

Univerza v Ljubljani
Fakulteta za elektrotehniko

doc. dr. Matej Možek

POLETNI TABOR INOVATIVNIH TEHNOLOGIJ

Tipala v elektronskem svetu:

Merilnik nivoja s senzorjem tlaka in Arduino modulom

Navodila za izvedbo

Ljubljana, 19. avgust 2019

Povzetek

Na delavnici se udeleženci podrobneje spoznajo z uporabo senzorskih elementov na praktičnem primeru. Delavnica je organizirana tako, da na enostavnem primeru prikaže proces izdelave senzorskega vmesnika za izbrano vrsto senzorja. Proces obsega meritve osnovnih lastnosti senzorskega elementa, načrtovanje in izdelavo vezja za obdelavo senzorskega signala in umerjanje izdelanega merilnega inštrumenta s senzorjem.

Kot primer smo si zadali obdelavo signala piezouporovnih senzorjev, med katera spada senzor tlaka. Končni cilj delavnice je izdelati merilnik nivoja vode s pomočjo senzorja tlaka in Arduino modula. Za ta namen bomo uporabili relativni senzor tlaka (Wheatstoneov mostiček, brez dodanega vezja za obdelavo signala) z merilnim področjem ca. 100 mbar, kar bi teoretično omogočalo meritev višine 1 m. Cilj naloge je celovit prikaz reševanja elektrotehniškega problema: Udeleženci se bodo najprej spoznali s teoretičnim delom o piezouporovnih senzorjih tlaka, nakar bodo senzorje tlaka izmerili ter določili osnovne parametre (občutljivost, ničelna napetost, nelinearnost). Za potrebe nadaljnjega snovanja vezij bodo izračunali, simulirali in izdelali simulator senzorja tlaka. Izdelani simulator senzorja tlaka bodo izmerili in ovrednotili ujemanje z resničnim senzorjem. Določili bodo parametre ojačevalnika senzorskega signala (ojačenje, referenčna napetost) ter dobljene parametre simulirali v povezavi s simulatorjem senzorja tlaka. Izračunani ojačevalnik bodo prilagodili merilnemu področju analogno-digitalnega pretvornika (ADC) vezja za obdelavo senzorskega signala (Arduino Nano). Udeleženci bodo programirali algoritme za digitalno obdelavo signala (segmentacija karakteristike in digitalno FIR in IIR filtriranje), umerjanje in komunikacijo z osebnim računalnikom. Končni izdelek naloge je delujoč in umerjen merilnik višine nivoja vode, ki izpisuje višino merjenega vodnega stolpca v cm na OLED zaslonu.

Predvidena časovna razdelitev aktivnosti

Tabela 1: Časovni pregled aktivnosti delavnice "Tipala v elektronskem svetu".

Dan	Dopoldne (8:30 DO 12:00 ure)	Popoldne (13:00 do 15:30)
PONEDELJEK 19.8.2019	Uvod v senzoriko in piezorezistivne senzorje tlaka (1 ura). Predstavitev senzorskih aktivnosti v LMSE (1 ura). Predstavitev inštrumentov v laboratoriju (1 ura).	Sestavljanje vmesnika za senzorski element (4 priključki 2.54). Meritve senzorskega elementa: ničelna napetost, občutljivost upornosti vej mostiča
TOREK 20.8.2019	Uvod v LTSpice Risanje vezja simulatorja senzorja Simulacije vezja - določitev ojačenja	Sestavljanje simulatorja senzorja Meritve izdelanega simulatorja Izračun parametrov instrumentacijskega ojačevalnika. Izbor uporov za izvedbo ojačenja
SREDA 21.8.2019	Ekskurzija v Zavod 404 (1.5 h) Uvod v KiCAD Risanje vezja v KiCAD	Spajkanje ojačevalnega modula.
ČETRTEK 22.8.2019	Spajkanje ojačevalnega modula. Meritve spajkanega ojačevalnika Uvod v programiranje v okolju Arduino IDE	Prilagoditev programa v Arduino IDE. Umerjanje izdelanih modulov
PETEK 23.8.2019	Umerjanje izdelanih modulov	Zaključek delavnice

Malica je vsak dan ob 10. uri. Kosilo je vsak dan ob 12. uri.

Podrobnejši opis predvidenih aktivnosti:

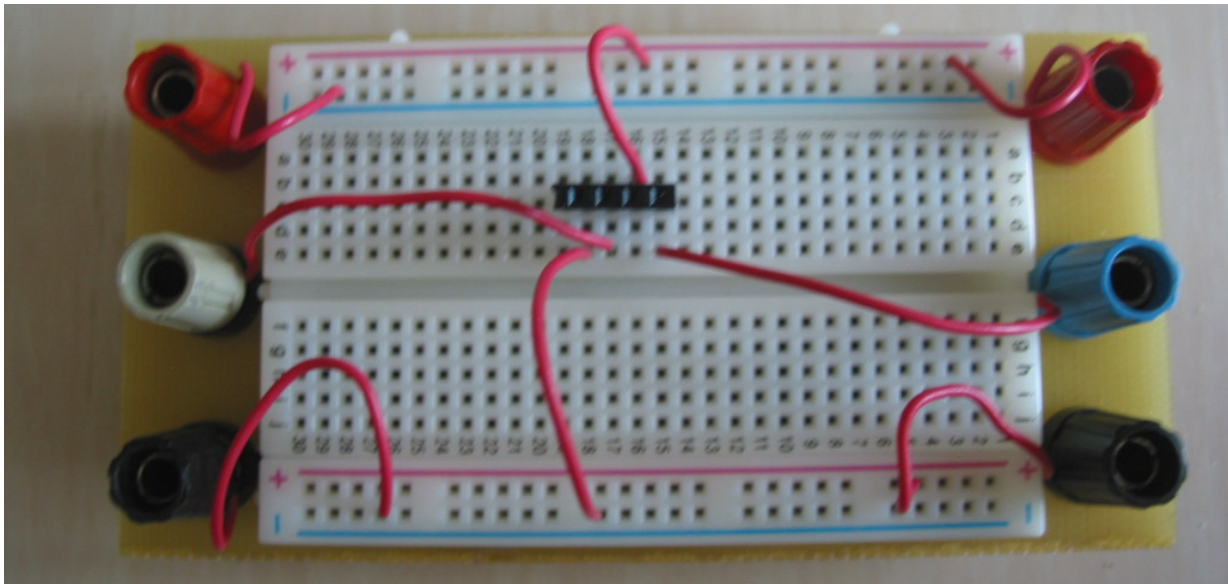
1. Uvodna predstavitev delovanja piezorezistivnih senzorjev tlaka (vrste, delovanje, splošne lastnosti senzorjev)
2. Meritve osnovnih parametrov senzorja tlaka:
 - 1.) Ničelna napetost Wheatstoneovega mostiča
 - 2.) Izhodna karakteristika senzorja - meritev višine vode (h) od odziva senzorskega elementa (V oziroma R).
 - 3.) Izračun tlačne občutljivosti senzorja ($\Delta V/\Delta h$)
 - 4.) Določitev upornosti senzorja pri maksimalnem nivoju vode - določitev merilnega področja (ang. measurement range).
3. Načrtovanje sheme vezja za obdelavo senzorskega signala (instrumentacijski ojačevalnik) - shema senzorskega vmesnika
4. Preizkus delovanja senzorja s simulacijo v programu LTSpice ob spreminjanju upornosti mostiča, iz izmerjenih vrednosti in določitev ojačenja stopnje instrumentacijskega ojačevalnika.
5. Načrtovanje ploščice tiskanega vezja senzorskega vmesnika v programu KiCAD.
6. Spajkanje in preverjanje delovanja ploščice tiskanega vezja senzorskega vmesnika
7. Programiranje Arduino modula:
 - 1.) Priključitev I²C OLED prikazovalnika na modul Arduino preko preizkusne ploščice.
 - 2.) Programiranje izpisa različnih znakov ter grafičnih simbolov.
 - 3.) Programiranje cikličnega branja surovega odčitka ADC.
 - 3.) Izpis surovega odčitka ADC na OLED prikazovalniku.
 - 4.) Preverjanje delovanja ADC s potenciometrom.
8. Umerjanje izdelanega merilnika ojačene napetosti senzorskega elementa s pomočjo merilnega traku. Rezultat je tabela vrednosti višine (cm) in izpisanih surovih odčitkov analogno-digitalnega pretvornika.
9. Programiranje preračuna filtriranega odčitka analogno-digitalnega pretvornika v višino vodnega stolpca preko linearno segmentirane prenosne funkcije.

Predstavitev uporabljenih inštrumentov v laboratoriju

Preizkusna ploščica

Preizkusna ali prototipna ploščica (ang. proto-board) je ploščica, ki omogoča enostavno povezovanje elementov z razmakom priključkov 2.54 mm oz. 100 mil (ali mnogokratnikom). Povezave na ploščici so vezane skupaj v vrsticah (vrh in dno slike 1) - te spoznamo tako, da je ob njih rdeča oz. modra črta. Tovrstne povezave običajno uporabljamo za povezovanje napajanja (rdeča puša ob strani ploščice - pozitivna sponka napajanja in črna puša - negativna sponka napajanja).

V sredinskem delu slike 1 so povezave urejene po stolpcih - posamezen stolpec je označen s številko, od 1 (skrajno desno) do 30 (skrajno levo). Posamezen stolpec ima označen priključek v vrstici s črko od a (skrajno zgoraj) do e (skrajno spodaj). Na spodnjem delu velja podobno. Zgornji in spodnji del ločuje meja (črta na preizkusni ploščici med sivo in modro pušo).



Slika 1:Preizkusna ploščica s povezanimi elementi.

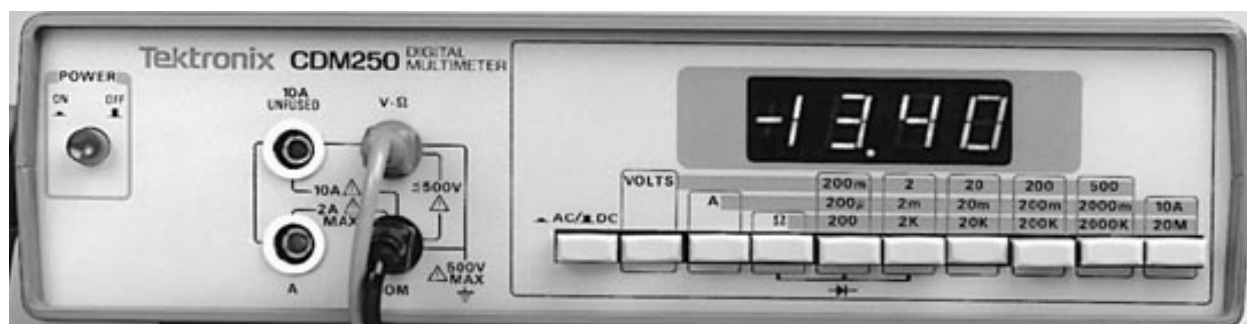
Če kratko opišemo povezave na sliki 1: Na sredinskem delu ploščice je povezana letvica s štirimi priključki (črn element sredi slike) na koordinatah c18 do c15. V zgornjem desnem kotu je povezana rdeča puša na pozitivno napajanje (+ vrsta zgoraj), medtem ko je v zgornjem levem kotu povezana rdeča puša na negativno napajanje (- vrsta zgoraj). Iz + vrste zgoraj gre povezava na tretji priključek z leve (c16) letvice. Če sledimo spodnji del slike, sta obe črni puši povezani na negativno napajanje (- vrsta spodaj), od koder vodi povezava na prvi priključek z leve (c18) letvice. Drugi priključek letvice (c19) je povezan na sivo pušo, četrti (c15) pa na modro pušo.

Digitalni multimeter CDM250

Merjenje enosmerne napetosti, toka in upornosti bomo izvajali z digitalnim multimetrom CDM250. V stolpcu modrih inštrumentov Tektronix na delovnem mestu sta dva taka inštrumenta: drugi in tretji po vrsti od zgoraj. Inštrument lahko meri izmenični ali enosmerni tok oz. napetost, kar izbirate s tipko AC/DC. Na delavnici bomo uporabljali samo enosmerne meritve, zato tipka ne sme biti pritisnjena. Merjeno veličino (napetost, tok, upornost) izbirate s tipkami VOLTS/A/ Ω (slika 2), pri čemer je naenkrat pritisnjena samo ena tipka. Merilno območje izbirate s tipkami desno od tipke Ω (spet pritisnjena samo ena tipka naenkrat). Na čelni plošči inštrumenta (slika 2) so označene veličine, ob njih pa nad ustrezno tipko merilno področje. Prikazani položaj pritisnjenih tipk na sliki 2 nastavi inštrument kot voltmeter na področju meritve do 20 V. Inštrument trenutno na sliki 2 meri napetost -13.40 V. Za meritve napetosti in upornosti sta uporabljena merilna vhoda V/ Ω in COM. Če bi želeli meriti tok, bi morali uporabiti merilni vhod A.

Tabela 2: Pregled merilnih območij in vhodov multimetra CDM250.

Veličina	Merilno področje						Vhod	
							+	-
Napetost	200 mV	2 V	20 V	200 V	500 V	/	V/ Ω	COM
Tok	200 μ A	2 mA	20 mA	200 mA	2000 mA (2 A)	10 A*	A	COM
Upornost	200 Ω	200 Ω (2 k Ω)	20000 Ω (20 k Ω)	200 k Ω	2000 k Ω (2 M Ω)	20 M Ω	V/ Ω	COM



Slika 2: Čelna plošča digitalnega multimetra Tektronix CDM250.

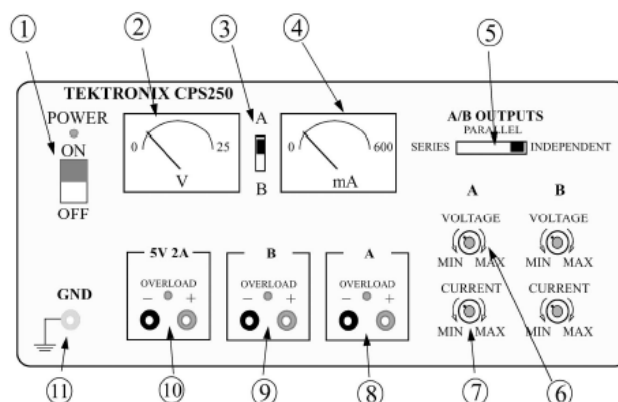
Vir enosmerne napetosti CPS250

CPS250 je 3 kanalni vir enosmerne napetosti. Inštrument se nahaja na dnu stolpca modrih inštrumentov Tektronix na delovnem mestu. Na kanalih A in B lahko nastavimo napetost med 0 in 25 V ter tok med 0 do 0.5 A. Tretji kanal je fiksni 5V/2A in ga ne bomo uporabljali.

Kanala A in B lahko delujeta neodvisno (ang. independent). Kanala lahko vezemo zaporedno (ang. series) za povečanje izhodne napetosti ali vzporedno (ang. parallel) za povečanje toka. Način delovanja izbiramo s preklopnikom 5 na sliki 3. Na delavnici bomo uporabljali samo kanal A v neodvisnem načinu, zato preklopnik 5 postavite v položaj "independent". Če želite nastavljati napetost, vrtite potenciometer 6, za nastavljanje toka pa potenciometer 7 na sliki 3.

Inštrument ima dva kanala, na vsakem lahko nastavljamo napetost in tok, torej bi morali za prikaz imeti 4 merilnike - inštrument ima namesto tega dva merilnika, pri čemer s preklopnikom 3 izbirate A/B kanal, ki ga inštrument prikazuje na merilnikih 2 in 4. Postavite ga v položaj A.

Izhodna napetost kanala A se pojavi na sponkah 8, pri čemer je črna sponka masa, rdeča pa pozitivna sponka izhodne napetosti. Zelena sponka 11 je šasija (ozemljitev) in se na teh meritvah ne uporablja. Če tokovno omejitev (gumb 7) zavrtite do konca v položaj MIN zasveti dioda preobremenitve (ang. overload). Dokler ta dioda sveti, vir na izhodu ne daje več nastavljene napetosti, pač pa je tok omejuje na nastavljeno vrednost. Povedano drugače - breme, ki je na izhodu je sposobno porabljati večji tok, vendar ga vir zaradi omejitve toka ne more zagotoviti. Ko dioda sveti, se vir obnaša kot tokovni vir. Tokovno omejitev uporabljamo *predvsem kot zaščito pred kratkim stiki*, saj bi brez zaščite iz vira stekel največji tok (0.5 A), kar lahko uniči priključeno vezje. V tej delavnici tokovi ne presegajo 60 mA, tako da tokovno omejitev nastavite na 60 mA tako, da zavrtite potenciometer 7 na minimum, izhod A kratko staknete (žica med rdečo in črno kontaktov 8), nakar zasveti dioda preobremenitve, nato pa počasi vrtite gumb 7 in na desnem prikazovalniku (4) nastavite približno vrednost 60 mA.



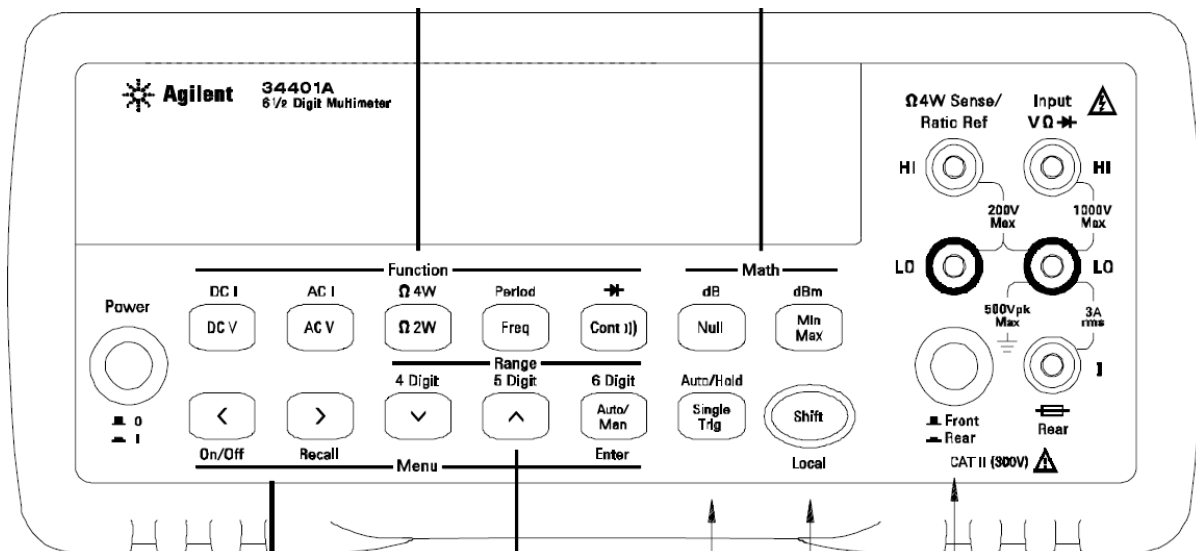
Slika 3: Čelna plošča vira enosmerne napetosti Tektronix CPS250.

Digitalni multimeter Agilent 33401A

Natančneje merjenje enosmerne napetosti, toka in upornosti bomo izvajali z digitalnim multimetrom Agilent 33401A. Na delovnem mestu se nahaja desno ob stolpcu modrih inštrumentov Tektronix. Na delavnici bomo uporabljali samo enosmerne meritve (DCV za napetost, DCI za tok, $\Omega 2W$ za upornost). Vrsto meritve nastavite s pritiskom na tipko DCV, DCI oz. $\Omega 2W$ na sliki 4. Merilno območje se pri tem inštrumentu nastavlja samodejno. Za meritve napetosti in upornosti sta uporabljena merilna vhoda V/ Ω (desni HI vhod) in LO vhod (ni važno kateri). Če bi želeli meriti tok, bi morali uporabiti merilni vhod I (desno spodaj, slika 4). Inštrument ima vhode tudi na zadnji strani (ang. rear), zato preklopnik FRONT/REAR ne sme biti pritisnjen.

Tabela 3: Pregled funkcij in vhodov multimetra Agilent 33401A

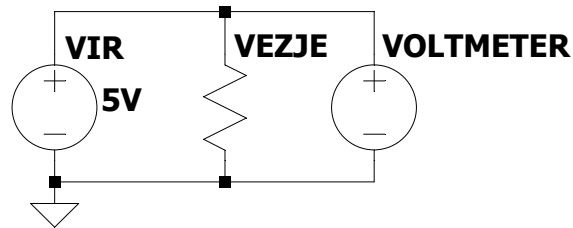
Veličina	Tipke za izbiro funkcije	Vhod	
		+ (HI)	- (LO)
Napetost	DCV	V/ Ω	LO
Tok	SHIFT+DCV	I	LO
Upornost	$\Omega 2W$	V/ Ω	LO



Slika 4: Čelna plošča digitalnega multimetra Agilent 33401A.

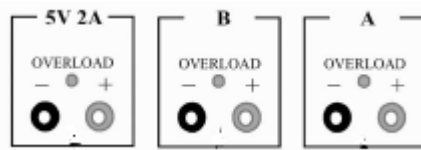
Nastavljanje in merjenje enosmerne napetosti

Voltmeter zaznava napetostno razliko, ki se nahaja med pozitivno (V/ Ω) in negativno priključno sponko (COM). **VOLTMETER** praviloma priključujemo **VZPOREDNO** (pozitivni sponki skupaj, negativni skupaj) z merjenim vezjem. Takrat je napetost na merjenem vezju enaka napetosti na voltmetru (slika 5).



Slika 5: Vzoredna priključitev treh elementov v vezju.

Preden enosmerne vira CPS250 ne nastavite, ne priključujte vezij na njegov izhod! Ko vir CPS250 vključite, se prižge dioda POWER (slika 3, nad stikalom 1). Izhodna napetost na viru se pojavi **takoj** med sponkama + in COM (COMMon - skupni).



Slika 6 Razlaga pomena priključnih sponk vira CPS250.

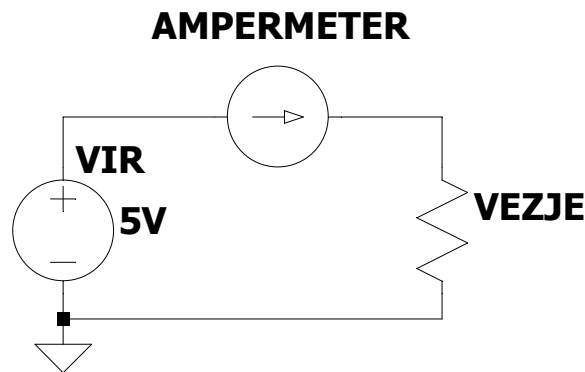
Najprej vključite digitalni multimeter CDM250 (gumb levo zgoraj - Power). Preverite, če je v načinu merjenja enosmerne napetosti (tipka VOLTS je pritisnjena in tipka AC/DC ni pritisnjena) in na če je izbrano merilno območje 20 V (pritisnjeni so isti gumbi kot na sliki 3).

Nastavite vir CPS250: Uporabljamo ga v neodvisnem načinu (A in B vira nista odvisna), zato postavite preklopnik 5 (slika 3) v položaj neodvisno (ang. independent). Postavite preklopnik 3, slika 3 merilnika na CPS250 na položaj A - takrat merilnika kažeta napetost in tok vira A.

Povežite multimeter in vir: Vhodni sponki za meritve napetosti na multimetru CDM250 sta V/ Ω (+) in COM (-) (slika 2). Izhodni sponki vira CPS250 sta + in - na kanalu A. Gumb 6 (slika 3) za nastavljanje napetosti na kanalu A vira CPS250 vrtite toliko časa, dokler se na voltmetru ne pojavi vrednost 5 (nastavljena vrednost naj bo točna na eno decimalko, se pravi 5.0x).

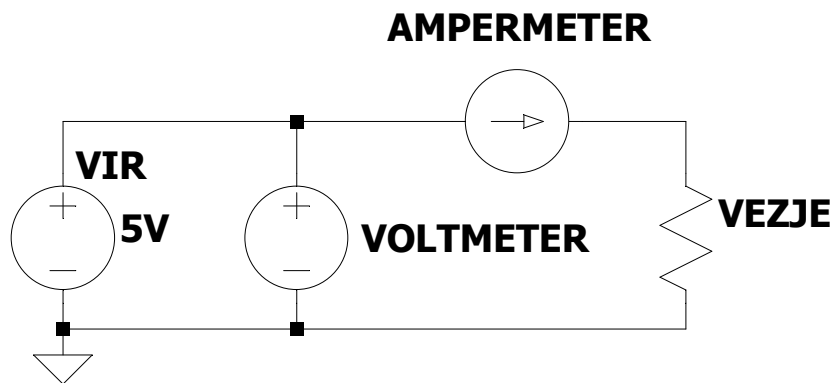
Nastavljanje in merjenje enosmernega toka

Ampermeter CDM250 zaznava tok, ki teče med priključnima sponkama A in COM. **AMPERMETER** priključujemo **ZAPOREDNO** (pozitivni izhod gre na negativni vhod ampermetra, nakar gre negativni izhod ampermetra na izhod vezja) z merjenim vezjem. Takrat je tok skozi merjeno vezje enak toku skozi ampermeter (slika 6).



Slika 7: Zaporedna priključitev treh elementov v vezju.

Vključite **drugi** digitalni multimeter CDM250 v stolpcu inštrumentov. Preverite, če je v načinu merjenja enosmernega toka (tipka A je pritisnjena in tipka AC/DC ni pritisnjena) in na če je izbrano merilno območje 200 mA (tretji gumb z desne). Na viru CPS250 zaprite tokovno omejitev na minimum: Gumb 7 (slika 3) za nastavljanje toka na kanalu A vira CPS250 zavrtite v položaj MIN (vrtite v levo).



Slika 8: Vezava voltmetra, ampermetra in enosmernega vira.

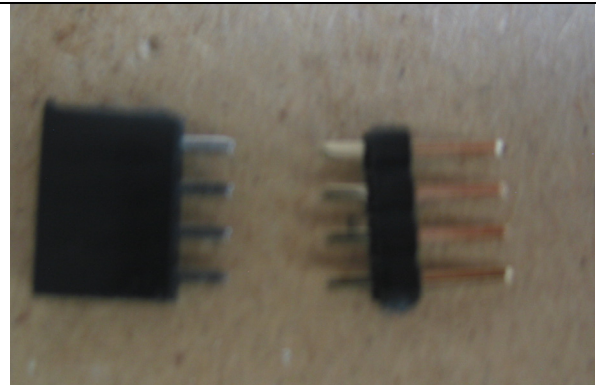
Izdelava senzorskega vmesnika

Senzorski vmesnik bomo sestavili zato, da lahko senzorski element brez težav zatakujemo v preizkusno ploščico. Senzorski element ima namreč precej tanke priključke in jih lahko zelo hitro polomimo, če bi ga neposredno vtaknili v preizkusno ploščo. Vmesnik bomo naredili iz visoke moške in ženske letvice z razmakom priključkov (ang. raster) 2.54 mm.

Najprej si pripravimo material in potrebno orodje. Potrebovali bomo moško in žensko visoko letvico, koničaste klešče (slika 1, spodaj), ščipalke (desno).



Slika 9: Potrebno orodje za izdelavo senzorskega vmesnika.



Slika 10: Odrezani letvici pred spajkanjem.



Slika 11: Odrezani letvici, pripravljeni za spajkanje.

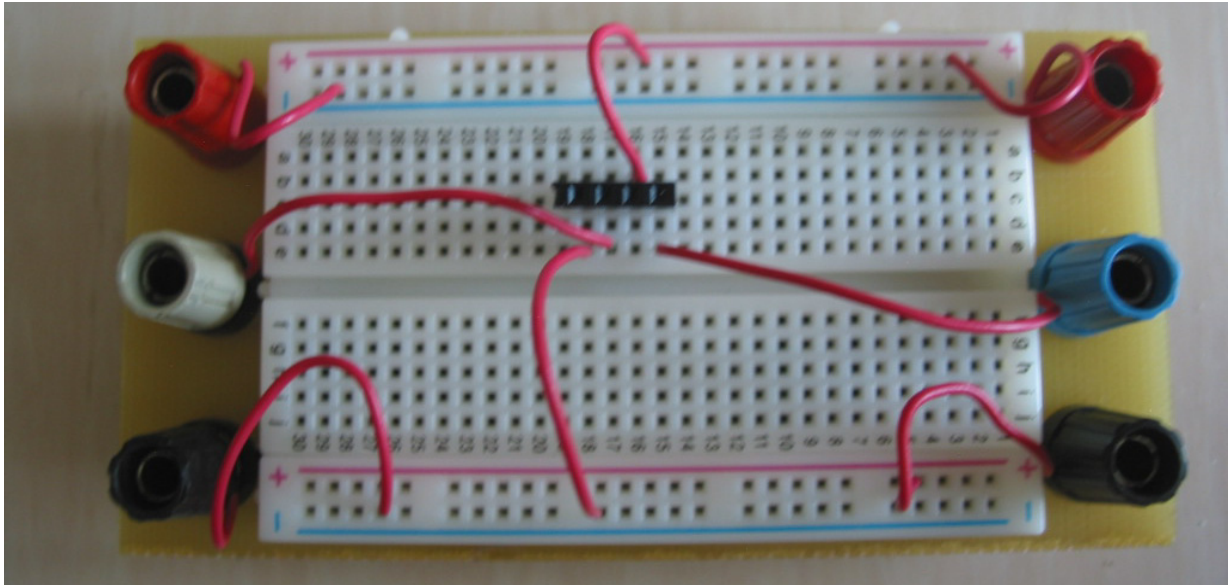


Slika 12: Spajkalna postaja Ersas i-Con 2

S koničastimi kleščami izvlečemo peti priključek (ang. pin) iz ženske letvice - pred njim ostanejo štirje (slika 9). Moško letvico preprosto odščipnemo tako, da ostanejo skupaj 4 priključki. Na leseno klado (slika 10) položimo letvici tako, da se priključki prekrivajo. Moško letvico dodatno pritrdimo tako, da jo vtaknemo v košček ženske letvice (slika 11, skrajno desno). Ženski letvici prilepimo z lepilnim trakom na klado. Spajkalno postajo nastavimo na 350 °C in priključke spajkamo. Začnemo na enem koncu (slika 11, priključek z vrha) in nadaljujemo na nasprotnem koncu (slika 11, priključek spodaj) proti sredini. Ko smo spojili vse priključke, nastalo letvico obrnemo in preverimo, če se je spajkalni cini prelil tudi na drugo stran priključkov.

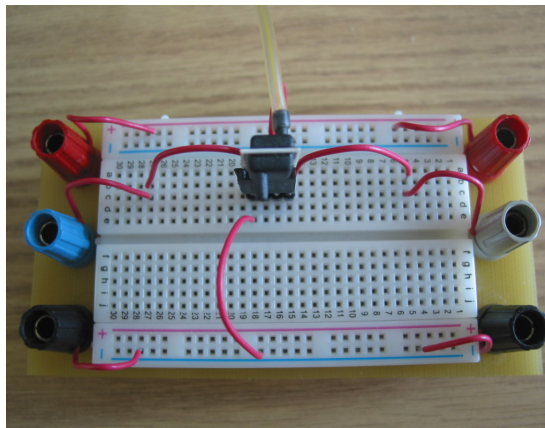
Prva priključitev senzorja

V preizkusno ploščico najprej zataknete izdelani senzorski vmesnik, tako da bo prvi priključek (skrajno levi) na položaju c18. Na skrajno levi priključek senzorskega vmesnika z dodatno žico povežite levo spodnjo pušo. Na drugi priključek senzorskega vmesnika povežite sivo pušo (leva srednja), na tretjega pozitivno napajanje (zgornja leva rdeča puša) in na četrti priključek senzorskega vmesnika modro pušo.



Slika 13: Priključitev napajanja in izhodnih priključkov senzorja na preizkusni ploščici.

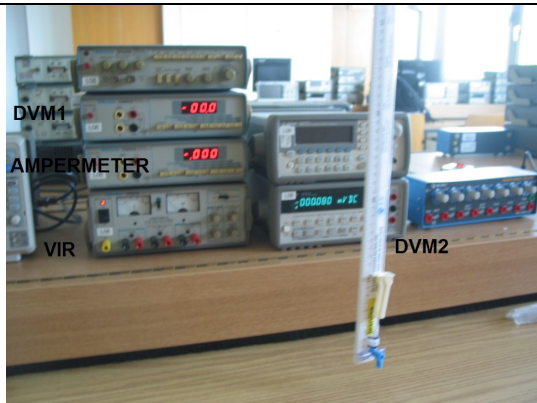
Nato v vmesnik previdno vtaknete senzor tlaka, pri čemer pazite na orientacijo tlačnih priključkov: Če opazujemo sliko 14, je spodnji tlačni priključek (črna cevka, ki štrli iz senzorja) zamaknjen levo glede na zgornjega. Če tlačna priključka obrnete, senzor ne bo imel pravilno priključenega napajanja in z njim ne bomo mogli meriti. Napačno električno priključen senzor ob priključitvi kaže ca. 700 mV, pravilno priključen pa napetost med 30 in 50 mV.



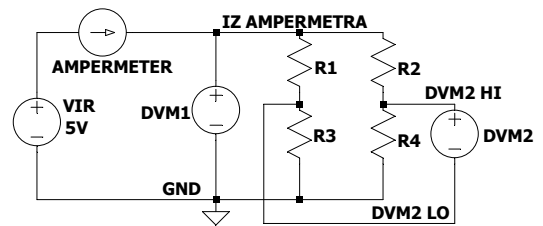
Slika 14: Tlačna in električna priključitev senzorja.

Meritev senzorske karakteristike

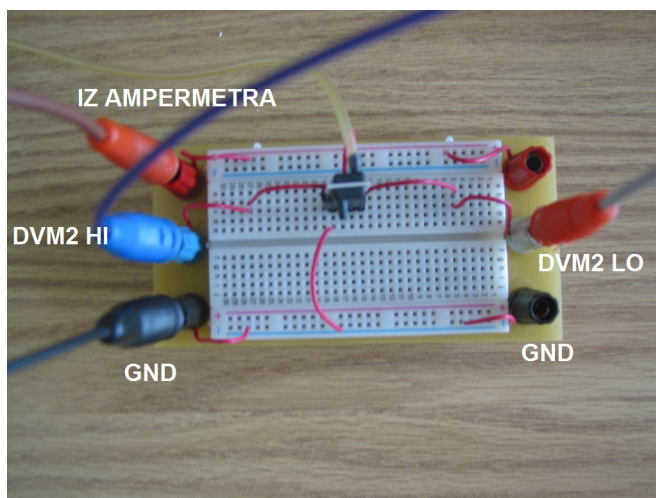
Preden priključite preizkusno ploščico s senzorjem, nastavite napajalno napetost 5 V in omejite tok vira na 60 mA po vezju na sliki 8. Potem povežite senzor po shemi na sliki 16, pri čemer si za dejansko postavitev senzorja na preizkusni ploščici pomagajte s sliko 17.



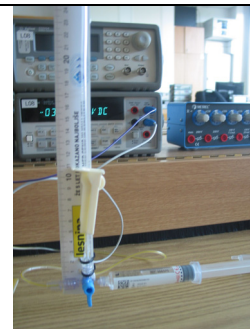
Slika 15: Uporabljeni inštrumenti.



Slika 16: Meritev karakteristike senzorja



Slika 17: Vezava senzorja.



Slika 18: Nastavljanje nivoja vode z brizgo.

Najprej izmerite ničelno napetost senzorja - zapišete si izmerjeno napetost na DVM2, če tlačni dovod senzorja (plastična cev z Luer-Lock zaključkom) ni priključen nikamor (ni tlačne razlike med priključkoma senzorja).

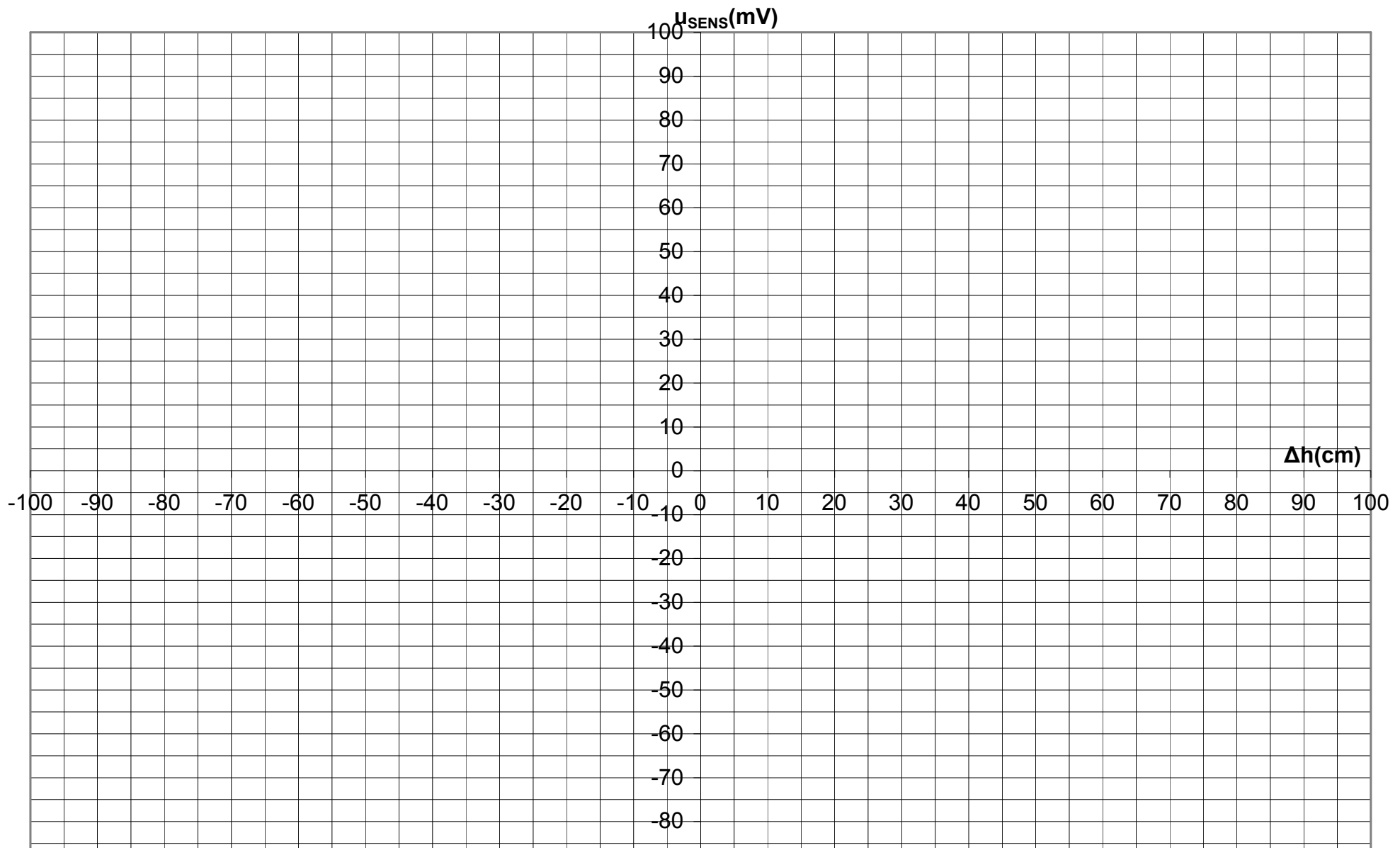
Nato priključite tlačni dovod na levo stran ventila pri vodnem stolpcu kot kaže slika 18. Modri ventil naj bo v takem položaju, kot je na sliki 18. Nastavite višino vodnega stolpca - narahlo potisnete vodo z brizgo v vodni stolpec in spremljate višino nivoja vode. Ko vodo potiskate po stolpcu navzgor pazite, da v vodnem stolpcu ni zračnih mehurčkov. Vodni stolpec se mora jasno razširjati samo po

vertikalnem vodu - ne proti senzorju. Če se širi tudi proti senzorju, pomeni da tlačna povezava pušča. Odčitke DVM2 zapisujte v spodnjo tabelo, pri čemer v prvi stolpec tabele 4 pišete odčitke, ko je uporabljen en tlačni priključek, nato meritev ponovite še za drugi tlačni priključek in pišete v drugi stolpec tabele 4. Vrednosti do 20 cm bomo izmerili bolj na gosto.

Tabela 4: Karakteristika senzorja tlaka HPSA1000

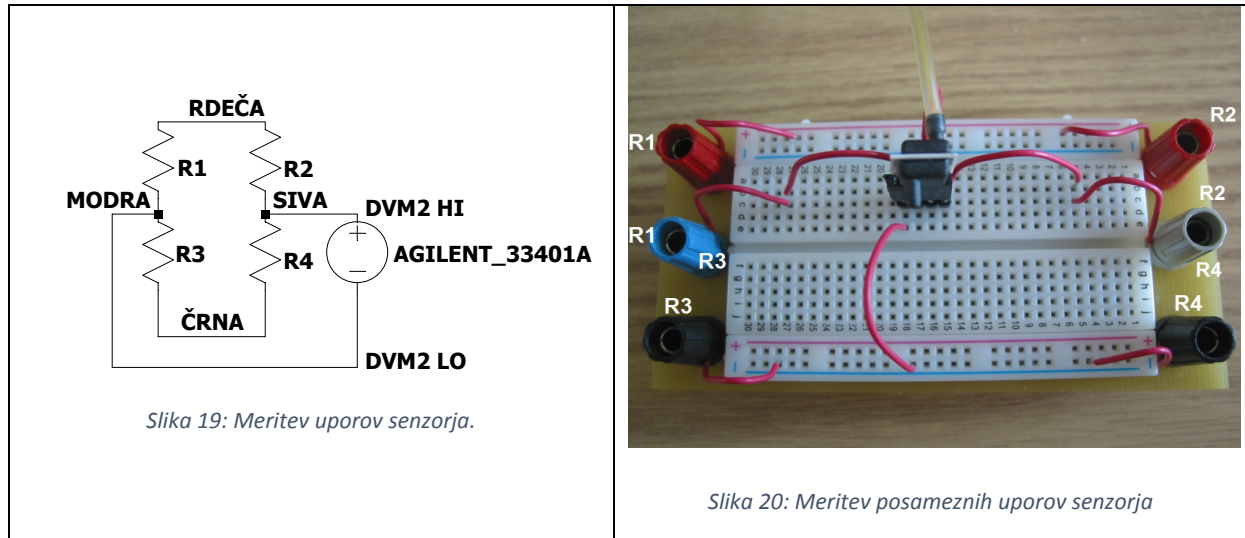
$\Delta h(\text{cm})$	Tlačni priključek 1	Tlačni priključek 2
	$\Delta u_{\text{SENSOR}}(\text{mV})$	$\Delta u_{\text{SENSOR}}(\text{mV})$
0*		
5		
10		
15		
20		
30		
40		
50		
60		
70		
80		
90		
100		

* ničelna napetost



Meritev odziva posameznih uporov senzorja

Za meritev posameznih uporov Wheatstoneovega mostiča senzorja uporabite samo Agilent 33401. Izberete meritev upornosti (gumb Ω W) in merilni žici, ki ju priključite na preizkusno ploščico po sliki 19. Oznake uporov mostiča in vezavo razberete iz slike 20. Na koncu pomerite še vhodno (med rdečo in črno) in izhodno (med modro in sivo) upornost mostiča. Senzorske upore merimo zato, da bomo v naslednjem razdelku lahko sestavili simulator senzorja.



Slika 19: Meritev uporov senzorja.

Slika 20: Meritev posameznih uporov senzorja

$\Delta h(\text{cm})$	Uporabljen tlačni priključek 1				Uporabljen tlačni priključek 2			
	$R_1(\Omega)$	$R_2(\Omega)$	$R_3(\Omega)$	$R_4(\Omega)$	$R_1(\Omega)$	$R_2(\Omega)$	$R_3(\Omega)$	$R_4(\Omega)$
0								
20								
40								
60								
80								
100								

Na koncu izmerite še izhodno in vhodno upornost, tako da izmerite upornost med rdečo in črno pušo (vhodna upornost) in modro in sivo (izhodna upornost).

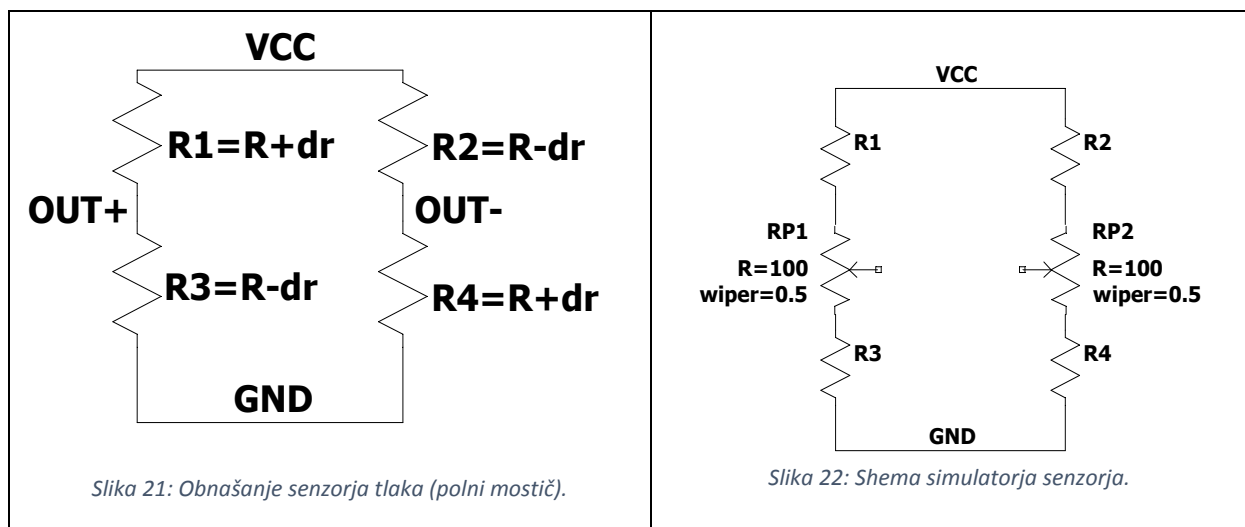
$$R_{\text{vhodna}} = \text{_____} \Omega$$

$$R_{\text{izhodna}} = \text{_____} \Omega$$

Simulator senzorja

Izračun komponent simulatorja senzorja

Iz izmerjenih vrednosti uporov $R_1 \dots R_4$ vidimo, da se potencial točke $OUT+$ povečuje, medtem ko se potencial točke $OUT-$ zmanjšuje proti GND . Namen simulatorja senzorja je, da nadomesti delovanje senzorja v tej meri, da ima podobno izhodno upornost in da se mu izhod lahko nastavlja v podobnem območju kot resničnemu senzorju. Simulatorje senzorjev izdelujemo zato, ker poenostavljajo stvari, saj izpostavijo samo električni vidik določene naprave (npr. senzorja). Poleg tega je simulator senzorja mnogo cenejši in robustnejši od samega senzorja.



Za zgornji mostič na sliki 21 lahko zapišemo izhodno napetost kot razliko leve in desne veje ($OUT+$ in $OUT-$).

$$U_{OUT} = U_{OUT+} - U_{OUT-} = V_{CC} \left(\frac{R_3}{R_1 + R_3} - \frac{R_4}{R_2 + R_4} \right)$$

$$U_{OUT} = V_{CC} \left(\frac{R - \Delta R}{(R + \Delta R) + (R - \Delta R)} - \frac{R + \Delta R}{(R + \Delta R) + (R - \Delta R)} \right) = V_{CC} \left(\frac{\Delta R}{R} \right)$$

Dobljena enačba določa razmerje spremembe upornosti (ΔR) in upornosti neobremenjenega senzorja (R). Napajalna napetost senzorja je $V_{CC} = 5V$. Pričakujemo izhodno napetost velikostnega razreda 100 mV. Uporabljeni senzor je *relativen*, torej meri pozitivno ali negativno *razliko tlaka* v tlačnih vodih in ustrezne napetosti. Zato v rezultatih meritev ne govorimo o absolutni višini (h), ampak o višinski razliki (Δh). Tipično področje izhodnih napetosti je od $U_{OUT} = 50 \text{ mV}$ do $U_{OUT} = -50 \text{ mV}$, to pa upoštevamo tako, da pišemo:

$$\frac{U_{OUT}}{V_{CC}} = \frac{100 \text{ mV}}{5V} = \frac{\Delta R}{R} = 0.02$$

Poleg tega lahko za mostič zapišemo še enačbo vhodne oz. izhodne upornosti. Za vhodno (ali izhodno) upornost mostiča zapišemo:

$$R_{vhodna} = (R_1 + R_3) \parallel (R_2 + R_4) = 2R \parallel 2R = \frac{2R}{2} = R$$

Iz zgornje enačbe sledi, da je R kar upornost veje mostiča pri senzorju brez tlačne razlike (ničelna upornost). Upornost veje R pri senzorju MPSA1000 je tipično 3.8 kΩ. Če vstavimo vrednosti v zgornje enačbe dobimo:

$$\Delta R = 0.02 \cdot R = 0.02 \cdot 3800\Omega = 76\Omega$$

Za izhodno napetost 100 mV na izhodu, bi morali na sredino v vsake veje mostiča vstaviti potenciometer RP1= RP2 = 76 Ω (glej sliko 22).

Izbira komponent simulatorja senzorja

Potenciometri ne obstajajo v poljubnih vrednostih, ampak so v področju do 100 Ω lahko izbiramo med 47Ω in 100 Ω. Če izberemo 100 Ω potenciometer, se izhodna napetost mostiča poveča na 131 mV (izhodna napetost med -65. mV ... 65.5 mV), če bi vzeli 47 Ω bi dobili 61 mV, kar je premalo:

$$U_{OUT} = \frac{100\Omega}{3.8k\Omega} \cdot 5V = 131mV$$

$$U_{OUT} = \frac{47\Omega}{3.8k\Omega} \cdot 5V = 61mV$$

Dobljeno ničelno upornost R bi sicer lahko izbrali kot 1% upor z vrednostjo 3800Ω, vendar so ti dragi (1.2€/kos), zato izbiramo med vrednostmi iz E24 lestvice (5% toleranca), kjer je cena mnogo bolj ugodna (0.01 €/kos). Izbiramo med vrednostma 3.6 kΩ in 3.9 kΩ.

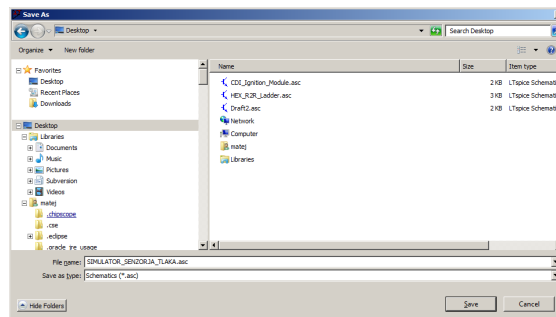
Simulacija simulatorja senzorja :)

Preden se lotimo sestavljanja vezja, moramo delovanje simulatorja preveriti v simulatorju električnih vezij LTSpiceXVII. Poženemo program na Namizju (ang. Desktop).



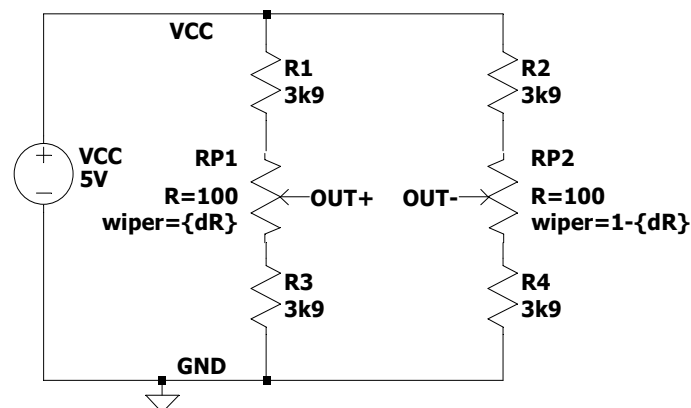
Slika 23: Orodna vrstica v LTSpice XVII.

Za risanje novega vezja izberemo File→New Schematic oz. pritisnemo CTRL+N. Odpre se novo prazno, sivo okno z imenom "Untitled". Najprej vsebino shranimo (v meniju izberemo File→Save As). Odpre se pogovorno okno, vanj vpišemo ime datoteke in izberemo lokacijo Namizje (ang. Desktop). Vnesemo ime datoteke "SIMULATOR_SENZORJA_TLAKA" in potrdimo (Save).



Slika 24: Shranjevanje datoteke "SIMULATOR_SENZORJA_TLAKA".

Narisali bomo spodnje vezje:



.tran 0 2m 1m

.save V(OUT+, OUT-)

.step param dR 0 1.0 0.2










.measure Vsens avg V(OUT+, OUT-)

.option plotwinsize=0 numdgt=15

Slika 25: Vezje simulatorja mostiča.


Vežje začnemo risati tako, da razporedimo osnovne upore $R_1 \dots R_4$ s slike 25 in jih povežemo. Za to uporabljamo orodja v orodni vrstici. Za izbiro orodja kliknemo na ustrezno ikono orodne vrstice. S tem ostaja orodje aktivno, dokler ne izberemo drugega orodja. Z miškinim kolescem shemo približujemo in oddaljujemo (ang. zoom in/out). S klikom na besedilo imena upora (npr. R1) lahko preimenujemo komponento, s klikom na polje vrednosti (novi upori imajo vrednost R) vpišemo upornost - lahko pišemo v Ohmih, za večje vrednosti pa uporabljamo okrajšave (3900 Ω kratko pišemo 3k9 ali 3.9k).

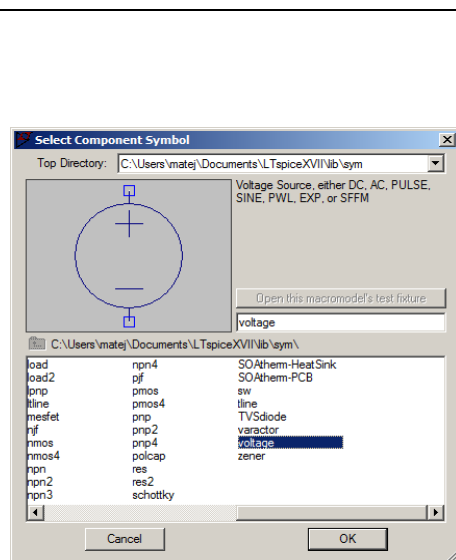
Tabela 5: Orodna vrstica LTSpiceXVII

Orodje	Funkcija orodja	Komentar
	Risanje povezav	Postavimo se na nek upor, kliknemo na priključek in vlečemo povezavo do naslednjega upora. Če želimo povezavo fiksirati v neki točki, ponovno kliknemo.
	Postavljanje referenčne točke (masa sheme)	<i>Simulacija brez referenčne točke ne deluje!.</i>
	Poimenovanje vozlišča	Z imenovanjem vozlišča lahko v poznejših analizah enostavno dostopamo do vrednosti spremenljivke s tem imenom. Če damo vozlišču ime OUT+ do njega dostopamo do napetosti točke kot V(OUT+).
	Postavljanje upora	Če se ob kliku na upor postavimo na delovno površino, se pojavi upor, ki ga lahko prosto premikamo, dokler ponovno ne kliknemo. Komponento lahko med postavljanjem vrtimo (CTRL+R) ali zrcalimo (CTRL+E).
	Izbira komponente	
	Postavljanje besedila (komentarja)	
	Postavljanje SPICE ukaza	
	Vlečenje (ang. drag) izbranega	
	Premikanje (ang. move) izbranega	

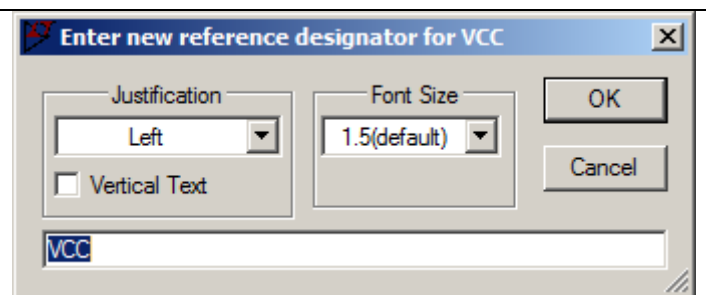
Postavljanje enosmerne napetostnega vira.

Enosmerni napetostni vir se nahaja med komponentami - ni ga med bližnjicami v orodni vrstici.

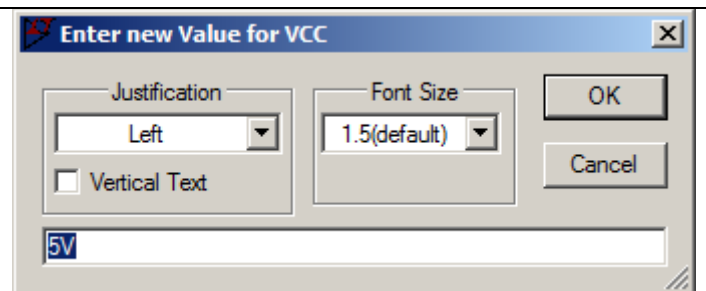
Izberemo ustrezno orodje . Odpre se pogovorno okno na sliki 26. Spodnji drsnik okna pomaknemo skrajno desno in s klikom izberemo komponento "voltage". Med izbiranjem se v zgornjem desnem delu okna izriše slika trenutno izbranega elementa. Ko smo izbrali, kliknemo - okno se zapre in vir lahko prosto premikamo po shemi. Postavimo ga na ustrezno mesto in ponovno kliknemo nanj. Postavljeni vir ima privzete parametre V1 in V. Parameter V1 je ime vira - če desno kliknete na besedilo "V1", se odpre pogovorno okno na sliki 27. Vpišete vrednost VCC. Podobno desno kliknete na besedilo "V", se pojavi pogovorno okno na sliki 28. Vpišete 5V (brez presledkov).



Slika 26: Izbira napetostnega vira v LTSpiceXVII.




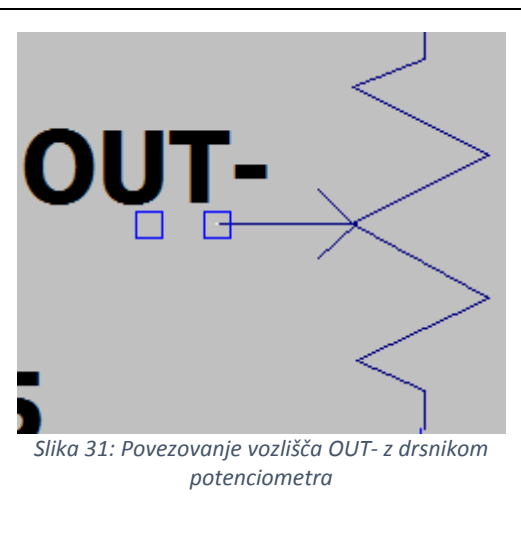
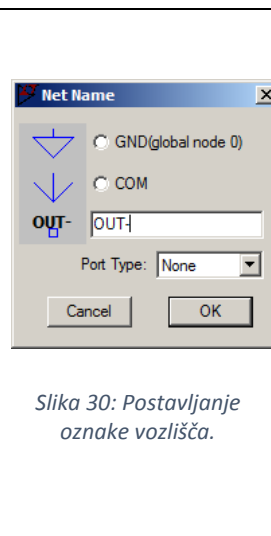
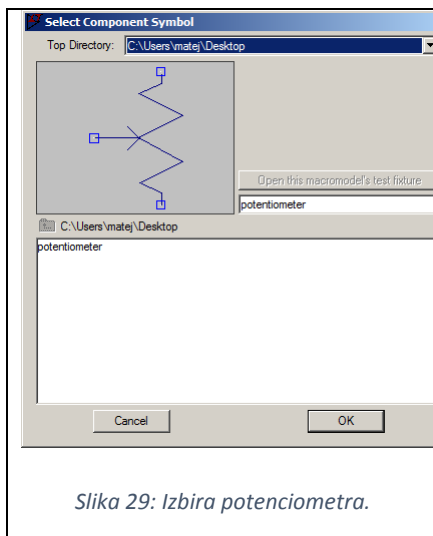
Slika 27: Nastavljanje oznake (ang. reference designator) vira.




Slika 28: Nastavljanje vrednosti vira.

Postavljanje potenciometra


Potenciometer ni generična SPICE komponenta, ampak je uporabniško ustvarjena. Preden nadaljujete, kopirajte datoteki "potentiometer.asy" in "potentiometer.lib" na Namizje (ang. Desktop) računalnika. Izberete orodje za postavljanje komponent . Podobno kot prej, se odpre pogovorno okno na sliki 26. Na vrhu okna je izbiralnik "Top directory", katerega privzeta vrednost je "C:\Users\Vaje\Documents\LTspiceXVII\Lib\Sym". Namesto tega izberite svoje Namizje "C:\Users\Vaje\Desktop". Ko to storite, se med izbirami na dnu pojavi samo ena komponenta "potentiometer". Izberite jo (kliknete), in potrdite z Ok. Na shemo postavite dva potenciometra in ju preimenujete iz U1 in U2 v RP1 in RP2. Vrednosti in imena komponente lahko premikate z orodjem za vlečenje (ang. drag) ali premikanje (ang. move). Drugi potenciometer zrcalite s CTRL+E. Obema potenciometroma prestavite vrednost upornosti iz 1k na 100 in potenciometru RP1 polje "wiper" (slov. drsnik) nastavite na {dR}, potenciometru RP2 pa polje "wiper" nastavite na 1-{dR}. S tem smo uvedli parameter {dR}, s katerim simulatorju povemo, koliko je zavrten določen potenciometer. Vrednost parametra {dR} bomo pozneje spreminjali od 0 do 1, tako da smo RP1 nastavili na povečevanje upornosti ({dR}), RP2 pa na hkratno zmanjševanje upornosti (1-{dR}).




Postavljanje oznak (ang. label)

Izberete orodje za oznake  in pojavi se pogovorno okno na sliki 30, v katerega vpišete ime vozlišča. Ustvarili bomo dve vozlišči in ju poimenovali OUT+ in OUT-. Po shemi na sliki 25 bomo OUT+ postavili na drsnik (srednji priključek) potenciometra RP1, OUT- pa na drsnik RP2. Ko potrdite izbiro v oknu na sliki 30, lahko simbol OUT+ prosto premikate po shemi. Pomembno je, da se moder kvadratac pokrije z modrim kvadratom srednjega odcepa na potenciometru (kvadratka na sliki 31). Šele takrat je vozlišče poimenovano - do takrat ni vezano nikamor (kot je prikazano na sliki 31)

Izbira vrste analize simulatorja

Izberete orodje za SPICE ukaz  in v pogovorno okno vpišete ukaz ".tran 0 2m 1m". S tem ste simulator nastavili na časovno analizo (ang. transient analysis), ki bo trajala 2 ms, pri čemer si bo simulator šele po 1 ms začel zapisovati rezultate.

Nastavitev obdelave podatkov po končani analizi

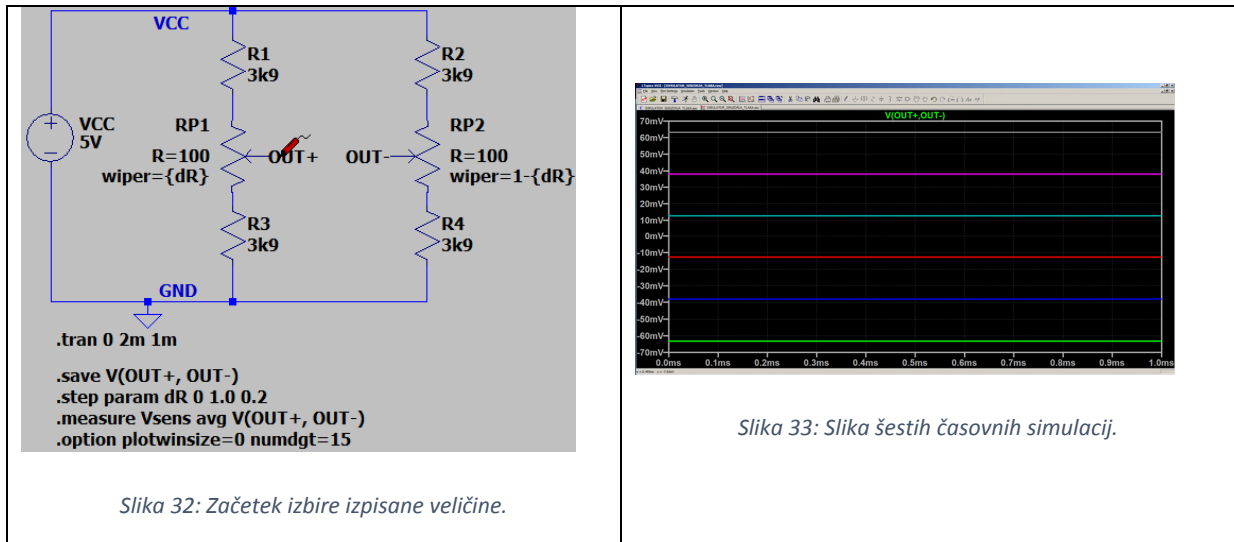
Izberete orodje za SPICE ukaz  in v pogovorno okno vpišete ukaz:

```
.save V(OUT+, OUT-)  
  
.step param dR 0 1.0 0.2  
  
.measure Vsens avg V(OUT+, OUT-)  
  
.option plotwinsize=0 numdgt=15
```

S tem ste simulator nastavili, da bo shranjeval (ang. save) napetostno razliko med vozliščema OUT+ in OUT-. Pri analizah bo izvajal koračno spremembo parametra (ang step parameter analysis) dR v območju od 0 do 1, s korakom 0.2. Izvedel bo torej šest časovnih analiz (pri dR = 0, 0.2, 0.4, 0.6, 0.8, 1.0). Pri vsaki časovni analizi bo izmeril povprečno (ang. average, funkcija avg) vrednost razlike napetosti med OUT+ in OUT- ter rezultat shranil v spremenljivko Vsens. Zadnji ukaz je za nastavljanje skale izhodnega grafa in natančnost izračuna.

Poganjanje simulacije in izpis rezultatov

Simulacijo poženete tako, da v meniju izberete Simulate→Run. Izriše se graf vseh šestih časovnih analiz. Graf je zaenkrat prazen, saj nismo določili izpisane veličine (izhodne napetosti senzorja). Za določitev izpisane veličine (napetostne razlike med OUT+ in OUT-), se najprej postavimo na vozlišče OUT+. Ob tem dobi kurzor obliko rdeče merilne sonde (slika 32). Kliknemo in vlečemo do vozlišča OUT-, nakar dobi kurzor obliko črne merilne sonde. Ponovno poženemo simulacijo (Simulate→Run) in dobite časovno zaporedje šestih simulacij, vsaka označena s svojo barvo (slika 33).



Šest časovnih simulacij nam dejansko nič ne pomaga - namesto tega bi radi videli, kako se spreminja izhodna napetost mostiča v odvisnosti od položaja potenciometrov RP1 in RP2. Za to se vrnemo nazaj na okno simuliranega vezja in pritisnemo CTRL+L, s čimer prikažemo dnevnik napak (ang. error log).

Tabela 6: Izpis dnevnika napak.

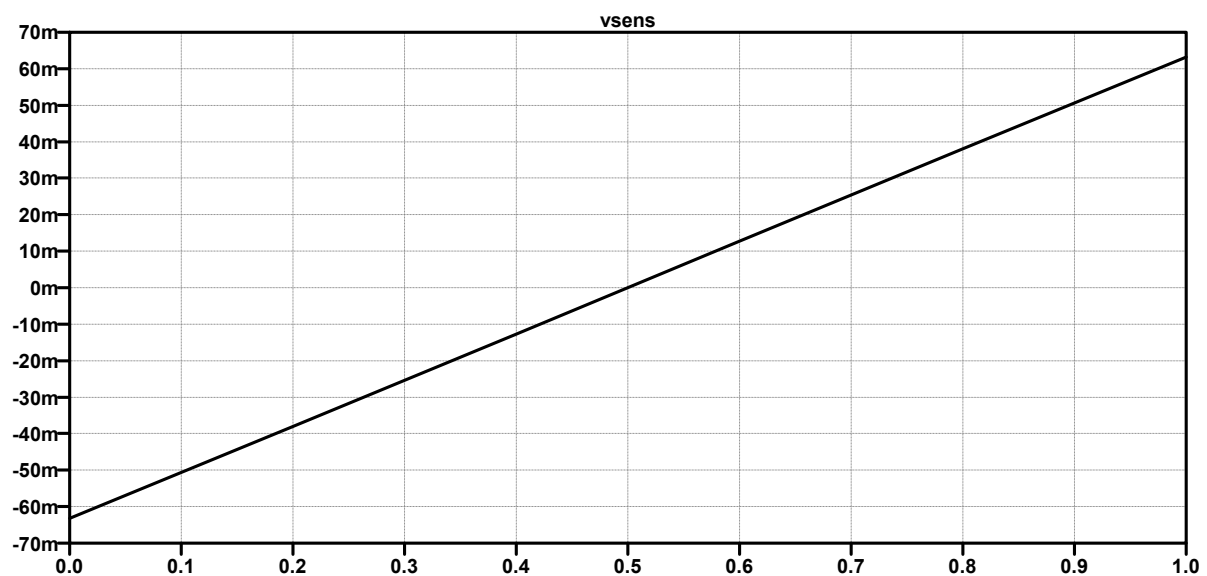
```
Circuit: * C:\Users\matej\Desktop\SIMULATOR_SENZORJA_TLAKA.asc
Direct Newton iteration for .op point succeeded.
.step dr=0
.step dr=0.2
.step dr=0.4
.step dr=0.6
.step dr=0.8
.step dr=1

Measurement: vsens
step      AVG(v(out+, out-))  FROM  TO
1         -0.0631646        0     0.001
2         -0.0379747        0     0.001
3         -0.0126582        0     0.001
4          0.0126582        0     0.001
5          0.0379747        0     0.001
6          0.0631646        0     0.001

Date: Wed Aug 14 12:53:34 2019
Total elapsed time: 0.268 seconds.
```

```
tnom = 27
temp = 27
method = modified trap
totiter = 2091
traniter = 2088
tranpoints = 1045
accept = 1045
rejected = 0
matrix size = 8
fillins = 0
solver = Normal
Matrix Compiler1: 282 bytes object code size 0.2/0.1/[0.0]
Matrix Compiler2: 540 bytes object code size 0.1/0.1/[0.0]
```

Iz izpisa v tabeli 6 razberemo povprečno izhodno napetost senzorja ($\text{AVG}(v(\text{out+}, \text{out-}))$) glede na korak (ang. step) simulacije v času od 0 do 1 ms. Če želimo te vrednosti izrisati v XY grafu (ne časovnem), potem kliknemo desno in izberemo "Plot stepp'ed meas. data" in pojavi se graf senzorske karakteristike - izhodna napetost senzorja v odvisnosti od lege potenciometra.



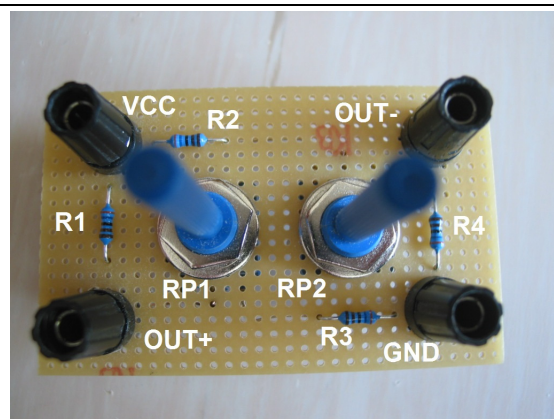
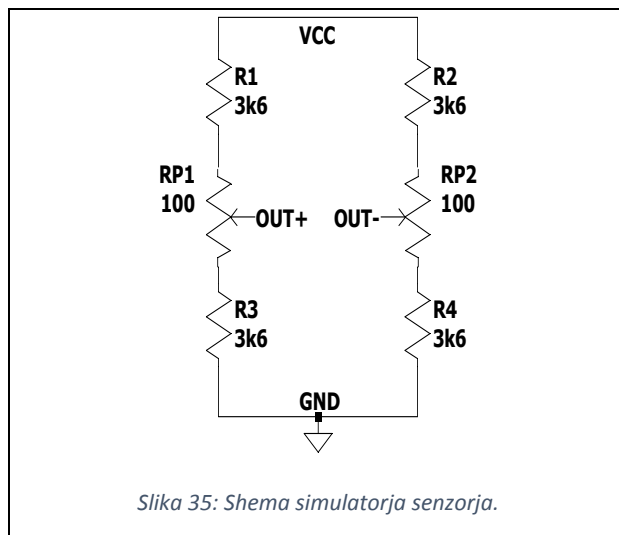
Slika 34: Izhodna napetost senzorja (V_{sens}) v odvisnosti od lege potenciometra (dR).

Simulirani rezultat se kar dobro ujema z izračunanimi vrednostmi. Preizkusite še ostale vrednosti upornosti ($3.6 \text{ k}\Omega$) in potenciometra (50Ω).

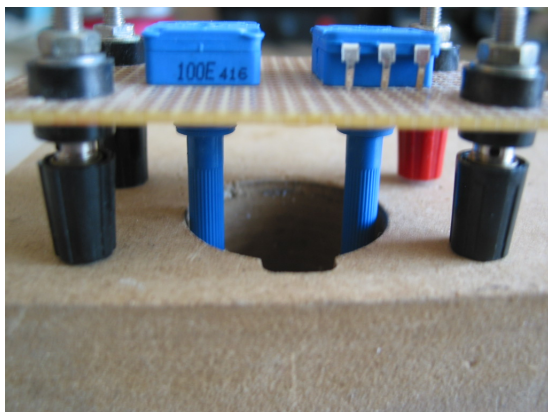
Sestavljanje simulatorja senzorja

Simulator senzorja sestavite na pripravljene luknjaste vitroplast plošči, ki že ima montirane štiri puše in dva potenciometra. Shema simulatorja senzorja je na sliki 35. Oznake elementov na sliki 35 in na ploščici (slika 36) se ujemajo. Simulator senzorja na sliki 36 ima štiri priključke: napajalnega (označen VCC), diagonalno njemu se nahaja masa senzorja (označena GND), na nasprotni diagonali pa se nahajata izhoda senzorja (označena OUT+ in OUT-).

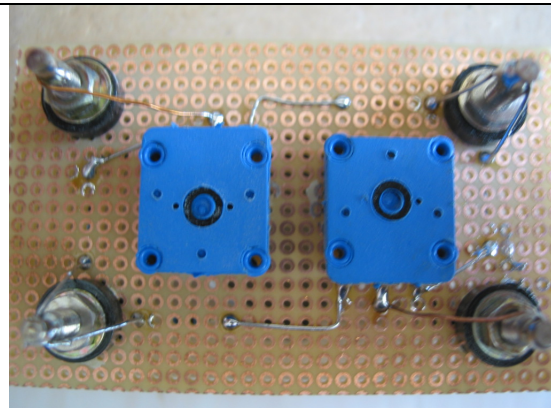
Simulator senzorja zaradi lažje montaže sestavljate na leseni kladi, saj drsnika potenciometrov segata nad nivo puše. Naprej vstavite upore R1...R4 v ustrezne luknje na ploščici (glej sliko 36) - nožic upora ne režete, ampak jih samo fiksirate s kapljico cina, da upor ne pade iz ploščice. Nožice nato skrivite tako, kot je narisano na sliki 38. Za povezavo obeh drsnikov potenciometrov (srednji priključek) uporabite bakreno žico, ki jo najprej olupite z olfa nožem, nato vtaknete skozi drsniku najbližjo luknjo, potegnite skozi do puše in jo ovijete okoli nje (slika 38).



Slika 36: Pogled na zgornjo stran sestavljenega simulatorja senzorja



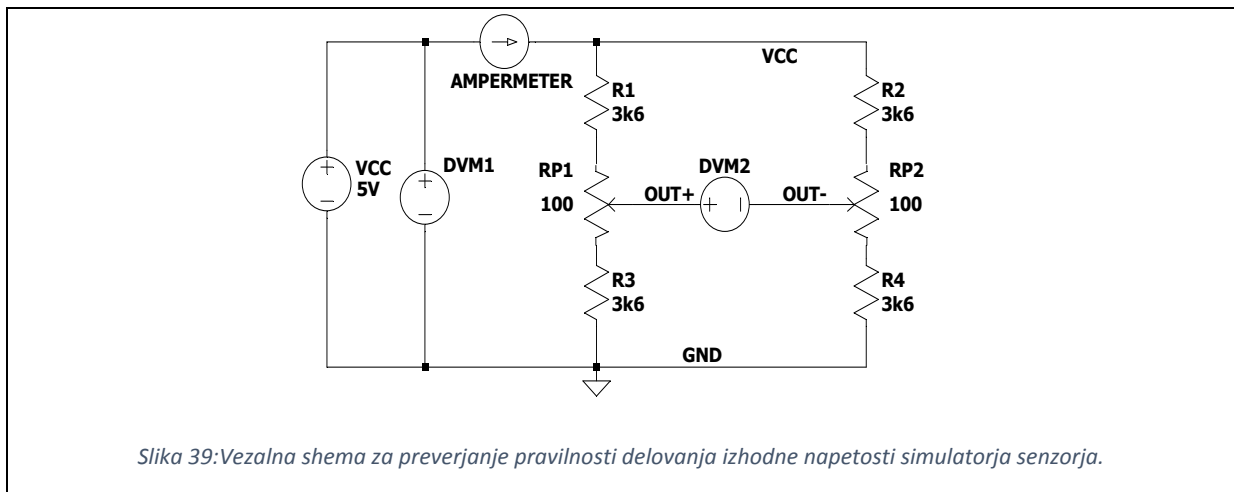
Slika 37: Vstavljanje ploščice v kos lesa z luknjo.



Slika 38: ogled na spodnjo stran sestavljenega simulatorja senzorja

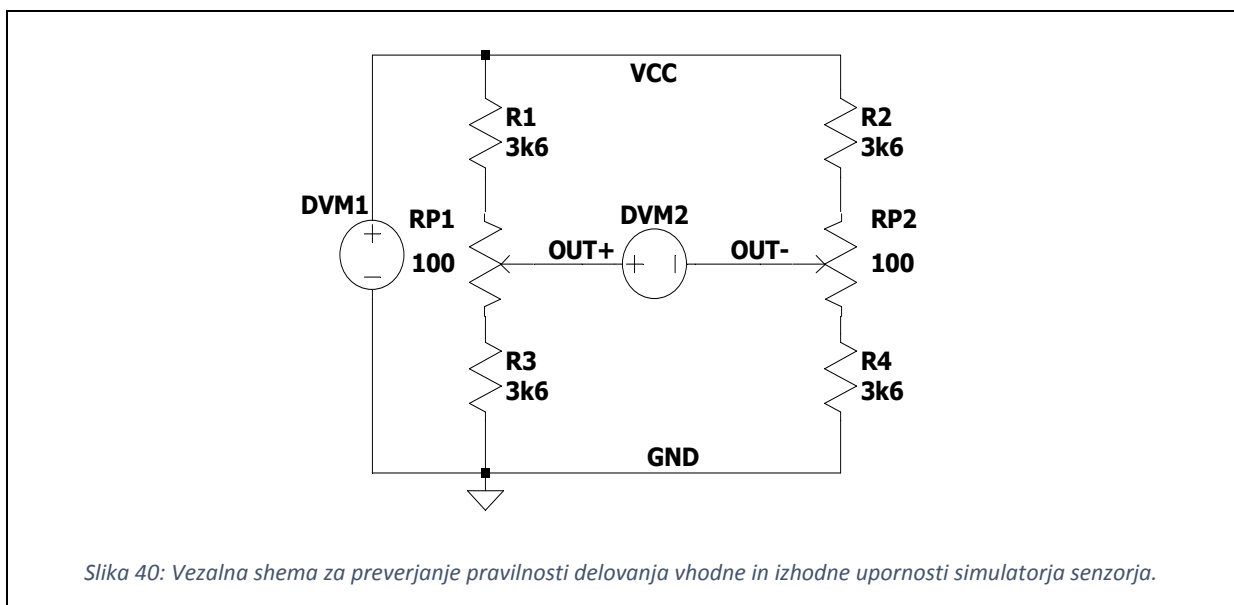
Preverjanje delovanja simulatorja senzorja

Sestavljen simulator senzorja priključite po vezalni shemi na sliki 39. Vir enosmerne napetosti nastavite na 5 V in tokovno omejitev na 60 mA. Za nastavljanje napetosti senzorja DVM1 uporabite voltmeter CDM250, za nastavljanje toka uporabite drugi CDM250, medtem ko za meritev napetosti mostiča (DVM2) uporabite voltmeter Agilent 33401A.



Potenciometra RP1 in RP2 zavrtite v skrajni legi (enega skrajno desno, drugega skrajno levo), pri čemer se izhodna napetost mostiča na DVM2 postavi na ca. 60 mV. Če položaja potenciometrov obrnete, napetost na DVM2 spremeni predznak (ca. -60 mV).

Vhodno in izhodno upornost mostiča izmerite po vezalni shemi na sliki 40, pri čemer izključite vir in ampermeter, DVM1 in DVM2 nastavite na merjenje upornosti. Vhodna upornost je med pušama drsnikov potenciometrov, izhodna med priključnima pušama za napajalno napetost.

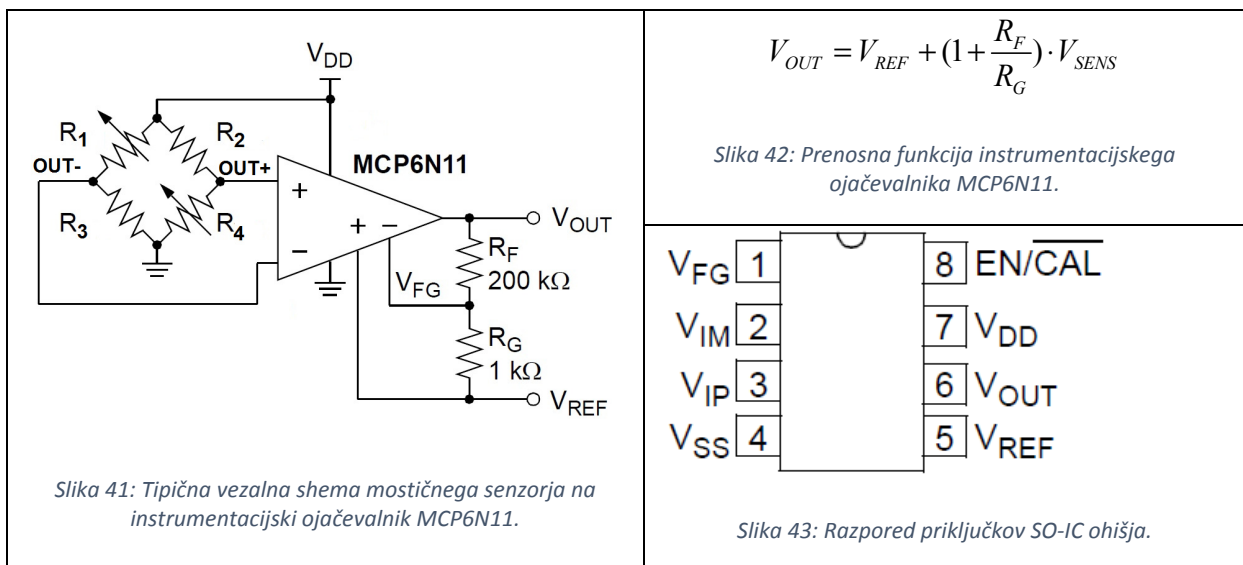


Izmerjeni vhodna in izhodna upornost mostiča sta praktično enaki, obe ca. 3.7 k Ω .

Načrtovanje ojačevalnika senzorskega signala

V preteklih meritvah senzorja smo ugotovili, da je izhodna napetost senzorja reda nekaj 10 mV. Poleg tega smo ugotovili, da je izhod senzorja med OUT+ in OUT-, čemur strokovno pravimo, da je izhod diferencialen (ang. differential output). Večino vezij za obdelavo senzorskega signala (npr mikrokrmilnik na Arduino modulu) ni diferencialnih, ampak si več merilnih kanalov deli eno skupno referenčno točko - maso (GND) proti kateri merijo napetosti. Mikrokrmilnik meri potencial vhoda analogno-digitalnega pretvornika proti masi - takemu načinu meritve pravimo *meritev s skupno maso* (ang. single-ended measurement).

Če želimo senzorski element meriti z mikrokrmilnikom, kot je na Arduino modulu, moramo signal senzorja ustrezno ojačiti in pretvoriti iz diferencialnega v meritev s skupno maso. Vezju, ki opravlja to funkcijo, pravimo ojačevalnik za prilagoditev signala senzorja (ang. signal-conditioning amplifier). V senzorskih sistemih to funkcijo pretvorbe in ojačenja največkrat opravlja *instrumentacijski ojačevalnik* (ang. instrumentation amplifier). Na vezju za prilagoditev senzorskega signala bomo uporabili ojačevalnik [MCP6N11](#) proizvajalca Microchip. Za namen delavnice bomo delovanje instrumentacijskega ojačevalnika poenostavili in ga bomo obravnavali na nivoju njegove prenosne funkcije - zgolj matematične funkcije med vhomom in izhodom. Na sliki 41 je narisana tipična vezalna shema mostičnega senzorja (kot je npr. naš senzor tlaka) na instrumentacijski ojačevalnik. Mostični senzor ima izhodno napetost (V_{SENS}) med sponkama OUT+ in OUT-, ki ju priključimo na + in - vhoda instrumentacijskega ojačevalnika. Naj bo izhodna napetost senzorja $V_{SENS} = V_{OUT+} - V_{OUT-}$. Prenosno funkcijo na sliki 42 preberemo kot: Na izhodu vezja dobimo potencial V_{OUT} , ki predstavlja ojačen in premaknjen senzorski signal V_{SENS} . Senzorski signal je ojačen (množen) s faktorjem $A_U = (1 + R_F/R_G)$. Nivo ničelne napetosti senzorja (ko je $V_{OUT+} = V_{OUT-}$) premikamo s spreminjanjem potenciala V_{REF} .



Groba ocena ojačenja instrumentacijskega ojačevalnika

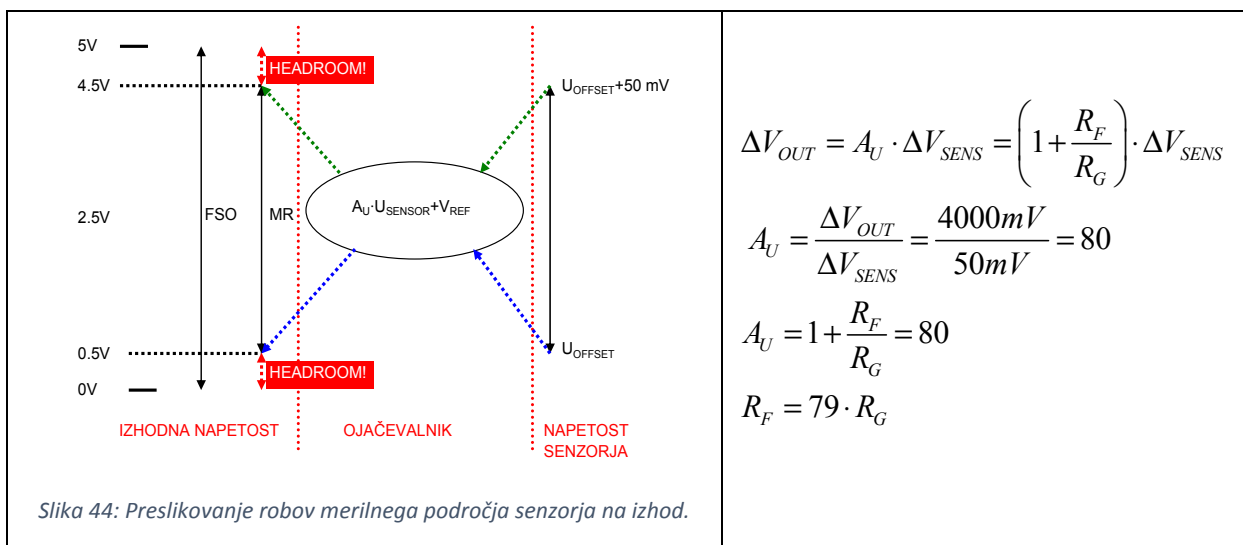
Celotno merilno področje analogno-digitalnega pretvornika na Arduino modulu je med 0 in $V_{DD}=5\text{ V}$. To merilno področje je obenem skoraj enako skrajnim mejam izhoda instrumentacijskega ojačevalnika, saj je izhod V_{OUT} v intervalu $[V_{DD} - 25\text{ mV} \dots 25\text{ mV}]$. Iz izmerjene karakteristike senzorja določimo občutljivost S (ang. sensitivity) kot spremembo napetosti (izhod senzorja) proti spremembi višine (vhod senzorja).

$$S = \frac{\Delta V_{SENS}}{\Delta h} \approx 0.5 \frac{mV}{cm}$$

Izberemo merilno področje (ang. measurement range) višinske razlike senzorja - npr. med 0 in 100 cm.

$$\begin{aligned} V_{SENS-}(0cm) &= -30mV & V_{SENS-}(100cm) &= 20mV \\ V_{SENS+}(0cm) &= -30mV & V_{SENS+}(100cm) &= -80mV \\ \Delta V_{SENS+} &= \Delta V_{SENS-} & &= 50mV \end{aligned}$$

Senzor bo pri $\Delta h = 100\text{ cm}$ višinski razliki na izhodu dal ca. 50 mV višjo napetost kot pri $\Delta h = 0\text{ cm}$. Merilno področje izhodne napetosti senzorja $V_{SENS} \in [U_{OFFSET} \dots U_{OFFSET} + 10\text{ mV}]$ želimo torej preslikati na napetost izhoda $V_{OUT} \in [0 \dots 5\text{ V}]$. Pri načrtovanju ojačevalnikov nikdar ne postavimo točke ojačenja čisto na rob merilnega področja izhodne napetosti - ne na 0 V, ne na 5 V, ampak si običajno pustimo neko *prepovedano območje oz. rezerviran prostor* (ang. headroom). Standardna velikost rezerviranega prostora je 0.5 V pri napajalni napetosti 5 V. Smisel rezerviranega prostora je detekcija napak: Če bomo na Arduino modulu zaznali napetost znotraj rezerviranega prostora, bomo to lahko javili uporabniku kot napako na senzorju ali na ojačevalniku. Na sliki 44 je označeno preslikovanje vhoda senzorja na izhod ojačevalnika. Celotno področje izhoda (ang. full scale output, FSO) je 5 V. Rezerviran prostor je 0.5V od obeh napajalnih nivojev. Od tega je merilno področje izhoda (ang. measurement range) $\Delta V_{OUT} = 4.5\text{ V} - 0.5\text{ V} = 4\text{ V}$, razpon senzorskega signala je 50 mV.



Še nasvet pri izbiri velikosti ojačenja - na prvi pogled se zdi, kot da ojačenje lahko nastavljamo do zelo visokih vrednosti (1000 in več). Običajno se ojačenja nad nekaj 100 redko delujejo brez oscilacij izhoda V_{OUT} , zato se v tem primeru ojačevalnik naredi s pomočjo dveh ali več stopenj, pri čemer se stopnje vežejo v verigo, ojačenja posameznih stopenj pa se med sabo množijo.

Naslednji problem so merilna področja senzorjev, pri katerih so razponi izhodne napetosti (ΔU_{SENS}) manjši, kot je velikost ničelne napetosti U_{OFFSET} . Po enačbi na sliki 42 instrumentacijski ojačevalnik ojačuje razliko vhodnih napetosti - torej tudi U_{OFFSET} , kljub temu, da nam pri sami meritvi ne koristi. Če ojačuje U_{OFFSET} , potem se dejansko merilno področje izhoda zmanjša, saj se pod potencialom ojačene U_{OFFSET} ne "dogaja nič". Tega se ne moremo enostavno znebiti, pač pa običajno uporabimo tak senzor, ki ima odprt Wheatstoneov mostič (pet kontaktov - spodnji veji mostiča sta vsaka zase pripeljana na izhod in ju vežemo na zunanji potenciometer, ki ničelno napetost popolnoma kompenzira). Žal naš senzor ni odprt mostič, ampak navaden, tako da moramo zmanjšanje merilnega področja zaradi U_{OFFSET} upoštevati, kot bomo videli v nadaljevanju.

Izračun prilagoditve ničelne napetosti

Senzor tlaka bo meril višinsko razliko, kar pomeni da bomo izkoriščali samo eno polovico izmerjene karakteristike (višinska razlika je recimo samo pozitivna). Ničelno napetost senzorja (ang. offset voltage) bomo odštevali od izhoda tako, da dobimo spodnjo mejo merilnega področja ojačevalnika (0.5V).

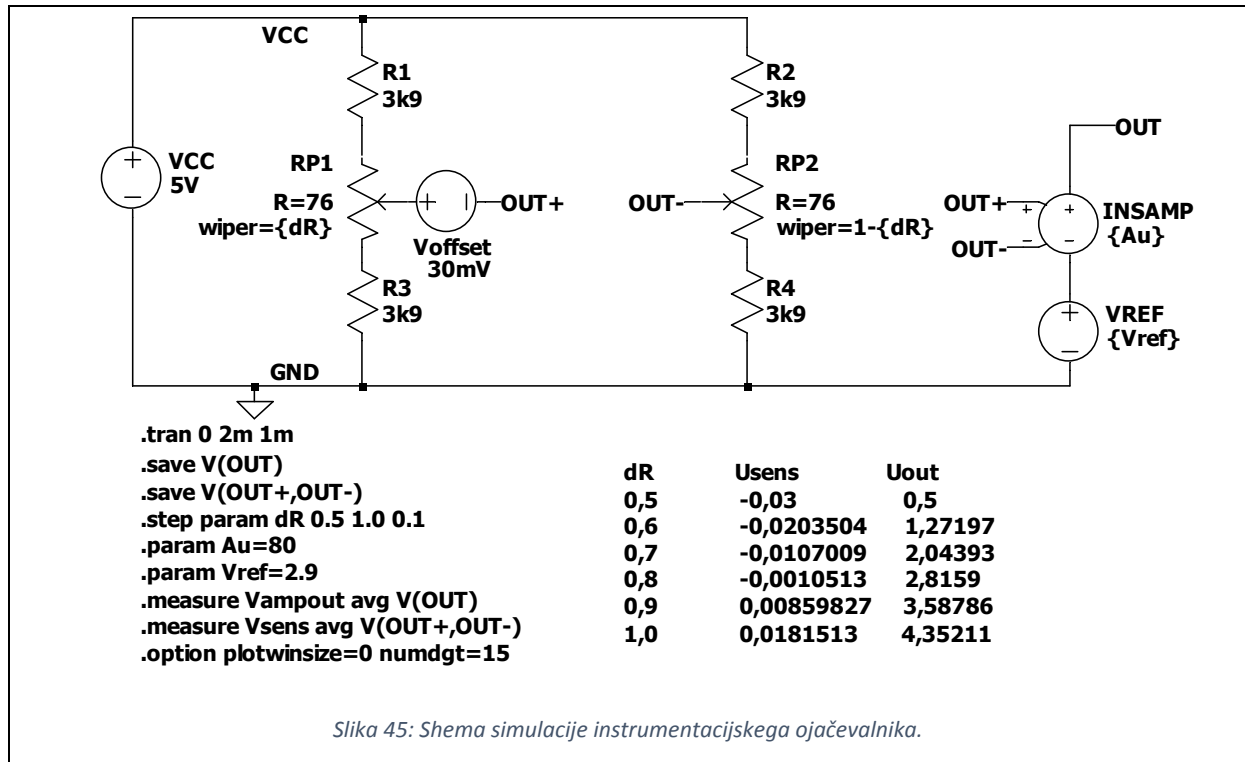
$$\begin{aligned}0.5V &= A_U \cdot V_{OFFSET} + V_{REF} \\0.5V &= 80 \cdot (-30mV) + V_{REF} \\V_{REF} &= 2.9V\end{aligned}$$

Iz zgornje enačbe sledi, da se V_{REF} nahaja nekje okoli sredine območja napajalne napetosti (med 0 in 5 V).

Simulacija ojačevalnika senzorskega signala

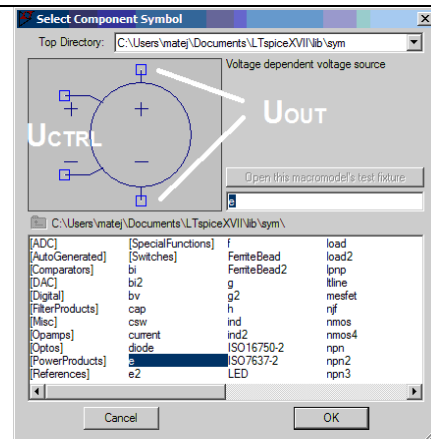
Vezju simulatorja senzorja dodamo še elemente, s katerimi bomo modelirali (poustvarili z električnimi elementi) prenosno karakteristiko instrumentacijskega ojačevalnika

$$V_{OUT} = A_U \cdot V_{SENS} + V_{REF}$$




Slika 45: Shema simulacije instrumentacijskega ojačevalnika.

$$U_{OUT} = A_U \cdot U_{CTRL}$$



Slika 46: Napetostno krmiljeni napetostni vir v SPICE

Ojačenje instrumentacijskega ojačevalnika modeliramo s pomočjo napetostno krmiljenega napetostnega

vira (izberete orodje za izbiro komponent , odpre se pogovorno okno na sliki 45). Napetostno krmiljenega napetostnega vir ojačuje krmilno napetost U_{CTRL} (glej sliko 46) z napetostnim ojačenjem A_U , ki je parameter komponente. Produkt ojačenja in krmilne napetosti U_{CTRL} je izhodna napetost U_{OUT}

(glej sliko 46). Na sliki 45 je napetostno ojačenje nastavljeno na $A_U=80$, referenčna napetost $V_{REF} = 2.9 \text{ V}$.

Poganjanje simulacije in izpis rezultatov

Simulacijo poženite s Simulate→Run. Izriše se graf vseh šestih časovnih analiz nastavitve drsnika potenciometra. Drsnik potenciometra nastavljamo med 0.5 in 1.0, podobno kot bi v resničnem svetu senzor tlaka obremenjevali samo v eno stran. Podobno kot prej pritisnite CTRL+L za izpis dnevnika napak.

```
.step dr=0.5
.step dr=0.6
.step dr=0.7
.step dr=0.8
.step dr=0.9
.step dr=1
Measurement: vampout
  1  0.5      0      0.001
  2  1.27197 0      0.001
  3  2.04393 0      0.001
  4  2.8159  0      0.001
  5  3.58786 0      0.001
  6  4.35211 0      0.001
Measurement: vsens
  1 -0.03     0      0.001
  2 -0.0203504 0      0.001
  3 -0.0107009 0      0.001
  4 -0.0010513 0      0.001
  5  0.00859827 0      0.001
  6  0.0181513 0      0.001
```

Slika 47: Izpis simulacije prenosne karakteristike instrumentacijskega ojačevalnika.

Iz slike 47 sledi, da bi pri maksimalnem senzorskem signalu ($U_{SENS}=+18.15 \text{ mV}$, ki ustreza višinski razliki $\Delta h = 100 \text{ cm}$) izhod instrumentacijskega ojačevalnika V_{OUT} dosegel vrednost 4.352 V , medtem ko bi pri najmanjšem senzorskem signalu ($U_{SENS} = -30 \text{ mV}$, ki ustreza $\Delta h = 0 \text{ cm}$) izhod dosegel $V_{OUT} = 0.5 \text{ V}$, kar se precej dobro ujema z velikostjo rezerviranega prostora (ang. headroom).

Izbira realnih vrednosti upornosti R_F in R_G

Po podatkovnem listu instrumentacijskega ojačevalnika postavimo vrednost upora R_G v območje kilohm (npr. $R_G = 10 \text{ k}\Omega$) R_F določimo po enačbi na sliki 42, da bo napetostno ojačenje $A_U \approx 80$. Točna vrednost upora bi bila $R_F = 790 \text{ k}\Omega$. Takšna vrednost bi zahtevala upor E192 lestvice, ki je drag. Najbližja vrednost iz cenejše E12 lestvice znaša $R_F = 820 \text{ k}\Omega$. Upor R_F lahko sestavimo iz nekaj zaporedno vezanih upornosti, pri čemer uporabimo upore $680 \text{ k}\Omega$, $10 \text{ k}\Omega$ in $1 \text{ k}\Omega$.

Druga možnost je, da tabeliramo vrednosti E12 lestvice za R_F in R_G , izračunamo (R_F/R_G) ter poiščemo minimum odstopanja od idealnega razmerja. Na Excelovem listu naredimo tabelo E12 vrednosti uporov: R_F damo v prvi stolpec in R_G v drugo vrstico. V celico ($R_F=1$, $R_G=1$) na koordinati lista B3 vpišemo formulo: $=\text{IF}((\text{\$A3/B\$2})<1;(\text{\$A3/B\$2})*10;(\text{\$A3/B\$2}))-7,90)^2$ in jo kopiramo preko ostalih celic. Formula normalizira razmerje R_F/R_G (skalira v dekada med 1 in 10), izračuna razliko do idealnega razmerja 7.90 in to odstopanje kvadrira.

Tabela 7: Iskanje vrednosti minimalnega odstopanja od razmerja $R_F/R_G=7.9$.

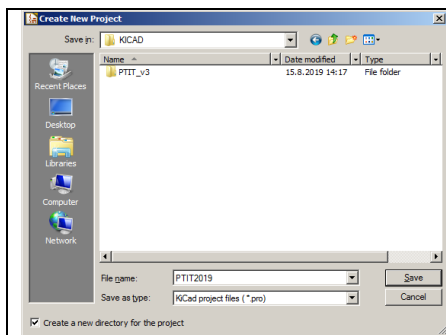
$R_F \backslash R_G$	1,00	1,20	1,50	1,80	2,20	2,70	3,30	3,90	4,70	5,60	6,80	8,20
1,00	47,61	0,19	1,52	5,50	11,25	17,61	23,71	28,47	33,32	37,38	41,34	44,63
1,20	44,89	47,61	0,01	1,52	5,98	11,94	18,18	23,26	28,59	33,14	37,64	41,43
1,50	40,96	44,22	47,61	0,19	1,17	5,50	11,25	16,43	22,17	27,26	32,42	36,85
1,80	37,21	40,96	44,89	47,61	0,08	1,52	5,98	10,79	16,57	21,96	27,59	32,55
2,20	32,49	36,80	41,39	44,59	47,61	0,06	1,52	5,10	10,36	15,77	21,76	27,22
2,70	27,04	31,92	37,21	40,96	44,53	47,61	0,08	0,95	4,65	9,48	15,44	21,23
3,30	21,16	26,52	32,49	36,80	40,96	44,59	47,61	0,32	0,77	4,03	9,28	15,02
3,90	16,00	21,62	28,09	32,87	37,54	41,67	45,13	47,61	0,16	0,88	4,69	9,88
4,70	10,24	15,87	22,72	27,97	33,22	37,94	41,94	44,82	47,61	0,24	0,98	4,70
5,60	5,29	10,45	17,36	22,93	28,67	33,94	38,48	41,78	45,00	47,61	0,11	1,15
6,80	1,21	4,99	11,33	16,99	23,13	28,96	34,10	37,90	41,64	44,70	47,61	0,15
8,20	0,09	1,14	5,92	11,19	17,41	23,65	29,32	33,61	37,89	41,42	44,81	47,61

Iz tabele 7 naboljši približek idealnega razmerja ($R_F/R_G = 79$) dobimo pri vrednostih upora $R_F = 120 \text{ k}\Omega$ in $R_G = 1.5 \text{ k}\Omega$. Dejanska vrednost razmerja R_F/R_G je 80, celotno ojačenje pa po enačbi instrumentacijskega ojačevalnika (slika 42) enako $A_U = 81$. Če ponovimo LTSpice analizo in parameter ojačenja postavimo na (.param $A_U=81$), dobimo pri skrajnih točkah senzorske karakteristike rahel premik nivojev izhodne napetosti: $V_{\text{OUT}}(U_{\text{SENS}}=U_{\text{OFFSET}}) = 0.47 \text{ V}$, pri $V_{\text{OUT}}(U_{\text{SENS}}=U_{\text{OFFSET}}+50 \text{ mV}) = 4.37 \text{ V}$. Dobljeni vrednosti se še vedno dobro ujemata z velikostjo rezeviranega prostora (ang. headroom).

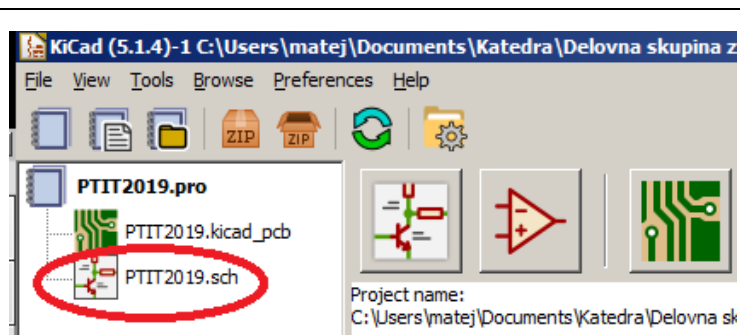
Načrtovanje vezja ojačevalnika senzorskega signala v KICAD

Na Namizju (ang. Desktop) poženemo program KiCAD. V glavnem oknu izberemo File→New→Blank. Pojavi se pogovorno okno na sliki 48, v katerega vnesemo ime projekta "PTIT2019" in potrdimo (Save). Program vpraša, če ustvari ločen imenik za projekt - potrdimo (Yes). Odpre se projektno okno (del je na sliki 49). V orodni vrstici projektnega okna lahko poženemo tri ločene programe:

- shematski urejevalnik (EESchema), s katerim narišemo vezalno shemo,
- program za načrtovanje tiskanih vezij (PCB editor)
- preslikovalnik (cvPCB), s katerim določimo ohišja elementov in preslikujemo spremembe med programoma EESchema in PCB editor.



Slika 48: Okno ob odpiranju praznega projekta.



Slika 49: Del glavnega projektnega okna.



Program za risanje vezalnih shem (Schematic layout editor)





Urejevalnik elementov (Symbol editor)



Program za risanje tiskanih vezij (PCB editor)

Projekt že vsebuje predpripravljena dokumenta za risanje sheme (PTIT2019.sch) in načrt tiskanega vezja (PTIT2019_kicad.pcb). Z dvoklikom na dokument PTIT2019.sch (slika 49) v projektne oknu se odpre program za risanje vezalnih shem EESchema.

Elemente začnemo postavljati po vezalni shemi s klikom na ikono  (ang. place symbol), ki se nahaja v orodni vrstici **na desnem robu programa** (na vrhu je podobna ikona). Odpre se izbiralnik elementov (ang. choose symbol). V iskalno okno izbiralnika elementov (slika 50) vnesemo ime komponente, ki jo želimo postaviti. Vnesemo "resistor" in se pomaknemo po izbirah na komponento "R". Komponento premikate tako, da se postavite nanjo in kliknete levo. Dodatne operacije so na voljo na desnem kliku. Za podvojevanje (ang. duplicate) več enakih elementov uporabite tipko 'C'. Komponento obračate s tipko 'R', ko je označena. Povezave (ang. wire) med elementi naredite z izbiro orodja  v orodni vrstici na desnem robu programa EESchema.

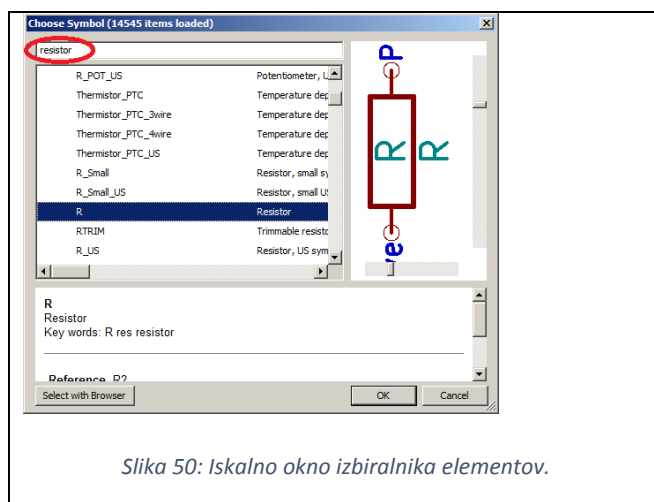








Tabela 8: uporabljeni elementi senzorskega ojačevalnika.

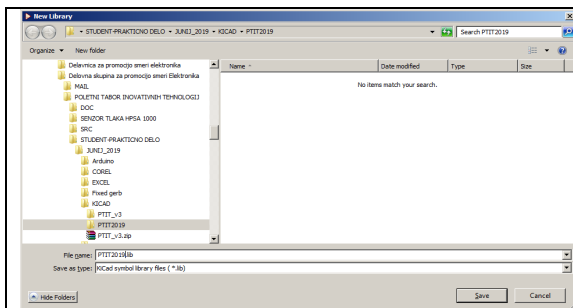
Element sheme	Ime elementa v iskalniku	Slika elementa v EESchema
Upor	R	
Potenciometer	R_POT_TRIM	
Konektor	Conn_01x02	
Napajanje in masa	VCC GND	

Risanje nove komponente v urejevalniku simbolov

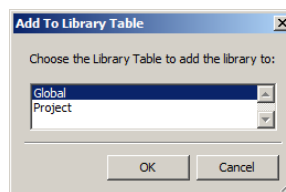
V vezju moramo dodati tudi komponento instrumentacijskega ojačevalnika, ki je v knjižnicah KiCAD

ni, zato jo moramo narisati znova. S klikom na ikono  (ang. create, edit and delete symbols), ki se nahaja v orodni vrstici **pod menijem programa**, poženemo urejevalnik simbolov (ang. symbol editor). Najprej naredimo novo knjižnico elementov, tako da v meniju izberemo File→New library. V pogovorno okno vnesemo ime knjižnice "PTIT2019" in potrdimo (Save). ter izberemo "Global", da bo knjižnica na voljo za vse projekte, ne samo za trenutnega.

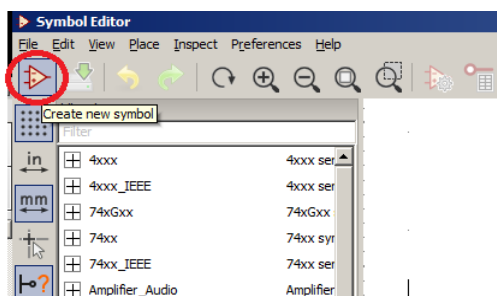
V zgornjem levem kotu (slika 53) kliknemo  (ang. create new symbol). Odpre se okno izbire ciljne knjižnice, kamor bomo shranili novo komponento (slika 54). V filter vnesemo PTIT2019 in izberemo knjižnico s tem imenom.



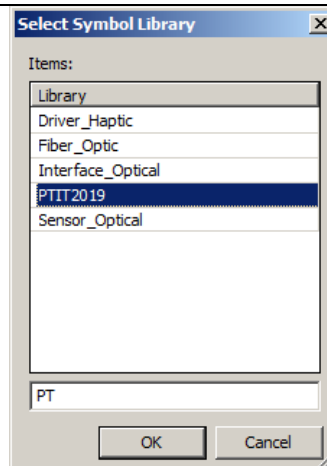
Slika 51: Shranjevanje knjižnice





Slika 52: Razpoložljivost knjižnice.

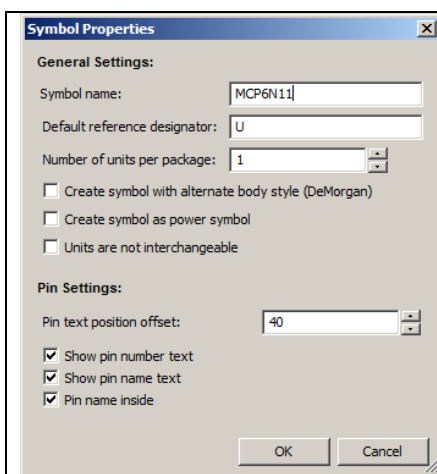


Slika 53: Izsek okna urejevalnika simbolov.

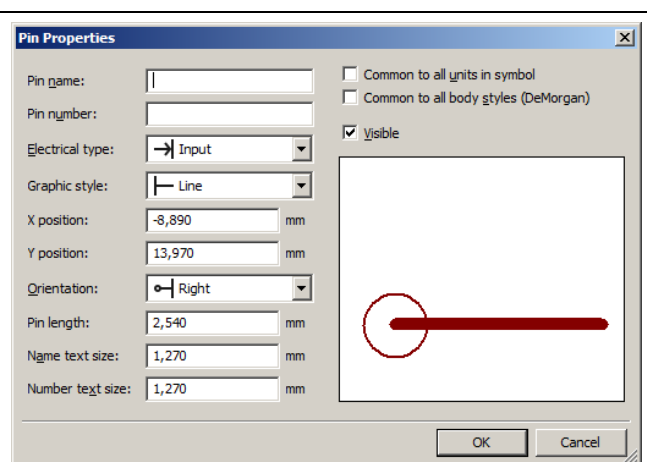


Slika 54: Izbira ciljne knjižnice

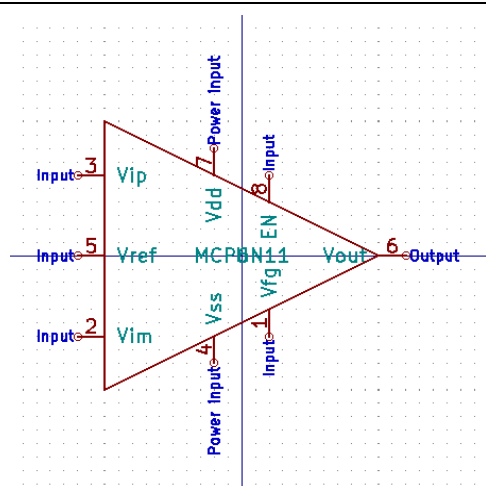
Pojavi se okno z nastavitvami elementa - kot ime simbola (ang. symbol name) vnesemo ime komponente "MCP6N11". S klikom na ikono  (ang. add lines and polygons to symbol body), ki se nahaja v orodni vrstici **na desnem robu programa**, začnemo risati osnovni trikotnik ojačevalnika. Če se zmotimo, kliknemo desno in oglišče trikotnika prestavimo s tipko 'G' (ang. drag edge point). Ko narišemo osnovni trikotnik, mu s klikanjem na  dodamo vseh 8 priključkov po seznamu iz tabele 9. Za vsak priključek se pojavi novo pogovorno okno "lastnosti priključka" (ang. pin properties), v katerega vnesemo ime (ang. name), številko (ang. number) in vrsto (ang. type). Vrsta priključka je pomembna pri preverjanje pravil (ang. design-rule-checking), pri čemer KiCAD preveri, če smo kje povezali dva izhoda med sabo. Ko smo končali s postavljanjem priključkov, shranimo simbol File→Save oz. pritisnemo CTRL+S.



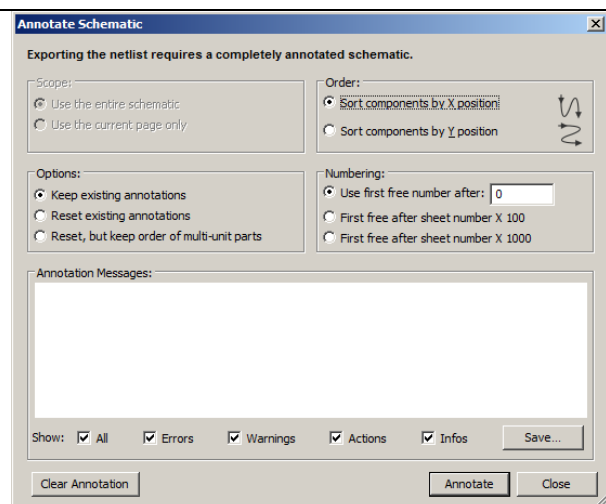
Slika 55: Nastavitve nove komponente.



Slika 56: Dodajanje priključka komponenti



Slika 57: Končana komponenta MCP6N11

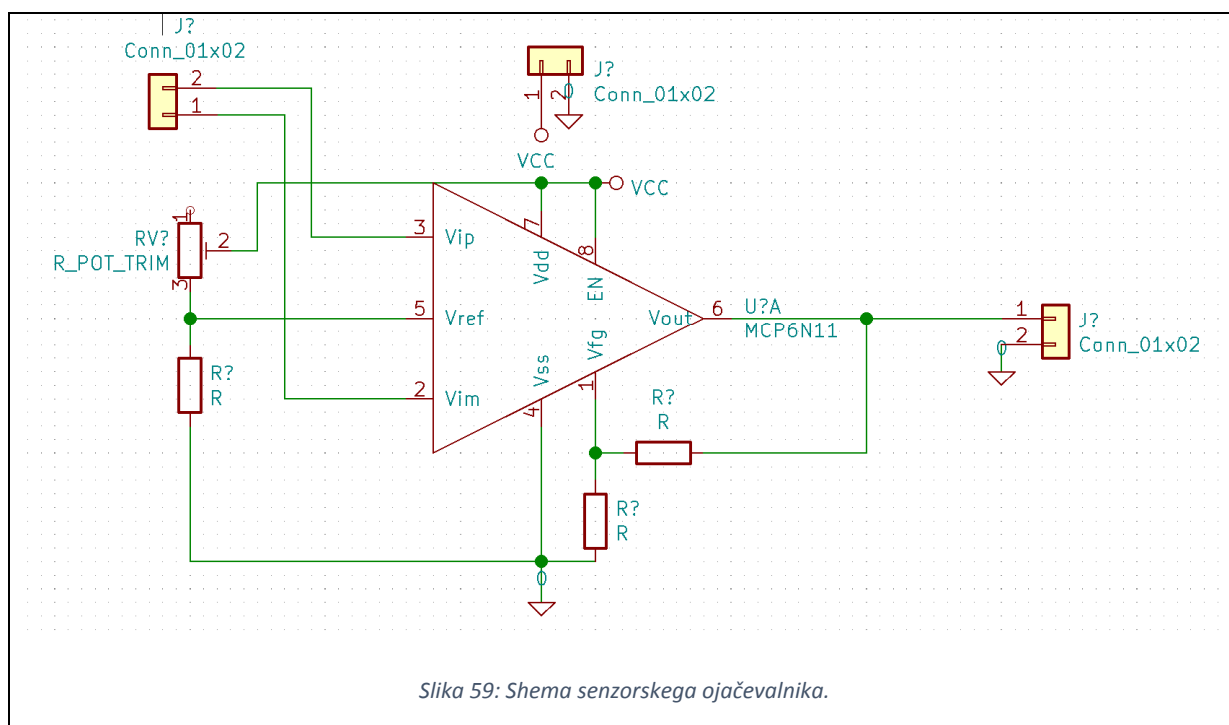


Slika 58: Oštevilčenje komponent.

Tabela 9: Seznam priključkov MCP6N11.


#	Opis	Ime	Vrsta
1	Vhod delilnika ojačevalnika	VFG	vhod (ang. input)
2	Invertirajoči vhod	-	vhod (ang. input)
3	Neinvertirajoči vhod	+	vhod (ang. input)
4	Masa	VSS	močnostni vhod (ang. power input)
5	Referenčna napetost	VREF	vhod (ang. input)
6	Izhod	VOUT	Izhod (ang. output)
7	Napajanje	VDD	močnostni vhod (ang. power input)
8	Delovanje/Umerjanje	EN	vhod (ang. input)

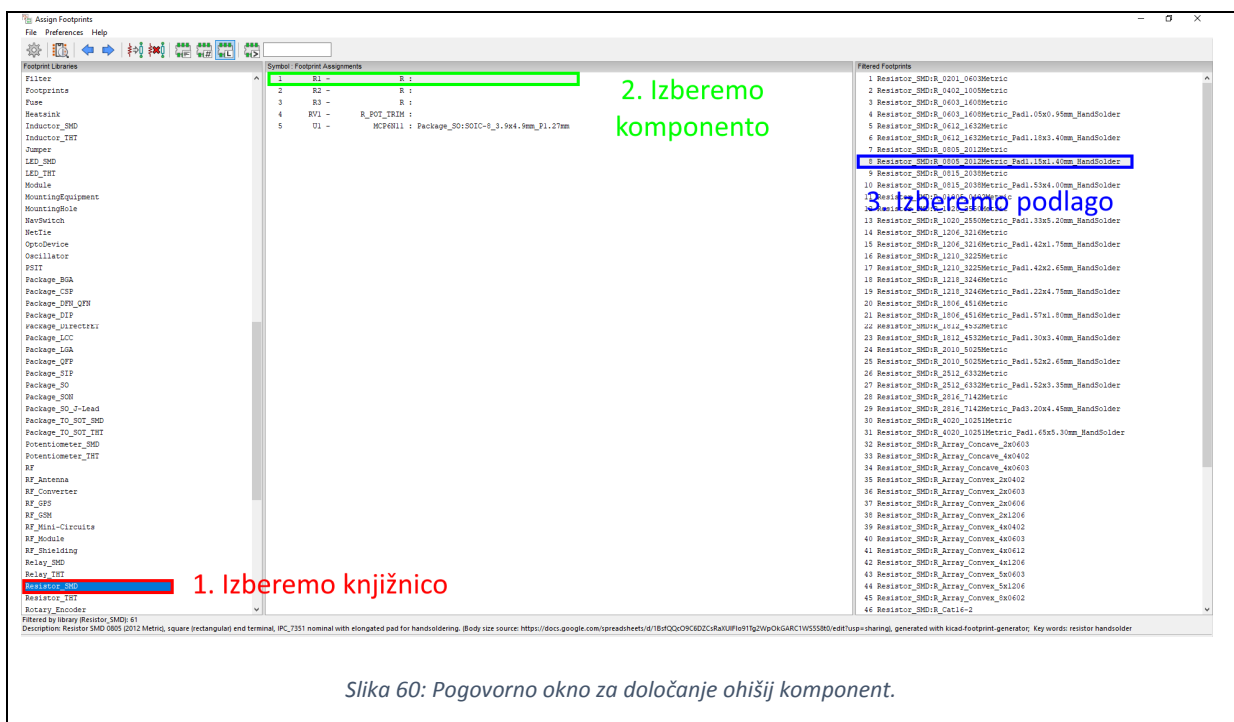
Ko smo simbol narisali, ga postavimo v shemo podobno, kot smo z iskalnikom simbolov prej postavljali preostale elemente.



Slika 59: Shema senzorskega ojačevalnika.

Povezovanje ohišij elementov in simbolov scheme.

S klikom na ikono  (ang. assign PCB footprints to schematic symbols), ki se nahaja v orodni vrstici **pod menijem programa**, začnemo izbirati ohišja postavljenih komponent. Pojavi se pogovorno okno na sliki 58 za avtomatsko oštevilčenje komponent na shemi (ang. annotate schematic). Potrdimo (Annotate) in v oknu se pojavi seznam dejanj, ki jih bo KICAD izvedel. Potrdimo (Ok) in če je vse pravilno, se okno zapre. Odpre se pogovorno okno za določanje ohišij (ang. assign footprints). V tem oknu se na sredini pojavijo komponente naše sheme. Na sliki 60 sledimo zaporedje izbiranja in določimo ohišja za vse elemente iz tabele 10.






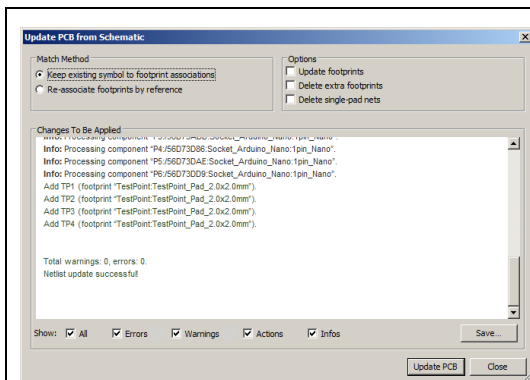
Slika 60: Pogovorno okno za določanje ohišij komponent.

Tabela 10: Seznam ohišij komponent za ojačevalnik senzorja.

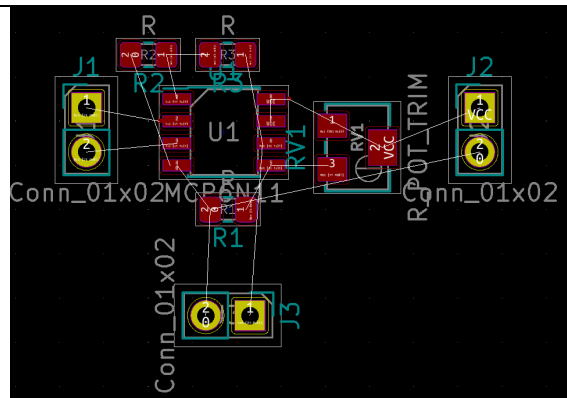
Komponenta	Oznaka	Knjižnica	Komponenta
MCP6N11	U1	Package_SO	SOIC-8_3.9x4.9mm_P1.27mm
R	R1, R2, R3	Resistor_SMD	R_0805_2012Metric_Pad1.15x1.40mm_ HandSolder
R_POT_TRIM	RV1	Potentiometer_SMD	Potentiometer_Bourns_3214W_Vertical
Conn_01x02	J1, J2, J3	Connector_PinHeader_2.54mm	PinHeader_1x02_P2.54_Vertical

Risanje ploščice tiskanega vezja

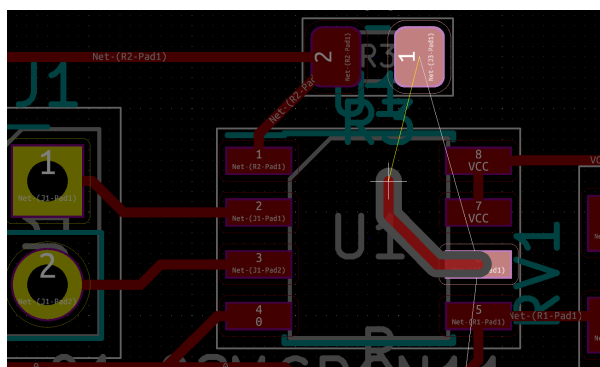
V glavnem projektnem oknu kliknemo ikono  urejevalnika tiskanih vezij (PCBEditor). V orodni vrstici pod menijem programa kliknemo ikono  (ang. update PCB from schematic). Odpre se pogovorno okno za vstavljanje elementov iz načrtovalnika shem (ang. update PCB from schematic). Potrdimo (Update PCB) in zapremo (Close). Pojavi se začetna postavitev elementov (slika 62). Elemente nato premikamo in vrtimo dokler vsa ohišja niso na primerem mestu (konektorji na robu, čimmanj križanih povezav). Nato kliknemo ikono za ustvarjanje povezav  in pričnemo z risanjem (slika 63). Če pot od začetka do cilja (osvetljeno na sliki 63) ni mogoča po trenutni strani, lahko plast (ang. layer) zamenjamo tako, da s pritiskom na tipko 'V' med vlečenjem povezave, vstavimo skozniki (ang. via).



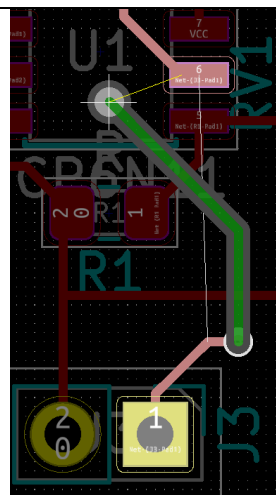
Slika 61: Vstavljanje elementov iz načrtovalnika shem.



Slika 62: Začetna postavitev komponent.



Slika 63: Risanje povezave.

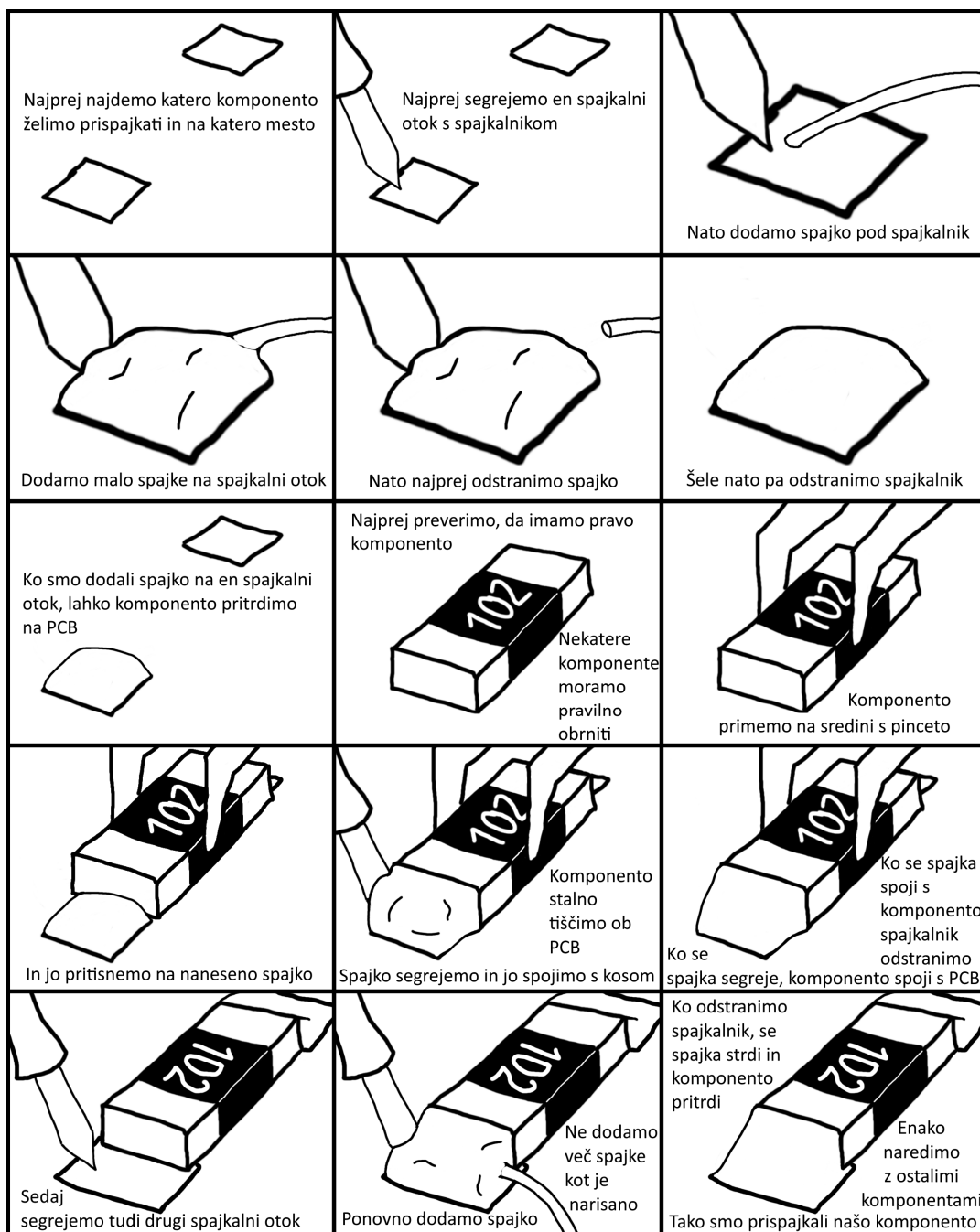


Slika 64: Povezava, izvedena s skozniki.

Če ima KICAD nameščeno komponento FreeCAD, si lahko ogledamo (meni View→3D View) tudi 3D model izdelanega vezja.

Spajkanje SMD elementov

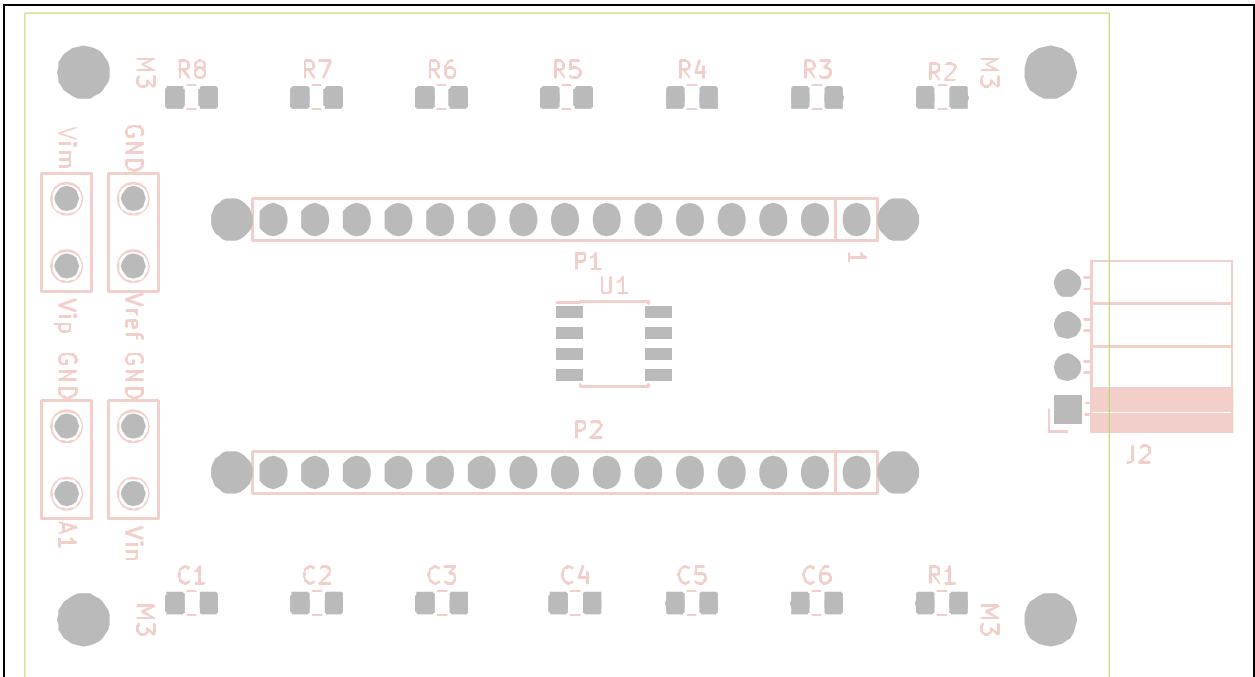
Slika 65 prikazuje zaporedje korakov pri spajkanju SMD elementov. Ne glede na to, koliko nogic vezja moramo spajkati - začnemo vedno tako, da na eno (referenčno nogico) naneseemo fluks, nato naneseemo kapljico spajke, element primemo s pinceto, ga postavimo na pravilno mesto in referenčno nožico spajkamo. Nato postopek ponovimo za nasprotno (če je vezje z več nožicami - diagonalno) nožico. Šele ko sta obe nožici do konca spajkani, preidemo na ostale nožice. Na koncu obrišemo fluks.



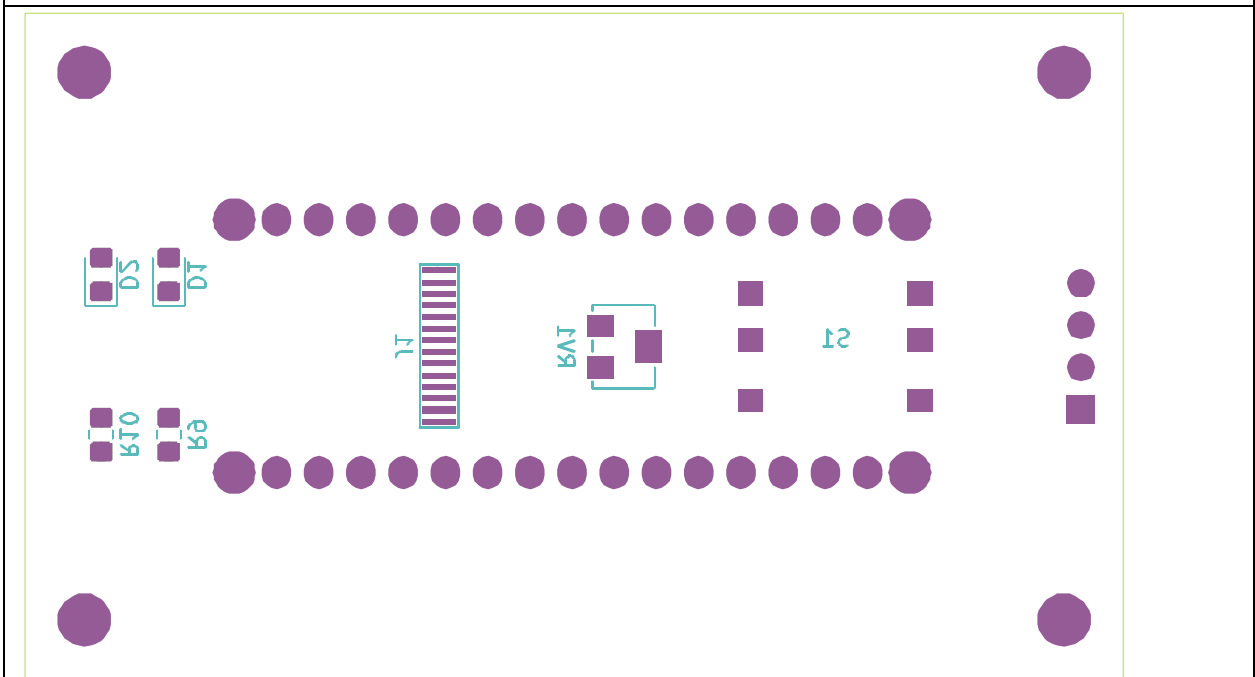
Slika 65: Postopek spajkanja SMD elementa.

Ploščica tiskanega vezja

Na slikah 68 in 69 najdete ustrezne elemente, ki jih želite spajkati. V tabeli 11 najdete vrednost ustreznega elementa. Začnite s fizično najmanjšimi in najnižjimi elementi: najprej upori, kondenzatorji nato ojačevalnik U1. Letvici P1 in P2 ter konektor J2 spajkate na koncu. Zaslona je že zaspajkan (J1). Na skrajni levi strani TIV (slika 68) so testne točke (GND, Vinp, Vinn, Vref, A1), ki se ne spajkajo.



Slika 66: TIV - elementi zgornja stran (pogled na zgornjo stran ploščice)



Slika 67: TIV - elementi spodnja stran ("rentgenski" pogled skozi ploščico)

Seznam uporabljenega materiala (ang. bill of materials)

Tabela 11: Seznam uporabljenega materiala

Oznaka na TIV	Vrednost	Opis elementa
C1, C2, C5, C6	1 μ F	1 μ F, 16V, 10%, X7R, 0805
C3	4,7 μ F	4.7 μ F, 16V, 10%, X7R, 0805
C4	2,2 μ F	2.2 μ F, 16V, 10%, X7R, 0805
D1**	LED RDEČA	0805, RED, 40MCD, 635NM
D2**	LED ZELENA	0805, GREEN, 40MCD, 566NM
P1,P2	Letvica moška 2.54, 15 pin	40POS, 1ROW, 2.54MM
J1*	Zaslon	OLED, GRAPHIC, TAB, 64X32 PIXELS
J2	Priključek za senzor	SSW-104-02-G-S-RA:RECEPTICLE, 2.54MM
R1, R7, R8	100 Ω	100R, 5%, 0.1W,0805, THICK FILM
R2	1,5 k Ω	1K5, 1%, 0805, THIN FILM
R3	120 k Ω	120K, 1%, 0.125W,0805, THIN FILM
R4	820 Ω	820R, 1%, 0.125W,0805, THICK FILM
R5, R6	4,7 k Ω	4K7, 1%, 0.125W,0805, THICK FILM
R9,R10	330 Ω	330R, 5%, 0.1W,0805, THICK FILM
RV1	Potenciometer	22KOHM, 10%, 5TURN, SMD
S1	Navigacijsko stikalo	NAVIGATION SWITCH, 5WAY, SMD
U1	Instrumentacijski ojačevalnik	AMP, INSTR, 800UA, 8SOIC

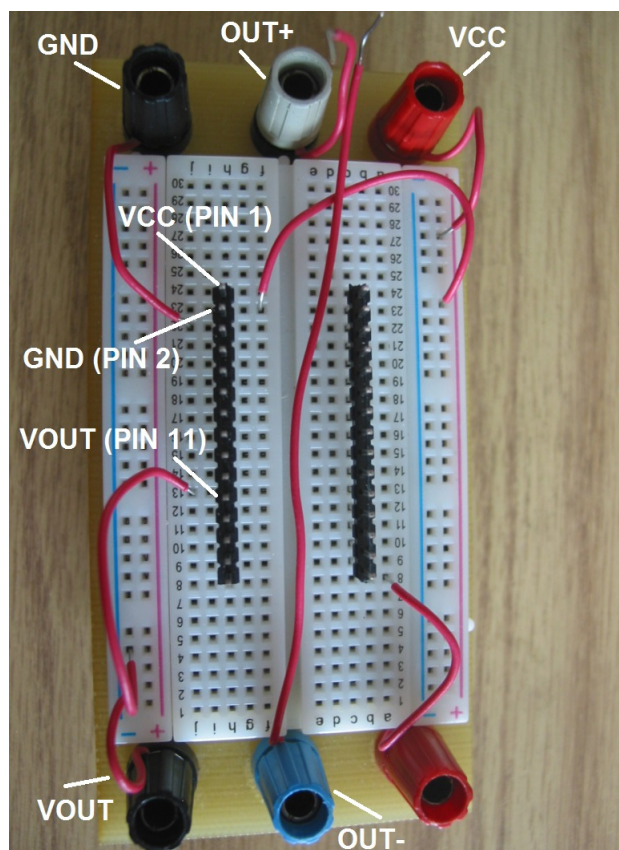
* Element je že zaspajkan.

** Element se ne montira.

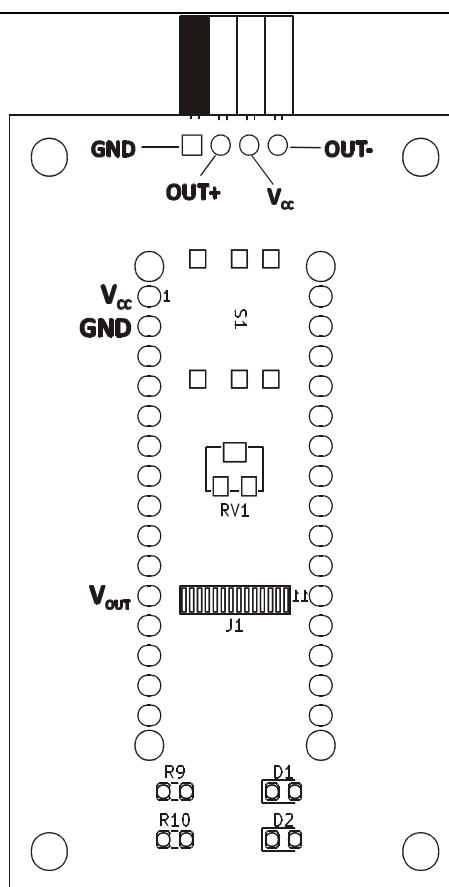
Meritve ojačevalnika senzorskega signala

Sestavljeno tiskano vezje (TIV) priključimo na preizkusno ploščico s pomočjo dveh moških letvic (Slika 68). Na levi letvici zgoraj je priključek 1 - tja povežemo žico puše VCC, sledi mu puša GND na priključek 2, na priključku 11 se nahaja izhodna napetost instrumentacijskega ojačevalnika VOUT (spodnja leva črna puša, slika 68).

Na sliki 69 je predstavljeno vezje senzorskega ojačevalnika s spodnje strani, pripravljen, da se letvici P1 in P2 vtakneta v letvici na preizkusni plošči. Vhoda za senzorski modul OUT- in OUT+ sta povezana na konektor na vrhu slike 69 - sivo pušo (OUT+) z žico povežete na 3. priključek z desne konektorja za senzor, modro pušo pa povežete na prvi priključek konektorja za senzor na sliki 69.

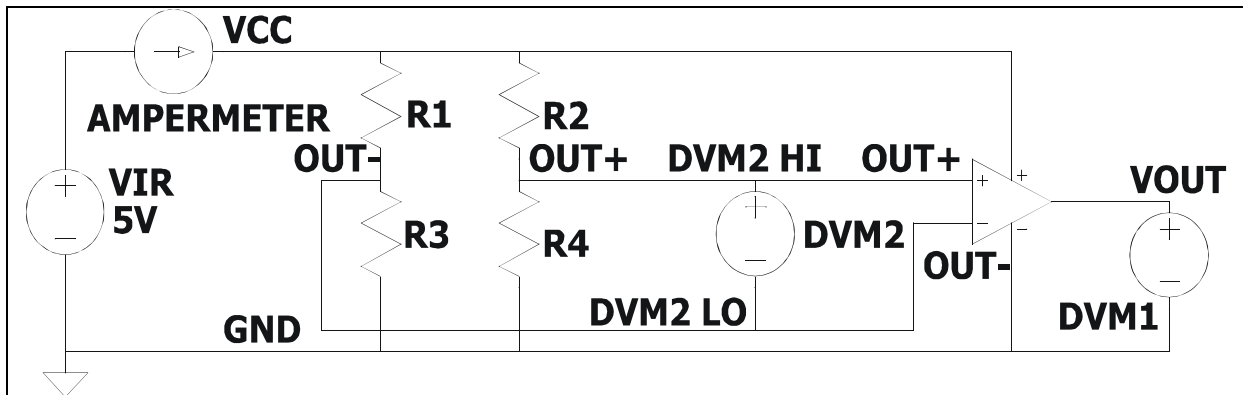


Slika 68: Vmesnik za vezje ojačevalnika senzorskega signala.

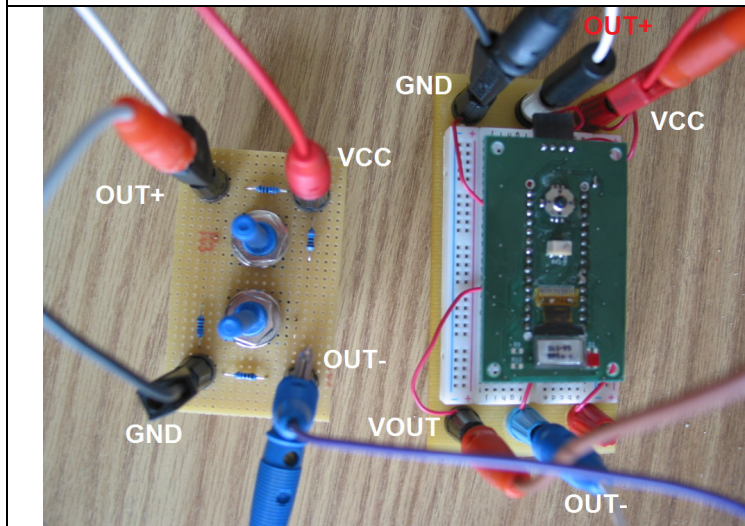


Slika 69: Vezje ojačevalnika senzorskega signala s spodnje strani.

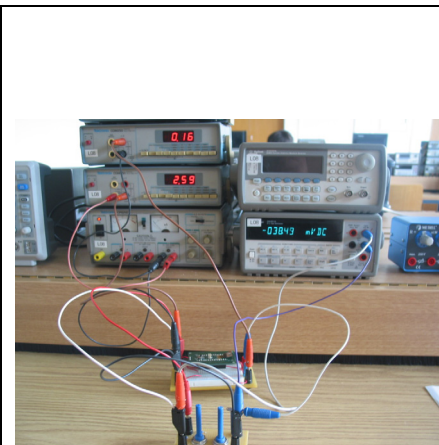
Povežite vezje simulatorja senzorja in vezje senzorskega ojačevalnika po shemi na sliki 70. Pomagajte si s sliko 71, pri čemer imena vozlišč na shemi usklajujte z oznakami pri pušah na sliki 71. Enosmerni vir nastavite na 5 V/60 mA. Voltmeter DVM1 je Tektronix CDM250, DVM2 je Agilent 33401.



Slika 70: Vezalna shema simulatorja senzorja (levo) in senzorskega ojačevalnika (desno).



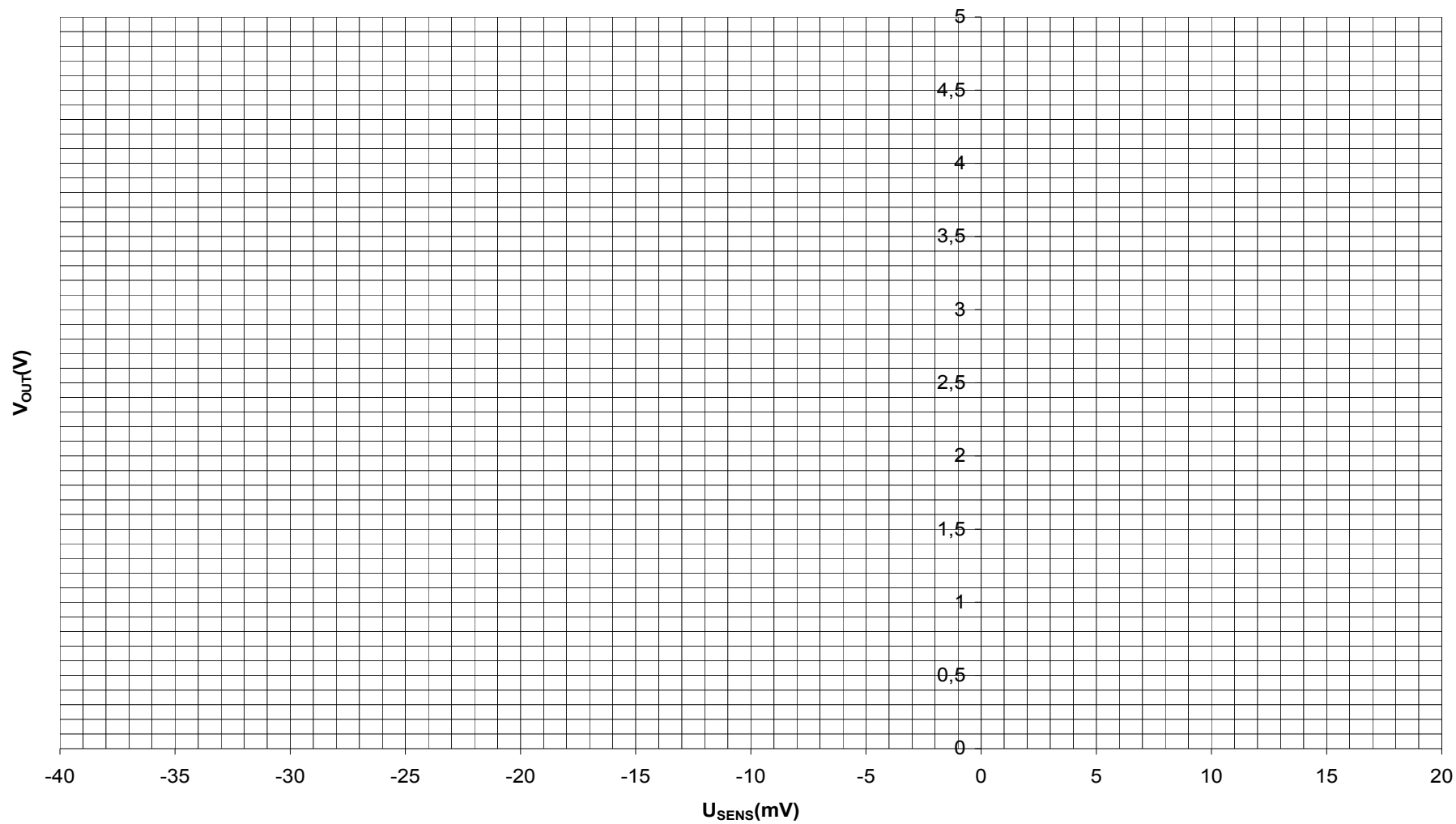
Slika 71: Oznake priključkov pri povezovanju simulatorja senzorja (levo) in senzorskega ojačevalnika (desno).



Slika 72: Priključitev merilnih inštrumentov.

Izhodno napetost simulatorja mostiča nastavljajte od pozitivne skrajnosti do negativne, pri čemer si zapisujete napetost na DVM1 in DVM2. Trimer potenciometer RV1 nastavite tako, da bo ojačevalnik ojačeval signale med izmerjeno ničelno napetostjo senzorja U_{OFFSET} do +20 mV.

$U_{\text{SENS}}[\text{mV}]$	$V_{\text{OUT}}[\text{V}]$
... $-U_{\text{OFFSET}}$	
-30	
-25	
-20	
-15	
-10	
0	
+5	
+10	
+15	
+20	
+25	
+25	

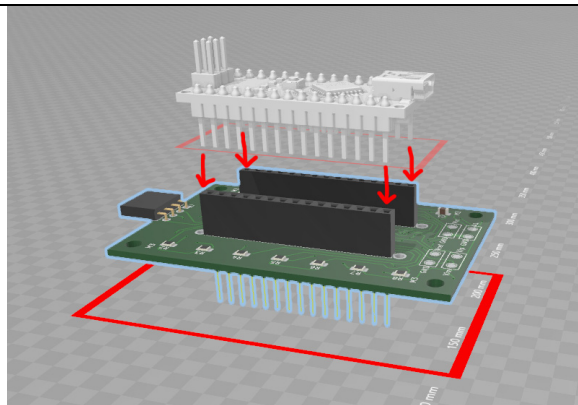


Slika 73: Karakteristika izdelanega senzorskega ojačevalnika.

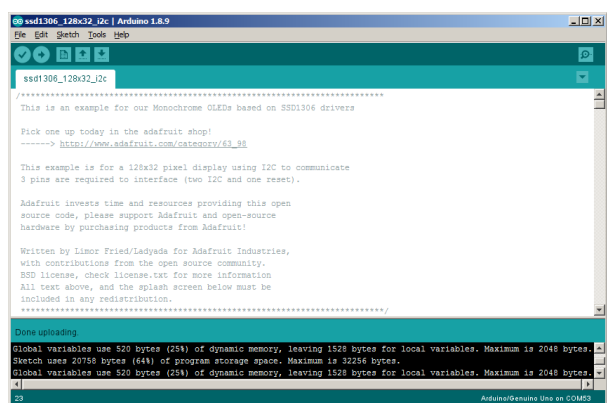
Zajem signala senzorja in digitalna obdelave senzorskega signala v Arduino IDE

Najprej staknite skupaj *preizkušeno* ploščico senzorskega ojačevalnika in Arduino modul, pri čemer pazite da je senzorski priključek na nasprotni strani kot USB priključek Arduino modula. Potem povežite Arduino modul preko USB povezave na osebni računalnik. Ob prvi priključitvi sistem zazna nov gonilnik za FT232R - potrdite namestitev gonilnika. Ustvarijo se serijska vrata (COMxx), preko katerih bo Arduino komuniciral s svojim okoljem (IDE).

Na Namizju (ang. Desktop) poženete delovno okolje Arduino IDE. V nadaljevanju bomo postopoma preverjali delovanje posameznih komponent programa in tako gradili program za zajem vrednosti s senzorja in digitalno obdelavo senzorskega signala.



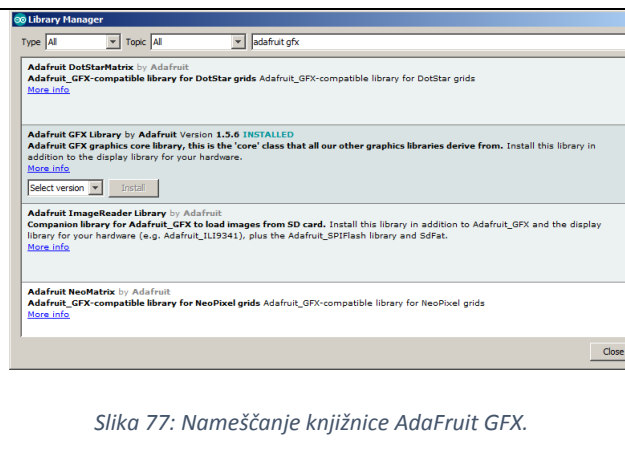
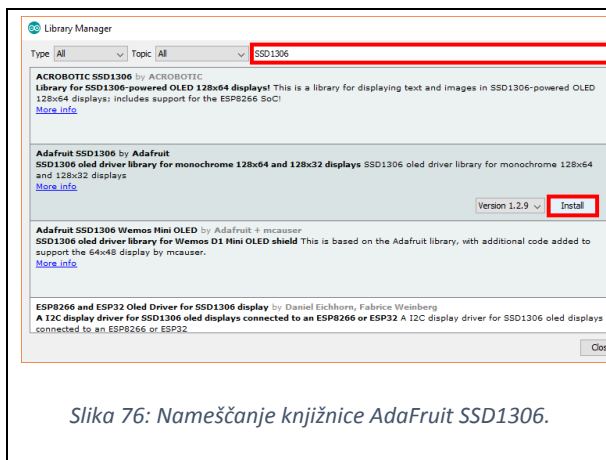
Slika 74: Vstavljanje Arduino modula v senzorski ojačevalnik.



Slika 75: Okolje Arduino IDE.

Preverjanje pravilnosti delovanja OLED zaslona.

Za pravilnost delovanja OLED zaslona bomo v Arduino modul naložili že predpripravljen zgled (ssd1306_128x32_i2c), s katerim bomo na zaslon izrisovali različne grafične objekte. Pred zagonom zglada moramo namestiti potrebne knjižnice za delo z zaslonom SSD1306. Za namestitev knjižnic v meniju izberete Tools→Manage libraries (ali pritisnete CTRL+SHIFT+I) nakar se pojavi pogovorno okno upravitelja knjižnic (ang. library manager). V iskalnik (ang. filter) vpišete "SSD1306" in iz seznama zadetkov izberete "AdaFruit SSD1306 by Adafruit" in knjižnico namestite (Install). Ko je knjižnica nameščena, v iskalnik vpišete "Adafruit GFX" in izberete "Adafruit GFX Library by Adafruit" ter jo namestite (Install).



Ko ste obe knjižnici namestili, v meniju Arduino IDE izberete:



File→Examples→Adafruit SSD1306→ ssd1306_128x32_i2c. Izbira "Adafruit SSD1306" se nahaja čisto na dnu rolete "Examples". Program najprej preverite (ang. verify) s klikom na ikono  v orodni vrstici pod menijem. Ko se preverjanje izvede, bo program v konzoli na dnu okna napisal stanje porabe podatkovnega in spominskega prostora (Tabela 12). Nato kliknete na ikono  nalaganje (ang. upload), ki je sosedna ikoni za preverjanje.

Tabela 12: Pregled porabe podatkovnega in spominskega prostora.

```
Sketch uses 20758 bytes (64%) of program storage space. Maximum is 32256 bytes.

Global variables use 520 bytes (25%) of dynamic memory, leaving 1528 bytes for local variables. Maximum is 2048 bytes.
```

Do zaslona v Arduino IDE dostopamo z objektom Display. Objekt ima lahko določene metode (oz. funkcije), s katerimi izvajamo dejanja kot so npr. brisanje(clearDisplay), določanje velikosti besedila (TextSize), postavljanje izhodišča za pisanje (setCursor) in končno izris na zaslon (display). Vsako funkcijo kličemo kot objekt.ime_funkcije (npr. Display.display).

Tabela 13: Nekaj funkcij za delo z besedilom na zaslonu.

Funkcija	Pomen
println(F(»besedilo«))	Shrani besedilo v spomin zaslona. Funkcija s katero pišemo na zaslon.
clearDisplay()	zbriše spomin zaslona
setTextSize()	določi velikost risane besedila na zaslonu
setTextColor(WHITE):	določi barvo besedila (za večbarvne zaslone) v našem primeru moramo izbrati opcijo white (slov. bela)
setCursor(32,0) :	postavi izhodišče zaslona na določeno pozicijo, v našem primeru 0 32, zaradi načina naslavljanja zaslona

Poskusno izpisovanje besedila na OLED zaslonu.

Zaslon bomo uporabljali za izpis izmerjenih odčitkov senzorja in izpis ostalih sporočil v besedilni obliki. Zato bomo potrebovali funkcijo, ki bo izpisala podano številko na zaslon. Ustvarimo nov primer, tako da izberemo File→New (ali pritisnemo CTRL+N). Na meniju izberemo (File→Save as) "shrani kot" (ang,. save as). Pojavi se pogovorno okno, v katerega vnesemo nov imenik z imenom primera in vnesemo ime primera (npr. "IZPIS_BESEDILA"). Pojavi se prazno okno s predpripravljenima funkcijama setup in loop. Funkcija setup (slov. vzpostavi), se izvede enkrat ob zagonu, funkcija loop (slov. zanka) pa se izvaja ponavljajoče.

Program za izpis besedila bomo izdelali tako, da bo periodično, na vsakih 500 ms, povečal spremenljivko (Output) in jo izpisal v oblikovani vrstici na zaslon. Oblika vrstice bo " $\Delta h=12.34$ cm".

Windows Terminal - 6 point																	
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF	
0x		0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
1x	!	"	#	\$	%	&	'	()	*	+	,	-	.	/		
2x		!	"	#	\$	%	&	'	()	*	+	,	-	.	/	
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_	
6x	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	
7x	p	q	r	s	t	u	v	w	x	y	z	{		}	~		
8x									¡	¢	£	¤	¥	¦	§	¨	©
9x	ª	«	¬	­	®	¯	°	±	²	³	´	µ	¶	·	¸	¹	º
Ax	»	¼	½	¾	¿										¡	¢	£
Bx	¤	¥	¦	§	¨	©	ª	«	¬	­	®	¯	°	±	²	³	´
Cx	µ	¶	·	¸	¹	º	»	¼	½	¾	¿						
Dx				¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	­	®
Ex	¯	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Fx										¡	¢	£	¤	¥	¦	§	¨

Slika 78:Kodna tabela znakov, ki jih podpira zaslon.

Tabela 14: Izpis programa za izpisovanje besedila.

```

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 96 // OLED sirina zaslona v pikslih
#define SCREEN_HEIGHT 32 // OLED visina zaslona v pikslih
#define OLED_RESET 4 // Stevilka reset priključka (ali -1, ce si deli reset
prikljucek z Arduino modulom)

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); //
deklaracija zaslona

float Output = 0.0; // spremenljivka, ki jo izpisujemo

void setup() {
  // simbol SSD1306_SWITCHCAPVCC - zaslon tvori 3.3V napetost sam od sebe
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    for(;;); // Ostani tukaj do ponovnega reseta
  }

  delay(2000); // pocakaj 2 sekundi

  display.clearDisplay(); // brisanje zaslona
  display.setTextSize(1); // nastavi velikost pisave
  display.setTextColor(WHITE); // nastavi barvo pisave

  pinMode(A1,INPUT); // nastavi prikljucek A1 kot vhod
}

```

```
void loop() {
  Output = Output + 0.1; // povecaj spremenljivko
  drawHeight( Output ); // izpisi besedilo
  delay( 500 ); // pocakaj 500 ms
}

void drawHeight ( float h ) {

  char floatString[10]; // hrani ASCII zapis besedila, velikost 5 znakov -
  spremenljivko h v obliki (12.34)
  char print_buf[20]; // hrani ASCII zapis celotne vrstice, ki se izpiše

  dtostrf( h, 0, 2, floatString); // pretvorba spremenljivke h v ASCII zapis -
  rezultat gre v floatString
  sprintf( print_buf, "\x7Fh=%s cm", floatString); // sestavljanje besedila:
  "Δh=" gre pred in "cm" za floatString besedilo gre pred in "cm"
  display.clearDisplay(); // brisanje zaslona
  display.setCursor(32, 20); // postavljanje kurzorja
  display.print(print_buf); // izpis celotnega besedila na zaslon
  display.display(); // prikaz besedila na zaslonu
}
```

Poskusno branje analogno-digitalnega pretvornika in izpis surovega odčitka na OLED zaslonu.

Zaslon bomo uporabljali za izpis izmerjenih odčitkov senzorja in izpis ostalih sporočil v besedilni obliki. Zato bomo potrebovali funkcijo, ki bo izpisovala celoštevilsko vrednost analogno-digitalnega pretvornika na kanalu A1.

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 96 // OLED sirina zaslona v pikslih
#define SCREEN_HEIGHT 32 // OLED visina zaslona v pikslih
#define OLED_RESET 4 // Stevilka reset priključka (ali -1, ce si deli reset
priključek z Arduino modulom)

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); //
deklaracija zaslona
int ADC_readout = 0;

void setup() {
  // simbol SSD1306_SWITCHCAPVCC pomeni, da zaslon tvori 3.3V napetost sam od
  sebe
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    for(;;); // Ostani tukaj do ponovnega reseta
  }

  delay(2000); // počakaj 2 sekundi

  display.clearDisplay(); // brisanje zaslona
  display.setTextSize(1); // nastavi velikost pisave
  display.setTextColor( WHITE ); // nastavi barvo pisave

  pinMode( A1, INPUT ); // nastavi priključek A1 kot vhod
  analogReference( DEFAULT ); // nastavi referencno napetost ADC kot napajalno
  napetost 5 V
}

void loop() {
  analogRead(A1);
  ADC_readout = ADCL + (ADCH *256); // sestavi zgornji in spodnji zlog
  odcitka ADC
  drawADC_readout( ADC_readout ); // izpisi odcitek ADC
  delay(200); // počakaj 200 ms
}

void drawADC_readout( int readout ) {

  char print_buf[20]; // hrani ASCII zapis vrstice, ki se izpiše na zaslon
  sprintf( print_buf, "ADC=%04d", readout); // sestavljanje besedila in
  spremenljivke - zapis na starih mestih (0 do 1023)
  display.clearDisplay(); // brisanje zaslona
  display.setCursor(32, 20); // postavljanje kurzorja
  display.print(print_buf); // izpis celotnega besedila na zaslon
  display.display(); // prikaz besedila na zaslonu
}
```

Branje stanja navigacijskega stikala in izpis na OLED zaslonu.

Stanje navigacijskega stikala (ang. navigation switch, joystick) bomo izpisovali na zaslon. Program razdelimo na dve funkciji - branje stanja (read_NAV_SW_state) in izpis stanja (draw_NAV_SW_state).

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 96 // OLED sirina zaslona v piksljih
#define SCREEN_HEIGHT 32 // OLED visina zaslona v piksljih
#define OLED_RESET 4 // Stevilka reset priključka (ali -1, ce si deli reset
priključek z Arduino modulom)

// polozaji bitov pri branju navigacijskega stikala, ki jih vrne funkcija
read_NAV_SW_state
#define NAV_SW_BIT_A 0
#define NAV_SW_BIT_B 1
#define NAV_SW_BIT_C 2
#define NAV_SW_BIT_D 3
#define NAV_SW_BIT_BTN 4

int nav_sw_pin_a = 6; // navigacijsko stikalo - kontakt a
int nav_sw_pin_b = 5; // navigacijsko stikalo - kontakt b
int nav_sw_pin_c = 3; // navigacijsko stikalo - kontakt c
int nav_sw_pin_d = 4; // navigacijsko stikalo - kontakt d
int nav_sw_pin_btn = 2; // navigacijsko stikalo - gumb

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); //
deklaracija zaslona

void setup() {
  // simbol SSD1306_SWITCHCAPVCC pomeni, da zaslon tvori 3.3V
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    for(;;); // Ostani tukaj do ponovnega reseta
  }
  delay(2000); // počakaj 2 sekundi
  display.clearDisplay(); // brisanje zaslona
  display.setTextSize(1); // nastavi velikost pisave
  display.setTextColor( WHITE ); // nastavi barvo pisave

  // nastavi priključke navigacijskega stikala kot vhodne s pull-up uporom
  pinMode( nav_sw_pin_a, INPUT_PULLUP); pinMode( nav_sw_pin_b, INPUT_PULLUP);
  pinMode( nav_sw_pin_c, INPUT_PULLUP); pinMode( nav_sw_pin_d, INPUT_PULLUP);
  pinMode( nav_sw_pin_btn, INPUT_PULLUP);

  pinMode( A1, INPUT ); // nastavi priključek A1 kot vhod
}

void loop() {
  draw_NAV_SW_state( read_NAV_SW_state()); // preberi in izpisi stanje
navigacijskega stikala
  delay(100); // počakaj 100 ms
}
```

Navigacijskega stikalo je dejansko pet stikal v enem ohišju, s katerimi lahko upravljamo kot z igralno palico (ang. joystick). Na načrtu Arduino modula (sliki 80) je navigacijsko stikalo označeno kot S1. Kontakti stikala (A, B, C, D in BTN) so na Arduino Nano povezani na digitalne priključke 6→A, 5→B, 3→C, 4→D ter 4→BTN.

Funkcija za branje stanja navigacijskega stikala (`read_NAV_SW_state`) bere stanje štirih digitalnih priključkov (sliki 80) in njihovo stanje vrne (ang. return) v eni celoštevilski (ang. integer) spremenljivki. Stikalo ima lahko dve stanji (HIGH, LOW). V funkciji najprej deklariramo interno spremenljivko (`nav_sw_state`), ki hrani stanje stikal, nato pa s funkcijo (`digitalRead`) beremo stanje posameznega priključka in ustrezen bit spremenljivke (`nav_sw_state`) postavimo (`bitSet`) ali brišemo (`bitClear`). Ko končamo, vrnemo (`return`) vrednost spremenljivke (`nav_sw_state`).

```
int read_NAV_SW_state( void ) {
    int nav_sw_state = 0; // ponastavi vse bite na nic

    if ( digitalRead( nav_sw_pin_a ) == HIGH ) {
        bitSet( nav_sw_state, NAV_SW_BIT_A); // postavi bit kontakta A
    }
    else {
        bitClear( nav_sw_state, NAV_SW_BIT_A); // zbrisi bit kontakta A
    }

    if ( digitalRead( nav_sw_pin_b ) == HIGH ) {
        bitSet( nav_sw_state, NAV_SW_BIT_B); // postavi bit kontakta B
    }
    else {
        bitClear( nav_sw_state, NAV_SW_BIT_B); // zbrisi bit kontakta B
    }

    if ( digitalRead( nav_sw_pin_c ) == HIGH ) {
        bitSet( nav_sw_state, NAV_SW_BIT_C); // postavi bit kontakta C
    }
    else {
        bitClear( nav_sw_state, NAV_SW_BIT_C); // zbrisi bit kontakta C
    }

    if ( digitalRead( nav_sw_pin_d ) == HIGH ) {
        bitSet( nav_sw_state, NAV_SW_BIT_D); // postavi bit kontakta D
    }
    else {
        bitClear( nav_sw_state, NAV_SW_BIT_D); // zbrisi bit kontakta D
    }

    if ( digitalRead( nav_sw_pin_btn ) == HIGH ) {
        bitSet( nav_sw_state, NAV_SW_BIT_BTN); // postavi bit stikala
    }
    else {
        bitClear( nav_sw_state, NAV_SW_BIT_BTN); // zbrisi bit stikala
    }

    return ( nav_sw_state );
}
```


Funkcija za izpis stanja navigacijskega stikala bo na vhodu sprejela stanje stikal v celoštevilski spremenljivki (`nav_sw_state`) in na zaslону izrisala smerne znake ($\uparrow, \downarrow, \leftarrow, \rightarrow$), glede na to, katere tipke so pritisnjene. Za izpis bomo potrebovali polje znakov oz. besedilo (ang. string), ki ga deklariramo v spremenljivki (`print_buf`). Z branjem ustreznega bita vhodne spremenljivke (`nav_sw_state`) z operacijo (`bitRead`) bomo ob pritisku na tipko (stanje LOW) v besedilo (`print_buf`) s funkcijo (`strcat`) na konec dodali ustrezen znak smerne puščice. Kode znakov preberemo iz slike 78: Kodo sestavlja 6 bitov, zgornja skupina 4 bitov (ang. nibble) je v stolpcu, spodnja v vrstici slike 78. Prebrana koda znaka \uparrow je 18_{16} . Zapis šestnajstiškega znaka s kodo 18_{16} izvedemo s formatnim določilom ("`\x18`"). Ko preberemo stanja vseh stikal, izpišemo besedilo.

```
void draw_NAV_SW_state( int nav_sw_state ) {

    char print_buf[6] = {0};    // hrani ASCII zapis vrstice, ki se izpiše na zaslon

    // navigacijsko stikalo daje kontakt na maso - ce stikalo ni pritisnjeno, je
    bit stanja HIGH

    if ( bitRead( nav_sw_state, NAV_SW_BIT_A ) == LOW ) {
        strcat( print_buf, "\x1B");    // dodaj kodo znaka "levo" kot indikator
        pritiska na kontakt A navigacijskega stikala
    }

    if ( bitRead( nav_sw_state, NAV_SW_BIT_B ) == LOW ) {
        strcat( print_buf, "\x18");    // dodaj kodo znaka "gor" kot indikator
        pritiska na kontakt B navigacijskega stikala
    }

    if ( bitRead( nav_sw_state, NAV_SW_BIT_C ) == LOW ) {
        strcat( print_buf, "\x19");    // dodaj kodo znaka "dol" kot indikator
        pritiska na kontakt C navigacijskega stikala
    }

    if ( bitRead( nav_sw_state, NAV_SW_BIT_D ) == LOW ) {
        strcat( print_buf, "\x1A");    // dodaj kodo znaka "desno" kot indikator
        pritiska na kontakt D navigacijskega stikala
    }

    if ( bitRead( nav_sw_state, NAV_SW_BIT_BTN ) == LOW ) {
        strcat( print_buf, "\x14");    // dodaj kodo znaka "ENTER" kot indikator
        pritiska na stikalo navigacijskega stikala
    }

    display.clearDisplay();          // brisanje zaslona
    display.setCursor(32, 10);      // postavljanje kurzorja
    display.print(print_buf);        // izpis celotnega besedila na zaslon
    display.display();              // prikaz besedila na zaslonu
}
```

Enostaven preračun surovega odčitka A/D pretvornika v višino in izpis na OLED zaslonu.

Najbolj enostaven preračun odčitka A/D pretvornika izvedemo z ovrednotenjem naklonskega koeficienta premice in presečišča z 0, ki povezuje dve točki senzorske karakteristike. Iz prejšnjih meritev surovega A/D pretvornika že imamo odčitka pri višinski razliki $\Delta h = 0$ cm in $\Delta h = 10$ cm. Če prvo točko umerjanja označimo kot CP1 (ang. calibration point 1), lahko zapišemo CP1= (ADC1, 0 cm) in za drugo točko CP2 (ADC2, 10 cm). Če bi bila izračunana višina celoštevilskega tipa, bi lahko uporabili vgrajeno funkcijo (`map`). Izračunana višina (h) je realna vrednost (ang. float), zato moramo napisati svojo funkcijo (`mapFloat`), ki izračuna naklonski koeficient premice in presečišče z 0. Točko umerjanja zapišemo s surovim odčitkom A/D pretvornika (`CPx_ADC`) in ustrezno višino (`CP1_H`). Za izpis izračunane višine v spremenljivki (h) uporabimo funkcijo (`drawHeight`), ki smo jo programirali v enem prejšnjih zgledov.

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 96 // OLED sirina zaslona v piksljih
#define SCREEN_HEIGHT 32 // OLED visina zaslona v piksljih
#define OLED_RESET 4 // Stevilka reset priključka (ali -1, ce si deli reset
priključek z Arduino modulom)

#define CP1_ADC 123
#define CP1_H 0.0

#define CP2_ADC 890
#define CP2_H 10.0

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); //
deklaracija zaslona
int ADC_readout = 0; // surov ADC odcitek
float h = 0.0; // izracunana visina vodnega stolpca

void setup() {

  // simbol SSD1306_SWITCHCAPVCC pomeni, da zaslon tvori 3.3V napetost sam od
sebe
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    for(;;); // Ostani tukaj do ponovnega reseta
  }

  delay(2000); // pocakaj 2 sekundi

  display.clearDisplay(); // brisanje zaslona
  display.setTextSize(1); // nastavi velikost pisave
  display.setTextColor( WHITE ); // nastavi barvo pisave

  pinMode( A1, INPUT ); // nastavi priključek A1 kot vhod
  analogReference( DEFAULT ); // nastavi referencno napetost 5 V
}
```

```

void loop() {
  analogRead(A1);
  ADC_readout = ADCL + (ADCH *256);    // sestavi zg. in sp. zlog odcitka ADC
  h = mapFloat( ADC_readout, CP1_ADC, CP2_ADC, CP1_H, CP2_H ); // mapiranje
  odcitka
  drawHeight( h ); // pretvori surov odcitek na ADC in izpisi visino
  delay(200);      // pocakaj 200 ms
}

void drawHeight( float h ) {

  char floatString[10]; // hrani ASCII zapis besedila, velikost 5 znakov -
  spremenljivko h v obliki (12.34)
  char print_buf[20]; // hrani ASCII zapis celotne vrstice, ki se izpiÅje na
  zaslon

  dtostrf( h, 0, 2, floatString); // pretvorba spremenljivke h v ASCII zapis -
  rezultat gre v floatString
  sprintf( print_buf, "\x7Fh=%s cm", floatString); // sestavljanje besedila:
  "h=" gre pred in "cm" za floatString besedilo
  display.clearDisplay(); // brisanje zaslona
  display.setCursor(32, 20); // postavljanje kurzorja
  display.print(print_buf); // izpis celotnega besedila na zaslon
  display.display(); // prikaz besedila na zaslonu
}

float mapFloat( int x, int in_min, int in_max, float out_min, float out_max)
{
  return (float)(x - in_min) * (out_max - out_min) / (float)(in_max - in_min) +
  out_min;
}

```

Segmentiran preračun surovega odčitka A/D pretvornika v višino in izpis na OLED zaslonu.

V prejšnjem primeru smo izvedli najbolj enostaven preračun odčitka A/D pretvornika s pomočjo linearne aproksimacije karakteristike senzorja. Tovrstni izračun je običajno hiter, vendar ni dovolj natančen, če je senzorska karakteristika nelinearna oz. ukrivljena.

Za izničenje oz. kompenzacijo nelinearnosti, karakteristiko razdelimo na več delov ali segmentov, znotraj posameznega pa izračunavamo linearno interpolacijo (PWL - piecewise linear approximation). Linearno aproksimacijo in določitev segmenta, na katerem se nahaja trenutni odčitek A/D izdelamo s funkcijo (`segmented_map`).

Tabela 15: Segmentirani preračun surovega odčitka A/D pretvornika v višino izračun višine.

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 96 // OLED sirina zaslona v pikslih
#define SCREEN_HEIGHT 32 // OLED visina zaslona v pikslih
#define OLED_RESET 4 // Stevilka reset priključka (ali -1, ce si deli reset
prikljucjek z Arduino modulom)

#define CP1_ADC 123
#define CP1_H 0.0

#define CP2_ADC 220
#define CP2_H 10.0

#define CP3_ADC 320
#define CP3_H 20.0

#define CP4_ADC 450
#define CP4_H 30.0

#define CP5_ADC 520
#define CP5_H 40.0

#define CP6_ADC 630
#define CP6_H 50.0

#define CP7_ADC 720
#define CP7_H 60.0

#define CP8_ADC 810
#define CP8_H 70.0

#define CP9_ADC 930
#define CP9_H 80.0

#define CP10_ADC 1000
#define CP10_H 90.0

#define CP11_ADC 1020
#define CP11_H 100.0
```

```

#define NR_CAL_PTS    sizeof(cp_in)/sizeof(cp_in[0])
// stevilo tock umerjanja (11 v tem primeru)

int  cp_in[]  = { CP1_ADC, CP2_ADC, CP3_ADC, CP4_ADC, CP5_ADC, CP6_ADC,
CP7_ADC, CP8_ADC, CP9_ADC, CP10_ADC, CP11_ADC};
float cp_out[] = { CP1_H, CP2_H, CP3_H, CP4_H, CP5_H, CP6_H, CP7_H, CP8_H,
CP9_H, CP10_H, CP11_H};

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); //
deklaracija zaslona
int  ADC_readout = 0;    // surov ADC odcitek
float h = 0.0;          // izracunana visina vodnega stolpca

void setup() {

    // simbol SSD1306_SWITCHCAPVCC pomeni, da zaslon tvori 3.3V napetost sam od
sebe
    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        for(;;); // Ostani tukaj do ponovnega reseta
    }

    delay(2000); // pocakaj 2 sekundi

    display.clearDisplay();    // brisanje zaslona
    display.setTextSize(1);    // nastavi velikost pisave
    display.setTextColor( WHITE ); // nastavi barvo pisave

    pinMode( A1, INPUT );    // nastavi prikljucek A1 kot vhod
    analogReference( DEFAULT ); // nastavi referencno napetost ADC kot napajalno
napetost 5 V
}

void loop() {
    analogRead(A1);
    ADC_readout = ADCL + (ADCH *256);    // sestavi zgornji in spodnji zlog
odcitka ADC
    h = segmented_map( ADC_readout, cp_in, cp_out, NR_CAL_PTS );
// segmentiran izracun visine iz odcitka ADC
    drawHeight( h ); // pretvori surov odcitek na ADC in izpisi visino
    delay( 200 );    // pocakaj 200 ms
}

// note: polje _in mora biti urejeno v narascajocem redu ADC odcitkov
int segmented_map(int val, int* _in, float* _out, const uint8_t size)
{
    if (val <= _in[0]) return _out[0];    // pod sp. mejo intervala umerjanja
    if (val >= _in[size-1]) return _out[size-1]; // nad zg. mejo intervala
umerjanja

    // val je znotraj intervala umerjanja - poisci ustrezen segment
    uint8_t pos = 1;    // _in[0] smo ze preverili
    while(val > _in[pos]) pos++;

    if (val == _in[pos]) return _out[pos]; // ce je tocka val enaka spodnji meji
segmenta
    // linearna interpolacija znotraj ustreznega segmenta
    return (val - _in[pos-1]) * (_out[pos] - _out[pos-1]) / (_in[pos] - _in[pos-
1]) + _out[pos-1];
}

```

```
void drawHeight( float h ) {  
  
    char floatString[10]; // hrani ASCII zapis besedila, velikost 5 znakov -  
    spremenljivko h v obliki (12.34)  
    char print_buf[20]; // hrani ASCII zapis celotne vrstice, ki se izpiše na  
    zaslon  
  
    dtostrf( h, 0, 2, floatString); // pretvorba spremenljivke h v ASCII zapis -  
    rezultat gre v floatString  
    sprintf( print_buf, "\x7Fh=%s cm", floatString); // sestavljanje besedila:  
    "h=" gre pred in "cm" za floatString besedilo  
    display.clearDisplay(); // brisanje zaslona  
    display.setCursor(32, 20); // postavljanje kurzorja  
    display.print(print_buf); // izpis celotnega besedila na zaslon  
    display.display(); // prikaz besedila na zaslonu  
}
```

Filtriranje surovega odčitka A/D pretvornika s tekočim povprečenjem (ang. moving average)

Če podatki s sensorja precej spreminjajo (rečemo, da izhod sensorja šumi), potem moramo odčitke sensorja povprečiti. Navadno aritmetično povprečenje bi zahtevalo zajem nekaj (npr. 16) vzorcev, izračun povprečja zajetih vzorcev, deljenje s številom elementov in izpis tega povprečja. Hitrost izpisa se ob tovrstnem filtriranju upočasni (npr. 16x), saj moramo zajeti vse vzorce, preden lahko izračunamo povprečje in ga izpišemo. Hitrost izpisa ohranimo *enako* hitrosti zajemanja vzorcev (vzorčenja), če namesto navadnega povprečenja naredimo tekoče povprečenje (ang. moving average). Tak filter deluje na osnovi *polja vzorcev* (MAVG_Buffer), v katerega vedno vstavljamo nov odčitek (ADC_readout) na naslednje prosto mesto v polju. Ob tem, ko vzorec vstavimo, indeks prostega mesta (NextIdx) povečamo in izračunamo povprečje trenutne vsebine polja (MAverage). Velikost polja, ki opravlja tekoče povprečenje, je (MAVG_SIZE). Smiselno je postaviti velikost (MAVG_SIZE) na vrednost potence števila 2, torej 2ⁿ, (2, 4, 8, 16, 32, 64, 128 ...). Povprečje namreč na koncu izračunamo z operacijo deljenja (`return (MAverage / MAVG_SIZE);`). Ta operacija je za vrednosti, ki niso potenca števila 2, precej časovno zahtevna operacija. Za vrednosti, ki so potence števila 2, pa to preprosto pomeni aritmetično pomikanje za n mest desno, kar Arduino opravi v eni operaciji aritmetičnega pomika desno ASR (ang. arithmetic shift right).

Tabela 16: Filtriranje surovega odčitka A/D pretvornika s tekočim povprečenjem.

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 96 // OLED sirina zaslona v pikslih
#define SCREEN_HEIGHT 32 // OLED visina zaslona v pikslih
#define OLED_RESET 4 // Stevilka reset priključka (ali -1, ce si deli reset
priključek z Arduino modulom)

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); //
deklaracija zaslona
int ADC_readout = 0;
#define MAVG_SIZE 16 // velikost filtra naj bo 2^n, ker je deljenje s tako
vrednostjo enako pomikanju desno
int MAVG_Buffer[MAVG_SIZE];
int NextIdx;

void setup() {
  // simbol SSD1306_SWITCHCAPVCC pomeni, da zaslon sam tvori 3.3V
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    for(;;); // Ostani tukaj do ponovnega reseta
  }
  delay(2000); // počakaj 2 sekundi
  display.clearDisplay(); // brisanje zaslona
  display.setTextSize(1); // nastavi velikost pisave
  display.setTextColor( WHITE ); // nastavi barvo pisave
  pinMode( A1, INPUT ); // nastavi priključek A1 kot vhod
  analogReference( DEFAULT ); // nastavi referencno napetost ADC na 5 V
}
```

```

void loop() {
  analogRead(A1);
  ADC_readout = ADCL + (ADCH * 256);    // sestavi zgornji in spodnji zlog
  odčitka ADC
  drawADC_readout( MAVG_Filter( ADC_readout ) );    // filtriraj surov odcitek
  ADC in izpisi besedilo
  delay(200);                            // počakaj 200 ms
}

void drawADC_readout( int readout ) {

  char print_buf[20]; // hrani ASCII zapis vrstice, ki se izpiše na zaslon
  sprintf( print_buf, "ADC=%04d", readout); // sestavljanje besedila in
  spremenljivke - zapis na starih mestih (0 do 1023)
  display.clearDisplay(); // brisanje zaslona
  display.setCursor(32, 20); // postavljanje kurzorja
  display.print(print_buf); // izpis celotnega besedila na zaslon
  display.display(); // prikaz besedila na zaslonu
}

int MAVG_Filter( int ADC_readout ) {
  long MAverage = 0;

  MAVG_Buffer[ NextIdx++ ] = ADC_readout; // vstavi element

  if (NextIdx >= MAVG_SIZE) {
    NextIdx = 0; // prisli smo do konca filtra, postavi se nazaj
  }

  for( int i = 0; i < MAVG_SIZE; ++i ) {
    MAverage += MAVG_Buffer[i]; // izračunaj povprečje trenutne vsebine filtra
  }

  return ( MAverage / MAVG_SIZE );
}

```


Filtriranje surovega odčitka A/D pretvornika s eksponentnim povprečenjem

Filter s tekočim povprečjem je tako imenovane FIR izvedbe (ang. finite impulse response). Za te filtre je značilna zagotovljena stabilnost (ne oscilirajo sami po sebi). Ti filtri so odvisni samo od zgodovine vhoda x_k (v našem primeru - odčitkov A/D pretvornika) Po drugi strani pa zahtevajo veliko spomina, saj z rastočim številom elementov polja odčitkov x_k dobimo bolj stabilen odčitek. Alternativa FIR izvedbam filtrov so IIR izvedbe (ang. infinite impulse response), s katerim zaobidemo pomnenje dolge zgodovine vhoda (x_k). Namesto tega shranimo zgodovino izhoda filtra y_k (v našem primeru prejšnjo višino stolpca). Dobimo t.im. rekurzivno izvedbo, ki rada sama zaoscilira! Značilen predstavnik IIR filtrov, ki se uporablja za "glajenje" vhoda je eksponentni filter (ang. exponential filter), pri katerem se izhod filtra y_k izračuna kot funkcija vhoda x_k in prejšnjega izhoda y_{k-1} .

$$y_k = w \cdot x_k + (1-w) \cdot y_{k-1}$$

Parameter w (ang. weight) se nastavlja med 0 in 1 (v Arduino izvedbi med 0 in 100%). Utež w določa (procentualno) razmerje med svežim vhodnim podatkom $w \cdot x_k$ in preteklo zgodovino $(1-w) \cdot y_{k-1}$. Višja ko je utež, večjo težo imajo sveži vhodni podatki in manj zgodovina. Če se vhodni podatki spreminjajo hitro, filter ob visoki vrednosti uteži ne bo dušil hitrih sprememb. Možno je nastavljati tudi začetno vrednost izhoda y_0 . Arduino že ima knjižnico za delo s eksponentnim filtrom (ExponentialFilter). Namestite jo tako, da v meniju Arduino IDE izberite Tools→Manage Libraries, ter v iskalno okno vpišete besedo "filter". Pomaknite se na zadetek "MegunoLink by NumberEight" ter ga namestite (Install). Filter uporabljamo tako, da vključimo definicijo protipov funkcij (`#include "Filter.h"`), ga ustvarimo (`ExponentialFilter<long> expFilter(20, 0);`), vhodni podatek filtriramo (`expFilter.Filter(ADC_readout);`) in preberemo trenutni (ang. current) izhod filtra (`expFilter.Current()`).

```
#include <SPI.h>
#include <Wire.h>

// Preden uporabite ta projekt, morate namestiti knjižnico Filter
// https://www.megunolink.com/documentation/arduino-libraries/exponential-filter/
// V meniju izberite Tools->Manage Libraries
// V iskalno okno vpišite besedo "filter"
// Pomaknite se na zadetek MegunoLink by NumberEight

#include "Filter.h"
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 96 // OLED sirina zaslona v pikslih
#define SCREEN_HEIGHT 32 // OLED visina zaslona v pikslih
#define OLED_RESET 4 // Stevilka reset priključka (ali -1, ce si deli reset
prikljucek z Arduino modulom)

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); //
deklaracija zaslona
int ADC_readout = 0;
```

```

// Ustvari nov eksponencialni IIR filter z utezjo 20 in zacetno vrednostjo 0
ExponentialFilter<long> expFilter(20, 0);

void setup() {

    // simbol SSD1306_SWITCHCAPVCC pomeni, da zaslon tvori 3.3V napetost sam od
    sebe
    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        for(;;); // Ostani tukaj do ponovnega reseta
    }

    delay(2000); // pocakaj 2 sekundi

    display.clearDisplay(); // brisanje zaslona
    display.setTextSize(1); // nastavi velikost pisave
    display.setTextColor( WHITE ); // nastavi barvo pisave

    pinMode( A1, INPUT ); // nastavi prikljucek A1 kot vhod
    analogReference( DEFAULT ); // nastavi referencno napetost ADC 5 V
}

void loop() {
    analogRead(A1);
    ADC_readout = ADCL + (ADCH *256); // sestavi zgornji in spodnji zlog
    odcitka ADC
    expFilter.Filter( ADC_readout ); // eksponentno filtriraj surov odcitek
    ADC
    drawADC_readout( expFilter.Current() ); // izpisi vrednost eksponentno
    filtriranega surovega odcitka ADC
    delay(200); // pocakaj 200 ms
}

void drawADC_readout( int readout ) {

    char print_buf[20]; // hrani ASCII zapis vrstice, ki se izpiše na zaslon
    sprintf( print_buf, "ADC=%04d", readout); // sestavljanje besedila in
    spremenljivke - zapis na stirih mestih (0 do 1023)
    display.clearDisplay(); // brisanje zaslona
    display.setCursor(32, 20); // postavljanje kurzorja
    display.print(print_buf); // izpis celotnega besedila na zaslon
    display.display(); // prikaz besedila na zaslonu
}

```

Serijski komunikacijski vmesnik za povezavo z osebnim računalnikom

Arduino moduli vsebujejo UART enoto - serijski asinhroni komunikacijski vmesnik, s katerim lahko pošljamo podatke na osebni računalnik. Za ta namen najprej komunikacijo omogočimo (`Serial.begin(9600);`) s čimer nastavimo hitrost komunikacijskih vrat na 9600 bps. Podatek (`data`) nato pošljemo z ukazom (`Serial.println(data);`). Če želimo podatke risati v enostaven graf za tem ukazom dodamo še ločilo (ang. delimiter) - najbolj enostavno ločilo je presledek (`Serial.print(" ");`). Za izris uporabimo orodje SerialPlotter (v meniju izberete Tools→Serial plotter).

```
#include <SPI.h>
#include <Wire.h>

// Preden uporabite ta projekt, morate namestiti knjižnico Filter
// https://www.megunolink.com/documentation/arduino-libraries/exponential-filter/
// V meniju izberite Tools->Manage Libraries
// V iskalno okno vpišite besedo "filter"
// Pomaknite se na zadetek MegunoLink by NumberEight
#include "Filter.h"
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 96 // OLED sirina zaslona v pikslih
#define SCREEN_HEIGHT 32 // OLED visina zaslona v pikslih
#define OLED_RESET 4 // Stevilka reset priključka (ali -1, ce si deli reset
prikljucek z Arduino modulom)

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); //
deklaracija zaslona
int ADC_readout = 0;

// Ustvari nov eksponencialni IIR filter z utezjo 20 in zacetno vrednostjo 0
ExponentialFilter<long> expFilter(20, 0);

void setup() {

  // simbol SSD1306_SWITCHCAPVCC pomeni, da zaslon tvori 3.3V napetost sam od
sebe
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    for(;;); // Ostani tukaj do ponovnega reseta
  }

  Serial.begin(9600); // odpri serijska komunikacijska vrata

  delay(2000); // pocakaj 2 sekundi

  display.clearDisplay(); // brisanje zaslona
  display.setTextSize(1); // nastavi velikost pisave
  display.setTextColor( WHITE ); // nastavi barvo pisave

  pinMode( A1, INPUT ); // nastavi prikljucek A1 kot vhod
  analogReference( DEFAULT ); // nastavi referencno napetost 5 V
}
```

```

void loop() {

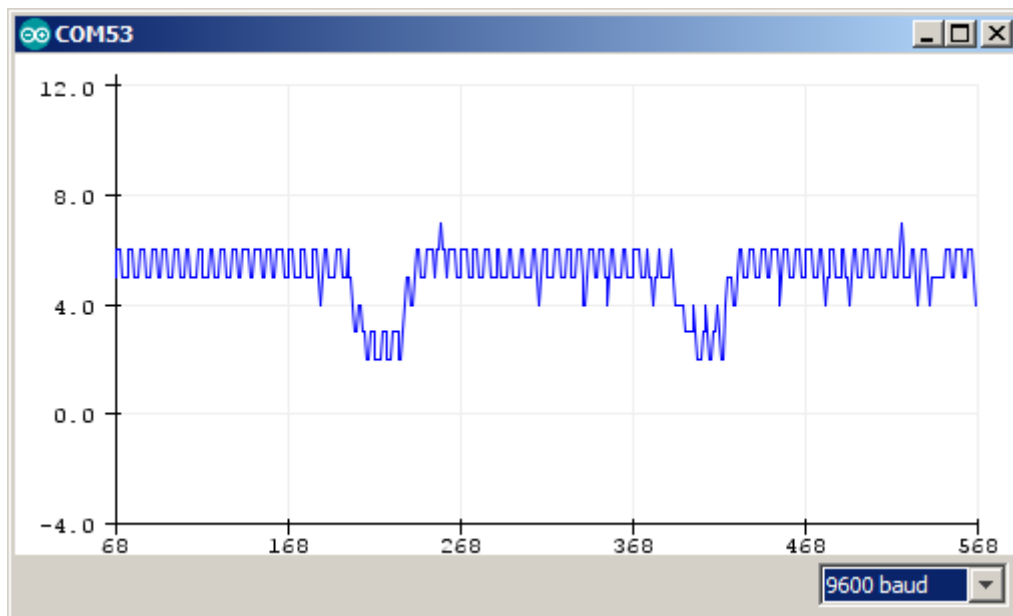
    analogRead(A1);
    ADC_readout = ADCL + (ADCH *256);    // sestavi zgornji in spodnji zlog
    odčitka ADC
    expFilter.Filter( ADC_readout );    // eksponentno filtriraj surov odcitek
    ADC
    drawADC_readout( expFilter.Current() ); // izpisi vrednost eksponentno
    filtriranega surovega odcitka ADC
    Serial.println( expFilter.Current() ); // poslji podatke preko UART na
    SerialPlotter
    Serial.print(" ");

    delay(20);                            // počakaj 20 ms
}

void drawADC_readout( int readout ) {

    char print_buf[20]; // hrani ASCII zapis vrstice, ki se izpiše na zaslon
    sprintf( print_buf, "ADC=%04d", readout); // sestavljanje besedila in
    spremenljivke - zapis na starih mestih (0 do 1023)
    display.clearDisplay(); // brisanje zaslona
    display.setCursor(32, 20); // postavljanje kurzorja
    display.print(print_buf); // izpis celotnega besedila na zaslon
    display.display(); // prikaz besedila na zaslonu
}

```



Slika 79: Izpis odčitka surovega senzora na programu "Serial plotter".

Program za umerjanje in izpis obdelanega podatka s senzorja tlaka na zaslonu in UART

```
#include <SPI.h>
#include <Wire.h>
#include <EEPROM.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET      7 // Reset pin # (or -1 if sharing Arduino reset pin)

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#define WINDOW_SIZE_EXP      10 // Exponent for size of moving window
#define WINDOW_SIZE          0x01 << WINDOW_SIZE_EXP // Size of moving average
window
#define SAMPLES_PER_AVERAGE_EXP 8 // Exponent for samples per average
#define SAMPLES_PER_AVERAGE  0x01 << SAMPLES_PER_AVERAGE_EXP // No. of samples
per average
#define NUM_OF_AVERAGES      0x01 << (WINDOW_SIZE_EXP-SAMPLES_PER_AVERAGE_EXP)
//No. of partial averages

int MovMean = 0;
float Output = 0;

long Window[NUM_OF_AVERAGES];
byte WindowPtr = 0;

String inputString = "";
String outputString = "";
char charStr[20];
bool stringComplete = false;

int cmd = 0;

float sum_x = 1;
float sum_y = 0;
float sum_x2 = 0;
float sum_y2 = 0;
float sum_xy = 0;
float k = EEPROM.read(0);
float n = EEPROM.read(1);
int num = 0;

bool LEDs[] = {0, 0, 0, 0};
float y = 0;
float x = 0;

void setup() {
  delay(100);
  for (int i = 2; i < 7; i++) {
    pinMode(i, INPUT_PULLUP);
  }
  pinMode(13, OUTPUT);
```

```

pinMode(A1, INPUT);

Serial.begin(9600);
if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);
}
}

void loop() {
    if (!digitalRead(2)) {
        for (int i = 3; i < 7; i++) {
            LEDs[i - 3] = 1;
        }
    } else {
        for (int i = 3; i < 7; i++) {
            if (digitalRead(i)) {
                LEDs[i - 3] = 0;
            }
            else {
                LEDs[i - 3] = 1;
            }
        }
    }
}

WindowPtr = ((++WindowPtr) % (NUM_OF_AVERAGES));
Window[WindowPtr] = 0;
for (int i = 0; i < SAMPLES_PER_AVERAGE; i++) {
    Window[WindowPtr] += analogRead(A1);
}

MovMean = Avg(Window, WINDOW_SIZE_EXP, SAMPLES_PER_AVERAGE_EXP);
Output = (MovMean - n) / k;
drawtext(Output);
delay(1);
if (Serial.available() > 0) {
    inputString = Serial.readString();
    inputString.toCharArray(charStr, 20);
    if (charStr[0] == 'y') {
        k = (num * sum_xy - sum_x * sum_y) / (num * sum_x2 - sum_x * sum_x);
        n = (sum_x2 * sum_y - sum_x * sum_xy) / (num * sum_x2 - sum_x * sum_x);
        EEPROM.write(0, k);
        EEPROM.write(1, n);
        Serial.println("Umerjen!");
        goto JMP;
    }
    else if ( charStr[0] == 'r' ) {
        sum_x = 0;
        sum_y = 0;
        sum_x2 = 0;
        sum_y2 = 0;
        sum_xy = 0;
        k = 1;
        n = 0;
        num = 0;
        Serial.println("Ponastavljen!");
        goto JMP;
    }
    x = (int)(inputString.toInt());
    Serial.print("x=");
}

```

```

Serial.println(x);
y = MovMean;
sum_x += x;    sum_y += y;    sum_x2 += x * x;    sum_y2 += y * y;
sum_xy = sum_xy + x * y;    num++;
LinReg(sum_x, sum_y, sum_x2, sum_y2, sum_xy, num, MovMean);
JMP:
    inputString = "";
    while (Serial.available()) {
        Serial.read();
    }
}
}
//}
void LinReg(float sum_x, float sum_y, float sum_x2, float sum_y2, float sum_xy,
int num, float input) {
    float k = (num * sum_xy - sum_x * sum_y) / (num * sum_x2 - sum_x * sum_x);
    float n = (sum_x2 * sum_y - sum_x * sum_xy) / (num * sum_x2 - sum_x * sum_x);
    char floatString[10];
    dtostrf(k, 0, 2, floatString);
    Serial.print("k=");
    Serial.println(floatString);
    Serial.print("n=");
    dtostrf(n, 0, 2, floatString);
    Serial.println(floatString);
    Serial.print("y=");
    dtostrf(input, 0, 2, floatString);
    Serial.println(floatString);
    Serial.println("//-----");
    delay(1);
}

float Avg(long Array[], int WinExp, int SampExp) {
    long Avg = 0;
    for (int i = 0; i < 0x01 << (WinExp - SampExp); i++) {
        Avg += Array[i];
    }
    Avg >>= WinExp;
    return Avg;
}

void drawtext(float depth) {
    delay(1);
    display.clearDisplay();
    char floatString[20];
    display.clearDisplay();
    LED0(LEDs[0]);
    LED1(LEDs[1]);
    LED2(LEDs[2]);
    LED3(LEDs[3]);
    dtostrf(depth, 0, 2, floatString);
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(32, 0);
    display.println(F("\x7Fh:"));
    display.setCursor(32, 10);
    display.print(floatString);
    display.print(F(" cm"));
    display.display();
}

```

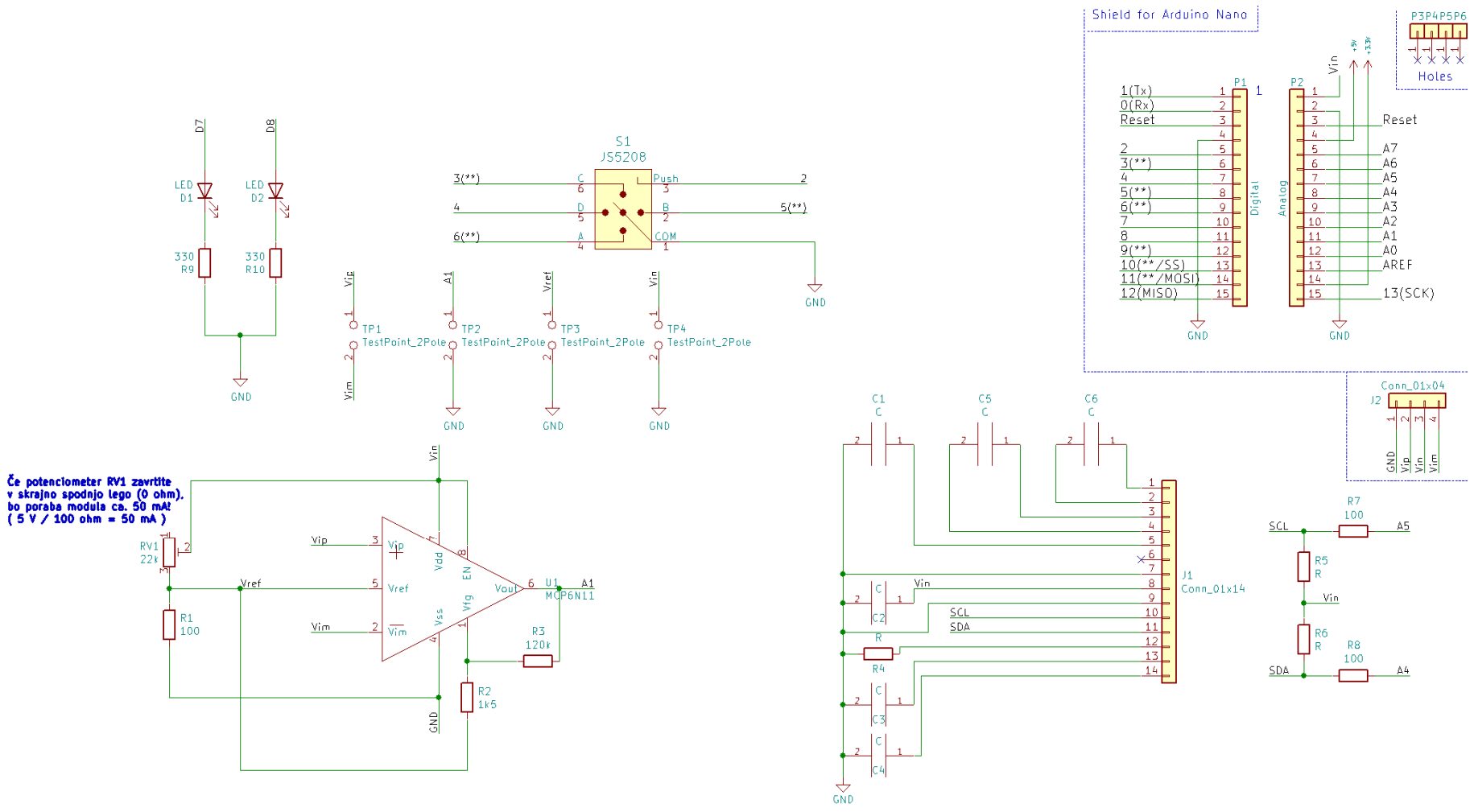
```
    delay(1);
}

void LED0(bool state) {
    delay(1);
    if (state) {
        display.fillRect(32, 20, 10, 10, WHITE);
    }
    else {
        display.fillRect(32, 20, 10, 10, BLACK);
    }
    delay(1);
}

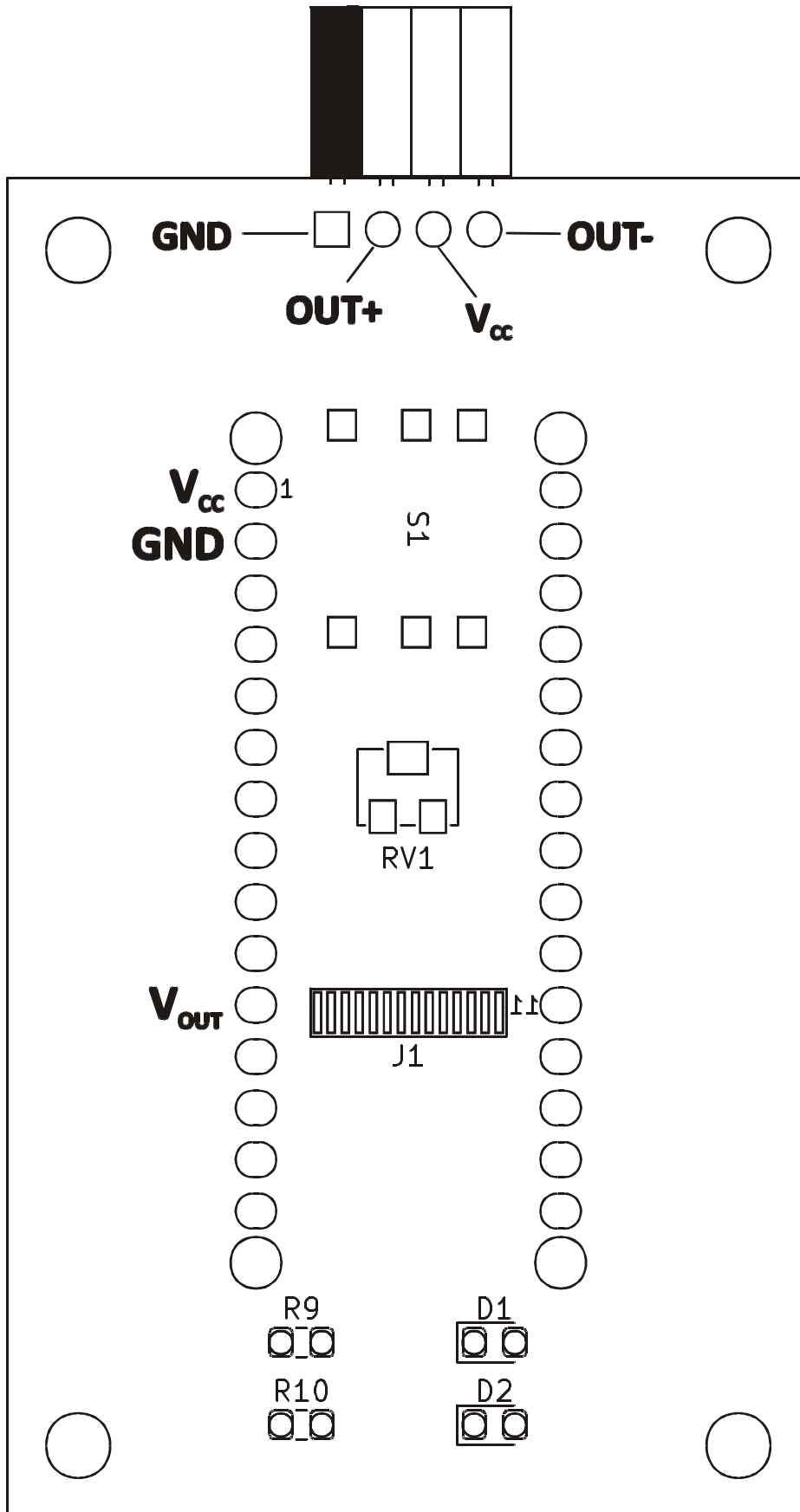
void LED1(bool state) {
    delay(1);
    if (state) {
        display.fillRect(48, 20, 10, 10, WHITE);
    }
    else {
        display.fillRect(48, 20, 10, 10, BLACK);
    }
    delay(1);
}

void LED2(bool state) {
    delay(1);
    if (state) {
        display.fillRect(64, 20, 10, 10, WHITE);
    }
    else {
        display.fillRect(64, 20, 10, 10, BLACK);
    }
    delay(1);
}

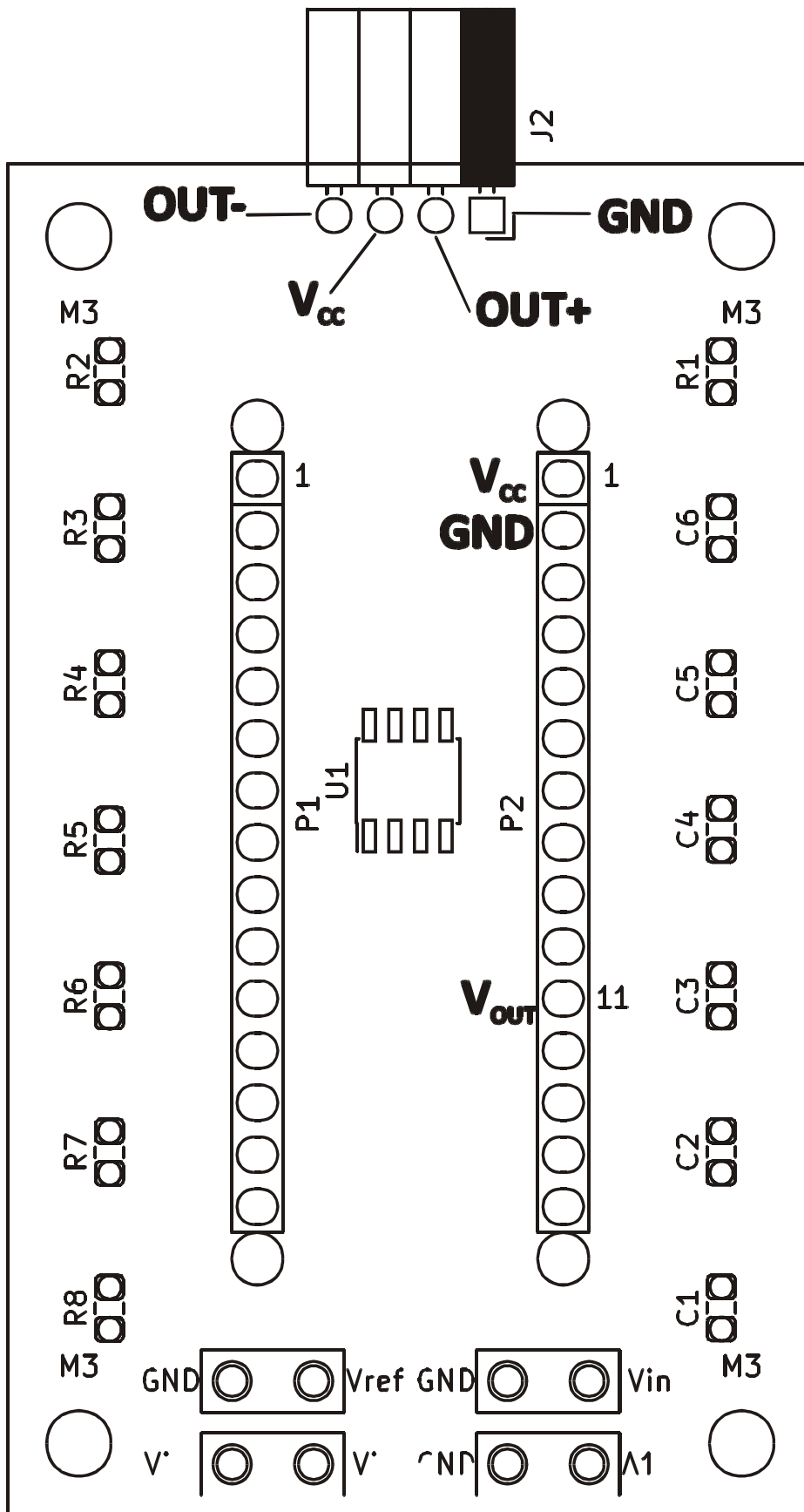
void LED3(bool state) {
    delay(1);
    if (state) {
        display.fillRect(80, 20, 10, 10, WHITE);
    }
    else {
        display.fillRect(80, 20, 10, 10, BLACK);
    }
    delay(1);
}
```

Slika 80: Vezalna shema senzorskega ojačevalnika in Arduino vmesnika..



Slika 81: TIV senzorskega ojačevalnika in Arduino vmesnika.- spodnja stran.



Slika 82: TIV senzorskega ojačevalnika in Arduino vmesnika. - zgornja stran.

