

29. ARDUINO VE FYZIKALNÍ LABORATOŘI

Úvod

1 Základní popis hw a sw

- 1.1 Základní popis hardware Arduina a vstupně výstupních pinů (tzv. pin-out)
- 1.2 Základní popis programátorského prostředí Arduino IDE
- 1.3 Popis měřicí sady s Arduinem

2 Pracovní úkol 1 – ukázkové příklady práce a Arduinem

- 2.1 Programování mikroprocesorové desky
- 2.2 Ukázkový program - Použití digitálních výstupů – Blikání vestavěné LED diody.
 - 2.2.1 Popis programu
 - 2.2.2 Postup provedení – program Blink
- 2.3 Ukázkový program - Použití analogových vstupů a Sériového monitoru
 - 2.3.1 Sériová komunikace
 - 2.3.2 Analogové vstupy
 - 2.3.3 Postup provedení – program ReadAnalogVoltage

3 Zapojování obvodů s pomocí nepájivé kontaktní desky

4 Pracovní úkol 2 – měření voltampérových charakteristik LED

- 4.1 Popis měření
- 4.2 Postup provedení – Voltampérová charakteristika LED

5 Pracovní úkol 3 – měření časové závislosti vybíjení a nabíjení kondenzátoru v sériovém RC obvodu

- 5.1 Popis měření
- 5.2 Program RealTerm – monitor sériové linky
- 5.3 Postup provedení - měření časové závislosti vybíjení a nabíjení kondenzátoru v sériovém RC obvodu

Závěr

Úvod

Laboratorní úloha „Arduino ve fyzice“ představuje novou generaci školních laboratorních úloh, kde je kladen důraz nejenom na vlastní fyzikální měření, ale též na **digitalizaci, záznam a přenos naměřených dat**. Kolem nás jsou nejenom klasické počítače, ale též nově mikropočítače a **jednodeskové počítače** (např. Arduino aj.), které mají zabudované vstupní i výstupní analogové digitální a digitálně analogové převodníky a dávají prostor pro využití Arduina jako měřicího a řídicího nástroje ve fyzikální laboratoři.

Arduino je open-source elektronická platforma založená na cenově dostupném a snadno použitelném hardwaru a softwaru [1]. Bylo vyvinuto jako snadný nástroj pro prototypování, zaměřený na uživatele bez znalostí elektroniky a programování. Mikroprocesorové desky Arduino jsou schopny číst vstupy, zpracovávat je a posílat výsledný signál na vybrané výstupy, případně vysílat naměřená a zpracovaná data na datové rozhraní (COM). K programování desek lze použít volně dostupný software Arduino IDE [1]. Dostupnost Arduina, jednoduchost jeho programování a existence velkého množství připojitelných senzorů vedlo k masovému používání amatérskými uživateli, studenty a učiteli ve školách, ale i profesionály k vytváření nepřehledného množství aplikací.

V úloze se nejdříve seznámíme s Arduinem a jeho základním programátorským prostředím Arduino IDE. Vyzkoušíme si první základní ukázkové programy, které nás uvedou do zapojování Arduina a do jeho programátorské obsluhy. Otestujeme měření a záznam dat Arduinem přes sériový port COM. Data ze sériového portu COM budeme snímat přímo pomocí vestavěné aplikace Sériový Monitor nebo Sériový Plotter v Arduino IDE nebo programovým terminálem *RealTerm*.

Dále se též seznámíme s prototypováním vlastních zapojení na nepájivém poli. Sestavíme jednoduchý obvod pro měření voltampérových charakteristik vybraných diod a proměříme časový průběh nabíjení a vybíjení kondenzátoru v sériovém RC obvodu.

1 Základní popis hw a sw

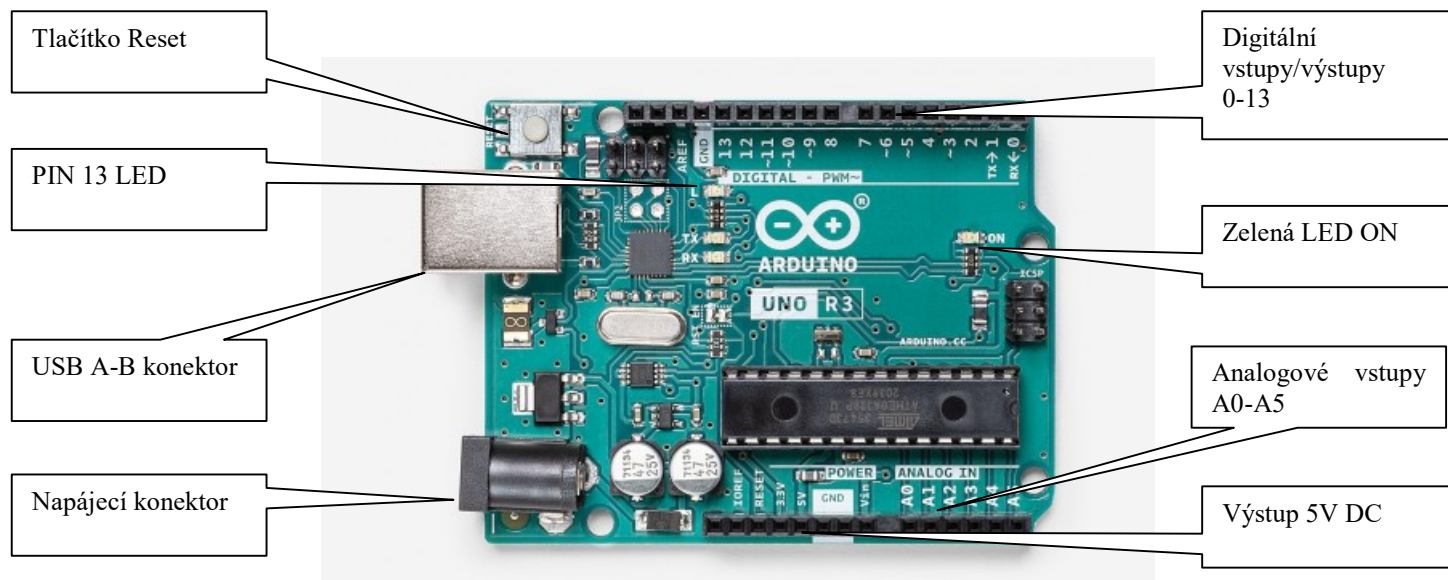
1.1 Základní popis hardware Arduina a vstupně výstupních pinů (tzv. pin-out)

Originální stránky tvůrců Arduina jsou na <https://arduino.cc>. V úloze budeme používat nejrozšířenější mikroprocesorovou desku Arduino UNO. Arduino UNO je deska založená na procesoru ATmega328P. Na obr.1 je deska Arduino UNO s základním popisem.

Deska Arduino UNO má 14 digitálních vstupních/výstupních pinů (označeny 0-13), z nichž 6 lze použít jako výstupy PWM (Pulse WidthModulation, označeny vlnovkou), 6 analogových vstupů (označeny A0-A5), keramický rezonátor 16 MHz (CSTCE16M0V53-R0), připojení USB, napájecí konektor, resetovací tlačítko, výstupy stabilizovaných napětí 5VDC a 3,3VDC. Obsahuje všechnu elektroniku potřebnou pro podporu mikrokontroléru. Deska se připojuje k PC přes USB rozhraní standardním USB A-B kabelem (přes USB rozhraní je i napájena), desku je možné napájet i pomocí AC-DC adaptéru (7-12VDC). USB připojení k PC je nutné hlavně pro komunikaci s deskou při jejím programování.

Pozn.: Arduino UNO pracuje SAMOSTATNĚ bez nutnosti připojení k počítači. Po zapojení obvodu (fyzikálního experimentu), vytvoření programu a jeho nahrání do desky Arduina se

automaticky spustí nahraný program, který běží autonomně ve smyčce. Sestavená úloha s naprogramovaným Arduinem se automaticky spustí po připojení napájecího zdroje 7-12V. (Sestavenou úlohu s naprogramovaným Arduinem je možné napájet USB kabelem).



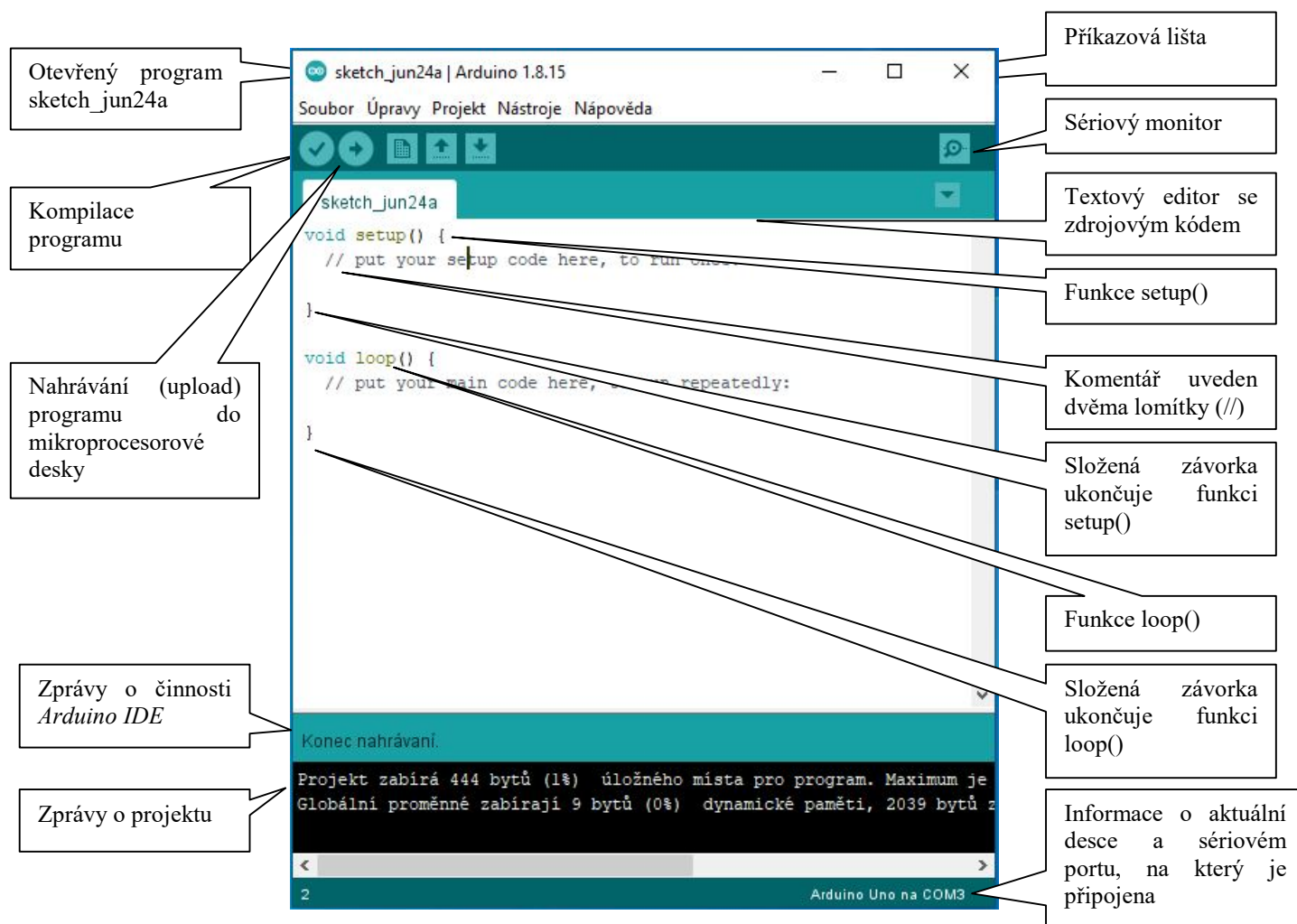
Obr.1 Deska Arduino UNO

1.2 Základní popis programátorského prostředí Arduino IDE

Každý mikropočítač je potřeba nejdříve naprogramovat a zapsat do něj obslužný program. Vhodné vývojové programátorské prostředí je např. Arduino IDE, volně stažitelné na <https://www.arduino.cc/en/software>. Programátorské prostředí Arduino IDE si nainstalujete podle instrukcí na stránkách <https://www.arduino.cc/en/Guide>. [1]

Detailní popis vývojového prostředí Arduino IDE lze nalézt např. na <https://arduino.cz/arduino-ide/> nebo je k dispozici celá volně stažitelná příručka „Průvodce světem Arduina“ na <https://bastlirna.hwkitchen.cz/>. [2]

Vývojové prostředí Arduino IDE sestává z textového editoru pro psaní kódu, prostoru pro zprávy, textové konzole a lišty s příkazy pro běžné funkce (viz. obr.2). Arduino IDE umožňuje vytvoření ovládacího programu pro mikroprocesorovou desku, jeho kompilaci a nahrání přímo do hardware Arduina [2].



Obr.2: Vývojové prostředí Arduino IDE

Softwaru, který píšeme pomocí Arduina IDE, říkáme sketch (návrh). Sketche se píše v textovém editoru, ukládají se do souborů s koncovkou .ino. Na obr.2 je otevřený sketch *sketch_jun24a.ino*. K všem funkcím programu Arduina IDE se lze dostat přes příkazovou lištu. Textový editor umožňuje psaní programového kódu a obsahuje funkce pro vyjímání/vkládání a pro hledání/nahrazování textu. Zprávy (messages) nabízí zpětnou vazbu při ukládání a upozorňují na chyby v napsaném programovém kódu. Na konzoli se ukazují textové výstupy z prostředí Arduina včetně kompletních chybových zpráv a jiných informací. V pravém dolním rohu je zobrazena informace o aktuální mikroprocesorové desce a sériovém portu, přes který je připojena k počítači (bude vysvětleno v části *sériová komunikace*).

Programovací jazyk Arduino IDE lze rozdělit do tří hlavních částí: funkce, hodnoty (proměnné a konstanty) a strukturní elementy. Souhrn dostupných funkcí, typů proměnných a strukturních elementů (operátorů) programovacího jazyka najdete na <https://www.arduino.cc/reference/en/>. Funkce jsou určeny pro řízení mikroprocesorové desky a provádění výpočtů, v proměnných různých typů se uchovávají číselná a datová data, k základním elementům programovacího jazyka patří operátory (binární, aritmetické atd.), kontrolní struktury

(příkazy `if`, `for`, `goto` atd.) a další syntaxní příkazy (středník, složené závorky, `#include`, funkce `setup()`, `loop()`).

Programovací jazyk Arduino IDE je odvozen z C++. Příklad na obr.4 ukazuje prostředí Arduino IDE s nejjednodušším kódem, který je potřeba ke správné kompilaci na Arduino: funkci `setup()` a funkci `loop()`.

(Originál popisu naleznete na <https://www.arduino.cc/en/Tutorial/BuiltInExamples/BareMinimum>).

Funkce `setup()` je volána při spuštění programu. Slouží k inicializaci proměnných, režimů pinů, zahájení používání knihoven atd. Funkce se spustí pouze jednou, po každém zapnutí nebo resetování desky. Po vytvoření funkce `setup()` provádí funkce `loop()` přesně to, co její název napovídá, probíhá postupně ve smyčce. Kód v části `loop()` se používá k aktivnímu ovládání desky. Definice funkce začíná klíčovým slovem `void` a jménem funkce, příkazy, které se mají ve vybrané funkci vykonávat (stejně jako ve smyčce nebo po podmíněném příkazu) se uzavírají do složených závorek (obr.2), jednotlivé příkazy se oddělují středníkem.

Níže uvedený kód (obr.2) ve skutečnosti nic nedělá, ale jeho struktura je užitečná pro kopírování a vkládání, abyste mohli začít s jakýmkoli vlastním programem. Příkazy, které se mají provádět ve vybrané funkci (stejně jako ve smyčce nebo po podmíněném příkazu) se uzavírají do složených závorek (obr.2), jednotlivé příkazy se oddělují středníkem (obr.2).

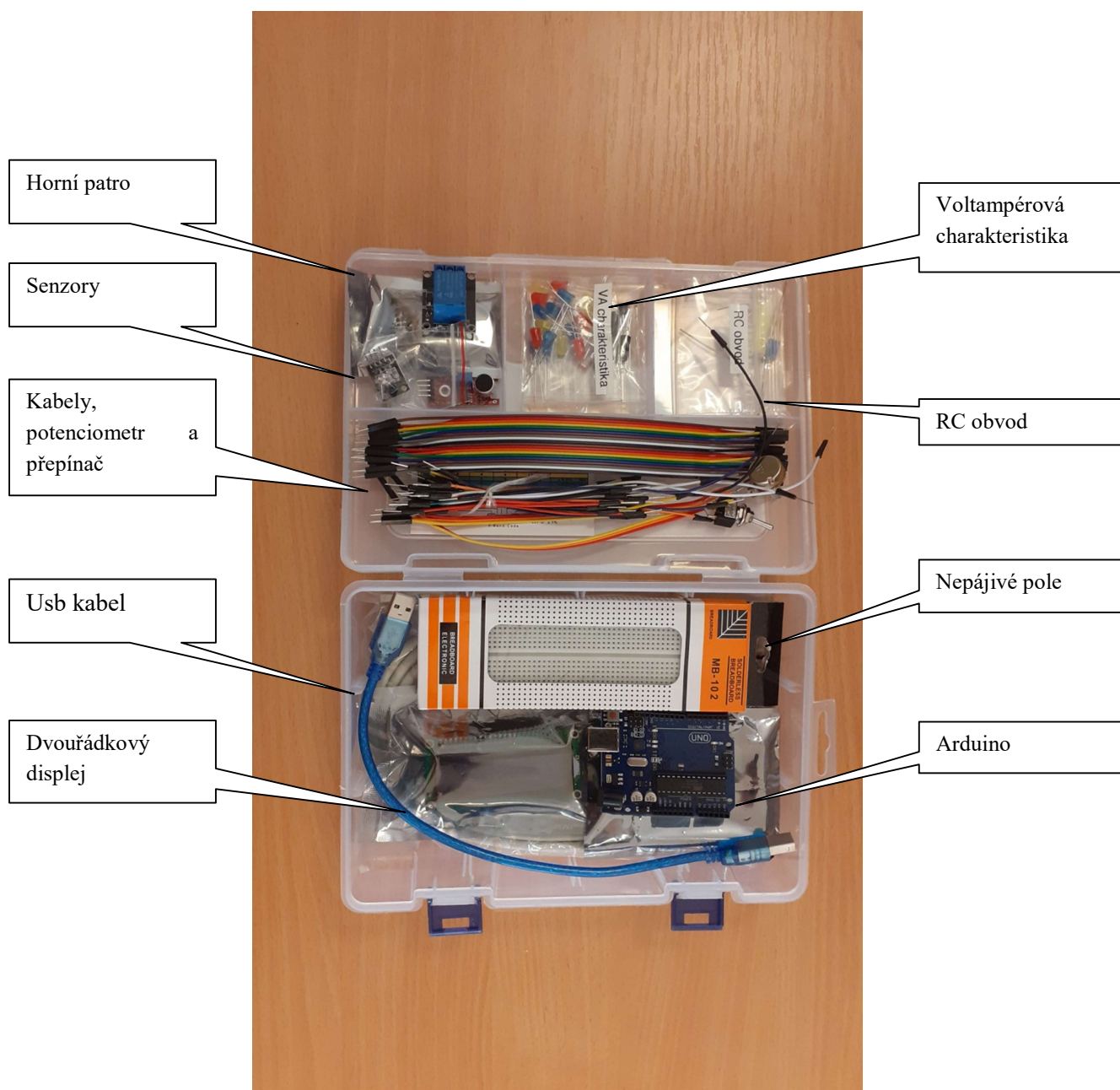
Z příkladu na obr.2 je také jasné, jak lze v kódu vytvářet komentáře. Jakýkoli řádek, který začíná dvěma lomítky (`//`) kompilátor nečte, takže po něm můžete napsat, co chcete. Dvě lomítka mohou být uvedena za funkční kód, aby byly komentáře na stejném řádku, na kterém je kód. Komentování tohoto kódu může být obzvláště užitečné při vysvětlování, jak program funguje krok za krokem.

1.3 Popis měřicí sady s Arduinem

Na obr.3 je fotografie měřicí sady s Arduinem a s komponentami pro vypracování zadaných pracovních úkolů. V plastové krabici jsou komponenty uloženy ve dvou patrech. V horním patře je podélná přihrádka na kabely (potenciometr a přepínač) a 3 menší přihrádky na sensory a na komponenty pro měření pracovních úkolů voltampérové charakteristiky LED a sériového RC členu. Komponenty v malých přihrádkách jsou v igelitovém sáčku s popisem. Ve spodním patře je mikroprocesorová deska Arduino UNO, Usb kabel pro připojení mikroprocesorové desky k PC, nepájivé pole a dvouřádkový displej.

Na vnitřní straně víčka krabice je seznam všech součástek, které sada obsahuje.

Na vnitřní straně víčka krabice je seznam všech součástek, které sada obsahuje.



Obr.3 Měřící sada s Arduinem

Spodní patro: Arduino UNO, nepájivé pole, 2 řádkový displej, Usb propojovací kabel, sada odporů (220Ω , $1k\Omega$, $10k\Omega$)

Horní patro, oddíl kabely: potenciometr, přepínač, kabely (20x tatínek-maminka, 30x tatínek-tatínek)

Horní patro, oddíl VA charakteristika: odpor 100Ω , LED červená 5x, LED žlutá 5x, LED modrá 5x




Horní patro, oddíl RC obvod: kapacita $3.3\mu F$, odpory $10k\Omega$ (označeno 22k), $22.1k\Omega$ (označeno 22k1), $56k\Omega$, $100k\Omega$ (označeno M1), $240k\Omega$ (označeno M24)

Horní patro, oddíl Sensory: Senzor teploty a vlhkosti vzduchu DHT11, fotorezistor 3x, zvukový sensor, 1-kanál relé modul, sensor plamene, RGB LED, zvukový sensor, IR přijímač HX1838, otřesové čidlo 2x.

2 Pracovní úkol 1 – Ukázkové příklady práce a Arduinem

2.1 Programování mikroprocesorové desky

Postup provedení:

- 1) Propojte USB kabelem Arduino UNO s počítačem. Kontrolka napájení (označená ON) by měla svítit.
- 2) Spustíte vývojové prostředí Arduino IDE .
- 3) Nejdříve vyberte (ověřte) typ připojené mikroprocesorové desky (na příkazové liště *Nástroje–Vývojová deska – ArduinoUno*)
- 4) Vyberte (ověřte) Port, na kterém se implementovalo Arduino (na příkazové liště *Nástroje–Port: - COM XX (např. COM3 na obr.2)*, vyberete COM port, na kterém je mikroprocesorová deska připojena, je u něj zapsána). Pozn.: XX je číslo portu, které zjistíme např. ve Správci zařízení v počítači.
- 5) Otevřete nový program (na příkazové liště *Soubor – Nový*). Otevře se nové okno s prázdným programem jako na obr.2. Program obsahuje pouze základní nutné klíčové příkazy *setup()* a *loop()*.
- 6) Zkontrolujte (kompilujte) program, jestli neobsahuje chyby (tlačítkem , Ctrl+R nebo *Projekt – Kontrola/Kompilace* na příkazové liště). Objeví se na dolní liště zpráva Kompilace ukončena a v oknu zpráv jsou pouze zprávy o projektu a nejsou zde žádné informace o chybách (obr.2).
- 7) Nahrajte (upload) program do mikroprocesorové jednotky (tlačítkem , Ctrl+U nebo *Projekt – Nahrát* na příkazové liště). Pokud se program správně nahraje, objeví se na dolní liště zpráva *Konec nahrávání* a v oknu zpráv jsou pouze zprávy o projektu a nejsou zde žádné informace o chybách (obr.2).

2.2 Ukázkový program - Použití digitálních výstupů – Blikání vestavěné LED diody.

2.2.1 Popis programu

Arduino IDE obsahuje vestavěné základní programy (Examples – Příklady), na kterých jsou prezentovány základní možnosti Arduina. Na těchto příkladech si ukážeme základní použití digitálních vstupů/výstupů, analogových vstupů a sériového monitoru. Kompletní popis příkladů najdete na <https://www.arduino.cc/en/Tutorial/BuiltInExamples>.

Nejdříve si ukážeme ovládání digitálního výstupu na příkladu programu *Blink*, výpis programu je na obr.4. Postup spuštění programu je uveden níže pod popisem jednotlivých kroků programu. Tento příklad používá LED integrovanou na mikroprocesorové desce (obr.1) a ovládanou u Arduina UNO přes digitální pin 13. Číslo pinu se u různých desek liší, proto je nahrazeno konstantou LED_BUILTIN. První věc, kterou program udělá, je inicializace digitálního pinu LED_BUILTIN jako výstupní linky následujícím příkazem ve funkci *setup()*.

```
pinMode(LED_BUILTIN, OUTPUT);
```

V hlavní smyčce programu (*loop()*) se LED zapne příkazem.

```
digitalWrite(LED_BUILTIN, HIGH);
```

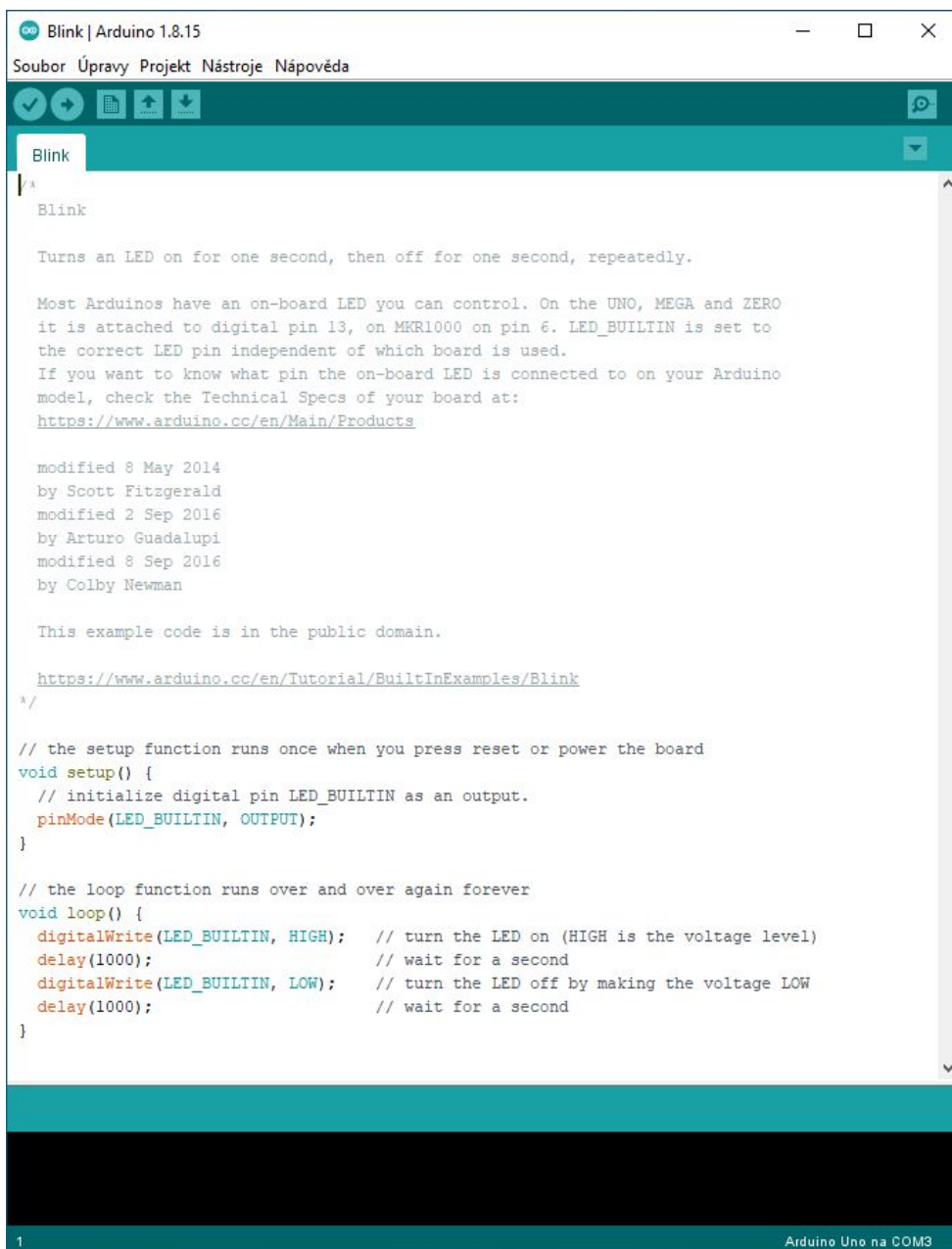
Tento příkaz nastaví úroveň *HIGH* (připojí napětí 5V) na pin 13 a tím i napětí na anodu LED (přes ochranný odpor) a zapne LED. LED se vypne příkazem

```
digitalWrite(LED_BUILTIN, LOW);
```

který nastaví úroveň *LOW*, připojí na pin 13 nulové napětí, LED se vypne. Aby bylo vypínání a zapínání LED okem postřehnutelné, jsou za tyto příkazy vloženy příkazy

```
delay(1000);
```

které říkají mikroprocesoru, aby po určitou dobu nic nedělal, v tomto případě 1s (1000 milisekund).

The image is a screenshot of the Arduino IDE interface. The title bar reads "Blink | Arduino 1.8.15". The menu bar includes "Soubor", "Úpravy", "Projekt", "Nástroje", and "Nápověda". Below the menu bar is a toolbar with icons for opening files, saving, uploading, and downloading. The main text area is titled "Blink" and contains the following text: "Blink", "Turns an LED on for one second, then off for one second, repeatedly.", "Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to the correct LED pin independent of which board is used. If you want to know what pin the on-board LED is connected to on your Arduino model, check the Technical Specs of your board at: <https://www.arduino.cc/en/Main/Products>", "modified 8 May 2014", "by Scott Fitzgerald", "modified 2 Sep 2016", "by Arturo Guadalupi", "modified 8 Sep 2016", "by Colby Newman", "This example code is in the public domain.", and "<https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink>". Below this text is the code for the Blink example:

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
```



 At the bottom of the window, the status bar shows "1" on the left and "Arduino Uno na COM3" on the right.

Obr.4 Příklad programu *Blink*

Dobu blikání LED můžete změnit změnou doby v příkazu *delay()*. Změna se projeví po uploadování změněného programu do desky (viz bod 7 v čl. 2.1 - Programování mikroprocesorové desky).

Vyzkoušejte si různou dobu svícení a zhasnutí LED. Vyzkoušejte si, jakou nejvyšší frekvenci blikání vaše oko rozliší.

2.2.2 Postup provedení – program *Blink*:

- 1) Otevřete si příklad *Blink*. (Soubor - Příklady - 01.Basic – *Blink*, <https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink> obr.4).
- 2) Nastavte v Nástrojích správnou desku *Arduino UNO*, v Nástrojích rovněž nastavte správný Port COM.
- 3) Zkontrolujte (kompilujte) program, jestli neobsahuje chyby (tlačítkem , Ctrl+R nebo *Projekt – Kontrola/Kompilace* na příkazové liště).
- 4) Nahrajte (upload) program do mikroprocesorové jednotky (tlačítkem , Ctrl+U nebo *Projekt – Nahrát* na příkazové liště). Pokud se program správně nahraje, objeví se na dolní liště zpráva *Konec nahrávání* a v oknu zpráv jsou pouze zprávy o projektu a nejsou zde žádné informace o chybách (obr.2).
- 5) Vyzkoušejte nastavit program pro různou dobu svícení a zhasnutí LED a svícení LED.

2.3 Ukázkový program - Použití analogových vstupů a Sériového monitoru

Použití analogových vstupů a sériového monitoru si ukážeme na příkladu *ReadAnalogVoltage*. Výpis programu je na obr.5, postup spuštění programu je uveden v odst. 2.3.3 Tento příklad ukazuje, jak načíst hodnotu napětí na analogovém vstupu (pin A0), převést hodnoty získané z funkce *analogRead()* na napětí ve voltech a poslat je z mikroprocesorové desky po sériové lince na sériový monitor softwaru Arduino IDE.

Mikrokontrolér na desce ArduinoUno má uvnitř obvod zvaný analogově digitální převodník (ADC), který čte měnící se napětí v rozsahu 0-5V a převádí ho na celé číslo v rozmezí mezi 0 a 1023 (10 bitový převodník)

2.3.1 Sériová komunikace

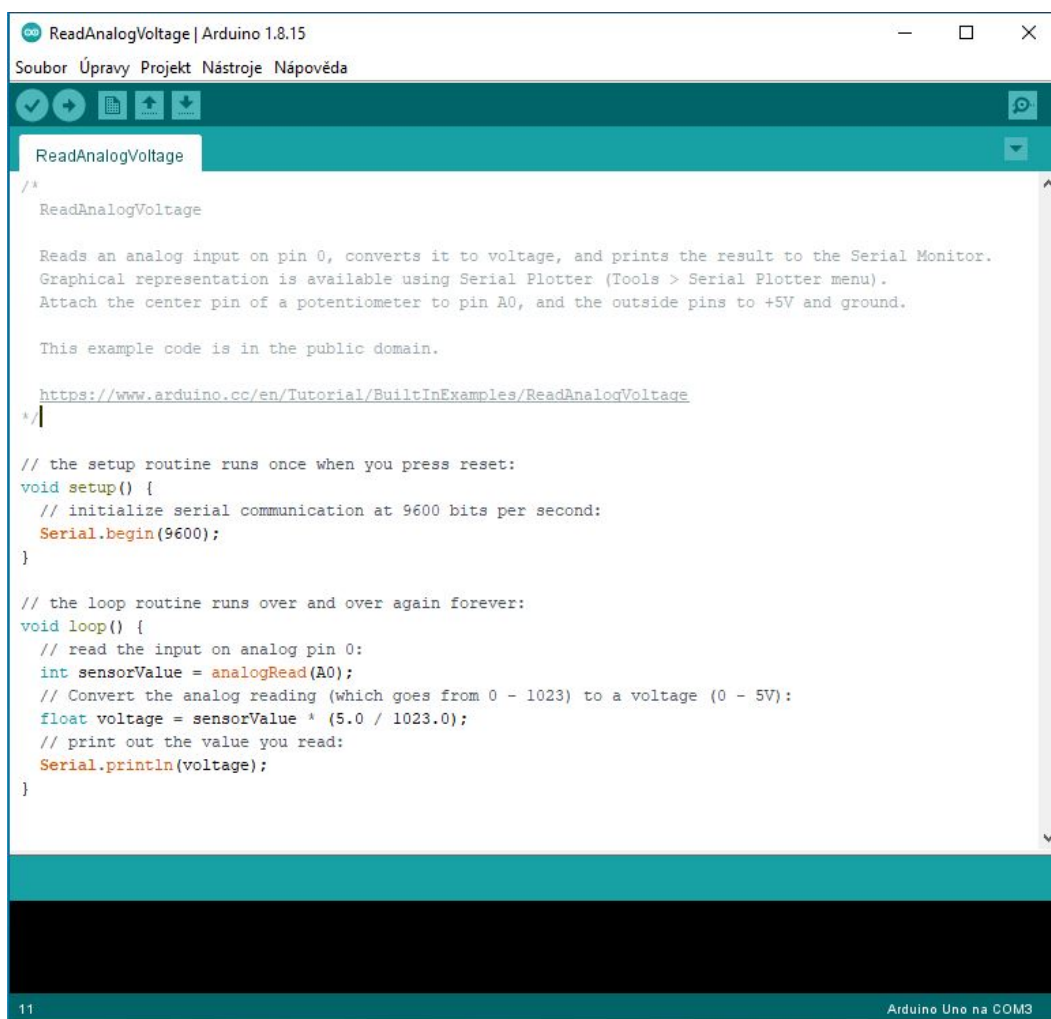
Sériový monitor je jednou ze základních prostředků v programu *Arduino IDE* při vytváření projektů s Arduinem. Může být použit pro ladění a testování projektů. Sériový monitor čte data přicházející po sériové lince. Sériová linka (port) se používá pro komunikaci mezi mikroprocesorovou deskou Arduino a počítačem nebo jinými zařízeními. Všechny desky Arduino mají alespoň jeden sériový port.

Sériová komunikace nebo sériový přenos je v telekomunikacích a v informatice nazýván proces přenosu dat postupně po jednotlivých bitech (tj. sekvenčně) pomocí komunikačního kanálu

nebo sběrnice. Sériová sběrnice používá pro přenos dat a řízení sběrnice jeden vodič (resp. dvojici signál-nulový vodič). Protokol sběrnice popisuje, jaký je formát přenášených dat, časování a řízení přenosu.

Nejčastěji používané sériové komunikační rozhraní osobních počítačů a další elektroniky je sériový port nebo linka RS-232, který umožňuje propojení a vzájemnou sériovou komunikaci dvou zařízení. Podrobnější popis rozhraní lze najít například na <https://cs.wikipedia.org/wiki/RS-232>. V počítačích se sériové porty označují názvem *COM* a číslem.

I když komunikující zařízení při sériové komunikaci znají rychlost, jakou se data přenášejí, musí přijímač začít přijímat ve správný okamžik, tedy musí proběhnout *synchronizace* vysílače a přijímače. Vysílač pošle nějaká definovaná data po datovém vodiči, po jejichž přijetí se přijímač zasynchronizuje. V případě RS232 každé sekvenci datových bitů předchází jeden *start bit*, kterým se logická hodnota na lince přepne (původně v klidovém stavu) do opačného stavu. Po datových bitech následuje *paritní bit* a za ním jeden nebo více *stop bitů*, během kterých je linka opět v klidovém stavu. Paritní bit je nejjednodušší způsob, jak bez nároků na výpočetní výkon zabezpečit (zkontrolovat) bezchybnost přenosu dat. Ve vysílacím zařízení se sečte počet jedničkových bitů a doplní se paritním bitem tak, aby byla zachována předem dohodnutá podmínka sudého nebo lichého počtu jedničkových bitů.



```
ReadAnalogVoltage | Arduino 1.8.15
Soubor Úpravy Projekt Nástroje nápověda

ReadAnalogVoltage

/*
  ReadAnalogVoltage

  Reads an analog input on pin 0, converts it to voltage, and prints the result to the Serial Monitor.
  Graphical representation is available using Serial Plotter (Tools > Serial Plotter menu).
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/ReadAnalogVoltage
*/

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
  // print out the value you read:
  Serial.println(voltage);
}
```

11 Arduino Uno na COM3

Obr.5 Příklad programu *ReadAnalogVoltage*

Aby se komunikující zařízení byla schopná domluvit, je nutné na obou správně nastavit parametry přenosu. Nastavitelné parametry přenosu jsou: Rychlost v bitech za sekundu (baud rate): 1200 až 115200 baud, počet datových bitů: 5-8, počet stop bitů: 0-2 (Stop bit definuje ukončení datového rámce), parita :n,o,e,m (none-žádná, odd-lichá, e-sudá, m-mark)

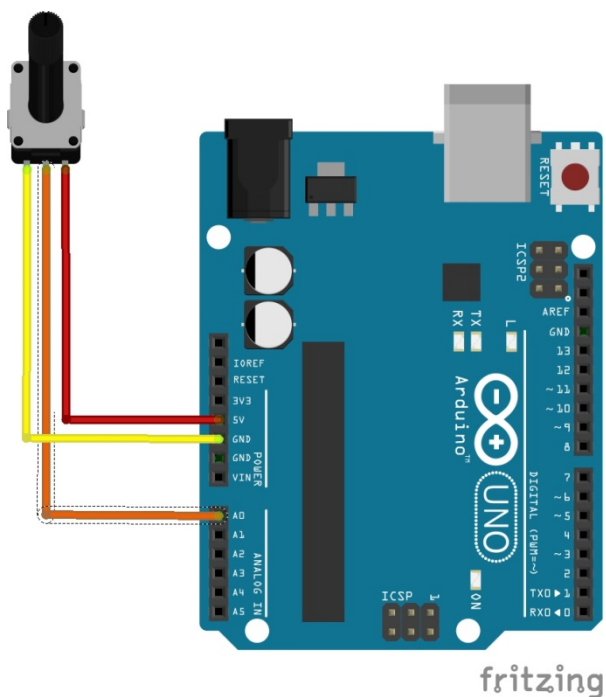
V případě desek Arduino je přednastaveno standardní nastavení parametrů sériové komunikace **9600, 8,N,1**. (Baudová rychlost **9600**, **8** datových bitů, **N** – bez PARITY, **1** STOP bit).

2.3.2 Analogové vstupy

Na obr.6 je zapojení pro realizaci příkladu *ReadAnalogVoltage*. K desce Arduino jsou připojeny tři vodiče potenciometru. První vodič jde z jednoho z vnějších kolíků potenciometru na zem (pin GND, žlutý). Druhý jde z druhého vnějšího kolíku potenciometru na napájení 5 voltů (pin 5V, červený). Třetí jde od prostředního kolíku potenciometru k analogovému vstupu 0 (pin A0, oranžový).

Otáčením hřídele potenciometru měníte velikost odporu na obou stranách kontaktu, který je připojen ke středovému kolíku potenciometru. Tím se změní napětí na středovém pinu. Když se odpor mezi středem a stranou připojenou k 5 V blíží nule (a odpor na druhé straně se blíží 10 kΩ), napětí na středovém kolíku se blíží k 5 V. Když jsou odpory obráceny, napětí na středovém kolíku se blíží 0 V (zemi). Toto proměnné napětí je analogové napětí, které je přivedeno na analogový vstup mikroprocesorové desky.

Analogově digitální převodník (ADC) mikrokontroléru na desce ArduinoUno čte měnící se analogové napětí v rozsahu 0-5V a převádí ho na celé číslo v rozmezí mezi 0 a 1023 (10 bitový převodník). Když je hřídel potenciometru otočena úplně na jednu stranu, na středovém kolíku je napětí 0V a vstupní hodnota je 0. Když je hřídel otočena úplně v opačném směru, na kolík jde 5V a vstupní hodnota je 1023. Při nastavení potenciometru mezi mezními hodnotami funkce *analogRead()* vrátí číslo mezi 0 a 1023, které je úměrné velikosti napětí aplikovaného na středový kolík.



Obr.6 Zapojení obvodu pro příklad *ReadAnalogVoltage*

V příkladu výpisu programu na obr.5 je ve funkci *setup()* jediný příkaz, a to příkaz pro zahájení sériové komunikace mezi deskou a počítačem rychlostí 9600 bitů za sekundu (baud rate)

```
Serial.begin(9600);
```

Pozn.: Trochu vysvětlíme, kde se vzalo 9600 aj. Sériová komunikace COM posílá data po jednom vodiči (samozřejmě proti zemnímu vodiči), jsou tedy potřeba dva vodiče. Data se posílají ve formátu nul a jedniček (tzv. logických úrovní HIGH a LOW. Datová digitální informace (jedna hodnota) má nejčastěji 8 bitů (umožňuje rozsah 0 až 255). Naše analogová informace na pinu A0 má 10 bitový rozsah a proto je tato informace posílána ve dvou po sobě jdoucích datových balíčcích. Datový balíček obsahuje 8-mi bitovou datovou informaci a je ještě obalen tzv. datovou synchronizační směsí, která sestává z úvodního START bitu, z konečného STOP bitu a z PARITY (čili jeden datový balíček má 10 úrovní HIGH a LOW). Přenáší se asynchronně, to znamená nepravidelně, jenom tehdy, když je potřeba. Tyto digitální změny HIGH a LOW uvnitř datového balíčku se mohou vysílat různou (tzv. Baudovou) rychlostí. V našem případě se využívá rychlosti 9600 Baudů (bitů za sekundu). Někdy se tato informace o nastavení přenosu sériové linky zapisuje např. **9600, 8, N, 1**. (Baudová rychlost **9600**, **8** datových bitů, **N** – bez PARITY, **1** STOP bit).

Dále v hlavní smyčce vašeho kódu musíte vytvořit proměnnou pro uložení hodnoty napětí, přicházející z vašeho potenciometru, a která bude celé číslo mezi 0 a 1023. Na to je ideální typ proměnné *int*. Následující řádek definuje novou proměnnou *sensorValue* typu *int* a ukládá do ní hodnotu naměřenou AD převodníkem na pinu A0.

```
int sensorValue = analogRead(A0);
```

chcete-li změnit hodnoty, které čte AD převodník z rozsahu 0-1023 na rozsah, který odpovídá hodnotě napětí ve voltech, budete muset vytvořit další proměnnou *voltage*, která je reálné číslo (*float*), a hodnoty z proměnné *sensorValue* přepočítat, jak je ukázáno na následujícím řádku

```
floatvoltage= sensorValue * (5.0 / 1023.0);
```




Nakonec musíte naměřené hodnoty poslat po sériové lince do počítače. To lze provést následujícím příkazem v posledním řádku kódu (obr.5)

```
Serial.println(voltage)
```

Když nyní otevřete svůj sériový monitor v Arduino IDE (kliknutím na ikonu na pravé straně horního zeleného pruhu nebo stisknutím Ctrl+Shift+M), měli byste vidět sloupec stále se pod sebe vypisujících čísel v rozmezí od 0,0 do 5,0. Jak otáčíte potenciometrem, hodnoty se budou měnit a budou odpovídat napětí přicházejícímu na pin A0.

Pro několik hodnot nastavení potenciometru vypojte kabel ze vstupu A0, změřte hodnotu napětí voltmetrem a porovnejte ji s hodnotou zobrazenou sériovým monitorem.

2.3.3 Postup provedení – program *ReadAnalogVoltage*:

- 1) Připojte potenciometr k mikroprocesorové desce podle obr.6
- 2) Otevřete si příklad *ReadAnalogVoltage*. (Soubor - Příklady - 01.Basic - *ReadAnalogVoltage*, <https://www.arduino.cc/en/Tutorial/BuiltInExamples/ReadAnalogVoltage>, obr.5).
- 3) Zkontrolujte (kompilujte) program, jestli neobsahuje chyby (tlačítkem , Ctrl+R nebo *Projekt – Kontrola/Kompilace* na příkazové liště).
- 4) Nahrajte (upload) program do mikroprocesorové jednotky (tlačítkem , Ctrl+U nebo *Projekt – Nahrát* na příkazové liště). Pokud se program správně nahraje, objeví se na dolní liště zpráva *Konec nahrávání* a v oknu zpráv jsou pouze zprávy o projektu a nejsou zde žádné informace o chybách.
- 5) Spusťte sériový monitor (tlačítkem  v pravém horním rohu, Ctrl+Shift+M nebo *Nástroje -Sériový monitor* na příkazové liště)
- 6) Otáčejte potenciometrem a sledujte výstupní hodnotu napětí v okně sériového monitoru. Změřte hodnotu napětí na středním kolíku potenciometru voltmetrem a porovnejte ji s hodnotou zobrazenou sériovým monitorem

Tento příklad demonstruje, jak lze mikroprocesorovou desku Arduino použít pro naměření a digitalizaci analogových signálů a přenos dat do počítače.

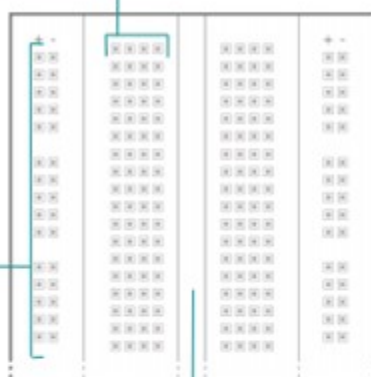
3 Zapojování obvodů s pomocí nepájivé kontaktní desky

Při realizaci dalších příkladů budeme zapojovat obvody s pomocí nepájivé kontaktní desky. Kontaktní deska je základní pomůcka pro vytváření elektronických obvodů bez nutnosti pájení. Popis kontaktní nepájivé desky je na obr.7. Jednotlivé otvory v desce se využívají pro přidání částí obvodu. Uvnitř kontaktní desky jsou vodiče propojující zdířky viz obr.7. Každá horizontální řada zdířek je spojena vodičem, dlouhé vertikální řady na stranách desky jsou spojeny vodičem, tyto zdířky jsou většinou používány pro přivedení napájení a země. Na obr.8 je fotografie kontaktní nepájivé desky.

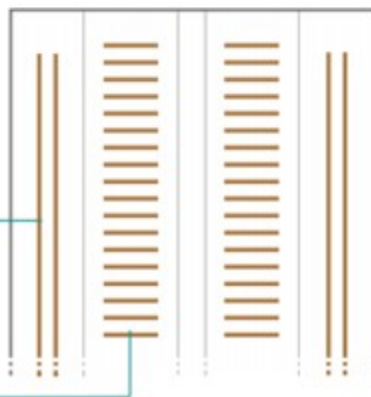
Každá řada horizontálních zdířek kontaktního pole, jsou spojeny vodičem, který je uvnitř kontaktní desky.

Vertikální řada zdířek je propojena vodičem po celé délce kontaktní desky. Tyto zdířky jsou obvykle používány pro přivedení napájení a země.

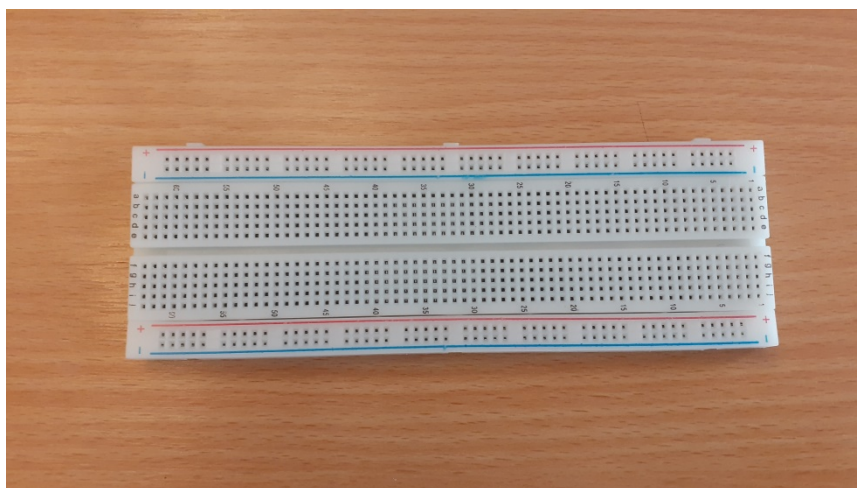
Oddělovací sloupek dvou polovin kontaktní desky.



Uvnitř kontaktní desky jsou vodiče propojující zdířky, jak je vidět na obrázku.



Obr.7 Popis kontaktní nepájivé desky



Obr.8 Fotografie kontaktní nepájivé desky

4 Pracovní úkol 2 – měření voltampérových charakteristik LED

4.1 Popis měření

Základní teorii týkající se polovodičových diod najdete ve studijním textu k úloze č.11 - Charakteristiky diod [3].

Pro měření voltampérových charakteristik LED diod zapojíme obvod do nepájivého pole podle obr.9 a schématu na obr.10. K měření použijeme upravený program *ReadAnalogVoltage*.

Stejně jako v předchozím příkladu je potenciometr použit pro nastavení napětí na měřené diodě, na krajní kolíky potenciometru je přivedeno napětí 5V (červený kabel) a zem (žlutý kabel), napětí ze středního kolíku je vedeno na měřenou LEDdiodu(oranžový kabel) a na vstupní AD kanál mikroprocesorové desky A0 (oranžový kabel, U_{A0}). Toto napětí se rozdělí mezi sériově zapojenou LED diodu a 100Ω odpor R . Jedna strana odporu R (mezi odporem a diodou) je modrým kabelem připojena na vstupní AD kanál A1 (napětí U_{A1}). Druhá strana odporu R je připojena na zem (žlutý kabel). Měření napětí na odporu R vůči zemi (U_{A1} , kanál A1) při známé hodnotě velikosti odporu R umožní pomocí Ohmova zákona určit proud LED diodou I_{LED}

$$I_{LED} = U_{A1} / R_2 \quad (1)$$

Napětí na LED diodě U_{LED} je dané rozdílem napětí naměřených na kanálech A0 a A1, $U_{A0} - U_{A1}$.

$$U_{LED} = U_{A0} - U_{A1} \quad (2)$$

Program upravíme tak, aby měřil hodnoty na dvou kanálech A0, A1, přepočítával naměřené hodnoty na napětí ve voltech a výsledné hodnoty posílal po sériové lince. Upravený program pro měření VA charakteristiky diod je na **obr.11**. Program byl uložen pod jménem *ReadAnalogVoltageVAdiody*. Program vznikl úpravou příkladu *ReadAnalogVoltage*. Ve funkci *loop()* byly doplněny následující příkazy

```
int sensorValue2 = analogRead(A1);
```

tímto příkazem byla vytvořena nová proměnná *sensorValue2* pro druhou hodnotu signálu čtenou AD převodníkem na pinu A1

```
float voltage2= sensorValue2 * (5.0 / 1023.0);
```

celočíslná hodnota *sensorValue2* v rozsahu 0 až 1023 je přepočtena na reálnou hodnotu napětí ve voltech na pinu A1 (U_{A1}). Byly doplněny i příkazy pro vizualizaci naměřených hodnot přes sériovou linku.

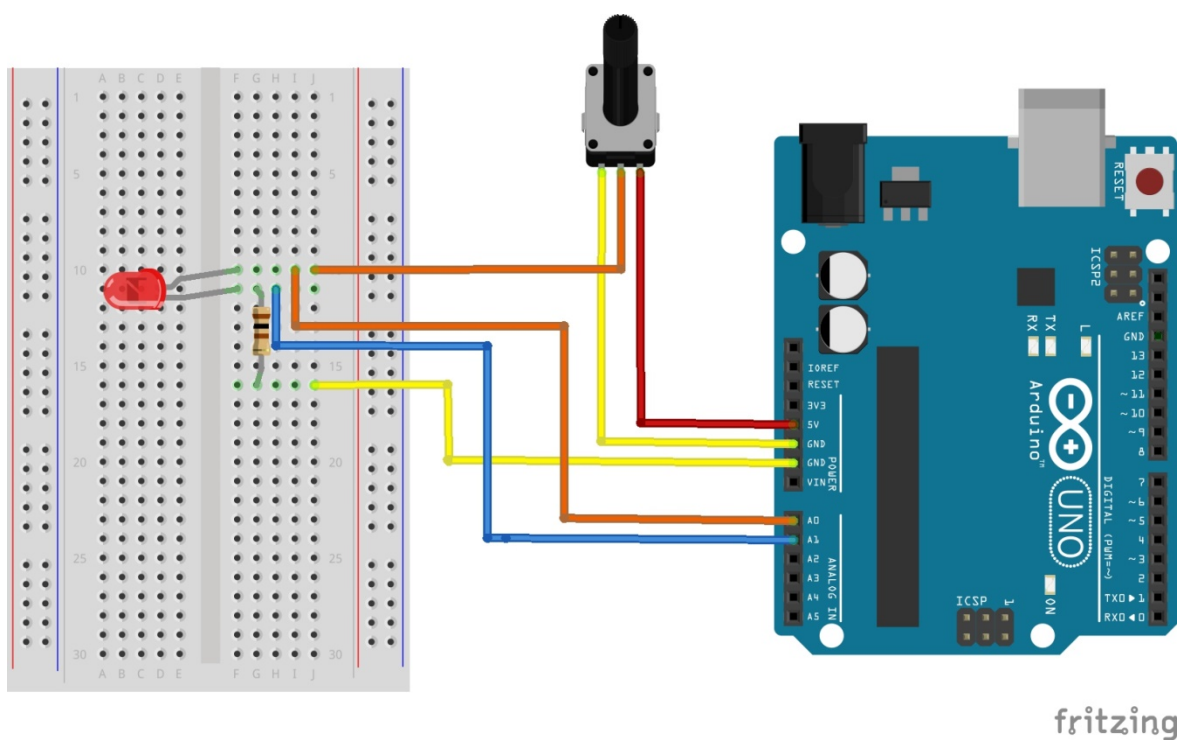
```
Serial.print(voltage);
```

```
Serial.print(";");
```

```
Serial.println(voltage2);
```

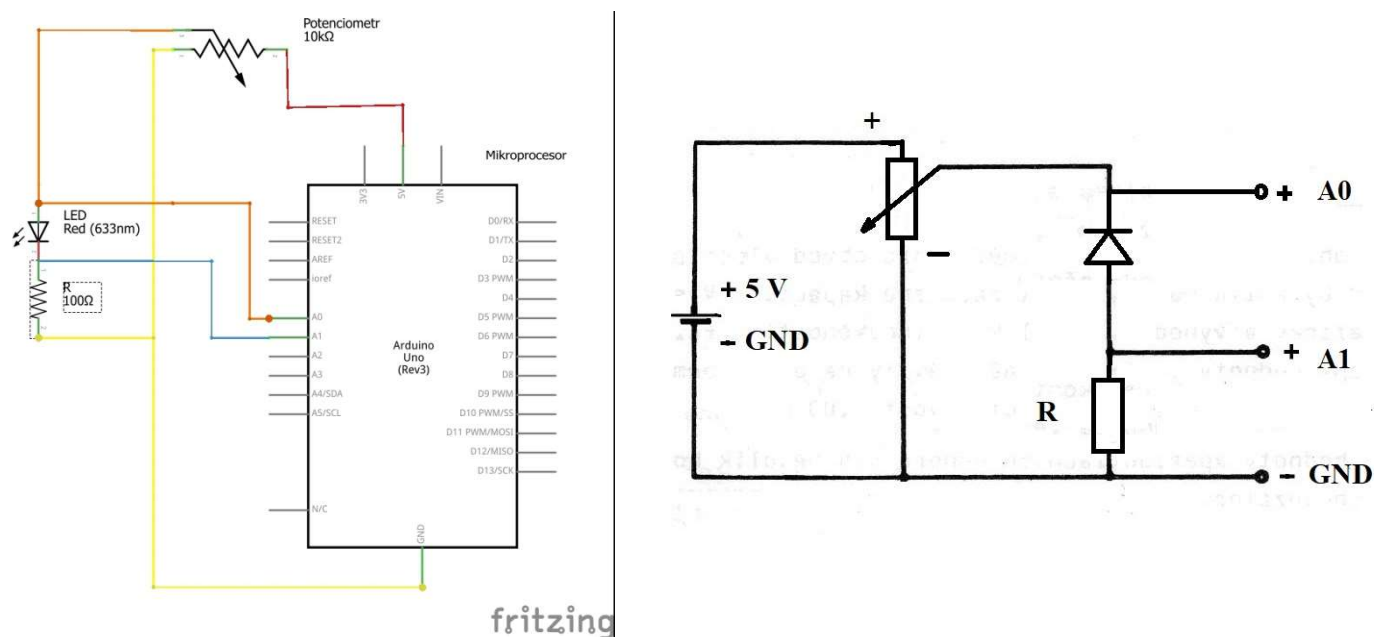
Po sériové lince je prvním příkazem do počítače (sériového monitoru) poslána hodnota napětí na pinu A0 (U_{A0}), dalším příkazem středník, třetím příkazem hodnota napětí na pinu A1. Příkaz *println()* posílá na sériovou linku hodnotu proměnné následovanou znakem návratu vozíku (ASCII 13, or 'r') a znakem nového řádku (ASCII 10, or 'n'), to způsobí, že další hodnoty posílané po sériové lince budou zobrazeny na nové řádce. V sériovém monitoru budou, jak je zřejmé na obr.12, na každé řádce zobrazeny vždy 2 hodnoty naměřených napětí oddělené středníkem.

Řádky byly v programu doplněny kopírováním původních příkazů použitím standardních zkratkových příkazů Ctrl+C, Ctrl+V a dodatečnou úpravou na finální verzi.






Obr.9 Zapojení obvodu pro měření voltampérových charakteristik diod

Voltampérovou charakteristiku diod proměřte v propustném i závěrném směru (otočte diodu v nepájivém poli). Uvědomte si, že v závěrném směru diodou neteče proud a dioda nesvíí



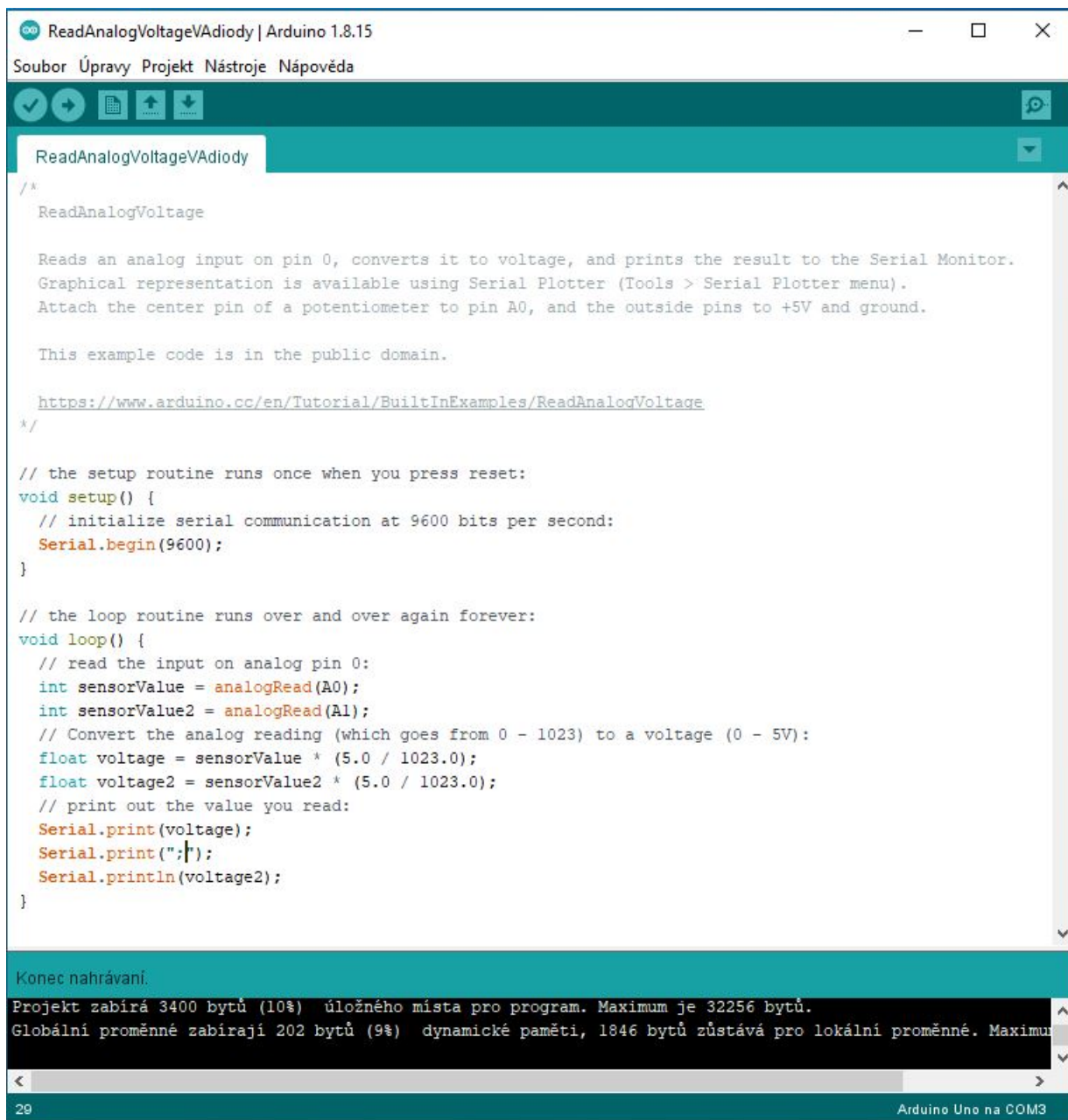
Obr.10 Schéma obvodu pro měření voltampérových charakteristik diod

4. 2 Postup provedení – Voltampérová charakteristika LED:

- 1) Zapojte obvod podle obr.9 a schématu na obr.10.
- 2) Otevřete si program *ReadAnalogVoltage*. (Soubor - Příklady - 01.Basic - *ReadAnalogVoltage*, <https://www.arduino.cc/en/Tutorial/BuiltInExamples/ReadAnalogVoltage>, obr.5).
- 3) Upravte program pro měření a zobrazování dvou napětí podle obr.11
- 4) Zkontrolujte (kompilujte) program, jestli neobsahuje chyby (tlačítkem , Ctrl+R nebo *Projekt – Kontrola/Kompilace* na příkazové liště). Pokud je program bez chyb, objeví se na dolní liště zpráva *Kompilace ukončena* a v oknu zpráv jsou pouze zprávy o projektu a nejsou zde žádné informace o chybách.
- 5) Nahrajte (upload) program do mikroprocesorové jednotky (tlačítkem , Ctrl+U nebo *Projekt – Nahrát* na příkazové liště). Pokud se program správně nahraje, objeví se na dolní liště zpráva *Konec nahrávání* a v oknu zpráv jsou pouze zprávy o projektu a nejsou zde žádné informace o chybách (obr.2).
- 6) Spustěte sériový monitor (tlačítkem  v pravém horním rohu, Ctrl+Shift+M nebo *Nástroje - sériový monitor* na příkazové liště)
- 7) Otáčejte potenciometrem a zapisujte naměřené hodnoty napětí U_{A0} , U_{A1} . Ze změřených hodnot vypočítejte napětí a proud tekoucí měřenou LED diodou, sestrojte VA charakteristiku pro 2 různé LED.

Pokyny k měření:

- Odpor 100Ω (označený 100R) a LED diody pro měření jsou v sáčku označeném VA charakteristika
- Pro měření si vyberte dvě ze tří diod (červená, modrá, zelená)
- Voltampérovou charakteristiku diod proměřte v propustném i závěrném směru (otočte diodu v nepájivém poli). Uvědomte si, že v závěrném směru diodou neteče proud a dioda nesvítí.



```
ReadAnalogVoltageVAdiody | Arduino 1.8.15
Soubor Úpravy Projekt Nástroje nápověda

ReadAnalogVoltageVAdiody

/*
  ReadAnalogVoltage

  Reads an analog input on pin 0, converts it to voltage, and prints the result to the Serial Monitor.
  Graphical representation is available using Serial Plotter (Tools > Serial Plotter menu).
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.

  This example code is in the public domain.

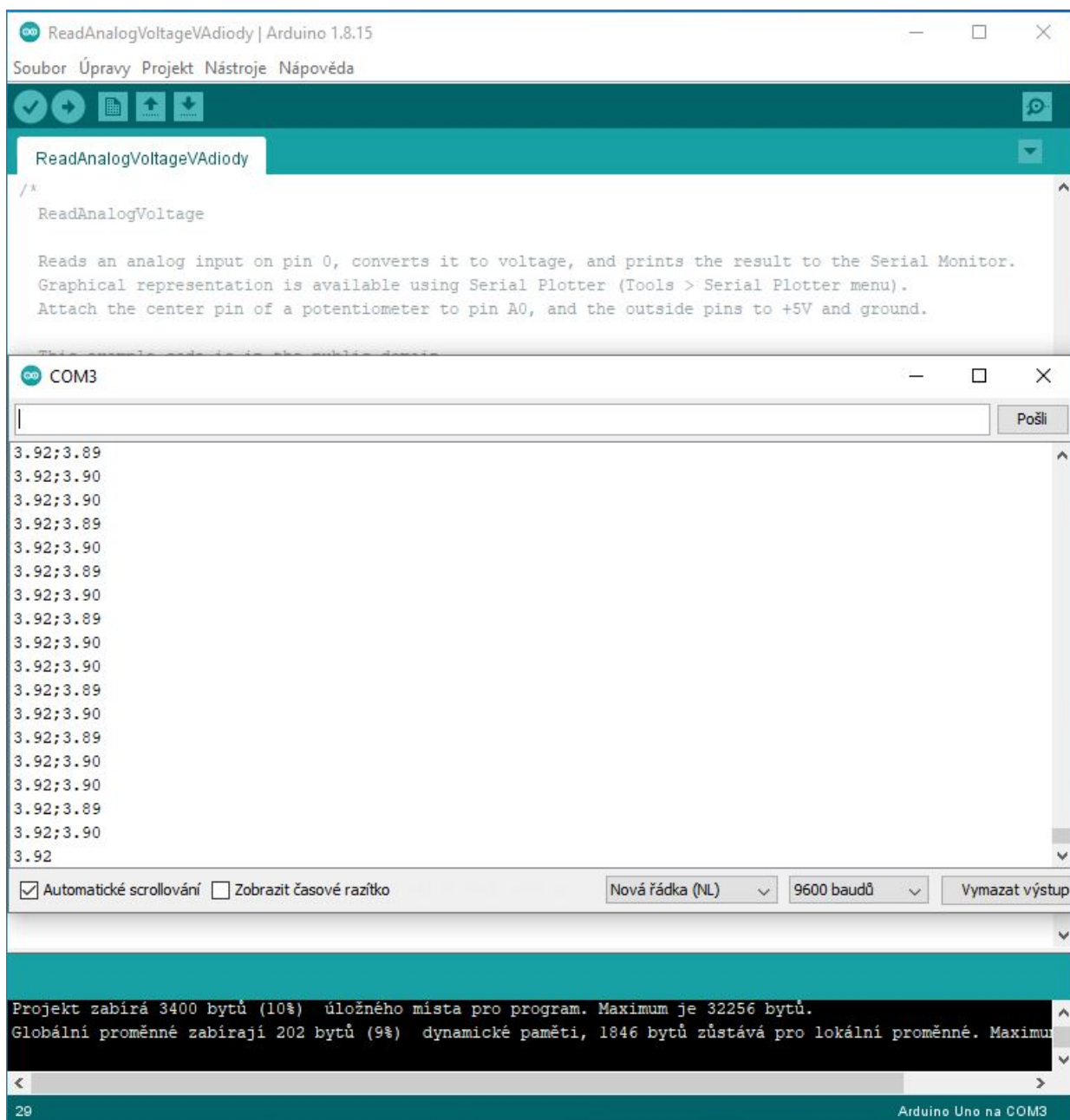
  https://www.arduino.cc/en/Tutorial/BuiltInExamples/ReadAnalogVoltage
*/

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  int sensorValue2 = analogRead(A1);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
  float voltage2 = sensorValue2 * (5.0 / 1023.0);
  // print out the value you read:
  Serial.print(voltage);
  Serial.print(";");
  Serial.println(voltage2);
}

Konec nahrávání.
Projekt zabírá 3400 bajtů (10%) úložného místa pro program. Maximum je 32256 bajtů.
Globální proměnné zabírají 202 bajtů (9%) dynamické paměti, 1846 bajtů zůstává pro lokální proměnné. Maximum
29
Arduino Uno na COM3
```

Obr.11 Program Read AnalogVoltageVAdiody



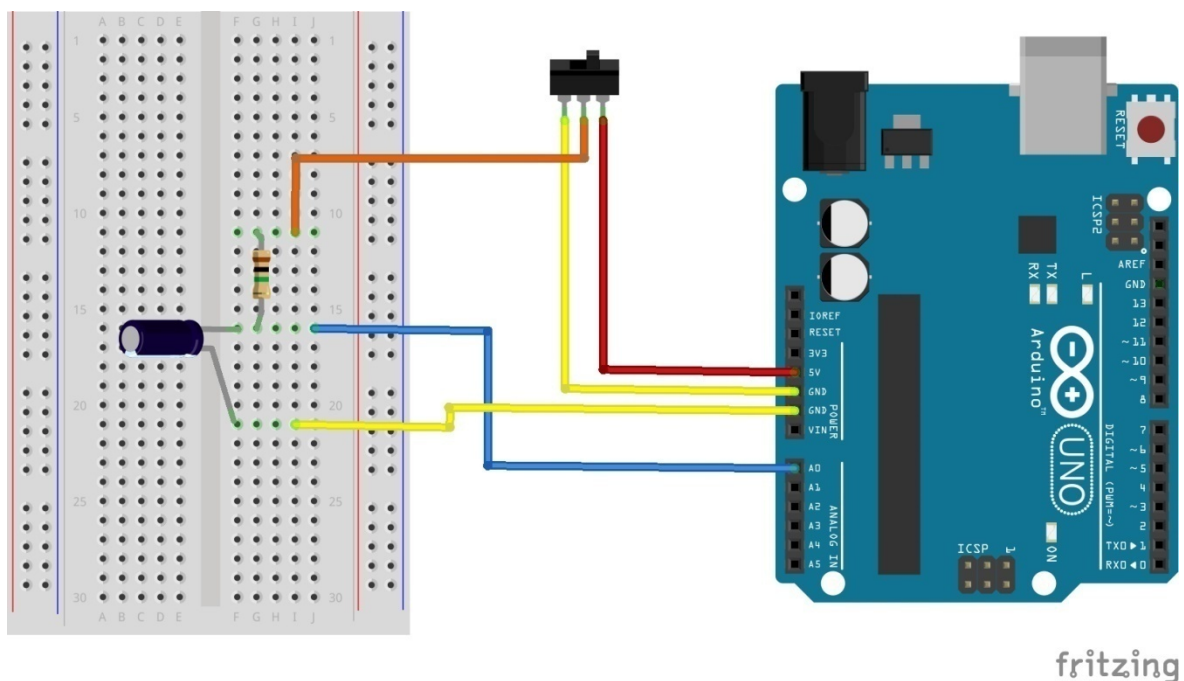
Obr.12 Program ReadAnalogVoltageVAdiody se spuštěným sériovým monitorem

5 Pracovní úkol 3 – měření časové závislosti vybíjení a nabíjení kondenzátoru v sériovém RC obvodu

5.1 Popis měření

Základní teorii týkající se přechodových jevů v sériovém RC obvodu najdete ve studijním textu k úloze č.18 - Přechodové jevy v RCL obvodu [4].

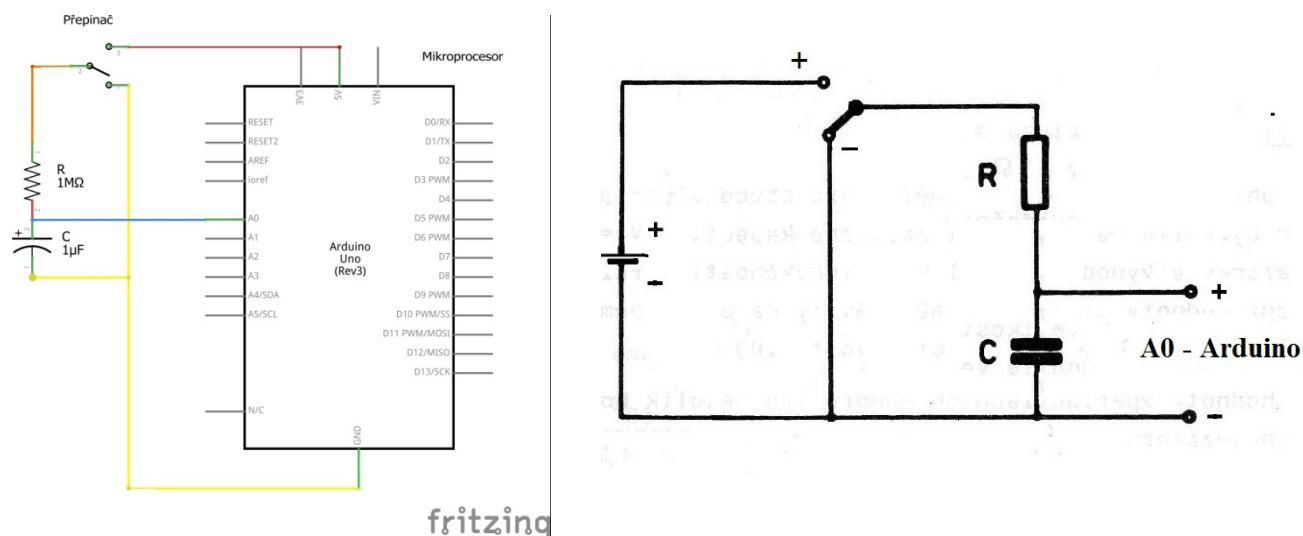
Pro měření časové závislosti vybíjení a nabíjení kondenzátoru v RC obvodu zapojíme obvod do nepájivého kontaktního pole podle obr.13, schéma obvodu je na obr.14. K měření použijeme upravený program *ReadAnalogVoltage*. Postup spuštění a nastavení programu je uveden v odst. 5.3.



Obr.13 Zapojení obvodu pro měření časové závislosti vybíjení a nabíjení kondenzátoru v RC obvodu

Přepínač je použit pro změnu napětí v obvodu, na krajní kolíky přepínače je přivedeno napětí 5V (pin 5V, červený kabel) a zem (pin GND, žlutý kabel), přepnutím přepínače se na střední kontakt přepínače připojí +5V nebo zem. Napětí je ze středního kolíku přepínače přivedeno (oranžový kabel) na měřený obvod. V obvodu je sériově zapojený odpor R a kondenzátor s kapacitou C . Při přepnutí z 0V na 5V dojde k nabíjení a při přepnutí z 5V na 0V k vybíjení kondenzátoru přes sériově připojený odpor.

Pro grafické sledování vývoje změn signálů na analogových vstupech je možné v Arduino IDE použít sériový plotter, který čte data ze sériového portu a zobrazuje je ve formě grafu. Na obr.15 je ukázka použití sériového plotteru pro zobrazení změn napětí na kondenzátoru v RC obvodu.



Obr.14 Schéma obvodu pro měření časové závislosti vybíjení a nabíjení kondenzátoru v RC obvodu

Grafické zobrazení naměřených dat pomocí sériového ploteru nám dává představu o vývoji napětí po přepnutí přepínače, ale neumožňuje nám přesně kvantitativně vyhodnotit časový průběh měření (na x ose ploteru je pořadí měření). Abychom mohli určit čas měření provedeného AD převodníkem mikroprocesoru, musíme program mikroprocesoru rozšířit o příkaz *millis()* (viz obr.16,

```
Serial.print(millis());
```

který pošle po sériové lince hodnotu času v počtu milisekund, které uběhly od startu programu (viz <https://www.arduino.cc/reference/en/language/functions/time/millis/>) v mikroprocesorové desce. Příkaz je následován standardní dvojicí příkazů,

```
Serial.print(";");
```


```
Serial.println(voltage);
```

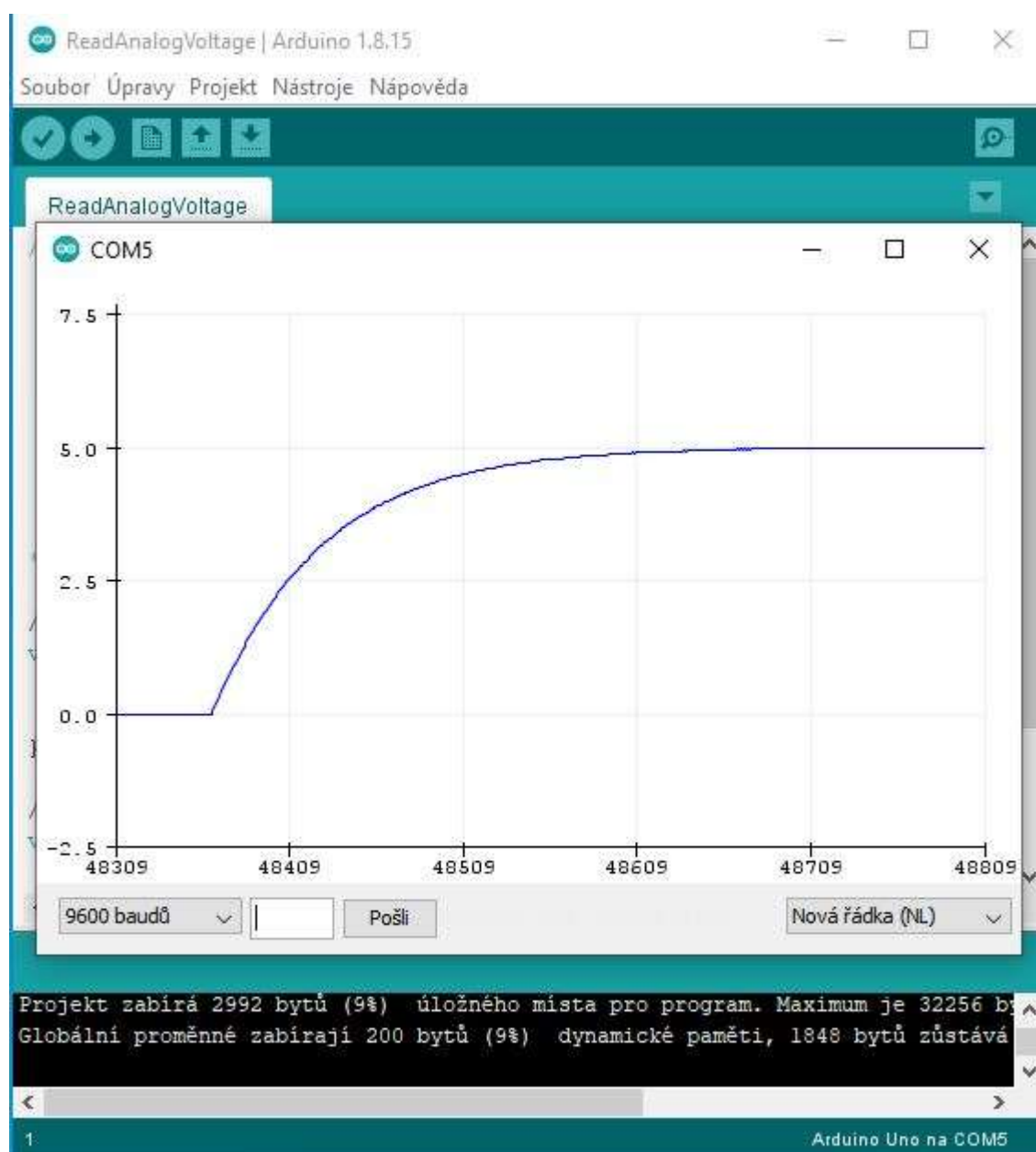
které pošlou po sériové lince středník pro oddělení čísel a hodnotu naměřeného napětí, následovanou znaky pro ukončení řádky. V sériovém monitoru budou zobrazeny, stejně jako na obr.12, na každé řádce vždy 2 hodnoty oddělené středníkem, první hodnota je čas měření v milisekundách, druhá hodnota je naměřené napětí na kondenzátoru. Takto máme ke každé naměřené hodnotě napětí i přesnou hodnotu času, kdy byla naměřena.

Při měření VA charakteristiky diod v pracovním úkolu 2 jsme měřili statickou charakteristiku, kterou je možné odečíst ze sériového monitoru bod po bodu při daném nastavení potenciometru a ustálení hodnot. Při měření časové závislosti nabíjení kondenzátoru (a jiných přechodových jevů v obvodu) je nutné hodnoty naměřené mikroprocesorem a poslané po sériové lince zaznamenat přímo v počítači. Takovou možnost bohužel sériový monitor integrovaný v *Arduino IDE* neumožňuje (umožňuje pouze manuální zkopírování naměřených hodnot z okna monitoru standardním příkazem *Ctrl+C*). K monitorování sériové linky je na internetu k dispozici

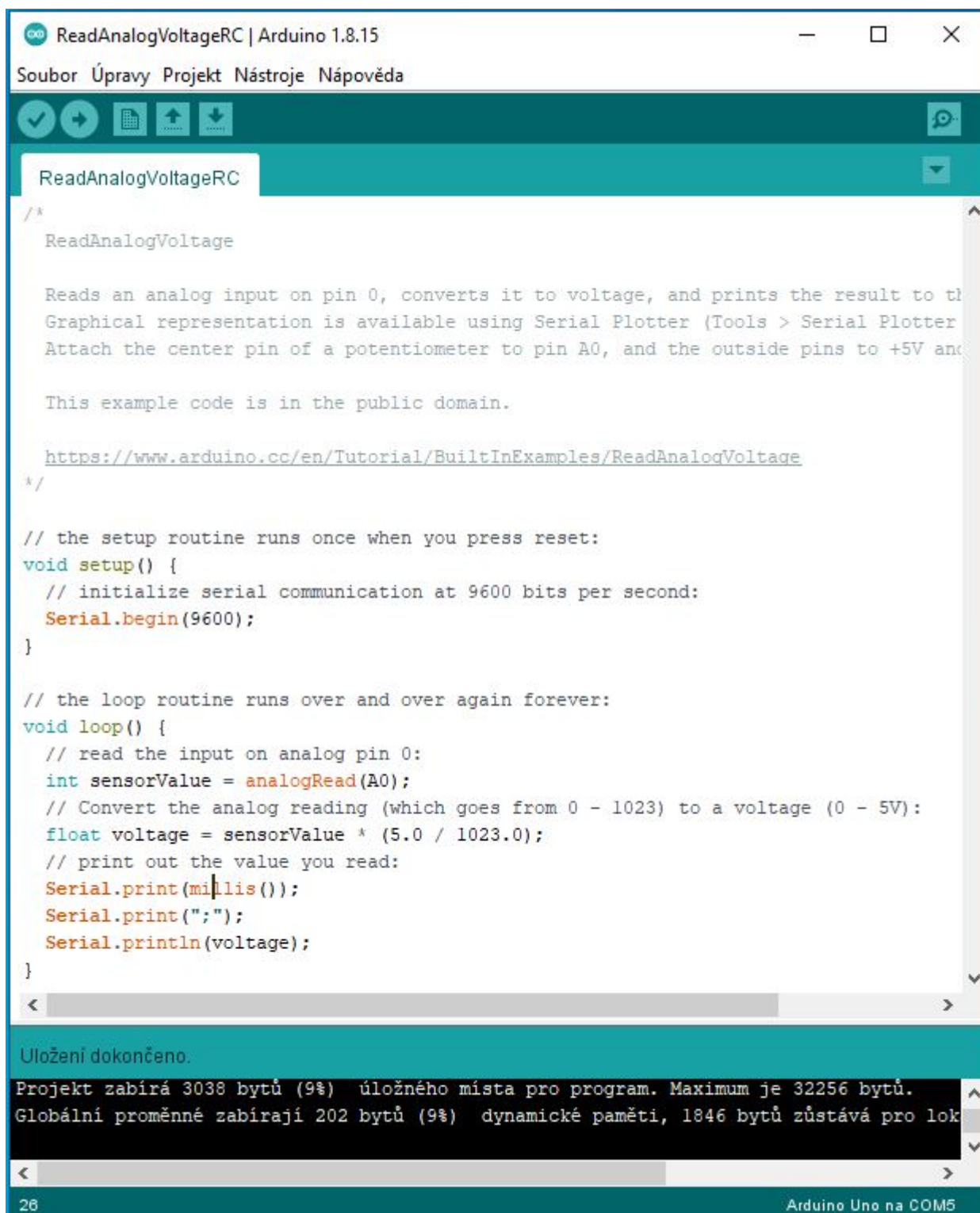
více volně dostupných (freeware) programů. V tomto příkladu si ukážeme použití programu *RealTerm*.

5.2 Program *RealTerm* – monitor sériové linky

Program *RealTerm*  je určen pro monitorování dat posílaných do počítače po sériové lince z externího zařízení (mikroprocesorová deska, voltmetr, LCR metr atd.). Kompletní informace k programu najdete na webu na stránkách <https://realterm.sourceforge.io/>. Instalační soubory najdete na <https://sourceforge.net/projects/realterm/files/> nebo po vyhledání hesla *realterm* download.



Obr.15 Program *ReadAnalogVoltage* se spuštěným sériovým ploterem



```
ReadAnalogVoltageRC | Arduino 1.8.15
Soubor Úpravy Projekt Nástroje nápověda

ReadAnalogVoltageRC

/*
  ReadAnalogVoltage

  Reads an analog input on pin 0, converts it to voltage, and prints the result to the
  Serial Monitor. Graphical representation is available using Serial Plotter (Tools > Serial Plotter)
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and GND.

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/ReadAnalogVoltage
*/

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
  // print out the value you read:
  Serial.print(millis());
  Serial.print(" ");
  Serial.println(voltage);
}

Uložení dokončeno.
Projekt zabírá 3038 bytů (9%) úložného místa pro program. Maximum je 32256 bytů.
Globální proměnné zabírají 202 bytů (9%) dynamické paměti, 1846 bytů zůstává pro lokální proměnné.

26 Arduino Uno na COM5
```

Obr.16 Program *ReadAnalogVoltageRC*

Před spuštěním programu *RealTerm* je nutné uzavřít Arduino IDE, aby se uvolnil přístup k sériovému portu, na kterém je připojena mikroprocesorová deska. Po spuštění programu *RealTerm* je nutné nejdříve v záložce *Port* nastavit parametry komunikace nastavené na vysílající mikroprocesorové desce (obr.17).

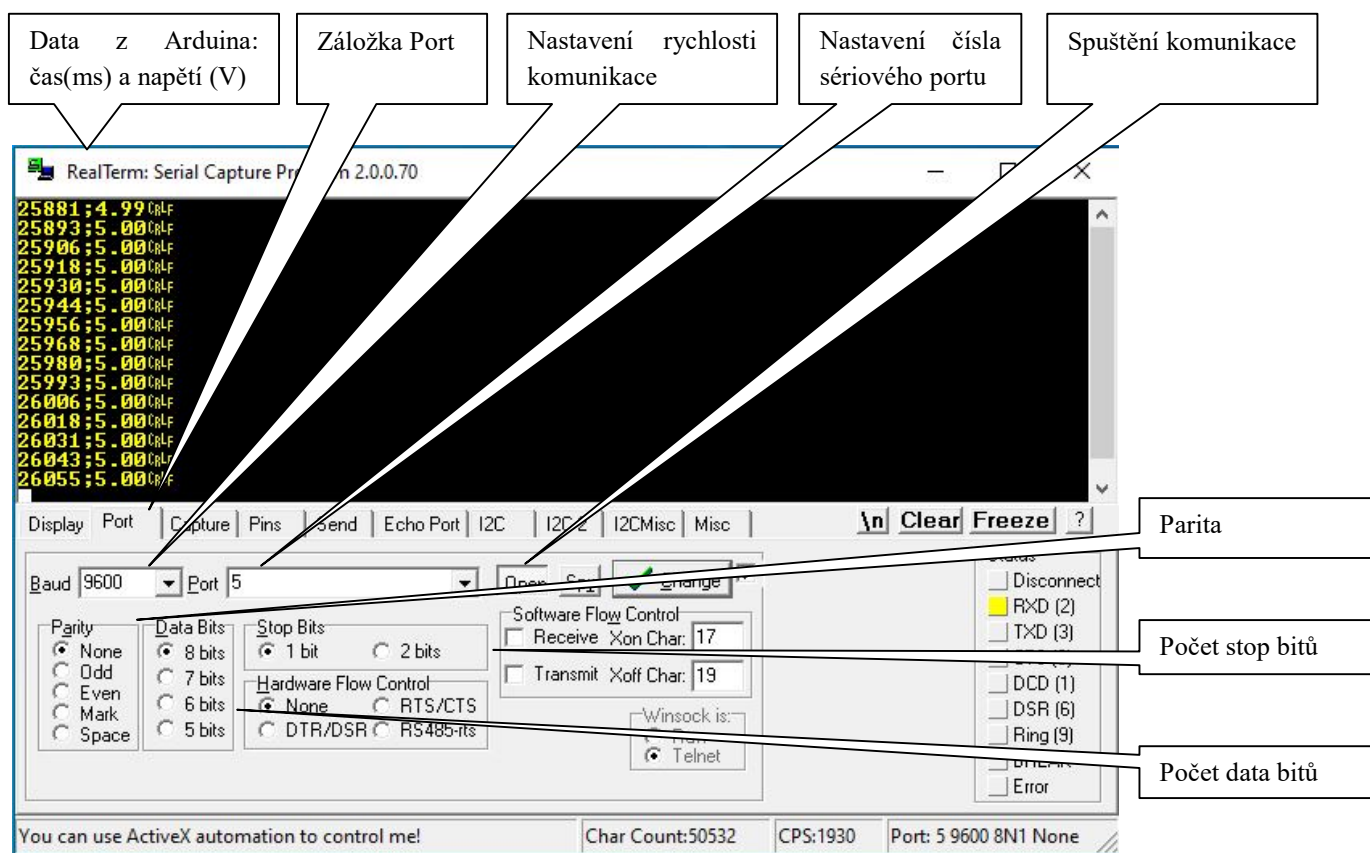
Rychlost komunikace - Baudrate: 9600 byla nastavena na desce Arduina při inicializaci sériové linky příkazem


```
Serial.begin(9600);
```

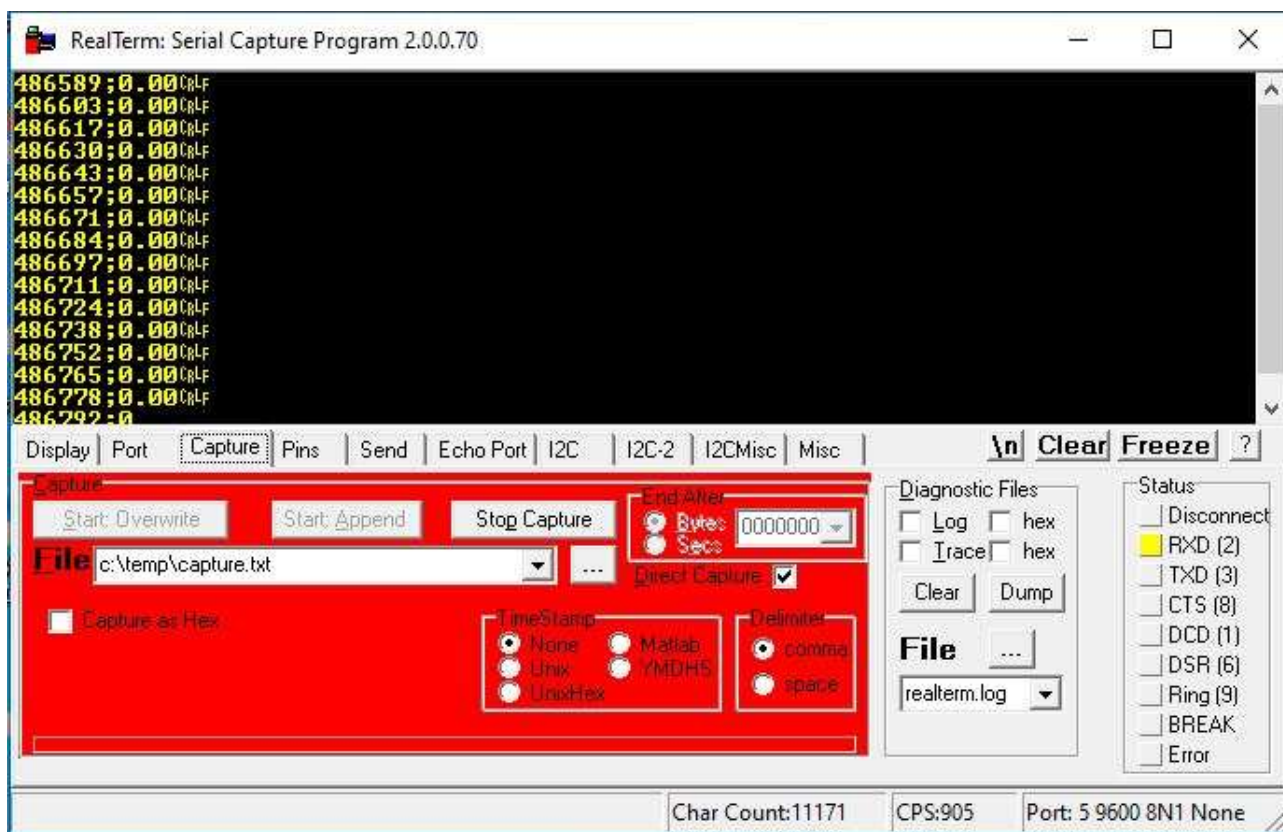
Číslo portu, na který je Arduino připojené zjistíme v *Arduino IDE*, v pravém dolním rohu okna (obr.16, v našem případě COM5) nebo na příkazové liště (*Nástroje-Port: - COM XX*). Ostatní parametry jsou standardní přednastavené parametry sériové komunikace u Arduina. Na obr.17 je ukázka nastavení standardních parametrů komunikace - Baudrate: 9600, Port: 5, Parity: None, Data bits:8, Stop bits:1.

Po nastavení parametrů komunikace a spuštění tlačítkem *Open* se komunikace spustí a v okně se začnou objevovat data vysílaná Arduinem do počítače po sériové lince a naprogramovaná v programu *ReadAnalogVoltageRC*. V našem případě jsou to dvojice čísel čas měření (v milisekundách) a hodnota naměřeného napětí (ve voltech), oddělené středníkem a následované znaky pro ukončení řádky (obr.17).

Program *RealTerm* umožňuje data čtená po sériové lince uložit do souboru. Na záložce *Capture* zadáte jméno a cestu k souboru, kam chcete data ukládat (předdefinován je soubor *Capture.txt* v adresáři *temp* viz obr.18), ukládání spustíte tlačítkem *Start: overwrite*. Ukládání dat ukončíte tlačítkem *Stop capture*. Pokud nastavíte nenulovou hodnotu v kolonce *End after*, ukládání se ukončí po zadaném počtu sekund nebo přečtených bytech. Na obr.19 je ukázka souboru *Capture.txt* s uloženými daty čtenými po sériové lince z Arduina. Tento soubor lze importovat do vybraného software (Excell, Origin, gnuplot) a dále zpracovávat.








Obr.17 Program RealTerm, záložka Port, nastavení parametrů komunikace po sériové lince



Obr.18 Program RealTerm, záložka Capture aktivní

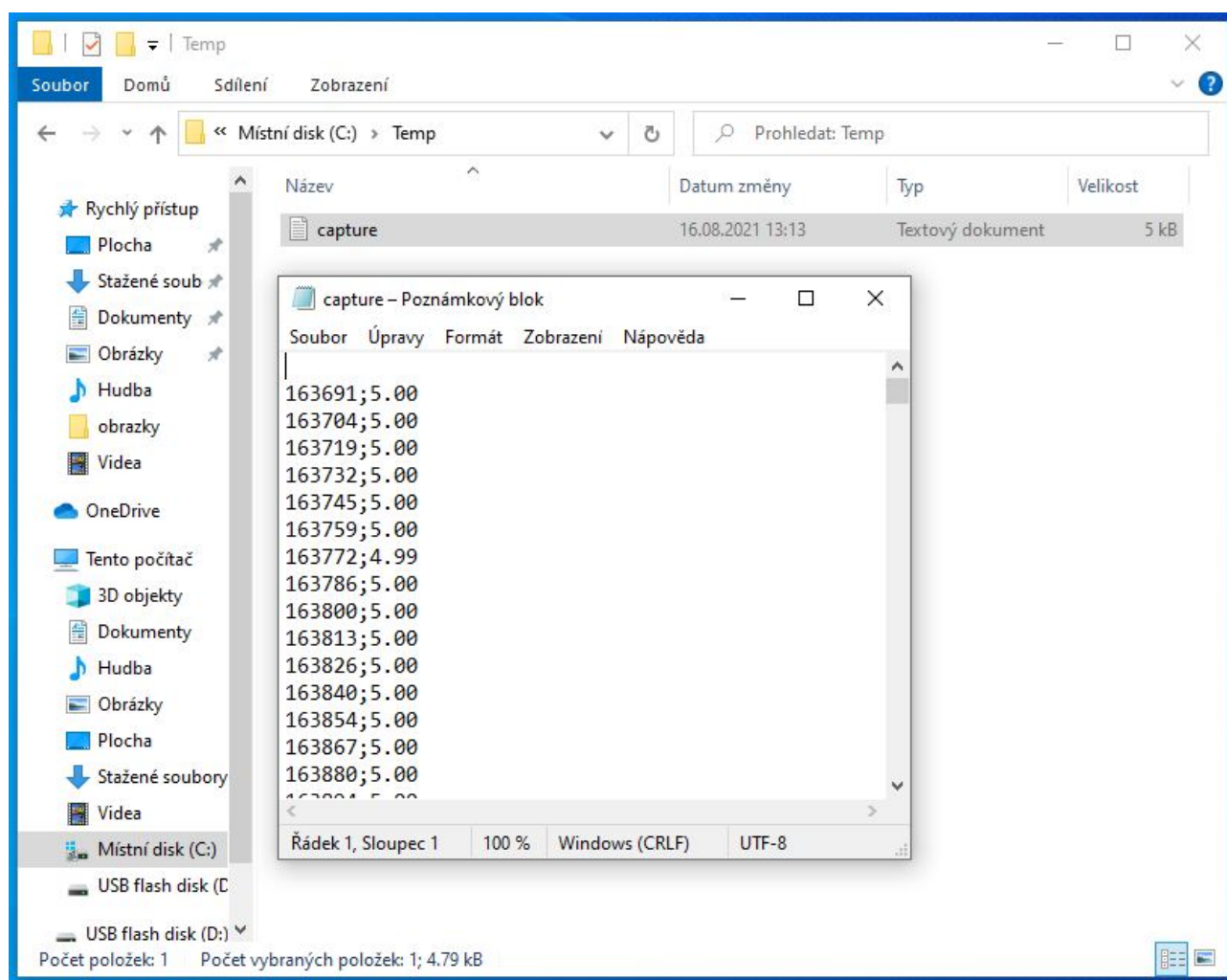
5.3 Postup provedení - měření časové závislosti vybíjení a nabíjení kondenzátoru v sériovém RC obvodu

- 1) Zapojte obvod podle obr.13 a schématu obr.14.
- 2) Otevřete si program *ReadAnalogVoltage*. (Soubor - Příklady - 01.Basic - *ReadAnalogVoltage*, <https://www.arduino.cc/en/Tutorial/BuiltInExamples/ReadAnalogVoltage>, obr.4).
- 3) Zkontrolujte (kompilujte) program, jestli neobsahuje chyby (tlačítkem , Ctrl+R nebo *Projekt – Kontrola/Kompilace* na příkazové liště). Pokud je program bez chyb, objeví se na dolní liště zpráva *Kompilace ukončena* a v oknu zpráv jsou pouze zprávy o projektu a nejsou zde žádné informace o chybách.
- 4) Nahrajte (upload) program do mikroprocesorové jednotky (tlačítkem , Ctrl+U nebo *Projekt – Nahrát* na příkazové liště). Pokud se program správně nahraje, objeví se na dolní liště zpráva *Konec nahrávání* a v oknu zpráv jsou pouze zprávy o projektu a nejsou zde žádné informace o chybách (obr.4).
- 5) Spusťte sériový ploter (Ctrl+Shift+L nebo *Nástroje - sériový plotter* na příkazové liště)
- 6) Přepínejte přepínačem vstupní napětí na sériovém RC obvodu a sledujte na sériovém ploteru vývoj napětí na kondenzátoru na pinu A0.
- 7) Vyměňte odpory a sledujte na sériovém ploteru, jak se změnila časová konstanta vybíjení a nabíjení kondenzátoru při přepnutí přepínače v závislosti na použitém odporu.

- 8) Upravte program *ReadAnalogVoltage* pro posílání času na sériový port (obr.16 - *ReadAnalogVoltageRC*)
- 9) Zkontrolujte (kompilujte) upravený program, jestli neobsahuje chyby (tlačítkem , Ctrl+R nebo *Projekt – Kontrola/Kompilace* na příkazové liště). Pokud je program bez chyb, objeví se na dolní liště zpráva *Kompilace ukončena* a v oknu zpráv jsou pouze zprávy o projektu a nejsou zde žádné informace o chybách.
- 10) Pokud by se Vám nedařilo upravit program bez chyb, otevřete si již upravený program *ReadAnalogVoltageRC* (Ctrl+O nebo *Soubor-Otevřít...*)
- 11) Nahrajte (upload) upravený program do mikroprocesorové jednotky (tlačítkem , Ctrl+U nebo *Projekt – Nahrát* na příkazové liště). Pokud se program správně nahraje, objeví se na dolní liště zpráva *Konec nahrávání* a v oknu zpráv jsou pouze zprávy o projektu a nejsou zde žádné informace o chybách (obr.2).
- 12) Spustíte program *RealTerm*  nastavte v záložce Port parametry sériové komunikace (viz obr.17- Baud rate: 9600, Port: 5, Parity: None, Data bits:8, Stop bits:1)
- 13) Nastavte v programu *RealTerm* v záložce *Capture* položku *End After* na předpokládaný čas záznamu dat 5s (obr.18) a spustíte ukládání dat do souboru tlačítkem *Start:Overwrite*, přepněte přepínačem napětí na RC obvodu. Okno *Capture* během záznamu dat zčervená (obr.18). Ukládání dat můžete i před stanoveným časem ukončit tlačítkem *Stop Capture*. Zpracujte uložená data.

Pokyny k měření:

- Kondenzátor a 5 odporů pro měření jsou v sáčku označeném *RC obvod*
- Odporů i kondenzátor jsou popsány číslem určujícím jejich nominální velikost
Odporů 1k = 1kΩ, 22k = 22kΩ, 53k = 53kΩ, M1 = 100kΩ, M23 = 230kΩ
Kondenzátor 3μ3 = 3,3μF
- Při měření doma nemáte k dispozici ohmmetr, jako hodnotu velikosti odporů a kapacity vezměte jejich nominální hodnoty
- Chyba odporů je 0,5%, chyba kapacity kondenzátoru je 5%



Obr.19 Program RealTerm, ukázka souboru Capture.txt s uloženými daty naměřenými Arduinem

Závěr

Laboratorní úloha „Arduino ve fyzikální laboratoři“ Vám měla přiblížit moderní měřicí metody s využitím počítače. Cílem úlohy je pochopit, jak se získávají (digitalizují) měřené hodnoty z experimentů, co je to analogově digitální převodník, analogová či digitální data, aj. Využili jsme snadno dostupný měřicí nástroj Arduino. Tento a podobné nástroje jsou univerzálně použitelné i při jiných měřeních. V úloze jste se seznámili s přenosem dat z experimentu do počítače, odzkoušeli jste si sériový přenos dat (Sériový monitor, resp. samostatný program RealTerm).

Pozn.: Pokud Vás práce s Arduinem zaujala, můžete si odzkoušet další zapojení, která naleznete v naší rozšířené sadě senzorů připojitelných k Arduino. Případně si poříďte vlastní Arduino s mnoha dalšími senzory (odkaz na Hwkitchen, dratek.cz, laskarduino.cz aj.).

https://www.laskarduino.cz/user/related_files/laskkit_arduino_maxi_starter_kit_v1_4-1.pdf

Literatura:

[1] – Co je Arduino ? [online]. [cit. 2021-10-21]. Dostupné z:
<https://www.arduino.cc/en/Guide/Introduction>

[2] – Průvodce světem Arduina [online]. [cit. 2021-10-21]. Dostupné z:
<https://bastlirna.hwkitchen.cz/>

[3] – Přechodové jevy v RCL obvodu - studijní text k úloze 18, ZFP -Fyzikální praktikum II

[4] – Charakteristiky diod - studijní text k úloze 11, ZFP -Fyzikální praktikum II