Středoškolská technika 2016

Setkání a prezentace prací středoškolských studentů na ČVUT

OPEN-SOURCE TEPLOMĚR

Adam Bohbot

Lauderova mateřská škola základní škola a gymnázium
při Židovské obci v Praze
Praha, Belgická 25

Lauderova MŠ, ZŠ A gymnázium při ŽOP

OPEN-SOURCE TEPLOMĚR

Adam Bohbot, 7. O

Vedoucí práce: Šárka Kvasničková

2015/2016

Abstract

This seminar worked on creating a thermometer from available electronics and explaining it. I used Arduino hardware and Processing and Arduino software. With these two programs, I was able to create a working thermometer with a range from -50 to 150 degrees Celsius but the code of the Processing software was created to measure human temperature only.

Abstrakt

Tato seminární práce se zabývala vytvořením funkčního teploměru z dostupné elektroniky a vysvětlení jeho principu. K sestrojení bylo použito Arduino a k programování programy Arduino a Processing. S těmito dvěma programy byl úspěšně naprogramován teploměr s měřícím rozmezím od -50 do 150 ^{o}C. Program Processing a jeho kód v našem případě umožnil měření výhradně tělesné teploty u člověka.

Key words / klíčová slova

Thermometer, open-source electronics, Arduino, Processing, programming

Teploměr, open source elektronika, Arduino, Processing, programování

Úvod

Tato seminární práce se zabývá open-source elektronikou, teploměry a Arduinem. Cílem práce je sestrojit funkční teploměr z Arduina a open-source elektroniky, naprogramovat ho a otestovat. Použiji 2 důležité programy. Prvním z nich bude Arduino, které bude komunikovat s hardwarovou složkou obvodu. Druhý je program Processing, který bude zpracovávat data a vykreslovat graf. Studovat k tomu budu jazyk Arduino a Java, který je velmi podobný jazyku, který využívá Processing. Předpokládám, že výsledky měření nebudou úplně přesné, ale pokusím se pomocí zapojení různých odporů do obvodu dosáhnout optimálních výsledků.

Výsledkem z teoretické části by měly být jasné a stručné informace o open-source elektronice, Arduinu, a jejich využití. Z praktické části by výstupem měl být fungující teploměr, naměřená data a jejich interpretace.

1. Open-source elektronika

1.1 Open-source

„Open-source“ jednoduše v překladu znamená „dostupný“ či „otevřený“. V případě open-source software se jedná o software vyvinutý za účelem nahradit existující placený program programem s podobnou funkcí, který je zadarmo. Nejedná se však pouze o elektroniku, ale existují například open-source Coly, tedy nápoje ve stylu Coca-Coly vyvíjené konkurenčními značkami za použití dobrovolníků, kteří vybírají nejlepší možnou příchuť z velké nabídky. Trochu známějším příkladem „Open-source“ je například stránka Wikipedia, která je kompletně dostupná i otevřená. Každý má možnost ji upravit, či připsat další informace, což z ní činí open-source informační zdroj.

Zaměříme-li se opět na elektroniku – i mobil má možnost být open-source. Většina moderních smartphonů má vývojářský režim, který umožňuje větší kontrolu nad zařízením. V rukou odborníka to může vést k optimalizaci nastavení a tedy zrychlení výkonu a úspoře baterie, naopak v rukou nezručného experimentátora může být výsledkem zablokování funkcí, zpomalení chodu, nebo až totální kolaps zařízení a nutnost uvést jej do továrního nastavení. Open-source elektronika není výjimkou. V rukou zručného nadšence může být výstupem nekonečná řada užitečných zařízení, zatímco v opačném případě nemusí fungovat nic. Open-source elektronika je jednoduše řečeno elektronika dostupná ke koupi v běžných obchodech – počínaje breadboardem (stavební desky pro elektrické obvody), přes kabely až po součástky na první pohled nesrozumitelných jmen jako LM35 (technické jméno teploměru, kterým se budu dále zabývat).

1.2 Typy open-source elektroniky

Z open-source elektroniky se dají sestrojit různá zařízení. Všechny výrobky a vynálezy můžeme třídit do dvou jednoduchých skupin. První, které nepotřebují k chodu či užitečnosti počítač nebo zařízení, které bude zpracovávat výstup, a druhé, které jsou bez počítače nepoužitelné.

1.2.1 Nezávislé na počítači

Některá ze zařízení nepotřebují k chodu počítač. Mezi ně patří například rádio, mikrofon nebo lampy. Spojuje je fakt, že nepotřebují větší výpočetní prostor pro ukládání či zpracování dat, které přijmou. I tato zařízení však potřebují elektrický zdroj a nějaké nastavení. Měnit frekvenci rádia není na první pohled složité, stačí jen točit knoflíkem nebo zmáčknout tlačítko. Aby však rádio samo našlo čisté frekvence, k tomu už je potřeba počítač.

1.2.2 Závislé na počítači

Další přístroje už jsou trochu složitější z hlediska funkčnosti (ne vždy sestrojení). Například teploměr, kterým se budu zabývat později v práci, bude počítat data vždy (pokud je správně sestrojen). Tato data však nepůjde nijak vyčíst, nebudu-li mít tento teploměr napojený na počítač, který bude přijímat data z teploměru a zpracovávat je do grafu. Na první pohled je to jednoduché, ale nejjednodušší cesta je přes 2 počítačové programy – jeden, který bude přijímat data ve formě napětí ve voltech (V) a další, který toto číslo přepočítá na stupně Celsia (°C), zapíše je a zároveň z těchto čísel vykreslí graf. Na podobném principu pracuje například EKG (přístroj na měření srdeční aktivity) nebo osciloskop (přístroj na měření elektrického napětí). Oba tyto přístroje potřebují počítač a displej.

1.2.3 Arduino

Arduino je open-source platforma. Je to „součástka“ spojující přístroje nezávislé i závislé na PC. Může sloužit jako breadboard, jelikož přijímá signál, ale může také tento signál vyslat do počítače přes USB kabel (nebo jiný druh kabelu). Arduino bylo poprvé představeno v roce 2005 a od té doby prošlo mnoha změnami, kdy každá z nich rozšířila možnosti jeho použití [1]. Může sloužit jako náhrada počítače, propojení s displejem, nebo propojení s počítačem, díky mnoha dostupným funkcím, tlačítkům, diodám a vstupům pro vodiče. Kromě Arduina, který je vlastním hardwarem, existuje také stejnojmenný software pro počítače, který po zkontrolování zadaného kódu ovládá napojené zařízení a zpracovává data, která z něj počítač přijímá.

Některé z naprogramovatelných funkcí jsou například frekvence přijímání signálů počítačem, upřednostňování signálů, rozsvěcování diod, stanovení podmínek funkcí atd. Arduino má mnoho využití, tato práce se jím bude zabývat podrobněji později v praktické části, kde budou ukázky softwaru i hardwaru.

2. Arduino

2.1 Princip

Arduino se dá pokládat za malý počítač (přesněji jednodeskový), vývojovou platformu, nebo jen součástku potřebnou ke komunikaci mezi elektrickým obvodem a počítačem [2]. Není možné s ním však interagovat přímo skrze myš či klávesnici, ale při propojení přes USB kabel s počítačem mu lze zadat příkazy, které bude poslouchat a opakovat. Umí reagovat na čipy napojené na něj, ovládat LED diody, přijímat data z různých zadaných zdrojů (stejně jako teploměr použitý v této práci) a mnoho dalšího. Další jeho nezbytnou součástí jsou vstupy a výstupy uzemnění a zdrojů, analogové piny (vstupy/výstupy Arduina), a další, které však k experimentu s měřením teploty nebudu potřebovat. Analogových pinů má klasická deska Arduino 6 (0-5). Každý z nich může najednou přijímat a zpracovávat různé informace, k našemu teploměru stačí jen jeden. Arduino říká pinům díky zadanému kódu, kolikrát informace přijímat, v mém případě každých 50 ms, tedy 20 krát za vteřinu. Arduino poté tyto informace přepočítá na volty a tuto informaci pošle dále. K přepočítání používá bity (více v kapitole 3.4 Matematika softwaru a 4.1 Postup, popis subjektů). Výhody Arduina oproti jiným platformám jsou například jednoduchá programovatelnost, spousta online návodů, aktivní komunita a možnost použití libovolného operačního systému počítače.

2.2 Teploměr

Teploměr jako takový je v našem případě pouze součástka jménem LM35 (celým jménem LM35 ±0.5 °C teplotní sensor s analogovým výstupem a 30V kapacitou). LM35 je součástí série analogových teploměrů. Součástka sama o sobě nic neměří, dokud není správně zapojena a nastaveno její chování. Teploměr LM35, který používáme pro měření, je analogový teploměr s lineárním průběhem [3]. Funguje na principu změny odporu při změně teploty. Změna teploty je přímo úměrná změně napětí na výstupu teploměru. Z tohoto napětí software vypočítá tělesnou teplotu. Asi nejvhodnějším umístěním pro měření je v podpaží, kde se dá teploměr dobře uzavřít a dojde

k dobrému měření. Dokáže změřit teploty v rozmezí -50 °C a 150 °C s chybovostí 0,5 °C. Při zapojování je připojen k napětí 5 V, uzemněn a výstupní signál je zesílen operačním zesilovačem, aby se využil plný rozsah ACD převodníku Arduina, ke kterému je výstup připojen. Arduino je napojeno na počítač.

2.3 Software

Po dokončení hardwaru, je potřeba říct počítači, co s ním má dělat. Arduino má vlastní software pod stejným názvem, který plní jednoduchou funkci – informuje počítač, aby přijímal data z Arduina v jistém intervalu (v našem případě 50ms – tedy 20x za vteřinu. Počítač teď tedy přijímá data, ale nic s nimi nedělá. Tato data jsou automaticky v rozmezí 0 – 5 V (voltů) přepočítána na počítačovou jednotku „bity“. Pro počítač je jednodušší zpracovávat data v této jednotce především proto, že má problém s desetinnými čísly, která přijímá z Arduina ve voltech. Proto Arduino software automaticky posílá počítači data přepočítaná na 0 – 1023 bitů. Protože Arduino nedokáže bity zpracovat, je zapotřebí použít program Processing, který umí vykonávat matematické operace.

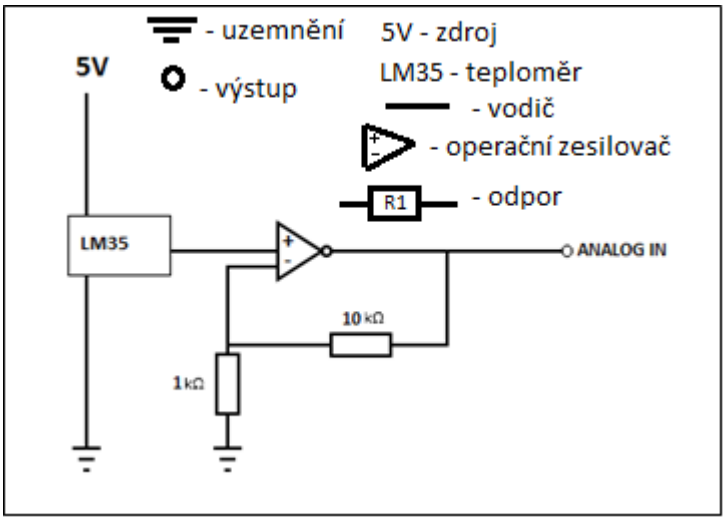
2.4 Matematika softwaru

Data přijatá Arduinem a předaná počítači nejsou automaticky ve stupních Celsia (°C). Je potřeba tedy naprogramovat nějaký přepočet z původního napětí ve voltech (které se pohybují v rozmezí od 0 do 5). Tyto volty jsou automaticky přepočítány na bity, od 0 do 1023. Musíme vzít v potaz také operační zesilovač, který signál zesiluje, takže je třeba výsledek vydělit přesným zesílením signálu, aby nedošlo ke zkreslení dat. Hodnota této konstanty je 9.2 (vypočítáno přes zapojení rezistorů v zesilovači), takže celkový počet bitů je vydělen touto konstantou. Po všech uvedených úpravách a matematických operacích by měl být přijatý signál přepočítán na stupně Celsia.

3. Sestrojování teploměru

3.1 Postup, ukázky

Při stavbě je možné položit si otázku „čím začít“, zda hardwarem (tedy stavbou obvodu dle schématu, jehož funkčnost však nebude možné ověřit kvůli absenci softwaru), nebo softwarem (tedy programem, který však nebudeme moci vyzkoušet kvůli absenci hardwaru). Z vlastních zkušeností preferuji začít ve většině případů hardwarem, neboť software je na něm více závislý – alespoň v tomto případě. Původním zadáním bylo změřit data a poslat je počítači, který je zpracuje. Potřebujeme tedy nejprve data a až poté počítač, který je bude zpracovávat. A ať už bude náš obvod posílat data správná, nebo ne, náš program nám to zjistí. Začal jsem tedy sestavováním teploměru na breadboard dle schématu (viz obrázek 1).



Obrázek 1 - schéma obvodu – zdroj: Talnet

Na obrázku 1 můžeme vidět několik součástek. LM35 je teploměr, který používám v sestavovaném obvodu. Trojúhelníkový tvar je takzvaný operační zesilovač, který zesiluje signál z teploměru 9.2 krát, aby využil převodník Arduina na plný potenciál. Tím je myšleno, že větší rozdíly mezi výchozími bity umožňují větší přesnost měření. ANALOG IN je výstup analogového pinu Arduina, ze kterého jsou informace posílány do počítače a zpracovány softwarem.

3.2 Vysvětlení softwaru

Processing je známý open-source programovací jazyk vyvinutý k designu, umění i zpracování dat. Stejnojmenný program je vybavený mnoha jednoduchými grafickými funkcemi. Především jeho přehledný souřadnicový systém je schopen vytvořit velmi komplikované grafické nástroje díky zadání z kódu. Jeho hlavní součástí, kterou potřebuji, je funkce „serial“[4]. Tato funkce přijímá data z Arduina, které si najde na určitém portu (USB portu v našem případě). Jelikož se jedná o proměnlivý zdroj dat, program potřebuje znát rozmezí mezi „screen-updaty“, tedy frekvence vykreslování dat. Nám stačí nechat původních 50 ms, tedy frekvence 20 Hertzů. Celý kód v Processingu má přes 7000 znaků, zatímco kód v Arduinu pouze 400. Processing na rozdíl od Arduina nepřijímá jen data, musí ale také vykreslovat graf, a k tomu potřebuje informace o souřadnicích x a y, barvě, rychlosti přijímání dat, a pro zestetičtění byla připojena funkce automatického ukončení programu, když se měřená teplota ustálí.

3.3 Plán měření

Měření bude probíhat následovně – se svým funkčním teploměrem změřím alespoň 10 lidí, každého člověka 3krát pro přesnější výsledky. Abych měl s čím porovnávat, budu měřit také obyčejným dostupným elektronickým teploměrem (který ukazuje desetinná čísla), abych viděl co nejpřesnější odchylku mezi výsledky obou teploměrů. Výsledky pak vložím přes Microsoft Excel do tabulek a grafů, které přiložím k práci.

3.4 Možné zdroje problémů

Při stavbě může dojít ke spoustě problémů, které by mohly prodloužit či ohrozit stavbu teploměru. Zcela nejdůležitější je porozumění programovacím jazykům Arduino a Processing. Bez nich není možné pokračovat dále. V každém z nich je možné se ztratit, každý z nich funguje na jiném principu. Naštěstí však program hlásí případné chyby i s očíslovanými řádky, na kterých vypíše, které frázi nerozumí, nebo co nemůže splnit.

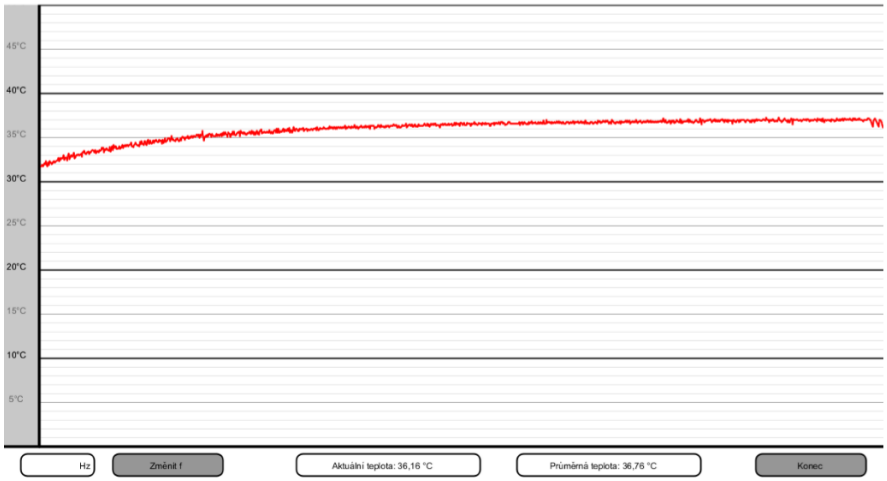
Je také třeba znát základy počítačových USB portů, přenosu dat a kompilace kódů (proces skládání částí kódu k sobě ve snaze zmenšit velikost souboru). Hardware má mnohem rozmanitější problémové oblasti – zaprvé znalost čtení a sestavování elektrického obvodu dle schématu. Schéma použité v této seminární práci není na první pohled problematické, ale jak je možné dočíst si v kapitole 5. Výsledky měření, i zde nastal problém, základní a neprůchodný. Po celou dobu měření je třeba zálohovat data, protože programy Arduino ani Processing samy od sebe zálohy nedělají. U schémat také není vždy napsané přesné jméno součástky, doporučuje se ověřit si u zkušenějších techniků účinnost a použitelnost součástky před zakoupením či vestavěním do obvodu. Stejně je potřeba dávat si pozor na používané odpory, aby v mém případě operační zesilovač fungoval správně. U softwaru je také důležité mít předem danou představu o výsledku, kterého chceme programováním dosáhnout – grafická podoba, funkční podoba a funkce vložené do kódu. Tato seminární práce na mnohé z těchto problému narazila, což se výrazně podepsala na výsledném výstupu.

4. Měření

4.1 Postup, popis subjektů

Jak bylo již řečeno v kapitole Arduino - software, je potřeba další program, který bude data zpracovávat a vykreslovat graf. Tento program bude v našem případě Processing. Slouží k interaktivnímu vykreslování dat, ukázce matematických tvarů a dalších grafických prvků. Použijeme ho proto, že umí jednoduše zpracovávat data z Arduino softwaru. Pomocí kódu vně programu vytvoříme systém pro přepočítávání již vysvětleného bitového přepočtu na stupně Celsia za pomoci známých faktů o teploměru LM35, který používáme. Víme, že umí měřit od -55 do 150 ^{o}C. To znamená, že bity budou rozprostřeny na stejné škále. Musíme však vzít v úvahu operační zesilovač, který zesiluje signál, aby byl čitelný a přesný pro odpory ve schématu. V našem případě to je cca 9.2, tedy tímto číslem musí kód výsledek vydělit, aby údaje nebyly zkreslené. Processing tato data zakreslí do naprogramovaného grafu, který se pro naše účely pohybuje v dostatečném měřítku od 0 do 50 ^{o}C. Po spuštění se otevře okno, ve kterém je vidět graf, průměrná teplota a aktuální teplota. Aktuální teplota se zobrazuje v dolním pravém rohu a program také porovnává posledních 20 přijatých čísel. Je-li rozdíl mezi nimi méně než půl stupně, program zastaví přijímání dat a vykreslování grafu a my získáme výslednou teplotu měřeného subjektu.

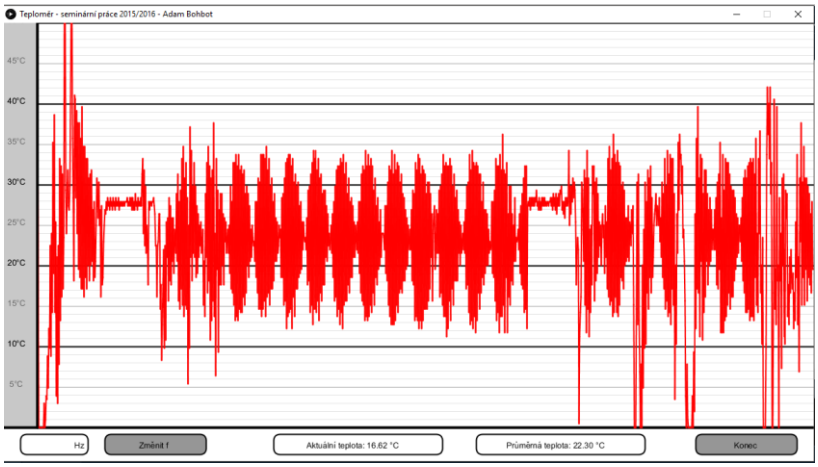
4.2 Ukázka měření



Obrázek 2 – graf v Processingu – zdroj: autor

Na obrázku 2 je správně vypadající graf vytvořený v Processingu. Červená linka znázorňuje naměřenou teplotu, je na ní vidět změna z pokojové teploty (či teploty při nastartování programu) až po ustálení. Tato data však nebyla naměřena, nýbrž napsána do textového souboru, který program přečetl a vykreslil. Tato zkouška dokázala, že program vykresluje graf správně podle zadaných informací a i správně převádí jednotky z bitů na stupně Celsia. I funkce „konec“ a průměrná teplota fungují správně.

I přes původní technický neúspěch se podařilo vytvořit funkční program na zpracování dat z Arduina, zprvu se však na neznámém místě objevila chyba a data byla nepoužitelná k přesnému měření. Takto vypadalo měření s reálnými daty přijatými z Arduina v reálném čase.



Obrázek 3 – naměřená data z prvních několika verzí teploměru – zdroj: autor

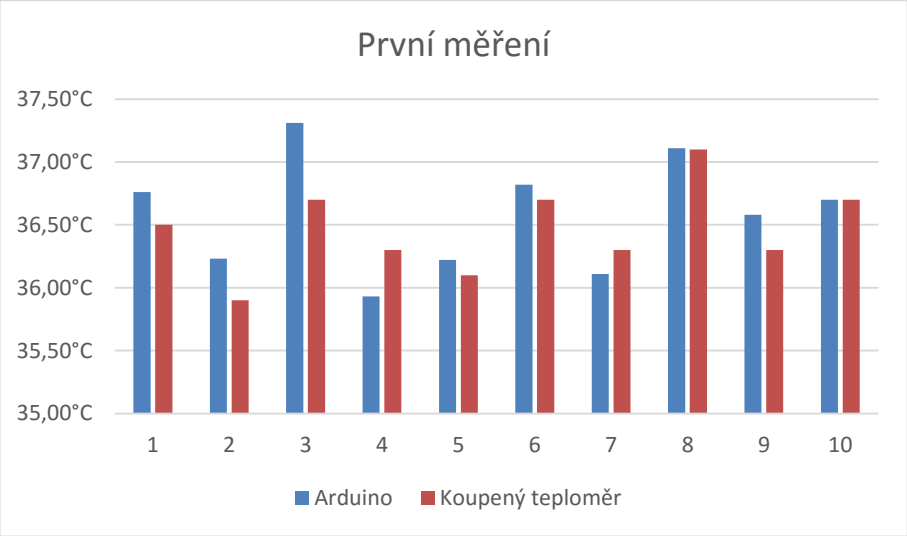
Data na obrázku 3 spíše připomínají nepovedený pulsní oxymetr (přístroj k měření lidského pulsu v aktuálním čase), nežli teploměr, a ke konci grafu je šum způsoben pravděpodobně pohybem s Arduinem a nepřesným měřením. Na druhou stranu je vidět, že program umí počítat aktuální teplotu i průměrnou teplotu, což znamená, že chyba je pravděpodobně na straně hardwaru.

Po konzultaci s odborníkem[1] na techniku a Arduino byly objeveny následující chyby na zapojení hardwaru – operační zesilovač nebyl připojen ke zdroji, tedy nemohl zesílit signál a vysílal velmi zkreslená data. Dále jeden z odporů, který měl být uzemněn, neustále vypadával kvůli špatnému zapojení a data byla zkreslená o to víc. Nesmyslná data v grafu na obrázku 3 byla tedy způsobena výše zmíněnou chybou v zesilovači, která vedla k tomu, že měřené napětí bylo to v zásuvce nebo notebookové baterii a danému proudění.

[1] Informace poskytl: Lukáš KYZLÍK, student, odborná pomoc dne 20. 4. 2016 a 22. 4. 2016, Praha

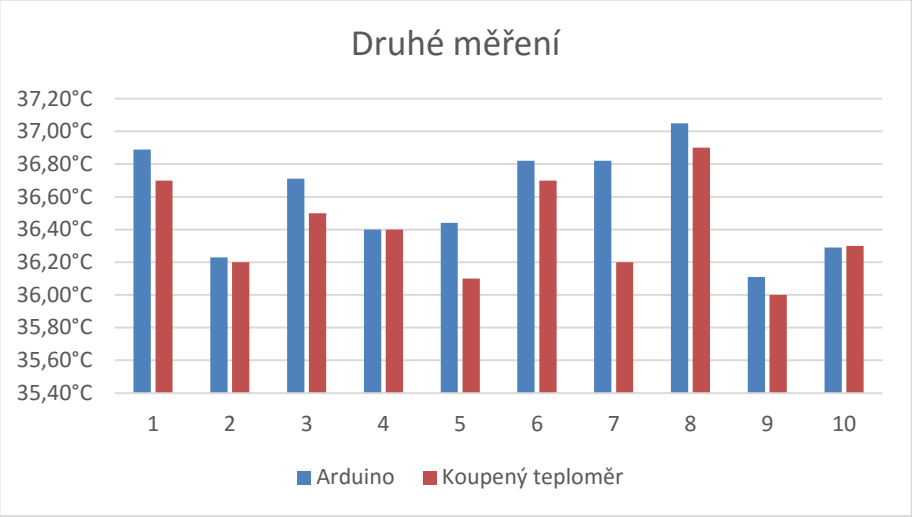
5. Výsledky měření

Jelikož došlo k chybě na obvodu, sestrojený teploměr měřil nesmyslná data, která nebylo možné analyzovat. Při konzultaci s odborníkem byla objevena závada, kterou bylo chybějící napojení operačního zesilovače ke zdroji. Po opravení chyby a výměně odporu začal teploměr fungovat správně. Zde jsou výsledky měření v tabulce i s popisky.



Obrázek 4 – výsledky prvního měření vložené do grafu
zdroj: autor

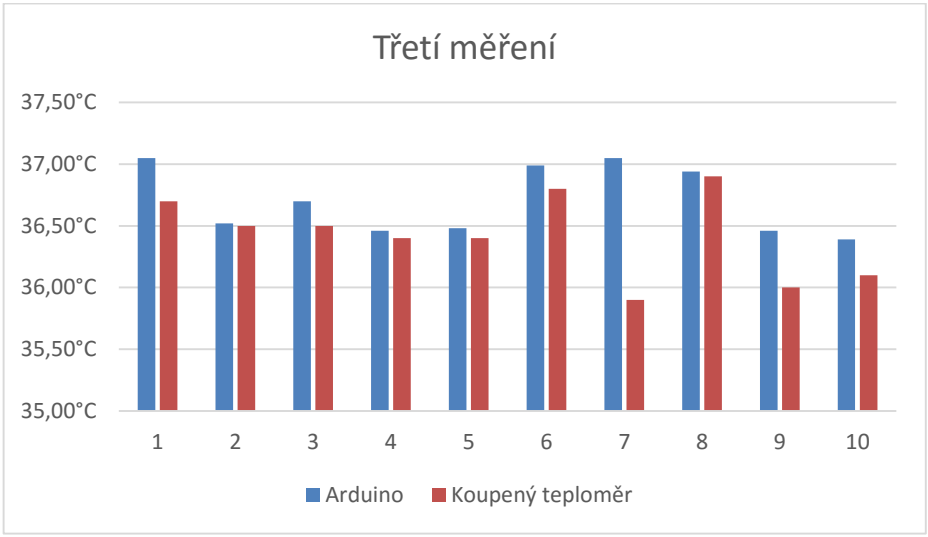
Čísla po levé straně představují teplotu, číslo pod grafem číselné označení měřeného subjektu a barevné označení rozlišuje sestrojený Arduino teploměr a komerčně dostupný zakoupený teploměr.



Obrázek 5 – výsledku druhého měření vložené do grafu

zdroj: autor

Na obrázku 5 jsou výsledky druhého měření stejných subjektů, většinou s podobným výsledkem. Větší rozdíly jsou pravděpodobně způsobeny jiným sevřením teploměru.



Obrázek 6 – výsledky třetího měření vložené do grafu

zdroj: autor

Třetí a poslední měření oběma teploměry na obrázku 6 ukazuje, že teploty jsou většinou podobné předešlým výsledkům u měřených subjektů. Patrné rozdíly jsou pravděpodobně způsobené opět jiným uchytnutím a sevřením v podpaží měřeného.

Jak je na grafech vidět, naměřené teploty nejsou vždy stejné. Na jednu stranu mají oba teploměry odchylku 0.5^{o}C, na druhou stranu je velmi nepravděpodobné, že by oba naměřily zrovna s maximální odchylkou na obou koncích chybnosti. Nepřesnosti jsou tedy pravděpodobně způsobeny různým držením v měřeném podpaží, jelikož lidské tělo nezmění svou teplotu tak rychle bezdůvodně. Místnost, ve které byla teplota měřena, by na výsledek neměla mít dopad, jelikož od ní byl teploměr izolován při zmáčknutí v podpaží měřeného subjektu. I přesto však je měření úspěšné, protože naměřená data nejsou v žádném z případů extrémně odlišná.

6. Závěr

I přes původní selhání se po konzultaci s odborníkem na techniku a Arduino podařilo zprovoznit hardwarovou část experimentu, který tím pádem dopadl úspěšně. Při porovnávání výsledků s hodnotami naměřenými normálně dostupným teploměrem bylo zjištěno, že se výsledky neliší o více než půl stupně, což je poměrně úspěch. Vytyčené cíle byly tedy splněny – teploměr byl sestrojen, naprogramován i otestován úspěšně. I přes četné potíže je výsledek na konci uspokojivý. Propříště, abych se vyhnul chybám, začnu (a doporučuji to všem, kdo budou pracovat na podobných pracích) dříve s testováním a konstrukcí obvodů, nežli psaním. Obecně časová rezerva při stavbě přijde vždy vhod, jelikož existuje spousta věcí, které se mohou pokazit, nebo nemusí fungovat. Při vypracovávání jsem se naučil plynuleji objevovat chyby především v programech díky většímu porozumění chybovým hlášením. V oboru open source elektroniky je mnohem víc, co se dá zkoumat. Teploměr je jen jedním z mnoha lékařských nástrojů, které jsou jednoduše konstruovatelné – další témata bádání mohou být například pulsní oxymetr nebo EKG.

7. Zdroje

[1] Arduino. *Wikipedia: the free encyclopedia* [online]. Wikimedia Foundation [cit. 2015-12-13]. Dostupné z: https://cs.wikipedia.org/wiki/Arduino

[2] Czechduino: Co je to Arduino? [online]. [cit. 2016-05-15]. Dostupné z: http://czechduino.cz/?co-je-to-arduino,29

[3] LM35: datasheet. *Texas Instruments* [online]. [cit. 2016-05-15]. Dostupné z: http://www.ti.com/product/LM35/datasheet

[4] *Processing: Tutorials* [online]. [cit. 2016-05-15]. Dostupné z: https://processing.org/tutorials/

PŘÍLOHY

1. Kód Arduina

```
#define ACD_PIN A0

int acdRead = 0;

void setup() {

  Serial.begin(9600);

}

void loop() {

  acdRead = analogRead(ACD_PIN);

  Serial.println(acdRead);

  delay(50);

}
```

2. Kód Processingu

```
// IMPORT

import processing.serial.*;

// OBJECTS

Serial serial; //načítání

// VARIABLES

int h = 650, w = 1200; //h - výška okna, w - šířka okna - NEovlivňuje skutečnou velikost okna!

//proměnné pro interface (vzdálenosti, vlikosti atd.)

int window = 600, taskbar = 50;

int leftSide = 50; //určuje šířku odstavení na levé straně (pro popisky)

int y1 = h - 10, y2 = h -40; // proměnné pro umístění tlačítek na ose y

int x50 = 50, x25 = 25; //proměnné pro umístění tlačítek na ose x, používat násobky

int z = 10; //zaoblení rohů u tlačítek
```

```
//proměnné pro funkci fil

float min = 100, max = 0; //peak max a min

float val = 0;

float t = 0;

float koeficient = 9.15;

//

int data = 400;

float rozdil = 0.4;

float[] teploty = new float[data];

//proměnné pro nastavení barev

color rgb1 = color(255, 0, 0), rgb2 = color(0, 255, 0), rgb3 = color(0, 0, 255), rgb4 =
color(255, 255, 0); //zatím nepoužito

color akt1 = color(255), akt2 = color(0), akt3 = color(100), akt4 = color(200), akt5 =
color(150), akt6 = color(255, 0, 0); //berevné schéma celého programu

color choose = color (150); //zatím nepoužito

int coloropt = 0; //nepoužito

color grafDone = (0);

//nerozřazené proměnné

int maxU = 50; //volty na stupnici

String frekvence = "";

int frekvence1 = 20;

int usedf = 0; //max počet znaků u frekvence

boolean vos = false; //vykreslení vertikálních os

boolean run = true;

  float prumer = 0;

// ARRAYS
```

```
float[] graf = new float[w-leftSide]; //pole pro vykreslování grafu. uchovává 1175 hodnot - pro červené světlo

void setup() {

  frame.setTitle("Seminární práce - Adam Bohbot 2015/16");

  size(1200, 650); //nastavení velikost okna

  serial = new Serial(this, Serial.list()[0], 9600); //objekt serial - číslo v [] určuje pořadí použitého portu

 serial.bufferUntil('\n'); //blba bla bla

  println(Serial.list()); //vypíše porty

  //setup

  rectMode(CORNERS);

  //naplnění pole

  for (int i = 0; i < graf.length - 1; i++) {

    graf[i] = 0;

  }

}

void draw(){

  if (run){

  background(akt1);

  if (vos) drawVerLines(); //pokud vos == true, vykreslí v. čáry

  drawBcg(); //zavolá funkci, která vykreslí pozadí, rámečky, osy a většinu interface

  drawButtons(); //zavolá funkci, která vykreslí tlačítka, vstupy a texty

  drawGraph();

  //println(grafIR[grafIR.length - 1] + " " + grafR[grafR.length - 1]);

  }

}

void drawBcg() {
```

```
//vykreslení os
//vykresí slabší osy
strokeWeight(0.5);
stroke(akt4);
for( int i = 0; i < maxU; i++){
  line(leftSide, i * (window/maxU), width, i * (window/maxU));
}
stroke(akt3);
for( int i = 0; i < maxU; i+=5){
  line(leftSide, i * (window/maxU), width, i * (window/maxU));
}
//vykreslí hlavní osy
stroke(akt2);
fill(akt2);
strokeWeight(2);
textAlign(CENTER); //centrování textu
for (int i = 0; i < maxU  ; i += 10) {
  line(leftSide, i * (window/maxU), width, i * (window/maxU));
 // text();
}
//zbytek interface (taskbar, stupnice, tad.
strokeWeight(4);
fill(akt1);
noStroke();
rect(0, height - taskbar, width, height);
fill(akt4);
```

```
  rect(0, 0, leftSide, window);
  stroke(0);
  line(0, window, width, window);
  line(leftSide, 0, leftSide, window);
  //popisky os
     fill(akt3);
  for (int i = maxU; i > 0  ; i -= 5) {
    text(i + "°C", 20, window - i * (window/maxU));
  }
  fill(akt2);
  for (int i = maxU; i > 0  ; i -= 10) {
    text(i + "°C", 20, window - i * (window/maxU));
  }
}
void drawButtons(){
  //vykreslí tlačítka atd. a texty
  //SETTINGS
  strokeWeight(2); //nastaví tloušťku
  fill(akt1);
  //INPUTS
  rect( x25, y1, 5 * x25, y2, z); //políčko pro frekvenci
  rect( 8 * x50, y1, 13 * x50, y2, z); //políčko pro frekvenci
  rect( 14 * x50, y1, 19 * x50, y2, z); //políčko pro frekvenci
  //BUTTONS
  //tlačítko konec - gragika
  if (mouseX > 41 * x25 && mouseX < 47 * x25 && mouseY < y1 && mouseY > y2)fill(akt4);
```

```
  else fill(akt5);
  rect(41 * x25, y1, 47 * x25, y2, z); //talčítko konec
  //tlačítko změnit frekvenci
  if (mouseX > 3 * x50 && mouseX <6 * x50 && mouseY < y1 && mouseY > y2)
fill(akt4);//změna barvy po najetí na tlačítko spustit
  else fill(akt5);
  rect(3 * x50, y1, 6 * x50, y2, z);
  //TEXTS
  fill(akt2); //nastavení barvy textů
  textAlign(CENTER, CENTER); //centrování textu
  text("Konec", 41 * x25, y1, 47 * x25, y2);
  text("Změnit f", 3 * x50, y1, 6 * x50, y2); //načíst - popis
  text("Hz", 2 * x50, y1, 5 * x25, y2); //Hz - popis
  text(frekvence, x25, y1, 5 * x25, y2); //zápis do pole frekvence
  String s = nf( teploty[teploty.length - 1],0,2);
  text("Aktuální teplota: "+s + " °C", 8 * x50, y1, 13 * x50 ,y2);
        String s1 = nf((prumer/data), 0, 2);
 // fill(rgb1);
  text("Průměrná teplota: "+s1 + " °C", 14 * x50, y1, 19 * x50 ,y2);
}
void drawGraph(){
  //vykreslí oba grafy
  //vykreslení grafu
  fill(akt2);
  strokeWeight(2); //nastavení tloušťky grafů
  stroke(akt6);//barva pro grafR
```

```
  for (int g = 0; g < graf.length -1; g++) {
    line(g + leftSide, map(graf[g], 0, 50, window, 0), g+1+leftSide, map(graf[g+1], 0, 50, window , 0));
  }
  //překreslení čáry na levém kraji
  strokeWeight(3);
  stroke(akt2);
  line(leftSide, 0, leftSide, window);
}
void drawVerLines(){
  //vykreslí vertikální čáry
    for (int i = 0; i*frekvence1+leftSide<width; i++) {
      stroke(akt4);
      strokeWeight(0.5);
      line(leftSide + (i*frekvence1), 0, leftSide + (i*frekvence1), height - taskbar );
    }
}
void fil(float val) {
  //upraví hodnoty, aby se pohybovaly kolem nuly
  t = val;
  t = t / 1023 * 5 ;
  t = (t / koeficient) / 0.01;
  float teplota = t;
prumer = 0;
  println(" test:" + teplota);
  teploty[teploty.length - 1] = teplota;
```

```
  //posování pole teplot
  for ( int i = 0; i < teploty.length - 1; i++) {
    teploty[i] = teploty[i + 1];
  }
  graf[graf.length - 1] = teplota;
    //posunutí hodnot v poli
  for (int i = 0; i < graf.length-1; i++) {
    graf[i] = graf[i+1];
  }
  for(int i = 0; i < teploty.length-1; i++){
    prumer += teploty[i];
    min = min(min, teploty[i]);
    max = max(max, teploty[i]);
  }
  if(min > (prumer/data) - rozdil && max < (prumer/data) + rozdil){
    println("DONE!!" + prumer/data);
    run = false;
  }
  println( prumer/data + " " + teploty[teploty.length-1]);
  max = 0;
  min = 1000;
}
void keyPressed(){
  //vykoná akci po stisknutí klávesy
  if (key != BACKSPACE  && usedf < 3) { //pokud není zmáčknutý Backspace, tak se zapíše
do proměnné frekvence
```

```
    frekvence+= key;
    usedf++; //obsazení jednoho ze tří slotů pro znaky
  }
  if ( key == BACKSPACE) { //překreslí se pole a vymaže poslední znak při stisku BACKSPACE
    if (frekvence.length() > 0) {
      frekvence = frekvence.substring(0, frekvence.length()-1);
      usedf--;
      redraw();
    }
  }
}
void mouseClicked(){
  //provede akci při zmáčknutí myši
  if (mouseX > 41 * x25 && mouseX < 47 * x25 && mouseY < y1 && mouseY > y2) exit();
  if (mouseX > 3 * x50 && mouseX <6 * x50 && mouseY < y1 && mouseY > y2) { //načíst frekvence
    frekvence1 = int(frekvence); //do proměnné frkvence1 vpíše aktuální vloženou frekvenci
    if (frekvence1 > 0) vos = true; //pokud je vos pravda, vypíší se v. osy
  }
}
void serialEvent(Serial serial) {
  //read value from serial
  String string = serial.readStringUntil('\n');
  if (string != null) {
    string = trim(string); //remove '\n'
    float y = int (string); }}
```