného zdroja napájania. Schottkyho dióda taktiež slúži

ako ochrana proti prepólovaniu napájacieho napätia. Napájanie signalizuje zelená LED dióda s označením LED9.

Väčšina použitých komponentov vyžaduje napájacie napätie 3,3 V. To je generované zo vstupného napätia 5 V napäťovým stabilizátorom U2 s maximálnym prúdom 1,5 A. Integrované obvody pre generovanie zvuku majú dedikovaný zdroj napájania 3,3 V, ktoré zabezpečuje stabilizátor U9 s maximálnym prúdom 300 mA. Mikrokontrolér vývojovej dosky je napájaný vlastným impulzným napájacím zdrojom, ktorý je napojený na napätie 5 V.

3.2.2 Debugger

Ako debugger je možné použiť ďalšiu vývojovú dosku Raspberry Pi Pico s nahraným programom PicoProbe a vhodným zapojením, ktoré je bližšie popísané v [10]. V tomto zapojení je však namiesto vývojovej dosky použitý iba samotný mikrokontrolér RP2040, čo umožní menšie rozmery výsledného vývojového kitu. Mikrokontrolér sa ku doske Raspberry Pi Pico napojí cez SWD rozhranie, ktoré slúži na nahrávanie a odladzovanie programu. Zapojenie vychádza z požadovaného minimálneho zapojenia uvedeného v [13].

K mikrokontroléru je pripojená flash pamäť o veľkosti 2 MB cez QSPI rozhranie a 12 MHz kryštál so zaťažovými kondenzátormi s hodnotou 12 pF. Spoľahlivé vyhladenie napájacieho napätia zabezpečujú blokovacie kondenzátory C4 až C14. Fyzickú vrstvu USB tvoria dva 27 Ω rezistory zapojené na diferenciálnych signáloch D+ a D–.

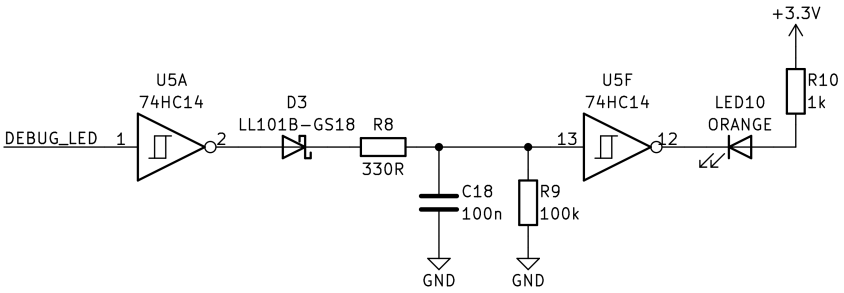
Debugger a vývojová doska nie je napájaná rovnakým zdrojom, a preto medzi napätím logických úrovní môže byť malý rozdiel, čo by viedlo ku vzniku určitých vyrovnávacích prúdov. Aby sa tomu predišlo, medzi tieto logické úrovne je vložený rezistor s hodnotou 1 kΩ, čo obmedzí tento prúd.

K debuggeru je pripojený aj UART port vývojovej dosky, vďaka čomu je možné komunikovať s doskou prostredníctvom sériového portu cez rovnaké USB rozhranie, ktoré slúži na napájanie, nahrávanie programu a odladzovanie.

Pre činnosť debuggera je nevyhnutné nahrať do flash pamäte program PicoProbe cez USB rozhranie. Aby nedošlo k jeho náhodnému prepísaniu, pre nahratie je potrebné zoskratovať prepojku JP1 v čase resetu mikrokontroléra.

Debugger má vyvedené vlastné SWD rozhranie a UART port vo forme testovacích plôšok. To je možné využiť pre odladzovanie samotného programu PicoProbe.

Program PicoProbe pri svojej činnosti vysiela na vývod GPIO25 krátke impulzy s úrovňou *log. 0*. Aby bolo tieto impulzy možné pozorovať, na vývod je pripojená oranžovaná LED dióda LED10. Časová dĺžka impulzov je veľmi krátka a bliknutie LED diódou by nebolo pozorovateľné. Tento problém je vyriešený tým, že tieto



Obr. 3.2: Zapojenie signalizačnej LED debuggera [50]

impulzy sú predĺžené pomocou monostabilného klopného obvodu (obr. 3.2) a až tak sú privedené na LED diódu. Vo výsledku sa to prejaví tým, že vstupný impulz s úrovňou *log. 0* rozsvieti LED diódu na čas τ. Dĺžku času τ definuje v tomto prípade rezistor R9 a kondenzátor C18, čo znamená, že pre tieto hodnoty platí $\tau = 10\,\text{ms}$ (vzťah 3.1).

$$\tau = R \cdot C = (100 \cdot 10^3) \cdot (100 \cdot 10^{-9}) = 10ms \tag{3.1}$$

Monostabilný obvod je tvorený z dvoch invertujúcich Schmittových klopných obvodov. Na vstup prvého z nich je privedený signál z debuggera DEBUG_LED. Vysoký vstupný odpor Schmittovho klopného obvodu zaisťuje integritu vstupného signálu. Schottkyho dióda D3 zabezpečuje, že obvod reaguje iba na závernú hranu. Rezistor R8 obmedzuje nabíjací prúd kondenzátora C18. Výstup druhého klopného obvodu spína napätia na katóde LED diódy LED10. Schéma je prevzatá z [50] a ďalej upravená.

3.2.3 Základné užívateľské rozhranie

Medzi základné užívateľské rozhranie patria štyri LED diódy, rotačný enkóder, otočný potenciometer a štyri tlačidlá pripojené na analógový vstup.

LED diódy

Štyri LED diódy červenej farby LED1 až LED4 ako základné zobrazovacie prvky sú pripojené na výstupy GPIO6 až GPIO9. Ako bude ďalej spomenuté, na tieto výstupy sú pripojené aj iné komponenty. Aby v prípade ich využívania nedochádzalo k prúdovému zaťažovaniu výstupov LED diódami a teda poškodeniu ich integrity, diódy sú ovládané MOSFET tranzistormi s kanálom typu N, ktoré majú veľmi nízky odpor $R_{DS_{(on)}}$, a preto majú malé saturačné napätie U_{DS}.

Rotačný enkóder

Vývody rotačného enkódera A a B sú pripojené na vstup GPIO21 a GPIO22. Tlačidlo enkódera je na vstupe GPIO11. Pre správnu funkciu je potrebné softvérovo pripojiť na tieto vývody pull-down rezistory. Všetky tri vývody sú ku GPIO pripojené cez rezistor o hodnote 1 kΩ, aby sa predišlo skratu. Ten by mohol nastať v prípade, že by vývody boli nastavené ako výstupy a v čase stlačenia tlačidla by bola na výstupe *log. 0*.

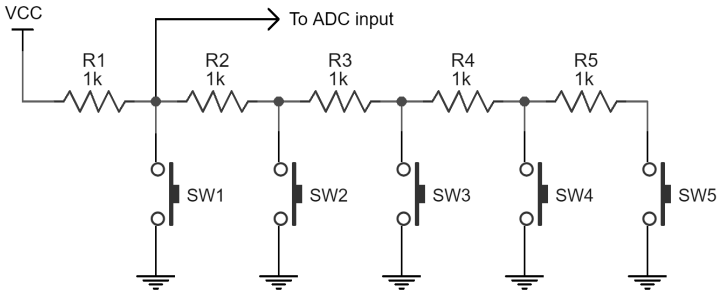
Otočný potenciometer

Otočný potenciometer je pripojený k pinu GPIO27, na ktorom je analógový vstup prevodníka ADC1. Vyfiltrovanie vstupného signálu zabezpečuje filter 1. rádu typu dolná priepusť, ktorý tvoria 100 Ω rezistor a 10 nF kondenzátor a teda časová konštanta filtra τ je 1 µs.

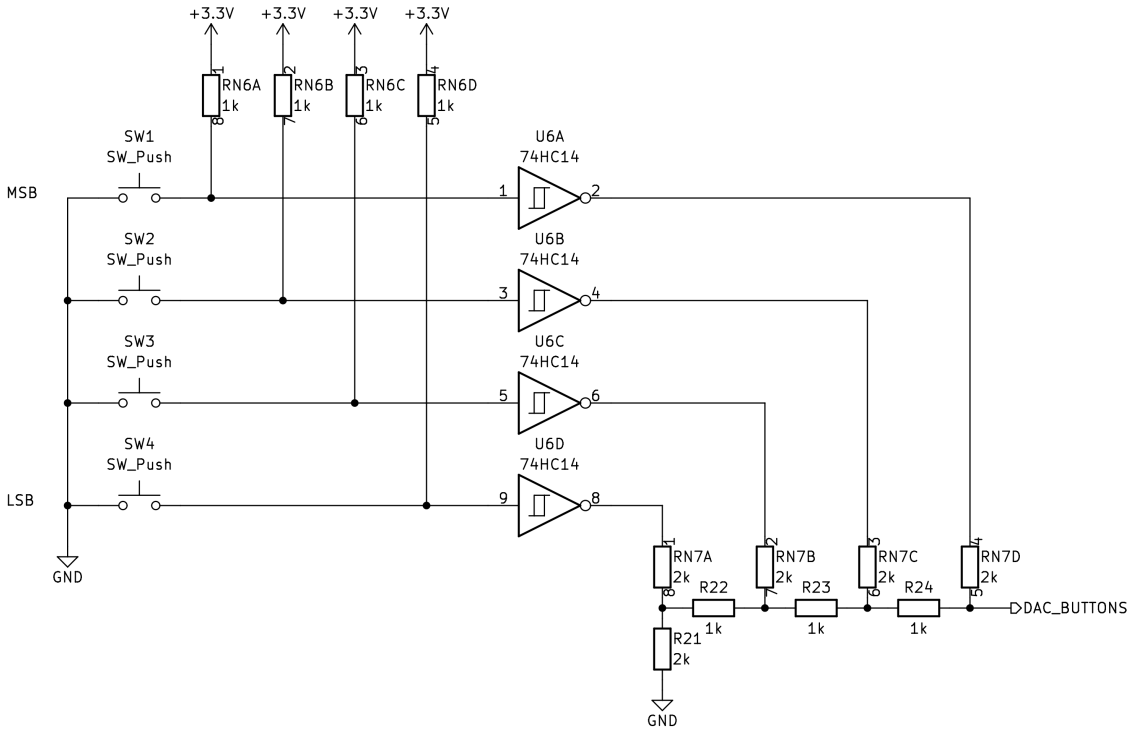
$$\tau = R \cdot C = 100 \cdot (10 \cdot 10^{-9}) = 1\mu s \tag{3.2}$$

Tlačidlá pripojené na analógový vstup

Pripojenie štyroch tlačidiel priamo na digitálne vstupy by bolo na úkor iných súčastí vývojového kitu, ktoré by tak nemohli byť použité. Bežne sa v praxi používa zapojenie viacerých tlačidiel na jeden analógový vstup pomocou jednoduchej rezistorovej siete (obr. 3.3). Stlačenie tlačidla vygeneruje na výstupe napätie, ktoré prináleží stlačeniu daného tlačidla. Hlavnou nevýhodou tohto zapojenia je nemožnosť detekcie stlačenia viacerých tlačidiel naraz. V prípade, že je stlačené tlačidlo SW2 a súčasne SW4, program bude detekovať, že je stlačené tlačidlo SW2 a podobne.



Obr. 3.3: Tlačidlá pripojené na analógový vstup pomocou rezistorovej siete [25]



Obr. 3.4: Tlačidlá pripojené na analógový vstup pomocou R-2R rezistorovej siete

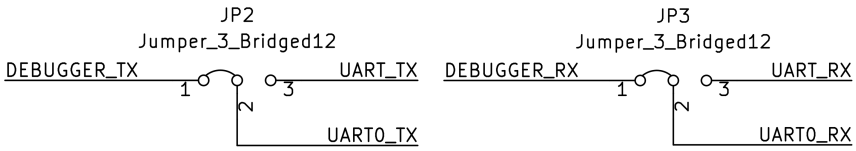
Problém predošlého zapojenia je vyriešený použitím jednoduchého 4-bitového D/A prevodníka zhotoveného z R-2R rezistorovej siete (obr. 3.4), čo umožňuje detekciu akejkoľvek kombinácie stlačených tlačidiel. Zapojenie však vyžaduje na svojich vstupoch priame napojenie na 0 alebo 5 V. Aby takéto zapojenie bolo možné použiť s tlačidlami, ktoré takýto výstup negenerujú kvôli použitému pull-up rezistoru, tlačidlá SW1 až SW4 sú pripojené k rezistorovej sieti pomocou Schmittovho klopného obvodu. Napäťové príspevky od jednotlivých stlačených tlačidiel sa sčítajú a výstup tohto D/A prevodníka je privedený na analógový vstup GPIO28. Hodnota výstupného napätia $U_{OUT}[V]$ je daná vzťahom 3.3 [48], pričom referenčné napätie U_{REF} predstavuje hodnotu napätia *log. 1*, čiže 3,3 V, N predstavuje počet bitov D/A prevodníka, čiže 4 a x je binárne kódovaný stav tlačidiel.

$$U_{OUT} = U_{REF} \cdot \frac{x}{2^N} \tag{3.3}$$

Zmena stavu tlačidiel sa prejavuje na najvyšších štyroch bitoch 12-bitového A/D prevodníka, ktorého efektívny počet bitov $ENOB$ má hodnotu $ENOB = 8,7$. Užitočný signál je teda dostatočne vzdialený od šumu, a preto je zapojenie pre štyri tlačidlá dostatočne spoľahlivé.

3.2.4 UART rozhranie

Komunikáciu na UART rozhraní signalizujú oranžové LED diódy, LED11 pre vysielanie správ a LED12 pre prijímanie správ. Diódy sú pripojené rovnakým spôsobom ako signalizačná LED dióda debuggera (obr. 3.2). Vďaka tomu sa dá okom zachytiť aj občasná krátka komunikácia na vysokej prenosovej rýchlosti, čo by inak nebolo možné. Zapojenie taktiež zachováva integritu signálov RX a TX. Pre komunikáciu s ďalšími zariadeniami prostredníctvom UART rozhrania je na kite vyvedený port J3 v podobe pinovej lišty. Kolíkové prepojky JP2 a JP3 určujú (obr. 3.5), či sa UART port mikrokontroléra pripojí ku debuggeru alebo k externému zariadeniu cez port J3.



Obr. 3.5: Voľba pripojenia UART portu k debuggeru alebo k externému zariadeniu

3.2.5 SPI zariadenia

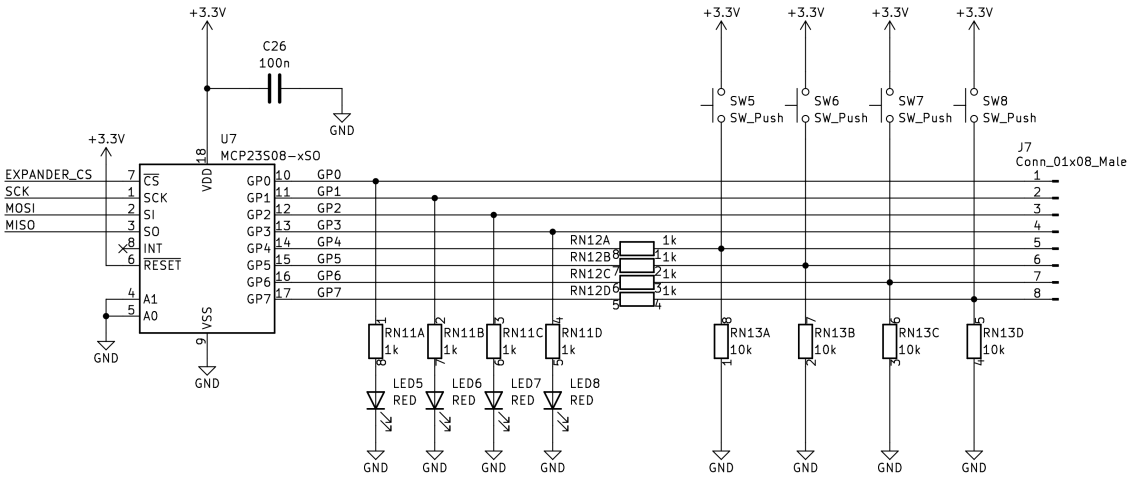
Na zbernici SPI je pripojený expander binárnych vstupov a výstupov a slot pre micro-SD kartu.

Expander binárnych vstupov a výstupov

V počiatočnom návrhu vývojového kitu bol ako expander binárnych vstupov na zbernici SPI použitý posuvný register 74LS165 a posuvný register 74LS595 ako expander binárnych výstupov. V rámci laboratórnych cvičení by bolo na týchto posuvných registroch jednoduché vysvetliť princíp fungovania SPI zbernice vzhľadom na to, že nepoužívajú žiaden komunikačný protokol. Tieto posuvné registre však nie sú plne kompatibilné so zbernicou SPI, pretože výstup MISO posuvného registra 74LS165 nie je trojstavový. Použitie ďalších zariadení na SPI zbernici by tak nebolo možné.

Ako expandér binárnych vstupov a výstupov na SPI zbernici je vo finálnej verzii použitý integrovaný obvod MCP23S18, ktorý poskytuje osem vývodov, ktoré môžu byť použité ako vstupy aj ako výstupy. Obvod taktiež obsahuje interné pull-up rezistory. Tie sa dajú pripájať zvlášť pre každý zo vstupov.

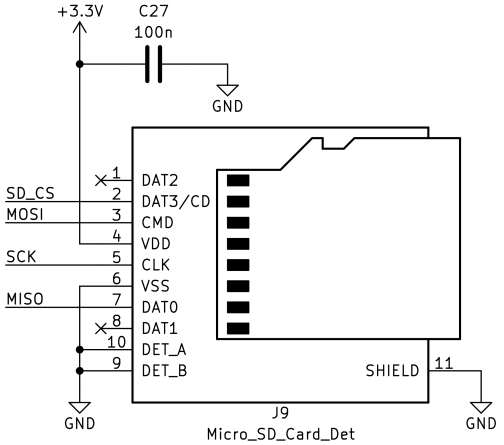
Expander disponuje dvoma vstupmi pre nastavenie adresy, čo znamená, že na SPI zbernici je možné použiť až štyri takéto obvody na jednom CS pine, ktoré sa budú líšiť svojou adresou.



Obr. 3.6: Zapojenie expandera binárnych vstupov a výstupov na SPI zbernici

K expanderu sú pripojené štyri LED diódy LED5 až LED8 na výstupoch GP0 až GP3 a štyri tlačidlá SW5 až SW8 spolu s pull-down rezistormi a ochranou proti skratu na vstupy GP4 až GP7 (obr. 3.6). Všetky vývody expandera sú zaroveň vyvedené spolu s napájaním na trojradovú pinovú lištu pre pripojenie ďalších komponentov, ktoré nebudú kolidovať s použitými LED diódami a tlačidlami. Povoľovací pin CS je pripojený ku pinu GPIO26.

SD karta



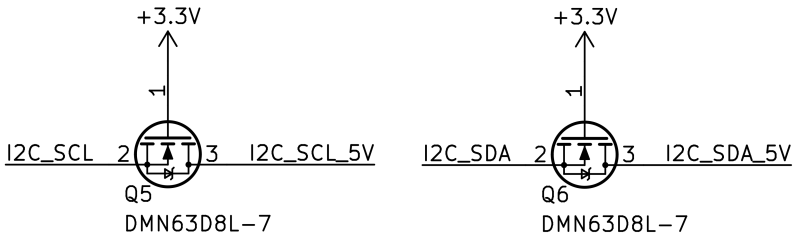
Obr. 3.7: Zapojenie slotu pre micro-SD kartu

SD karta sa správa ako SPI zariadenie. Pristupovať k nej je možné v 1-bitovom alebo 4-bitovom režime. Pre ušetrenie počtu použitých vývodov na vývojovej do-

ske Raspberry Pi Pico je použitý 1-bitový režim (obr. 3.7). Povoľovací pin CS je pripojený ku pinu GPIO5.

3.2.6 I²C zariadenia

Vývojový kit obsahuje niekoľko I²C zariadení. Ako pull-up rezistory pre vodiče SDA a SCL sú použité rezistory s odporom 4,7 kΩ. Tieto vodiče sú pripojené na vývody GPIO2 a GPIO3. Niektoré zariadenia na zbernici I²C pracujú s 3,3 V logikou a niektoré s 5 V logikou. Prevod signálov medzi logickými úrovňami zabezpečuje prevodník logických úrovní zhotovený z MOSFET tranzistorov Q5 a Q6 s kanálom typu N (obr. 3.8).



Obr. 3.8: Prevodník logických úrovní I²C zbernice

Vývojový kit poskytuje dva porty v podobe pinovej lišty pre pripojenie ďalších zariadení ku I²C zbernici. Port s označením J11 pracuje s 3,3 V logikou a port J12 s 5 V logikou.

OLED Displej

Na základné zobrazovanie dát je využitý 0,96" OLED displej s rozlíšením 128×64 pixelov. Displej nepotrebuje žiadne podsvietenie, keďže zdrojom svetla sú samotné pixely. O zobrazovanie a komunikáciu s I²C zbernicou sa stará vstavaný ovládač SSD1306. Napájacie napätie displeja je v rozmedzí 3,3 V až 5 V a jeho prúdový odber je maximálne 80 mA. Na zbernici I²C má pridelenú adresu 0x3C [51].

EEPROM

Pre uchovávanie konfiguračných dát je na vývojom kite osadená EEPROM pamäť pripojená ku I²C zbernici. Jej veľkosť je 4 kB a na zbernici má pridelenú adresu 0x51.

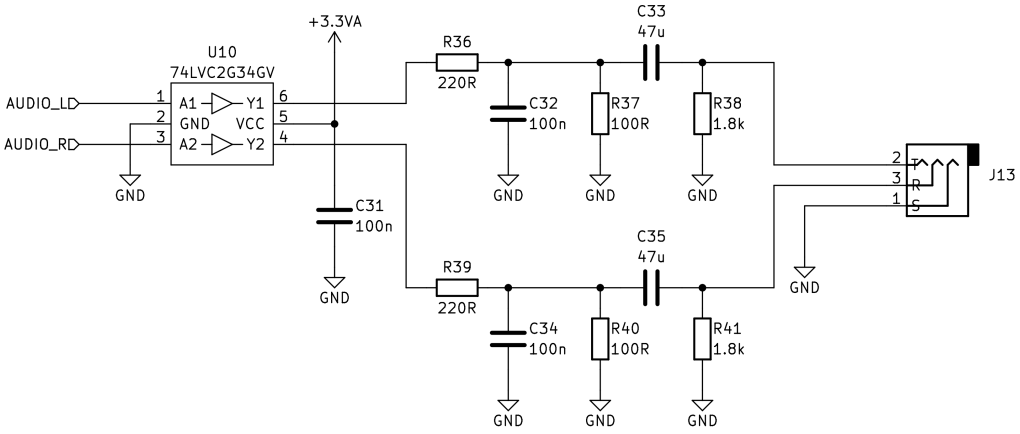
3.2.7 Zvukový výstup

Zvukový výstup je realizovaný tromi spôsobmi, v nasledujúcom texte bude popísaný každý z týchto spôsobov.

Piezoelektrický menič

Najjednoduchší zvukový výstup je generovaný pomocou pasívneho piezoelektrického meniča BZ1 pripojeného k vývodu GPIO10. Piezomenič je spínaný cez MOSFET tranzistor Q7 s kanálom typu N. Aby ho bolo možné jednoducho vypnúť, hradlo tranzistora je pripojené cez kolíkovú prepojku JP14.

PWM generátor zvuku

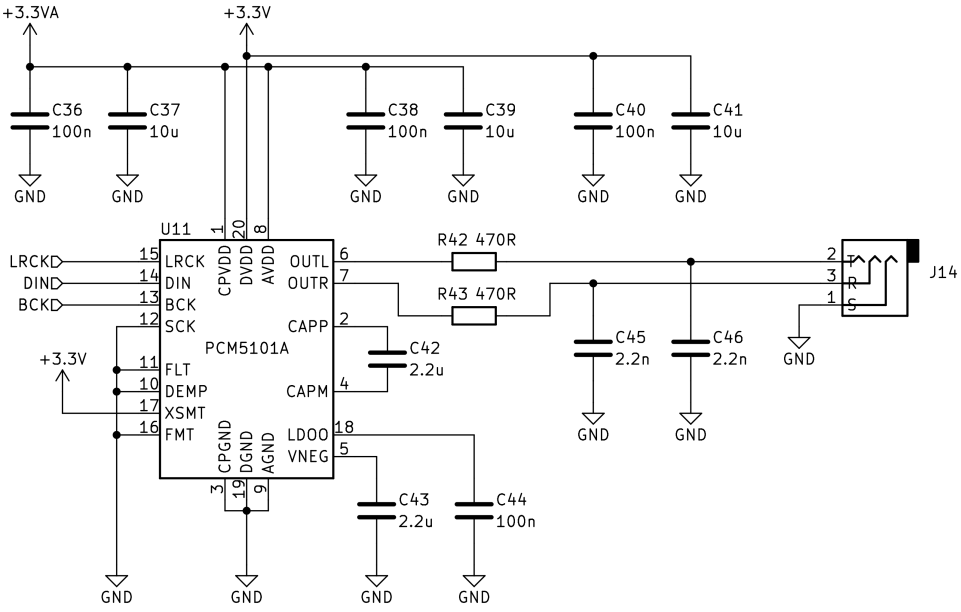


Obr. 3.9: Zapojenie PWM generátora zvukového výstupu

Druhá metóda generuje dvojkanálový stereo zvuk pomocou PWM modulácie (obr. 3.9). Digitálny PWM výstup z vývodov GPIO8 a GPIO9 je privedený do oddeľovacieho buffera U10, ktorý je napájaný dedikovaným stabilizátorom U9, aby výstupný zvuk bol čo najmenej zarušený digitálnymi obvodmi. Výstup buffera je cez filter a striedavú väzbu pripojený k výstupnému 3,5 mm jacku. Schéma zapojenia je prevzatá z [13].

PCM generátor zvuku

Treťou možnosťou je zvukový výstup generovaný PCM moduláciou využitím integrovaného obvodu PCM5101A (obr. 3.10). Zvuk kódovaný PCM moduláciou je privedený do integrovaného obvodu pomocou protokolu I²S za využitia vývodov GPIO8 až GPIO10. Výstupom je dekódovaný zvuk privedený na 3,5 mm jack. Okolité súčiastky vychádzajú z typického zapojenia obvodu. Schéma zapojenia je prevzatá z [13].



Obr. 3.10: Zapojenie PCM generátora zvukového výstupu

3.2.8 Obrazový výstup

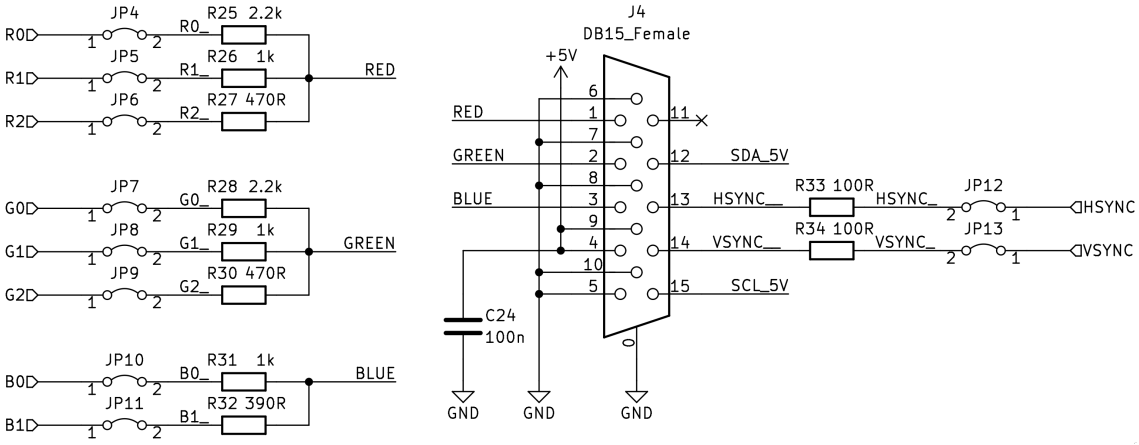
Obrazový výstup je vyvedený cez VGA a DVI rozhranie.

VGA rozhranie

Výstup farebného signálu zabezpečuje 8 binárnych výstupov, ktoré musia byť pripojené k doske Raspberry Pi Pico na osem za sebou idúcich výstupov, pretože tak si to vyžaduje použitá softvérová knižnica [29]. Na výstupy GPIO11 až GPIO18 sú pripojené rezistory R25 až R32, ktoré slúžia na naváhovanie jednotlivých bitov farebných zložiek. Rozlíšenie jednotlivých farieb je nasledovné:

- červená zložka 3 bity (R0 až R2)
- zelená zložka 3 bity (G0 až G2)
- modrá zložka 2 bity (B0 až B1)

Bity R0, G0 a B0 sú bity s najnižším významom (LSB). Použitie 8-bitového výstupu zaistí malé nároky na pamäť RAM. Jednotlivé farebné výstupy sú pripojené ku VGA konektoru (obr. 3.11). V zobrazovacom zariadení sa nachádza pre každú farebnú zložku ukončovací rezistor s odporom 75 Ω. Vhodnou voľbou hodnôt váhovacích rezistorov sa docieli potrebný výstup farebnej zložky v rozmedzí 0 až 0,7 V. Horizontálny synchronizačný signál HSYNC a vertikálny VSYNC je generovaný na výstupoch GPIO19 a GPIO20. Schéma zapojenia VGA rozhrania je prevzatá z [29].



Obr. 3.11: Zapojenie obrazového VGA výstupu

Pre čítanie EDID metadát[1] je VGA rozhranie pripojené ku I²C zbernici, kde displej zaberá adresy 0x37, 0x49, 0x50 a 0x59. Z týchto adries je pre EDID metadáta relevantná iba adresa 0x50.

Pripojenie VGA konektora ku I²C zbernici je prevzaté z [52]. Signál SDA je pripojený k vývodu č. 12, SCL ku vývodu číslo 15 a na vývody č. 4 a 9 je pripojené napájacie napätie 5 V. Ako bude neskôr spomenuté, toto zapojenie sa pri testovaní finálnej realizácii ukázalo ako nesprávne, pretože externý monitor má na vývode č. 4 napojenú zem, čo vytvára skrat medzi napájaním 5 V a zemou.

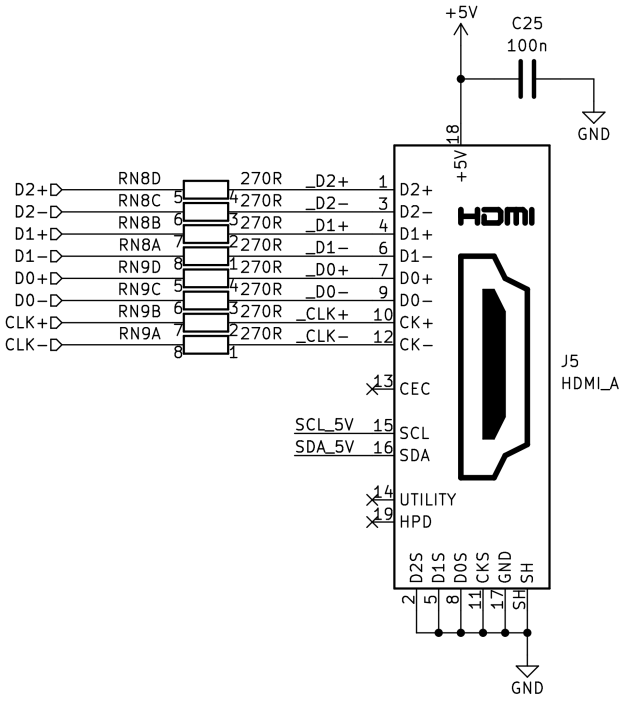
DVI rozhranie

Digitálny obrazový výstup je realizovaný prostredníctvom DVI rozhrania. Výstupom je konektor HDMI, ktorý je možné využiť, pretože HDMI je spätne kompatibilné s DVI rozhraním. HDMI rozhranie však nie je možné použiť, pretože ide o proprietárne riešenie, čiže k jeho použitiu by bola potrebná príslušná licencia.

DVI signál sa skladá z troch dátových diferenciálnych výstupov D0 až D2 a jedného taktovacieho diferenciálneho výstupu CLK. Tieto signály sú pripojené cez 270 Ω rezistory k vývodom GPIO12 až GPIO19 (obr. 3.12), tak ako si to vyžaduje softvérová knižnica [35].

Rozhranie DVI je taktiež pripojené k zbernici I²C, kde pripojený displej zaberá adresy 0x3A, 0x50, 0x55 a 0x5A. Z týchto adries je pre čítanie EDID metadát relevantná iba adresa 0x50.

[1]Extended Display Identification Data (EDID) v preklade rozšírené identifikačné dáta displeja popisujú parametre zobrazovacej jednotky, ako napríklad výrobné číslo, veľkosť displeja, podporované rozlíšenia, podporované časovanie signálov či mnohé iné [26].



Obr. 3.12: Zapojenie obrazového DVI výstupu

3.3 Rozloženie vývodov dosky Raspberry Pi Pico

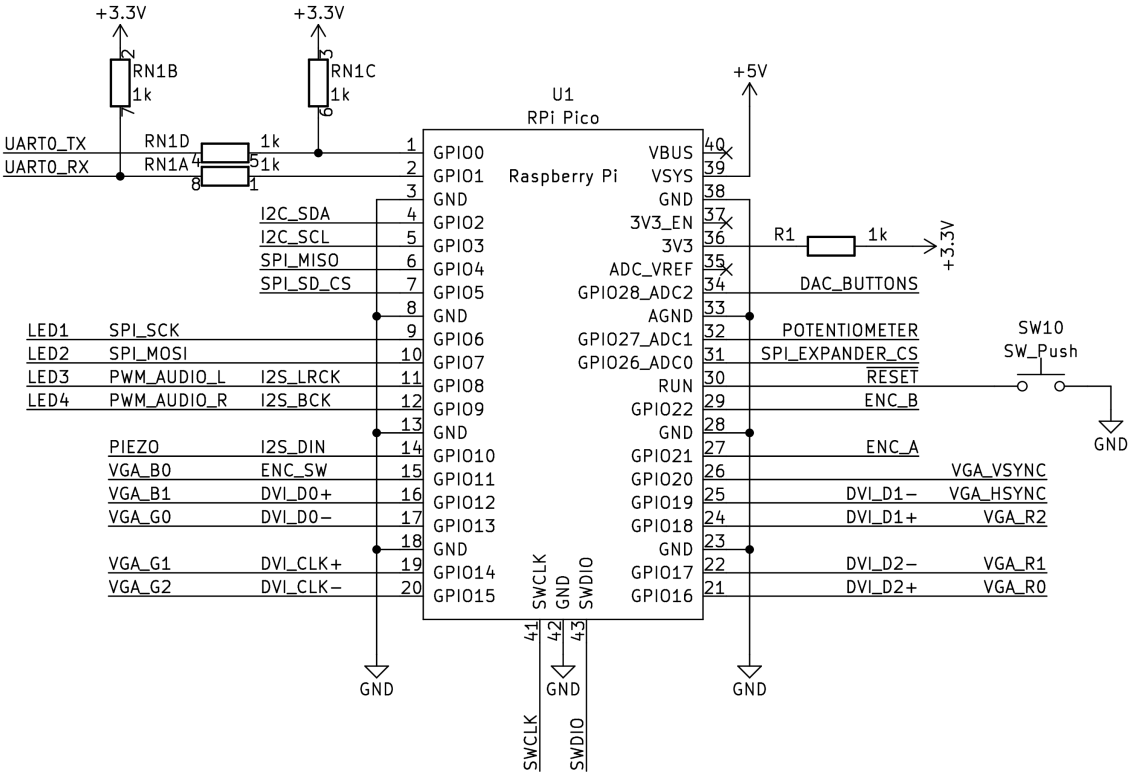
Súčasti HW platformy uvedené v predošlej časti sú pripojené k vývodom vývojovej dosky Raspberry Pi Pico (obr. 3.13). Rozloženie zapojenia sumarizuje tabuľka 3.1.

Doska Raspberry Pi Pico neposkytuje dostatok vývodov pre všetky navrhnuté súčasti. Niektoré vývody dosky sú preto použité viacnásobne. Kombinácia jednotlivých funkcií vývodu je vždy zvolená tak, aby sa dané funkcie vzájomne neovplyvňovali a ich súčasné použitie sa vzájomne vylučovalo. To znamená, že buď sa použije PWM alebo PCM zvukový výstup, nie však obe naraz a podobne.

Rozhrania VGA a DVI sú pripojené na rovnaké vývody, čo vylučuje ich súčasné použitie. Zapojenie VGA by však ovplyvňovalo elektrické zapojenie DVI rozhrania. VGA rozhranie je preto pripojené cez kolíkové prepojky JP4 až JP13 (obr. 3.11).

Reset

Ako bolo spomenuté v časti 1.3.1, Raspberry Pi Pico nedisponuje resetovacím tlačidlom, preto bolo externe pridané pripojením na pin RUN. Stlačením tohto tlačidla s označením SW10 sa prepojí pin RUN so zemou a mikrokontrolér sa resetuje. Zapojenie si nevyžaduje pull-up rezistor, ktorý definuje logickú úroveň na pine v prípade nestlačeného tlačidla, lebo mikrokontrolér vnútorne obsahuje takýto pull-up rezistor.



Obr. 3.13: Zapojenie vývodov vývojovej dosky Raspberry Pi Pico

Tab. 3.1: Rozloženie jednotlivých vývodov vývojovej dosky Raspberry Pi Pico

Vývod	**Funkcia vývodu**	**Vývod**	**Funkcia vývodu**
GPIO0	UART0 TX	GPIO13	VGA G0, DVI D0–
GPIO1	UART0 RX	GPIO14	VGA G1, DVI CLK+
GPIO2	I²C SDA	GPIO15	VGA G2, DVI CLK–
GPIO3	I²C SCL	GPIO16	VGA R0, DVI D2+
GPIO4	SPI MISO	GPIO17	VGA R1, DVI D2–
GPIO5	SPI SD CS	GPIO18	VGA R2, DVI D1+
GPIO6	LED1, SPI SCK	GPIO19	VGA HSYNC, DVI D1–
GPIO7	LED2, SPI MOSI	GPIO20	VGA VSYNC
GPIO8	LED3, PWM AUDIO L, I²S LRCK	GPIO21	ENC A
GPIO9	LED4, PWM AUDIO R, I²S BCK	GPIO22	ENC B
GPIO10	PIEZO, I²S DIN	GPIO26	SPI EXPANDER CS
GPIO11	VGA B0, ENC SW	GPIO27	POTENTIOMETER
GPIO12	VGA B1, DVI D0+	GPIO28	DAC BUTTONS

4 Otestovanie konceptu HW platformy

Otestovanie navrhnutej platformy bolo zrealizované na kontaktnom poli. Základným nástrojom pre testovanie je debugger, ktorý je pre účely testovania zrealizovaný za pomoci ďalšej vývojovej dosky a do nej je nahraný program PicoProbe. Obrázok 1.9 na strane č. 33 zobrazuje ako sú tieto dosky prepojené.

Testovanie prebiehalo postupne po jednotlivých HW súčastiach. K programovanej doske boli preto postupne pripájané všetky potrebné komponenty pre testovanie danej súčasti podľa schémy zapojenia uvedenej v prílohe A. Pre testovanie boli použité výlučne THT súčiastky, ktorých hodnoty sú uvedené priamo v schéme zapojenia alebo v prílohe B.

4.1 Popis testovacích programov

V tejto časti je uvedený stručný popis testovacích programov. Väčšina z nich je k dispozícii v elektronickej prílohe F v priečinku `pico-kit-examples`, ak nie je uvedené inak. Jednotlivé programy pre ich jednoduchosť používajú C/C++ SDK a nepracujú priamo s registrami.

Pre overenie správnosti prepojenia debuggera s programovanou doskou je možné využiť program `pico_blink`, ktorý má rozblikať vstavanú LED diódu programovanej dosky s periódou 500 ms. Program bol taktiež použitý v časti 1.4.2.

4.1.1 Základné užívateľské rozhranie

Táto časť popisuje testovanie základného užívateľského rozhrania.

LED diódy

Testovací program pre diódy LED1 až LED4 sa nachádza v priečinku `leds`. Po spustení sa rozblikajú LED diódy, pre ktoré je implementovaný 4-bitový čítač.

Rotačný enkóder

Program rotačného enkódera `quadrature_encoder` pracuje v kvadratúrnom režime a využíva periférium PIO, ktorá bola popísaná v časti 1.2.4. Po spustení začne program na UART port vypisovať stav pootočenia enkódera každých 100 ms. Program je prevzatý z [37].

Otočný potenciometer

Po spustení programu `potentiometer` začne dióda LED1 za použitia PWM modulácie meniť svoj jas podľa pootočenia potenciometra. Program je prevzatý z [37] a ďalej upravený.

Tlačidlá pripojené na analógový vstup

Funkcionalitu tlačidiel SW1 až SW4 pripojených na analógový vstup testuje program `dac_buttons`. Nameraná hodnota z analógového vstupu sa najprv prepočíta na elektrické napätie pomocou vzťahu 4.1, kde $U_{REF}[V]$ je hodnota napájacieho napätia 3,3 V, $N[-]$ je počet bitov A/D prevodníka, $x_{ADC}[-]$ je hodnota prečítaná z analógového vstupu a $U_{MEAS}[V]$ je výsledná hodnota meraného napätia.

$$U_{MEAS} = \frac{U_{REF}}{2^N - 1} \cdot x_{ADC} \tag{4.1}$$

Ďalej sa napätie U_{MEAS} prepočíta podľa vzťahu 4.2 na stav tlačidiel $x_{BUTTONS}[-]$, pričom $M[-]$ je počet bitov D/A prevodníka, čo má podľa počtu tlačidiel hodnotu $M = 4$. Výsledná hodnota $x_{BUTTONS}$ je zaokrúhlená na celé číslo a udáva binárne zapísaný stav tlačidiel.

$$x_{BUTTONS} = round\left(2^M \cdot \frac{U_{MEAS}}{U_{REF}}\right) \tag{4.2}$$

Program číta stav tlačidiel s periódou 100 ms a ich stav vypisuje na UART port. Diódy LED1 až LED4 svietia podľa stavu príslušného tlačidla.

4.1.2 UART rozhranie

Rozhranie UART už bolo otestované v predošlých testovacích programoch. K dispozícii je však aj program pre otestovanie výhradne UART rozhrania. Program `uart_loopback` implementuje funkciu loopback-u, čiže všetky prijaté znaky odošle naspäť. Funkciu je možné vyskúšať pomocou počítača s terminálom pre sériový port, ako je napríklad program Putty.

4.1.3 SPI zariadenia

Táto časť popisuje testovanie zariadení pripojených ku SPI zbernici.

Expander binárnych vstupov a výstupov

Program pre otestovanie expandera MCP23S018 je vypracovaný pomocou programovacieho jazyka C++ a nachádza sa v priečinku `spi_expander`. Trieda `SPI` implementuje základné metódy pre čítanie a zápis ľubovoľného počtu bajtov pri práci s SPI zbernicou.

Ďalej je vytvorená trieda `MCP23S18`, v ktorej sú na základe katalógového listu [16] definované registre pomocou `enum class REG`. Trieda `MCP23S18` obsahuje nasledujúce atribúty:

- `SPI &spi` – referencia na objekt triedy `SPI`
- `const uint cs_pin` – číslo použitého vývodu pre CS pin
- `const uint8_t deviceOpcode` – adresa pre identifikáciu viacerých expanderov MCP23S018 na SPI zbernici

V triede `MCP23S18` sú taktiež vytvorené metódy pre prácu s SPI expanderom:

- `void setDirection(uint8_t direction)` – nastavenie vstupu alebo výstupu
- `void setPullup(uint8_t pullup)` – pripojenie interných pull-up rezistorov
- `uint8_t read()` – prečítanie vstupov expandera
- `void write(uint8_t value)` – zápis na výstupy expandera

V hlavnom programe `spi_expander.cpp` sú vytvorené inštancie tried `SPI` a `MCP23S18`. Vývody expandera GP0 až GP3, na ktorých sú pripojené LED diódy LED5 až LED8 sú nastavené ako výstupy a vývody GP4 až GP7, na ktorých sú pripojené tlačidlá SW5 až SW8 sú nastavené ako vstupy. V nekonečnej slučke sa s periódou 10 ms číta stav tlačidiel SW5 až SW8 a ten sa zapisuje na LED5 až LED8.

SD karta

Spoločnosť Raspberry Pi poskytuje knižnicu pre prácu s SD kartou v repozitári [38]. Knižnica však využíva iné rozloženie pinov ako využíva RPi Pico Kit a toto rozloženie sa nedá meniť. Knižnica taktiež nepodporuje prácu so súborovými systémami FAT16, FAT32 alebo exFAT, ktoré sa spolu s SD kartami bežne využívajú.

Knižnica SdFat [40] podporuje prácu s uvedenými súborovými systémami. Napísaná je však pre vývojové dosky od spoločnosti Arduino. Jej prispôsobenie pre vývojovú dosku Raspberry Pi Pico by však bolo nad časový rámec zodpovedajúci diplomovej práci. Z tohto dôvodu otestovanie práce s micro-SD kartou neprebehlo.

4.1.4 I²C zariadenia

Na I²C zbernici je pripojených niekoľko zariadení. Pre detekciu adries aktívnych zariadení slúži program `scan_i2c_bus`. Jeho úlohou je prejsť celý adresový priestor 0x00 až 0x7F na zbernici I²C a zistiť, či na danej adrese je aktívne zariadenie. Program je prevzatý z [37] a ďalej upravený.

OLED Displej

Na demonštráciu možností použitej knižnice pre OLED displej slúži program `oled`. Knižnica obsahuje funkcie pre základnú prácu s displejom, ako je napríklad vypnutie a zapnutie displeja, nastavenie kontrastu alebo vykreslenie základnych geometrických útvarov. K dispozícii je taktiež font o základnej veľkosti 5×8 px pre vypisovanie textových správ. Program je prevzatý z [41].

EEPROM

Program pre otestovanie EEPROM pamäte z rady AT24Cxx je vypracovaný pomocou programovacieho jazyka C++ a nachádza sa v priečinku `eeprom`. Trieda `AT24Cxx` implementuje metódy pre čítanie a zápis na ľubovoľnú adresu pamäte, ktoré sú naprogramované na základe katológového listu danej EEPROM pamäte [17].

Demonštračný program pozostáva zo zápisu 10-prvkového poľa do EEPROM pamäte a následného prečítania tohto pamäťového miesta.

Čítanie EDID informácií

Čítanie EDID informácií zobrazovacej jednotky pripojenej cez VGA alebo DVI rozhranie je možné pomocou programu `edid_info`. Program na začiatku prečíta 128 bajtov z EEPROM pamäte zobrazovacej jednotky, ktorá je pripojená ku I²C zbernici a jej adresa je 0x50. Interpretácia týchto dát je uvedená v popise EDID protokolu [26].

Načítané dáta program ďalej parsuje a získané informácie spolu s ich popisom vypisuje na UART port. Pre parsovanie je využitý program z [27], ktorý je napísaný v jazyku Python, a preto bol prepísaný do jazyka C.

Program taktiež číta ďalších 128 bajtov z EEPROM pamäte zobrazovacej jednotky, čo predstavuje rozšírenie základných dát. Tie však už v tomto programe nie sú parsované.

4.1.5 Zvukový výstup

Táto časť popisuje testovanie troch rôznych možností zvukového výstupu.

Piezoelektrický menič

Funkčnosť piezoelektrického meniča testuje program v priečinku `piezo`. Program postupne mení frekvenciu generovaného tónu v zadanom intervale s určitým krokom.

PWM generátor zvuku

Testovací program pre PWM generátor zvuku `pwm_audio` generuje tón pre jeden zvukový kanál o stálej frekvencii so vzorkovacou frekvenciou 24 kHz a rozlíšením 16 bitov. Frekvencia a hlasitosť generovaného tónu sa dá nastavovať pomocou terminálu cez UART port. Program sa nepodarilo upraviť tak, aby generoval zvuk pre obe zvukové kanály.

Aby bolo možné projekt skompilovať je nutné mať stiahnutý repozitár s názvom `pico-extras` [38], ktorý obsahuje knižnicu pre generovanie zvuku pomocou PWM modulácie. Program je prevzatý z [39].

PCM generátor zvuku

Pre demonštráciu funkcie PCM generátora zvuku bol využitý program `pcm_audio`, ktorý je prevzatý z [39]. Testovací program implementuje USB zvukovú kartu. Zvuk je generovaný PCM moduláciou prostredníctvom integrovaného obvodu PCM5101A. Program s týmto čipom komunikuje cez I²S rozhranie. Aj tento program si vyžaduje stiahnutý repozitár `pico-extras`, v ktorom sú implementované potrebné knižnice.

Zvuková karta poskytuje dvojkanálový výstup s rozlíšením 16 bitov a možnosťou výberu vzorkovacej frekvencie 44,1 kHz alebo 48 kHz. Pre pripojenie je potrebné využiť USB konektor programovanej dosky a nie konektor, ktorý slúži na programovanie a odladzovanie. Po pripojení k USB portu počítača sa zariadenie zobrazuje pod názvom RPi Pico Kit Sound Card. Hlasitosť zvuku sa nastavuje ovládaním hlasitosti na počítači.

4.1.6 Obrazový výstup

Táto časť popisuje testovanie obrazového výstupu prostredníctvom VGA a DVI rozhrania.

VGA rozhranie

Pre generovanie obrazového výstupu na zobrazovaciu jednotku pripojenú cez rozhranie VGA je možné využiť oficiálnu knižnicu pico-scanvideo od spoločnosti Raspberry Pi, ktorá sa nachádza v repozitári `pico-extras` [38] alebo knižnice PicoVGA či PicoQVGA, ktoré sú umiestnené v rovnomenných repozitároch [42] a [43]. Porovnanie jednotlivých knižníc je uvedené v tabuľke 4.1, pričom farebná paleta popisuje

akú bitovú hĺbku majú jednotlivé farby RGB. Z tabuľky je zrejmé, že najvyššiu bitovú hĺbku poskytuje knižnica pico-scanvideo a knižnica PicoVGA poskytuje najvyššie rozlíšenie obrazu. Všetky tri uvedené knižnice využívajú pre generovanie obrazu periférium PIO.

Tab. 4.1: Porovnanie knižníc pre generovanie obrazového výstupu na VGA rozhranie

	pico-scanvideo	**PicoVGA**	**PicoQVGA**
Bitová hĺbka	16 bitov	8 bitov	8 bitov
Farebná paleta	R5 – G6 – B5	R3 – G3 – B2	R3 – G3 – B2
Maximálne rozlíšenie obrazu	1280×1024 px pri 60 fps	1360×768 px pri 60 fps	320×240 px pri 60 fps

Vzhľadom na to, že knižnica pico-scanvideo by si vyžadovala použitie 16 vývodov dosky Raspberry Pi Pico kvôli 16-bitovej hĺbke, toto riešenie nie je vhodné. Pre účely vývojového kitu RPi Pico Kit je 8-bitová hĺbka postačujúca. Hlavnou výhodou knižníc PicoVGA a PicoQVGA je taktiež to, že poskytujú výrazne väčšie množstvo demonštračných programov oproti knižnici pico-scanvideo a natívne podporujú jednokanálový zvukový výstup pomocou PWM modulácie.

Knižnice PicoVGA a PicoQVGA ďalej poskytujú niekoľko pomocných nástrojov slúžiacich napríklad pre prevod obrázkov a zvukových súborov na polia hodnôt, ktoré sa k programu pridajú vo forme hlavičkových súborov jazyka C. Medzi tieto nástroje patria nasledujúce programy [28]:

- `pal332` – program generuje 8-bitovú farebnú paletu R3 – G3 – B2 s príponami `.csv` a `.act`. Vygenerovaný súbor `.act` je možné použiť v grafických editoroch, ako je napríklad Adobe Photoshop alebo Gimp pre prevod ľubovoľných obrázkov do palety R3 – G3 – B2.
- `RaspPicoImg` – program pre prevod obrázku na pole hodnôt. Vstupný obrázok musí využívať 8-, 4- alebo 1-bitovú paletu a musí byť vo formáte BMP s PCM kompresiou a zapnutým obrátením poradia liniek.
- `RaspPicoRle` – program pre prevod obrázku do RLE[1] komprimovaného formátu. Vstupný obrázok musí využívať 8-bitovú paletu a byť vo formáte BMP s rovnakými nastaveniami ako v predošlom bode. Ďalším vstupným parametrom programu je číslo farby, ktorá sa v obrázku zamení za priehľadnú. Program je dostupný iba v knižnici PicoVGA.

[1]RLE (Run-length Encoding) je bezstratová kompresia, ktorá vo vstupnom dátovom toku kóduje postupnosti opakujúcich sa znakov. Miesto tejto postupnosti je uložený iba špeciálny znak predstavujúci indikátor, opakovaný znak a počet opakovaní [49].

- `RaspPicoSnd` – program pre prevod zvukového súboru na pole hodnôt. Vstupný zvukový súbor musí byť jednokanálový, s 8-bitovou hĺbkou a vzorkovacou frekvenciou 22050 Hz. Uložený musí byť vo formáte WAV s PCM kompresiou.

Knižnice PicoVGA a PicoQVGA sa okrem maximálneho podporovaného rozlíšenia obrazu líšia aj v použití synchronizačných signálov. Knižnica PicoQVGA využíva horizontálny a vertikálny synchronizačný signál HSYNC a VSYNC (zapojenie na strane č. 52) [29]. Knižnica PicoVGA tieto dva signály spája do jedného kompozitného synchronizačného signálu CSYNC a predošlé zapojenie mení tak, že signál CSYNC využíva vývod č. 13 VGA konektora, zatiaľ čo vývod č. 14 ostáva nevyužitý [28].

Jednotlivé zobrazovacie jednotky využívajúce VGA rozhranie sa môžu líšiť tým, aké synchronizačné signály podporujú. Túto informáciu je možné vyčítať zo zobrazovacej jednotky z EDID metadát, čo umožňuje program `edid_info` popísaný v predošlom texte.

Autor práce mal v čase písania tejto práce k dispozícii počítačový monitor HP LE1901W s maximálnym rozlíšením 1440×900 px pri 60 fps. Uvedený monitor podporuje iba oddelené synchronizačné signály HSYNC a VSYNC a kompozitný synchronizačný signál CSYNC nepodporuje. Preto je v ďalšej časti práce použitá knižnica PicoQVGA, ktorá podporuje signály HSYNC a VSYNC.

Pôvodný zdrojový kód knižnice PicoQVGA má vytvorený vlastný zostavovací systém využívajúci Makefile súbory a je prispôsobený pre operačný systém Windows. Knižnica taktiež využíva aj C/C++ SDK, ktoré je prevzaté z oficiálne poskytovaného SDK [36] a ďalej je upravené tak, aby vyhovovalo použitému zostavovaciemu systému. Použitie vlastného zostavovacieho systému autor knižnice odôvodňuje ako výhodné z dôvodu menšej náročnosti na minimálne systémové požiadavky počítača.

Autor práce však považuje zostavovací systém knižnice za značne obmedzujúci, pretože programy nie je možné jednoducho začleniť do oficiálne podporovaného zostavovacieho systému. Do značnej miery je sťažená aj aktualizácia upraveného SDK, ktoré sa neustále vyvíja. V neposlednom rade použitie oficiálneho zostavovacieho systému funguje na všetkých bežne používaných operačných systémoch, tak ako to bolo uvedené v časti 1.4.

Kvôli uvedeným skutočnostiam bola celá knižnica PicoQVGA prerobená tak, aby sa dala začleniť do oficiálne podporovaného zostavovacieho systému. Spolu so súbormi knižnice boli prerobené všetky demonštračné programy.

V knižnici PicoQVGA bolo taktiež zmenené rozloženie pinov pre obrazový a zvukový výstup, tak aby bolo v súlade s použitou schémou zapojenia na strane č. 54.

Pre jednoduché pridávanie obrázkov vo formáte BMP bola vytvorená funkcia programu CMake `qvga_generate_img_header` pre vygenerovanie hlavičkového sú-

boru z obrázka. Funkcia využíva vyššie popísaný program `RaspPicoImg`.

Pre vytvorenie hlavičkového súboru stačí vložiť príkaz z výpisu 4.1 do príslušného `CMakeLists.txt` súboru projektu. Prvým argumentom funkcie je názov projektu v programe CMake. Ďalší argument predstavuje cestu ku obrázku vo formáte BMP a posledný argument označuje názov vygenerovaného poľa, ktoré je následne možné použiť v programe. Vygenerovaný hlavičkový súbor sa do projektu pridá vložením hlavičkového súboru `<name>.h`, kde `<name>` predstavuje názov pôvodného súboru vrátane prípony `.bmp`. Vygenerované pole je dátového typu `static const uint8_t`.

Výpis 4.1: Príkaz pre vygenerovanie hlavičkového súboru z obrázka

```
qvga_generate_img_header(fifteen
    ${CMAKE_CURRENT_LIST_DIR}/img/tiles.bmp TilesImg)
```

Výpis 4.2: Príkaz pre vygenerovanie hlavičkového súboru zo zvukového súboru

```
qvga_generate_snd_header(fifteen
    ${CMAKE_CURRENT_LIST_DIR}/snd/bump.wav BumpSnd)
```

Podobne je pre jednoduché pridávanie zvukových súborov vo formáte WAV vytvorená funkcia programu CMake `qvga_generate_snd_header` pre vygenerovanie hlavičkového súboru zo zvukového súboru. Funkcia využíva vyššie popísaný program `RaspPicoSnd`.

Pre vytvorenie hlavičkového súboru stačí vložiť príkaz z výpisu 4.2 do príslušného `CMakeLists.txt` súboru projektu. Význam argumentov funkcie je podobný ako pri predošlom príkaze pre prácu s obrázkami, s tým rozdielom, že táto funkcia pracuje so zvukovými súbormi s príponou `.wav`.

Celá knižnica PicoQVGA spolu s demonštračnými programami sa nachádza v elektronickej prílohe F v repozitári `pico-kit-qvga`.

DVI rozhranie

Generovanie obrazu pre zobrazovaciu jednotku pripojenú cez DVI rozhranie zabezpečuje knižnica v repozitári PicoDVI [35], v ktorom sa nachádza aj príslušná dokumentácia. Knižnica umožňuje generovať obraz s 8-bitovou hĺbkou R3 – G3 – B2 alebo 16-bitovou hĺbkou R5 – G6 – B5. Maximálne podporované rozlíšenie obrazu je 1280×720 px pri 30 fps.

Knižnica PicoDVI predvolene používa rovnaké rozloženie vývodov ako navrhnutý koncept HW platformy RPi Pico Kit na strane č. 54. To znamená, že v knižnici

nie je potrebné nič meniť a demonštračné programy stačí iba skompilovať a nahrať do programovanej dosky Raspberry Pi Pico.

Testovanie DVI rozhrania bolo uskutočnené s rovnakým počítačovým monitorom HP LE1901W ako pri testovaní VGA rozhrania. Uvedený monitor má však iba VGA vstup. Navrhnutá platforma využíva pre výstup DVI rozhrania HDMI konektor, ako to bolo popísané v časti 3.2.8. Pre prepojenie HW platformy a počítačového monitora bol preto využitý adaptér pre prevod z HDMI na VGA.

Pri testovaní sa podarilo dosiahnuť rozlíšenie 640×480 px pri 60 fps so 16-bitovou hĺbkou, čo je najnižšie podporované rozlíšenie knižnice. Pri vyššom rozlíšení monitor nezobrazoval žiaden obraz. Riešenie tohto nedostatku by však bolo nad časový rámec zodpovedajúci diplomovej práci.

4.2 Zhodnotenie testovania HW platformy

V tejto kapitole boli otestované jednotlivé časti konceptu HW platformy, okrem práce s micro-SD kartou. Vytvorené boli viaceré testovacie programy, ktoré slúžia rovnako aj na demonštráciu funkcie danej súčasti. Zistené nedostatky boli priebežne zapracované do návrhu HW platformy a jednotlivých programov.

Výsledkom je navrhnutý a otestovaný koncept HW platformy, ktorého všetky otestované súčasti fungujú správne. V nasledujúcej kapitole bude popísaný návrh a zrealizovanie finálnej verzie tejto HW platformy.

5 Realizácia finálnej revízie HW platformy

Realizácia finálnej revízie HW platformy pozostávala z výberu súčiastok, návrhu a výroby dosky plošných spojov a jej následného osadenia a otestovania. Obsahom tejto kapitoly je detailné popísanie týchto krokov.

5.1 Výber súčiastok

Pre testovanie funkcionality jednotlivých súčastí v kapitole č. 4 boli využité výlučne vývodové THT súčiatky. Použitie THT súčiastok by vo finálnej verzii znamenalo príliš veľké rozmery prípravku. S cieľom zachovania čo najmenších rozmerov výsledného vývojového kitu boli THT súčiastky nahradené za ekvivalentné súčiastky určené pre povrchovú montáž SMD. Potenciometer, enkóder, piezoelektrický menič rovnako ako aj všetky konektory vrátane pinových líšt boli ponechané ako THT súčiastky, z dôvodu zachovania lepšej mechanickej odolnosti pri manipulácii s nimi.

Výsledný vývojový kit využíva dohromady takmer 200 súčiastok. Kompletný zoznam súčiastok je k dispozícii v prílohe B. Pre každú súčiastku je uvedená jej menovitá hodnota, požadované prevedenie alebo puzdro, predajca, od ktorého bola súčiastka zakúpená a označenie súčiastky výrobcom, podľa ktorého sa dá súčiastka u uvedeného predajcu zakúpiť. Použitie uvedených súčiastok nie je striktne vyžadované, podstatná je iba ich menovitá hodnota a prevedenie. Informácie o označení súčiastky výrobcom a predajcovi sú uvedené len informatívne pre zjednodušenie objednávky súčiastok v prípade výroby ďalších kusov vývojového kitu.

Ako je zrejmé zo zoznamu súčiastok, väčšina pasívnych súčiastok využíva veľkosť puzdra 0603. Pre LED diódy je využité puzdro 0805. Rezistorové siete a polymérové PTC poistky sú vo veľkosti 1206. Pinové lišty využívajú štandardný rozostup 2,54 mm.

Vývojová doska Raspberry Pi Pico je k vývojovému kitu pripojená modulárne cez kombináciu dutinkových a kolíkových líšt. Rovnakým spôsobom sa pripája aj OLED displej.

Výsledná cena súčiastok sa pohybuje okolo hodnoty 1000 CZK v závislosti na množstevných zľavách pri väčších objednávkach.

Na základe vybraných súčiastok bolo ku každej súčiastke v projekte programu KiCad pridelené príslušné puzdro a 3D model.

5.2 Návrh dosky plošných spojov

Návrh dosky plošných spojov (DPS alebo po anglicky PCB) je rovnako ako aj schéma vypracovaný v programe KiCad, ktorý je k dispozícii v elektronickej prílohe D.

5.2.1 Požiadavky pre návrh DPS

Na základe nízkej ceny a pozitívnych predošlých skúseností bola pre výrobu dosky zvolená čínska spoločnosť JLCPCB. Vodivé cesty, ktoré sú vedené k USB zbernici a ku VGA a DVI rozhraniu si vyžadujú impedančné prispôsobenie, ktoré bude popísané v ďalšom texte. Aby bola pri výrobe zaručená požadovaná impedancia, výrobca musí poskytovať možnosť riadenej impedancie. Výrobca JLCPCB túto požiadavku spĺňa pre výrobu 4- a 6-vrstvových DPS, nie však pre 2-vrstvové DPS. Z tohto dôvodu bola pre vývojový kit použitá 4-vrstvová DPS.

Výrobca má pre výrobu DPS do rozmerov $100 \times 100\,\text{mm}$ pevne stanovenú cenu na 8 USD pre 5 ks alebo 14 USD pre 10 ks. Pre väčšie rozmery je cena vypočítavaná podľa plochy DPS a do ceny je navyše zarátaný fixný poplatok 35 USD. Z tohto dôvodu sú najoptimálnejšie rozmery pre výsledný prípravok $100 \times 100\,\text{mm}$.

Ako základný materiál DPS bol vybratý materiál FR-4 kvôli jeho nízkej cene. Hrúbka DPS bola stanovená na hodnotu 1,6 mm s hrúbkou medi 0,035 mm, čo je označované aj ako $1\,\text{oz/ft}^2$.

Skladba vrstiev DPS

Výrobca JLCPCB pre hrúbku DPS 1,6 mm poskytuje na výber z dvoch možností pre skladbu vrstiev DPS, ktoré sa vzájomne líšia v hrúbke prepreg vrstiev a v dielektrických vlastnostiach. Z týchto možností bola vybratá skladba vrstiev, ktorú výrobca označuje pod kódom JLC7628, pretože využíva prepreg s označením JLC7628. Popis jednotlivých vrstiev je uvedený v tabuľke 5.1, pričom ID vrstvy je označenie vrstvy v programe KiCad a $\varepsilon_r[-]$ je relatívna permitivita.

Tab. 5.1: Popis skladby vrstiev DPS [33]

ID vrstvy	**Typ vrstvy**	**Materiál**	**Hrúbka**	$\varepsilon_\mathbf{r}[-]$
F.Mask	Spájkovacia maska	–	0,02 mm	3,8
F.Cu	Horná vrstva spojov	Meď	0,035 mm	–
Dielectric 1	Prepreg	JLC7628	0,2 mm	4,6
In1.Cu	Vnútorná vrstva spojov č. 1	Meď	0,0175 mm	–
Dielectric 2	Jadro	FR-4	1,065 mm	4,6
In2.Cu	Vnútorná vrstva spojov č. 2	Meď	0,0175 mm	–
Dielectric 3	Prepreg	JLC7628	0,2 mm	4,6
B.Cu	Dolná vrstva spojov	Meď	0,035 mm	–
B.Mask	Spájkovacia maska	–	0,02 mm	3,8

Výrobca ďalej udáva, že hrúbka 0,2 mm prepreg vrstvy JLC7628 je nominálna hodnota a pre výpočet riadenej impedancie je potrebné použiť hodnotu 0,18 mm v prípade, že vodivé cesty sú v hornej alebo dolnej vrstve spojov. V prípade, že tieto cesty sú vo vnútorných vrstvách spojov, pre výpočet sa má ako hrúbka vrstvy použiť hodnota 0,21 mm [33].

Definovanie výrobných a technologických možností

Každý výrobca DPS definuje svoje výrobné a technologické možnosti, ktoré musí DPS spĺňať, aby ju bolo možné vyrobiť. Všetky požiadavky výrobcu JLCPCB sú dostupné na stránke výrobcu [32]. Zhrnutie základných požiadavok pre 4-vrstvové DPS je nasledovné.

- Minimálna vzdialenosť medzi cestami: 0,09 mm
- Minimálna šírka cesty: 0,09 mm
- Najmenší vŕtaný otvor: 0,2 mm
- Minimálny priemer prstenca okolo prekovenia: 0,4 mm
- Minimálna vzdialenosť od okraja dosky: 0,4 mm

Tieto parametre boli vložené do programu KiCad, ktorý na základe nich spúšťa program DRC[1].

Šírka vodivých ciest

Voľbu šírky vodivých ciest obmedzujú technologické možnosti výrobcu a zároveň prúd pretekajúci danou cestou.

Maximálny prúd vývojového kitu je obmedzený polymérovou PTC poistkou s menovitou hodnotou prúdu 500 mA. Pre tento prúd a maximálne oteplenie spoja o 10 °C je pre vonkajšie vrstvy medi požadovaná minimálna šírka cesty 0,3 mm. Pre cesty vedené vo vnútorných vrstvách, ktoré majú polovičnú hrúbku medi, je minimálna šírka dvojnásobná a to 0,6 mm. Tieto hodnoty boli vypočítané pomocou internetového nástroja [30]. Pre všetky napájacie cesty bola zvolená šírka ciest 0,8 mm.

Prúd signálovými cestami je zanedbateľný, a preto šírku signálových ciest obmedzujú iba technologické možnosti výrobcu. Pre signálové cesty boli podľa potreby použité šírky ciest v rozmedzí 0,127 až 0,4 mm.

Pre signály USB zbernice a obrazového VGA a DVI rozhrania je navyše kladená požiadavka pre ich impedanciu. V tabuľkách 5.2 a 5.3 sú uvedené požadované šírky ciest pre tieto signály ako aj parametre, na základe ktorých bola daná hodnota vypočítaná. Pre výpočet boli použité hodnoty z tabuľky 5.1. Požadované hodnoty

[1]DRC (Design Rule Checking) je program, ktorý kontroluje splnenie požiadavok pre výrobu DPS. Je súčasťou návrhového programu pre DPS.

impedancie boli získané z katalógových listov [13] a [35]. Šírky ciest boli vypočítané pomocou internetového nástroja [31].

Tab. 5.2: Výpočet šírky impedančne riadených ciest signálov VGA rozhrania

Signál	**Impedancia**	**Hrúbka medi**	**Hrúbka prepregu**	$\varepsilon_{\mathbf{r}}[-]$	**Šírka cesty**
VGA	75 Ω	0,035 mm	0,18 mm	4,6	0,127 mm

Tab. 5.3: Výpočet šírky impedančne riadených ciest diferenciálnych signálov

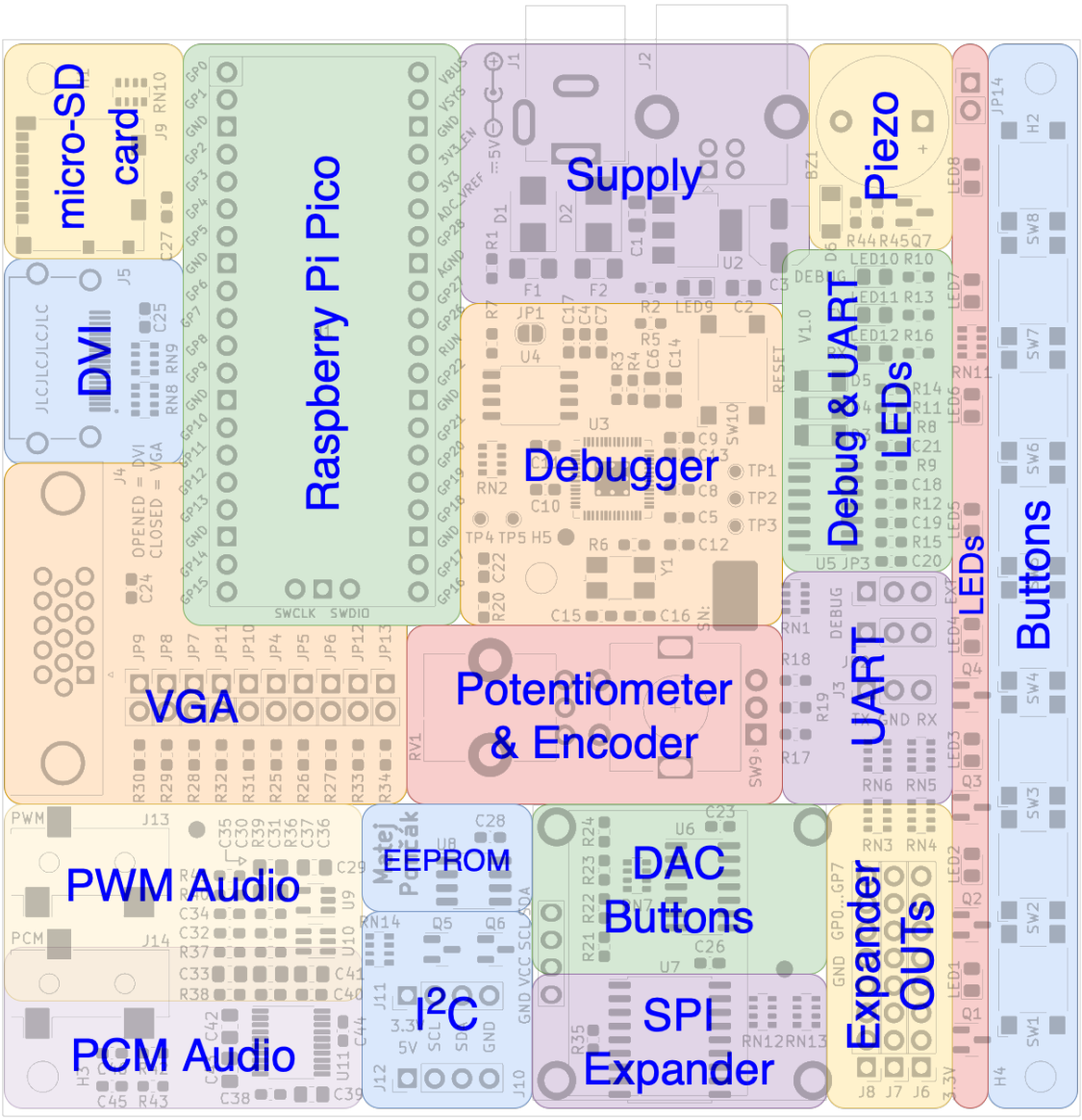
Signál	**Impedancia**	**Hrúbka medi**	**Hrúbka prepregu**	$\varepsilon_{\mathbf{r}}[-]$	**Medzera medzi cestami**	**Šírka cesty**
USB	90 Ω	0,035 mm	0,18 mm	4,6	0,254 mm	0,381 mm
DVI	100 Ω	0,035 mm	0,18 mm	4,6	0,2032 mm	0,3048 mm

5.2.2 Návrh obrazcov dosky plošných spojov

Prvým krokom návrhu obrazcov dosky plošných spojov bolo vymedzenie maximálnych rozmerov 100×100 mm. Po rohoch tohto štvorca boli pridané štyri montážne otvory s priemerom 2,7 mm, aby bolo možné vyvinutý prípravok neskôr prichytiť do krabičky pomocou skrutiek. Neskôr bol pridaný aj otvor uprostred dosky, ktorej použitie zamedzí prehýbaniu dosky.

Ďalším krokom bolo rozmiestnenie všetkých konektorov, tak aby boli na okraji dosky. Vývojová doska Raspberry Pi Pico taktiež obsahuje USB konektor, a preto bola umiestnená tak, aby tento konektor bol rovnako na okraji dosky. Nasledovalo rozmiestnenie ôsmich tlačidiel spolu s príslušnými ôsmimi LED diódami po voľnom okraji dosky. Potenciometer a enkóder boli umiestnené napravo od OLED displeja pre intuitívne ovládanie. Rozmiestnenie týchto prvkov jasne zadefinovalo obmedzenia pre umiestnenie ostatných súčiastok.

Pre zjednodušenie ďalšieho návrhu boli súčiastky jednotlivých súčasti HW zoskupené. Jednotlivé celky boli postupne ukladané do vymedzenej plochy DPS, tak aby boli čo najbližšie blízko seba a rešpektovali pravidlá návrhu DPS [4]. Vzhľadom na to, že vývojová doska Raspberry Pi Pico a OLED displej sa pripájajú k prípravku modulárne cez dutinkové a kolíkové lišty, súčiastky mohli byť umiestnené aj pod tieto prvky. Výsledok rozmiestnenia súčiastok je na obrázku 5.1.



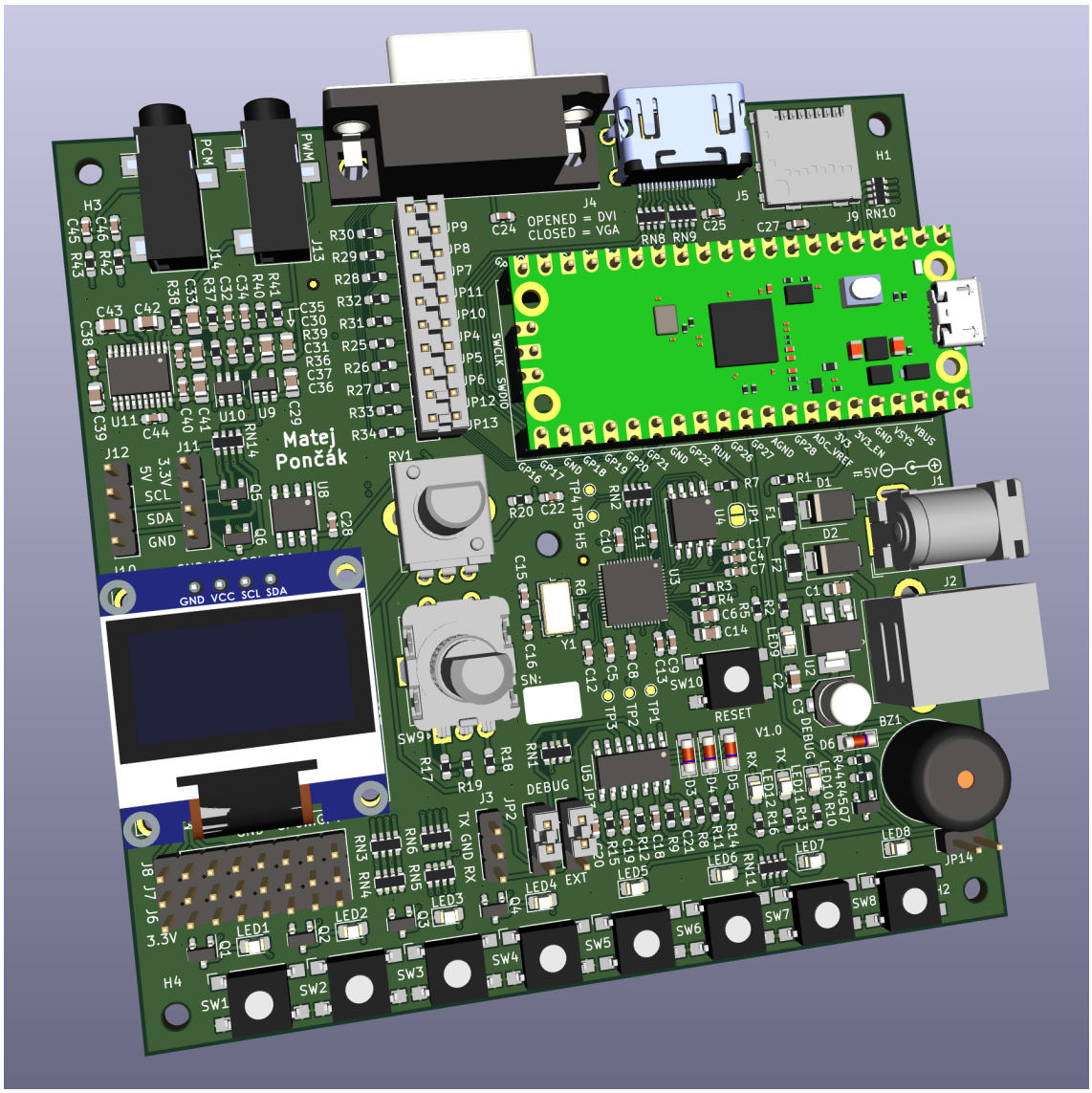
Obr. 5.1: Rozmiestnenie jednotlivých HW súčastí na navrhnutej DPS

Po rozmiestnení všetkých prvkov sa prešlo k ďalšiemu kroku, a to ku pospájaniu jednotlivých komponentov podľa schémy zapojenia. Na začiatku bol počet nepospájaných ciest 534. Ako prvé boli pospájané cesty s riadenou impedanciou s presne definovanou šírkou ciest. Následne boli pospájané všetky cesty v jednotlivých súčastiach HW platformy. Ďalej nasledovalo privedenie kladnej vetvy napájania pre každú súčasť a napojenie súčastí na vývody dosky Raspberry Pi Pico. Posledným krokom bolo vytvorenie zemniacich plôch v každej vodivej vrstve. Vyrovnanie potenciálov zeme bolo zabezpečené pridaním niekoľkých desiatok prekovení po celej ploche DPS medzi všetkými štyrmi vrstvami.

Poslednou časťou návrhu DPS bolo usporiadanie popisu súčiastok, vyznačenie

menovitého napájacieho napätia a správneho zapojenia napájacieho DC jacku. Pridané bolo aj miesto pre vyznačenie sériového čísla pre identifikáciu dosiek v prípade výroby viacerých kusov.

Výsledný návrh obrazcov jednotlivých vrstiev plošných spojov ako aj osadzovací plán súčiastok sa nachádza v prílohe C. V prílohe je tiež obrazec šablóny pre nanesenie spájkovacej pasty, ktorej použitie bude vysvetlené v ďalšom texte. Obrázok 5.2 zobrazuje 3D model navrhnutého prípravku RPi Pico Kit.



Obr. 5.2: 3D model navrhnutého prípravku RPi Pico Kit

5.2.3 Výrobná a zadávacia dokumentácia

Navrhnutý prípravok spolu so šablónou pre nanesenie spájkovacej pasty boli zadané do výroby v čínskej spoločnosti JLCPCB. Táto časť popisuje spôsob vytvorenia potrebnej výrobnej a zadávacej dokumentácie.

Výroba DPS

Pre výrobu DPS boli pomocou návodu uvedeného na stránke výrobcu [34] vygenerované gerber súbory, ktoré slúžia ako výrobná dokumentácia. Súbory sú k dispozícii v elektronickej prílohe E. V tejto prílohe je popísaný aj význam jednotlivých súborov.

Doska plošných spojov bola zadaná do výroby s parametrami uvedenými v tabuľke 5.4. Cena výroby 10 ks DPS bola 14 USD.

Tab. 5.4: Špecifikácia parametrov pre výrobu DPS

Parameter	**Hodnota**	**Parameter**	**Hodnota**
Rozmery	100×100 mm	Hrúbka vonkajšej medi	1 oz/ft^2
Hrúbka DPS	1,6 mm	Minimálny otvor	0,2 mm
Počet vrstiev	4	Úprava povrchu	HASL (s olovom)
Typ materiálu	FR4-Standard Tg 130–140°C	Farba masky	zelená
Riadená impedancia	áno	Farba potlače	biela
Skladba vrstiev DPS	JLC7628	Umiestnenie čísla objednávky	špecifikované

Výroba šablóny pre nanesenie spájkovacej pasty

Vygenerované gerber súbory obsahujú aj potrebné podklady pre výrobu šablóny na nanesenie spájkovacej pasty. Tabuľka 5.5 špecifikuje parametre pre výrobu. Cena výroby šablóny bola 10 USD.

Tab. 5.5: Špecifikácia parametrov pre výrobu šablóny na nanesenie spájkovacej pasty

Parameter	**Hodnota**	**Parameter**	**Hodnota**
Rozmery	380×280 mm	Výrobný proces	leptanie
Užívateľské rozmery	150×150 mm	Orámovanie šablóny	nie
Strana súčiastok	horná	Zameriavacie body	žiadne

5.3 Osadenie vývojového kitu

Výrobca JLCPCB poskytuje aj možnosť osadenia DPS. Pre výrobu zopár testovacích prototypov prípravku to však nebolo cenovo výhodné. V tejto časti je preto popísaný postup osadenia DPS.

5.3.1 Osadenie pomocou pretavenia

Priemyselné osadzovanie SMD súčiastok funguje tak, že pomocou šablóny sa na DPS nanesie tenká vrstva spájkovacej pasty. Následne pomocou osadzovacieho automatu sa na DPS poukladajú všetky SMD súčiastky. Nakoniec sa celá doska vloží do pretavovacej pece, v ktorej sa spájkovacia pasta pretaví a súčiastky sa tak pevne prichytia k DPS.

Autor práce si chcel tento proces vyskúšať v tzv. domácich podmienkach. Pre osadenie boli postupne vykonané podobné kroky ako pri priemyselnom osádzaní. Na DPS bola za pomoci šablóny nanesená spájkovacia pasta. Výsledok tohto kroku nebol najlepší, pretože pasta sa mierne rozliala a prepojila plôšky SMD súčiastok, ktoré boli blízko seba. Nasledovalo ručné poukladanie všetkých SMD súčiastok.

Namiesto pretavovacej pece bola použitá IR lampa Jovy Systems RE-7500, ktorá bola k dispozícii na VUT FEKT UAMT v skupine meracej techniky. IR lampa je vybavená aj predohrevom, čo zmierňuje tepelný šok súčiastok pri pretavovaní. Dané zariadenie je určené predovšetkým pre výmenu súčiastok s puzdrom BGA na doskách plošných spojov. Veľkosť ožarovanej plochy je preto pomerne malá a to približne $50 \times 50\,\mathrm{mm}$. Z toho vyplýva, že proces pretavenia musel byť zopakovaný štyrikrát, aby sa pretavila celá plocha DPS.

Proces osadenia súčiastok dopadol dobre aj napriek pôvodnému prepojeniu niektorých SMD plôšok spájkovacou pastou. Dosku sa však oživiť nepodarilo pravdepodobne kvôli vystaveniu súčiastok vysokej teplote opakovane po sebe. Taktiež nebol dodržaný teplotný profil osadzovania [18] z dôvodu nedostatku skúseností s používaním danej IR lampy.

Výsledok tohto spôsobu osadenia prípravku nebol úspešný, avšak autorovi priniesol cenné skúsenosti a zlepšil prehľad v spôsoboch priemyselného osadzovania SMD súčiastok.

5.3.2 Ručné osadenie

Doska plošných spojov vyvinutého prípravku bola napokon kompletne osadená ručne pomocou mikrospájkovačky. Jediný problém pri osadzovaní predstavoval čip RP2040 s puzdrom QFN-56 a HDMI konektor, ktorého vývody sú veľmi blízko seba. Dosku sa však úspešne podarilo osadiť.

5.4 Oživenie a otestovanie vývojového kitu

Osadený prípravok bol pripojený k napájaciemu napätiu 5 V, pričom neustále bol sledovaný prúd tečúci zo zdroja. Doska nevykazovala žiaden skrat a všetky ostatné generované napájacie napätia mali správnu hodnotu.

Nasledovalo nahratie programu PicoProbe cez rozhranie USB do mikrokontroléra RP2040, ktorý na doske slúži ako debugger. To sa však nepodarilo a preto pre nahratie programu boli využité vyvedené testovacie plôšky SWD rozhrania. Po následnom odladzovaní bolo zistené, že USB rozhranie debuggera nefunguje preto, lebo mikrokontrolér nemá hodinový signál generovaný kryštálom Q1. Po ďalšom testovaní bolo zistené, že na čipe mikrokontroléra RP2040 s QFN puzdrom bol voľným okom nepozorovateľný skrat. Po odstránení skratu doska začala pracovať podľa očakávaní.

Po oživení bola doska ďalej otestovaná testovacími programami navrhnutými v kapitole 4. Testovaním boli zistené dva nedostatky.

Prvým nedostatkom je skrat, ktorý sa vytvorí pripojením počítačového monitora. Skrat vzniká z dôvodu chyby v schéme zapojenia [52], z ktorej autor čerpal. Na vývody č. 4 a 9 konektora VGA je pripojené napájacie napätie 5 V. Externý monitor má však na vývode č. 4 napojenú zem, čo prepája napájacie napätie so zemou. Tento skrat bol odstránený ručným preškrabaním napácej vetvy pre konektor VGA. Vďaka tomu sa podarilo otestovať funkciu generovania obrazového výstupu na VGA rozhranie. Preškrabaním cesty sa však znemožnilo použitie I²C zbernice pre čítanie EDID metadát VGA monitora.

Druhým nedostatkom je, zámena vývodov LRCLK a BCLK I²S zbernice pre komunikáciu s integrovaným obvodom PCM5101A pre generovanie zvuku pomocou PCM modulácie. Tento problém bol odstránený úpravou softvérovej knižnice dodávanej spoločnosťou Raspberry Pi. Pre implementovanie tohto riešenia je potrebné dodaný súbor `audio_i2s.pio` v priečinku testovacieho programu `pcm_audio` zameniť so súborom:

`pico-extras/src/rp2_common/pico_audio_i2s/audio_i2s.pio`

Všetky ostatné súčasti HW platformy fungujú správne. Zistené nedostatky budú odstránené na hardvérovej úrovni v ďalšej verzii prípravku RPi Pico Kit.

6 Návrh a vypracovanie laboratórnych úloh

V časti 2.3.1 tejto práce bol v rámci inovácie predmetu Vstavané systémy a mikroprocesory navrhnutý koncept laboratórnych úloh, ktoré by využívali vývojovú dosku Raspberry Pi Pico. V ďalšom texte budú vytvorené a popísané finálne zadania týchto úloh a tiež bude uvedené ich vzorové riešenie.

6.1 Popis zadaní laboratórnych úloh

Zadania laboratórnych úloh sa nachádzajú v prílohe G. Jednotlivé úlohy na seba nadväzujú a využívajú poznatky a skúsenosti získané z predošlých úloh. Študenti tak môžu svoje vedomosti postupne upevňovať a ďalej rozširovať.

Úlohy nevyužívajú C/C++ SDK popísané v časti 1.4.2, ale používajú sa priamo registre mikrokontroléra, ktorých hodnoty sú nastavované pomocou bitových operácií. To zaručí lepšie porozumenie princípu fungovania mikrokontroléra.

Každé zadanie laboratórnej úlohy obsahuje krátke zhrnutie toho, čo je náplňou danej úlohy a stručný prehľad toho, čo je potrebné vedieť pre jej vypracovanie. Tieto zhrnutia môžu taktiež poslúžiť ako podklady pre prípravu vyučujúcich na laboratórne cvičenia. Ďalej zadanie úlohy v niekoľkých odrážkach uvádza ciele, ktoré majú byť v danej úlohe splnené.

Nasleduje domáca príprava, ktorá študentov prevedie základnými poznatkami, ktoré potrebujú pre vyriešenie samotnej laboratórnej úlohy. Jej cieľom je študentov vopred pripraviť na riešenie laboratórnej úlohy, čo môže výrazne zvýšiť ich rýchlosť práce na cvičeniach. V domácej príprave študentov čaká niekoľko úloh určených na samostatnú prácu. Úlohy sú zväčša zamerané na hľadanie potrebných informácií v katalógových listoch mikrokontroléra RP2040 a vytvoreného laboratórneho prípravku RPi Pico Kit. Ďalšie úlohy sú zamerané na zopakovanie základov výpočtovej techniky a programovania v jazyku C, pomocou ktorého majú byť úlohy naprogramované.

Hlavnou časťou každého zadania je samotné zadanie práce na cvičenia. Prvé tri body každej úlohy informujú o tom, ako správne založiť projekt a nastaviť zostavovací systém programu CMake, tak aby správne generoval výstupný kód pre mikrokontrolér. V tomto popise sú tiež uvedené knižnice jazyka C, ktoré sa majú zahrnúť do zdrojového kódu pre jeho úspešný preklad.

V ďalších bodoch zadania je krok po kroku popísaný postup pre vypracovanie úlohy, ako napríklad, ktoré registre sa majú ako nastaviť pre správnu funkciu periférie alebo aké funkcie je potrebné vytvoriť a akú majú mať hlavičku. Riešenie úlohy stavia na poznatkoch získaných z domácej prípravy študentov.

6.2 Popis vzorových riešení laboratórnych úloh

Vzorové riešenie laboratórnych úloh sa nachádza v elektronickej prílohe H a pozostáva zo vzorového riešenia domácej prípravy a vzorových zdrojových kódov. Všetky zdrojové kódy sú priamo zdokumentované formou komentárov. Navyše hlavičky všetkých vytvorených funkcií obsahujú štandardizovaný popis pre generovanie dokumentácie pomocou programu doxygen. Tieto popisy taktiež využíva vývojové prostredie Visual Studio Code ako nápovedu pri písaní programu.

Priečinok so zdrojovými kódmi `pico-kit-exercises` je možné otvoriť vo vývojovom prostredí Visual Studio Code a v ňom programy zostaviť a odladzovať tak, ako to bolo uvedené v časti 1.4.6.

6.3 Zapracovanie pripomienok vyučujúcich do laboratórnych úloh

Pripomienky k zadaniam laboratórnych úloh boli priebežne zapracované do konečnej verzie úloh. Pripomienky sa zväčša týkali zvolenia správnej metodiky a pridania stručného popisu laboratórnych úloh pre uľahčenie prípravy vyučujúcich na laboratórne cvičenia.

Úlohy boli spočiatku vypracované za použitia C/C++ SDK, čo však bolo neskôr prerobené tak, aby boli priamo využívané registre mikrokontroléra.

7 Zhodnotenie dosiahnutých výsledkov

Výsledky tejto práce sa dajú rozdeliť do troch celkov. Návrh vývojového kitu, vytvorenie testovacích a demonštračných programov a návrh zadaní laboratórnych úloh. Dosiahnuté výsledky jednotlivých celkov sú uvedené v nasledujúcom texte a na konci tejto kapitoly je uvedené ich zhrnutie.

7.1 Návrh vývojového kitu

Hlavným výsledkom tejto práce je navrhnutý a zrealizovaný funkčný prototyp vývojového kitu s názvom RPi Pico Kit o rozmeroch $100 \times 100\,\text{mm}$. Jadrom dosky je vývojová doska Raspberry Pi Pico. Kit obsahuje viacero súčastí, medzi ktoré patrí napríklad debugger, základné užívateľské rozhranie, komponenty na základných embedded zberniciach, generátory zvukových výstupov či obrazový výstup pomocou VGA a DVI rozhrania.

Cena súčiastok pre prototypovanie vývojového kitu bola približne 1700 CZK. Cena pre hromadnú výrobu by sa však mohla pohybovať okolo hodnoty 1200 CZK.

Hlavnou prednosťou vývojového kitu je, že jednotlivé HW súčasti kombinuje tak, aby kit poskytoval čo najväčšie množstvo funkcií. Nemenej dôležitou skutočnosťou je aj to, že RPi Pico Kit obsahuje debugger, ktorý umožňuje jednoduché odladzovanie vyvíjaného projektu. Vďaka vyvedeniu binárnych vstupov a výstupov, UART portu a I^2C zbernice, funkcie dosky je možné ďalej rozširovať podľa potreby.

Nedostatkom vývojového kitu je skrat na VGA konektore, ktorý bol na prototype kitu ručne odstránený. Ďalším nedostatkom je zámena vývodov I^2S zbernice pre generovanie zvukového výstupu pomocou PCM modulácie. Kvôli tomu musela byť upravená oficiálna softvérová knižnica poskytovaná spoločnosťou Raspberry Pi. Uvedené nedostatky budú odstránené v ďalšej verzii RPi Pico Kitu.

Do budúcna by bolo vhodné pridať prepäťovú ochranu vo forme transil diódy pre napájanie cez napájací DC jack aj napriek vyznačenému menovitému napájaciemu napätiu na doske plošných spojov.

7.2 Vytvorenie testovacích programov

Ďalej boli vytvorené programy pre otestovanie jednotlivých súčastí, ktoré slúžia aj na demonštráciu ich funkcie. Práca s micro-SD kartou nebola otestovaná, pretože vytvorenie softvérovej knižnice pre prácu s ňou by presahovalo časový rámec diplomovej práce. Za zmienku stojí aj spomenúť prerobenie celej knižnice PicoQVGA pre generovanie obrazového výstupu pre VGA rozhranie spolu s množstvom demonštračných programov.

Nedostatkom demonštračných programov je nemožnosť generovania obrazu cez DVI rozhranie vo vyššej kvalite ako je 640×480 px pri 60 fps. Pre odstránenie tohto nedostatku by bolo potrebné najprv identifikovať jeho príčinu. Za vyskúšanie by stálo zníženie bitovej hĺbky generovaného obrazu, či optimalizácia kódu programu.

Práca by mohla pokračovať vytvorením knižnice pre prácu s micro-SD kartou s podporou bežne používaných súborových systémov. Nasledovať by mohlo prispôsobenie knižnice od spoločnosti Adafruit pre OLED displej, čo by znamenalo lepšie možnosti pre generovanie grafického obsahu displeja. Prispôsobenie knižnice PicoVGA pre prácu s RPi Pico Kitom by umožnilo generovanie obrazu vo vyššej kvalite, čo by umožnilo podporu ďalšieho množstva demonštračných programov vytvorených v knižnici.

7.3 Návrh zadaní laboratórnych úloh

Vytvorený vývojový kit má nájsť svoje použitie najmä vo výuke predmetu Vstavané systémy a mikroprocesory. Navrhnutá teda bola aj sada zadaní piatich laboratórnych úloh pre laboratórne cvičenia tohto predmetu a vzorové riešenia úloh vrátane ich dokumentácie.

Výhodou týchto zadaní je, že obsahujú domácu prípravu pre študentov, ktorá ich prevedie základnými poznatkami a tak pripraví na ďalšiu prácu. To môže v konečnom dôsledku výrazne zvýšiť ich rýchlosť práce na laboratórnych cvičeniach.

Do budúcna by bolo vhodné vytvoriť zadania pre ďalšie laboratórne úlohy. Svoje využitie by mohol nájsť aj podrobný návod pre inštaláciu potrebných programov na bežne používaných operačných systémoch a tiež návod pre prácu s programom Visual Studio Code.

7.4 Zhrnutie dosiahnutých výsledkov

Výsledkom tejto práce teda je vývojový kit RPi Pico Kit, ktorý svoje využitie nájde vo výuke predmetu Vstavané systémy a mikroprocesory. Vytvorené testovacie programy slúžia pre jednoduché otestovanie súčastí kitu a tiež na demonštráciu jeho funkcií. Navrhnuté zadania úloh poslúžia pre výuku laboratórnych cvičení uvedeného predmetu.

Pre jednotlivé časti práce boli popísané výhody, nedostatky ako aj ich možné riešenia. Spomenuté možnosti pre vylepšenia môžu slúžiť ako námet pre zadania ďalších záverečných prác.

Na základe uvedených skutočností autor práce považuje inováciu laboratórnych úloh spomenutého predmetu, ako hlavný cieľ tejto práce, za splnený.

Záver

Cieľom tejto práce bolo zmodernizovanie laboratórnych úloh predmetu Vstavané systémy a mikroprocesory za použitia vývojovej dosky Raspberry Pi Pico. Aby to však bolo možné, najprv bolo nutné naštudovať si dokumentáciu a možnosti vývojovej dosky. Začiatok práce bol teda venovaný oboznámeniu sa so základnými vlastnosťami dosky a mikrokontroléra RP2040, ktorý je jej jadrom. Boli popísané jeho základné vlastnosti a taktiež bola na pár príkladoch ukázaná práca s doskou.

Ďalej bola vysvetlená potreba inovácie predmetu Vstavané systémy a mikroprocesory. Aby bolo možné vhodne navrhnúť nové úlohy pre laboratórne cvičenia predmetu, musela byť naštudovaná jeho osnova a zadania súčasných laboratórnych úloh. Následne boli stručne navrhnuté nové zadania laboratórnych úloh pre tento predmet.

Hlavným cieľom práce bol návrh konceptu HW platformy, ktorá by sa využívala na spomenutom predmete. Navrhnutý vývojový kit RPi Pico Kit obsahuje viacero súčastí, medzi ktoré patrí napríklad debugger alebo základné rozhranie pre interakciu s užívateľom v podobe 8 LED diód, 8 tlačidiel, enkodéra a potenciometra. Pre obrazový výstup je možné využiť OLED displej alebo počítačový monitor pripojený cez VGA alebo DVI rozhranie. Generovanie tónov zabezpečuje piezoelektrický menič. K dispozícii je aj plnohodnotný dvojkanálový zvukový výstup generovaný PWM alebo PCM moduláciou. Užívateľské konfiguračné dáta do veľkosti 4 kB je možné pre navrhnuté programy uložiť do EEPROM pamäte. Väčšie súbory sa dajú uložiť na micro-SD kartu, pre ktorú je na vývojovom kite osadený príslušný slot. Vývojový kit využíva komponenty na základných embedded zberniciach, ako je napríklad UART, SPI alebo I^2C.

Pre otestovanie jednotlivých súčastí návrhu konceptu HW platformy boli vytvorené testovacie programy, ktoré môžu rovnako poslúžiť pre demonštráciu funkcií vývojového kitu. Potrebné bolo prerobiť celú knižnicu PicoQVGA, aby pomocou nej bolo možné generovať obrazový výstup pre VGA rozhranie. Zistené nedostatky pri testovaní boli priebežne zapracované.

Práca ďalej pokračovala výberom konkrétnych súčiastok a ich puzdier a návrhom dosky plošných spojov. Výsledkom je 4-vrstvová DPS s rozmermi 100×100 mm, pričom jej kompletná výrobná dokumentácia, podľa ktorej bola vyrobená, je k dispozícii v elektronickej prílohe tejto práce. Vyrobená DPS bola následne ručne osadená, oživená a otestovaná. Zistené boli iba dva nedostatky, ktoré však do určitej miery boli odstránené. Ostatné súčasti fungovali podľa očakávaní.

V ďalšej časti práce bola navrhnutá sada zadaní piatich laboratórnych úloh pre laboratórne cvičenia predmetu Vstavané systémy a mikroprocesory. Prvá úloha oboznámi študentov s inštručnou sadou ARMv6-M, ktorú mikrokontrolér využíva.

V ďalšej úlohe sa naučia obsluhovať binárne vstupy a výstupy. V tretej úlohe prácu s výstupmi obohatia o prácu s časom mikrokontroléra, vďaka čomu rozblikajú LED diódy viacerými možnými spôsobmi. Ďalej sa naučia využívať periféria A/D prevodníka pre meranie elektrockého napätia a napokon v poslednej úlohe si naprogramujú komunikáciu cez UART rozhranie.

K jednotlivým úloham boli vytvorené aj vzorové riešenia vrátane ich dokumentácie. Pripomienky od vyučujúcich predmetu boli priebežne zapracované do zadania úloh.

Výsledkom práce je navrhnutý vývojový kit, spolu s demonštračnými programami a zadaniami úloh pre laboratórne cvičenia predmetu. Vývojový kit poskytuje veľké množstvo funkcií a výrazne presahuje potreby laboratórnych cvičení predmetu Vstavané systémy a mikroprocesory, pre ktoré je predovšetkým určený. Využitý tak môže byť aj na ďalšom nadväzujúcom predmete Vstavané systémy, či dokonca na predmete Subsystémy PC vďaka svojmu VGA, DVI a USB rozhraniu.

Na záver boli v kapitole 7 zhrnuté prínosy práce ale tiež jej nedostatky spolu s ich možným riešením. Medzi možné vylepšenia patrí napríklad rozšírenie podpory ďalších knižníc pre vývojovú dosku Raspberry Pi Pico ako je napríklad knižnica pre prácu so súborovými systémami na SD karte alebo vylepšenie grafických možností pre obraz OLED displeja. Prerobenie knižnice PicoVGA by umožnilo generovanie obrazu vo vyššej kvalite ako umožňuje použitá knižnica PicoQVGA. Rovnako by sa dalo vylepšiť rozlíšenie obrazu pre DVI rozhranie. Spomenuté možnosti pre vylepšenia môžu slúžiť ako námet pre zadania semestrálnych projektov či ďalších záverečných prác.

Autor prácu prezentoval na študentskej konferencii EEICT 2022, kde táto práca bola tiež publikovaná formou krátkeho článku. Autor má ďalej v pláne navrhnutý vývojový kit ako aj testovacie programy k nemu ďalej publikovať ako open-source projekty na platforme GitHub [44], kde na ďalšom vývoji a vylepšovaní budú môcť okrem autora pracovať aj iní vývojári.

Vďaka sprístupneniu všetkej výrobnej dokumentácie, si prípravok bude môcť vyrobiť ktokoľvek a následne využiť pre svoju potrebu.

Naštudovanie možností vývojovej dosky Raspberry Pi Pico, ako aj práce s ňou, návrh konceptu HW a testovacích programov, zrealizovanie finálnej revízie prípravku, návrh zadaní laboratórnych úloh a v neposlednom rade písanie textu práce autora zamestnali na toľko, ako keby na projekte pracoval na polovičnom úväzku od začiatku školského roka do odovzdania práce.

Literatúra

[1] HARRIS, S. L.; HARRIS, D. M., *Digital design and computer architecture: ARM edition*, Waltham, MA: Morgan Kaufman, 2016, 559 s. ISBN 978-0-12-800056-4.

[2] SMITH, S., *RP2040 Assembly Language Programming: ARM Cortex-M0+ on the Raspberry Pi Pico*, Apress, 2021, 344 s. ISBN 978-1-4842-7752-2.

[3] HRBÁČEK, J., *Komunikace mikrokontroléru s okolím. 1. díl*, Praha: BEN, 1999, 159 s. ISBN 80-86056-42-2.

[4] ŠANDERA, J., *Návrh plošných spojů pro povrchovou montáž*, Praha: BEN, 2006, 270 s. ISBN 80-7300-181-0.

[5] PONČÁK, M., *Rozšíření překladače jazyka C o podporu dalších embedded mikroprocesorů*, VUT FEKT Brno, 2020, [cit. 28. 12. 2021]. Dostupné z URL: <`http://hdl.handle.net/11012/194889`>.

[6] MIKESKA, J., *Laboratorní úlohy pro mikrokontroléry MC9S08LH firmy Freescale*, VUT FEKT Brno, 2013, [cit. 28. 12. 2021]. Dostupné z URL: <`http://hdl.handle.net/11012/27989`>.

[7] Raspberry Pi, [online katalogový list], *Raspberry Pi Pico product brief*. [cit. 29. 12. 2021]. Dostupné z URL: <`https://datasheets.raspberrypi.com/pico/pico-product-brief.pdf`>.

[8] Raspberry Pi, [online katalogový list], *RP2040 Datasheet: A microcontroller by Raspberry Pi*. Rev. 1.7.1, 11/2021, [cit. 27. 12. 2021]. Dostupné z URL: <`https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf`>.

[9] Raspberry Pi, [online katalogový list], *Raspberry Pi Pico Datasheet: An RP2040-based microcontroller board*. Rev. 1.7.1, 11/2021, [cit. 29. 12. 2021]. Dostupné z URL: <`https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf`>.

[10] Raspberry Pi, [online katalogový list], *Getting started with Raspberry Pi Pico: C/C++ development with Raspberry Pi Pico and other RP2040-based microcontroller boards*. Rev. 1.7.1, 11/2021, [cit. 27. 12. 2021]. Dostupné z URL: <`https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf`>.

[11] Raspberry Pi, [online katalogový list], *Raspberry Pi Pico C/C++ SDK: Libraries and tools for C/C++ development on RP2040 microcontrollers*. Rev.

1.7.1, 11/2021, [cit. 27. 12. 2021]. Dostupné z URL: <https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-c-sdk.pdf>.

[12] Raspberry Pi, [online katalogový list], *Raspberry Pi Pico Python SDK: A MicroPython environment for RP2040 microcontrollers*. Rev. 1.7.1, 11/2021, [cit. 27. 12. 2021]. Dostupné z URL: <https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-python-sdk.pdf>.

[13] Raspberry Pi, [online katalogový list], *Hardware design with RP2040: Using RP2040 microcontrollers to build boards and products*. Rev. 1.7.1, 11/2021, [cit. 27. 12. 2021]. Dostupné z URL: <https://datasheets.raspberrypi.com/rp2040/hardware-design-with-rp2040.pdf>.

[14] ARM, [online katalogový list], *Cortex-M0+ Devices Generic User Guide*. Rev. 2.0, 12/2012, [cit. 27. 12. 2021]. Dostupné z URL: <https://developer.arm.com/documentation/dui0662/b>.

[15] ARM, [online katalogový list], *ARMv6-M Architecture Reference Manual*. Rev. E, 06/2018, [cit. 27. 12. 2021]. Dostupné z URL: <https://developer.arm.com/documentation/ddi0419/e>.

[16] Microchip, [online katalogový list], *MCP23008/MCP23S08*. Rev. F, 03/2019, [cit. 16. 05. 2022]. Dostupné z URL: <https://ww1.microchip.com/downloads/en/DeviceDoc/MCP23008-MCP23S08-Data-Sheet-20001919F.pdf>.

[17] ON Semiconductor, [online katalogový list], *CAT24C32: EEPROM Serial 32-Kb I2C*. Rev. 26, 05/2018, [cit. 16. 05. 2022]. Dostupné z URL: <https://www.onsemi.com/pdf/datasheet/cat24c32-d.pdf>.

[18] Chipquik, [online katalogový list], *Thermally Stable Solder Paste TS391SNL50*. Rev. 1.1, [cit. 18. 05. 2022]. Dostupné z URL: <https://www.chipquik.com/datasheets/TS391SNL50.pdf>.

[19] ADAMS, J., *Meet Raspberry Silicon: Raspberry Pi Pico now on sale at $4*, [cit. 30. 12. 2021]. Dostupné z URL: <https://www.raspberrypi.com/news/raspberry-pi-silicon-pico-now-on-sale>.

[20] ADAMS, J., *Raspberry Pi RP2040: Our Microcontroller for the Masses*, [cit. 30. 12. 2021]. Dostupné z URL: <https://www.arm.com/blogs/blueprint/raspberry-pi-rp2040>.

[21] *Turning Raspberry Pi Pico into an SWD Debug Probe with Picprobe*, [cit. 30. 12. 2021]. Dostupné z URL: <https://visualgdb.com/tutorials/raspberry/pico/picoprobe/>.

[22] VUT v Brně, *Detail předmětu: Vestavné systémy a mikroprocesory*, Ak. rok: 2020/2021, [cit. 28. 12. 2021]. Dostupné z URL: <https://www.vut.cz/studenti/predmety/detail/224207>.

[23] *TWR-S08LL64, MC9S08LL 8-bit Segment LCD Tower System Module based on MC9S08LL64 MCU: Reference Design using part MC9S08LL64CL by NXP Semiconductors*, [cit. 28. 12. 2021]. Dostupné z URL: <https://www.arrow.com/en/reference-designs/twr-s08ll64-mc9s08ll-8-bit-segment-lcd-tower-system-module-based-on-mc9s08ll64-mcu/6ee1bfbca240eed4ebf6418f2a66191b>.

[24] POUNDER L., *Best Raspberry Pi Pico Accessories and Add-Ons 2022*, [cit. 14. 05. 2022]. Dostupné z URL: <https://www.tomshardware.com/best-picks/best-raspberry-pi-pico-accessories>.

[25] *The perfect multi-button input resistor ladder*, [cit. 14. 05. 2022]. Dostupné z URL: <http://www.ignorantofthings.com/2018/07/the-perfect-multi-button-input-resistor.html>.

[26] *Understanding EDID - Extended Display Identification Data*, [cit. 15. 05. 2022]. Dostupné z URL: <https://www.extron.com/article/uedid>.

[27] SHIRRIFF K., *Reading a VGA monitor's configuration data with I2C and a PocketBeagle*, [cit. 16. 05. 2022]. Dostupné z URL: <http://www.righto.com/2018/03/reading-vga-monitors-configuration-data.html>.

[28] NĚMEČEK M., *PicoVGA - displej VGA/TV na Raspberry Pico*, [cit. 16. 05. 2022]. Dostupné z URL: <http://www.breatharian.eu/hw/picovga/index.html>.

[29] NĚMEČEK M., *PicoQVGA - minimalistický displej QVGA na Raspberry Pico*, [cit. 31. 12. 2021]. Dostupné z URL: <http://www.breatharian.eu/hw/picoqvga/index.html>.

[30] Digi-Key Electronics, *PCB Trace Width Calculator*, [cit. 18. 05. 2022]. Dostupné z URL: <https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-pcb-trace-width>.

[31] PCBWay, *Impedance Calculation*, [cit. 17. 05. 2022]. Dostupné z URL: <https://www.pcbway.com/pcb_prototype/impedance_calculator.html>.

[32] JLCPCB, *Capabilities*, [cit. 17. 05. 2022]. Dostupné z URL: <https://jlcpcb.com/capabilities/Capabilities>.

[33] JLCPCB, *Multilayer high precision PCB's with impedance control*, [cit. 17. 05. 2022]. Dostupné z URL: <https://cart.jlcpcb.com/impedance>.

[34] JLCPCB, *How to generate Gerber and Drill files in KiCad 5*, [cit. 17. 05. 2022]. Dostupné z URL: <https://support.jlcpcb.com/article/149-how-to-generate-gerber-and-drill-files-in-kicad>.

[35] WREN L., *Bitbanged DVI on the RP2040 Microcontroller*, [cit. 31. 12. 2021]. Dostupné z URL: <https://github.com/Wren6991/PicoDVI>.

[36] Raspberry Pi, *Raspberry Pi Pico SDK*, [cit. 28. 12. 2021]. Dostupné z URL: <https://github.com/raspberrypi/pico-sdk>.

[37] Raspberry Pi, *Raspberry Pi Pico SDK Examples*, [cit. 28. 12. 2021]. Dostupné z URL: <https://github.com/raspberrypi/pico-examples>.

[38] Raspberry Pi, *Raspberry Pi Pico Extra Library*, [cit. 16. 05. 2022]. Dostupné z URL: <https://github.com/raspberrypi/pico-extras>.

[39] Raspberry Pi, *Raspberry Pi Pico Playground*, [cit. 16. 05. 2022]. Dostupné z URL: <https://github.com/raspberrypi/pico-playground>.

[40] GREIMAN B., *Arduino FAT16/FAT32 exFAT Library*, [cit. 16. 05. 2022]. Dostupné z URL: <https://github.com/greiman/SdFat>.

[41] SCHRAMM D., *Raspberry Pi Pico SSD1306 library*, [cit. 16. 05. 2022]. Dostupné z URL: <https://github.com/daschr/pico-ssd1306>.

[42] NĚMEČEK M., *VGA/TV display on Raspberry Pico*, [cit. 16. 05. 2022]. Dostupné z URL: <https://github.com/Panda381/PicoVGA>.

[43] NĚMEČEK M., *Minimalistic QVGA Display on Raspberry Pico*, [cit. 16. 05. 2022]. Dostupné z URL: <https://github.com/Panda381/PicoQVGA>.

[44] PONČÁK M., *Github Repositories*, [cit. 17. 05. 2022]. Dostupné z URL: <https://github.com/999matej999>.

[45] Wikipedia, The free encyclopedia, *Background debug mode interface*, [cit. 28. 12. 2021]. Dostupné z URL: <https://en.wikipedia.org/wiki/Background_debug_mode_interface>.

[46] Wikipedia, The free encyclopedia, *Operating temperature*, [cit. 29. 12. 2021]. Dostupné z URL: <https://en.wikipedia.org/wiki/Operating_temperature>.

[47] Wikipedia, The free encyclopedia, *API*, [cit. 29. 12. 2021]. Dostupné z URL: <https://cs.wikipedia.org/wiki/API>.

[48] Wikipedia, The free encyclopedia, *Resistor ladder*, [cit. 15. 05. 2022]. Dostupné z URL: <https://en.wikipedia.org/wiki/Resistor_ladder>.

[49] *Metoda RLE*, [cit. 16. 05. 2022]. Dostupné z URL: <http://www.cs.vsb.cz/benes/vyuka/pte/texty/komprese/ch02s01.html>.

[50] *OpAmp driven RX/TX activity LED for UART*, [cit. 14. 05. 2022]. Dostupné z URL: <https://www.edaboard.com/threads/opamp-driven-rx-tx-activity-led-for-uart.350384/>.

[51] *0.96"128x64 OLED displej, I2C, modro/žlutý*, [cit. 30. 12. 2021]. Dostupné z URL: <https://www.laskakit.cz/oled-displej-modry-a-zluty-128x64-0-96--i2c/>.

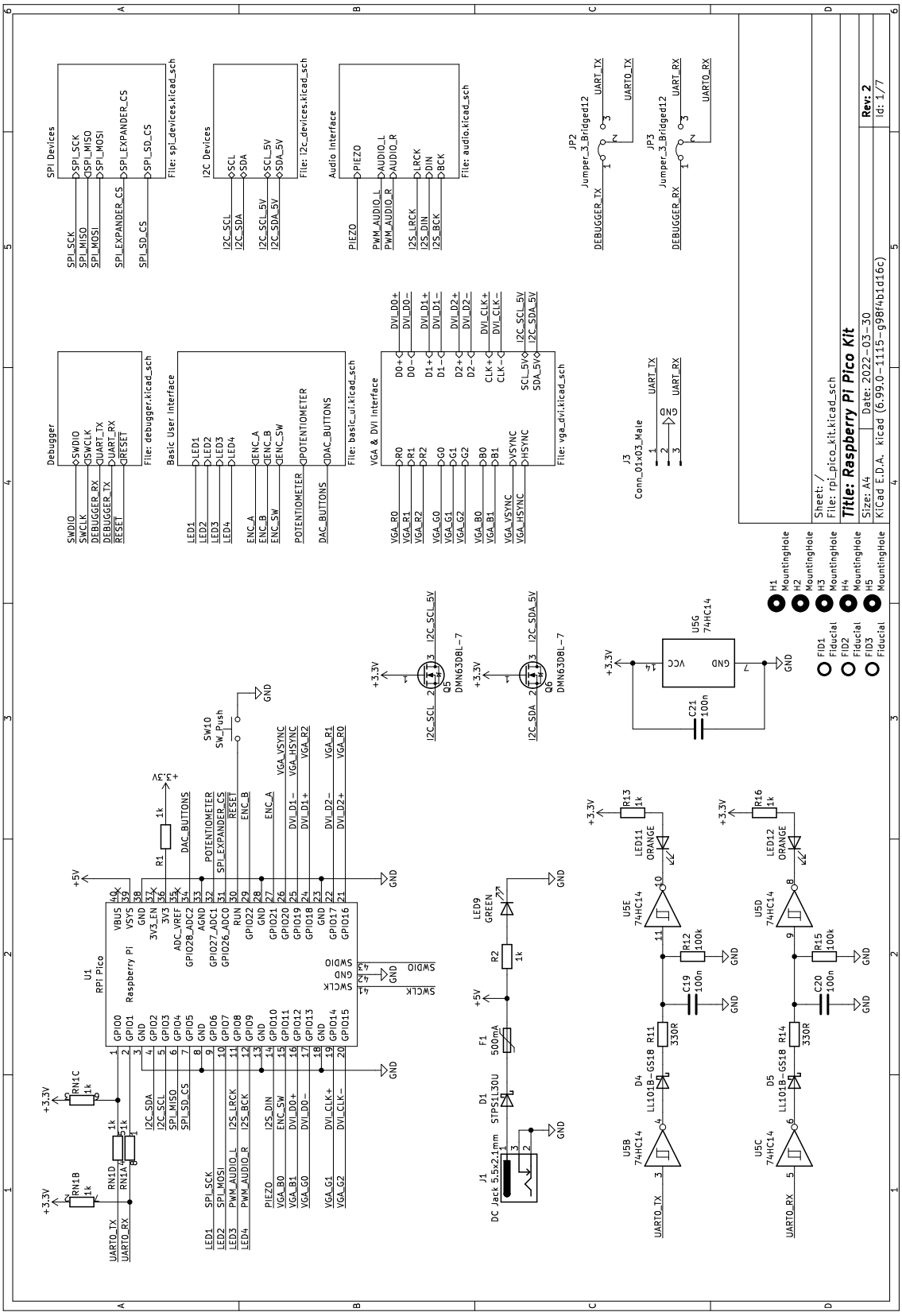
[52] GAJDUSEK J., *TWILight*, [cit. 15. 05. 2022]. Dostupné z URL: <https://github.com/atx/TWILight/blob/master/pcb/i2c-vga.pdf>.

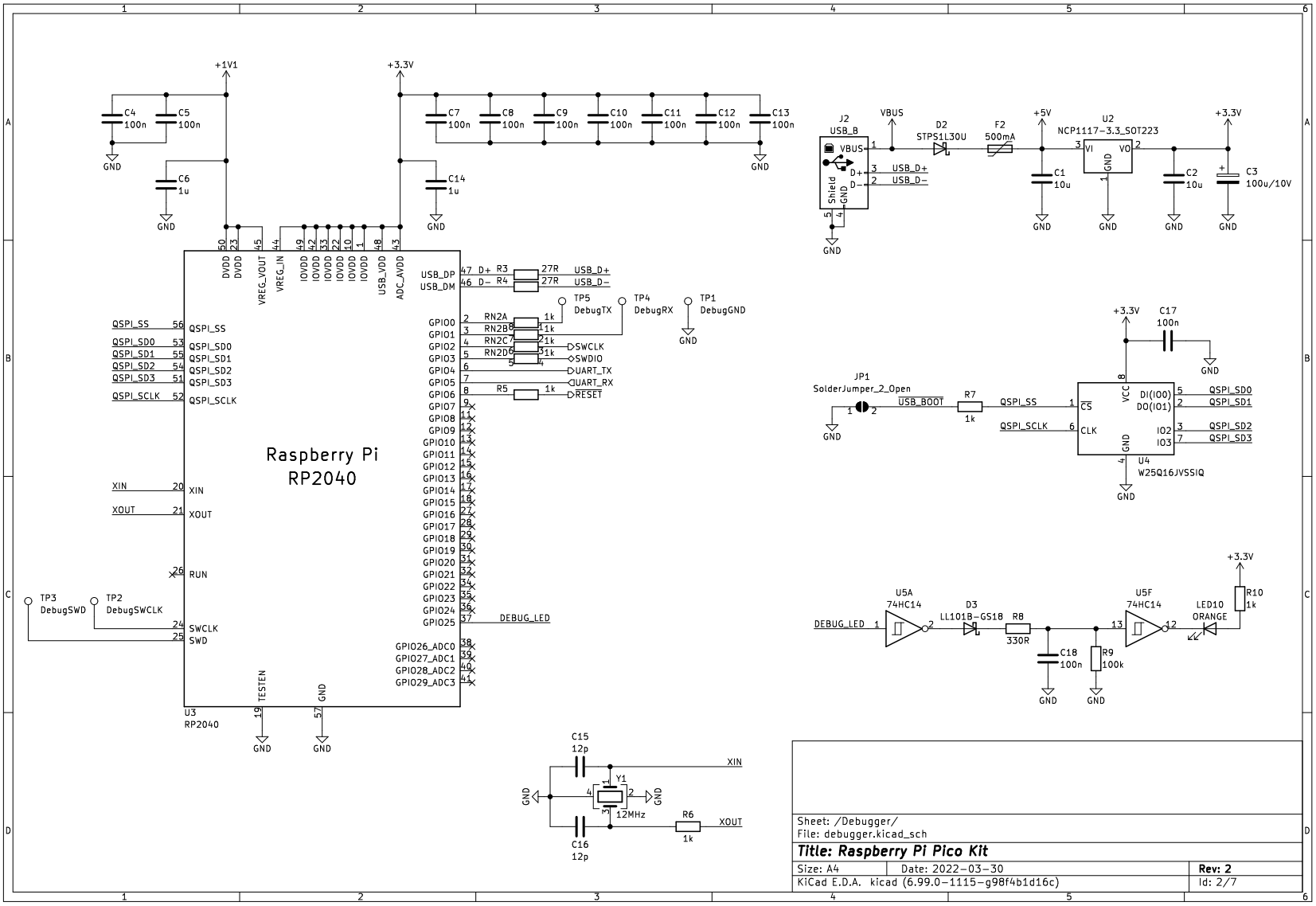
Zoznam symbolov a skratiek

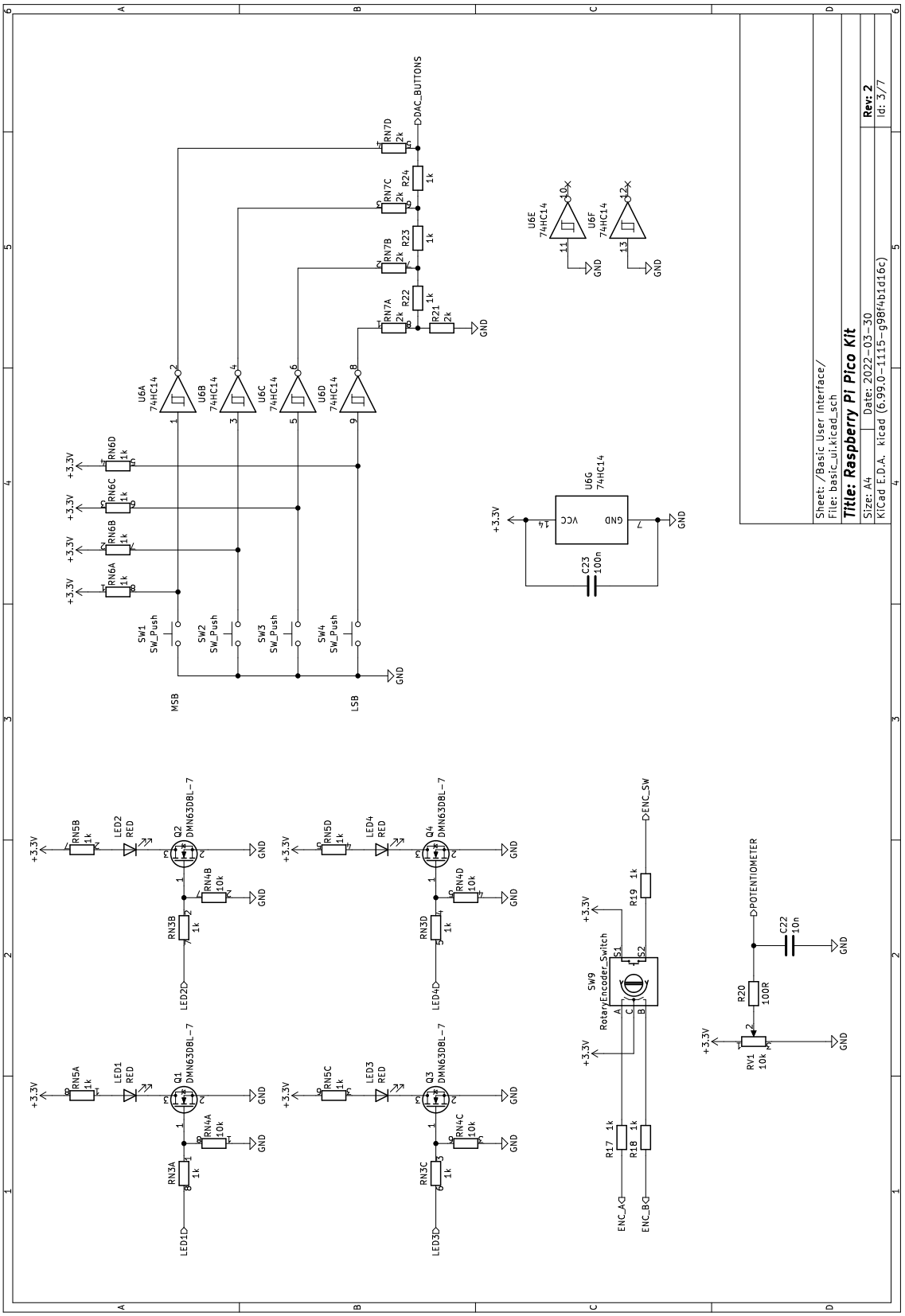
PLL	Phase-locked loop – Fázový záves
SIO	Single-cycle I/O block – Jednocyklový vstupno-výstupný blok
XIP	Execute-in-place – Subsystém určený na priame vykonávanie kódu z (Q)SPI rozhrania
GPIO	General-purpose input/output – Univerzálny vstupný/výstupný pin
RTC	Real-time clock – Hodiny reálneho času
PWM	Pulse-width modulation – Impulzová šírková modulácia
DMA	Direct memory access – Priamy prístup do pamäte
PIO	Programmable I/O – Programovateľné vstupno-výstupné rozhranie
SDK	Software Development Kit – Súbor nástrojov umožňujúci vývoj softvéru pre určitú platformu
EDID	Extended Display Identification Data – Rozšírené identifikačné dáta displeja
THT	Through-hole technology – Vývodové súčiastky
SMD	Surface-mount device – Súčiastky určené pre povrchovú montáž
PCB	Printed circuit board – Doska plošných spojov
DRC	Design Rule Checking – Program na kontrolu splenia požiadavok pre výrobu DPS

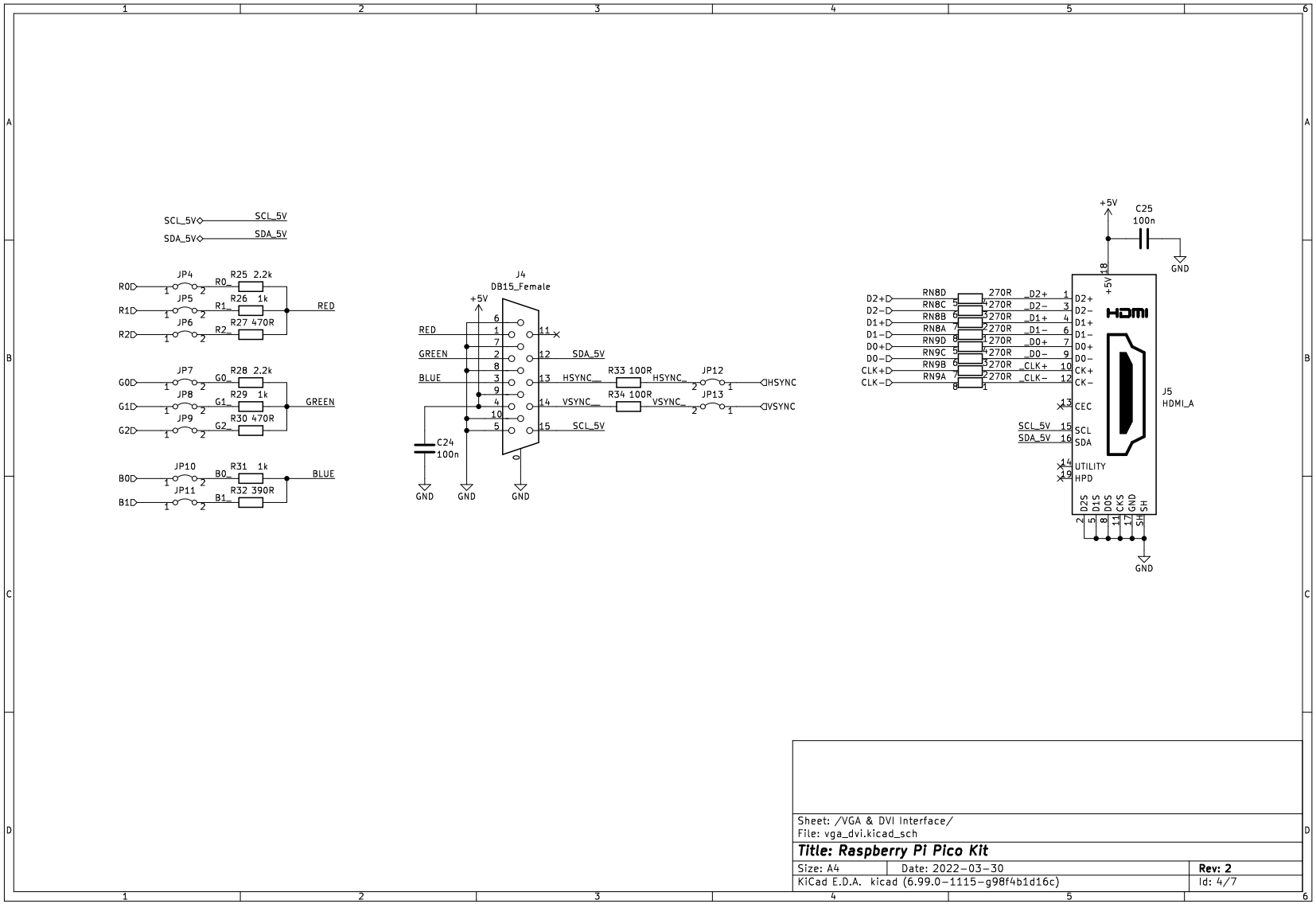
Zoznam príloh

A Schéma zapojenia HW platformy

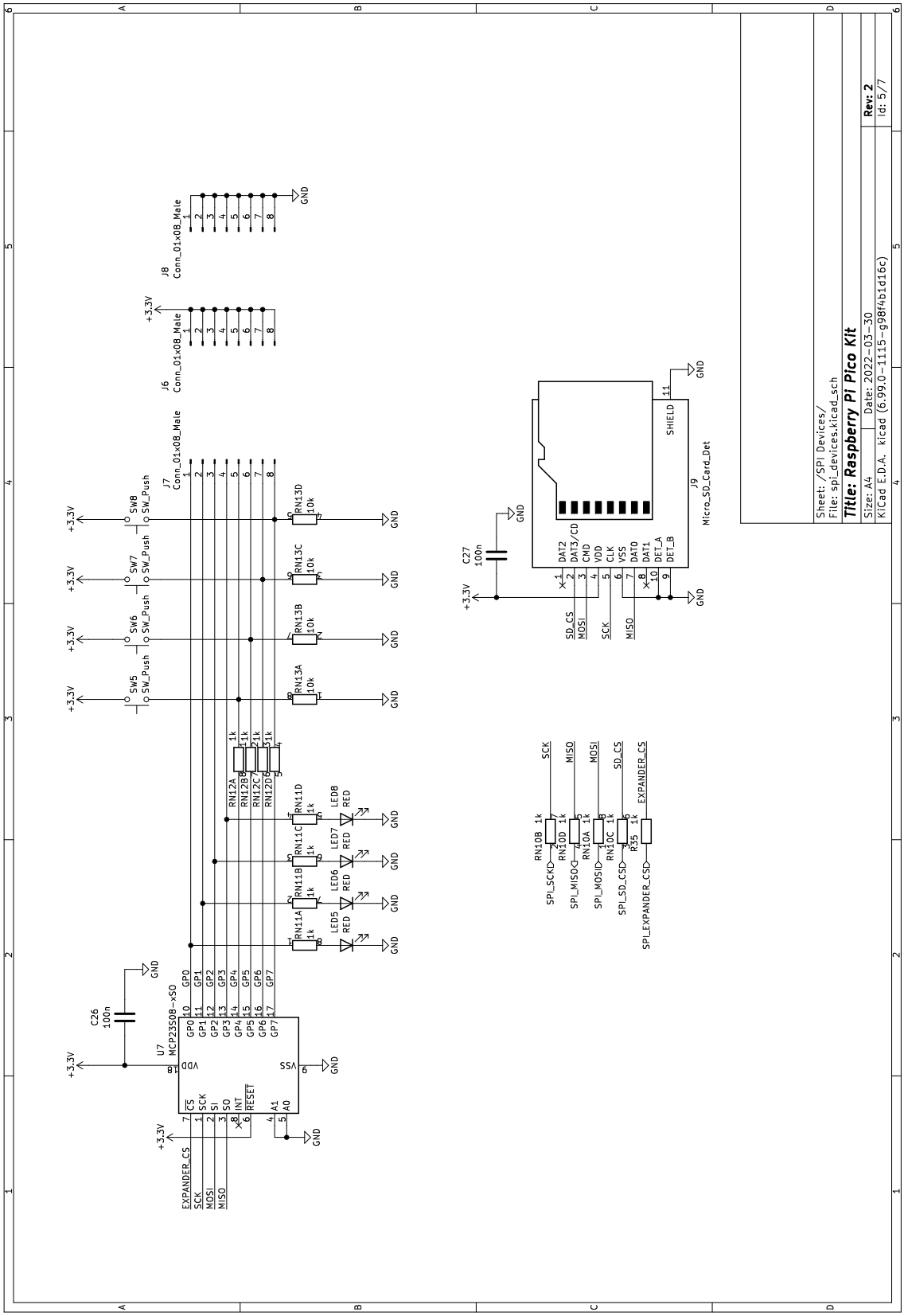


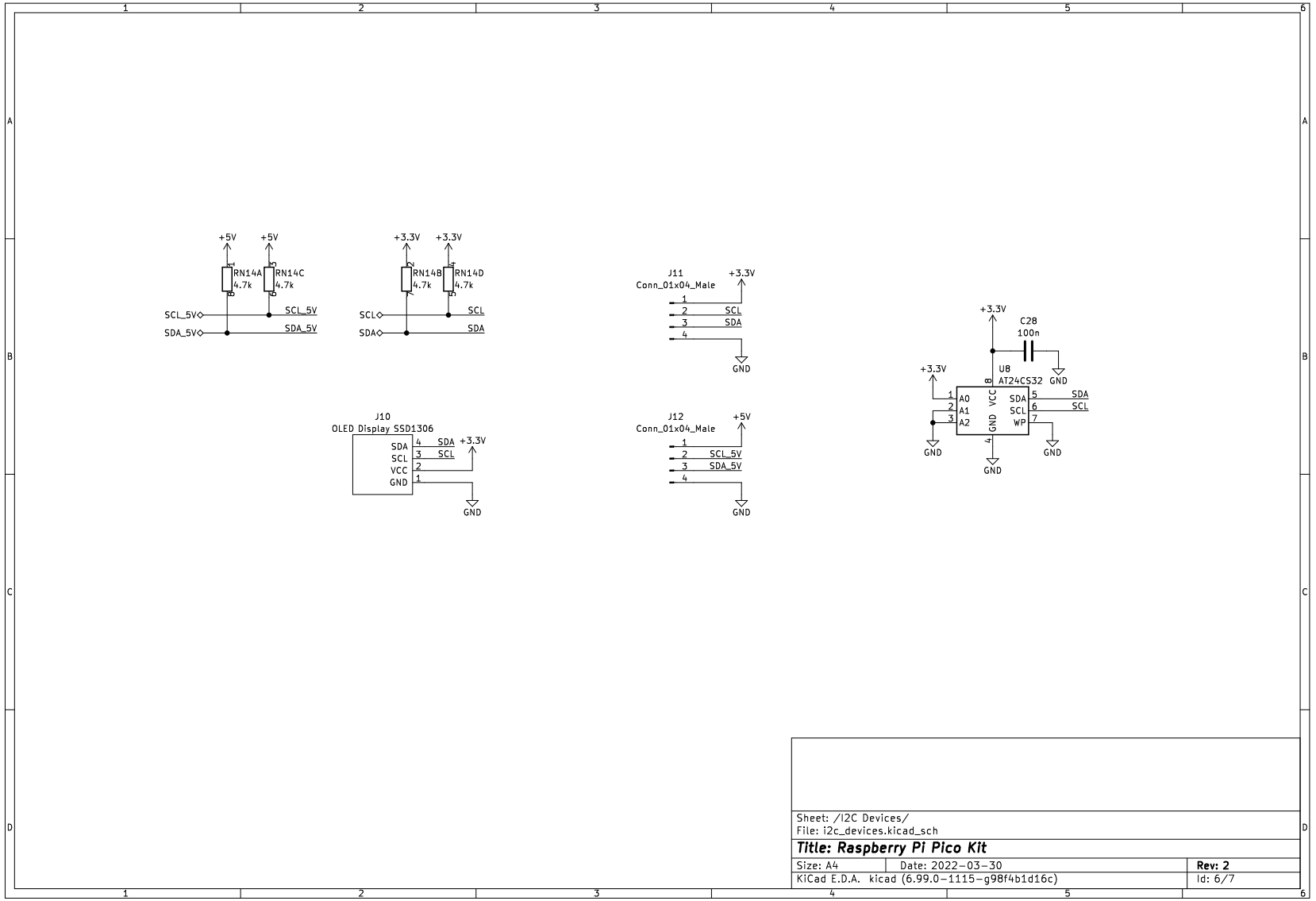

Raspberry Pi
RP2040
U3
RP2040
+1V1
+3.3V
GND
C4 100n
C5 100n
C6 1u
C7 100n
C8 100n
C9 100n
C10 100n
C11 100n
C12 100n
C13 100n
C14 1u
DVDD
VREG_VOUT
VREG_IN
IOVDD
USB_VDD
ADC_AVDD
USB_DP
USB_DM
R3 27R
R4 27R
USB_D+
USB_D-
TP5 DebugTX
TP4 DebugRX
TP1 DebugGND
QSPI_SS
QSPI_SD0
QSPI_SD1
QSPI_SD2
QSPI_SD3
QSPI_SCLK
XIN
XOUT
RUN
SWCLK
SWD
TP3 DebugSWD
TP2 DebugSWCLK
GPIO0
GPIO1
GPIO2
GPIO3
GPIO4
GPIO5
GPIO6
RN2A 1k
RN2B 1k
RN2C 1k
RN2D 1k
SWCLK
SWDIO
UART_TX
UART_RX
R5 1k
RESET
GPIO7
GPIO8
GPIO9
GPIO10
GPIO11
GPIO12
GPIO13
GPIO14
GPIO15
GPIO16
GPIO17
GPIO18
GPIO19
GPIO20
GPIO21
GPIO22
GPIO23
GPIO24
GPIO25
DEBUG_LED
GPIO26_ADC0
GPIO27_ADC1
GPIO28_ADC2
GPIO29_ADC3
TESTEN
J2
USB_B
VBUS
D+
D-
Shield
USB_D+
USB_D-
D2
STPS1L30U
F2
500mA
U2
NCP1117-3.3_SOT223
VI
VO
C1 10u
C2 10u
C3 100u/10V
C15 12p
C16 12p
Y1
12MHz
R6 1k
XIN
XOUT
JP1
SolderJumper_2_Open
USB_BOOT
R7 1k
C17 100n
U4
W25Q16JVSSIQ
DI(IO0)
DO(IO1)
IO2
IO3
CS
CLK
U5A
74HC14
U5F
74HC14
D3
LL101B-GS18
R8 330R
C18 100n
R9 100k
LED10
ORANGE
R10 1k
Sheet: /Debugger/
File: debugger.kicad_sch
Title: Raspberry Pi Pico Kit
Size: A4
Date: 2022-03-30
Rev: 2
KiCad E.D.A. kicad (6.99.0-1115-g98f4b1d16c)
Id: 2/7

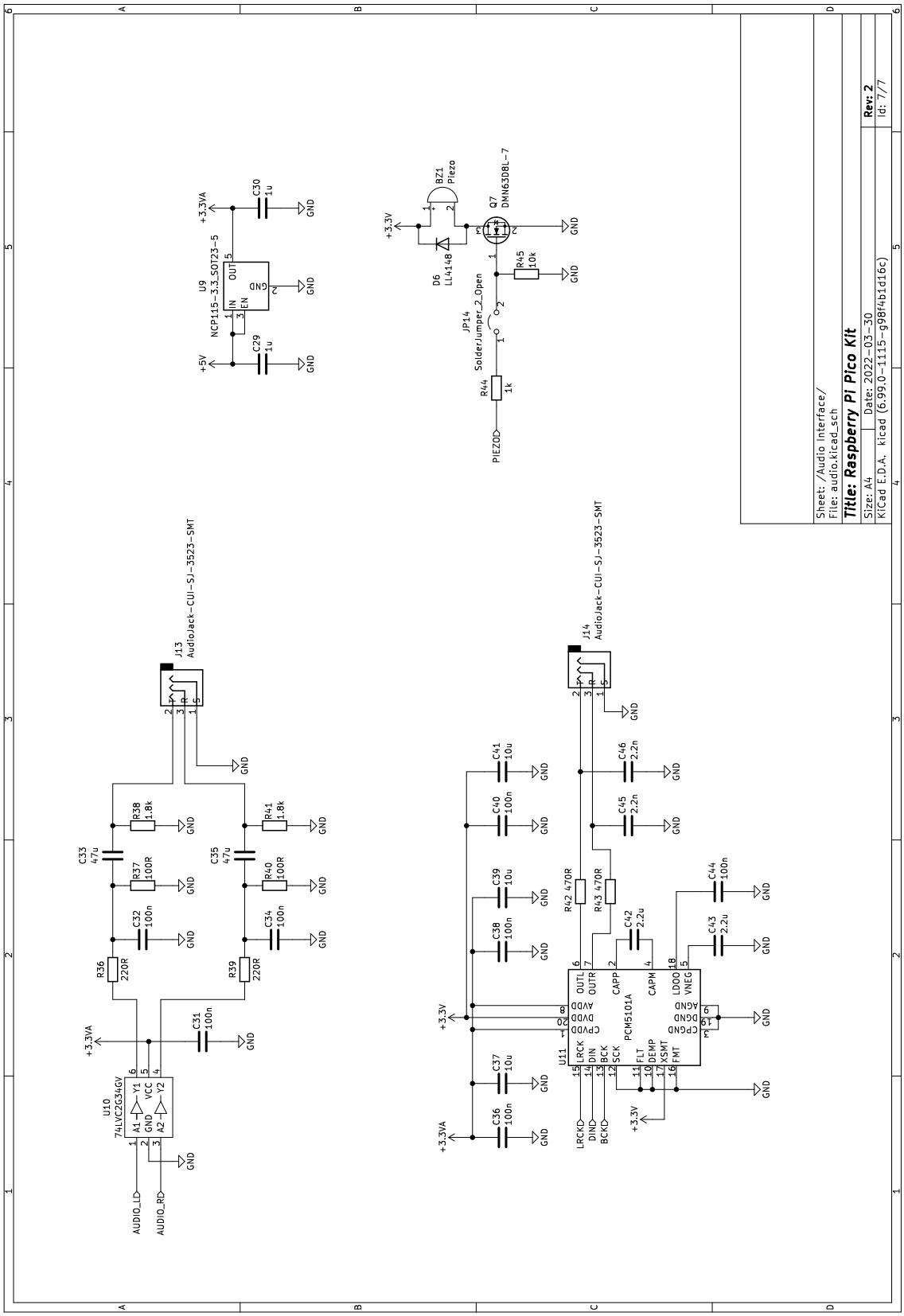

Sheet: /Basic User Interface/
File: basic_ui.kicad_sch
Title: Raspberry Pi Pico Kit
Size: A4
Date: 2022-03-30
Rev: 2
KiCad E.D.A. kicad (6.99.0-1115-g98f4b1d16c)
Id: 3/7
DAC_BUTTONS
POTENTIOMETER
ENC_SW
ENC_A
ENC_B
MSB
LSB
U6A 74HC14
U6B 74HC14
U6C 74HC14
U6D 74HC14
U6E 74HC14
U6F 74HC14
U6G 74HC14
SW1 SW_Push
SW2 SW_Push
SW3 SW_Push
SW4 SW_Push
SW9 RotaryEncoder_Switch
Q1 DMN63D8L-7
Q2 DMN63D8L-7
Q3 DMN63D8L-7
Q4 DMN63D8L-7
LED1 RED
LED2 RED
LED3 RED
LED4 RED
RN3A 1k
RN3B 1k
RN3C 1k
RN3D 1k
RN4A 10k
RN4B 10k
RN4C 10k
RN4D 10k
RN5A 1k
RN5B 1k
RN5C 1k
RN5D 1k
RN6A 1k
RN6B 1k
RN6C 1k
RN6D 1k
RN7A 2k
RN7B 2k
RN7C 2k
RN7D 2k
R17 1k
R18 1k
R19 1k
R20 100R
R21 2k
R22 1k
R23 1k
R24 1k
RV1 10k
C22 10n
C23 100n
+3.3V
GND



Sheet: /VGA & DVI Interface/		
File: vga_dvi.kicad_sch		
Title: Raspberry Pi Pico Kit		
Size: A4	Date: 2022-03-30	**Rev: 2**
KiCad E.D.A. kicad (6.99.0-1115-g98f4b1d16c)		Id: 4/7


Sheet: /SPI Devices/
File: spi_devices.kicad_sch
Title: Raspberry Pi Pico Kit
Size: A4
Date: 2022-03-30
Rev: 2
KiCad E.D.A. kicad (6.99.0-1115-g98f4b1d16c)
Id: 5/7
U7
MCP23S08-xSO
C26
100n
C27
100n
J6
Conn_01x08_Male
J7
Conn_01x08_Male
J8
Conn_01x08_Male
J9
Micro_SD_Card_Det
SW5
SW6
SW7
SW8
SW_Push
RN10A 1k
RN10B 1k
RN10C 1k
RN10D 1k
RN11A 1k
RN11B 1k
RN11C 1k
RN11D 1k
RN12A 1k
RN12B 1k
RN12C 1k
RN12D 1k
RN13A 10k
RN13B 10k
RN13C 10k
RN13D 10k
R35 1k
LED5 RED
LED6 RED
LED7 RED
LED8 RED
+3.3V
GND
EXPANDER_CS
SCK
MOSI
MISO
SD_CS
SHIELD
SPI_SCKD
SPI_MISOD
SPI_MOSID
SPI_SD_CSD
SPI_EXPANDER_CSD


+5V
+5V
RN14A
4.7k
RN14C
4.7k
SCL_5V
SDA_5V
SCL_5V
SDA_5V
+3.3V
+3.3V
RN14B
4.7k
RN14D
4.7k
SCL
SDA
SCL
SDA
J11
Conn_01x04_Male
+3.3V
1
2 SCL
3 SDA
4
GND
J10
OLED_Display_SSD1306
SDA 4 SDA
SCL 3 SCL
VCC 2
GND 1
+3.3V
GND
J12
Conn_01x04_Male
+5V
1
2 SCL_5V
3 SDA_5V
4
GND
+3.3V
C28
100n
GND
+3.3V
U8
AT24CS32
1 A0
2 A1
3 A2
GND 4
VCC 8
SDA 5 SDA
SCL 6 SCL
WP 7
GND
GND
GND
Sheet: /I2C Devices/
File: i2c_devices.kicad_sch
Title: Raspberry Pi Pico Kit
Size: A4
Date: 2022-03-30
Rev: 2
KiCad E.D.A. kicad (6.99.0-1115-g98f4b1d16c)
Id: 6/7

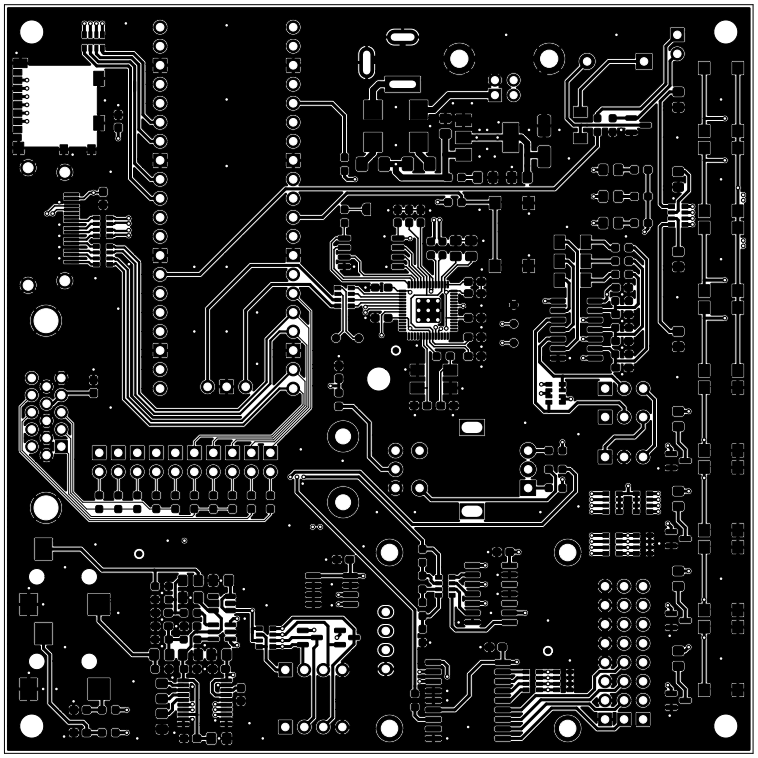

Sheet: /Audio Interface/
File: audio.kicad_sch
Title: Raspberry Pi Pico Kit
Size: A4
Date: 2022-03-30
Rev: 2
KiCad E.D.A. kicad (6.99.0-1115-g98f4b1d16c)
Id: 7/7
U9
NCP115-3.3_SOT23-5
+3.3VA
+5V
C30
1u
C29
1u
GND
BZ1
Piezo
D6
LL4148
Q7
DMN63D8L-7
R45
10k
JP14
SolderJumper_2_Open
R44
1k
PIEZO
J13
AudioJack-CUI-SJ-3523-SMT
J14
AudioJack-CUI-SJ-3523-SMT
R38
1.8k
R41
1.8k
C33
47u
C35
47u
R37
100R
R40
100R
C32
100n
C34
100n
R36
220R
R39
220R
C31
100n
U10
74LVC2G34xGV
AUDIO_LD
AUDIO_RD
C41
10u
C40
100n
C39
10u
C38
100n
C37
10u
C36
100n
R42 470R
R43 470R
C46
2.2n
C45
2.2n
C42
2.2u
C44
100n
C43
2.2u
U11
PCM5101A
OUTL
OUTR
CAPP
CAPM
LDOO
VNEG
AVDD
DVDD
CPVDD
AGND
DGND
CPGND
LRCK
DIN
BCK
SCK
FLT
DEMP
XSMT
FMT
LRCKD
DIND
BCKD

B Zoznam súčiastok HW platformy

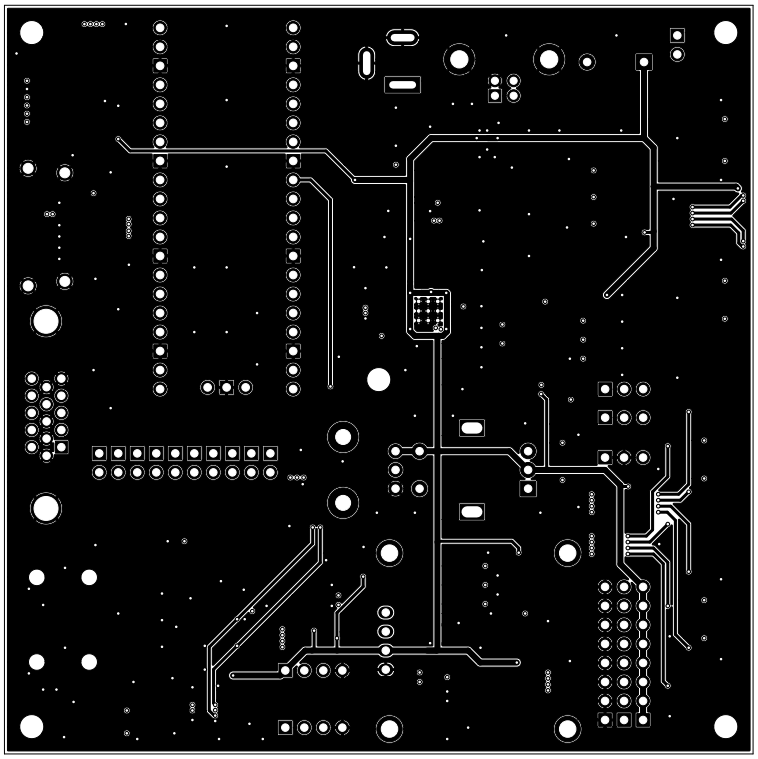
P. č.	Dodávateľ	Označenie výrobcu	Označenie	Ks	Hodnota	Prevedenie
1	Laskakit	LA216001	BZ1	1	Buzzer	12 x 9.5 mm RM 7.6
2	Digikey	CL21A106KOQNNNG	C1, C2, C37, C39, C41	5	10u	0805
3	Digikey	EEE-1AA101WR	C3	1	100u/10V	C, 5 x 5.4 mm
4	Digikey	CL10B104KB8NNWC	C4, C5, C7, C8, C9, C10, C11, C12, C13, C17, C18, C19, C20, C21, C23, C24, C25, C26, C27, C28, C31, C32, C34, C36, C38, C40, C44	27	100n	0603
5	Digikey	C0805C105Z4VAC7800	C6, C14, C29, C30	4	1u	0805
6	Digikey	C0603C120K5HAC7867	C15, C16	2	12p	0603
7	Digikey	CL10B103KB8NNNC	C22	1	10n	0603
8	Digikey	CL21A476MRYNNNE	C33, C35	2	47u	0805
9	Digikey	C0805C225Z4VAC7800	C42, C43	2	2.2u	0805
10	Digikey	0603B222K500CT	C45, C46	2	2.2n	0603
11	TME	STPS1L30U	D1, D2	2	STPS1L30U	SMB
12	Digikey	LL101B-GS18	D3, D4, D5	3	LL101B-GS18	MiniMELF
13	Digikey	LL4148	D6	1	LL4148	MiniMELF
14	Digikey	0ZCJ0050FF2G	F1, F2	2	500mA	1206
15	Digikey	54-00129	J1	1	DC Jack 5.5x2.1mm	THT, Horizontal
16	Digikey	USB-B1HSW6	J2	1	USB type B	THT, Horizontal
17	GME	PinHeader_1x03	J3	1	01x03	Male, THT, P2.54mm, Vertical
18	Digikey	DS1038-15-FBNSiA74	J4	1	DB15_Female	Female, THT, Horizontal
19	Digikey	SS-53000-001	J5	1	HDMI type A	Female, THT, Horizontal
20	GME	PinHeader_1x08	J6, J7, J8	3	01x08	Male, THT, P2.54mm, Vertical
21	Digikey	1040310811	J9	1	Micro-SD slot	SMT, Molex 104031-0811
22	Laskakit	OLED Display SSD1306	J10	1	OLED SSD1306	OLED Display SSD1306
23	GME	PinHeader_1x04	J11, J12	2	01x04	Male, THT, P2.54mm, Vertical
24	Digikey	SJ-3523-SMT-TR	J13, J14	2	CUI-SJ-3523	SMT
25	GME	PinHeader_1x03	JP2, JP3	2	01x03	Male, THT, P2.54mm, Vertical
26	GME	PinHeader_1x02	JP4, JP5, JP6, JP7, JP8, JP9, JP10, JP11, JP12, JP13	10	01x02	Male, THT, P2.54mm, Vertical
27	GME	PinHeader_1x02	JP14	1	01x02	Male, THT, P2.54mm, Vertical
28	Digikey	150080SS75000	LED1, LED2, LED3, LED4, LED5, LED6, LED7, LED8	8	RED, 60 mcd	0805
29	Digikey	150080VS75000	LED9	1	GREEN, 40 mcd	0805
30	Digikey	LTST-C170KFKT	LED10, LED11, LED12	3	ORANGE, 90 mcd	0805
31	Digikey	DMN63D8L-7	Q1, Q2, Q3, Q4, Q5, Q6, Q7	7	DMN63D8L-7	SOT-23

P. č.	Dodávateľ	Označenie výrobcu	Označenie	Ks	Hodnota	Prevedenie
32	Digikey	RMCF0603FT1K00	R1, R2, R5, R6, R7, R10, R13, R16, R17, R18, R19, R22, R23, R24, R26, R29, R31, R35, R44	19	1k	0603
33	Digikey	RMCF0603FT27R0	R3, R4	2	27R	0603
34	Digikey	RMCF0603FT330R	R8, R11, R14	3	330R	0603
35	Digikey	RMCF0603FT100K	R9, R12, R15	3	100k	0603
36	Digikey	RMCF0603FT100R	R20, R33, R34, R37, R40	5	100R	0603
37	Digikey	RMCF0603FT2K00	R21	1	2k	0603
38	Digikey	RMCF0603FT2K20	R25, R28	2	2.2k	0603
39	Digikey	RMCF0603FT470R	R27, R30, R42, R43	4	470R	0603
40	Digikey	RMCF0603FT390R	R32	1	390R	0603
41	Digikey	RMCF0603FT220R	R36, R39	2	220R	0603
42	Digikey	RMCF0603FT1K80	R38, R41	2	1.8k	0603
43	Digikey	RMCF0603FT10K0	R45	1	10k	0603
44	Digikey	YC164-JR-071KL	RN1, RN2, RN3, RN5, RN6, RN10, RN11, RN12	8	1k	1206
45	Digikey	YC164-JR-0710KL	RN4, RN13	2	10k	1206
46	Digikey	YC164-JR-072KL	RN7	1	2k	1206
47	Digikey	YC164-JR-07270RL	RN8, RN9	2	270R	1206
48	Digikey	YC164-JR-074K7L	RN14	1	4.7k	1206
49	Digikey	PTV09A-4025F-B103	RV1	1	10k	PTV09A-1, 25mm
50	GME	TD-03XAT-A00-TR	SW1, SW2, SW3, SW4, SW5, SW6, SW7, SW8, SW10	9	SW_Push	SMT Switch Omron
51	Digikey	EN11-HSM1BF20	SW9	1	Rotary Encoder	EC11E, H20mm
52	Laskakit	Raspberry Pi Pico	U1	1	RPi Pico	Raspberry Pi Pico
53	TME	NCP1117ST33T3G	U2	1	NCP1117-3.3	SOT-223-3
54	Digikey	SC0914(13)	U3	1	RP2040	QFN-56
55	Digikey	W25Q16JVSSIQ	U4	1	W25Q16JVSSIQ	SOIC-8
56	TME	74HC14D.653	U5, U6	2	74HC14	SOIC-14
57	TME	MCP23S08-E/SO	U7	1	MCP23S08	SOIC-18W
58	Digikey	CAT24C32WI-GT3	U8	1	AT24CS32	SOIC-8
59	Digikey	NCP115ASN330T2G	U9	1	NCP115-3.3	SOT-23-5
60	Digikey	74LVC2G34GV,125	U10	1	74LVC2G34GV	SC-74-6
61	Digikey	PCM5101APWR	U11	1	PCM5101A	TSSOP-20
62	Digikey	ABM3B-12.000MHZ-10-1	Y1	1	12MHz	SMT

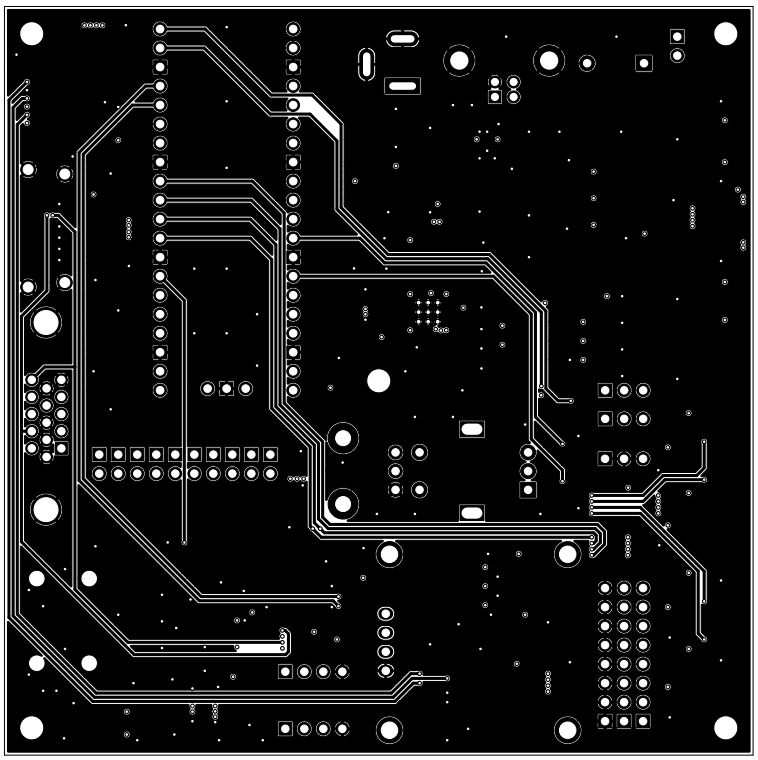
C Obrazce dosky plošných spojov



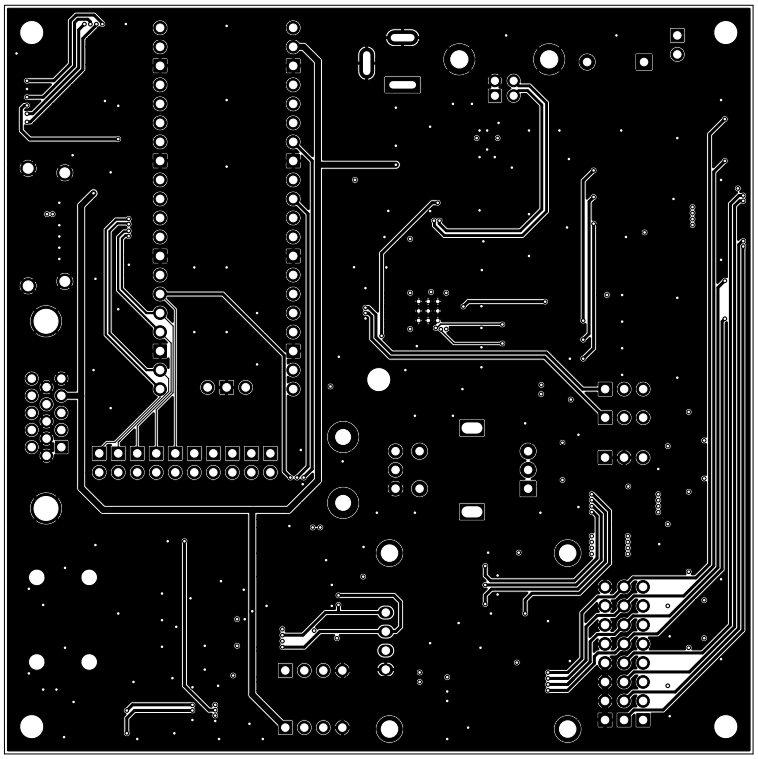
Obr. C.1: Obrazec dosky plošných spojov – horná vrstva



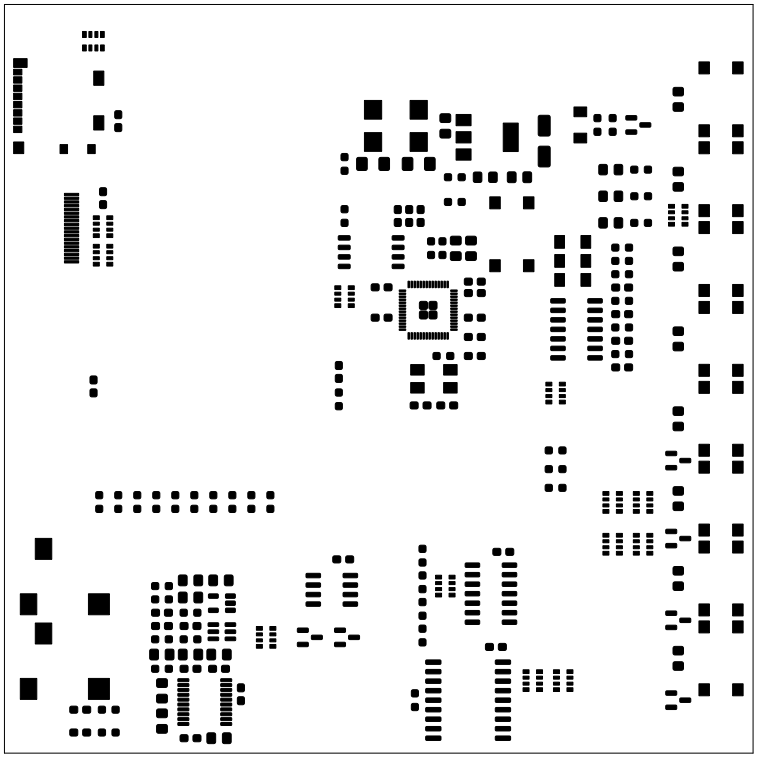
Obr. C.2: Obrazec dosky plošných spojov – vnútorná vrstva č. 1



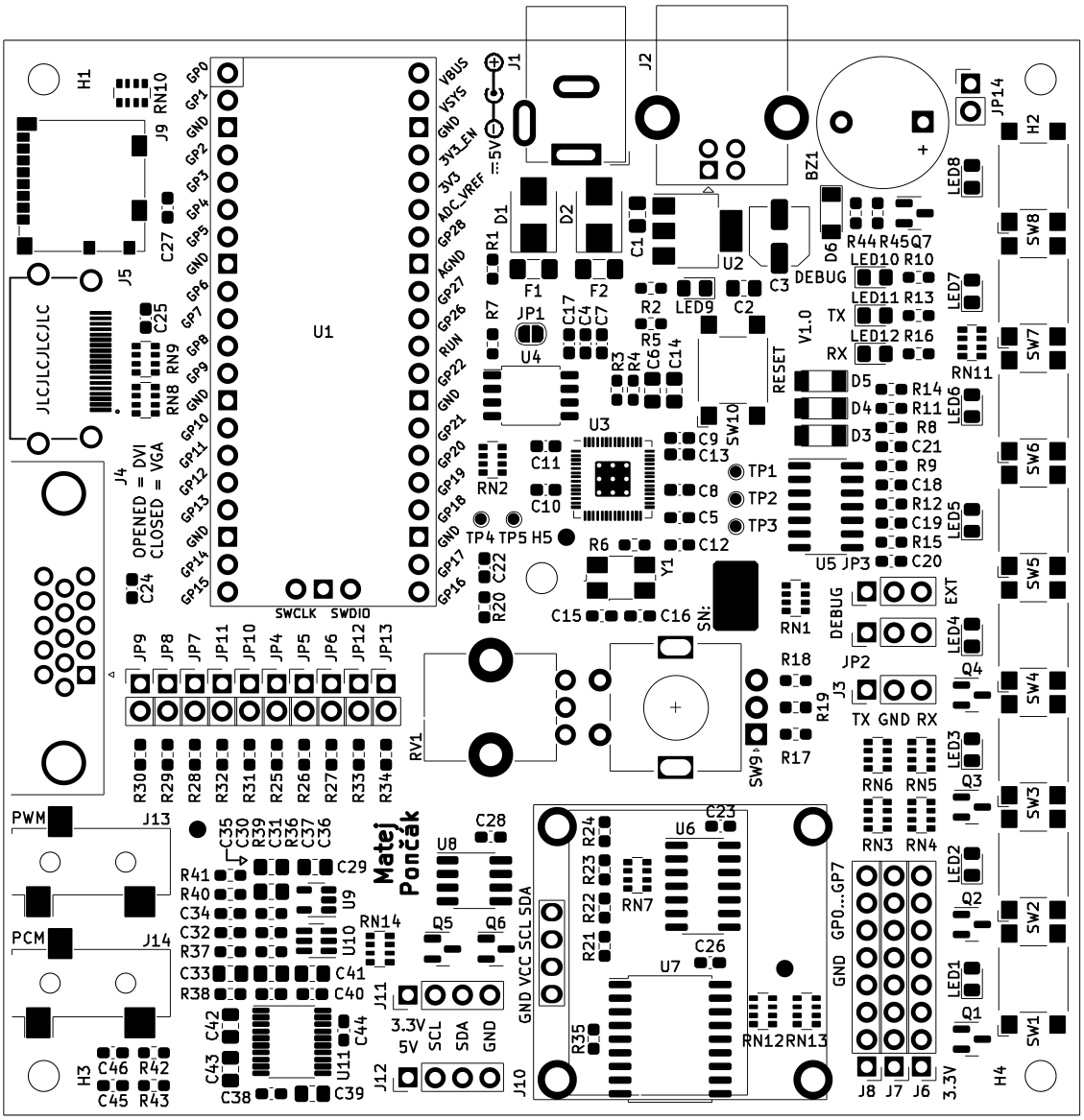
Obr. C.3: Obrazec dosky plošných spojov – vnútorná vrstva č. 2



Obr. C.4: Obrazec dosky plošných spojov – dolná vrstva



Obr. C.5: Obrazec šablóny pre nanesenie spájkovacej pasty



Obr. C.6: Osadzovací plán súčiastok

D Projekt v programe KiCad

Výpis súborov projektu v programe KiCad verzie 6.99 (Nightly Build 2022-02-11), ktoré sa nachádzajú v priečinku `rpi_pico_kit`:

`rpi_pico_kit/`

- `3D_models/` ..... priečinok s 3D modelmi
- `Custom.pretty/` ..... knižnica pre Omron tlačidlá
- `JumperTHT.pretty/` ..... knižnica pre jumpre
- `SS-53000-001/` ..... knižnica pre HDMI konektor
- `SSD1306.pretty/` ..... knižnica pre OLED displej
- `RPi_Pico.pretty/` ..... knižnica pre Raspberry Pi Pico
- `RP2040_minimal.pretty/` ..... knižnica pre mikrokontrolér RP2040
- `rpi_pico_kit.kicad_pro` ..... projekt programu KiCad
- `rpi_pico_kit.kicad_pcb` ..... návrh DPS
- `rpi_pico_kit.kicad_sch` ..... hlavný list schémy
- `debugger.kicad_sch` ..... schéma zapojenia debuggera
- `basic_ui.kicad_sch` ..... schéma zapojenia základného užívateľského rozhrania
- `spi_devices.kicad_sch` ..... schéma zapojenia zariadení na SPI zbernici
- `audio.kicad_sch` ..... schéma zapojenia zvukových obvodov
- `i2c_devices.kicad_sch` ..... schéma zapojenia zariadení na I2C zbernici
- `vga_dvi.kicad_sch` ..... schéma zapojenia VGA a DVI rozhrania

E Gerber súbory pre výrobu dosky plošných spojov

Dokumentácia pre výrobu DPS je zhodovetná vo forme gerber súborov v priečinku `gerbers` a ich význam je nasledovný:

`gerbers/`

- `rpi_pico_kit-B_Cu.gbl` ..... obrazec dolnej medenej vrstvy
- `rpi_pico_kit-B_Mask.gbs` ..... maska dolnej vrstvy
- `rpi_pico_kit-B_Paste.gbp` ..... šablóna pre nanesenie pasty v dolnej vrstve
- `rpi_pico_kit-B_Silkscreen.gbo` ..... potlač dolnej vrstvy
- `rpi_pico_kit-Edge_Cuts.gm1` ..... obrys DPS
- `rpi_pico_kit-F_Cu.gtl` ..... obrazec hornej medenej vrstvy
- `rpi_pico_kit-F_Mask.gts` ..... maska hornej vrstvy
- `rpi_pico_kit-F_Paste.gtp` ..... šablóna pre nanesenie pasty v hornej vrstve
- `rpi_pico_kit-F_Silkscreen.gto` ..... potlač hornej vrstvy
- `rpi_pico_kit-In1_Cu.g2` ..... obrazec vnútornej medenej vrstvy č. 1
- `rpi_pico_kit-In2_Cu.g3` ..... obrazec vnútornej medenej vrstvy č. 2
- `rpi_pico_kit-NPTH-drl_map.gbr` ..... súradnice pre neprekovené otvory
- `rpi_pico_kit-NPTH.drl` ..... súradnice pre neprekovené otvory
- `rpi_pico_kit-PTH-drl_map.gbr` ..... súradnice pre prekovené otvory
- `rpi_pico_kit-PTH.drl` ..... súradnice pre prekovené otvory

F Testovacie programy

Testovacie programy sa nachádzajú v priečinkoch `pico-kit-examples` a `PicoQVGA`. Obsah priečinkov je vypísaný bez súborov `CMakeLists.txt`.

- `/` — koreňový priečinok príloh
  - `pico-kit-examples/` — priečinok s demo programami pre RPi Pico Kit
    - `.vscode/` — priečinok pre nastavenie programu VS Code
      - `launch.json` — nastavenia debuggera
      - `settings.json` — nastavenia programu Visual Studio Code
    - `hw_test/` — priečinok s demonštračnými programami
      - `audio/` — demo programy pre generovanie zvuku
        - `pwm_audio/` — generovanie zvukového PWM výstupu
        - `pcm_audio/` — generovanie zvukového PCM výstupu
        - `piezo/` — ovládanie piezoelektrického meniča
      - `basic_ui/` — demo programy pre prácu s užívateľským rozhraním
        - `leds/` — ovládanie LED diód
        - `quadrature_encoder/` — práca s enkóderom v kvadratúrnom režime
        - `potentiometer/` — ovládanie jasu LED pomocou potenciometra
        - `dac_buttons/` — čítanie stavu tlačidiel pripojených cez DAC
      - `i2c/` — demo programy využívajúce I²C zbernicu
        - `scan_i2c_bus/` — vyhľadávanie zariadení na I²C zbernici
        - `eeprom/` — práca s EEPROM pamäťou
        - `edid_info/` — čítanie EDID metadát
        - `oled/` — grafický výstup na OLED displej
      - `pico_blink/` — otestovanie blikania vstavanej LED na RPi Pico
      - `spi/` — demo programy využívajúce SPI zbernicu
        - `spi_expander/` — obsluha expandera
      - `uart/` — demo programy využívajúce UART rozhranie
        - `uart_loopback/` — funkcia loopbacku
    - `pico_sdk_import.cmake` — pomocný skript pre nájdenie C/C++ SDK
    - `pico_extras_import.cmake` — pomocný skript pre nájdenie pico-extras
  - `PicoQVGA/` — knižnica pre generovanie obrazu na VGA rozhranie
    - `qvga/` — hlavný priečinok knižnice
      - `qvga_lib/` — hlavičkové a zdrojové kódy knižnice
      - `tools/` — pomocné nástroje knižnice
      - `examples/` — demo programy knižnice
    - `pico_sdk_import.cmake` — pomocný skript pre nájdenie C/C++ SDK

G Zadania laboratórnych úloh

V tejto kapitole sú uvedené vytvorené laboratórne úlohy, ktoré využívajú navrhnutú HW platformu RPi Pico Kit.

G.1 Zoznámenie sa s inštrukčnou sadou

Mikrokontrolér RP2040 využíva inštrukčnú sadu ARMv6-M, a preto prvá úloha je venovaná zoznámeniu sa s týmito inštrukciami. Pre úspešné vypracovanie úlohy je potrebné ovládať základné informácie o mikrokontroléri RP2040 a vedieť využiť inštrukčnú sadu ARMv6-M [15].

G.1.1 Ciele

- Oboznámenie sa s programátorským modelom mikrokontroléra RP2040.
- Oboznámenie sa s inštrukčnou sadou ARMv6-M.
- Hľadanie minima a maxima za použitia inštrukčnej sady.
- Návrh funkcií využívajúcich volacie konvencie jazyka C.

G.1.2 Domáca príprava

1. Mikrokontrolér RP2040 obsahuje sadu 32-bitových registrov R0 až R12, ktoré slúžia pre bežné použitie. Ďalej obsahuje registre R13 až R15 a ďalšie pomocné registre. V katalógovom liste mikrokontroléra [8] vyhľadajte ako sú označené a aké využitie majú registre R13 až R15.

R13: .....
R14: .....
R15: .....

2. Inštrukčná sada definuje aké základné operácie vie mikrokontrolér vykonávať. V mikrokontroléri RP2040 je implementovaná inštrukčná sada ARMv6-M [15], ktorú budete v tejto laboratórnej úlohe využívať. Inštrukcie môže obsahovať operandy, s ktorými pracujú. Podľa toho ako inštrukcie pristupujú k operandom, rozlišujeme viaceré adresovacie módy. Tu sú uvedené niektoré z nich:

- priamy operand, napr.: `ADD R0, #24` alebo `LDR R4, =0x5D7E30A7`
- priamy register, napr.: `ADD R0, R4`
- register + offset, napr.: `ADD R2, [R0, #4]` alebo `ADD R2, [R0, R1]`

V katalógovom liste nájdite nasledujúce inštrukcie a ich syntax zapíšte.

Sčítanie (priamy 8-bitový operand): .....
Porovnanie: .....
Breakpoint: .....

3. Inštrukcie sa vykonávajú sekvenčne a ich vykonávanie riadi programový čítač PC. V prípade, že potrebujeme vykonávať inštrukcie inak ako sekvenčne, musíme použiť skoky. Skoky sa môžu vykonávať nepodmienene pomocou inštrukcie `B <label>` alebo podmienene pomocou inštrukcie `B<cc> <label>`, kde `<label>` označuje cieľ skoku a `<cc>` podmienku. Všetky možné prípony inštrukcie `B<cc> <label>` zobrazuje tabuľka na obr. G.1. Splnenie podmienok záleží na príznakových bitoch N, Z, C a V. Tie môžu nastavovať iné inštrukcie, ako napríklad inštrukcia porovnávania.

Suffix	Flags	Meaning
EQ	Z = 1	Equal, last flag setting result was zero.
NE	Z = 0	Not equal, last flag setting result was non-zero.
CS or HS	C = 1	Higher or same, unsigned.
CC or LO	C = 0	Lower, unsigned.
MI	N = 1	Negative.
PL	N = 0	Positive or zero.
VS	V = 1	Overflow.
VC	V = 0	No overflow.
HI	C = 1 and Z = 0	Higher, unsigned.
LS	C = 0 or Z = 1	Lower or same, unsigned.
GE	N = V	Greater than or equal, signed.
LT	N != V	Less than, signed.
GT	Z = 0 and N = V	Greater than, signed.
LE	Z = 1 or N != V	Less than or equal, signed.
AL	Can have any value	Always. This is the default when no suffix is specified.

Obr. G.1: Prípony podmienenej inštrukcie skoku [14]

V katalógovom liste vyhľadajte za akých podmienok sa bity N, Z, C a V v registri APSR nastavia na *log. 1.*

bit N: .....
bit Z: .....
bit C: .....
bit V: .....

4. Zistite a napíšte akú najmenšiu a najväčšiu celočíselnú hodnotu je možné uchovávať v 32-bitovom registri.

Minimálna hodnota: $(\ldots\ldots\ldots\ldots)_{16} = (\ldots\ldots\ldots\ldots\ldots\ldots)_{10}$
Maximálna hodnota: $(\ldots\ldots\ldots\ldots)_{16} = (\ldots\ldots\ldots\ldots\ldots\ldots)_{10}$

G.1.3 Práca na cvičenia

1. V predpripravenom priečinku `exercises` vytvorte priečinok s názvom `01_asm`. Priečinok nezabudnite zahrnúť do zostavovacieho systému programu CMake, a to tak, že na koniec súboru `CMakeLists.txt` v priečinku `exercises` dopíšete riadok s príkazom `add_subdirectory(01_asm)`.

2. Aby bolo možné projekt zostavovať, v priečinku `01_asm` vytvorte súbor s názvom `CMakeLists.txt`, do ktorého vložte príkazy uvedené vo výpise G.1.

Výpis G.1: Súbor `CMakeLists.txt` pre úlohu č. 1

```
add_executable(01_asm 01_asm.S)
target_link_libraries(01_asm pico_stdlib)
pico_add_extra_outputs(01_asm)
```

3. V priečinku `01_asm` vytvorte súbor `01_asm.S` a vložte doň základnú kostru programu uvedenú vo výpise G.2.
4. Na koniec programu pridajte sekciu `.data` a definujte 10-prvkové pole s názvom `array`. Pole bude obsahovať 32-bitové celočíselné hodnoty, a preto pre definovanie hodnôt použite direktívu `.word`. Pole naplňte ľubovoľnými číslami. Napíšte program, ktorý toto pole prehľadá a nájde v ňom minimálnu hodnotu.

5. Do registra R0 načítajte adresu poľa `array`. V registri R3 inicializujte offset na hodnotu 0. Pomocou offsetu budete adresovať jednotlivé prvky poľa. V registri R4 budete uchovávať doposiaľ zistenú minimálnu hodnotu, na začiatok ju však treba inicializovať na najvyššiu možnú hodnotu.

Výpis G.2: Základná kostra programu za použitia inštrukčnej sady

```
.thumb_func
.global main          @ provide program starting address

.align  4             @ necessary alignment

main:
BKPT #0               @ set breakpoint at start

loop:
  B loop              @ loop forever
```

6. V cykle postupne načítavajte prvky poľa do registra R2. Načítaný prvok porovnajte s aktuálnou minimálnou hodnotou uloženou v registri R4. V prípade, že aktuálny prvok má nižšiu hodnotu, aktualizujte hodnotu minima. Ďalej pokračujte zvýšením offsetu o 4 bajty. Ak offset dosiahol maximálnu hodnotu pre 10-prvkové pole, hľadanie minima ukončte a nájdenú minimálnu hodnotu uložte do registra R0. V opačnom prípade pokračujte v prehľadávaní poľa. Funkčnosť algoritmu overte zmenou hodnôt poľa `array`.

7. V prípade, že program funguje správne, časť kódu hľadajúceho minimálnu hodnotu v poli presuňte do funkcie `find_min`. Dodržte správne volacie konvencie jazyka C, tak aby funkciu bolo možné volať z programu napísanom v jazyku C. Do registra R0 teda nech načítava adresu poľa volacia funkcia a výsledok funkcie nech volaná funkcia ukladá do registra R0. Pre návrat do volacej funkcie použite inštrukciu `BX LR`.

8. Podobne postupujte podľa bodov 5 až 7 a naprogramujte funkciu `find_max`, ktorá nájde maximum v poli `array`.

Poznámka: Bližšie informácie o programovaní mikrokontroléra RP2040 je možné nájsť v [2].

G.2 Práca s binárnymi vstupmi a výstupmi

Táto úloha slúži na oboznámenie s perifériou SIO a GPIO. Pre úspešné zvládnutie úlohy je potrebná základná znalosť programovania v jazyku C a schopnosť orientácie sa v katalógových listoch mikrokontroléra RP2040 [8] a prípravku RPi Pico Kit.

G.2.1 Ciele

- Základné zorientovanie sa v elektrickom zapojení prípravku RPi Pico Kit.
- Zorientovanie sa v katalógovom liste mikrokontroléra RP2040.
- Opakovanie práce s bitovými operáciami v jazyku C.
- Konfigurácia binárnych vstupov a výstupov, čítanie binárnych vstupov a ovládanie binárnych výstupov.
- Ovládanie LED diódy tlačidlom enkódera.

G.2.2 Domáca príprava

1. Aby ste ďalej mohli pracovať s tlačidlom a LED diódami, v dokumentácii prípravku RPi Pico Kit vyhľadajte, na ktorý vstup je pripojené tlačidlo enkodéra a na ktoré výstupy sú pripojené LED diódy.

Vstup pre tlačidlo enkodéra: GPIO ...
Výstup pre LED1: GPIO ...
Výstup pre LED2: GPIO ...
Výstup pre LED3: GPIO ...
Výstup pre LED4: GPIO ...

2. Každý pin vývojovej dosky RPi Pico môže mať v závislosti na konfigurácii rôzne funkcie. V dokumentácii mikrokontroléra RP2040 [8] vyhľadajte akými funkciami piny disponujú a doplňte, akú funkciu je potrebné nastaviť v registri `FUNCSEL`, aby pin pracoval vo funkcii GPIO.

Číslo funkcie pre prácu v GPIO režime: ...

3. V programe chceme nastaviť obmedzenie prúdu výstupného pinu na hodnotu 4 mA. V dokumentácii mikrokontroléra RP2040 vyhľadajte, ktoré bity registra `PADS_BANK0_GPIO`n, kde n označuje číslo GPIO, slúžia na nastavenie prúdového obmedzenia. Ďalej zistite, akú hodnotu je potrebné nastaviť pre maximálny výstupný prúd 4 mA. Nezabudnite, že mikrokontrolér má 32-bitovú architektúru a každý register má veľkosť 32 bitov.

Maska pre nastavenie prúdového obmedzenia GPIO: 0x.....
Hodnota bitov pre maximálny prúd 4 mA: 0b.....

4. Pri programovaní mikrokontrolérov sa bežne zapisujú hodnoty do registrov. Ako vyplýva aj z predošlej úlohy, v jednom registri sa zvyčajne nastavujú viaceré funkcie.

Aby sa pri nastavovaní jednej funkcie, neprepisovali bity pre nastavenie inej funkcie, je potrebné využiť maskované zapisovanie. Rovnako pre čítanie je potrebné čítať hodnotu iba určitých bitov. Uvažujme register s názvom `reg`. V nasledujúcej úlohe napíšte, ako by ste v registri `reg` bity *b3* a *b4*:

- nastavili:
  `reg .....`
- resetovali:
  `reg .....`
- zmenili na opačnú hodnotu:
  `reg .....`
- urobili všeobecný zápis (nastavenie aj resetovanie):
  `reg .....`
- prečítali (iba bit *b4*):
  `bool state = .....`

5. Na prípravku RPi Pico Kit sa nachádza 8 tlačidiel. V tejto úlohe však budeme pracovať s tlačidlom enkodéra. Na základe dokumentácie prípravku stručne vysvetlite, prečo tieto tlačidlá nie je možné použiť pre prácu s GPIO.

.....

G.2.3 Práca na cvičenia

1. V predpripravenom priečinku `exercises` vytvorte priečinok s názvom `02_gpio`. Priečinok nezabudnite zahrnúť do zostavovacieho systému programu CMake, a to tak, že na koniec súboru `CMakeLists.txt` v priečinku `exercises` dopíšete riadok s príkazom `add_subdirectory(02_gpio)`.

2. Aby bolo možné projekt zostavovať, v priečinku `02_gpio` vytvorte súbor s názvom `CMakeLists.txt`, do ktorého vložte príkazy uvedené vo výpise G.3.

Výpis G.3: Súbor `CMakeLists.txt` pre úlohu č. 2

```
add_executable(02_gpio 02_gpio.c)
target_link_libraries(02_gpio pico_stdlib)
pico_add_extra_outputs(02_gpio)
```

3. V priečinku `02_gpio` vytvorte súbor `02_gpio.c` a do zdrojového kódu pridajte knižnice uvedené vo výpise G.4.

Výpis G.4: Potrebné knižnice pre úlohu č. 2

```
// libraries for GPIO
#include "hardware/structs/sio.h"
#include "hardware/structs/padsbank0.h"
#include "hardware/structs/iobank0.h"
```

4. Zadefinujte vstupný pin pre tlačidlo enkodéra a výstupný pin pre LED1. Vytvorte funkciu `int main()`, v ktorej vypracujete ďalšie úlohy.

5. Inicializujte GPIO piny pre tlačidlo a diódu LED1 do východzieho stavu nasledujúcim spôsobom:

- Resetovaním príslušného bitu v registri `sio_hw->gpio_oe` nastavte pin ako vstup.
- Resetovaním príslušného bitu v registri `sio_hw->gpio_out` inicializujte výstup na *log. 0*.
- Nastavením bitu `IE` v registri `padsbank0_hw->io[`n`]` povoľte funkciu vstupu.
- Resetovaním bitu `OD` v registri `padsbank0_hw->io[`n`]` povoľte funkciu výstupu.
- Do registra `iobank0_hw->io[`n`].ctrl` zapíšte správnu hodnotu bitov `FUNCSEL`, tak aby pin pracoval vo funkcii GPIO.

6. Inicializáciu GPIO budete potrebovať viackrát použiť aj v ďalších laboratórnych úlohách. Príkazy z predošlého bodu preto premiestnite do funkcie s hlavičkou:

```
void init_gpio(uint32_t gpio, uint8_t function);
```

Pričom parameter `gpio` vyjadruje číslo GPIO pinu a `function` číslo funkcie.

7. GPIO pin, na ktorý je pripojené tlačidlo enkódera nastavte ako vstup a to tak, že v registri `sio_hw->gpio_oe` resetujete príslušný bit.

8. V príslušnom registri pre GPIO pin tlačidla `padsbank0_hw->io[`n`]` pomocou bitu `PUE` odpojte pull-up rezistor a pomocou bitu `PDE` pripojte pull-down rezistor.

9. GPIO pin, na ktorom je pripojená LED1 nastavte ako výstup zapísaním *log. 1* na príslušný bit.

10. V príslušnom registri pre GPIO pin LED1 `padsbank0_hw->io[`n`]` nastavte `DRIVE` bity na takú hodnotu, aby bol prúd obmedzený na hodnotu 4 mA.

11. V nekonečnej slučke čítajte stav tlačidla enkódera z registra `sio_hw->gpio_in` a tento stav zapisujte na LED1 do registra `sio_hw->gpio_out`.

Poznámka: Používanie registrov `sio_hw->gpio_oe` a `sio_hw->gpio_out` do určitej miery spomaľuje program, pretože je potrebné najprv prečítať hodnotu registra, ďalej hodnotu zmeniť a následne do registra zapísať (read – modify – write). Pre rýchlu prácu s GPIO pinmi je možné použiť atomický zápis, ktorý sa vykoná v rámci jedného hodinového cyklu. To funguje tak, že sa zapisuje priamo do registrov slúžiacich na nastavenie, resetovanie alebo zmenu hodnoty daného bitu registra:

- `sio_hw->gpio_oe_set`, `sio_hw->gpio_set`
- `sio_hw->gpio_oe_clr`, `sio_hw->gpio_clr`
- `sio_hw->gpio_oe_togl`, `sio_hw->gpio_togl`

G.3 Obsluha časovača a práca s prerušeniami

Náplňou tohto cvičenia je vytvorenie oneskorovacích funkcií a obsluha prerušení časovača. Pre úspešne zvládnutie úlohy je potrebné vedieť pracovať s časovačom v podobe alarmu a vedieť konfigurovať a spracovávať prerušenia programu [8].

G.3.1 Ciele

- Oboznámenie sa s možnosťami práce s časom v mikrokontoléri RP2040.
- Hľadanie potrebných informácii v príslušných katalógových listoch.
- Programovanie blokujúceho a neblokujúceho oneskorenia programu.
- Základy práce s obsluhou prerušení.
- Blikanie LED diódou.

G.3.2 Domáca príprava

1. Mikrokontrolér RP2040 poskytuje pre prácu s časom signál s periódou 1 µs. Tento časovač inkrementuje 64-bitový čítač. Čítač sa nuluje vždy pri resete mikrokontroléra. Dá sa teda povedať, že hodnota čítača zodpovedá času od posledného spustenia mikrokontroléra v mikrosekundách. Architektúra mikrokontroléra je 32-bitová,

a preto aj všetky registre sú 32-bitové. Hodnota 64-bitového čítača je preto uložená v dvoch registroch. Spodných 32 bitov je uložených v registri `timer_hw->timelr` a horných 32 bitov je uložených v registri `timer_hw->timehr`. Uveďte prevod hodnôt týchto dvoch registrov na jednu 64-bitovú premennú.

```
uint64_t time = ........................................... ;
```

2. V mikrokontroléri je implementovaných niekoľko alarmov, pričom každý z nich vie vygenerovať prerušenie pri zhode určitej hodnoty so spodnými 32 bitmi 64-bitového čítača časovača. V dokumentácii mikrokontroléra nájdite koľko takýchto alarmov obsahuje a na aký najdlhší čas je možné alarm použiť.

Počet alarmov: .....
Maximálny čas pre vypršanie alarmu: .....

Pre prácu s väčšími časmi ako umožňuje alarm je možné využiť periférium hodín reálneho času RTC, ktorá umožňuje generovanie prerušenia presne v daný dátum a čas.

3. Ak je hodnota nejakej premennej upravovaná v obslužnej rutine prerušenia a ďalej má byť táto premenná použitá mimo tejto rutiny, deklarácia takejto premennej musí obsahovať určitý kvalifikátor. Použitie tohto kvalifikátora zabezpečí, že v programe bude hodnota danej premennej vždy načítavaná z pamäte, čím sa zaručí použitie jej správnej hodnoty. Napíšte ako sa tento kvalifikátor v jazyku C volá.

.....

G.3.3 Práca na cvičenia

1. V predpripravenom priečinku `exercises` vytvorte priečinok s názvom `03_timer`. Priečinok nezabudnite zahrnúť do zostavovacieho systému programu CMake, a to tak, že na koniec súboru `CMakeLists.txt` v priečinku `exercises` dopíšete riadok s príkazom `add_subdirectory(03_timer)`.

2. Aby bolo možné projekt zostavovať, v priečinku `03_timer` vytvorte súbor s názvom `CMakeLists.txt`, do ktorého vložte príkazy uvedené vo výpise G.5.

Výpis G.5: Súbor `CMakeLists.txt` pre úlohu č. 3

```
add_executable(03_timer 03_timer.c)
target_link_libraries(03_timer pico_stdlib)
pico_add_extra_outputs(03_timer)
```

3. V priečinku `03_timer` vytvorte súbor `03_timer.c` a do zdrojového kódu pridajte knižnice uvedené vo výpise G.6.

Výpis G.6: Potrebné knižnice pre úlohu č. 3

```
// libraries for GPIO
#include "hardware/structs/sio.h"
#include "hardware/structs/padsbank0.h"
#include "hardware/structs/iobank0.h"

// libraries for timer
#include "hardware/structs/timer.h"

// libraries for interrupts
#include "hardware/regs/intctrl.h"
#include "hardware/regs/m0plus.h"
#include "hardware/structs/scb.h"
```

4. Zadefinujte výstupné piny pre LED1 až LED3. Vytvorte funkciu `int main()`, v ktorej vypracujete ďalšie úlohy.

5. Pomocou funkcie `init_gpio` z predošlých cvičení inicializujte výstupné piny pre LED1 až LED3, tak aby pracovali v režime GPIO. Nezabudnite nastaviť, aby pracovali vo funkcii logických výstupov.

6. Vytvorte funkciu `get_time_us`, ktorá bude navracať aktuálny čas behu programu získaný z registrov `timer_hw->timelr` a `timer_hw->timehr` a jej hlavička bude:

```
uint64_t get_time_us();
```

7. Vytvorte funkciu `delay_us`, ktorá na základe aktuálneho času získaného z funkcie `get_time_us` bude realizovať blokujúce oneskorenie a jej hlavička bude:

```
void delay_us(uint64_t delay_time_us);
```

Pričom premenná `delay_time_us` vyjadruje dĺžku oneskorenia v mikrosekundách. Funkcia si na začiatku zistí aktuálny čas a následne čaká kým uplynie požadovaný čas.

8. V nekonečnej slučke naprogramujte pomocou funkcie `delay_us` blikanie diódy LED1 s periódou 500 ms.

9. Do nekonečnej slučky pridajte neblokujúce oneskorenie, ktoré v každej iterácii zisťuje aktuálny čas a v prípade, že uplynul určitý čas od poslednej akcie vykoná sa ďalšia akcia. Týmto spôsobom naprogramujte blikanie diódy LED2 s periódou 1 s.

10. Vytvorte funkciu `alarm_in_us` pre spúšťanie alarmu a nastavie prerušenia, ktoré sa vygeneruje po vypršaní času `time_us`. Funkcia bude mať hlavičku:

```
void alarm_in_us(uint32_t time_us);
```

Na začiatku funkcie povoľte generovanie prerušení od alarmu č. 0 nastavením bitu `ALARM_0` v registri `timer_hw->inte`. Do vektora prerušení zapíšte hodnotu ukazovateľa obslužnej rutiny `alarm_isr` pre prerušenie typu `TIMER_IRQ_0` pomocou príkazu:

```
((void **)scb_hw->vtor)[16 + TIMER_IRQ_0] = &alarm_isr;
```

Povoľte volanie prerušení na úrovni mikrokontroléra príkazmi:

```
*((io_rw_32 *) (PPB_BASE + MOPLUS_NVIC_ICPR_OFFSET)) =
     (1ul « TIMER_IRQ_0) & MOPLUS_NVIC_ICPR_BITS;

*((io_rw_32 *) (PPB_BASE + MOPLUS_NVIC_ISER_OFFSET)) =
     (1ul « TIMER_IRQ_0) & MOPLUS_NVIC_ISER_BITS;
```

Na základe aktuálneho času v registri `timer_hw->timerawl` a požadovanej vstupnej hodnoty `time_us` vypočítajte čas, kedy má alarm vygenerovať prerušenie. Túto hodnotu vložte do registra príslušného alarmu `timer_hw->alarm[0]`.

Zavolanie funkcie `alarm_in_us` nezabudnite pridať do funkcie `main` pre prvotné spustenie alarmu.

11. Vytvorte obslužnú rutinu prerušenia od alarmu `alarm_isr` s hlavičkou:

```
void alarm_isr();
```

Vo funkcii najprv zrušte príznak prerušenia nastavením bitu `ALARM_0` v registri `timer_hw->intr`. Následne zavolajte funkciu `alarm_in_us` pre spustenie ďalšieho alarmu.

12. Do obslužnej rutiny prerušenia pridajte program, ktorý bude realizovať blikanie diódy LED3 a čas pre vypršanie alarmu nastavte, tak aby táto LED dióda blikala s periódou 100 ms.

G.4 Práca s perifériou - A/D prevodník

Úloha slúži pre oboznámenie sa s perifériou A/D prevodníka. Informácie o tejto periférii sú uvedené v katalógom liste mikrokontroléra RP2040 [8]. Taktiež treba poznať vzťah pre prevod prečítanej hodnoty z prevodníka na veľkosť napätia a vedieť sa orientovať v schéme zapojenia prípravku.

G.4.1 Ciele

- Oboznámenie sa s perifériou A/D prevodníka.
- Konfigurácia A/D prevodníka.
- Čítanie hodnoty napätia na analógovom vstupe.
- Ovládanie stavu LED diód výstupným napätím potenciometra.

G.4.2 Domáca príprava

1. V dokumentácii mikrokontroléra RP2040 vyhľadajte nasledujúce informácie o implementovanom A/D prevodníku.

Typ A/D prevodníka: .....
Rozlíšenie prevodníka: ..... bitov (min. hodnota: ....., max. hodnota: .....)
Efektívny počet bitov (ENOB): ..... bitov
Počet vstupných kanálov: ...

2. Aby ste ďalej mohli pracovať s potenciometrom, v dokumentácii prípravku RPi Pico Kit vyhľadajte, na ktorý pin a ktorý kanál A/D prevodníka je potenciometer

pripojený.

Vstupný pin potenciometra: GPIO ...
Kanál A/D prevodníka: ADC ...

3. Na základe informácie o rozlíšení prevodníka $n[-]$ a hodnote napájacieho napätia $U_{CC} = 3,3\,V$, napíšte vzťah pre výpočet meraného napätia $U_{meas}[V]$ z hodnoty prečítanej z A/D prevodníka $x[-]$.

$$U_{meas} =$$

4. V tejto úlohe budete programovať LED diódy, aby sa postupne rozsvecovali podľa meraného vstupného napätia potenciometra. Do tabuľky G.1 zapíšte jednotlivé rozhodovacie úrovne napätia a stavy LED diód v týchto intervaloch napätia.

Tab. G.1: Logické úrovne LED diód podľa meraného napätia potenciometra

Vstupné napätie	**LED1**	**LED2**	**LED3**	**LED4**
$\langle 0V; \ldots\ldots V\rangle$				
$(\ldots\ldots V; \ldots\ldots V\rangle$				
$(\ldots\ldots V; \ldots\ldots V\rangle$				
$(\ldots\ldots V; \ldots\ldots V\rangle$				
$(\ldots\ldots V; 3,3V\rangle$				

G.4.3 Práca na cvičenia

1. V predpripravenom priečinku `exercises` vytvorte priečinok s názvom `04_adc`. Priečinok nezabudnite zahrnúť do zostavovacieho systému programu CMake, a to tak, že na koniec súboru `CMakeLists.txt` v priečinku `exercises` dopíšete riadok s príkazom `add_subdirectory(04_adc)`.

2. Aby bolo možné projekt zostavovať, v priečinku `04_adc` vytvorte súbor s názvom `CMakeLists.txt`, do ktorého vložte príkazy uvedené vo výpise G.7.

Výpis G.7: Súbor `CMakeLists.txt` pre úlohu č. 4

```
add_executable(04_adc 04_adc.c)
target_link_libraries(04_adc pico_stdlib)
pico_add_extra_outputs(04_adc)
```

3. V priečinku `04_adc` vytvorte súbor `04_adc.c` a do zdrojového kódu pridajte knižnice uvedené vo výpise G.8.

Výpis G.8: Potrebné knižnice pre úlohu č. 4

```
// libraries for GPIO
#include "hardware/structs/sio.h"
#include "hardware/structs/padsbank0.h"
#include "hardware/structs/iobank0.h"

// libraries for ADC
#include "hardware/structs/resets.h"
#include "hardware/structs/adc.h"
```

4. Zadefinujte výstupné piny pre LED1 až LED4, vstupný pin potenciometra a príslušný kanál A/D prevodníka. Vytvorte funkciu `int main()`, v ktorej vypracujete ďalšie úlohy.

5. V registri `resets_hw->reset` nastavte bit `ADC` a hneď potom tento bit resetujte, čím resetujete periférium A/D prevodníka. Počkajte kým sa bit `ADC` v registri `resets_hw->reset_done` nastaví na hodnotu *log. 1*, čo indikuje, že reset periférie sa úspešne dokončil.

6. Zapnite periférium A/D prevodníka nastavením bitu `EN` v registri `adc_hw->cs` a počkajte kým sa úspešne zapne, čo je indikované bitom `READY` v registri `adc_hw->cs`.

7. Využite funkciu `init_gpio` z predošlých cvičení pre inicializáciu vstupného pinu potenciometra, tak aby pracoval v režime `NULL`. Na základe už získaných vedomostí odpojte pull-down a pull-up rezistory z tohto pinu. Zakážte funkciu čítania digitálnej hodnoty zo vstupu resetovaním bitu `IE` v registri `padsbank0_hw->io[`n`]`, kde n značí číslo pinu, na ktorom je potenciometer pripojený. Do registra `adc_hw->cs` zapíšte správnu hodnotu bitov `AINSEL`, ktoré určujú kanál A/D prevodníka, na ktorom sa bude merať hodnota vstupného napätia.

8. Pomocou funkcie `init_gpio` inicializujte výstupné piny pre LED1 až LED4, tak aby pracovali v režime GPIO. Nezabudnite nastaviť, aby pracovali vo funkcii logických výstupov.

9. V nekonečnej slučke čítajte napätie na vstupnom pine potenciometra nasledujúcim spôsobom:

1. Nastavte bit `START_ONCE` v registri `adc_hw->cs` pre začatie prevodu napätia z vybratého vstupného kanálu na digitálnu hodnotu.
2. Prevod trvá určitú dobu a preto počkajte kým sa prevod dokončí, čo je indikované bitom `READY` v registri `adc_hw->cs`.
3. Výsledná hodnota A/D prevodu sa nachádza v registri `adc_hw->result`.

10. Naprogramujte LED diódy tak, aby sa postupne rozsvecovali na základe meraného napätia na vstupnom pine potenciometra podľa rozhodovacích úrovní vypočítaných v domácej príprave v tabuľke G.1.

G.5 Komunikácia mikrokontroléra s okolím pomocou UART

Táto úloha slúži na oboznámenie sa s UART rozhraním, preto je potrebné poznať zloženie UART rámca a spôsob práce s týmto rozhraním pri použití mikrokontroléra RP2040 [8]. Potrebné je ovládať aj základy práce s ASCII tabuľkou.

G.5.1 Ciele

- Oboznámenie sa s UART komunikáciou.
- Parametre UART portu a ich konfigurácia.
- Posielanie a prijímanie správ cez UART rozhranie.
- Práca s ASCII tabuľkou.

G.5.2 Domáca príprava

1. Každý znak, ktorý chceme poslať prostredníctvom UART portu je zaobalený v určitom rámci. Nakreslite rámec UART komunikácie s vysvetlením jednotlivých bitov.

2. Rámec UART komunikácie je možné nastaviť podľa potreby. Mikrokontrolér RP2040 obsahuje dve vstavané periférie UART. Prípravok RPi Pico Kit konkrétne využíva periférium UART0. V katalógovom liste mikrokontroléra vyhľadajte, ktoré bity v registri `uart0_hw->lcr_h` slúžia pre:

- nastavenie počtu dátových bitov: .....
- nastavenie počtu stop bitov: .....
- povolenie funkcie parity: .....
- výber medzi párnou a nepárnou paritou: .....

3. Komunikácia UART si vyžaduje použitie dvoch dátových pinov. Pre prijímanie správ sa používa pin RX a na odosielanie pin TX. V dokumentácii prípravku vyhľadajte, ktoré piny sú použité pre komunikáciu cez UART rozhranie.

RX pin: GPIO ...
TX pin: GPIO ...

4. Najjednoduchší spôsob ako v počítači reprezentovať číslice a znaky latinskej abecedy je pomocou 7-bitovej hodnoty, ktorú dostaneme zakódovaním príslušného znaku pomocou ASCII tabuľky. Nájdite a zapíšte, akou hodnotou sú reprezentované nasledujúce znaky v ASCII tabuľke.

`'A'`: $(\ldots\ldots)_{16} = (\ldots\ldots)_{10}$
`'t'`: $(\ldots\ldots)_{16} = (\ldots\ldots)_{10}$
`'4'`: $(\ldots\ldots)_{16} = (\ldots\ldots)_{10}$
`':'`: $(\ldots\ldots)_{16} = (\ldots\ldots)_{10}$

G.5.3 Práca na cvičenia

1. V predpripravenom priečinku `exercises` vytvorte priečinok s názvom `05_uart`. Priečinok nezabudnite zahrnúť do zostavovacieho systému programu CMake, a to tak, že na koniec súboru `CMakeLists.txt` v priečinku `exercises` dopíšete riadok s príkazom `add_subdirectory(05_uart)`.

2. Aby bolo možné projekt zostavovať, v priečinku `05_uart` vytvorte súbor s názvom `CMakeLists.txt`, do ktorého vložte príkazy uvedené vo výpise G.9.
3. V priečinku `05_uart` vytvorte súbor `05_uart.c` a do zdrojového kódu pridajte knižnice uvedené vo výpise G.10.

Výpis G.9: Súbor `CMakeLists.txt` pre úlohu č. 5

```
add_executable(05_uart 05_uart.c)
target_link_libraries(05_uart pico_stdlib)
pico_add_extra_outputs(05_uart)
```

Výpis G.10: Potrebné knižnice pre úlohu č. 5

```
// standard libraries
#include <string.h>
#include "pico/stdlib.h"

// libraries for GPIO
#include "hardware/structs/sio.h"
#include "hardware/structs/padsbank0.h"
#include "hardware/structs/iobank0.h"

// libraries for UART
#include "hardware/resets.h"
#include "hardware/clocks.h"
#include "hardware/structs/uart.h"
#include "hardware/structs/resets.h"
```

4. Vytvorte dátový typ `enum parity_t`, ktorý môže nadobúdať tieto hodnoty:

- `PARITY_NONE`,
- `PARITY_EVEN`,
- `PARITY_ODD`.

5. Zadefinujte RX a TX piny a parametre UART komunikácie s týmito hodnotami:

- prenosová rýchlosť: 9600 Bd,
- počet dátových bitov: 8,
- počet stop bitov: 1,
- parita: žiadna (dátového typu `enum parity_t`).

6. Deklarujte a inicializujte statickú premennú `uint received_chars`, ktorá bude slúžiť ako počítadlo prijatých znakov. Vytvorte funkciu `int main()`, v ktorej vypracujete ďalšie úlohy.

7. V registri `resets_hw->reset` nastavte bit `UART0` a hneď potom tento bit resetujte, čím resetujete periférium UART komunikácie. Počkajte kým sa bit `UART0` v registri

`resets_hw->reset_done` nastaví na hodnotu *log. 1*, čo indikuje, že reset periférie sa úspešne dokončil.

8. Pre nastavenie požadovanej prenosovej rýchlosti je potrebné vložiť správne hodnoty do registrov `uart0_hw->ibrd` a `uart0_hw->fbrd`. Pre výpočet ich hodnôt použite funkciu `uart_baudrate` uvedenú vo výpise G.11, do ktorej vložte požadovanú hodnotu prenosovej rýchlosti. Funkcia na základe vypočítaných hodnôt registrov navracia skutočnú hodnotu prenosovej rýchlosti, ktorá bude použitá.

Výpis G.11: Výpočet konfiguračných parametrov prenosovej rýchlosti UART portu [36]

```
uint uart_baudrate(uint baudrate, uint32_t *ibrd, uint32_t *fbrd)
{
    uint32_t rate_div = (8 * clock_get_hz(clk_peri) / baudrate);
    *ibrd = rate_div >> 7;

    if (*ibrd == 0)
    {
        *ibrd = 1;
        *fbrd = 0;
    }
    else if (*ibrd >= 65535)
    {
        *ibrd = 65535;
        *fbrd = 0;
    }
    else
    {
        *fbrd = ((rate_div & 0x7f) + 1) / 2;
    }

    return (4 * clock_get_hz(clk_peri)) / (64 * (*ibrd) + (*fbrd));
}
```

9. Inicializujte register `uart0_hw->lcr_h` na hodnotu 0 a vykonajte nastavenia UART rámca podľa požadovaných hodnôt.

10. Periférium UART0 zapnite nastavením bitu `UARTEN` v registri `uart0_hw->cr`. V rovnakom registri nastavte aj bity `TXE` a `RXE` pre povolenie prijímania a odosielania správ. V tomto registri ešte resetujte bity `RTSEN` a `CTSEN` pre vypnutie harvérového riadenia toku dát.

11. Využite funkciu `init_gpio` z predošlých cvičení pre inicializáciu pinov RX a TX, tak aby pracovali v režime `UART`.

12. Naprogramujte odosielanie jedného znaku a to tak, že program najprv čaká, kým sa uvoľní výstupný buffer, čo indikuje bit `TXFF` v registri `uart0_hw->fr`. Keď sa buffer uvoľní, do registra `uart0_hw->dr` vložte znak, ktorý chcete odoslať. Ďalej tento program upravte tak, aby bol schopný odosielať ľubovoľné množstvo znakov. Na UART port pošlite správu `"Hello World"`. Prijatie správy overte na počítači monitorovaním príslušného portu.

13. Program na odosielanie znakov z predošlého bodu presuňte do funkcie s hlavičkou:

```
void uart_write(const char *src, size_t len);
```

Pričom parameter `const char *src` je pole znakov, ktoré sa má odoslať a `size_t len` je počet znakov na odoslanie.

14. V nekonečnej slučke kontrolujte, či do vstupného buffera UART portu bol prijatý nejaký znak, čo indikuje bit `RXFE` v registri `uart0_hw->fr`. V prípade, že nejaký znak bol prijatý, prečítajte jeho hodnotu z registra `uart0_hw->dr` a uložte ju do dočasnej premennej. Prijatý znak pošlite naspäť na UART port pomocou funkcie `uart_write`. Implementujte počítanie prijatých znakov v premennej `received_chars`.

15. Program rozšírte o možnosť prijímania ľubovoľného množstva znakov a tento kód vložte do funkcie s hlavičkou:

```
void uart_read(char *dst, size_t len);
```

Pričom parameter `char *dst` je pole znakov, do ktorého sa majú prijaté znaky uložiť a `size_t len` je počet znakov na prečítanie.

16. Upravte program tak, aby každý prijatý znak bol analyzovaný a v prípade, že ide o malé písmeno, bol zmenený na veľké písmeno a naopak v prípade, že ide o veľké písmeno, bol zmenený na malé písmeno. Ostatné znaky zostávajú nezmenené. Takto upravené prijaté znaky pošlite naspäť.

17. Program z predošlého bodu pre zmenu veľkých písmen na malé a malých písmen na veľké presuňte do funkcie s hlavičkou:

```
char change_letter_case(char x);
```

Pričom parameter `char x` je vstupný znak a funkcia navracia zmenený znak.

H Vzorové riešenie laboratórnych úloh

Vzorové riešenie laboratórnych úloh nie je verejné, avšak vedúcemu práce boli všetky súbory poskytnuté.

Popis súborov v priečinku `pico-kit-exercises`:

```
pico-kit-exercises/
├── .vscode/
│   ├── launch.json ..................... nastavenia debuggera
│   └── settings.json ................... nastavenia programu Visual Studio Code
├── exercises/ ..........................priečinok pre vypracovanie úloh
│   ├── 01_asm/ ......................... vzorové riešenie úlohy č. 1
│   │   ├── 01_asm.S
│   │   └── CMakeLists.txt
│   ├── 02_gpio/ ........................ vzorové riešenie úlohy č. 2
│   │   ├── 02_gpio.c
│   │   └── CMakeLists.txt
│   ├── 03_timer/ ....................... vzorové riešenie úlohy č. 3
│   │   ├── 03_timer.c
│   │   └── CMakeLists.txt
│   ├── 04_adc/ ......................... vzorové riešenie úlohy č. 4
│   │   ├── 04_adc.c
│   │   └── CMakeLists.txt
│   └── 05_uart/ ........................ vzorové riešenie úlohy č. 5
│       ├── 05_uart.c
│       └── CMakeLists.txt
├── CMakeLists.txt .................. hlavný konfiguračný súbor programu CMake
└── pico_sdk_import.cmake ............pomocný skript pre nájdenie C/C++ SDK
```