SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

**Predmet: Pokročilé informačné technológie   
Záverečné zadanie:**

**Ovládanie RC obvodov**

**Vypracoval:** Samuel Gomolčák

**Ročník:** 1. Ing.

**Odbor a študijná skupina:** AM

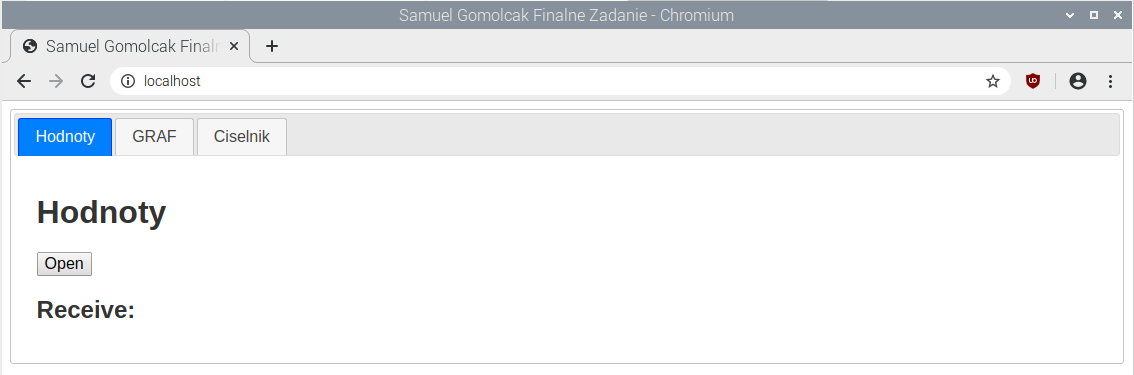
**Technická dokumentácia**

1. **Používateľská príručka**
   1. **Popis programu**

Program ovláda ustálenie signálov RC obvodov 2. rádu na požadovanú hodnotu napätia *[U]*. Tieto obvody sú realizované na platforme *Arduino* resp. na jeho virtuálnom simulátore. *Raspberry Pi* s *Arduinom* komunikuje cez sériové porty, ktoré sú umelo vytvorené kvôli ich virtuálnym verziám. Následne tieto signály z vysielajúceho *Arduina* aplikácia monitoruje a spracováva. Neposlednou časťou je vytvorenie serveru a klienta, ktorými máme možnosť naše dáta prijímať a regulovať.

* 1. **Popis používateľského rozhrania**

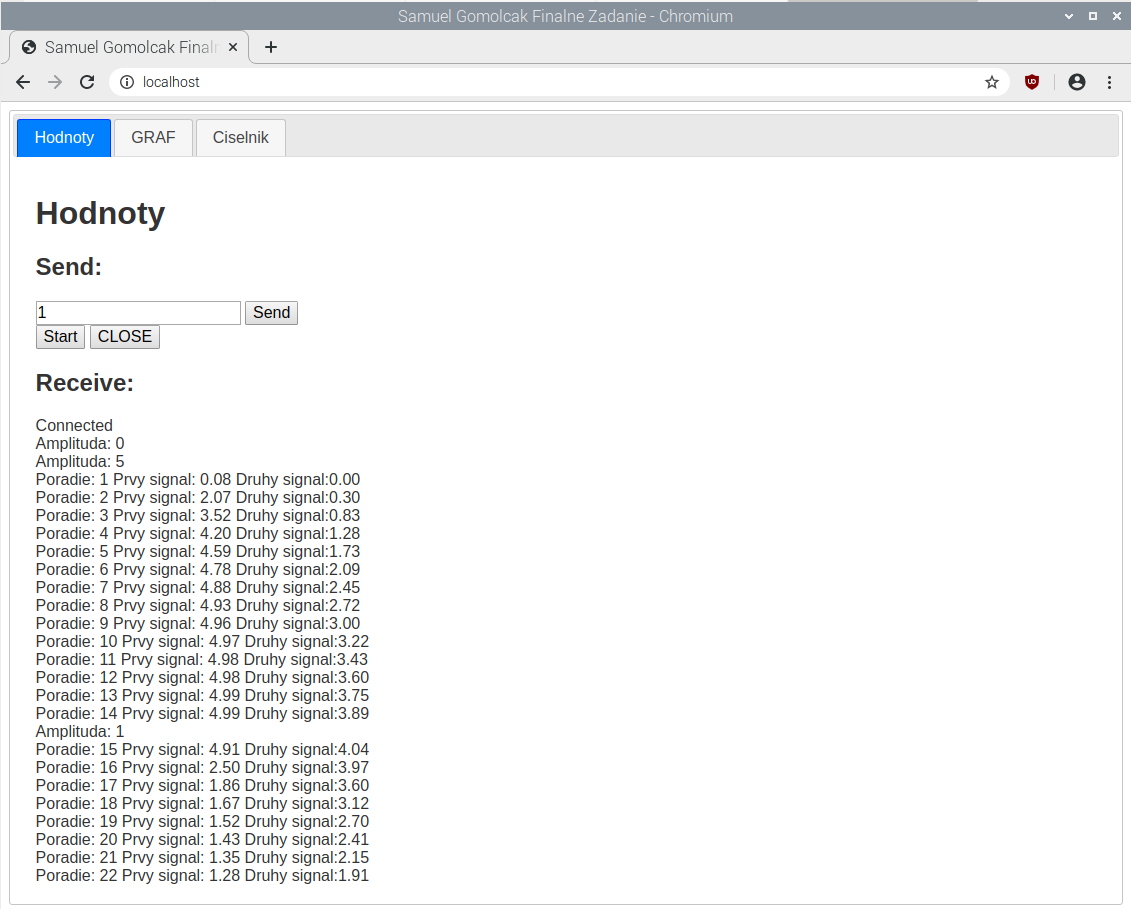
Po spustení klienta (web stránky) sa zobrazí jednoduché rozhranie *Obr.1* s troma rôznymi záložkami – Hodnoty, GRAF, Číselník.



*Obr.1: Ukážka web klienta*

* **Záložka Hodnoty**

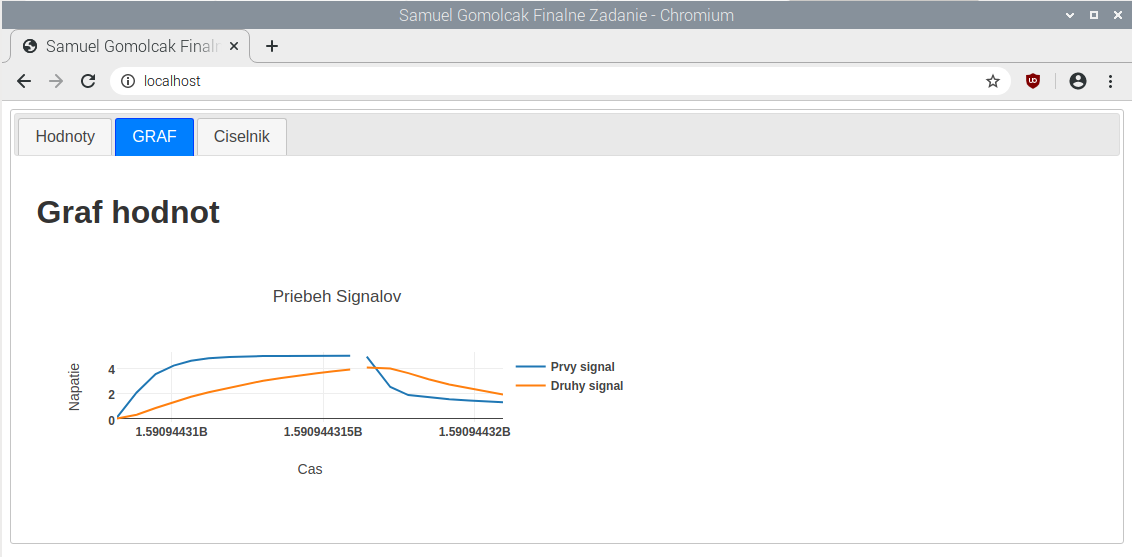
Nasledujúca záložka je jadrom našej aplikácie. Po kliknutí na tlačidlo *Open* sa náš systém inicializuje, pripojí a pripraví na chod celého programu. Zobrazí sa pole na zadanie celočíselnej hodnoty napätia (*1-5[V]),* na ktoré sa signály budú ustaľovať a tlačidlo *Send,* ktorým túto hodnotu odošle na *Arduino*. Ďalej sa objavia 2 nasledovné tlačidlá *Start* a *CLOSE*. Ako posledné je vidieť hodnotu amplitúdy, ktorá je na hodnote *0* a potvrdzujúci text *Connected* slúžiaci na overenie spojenia. Po zakliknutí tlačidla *Start*, môžeme zadať prvú hodnotu napätia. Po odoslaní tejto hodnoty začne regulácia na príslušné napätie, monitorovanie daných signálov a ich výpis na obrazovke. Počas behu programu, môžeme hodnotu ustálenia meniť, čím sa signály opäť zregulujú. Tlačidlo *CLOSE* slúži na deaktiváciu systému a ukončenie spojenia sprevádzané upozorňovacím textom *Disconnected*.



*Obr.2: Príklad použitia programu záložka Hodnoty*

* **Záložka Graf**

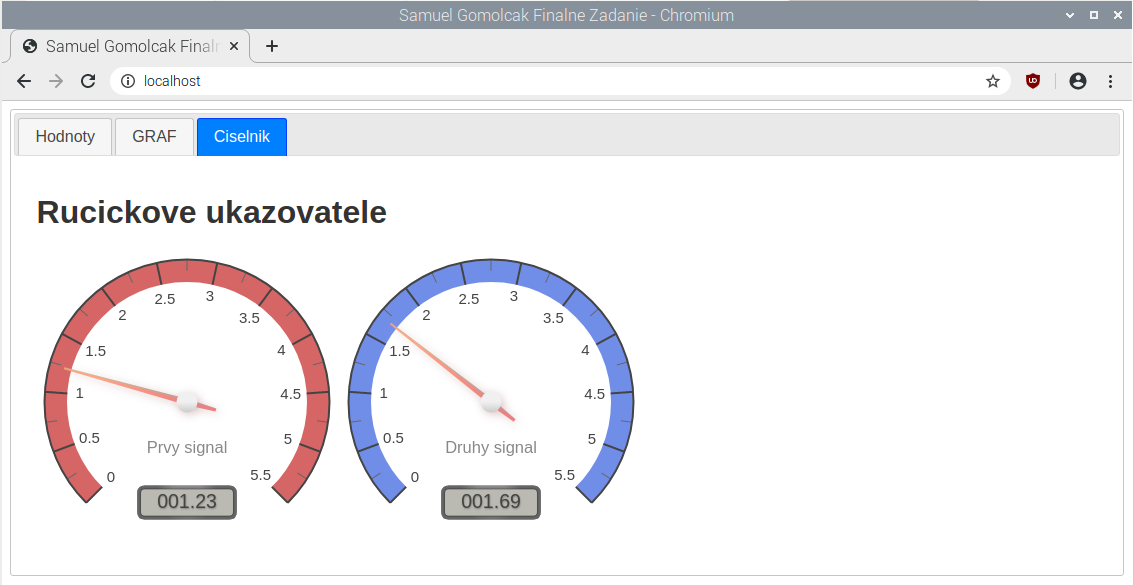
Záložka graf slúži na grafické zobrazenie prijímaných hodnôt (signálov). Vykresľujú sa tak dva rozličné priebehy závislé od času.



*Obr.3: Príklad použitia programu záložka Graf*

* **Záložka Číselník**

Záložka číselník, taktiež slúži na grafické vyobrazenie signálov, no za pomoci ručičkových ukazovateľov (ciferníkov).



*Obr.4: Príklad použitia programu záložka Číselník*

* 1. **Obmedzenia a chyby programu**

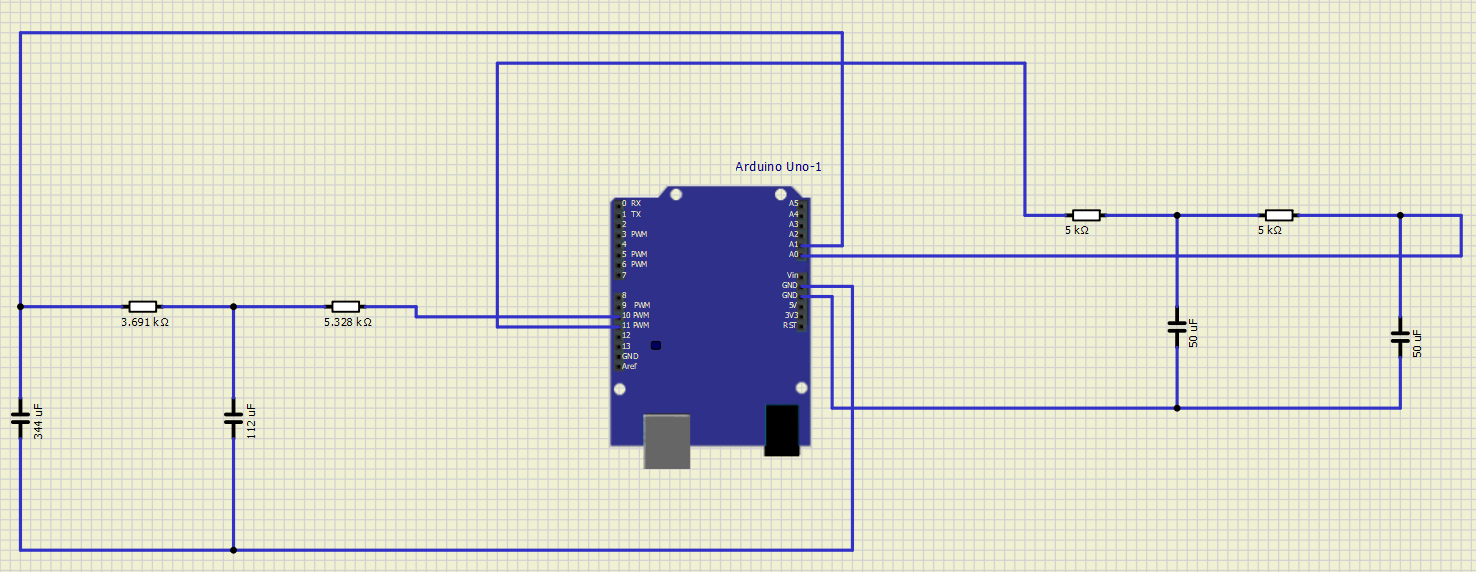
Program je stavaný iba na celočíselné hodnoty ustáleného napätia, a to kvôli sériovej komunikácií, ktorá nedokáže zapisovať číselné hodnoty ale iba reťazce (*string*). Chybou aplikácie je občasné vynechanie zapisovaných dát, najčastejšie pri zmene hodnoty ustálenia *Obr.3.* Je to spôsobené opäť chybou v sériovej komunikácii a používania nelicencovaného softwaru simulátora *Arduina*. O týchto chybách a obmedzeniach bude spomenuté viac vo vývojárskej príručke.

1. **Vývojárska príručka**
   1. **Špecifikácia požiadaviek**

Hlavnou požiadavkou aplikácie je spracovávať posielané dáta z *Arduina* resp. simulátora *Arduina* cez sériovú komunikáciu na *Raspberry Pi* resp. virtuálneho *Raspberry Pi* a opačne. Následne vytvoriť server a web klienta, ktoré budú naším používateľským rozhraním pre prácu s dátami. Požadovanou výslednou aplikáciou je vytvorenie web klienta, ktorým budeme dáta numericky aj graficky spracovávať. Potom ako doňho zadáme požadovanú hodnotu ustáleného napätia, ktorá sa odošle na server a teda *Raspberry Pi*, ktoré následne pošle tuto hodnotu *Arduinu* na ktorom sa nachádzajú simulované elektrické obvody. Spracovávane dáta na web klientovi sú následne výstupy z týchto obvodov posielane na *Raspberry* a jeho server.

* 1. **Arduino**
* **Hardware**

Pre aplikáciu sa požívala náhrada reálneho hardwaru a teda virtuálne *Arduino* bolo stvárnené cez simulačný software *SimulIDE* . Jednou z požiadaviek bolo spracovávať viac ako jeden signál, preto v simulácii boli vytvorené dva rozličné RC obvody druhého rádu. V požiadavkách je taktiež riadenie signálov pre ktoré bol navrhnutý PID regulátor ktorého, výstup resp. žiadanú hodnotu na výstupe riadime premennou prijímanou cez sériovú komunikáciu z *Raspberry*. Pre obe RC obvody boli zvolené rozličné hodnoty komponentov (veľkosť odporu *R* a kapacity *C*) s použitím rovnakého regulátora. Môžeme tak sledovať vplyv rozličných systémov na rovnaký regulátor. V obvode bolo použité *Arduino Uno*. Pre vstupy simulovaných RC obvodov boli použité digitálne PWM piny 10, 11 a pre sledovanie výstupov analógové piny A0 , A1.



*Obr.5: Elektrický obvod v simulátore SimulIDE*

* **Software**

Pre „hardware“ *Arduino Uno* bol v simulátore taktiež vytvorený program. Ako prvé boli zadefinované používane premenné. Ďalej bola vytvorená funkcia PID, ktorá plní úlohu bežného PID regulátora. V *setupe* sme nastavili digitálne piny 10 a 11 ako výstupy teda posielajú signál z *Arduina* „von“ do obvodu (napájanie obvodov). V cykle *loop* ďalej prebieha náš program. Ako prvé v tomto cykle vidíme načítavanie posielaných hodnôt z *Raspberry Pi.* Do premennej sa posiela neustále pomocná hodnota *9,* ktorá bude vysvetlená neskôr nižšie v kóde. Následne máme pod touto premennou podmienku, ak zmení svoj stav na inú hodnotu ako *9* a zároveň nie je menšia ako nula, zapíše tuto novú hodnotu na naše požadované ustálenie. Túto hodnotu posielame my z *Raspberry* resp. webového klienta, a tak dokážeme meniť veľkosť ustáleného napätia signálov. Ďalej sa vykonáva spomínaná regulácia RC obvodov a ako posledné sa zapisujú a posielajú cez sériovú komunikáciu hodnoty výstupov spoločne so špecifickým znakom, ktorý bude vysvetlený v časti kódu pre *Raspberry*. Výpis je však podmienený práve pomocnou hodnotou *9*, ktorá sa posiela z *Raspberry* každých 0.5 sekúnd. To znamená že *Arduino* vypíše a pošle výstupy iba dvakrát za sekundu a tak zabezpečíme nepreplňovanie buffera a aj *Arduino* môže pracovať s nízkym oneskorením (*delayom*). Ak by sme zvýšili *delay,* RC obvody by začali byť nestabilné a ťažko regulovateľné.

*//deklarácia používaných premenných*

*char c; // čítaná hodnota z Raspberry*

*int PWMpin\_RC = 11; //zápis dig. pinu prveho obvodu*

*int PWMpin\_RC\_1 = 10; //zápis dig. pinu druheho obvodu*

*float U\_RC, U\_RC\_1, i, x\_RC, y\_RC\_1; // výstupy*

*//premenné pre PID*

*unsigned long currentTime, previousTime;*

*double elapsedTime;*

*float error,last,cum, rate;*

*float sp1; // požadovaná hodnota ustálenia*

*float output\_RC,output\_RC\_1;*

*//parametre PID regulátora*

*float P = 13;*

*float I = 0.0066;*

*float D = 5;*

*//PID regulátor*

*float PID(float inp,float Setpoint){*

*currentTime = millis();*

*elapsedTime = (double)(currentTime - previousTime);*

*error = Setpoint - inp;*

*cum += error \* elapsedTime;*

*rate = (error - last)/elapsedTime;*

*float out = P\*error + I\*cum + D\*rate;*

*last = error;*

*previousTime = currentTime;*

*return out;*

*}*

*void setup() {*

*pinMode(PWMpin\_RC, OUTPUT);*

*pinMode(PWMpin\_RC\_1, OUTPUT);*

*Serial.begin(9600);*

*}*

*void loop() {*

*c = Serial.read(); // čítanie hodnoty z Raspberry*

*if((c!=57)&&(float(c)>0)) { //podmienka ktorou zapisujeme požadovaný výstup*

*sp1 = float(c);*

*sp1=sp1-48; //prepočet z ASCII na reálne číslo*

*}*

*//prvý RC obvod*

*U\_RC=analogRead(A0);*

*output\_RC = PID(U\_RC\*0.004882, sp1);*

*if (output\_RC>5){*

*output\_RC=5;*

*}*

*if (output\_RC<0){*

*output\_RC=0;*

*}*

*analogWrite(PWMpin\_RC,(output\_RC)\*51);*

*x\_RC=U\_RC\*0.004882; //vystup prvého RC obvodu*

*//druhý RC obvod*

*U\_RC\_1=analogRead(A1);*

*output\_RC\_1 = PID(U\_RC\_1\*0.004882, sp1);*

*if (output\_RC\_1>5){*

*output\_RC\_1=5;*

*}*

*if (output\_RC\_1<0){*

*output\_RC\_1=0;*

*}*

*analogWrite(PWMpin\_RC\_1,(output\_RC\_1)\*51);*

*y\_RC\_1=U\_RC\_1\*0.004882; //vystup prvého RC obvodu*

*//zápis výstupov*

*if ((c == 57)&&(x\_RC>0)) {*

*Serial.print(x\_RC);*

*Serial.print(",");*

*Serial.println(y\_RC\_1);*

*}*

*delay(50); //oneskorenie 0.05sekundy*

*}*

* 1. **Raspberry Pi**

Tak ako aj v predošlom prípade, je pre aplikáciu použité virtuálne *Raspberry* a to v softwari *Oracle VM VirtualBox*. Požiadavkou je práca s dátami medzi zariadeniami, čo znamená odosielanie a prijímanie dát z *Arduina*. Dáta následne ukladá do databázy a do textového súboru. Virtuálne *Raspberry* ďalej slúži na prevádzku servera a web klienta.

* **Software**

V cykle *while* sa odohráva základná časť nášho programu. Ako prvé je nastavenie oneskorenia na 0.5 sekundy. To nám zaručí nepreplňovanie buffera a hladký priebeh. Ďalší krok je načítavanie premennej z web klienta. Toto načítavanie je podmienené tak, aby zapísal číslo (požadovanú hodnotu ustálenia) posielané z web klienta do premennej vtedy, ak mu ho zadáme. Podmienka slúži aj na predídenie chýb, ak sa do premennej práve nič neposiela. Premenná by inak čítala hodnotu *NoneType*. Do nasledujúcej premennej sa načítava hodnota, ktorú meníme zatlačením tlačidla. Po jeho kliknutí sa hodnota zmení na *start*, ktorá sa využíva v ďalšej podmienke, čo slúži na odštartovanie aplikácie práve daným tlačidlom. Pred tým máme ešte jednu podmienku, aby načítavaná hodnota požadovaného ustálenia bola celočíselná od *1* po *5*. Je to z dôvodu obmedzenia „hardwaru“ *Arduina*, ktoré pracuje s napätím maximálne *5[V]* a toho, že po sériovej komunikácii vieme odosielať len hodnotu *string*, ktorá sa dá jednoducho v *Arduine* prerobiť na celočíselný *integer* alebo *float* (po požutí funkcie *int()* alebo *float()* dostávame ASCII hodnotu, ktorá sa prepočíta na reálnu), avšak neceločíselná hodnota sa prerobiť v *Arduine* na *float* dá len veľmi ťažko. Nasleduje ďalšia podmienka, ktorá zaručuje, že môžeme meniť hodnotu požadovaného ustálenia v čase prebiehania programu bez toho, aby sme narušili odosielanie pomocnej hodnoty *9* a to tak, že zbehne iba vtedy, ak hodnotu želaného ustálenia zmeníme na inú ako predošlú. Ďalej nasleduje ukladanie času, zapisovanie pomocnej hodnoty *9,* načítavanie dát posielaných z *Arduina* a počítadlá, ktoré bežia v cykle každých 0.5 sekundy za splnených vyššie spomenutých podmienok. Do premennej *data*, kde sa načítavajú posielané dáta sa taktiež (rovnako ako z *Raspberry* do *Arduina*) posiela hodnota *string*. V nasledujúcej premennej *values* sa táto hodnota rozdelí na dve samostatné polia (*values[0]* a *values[1]*) príkazom *split*, ktorý reaguje na špecifický znak a rozdeľuje tieto hodnoty v posielanom *stringu*. Ako posledné v tomto podmienenom cykle prebieha zápis do databázy a do textového súboru. Samozrejme program v *Raspberry* vykonáva viacero funkcií ako aj spustenie samotného servera, avšak pre vysvetlenie funkčnosti aplikácie postačuje jadro tohto programu.

***while True****:*

*socketio.sleep(0.5)*

***if*** *dict(args).get('A')* ***is not None****:*

*A = dict(args).get('A')*

*A = int(A)*

*dbV = dict(args).get('db\_value')*

***if*** *A==1 or A==2 or A==3 or A==4 or A==5:*

***if*** *dbV == 'start':*

***if*** *i==0* ***or*** *i!=A:*

*ser.write(str(A))*

*i=A*

*print(A)*

*cas = time.time()*

*ser.write(str(int(9)))*

*data = ser.readline()*

*values = data.split(',')*

*#print(values[0])*

*#print(values[1])*

*count += 1*

*dataCounter +=1*

*#zapis dat*

*dataDict = {*

*"t": time.time(),*

*"x": dataCounter,*

*"y": values[0],*

*"z": values[1],*

*}*

*dataList.append(dataDict)*

***if*** *len(dataList)>0:*

*#print (str(dataList))*

*fuj = str(dataList).replace("'", "\"")*

*#print fuj*

*textak = open("static/files/data.txt","a+")*

*textak.write("%s\r\n" %fuj)*

*cursor = db.cursor()*

*cursor.execute("SELECT count(id) FROM graph")*

*maxid = cursor.fetchone()*

*cursor.execute("INSERT INTO graph (hodnoty) VALUES ('%s')"%(fuj))*

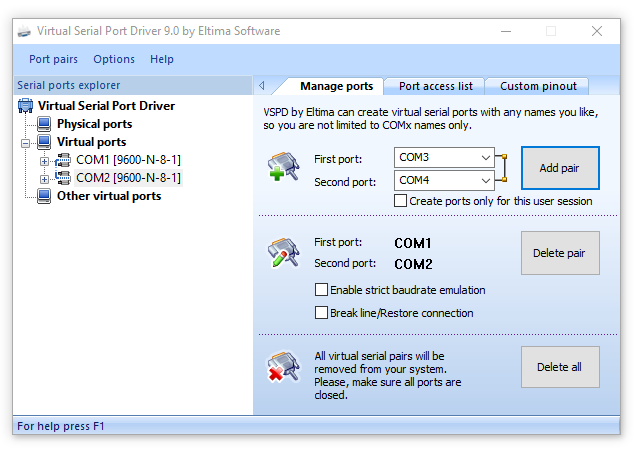
*db.commit()*

*dataList = []*

*dataCounter = 0*

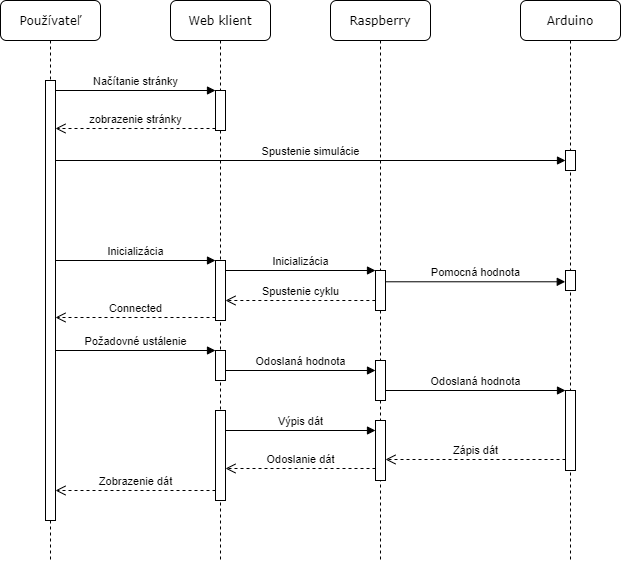
* 1. **Sériová komunikácia**

Ako už bolo spomenuté zaradenia musia medzi sebou dáta prijímať a posielať, teda musí byť zabezpečená komunikácia. Táto komunikácia bola zavedená sériovo cez porty v počítači. Keďže sa jedná o virtuálne zariadenia, porty museli byť umelo vytvorené v softwari *Virtual Serial Port Driver.* Program nám porty nie len vytvorí, ale taktiež zabezpečí ich prepojenie tak, aby mohli medzi sebou komunikovať. Pre *Arduino* bol použitý virtuálny port *COM1* a pre virtuálny *Raspberry* port *COM2*.



*Obr.6: Ukážka sofwaru Virtual Serial Port Driver s virtuálnymi portmi*

* 1. **UML**

****

*Obr.7: UML Sekvenčný diagram*

1. **Záver**

Zadaním bolo navrhnúť aplikáciu, ktorou budeme riadiť a monitorovať signály na RC obvodoch vo virtuálnom *Arduine* cez sériovú komunikáciu s *Raspberry Pi.* Pre danú aplikáciu bol vytvorený server a jeho webový klient, kde spomínané obvody ovládame a jeho signály následne monitorujeme. Webový klient má tri rozličné sledovania primaných signálov. Jedno numerické a dve grafické pomocou grafu a ručičkových ukazovateľov. Hodnoty signálov sa tiež ukladajú do databázy a textového súboru. Pre virtuálne *Arduino UNO* boli vytvorené simulačné RC obvody 2. rádu a následne navrhnutý program. Po spustení aplikácie cez webového klienta dokážeme meniť hodnotu ustáleného napätia na týchto obvodoch, teda je splnená podmienka ovládania signálov. Software použitý pre virtuálne *Arduino* nie je licencovaný a nefunguje vždy správne. Vznikajú tak chyby, napríklad pri sériovej komunikácii, avšak sú stále akceptovateľné. Počas procesu vývoja aplikácie bol používaný verzionovací systém *GitHub*, ktorý má 17 postupných verzií. Pre nazretie do repozitáru *GitHubu* je priložený link. Následne bola pre aplikáciu spísaná technická dokumentácia, ktorá obsahuje užívateľskú a vývojársku príručku. Link pre všetky súbory uložené na *Google Drive* je priložený nižšie.

Link GitHub: <https://github.com/samuelgomolcak/final>

Link Google Drive: