UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Systém pro měření impedance

Rauan Zhexenbay

Bakalářská práce
2024

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Rauan Zhexenbay**
Osobní číslo:	**I21102**
Studijní program:	**B0714P060001 Aplikovaná elektrotechnika**
Téma práce:	**Systém pro měření impedance**
Zadávající katedra:	**Katedra elektrotechniky**

Zásady pro vypracování

Cílem práce je navrhnout systém pro měření impedance materiálu umístěného v nádobě (kapalina nebo pevné částice ponořené v kapalině) a to pro různé frekvence. Zařízení bude využíváno k monitorování změn impedance způsobené v důsledku vybuzení elektrickým impulsem. V teoretické části student nastuduje problematiku měření impedancí a jak tato měření realizovat. Na základě poznatků student navrhne systém a ověří jeho funkčnost pomocí měření na vybraných vzorcích.

Rozsah pracovní zprávy: **30**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

[1] MATOUŠEK, David. *Práce s mikrokontroléry ATMEL AT89C2051: [měření, řízení a regulace pomocí několika jednoduchých přípravků]*. MC & praxe. Praha: BEN – technická literatura, 2006. ISBN 80-7300-048-2.
[2] ROMÁNEK, Karel a TOŠOVSKÝ, Petr. *Programovatelný generátor signálu: Programmable signal generator*. Brno: Vysoké učení technické, Fakulta elektrotechniky a komunikačních technologií, 2009.
[3] HAASZ, Vladimír a SEDLÁČEK, Miloš. *Electrical measurements*. Praha: Česká technika – nakladatelství ČVUT, 2006. ISBN 80-01-03375-9.

Vedoucí bakalářské práce: **Ing. Luboš Rejfek, Ph.D.**
Katedra elektrotechniky

Datum zadání bakalářské práce: **15. prosince 2023**
Termín odevzdání bakalářské práce: **10. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

doc. Ing. Jan Pidanič, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 24. ledna 2024

Prohlášení autora

Prohlašuji:
Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.
Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 02. 05. 2024 Rauan Zhexenbay

Poděkování

Chtěl bych poděkovat především svým rodičům za dobrou výchovu, otci, který mi vštípil lásku k technice. Dále bych chtěl poděkovat občanům České republiky za to, že dávají lidem z různých zemí možnost získat slušné vzdělání a uplatnit se v životě. A nakonec bych chtěl poděkovat všem, kteří mi pomohli s mou kvalifikační prací, a to panu Nihadu Huremovi, mému kolegovi Stanislavu Taborovci a mému učiteli/vedoucímu projektu Luboši Rejfkovi.

Anotace

Tato bakalářská práce se zabývá návrhem, konstrukcí a ověřením funkčnosti zařízení pro měření impedancí. Jednotlivé komponenty byly vyhodnocovány jak z hlediska požadovaných parametrů, tak pořizovací ceny. Řízení zařízení je prováděno pomocí rotačního enkodéru a výsledky měření jsou zobrazovány na displeji. Ověření funkčnosti je realizováno porovnáním výsledků s průmyslovým měřičem impedance Bench LCR meter Model 891.

Klíčová slova

Měřič impedance, TFT displej, MCU, DPS, DAC, ADC, SPI

Title

Impedance measurement system

Annotation

This bachelor thesis deals with the design, construction and functional verification of impedance measurement equipment. The individual components were evaluated in terms of both required parameters and purchase price. The control of the device is carried out by a rotary encoder and the measurement results are displayed on the screen. Verification of functionality is performed by comparing the results with an industrial impedance meter Bench LCR meter Model 891.

Keywords

Impedance measurement, TFT display, MCU, PCB, DAC, ADC, SPI

Obsah

Seznam zkratek

OZ	Operační zesilovač (Operational Amplifier)
SPI	Sériové periferní rozhraní (Serial Peripheral Interface)
DAC	Digitálně analogový převodník (Digital to Analog Converter)
ADC	Analogově digitální převodník (Analog to Digital Converter)
MCU	Mikrokontrolér (Microcontroller Unit)
DDS	Generátor přímé syntézy (Direct Digital Synthesizer)
FSK	Frekvenční klíčování (Frequency-Shift Keying)
VAC	Střídavé napětí (Alternating Current)
VDC	Stejnosměrné napětí (Direct Current)
PCB	Deska plošných spojů (Printed Circuit Board)
DPS	Deska plošných spojů (Printed Circuit Board)
GPIO	Obecný vstup/výstup (General Purpose Input/Output)
SWD	Sériové vyzvednutí dat (Serial Wire Debug)

Seznam obrázků

Seznam tabulek

Úvod

Cílem bakalářské práce je návrh a otestování měřícího systému, který bude měřit impedanci testovaných prvků v závislosti na frekvenci.

Motivace pro tvorbu této bakalářské práce je výsledkem několika faktorů. Autor pracuje pro společnost, která potřebovala vyrobit kompaktní modul pro měření impedance. V průběhu vývoje bylo od záměru upuštěno, tak bylo rozhodnuto přepracovat řešení na samostatný měřící systém, který bude měřit impedanci různých prvků. Proto některá technická řešení, použitá v této práci, vyplývají z předchozích iterací DPS. Dalším faktorem je to, že autor projektu má zájem na vytvoření konečného zařízení, tj. zařízení, které bude mít tržní podobu a které lze uplatnit v komerční sféře.

V současnosti se prodejem impedančních měřičů a analyzátorů zabývá mnoho firem, například.: Keysight, Wpiinc, Sourcetronic, Zurich Instruments, Ivium Technologies, Hioki.

Princip měření je u všech přístrojů stejný – měření impedance na základě znalosti elektrického proudu a napětí protékajícího měřeným objektem, jejich přístroje se liší hlavními charakteristikami – frekvenčním rozsahem přístroje, vyjádřený v "Hz", Rozsahem měřených impedancí, vyjádřený v "Ω", třídou přesnosti přístroje vyjádřený v "%".

Společnost BK PRECISION, nabízí přístroj "Model 891 300kHz Bench LCR Meter", (Obrázek 0.1). Tento přístroj byl vybrán pro porovnávací měření s navrhovaným zařízením. Přístroj 891 má 4,3" barevný displej, standardní rozhraní USB (Universal Serial Bus), GPIB (General Purpose Interface Bus) a LAN (Local Area Network) pro dálkové ovládání, možnost uložit/pamatovat si až 100 nastavení včetně 1000 měření a snímků obrazovky. Parametry tohoto zařízení jsou: testovací frekvence od 20Hz do 300kHz, měření odporu/impedance od 10mΩ do 100MΩ, nejlepší přesnost měření impedance 0,05%. Důvodem pro vytvoření nového zařízení místo nákupu tohoto řešení je cena toho přístroje 42 323 Kč.

Obrázek 0.1 – Model 891, 300kHz Bench LCR Meter [1]

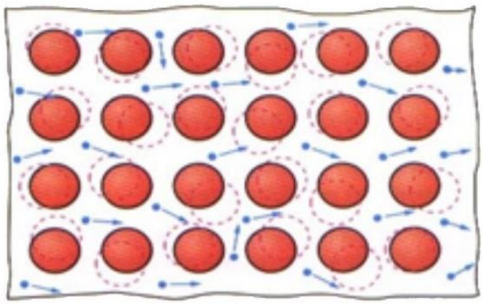
V úvodní části práce jsou popsány teoretické rešerše fyzikálních principů a vlastností elektronických obvodů, které jsou potřebné k návrhu a vytvoření požadovaného měřiče impedancí. V další části je popsán koncept navrhovaného zařízení a popis provedeních průzkumů trhu za účelem vybrání vhodných komponent pro tvorbu zařízení. Následuje návrh finálního zařízení a popis jeho oživování. Dále je pak popsán firmware, který slouží k řízení přístroje. V závěrečné části jsou pak provedena testovací měření za účelem ověření funkčnosti navrženého měřiče impedancí.

1 Teorie

V této části jsou shrnuty potřebné předpoklady pro návrh zařízení, které byly získány na základě rešerší. V první části jsou shrnuty fyzikální principy a ve druhé části schémata dílčích zapojení, která mohou být využita (například OZ jako napěťového sledovače) a jejich vlastnosti.

1.1 Elektrický odpor

„Při průchodu proudu vodičem se v něm pohybují volné elektrony mezi atomy kovu (Obrázek 1.1). Tyto atomy nejsou v klidu, ale kmitají i při běžné teplotě kolem svých klidových poloh, a tím brzdí pohyb volných elektronů. To se projevuje jako elektrický odpor vodiče. Značka elektrického odporu R. Jednotka je Ω (ohm).“ [2]



Obrázek 1.1 – Znázornění elektrického odporu [3]

Elektrický odpor závisí na třech parametrech: materiálu, průřezu a délce vodiče. Vzájemnou závislost těchto parametrů ukazuje následující rovnice 1.1:

$$R = \rho \frac{l}{S} \tag{1.1}$$

kde R – je odpor vodiče [Ω],
ρ – je rezistivita vodiče [Ω·m],
l – je délka vodiče [m],
S – je průřez vodiče [m^2].

1.2 Elektrická impedance

Impedance má stejné aspekty jako odpor, ale impedance je kombinací činného a jalového odporu, přičemž činný odpor zahrnuje odporovou složku, zatímco jalový odpor induktivní a kapacitní složku. Každá z těchto složek ovlivňuje impedanci. Ale pouze jalové složky mění svůj odpor v závislosti na frekvenci. Elektrický odpor se tedy s frekvencí nemění, ale impedance ano. Výpočet impedance je definován rovnici 1.2:

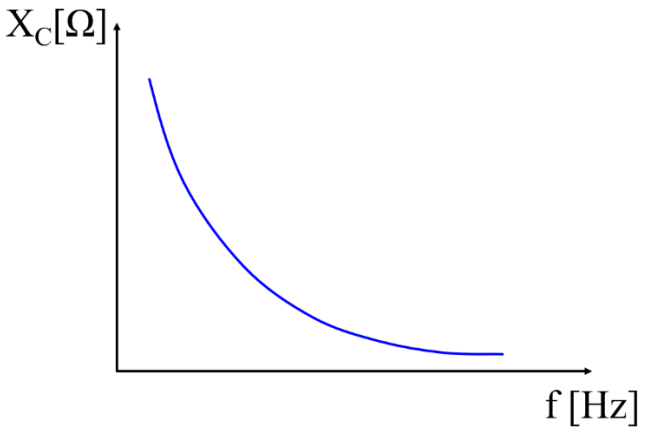
$$\hat{Z} = R + j(XL - XC) \tag{1.2}$$

kde $\hat{Z}$ – je impedance obvodu [Ω],
X_c – je reaktance kapacitní složky [Ω],
X_l – je reaktance indukční složky [Ω].

Reaktance kapacitní složky (kapacitance) je definována rovnicí 1.3. Graf závislosti reaktance kondenzátoru na frekvenci je ukázán na obrázku 1.2.

$$X_c = \frac{1}{2\pi f C} \tag{1.3}$$

kde π – je matematická konstanta [-],
f – je frekvence střídavého proudu [Hz],
C – je kapacita kondenzátoru [F].

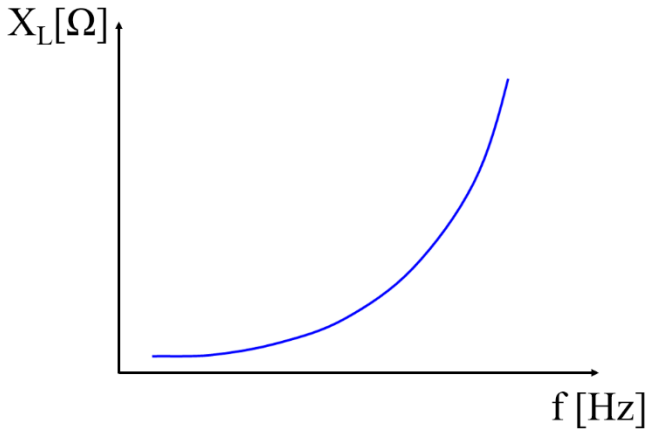


Obrázek 1.2 – Závislost kapacitního odporu na frekvenci, [vlastní obrázek]

Reaktance indukční složky (induktance) je definována rovnicí 1.4. Graf závislosti reaktance kondenzátoru na frekvenci je ukázán na obrázku 1.3.

$$X_l = 2\pi f L \tag{1.4}$$

kde L – je indukčnost [H].



Obrázek 1.3 – Závislost indukčního odporu na frekvenci, [vlastní obrázek]

Kvůli jalovým složkám je tedy obtížné předpovědět impedanci na základě dvou souřadnic, jako je tomu u odporu, který má lineární závislost. Proto neexistuje přístroj, který by dokázal určit impedanci na všech frekvencích jedním měřením, buď změří impedanci na určité frekvenci, nebo dává graf závislosti impedance na frekvenci.

1.3 Druhý Kirchhoffův zákon

Druhý Kirchhoffův zákon neboli Kirchhoffův zákon pro smyčkové proudy vysvětluje vztah mezi napájecími zdroji a jejich spotřebiči/rezistory. Algebraický součet všech napětí v uzavřené smyčce je roven nule, lze vypočítat podle rovnice 1.5. To znamená, že součet napětí na napájecích zdrojích a součet napětí na spotřebičích/rezistorech v uzavřené smyčce jsou stejné velikosti, ale opačného znaménka.

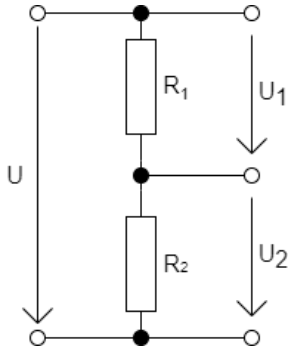
$$\sum U_z = \sum U_s \tag{1.5}$$

kde U_z – je napětí na zdroji [V].
U_s – je napětí na spotřebiči [V].

Tento zákon je základem bakalářské práce, protože pomocí známé hodnoty generovaného napětí, stejně jako napětí na známém rezistoru, lze zjistit, jaké napětí prochází neznámým rezistorem a pomocí této závislosti pak lze vypočítat hodnotu odporu neznámého rezistoru, vzhledem k tomu, že analogový sinusový signál o určité frekvenci bude procházet neznámým odporem/materiálem, odpor, který z toho vyjde, bude impedance.

1.4 Dělič napětí

Druhý Kirchhoffův zákon umožňuje rozdělit napětí pomocí zátěže/rezistoru, dva sériově zapojené rezistory mají na svých koncích různý elektrický potenciál/napětí, schéma zapojení je uvedeno na obrázku 1.4.



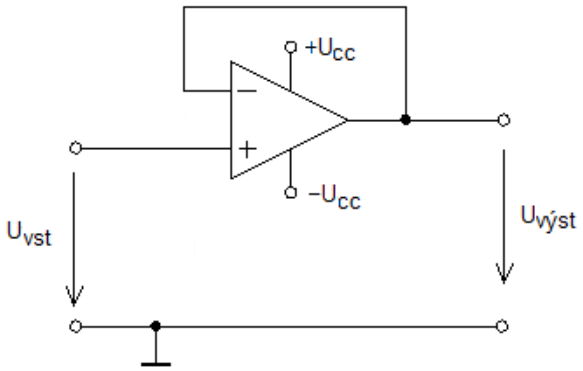
Obrázek 1.4 – Nezatížený dělič napětí, [vlastní obrázek]

Napětí U_2 lze nastavit změnou hodnoty rezistorů nebo zvýšením či snížením napětí zdroje. Podle rovnice 1.6:

$$U_2 = \frac{R_2}{R_1 + R_2} U \tag{1.6}$$

1.5 Sledovač napětí

„Sledovač napětí (Obrázek 1.5) má na výstupu napětí rovné vstupnímu. Vstup má podobně jako u neinvertujícího zesilovače impedanci blížící se nekonečnu. Výstupní impedance je daná vlastnostmi použitého operačního zesilovače a je velmi nízká. Sledovače se používá pro oddělení vysokoimpedančního vstupu a nízkoimpedančního výstupu. Reálně odpovídá vstupní impedanci samotného operačního zesilovače, která je ovšem typicky velmi vysoká (1MΩ až 10TΩ).“ [4]

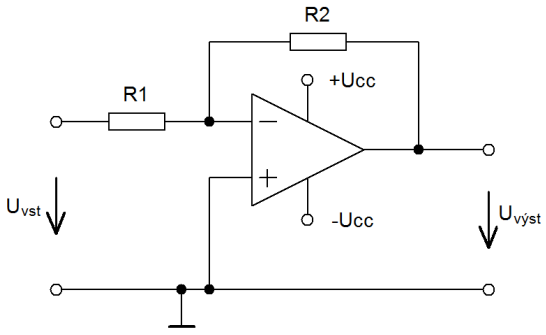


Obrázek 1.5 – Sledovač napětí [4]

1.6 Invertující zesilovač

„Invertující zesilovač (Obrázek 1.6) je jedno z nejpoužívanějších zapojení. Na výstupu se objeví vstupní napětí vynásobené zápornou konstantou (tedy zinvertované). Velikost zesílení je daná poměrem odporů R2 a R1.

Vstupní napětí je přivedeno přes rezistor R1 na invertující vstup OZ (Operační Zesilovač). Ten toto napětí zesílí a na jeho výstupu se tedy objeví zesílené vstupní napětí, avšak s opačnou polaritou. Toto výstupní napětí je přes rezistor R2 rovněž přivedeno na invertující vstup OZ, a protože má opačnou polaritu, zmenšuje napětí na invertujícím vstupu. Protože má OZ velké (ideálně nekonečné) zesílení, ustálí se obvod ve stavu, kdy je na invertujícím vstupu jen velmi malé (ideálně nulové) napětí (tento bod se proto někdy nazývá virtuální zem).“ [5].



Obrázek 1.6 – Invertující zapojení operačního zesilovače [5]

Vzorec pro zjištění výstupního napětí je definována rovnicí 1.7:

$$U_{vyst} = -U_{vst}\frac{R_2}{R_1} \tag{1.7}$$

Vzorec pro zjištění vstupní impedance je definována rovnicí 1.8:

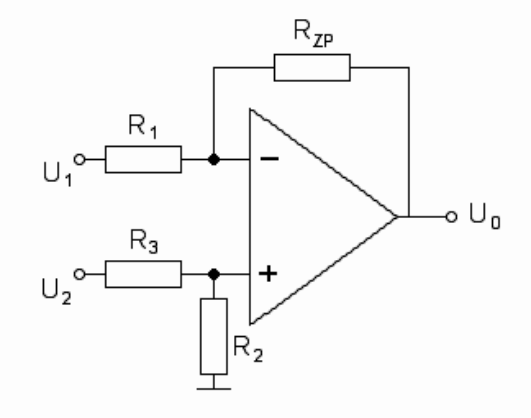
$$Z_{vst} = R_{1_{vst}} \tag{1.8}$$

Vzorec pro zjištění napěťového zesílení je definována rovnicí 1.9:

$$A_U = -\frac{R_2}{R_1} \tag{1.9}$$

1.7 Rozdílový zesilovač

„Toto zapojení (Obrázek 1.7) se nejčastěji používá pro sledování dvou napěťových signálů s velmi málo odlišnými hodnotami napětí, výstupní napětí je pak úměrné rozdílu napětí na vstupech.“ [6]. Používá se také k vyrovnání horizontálního posunu volbou správné hodnoty rezistorů a přivedením konstantního napětí na kladný kanál reference.



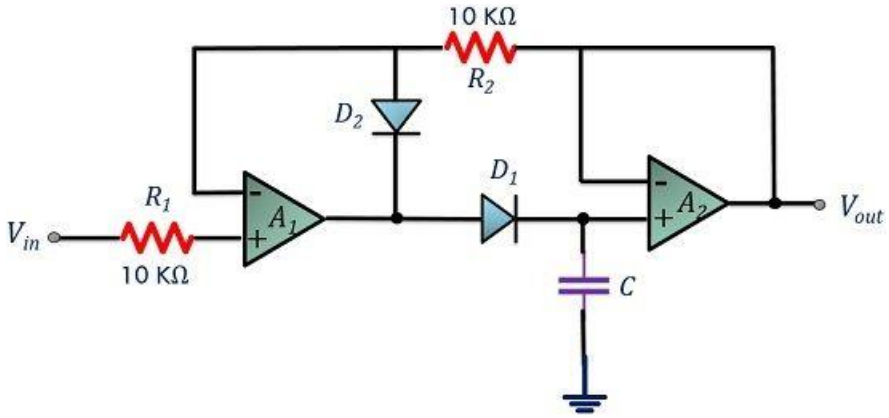
Obrázek 1.7 – Rozdílové zapojení operačního zesilovače [6]

Výstupní napětí je definována rovnicí 1.10:

$$U_o = \frac{R_{zp}}{R_1}(U_2 - U_1) \tag{1.10}$$

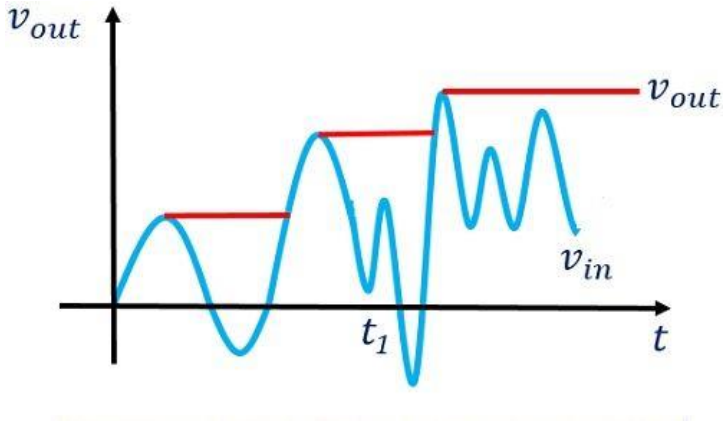
1.8 Detektor špiček

„Obvody detektoru špiček (Obrázek 1.8) se používají k určení špičkové (maximální) hodnoty vstupního signálu. Uchovává špičkovou hodnotu vstupních napětí po nekonečně dlouhou dobu, dokud nenastane stav resetování. Obvod detektoru špiček využívá své vlastnosti sledovat nejvyšší hodnotu vstupního signálu a má schopnost držet maximální hodnotu, a to, dokud nedojde k vybití kondenzátoru.“ [7]



Obrázek 1.8 – Provoz detektoru špiček[7]

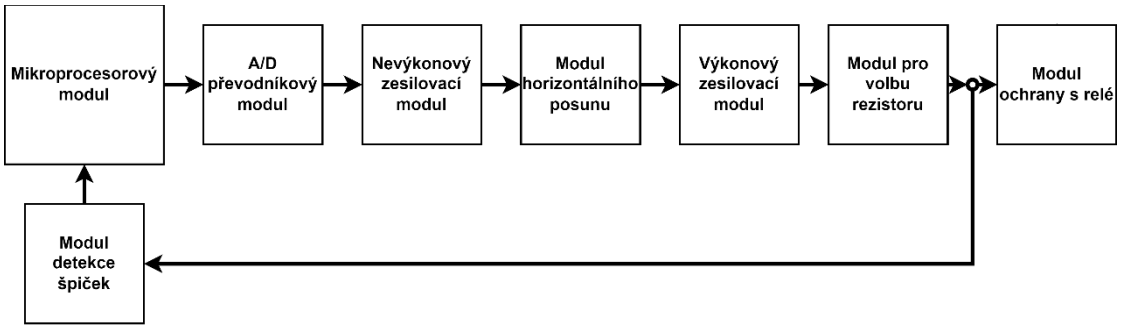
Na základě činnosti detektoru špiček, graf je uveden na obrázku 1.9 lze usoudit, že obvod nepřijímá/neukládá špičky, které jsou menší než uložené, ale jakmile se objeví signál vyšší než uložený, kondenzátor se na něj nabije.



Obrázek 1.9 – Činnost detektoru špiček[7]

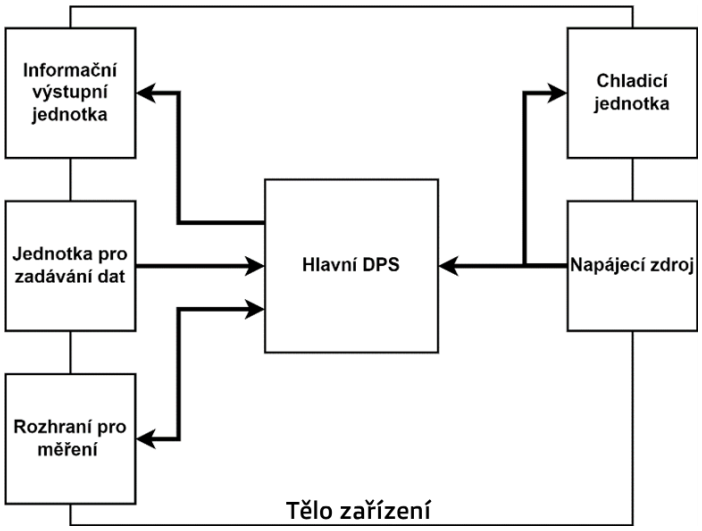
2 Koncepční část

Koncepce DPS (Desky Plošných Spojů), která je základem kvalifikační práce, je postavena na modulech/blocích, které jsou uvedeny na obrázku 2.1. Přičemž každý modul je zodpovědný za jednu, v případě mikroprocesoru za několik úloh, zpracovávat/měnit signály, a bude uvedeno porovnání a výběr komponent se stručným popisem důvodů jejich výběru.



Obrázek 2.1 – Koncepce DPS, [vlastní obrázek]

Následuje koncept na obrázku 2.2, je koncept celého zařízení s DPS uprostřed.



Obrázek 2.2 – Koncepce zařízení, [vlastní obrázek]

2.1 Mikroprocesorový modul

Tento modul bude zodpovědný za řízení průběhu měření, spuštění testu probíhá na povel hlavního mikroprocesoru. Mikroprocesor musí obsahovat SPI (Serial Peripheral Interface), a ADC (Analog to Digital Converter), na základě rozboru byl vybrán MCU (Microcontroller Unit) z řady STM32.

Výběr probíhal mezi MCU STM32F103C8 a STM32G061C8U6, jejichž parametry jsou popsány v tabulce 1. Jelikož oba MCU mají parametry dostatečné pro zamýšlené zařízení. Rozhodujícím faktorem bylo pohodlí práce a dostatek školicích materiálů na internetu.

Tabulka 1 - Porovnání mikroprocesorů STM32, parametry byly převzaty z [8] a [9].

Mikroprocesor	STM32F103C8	STM32G061C8U6
Jádro	Arm32-bit Cortex-M3	Arm32-bit Cortex-M0+
Taktovací frekvence	72 MHz	64 MHz
Kapacita paměti Flash	64 nebo 128 kB	64 kB
Kapacita paměti RAM (Random Access Memory)	20 kB	18 kB
Periferie	ADC, USART, SPI	ADC, USART, SPI
Cena za kus	130,8 Kč	82,3 Kč

Na základě zhodnocení bylo rozhodnuto použít MCU STM32F103C8.

2.2 D/A převodníkový modul

Mikroprocesor STM32F103C8 má vestavěný digitálně analogový převodník, jeho vlastnosti jsou dobré, ale je určen pro jiné úlohy. Pro fungování projektu je nutné generovat fázově a frekvenčně stabilní sinusový signál, což vyžaduje speciální technologii DDS (Direct Digital Synthesis), která se používá právě pro generování sinusových obdélníkových a trojúhelníkových signálů. K tomuto účelu je zapotřebí externí DAC (Digital to Analog Converter). Parametry tří zvažovaných DACu AD9833, AD9834 a AD9858 jsou popsány v tabulce 2.

Tabulka 2 - Srovnání D/A převodníků, parametry byly převzaty z [10], [11] a [12].

DAC	AD9833	AD9834	AD9858
Typ generátoru	DDS		
Funkce	Generování sinusových, obdélníkových a trojúhelníkových signálů		
Rozlišení	10 bitů	10 bitů	12 bitů
Rozhraní	SPI		
Cena za kus	314,23 Kč	387,12 Kč	2 799,60 Kč

AD9858 je velmi sofistikovaný a drahý model, který umí generovat signály FSK (Frequency Shift Keying). AD9833 a AD9834 jsou podobné, ale druhý jmenovaný je o něco pokročilejší, pro účely projektu však stačí základní převodník s DDS. Volba padla na AD9833.

2.3 Nízko výkonový zesilovač

AD9833 produkuje sinusový signál s amplitudou 0,65 V, toto napětí není dostatečné pro plánované zařízení, proto je nutné signál zesílit s minimálním zkreslením, k čemuž bude využito schéma invertujícího zesilovače. Zesilovač bude využívat OZ napájený napětím ±9V a výstupní signál bude použit jako referenční. Zde již neexistují žádná populární řešení, vše záleží na potřebách. Pro aplikaci v zesilovači byly zvažovány 3 varianty operačních zesilovačů, viz tabulka 3.

Tabulka 3 - Srovnání nízko výkonových zesilovačů, převzaty z [13], [14] a [15].

Zesilovač	ADA4098	AD8065	MAX4476AUT
Tranzitní kmitočet	1,05 MHz	145 MHz	10 MHz
Sledovací rychlost	850 mV/us	180 V/us	3 V/us
Výstupní proud	34 mA	35 mA	48 mA
Napájecí napětí – Max	50V	24V	5,5V
Cena za kus	175,92 Kč	165,6 Kč	66,96 Kč

Operační zesilovač ADA4098 má nedostatečnou sledovací rychlost, MAX4476AUT není schopen pracovat při napájení ±9V, tak byl vybrán AD8065.

2.4 Modul horizontálního posunu

Protože převodník D/A vysílá od špičky ke špičce signál 0 až 0,65 V, není fázová čára na nule, což je třeba řešit pomocí modulu horizontálního posunu. Tento modul bude postaven při využití zesilovače AD8065.

2.5 Výkonový zesilovací modul

Na rozdíl od nízko výkonového zesilovače, jehož výstupní proud je 35 mA, což je vhodné pro referenční účely, ale ne pro spotřebitelské použití, je nutné použít zesilovač, který poskytuje vyšší výstupní proud.

Zesilovač, které mají výstupní proud nad 500 mA a tranzitní kmitočet nad 320 MHz, byl vybrán – THS3491. Parametry jsou v tabulce 4.

Tabulka 4 - Charakteristika THS3491, parametry byly převzaty z [16].

Zesilovač	THS3491
Tranzitní kmitočet	900 MHz
Sledovací rychlost	8 kV/us
Výstupní proud	520 mA
Napájecí napětí – Max	32 V
Cena za kus	309,60 Kč

2.6 Modul pro volbu rezistoru

Pro správnou funkci zařízení je nutné zvolit referenční rezistor, pro přepínání je nutný 8 kanálový multiplexor na napětí ±9V. Pro aplikaci byly porovnány 3 různé multiplexory, viz tabulka 5.

Tabulka 5 - Srovnání multiplexorů, parametry byly převzaty z [17], [18] a [19].

Multiplexor	ADG1408	ADG608	MAX306
Počet kanálů	8		
Maximální duální napájecí napětí	16,5 V	5,5V	30 V
Odpor při sepnutí – max	11,2 Ohms	30 Ohms	175 Ohms
Doba sepnutí – max	285 ns	75 ns	600 ns
Cena za kus	464,88 Kč	153,84 Kč	631,44 Kč

ADG608 nebude fungovat při ±9V. Na základě rozboru bylo rozhodnuto použít ADG1408.

2.7 Modul detekce špiček

Detektor špiček bude použit pro měření amplitudy napěťového signálu na referenčním rezistoru. Pro tento modul použijeme stejný operační zesilovač jako pro modul horizontálního posunu a nízko výkonový zesilovací modul, AD8065.

2.8 Modul ochrany s relé

Elementy desky mohou být připojeny k měřenému prvku, ke stejnému prvku jsou připojeny kontakty, mezi kterými je napětí 3000 V. Toto napětí je tvořeno impulsy, aby nedošlo k popálení měřicí desky, budou použita relé, dvě v sérii, první se zapne, když si je uživatel jistý, že nespustí impulsy, druhé se zapne procesor, když spustí měření. Relé jsou napájeny napětím 24 V a mělo by zvládnout více než 3000 V. Je také nutné, aby na desce zabíralo málo místa. Jedinou možností je zde H24-1A69 a jeho varianty H24-1B83, H24-1A83. Jediným rozdílem jsou odporné jímky. Pro tento projekt je vhodný 700ohm, tj. H24-1A69.

2.9 Napájecí zdroj

Napájecí zdroj musí splňovat dva požadavky vstupní napětí 230VAC (Volts Alternating Current) výstupní 24VDC (Volts Direct Current), výkon nejméně 10W. Volba bude záviset spíše na tvarovém faktoru než na elektrických vlastnostech, viz tabulka 6:

Tabulka 6 - Srovnání napájecích zdrojů, parametry byly převzaty z [20], [21] a [22].

Napájecí zdroj	WX-DC24025	K22SA37	MDR-20-24
Výstupní proud	1A		
Výstupní napětí	24V		
Výkon	24W		
Cena za kus	120,00 Kč	279,50 Kč	372,90 Kč

Napájecí zdroj MDR-20-24 byl vybrán pro bezpečné použití.

2.10 Chladicí jednotka

Protože bude používat napětí 24 V, má smysl použít ventilátor pro stejné napětí. Volba ventilátorů v této kvalifikační práci závisela na tvaru, viz tabulka 7.

Tabulka 7 - Srovnání chladičů, parametry byly převzaty z [23], [24] a [25].

Multiplexor	Usongshine 4010	Airen RedWings 40H	Ventilátor 4010
Vstupni proud	0.04A	0,08A	0.1A
Vstupni napeti	24V	12V	24V
Vykon	1W	0.96W	2.4W
RPM	7000	4500	7000
Sum	31dB	18.5dB	27dB
Cena za kus	54,75 Kč	159 Kč	67 Kč

Na první pohled je zřejmé, že “ventilátor 4010” je dobrou volbou se spotřebou 2,4 W. Airen RedWings 40H není vhodný, protože má 12 voltový napájecí zdroj. Pro kvalifikační práci je vhodný i ventilátor Usongshine 4010 se spotřebou 1W.

2.11 Informační výstupní jednotka

Hlavní rozdíly mezi jednotlivými moduly displeje spočívají v úhlopříčce displeje, přítomnosti dotykového vstupu a především v tom, na jakém čipu displej pracuje. V tabulce 2 budou porovnány tři displeje, protože na aliexpress displeje nemají názvy konkrétních modelů a výrobců, objeví se stejný displej pro různé výrobce, srovnávací tabulka bude uvádět prodejce místo modelu, viz tabulka 8.

Tabulka 8 - Porovnání displejových modulů, parametry byly převzaty z [26], [27] a [28].

Displej	Estardyn Official Store	AITEXM ROBOT Official Store	XMKJ Store
Čip	ST7735	ILI9341	ILI9488
Dotykový displej	ne	ne	ano
Diagonála	0.96″	2.8″	4.0″
Komunikační protokol	SPI	SPI	SPI
Barevný displej	ano	ano	ano
Cena za kus	39,65 Kč	113.23 Kč	709.49 Kč

Displej na čipu ST7735 má malou velikost, na čipu ILI9488 je dotykový displej, který není pro tuto kvalifikační práci potřeba. Bude použit displej na čipu ILI9341.

2.12 Jednotka pro zadávání dat

Pro zadávání údajů je zapotřebí zařízení, které je ergonomicky začleněno do konstrukce zařízení, použití numerické klávesnice by bylo neopodstatněné vzhledem k omezenému počtu možných nastavení zařízení. Proto byl vybrán rotační enkodér s tlačítkem pro zadávání dat. Existují dvě varianty, ale všechny jsou založeny na jediném snímači. Tyto varianty budou porovnány v tabulce 2. U prvního není uveden přesný model, proto bude uveden obchod, viz tabulka 9.

Tabulka 9 - Porovnání displejových modulů, parametry byly převzaty z [29] a [30].

Rotační snímač	AITEXM ROBOT Official Store	TZT 360
Napájecí napětí	5V	5V
Pulzní kruh	ne	ne
Provozní teplota	-40+85	-40+85
Cena za kus	17,99 Kč	10.79 Kč

Volba závisí pouze na tvaru desky plošných spojů, pro tuto kvalifikační práci byl zvolen TZT 360.

2.13 Rozhraní pro měření

Měřicí rozhraní se skládá z konektorů, ke kterým budou připojeny vodiče, na jejichž koncích budou použity různé trysky pro uchopení kontaktů měřicího prvku. Protože se konektory liší pouze vzhledem, bude místo tabulky uvedena fotografie modelů, obrázek 2.3. Protože nemají přesný model, budou upřesněny obchody, viz tabulka 10.

Obrázek 2.3 – Typy rozhraní pro měření, obrázky byly převzaty z [31] a [32]

Tabulka 10 - Srovnání rozhraní pro měření, parametry byly převzaty z [31] a [32].

Konektor	SANDOMEY Store	Shenzhen Yihe Technology Co.,LTD store
Cena za kus	8,54 Kč	6.62 Kč

Pro zařízení budou použity konektory od dodavatele Shenzhen Yihe Technology Co.,LTD store.

3 Návrh zařízení

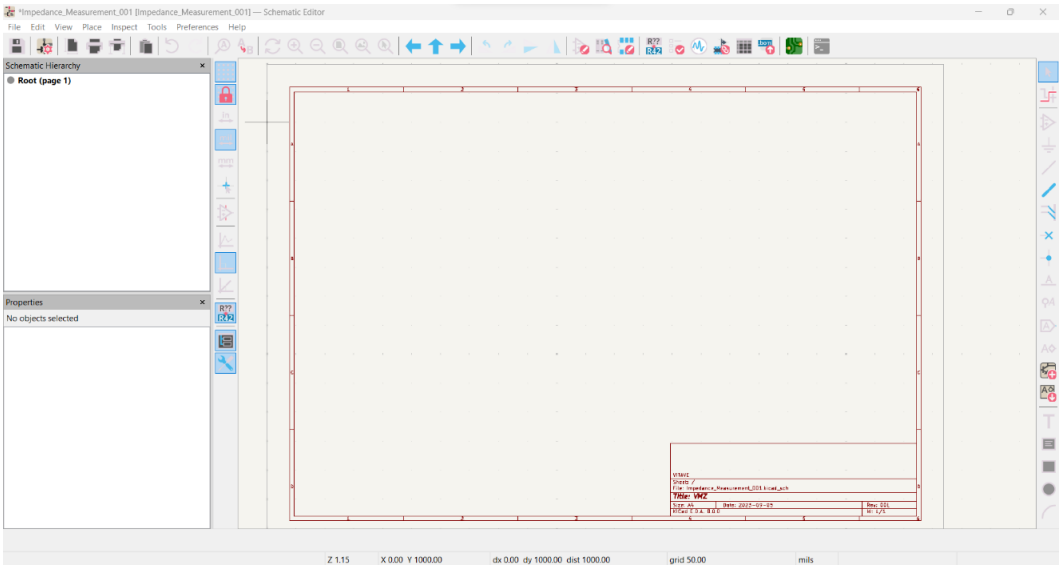
V této kapitole jsou uvedeny informace o práci s KiCadem, aplikaci pro návrh DPS, práci s datovým listem a také jeden z mnoha problémů, se kterými se autor setkal při tvorbě kvalifikační práce.

3.1 Návrh DPS

Návrh zařízení začíná rozvržením desky plošných spojů. Schéma a rozvržení se nakreslí ve volně dostupném programu KiCad, (Logo KiCad je na obrázku 3.1). Tento program dává možnost nejen kreslit schémata a nastavovat rozložení součástek, ale také vytvářet nové součástky a symboly, přidávat na desku plošných spojů fotografie, které lze použít pro značky a nápisy.

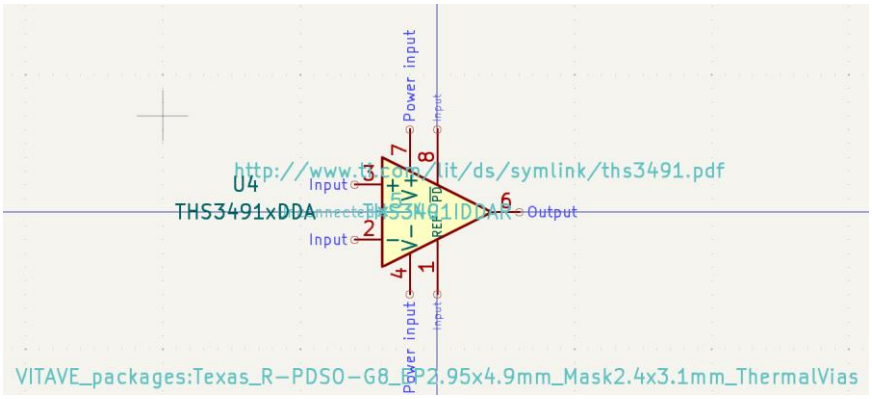
Obrázek 3.1 – Logo KiCad, obrázek byl převzat z [33].

Po vytvoření projektu v programu KiCad se zobrazí prázdný dokument, do kterého lze vkládat elektronické prvky a provádět jejich propojování, tento prázdný dokument je ukázán na obrázku 3.2.



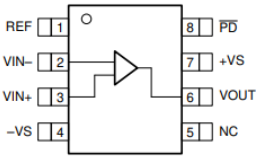
Obrázek 3.2 – Pracovní prostředí v KiCadu pro tvorbu schématu, [vlastní obrázek]

Dále budou vloženy všechny symboly, symboly označuje prvky v diagramu. Ne všechny symboly jsou však v knihovně KiCadu k dispozici, proto je nutné je vytvořit ručně, takže například pro operační zesilovač THS3491 byl vytvořen symbol. Při vytváření symbolu je nutné prostudovat katalogový list součástky. Například operační zesilovač THS3491 jehož symbol je na obrázku 3.3 a výňatek z katalogového listu je na obrázku 3.4.



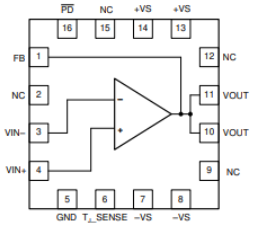
Obrázek 3.3 – Vytvoření symbolu THS3491 v programu KiCad, [vlastní obrázek]

6 Pin Configuration and Functions



NC - no internal connection

Figure 6-1. DDA Package, 8-Pin HSOIC With PowerPAD (Top View)



NC - no internal connection

Figure 6-2. RGT Package, 16-Pin VQFN With Exposed Thermal Pad (Top View)

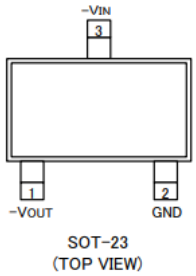
Table 6-1. Pin Functions

PIN (1)			TYPE (2)	DESCRIPTION
NAME	HSOIC	VQFN		
FB	—	1	O	Input side feedback pin
GND	—	5	GND	Ground, $\overline{PD}$ logic reference on the VQFN-16 (RGT) package
NC	5	2, 9, 12, 15	—	No connect (there is no internal connection). Recommended connection to a heat spreading plane, typically GND.
$\overline{PD}$	8	16	I	Amplifier power down: low = amplifier disabled, high (default) = amplifier enabled
REF	1	—	I	$\overline{PD}$ logic reference on the SOIC-8 (DDA) package. Typically connected to GND.
T_{J_SENSE}	—	6	O	Voltage proportional to die temperature
VIN–	2	3	I	Inverting input
VIN+	3	4	I	Noninverting input
VOUT	6	10, 11	O	Amplifier output
–VS	4	7, 8	P	Negative power supply
+VS	7	13, 14	P	Positive power supply
Thermal pad			—	Thermal pad. Electrically isolated from the device. Recommended connection to a heat spreading plane, typically GND.

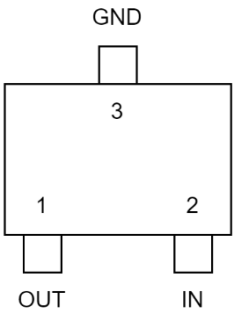
Obrázek 3.4 – Stránka z datového listu THS3491, převzato z [16].

Při vytváření symbolu je nutné nastavit vývody z katalogového listu s vývody na symbolu, jejich číselné pořadí je velmi důležité, protože v budoucnu pro něj bude vytvořen footprint.

Příkladem může být konkrétní chyba, která se stala autorovi kvalifikační práce. Při použití stabilizátoru na -3,3V, byl použit stabilizátor XC6902N331MR, schéma vývodů je uvedeno na obrázcích 3.5 a na obrazku 3.6 je schéma vývodu jiného stabilizátoru.



Obrázek 3.5 – Vývody stabilizátoru XC6902N331MR, údaje převzaté z [34].



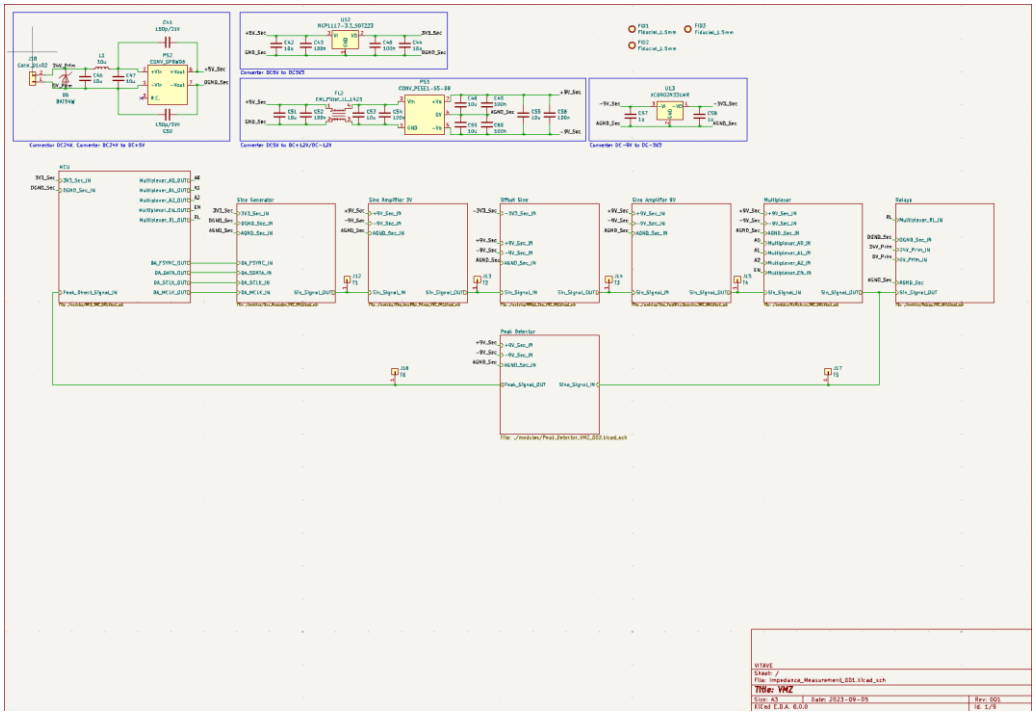
Obrázek 3.6 – Vývody neznámého stabilizátoru, [vlastní obrázek]

Na obou obrázcích je zobrazeno stejné pouzdro, ale různá zapojení. V době práce se stabilizátorem měl autor k dispozici katalogový list jiného stabilizátoru, ale domníval se, že když mají stabilizátory stejné pouzdro, bude i zapojení vývodů stejné. Tato chyba vedla ke dvoudennímu hledání příčiny zkratu za předpokladu, že ke zkratu docházelo pouze při zapnutém napájení.

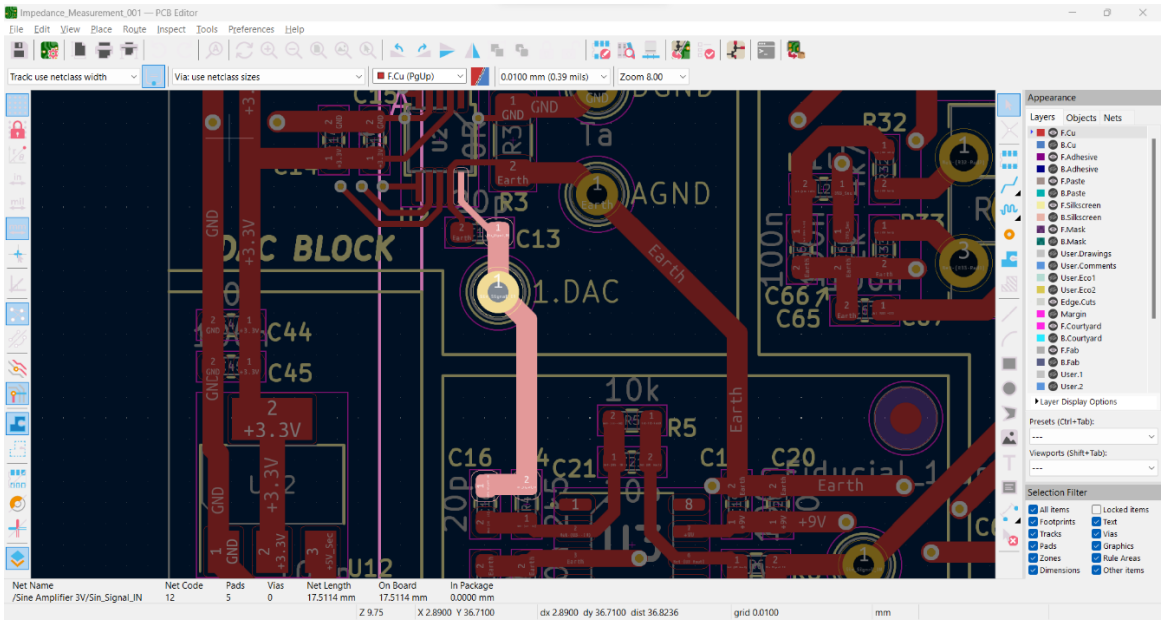
Po zjištění problému bylo nutné jej opravit, protože objednání opravené desky stojí čas a peníze a také nemá vliv na výkon ve srovnání s ruční opravou, je pro takové okamžiky nejlepší kreslit oboustranné nebo jednostranné desky, protože jejich oprava stojí málo úsilí, fotografii po opravě chyby lze nalézt na obrázku 3.7.

Obrázek 3.7 – Ruční oprava chyby na desce plošných spojů, [vlastní obrázek]

Po dokončení schématu, jehož část je ukázána na obrázku 3.8, je nutné uspořádat komponenty na desce. Za tímto účelem je nutné přepnout do aplikace s názvem "PCB Editor" (Printed Circuit Board), obrázek 3.9.

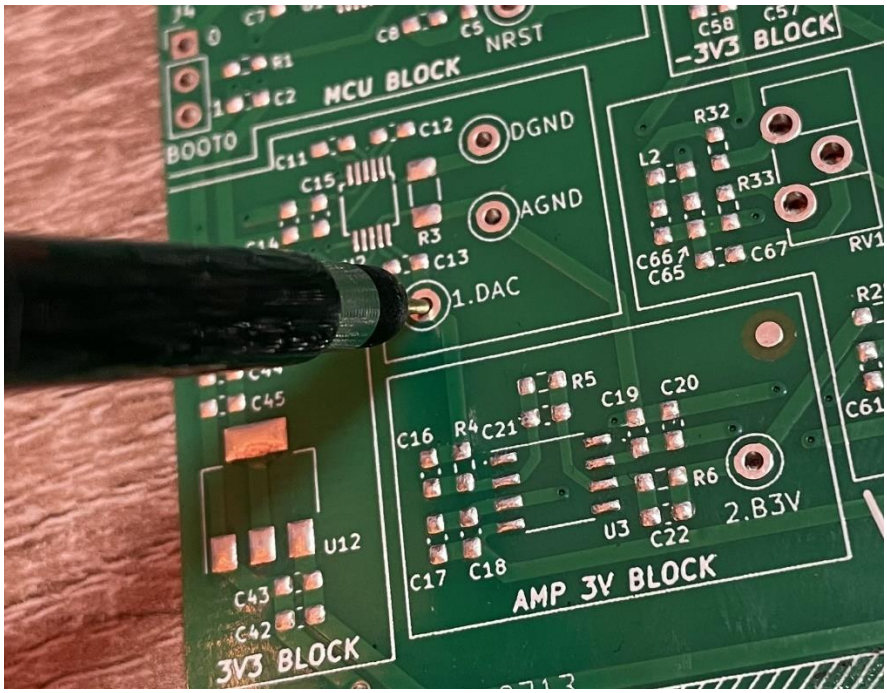


Obrázek 3.8 – Hlavní stránka schémy hotového projektu, [vlastní obrázek]

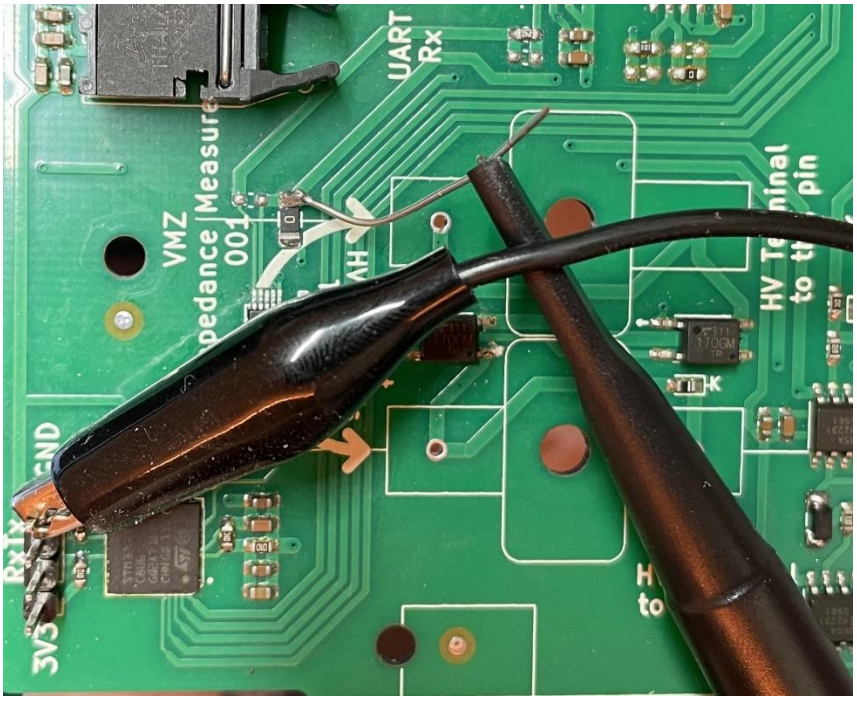


Obrázek 3.9 – Pracovní prostředí v KiCadu pro návrh rozvržení, [vlastní obrázek]

Prvním aspektem při vytváření jakéhokoli experimentálního zařízení je mít na důležitých místech testovací body, v tomto případě jsou těmito testovacími body, které mění signál. Tyto body pomohou mít snadný přístup k měřicímu zařízení. Příklad na obrázku 3.10, ukazuje konec měřicího prvku, který se drží na místě, což značně usnadňuje testování. Pokud by tyto body nebyly, musel by autor připájet vodiče, aby konec měřicího prvku zaháknul, obrázek 3.11, což zabere čas a také připouští možnost zkratu.



Obrázek 3.10 – Příklad použití testovacích bodů na DPS (Deska Plošných Spojů), [vlastní obrázek]



Obrázek 3.11 –Příklad použití testování na DPS, která neobsahuje testovací body, [vlastní obrázek]

Kompletní elektrická schémata potřebná k tvorbě tohoto zařízení se nacházejí v příloze A.

Navržené DPS a fotografie vyrobené a osazené DPS se nacházejí v příloze B.

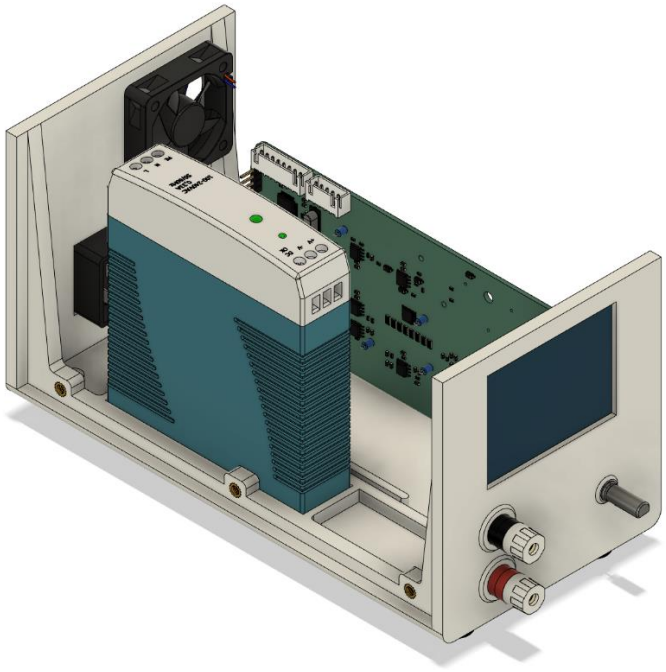
3.2 Návrh těla zařízení

Tělo zařízení bylo vymodelováno ve studentské verzi programu AutoDesk Fusion 360, (Logo AutoDesk Fusion 360 je na obrázek 3.12). Za základ modelu byly vzaty laboratorní napájecí zdroje typu LW-K3010D na obrázku 3.13. Kde je možné převzít koncepci displeje a rozhraní vstupních a výstupních informací, stejně jako chlazení a připojení napájení.

Obrázek 3.12 – Logo Fusion 360, obrázek převzatý z [35]

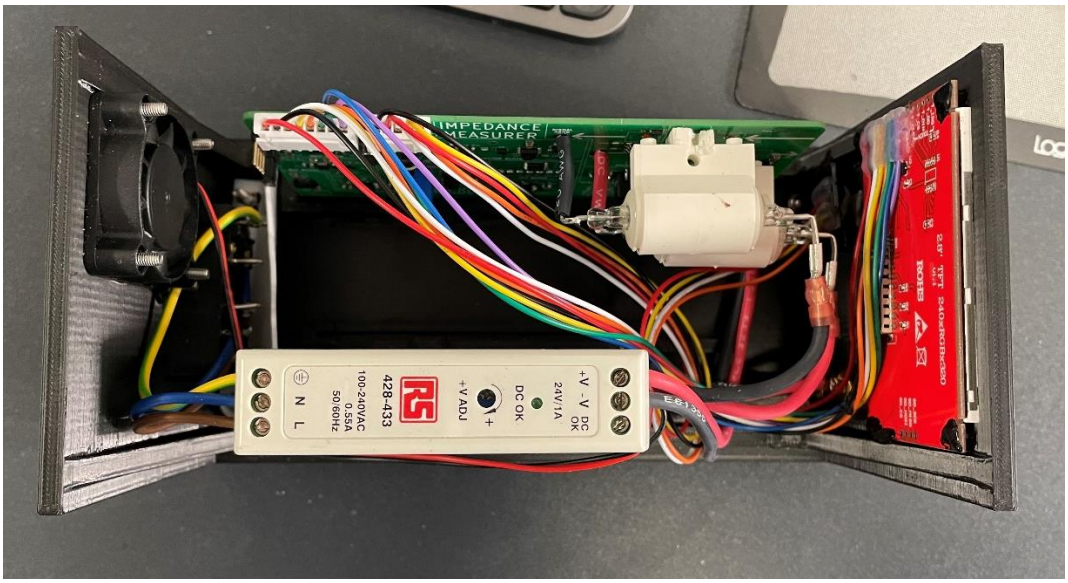
Obrázek 3.13 – Laboratorní napájecí jednotka LW-K3010D, obrázek převzatý z [36]

Důležitým bodem při modelování je najít co nejvíce 3D sestav již hotových zařízení, která budou v projektu použita, protože to výrazně urychlí a zpřesní model. Příklad 24V napájecí jednotky od společnosti MeanWell MDR-20-24, kde byla vytvořena plastová stěna po obvodu jednotky, která drží napájecí jednotku. Také v programu KiCad s jeho pomocí je možné vytvořit soubor .stp, což je soubor 3D modelu, který lze použít pro následné modelování skříně, v tomto případě autor zajistil, aby kabely od 230 V nezasahovaly do uchycení desky a také aby deska vešla do skříně. 3D model zařízení z jedné strany je na obrázku 3.14.



Obrázek 3.14 – 3D model zařízení, [vlastní obrázek]

Při modelování je také nutné mít na paměti, kudy a jak povedou vodiče, a věnovat jim dostatečný prostor. Aby se zabránilo kontaktu 230V s DPS, byl z teflonu vyříznut obdélník o tloušťce 5mm, který byl upevněn šroubem mezi kontakty 230V a samotnou DPS. Fotografie hotového zařízení s pohledem shora je na obrázku 3.15.



Obrázek 3.15 – Hotové zařízení, pohled shora, [vlastní obrázek]

Fotografie vytvořeného zařízení jsou v příloze C.

4 Firmware

Psaní programu začíná potřebnými funkcemi, tedy tím, co bude MCU dělat. Na předchozích stránkách bylo řečeno, s jakými periferiemi a s jakými součástkami na desce plošných spojů bude MCU pracovat. STM32F103 je výrobek firmy STMicroelectronics, pro programování je nutné použít dva programy.

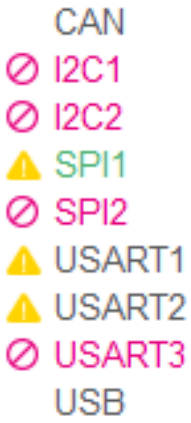
STM32CubeMX (obrázek 4.1) - tento program je určen pro základní konfiguraci mikroprocesoru, přepínání registrů pro změnu funkčnosti pinů GPIO (General Purpose Input Output), připojení externího krystalu a mnoho dalšího.

STM32CubeIDE (obrázek 4.2) - tento program je potřebný pro zápis logiky MCU, umožňuje také ladění MCU v reálném čase pomocí rozhraní SWD (Serial Wire Debug).

Obrázek 4.1 – Logo STM32 CubeMX, bylo převzato z [37].

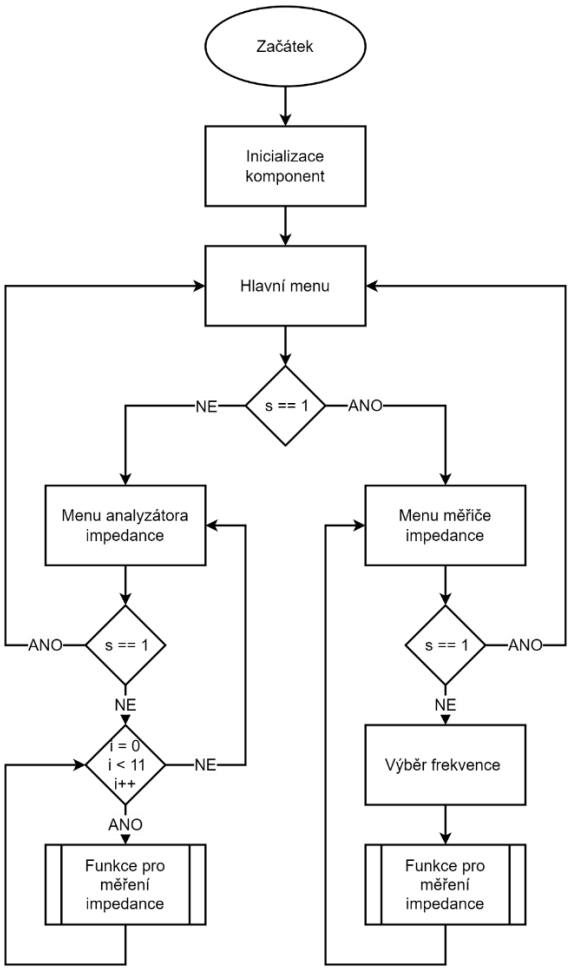
Obrázek 4.2 – Logo STM32 CubeIDE, bylo převzato z [38].

Nejdůležitější věcí, které je třeba věnovat pozornost v tomto programu, je nastavení frekvence procesoru, protože deska má externí krystal na 8 MHz, lze frekvenci procesoru zvýšit na maximální hodnotu 72 MHz. Také je třeba vybrat ty piny, které se nebudou navzájem rušit, program STM32 CubeMX upozorní, jako na obrázku 4.3, uživatele. Jaké může pro jednotlivé funkce využít piny a zamezuje tak spuštění dvou různých funkcí na jednom pinu.



Obrázek 4.3 – Oznámení uživatele v programu STM32 CubeMX, [vlastní obrázek]

Po přiřazení všech pinů, změněně frekvence mikroprocesoru a nakonfigurování všech periférií mikroprocesoru SPI (Serial Peripheral Interface), SWD, ADC. Je potřeba nastavení uložit a přistoupit k psaní programu, u kterého je nejdříve nutné nakreslit přibližný algoritmus, aby bylo snazší jej rozdělit na funkce. Diagram logiky programu naleznete na obrázku 4.4.



Obrázek 4.4 – Algoritmus programu, [vlastní obrázek]

4.1 Programování logiky měření

Základem celého programu je funkce pro měření impedance. Inicializuje proměnné a konstanty, obrázek 4.5.

```
int ImpedanceOnFrequency(int frequency)
{
    const float refVoltage = 3.29;
    const int multiplexerResistance [] = {10, 33, 100, 330, 1000, 3300, 10000, 33000};
    const float targetVoltage [] = {1, 1.98, 2.21, 2.32, 2.29, 2.11, 1.93, 1.42};
    float impedance;
    int adcValue;
    float voltage;
    float prevVoltage = refVoltage;
    int multiplexerCS = 7;
    int multiplexerCount = 7;
```

Obrázek 4.5 – Inicializace proměnných a konstant, [vlastní obrázek]

Poté se převodník spustí na základě zadané frekvence, multiplexor vybere rezistor s nejvyšším odporem, jdou v pořadí, takže multiplexor otevře kanál 7, pokud počítání začíná od nuly a pak se vydá příkaz k zapnutí multiplexoru. Pak se otevře relé, obrázek 4.6.

```
    AD9833_Init(SIN, frequency, 0);

    MultiplexerChannelSelect(multiplexerCS);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, 1); //Enable Multuplexer
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 1); //Open Reles
```

Obrázek 4.6 – Spuštění převodníku, zapnutí a nastavení multiplexoru, sepnutí relé, [vlastní obrázek]

Poté se zapne ADC obrázek 4.7, získaná hodnota se převede na volty a pomocí Kirchhoffova zákona pro napětí můžeme zjistit impedanci neznámého prvku použitím vzorce vzorci 4.1.

$$Z = \frac{RU_z}{U - U_z} \tag{4.1}$$

```
        HAL_ADC_Start(&hadc1);
        HAL_ADC_PollForConversion(&hadc1, 100);
        adcValue = HAL_ADC_GetValue(&hadc1);
        HAL_ADC_Stop(&hadc1);
        voltage = (adcValue/4095.0)*refVoltage;

        impedance = (multiplexerResistance[multiplexerCS] * voltage)/((targetVoltage[multiplexerCS]*2) - voltage);
```

Obrázek 4.7 – Provoz ADC, výpočet impedance, [vlastní obrázek]

Poté se provede kontrola napětí obrázek 4.8, pokud je napětí menší než polovina celkového napětí na rezistoru a měřeném prvku, znamená to, že odpor měřeného prvku je menší než referenční rezistor, pak multiplexor sníží pracovní kanál o jeden stupeň. A pokud je větší, pak se zkontroluje rozdíl přijatých napětí, přičemž ten, který se bude blížit polovině celkového napětí, se použije ke zjištění impedance.

```
353         if(voltage <= targetVoltage[multiplexerCS])
354         {
355             multiplexerCS--;
356             multiplexerCount--;
357             multiplexerCS = (multiplexerCS < 1) ? 1 : multiplexerCS;
358             prevVoltage = voltage;
359             MultiplexerChannelSelect(multiplexerCS);
360         }
361         else
362         {
363             if(multiplexerCS != 7)
364                 if((voltage - targetVoltage[multiplexerCS]) >= (targetVoltage[multiplexerCS] - prevVoltage))
365                 {
366                     multiplexerCS++;
367                     voltage = prevVoltage;
368                 }
369 
370             break;
371         }
```

Obrázek 4.8 – Nalezení správného referenčního rezistoru, [vlastní obrázek]

4.2 Programování logiky rotačního enkodéru

Pro výběr musí uživatel otáčet snímačem, pro tento účel byla vytvořena funkce, která vrací směr otáčení snímače "-1" otáčení doleva, "0" snímač je v klidu, "1" otáčení doprava.

Při psaní této funkce narazil autor na problém, že při otočení snímače o jeden krok se změní hodnota z 0 na 2 místo 1, takže musel funkci doplnit, aby pracovala pouze se sudými čísly.

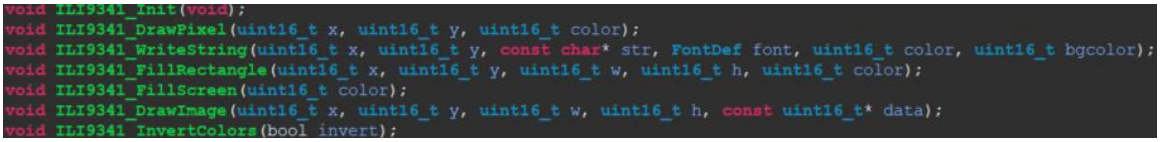
Dále bylo nutné vzít v úvahu, že při otočení enkodéru doprava, přidání hodnoty, která je na hodnotě 255, se jeho hodnota vynuluje na 0, aby si program nemyslel, že enkodér byl otočen doleva, byla vytvořena logika popsaná na obrázku 4.9, která kontroluje, zda enkodér na hodnotě 0 nebyl dříve na hodnotě 250, nebo zcela opačná situace, kdy z hodnoty 0 přejde na 255, pak se kontroluje, zda hodnota 255 nebyla dříve na hodnotě 0. Tímto způsobem lze enkodér otočit pouze doprava a program nebude při každém přechodu z hodnoty 255 na 0 předpokládat, že byl otočen doleva.

```
434     if((encoderPosition % 2 == 0) && (encoderPosition > prevEncoderPosition))
435         encoderDirection = (encoderPosition - prevEncoderPosition > 250) ? -1 : 1;
436     else if((encoderPosition % 2 == 0) && (encoderPosition < prevEncoderPosition))
437         encoderDirection = (prevEncoderPosition - encoderPosition > 250) ? 1 : -1;
438     else
439         encoderDirection = 0;
440     prevEncoderPosition = encoderPosition;
```

Obrázek 4.9 – Nastavení činnosti enkodéru, [vlastní obrázek]

4.3 Programování uživatelského rozhraní

Pro práci s displejem byla použita knihovna ILI9341, která má pouze základní funkce, textový výstup, tvorbu obdélníků a práci s barvami, obrázek 4.10. Tyto funkce stačí pouze pro práci s textovým rozhraním, které je obsaženo v nabídce Hlavní menu, Menu měřiče impedance. Pro zobrazení grafu závislosti impedance na frekvenci však bude potřeba více funkcí. K naštěstí pro autora knihovna ILI9341 má funkci vykreslení jednoho pixelu. Lze ji použít ke kreslení libovolných tvarů.



Obrázek 4.10 – Dostupné funkce v knihovně ILI9341, [vlastní obrázek]

Pro zobrazení křížových čar grafu je snadné použít funkci For(), obrázek 4.11.

```
453     for(int x = 35; x < 287; x++)
454         ILI9341_DrawPixel(x, 200, ILI9341_WHITE);
455     for(int y = 25; y < 201; y++)
456         ILI9341_DrawPixel(35, y, ILI9341_WHITE);
```

Obrázek 4.11 – Zobrazení křížových čar grafu, [vlastní obrázek]

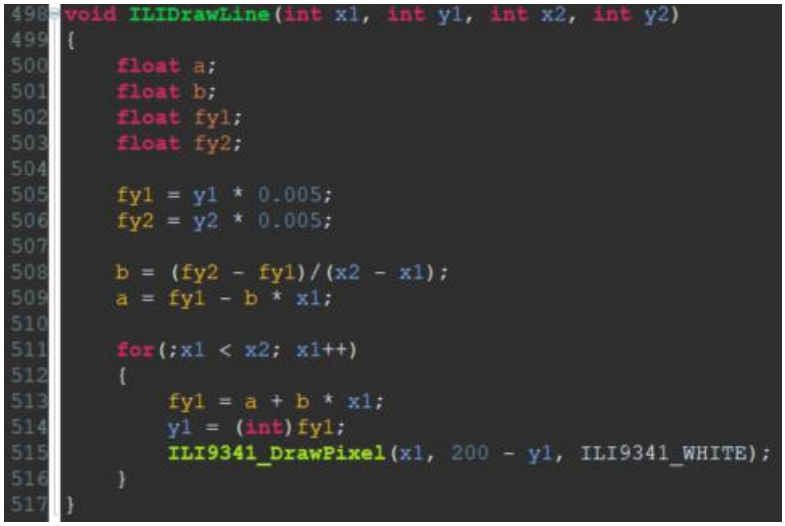
Pro vytvoření grafu závislosti z dostupných údajů o frekvenci a odporu je k dispozici 11 párů. Vezmeme-li první a druhou dvojici, vytvoříme dva body v souřadnicové rovině: A(x_1, y_1) a B(x_2, y_2), kde ‘x’ představuje frekvenci a ‘y’ odpor. Pomocí lineární funkce pak lze určit koeficienty ‘a’ a ‘b’, které umožní nakreslit přímku odpovídající danému vztahu, vzorce 4.2, 4.3 a 4.4. Postup se pak opakuje s body B(x_2, y_2) a C(x_3, y_3) až do posledního bodu.

$$y = a + bx \tag{4.2}$$

$$b = \frac{y_2 - y_1}{x_2 + x_1} \tag{4.3}$$

$$a = y - bx \tag{4.4}$$

Dále byl na základě těchto rovnic napsán program, obrázek 4.12.



```
void ILIDrawLine(int x1, int y1, int x2, int y2)
{
    float a;
    float b;
    float fy1;
    float fy2;

    fy1 = y1 * 0.005;
    fy2 = y2 * 0.005;

    b = (fy2 - fy1)/(x2 - x1);
    a = fy1 - b * x1;

    for(;x1 < x2; x1++)
    {
        fy1 = a + b * x1;
        y1 = (int)fy1;
        ILI9341_DrawPixel(x1, 200 - y1, ILI9341_WHITE);
    }
}
```

Obrázek 4.12 – Odvození křivky závislosti, [vlastní obrázek]

Celek kódu souboru main.c naleznete v příloze D.

5 Testy

Ověření funkce navrženého zařízení bylo provedeno sadou měření. Zapojení přístroje v průběhu měření je na obrázku 5.4. Výsledky jsou uvedeny v tabulkách 11, 12, 13, 14, kde jsou hodnoty změřené komerčním přístrojem (ty budou sloužit jako referenční) a změřené pomocí navrženého měřiče impedancí, v případě kondenzátoru a cívky existuje teoretická hodnota založená na kapacitě nebo indukci a frekvenci. Také byly odvozeny dva grafy pro kondenzátor a cívku, které ukazují srovnání komerčního zařízení, vlastního a teoretického modelu, pro kondenzátor obrázek 5.1, pro cívku obrázek 5.2. Pro přesné měření je možné eliminovat odpor samotných sond jejich jednoduchým spojením. Jejich odpor je 3 ohmy od 20 Hz do 200 kHz a 4 ohmy od 200 kHz do 1MHz. Na základě naměřených hodnot v tabulce lze odvodit teorii, že přesnost klesá při vysokých frekvencích a při vysokých hodnotách odporu měřeného objektu. Takovou zákonitost lze pozorovat na rezistoru 5600 ohmů, je vidět, že při frekvenci 100 kHz je relativní chyba 238 ohmů a při 300 kHz je 1373. Pro měření absolutní chyby byly vzaty pouze výsledky rezistorů a testovacích komor. Průměrná absolutní chyba je 1,94%, maximální absolutní chyba je 33,84% a minimální je 0%. Pro test byla také použita LED červené barvy, tabulka 15, byly provedeny dva testy, při kterých byla katoda a anoda prohozena. Je vidět, že dioda se chová stejně jako kondenzátor, což potvrzuje diodový diagram na obrázku 5.3.

Tabulka 11 - porovnání testovaných výsledků resitorů s modelem 891 a teorií

	Rezistory									
f[Hz]	20	50	100	200	500	1k	10k	100k	300k	1M
	Rezistor na 56Ω									
Z_{A1}[Ω]	56	56	56	56	56	56	56	56	56	-
Z_{A2}[Ω]	57	57	57	57	57	57	57	57	55	58
ΔZ[Ω]	1	1	1	1	1	1	1	1	-1	-
	Rezistor na 560Ω									
Z_{B1}[Ω]	552	552	552	552	552	552	552	552	552	-
Z_{B2}[Ω]	515	520	523	522	519	521	521	518	518	512
ΔZ[Ω]	-37	-32	-29	-30	-33	-31	-31	-34	-34	-
	Rezistor na 5600Ω									
Z_{C1}[Ω]	5573	5573	5573	5573	5573	5573	5573	5573	5430	-
Z_{C2}[Ω]	5624	5592	5578	5592	5601	5520	5574	5335	4057	1695
ΔZ[Ω]	51	19	5	19	28	-53	1	-238	-1373	-

Tabulka 12 - porovnání testovaných výsledků testovacích komor s modelem 891 a teorií

	Testovací komory									
f[Hz]	20	50	100	200	500	1k	10k	100k	300k	1M
	Testovací komora na 50Ω při 300kHz									
Z_{D1}[Ω]	56	55	55	55	55	55	55	55	55	-
Z_{D2}[Ω]	55	55	54	54	54	54	53	54	54	55
ΔZ[Ω]	-1	0	-1	-1	-1	-1	-2	-1	-1	-
	Testovací komora na 200Ω při 300kHz									
Z_{E1}[Ω]	201	199	198	197	197	196	196	196	196	-
Z_{E2}[Ω]	205	206	202	200	201	201	201	200	200	200
ΔZ[Ω]	4	7	4	3	4	5	5	4	4	-

Tabulka 13 - porovnání testovaných výsledků keramických kondenzátorů s modelem 891 a teorií

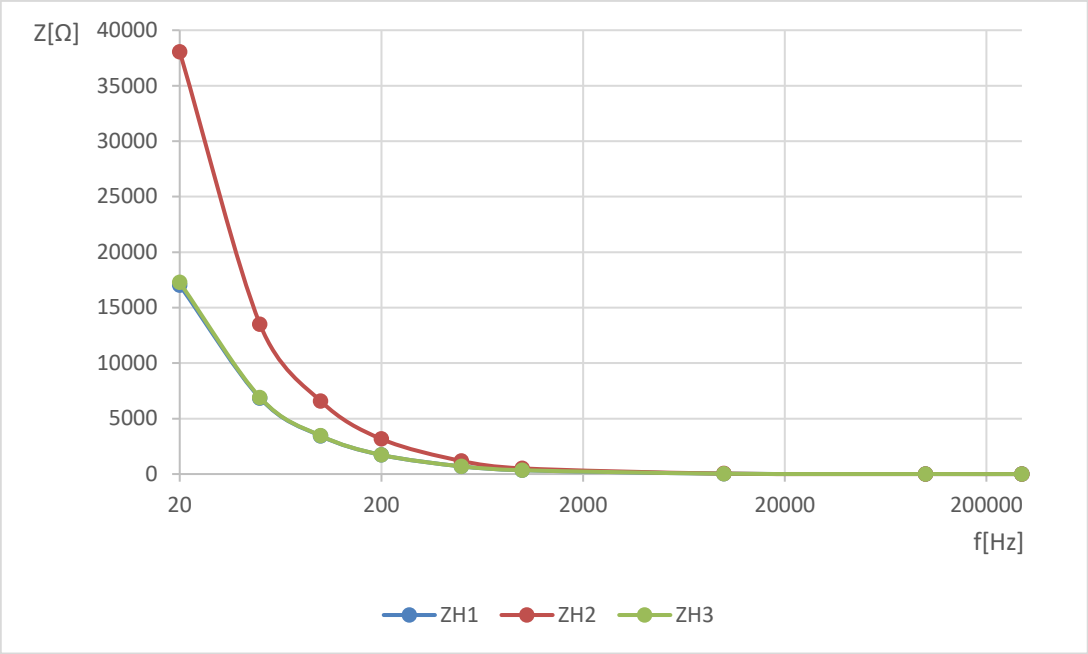
	Keramické kondenzátory									
f[Hz]	20	50	100	200	500	1k	10k	100k	300k	1M
	Keramický kondenzátor 0.9nF									
Z_{F1}[Ω]	8335000	3307000	1686000	854000	342500	171000	17810	1761	593	-
Z_{F2}[Ω]	59188	61276	60370	60071	60146	59997	36205	2815	831	236
Z_{F3}[Ω]	8846426	3538570	1769285	884643	353857	176929	17693	1769	590	18
ΔZ[Ω]	8275812	3245724	1625630	793929	282354	111003	18395	1054	238	
	Keramický kondenzátor 95.21nF									
Z_{G1}[Ω]	83160	33490	16830	8444	3393	1699	172	17	6	-
Z_{G2}[Ω]	57471	51441	38672	16601	6510	3109	253	17	6	0
Z_{G3}[Ω]	83623	33449	16725	8362	3345	1672	167	17	6	0
ΔZ[Ω]	-25689	17951	21842	8157	3117	1410	81	0	0	
	Keramický kondenzátor 461.2nF									
Z_{H1}[Ω]	17020	6846	3434	1719	691	346	36	3	1	-
Z_{H2}[Ω]	38061	13501	6577	3159	1176	513	44	1	0	1
Z_{H3}[Ω]	17263	6905	3453	1726	691	345	35	3	1	0
ΔZ[Ω]	21041	6655	3143	1440	485	167	8	-2	-1	-

Tabulka 14 - porovnání testovaných výsledků cívek s modelem 891 a teorií

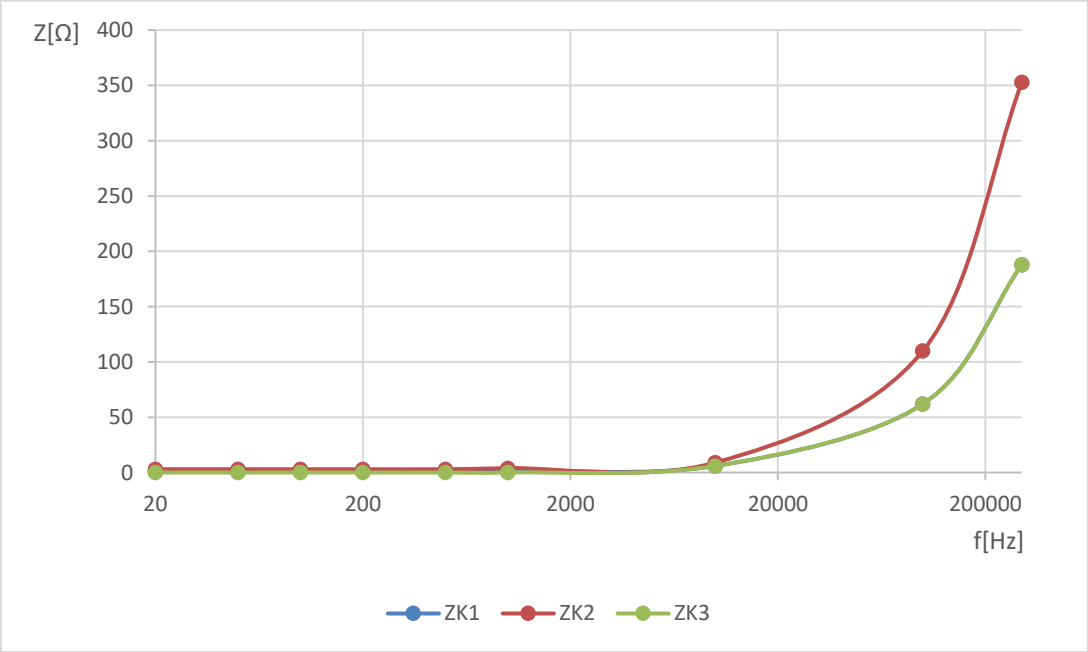
	Cívky									
f[Hz]	20	50	100	200	500	1k	10k	100k	300k	1M
	Cívka 1.113uH									
Z_{I1}[Ω]	0	0	0	0	0	0	0	0	2	-
Z_{I2}[Ω]	3	3	3	3	3	3	3	5	10	45
Z_{I3}[Ω]	0	0	0	0	0	0	0	0	2	7
ΔZ[Ω]	3	3	3	3	3	3	3	5	8	-
	Cívka 9.96uH									
Z_{J1}[Ω]	1	1	1	1	1	1	1	6	18	-
Z_{J2}[Ω]	3	3	3	3	4	4	4	9	30	133
Z_{J3}[Ω]	0	0	0	0	0	0	0	6	18	62
ΔZ[Ω]	2	2	2	2	3	3	3	3	12	-
	Cívka 100uH									
Z_{K1}[Ω]	1	1	1	1	1	2	6	62	188	-
Z_{K2}[Ω]	3	3	3	3	3	4	9	110	353	3620
Z_{K3}[Ω]	0	0	0	0	0	0	6	62	188	628
ΔZ[Ω]	2	2	2	2	2	2	3	48	165	-

Tabulka 15 - porovnání testovaných výsledků LED s modelem 891 a teorií

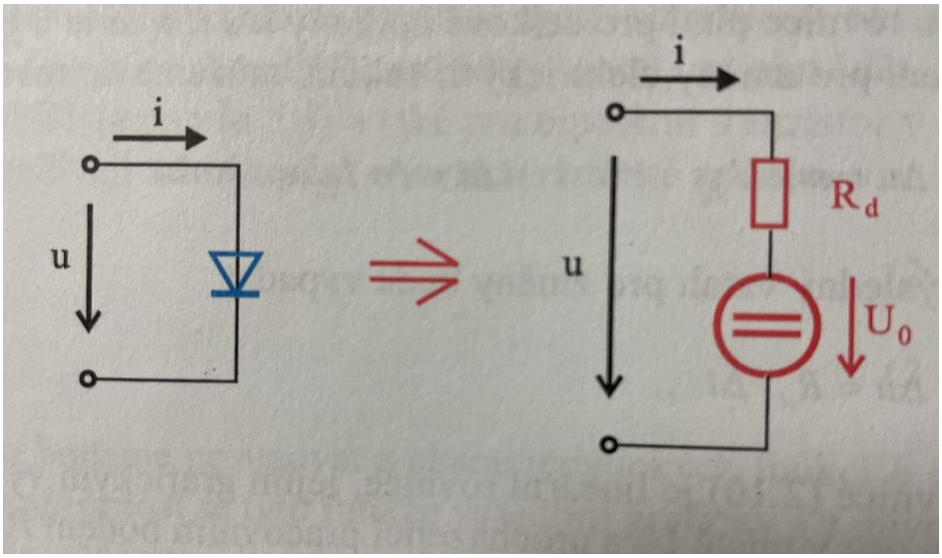
	Diody červené									
f[Hz]	20	50	100	200	500	1k	10k	100k	300k	1M
	Dioda Červená (A-K)									
Z_{L1}[Ω]	162000k	144000k	132000k	46000k	105000k	67000k	4000k	384k	162k	-
Z_{L2}[Ω]	55816	56912	57260	57121	57121	56982	57191	10422	304	302
ΔZ[Ω]	161944k	143943k	131942k	45942k	104942k	66943k	3942k	373k	161k	-
	Dioda Červená (K-A)									
Z_{M1}[Ω]	142000k	132000k	121000k	45000k	121000k	65000k	3000k	382k	162k	-
Z_{M2}[Ω]	59481	62200	60670	60146	61276	60670	59554	19568	6318	1687
ΔZ[Ω]	141940k	131937k	120939k	44939k	120938k	64939k	2940k	362k	155k	-



Obrázek 5.1 – Odvození křivky závislosti keramického kondenzátoru [vlastní obrázek]



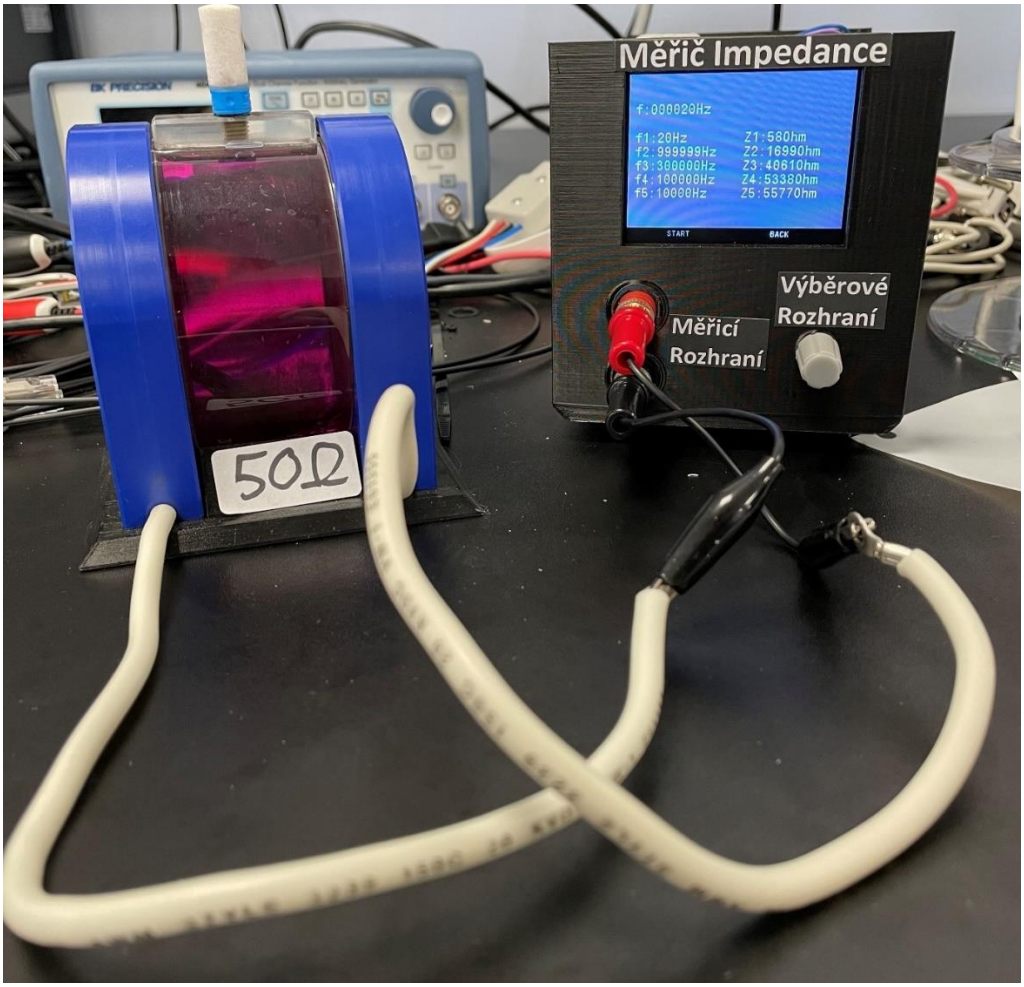
Obrázek 5.2 – Odvození křivky závislosti cívky, [vlastní obrázek]



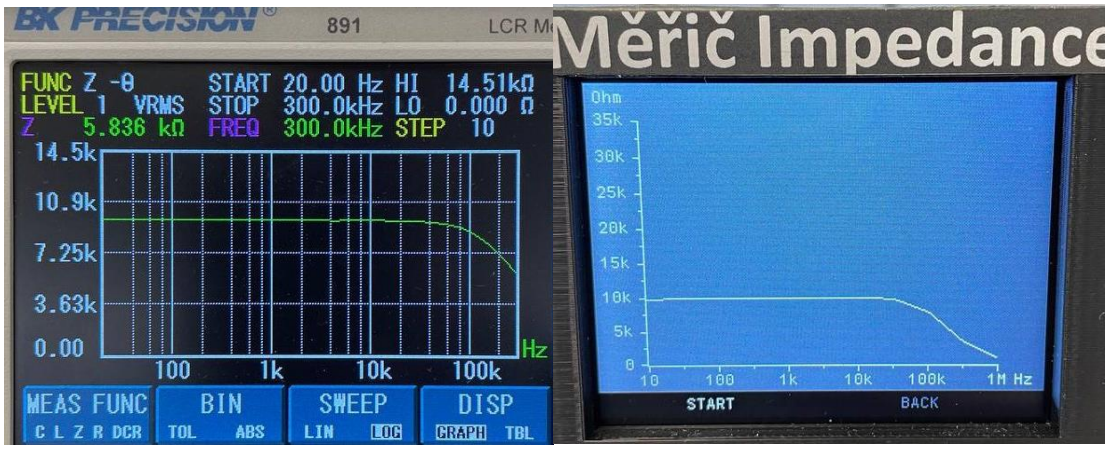
Obrázek 5.3 – Náhradní schéma diody, [39]

Je třeba vzít v úvahu, že odpory menší než 1 ohm nebyly pro výpočet přesnosti použity, protože přístroj není schopen měřit hodnoty menší než 1 ohm. Přesnost také prudce klesá s rostoucím odporem až na 40 %. Vzhledem k tomu, že zařízení používá k převodu napětí na impedanci maximálně 3,3 V, je pokles přesnosti nevyhnutelný, například při impedanci 33 kiloohmů, na každý kiloohm je 0,1 voltu.

Pohled na impedanční analyzátor je na obrázku 5.5.



Obrázek 5.4 – Zapojení přístroje v průběhu měření, [vlastní obrázek]



Obrázek 5.5 – Pohled na impedanční analyzátory, [vlastní obrázky]

Závěr

Cílem práce bylo vytvořit cenově dostupný měřič impedance. Toto zadání bylo splněno vytvořením výše popsaného zařízení, které na základě měření vykreslí impedanci v závislosti na frekvenci, nebo měří impedanci pro jednotlivé frekvence dle zadání uživatele.

V teoretické části byl proveden rozbor potřebných fyzikálních principů pro tvorbu zařízení. Dále byl proveden průzkum trhu za účelem zjištění dostupných obvodů, které lze využít k tvorbě zařízení a jejich parametrů. V rámci průzkumu trhu byla sledována také pořizovací cena jednotlivých obvodů a výběr byl prováděn jako funkce v závislosti na požadovaných parametrech a tlaku na minimalizaci nákladů. Cena navrženého zařízení je kolem 5000 CZK, což je cena cca 8x nižší než v případě komerčního řešení (například: Model 891, 300kHz Bench LCR Meter dostupný na webu www.bkprecision.com za cenu 42323 CZK). Cenou za toto snížení ceny je nižší přesnost zařízení.

Funkčnost a vlastnosti navrženého přístroje byly ověřeny sadou měření na referenčních zátěžích a výsledky porovnány s komerčním měřičem impedancí, případně s teoretickými předpoklady (u cívek a kondenzátorů). Z měření plyne, že vytvořený přístroj pracuje s přesností 1,94% v rozsahu impedancí od 0 do 10000 Ohm. Omezení měřitelných impedancí, které lze měřit je dáno použitými referenčními rezistory, pokud je potřeba měřit jiný rozsah impedancí, bylo by nutné nahradit (přepájet) referenční rezistory v přístroji.

V budoucnu by se autor chtěl zaměřit na zvýšení kvality měření, možností je zvýšení výstupního signálu na ±9 V (místo aktuálních 3,3 V) a použití externího ADC. Případně upravit hodnoty přepínaných referenčních rezistorů nebo navýšení jejich počtu použitím více multiplexorů nebo multiplexoru s více stavy. Dalšími plány do budoucna je přidat funkcí automatického potlačení vlivu přívodních kabelů (odečtení jejich příspěvku k celkové impedanci).

Literatura

[1] *Model 891, 300kHz Bench LCR Meter* [online]. In: . [cit. 2024-04-09]. Dostupné z: https://www.bkprecision.com/products/component-testers/891

[2] *Elektrický odpor* [online]. In: . [cit. 2024-04-09]. Dostupné z: https://eluc.ikap.cz/verejne/lekce/421

[3] TKOTZ, Klaus et al. *Příručka pro elektrotechnika*. 2. doplněné vyd. Praha: Europa – Sobotáles, 2006. ISBN 80-86706-13-3.

[4] *Sledovač napětí* [online]. In: . [cit. 2024-04-09]. Dostupné z: https://eluc.ikap.cz/verejne/lekce/690

[5] *Základní zapojení s operačními zesilovači* [online]. In: . [cit. 2024-04-09]. Dostupné z: https://eluc.ikap.cz/verejne/lekce/689

[6] *Rozdílové zapojení operačního zesilovače* [online]. In: . [cit. 2024-04-09]. Dostupné z: https://eluc.ikap.cz/verejne/lekce/692

[7] *Peak Detector* [online]. In: . [cit. 2024-04-09]. Dostupné z: https://electronicscoach.com/peak-detector.html

[8] *STM32F103x8: STM32F103xB* [online]. In: STMICROELECTRONICS. 2023 [cit. 2024-04-09]. Dostupné z: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf

[9] *STM32G061x6/x8* [online]. In: STMICROELECTRONICS. 2021 [cit. 2024-04-09]. Dostupné z: https://www.mouser.com/datasheet/2/389/stm32g061c6-2450003.pdf

[10] *Low Power, 12.65 mW, 2.3 V to 5.5 V, Programmable Waveform Generator* [online]. In: ANALOG DEVICES. [cit. 2024-04-09]. Dostupné z: https://www.analog.com/media/en/technical-documentation/data-sheets/ad9833.pdf

[11] *20 mW Power, 2.3 V to 5.5 V, 75 MHz Complete DDS* [online]. In: ANALOG DEVICES. [cit. 2024-04-09]. Dostupné z: https://www.analog.com/media/en/technical-documentation/data-sheets/ad9834.pdf

[12] *1 GSPS Direct Digital Synthesizer* [online]. In: ANALOG DEVICES. [cit. 2024-04-09]. Dostupné z: https://www.analog.com/media/en/technical-documentation/data-sheets/AD9858.pdf

[13] *ADA4098-1/ADA4098-2* [online]. In: ANALOG DEVICES. [cit. 2024-04-09]. Dostupné z: https://www.analog.com/media/en/technical-documentation/data-sheets/ada4098-1_ada4098-2.pdf

[14] *High Performance, 145 MHz FastFET Op Amps* [online]. In: ANALOG DEVICES. [cit. 2024-04-09]. Dostupné z: https://www.analog.com/media/en/technical-documentation/data-sheets/ad8065_8066.pdf

[15] *MAX4475–MAX4478: MAX4488/MAX4489* [online]. In: MAXIM INTEGRATED. 2019 [cit. 2024-04-09]. Dostupné z: https://www.analog.com/media/cn/technical-documentation/data-sheets/3085.pdf

[16] *THS3491 900-MHz, 500-mA High-Power Output Current Feedback Amplifier* [online]. In: TEXAS INSTRUMENTS. 2023 [cit. 2024-04-09]. Dostupné z: https://www.ti.com/lit/ds/symlink/ths3491.pdf?ts=1712652506424&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTHS3491

[17] *4 Ω RON, 4-/8-Channel, ±15 V/+12 V/±5 V iCMOS Multiplexers: ADG1408/ADG1409* [online]. In: ANALOG DEVICES. [cit. 2024-04-09]. Dostupné z: https://www.analog.com/media/en/technical-documentation/data-sheets/adg1408_1409.pdf

[18] *Low Power Mixer/Limiter/RSSI 3 V Receiver IF Subsystem: AD608* [online]. In: ANALOG DEVICES. [cit. 2024-04-09]. Dostupné z: https://www.analog.com/media/en/technical-documentation/data-sheets/ad608.pdf

[19] *Precision, 16-Channel/Dual 8-Channel, High-Performance, CMOS Analog Multiplexers: MAX306/MAX307* [online]. In: ANALOG DEVICES. [cit. 2024-04-09]. Dostupné z: https://cz.mouser.com/datasheet/2/609/MAX306_MAX307-3128338.pdf

[20] *24V 1A 24W ISOLATED SWITCH POWER SUPPLY MODULE* [online]. [cit. 2024-04-09]. Dostupné z: https://ifuturetech.org/product/24v-1a-24w-isolated-switch-power-supply-module/

[21] OEM. *Napájecí adaptér 230V / 24V / 1A stejnosměrný* [online]. [cit. 2024-04-09]. Dostupné z: https://www.softcom.cz/eshop/napajeci-adapter-230v-24v-1a-stejnosmerny_d168037.html

[22] MEANWELL. *ADVANTECH BB-MDR-20-24* [online]. [cit. 2024-04-09]. Dostupné z: https://www.westercom.eu/cs/napajeci-zdroje/6356/advantech-bb-mdr-20-24.html?gad_source=1&gclid=Cj0KCQjwztOwBhD7ARIsAPDKnkCOaDd-pTDSH-WkxiM8erQeI0OPvbFn6xtzZ5aJSKywDct6sEtyEWUaAgMREALw_wcB

[23] USONGSHINE. *DC 5V/12V/24V Computer CPU Cooler Mini Cooling Fan* [online]. [cit. 2024-04-09]. Dostupné z:

https://aliexpress.ru/item/4001345763935.html?sku_id=10000015757632568&spm=a2g2w.productlist.search_results.2.70205c0aIqo5lO

[24] AIREN. *Airen RedWings 40H* [online]. [cit. 2024-04-09]. Dostupné z: https://www.czc.cz/airen-redwings-40h/95661/produkt

[25] *VENTILÁTOR 4010, 24V, 2 PINY, FAN4010242* [online]. [cit. 2024-04-09]. Dostupné z: https://www.levne3dtiskarny.cz/cs/chlazeni/233-ventilator-4010-24v-2-piny-fan4010242.html

[26] *TFT Display 0.96/1.3/1.44/1.77/1.8/2.0 inch IPS 7P SPI HD 65K TFT Full Color LCD Module ST7735 Drive For Arduino* [online]. [cit. 2024-04-09]. Dostupné z: https://aliexpress.ru/item/1005004259333504.html?sku_id=12000028552457421&spm=a2g2w.productlist.search_results.7.4830139629ZsQm

[27] *1.8/2.4/2.8/3.5 Inch TFT Full Color Screen LCD Display Module With Touch Drive IC 9341 7789 SPI 128*128 240*320 For Arduino* [online]. [cit. 2024-04-09]. Dostupné z: https://aliexpress.ru/item/1005004542864792.html?sku_id=12000034614053664&spm=a2g2w.productlist.search_results.9.4830139629ZsQm

[28] *New 4.0 inch SPI serial port LCD Touch screen Module 480*320 (Free Shipping)TFT display ILI9488 HD Electronic STM32 ESP32 AR C51* [online]. [cit. 2024-04-09]. Dostupné z: https://aliexpress.ru/item/1005005791515997.html?sku_id=12000034370064544&spm=a2g2w.productlist.search_results.0.4830139629ZsQm

[29] *Rotary Encoder Module for Arduino Brick Sensor Development Round Audio Rotating Potentiometer Knob Cap EC11* [online]. [cit. 2024-04-09]. Dostupné z: https://aliexpress.ru/item/1005002132981406.html?sku_id=12000018845576505&spm=a2g2w.productlist.search_results.0.4830139629ZsQm

[30] *TZT 360 Degrees Rotary Encoder Module For Arduino Brick Sensor Switch Development Board KY-040 With Pins* [online]. [cit. 2024-04-09]. Dostupné z: https://aliexpress.ru/item/32790788377.html?sku_id=12000037069013103&spm=a2g2w.productlist.search_results.7.4830139629ZsQm

[31] *5pcs Terminal 4mm banana socket copper plug socket panel test 4 mm socket high insulation safety mother seat Connector* [online]. [cit. 2024-04-09]. Dostupné z:

https://aliexpress.ru/item/32960228827.html?sku_id=66548272990&spm=a2g2w.productlist.search_results.0.b0d92ad5CPpYKR

[32] *4mm Banana Socket Panel Mount 4mm Banana Female Binding Post Connector 10Pcs/Lot* [online]. [cit. 2024-04-09]. Dostupné z: https://aliexpress.ru/item/32819103491.html?gatewayAdapt=glo2rus&sku_id=10000004293315206

[33] *KiCad-Logo.svg* [online]. In: , KiCad Developers Team. KICAD. 2016 [cit. 2024-04-09]. Dostupné z: https://en.m.wikipedia.org/wiki/File:KiCad-Logo.svg

[34] *XC6902* [online]. In: TOREX. [cit. 2024-04-09]. Dostupné z: https://www.mouser.com/datasheet/2/760/xc6902-3371519.pdf

[35] *Fusion360 Logo.png* [online]. In: AUTODESK. 2022 [cit. 2024-04-09]. Dostupné z: https://en.wikipedia.org/wiki/File:Fusion360_Logo.png

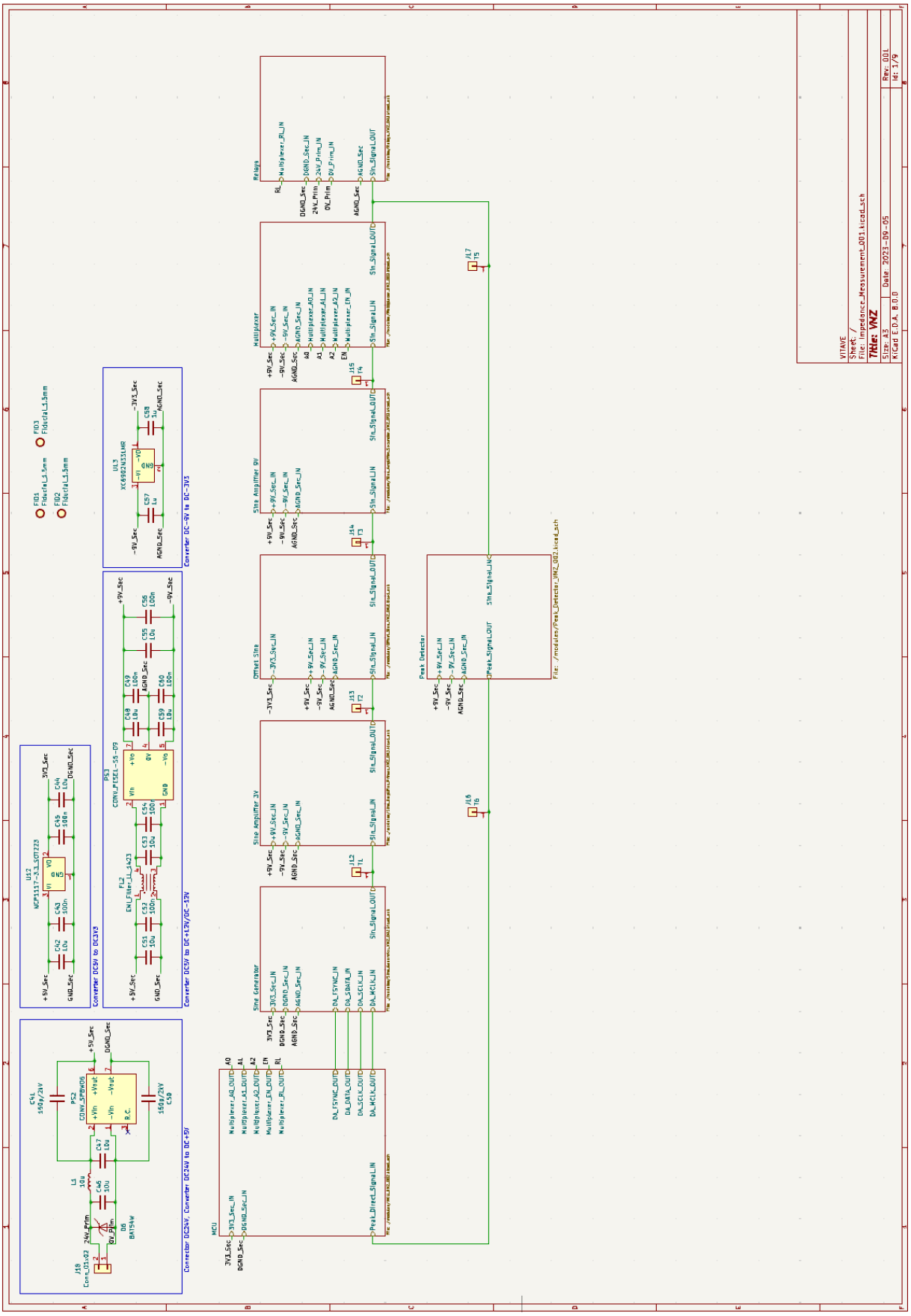
[36] *Laboratorní zdroj LW-K3010D 0-30V/0-10A* [online]. [cit. 2024-04-09]. Dostupné z: https://www.hadex.cz/g853-laboratorni-zdroj-lw-k3010d-0-30v0-10a/?utm_source=google&utm_medium=cpc&utm_campaign=17937749015&gad_source=1&gclid=Cj0KCQjwztOwBhD7ARIsAPDKnkB1mS0vKRmMTDYOg8H_Um6RYXGSUvO_HKteESOCKwD72CDNa7e2HOcaAhjGEALw_wcB

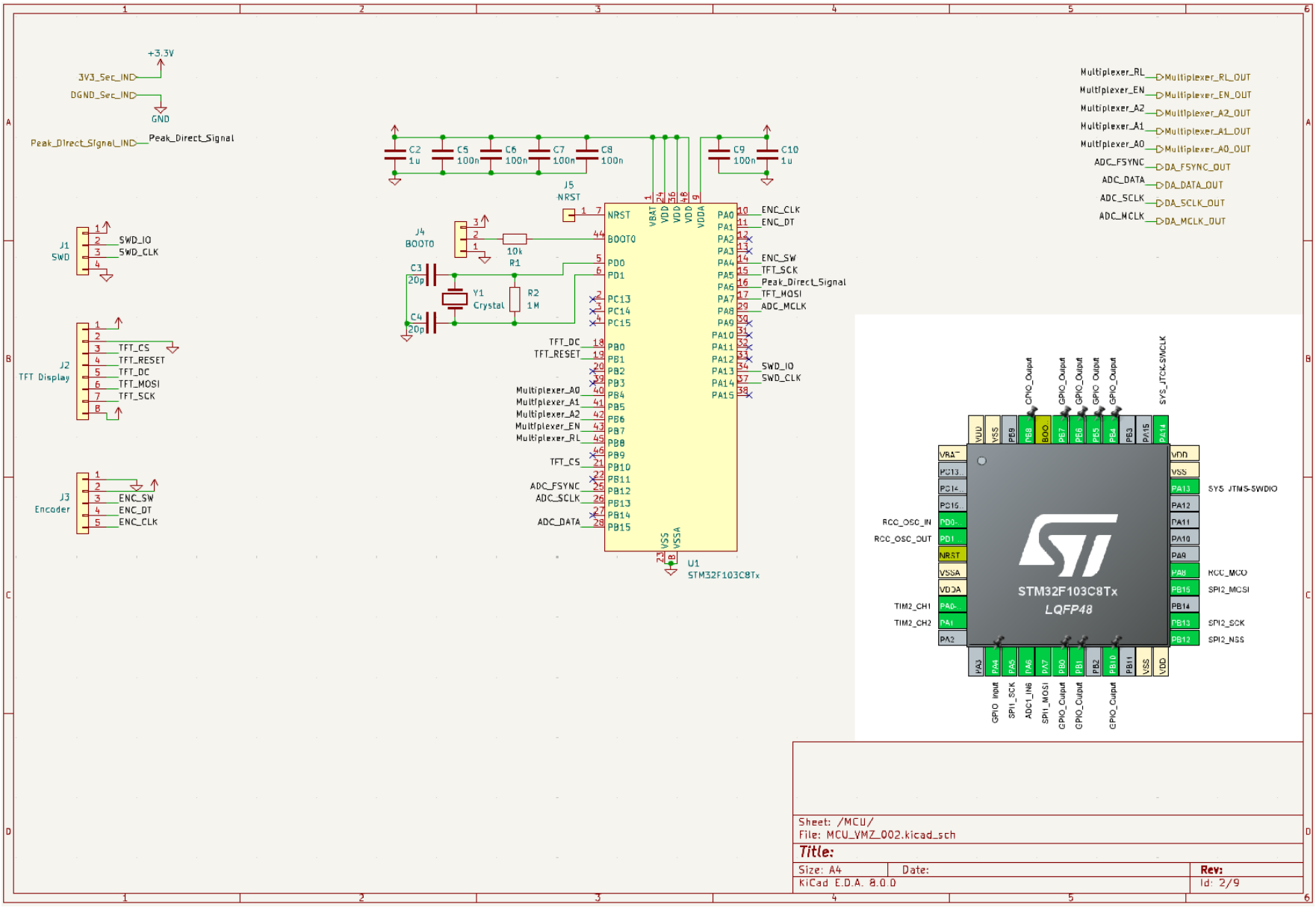
[37] *STM32CubeMX* [online]. In: . [cit. 2024-04-09]. Dostupné z: https://wiki.st.com/stm32mpu/wiki/STM32CubeMX

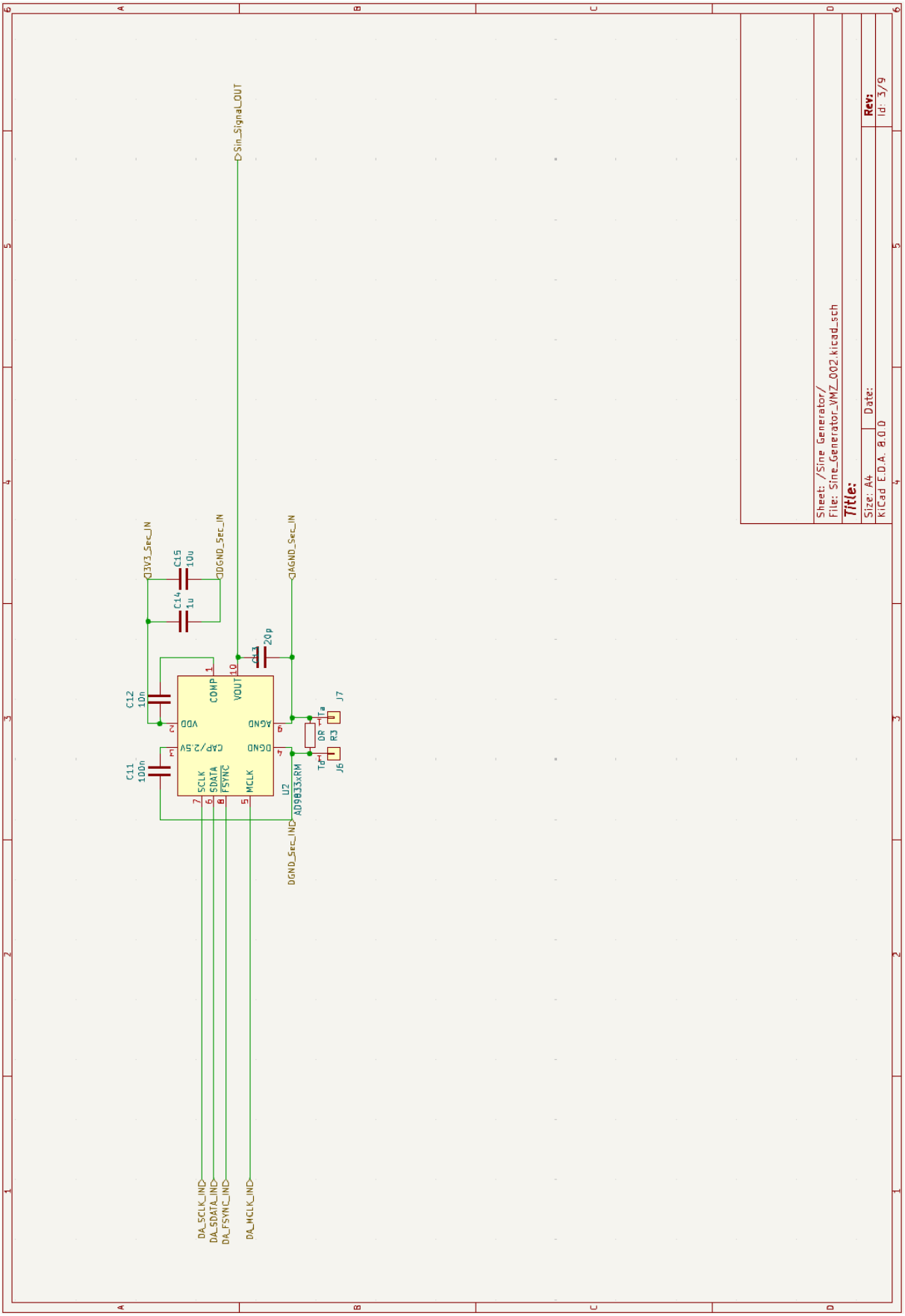
[38] *STM32CubeIDE* [online]. In: . [cit. 2024-04-09]. Dostupné z: https://www.st.com/content/st_com/en/stm32cubeide.html

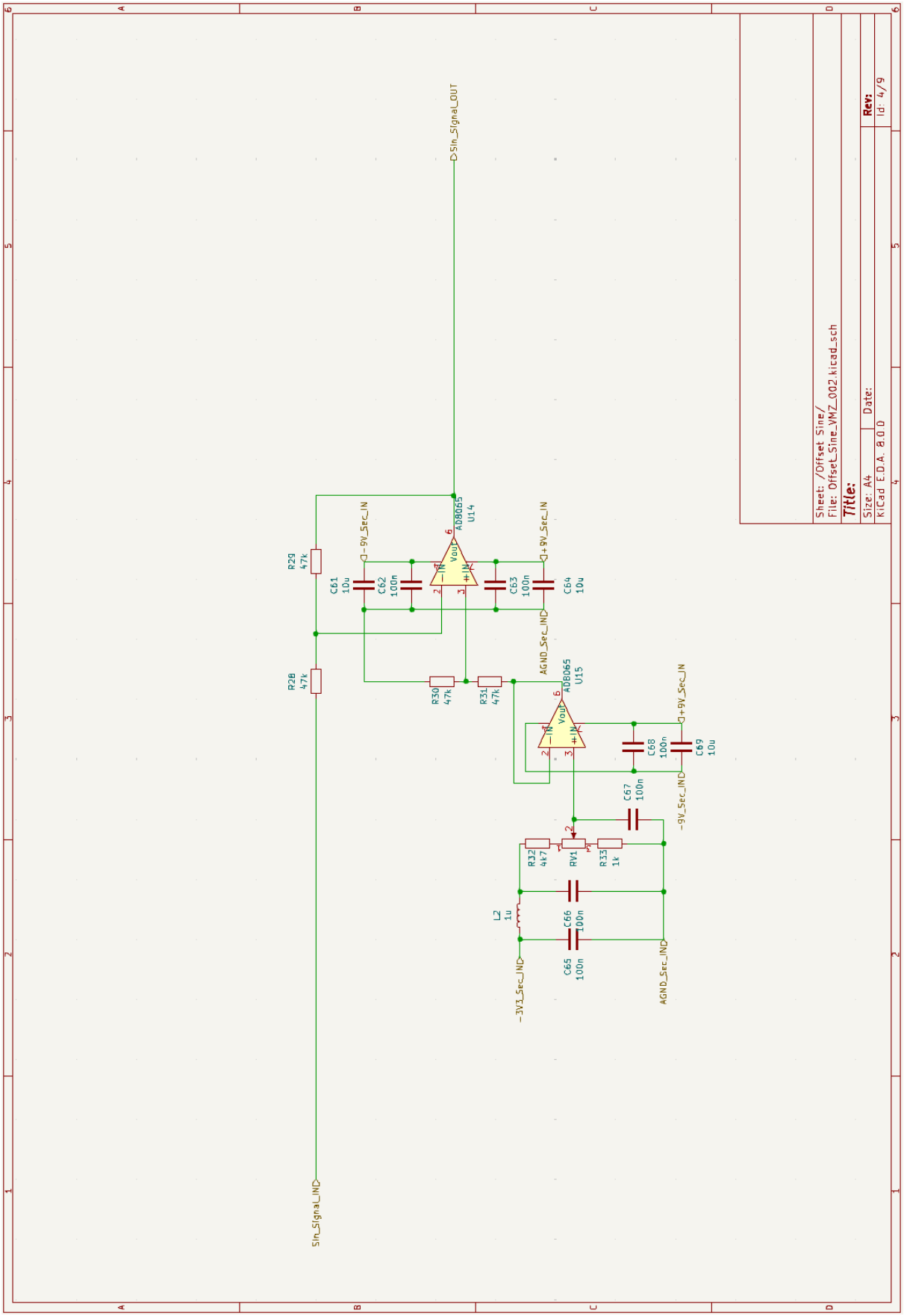
[39] VOBECKÝ, Jan a Vít ZÁHLAVA. *Elektronika: součástky a obvody, principy a příklady*. 3., rozš. vyd. Praha: Grada, 2005. ISBN 8024712415.

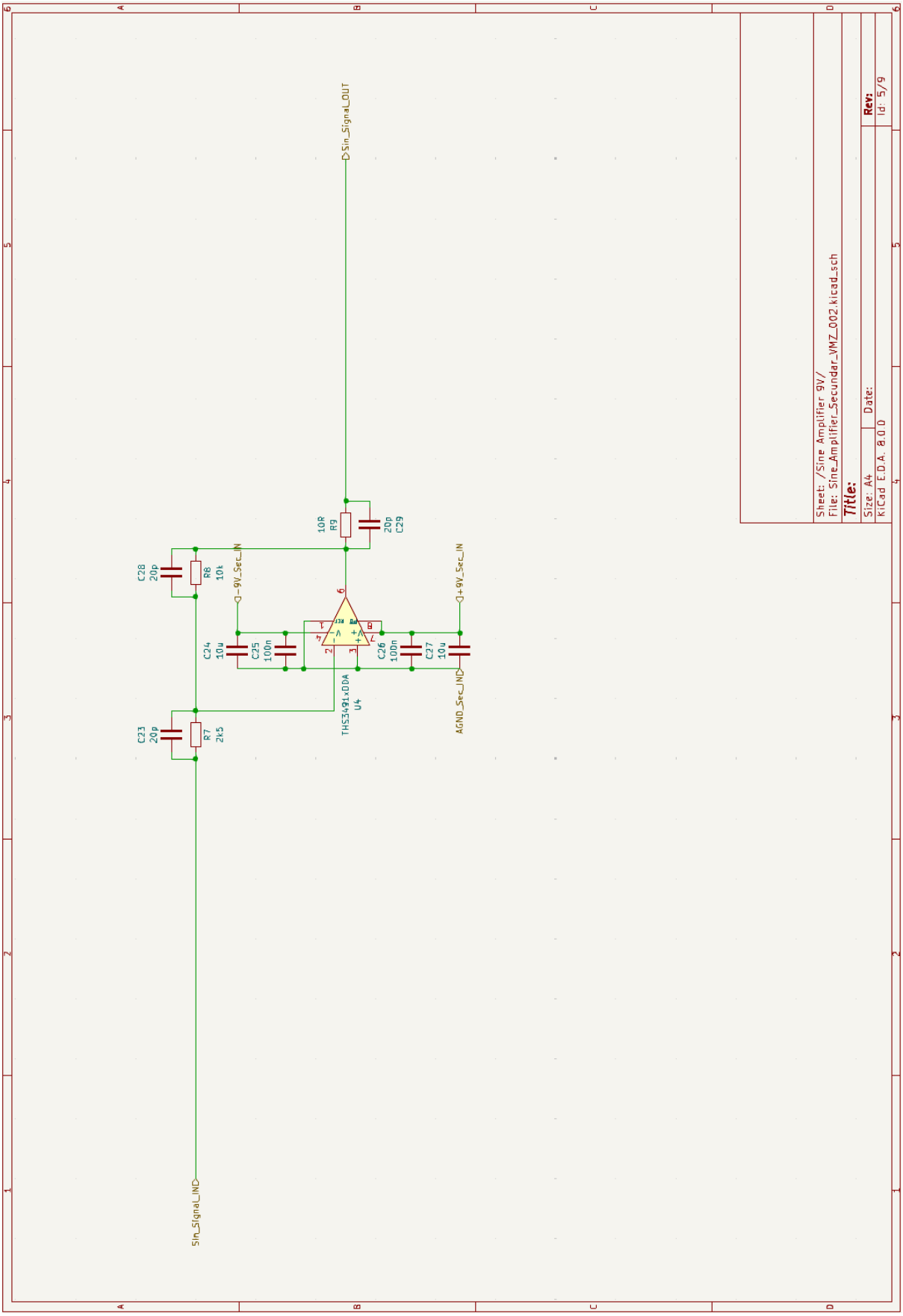
Příloha A –

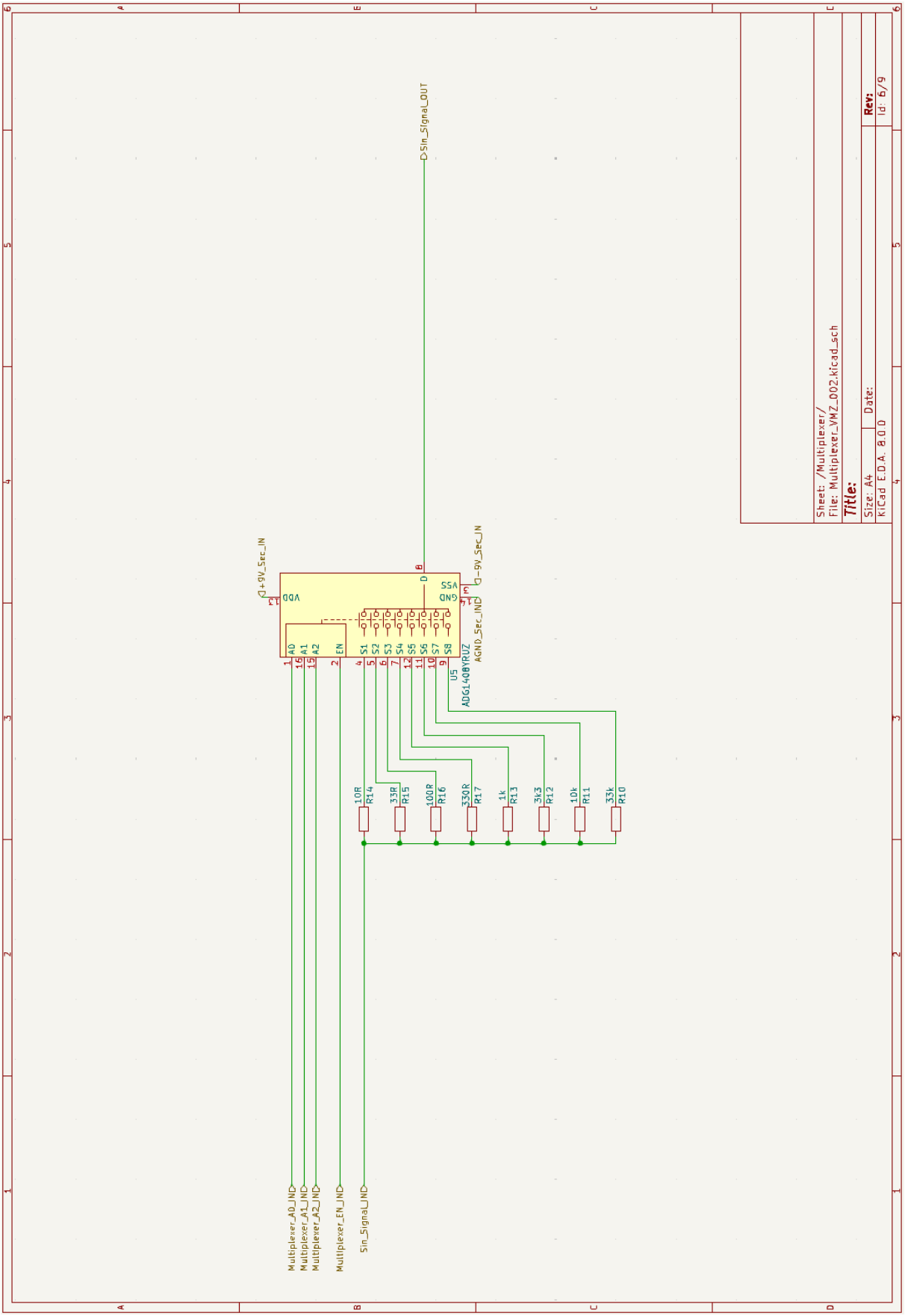

VITAVE
Sheet: /
File: Impedance_Measurement_001.kicad_sch
Title: VNZ
Size: A3
Date: 2023-09-05
KiCad E.D.A. 8.0.0
Rev: 001
Id: 1/9
Converter DC5V to DC3V3
Converter DC5V to DC+12V/DC-12V
Converter DC-9V to DC-3V3
Converter DC24V, Converter DC24V to DC+5V
MCU
Sine Generator
Sine Amplifier 3V
Sine Amplifier 9V
Offset Sine
Peak Detector
Multiplexer
Relays


+3.3V
3V3_Sec_IND
DGND_Sec_IND
GND
Peak_Direct_Signal_IND
Peak_Direct_Signal
J1
SWD
SWD_IO
SWD_CLK
J2
TFT Display
TFT_CS
TFT_RESET
TFT_DC
TFT_MOSI
TFT_SCK
J3
Encoder
ENC_SW
ENC_DT
ENC_CLK
C2 1u
C5 100n
C6 100n
C7 100n
C8 100n
C9 100n
C10 1u
J5
NRST
J4
BOOT0
10k
R1
C3 20p
C4 20p
Y1
Crystal
R2 1M
U1
STM32F103C8Tx
Multiplexer_RL
Multiplexer_EN
Multiplexer_A2
Multiplexer_A1
Multiplexer_A0
ADC_FSYNC
ADC_DATA
ADC_SCLK
ADC_MCLK
Multiplexer_RL_OUT
Multiplexer_EN_OUT
Multiplexer_A2_OUT
Multiplexer_A1_OUT
Multiplexer_A0_OUT
DA_FSYNC_OUT
DA_DATA_OUT
DA_SCLK_OUT
DA_MCLK_OUT
ENC_CLK
ENC_DT
ENC_SW
TFT_SCK
Peak_Direct_Signal
TFT_MOSI
ADC_MCLK
TFT_DC
TFT_RESET
SWD_IO
SWD_CLK
STM32F103C8Tx
LQFP48
RCC_OSC_IN
RCC_OSC_OUT
TIM2_CH1
TIM2_CH2
SYS_JTMS-SWDIO
RCC_MCO
SPI2_MOSI
SPI2_SCK
SPI2_NSS
Sheet: /MCU/
File: MCU_VMZ_002.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.0
Id: 2/9

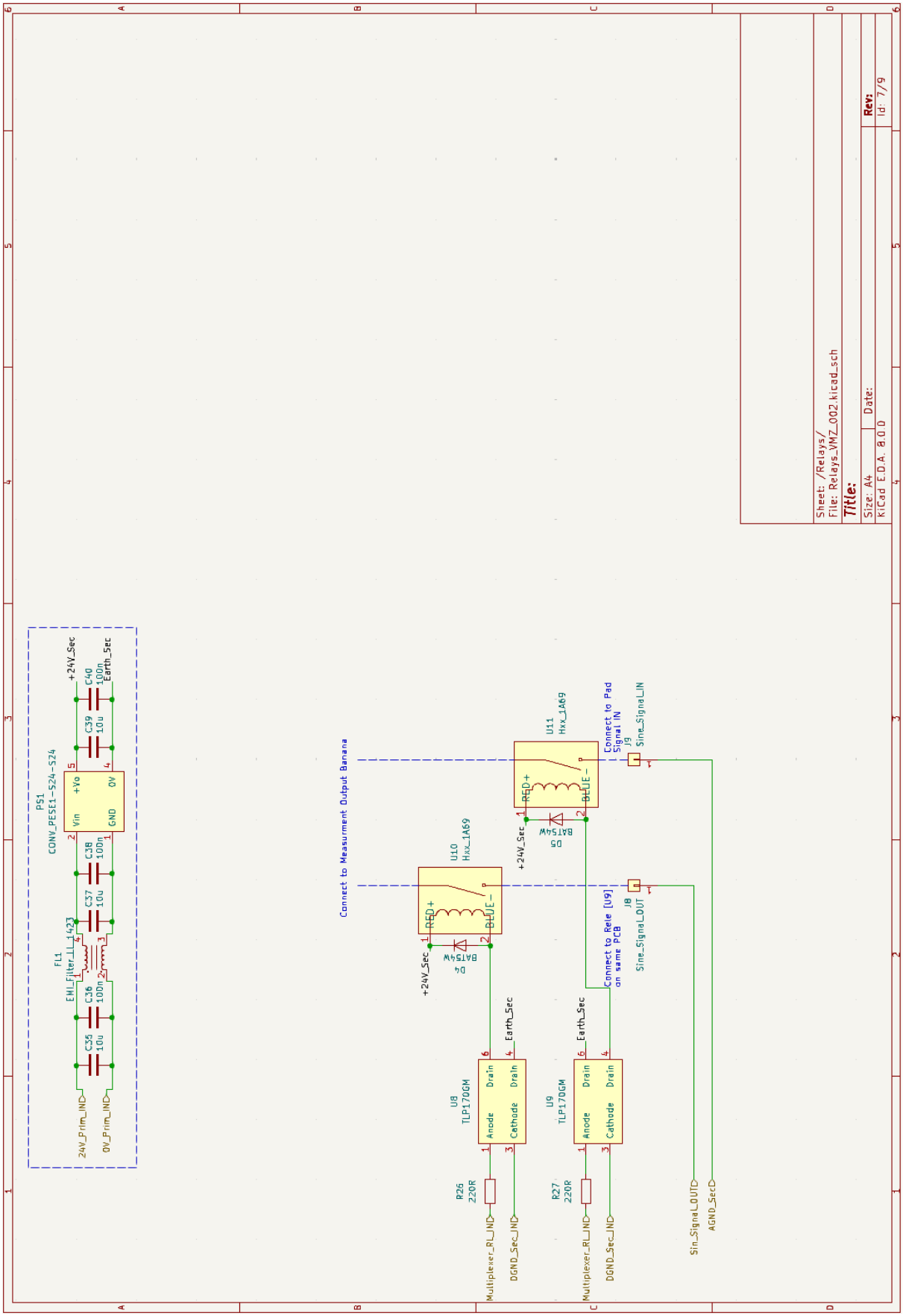

Sheet: /Sine Generator/
File: Sine_Generator_VMZ_002.kicad_sch
Title:
Size: A4
Date:
KiCad E.D.A. 8.0.0
Rev:
Id: 3/9
Sin_SignaL_OUT
3V3_Sec_IN
C15
10u
C14
1u
DGND_Sec_IN
AGND_Sec_IN
C13
20p
C12
10n
C11
100n
COMP
VOUT
VDD
AGND
CAP/2.5V
DGND
SCLK
SDATA
FSYNC
MCLK
U2
AD9833xRM
R3
J6
J7
DGND_Sec_IND
DA_SCLK_IND
DA_SDATA_IND
DA_FSYNC_IND
DA_MCLK_IND


Sin_Signal_OUT
-9V_Sec_IN
+9V_Sec_IN
AD8065
U14
R29
47k
R28
47k
C61
10u
C62
100n
C63
100n
C64
10u
R30
47k
R31
47k
AGND_Sec_IND
AD8065
U15
C68
100n
C69
10u
C67
100n
-9V_Sec_IND
R32
4k7
RV1
R33
1k
L2
1u
C66
100n
C65
100n
-3V3_Sec_IND
AGND_Sec_IND
Sin_Signal_IND
Sheet: /Offset Sine/
File: Offset_Sine_VMZ_002.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.0
Id: 4/9

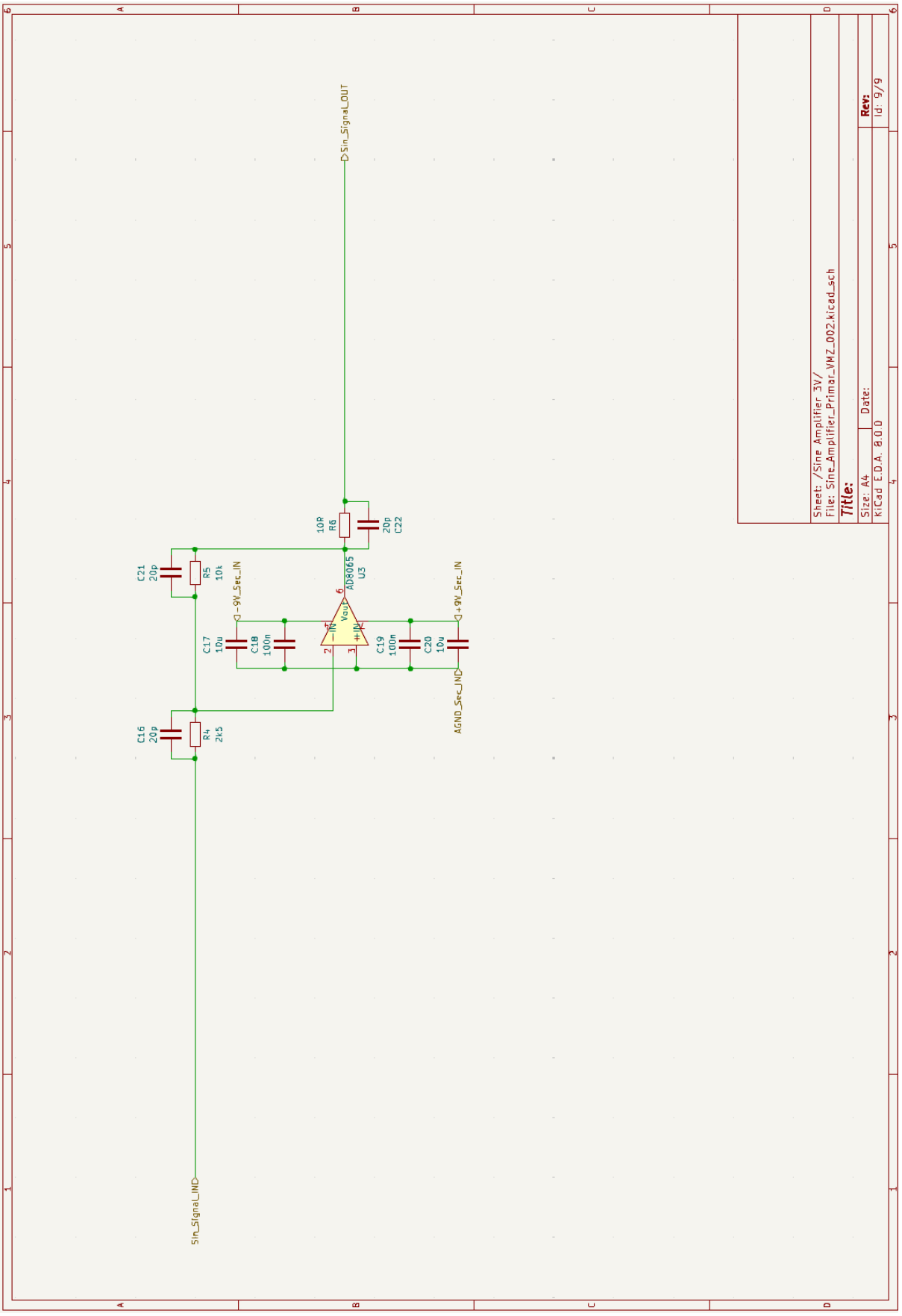

Sin_Signal_OUT
10R
R9
20p
C29
C28
20p
R8
10k
-9V_Sec_IN
+9V_Sec_IN
C24
10u
C25
100n
C26
100n
C27
10u
THS3491xDDA
U4
C23
20p
R7
2k5
Sin_Signal_IND
AGND_Sec_IND
Sheet: /Sine Amplifier 9V/
File: Sine_Amplifier_Secundar_VMZ_002.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.0
Id: 5/9


Sin_Signal_OUT
+9V_Sec_IN
-9V_Sec_IN
AGND_Sec_IND
U5
ADG1408BYRUZ
VDD
GND
VSS
A0
A1
A2
EN
S1
S2
S3
S4
S5
S6
S7
S8
D
10R
R14
33R
R15
100R
R16
330R
R17
1k
R13
3k3
R12
10k
R11
33k
R10
Multiplexer_A0_IND
Multiplexer_A1_IND
Multiplexer_A2_IND
Multiplexer_EN_IND
Sin_SignalIND
Sheet: /Multiplexer/
File: Multiplexer_VMZ_002.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.0
Id: 6/9

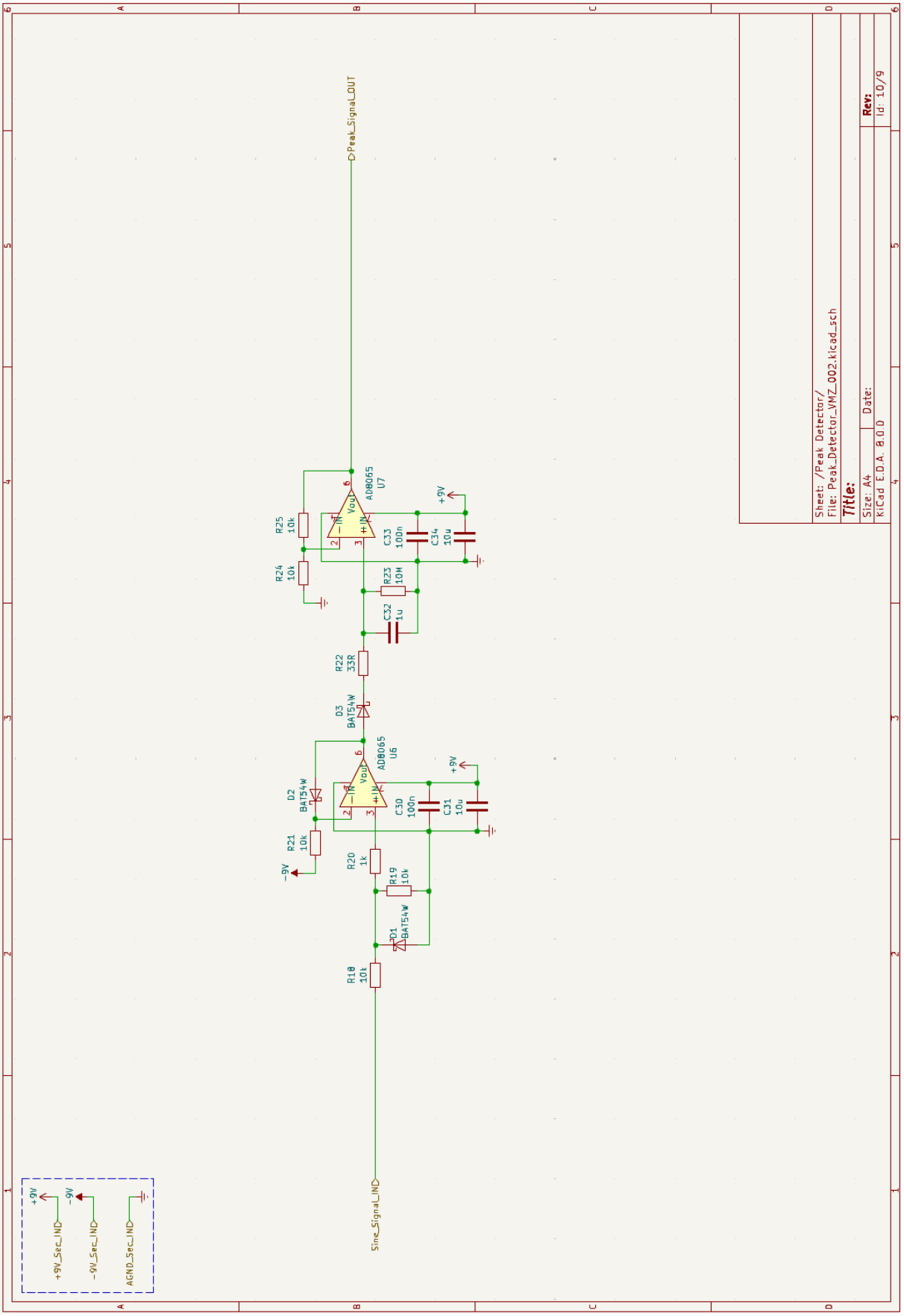
d


Sheet: /Relays/
File: Relays_VMZ_002.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.0
Id: 7/9
PS1
CONV_PESE1-S24-S24
Vin
+Vo
GND
0V
+24V_Sec
Earth_Sec
C40
100n
C39
10u
C38
100n
C37
10u
C36
100n
C35
10u
FL1
EMI_Filter_LL_1423
24V_Prim_IND
0V_Prim_IND
Connect to Measurment Output Banana
U10
Hxx_1A69
U11
Hxx_1A69
RED+
BLUE-
D4
BAT54W
D5
BAT54W
Connect to Pad Signal IN
J9
Sine_SignalIN
Connect to Rele [U9] on same PCB
J8
Sine_SignalOUT
U8
TLP170GM
U9
TLP170GM
Anode
Cathode
Drain
R26
220R
R27
220R
Multiplexer_RL_IND
DGND_Sec_IND
Sin_SignalOUTD
AGND_SecD

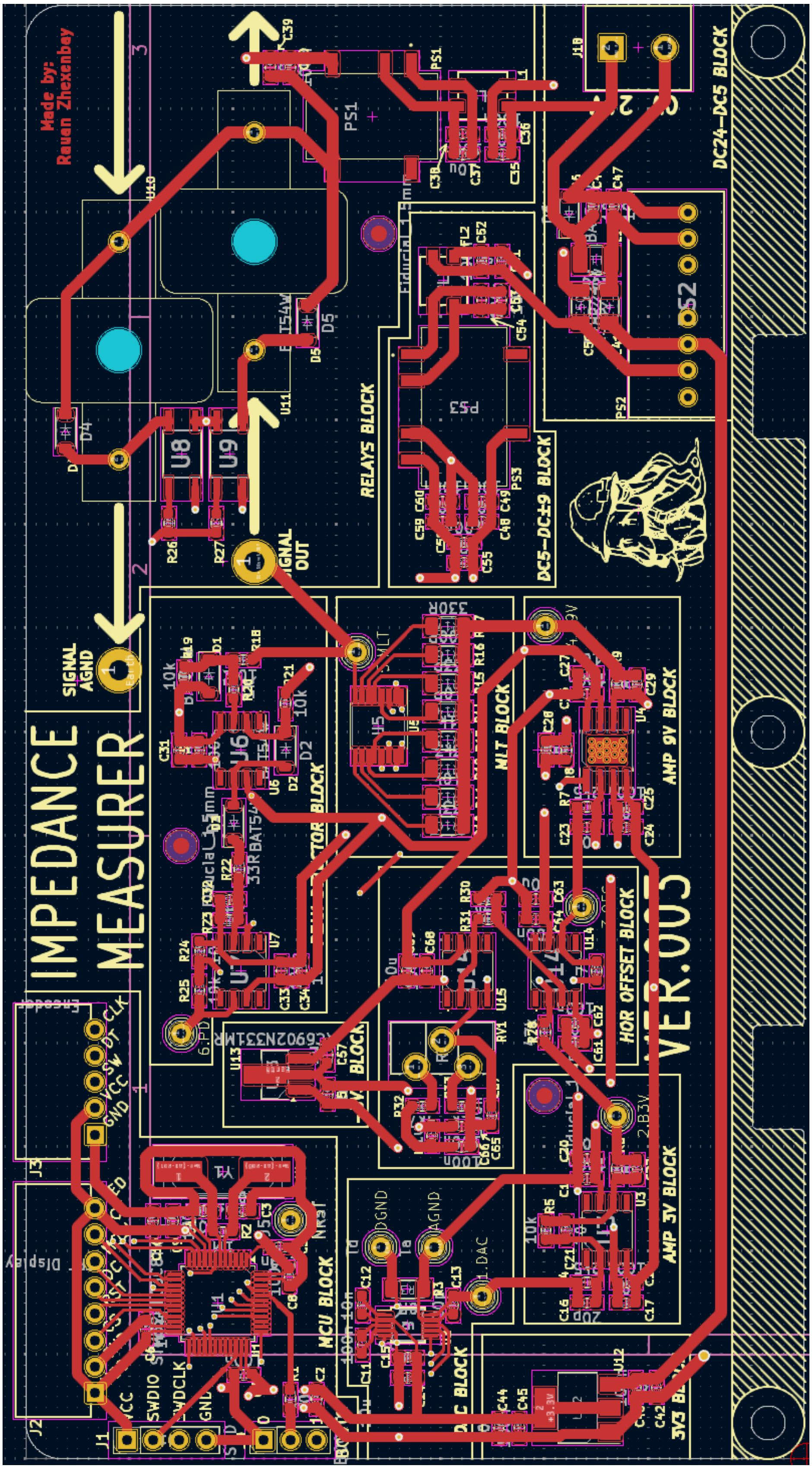
d

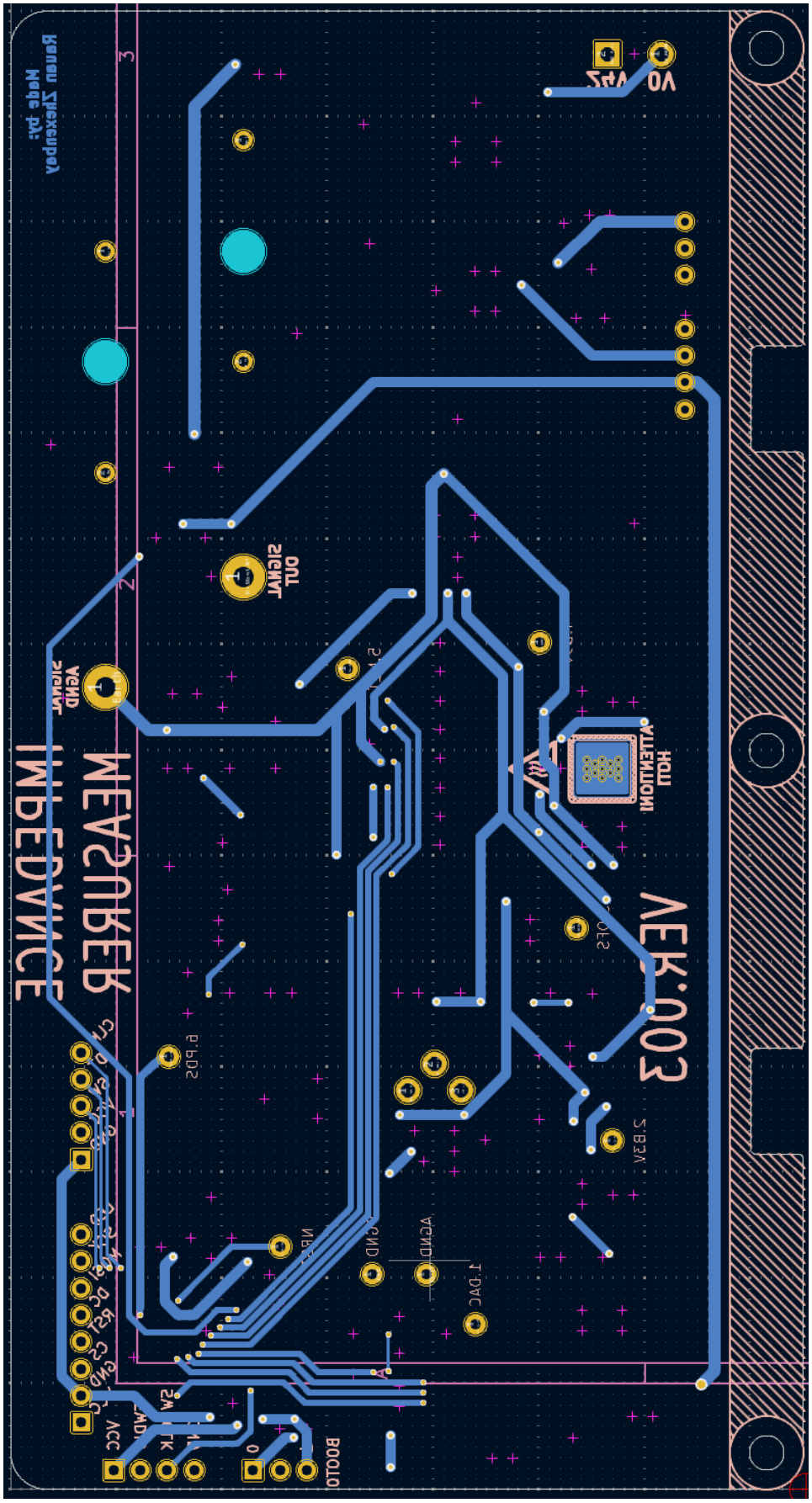

Sin_Signal_OUT
10R
R6
20p
C22
C21
20p
R5
10k
-9V_Sec_IN
AD8065
U3
+9V_Sec_IN
C17
10u
C18
100n
C19
100n
C20
10u
C16
20p
R4
2k5
Sin_Signal_IND
AGND_Sec_IND
Sheet: /Sine Amplifier 3V/
File: Sine_Amplifier_Primar_VMZ_002.kicad_sch
Title:
Size: A4
Date:
KiCad E.D.A. 6.0.0
Rev:
Id: 9/9

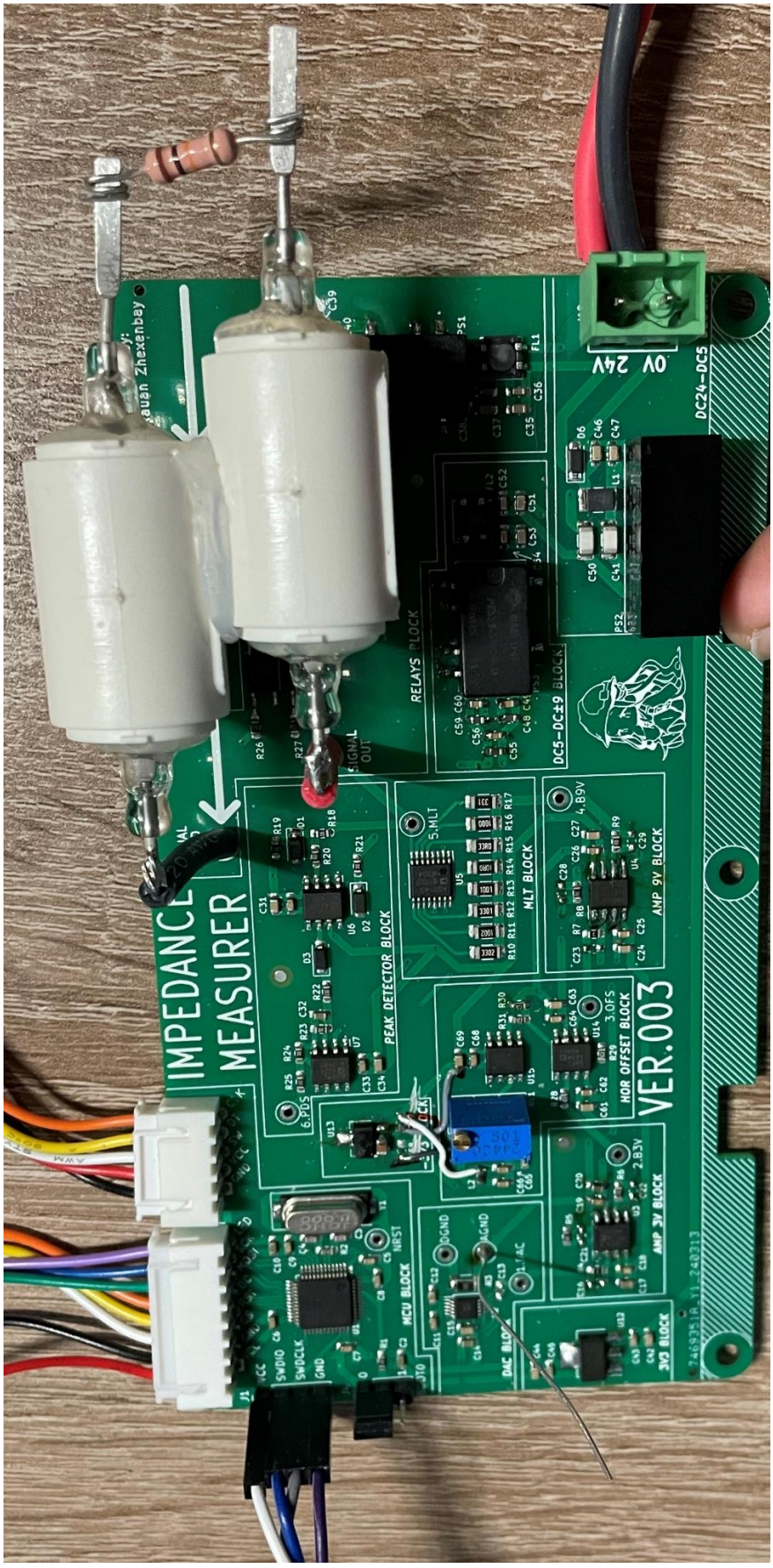
d


Peak_SignaLOUT
AD8065
U7
R25
10k
R24
10k
C33
100n
C34
10u
+9V
R23
10M
C32
1u
R22
33R
D3
BAT54W
AD8065
U6
D2
BAT54W
R21
10k
-9V
C30
100n
C31
10u
+9V
R20
1k
R19
10k
D1
BAT54W
R18
10k
Sine_Signal_IND
+9V
-9V
+9V_Sec_IND
-9V_Sec_IND
AGND_Sec_IND
Sheet: /Peak Detector/
File: Peak_Detector_VMZ_002.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.0
Id: 10/9

Příloha B –


IMPEDANCE MEASURER
VER.003
Made by: Rauan Zhexenbey
RELAYS BLOCK
DC24-DC5 BLOCK
MLT BLOCK
AMP 9V BLOCK
AMP 3V BLOCK
MCU BLOCK
3V3 BLOCK
SIGNAL OUT
SIGNAL AGND

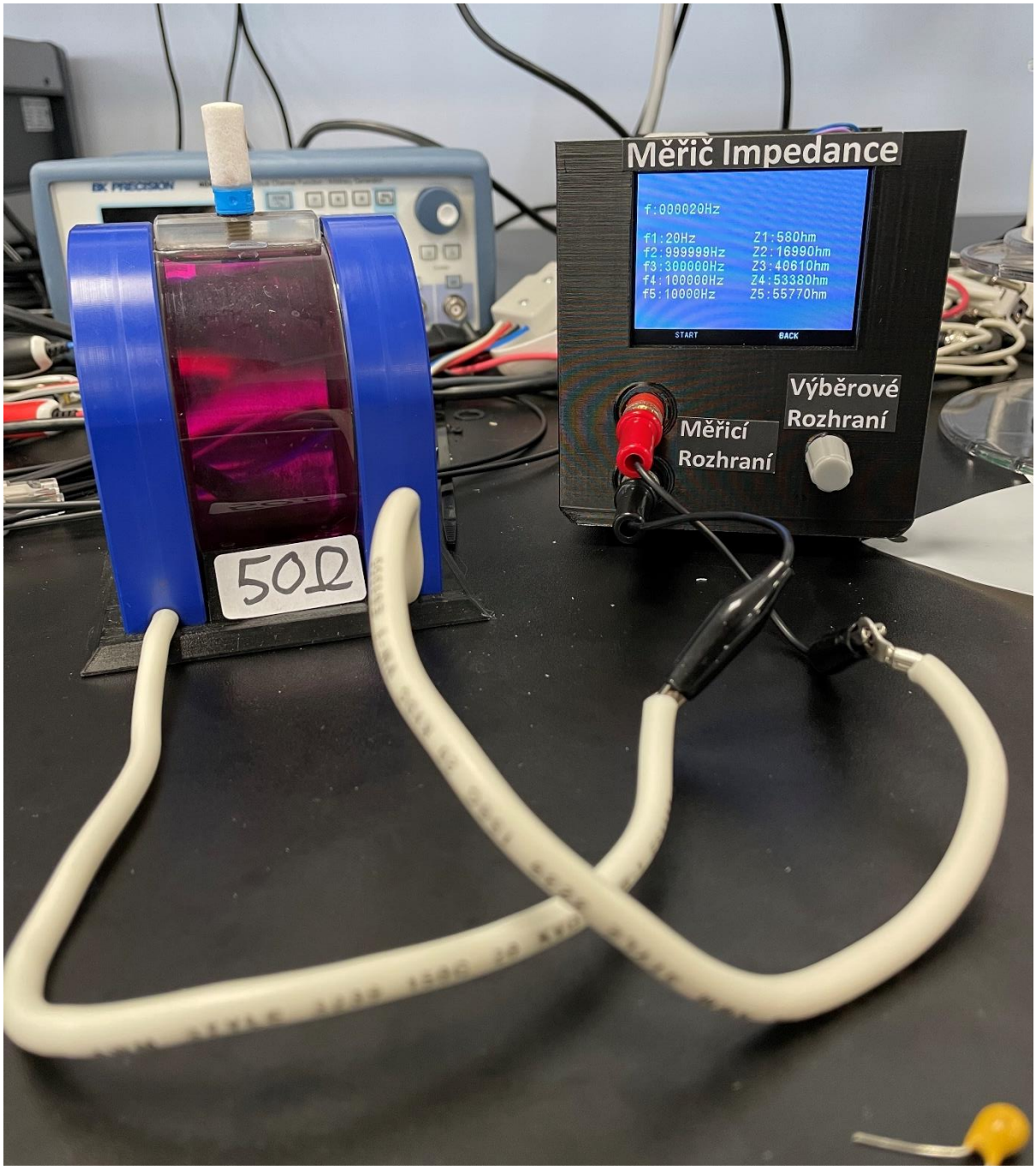

AEK.003
MEASURER
INTERFACE
OUT SIGNAL
VCC
GND
BOOT0


IMPEDANCE
MEASURER
VER.003
RELAYS BLOCK
DC5-DC±9 BLOCK
DC24-DC5
0V 24V
PEAK DETECTOR BLOCK
MLT BLOCK
AMP 9V BLOCK
HOR OFFSET BLOCK
MCU BLOCK
AMP 3V BLOCK
DAC BLOCK
SIGNAL OUT
SWDIO
SWCLK
GND
NRST

Příloha C –


2.8' TFT 240xRGBx320
ROHS
+V -V DC OK
24V/1A
DC OK
+V ADJ
428-433
100-240VAC
0.55A
50/60Hz
N L
IMPEDANCE MEASURER

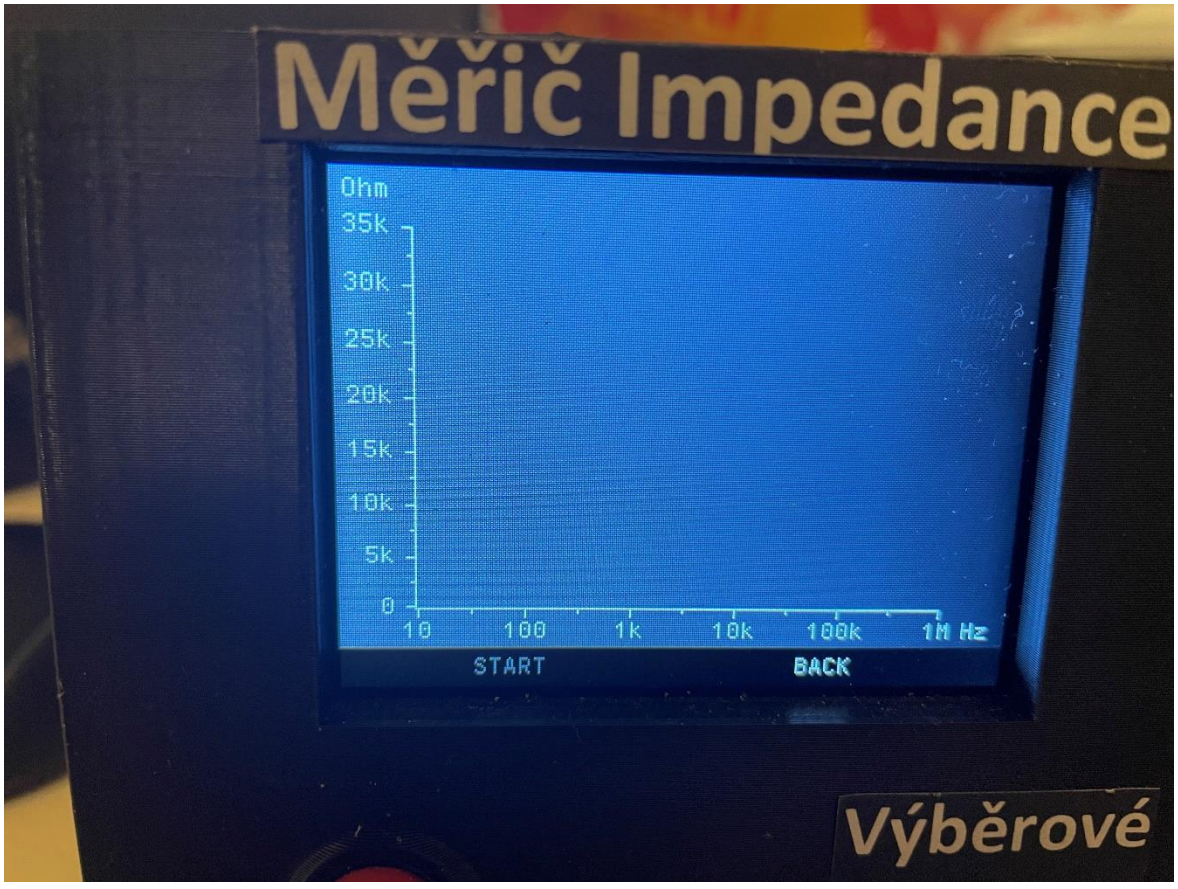

428-433
INPUT: 100-240VAC 0.55A
50/60Hz
OUTPUT: 24V ⎓ 1A
Model No. MDR-20-24
MADE BY MEAN WELL (TAIWAN)

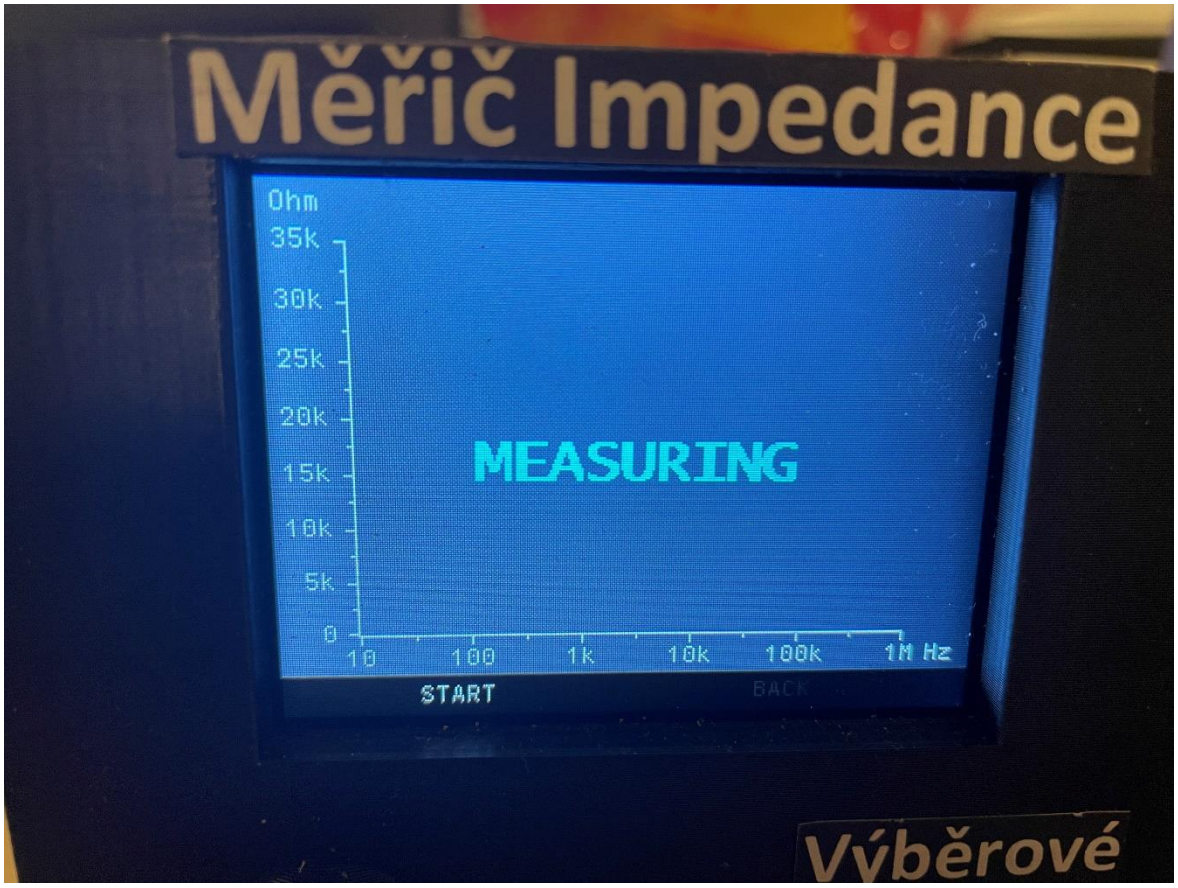

Měřič Impedance
f:000020Hz
f1:20Hz Z1:580hm
f2:999999Hz Z2:16990hm
f3:300000Hz Z3:40610hm
f4:100000Hz Z4:53380hm
f5:10000Hz Z5:55770hm
START
BACK
Výběrové Rozhraní
Měřicí Rozhraní
50Ω


Měřič Impedance
IMPEDANCE
ANALYZER/MEASURER
ANALYZER
MEASURER
Výběrové
Rozhraní
Měřicí


Měřič Impedance
f:000001Hz
f1:0Hz
f2:0Hz
f3:0Hz
f4:0Hz
f5:0Hz
Z1:00hm
Z2:00hm
Z3:00hm
Z4:00hm
Z5:00hm
START
BACK
Výběrové
Rozhraní


Měřič Impedance
f:000001Hz
MEASURING
START
BACK
Výběrové


Měřič Impedance
Ohm
35k
30k
25k
20k
15k
10k
5k
0
10
100
1k
10k
100k
1M Hz
START
BACK
Výběrové


Měřič Impedance
Ohm
35k
30k
25k
20k
15k
10k
5k
0
MEASURING
10
100
1k
10k
100k
1M Hz
START
BACK
Výběrové

Příloha D –

```
/* USER CODE BEGIN Header */
/**

******************************************************************************
*****
  * @file           : main.c
  * @brief          : Main program body

******************************************************************************
*****
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the
LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *

******************************************************************************
*****
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------
-----*/
#include "main.h"

/* Private includes ----------------------------------------------------------
-----*/
/* USER CODE BEGIN Includes */
#include <stdint.h>
#include <stdio.h>
#include <string.h>
#include <stdarg.h>

#include "ili9341.h" //LCD Dipslay Library
#include "fonts.h"

#include "AD9833.h" //Digital Analog Converter Library
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------
-----*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------
-----*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------
-----*/
/* USER CODE BEGIN PM */
```

```
/* USER CODE END PM */

/* Private variables ---------------------------------------------------------
-----*/
ADC_HandleTypeDef hadc1;

SPI_HandleTypeDef hspi1;

TIM_HandleTypeDef htim2;

/* USER CODE BEGIN PV */
int analyzerArray[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
int measurerArray[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
int arrayImp [] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
char tempString[16];
int prevEncoderPosition;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------
-----*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_SPI1_Init(void);
static void MX_TIM2_Init(void);
static void MX_ADC1_Init(void);
/* USER CODE BEGIN PFP */
void MainMenu(void);
void MeasurerMenu(void);
void AnalyzerMenu(void);

int EncoderDirection(void);
int ImpedanceOnFrequency(int);
void MultiplexerChannelSelect(int);
void ILIDrawLine(int, int, int, int);
void ILIDrawGraph(void);

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------
-----*/
/* USER CODE BEGIN 0 */

void MainMenu()
{
	int option = 1;
	int encoderDirection;

	ILI9341_FillScreen(ILI9341_BLACK);
	ILI9341_FillRectangle(0, 0, 320, 220, ILI9341_COLOR565(128, 128,
128));
	ILI9341_WriteString(88, 74, "IMPEDANCE", Font_16x26, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));
	ILI9341_WriteString(66, 107, "ANALYZER/MEASURER", Font_11x18,
ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));

	while(1)
	{
		encoderDirection = EncoderDirection();
		if(encoderDirection != 0)
```

```
                option = encoderDirection;

          if(option == -1)
                ILI9341_WriteString(47, 225, "ANALYZER", Font_7x10,
ILI9341_WHITE, ILI9341_BLACK);
          else
                ILI9341_WriteString(47, 225, "ANALYZER", Font_7x10,
ILI9341_COLOR565(128, 128, 128), ILI9341_BLACK);

          if(option == 1)
                ILI9341_WriteString(197, 225, "MEASURER", Font_7x10,
ILI9341_WHITE, ILI9341_BLACK);
          else
                ILI9341_WriteString(197, 225, "MEASURER", Font_7x10,
ILI9341_COLOR565(128, 128, 128), ILI9341_BLACK);

          if(!HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_4))
          {
                if(option == -1)
                {
                      ILI9341_WriteString(47, 225, "ANALYZER",
Font_7x10, ILI9341_COLOR565(128, 128, 128), ILI9341_BLACK);
                      ILI9341_WriteString(197, 225, "MEASURER",
Font_7x10, ILI9341_COLOR565(62, 62, 62),ILI9341_BLACK);
                      HAL_Delay(100);
                      ILI9341_WriteString(47, 225, "ANALYZER",
Font_7x10, ILI9341_WHITE, ILI9341_BLACK);
                      ILI9341_WriteString(197, 225, "MEASURER",
Font_7x10, ILI9341_COLOR565(62, 62, 62),ILI9341_BLACK);

                      AnalyzerMenu();
                }

                if(option == 1)
                {
                      ILI9341_WriteString(47, 225, "ANALYZER",
Font_7x10, ILI9341_COLOR565(62, 62, 62), ILI9341_BLACK);
                      ILI9341_WriteString(197, 225, "MEASURER",
Font_7x10, ILI9341_COLOR565(128, 128, 128),ILI9341_BLACK);
                      HAL_Delay(100);
                      ILI9341_WriteString(47, 225, "ANALYZER",
Font_7x10, ILI9341_COLOR565(62, 62, 62), ILI9341_BLACK);
                      ILI9341_WriteString(197, 225, "MEASURER",
Font_7x10, ILI9341_WHITE,ILI9341_BLACK);

                      MeasurerMenu();
                }

                ILI9341_FillScreen(ILI9341_BLACK);
                ILI9341_FillRectangle(0, 0, 320, 220,
ILI9341_COLOR565(128, 128, 128));
                ILI9341_WriteString(88, 74, "IMPEDANCE", Font_16x26,
ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));
                ILI9341_WriteString(66, 107, "ANALYZER/MEASURER",
Font_11x18, ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));
          }
      }
}

void AnalyzerMenu(void)
```

```
{
	int arrayFreq [] = {10, 31, 100, 316, 1000, 3162, 10000, 31622,
100000, 316228, 1000000};
	int arrayX [] = {36, 61, 86, 111, 136, 161, 186, 211, 236, 261,
286};
	int option = 1;
	int encoderDirection;

	ILI9341_FillScreen(ILI9341_BLACK);
	ILI9341_FillRectangle(0, 0, 320, 220, ILI9341_COLOR565(128, 128,
128));
	ILIDrawGraph();
	for(int i = 0; i < 10; i++)
		ILIDrawLine(arrayX[i], arrayImp[i], arrayX[i+1],
arrayImp[i+1]);

	while(1)
	{
		encoderDirection = EncoderDirection();
		if(encoderDirection != 0)
			option = encoderDirection;

		if(option == -1)
			ILI9341_WriteString(62, 225, "START", Font_7x10,
ILI9341_WHITE, ILI9341_BLACK);
		else
			ILI9341_WriteString(62, 225, "START", Font_7x10,
ILI9341_COLOR565(128, 128, 128), ILI9341_BLACK);

		if(option == 1)
			ILI9341_WriteString(217, 225, "BACK", Font_7x10,
ILI9341_WHITE, ILI9341_BLACK);
		else
			ILI9341_WriteString(217, 225, "BACK", Font_7x10,
ILI9341_COLOR565(128, 128, 128), ILI9341_BLACK);

		if(!HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_4))
		{
			if(option == -1)
			{
				ILI9341_WriteString(62, 225, "START", Font_7x10,
ILI9341_COLOR565(128, 128, 128), ILI9341_BLACK);
				ILI9341_WriteString(217, 225, "BACK", Font_7x10,
ILI9341_COLOR565(62, 62, 62),ILI9341_BLACK);
				HAL_Delay(100);
				ILI9341_WriteString(62, 225, "START", Font_7x10,
ILI9341_WHITE, ILI9341_BLACK);
				ILI9341_WriteString(217, 225, "BACK", Font_7x10,
ILI9341_COLOR565(62, 62, 62),ILI9341_BLACK);


				ILI9341_FillRectangle(36, 0, 320, 200,
ILI9341_COLOR565(128, 128, 128));
				ILIDrawGraph();
				for(int i = 0; i < 11; i++)
					arrayImp[i] =
ImpedanceOnFrequency(arrayFreq[i]);
				for(int i = 0; i < 10; i++)
					ILIDrawLine(arrayX[i], arrayImp[i],
arrayX[i+1], arrayImp[i+1]);
```

```
                  }

                  if(option == 1)
                  {
                        ILI9341_WriteString(62, 225, "START", Font_7x10,
ILI9341_COLOR565(62, 62, 62), ILI9341_BLACK);
                        ILI9341_WriteString(217, 225, "BACK", Font_7x10,
ILI9341_COLOR565(128, 128, 128),ILI9341_BLACK);
                        HAL_Delay(100);
                        ILI9341_WriteString(62, 225, "START", Font_7x10,
ILI9341_COLOR565(62, 62, 62), ILI9341_BLACK);
                        ILI9341_WriteString(217, 225, "BACK", Font_7x10,
ILI9341_WHITE,ILI9341_BLACK);
                        return;
                  }
            }
      }
}

void MeasurerMenu(void)
{
      int option = 1;
      int tempForArray;
      int measurerFrequency = 1;
      int encoderDirection;
      int freqSelectIndex = 5;
      int freqSelectArray [] = {0, 0, 0, 0, 0, 1};

      ILI9341_FillScreen(ILI9341_BLACK);
      ILI9341_FillRectangle(0, 0, 320, 220, ILI9341_COLOR565(128, 128,
128));
      sprintf(tempString,"f:%d%d%d%d%d%dHz",freqSelectArray[0],freqSelect
Array[1],freqSelectArray[2],freqSelectArray[3],freqSelectArray[4],freqSel
ectArray[5]);
      ILI9341_WriteString(10, 40, tempString, Font_11x18, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));

      for(int i = 0; i < 5; i++)
      {
            sprintf(tempString,"f%d:%dHz      ", i+1,
measurerArray[i+i]);
            ILI9341_WriteString(10, 80+i*20, tempString, Font_11x18,
ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));
            sprintf(tempString,"Z%d:%dOhm", i+1,measurerArray[i+i+1]);
            ILI9341_WriteString(170, 80+i*20, "          ", Font_11x18,
ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));
            ILI9341_WriteString(170, 80+i*20, tempString, Font_11x18,
ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));
      }

      while(1)
      {
            encoderDirection = EncoderDirection();
            if(encoderDirection != 0)
                  option = encoderDirection;

            if(option == -1)
                  ILI9341_WriteString(62, 225, "START", Font_7x10,
ILI9341_WHITE, ILI9341_BLACK);
            else
```

```
                  ILI9341_WriteString(62, 225, "START", Font_7x10,
ILI9341_COLOR565(128, 128, 128), ILI9341_BLACK);

            if(option == 1)
                  ILI9341_WriteString(217, 225, "BACK", Font_7x10,
ILI9341_WHITE, ILI9341_BLACK);
            else
                  ILI9341_WriteString(217, 225, "BACK", Font_7x10,
ILI9341_COLOR565(128, 128, 128), ILI9341_BLACK);

            if(!HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_4))
            {
                  HAL_Delay(200);

                  if(option == -1)
                  {
                        ILI9341_WriteString(62, 225, "START", Font_7x10,
ILI9341_COLOR565(128, 128, 128), ILI9341_BLACK);
                        ILI9341_WriteString(217, 225, "BACK", Font_7x10,
ILI9341_COLOR565(62, 62, 62),ILI9341_BLACK);
                        HAL_Delay(100);
                        ILI9341_WriteString(62, 225, "START", Font_7x10,
ILI9341_WHITE, ILI9341_BLACK);
                        ILI9341_WriteString(217, 225, "BACK", Font_7x10,
ILI9341_COLOR565(62, 62, 62),ILI9341_BLACK);

                        while(1)
                        {
                              encoderDirection = EncoderDirection();

                              freqSelectArray[freqSelectIndex] +=
encoderDirection;

                              if (freqSelectArray[freqSelectIndex] < 0)
                                    freqSelectArray[freqSelectIndex] = 0;
                              if (freqSelectArray[freqSelectIndex] > 9)
                                    freqSelectArray[freqSelectIndex] = 9;


      sprintf(tempString,"f:%d%d%d%d%d%dHz",freqSelectArray[0],freqSelect
Array[1],freqSelectArray[2],freqSelectArray[3],freqSelectArray[4],freqSel
ectArray[5]);
                              ILI9341_WriteString(43 + freqSelectIndex *
11, 20, " ", Font_11x18, ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));
                              ILI9341_WriteString(32 + freqSelectIndex *
11, 20, "_", Font_11x18, ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));
                              ILI9341_WriteString(10, 40, tempString,
Font_11x18, ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));


                              if(!HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_4))
                              {
                                    HAL_Delay(200);
                                    freqSelectIndex--;
                                    if (freqSelectIndex < 0)
                                          break;

                              }
                        }
```

```
                        ILI9341_WriteString(43 + freqSelectIndex * 11,
20, " ", Font_11x18, ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));
                        freqSelectIndex = 5;
                        ILI9341_WriteString(42, 20, " ", Font_11x18,
ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));
                        measurerFrequency = freqSelectArray[0] * 100000 +
freqSelectArray[1] * 10000 + freqSelectArray[2] * 1000 +
freqSelectArray[3] * 100 + freqSelectArray[4] * 10 + freqSelectArray[5] *
1;

                        for(int i = 7; i >= 0; i--)
                        {
                              tempForArray = measurerArray[i];
                              measurerArray[i+2] = tempForArray;
                        }

                        measurerArray[0] = measurerFrequency;
                        measurerArray[1] =
ImpedanceOnFrequency(measurerFrequency);

                        for(int i = 0; i < 5; i++)
                        {
                              sprintf(tempString,"f%d:%dHz      ", i+1,
measurerArray[i+i]);
                              ILI9341_WriteString(10, 80+i*20,
tempString, Font_11x18, ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));
                              sprintf(tempString,"Z%d:%dOhm",
i+1,measurerArray[i+i+1]);
                              ILI9341_WriteString(170, 80+i*20, "
", Font_11x18, ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));
                              ILI9341_WriteString(170, 80+i*20,
tempString, Font_11x18, ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));
                        }


                  }

                  if(option == 1)
                  {
                        ILI9341_WriteString(62, 225, "START", Font_7x10,
ILI9341_COLOR565(62, 62, 62), ILI9341_BLACK);
                        ILI9341_WriteString(217, 225, "BACK", Font_7x10,
ILI9341_COLOR565(128, 128, 128),ILI9341_BLACK);
                        HAL_Delay(100);
                        ILI9341_WriteString(62, 225, "START", Font_7x10,
ILI9341_COLOR565(62, 62, 62), ILI9341_BLACK);
                        ILI9341_WriteString(217, 225, "BACK", Font_7x10,
ILI9341_WHITE,ILI9341_BLACK);
                        return;
                  }
            }
      }
}

int ImpedanceOnFrequency(int frequency)
{
      const float refVoltage = 3.29;
      const int multiplexerResistance [] = {10, 33, 100, 330, 1000, 3300,
10000, 33000};
```

```
	const float targetVoltage [] = {1, 1.98, 2.21, 2.32, 2.29, 2.11,
1.93, 1.42};
	float impedance;
	int adcValue;
	float voltage;
	float prevVoltage = refVoltage;
	int multiplexerCS = 7;
	int multiplexerCount = 7;

	AD9833_Init(SIN, frequency, 0);

	MultiplexerChannelSelect(multiplexerCS);
	HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, 1); //Enable Multuplexer
	HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 1); //Open Reles

	while(multiplexerCount >= 0)
	{
		HAL_Delay(2000);

		HAL_ADC_Start(&hadc1);
		HAL_ADC_PollForConversion(&hadc1, 100);
		adcValue = HAL_ADC_GetValue(&hadc1);
		HAL_ADC_Stop(&hadc1);
		voltage = (adcValue/4095.0)*refVoltage;

		impedance = (multiplexerResistance[multiplexerCS] *
voltage)/((targetVoltage[multiplexerCS]*2) - voltage);

		if(voltage <= targetVoltage[multiplexerCS])
		{
			multiplexerCS--;
			multiplexerCount--;
			multiplexerCS = (multiplexerCS < 1) ? 1 :
multiplexerCS;
			prevVoltage = voltage;
			MultiplexerChannelSelect(multiplexerCS);
		}
		else
		{
			if(multiplexerCS != 7)
				if((voltage - targetVoltage[multiplexerCS]) >=
(targetVoltage[multiplexerCS] - prevVoltage))
				{
					multiplexerCS++;
					voltage = prevVoltage;
				}

			break;
		}
	}

	impedance =  (multiplexerResistance[multiplexerCS] *
voltage)/((targetVoltage[multiplexerCS]*2) - voltage);

	HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, 0); //Close Reles
	HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, 0); //Enable Multuplexer
	HAL_Delay(1000);
	return impedance;
}
```

```
void MultiplexerChannelSelect(int cs)
{
      switch(cs)
          {
              case 0:
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 0);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, 0);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);
                  break;
              case 1:
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 1);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, 0);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);
                  break;
              case 2:
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 0);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, 1);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);
                  break;
              case 3:
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 1);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, 1);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 0);
                  break;
              case 4:
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 0);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, 0);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 1);
                  break;
              case 5:
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 1);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, 0);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 1);
                  break;
              case 6:
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 0);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, 1);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 1);
                  break;
              case 7:
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, 1);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, 1);
                  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, 1);
                  break;
          }
}

int EncoderDirection(void)
{
      int encoderDirection;
      int encoderPosition;
      encoderPosition = TIM2 -> CNT;
      if((encoderPosition % 2 == 0) && (encoderPosition >
prevEncoderPosition))
            encoderDirection = (encoderPosition - prevEncoderPosition >
250) ? -1 : 1;
      else if((encoderPosition % 2 == 0) && (encoderPosition <
prevEncoderPosition))
            encoderDirection = (prevEncoderPosition - encoderPosition >
250) ? 1 : -1;
```

```
	else
		encoderDirection = 0;
	prevEncoderPosition = encoderPosition;

	//sprintf(tempString,"EncPos:%d    ", encoderPosition);
	//ILI9341_WriteString(20, 20, tempString, Font_11x18,
ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));

	return encoderDirection;
}

void ILIDrawGraph(void)
{
	int arrayX [] = {36, 61, 86, 111, 136, 161, 186, 211, 236, 261,
286};
	int arrayY [] = {25, 37, 50, 62, 75, 87, 100, 112, 125, 137, 150,
162, 175, 187, 199};

	for(int x = 35; x < 287; x++)
		ILI9341_DrawPixel(x, 200, ILI9341_WHITE);
	for(int y = 25; y < 201; y++)
		ILI9341_DrawPixel(35, y, ILI9341_WHITE);

	ILI9341_WriteString(36 - 7, 207, "10", Font_7x10, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));
	ILI9341_WriteString(86 - 10, 207, "100", Font_7x10, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));
	ILI9341_WriteString(136 - 7, 207, "1k", Font_7x10, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));
	ILI9341_WriteString(186 - 10, 207, "10k", Font_7x10, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));
	ILI9341_WriteString(236 - 14, 207, "100k", Font_7x10,
ILI9341_WHITE, ILI9341_COLOR565(128, 128, 128));
	ILI9341_WriteString(286 - 7, 207, "1M", Font_7x10, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));
	ILI9341_WriteString(305 - 7, 207, "Hz", Font_7x10, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));

	ILI9341_WriteString(5, 5, "Ohm", Font_7x10, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));
	ILI9341_WriteString(5, 20, "35k", Font_7x10, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));
	ILI9341_WriteString(5, 45, "30k", Font_7x10, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));
	ILI9341_WriteString(5, 70, "25k", Font_7x10, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));
	ILI9341_WriteString(5, 95, "20k", Font_7x10, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));
	ILI9341_WriteString(5, 120, "15k", Font_7x10, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));
	ILI9341_WriteString(5, 145, "10k", Font_7x10, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));
	ILI9341_WriteString(5, 170, " 5k", Font_7x10, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));
	ILI9341_WriteString(5, 195, "  0", Font_7x10, ILI9341_WHITE,
ILI9341_COLOR565(128, 128, 128));

	for(int x = 0; x < 11; x++)
	{
		ILI9341_DrawPixel(arrayX[x], 201, ILI9341_WHITE);
```

```
            ILI9341_DrawPixel(arrayX[x], 202, ILI9341_WHITE);
            if(x%2==0)
            {
                  ILI9341_DrawPixel(arrayX[x], 203, ILI9341_WHITE);
                  ILI9341_DrawPixel(arrayX[x], 204, ILI9341_WHITE);
            }
      }
      for(int y = 0; y < 15; y++)
      {
            ILI9341_DrawPixel(34, arrayY[y], ILI9341_WHITE);
            ILI9341_DrawPixel(33, arrayY[y], ILI9341_WHITE);
            if(y%2==0)
            {
                  ILI9341_DrawPixel(32, arrayY[y], ILI9341_WHITE);
                  ILI9341_DrawPixel(31, arrayY[y], ILI9341_WHITE);
            }
      }
}

void ILIDrawLine(int x1, int y1, int x2, int y2)
{
      float a;
      float b;
      float fy1;
      float fy2;

      fy1 = y1 * 0.005;
      fy2 = y2 * 0.005;

      b = (fy2 - fy1)/(x2 - x1);
      a = fy1 - b * x1;

      for(;x1 < x2; x1++)
      {
            fy1 = a + b * x1;
            y1 = (int)fy1;
            ILI9341_DrawPixel(x1, 200 - y1, ILI9341_WHITE);
      }
}

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  prevEncoderPosition = TIM2 -> CNT; //The start of encoder position
  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------
-----*/

  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */
```

```
  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_SPI1_Init();
  MX_TIM2_Init();
  MX_ADC1_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_Encoder_Start(&htim2, TIM_CHANNEL_ALL);
  ILI9341_Unselect();
  ILI9341_Init();
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
        MainMenu();
    /* USER CODE END WHILE */
  }

    /* USER CODE BEGIN 3 */

  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};

  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
```

```
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_ADC;
  PeriphClkInit.AdcClockSelection = RCC_ADCPCLK2_DIV6;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
  HAL_RCC_MCOConfig(RCC_MCO, RCC_MCO1SOURCE_PLLCLK, RCC_MCODIV_1);
}

/**
  * @brief ADC1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC1_Init(void)
{

  /* USER CODE BEGIN ADC1_Init 0 */

  /* USER CODE END ADC1_Init 0 */

  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC1_Init 1 */

  /* USER CODE END ADC1_Init 1 */

  /** Common config
  */
  hadc1.Instance = ADC1;
  hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc1.Init.ContinuousConvMode = DISABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    Error_Handler();
  }

  /** Configure Regular Channel
  */
  sConfig.Channel = ADC_CHANNEL_6;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
```

```
  {
    Error_Handler();
  }
  /* USER CODE BEGIN ADC1_Init 2 */

  /* USER CODE END ADC1_Init 2 */

}

/**
  * @brief SPI1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI1_Init(void)
{

  /* USER CODE BEGIN SPI1_Init 0 */

  /* USER CODE END SPI1_Init 0 */

  /* USER CODE BEGIN SPI1_Init 1 */

  /* USER CODE END SPI1_Init 1 */
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_1LINE;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_32;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI1_Init 2 */

  /* USER CODE END SPI1_Init 2 */

}

/**
  * @brief TIM2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM2_Init(void)
{

  /* USER CODE BEGIN TIM2_Init 0 */

  /* USER CODE END TIM2_Init 0 */

  TIM_Encoder_InitTypeDef sConfig = {0};
```

```
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM2_Init 1 */

  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 0;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 255;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  sConfig.EncoderMode = TIM_ENCODERMODE_TI1;
  sConfig.IC1Polarity = TIM_ICPOLARITY_RISING;
  sConfig.IC1Selection = TIM_ICSELECTION_DIRECTTI;
  sConfig.IC1Prescaler = TIM_ICPSC_DIV1;
  sConfig.IC1Filter = 0;
  sConfig.IC2Polarity = TIM_ICPOLARITY_RISING;
  sConfig.IC2Selection = TIM_ICSELECTION_DIRECTTI;
  sConfig.IC2Prescaler = TIM_ICPSC_DIV1;
  sConfig.IC2Filter = 0;
  if (HAL_TIM_Encoder_Init(&htim2, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) !=
HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */

  /* USER CODE END TIM2_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_10|GPIO_PIN_12
                          |GPIO_PIN_13|GPIO_PIN_15|GPIO_PIN_4|GPIO_PIN_5
                          |GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_8,
GPIO_PIN_RESET);

  /*Configure GPIO pin : PA4 */
```

```
  GPIO_InitStruct.Pin = GPIO_PIN_4;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : PB0 PB1 PB10 PB12
                           PB13 PB15 PB4 PB5
                           PB6 PB7 PB8 */
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_10|GPIO_PIN_12
                        |GPIO_PIN_13|GPIO_PIN_15|GPIO_PIN_4|GPIO_PIN_5
                        |GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_8;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : PA8 */
  GPIO_InitStruct.Pin = GPIO_PIN_8;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line
number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line
number,
```

```
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```