Univerzita Hradec Králové

Přírodovědecká fakulta

Katedra fyziky

Využití platformy Arduino ve výuce

Bakalářská práce

Autor: Andrea Hladíková

Studijní program: BFY-BMAT

Studijní obor: Fyzika a matematika pro vzdělávání

Vedoucí práce: RNDr. Jiří Hubeňák, Ph.D.

Hradec Králové květen 2016

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracovala samostatně s využitím uvedených pramenů a literatury.

.....

Podpis autora práce

Poděkování

Nejprve bych ráda poděkovala vedoucímu bakalářské práce RNDr. Jiřímu Hubeňákovi, Ph.D., za odborné vedení a ochotu ujmout se této práce. Dále také panu RNDr. Danielu Jezberovi a panu Ing. Karolu Radochovi, Ph.D. za poskytnutí pomůcek pro měření.

A v neposlední řadě celé své rodině a partnerovi Vítu Bednářovi za neustálou podporu a trpělivost.

Bibliografický záznam

HLADÍKOVÁ, Andrea. *V*využití platformy Arduino ve výuce. Hradec Králové, 2016, 103 stran. Bakalářská práce. Univerzita Hradec Králové, Přírodovědecká fakulta, Katedra fyziky. Vedoucí práce RNDr. Jiří Hubeňák, Ph.D.

Anotace

Bakalářská práce „Využití platformy Arduino ve výuce“ se zabývá možnostmi, jak propojit moderní oblast IT s výukou fyziky. Při výuce fyziky není možné se omezovat pouze na teoretický výklad, proto je nutné ho doprovázet praktickými ukázkami. Pomocí platformy Arduino je možné propojit fyzikální měření s programováním, stavbou měřících zařízení a vyhodnocováním naměřených dat. První část práce je věnovaná obecným informacím o Arduinu a jeho programování. V druhé části jsou vlastní zadání fyzikálních měření, které lze snadno pomocí platformy realizovat. V přílohách se pak nachází vlastní vypracování zadaných úloh.

Klíčová slova

Arduino, fyzikální měření, programování, zpracování dat, tvorba měřicích modulů

Anotation

The bachelor thesis „Use of Arduino Platform in Education“ deals with possibilities of connecting modern IT with physics education. While teaching physics it is not possible to be limited to the presentation of theoretical knowledge. Such a presentation needs to be accompanied by practical demonstrations. By using Arduino Platform it is possible to connect practical classes of physics with programming, building measure devices and data assessment. The first part of the thesis is focused on general information about Arduino and its programming. The second part contains measurements in the form of assignments that are simple to implement. The appendix contains solutions to these assignments.

Keywords

Arduino, physical measurement, programming, evaluation of values, creation measuring modules

Obsah

Úvod

S rozvojem informačních technologií se pojí i vývoj v oblasti vědeckých měření, kde jsou tyto technologie využívány. V dnešní době lze s vynaložením relativně malých nákladů vytvořit domácí laboratoř s poměrně vysokou přesností měření. Jeden takový model domácí laboratoře na platformě Arduino ukazuje tato práce. Obsahem práce je také několik vzorových měření, které lze využít ve školním prostředí. Umožňuje propojit oblast výuky informačních technologií, fyziky a odborných předmětů, sbírání a analýzu naměřených dat.

Výuku fyziky je nutné doprovázet také praktickými ukázkami, které napomáhají vizualizaci probírané problematiky. Mnozí studenti si tak lépe osvojí učivo a někteří dokonce naleznou zalíbení ve fyzice. Ale ne všechny školy si mohou dovolit mít výborně vybavené fyzikální laboratoře. Z toho důvodu jsem začala rozvíjet nápad, který se zabýval fyzikální laboratoří, která by byla minimálně finančně náročná. Kromě malé finanční náročnosti, je také velkou výhodou skutečnost, že takto sestavenou fyzikální laboratoř si může pořídit každý domů. A může tak docházet k rozvoji v oblasti fyziky nejen ve škole, ale i doma.

Vzhledem širokému obsahu fyziky, není možné v této práci obsáhnout všechny její oblasti. Proto je zaměřena pouze na základní úlohy mechaniky, elektřiny, kmitání a vlnění. U těchto oblastí ukazuje, jak je možné využít platformu Arduino pro jejich měření. K tomu byly sestaveny vzorové úlohy, které mají sloužit jako inspirace pro tvorbu dalších vlastních měření.

Teoretická část

1 Co je Arduino?

Arduino je open-source elektronická platforma, která byla vyvinuta v roce 2005 v Itálii. Tato platforma vznikla pro účely výuky na technických vysokých školách. Postupem času se vyvinuly různé typy. V roce 2010 se na světě již prodalo kolem 120 tisíc kusů. Postupem času se kromě různých typů platfotmy Arduino začaly na trhu objevovat i kopie [27].

Tato platforma se velmi rozšířila mezi studenty technických škol, domácí kutily a další. Platforma má téměř bezmezné využití vzhledem k možnostem programování a připojování periférií, které se neustále vyvíjejí.

Arduino je vystavěno na mikroprocesorech ATMega328 (popřípadě na mikroprocesoru ATMega168). Procesory vytváří firma Atmel, která se zabývá vývojem mikroprocesorů [26].

Programování Ardiuna probíhá ve vývojovém prostředí Arduino IDE, které je volně stažitelné na internetu. Jako programovací jazyk slouží jazyky C, C++. Nejčastěji dochází k programování za pomoci knihovny Wiring, která vychází z jazyka C++.

2 Hardware

Arduino má různé typy desek, podle nich se mění jednotlivé specifikace. V dalších částech práce se zaměříme hlavně na konkrétní typy desky Arduino, které jsou nejčastěji prodávané a v praktické části budou používány. Pro měření byl vybrán typ Arduino UNO, dále je vhodné využít typ Arduino Nano.

2.1 Užívaný Hardware

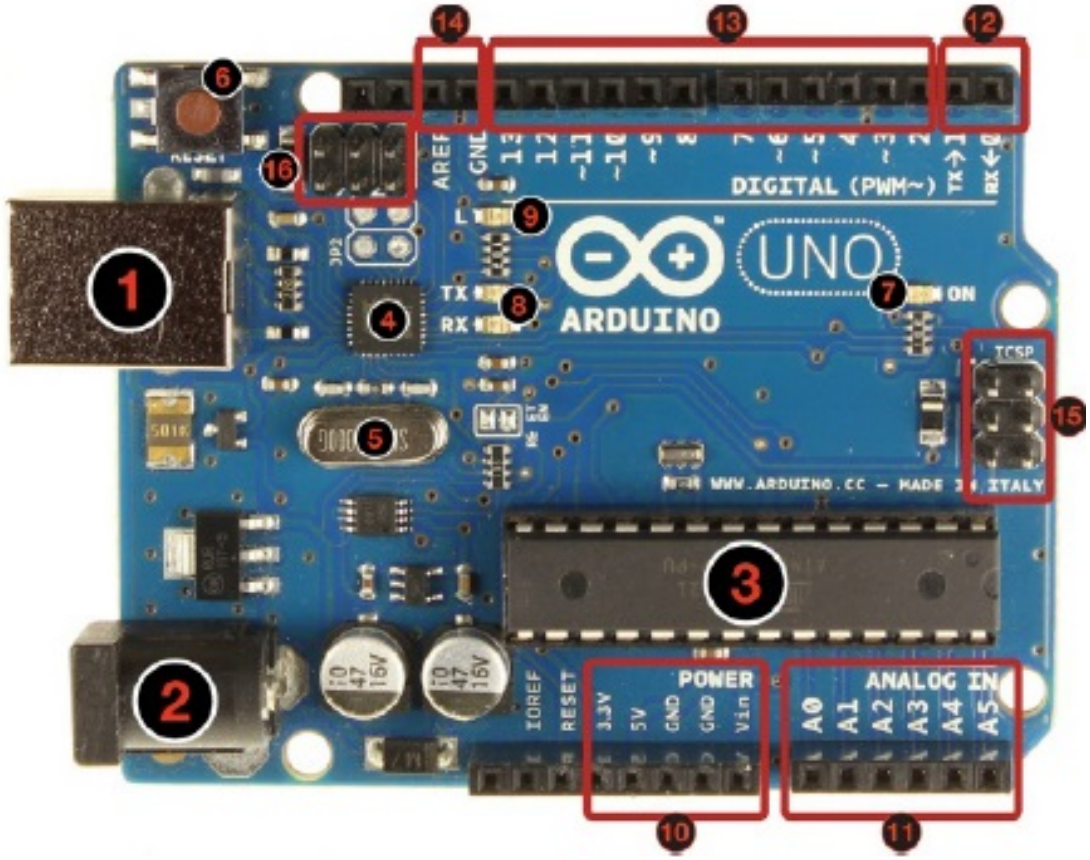
2.1.1 Platforma Arduino

Platforma Arduino UNO je složena z analogových a digitálních výstupů, mikročipu ATMega328, kontrolních LED diod. Arduino obsahuje paměť SRAM (Static Random Access Memory), jedná se o druh polovodičové paměti RAM. SRAM paměť je statická, tedy nepotřebuje pravidelnou obnovu a je vystavěna na bistabilním klopném obvodu, který uchovává informaci o velikosti 1 bitu na 6 tranzistorech. Dále je využívána EEPROM (Electrically Erasable Programmable Read-Only Memory), jedná se o elektronicky mazatelnou paměť. Tato paměť pracuje na principu tunelování elektrického náboje mezi vrstvu oxidu křemíku a nitridu křemíku. Tato paměť slouží hlavně k uchovávání informací, které se často nemusejí přepisovat, na paměť dochází k ukládání firmware. Firmware je jednou z nejnižších forem operačního systému, který slouží pro konkrétní systém (př. kalkulačka). Arduino je také schopné tvořit modulované signály za pomoci PWM [20].

Hardware Arduino Uno je zobrazen na Obrázku 1.

Modul se skládá ze šestnácti částí [31]:

1. USB konektor pro komunikaci se zařízeními
2. SPI konektor
3. Procesor ATMega328, odnímatelný a programovatelný
4. Procesor pro komunikaci
5. Externí hodiny, krystal s frekvencí 16 MHz
6. Tlačítko pro restartování
7. Kontrolní LED-dioda (On)
8. Komunikační diody TX, RX (vysílací a přijímací)
9. LED-dioda připojená na pin 13 (programovatelná)
10. Zdroje napětí, GND (uzemění), 5 V a 3,3 V regulovatelné
11. Analogové vstupní piny



Obrázek 1: Části hardware Arduino UNO [36]

12. TX, RX, digitální pin
13. Digitální vstupně-výstupní piny, piny s PWM regulací, regulace až pro 40 mA
14. Pin pro zem a AREF (Nastavení referenčního napětí)
15. ICPS pro ATMega328
16. ICPS pro USB

2.2 Technické parametry

Arduino v dnešní době nabízí přibližně deset sériově vyráběných typů desek. Pro účely této práce bude používán velmi rozšířený typ Arduino UNO. Také je vhodné Arduino Nano, které je možné umístit přímo do kitu a je vhodné svým propojením výkonu a miniaturnosti. Soupis jednotlivých technických parametrů naleznete v následujících tabulkách. Arduino Nano je vhodnější například pro

konstrukci robotů, vzhledem k malému rozměru, nízké váhy a možností přímého spojení s kitem. Arduino Nano se vyrábí se dvěma druhy čipů, ATMega328 a ATMega168. Dle druhu čipu se liší velikost paměti Arduina. V praktické části je využito Arduino UNO.

Parametry Arduino UNO		
čip	ATMega328	
Frekvence hodin	16 MHz	
Pracovní napětí	5 V	
Vstupní napětí	7-12 V	
Vstupní napětí (limit)	6-20 V	
DC proud	20 mA	
DC proud při 3,3 V	50 mA	
Paměť	32 kB	
SRAM	2 kB	
EEPROM	1 kB	
Vstupně výstupní piny	14x digitální	6x analogový
PWM digitální	6x IO	
Konektory	USB 2,0 SPI	
Rozměry	6,85 x 5,33 cm	
Hmotnost	25 g	

Tabulka 1: Parametry Arduino Uno [23]

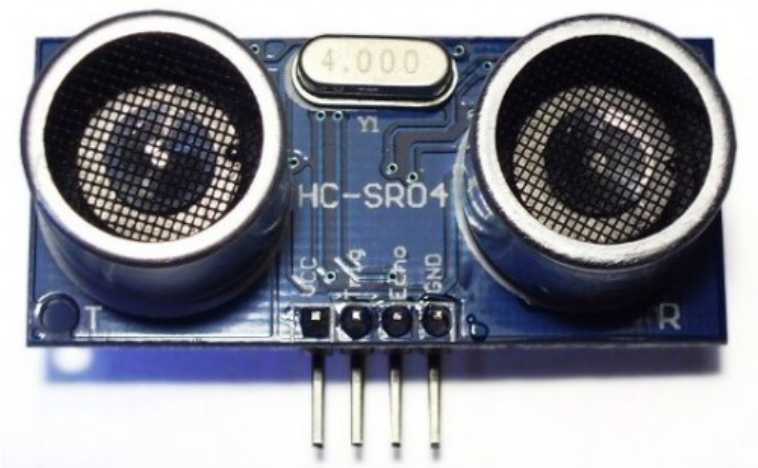
Parametry Arduino NANO		
Čip	ATMega168	ATMega328
Frekvence hodin	16 MHz	
Provozní napětí	5 V	
Vstupní napětí	7-12 V	
Vstupní napětí (limit)	6-20 V	
DC proud	40 mA	
Paměť	16 kB	32 kB
SRAM	1 kB	2 kB
EEPROM	512 B	1kB
Vstupně výstupní piny	14x digitální	8x analogový
PWM digitální	6x IO	
Konektory	microUSB	
Rozměry	4,5 x 1,8 cm	
Hmotnost	5 g	

Tabulka 2: Parametry Arduino Nano [24]

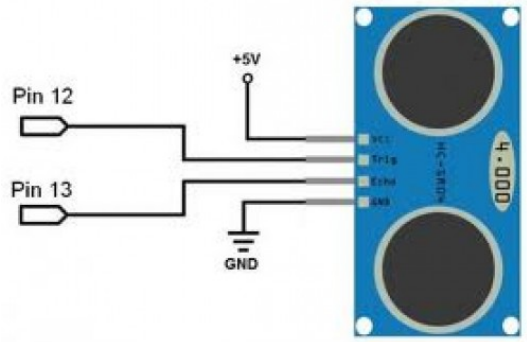
Obě desky se finančně pohybují kolem 500 Kč až 1000 Kč. Je možné sehnat levnější kopie této platformy, například Funduino.

2.2.1 Měřící moduly

Pro platformu Arduino jsou vyvinuty mnohé měřící moduly, které mají široký rozsah užívání (měření teploty, měření vzdálenosti, osvětlení a další). V této práci budou měřící systémy sestavovány, až na jednu výjimku a tou je měřící modul sonar. Sonar funguje na principu vysílání a přijímání ultrazvukových vln. Ze zařízení je vyslán zvukový impuls. Od začátku jeho vyslání je měřen čas do jeho přijetí. Tento modul je velmi přesný dokáže měřit vzdálenost v rozmezí 2 cm až 4 m s přesností několika milimetrů [13].



Obrázek 2: Externí modul sonar HC-SR04 [40]



Obrázek 3: Schéma modulu sonar HC-SR04 [40]

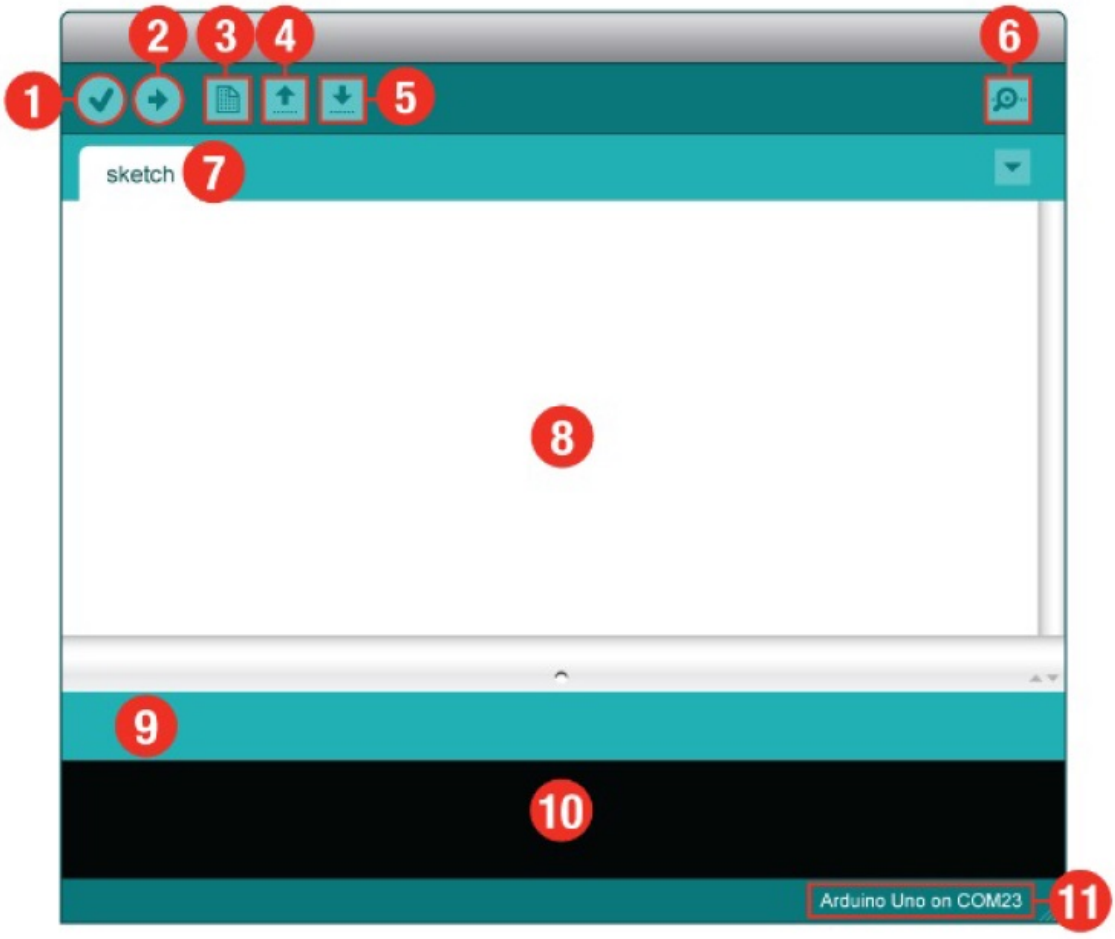
3 Software

Na programování platforem Arduino bylo vyvinuto vývojové prostředí Arduino IDE (integrated development enviroment = integrované vývojové prostředí), které podporuje jazyk Wiring.

3.1 Arduino IDE

Arduino IDE je freewarové vývojové prostředí, je možné ho tedy stáhnout a využívat pro libovolnou platformu. Arduino IDE je vyvíjeno pro různé operační systémy od Windows, MacOS, Linux. Instalace vývojového prostředí je rozdílná dle operačních systémů, ale velmi intuitivní. Obsahem jsou veškeré ovladače a hlavně zdrojové kódy, na kterých se i naprostý začátečník naučí rozumět syntaxi programů a jejich základním funkcím.

Při každém spuštění se otevře základní vývojové prostředí Arduino IDE, které je následující [22]:



Obrázek 4: Vývojové prostředí Arduino IDE [22]

1. **Verify** - ikona pro kontrolu kódu před jeho nahráním na vlastní desku. Při nalezení chyby v syntaxi dojde k jejímu vyhlášení a zvýraznění.
2. **Upload** - spustí kontrolu programu, v případě úspěšné kontroly programu dojde k jeho zkompilování a odeslání na desku.
3. **New** - slouží k tvorbě nového souboru, vytvoří novou záložku v okně.
4. **Open** - slouží k otevření již existujícího souboru.
5. **Save** - tlačítko pro uložení aktuálního projektu.
6. **Serial Monitor** - otevírá okno, ve kterém se nalézají veškeré informace, posílané přes sériové porty.
7. **Sketch Name** - obsahuje jméno daného programu.

8. Vlastní plocha pro psaní programu.
9. Okno chybového hlášení, odkazuje na chyby v syntaxi.
10. Okno pro chyby s kompletním hlášením, vhodné pro ladění funkčnosti programu.
11. Soupis desek zapojených přes sériovou linku (USB).

3.2 Programovací jazyk

3.2.1 Základní struktura

Při programování budeme používat knihovnu Wiring. Při otevření vývojového prostředí Arduino IDE se objeví bílá plocha, kde je možné vpisovat zdrojový kód. Zdrojový kód jako u všech programovacích jazyků má pevně danou strukturu, při jejím nedodržení nelze program zkompilovat a odeslat na desku.

První část programu je funkce **void setup (){}**. **Void setup (){}** je funkce, kterou program udělá pouze jednou. Mezi složené závorky se zapisuje inicializace vstupních a výstupních pinů a dalších funkcí. Tato funkce se provádí při připojení platformy, popřípadě po jejím restartování. Další částí programu je opakující se funkce **void loop (){}**. Do složených závorek v této funkci je zapisován zdrojový kód, který se neustále opakuje, dokud nedojde k odpojení napájení, popřípadě přehrání jiným programem [20].

Program

Základní struktura prázdného programu [20]

```
void setup(){
   //inicializační funkce, probíhá jednou na začátku
   //programu, užívá se na nastavení
}

void loop {
  //opakující se funkce, příkazy obsažené v jejím těle
  //se opakují do nekonečna až do zastavení procesu
}
```

3.3 Proměnné

Jako v každém programovacím jazyku, tak i zde je nutné vytvořit si místo, kam ukládáme získané hodnoty a další podstatné informace. Toto místo nazveme proměnná. Každá proměnná je deklarována pomocí jejího vlastního jména, datovým typem (viz kapitola 3.3.4) a hodnotou. Jako hodnota proměnné může sloužit i název pinu, na kterém se součástky nacházejí. Proměnné můžeme rozdělit do dvou kategorií na globální a lokální. Globální proměnné se deklarují nejprve. Ve struktuře programu se deklarují před vlastní funkcí **void setup ()**, která již globální proměnné může ve vlastním obsahu využívat, stejně tak je možné globální proměnné využívat ve funkci **void loop ()**. V mnohých situacích je vhodné mít proměnnou pouze dočasnou, na užití pouze uvnitř určité funkce. V takovém případě je vhodné deklarovat pouze lokální proměnnou. Lokální proměnné je možné deklarovat jak ve funkci **void setup ()**, tak i ve funkci **void loop()**. Na rozdíl od globálních proměnných nelze lokální proměnné využívat jinde než ve funkci, ve které byly deklarovány [20].

Program

Způsoby inicializace proměnných [20].

```
int x,y; //inicializace globálních proměnných typu
         //integer viz kapitola (3.3.4)
x = 10; //přiřazení hodnoty proměnné x

int pin_dioda = 13; //deklarace globální proměnné,
           //jméno dioda, hodnota digitální pin 13

int pin:napeti = A0; //deklarace analogového pinu A0

void setup (){
y = 1; //přiřazení hodnoty globální proměnné y
int a = 10; //deklarace lokální proměnné a funkce
           //setup a její hodnoty
}

void loop() {
int b;  //lokální proměnná pro funkci loop, její
       //deklarace probíhá stále dokola
}
```

3.3.1 Základní konstanty

Konstanta je druh proměnné, jejíž hodnota se nemění v čase. Konstanty bývají předdefinované v systému. Mezi základní konstanty patří výsledky logických funkcí. Dále to jsou hodnoty pinu. Při práci s Arduinem je nutné užívaným pinům nadefinovat jejich hodnoty a tedy i způsob jejich funkce. Pro deklaraci vstupu a výstupu jsou předdefinované konstanty **INPUT**, **OUTPUT** a **INPUT_PULLUP** [20].

OUTPUT - nastaví pin jako výstupní s maximální hodnotou výstupního proudu 40 mA.

INPUT - nastaví pin na funkci vstupu.

INPUT_PULLUP - pin je nastaven jako výstupní s invertibilní funkcí.

Pokud Arduino užíváme jako zdroj napětí nebo proudu, užívají se další předdefinované konstanty **HIGH** a **LOW**.

HIGH - nastaví výstupní pin jako zdroj napětí o hodnotě 5 V.

LOW - výstupní napětí nastaveno na 0 V.

Konstanty **HIGH** a **LOW** je možné také využít při nastavení pinu jako vstupu. Tyto konstanty jsou využívány, pokud jako zpětná vazba stačí informace, zda je na pin přiváděno napětí nad konkrétní hodnotou.

HIGH - při čtení hodnot z pinu bude vyhlášeno, pokud je na pin přiváděno napětí větší jak 3 V.

LOW - při čtení hodnoty z pinu není dosažena potřebná hodnota pro **HIGH**.

3.3.2 Ovládání digitálních pinů

Jak bylo výše popsáno u jednotlivých platforem, každá deska se skládá kromě vlastního čipu také z pinů, které slouží pro připojení periférií.
Pro ovládání pinů existují předdefinované funkce, dle toho zda slouží jako vstup nebo výstup. Deklarace pinu má následující tvar:
pinMode(cislo_pinu, INPUT/OUTPUT). Parametr **INPUT** a **OUTPUT** jsou popsané výše. Při deklaraci pinu je nutné uvést název pinu. Číslo pinu je napsáno

přímo u pinu na vlastní desce. Pro práci s piny u delších programů je vhodné si nadeklarovat proměnné, které označují, co daný pin dělá. Tím, že Arduinu nadeklarujeme pin jako vstup, popřípadě jako výstup, prozatím neříkáme, jakým způsobem ho budeme dále využívat.

K ovládání výstupního pinu slouží funkce **digitalWrite (číslo_pinu/název, HIGH/LOW)**. Tato funkce nařídí pinu, aby zaujal konkrétní hodnotu. Hodnoty **HIGH** a **LOW** jednoduše říkají, zda daným pinem protéká či neprotéká proud, jak bylo zmíněno výše v podkapitole „Základní konstanty". Pokud pin nastavíme jako vstupní, pak existují obdobné funkce jako pro výstupní pin, v tomto případě **digitalRead (číslo_pinu/název_proměnné)**. Tato funkce je schopna také vracet pouze dvě hodnoty **HIGH** a **LOW**.

Program

Blikání LED diody, která je součástí Arduino UNO a je přímo připojena na jeho digitální pin č.13 [28].

```
 void setup () {
  pinMode(13, OUTPUT);
  // inicializace digitálního pinu na pozici 13
  //jako výstup
}

void loop {
  digitalWrite(13, HIGH);   //nastaví napětí výstupu
                            //na maximum
  delay(1000);              //pozastaví proces (počká
                            //sekundu)
  digitalWrite(13, LOW);    //nastaví napětí výstupu
                            //na minimum
  delay(1000);              //pozastaví proces (počká
                            //sekundu)
}
```

3.3.3 Ovládání analogových pinů

Kromě digitálních pinů se na desce vyskytují také piny analogové. Pro inicializaci analogového pinu opět užijeme příkaz **pinMode()**, kde se pro inicializaci analogového pinu do závorek zapisuje **A číslo pinu**. Kde A je znamení, že se

jedná o analogový pin a následně jeho číslo. Stejně tak jako pro digitální piny, tak i pro analogové piny platí obdobné funkce **analogRead(číslo_pinu/jméno _proměnné)**, popřípadě **analogWrite(jméno_pinu, hodnota)**. Při nastavování analogových pinů, na rozdíl od pinů digitálních, není jedno, který pin použijeme jako vstupní, a který jako výstupní. Jako výstupní analogové piny slouží pouze ty, které mají označení PWM. Pulzní šířková modulace (PWM) slouží k nastavení střední hodnoty napětí na výstupním pinu. Nastavení napětí PWM na Arduinu neprobíhá přímo, ale pomocí přepočtu. Na Arduinu se nastavují hodnoty 0 až 255, kde 0 odpovídá 0 V a 255 odpovídá výstupu 5 V. Tedy pro nastavení 2,5 V se na Arduinu nastaví hodnota 128 [20] [**?**].

Pokud bychom tuto hodnotu chtěli využít PWM výstup, syntaxe bude následující: **analogWrite(A3, 64)** hodnota 64 odpovídá přibližně přibližně 25% z maxima, které je 255. Pak by výstupní napětí na daném pinu mělo hodnotu přibližně 1,25 V.

Funkce **analogRead** také neudává přímou hodnotu napětí, kterou je nutno přepočítat, využívá svoji vlastní stupnici v rozsahu 0 až 1024. Kde opět 0 se dá považovat za hodnotu 0 V hodnota 1023 je maximem tedy 5 V [20].

Program

Sériová komunikace a čtení hodnoty z analogového pinu [20]

```
int pin_napeti=A0;
int napeti;

void setup () {

   pinMode (pin_napeti, INPUT);
    Serial.begin(9600); //Inicializace sériiové komu-
                        //nikace (viz 3.3.10), 9600 bitů
                        //za sekundu
}

void loop () {
   napeti = analogRead(pin_napeti);
   Serial.println(napeti);
        // vypíše číselnou hodnotu proměnné
   delay(1);
        //pozastavení cyklu v milisekundách
}
```

3.3.4 Datové typy

Každý programovací jazyk v sobě zahrnuje datové typy, se kterými je schopen pracovat. Při deklaraci proměnné bylo udáno, že je nutné deklarovat její typ. Knihovna Wiring je schopna pracovat s následujícími datovými typy. Datový typ udává jak uživateli, tak i samotnému programu informace, s jakými daty se pracuje. Datové typy umožňují pracovat s numerickými, logickými hodnotami a znaky [3].

Číselné datové typy

byte - je nejmenší datový typ, jeho velikost je 8 bitů, zvládá uchovat pouze celočíselné hodnoty v rozmezí 0-255.

integer - při programování se používá často zkratka int, jedná se o další pouze celočíselný typ, který může mít velikost 16 nebo 32 bitů, dle druhu procesoru platformy, jeho rozsah je tedy -32 768 až 32 767 pro 16 bitů, popřípadě -2 147 483 648 až 2 147 483 647.

long - je celočíselný datový typ o velikosti 32 bitů, tedy v případě 32 bitového integeru mezi nimi není rozdíl.

float - na rozdíl od ostatních je možné v něm uchovávat reálná čísla (konkrétně čísla s desetinnou tečkou), velikost typu je 32 bitů.

Základní logické typy

boolean - jedná se o logickou hodnotu, která má přípustné pouze dvě hodnoty **true** (pravda) a **false** (nepravda). Je vhodné je využívat tam, kde potřebujeme pouze dvě hodnoty. Druhotně je hodnota false rovna 0 a hodnota true libovolnému nenulovému číslu.

Základní znakové typy

char - tento znakový typ v sobě uchovává pouze jeden znak textu. Znak je převeden do číselné hodnoty, dle ASCII tabulky, znak se uvádí v apostrofech.

string - uchovává řetězec znaků, je nutné ho uvést v uvozovkách.

Program

Inicializace proměnné konkrétního typu a funkce v programu, program na zaznamenání napětí v průběhu času

```
int pin_napeti = A0 //deklarace užívaného pinu
int napeti;         //deklarace obecné proměnné
long t;  //deklarace proměnné typu long integer

void setup() {
  pinMode(pin_napeti, INPUT);
                     //deklarace vstupního pinu
  Serial.begin(9600);
  }

void loop() {
  napeti=analogRead(pin_napeti);
                     //uložení hodnoty do proměnné
  t=millis();   //uložení hodnoty času do proměnné
  Serial.print(napeti); //vypsání hodnoty proměnné
  Serial.print(' ');    //vypsání mezery
  Serial.print(t);    //vypsání hodnoty proměnné t
  Serial.println();     //zalomení řádku

  delay(100);
}
```

3.3.5 Pole

Proměnné jsou schopné uchovat data, ale je velmi nepraktické s nimi pracovat, pokud potřebujeme uchovat větší množství dat, která mohou být i provázaná. K tomu je vhodné využívat pole. Pole můžeme považovat za speciální druh proměnné, která v sobě uchová mnoho hodnot. Při deklaraci pole je nutné určit, jaký typ hodnot do něj bude ukládán. Není možné ukládat současně různé datové typy do jednoho pole [**?**].

Deklarace pole je následující: **datový_typ_jméno[rozměr_pole]= prvky_pole**. Při deklaraci ale nemusíme rovnou všechny hodnoty znát, můžeme tedy pole před deklarovat a následně doplnit prvky, či rozměr. V případě práce s polem je nutné, moci se dostat k jednotlivým hodnotám. K hodnotám v poli se lze dostat pomocí příkazu: **jmeno_pole[pozice_čtené_hodnoty]**. Důležité je dát pozor že index prvního prvku je 0 [20].

Program

Základní inicializace pole

```
int [] čísla =  new int [10] //inicializuje pole
                             //čísla pro 10 integerů

int [] fibonacci = {1,1,2,3,5,8,13,21}
//inicializace pole pomoci výčtu jeho prvků

fibonacci[6] = 8 //nastaví prvek na 7.místě v poli
                 //fibonacci na hodnotu 8
                 //dostaneme pole {1,1,2,3,5,8,8,21}

x = fibonacci[3] //do proměnné x načte hodnotu
                 //ze třetí pozice pole
                 //tedy zde x = 2
```

3.3.6 Operátory

Při práci programu je často nutné, aby procesy probíhaly jen za určitých podmínek. Pro porovnávání a další operace používáme tzv. operátory. Operátory pracují s hodnotami, které lze porovnat. Arduino je schopné pracovat se základními

matematickými operacemi, tj. porovnávání rovností a nerovností daných číselných hodnot. Mezi základní matematické operace patří rovnost prvků **A == B**, nerovnost prvků **A != B** a porovnávací relace, které lze udělat jak v ostrém, tak neostrém tvaru **A <= B**. Také je možné využívat logické funkce, pro konjunkci prvků **A && B**, disjunkci **A || B** a v neposlední řade negaci **!A** . Logické a matematické operace je možné kombinovat, vždy ale záleží na druhu užívaných hodnot [20].

3.3.7 Základní vnitřní funkce

Jako základní funkce zde budou uvedeny základní matematické, časové a další funkce, které může využívat Arduino při svém běhu.

Časové funkce

Při měřeních je vhodné používat funkce pro práci s časem. Pomocí nich je možné nastavit vzorkování po určitých časových prodlevách, jinak by snímání probíhalo dle rychlosti procesoru, tedy frekvencí 16 MHz [12].

Pro zastavení většiny procesů, které jsou přímo závislé na procesoru, se užívá funkce **delay(parametr)**. Parametrem funkce je čas v milisekundách. Rozsah se udává celým číslem v rozmezí od 0 do 4 294 967 295. V některých případech je doba čekání v milisekundách příliš dlouhá, a proto je nutné se uchýlit ke kratším časovým intervalům a to microsekundám. V takovém případě je nutné využít funkci **delayMicroseconds(parametr)**, kde je opět parametrem doba trvání, jejíž rozsah je 0 až 65 535.

V mnohých případech je nutné počkat, ale není vhodné aby došlo k zastavení celého běhu. V takovém případě se využívá znalosti vnitřního času Arduina, který lze zjistit za pomoci funkce **millis()**. Následně, za využití dané funkce a vhodného cyklu lze provést funkci, která bude fungovat obdobě jako funkce **delay ()**, ale nebude zastavovat běh celé platformy. Vlastní čas Arduina je v rozsahu 0 až 4 294 967 295 v případě, kdy se Arduino dostane do maximální hodnoty daného času, pak dochází k jeho nastavení opět na 0. Obdobně existuje i funkce **micros()**, je ale nutné poznamenat, že vzhledem k frekvenci procesoru, která je 16 MHz, je nejkratší možná vzorkovací frekvence 4 μs a u 8 MHz 8 μs [20].

Matematické funkce

Platforma Arduino může vykonávat různé matematické operace, mezi základní patří sčítání, odčítání, násobení, dělení apod. Mimo jiné je možné používat matematický operátor modulo, který vrací zbytek po celočíselném dělení. Operátor modulo voláme za pomoci znaku pro procenta. Kromě základních matematických operátorů lze také užívat následující matematické funkce [15]:

min(hodnota1:real,hodnota2:real)

Funkce vybere minimum ze zadaných hodnot.

max(hodnota1:real,hodnota2:real)

Funguje obdobně jako funkce **min()**, jen slouží k nalezení největší z hodnot.

abs(hodnota:real)

Funkce pro výpočet absolutní hodnoty z daného prvku. Vstupem funkce je číslo, pro které bude vypočtena jeho absolutní hodnota (vzdálenost od nulové hodnoty na číselné ose).

constrain(hodnota, dolni_mez, horni_mez)

Jedná se o funkci, která vymezuje rozsah hodnot. Pokud je hodnota v daném rozmezí, nedojde k její změně a je vypsána. V případě, že je některá z mezí překročena, pak dojde k vypsání překročené meze, tedy maxima nebo minima.

map(hodnota, min_puvodni, max_puvodni, min_nove, max_nove)

Tato funkce slouží ke změně rozsahu hodnot. Na rozdíl do předešlé funkce nedochází k ořezu hodnot, ale jejich úpravě za pomoci vhodné matematické operace.

pow(základ, mocnina)

Funkce počítající mocninu základu.

sqrt(hodnota:float)

Funkce vypočítá druhou odmocninu zadané hodnoty.

Goniometrické funkce

Platforma Arduino při práci s goniometrickými funkcemi a obecně
při práci s úhly pracuje s velikostmi, které jsou udané v obloukové míře, tedy tak zvaných radiánech. Veškeré goniometrické funkce mají za vstupní parametr pouze hodnotu úhlu v radiánech a výstupem jsou hodnoty

v rozsahu jejich oboru hodnot pro daný úhel. Je možné užívat následující goniometrické funkce: **sin(hodnota:real)**, **cos(hodnota:real)**, **tan(hodnota:real)**.

Náhodné hodnoty

Pro generování pseudonáhodných hodnot slouží funkce **random()**. Tato funkce může mít jeden až dva parametry. Pokud má funkce random jeden parametr, pak tato hodnota je rovna maximu náhodných hodnot. Funkce random ale této hodnoty nikdy nedosáhne, generovaná čísla jsou o jednotku menší (parametr-1). Pokud bychom deklarovali dva parametry, pak deklarujeme hodnotu minima a maxima pro dané pseudonáhodné hodnoty. Aby bylo možné tyto pseudonáhodné hodnoty vytvářet, je nutné nejprve Arduinu říci, že bude náhodné hodnoty používat, tedy je inicializovat. K tomu slouží funkce **randomSeed()**, která se umisťuje do funkce **void setup()**. Jako parametr funkce **randomSeed()** slouží hodnota analogového pinu, získaná pomocí funkce **analogRead(jméno_pinu)**, na který není nic připojeno. Na tomto pinu dochází k zachytávání elektromagnetického šumu, který slouží jako náhodná vstupní hodnota [20].

3.3.8 Podmínky

Každý zdrojový kód má svoji strukturu, jejíž základ byl popsán v kapitole 3.2.1. Ve funkcích **void setup** a **void loop**, je možné používat podmínky, které říkají, kdy a k jaké činnosti dojde [3].

Základní podmínky jsou:

if () Je základní podmínka, která říká, kdy budou dané příkazy probíhat. Příkazy, které jsou jejím obsahem, proběhnou jen pokud je základní podmínka splněna. V praxi to syntakticky vypadá: **if(podmínka) příkazy**. Příkazy budou tedy splněny, pokud bude splněna daná podmínka. Pokud podmínka nebude splněna, neproběhne ani jeden z příkazů a program následně pokračuje za daným blokem.

else if () Přidává se, pokud chceme napsat další podmínku, popřípadě podmínku s více argumenty, které spojujeme logickými operátory **if()**.

else Else vyjadřuje prakticky podmínku jinak. Else nemá žádné vnitřní podmínky. Ve složených závorkách jsou napsané příkazy, které se provedou, pokud neplatí podmínka příkazu **if()**, popřípadě **else if()**.

Switch Tato podmínka testuje hodnotu proměnné. A dle její hodnoty jsou vykonány určité příkazy. Syntakticky se zadává následovně:
Swich (název_proměnné) case1: příkazy break; case2: příkazy break; ...

Program

Základní struktura užití podmínky if

```
if(i < 10){

Serial.print ('Nízká hodnota napětí')
     //pokud bude hodnota proměnné i menší jak 10
     //pak po seriové lince bude vypsáno dané hlášení
}
else{
Serial.print(i); //pokud nebude i<10 vypíše po sériové
                 //lince hodnotu i
}
```

Program

Základní struktura podmínky Switch, převzatý kód [29]

```
int i;

switch (i) {
  case 1:
   /* Blok 1 */
    break;
  case 2:
    /* Blok 2 */
    break;
  case 3:
    /* Blok 3 */
    break;
  default:
    /* Blok default */
    break;
}
```

3.3.9 Cykly

Podmínky, které byly uvedeny vždy proběhnou alespoň jednou. Na rozdíl od nich cykly nemusejí proběhnout vůbec, mohou proběhnout jednou, ale také několikrát [17].

while(podmínka) Nejprve se testuje podmínka a po jejím splnění jsou vykonány příkazy, které jsou uvedeny v těle cyklu. Ve chvíli, kdy podmínka splněna nebude, dojde k přeskočení bloku programu, který je obsažen v těle cyklu.

do příkazy while(podmínky)

Při zadání cyklu do ... while dojde k vyhodnocení podmínky až na samotném konci. Tedy příkazy proběhnou minimálně jednou.

for(proměnná s počáteční hodnotou; podmínka; změna hodnoty) příkazy

Při zadávání cyklu **for()** je předem dáno, kolik opakování bude provedeno. Proto tento cyklus mívá vlastní proměnnou, která hlídá, kolik opakování již proběhlo.

3.3.10 Sériová komunikace

Při užívání Arduina nastávají situace, kdy je nutné, aby byly hodnoty odesílány do počítače popřípadě do dalších zařízení. Arduino může komunikovat dle svého druhu za pomoci integrované wi-fi či bluetooth. V případě Arduino UNO a Nano připadá v úvahu hlavně sériový port. Aby docházelo k sériové komunikaci, je nutné nejprve inicializovat sériovou komunikaci a to pomocí funkce:

Serial.begin(parametr)

Funkce **Serial.begin(parametr)** má jako parametr rychlost přenosu.
Rychlost přenosu nelze nastavit libovolně, rychlost se udává následujícími hodnotami: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200. Rychlost komunikace udává, kolik přenosů proběhne v rámci jedné sekundy. Rychlost komunikace kromě toho, že udává, kolik přenosů proběhne, je charakteristickou pro způsob přenosu. Nejčastěji užívaná hodnota je 9600 [20]. Funkce **Serial.begin()** se deklaruje v části programu **void setup()**, v některých případech je možné deklarovat více sériových komunikací, v takovém případě se jednotlivé

sériové komunikace číslují, například **Serial2.begin(parametr)**.

Pro odesílání hodnot je možné využít funkce **Serial.print()** a **Serial.println()**. Obě funkce slouží k přenesení dat a jejich vypsání v připojeném zařízení, hlavně počítači. Funkce **Serial.print()** odesílá jednotlivá data, která jsou zapisována za sebou do jednoho řádku. Funkce **Serial.println()** také odesílá data pomocí sériové komunikace, ale na rozdíl od předchozí funkce, spolu s daty je zasílán i znak pro zalomení řádku (konkrétně #10 a #13), tedy hodnoty se vypisují pod sebe. Informace funkce odesílá ve tvaru číselných hodnot dle ASCII tabulky. Je možné zadat způsob, jakým mají být hodnoty vypisovány, tj. číselné soustavy, počet desetinných míst, popřípadě hodnota z ASCII tabulky viz [32].

Komunikace ale nemusí probíhat pouze směrem z Arduina do počítače, je možné zasílat také data z počítače do Arduina. Pokud do Arduina zasíláme informace, dochází nejprve k jejich nahrání do zásobníku, který zvládne uchovat až 64 bitů. Jelikož je zásobník omezené velikosti, je nutné kontrolovat si velikost volného místa, aby nedošlo k přeplnění paměti, k tomu slouží funkce **Serial.available()**. Po nahrání všech dat, která byla odeslána, dochází k jejich zpracování v pořadí, v jakém byla nahrávána. Tak jak jsou nahrávána data, tak dochází k uvolňování jednotlivých bytů. Pro přenášení dat se užívá funkce **Serial.read()**. Tato funkce vezme první byt nahrané informace ze zásobníku a přečte ji, takto pokračuje přes celý zásobník. Zároveň dochází k uvolňování jednotlivých bytů.

Sériová komunikace nemusí probíhat neustále, ale je možné ji v určitou chvíli zastavit. K zastavení sériové komunikace slouží funkce **Serial.end()**, tato funkce nepotřebuje žádný parametr [20].

Program

Užití sériové komunikace a vypisování na sériový monitor

```
int i; // inicializace  globální proměnné i

void setup() {
  i=0;
  Serial.begin(9600); // inicializace sériové komunikace
}
```

```
void loop() {
  Serial.println(i); //vypíše hodnotu proměnné i
                     //a odskočí na další řádek
  i++ ;              //přičte k hodnotě proměnné číslo 1
  if (i>=100000) i=0;//podmínka, při dosažení popřípadě
        //překročení hodnoty 100000 nastaví i opět na 0
  delay(500); // proces je pozastaven na 500ms
}
```

3.3.11 Vlastní funkce

Prozatím zde byly rozebírány jen cykly a podmínky, které jsou předdefinované v jazyku Wiring. Složitější programy je vhodné dělit na menší celky, které se v programu opakují. Takové celky nazveme vlastními funkcemi, které sami definujeme a následně voláme. Deklarace vlastní funkce je nutno provést mimo těla funkcí **void setup()** a **void loop()**. Při deklaraci je nutné uvést její typ, název,případě potřeby parametry, se kterými bude funkce pracovat. Následně příkazy. Syntaxe vlastní funkce je: **void název_funkce (parametry){příkazy}**.

K tomu aby příkazy v dané funkci proběhly, je nutné funkci v programu zavolat. Volání funkce dochází v základních funkcích **void setup()** a **void loop()**. Funkce je následně volána svým jménem a popřípadě parametry, které používá [20].

Praktická část

Teoretická část obsahuje informace, které jsou potřeba k programování platformy Arduino. Tuto platformu je možné využít například jako měřící modul, který je připojený k počítači ve fyzikální laboratoři. A k tomu celá tato práce směřuje, jak využít platformu Arduino ve školní fyzikální laboratoři.

Pro naučení práce s platformou Arduino byla sestavena zadání fyzikálních měření. Která se nacházejí v následující praktické části práce. Pro seznámení s měřením za pomoci platformy Arduino byly vybrány tři oblasti středoškolské fyziky, na kterých se lze naučit sestavovat jednoduché programy, měřící zařízení a následně vyhodnocovat experimentálně naměřená data.

Praktická část je složena z jednotlivých samostatných zadání protokolů, které si student může vytisknout a dle nich postupovat při měření. Zadání mají sloužit pouze jako námět na měření. Studenti, kteří by si nevěděli rady mohou využít přímo navrhovaná řešení.

Autorská řešení úloh jsou uvedena v příloze, ve formě vypracovaného protokolu k měření.

4 Mechanika

Mechanika je oborem fyziky, který se zabývá pohybem těles a jeho příčinami. Tomu se věnují i jednotlivá zadání úloh. Praktická část obsahuje tři základní zadání pro měření.

4.1 Měření rychlosti

Úloha 1: Stanovte rychlost pohybujícího se vozíku.

Pomůcky

Arduino, počítač, autíčko, sonar, měřítko, stínítko

Teorie

Hmotný bod se neustále pohybuje vůči určité vztažné soustavě. Jeho pohyb se dá popsat křivkou, která se nazývá trajektorie a propojuje veškeré body, kterými hmotný bod za určitý časový úsek prošel. Její tvar je závislý na volbě vztažné soustavy. Podle jejího tvaru rozlišujeme pohyby přímočaré a křivočaré.

Dráha je délka trajektorie.

Rychlost je dána jako změna polohy hmotného bodu za jednotku času.

Můžeme rozlišovat rychlost okamžitou a průměrnou. Okamžitá rychlost je dána následovně [2]

$$\overrightarrow{v} = \lim_{\Delta t \to 0} \frac{\overrightarrow{r}(t + \Delta t) - \overrightarrow{r}(t)}{\Delta t} \tag{1}$$

kde $\overrightarrow{r}$ je polohový vektor daného hmotného bodu v určitém časovém okamžiku. Je tedy zjevné, že okamžitá rychlost je vektorová veličina. To lze přepsat do diferenciálního tvaru následovně:

$$\overrightarrow{v} = \frac{d\overrightarrow{r}}{dt} \tag{2}$$

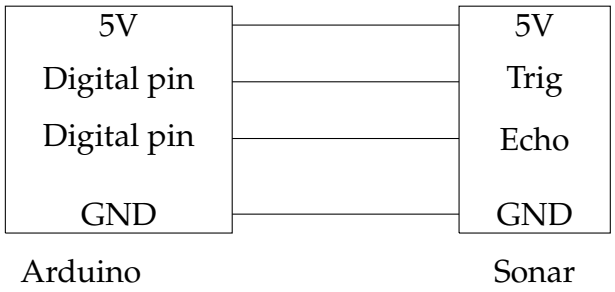
Dále je možné zavést průměrnou rychlost, která udává jakou vzdálenost by hmotný bod urazil za určitý časový interval, pokud by se pohyboval konstantní rychlostí v. Průměrnou rychlost vypočítáme následovně [11]

$$v = \frac{s}{t} \tag{3}$$

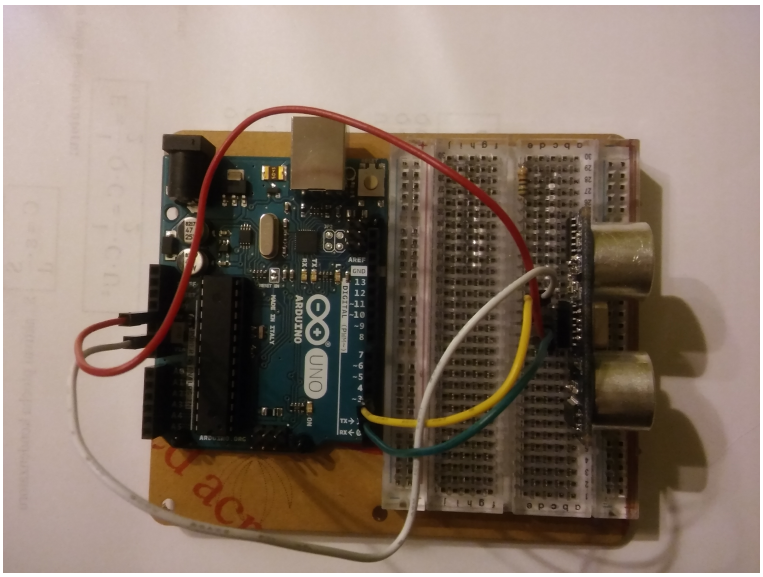
kde s je dráha, kterou urazil hmotný bod za čas t.

Úkoly:

1. Zjistěte rychlost pohybujícího se vozíku.



Obrázek 5: Schéma zapojení modulu sonar k platformě Arduino



Obrázek 6: Zapojení měřícího modulu sonar

2. Zakreslete grafy vývoje vzdálenosti na čase s = f(t) pro jednotlivá měření a proveďte diskuzi sledovaného jevu.

Postup měření

1. K Arduinu pomocí kitu připojte modul sonar. Jednotlivé díly sestavte dle následujícího schématu:

Ve vývojovém prostředí vytvořte zdrojový kód pro Arduino na snímání polohy a času. (V případě větší programátorské zdatnosti můžete vytvořit program, který bude rovnou vyhodnocovat hodnotu okamžité rychlosti a zrychlení.) Modul sonar zapojte do digitálních pinů. Ten jako své jediné hodnoty vrací informace, zda přichází nebo nepřichází signál. Zkompilovaný program nahrajte na platformu a otestujte pomocí stínítka a měřítka. Hodnoty, které ukazuje Arduino,

nejsou vlastní hodnoty vzdálenosti, ale hodnoty času, jaký časový interval uběhl mezi vysláním a zaznamenáním vráceného pulzu. Proto je nutné přepočítat získané časové hodnoty na hodnoty vzdálenosti. Tabulková hodnota rychlosti zvuku je $346,3$ ms^{-1} [25]. Na základě znalosti rychlosti zvuku určete přepočet na vzdálenost.

Po nahrání upraveného zdrojového kódu do Arduina proměřte vývoj vzdálenost daného vozíku. Získané hodnoty zpracujte.

t (ms)	t (s)	s (cm)	v (cm/s)	v (m/s)	a (m/s^2)

Tabulka 3: Vzorová tabulka pro vyplnění naměřených hodnot

2. Naměřené hodnoty zpracujte a vyneste grafy k jednotlivým měřením a okomentujte sledované jevy a chyby, ke kterým docházelo.

4.2 2. Newtonův pohybový zákon a nakloněná rovina

Úloha 1: Druhý Newtonův pohybový zákon - Zákon síly.

Pomůcky

Arduino, sonar, počítač, kladka, spojovací materiál, závaží různých hmotností, vozíček, váhy

Teorie

Druhý Newtonův pohybový zákon říká:

Změna pohybu je úměrná hybné vtištěné síle a nastává podél přímky, v níž síla působí [16].

Druhý pohybový zákon hovoří o změně pohybu. Proto je nutné vycházet z hybnosti, které má těleso. Hybnost hmotného bodu je dána:

$$\overrightarrow{p} = m\overrightarrow{v} \tag{4}$$

Jelikož je změna pohybu úměrná vtištěné síle, je možné zákon síly přepsat do tvaru:

$$\frac{d\overrightarrow{p}}{dt} = \frac{d(m\overrightarrow{p})}{dt} = \overrightarrow{F} \tag{5}$$

Pro malé rychlosti lze předpokládat, že nedochází ke změně hmotnosti, tedy hmotnost tělesa můžeme považovat za konstantní. A časově proměnná je pouze rychlost. Tím dostáváme:

$$m\frac{d\overrightarrow{v}}{dt} = m\overrightarrow{a} = \overrightarrow{F} \tag{6}$$

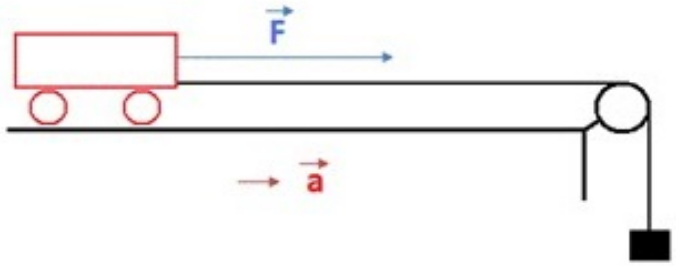
V případě obecného řešení by bylo nutné znát počáteční podmínky pohybu hmotného bodu. [16]

Úkoly

1. Ověřte platnost druhého Newtonova pohybového zákona, změřte závislost dráhy na čase pro jednotlivé hmotnosti vozíku a závaží

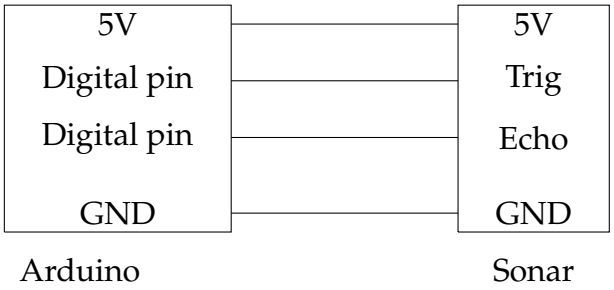
2. Vyneste grafy závislosti vzdálenosti na čase s = f(t) pro jednotlivá měření

Postup měření

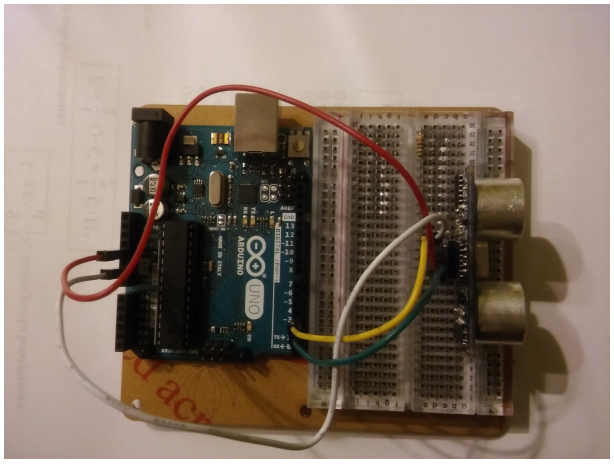


Obrázek 7: Nákres sestavy pro měření 2. Newtonova pohybového zákona [41]

1. Za pomoci platformy Arduino a externího hardwarového sonaru sestavte měřicí zařízení podle následujícího schématu. (S obdobným měřením jste se setkali v první seznamovací úloze pro měření rychlosti.)



Obrázek 8: Schéma zapojení modulu sonar k Arduinu



Obrázek 9: Zapojení měřícího modulu sonar

Vozíček za pomoci dostatečně dlouhého spojovacího vlákna, spojte se závažím. Provázek veďte přes kladku. Sonar umístěte do počátku a vozíček do jeho blízkosti. Vozíček držte, dokud nedojde ke spuštění měření. Jednotlivá data uložte do souborů. Pro každou kombinaci závaží (tj. změnu hmotnosti autíčka a závaží) proměřte alespoň třikrát.

t (ms)	t (s)	s (cm)	v (cm/s)	v (m/s)	a (m/s^2)

Tabulka 4: Vzorová tabulka pro vyplnění naměřených hodnot

2. Naměřené hodnoty zaneste do grafu a stanovte hodnotu zrychlení. Pro ověření správnosti měření zvažte jednotlivá použitá závaží a vozíček. Ze znalosti hmotností a tabulkové hodnoty tíhového zrychlení teoreticky určete hodnotu zrychlení a porovnejte s experimentálně naměřenou hodnotou. Porovnejte získané výsledky.

Úloha 2: Nakloněná rovina

Pomůcky

Arduino, počítač, sonar, kladka, deska, spojovací materiál, materiál na podložení, závaží o různých hmotnostech, vozíček, váhy

Teorie

Druhý Newtonův pohybový zákon říká:

Změna pohybu je úměrná hybné vtištěné síle a nastává podél přímky v níž síla působí [16].

Druhý pohybový zákon hovoří o změně pohybu. Proto je nutné vycházet z hybnosti, kterou má těleso. Hybnost hmotného bodu je dána:

$$\overrightarrow{p} = m\overrightarrow{v} \tag{7}$$

Jelikož je změna pohybu úměrná vtištěné síle, je možné zákon síly přepsat do tvaru:

$$\frac{d\overrightarrow{p}}{dt} = \frac{d(m\overrightarrow{p})}{dt} = \overrightarrow{F} \tag{8}$$

Pro malé rychlosti lze předpokládat, že nedochází ke změně hmotnosti, tedy hmotnost tělesa můžeme považovat za konstantní. A časově proměnná je pouze rychlost. Tím dostáváme:

$$m\frac{d\overrightarrow{v}}{dt} = m\overrightarrow{a} = \overrightarrow{F} \tag{9}$$

V případě obecného řešení by bylo nutné znát počáteční podmínky pohybu hmotného bodu.

Předešlé vztahy ale platí ve chvíli, kde se na pohybovém stavu primárně podílí pouze jedna síla. V případě kladky už dochází k superpozici sil, které se podílejí na výsledném pohybu daného tělesa. Z toho důvodu je nutné upravit rovnici (9) pro nakloněnou rovinu. V případě nakloněné roviny se na pohybovém stavu tělesa také podílí tíhová síla, která stahuje těleso zpět [16].

Úkoly

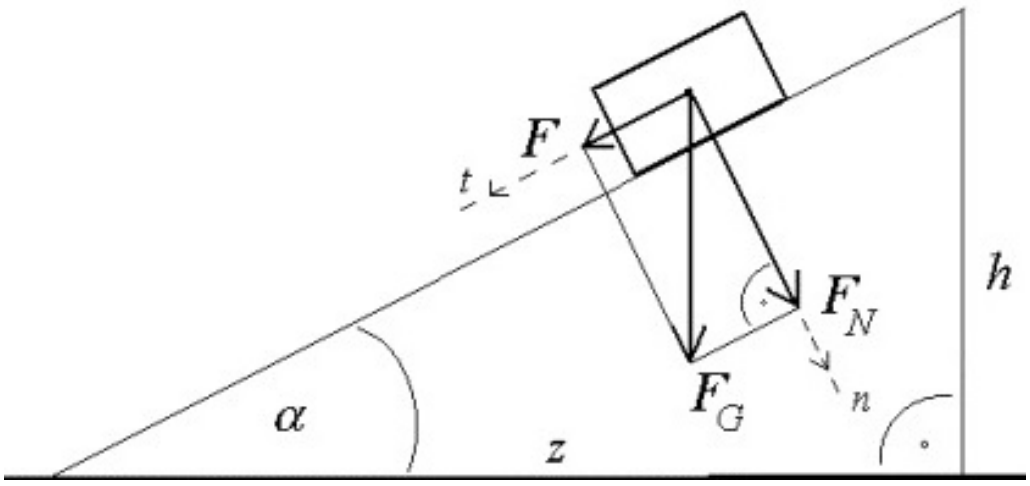
1. Za domácí přípravu udělejte rozbor působících sil a za jeho pomoci určete rovnici pro druhý Newtonův pohybový zákon na nakloněné rovině (upravte rovnici (9)).

2. Ověřte platnost vámi stanovené rovnice, a tak i druhý Newtonův pohybový zákon pro nakloněnou rovinu.

3. Vyneste grafy závislosti dráhy na čase pro jednotlivá měření.

Postup měření

1. Pomocný obrázek pro odvození 2. Newtonova pohybového zákonu pro nakloněnou rovinu:

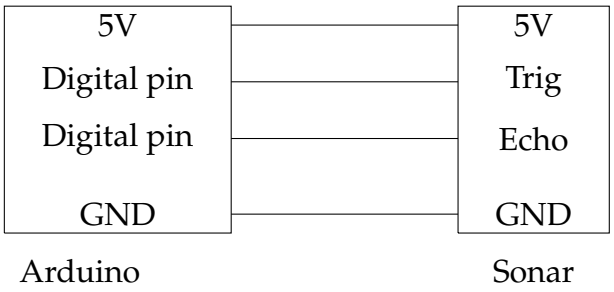


Obrázek 10: Znázornění sil na nakloněné rovině [37]

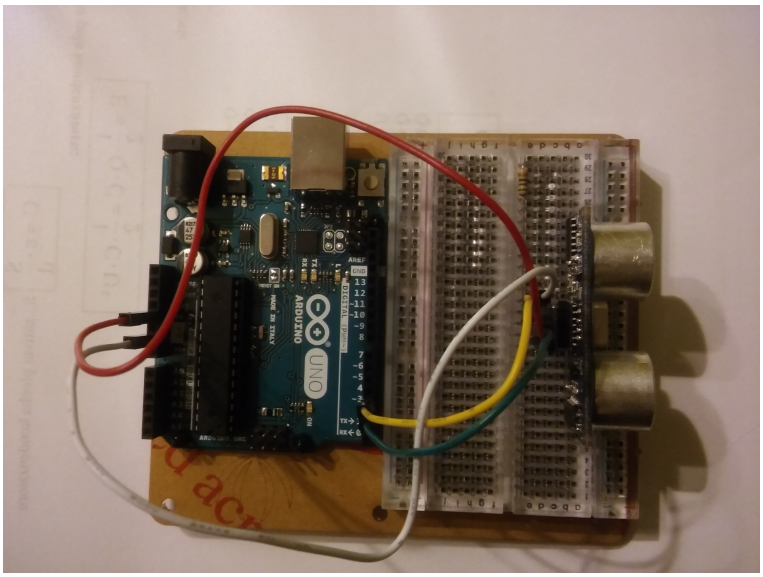
Na obrázku 10 jsou zakresleny působící síly na nakloněné rovině. F_G je tíhová síla, F_N je síla normálová. Superpozicí sil působících vzniká síla F, která působí zrychlení tělesa. α je úhel, který svírá nakloněná rovina s podložkou. h je výška nakloněné roviny.

2. Za pomoci platformy Arduino a externího hardwarového sonaru sestavte měřící zařízení podle následujícího schématu.

   Vozíček za pomoci dostatečně dlouhého vlákna spojte se závažím. Provázek veďte přes kladku. Sonar umístěte na patu nakloněné roviny (na opačnou



Obrázek 11: Schéma zapojení modulu sonar k Arduinu



Obrázek 12: Zapojení měřícího modulu sonar

stranu, než je kladka) a vozíček do jeho blízkosti. Vozíček držte, dokud nedojde ke spuštění měření. Jednotlivá data uložte do souborů. Pro každou kombinaci závaží (tj. změnu hmotnosti autíčka a závaží) proměřte alespoň třikrát.

3. Naměřené hodnoty zaneste do grafu a stanovte hodnotu zrychlení. Pro ověření správnosti měření zvažte jednotlivá použitá závaží a vozíček. Ze znalosti hmotností a tabulkové hodnoty tíhového zrychlení určete teoreticky hodnotu zrychlení a porovnejte s experimentálně naměřenou hodnotou. Porovnejte získané výsledky.

t (ms)	t (s)	s (cm)	v (cm/s)	v (m/s)	a (m/s^2)

Tabulka 5: Vzorová tabulka pro vyplnění naměřených hodnot

5 Elektřina a magnetismus

Na střední škole bývá elektřině a magnetismu věnováno mnoho času. Jedná se o velmi rozsáhlou kapitolu fyziky. V této práci se budeme pouze věnovat elektrickým obvodům a elektronickým součástkám, ze kterých jsou sestaveny. Pro měření byla sestavena jedna fyzikální úloha, která studenta seznámí se zapojením elektrického obvodu a různými elektronickými součástkami a jejich vlastnostmi.

5.1 Voltampérová charakteristiky pasivních součástek

Úloha 1: Voltampérová charakteristika

Pomůcky Arduino, počítač, odpory různých hodnot, spojovací kit, LED-dioda, Zenerova dioda, dioda, žárovka, propojovací vodiče, potenciometr, chemický zdroj, regulovatelný zdroj, kontrolní multimetr

Teorie

Jednou z charakteristik elektronických součástek je elektrický odpor. Odpor součástky může být proměnný, a to jak s časem, osvětlením, teplotou, tak například s napětím, ke kterému je daná součástka připojena. K tomu, aby bylo možné sledovat, jak se některá součástka chová v elektrickém obvodu s danou hodnotou napětí, slouží voltampérová charakteristika. Ta udává závislost mezi napětím a proudem. Graf voltampérové charakteristiky je specifický pro jednotlivé druhy elektronických součástek.

Rezistor (odpor) je pasivní elektronická součástka, která má proud lineárně závislý na napětí. Lineární závislost vyplývá i z Ohmova zákona [4]:

$$I = \frac{U}{R} \tag{10}$$

Další součástky již nemají lineární průběh proudu na napětí.

Diody jsou polovodičové součástky, které jsou složeny z polovodiče typu N a typu P. Tyto polovodiče se rozlišují podle druhu nosiče náboje. Polovodič typu N má majoritní nosiče elektrony, polovodič typu P díry. Tím je zjevné, že diodu můžeme zapojit ve dvou směrech, propustném a závěrném. V případě zapojení diody v závěrném směru, jí protéká minimální proud, jelikož dojde k roz-

šíření přechodové vrstvy. (Výjimkou je Zenerova dioda.) Pokud diodu zapojíme v propustném směru, dojde ke zúžení přechodové vrstvy. Dioda pro nízké hodnoty nevede proud. Po překročení prahového napětí dojde k prudkému růstu protékajícího proudu [5].

Zenerova dioda se využívá v závěrném směru. Tato dioda snese opakované průrazy, na rozdíl od normálních diod, pro které je proražení v závěrném směru destruktivní. Zenerova dioda se při zapojení v propustném směru chová jako standardní dioda [5].

A odpor žárovky je nelineární a je závislý na napětí, ke kterému je žárovka připevněna. V průběhu času také dochází k ohřívání vlákna, které ovlivňuje odpor žárovky [5].

Úkoly

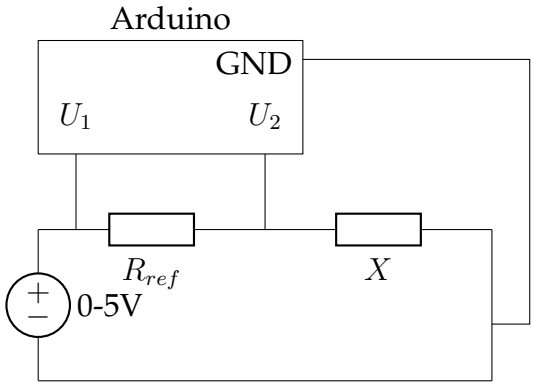
1. Určete rovnici pro výpočet procházejícího elektrického proudu daným obvodem.

2. Proměřte voltampérové charakteristiky přiložených součástek.

3. Zakreslete jednotlivé grafy a diskutujte výsledky s předpoklady.

Postup měření

Pro měření voltampérové charakteristiky je možné sestavit dva druhy zapojení. Při využití regulovatelného zdroje napětí použijte následující schéma:



Obrázek 13: Schéma pro zapojení měřící sestavy pro voltampérovou charakteristiku

Ve schématu je odpor R_{ref}, který slouží jako referenční, z něhož odečítáme protékající proud obvodem. X je součástka, pro kterou měříme voltampérovou charakteristiku.

Ze zapojení je zjevné, že hodnota protékajícího proudu obvodem je dána:

$$I = \frac{U_1 - U_2}{R_{ref}} \tag{11}$$

V případě, kdy není možné použít regulovatelný zdroj, je nutné ho vytvořit za pomoci například chemického zdroje a potenciometru. Další možností by bylo využít PWM výstup Arduina, kde by bylo nutné vyhladit signál, například za pomoci kondenzátoru a dalších součástek.

V případě využití chemického zdroje a potenciometru nejprve pomocí multimetru zkontrolujte, jak velké proudy a napětí protékají obvodem, aby nedošlo ke zničení Arduina.

V tomto měření je nutné snímat konkrétní hodnotu napětí na pinu, a proto je nutné zapojit analogové piny pro čtení hodnot.

Jako ampérmetr zapojte analogový vstup Arduina jako senzor napětí na malém odporu a následným přepočítáním. Obdobně bude Arduino zapojené i pro měření hodnoty napětí na měřené součástce. Pro všechny součástky naměřte alespoň dvacet hodnot s co největším rozsahem, aby byly hodnoty dostatečně rozmístěny pro vykreslování grafů. Při měření postupujeme od minimální hodnoty po maximální hodnotu napětí zdroje. A kontrolujte, zda nebylo dosaženo mezní hodnoty pro Arduino.

U_1 (V)	R_{ref} (Ω)	U_2 (V)	ΔU (V)	I (mA)

Tabulka 6: Vzorová tabulka pro vyplnění naměřených hodnot

6 Kmity, vlny, optika

Kmitání a vlnění nás obklopuje jak v mikroskopickém, tak i makroskopickém světě, od kmitání mrakodrapu nebo mostní konstrukce až po průlet fotonu. Mnoho fyzikálních problémů je popisováno pomocí kmitavého pohybu. Na měření kmitavého pohybu byly sestaveny tři základní úlohy na měření.

6.1 Fyzické a matematické kyvadlo

Úloha 1: Určení hodnoty tíhového zrychlení

Pomůcky

Arduino, počítač, optická závora, matematické kyvadlo, vhodné fyzické kyvadlo

Teorie

Na veškeré objekty na povrchu planety působí tíhová síla.

Tato síla se skládá ze dvou sil, síly gravitační, která působí směrem do středu planety a přitahuje tak těleso k ní, a síly odstředivé, ta naopak působí směrem od planety a je kolmá na její osu otáčení. Je tedy zřejmé, že výsledná tíhová síla je ovlivněna tím, kde se objekt na planetě nachází, je závislá na zeměpisné šířce. Pro určení tíhového zrychlení je nutné vycházet z Newtonova gravitačního zákona [1] [10]:

$$\overrightarrow{F_g} = \kappa \frac{m M_z}{r^2} \frac{\overrightarrow{r}}{r} \tag{12}$$

kde r je vzdálenost objektu od středu Země, M_z je hmotnost planety Země a κ je gravitační konstanta.

Odstředivá síla na povrchu země se vypočítá následovně:

$$\overrightarrow{F_{od}} = m\overrightarrow{a_{od}} \tag{13}$$

V tomto případě je ale vhodnější ji vyjádřit za pomoci úhlové rychlosti, jelikož

ta je na celé planetě stejná, na rozdíl od odstředivého zrychlení. Pak získáváme tvar:

$$\overrightarrow{F_{od}} = m\overrightarrow{r_0}\omega_z^2 \tag{14}$$

kde m je hmotnost tělesa, na které daná síla působí, ω je úhlová rychlost otáčení planety a r_0 je vektor, který má směr jako odstředivá síla a jeho velikost je rovna vzdálenosti tělesa od osy otáčení. Je tedy zjevné, že s polohou na zemi bude docházet ke změně velikosti odstředivé síly, která bude největší na rovníku a nulová na místech, kde osa rotace prochází povrchem země.

Pro tíhovou sílu platí na základě superpozice sil:

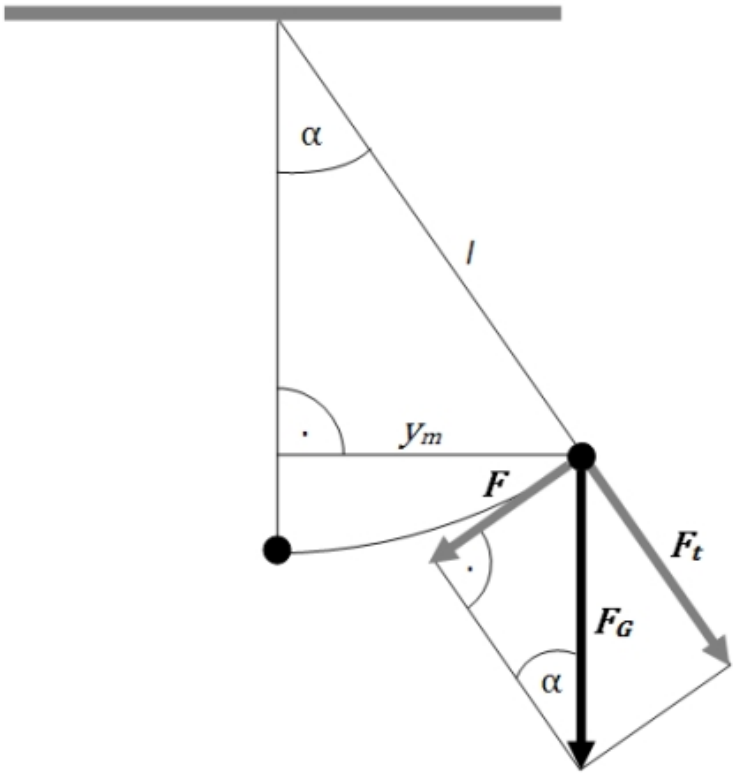
$$\overrightarrow{F_G} = \overrightarrow{F_g} + \overrightarrow{F_{od}} \tag{15}$$

Z toho pro tíhové zrychlení dostáváme i pomocí druhého Newtonova pohybového zákona [1]:

$$\overrightarrow{g} = \overrightarrow{a_g} + \overrightarrow{a_{od}} \tag{16}$$

Pro měření tíhového zrychlení je možné využít matematické kyvadlo. Na matematické kyvadlo působí několik sil, které jsou znázorněné na
obrázku 14. Při vychýlení kyvadla z rovnovážné polohy dochází k rozkladu tíhové síly $\overrightarrow{F_G}$ na sílu $\overrightarrow{F}$, která uvádí těleso do pohybu, a kolmou složku na ni.

Na obrázku 14 jsou znázorněny síly působící na matematické kyvadlo. F_G je tíhová síla, F_t je tahová síla, kterou působí hmotný bod na závěs. A síla F, která navrací hmotný bod do rovnovážné polohy. l je délka závěsu, α je úhel, o který se kyvadlo vychýlilo, a y_m je výchylka hmotného bodu.



Obrázek 14: Znázornění sil působících na matematické kyvadlo [38]

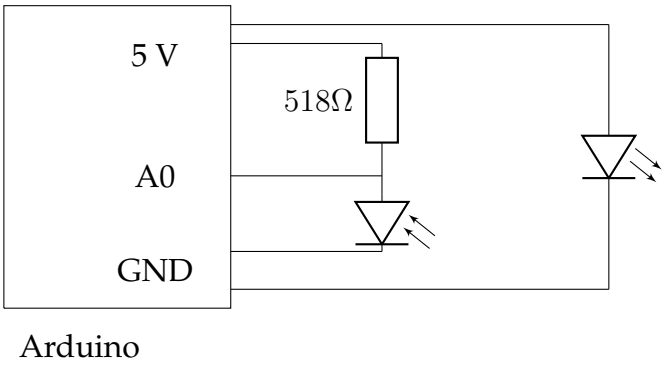
Úkoly

1. Odvoďte rovnici pro matematické kyvadlo.
2. Proměřte periodu kmitu matematického kyvadla.
3. Určete hodnotu tíhového zrychlení z naměřené hodnoty doby kmitu.

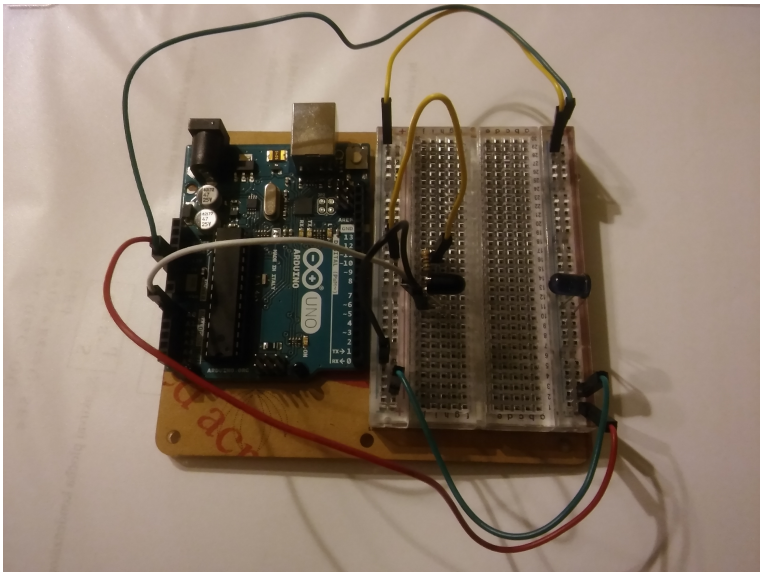
Postup měření

Připravte si matematické kyvadlo a proměřte délku jeho závěsu. Určíte místo, kde je kyvadlo v rovnovážné poloze a do daného místa umístěte optickou závoru, za pomoci které budete zjišťovat, kdy jí prošlo matematické kyvadlo.

Schéma optické závory:



Obrázek 15: Schéma zapojení optické závory pomocí platformy Arduino



Obrázek 16: Zapojení domácí optické závory

Pro měření je možné využít infra LED diodu TSAL6100 a fotodiodu BPV10NF. Vhodně nastavíme měřicí zařízení pro zvolené elektronické součástky.

U (V)	t (ms)	t (s)	T (s)	g (m/s^2)

Tabulka 7: Vzorová tabulka pro vyplnění naměřených hodnot

Měření provedeme alespoň pro tři délky závěsu. Prodiskutujte vzniklé chyby, ke kterým došlo při měření.

Úloha 2:

Určete dekrement útlumu fyzického kyvadla

Pomůcky

Fyzické kyvadlo, potenciometr, arduino, počítač, propojovací vodiče

Teorie

Každé reálné kyvadlo je kyvadlem fyzickým, které se může jen přibližovat idealizovanému matematickému kyvadlu. Ve skutečnosti však žádný objekt nebude kmitat do nekonečna. V průběhu času se energie dodaná kyvadlu vlivem tření a odporu vzduchu přeměňuje na jiné druhy energie, a tak dochází k tlumení kmitů.

Na veškeré pohybující se objekty na zemi působí odporová síla, která je přímo úměrná rychlosti pohybu tělesa.

$$\overrightarrow{F_t} = -r\overrightarrow{v} \tag{17}$$

kde r je koeficient odporu prostředí, tato síla má opačný směr než síla, která uvádí těleso do pohybu, proto je ve vzorci znaménko mínus.

Pro odvození rovnice pro dekrement útlumu budeme předpokládat, že jako oscilátor máme pružinu o tuhosti k. Kmitavá síla pružiny je:

$$\overrightarrow{F_p} = -k\overrightarrow{y} \tag{18}$$

kde k je tuhost pružiny a y je výchylka od rovnovážné polohy.

Síla, která působí na oscilátor, je:

$$\overrightarrow{F} = \overrightarrow{F_t} + \overrightarrow{F_p} \tag{19}$$

Po dosazení z rovnic (20) a (21) dostáváme:

$$m\overrightarrow{a} = -k\overrightarrow{y} - r\overrightarrow{v} \tag{20}$$

Pokud rovnici upravíme do diferenciálního tvaru, získáme:

$$\frac{d^2y}{dt^2} + \frac{r}{m}\frac{dy}{dt} + \frac{k}{m}y = 0 \tag{21}$$

Tato pohybová rovnice má standardní řešení ve tvaru $y = Asin(\omega t + \phi_0)$

A je velikost amplitudy, která u tlumených kmitů nemá konstantní hodnotu, ale je proměnná dle rovnice (24).

$$A = A_0 e^{-\frac{r}{2m}t} \tag{22}$$

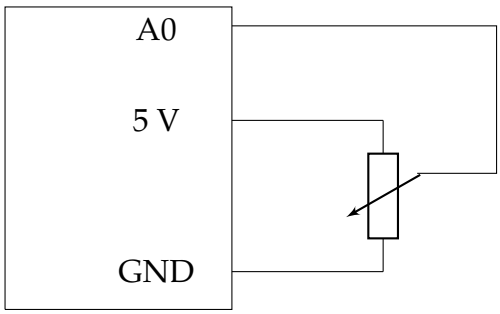
V rovnici (22) vystupuje zlomek, který nazveme dekrement útlumu [9].

Úkoly

1. Zaznamenejte průběh pohybu fyzického kyvadla a vyneste graf.
2. Vypočítejte dekrement útlumu a diskutujte vznik chyb.

Postup měření

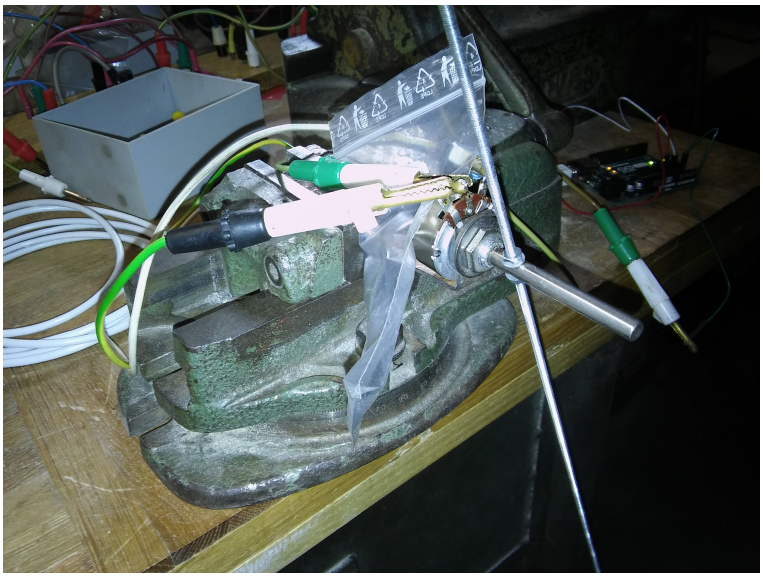
Pro měření útlumu fyzického kyvadla využijte měření jeho výchylky v průběhu času. Na měření výchylky využijte potenciometr, na který lze přenášet výchylku, a tak měnit hodnotu jeho vnitřního odporu. Jeden z možných způsobů, jak propojit kyvadlo s potenciometrem je na obrázku 18. Tím budete měřit vyvolanou změnu napětí. Pro zapojení využijte schématu, které je na obrázku 17.



Obrázek 17: Schéma zapojení potenciometru pro měření fyzického kyvadla

Po sestavení obvodu nastavte Arduino jako snímač napětí na analogovém pinu. Kyvadlo vychylte z rovnovážné polohy a spusťte měření. Měření opakujte pro různá postavení závaží na kyvadle.

Pro jednotlivá měření vyneste grafy a určete dekrement útlumu.



Obrázek 18: Spojení fyzického kyvadla a potenciometru

U (V)	t (ms)	t (s)	t_{max} (s)	t_{min} (s)

Tabulka 8: Vzorová tabulka pro vyplnění naměřených hodnot

6.2 Ohnisková vzdálenost

Úloha 3:

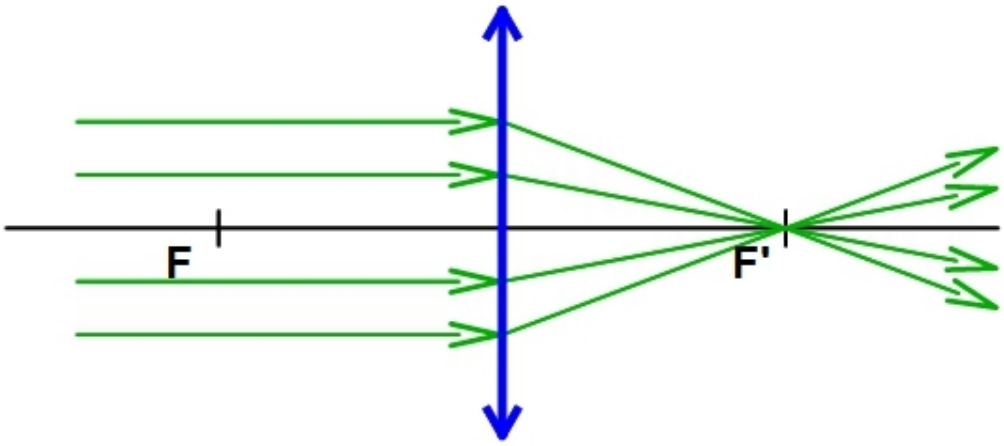
Určete ohniskovou vzdálenost spojné čočky.

Pomůcky

Arduino, pořítač, fotorezistor, fotodioda, zdroj světla, spojná čočka, měřítko

Teorie

Spojná čočka je charakteristická tím, že paprsky rovnoběžné s optickou osou láme do ohniska viz obrázek 19



Obrázek 19: Lom rovnoběžných paprsků spojnou čočkou [39]

Hlavní charakteristikou každé čočky je její ohnisková vzdálenost a zvětšení. Pro spojnou čočku platí zobrazovací rovnice [18]:

$$\frac{1}{a} + \frac{1}{a'} = \frac{1}{f} \tag{23}$$

Z rovnice (23) lze vyjádřit ohniskovou vzdálenost následovně:

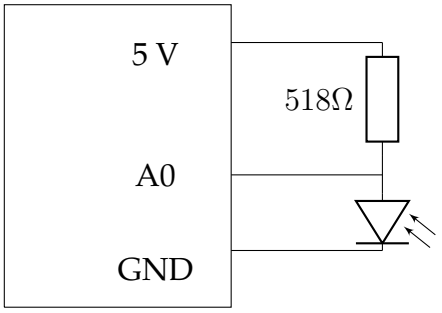
$$f = \frac{a \cdot a'}{a + a'} \tag{24}$$

Úkoly

1. Proměřte ohniskovou vzdálenost spojné čočky.

Postup měření

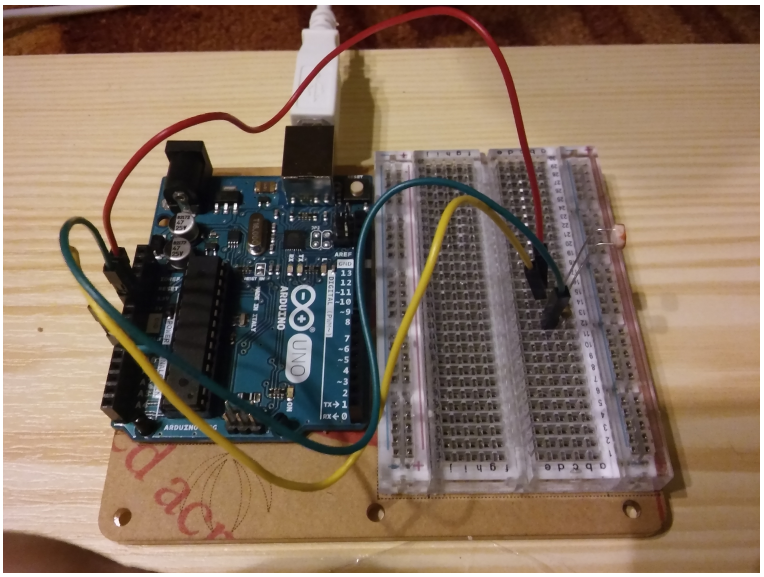
Do jedné přímky na měřítko sestavte zdroj světla, čočku a foto-činný prvek a snímejte na něm hodnotu napětí v závislosti na vzdálenosti
od spojné čočky. Měřicí systém sestavte dle následujícího schématu:



Obrázek 20: Schéma zapojení měřícího systému pro měření ohniskové vzdálenosti

Pro měření je možné využít fotodiodu popřípadě fotorezistor.

Zapněte zdroj světla, pomocí stínítka ověřte, že světlo prochází čočkou. Následně pohybujte měřícím systémem po měřítku a nalezněte místo s extremální hodnotou napětí. Zaznamenejte vzdálenost fotorezistoru nebo fotodiody od spojné čočky. Měření opakujte alespoň pětkrát. Následně změňte vzdálenost zdroje světla od spojky. Měření proveďte pro různé vzdálenosti zdroje světla a diskutujte výsledky.



Obrázek 21: Zapojení pro měření ohniskové vzdálenosti

U_{ard} (V)	d (cm)

Tabulka 9: Vzorová tabulka pro vyplnění naměřených hodnot

Pro zdatnější je možné nakombinovat měření ohniskové vzdálenosti spolu s měřením vzdálenosti a využít zároveň modulu sonar pro určování vzdálenosti, nikoli běžného odečítání hodnot z měřítka.

Závěr

Platforma Arduino je velmi rozšířená a má mnoho různých podob. Tím je velmi variabilní, a tak je možné obsáhnout mnoho problémů, které je s její pomocí možno zkoumat. Tato bakalářská práce ukazuje, jak je možné ji využít při zkoumání základních fyzikálních úloh. A také jak propojit oblast informačních technologií, zpracování dat a vlastního měření. Platformu Arduino by bylo možné využít i pro další oblasti fyziky. V současné době existuje mnoho přídavných modulů, které by bylo možné využít například v termice, optice a nebo částicové fyzice.

Řešení jednotlivých protokolů se nachází v přílohách. Při vypracování jednotlivých měření se student seznamuje s konstrukcí a programováním dané platformy. V současné chvíli tato práce obsahuje pouze zanedbatelný okruh problémů, kde by bylo možné platformu využít.

Vlastní měření jednotlivých úloh časovou náročností odpovídá přibližně na jednu až dvě vyučovací hodiny, to dle druhu úlohy a počtu opakování měření. Největší časovou náročnost má vlastní vyhodnocení dat. Množství naměřených dat je dáno hlavně vzorkovací frekvencí, kterou je nutné volit s rozvahou, ideální je provést pár měření s různými vzorkovacími frekvencemi a dle naměřených dat vybrat nejvhodnější vzorkovací frekvenci.

Měření s platformou Arduino může probíhat se vzorkovací frekvencí až 16 MHz. Proto je velmi důležité se primárně naučit s jakým časovým rozestupem bude docházet k měření hodnot. Při využití vyšší vzorkovací frekvence, získáváme velké množství hodnot. Pro některá měření může být vysoká vzorkovací frekvence nevhodná, na příklad pro měření vzdálenosti. Měřící modul, který byl pro měření vzdálenosti využit, je schopný měřit s přesností na 3 mm. Chyba měření modulu spolu s vysokou vzorkovací frekvencí vyvolá chybu měření. Naopak pro měření matematického kyvadla je vhodné využit vyšší vzorkovací frekvence, která umožňuje určit přesnější hodnotu, kdy kyvadlo procházelo rovnovážnou polohou.

Práce z mého pohledu naplnila vše, co jsme od ní očekávala. Je mnoho ob-

lastí, kam by bylo možné tuto práci rozšiřovat. Také by bylo vhodné zaměřit se podrobněji na zde již zpracované úlohy, vytvořit další možnost jejich řešení, které by mělo menší chyby měření.

Seznam obrázků

Seznam tabulek

Zdroje

Seznam použité literatury

[1] BEDNAŘÍK, Michal, Ondřej JIŘÍČEK a Petr KONÍČEK. *Fyzika I a II: laboratorní cvičení*. Dot. 1. vyd. Praha: Vydavatelství ČVUT, 1997. ISBN 80-01-01296-4.

[2] FEYNMAN, Richard Phillips, Robert B LEIGHTON a Matthew SANDS. *Feynmanovy přednášky z fyziky: s řešenými příklady*. 1. vyd. Havlíčkův Brod: Fragment, 2000-. ISBN 80-7200-405-0.

[3] HEROUT, Pavel. *Učebnice jazyka C. 1. vyd*. České Budějovice: Kopp, 1992. ISBN 80-901342-1-1.

[4] HUBEŇÁK, Josef a Jiří HUBEŇÁK. *Aplikovaná fyzika a moderní elektronika*. Vyd. 1. Ostrava: Repronis, 2007. ISBN 978-80-7329-158-7.

[5] HUBEŇÁK, Josef. *Elektronika: učební text pro 3. ročník studia učitelství fyziky*. 1. vyd. Hradec Králové: Gaudeamus, 1991. ISBN 80-7041-348-4.

[6] HUBEŇÁK, Josef. *Fyzikální měření pro učitele*. Vyd. 1. Hradec Králové: Gaudeamus, 2006. ISBN 80-7041-236-4.

[7] HUBEŇÁK, Josef. *Technika a fyzika: studijní materiál pro další vzdělávání učitelů fyziky*. Hradec Králové: Gaudeamus, 1996. Scio me multa nescire. ISBN 80-7041-685-8.

[8] KAČMÁŘ, Dalibor. *Jazyk C*. Vyd. 1. Praha: Computer Press, 2000. Učebnice pro střední školy. ISBN 80-7226-295-5.

[9] KOPEČNÝ, Jan. JANUROVÁ, Eva. FOUKAL, Jaroslav, BARČOVÁ, Karla. UHLÁŘ, Radim. KUŠNEROVÁ, Milena. *Fyzika pro bakaláře* 1. vyd. Ostrava: VŠB - Technická univerzita Ostrava, 2006. ISBN: 80-248-1200-2. [Cit. 10.5.2016] Dostupné z: http://www.studopory.vsb.cz/studijnimaterialy/Fyzikaprobakalare /PDF/1_7_2_tlum.pdf

[10] LIBRA, Martin, Václav VACEK a František ČERNÝ. *Laboratorní cvičení z fyziky I*. Vyd. 1. Praha: Vydavatelství ČVUT, 1998. ISBN 80-01-01737-0.

[11] LOŠŤÁK, Jiří. *Lexikon fyziky: (přehled učiva ZŠ a SŠ)*. 2. opr. a rozš. vyd. Olomouc: Nakladatelství Olomouc, 1997. ISBN 80-7182-022-9.

[12] MARGOLIS, Michael, 2012. *Arduino Cookbook, Second Edition* [Cit. 10.5.2016] Dostupné z: https://drive.google.com/folderview?id=0B3oJJ-xRGOqWbWU3RHZ6SXE5dE0

[13] MELGAR, Enrique Ramos and DIEZ, Ciriaco Castro with JAWORSKI Przemek, *Arduino and Kinect Projects*. [Cit. 10.5.2016] Dostupné z: https://drive.google.com/folderview?id=0B3oJJ-xRGOqWbWU3RHZ6SXE5dE0

[14] NOVÁKOVÁ, Danuše. *Laboratorní cvičení z fyziky II*. 2. vyd. Praha: Vydavatelství ČVUT, 1999. ISBN 80-01-01387-1.

[15] PRATA, Stephen. *Mistrovství v C++. 4., aktualiz. vyd*. Přeložil Boris Sokol. Brno: Computer Press, 2013. Bestseller. ISBN 978-80-251-3828-1.

[16] SVOBODA, Emanuel. *Přehled středoškolské fyziky*. 1. vyd. Praha: Státní pedagogické nakladatelství, 1991. Kostka 43. ISBN 80-04-22435-0.

[17] ŠALOUN, Petr. *Jazyk C pro zelenáče*. Praha: Neokortex, 1999. Bestseller for all. ISBN 80-86330-02-X.

[18] TOMAN, Jan a Petr SEMERÁK. *Fyzika 10: praktická cvičení*. 1. vyd. Praha: Vydavatelství ČVUT, 1996. ISBN 80-01-01447-9.

[19] VEČERKA, Arnošt. *Jazyk C++: popis jazyka s příklady*. 1. vyd. Olomouc: Vydavatelství Univerzity Palackého, 1996. ISBN 80-7067-658-2.

[20] VODA, Zbyšek a Tým HW kitchen, 2015. *Průvodce světem Arduina* [online] 1. vyd. Bučovice: nakladatelství Martin Stříž, 2015, ISBN: 978-80-87106-90-7 [Cit. 20.10.2015] Dostupné z: http://arduino.cz/

[21] Lekce 9 - Měříme vzdálenost s HC-SR04. *Arduino* [online]. Poslední změna: 06.03.2013 v 14:54 [Cit. 26.1.2016]. Dostupné z: http://arduino8.webnode.cz/news/lekce-9-merime-vzdalenost-s-hc-sr04/

[22] SIK Experiment Guide for Arduino - V3.2. *sparkfun* [online]. [Cit. 23.11.2015]. Dostupné z: https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino—v32

[23] Alza.cz a.s., Arduino UNO Rev3. *Alza* [online]. [Cit. 23.11.2015]. Dostupné z: https://www.alza.cz/arduino-uno-rev3-d569244.htm?o=10

[24] Alza.cz a.s., Arduino Nano V3.0. *Alza* [online]. [Cit. 23.11.2015]. Dostupné z: https://www.alza.cz/arduino-nano-v3-0-d569054.htm?o35#popis

[25] Rychlost zvuku. *Wikipedie* [online]. Poslední změna: 18.4.2016 v 08:57 [Cit. 10.5.2016]. Dostupné z: http://cs.wikipedia.org/wiki/Rychlost_zvuku

[26] KYSILKA Pavel 2011. Arduino I, *Linuxsoft.cz* [online]. Poslední změna: 21.11.2011 [Cit. 20.10.2015]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=1881

[27] Arduino. *Wikipedie* [online]. Poslední změna: 11.5.2016 v 19:13 [Cit.11.5.2016]. Dostupné z: https://cs.wikipedia.org/wiki/Arduino

[28] Blink. *Arduino* [online]. Poslední změna: 28.07.2015 [Cit. 26.01.2016] Dostupné z: https://www.arduino.cc/en/tutorial/blink

[29] KYSILKA Pavel 2011. C/C++ (19) - Příkaz switch a bitové operátory. *Linuxsoft.cz* [online]. Poslední změna: 4.5.2005 v 7:00 [Cit. 21.11.2015]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=741

[30] REICHL Jaroslav, VŠETIČKA Martin. Matematické kyvadlo. *Encyklopedie fyziky* [online]. [Cit. 10.3.2016]. Dostupné z: http://fyzika.jreichl.com/main.article/view/205-matematicke-kyvadlo

[31] [online] [Cit 23.11.2015] http://4.bp.blogspot.com/-UwY3X19nXe8/UET1ZgU_xGI/

AAAAAAAAAzA/1VJSxKnypSU/s1600/
ArduinoUno_R3_FrontScheme.jpg

[32] MAIN 2006. Tabulka ASCII znaků *tyl.cz* [online]. Poslední změna: 08.02.2006 [Cit. 10.5.2016]. Dostupné z: http://www.tyl.cz/Pocitace/Internet/Tabulka-ASCII-znaku.html

Seznam zdrojů obrázků

[34] SIK Experiment Guide for Arduino - V3.2. *sparkfun* [online]. [Cit. 23.11.2015]. Dostupné z: https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino—v32

[35] [online] [Cit 23.11.2015]. Dostupné z: http://becuo.com/arduino-uno-r3

[36] [online] [Cit 23.11.2015]. Dostupné z: http://4.bp.blogspot.com/-UwY3X19nXe8/UET1ZgU_xGI/AAAAAAAAAzA/1VJSxKnypSU/s1600 /ArduinoUno_R3_FrontScheme.jpg

[37] [online] [Cit. 5.4.2016]. Dostupné z: http://www.ddp.fmph.uniba.sk/k̃oubek /UT_html/ut02_html/2-5_soubory/image008.gif

[38] Mgr. SZOTKOWKSI, Henrik. 2011. Matematické kyvadlo *Webová sbírka řešených příkladů z fyziky* [online] Poslední úpravy: 30.9.2011 [Cit. 10.4.2016]. Dostupné z: http://www.sbirkaprikladu.cz/userfiles/6/image/c8-obr.jpg

[39] [online]. [Cit. 10.4.2016]. Dostupné z: https://eluc.kr-olomoucky.cz/uploads/images/12412/content_obr2.jpg

[40] Arduino Ulatrasonic Sonar Module, *HITRONICS Solutions* [online]. [Cit. 23.11.2015]. Dostupné z: http://www.hitronics.net/index.php?route=product/product&product_id=61

[41] Druhý Newtonův pohybový zákon (zákon síly), *FYZIKA 007* [online]. [Cit. 10.4.2016]. Dostupné z: http://www.fyzika007.cz/mechanika/druhy-newtonuav-pohybovy-zakon

Přílohy: Vypracované protokoly

Protokol z fyzikálních měření Mechanika

Jméno:	Andrea Hladíková
Obor:	P–BFY, P–BMAT
Školní rok:	2015/2016
Ročník:	třetí
Naměřeno:	18. 4. 2016
Odevzdáno:	13. 5. 2016
Úloha:	č. 1

Téma: Měření rychlosti

Pomůcky: Arduino, počítač, modul sonar, měřítko, stínítko

Úkoly:

a) Zjistěte rychlost pohybujícího se vozíku.

b) Zakreslete grafy vývoje vzdálenosti na čase s = f(t) pro jednotlivá měření a proveďte diskuzi sledovaného jevu.

Vlastní měření: Pro měření vzdálenosti využijeme následující zdrojový kód:

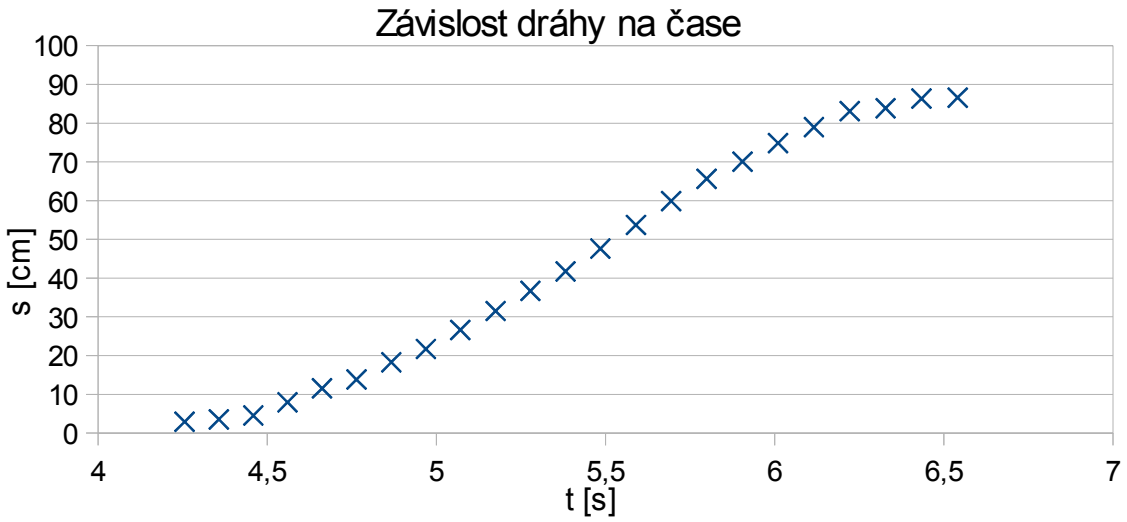
```
int ECHOPIN = 3;
int TRIGPIN = 2;

void setup () {
  Serial.begin(9600);
   pinMode(ECHOPIN, INPUT);
   pinMode(TRIGPIN, OUTPUT);
}
```

```
void loop() {

 digitalWrite(TRIGPIN, LOW);
 delayMicroseconds(2);
 digitalWrite(TRIGPIN, HIGH);
 delayMicroseconds(10);
 digitalWrite(TRIGPIN, LOW);

 float distance = pulseIn(ECHOPIN, HIGH);
 distance= distance*0.017315;
 t= millis();
 Serial.print(distance);
 Serial.print("  ");
 Serial.print("cm");
 Serial.print("  ");
 Serial.print(t);
 Serial.println();
 delay(50);
}
```

Zdrojový kód je převzat z internetu [21].

Za pomoci daného zdrojového kódu, který byl nahrán do platformy Arduino, byla získána následující data a následně zpracována viz tabulka 10 a obrázek 22.

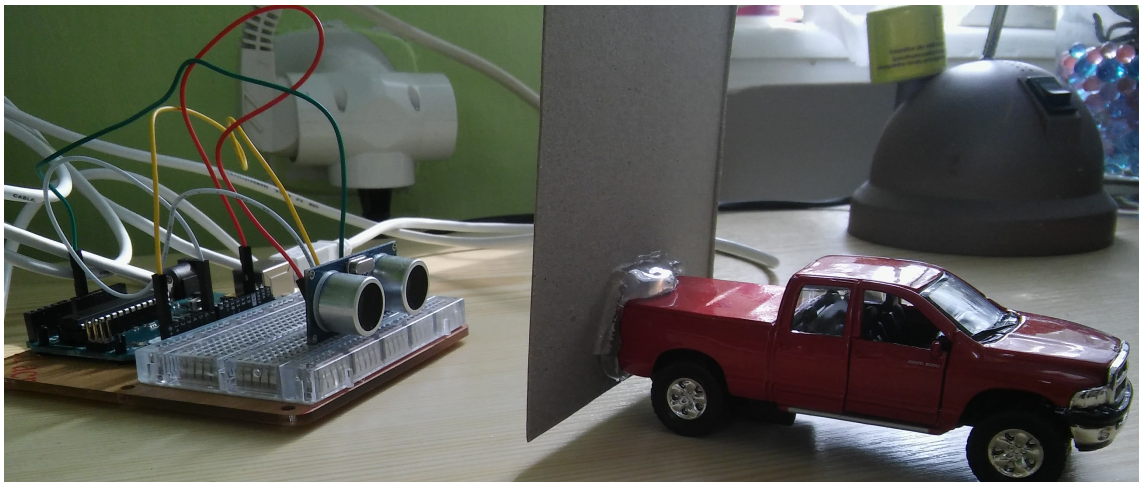
Měření probíhalo za pomoci zakoupeného autíčka na setrvačník, kterému bylo nutné zvětšit zadní odrazovou část pro sonar.



Obrázek 22: Graf vývoje vzdálenosti na čase

Tabulka 10: Měření vzdálenosti autíčka v čase

n	t **(ms)**	t **(s)**	s **(cm)**
1	4256	4,256	2,94
2	4357	4,357	3,55
3	4459	4,459	4,54
4	4560	4,56	7,96
5	4662	4,662	11,55
6	4764	4,764	13,83
7	4867	4,867	18,27
8	4969	4,969	21,70
9	5071	5,071	26,61
10	5175	5,175	31,50
11	5278	5,278	36,69
12	5382	5,382	41,73
13	5485	5,485	47,62
14	5590	5,59	53,73
15	5694	5,694	59,91
16	5799	5,799	65,64
17	5905	5,905	70,06
18	6010	6,01	74,84
19	6116	6,116	78,97
20	6222	6,222	83,08
21	6328	6,328	83,86
22	6434	6,434	86,32
23	6541	6,541	86,58



Obrázek 23: Sestava pro měření vzdálenosti

Závěr a diskuse: Vlastní měření vzdálenosti není nikterak komplikované, ale může být rušeno okolními vlivy. Vzhledem ke skutečnosti, že měření probíhá pomocí zvuku, který se šíří ve vlnoplochách ve všech směrech, mohou se projevovat vlivy okolí na odražený zvukový signál. Proto je nutné, aby místo, od kterého se odráží daný signál, bylo dostatečně velké a přitom v okolí bylo minimum rušivých jevů. Pokud dostatečně eliminujeme nežádoucí jevy z okolí, je možné měřit hodnoty vzdálenosti, tak i času s poměrně velkou přesností. Občas je ale daná přesnost až na škodu. Signál byl vysílán v rozestupu 50 ms, vzhledem k tomu, aby bylo dostatečné množství dat. Pomocí polynomické regrese jsou daná data přesná a odpovídají předpokladům pro vývoj dráhy na čase. Pokud bychom ale chtěli z daných hodnot vypočítat zrychlení pomocí numerické derivace, byly by dané hodnoty zatíženy velkou chybou, která je způsobena chybami měření vzdálenosti a tím, že by docházelo k dělení pomocí hodnot, které se blíží nule. Proto je nutné data naředit, tedy vybrat pouze některá a nebo využít zmiňované polynomické regrese. Na grafu je zjevné jak probíhá pohyb autíčka, které bylo využíváno na daný pokus. V první části dochází k rozjíždění, kdy objekt zrychluje a závislost jeho dráhy na čase má parabolický tvar. Po chvíli dochází k ustálení pohybu a těleso se pohybuje rovnoměrnou rychlostí, v daný okamžik dráha v čase lineárně roste. Poté, co již tahová síla motorku zeslábne, začnou převažovat odporové síly prostředí. Začne docházet ke zpomalování.

Protokol z fyzikálních měření
Mechanika

Jméno:	Andrea Hladíková
Obor:	P–BFY, P–BMAT
Školní rok:	2015/2016
Ročník:	třetí
Naměřeno:	18. 4. 2016
Odevzdáno:	13. 5. 2016
Úloha:	č. 2

Téma: 2. Newtonův pohybový zákon a nakloněná rovina

Úloha: Druhý Newtonův pohybový zákon

Pomůcky: Arduino, počítač, modul sonar, kladka, spojovací materiál, závaží různých hmotností, vozíček, váhy

Ukol:

a) Ověřte platnost druhého Newtonova pohybového zákona, změřte závislost dráhy na čase pro jednotlivé kombinace závaží.

b) Zakreslete grafy pro jednotlivá měření, vývoje vzdálenosti na čase s = f(t), okomentujte naměřené hodnoty.

Vlastní měření: Pro měření vzdálenosti využijeme následující zdrojový kód:

```
int ECHOPIN = 3;
int TRIGPIN = 2;

void setup () {
  Serial.begin(9600);
   pinMode(ECHOPIN, INPUT);
```

```
  pinMode(TRIGPIN, OUTPUT);
}

void loop() {

 digitalWrite(TRIGPIN, LOW);
 delayMicroseconds(2);
 digitalWrite(TRIGPIN, HIGH);
 delayMicroseconds(10);
 digitalWrite(TRIGPIN, LOW);

 float distance = pulseIn(ECHOPIN, HIGH);
 distance= distance*0.017315;
 t= millis();
  Serial.print(distance);
  Serial.print("  ");
  Serial.print("cm");
  Serial.print("  ");
  Serial.print(t);
  Serial.println();
 delay(50);
}
```

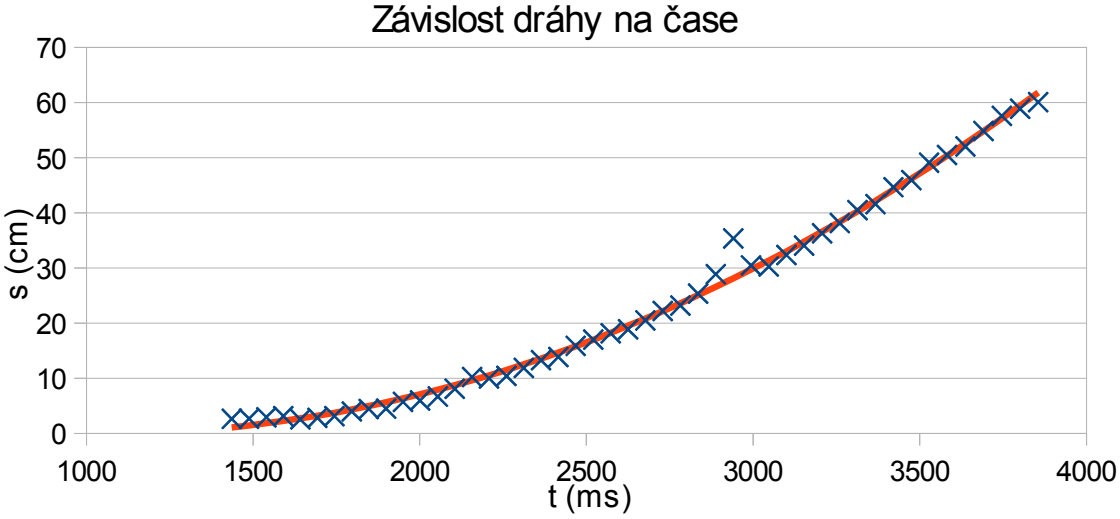
Zdrojový kód je převzat z internetu [21].

Měření probíhalo za pomoci sestaveného vozíku z LEGO stavebnice. Vozík se skládal pouze z podvozku a místa pro umístění závaží a napojení vodícího vlákna. Jako závaží sloužily různě hmotné sáčky s korálky.

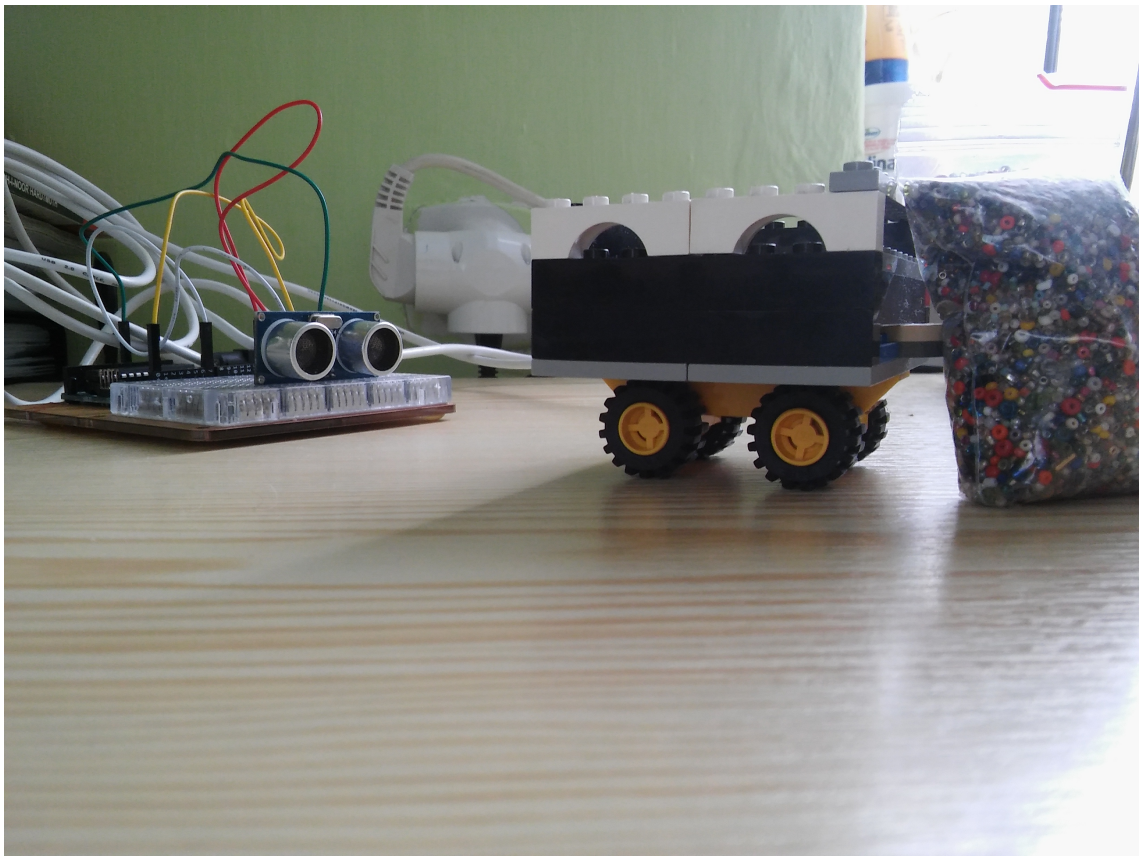
Naměřené hodnoty:

Tabulka 11: Naměřené hodnoty vzdálenosti a času

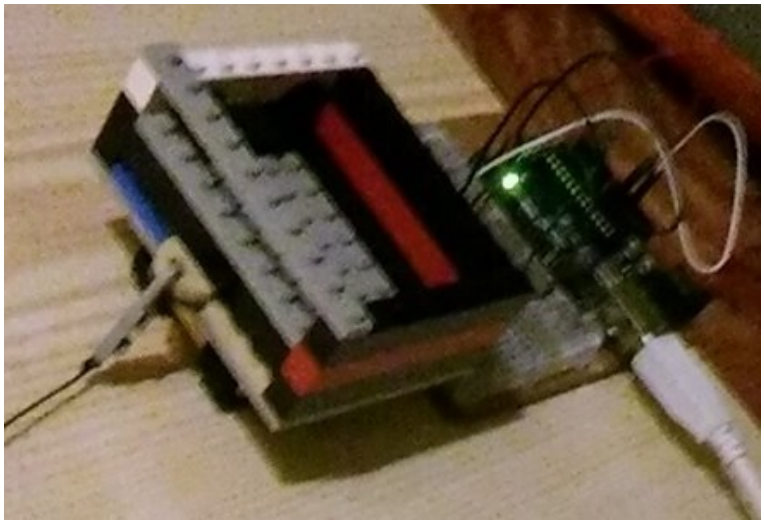
n	s (cm)	t (ms)	t (s)
1	3,50	1026	1,026
2	4,03	1077	1,077
3	4,55	1128	1,128
4	6,53	1179	1,179
5	7,41	1231	1,231
6	9,89	1283	1,283
7	10,89	1335	1,335
8	13,18	1386	1,386
9	15,51	1438	1,438
10	19,08	1490	1,49
11	21,68	1543	1,543
12	24,80	1596	1,596
13	28,86	1648	1,648
14	32,08	1701	1,701
15	36,60	1755	1,755
16	40,95	1808	1,808
17	47,15	1862	1,862
18	48,90	1916	1,916



Obrázek 24: Graf vývoje vzdálenosti na čase pro měření 2.Newtonova pohybového zákona



Obrázek 25: Sestava pro měření 2.Newtonova pohybového zákona



Obrázek 26: Detail vozíku

Závěr a diskuse: Naměřené hodnoty byly zpracovány pomocí polynomické regrese, a to pro polynom druhého stupně. Z koeficientů polynomu lze získat hodnotu pro zrychlení a rychlost daného vozíčku. Hodnota zrychlení byla vypo-

čítána na:

$$a = 1,03 \text{ m s}^{-2}$$

Při teoretickém výpočtu pro ověření hodnoty zrychlení byla měřena hmotnost vozíku a závaží. Hmotnost vozíku je 62 g, ten byl v tomto měření zatížen závažím o hmotnosti 100 g a urychlován závažím o hmotnosti 51 g. Při teoretickém výpočtu byla získána hodnota zrychlení:

$$a = 2,35 \text{ m s}^{-2}$$

Dle předpokladu je teoreticky vypočítaná hodnota vyšší než hodnota naměřená, a to vzhledem k faktu, že při teoretických výpočtech dochází
k zanedbání tření a odporu prostředí.

Odpor prostředí je dán kombinací materiálu, po kterém se vozíček pohybuje, koleček vozíčku, tření kladky a její setrvačnosti, třením vlákna o kladku a odporem vzduchu. V tomto případě bylo tření ještě zvýšeno nevhodnou volbou vozíčku, jehož gumová kolečka přilnula k povrchu. Tuto teorii by bylo nutné ověřit pomocí dalšího měření (např. siloměrem). Měření bylo provedeno pro několik kombinací hmotností vozíčku a zátěže. Veškerá měření byla ovlivněna obdobně velkou chybou, z toho usuzuji, že se jedná o chybu, která je způsobena odporem prostředí, a tedy ztrátou energie při rozjezdu i pohybu vozíčku.

Pro konstrukci tohoto pokusu doporučuji zvolit větší hmotnosti závaží aby bylo možné pomocí siloměru změřit odporovou sílu prostředí. Při tomto měření byly odporové síly tak slabé, že je nebylo možné siloměrem naměřit.

Protokol z fyzikálních měření
Mechanika

Jméno:	Andrea Hladíková
Obor:	P–BFY, P–BMAT
Školní rok:	2015/2016
Ročník:	třetí
Naměřeno:	18. 4. 2016
Odevzdáno:	13. 5. 2016
Úloha:	č. 3

Téma: 2. Newtonův pohybový zákon a nakloněná rovina

Úloha: Druhý Newtonův pohybový zákon

Pomůcky: Arduino, počítač, modul sonar, kladka, spojovací materiál, závaží různých hmotností, vozíček, váhy

Ukol:

a) Ověřte platnost druhého Newtonova pohybového zákona, změřte závislost dráhy na čase pro jednotlivé kombinace závaží.

b) Zakreslete grafy pro jednotlivá měření, vývoje vzdálenosti na čase a rychlosti na čase s = f(t) a v = f(t), okomentujte naměřené hodnoty.

Vlastní měření: Pro měření vzdálenosti využijeme následující zdrojový kód:

```
int ECHOPIN = 3;
int TRIGPIN = 2;

void setup () {
  Serial.begin(9600);
   pinMode(ECHOPIN, INPUT);
```

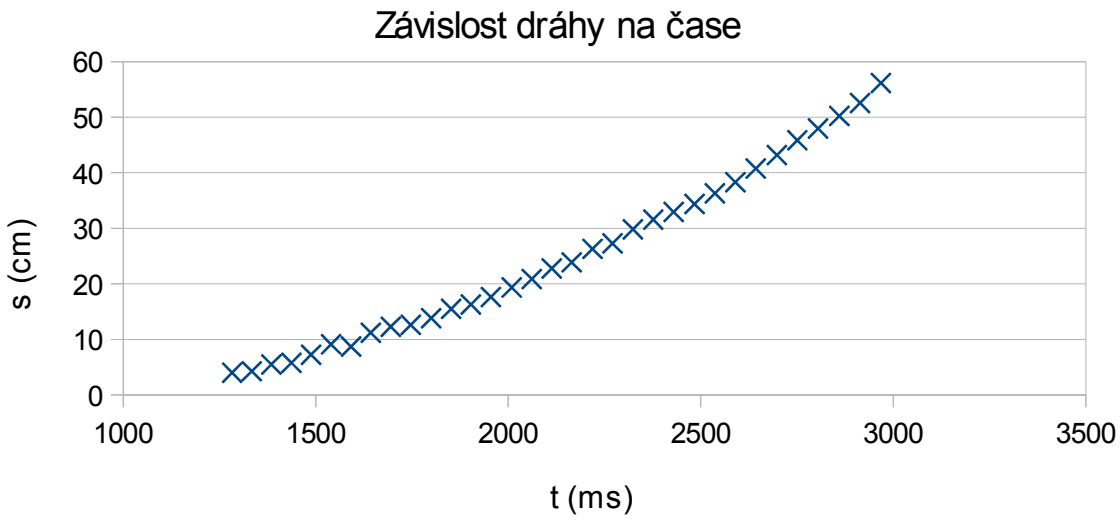
```
  pinMode(TRIGPIN, OUTPUT);
}

void loop() {

 digitalWrite(TRIGPIN, LOW);
 delayMicroseconds(2);
 digitalWrite(TRIGPIN, HIGH);
 delayMicroseconds(10);
 digitalWrite(TRIGPIN, LOW);

 float distance = pulseIn(ECHOPIN, HIGH);
 distance= distance*0.017315;
 t= millis();
  Serial.print(distance);
  Serial.print("  ");
  Serial.print("cm");
  Serial.print("  ");
  Serial.print(t);
  Serial.println();
 delay(50);
}
```

Zdrojový kód je převzat z internetu [21].

Měření probíhalo za pomoci sestaveného vozíku z LEGO stavebnice. Vozík se pouze skládal z podvozku a místa pro umístění závaží a napojení vodícího vlákna. Jako závaží sloužily různě hmotné sáčky s korálky.

Tabulka 12: Naměřené hodnoty pro 2. Newtonův pohybový zákon na nakloněné rovině

n	s (cm)	t (ms)	t (s)
1	4,02	1283	1,283
2	4,28	1334	1,334
3	5,49	1385	1,385
4	5,80	1437	1,437
5	7,25	1488	1,488
6	9,11	1540	1,540
7	8,71	1592	1,592
8	11,20	1643	1,643
9	12,28	1695	1,695
10	12,61	1747	1,747
11	13,83	1800	1,800
12	15,50	1852	1,852
13	16,28	1903	1,903
14	17,63	1955	1,955
15	19,34	2009	2,009
16	20,86	2061	2,061
17	22,77	2113	2,113
18	23,88	2165	2,165
19	26,30	2219	2,219
20	27,29	2271	2,271
21	29,83	2324	2,324
22	31,57	2377	2,377
23	32,95	2430	2,430
24	34,37	2484	2,484
25	36,29	2537	2,537
26	38,30	2590	2,590
27	40,76	2643	2,643
28	43,18	2698	2,698
29	45,85	2751	2,751
30	47,96	2805	2,805
31	50,21	2860	2,860
32	52,55	2914	2,914
33	56,14	2968	2,968



Obrázek 27: Graf vývoje vzdálenosti na čase pro měření 2.Newtonova pohybového zákona na nakloněné rovině

Obrázek 28: Sestava pro měření nakloněné roviny

Závěr a diskuse: Naměřené hodnoty byly zpracovány pomocí polynomické regrese a to pro polynom druhého stupně. Z koeficientů polynomu lze získat hodnotu pro zrychlení a rychlost daného vozíčku. Hodnota zrychlení byla vypočítána na:

$$a = 0,19 \text{ m s}^{-2}$$

Při měření 2. Newtonova pohybového zákona vycházela teoretická hodnota zrychlení přibližně dvakrát větší než hodnota naměřená. V případě nakloněné roviny se tento rozdíl ještě zvyšuje a teoreticky vypočítaná hodnota je přibližně trojnásobkem hodnoty získané polynomickou regresí.

To je způsobeno tou skutečností, že dochází ještě k vyššímu podílu tření mezi podložkou a kolečky daného vozíku, které mají vyšší tendenci setrvávat v dané

poloze. Vozíček je navíc stahován tíhovou silou z nakloněné roviny, který zpomaluje jeho pohyb. Tato síla má opačný směr než síla, která je způsobena tíhovou silou závaží, které vozíček uvádí do pohybu.

Platnost Newtonova zákona byla relativně potvrzena. Pro další měření by bylo vhodnější volit jiný materiál na kolečka vozíku tak, aby neměla tendenci přilnout k danému povrchu. Další možností, jak zvýšit přesnost měření, by byla volba hmotnějších závaží a kladky s menším třením a setrvačností.

Protokol z fyzikálních měření
Elektřina a magnetismus

Jméno:	Andrea Hladíková
Obor:	P–BFY, P–BMAT
Školní rok:	2015/2016
Ročník:	třetí
Naměřeno:	18. 4. 2016
Odevzdáno:	13. 5. 2016
Úloha:	č. 1

Téma: Voltampérová charakteristika pasivních součástek

Úloha: Voltampérová charakteristika

Pomůcky: Arduino, počítač, odpor, diody, spojovací kit, žárovka, propojovací vodiče, potenciometr

Úkol:

a) Určete rovnici pro výpočet proudu procházejícího daným obvodem.

b) Proměřte voltampérové charakteristiky daných součástek.

c) Zakreslete grafy a diskutujte výsledky s předpoklady.

Vlastní měření: Pro měření napětí na stanovených místech v obvodu využijeme následující zdrojový kód:

```
int pin_zdroj = A0;
int zdroj;
int pin_proud = A5;
int proud;
```

```
void setup () {
 pinMode( pin_zdroj, INPUT);
 pinMode( pin_proud, INPUT);
  Serial.begin(9600);
}

void loop() {
   zdroj = analogRead(pin_zdroj);
   proud = analogRead(pin_proud);
    Serial.print(zdroj);
    Serial.print(' ');
    Serial.print(proud);
    Serial.println();
   delay(1000);
}
```

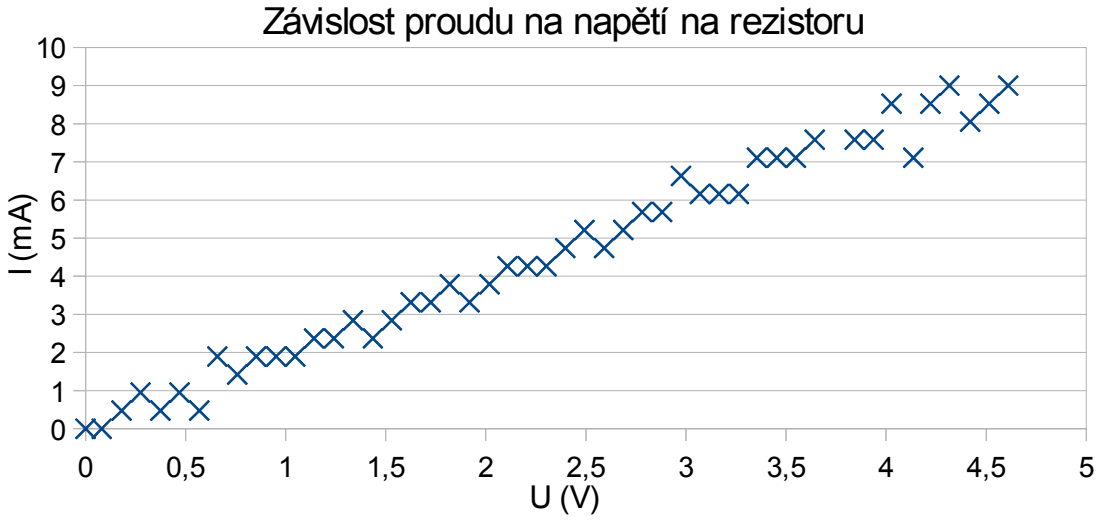
Při měření byla získána následující data:

n	U_1 (V)	U_2 (V)	ΔU (V)	I (mA)	n	U_1 (V)	U_2 (V)	ΔU (V)	I (mA)
1	0,000	0,000	0,000	0,000	25	2,345	2,303	0,043	4,264
2	0,081	0,081	0,000	0,000	26	2,445	2,397	0,047	4,738
3	0,185	0,180	0,005	0,474	27	2,544	2,492	0,052	5,212
4	0,284	0,275	0,009	0,948	28	2,639	2,592	0,047	4,738
5	0,379	0,374	0,005	0,474	29	2,739	2,686	0,052	5,212
6	0,479	0,469	0,009	0,948	30	2,838	2,781	0,057	5,685
7	0,573	0,569	0,005	0,474	31	2,938	2,881	0,057	5,685
8	0,678	0,659	0,019	1,895	32	3,042	2,975	0,066	6,633
9	0,772	0,758	0,014	1,421	33	3,132	3,070	0,062	6,159
10	0,872	0,853	0,019	1,895	34	3,227	3,165	0,062	6,159
11	0,971	0,952	0,019	1,895	35	3,326	3,264	0,062	6,159
12	1,066	1,047	0,019	1,895	36	3,426	3,354	0,071	7,107
13	1,166	1,142	0,024	2,369	37	3,525	3,454	0,071	7,107
14	1,265	1,241	0,024	2,369	38	3,620	3,549	0,071	7,107
15	1,365	1,336	0,028	2,843	39	3,719	3,643	0,076	7,581
16	1,459	1,436	0,024	2,369	40	3,918	3,842	0,076	7,581
17	1,559	1,530	0,028	2,843	41	4,013	3,937	0,076	7,581
18	1,658	1,625	0,033	3,317	42	4,113	4,027	0,085	8,528
19	1,758	1,725	0,033	3,317	43	4,207	4,136	0,071	7,107
20	1,857	1,819	0,038	3,790	44	4,307	4,221	0,085	8,528
21	1,952	1,919	0,033	3,317	45	4,406	4,316	0,090	9,002
22	2,056	2,018	0,038	3,790	46	4,501	4,420	0,081	8,054
23	2,151	2,108	0,043	4,264	47	4,601	4,515	0,085	8,528
24	2,251	2,208	0,043	4,264	48	4,700	4,610	0,090	9,002

Tabulka 13: Naměřené hodnoty pro voltampérovou charakteristiku rezistoru

n	U_1 (V)	U_2 (V)	ΔU (V)	I (mA)	n	U_1 (V)	U_2 (V)	ΔU (V)	I (mA)
1	0,00	0,00	0,000	0,000	26	2,50	2,34	0,15	0,15
2	0,08	0,09	-0,010	-0,010	27	2,60	2,40	0,20	0,20
3	0,19	0,19	0,000	0,000	28	2,70	2,46	0,24	0,24
4	0,29	0,29	-0,005	-0,005	29	2,80	2,52	0,28	0,28
5	0,39	0,39	-0,005	-0,005	30	2,90	2,56	0,33	0,33
6	0,49	0,49	-0,005	-0,005	31	3,00	2,61	0,39	0,39
7	0,59	0,59	0,000	0,000	32	3,10	2,66	0,45	0,45
8	0,69	0,69	0,000	0,000	33	3,20	2,70	0,50	0,50
9	0,79	0,79	0,000	0,000	34	3,30	2,73	0,57	0,57
10	0,89	0,89	0,000	0,000	35	3,40	2,77	0,63	0,63
11	0,99	1,00	-0,005	-0,005	36	3,50	2,81	0,69	0,69
12	1,09	1,09	0,000	0,000	37	3,60	2,84	0,76	0,76
13	1,19	1,19	0,000	0,000	38	3,70	2,87	0,83	0,83
14	1,29	1,29	-0,005	-0,005	39	3,80	2,90	0,90	0,90
15	1,39	1,39	0,005	0,005	40	3,90	2,93	0,97	0,97
16	1,49	1,49	0,005	0,005	41	4,01	2,96	1,05	1,05
17	1,59	1,59	0,005	0,005	42	4,10	2,98	1,12	1,12
18	1,70	1,68	0,015	0,015	43	4,20	3,00	1,19	1,19
19	1,79	1,78	0,015	0,015	44	4,30	3,03	1,27	1,27
20	1,90	1,87	0,024	0,024	45	4,40	3,05	1,35	1,35
21	2,00	1,96	0,034	0,034	46	4,50	3,07	1,43	1,43
22	2,10	2,05	0,053	0,053	47	4,60	3,09	1,51	1,51
23	2,20	2,12	0,073	0,073	48	4,70	3,11	1,59	1,59
24	2,30	2,20	0,097	0,097	49	4,80	3,13	1,67	1,67
25	2,40	2,27	0,131	0,131	50	4,90	3,15	1,75	1,75

Tabulka 14: Naměřené hodnoty pro voltampérovou charakteristiku Zenerovy diody v závěrném směru



Obrázek 29: Graf závislosti proudu na napětí na rezistoru

n	U_1 (V)	U_2 (V)	ΔU (V)	I (mA)	n	U_1 (V)	U_2 (V)	ΔU (V)	I (mA)
1	0,00	0,00	0,00	0,00	26	2,50	2,47	0,03	0,03
2	0,09	0,09	0,00	0,00	27	2,60	2,52	0,08	0,08
3	0,18	0,19	0,00	0,00	28	2,70	2,55	0,15	0,15
4	0,29	0,29	0,00	0,00	29	2,80	2,57	0,23	0,23
5	0,39	0,39	0,00	0,00	30	2,90	2,58	0,31	0,31
6	0,49	0,49	0,00	0,00	31	3,00	2,59	0,41	0,41
7	0,59	0,59	0,00	0,00	32	3,10	2,60	0,49	0,49
8	0,69	0,69	0,00	0,00	33	3,20	2,62	0,58	0,58
9	0,79	0,79	0,00	0,00	34	3,30	2,63	0,67	0,67
10	0,89	0,89	0,00	0,00	35	3,40	2,64	0,77	0,77
11	1,00	0,99	0,00	0,00	36	3,50	2,65	0,85	0,85
12	1,09	1,09	0,00	0,00	37	3,60	2,66	0,94	0,94
13	1,19	1,20	-0,01	-0,01	38	3,70	2,66	1,04	1,04
14	1,29	1,30	0,00	0,00	39	3,81	2,67	1,14	1,14
15	1,39	1,39	0,00	0,00	40	3,90	2,68	1,22	1,22
16	1,49	1,49	0,00	0,00	41	4,00	2,68	1,32	1,32
17	1,59	1,59	0,00	0,00	42	4,10	2,69	1,41	1,41
18	1,70	1,69	0,00	0,00	43	4,20	2,69	1,51	1,51
19	1,80	1,80	0,00	0,00	44	4,31	2,70	1,61	1,61
20	1,90	1,89	0,00	0,00	45	4,40	2,71	1,69	1,69
21	2,00	1,99	0,00	0,00	46	4,50	2,71	1,79	1,79
22	2,10	2,10	0,00	0,00	47	4,60	2,72	1,88	1,88
23	2,20	2,20	0,00	0,00	48	4,71	2,72	1,99	1,99
24	2,30	2,30	0,00	0,00	49	4,80	2,73	2,07	2,07
25	2,40	2,40	0,00	0,00	50	4,90	2,73	2,17	2,17

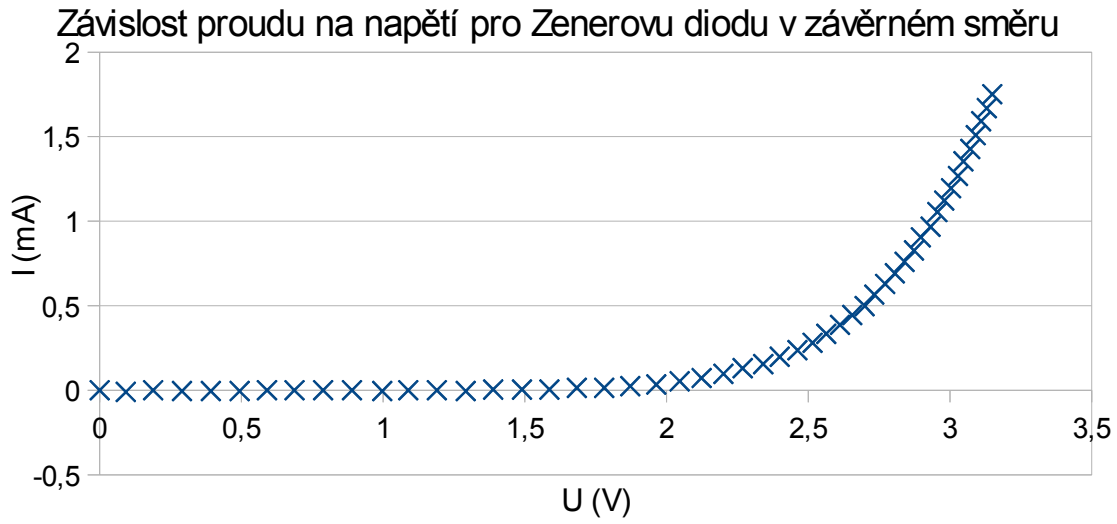
Tabulka 15: Naměřené hodnoty pro voltampérovou charakteristiku LED diody v propustném směru

n	U_1 (V)	U_2 (V)	ΔU (V)	I (mA)	n	U_1 (V)	U_2 (V)	ΔU (V)	I (mA)
1	0,00	0,00	0,00	0,00	26	2,62	0,65	1,97	1,97
2	0,08	0,08	0,00	0,00	27	2,72	0,65	2,07	2,07
3	0,19	0,20	0,00	0,00	28	2,83	0,65	2,17	2,17
4	0,30	0,29	0,01	0,01	29	2,93	0,65	2,28	2,28
5	0,39	0,38	0,01	0,01	30	3,03	0,65	2,38	2,38
6	0,49	0,45	0,04	0,04	31	3,13	0,66	2,47	2,47
7	0,60	0,50	0,10	0,10	32	3,23	0,67	2,56	2,56
8	0,69	0,53	0,17	0,17	33	3,33	0,66	2,67	2,67
9	0,80	0,54	0,26	0,26	34	3,43	0,66	2,77	2,77
10	0,90	0,56	0,34	0,34	35	3,54	0,67	2,87	2,87
11	1,01	0,57	0,43	0,43	36	3,64	0,67	2,97	2,97
12	1,10	0,58	0,52	0,52	37	3,74	0,67	3,07	3,07
13	1,21	0,59	0,62	0,62	38	3,84	0,67	3,17	3,17
14	1,30	0,60	0,71	0,71	39	3,94	0,67	3,26	3,26
15	1,41	0,60	0,81	0,81	40	4,04	0,67	3,37	3,37
16	1,51	0,62	0,89	0,89	41	4,13	0,68	3,46	3,46
17	1,61	0,62	0,99	0,99	42	4,25	0,68	3,57	3,57
18	1,71	0,62	1,09	1,09	43	4,35	0,68	3,66	3,66
19	1,81	0,63	1,19	1,19	44	4,44	0,69	3,75	3,75
20	1,92	0,63	1,29	1,29	45	4,55	0,68	3,87	3,87
21	2,02	0,63	1,40	1,40	46	4,64	0,68	3,96	3,96
22	2,12	0,64	1,48	1,48	47	4,75	0,69	4,06	4,06
23	2,22	0,64	1,58	1,58	48	4,85	0,69	4,16	4,16
24	2,32	0,64	1,68	1,68	49	4,95	0,69	4,26	4,26
25	2,42	0,65	1,78	1,78	50	5,00	0,69	4,31	4,31

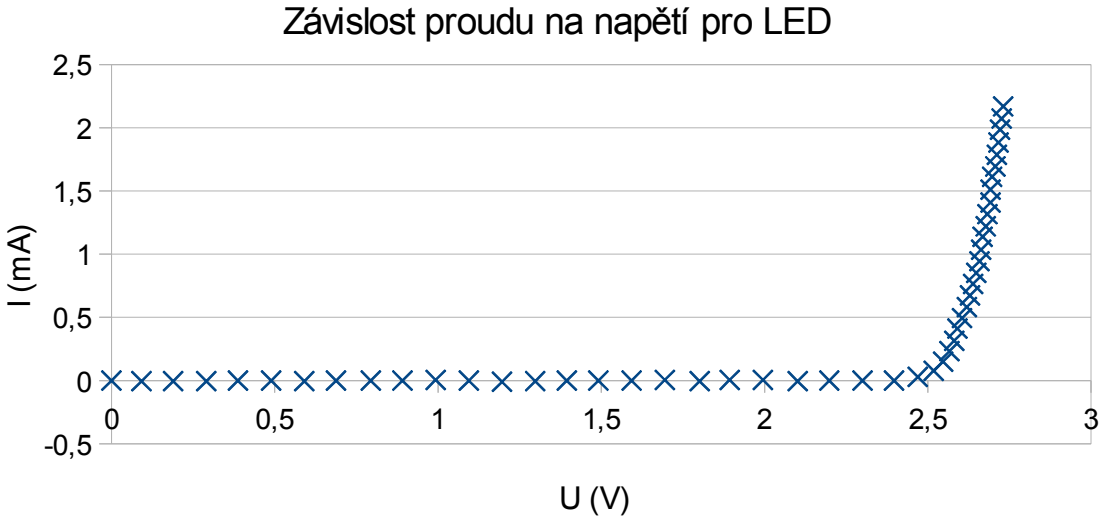
Tabulka 16: Naměřené hodnoty pro voltampérovou charakteristiku diody v propustném směru

n	U_1 (V)	U_2 (V)	ΔU (V)	I (mA)	n	U_1 (V)	U_2 (V)	ΔU (V)	I (mA)
1	0,00	0,00	0,00	0,00	25	2,10	1,01	1,09	5,45
2	0,08	0,01	0,07	0,33	26	2,19	1,07	1,12	5,60
3	0,17	0,03	0,14	0,68	27	2,27	1,13	1,15	5,73
4	0,25	0,05	0,20	1,01	28	2,36	1,19	1,17	5,84
5	0,34	0,07	0,28	1,38	29	2,45	1,25	1,20	5,99
6	0,43	0,09	0,34	1,68	30	2,54	1,31	1,23	6,15
7	0,52	0,12	0,40	2,00	31	2,63	1,37	1,26	6,28
8	0,60	0,15	0,46	2,28	32	2,71	1,44	1,27	6,36
9	0,69	0,17	0,52	2,60	33	2,80	1,50	1,31	6,54
10	0,78	0,20	0,58	2,92	34	2,89	1,57	1,32	6,60
11	0,87	0,23	0,64	3,18	35	2,97	1,63	1,35	6,73
12	0,95	0,27	0,68	3,41	36	3,06	1,69	1,37	6,84
13	1,04	0,31	0,73	3,64	37	3,15	1,76	1,38	6,92
14	1,13	0,36	0,77	3,86	38	3,23	1,82	1,42	7,08
15	1,22	0,41	0,81	4,04	39	3,32	1,89	1,44	7,18
16	1,30	0,47	0,83	4,17	40	3,41	1,95	1,46	7,31
17	1,41	0,53	0,88	4,39	41	3,51	2,01	1,50	7,48
18	1,48	0,59	0,90	4,48	42	3,59	2,08	1,51	7,55
19	1,66	0,69	0,97	4,86	43	3,68	2,14	1,53	7,66
20	1,73	0,75	0,98	4,91	44	3,76	2,21	1,55	7,76
21	1,83	0,81	1,03	5,13	45	3,85	2,28	1,57	7,87
22	1,83	0,83	1,01	5,03	46	3,94	2,35	1,59	7,95
23	1,92	0,89	1,04	5,18	47	4,03	2,40	1,62	8,11
24	2,01	0,95	1,06	5,31	48	4,11	2,47	1,64	8,22
					49	4,20	2,54	1,66	8,32

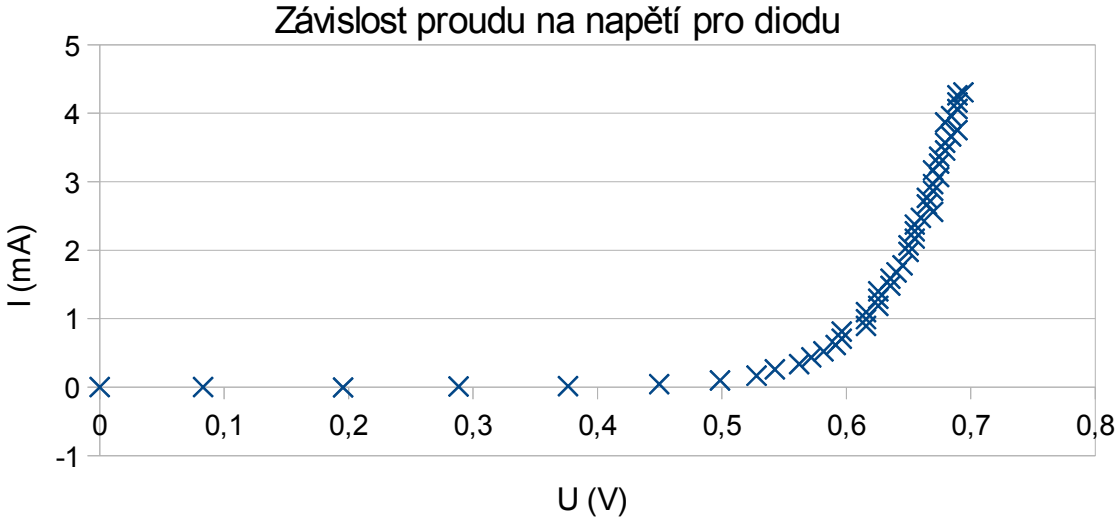
Tabulka 17: Naměřené hodnoty voltampérová charakteristika Žárovka



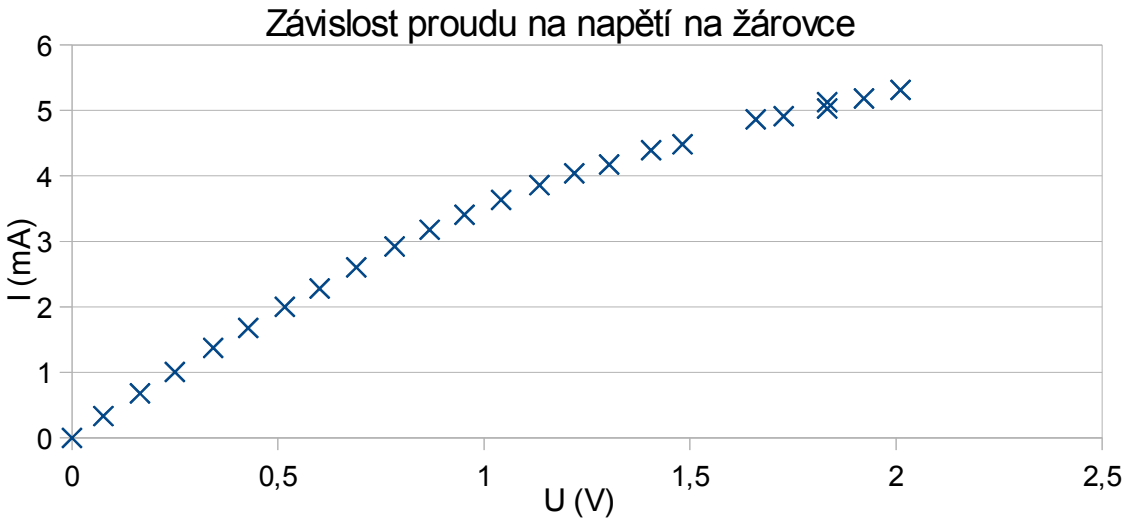
Obrázek 30: Graf závislosti proudu na napětí pro Zenerovu diodu v závěrném směru



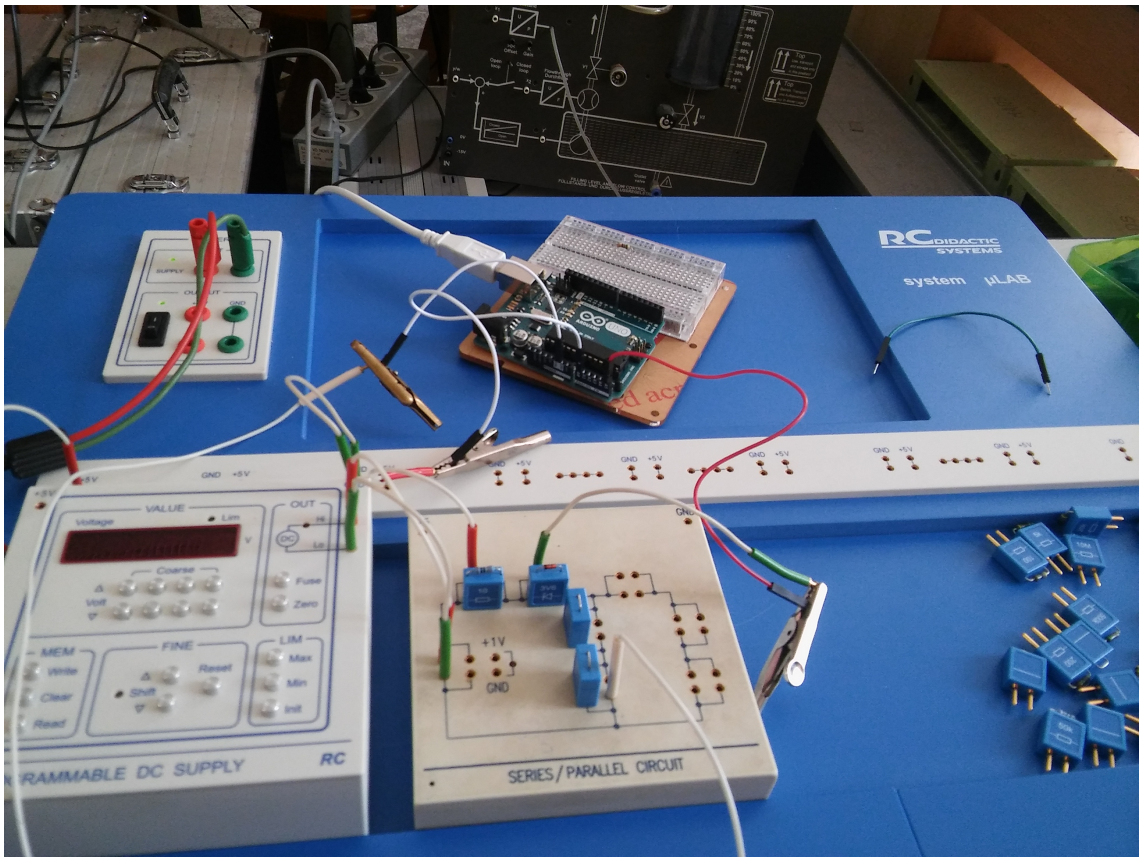
Obrázek 31: Graf závislosti proudu na napětí pro LED diodu v propustném směru



Obrázek 32: Graf závislosti proudu na napětí pro diodu v propustném směru



Obrázek 33: Graf závislosti proudu na napětí na žárovce



Obrázek 34: Sestava pro měření voltampérové charakteristiky

Závěr a diskuse: Naměřené voltampérové charakteristiky jednotlivých součástech souhlasí s předpoklady. Průběh proudu na rezistoru v závislosti na napětí je lineární, i když měření vykazuje značné skoky, to může být zapříčiněno nevhodnou volbou referenčního odporu, pomocí něhož byla hodnota proudu měřena. Pokud bychom data proložili regresní přímkou, její předpis by odpovídal lineární závislosti.

Polovodičové součástky (diody) po dosažení prahového/Zenerova napětí začínají vést proud, který roste téměř exponenciálně, a to hlavně
u Zenerovy diody. Na posledním grafu je znázorněn průběh proudu a napětí u žárovky. Jedná se o nelineární součástku, kde zprvu je závislost proudu na napětí téměř lineární. Následně vlivem zahřívání vlákna se zvyšuje odpor žárovky a tím se snižuje hodnota protékajícího proudu. Z voltampérové charakteristiky by bylo možné pomocí proložení dat regresní přímkou zjistit vývoj odporu dané žárovky.

Pro měření voltampérové charakteristiky byl využit regulovatelný zdroj s rozsahem 0 až 5 V. Větší rozsah není možné vzhledem k možnostem platformy využít. Naměřené charakteristiky byly měřeny jako diskrétní hodnoty pro nastavenou hodnotu napětí na zdroji. Měření by bylo možné provést s kontinuální změnou napětí na zdroji, například sestavením vlastního zdroje. Je možné vyrobit zdroj z baterie a potenciometru, za pomoci je zdroj regulován. Samozřejmě by bylo možné jako zdroj využít výstup s PWM, spojený s klopným obvodem a kondenzátorem.

Protokol z fyzikálních měření Kmity, Vlny, Optika

Jméno: Andrea Hladíková

Obor: P–BFY, P–BMAT

Školní rok: 2015/2016

Ročník: třetí

Naměřeno: 18. 4. 2016

Odevzdáno: 13. 5. 2016

Úloha: č. 1

Téma: Fyzické a matematické kyvadlo

Úloha: Určení tíhového zrychlení za pomoci matematického kyvadla

Pomůcky: Arduino, počítač, optická závora, matematické kyvadlo

Ukol:

a) Odvoďte rovnici pro matematické kyvadlo.

b) Proměřte periodu kmitu matematického kyvadla.

c) Určete tíhové zrychlení.

Vlastní měření: Rovnice matematického kyvadla:

$$F = -F_G sin\alpha \tag{25}$$

Pro malé úhly lze sinus přepsat následovně:

$$F = -F_G \frac{y}{l} \tag{26}$$

kde F_G je tíhová síla, která působí na hmotný bod, y je velikost výchylky l je délka závěsu viz obrázek 14.

Srovnáním s rovnicí pro harmonický oscilátor dostáváme:

$$-mg\frac{y}{l} = m\omega^2 y \tag{27}$$

Kde úhlová rychlost ω lze vyjádřit ve tvaru $\omega = 2\Pi f$, kde frekvence f je rovna $f = \frac{1}{T}$, kde T je perioda.

Z rovnice (27) lze snadno určit úhlovou rychlost:

$$\omega = \sqrt{\frac{g}{l}} \tag{28}$$

Z toho lze snadno určit periodu, kterou jsme schopni změřit:

$$T = 2\Pi\sqrt{\frac{l}{g}} \tag{29}$$

Pro možnost určení tíhového zrychlení je tedy nutné naměřit periodu daného matematického kyvadla [30].

Pro měření pomocí optické závory využijeme předcházející zdrojový kód.

```
int pin_napeti = A0; //deklarace užívaného pinu
int napeti;   //deklraca proměnné
long t;

void setup () {
  pinMode(pin_napeti, INPUT);
   Serial.begin(9600);

}

void loop() {
  napeti=analogRead(pin_napeti);
  t=millis();

  if (napeti >= 750) {
   Serial.print(napeti);
   Serial.print(' ');
   Serial.print(t);
   Serial.println();
```

```
  }

  delay(10);
}
```

Před vlastním měřením je nutné vyzkoušet, zda hledáme minimální či maximální hodnotu napětí a dle toho upravit následující zdrojový kód. Je samozřejmě možné využít i jiné způsoby zapojení, programu.

Při měření tíhového zrychlení byly získány hodnoty, které jsou v tabulce.

Závěr a diskuse: Pomocí matematického kyvadla bylo proměřeno tíhové zrychlení. Naměřené hodnoty odpovídají očekávaným výsledkům, které se blíží tabulkovým hodnotám.

$$g = (9,70 \pm 0,04)\ \mathrm{ms}^{-2}$$

Chyba měření je vzhledem k průběhu měření a množství možností, jak způsobit hrubou chybu, přijatelná. Pro měření je vhodné zvolit vyšší vzorkovací frekvenci, aby optická závora s jistotou zaznamenala prolétávající objekt. Je vhodné, aby byl signál přerušen při průchodu tělesa rovnovážnou polohou několikrát. Z těchto hodnot se dá vysledovat průběh průchodu tělesa, a ustanovit tak okamžik, kdy těleso procházelo rovnovážnou polohou. Při měření je nutné dát si pozor na zvolenou vzorkovací frekvenci. Při volbě příliš malé vzorkovací frekvence, může dojít k tomu, že by matematické kyvadlo při průchodu optickou závorou nebylo zaznamenáno. Proto je vhodné volit vyšší vzorkovací frekvenci, a to tak aby jeden průchod matematického kyvadla optickou závorou byl zaznamenán několikrát. Z toho lze přibližně stanovit, kdy prošlo kyvadlo těžištěm a zvýšit tak přesnost měření.

V případě mnou naměřených hodnot je možné vysledovat, že optická závora nebyla zcela v místě těžiště. To je možné vidět na tom, že kyv do jedné ze stran probíhá o několik milisekund déle. Proto je následně vhodné nepočítat tíhové zrychlení pouze s půl periodami, ale periodami celými. V ideálním případě také vyhodnocovat obě možnosti. Měření tíhového zrychlení bylo přesné.

Tabulka 18: Naměřené hodnoty pro tíhové zrychlení

n	t (ms)	t (s)	T (s)	g (m/s^2)
1	2681,750	2,682		
2	4703,750	4,704	2,022	9,752
3	6746,000	6,746	2,042	9,560
4	8776,900	8,777	2,031	9,667
5	10799,500	10,800	2,023	9,746
6	12829,091	12,829	2,030	9,679
7	14853,364	14,853	2,024	9,730
8	16878,182	16,878	2,025	9,725
9	18905,333	18,905	2,027	9,702
10	20934,091	20,934	2,029	9,687
11	22961,000	22,961	2,027	9,705
12	24984,167	24,984	2,023	9,741
13	27013,385	27,013	2,029	9,683
14	29048,778	29,049	2,035	9,624
15	31068,000	31,068	2,019	9,779
16	33097,154	33,097	2,029	9,683
17	35118,429	35,118	2,021	9,759
18	37145,786	37,146	2,027	9,701
19	39168,267	39,168	2,022	9,747
20	41195,933	41,196	2,028	9,698
21	43222,615	43,223	2,027	9,707
22	45251,125	45,251	2,029	9,689
23	47281,875	47,282	2,031	9,668
24	49308,556	49,309	2,027	9,707
25	51335,625	51,336	2,027	9,703
26	53366,375	53,366	2,031	9,668
27	55389,000	55,389	2,023	9,746
28	57418,000	57,418	2,029	9,685
29	59442,273	59,442	2,024	9,730
30	61470,625	61,471	2,028	9,691
31	63496,286	63,496	2,026	9,717
32	65523,200	65,523	2,027	9,705
33	67551,000	67,551	2,028	9,696
34	69578,750	69,579	2,028	9,697
35	71610,375	71,610	2,032	9,660
36	73638,000	73,638	2,028	9,698
37	75658,083	75,658	2,020	9,771
38	77689,385	77,689	2,031	9,663
39	79709,833	79,710	2,020	9,767
40	81739,000	81,739	2,029	9,683
41	83765,750	83,766	2,027	9,706
			průměr	**9,7031**
			abs. chyba	**0,0412**
			rel. chyba	**4,12%**

Protokol z fyzikálních měření Kmity, Vlny, Optika

Jméno:	Andrea Hladíková
Obor:	P–BFY, P–BMAT
Školní rok:	2015/2016
Ročník:	třetí
Naměřeno:	18. 4. 2016
Odevzdáno:	13. 5. 2016
Úloha:	č.2

Téma: Matematické a fyzické kyvadlo

Úloha: Určete dekrement útlumu fyzického kyvadla

Pomůcky: Fyzické kyvadlo, potenciometr, arduino, počítač, propojovací vodiče

Ukol:

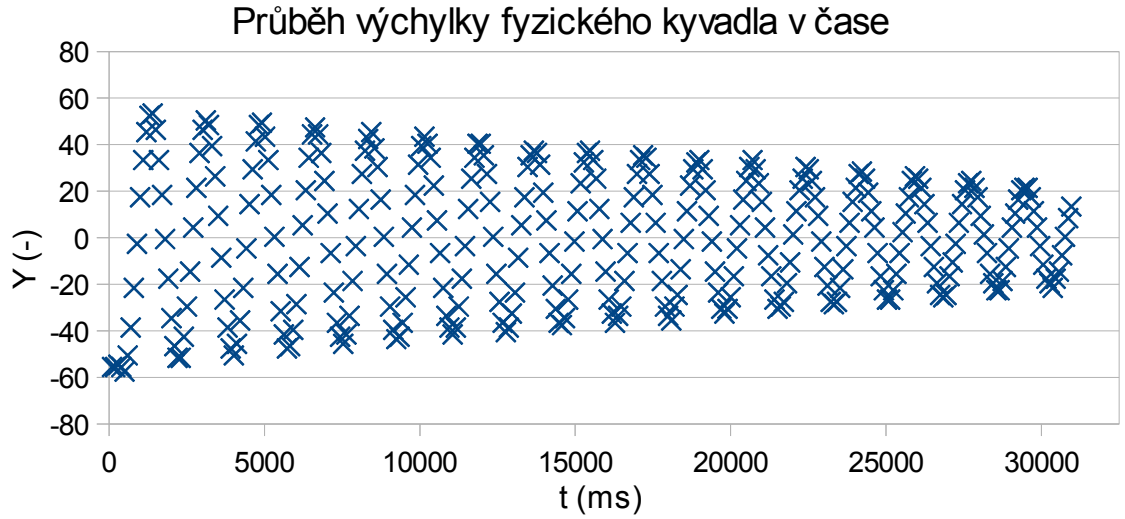
a) Zaznamenejte průběh pohybu fyzického kyvadla.

b) Vypočítejte dekrement útlumu fyzického kyvadla, diskutujte vznik chyb.

Vlastní měření: Pro měření vzdálenosti využijeme následující zdrojový kód:

```
int pin_napeti = A0; //deklarace užívaného pinu
int napeti;   //deklraca proměnné
long t;

void setup () {
  pinMode(pin_napeti, INPUT);
  Serial.begin(9600);
  }

void loop() {
  napeti=analogRead(pin_napeti);
  t=millis();
   Serial.print(napeti);
   Serial.print(' ');
   Serial.print(t);
   Serial.println();

  delay(100);
}
```

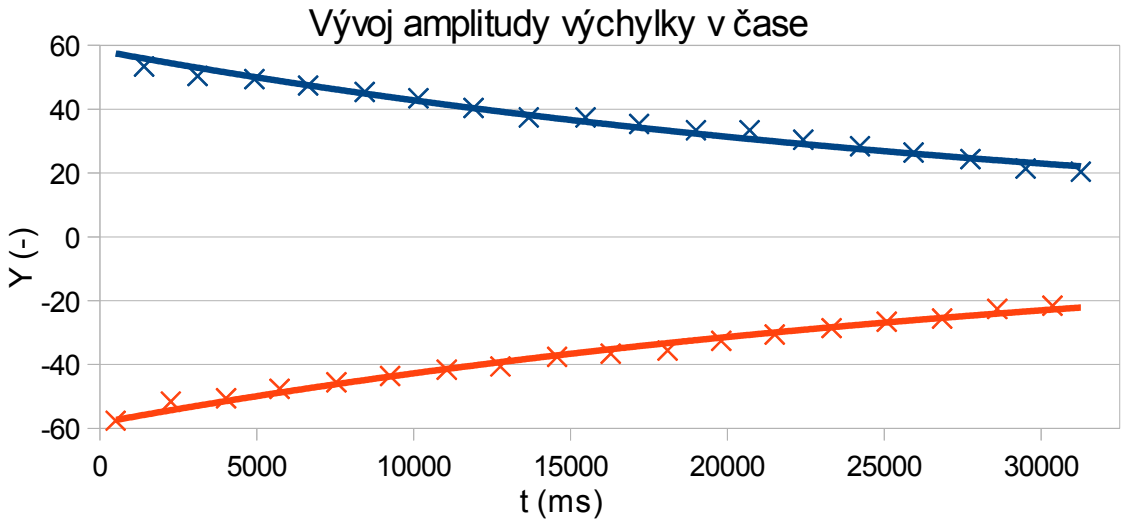
Za pomoci předchozího programu byl nasnímán pohyb fyzického kyvadla a hodnoty byly vyneseny do grafu viz graf 35.



Obrázek 35: Graf průběhu výchylky fyzického kyvadla v čase

Měření průběhu výchylky fyzického kyvadla v čase probíhalo pomocí změny odporu potenciometru, a tím vyvolané změny napětí. Hodnota výchylky je u

grafu bezrozměrná a to z toho důvodu, že hodnota je nechaná ve tvaru, které měřilo Arduino, pouze jsou posunuty k nulové hodnotě.



Obrázek 36: Graf změny amplitudy fyzického kyvadla v čase

Z naměřených hodnot byly vyfiltrovány extrémy, které byly zaneseny do grafu a proloženy regresními rovnicemi. Rovnice regrese pro maxima má tvar:

$$A(t) = 58,36e^{-3,11\cdot10^{-5}t} \tag{30}$$

Pro hodnoty minim je regrese tvaru:

$$A(t) = -58,26e^{-3,09\cdot10^{-5}t} \tag{31}$$

Obrázek 37: Fyzické kyvadlo v průběhu měření

Závěr a diskuse: Naměřená data byla proložena regresními přímkami a z nich byly získány následující hodnoty dekrementu útlumu.

$$b_1 = 3,11 \cdot 10^{-5}\ \mathrm{t}^{-1}$$

a pro druhou rovnici

$$b_1 = 3,09 * 10^{-5}\ \mathrm{t}^{-1}$$

Z hodnoty velikosti dekrementu útlumu je zjevné, že útlum probíhá pozvolně a jeho průběh se blíží lineárnímu poklesu. Tato skutečnost je viditelná i na datech, která jsou vynesena do příslušných grafů. Tento tvar tlumení kyvadla mohl nastat nevhodnou konstrukcí jak vlastního kyvadla, tak měřící soustavy. Měření velikosti výchylky daného kyvadla probíhalo za pomoci potenciometru, se kterým bylo kyvadlo spojeno a svým pohybem měnilo velikost jeho odporu. Vzhledem ke skutečnosti, jak je potenciometr konstruován, dochází v něm k vnitřnímu tření, které může zapříčinit změnu tvaru útlumu kyvadla. Další možností je nevhodná konstrukce kyvadla. Vzhledem ke způsobu měření by se dala chyba zmenšit, pokud by bylo použito kyvadlo s větší setrvačností, a to jak například vhodnějším tvarem kyvadla, tak i jeho hmotností.

Protokol z fyzikálních měření Kmity, Vlny, Optika

Jméno:	Andrea Hladíková
Obor:	P–BFY, P–BMAT
Školní rok:	2015/2016
Ročník:	třetí
Naměřeno:	18. 4. 2016
Odevzdáno:	13. 5. 2016
Úloha:	č. 3

Téma: Ohnisková vzdálenost

Úloha: Určete ohniskovou vzdálenost čočky

Pomůcky: Arduino, počítač, fotorezistor, fotodioda, měřítko, spojná čočka

Ukol:

a) Změřte ohniskovou vzdálenost čočky.

Vlastní měření: Pro měření vzdálenosti využijeme následující zdrojový kód:

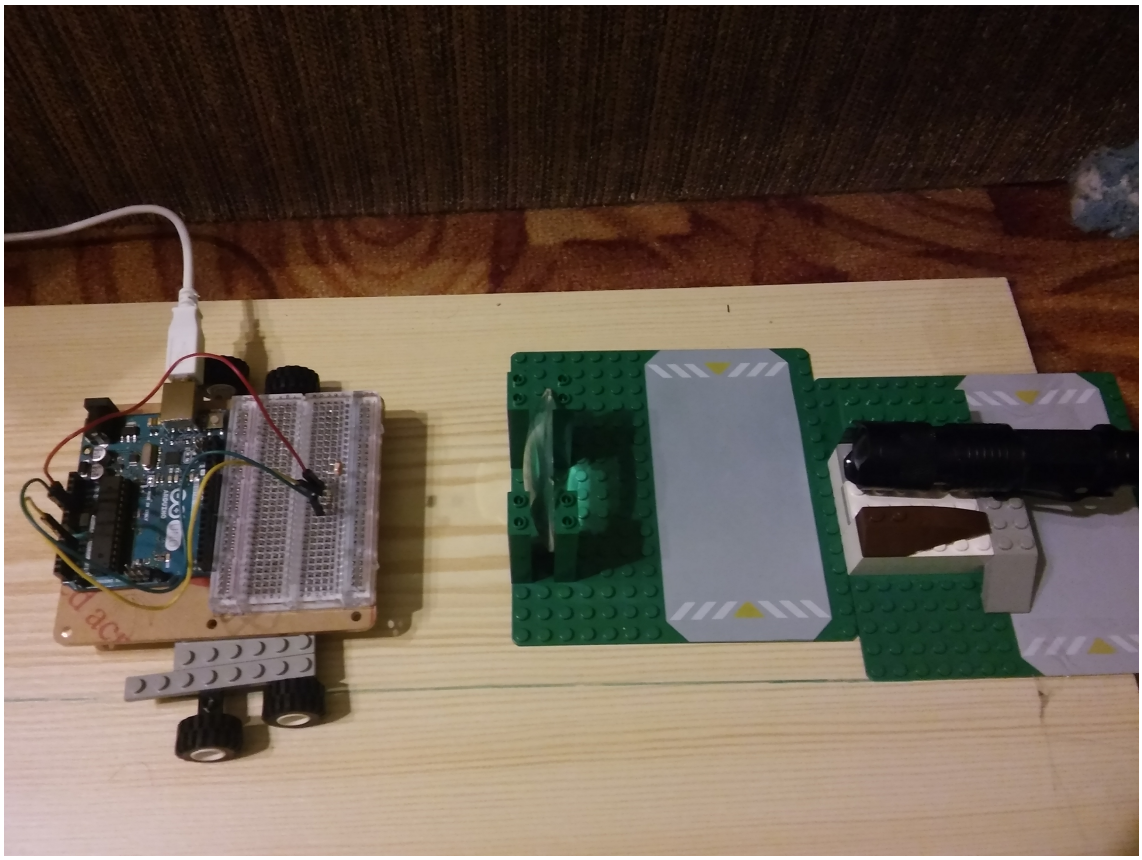
```
int pin_napeti = A0; //deklarace užívaného pinu
int napeti;   //deklraca proměnné
int i;

void setup () {
  pinMode(pin_napeti, INPUT);
  Serial.begin(9600);
  i=0;
}

void loop() {
  napeti=analogRead(pin_napeti);
```

```
  i++;

   Serial.print(napeti);
   Serial.print(' ');
   Serial.print(i);
   Serial.println();
  delay(2000);
}
```

Zdrojový kód pro měření ohniskové vzdálenosti je vytvořen nejjednodušším způsobem. Arduino vypisuje hodnotu napětí na fotodiodě a nebo fotorezistoru a vypisuje jí. To se provádí každé 2 sekundy, aby se dalo přesunout měřící zařízení do další polohy. Tento kód není rozšířen o sledování polohy, je tedy nutné mít optickou lavici. Pokud není možné měřit na optické lavici, je nutné si vytvořit vlastní sestavu. Do jedné přímky sestavíme jednotlivé pomůcky pro měření. Jako osu můžeme použít měřítko, na které prvky seskládáme. Arduino má v programu nastaven posun o 1 cm.

Tabulka 19: Ohnisková vzdálenost

n	U (V)	d (cm)	U (V)	d (cm)	U (V)	d (cm)	U (V)	d (cm)	U (V)	d (cm)
1	749	1	765	1	763	1	748	1	740	1
2	749	2	764	2	763	2	748	2	740	2
3	749	3	764	3	762	3	748	3	740	3
4	744	4	769	4	770	4	753	4	746	4
5	752	5	754	5	774	5	760	5	755	5
6	740	6	777	6	771	6	772	6	748	6
7	731	7	704	7	790	7	780	7	752	7
8	769	8	645	8	796	8	777	8	741	8
9	780	9	814	9	799	9	796	9	781	9
10	767	10	814	10	802	10	804	10	774	10
11	778	11	817	11	795	11	804	11	755	11
12	744	12	819	12	791	12	806	12	801	12
13	754	13	817	13	786	13	809	13	810	13
14	744	14	810	14	774	14	806	14	805	14
15	810	15	788	15	806	15	799	15	791	15
16	808	16	814	16	805	16	804	16	794	16
17	803	17	810	17	805	17	803	17	799	17
18	803	18	807	18	804	18	803	18	798	18
19	796	19	804	19	802	19	801	19	798	19
20	799	20	801	20	800	20	800	20	795	20
21	794	21	799	21	796	21	798	21	791	21
22	789	22	795	22	798	22	792	22	789	22
23	790	23	789	23	794	23	790	23	783	23
24	406	24	785	24	792	24	789	24	780	24
25	443	25	780	25	790	25	787	25	778	25
26	426	26	777	26	786	26	784	26	776	26
27	780	27	773	27	785	27	782	27	775	27
28	778	28	771	28	784	28	779	28	772	28
29	774	29	767	29	780	29	775	29		
30	771	30			780	30	773	30		
31	768	31			777	31	773	31		
32					744	32	774	32		
33					734	33				



Obrázek 38: Sestava pro měření ohniskové vzdálenosti

Závěr a diskuse: Pomocí přímé metody byla naměřena ohnisková vzdálenost:

$$f = (13 \pm 1)cm$$

Relativní chyba měření je 8%. Tato chyba je vzhledem k okolnostem měření přijatelná. Měření probíhalo pomocí optické lavice a měřítka. Přesnost by mohla být zvýšena ve chvíli, kdy by součástí měření bylo i přímé měření vzdálenosti. Pro měření vzdálenosti by se dalo užít stejné sestavy jako v prvním měření mechaniky. Pro měření lze využít fotorezistor a fotodiodu, je ale nutné zkontrolovat, jaká je svítivost zdroje a jaké vlastnosti má daná součástka.

Čočka, která byla měřena, byla využita ve fyzikálním praktiku.

Pro porovnání hodnota naměřená při fyzikálním praktiku byla:

$$f = (12.6 \pm 0,1)cm$$

Tato hodnota slouží pouze pro kontrolu měření za pomoci platformy Arduino. Tato hodnota byla naměřena pomocí Besselovy metody. Besselova metoda určuje ohniskovou vzdálenost za pomoci nalezení ostrého obrazu předmětu. Pokud na optickou lavici umístíme zdroj světla, stínítko, objekt a čočku. Pak je možné nalezt právě dvě polohy, kdy se na stínítku zobrazí ostrý obraz. V jednom případě se bude jednat o obraz zvětšený a v druhém případě o obraz zmenšený. Pro oba obrazy musí objekt a zdroj stát na stejném místě, hledání obrazu tedy probíhá pouze za pomoci změny polohy spojné čočky. Následně za pomoci známé polohy čočky lze ze zobrazovací rovnice dopočítat ohniskovou vzdálenost. Při školních praktikách mi toto měření vycházelo jako nejpřesnější, z toho důvodu ho zde využívám jako referenční. Ohniskovou vzdálenost je možné měřit i dalšími způsoby.

Dle porovnání výsledků lze konstatovat, že je možné tuto metodu pro měření čočky využít.