REPUBLIKA SLOVENIJA
MINISTRSTVO ZA ŠOLSTVO IN ŠPORT

KONZORCIJ ŠOLSKIH CENTROV

Naložba v vašo prihodnost
OPERACIJO DELNO FINANCIRA EVROPSKA UNIJA
Evropski socialni sklad

***MEHATRONIKA** JE POKLIC PRIHODNOSTI*

PROJEKTNO DELO

V

MEHATRONIKI

Matej Veber, Andro Glamnik

www.bodiprofi.si

munus 2

Uvajanje novih izobraževalnih programov v srednjem poklicnem in strokovnem izobraževanju s področja tehnike za obdobje 2008–2012 . Operacijo delno financira Evropska unija iz Evropskega socialnega sklada in Ministrstvo za šolstvo in šport.

SPLOŠNE INFORMACIJE O GRADIVU

Izobraževalni program
TEHNIK MEHATRONIKE, TEHNIK MEHATRONIKE PTI

Ime modula
PDS, PDP

Naslov učnih tem ali kompetenc, ki jih obravnava učno gradivo:

Delati v skupini, komunicirati s sodelavci in nadrejenimi ter strankami, uporabljati tehniške predpise in standarde pri pripravi tehniške dokumentacije, uporabljati tehniška navodila, izdelati delovno poročilo, izdelovati, spremljati in dopolnjevati delavniško dokumentacijo, izdelati in brati delavniške risbe in drugo tehnično dokumentacijo, uporabljati načine in pravila za vodenje dokumentacije, s pomočjo programske opreme izdelati enostavno tehniško dokumentacijo, aktivno sodelovati pri zagotavljanju zdravega in varnega dela, uporabljati osebni računalnik in spletne aplikacije za optimizacijo delovnega procesa, predstavitev osebnega ali timskega dela ožji ali širši javnosti.

Avtorja:

Matej Veber, univ. dipl. inž.

mag. **Andro Glamnik,** univ. dipl. inž.

Recenzent: Gvido Par, inž.

Lektorica: Lucija Lipovšek, prof.

Julij 2012

Izdajatelj: Konzorcij šolskih centrov Slovenije v okviru projekta MUNUS 2

Slovenija, Julij 2012

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

PREDGOVOR

V sodobnem svetu hitrih sprememb, hitrega razvoja in neizprosne konkurence se projektno delo celovito integrira v celoten proces sodobnih proizvodnih procesov. Družbo sestavljajo posamezniki, ki ustvarajajo in uvajajo spremembe, novosti in razvijajo nove ideje, invencije in inovativne izdelke. Pri tem si v prvem koraku običajno zastavijo cilj, ki pa ga lahko dosežejo na različne načine, poleg tega pa se običajno pojavijo različne ovire. Način reševanja ovir in problemov je zelo različen. Določeni subjekti problem rešijo, nekateri ga obbidejo, določeni pa problema preprosto ne rešijo. V ta namen je potrebno uporabiti določena orodja in strategije s pomočjo katerih dosežemo določeni cilj. Končni cilj lahko imenujemo projekt, pot do cilja pa strategija. Projektno delo nam omogoča sistema čni pristop k različnim aktivnostim, ki vodijo do končnega izdelka ali dejavnosti. V okviru izobraževanja morajo dijaki spoznati projektno delo in značilnosti le tega, saj bodo v relanem okolju neprestano v svojem pokliicnem okolju vpeti v aktivnosti projektnega dela. Poznamo projektna podjetja in proizvodna podjetja. Projektna podjetja izvajajo svoje aktivnosti znotraj časovnega okvira in običajno za zunanjega naročnika. Proizvodna podjetja prejemajo storitve projektnih podjetij in se običajno v okviru projekta ukvarjajo z masovno proizvodno izdelkov. V Sloveniji imamo veliko manjših podjetij, ki so konkurenčna v svetovnem merilu. Visokotehnološko podjetje kot je Pipistrel razvija svoje tehnološke dosežke in inovacije in ima naročila zagotovljena za pet let v naprej. Vendar takšno podjetje ne zaposluje veliko ljudi, poleg tega pa podjetje ni masovno proizvodno podjetje. Večina zaposlenih se ukvarja z razvojem in inovacijami. Sklope, ki jih vgradijo v letala jim dobavijo zunanji dobavitelji. Na drugi strani imamo veliko proizvodnih podjetij, ki proizvajajo manjše sklope za večja svetovne proizvajalce avtomobilov. Med drugim lahko omenim Schenker, Odelo, Unior, Šumer d.o.o., GKN Driveline in druge. Izdeljuje manjše enote kot so luči, vzmeti, različni zobniki in zglobi ter jih dobavljajo večjim proizvajalcem avtomobilov, ki jih nato vgradijo v osebna vozila. V Sloveniji imamo tudi mnogo projektnih podjetij, ki se ukvarjajo z industrijsko avtomatizacijo in orodjarstvom. V tej panogi lahko omenim Gorenje Indop, Gorenje Orodjarna, Emo Orodjarna, Unior, Roboteh, Etra in druga. V teh podjetjih so odvisni od trenutno izvajanih projektov in kakovosti izvedbe, ki je pogoj za nadaljna naročila. Tukaj je potrebno omeniti podjetje Emo Orodjarna, ki med drugim izdeluje orodja stiskalnic.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Ključne besede: Projektno delo, projekt, projektna organizacija, vodenje projektov, timsko delo, inovativnost, razvoj, pridobivanje idej, projektna programska orodja

Key words: *Project work, project, project organisation, project management, team work, inovativity, development, idea menegament, project software.*

KAZALO VSEBINE

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

KAZALO SLIK

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

PREDSTAVITEV CILJEV UČNE ENOTE

Dijaki in uporabniki bodo v okviru gradiva Projektno delo v mehatroniki dosegli sledeče informativne cilje učne enote :

- ✓ spozna cilje in aktivnosti projektnega dela,
- ✓ spozna vloge in zadolžitve v projektni skupini,
- ✓ spozna metode in postopke za razvoj od ideje do izvedbe projekta,
- ✓ spozna pomen odnosa do dela in delovnih sredstev,
- ✓ spozna pomen urejenega delovnega okolja,
- ✓ spozna delo na posameznih strokovnih področjih,
- ✓ spozna individualno in timsko delo,
- ✓ spozna časovni načrt izpeljave projekta,
- ✓ razdela idejni projekt po posameznih strokovnih
- ✓ področjih(elektrotehnika,mehanika in informatika),
- ✓ spozna finančni predračun projekta,
- ✓ spozna kakovost, nadzor in vrednotenje projekta,
- ✓ spozna osnove projektnega, timskega in individualnega dela,
- ✓ spozna pomen lastne pobude pri projektnem delu,
- ✓ spozna pojme kot so zagonska faza, razvojna faza, izvedbena faza in
- ✓ zaključna faza projekta
- ✓ spozna osnovni bonton javnega komuniciranja in nastopanja,
- ✓ spozna osnove tehnične dokumentacije
- ✓ spozna pomen uporabe strokovne literature in spletnih aplikacij,
- ✓ spozna osnove podjetništva,
- ✓ spozna vrste podjetij in njihove lastnosti (s.p., d.o.o., d.n.o, d.d., k.d…)
- ✓ spozna osnove projektnega managementa,
- ✓ spozna metode za iskanje idej (Brainstorming…),
- ✓ spozna načine predstavitve projektnega dela.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

POVZETEK

V današnjem času je zelo pogosto uporabljena beseda projektno delo. Velikokrat pa se poraja vprašanje kaj pravzaprav projektno delo sploh je. Z pričajočim novim gradivom sva avtorja želela približati modul projektno delo v stroki - Mehatroniki bralkam in bralcem. V začetku knjige so naprej podrobno razloženi osnovnih pojmi kaj je to projekt, projektno vodenje, planiranje.. Za učno situacijo je prikazano planiranje projekta s pomočjo Gantovega diagrama ter programskega okolja Microsoft Project. V nadaljevanju bralec spozna načine programiranje in vrste programskih jezikov, ki jih danes lahko zasledimo v uporabi. V naslednjem poglavju je podrobno razloženo delovanje mikroračnalnika in njegova zgradba. V tem času si praktično ne moremo zamisliti delujoče naprave, ki nebi imela integriranega mikroračunalnika kot primer lahko navedemo dlančnik, televizor kalkulator.. V poglavju senzorska tehnika je podrobno predstavljena senzorika, ki omogoča zajemanje fizikalnih veličin iz okolja in pretvorbo v električni signal. Senzorika je zelo veliko področje tako ga poglavje v celoti nikakor ne pokrije predstavljeni predstavljeni pa so najpomembnejši elementi senzorske tehnike. V poglavju Komponente močnostne elektronike so predstavljeni elementi ki služijo pretvarjanje električne napetosti v različne oblike. Dandanes imajo komponente močnostne elektronike posebno vlogo saj je vedno bol atraktiven vir energije sonce. Pri foto voltajičnih celicah pa moramo napetost razsmeriti, za kar potrebujemo pretvornike kateri so grajeni iz komponent močnostne elektronike. Projektna dokumentacija prestavlja zelo pomemben del pri načrtovanju in izvedbi projekta. Zato je v knjigi podrobno predstavljen program EPLAN, ki je najbolj razširjen program v Evropi za elektro projektivo. V sami knjigi je prikazan izveden projekt z EPLAN-om kjer lahko bralec usvoji osnovno rokovanje z tem programom. Želja avtorjev je bila ustvariti gradivo, ki bo omogočali sistematični prikaz načrtovanja in izvedbe projekta. Upava da nama je to uspelo. Vsekakor pa bova spoštovala strokovno kritiko, kar bo pripomoglo da bo gradivo v prihodnosti še boljše

Avtorja

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

1 KAJ JE PROJEKT?

Projekt je enkraten poslovni proces z jasno opredeljenimi cilji in časovnimi omejitvami, stroški in kakovostjo. Projekt lahko imenujemo tudi *strategija*.

Značilnosti: ***enkratnost, kompleksnost, interdisciplinarnost in tveganje*.**

Projekt je torej zaokrožen, časovno omejen skupek aktivnosti, ki privedejo do vnaprej zastavljenega cilja.

Rezultat projekta je nov izdelek ali storitev.
Viri za izvedbo projekta so omejeni.
Vsak projekt ima svoj konec.

V praksi poznamo dva tipa podjetij glede na način delovanja:

- **Podjetja kontinuiranega poslovanja**. To so običajno proizvodna podjetja (Gorenje, Krka, Revoz, Odelo…). *Razvoj izdelkov v podjetju mora biti prilagojen tržnim razmeram.*

- **Projektno usmerjena podjetja** (Metronik, Roboteh, Emo Orodjarna…), kjer je razvoj podprt s 3D-modeliranjem in simulacijo. *Primer projekt izdelave novega modela avtomobila,robotizacije proizvodne linije itd.*

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

TIMSKO DELO

Timsko delo pomeni sodelovanje posameznikov, postavljenih pred skupno nalogo oziroma skupen cilj – projekt. Tim je lahko v primerjavi s posameznikom bolj prilagodljiv, produktiven in kreativen, hkrati pa predstavlja večji potencial za generiranje idej. Med najbolj pomembnimi dejavniki, ki vplivajo na učinkovitost tima, so vodenje, medsebojno spremljanje uspešnosti, medsebojna podpora, prilagodljivost, timska naravnanost sodelujočih, komunikacija in zaupanje. Za uspešno timsko delo je potrebno zagotoviti tudi učinkovito in nemoteno komunikacijo ter razvoj in ohranjanje zaupanja med člani tima, saj pomanjkanje medsebojnega zaupanja pomeni izgubljanje časa v medsebojnem preverjanju in ščitenju lastnih interesov.

Za uspešnost tima je pomembno, da so njegovi člani visoko motivirani za delo, da so pripravljeni vložiti veliko napora v uresničevanje ciljev, da bi zadovoljili tako svoje interese kot tudi interese organizacije

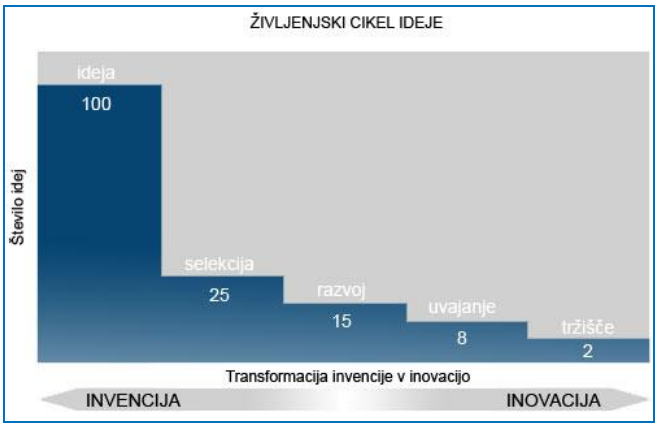
Glavne prednosti uspešnega timskega dela so:

- ✓ bolj učinkovito doseganje ciljev zaradi medsebojne podpore in sodelovanja,
- ✓ **člani tima sodelujejo pri odločanju,**zato je višja tudi zavezanost ciljem, ki jih sooblikujejo,
- ✓ **spodbuja občutek zaupanja med člani,**
- ✓ **spodbuja sproščeno izražanje idej, mnenj, vprašanj, dilem,**
- ✓ **spodbuja odprto in iskreno komunikacijo,**
- ✓ člani tima se medsebojno dopolnjujejo v znanju in sposobnostih,
- ✓ pozitiven vpliv na kvaliteto storitev,
- ✓ spodbuja prenos znanja in izkušenj.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

DOBRA IDEJA

Pri razvoju izdelka so pomembne ideje. Vse ideje pa niso dobre oziroma jih kot izdelek tržišče ne absorbira. Zato so potrebne **metode za izbiro najboljših idej** in s tem izdelkov, ki so konkurenčni na trgu ter dovolj poceni in kvalitetni.



Slika 1.1 : Življenjski cikel ideje

Vir: Lastni

Inovacija

Inovacija je nov izdelek, storitev ali postopek ali bistveno izboljšan izdelek, storitev ali postopek, ki se pojavi na trgu (inovacija izdelka, storitve) ali uporabi v okviru procesa (inovacija postopka).

Inovacije zajemajo vrsto znanstvenih, tehnoloških, organizacijskih, finančnih in gospodarskih aktivnosti. Inovativno podjetje je tisto, ki je v opazovanem obdobju uvedlo nov ali bistveno izboljšan proizvod ali postopek

Invencija

Razlika med invencijo in inovacijo je v tržnem zanimanju in ekonomski koristi za avtorja – **invencija postane inovacija šele, ko jo sprejmejo kupci** oziroma, ko ima avtor od inovacije neko ekonomsko korist. Izum in invencijo lahko enačimo, **ekonomske koristi pa prinašajo le inovacije.**

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Poznamo sledeče **metode za generiranje idej**.

Analitične, ki temeljijo nalogičnih miselnih procesih:

- ✓ Analiza lastnosti (analiza obstoječega izdelka, izboljšava)
- ✓ Raziskovanje potreb (sistematična analiza potreb)
- ✓ Tehnološko spremljanje (kombinacije različnih znanj)
- ✓ …

Neanalitične, ki spodbujajo domiselno razmišljanje:

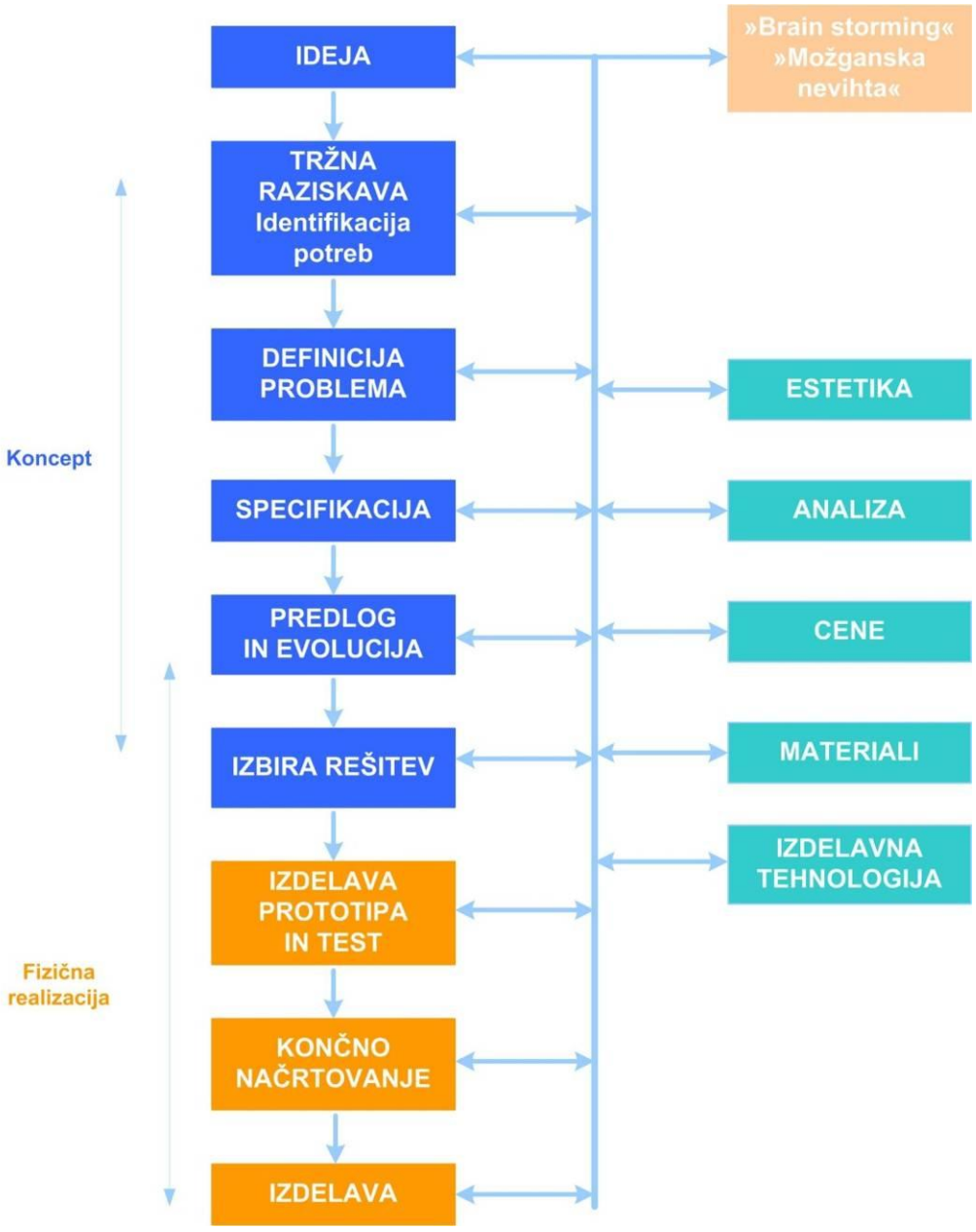
- ✓ Možganska nevihta (angl. Brain Storming)
- ✓ Sinektika (skupina tehnik)
- ✓ Lateralno razmišljanje (pomen vzorcev)
- ✓ Metoda SIL (multidimenzionalna vprašanja)
- ✓ Metoda 635 (vsak 3 ideje, listek)
- ✓ Metoda Delfi (na daljavo, pismo)

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

METODA ZA PRIDOBIVANJE IDEJ »BRAIN STORMING« OZ. NEVIHTA MOŽGAN

- ✓ Zbere se več oseb, npr. projektni tim v podjetju,
- ✓ zaželene so tudi osebe, ki pokrivajo druga strokovna področja (npr. filozofi, umetniki…) , ki imajo drugačen pogled na problem,
- ✓ vodja skupine, ki je dovolj avtoritativen in izkušen, vodi celoten proces pridobivanja idej,
- ✓ v tej metodi hierarhija ne pride v poštev,
- ✓ proces pridobivanja idej traja najdlje eno uro,
- ✓ problem se predstavi na razumljiv način,
- ✓ ideje se zapisujejo,
- ✓ zapisujejo se tudi najbolj neumne ideje,
- ✓ nastanejo verižne ideje in zamisli (nevihta možganov),
- ✓ izbira najbolj primerne rešitve.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 1.2. : Razvoj mehatronskih izdelkov od ideje do izvedbe

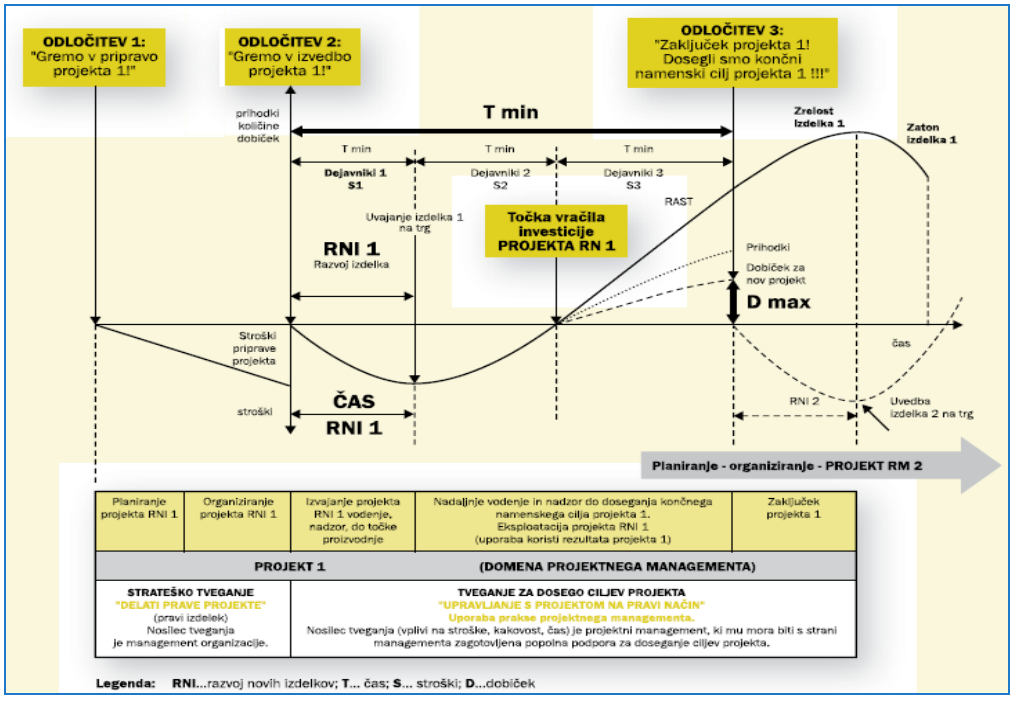
Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

V fazi razvoja, načrtovanja in izdelave izdelka ali sistema moramo upoštevati:

- **Kakovost** se nanaša na izdelke, storitve, trženje in druge procese ter na človeka kot ustvarjalca -> kakovost za potrebe višje konkurenčnosti,
- **minimalne stroške,**
- **čas izvedbe**-oziroma izvesti projekte prej kot konkurenca,
- **maksimalno ali zadostno proizvodnjo.**

Projekt ima svoj življenjski cikel. Na podlagi dobre ideje se odločimo ali gremo v izvedbo projekt. Sledi planiranje in orgniziranje projekta. V tej točki moramo imeti dobro zagotovljena finasnčna sredstva. Sledi izvedba, vodenje in nadzor projekta, doseči želimo namenski cilj. V tej fazi se začneja vračati investicija. Sledi uvajanje novega projekta, običajno je projekt že v fazi izvedbe, ko je prejšnji projekt najbolj donosen. Primer je novi model avtomobila. Avto pripravljajo že v fazi ko se starejši zelo dobro prodaja (modelno leto).



Slika 1.3 : Življenjski cikel projekta

Vir: www.gorenje.si

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

VRSTE POSLOVNIH SUBJEKTOV – PODJETIJ

Ob ustanovitvi je treba podjetje registrirati na sodišču, da podjetje postane pravna oseba. Običajno lahko osnovne vrste podjetij registriramo na E-Vem točkah na upravnih enotah, ostale pa v okviru notariata.

S pogodbo ustanovitelj določi:

- firmo in sedež podjetja (firma je ime, s katerim podjetje posluje, mora vsebovati oznako, ki nakazuje dejavnost družbe);
- pravno obliko podjetja, kjer se opredeli vrsta in obseg odgovornosti;
- statut podjetja:
  - ureja pravice in obveznosti,
  - način upravljanja podjetja,
  - način ugotavljanja dobička,
  - delež podjetja pri dobičku,
  - določijo se tudi organi podjetja.

Podjetja delimo na **osebne in kapitalske družbe**.

Samostojni podjetnik (s.p)

Podjetnik je **fizična oseba**, ki na trgu samostojno opravlja pridobitno dejavnost v okviru organiziranega podjetja. Poslovni izid te dejavnosti je obdavčen z dohodnino. Po zakonu o pokojninskem in invalidskem zavarovanju se mora vključiti v sistem obveznega zavarovanja in vsak mesec plačevati prispevke za socialno varnost. Kljub temu, da se vključi v sistem obveznega zavarovanja, pa to še ne pomeni, da je v svojem podjetju sklenil delovno razmerje, temveč gre za **samozaposleno osebo**. Zato si ne izplačuje plače, regresa in drugih bonitet, ki sicer pripadajo zaposlenim. Za razliko od gospodarskih družb lahko samostojni podjetnik **prosto razpolaga z zaslužkom** in ga lahko kadarkoli prenaša v porabo gospodinjstva.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Kapitalske družbe so:

- ✓ **Družba z omejeno odgovornostjo (d.o.o.)**

  Družba z omejeno odgovornostjo je družba, katere osnovni kapital sestavljajo osnovni vložki družbenikov. Vrednost vložkov je lahko različna. Osnovni kapital mora znašati vsaj 7.500 EUR, vsak osnovni vložek pa najmanj 50 EUR. Osnovni vložek je lahko zagotovljen v denarju ali kot stvarni vložek ali stvarni prevzem. Stvarni vložek so lahko premičnine in nepremičnine, pravice in podjetje ali del podjetja. Za stvarni vložek se šteje tudi plačilo za premoženjske predmete, ki jih je družba prevzela in jih prišteje družbenikovemu vložku. Družbo z omejeno odgovornostjo lahko ustanovi ena ali več fizičnih oziroma pravnih oseb, ki postanejo z ustanovitvijo družbe družbeniki. Družba se ustanovi s pogodbo, ki je lahko sklenjena v obliki notarskega zapisa ali na posebnem obrazcu, v fizični ali elektronski obliki, podpišejo pa jo vsi družbeniki. Če je družbena pogodba sklenjena na posebnem obrazcu, morajo biti podpisi družbenikov overjeni. Družbo z omejeno odgovornostjo lahko ustanovi tudi samo ena oseba (ustanovitelj), ki sprejme akt o ustanovitvi, za katerega ni potrebno, da je v obliki notarskega zapisa.

- ✓ **Delniška družba (d.d.)**

  Delniška družba je družba, kjer je osnovni kapital razdeljen na delnice. Najnižji znesek osnovnega kapitala je 25.000 EUR, najnižji nominalni znesek delnice je 1 EUR. Višji nominalni zneski se morajo glasiti na večkratnik 1 evra. Delniško družbo lahko ustanovi ena ali več fizičnih ali pravnih oseb, ki sprejmejo statut, ki mora biti izdelan v obliki notarskega zapisa. Delnice se lahko vplačujejo v denarju ali s stvarnimi vložki, vendar morajo vsaj tretjino osnovnega kapitala sestavljati delnice, ki se vplačajo v denarju. Organi vodenja ali nadzora so uprava, upravni odbor in nadzorni svet. Družba lahko izbere dvotirni sistem upravljanja družbe z upravo in nadzornim svetom ali enotirni sistem upravljanja družbe z upravnim odborom.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Osebne družbe so:

- ✓ **Družba z neomejeno odgovornostjo (d.n.o)**

  Družba z neomejeno odgovornostjo je družba dveh ali več oseb, ki odgovarjajo za obveznosti družbe z vsem svojim premoženjem. Družba z neomejeno odgovornostjo se ustanovi s pogodbo med družbeniki (družbena pogodba). Za ustanovitev ni predpisan osnovni kapital. Če ni drugače dogovorjeno, morajo družbeniki vplačati enake vložke. Vložek je lahko izražen v denarju, stvareh, pravicah ali storitvah. Poleg označbe dejavnosti in organizacijske oblike družbe (d. n. o) mora ime firme vsebovati priimek vsaj enega družbenika z navedbo, da je družbenikov več.

- ✓ **Komanditna družba (k.d.)**

  Komanditna družba je družba dveh ali več oseb, v kateri najmanj en družbenik odgovarja za obveznosti družbe z vsem svojim premoženjem (komplementar), medtem ko najmanj en družbenik za obveznosti družbe ne odgovarja (komanditist). Komanditna družba se ustanovi z družbeno pogodbo. Za ustanovitev ni predpisan osnovni kapital. V firmi komanditne družbe mora biti jasno naznačeno, kdo je komplementar, imena komanditistov pa je prepovedano vnašati. Kritje izgube in razdelitev dobička sta enaka kot v d. n. o. Posle družbe so dolžni voditi komplementarji. Če jih vodi komanditist, potem enako odgovarja za obveznosti družbe kot komplementar.

Ostale znane oblike podjetij so tudi **osebno dopolnilno delo** (popoldanska dejavnost, s.p.), **socialno podjetje** (zaposlene dolgotrajno brezposelne osebe, invalidi idt., so.p.) ter komanditno delniška družba (k.d.d.).

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

VODENJE PROIZVODNJE

Globalizacija je v industrijski menedžment vključila različne nove pristope kot so:

- Vitka proizvodnja (angl. ***lean manufacturing***), pomeni pripraviti učinkovito proizvodnjo.
- Poenostavljanje in vseobsežno »klestenje» (angl. ***downsizing***).
- Najemanje podizvajalcev in preselitev proizvodnje (angl. ***outsourcing***).
- Nenehno prilagajanje in spreminjanje (angl. ***reengineering***).

Za japonske poslovne modele je značilno:

- ***poudarjanje hitrosti in prožnosti, ne pa stroškov in produktivnosti;***
- ***poudarek na interdisciplinarni usposobljenosti zaposlenih (npr. mehatronika);***
- ***horizontalno neformalno komuniciranje med zaposlenimi ;***
- ***uporaba programiljivih univerzalnih obdelovalnih sistemov;***
- ***organiziranje v skupine.***

Omenjeno omogoča japonskim industrijskim poslovnim sistemom konkurenčnost na svetovnem trgu.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Kaizen predstavlja splošni model, ki ga sestavljajo pristopi:

▪ **Podjetje nima svojih zalog**	**Just in Time**	**JIT**
▪ **vitka proizvodnja**	Lean Manufacturing	
▪ **celovito obvladovanje kakovosti**	Totaly Qualitiy Managment	**TQM**
▪ **preobrazba jedra procesa**	Core Process Reenginering	**CPR**
▪ **računalniško integrirana proizvodnja**	Computer Integrated Manufacturing	**CIM**
▪ **biološki proizvodni sistem**	Biological Manufacturing System	**BMS**
▪ **agilna proizvodnja**	Agile Manufactoring	
▪ **prilagodljivost uporabnikom**	Mass Costumization	
▪ **navidezno podjetje**	Virtual Company	
▪ **fraktalno podjetje**	Fractal Company	

Izraz **Kaizen** predstavlja obnašanje posameznika v okviru skupine. Skupina ves čas stremi k izboljšanju aktivnosti v delovnem procesu.

Temeljna načela **Kaizena** so:

- ✓ **biti vsak dan boljši;**
- ✓ **narediti še tako majhno stvar bolje;**
- ✓ **obravnavati vsakogar kot kupca.**

Kot sistem temeljnih načel je **Kaizen** že desetletja prisoten v japonskem industrijskem okolju,v Evropi in ZDA pa se zelo uveljavlja. Primer je podjetje Odelo Prebold, ki proizvaja zadnje avtomobilske luči za različne svetovne proizvajalce avtomobilov. Med drugim izdelujejo luči za Porche, BMW, Audi itd. V podjetju se trudijo uporabljati **načela kaizena**. Vsak delavec mora poskrbeti za nenehne izboljšave in inovacije. Vsak delovni proces se trudijo vsak dan izboljšati. Delajo v skupinah, ki se nenehno izobražujejo in izboljšujejo. Velik poudarek dajejo tudi čistoči delovnega prostora. V veliki meri poskušajo odstraniti človeški faktor oziroma

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

napake, ki se pojavijo v procesu zaradi napak zaposlenih. Uvajajo **jogo-pogo** (»trotl ziher«) sisteme, ki onemogočajo napake pri delu. Poleg omenjenega uvajajo sistem Just in Time (**JIT**) in sistem FIFO, ki omogoča, da delavec povleče določen sklop, za njim pa se vsi elementi prestavijo na naslednji proces. Omenjeno je del **vitke proizvodnje**, ki je vodilo celotne proizvodnje podjetja. Zadane imajo tudi cilje in strategije za dosego kakovosti in statusa v rangu najboljših svetovnih podjetij. **Vitka proizvodnja** (klestenje) temelji na osnovnem načelu Kaizena. Timi zaposlenih se trudijo neprestano izboljšati proizvodni peoces. To pomeni, da podjetje porabi :

- **manj dela in naporov;**
- **manj proizvodnega prostora;**
- **manj investicij;**
- **manj orodij;**
- **manj časa;**
- **manj vsega.**

Nekaj pojmov, ki se navezujejo na izboljšanje učinkovitosti dela:

- ✓ **JOGO POGO (»TROTL ZIHER«)** - sistem onemogoča človeške napake. Pri tem projektant upošteva vse možne napake in sistem načrtuje čim bolj logično brez možnosti napak in človeških vplivov na delovanje naprave.

- ✓ **MILK RUN** – sistem za logistiko materiala in obdelovancev. Poznamo logistiko zunaj podjetja in logistiko znotraj podjetja. Najprej se na delovno mesto dostavi polni voziček. Nato »mlekar« vozi material in polni vozičke na podlagi zahtev delavcev na določeni točki proizvodnega procesa.

- ✓ **KANBAN** – sistem vozičkov za dobavo material v proizvodni proces. Delavec postavi kartico z naročilom. Dostavna kompozicija pobere kartice in napolni vozičke, nato pa vsem dostavi material.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

UPRAVLJANJE KAKOVOSTI

Razvoj in konkurenčnost podjetij postavlja podjetja v vlogo ko morajo zagotavljati kakovost in posledično kontrolo le te. Potrošniki postavljajo visoke zahteve, ki pa so tudi konkurenčna prednost:

- visok tehnološki nivo izdelka,
- upoštevanje rokov dobave,
- visoka kakovost.

V času turbo kapitalizma, kjer imajo izdelki relativno kratek in omejen rok trajanja, je kakovost najpomembnejši argument pri nakupu izdelka. V ta namen imajo podjetja običajno oddelek za kakovost in kontrolo le te. Gospodarno se ukvarjajo se z različnimi meritvami, umerjanji, načrtovanjem in izobraževanjem zaposlenih na področju zagotavljanja kakovosti. Govorimo o sistemu zagotavljanja kakovosti in politiki kakovosti v podjetju. Namen kakovosti je doseganje ciljev, ki so povezani s:

- stroški,
- prožnostjo,
- kakovostjo,
- zanesljivostjo dobaviteljev,
- upoštevanjem rokov.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

PROJEKTNO DELO V PEDAGOŠKEM PROCESU

Projektno delo v mehatroniki sestavljajo učitelj teoretične vsebine, učitelj praktičnega pouka (zaželeno, da oboje pokrije en učitelj), učitelji splošnih predmetov (medpredmetno povezovanje in integracija), učenec in učne vsebine, ki se navezujejo na projekt ter realne vsebine in procese.

Osnovne značilnosti pedagoškega projektnega učnega dela:

- Tematsko – problemski pristop
- Konkretne vsebine, usmerjene na življenjsko situacijo in omejene s cilji strokovnih modulov
- Ciljno usmerjena in načrtovana dejavnost s poudarkom na aktivnosti dijakov
- Upoštevanje potreb in sposobnosti dijakov
- Poudarek na izkustvenem učenju
- Kooperativnost
- Sodelovanje med učitelji
- Sodelovanje med učitelji in dijaki
- Sodelovanje med dijaki
- Odprtost projektnega dela
- Poudarek na učenju kot procesu

Projektne faze:

1. **Idejna zasnova projektnega dela**
2. **Makro priprava projektnega dela**
3. **Mikro priprava projekta (interdisciplinarni pristop)**
4. **Izvedba projekta:**
   a. **Projektna dokumentacija (zbrana v mapi)**
   b. **Fizična izvedba**
5. **Sklepna faza projekta z evalvacijo.**

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Fizična izvedba projekta mora vsebovati:

1. *Sestavno risbo izdelka ali (tehnično dokumentacijo v primeru storitvene naloge).*
2. *Delavniške risbe za vse sestavne dele ali (prospekte in navodila za storitev).*
3. *Montažni načrt*
4. *Inštalacijski načrt*
5. *Tehnološke postopke za izdelavo ali (opis postopka za storitveno dejavnost).*
6. *Popis potrebnega materiala, ki izhaja iz sestavne risbe (vrsta in kvaliteta, dimenzija).*
7. *Spisek potrebnih strojev in orodja za izvedbo*

Učitelj mora pripraviti:

Zbirni seznam, iz katerega je razvidna razporeditev dijakov v skupine in vloga posameznega dijaka pri projektnem delu.

Ocenjevalni list, ki bo spremljal uspehe dela posameznega dijaka in skupine (od ideje do predstavitve).

Vir: J. Trotovšek

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

UČNA SITUACIJA 1: NAČRTOVANJE PROJEKTA

- Dijaki se razdelijo v skupine in izvajajo timsko delo. S pomočjo metode Brain Storming dijaki izberejo projekt iz področja mehatronike.
- Sledi izdelava zagonskega elaborata projekta.
- Dijaki določijo strategijo in cilje projekta.
- S pomočjo programske opreme MS Project načrtujejo projekt.
- Pred publiko javno predstavijo svoje delo.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Poglavja zagonskega elaborata:

1. **Uvod** – povzetek naročila z namenom, obsegom in omejitvami .

2. **Podrobnejša specifikacija proizvodov** projekta – včasih je opredelitev proizvodov, ki jih naročnik posreduje v naročilu nepopolna, zato projektni manager v sodelovanju z naročnikom razjasni dileme in dopolni specifikacijo (ki jo naročnik potrdi s podpisom); občasno so potrebne tudi dodatne analiza in študije;

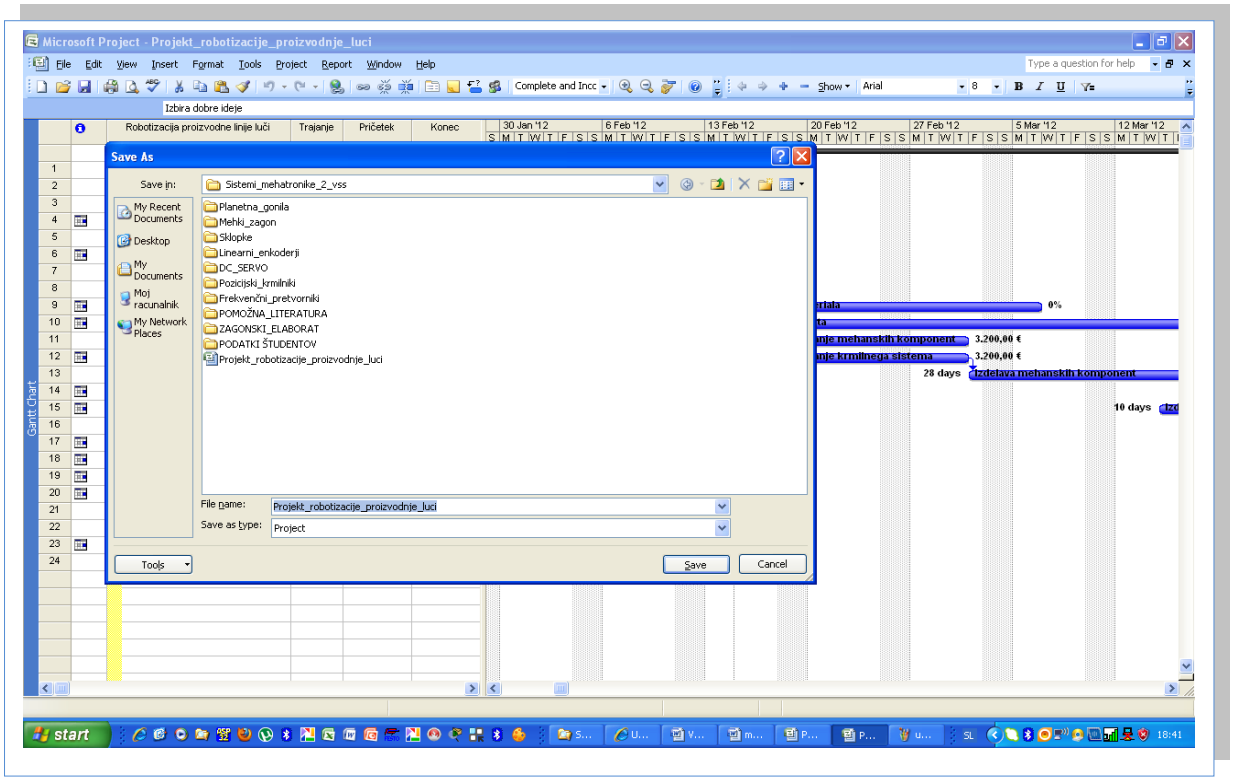
3. **Plani projekta** – taktika izvedbe, WBS, terminski plan, plan virov, plan stroškov, plan obvladovanja tveganj, (opcijsko pa tudi plani kontrole, nabave, zagotavljanja kakovosti, …) –MS Project.

4. **Organizacija projekta** – organizacijska struktura, organigram udeležencev projekta, opredelitev vlog, management dokumentacije in informiranja, poslovnik projekta.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

KRATKA NAVODILA ZA DELO S PROGRAMSKO OPREMO *MS PROJECT*

- ✓ Najprej projekt shranimo tako, da kliknemo **File -> New**.
- ✓ Nato shranimo projekt **File -> Save as...**
- ✓ Projekt imenujemo: uporabimo eno ali največ dve besedi npr. **Projekt robotizacije.**

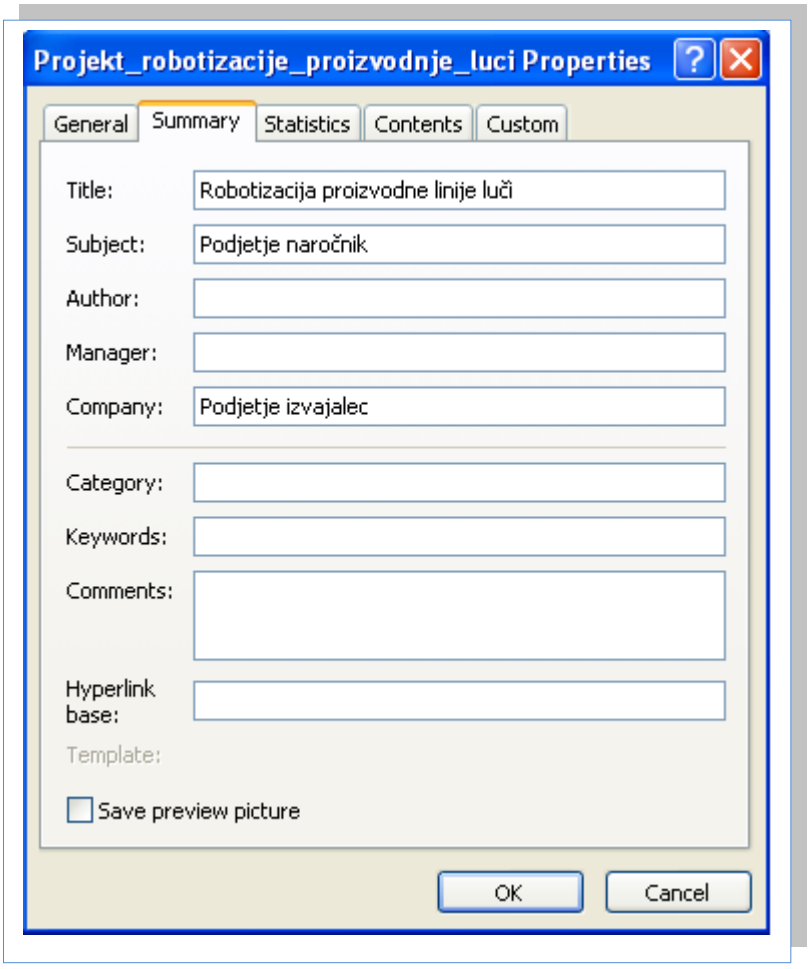


Slika 1.4 : Programsko okolje MS Project

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

✓ *Vnašanje splošnih podatkov o projektu* ***File, Properties.***

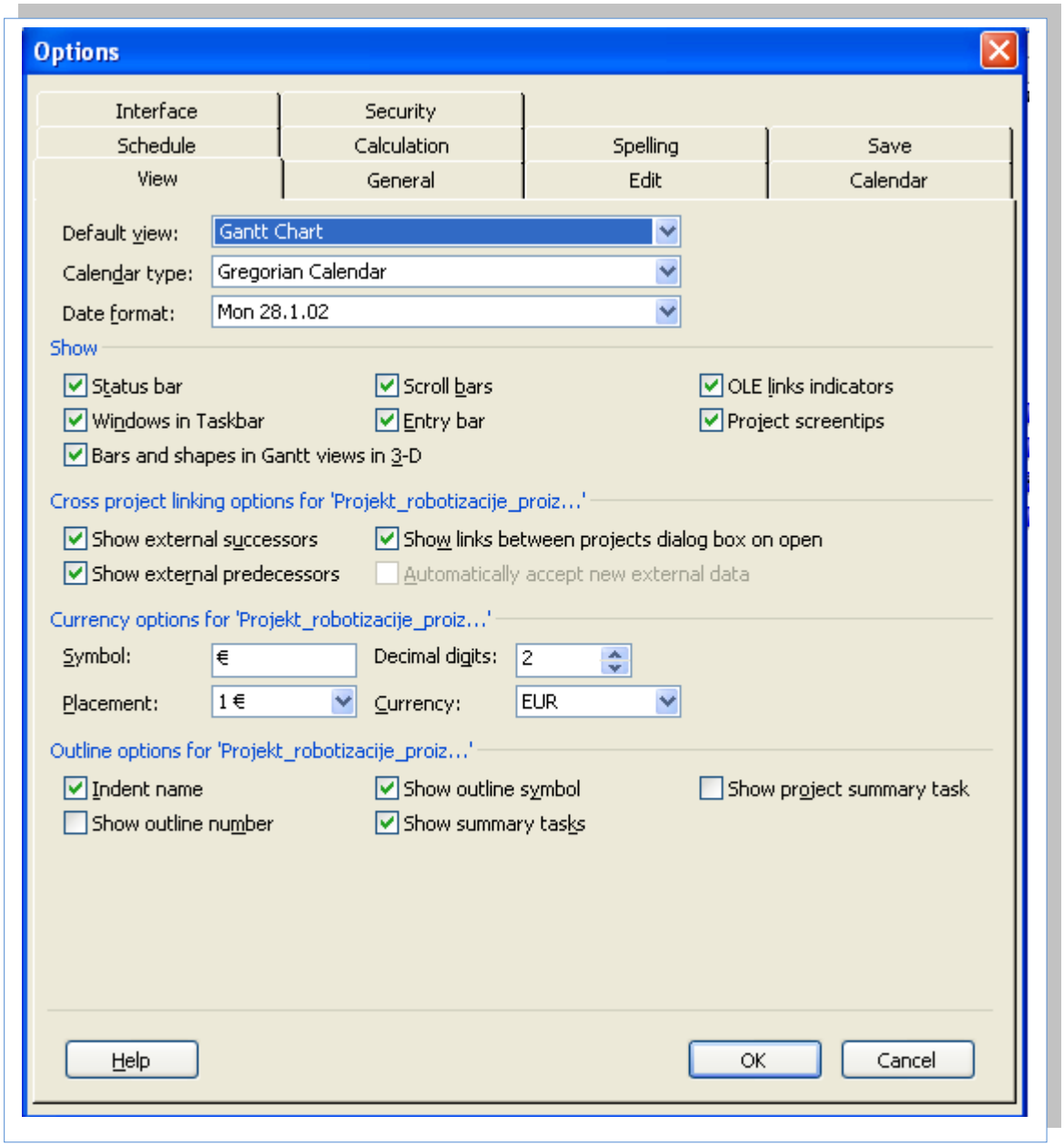


Slika 1.5 : Nastavitev lastnosti

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

✓ *Nastavitev možnosti **Tools-> Options** -> **Pogled (View)***

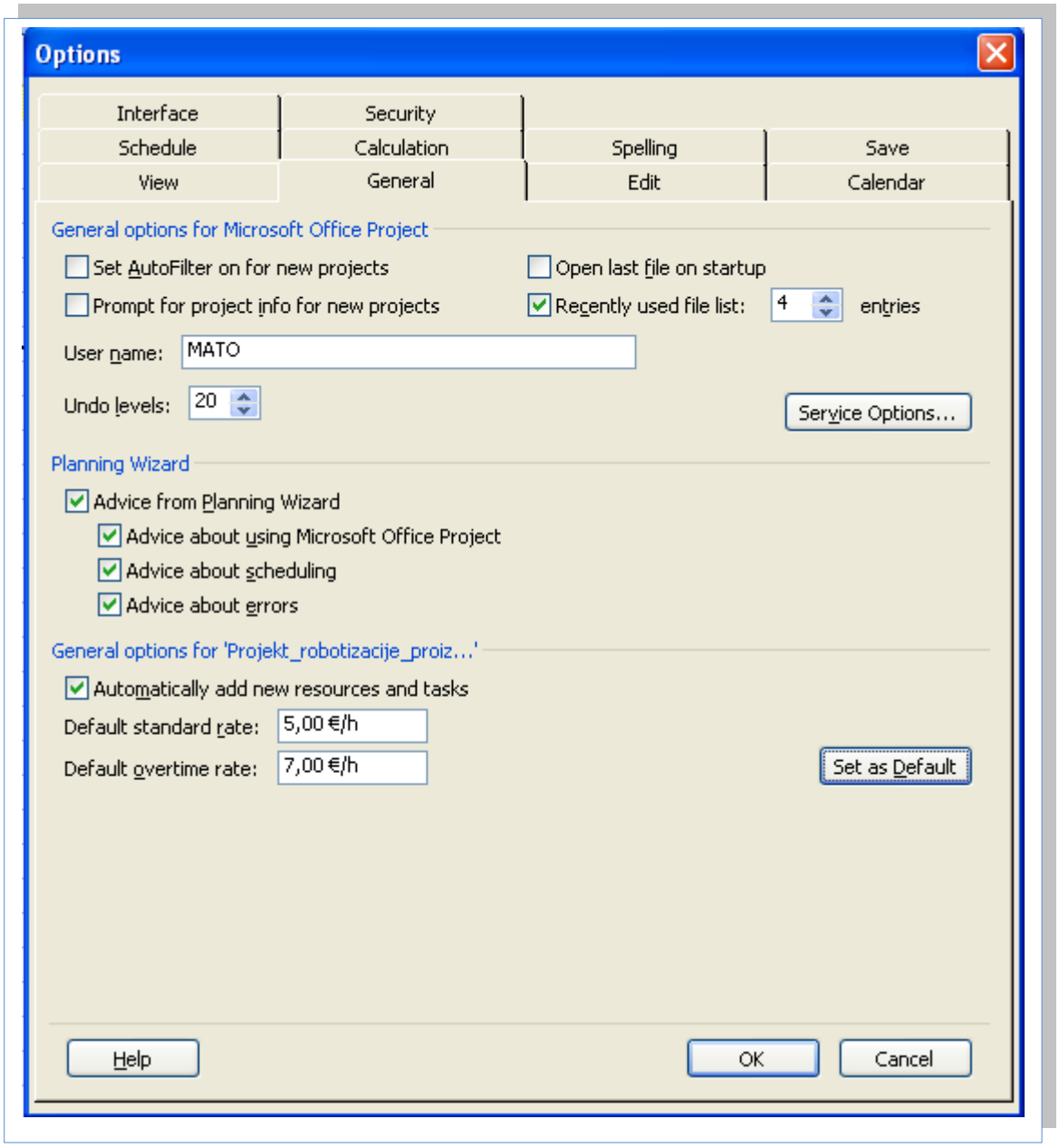


Slika 1.6: Nastavitev pogleda

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

✓ *Nastavitev možnosti **Tools -> Options-> Splošno (General)***

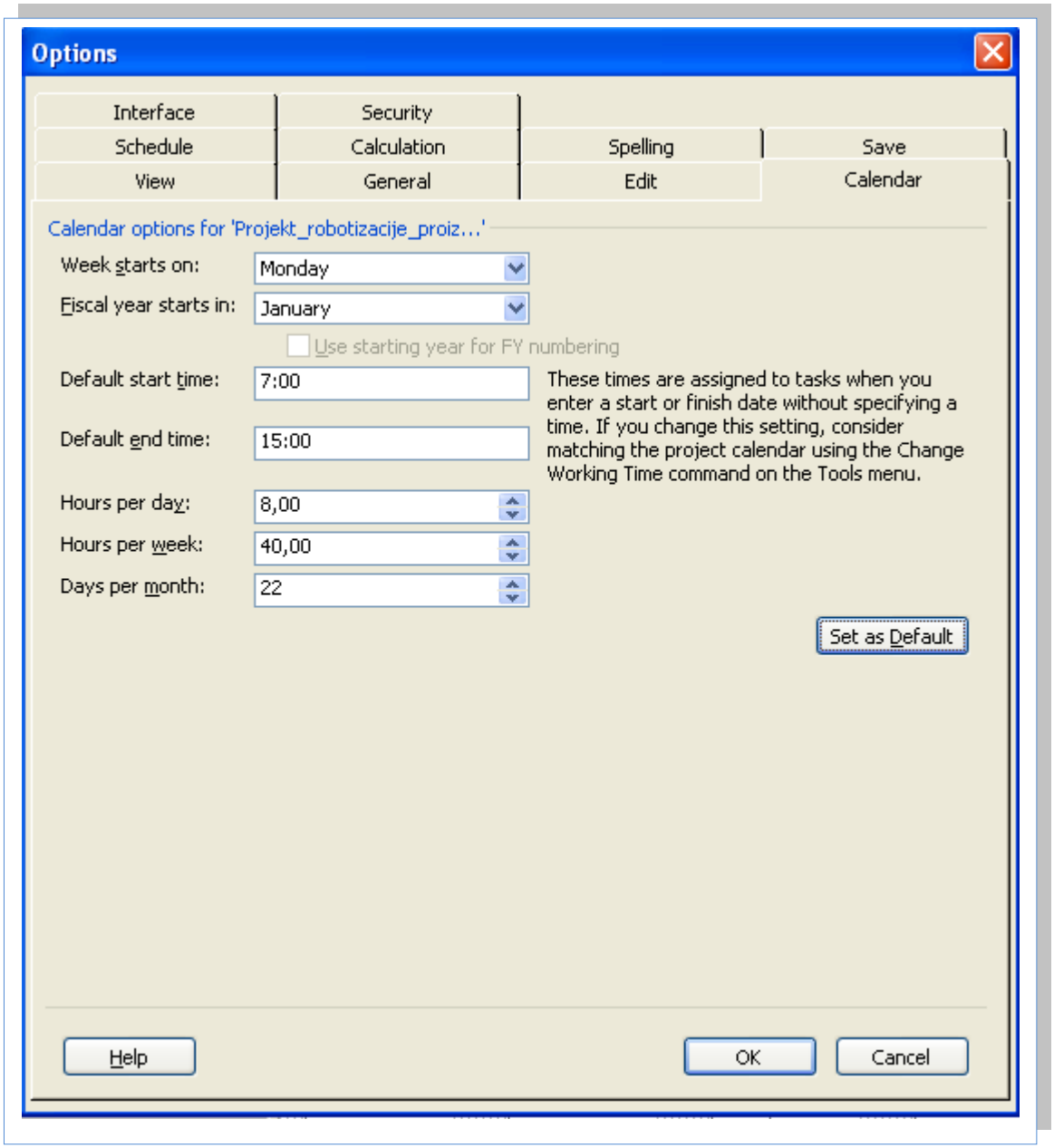


Slika 1.7 : Nastavitev možnosti

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

✓ *Nastavitev možnosti **Tools -> Options -> Koledar (Calendar)***

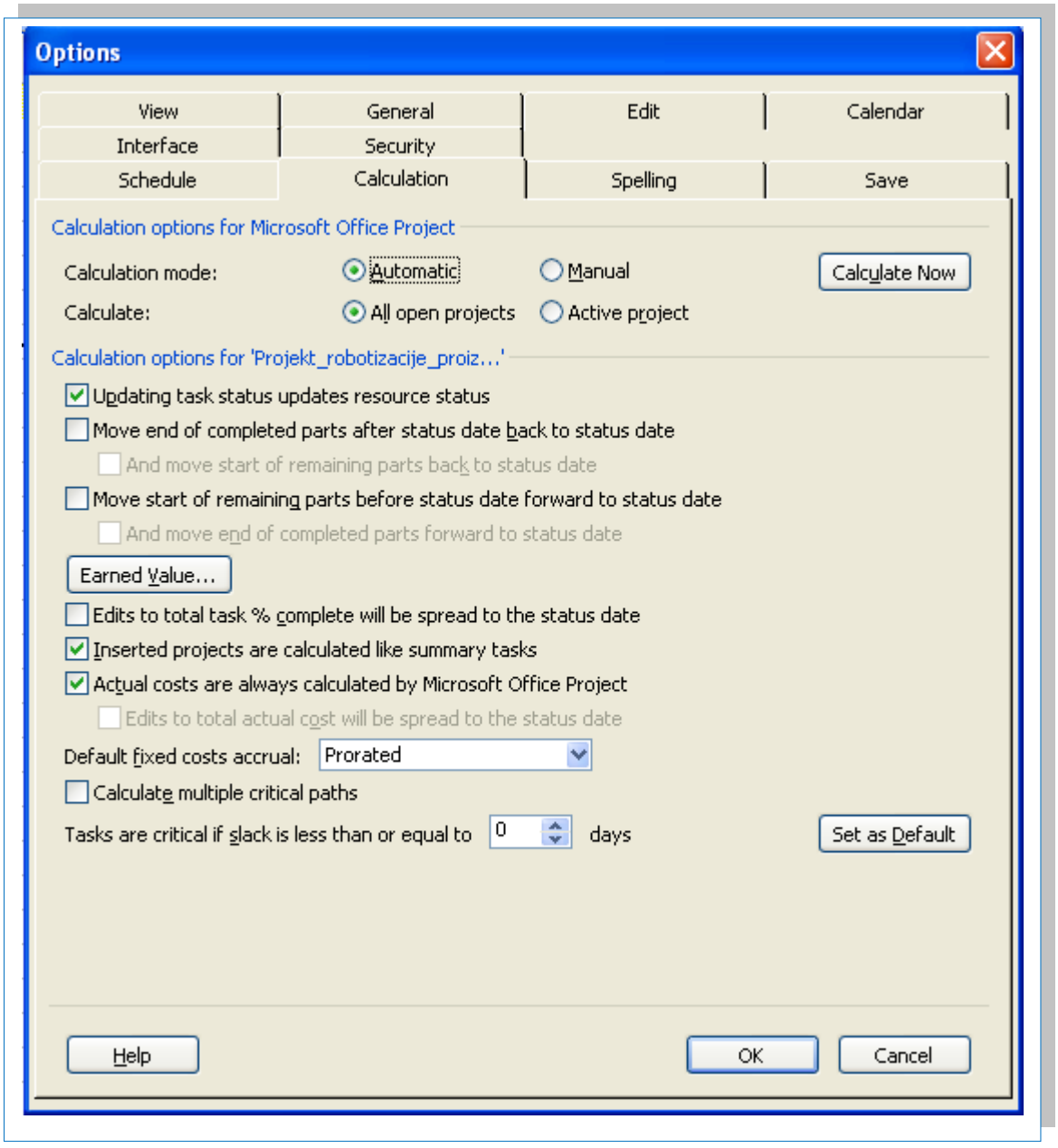


Slika 1.8 : Nastavitev časovnih možnosti

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

✓ *Nastavitev možnosti **Tools -> Options-> Kalkulacija (Calculation)***

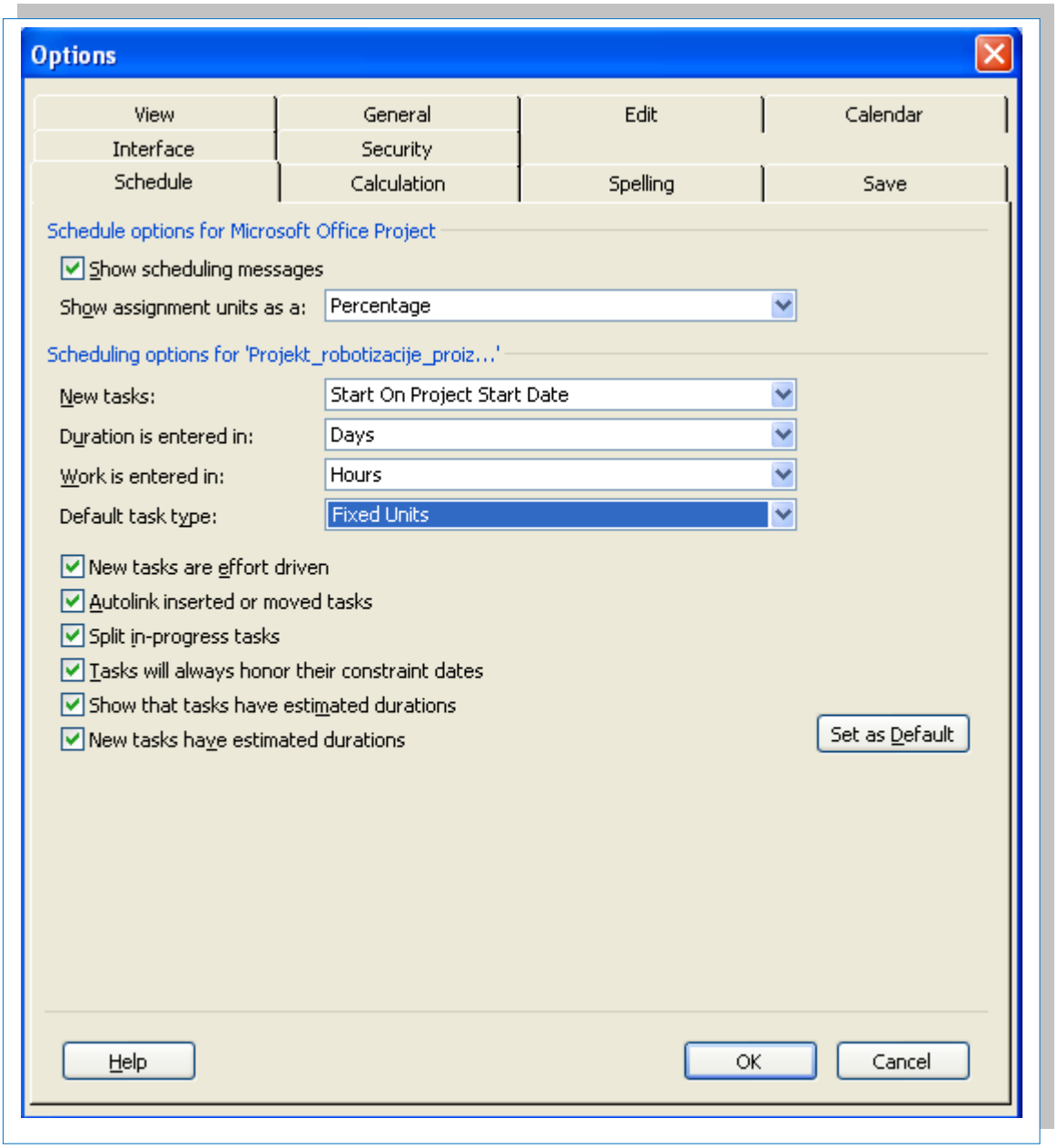


Slika 1.9: Nastavitev kalkulacije

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

✓ *Nastavitev možnosti **Tools -> Options-> Časovni načrt (Schedule)***

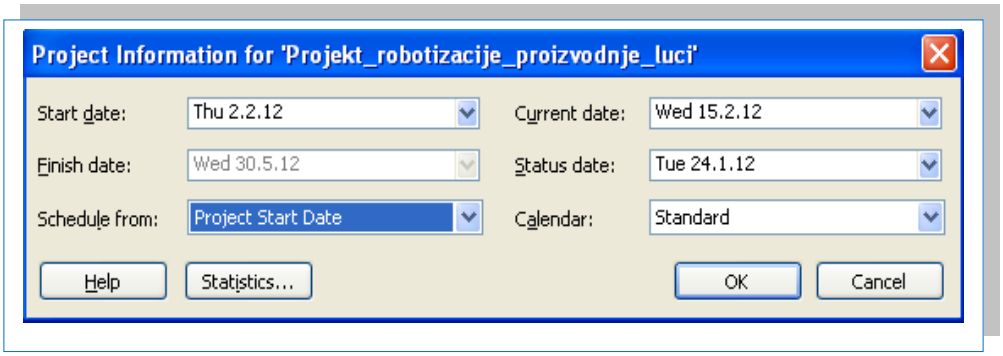


Slika 1.10: Nastavitev časovnega plana

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

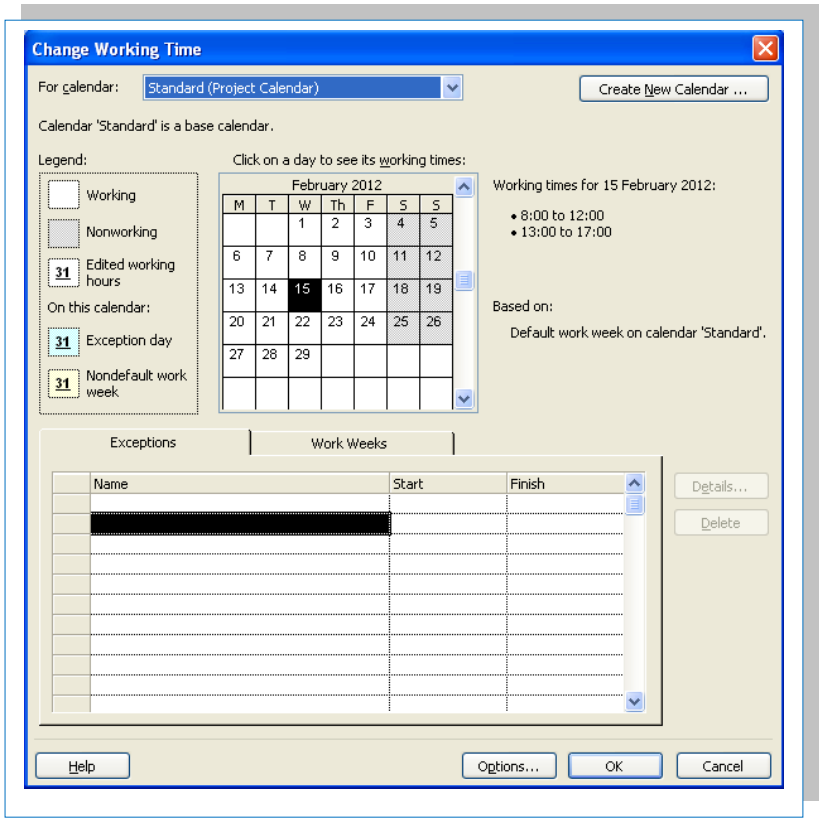
✓ *Podatki o začetku projekta* ***Project -> Project information***



Slika 1.11: Časovni okvir

Vir: Lastni

✓ *Nastavitev delovnega koledarja projekta* ***-> Tools -> Change Working Time***



Slika 1.12: Nastavitev delovnega časa

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

✓ *Vnos dejavnosti in njihovega trajanja* ***View-> More Views ->Task Sheet***

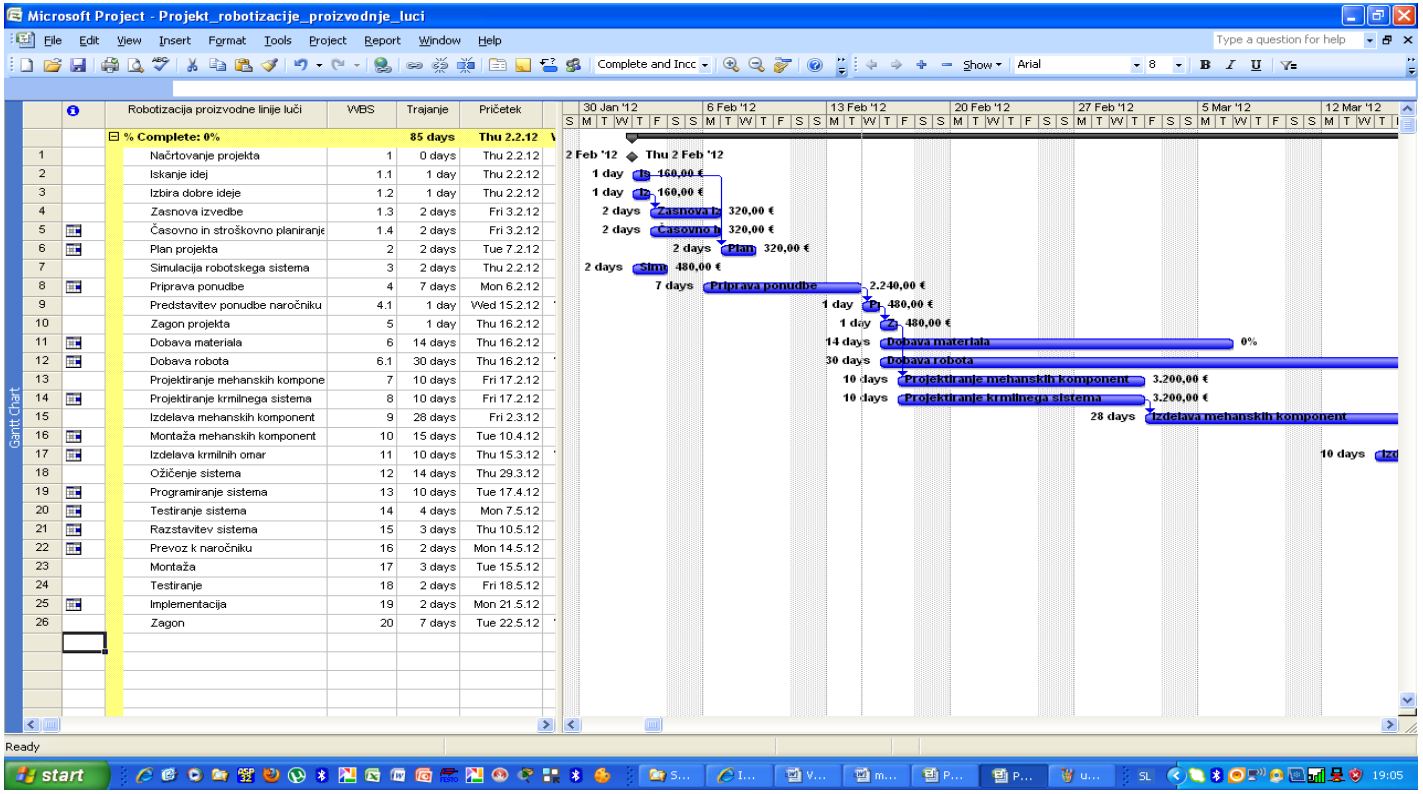
		Robotizacija proizvodne linije luči	WBS	Trajanje	Pričetek	Konec	Predecessors	Resource Names
1		Načrtovanje projekta	1	0 days	Thu 2.2.12	Thu 2.2.12		
2		Iskanje idej	1.1	1 day	Thu 2.2.12	Thu 2.2.12		Zaposleni
3		Izbira dobre ideje	1.2	1 day	Thu 2.2.12	Thu 2.2.12		Zaposleni
4		Zasnova izvedbe	1.3	2 days	Fri 3.2.12	Mon 6.2.12	3	Zaposleni
5		Časovno in stroškovno planiranje proj	1.4	2 days	Fri 3.2.12	Mon 6.2.12		Zaposleni
6		Plan projekta	2	2 days	Tue 7.2.12	Wed 8.2.12	2	Zaposleni
7		Simulacija robotskega sistema	3	2 days	Thu 2.2.12	Fri 3.2.12		Programer
8		Priprava ponudbe	4	7 days	Mon 6.2.12	Tue 14.2.12		Marketing
9		Predstavitev ponudbe naročniku	4.1	1 day	Wed 15.2.12	Wed 15.2.12	8	Vodstvo
10		Zagon projekta	5	1 day	Thu 16.2.12	Thu 16.2.12	9	Vodstvo
11		Dobava materiala	6	14 days	Thu 16.2.12	Tue 6.3.12		Robot
12		Dobava robota	6.1	30 days	Thu 16.2.12	Wed 28.3.12		Robot
13		Projektiranje mehanskih komponent	7	10 days	Fri 17.2.12	Thu 1.3.12	10	Projektant
14		Projektiranje krmilnega sistema	8	10 days	Fri 17.2.12	Thu 1.3.12		Projektant
15		Izdelava mehanskih komponent	9	28 days	Fri 2.3.12	Tue 10.4.12	14	Mehatronik
16		Montaža mehanskih komponent	10	15 days	Tue 10.4.12	Mon 30.4.12		Mehatronik
17		Izdelava krmilnih omar	11	10 days	Thu 15.3.12	Wed 28.3.12		Mehatronik
18		Ožičenje sistema	12	14 days	Thu 29.3.12	Tue 17.4.12	12	Mehatronik
19		Programiranje sistema	13	10 days	Tue 17.4.12	Mon 30.4.12		Mehatronik
20		Testiranje sistema	14	4 days	Mon 7.5.12	Thu 10.5.12		Mehatronik
21		Razstavitev sistema	15	3 days	Thu 10.5.12	Mon 14.5.12		Mehatronik
22		Prevoz k naročniku	16	2 days	Mon 14.5.12	Tue 15.5.12		Prevoznik
23		Montaža	17	3 days	Tue 15.5.12	Thu 17.5.12	21	Mehatronik
24		Testiranje	18	2 days	Fri 18.5.12	Mon 21.5.12	23	Mehatronik
25		Implementacija	19	2 days	Mon 21.5.12	Tue 22.5.12		Mehatronik
26		Zagon	20	7 days	Tue 22.5.12	Wed 30.5.12	24	Mehatronik

Slika 1.13: Nastavitev kalkulacije

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

✓ *Prikaz vnešenih dejavnosti v pogledu Ganttov diagram* ***View -> Gantt Chart***



Slika 1.14: Ganttov diagram

Vir: Lastni

- ✓ **Vpišite podatke iz spodnje tabele v spodnja okna programa MS Project: Definicija razpoložljivih resursov.**

- ✓ **Iz tabele lahko razberemo stroške, ki nastajajo pri naših projektih, merske enote (ME) in njihova cena na enoto v EUR.**

Stroški	Količina	ME	Cena/enoto (v EUR)	Stroški (v EUR)
1. Človeški viri				
1. 1 Plača direktorja		ur	5,43	
1. 2 Plača mehatronika		ur	5,43	
1.3 ???				
Vmesni znesek človeški viri				
2. Amortizacija opreme				
2. 1 Amortizacija računalnika		ur	0,18	
2. 2. Amortizacija pisarniškega pohištva		ur	0,09	
2. 3 Amortizacija fotokopirnega stroja		ur	0,24	
2. 4 Amortizacija skenerja		ur	0,03	
2. 5 Amortizacija tiskalnika		ur	0,06	
Vmesni znesek amortizacija opreme				
3. Stroški storitev drugih				
3. 1 Plačilo telefona, faksa, interneta	1	ur	1,2	
3. 2 Plačilo elektrike	100	KWh	0,06	
3. 3 Plačilo ogrevanja	0,3	MWh	39,20	
3. 4 Plačilo vzdrževanja	1	ure	16,69	

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Vmesni znesek stroški storitev drugih				
4. Materialni stroški				
Pisarniški listi A4		kos	0,01	
Kemični svinčnik		uporab	0,01	
toner		listi	0,21	
Vmesni znesek materialnih stroškov				
5 Skupaj stroški projekta (1. – 4.)				
6 Rezerva na nepredvidene stroške (največ 5 % od 5)				
7 Skupaj stroški (5+6)				

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

✓ *Definicija razpoložljivih virov* ***View -> Resource sheet.***

Resource Sheet

	Ime resursa	Kratice	Skupina resursov
1	Strokovni sodelavec 1	MR	Delo
2	Strokovni sodelavec 2	TB	Delo
3	Računalnik 03	RAČ 03	Stroj
4	Računalnik 04	RAČ 04	Stroj
5	Pisarniško pohištvo 03	PIS 03	Oprema
6	Pisarniško pohištvo 04	PIS 04	Oprema
7	Foto kopirni stroj	FOT	Stroj
8	Skener	SKE	Stroj
9	Tiskalnik	TIS	Stroj
10	Telekom	TEL	Storitve
11	Elektro	ELE	Material
12	Ogrevanje	OGR	Material
13	Vzdrževalec	VZD	Delo
14	Pisarniški listi A4	A4	Material
15	Kemični svinčnik	KEM	Material
16	Toner	TON	Material

Slika 1.15: Tabela virov

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

*Potrditev potrebnih resursov na dejavnosti **View, Task Usage** – nato izberemo dejavnosti in stisnemo tipki **Alt + F10** za prikaz okna za potrditev resursov.*

- ✓ *Uporaba virov **View -> Task Usage**.*

Task Usage

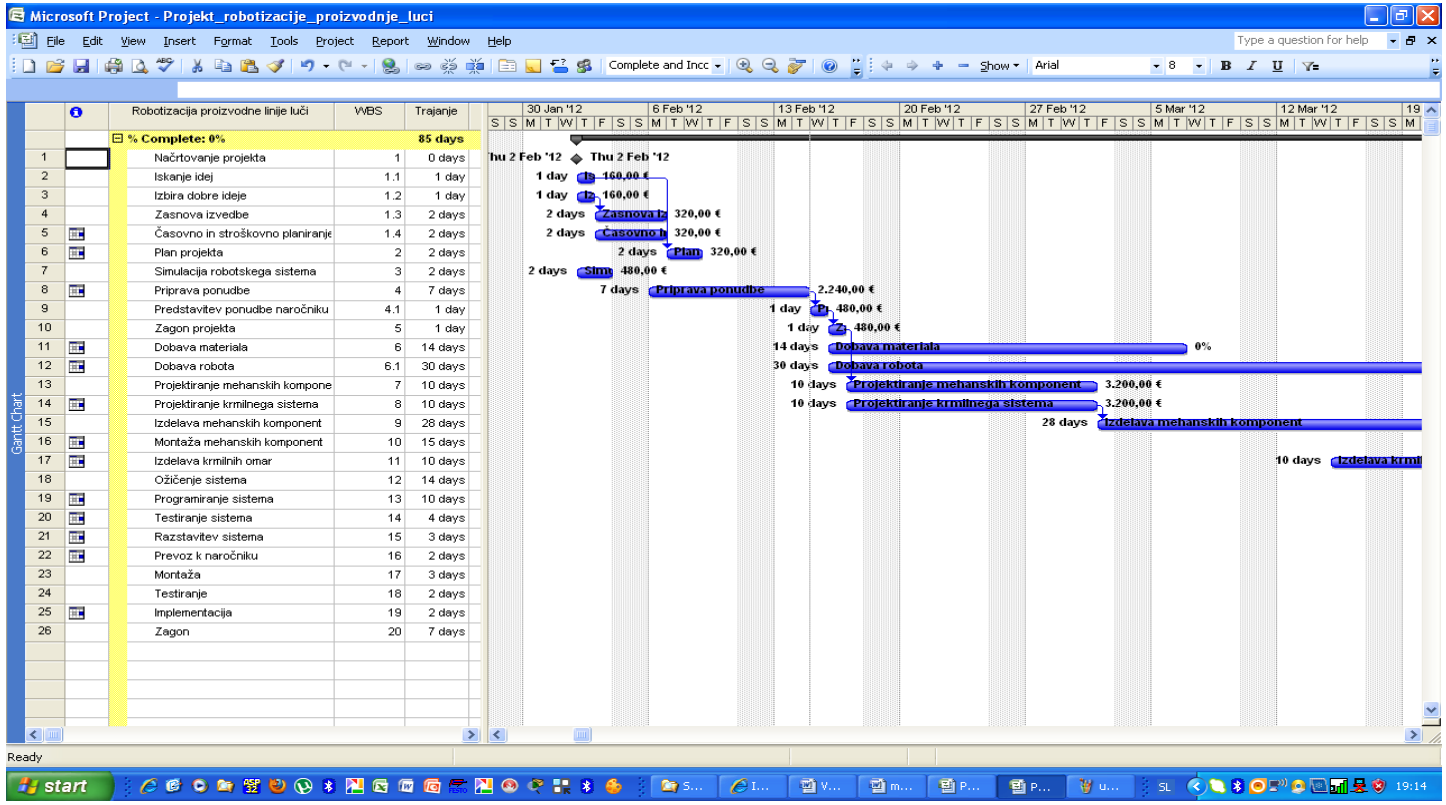
	Ime faze, dejavnosti	Predvideno trajanje	Št. Asigniranih	Planirano delo
1	**Načrtovanje projekta**	**6 wks**		**93 hrs**
2	Zamisel projekta	2 wks		18 hrs
	Strokovni sodelavec 1		*100%*	*3 hrs*
	Strokovni sodelavec 2		*100%*	*3 hrs*
	Računalnik 03		*50%*	*3 hrs*
	Računalnik 04		*50%*	*3 hrs*
	Pisarniško pohištvo 03		*100%*	*3 hrs*
	Pisarniško pohištvo 04		*100%*	*3 hrs*
3	Zasnova projekta	2 wks		30 hrs
	Strokovni sodelavec 1		*100%*	*6 hrs*
	Strokovni sodelavec 2		*100%*	*6 hrs*
	Računalnik 03		*100%*	*6 hrs*
	Pisarniško pohištvo 03		*100%*	*6 hrs*
	Pisarniško pohištvo 04		*100%*	*6 hrs*
	Pisarniški listi A4		*10*	*10*
4	Časovno in stroškovno planiranje projekta	3 wks		45 hrs
	Strokovni sodelavec 1		*100%*	*9 hrs*
	Strokovni sodelavec 2		*100%*	*9 hrs*
	Računalnik 03		*50%*	*4,5 hrs*
	Računalnik 04		*50%*	*4,5 hrs*
	Pisarniško pohištvo 03		*100%*	*9 hrs*
	Pisarniško pohištvo 04		*100%*	*9 hrs*
	Pisarniški listi A4		*4,5*	*4,5*
5	**Izvajanje projekta**	**10,67 wks**		**207,6 hrs**

Slika 1.16: Razporeditev nalog

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

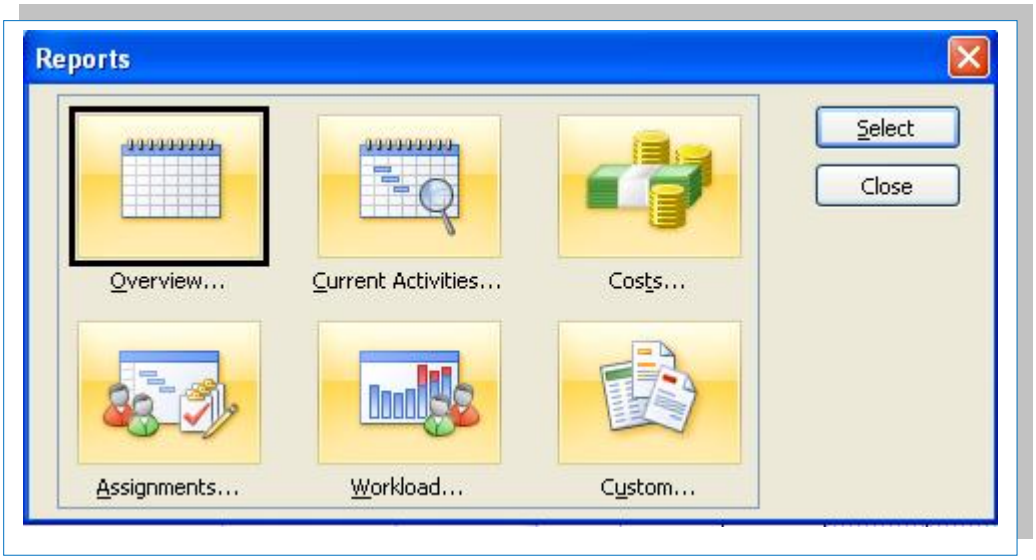
✓ *Vpogled v skupne stroške projekta* ***View -> Gantt Chart.***



Slika 1.17: Stroški

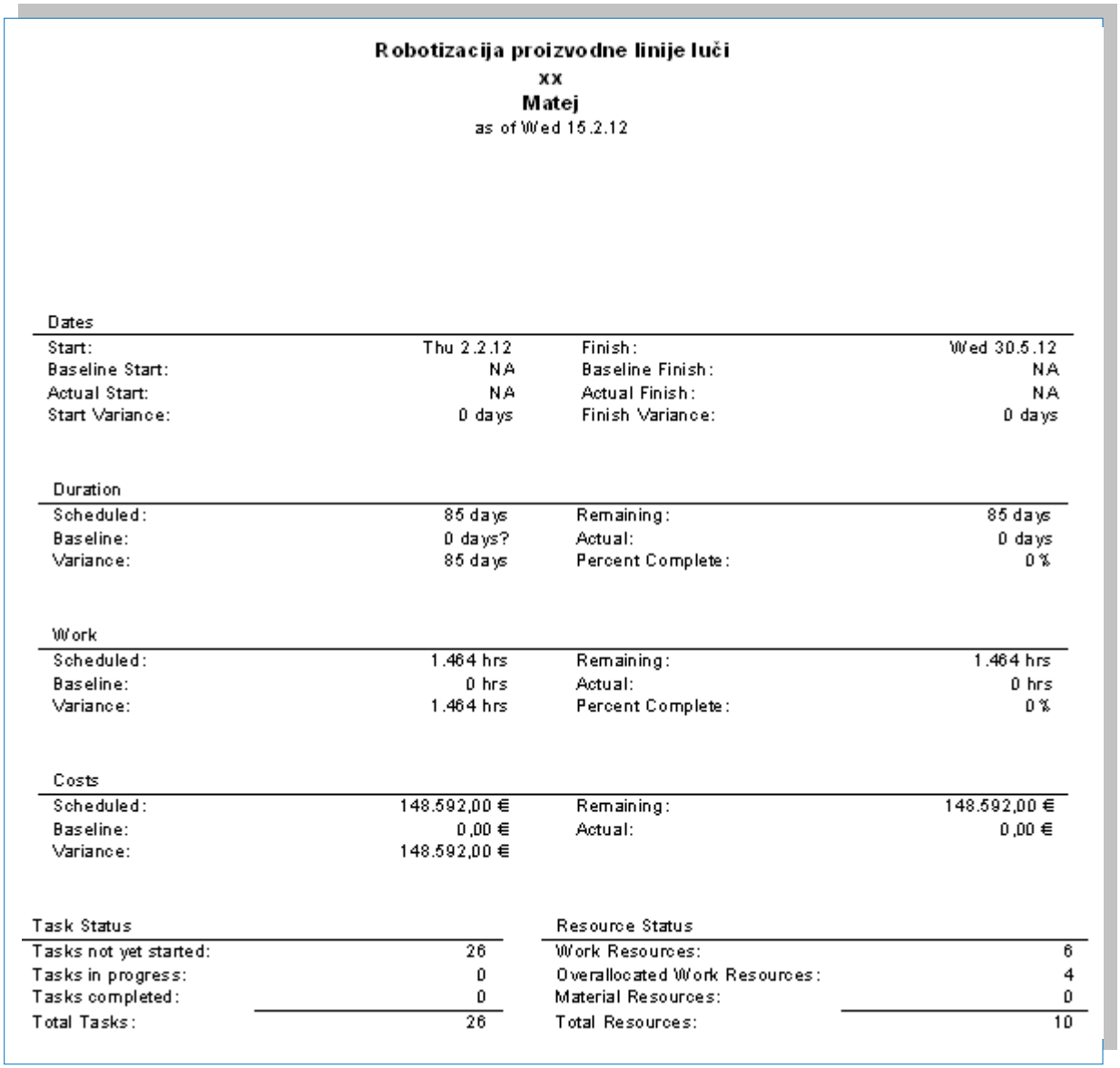
Vir: Lastni

✓ *Poročila o projektu : Overview, Current Activities, Costs itd, ->* ***Report***



Slika 1.18: Poročila o projektu

Vir: Lastni



Robotizacija proizvodne linije luči

xx

Matej

as of Wed 15.2.12

Dates			
Start:	Thu 2.2.12	Finish:	Wed 30.5.12
Baseline Start:	NA	Baseline Finish:	NA
Actual Start:	NA	Actual Finish:	NA
Start Variance:	0 days	Finish Variance:	0 days

Duration			
Scheduled:	85 days	Remaining:	85 days
Baseline:	0 days?	Actual:	0 days
Variance:	85 days	Percent Complete:	0%

Work			
Scheduled:	1.464 hrs	Remaining:	1.464 hrs
Baseline:	0 hrs	Actual:	0 hrs
Variance:	1.464 hrs	Percent Complete:	0%

Costs			
Scheduled:	148.592,00 €	Remaining:	148.592,00 €
Baseline:	0,00 €	Actual:	0,00 €
Variance:	148.592,00 €		

Task Status		Resource Status	
Tasks not yet started:	26	Work Resources:	6
Tasks in progress:	0	Overallocated Work Resources:	4
Tasks completed:	0	Material Resources:	0
Total Tasks:	26	Total Resources:	10

Slika 1.19: Poročilo

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

2 SENZORJI IN MERILNI PRETVORNIKI

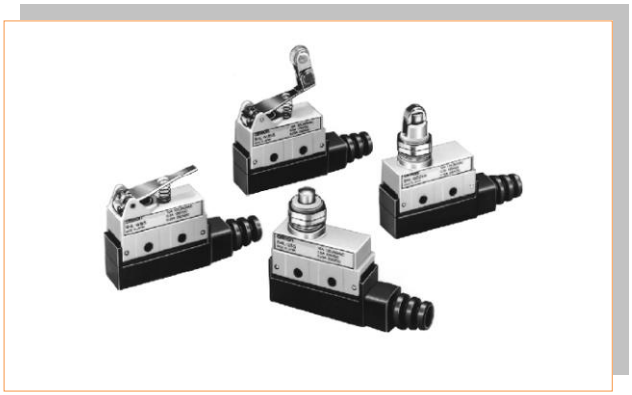
Vloga merilnega sistema je, da zagotovi informacijo za izvršitev povratne regulacijske zanke.

Slika 2.1: Procesiranje merilnih informacij

Vir: Lastni

Predprocesiranje prilagodi vhodni signal na signal primeren za senzor. **Senzor** reagira na fizično simulacijo vhodnega signala in na izhodu poda informacijo merilnemu pretvorniku. Izhodni signal je nato uporabljen pri **poprocesiranju**, kjer se tvori dokončni signal merilnega sistema.

Senzor $\Rightarrow$ meritev fizične vrednosti.

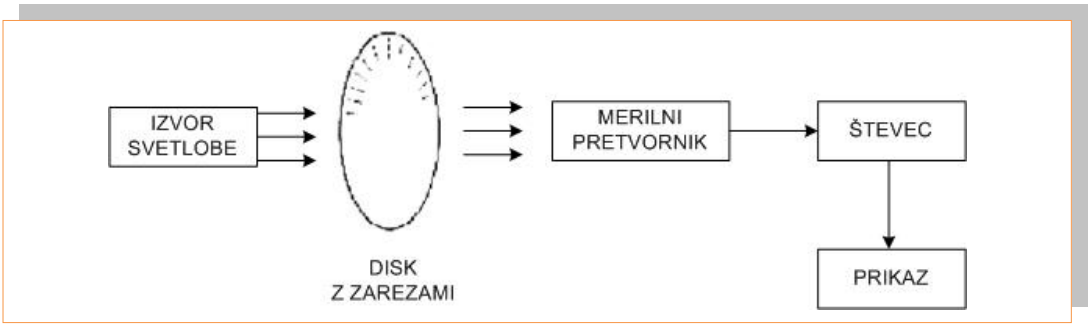


Slika 2.2: Končna stikala, meritev fizične vrednosti

Vir: Lastni

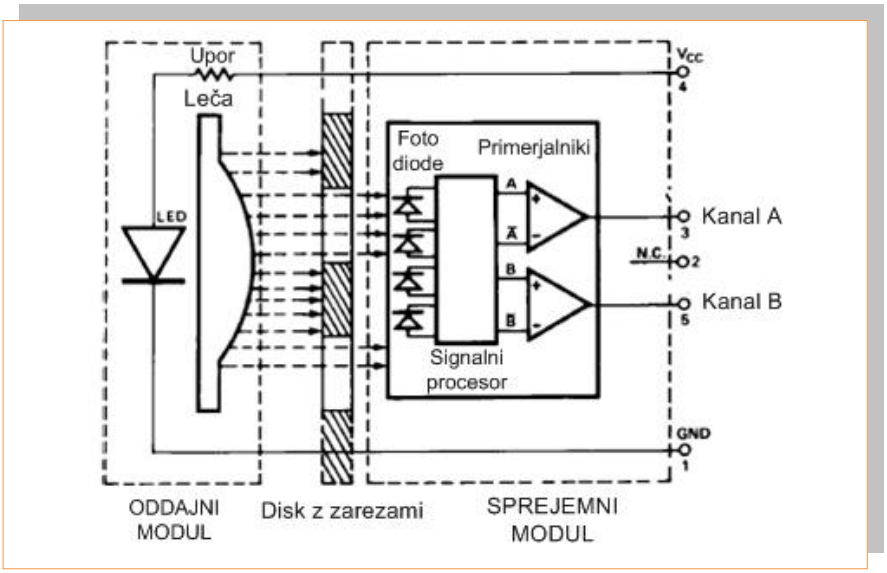
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Merilni pretvornik ⇒ prenaša informacijo v obliki energije iz enega dela merilnega sistema do drugega dela in hkrati pretvarja nosilno energijo informacije.



Slika 2.3: Merjenje položaja z optičnim inkrementalnim dajalnikom položaja

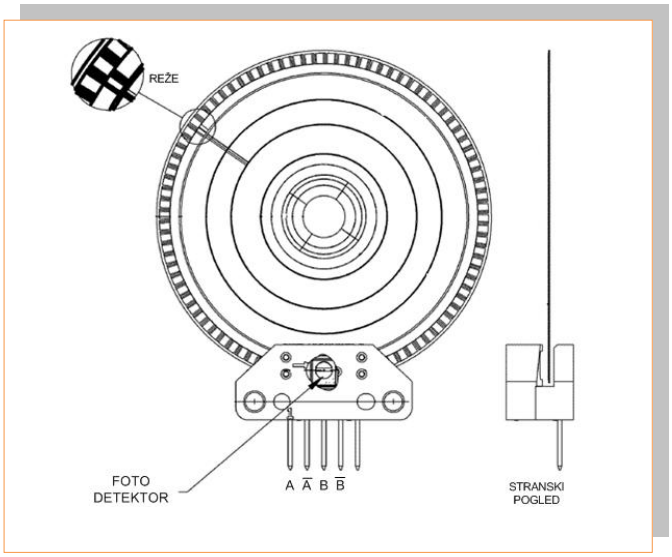
Vir: www.feri.uni-mb.si



Slika 2.4: Princip delovanja optičnega dajalnika stanja (enkoderja)

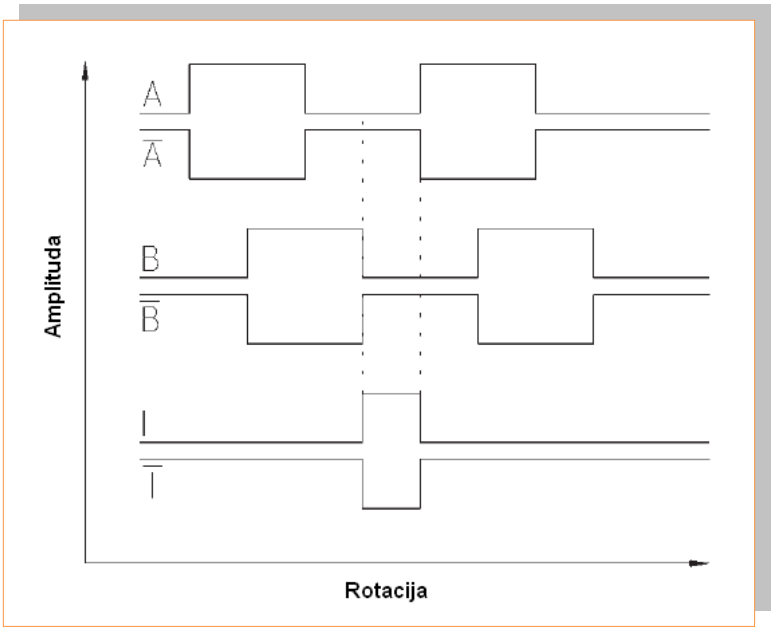
Vir: www.feri.uni-mb.si

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 2.5: Zgradba inkrementalnega dajalnika stanja

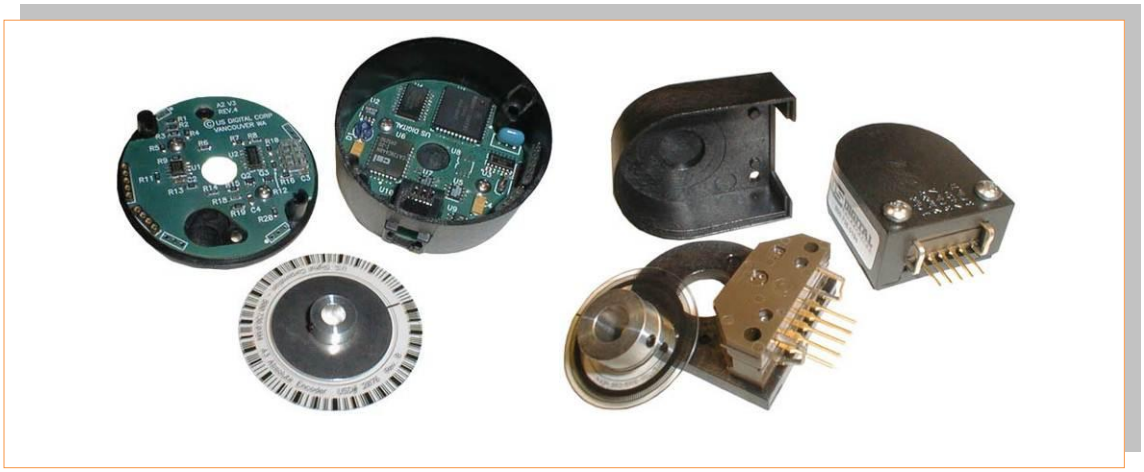
Vir: www.feri.uni-mb.si



Slika 2.6: Izhodni signali inkrementalnega dajalnika stanja

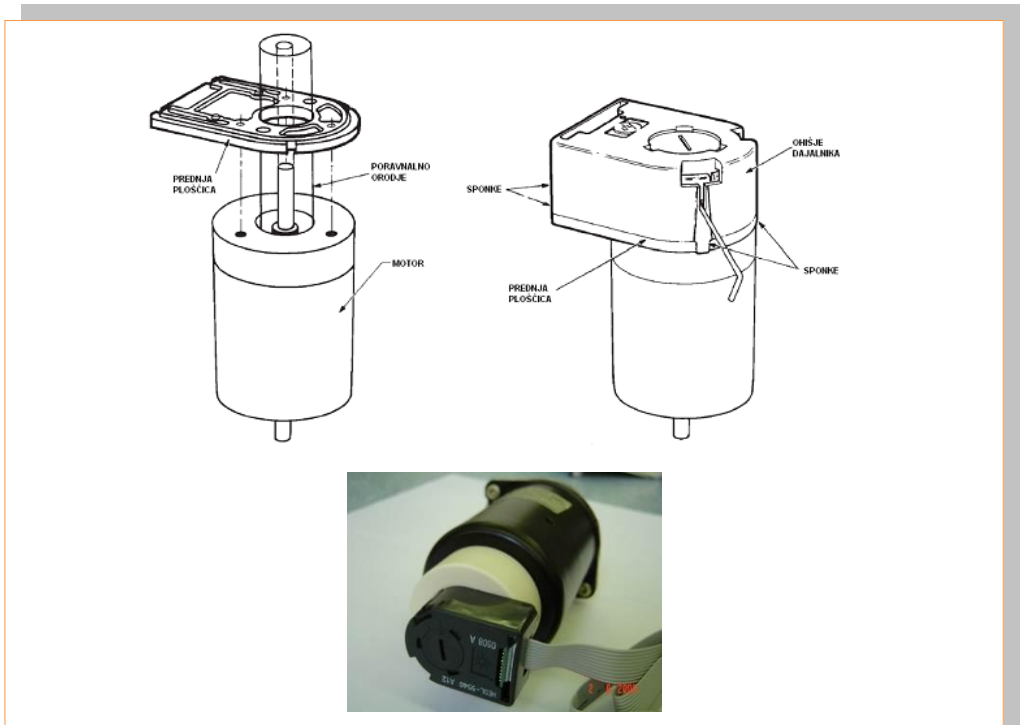
Vir: www.feri.uni-mb.si

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 2.7: Različni tipi optičnih inkrementalnih dajalnikov stanja

Vir: www.feri.uni-mb.si



Slika 2.8: Primer montaže inkrementalnega dajalnika

Vir: www.feri.uni-mb.si

Prenos informacije z merilnim pretvornikom je v bistvu prenos energije. Energija se lahko prenaša **radiacijsko, mehanično, termalno, električno, magnetno ali kemično**. Glede na prenosno energije merilnega pretvornika poznamo direktne (termočlen) in indirektne (aktivne) merilne pretvornike (Hallova sonda).

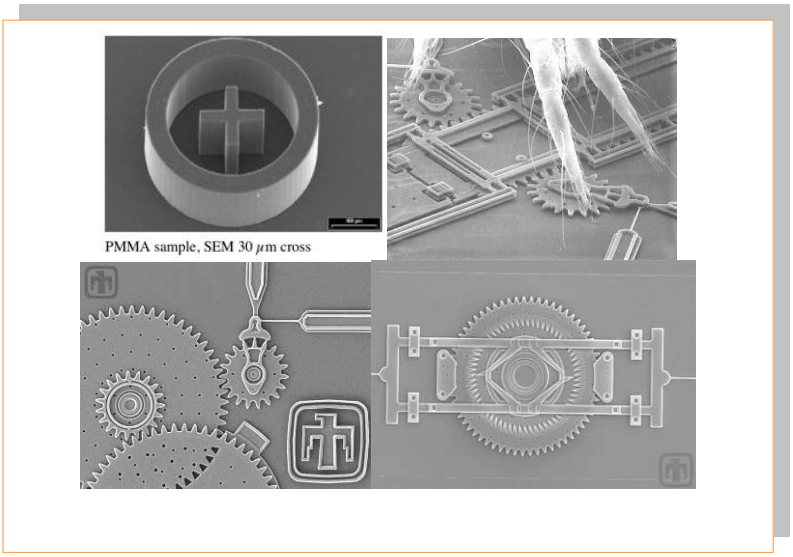
Merilne sisteme lahko razvrstimo glede na :

- **vhod**, ki je lahko katerakoli fizikalna veličina
- **funkcijo,** ki je lahko razdalja, hitrost, pospešek, dimenzija, masa, sila, trdota, viskoznost …;
- **lastnosti** oziroma parametri merilnega sistema (natančnost, ponovljivost, linearnost, občutljivost, področje);
- **Izhod,** ki je lahko analogen, digitalen, frekvenčni in kodiran (hexadecimalno, Grayeva koda).

Razvoj tehnologije merilnih pretvornikov gre v smeri uvajanja polprevodnih merilnih pretvornikov kot so:

- **MEMS sistemi** (mikro elektro mehanski sistemi – micro electrical mechanical system);
- **optični sistemi** (laserji, fotodetektorji , optična vlakna …);
- **piezzo električne naprave** (miniaturni merilniki pospeška);
- **ultrazvočni merilni pretvorniki.**

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



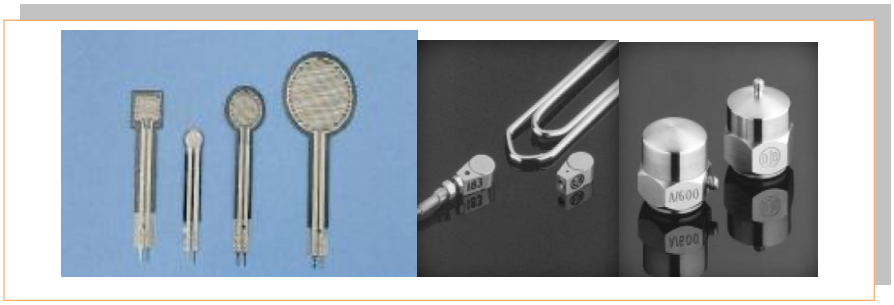
Slika 2.9: Primeri MEMS aktuatorjev , gonila

Vir: www.feri.uni-mb.si

Slika 2.10: Optični vodniki

Vir: www.feri.uni-mb.si

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 2.11: Piezzo merilniki pospeška

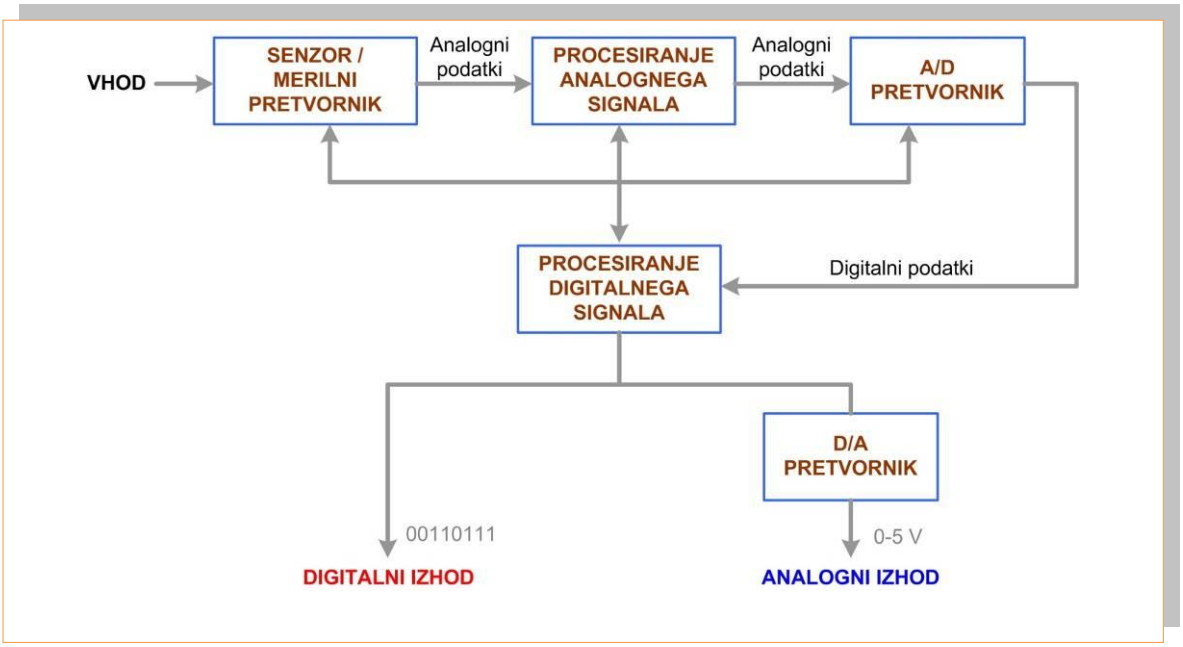
Vir: www.feri.uni-mb.si

Slika 2.12: Ultrazvočni merilniki razdalje

Vir: www.feri.uni-mb.si

Merilni sistemi imajo dodane pred in po procesorske tehnike kar na polprevodniškem integriranem vezju (čipu) merilnega pretvornika ali senzorja. Takšen tip merilnega sistema se imenuje pametni merilni sistem (smart senzor). Na sliki 2.13 je prikazana blokovna shema pametnega senzorja.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



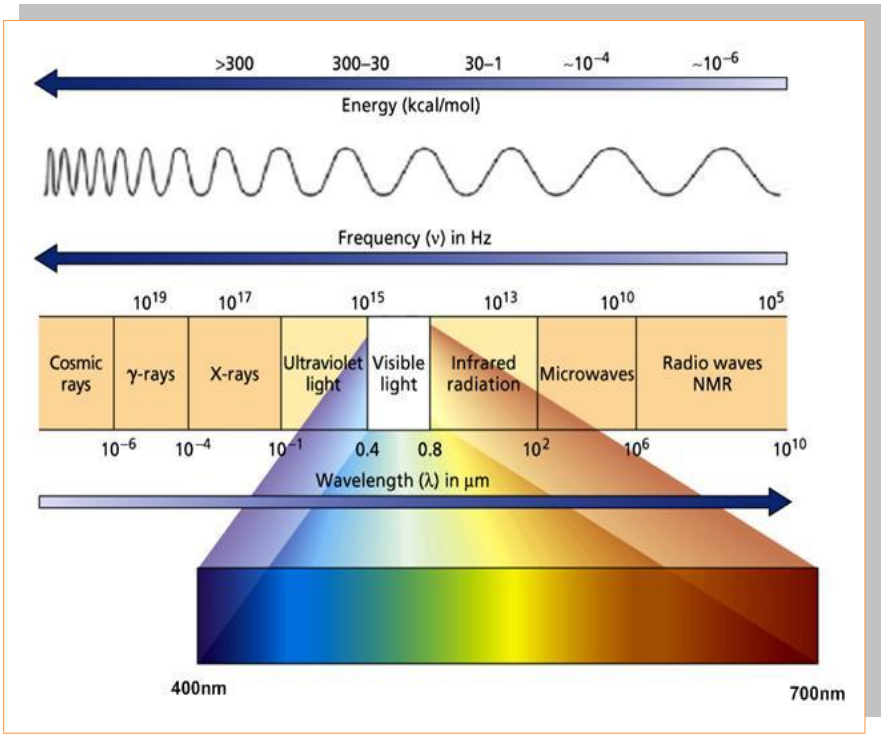
Slika 2.13: Blokovna shema pametnega merilnega sistema

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

OPTIČNI MERILNI SISTEMI

Optični merilni pretvorniki delujejo v elektromagnetnem spektru od infrardečih do ultravioličnih frekvenc (slika 4.2.1).



Slika 2.14: Elektromagnetni spekter

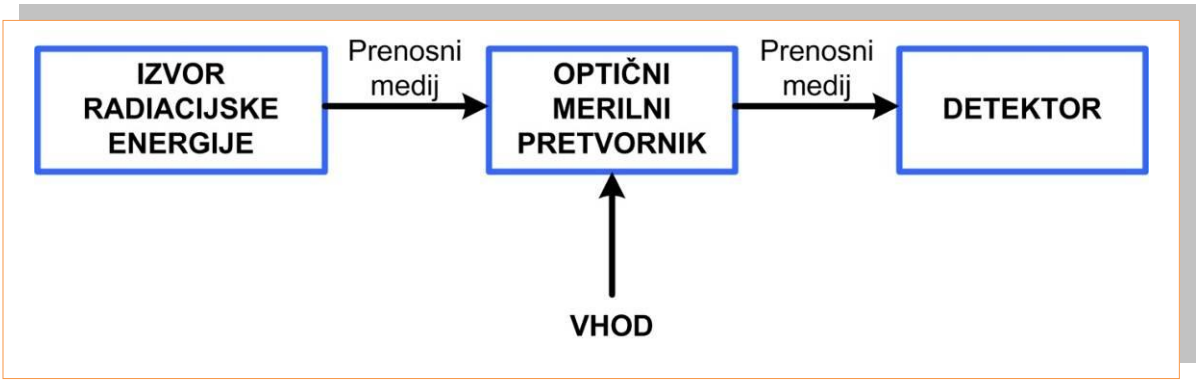
Vir: http://www.kvarkadabra.net/mediagallery/media.php?s=20060326214527911

Optični merilni sistemi lahko brez problemov delujejo v območjih z visokimi temperaturami in v območjih z visoko vsebnostjo elektromagnetnih motenj.

Elementi optičnega merilnega sistema:

- **Izvor** svetlobne energije;
- **Prenosni sistem,** ki usmerja svetlobno energijo;
- **Merilni pretvornik**, ki pretvori svetlobno energijo primerno obliko energije.
- **Detektor,** ki zaznava spremembe energije iz merilnega pretvornika

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 2.15: Blokovna shema optičnega merilnega sistema

Vir: Lastni

Poznamo naslednje vrste izvorov radiacijske energije:

- **Žarnice z žarilno nitko**, ki so najcenejši vir svetlobe. Slabost : kratka življenjska doba in termična omejenost.
- **Flourescentni viri.**
- **LED diode**, cenen venfar šibek vir svetlobe z ozkimi spektralnimi pasovi v vidnem in infrardečem spektru.
- **Laserski viri** imajo visok energijski izvor svetlobe (mW-kW). Glavna lastnost laserskih virov je koherentnost oziroma paralelnost žarkov.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

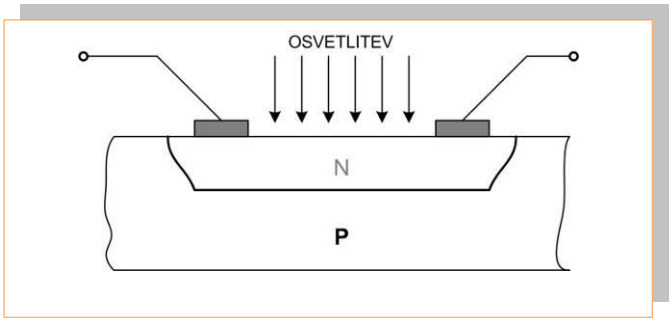
FOTODETEKTORJI

V osnovi ločimo termične fotodetektorje in kvantne fotodetektorje. **Termični** fotodetektorji delujejo na principu toplote sproščene na fotodetektorju. Toploto proizvede svetlobna energija, ki obsije fotodetektor. Njihova slabost je počasen odziv na spremembo svetlobe.

Kvantni fotodetektorji so narejeni iz polprevodniških slicijevih rezin (PN spoj). Slabost kvantnih detektorjev je, da so temperaturno odvisni in zelo občutljivi na elektromagnetne šume iz okolice.

Poznamo več vrst kvantnih fotodetektorjev:

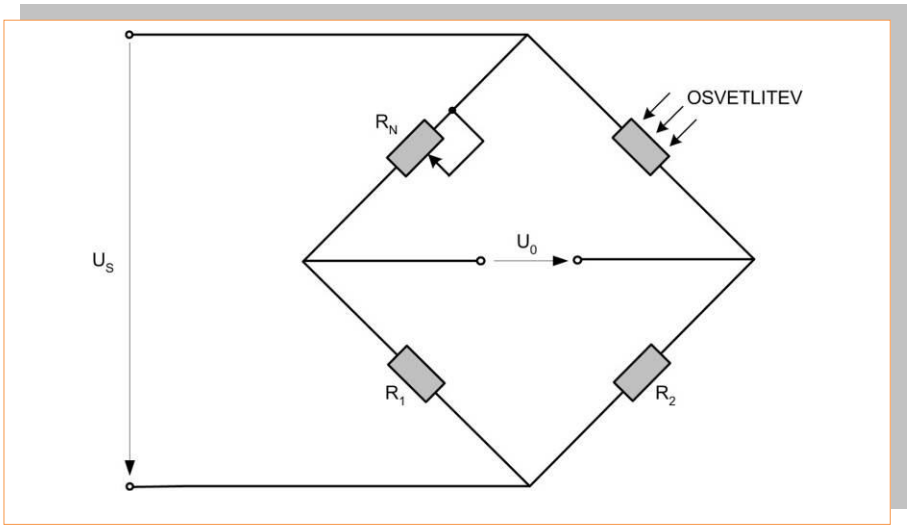
- **Fotoprevodni detektor** je sestavljen iz P-sloja na katerega je nanesena tanka plast N-sloja. V primeru ko N-sloj osvetlimo se spremeni prevodnost N-sloja in s tem padec napetosti na spoju. Fotoupor je pasivni el. element. Smer el. toka določa polariteta priključne napetosti. Fotoupori so ceneni in občutljivi v vidnem svetlobnem spektru (npr v temi upornost 100 kΩ, osvetljen 10 Ω,) hkrati pa počasni. Zato se uporabljajo za enostavna el.vezja, meritve svetlobe, varnostne naprave…)



Slika 2.16: Struktura fotoprevodnega detektorja

Vir: Lastni

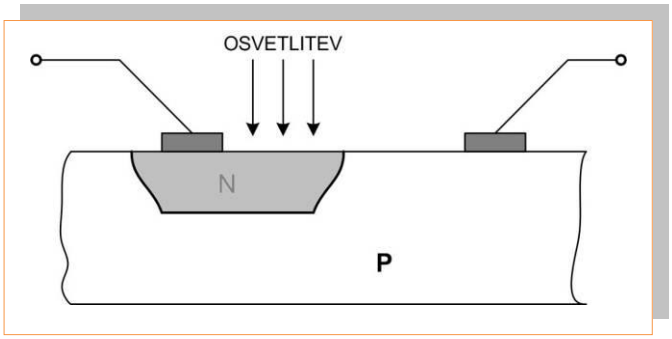
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 2.17: Mostično vezje za merjenje svetlobe

Vir: Lastni

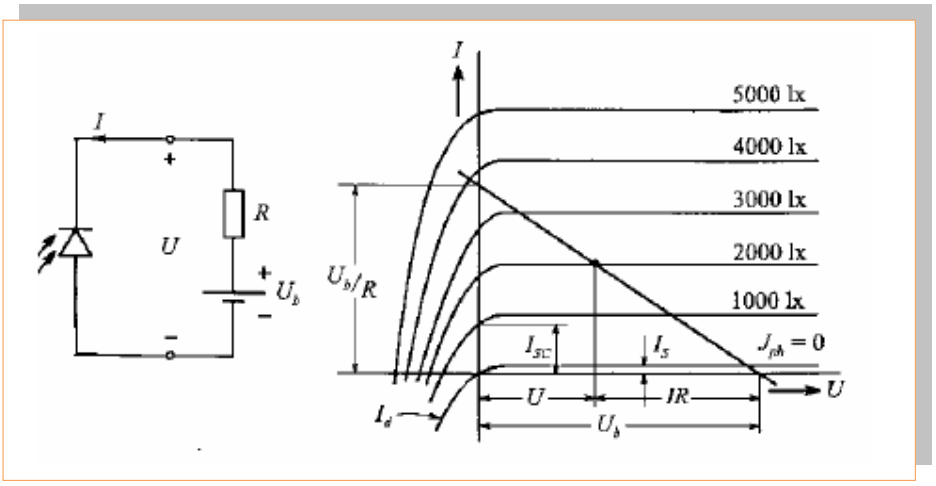
- **Fotodioda** je sestavljena iz PN spoja, ki je polariziran obratno kot fotovoltni detektor. PN spoj loči svetlobno generirane elektrone od vrzeli. Ko je fotodioda osvetljena prepušča tok in obratno ko ni osvetljena. Napetost in tok se torej spreminjata v odvisnosti od osvetlitve.



Slika 2.18: Zgradba fotodiode

Vir: Lastni

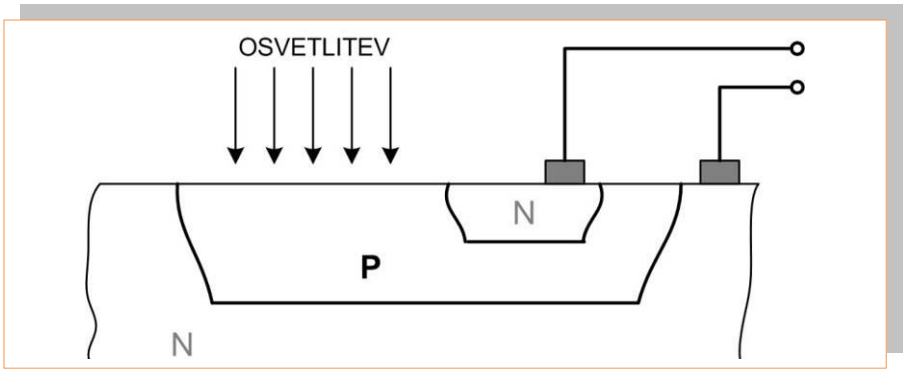
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 2.19: Izhodna karakteristika PN fotodiode

Vir: www.feri.uni-mb.si

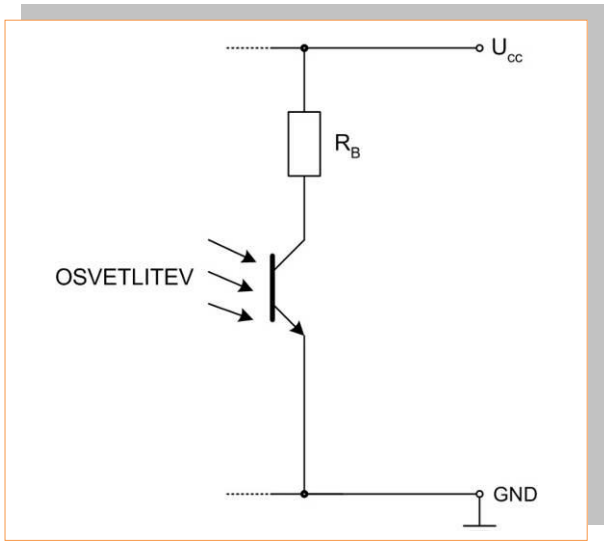
- **Fototranzistor** je nadgradnja fotodiode, ki ji dodamo N-sloj. Če je bazni P-sloj osvetljen, steče tok med kolektorjem in emiterjem. V primeru, da bazni sloj ni osvetljen pa je fototranzistor zaprt.



Slika 2.20: Zgradba fototranzistroja

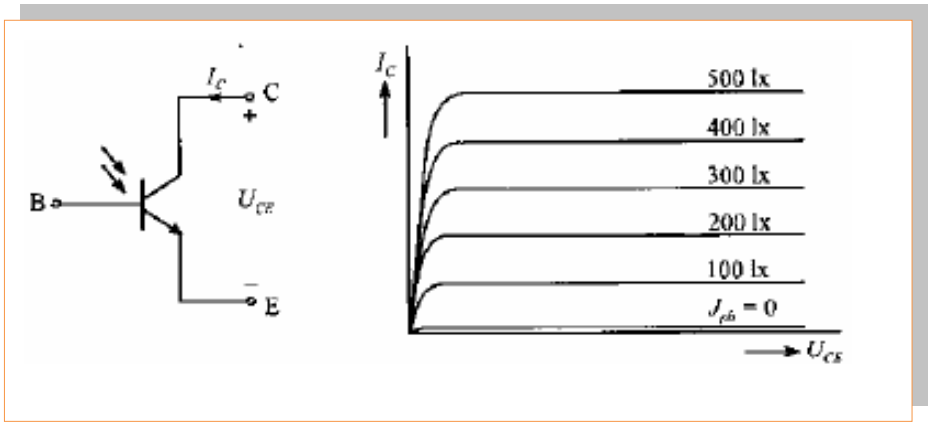
Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 2.21: Primer vezave fototranzistorja

Vir: Lastni



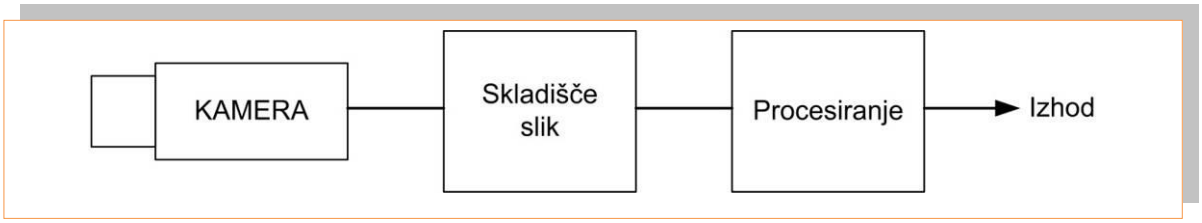
Slika 2.22: Izhodna karakteristika fototranzistorja

Vir: Lastni

- **Fotovoltni detektor**, ki spreminja vhodno izhodno (I/U) karakteristiko v odvisnosti od osvetljenosti n-plasti spoja.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

SISTEMI UMETNEGA VIDA S KAMERO



Slika 2.23: Osnovni sistem umetnega vida

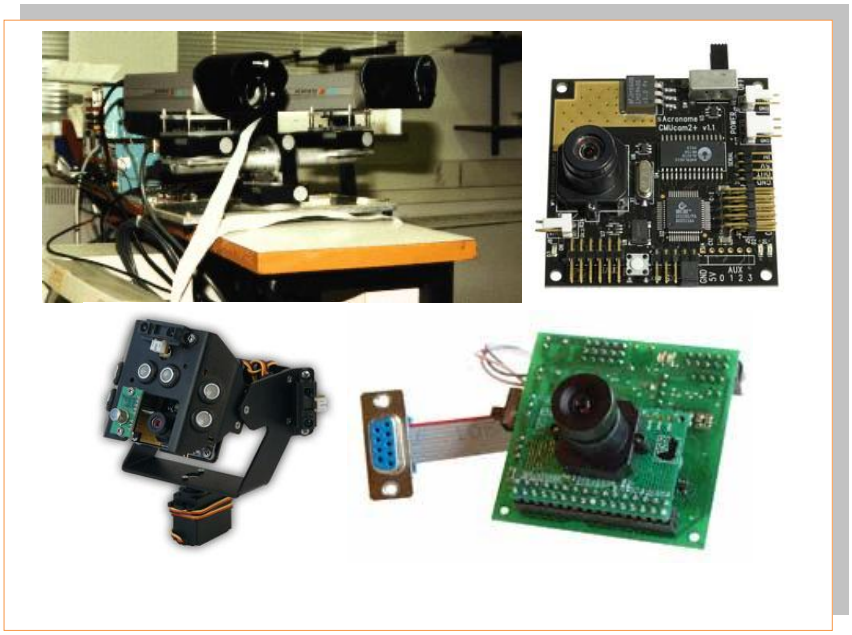
Vir: Lastni

Kamera sliko zajema in jo oblikuje. Skladišče slik (frame grabber) pretvori serijo slik v digitalno obliko in jih shrani. Modul procesiranja pa slike obdela in analizira glede na zahtevano nalogo (prepoznava vzorcev slike). Torej modul procesiranja iz slike potegne želeno informacijo.

Procesiranje se izvede v več korakih:

- Pred-procesiranje ⇒ izboljšava zajete in popačen slike.
- Ekstrakcija lika ⇒ detekcija roba lika.
- Analiza ⇒ izračun površine, centra, razpoznava vzorcev, npr n-tulpe tehnika, ki ugotovi neznani vzorec med znanimi.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



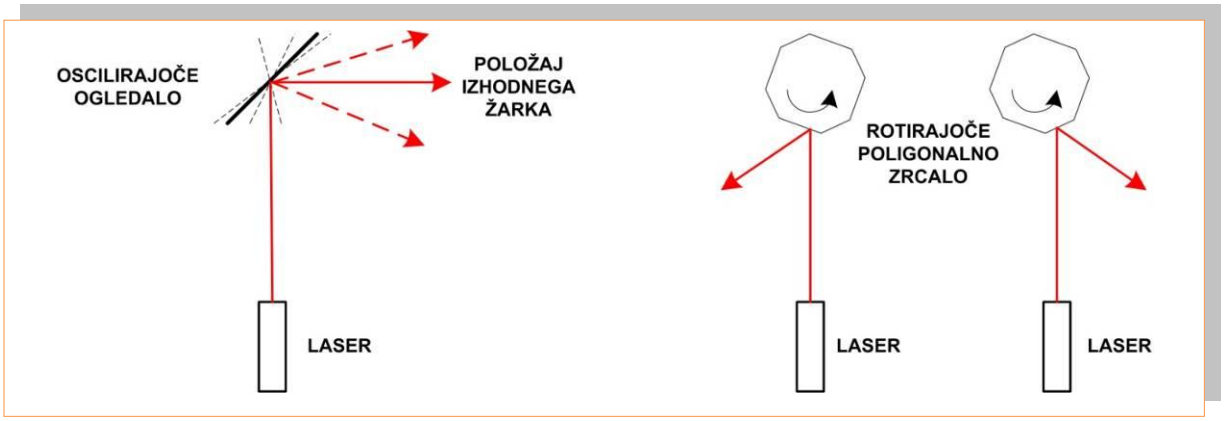
Slika 2.24: Primer sistema umetnega vida

Vir: www.ni.com

RAZPOZNAVA Z LASERJEM

Pri sistemih umetnega vida s kamero, kamera zajema sliko osvetljene površine. Pri sistemih, ki skenirajo z laserjem pa laser oddaja ozek laserski žarek, ki samo v določeni točki osvetli objekt. Odbiti laserski žarek pa se zajame s široko področnim detektorjem. Prednost pred zajemanjem s kamero je, da laser ne potrebuje svetlobe okolja, ampak si generira sam. Skeniranje (razpoznava) ima velko resolucijo (5000 do 30000 točk), hitrost skeniranja pa je preko 10^8 točk na sekundo. Linije skeniranja se dosežejo z zapletenim sistemom prizem in ogledal. Za enostavne sisteme (malo točk in majhne hitrosti skeniranja) se uporabljajo oscilirajoča ravna ogledala. Pri večjih hitrostih skeniranja in večji resoluciji se uporabljajo rotirajoča poligonalna ogledala.

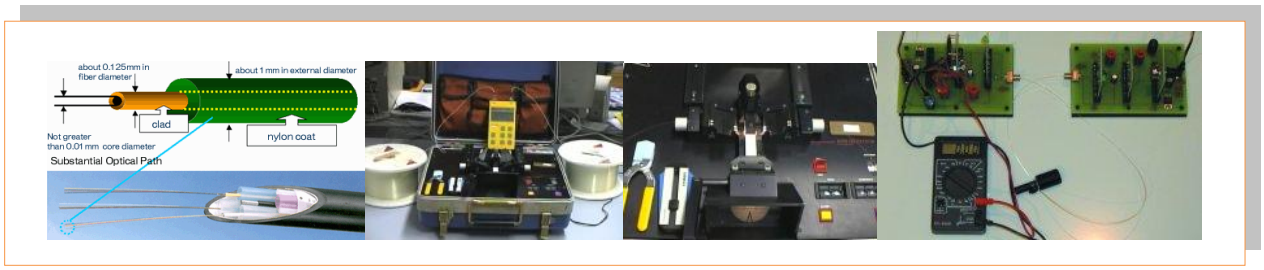
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 2.25: Oscilirajoče zrcalo in rotirajoče poligonalno zrcalo

Vir: Lastni

MERILNI PRETVORNIKI Z OPTIČNIMI VLAKNI



Slika 2.26: Optično vlakno in naprava za varjenje vlaken

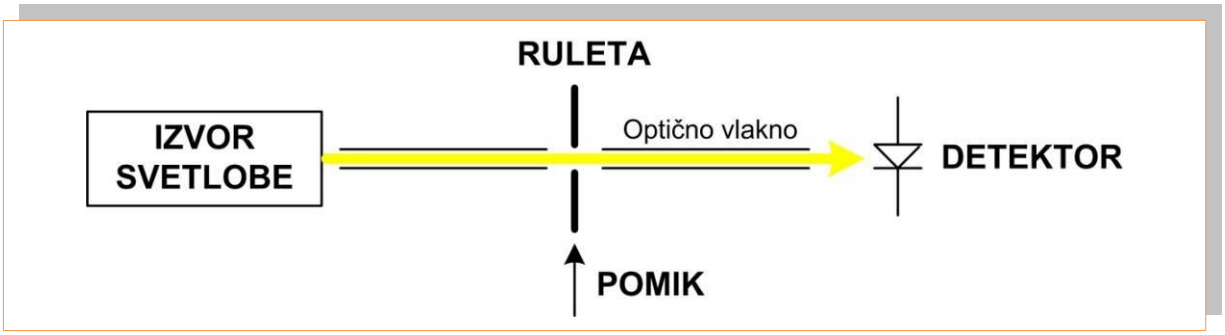
Vir: Lastni

V merilnih sistemih z optičnimi vlakni je nivo motenj zelo nizek, ker elektromagnetna valovanja ne vplivajo na prenos svetlobe v optičnem vlaknu. Optična vlakna se uporabljajo na dva načina :

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

- **Modulacija intenzivnosti prepuščene svetlobe skozi vlakno**

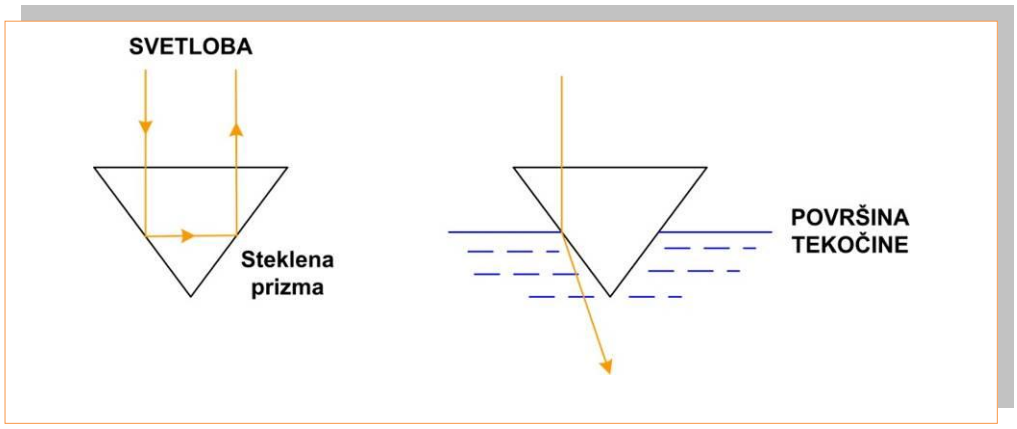
Če med dva optična vlakna namestimo premikajočo ruleto, se spremeni intenzivnost svetlobe skozi vlakna. Tako lahko merimo pomik rulete in s tem posredno majhne pomike.



Slika 2.27: Detektor pomika z optičnim vlaknom

Vir: Lastni

Izkoristimo lahko tudi lom svetlobnega žarka na meji steklene prizme in zraka ali tekočine. Na tem principu deluje detektor nivoja tekočine.

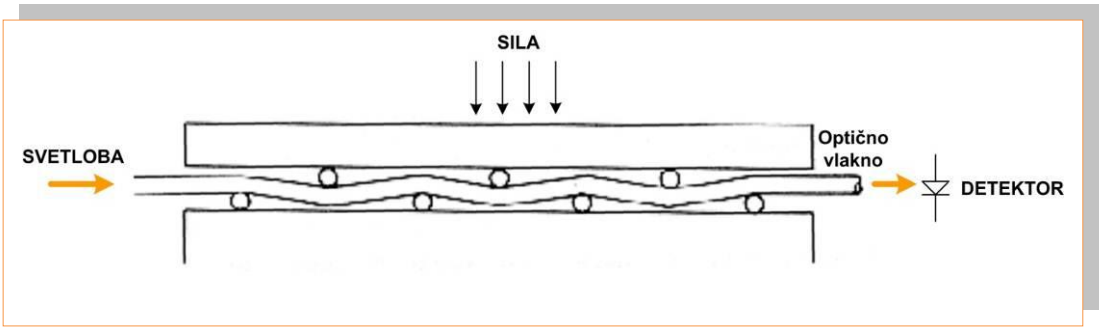


Slika 2.28: Optični detektor nivoja tekočine

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

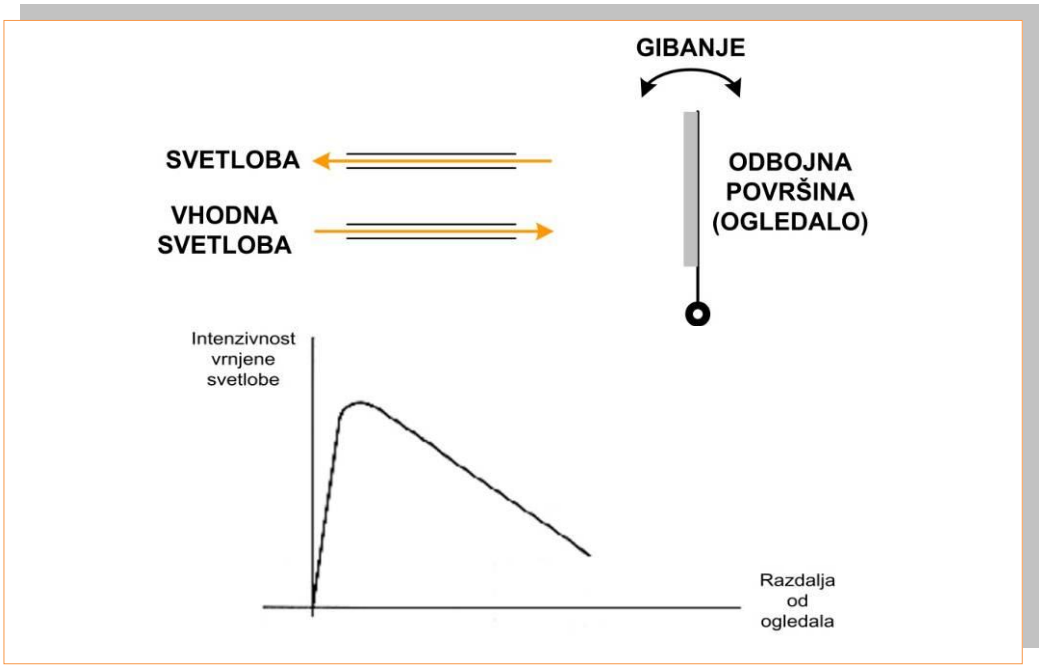
Mikroupogibanje optičnega vlakna je primerno za meritev sile na senzor. Z upogibanjem optičnega vlakna se spreminja propustnost svetlobe skozi vlakno. Sila je sorazmerna prepuščeni svetlobi skozi vlakno.



Slika 2.29: Optični mikroupogibni merilnik sile

Vir: Lastni

Odboj svetlobe lahko uporabimo pri merjenju rotacije okoli osi:



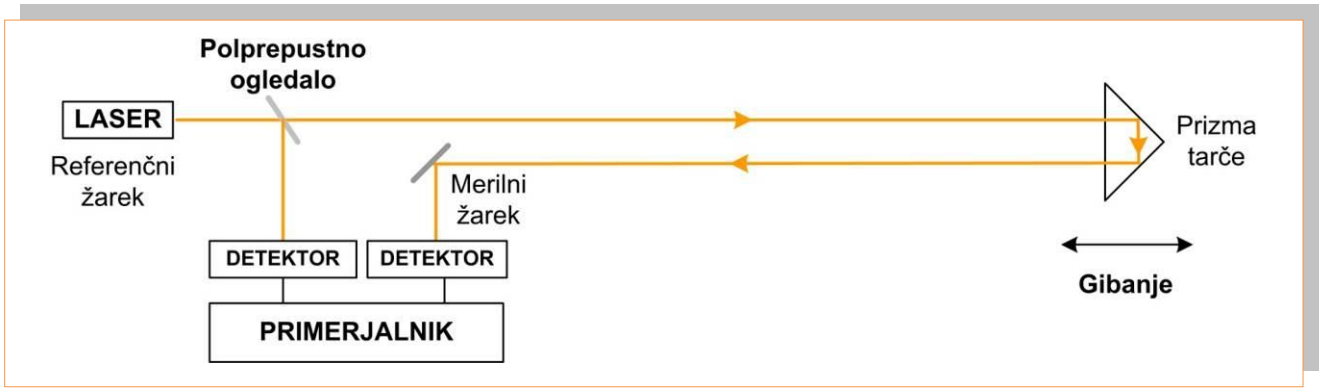
Slika 2.30: Merjenje položaja rotacije z odbito svetlobo

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

- **Modulacija faze svetlobe**

Modulacija faze svetlobe se uporablja za natančne meritve razdalje. Laserska svetloba se deloma odbije od pol prepustnega srebrnega ogledala, deloma pa potuje nemoteno naprej. Ne odbiti žarek se odbije od prizme in se vrne do detektorja faze, kamor se usmeri tudi odbiti del laserskega žarka. Z meritvijo faznega pomika obeh žarkov lahko direktno izračunamo oddaljenost prizme od izvora laserskega žarka.

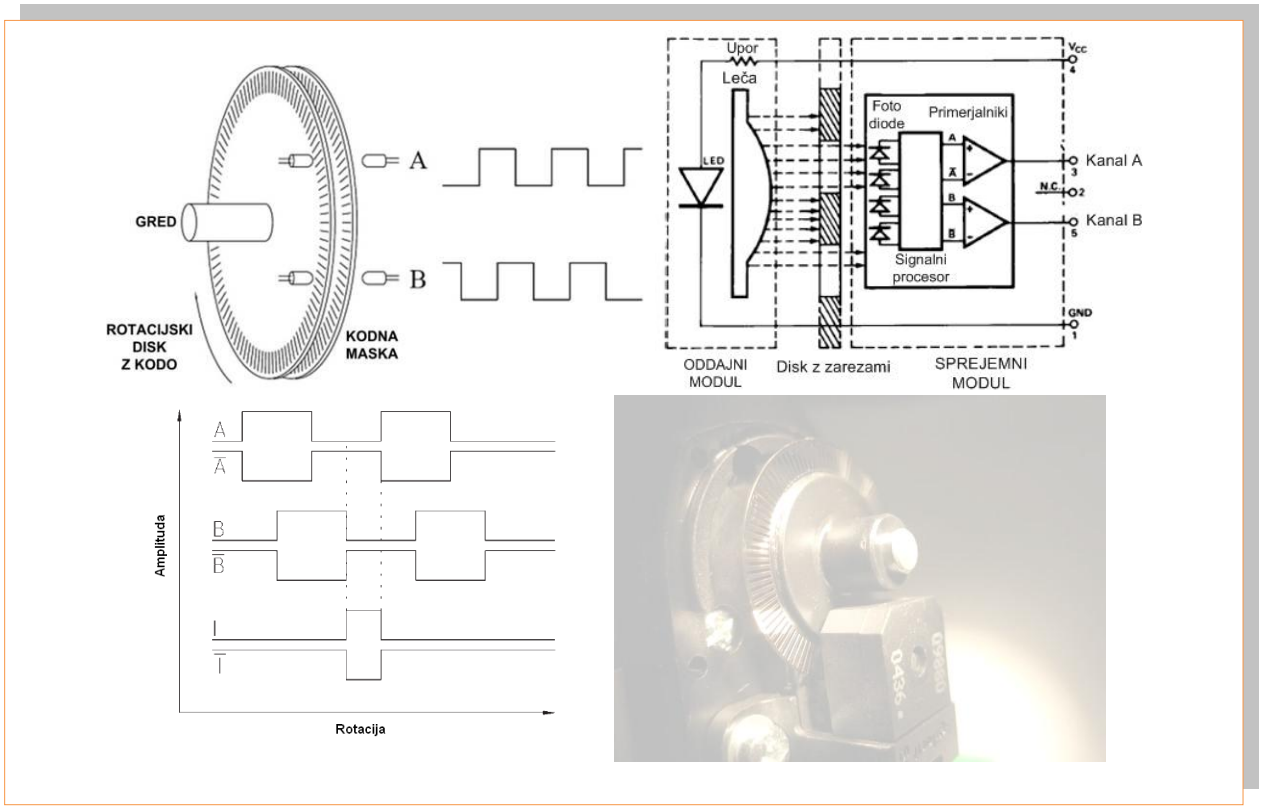


Slika 2.31: Interferometer, princip merjenja razdalje

Vir: Lastni

OPTIČNI KODIRNIKI

Osnove delovanja inkrementalnega dajalnika položaja smo spoznali na strani 28-31. **Inkrementalni dajalniki položaja** so cenen pripomoček za merjenje rotacijskega in linearnega pomika. Kot smo spoznali so rotacijski dajalniki zgrajeni iz diska z režami, vira svetlobe in na drugi strani fotodetektorja, ki proizvaja napetostni signal (izhodni pulzi) v odvisnosti od prepuščene svetlobe skozi reže. Izhodna signala sta zamaknjena zaradi ugotavljanja smeri vrtenja diska.



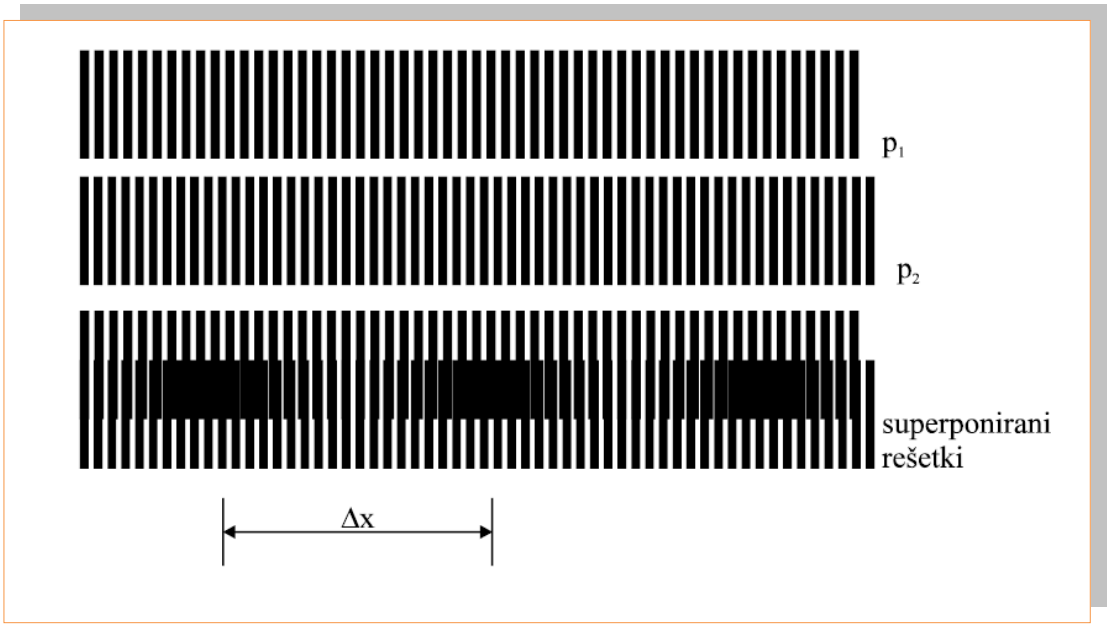
Slika 2.32: Optični kodirnik oz. inkrementalni dajalnik položaja

Vir: www.feri.uni-mb.si

Če pride signal A pred signalom B iz logične »0« na logično »1«, tedaj imamo pozitivno smer gibanja in obratno ko je na izhodu B logična »1« pred A logičnim izhodnim

signalom. Digitalno vezje, ki ugotavlja smer gibanja, daje signal digitalnemu števcu pulzov, ki pulze prišteva (pozitivna smer vrtenja) in odšteva (negativna smer vrtenja). Na ta način lahko merimo kot zasuka na +\- 1 kvant (črtica, pulz) informacije natančno. Inkrementalni dajalniki lahko imajo nekaj sto črtic na obrat ter do nekaj sto tisoč črtic na obrat. V odvisnosti od aplikacije in zahtev po natančnem vodenju izberemo primerni optični kodirnik.

Linearni inkrementalni dajalniki delujejo na principu Moirejevih resic. To sta dve plošči z resicami, ki se pomikata relativno druga na drugo. Tako se tvori vzorec črtic, ki prepuščajo svetlobo. To spremembo zaznava detektor, ki daje na izhodu primerne signale. Linearni inkrementalni dajalniki imajo natančnost 1μm pri dolžini preko 1m.



Slika 2.33: Princip Moirejevega interferometra – linearnega dajalnika položaja

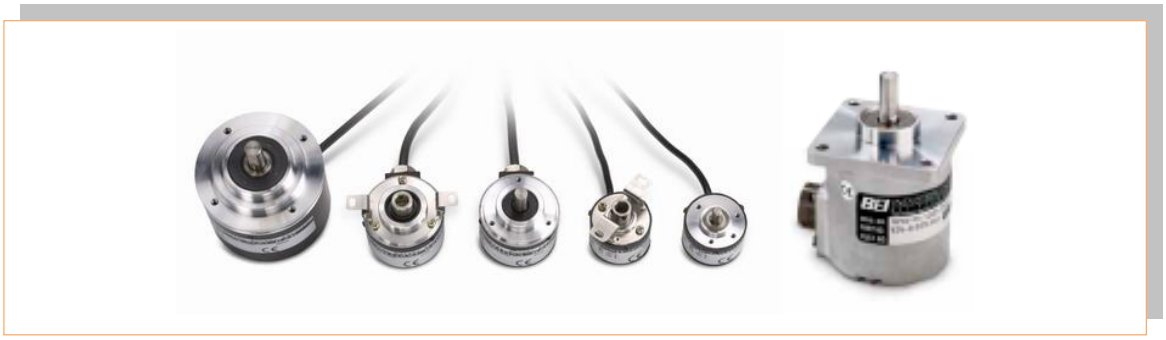
Vir: Lastni

Podrobnosti na :

http://lpa.feri.uni-mb.si/~marjan/ped/senzor_t_1/moire.html

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Absolutni dajalniki položaja generirajo na izhodu absolutni položaj (Ta lastnost je koristna ob izpadih električne energije), medtem ko inkrementalni dajalniki dajejo le relativni položaj glede na referenčni signal (ob izpadu el. energije se števec pulzev briše). Slabost je v primeru kose disk zavrti za več kot en obrat. V tem primeru je potreben števec obratov.

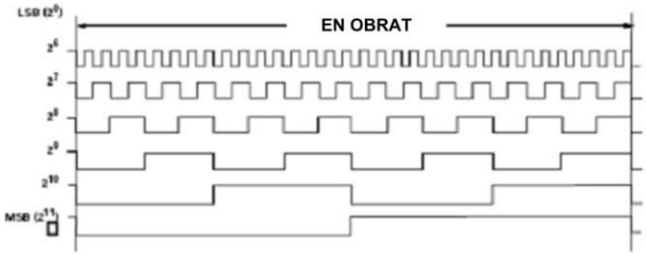


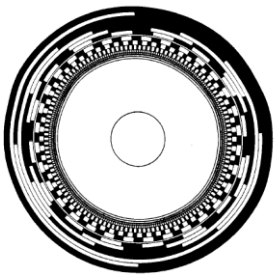
Slika 2.34: Absolutni dajalniki položaja

Vir: http://www.directindustry.com/industrial-manufacturer/optical-encoder-65687-_3.html

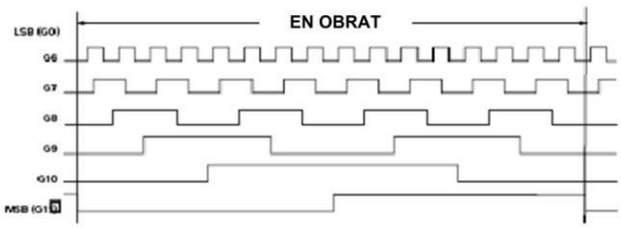
Poznamo dva načina aboslutnega kodiranja:

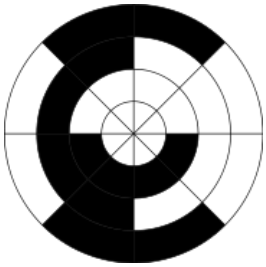
- Kodiranje z **binarno kodo**





- Kodiranje **z Grayevo kodo**





Prednost Grayeve kode je, da se naenkrat spremeni samo en bit informacije. To zmanjšuje možnost napake pri meritvi, posebno pri velikih hitrostih vrtenja diska. Poznamo enkoderje za en zasuk in enkoderje za več zasukov diska.

Dec	Gray	Binarna koda	Dec	Gray	Binarna koda
0	000	000	4	110	100
1	001	001	5	111	101
2	011	010	6	101	110
3	010	011	7	100	111

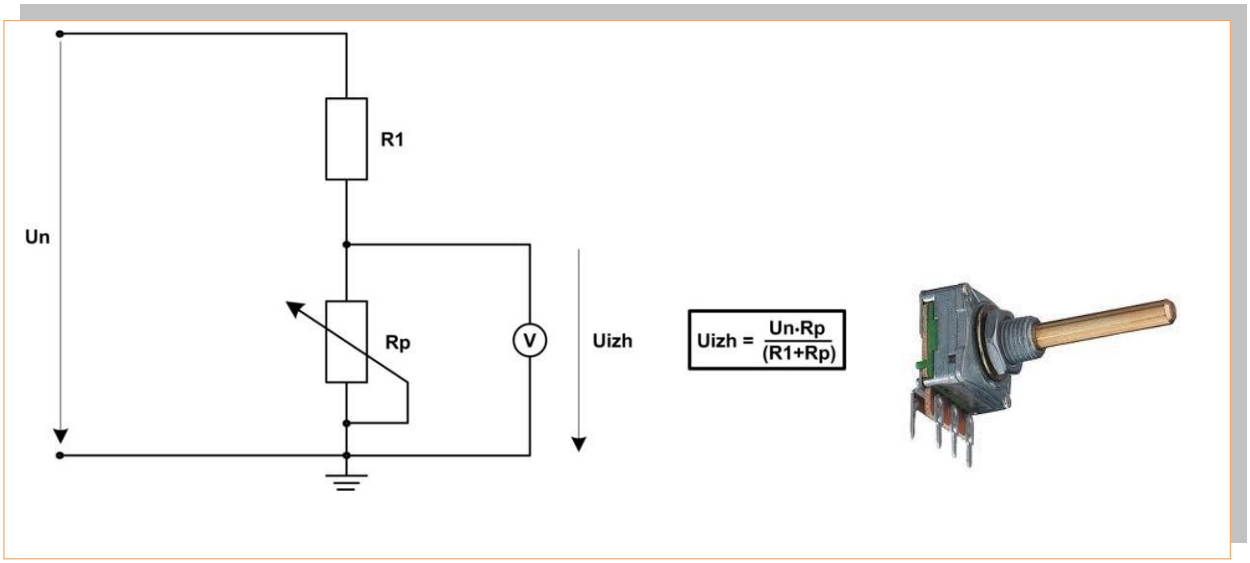
Triangulacija je merilna metoda, ki se uporablja za merjenje razdalje v območjih vsaj nekaj sto metrov. Deluje na principu dveh laserskih izvorov s senzorji odbite svetlobe, ki merita kot odbite svetlobe. Ob znani razdalji med laserskima izvoroma, lahko izračunamo razdaljo do tarče.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

UPOROVNI MERILNI PRETVORNIKI

Potenciometri

Potenciometri so najpreprostejši in najcenejši uporovni merilni pretvorniki. Uporabljajo za merjenje linearnega in rotacijskega pomika. Slaba lastnost uporovnih merilnih pretvornikov je vpliv spremembe temperature na meritev.



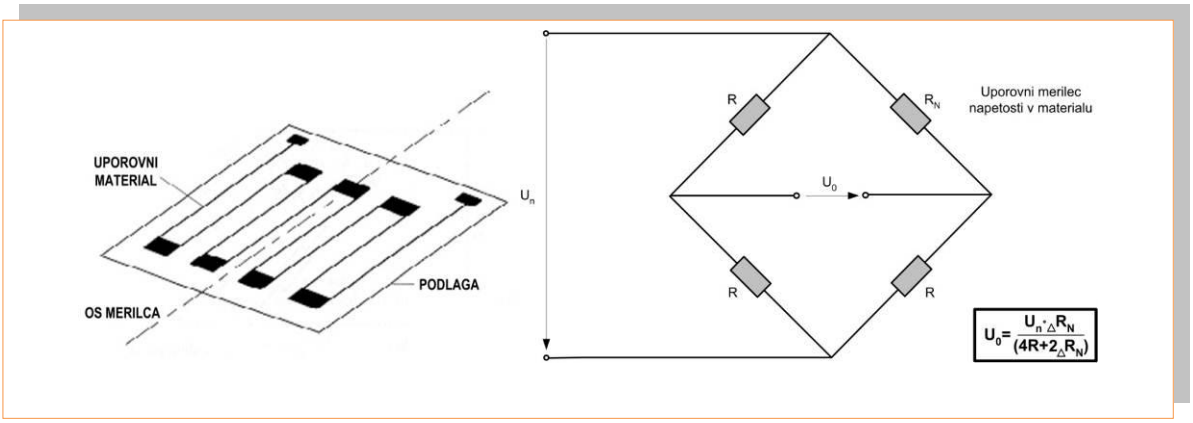
Slika 2.35: Vezava potenciometra in vizualni izgled

Vir: Lastni

Uporovni merilec napetosti materiala (Strain gauge)

Je sestavljen iz uporovne folije, ki je nanesena na podlago. Če sila deluje na os merilca napetosti, se podlaga in s tem uporovna folija nekoliko deformira. Tako se spremeni upornost folije in posledično padec napetosti, ki jo merimo. Običajno se sprememba upornosti meri s pomočjo mostičnega vezja.

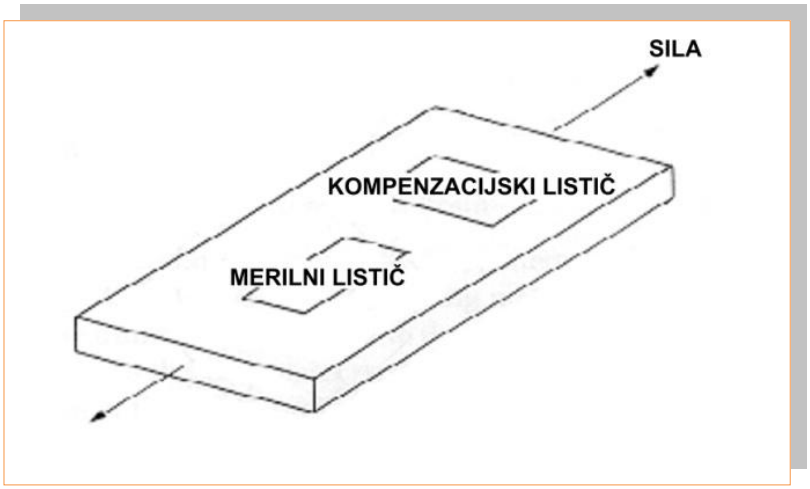
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 2.36: Merilni list in princip merjenja

Vir: Lastni

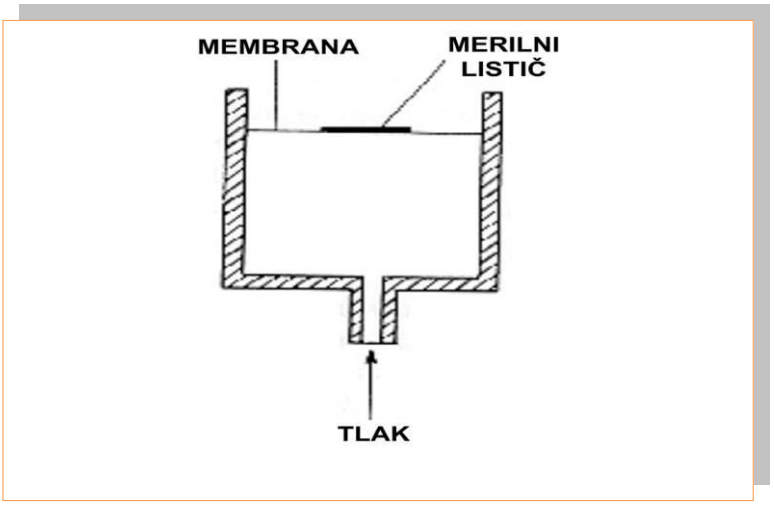
Slabost uporovnih merilcev napetosti v materialu je, da so zelo temperaturno odvisni. Zato moramo pri natančnih meritvah kompenzirati popačitev merilnih signalov zaradi temperaturnih sprememb.. Kompenzacija se izvede s kompenzacijskim mostičnim vezjem. Kompenzacijski merilnik napetosti v materialu je nameščen nam merilnem mestu z osjo meritv, ki je pravokotna na os merilca napetosti v materialu. Kadar os meritve ni znana, tedaj se merilni lističi namestijo v obliki rozete. Merilni lističi se lahko uporabijo za merjenje tlaka ali sile.



Slika 2.37: Kompenzacija temperature

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

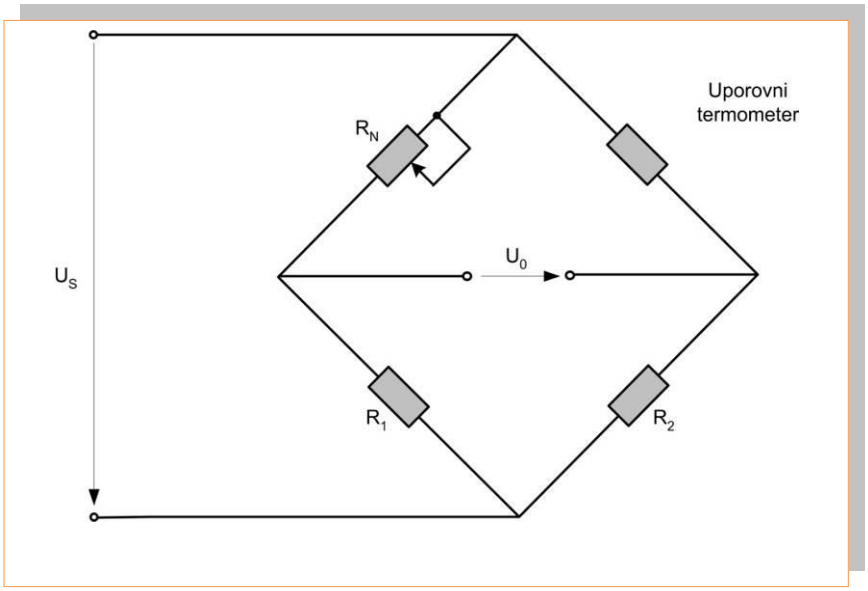


Slika 2.38: Merjenje tlaka z uporovnim lističem

Vir: Lastni

Uporovni temperaturni merilni pretvorniki

Uporovni termometer je merilni pretvornik izdelan iz dolge tanke platinaste žice, ki je navita in vlita v zaščitno ohišje. Uporovnemu termometru se upornost spreminja v odvisnosti od temperature. Običajno je vezan v mostično vezje (slika 4.7.4).

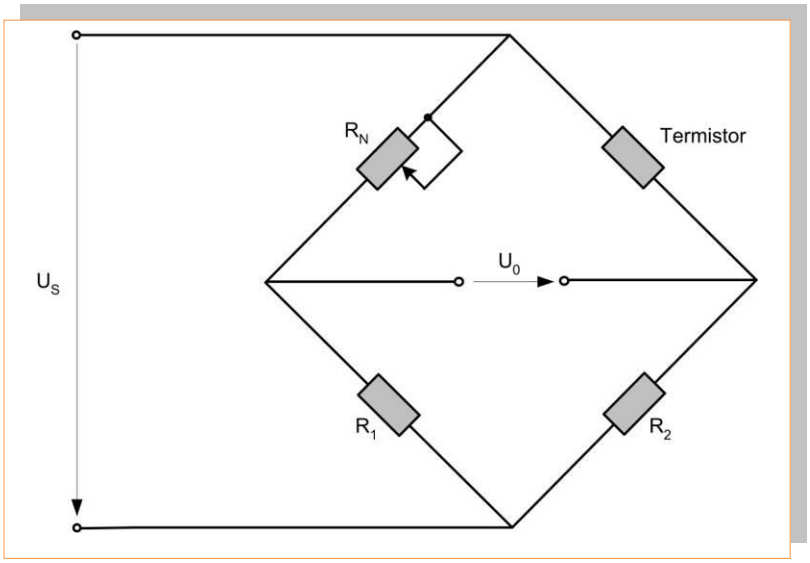


Slika 2.39: Uporovni termometer v merilnem mostiču

Vir: Lastni

Termistor je upor sestavljen iz polprevodniškega materiala. Uporablja se za temperaturna območja od -30° do +200° C. Slabost termistorjev je, da se grejejo zaradi toka, ki teče skozi njih. Običajno vežemo termistor v merilni mostič in merimo razliko potencialov zaradi spremembe padca napetosti zaradi spremembe upornosti termistorja (zaradi spremembe temperature)

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 2.40: Termistor v merilnem mostiču

Vir: Lastni

KAPACITIVNI MERILNI PRETVORNIKI

Kapacitivnost kondenzatorja sestavljenega iz dveh paralelnih plošč se izračuna z naslednjo enačbo:

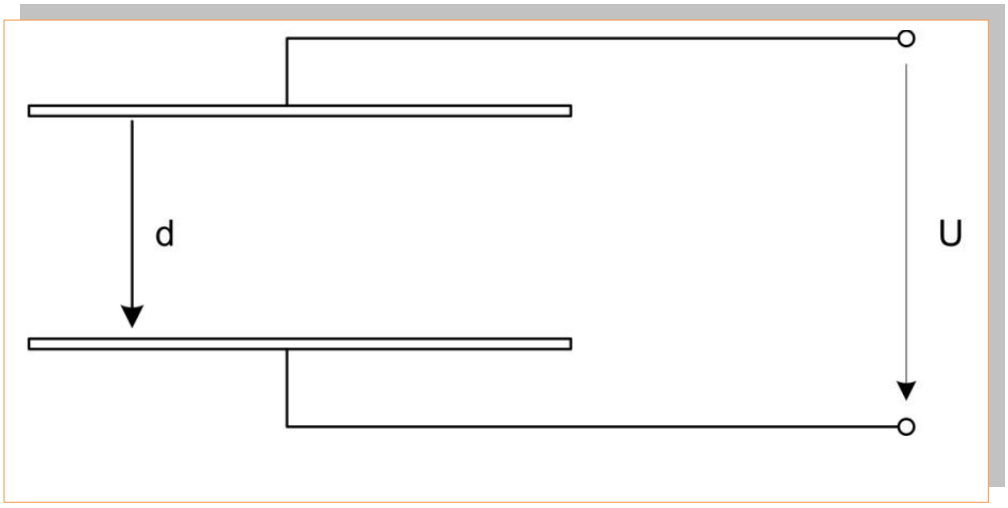
$$C = \varepsilon_0 \varepsilon_r A / d$$

ε_0 - dielektrična konstanta praznega prostora

ε_r - relativna dielektrična konstanta dielektrika med ploščama

A - površina plošč

d - razdalja med ploščama



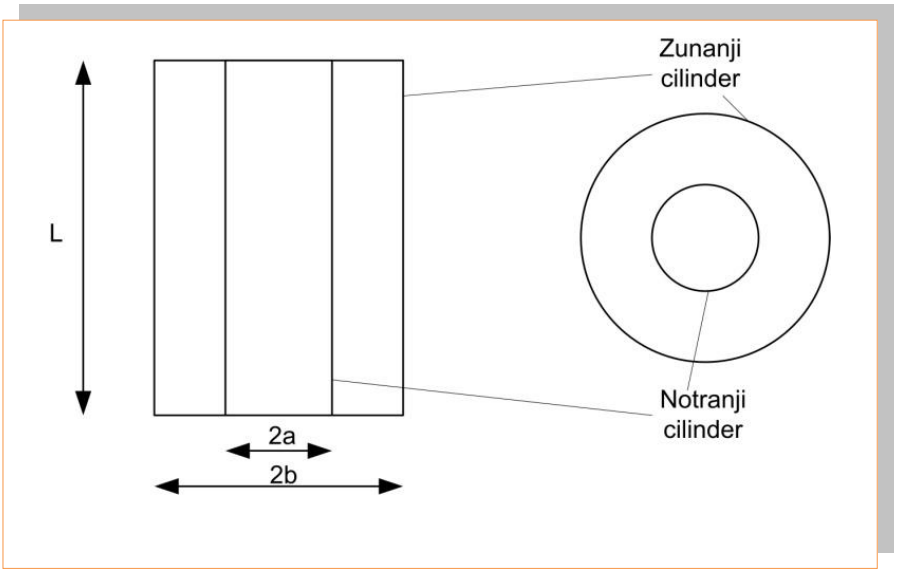
Slika 2.41: Princip delovanja ploščatega kondenzatorja

Vir: Lastni

Pri cilindričnem kondenzatorju kapacitivnost izrazimo z naslednjo enačbo:

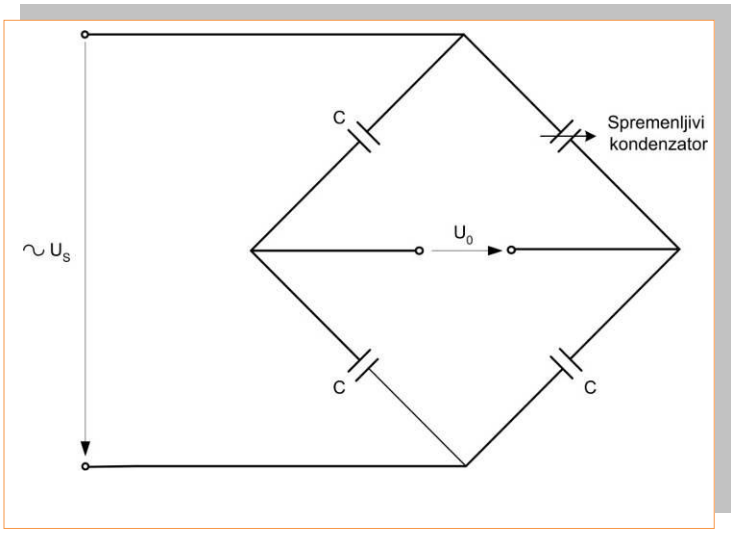
$$C = 2\pi\varepsilon_0\varepsilon_r / \ln(b/a)$$

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 2.42: Princip delovanja cilindričnega kondenzatorja

Vir: Lastni

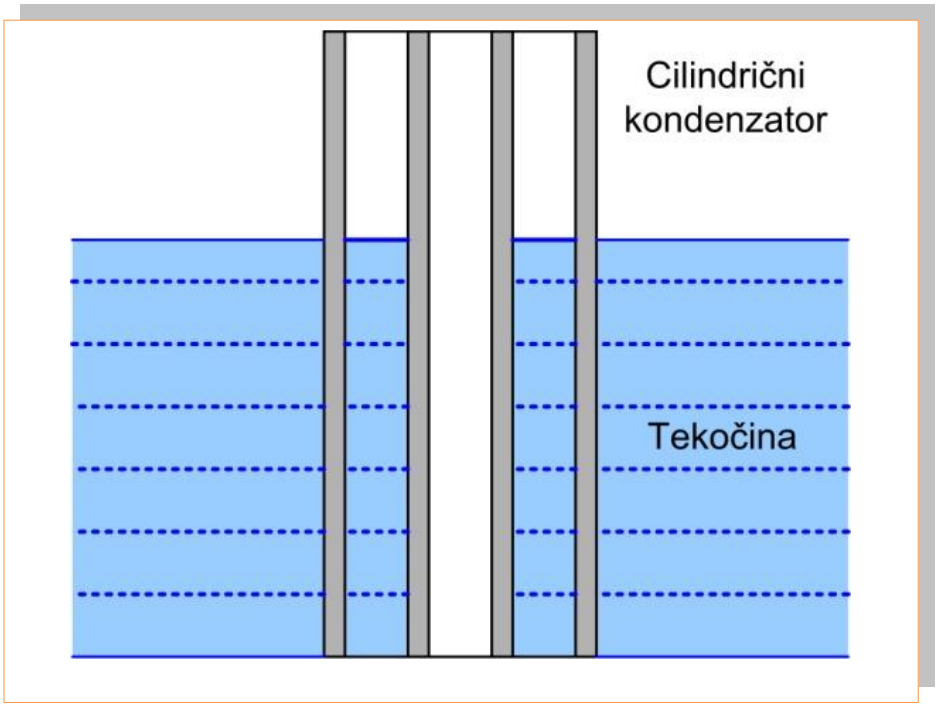


Slika 2.43: Kapacitivno mostično vezje

Vir: Lastni

Na sliki 4.43 vidimo kapacitivno mostično vezje, ki se uporablja za merjenje nivoja tekočine s pomočjo cilindričnega kondenzatorja.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



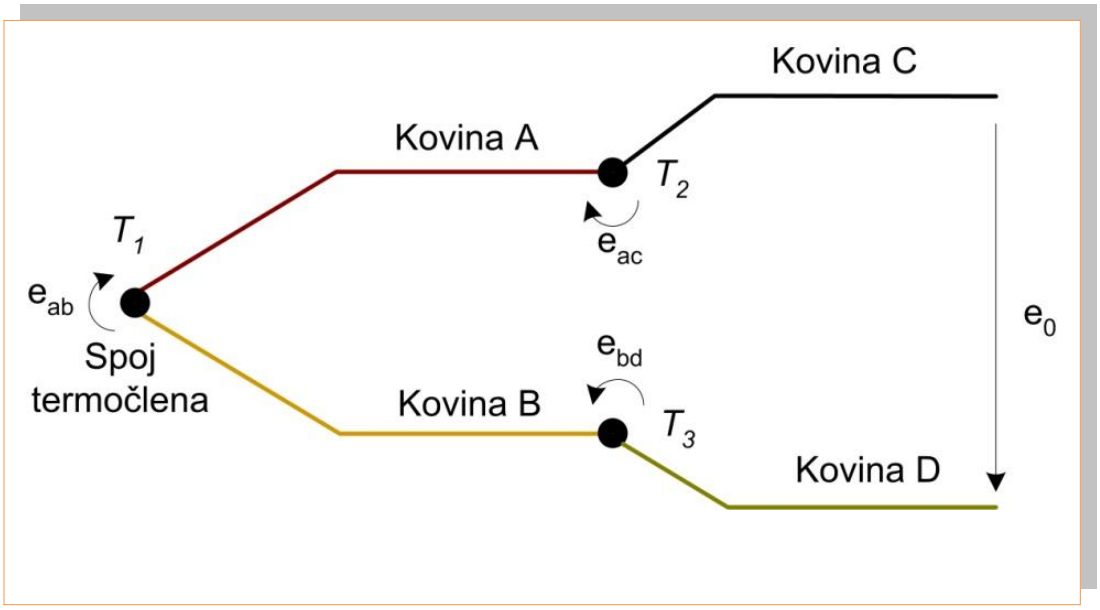
Slika 2.44: Meritev nivoja tekočine s pomočjo cilindričnega kondenzatorja

Vir: Lastni

TERMOELEKTRIČNI MERILNI PRETVORNIKI

Dve različni kovini na enem koncu spojimo. Ko pride na spoju do spremembe temperature (gretje, hlajenje), se na spoju inducira napetost. Inducirana napetost je funkcija temperaturne razlike med dvema spojema različnih kovin. Ta pojav se imenuje Seebeckov efekt. Inducirana napetost se izračuna na sledeči način:

$$e_o = e_{ab}+e_{bc}+e_{ac}$$



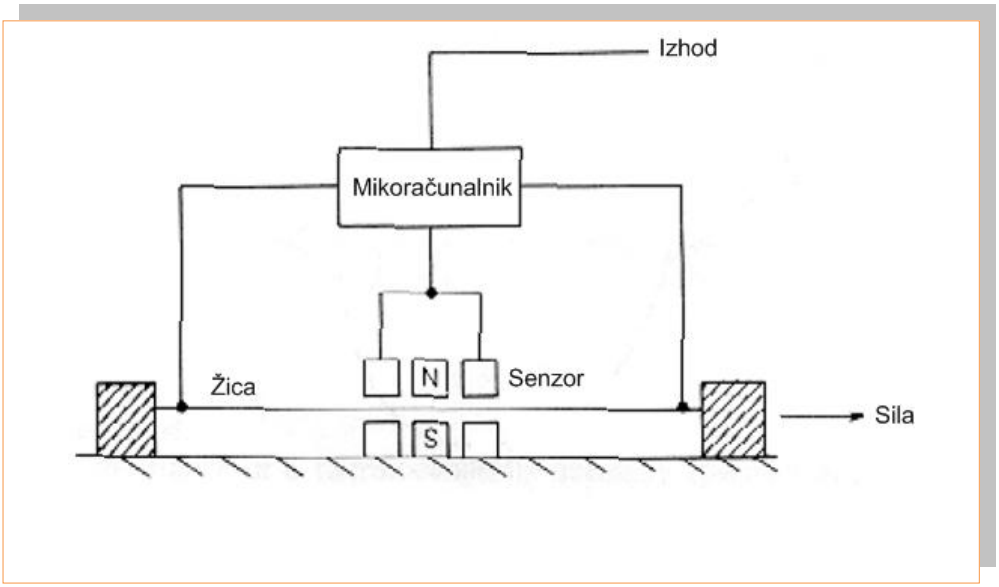
Slika 2.45: Termočlen, sestavljen iz štirih različnih kovin

Vir: Lastni

Termočleni so ceneni ter enostavni za izdelavo in uporabo. Njihova slabost je mala inducirana napetost e_o, ki je pogosto manjša od elektromagnetnih šumov, ki se inducirajo na senzorju. Zato morajo biti signali ojačani in filtrirani z diferencialnimi ojačevalniki.

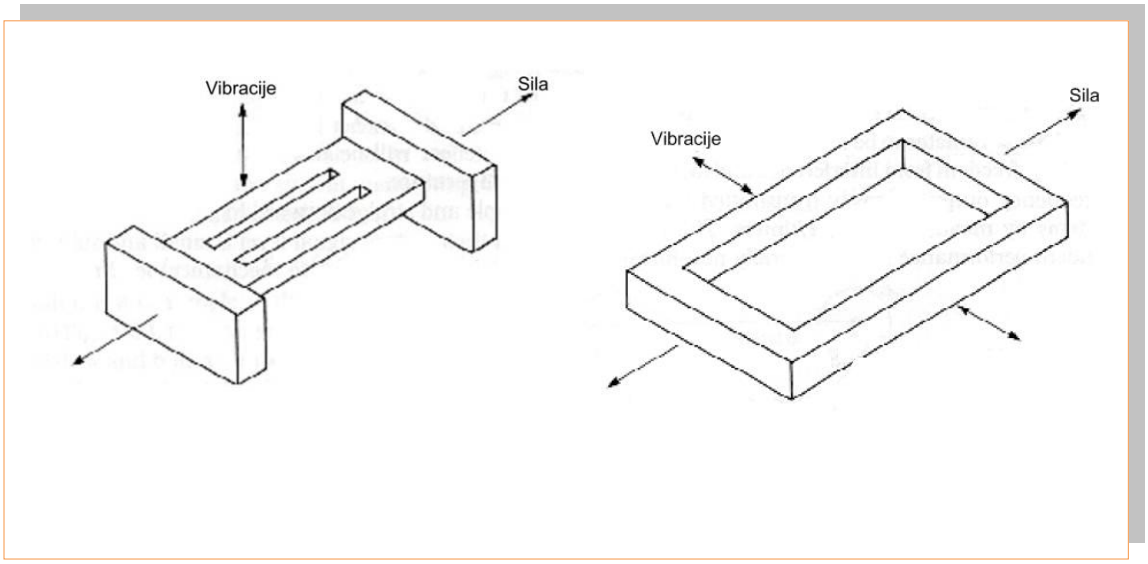
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

VIBRACIJSKI PRETVORNIKI



Slika 2.46: Merilec napetosti v vibrirajoči žici

Vir: Lastni



Slika 2.47: Merilmi pretvornik z vibrirajočo vilico

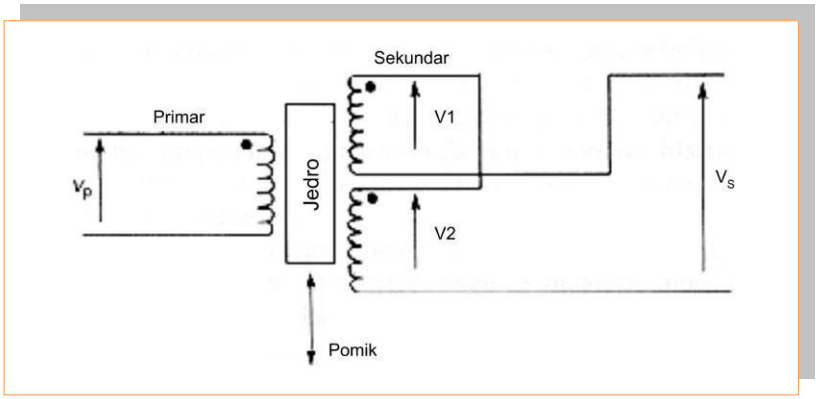
Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

INDUKTIVNI MERILNI PRETVORNIKI

LINEARNI VARIABILNI DIFERENCIALNI TRANSFORMATOR

Linearni variabilni transformator je sestavljen iz primarnega navitja in dveh sklopljenih sekundarnih navitij. Izhodna inducirana napetost se spreminja v odvisnosti od pomika jedra v transformatorju. Torej lahko na takšen način merimo pomik.

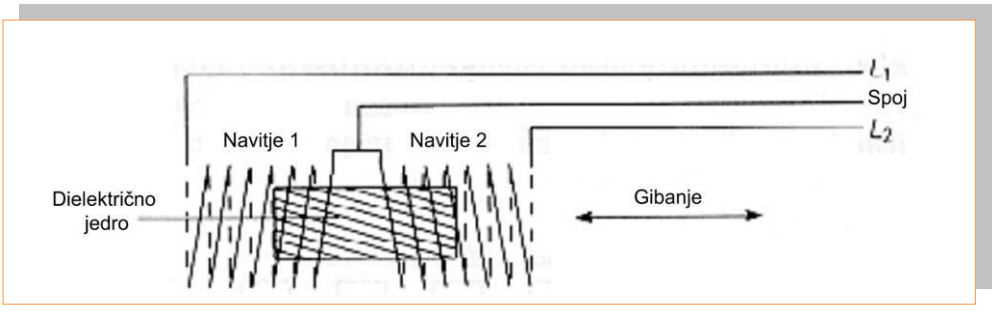


Slika 2.48: Linearni variabilni diferencialni transformator

Vir: Lastni

LINEARNI VARIABILNI MERILNI PRETVORNIK

Ima za razliko od variabilnega diferencialnega transformatorja namesto železnega jedra dielektrično jedro. Ob pomiku zunanjega ali notranjega jedra se spremeni induktivnost tuljav. Ob vzbujalni frekvenci nekaj 100 kHz lahko merimo razdalje do 220 mm.



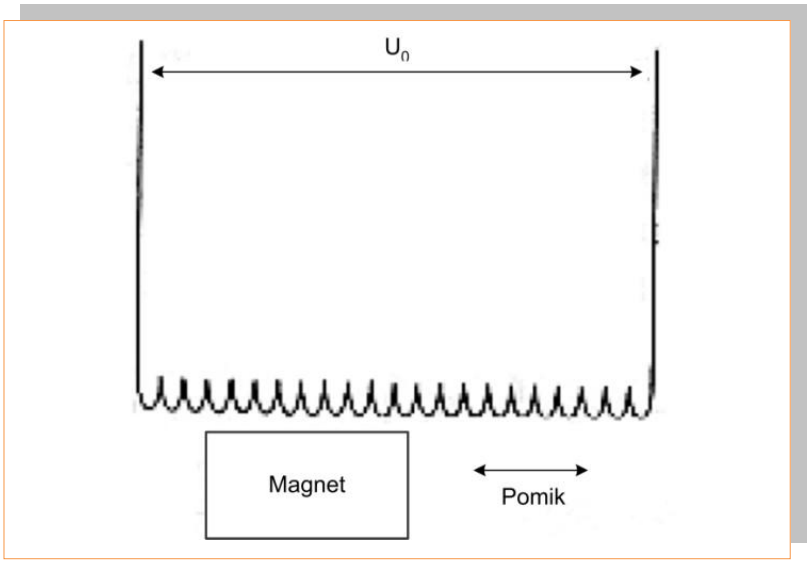
Slika 2.49: Linearni variabilni merilni pretvornik

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

INDUKTIVNI MERILNIKI HITROSTI

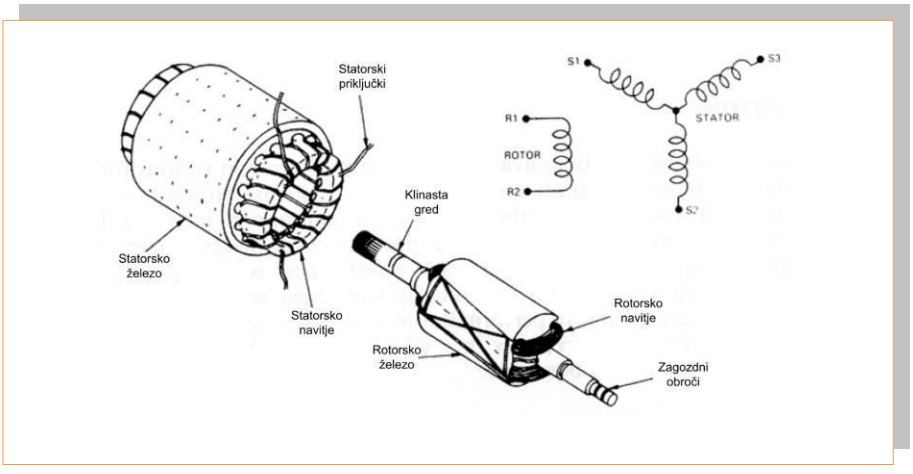
Večja kot je hitrost pomika magneta, večja je inducirana napetost. Na tem principu delujejo meritve krožne hitrosti s tahogeneratorjem, ki je v bistvu DC ali AC električni motor s permanentnim magnetom v rotorju.



Slika 2.50: Induktivni merilnik hitrosti

Vir: Lastni

RESOLVERJI

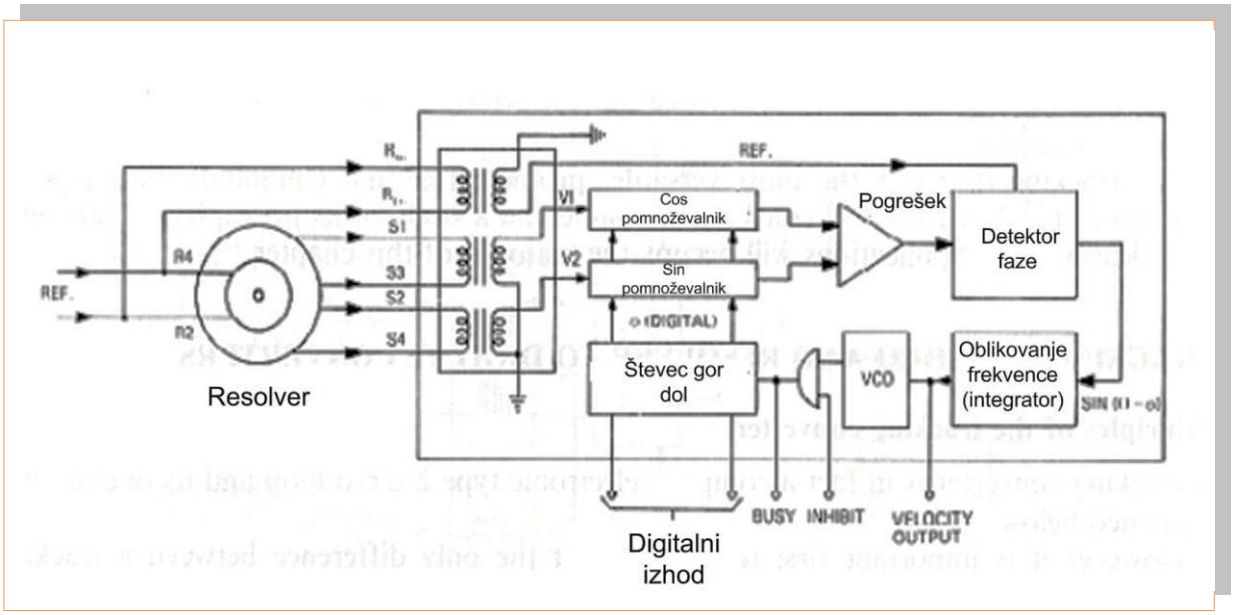


Slika 2.51: Zgradba resolverja

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Resolver je zgrajen iz rotorskega navitja in treh statorskih navitij povezanih v zvezdo. Navitja generirajo tri sinusne napetosti električno zamaknjene za 120 stopinj. Synhro resolver pa ima eno rotorsko navitje in dva statorksa navitja, ki generirata dvoje za 90 stopinj električno premaknjenih sinusnih signalov.



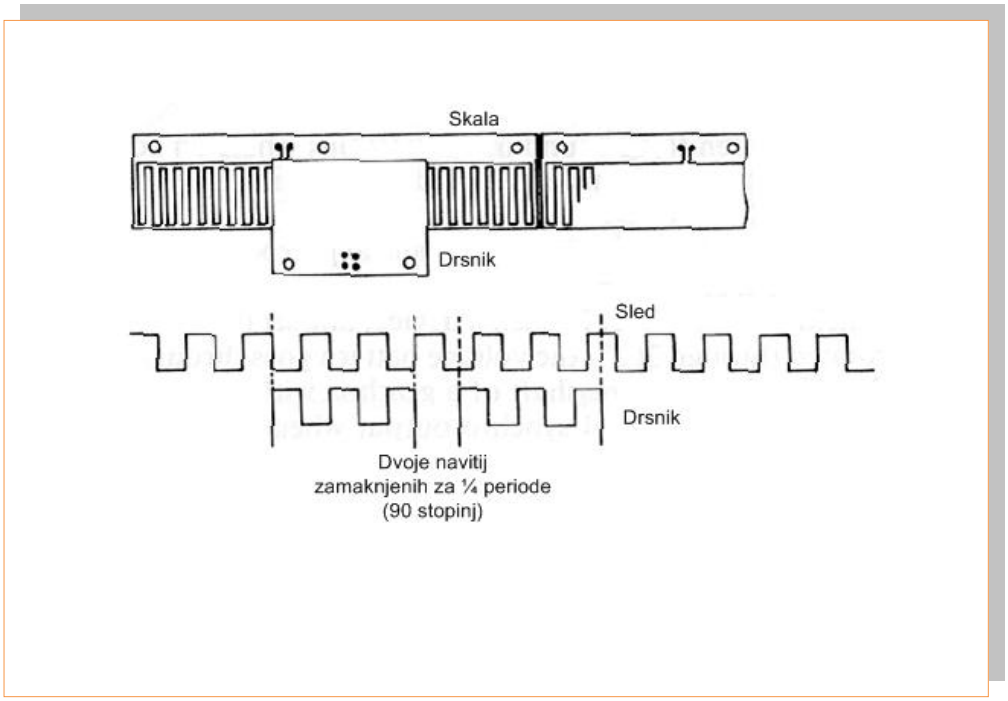
Slika 2.12: Pretvorba resolverskih statorskih napetosti v digitalno vrednost

Vir: Lastni

INDUCTOSYN

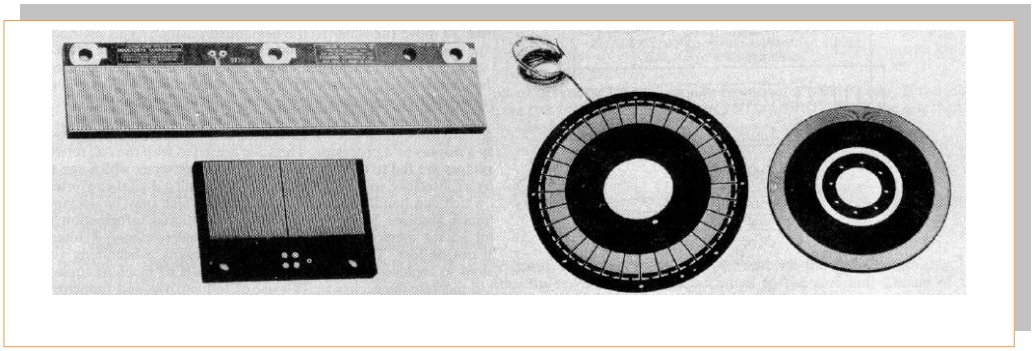
Linearni inductosy je sestavljen iz induktivne letve in induktivnega drsnika. Vezje se vzbuja s frekvenco od 6 – 10 kHz, kjer je izhod oblike Vsin ωt.sin(2pX/S) in Vsin ωt.cos(2pX/S). X je pomik drsnika, S pa ciklična dolžina drsnika. Poznamo tudi rotacijske inductosyne, ki delujejo na istem principu. V tem primeru gre za dva (stator, rotor) vrtljiva diska.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 2.53: Delovanje linearnega inductosyna

Vir: Lastni



Slika 2.54: Linearni in rotacijski inductosyn

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

POVZETEK

Xxx

PONOVIMO

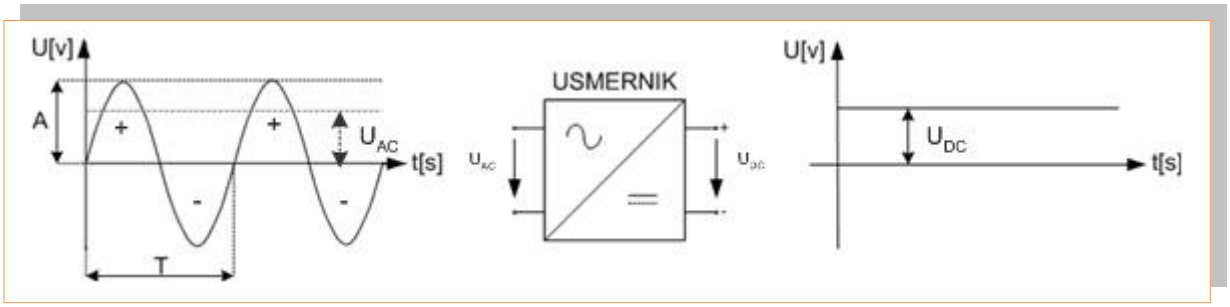
1. Xxxxxx

3 KOMPONENTE MOČNOSTNE ELEKTRONIKE

USMERNIŠKA VEZJA

Usmernik je elektronsko vezje, ki spremeni izmenično napetost in tok v enosmerno napetost in tok. Napetost in tok, ki se pojavita na izhodu usmerniškega vezja, imata poleg enosmerne komponente tudi izmenične komponente. Valovitost toka in napetosti je posledica usmerniškega procesa. Odpravimo jo s filtriranjem oziroma glajenjem napetosti in toka.

V osnovi poznamo enofazna, trifazna, polvalna, polnovalna in druga usmerniška vezja. Vsa usmerniška vezja delujejo na istem principu. Izmenično napetost in tok iz omrežja s pomočjo usmerniških elementov usmerijo v enosmerno napetost.



Slika 3.1: Princip delovanja usmerniškega vezja

Vir: Lastni

$$f = \frac{1}{T}$$

f – frekvenca nihanja izmenične napetosti;

T – čas trajanja ene periode;

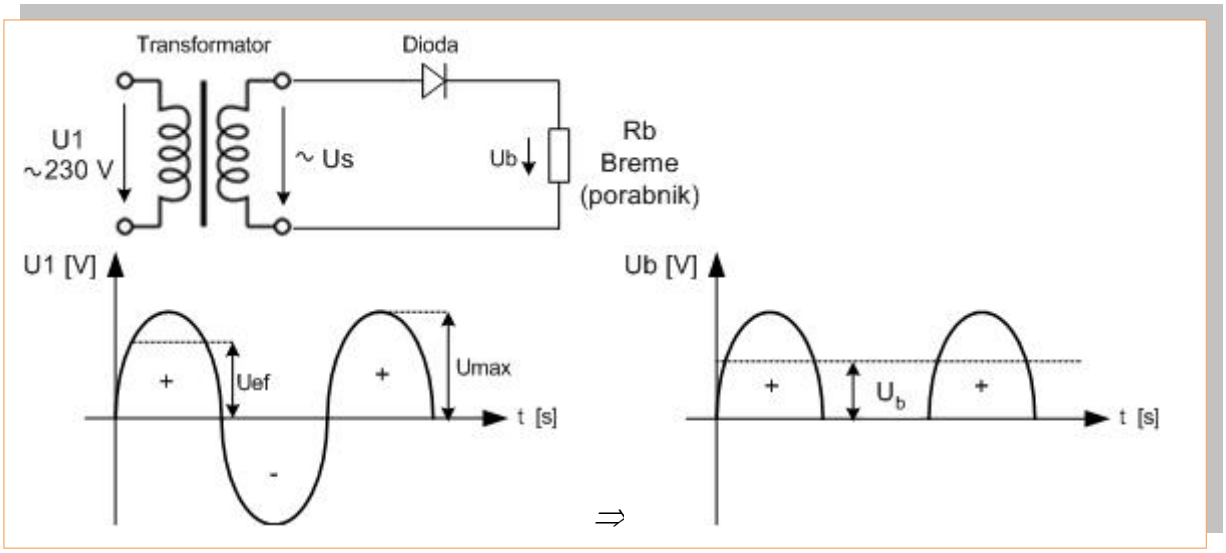
A – amplituda, maksimalna vrednost izmenične napetosti.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Izmenična napetost je torej tista napetost, ki s časom spreminja smer (primer : napetost vtičnice 230 V 50 Hz… AC napetostni vir). Enosmerna napetost je tista napetost, ki s časom ne spreminja smeri oziroma polaritete (primer: baterije, akumulatorji…DC napetostni viri). Veliko porabnikov npr. TV, osebni računalnik, zabavna elektronika itd. deluje na enosmerni vir napajanja. Zato moramo izmenično napetost pretvoriti oziroma usmeriti v enosmerno napetost. To lahko izvedemo z usmerniškimi vezji, ki jih bomo spoznali v nadaljevanju.

POLVALNI USMERNIK

Polvalni usmernik je najenostavnejše usmerniško vezje saj vsebuje samo eno diodo. Zaradi preproste zgradbe ima dokaj veliko valovitost izhodne napetosti. Napetost na bremenu je sicer enosmerna (ves čas pozitivna), vendar ne konstantna. Napetost na bremenu vsebuje izmenično in enosmerno komponento. Izmenična komponenta povzroča valovitost usmerjene napetosti. Valovitost umerjene napetosti je neželena, zato jo odpravljamo z gladilnimi filtri.



Slika 3.2: Polvalni usmernik

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

$$U_{ef} = \frac{U_{max}}{\sqrt{2}} \qquad U_{DC} = U_b = \frac{U_S}{\pi} = 0{,}318 \cdot U_2$$

U_{ef} - efektivna vrednost izmenične napetosti

U_{max} - maksimalna vrednost izmenične napetosti

U_s – potrebna sekundarna napetost transformatorja

U_b – napetost na bremenu

Slika 3.3: Dioda

Vir: Lastni

Delovanje polvalnega usmerniškega vezja

Na vhodu (U1) je priključena izmenična omrežna napetost. Transformator v razmerju primarnih in sekundarnih ovojev navitja pretvori omrežno izmenično napetost v nižjo izmenično napetost (Us), ki jo izberemo glede na našo želeno enosmerno napetost. Hkrati transformator galvansko loči vezje, ki ga napajamo od omrežne napetosti 230V. V zaporni smeri dioda toka ne prevaja. Vsa negativna sekundarna napetost se porabi kot napetostni padec na diodi. Za polvalni usmernik moramo izbrati takšno diodo, da je njena maksimalna dovoljena zaporna napetost nekoliko večja od Us. Maksimalno dovoljeno zaporno napetost označujemo z $U_{RRM.}$ Povprečna vrednost usmerjenje napetosti požene skozi breme enosmerni tok I_{DC}, ki je omejen le z upornostjo bremena.

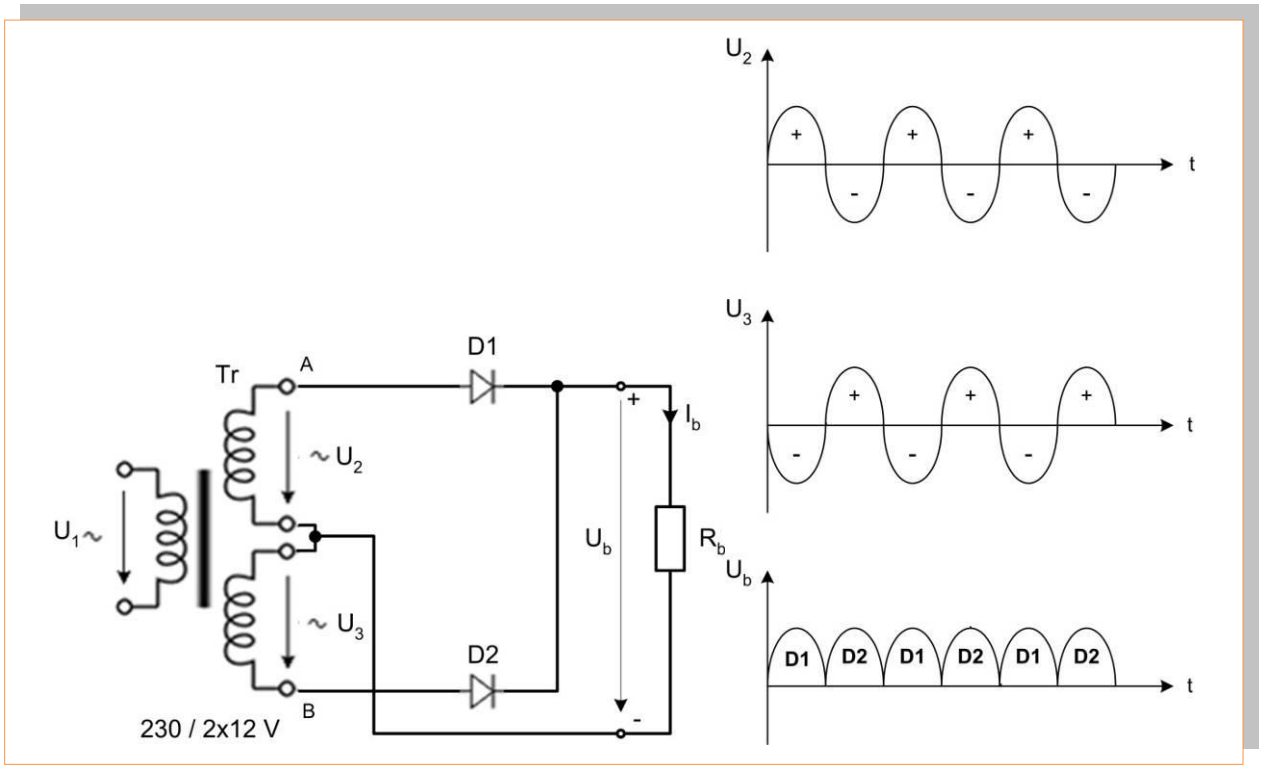
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

$$I_{DC} = I_b = \frac{U_{DC}}{R_b} = \frac{U_b}{R_b}$$

Polnovalni usmernik uporabljamo za napajanje nezahtevnih in preprostih naprav nizke moči. Zaradi polvalnega usmerjanja je izkoristek moči majhen.

POLNOVALNI USMERNIK

Izkoristek polvalnega usmernika želimo povečati. Zato sestavimo usmernik z dvema diodama. Vezje deluje kot dva polvalna usmernika, od katerih vsak deluje eno polperiodo.



Slika 3.4: Polnovalni usmernik in usmerjena napetost

Vir: Lastni

V tem primeru je transformator nekoliko drugačen. Ima tri priključke. Srednji odcep je nameščen na polovici sekundarnega navitja. Sekundarna (izhodna) napetost se torej deli na dva enaka dela.

Delovanje polnovalnega usmerniškega vezja

Pozitivna napetost v točki A požene tok skozi diodo D1 in breme Rb. Tokokrog se zaključi v srednjem odcepu transformatorja. Dioda D2 je v tem primeru zaporno polarizriana. Na bremenu se pojavi padec napetosti Ub. V naslednji negativni polperiodi se polaritete zamenjajo. V tem primeru prevaja dioda D2, D1 pa je zaporno polarizirana. Skozi diodo D2 teče tok preko bremena in se zaključi v srednjem odcepu transformatorja. Pri izbiri diode moramo paziti, da izberemo zaporno napetost diode večjo od amplitudne vrednosti sekundarne napetosti. Povprečna vrednost polnovalno usmerjene napetosti bo dvakrat večja od povprečne vrednosti polvalno usmerjene napetosti.

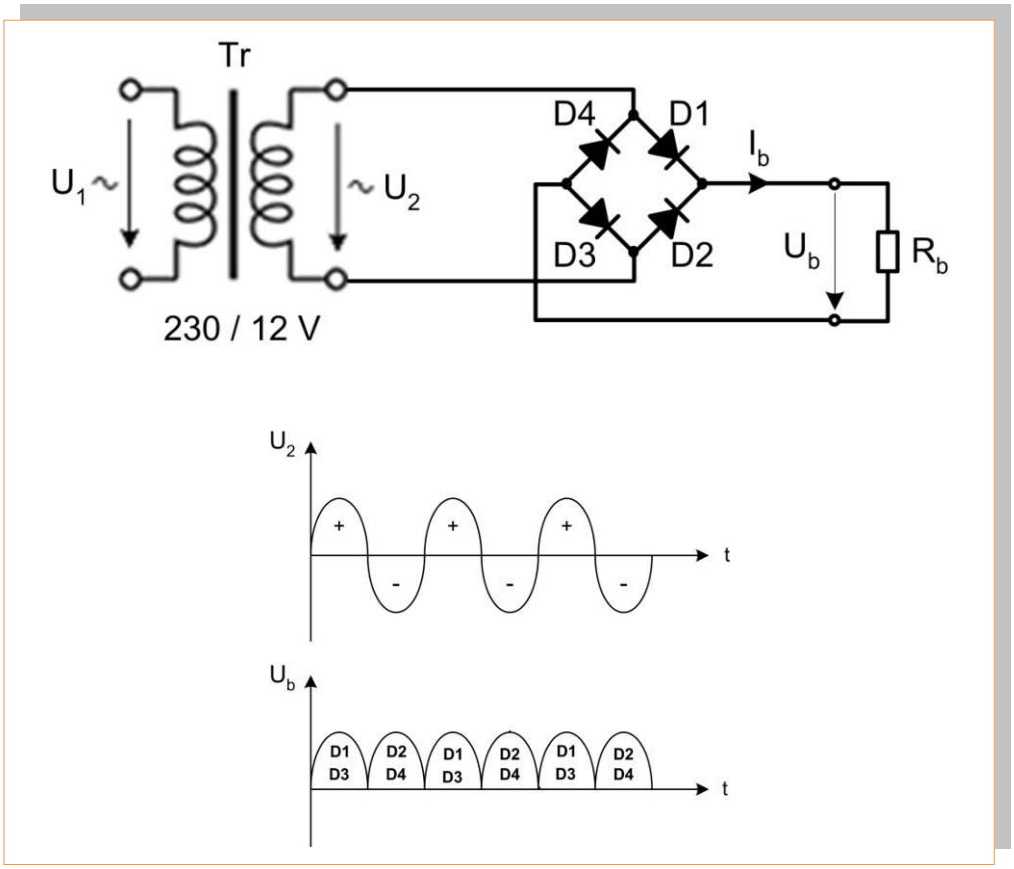
$$U_{DC} = U_b = \frac{U_2}{2} \cdot \frac{2}{\pi} = 0{,}637 \cdot \frac{U_2}{2}$$

Tok I_{DC}je enosmerni tok, ki teče skozi breme. Obakrat teče tok skozi breme v isti smeri in v srednji odcep. Skozi diodo pa teče tok le v eni polperiodi. Povprečna vrednost toka skozi diodo (enosmerni tok skozi diodo) je zato dvakrat manjši od izhodnega bremenskega toka. **Izkoristek polnovalnega usmernika je dvakrat večji od izkoristka polvalnega usmernika.**

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

MOSTIČNI POLNOVALNI USMERNIK – GREATZOV MOSTIČ (SPOJ)

Slaba lastnost polnovalnega usmernika z dvema diodama je uporaba transformatorja s sredinskim odcepom, ki podraži usmnernik. Polnovalno usmerjanje lahko dosežemo tudi z običajnim transformatorjem in uporabo Greatzovega ali mostičnega usmerniškega vezja zgrajenega iz štirih diod. To vezje se v usmerniških naprav najpogosteje uporablja.



Slika 3.5: Polnovalni mostični usmernik in usmerjena napetost

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Delovanje polnovalnega mostičnega vezja

Če pogledamo sliko 3.5 vidimo, da sta dve diodi obrnjeni v isto smer, dve pa vsaka v drugo smer. V pozitivni polperiodi steče tok skozi diodo D1, preko bremenskega upora Rb in nato preko diode D3 na spodnjo sponko sekundarnega dela transformatorja. Medtem sta diodi D2 in D4 zaporno polarizirani. V negativni polperiodi steče tok skozi diodo D2 preko bremenskega upora Rb in nato preko diode D4 na zgornjo sponko sekundarnega dela transformatorja. Napetost na bremenu je enaka sekundarni napetosti zmanjšani za padec napetosti na obeh diodah.

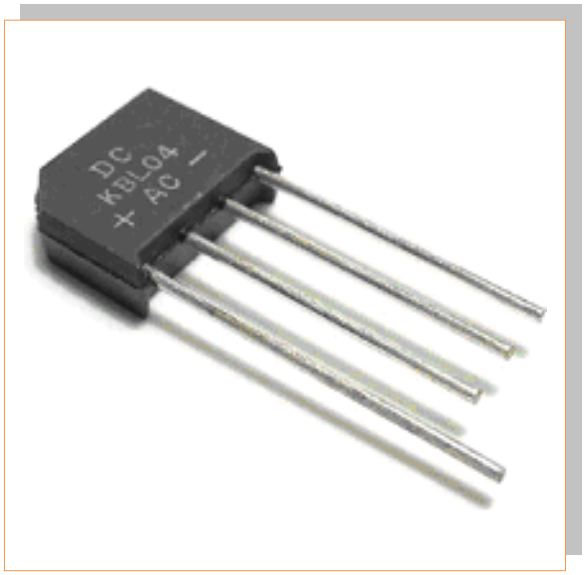
Če padec napetosti na diodah zanemarimo, lahko rečemo, da je na bremenu napetost približno enaka sekundarni napetosti transformatorja. Pri izbiri diode moramo izbrati takšno diodo, ki ima zaporno napetost večjo od povprečne vrednosti napetosti na

bremenu. Povprečna vrednost napetosti na bremenu je enaka kot pri polnovalnem usmerniku z dvema diodama (amplituda je v tem primeru U2 in ne U2/2 kot pri polnovalnem usmerniku z dvema diodama.

$$U_{DC} = U_b = 2 \cdot \frac{U_2}{\pi} \cdot = 0{,}637 \cdot U_2 \qquad I_{DC} = I_b = \frac{U_{DC}}{R_b}$$

Gretzovo vezavo diod lahko izvedemo s štirimi diodami, običajno pa se uporabi kar diodni mostiček, ki je zaprt v plastično ohišje s štirimi priključki.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 3.6: Greatzov mostič

Vir: Lastni

GLAJENJE USMERJENE NAPETOSTI

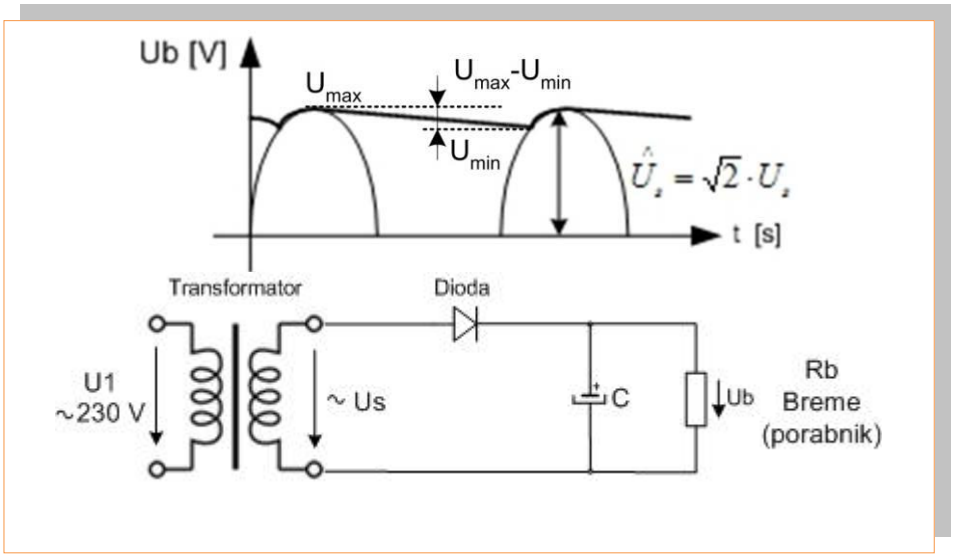
Valovitost el. napetosti in toka zmanjšamo s glajenjem, ki ga opravimo s filtri. To so el. vezja, ki vsebujejo kondenzatorje in tuljave (dušilke). Upornost teh elementov je odvisna od frekvence toka. Pri izmeničnih tokovih je upornost drugačna kot pri enosmernih tokovih.. To lastnost lahko izkoristimo in v kombinaciji dosežemo dušenje izmenične komponente usmerjene napetosti.

Glajenje napetosti lahko izvedemo s tremi tipi filtrov:

- glajenje s kondenzatorjem
- glajenje z L - filtrom
- glajenje s π - filtrom.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Glajenje s kondenzatorjem (kapacitivnim filtrom)



Slika 3.7: Polvalni usmernik z gladilnim členom in usmerjena napetost

Vir: M.Milanovič. Močnostna elektrotehnika

Glajenje s kapacitivnim filtrom se uporablja v mnogih aplikacijah predvsem zaradi cenene izvedbe.

Delovanje :

V pozitivni polperiodi teče tok preko diode in bremena Rb. Del toka teče tudi na kondenzator in ga polni. V negativni polperiodi ko dioda ne prevaja pa daje tok bremnu kondenzator, ki se v drugi polperiodi prazni. Namesto enosmerne pulzirajoče napetosti bomo na bremenu dobili glajeno napetost kot vidimo na zgornji sliki 7.7. Pri glajenju s kondenzatorjem je pomemben faktor valovitosti. Faktor valovitosti je definiran kot razmerje med amplitudo prve sinusne izmenične napetosti in enosmerno komponento:

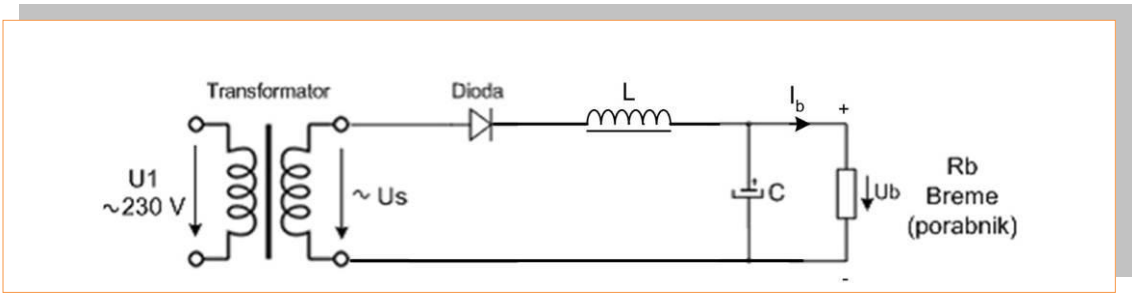
$$\eta = \frac{U_{\max} - U_{\min}}{2 \cdot U}$$

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Valovitost bo majhna, če bo imel kondenzator veliko časovno konstanto praznjenja ($\tau = R \cdot C$)**.** Kapacitivnost in bremenska upornost morata biti v torej velika. Časovna konstanta praznjenja naj bo mnogo daljša od periode. Običajno se postavi zahteva, da je 10-krat daljša od periode izmeničnega signala iz transformatorja. Kondenzator izberemo tako, da je U_{max}- U_{min} manjša od 10%temenske vrednosti sekundarne napetosti transformatorja.

Pri gradnji usmernikov z gladilnim kondenzatorjem se ravnamo po zelo preprostem pravilu, ki da razmeroma dovolj dobro oceno. Pri danem transformatorju (z dano efketivno napetostjo U_{ef}) računamo, da bo napetost na bremenu približno enakA temenski vrednosti sekundarja, ki je za $\sqrt{2}$-krat večja od efektivne vrednosti sekundarne napetosti. Do odstopanj pride, če je čas polnjenja kondenzatorja prekratek.Tipične vrednosti kondenzatorjev se gibljejo od 470 μF do nekaj tisoč μF. Zaradi takšnih vrednosti se običajno uporablja elektrolitski kondenzator, pri katerem moramo paziti na polariteto. Izbiro kondenzatorjev lahko opravimo tudi na podlagi posebnih tabel proizvajalcev.

Glajenje z L-filtrom (induktivni filter)

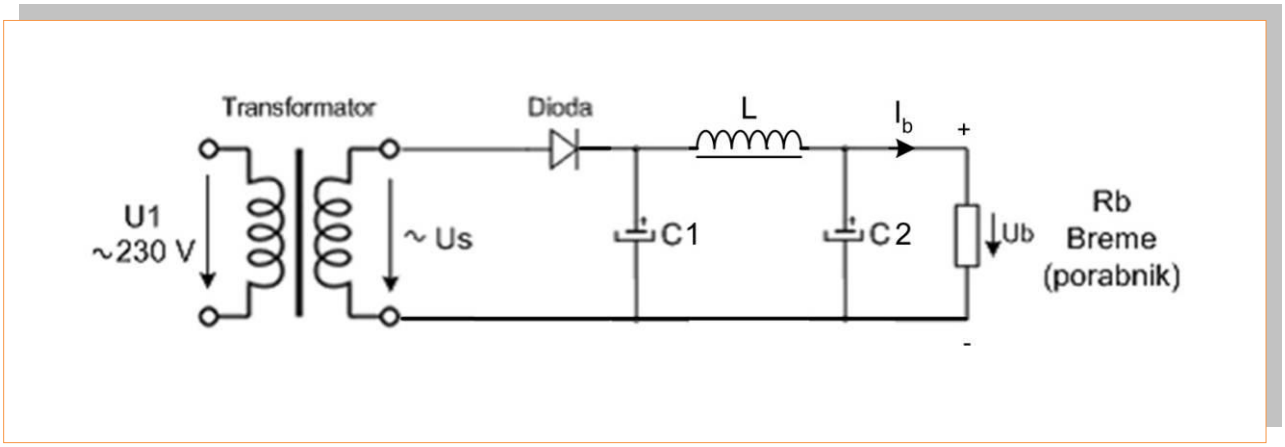


Slika 3.8: Polvalni usmernik z L - gladilnim členom

Vir: M.Milanovič. Močnostna elektrotehnika

Tuljava ima lastnost, da se upira hitrim spremembam toka.Če skozi njo teče valovit tok, se bo tuljava upirala tej valovitosti in jo zmanjševala. Tok skozi tuljavo bo manj valovit. Za enosmerno napetost in tok predstavlja tuljava zelo malo upornost (upornost žice). Kondenzator tok, ki ga je gladila dušilka še dodatno zgladi.Dobili bomo skoraj povsem zglajeno enosmerno napetost z zelo majhno izmenično komponento. Slabost L-filtra je tuljava, ki je razmeroma drag element, zato takšen filter redko uporabljamo.

Glajenje s π - filtrom



Slika 3.9: Polvalni usmernik z π - gladilnim členom

Vir: M.Milanovič. Močnostna elektrotehnika

Zelo dobro lahko gladimo usmerjeno napetost, če uporabimo oba filtra zaporedno. S takšnim filtrom lahko zmanjšamo valovitost za približno 10-krat. Zaradi visoke cene in dimenzij ga uporabljamo le v usmernikih za velike moči.

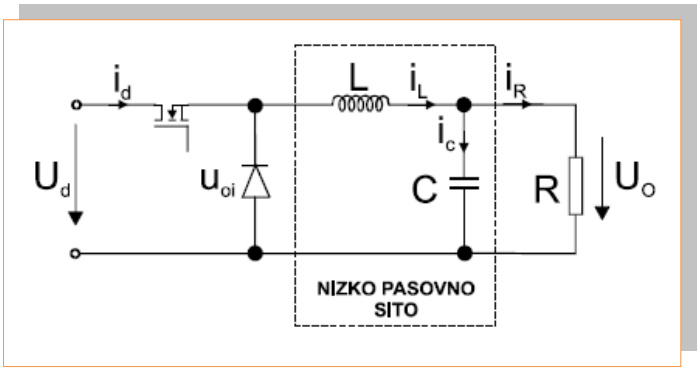
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

PRESMERNIŠKA IN RAZSMERNIŠKA VEZJA

DC/DC pretvorniki se uporabljajo za napajanje enosmernih porabnikov, ki jih najdemo v vseh znanih elektronskih napravah in pri pogonih enosmernih motorjev (enosmerni servomotorji). Enosmerna napetost je običajno neregulirana, ker je usmerjena z diodnim usmerjanjem in filtriranjem izhodne napetosti. Z DC/DC pretvornikom torej lahko reguliramo tudi enosmerno napetost hkrati pa ima DC/DC pretvornik še funkcijo galvanske ločitve pri enosmernem napajanju elektronskih naprav.

PRETVORNIK NAVZDOL (BUCK KONVERTER)

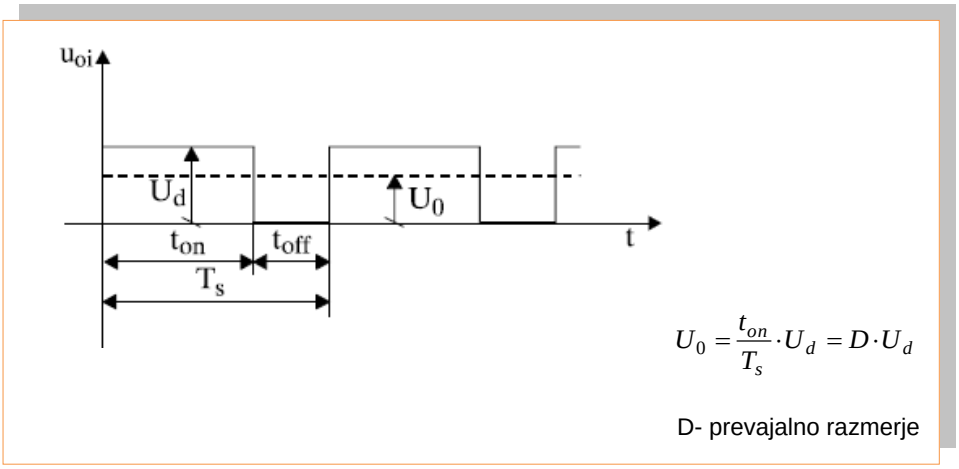
Iz imena pretvornika lahko razberemo, da pretvornik navzdol na svojem izhodu proizvaja nižjo srednjo vrednost napetosti, kot je priključena na vhodu (npr U_d=12V, U_0 = 5v. V praksi se uporablja povsod, kjer je potrebna nižja napetost nižja od enosmerne vhodne napajalne napetosti.



Slika 3.10: Pretvornik navzdol (buck converter, step down converter)

Vir: M.Milanovič. Močnostna elektrotehnika

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 3.11: Časovni potek napetosti na vhodu in izhodu nizkopasovnega sita

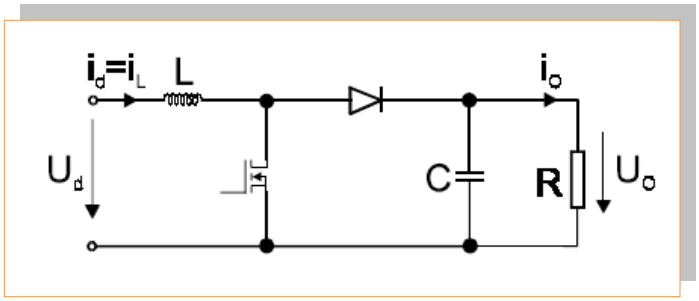
Vir: M.Milanovič. Močnostna elektrotehnika

Delovanje: S spreminjanjem prevajalnega razmerja vplivamo na izhodno napetost. Prevajalno razmerje spreminjamo s pomočjo tranzistorja,, ki ga vklapljamo in izklapljamo. S tem vklopimo in izklopimo vhodno napetost. Glede na čas vklopa in izklopa se tuljava in kondenzator polneta na različne vrednosti. Z razmerjem časa vklopa in izklopa tranzistorja (prevajalno razmerje) določamo velikost izhodne napetosti. Ko je stikalo (tranzistor) sklenjeno se energija pretaka od izvora proti bremenu, ko pa je stikalo (tranzistor) izklopljeno se breme napaja z energijo z energijo, ki ostane shranjena v tuljavi in kondenzatorju. Shranjena energija bi lahko uničila tranzistor, zato je v vezju zaporno polarizirana dioda, ki premosti konico na negativni potencial.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

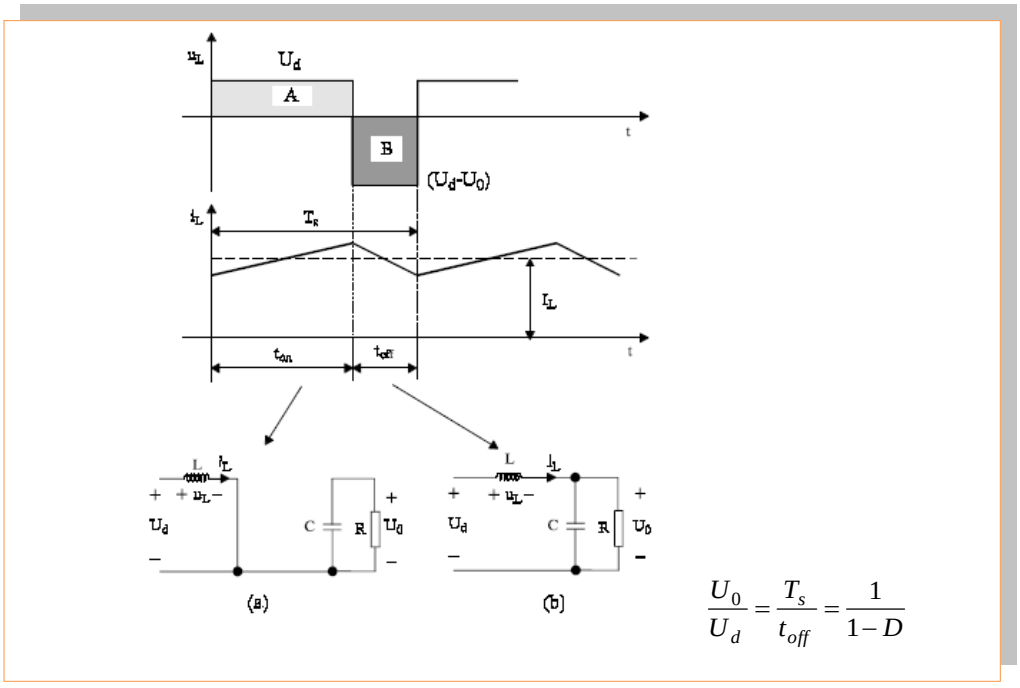
PRETVORNIK NAVZGOR (BOOST CONVERTER)

Iz imena pretvornika lahko razberemo, da pretvornik navzgor proizvaja na izhodu višjo napetost kot je vhodna napetost.



Slika 3.12: Pretvornik navzgor (boost convereter)

Vir: M.Milanovič. Močnostna elektrotehnika



Slika 3.13: Tok na tuljavi in nadomestna vezje ko je tranzistor vklopljen in izklopljen

Vir: M.Milanovič. Močnostna elektrotehnika

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Delovanje: Ko je tranzistor vključen, je dioda zaporno polarizirana, izhod pa je ločen od vhodnega dela. Energija se pretaka od izvora skozi tuljavo (induktivnost) L. Ko je stikalo (tranzistor) izključen sprejema izhodna stopnja energijo izvora in hkrati energijo, ki je shranjena v tuljavi.

Poleg omenjenih pretvornikov poznamo še:

- **Pretvornik navzdol/navzgor** (buck/boost converter), ki se uporablja za generiranje negativne napetosti na izhodu.
- **Čukov pretvornik**, ki se uporablja za generiranje negativne napetosti na izhodu. V tem primeru se za prenos energije uporablja kondenzator.

RAZSMERNIŠKA VEZJA (DC-AC PRETVORNIK)

Za napajanje izmeničnih električnih pogonov iz enosmernega izvora potrebujemo ustrezne naprave, ki jih imenujemo razsmerniki. V splošnem se delijo na enofazna in trifazna razsmerniška vezja. Pri tem želimo spreminjati tako amplitudo kot frekvenco izhodnega toka, ki mora biti čimbolj sinusne oblike, hkrati pa mora imeti pretvornik tudi minimalne stikalne izgube. V industriji se razsmerniki uporabljajo v dveh velikih skupinah porabnikov:

- Sistemi neprekinjenega napajanja (UPS, **U**ninteruptible **P**ower **S**upplies **S**ystem)

Slika 3.14: Sistem neprekinjenega napajanja

Vir: M.Milanovič. Močnostna elektrotehnika

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

- Izmenični elektromotorni pogoni

Slika 3.15: Izmenični elektromotorni pogon

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

POVZETEK

Xxx

PONOVIMO

1. Xxxxxx

4 OBDELAVA PODATKOV

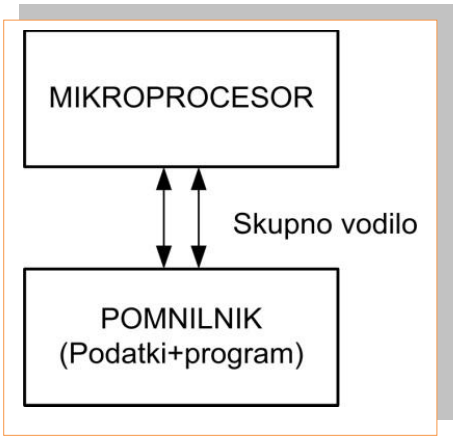
Obdelavo podatkov je možno izvesti na dva načina

- **Analogno**, z operacijskimi ojačevalniki (analogni računalnik);
- **Diskretno,** z digitalnim računalnikom.

Digitalni računalnik se uporablja (Von Neumann tip računalnika) bistveno pogosteje zaradi nizke cene, enostavne uporabe (programiranje). Hkrati nima problemov, ki jih ima analogni računalnik zgrajen na osnovi operacijskih ojačevalnikov (off-set, drift, občutljivost na motnje itd.) Poznamo dve vrsti arhitekture mikroračunalnikov :

- Von Neumann model

V tem primeru so podatki in program v istem pomnilniku. (npr. Pentium).



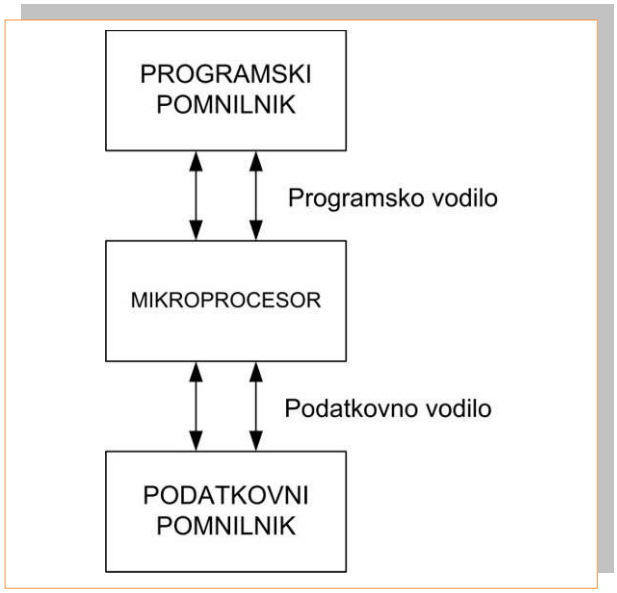
Slika 4.1: Von Neumann-ov model mikroračunalnika

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

- Harwardski model

V tem primeru so podtaki in program v ločenih pomnilnikih (PIC mikroračunalnik)

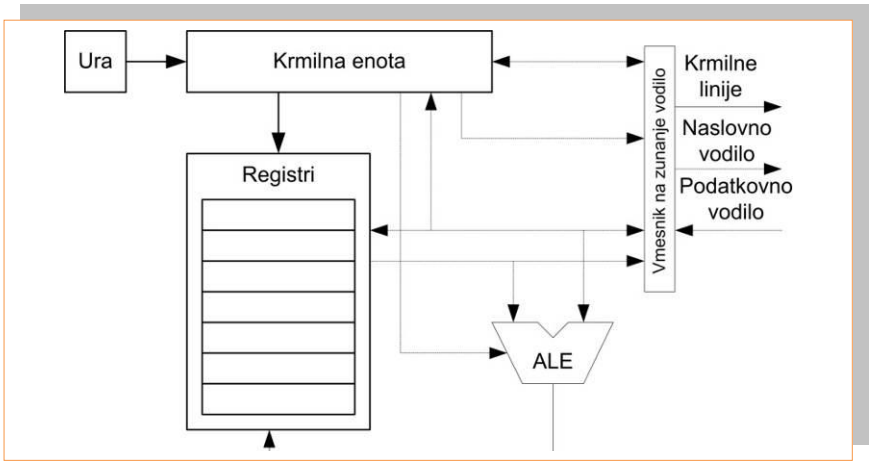


Slika 4.2: Harwardski model mikroračunalnika

Vir: Lastni

Mikroprocesor je centralno procesna enota, izdelana v VLSI tehnologiji na enem ali več polprevodniških čipih, ki so združeni v celoto. V osnovi je sestavljen iz **krmilne enote**, **enote za obdelavo podatkov** (ALU) ter notranjega pomnilnika (registri) . Poleg tega ima vmesnik za priključitev na sistemsko vodilo, ki predstavlja hrbtenico mikroračunalnika.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



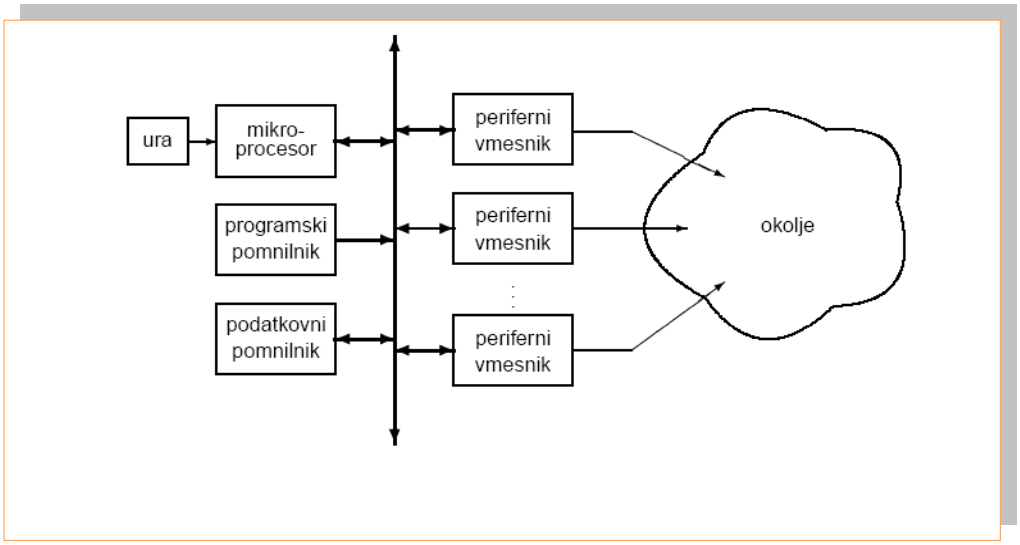
Slika 4.3: Hipotetični model mikroprocesorja

Vir: Lastni

Mikroračunalnik je računalnik sestavljen na osnovi mikroprocesorja. Sestavljajo ga:

- Mikroprocesor ⇒ upravlja delovanje mikroračunalnika. Deluje po taktu, ki ga daje ura. (npr. vaš osebni računalnik ima takt npr. 2,5 GHz).
- Programski pomnilnik ⇒ od koder mikroprocesor bere strojne ukaze in jih izvaja.
- Podatkovni pomnilnik ⇒ kjer so shranjeni podatki, ki jih procesor obdeluje.
- Periferni vmesniki ⇒ preko katerih je vzpostavljen stik med mikroračunalnikom in okoljem v katerem deluje (npr. tipkovnica, prikazovalnik …).
- Vodilo ⇒ enote so med sabo povezane z vodilom, po katerem se pretakajo naslovi, podatki in krmilni signali.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



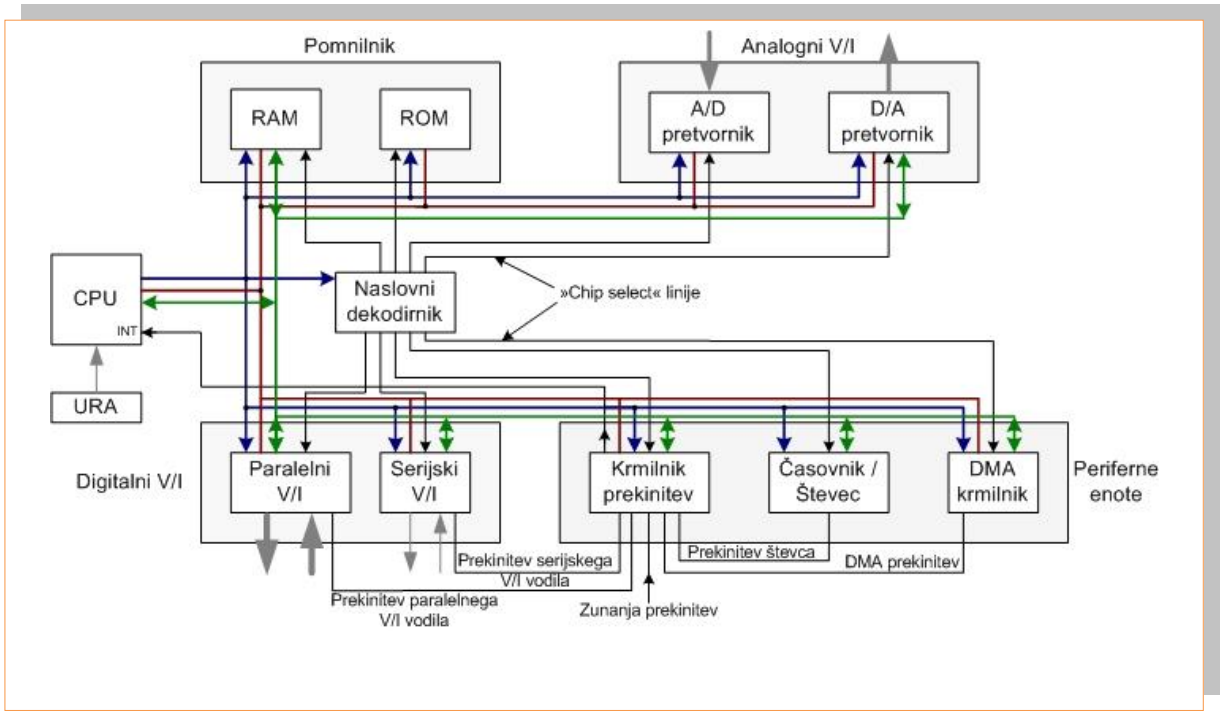
Slika 4.4: Princip delovanja mikroračunalnika

Vir: Lastni

Mikroračunalnik je sestavljen iz treh komponent:

- strojne opreme (hardware) ⇒ mikroprocesor, pomnilnik, vhodno-izhodne enote…
- programske opreme (software) ⇒ sistemska in uporabniška
- in strojno programske opreme (firmware) ⇒ Tovarniško trajno vgrajena v pomnilnik v mikroprocesorja (npr. Bios pri osebnem računalniku)

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 4.5: Blokovna shema vgrajenega mikroračunalniškega sistema

Vir: Lastni

Centralno procesna enota (CPU) nadzoruje delovanje mikroprocesorja. Deluje z določenim taktom ure. Analogni vhodni kanal mikroračunalnika je zagotovljen z A/D pretvornikom in se uporablja za branje analognega merilnega signala. Analogni izhodni kanal je zagotovljen z D/A pretvornikom in se uporablja za procesiranje anolgnih veličin na vhodne enote izvršilnih členov(npr. motorjev). Paralelna vhodno-izhodna enota zagotavlja digitalne vhode in izhode.

Serijska vhodno/izhodna enota se uporablja za komunikacijo z drugimi mikroprocesorskimi sistemi. **Pomnilnike** delimo na delovne ali hitre pomnilnike (polprevodniška tehnologija, npr. RAM pomnilnik) in periferne pomnilnike (magnetni mediji, npr.disk). V delovnem pomnilniku so naloženi programi, ki jih izvaja mikroprocesor in podatki, ki jih ti programi obdelujejo.V pomnilniku hranimo podatke in programe. Bistveni del pomnilnika je pomnilni element. To je element, ki sposoben za določen čas ohraniti neko stanje (informacijo). Primeri takšnih elementov so

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

kondenzatorji in tuljave (Naboj oziroma magnetni pretok se ohrani tudi po doklopu vira energije), magnetni mediji (trakovi, diski), flip flopi in bistabilni multivibratorji.

Nekateri elementi vzdržujejo informacijo trajno, drugi pa samo dokler so priključeni na vir energije, tretji pa kratek čas in je potrebno njihovo stanje neprestano obnavljati.

V osnovi delimo pomnilnike na bralno-pisalne (read/write, RAM) in bralne pomnilnike (read only, ROM). Pri bralno – pisalnih pomnilnikih je vsak celica dostopna za pisanje in branje, medtem ko lahko iz bralnih celic samo beremo podatke. Bralne pomnilnike vpišemo ob snovanju sistema in nato ostanejo v tem stanju.

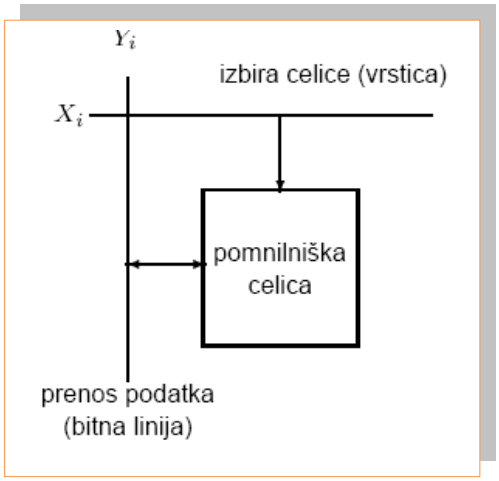
Nekateri tipi pomnilnikov:

- ROM (Read Only Memory, iz pomnilnika lahko beremo podatke),
- RAM (Random Access Memory, branje/vpisovanje podatkov iz/v pomnilnik, poznamo⇒statični ⇒ dinamični RAM),
- EPROM (Eresable Programable ROM, zbrisljiv ROM z UV svetlobo),
- EEPROM (Electrical Erasable Programable ROM, električno zbrisljiv programirljiv ROM),
- FLASH ROM.

…

Pomnilnik je sestavljen iz pomnilniških celic, ki hranijo en bit informacije. Vsaka celica ima dva priključka. S prvim celico izberemo (naslovimo), z drugim pa prenesemo podatek vanjo ali iz nje.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 4.6: Pomnilniška celica pomnilnika

Vir: Lastni

Mikroračunalnik vsebuje tudi **krmilnik prekinitev**. Krmilnik določi prioriteto določene prekinitve in jo posreduje mikroprocesorju. Trenutni program se prekine in nadaljuje po prekinitvi. **Programabilni števec/timer** se uporablja za štetje vhodnih signalov oziroma proženje izhodnega signla po dolоločenem času. **DMA** enota se uporablja za neposrednii dostop do pomnilnika. DMA krmilnik dodeljuje pomnilnik posameznim perifernim napravam. V tem času mikroprocesor ni aktiven.

SISTEMSKO VODILO

Vse enote mikororačunalnika so z CPU povezane z sistemskim vodilom. Sistemsko vodilo je sestavljeno iz paralelnih povezav in sicer iz:

- **podatkovnega** vodila (npr. 8 linij),
- **naslovnega** (adresnega vodila, 16 linij),
- **krmilnega vodila** (št. linij po potrebi).

CPU »vidi« vse ostale enote (vhodno-izhodne enote, DMA krmilnik, pomnilnike…) kot navidezne lokacije znotraj nabora 2^{16}-1 = 65535 lokacij. Večino z adresnim vodilom določenih lokacij zavzamejo pomnilniške celice (lokacije). CPU lahko shrani ali prebere

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

8 bitni podatek (8 bitno podatkovno vodilo) iz vsake od 65535 lokacij. Podatki (8-bitni) se prenašajo po podatkovnem vodilu. Krmilno vodilo daje informacijo, npr pomnilniku (RAM), ali se naj podatek vpisuje ali bere iz naslovne lokacije. Podatek zapišemo s pomočjo WRITE linije, ki jo v primeru pisanja podatka postavimo na »0«. Podatek beremo s pomočjo READ linije, ki jo moramo v primeru branja postaviti na logično »0«. Naslovno vodilo je izhod iz CPU. V primeru, da je v mikoračunalniku tudi DMA krmilnik lahko tudi le-ta naslavlja pod nadzorom gospodarja (CPU, master, DMA, slave). Naslovno vodilo je torej vhod v ostale pod enote mikroračunalnika. Krmilne liniji so lahko izhodne linije CPU in krmilijo preriferne in ostale enote mikroračunalnika. Nekatere krmilne linije pa posredujejo informacije CPU in so torej vhodne krmilne linije. (RESET, različne prekinitve, READY linija itd…).

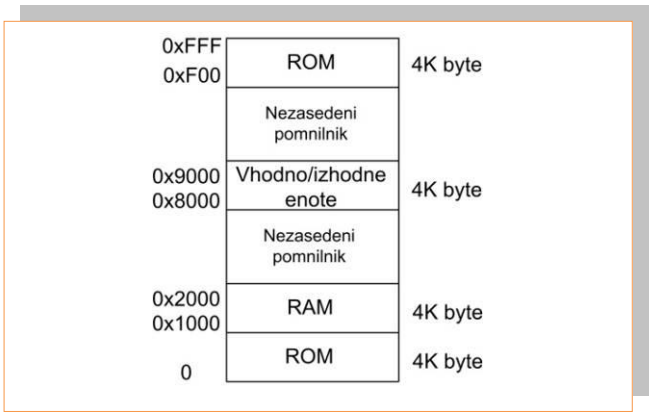
Da se prepreči popolni kaos, če bi imele nekatere enote logično »1« druge pa logično »0« na isti podatkovni liniji, kar bi pomenilo kratki stik na podatkovnem vodilu, se na vsaki enoti na njenem podatkovnem vhodu/izhodu uvede še tako imenovan pogoj tretjega stanja (tristate condidtion). Ko enota ni direktmno naslovljena (krmilna linija ENABLE=0) se postavi podatkovni vhod/izhod tretje ozriroma visokoimpedančno stanje. V praksi to pomeni, da so vsi podatkovni vhodi/izhodi fizično odklopljeni od podatkovnega vodila. Pogoj tretjega stanja omogoča, da si lahko več enot deli isto podatkovno vodilo.

RAZDELITEV PROSTORA POMNILNIKA

V primeru, da ima podatkovno vodilo 8 linij predstavlja podatek 8 bitno binarno število. 8 bitov predstavlja 1 byte, ki lahko predstavlja števila 0 – 255 = (2^n-1, n je število linij torej 2^8-1 = 255). Naslovno vodilo ima 16 linij in torej lahko naslovimo lokacije od 0 – 65535. To število predstavlja 64 k bytov pomnilniškega prostora (1 kbyte = 1024 bytov).

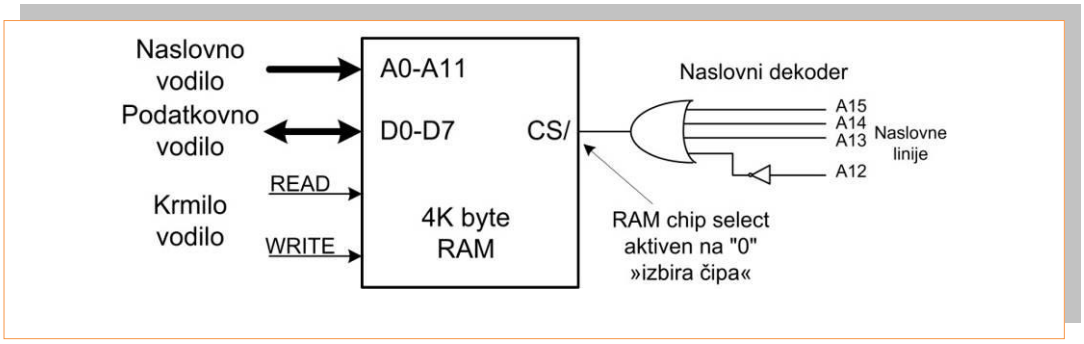
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

V splošnem obstajajo 16 bitni in 32 bitni mikroračunalniki, ki imajo 16, 24 ali pa celo 32 bitna naslovna vodila. Del pomnilnika zasedejo tudi vhodno izhodne enote (lokacije 0x8000 do 0x9000, 4kbyte pomnilnika), ROM zaseda pomnilniški prostor od 0xF000 do 0Xfff(4kbyte) ter od 0x000 do 0x1000(4kbyte) ter RAM od 0x1000 do 0x2000(4kbyte). Ves ostali pomnilni prostor je nezaseden.



Slika 4.7: Pomnilniško področje mikrokrmilnika

Vir: Lastni

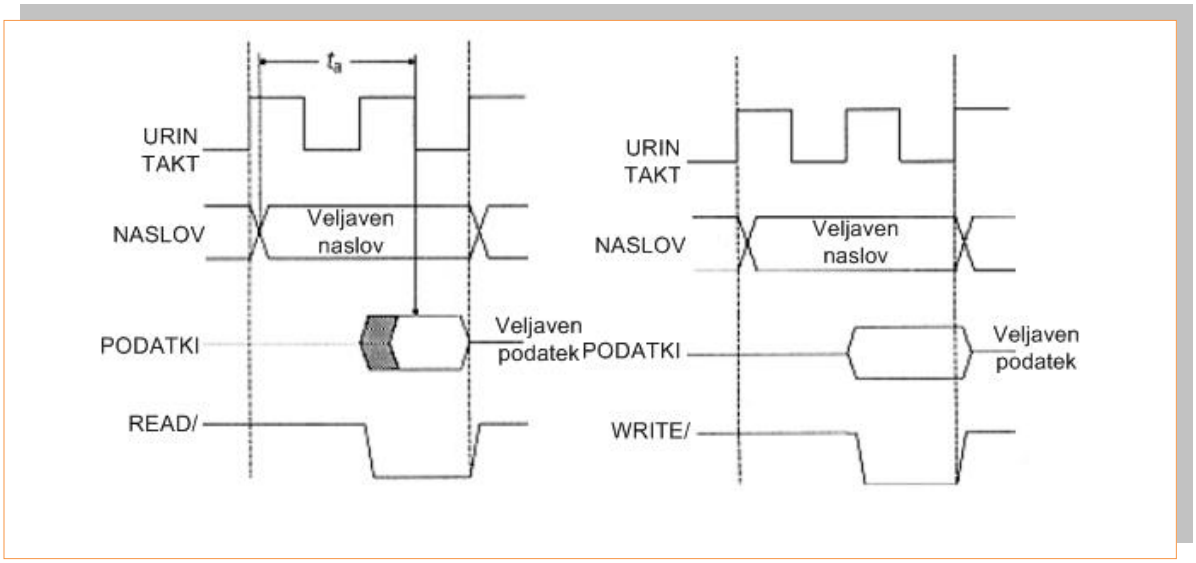


Slika 4.8: Naslovni dekoder

Vir: Lastni

Za izbiro posameznega pomnilnika se uporablja naslovni dekoder, ki je sestavljen iz logičnih vrat. Pomnilnik ima vhod za izbiro posameznega pomnilnega elementa (CS/, chip select).

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 4.9: Časovni diagram sistemskega vodila, bralni cikel in pisalni cikel

Vir: Lastni

CPU je odgovorna za krmiljenje adresnega, podatkovnega in kontrolnega vodila, zato se imenuje »gospodar vodila« (bus master).

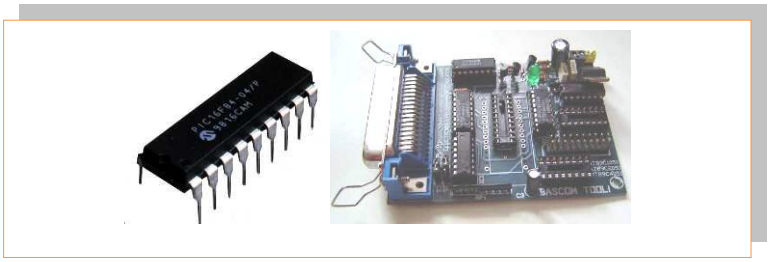
Na sliki 4.9 vidimo, da CPU deluje po taktu ure. Takt dobiva kot vhodni signal oziroma signal ure (clock) iz generatorja ure. Generator generira serijo pravokotnih signalov s točno določeno širino in frekvenco pulza. Vse akcije znotraj CPU in posledično na sistemskem vodilu, V/I enotah in pomnilniku se izvajajo sinhrono z urinim taktom. Spremembe signalov na sistemskem vodilu se dogajajo po negativnih ali pozitivnih frontah urinega signala.

Kot vidimo na sliki 4.9 **bralni cikel** traja dva urina cikla. Po prvem ciklu je prebran naslov pomnilniške lokacije v CPU-ju in ta naslov je prenesen na naslovno vodilo. Dekodirna logika najde in ustrezno izbere pomnilniško lokacijo pomnilnika. V tem času so podatkovni izhodi pomnilniškega čipa v visoko impedančnem (tristatre stanju). Da se

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

tristate stanje prekine mora pomnilnik sprejeti READ=0 signal. Nato se po krajši zakasnitv na podatkovnem izhodu pomnilnika pojavijo podatki. To se zgodi na začetku drugega urinega cikla.

Na naslednji negativni fronti se vsebina podatkov kopira v register CPU-ja. Pri **pisalnem ciklu** (slika 4.9) moramo postaviti WRITE=0 linijo. V primeru počasnega pomnilnika dodamo v bralni cikel en ali več čakalnih urinih ciklov. Tako zagotovimo dovolj časa za branje počasnih V/I enot pomnilnikov. Ta način branja se imenuje bralni cikel z dodanim čakalnim stanjem.



Slika 4.10: Integrirano vezje (čip) mikrokrmilnika ter programator

Vir: Lastni

Na sliki 4.10 vidimo primer mikrokrmilnika. Na tržišču najdemo več proizvajalcev in tipov mikrokrmilnikov. Ločijo se po zmogljivosti, npr po velikosti pomnilnika, urinem taktu, številu vhodov/ izhodov itd. Na sliki 4.10 vidimo primer programatorja v katerega vstavimo čip in ga sprogramiramo. Preko vmesnika na osebnem računalniku, kjer napišemo program za delovanje naše aplikacije nato prenesemo program na mikrokrmilnik s pomočjo omenjenega programatorja.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Slika 4.11: Programabilni industrijski krmilnik Siemens Simatic

Vir: Lastni

Na sliki 4.11 vidimo industrijski programabilni krmilnik. Zgrajen je na podobnem prncipu kot mikrokrmilnik. Cel sistem je bolj robusten in odporen na zunanje motnje (temperatura, vlaga, vibracije..), hkrati pa prilagojen za delo v realnem času.

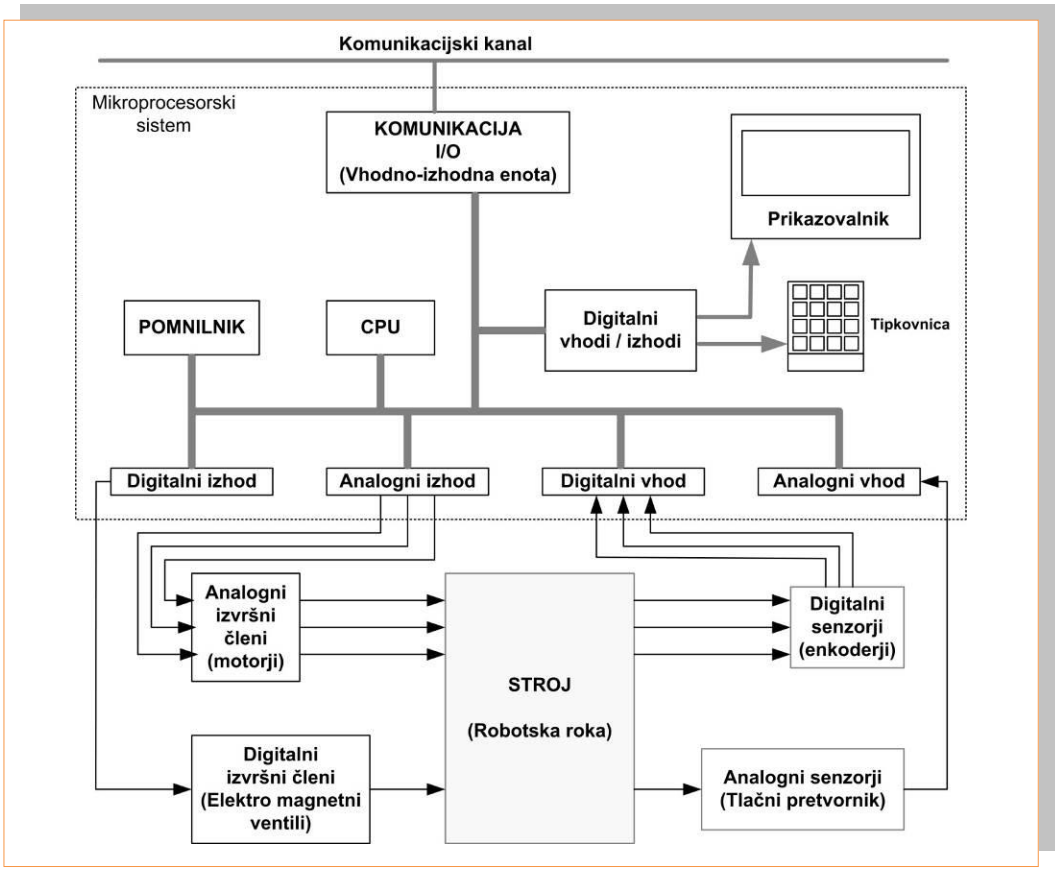
Podobno kot mikrokmilnik ima tudi ta industrijski krmilnik CPU, vhodno/izhodne enote (vhodi-senzorji, izhodi-npr motorji itd), periferne enote (osebni računalnik, touch panel…), pomnilnik (običajno pomnilniška kartica) na katerega shranimo program itd.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

MIKRORAČUNALNIK ZA DELO V REALNEM ČASU

Mikroračunalnik za delo v realnem času (real-time) se široko uporablja v industrijskih aplikacijah kot element regulacije in informacijskega procesiranja podatkov. Najpomembnejša lastnost mikroračuinalnika je **hitrost izvajanja ukazov**. Pomembna lastnost je tudi **čas odziva**. Čas odziva pri običajnem računalniku ko prikazuje sliko na zaslon ni katastrofalen za delovanje celotnega sistema. Operater čaka nekaj sekund na sliko. Po drugi strani pa mora mikoračunalnik, ki nadzoruje let letala pravočasno reagirati (v nekaj milisekundah). V nasprtnem primeru lahko pride do katastrofalne napake. Zato poznamo **mikroračunalnike strogega realnega časa** in **mikroračunalnike mehkega realnega čas**a. Mikroračunalniki realnega časa se uporabljajo povsod, kjer je od reakcije odvisno človeško življenje ali velika materilan škoda (letalstvo, vesoljska tehnika…)

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 4.12: Mehatronska aplikacija z mikroračunalnikom

Vir: Lastni

Na sliki 4.12 vidimo blokovno shemo mikroračunalnika za krmiljenje stroja, robotske roke itd. Sistem je sestavljen iz centralno procesne enote (CPU), programskega pomnilnika, podatkovnega pomnilnika in različnih vhodno/izhodnih enot. Funkcija robotskega krmilnika zgrajenega z mikroračunalnikom je regulacija posameznih osi robota. Vsaka posamezna os skupaj s senzorji in izvršnimi členi ter mikroprocesorjem tvori posamezno regulacijsko položajno zanko. Mikroračunalnik mora računati hitrost, pospešek in ploložaj vsake posamezne osi v določenem trenutku, smer gibanja itd. To zahteva CPU z dovolj velikimi sposobnostmi (hitrost izvajanja). V primeru robota je potrebno izvesti veliko število aritmetičnih operacij (inverzni in direktni dinamični model) v kratkem času. Zato se pogosto doda k CPU še dodatni aritmetični procesor, ki zmore računati matematične funkcije zelo hitro. V takšnih aplikacijah je običajno, da se čas

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

izvajanja ukazov v mikroračunalniku razdeli. Posamezne naloge se izvajajo določen čas, kar se imenuje multitasking v realnem času.

Mehanizem robotske roke je povezan preko digitalnih senzorjev(inkrementalni ali absolutni dajalnik stanja) na digitalne vhode mikroračunalnika. Na analogne vhode mikroračunalnika so priključeni tlačni senzorji, ki merijo tlak v pnevmatskem prijemalu. A/D pretvornik pretvori analogno vrednost v vrednost razumljivo mikroračunalniku (digitalno vrednost). Na digitalni izhod je povezan elektromagnetni ventil, ki krmili tlak v pnevmatskem robotskem prijemalu. Na analogni izhod so povezani električni motorji, ki zagotavljajo pomik posamezne osi. Robotski sistem redko deluje kot samostojna enota, zato je povezan z ostalimi enotami.(CNC stroji, ostali roboti, tekoči trakovi, nadzorni sistem) preko vhodno/izhodnega komunikacijskega kanala (vmesnik stroj-stroj). Preko tega komunikacijskega kanala si posamezni elementi industrijskega procesa izmenjujejo podatke potrebne za delovanje industrijskega procesa. Robotski krmilnik ima običajno tudi tipkovnico in prikazovalnik, ki služi za komunikacijo med človekom in strojem (vmesnik človek-stroj). Preko tega vmesnika operater vnaša ali popravlja uporabniški program, parametre itd.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

5 NAČINI PROGRAMIRANJA RAČUNALNIKA

V prvem poglavju bomo spoznali pomen računalniškega programa, značilnosti programskih jezikov ter programska orodja in faze pri izdelavi programa. Obravnavani so ***pojmi računalnik, računalniški program*, *programer* in *programiranje*** ter predstavljeni ***programski jeziki s primeri***, razdeljeni glede na način programiranja, na čas prevajanja in na način vnosa. Predstavljeni so tudi programi za zajemanje in obdelavo podatkov Lego Mindstorm, Crocodile Technology in Labview (http://www.ni.com/labview/applications/) ter osnovni pristop k programiranju.

RAČUNALNIK, RAČUNALNIŠKI PROGRAM IN PROGRAMIRANJE

Računalnik *(computer)* je stroj za obdelavo informacij, ki izvaja operacije ali instrukcije po računalniškem programu.

Računalniški program *(computer program)* so navodila stroju, kaj in kako naj obdeluje informacije.

Programiranje *(programming)* je zapisovanje naloge v procesorju razumljivi obliki.

Programer *(computer programmer)* ali razvijalec programske opreme piše računalniške programe. Pri tem mora poznati probleme in rešitve nalog, ki jih bo pozneje reševal računalnik s pomočjo napisanega programa.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

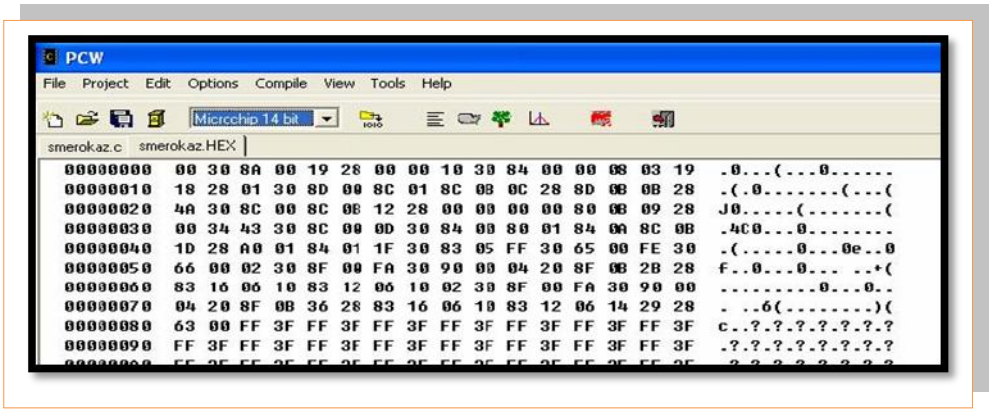
RAZDELITEV PROGRAMSKIH JEZIKOV S PRIMERI

a) **Strojna koda računalnika**, **strojni jezik** ali **strojno besedilo programa** (angl. *Machine code* ali *Machine language*) je besedilo oziroma koda v izvršljivih datotekah, ki so jih iz izvornega besedila ustvarili prevajalniki ali zbirniki. Strojno besedilo programa je sestavljeno iz zaporedij izvršljivih strojnih ukazov, ki jih

na osebnih računalnikih izvaja centralna procesna enota. Vsak procesorski sistem ima svojo arhitekturno zasnovo (platformo), kar pomeni, da je strojni jezik med različnimi procesorji zelo drugačen, a enak za tiste na enaki zasnovi. Za vsako platformo je tako treba uporabiti drugačen prevajalnik (angl. *Compiler*), da prevede izvorno kodo v strojno. Platforma je skupek lastnosti vseh delov računalnika, čeprav je najbolj odvisna od procesorske zasnove, v kar se štejejo tudi operacijski sistem (angl. *Software*) in drugi pomembni programi ali strojni deli (angl. *Hardware*).

Kljub temu, da centralna procesna enota razume objektno kodo neposredno, potrebujemo za uspešno izvajanje na danem operacijskem sistemu prav tako določen izvršni zapis (angl. *Executable format*), ki je določen z ABI vmesnikom (angl. *Application Binary Interface*) operacijskega sistema. V izvršnih datotekah se torej nahaja strojna koda določenega izvršnega zapisa, kar je osrednji razlog za neprenosljivost računalniških programov med posameznimi operacijskimi sistemi. izvršni zapis ELF (angl. *Executable and Linkable Format*) uporabljajo GNU/Linux in družina BSD, Mac uporablja Mach-o, Windows pa PE (angl. *Portable and Executable Format*), lahko pa je izvršna datoteka s strojno kodo tudi brez izvršnega zapisa. Datoteko, ki jo je moč izvajati brez operacijskega sistema na centralni procesni enoti, imenujemo ploščata izvršna datoteka (angl. *Flat Binary File*). Strojno kodo pogosto označujemo kot prvo generacijo programskih jezikov.
Primer strojne kode prikazuje slika 5.1.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

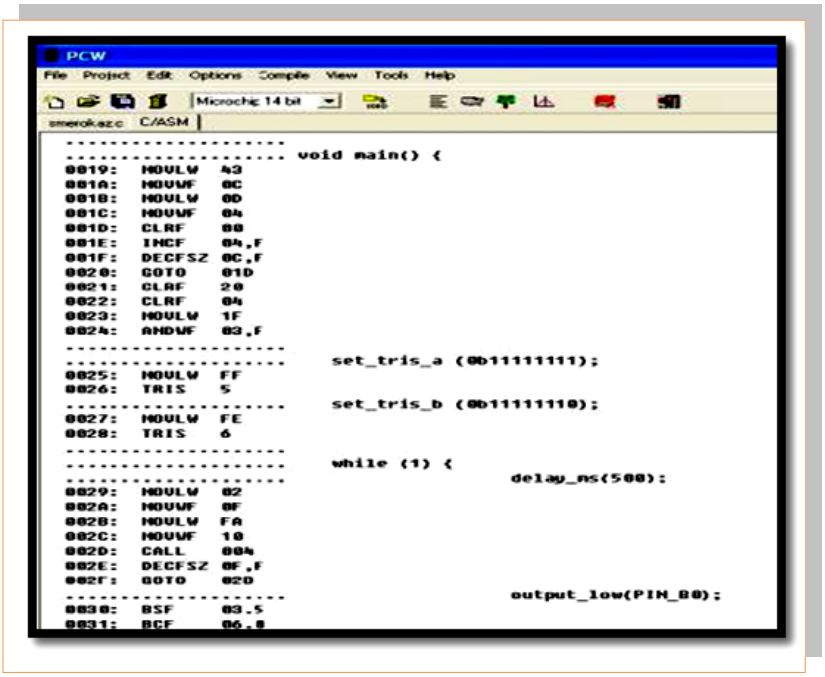


Slika 5.1: Primer programa v strojni kodi

Vir: Lastni

b) **Zbirnik** (angl. *Assembly language, assemble* – sestavljati) je nizkonivojski programski jezik druge generacije (jezik prve generacije je strojna koda), ki je napisan z mnemoniki. Mnemoniki predstavljajo berljive inačice dvojiških zaporedij (ničle in enice) – te je potrebno sestaviti tako, da dobimo kodo, ki bo razumljiva centralnemu procesorju. Natančneje, mnemoniki predstavljajo ukazne kode (angl. *Operation codes*, skrajšano *Opcodes*), ki so v centralni procesni enoti definirani po ISA arhitekturi (angl. *Instruction Set Architecture*). To kodo moramo običajno še povezati z določenimi strukturami, če želimo dobiti izvedljiv delujoč program. Določene programske opreme zbirnikov, kot je npr. FASM, samo zamenjajo mnemonike in operande oziroma parametre z ustreznimi instrukcijami v strojnem programskem jeziku. Na ta način dobimo ploščate binarne izvršilne datoteke, ki vsebujejo strojno kodo z izjemno algoritemsko učinkovitostjo, kar pa je sicer odvisno od izkušenj posameznega računalniškega programerja. Programska koda v zbirniku je prikazana na sliki 5.2.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



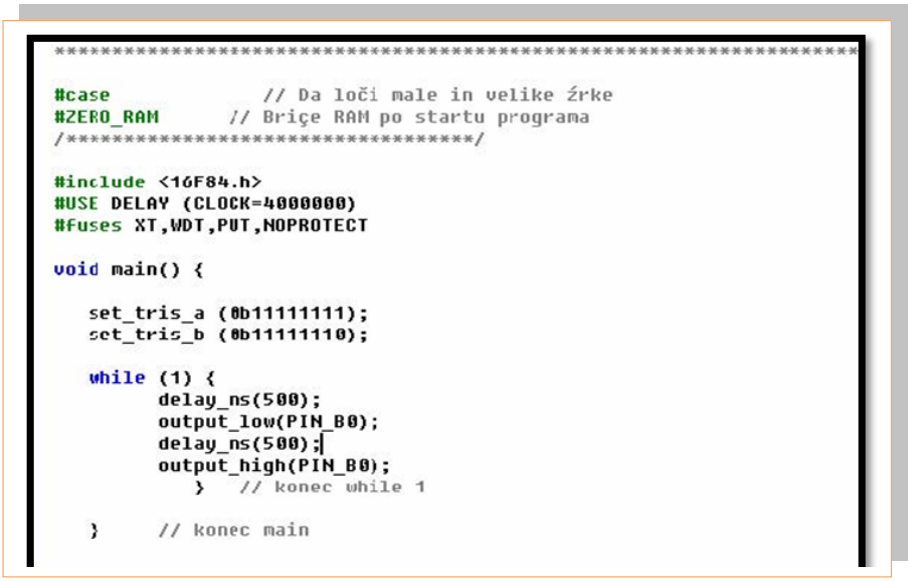
Slika 5.2: Primer programa v zbirniku

Vir: Lastni

c) **Visokonivojski jezik** (angl. *High-level language*) je programski jezik, ki je zasnovan tako, da ustreza programerjevim zahtevam in ni odvisen od interne strojne kode konkretnega računalnika. Ti jeziki se uporabljajo za reševanje problemov in jim večkrat pravimo tudi problemsko orientirani jeziki (npr. BASIC je bil zasnovan tako, da je začetnikom omogočil hitro učenje, COBOL se uporablja za pisanje poslovnih programov, FORTRAN pa za programe, ki rešujejo znanstvene in matematične probleme). Nizkonivojski jeziki močno odražajo značilnosti strojne kode določenega računalnika in jim zato pravimo tudi strojno orientirani jeziki, kar je v nasprotju z visokonivojskimi jeziki. Visokonivojskih jezikov se je relativno enostavno naučiti, saj so njihovi ukazi podobni človeškemu jeziku, zato programerju ni treba podrobno poznati notranjega ustroja računalnika. Vsak ukaz visokonivojskega jezika je ekvivalenten več strojnim ukazom, zato so visokonivojski programi bolj kompaktni kot ekvivalentni nizkonivojski programi. Vsak visokonivojski program pa moramo prevesti v strojni jezik, preden ga lahko poženemo (s prevajalnikom ali z interpreterjem. Visokonivojski jeziki so zasnovani tako, da so prenosljivi. To pomeni,

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

da program, napisan v visokonivojskem jeziku, lahko poženemo na vsakem računalniku, ki ima prevajalnik ali interpreter za ta jezik. Visokonivojski jezik je C++ prikazan na sliki 5.3.



```
*************************************************************************

#case              // Da loči male in velike źrke
#ZERO_RAM       // Briçe RAM po startu programa
/*********************************/

#include <16F84.h>
#USE DELAY (CLOCK=4000000)
#fuses XT,WDT,PUT,NOPROTECT

void main() {

   set_tris_a (0b11111111);
   set_tris_b (0b11111110);

   while (1) {
         delay_ns(500);
         output_low(PIN_B0);
         delay_ns(500);
         output_high(PIN_B0);
            }   // konec while 1

   }     // konec main
```

Slika 5.3: Primer programskega jezika v C++

Vir: Lastni

Za približanje strojnega ukaznega jezika človeškemu uporabimo programski jezik kot vmesno stopnjo. Programski jezik ni enak naravnemu jeziku. Na sliki 5.3 je prikazana programska koda v okolju C++, kjer pišemo program z naborom ukazov, ki jih ponuja programsko orodje npr. while. Ta nam predstavlja zanko, z ukazom include pa lahko vključimo potrebno knjižnico.

Naravni jezik ima zapletena pravila in obsežna slovnična pravila. Programski jezik je nedvoumen in formalno opisljiv – je bliže strojnemu jeziku.

Programski jezik mora omogočati:

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

- **opis problema** (opis podatkov, rezultatov in relacij med njimi). Primer: izračun produkta – opis števil, rezultat – njihov produkt;
- **opis postopka** (opis zaporedja korakov, ki nas pripelje do rezultata). Algoritem za množenje, zapisan v osnovnih korakih.

Programski jezik je zbirka dogovorjenih ukazov, običajno v angleščini, ki opravijo določeno akcijo. Višji programski jezik ni vezan na tip mikroprocesorja. Izbrati moramo ustrezen prevajalnik, ki napisan program iz izvorne kode (npr. C) prevede v ustrezno strojno kodo razumljivo procesorju (npr. EXE). Vsak ukaz se prevede v več strojnih kod. Programer potrebuje več programskih orodij. Izvorna koda se natipka v urejevalniku.

Editor – prevede se v modul strojne kode s pomočjo ustreznega prevajalnika.

Compiler – prevedeni modul se skupaj z že predhodno pripravljenimi moduli iz knjižnice (Library) združi v izvršljiv program s pomočjo povezovalnika.

Linker – delovanje programa se testira postopoma s pomočjo iskalca napak.

Debugger – običajno so vsa ta orodja združena v programskem okolju z meniji, npr. DEVCPP, CCSC, Visual Studio NET itd.

DELITEV PROGRAMSKIH JEZIKOV

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Glede na čas prevajanja:

a) **Tolmač** (INTERPRETER – RUN TIME).
b) **Prevajalnik** (COMPILER) tvori kodo za procesor, na katerem deluje tudi sam prevajalnik (npr. TurboC, Delphi za PC).
c) **Križni prevajalnik** (CROSS COMPILER) pa deluje na razvojnem računalniku (npr. PC-ju) in tvori programsko kodo za drug tip mikroprocesorja, za razne mikrokrmilnike (npr. PICC, BASCOM in industrijske krmilnike (npr. Mitshubishi-Melsec MEDOC ali Siemens STEP 7 – http://www.crocodile-clips.com/en/Crocodile_Technology/.)

Glede na način programiranja:

- **Postopkovni** – BASIC, C-jezik, PASCAL (primerni za mikrokrmilnike in večje računalnike).
- **Objektni** – C++, Delphi, JAVA (uporabljajo razrede objektov CLASS).

Glede na način vnosa programa:

a) **Besedilni (tekstualni)** – to so do sedaj omenjeni programski jeziki.
b) **Grafični** – Visual Basic, Visual C, LabVIEW.

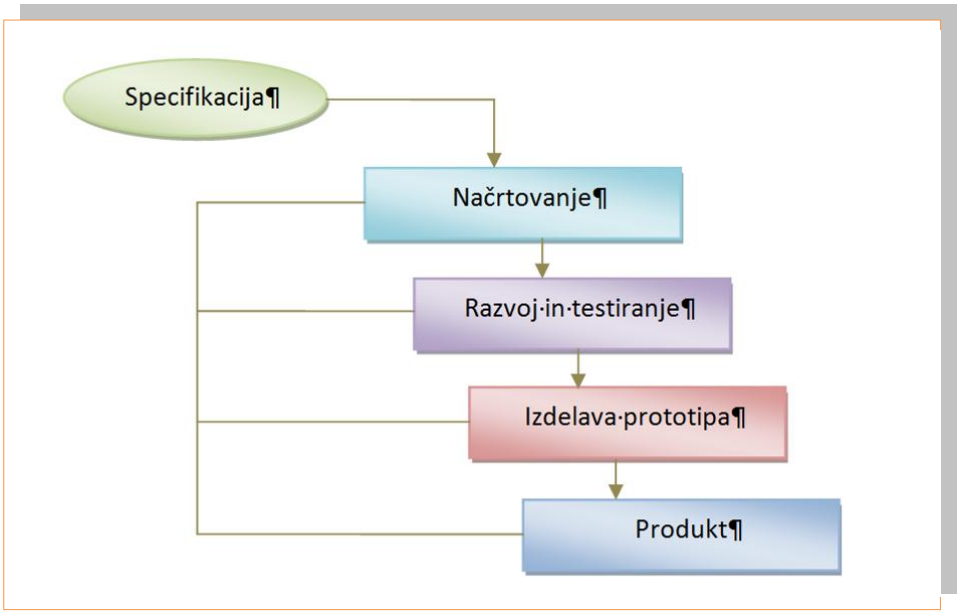
To so jeziki nove generacije, programiranje poteka v grafičnem vmesniku. Programer zlaga kocke, module, v ozadju pa se tvori ustrezna programska koda.

PRISTOP K PROGRAMIRANJU

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Razvoja programa se lotimo podobno kot razvoja drugih izdelkov ali produktov.

Proces načrtovanja in razvoja:



Slika 5.4: Proces načrtovanja in razvoja

Vir: Lastni

Faze programiranja

a) Načrtovanje – algoritem, diagram poteka.
b) Kodiranje – pisanje v izbranem programskem jeziku.
c) Prevajanje in testiranje – prevajalnik najde pravopisne napake, debugger pa omogoča, da najdemo tudi logične napake.
d) Dokumentacija programa – komentarji in opisi za poznejše razumevanje napisanega in navodila uporabnikom programa.

Algoritem

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Rešitev naloge se izvede v korakih, v **algoritmu.**

Definicija: **Algoritem je koračen opis postopka, s katerim lahko rešimo problem.**

Primer: **Izdelava programa, ki sešteje dve števili**

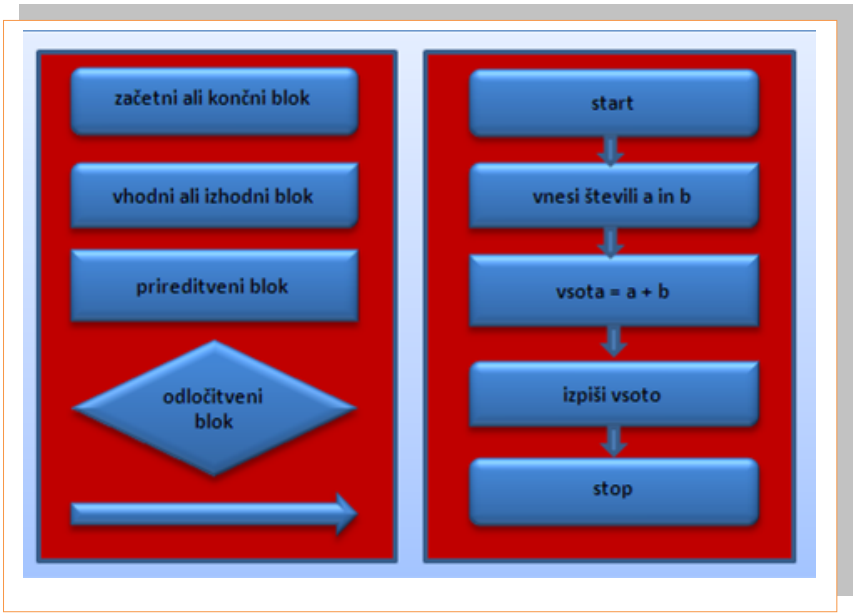
- Deklariranje spremenljivk x, y, z;
- Vnesite prvo število in ga vpišite v spremenljivko x;
- vnesite drugo število in ga vpišite v spremenljivko y;
- seštejte spremenljivki x in y;
- izpišite spremenljivko z.

Diagram poteka (Flow Chart)

Je grafično prikazan algoritem, kar pomeni, da je človeku na razumljiv način opisan potek dogodkov. Uporablja se pri načrtovanju programov in tudi pri opisovanju ostalih dejavnosti (npr. Domači zdravnik v knjižni obliki, Občinska navodila za pridobitev dovoljenj itd).

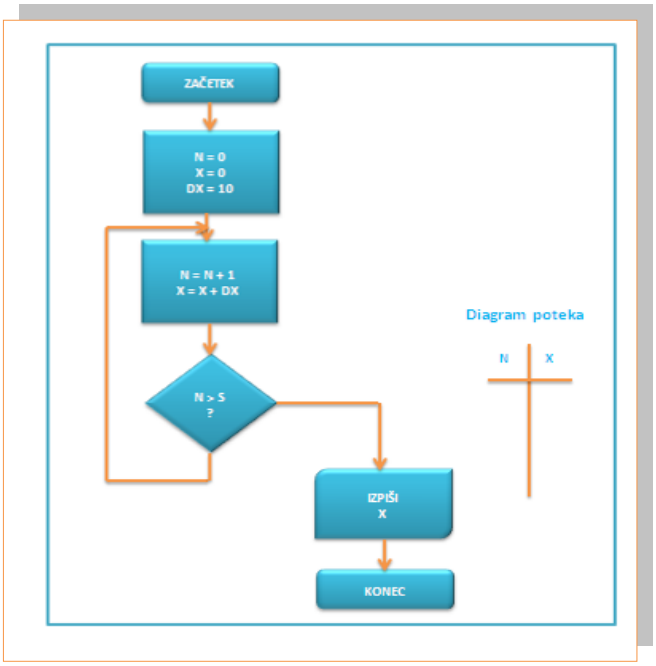
Da si razjasnite izvajanje po korakih in predvidite, kaj bo moral narediti računalnik, se je pametno lotiti programskega problema na način diagrama poteka.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 5.5: Bloki diagrama poteka

Vir: Lastni *Vir: Lastni*



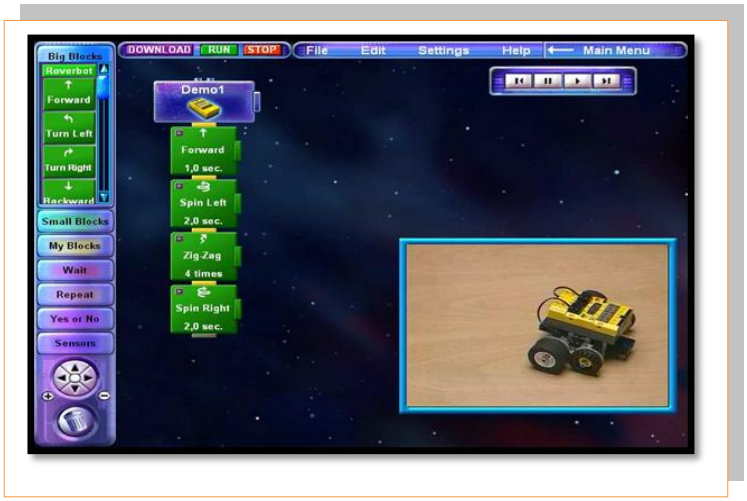
Slika 5.6: Primer diagrama poteka

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

PROGRAMIRANJE MEHATRONSKIH NAPRAV LEGO MINDSTORM IN FLOWCODE

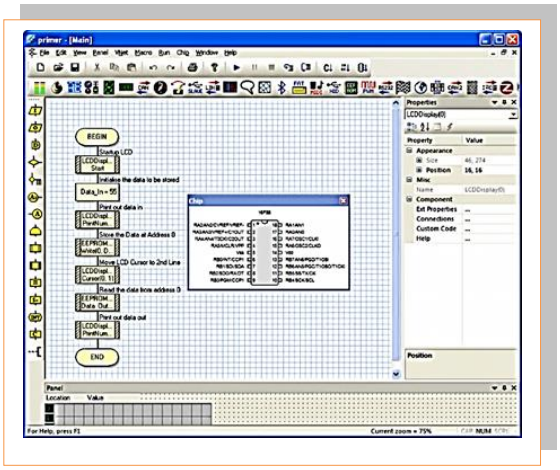
Programa LEGO MINDSTORM in SIMBOT sta zelo primerna za učenje programiranja. Iz LEGO kock sestavimo avtomobilček, ki ima vgrajen mikroračunalnik, ki mu napišemo program, da bo krmilil motorje in upošteval signale iz senzorjev.



Slika 5.7: Lego Mindstorm

Vir: http://mindstorms.lego.com/en-us/Default.aspx

Ena izmed možnosti programiranja je tudi Flowcode.



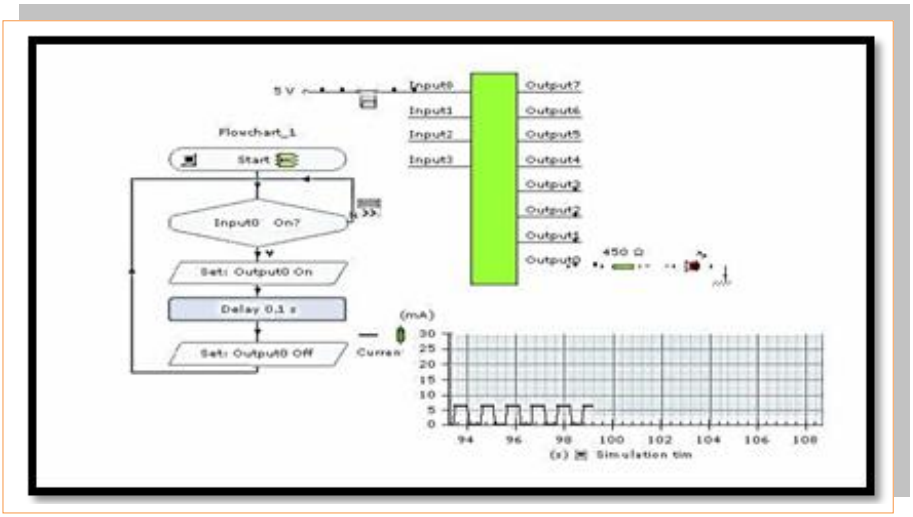
Slika 5.8: Primer programa v Flowcode

Vir: http://imagesco.com/microcontroller/flowcode-compiler.html

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

PROGRAM ZA SIMULACIJO CROCODILE TECHNOLOGY

Za učenje programiranja mikrokrmilnika PIC je primeren simulacijski program Crocodile Technology. V programu narišemo električno shemo vezja z elementi iz knjižnice, lahko pa tudi sestavimo diagram poteka programa, ki ga med simulacijo delovanja zaženemo in s tem preizkusimo.

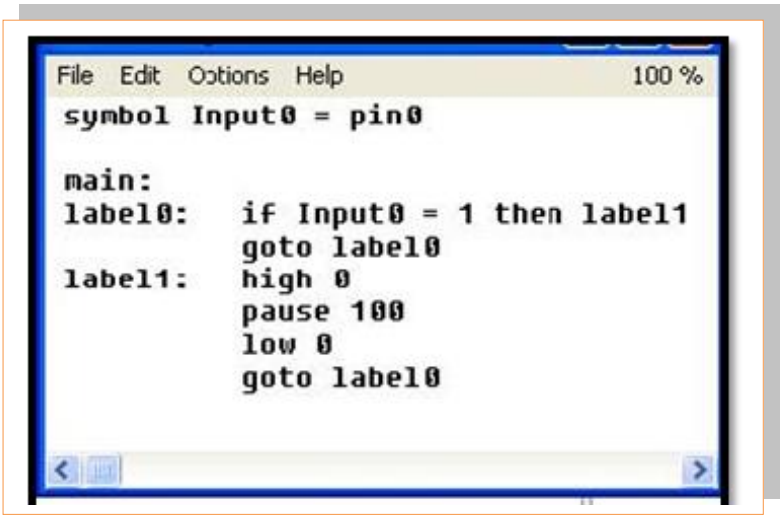


Slika 5.9: Primer programskega okolja za programiranje mobilnih robotov

Vir: www.crocodile-clips.com/en/Crocodile_Technology/

Program je možno tudi izvoziti v BASIC ali direktno preko programatorja vpisati v realni čip.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 5.10: Primer programske kode v Basicu

Vir: www.justbasic.com/learnmore.html

PROGRAMIRANJE MIKROKRMILNIKOV – PLC

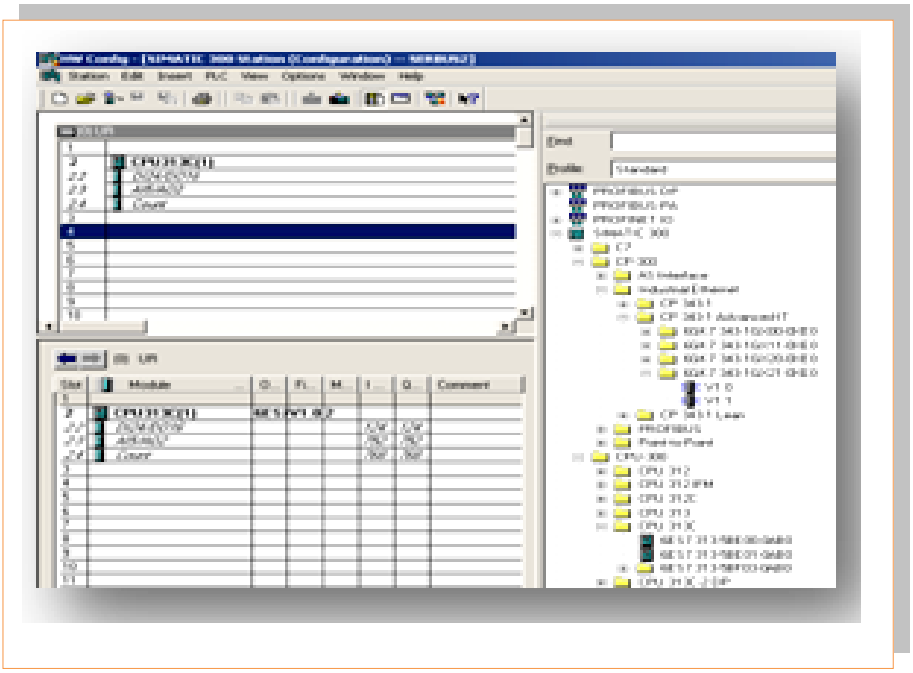
Tudi programiranje industrijskih krmilnikov je lahko grafično in hitro razumljivo.

Programska orodja za PLC podpirajo tri načine programiranja:

- preko relejske električne sheme krmilja (LADDER),
- z elementi digitalne elektronike – blok diagram (FBD),
- z besednimi ukazi – instrukcijami (INSTR),
- pa še kakšen dodatni grafični način.

Takšna programska orodja omogočajo nadzor in testiranje programa po nalaganju v krmilnik, nekatera pa tudi simulacijo delovanja kar na PC-ju (off-line).

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 5.11: Primer PLC programskih jezikov

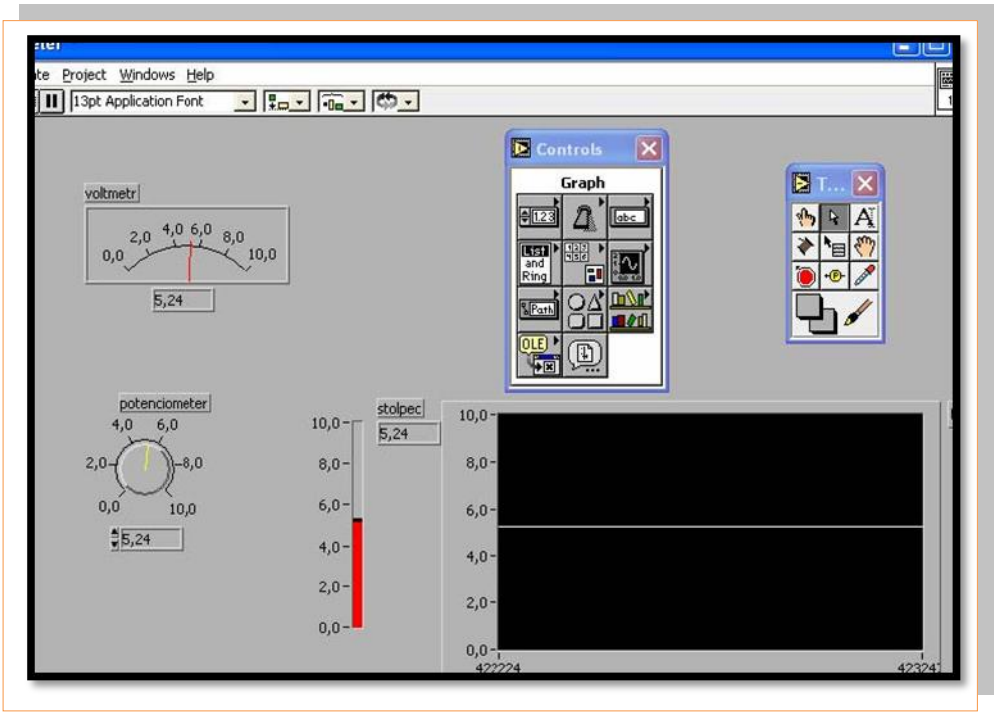
Vir: http://support.automation.siemens.com

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

PROGRAMIRANJE VIRTUALNE INSTRUMENTACIJE – LABVIEW

PC računalnik lahko uporabimo kot industrijski krmilnik pri avtomatizaciji merilnih procesov. V ta namen se dobijo skupaj z aparaturno opremo (merilni vmesniki za PC ipd.) tudi grafična programska okolja za programiranje. V Sloveniji sta zelo razširjena oprema podjetja National Instruments in njihovo programsko orodje LabVIEW (http://www.ni.com/labview/applications/).

Na sprednjo ploščo (Panel) nanesemo prikazovalnike in kontrolne gumbe za navidezni instrument (VI-Virtual Instrument).

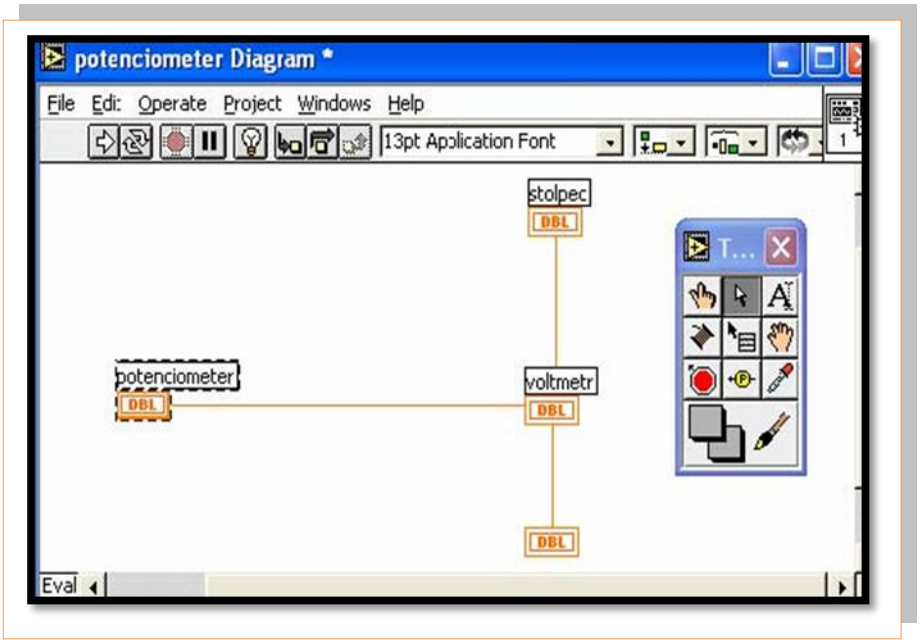


Slika 5.12: Prikaz programiranja v programu LabVIEW

Vir: http://www.ni.com/labview/

Izvore in porabnike podatkovnega toka povežete v ozadju v programskem oknu (Diagram) in zgradite programske zanke.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 5.13: Diagram

Vir: http://www.ni.com/labview/

Program lahko prevedemo v izvršljivo (exe) datoteko ali pa ga zaženemo kar na razvojnem okolju. Računalnik bere merjene podatke iz zunanjih merilnih instrumentov ali merilnih vmesnikov ter se na njihovi osnovi odloča za ustrezno akcijo preko krmilnih izhodov iz PC-ja, zapis v datoteko na disk, javljanje na ekranu ali preko interneta.

PROGRAMIRANJE NUMERIČNO KRMILJENIH STROJEV – CNC G-KODA

CNC (Computer Numerical Control) stroj je stružnica ali podoben kovinsko obdelovalni stroj, voden z računalniškim krmilnikom. G kode so standard za numerično krmiljene obdelovalne stroje.

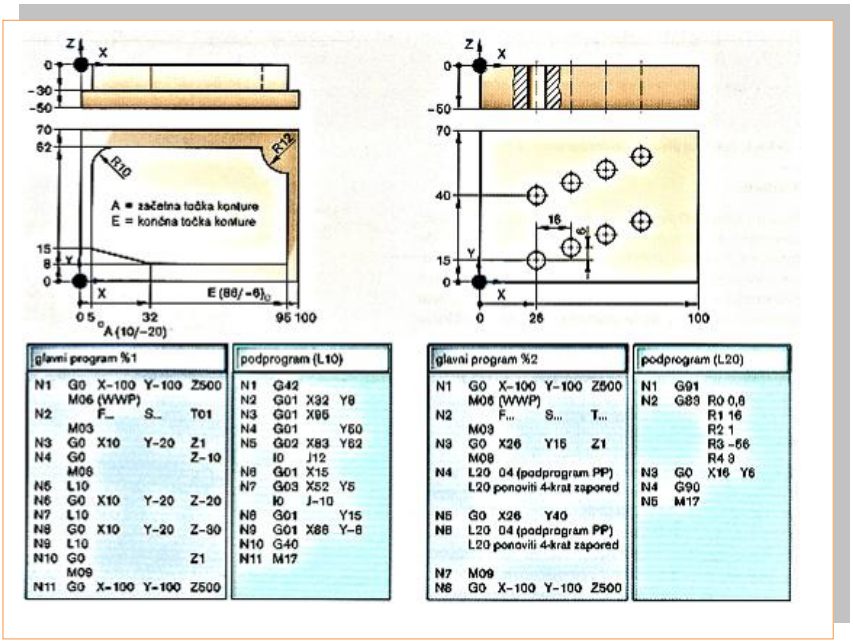
CNC programerji pišejo programe iz tehnoloških risb in jih vnašajo v krmilnik stroja. Dobri strojniški CAD programi lahko iz narisanega 3D objekta avtomatsko tvorijo G-kodo za izdelavo objekta na CNC stroju.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Slika 5.14: CNC stroj

Vir: Lastni

Slika 5.15 prikazuje primer kode za programiranje CNC rezkalnega stroja, krmiljenega s PC računalnikom. Pod vsakim od objektov je navedena tudi koda za izdelavo objekta.



Slika 5.15: Primer CNC kode z vključenimi podprogrami

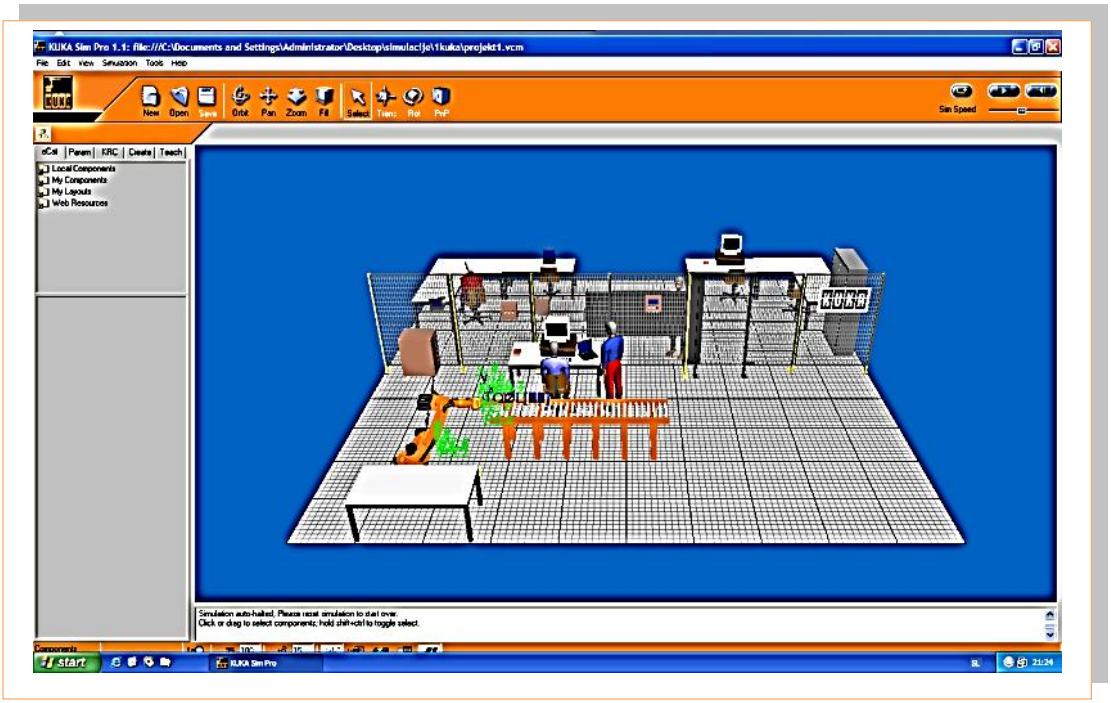
Vir: J. Bartenschlager, MEHATRONIKA 2009

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

ROBOTSKI PROGRAMSKI JEZIK: INDUSTRIJSKI ROBOTI KUKA

Simulacijski program omogoča učenje gibov in akcij v simulatorju. Zgradimo lahko robotsko celico in napišemo robotski program ter ga nato prenesemo na robotski krmilnik in preizkusimo.

Pred izgradnjo realne robotske celice je dobro napraviti simulacijo in v njej predstaviti vse robne pogoje delovanja.

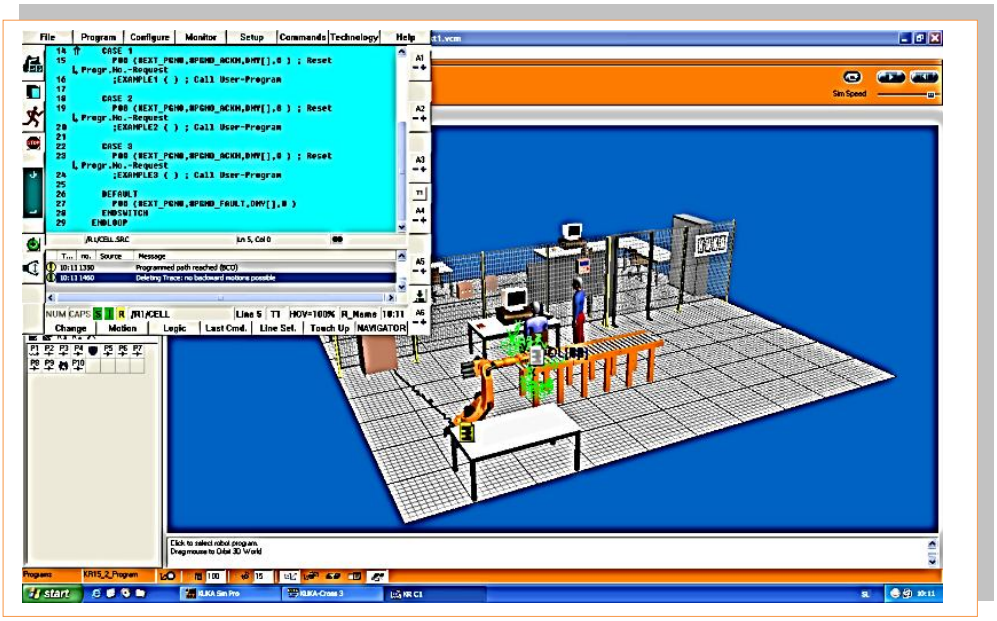


Slika 5.16: Simulacijski program KUKA Sim Pro

Vir: http://www.kuka-robotics.com/en/products/software

Poleg razvojnega programskega okolja lahko povežemo tudi simulator s programom Office Lite; ta omogoča simulacijo programirne naprave.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 5.17: Simulacijski program

Vir: http://www.kuka-robotics.com/en/products/software

Na sliki 5.17 je prikazana povezava simulacijskega programa KUKA Sim Pro s simulacijskim podprogramom operacijskega panela Office Lite. Če povežemo programski paket Kuka Sim Pro in Office Lite 4.1, lahko na osebnem računalniku izvajamo iste akcije kot na realnem robotu. Izvajamo lahko gibe, zanke, nastavljamo hitrosti in končani program direktno prenesemo preko mreže na robotski KRC krmilnik.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Slika 5.18. Programirna naprava na podlagi snemanja gibov

Vir: Lastni

POVZETEK

Programske jezike lahko v osnovi razdelimo na tri dele: na strojno kodo, zbirnik in višje programski jezike.

Strojna koda programa je sestavljena iz zaporedij izvršljivih strojnih ukazov, ki jih na osebnih računalnikih izvaja centralna procesna enota.

Zbirnik (angl. *assembly language, assemble* – sestavljati) je nizkonivojski programski jezik druge generacije (prve generacije je strojna koda), ki je napisan z mnemoniki. Posplošeno velja, da mnemoniki predstavljajo berljive inačice dvojiških zaporedij (ničle in enice), ki jih je potrebno sestaviti, da dobimo centralnemu procesorju razumljivo kodo.

Visokonivojski jezik (*high-level language*) je programski jezik, zasnovan tako, da ustreza programerjevim zahtevam. Ni odvisen od interne strojne kode konkretnega računalnika. Visokonivojske jezike uporabljamo za reševanje problemov in jim velikokrat pravimo tudi problemsko orientirani jeziki.

Pred začetkom programiranja je vsekakor potrebno sam problem, ki ga bomo reševali s programom, dobro razdelati. To napravimo z algoritmom, kjer problem razdelamo na posamezne korake in je tako lažje rešljiv.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

PONOVIMO

1. Kaj pomeni pojem »računalniški program« in kaj »programer«?
2. Kakšne so lastnosti programskega jezika in kaj omogoča?
3. Katera programska orodja potrebuje programer in čemu služijo?
4. Razdelite programske jezike in podajte nekaj primerov.
5. Skozi katere faze mora programer pri izdelavi programa?
6. Kaj pomeni beseda algoritem?
7. Napišite algoritem za avtomat za kuhanje kave in ga pojasnite.
8. Kateri simboli so uporabljeni v diagramu poteka in kaj pomenijo?
9. Kateri načini pisanja programa so uporabljeni pri programirljivih logičnih krmilnikih (PLC)?
10. Kako izgleda program za CNC stroj? Napišite vsaj 3 ukaze.
11. Kako se programirajo roboti? Opišite primer.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

6 OSNOVE PROGRAMSKIH JEZIKOV

V tem poglavju se bomo seznanili s programskimi jeziki in njihovimi osnovnimi ukazi, ponazorjenimi s primeri.

Današnji računalniki razumejo le bitni jezik, ki mu pravimo tudi strojna koda (*machine code*). Če torej želimo, da naš računalnik izvede neko nalogo ali opravilo, mu moramo le-to podati v obliki, ki jo razume, torej v enicah in ničlah. Ker pa nam ljudem razmišljanje v bitnem jeziku ni najbolj naravno, se je pojavila potreba po nekem višjem, človeku bolj razumljivem jeziku, ki ga nato lahko prevedemo v strojno kodo. Programski jezik nam omogoča, da računalniku podajamo ukaze ter naloge z vsakdanjimi izrazi, kot sta npr. »beri in piši«. Eden izmed razvojnih programskih orodij je jezik C++, ki je vsestransko uporaben. Srečamo ga pri programiranju mikroprocesorjev, izdelavi programov za izpisovanje potnih nalogov, vse pogosteje pa je prisoten pri izdelavi spletnih aplikacij.

UKAZI V C++

Program je zaporedje ukazov, programiranje pa pisanje ukazov za procesor. Ukazi so v vsakem programskem jeziku drugačni. V tem poglavju bodo opisani ukazi v programskem jeziku C++.

Poleg C++ in C# poznamo še programske jezike, kot so Visual basic, Delphi, Java, SQL, Python, itd.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Kratka zgodovina C++:

Konec sedemdesetih je Bjarne Stroustrup začel eksperimentirati z razširitvijo zelo razširjenega programskega jezika C. Programski jezik C++ je nadgradnja programskega jezika C, nekatere dodatke pa je Bjarne Stroustrup črpal iz že obstoječih jezikov, kot sta Simula67 ter Algol68. Ime C++ se prvič pojavi v letu 1983, v drugi polovici osemdesetih let pa C++ doživi korenite spremembe.

Ukazi v C++:

Glavni program (blok ukazov)

Vsi programi morajo imeti glavni program (*main*), v nasprotnem primeru se ne morejo izvajati. To je jedro, v katerega bomo začeli programirati. V vsak program vključimo metodo *main()*, ki ji sledita zavita oklepaja {}. V program moramo navesti začetek in konec programa, za blok ukazov pa je primerno, da je unikatno označen.

```
int main()   (začetek glavnega programa)
{                          - začetek bloka
cout <<  "To je program v c++" ;
return 0;
}                                  - konec bloka
```

Na levi strani je prikazan program, ki nam izpiše stavek na zaslonu – to je program C++.

Knjižnice

Poznamo veliko knjižnic. Knjižice so datoteke, kjer je napisana strojna koda funkcij. Pri pisanju programov moramo uporabljene knjižnice vključiti v program npr. *(#include<iostream>)* ter vključiti vhodno izhodni tok podatkov. Spodaj so navedene najpogosteje uporabljene knjižnice:

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

#include <iostream>	Vhodno-izhodni ukazi (knjižnica uporabljena samo v C++)
#include <stdlib.h>	Standardni ukazi
#include <string>	Za lažje delo z nizi, knjižnica je veljavna le v C++

Komentar

Komentar je namenjen programerju za lažje razumevanje programa. Pri kompleksnejših programih, ki niso dosledno komentirani, pride do velikih težav ob primeru, da želimo

karkoli spremeniti oz. popraviti. Značilnost komentarja je, da ga prevajalnik preskoči. Komentarje pišemo kratko in jedrnato; lahko so enovrstični, ki jih označimo z (///) ali večvrstični, označeni z (/* …..*/).

Spremenljivke

Spremenljivke so košček pomnilnika, ločijo se po tipu ter velikosti. Tip spremenljivke nam pove, kaj vsi ti biti, ki jih neka spremenljivka zasede, sploh pomenijo. Ali gre za črko, število ali kaj drugega.

Pravila za poimenovanje spremenljivk:

- Prvi znak mora biti črka, ne številka.
- Preostali znaki so lahko črke, številke ali podčrtaji "_".
- V imenih ni presledkov.
- Pomembna je velikost črk. Vsakokrat jih moramo zapisati povsem enako. Tako npr. *avtomobil* ni isto kot *Avtomobil*.
- Posebnost, pomembna za Slovence –šumnikov (č,ž,š) ne smemo uporabljati.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

- Navada je, da imena spremenljivk začnete z malo črko, pri sestavljenih besedah pa vsako naslednjo začnemo z veliko. Nekaj primerov: *starostOtroka*, *hitrostAvtomobila*, *velikostNoge*.
- Imena spremenljivk naj bodo smiselna: besede, ki povedo pomen spremenljivke npr. *premerKroga*, *dolzinaPalice*.

Psevdokoda za deklaracijo spremenljivke:
TIP IME_SPREMENLJIVKE;

Psevdokoda za definicijo spremenljivke:
IME_SPREMENLJIVKE = NEKA_VREDNOST;

Spremenljivka mora biti najprej deklarirana, šele potem jo lahko tudi definiramo. Ta dva koraka lahko združimo:

TIP IME_SPREMENLJIVKE = NEKA_VREDNOST;

Najmanjša velikost spremenljivke, ki jo lahko naslovimo v programskem jeziku C++, je en (1) byte, torej osem bitov, kar znese 256 (2^8) različnih stanj. Takšen tip se v C++ imenuje **char** (character), vanj pa lahko vpisujemo **cela števila**. Poleg njega poznamo tudi celoštevični **int**, ki ponavadi zasede 4 byte. Poznamo tudi manjše ter večje celoštevilčne tipe, ki zasedejo 2 ali 8 bytov, vendar pa je njihova deklaracija različna od prevajalnika do prevajalnika. Pri definiciji spremenljivk lahko uporabljamo tudi predznake.

Primer:

```
/*
Primeri deklaracij ter definicij celoštevilčnih spremenljivk.
```

Program na levi strani prikazuje deklaracijo spremenljivk. Začne se z

```
*/
// deklaracija
char a;
int  b;
// deklaracija ter definicija, uporaba predznaka
char c =  10;
int  d =+10;
char e =-10;
int f =+10;
```

prikazom celovrstičnega komentarja, sledi deklaracija spremenljivk, npr. spremenljivka b je tipa integer.

C++ pozna tudi **modifikatorje** tipov (samo za celoštevilčne tipe): *short, long, unsigned, signed.* **Short** in **long** se uporabljata v navezi s tipom **int**, določata pa velikost le-tega (odvisno od posameznih prevajalnikov).

Psevdokoda:

MODIFIKATOR TIP IME_SPREMENLJIVKE;
MODIFIKATOR TIP IME_SPREMENLJIVKE = NEKA_VREDNOST;

Primer:

```
// uporaba short ter long
shortint majhno = 100;      // velikost spremenljivke naj bi bila 2 byta
longint veliko = 10000;     // velikost spremenljivke naj bi bila 8 bytov
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Pri uporabi modifikatorjev lahko tip **int** izpustimo, saj prevajalnik ugotovi, da gre za celoštevilčni tip **int.**

Modifikatorja **signed** in **unsigned** pa določata, ali imajo naši celoštevilčni tipi tudi predznak, oz. ali lahko z njimi zapisujemo tudi negativna števila. Vsi osnovni celoštevilčni tipi znajo zapisovati tudi negativna števila (privzeti način). Na primer tip **char**. Ker je velik 1 byte, lahko z njim predstavimo 256 različnih možnosti (stanj). Tako lahko z njim zapišemo števila od –128 do +127 (tudi 0). Če na tipu **char** uporabimo modifikator **unsigned**, negativna števila odpadejo, tako da lahko vanj zapišemo števila od 0 do 255. To velja tudi za vse druge celoštevilčne tipe.

Primer:

```
// uporaba modifikatorjev signed ter unsigned

char a;          // -127 do +128, signed je privzeta vrednost
signedchar b;    // -127 do +128
unsignedchar c;  //    0 do 255
int d;           // - 21474836648 do +21474836647
unsignedint e;   //    0 do 42944967296
```

Zapisujemo lahko tudi črke ter pripadajoče simbole. Imamo več možnosti. Nekako najbolj uporabljena sta **ASCII** in **UNICODE** zapis. Zapis simbola po ASCII tabeli je velik en byte, medtem ko UNICODE uporablja za zapis 2 byta. UNICODE zapis je primeren predvsem za razvijanje prenosljivih programov za tuje trge (oz. jezike). ASCII pa nam nudi preprost ter predvsem manj pomnilniško požrešen zapis. ASCII zapis uporablja tako imenovano kodno tabelo, ki je odvisna od nastavitev sistema (jezik ter država, od

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

tod izvirajo težave s šumniki). UNICODE pa uporablja drugačen zapis, namreč vsak simbol ima svojo unikatno vrednost. Črke in simboli so v računalniku zapisani kot številke, šele ko jih začnemo "gledati" ali izpisovati, dobijo svojo pravo obliko, ki jo mi razumemo. Tako, na primer, ima velika črka A vrednost 65, črka B vrednost 66 itd.

Primer:

```
char    a ='A';        // ascii zapis
wchar_t b ='B';    // unicode zapis
char    c = 65;     // ascii zapis, tokrat uporabim kar ascii vrednost ('A')
```

Poleg teh osnovnih tipov pozna C++ tudi logični **bool** tip, ki je po velikosti in zgradbi enak tipu **char**, ter tako imenovani ničelni tip **void** (nič).

Imena spremenljivk ne smejo imeti enakih imen kot ključne besede. Seznam vseh ključnih besed, ki jih ne smemo uporabiti za imena spremenljivk:
asm, auto, break, case, catch, char, class, const, continue, default, delete, do, double, else, enum, extern, float, for, friend, goto, if, inline, int, long, new, operator, private, protected, public, register, return, short, signed, sizeof, static, struct, switch, template, this, throw, try, typedef, union, unsigned, virtual, void, volatile, wchar_t, while itd.

Imena spremenljivk se tudi ne smejo začeti s številko (npr. *100abcd*), lahko pa vsebujejo številke na katerikoli drugi poziciji (npr. *abcd123).*

Odločitev (IF stavek)

Tako kot v vsakdanjem življenju, kjer so potrebne nenehne odločitve, uporabljamo tudi v programiranju odločitveni stavek (*if); primer:* če je spremenljivka a večja kot

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

spremenljivka b, izpišemo vrednost c. Pri pogojnem stavku imamo dve vrednosti rešitve, in sicer vrednost pravilna *(true)* in vrednost ni pravilna *(false).* Pogoj je trditev (zapis), ki

lahko vsebuje oklepaje, vrednosti različnih tipov, spremenljivke, konstante različnih tipov in operatorje. Operatorji so posebni simboli, s katerimi lahko kombiniramo spremenljivke oz. vrednosti.

Prva skupina operatorjev predstavlja matematične operacije, kot npr. seštevanje, odštevanje, množenje in deljenje.

Operator	Ime	Primer	Povedano z besedami
+	plus	1 + 2	ena plus dva
-	minus	7 – 3	sedem minus tri
*	krat	5 * 2	pet krat dva
/	deljeno	9 / 3	devet deljeno s tri

Relacijski operatorji primerjajo med seboj dve vrednosti, da bi ugotovili, ali sta enaki ali pa je prva večja ali manjša od druge.

Operator	Pomen	Primer	Povedano z besedami
==	je enako	a == b	a je enako b
!=	ni enako	c != d	c ni enako d
<	manjše kot	x = 10	x je manjši od 10
>	večje kot	y = 0	y je večji od 10
<=	manjše ali enako kot	starost <= 25	starost je manjša ali enaka 25 (vključuje 25)
>=	večje ali enako kot	višina >= 180	višina je večja ali enaka 180 (vključuje 180)

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Pogoji se delijo na:

- enostavne,
- sestavljene (morajo vsebovati vsaj 1 logični operator).
-

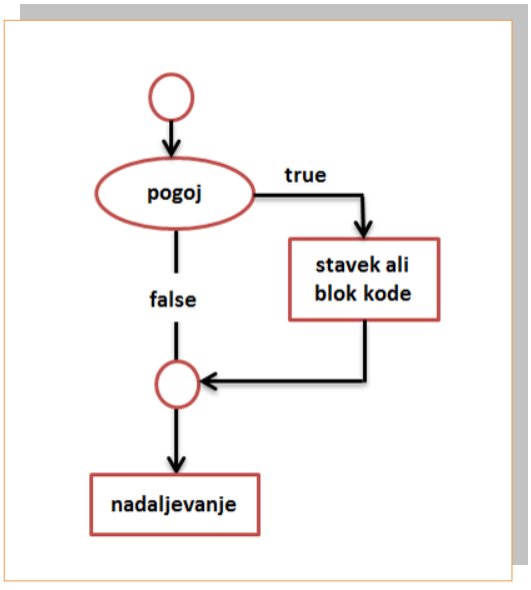
Ukazi, napisani v okviru pogojnega stavka, se izvedejo, če je izpolnjen podan logični pogoj. Pogojni stavek IF uporabimo, ko želimo, da se neka koda izvede le, če je pogoj izpolnjen.

Splošna oblika IF stavka je naslednja:

```
if (pogoj)          // Če je pogoj izpolnjen
{
        true
}
else            // Če pogoj ni izpolnjen
{
        false
}
```

V primeru, da je pogoj odobren in je vrednost true, se bo akcija izvedla.

V primeru, da pogoj ni zadoščen, se bo program nadaljeval brez programske akcije.



Slika 6.1: Primer IF zanke

Vir: Lastni

Primer programa z if: (program prešteje in izpiše števila od 1 do 10):

```
#include<iostream>
using namespace std;
int main()
{
int stevec;
stevec=1;

nazaj:

cout<<stevec<<endl;
stevec=stevec+1;
if(stevec>10)
{
system("Pause");
return 0;
}
else
{
goto nazaj;
}
}
```

Program na levi prešteje in izpiše števila od ena do deset. Najprej v program vključimo vhodno izhodni tok podatkov (iostream), nato se začne izvajati glavni program. Deklariramo spremenljivko števec in jo postavimo na vrednost ena. Vrednost spremenljivke števec se povečuje za ena in to tako dolgo, dokler ni vrednost spremenljivke števec večja od deset.

Izpis

V C++ poznamo ukaz **cout,** s katerim izpišete neko besedilo ali številke na ekran. Podobna ukaza coutu sta *izpisit (*izpis teksta) *ter izpisist (*izpis števil). Ta dva ukaza uporabljamo v »podprogramih«, ki bodo opisani nekoliko kasneje.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Zgradba:

```
cout<<" Neko besedilo ki se izpise na ekran "<<endl

Primer izpisa na ekran:

#include<iostream>
using namespace std;
int main()
{
cout<<"To je izpis na ekran (cout)"<<endl;
system("pause");
return 0;
}
```

Branje

Za branje v C++ uporabljamo ukaza **cin** in **getline.**

Cin bere do prvega praznega znaka in zato ni primeren za večznakovni tip. Ta problem reši ukaz **getline, ki je v knjižnici string.**

Imamo različne možnosti, in sicer:
getline(cin,ime_prostora); → beremo, dokler ne stisnete enter – do konca vrstice;
getline(cin, ime_prostora,znak); → beremo, dokler ne naletite na pravilen znak;
getline(ime_prostora,100); → beremo tolikokrat, kolikor je napisano(v tem primeru 100);
getline(ime_prostora;500,b); → beremo, dokler ne pridemo do b, lahko je največ 500 znakov, več ne, v tem primeru, da je znakov več kot 500, lahko beremo do 500 znakov, ali pa do takrat, ko pridemo do znaka b).

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Primer programa:

```
#include <iostream>
using namespace std;

int main () {
char name[256], title[256];

cout << "Vase ime je: ";
cin.getline (name,256);

cout << "Vas najljubsi film: ";
cin.getline (title,256);

cout << name << " vas najljubsi film je: " << title;

system("pause");
return 0;
}
```

Izbira

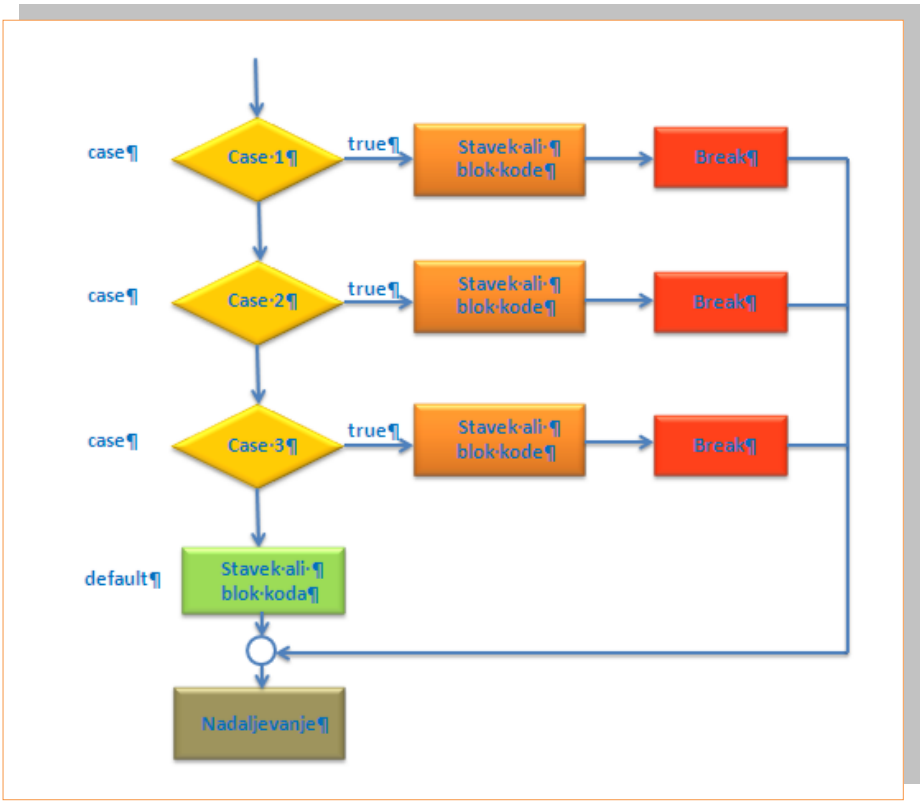
Izbira – izberemo med več variantami.

V C++ obstaja ukaz za izbiro, ki se imenuje **switch/case**.

Zgradba switch case:

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
switch (izraz)
{
case vrednost A; blok ukazov A
break;
case vrednost XY; blok ukazov XY
break;
default;              // ni obvezen
akcija default;
```



Slika 6.2: Prikaz ukaza izbira (switch/case)

Vir: Lastni

Ukaz:

break – preskoči celo strukturo;

continue – en korak, trenutno akcijo prekine;

goto– ta stavek uporabite, kadar ne želite, da se stavki v programski kodi izvajajo po vrsti (ukaza goto se izogibamo, ker lahko vodi k nestrukturirani kodi!!!).

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Primer programa s switch zanko:

```
#include <iostream>
#include <stdlib.h>
using namespace std;

int main()
{
    int a;
    nazaj:
    cout<<"1. Ime"<<endl;
    cout<<"2. Priimek"<<endl;
    cout<<"3. Razred"<<endl;
   cout<<"Izhod iz programa"<<endl;
    cout<<"Vnesi stevilko za prikaz podatkov: "<<endl;
    cin>>a;
    switch(a)
    {
          case 1:
            cout<<"Alen\n";
                                  break;
                                  case 2:
                                  cout<<"Korosec\n";
                                  break;
                                  case 3:
                                  cout<<"2.a\n";
                                  break;
                                  case 9:
                                  cout<<"Izhod iz programa\n";
          system("pause");
          return 0;
          default:
          cout<<"Stevilo ni veljavno! Vnse drugo"<<endl;
   }
    system("pause");
    system("cls");
    goto nazaj;
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Polje (Array)

Polja so skupine enakih objektov ali podatkov. Oštevilčene so od 0 naprej. Predstavljena bo skupina celih števil (int). Pozor na format zapisa: {12, 7, 32, 15, 113, 0, 7}. Poglejmo, kako bi bile te vrednosti zapisane v računalniku, zraven pa podajmo še tekoče številke posameznih podatkov v skupini:

indeks	0	1	2	3	4	5
vrednosti	12	7	32	15	113	7

Tekočo številko podatka v takem polju imenujemo indeks. Če želimo dobiti posamezni element polja, moramo navesti ime polja, v oglatih oklepajih [ ], za imenom pa zapisati številko indeksa. Recimo, da damo takemu polju ime *tockeTekmovalcev*.

Polje je podatkovna struktura, ki se uporabi takrat, ko imamo več spremenljivk istega tipa in za isti namen.

TIP_IME[n] (deklaracija polja)
n – indeks – število elementov
n-ti indeks je v C++ prepovedan!

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Primer programa s poljem: (program izpiše števila od 0–9)

```
#include<iostream>
using namespace std;

int main()
{
int polje[10];
int i;

for(i=0;i<10;i=i+1)
{
polje[i]=i;
}

for(i=0;i<10;i=i+1)
{

cout<<polje[i]<<endl;
}

system("pause");
return 0;
}
```

Program na levi strani prikazuje uporabo polja. Deklarirano je polje tipa integer in spremenljivka i, prav tako tipa integer.

For zanka pravi:

dokler je i manjši od deset, povečujte i za ena oz. napravite inkrement i-ja ter vrednosti zapisujte v polje. Z ukazom *cout* izpišete vrednosti iz polja.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Zanke

Zanka je osnovni ukaz, ki ga uporabimo takrat, ko se del kode (zaporedno) ponavlja za isti namen.

Teoretičen opis zank:

Vsaka zanka ima glavo in telo. V glavi se odločimo ali boste ukaz ponavljali ali ne, v telesu pa so ukazi za ponavljanje. V glavi je pogoj, v pogoju pa vsaj ena spremenljivka, na katero lahko vplivamo.

For Zanka

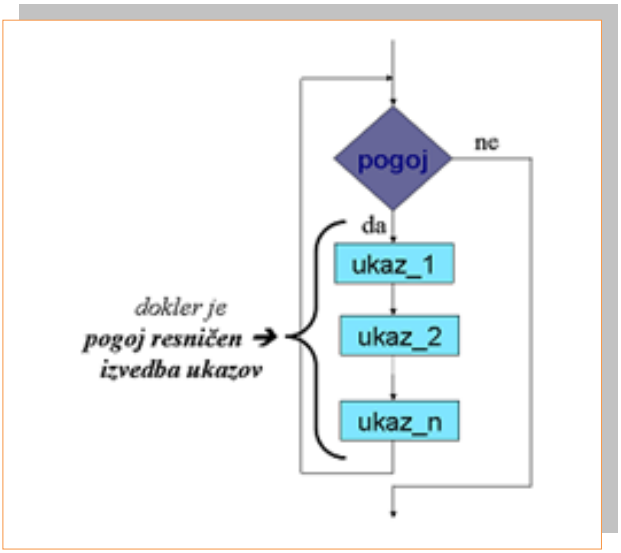
For zanka se ponavlja, dokler ni pogoj true. Primerna je takrat, ko je vnaprej znano število ponavljanj (ne izvede se, če pogoj ni izpolnjen).

Oblika For zanke:

```
for ( ;pogoj; )

{
Blok ukazov
Vpliv na spremenljivke, ki se nahajajo v pogoju
 }
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 6.3: Primer For zanke

Vir: Lastni

Primer programa s for zanko (program 10 x izpiše ime na ekran):

```
#include <stdio.h>
#include <iostream>
using namespace std;

int main()
{
int index;

for(index = 0 ; index < 10 ; index = index + 1)
cout<<"Alen"<<endl;
system ("pause");
return 0;
}
```

Program na levi strani prikazuje delovanje for zanke.

For zanke so v bistvu zgoščen zapis za:

- iniciacijo števca (ali drugega pogoja),
- nastavljanje in preverjanje pogoja in
- povečevanje števca.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

While zanka

Oblika While zanke:

```
while (pogoj)

{

Blok ukazov

}
```

While zanka nadaljuje z izvajanjem, dokler je nek pogoj resničen. Ko postane pogoj neizpolnjen, se zanka prekine. Torej zanka počne ravno tisto, kar je razvidno že iz njenega imena.

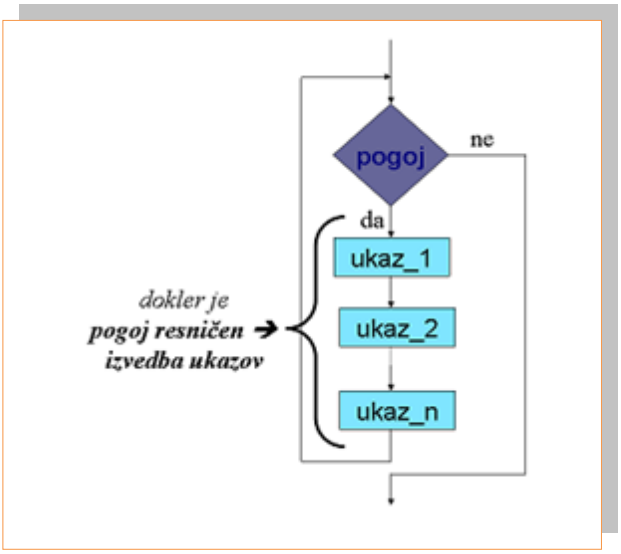
Primer programa z While zanko (program 10 x izpiše ime na ekran):

```
#include <stdio.h>
#include <iostream>
using namespace std;

int main() /
{
int count;

 count = 0;
 while (count < 10)
{
 cout<<"Alen"<<endl;
 count = count + 1;
 }
system ("pause");
return 0;
}
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 6.4: Primer While zanke

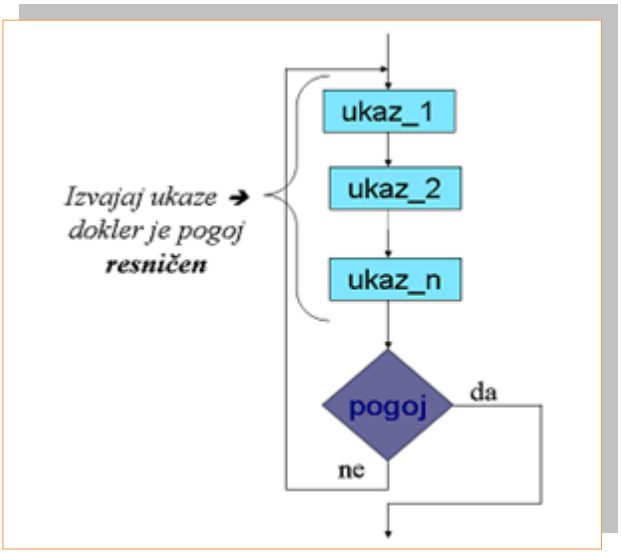
Vir: Lastni

Do While zanka

To zanko je priporočljivo uporabiti takrat, ko podatke že imamo in se na osnovi njih odločamo, ali bomo ponavljali ali ne. Zanka ponavlja, dokler je pogoj true. Do While zanka se za razliko od While zanke izvede vsaj enkrat, pa četudi pogoj ne bo izpolnjen.

Oblika Do While zanke:

```
do
{
}while();
```



Slika 6.5: Primer Do While zanke

Vir: Lastni

Primer programa z Do While zanko (program 10 x izpiše ime na ekran):

```
#include <stdio.h>
#include <iostream>
using namespace std;

int main() /
{
int i;

   i = 0;

   do
   {
      cout<<"Alen"<<endl;
      i = i + 1;
   } while (i < 10);
system("pause");
return 0;
}
```

V danem programu je uporabljena Do While zanka. Zanka se izvaja tako dolgo, dokler je spremenljivka i manjša od vrednosti 10. Vsakič, ko se zanka izvede, se spremenljivka i poveča za 1.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Podprogrami

Ob izvajanju glavnih programov prihaja do klicanja podprogramov. Podprogrami se večinoma uporabljajo takrat, ko se del kode ponavlja na različnih mestih in ko se koda ponavlja zaporedno, a vsakič z različnimi podatki. Značilnost podprograma je, da mora delovati sam zase in ne sme biti odvisen od ostalih programov v kodi. Podprograme delimo v procedure in funkcije (procedura – ime podprograma ne uporabljamo za prenos rezultatov, funkcija – ime podprograma uporabimo za prenos rezultata).

Zgradba podprograma:

* GLAVA (V glavi je tip podprograma ter ime podprograma).

Parametri – so spremenljivke, uporabljene zgolj v podprogramu (v glavi).

* TELO

- se začne z znakom za začetek bloka→ { in konča z znakom za konec bloka → };
- v telesu je koda (v kodi je lahko vse, kar poznamo, razen začeti ne smemo novega podprograma).

Delitev podprogramov:

- Procedure (ime podprograma **ne** uporabljamo za prenos rezultatov);
- Funkcija (ime podprograma uporabimo za prenos rezultatov).

Prenos podatkov v podprogram:

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

a) Globalne spremenljivke.

b) Prenos preko parametrov:

- po vrednosti (spremembe podatkov se ne ohranijo),
- po referenci (spremembe podatkov se ohranijo).

Referenca vpliva na mesto klicanja.

c) Preko datoteke.

d) Nič ne prenašamo in samo beremo.

Prenos podatkov iz podprograma:

a) Globalna spremenljivka:

- vsaka sprememba se ohrani tudi izven podprograma.

b) Raba parametrov:

- vsaka sprememba parametra vpliva na spremenljivko, ki je dala podatke temu parametru.

c) Uporaba datoteke.

Vsak programski jezik ima poseben ukaz, da lahko s pomočjo imena funkcije prenesemo rezultat izven podprograma. V C++ je to ukaz **return.**

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

POVZETEK

Program je zaporedje ukazov, programiranje pa pisanje ukazov za procesor. Ukazi so v vsakem programskem jeziku drugačni, v tem poglavju so opisani ukazi v programskem jeziku C++.Poznamo veliko knjižnic za različne aplikacije.

Knjižice so datoteke, kjer je napisana strojna koda funkcij in jih moramo vključiti v program, če želimo te datoteke v programu uporabiti. Spremenljivke so košček pomnilnika, ločijo se po tipu ter velikosti. Tip spremenljivke nam pove, kaj vsi ti biti, ki jih neka spremenljivka zasede, sploh pomenijo.

Vsaka zanka ima glavo in telo. V glavi se odločamo ali bomo ponavljali ali ne. V telesu pa so ukazi za ponavljanje. V glavi je pogoj, v pogoju pa vsaj ena spremenljivka, na katero lahko vplivamo.

PONOVIMO

1. Naštejte lastnosti glavnega programa.
2. Kaj so knjižnice?
3. Kaj nam pove tip spremenljivke?
4. Kaj se zgodi, če na tipu char uporabimo modifikator unsigned?
5. Razložite razlike ASCII in UNICODE zapisov modifikatorjev.
6. Zapišite primer izpisa na ekran.
7. Napišite program, ki bo zahteval vnos dveh števil in bo izračunal ter izpisal vsoto, produkt, razliko in količnik.
8. Kaj je zanka? Naštejte nekaj zank in opišite Do While zanko.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

9. Narišite diagram poteka in napiši program, ki bo zahteval vnos 10 števil in jih bo izpisal po vrstnem redu od največjega do najmanjšega.
10. Katere tipe spremenljivk poznamo v programskem jeziku C++?
11. Komentirajte spodnji program:

```
#include<stdio.h>
int main ()
{
    int a,b,c;

    a=5;
```

7 MIKROPROCESOR

To poglavje zajema osnove mikroprocesorja in njegove sestavne dele (aritmetično logična enota, krmilna enota, register). V njem je predstavljeno tudi delovanje mikroprocesorja in osnovne funkcije, na kratko pa je prikazano tudi programiranje mikroprocesorja.

Mikroprocesor je osnovni element, okoli katerega je zgrajen mikroračunalnik. Narejen je kot integrirano vezje z visoko stopnjo integracije. Je osrednji del računalnika, ki obdeluje podatke, nadzoruje in upravlja ostale enote. Izveden je v enem samem čipu, to je integriranem elektronskem vezju, ki je izdelano na silicijevi rezini. Oznaka modela mikroprocesorja veliko pove o njegovi procesni zmogljivosti.

V splošnem ga sestavljajo:

- aritmetično logična enota ALE,
- krmilna enota KE,
- registri,
- notranje vodilo.

ARITMETIČNO LOGIČNA ENOTA

Je enota z naslednjimi funkcijami:

- seštevanje, odštevanje, množenje, deljenje;
- pomik vsebin registrov;
- komplementiranje vsebin registrov;
- povečevanje in zmanjševanje vsebin določenih registrov za 1;
- logične operacije (negacija, konjunkcija, disjunkcija, ekvivalenca, antivalenca).

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

ALE je pridružen register stanj, v katerem so posamezni, med seboj neodvisni flip-flopi (zastavice), ki s svojimi stanji 0 ali1 opisujejo stanja v ALE.

Ta so lahko:

- rezultat aritmetične ali logične operacije je enak nič (bit Z – Zero);
- rezultat aritmetične operacije je negativen (bit N – Negative);
- rezultat aritmetične operacije oziroma njegov bitni zapis presega kapaciteto ciljnega registra (bit OV – Overflow), bit za prekoračitev;
- pri aritmetični operaciji je prišlo do prenosa (bit C – Carry);
- procesor pri izvajanju programa dovoljuje ali pa ne prekinitve (bit I – Interrupt).

KRMILNA ENOTA

KE vodi delovanje mikroprocesorja, tako da v skladu s sprejeto instrukcijo posreduje (ukaz programa) krmilne signale ostalim enotam. Izvajanje ene instrukcije ali ukaza lahko razdelimo na dve fazi:

- faza dostavljanja instrukcije (prevzem ukaza – fetch),
- faza izvajanja instrukcije (izvršitev ukaza – execute).

V prvi fazi prinaša KE instrukcijo v instrukcijski register in dekodira operacijsko kodo. V drugi fazi pa v odvisnosti od operacijske kode spreminja stanja v uP in pošilja ustrezne krmilne signale.

REGISTRI

Število registrov in njihove oznake so pri raznih izvedbah uP različne. V grobem ločimo registre, ki so programsko dostopni in programsko nedostopni. Programsko dostopne registre lahko v programu naslavljate in tudi spreminjate njihove vsebine.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

a. Akumulator ACC:

V 8-bitnem uP je 8-bitni register, preko katerega poteka največ operacij v procesorju:

- shranjevanje operandov, nad katerimi izvaja uP aritmetične in logične operacije;
- shranjevanje rezultatov teh operacij;
- posredništvo pri prenosu iz ene v drugo pomnilniško lokacijo ali iz vmesnikov v pomnilnik oz. obratno.

b. Programski števec – PC:

Ta 16-bitni register (*Program Counter*) vodi procesor skozi program. V njem je vpisan naslov naslednje instrukcije, ki jo bo začel izvajati procesor, ko bo zaključil tekočo instrukcijo.

c. Kazalec sklada – SP:

Poseben prostor v podatkovnem pomnilniku, ki se začenja z vnaprej določeno lokacijo, imenujemo sklad. Kazalec sklada (*Stack Pointer*) je 16-bitni register, v katerem je vpisan naslov prve prazne lokacije v skladu pri vpisovanju in zadnje polne pri čitanju iz sklada.

d. Indeksni register – IX:

Je 16-bitni register, ki ga uporabljamo za:

- shranjevanje naslovov pri indeksnem naslavljanju pomnilniških lokacij,
- števec programskih obhodov v zanki določenega programa.

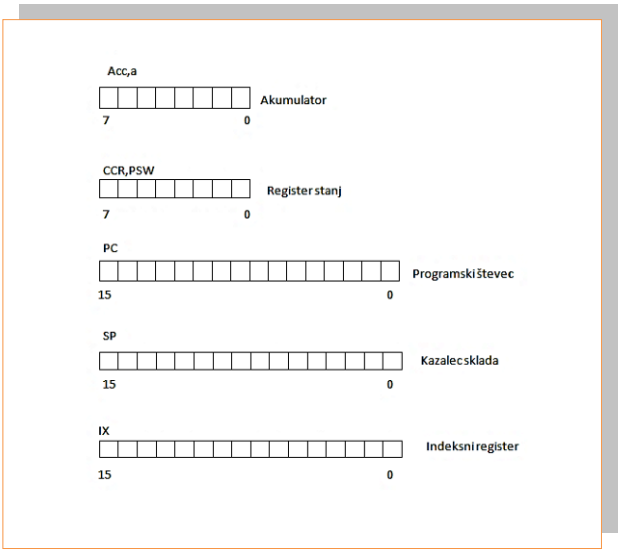
Primer:

Sešteti morate N podatkov. V tem primeru boste v IX vpisali začetni naslov in ga po vsaki izvedeni operaciji povečali za 1, dokler ne boste sešteli vseh N podatkov.

e. Register stanj – R, PSW:

Je 8-bitni register, ki opisuje stanje ALE (*Condition Code Register, Program Status Word*). Glej pri ALE.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



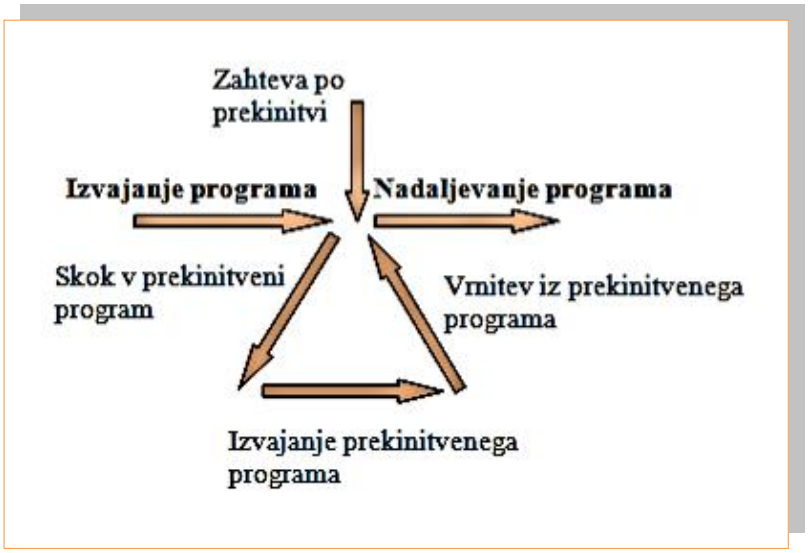
Slika 7.1: Prikaz registrov ALE

Vir: Lastni

PREKINITVE (INTERRUPT)

Ena od zelo pomembnih lastnosti vsakega uP je njegova sposobnost, da na zunanjo ali notranjo zahtevo prekine izvajanje trenutno izvajajočega programa in prične izvajati **prekinitveni servisni program**. Ob zaključku slednjega se vrne na prekinjeno mesto in nadaljuje z izvajanjem prekinjenega programa.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 7.2: Prikaz delovanja prekinitev

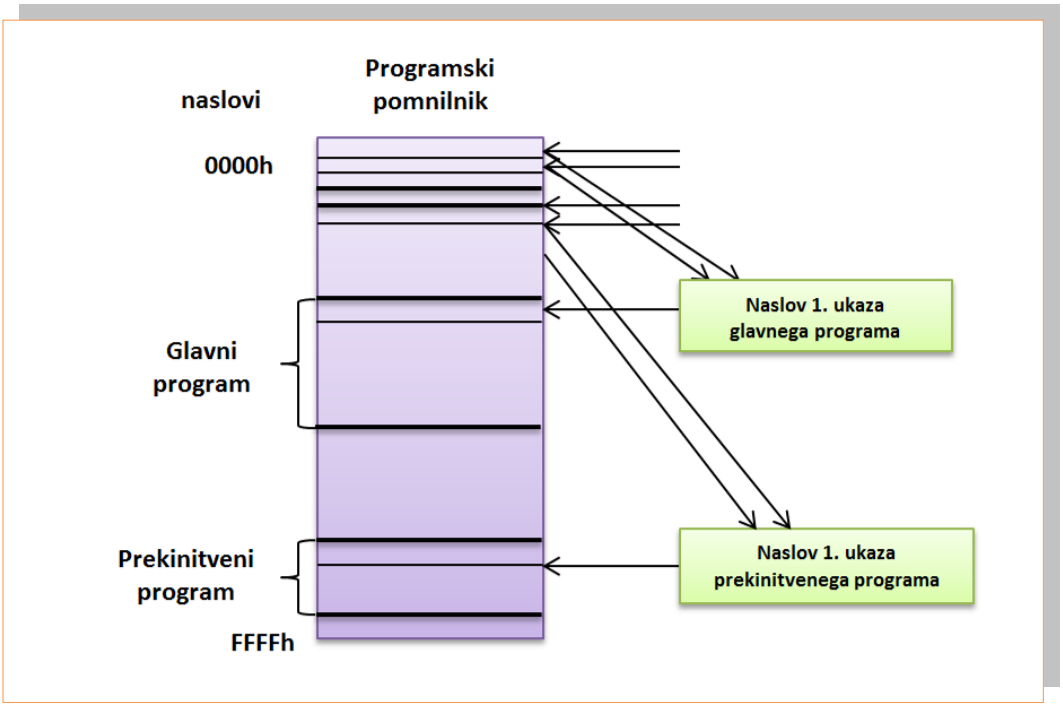
Vir: Lastni

Osnovna zahteva pri prekinitvah je, da se po vrnitvi iz prekinitvenega servisnega programa v CPE vzpostavi stanje, ki je popolnoma enako tistemu ob nastopu prekinitve. Pravimo, da morajo biti prekinitve **transparentne – nevidne**. S tem dosežemo, da se prekinjeni program prekinitev ne zaveda in da so njegovi rezultati enaki, ne glede na to, ali je bil prekinjen ali ne in tudi ne glede na to, v kateri točki je bil prekinjen.

Vektorske prekinitve

Večina uP uporablja to zvrst prekinitev, kjer je vektor pomnilniški naslov v programskem pomnilniku, na katerem je shranjen naslov prekinitvenega servisnega programa. Ker gre za naslov, se imenuje tudi **prekinitveni naslovni vektor**. Če je možnih več prekinitvenih virov, ima vsak vir svoj prekinitveni vektor.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 7.3: Prikaz zasedbe programskega pomnilnika

Vir: Lastni

VIRI PREKINITEV

Zahtevo po prekinitvi lahko uP sporočajo različni viri.

Tako poznamo:

- zunanje vire (signali),
- notranje vire (časovniki, števci),
- serijsko komunikacijo.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

MASKIRANJE PREKINITEV

Zahteva za prekinitev se servisira le, če je to dovoljeno (prekinitev omogočena), v nasprotnem primeru se zahteva ignorira in govorimo o maskiranju prekinitve. Ali neko prekinitev dovolimo ali ne je odvisno od tega, kaj v programu vpišemo v register za dovolitev prekinitev IE (*Interrupt Enable*), ki je sestavljen iz posameznih omogočitvenih bitov za posamezni prekinitveni vir.

Prioriteta prekinitev

V primeru istočasnega nastopa dveh ali več zahtev po prekinitvi imajo nekateri viri prednost pred drugimi. To podaja tabela prioritete proizvajalca uP. Ta določa tudi, ali lahko zahteva iz določenega vira prekine delovanje uP, če je v teku že določeni prekinitveni servisni program. Tudi prioriteto lahko določate programsko v registru IP (*InterruptPriority*).

Servisiranje prekinitve

Če je prekinitev zahtevana in omogočena, se začne servisiranje prekinitve:

- na sklad se shrani vsebina vseh pomembnih registrov, razen SP:
  - vsebina PC, da shranimo naslov ukaza, pri katerem je prišlo do prekinitve, oz. naslov naslednjega ukaza prekinjenega programa;
  - vsebina ACC, da shranimo trenutni rezultat oz. operand pred prekinitvijo;
  - vsebina registra stanj, da shranimo trenutno stanje ALE pred prekinitvijo;
  - vsebina IX, da shranimo v njem zapisan naslov pred prekinitvijo.

- v PC se prenese vsebina viru pripadajočega prekinitvenega naslovnega vektorja, ki je zahteval prekinitev;

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

- izvrševanje ukazov viru in vektorju pripadajočega prekinitvenega servisnega programa. Mikroprocesor ob vsaki prekinitvi sam shrani na sklad vsebino pomembnih registrov in jih tudi iz sklada vrne v registre. S tem je transparentnost zagotovljena, ne da bi moral uporabnik o njej posebej razmišljati;
- z ukazom, ki je zadnji v prekinitvenem programu in pomeni vrnitev iz prekinitve (*Return from Interrupt*) mikroprocesor, vrne podatke iz sklada nazaj v registre in vzpostavi stanje pred prekinitvijo.

SKLAD

Pri večini uP se nekatere operacije vedno nanašajo na sklad. To so predvsem **prekinitve in klici podprogramov**. Sklad je pomemben tudi kot orodje za realizacijo nekaterih programskih rešitev (rekurzija).

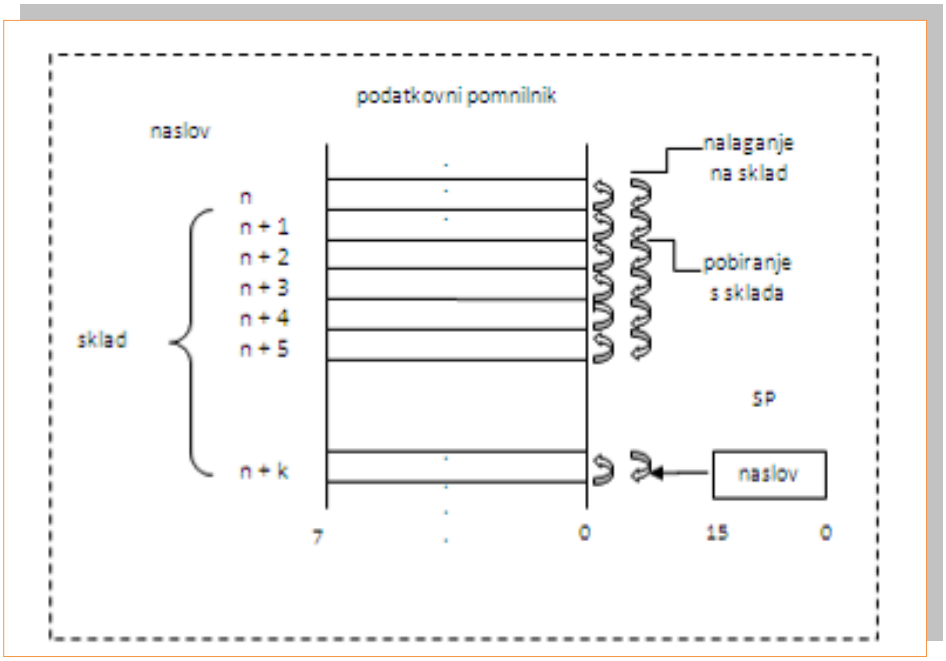
Sklad definiramo na naslednji način:

- rezerviramo del podatkovnega pomnilnika za sklad. Kako velik del pomnilnika bomo rezervirali, je odvisno od obsega uporabe sklada, ta pa od vrste programov, ki jih bo uP izvajal. V vsakem primeru pa moramo zagotoviti, da se prostor, rezerviran za sklad, ne uporabi za nič drugega;
- v kazalec sklada vpišemo vrednost, ki je enaka najvišjemu naslovu pomnilnika, rezerviranega za sklad.

Značilno za sklad je:

- da se širi od višjih naslovov proti nižjim (ali obratno);
- da SP kaže vedno na naslednjo prazno lokacijo v skladu, v katero lahko uP opravi naslednji vpis oziroma na zadnjo polno pri branju iz sklada;
- delovanje sklada je podobno delovanju registra LIFO (last in first out) – podatek, ki smo ga nazadnje shranili na sklad, je prvi, ki ga lahko preberemo.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 7.4: Prikaz podatkovnega pomnilnika

Vir: Lastni

DELOVANJE MIKROPROCESORJA

Reset vektor (glavni vektor) je rezervirana lokacija v programskem pomnilniku, kamor programator vpiše naslov prvega ukaza v glavnem programu, v katerem program steče po resetu (slika 15). Zaradi tega kaže PC po resetu na glavni vektor. Reset uP pomeni, da se njegovi registri postavijo v točno določena začetna stanja. Prekinitveni vektor določenega vira prekinitve pa je rezervirana lokacija v programskem pomnilniku, kamor programator vpiše naslov prvega ukaza prekinitvenega programa. Ker so naslovi običajno 16-bitni, lokacije programskega pomnilnika pa 8-bitne, zavzemajo vektorji dve ali več lokacij.

Mikroprocesor se pri svojem delovanju ravna po signalih ure (clock) ali takta. Njegova opravila so organizirana v tako imenovanih **strojnih ciklih**. En strojni cikel je lahko dolg eno ali več urinih period, kar je odvisno od izvedbe uP. Strojni cikli se imenujejo glede na dogajanja znotraj njih.

Tako poznamo:

- bralni cikel,
- pisalni cikel,
- prekinitveno prevzemni cikel,
- čisti strojni ali izvržbeni cikel.

Bralni cikel

Pri tem ciklu uP pošlje na naslovne signale A0-A15 naslov lokacije, iz katere želi prebrati ukaz ali podatek. Na krmilno vodilo pošlje signal za branje (RD – read). Na podatkovnih signalih D0-D7 pa pričakuje ukaz ali podatek.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

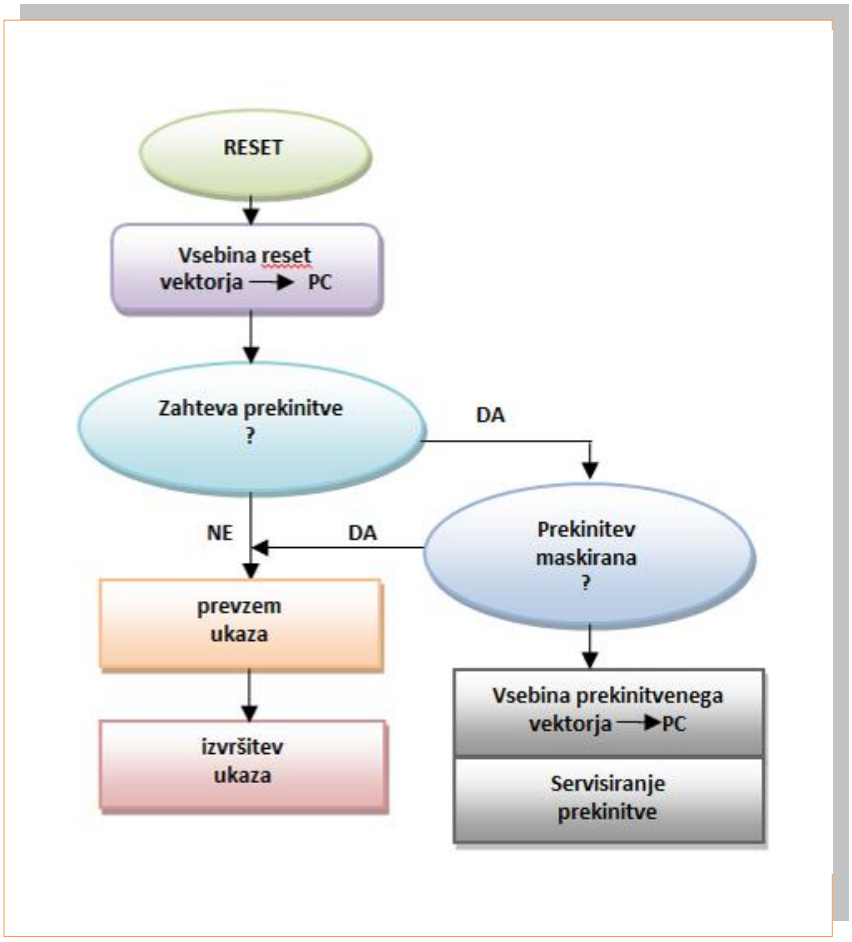
Pisalni cikel

Pri tem ciklu uP pošlje na naslovne signale A0-A15 naslov lokacije, v katero želi vpisati podatek. Na krmilno vodilo pošlje signal za pisanje (WR – write). Ta podatek pošlje na podatkovne signale D0-D7.

Program je sestavljen iz ukazov ali instrukcij, ki povedo uP, kaj mora narediti. Izvajanje vsake instrukcije poteka preko več strojnih ciklov, ki sestavljajo **instrukcijski cikel**. **Prevzem ukaza** ali fetch je eden ali več bralnih ciklov, v katerih uP bere iz lokacij programskega pomnilnika. **Izvršitev ukaza** ali execute pa je korak realizacije ukaza.

Primer: instrukcija c=a+b
- fetch
1. bralni cikel operacijske kode
- execute
1. bralni cikel operanda a
2. bralni cikel operanda b
3. strojni cikel seštevanja
4. pisalni cikel shranjevanja rezultata

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 7.5: Prikaz delovanja mikroprocesorja

Vir: Lastni

POMNILNIK

Idealen pomnilnik bi moral imeti vse naslednje lastnosti:

- iz njega bi lahko izvajali neomejeno število branj (to lastnost imajo pomnilniki v obliki IC),
- vanj bi lahko opravljali neomejeno število vpisovanj (to lastnost ima samo RAM),
- ob izpadu napajalne napetosti bi moral vsebino zadržati (to lastnost imajo ROM in Flash RAM),
- vpisovanje bi moralo potekati enako hitro kot branje (samo RAM),
- kapaciteta pomnilnika in hitrost dostopa bi morala biti dovolj velika pri sorazmerno nizki ceni.

V grobem delimo pomnilnike v obliki IC na bralne in bralno-zapisovalne. Iz vseh je možno neomejeno branje, število vpisov pa se razlikuje.

V skupino, ki imajo končno število vpisov in zadržujejo vsebino tudi ob izpadu napajalne napetosti, sodijo:

- ROM (*read only memory*)
  Vpis vanj izvede proizvajalec. Uporaben je za program, konstante in tabele.
- PROM (*programable ROM*)
  Omogoča enkraten vpis vsebine, kar lahko izvede programer s pomočjo programatorja. Uporabnost je takšna kot pri ROM-u.

- EPROM (*erasable PROM*)
  S pomočjo UV svetlobe je možno njegovo vsebino izbrisati, tako da omogoča nekaj deset vpisov s pomočjo programatorja. Uporabnost je podobna kot pri PROM-u.
- EEPROM (*electric EPROM*)

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Vsebino je možno izbrisati kar v ciljnem sistemu, vendar le nekaj 100 000-krat. Zaradi tega nima enake uporabne zmožnosti kot RAM. Uporabljate ga lahko za podatke, ki se redkeje spreminjajo oz. jih spreminja uporabnik s pomočjo tipkovnice.

- FLASH RAM
  Je nižje cenovna verzija EEPROM-a in omogoča do 1000 vpisov.
  Vpisovanje v zgoraj naštete pomnilnike poteka po določeni proceduri, vsekakor pa počasneje kot pri pomnilniku, v katerega lahko neomejeno vpisujemo.
- RAM (*Random Access Memory*)
  Svojo vsebino ob izpadu napajalne napetosti izgubi. Uporaben je za shranjevanje podatkov. Če želite te podatke trajno ohraniti, morate uporabljati sistem neprekinjenega napajanja ali pa ob izpadu napajanja podatke prepišete v EEPROM. Če želite RAM uporabljati za programski pomnilnik, morate vsakič ob vklopu naložiti program.

VHODNO-IZHODNI VMESNIK

Poznamo več vrst vhodno-izhodnih vmesnikov. Nekateri od njih imajo programsko dostopne registre, s katerimi lahko uP komunicira. Razen podatkov, ki se prenašajo preko vmesnikov, pošilja uP ukaze, s katerimi določa, kako naj takšen vmesnik deluje (smer pretoka podatkov, usklajenost delovanja, možnost prekinitev ...).

Takšne vmesnike imenujemo **PIA (*Paralel Interface Adapter*).** Pomembno je, da ima vsak register vmesnika svoj naslov ali pa, da register določimo s pomočjo naslovnega dekoderja. Tako lahko v programu vmesnik obravnavamo kot običajno pomnilniško lokacijo, preko katere komuniciramo z zunanjim svetom.

Za vmesnike pa največkrat uporabljamo registre, ki jim pravimo zadrževalniki (latch). Tem moramo pošiljati kontrolne signale, ki te vmesnike odpirajo in zapirajo. Dobimo jih s pomočjo naslova preko naslovnega dekoderja.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Vmesniki imajo naslednje vhodne kontrolne signale:

- CE (*chip enable*)

  Vmesnik je omogočen oz. pripravljen sprejemati podatke (vpisovanje), kadar je na tem vhodu logičns 1. Različica tega signala je CS (*chip select*).
- OE (*output enable*)

  Omogočeno je branje iz vmesnika oz. na izhodih vmesnika se pojavi v registru zapisana vsebina, kadar je na tem vhodu 1log. Takšne vmesnike ne moremo uporabljati kot dvosmerne, kajti smer je določena z ožičenjem. Dvosmerni vmesniki imajo še dodaten kontrolni signal, s katerim lahko določamo smer. Naslovni dekoder kreiramo s pomočjo logične enačbe.

PROGRAMIRANJE MIKROKRMILNIKA

Vsak mikroprocesor pozna določene **ukaze**. Množico ukazov za neki mikroprocesor imenujemo **nabor ukazov.** Uporabnik uporablja nabor ukazov, tako da z njimi sestavi neko smiselno zaporedje. Temu zaporedju pravimo **program**, njegovemu sestavljanju pa **programiranje.**

Programiranje lahko opravljamo na **strojnem ali zbirniškem nivoju**, kjer neposredno uporabljamo nabor ukazov (zbirnik ali *assembler*). Druga možnost je programiranje v **višjem programskem jeziku**, ki je človeku bližji (Basic, Fortran, Pascal, C …).

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Programer mora do podrobnosti razumeti problem, za katerega piše program. Njegovo delo lahko razčlenimo v naslednja opravila:

- **Analiza problema** je razvoj postopka, ki korak za korakom reši dani problem. Temu postopku pravimo **algoritem**.

- **Organizacija programa**. Algoritem organizirate v zaporedje operacij, pri čemer uporabljamo **diagrame poteka**. Delo lahko opravljamo s pomočjo osebnega računalnika, če imamo orodje za risanje diagramov poteka.

- **Kodiranje**. Diagram poteka zapišemo v zaporedje ukazov – program s pomočjo **urejevalnika teksta** za osebni računalnik. Shranimo ga v t. i. izvorni datoteki.

- **Povezovanje in prevajanje**. Program povežemo z drugimi deli programa, ki smo jih napisali že v preteklosti, ali pa s programi, ki jih dobimo v knjižnici (napisal jih je proizvajalec prevajalnika in povezovalnika). Povezavo opravi **povezovalnik – linker**. Celoten program prevedete s **prevajalnikom – compilerjem** v obliko, ki jo razume uP. Povezovalnik in prevajalnik sta programski opremi za osebni računalnik.

- **Vstavljanje programa v mikroračunalnik** opravimo s pomočjo **programatorja**, ki ga priključimo na osebni računalnik s pripadajočo programsko opremo.

- **Preizkušanje**. Program preizkusimo tako, da z izvajanjem na mikroprocesorju ugotovimo ali da ustrezne rezultate. V ta namen izdelamo testno kartico. Napake, ki jih pri tem odkrijemo, odpravimo tako, da se vrnemo na kodiranje in ponovimo postopek.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

- **Dokumentiranje**. Za kasnejšo dodelavo ali spremembo programa je potrebno program dokumentirati. Dokumentacijo s potrebnimi komentarji uredimo pri vseh opravilnih točkah (tehnični opis, poročilo, elaborat).

Primer:

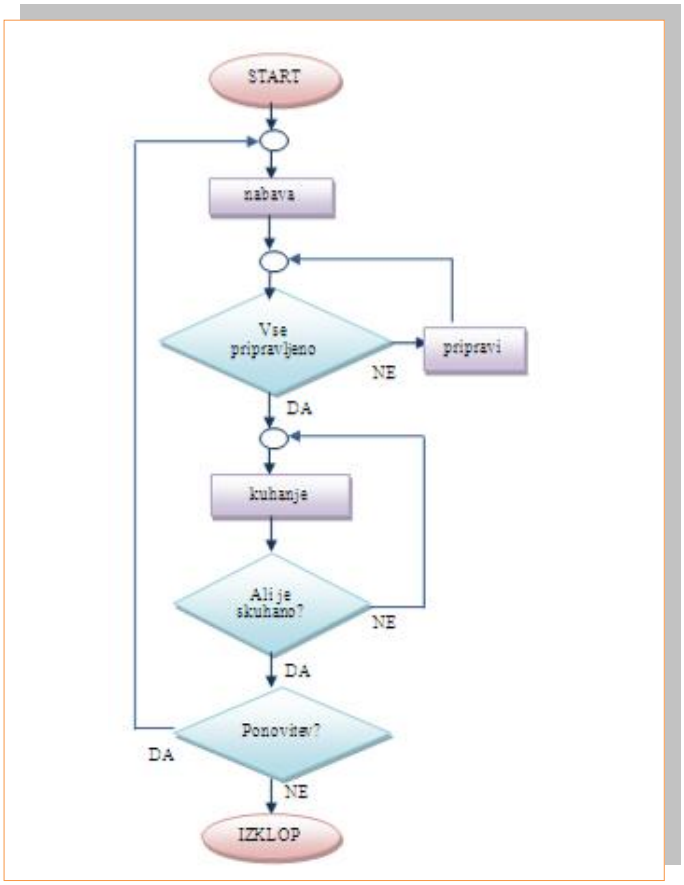
Realizirajte prvi dve točki za problem kuhanja kave.

1. Analiza problema:

a) nabava potrebščin (kava, sladkor in voda), opreme (štedilnik, žlica, kuhalna posoda) in energije (elektrika ali plin) in njihova priprava,
b) kuhanje,
c) testiranje (ali je že kuhano),
d) testiranje ponovitve,
e) izklop.

2. Organizacija programa:
   diagram poteka

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 7.6: Primer programa

Vir: Lastni

PROGRAMIRANJE V ZBIRNEM JEZIKU

Zbirni jezik (zbirnik) ali **assembler** je jezik določenega procesorja oziroma družine mikrokontrolerjev, ki bazirajo na istem mikroprocesorju. Uporablja nabor oziroma zbirek ukazov, ki jih uP pozna.

V vsakem ukazu mora biti prisotna informacija o dveh stvareh:

- operaciji,
- operandih.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Informacija o operaciji je vsebovana v t. i. **operacijski kodi**. Informacija o operandih pa je lahko podana na več načinov. V mikroračunalniku so operandi lahko shranjeni v registrih procesorja, v pomnilniku ali v registrih vhodnih naprav. V vsakem ukazu je zato potrebno na nek način povedati, kje se operandi nahajajo. Ker to običajno povemo z naslovom, na katerem se operand nahaja, pravimo temu **naslavljanje.** Kje se operandi nahajajo, lahko povemo na več načinov in tem pravimo **načini naslavljanja**.

Ukazi lahko v programskem pomnilniku zasedajo eno ali več lokacij. Tako poznamo eno-,dve- ali večzložne ukaze (zlog=byte=8 bitov).

MODEL MIKROPROCESORJA

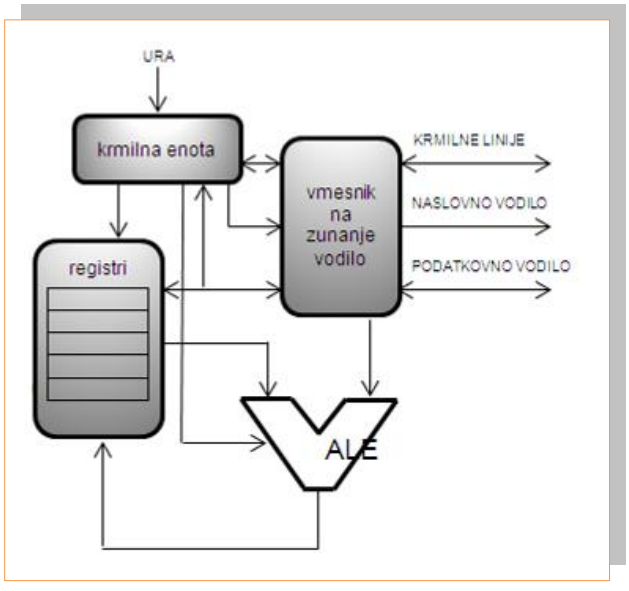
Arhitekture mikroprocesorjev se zelo razlikujejo med seboj, večina pa jih temelji na von Neumannovem modelu, kjer se podatki in programska koda nahajajo skupaj v istem pomnilniškem naslovnem področju. To je bila nesporno ustrezna in zelo koristna rešitev v časih, ko je obdelava ukazov zahtevala bistveno več časa kot njihov prenos. V zadnjem času se je ozko grlo preselilo na slednje. Von Neumannova arhitektura je postala šibka točka, ki jo predvsem v RISC mikroprocesorjih nadomešča harwardska: ta ima ločeni pomnilniški področji in podatkovne poti za podatke in ukaze. Prenos enih in drugih zato lahko poteka popolnoma vzporedno.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Slika 7.7: Model mikroprocesorjev

Vir: Lastni

Arhitekturo mikroprocesorja lahko v grobem razdelimo na tri glavne sestavne dele: **krmilno enoto, aritmetično-logično enoto in nabor registrov.**



Slika 7.8: Zgradba mikroprocesorja

Vir: Lastni

Najpomembnejši del mikroprocesorja predstavlja krmilna enota, ki usklajuje delovanje posameznih delov mikroprocesorja po navodilih programske kode. Krmilna enota bere

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

ukaz za ukazom iz programa, zapisanega v strojni obliki, jih dekodira in preko množice krmilnih signalov krmili njihovo izvajanje. Za izvajanje operacij nad podatki, ki jih obdeluje program, skrbi aritmetično-logična enota (ALE). Ta je sposobna opraviti različne matematične in logične operacije nad podatki različnih tipov.

Registri služijo za začasno hranjenje in obdelavo podatkov in njihovih naslovov. Izvedeni so kot zelo hitre pomnilne celice (običajno 5- do 10-krat hitrejše od zunanjega – delovnega). Obstaja več vrst registrov, od katerih so nekateri dostopni programerju, drugi pa so namenjeni internemu delovanju mikroprocesorja. Nekateri registri nastopajo tudi kot del krmilne oziroma aritmetične in logične enote. Posamezne enote mikroprocesorja so med seboj povezane preko več internih vodil. Širina teh vodil običajno po širini ustreza velikosti registrov. Prav tako je mikroprocesor z okolico povezan preko zunanjih vodil, ki jih pri večini mikroprocesorjev sestavljajo:

- **naslovno vodilo** – ta določa naslov pomnilniške lokacije, do katere želi mikroprocesor dostopati;
- **podatkovno vodilo** – preko njega poteka izmenjava vsebine (programskih ukazov in podatkov) med registri in celicami pomnilnika, torej v mikroprocesor ali iz njega;
- **krmilne linije** –krmilijo komunikacijo z okolico (povejo npr. ali želi mikroprocesor brati ali pisati v pomnilniško področje).

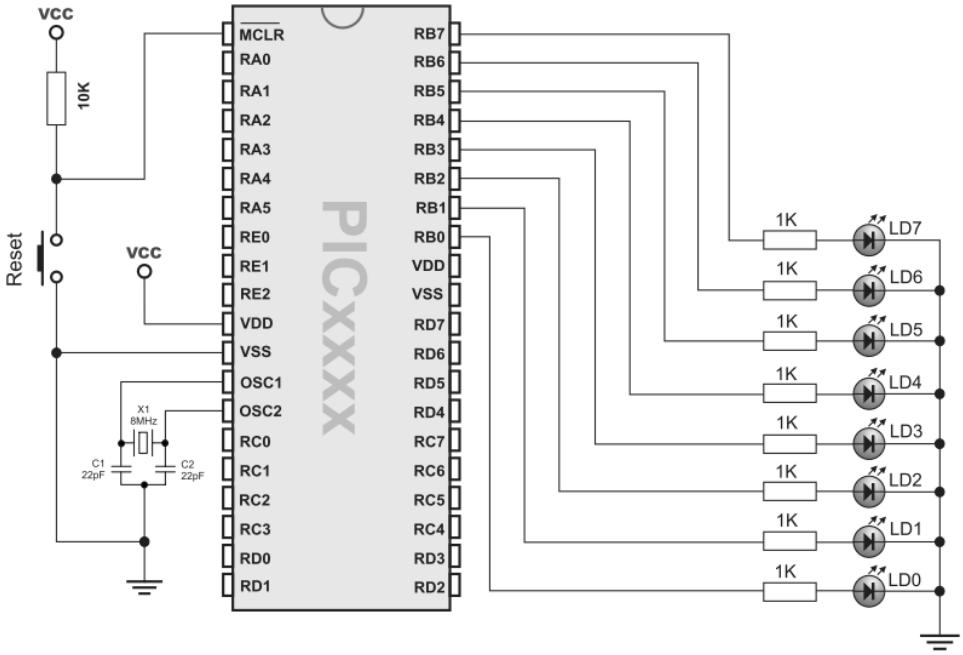
Zaradi tehnoloških omejitev so bile širine zunanjih vodil pri starejših mikroprocesorjih pogosto manjše od širine internih vodil. To je pomenilo, da je bilo potrebno za prenos ene besede izvesti dva ali več bralnih oz. pisalnih ciklov. V sodobnih mikroprocesorskih arhitekturah težimo k čim večji prepustnosti podatkov in nimajo takšnih omejitev.

Zunanje in notranje vodilo sta povezana preko posebnega vmesnika, katerega delovanje je pod nadzorom krmilne enote. Nekateri preprostejši mikroprocesorji in mikroračunalniki imajo programski in podatkovni pomnilnik vgrajen na isti silicijevi rezini. Takšni mikroprocesorji seveda zunanjega naslovnega in podatkovnega vodila ne potrebujejo.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

UČNA SITUACIJA: UPORABA MIKROKRMILNIKA V PRAKSI

MICROCHIP PIC MIKROKRMILNIK Priključitev



Slika 7.9 : Prilkjučna shema PIC 16F877

Vir: http://www.mikroe.com/pdf/mikroc/1st_project_pic_c.pdf

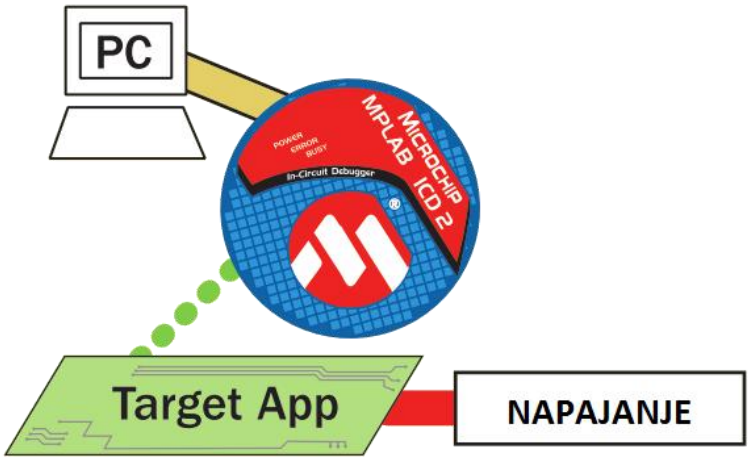
Za delovanje mikrokrmilnika moramo zadostiti sledečim zahtevam:

- ✓ **Napajanje PIC mikrokmilnika 5V DC (Vcc).**
  Stabilizacijsko vezje LM 7805 priključimo na višjo napetost in povežemo na napajalne priključke.
- ✓ **Priključitev quartz kristala** (takt ure, 4MHz, 8MHz) na priključke OSC1 in OSC2. Na maso priključimo keramična kondenzatorja 22pF (odprava motenj).
- ✓ **Priključitev potenciala 5V na priključek /MCLR**(memory clear), **preko upora 10 kΩ** »pull down« tipka na maso za RESET mikrokrmilnika.
- ✓ **Uporaba »pull down« ali »pull up« uporov** pri vhodnih priključkih mikrokrmilnika.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Za programiranje se lahko uporabi programsko orodje MPLAB in MikroC ali kateri drugi programski jezik (Basic, C++…)

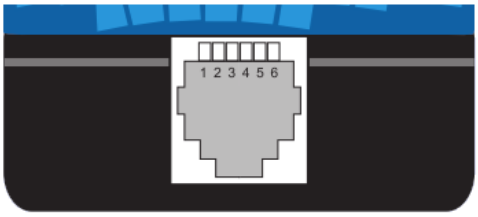
Priključitev programatorja ICD2



Slika 7.10 : Priključitev ICD2 programatorja

Vir: Lastni

ICD2 programtor priključimo na osebni računalnik preko USB povezave. Paziti moramo, da imamo na osebnem računalniku namščene gonilnike za programator (običajno na pripadajočem cd-ju). Ciljni sistem (projekt) priključimo preko RJ45 konektorja na ICD2 programator. Paziti moramo na pravilno razvrstitev priključkov.

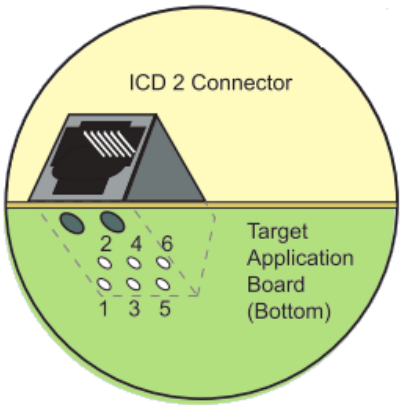


Pin	Signal
Not Used	1
PGC	2
PGD	3
Ground	4
VDD	5
$\overline{MCLR}$/VPP	6

Slika 7.11: Razporeditev priključkov na ICD2

Vir:www.microchip.com

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Modular Connector Pin	Microcontroller Pin
6	Not Used
5	PGC
4	PGD
3	Ground
2	V_{DD}
1	$\overline{MCLR}/V_{PP}$

Slika 7.12: Razporeditev priključkov na ciljnem sistemu (projektu)

Vir:www.microchip.com

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

PROGRAMIRANJE

Primer programa v mikroC za prižig lučke ob pritisku na tipko:

```
dim oldstate as byte                    //definiramo spremenljivko kot byte

main:                                   //glavni program

 oldstate = 0                           //postavimo na vrednost 0
 TRISA = 0xFF                           //PORT A določimo kot vhod
 TRISB = 0                              //PORT B določimo kot izhod
 PORTB = 0x0F                           //PORT B postavimo na 0

 while true                             //while zanka
  if (Button(PORTA, 1, 1, 1)) then      //pogoj, če je pritisnjena tipka na PORT A
  oldstate = 1                          //postavimo na 1
  end if                                //konec pogojnega stavka

 if (oldstate and Button(PORTA, 0, 0, 0)) then   //če prejšnje stanje in tipka
  PORTB = not PORTB                              //negacija
  oldstate = 0                                   //spremenljivko postavimo na 0
  end if                                         //konec pogojnega stavka
wend                                             //konec while zanke

end.                                             //konec glavnega programa
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Naloga:

Izdelajte in programirajte elektronsko vezje za mobilnega robota, ki se vozi po črni črti. Kot senzor lahko uporabite svetlobni senzor CNY70.

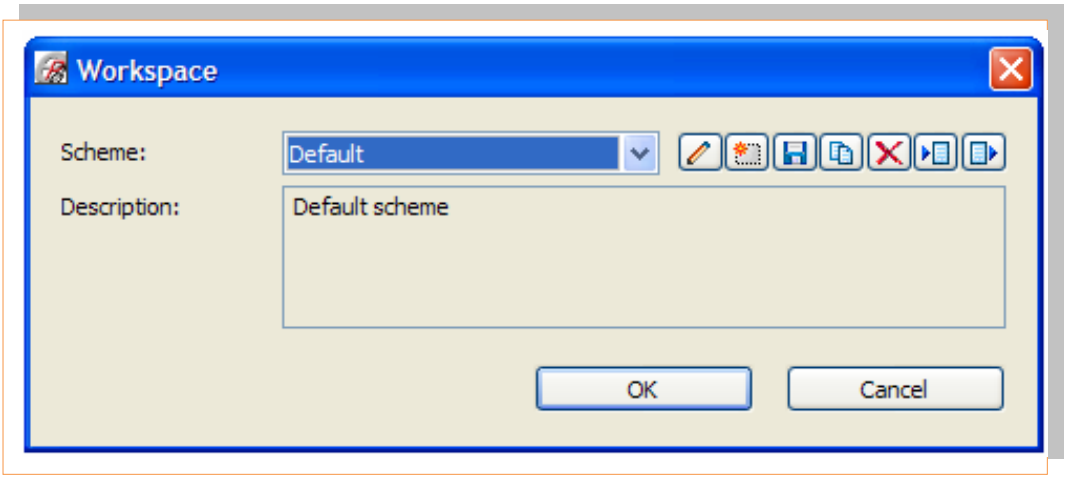
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

8 PROGRAMSKO ORODJE ZA PROJEKTIRANJE ELEKTRIČNIH INŠTALACIJ EPLAN P8

Program zaženemo z dvojnim klikom na ikono ePlan, ki se nahaja na namizju.

Grafični vmesnik programa je sestavljen iz oken Page navigator, Graphical preview in grafičnega okna, v katerem rišemo sam projekt. Izgled samega vmesnika si lahko vsak posameznik prilagaja svojim željam in potrebam, tako je mogoče dodajati/skrivati opravilne vrstice, dodajati gumbe na opravilne vrstice, izdelava lastnih opravilnih vrstic, določanje bližnjic do ukazov preko tipkovnice [Options -> Keyboard shortcuts], položaj posameznih oken, itd …

Izgled samega vmesnika je mogoče shraniti preko t.i. workspace-a [view -> workspace], kjer lahko izbiramo med že izdelanimi predlogami oz. jih lahko sami prilagodimo po svojih potrebah ali naredimo nove.



Slika 8.1: Nastavitev delovnega prostora

Vir: Lastni

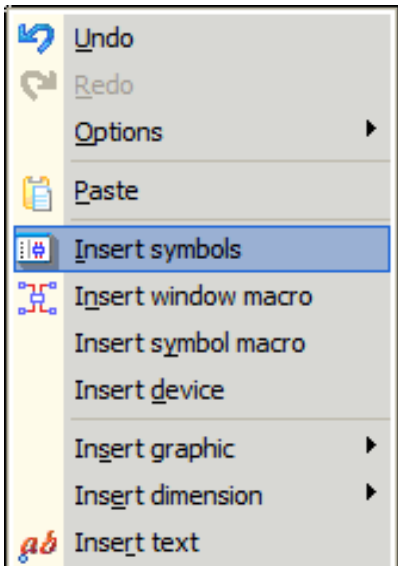
Sočasno imamo lahko odprtih več projektov in več strani, med katerimi lahko kopiramo podatke/simbole …

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Dostopanje do ukazov oziroma simbolov

EPLAN P8 omogoča dostop do različnih ukazov na več različnih načinov, tako lahko dostopamo do posameznih ukazov preko spustnih menijev, lahko si priredimo bližnjico na tipkovnici, pogosto so posamezni ukazi prisotni tudi v meniju, do katerega dostopamo preko desnega gumba miške, program pa tudi omogoča da si izdelamo gumb v opravilni vrstici, ki nam bo omogočal hitri dostop do ukaza. Na primeru prikažimo dostopa do sponke, ki jo pogosto uporabljamo tekom izdelave projekta.

Dostop do sponke s pomočjo menija, ki ga dobimo s klikom na desno miškino tipko:

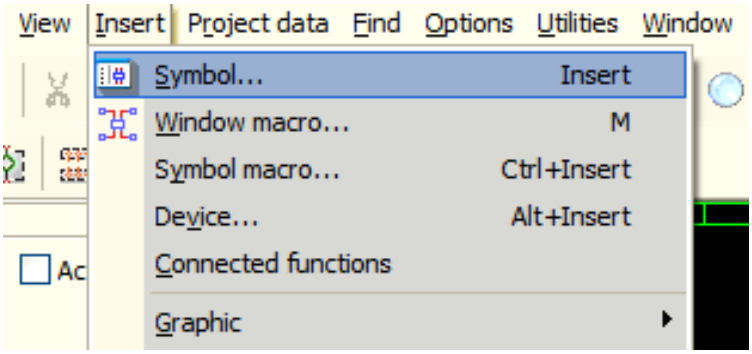


Slika 8.2: Meni

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Dostop preko glavnega menija v programu:

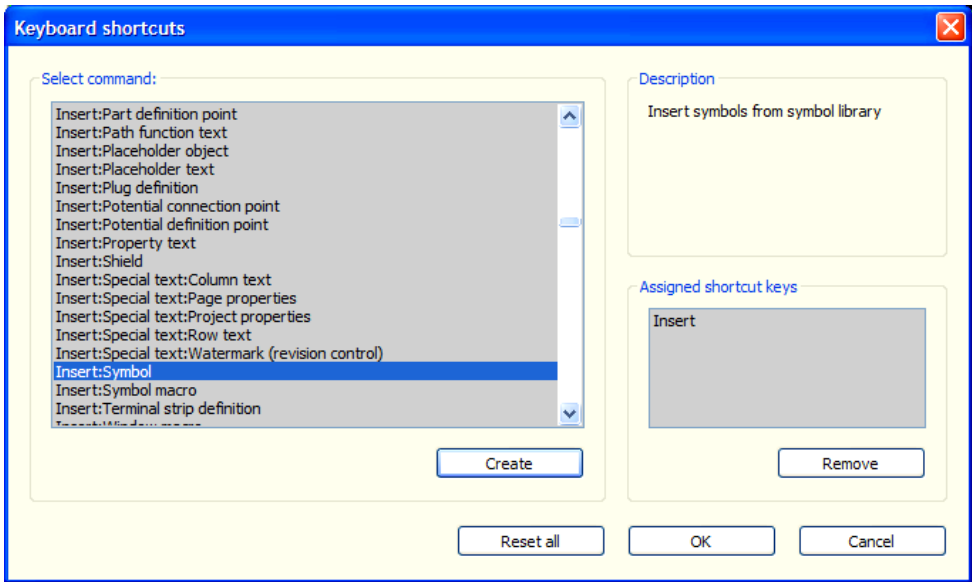


Slika 8.3: Vstavljanje

Vir: Lastni

Dostop preko bližnjice, ki jo lahko sami določimo preko:

(Options > Keyboard shortcuts >(v oknu Select command poiščemo) Insert:Symbol. S klikom na gumb Create se nam odpre okno. Sedaj pritisnemo kombinacijo tipk, katero želimo prirediti za hitri dostop do knjižnice simbolov).

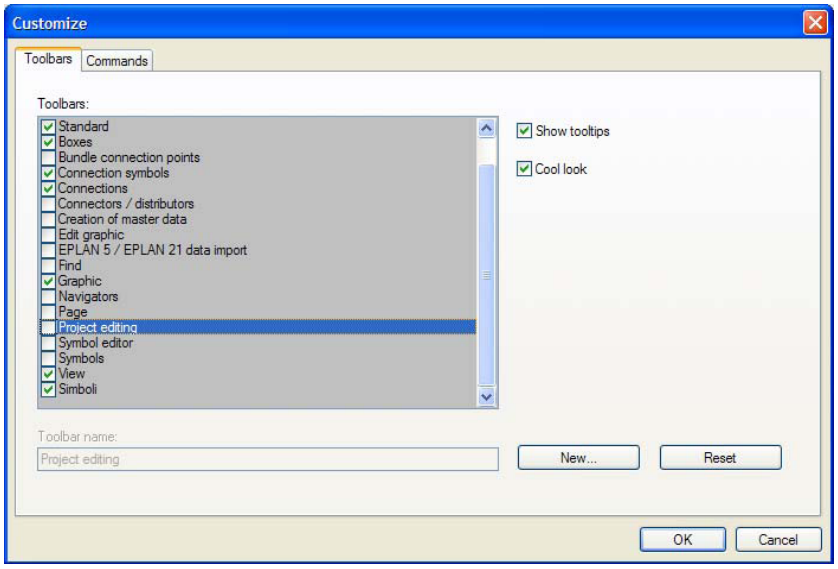


Slika 8.4: Hitri dostop

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Naslednji način je da si izdelamo gumb, s katerim bomo neposredno dostopali do sponke. Preko menija (options > Toolbars > Customize) dostopamo do okna v katerem vidimo vse opravilne vrstice, ki so nam na voljo.



Slika 8.5: Izdelava ikone

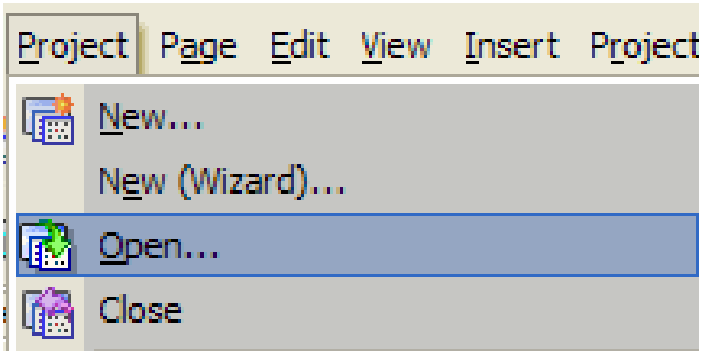
Vir: Lastni

Na zavihku Toolbars, kliknemo na gumb New in odpre se novo okno. Vanj vpišemo ime nove opravilne vrstice. Po vpisu se izdela nova vrstica, ki jo vidimo na vmesniku programa. Gremo na zavihek Commands, kjer v oknu Commands izberemo opcijo Actions nato pa v oknu Buttons možnost Insert symbol; le-tega potegnemo v svojo novo opravilno vrstico, ki jo imamo na vmesniku programa. Ko smo potegnili opcijo Insert symbol v svojo opravilno vrstico, se nam odpre okno, v katerem določimo na kateri simbol nam bo kazala bližnjica.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Odpiranje projekta

[Project > Open]

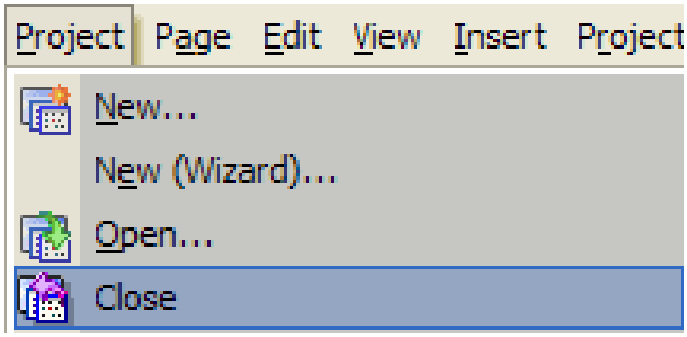


Slika 8.6: Odpiranje projekta

Vir: Lastni

Zapiranje Projekta

[Project > Close]



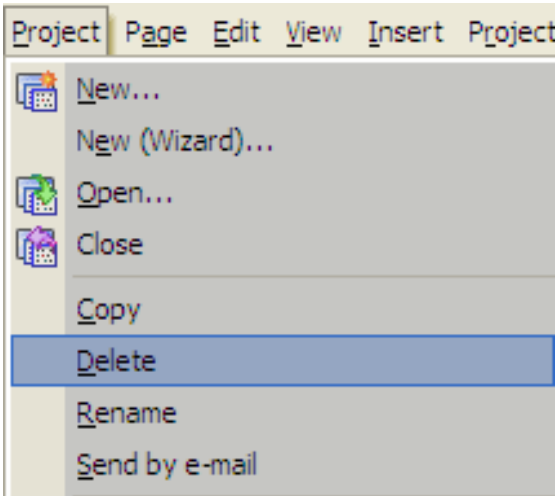
Slika 8.7: Zapiranje projekta

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Kopiranje, brisanje projekta

[Project > copy oz. delete]



Slika 8.8: Brisanje projekta

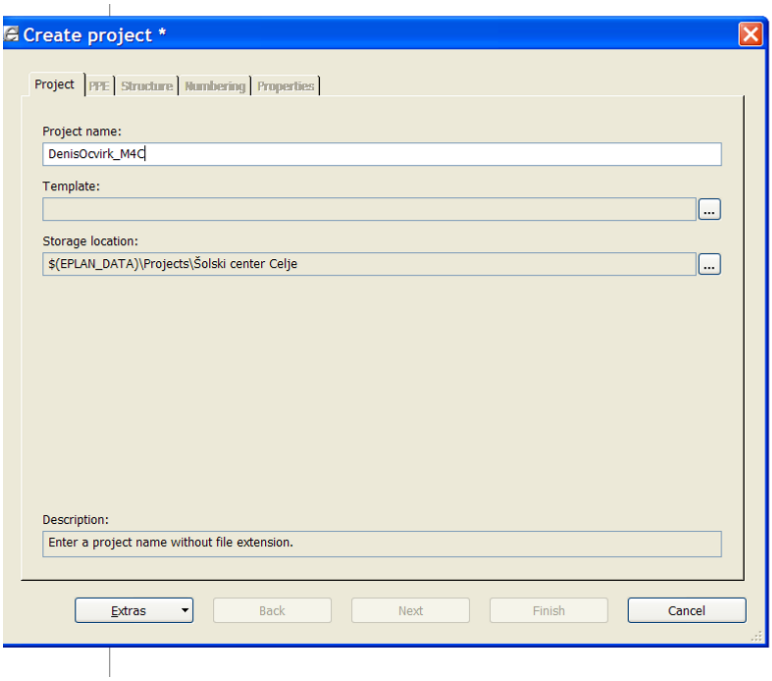
Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Kreiranje novega projekta

Izdelavo novega projekta izvedemo preko menijskega ukaza:
[Project -> New(wizard)]

Odpre se nam okno:



Slika 8.9: Ustvarjanje novega projekta

Vir: Lastni

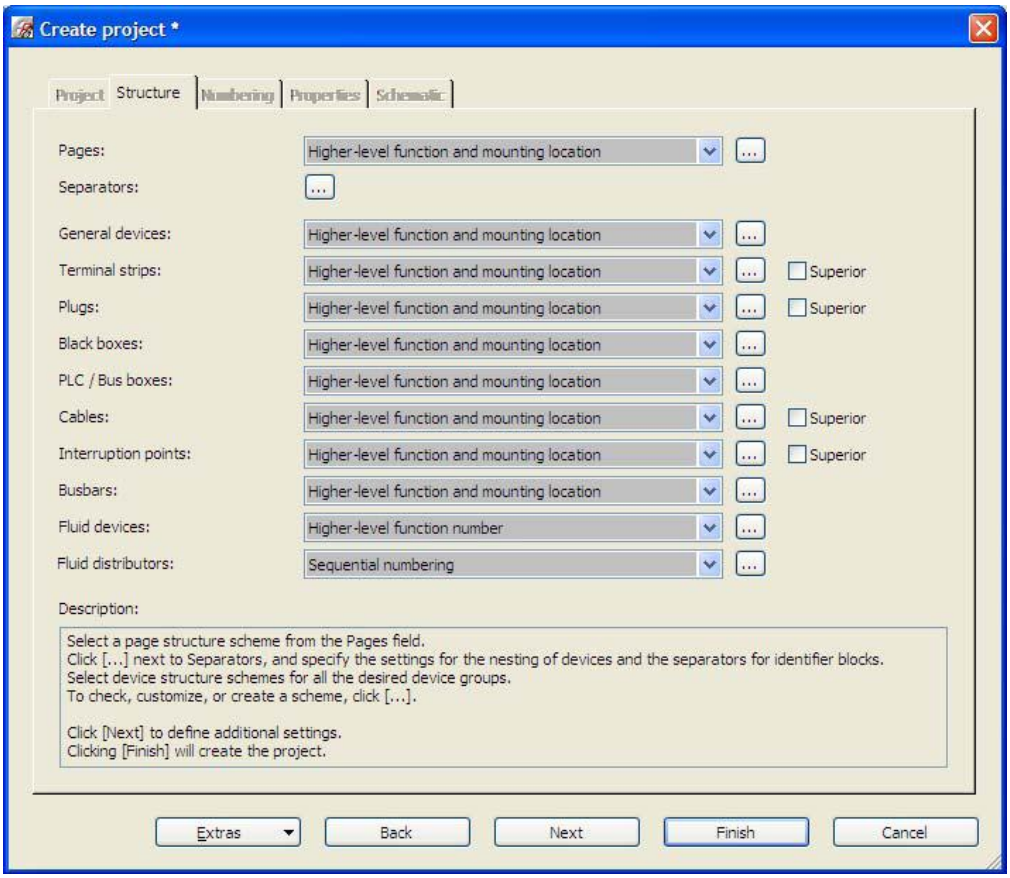
Project name: ime projekta

Template: kliknemo izbirni gumb ter izberemo eno izmed predlog (npr.: Num_tpl001.ept, IEC_tpl001.ept); razlika med obema predlogama je v organizaciji projekta: Num_tpl001.ept ima definirano enostavno številčenje strani, brez določene drevesne strukture IEC_tpl001.ept ima definirano organizacijo strani po IEC standardu, kateri določa označevanje po višji funkciji [HLA], ter po lokacijski oznaki.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Storage location: preko izbirnega gumba določimo lokacijo, kjer se nam shranijo vse datoteke, povezane z samim projektom – lokacija projekta ni omejena na mape programa, projekte lahko shranjujemo na poljubno mesto na disku oz. na mrežne pogone.

Ko smo vse skupaj določili, kliknemo Next.



Slika 8.10: Nastavitve

Vir: Lastni

Na zavihku Structure je mogoče vsakemu sklopu komponent (sponke, PLC, kabli, naprave, …) določiti svoj način označevanja, lahko pa uporabimo tudi prednastavljen način, kateri je shranjen v sami predlogi.

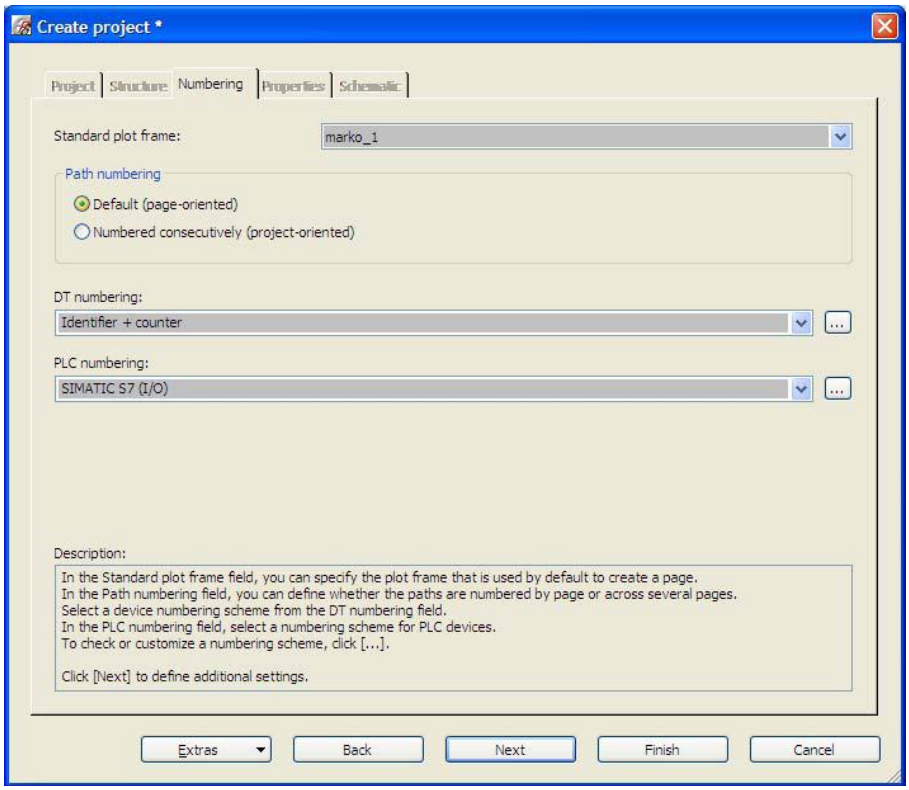
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Po končani nastavitvi zopet kliknemo Next.

V zavihku Numbering pod Standard plot frame določimo glavo projekta, ki jo lahko tekom projekta spremenimo/zamenjamo za celotni projekt ali posamezno stran.

DT numbering: določimo številčenje naprav, preko spustnega menija izberemo eno izmed prednastavljenih oblik številčenja ali pa si preko izbirnega gumba izdelamo lastno obliko številčenja.

PLC numbering: izberemo obliko številčenja PLC naprav glede na proizvajalca, ravno tako lahko ustvarimo lastno obliko številčenja preko izbirnega gumba.



Slika 8.10: Številčenje strani

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

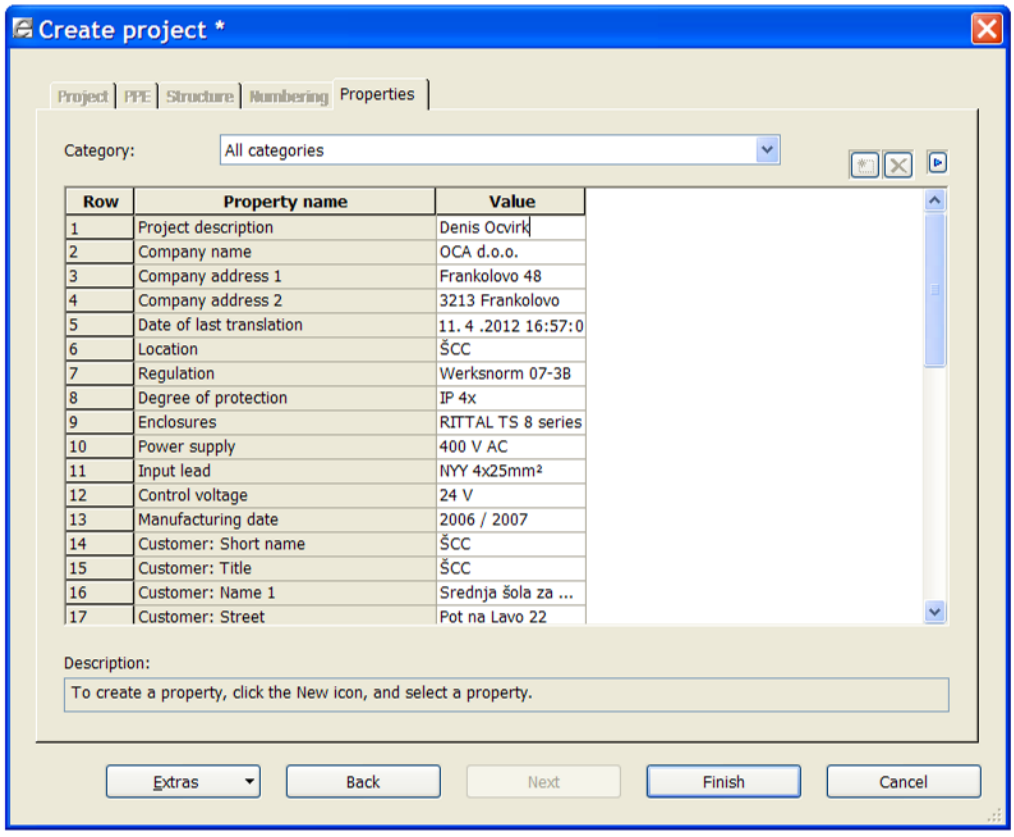
V naslednjem zavihku Properties vnesemo podatke o samem projektu, naročniku, …

Lastnosti lahko po potrebi dodajamo/odstranimo preko gumbov New in delete (tu lahko določimo tip projekta).

Pod lastnost Type of project nastavimo:

- Schematic project (navadni tip projekta) ali

- Macro project (projekt namenjen delanju makrojev, katere kasneje v enem koraku shranimo kot makroje.



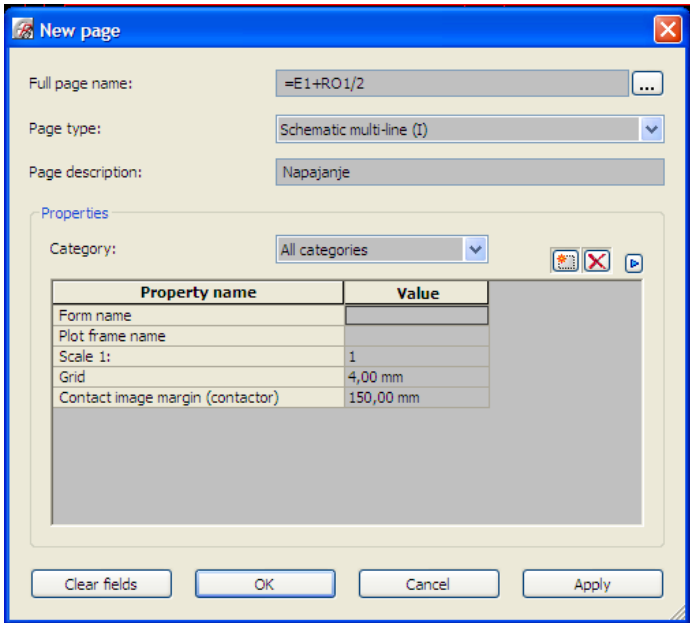
Slika 8.11: Podatki projekta

Vir: Lastni

Urejanje shem

Vstavljanje nove strani v projekt

Novo stran vstavimo [Page > new], odpre se okno New page.



Slika 8.12: Nova stran

Vir: Lastni

Full page name: vpišemo številko strani.

Page type: iz spustnega menija izberemo tip strani, ki jo potrebujemo (za tokovne sheme se uporablja Shematic multi-line oz. Shematic single-line, za grafične strani graphic ,…).

Page description: ime strani.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Properties: dodajamo/odvzemamo/ spreminjamo lastnosti za posamezno stran (določimo lastno formo za glavo strani, nastavimo privzeto velikost mreže (grid), … ostale lastnosti lahko dodamo preko gumba new).

Kasneje lahko spreminjamo podatke/lastnosti za stran enostavno tako da kliknemo na stran, ki jo želimo posodobiti (v drevesni strukturi).

Brisanje strani iz projekta

V oknu Pages se postavimo na stran, ki jo bomo izbrisali (desni gumb na miški >Delete).

Uporaba mreže [grid]

Za lažje postavljanje in povezovanje elementov je priročna uporaba mreže. Velikost mreže definiramo pod [Options > settings >user > graphical editing > general] v razdelku Default grid sizes, lahko predhodno dodelimo črkam A-E velikosti mreže.

Skozi projekt imamo na voljo opravilno vrstico, preko katere dostopamo do prednastavljenih velikosti mreže, vklopa mreže, postavljenja simbola na mrežo itd.

Vstavljanje simbolov

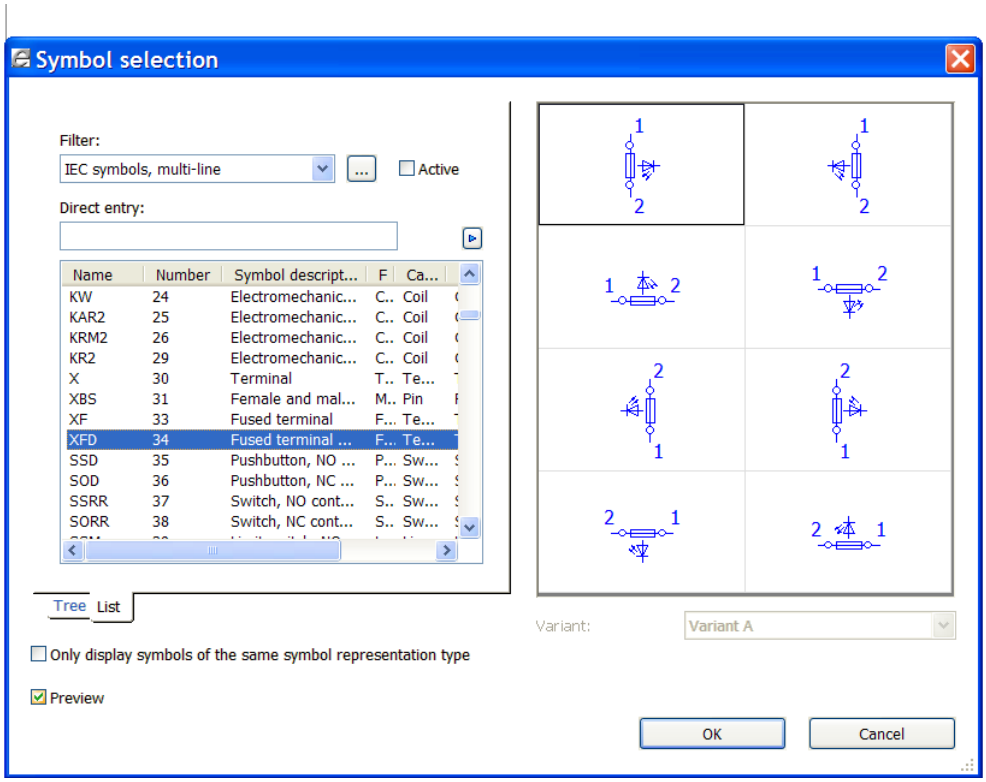
Razlika med simbolom in komponento:

Simbol: Simbol je grafika, ki je v e-planu namenjena prikazu funkcije, ne vsebuje pa nobene informacije.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Komponenta: Komponenta je ravno tako grafika, ki pa ima poleg same grafične podobe definirane tudi lastnosti, priključne sponke, kataloške številke, …)

Do simbolov iz baze simbolov dostopamo preko opravilne vrstice [insert > symbol], z desnim miškinim gumbom insert symbols ali s pomočjo bližnjice na tipkovnici, katero si uporabnik lahko prilagodi svojim željam in potrebam. Odpre se nam okno Symbol selection, kjer izbiramo med simboli.



Slika 8.13: Baza simbolov

Vir: Lastni

Na voljo imamo dva načina iskanja simbolov – preko zavihka **Tree** kjer iščemo elemente v drevesni strukturi glede na vrsto elementa.

Naslednja možnost je iskanje preko zavihka **List**, kjer iščemo simbole po imenu. Za lažje iskanje simbolov, imamo na voljo filter, v katerem izberemo katero knjižnico simbolov naj nam prikazuje, ter označimo izbirno okno **Active**.

Predhodna verzija programskega paketa EPLAN 5.7 je omogočala iskanje simbolov le po imenu. V navodilih bomo zaradi preglednejšega prikaza uporabljali prikaz simbolov v seznamu (List).

Osredotočili se bomo na IEC knjižnico simbolov *IEC Symbols, multi-line*, zato bomo na vrhu okna **Symbol selection** v razdelku **Filter** v spustnem meniju izbrali **Symbols, multi-line** ter potrdili okence **Active**.

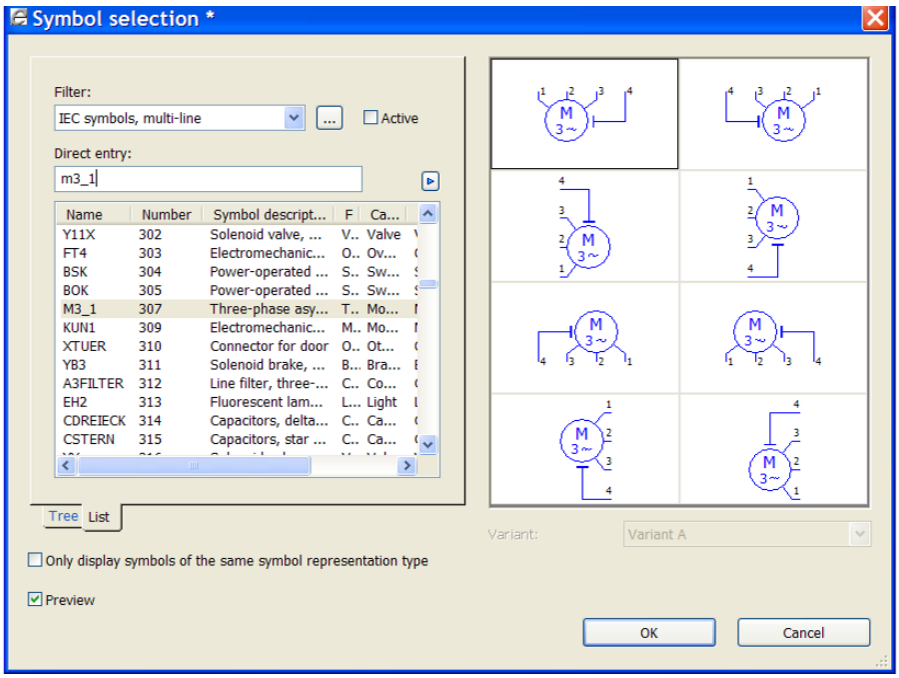
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

EPLAN omogoča delo z simboli definiranimi po različnih standardih, tako je programskemu paketu EPLAN poleg IEC-jeve knjižnice simbolov priložena še knjižnica simbolov po ameriškem standardu (NFPA).

Knjižnice simbolov enostavno dodajamo v projekt ***[Options >Settings > Projects > [ImeProjekta] >Management > Symbol libraries]***.

Primer iskanja motorja:

V okno direct entry vpišemo oznako za motor [M3_1]



Slika 8.14: Vstavljanje simbolov

Vir: Lastni

V desnem delu okna imamo prikazanih 8 različic simbolov (4 rotacije in 4 preslikave, označene so s črkami od A do H). Izberemo želeno različico (**A**) in potrdimo z **[OK]**. Ko postavimo simbol na shemo, se nam odpre okno. Nekateri podatki so že preddoločeni, vendar jih lahko tudi spreminjamo.

Zavihek **Motor:**

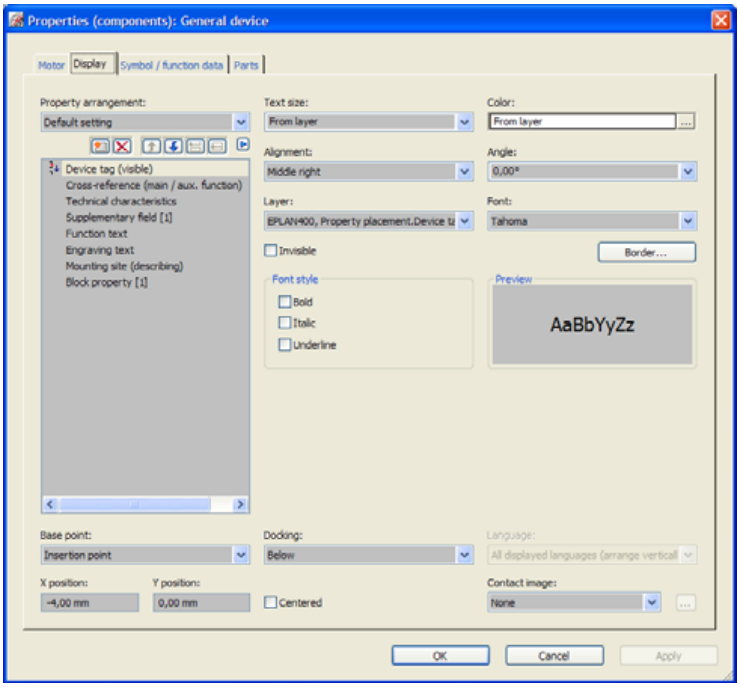
Display DT: po IEC standardu nam program avtomatsko označi simbol s pomišljajem, oznako simbola in zaporedno številko simbola v projektu [-M1]. Mogoče je pa tudi na začetku oz. tekom izdelave projekta prilagoditi označevanje elementov [za spremembo preštevilčenja celotnega projekta se postavimo na vrh projekta in gremo pod ***[Project data > Devices > number].***

Connection point designation: vsak simbol ima prednastavljene oznake povezav, katere je mogoče spremeniti. Med oznake povezav moramo vstaviti ločilni znak [¶ =Ctrl+Enter].

Technical characteristics: napišemo tehnične karakteristike simbola (npr. moč motorja).

Function text: funkcijski tekst, kateri se izpiše poleg simbola na shemi, ter kasneje v različnih izpisih/seznamih.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 8.15: Nastavitev

Vir: Lastni

Tukaj določimo, kaj vse bomo prikazali poleg samega simbola na shemi. Najpogosteje se prikazuje oznaka simbola [Device tag], tehnične karakteristike [Tehnical characteristics], funkcijski tekst [Function text] in kataloške oznake elementa [Part number]. Do dodatnih elementov, katere lahko prikažemo poleg simbola dostopamo preko gumba New.

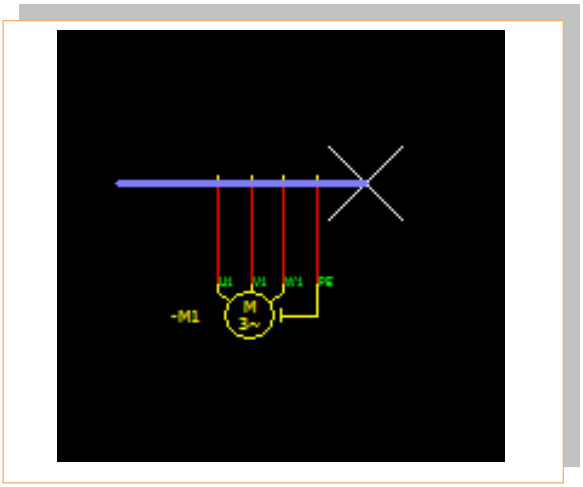
Zavihek **Parts:**

Preko gumba **Device selection** dostopamo do filtra, ki nam prikaže elemente iz baze, ki so skladni z našim simbolom oz. funkcijo simbola (v našem primeru prikaže vse kataloške številke trifaznih motorjev).

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Vstavljanje sponk

Simbol za sponke izberemo enako kot simbol za motor iz menija ***[insert > symbol]***, s tem da so sponke v programu označene z oznako ***[x]***. Na primeru postavljanja sponk v projekt, bomo omenili dodatno možnost programa eplan, torej skupinsko postavljanje simbolov, kar precej poenostavi in pohitri samo izdelavo projekta. Če gremo na shemo, kjer imamo že postavljen motor **–M1**, pred prvim konektorjem motorja kliknemo in držimo levo miškino tipko, potegnemo vodoravno preko vseh konektorjev motorja in spustimo tipko miške.

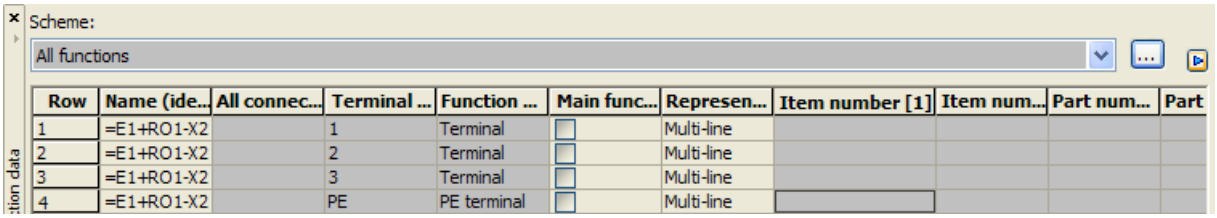


Slika 8.16: Izrisan element in povezave

Vir: Lastni

Tako smo avtomatsko postavili sponke. Če smo narisali vse sponke v isti ravnini, nam jih program avtomatsko številči kot sponke na eni spončni letvi. Če želimo naknadno urejati oznake ali tip sponk, enostavno označimo z miško sponke, katere želimo urediti, s klikom desnega gumba miške dostopamo do menija, v katerem izberemo ukaz *edit in table.*

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Row	Name (ide...	All connec...	Terminal ...	Function ...	Main func...	Represen...	Item number [1]	Item num...	Part num...	Part
1	=E1+RO1-X2		1	Terminal		Multi-line				
2	=E1+RO1-X2		2	Terminal		Multi-line				
3	=E1+RO1-X2		3	Terminal		Multi-line				
4	=E1+RO1-X2		PE	PE terminal		Multi-line				

Slika 8.17: Urejanje oznak

Vir: Lastni

Če pogledamo na shemo, bomo videli da imamo vse sponke oštevilčene po vrstnem redu, pri čemer želimo sami urediti, da bo sponka, ki je priključena na PE sponko motorja, ravno tako označena z oznako PE in ne številko 4. Ravno tako bomo tej sponki spremenili funkcijo, iz navadne sponke v PE sponko, kar bo prišlo prav pri postavitvi kabla, da bo program vedel katero žilo bo označil z oznako PE. Torej najenostavneje spremenimo oznako sponke in definicijo sponke, če kliknemo v okence posamezne sponke, ter spremenimo ime sponke.

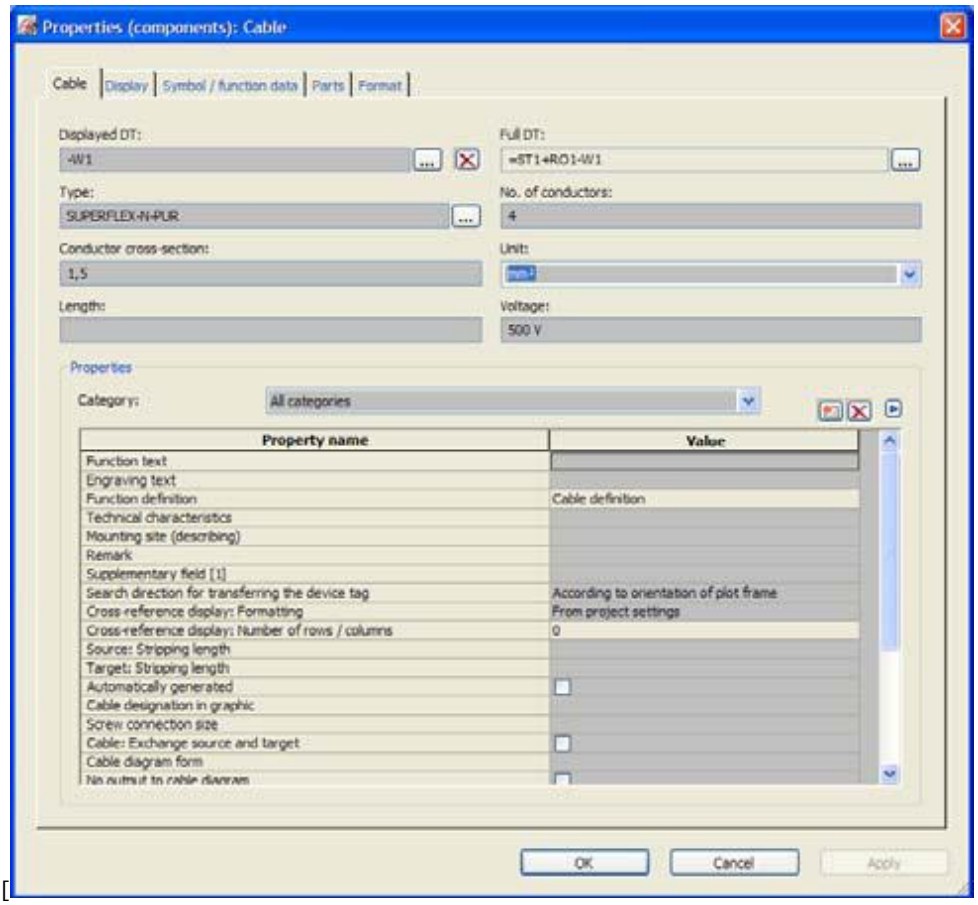
Za zamenjavo definicije sponke, pa enostavno kliknemo v okence ***Function definition*** posamezne sponke, kjer se nam pojavi izbirni gumb. Odpre se nam okno z seznamom funkcij, kjer izberemo *PE terminal 2 targets*.

Definiranje kabla

Kabel v projektu prikažemo z grafično črto definicije kabla ***[Insert > Cable definition]*** Na miškinem kazalcu dobimo simbol za kabel, kliknemo levo od prve žile, ter gremo preko vseh žil katere želimo imeti v kablu, ter na koncu ponovno kliknemo. Odpre se nam okno, v katerem najpogosteje pod zavihkom **Parts** s klikom na gumb **Device selection** izberemo iz baze kablov želeni kabel. S tem ko smo izbrali kabel iz baze, se

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

nam v zavihku **Cable** avtomatsko izpolnijo okenca **Type**, **Conductor cross-section**, **No. of conductors**, **Voltage**. Vsa našteta polja je mogoče tudi ročno spreminjati.



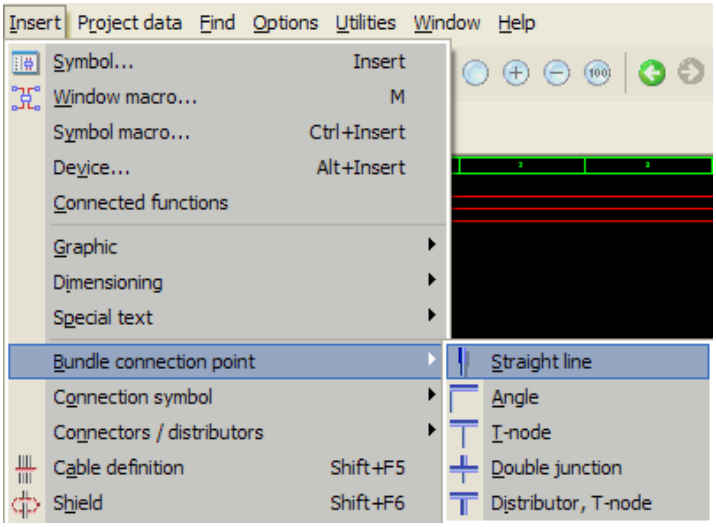
Slika 8.18: Definiranje kablov

Vir: Lastni

Definiranje kabelskega snopa

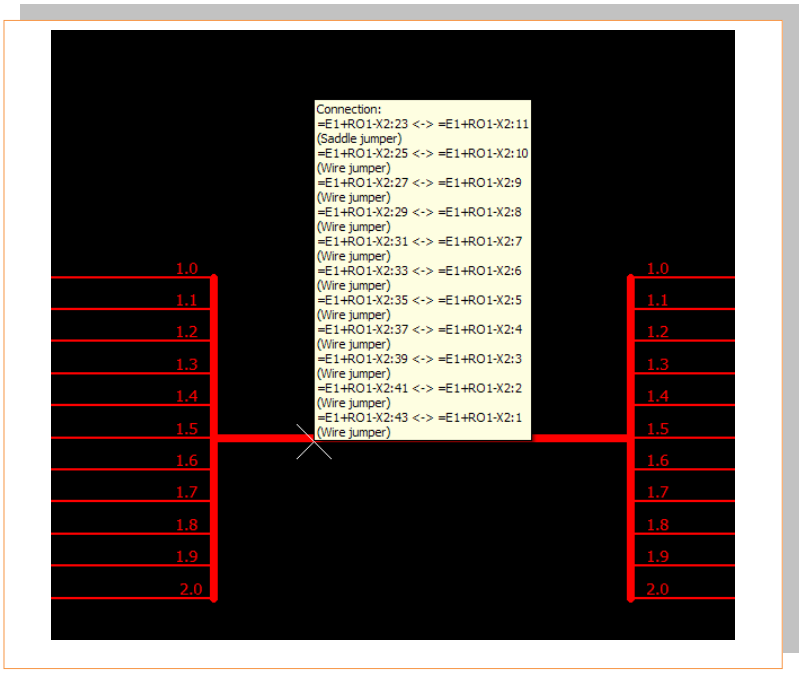
EPLAN P8 omogoča da več žil vodimo preko strani projekta kot en snop, kar je v primeru, da moramo po projektu voditi večje število žic, priročno. Do elementov za izdelavo snopa dostopamo preko menija ***[Insert > Bundle connection point].*** Slika prikazuje izgled takšnega snopa povezav, s prikazanimi povezavami. Pomembno je da povezave na obeh straneh snopa definiramo z enako oznako, saj le tako program ve, katere povezave so skupne.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 8.19: Definiranje kabelskega snopa

Vir: Lastni



Slika 8.11: Kabelski snop

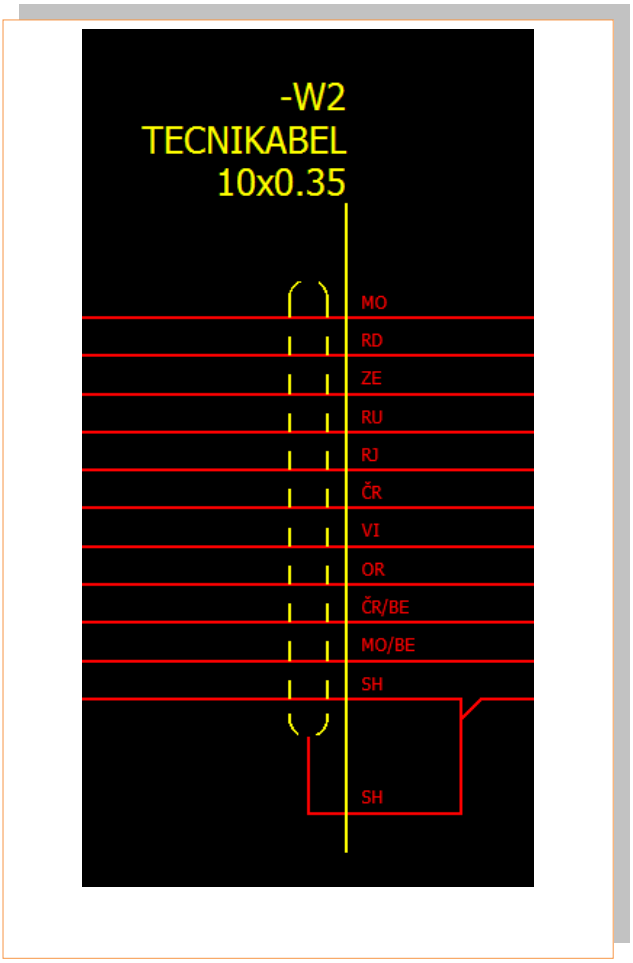
Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Definiranje kabelskega oklopa

Do oklopa pridemo preko menija ***[Insert > Shield]***. Ponavadi najprej definiramo kabel, kateri mora imeti definiran oklop, ki ga kasneje narišemo in povežemo s kablom.

Primer oklopa:

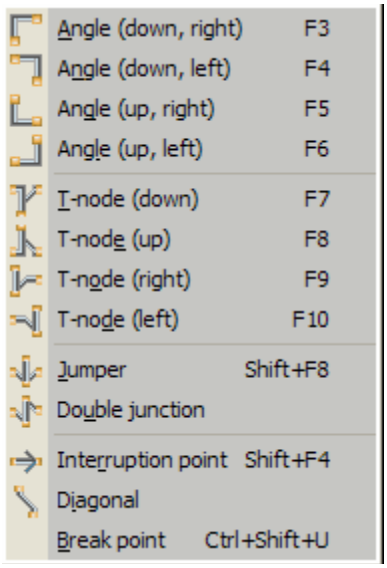


Slika 8.20: Kabelski oklop

Vir: Lastni

Vstavljanje povezav

Povezovanje med elementi se vršijo avtomatsko, ker program ne preverja funkcionalnosti povezave je potrebno biti previden. Kotne in T-povezave so nam dosegljive preko menija ***[Insert > Connection symbol].***



Slika 8.21: Vstavljanje povezav

Vir: Lastni

Povezave so razdeljene v nekaj sklopov: kotne povezave, T-povezave, posebne povezave (Interruption point [točka potenciala], diagonal [poševna povezava], break point [prekinitvena točka]).

Vse povezave je mogoče enostavno rotirati okoli osi s pomočjo tipke *[Tab]* oz. s kombinacijo tipke *[Ctrl]* + pomik miške za 360° okoli povezave. Tip povezave *Break point* je namenjena enostavnemu prekinjaju povezave med dvema elementoma.

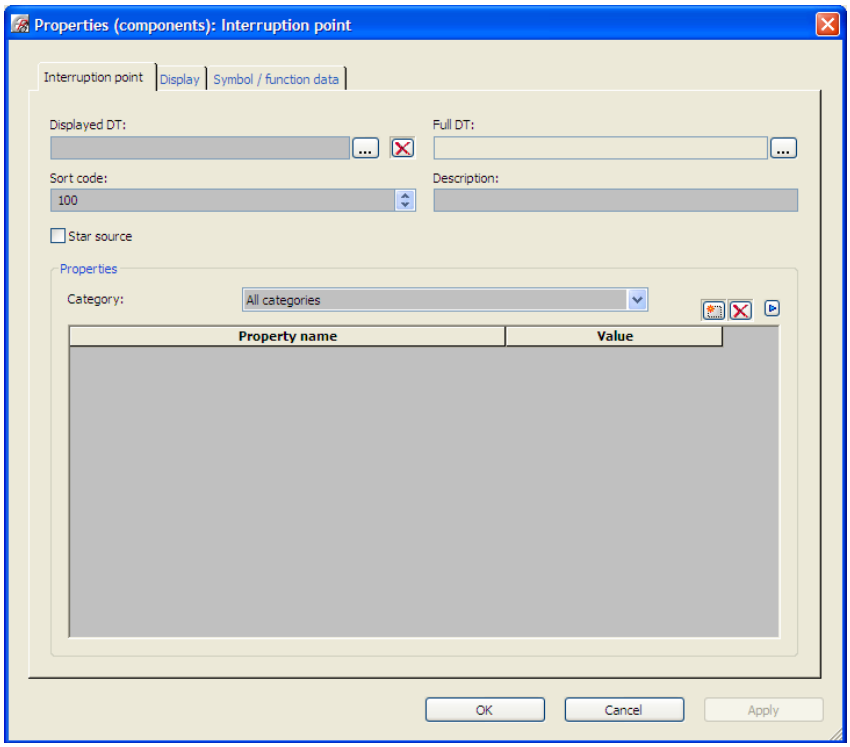
OPOMBA! V EPLAN P8 verziji lahko ravno tako uporabljamo Insertion point [view Insertion point], s pomočjo katere lahko prikažemo centralno točko elementa.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Primerna je predvsem če uporabljamo prekinitvene točke [break point], katere se običajno ne vidijo, ter če jih želimo odstraniti jih je potrebno označiti.

Definiranje potencialov

Z uporabo Interruption point [Insert > connection point > interruption point] enostavno definiramo potencial. Puščico za določitev začetka ali konca potenciala lahko z uporabo tipke [Tab] obračamo. Ko postavimo puščico na želeno mesto na strani tokovne sheme, se nam odpre pogovorno okno, kjer vpišemo pod Display DT ime potenciala, v primeru da imamo v projektu že definirane kakšne potenciale se poslužujemo izbirnega gumba, s katerim dostopamo do seznama vseh potencialov v projektu. V seznamu potencialov je vidno, kako so vodeni potenciali iz ene strani na drugo stran. Pomembno pri potencialih je da so v parih, torej da je potencial voden dalje ali pa zaključen na kakšno napravo.



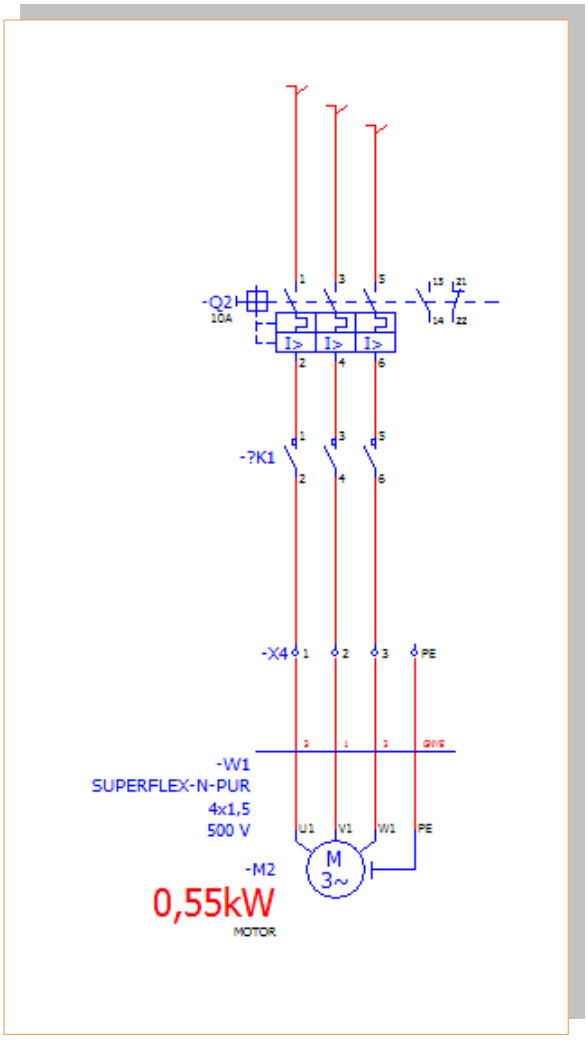
Slika 8.22: Definicija potencialov

Vir: Lastni

Makroji [izdelava/vstavljanje]

Osnovna funkcija makrojev je da si z njihovo uporabo poenostavimo izdelovanje projektov. Makroje običajno uporabljamo za sklope naprav, katere pogosto uporabljamo, ter se z uporabo makrojev izognemo risanju enakih sklopov.

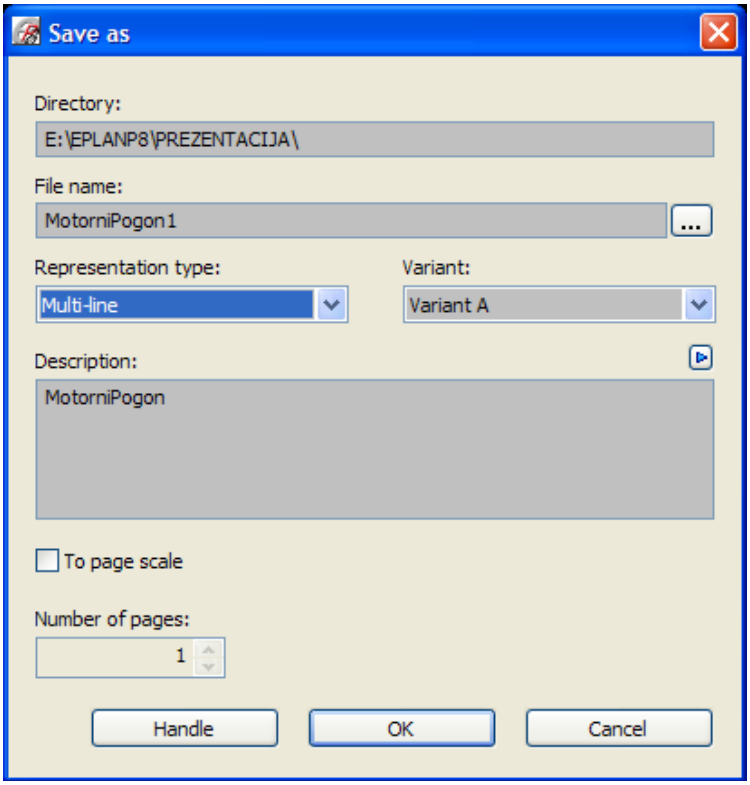
Primer enostavnega sklopa:



Slika 8.23: Izris el. sklopa

Vir: Lastni

Makroji se delijo na window, symbol in page makroje. Postopek izdelave window oz. symbol makroja je sledeči: označimo sklop elementov, ki jih želimo shraniti kot makro, ***[edit > create window macro / create symbol macro]***. Odpre se nam okno, v katerem določimo ime in mapo v katero se bo shranil makro, različico in opis makroja.



Slika 8.24: Kreiranje makrojev

Vir: Lastni

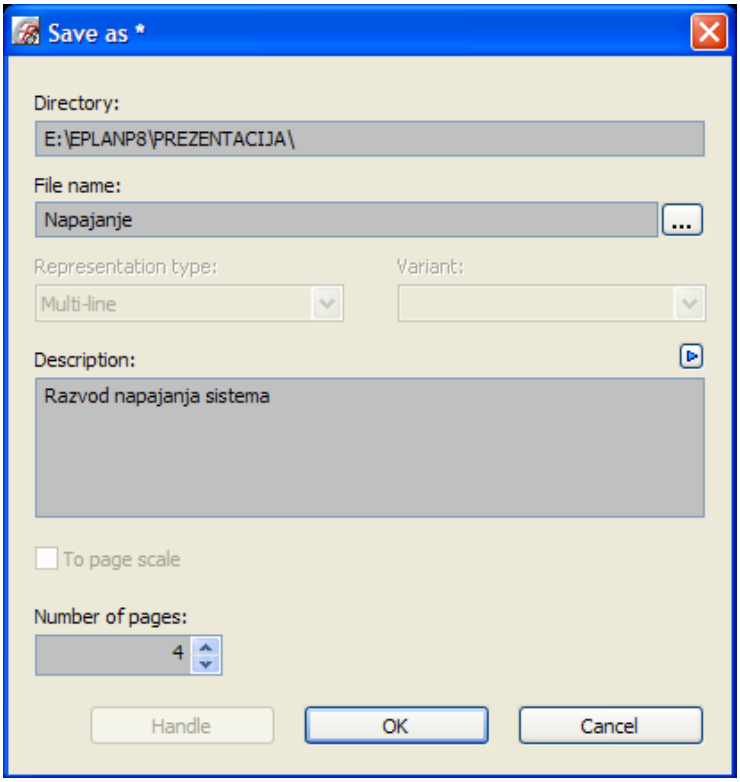
Makroje vstavljamo preko menija ***[insert > window macro / symbol macro]***. Poiščemo naš makro in ga enostavno vstavimo v projekt. Makroje je mogoče shranjevati v več različicah (8 različic), kar je uporabljeno npr. pri makrojih za PLC naprave.

Page makroji so uporabni, ko imamo v večih projektih podobnih več strani, katere shranimo kot page macro in jih kasneje vstavljamo in modificiramo v ostalih projektih. Page makro izdelamo enostavno tako, da se postavimo na prvo stran, katero želimo shraniti kot page makro, gremo na [Page > page macro > create], odpre se nam

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

pogovorno okno, v katerem določimo mapo, ime page makroja, napišemo opis makroja ter določimo število strani, ki jih bomo shranili v makro.

Če želimo shraniti več strani pod en page makro, se postavimo na prvo stran projekta, ki jo želimo imeti v makroju. Če npr. določimo, da bomo shranili 4 strani, to pomeni da bomo shranili stran na kateri se trenutno nahajamo in 3 strani, ki sledijo prvi strani.

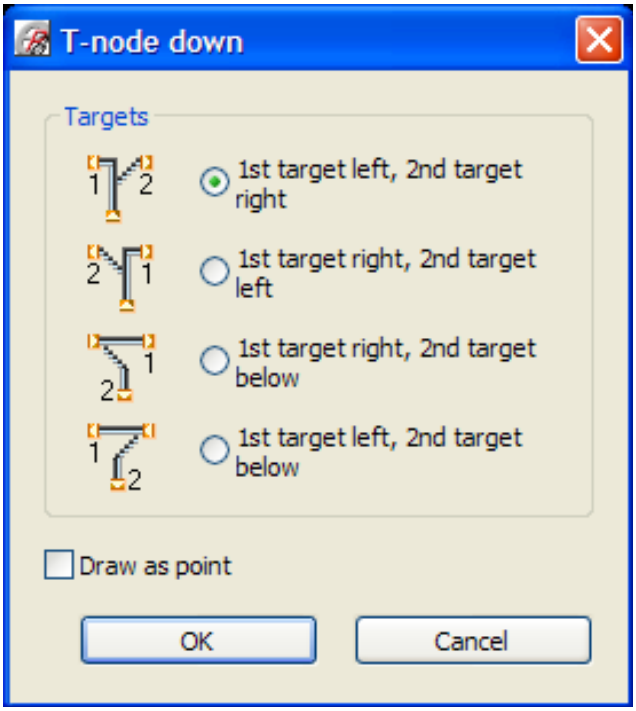


Slika 8.25: Makro

Vir: Lastni

Vstavljanje T-povezav

V razdelku povezav imamo 4 različice T-povezav, za vsako smer po ena. Do te povezave dostopamo preko *[Insert > Connection symbol > T-node (down)]*, odpre se nam izbirno okno.



Slika 8.26: T povezave

Vir: Lastni

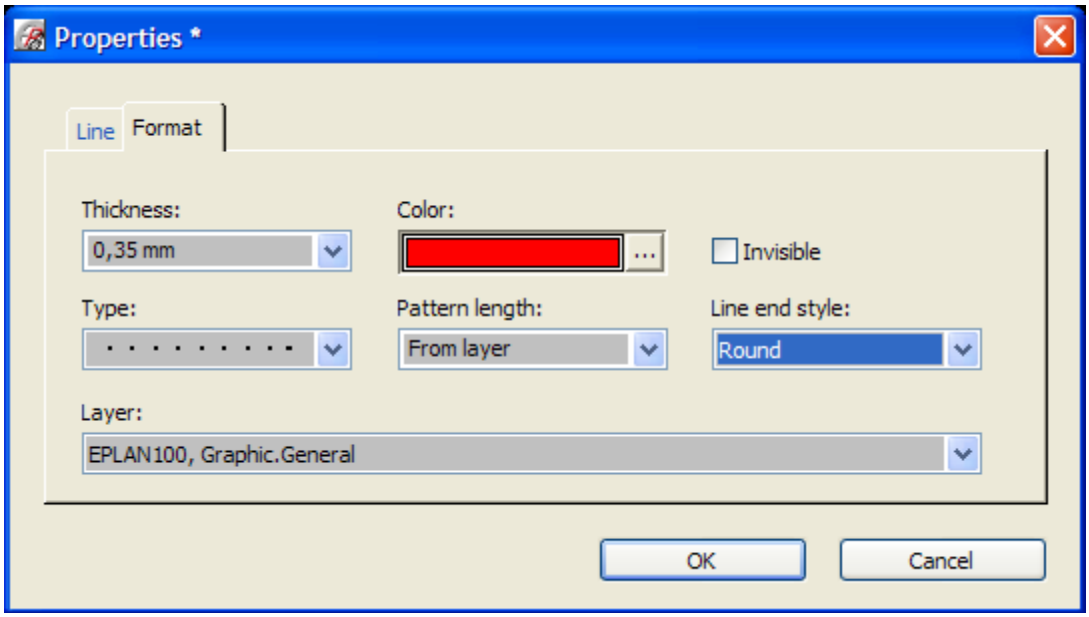
Eplan nam ponuja za prikaz vozlišča ožičenja dve možnosti, prikaz s piko, ter prikaz z inteligentno povezavo, ki prikazuje s katerega elementa prihaja povezava, do katerega elementa, ter kam se nadaljuje. Na vidimo možnost izbire štirih rotacij intelegentne povezave. Rotacijo vozlišča izbiramo lahko s tipko [Ctrl]. Izgled vozlišč lahko na koncu projekta spremenimo v samih nastavitvah projekta [Options > Settings > Projects > [ImeProjekta] > Graphical editing > General > Display connerction junctions as:].

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Prosta grafika

V eplan-u P8 je preprosto rokovanje z grafičnimi elementi, delo je zelo podobno delu v AutoCAD-u. Do posameznih ukazov dostopamo preko menija *[insert > graphic]*.

Vsak element proste grafike (črta, krog, …) lahko kliknemo (dvojni klik) in v oknu uredimo izgled elementa (debelina, barva, tip črt, …)



Slika 8.27: Definiranje grafike

Vir: Lastni

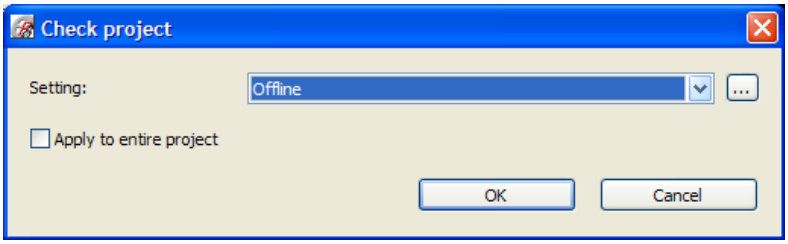
Program nam omogoča tudi enostavno kotiranje elementov ki smo jih narisali *[insert > dimensioning].*

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Odpravljanje napak v projektu

Med samo izdelavo projekta ali na koncu projektiranja, preverimo pravilnost izdelave projekta. Do ukaza za preverjanje projekta dostopamo preko menija ***[Project data > messages > Check projects].*** Odpre se nam okno, v katerem lahko izbiramo med različnimi profili nastavitev, privzeto je na voljo nekaj običajnih profilov (najpogosteje se uporablja profil Offline), lahko si pa preko izbirnega gumba uredimo obstoječi profil ali pa ustvarimo novi profil.

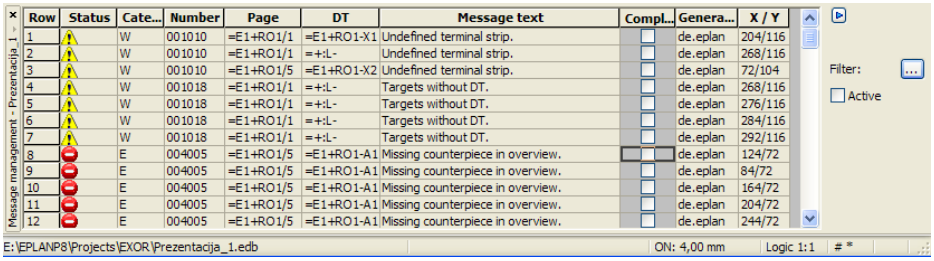
Program nam omogoča da pregledujemo projekt po straneh ali pa v celoti, to izbiro nam omogoča potrditveno okno *Apply to entire project*



Slika 8.28: Preverjanje projekta

Vir: Lastni

Ko je program zaključil s pregledom projekta, se postavimo na ***[Project data > messages > Management]***



Slika 8.29: Pregled napak

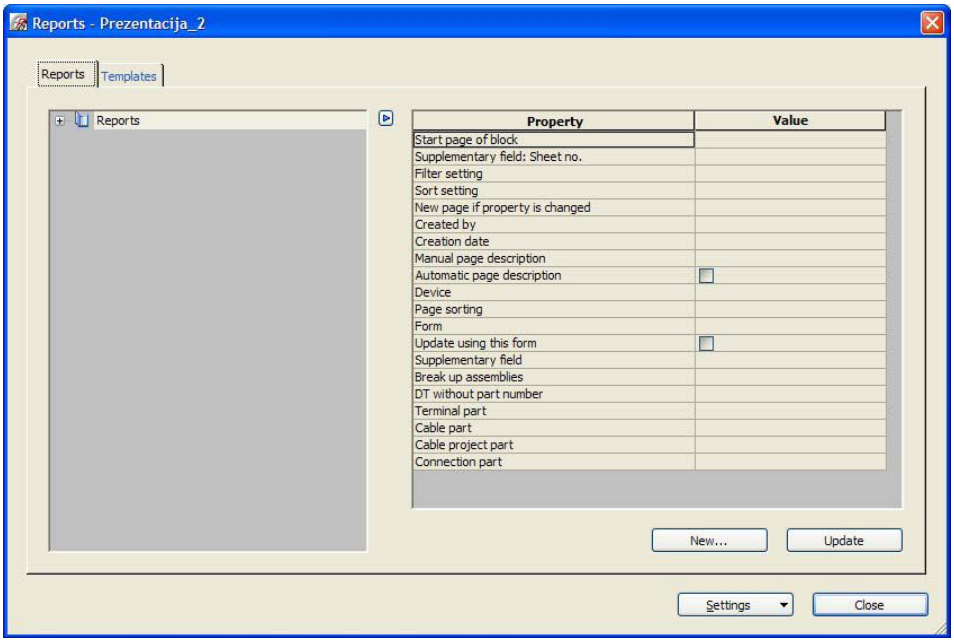
Vir: Lastni

Na desni strani okna imamo ponovno izbirni gumb, s katerim določimo katere informacije bomo prikazali v seznamu (npr. napake, opozorila, opombe), ter potrdimo polje Active. Z dvoklikom miške na ustrezno vrstico, nas program locira na mesto določene napake. Če potrebujemo podrobnejši opis napake kliknemo z miško na želeno vrstico, ter pritisnemo tipko [F1].

Izdelava dokumentacije

Programski paket EPLAN omogoča hitro in preprosto izdelavo vseh vrst seznamov in dokumentacije. Odvisno od verzije programa, imamo na voljo več kot 25 različnih vrst dokumentacije, pri čemer je vsaka v treh različicah [statična forma, dinamična in grafična forma], ki si jih vsak uporabnik lahko prilagodi svojim zahtevam in potrebam.

Vse forme so prevedene v več jezikov, med drugim tudi v slovenščino. Do dokumentacije dostopamo preko ***[utilities > reports > generate].***



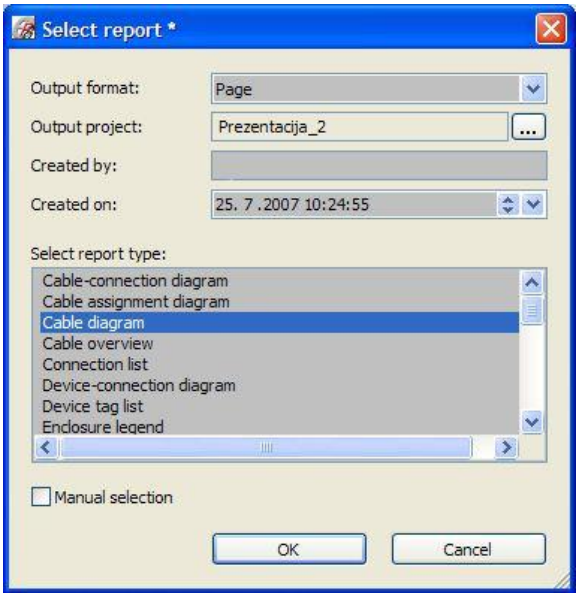
Slika 8.30: Izdelava dokumentacije

Vir: Lastni

Na voljo imamo dva načina generiranja dokumentacije in seznamov.

Pri prvem načinu dostopamo preko zavihka *Reports*. V oknu Output to pages [Settings > Output to pages] imamo prikazane vse vrste dokumentacij, katerim v razdelku *Form* lahko določimo katero formo bomo uporabili, kam v projektu bomo postavili dokumentacijo (določimo v razdelku *Page sorting*), itd.

Preko gumba *New*, dostopamo do okna izbora vrste seznama.

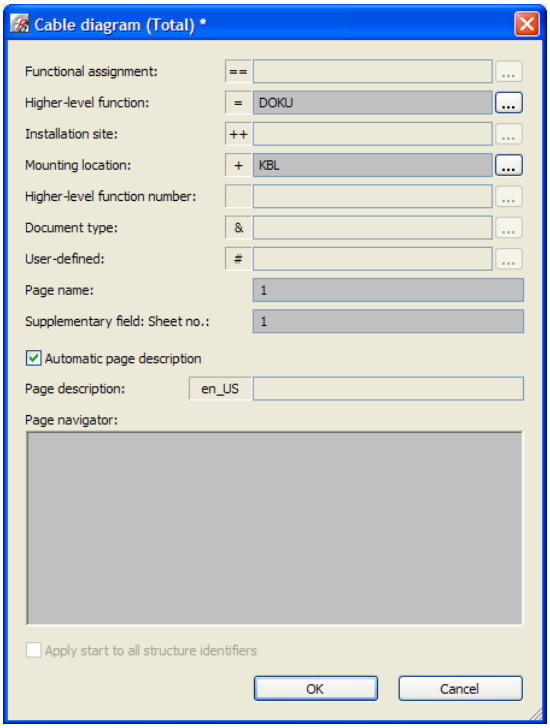


Slika 8.31: Izbira izpisa

Vir: Lastni

Output format: izberemo izris seznama [možnosti sta *Page* – postavitev seznama na svojo stran in *Manual placement* – postavitev seznama ročno (npr. poleg montažne plošče).

Select report type: iz seznama izberemo želeno vrsto forme. S klikom na gumb [OK] potrdimo izbiro, odpre se nam novo okno, v katerem določimo katere elemente želimo imeti v seznamu, ter kako jih želimo razporediti v seznamu. Če želimo aktivirati filter moramo potrditi poleg načina filtriranja še potrditveno okence *Active*. S ponovnim klikom na gumb [OK], se nam odpre novo okno, v katerem določimo lokacijo seznamov dokumentacije.

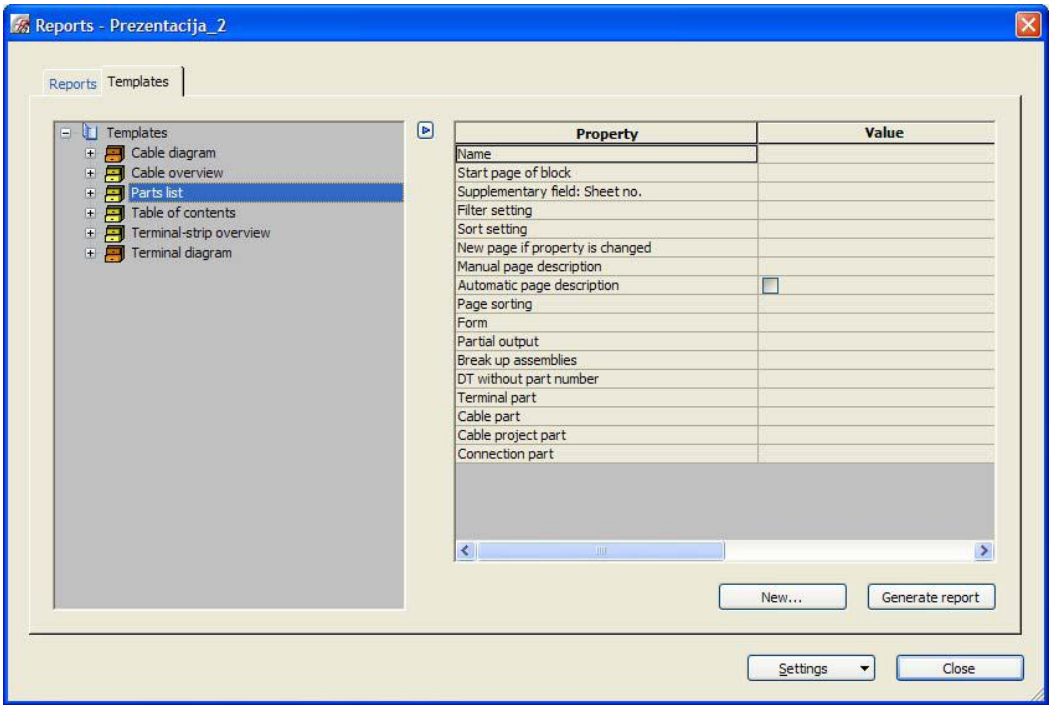


Slika 8.32: Izpis

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

To je bil primer postavljanja vsakega seznama posamezno, v primeru da imamo v več projektih enake zahteve glede vrste dokumentacije, gremo na zavihek *Templates.*



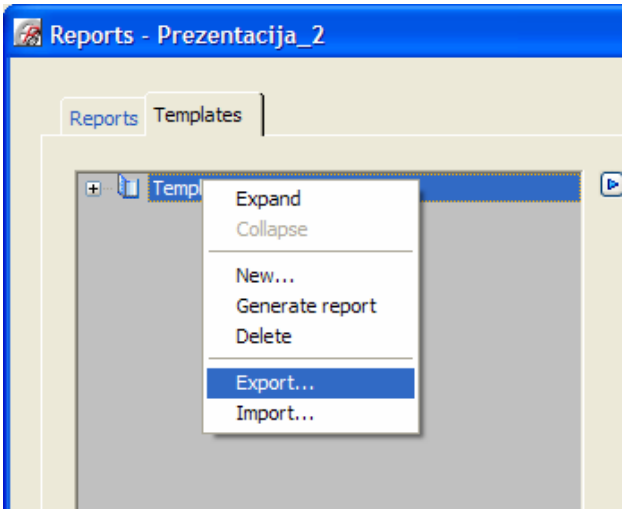
Slika 8.33: Izpis seznamov

Vir: Lastni

Tu preko gumba *New* izberemo vrste seznamov, določimo filter in lokacijo, kje se bo nahajala posamezna vrsta dokumentacije znotraj projekta. Na ta način naredimo predlogo [template] seznamov. Ko smo izdelali predlogo seznamov, lahko le-to predlogo preprosto izvozimo, z desno tipko miške kliknemo na *Templates* in iz priročnega menija izberemo opcijo *export* ter določimo lokacijo in ime datoteke, v katero se bo shranila predloga [template].

Na enak način lahko uvozimo predlogo [Template], vendar izberemo *Import* in poiščemo na disku, kje imamo shranjeno datoteko.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 8.34: Izbira predloge

Vir: Lastni

Ko smo določili vse sezname, ki jih potrebujemo, se poslužimo gumba *Generate report*, ki nam zgenerira vse sezname, ki smo jih imeli določene na listi predlog.

Arhiviranje projekta

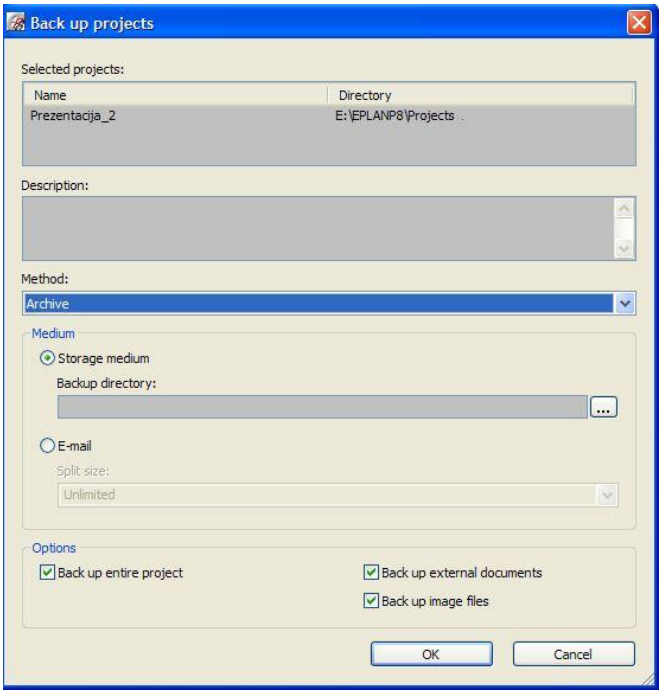
Ko smo projekt dokončali, odpravili vse napake, vnesli vse popravke in ga posredovali dalje, ga je priporočljivo arhivirati [Back up]. Na voljo imamo več načinov arhiviranja, ponavadi projekt arhiviramo na strežnik, kjer imamo vse dokončane projekte, ter s tem na svojem disku prostor za naslednje projekte. Kliknemo na projekt, ki ga želimo arhivirati. Do funkcije arhiva dostopamo preko menija ***[Project > Back up > Project]***.

Odpre se nam okno, kjer v razdelku *Description* lahko vpišemo kratek opis projekta, pod *Method* izbiramo med različnimi načini arhiviranja:

- *Save additionally*: arhiviramo projekt, vendar imamo na disku še vedno izvorni projekt.
- *File off for external editing*: arhiviramo projekt, ter ga zaščitimo pred pisanjem.
- *Archive*: arhiviramo projekt na drugo lokacijo, izvorni projekt se izbriše.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

- V razdelku *Medium* določimo na katero lokacijo [disk] bomo shranili arhiv projekta, na voljo imamo možnost shranjevanja na disk in možnost pošiljanja arhiva preko elektronske pošte, pri čemer lahko določimo na kako velike dele nam EPLAN razdeli arhiv. Na dnu pogovornega okna imamo še razdelek *Options*, v katerem določimo kaj vse se naj poleg samega projekta arhivira.
- *Back up entire project:* poleg projekta se arhivirajo vse datoteke, ki so v mapi projekta.
- *Back up external documents:* poleg projekta arhiviramo tudi zunanje dokumente, ki smo jih uporabili v projektu.
- *Back up image file:* poleg projekta arhiviramo tudi grafične datoteke (različne slike, ki smo jih uporabili v samem projektu).



Slika 8.35: Nastavitev arhiviranja

Vir: Lastni

Izvoz projekta

EPLAN nam nudi možnost izvoza celotnega projekta oz. posameznih strani v različne formate [AutoCAD-ov format (dxf/dwg), slikovne datoteke (jpg/bmp/png/tif/gif), pdf]. Za izvoz v format pdf, je značilno da EPLAN izdela »intelegentno« pdf datoteko, katera obdrži strukturo projekta, omogoča hitre skoke iz glavnega elementa na podrejeni element (npr. s klikom na kontakt tuljave nas prestavi na navitje tuljave).
Izvoz v pdf datoteko, dobro služi namenu pregledovanja projekta, česar se podjetja poslužujejo ko dostopajo do projektov ljudje, ki potrebujejo samo vpogled v projekt, brez potrebe po dodelavi oz. spreminjanju projekta in nebi bilo smiselno kupovati licence za program.

Do menija za izvoz projekta oz. posameznih strani dostopamo preko ***[Page > Export]***, kjer izberemo format izvozne datoteke.

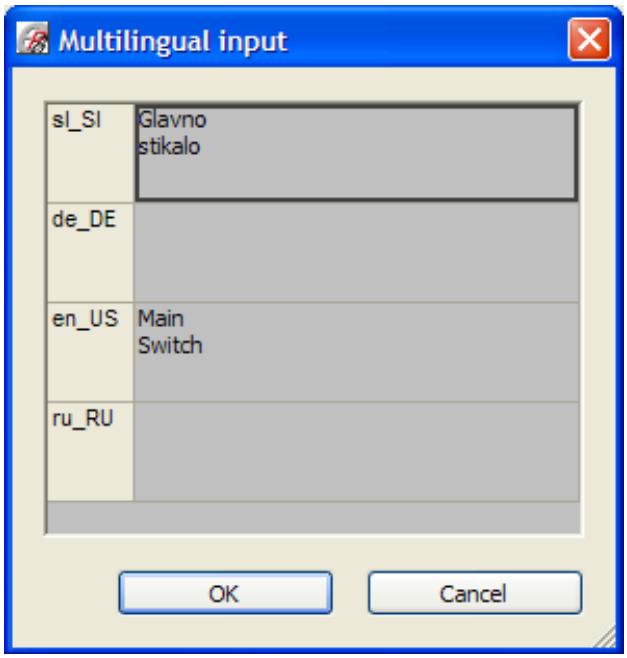
Prevajanje

Odvisno od verzije programskega paketa je mogoče prevajati projekt na različne načine. Verzija compact omogoča prevajanje edino on-line, kar pomeni, da sproti, ko rišemo projekt, pišemo opise in lastnosti elementov v več različnih jezikih. Kasneje, ko končamo projekt, lahko enostavno zamenjamo jezik, ki ga želimo prikazovati v projektu.

Ko kreiramo projekt nastavimo v meniju ***[Options > Settings > Projects > [ImeProjekta] > Translation > general]*** v okencu *Translation* jezike, v katerih želimo imeti projekt.

Ko rišemo projekt, z desnim gumbom kliknemo na okence v katerem želimo pisati besedilo v več različnih jezikih, v priročnem meniju izberemo opcijo **Multilingual input**, odpre se nam okno, v katerega vpišemo v posameznih jezikih izraze.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



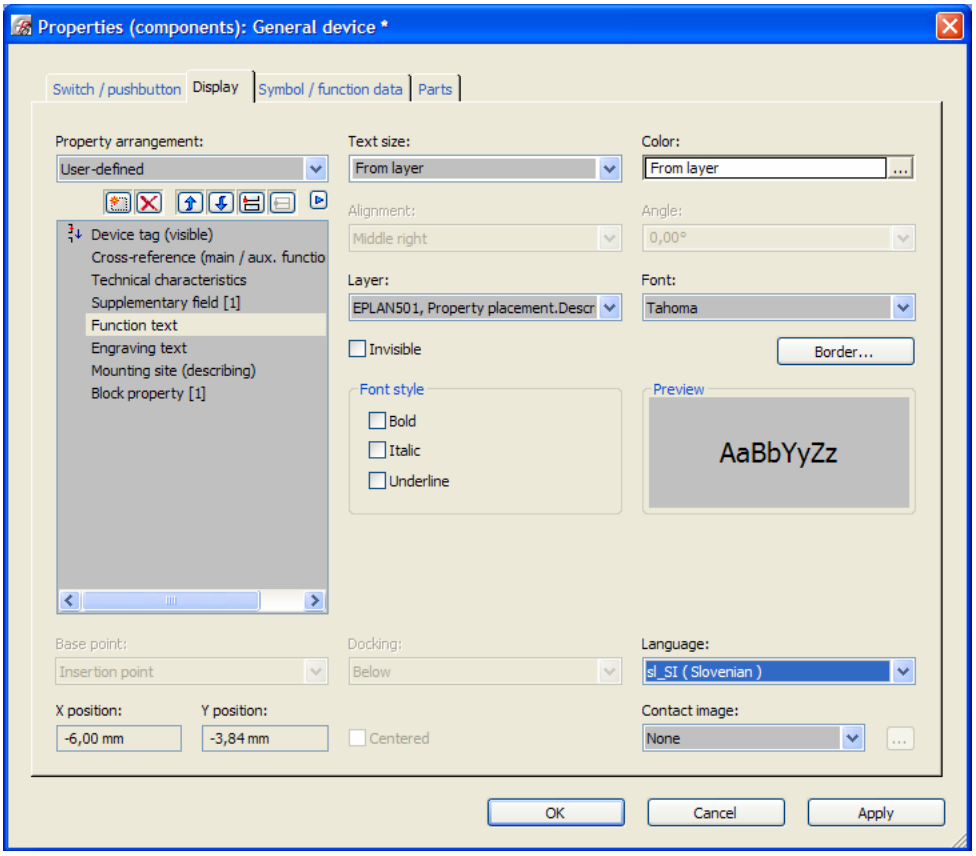
Slika 8.36: Prevajanje izrazov

Vir: Lastni

Tekom projekta lahko določamo kateri jeziki se bodo videli v projektu, mogoče je izbirati med posameznimi jeziki, ki smo jih določili za projekt, lahko prikazujemo vse jezike sočasno itd. ***[Options > Settings > Projects > [ImeProjekta] > Translation > general]*** v okencu *Display*.

Kateri jezik bomo prikazovali, lahko določimo za vsak element posebej ali pa za celotni projekt skupaj. Seveda pa je mogoče tudi da določenemu elementu predpišemo izraz samo v enem jeziku, ne glede na to kateri jezik je določen za celoten projekt, to dosežemo če definiramo pod Language, *One language (fixed)*.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 8.37: Nastavitev jezika

Vir: Lastni

Layer management

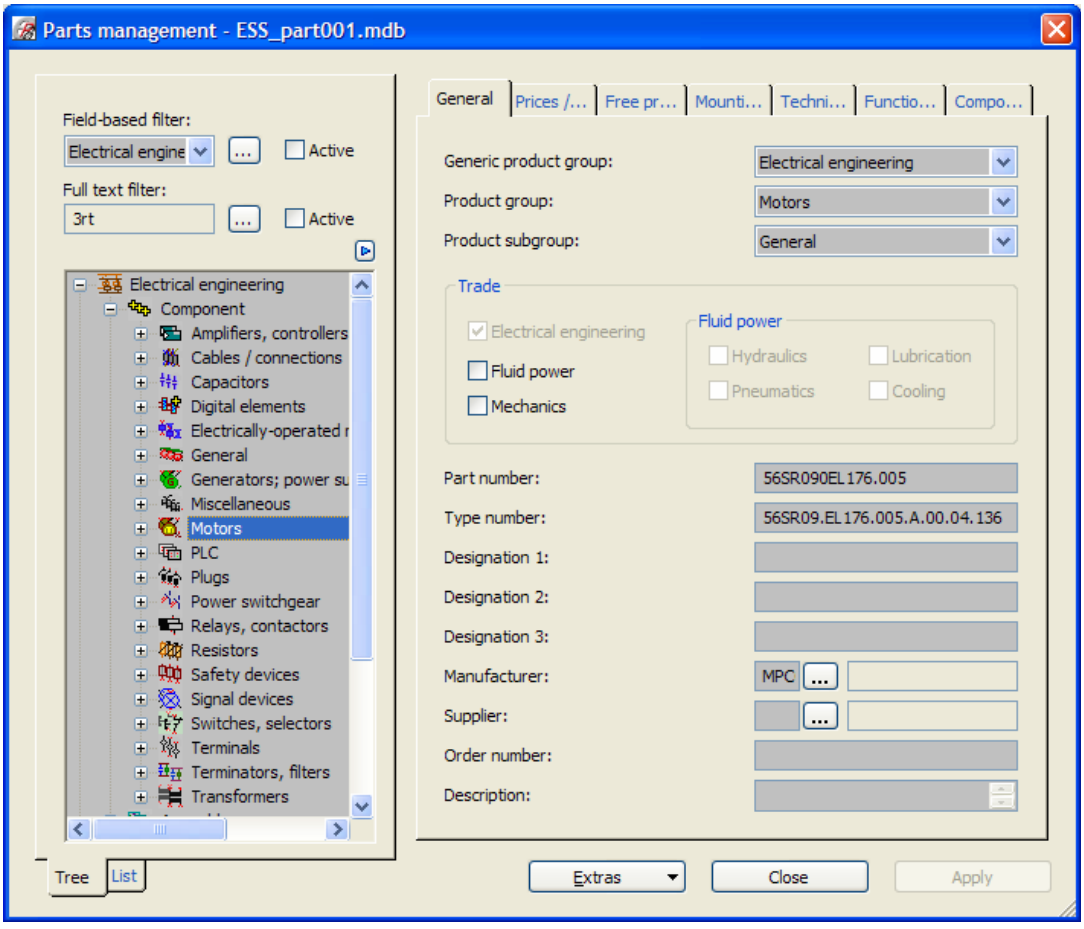
Za risanje lahko uporabljamo različne plasti (layer). Upravljanje z plastmi omogoča podobno kot v CAD programih urejanje in izdelavo novih plasti, s katerimi enostavno določamo barve, velikosti pisav, debelino, vrsto linij, barvo simbolov, itd. Če želimo v projektu spremeniti vsem simbolom, ki so na isti plasti (npr.) barvo, to preprosto naredimo v layer managementu, s čimer določimo vsem elementom te plasti novo lastnost.

Do layer managementa dostopamo preko menija ***[Options > Layer management]***.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Baze elementov

EPLAN-u je priložena baza elementov več kot 40 proizvajalcev, vendar je potrebno omeniti da proizvajalci niso predstavljeni s celotnim seznamom proizvodov. Glede na potrebe projekta, je mogoče osnovno bazo dopolnjevati, izdelati lastno bazo podatkov, brisati podatke iz baze, itd. Do baze dostopamo preko menija ***[utilities > parts > management]***.



Slika 8.37: Baza elementov

Vir: Lastni

Če želimo dodati komponento v bazo proizvodov, se postavimo v drevesni strukturi v poddirektorij, v katerega sodi naša komponenta, saj je od samega poddirektorija odvisen seznam lastnosti za komponento.

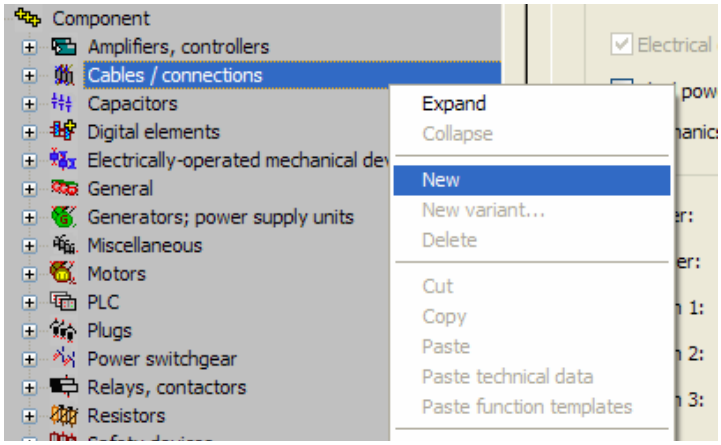
Torej če želimo dodati motor v bazo podatkov, se postavimo v drevesni strukturi na [Electrical engineering > component> motors], z desnim gumbom kliknemo na motors, odpre se nam priročni meni v katerem izberemo New. Tako smo kreirali novo komponento, kateri dodelimo ime, kataloško številko, tehnične lastnosti, sliko, makro, itd. Ko smo napisali vse lastnosti za element, kliknemo na eno drugo komponento, ter nas vpraša če želimo komponento, ki smo jo ravnokar izdelali shraniti, kliknemo Yes. Tako smo kreirali komponento. Ravno tako lahko izdelamo svoje releje, kable in ostale komponente. Ravno tako je preprosto ustvarjati lastne baze, v katere lahko uvažamo ostale).

Svojo bazo kreiramo preko [utilities > parts > management > extras > New database]. Sedaj imamo novo bazo v katero lahko sedaj ročno vpisujemo komponente, lahko pa tudi preko [utilities > parts > management > extras > import] uvažamo baze, ki so dosegljive na spletni strani zastopnika za EPLAN ali pa na CD-ju z bazami in makroji.

Vpis novega kabla v bazo proizvodov

Če želenega kabla še ni v bazi proizvodov, ga lahko enostavno sami dodamo. Gremo v bazo proizvodov ***[Utilities > parts > management]***, se v drevesnem seznamu pomaknemo do seznama kablov, z desnim gumbov pridemo do menija, kjer izberemo New.

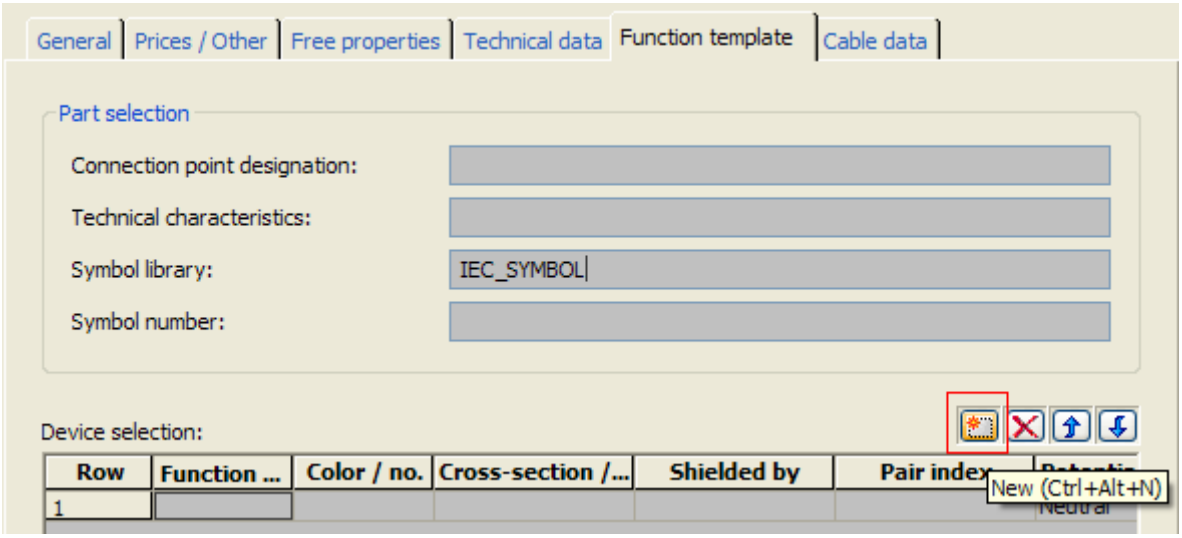
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 8.38: Vpis novega elementa

Vir: Lastni

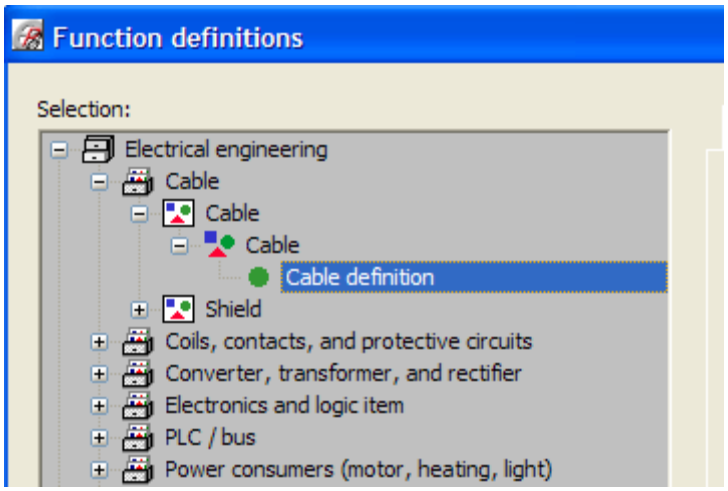
Izdela se nam novi kabel z imenom New_1, dopolnimo podatke o novem kablu, kot so kataloška številka, tip kabla, proizvajalec, najpomembnejše je definiranje števila in vrste žil, kar izvedemo pod zavihkom **Function template**. Kliknemo na gumb New



Slika 8.39: Podatki o elementu

Vir: Lastni

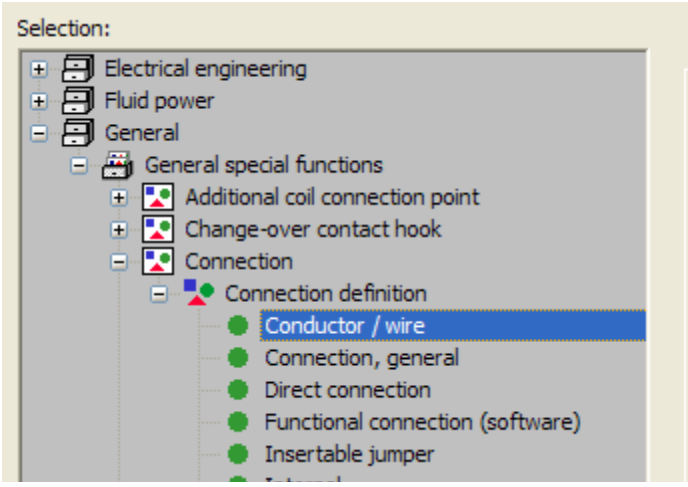
preko katerega najprej definiramo kabel, kasneje pa vsako žilo posebej. Vsaki žili določimo tip (navadna žila, PE žila, …), presek, barvo oz. številko. Torej začnemo z definicijo kabla, kliknemo v okence Function definition



Slika 8.40: Definicija kabla

Vir: Lastni

Sedaj definiramo še posamezne žile, preko gumba New dodajamo žile in jim v okencu Function definition določimo funkcijo žice, ter v okencu potential vrsto (L, PE, …).



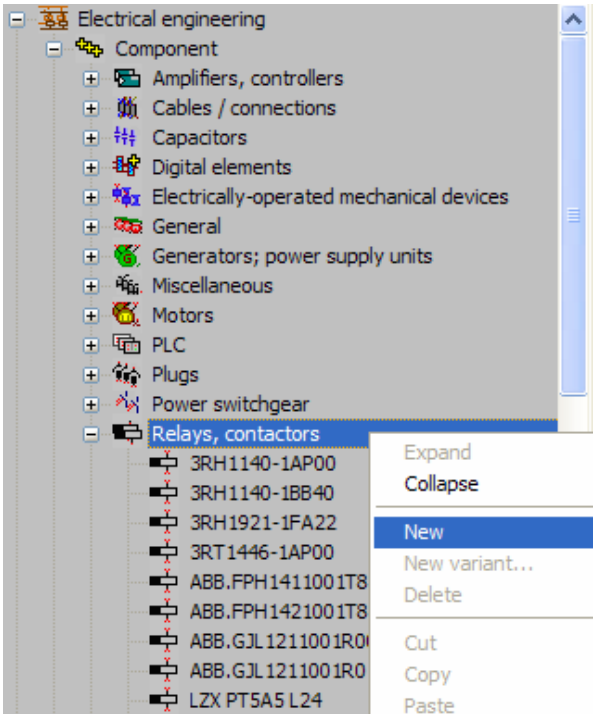
Slika 8.41: Definicija lastnosti elemeneta

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Vpis novega releja v bazo proizvodov

Če želenega releja še ni v bazi proizvodov, ga lahko enostavno sami dodamo. Gremo v bazo proizvodov ***[Utilities > parts > management]***, se v drevesnem seznamu pomaknemo do seznama relejev, z desnim gumbov pridemo do menija, kjer izberemo New.



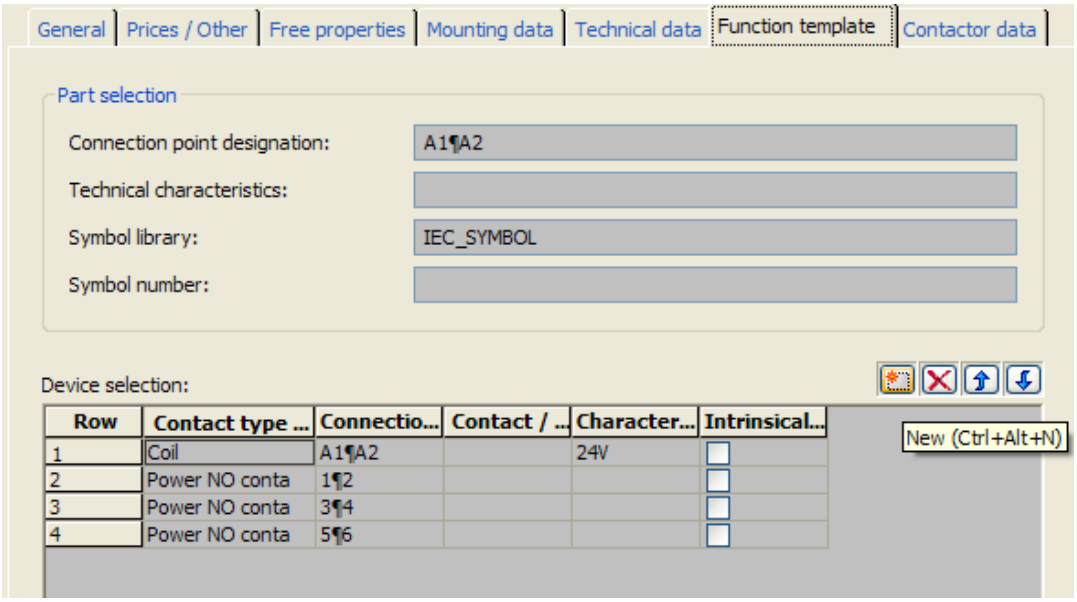
Slika 8.42: Primer vnosa rekeja

Vir: Lastni

Izdela se nam novi rele z imenom New_1, dopolnimo podatke o novem releju, kot so kataloška številka, tip releja, proizvajalec, najpomembnejše je definiranje števila in vrste kontaktov, kar izvedemo pod zavihkom **Function template**.

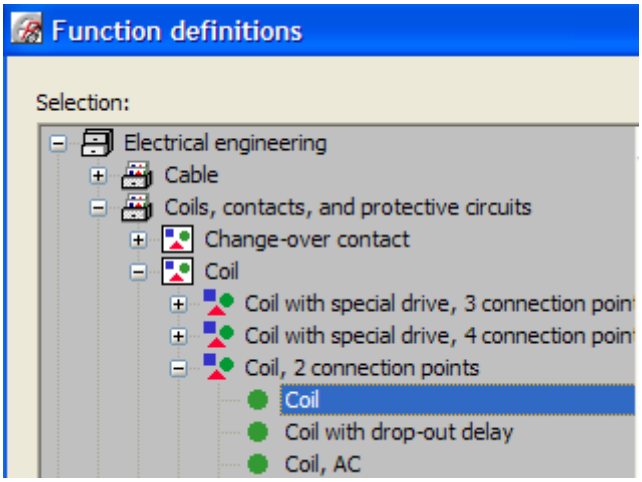
Kliknemo na gumb New, preko katerega najprej definiramo tuljavo, kasneje pa vsako kontakt posebej. Vsaki kontaktu določimo tip (delovni, mirovni, preklopni), številko korekcije. Torej začnemo z definicijo kabla, kliknemo v okence **Function definition.**

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 8.43: Lastnosti elementa

Vir: Lastni

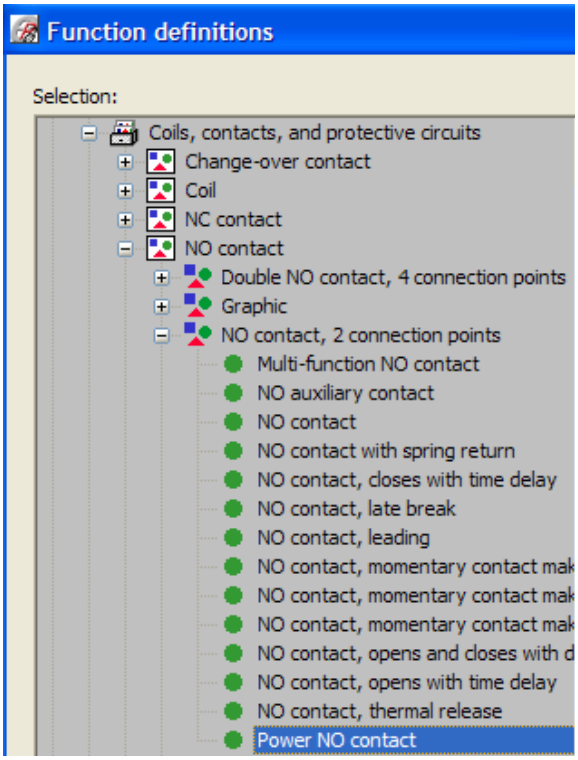


Slika 8.44: Lastnosti elelementa

Vir: Lastni

Sedaj definiramo še posamezne kontakte, preko gumba New dodajamo kontakte in jim v okencu Function definition določimo tip, v okence Connection point designation pa številko kontakta.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 8.44: Dodajanje lastnosti

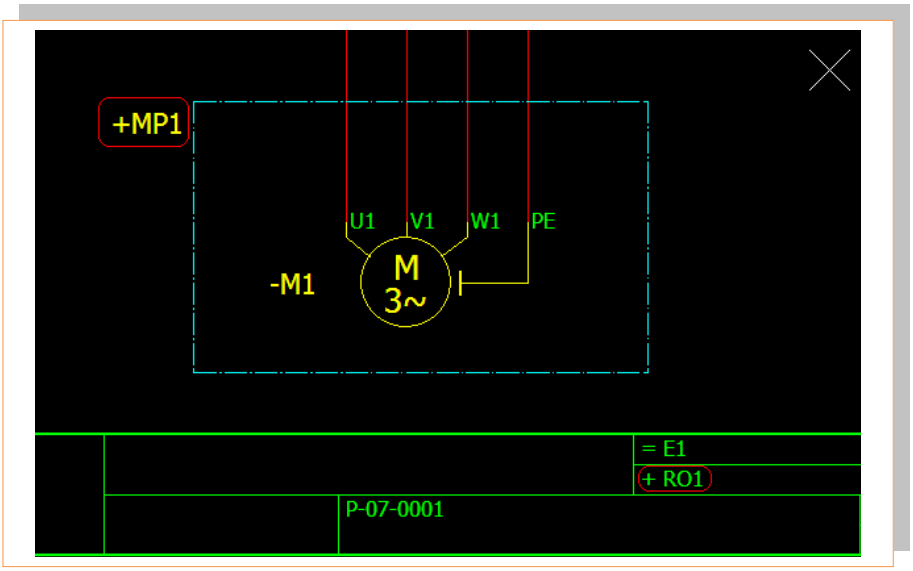
Vir: Lastni

Pomembno je še da v zavihku Function template v okence Connection point designation vpišemo kontakta tuljave, ter v zavihku Tehnical data v polje Identifier oznako releja, torej K.

Uporaba Location box –a

Location box je namenjen definiranju nove lokacije posameznim sklopom naprav ali elementom v projektu, primer je prikazan na spodnji sliki. Do ukaza za Location box dostopamo preko menija ***[insert > Box/ connection … > Location box] .***

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 8.45: Vstavljanje zgrajenih sklopov

Vir: Lastni

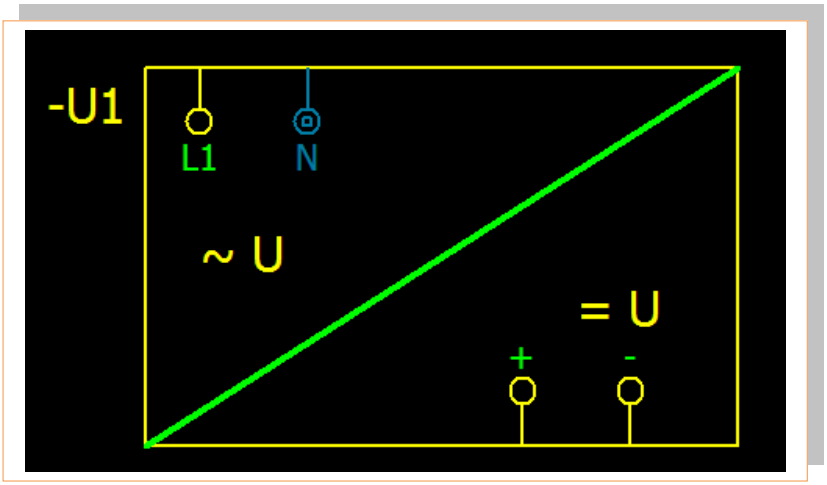
Uporaba črne škatle [Black box]

Uporabna je kadar ni pomembno kaj je znotraj določene naprave, temveč so pomembni vhodi in izhodi naprave oz. določena njegova funkcija. Do črne škatle dostopamo preko menija ***[Insert > box … > black box]***, na miškinem kazalcu se nam pojavi simbol za črno škatlo, izrišemo kvadrat primerne velikosti.

Ko narišemo black box, se nam odpre okno properties, v katerem določimo ime, izgled box-a, definiramo kataloško številko, itd. Kontake oz. vhode in izhode za črno škatlo najdemo pod ***[Insert > box … > Device connection point].*** Ime črne škatle vpišemo pod *Displayed DT*, pri kontaktih pa vpišemo ime pod *Connection point designation.*

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

Primer črne škatle:



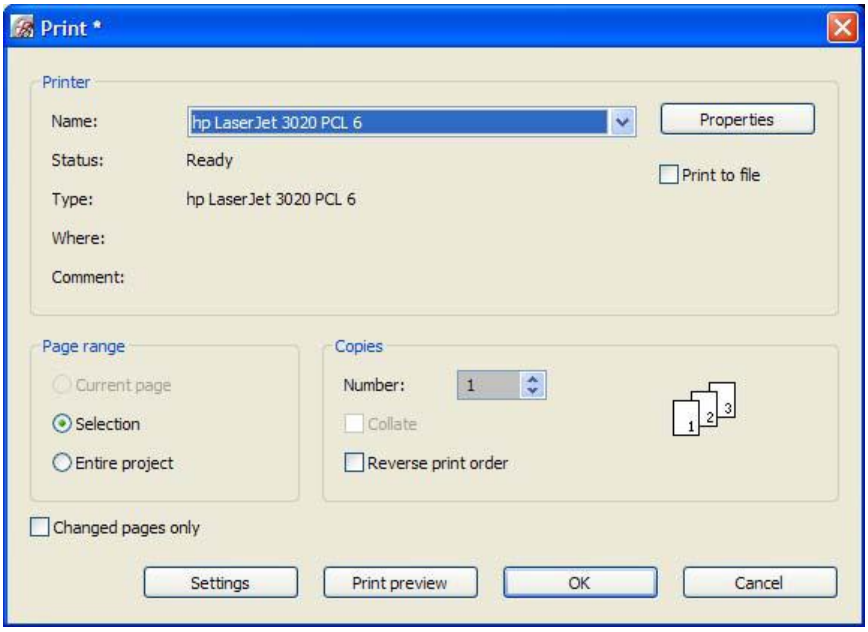
Slika 8.46: Primer črne škatle – zgrajenega sklopa

Vir: Lastni

Tiskanje projekta

V oknu *Pages* označimo projekt, ki ga želimo natisniti, če želimo natisniti celotni projekt se postavimo na glavno stran projekta, mogoče je pa tudi tiskanje posameznih strani, katere izberemo s klikom na posamezno stran in držanjem gumba [Ctrl]. Do ukaza za tiskanje dostopamo preko menija [Project > print]. Odpre se nam sledeče okno:

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 8.47: Tiskanje

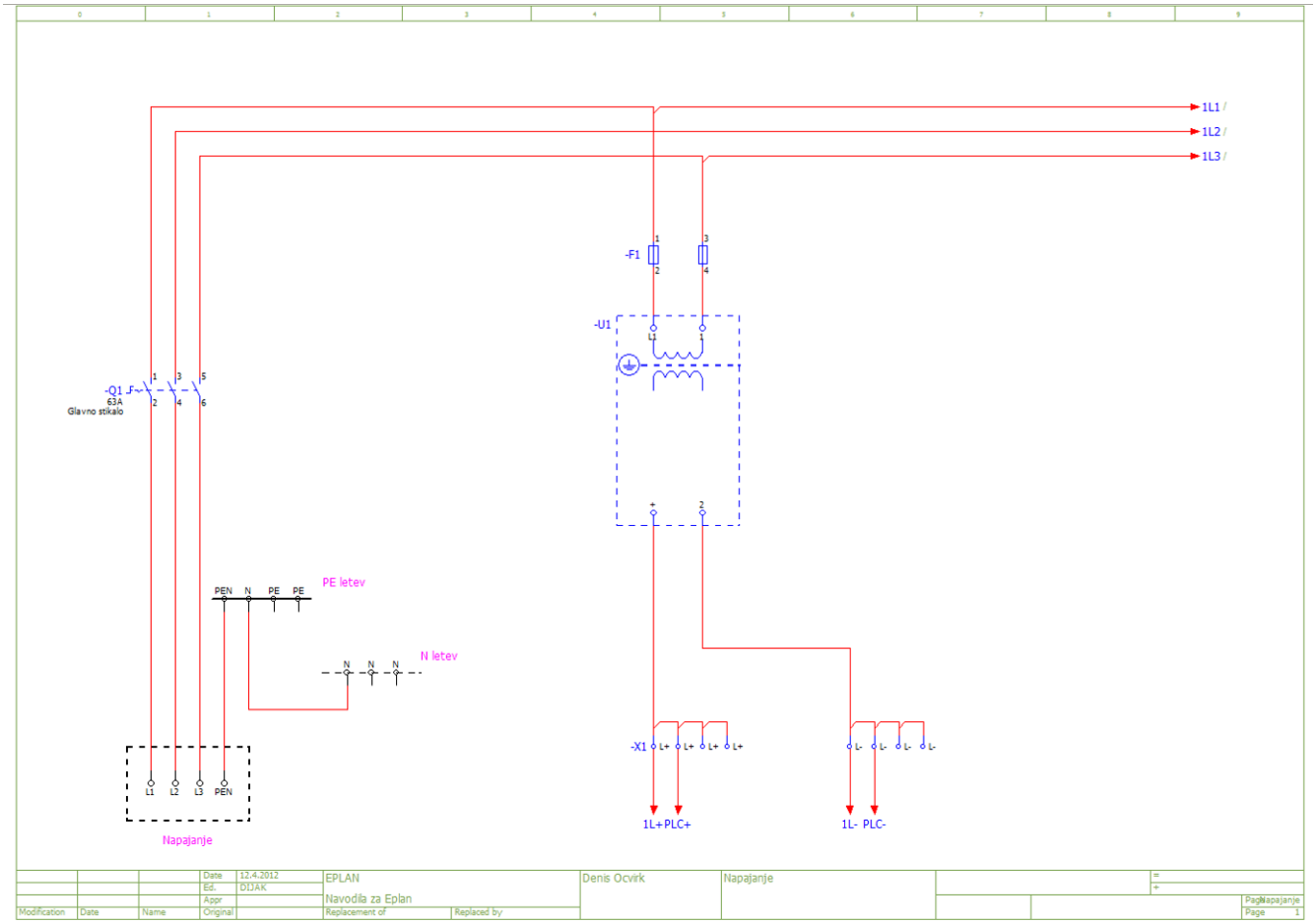
Vir: Lastni

Na voljo je možnost *Print preview*, ki nam omogoča da predno natisnemo želeni projekt, da vidimo kako bo izgledal, ter če nam ne ustreza izgled, lahko postopek prekinemo, odpravimo napako in se ponovno lotimo tiskanja.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

UČNA SITUACIJA:

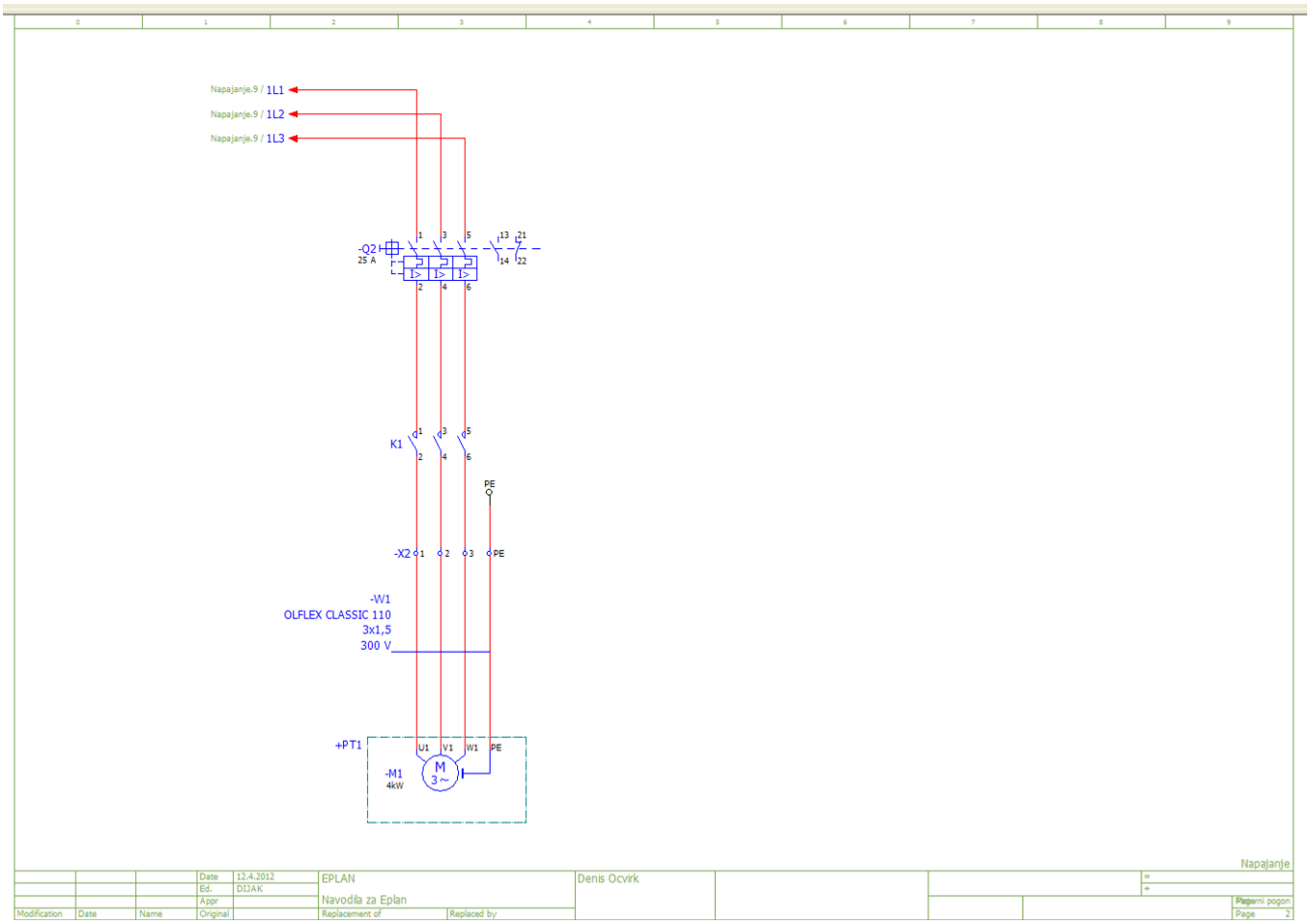
V PROGRAMSKEM ORODJU EPLAN IZRIŠITE SPODNJE EL. NAČRTE



Slika 8.48: Napajanje naprave

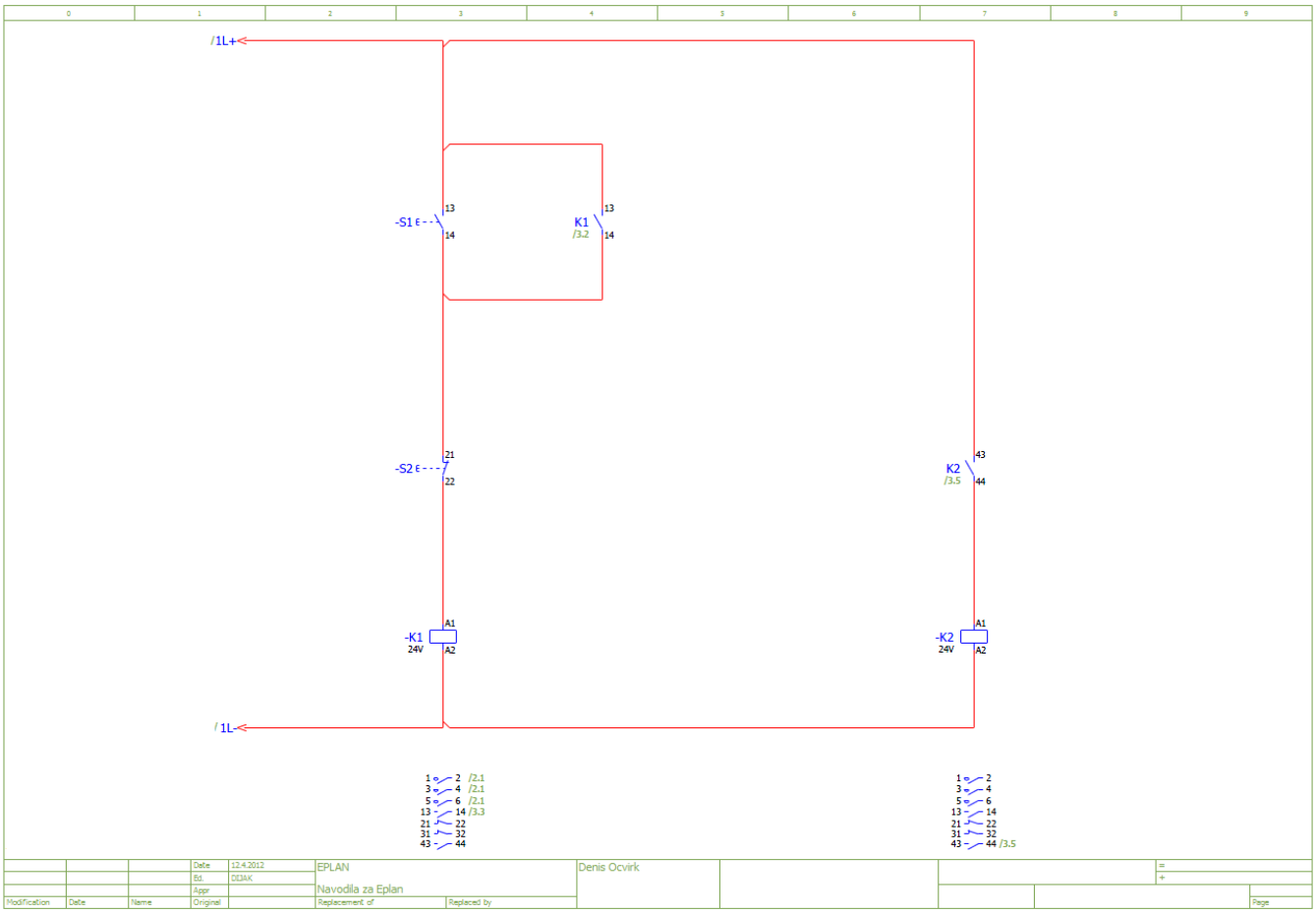
Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



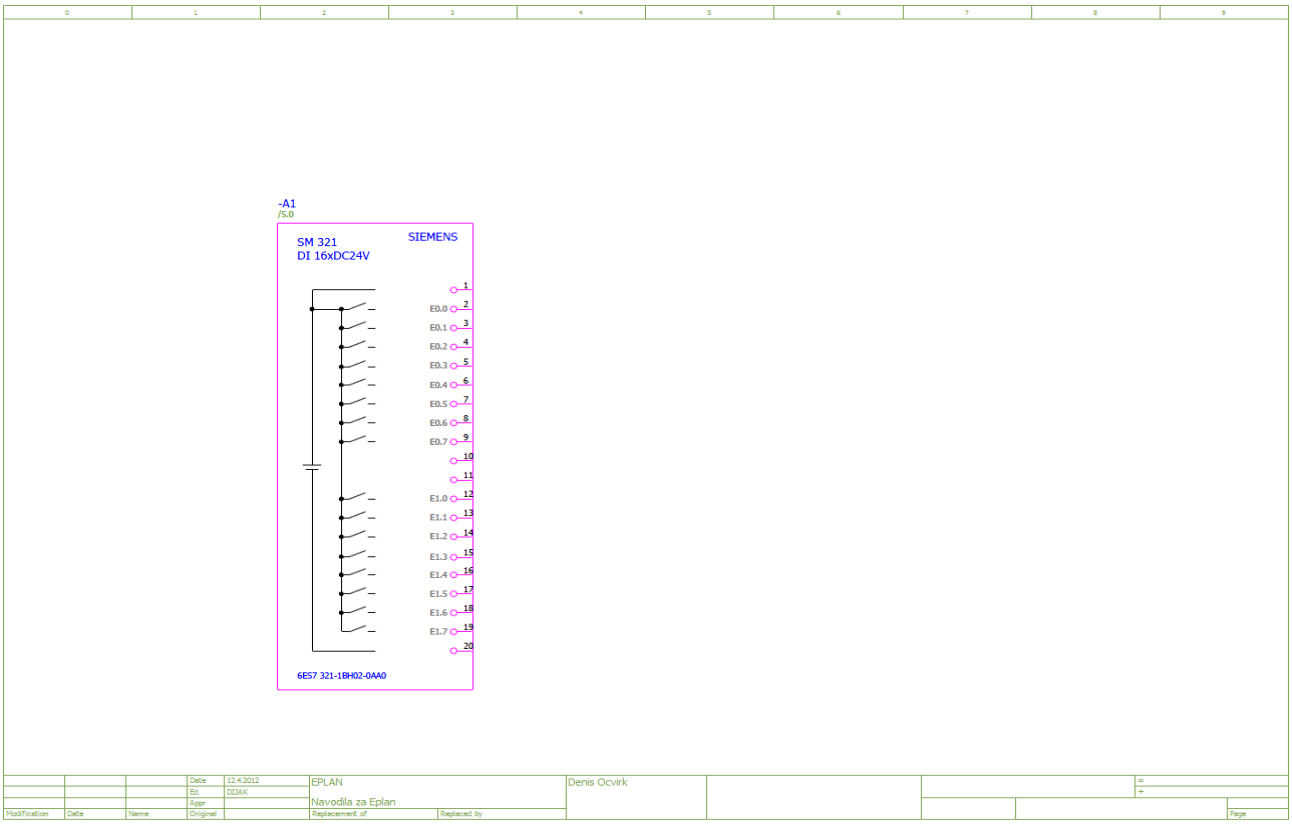
Slika 8.49: Priklop naprave

Vir: Lastni



Slika 8.50: Krmilje naprave

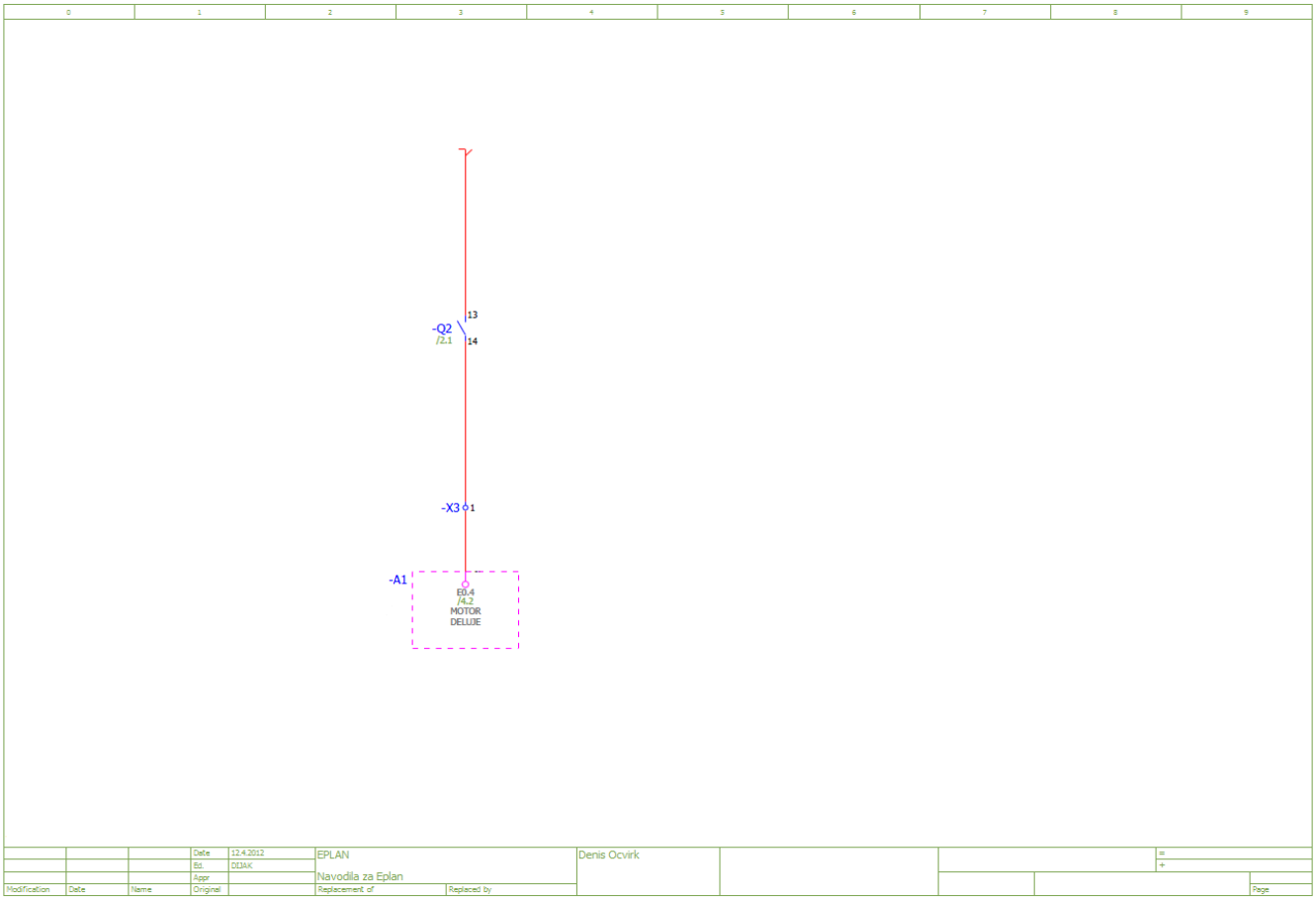
Vir: Lastni



Slika 8.52: Vezava vhodov/izhodov

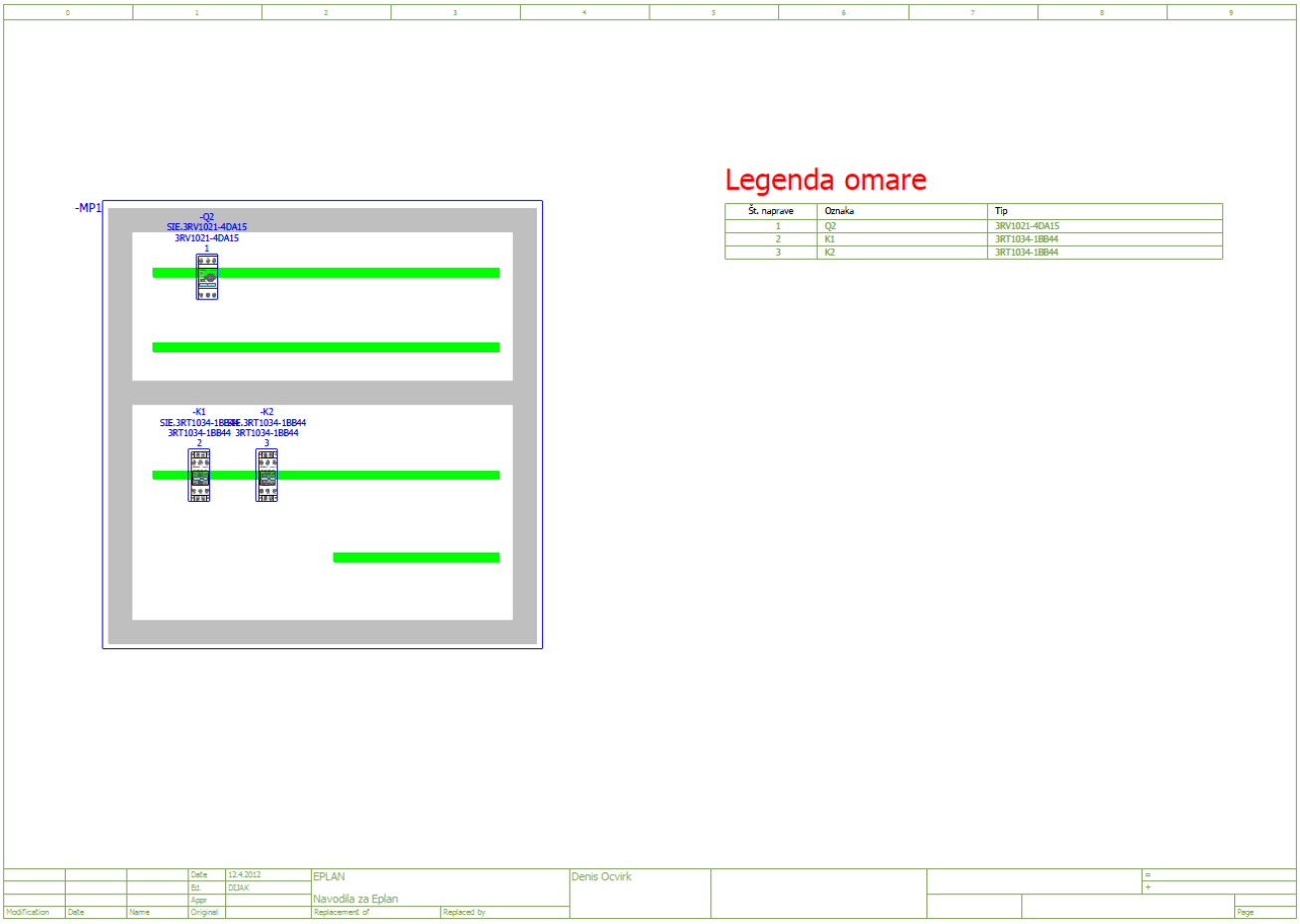
Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.



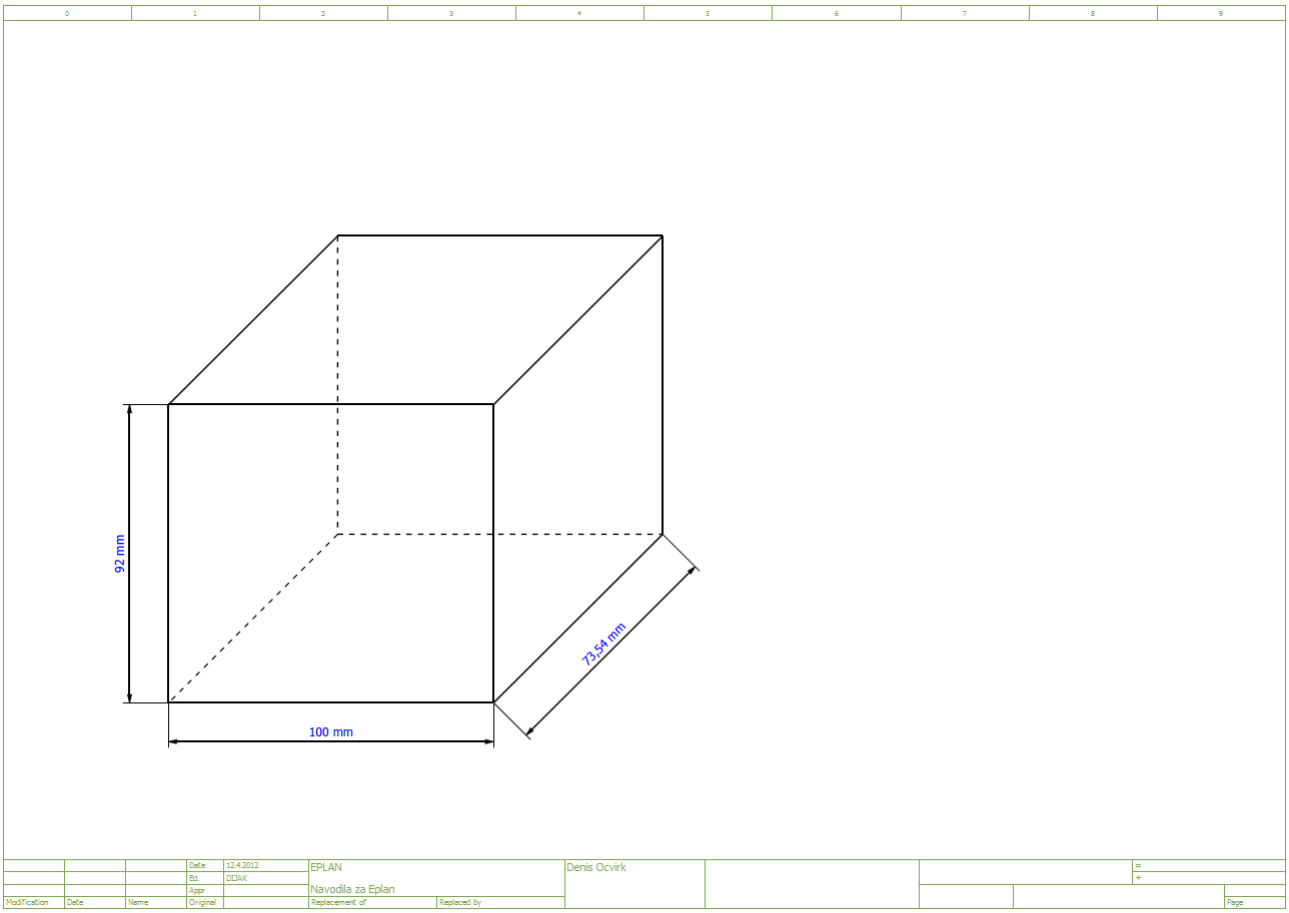
Slika 8.53: Indikacija vklopa motorja

Vir: Lastni



Slika 8.54: Postavitev elementov v krmilni omari

Vir: Lastni



Slika 8.55: Grafični izris

Vir: Lastni

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

PRIMERI DOBRE PRAKSE

http://www.youtube.com/watch?v=8wbwly5ugtg
http://www.youtube.com/watch?v=_7ewyvvb1hu
http://www.youtube.com/watch?v=kjedlglij7c
http://www.youtube.com/watch?v=62gsdb8ufhc

PRIMERI PROJEKTNEGA DELA:

- ✓ MOBILNI ROBOT
- ✓ PALETIZACIJA Z ROBOTOM
- ✓ TRINADSTROPNO DVIGALO
- ✓ REGALNO SKLADIŠČE
- ✓ ZAPORNICA NA PARKIRIŠČU
- ✓ AVTOMATIZACIJA CNC REZKALNEGA STROJA (LINUX CNC, EMC2)
- ✓ SPLETNO VODENJE INDUSTRIJSKIH APLIKACIJ
- ✓ LOČEVALNIK PLASTENK IN PLOČEVINK
- ✓ AVTOMATIZIRANA NAPRAVA ZA MLETJE JABOLK
- ✓ LOTERIJSKI MEHANIZEM
- ✓ AVTOMATIZIRANA ŠOLSKA TABLA
- ✓ LIGHT SHOW
- ✓ AVTOMATIZIRANA STISKALNICA ZA PLOČEVINKE
- ✓ VODENJE ASINHRONSKEGA MOTORJA
- ✓ VODENJE TEKOČEGA TRAKU
- ✓ AVTOMOBILSKI RAČUNALNIK
- ✓ REGULACIJA VLAGE IN TEMPERATURE V SUŠILNICI
- ✓ …

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega sklada Evropske unije in Ministrstva za šolstvo in šport.

LITERATURA IN VIRI

Hauc A. (2007). Projektni management. Ljubljana. GV založba.

Pridobljeno 25.6.2012 iz http://www.mikavna.si/2010/04/timsko-delo-za-vecjo-uspesnost/

Pridobljeno 25.6.2012 iz http://ww1.microchip.com/downloads/en/devicedoc/51265e.pdf

Colnarič, M.(2002). Osnove digitalne tehnike v računalništvu. Maribor. UM FERI.

Colnarič M., Verber D. (2000). Mikroprocesorji. Maribor. UM FERI.
Dostopno na naslovu: www.rts.uni-mb.si/misc/materiali/mikrorac/mp.pdf.

Bartenschlager, J. (2009). Mehatronika. Založba Pasadena.

D. Novak, P. Drofenik.(2012). Navodila za delo ePlan electric P8.

Glamnik A. (2011). Programiranje v avtomatiki.
Dostopno na naslovu:
http://www.impletum.zavod-irc.si/docs/Skriti_dokumenti/Programiranje_v_avtomatiki-Glamnik.pdf