SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

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

**Ovládanie RC obvodov**

**Vypracoval:** Samuel Gomolčák

**Ročník:** 1. Ing.

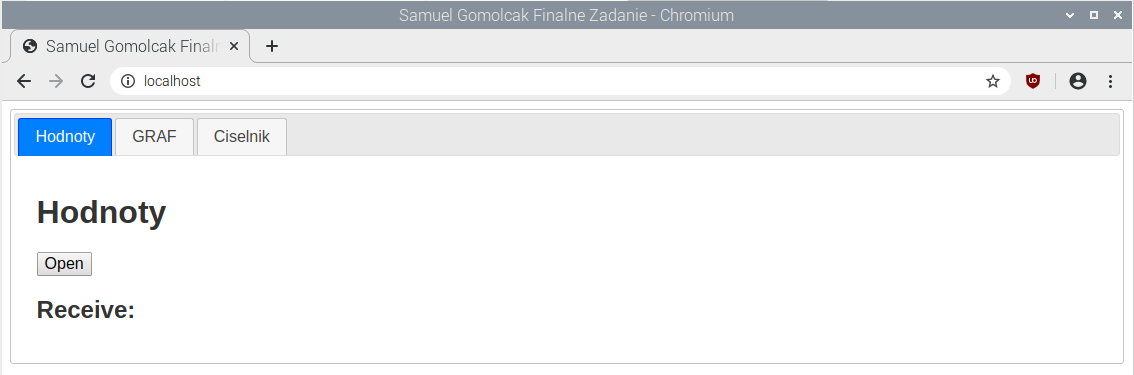
**Odbor a študijná skupina:** AM

1. **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 častou je vytvorenie serveru a klienta ktorými máme možnosť naše dáta prijímať a regulovať.

* + 1. **Popis používateľského rozhrania**

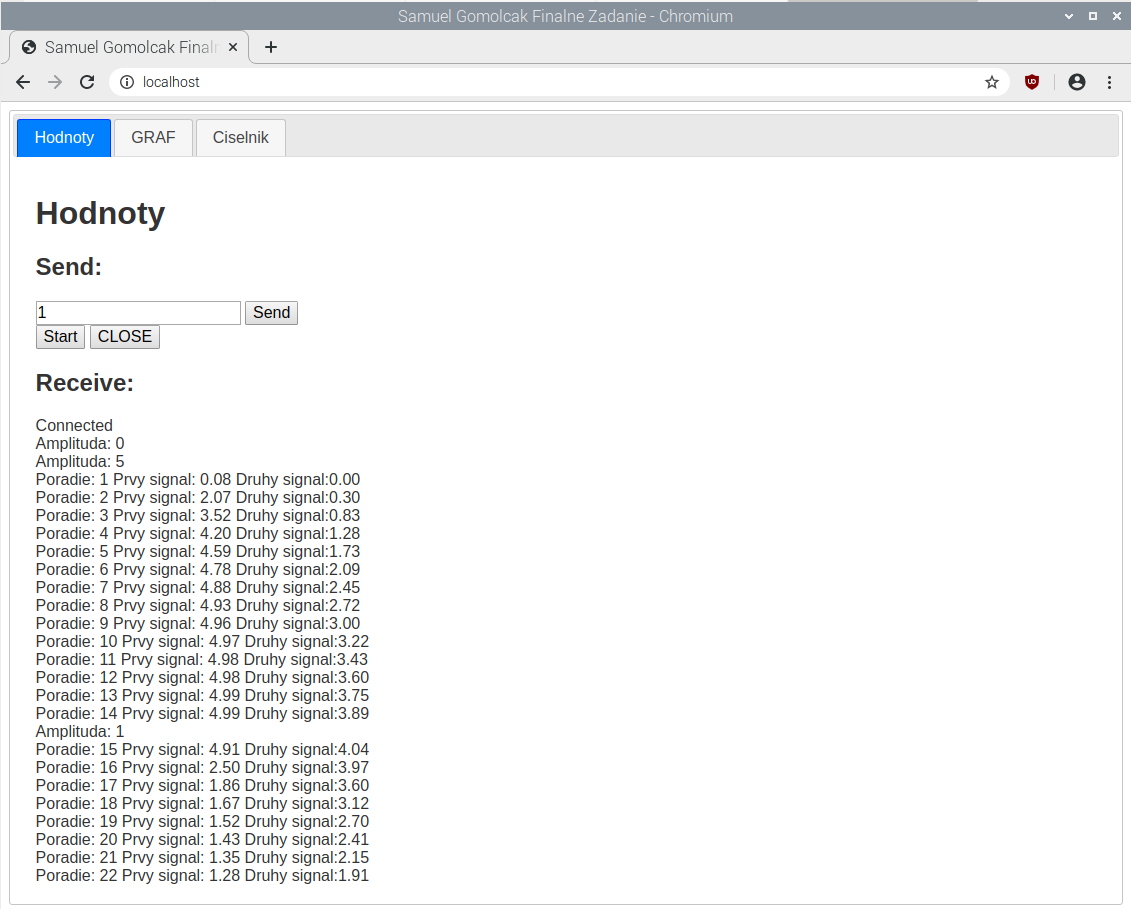
Po spustení klienta (web stránky) sa zobrazí jednoduché rozhranie *Obr.1* s troma rôznymi záložkami – Hodnoty, GRAF, Číselník.



*Obr.1: Ukážka web klienta*

* **Záložka Hodnoty**

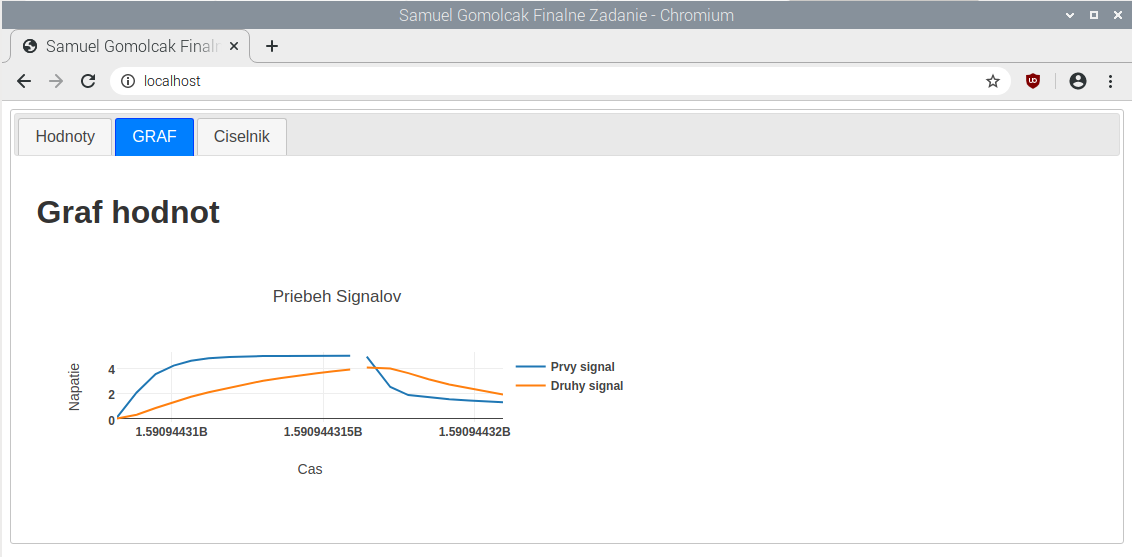
Nasledujúca záložka je jadrom našej aplikácie. Po kliknutí na tlačidlo *Open* sa náš systém inicializuje, pripojí a pripraví na chod celého programu. Zobrazí sa pole na zadanie celočíselnej hodnoty napätia *1-5[V]* na ktoré sa budú signály ustaľovať a tlačidlo *Send* ktorým túto hodnotu odošle na Arduino .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**

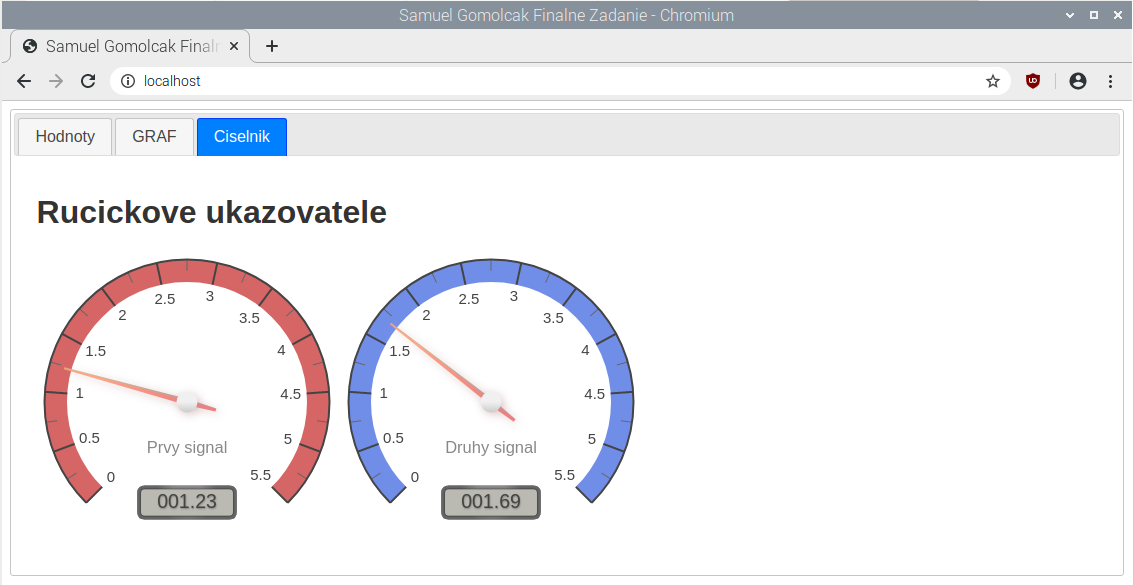
Záložka graf slúži na grafické zobrazenie primaných hodnôt (signálov). Vykresľujú sa tak 2 rozličné priebehy závislé od času.



*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**

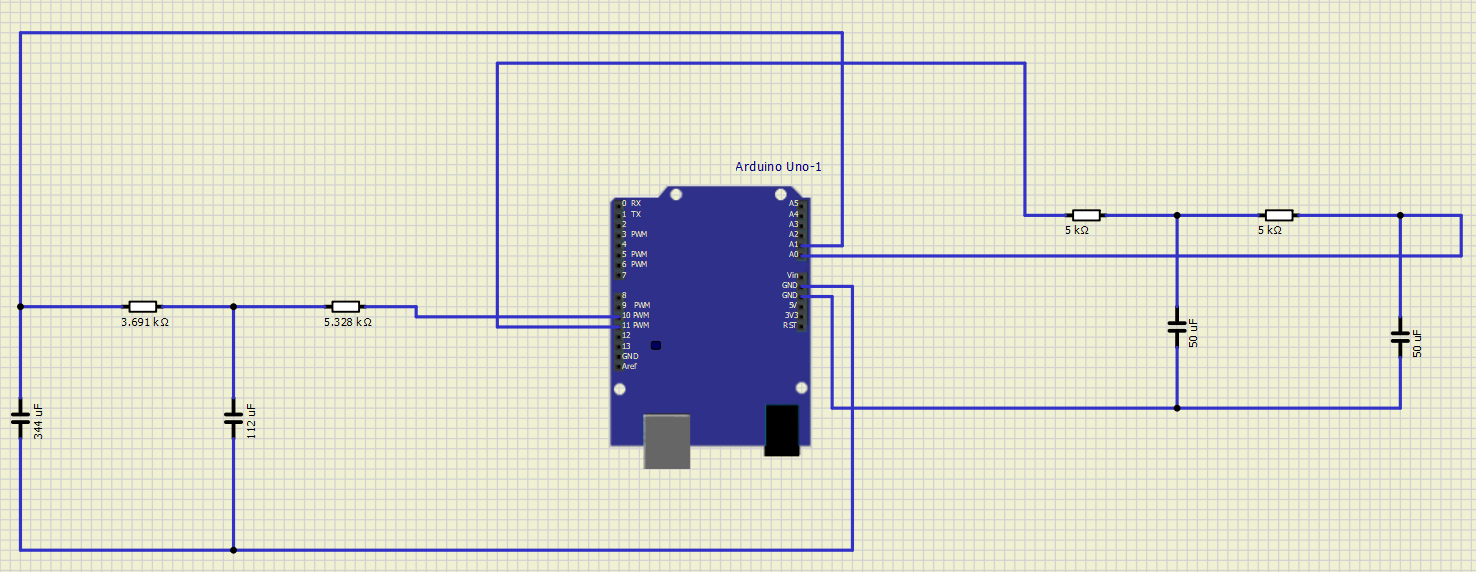
Program je stavaný iba na celočíselné hodnoty ustáleného napätia a to kvôli sériovej komunikácií ktorá nedokáže zapisovať číselné hodnoty ale iba reťazce (*string*). Chybou aplikácie je občasné vynechanie zapisovaných dát, najčastejšie pri zmene hodnoty ustálenia *Obr.3.* Je to spôsobené opäť chybou 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 aplkáciou je vytvorenie web klineta ktorým budeme dáta numericky aj graficky spraovávať potom ako donho zadame požadovanu hodnotu ustaleneho napatia ktorá sa odošle na server a teda Raspberry Pi, ktoré následne pošle tuto hodnotu Arduinu na ktorom sa nachádzajú simulované elektrické obvody. Spracuvovane data na web klientovy su násldne výstupy z týchto obvdovov posielane na rasbery a jeho server.

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

Pre aplikáciu sa požívala náhrada realneho hardweru a teda virtálne *Arduino* bolo ztvárnené cez simulačný software *SimulIDE* . Jednou z požiadaviek bolo spracovávať viac ako jeden signál, preto v simulacíí boli vytvorené dve RC obvody druhého rádu. Taktiež v požiadvkách je riadenie signálov pre ktoré bol navrhnutý PID regulátor ktorého výstup resp. žiadanú hodnotu na výstupe riadime premennov prímanou cez sériovu komunikáciu z *Raspberry Pi*. Pre obe RC obvody boli zvolené rozličné hodnoty komponetov (velkosť odporu *R* a kapacity *C*) s použitím rovnakého regulátora. Možeme tak sledovaanie vlyvu rozličných systémov na rovnaký reglátor. V obvode bolo použité *Arduino Uno*. Pre vstupy simulovaných RC obvodou boli použité digitálne PWM piny 10, 11 a pre sledovanie výstupov analogové piny A0 , A1.



*Obr.5: Elektrický obvod v simulátore SimulIDE*

* **Software**

Pre hardware *Arduino Uno* bol vytvorený program. Ako prvé boli zadefinované používane premenné. Dalej bola vytvorená funkcia PID ktorá plní ulohu bežného PID regulátora. V setupe sme nastavili digitálne pini 10 a 11 ako výstupy teda posielajú signál z *Arduina* „von“ do obvodu (napájanie obvodov). V cykle loop dalej priebeha náš program. Ako prvé v tomto cykle vidime načítavanie posielných hodnot z *Raspberry Pi.* Do premennej sa posiela neustále pomocná hodnota *9* ktorá bude vysvetlená neskor nižšie v kode. Následne máme pod touto premennou podmienku ak zmení svoj stav na inu hodnotu ako *9* a zaroven nieje manšia ako nula zapíše tuto novu hodnotu na naše požadované ustalenie. Tuto hodnotu posielame my z Raspberry resp. weboveho klienta a tak dokážeme meniť velkosť ustaleneho napatia signálov. Ďalej sa vykonáva spominaná regulácia RC obvodov a ako posledné sa zapisuju a posielaju cez seriovu komunikáciu hodnoty výstupov. Výpis je však podmienený práve pomocnou hodnotou *9* ktorá sa posiela z raspeberry každých 0.5 sekúnd. To znamená že Arduino vypíše a pošle výstupy iba 2 krát za sekundu a tak zabezpečíme nepreplnovanie buffera a aj Arduino može pracovaž z nízkym oneskorením (*delayom*). Ak by sme zvýšili *delay* RC obvody by začili 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**

Ako aj v predošlo prípade, je pre aplikáciu použité virtualne Raspberry a to vo softwary *Oracle VM VirtualBox*. Tak ako už bolo spomenute, požiadavkov je práca s dátami medzi zariadeniami čo zanemná odosielanie a primanie dat z Arduina. Dáta následne uklada do databazy a do textoveho suboru. Virtualne Raspberry dalej služi na prevádzku servera a web klienta.

* **Software**

V cykle while...

***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*