Теорија електричних кола

Хардверски пројекат

Дигитални једносмерни волтметар

Студенти:

Глорија Дошло 2019/0065

Кристина Рајковић 2019/0447

Ментор:

доц. др Никола Баста

Фебруар 2022. г.

Садржај

Увод

У оквиру предмета ***Теорија електричних кола*** који се изводи на смеру ***Рачунарска техника и информатика*** у понуди је велики број како софтверских, тако и хардверских пројеката.

У овом раду представљена је једна хардверска реализација дигиталног једносмерног волтметра уз помоћ Ардуино микроконтролера који би омогућио очитавање биполарног напона.

Најпре је пројектовано принципијелно коло које би се понашало као овај елемент и урађена анализа користећи алате за симулацију електричних кола.

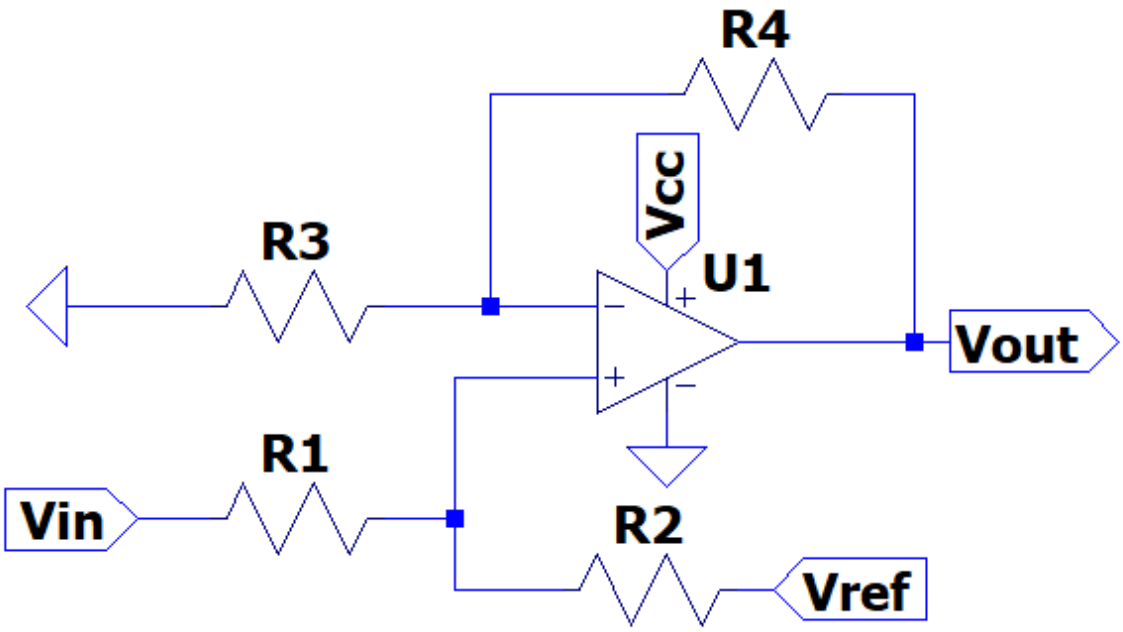
Након потврде исправности пројектовано је реално коло користећи доступне компоненте и испрограмиран микроконтролер да извршава задату функцију.

На крају је извршено повезивање са LCD модулом ради лакшег очитавања резултата. Као додатна погодност направљен је и GUI који служи за графичку презентацију резултата.

Функционалност волтметра је тестирана на простом колу које се напаја батеријом од $9V$. Тест напони су у опсегу $\pm 9V$ и мапирани су на напоне из опсега од $0V$ до $5V$.

1. Анализа кола

За почетак погледајмо иницијалну шему неинвертујућег сабирача који је коришћен као главни елемент кола.



Слика 1: неинвертујући сабирач

Основна идеја је да користећи овај сабирач задати опсег улазног напона мапирамо у одговарајући опсег излазног. Закључак је да између ових величина постоји линеарна зависност:

$$V_{out} = V_{in}g + V_{off} \tag{1}$$

где је g појачање операционог појачавача, а V_{off} офсетни напон.

То нас води до следећих једнакости:

$$V_{outmin} = V_{inmin}g + V_{off} \tag{2}$$

$$V_{outmax} = V_{inmax}g + V_{off} \tag{3}$$

Минимална вредност у нашем случају је $0V$, а максимална ${V_{cc}}^{\Gamma}$(приближна максимална вредност напајања). Уколико бисмо одузели једначину (2) од једначине (3) добили бисмо следећи израз:

$$g = \frac{{V_{cc}}^{\Gamma}}{\Delta V_{in}}.$$

(4)

Ако заменимо (4) у једначину (3) добијамо:

$$V_{off} = \frac{-V_{inmin}{V_{cc}}^{\Gamma}}{\Delta V_{in}}.$$

(5)

Трансфер функција овог сабирача која се лако може извести користећи принцип суперпозиције гласи:

$$V_{out} = \left(V_{in}\frac{R_2}{R_1 + R_2} + V_{ref}\frac{R_1}{R_1 + R_2}\right)\left(1 + \frac{R_4}{R_3}\right).$$

(6)

Следећи корак је да изједначимо коефицијенте у једначинама (1) и (6). Одатле долазимо до следећих израза:

$$\frac{R_2}{R_1 + R_2}\left(1 + \frac{R_4}{R_3}\right) = \frac{{V_{cc}}^{\Gamma}}{\Delta V_{in}}$$

(7)

$$\frac{R_1}{R_1 + R_2}\left(1 + \frac{R_4}{R_3}\right) = \frac{-V_{inmin}{V_{cc}}^{\Gamma}}{\Delta V_{in}V_{ref}}.$$

(8)

Увешћемо и следеће супституције за односе отпорника:

$$y = \frac{R_4}{R_3}$$

(9)

$$x = \frac{R_1}{R_2}.$$

(10)

Као познате константе ћемо сматрати следеће величине: минимални и максимални улазни напон, односно њихову апсолутну разлику, напајање операционог појачавача, као и један однос отпорника, конкретно је изабрано да то буде x.

Када саберемо једначине (7) и (8) и извршимо замену, добијамо:

$$1 + y = \frac{{V_{cc}}^{\Gamma}}{\Delta V_{in}} - \frac{V_{inmin}{V_{cc}}^{\Gamma}}{\Delta V_{in} V_{ref}}.$$

(11)

Такође, сада се једначина (7) може записати на другачији начин:

$$\frac{1 + y}{1 + x} = \frac{{V_{cc}}^{\Gamma}}{\Delta V_{in}} - \frac{V_{inmin}{V_{cc}}^{\Gamma}}{\Delta V_{in} V_{ref}}.$$

(12)

Из једначина (11) и (12) следи:

$$x = \frac{\left(\frac{{V_{cc}}^{\Gamma}}{\Delta V_{in}} - \frac{V_{inmin}{V_{cc}}^{\Gamma}}{\Delta V_{in} V_{ref}}\right)\Delta V_{in} - {V_{cc}}^{\Gamma}}{{V_{cc}}^{\Gamma}}.$$

(13)

Сређивањем израза (13) долазимо до финалног израза за израчунавање референтног напона:

$$V_{ref} = -\frac{V_{inmin}}{x}.$$

(14)

Сада из једначине (11) заменом једначине (14) добијамо и израз за рачунање y:

$$y = \frac{{V_{cc}}^{\Gamma}(1 + x)}{\Delta V_{in}} - 1.$$

(15)

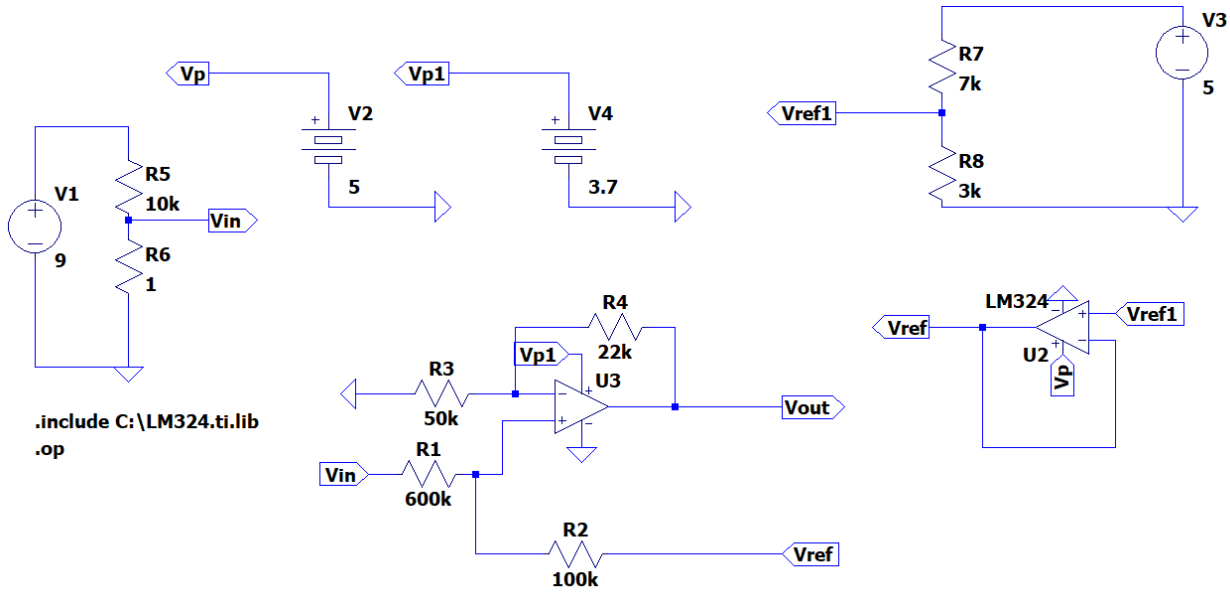
Иако x сматрамо за познату вредност, морамо водити рачуна и о томе да она мора бити позитивна, тако да на основу израза (15) долазимо до следеће неједнакости:

$$x > \frac{\Delta V_{in}}{{V_{cc}}^{\Gamma}} - 1.$$

(16)

Сада смо видели како можемо на основу улазних параметара да израчунамо и остале непознате вредности, а то су референтни напон и однос друге две отпорности.

Погледајмо комплетну шему електричног кола. Симулација је вршена у алату ***LTSpice***.



Слика 2: шема електричног кола (*LTSpice*)

У конкретној реализацији узето је да је улазни напон у опсегу $\pm 9V$, а излазни у опсегу од $0V$ до $3{,}7V$ јер је симулација прављена тако да одговара реалном колу, а одговарајући операциони појачавач који је коришћен има ограничење да му је горња граница управо $3{,}7V$ уместо очекиваних $5V$. Због разлике у реалној компоненти и компоненти доступној у библиотеци алата је операциони појачавач U_3 представљен као универзални којем се доводи напајање Vp_1 које је представљено батеријом V_4.

Такође, узето је да је вредност $x = \frac{R_1}{R_2} = 6$. На основу раније приказаних једначина добија се да је $y = \frac{R_4}{R_3} = 0{,}44$, а $V_{ref} = 1{,}5V$. Вредности отпорника R_1, R_2, R_3 и R_4 на шеми изабране су у складу са овим захтевима.

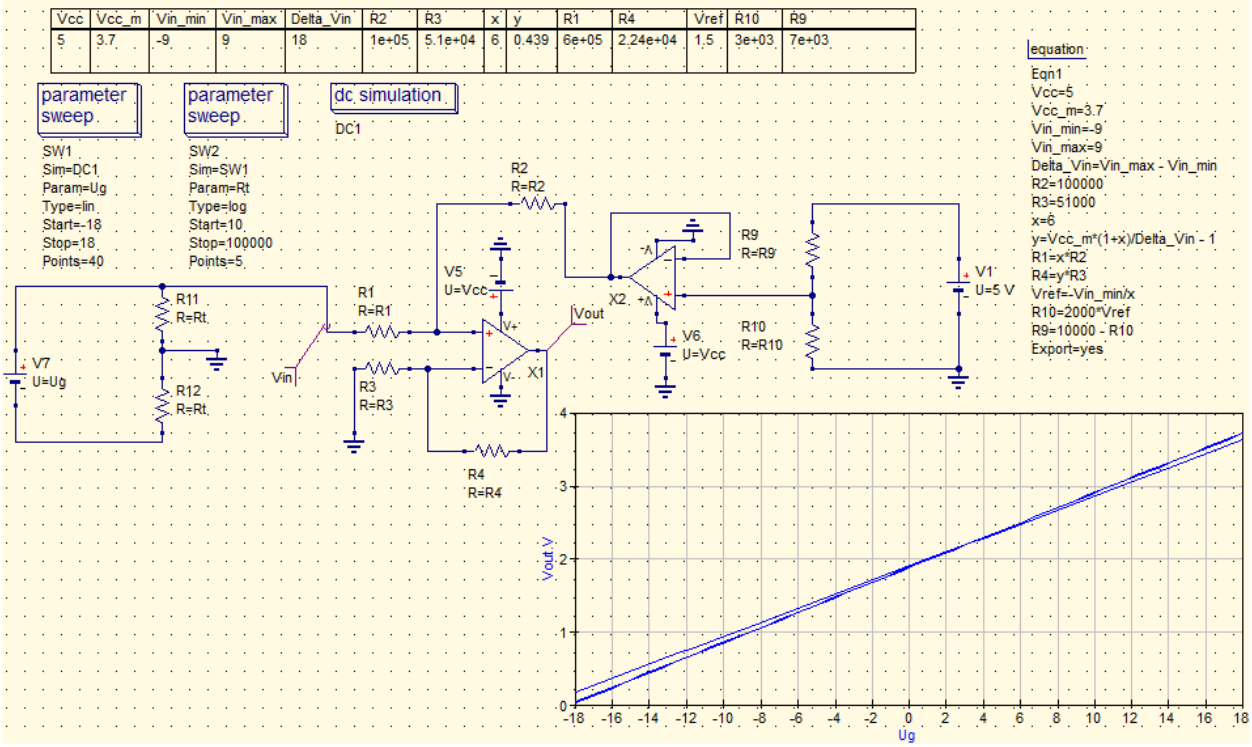
Као тест коло коришћено је просто коло које се налази на левој страни шеме. Оно се напаја батеријом од $9V$, а напон који се мери је напон на отпорнику R_6.

Просто коло на десној страни шеме које се напаја батеријом од $5V$ служи као разделник напона да би се добио неопходан референтни напон. Ова вредност батерије није

случајно изабрана, већ је бирана тако да се искористи напон који долази од Ардуино микроконтролера који има пин који има управо вредност од $5V$.

Операциони појачавач U_2 служи као јединични и он је уведен у шему да би пропуштао неизмењен референтни напон. Он се напаја такође на пин микроконтролера. Искоришћена је компонента која није у стандардној библиотеци алата већ се мора накнадно преузети и командом са слике убацити у симулацију.

Поред симулирања електричног кола у алату ***LTSpice,*** симулација је направљена и у алату ***QucsStudio***.



Слика 3: шема електричног кола (*QucsStudio)*

Шема је готово иста као са слике 2, осим што је овде тест коло направљено тако да параметар U_g узима вредности из опсега $\pm 18V$ тако да када се он преполови добијамо жељени улазни напон.

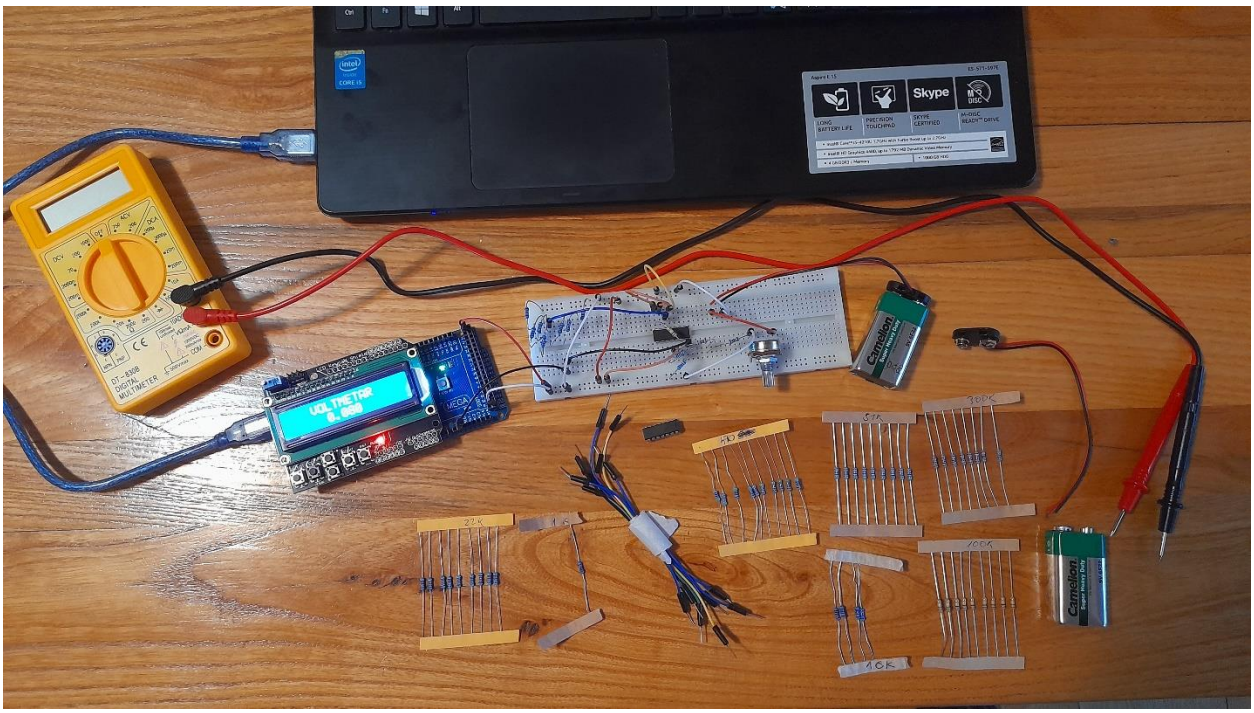
Искоришћена је погодност овог алата да се вредности које су фиксне задају, а све зависне величине израчунавају уз помоћ задатих једначина у оквиру блока „equation“. Табеларно су приказане вредности свих параметара који фигуришу.

Такође, добили смо и график зависности излазног напона од параметра U_g где је вршена промена вредности параметра R_t по логаритамској скали од 10Ω до $100k\Omega$.

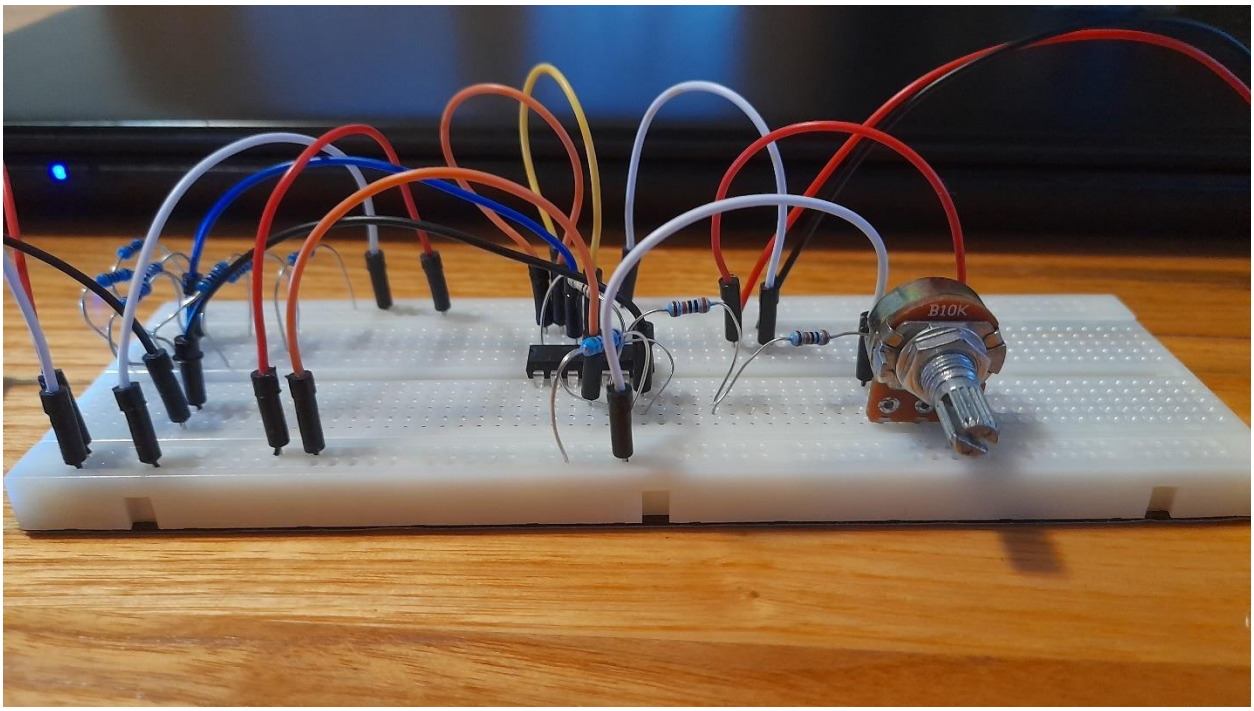
2. *Реално електрично коло*

Реално електрично коло састављено је на основу раније приказаних шема. Компоненте које су коришћене биће приказане табеларно.

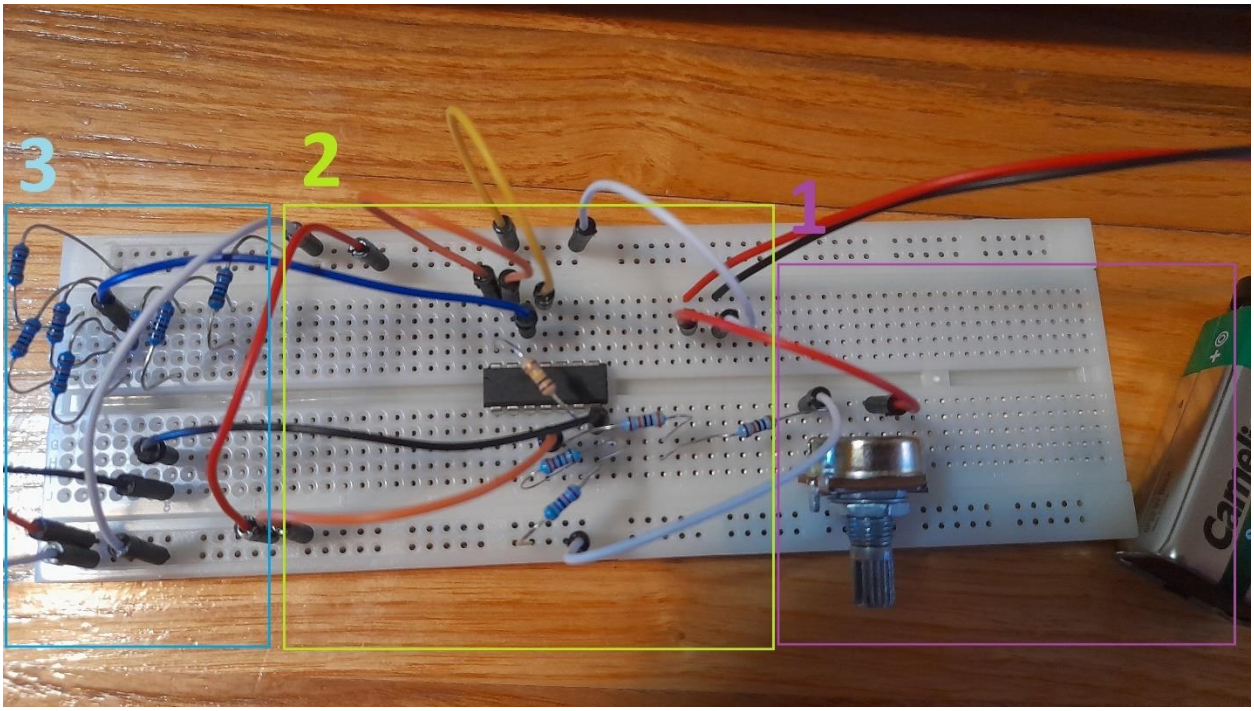
Назив компоненте	Количина
Каблови за повезивање контаката	**13**
Arduino ATMega 2560	**1**
LCD модул	**1**
Батерија од $9V$	**1**
Конектор за батерију од $9V$	**1**
Операциони појачавач LM324N(quad bipolar)	**1**
Потенциометар ($10k\Omega$)	**1**
Отпорник $10k\Omega$	**2**
Отпорник $1k\Omega$	**5**
Отпорник $22k\Omega$	**1**
Отпроник $51k\Omega$	**1**
Отпроник $100k\Omega$	**1**
Отпроник $300k\Omega$	**2**
Protoboard	**1**



Слика 4: реално електрично коло



Слика 5: реално електрично коло



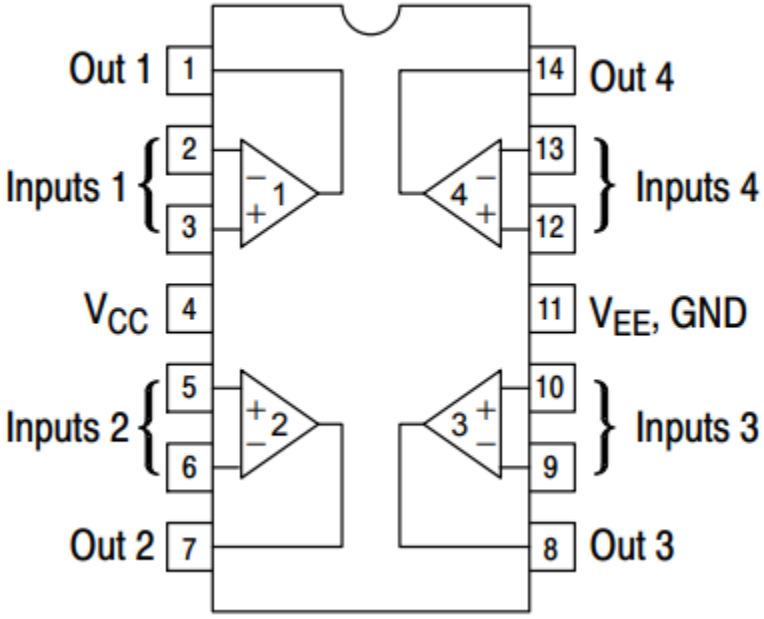
Слика 6: реално електрично коло

За реализацију кола коришћене су доступне компоненте тако да оно приближно одговара идеалном.

Први део кола који је на слици 6 нумерисан као „1“ је тест коло које се састоји од батерије од $9V$ и потенциометра који служи за промену напона који се мери.

Просто коло које служи као разделник напона (слика 6 – „3“) се напаја са Ардуино микроконтролера, на пин од $5V$. Пошто вредности отпорника који се користе у симулацији нису доступне, потребне вредности су представљене на следећи начин: отпорност од $7k\Omega$ је направљена тако што су везана у паралелу два отпорника од по $10k\Omega$, а затим везана на ред са два отпорника од $1k\Omega$; отпорност од $3k\Omega$ је добијена помоћу три отпорника везана на ред од $1k\Omega$.

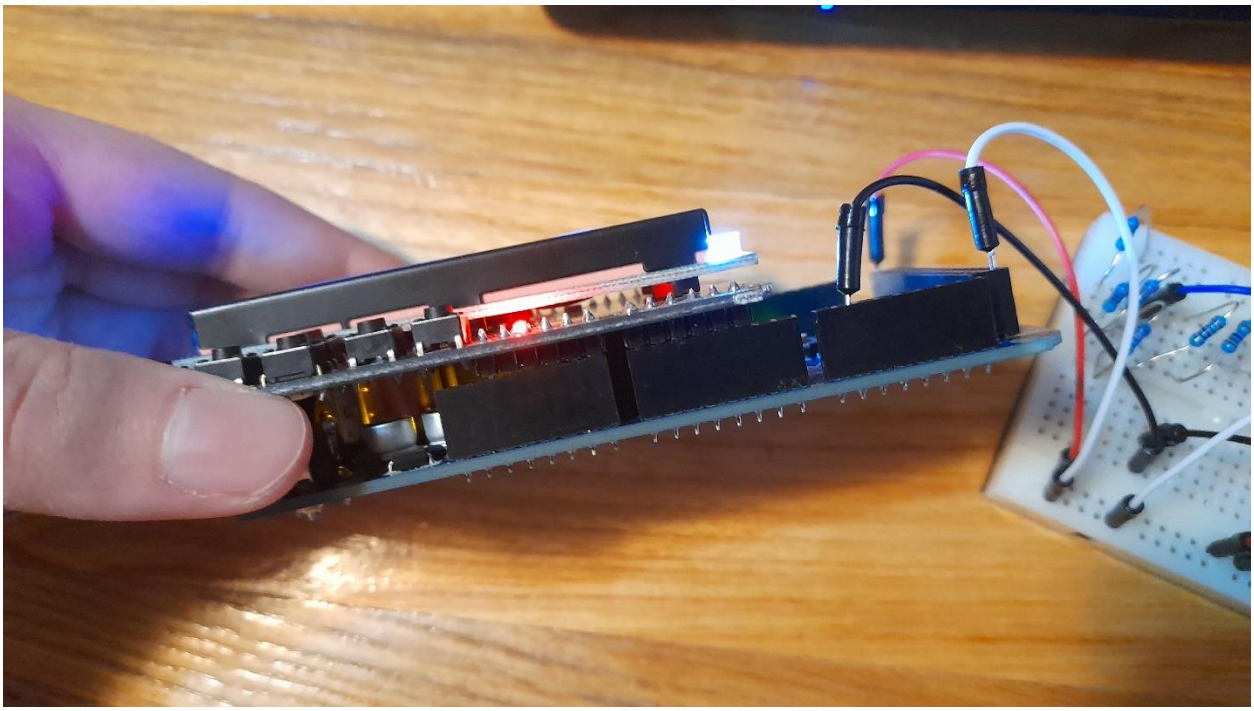
Операциони појачавач који је коришћен има следећу шему која се може пронаћи у његовом „datasheet”-у на интернету:



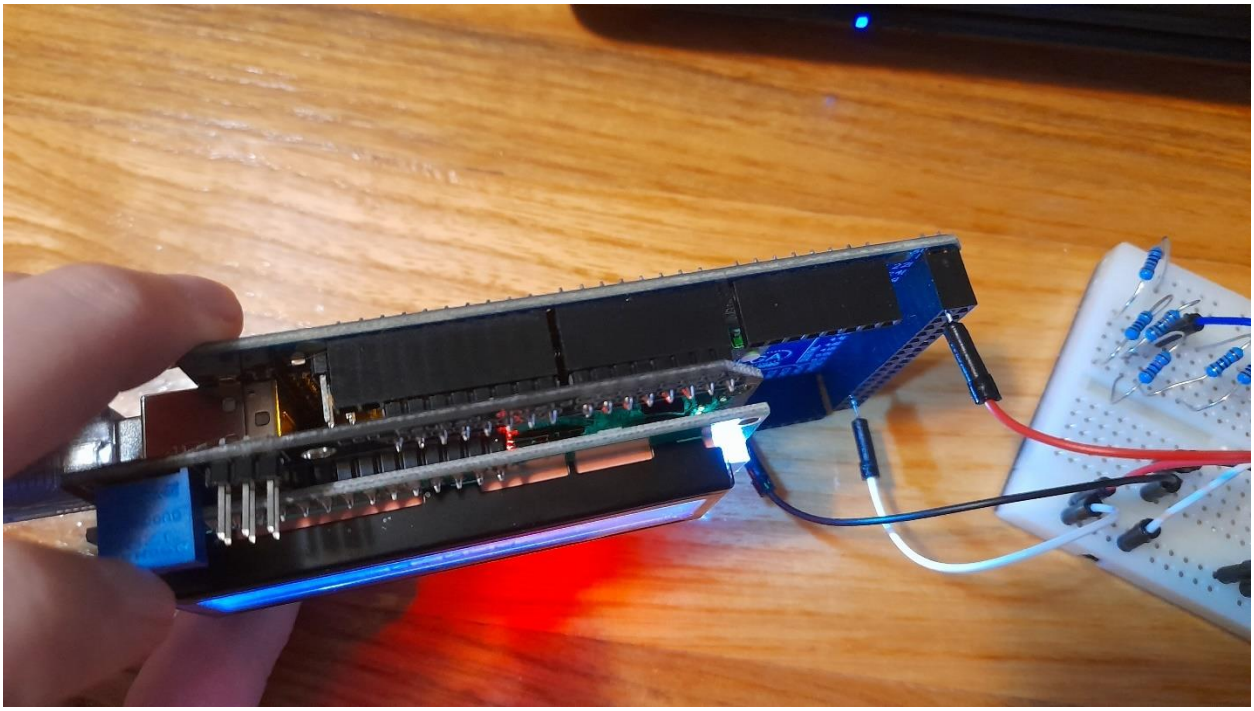
Слика 7: шема операционог појачавача LM324N

С обзиром да се у њему заправо садрже 4 операциона појачавача, број 4 је искоришћен као јединични који пропушта референтни напон, а излаз операционог појачавача број 2 се доводи на један од аналогних пинова Ардуино микроконтролера и он се користи као део кола које представља сабирач (слика 6 – „2“). Отпорници који се користе су бирани тако да најприближније одговарају шемама из симулација. Цео операциони појачавач се напаја на пин микроконтролера.

Такође, повезан је и LCD модул који је стављен директно на микроконтролер.



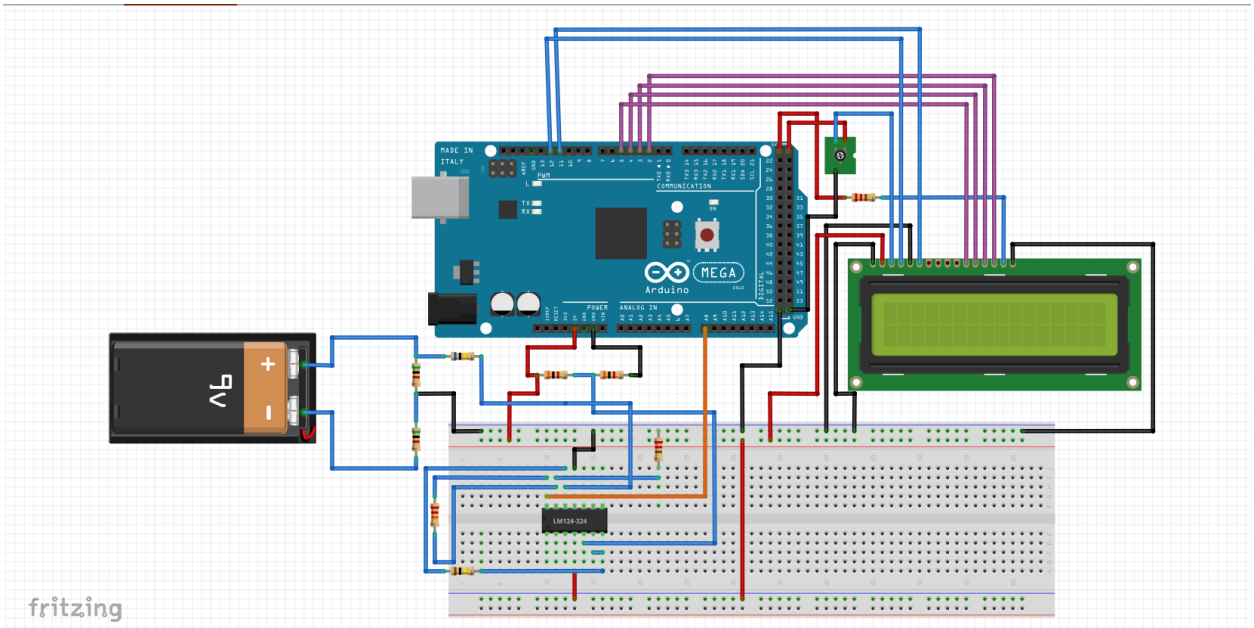
Слика 8: повезивање LCD модула



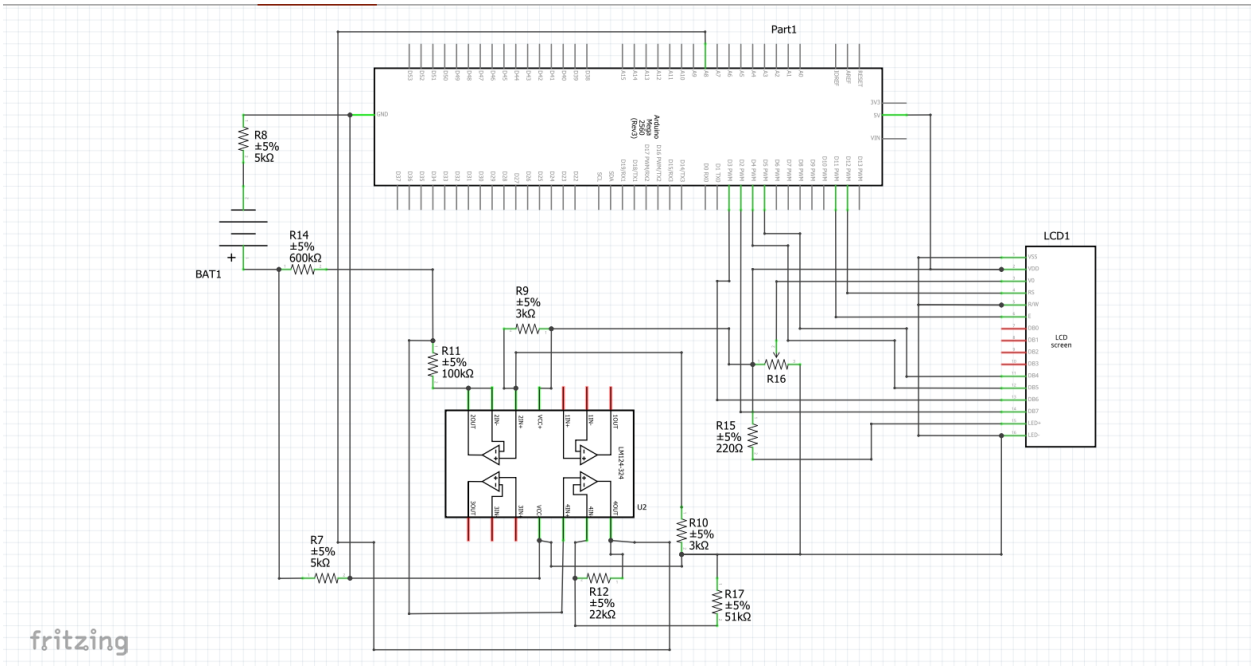
Слика 9: повезивање LCD модула

Слика 10: повезивање LCD модула

Комплетна шема урађена је и у алату ***Fritzing*** који омогућава да се повежу сви елементи као и у реалној шеми укључујући и периферије, а он црта шему тог електричног кола.



Слика 11: шема електричног кола (*Fritzing)*



Слика 12: шема електричног кола (*Fritzing)*

3. Програмирање Ардуино микроконтролера

Програм који извршава Ардуино микроконтролер је следећи:

```
/*
 * Biblioteka za lcd displej
 * Displej koji smo koristili je 16 x 2 sto znaci da
 * ima dve vrste i 16 kolona
*/
#include <LiquidCrystal.h>
/*
 * rs - register select pin koji kontrolise gde u LCD memoriji pisemo
 * en - enable pin koji omogucuje upis u registre
 * d4, d5, d6, d7 - data pins su bitovi koje upisujemo u registre ili
 * citamo iz njih
*/
// inicijalizuje lcd displej, ako lcd ne ispisuje nista proverite
// da li su brojevi pinova dobri, ne moraju biti isti brojevi kao ovde
const int rs = 8, en = 9, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
int numOfColumns = 16, numOfRows = 2;
float vin; //mereni napon
int deltaVin = 18; //razlika maksimalnog i minimalnog ulaznog napona
int maxAnalogValue = 1023; //maksimalni broj koraka
int reducedMaxAnalogValue; //redukovani broj koraka
int vinmin = -9; //minimalna vrednost ulaznog napona
int vout = 0; //izlazni napon
float arduinoMaxVoltage = 5; //maksimalna vrednost napona
float reducedMaxVoltage = 3.641; //redukovana vrednost napona


// funkcija koja se izvrsi samo jednom na pocetku i sluzi za pocetna
podesavanja
void setup()
{
  lcd.begin(numOfColumns, numOfRows); // podesava broj kolona i vrsta
  // 9600 je maksimalan broj bitova koji se moze preneti
  Serial.begin(9600); // sluzi za pokretanje serial monitora
  lcd.setCursor(3,0); // pomera kursor na 0 vrstu i trecu kolonu
  lcd.print("VOLTMETAR"); // ispisuje na lcd
  // 5 : 1023 = 3,641 : x => x = 1023 * 3,641 / 5 => x = 745
  reducedMaxAnalogValue = round((maxAnalogValue * reducedMaxVoltage) /
arduinoMaxVoltage);
}

// funkcija koju arduino izvrsava u petlji sve dok se ne iskljuci ploca
void loop()
{
  lcd.setCursor(5, 1);
  // cita sa A9 analognog pina i smesta u celobrojnu promenljivu vout
  vout = analogRead(A9); //izlazni napon
  Serial.print(vout);  // ispisuje na serijski monitor
  Serial.print(",");
```

```
  vin = (vout * deltaVin) / (float)reducedMaxAnalogValue + vinmin; //mereni
napon
  //Serial.print("\nU = "); // '\n' oznacava prelazak u nov red
  Serial.println(vin);
  // ispisuje vrednost promenljive vin
  lcd.print(vin);
  delay(200);  // sacekaj 200 milisekundi
}
```

На почетку програм иницијализује и подешава LCD модул и покреће „serial monitor" у којем микроконтролер врши свој испис. Такође, врши се израчунавање редукованог броја корака који се могу очитати. С обзиром да овај микроконтролер може да очитава само напоне из опсега од $0V$ до $5V$ и постоји конверзија из аналогних у дигиталне вредности који ради са десетобитним величинама, нама се овај интервал пресликава у вредности од 0 до 1023. Како је граница операционог појачавача $3{,}7V$, највећа вредност која се може очитати је 745 и она је коришћена у даљем израчунавању.

У петљи се на сваких $200ms$ учитава нова вредност са аналогног пина (може се користити било који осим резервисаног пина A_0). Израчунава се мерени напон на основу једначине која у нашем конкретном случају гласи:

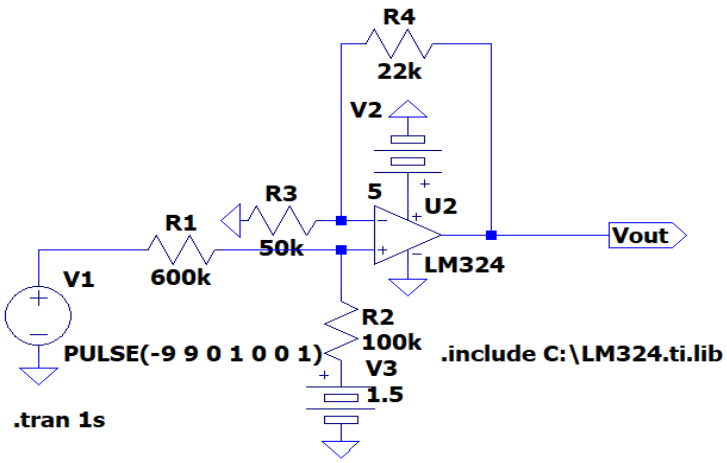
$$V_{in} = V_{out}\frac{18}{3.7} - 9.$$

Добијени улазни напон исписујемо на LCD модул и „serial monitor".

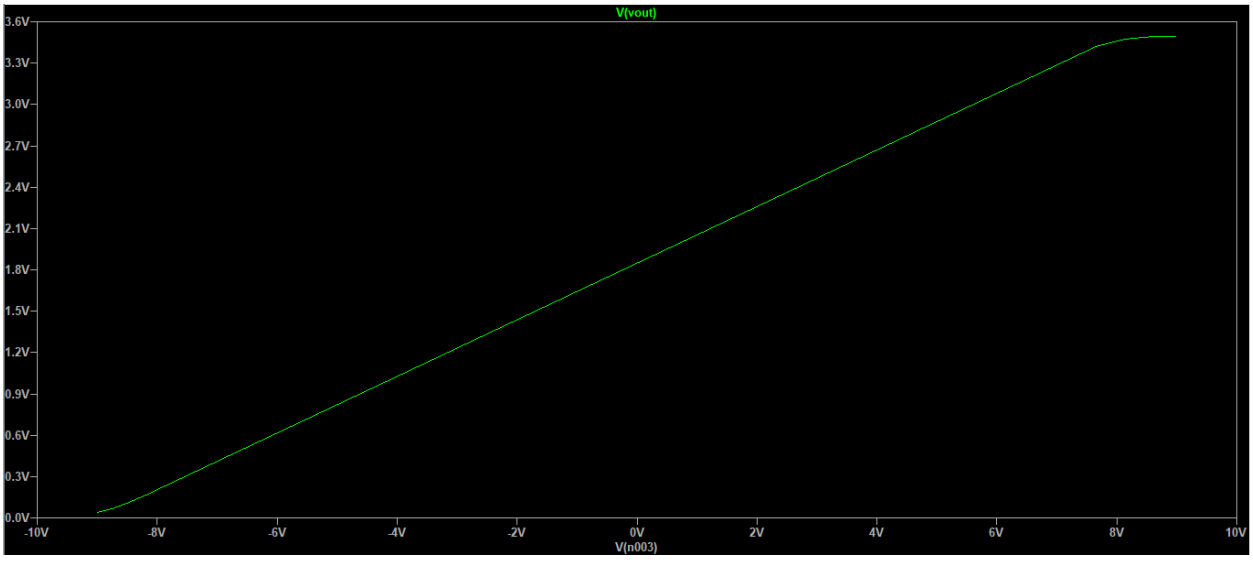
4. *Анализа грешке*

Један од првих проблема који је настао услед непостојања одговарајућег реално доступног операционог појачавача је био смањење скале. У односу на првобитну скалу, интервал је смањен за $1{,}3V$.

Ово ћемо илустровати на примеру следећег кола које се разликује од пређашњег по томе што су улазни и референтни напон представљени батеријама. Такође, коришћен је операциони појачавач LM324 чије је ограничење мало веће него оно које је експериментално утврђено да би симулација била исправна јер универзални који се нуди у библиотеци алата нема ограничења.



Слика 13: шема електричног кола за испитивање рада операционог појачавача



Слика 14: график зависности излазног напона од улазног

Можемо приметити да долази до заравњења скале и због овога је одабрано да се вредности мапирају на опсег од $0V$ до $3{,}7V$.

С обзиром да је смањен број подеока које Ардуино може очитати, резолуција, односно најмања теоретска вредност која се може детектовати је $\frac{18V}{745} = 24{,}16mV$ (величина интервала подељена са бројем корака).

Грешку мерења ћемо испитати у екстремним тачкама:

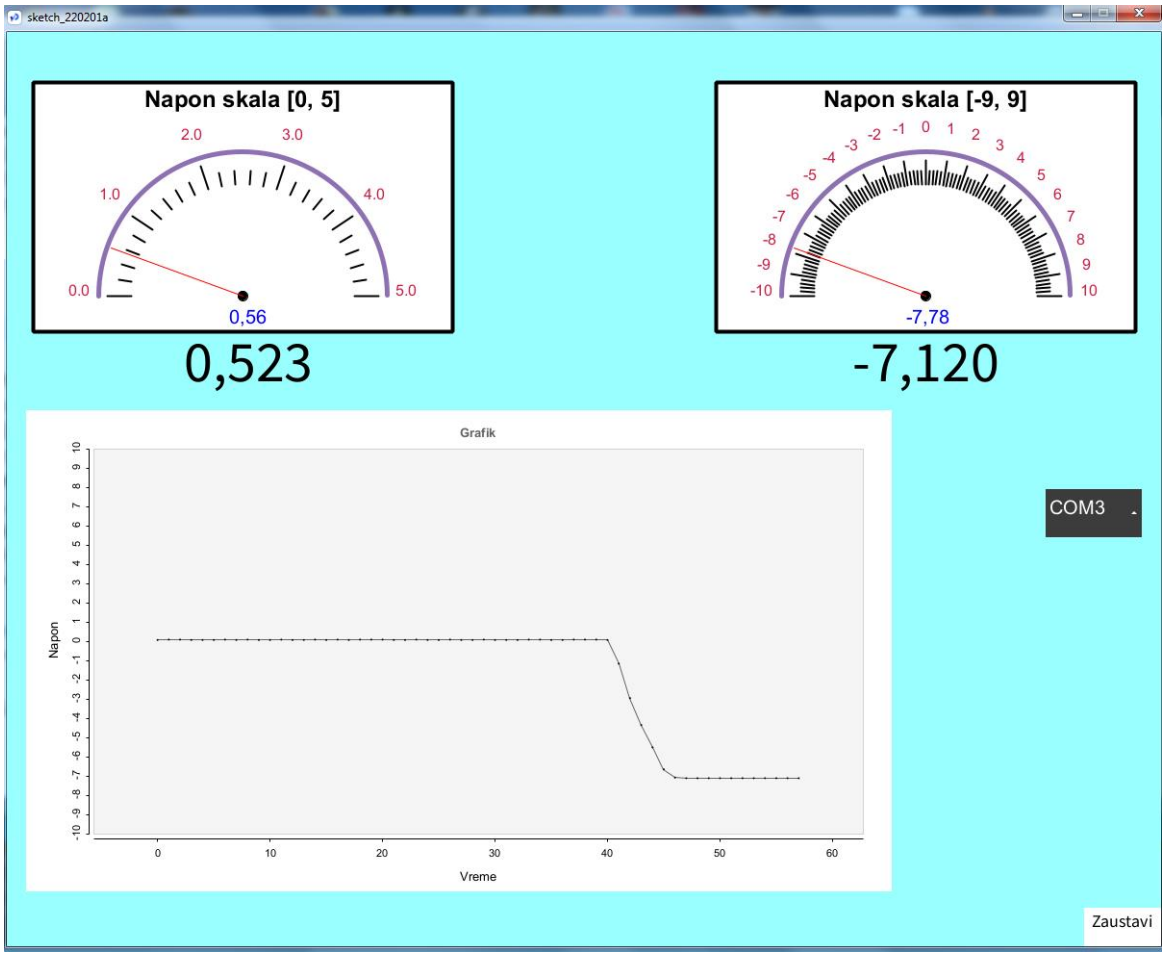
Тачна вредност	Измерена вредност	Апсолутна грешка
$\boldsymbol{-9V}$	$\boldsymbol{-9{,}00V}$	$\boldsymbol{0}$
$\boldsymbol{0V}$	$\boldsymbol{0{,}011V}$	$\boldsymbol{0{,}011}$
$\boldsymbol{9V}$	$\boldsymbol{8{,}816V}$	$\boldsymbol{0{,}184}$

Мерења су вршена и уз помоћ мултиметра. Упоредићемо вредности које очитава Ардуино микроконтролер и мултиметар:

Тачна вредност V_{in}	Измерена вредност V_{in} мултиметром	Измерена вредност V_{in} Ардуином	Апсолутна грешка
$\boldsymbol{-9V}$	$\boldsymbol{-9{,}02V}$	$\boldsymbol{-9{,}00V}$	$\boldsymbol{0{,}02}$
$\boldsymbol{0V}$	$\boldsymbol{0{,}16V}$	$\boldsymbol{0{,}011V}$	$\boldsymbol{0{,}149}$
$\boldsymbol{9V}$	$\boldsymbol{9{,}03V}$	$\boldsymbol{8{,}816V}$	$\boldsymbol{0{,}214}$

5. *Графички кориснички интерфејс*

Ради лакшег очитавања направљен је и графички кориснички интерфејс у алату ***Processing*** који нам омогућује да читамо податке са Ардуино микроконтролера и користећи синтаксу програмског језика ***Java*** напишемо код за нашу апликацију.



Слика 15: GUI апликација

На слици видимо два метра који служе за очитавање напона који су представљени у одговарајућим скалама. Такође, приказујемо вредности улазног напона и графички кроз време. Постоји и падајућа листа која служи за бирање порта са кога примамо мерени сигнал, као и дугме за заустављање целе апликације. Уколико се на ово дугме кликне поново, програм се изнова покреће.

Код овог програма је следећи:

```
import controlP5.*;
import meter.*;
import processing.serial.*;
import grafica.*;
import java.util.Arrays;

color backgroundColor = color(153, 255, 255);
color textColor = color(0, 0, 0);

Meter m, m2;
Serial port;
GPlot plot;
ControlP5 cp5;
DropdownList d1;

String arduinoPortName;
String portName;


boolean newData = true;
int xPos = 0;
boolean rectOver = false;
boolean begin = true;
boolean arduinoConnected = false;
boolean portLost = false;

int rectX, rectY, rectWidth, rectHeight;
color rectHighlight = color(204);
color rectColor = color(255);
String[] lastComList;


void setup() {
  lastComList = Serial.list();
  portName = Serial.list()[0]; //0 as default
  port = new Serial(this, portName, 9600);
  //port.bufferUntil('\n');

  size(1200, 950);
  background(backgroundColor);
  cp5 = new ControlP5(this);

  PFont pfont = createFont("Arial", 10, true); //Create a font
  ControlFont font = new ControlFont(pfont, 20); //font, font-size

  d1 = cp5.addDropdownList("myList-d1")
    .setPosition(width - 120, height/2)
    .setSize(100, 200)
    .setHeight(210)
    .setItemHeight(40)
    .setBarHeight(50)
    .setFont(font)
    .setColorBackground(color(60))
    .setColorActive(color(255, 128))
    ;
```

```
  d1.getCaptionLabel().set("PORT"); //set PORT before anything is selected



  m = new Meter(this, 25, 50);

  int mx = m.getMeterX();
  int my = m.getMeterY();
  int mw = m.getMeterWidth();
  int mh = m.getMeterHeight();

  m.setTitleFontName("Arial bold");
  m.setTitle("Napon skala [0, 5]");

  m.setScaleFontColor(color(200, 30, 70));
  m.setDisplayDigitalMeterValue(true);

  m.setArcColor(color(141, 113, 178));
  m.setArcThickness(5);
  m.setMinScaleValue(0);
  m.setMaxScaleValue(5);
  m.setMinInputSignal(-9);
  m.setMaxInputSignal(9);


  m2 = new Meter(this, width - mx - m.getMeterWidth(), my);
  m2.setTitleFontName("Arial bold");
  m2.setTitle("Napon skala [-9, 9]");
  String[] scaleLabels = { "-10", "-9", "-8", "-7", "-6", "-5", "-4", "-3",
"-2", "-1",
    "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10"};

  m2.setScaleLabels(scaleLabels);
  m2.setScaleFontColor(color(200, 30, 70));
  m2.setDisplayDigitalMeterValue(true);

  m2.setArcColor(color(141, 113, 178));
  m2.setArcThickness(5);

  m2.setMinScaleValue(-10);
  m2.setMaxScaleValue(10);
  m2.setMinInputSignal(-9);
  m2.setMaxInputSignal(9);

  int m2x = m2.getMeterX();
  int m2y = m2.getMeterY();
  int m2w = m2.getMeterWidth();
  int m2h = m2.getMeterHeight();


  plot = new GPlot(this);
  plot.setPos(20, my + mh + 80);
  plot.setDim(800, 400);
  plot.setPointColor(color(0, 0, 0, 255));
  plot.setPointSize(2);
  plot.setTitleText("Grafik");
```

```
  plot.getXAxis().getAxisLabel().setText("Vreme");
  plot.getYAxis().getAxisLabel().setText("Napon");
  plot.drawGridLines(GPlot.BOTH);
  plot.setGridLineColor(color(0, 0, 0));
  plot.activateZooming(2, CENTER, CENTER);
  plot.activateReset();
  plot.activatePanning();
  plot.setYLim(-10, 10);
  plot.setVerticalAxesNTicks(18);

  rectWidth = 80;
  rectHeight = 40;
  rectX = width - rectWidth;
  rectY = height - rectHeight;

  background(backgroundColor);
  textSize(60);
  textAlign(CENTER);
  fill(textColor);
  text(0, mx + mw/2, my + mh + 50);
  text(0, m2x + m2w/2, m2y + m2h + 50);

  thread("DropListThread");
  thread("portsListThread");
}

void draw() {
  if (begin) {
    m.updateMeter(0);
    m2.updateMeter(0);
    plot.defaultDraw();
    update();
  }

  if (!arduinoConnected || portLost) {
    fill(backgroundColor);
    stroke(backgroundColor);
    rect(width - 2*rectWidth - 115, height/2 - 80, 4*rectWidth, rectHeight +
30);
    fill(textColor);
    textSize(20);
    textAlign(CENTER);
    if (portLost)
      text("Arduino izgubljen!\nIzaberite port.", width - 2*rectWidth + 20,
height/2 - 60);

    else {
      text("Nema konekcije sa Arduinom!\nIzaberite port.", width -
2*rectWidth + 20, height/2 - 60);
    }
  }

  int mx = m.getMeterX();
  int my = m.getMeterY();
  int mw = m.getMeterWidth();
  int mh = m.getMeterHeight();
```

```
  int m2x = m2.getMeterX();
  int m2y = m2.getMeterY();
  int m2w = m2.getMeterWidth();
  int m2h = m2.getMeterHeight();


  String str = "";
  String[] tokens;
  if (port.available() > 0) {
    str = port.readStringUntil('\n');

    if (str != null) {

      tokens = split(str, ',');
      if (tokens.length == 2) {
        float voltage = float(tokens[1]);

        if (newData) {
          background(backgroundColor);
          textSize(60);
          textAlign(CENTER);

          float val = (float(tokens[0]) / 745.0) * 5;
          float val2 = (float(tokens[0]) / 745.0 * 18) - 9;
          m2.updateMeter((int)val2);
          m.updateMeter((int)voltage);
          fill(textColor);
          text(val, mx + mw/2, my + mh + 50);
          text(voltage, m2x + m2w/2, m2y + m2h + 50);
          if (xPos > 300) {
            while (xPos > 0) {
              plot.removePoint(--xPos);
            }
          }

          plot.addPoint(xPos++, voltage);
          plot.defaultDraw();
        }
      }
      fill(backgroundColor);
      stroke(backgroundColor);
      rect(0, height - rectHeight, 400, rectHeight);

      fill(textColor);
      textSize(20);
      textAlign(LEFT);
      if (plot.isOverBox(mouseX, mouseY)) {
        float[] value = plot.getValueAt(mouseX, mouseY);
        text("x = " + value[0] + ", y = " + value[1], 0, height - 20);
      }
      update();
    }
  }
}

boolean overRect(int x, int y, int width, int height) {
  if (mouseX >= x && mouseX <= x+width &&
```

```
    mouseY >= y && mouseY <= y+height) {
    return true;
  } else {
    return false;
  }
}

void update() {
  if (overRect(rectX, rectY, rectWidth, rectHeight)) {
    rectOver = true;
    fill(rectHighlight);
  } else {
    rectOver = false;
    fill(rectColor);
  }
  stroke(255);
  rect(rectX, rectY, rectWidth, rectHeight);
  textAlign(CENTER);
  fill(textColor);
  textSize(18);
  text("Zaustavi", rectX + rectWidth / 2, rectY + rectHeight / 2);
}

boolean pressedOnce = true;
void mousePressed() {

  if (rectOver && pressedOnce) {
    newData = false;
    pressedOnce = false;
    fill(backgroundColor);
    stroke(backgroundColor);
    rect(rectX - 3*rectWidth, rectY, 3*rectWidth, rectHeight);
    fill(textColor);
    textSize(20);
    text("Zaustavljeno!", width - rectWidth - rectWidth, height - 10);
  } else if (rectOver && !pressedOnce) {
    newData = true;
    pressedOnce = true;
  }
}

void controlEvent(ControlEvent theEvent) { //when something in the list is
selected
  port.clear(); //delete the port
  port.stop(); //stop the port
  begin = false;
  if (theEvent.isController() && d1.isMouseOver()) {
    portName = Serial.list()[int(theEvent.getController().getValue())];
//port name is set to the selected port in the dropDownMeny
    port = new Serial(this, portName, 9600); //Create a new connection
    delay(1000);  // ceka arduino
    if (port.readStringUntil('\n') == null) {
      arduinoConnected = false;
    } else {
      arduinoPortName = portName;
      arduinoConnected = true;
    }
```

```
    //delay(2000);
  }
}

void DropListThread() {
  println("DropListThread started");
  while (true) {
    if (d1.isMouseOver()) {
      d1.clear(); //Delete all the items
      for (int i=0; i<Serial.list().length; i++) {
        d1.addItem(Serial.list()[i], i); //add the items in the list
      }
    }
    delay(800);
  }
}

void portsListThread() {
  println("portsListThread started");
  while (true) {
    println("proveravam uslov");
    String[] tempList = Serial.list();
    if (!Arrays.equals(lastComList, tempList)
      && !Arrays.asList(tempList).contains(arduinoPortName)) {
      println("Arduino izgubljen");
      portLost = true;
    }
    if (Arrays.equals(lastComList, tempList) && portLost) {
      portLost = false;
    }
    delay(3000);
  }
}
```

На почетку увозимо све неопходне библиотеке и подешавамо боју позадине и текста и дефинишемо променљиве .

У функцији setup() прво стварамо листу доступних портова и бирамо први као подразумевани. Вршимо почетна подешавања величине прозора и фонта. Након тога стварамо све горе поменуте компоненте и њих позиционирамо у оквиру прозора и додељујемо им потребна својства. Поред њих креирамо и две нити које ће ослушкивати промене везане за падајућу листу портова.

Функција draw() прво пушта у рад наша два метра и црта прозор. Позива се и функција update() која служи за подешавање дугмета. Поставља се натпис дугмета на „Заустави“ и уколико је миш позициониран на њега мења се боја. За испитивање позиције миша користимо функцију overRect(). Потом испитујемо да ли постоји конекција са Ардуино микроконтролером или да ли је дошло до прекида у конекцији и у том случају правимо правоугаоник у ком ћемо исписивати до које врсте грешке је дошло.

Онда уколико је порт активан примамо улазне податке у паровима раздвојене запетама који треба да се састоје од броја корака које је Ардуино микроконтролер очитао и напона који је израчунао. Подешавамо метре да се поставе на очитане вредности и

исцртавамо график. Такође, исписујемо очитане вредности испод одговарајућих метара, а уколико се детектује и да је миш постављен на график у доњем левом углу се приказују вредности x и y координата.

За заустављање апликације користимо функцију mousePressed() која детектује да ли је извршен један клик мишем и у том случају исписује на екрану „Заустављено!“, а у случају да то није први клик поставља вредност промењиве newData на 1 која онда омогућава поновно постављање нових вредности на метре.

Такође имамо и функцију controlEvent() која прима један аргумент типа ControlEvent која детектује да је селектовано нешто из падајуће листе и зауставља до сада коришћени порт и пушта у рад селектовани и испитује да ли је заправо на њему повезан наш микроконтролер у зависности од тога да ли му стижу подаци или не. Претпостављено је да нећемо имати друге периферије са којима се повезујемо, тако да ако стижу нови подаци то је сигурно од Ардуино микроконтролера.

На крају имамо и код за две додатне нити. Прва је нит DropListThread која доступне портове уписује у падајућу листу. Друга portsListThread која ослушкује да ли је дошло до губитка конекције на сваке три секунде тако што пореди листу првобитно очитаних портова са оном тренутном и испитује да ли се у тренутној налази порт на који је повезан Ардуино микроконтролер.

Закључак

Симулације и реална репрезентација показују да се приказано електрично коло заиста понаша као дигитални једносмерни волтметар. Приказана су и одступања од теоретских вредности која су проузрокована разним карактеристикама које реалне компоненте поседују. Она нису превелика, тако да се оправдава коришћење одабраних компоненти.

Као што смо и раније видели, највећи проблем који је настао при изради овог пројекта је неизбежност смањења скале услед недоступности одговарајућег операционог појачавача. Овај недостатак је превазиђен прилагођењем програма Ардуино микроконтролера.

Такође, прорачун који је приказан у овом раду се може применити и на било који други опсег вредности улазних и излазних напона (док се води рачуна о ограничењима Ардуино микроконтролера) и уз минималне измене искористити за пројектовање сличног електричног кола које би вршило мапирање биполарног напона.

Литература

1. Теорија електричних кола – сајт предмета (TEORIJA ELEKTRIČNIH KOLA, Univerzitet u Beogradu -- Elektrotehnički fakultet (etf.rs))
2. Биполарни напон (Measure a Bipolar Signal with an Arduino Board – Mastering Electronics Design)
3. Сабирач (Summing Amplifier Calculator – Mastering Electronics Design)
4. LTSpice (LTspice Simulator | Analog Devices)
5. QucsStudio (Home - QucsStudio)
6. Fritzing (Fritzing)
7. Arduino (Arduino - Home)
8. Tinkercad (Tinkercad | Create 3D digital designs with online CAD | Tinkercad)
9. Processing (Welcome to Processing! / Processing.org)