VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2024 Roman Křivánek

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV RADIOELEKTRONIKY

DEPARTMENT OF RADIO ELECTRONICS

HARDWAROVÝ MODUL PRO INTERAKTIVNÍ OVLÁDÁNÍ PROJEKTŮ V PUREDATA

HARDWARE MODULE FOR INTERACTIVE CONTROL OF PROJECTS IN PUREDATA

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Roman Křivánek

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Jiří Schimmel, Ph.D.

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Elektronika a komunikační technologie**

Ústav radioelektroniky

Student: Roman Křivánek ***ID:*** 240642

Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Hardwarový modul pro interaktivní ovládání projektů v PureData

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte koncept připojení periferií pro Arduino UNO R3 formou základové desky a jejich rozložení s ohledem na jejich reálné rozměry a ergonomii ovládání. Deska bude obsahovat tahový potenciometr, rotační enkoder, přepínače, tlačítka nebo tlačítkovou klávesnici, LED diody, mikrofon a piezoelektrický reproduktor. Všechny periferie doplňte nezbytnými obvody, aby bylo možné je přímo propojit se vstupními a výstupními piny procesoru. K desce bude také připojen modul gyroskopu s akcelerometrem, ultrazvukovým měřičem vzdálenosti a modul pro bezdrátové propojení. Tyto moduly lze použít již hotové, dostupné na trhu. Jednotlivé periferie samostatně odzkoušejte.

Navrženou základovou desku realizujte, osaďte ji a všechny periferie odzkoušejte. Vytvořte kód pro Arduino, který bude pracovat se všemi periférierni současně a komunikovat s prostředím Pure Data. Dále vytvořte ukázkový projekt pro Pure Data, který jednoduchým způsobem demonstruje využití všech periferií. Vytvořte dokumentaci k využití periférií v prostředí Pure Data.

DOPORUČENÁ LITERATURA:

[1] SELECKÝ, Matúš. Arduino: uživatelská příručka. Brno: Computer Press, 2016. ISBN 978-802-5148-402.

[2] KAVAN, Jan. Pure Data: platforma pro tvorbu interaktivního díla. Brno: Janáčkova akademie múzických umění v Brně, 2013. ISBN 8074600335

Termín zadání: 16.2.2024 ***Termín odevzdání:*** 27.5.2024

Vedoucí práce: doc. Ing. Jiří Schimmel, Ph.D.

doc. Ing. Lucie Hudcová, Ph.D.
předseda rady studijního programu

Abstrakt

Práce se zabývá základními teoretickými informacemi o desce Arduino Uno R3. Poté se zabývá teoretickými informacemi o použitých součástkách. Dále se v práci nachází kapitola, kde se jednotlivé součástky testují s deskou Arduino Uno R3. Při testování součástek jsou přiložena schémata zapojení a ukázkové kódy k otestování funkčnosti součástek a modulů. Na konci práce je kapitola o návrhu deky plošného spoje, která je výstupem této práce.

Klíčová slova

Arduino Uno, Pure Data, Deska plošných spojů, Testování elektronických součástek, Bezdrátový přenos dat.

Abstract

In the work, there are basic theoretical information about the Arduino Uno R3 board. Then there are theoretical information about the components used. Furthermore, there is a chapter in the work where individual components are tested with the Arduino Uno R3 board. During the testing of the components, circuit diagrams and sample codes are provided to test the functionality of the components and modules. At the end of the work, there is a chapter on the design of the printed circuit board, which is the output of this work

Keywords

Arduino, Pure Data, Printed Circuit Board, Electronic component testing, Wireless data transmission.

Bibliografická citace

KŘIVÁNEK, Roman. *Hardwarový modul pro interaktivní ovládání projektů v PureData* [online]. Brno, 2024 [cit. 2023-10-17]. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/153421. Semestrální práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky. Vedoucí práce Jiří Schimmel

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Roman Křivánek*
VUT ID studenta:	*240642*
Typ práce:	*Bakalářská práce*
Akademický rok:	*2023/24*
Téma závěrečné práce:	*Hardwarový modul pro interaktivní ovládání projektů v PureData*

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.

V Brně dne: 2. ledna 2024 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

Úvod

Tato práce se zabývá návrhem a realizací interaktivního modulu na vývojovou desku Arduino Uno R3. Tento modul je schopný ovládat projekty v programovacím prostředí PureData, které studenti mohou upravovat.

Dokument začíná teoretickým úvodem použitých platforem Arduino a PureData a použitých prvků a modulů. Dále navazuje způsob otestování funkčnosti jednotlivých komponent na platformě Arduino. Poté následuje návrh desky plošných spojů (DPS) a přehled verzí s uvedením změn v návrzích. Poté následuje přehled verzí kódu Arduino (c++) a popis programu Pure Data.

Přehled jednotlivých verzí DPS, Arduino kódů a celý Pure Data program je umístěn na konci souboru v příloze.

1. CÍL PRÁCE

Hlavním cílem práce je vytvořit interaktivní modul na platformu Arduino uno R3, který dále bude sloužit k výuce. Dílčími cíli jsou otestovat jednotlivé komponenty, které budou dále používány. Dalším dílčím cílem je vytvořit desku plošného spoje, která bude obsahovat požadované komponenty a zároveň bude co nejmenší. K desce bude vytvořen Arduino kód, který pracuje se všemi komponentami a s programem Pure Data, kde bude šablona programu pro práci s daty z jednotlivých komponent. V rámci bezpečnosti a prodloužení životnosti desky bude navrhnuta krabička k zamezení zničení součástek pomocí elektrostatických výbojů.

2. TEORETICKÝ ÚVOD

2.1 Arduino

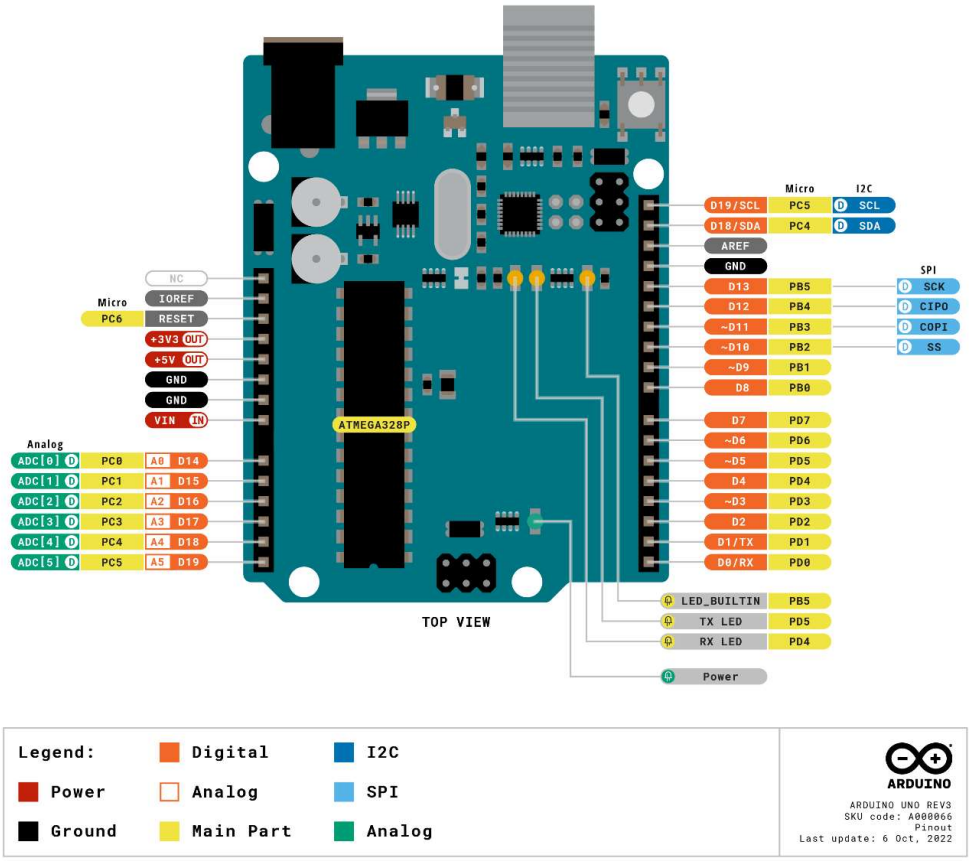
Arduino je open-source elektronická platforma založená na snadném používání hardwaru a softwaru. Desky řady Arduino dokáží číst data z jednotlivých pinů, jako například stav připojeného tlačítka nebo otočení potenciometrem a reagovat na ně změnou výstupních pinů, například rozsvítit LED nebo spustit motor a mnohem dalších úkonů. K určení funkce desky Arduino je zapotřebí ji nejprve naprogramovat pomocí jazyka C/C++, jež se překládá pomocí překladače na strojový kód, který se posílá na 8bitový mikroprocesor ATmega umístěn na deskách Arduino.

2.1.1 Historie

Arduino vzniklo v Ivrea Interaction Design Institute v Itálii v roce 2005 jako nástroj určený k rychlému prototypování a k pomoci studentům elektroniky a programování. V průběhu let, jak se postupně šířilo do komunity, se i desky měnily, aby se podřídily čím dál větším nárokům a výzvám. V roce 2010 vyšla verze Arduina s označením Uno, která nahradila svého předchůdce Duemilanove. Dále od roku 2012 jsou v prodeji desky s označením DUE, které pracují s procesorem ARM a Leonardo. V roce 2012 také vyšla nová revize desky Arduino Uno R3. V prodeji je také verze Arduino nano, Micro, které jsou velikostí přizpůsobeny menším projektům, a mnoho dalších typů. Poslední hlavní deskou je nové Arduino Uno R4 Minima a Uno R4 WIFI, které přináší konektor USB-C a nový 32bitový mikrokontroler řady RA4M1 od ARM Cortex-M4, který toto Arduino dělá 4x až 16x rychlejší než jeho předchůdce Arduino Uno R3.

2.1.2 Struktura Arduino Uno R3

Arduino Uno R3 použité v této práci je postaveno okolo 8bitovového mikroprocesoru řady ATmega 328P, který používá harvardskou architekturu a je od firmy Atmel. Dále obsahuje 32 kB flash paměti, 2 kB SRAM paměti a 1 kB EEPROM paměti, které jsou použity pro ukládání programu do flash paměti a dat do SRAM a EEPROM paměti. Dále deska obsahuje dva 8bitové čítače/časovače a jeden 16bitový čítač/časovač, které se používají pro výrobu časovacích signálů k řízení mikroprocesoru a přerušení.



Obrázek 2.1 Pinout Arduina Una R3 (převzato z [4])

Deska je napájena z 9V baterie nebo zdroje stejnosměrného napětí pomocí osazeného 2,1 mm jack konektoru, pomocí stejnosměrného napětí 6 až 20 V na pinu Vin nebo pomocí osazeného konektoru USB typu B, který zároveň slouží i pro přenos dat z a do počítače. Při napájení desky z DC jack konektoru je pin Vin použit i jako výstupní pin napájecího napětí.

Deska Arduino Uno R3 obsahuje vývody k připojení čtrnácti digitálních vstupů/výstupů, tzv. GPIO pinů, pět analogových vstupů a jeden analogový komparátor. Pro komunikaci se využívá synchronní/asynchronní přijímač/vysílač, jedno sériové rozhraní SPI a jedno I^2C rozhraní.

Digitální GPIO piny připojeny k D0-D13 se konfigurují pomocí příkazu pinMode(), zda jsou vstupní či výstupní, dále je zde možnost nastavit, zda má být interní pull-up rezistor připojen či odpojen. Typické hodnoty vnitřních pull-up rezistorů se pohybují mezi 20 kΩ až 50 kΩ. Pull-up rezistory mají funkci ke spřažení neaktivního signálu k vysoké hodnotě napětí a tím se docílí zamezení čtení neurčeného stavu na pinu. Tudíž, pokud je zmáčknuto tlačítko, tak na vstupu digitálního pinu je logická nízká úroveň. Dále se dají zapojit i pull-down rezistory, které toto chování obrátí. Tyto rezistory jsou připojeny k zemi a vztahují k ní i neaktivní linku digitálního pinu. Tyto piny slouží

k obsluze digitálních periférií, jako například rozsvícení LED nebo sepnutí relé pomocí příkazu digitalWrite() nebo čtení stavu tlačítka pomocí příkazu digitalRead(). Každý z těchto pinů má výstupní proud 20 mA**,** maximálně 40 mA. Piny D3, D5, D6, D9, D10 jsou navíc schopné generovat 8bitový PWM signál, který jde definovat pomocí příkazu analogWrite(). Analogové vstupy A0-A5 používají 10bitový AD převodník, který měří napětí mezi pinem a referencí pomocí příkazu analogRead().

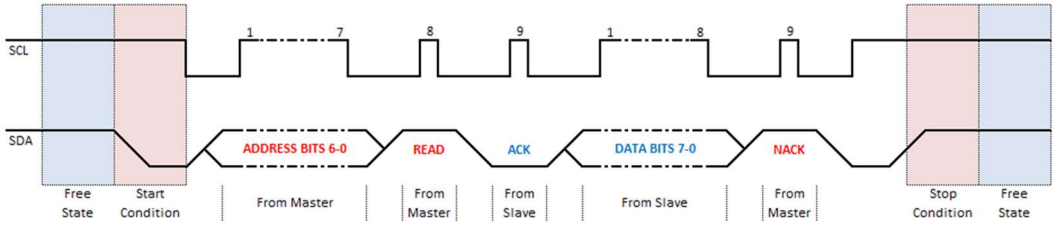
Kvůli vnitřnímu zapojení nemůžeme využívat všechny rozhraní zároveň, jelikož například SPI je připojeno k GPIO pinům D11, D12 a D13, nebo I^2C je připojeno k A4 a A5.

Sériové rozhraní komunikuje pomocí synchroního/asynchronního přijímače/vysílače a je vyvedeno na piny D0(Rx) a D1(Tx). Pomocí sériového rozhraní Arduino Uno komunikuje s počítačem a je díky němu i programováno, data z USB rozhraní jsou překládána pomocí integrovaného převodníku na sériová data. Dále pomocí tohoto rozhraní Arduino komunikuje s moduly využívajícími UART komunikaci, jako je například WIFI modul ESP-01 anebo s dalšími Arduiny pomocí dvou vodičů, což velmi ulehčuje propojení.

I^2C, neboli Inter-Integrated Circuit je sériová sběrnice od firmy Philips.

TWI – Two Wire Interface je dvou drátové rozhraní prakticky totožné rozhraní jako I^2C, ale nepodléhá ochranné značce.

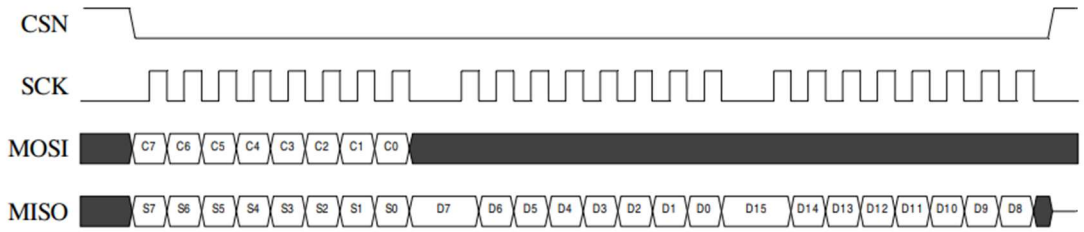
Tato rozhraní jsou používána pro pomalejší přenos dat po dvou vodičích Scl a Sda, pro komunikaci mezi Master a Slave zařízeními, jako jsou například IC gyroskopy, IC rozšiřující piny GPIO a mnoho dalších modulů a čipů. Rychlost těchto rozhraní se udává od 10 kb/s do 3,4 Mb/s, běžně se také používá přenosová rychlost 400 kb/s.



Obrázek 2.2 Časování TWI komunikace po linkách SDA a SCL (převzato z [5])

SPI, neboli sériové periferní rozhraní, je určeno pro komunikaci mezi Master a Slave zařízeními s velmi rychlým přenosem dat. Jedná se o synchronní přenos dat a tudíž je potřeba přenos časového signálu ze zařízení Master do zařízení Slave. K přenosu dat po sběrnici SPI je zapotřebí připojení čtyř vodičů, kde jeden je SCL časování, další je MOSI - Master Out Slave In, který je určen pro přenos dat z Master do Slave, další je MISO - Master In Slave Out pro přenos dat ze Slave zařízení do Master zařízení a poslední je Chip Select (CS) nebo Slave Select (SS) pin, který udává, které Slave zařízení je zrovna vybráno. U některých nových zařízení se označení může lišit, ale funkce zůstává

stejná. Rychlost toho rozhraní se udává až 8 Mb/s. Tato sběrnice se používá pro připojení EEPROM pamětí, displejů, A/D a D/A převodníků, rozšiřovacích IC GPIO pinů či bezdrátových WIFI modulů, jako například EPS01 nebo nRF24L01.



Obrázek 2.3 Ukázka SPI časování z technické dokumentace modulu nRF24L01(převzato z [6]**Chyba! Nenalezen zdroj odkazů.**)

2.2 Pure Data

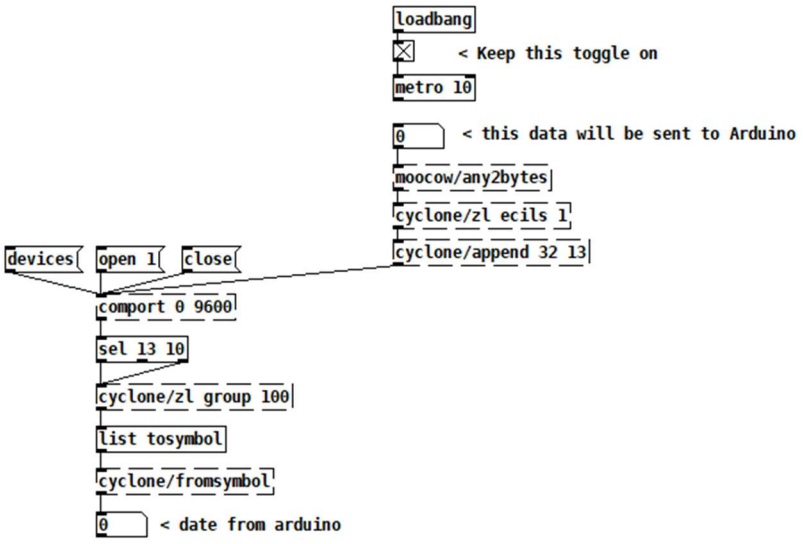
Pure Data je open-source vizuální programovací jazyk určený pro multimédia. Používá se pro pracování s daty v reálném čase. Program má využití pro zpracování a generování zvuku, dále jde použít i pro analýzu videa či obrazu. Program zvládá i komunikaci po internetu či práci s fyzickým světem pomocí vývojových desek, např. Arduino. Program je oblíben z důvodu jeho grafické přehlednosti a jednoduchosti porozumění.

2.2.1 Historie

Program Pure Data byl vytvořen panem Millerem Puckettem v roce 1996, který pracuje na Kalifonské Univerzitě v San Diegu (UCSD) na katedře hudby. Dále je program vyvíjen komunitou. Pan Miller Puckette také vytvořil komerční verzi tohoto programu pod názvem Max/MSP, ale ten není tak populární a většina aktualizací této komerční varianty je zahrnuta i do bezplatné verze Pure Data.

2.2.2 Používání

Program Pure Data je vizuální programovací jazyk, což znamená, že se programuje pomocí vizuálních obdélníků umístěných do programovacích ploch. Tyto obdélníky jsou označovány „patch“, v množném čísle to jsou „patche“. Do těchto obdélníků uživatel zadává psaný kód a obdélníky se vizuálně mění podle daného kódu. Dále je možné vizuálně propojit výstup obdélníku se vstupem jiného obdélníku.



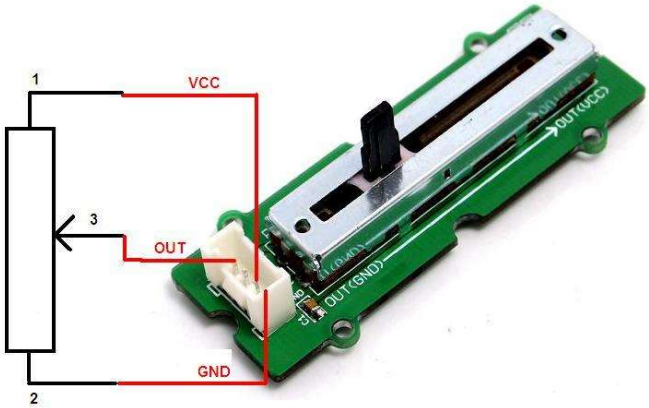
Obrázek 2.4 Ukázka programu v prostředí Pure Data

2.3 Jednotlivé součástky a moduly

Tato kapitola se zabývá fyzikálními principy a fungováním jednotlivých součástek a na fungování modulů.

2.3.1 Tahový potenciometr

Tahový potenciometr je pasivní součástkou, která je založena na funkci rezistoru popsané Ohmovým zákonem a je závislá na poloze táhla této součástky. Nejjednodušší konstrukce potenciometrů je z odporové dráhy, po které se pohybuje jezdec. Odporová dráha je často realizována uhlíkovou vrstvou, kovovou vrstvou ale i odporovým drátem či vodivým plastem. Jezdec bývá výliskem z elektrografitu, vodivého plastu či jedním nebo více kovovými perky.



Obrázek 2.5 Tahový potenciometr a jeho schématická značka (převzato z [8])

Tato součástka má tři svorky, VCC, GND a měřicí svorku. VCC je připojena ke zdroji napětí a GND je připojeno k zemi. Pomocí multimetru lze na potenciometru měřit napětí a proud procházející potenciometrem. Z těchto veličin lze vypočítat celkový odpor potenciometru anebo odpor určený jezdcem. Pokud se voltmetrem měří napětí nebo odpor mezi VCC nebo GND a měřicí svorkou, chová se tahový potenciometr jako nezatížený dělič napětí a odpor potenciometru se rozdělí na odpor mezi VCC a měřicí svorkou a odpor mezi měřicí svorkou a GND.

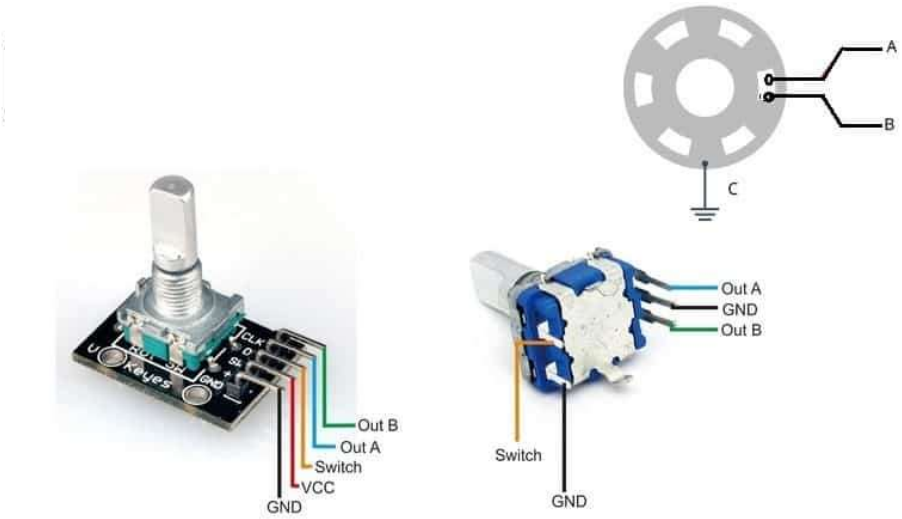
Pro zlepšení měření jsou některé tahové potenciometry vybaveny dvěma měřicími svorkami, tudíž je hodnota měřena vícenásobně a lze ji průměrovat. Často se také do obvodu s tahovým potenciometrem připojuje sériově k VCC nebo GND vývodu sériový rezistor, který má za funkci omezit proud procházející potenciometrem.

2.3.2 Rotační Enkodér

Rotační enkodér je elektro-mechanickou součástkou, která převádí otočný pohyb na analogový či digitální výstupní signál. Rozeznávají se dva typy, absolutní a inkrementální. U absolutního enkodéru je výstupem aktuální pozice jezdce v daném rozsahu otáčení. Výstupem inkrementálního enkodéru je informace o pohybu otočné hřídele, které se poté vyhodnocuje na pozici, rychlost a vzdálenost.

Absolutní rotační enkodér se skládá z několika zdrojů světla a stejného počtu detektorů světla, které se podílejí na určení polohy hřídele pomocí kódových kruhů, kde je laserem vyřezaný BCD kód. Tento enkodér může obsahovat i více kódových kruhů k zvýšení přesnosti a kroku. Tato konstrukce zajišťuje určení správné polohy i po odpojení napájecího napětí.

Inkrementální rotační enkodér obsahuje tři řady na kódovém kruhu. Jeden je určen jako referenční, kde je pouze jedna díra. Další dva kruhy jsou vyřezány s vícero dírami, které určují posun hřídele obdobně jako u absolutního. Díry u druhého a třetího kruhu jsou navzájem posunuty o určitý úhel, tak aby výstupní signály byly posunuty o 90°. Směr otáčení se udává až ve vyhodnocovací jednotce, která porovnává signály A a B a podle toho, zda je náběžná hrana signálu A při hodnotě low nebo high signálu B, se určí, zda se inkrementuje či dekrementuje. Nevýhodou tohoto provedení je nutnost synchronizace po každém odpojení napájení pomocí značky na referenčním kroužku.



Obrázek 2.6 Struktura rotačního enkodéru (převzato z [10])

2.3.3 Dvoupólový přepínač

Jedná se o pasivní součástku, která má tři svorky. Tato součástka funguje tak, že přepínač určuje, který obvod je uzavřen, buď obvod A-GND nebo B-GND. Dvoupólový přepínač je složen z nevodivého těla, vodivých svorek určených k zapájení a nevodivého přepínače s vodivou ploškou, která jezdí mezi A a B a je vždy připojena k GND svorce.

Poté záleží na použití, buď je součástka určena pro zapínání a vypínání obvodu, např s napájecím napětím, nebo pro určení stavu A nebo B, který je dále určen podle kódu mikropočítače.

2.3.4 Tlačítko

Tlačítko je pasivní součástkou, která funguje jako propojovací člen obvodu. Existuje mnoho variant této součástky. Dá se dělit podle velikosti nebo funkčnosti. Podle funkčnosti se dělí podle typu spouštění, a to na držení a stisknutí. Funkčnost je určena mechanickým provedením. Často se u provedení setkáte s čtyřmi vývody, poté je důležité zkontrolovat které dva jsou permanentně propojené a které dva se spojují pomocí zmáčknutí.

2.3.5 LED – Light emiting diode

Jedná se o aktivní součástku, která je určena pro vizuální signalizaci. Tato součástka musí být vždy doplněna o rezistor omezující proud procházející přes LED, proud diodou se normovaně udává 20 mA, který by se neměl překročit. U připojení LED je důležité dodržet polaritu zapojení, katoda musí být připojena k zemi. Velikost napětí je závislá na barvě LED, typické hodnoty jsou uvedeny v tabulce 2.1.

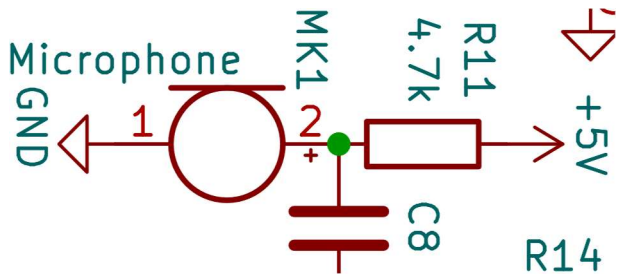
Tabulka 2.1 Tabulka hodnot napětí na LED a doporučený odpor

Barva LED	Typické rozpětí napětí na LED při proudu 20 mA	Doporučená hodnota rezistoru při VCC=5V z řady E24
Bílá	3,0 až 5,0 V	100 Ω
Ultrafialová	3,1 až 4,4 V	100 Ω
Fialová	2,8 až 4,0 V	100 Ω
Modrá	2,5 až 3,7 V	120 Ω
Zelená	1,6 až 4,0 V	180 Ω
Žlutá	2,0 až 2,4 V	150 Ω
Oranžová	2,0 až 2,1 V	150 Ω
Červená	1,5 až 2,0 V	180 Ω
Infračervená	1,2 až 1,9 V	200 Ω

Provedení LED je buď THT, což je větší provedení určené pro montáž skrz DPS, anebo SMD, což je menší provedení určené pro montáž na povrch DPS. Často jsou LED normovány s označením 0603, 0806, 1208 v palcích, obdobně jako rezistory, ale také se dělají v různých rozměrech pouzder, na které si musíte při návrhu DPS dát pozor. Výhodou SMD je menší plocha na DPS.

2.3.6 Piezoelektrický mikrofon

Piezoelektrický mikrofon je aktivní součástkou, která funguje na principu generování napětí pomocí piezoelektrického jevu. V závislosti na výchylce membrány se generuje střídavé napětí charakterizující zvukový signál. Toto napětí se dá vzorkovat a převádět na digitální hodnoty, aby se signál mohl dále zpracovávat pomocí procesorů. Tato součástka má dvě svorky, jedna se připojuje k měřicí svorce a druhá je připojena na kladný potenciál napětí přes omezující rezistor a blokovací kondenzátor.



Obrázek 2.7 Základní zapojení kondenzátorového mikrofonu (převzato z [14])

2.3.7 Piezoelektrický reproduktor

Jedná se o aktivní součástku určenou ke zvukové signalizaci. Funguje na principu piezoelektrického jevu, kdy se membrána reproduktoru vychyluje v závislosti na vstupním střídavém signálu. Často se používá PWM signál. Součástka má dvě svorky, jedna je připojena k PWM výstupním pinu a druhá je připojena k zemi. Při zapojení se musí dbát na určeném napěťovém a proudovém omezení, které je určeno v dokumentaci součástky.

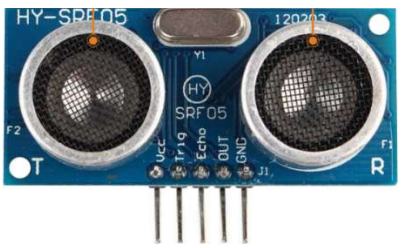
2.3.8 Gyroskop

Gyroskop je aktivní součástkou, která měří polohu a zrychlení v závislosti na změně magnetického pole. Čip gyroskopu se provádí v různém provedení v závislosti na počtu snímaných os, provádí se v provedení 1, 2 nebo 3 osy, převážně poloha nebo zrychlení v ose X, Y a Z, nebo společně více uvedených os, poté 6 os – poloha a zrychlení v osách X, Y a Z nebo 9 os – poloha, zrychlení a magnetické pole země, tj. kompas.

Obrázek 2.8 Obrázek modulu gyroskopu MPU-9265 (převzato z [16])

2.3.9 Ultrazvukový měřič vzdálenosti

Jedná se o aktivní součástku zařazenou do skupiny snímačů vzdálenosti, která používá piezoelektrický reproduktor a mikrofon. Vysílá a přijímá signál o vysoké frekvenci, který se označuje jako Ultrazvukový, jelikož nelze slyšet lidským uchem. Poté vyhodnocovací jednotka počítá čas mezi vysláním vlny a příjmem vlny, který se přepočítá na vzdálenost podle známé rychlosti šíření zvuku ve vzduchu. Tato rychlost je závislá na teplotě a tlaku vzduchu, ve kterém se vlna pohybuje.



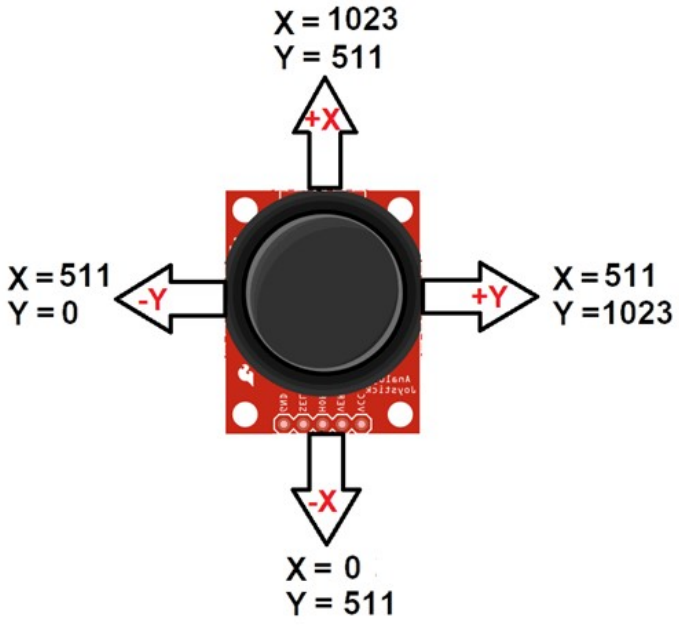
Obrázek 2.9 Obrázek Ultrazvukového snímače vzdálenosti HY-SRF05 (převzato z [17])

Typy ultrazvukového měřiče vzdálenosti jsou jednohlavé a dvouhlavé. Jednohlavé měřiče se vyznačují menším fyzickým provedením, ale delší dobou odezvy, jelikož se vysílač musí přepínat do přijímacího módu a nazpět. Dvouhlavé mohou zároveň vysílat i přijímat, což z nich dělá rychlejší a důvěryhodnější měřiče vzdálenosti. Vyhodnocovací člen na výstupu signálu upravuje výstupní signál na obdélníkový PWM signál, který je dále zpracováván mikroprocesorem.

Jelikož se jedná o aktivní součástku, tak musí být připojena na zdroj napětí nebo proudu a zásadně má jednu výstupní svorku, na které je často obdélníkový signál, který je lehčí ke zpracovávání mikroprocesory. Některé moduly mají i více svorek, například HY-SRF04 a HY-SRF05, což jsou už postavené moduly, kde je svorka TRIG určená jako vstup obdélníkového PWM signálu a svorka ECHO určená jako výstup obdélníkového PWM signálu. Pomocí těchto dvou svorek se budí a vyhodnocuje signál pomocí mikroprocesoru.

2.3.10 Joystick

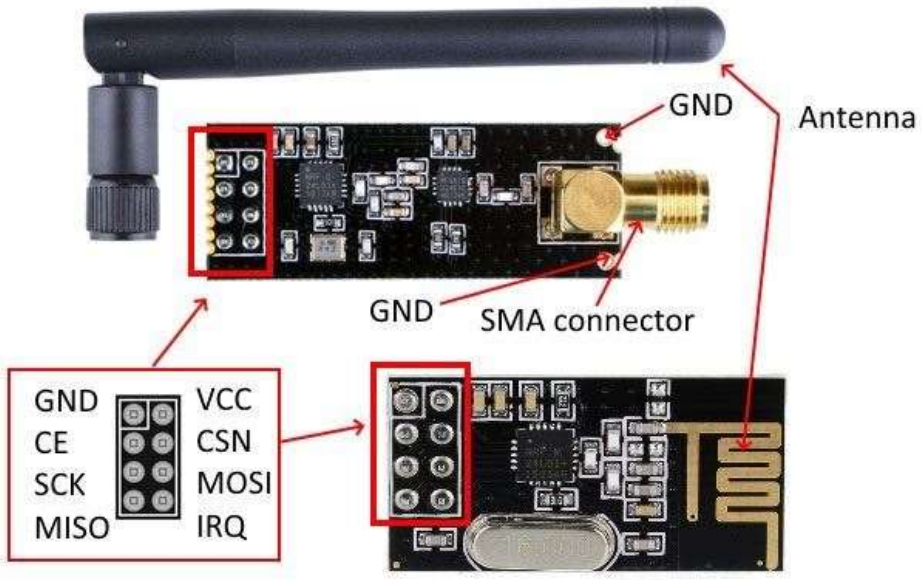
Jedná se o pasivní součástku, která je složena ze dvou potenciometrů a často i z tlačítka. Oba potenciometry jsou otočeny o 90°, aby snímaly osy X a Y pohybu Joysticku. Potenciometry se skládají z odporového pásku ve tvaru kruhu, na kraje je přivedeno napájecí napětí a zem, a po tomto pásku se pohybuje vodivý jezdec, který je určen polohou natočení joysticku. Poté se odečítá hodnota napětí mezi jedním koncem potenciometru a vývodem jezdce, tato hodnota se následně přepočítává na pozici v závislosti na rozložení napětí. Při použití mikrokontroleru se analogové napětí převádí na digitální hodnoty, pomocí A/D převodníku.



Obrázek 2.10 Obrázek modulu analogového joysticku(převzato z [18])

2.3.11 Bezdrátový modul – nRF24L01

Modul nRF24L01 je jednou z možností bezdrátového přenosu dat v pásmu 2,4 GHz. Tento modul komunikuje pomocí SPI sběrnice s mikroprocesorem. Modul funguje v pásmu 2,4 GHz, ve kterém funguje i známá WiFi síť, ale tento modul je schopný komunikace pouze se stejným modulem.



Obrázek 2.11 Obrázek modulu nRF24L01 v různých provedeních (převzato z [19])

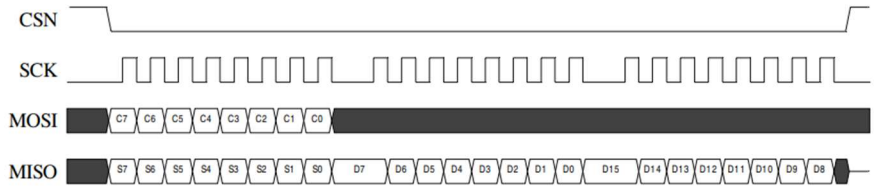
Modul nRF24L01 je složen ze stejnojmenného integrovaného čipu nRF24L01 a doplňujícím zapojením součástek pro správnou funkci čipu. Modul se vyrábí ve více variantách, jednou variantou je s integrovanou anténou, jiná má konektor k připojení externí antény, který zvýší dosah přenosu dat.

Modul má 8 pinů, dva jsou určeny pro napájení pomocí 3,3V, tři jsou určeny pro komunikaci pomocí SPI sběrnice. Jedná se o piny SCL, MOSI a MISO. Další dva piny jsou označené CE, který určuje, zda je čip aktivní a zda je ve stavu přijímače nebo vysílače, a CSN, který je adresou čipu pro komunikaci pomocí SPI. Poslední pin má označení IRQ a je určen pro vnější přerušení. Tento pin nemusí být připojen, ale ostatní musí. MOSI, CE, CSN jsou digitální vstupní piny, které pracují na 5 V logice. Piny MISO a IRQ jsou digitální výstupy také pracující na 5 V logice.

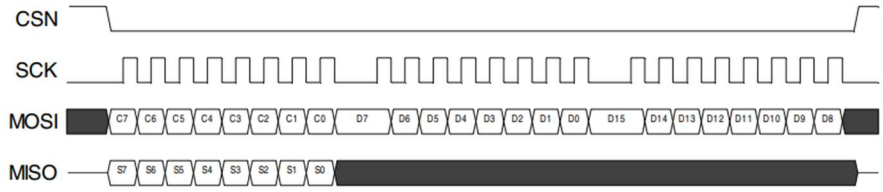
Modul pracuje v pásmu 2,4 GHz až 2,525 GHz, což mu umožňuje pracovat na určitém kanále v rozmezí těchto frekvencí. Výrobce udává mezeru mezi pásmy 1 MHz pro přenos s rychlostí 1000 kb/s anebo 2 MHz pro přenos s rychlostí 2000 kb/s. Pracovní frekvence se volí kódově pomocí knihoven určeným k modulu. Platí, že pro správný přenos dat mezi dvěma moduly musí být oba moduly na stejné pracovní frekvenci. Kódově lze nastavit sílu vyřazovaného signálu. Modul je schopný pracovat ve čtyřech módech, PA-MIN, PA-LOW, PA-HIGH a PA-MAX, jednotlivé módy určují zesílení vysílaného signálu.

Dále lze nastavit i přenosovou rychlost. Ta se pohybuje v rozmezí 0 až 2 Mb/s. Modul má kódově definované čtyři módy, 0,25 Mb/s, 0,5 Mb/s, 1 Mb/s a 2 Mb/s.

Pro správnou funkčnost modulu je zapotřebí správně připojit jednotlivé piny modulu a mikroprocesoru. Dále je zapotřebí správně nastavit oba moduly, stejnou pracovní frekvenci, dostatečnou sílu vysílaného signálu a správnou komunikaci po SPI sběrnici.



Obrázek 2.12 Výstřižek z technické dokumentace čipu nRF24L01 zobrazující časování SPI signálu na jednotlivých pinech sběrnice při přijímání dat (převzato z [20])



Obrázek 2.13 Výstřižek z technické dokumentace čipu nRF24L01 zobrazující časování SPI signálu na jednotlivých pinech sběrnice při odesílání dat (převzato z [20])

3. REALIZACE MODULU PRO INTERAKTIVNÍ ŘÍZENÍ

Tato část textu se zabývá praktickou částí práce, jako je otestování jednotlivých součástek a modulů, návrh DPS modulu, návrh kódu pro Arduino Uno R3, návrh šablony programu Pure Data anebo návrh ochranné krabičky.

3.1 Otestování jednotlivých komponent

Tato kapitola ukazuje, jak lze ozkoušet funkčnost jednotlivých periférií na nepájivém poli pomocí Arduino Uno R3 a kabelových propojek. Všechny ukázkové kódy jsou přiloženy v elektronické příloze.

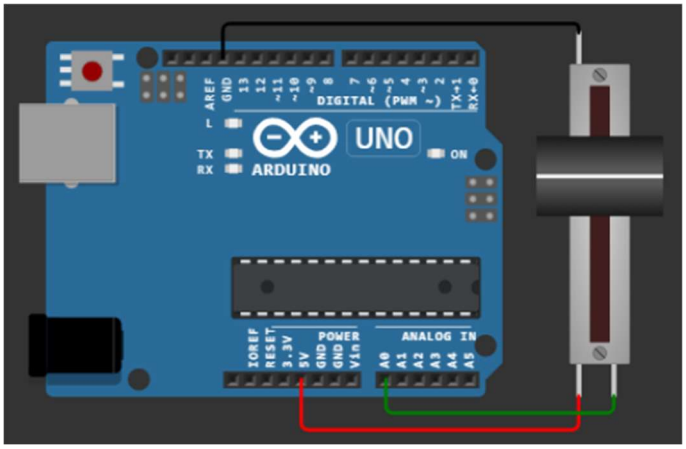
3.1.1 Tahový potenciometr

Pro otestování tahového potenciometru je použito schéma na obrázku 3.1. Dále je do Arduina nahrán následující program.

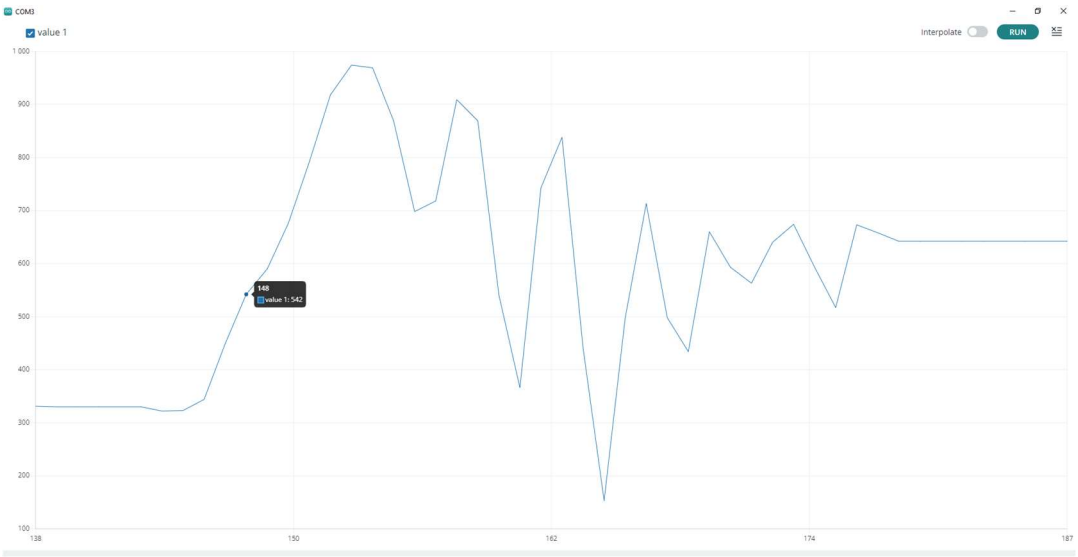
```
void setup() {
  Serial.begin(115200);
  pinMode(A0, INPUT);
}

void loop() {
  int value = analogRead(A0);
  Serial.println(value);
  delay(100);
}
```

Tento program čte analogovou hodnotu napětí na pinu A0, kde je připojen jezdec potenciometru. Dále je napětí převedeno pomocí 10bitového A/D převodníku, takže v proměnné `value` není přesná hodnota napětí, ale zaokrouhlená hodnota v rozmezí 0 až 1023. Tato hodnota se poté vypíše do sériového monitoru jako vizuální zpětná vazba.



Obrázek 3.1 Schéma připojení tahového potenciometru [21]



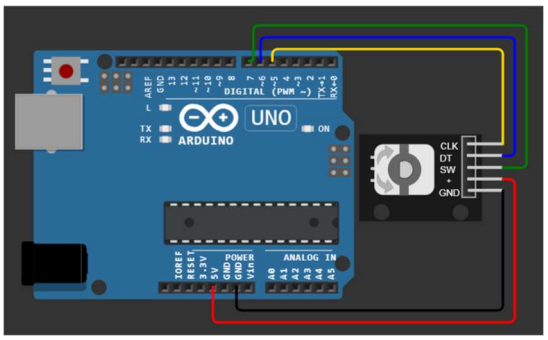
Obrázek 3.2 Výstřižek z výpisu sériového monitoru z programu Arduino IDE

3.1.2 Rotační Enkodér

Pro otestování funkčnosti rotačního enkodéru byl použit modul KY-040, který obsahuje inkrementální rotační enkodér, má 5 vývodů, VCC, GND, SW, DT a CLK. Zapojení pro otestování je na obrázku 3.3. Kód pro otestování funkčnosti byl použit ukázkový od elektronického prodejce Drátek.cz, který prodává tento modul. Kód lze najít na [34].

V tomto kódu se čte stav časovacího signálu, podle něj poznáme zda se osou otočilo. Dále se čte stav datového signálu, pokud je směr otáčení po směru hodinových ručiček, tak se vypíše „Rotace vpravo =>“ a přičte se jednička k pozici enkodéru, v opačném případě se vypíše „Rotace vlevo <=“ a odečte se jednička od pozice enkodéru. Dále nový stav přepíše starý stav v proměnné, aby se proces mohl opakovat. Pak je tu ještě podmínka pro ověření stisku tlačítka v rotačním enkodéru, tlačítko je přepnuto na active-low zapojení pomocí pull-up rezistoru.

Výsledkem tohoto testování je výpis ze sériového monitoru výše uvedených funkcí na obrázku 3.4.



Obrázek 3.3 Schéma zapojení modulu ky-040 rotačního enkodéru [31]

```
11:21:13.969 -> Rotace vlevo  <= | Pozice enkoderu: -12
11:21:20.235 -> Rotace vpravo => | Pozice enkoderu: -11
11:21:20.766 -> Rotace vlevo  <= | Pozice enkoderu: -12
11:21:21.281 -> Rotace vpravo => | Pozice enkoderu: -11
11:21:21.755 -> Rotace vlevo  <= | Pozice enkoderu: -12
11:21:22.280 -> Rotace vlevo  <= | Pozice enkoderu: -13
11:21:22.775 -> Stisknuto tlacitko enkoderu!
11:21:23.287 -> Rotace vlevo  <= | Pozice enkoderu: -14
11:21:23.774 -> Rotace vpravo => | Pozice enkoderu: -13
11:21:24.290 -> Stisknuto tlacitko enkoderu!
11:21:24.790 -> Rotace vlevo  <= | Pozice enkoderu: -14
11:21:25.289 -> Rotace vlevo  <= | Pozice enkoderu: -15
11:21:26.206 -> Rotace vlevo  <= | Pozice enkoderu: -16
```

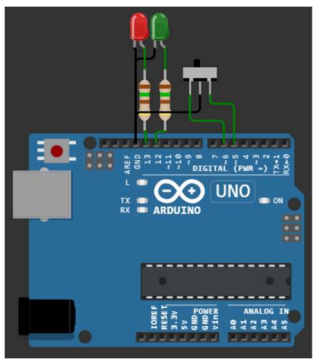
Obrázek 3.4 Výstřižek z výpisu sériového monitoru z programu Arduino IDE

3.1.3 Dvoupólový přepínač

Při otestování dvoupólových přepínačů proběhlo i otestování funkčnosti LED. Bylo použito zapojení na obrázku 3.5. Dále byl nahrán do Arduina následující kód.

```
void setup() {
  Serial.begin(115200);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(5, INPUT_PULLUP);
  pinMode(6, INPUT_PULLUP);
}
void loop() {
  digitalWrite(LED_BUILTIN, digitalRead((5)));
  digitalWrite(12, digitalRead((6)));
  if (digitalRead((5)) ){
  Serial.println("LED13 svítí");
  }else  Serial.println("LED12 svítí");
    delay(100);
}
```

V tomto kódu se čtou hodnoty digitálních pinů 5 a 6. Podle polohy přepínače svítí LED připojena k pinu 12 nebo 13. Dále je v kódu část pro výpis, která LED svítí.

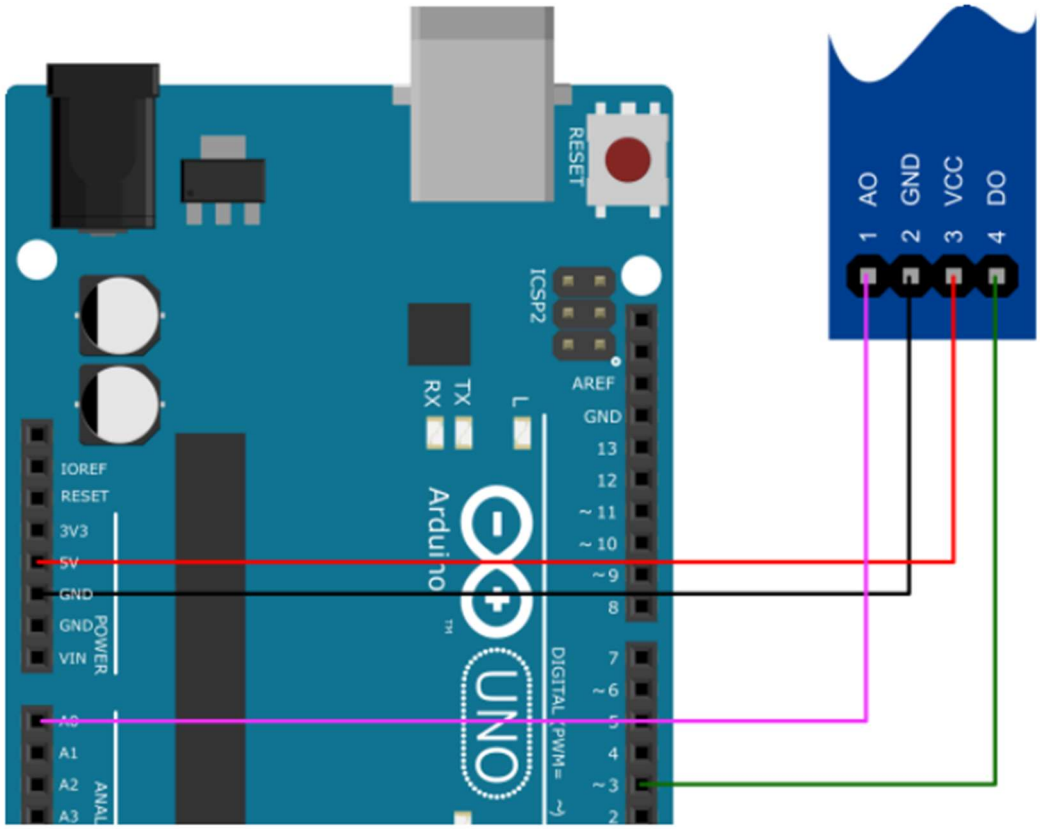



```
LED12 svítí
LED12 svítí
LED12 svítí
LED12 svítí
LED12 svítí
LED12 svítí
LED13 svítí
LED13 svítí
LED13 svítí
LED13 svítí
LED13 svítí
LED13 svítí
LED13 svítí
```

Obrázek 3.5 Schéma zapojení dvoupólového přepínače a výpis ze sériového monitoru (převzato z [28])

3.1.4 Mikrofon

K otestování funkčnosti a připojení mikrofonu byl použit modul SY-M213, který obsahuje elektretový mikrofon, analogový výstup, digitální výstup, jež je připojen na komparátor LM398. Ten porovnává hladinu hluku v okolí s přednastavenou hodnotou trimru. Modul je přizpůsoben k napájecímu napětí 5 V. Dále je modul doplněn několika odpory pro správnou funkčnost komparátoru a mikrofonu. U výstupní lišty jsou i dvě LED značící připojení modulu k napájecímu napětí a k překročení komparační hladiny a tedy k digitálnímu výstupu. Schéma zapojení je na obrázku 3.6. Kód je použit ukázkový od internetového prodejce Drátek.cz [22].

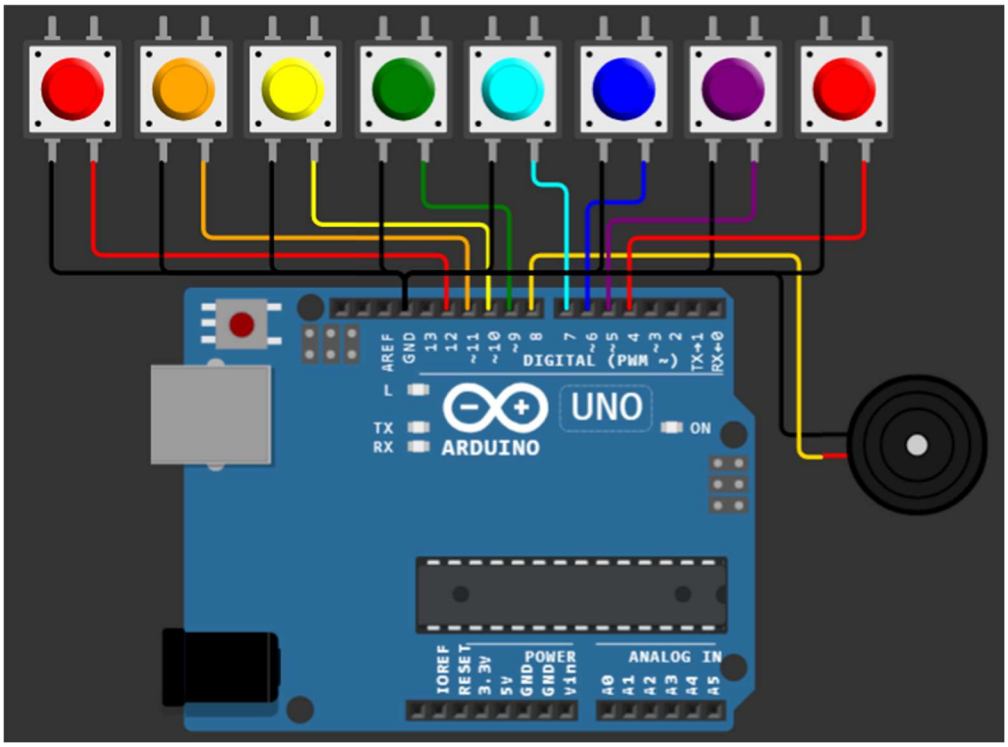


Obrázek 3.6 Schéma zapojení modulu ky-040 rotačního enkodéru (převzato z [23])

3.1.5 Piezoelektrický reproduktor

Při otestování piezoelektrického reproduktoru byly otestovány i tlačítka. Byl použit ukázkový projekt ze stránky wokwi [27]. Zapojení je podle schématu na obrázku 3.7.

V tomto kódu se při zmáčknutí tlačítka nastaví nota, která se bude hrát na reproduktoru.

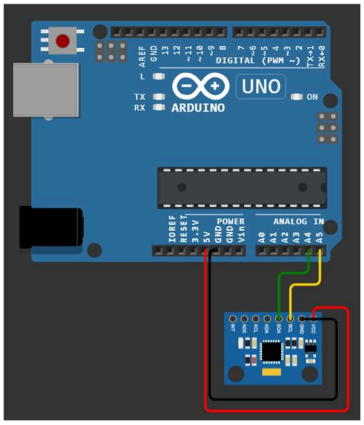


Obrázek 3.7 Schéma zapojení Piezoelektrického reproduktoru (převzato z [27])

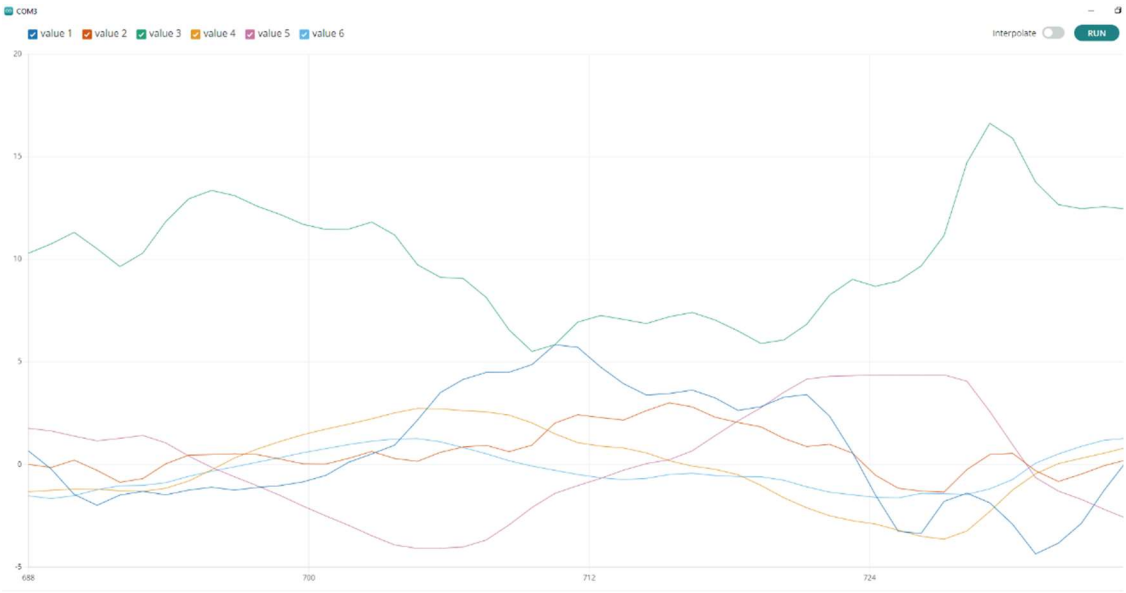
3.1.6 Gyroskop

Pro otestování gyroskopu byl použit ukázkový projekt ze stránek wokwi [26]. Gyroskop je připojen podle schématu na obrázku 3.8.

V tomto kódu se testuje, zda je gyroskop připojen k Arduinu přes I2C sběrnici. Dále se inicializuje, v jakých rozmezích gyroskop pracuje, tyto hodnoty jsou dány v knihovně od Adafruit pro modul MPU6050. Poté v loop části se jednotlivé hodnoty čipu gyroskopu vypisují do sériového monitoru. Výstupem je graf z programu Arduino IDE 2.2.1., kde jsou ukázány všechny průběhy x,y,z zrychlení a natočení.



Obrázek 3.8 Schéma zapojení modulu MPU6050 gyroskopu (převzato z [26])



Obrázek 3.9 Výstřižek z výpisu sériového plotteru z programu Arduino IDE

3.1.7 Ultrazvukový měřič vzdálenosti

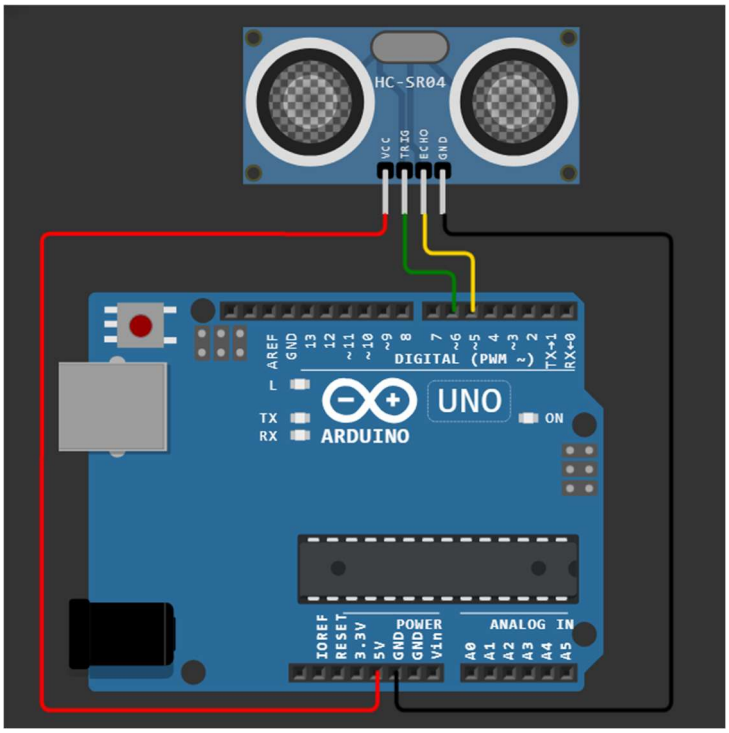
K otestování ultrazvukového měřiče vzdálenosti bylo použito zapojení na obrázku 3.10, pouze k otestování byl použit modul HC-SR05 místo HC-SR04, přičemž pin OUT není připojen. Dále byl do Arduina nahrán následující kód.

```
#define PIN_TRIG 5
#define PIN_ECHO 6

void setup() {
  Serial.begin(115200);
  pinMode(PIN_TRIG, OUTPUT);
  pinMode(PIN_ECHO, INPUT);
}

void loop() {
  digitalWrite(PIN_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG, LOW);
  int duration = pulseIn(PIN_ECHO, HIGH);
  Serial.print("Vzdálenost v centimetrech: ");
  Serial.println(duration / 58);
  delay(1000);
}
```

V tomto kódu se vyšle pulz o délce 10 µs s hodnotou HIGH přes pin Trig. Poté se měří čas pulzu na pinu Echo. Tato doba je určena rychlostí zvuku ve vzduchu a vzdáleností předmětu od modulu.



Obrázek 3.10 Schéma zapojení modulu HC-SR04 Ultrazvukového měřiče vzdálenosti [24]

```
21:04:15.527 -> Vzdálenost v centimetrech: 79
21:04:17.050 -> Vzdálenost v centimetrech: 139
21:04:18.061 -> Vzdálenost v centimetrech: 138
21:04:19.128 -> Vzdálenost v centimetrech: 137
21:04:20.092 -> Vzdálenost v centimetrech: 155
21:04:21.107 -> Vzdálenost v centimetrech: 137
21:04:22.115 -> Vzdálenost v centimetrech: 138
21:04:23.109 -> Vzdálenost v centimetrech: 155
21:04:24.137 -> Vzdálenost v centimetrech: 155
21:04:25.251 -> Vzdálenost v centimetrech: 74
21:04:26.269 -> Vzdálenost v centimetrech: 154
21:04:27.277 -> Vzdálenost v centimetrech: 4
21:04:28.284 -> Vzdálenost v centimetrech: 5
21:04:29.280 -> Vzdálenost v centimetrech: 10
21:04:30.261 -> Vzdálenost v centimetrech: 16
21:04:31.291 -> Vzdálenost v centimetrech: 15
21:04:32.412 -> Vzdálenost v centimetrech: 73
21:04:33.557 -> Vzdálenost v centimetrech: 74
21:04:34.569 -> Vzdálenost v centimetrech: 7
```

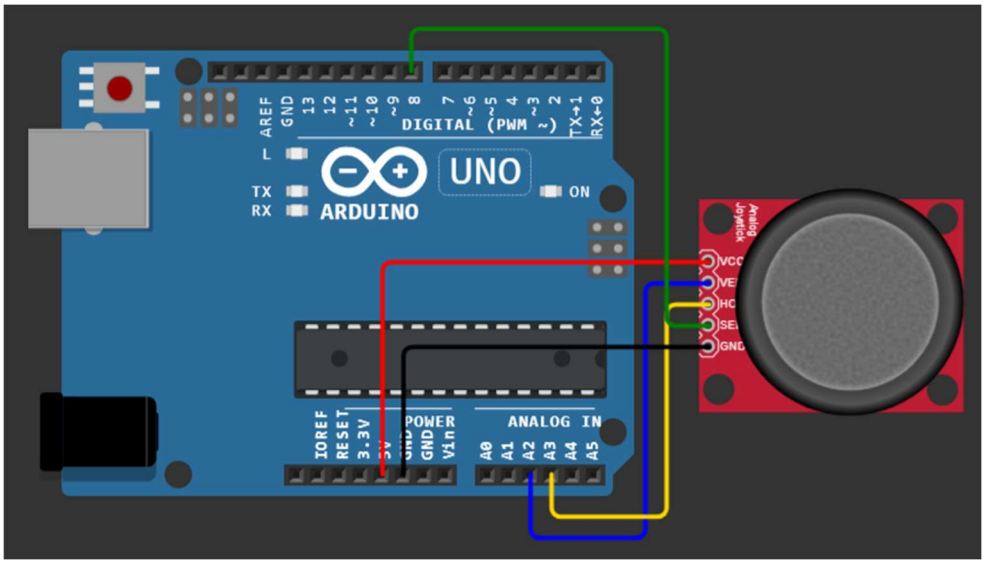
Obrázek 3.11 Výstřižek z výpisu sériového monitoru z programu Arduino IDE

3.1.8 Joystick

Otestování funkčnosti joysticku bylo provedeno připojením modulu analogového joysticku k Arduinu podle schématu na obrázku 3.12. Dále byl nahrán následující kódu.

```
#define VERT_PIN A2
#define HORZ_PIN A3
#define SEL_PIN  8
void setup() {
  Serial.begin(115200);
  pinMode(VERT_PIN, INPUT);
  pinMode(HORZ_PIN, INPUT);
  pinMode(SEL_PIN, INPUT_PULLUP);
}
void loop() {
  Serial.print("X: ");
  Serial.print(map(analogRead(VERT_PIN), 0, 1023, -100, 100));
  Serial.print(" Y: ");
  Serial.print(map(analogRead(HORZ_PIN), 0, 1023, -100, 100));
  if(digitalRead(SEL_PIN)==LOW){
    Serial.print(" Tlačítko bylo zmáčknuto");
  }
  Serial.println("");
  delay(1000);
}
```

V tomto kódu je výpis souřadnic analogového joysticku, která je pomocí funkce map() přepočítána do rozmezí od -100 do 100. Poté je zde funkce, která kontroluje, zda bylo tlačítko zmáčknuto. Tlačítko je připojeno k digitálnímu pinu 8, který má programově zapnutý pull-up odpor, takže tlačítko je zmáčknuto, pokud je na pinu hodnota LOW. Výstupem je výpis pozice joysticku, který je mapovaný od -100 do 100.



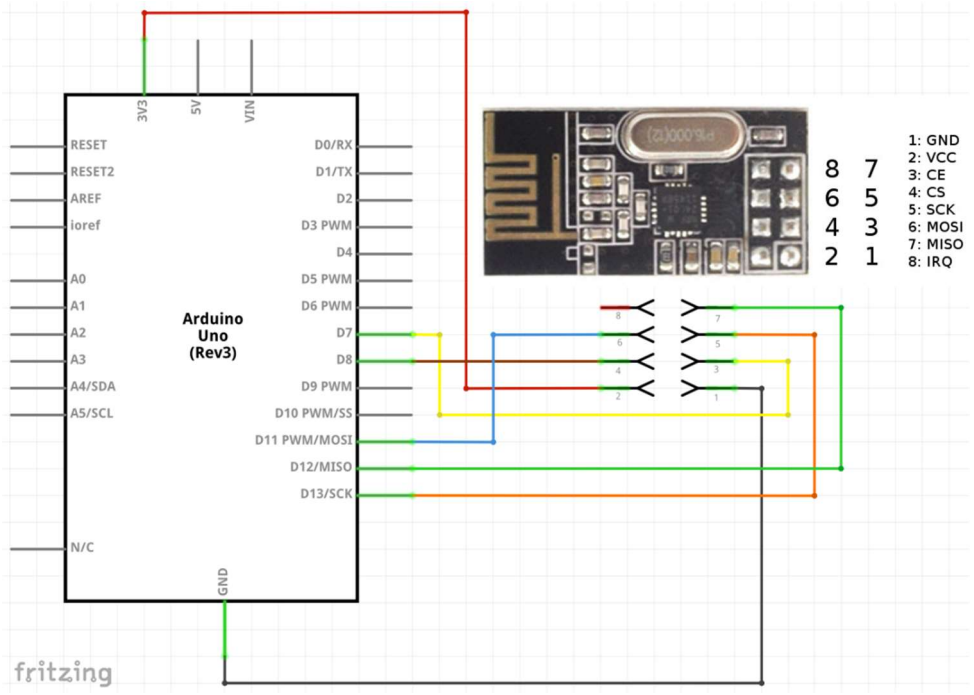
Obrázek 3.12 Schéma zapojení modulu Analogového joystiku [25]

```
20:44:41.099 -> X: 0 Y: 2 Tlačítko bylo zmáčknuto
20:44:42.110 -> X: -2 Y: -100
20:44:43.105 -> X: 98 Y: -100
20:44:44.098 -> X: 100 Y: 2
20:44:45.106 -> X: 100 Y: 7
20:44:46.115 -> X: 0 Y: 2
20:44:47.120 -> X: 9 Y: -100
20:44:48.117 -> X: -17 Y: 2
20:44:49.111 -> X: 0 Y: 100
20:44:50.123 -> X: 0 Y: 2
20:44:51.112 -> X: 0 Y: 2
20:44:52.076 -> X: 0 Y: 2 Tlačítko bylo zmáčknuto
20:44:53.116 -> X: -100 Y: -75
20:44:54.111 -> X: 0 Y: -82
```

Obrázek 3.13 Výstřižek z výpisu sériového monitoru z programu Arduino IDE

3.1.9 Bezdrátový modul – nRF24L01

Otestování bezdrátového modulu a komunikace bylo provedeno pomocí dvou modulů nRF24L01, které fungují ve frekvenčním pásmu 2,4GHz, obdobně jako WIFI. Pro otestování bylo použito ukázkového kódu ze stránek internetového prodejce Drátek.cz. Schéma zapojení je pro vysílač i přijímač stejné a je zobrazeno na obrázku 3.14.



Obrázek 3.14 Schéma zapojení modulu ky-040 rotačního enkodéru (převzato z [30])

Celé kódy jsou v elektronické příloze pod názvem prij.ino a vys.ino. Při testování bezdrátové komunikace mezi jedním vysílačem a jedním přijímačem je důležité mít stejné jméno pro vysílač i přijímač v obou programech. Dále je důležité mít moduly připojené k napájecímu napětí 3,3 V. V neposlední řadě je důležité mít nastaveny oba moduly stejné pásmo, stejný výkon a stejnou přenosovou rychlost, jinak může dojít k nedokonalému přenosu dat, či ztrátě dat. Ke komunikaci prostřednictvím tohoto modulu je nutno do programu Arduino IDE nahrát požadované knihovny uvedené na začátku kódu.

Kódy jsou umístěny v elektronické příloze. U vysílače je to dáno cyklem for, který posílá tři různé předvolby a poté měří čas za jak dlouho dostane odezvu. Poté je tam další část, která se stará o chyby a výpis do sériového monitoru. U přijímače je délka kódu udána hlavně switch podmínkovým stromem, který podle předvolby přijatých dat vypisuje délku odezvy buď v milisekundách, v sekundách anebo v mikrosekundách.

```
11:46:42.636 -> Posilam volbu 0
11:46:42.669 -> Odeslana volba: 0, prijata data: 0
11:46:42.702 -> Delka spojeni: 1852 mikrosekund.
11:46:43.680 -> Posilam volbu 1
11:46:43.680 -> Odeslana volba: 1, prijata data: 8026
11:46:43.714 -> Delka spojeni: 1856 mikrosekund.
11:46:44.697 -> Posilam volbu 2
11:46:44.731 -> Odeslana volba: 2, prijata data: 9
11:46:44.764 -> Delka spojeni: 1852 mikrosekund.
11:46:45.723 -> Posilam volbu 3
11:46:45.754 -> Odeslana volba: 3, prijata data: 10074224
11:46:45.797 -> Delka spojeni: 1824 mikrosekund.
11:46:46.756 -> Posilam volbu 0
11:46:46.756 -> Odeslana volba: 0, prijata data: 0
11:46:46.823 -> Delka spojeni: 1824 mikrosekund.
11:46:47.803 -> Posilam volbu 1
11:46:47.803 -> Odeslana volba: 1, prijata data: 12125
11:46:47.836 -> Delka spojeni: 1820 mikrosekund.
11:46:48.788 -> Posilam volbu 2
11:46:48.821 -> Odeslana volba: 2, prijata data: 13
11:46:48.853 -> Delka spojeni: 1884 mikrosekund.
11:46:49.840 -> Posilam volbu 3
11:46:49.840 -> Odeslana volba: 3, prijata data: 14175160
11:46:49.874 -> Delka spojeni: 1828 mikrosekund.
11:46:50.869 -> Posilam volbu 0
11:46:50.869 -> Odeslana volba: 0, prijata data: 0
11:46:50.901 -> Delka spojeni: 1824 mikrosekund.
11:46:51.864 -> Posilam volbu 1
11:46:51.897 -> Odeslana volba: 1, prijata data: 16226
11:46:51.929 -> Delka spojeni: 1824 mikrosekund.
11:46:52.918 -> Posilam volbu 2
11:46:52.918 -> Odeslana volba: 2, prijata data: 17
11:46:52.958 -> Delka spojeni: 1852 mikrosekund.
11:46:53.936 -> Posilam volbu 3
11:46:53.936 -> Odeslana volba: 3, prijata data: 18276104
11:46:54.003 -> Delka spojeni: 1824 mikro
```

Obrázek 3.15 Výpis sériové konzole pro komunikaci pomocí nRF24 [23]

3.2 Návrh desky plošného spoje

Tato kapitola ukazuje, jak lze postupovat při návrhu DPS, což je základní stavební kámen, pokud je předpokladem dlouhodobá funkční životnost projektu.

Při návrhu je důležité myslet na rozměry desky a použitých komponent. Deska v této práci je velikostně přizpůsobena rozměrům Arduina Una R3. Při návrhu desky s těmito rozměry se často hovoří jako Arduino shield, který obsahuje komponenty a je přizpůsoben k rychlému připojení Arduina, nahrání ukázkového kódu a funkčnosti zapojení. Tohoto principu se drží i tato práce.

Projekty jednotlivých verzí jsou přidány do elektronické přílohy.

Schémata a 3D modely jednotlivých verzích jsou umístěny v Příloha A -.

3.2.1 Verze 1.0

Verze 1.0 byla vytvořena pomocí webového prostředí flux.ai, kde byly k dispozici knihovny se všemi součástkami, které se v této práci využívají. Dále zde byla i předloha k Arduino shieldu, která určuje velikost desky a rozložení propojovacích konektorů pro Arduino. Toto webové prostředí vyniká svou jednoduchostí, přidáváním součástek vytvořených komunitou a dostupností technické dokumentace v informačním listu součástky.

Ve verzi 1.0 jsou následující komponenty: joystick, tahový potenciometr, ultrazvukový měřič vzdálenosti HC-SR04, inkrementální rotační enkodér s kvadratickým výstupem bez tlačítka, dva dipólové přepínače typu SPST, piezoelektrický reproduktor v SMD provedení, šesti osý gyroskop MPU-6050 v SMD provedení, elektretový piezoelektrický mikrofon POM-3042P-R, ESP-01 bezdrátový WiFi modul, osm tlačítek a osm LED, které jsou připojeny k GPIO rozšiřovacímu čipu MPC-23017 komunikujícímu přes I^2C sběrnici. Dále je obvod vybaven pull-up rezistory na I^2C sběrnici a blokujícími kondenzátory pro správnou funkčnost gyroskopu podle výrobního katalogu.

V tomto návrhu je gyroskop umístěn na spodní straně desky tak, aby se na horní část vešly všechny komponenty, které uživatel musí ovládat ručně. Dále je na spodní stranu vyvedena i lišta k odnímatelnému připojení ultrazvukového měřiče vzdálenosti. Modul EPS-01 byl zamýšlen k osazení do lišty na kraji desky v 90° poloze, což by zase ušetřilo nějaké místo na desce.

3.2.2 Verze 1.1

Ve verzi 1.1 se hledalo optimální osazení součástek pro danou velikost desky, seznam součástek je stejný, u několika součástek bylo změněno provedení. U piezoelektrického reproduktoru a LED se přešlo z SMD provedení na THT, což vede k možnosti táhnout cesty na spodní straně desky, jako například GND. Ultrazvukový senzor je opět umístěn zespodu desky, pouze teď je přes něj tlačítko, takže při osazování by tento návrh nebyl optimální.

3.2.3 Verze 1.2

Ve verzi 1.2 se přešlo od modulu ESP-01 na modul nRF24L01, který komunikuje pomocí SPI sběrnice. Z tohoto důvodu byl redukován i počet tlačítek i LED na šest. Díky tomuto přechodu bylo možné přesunout gyroskop i ultrazvukový senzor vzdálenosti na horní stranu desky, aby oba komponenty byly viditelné pro uživatele. Dvoupólové přepínače byly nahrazeny provedením SPDT k snazší obsluze uživatelem. Dále byly připojeny k MPC 23017. Dále byl přidán rotační enkodér s tlačítkem a obměněn konektor k ultrazvukovému měřiči vzdálenosti, pro modul HC-SR05.

3.2.4 Verze 1.3

Ve verzi 1.3 se dále obměňovaly pozice součástek, aby se docílilo co nejkratších cest mezi komponenty a Aduinem. A také aby se zamezilo co nejméně křížení a obcházení cest. Dále se při návrhu této verze začal vytvářet i návrh na použití krabičky, aby se předešlo k neopatrnému zacházení a zničení komponent pomocí elektrostatických výbojů. V tomto návrhu se přešlo zpět na SMD provedení LED a přesunu ultrazvukového snímače vzdálenosti na spodní stranu desky.

3.2.5 Verze 2.0

Ve verzi 2.0 se přešlo z vývojového prostředí Flux.ai na program KiCad 7.0, který je volně dostupný na internetu. Seznam součástek je stejný jako ve verzi 1.3. Připojení k Arduinu zůstalo také stejné, pouze se některé součástky přesunuly na desce, kvůli ušetření místa.

Tato verze se nechala vyrobit u čínské firmy JLCPCB a byla osazena. Při osazování a oživování se naskytly následující komplikace, které jsou v následující verzi vyřešeny. Pro kolíkové lišty byl vybrán špatný rozměr, místo 2,54 mm byl rozměr 2mm, to se týká připojení ultrazvukového snímače vzdálenosti a modulu pro bezdrátovou komunikaci nRF24L01. Dále byl špatně vytvořen model pro Joystick, kde nohy potenciometrů úplně nepasovaly na desku a zemnící nohy z kovového těla měly moc malé díry na desce. Na SDA a SCL linkách u I2C komunikace byly rezistory připojeny jako pull-down místo pull-up. U čipu mcp23017 chyběl k funkčnosti 10 kΩ pull-up rezistor a 100 nF filtrovací kondenzátor připojen k RESET pinu. Bez těchto součástek čip nešel najít v I2C adresách a tudíž nešel kontrolovat.

3.2.6 Verze 2.1

Ve verzi 2.1 se upravily chyby u návrhu verze 2.0. Velikosti kolíkových lišt byly sjednoceny všechny na 2,54 mm. Byl přepracován model joysticku podle nákresu ze stránek [32][31]. Rezistory na SCL a SDA linkách I^2C komunikace byly správně připojeny jako pull-up. Dále byly přidány součástky k MCP23017 reset pinu pro správnou funkčnost. Poté byl přidán zesilovací obvod k mikrofonu a odstraněn předřadný rezistor od tahového potenciometru.

3.3 Program pro Arduino Uno

Program pro Arduino Uno byl psán v programovacím prostředí Arduino IDE 2.0 v jazyce C++. Pro některé součástky jsou použity knihovny funkcí od výrobce či komunity. Pro propojení Arduina a Pure Data je zapotřebí požadovaná data ze senzorů posílat přes sériovou linku, odkud je program Pure Data schopný číst data a následně je rozdělit na jednotlivé hodnoty senzorů. Pro zpětnou komunikaci z Pure Data do Arduina se také používá sériové konzole, z které Arduino čte hodnoty pro nastavení LED a piezoelektrického reproduktoru.

V rámci práce jsou vytvořené tři verze programu, v první verzi není zahrnuta bezdrátová komunikace pomocí modulů nRF24L01. Na začátku jsou vloženy knihovny k jednotlivým komponentám, které potřebují podpůrné funkce, například k čipu mcp23017. Dále jsou v programu definovány jednotlivé komponenty a piny, ke kterým jsou připojeny, a proměnné potřebné pro tyto komponenty. Poté následuje funkce `setup`(), která je základní funkcí Arduino programů a spustí se pouze jednou. V této funkci se inicializuje komunikace po sériové lince a nastavují se jednotlivé piny, zda jsou vstupní nebo výstupní periferie. Dále následuje funkce `loop(),` která je také základní funkcí Arduino programů a spouští se opakovaně za sebou. V této funkci se čtou analogové a digitální hodnoty jednotlivých komponent a tisknou se do sériového monitoru, mezi jednotlivé hodnoty jsou vloženy mezery. Poté se v této funkci nachází čtení sériové komunikace a uložení do proměnné. Dále se pomocí logiky porovnávají hodnoty z proměnné a zapínají či vypínají se jednotlivé LED, porovnávací logika funguje na porovnávání jednotlivých bitů, které odpovídají jednotlivým LED. Na konci je funkce `delay(x),` která pozdrží celý program na x milisekund, aby se zamezilo příliš častému čtení dat, což by mělo za následek zahlcení sériové komunikace a degradování načtených hodnot.

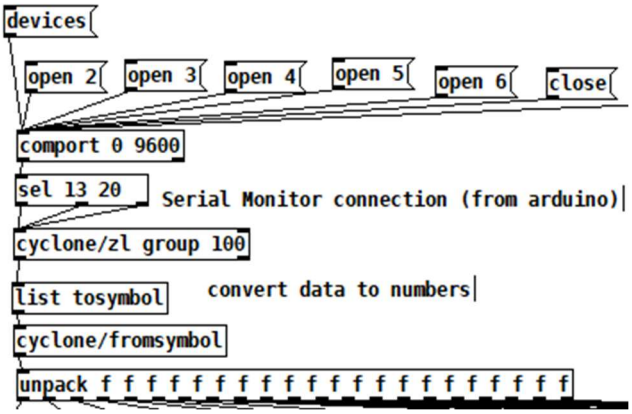
Ve druhé a třetí verzi byla zahrnuta i bezdrátová komunikace pomocí modulu nRF24L01. Druhá verze je určena jako vysílač a třetí verze je přijímač. Kvůli bezdrátové komunikaci se naměřené hodnoty musí ukládat do proměnné typu struktury, pomocí této proměnné se dají data přečíst, uložit a odeslat přes sériovou linku i přes bezdrátovou komunikaci. Struktura programu je podobná jako v první verzi. Ve funkci `setup`() přibylo nastavení nRF24 komunikace, kde se nastavuje kanál, vysílací výkon a rychlost přenosu dat. Pro správné připojení nRF24 modulů musí být nastavena stejná proměnná `pipe`, stejná přenosová rychlost a stejný kanál.

Třetí verze programu je určena jako přijímač dat pro komunikaci pomocí modulů nRF24L01. V tomto programu je inicializace komunikace stejná jako v druhé verzi, struktura dat je také stejná. Po příjmu dat se data uloží do struktury a poté se vypíší do sériové komunikace, kde je program Pure Data schopný tyto data přečíst. S tímto programem není možné používat LED část přípravku. Tento program je určen pro ovládání Pure Data pomocí přípravku na větší vzdálenost, než je délka USB kabelu.

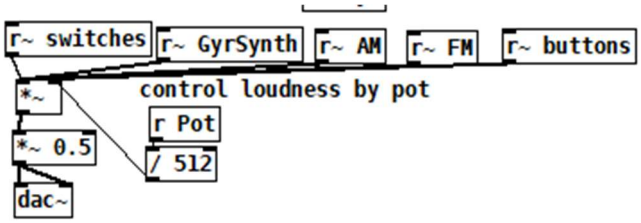
Všechny kódy jsou přiloženy v Příloha B -. Při optimalizování kódu a opravování chyb v kódu bylo použito AI nástrojů Blackbox.ai [35].

3.4 Program v PureData

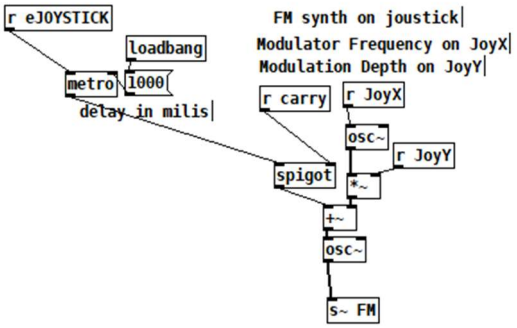
Pro komunikaci mezi PureData a Arduino Uno se používá sériový COM port. Tato komunikace se snadno naváže pomocí pár patchů, jednotlivé patche jsou dále vysvětleny. Po příjmu dat a rozdělení do jednotlivých proměnných lze s daty dále pracovat. S určitými daty lze dělat určité akce, jako například s analogovými daty lze měnit hlasitost programu anebo syntetizovat zvuk. Pomocí digitálních dat lze například spouštět zvuky nebo vytvářet ambientní beat. Výstřižek z celkového programu je umístěn v Příloha C -.



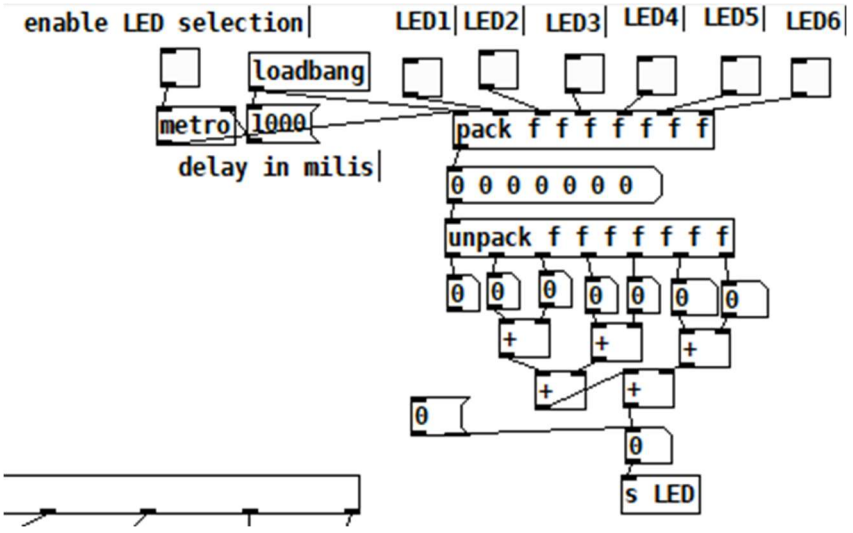
Obrázek 3.16 Výstřižek části kódu zajišťující čtení dat ze sériové komunikace



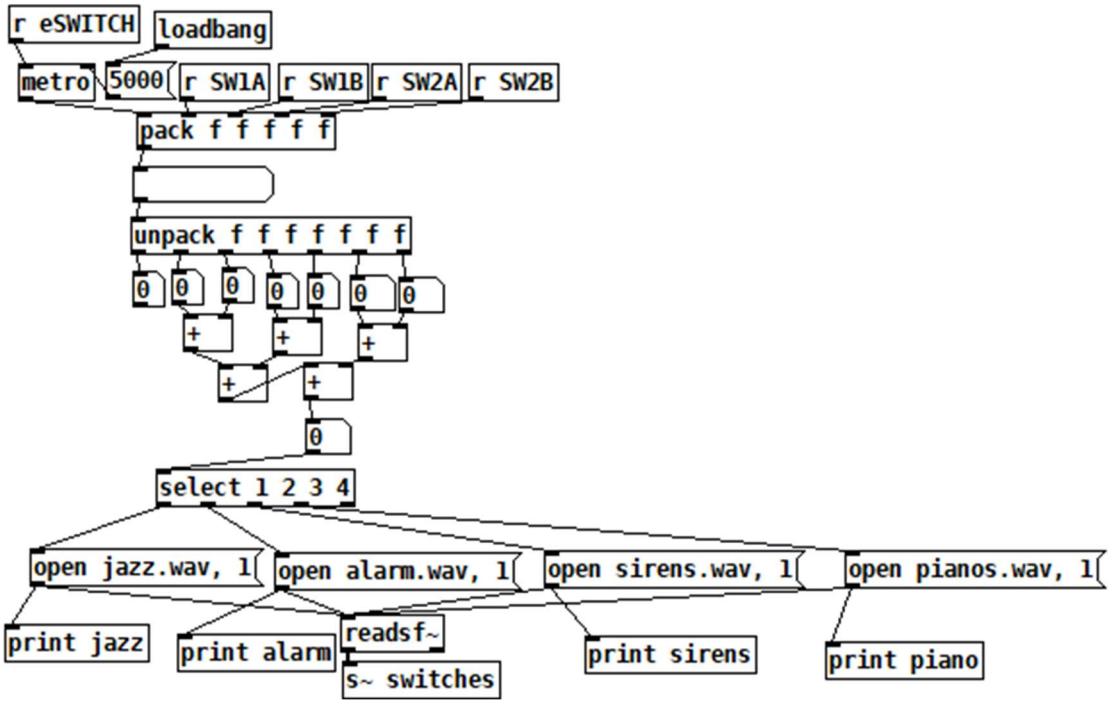
Obrázek 3.17 Výstřižek části kódu zajišťující přehrávání zvuků na počítači



Obrázek 3.18 Výstřižek části kódu pro FM modulaci zvuku o nosné frekvenci



Obrázek 3.19 Výstřižek části kódu obsluhující zpětnou vazbu pomocí LED



Obrázek 3.20 Výstřižek části kódu zajišťující přehrání zvuku v závislosti na poloze přepínačů

K práci s programem je nutné ponechat část kódu z obrázku 28, dále je uživatel schopný jednotlivá data upravovat pomocí matematických funkcí, vymýšlet vlastní logiku pro digitální data a tvořit vlastní program, který bude vytvářet hudbu dle jeho požadavků.

3.5 Návrh ochranné krabičky

V uvažování nad bezpečností elektroniky byla vytvořena plastová krabička k zamezení dotyku studentů součástek na desce a tudíž k poškození součástek elektrostatickými výboji. Dále má krabička účel zpevnit konstrukci a ulehčit manipulování s přípravkem.

Krabička je rozdělena do tří dílů, kde hlavní tělo má na spodní straně 4 díry na závity typu M3, kam se připevní Arduino uno pomocí distančních sloupků a ty připevní navrhovanou desku. Poté má na přední straně dva otvory na USB konektor a napájecí jack konektor. Tato část obklopuje pouze tři strany, aby bylo snazší připevnění Arduino desky k podstavě a ultrazvukového měřiče vzdálenosti ze zadní strany.

Na zadní straně je další plastová deska s vyříznutým otvorem na jednotlivé hlavy ultrazvukového snímače. Zadní deska se přišroubuje čtyřmi šrouby typu M3 k hlavní části krabičky. Výška obou částí je 63 mm.

Na horní části bude umístěna deska z průhledného materiálu s vyříznutými otvory na jednotlivé komponenty. Dále jsou zde čtyři díry pro šrouby typu M3, které přišroubují vrchní desku k distančním sloupkům držícím Arduino uno k hlavní části krabičky a k navržené desce. Tyto šrouby zpevní celou krabičku aby Arduino drželo a mechanicky se neopotřebovávalo.

Jednotlivé výkresy jsou přiloženy v Příloha D -.

4. ZÁVĚR

4.1 Závěr otestovaných komponent

Při testování součástek nebyl žádný velký problém, jelikož bylo postupováno podle ukázkových zapojení a byly použity ukázkové kódy. Velký problém byl s navázáním komunikace mezi dvěma bezdrátovými moduly nRF24L01. Hlavní problém zde byl SPI protokol a napájecí napětí, jelikož modul je velmi citlivý na úroveň napětí a hlavně na šum, který se přenáší po napájecích kabelech. Při testování byl modul připojen k napájecímu napětí +5 V a komunikace fungovala, po přepojení obou modulů k napájecímu napětí +3,3 V se modul vysílače nedokázal připojit k modulu přijímače, a tudíž nebyla komunikace navázána správně. Při návrhu DPS se vycházelo z otestovaných součástek, které se poté umístili na desku, tak aby se optimalizovaly cesty propojení jednotlivých komponent. Deska má vnější velikost obdobnou jako deska Arduino Uno R3 a je dvouvrstvá. V práci se nachází celkem šest verzí návrhu DPS, první čtyři verze jsou z prostředí flux.ai, které bylo dobré pro prvotní návrh desky, ale scházela tomu možnost posouvání názvů a hodnot součástek po desce. Další dvě verze jsou z programu KiCad 7.0, které je více vhodné pro navrhování profesionálních DPS. K práci byla vytištěna verze 2.0, která obsahuje pár návrhových chyb, tyto chyby jsou však v další verzi opraveny. Všechny schémata a 3D modely jednotlivých verzí jsou v Příloha A -. Gerber data, flux projekty a KiCad projekty jsou v elektronické příloze ve formátu .zip. Dalším úkolem práce bylo vytvořit kód pro Arduino, které bude zaznamenávat hodnoty z jednotlivých komponent, popřípadě je upravovat, a posílat je přes sériovou komunikaci do programu Pure Data. Celkem jsou v práci vypracované tři verze, kde první verze neobsluhuje bezdrátovou komunikaci, druhá verze je vysílačem bezdrátové komunikace, který je umístěn na Arduinu s vývojovou deskou. Třetí verze je poté přijímač, který je umístěn na Arduinu připojeném k PC. Všechny kódy jsou nachystány ke komunikaci se všemi komponentami, ale v kódu chybí implementace ovládání piezoelektrického reproduktoru, která se bude provádět obdobně jako ovládání LED. Dalším úkolem práce bylo vytvořit šablonu v programu Pure Data, kterou poté uživatel upraví dle vlastní potřeby. Tento se skládá z části navázání komunikace pomocí `comport,` poté z částí zpracující digitální a analogové signály, analogové signály jsou převedeny pomocí A/D převodníku na digitální data, a na část pro zpětnou komunikaci, např s LED. Většina kódu byla inspirována z youtube kanálu uživatele SoundSimulator [33]. Dalším přidaným bodem byl návrh ochranné krabičky na desku Arduino Uno R3 a navrhovanou desku. Tato krabička se skládá z tří částí, kde hlavní část obklopuje čtyři strany, zadní strana má vlastní plastovou desku a horní strana bude z průhledného materiálu s vyříznutými dírami pro obsluhu komponent.

LITERATURA

[1] *Arduino UNO R4 Minima*, © 2023 Arduino. Online. Arduino. Dostupné z: https://docs.arduino.cc/hardware/uno-r4-minima. [cit. 2023-12-27].

[2] *Arduino*, 2023. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2001-2023. Dostupné z: https://cs.wikipedia.org/wiki/Arduino. [cit. 2023-12-27].

[3] *What is Arduino?*, © 2023. Online. Arduino.cc. Last revision February 05, 2018, at 08:43 PM. Dostupné z: https://www.arduino.cc/en/Guide/Introduction. [cit. 2023-12-27].

[4] *Arduino UNO R3*, © 2023. Online. Arduino. Dostupné z: https://docs.arduino.cc/hardware/uno-rev3. [cit. 2023-12-27].

[5] June, 2014. *I2C Signals*. Online. HIENZSCH, Dan. Rheingoldheavy. Dostupné z: https://rheingoldheavy.com/i2c-signals/. [cit. 2023-12-27].

[6] *Serial Peripheral Interface: SPI*, 2023. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2001-2023, naposledy editována 19. 3. 2022 v 23:57. Dostupné z: https://cs.wikipedia.org/wiki/Serial_Peripheral_Interface. [cit. 2023-12-27].

[7] NOVOTNÝ, B. Sériové komunikace. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 39 s. Vedoucí bakalářské práce Ing.Zdeněk Martinásek. Dostupná na adrese: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=42245

[8] *Grove - Slide Potentiometer*, © 2023. Online. In: SEEED STUDIO, INC. BUILT WITH DOCUSAURUS. Seeedstudio. Dostupné z: https://wiki.seeedstudio.com/Grove-Slide_Potentiometer/. [cit. 2023-12-27].

[9] *Potenciometr*, 2023. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2001-2023, 13. 3. 2023 v 11:56. Dostupné z: https://cs.wikipedia.org/wiki/Potenciometr. [cit. 2023-12-29].

[10] *What is Rotary Encoder?*, 2022. Online. HOW TO ELECTRONICS. How2electronics. Updated: August 22, 2022. Dostupné z: https://how2electronics.com/construction-working-rotary-encoder/. [cit. 2023-12-27].

[11] TEJA, Ravi, 2021. *What is a Switch?* Online. In: Electronicshub.org. May 3, 2021. Dostupné z: https://www.electronicshub.org/switches/. [cit. 2023-12-29].

[12] *What is the symbol of a light emitting diode?*, c2023. Online. In: Quora. 2018. Dostupné z: https://www.quora.com/What-is-the-symbol-of-a-light-emitting-diode. [cit. 2023-12-29].

[13] *How to Indicate Placement Orientation of LED on Your Boards*, © 2023. Online. In: Raypcb. Dostupné z: https://www.raypcb.com/indicate-placement-orientation-of-led/. [cit. 2023-12-29].

[14] SALEEM, Kiran, © 2022. *Simple Mic Circuit*. Online. In: Circuits-diy. December 21, 2022. Dostupné z: https://www.circuits-diy.com/simple-mic-circuit/. [cit. 2023-12-29].

[15] *Buzzer symbol*, © 2023. Online. In: TheoryCIRCUIT. Dostupné z: https://theorycircuit.com/electronic-components-and-circuit-diagram-symbols/buzzer-symbol/. [cit. 2023-12-29].

[16] ELECTROTECH, Nandini, ©1996-2023. *MPU-9250*. Online. In: IndiaMART.com. Dostupné z: https://m.indiamart.com/proddetail/mpu-9250-9-axis-attitude-gyro-accelero-and-magnetometer-sensor-module-2852053629588.html. [cit. 2023-12-29].

[17] *How To Use The HY-SRF05 Ultrasonic Distance Sensor*, © 2023. Online. In: Jawher Sebai. Dostupné z: https://jawhersebai.com/tutorials/how-to-use-the-hy-srf05-ultrasonic-distance-sensor/. [cit. 2023-12-29].

[18] *Joystick Module*, © 2023. Online. In: Components101. 2 April 2018. Dostupné z: https://components101.com/modules/joystick-module. [cit. 2023-12-29].

[19] *Adaptér pro bezdrátový modul NRF24L01*, c 2023. Online. In: .neon-el. Dostupné z: https://www.neon-el.cz/adapter-pro-bezdratovy-modul-nrf24l01. [cit. 2023-12-29].

[20] *NRF24L01-datasheet*, © 2023. Online. Alldatasheet.com. Dostupné z: https://www.alldatasheet.com/view.jsp?Searchword=Nrf24l01%20datasheet&gad_source=1. [cit. 2023-12-29].

[21] *Pot.ino*, © 2021 - 2023. Online. CODEMAGIC LTD. Wokwi. Dostupné z: https://wokwi.com/projects/385377649506562049. [cit. 2023-12-29].

[22] *Modul Mikrofonu s Analogovým Výstupem*, © 2023. Online. ECLIPSERA S.R.O. Dratek.cz. Dostupné z: https://navody.dratek.cz/navody-k-produktum/modul-mikrofonu-s-analogovym-vystupem.html. [cit. 2023-12-29].

[23] *Eses modul mikrofonu pro jednodeskové počítače*, © 2023. Online. In: ESES. Dratek.cz. Dostupné

z: https://dratek.cz/docs/produkty/0/753/eses1467272055.pdf. [cit. 2023-12-29].

[24] *Wokwi-hc-sr04 Reference*, © 2021-2023. Online. CODEMAGIC LTD. Wokwi. Dostupné z: https://docs.wokwi.com/parts/wokwi-hc-sr04. [cit. 2023-12-29].

[25] *Wokwi-analog-joystick Reference*, © 2021-2023. Online. CODEMAGIC LTD. Wokwi. Dostupné z: https://docs.wokwi.com/parts/wokwi-analog-joystick. [cit. 2023-12-29].

[26] , urish, © 2021 - 2023. *Adafruit-mpu6050-plotter.ino*. Online. CODEMAGIC LTD. Wokwi. Dostupné z: https://wokwi.com/projects/305937156771152449. [cit. 2023-12-29].

[27] URISH, © 2021 - 2023. *Mini-piano.ino*. Online. CODEMAGIC LTD. Wokwi. Dostupné z: https://wokwi.com/projects/291958456169005577. [cit. 2023-12-29].

[28] *Prep.ino*, © 2021 - 2023. Online. CODEMAGIC LTD. Wokwi. Dostupné z: https://wokwi.com/projects/385374720900784129. [cit. 2023-12-29].

[29] M., Luboš, © 2023. *Rotační enkodér KY-040*. Online. ECLIPSERA S.R.O. Dratek.cz. Dostupné z: https://navody.dratek.cz/navody-k-produktum/rotacni-enkoder-ky-040.html. [cit. 2023-12-29].

[30] M., Luboš, c 2023. *Arduino WiFi modul NRF24L01*. Online. Dostupné z: https://navody.dratek.cz/navody-k-produktum/arduino-WiFi-modul-nrf24l01.html. [cit. 2023-12-29].

[31] *Enk.ino*, © 2021 - 2023. Online. In: CODEMAGIC LTD. Wokwi. Dostupné z: https://wokwi.com/projects/384845190899820545. [cit. 2023-12-29].

[32] *Rozměry Joysticku*, © 2019-2020. Online. In: Markershop. Dostupné z: https://makershop.ie/image/cache/catalog/p/00506/YXK089-3-1100x1100w.jpg.webp. [cit. 2024-05-26].

[33] *SoundSimulator Youtube návody k Pure Data* [@SoundSimulator]. Online. Dostupné z: https://www.youtube.com/@SoundSimulator. [cit. 2024-05-26].

[34] M., Luboš, © 2024. *Návody Drátek.cz*. Online. Drátek.cz. Dostupné z: https://dratek.cz/arduino/837-rotacni-enkoder.html. [cit. 2024-05-27].

[35] *BlackBox.ai*, 2023. Online. Dostupné z: https://www.blackbox.ai/. [cit. 2024-05-27].

SEZNAM PŘÍLOH

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

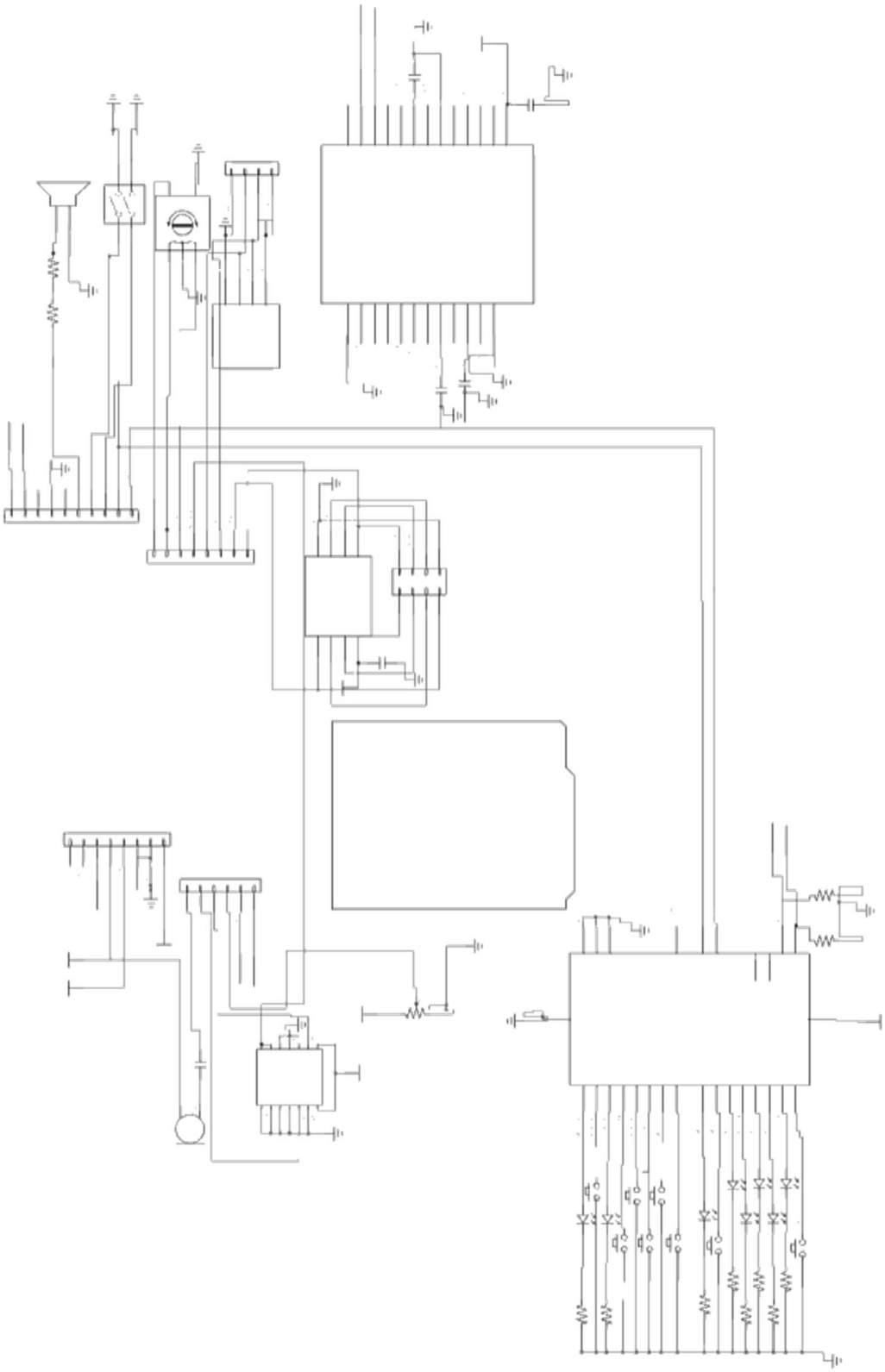
FEKT	Fakulta elektrotechniky a komunikačních technologií
VUT	Vysoké učení technické v Brně
DPS	Deska plošných spojů
IC	Integrated Chip – Integrovaný čip
USB	Univerzální sériová sběrnice
A/D	Analog-digitál převodník

Symboly:

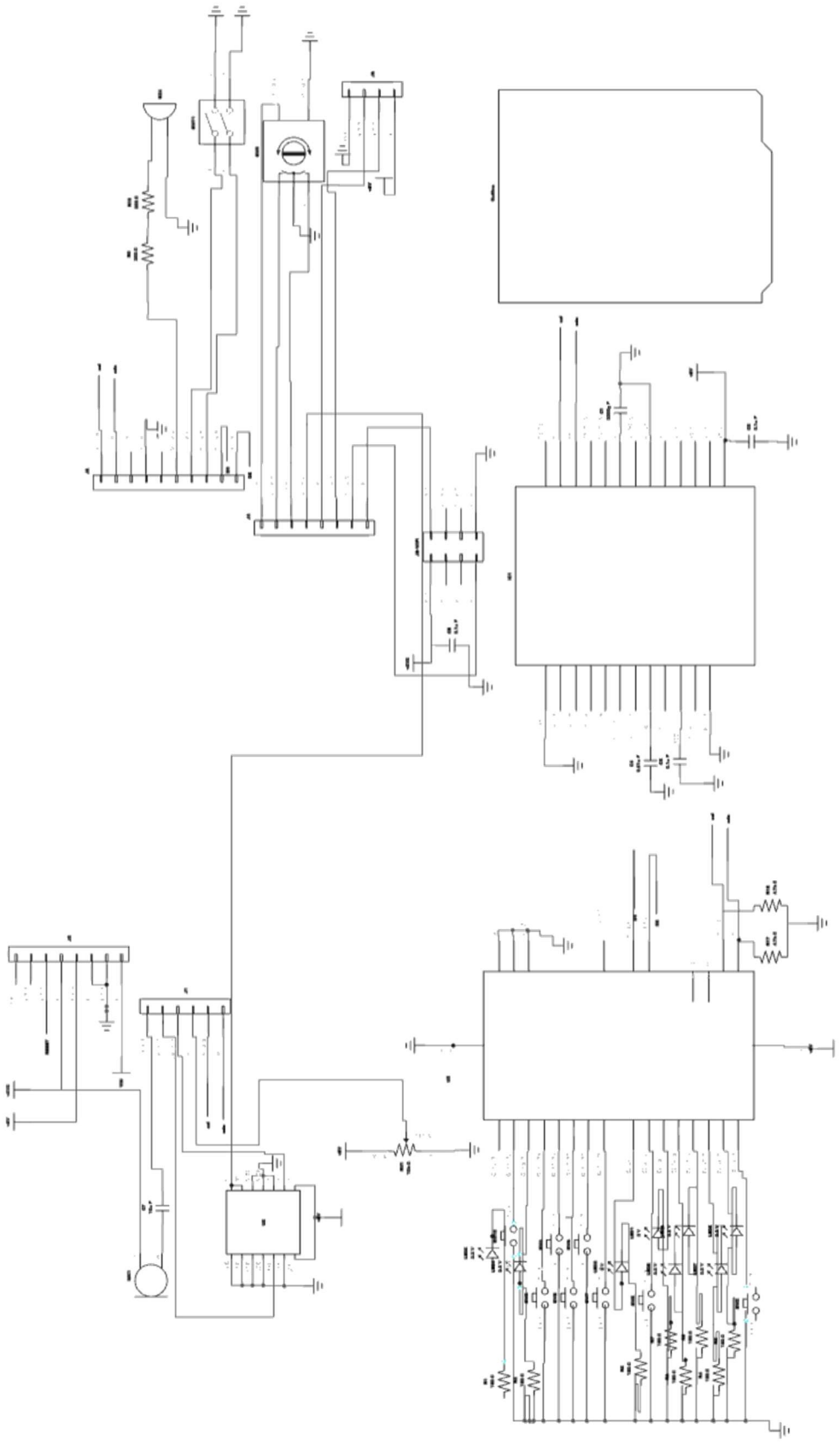
U	napětí	(V)
I	proud	(A)
R	odpor	(Ω)

Příloha A - Návrh DPS

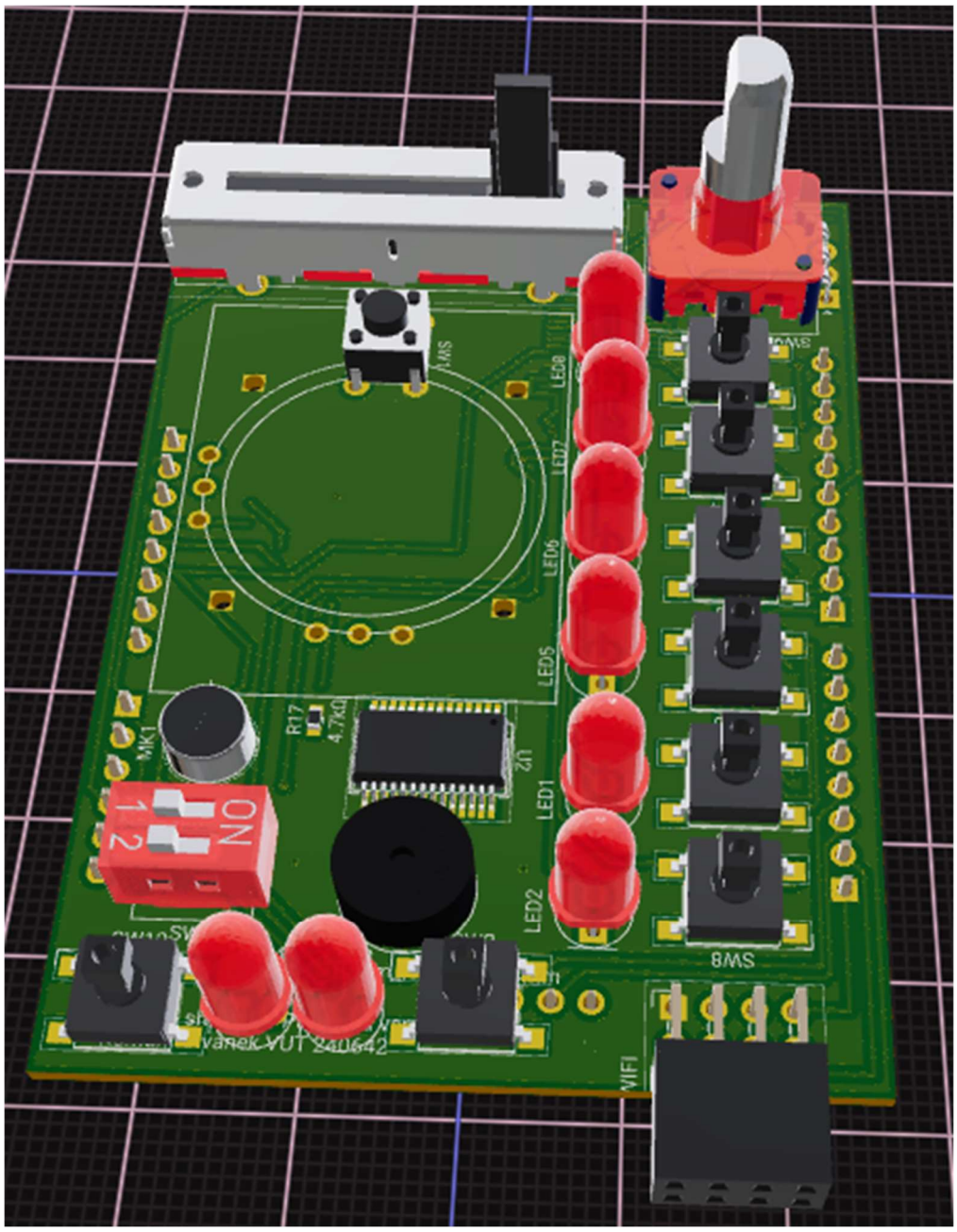
A.1 Schéma zapojení Verze 1.0



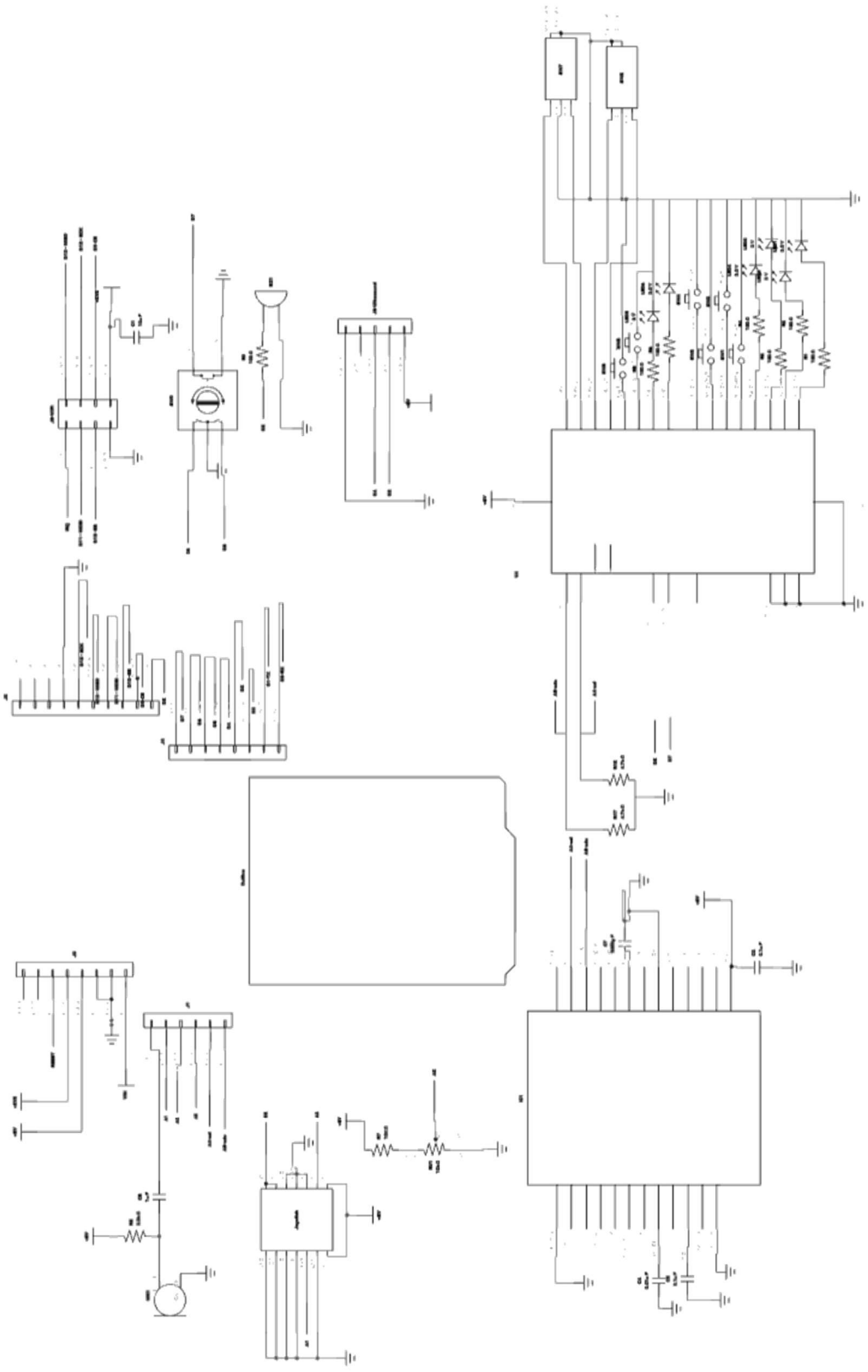
A.2 Schéma zapojení Verze 1.1



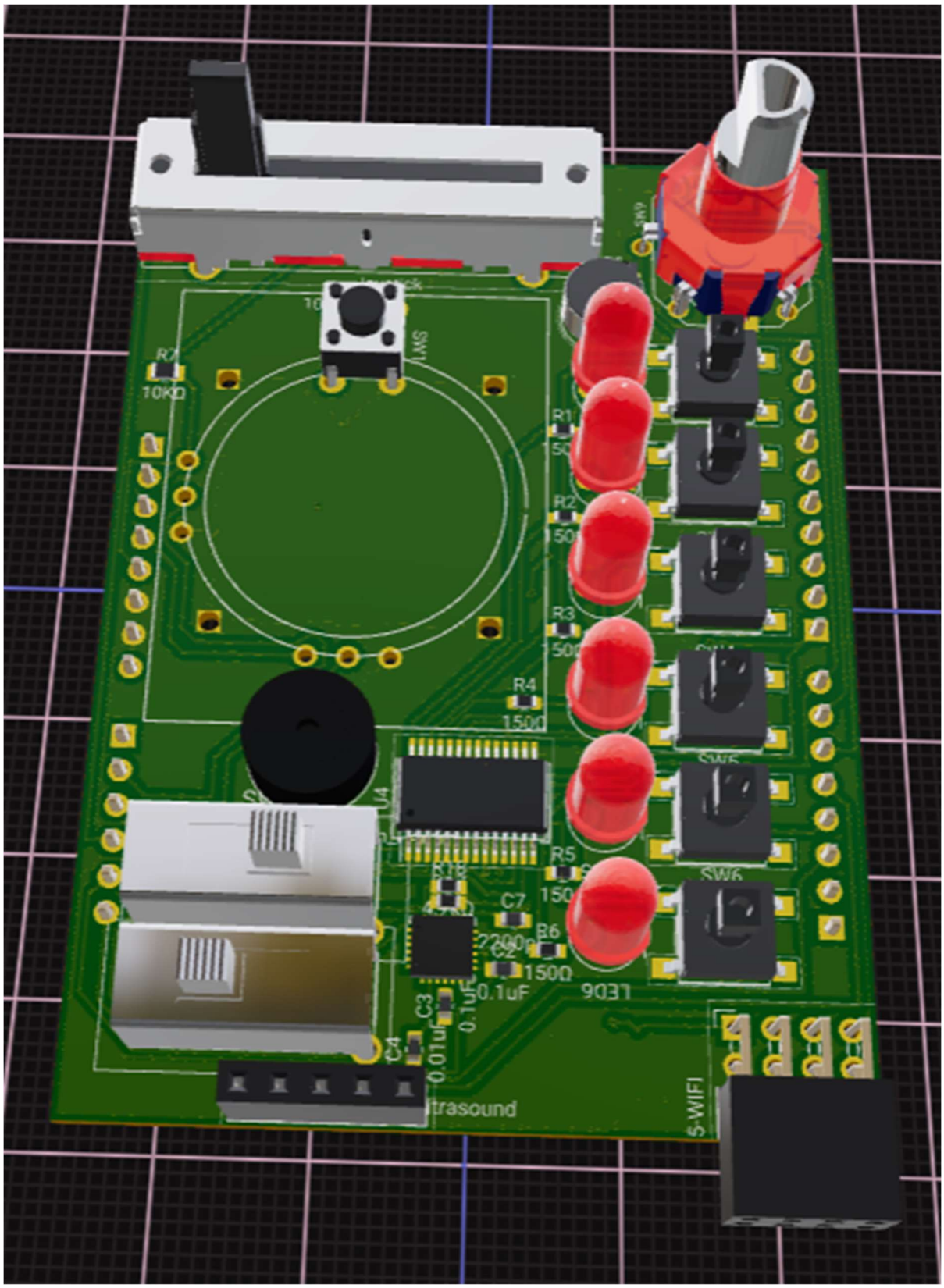
A.3 3D pohled na DPS verze 1.1



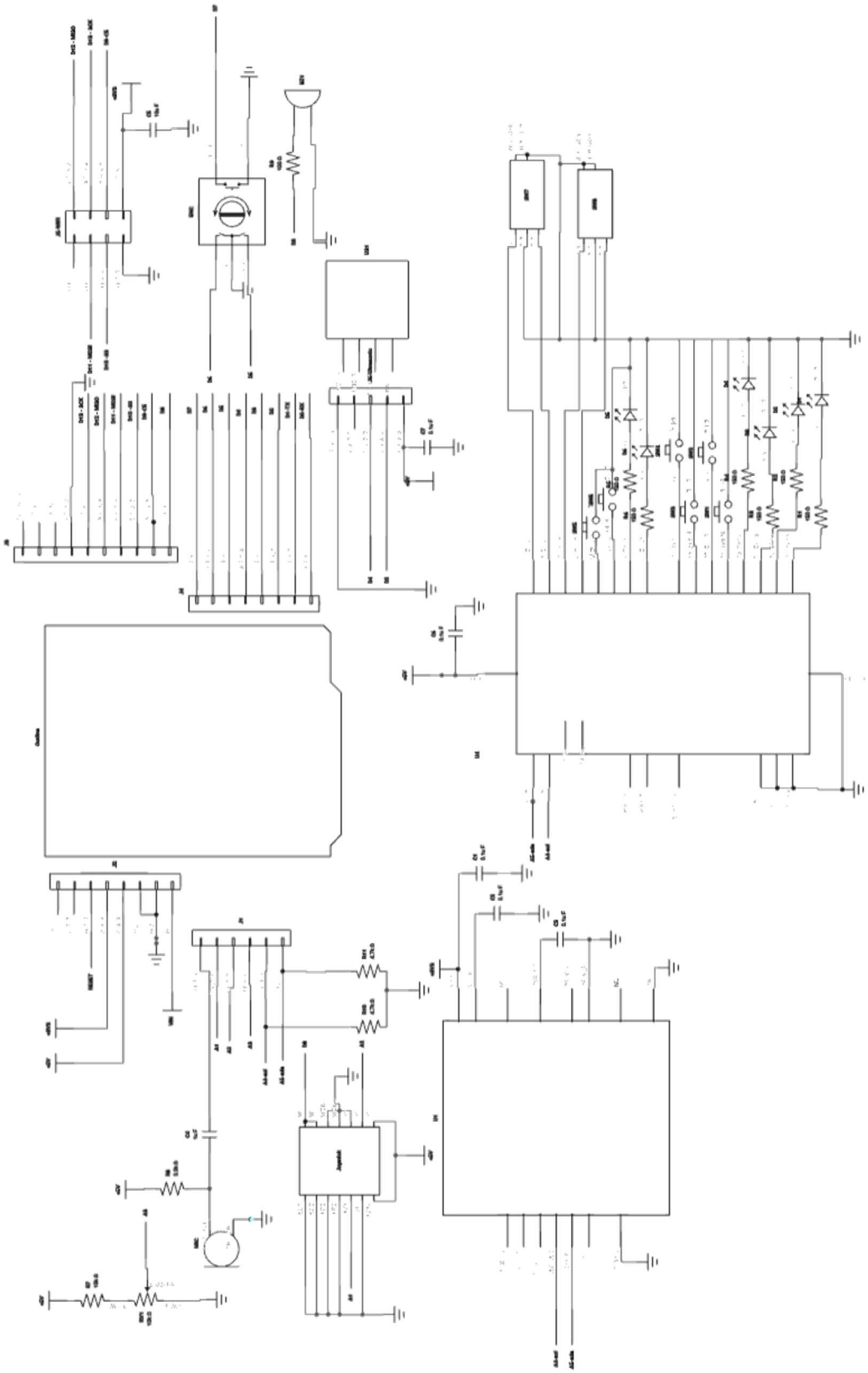
A.4 Schéma zapojení verze 1.2



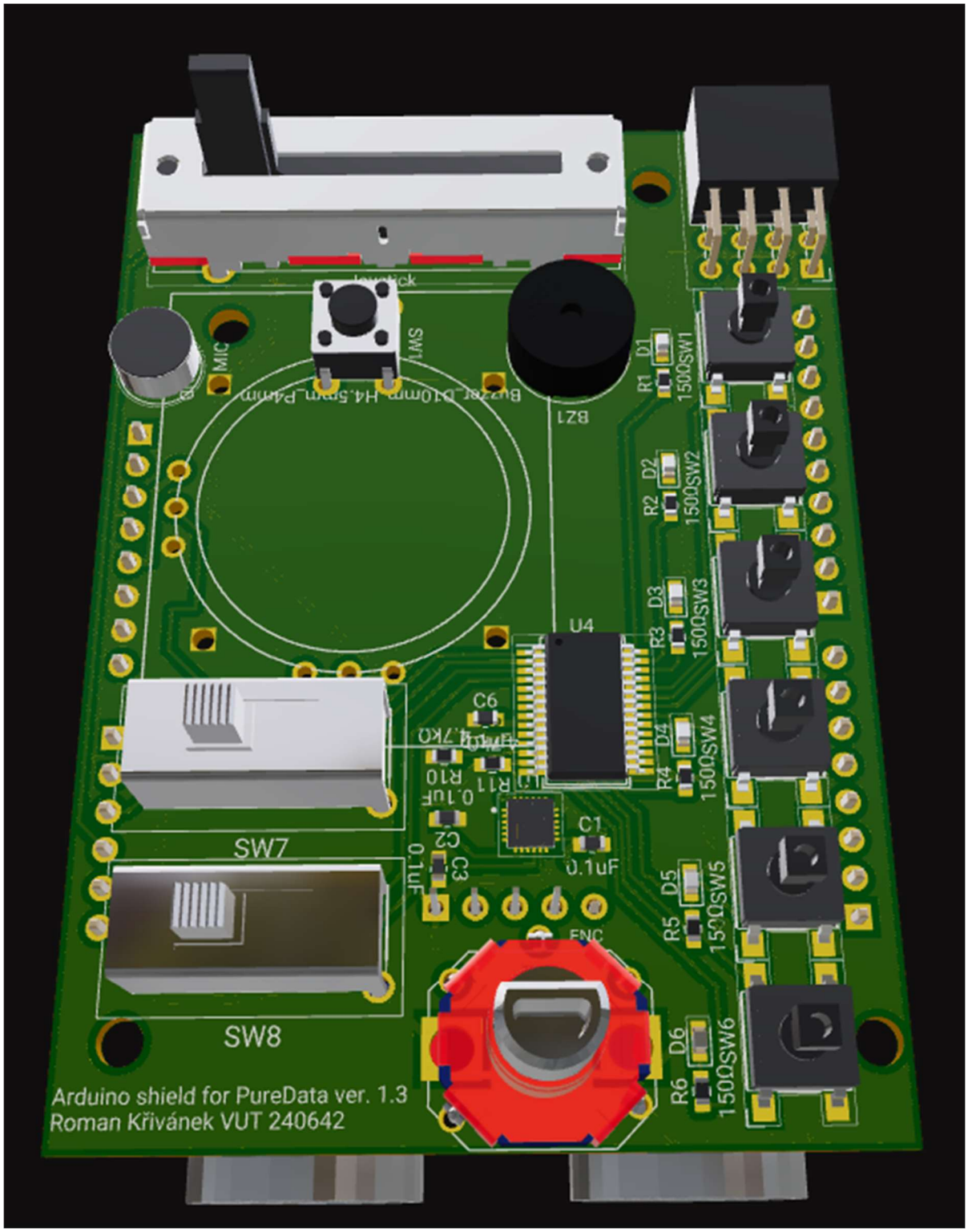
A.5 3D pohled na DPS verze 1.2



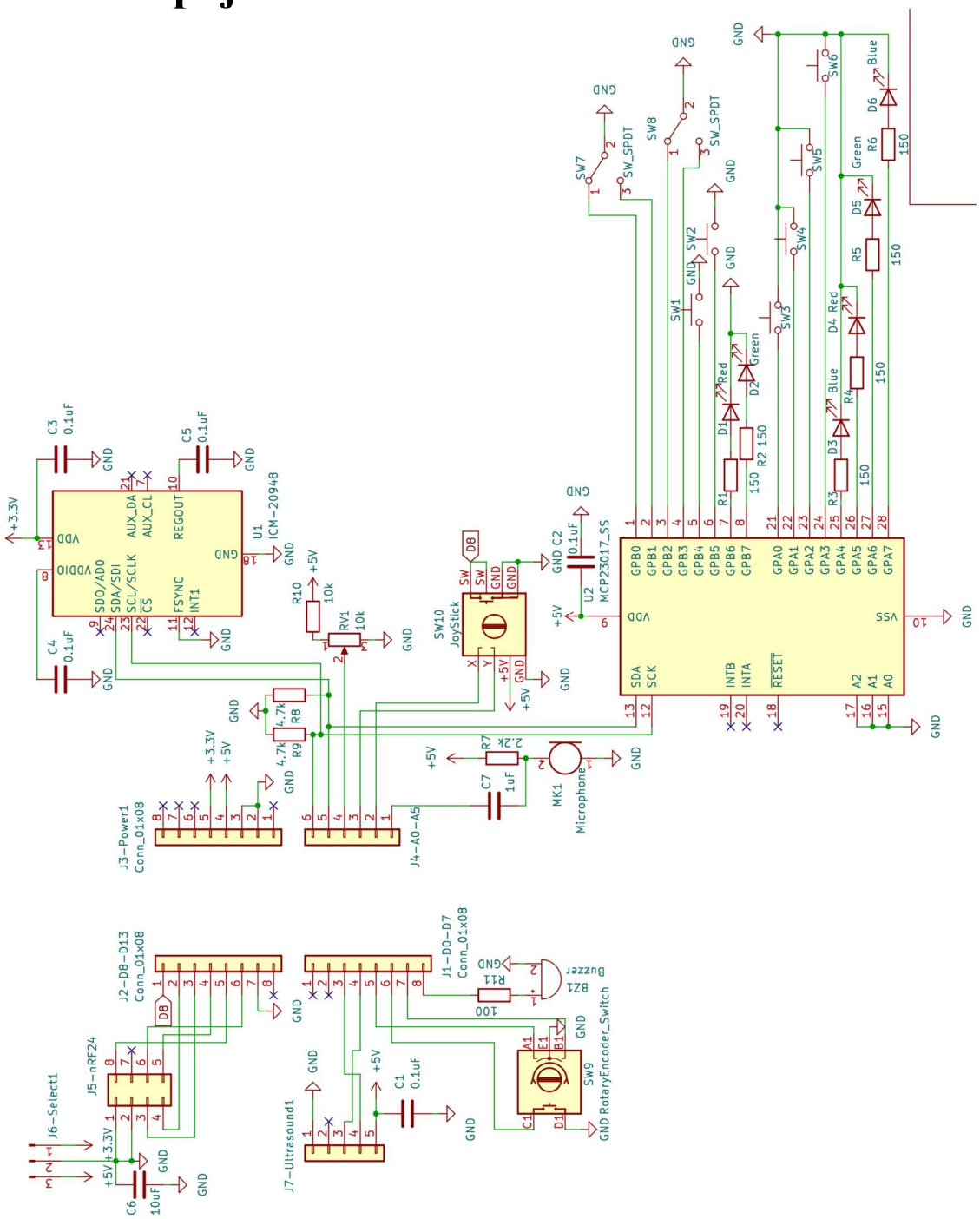
A.6 Schéma zapojení verze 1.3



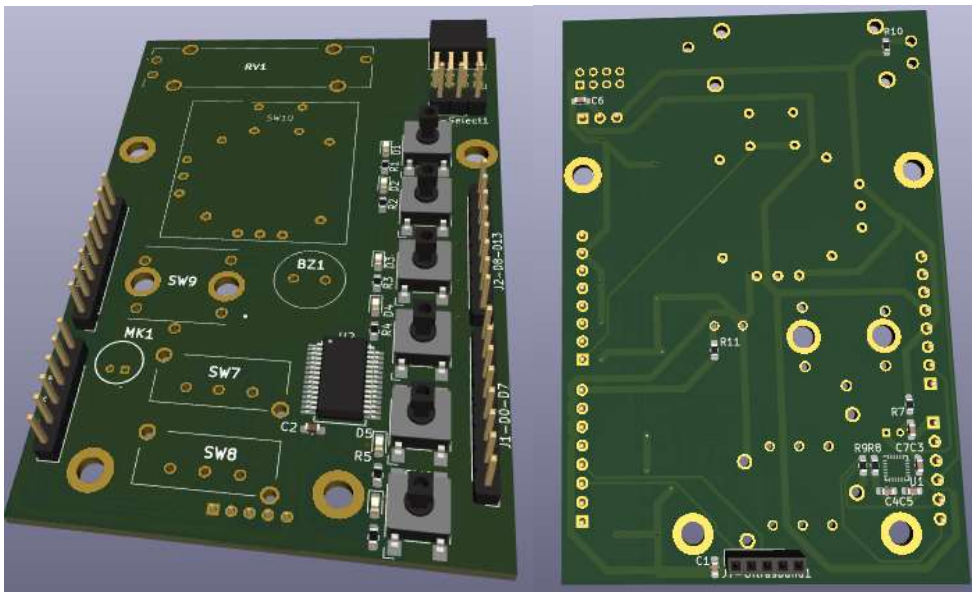
A.7 3D pohled na DPS verze 1.3



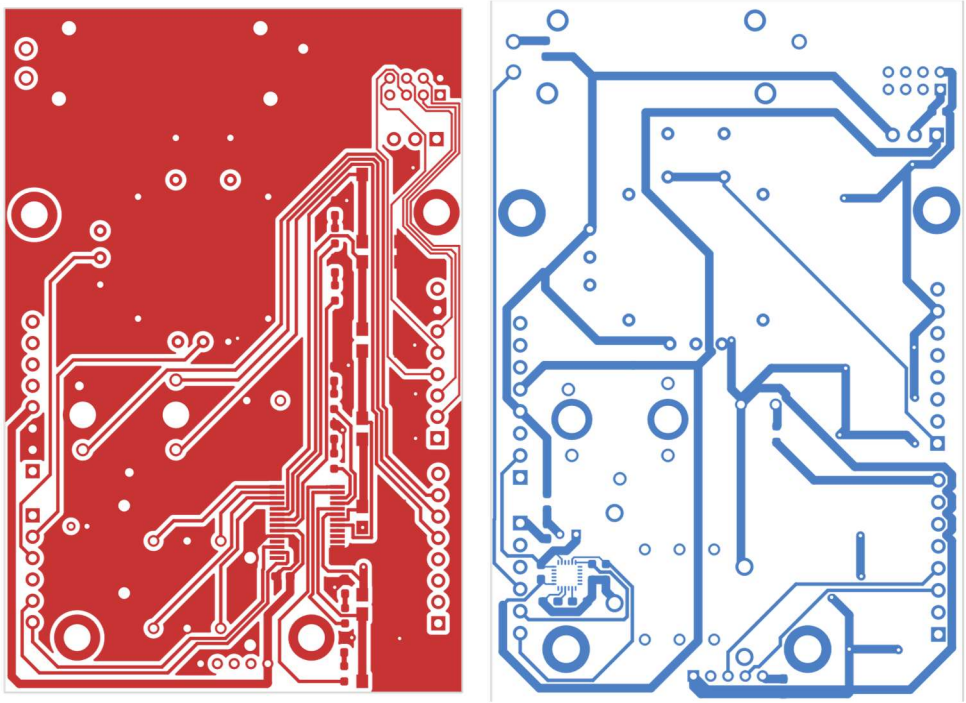
A.8 Verze zapojení 2.0



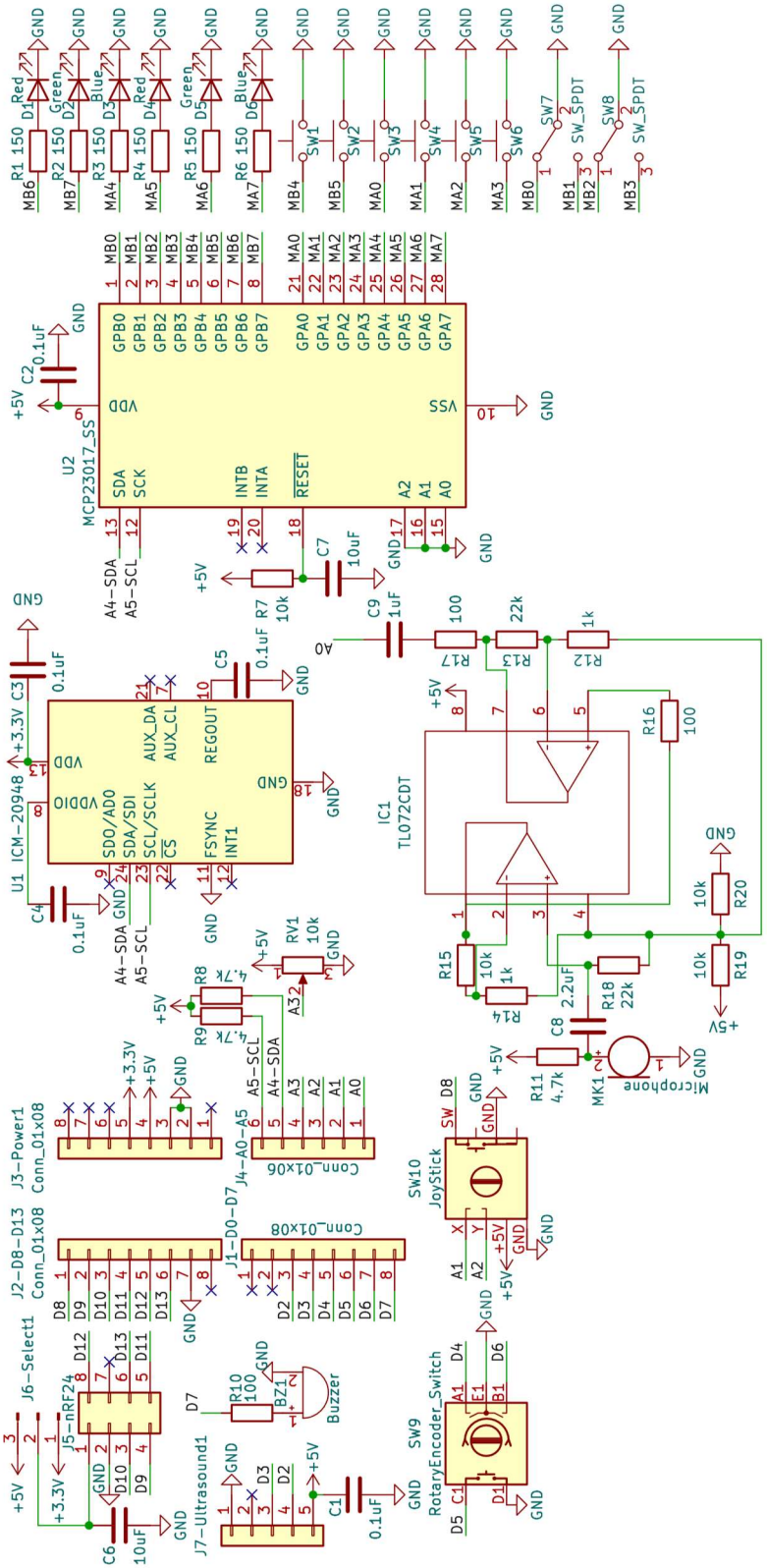
A.9 3D pohled na DPS verze 2.0



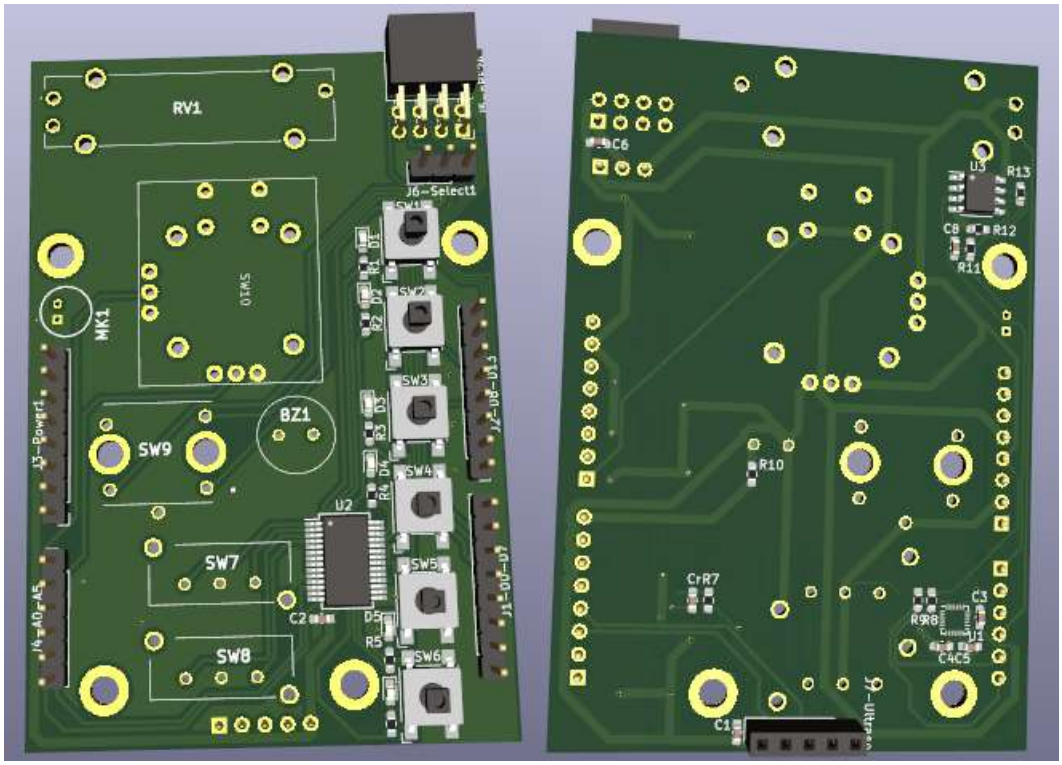
A.10 Osazovací výkres verze 2.0



A.11 Schéma zapojení verze 2.1



A.12 3D pohled na DPS verze 2.1



Příloha B - Arduino kód

A.13 Verze 1

```
/**

 */
#include <Wire.h>
#include <MCP23017.h>
#include <ICM20948_WE.h>
#include <Adafruit_Sensor.h>
#include <Encoder.h>
//#include <NewPing.h>

int potPin  = A0;
int micPin  = A1;
int XPin    = A2;
int YPin    = A3;
// A4 - SDA I2C
// A5 - SCL I2C
#define ICM20948_ADDR 0x68
ICM20948_WE myIMU = ICM20948_WE(ICM20948_ADDR);

#define BuzzPin 7
int measurements = 5; // number of measurements when distance is greater
than 0
// piny pro připojení Trig a Echo z modulu
int pTrig = 2;
int pEcho = 3;
// inicializace proměnných, do kterých se uloží data
long odezva, vzdalenost;

int APin = 5;
int BPin = 6;

int counter = 0;
int aState;
int aLastState;
MCP23017 mcp = MCP23017(0x27);
//pins goes from PA0=0 to PA7=7 and PB0=8 to PB7=15
// MCP23XXX pin LED is attached to
int ledPins[] = {7,6,5,4,14,15}; // Definice pinů pro LEDky

// MCP23XXX pin button is attached to
#define BUTTON1 12
#define BUTTON2 13
#define BUTTON3 0
#define BUTTON4 1
#define BUTTON5 2
#define BUTTON6 3
// MCP23XXX pin Switch is attached to
#define SWITCH1A 8
#define SWITCH1B 9
#define SWITCH2A 10
#define SWITCH2B 11

#define rotarySwitchPin 4
```

```
#define joykSwitchPin 8
// initialization of variales to store data and print them to Serial
monitor to send them to PureData
int state      = 0;
int UltraValue = 0;
void setup() {
  Wire.begin();
  Serial.begin(9600);

  while(!Serial) {}

  if(!myIMU.init()){
    Serial.println("ICM20948 does not respond");
  }
  else{
    Serial.println("ICM20948 is connected");
  }
  mcp.init();
  pinMode(pTrig, OUTPUT);
  pinMode(pEcho, INPUT);
 // configure LED pin for output
  for (int i = 0; i < 6; i++) {
    mcp.pinMode(ledPins[i], OUTPUT); // Nastavení pinů jako výstupy
  }
  // configure button pin for input with pull up
  mcp.pinMode(BUTTON1, INPUT_PULLUP);
  mcp.pinMode(BUTTON2, INPUT_PULLUP);
  mcp.pinMode(BUTTON3, INPUT_PULLUP);
  mcp.pinMode(BUTTON4, INPUT_PULLUP);
  mcp.pinMode(BUTTON5, INPUT_PULLUP);
  mcp.pinMode(BUTTON6, INPUT_PULLUP);
  // configure SWITCH pin for input with pull up
  mcp.pinMode(SWITCH1A, INPUT_PULLUP);
  mcp.pinMode(SWITCH2A, INPUT_PULLUP);
  mcp.pinMode(SWITCH1B, INPUT_PULLUP);
  mcp.pinMode(SWITCH2B, INPUT_PULLUP);
  pinMode(BuzzPin,OUTPUT);
  pinMode(pTrig, OUTPUT);
  pinMode(pEcho, INPUT);
  pinMode(APin,INPUT_PULLUP); // maybe add pull-up ?
  pinMode(BPin,INPUT_PULLUP);
  aState = digitalRead(APin); // read initial state of rotary encoder
}

void loop() {
  // read the value from the sensor:
  Serial.print(" ");
  Serial.print(analogRead(potPin));
  Serial.print(" ");
  Serial.print(analogRead(micPin));
  Serial.print(" ");
  Serial.print(analogRead(XPin));
  Serial.print(" ");
  Serial.print(analogRead(YPin));
  Serial.print(" ");

  // Get new sensor events with the readings
  sensors_event_t a, g, temp; // maybe will be changed if used another
gyro on PCB
```

```
  mpu.getEvent(&a, &g, &temp); // a - acceleration, g - gyro rotation
  Serial.print(a.acceleration.x);
  Serial.print(" ");
  Serial.print(a.acceleration.x);
  Serial.print(" ");
  Serial.print(a.acceleration.x);
  Serial.print(" ");
  //Serial.print(temp.temperature);
  /*
  Serial.print(g.gyro.x);
  Serial.print(" ");
  Serial.print(g.gyro.y);
  Serial.print(" ");
  Serial.print(g.gyro.z);
  Serial.print(" ");
  */
  Serial.print(!mcp.digitalRead(BUTTON1));
  Serial.print(" ");
  Serial.print(!mcp.digitalRead(BUTTON2));
  Serial.print(" ");
  Serial.print(!mcp.digitalRead(BUTTON3));
  Serial.print(" ");
  Serial.print(!mcp.digitalRead(BUTTON4));
  Serial.print(" ");
  Serial.print(!mcp.digitalRead(BUTTON5));
  Serial.print(" ");
  Serial.print(!mcp.digitalRead(BUTTON6));
  Serial.print(" ");
  Serial.print(!mcp.digitalRead(SWITCH1A));
  Serial.print(" ");
  Serial.print(!mcp.digitalRead(SWITCH1B));
  Serial.print(" ");
  Serial.print(!mcp.digitalRead(SWITCH2A));
  Serial.print(" ");
  Serial.print(!mcp.digitalRead(SWITCH2B));
  Serial.print(" ");
  Serial.print(!digitalRead(rotarySwitchPin));
  Serial.print(" ");
  Serial.print(!digitalRead(joykSwitchPin));
  Serial.print(" ");
  aState = digitalRead(APin); // Reads the "current" state of the outputA
  // If the previous and the current state of the outputA are different,
that means a Pulse has occured
  if (aState != aLastState){
     // If the outputB state is different to the outputA state, that
means the encoder is rotating clockwise
    if (digitalRead(BPin) != aState) {
      counter ++;
    } else {
      counter --;
    }
    Serial.print(counter);
    Serial.print(" ");
  }
  aLastState = aState; // Updates the previous state of the outputA with
the current state
  digitalWrite(pTrig, LOW);
  delayMicroseconds(2);
  digitalWrite(pTrig, HIGH);
```

```
  delayMicroseconds(5);
  digitalWrite(pTrig, LOW);
  // pomocí funkce pulseIn získáme následně délku pulzu v mikrosekundách
(us)
  odezva = pulseIn(pEcho, HIGH);
  // přepočet získaného času na vzdálenost v cm
  vzdalenost = odezva / 58.31;
  Serial.print(vzdalenost);
  Serial.println();
  delay(100);
  state=Serial.read(); // got data from Pure data in 0 0 0 0 0 0 format
into x dec number from 0 to 63
  // can split the number bit by bit or calculate the values when there
the button presed
  /*
  //play sound by BuzzPin
  tone(BuzzPin, 1000);
  delay (10);
  noTone(BuzzPin); //stop sound
  */
  //LED brightness cant do (dont have analogwrite on MCP) must use switch
to turn on the LED
    if (state == -1) {
    for (int i = 0; i < 6; i++) {
       digitalWrite(ledPins[i], LOW); // Vypnutí všech LED, pokud je
aktivacniHodnota -1
    }
  } else {
    for (int i = 0; i < 6; i++) {
      int mask = 1 << i; // Vytvoření masky pro i-tou LEDku
      if (state & mask) {
        mcp.digitalWrite(ledPins[i], HIGH); // Rozsvícení LEDky, pokud
je vybrána
      } else {
       mcp.digitalWrite(ledPins[i], LOW); // Zhasnutí LEDky, pokud není
vybrána
      }
    }
  }
  Serial.println();
  delay(100);
}
```

A.14 Verze 2

```
// Arduino to Pure Data - transmitter
#include <Wire.h>
#include <MCP23017.h>
#include <ICM20948_WE.h>
#include <Adafruit_Sensor.h>
#include "pitches.h"
#include <RF24.h>

// Create an RF24 object
RF24 radio(9, 10);
const uint64_t PC=1; // takes the number and selects the right pipe from
address[]
// do more of the address and function to select by PC number at the
```

```
start of the project
uint64_t address[6] = { 0x7878787878LL, // receiver PC=0
                        0xB3B4B5B6F1LL, // transmiter node 1 like PC1
                        0xB3B4B5B6CDLL, // transmiter node 2 like PC2
                        0xB3B4B5B6A3LL, // transmiter node 3 like PC3
                        0xB3B4B5B60FLL, // transmiter node 4 like PC4
                        0xB3B4B5B605LL  // transmiter node 5 like PC5
                        };

#define ICM20948_ADDR 0x68
ICM20948_WE myIMU = ICM20948_WE(ICM20948_ADDR);

int potPin  = A3;
int micPin  = A0;
int XPin    = A1;
int YPin    = A2;
// A4 - SDA I2C
// A5 - SCL I2C
#define BuzzPin 7
int measurements = 5; // number of measurements when distance is greater
than 0
// piny pro připojení Trig a Echo z modulu
int pTrig = 2;
int pEcho = 3;
// inicializace proměnných, do kterých se uloží data
long odezva, vzdalenost;
// Rotary encoder pins
 #define outputA 4
 #define outputB 6

 int counter = 0;
 int aState;
 int aLastState;
MCP23017 mcp = MCP23017(0x20);
//pins goes from PA0=0 to PA7=7 and PB0=8 to PB7=15
// MCP23XXX pin LED is attached to
int ledPins[] = {7,6,5,4,14,15}; // Definice pinů pro LEDky

// Define an array of MCP23017 pin numbers for the buttons and switches
// Structure pins[] B1  B2  B3  B4  B5  B6  SW1A  SW1B  SW2A  SW2B
const int pins[] = {12, 13, 0,  1,  2,  3,    8,    9,   10,   11};

#define rotarySwitchPin 4
#define joykSwitchPin 8
// initialization of variales to store data and print them to Serial
monitor to send them to PureData
int state      = 0;

struct SensorData {
  int potPin;
  int micPin;
  int XPin;
  int YPin;
  float gyrX;
  float gyrY;
  float gyrZ;
  int button1;
  int button2;
  int button3;
```

```
  int button4;
  int button5;
  int button6;
  int switch1A;
  int switch1B;
  int switch2A;
  int switch2B;
  int rotarySwitch;
  int joykSwitch;
  int rotaryPos;
  float ultraValue;
};
SensorData data;

void setup() {
  Wire.begin();
  Serial.begin(9600);

  while(!Serial) {}
   pinMode (outputA,INPUT);
   pinMode (outputB,INPUT);
   // Reads the initial state of the outputA
   aLastState = digitalRead(outputA);

  if(!myIMU.init()){
    Serial.println("ICM20948 does not respond");
  }
  else{
    Serial.println("ICM20948 is connected");
  }
   Serial.println("Position your ICM20948 flat and don't move it -
calibrating...");
  delay(1000);
  myIMU.autoOffsets();
  myIMU.setGyrRange(ICM20948_GYRO_RANGE_250);
  myIMU.setGyrDLPF(ICM20948_DLPF_6);
  Serial.println("Done!");
  pinMode(rotarySwitchPin, INPUT_PULLUP);
  pinMode(joykSwitchPin, INPUT_PULLUP);
  mcp.init();
 // configure LED pin for output
  for (int i = 0; i < 6; i++) {
    mcp.pinMode(ledPins[i], OUTPUT); // Nastavení pinů jako výstupy
  }
   // Set all buttons and switches connected to MCP23017 pins as
input_pullup
  for (int i = 0; i < pins; i++) {
    mcp.pinMode(pins[i], INPUT_PULLUP);
  }
  pinMode(BuzzPin,OUTPUT);
  pinMode(pTrig, OUTPUT);
  pinMode(pEcho, INPUT);
  // Initialize the RF24 object
  radio.begin();
  radio.openWritingPipe(address[PC]); // open writing pipe for PC1
  radio.setPALevel(RF24_PA_MIN);
  radio.setDataRate(RF24_250KBPS);
  radio.setChannel(76);
  radio.setRetries(15, 15);
```

```
  radio.powerUp();
}

void loop() {
  // Save sensor data to data structure
  data.potPin = analogRead(potPin);

  data.micPin = analogRead(micPin);

  data.XPin = analogRead(XPin);
  data.YPin = analogRead(YPin);
  data.joykSwitch = digitalRead(!joykSwitchPin);
  myIMU.readSensor();
  xyzFloat gyr = myIMU.getGyrValues();
  data.gyrX = gyr.x;
  data.gyrY = gyr.y;
  data.gyrZ = gyr.z;

  data.button1=(!mcp.digitalRead(pins[1]));
  data.button2=(!mcp.digitalRead(pins[2]));
  data.button3=(!mcp.digitalRead(pins[3]));
  data.button4=(!mcp.digitalRead(pins[4]));
  data.button5=(!mcp.digitalRead(pins[5]));
  data.button6=(!mcp.digitalRead(pins[6]));
  data.switch1A=(!mcp.digitalRead(pins[7]));
  data.switch1B=(!mcp.digitalRead(pins[8]));
  data.switch2A=(!mcp.digitalRead(pins[9]));
  data.switch2B=(!mcp.digitalRead(pins[10]));
  // ultrasonic sensor
  digitalWrite(pTrig, LOW);
  delayMicroseconds(2);
  digitalWrite(pTrig, HIGH);
  delayMicroseconds(5);
  digitalWrite(pTrig, LOW);
  // pomocí funkce pulseIn získáme následně délku pulzu v mikrosekundách
(us)
  odezva = pulseIn(pEcho, HIGH);
  // přepočet získaného času na vzdálenost v cm
  data.ultraValue = odezva / 58.31;

  // handling the rotary encoder without library
   aState = digitalRead(outputA); // Reads the "current" state of the
outputA
   // If the previous and the current state of the outputA are different,
that means a Pulse has occured
   if (aState != aLastState){
     // If the outputB state is different to the outputA state, that
means the encoder is rotating clockwise
     if (digitalRead(outputB) != aState) {
       counter ++;
     } else {
       counter --;
     }
   }
   data.rotaryPos = counter;
   aLastState = aState; // Updates the previous state of the outputA
with the current state
  data.rotarySwitch=(!digitalRead(rotarySwitchPin));
```

```
  // Send the sensor data via Serial monitor to Pure Data and via nRF24
to second arduino

  radio.write(&data, sizeof(data));
  //break inbetween sending half and receiving half of the program
  delay(1);
  // Receive the LED state from Pure Data
  if (Serial.available() > 0) {
    int ledState = Serial.parseInt();
    if (ledState != -1) {
      for (int i = 0; i < 6; i++) {
        int mask = 1 << i; // Create a mask for the i-th LED
        if (ledState & mask) {
          mcp.digitalWrite(ledPins[i], HIGH); // Turn on the LED if the
corresponding bit is set
        } else {
          mcp.digitalWrite(ledPins[i], LOW); // Turn off the LED if the
corresponding bit is not set
        }
      }
    } else {
      for (int i = 0; i < 6; i++) {
        mcp.digitalWrite(ledPins[i], LOW); // Turn off all LEDs if the
received value is -1
      }
    }
  }
}
```

A.15 Verze 3

```
// Arduino to Pure Data - receiver
#include <RF24.h>
#include <SPI.h>

// Create an RF24 object
RF24 radio(9, 10);
const uint64_t pipe = 0x7878787878LL; // receiver address
const uint64_t pipe1 = 0xB3B4B5B6F1LL; // transmiter node 1 like PC1
const uint64_t pipe2 = 0xB3B4B5B6CDLL; // transmiter node 1 like PC1
// do more of the address and function to select by PC number at the
start of the project
uint64_t address[6] = { 0x7878787878LL, // receiver
                        0xB3B4B5B6F1LL, // transmiter node 1 like PC1
                        0xB3B4B5B6CDLL, // transmiter node 2 like PC2
                        0xB3B4B5B6A3LL, // transmiter node 3 like PC3
                        0xB3B4B5B60FLL, // transmiter node 4 like PC4
                        0xB3B4B5B605LL  // transmiter node 5 like PC5
                        };
struct SensorData {
  int potPin;
  int micPin;
  int XPin;
  int YPin;
  float gyrX;
  float gyrY;
  float gyrZ;
  int button1;
```

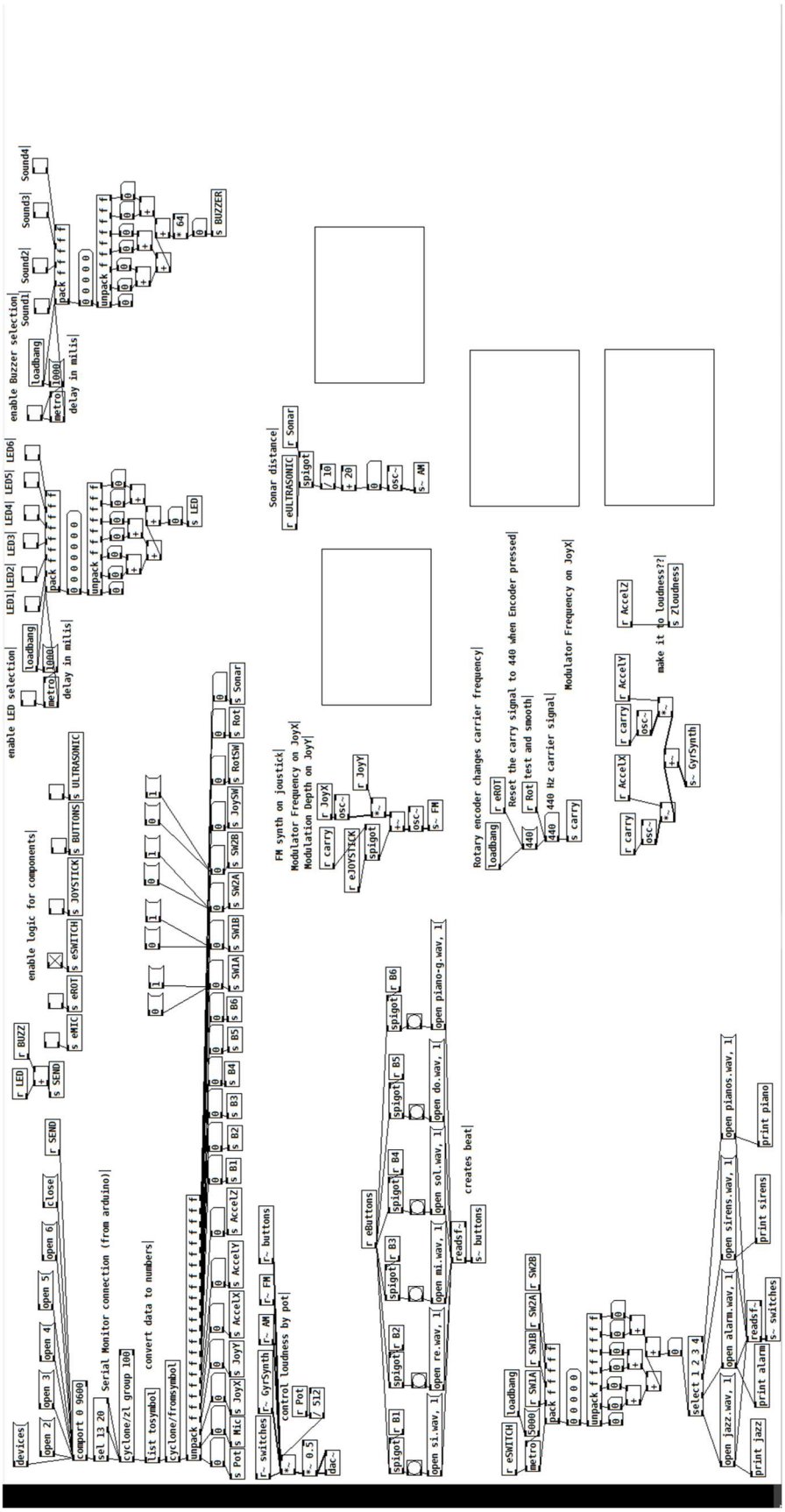
```
  int button2;
  int button3;
  int button4;
  int button5;
  int button6;
  int switch1A;
  int switch1B;
  int switch2A;
  int switch2B;
  int rotarySwitch;
  int joykSwitch;
  int rotaryPos;
  float ultraValue;
};
SensorData data;

void setup() {
  Serial.begin(9600);
  while(!Serial) {}
  // Initialize the RF24 object
  radio.begin();
  Serial.println("RF24 communication begining");
  radio.openWritingPipe(pipe);
  radio.openReadingPipe(1, pipe1); // open pipe 1 with the first address
- PC1
  radio.openReadingPipe(2, pipe2); // open pipe 2 with the second address
- PC2
  radio.setPALevel(RF24_PA_MIN);
  radio.setDataRate(RF24_250KBPS);
  radio.setChannel(76);
  radio.setRetries(15, 15);
  radio.powerUp();
  Serial.println("RF24 communication estamblished");
  radio.startListening();
}

void loop() {
  if (radio.available()) {
    Serial.print(" ");
    radio.read(&data, sizeof(data));
    Serial.print(data.potPin);
    Serial.print(" ");
    Serial.print(data.micPin);
    Serial.print(" ");
    Serial.print(data.XPin);
    Serial.print(" ");
    Serial.print(data.YPin);
    Serial.print(" ");
    Serial.print(data.gyrX, 2);
    Serial.print(" ");
    Serial.print(data.gyrY, 2);
    Serial.print(" ");
    Serial.print(data.gyrZ, 2);
    Serial.print(" ");
    Serial.print(data.button1);
    Serial.print(" ");
    Serial.print(data.button2);
    Serial.print(" ");
    Serial.print(data.button3);
```

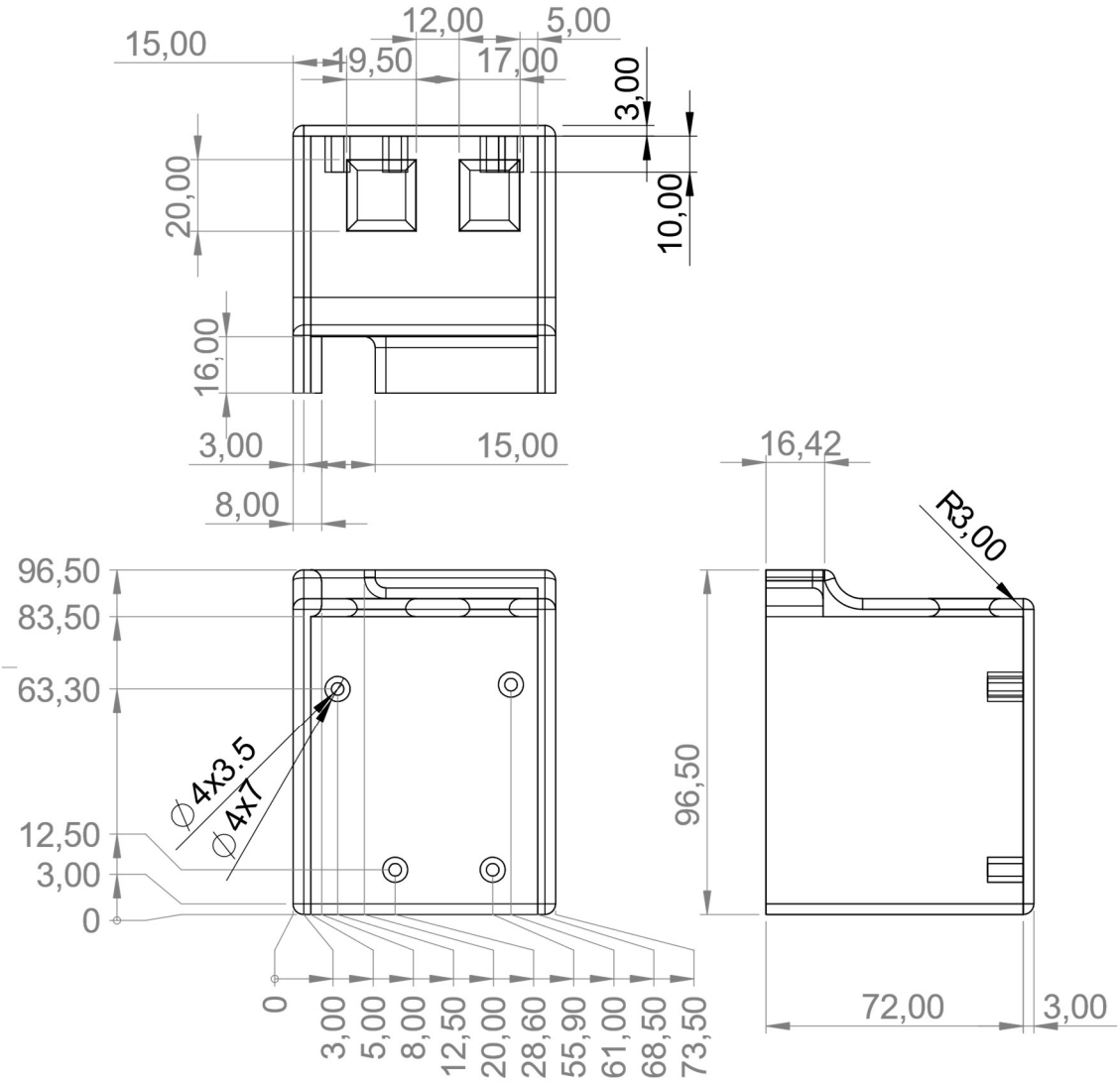
```
    Serial.print(" ");
    Serial.print(data.button4);
    Serial.print(" ");
    Serial.print(data.button5);
    Serial.print(" ");
    Serial.print(data.button6);
    Serial.print(" ");
    Serial.print(data.switch1A);
    Serial.print(" ");
    Serial.print(data.switch1B);
    Serial.print(" ");
    Serial.print(data.switch2A);
    Serial.print(" ");
    Serial.print(data.switch2B);
    Serial.print(" ");
    Serial.print(data.rotarySwitch);
    Serial.print(" ");
    Serial.print(data.joykSwitch);
    Serial.print(" ");
    Serial.print(data.rotaryPos);
    Serial.print(" ");
    Serial.print(data.ultraValue, 2);
    Serial.println(" ");
  }
  delay(1);
  // Read data from Serial port and send to transmitter
  if (Serial.available() > 0) {
    int ledState = Serial.parseInt();
    if (ledState != -1) {
      radio.stopListening();
        radio.openWritingPipe(address[1]);  //  open  writing  pipe  to
transmitter node 1
      radio.write(&ledState, sizeof(ledState));
      radio.startListening();
    }
  }
}
```

Příloha C - Kód v Pure Data

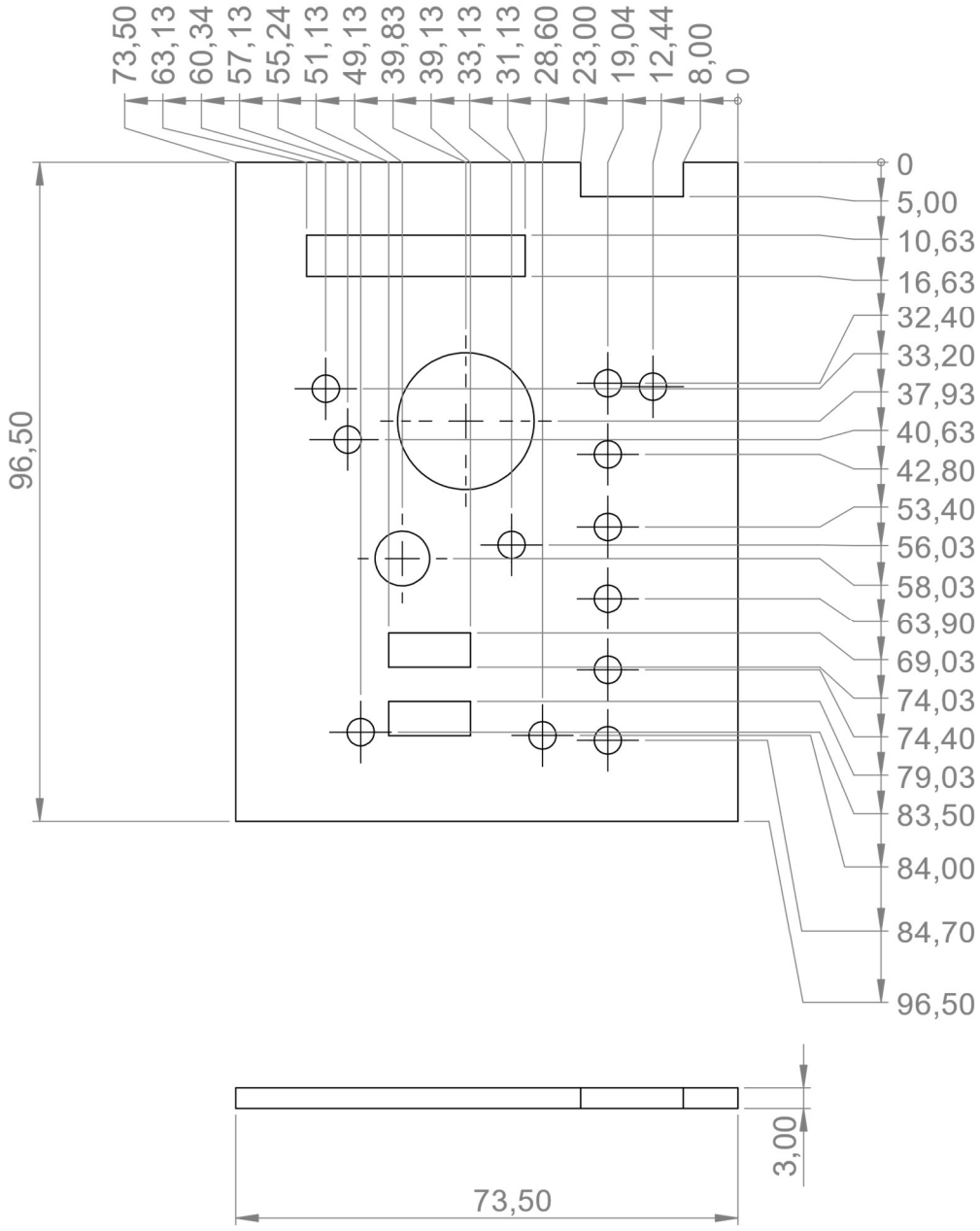


Příloha D - Návrh ochranné krabičky

A.16 Hlavní část krabičky - Spodek



A.17 Vrchní část krabičky



A.18 Zadní část krabičky