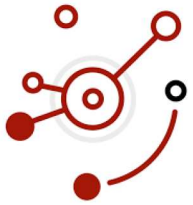
Wheatstoneov most za mjerenje otpora

Gača, Fran

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU ODJEL ZA FIZIKU



FRAN GAČA

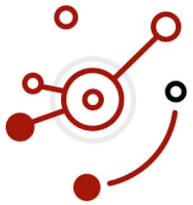
Izrada interaktivnih simulacija iz područja Osnova fizike

Wheatstoneov most za mjerenje otpora

Završni rad

Osijek, 2023.

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU ODJEL ZA FIZIKU



FRAN GAČA

Izrada interaktivnih simulacija iz područja Osnova fizike

Wheatstoneov most za mjerenje otpora

Završni rad

Predložen Odjelu za fiziku Sveučilišta Josipa Jurja Strossmayera u Osijeku radi stjecanja zvanja prvostupnika fizike

Osijek, 2023.

Ovaj završni rad izrađen je u Osijeku pod mentorstvom izv. prof. dr. sc. Branka Vukovića i komentora doc. dr. sc. Ivana Vazlera u sklopu Sveučilišnog preddiplomskog studija Fizike na Odjelu za fiziku Sveučilišta Josipa Jurja Strossmayera u Osijeku.

Zahvaljujem Izv. prof. dr. sc Branku Vukoviću, doc. dr. sc. Ivanu Vazleru i svojoj obitelji na podršci, pomoći i uputama pri izradi ovog završnog rada.

Sveučilište Josipa Jurja Strossmayera u Osijeku Završni rad

Odjel za fiziku

WHEATSTONEOV MOST ZA MJERENJE OTPORA

FRAN GAČA

Sažetak

U prvom dijelu završnog rada objašnjeni su pojmovi električnog naboja, električne struje, razlozi zašto se naboj kreće vodičem kao i izvod Ohmovog zakona te zašto je on važan u polju elektrodinamike.

U drugom dijelu objašnjavaju se Kirchhoffova pravila, a kao jedan od praktičnih načina mjerenja otpora objašnjava se Wheatstoneov most za mjerenje otpora čija je interaktivna simulacija mjerenja otpora sastavni dio ovog završnog rada.

Ključne riječi: Ohmov zakon/ Kirchhoffovi zakoni/ Wheatstoneov most/ Otpor/ Simulacija

(38 stranica, 15 slika, 18 literaturnih navoda)

Rad je pohranjen u Knjižnici Odjela za fiziku.

Mentor: Izv. prof. dr. sc. Branko Vuković

Komentor: Doc. dr. sc. Ivan Vazler

Ocjenjivač:

Rad prihvaćen:

Josip Juraj Strossmayer University of Osijek Bachelor of Physics Thesis

Department of Physics

WHEATSTONE BRIDGE FOR MEASURING RESISTANCE

FRAN GAČA

Abstract

The first part of the Thesis explains terms of electric charge, electric current, reasons why charge moves through the conductor as well as Ohm's law and why is it important in field of electrodynamics.

The second part of the Thesis explains Kirchhoff's circuit laws and as one of the few ways to measure resistance is presented Wheatstone bridge for measuring resistance which is also the core theme of this Thesis.

Key words: Ohm's law/ Kirchhoff's circuit laws/ Wheatstone bridge/ Resistance/ Simulation

(38 pages, 15 pictures, 18 citations)

Thesis is deposited in the Library of the Department of Physics.

Supervisor: Assoc. Prof. Ph.D. Branko Vuković

Comentor: Asst. Ph.D. Ivan Vazler

Reviewer:

Thesis accepted:

SADRŽAJ:

1. UVOD

Eksperiment u fizici je vrlo važan alat u proučavanju, spoznavanju i shvaćanju svakodnevnih pojava u prirodi. U 17. stoljeću su znanstvenici poput Galilea Galileia i Isaaca Newtona shvatili da bez eksperimenta nije moguće jasno i kvalitetno objasniti pojave s kojima su se susretali u to vrijeme, a objašnjavanje istih ljudima koji nisu bili u potpunosti upoznati i obrazovani u područjima matematike i fizike činilo se potpuno besmisleno.

U današnje vrijeme eksperimenti se provode u svim granama znanosti i doprinose, osim boljem shvaćanju naučenog, motivaciji učenika za daljnji rad i zanimanje za određeni tip problema ili skup problema s kojima se susreću. Današnji učenici u velikoj većini imaju strah od eksperimenta.

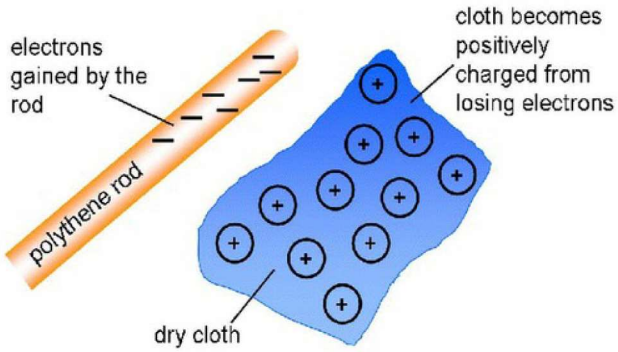
Moj završni rad potaknut je upravo time da učenicima pomognem prebroditi strah od eksperimenta. Koristeći interaktivni program iz područja fizike učenici će moći na bezbolan način ući u svijet eksperimenta i primijeniti svoje znanje u praksi. Program će im omogućiti ne samo bolje shvaćanje naučenog gradiva već možda i potaknuti njih same da se upuste u izradu sličnog programa kako bi i drugima pomogli prebroditi strah.

2. TEORIJSKI DIO

Još u vrijeme Talesa iz Mileta (600. god. Pr. Kr.) ljudi su znali kako postoje tvari koje se mogu naelektrizirati, no u to vrijeme takve pojave još nisu znali jasno i precizno objasniti. Zašto trljanjem jantara dolazi do privlačenja sitnih čestica prašine? Jantar je nazvan u grčkom jezikoslovlju *elektron*, a od tuda je potekao današnji naziv elektricitet. [1] Kada gledamo tvari na atomskom nivou one teže neutralnosti tj. ravnoteži sustava. Djelovanjem različitih čimbenika na tvari one mogu, između ostalog, postati i električki nabijene odnosno sadržavati naboj. Električni naboj je svojstvo čestica koje međudjeluju električnim silama. Ono što su ljudi uočili kao prvi električni efekt bilo je upravo trljanje jantara vunenom krpom što je uzrokovalo prijenos slobodnih elektrona s krpe na jantar te bi tako vunena krpa dotad neutralni jantar učinila naelektriziranim i sposobnim privući lagane tvari poput čestica prašine ili pramena kose. Kasnije su ljudi proučavanjem temelja elektrostatike i elektrodinamike kao važnih grana znanosti fizike spoznali kako su elektroni nosioci naboja. Danas znamo da se usmjereno gibanje elektrona naziva električna struja ili struja elektrona. Električna struja tj. gibanje elektrona događa se ako u

vodiču kojim teče struja postoji razlika potencijala. Razlika potencijala je naziv za razliku energija među nosiocima naboja na nekom mjestu u vodiču. Električni potencijal po definiciji je rad potreban da bi se jedinični naboj iz neke referentne točke (najčešće se uzima neka točka u beskonačnosti) premjestio u određenu točku prostora. Tokom proučavanja temeljnih fizikalnih zakonitosti u elektrodinamici došlo se do zaključka kako su električni potencijal i električno polje (polje sila kojima naboj djeluje na ostale naboje) povezani i to linearno. Ako se vratimo na pojam struje s početka poglavlja, posljedično razlika potencijala odnosno električno polje utječe na kretanje elektrona tj. električnu struju unutar nekog vodiča.

Ako polje cijelo vrijeme zadržava isti smjer, radi se o istosmjernoj struji elektrona. Ako polje mijenja smjer tada se radi o izmjeničnoj struji elektrona. Važno je naglasiti da se elektroni pri svom gibanju kroz vodič gibaju suprotno od smjera vanjskog električnog polja(električne struje). Odgovor zašto je to tako leži u tome da je po definiciji smjer električnog polja smjer kojim bi se gibao pozitivni naboj, a pošto je elektron negativno nabijena čestica, giba se suprotno od smjera električnog polja prema Franklinovoj teoriji elektriciteta.[2]



Slika 1. Prikaz naelektriziranog štapa pomoću krpe [3]

3. GEORG SIMON OHM

Georg Simon Ohm je bio njemački fizičar i matematičar rođen 1789. godine u njemačkoj pokrajini Bavarskoj. Od svoga djetinjstva zanimao se za područja fizike i matematike, posebno elektriciteta i magnetizma. Godine 1820. zainteresirao se za rad danskog fizičara Hans Christiana Ørsteda koji je eksperimentirao s magnetskom iglom i baterijom te zaključio kako istosmjerna struja iz baterije može zakretati magnetsku iglu.

Time je Ørsted zapravo po prvi puta doveo u vezu magnetizam i elektricitet. Ohm je već 1825. odlučio pokrenuti eksperimentalni rad na temelju Ørstedovih otkrića. Zanimalo ga je kako duljina žice kojom teče struja elektrona utječe na tu struju. Došao je do zaključka da što je duža žica to je slabija jakost struje elektrona kroz nju. No, iz tog eksperimenta nije mu bilo jasno zašto se to točno događa pa je nastavio sa svojim eksperimentima. Ujedno je eksperimentirajući s raznim žicama uveo pojam vodljivosti različitih materijala.[4],[6]

U to vrijeme uvelike je pri eksperimentima korišten tzv. termočlanak – uređaj koji se sastoji od dva različita spojena metala čijim zagrijavanjem se pobuđuju elektroni unutar metala što stvara razliku potencijala (napon). Godine 1826. Ohm je iskoristio taj uređaj kako bi vidio što se događa s električnom strujom za različite duljine žica. Uvidio je kako se jakost električne struje smanjuje s duljinom žice, a osim duljine žice jakost struje ovisila je i o veličinama a i b (vidi sliku 4.). Daljnjim eksperimentima došao je do zaključka kako veličina a predstavlja neku vrstu „sile pobuđenja“ kako ju je u to vrijeme nazvao dok se veličina b nije zamjetnije mijenjala. Veličina X koja je u njegovim izračunima bila proporcionalna električnoj struji sada je ovisila proporcionalno o sili pobuđenja i obrnuto proporcionalno o njenoj duljini (on ju je nazvao otporna duljina). Silu pobuđenja danas zovemo naponom, a otpornu duljinu jednostavno otporom. To značajno otkriće prvi put je objavio 1826. godine u svom članku pod nazivom „*Determination of the Law According to which Metals Conduct Contact Electricity, Together with the Outlines of a Theory of Volta's Apparatus and Schweigger's Multiplicator*“[5]

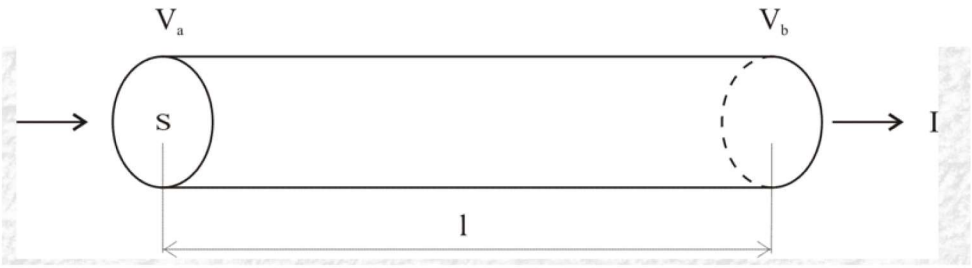
Slika 2. Ilustracija Georg Simon Ohma na kojoj mjeri električnu struju koja prolazi kroz bakrenu žicu pomoću termočlanka [7]

3.1. Ohmov zakon

Nakon što je eksperimentalno dobio ovisnost jakosti struje elektrona kroz vodič o naponu i otporu, Ohm je odlučio još malo matematički usavršiti svoj zakon. Godine 1827. objavljuje knjigu pod nazivom *„The Galvanic Circuit Investigated Mathematically“* koja je bila na lošem glasu kao nejasna i matematički prekomplicirana, što je Ohma jako razočaralo. Također, njegovi zaključci bili su u suprotnosti s dotad postavljenim zakonima. Među ostalim, vjerovalo se kako jakost struje elektrona nije ovisna o naponu kroz vodič. Zanemarujući unutarnji otpor u izvorima elektriciteta (baterijama) znanstvenici su došli do krivog zaključka kako povećanje broja baterija ne utječe na jakost struje kroz vodič. Više baterija značilo je više napona, ali također i više unutarnjeg otpora što je, gledajući nekoliko godina kasnije, prema Ohmovom zakonu značilo da jakost struje elektrona praktički ostaje nepromijenjena.[5] Ohmov zakon je počeo biti cijenjen tek u Engleskoj oko 1837., kada je objavljen članak o izumu motora ruskog izumitelja Moritza von Jacobia u novinama engleskog fizičara Williama Sturgeona. Naime, Jacobi se tokom izrade svog modela motora oslonio na zaključke koje je donio Ohm svojim zakonom tvrdeći kako je njegov zakon uvelike olakšao shvaćanje i izradu izuma iz područja elektrodinamike. Veliku potporu radu Georg Simon Ohma dao je i Sir Charles Wheatstone, engleski fizičar koji je bio zaposlen kao profesor na sveučilištu King's College u Londonu. Pročitavši Ohmove eksperimentalne rezultate odlučio ga je promovirati prijevodom njegovih radova do 1841. godine kada je napokon i šira publika počela upoznavati Ohmov zakon i zaključke koji iz njega proizlaze. Godine 1861. u čast Georg Simon Ohmu *„British Association for the Advancement of Science“* odlučilo je međunarodnoj jedinici za otpor dodijeliti naziv „om“, a za simbol su uzeli zadnje slovo grčkog alfabeta omega.[6],[8]

3.2. Izvod Ohmovog zakona

Promatramo vodič duljine l i konstantnog presjeka S kojim teče struja jakosti I. Na krajevima vodiča nalaze se potencijali V_a i V_b.



Slika 3. Prikaz vodiča kojim teče struja *I* [2]

Uvodimo konstantu κ koja se naziva provodnost i govori o tome koliko dobro neki materijal provodi struju. To je omjer gustoće struje J i jakosti električnog polja E. Iz te ovisnosti možemo izvesti *jednadžbu vodljivosti* koja kaže da je gustoća struje proporcionalna električnom polju.

$$\kappa = \frac{J}{E} \rightarrow J = \kappa E$$

U praksi ova ovisnost nam nije baš nije idealna za mjerenje pa ćemo je malo preoblikovati.

Gustoću struje J možemo zamijeniti s omjerom jačine struje i jedinične površine kroz koju struja prolazi. Nadalje, jakost električnog polja E ćemo zapisati negativnim gradijentom potencijala V

$$I = -\kappa S \frac{dV}{dx}$$

Iz gornjeg izraza želimo dobiti vezu između struje u vodiču i razlike potencijala na njegovim krajevima pa koristimo metodu parcijalne integracije

$$Idx = -\kappa S dV$$

$$I \int_0^l dx = -\kappa S \int_{V_a}^{V_b} dV$$

$$Il = -\kappa S (V_b - V_a)$$

$$I = \frac{\kappa S}{l}(V_a - V_b)$$

Gornji omjer naziva se električna vodljivost (konduktancija)

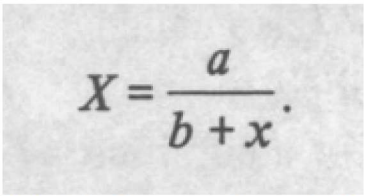
$$G = \frac{\kappa S}{l}$$

Recipročna vrijednost električne vodljivosti naziva se otpor

$$R = \frac{1}{G}$$

Uvrštavanjem gornjeg izraza u izraz kojeg smo dobili za jakost struje *I* dobivamo poznatu jednadžbu *Ohmovog zakona* koja nam govori da je jakost električne struje razmjerna razlici potencijala na njegovim krajevima [2], [9]

$$I = \frac{V_{ab}}{R}$$



Slika 4. Jednadžba Ohmovog zakona iz 1826. [5]

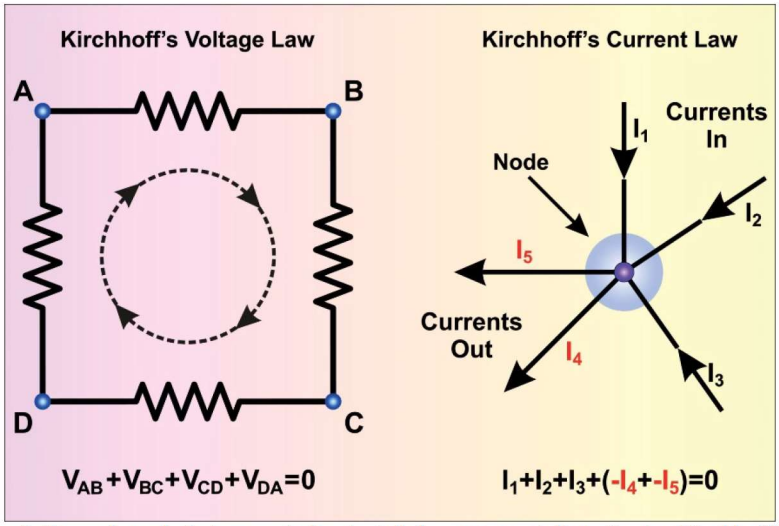
4. GUSTAV ROBERT KIRCHHOFF

Gustav Robert Kirchhoff je bio njemački fizičar 19. stoljeća rođen u Königsbergu u njemačkoj pokrajini Prusiji (današnji Kalinjingrad, glavni grad ruske enklave Kalinjingradske oblasti). Zanimao se za fiziku električnih sklopova, spektroskopiju,

zračenje crnog tijela, a danas je najpoznatiji po svojim zakonima za električne strujne krugove koji su pomogli u razvoju mnogobrojnih električnih sklopova i uređaja (primjerice Wheatstoneov most za mjerenje otpora se temelji na Kirchhoffovim zakonima). Danas, bez njegovih zakona, složeni uređaji čiji temelj su složeni strujni krugovi ne bi se mogli objasniti. Svoje zakone postavio je već 1845. dok je bio student na sveučilištu u Königsbergu.

Osim pravila za strujni krug zanimao se i za druga područja pa je tako 1857. pozvan na sveučilište u gradu Heidelbergu u Njemačkoj kako bi pomogao znanstveniku i kemičaru Robertu Bunsenu u njegovom radu. Naime, Bunsen i Kirchhoff su otkrili kako električni signal kroz tanku žicu gotovo bez otpora putuje brzinom svjetlosti.

U području spektroskopije iskazao se postuliranjem tri zakona koja su se odnosila na svjetlost emitiranu iz užarenih objekata. U optici je pronašao način kako najbolje objasniti Huygensov princip za valove rješavanjem i raščlambom valne jednadžbe.[10]



Slika 5. Dva zakona koje je za potrebe strujnih krugova postulirao Gustav Kirchhoff [11]

4.1. Kirchhoffovi zakoni

Za razliku od Ohmovog zakona koji vrijedi općenito, ali koristi se većinom za jednostavnije strujne krugove Kirchhoff je 1845. postulirao dva zakona za složene strujne krugove koji su danas poznati kao Kirchhoffovi zakoni. Koristeći se prethodnim znanjem i zaključcima od Ohma, Gustav Kirchhoff je otkrio kako na efikasan način računati

jakosti struja, otpore i napon u složenim spojevima. Kirchhoffovi zakoni se temelje na zakonima o očuvanju količine naboja i energije u električnim sklopovima. U složenoj električnoj mreži postoji nekoliko važnih karakterističnih dijelova: čvorovi, petlje i grane strujnog kruga. Grane predstavljaju vodiče kojim teče električna struja u strujnom krugu. Na granama se također nalaze i izvori napona (npr. baterije) te otpornici. Čvorovi su mjesta gdje se spajaju grane, dok su petlje mjesta gdje postoji zatvoreni krug u električnoj mreži gdje struja teče kružno.

Prvi Kirchhoffov zakon (Kirchhoffov zakon o struji) nalaže da suma jakosti struja koje ulaze u čvor mora biti jednaka sumi jakosti struja koje iz njega izlaze tj. da je zbroj struja u čvoru jednak nuli. Posljedica takvog zakona je činjenica da se u vodiču naboj ne nagomilava. Prema dogovoru struje koje ulaze u čvor imaju pozitivan predznak, a struje koje izlaze iz čvora negativan.

Drugi Kirchhoffov zakon (Kirchhoffov zakon o naponu) nalaže da u svakoj zatvorenoj petlji zbroj svih elektromotornih sila je jednak zbroju svih padova napona na otporniku. Drugi Kirchhoffov zakon je posljedica zakona očuvanja energije. Po dogovoru za svaku petlju se odabire fiksan način obilaska petlje. Prema tome, padovi napona bit će pozitivnog predznaka ako je smjer struje kroz otpornik u smjeru obilaženja petlje, a padovi napona bit će negativnog predznaka ako je smjer struje kroz otpornik suprotan od smjera obilaženja petlje.

Koristeći gore spomenute Kirchhoffove zakone za svaki strujni krug moguće je dobiti sustav algebarskih jednadžbi koji će dati rezultate o otporu, naponu ili jakosti struje. Tijekom izračuna može se dogoditi da se dobije negativna vrijednost jakosti struje. U takvim slučajevima nije bilo pogreške, osim što je izabran pogrešan način obilaženja petlje.[12]

5. WHEATSTONEOV MOST ZA MJERENJE OTPORA

Wheatstoneov most za mjerenje otpora je eksperimentalni postav imenovan po znanstveniku koji ga je najviše popularizirao u tadašnje vrijeme, Sir Charlesu Wheatstoneu. Iako je prvi znanstvenik koji je potpuno opisao koncept i ideju postava koji bi mogao mjeriti nepoznate otpore engleski znanstvenik Samuel Hunter Christie, tadašnji znanstveni krugovi pripisali su zasluge Wheatstoneu. Hunter Christie je svoju ideju stavio na papir 1833. godine dok je tražio najjednostavniji i najpraktičniji način kako u strujnom krugu, pomoću drugih otpornika, dobiti vrijednosti nekog otpornika čiji otpor ne znamo

unaprijed. Wheatstone je u svom radu 1843. uzeo koncept strujnog kruga kojeg je prvi osmislio gore spomenuti Hunter Christie. Eksperimentalni postav je bio strujni krug u obliku dijamanta. Električna struja iz izvora (baterije) bi se granala u dvije paralelne grane. Prva grana je imala jedan fiksan i jedan promjenjivi otpornik. Druga grana je imala još jedan fiksan otpornik i jedan čiji otpor ne znamo unaprijed. Između te dvije grane stajao je galvanometar koji je mjerio jakost električne struje između dvije grane (kao most). Upravo zbog tog galvanometra se cijeli postav naziva mostom. U to vrijeme galvanometar se koristio kao most između dvije grane zbog njegove velike osjetljivosti na male promjene u jakosti električne struje. Napretkom znanosti i tehnologije sve više možemo uočiti da se u crtežima ovakvog eksperimentalnog postava koristi voltmetar koji mjeri razliku potencijala između grana.

5.1. Sir Charles Wheatstone

Sir Charles Wheatstone bio je engleski filozof i izumitelj rođen u mjestu Barnwood 1802. godine. Od ranih godina zanimao se za fiziku, matematiku i francuski jezik.[13] Bio je nadaren za izume i inovacije pa već sa 16 godina izrađuje vlastiti instrument „*flute harmonique*“. Njegova inovativnost je zaintrigirala danskog fizičara Ørsteda koji mu je 1823. predložio da objavi svoj prvi znanstveni članak na temelju fizike glazbe. Iako nije imao puno obrazovanja u poljima koji su ga zanimali, zaposlio se kao profesor na sveučilištu King's College u Londonu. Na sveučilištu se bavio raznim eksperimentima iz polja elektrotehnike i elektronike, nadogradio dinamo, osmislio uređaj reostat koji je mogao pomoću kliznog otpornika mijenjati jakost struje koja teče u strujnom krugu.[13] Veliku popularnost je stekao radeći sa znanstvenikom Williamom Cookeom koji je zajedno sa Wheatstoneom 1837. osmislio električni telegraf koji je pomogao u tadašnjoj komunikaciji na velikim udaljenostima. Značajno je i to što je njihov izum bio prvi takav u Engleskoj što je Wheatstone odvelo do mjesta počasnog člana Kraljevskog društva Britanije, organizaciju u koju su ulazili samo odabrani pojedinci koji su se iskazali u znakovitim djelima za dobrobit znanosti, čovječanstva i društva općenito. Wheatstone je možda i najpoznatiji po tome što je 1843. godine poboljšao (dodao reostat) i vratio u uporabu zaboravljeni uređaj za mjerenje otpora kojeg je isprva osmislio i rabio engleski fizičar Samuel Hunter Christie. Uređaj se danas naziva Wheatstoneov most za mjerenje otpora.[14]

Slika 6. Koncept Christievog mosta za mjerenje otpora [15]

Slika 7. Samuel Hunter Christie [16]

Slika 8. Sir Charles Wheatstone [16]

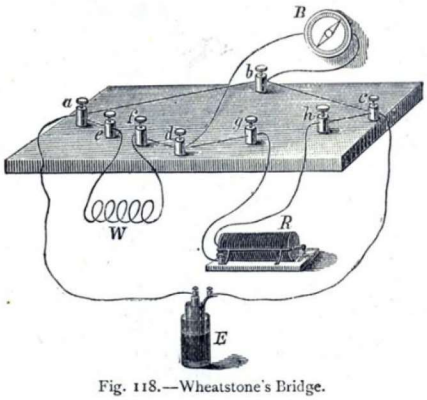
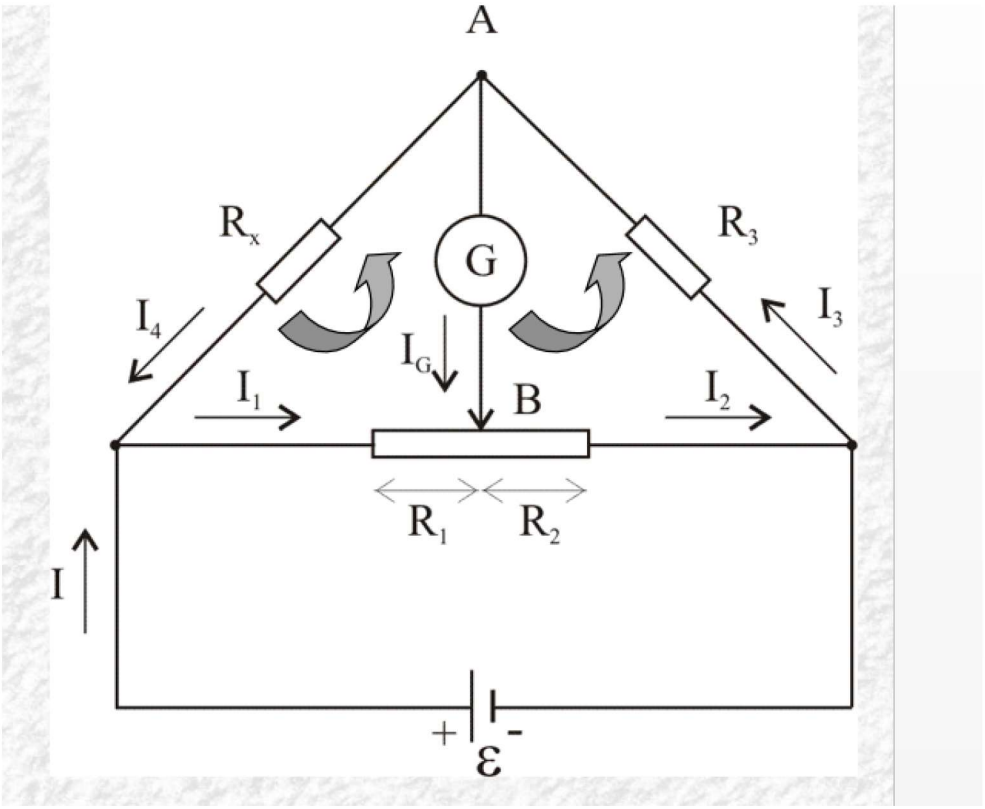


Fig. 118.—Wheatstone's Bridge.

Slika 8. Koncept Wheatstoneovog mosta za mjerenje otpora [17]

5.2. Mjerenje nepoznatog otpora pomoću Wheatstoneova mosta

Za potrebe ovog izvoda koristit ću slijedeću shemu



Slika 10. Prikaz jednog oblika Wheatstoneovog mosta za mjerenje otpora [2]

Na shemi je prikazan spoj izvora struje (baterija), 4 otpornika te galvanometra (koji mjeri jakost električne struje). Rx je otpornik nepoznatog otpora, R_3 je poznati fiksni otpornik dok su otpornici R_1 i R_2 predstavljeni pomoću kliznika (reostata) kojim se reguliraju njihovi otpori pa ih možemo nazvati i promjenjivi. Ono što je naš cilj je iskoristiti Kirchhoffove zakone za strujne krugove pomoću kojih ćemo dobiti jednadžbe za dvije karakteristične točke strujnog kruga A i B. Iz tih jednadžbi iskoristimo činjenicu da će galvanometar pokazivati jakost struje 0 kada je most uravnotežen između te dvije točke.

Za točke A i B iskoristimo prvi Kirchhoffov zakon

$$I_3 = I_4 + I_G$$

$$I_1 + I_G = I_2$$

Iskoristimo i drugi Kirchhoffov zakon

$$I_1 R_1 - I_G R_G + I_4 R_x = 0$$

$$I_2 R_2 + I_3 R_3 + I_G R_G = 0$$

Za Wheatstoneov most mora vrijediti

$$I_G = 0$$

$$I_1 = I_2$$

$$I_3 = I_4$$

Dobivamo jednadžbe

$$I_1 R_1 + I_4 R_x = 0$$

$$I_2 R_2 + I_3 R_3 = 0$$

$$I_1 R_1 = -I_4 R_x$$

$$I_2 R_2 = -I_3 R_3$$

Podijelimo dvije jednadžbe jednu sa drugom

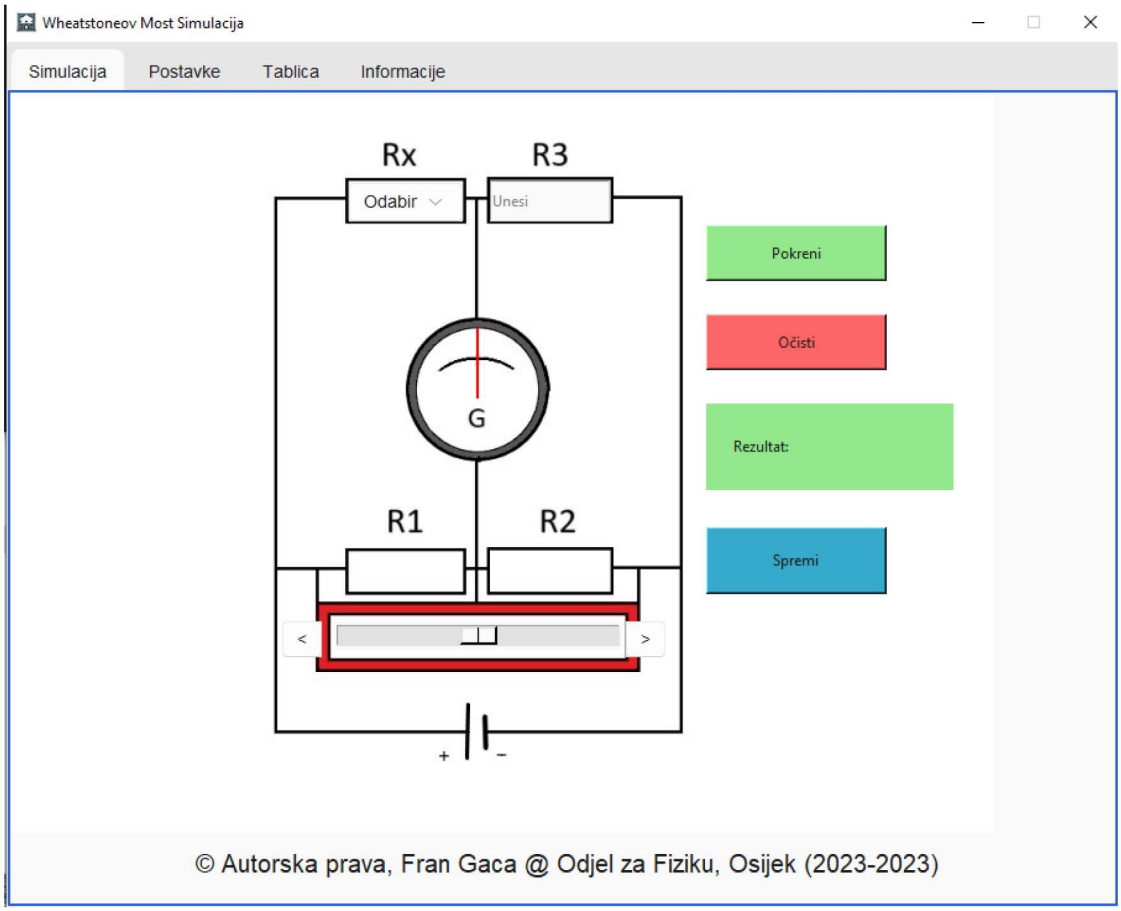
$$\frac{\cancel{I_1} R_1}{\cancel{I_2} R_2} = \frac{-\cancel{I_4} R_x}{-\cancel{I_3} R_3}$$

$$\frac{R_1}{R_2} = \frac{R_x}{R_3} \rightarrow R_x = R_3 \frac{R_1}{R_2}$$

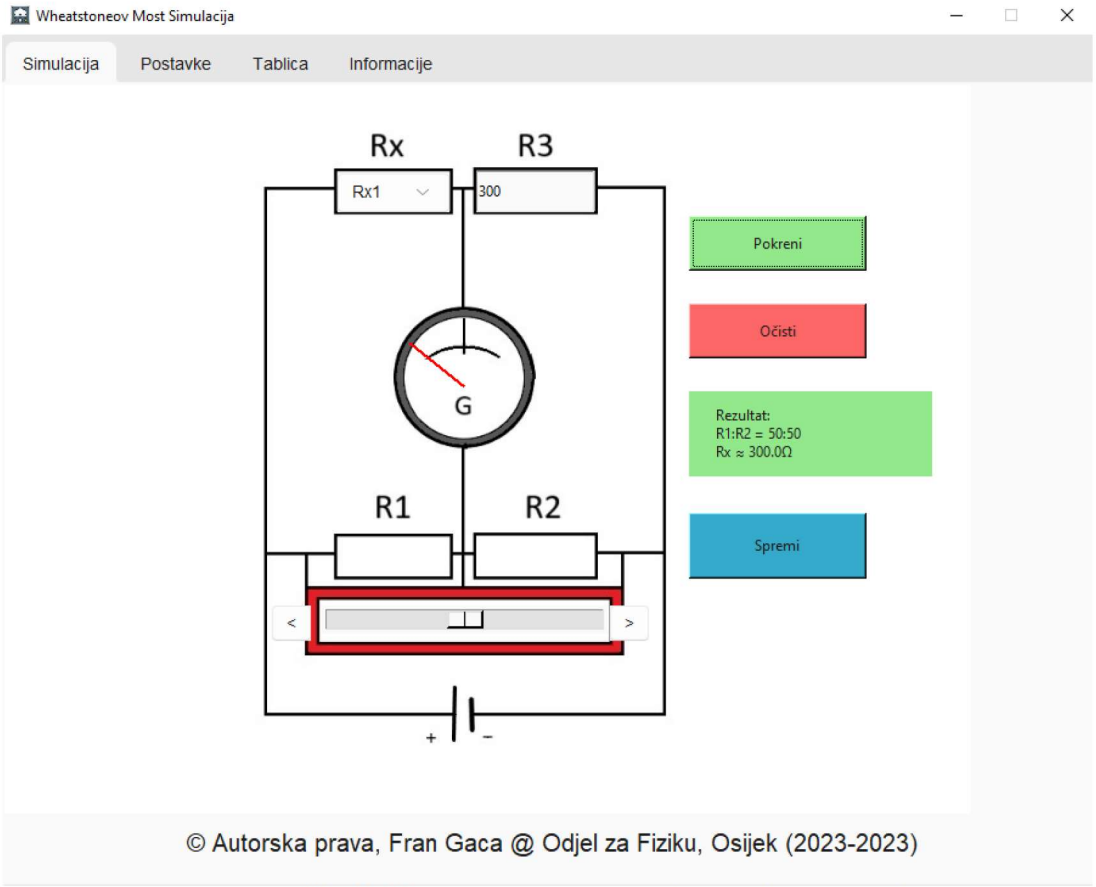
Dobili smo izraz za izračunavanje otpora nepoznatog otpornika koristeći metodu Wheatstoneovog mosta.[2],[7]

6. Interaktivna simulacija Wheatstoneovog mosta za mjerenje otpora izrađena u Pythonu

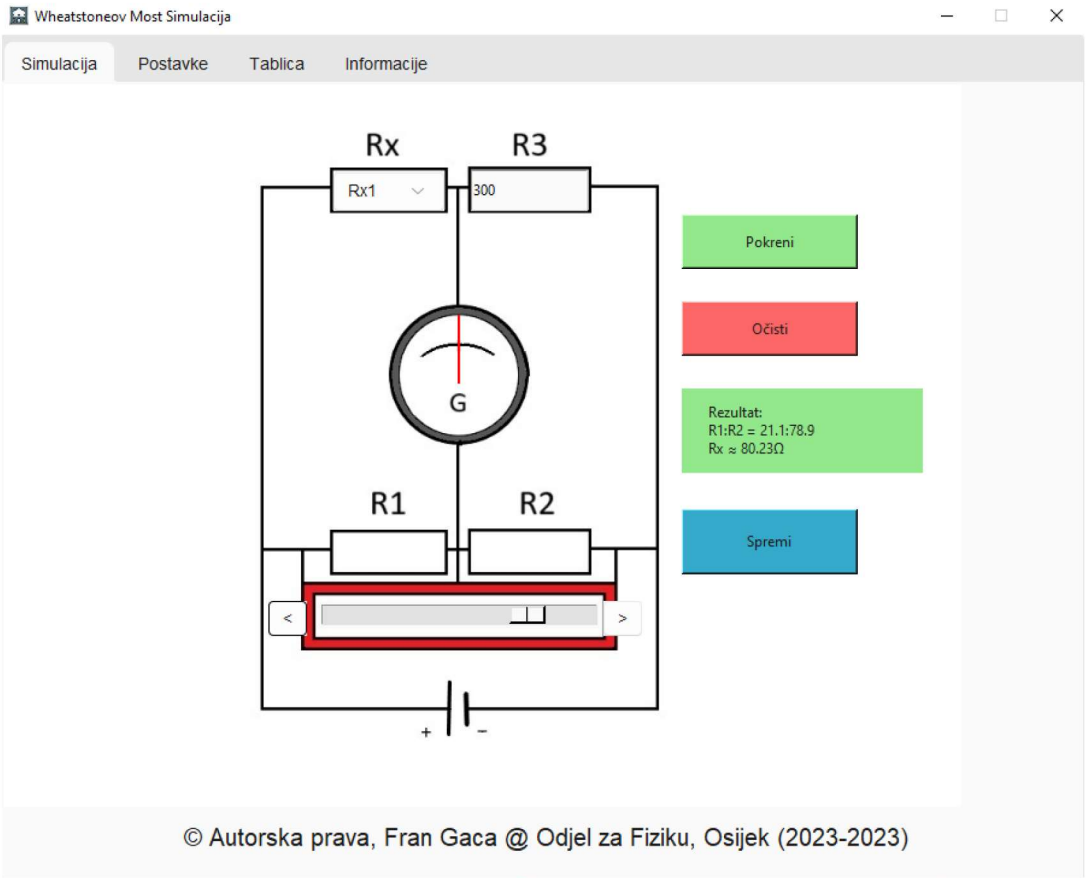
Za potrebe ovog Završnog rada izradio sam interaktivnu simulaciju u programskom jeziku Python (ver. 3.11) koja se temelji na zaključcima i zakonima koji su izveli Georg Simon Ohm, Gustav Robert Kirchhoff, Samuel Hunter Christie te Sir Charles Wheatstone. Kod je priložen u Dodatku. Interaktivna simulacija je napravljena u nadi da će pomoći učenicima naučena teorijska znanja primijeniti u stvarnim fizičkim eksperimentima, a također i potaknuti ih da u procesu učenja sve više koriste eksperimente i vizualno prezentirana rješenja raznih fizikalnih problema.



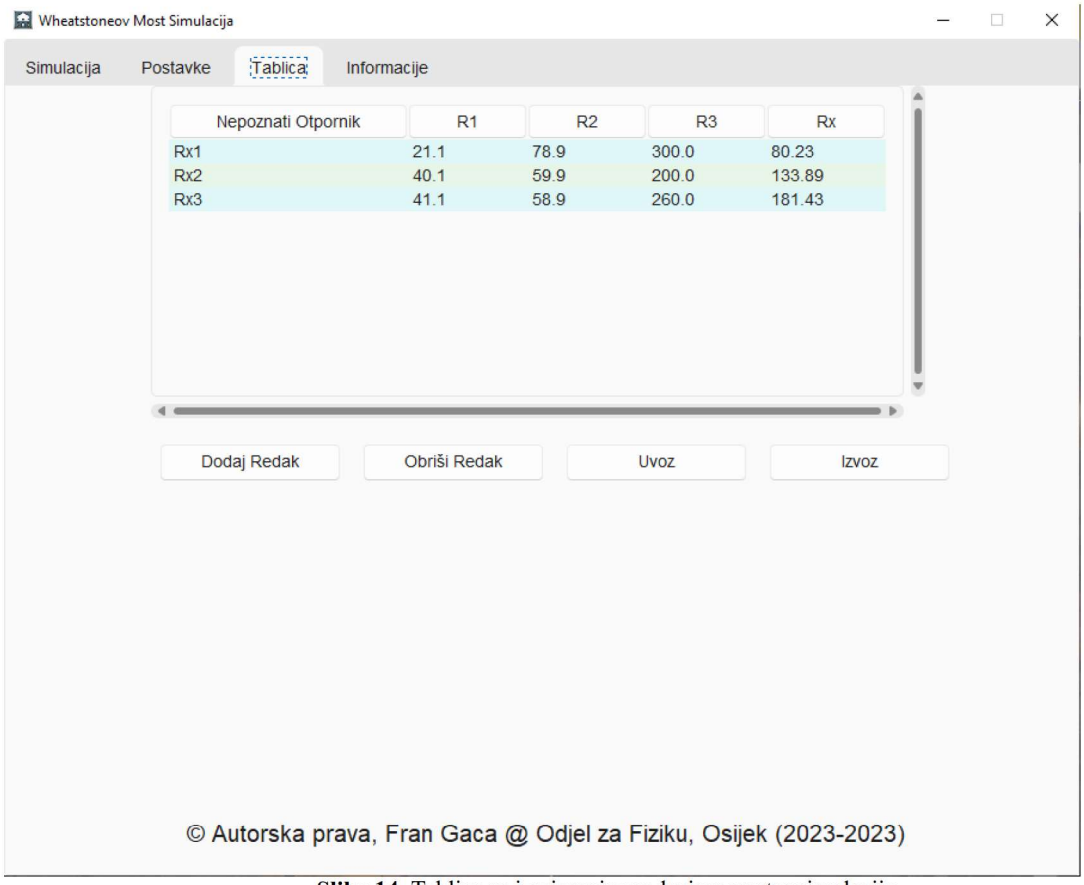
Slika 11. Interaktivna simulacija Wheatstoneovog mosta za mjerenje otpora



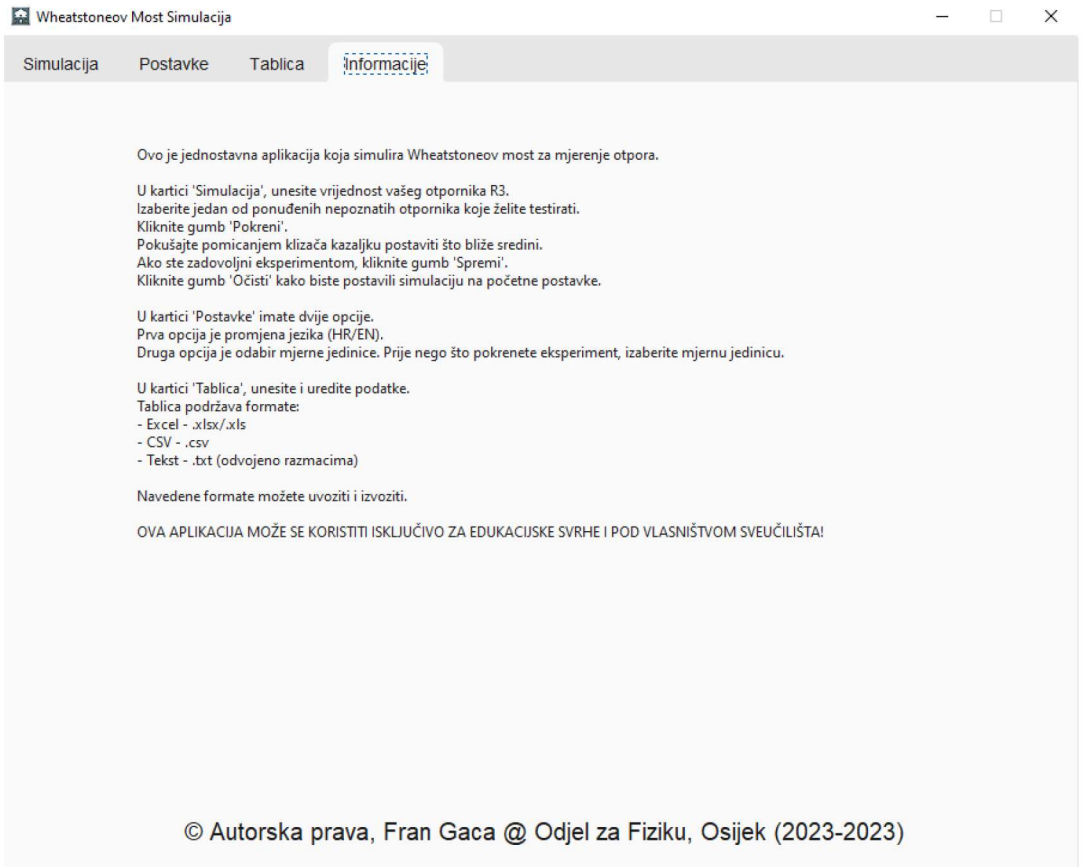
Slika 12. Wheatstoneov most za mjerenje otpora izvan ravnoteže



Slika 13. Wheatstoneov most za mjerenje otpora u ravnoteži



Slika 14. Tablica sa izmjerenim podacima unutar simulacije



Slika 15. Informacije i upute za korištenje simulacije

6.1. Upute za korištenje interaktivne simulacije

Simulacija se sastoji od korisničkog sučelja u kojemu korisnik unosi željene podatke o otporima otpornika. Slično kao što su to osmislili Christie i Wheatstone, strujni krug je zamišljen kao zatvorena petlja. Unutar kruga se nalazi izvor napona, jedan nepoznati fiksni otpornik Rx, drugi otpornik proizvoljnog otpora R3, klizač čiji položaj određuje omjer otpora dvaju otpornika R1 i R2 te galvanometar koji pokazuje jakost električne struje kroz galvanometar. Korisnik odabire jedan fiksni otpornik nepoznatog otpora predstavljen oznakama (Rx1, Rx2, Rx3..). Nakon toga korisnik unosi po želji iznos poznatog otpornika R3 u polje za unos. Nakon unosa korisnik koristi klizač u donjem dijelu strujnog kruga kako bi strujni krug doveo u ravnotežu, odnosno kako bi postigao da galvanometar strujnog kruga pokazuje približno 0 ampera. Nakon postizanja ravnoteže u strujnom krugu dobivene podatke korisnik jednostavnim klikom miša unosi u tablicu mjerenja. Ovime je mjerenje završeno, a korisnik može mjeriti traženi otpor i više puta. Poželjno je da se vrijednost svakog nepoznatog otpornika mjeri unošenjem nekoliko vrijednosti poznatog otpornika R3 (barem 5 različitih vrijednosti otpornika R3). Nakon završetka mjerenja korisnik može obrisati podatke iz tablice i pripremiti ju za iduće mjerenje, izvesti podatke u potrebni format ili jednostavno izaći iz simulacije klikom na gornji desni dio prozora.

6.2. O programskom jeziku Python

Programski jezik Python je interpretativni, objektno orijentirani programski jezik kojeg je kreirao nizozemski programer Guido van Rossum, a prva verzija je puštena u javnost 20. veljače 1991. godine. Ideju za ime programskog jezika van Rossum je dobio od BBC-eve televizijske komedije *Monty Python's Flying Circus*. Zanimljivo, Python danas nadograđuju i popravljaju mnogi drugi programeri okupljeni u organizaciju pod nazivom *Python Software Foundation*.

Prednosti korištenja ovog programskog jezika su lagano učenje, pristupačno sučelje, jednostavno instaliranje, nadogradnja i popravak programskog jezika. Ako se odlučite započeti programirati Python je odličan za početnike. Ne zamara nepreglednim i kompliciranim sučeljem, a samo izvođenje programa je brzo i efikasno. Python se kao programski jezik pokazao kao vrlo dobar izbor za razvoj web stranica, strojno učenje, inženjering, te razne druge primjene.[18]

7. ZAKLJUČAK

Prvu eksperimentalnu postavu Wheatstoneovog mosta za mjerenje otpora predložio je engleski znanstvenik Samuel Hunter Christie 1833. godine, a realizirao ju je engleski znanstvenik Sir Charles Wheatstone desetak godina kasnije, točnije 1843. godine. Wheatstone je u Christievu ideju dodao reostat tj. promjenjivi otpornik pomoću kojeg se moglo uravnotežiti strujni krug. Wheatstoneov most za mjerenje otpora počiva, osim na Christievoj ideji, na Ohmovom i Kirchhoffovim zakonima[14]. Georg Simon Ohm bio je engleski fizičar koji je postulirao danas možda i jedan od najvažnijih zakona u području fizike strujnih krugova. Svoju ideju kako je jakost električne struje proporcionalna naponu, a obrnuto proporcionalna otporu kroz strujni krug iznio je u knjizi „The Galvanic Circuit Investigated Mathematically“ izdanoj 1827. godine.[6] Isprva je njegov rad bio proglašavan zaostalim, prekompliciranim i netočnim. Slavu je stekao tek 1837. godine kada je u jednim znanstvenim novinama izašao članak o izumu ruskog izumitelja Moritza von Jacobia koji je tvrdio kako se njegov rad temeljio na Ohmovom zakonu i Ohmovim zaključcima. Još jedan važan znanstvenik koji je doprinio uspjehu Wheatstoneovog mosta bio je Gustav Robert Kirchhoff, njemački fizičar koji je iskazao dva zakona za strujne krugove koji danas čine temelj moderne elektronike i fizike strujnih krugova.

Danas se Wheatstoneov most za mjerenje otpora koristi u mnogim poljima elektrotehnike i elektronike. Primjerice za mjerenje otpora, elektroničke detektore, za mjerenje promjene jačine svjetla, tlaka, naprezanja. Napretkom znanosti i tehnologije Wheatstoneov izum je danas nadograđen kako bi mogao služiti široj svrsi. Takve modifikacije su primjerice Kelvinov most (metoda senzora s četiri priključka), Carey – Foster most (služi mjerenju malih otpora) te Maxwellov most (služi mjerenju samoindukcije u strujnim krugovima).[14]

Glavni zadatak ovog završnog rada bila je izrada interaktivne simulacije laboratorijske vježbe mjerenja otpora Wheatstoneovim mostom. Program nastoji što vjernije prikazati „živo“ mjerenje otpora, a svrha mu je da se učenici što bolje pripreme za realnu vježbu u laboratoriju. Nadam se da će program pomoći učenicima prevladati strah od eksperimenta.

8. LITERATURA

[1] Stari vijek – elektricitet i magnetizam
URL: https://ahyco.uniri.hr/povijestfizike/stari_elektricitet.htm
(Pristupljeno: 07.08.2023.)

[2] Power Point prezentacija Izv. prof. dr. sc. Branka Vukovića „OSNOVE FIZIKE 2 Električna struja“; 2013./2014.

[3] Halliday D.; Resnick R.; Walker J. (2004). Fundamentals of Physics, 10th Ed

[4] 19. stoljeće – elektricitet i magnetizam
URL: http://ahyco.uniri.hr/povijestfizike/19_elektricitet.htm
(Pristupljeno: 07.08.2023.)

[5] Gupta, M.(1980). Georg Simon Ohm and Ohm 's Law
URL: Georg_Simon_Ohm_and_Ohm's_Law
(Pristupljeno: 01.08.2023.)

[6] Ohm’s Law: History and Biography of George Ohm; History of Science by Kathy Joseph, August 24, 2021
URL: https://kathylovesphysics.com/history-and-biography-of-george-ohm/
(Pristupljeno: 07.08.2023.)

[7] Georg Simon Ohm, the law that bears his name represented the true beginning of electrical circuit analysis
URL: https://rinconeducativo.org/en/recursos-educativos/georg-simon-ohm-the-law-that-bears-his-name-represented-the-true-beginning-of-electrical-circuit-analysis/
(Pristupljeno: 09.08.2023.)

[8] Georg Ohm
URL: https://nationalmaglab.org/magnet-academy/history-of-electricity-magnetism/pioneers/georg-ohm/

(Pristupljeno: 09.08.2023.)

[9] Cindro, N. (1988). Fizika 2: Elektricitet i magnetizam

[10] Gustav Robert Kirchhoff; written by J J O'Connor and E F Robertson, Last Update August 2002
URL: https://mathshistory.st-andrews.ac.uk/Biographies/Kirchhoff/
(Pristupljeno: 09.08.2023.)

[11] How to use Kirchhoff 's laws for complex circuits
URL: https://www.autodesk.com/products/fusion-360/blog/kirchhoffs-law-for-complex-circuits/
(Pristupljeno 09.08.2023.)

[12] Young, Hugh D.; Freedman, Roger A.; Ford, A. Lewis (2011). Sears & Zemansky 's University Physics with Modern Physics Young and Freedman, 13th Edition

[13] Sir Charles Wheatstone
URL: https://victorianweb.org/technology/inventors/wheatstone.html
(Pristupljeno: 11.08.2023.)

[14] Wheatstone Bridge; March 3, 2020, by EngineeringClicks
URL: https://www.engineeringclicks.com/wheatstone-bridge/
(Pristupljeno: 11.08.2023.)

[15] Wheatstone Bridge – 1843
URL: history-of-electricity-magnetism/museum/wheatstone-bridge-1843/
(Pristupljeno: 11.08.2023.)

[16] National Portrait Gallery
URL: https://www.npg.org.uk/collections/search/person/
(Pristupljeno 11.08.2023.)

[17] Sir Charles Wheatstone: Father of the Wheatstone Bridge and British Electric Telegraph
URL: https://interestingengineering.com/culture/sir-charles-wheatstone-father-of-the-wheatstone-bridge-and-british-electric-telegraph
(Pristupljeno 11.08.2023.)

[18] O Pythonu
URL: https://pythoninstitute.org/about-python
(Pristupljeno: 26.08.2023.)

9. ŽIVOTOPIS

Fran Gača rođen je 10. listopada 2000. godine u Slavonskom Brodu, Republika Hrvatska. Pohađao je osnovnu školu Ivana Brlić – Mažuranić u Slavonskom Brodu (2007. – 2015.) nakon čega upisuje Gimnaziju „Matija Mesić“ općeg smjera u Slavonskom Brodu (2015. – 2019.). Godine 2019. započinje školovanje na Odjelu za fiziku Sveučilišta Josipa Jurja Strossmayera u Osijeku, nastavnički smjer. U listopadu 2020. godine nakratko upisuje Prirodoslovno – matematički fakultet Sveučilišta u Zagrebu, smjer geofizika. Od listopada 2022. nastavlja studij na Odjelu za fiziku Sveučilišta Josipa Jurja Strossmayera u Osijeku na kojem se i sada školuje. Tečno govori hrvatski i engleski jezik te poznaje osnove njemačkog jezika. Ima iskustva rada u raznim programima za obradu teksta i podataka te programskom jeziku Python.

10. DODATAK

Ovo je gotovi kod za interaktivnu simulaciju Wheatstoneovog mosta za mjerenje otpora napravljenu u programskom jeziku Python (v. 3.11).

Najprije ubacujem potrebne pakete koje ćemo koristiti pri izradi simulacije:

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
import math
import json
import sv_ttk
import random
import os
import pandas as pd
import csv
from datetime import datetime
import openpyxl
```

Definirat ću neke naredbe pomoću kojih će program otvarati buduće JSON fileove:

```
with open("config.json", "r", encoding="utf-8") as f:
    config = json.load(f)

    height = config["height"]
    width = config["width"]
    layout_image_path = config["layout_image"]
    resistors = config["resistors_ohm"]
    max_error_percentage = config["max_error_percentage"]

    slider_info = config["slider"]
    epsilon = slider_info["epsilon"]
    units = config["units"]

#with open("languages/en_EN.json", "r", encoding="utf-8") as f:
with open("languages/cr_CR.json", "r", encoding="utf-8") as f:
    language_data = json.load(f)
```

Definirat ću i prozor koji će sadržavati našu simulaciju uz navođenje tipičnih podataka o veličini samog prozora te njegovom nazivu:

```
# Init app
window = tk.Tk()

# Settings
window.title(language_data["app_name"])
screen_width = window.winfo_screenwidth()
screen_height = window.winfo_screenheight()
start_x = (screen_width - width) // 2
start_y = (screen_height - height) // 2
window.geometry(f"{width}x{height}+{start_x}+{start_y}")
window.resizable(width=False, height=False)
```

Definiram ikonu koja će predstavljati našu simulaciju u gornjem lijevom kutu prozora:

```
# App icon setup
icon = tk.PhotoImage(file=config["icon"])
window.tk.call("wm", "iconphoto", window._w, icon)
```

Dodajem u simulaciju potrebne kartice (tabove) pomoću kojih ćemo se orijentirati:

```
# Adding tabs
tabs = ttk.Notebook(master=window)
simulation_tab = ttk.Frame(master=tabs, width=1, height=1)
settings_tab = ttk.Frame(master=tabs, width=1, height=1)
table_tab = ttk.Frame(master=tabs, width=1, height=1)
info_tab = ttk.Frame(master=tabs, width=1, height=1)
```

```
tabs.add(child=simulation_tab, text=language_data["simulation_tab"]["tab_name"])
tabs.add(child=settings_tab, text=language_data["settings_tab"]["tab_name"])
tabs.add(child=table_tab, text=language_data["table_tab"]["tab_name"])
tabs.add(child=info_tab, text=language_data["info_tab"]["tab_name"])
tabs.pack(expand=True, fill="both")
```

Za spremanje podataka imamo posebnu karticu koja sadržava tabelu. Ovdje ću definirati što ta tablica sadrži te njene funkcije(uklanjanje, dodavanje redaka, unos i brisanje podataka):

```
### START OF TABLE TAB ###
def set_focus(event):
    event.widget.focus_set()

def clear_selection(event):
    pass
    tree.selection_remove(tree.selection())

def update_colors():
    for idx, row in enumerate(tree.get_children()):
        tag = "evenrow" if idx % 2 == 0 else "oddrow"
        tree.item(row, tags=(tag,))

def on_item_double_click(event):
    row_id = tree.identify_row(event.y)
    col = tree.identify_column(event.x)
    if row_id == "":
        return
    x, y, width, height = tree.bbox(row_id, col)

    cell_value = tree.item(row_id, "values")[column_map[col]]
    entry = ttk.Entry(tree)
    entry.insert(0, cell_value)
    entry.place(x=x, y=y, width=width, height=height)

    def on_entry_focus_out(event):
        new_value = entry.get()
        tree.set(row_id, col, new_value)
        entry.destroy()

    entry.bind("<FocusOut>", on_entry_focus_out)
    entry.focus_set()

def add_row():
    cur_idx = len(tree.get_children())
    tag = "evenrow" if cur_idx % 2 == 0 else "oddrow"
    values = tuple("" for _ in range(len(tree["columns"])))
    tree.insert("", "end", iid=str(cur_idx), tags=(tag,), values=values)

def delete_row():
    selected_items = tree.selection()
    for item in selected_items:
        tree.delete(item)
    #update_colors()
    children = tree.get_children()
    rows_data = [tree.item(child, "values") for child in children]

    for child in children:
        tree.delete(child)

    for idx, row_data in enumerate(rows_data):
        tag = "evenrow" if idx % 2 == 0 else "oddrow"
        tree.insert("", "end", iid=str(idx), tags=(tag,), values=row_data)

    #print(tree.get_children())

def import_data():
    file_path = filedialog.askopenfilename(
        filetypes=[("CSV files", "*.csv"),
                   ("Text files", "*.txt"),
                   ("Excel files", "*.xlsx *.xls")]
    )
```

```
if not file_path:
    return

ext = os.path.splitext(file_path)[-1]

for c in tree.get_children():
    tree.delete(c)

if ext == ".csv":
    with open(file_path, "r") as csv_f:
        reader = csv.reader(csv_f)
        for row in reader:
            tree.insert("", "end", values=row)
elif ext == ".txt":
    with open(file_path, "r") as txt_f:
        for line in txt_f.readlines():
            row = line.strip().split(" ")
            tree.insert("", "end", values=row)
elif ext in [".xlsx", ".xls"]:
    df = pd.read_excel(file_path)
    for _, row in df.iterrows():
        tree.insert("", "end", values=row)
update_colors()
```

Za izvoz podataka u određeni format definiram tipove formata i način kako se podaci iz gore napravljene tabele mogu prenijeti u određeni format:

```
def export_data():
    file_path = filedialog.asksaveasfilename(
        defaultextension="*.*",
        filetypes=[("CSV files", "*.csv"),
                   ("Text files", "*.txt"),
                   ("Excel files", "*.xlsx *.xls")]
    )

    if not file_path:
        return

    if file_path.endswith(".csv"):
        with open(file_path, "w", newline="") as f:
            writer = csv.writer(f)
            #writer.writerow(tree["columns"])
            for row_id in tree.get_children():
                row_data = tree.item(row_id, "values")
                writer.writerow(row_data)
    elif file_path.endswith(".txt"):
        with open(file_path, "w") as f:
            #f.write("\t".join(str(header) for header in tree["columns"]) + "\n")
            for row_id in tree.get_children():
                row_data = tree.item(row_id, "values")
                f.write("\t".join(str(cell) for cell in row_data) + "\n")
    elif file_path.endswith((".xlsx", ".xls")):
        df = pd.DataFrame(columns=tree["columns"])
        for index, row_id in enumerate(tree.get_children()):
            row_data = tree.item(row_id, "values")
            df.loc[index] = row_data
        df.to_excel(file_path, index=False)

frame = ttk.Frame(table_tab, width=500, height=500)
style = ttk.Style(frame)
style.theme_use("clam")
style.configure("Treeview", rowheight=100)
```

Kako bi mogli uspješno koristiti tabelu definiram sve njezine dijelove i kliznike koji se koriste za bolji pregled podataka:

```
# Scrolls
y_scroll = ttk.Scrollbar(frame, orient="vertical")
x_scroll = ttk.Scrollbar(frame, orient="horizontal")

columns = ("Unknown Resistor", "R1", "R2", "R3", "Rx")
```

```
column_map = {f"#{i+1}": idx for idx, i in enumerate(range(len(columns)))}
tree = ttk.Treeview(frame,
                    columns=columns,
                    show="headings",
                    height=10,
                    padding="10 10",
                    selectmode="extended",
                    yscrollcommand=y_scroll.set,
                    xscrollcommand=x_scroll.set
                    )
tree.column("R1", width=100)
tree.column("R2", width=100)
tree.column("R3", width=100)
tree.column("Rx", width=100)
tree.column("#0", width=0, stretch="NO")
```

Ovdje definiram naslove kolona tabele:

```
# Headings
tree.heading("#1", text=language_data["table_tab"]["unknown_resistor_heading"])
tree.heading("#2", text="R1")
tree.heading("#3", text="R2")
tree.heading("#4", text="R3")
tree.heading("#5", text="Rx")
tree.tag_configure("evenrow", background="#E0F7FA")
tree.tag_configure("oddrow", background="#E8F5E9")
y_scroll.configure(command=tree.yview)
x_scroll.configure(command=tree.xview)
```

Ovdje definiram tipke koje služe dodavanju, brisanju retka, izvozu i uvozu podataka:

```
# Buttons
add_row_button = ttk.Button(table_tab, text=language_data["table_tab"]["add_row_button"],
command=add_row)
delete_row_button = ttk.Button(table_tab,
text=language_data["table_tab"]["delete_row_button"], command=delete_row)
import_button = ttk.Button(table_tab, text=language_data["table_tab"]["import_button"],
command=import_data)
export_button = ttk.Button(table_tab, text=language_data["table_tab"]["export_button"],
command=export_data)
```

U programu definiram i točne pozicije tipki:

```
# Packs, grid and place
frame.pack()
y_scroll.grid(row=0, column=1, sticky="nsew")
x_scroll.grid(row=1, column=0, sticky="nsew")
tree.grid(row=0, column=0, sticky="nsew")
add_row_button.place(x=130, y=300, width=150, height=30)
delete_row_button.place(x=300, y=300, width=150, height=30)
import_button.place(x=470, y=300, width=150, height=30)
export_button.place(x=640, y=300, width=150, height=30)
### END OF TABLE TAB ###
```

Nakon definiranja tabele i pomoćnih elemenata prelazim na karticu „Simulacija“ koja je i temeljni dio ovog programa. Ovdje se povezuje galvanometar sa kliznikom koji određuje omjer između dvaju otpornika R1 i R2 u simulaciji.
Kako bi imali konzistentan pomak kliznika sa ponašanjem galvanometra, definiramo kazaljku galvanometra kao vektor u jediničnoj kružnici i koristimo određena znanja iz područja matematike kako bi ponašanje galvanometra bilo ispravno:

```
### START OF SIMULATION TAB ###
rx = None
cur_rx = None
current_angle = float("-inf")
is_programmatic_update = False
```

```
def update_line_run(angle):
    x = unit_circle_center[0] + radius * math.cos(angle * math.pi / 180)
    y = unit_circle_center[1] - radius * math.sin(angle * math.pi / 180)
    canvas.coords(line_id, unit_circle_center[0], unit_circle_center[1], x, y)
    global current_angle, cur_rx
    current_angle = angle
    cur_rx = float(resistor_entry.get())
    #result_label.configure(text=f"{language_data['simulation_tab']['result_label']}:
   \nR1:R2 = 50:50\nRx ≈ {round(cur_rx / units[settings_clicked.get()],
   2)}{settings_clicked.get()}")
    result_label.configure(text="{}: \nR1:R2 = 50:50\nRx ≈ {}{}".format(
        language_data['simulation_tab']['result_label'],round(cur_rx /
  units[settings_clicked.get()], 2),settings_clicked.get()))
    slider_var.set((config["slider"]["max_value"] + config["slider"]["min_value"]) / 2)

def update_line_slider(val):
    global current_angle, is_programmatic_update, cur_rx
    if is_programmatic_update:
        is_programmatic_update = False
        return
    if current_angle == float("-inf"):
        is_programmatic_update = True
        slider_var.set((config["slider"]["max_value"] + config["slider"]["min_value"]) /
  2)
        window.focus_set()
        messagebox.showerror(title=language_data["simulation_tab"]["error_msg_1"]["title"],
  message=language_data["simulation_tab"]["error_msg_1"]["body"])
        return

    angle = (float(val) / 100) * 180
    diff = min(angle + current_angle - 90, 180)
    diff = max(diff, 0)
    x = unit_circle_center[0] + radius * math.cos(diff * math.pi / 180)
    y = unit_circle_center[1] - radius * math.sin(diff * math.pi / 180)
    canvas.coords(line_id, unit_circle_center[0], unit_circle_center[1], x, y)
    x = round((angle / 180) * 100, 2)
    y = round(100 - x, 2)
    cur_rx = round(x / (y + epsilon) * float(resistor_entry.get()), 2)
    result_label.configure(text=f"{language_data['simulation_tab']['result_label']}: \nR1:R2
   = {x}:{y} \nRx ≈ {cur_rx / units[settings_clicked.get()]}{settings_clicked.get()}")

def get_x_y(z):
    x = 100 * z / (1 + z)
    y = 100 - x
    return x, y
```

Ovdje definiram funkciju „calculate“ koja se bavi izračunom unesenih podataka od strane korisnika i ukoliko korisnik nije unio sve podatke, unio je polovične podatke ili je unio podatke pod krivim formatom korisniku se pokazuje greška. Također, definiram traženi otpornik Rx kao globalnu varijablu i definiram maksimalnu pogrešku koja se može izbaciti kao rezultat mjerenja tj. korištenja simulacije:

```
def calculate():
    user_input = resistor_entry.get()
    if not user_input or user_input ==
     language_data["simulation_tab"]["user_resistor_entry"]:
        messagebox.showerror(title=language_data["simulation_tab"]["error_msg_2"]["title"],
     message=language_data["simulation_tab"]["error_msg_2"]["body"])
        return
    try:
        r3 = float(user_input)
    except ValueError:
        messagebox.showerror(title=language_data["simulation_tab"]["error_msg_3"]["title"],
     message=language_data["simulation_tab"]["error_msg_3"]["body"])
        return

    if abs(r3) < 1e-8:
        messagebox.showerror(title=language_data["simulation_tab"]["error_msg_4"]["title"],
     message=language_data["simulation_tab"]["error_msg_4"]["body"])
        return
```

```
    drop_val = simulation_clicked.get()
    if drop_val == language_data["simulation_tab"]["unknown_resistor_dropdown"]:
        messagebox.showerror(title=language_data["simulation_tab"]["error_msg_5"]["title"],
     message=language_data["simulation_tab"]["error_msg_5"]["body"])
        return

    global rx
    rx = resistors[drop_val]
    err = random.uniform(-max_error_percentage, max_error_percentage) / 100
    rx *= 1 + err
    res = round(rx / r3, 2)
    x, y = get_x_y(res)
    x, y = round(x, 2), round(y, 2)
    update_line_run((y / 100) * 180)

def on_entry_click(event):
    entry = event.widget
    if entry.get() == language_data["simulation_tab"]["user_resistor_entry"]:
       entry.delete(0, tk.END)
       entry.configure(fg="black")

def on_entry_focusout(event):
    entry = event.widget
    if entry.get() == "":
        entry.insert(0, language_data["simulation_tab"]["user_resistor_entry"])
        entry.configure(fg="grey")

def clear_widgets():
    resistor_entry_var.set(language_data["simulation_tab"]["user_resistor_entry"])
    resistor_entry.configure(fg="grey")

    simulation_clicked.set(language_data["simulation_tab"]["unknown_resistor_dropdown"])
    result_label.configure(text=f'{language_data["simulation_tab"]["result_label"]}: ')
    canvas.coords(line_id, unit_circle_center[0], unit_circle_center[1],
     unit_circle_center[0], unit_circle_center[1] - radius)
    slider_var.set((config["slider"]["max_value"] + config["slider"]["min_value"]) / 2)
    global current_angle, rx, cur_rx
    current_angle = float("-inf")
    rx = None
    rx = None
```

U ovom dijelu definiram funkciju „save_data“ koja se koristi da potvrdi unos korisnika i po potrebi izbaci poruku greške ako nešto ne valja:

```
def save_data():
    user_input = resistor_entry.get()
    if not user_input or user_input ==
     language_data["simulation_tab"]["user_resistor_entry"]:
        messagebox.showerror(title=language_data["simulation_tab"]["error_msg_2"]["title"],
     message=language_data["simulation_tab"]["error_msg_2"]["body"])
        return
    try:
        r3 = float(user_input)
    except ValueError:
        messagebox.showerror(title=language_data["simulation_tab"]["error_msg_3"]["title"],
     message=language_data["simulation_tab"]["error_msg_3"]["body"])
        return
    if abs(r3) < 1e-8:
        messagebox.showerror(title=language_data["simulation_tab"]["error_msg_4"]["title"],
     message=language_data["simulation_tab"]["error_msg_4"]["body"])
        return
    drop_val = simulation_clicked.get()
    if drop_val == language_data["simulation_tab"]["unknown_resistor_dropdown"]:
        messagebox.showerror(title=language_data["simulation_tab"]["error_msg_5"]["title"],
     message=language_data["simulation_tab"]["error_msg_5"]["body"])
        return
    if not cur_rx:
        messagebox.showerror(title=language_data["simulation_tab"]["error_msg_6"]["title"],
     message=language_data["simulation_tab"]["error_msg_6"]["body"])
        return

    r1 = float(slider_var.get())
```

```
            r2 = 100 - r1
            new_row_values = (drop_val, str(round(r1 / units[settings_clicked.get()], 5)),
             str(round(r2 / units[settings_clicked.get()], 5)), str(round(r3, 5)), str(round(cur_rx /
             units[settings_clicked.get()], 5)))
            idx = len(tree.get_children())
            add_row()
            tree.item(str(idx), values=new_row_values)

    layout_image = tk.PhotoImage(file=layout_image_path)
    canvas = tk.Canvas(simulation_tab, width=width, height=height)
    canvas.create_image(0, 0, anchor=tk.NW, image=layout_image)
    canvas.pack(side="top")

    radius = 57.5
    unit_circle_center = (380.0, 250.0)
    angle = 90
    x = unit_circle_center[0] + radius * math.cos(angle * math.pi / 180) # convert angle to
       radians
    y = unit_circle_center[1] - radius * math.sin(angle * math.pi / 180)
    line_id = canvas.create_line(unit_circle_center[0], unit_circle_center[1], x, y, fill="red",
       width=1.5)
```

Ovaj dio koda služi kako bi prostor za unos otpora proizvoljnog otpornika reagirao na klik miša korisnika:

```
    # User resistor entry
    resistor_entry_var = tk.StringVar(simulation_tab,
    value=language_data["simulation_tab"]["user_resistor_entry"])
    resistor_entry = tk.Entry(master=simulation_tab, textvariable=resistor_entry_var, fg="grey")
    resistor_entry.bind("<FocusIn>", on_entry_click)
    resistor_entry.bind("<FocusOut>", on_entry_focusout)
    resistor_entry.place(x=390.0, y=73.0, width=96.5, height=34.0)
```

Definiranje padajućeg izbornika za otpornik nepoznatog (traženog) otpora:

```
    # Dropdown menu for unknown resistor
    style = ttk.Style()
    style.theme_use("clam")
    style.configure("White.Horizontal.TScale", background="white")
    resistor_names = list(resistors.keys())
    simulation_clicked = tk.StringVar()
    simulation_clicked.set(language_data["simulation_tab"]["unknown_resistor_dropdown"])
    unknown_resistor_drop = ttk.OptionMenu(simulation_tab, simulation_clicked,
    language_data["simulation_tab"]["unknown_resistor_dropdown"], *resistor_names)
    unknown_resistor_drop.place(x=276.0, y=75.0, width=91.5, height=31.5)
```

Tipka za pokretanje simulacije:

```
    # Run button
    run_button = tk.Button(master=simulation_tab,
    text=language_data["simulation_tab"]["run_button"], command=calculate, bg="#92E88B")
    run_button.place(x=565.0, y=110.0, width=146.0, height=45.0)
```

Tipka za čišćenje podataka iz simulacije:

```
    # Clear button
    clear_button = tk.Button(master=simulation_tab,
    text=language_data["simulation_tab"]["clear_button"], command=clear_widgets, bg="#FB6666")
    clear_button.place(x=565.0, y=182.0, width=146.0, height=45.0)
```

Definiranje rezultat labela:

```
# Result label
result_label = tk.Label(master=simulation_tab,
text=f"{language_data['simulation_tab']['result_label']}: ", bg="#92E88B", anchor="w",
justify="left", padx=20)
result_label.place(x=565.0, y=254.0, width=200.0, height=70.0)
```

Tipka za spremanje podataka u tablicu:

```
# Save button
save_button = tk.Button(master=simulation_tab,
text=language_data["simulation_tab"]["save_button"], command=save_data, bg="#35aacb")
save_button.place(x=565.0, y=354.0, width=146.0, height=54.0)
```

Postavke kliznika koji kontrolira omjer otpornika R1 i R2:

```
# Slider
slider_var = tk.DoubleVar()
slider_var.set(50)
slider = tk.Scale(master=simulation_tab, from_=slider_info["max_value"],
to=slider_info["min_value"],  command=update_line_slider, variable=slider_var,
resolution=0.1, orient=tk.HORIZONTAL, label="", showvalue=False)
def increment_slider():
    current_val = slider_var.get()
    slider_var.set(current_val + 0.1)
    update_line_slider(current_val + 0.1)

def decrement_slider():
    current_val = slider_var.get()
    slider_var.set(current_val - 0.1)
    update_line_slider(current_val - 0.1)

ttk.Button(master=simulation_tab, text=">", command=decrement_slider).place(x=500, y=430)
ttk.Button(master=simulation_tab, text="<", command=increment_slider).place(x=222, y=430)
slider.focus_set()
slider.place(x=263, y=430, width=235)
### END OF SIMULATION TAB ###
```

U posljednjem dijelu koda definiram karticu „Postavke“ u kojoj je moguće promijeniti jezik simulacije te jedinicu otpora pri samom korištenju simulacije:

```
### START OF SETTINGS TAB ###
def allow_buttons(simulation_clicked_button, new_language):
    simulation_clicked_button.configure(state="disabled")
    for button in language_buttons:
        if button != simulation_clicked_button:
            button.configure(state="normal")

    change_language(new_language)

language_label = tk.Label(master=settings_tab,
text=f"{language_data['settings_tab']['language_label']}: ", font=("Helvetica", 13),
anchor="w")
language_label.place(x=125, y=100, width=105)

cr_language_button = ttk.Button(master=settings_tab, text="HR", command=lambda:
allow_buttons(cr_language_button, "languages/cr_CR.json"), state="disabled")
cr_language_button.place(x=240, y=95)

en_language_button = ttk.Button(master=settings_tab, text="EN", command=lambda:
allow_buttons(en_language_button, "languages/en_EN.json"))
en_language_button.place(x=290, y=95)
language_buttons = [cr_language_button, en_language_button]

unit_label = tk.Label(master=settings_tab,
text=f"{language_data['settings_tab']['resistance_scale_label']}: ", font=("Helvetica", 13),
justify="left")
```

```
unit_label.place(x=67, y=203, width=250)
unit_symbols = list(units.keys())
settings_clicked = tk.StringVar()
settings_clicked.set(unit_symbols[1])
units_drop = ttk.OptionMenu(settings_tab, settings_clicked, unit_symbols[1], *unit_symbols)
units_drop.place(x=270, y=195, width=91.5, height=31.5)
### END OF SETTINGS TAB ###
```

Ovdje se definira kartica „Informacije“ koja sadrži opće informacije o programu i način kako koristiti simulaciju na pravilan način:

```
### START OF INFORMATION TAB ###
with open(language_data["info_tab"]["info"], "r", encoding="utf-8") as f:
    info_text = f.read()
info_label = tk.Label(master=info_tab, text=info_text, anchor="nw", wraplength=730,
justify="left", padx=10)
info_label.place(x=100, y=50, height=600, width=750)
### END OF INFORMATION TAB ###

# Copyright label
copy_label = tk.Label(window, text=f"© {language_data['copyright_label']['author_rights']},
Fran Gaca @ {language_data['copyright_label']['department']}, Osijek (2023-
{datetime.now().year})", font=("Helvetica", 15))
copy_label.place(x=150, y=650)

def change_language(file):
    global language_data
    with open(file, "r", encoding="utf-8") as f:
        language_data = json.load(f)

    window.title(language_data["app_name"])
    copy_label.configure(text=f"© {language_data['copyright_label']['author_rights']}, Fran
Gaca @ {language_data['copyright_label']['department']}, Osijek (2023-
{datetime.now().year})")
    tabs.tab(0, text=language_data["simulation_tab"]["tab_name"])
    tabs.tab(1, text=language_data["settings_tab"]["tab_name"])
    tabs.tab(2, text=language_data["table_tab"]["tab_name"])
    tabs.tab(3, text=language_data["info_tab"]["tab_name"])
    add_row_button.configure(text=language_data["table_tab"]["add_row_button"])
    delete_row_button.configure(text=language_data["table_tab"]["delete_row_button"])
    import_button.configure(text=language_data["table_tab"]["import_button"])
    export_button.configure(text=language_data["table_tab"]["export_button"])
    run_button.configure(text=language_data["simulation_tab"]["run_button"])
    clear_button.configure(text=language_data["simulation_tab"]["clear_button"])
    save_button.configure(text=language_data["simulation_tab"]["save_button"])
    simulation_clicked.set(language_data["simulation_tab"]["unknown_resistor_dropdown"])
    resistor_entry_var.set(language_data["simulation_tab"]["user_resistor_entry"])
    result_label.configure(text=f'{language_data["simulation_tab"]["result_label"]}: ')
    tree.heading("#1", text=language_data["table_tab"]["unknown_resistor_heading"])
    with open(language_data["info_tab"]["info"], "r", encoding="utf-8") as f:
        info_text = f.read()
    info_label.configure(text=info_text)
    language_label.configure(text=f'{language_data["settings_tab"]["language_label"]}: ')
    unit_label.configure(text=f'{language_data["settings_tab"]["resistance_scale_label"]}:
')
```

U sklopu simulacije definirao sam i razne „eventove“ koje Python koristi kako bi pravilno funkcionirala simulacija:

```
# Bindings
add_row_button.bind("<Button-1>", set_focus)
delete_row_button.bind("<Button-1>", set_focus)
import_button.bind("<Button-1>", set_focus)
export_button.bind("<Button-1>", set_focus)
tree.bind("<Button-1>", set_focus)
# tree.bind("<FocusOut>", clear_selection)
tree.bind("<Double-1>", on_item_double_click)
frame.bind("<Button-1>", set_focus)
language_label.bind("<Button-1>", set_focus)
unit_label.bind("<Button-1>", set_focus)
```

```
copy_label.bind("<Button-1>", set_focus)
result_label.bind("<Button-1>", set_focus)
resistor_entry.bind("<Button-1>", set_focus)
units_drop.bind("<Button-1>", set_focus)
unknown_resistor_drop.bind("<Button-1>", set_focus)
run_button.bind("<Button-1>", set_focus)
save_button.bind("<Button-1>", set_focus)
clear_button.bind("<Button-1>", set_focus)
cr_language_button.bind("<Button-1>", set_focus)
en_language_button.bind("<Button-1>", set_focus)
window.bind("<Button-1>", set_focus)
```

Pokretanje simulacije:

```
# Run app
sv_ttk.set_theme("light")
window.mainloop()
```