ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

Katedra aplikované elektroniky a telekomunikací

BAKALÁŘSKÁ PRÁCE

Návrh měřící karty pro měření s termočlánkem

Martin Šiman **2015**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: 2014/2015

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	Martin ŠIMAN
Osobní číslo:	E12B0375P
Studijní program:	B2612 Elektrotechnika a informatika
Studijní obor:	Elektronika a telekomunikace
Název tématu:	Návrh měřicí karty pro měření s termočlánkem
Zadávající katedra:	Katedra aplikované elektroniky a telekomunikací

Zásady pro vypracování:

1. Prostudujte problematiku termočlánků, jejich pracovní parametry a možnosti zp
vání výstupního signálu.
2. Navrhněte vstupní část měřicí karty pro měření s termočlánky.
3. S navrženou kartou proveďtě měření.
4. Zhodnoťte výsledky měření.

Rozsah grafických prací:	podle doporučení vedoucího
Rozsah pracovní zprávy:	20 - 30 stran
Forma zpracování bakalářské práce:	tištěná/elektronická
Seznam odborné literatury:	

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí bakalářské práce:	Ing. Karel Slobodník Katedra teoretické elektrotechniky
Datum zadání bakalářské práce:	15. října 2014
Termín odevzdání bakalářské práce:	8. června 2015

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 15. října 2014

Abstrakt

Předkládaná bakalářská práce je zaměřena na návrh systému pro měření několika termočlánků. Zabývá se problematikou termočlánků, na jakém způsobu fungují a jejich dělení z hlediska provozních parametrů. Jaké jsou možnosti zpracování výstupního signálu termočlánku. To znamená návrh zesilovače. Jedna z částí je věnována návrhu filtru, ochranných zařízení a zásady pro návrh DPS. Srdcem všeho je mikroprocesor, ten je potřeba naprogramovat v programovacím jazyku C. Je potřeba data ukládat a vizualizovat. Dále je potřeba řídit někde danou činnost a provést případnou kompenzaci k tomu je využito Visual Studio za pomocí programovacího jazyka C#. Dosažené výsledky měření odpovídají reálným hodnotám.

Klíčová slova

Termočlánek, Operační zesilovač, Filtr, LABJACK, Multiplexor, Mikroprocesor, Programovací jazyk C, Programovací jazyk C# .NET, Framework 4.5.2, U3-HV, Visual Studio 2013, AD595

Abstract

The bachelor thesis is focused on designing a system for measuring several thermocouples. It deals with the issues of thermocouples on which way they work and their classification in terms of operational parameters. What are the options for processing the output signal of the thermocouple. That means the design of the amplifier. Part is devoted to the design of the filter, protective equipment and guidelines for PCB design. The microprocessor is the heart of everything; it is needed to program in C programming language. It is necessary to store and visualize data. It also needs to drive somewhere to perform the operation and possible compensation is used to using Visual Studio C# programming language. Achieved results of temperature measurements are to maximally accurate.

Key words

Thermocouple, Operational amplifier, Filter, LABJACK, Multiplexer, Microprocessor, Programming language C, Programming language C# .NET, Framework 4.5.2, U3-HV, Visual Studio 2013, AD595

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této bakalářské práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

.....

podpis

V Plzni dne 7.6.2015 Martin Šiman

Poděkování

Tímto bych rád poděkoval vedoucímu bakalářské práce Ing. Karlu Slobodníkovi a panu konzultantu bakalářské práce Ing. Davidu Pánkovi, Ph.D. za cenné profesionální rady, připomínky a metodické vedení práce.

Obsah

Obsah

Úvod

Tato práce je zaměřená na návrhu měřící karty pro měření s termočlánkem. Úkolem je dosáhnout řešení, kdy měření bude dosahovat až deset různých teplot za jednu vteřinu z hlediska deseti různě umístěných termočlánků na měřeném objektu. Kde pracovní rozsah teplot se bude pohybovat mezi 200-700°C. K měření hodnot je použito měřící zařízení U3-HV od firmy LABJACK.

Text je rozdělen do několika částí. První částí je seznámení s problematikou termočlánků, na jakém principu fungují tyto snímače teploty. Jejich dělení podle použitého materiálu, pracovní parametry v závislosti na daném materiálu a možností zpracování výstupního signálu. Druhá část textu je věnováno stručné seznámení s měřícím zařízení U3-HV. Třetí část textu je zaměřena na ucelený návrh celého systému a popis funkcí použitých součástek z hlediska měřící, řídící, ochranné a filtrační části. Čtvrtá část textu vysvětluje řízení mikroprocesoru pro přepínání signálů, algoritmy použité pro komunikaci a sběr dat mezi počítačem, kartou a mikroprocesorem pro následnou vizualizaci a uložení hodnot do počítače. Poslední částí této práce je měření, kalibrace a zhodnocení výsledků.

Seznam symbolů a zkratek

RISC ..... Reduced instruction set computer

PWM …… Pulzně šířková modulace

AD…… Analogově číslicový

DPS…….. Destička plošných spojů

SPI …… Serial Peripheral Interface

PC……. Počítač

I2C…… Inter Integrated Circuit

USB……..Universal Serial Port

CSV……..Hodnoty oddělené čárkami

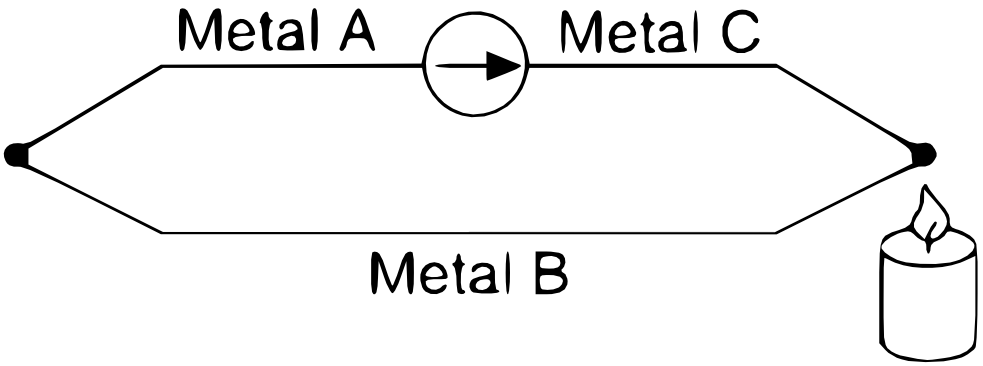
ms……..milisekunda

RC……..rezistor a kondenzátor

1 Termočlánek

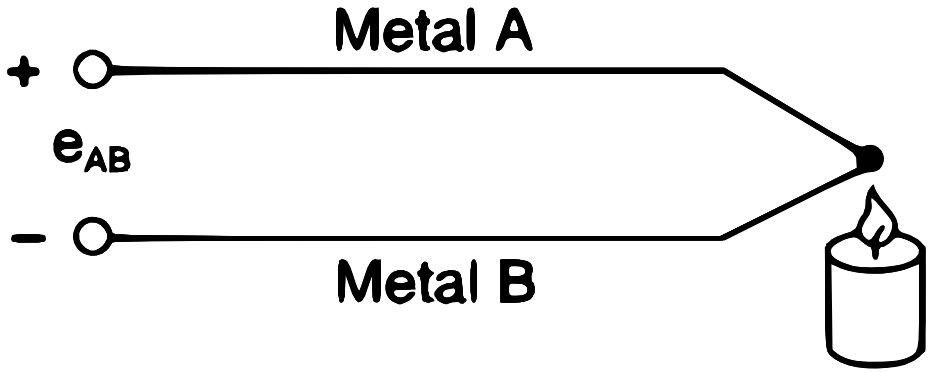
1.1 Princip [5]

Pokud dva vodiče z různorodého materiálu jsou na obou koncích spojeny a jeden konec začne být zahřívan. Začne procházet obvodem konstantní proud (obr. 1.1). Tento objev náleží Thomasu Seebeckovi. [1]



Obr. 1.1 *Seebeckův jev [1]*

Jestliže je tento obvod přerušen ve středu, tak jsou právě volné vodiče funkcí teploty a rozdílnosti kovů zahřívaného spoje (obr. 1.2). Seebeckovo napětí je označeno jako e_{AB}. [1]



Obr. 1.2 Seebeckovo napětí [1]

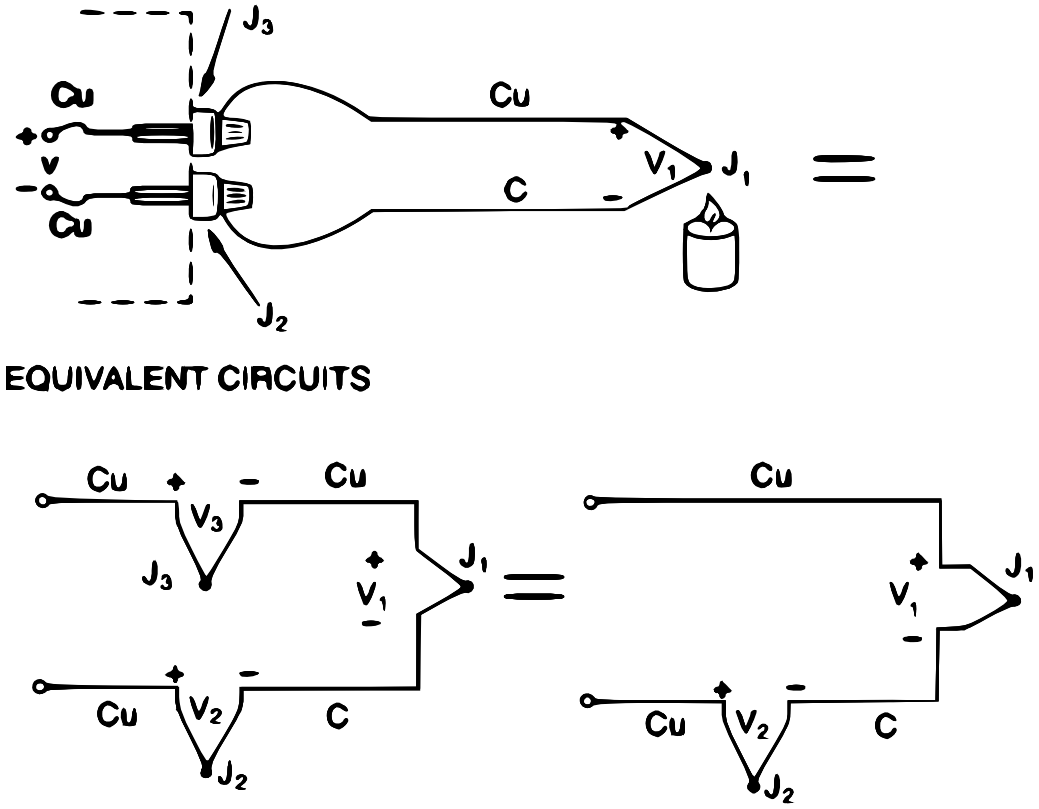
Pro všechny kovy platí tento jev. Nejběžnějšími kombinacemi kovů jsou uvedeny v tabulce (tab. 1.1). Pro malé změny teplot platí následující vztah 1.1:

$$\Delta e_{AB} = \alpha \Delta T \tag{1.1}$$

Kde α je Seebeckův koeficient a ΔT je změna teploty. [1]

1.2 Měření napětí termočlánku

Nelze měřit Seebeckovo napětí přímo, protože je potřeba nejdříve připojit voltmetr k termočlánku a to vede k vytvoření nového termoelektrického obvodu (obr. 1.3). Pro zjednodušení uvažujeme termočlánek typu T. [1]

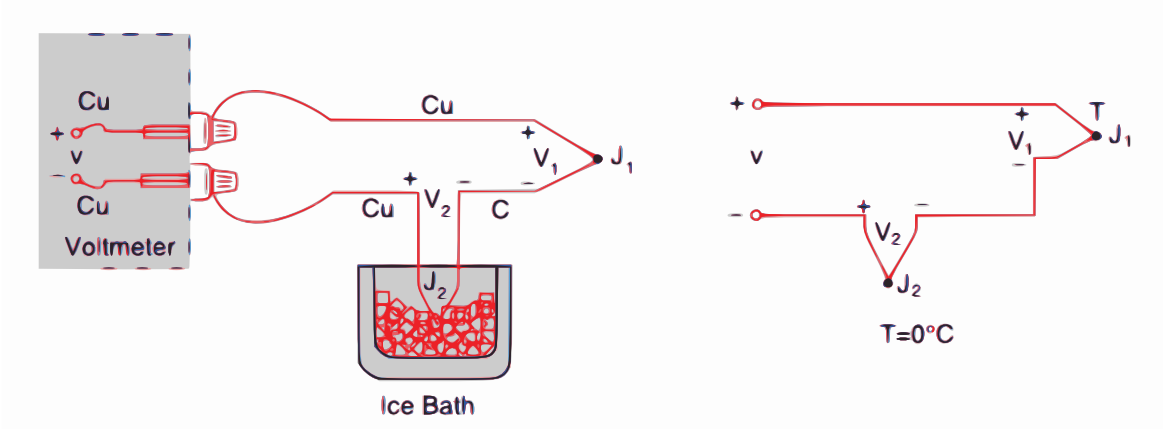


Obr. 1.3 Připojení voltmetru k termočlánku (typ T) [1]

Termočlánek typu T je složen z mědi Cu a sloučeniny niklu a mědi NiCu. Při spoji J3 (přechod z Cu na Cu) nevzniká žádný rozdíl potenciálu, proto lze spoj J3 zanedbat (obr. 1.3). Spoj J2 už zanedbat nelze, protože se skládá ze dvou materiálů. Výstupní napětí bude proto mezi spojem J1 a J2. To znamená, že nelze najít skutečnou teplotu, dokud není známa teplota ve spoji J2. [1]

1.3 Referenční spoj

Jedna z cest, jak eliminovat vliv spoje J2 je udržovat jeho teplotu na 0°C (obr. 1.4). [1]



Obr. 1.4 Eliminování spoje J2

Nyní pro obrázek 1.4 platí následující vztahy 1.2, 1.3, 1.4:

$$V = (V1 - V2) \cong \propto (t_{J1} - t_{J2}) \tag{1.2}$$

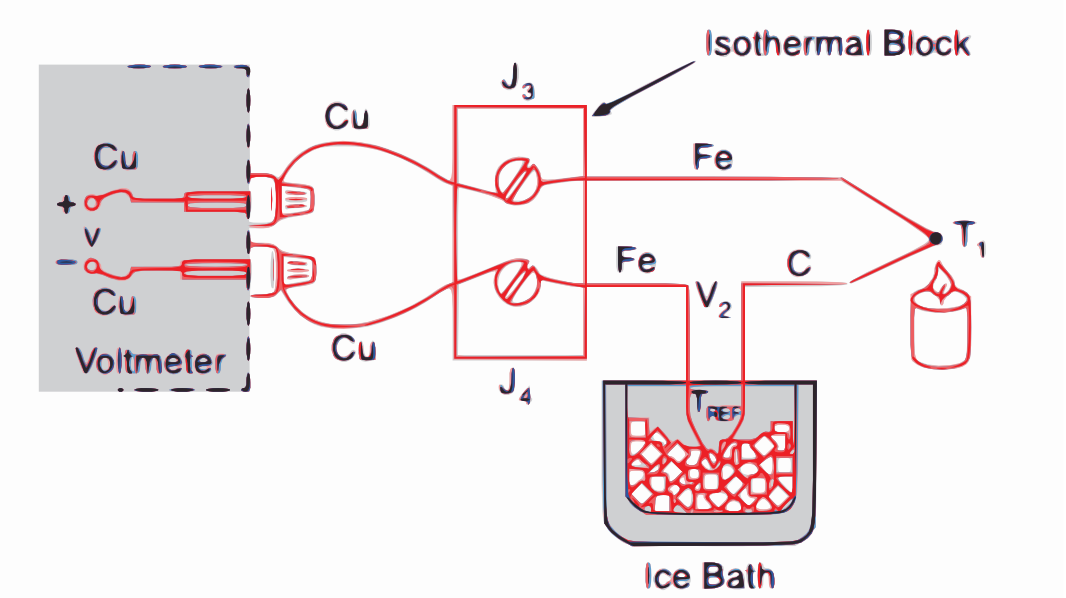
$$T_{J1}\,(°C) + 273.15 = t_{J1} \tag{1.3}$$

$$V = V_1 - V_2 = \propto \left[(T_{J1} + 273.15) - (T_{J1} + 273.15)\right] = \propto (T_{J1} - T_{J2}) = \propto (T_{J1} - 0) \tag{1.4}$$

Výstupní napětí je dáno jako rozdílem napětí V1 a V2 vztah 1.2. Ten je úměrný rozdílu teplot ve spoji J1 a J2. Vztah 1.3 ukazuje na linearitu mezi Kelvinovo a Celsiovo stupnicí. Po dosazení a uvědomění, že spoj J2 je udržován na 0°C. Je právě změna teploty úměrná změně napětí (rovnice 1.4). Tato metoda je velmi přesná, protože bod ledu může být precizně řízen.

Doposud byl případ vysvětlován na termočlánku typu J, kde je určitá výhoda se zanedbáním jednoho přechodu, protože dále pracujeme s měděnými (Cu) vodiči. Při

použití termočlánku typu J (Fe-C) už je nutno použít izotermální blok. Izotermální blok je dobrý elektrický izolátor a dobrý vodič tepla. Má za úkol udržovat spoj J3 a J4 na stejné teplotě. Když jsou přivedeny na voltmetr vodiče z rozdílných materiálů, je vnášena chyba do měření, jak už je výše ukázáno. [1] Na obrázku 1.5 je schematicky znázorněná výše popsaná situace.

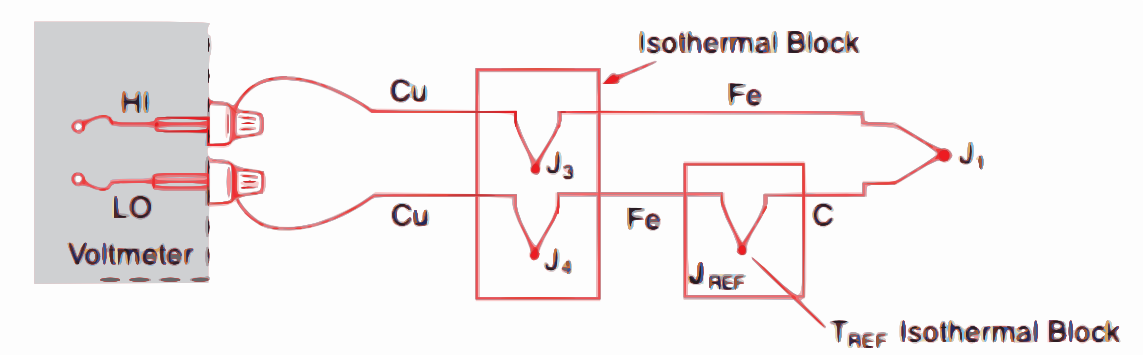


Obr. 1.5 Odstranění vzniku termočlánku na kontaktu voltmetru

Platí následující vztah:

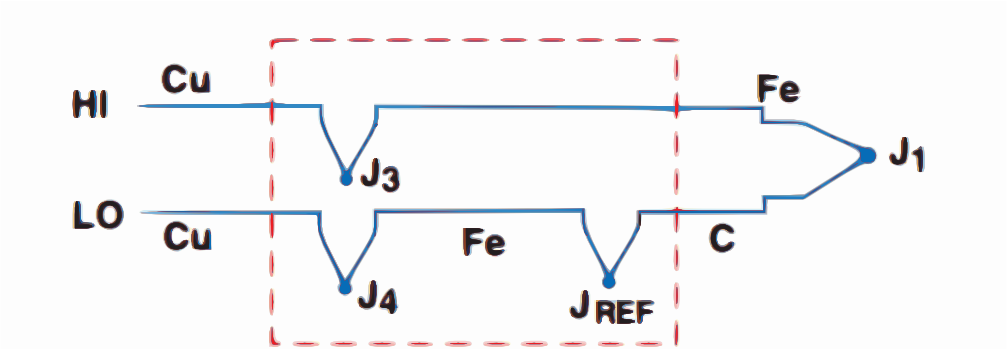
$$V = \propto (T_1 - T_{ref}) \tag{1.5}$$

Pro využití v praxi je nežádoucí se starat o tzv. „Ice Bath“ anglický termín pro ledovou lázeň, protože je to nepraktické. Proto led je nahrazen dalším izotermálním blokem viz obr.1.6 . Stále platí vztah 1.5



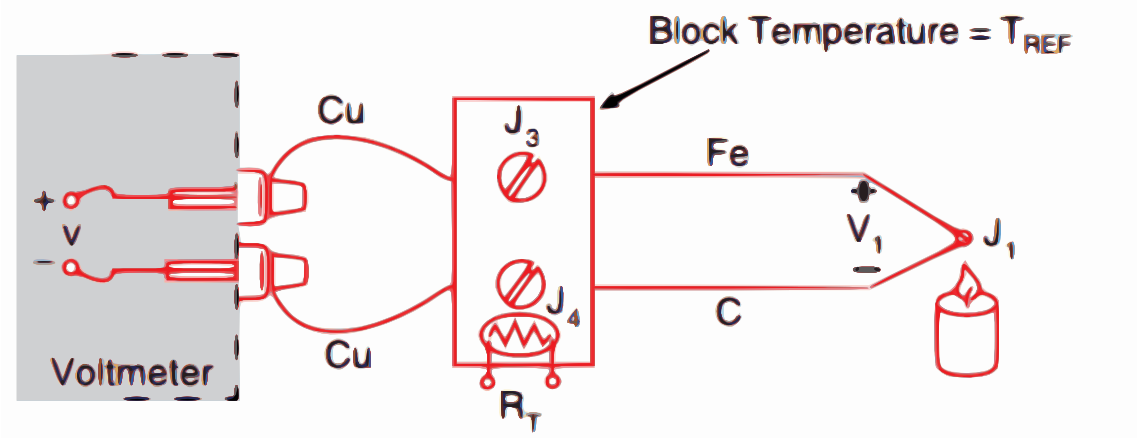
Obr. 1.6 Odstranění ledové lázně [1]

Spojením dvou izotermálních bloků v jeden nebude mít vliv na změnu vztahu 1.5 [1]



Obr. 1.7 Spojení v izotermálních bloků [1]

Dalším krokem vede k optimalizaci použití termočlánků. Pokud je to izotermální blok a teploty jsou všude stejné. Logickým závěrem je redundance vodiče Fe mezi J4 a JREF. V anglické literatuře se to značí jako zákon o „Intermediate Metals“. Výsledným zapojením je na obrázku 1.8. Kde k určení referenční teploty je použit odporový snímač teploty neboli termistor. Dále lze pomocí softwarové kompenzace zjistit skutečnou teplotu. [1]



Obr. 1.8 Výsledné zapojení s izotermální svorkovnicí a odporovým snímačem teploty [1]

Termistor Rt poskytuje cestu k měření absolutní teploty referenčního spoje. Pomocí softwaru je využit následující algoritmus. Je změřeno napětí Rt pro nalezení teploty Tref a pomocí teploty Tref lze dopočítat napětí Vref. Pokračujeme změřením napětí V. Napětí V je přičteno k napětí Vref k nalezení V1. V1 je přepočteno na teplotu TJ1. TJ1 je měřená (skutečná) teplota. [1]

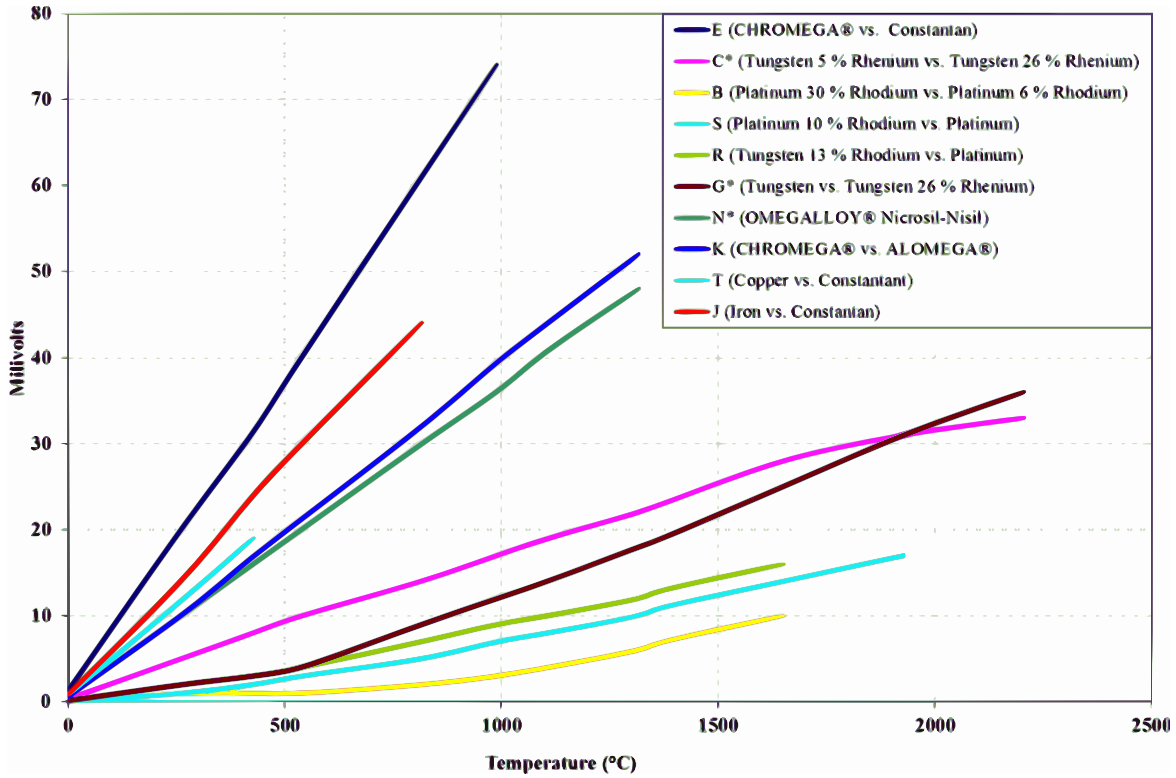
Otázkou je proč se zabývat termočlánky, když již existují odporové snímače teploty? Je to z toho důvodu, že každý teplotní snímač je vhodný do určitého prostředí, rozhoduje i cena. Například termistor nelze použít pro účely měření jako je měření teploty ohřevu materiálu po průchodu laserovým paprskem, kde se teploty pohybují kolem 500°C a více.

1.4 Druhy termočlánků a jejich pracovní parametry [4][5]

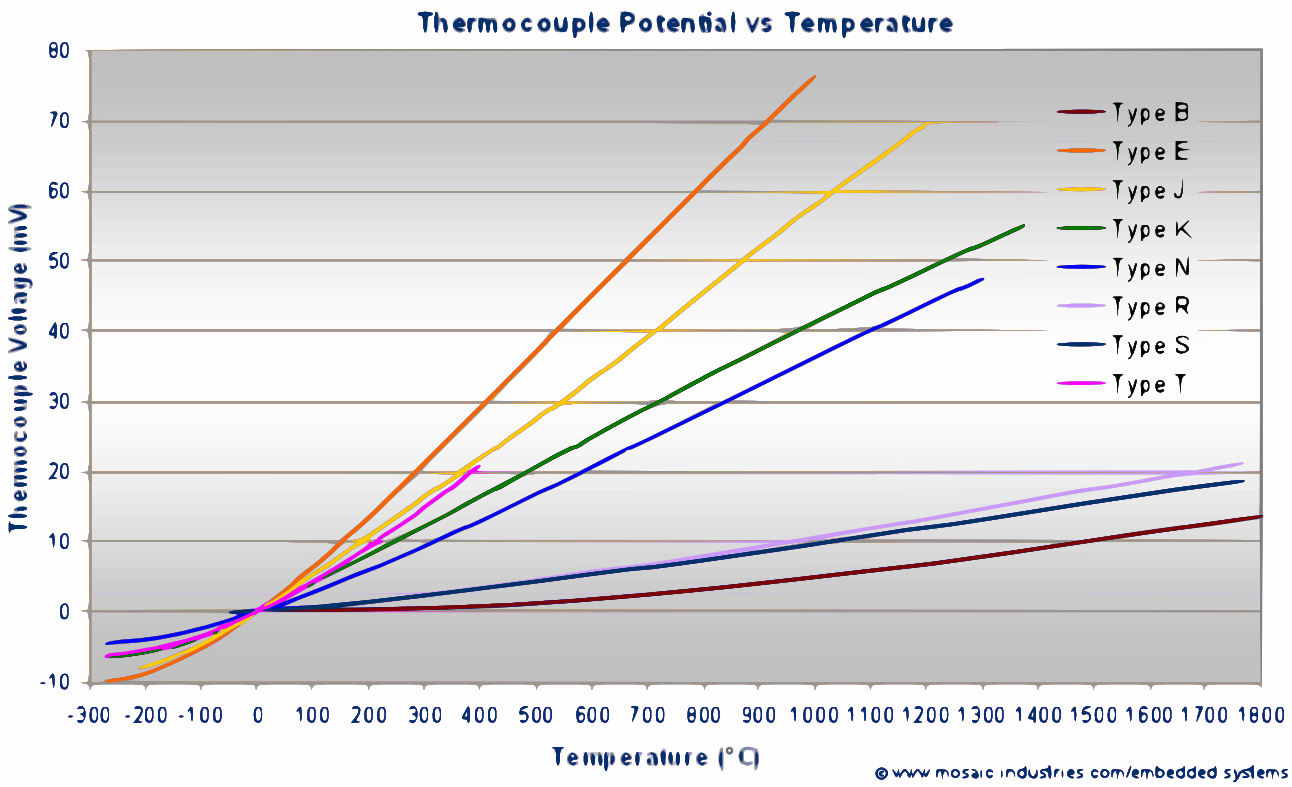
Existuje velké množství termočlánků, kdy termoelektrické napětí vzniká při všech různých kombinací materiálu při dané teplotě. Proto jsou zavedeny určité vlastnosti, které je zájem:

- ***Požadavky instalace***
- ***Chemická odolnost***
- ***Odolnost proti abrazi a vibracím***
- ***Teplotní rozsah***

Výběr materiálů jsou takové, aby změna napětí bylo co nejvyšší. Na obrázku 1.9 a 1.10 je závislost termoelektrického napětí na teplotě pro různé typy termočlánků.



Obr. 1.9 Závislost termoelektrického napětí na teplotě pro různé typy termočlánků [2]



Obr. 1.10 Závislost termoelektrického napětí na teplotě pro různé typy termočlánků [6]

Tím, že termočlánek vytvořen jako slitina různých prvků, dotujeme tak jeho vlastnosti z hlediska teplotního rozsahu, chemické odolnosti i vzniku vyššího termoelektrického napětí. V tabulce 1.1 jsou normou definovaný termočlánky s daným složením. Tím norma zaručuje, že každý termočlánek od různých výrobců musí splňovat dané materiálové složení a sloužit pro daný teplotní rozsah. Písmeno udává marketingové označení, druhý sloupec udává složení materiálu. Třetí sloupec udává teplotní rozsah a poslední sloupec tabulky 1.1 udává rozsah generovaného termoelektrického napětí v mikrovoltech.

Type	Conductors	Temperature range	µV
T	+ Copper (Cu) - Constantan (CuNi)	-200 °C / + 350 °C	-5603 to -17816
J	+ Iron (Fe) - Constantan (CuNi)	-40 °C / + 750 °C	-1960 to +42283
E	+ Chromel (NiCr) - Constantan (CuNi)	-200 °C / + 900 °C	-8824 to +68783
K	+ Chromel (NiCr) - Alumel (NiAl)	-200 °C / + 1200 °C	-5891 to +48828
N	+ Nicrosil (NiCrSi) - Nisil (NiSi)	-200 °C / + 1200 °C	-3990 to +43836
S	+ Platinum (Pt) - Platinum 10% Rhodium (Pt10%Rh)	0 °C / + 1600 °C	0 to +18771
R	+ Platinum (Pt) - Platinum 13% Rhodium (Pt13%Rh)	0 °C / + 1600 °C	0 to +18842
B	+ Platinum 30% Rhodium (Pt30%Rh) - Platinum 6% Rhodium (Pt6%Rh)	+ 600 °C / + 1700 °C	+1791 to + 12426

Tab. 1.1 Nejběžnější normou definované termočlánky [3]

1.5 Použití termočlánků

1.5.1 Výhody použití termočlánků

- ***Teplotní rozsah***

Pokud je zvoleno dobrých sloučenin materiálů, lze se pohybovat v rozmezí -200°C až +2500°C. Proto se termočlánek využívá ve všech možných aplikacích, pokud je to možné. [7]

- ***Robustnost***

Termočlánky jsou ve většině případů imunní vůči vibracím, proto je lze použít i v nehostinných aplikacích. [7]

- ***Rychlá odezva***

Právě díky tomu, že termočlánky jsou rozměrově malé, mají malou termální kapacitu. To znamená, že termočlánky téměř okamžitě odpovídají skutečné teplotě. [7,8]

1.5.2 Nevýhody použití termočlánků [7]

- ***Komplexnost signálu***

Je potřeba udělat velkou časovou investici v návrhu k zamezení chyb vzniklé nepřesnostmi.

- ***Přesnost***

Měření je právě jen tak přesné, jako je rychlost měření referenčního spoje.

- ***Náchylné vůči korozím***

Průběhem času mají termočlánky horší přesnost, proto je ochrana a údržba nezbytná.

- ***Náchylné vůči rušení***

Právě díky tomu, že jsou měřeny mikrovolty, můžou být termočlánky velice snadno rušeny elektrickými a magnetickými poli. Kroucením vodiče, lze potlačit magnetické pole přijímané z okolí. Užitím stíněného kabelu lze potlačit elektrické pole přijímané z okolí.

1.6 Možnosti zpracování výstupního signálu [7]

Problém nastává při využití výstupního signálu. Výstupní napětí termočlánku je příliš malé (viz tab. 1.1). Dalším problémem je nelineární vztah mezi napětím a teplotou. Dále je potřeba kompenzace referenčního spoje a můžou být problémy i se zemněním termočlánku. Kompenzace referenčního spoje už je výše vysvětlena.

1.6.1 Malé výstupní napětí [7]

Nepoužívanější termočlánky jsou typu J, K a T. V tabulce 1.2 je uvedena citlivost na jednotlivých druzích termočlánku.

Thermocouple Type	Seebeck Coefficient (µV/°C)
E	61
J	52
K	41
N	27
R	9
S	6
T	41

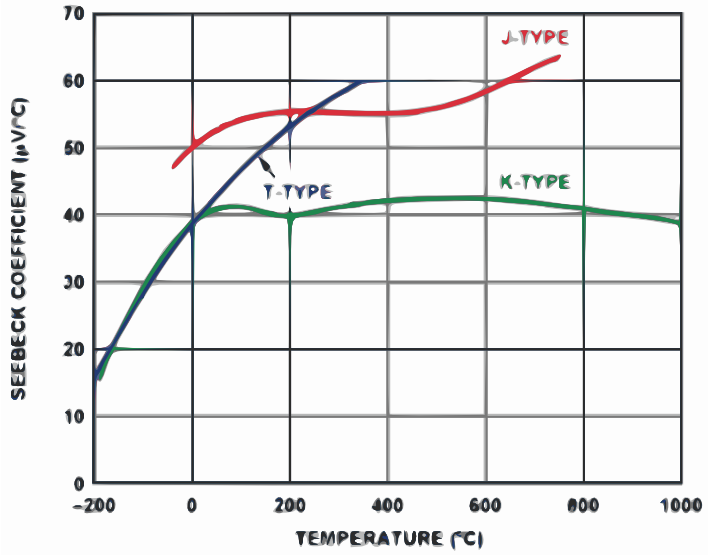
Tab. 1.2 Seebeckův koeficient dle typu termočlánku [7]

Je zvoleno typicky zesílení okolo 100 násobek pro možnosti další využití jako je například A/D převodník. Jelikož, ale termočlánkové vedení jsou dlouhé a často vedou skrz elektricky rušící prostředí. Může být obtížné rozlišit skutečný signál od rušícího signálu. Proto je třeba věnovat pozornost návrhu filtru před vlastním zesílení. Existují dvě

možnosti pro extrakci signálu ze šumu. Většinou jsou kombinovány. První možností je použitím diferenciálního zesilovače (přístrojový zesilovač). Rušení se objeví na obou vodičích, proto se šum eliminuje. Druhou možností je filtr typu dolní propusti anglicky „low-pass“. Filtr by měl eliminovat radiovou frekvenci nad 1MHz a frekvenci zdroje 50/60Hz. [7]

1.6.2 Výstupní napětí je nelineární [7]

Sklon charakteristiky se mění s teplotou (viz. obr. 1.11). V obrázku 1.11 vidíme jednu důležitou vlastnost pro naše měření a to je téměř konstantní seebeckův koeficient u termočlánku typu K od 0°C.

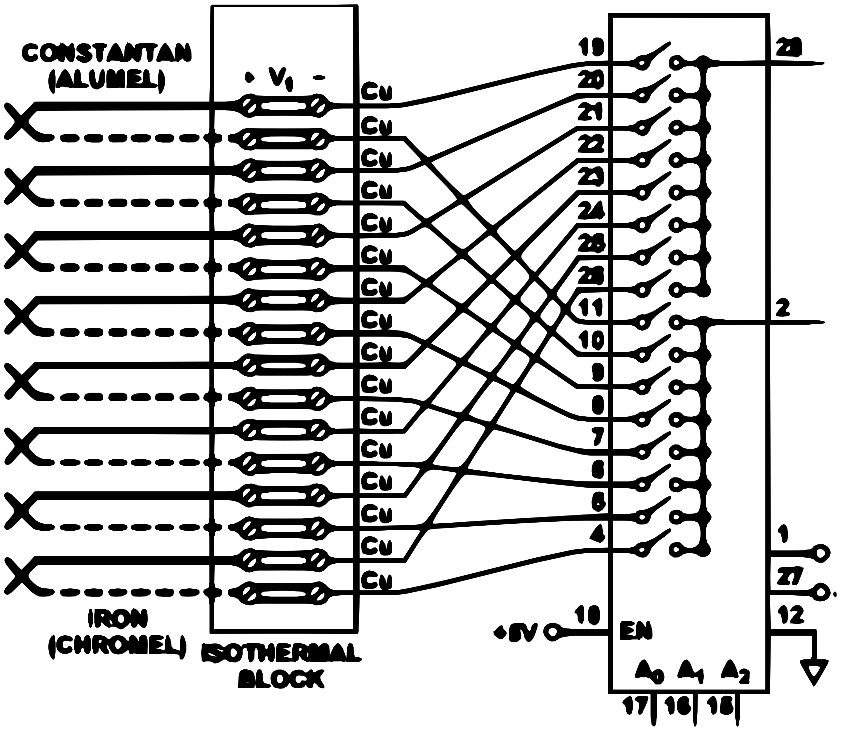


Obr. 1.11 Nelineární průběh seebeckových koeficientů v závislosti na teplotě [7]

2 Návrh vstupní části měřící karty

2.1 Návrh

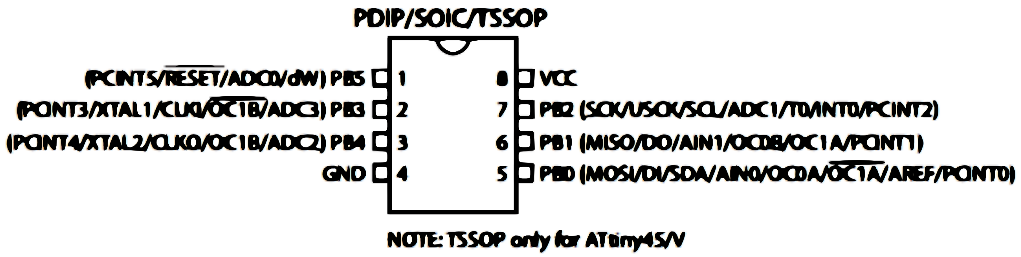
Cílem je dosáhnout měření teploty na deseti různých místech každou vteřinu. To znamená, že existuje několik možností. První možností je koupit zesilovač pro každý termočlánek a přivést vodiče na kartu U3-HV. Seznámení s měřící kartou je v kapitole 4. Ovšem toto řešení je neekonomické. Proto je rozhodnuto pro druhou možnost přepínání signálů z termočlánků na jeden kvalitnější zesilovač. Toto řešení se zdá ekonomicky přijatelnější, ale vede to ke komplexnějšímu návrhu celého systému. To znamená, že je potřeba zařízení, které bude opakovaně vykonávat činnost, které je žádáno. Toto zařízení se jmenuje mikroprocesor. Pro naší jednoduchou činnost stačí 8bitový RISC mikroprocesor. RISC mikroprocesor je neuvěřitelně složité zařízení, ale díky sériovosti výroby se dá pořídit za pár korun. Dále je potřeba zařízení, které umí přepínat analogové vstupy podle potřeby. Toto zařízení se nazývá multiplexor. Lépe zřetelnější činnost multiplexoru je vysvětlena v kapitole 2.2.1. Na obrázku 2.1 je ukázáno řešení přepínání signálu. Kompletní schematický návrh je přiložen v příloze A.



Obr. 2.1 Možnost přepínání signálů [9]

2.1.1 Řídící část – Přepínání signálu, Mikroprocesor

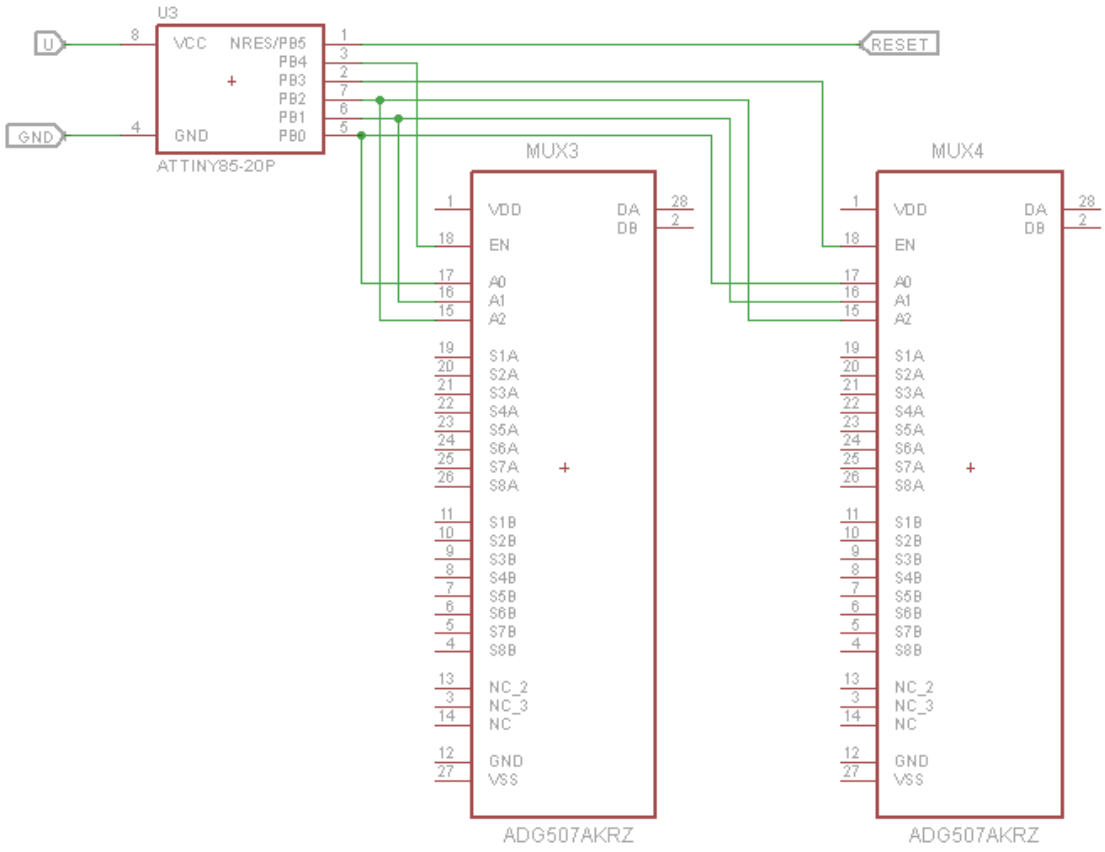
Pro tento projekt je vybrán mikroprocesor ATTiny 85. Mikroprocesor je od firmy Atmel AVR. Má zabudovaných plno funkcí jako je například PWM modulace, 10-Bitový AD převodník atd. [10] Pro naše účely stačí řízení vstupů a výstupů a použití 8-bitového čítače a časovače. Na obrázku 2.2 je uvedeno označení jednotlivých pinů.



Obr. 2.2 Mikroprocesor ATtiny 85 [11]

Když lze opominout napájení a zem je k dispozici 6 vstupů nebo výstupů podle konfigurace mikroprocesoru. Jeden signál je potřeba na řízení činnosti z počítače, tím je pin číslo 1 označený jako RESET. Logickou hodnotou 1 je povolena činnost mikroprocesoru a logickou hodnotou 0 je jeho činnost zakázaná. Toho je dále využíváno v řízení běhu programu a synchronnosti činností v kapitole 5 a 6. Proto signál RESET použít nelze jako výstupní signál. K dispozici je 5 řídicích signálů. Tři signály jsou použity na binární kód potřebný pro dekodér zabudovaný v multiplexoru a zbylé dva signály jsou použity na výběr aktuálního multiplexoru. Na obrázku 2.3 je zobrazen návrh pro řízení.

Procesor je potřeba ještě naprogramovat. To znamená napsat daný kód v programovacím jazyku, v tomto případě se jedná o programovací jazyk C a pomocí kompilátoru vytvořit strojový kód. Tento soubor pomocí programátoru je potřeba odeslat do paměti mikroprocesoru více v kapitole 5.



Obr. 2.3 Řídící signály z mikroprocesoru pro řízení činnosti

Sekvence činnosti, která probíhá v daném čase a periodicky se opakuje je uvedena v následující tabulce 2.1.

A2	A1	A0	EN1	EN2
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0
0	0	0	0	1
0	0	1	0	1

Tab. 2.1 Sekvence činnosti mikroprocesoru

Možná se ptáte, proč tabulka nepokračuje dále. Lze pokračovat a využít plný potenciál pěti vodičů, ale je respektováno zadání a tento návrh je konstruován pro deset termočlánků.

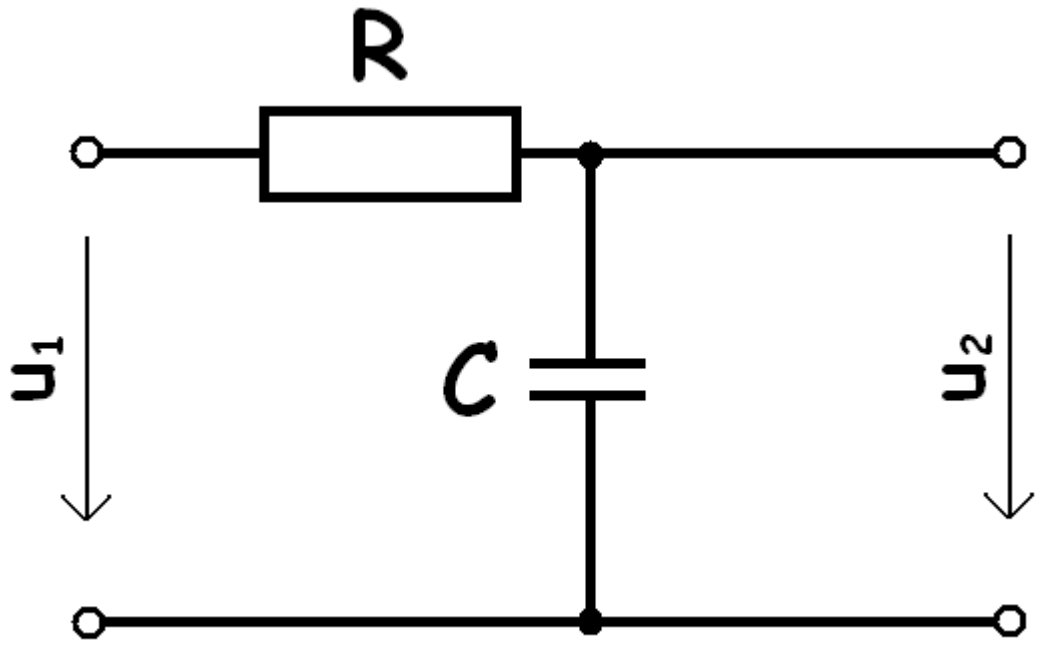
To znamená, že na každou kombinaci hodnot je 100ms. Realizace tabulky 2.1 v programovém jazyce C je uvedena v 5. kapitole.

2.1.2 Vstupní část

To, jakým způsobem jsou přiváděny signály z nezávislých termočlánků na jeden společný zesilovač, jsou objasněny v řídící části. Zbývá otázka, jak navrhnout filtr pro vstupní část. Důvody proč použít filtr, jsou uvedeny v kapitole 1.6.

2.1.2.1 Návrh filtru

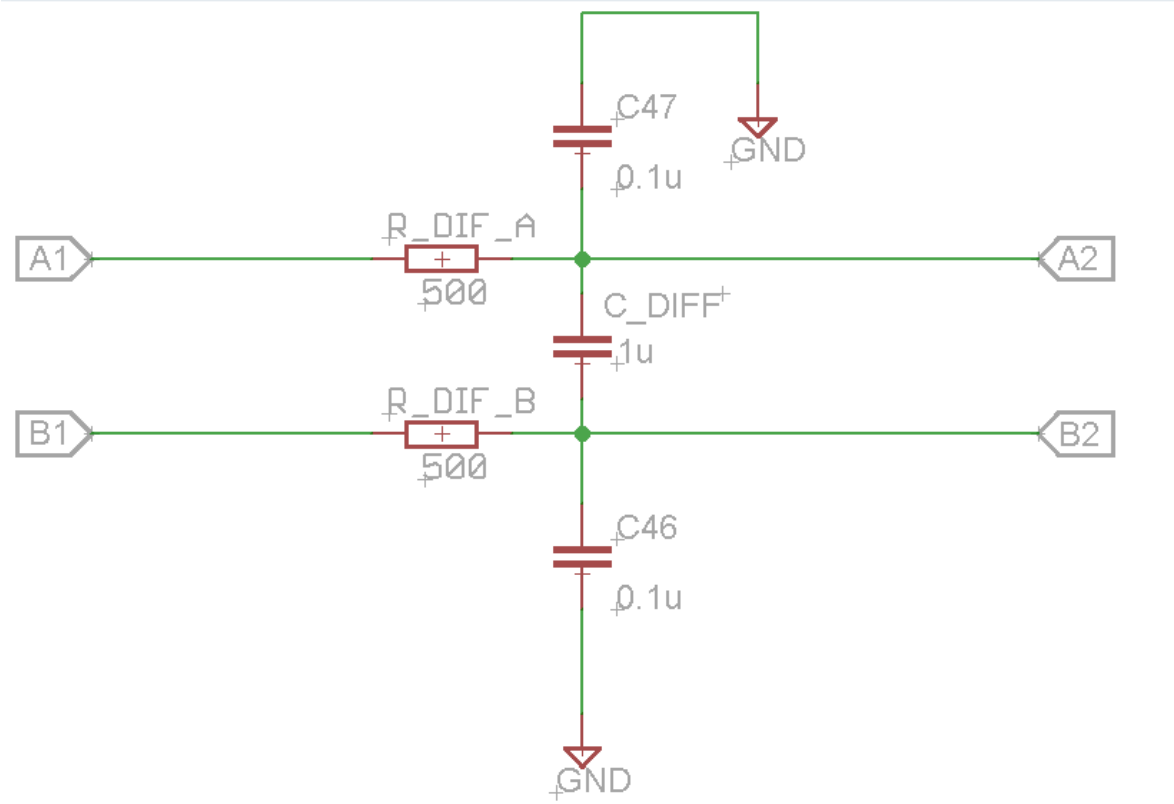
To, že termočlánek může fungovat, jako anténa už je zmíněno v přechozích kapitolách. Proto je zapotřebí utlumit vyšší frekvence. Jelikož teplota není rychle měnící se veličina, můžeme použít filtr typu dolní propust anglicky „low pass“ před zesilovačem. Dolní propust lze sestavit ze základních součástek jako je rezistor a kondenzátor. [13] Na obrázku 2.4 je uveden základní filtr typu dolní propust.



Obr. 2.4 Filtr typu dolní propust [13]

Jelikož jsou vstupy jako diferenciální je filtr potřeba upravit. Na obrázku 2.5 lze vidět diferenciální filtr typu dolní propust. Filtr je zde i z důvodu toho, že do karty U3-HV je vedeno analogové napětí. To je uvnitř karty převáděno na digitální signál. Kdyby filtr

nebyl nepoužit docházelo by k aliasingu na AD převodu. Diferenciální RC filtr složen z 500Ω rezistorů (R_DIF_A a R_DIF_B) a 1μF (C_DIFF) poskytují oříznutí frekvence přibližně nad 320 Hz. Filtr lze dle potřeby přelaďovat. Důležité je nepřekročit hodnotu rezistoru nad 1kΩ. Docházelo by k velkým časovým zpožděním. Kondenzátory C46 a C47 slouží také k utlumení vyšších frekvencí. [12]



Obr. 2.5 Diferenciální filtr typu dolní propust [12]

Je doporučeno, aby diferenciální kondenzátor byl 10x větší, než kondenzátory souhlasného rušení. Pro dosažení nízkého elektromagnetického rušení (EMI) je dobré používat více kondenzátoru paralelně, než jeden velký. Pro filtr 2.5 platí vztah 2.1 a 2.2. [12]

$$f_{diff} = \frac{1}{2\pi R_{DIFF}(2C_{DIFF} + Cc)} \tag{2.1}$$

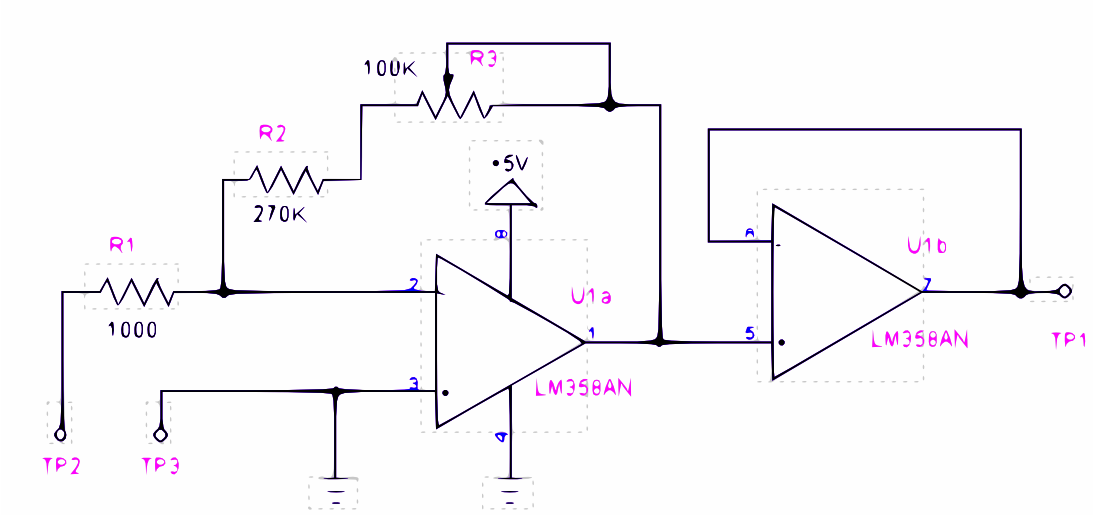
$$f_{CM} = \frac{1}{2\pi R_{DIFF} Cc} \tag{2.2}$$

2.1.2.2 Ochrana I/O

Většinou integrované obvody mají na svých vstupech ochranné obvody, ale ne vždy se na to lze vždy spoléhat. Jedno z řešení jak chránit vstupní a výstupní obvody je použití Schotkyho diod. Shotkyho diody jsou velmi rychlé usměrňovací diody [14]. Pokud pracujeme v rizikovém prostředí, jsou vloženy Schotkyho diody před filtr.

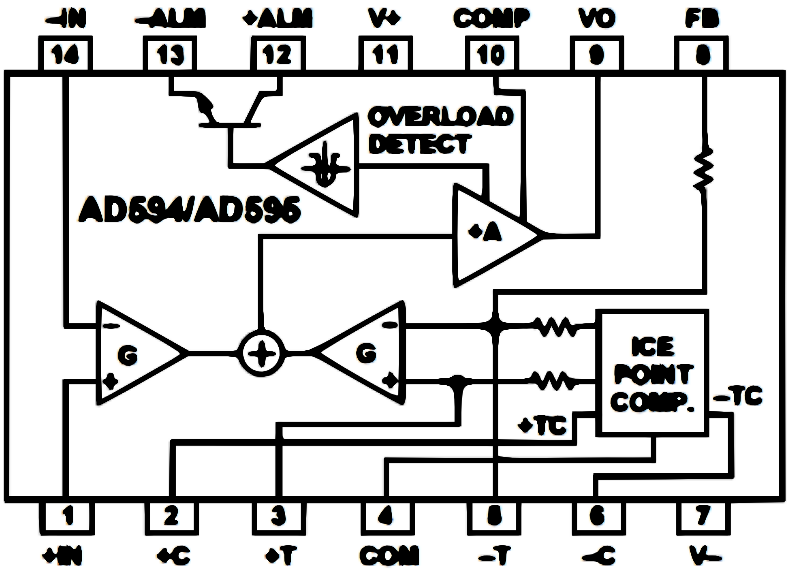
2.1.2.3 Operační Zesilovač

Z počátku bylo uvažováno o použití zapojení na obrázku 2.6. Je zde využito invertujicí zapojení s operačním zesilovačem, kde je pomocí odporu řízeno zesílení, a k výstupu je připojen napěťový sledovač, který nám zajištuje poměrně vysoký výkon a je zde využit i z hlediska otočení fáze.

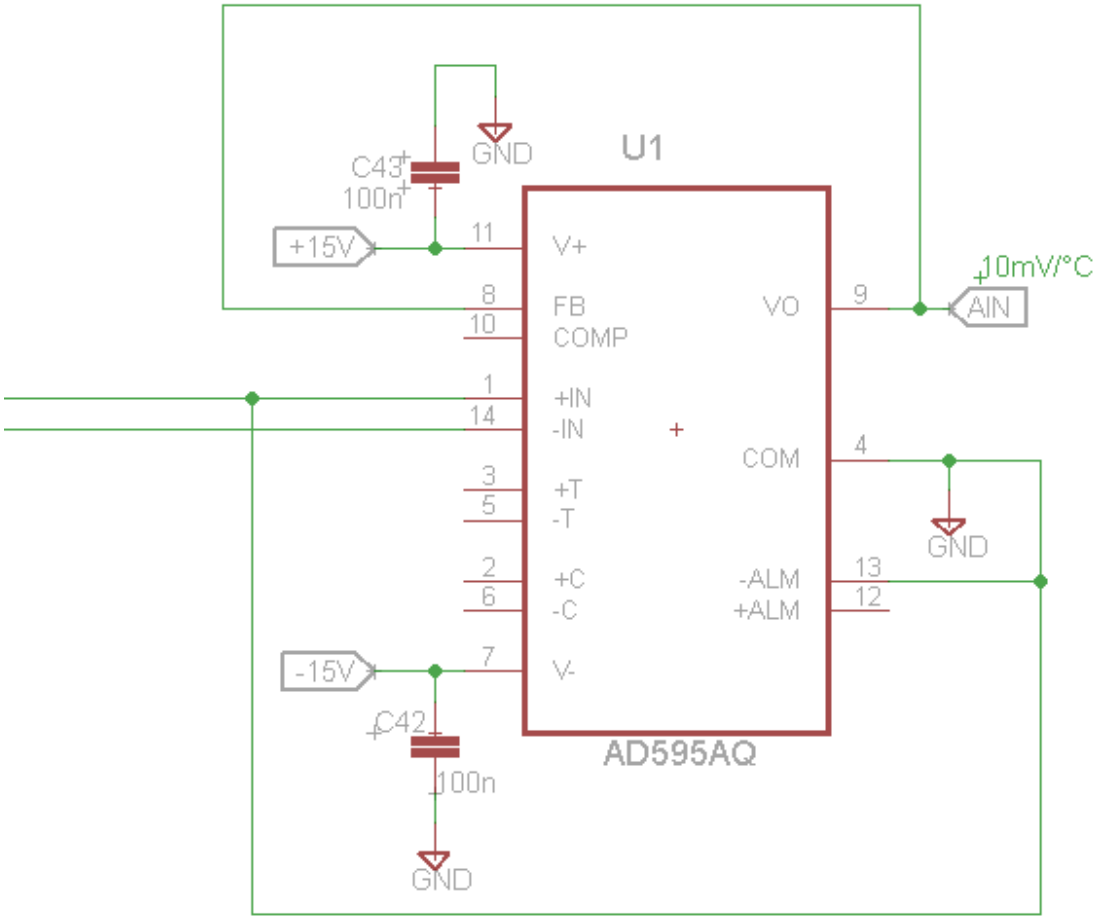


Obr. 2.6 Klasické invertující zapojení s OZ s napěťovým sledovačem [16]

Nakonec bylo rozhodnuto použít zesilovač AD595, který je konstruován speciálně pro termočlánky typu K. Má zabudovanou ledovou kompenzaci „ice point“. Proto odpadá práce měřit referenční hodnotu pomocí termistoru. Má velkou přesnost. A jeho výstup dává 10mV/°C. Na obrázku 2.7 je vnitřní zapojení použitého zesilovače AD595 a na obrázku 2.8 je jeho vnější zapojení.



Obr. 2.7 Vnitřní zapojení AD595 [17]

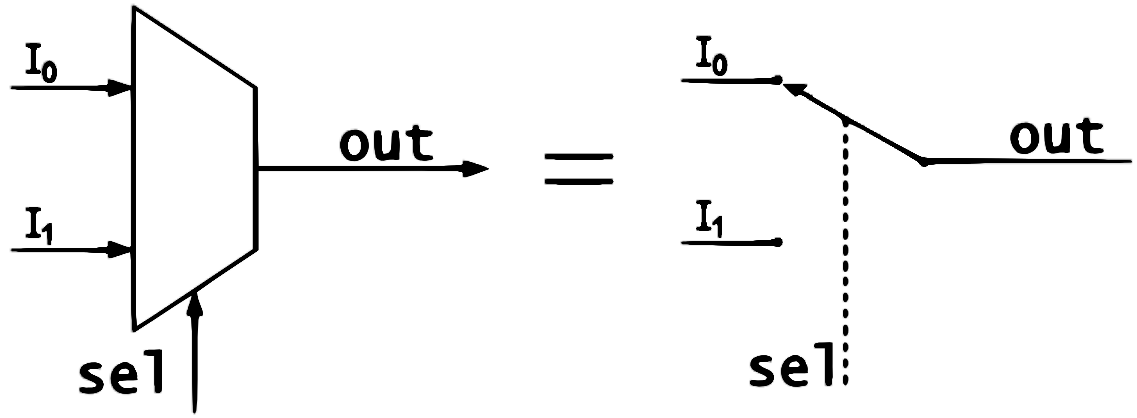


Obr. 2.8 Vnější zapojení AD595

2.2 Zjednodušený princip použitých komponentů

2.2.1 Multiplexor [18]

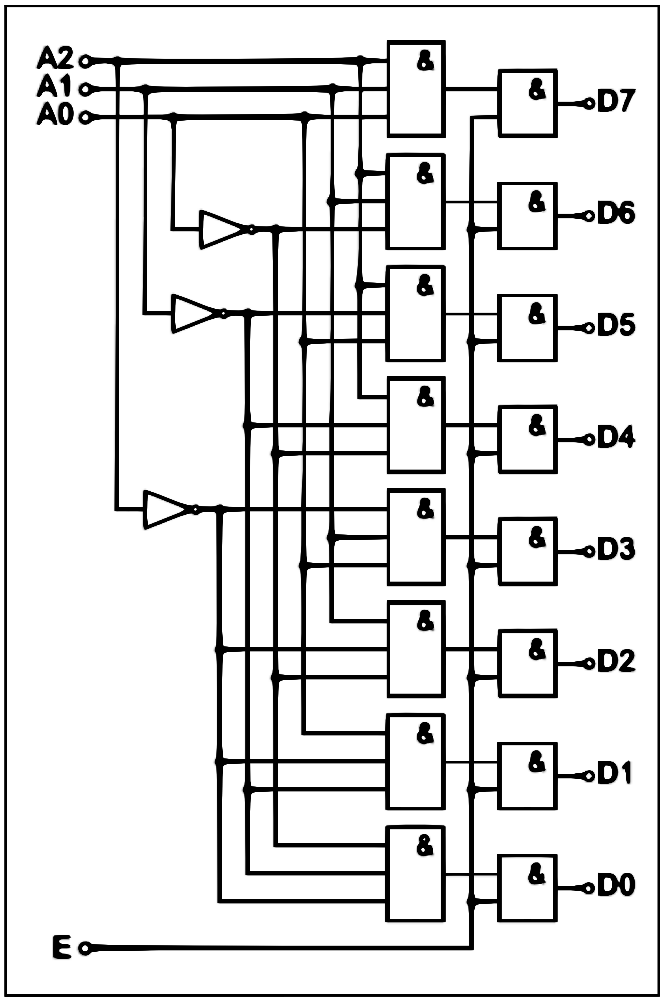
Jedná se o zařízení, které vybírá jeden z několika vstupních signálů. Opakem multiplexoru je demultiplexor. Na obrázku 2.9 je vysvětlen základní princip multiplexoru.



Obr. 2.9 Základní princip multiplexoru [18]

2.2.2 Dekodér [19]

Jedná se o kombinační obvod, který na základě vstupní kombinace dat vytváří výstupní kombinaci dat. Na obrázku 2.10 je vysvětlen základní princip dekodéru. Tento typ případu převádí binární kód na výběr 1 z N. V tomto případě 3bitový vstup na 7 výstupů.



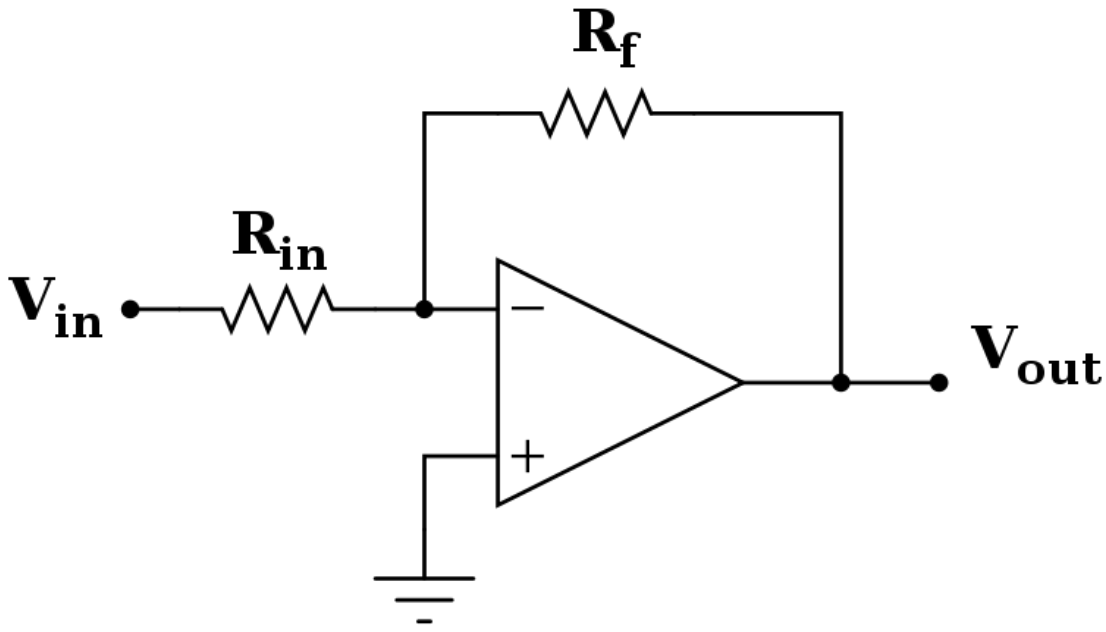
Obr. 2.10 Dekodér binárního kódu na 1 z N [19]

2.2.3 Operační zesilovač [15]

Nejběžnějšími zapojení s operačními zesilovači je invertující zesilovač, neinvertující zesilovač a sledovač napětí. Předpokládá se ideální operační zesilovač pro snadné odvození vztahů.

2.2.3.1 Invertující zesilovač [15]

Výstupem je vstupní napětí vynásobené zesílením. Velikost zesílení je voleno poměrem hodnot v rovnici 2.3. Zapojení je na obr. 2.11.

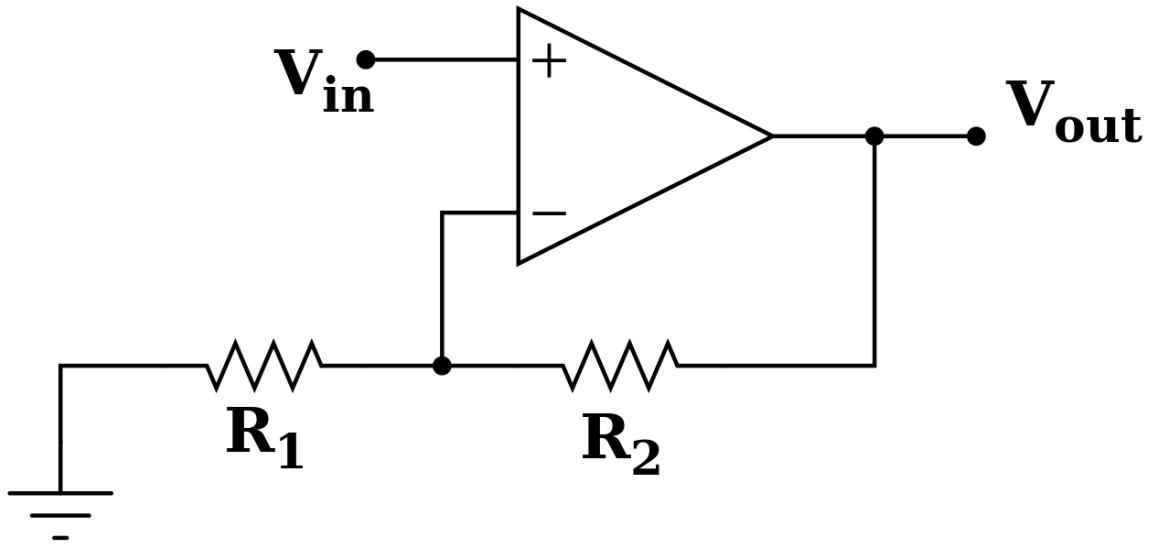


Obr.2.11 Invertující zapojení s OZ [15]

$$U_{vyst} = -\left(\frac{R_f}{R_{in}}\right) U_{vst} \tag{2.3}$$

2.2.3.2 Neinvertujicí zesilovač [15]

Neinvertující zesilovač vždycky zesiluje. Jeho zesílení je větší než 1. Vztah pro toto zapojení je v rovnici 2.4. Zapojení je na obr. 2.12.

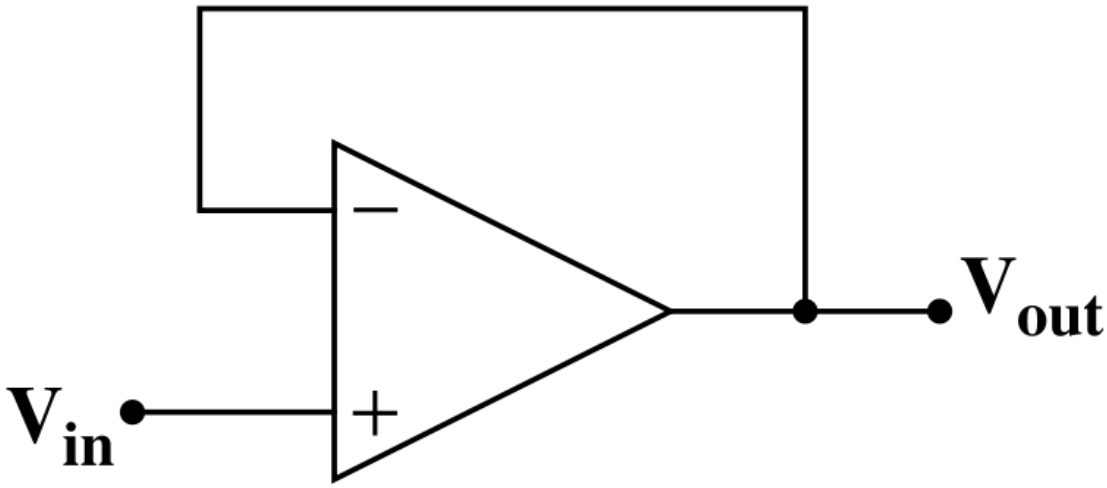


Obr. 2.12 Neinvertující zapojení s OZ [15]

$$U_{vyst} = U_{vst}\left(1 + \frac{R_2}{R_1}\right) \tag{2.4}$$

2.2.3.3 Sledovač napětí [15]

Sledovač napětí nezesiluje ani nezeslabuje jeho zesílení je rovno 1. Používá se k impedančnímu přizpůsobení velké impedanci k malé. Jeho zapojení je na obrázku 2.13.



Obr. 2.13 Sledovač napětí

3 Zásady pro návrh DPS [20]

Tato kapitola není obsažena zadáním bakalářské práce, ale zásady pro návrh destičky plošných spojů by měli být alespoň připomenuty alespoň ty nejdůležitější.

3.1 Návrhová pravidla [20]:

- ***Správná obvodová funkce***
- ***Vyrobitelnost a snadné osazování***
- ***Spolehlivost a snadná opravitelnost***
- ***Estetický design***
- ***Nízká cena***
- ***Normy (EMC)***

3.2 Rozmístění součástek [20]

Základní principy pro rozmístění součástek patří:

- ***Od vyšší k nižší šířce pásma***
- ***Separace funkčních bloků (analog, digitál, napájení0***

- ***Minimalizace vzdáleností (proudové smyčky)***

3.3 Zemnění [20]

Nutné zvolit správnou metodu:

- ***Jednobodové***
- ***Vícebodové***

Z hlediska analogového a číslicového systému je nutné země odseparovat. Na desce se nachází jak citlivá analogová část, tak i silně rušící číslicová část.

3.4 Blokování napájení [20]

Blokování napájení patří k jednomu z nejdůležitějšího návrhového pravidla. Je to z toho důvodu, že určitý čas trvá, než zdroj dodá potřebnou energii. I samotné vedení vykazuje zpoždění. Z toho to důvodu využíváme jako místní zdroj energie kondenzátor, který je co nejblíže ke spotřebiči (integrovaný obvod). Kdy slouží jako zdroj energie. Například kdy se klopný obvod překlápí. Podle funkce rozlišujeme kondenzátory jako:

- ***Filtrační kondenzátor***
- ***Lokální kondenzátor***
- ***Skupinový kondenzátor***

4 Měřící karta

4.1 Seznámení s LABJACK U3-HV

U3-HV od firmy Labjack je zařízení, které zpracovává signály vzorkováním a pomocí kabelu USB posílá data k dalšímu zpracování. Zařízení má 16 flexibilních vstupů a výstupů, 2 čítače a 2 časovače. Podporuje SPI, I2C a mnoho dalších. [21] Pro tyto účely stačí měřit výstupní signál ze zesilovače, řídit činnost procesoru a snímat změnu nejnižšího bitu pro synchronnosti činností mezi PC a procesoru. Na obrázku 4.1 je zobrazena měřící karta a její označení vstupů, výstupů, napájení a země. Pro správnou funkci tohoto zařízení je potřeba nainstalovat ovladač ke kartě.

Obr. 4.1 Labjack U3-HV [21]

5 Návrh programové části pro řídící mikroprocesor

O návrhu řídícího procesoru je již zmíněno v kapitole 2. Pro danou činnost je využita tabulka 2.1.

5.1 Popis funkce

Procesor vykonává svojí činnost v nekonečném cyklu příkazem:

```
While(1)
{

}
```

Každých 100ms je přiřazena na výstupy procesoru právě unikátní hodnotu z tabulky 2.1:

```
PORTB.3 = 0;
PORTB.4 = 1;

PORTB.0 = 0;
PORTB.1 = 0;
PORTB.2 = 0;
```

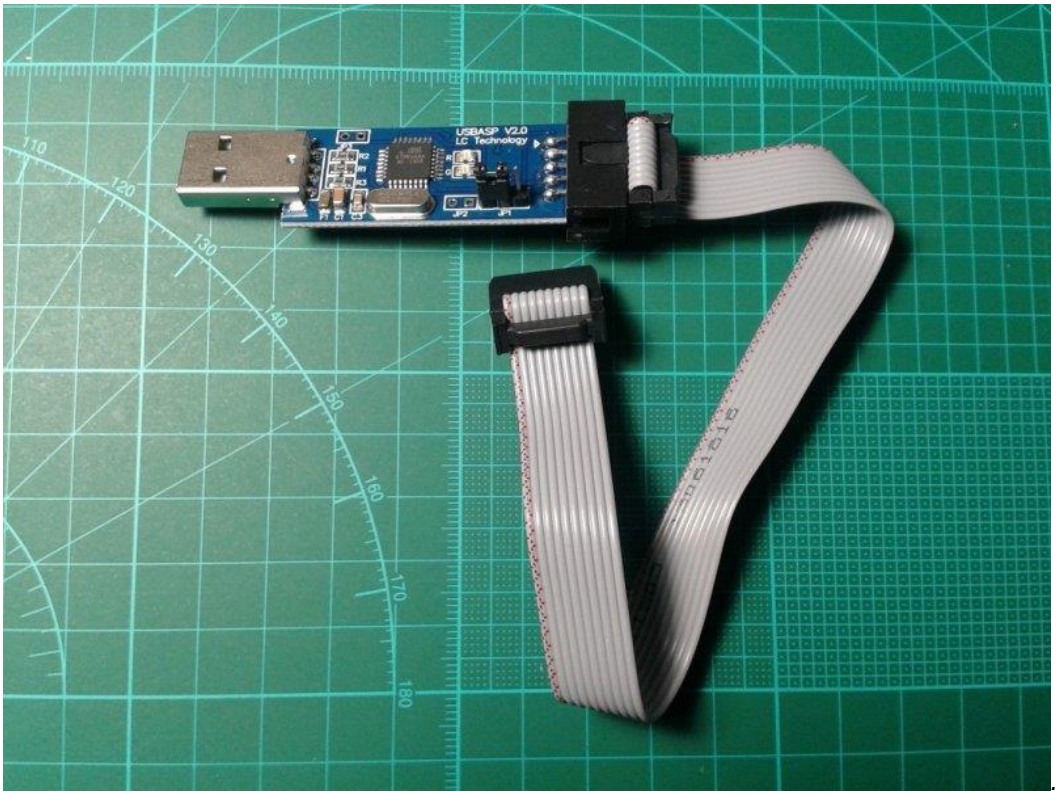
Následuje zpoždění.

```
delay_ms(100);
```

Následuje další kombinace hodnot tabulky, dokud se neprojede celá tabulka 2.1 a celá činnost se opakuje. Kompletní kód s konfigurací procesoru je uveden v příloze.

5.2 Programování – Programátor

Při sestavování kódu vzniká unikátní soubor s příponou .HEX. Je to tzv. strojový kód v HEX formátu. Ten je potřeba dostat do procesoru. K tomu slouží programátor. Na obrázku 5.1 a 5.2 je příklad programátoru. Po nahrání programu do procesoru vždy procesor vykonává naprogramovanou činnost po náběhu napájení.



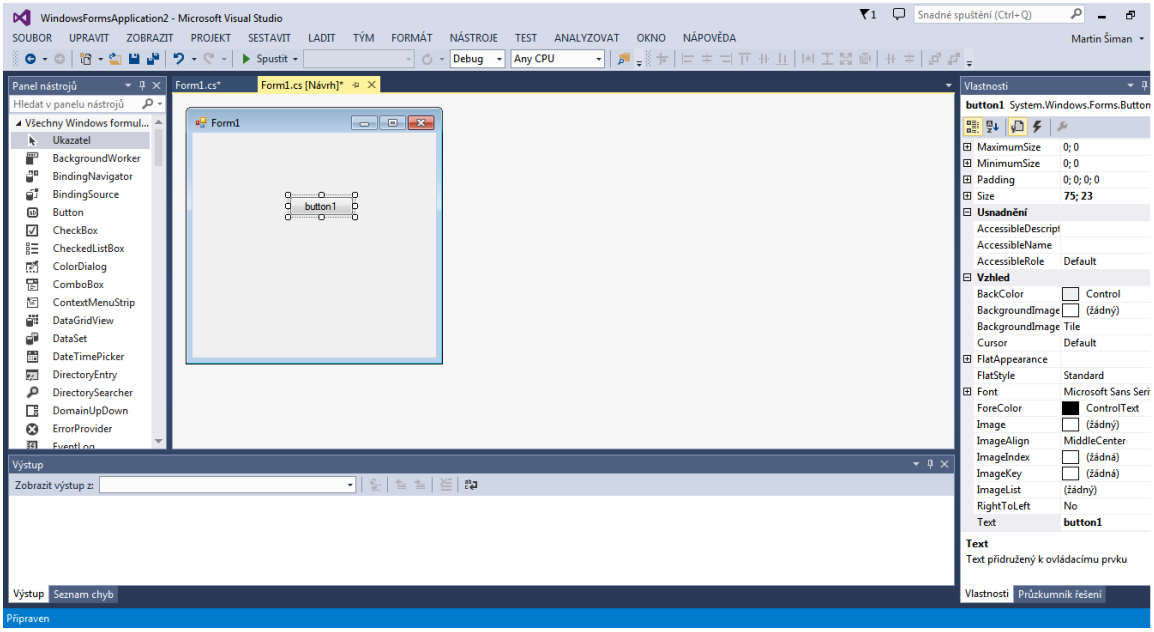
Obr. 5.1 USB programátor [22]

Obr. 5.2 USB programátor 2 [23]

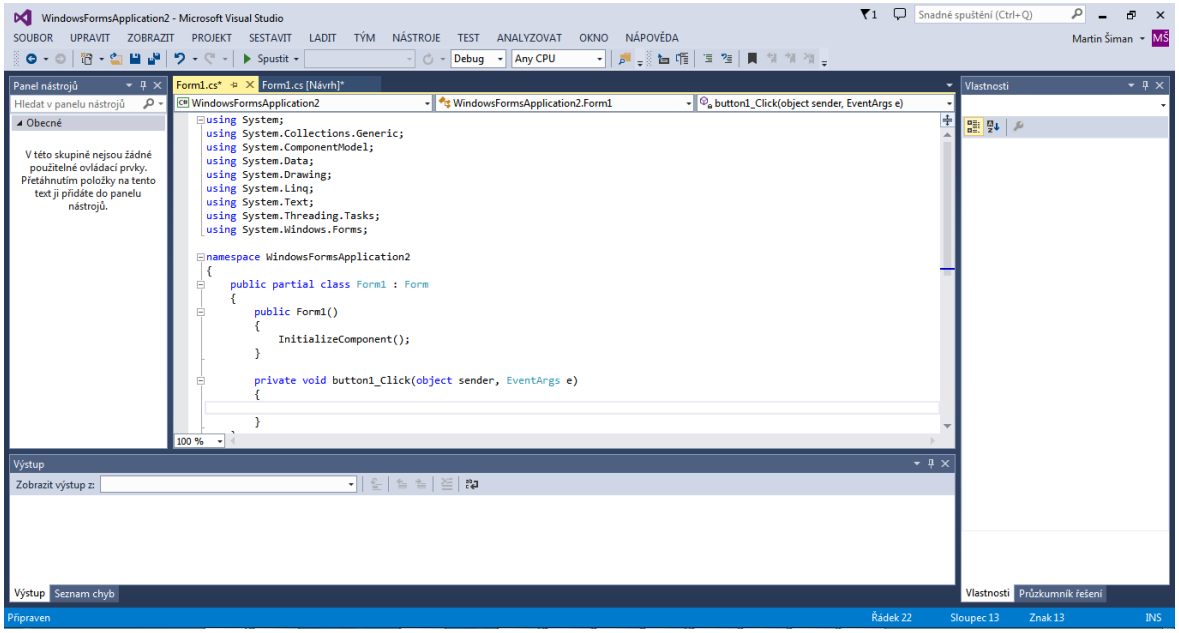
6 Návrh programové části pro koncového uživatele

6.1 Seznámení s vývojovým prostředím

Pro vývoj programu je využíván program Visual Studio, které podporuje několik programovacích jazyků. Pro tento projekt je využito C#. Visual Studio velmi pomáhá při psaní kódu. Na obrázku 6.1 je ukázka na vytváření designu. A na obrázku 6.2 je ukázka programovací části.



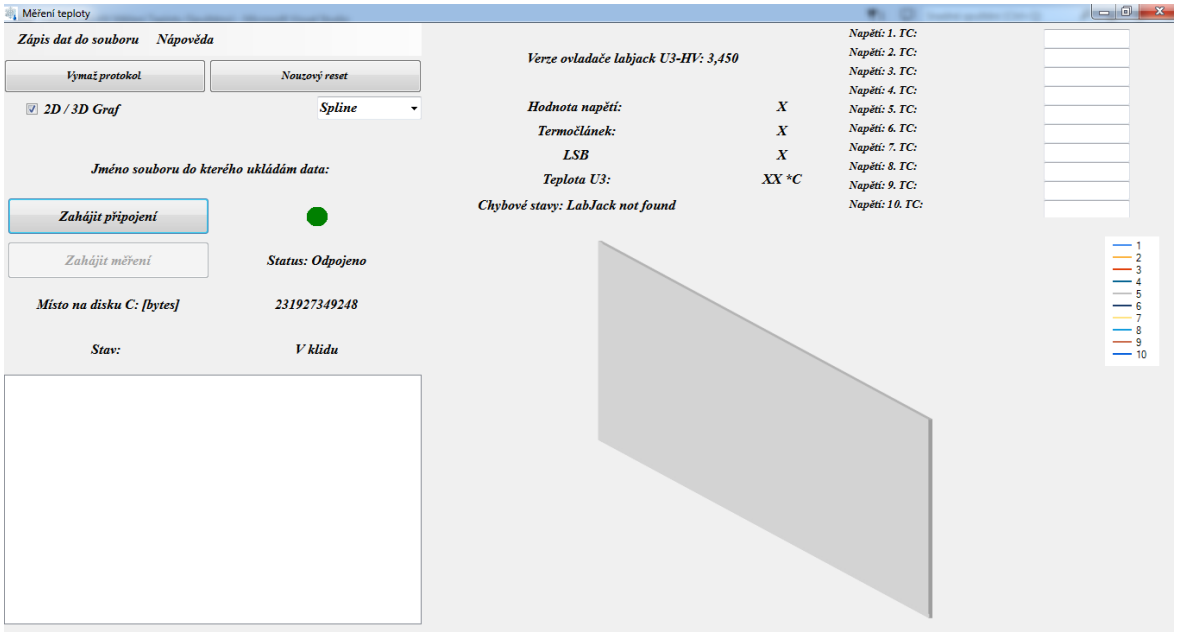
Obr. 6.1 Ukázka vytváření návrhu aplikace



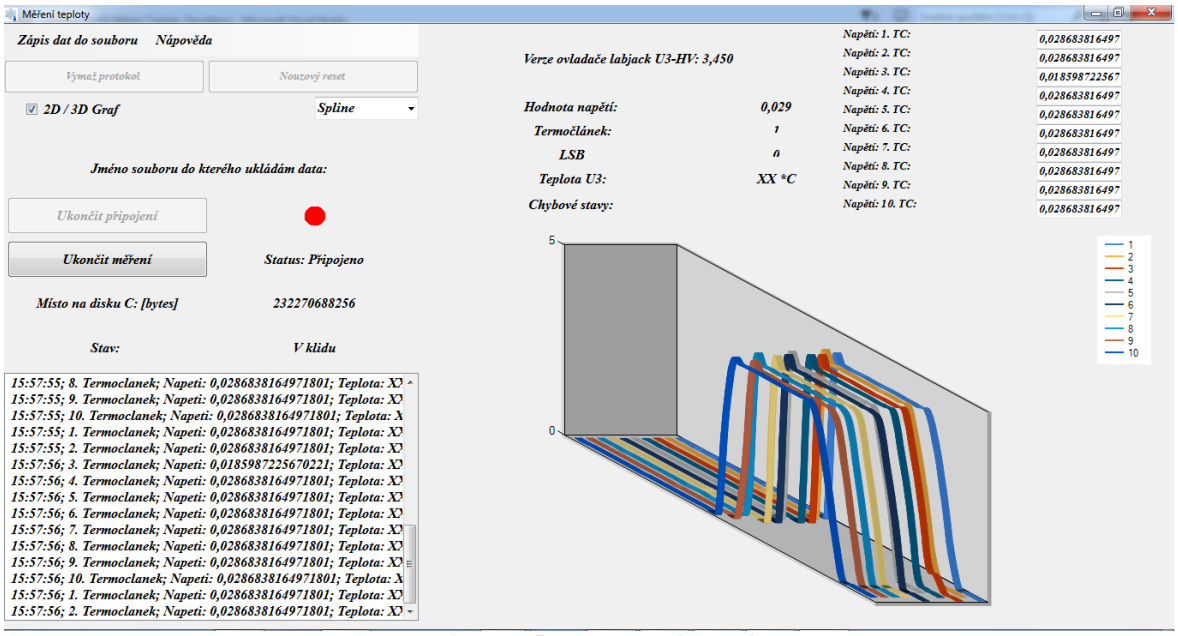
Obr. 6.2 Ukázka programování aplikace

6.2 Design programu

Na obrázku 6.3 je znázorněn návrh měřícího programu. Rozmístění prvků a grafika určitě není ideální, ale většinu času bylo věnováno tomu, aby program skutečně vykonával to co má. Na obrázku 6.4 je program v činnosti měření.



Obr. 6.3 Design programu



Obr. 6.4 Program v činnosti

6.3 Zjednodušený popis funkce programu

Nasázet komponenty na pracovní plochu je jedna věc, ale přiřadit jim události, vlastnosti a naprogramovat celou činnost řízení už je věc druhá. Kód je velmi komplexní i já jako autor kódu mám problém se vyznat. Ponaučení do příště rozepsat si kód do více tříd pro lepší přehlednost. Proto jsou následující algoritmy popsány velmi stručně. Pro větší zájemce je kód přiložen v příloze C a D nebo po případně na přenosném záznamu. Pro správnou komunikaci s kartou je nutné mít nainstalovaný ovladač od firmy labjack.

Na začátku programu jsou uvedené konstanty. Díky nim může i nezkušený uživatel modifikovat program. Například změnit pin výstupního napětí. Doporučoval bych alespoň pro začátek nechat defaultní rozvržení pinů.

```
const int RESET_PROCESSOR_FIO_PIN = 4;
const int U_OUT_ZESILOVAC_AIN_PIN = 2;
const int LSB_PROCESSOR_FIO_PIN = 5;
const double U_CALLIBRATION = 0.25;
const double UAD595_25STUPNU = 0.25;
```

Následuje inicializace programu. To znamená zjištění verze ovladače, vygenerování objektů, nastavení měřítka. Dále se čeká na událost od uživatele. Uživatel může si přečíst nápovědu k programu nebo vytvoření csv souboru na disk do kterého se budou následně ukládat data. Poslední možností je připojit se ke kartě. Program je ošetřený proti většině havarijních stavů. Například vytáhnutím USB kabelu může způsobit havárii programu.

Po kliknutí na tlačítko connect se program připojí s kartou. Nastaví jí do defaultního nastavení a signalizuji stav zařízení pomocí ledky ve vizualizaci programu. Dále program čeká na stisknutí tlačítka zahájit měření. Jakmile je stisknuto. Je vydán povel do karty, která má nastavit pin reset mikroprocesoru na 0. Tím začne činnost naprogramovaného mikroprocesoru a program musí stíhat obsluhovat další události mezitím. Je povolen časovač, který mi vzorkuje výstupy. Jedním se ptám na změnu nejnižšího bitu a druhým se ptám na výstupní napětí od zesilovače. Tím lze spolehlivě určit, jaké napětí odpovídá danému termočlánku. Časovač pomáhá zpracovávat hodnoty. Tzn., vyvolává metody, které obsluhují grafy. Kliknutím na tlačítko ukončit měření je přivedeno napětí na reset mikroprocesoru. Mikroprocesor nastaví všechny výstupy do log. 0 a zastaví se nám činnost programu, protože je reagováno na změnu nejnižšího bitu. Pokud byl zvolen zápis do souboru. Hodnoty by se měli ukládat po každým 10tém měření. Následující kód slouží jako ukázka pro reset mikroprocesoru aneb na jakém způsobu karta funguje.

```
public void Reset_uProcessoru(int DigitalniPort ,int DigitalniUroven)
  {
   try
     {
        LJUD.AddRequest(u3.ljhandle, LJUD.IO.PUT_DIGITAL_BIT, DigitalniPort, DigitalniUroven, 0, 0);
     }
   catch (LabJackUDException exc)
     {
        ShowErrorMessage(exc);
        return;
     }
   try
     {
        LJUD.GoOne(u3.ljhandle);
     }
   catch (LabJackUDException)
     {
        MessageBox.Show("Reset uProcesoru nešel dle plánu");
        return;
     }
  }
```

Program se snaží vykonat žádost (AddRequest) na požadovaný bit požadovanou logickou úroveň (PUT_DIGITAL_BIT, DigitalniPort, DigitalniUroven). Pokud nastane výjimka, například odpojení USB kabelu, ukončí se metoda a algoritmy na detekci chyb ukončí program. Příkaz GoOne slouží už jen k vykonání daných žádostí. Lze mít i více žádostí, než jen jednu. Lze zpozorovat, jak obyčejný reset vypadá složitě. Proto nelze vysvětlovat všechny příkazy krok po kroku.

7 Měření s navrženou kartou

7.1 Měření

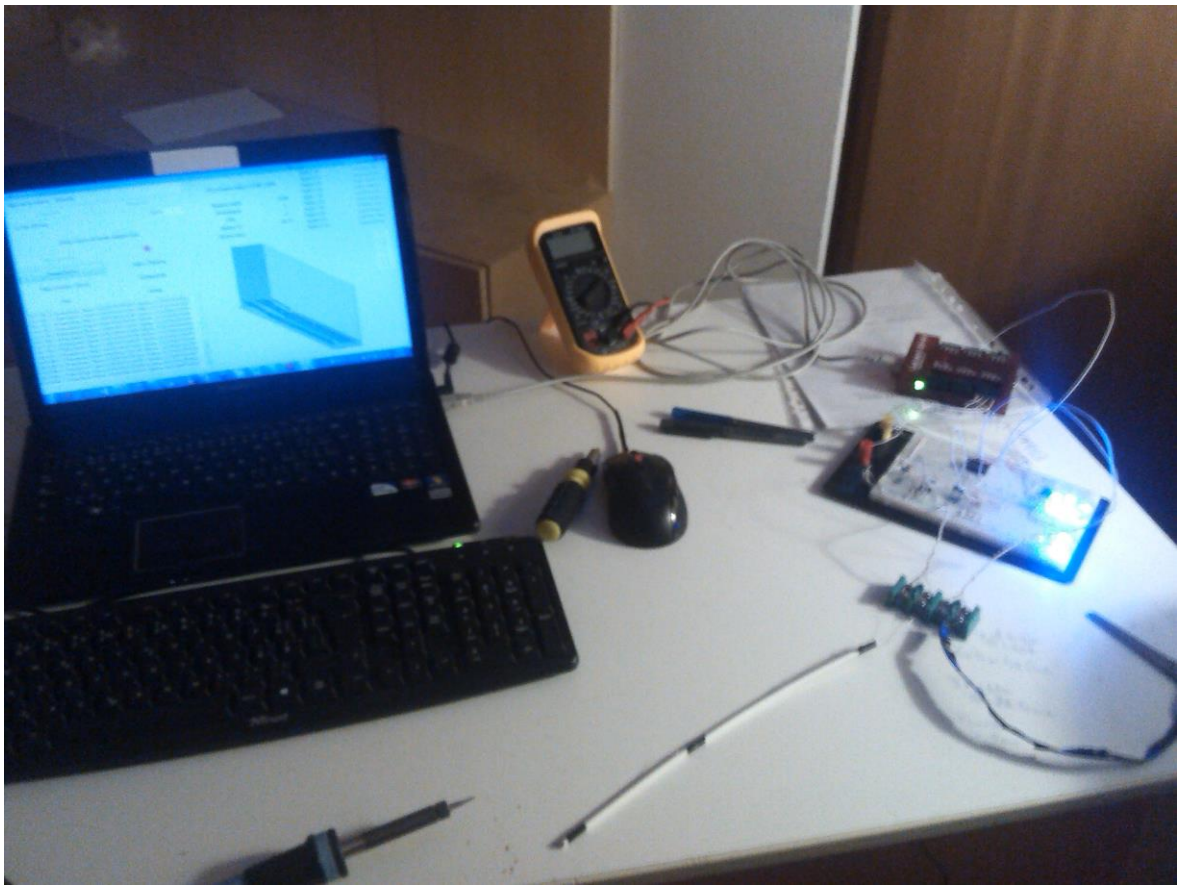
Měření je prováděno pouze s jedním termočlánkem, protože měření mám připravené na nepájivém poli a multiplexory jsou jenom v SMD provedení.

Pro stanovení velmi přesného měření teploty je program upravený tak, aby napětí při pokojové teplotě odpovídalo právě napětí pokojové teplotě získané z datasheetu pro AD595. Následující výňatek kódu ukazuje kalibraci zařízení a převod na °C.

```
Skutecna_Hodnota_Ve_Stupnich = (double.Parse(Data) - U_CALLIBRATION)
/ UAD595_ZISK;
```

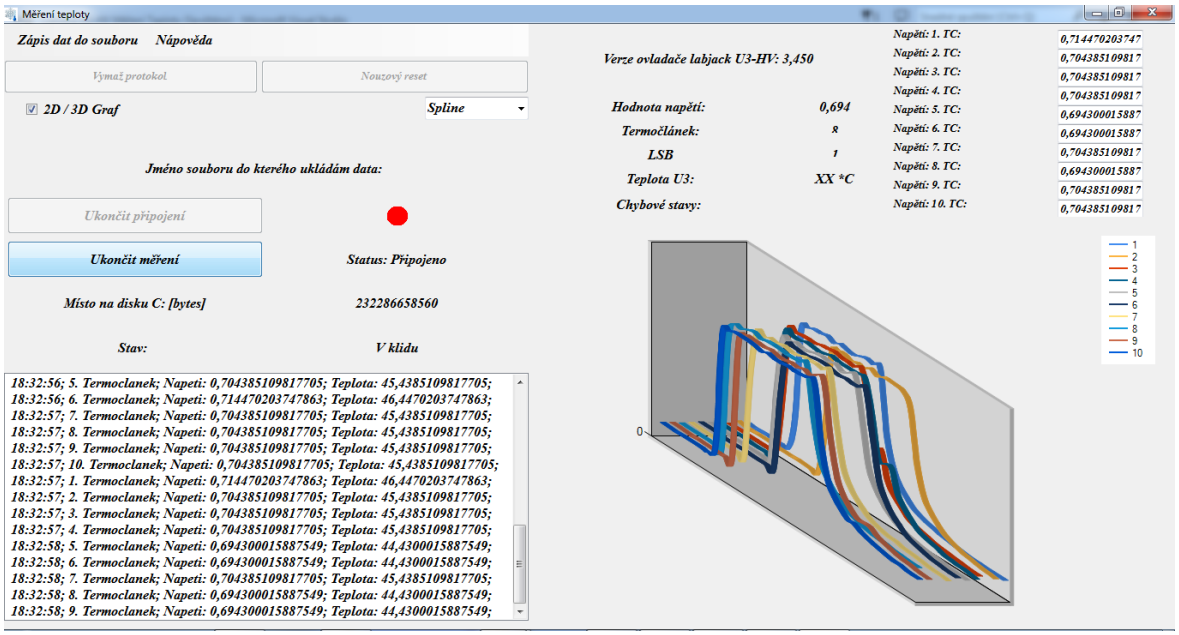
Napětí, které je přijímáno, je odečteno od kalibračního napětí, které je zjištěno podle pokojové teploty. A pokud má zesilovač AD595 citlivost 10mV/°C je následující napětí vyděleno 10mV. Je získána skutečná hodnota ve stupních, která je následně graficky zobrazena.

Pro pokusné měření byla změřena teplota pájky. Na obrázku 7.1 je zobrazeno principiální zapojení. Vlevo dole je pájka, která po přiložení ohřívá termočlánek. Termočlánek vede na izotermální svorkovnici, následuje diferenciální RC filtr a odečtení od referenční hodnoty. Následuje zesilovač. Výstupní napětí vede do karty U3-HV, kde je napětí vzorkováno, digitalizováno a posláno přes USB, kde je následně softwarově převedeno na teplotu a ta je graficky zobrazena na monitoru.



Obr. 7.1 Principiální zapojení

Na obrázku 7.2 je zachycen průběh teploty. Pájka dosahovala v maximech teplot kolem 300°C což odpovídá zhruba teplotě tání cínu, z toho vyplývá, že zařízení je kalibrováno dobře. Po odložení pájky teplota se blížila k pokojové teplotě. Měření je ukončeno předčasně při teplotě termočlánku kolem 44°C proto, aby byl zobrazen daný průběh teploty.



Obr. 7.2 Měření teploty pájky

Závěr

Měření jsem prováděl jenom s jedním termočlánkem, protože výrobek mám sestavený na nepájivém poli, kde nemůžu zapojit multiplexory, které jsou v SMD provedení. K systému stačí jen připojit multiplexor.

Veškerou činnost programu nelze popsat do práce. K tomuto problému bych rád využil prezentaci a veškerou činnost programu názorně předvedl.

Program pro koncového uživatele není dokonalý. Určitě je hodně co zlepšovat, co se týče designu. Toto byl můj první větší projekt. Co se týče optimalizace: Méně náročný operace a algoritmy by byly určitě na místě při větším časovém prostoru.

Kdybych nebyl závislý na zadání tak určitě bych nepoužil kartu LabJack, ale pokusil bych se navrhnout celý systém i s komunikací přes USB.

Seznam literatury a informačních zdrojů

[1] Omega. Measure and Control. [online]. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://www.omega.com/temperature

[2] Tzbinfo. Měření základních fyzikálních veličin. [online]. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://vytapeni.tzb-info.cz/teorie-a-schemata

[3] Termo electric. Thermocouple base materials acc to IEC 584 [online]. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://www.nanyange.com.tw/driver/drivers/Technical%20Information_02.pdf

[4] Omega Engineering Česká republika. [online]. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://www.omegaeng.cz/

[5] Wikipedia. Thermocouple. [online]. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://en.wikipedia.org/wiki/Thermocouple

[6] Mosaic Documentation Web. Thermocouple types. [online]. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://www.mosaic-industries.com/embedded-systems/microcontroller-projects/temperaturemeasurement/thermocouple/types-wire-element

[7] Analog Devices. Two Ways to Measure Temperature Using Thermocouples. Autors: Matthew Duff, Joseph Towey. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://www.analog.com/library/analogDialogue/archives/44-10/thermocouple.html

[8] Wikipedia. Měrná tepelná kapacita. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://cs.wikipedia.org/wiki/Měrná_tepelná_kapacita

[9] Analog Devices. Datasheet AD595. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://www.analog.com/media/en/technical-documentation/application-notes/AN-369.pdf

[10] GM electronics. ATTINY85. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://www.gme.cz/attiny85-20pu-dip8-atmel-p432-019

[11] GM electronics. ATTINY85. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://www.atmel.com/Images/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85_Datasheet.pdf

[12] Texas Instrument. Precision Thermocouple Measurement. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://www.ti.com/

[13] Wikipedia. Dolní propust. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://cs.wikipedia.org/wiki/Dolní_propust

[14] Wikipedia. Schottkyho dioda. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://cs.wikipedia.org/wiki/Schottkyho_dioda

[15] Wikipedia. Operační zesilovač. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://cs.wikipedia.org/wiki/Zapojení_s_operačním_zesilovačem

[16] Lewis Loflin. Zesilovač pro termočlánky. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://www.bristolwatch.com/ele/thermalcouple_amplifier.htm

[17] Analog Devices. Thermocouple Amplifiers with Cold Junction Compensation. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://www.analog.com/media/en/technical-documentation/datasheets/AD594_595.pdf

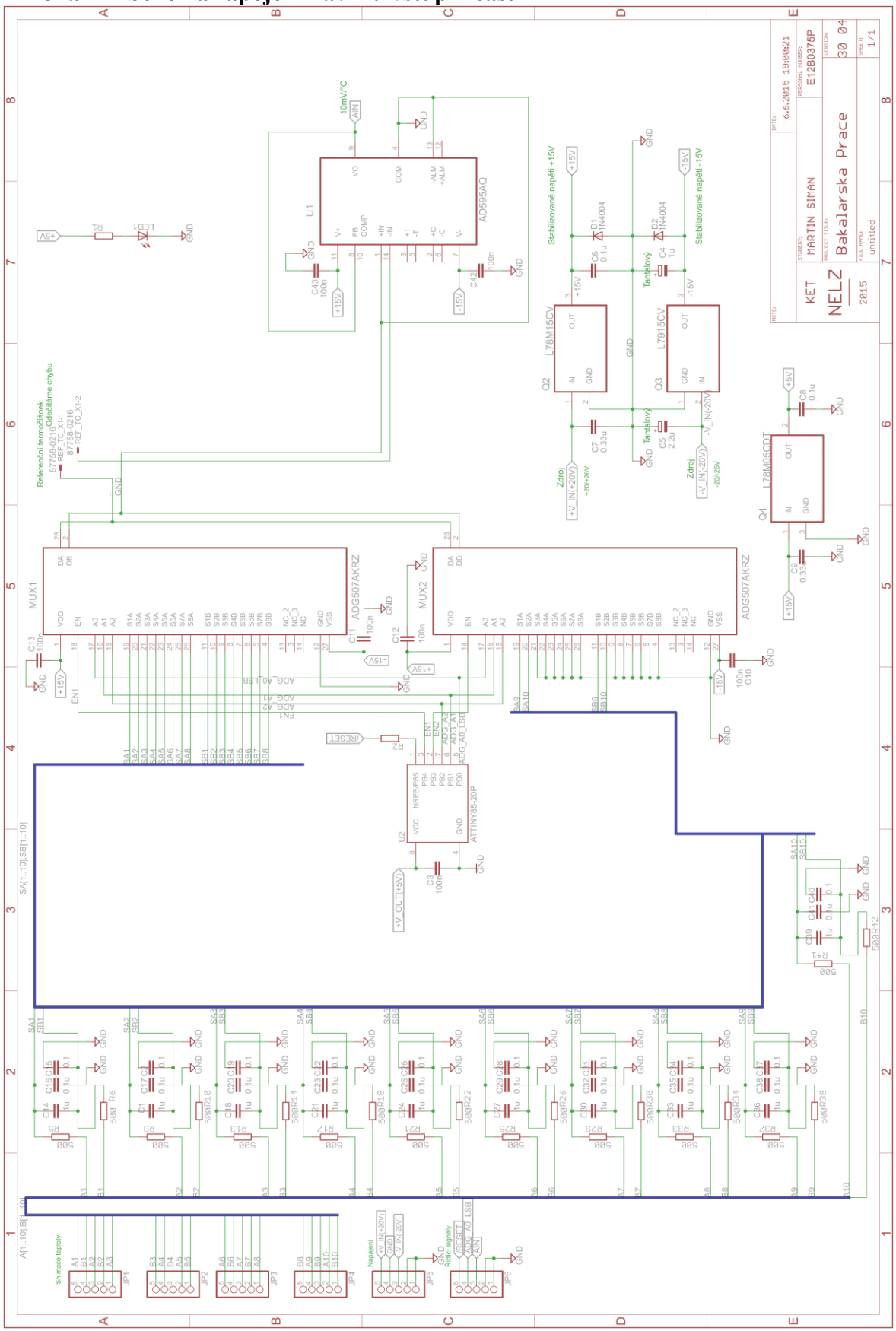
[18] Wikipedia. Multiplexer. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://en.wikipedia.org/wiki/Multiplexer

[19] Wikipedia. Dekodér. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://en.wikipedia.org/wiki/Dekodér

[20] Záhlava, Vít. *Návrh a konstrukce desek plošných spojů,* nakladatelství ČVUT, 2005, 75 s.
[21] Labjack. U3-HV. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://www.labjack.com
[22] RC modely. USB programátor pro procesory atmel. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://www.rc-levneji.cz/prislusenstvi-2/usb-programator-pro-procesory-atmel--10-pin-/
[23] Merkur Robot. Programátor pro řídící jednotky s procesory Atmel. Poslední změna 5.6.2015. [Cit. 5.6.2015]. Dostupné z: http://merkurrobot.cz/?p=

Přílohy

Příloha A – Schéma zapojení návrhu vstupní části



Příloha B – Výpis kódu v jazyce C pro procesor

```
/*****************************************************

Chip type               : ATtiny85
AVR Core Clock frequency: 8,000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 128
*****************************************************/

#include <tiny85.h>
#include <delay.h>

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=0x80;
CLKPR=0x00;
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif

// Input/Output Ports initialization
// Port B initialization
// Func5=In Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State5=T State4=0 State3=0 State2=0 State1=0 State0=0
PORTB=0x00;
DDRB=0x1F;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0A output: Disconnected
// OC0B output: Disconnected
TCCR0A=0x00;
TCCR0B=0x00;
TCNT0=0x00;
OCR0A=0x00;
OCR0B=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Timer1 Overflow Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
PLLCSR=0x00;

TCCR1=0x00;
GTCCR=0x00;
TCNT1=0x00;
```

```
OCR1A=0x00;
OCR1B=0x00;
OCR1C=0x00;

// External Interrupt(s) initialization
// INT0: Off
// Interrupt on any change on pins PCINT0-5: Off
GIMSK=0x00;
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Universal Serial Interface initialization
// Mode: Disabled
// Clock source: Register & Counter=no clk.
// USI Counter Overflow Interrupt: Off
USICR=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
ACSR=0x80;
ADCSRB=0x00;
DIDR0=0x00;

// ADC initialization
// ADC disabled
ADCSRA=0x00;

while (1)
      {

      PORTB.3 = 0;
      PORTB.4 = 1;

      PORTB.0 = 0;
      PORTB.1 = 0;
      PORTB.2 = 0;
        //spoždìní
        delay_ms(100);
      PORTB.0 = 1;
      PORTB.1 = 0;
      PORTB.2 = 0;
        //spoždìní
        delay_ms(100);
      PORTB.0 = 0;
      PORTB.1 = 1;
      PORTB.2 = 0;
        //spoždìní
        delay_ms(100);
      PORTB.0 = 1;
      PORTB.1 = 1;
      PORTB.2 = 0;
        //spoždìní
        delay_ms(100);
      PORTB.0 = 0;
      PORTB.1 = 0;
      PORTB.2 = 1;
        //spoždìní
        delay_ms(100);
      PORTB.0 = 1;
```

```
      PORTB.1 = 0;
      PORTB.2 = 1;
        //spoždìní
        delay_ms(100);
      PORTB.0 = 0;
      PORTB.1 = 1;
      PORTB.2 = 1;
        //spoždìní
        delay_ms(100);
      PORTB.0 = 1;
      PORTB.1 = 1;
      PORTB.2 = 1;
        //spoždìní
        delay_ms(100);
      PORTB.3 = 1;
      PORTB.4 = 0;

      PORTB.0 = 0;
      PORTB.1 = 0;
      PORTB.2 = 0;
        //spoždìní
        delay_ms(100);
      PORTB.0 = 1;
      PORTB.1 = 0;
      PORTB.2 = 0;
        //spoždìní
        delay_ms(100);
      }
}
```

Příloha C – Výpis kódu v jazyce C# pro zobrazování a ukládání hodnot a komunikaci

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using System.IO;
using LabJack.LabJackUD;
using System.Threading;



namespace LabJack_U3_HV_mereni_termoclanku
{
    public partial class Form1 : Form
    {
        private U3 u3;

        public Form1()
        {
            InitializeComponent();
        }
        const double U_CALLIBRATION = 0.25;
// kalibrační hodnota kalibriváno pro 25°C
        const double UAD595_0STUPNU = 0.25;
// výstupní hodnota při 25°C
        const double UAD595_ZISK = 0.01;
// zisk 10mV / °C



        const int RESET_PROCESSOR_FIO_PIN = 4;
// RESET = FIO4    // RESET ATTINY85 Připojit na FIO4
        const int U_OUT_ZESILOVAC_AIN_PIN = 2;
// Uout = AIN2     // U(out) AD595 Připojit na AIN2
        const int LSB_PROCESSOR_FIO_PIN = 5;
// LSB = FIO5      // LSB ATTINY85 Připojit na FIO5
        const int LOGICKA_UROVEN_0 = 0;
// Neměnit !
        const int LOGICKA_UROVEN_1 = 1;
// Neměnit !
        const int POCET_POVOLENYCH_HODNOT = 40;
// Lze modifikovat od 10 do N
// nesmí být větší nez meritko osy x
        const int CELKEM_SNIMACU = 10;
// Neměnit !
        const int MISTO_NA_ULOZISTI_PRI_KTEREM_UZ_NEDOVOLIM_UKLADAT = 10000000;
// <--- n    // 10 MB  // ((n / 1000 ) kB / 1000) MB
// lze modofikovat
        const int GRAF_MERITKO_OSY_Y_MIN = 0;
// Lze modifikovat
        const int GRAF_MERITKO_OSY_Y_MAX = 400;
// Lze modifikovat
        const int GRAF_MERITKO_OSY_X_MIN = 0;
// Lze modifikovat
        const int GRAF_MERITKO_OSY_X_MAX = 40;
// Lze modifikovat


        public bool LED_Power { get; set; }
        public bool LED_Measure { get; set; }
        public bool LED_Write { get; set; }

        Label[] lblTitul = new Label[CELKEM_SNIMACU];
        TextBox[] txtHodnota = new TextBox[CELKEM_SNIMACU];

        string[] lines = new string[CELKEM_SNIMACU];
        String JmenoSouboru = String.Empty;

        bool MereniFirstTime = true;
```

```
        bool FirstTimeError = true;
        bool FirstTimeError2 = true;

        bool MeasureStatus = false;
        bool ConnectStatus = false;


        LJUD.IO ioType = 0;
        LJUD.CHANNEL channel = 0;

        int dummyInt = 0;
        int CurrentThermocouple = 0;
        int Pocitadlo = 0;
        int Stridani_stavu = 0;
        Int64 MistoNaDisku = 0;

        double dblValue = 0;
        double dummyDouble = 0;
        double Value_AIN_DIF = 9999;
        double Skutecna_Hodnota_Ve_Stupnich = 0;
        double ValueDIBit = 9999;
        double Value_DriverVersion;


        public void VynulovaníParamatru()
        {
            dblValue = 0;
            CurrentThermocouple = 0;
            Pocitadlo = 0;

            Ain2_Voltage.Text = "X";
            FI04_Counter1.Text = "X";
            LSB.Text = "X";
        }

        public void Reset_uProcessoru(int DigitalniPort ,int DigitalniUroven)
// Reset (4,0); Reset (4,1); FIO4 log.0/1
        {
            try
            {
//Přidání žádosti - povolit uProcesor k činnosti tzn. RESET=1
// RESET = 2,5V (log.1)  (FIO4, log.hodnota, x, x)
                LJUD.AddRequest(u3.ljhandle, LJUD.IO.PUT_DIGITAL_BIT,
                                DigitalniPort, DigitalniUroven, 0, 0);
            }
            catch (LabJackUDException exc)
            {
                ShowErrorMessage(exc);
                return;
            }

// Vykoná všechny žádosti
            try
            {
                LJUD.GoOne(u3.ljhandle);
            }
            catch (LabJackUDException)
            {
                MessageBox.Show("Reset uProcesoru nešel dle plánu");
                return;
            }
        }


        public void ShowErrorMessage(LabJackUDException exc)
        {
            Label_Error.Text = "Chybové stavy: " + exc.ToString();
            if ((exc.ToString().Contains("no longer")) && FirstTimeError2)
            {
                FirstTimeError2 = false;
                MessageBox.Show("To jste vážně odpojil USB kabel při běhu programu ?!");
                this.Close();
            }
            if ((exc.ToString().Contains("not found")) && FirstTimeError)
            {
                FirstTimeError = false;
```

```
                MessageBox.Show("Prosím připojte USB kabel a k tomu požadovaný U3-HV");
            }
        }

        public void PrecteniVstupu()
        {
            try
            {
//Několik žádosti v jednotlivých low-level funkcí

//Zadaní žádosti - přečtení hodnoty z AIN 2 za použití speciálního rozsahu
                LJUD.AddRequest(u3.ljhandle, LJUD.IO.GET_AIN_DIFF, U_OUT_ZESILOVAC_AIN_PIN,
                0, 32, 0);

//Zadaní žádosti - přečtení digitálního vstupu FIO5
                LJUD.AddRequest(u3.ljhandle, LJUD.IO.GET_DIGITAL_BIT, LSB_PROCESSOR_FIO_PIN,
                0, 0, 0);

            }
            catch (LabJackUDException exc)
            {
                ShowErrorMessage(exc);
                return;
            }
            try
            {
// Vykoná všechny žádosti
                LJUD.GoOne(u3.ljhandle);

// Získám všechny výsledky. Vstupní hodnoty jsou uloženy. Všechny ostatní výsledky jsou pro
// Konfiguraci nebo pro výstupní žádost, takže jenom kontrolujeme, jestli tam není error
                LJUD.GetFirstResult(u3.ljhandle, ref ioType, ref channel, ref dblValue, ref
                dummyInt, ref dummyDouble);
            }
            catch (LabJackUDException exc)
            {
                ShowErrorMessage(exc);
                return;
            }

            bool finished = false;
            while (!finished)
// dokud nevyčteme všechny data v tomhle případě 2
            {
                switch (ioType)
                {
                    case LJUD.IO.GET_AIN_DIFF:
                        Value_AIN_DIF = dblValue;
                        break;

                    case LJUD.IO.GET_DIGITAL_BIT:
                        ValueDIBit = dblValue;
                        break;
                }
                try
                      {
                      LJUD.GetNextResult(u3.ljhandle, ref ioType, ref channel, ref
                      dblValue, ref dummyInt, ref dummyDouble);
                      }
                catch (LabJackUDException exc)
                {
// Pokud nastane vyjímka tak jsme hotový -> konec cyklu
                    if (exc.LJUDError == U3.LJUDERROR.NO_MORE_DATA_AVAILABLE)
                        finished = true;
                    else    // Jestliže nastane jiná -> nahlásit
                        ShowErrorMessage(exc);
                }
            }

// Zobrazení výsledků
            Ain2_Voltage.Text = String.Format("{0:0.###}", Value_AIN_DIF);
            LSB.Text = String.Format("{0:0.###}", ValueDIBit);
            FI04_Counter1.Text = String.Format("{0:0.###}", CurrentThermocouple);

        }
```

```
public void ZpracovaniHodnot()
// Metoda pro graf
        {
            try
            {
                Graf.Series[CurrentThermocouple].Points.Add(Skutecna_Hodnota_Ve_Stupnich);
            }
            catch
            {
                return;
            }


            if (Graf.Series[0].Points.Count > POCET_POVOLENYCH_HODNOT)
// staci 0ty
            {
                for (int i = 0; i < CELKEM_SNIMACU; i++)
                    Graf.Series[i].Points.RemoveAt(0);
            }
            Graf.ResetAutoValues();

        }


        private void Form1_Load(object sender, EventArgs e)
        {
// Inicializace před zahájením připojení atd.

            WindowState = FormWindowState.Maximized;

// this.MaximizeBox = false;
// pokud chci uživateli zakázat zvětšování a zmenšování okna

// Přečte a zobrazí verzi ovladače LABJACKU

            Value_DriverVersion = LJUD.GetDriverVersion();
            Label_DriverVersion.Text = String.Format("Verze ovladače labjack U3-HV:
{0:0.000}", Value_DriverVersion);

            VynulovaníParamatru();

// Programově si vytvořím n Labelů a n Textboxů, usnadnění práce, které dále využiji

            for (int i = 0; i < CELKEM_SNIMACU; i++)
            {
                lblTitul[i] = new Label();
                lblTitul[i].Text = "Napětí: " + (i+1) + ". TC: ";
                TabulkaProHodnoty.Controls.Add(lblTitul[i], 0, i);

                txtHodnota[i] = new TextBox();
                TabulkaProHodnoty.Controls.Add(txtHodnota[i], 2, i);
            }

            if (Checkbox_NastaveniGrafu.Checked == true)
            {
                Graf.ChartAreas[0].Area3DStyle.Enable3D = true;
                Graf.ChartAreas[0].Area3DStyle.Inclination = 45;
                Graf.ChartAreas[0].Area3DStyle.Rotation = 45;
            }

            Graf.ChartAreas[0].AxisX.IsStartedFromZero = true;
            Graf.ChartAreas[0].AxisX.ScaleView.Zoomable = false;
            Graf.ChartAreas[0].AxisX.ScaleView.SizeType =
System.Windows.Forms.DataVisualization.Charting.DateTimeIntervalType.Seconds;
            Graf.ChartAreas[0].AxisX.IntervalAutoMode =
System.Windows.Forms.DataVisualization.Charting.IntervalAutoMode.FixedCount;
            Graf.ChartAreas[0].AxisX.IntervalType =
System.Windows.Forms.DataVisualization.Charting.DateTimeIntervalType.Seconds;
            Graf.ChartAreas[0].AxisX.Interval = 0;

// Nastavení měřítka grafu
            Graf.ChartAreas[0].AxisY.Maximum = GRAF_MERITKO_OSY_Y_MAX;
            Graf.ChartAreas[0].AxisY.Minimum = GRAF_MERITKO_OSY_Y_MIN;

            Graf.ChartAreas[0].AxisX.Maximum = GRAF_MERITKO_OSY_X_MAX;
            Graf.ChartAreas[0].AxisX.Minimum = GRAF_MERITKO_OSY_X_MIN;
```

```
            for (int i = 0; i < CELKEM_SNIMACU; i++)
// defaultně pro 3D graf
                        {
                Graf.Series[i].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;
                         }
            Combo_Box_VyberGrafu.SelectedIndex = 3;

//Graf.Series[0].XValueType =
System.Windows.Forms.DataVisualization.Charting.ChartValueType.String

// Lokalizace místa na disku
            System.IO.DriveInfo di = new System.IO.DriveInfo(@"C:\");
            Label_MistoNaDiskuHodnota.Text = di.TotalFreeSpace.ToString();


// Program uProcesoru začne hned běžet proto já ho při zapnutí tohoto programu vypnu
            try
            {
                u3 = new U3(LJUD.CONNECTION.USB, "0", true);    // Připojení skrz USB
                LJUD.ePut(u3.ljhandle, LJUD.IO.PIN_CONFIGURATION_RESET, 0, 0, 0);
            }

            catch (LabJackUDException exc)
            {
                ShowErrorMessage(exc);
                return;
            }
            Reset_uProcessoru(RESET_PROCESSOR_FIO_PIN, LOGICKA_UROVEN_0);

        }

        private void Timer_Stream_Tick(object sender, EventArgs e)
        {
            PrecteniVstupu();

            ZpracovaniHodnot();

            Graf.Update();

        }



        private void Click_Connect(object sender, EventArgs e)
// navázání připojení s Kartou
        {
                try
                {
//Otevře první nalezený Labjack U3
                    u3 = new U3(LJUD.CONNECTION.USB, "0", true);
// Připojení skrz USB


// Vždy začínáme tím, že konfiguraci pinu dáme do defaultního nastavení
                    LJUD.ePut(u3.ljhandle, LJUD.IO.PIN_CONFIGURATION_RESET, 0, 0, 0);

// Configurace FIO0-3 jako Analog ostatní jako digital.
// Začínáme z kanálu (channel) 0 a obnovíme všech 16 flexibilních bitů
// Překročíme hodnotu b0000000000001111 nebo d15.
                    LJUD.ePut(u3.ljhandle, LJUD.IO.PUT_ANALOG_ENABLE_PORT, 0, 15, 16);
// ePut a eGet combinujou funkce AddRequest, Go a GetResult v jednom kroku
                }

                catch (LabJackUDException exc)
// pokud uživatel nepřipojí USB kabel například
                {
                    ShowErrorMessage(exc);
                    MessageBox.Show("Bohužel musíte připojit USB kabel");
                    Button_Mereni.Enabled = false;
                    return;
                }

                VynulovaniParamatru();

                if (!ConnectStatus)
// pokud není připojeno - připojím
```

```
                {
                    Button_Mereni.Enabled = true;
                    LEDka1.LED_Power = true;
// povolím algoritmus pro změnu barvy v LEDka.cs

                    Label_ConnectionStatus.Text = "Status: Připojeno";
                    ConnectStatus = true;
                    Button_Connect.Text = "Ukončit připojení";


                }
                else
// pokud je připojeno - odpojím (když se snažím mačkat tlačítko)
                {
                    Button_Mereni.Enabled = false;
                    LEDka1.LED_Power = false;

                    Label_ConnectionStatus.Text = "Status: Odpojeno";
                    ConnectStatus = false;
                    Button_Connect.Text = "Zahájit připojení";
                    u3 = null;
// zahodím jeho parametr
                }
        }

        private void Button_ZahajitMereni(object sender, EventArgs e)
        {
            if (MereniFirstTime)
               VynulovaníParamatru();

            if (!MeasureStatus && ConnectStatus)
// Zahajit měření
            {
                Button_FlipFlop.Enabled = false;
                Button_ProtocolClear.Enabled = false;


                LEDka1.LED_Measure = true;
// povolím algoritmus pro blikání LED v souboru LEDka.cs

                Reset_uProcessoru(RESET_PROCESSOR_FIO_PIN, LOGICKA_UROVEN_1);
// fyzicky povolím činnost uProcesoru


                Button_Connect.Enabled = false;

                Button_Mereni.Text = "Ukončit měření";
                Timer_Stream.Enabled = true;
// a povolím časovač, který mi vzorkuje hodnoty


                MeasureStatus = true;
                VynulovaníParamatru();
            }
            else
// Ukončit měření
            {
                Button_FlipFlop.Enabled = true;
                Button_ProtocolClear.Enabled = true;

                LEDka1.LED_Measure = false;
// zakážu blikání LED

                Reset_uProcessoru(RESET_PROCESSOR_FIO_PIN, LOGICKA_UROVEN_0);
// na pin Reset uProcesoru přivedu log. 0 tzn. blokování činnosti


                Button_Connect.Enabled = true;

                Button_Mereni.Text = "Zahajit měření";
                Timer_Stream.Enabled = false;
                MeasureStatus = false;
                VynulovaníParamatru();

                lbData.Items.RemoveAt(lbData.Items.Count - 1);
            }
        }
```

```
        private void LSB_TextChanged(object sender, EventArgs e)
// reaguji na změnu LSB
        {
            Value_AIN_DIF = Math.Round(Value_AIN_DIF, 4);
            txtHodnota[CurrentThermocouple].Text = Value_AIN_DIF.ToString();
// každému textboxu náleží právě jedna hodnota

            AddProtocol(Value_AIN_DIF.ToString());
// tu zároveň předám do AddProtocol algoritmu

            if (LSB.Text == "0" || LSB.Text == "1")
// reaguji na změnu LSB
                CurrentThermocouple += 1;
// posun

            if (CurrentThermocouple == CELKEM_SNIMACU)
// přetečení
                CurrentThermocouple = 0;


        }

//int Pocitadlo = 0;



        private void AddProtocol(string Data)
        {
         Pocitadlo++;
// počítám měření
         Skutecna_Hodnota_Ve_Stupnich = (double.Parse(Data) - U_CALLIBRATION) / UAD595_ZISK;
// Zápis do protokolu
            lbData.Items.Add(String.Format(" {0:HH:mm:ss}; {1}. Termoclanek; Napeti: {2};
Teplota: {3}; ", DateTime.Now, CurrentThermocouple + 1, Data,
Math.Round(Skutecna_Hodnota_Ve_Stupnich, 2)));

            if ((JmenoSouboru != String.Empty) && (Pocitadlo == CELKEM_SNIMACU))
            {
                for (int i = CELKEM_SNIMACU; i != 0; i--)
// každý 10-tý měření (v závislosti na počtu snimacu) přiřadím poli stringů právě naměřené
// hodnoty
                {
     lines[Math.Abs(i - CELKEM_SNIMACU)] = lbData.Items[lbData.Items.Count - i].ToString();
// zároveň setřídím
                }
               try
                {
                    File.AppendAllLines(JmenoSouboru, lines);
// otevřu soubor připíšu 10 hodnot právě ty pole stringů a zavřu
                }
                catch(System.IO.IOException)
// pokud se někdo snaží přistupovat k souboru z jiného zdroje
                {
                    Button_Mereni.PerformClick();
// vynutím ukončení měření
                    MessageBox.Show("Nemužu zapisovat hodnoty, když si je právě teď
                    prohlížíte. Ukončuji měření");
                }

                Pocitadlo = 0;
// vyresetuji pocitadlo
                System.IO.DriveInfo di = new System.IO.DriveInfo(@"C:\");
// obnovím volné místo na disku C
                Label_MistoNaDiskuHodnota.Text = di.TotalFreeSpace.ToString();

                Stav.Text = "Zapisuji hodnoty";

            }
            if (Pocitadlo == 2)
// fake zpoždění, aby to to uživatel stihl přečíst
                Stav.Text = "V klidu";

            lbData.SelectedIndex = lbData.Items.Count - 1;
// vždy chci vidět poslední hodnotu v protokolu
            lbData.SelectedIndex = -1;
```

```
            if (lbData.Items.Count > POCET_POVOLENYCH_HODNOT)
// pokud mi počet prvků (řádků) přesáhne pocet povolenych hodnot mažu vždy první
            {
                    lbData.Items.RemoveAt(0);
            }
        }


        private void souborToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            if(Int64.TryParse(Label_MistoNaDiskuHodnota.Text, out MistoNaDisku))
// bezpečnější převod stringu na int
            {
                if (MistoNaDisku <= 10000000)
// pokud je místo na disku menší jak 10MB tak neukládám
                {
                 MessageBox.Show("Máte málo místa na disku, proto hodnoty nebudu ukládat");
                 return;
                }
            }
            else
            {
                MessageBox.Show("Nepodařil se převod");
            }

            using (SaveFileDialog sfd = new SaveFileDialog())
            {
                sfd.Filter = "Soubory CSV (*.csv)|*.csv| Vsechny soubory (*.*)|*.*";

                if (sfd.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    JmenoSouboru = sfd.FileName;
Label_JmenoSouboru.Text = "Cesta k souboru do kterého ukládám data: " + JmenoSouboru;
                }
            }
            if (String.IsNullOrEmpty(JmenoSouboru))
            {
                MessageBox.Show("Jméno souboru je prázdné nebo nulové");
                return;
            }

        }

        private void funkceProgramuToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show(" Pro ukládání do souboru zvolte položku zápis dat do souboru.
\n Zvolte jméno pro budoucí soubor. Primárně je nastavena koncovka .CSV \n Nedoporučuji
měřit velmi dlouho. \n Může se stát, že o svoje hodnoty přijdete. \n Zápis do souboru se
vždy provádí po desátém měření. ");
        }

        private void navrhnulToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Verze 1.00 Měření teploty Martin Šiman");
        }




        private void Button_NouzovyReset_Click(object sender, EventArgs e)
        {
            if (u3 == null)
            {
    MessageBox.Show("Ztratil jsem referenci na USB připojení, tak já tedy  navazuji zpět");
                try
                {
                    u3 = new U3(LJUD.CONNECTION.USB, "0", true);
                }
                catch
                {
                    MessageBox.Show("Bohužel musíte připojit USB kabel");
                    return;
                }
            }
            if ((Stridani_stavu == 0) || (Stridani_stavu == 1))
                Reset_uProcessoru(RESET_PROCESSOR_FIO_PIN, Stridani_stavu);
            Stridani_stavu++;
```

```
            if (Stridani_stavu == 2)
                Stridani_stavu = 0;
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (Label_Error.Text == "Chybové stavy: ")
// pokuď je všechno v pohodě (nejsou žádné chybové stavy) tak mohu zastavit činnost
                Reset_uProcessoru(RESET_PROCESSOR_FIO_PIN, LOGICKA_UROVEN_0);
//this.Close();
        }

        private void Form1_FormClosed(object sender, FormClosedEventArgs e)
        {

        }

        private void Button_ProtocolClear_Click(object sender, EventArgs e)
        {
            if (lbData.Items.Count == 0)
                MessageBox.Show("To nedává smysl !");
            lbData.Items.Clear();
        }

        private void Checkbox_NastaveniGrafu_CheckedChanged(object sender, EventArgs e)
        {
            if (Checkbox_NastaveniGrafu.Checked == true)
            {
                Graf.ChartAreas[0].Area3DStyle.Enable3D = true;
                Graf.ChartAreas[0].Area3DStyle.Inclination = 45;
                Graf.ChartAreas[0].Area3DStyle.Rotation = 45;
            }
            else
            {
                Graf.ChartAreas[0].Area3DStyle.Enable3D = false;
            }
        }

        private void Combo_Box_VyberGrafu_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (Combo_Box_VyberGrafu.SelectedIndex == 0)
            {
                for (int i = 0; i < CELKEM_SNIMACU; i++)
                {
                    Graf.Series[i].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;
                }
            }
            if (Combo_Box_VyberGrafu.SelectedIndex == 1)
            {
                for (int i = 0; i < CELKEM_SNIMACU; i++)
                {
                    Graf.Series[i].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.FastPoint;
                }
            }
            if (Combo_Box_VyberGrafu.SelectedIndex == 2)
            {
                for (int i = 0; i < CELKEM_SNIMACU; i++)
                {
                    Graf.Series[i].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
                }
            }
            if (Combo_Box_VyberGrafu.SelectedIndex == 3)
            {
                for (int i = 0; i < CELKEM_SNIMACU; i++)
                {
                    Graf.Series[i].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;
                }
            }
            if (Combo_Box_VyberGrafu.SelectedIndex == 4)
            {
                for (int i = 0; i < CELKEM_SNIMACU; i++)
                {
                    Graf.Series[i].ChartType =
```

```
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.StepLine;
                }
            }
            if (Combo_Box_VyberGrafu.SelectedIndex == 5)
            {
                for (int i = 0; i < CELKEM_SNIMACU; i++)
                {
                    Graf.Series[i].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.FastLine;
                }
            }
            if (Combo_Box_VyberGrafu.SelectedIndex == 6)
            {
                for (int i = 0; i < CELKEM_SNIMACU; i++)
                {
                    Graf.Series[i].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Area;
                }
            }
        }


    }

}
```

Příloha D – Třída LEDka k Příloze C

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Drawing.Drawing2D;

namespace LabJack_U3_HV_mereni_termoclanku
{
    public partial class LEDka : Control
    {
        Timer TimerObnovovaniObrazu = null;


        public LEDka()
        {
            InitializeComponent();

            DoubleBuffered = true;

            TimerObnovovaniObrazu = new Timer();
            TimerObnovovaniObrazu.Interval = 500;
            TimerObnovovaniObrazu.Tick += tmr_ObnovovaniObrazu;
            TimerObnovovaniObrazu.Enabled = true;
            TimerObnovovaniObrazu.Start();
        }


        int Stridani_On_Off = 0;
        private void tmr_ObnovovaniObrazu(object sender, EventArgs e)
        {
            Stridani_On_Off++;
            if (Stridani_On_Off >= 100)
                Stridani_On_Off = 0;
            this.Invalidate();
        }




        public bool LED_Power = false;
        public bool LED_Measure = false;
        public bool LED_Write = false;

        protected override void OnPaint(PaintEventArgs pe)
        {
            Graphics Grafika = pe.Graphics;
            Rectangle PracovniPlocha = this.ClientRectangle;

            Point Pocatek = new Point(0,0);
            Point Konec = new Point(PracovniPlocha.Width, PracovniPlocha.Height);

            if (LED_Write)
            {
                Grafika.FillEllipse(Brushes.Blue, 0, 0, PracovniPlocha.Width, P
       racovniPlocha.Height);
            }
            else
            {
                Grafika.FillEllipse(Brushes.White, 0, 0, PracovniPlocha.Width,
PracovniPlocha.Height);
            }


            if (LED_Power)
            {
                if (LED_Measure)
                {
                    if ((Stridani_On_Off % 2) == 0)
                        Grafika.FillEllipse(Brushes.Red, 0, 0, PracovniPlocha.Width,
PracovniPlocha.Height);
```

```
                    else
                        Grafika.FillEllipse(Brushes.White, 0, 0, PracovniPlocha.Width,
PracovniPlocha.Height);
                }
                else
                    Grafika.FillEllipse(Brushes.Red, 0, 0, PracovniPlocha.Width,
PracovniPlocha.Height);
            }
            else
                Grafika.FillEllipse(Brushes.Green, 0, 0, PracovniPlocha.Width,
PracovniPlocha.Height);

        }
    }
}
```