Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Janez Škrlj

# Vodenje stroja za izdelovanje delov lesene palete

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Uroš Lotrič

Ljubljana, 2018

*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Paradigma Tovarna 4.0 želi v veliki meri izkoristiti možnosti modernih računalniških sistemov pri izboljšavi proizvodnih procesov. Eden od pomembnih konceptov paradigme je uporaba digitalnih dvojčkov fizičnih sistemov, ki močno pohitrijo in pocenijo razvoj. V skladu z modernimi koncepti napišite program za vodenje stroja za izdelovanje delov lesenih palet. Pri razvoju programa si pomagajte s 3D simulatorjem, ki ga izdelajte s prosto dostopnimi orodji. Primerjajte delovanje simulatorja in fizičnega stroja.

*Iskrena zahvala gre mentorju izr. prof. dr. Urošu Lotriču za vso strokovno pomoč in napotke pri izdelavi diplomske naloge. Zahvalil bi se tudi družini, ki mi vedno stoji ob strani.*

# Kazalo

# Seznam uporabljenih kratic

| kratica | angleško | slovensko |
|---|---|---|
| **JSON** | JavaScript object notation | zapis objektov JavaScript |
| **SCADA** | supervisory control and data acquisition | centralni nadzorni sistem in zajemanje podatkov |
| **CAD** | computer-aided design | računalniško podprto oblikovanje |
| **TCP** | transmission control protocol | protokol za nadzor prenosa |
| **IP** | internet protocol | internetni protokol |
| **PLC** | programmable logic controller | programirljiv logični krmilnik |
| **MQTT** | message queuing telemetry transport | protokol za prenos in veriženje sporočil telemetrije |
| **OPC** | open platform communications | zbirka standardov za komunikacijo industrijskih naprav |
| **OPC DA** | open platform communications data access | zbirka standardov OPC za zajemanje podatkov |
| **OPC UA** | open platform communications unified architecture | neodvisna različica zbirke standardov OPC |

# Povzetek

**Naslov:** Vodenje stroja za izdelovanje delov lesene palete

**Avtor:** Janez Škrlj

Stroj, opisan v diplomskem delu, iz ene lesene deske in treh lesenih kock brez posredovanja operaterja naredi sestavni del lesene palete. V okviru diplomskega dela smo izdelali program za avtomatsko vodenje stroja. Ker smo želeli, da bi zagon potekal hitreje in z manj težavami, smo najprej razvili 3D simulator stroja. 3D simulator smo razvili z uporabo igralnega pogona Unity in odprtokodne knjižnice EasyModbus. 3D simulator je bil uporabljen za razvoj programa za industrijski krmilnik Mitsubishi z oznako FX5U, ki upravlja s strojem. Napisan program smo sproti poganjali in testirali na simulatorju. S programom, razvitim na simulatorju, smo precej pohitrili zagon stroja.

**Ključne besede:** simulacija, digitalni dvojček, tovarna 4.0, avtomatizacija, proizvodnja.

# Abstract

**Title:** Control of pallet assembly machine

**Author:** Janez Škrlj

The machine described in this diploma thesis produces wooden pallet assembly parts consisting of one wooden plank and three wooden cubes. In the thesis we describe a control program for PLC to support automatic assembly of pallet parts. We also present a 3D simulator of the machine, which was prepared to reduce the number of possible problems related to commissioning and reducing the time of PLC program development. 3D simulator uses Unity game engine and EasyModbus library. The simulator was extensively used for testing and debugging during the development of program for Mitsubishi FX5U industrial PLC. We believe this approach considerably shortened the PLC programming as well as commissioning time.

**Keywords:** simulation, digital twin, factory 4.0, automation, manufacture.

# Poglavje 1

# Uvod

Podjetja vedno stremijo k zmanjševanju proizvodnih stroškov. Pomemben element predstavlja optimizacija proizvodnih procesov. V zadnjem času zato v poslovne procese vključujejo moderne pristope in tehnologije, ki so se v glavnem razvile na spletu, na primer oblaki, zbiranje in analiza velikega števila podatkov, simulacije, nadgrajena resničnost, omrežna varnost, internet stvari, horizontalna in vertikalna sistemska integracija, samostojni roboti, proizvodnja z dodajanjem [25]. Vse te tehnologije so združene pod terminom Tovarna 4.0. Horizontalna in vertikalna sistemska integracija pomeni čedalje močnejše povezovanje poslovnih sistemov s proizvodnimi. Industrijski internet stvari omogoča dostop do vse več vgrajenih naprav, ki so povezane z uporabo standardnih omrežnih tehnologij. Z zanesljivim nadzorom dostopa do teh naprav se zmanjšajo tudi tveganja, povezana z omrežno varnostjo. Hranjenje velikega števila podatkov se dogaja v oblaku, do katerega imajo dostop storitve proizvodnega sistema. Analiza teh podatkov omogoča sprejemanje odločitev v realnem času, optimiziranje proizvodne kvalitete in energijske prihranke. Proizvodnja z dodajanjem omogoča izdelavo večjega števila manjših saržnih proizvodov, ki so prilagojeni potrebam posamezne stranke. Sistemi nadgrajene resničnosti zagotavljajo podporo pri navodilih za popravilo posameznega stroja in njegovih delov. Roboti so v industriji 4.0 samostojni in prilagodljivi. Varno sodelujejo eden z drugim, kakor tudi z

ljudmi in se od nijh učijo. Simulacije prezrcalijo fizični svet v virtualni model, ki lahko vsebuje stroje, proizvode in ljudi. To omogoča operaterjem in razvijalcem, da testirajo in optimizirajo naprave v virtualnem svetu in s tem povečajo kvaliteto proizvoda brez prekinjanja proizvodnega cikla. Simulacije so temelj širše ideje digitalnega dvojčka.

Idejo digitalnega dvojčka je prvič definiral Dr. Michael Grieves iz univerze Michigan že leta 2002. Opisal ga je kot virtualno predstavitev tistega, kar se proizvaja, v kontekstu nadzora življenjskega cikla izdelka [16]. Glavna ideja je bila primerjava digitalnega dvojčka izdelka z dejanskim izdelkom in s tem zmanjševanje razlik med načrtom izdelka in dejanskim izdelkom. V najbolj optimalnem primeru bi po tej definiciji morali vsako informacijo, ki jo lahko dobimo iz fizičnega izdelka, dobiti tudi iz digitalnega dvojčka.

V članku podjetja Oracle so opisane naslednje prednosti koncepta digitalnega dvojčka [22]. Prva je nazornost, saj digitalni dvojček omogoča vpogled v delovanje stroja, pa tudi v vse ostale povezane sisteme proizvodnje. Z uporabo digitalnega dvojčka in različnih tehnik modeliranja na osnovi matematike in fizike lahko tudi predvidimo stanje stroja ali sistema v prihodnosti. Analiza kaj-če omogoča, da na modelu simuliramo različne pogoje delovanja, ki bi jih v realnosti težko izvedli. Digitalnega dvojčka lahko uporabimo tudi kot pomoč pri dokumentaciji, kjer pomaga razložiti in razumeti delovanje stroja ali sistema. Pravilno oblikovanega digitalnega dvojčka lahko povežemo z zaledjem poslovne aplikacije za predvidevanje poslovnih rezultatov dobavne verige, ki vključuje proizvodnjo, skladiščenje, transport, logistiko in popravila.

V tej diplomski nalogi bomo razvili 3D simulacijo stroja za izdelovanje sestavnih delov lesene palete, ki bo vključevala nekaj funkcionalnosti digitalnega dvojčka. Razvito simulacijo bomo uporabljali pri razvoju programa za industrijski krmilnik, ki bo upravljal s strojem. S pomočjo simulacije bomo testirali možna stanja sistema in skušali zmanjšati število nepredvidenih nezaželenih stanj. Na trgu obstaja več izdelkov namenjenih simulacijam.

Simulator FactoryI/O vsebuje 20 scen, v katerih najdemo različne indu-

strijske aplikacije, vse od preprostih tekočih trakov do naprednejših ločevalnih postaj in proizvodnih linij. Ta simulator ima na voljo 80 različnih delov industrijskih naprav, ki omogočajo da sestavimo svojo industrijsko aplikacijo z vsemi senzorji in izvršnimi členi. Izvršni členi in senzorji so lahko digitalni ali pa analogni. Sestavljeno industrijsko aplikacijo lahko pogledamo v treh različnih pogledih. Prvi je prvoosebni, v katerem se s smernimi puščicami sprehajamo po tleh prostora. Drugi je leteč, v katerem prosto letimo po prostoru. Tretji pa je krožni, v katerem je pogled osredotočen v določeno točko, z miško pa krožimo okoli te točke. Simulator FactoryI/O preko vgrajenih gonilnikov preslika statuse izvršnih členov in senzorjev stroja neposredno na dejanski industrijski krmilnik. Podprti so krmilniki proizvajalcev Allen-Bradley in Siemens. Za komunikacijo lahko uporabimo protokol Modbus ali pa OPC DA. Obstaja pa tudi možnost uporabe vhodno-izhodnega USB modula, na katerega povežemo vhode in izhode dejanskega krmilnika. Z uporabo prikazovanja 3D tekstovnih značk lahko označimo posamezne dele stroja ali pa senzorje tega simulatorja. Z uporabo teh značk in možnosti izvajanja industrijske aplikacije v upočasnjenem načinu nam omogoča lažji razvoj ter razhroščevanje programa za industrijski krmilnik. Ta simulator vsebuje tudi svojo razvojno programsko zbirko, s katero lahko razvijemo svoje vhodno izhodne gonilnike in druge razširitve [13, 12].

Simulator Machines Simulator vsebuje več kot 10 scen, ki vsebujejo različne industrijske aplikacije, najdemo pa tudi scene, v katerih se simulira delovanje prometne signalizacije. Za sestavljanje industrijskih aplikacij tega simulatorja moramo uporabiti program Machines Simulator Editor. V njem najdemo že narejene dele različnih strojev. Njihovo delovanje si prilagodimo z uporabo skript v jeziku C#. Prilagojene dele lahko potem uporabimo v simulaciji industrijske aplikacije v programu Machines Simulator. Po prostoru simulacije se premikamo v letečem načinu. Ob zakupu licence tega simulatorja dobimo tudi možnost uporabe gonilnikov, ki omogočajo neposredno povezovanje dejanskega krmilnika in simulatorja. Podprti so krmilniki proizvajalca Siemens in Beckhoff. Lahko pa uporabimo gonilnik OPC ali pa

vhodno izhodni USB modul, na katerega povežemo vhode in izhode dejanskega krmilnika ne glede na proizvajalca. Tudi ta simulator vsebuje svojo razvojno programsko zbirko, s katero lahko razvijemo svoje vhodno izhodne gonilnike in druge razširitve [20, 21].

Simulator PLCLogix 3Dworld je namenjen podpori pri učenju programiranja krmilnikov Logix 5000 in Logix 500 proizvajalca Rockwell. Vsebuje 10 različnih scen, ki vsebujejo tekoče trakove, kompresorje, prometno signalizacijo, dvigala in druge industrijske naprave. V tem simulatorju ne moremo sestavljati svojih industrijskih aplikacij. Pogled na simulacijo je določen s petimi kamerami. Ta simulator je od treh opisanih najbolj omejen, kar se tiče prilagodljivosti simulacije in povezovanja na krmilnike, ker je osredotočen zgolj na krmilnike proizvajalca Rockwell [19].

Vsi opisani simulatorji so predragi in prekompleksni za manjše stroje. Zaradi tega bomo razvili svojega z uporabo igralnega pogona Unity. V 3D simulator, ki ga bomo razvijali v okviru diplomske naloge, bomo vključili nekaj funkcionalnosti, ki jih prinaša uporaba digitalnega dvojčka. Nazornost delovanja stroja bomo zagotovili s 3D modelom stroja in uporabniškim vmesnikom, iz katerega bomo lahko razbrali statuse senzorjev in izvršnih členov stroja, kar bo koristilo novim operaterjem, ki se bodo seznanjali z delovanjem stroja. Statusa stroja v prihodnosti ne bomo predvidevali z uporabo različnih tehnik modeliranja, pač pa bomo z analizo kaj-če simulirali različne pogoje delovanja, kar nam bo prišlo prav pri testiranju programa razvitega za krmilnik. Z uporabo knjižnice EasyModbus bomo razvili vmesnik, ki omogoča komunikacijo med simulatorjem krmilnika in 3D simulatorjem stroja. Preko tega vmesnika se bodo lahko na 3D simulacijo povezale tudi različne zaledne poslovne aplikacije, s čimer bomo omogočili še eno funkcionalnost digitalnega dvojčka. Razvito 3D simulacijo se bo uporabljalo tudi kot del sistema SCADA, saj se preko prej omenjenega vmesnika lahko poveže na resnični krmilnik.

V drugem poglavju se bomo seznanili z delovanjem stroja za izdelovanje sestavnih delov lesene palete. V tretjem poglavju se bomo spoznali z orodji,

ki jih bomo uporabljali pri razvoju 3D simulatorja. S potekom razvoja 3D simulatorja se bomo seznanili v četrtem poglavju. V petem poglavju se bomo lotili razvoja programa za industrijski krmilnik Mitsubishi FX5U, ki bo upravljal s fizičnim strojem. Pravilnost delovanja razvitega programa bomo v šestem poglavju najprej preverili na 3D simulatorju, nato pa bomo program zagnali še na fizičnem sistemu. Primerjavo teh dveh sistemov bomo naredili v sedmem poglavju. V zaključku bomo povzeli in ocenili rezultate našega dela.

# Poglavje 2

# Predstavitev stroja

## 2.1 Izdelki stroja

Stroj izdeluje dele lesene palete. Gre za deske, na katere so pribite tri lesene kocke, ki predstavljajo enega od treh spodnjih nosilnih vzdolžnih ali pa prečnih delov lesene palete (slika 2.1). Dolžina in širina deske, na katero so pribite lesene kocke, sta tisti dve lastnosti, ki določata, katerega od teh dveh delov palete bo izdelek stroja predstavljal. Za nadaljnji postopek izdelave ene palete so potrebni trije izdelki tega stroja, to se pravi tri posamične deske s pribitimi kockami. Stroj lahko v eni uri izdela 400 spodnjih nosilih delov palete.

Slika 2.1: Možna izdelka stroja.

## 2.2 Začetek delovanja

Pred zagonom je potrebno, da je stroj priključen v električno omrežje. Stroj za svoje delovanje potrebuje stisnjeni zrak, zato mora biti prek hitre spojke priključen tudi na vir stisnjenega zraka.

Temu sledi vklop glavnega stikala na kontrolni omarici. Za tem je potrebno izvleči vse varnostne stop tipke. Ko so varnostne tipke v pravem položaju, lahko s tipko na kontrolni omarici vklopimo krmilno napetost. Krmilna napetost aktivira krmilne tokokroge in napajanje (napetost +24V) ključnih komponent, kot sta krmilnik in zaslon. Aktivacija teh se potrdi s kratkim zvočnim signalom. Po tem koraku krmilnik že zaznava vhodne statuse vseh senzorjev na stroju. Statusi izhodov krmilnika v tej fazi še nimajo nikakršnega učinka na izhodne naprave in izvršne člene stroja, ker še nimajo napetosti. Napetost jim zagotovimo s pritiskom na tipko z napisom vklop varnosti, ki aktivira varnostni tokokrog in pripelje napetost tudi izhodnim napravam (kontaktorjem, ventilom, relejem), seveda v odvisnosti od izhodnih statusov krmilnika.

Za pričetek delovanja stroja v avtomatskem načinu na kontrolni omarici pritisnemo tipko start cikla. Če so izpolnjeni vsi inicialni pogoji (napolnjeni zalogovniki z obdelovanci in napolnjeni zalogovniki pištol z žeblji), bo stroj pričel z delovanjem. V nasprotnem primeru pa se bo na zaslonu izpisalo opozorilo, ki pove, kateri inicialni pogoj ni izpolnjen, na primer zalogovnik št. 2 prazen.

## 2.3 Zalogovniki

Stroj ima štiri vhodne zalogovnike v katere se vlaga lesene kocke in deske, ki predstavljajo obdelovance. Obdelovanci so z uporabo pnevmatskih pištol za zabijanje povezani z žeblji iz zalogovnikov žebljev, ki se nahajajo na vseh treh pištolah in morajo biti ravno tako napolnjeni. Polni zalogovniki omogočijo, da stroj sploh lahko začne z avtomatskim ciklom in ob normalnem delovanju zagotavljajo nemoteno proženje pištol za zabijanje in tekoče doziranje

posameznih obdelovancev.

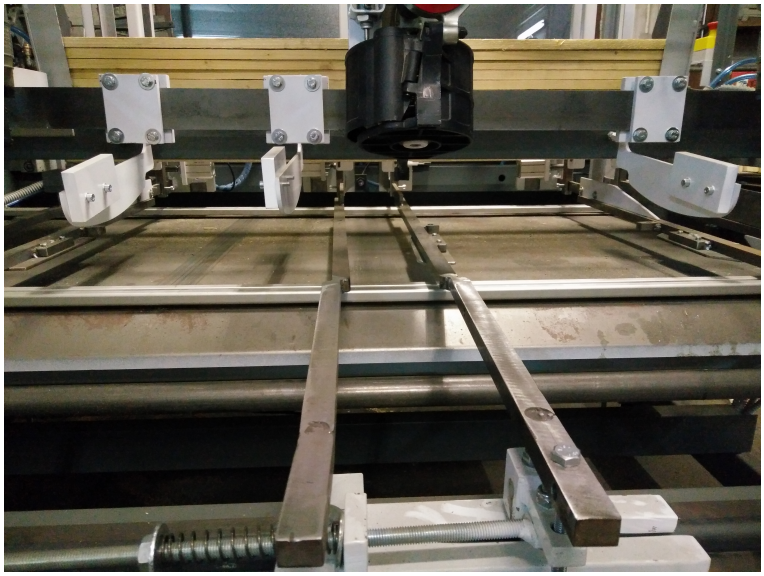
### 2.3.1 Zalogovniki za kocke

Naloga treh identičnih zalogovnikov je hranjenje in doziranje lesenih kock. Lesene kocke lahko merijo vse od 5 cm do 12 cm v dolžino, od 5 cm do 12 cm v širino in od 5 cm do 15 cm v višino. Pomembno je predvsem, da so kocke, ki se nahajajo v vseh treh zalogovnikih, enako velike. V nasprotnem primeru se lahko zgodi, da je paleta nestabilna in se ziblje. Za kvalitetno paleto je potrebno, da je širina kocke enaka širini deske, ker bo v tem primeru deska optimalno pribita na kocke. V primeru, da je kocka premajhna, obstaja možnost, da žebelj sploh ne bo pribit v kocko.

Z vsakim posameznim zalogovnikom s kockami upravljata dva bata. Prvi bat kocke pridržuje, da ne padejo iz zalogovnika. Drugi bat pa služi stiskanju kocke, ki je druga v vrsti in hkrati pridrževanju ostalih, da ne padejo dol, medtem ko je prvi bat umaknjen in je prva kocka izvržena na mizo. Ko je kocka uspešno spuščena na mizo, prvi bat ponovno zapre spodnji del zalogovnika. Temu pa sledi popuščanje drugega bata, ki spusti kocko, ki je bila prej druga v vrsti, da pade na prvi bat in tako postane prva za izmet v novem ciklu. Za drugo kocko padejo tudi vse ostale kocke, ki se ravno tako premaknejo eno mesto naprej v vrsti. Na stroju so trije taki zalogovniki. Večina lesenih embalažnih palet ima tri podložne kocke. Na tem stroju pa je možno izdelovati tudi spodnje nosilne dele palet s samo dvema kockama. Ob normalnem delovanju stroja se doziranje kock iz vseh treh prej opisanih zalogovnikov zgodi istočasno. Razmik med zalogovniki s kockami je ročno nastavljiv in se ga lahko prilagodi glede na dolžino deske. Vsi trije zalogovniki so pri deskah dolžine 50 cm in kockah širine 10 cm razmaknjeni za 10 cm eden od drugega, pri deskah dolžine 120 cm pa ti razmiki merijo 45 cm.

### 2.3.2 Zalogovnik za deske

Četrti zalogovnik zagotavlja hrambo in pridrževanje lesenih desk, ki vzdolžno povezujejo kocke. Zalogovnik z deskami je namenjen hrambi in pridrževanju desk, ki so lahko široke od 5 do 20 cm in dolge od 50 do 120 cm. Lesene deske so tako kot kocke zložene ena na drugi. Doziranje desk pa ravno tako poteka z uporabo dveh batov. Najprej bat na levi strani desko s strani vodoravno potisne na desno stran, da deska izgubi podporo s spodnje strani in ni več naslonjena na stransko polico in tako leva stran deske pade na dvignjena vodila kock obdelovalne mize (slika 2.2). Za tem bat z desne strani vodoravno potisne desko na levo stran, tako, da tudi desni del deske pade na vodila za kocke, ki desko pridržujejo, da se nahaja nad obdelovalno mizo. Razdalja med desko in mizo je malo več, kot je visoka lesena kocka. Ob potisku desnega bata se deska tudi potisne ob železno letev na levi strani, kar ji zagotavlja poravnavo, da ni zamaknjena postrani.



Slika 2.2: Vodila kock, na katerih sloni tudi deska.

## 2.4 Sojemala in premik obdelovancev

Po doziranju lesenih kock in desk se na mizi nahajajo tri lesene kocke in deska, ki je nekoliko višje kot je obdelovalna miza in je naslonjena na vodilih za kocke. V takem stanju je stroj pripravljen na naslednji korak.

V avtomatskem načinu delovanja stroja se prečna sojemala, gnana z elektromotorjem, premikajo po obdelovalni mizi tako, da najprej začnejo potiskati tri izvržene lesene kocke naprej proti pištolam. Na posameznem sojemalu sta nameščena tudi dva železna stebrička, ki sta postavljena pokončno. Ta dva stebrička pa začneta potiskati tudi desko, ki se nahaja na vodilih nad kockami, ko pride sojemalo do nje. Zalogovniki izvržejo nove kocke in desko potem, ko sojemalo opravi dovolj dolgo pot in odrine kocke ter desko dovolj daleč, da se pod zalogovniki naredi prostor za naslednje kocke in desko.

## 2.5 Pištole in zbijanje

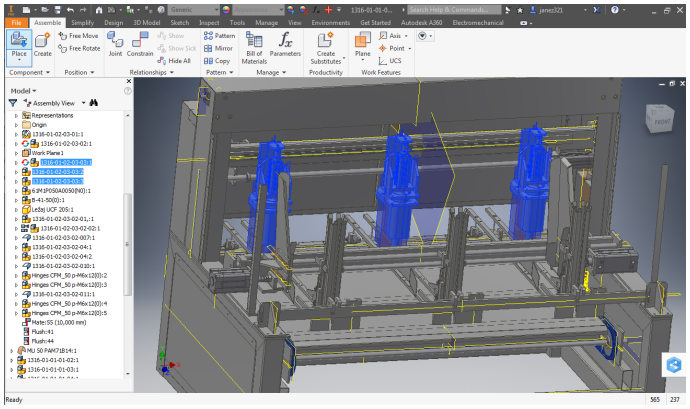
Stroj za povezovanje lesenih kock z deskami uporablja tri pnevmatske pištole, ki zabijejo žeblje skozi desko v kocko z uporabo zraka pod pritiskom. Vsaka posamezna pištola je nameščena tako, da jo v pozicijo zabijanja, to je dol, potisne posamezen pnevmatski bat. Vse tri pištole se nahajajo na eni gredi, ki se prečno pomika naprej in nazaj. Ta gred za premikanje uporablja bat, ki ravno tako kot pištole za svoje delovanje potrebuje zrak pod pritiskom. Premikanje gredi pištol omogoča, da sta žeblja pribita na dveh različnih pozicijah. V sodelovanju s premikanjem sojemala poskrbi, da sta žeblja diagonalno zamaknjena. Najprej je gred v skrčeni poziciji, ko se pištole spustijo dol in je pribit prvi žebelj. Nato se pištole dvignejo in gred, na katero so nameščene, se premakne v iztegnjeno pozicijo. Obenem pa se vključi tudi premik sojemala, ki sojemalo premakne za malo manj kot je širina deske naprej. Nato se pnevmatske pištole ponovno spustijo dol in zabijejo še drugi žebelj v desko in kocke. Za konec se pištole dvignejo. Gred, ki povezuje pištole, pa se premakne nazaj v začetni položaj. V takem položaju so pištole pripravljene na naslednjo desko in kocke.

# Poglavje 3

# Predstavitev orodij

## 3.1 Autodesk Inventor

Autodesk Inventor je program CAD, namenjen načrtovanju in oblikovanju tehnično podrobnih ter dovršenih 3D modelov proizvodov, ki se uporabljajo v industriji [18]. Z njim je lahko izdelana vsa tehnična dokumentacija fizičnih delov izdelka. Najnovejša verzija tega programa se imenuje Inventor® 2019. V njej so predstavili nove načine za izdelovanje lukenj na modelih, dodajanje komponent in omejitev preko skript, integrirano deljenje modela projekta za namene kooperacije in predstavitve produkta.



Slika 3.1: Označene pištole, pripravljene za izvoz.

Iz že pripravljenega tehničnega 3D modela stroja (slika 3.1) smo izvozili posamezne 3D modele sestavnih delov stroja kot več različnih datotek tipa .obj, ki smo jih kasneje obdelali v drugih programih.

## 3.2 Blender

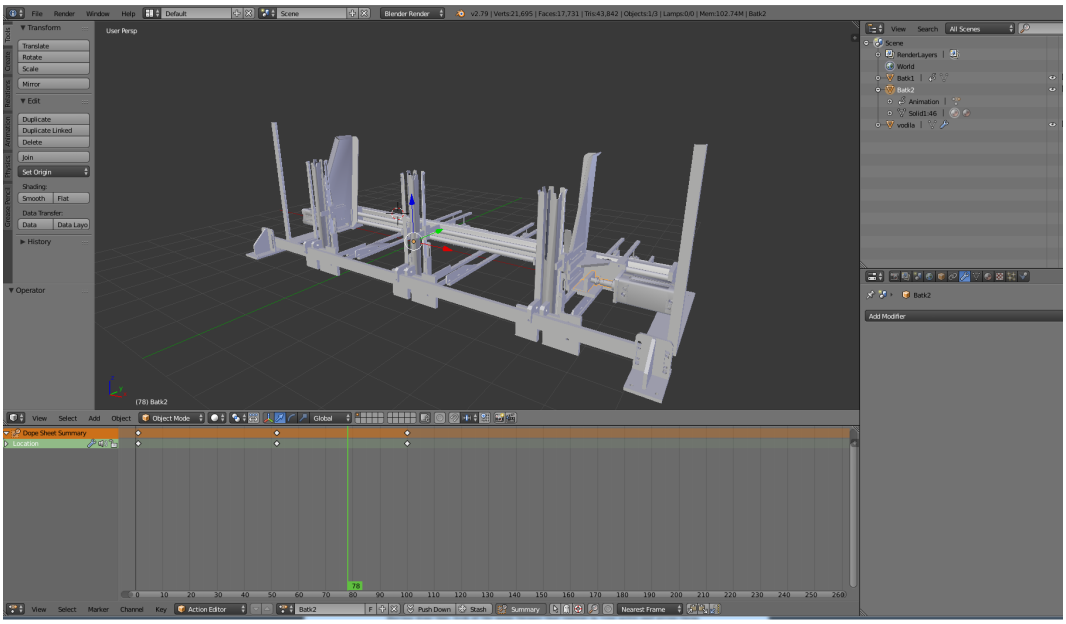
Je brezplačno in odprtokodno programsko orodje, namenjeno 3D oblikovanju. Podpira modeliranje, animacije, simulacije, izrisovanja, sledenja, urejanje videa in ustvarjanje iger. Ima svoj izrisovalni pogon, ki lahko uporabi procesor ali pa grafično kartico za izrisovanje. Ravno tako kot Unity ima Blender veliko skupnost, zato zanj obstaja tudi veliko število vtičnikov, ki omogočajo vse od dodatnih nastavitev izrisovanja svetlobe in osvetlitve pa do zunanjih izrisovalnih pogonov. Ker ima Blender tudi možnost uporabe skript po meri, se za avtomatizacijo opravil lahko v programskem jeziku Python napiše programe oziroma skripte, ki modeliranje pohitrijo. Python je ravno tako tudi jezik igralnega pogona, ki je vgrajen v Blender. Ta igralni pogon se največkrat uporablja za arhitekturno vizualizacijo, se pa z njim da izdelati tudi samostojne aplikacije [10, 32].

Celoten 3D model stroja je bil oblikovan v programu Autodesk Inventor. Ta program uporabljajo strojniki za načrtovanje, oblikovanje in dokumentiranje projektov, sestavljenih iz veliko različnih komponent. Datoteka tako oblikovanega modela zavzema veliko prostora na disku. Tak model ni optimiziran za uporabo v kakšnem igralnemu pogonu in tudi ne sme biti, saj je njegov namen, da se iz njega naredi natančne načrte delov stroja in vso ostalo dokumentacijo. Na tem mestu pa je nastopil Blender, ki je sposoben uvoza velikega števila različnih tipov datotek 3D modelov in njihove obdelave v 3D modele, ki so bolj optimizirani in prijaznejši igralnim pogonom. Določene objekte uvoženega modela programa Autodesk Inventor smo v programu Blender odstranili. Med njimi so se znašli vsi vijaki, vsi nevidni objekti, ki so sestavni deli stroja, a na simulacijo ne vplivajo, ter statični deli stroja, ki predstavljajo zaščitno ohišje stroja in bi pri simulaciji le ovirali

vidno polje. Za dele, ki so imeli veliko število robov, je bila uporabljena funkcija decimiranje, ki zmanjša njihovo število. Z optimalno nastavitvijo parametrov, ki določajo število odstranjenih robov, smo dosegli, da model kljub manjšemu številu robov ni popačen. V Blenderju smo tudi združili več različnih objektov, ki predstavljajo posamezne dele stroja, zato da se je njihova zahtevnost zmanjšala.

### 3.2.1 Animacije

Združevanje objektov je bilo uporabljeno tudi pri nekaterih animacijah, saj so določeni deli stroja vezani na isto os. Naredili smo tudi animacije spuščanja in dvigovanja pištol za zabijanje. Najprej smo združili dele nosilca, nato pa še pištole. Za animacije spuščanja in dviganja je bilo potrebno določiti ključne točke časovnega traku animacije in pozicije delov v odvisnosti od časa. Pri posamezni pištoli za zabijanje je šlo le za spremembo pozicije po y osi. Pri animaciji iztegovanja gredi, na katero so pričvrščene vse tri pištole, pa je šlo za spremembo pozicije vseh treh pištol v smeri osi x. Na podoben način sta bila animirana tudi stranska bata, ki dozirata deske (slika 3.2). Pri njima je šlo le za spremembo pozicije osi bata in nanj nameščenega porivala v obliki črke l. Nazadnje smo naredili še animacijo kroženja sojemal. Na štiri podolgovate kvadre, ki predstavljajo posamezno sojemalno letev, smo določili omejitev, ki določa eliptično gibanje izbranih komponent. Ključne točke animacije smo nastavili tako, da vsaka letev opravi pot preko celotne elipse in konča tam, kjer začne. Tako narejena animacija je funkcionirala v programu Blender. Ko smo jo izvozili iz programa Blender in uvozili v Unity, pa so se pojavile težave, saj je animacija izgledala tako, kot da na posamezne letve modifikator ni vplival, ker so se te gibale samo v smeri x-osi in niso krožile po elipsi, tako kot v programu Blender. Zaradi teh težav smo uporabili tudi program Autodesk 3DS Max.



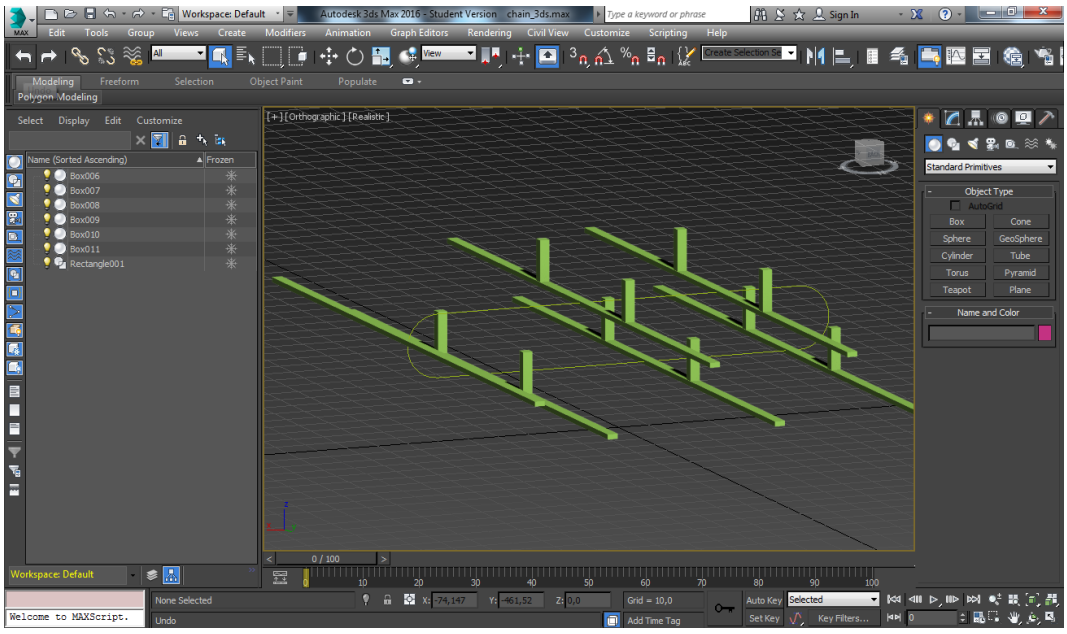
Slika 3.2: Animiranje gibanja stranskega bata v programu Blender.

## 3.3 Autodesk 3ds Max

3ds max je profesionalno orodje podjetja Autodesk, namenjeno 3D modeliranju, animaciji in izrisovanju. Najpogosteje je uporabljano pri razvoju iger in arhitekturi, saj je z njim možno poustvariti realistične scene najrazličnejših okolij. Uporabljajo pa ga tudi filmski studiji pri izdelavi efektov računalniške grafike. Eden bolj znanih filmov, za izdelavo katerega so uporabili 3ds Max, je Avatar iz leta 2012. 3ds max se dobro povezuje tudi z drugimi programi podjetja Autodesk. V najnovejši različici so bili predstavljeni novi vtičniki, simulacija tekočin in pa svoj programski jezik za pisanje senčilnikov [17, 31].

Zaradi nepravilnega delovanja animacije sojemal iz programa Blender, ki bi morala krožiti po elipsi, ki predstavlja verigo, smo to animacijo ponovno ustvarili v programu 3ds max. Najprej smo v stranskem pogledu narisali elipso, ki predstavlja pot potovanja posamezne letve sojemala. Zatem smo dodali objekt tipa škatla, ki smo mu spremenili obliko, da je izgledal kot posamezna letev. Temu objektu smo določili tudi omejitev poti. Za tarčo,

ki določa pot, pa je bila izbrana prej narisana elipsa. Po tem koraku je letev lahko že krožila po narisani elipsi. S kopiranjem prej narisane letve smo dodali še ostale letve in jim določili razmik (slika 3.3). Za animacijo smo na podoben način kot v programu Blender tudi v programu 3ds max najprej določili začetno točko na časovni osi in pozicijo letev, zatem pa še zaključno točko na časovni osi in končno pozicijo letev. Vmesne pozicije gibanja, med katerimi letve potujejo od začetne pozicije do končne, je program 3ds max izračunal sam. Za učinek cikličnega gibanja pa se je animacija le ponovila od začetka, saj je bila narejena tako, da se posamezna letev ustavi tam, kjer tudi začne z gibanjem. Animacijo smo izvozili kot datoteko tipa .fbx. Ob uvozu v program Unity je animacija delovala na pravilen način, saj so ob zagonu animacije letve potovale po prej definirani elipsi.




Slika 3.3: Animiranje sojemal v programu 3ds Max.

## 3.4 Microsoft Visual Studio

Visual studio je zbirka orodij za razvoj programske opreme. Omogoča vse od načrtovanja uporabniškega vmesnika, pisanja kode, testiranja, razhroščevanja, analize kode, do zagotavljanja podpore končnemu produktu s pomočjo telemetrije. Vsa ta orodja so integrirana v razvojno okolje, na tak način, da Visual Studio tvori integrirano razvojno okolje [2]. Z njim je možno razviti aplikacije, ki lahko tečejo na platformah Windows, Android in iOS. Omogoča tudi razvoj spletnih aplikacij in servisov, ki lahko temeljijo na tehnologiji .NET ali pa na različnih knjižnicah programskega jezika Javascript.

Microsoft Visual Studio je bil uporabljen pri razvoju 3D simulatorja stroja v programu Unity. Služil je kot urejevalnik kode vseh napisanih skript. Bil je alternativa urejevalniku MonoDevelop, ki je vgrajen v program Unity. Kot alternativa je bil izbran zaradi zanesljivejšega in bolj usklajenega delovanja na operacijskem sistemu Windows, na katerem je razvoj tudi potekal. Pri razvoju vmesnika med simulatorjem krmilnika in simulatorjem stroja pa je bil Visual Studio uporabljen kot integrirano razvojno okolje. Uporabniški vmesnik smo zasnovali na tehnologiji Windows Forms. Visual Studio je kot integrirano razvojno okolje omogočal nazorno razhroščevanje in hitro poganjanje programa.

## 3.5 Igralni pogoni

Pregledali smo najpopularnejše 3D igralne pogone in se odločili za uporabo igralnega pogona Unity [27].

### 3.5.1 Unity

Unity je eden najpogosteje uporabljenih igralnih pogonov za razvijanje 3D in 2D iger ter simulacij [29]. Njegova prednost je, da je preprost in podpira veliko različnih platform, kar pomeni, da se program lahko napiše v jeziku C# ali pa v različici jezika javascript in nato požene na različnih napravah,

ki segajo vse od igralnih konzol, računalnikov, pa do mobilnih naprav. Kot igralni pogon je zelo zmogljiv, enostaven, brezplačen za nekomercialno uporabo [28] in ima dobro definirano dokumentacijo ter veliko skupnost. Za to diplomsko nalogo je bila predvsem atraktivna zmožnost simulacije fizike [30], ki omogoča, da se objekti s prej določenimi fizikalnimi lastnostmi obnašajo kolikor toliko realno, kar pomeni, da na njih deluje gravitacija, da zaznavajo druge objekte in se tudi zaletavajo v njih, če so jim na poti. Možnost uvoza velikega števila različnih tipov datotek 3D modelov je med drugim tudi botrovala izbiri tega igralnega pogona. Predvsem je bila dobrodošla možnost uvoza datotek tipa .obj in .fbx, v katerih so zapisani 3D modeli, kakor tudi animacije, ki se dogajajo na teh 3D modelih. 3D modeli stroja, opisanega v diplomski nalogi, so bili prvotno oblikovani v programu Autodesk Inventor, nato obdelani v programih Blender in Autodesk 3ds Max ter na koncu uvoženi v Unity.

### 3.5.2 Unreal Engine

Je igralni pogon, ki ga je razvilo podjetje Epic Games. Ta igralni pogon je bil sprva uporabljen pri prvoosebnih streljačinah, nato pa se je njegova uporaba razširila še na druge žanre. Glavni programski jezik pri tem pogonu je C++. Podpira igralne konzole, računalnike in mobilne naprave. Unreal engine je osvojil tudi nagrado Guinessove knjige rekordov za najbolj uspešen igralni pogon. To nagrado si je prislužil s tem, da je bil uporabljen kot igralni pogon pri več kot 400 različnih igrah največjih studijev. Obstajajo 4 različice igralnega pogona Unreal Engine, ki si sledijo po starosti. Unreal Engine 1 je najstarejša, Unreal Engine 4 pa najnovejša. Leta 2014 je podjetje Epic Games omogočilo dostop do izvorne kode igralnega pogona Unreal Engine. Zaradi tega se za uporabo igralnega pogona Unreal Engine odloči več naprednih razvijalcev, ki svoji igri prilagodijo tudi igralni pogon. Igralni pogon Unreal Engine po zahtevnosti spada med zahtevnejše igralne pogone, kar lahko vidimo tudi po tem, da prazen projekt zavzame najmanj 200 MB prostora na disku. Zaradi svoje zahtevnosti omogoča večjo prilagodljivost,

ki jo izkoriščajo razvijalci naprednejših iger [27, 35, 11].

### 3.5.3 CryEngine

Je igralni pogon podjetja Crytek. Ravno tako kot Unreal Engine se je tudi CryEngine sprva uprabljal pri prvoosebnih streljačinah. V ospredju je bila predvsem igra Far Cry. Do danes se je CryEngine razvil, vendar pa nekateri razvijalci trdijo, da je neprimeren za razvoj nečesa, kar ni prvoosebna streljačina. Programski jeziki, ki jih CryEngine uporablja, so C++, Lua in C#. CryEngine podpira igralne konzole in računalnike. Trenutno pa še ne podpira mobilnih naprav. Obstaja več različic igralnega pogona CryEngine, ki si po starosti sledijo od CryEngine 1 pa do CryEngine V oziroma 5. Iz igralnega pogona CryEngine 1 je podjetje Ubisoft razvilo svojo prilagojeno različico igralnega pogona. Ta se je razvijala naprej in danes se imenuje Dunia Engine. CryEngine izstopa z dobrim izrisovanjem prizorišč narave, ki vsebujejo drevesa in vegetacijo. To počne hitreje in bolj optimizirano od ostalih igralnih pogonov. Dostop do izvorne kode igralnega pogona CryEngine je odvisen od zakupljene licence. Ravno tako kot Unreal Engine velja CryEngine za enega naprednejših igralnih pogonov [33, 14, 15].

## 3.6 GX Works3

GX Works3 je razvojno okolje, namenjeno razvoju programov za industrijske krmilnike proizvajalca Mitsubishi. Omogoča konfiguriranje nastavitev, razhroščevanje, izvajanje vzdrževanja in programiranje krmilnikov iz družin MELSEC iQ-R in MELSEC iQ-F. V družino MELSEC iQ-F spada tudi krmilnik FX5U, ki je bil uporabljen za upravljanje stroja, opisanega v diplomski nalogi. Program GX Works3 podpira standard IEC 61131-3. Ta standard določa osnovno programsko arhitekturo in programske jezike za programiranje industrijskih krmilnikov. Med te programske jezike spadata dva grafična in dva tekstovna jezika. V programu GX Works3 lahko grafično programiramo v obliki lestvičnih diagramov in diagramov funkcijskih blokov. Grafično

je možno tudi ustvarjati zaporedne funkcijske sheme. Kar se tiče tekstovnih jezikov, pa je v programu GX Works3 možno pisati v strukturiranem tekstu, v listi ukazov pa ne, ker velja za zastarano od tretje verzije standarda IEC 61131-3 naprej [24]. Tako napisani programi se potem lahko prenesejo na krmilnik ali pa preberejo s krmilnika. GX Works3 pa poleg običajnega pisanja in branja omogoča tudi modificiranje sekvenčnih programov med izvajanjem. Vpogled v delovanje teh in ostalih programov omogoča funkcija monitor, ki se uporablja pri razhroščevanju programov in diagnosticiranju napak. Funkcija diagnostike prikaže vse trenutne in prejšnje napake krmilnika, kar pomaga pri skrajševanju časa povrnitve krmilnika v delujoče stanje.[5].

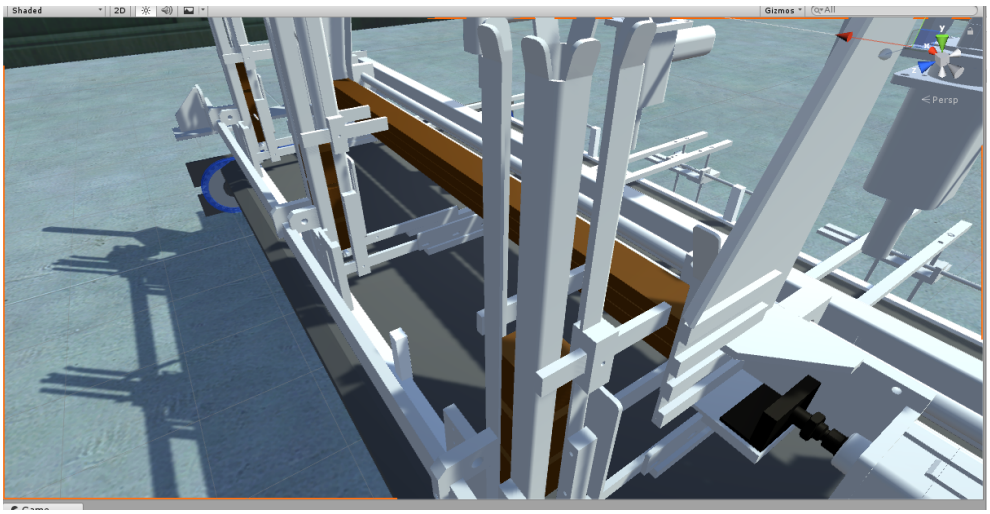
# Poglavje 4

# Izdelava 3D simulatorja

## 4.1 3D modeli

Za začetek smo v program Unity uvozili 3D modele delov stroja. Uvožene 3D modele delov stroja smo najprej umestili v prostor. To smo storili s kombinacijo 3D in 2D pogleda grafičnega urejevalnika programa Unity ter nastavljanjem koordinat 3D objektov s pomočjo miške in polj za vnos pozicij. Ko so bili deli stroja pravilno postavljeni, smo določili vrsto materiala posameznega dela. Vsem delom stroja smo dodali komponento togo telo programa Unity. Tej komponenti smo določili fizikalne lastnosti, ki definirajo kako se objekt, v tem primeru del stroja, obnaša ob stiku z drugim objektom. Objektu s komponento togo telo smo določili osnovne fizikalne lastnosti kot so masa, trenje, blaženje rotacije, moč gravitacije in tudi način zaznavanja trkov z drugimi objekti. Mizi, po kateri drsijo lesene kocke, smo nastavili zelo majhno trenje, zato da so se lesene kocke premikale brez zatikanja. Kockam pa smo nastavili malo večje trenje, zato da niso zdrsele naprej, potem ko se je sojemalo ustavilo. Nastavitev moči gravitacije je ostala taka, kot je bila že privzeta, tako, da je delovala na vse objekte enako. Način zaznavanja trkov je bil že privzeto nastavljen na diskretni način in tako je tudi ostalo, vse dokler nismo naleteli na problem, opisan v poglavju 6. Vrednosti mas objektov smo nastavljali le objektom, ki se gibljejo. Določili smo jih

tako, da so čim bližje vrednostim pravega stroja, katerega deli so v glavnem narejeni iz konstrukcijskega jekla in pa drugih jeklu podobnih materialov. Ti deli potem vplivajo na gibanje kock in kvadrov, ki so jim določene mase lesa in predstavljajo obdelovance. Objektom in materialom teh objektov, ki predstavljajo kovinske dele stroja, smo določili sive barve različnih odtenkov (slika 4.1). Objektom, ki predstavljajo lesene obdelovance, pa smo določili rjavo barvo.




Slika 4.1: Pogled na kovinske zalogovnike in lesene obdelovance.

## 4.2 Animacije

Zatem smo uvoženim objektom določili animacije. Med animirane dele stroja, ki se premikajo, spadajo vse tri pnevmatske pištole, ki se gibljejo vzdolžno v smeri x osi ter gor in dol v smeri y osi. Premikajo se tudi bati, ki pridržujejo kocke na zalogovniku kock. Dva bata, ki služita za izmet desk, se v animacijah tudi iztegneta. Ravno tako so animirana sojemala, ki krožijo okoli obdelovalne mize. Vsakemu animiranemu delu stroja smo nastavili komponento oziroma objekt animator, ki določa začetno stanje animacije in omogoča prehode v nadaljnje animacije. Za primer, bat za izmet desk smo nastavili

tako, da je v začetnem stanju skrčen, v naslednji animaciji pa iztegnjen. V komponenti animator smo tudi bolj natančno nastavili hitrost in blaženje prehodov med animacijami, saj Unity ponuja možnost oblikovanja grafa, ki določa blaženje prehodov. Pri modelu sojemal smo animacijo določili že prej v programu 3ds Max, v katerem smo določili tudi trajektorijo gibanja posamezne letve sojemala, ki potuje v obliki elipse okoli obdelovalne mize (slika 4.2). Zato smo v programu Unity določili le želeno hitrost animacije in pozicijo sojemal. Pri določanju pozicije sojemal nismo določali, kje na stroju se sojemala nahajajo, pač pa v kakšni poziciji so, oziroma kolikšno pot je posamezna letev že opravila.


Slika 4.2: Pogled na sojemala s strani.

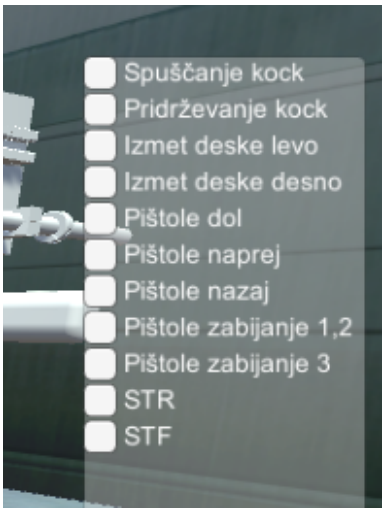
Za nastavitev pozicije letve smo nastavljali posamezni okvir animacije. Kasneje smo v uporabniški vmesnik dodali drsnik (slika 4.3), ki določa to pozicijo oziroma okvir animacije.

Po določenih animacijah je bilo že možno opazovati, na kakšen način stroj deluje, predvsem njegovi premikajoči se deli, vendar pa so delovali neusklajeno, zato smo morali napisati še skripte, ki upravljajo z delovanjem stroja preko uporabniškega vmesnika. Vse skripte, ki so uporabljene v programu Unity, smo napisali v jeziku C# v razvojnem okolju Visual Studio 2017.


Slika 4.3: Drsnik nastavljanja verige.

## 4.3 Izhodi stroja

Za ročno upravljanje stroja in njegovih izhodov smo naredili del uporabniškega vmesnika, preko katerega se lahko sproži posamezno animacijo, kar se odraža v premikanju posameznega dela stroja (slika 4.4). Ta del uporabniškega vmesnika smo naredili na podlagi izhodov krmilnika, ki upravlja z dejanskim strojem.




Slika 4.4: Ročno proženje izhodov simulatorja.

Da se ob kliku na posamezno polje, ki upravlja z izhodi, zgodi prava animacija, poskrbi skripta `ManualOutputAnim.cs`, ki poišče komponento oziroma skripto `AnimControl.cs` objekta stroja, ki upravlja z animacijami, ter pokliče metodo za vklop ali pa izklop želenega izhoda stroja (slika 4.5).

Skripta `AnimControl.cs` ob inicializaciji poišče objekte stroja, na kate-

```
public void change_spuscanje_kock(){
    if (!spuscanje_kock) {
        spuscanje_kock = true;
        kocka_mng.spuscanje_kock_on ();
    } else {
        spuscanje_kock = false;
        kocka_mng.spuscanje_kock_off ();
    }
}
```

Slika 4.5: Metoda za proženje spuščanja kock.

rih potekajo animacije, in shrani njihove komponente, imenovane animator, ki upravljajo z animacijami. Reference teh komponent se uporabljajo potem, ko so klicane metode za vklop izhodov oziroma spremembo animacij. S tem delom uporabniškega vmesnika se je dalo upravljati s strojem, tako da deluje usklajeno, vendar pa so manjkali še obdelovanci. Potrebam polnjenja zalogovnikov z obdelovalci je zadostil gumb uporabniškega vmesnika z napisom „napolni“, ki ob aktiviranju ustvari nove objekte kock in desk na pravih pozicijah zalogovnikov in tako poskrbi za polnitev simuliranega stroja z materialom. V takem stanju je možno popolnoma simulirati ročni način delovanja stroja, v katerem operater z ročnim pritiskanjem tipk in upravljanjem stroja pride do produkta. Za simuliranje avtomatskega načina pa so bili potrebni še vhodi stroja.

## 4.4 Vhodi stroja

Vsi senzorji, ki se nahajajo na stroju, so digitalni, kar pomeni, da imajo samo dve stanji. V enem senzor nekaj zaznava, v drugem pa ne. Optični in induktivni senzorji so priključeni na vhode krmilnika. Teh senzorjev je 26 in so opisani v tabeli 5.1. V programu Unity so namesto senzorjev narejeni objekti, ki smo določili, da so prožilci. Prožilci so postavljeni na tista mesta, kjer senzorji zaznavajo. Prožilec, ki zaznava, če so v enem od zalogovnikov za kocke prisotni obdelovanci, ni postavljen na mesto senzorja, pač pa direktno v zalogovnik na mesto kocke, ki mora biti zaznana s strani senzorja. Tako se na is-

tem mestu nahajata dva objekta, eden je v tem primeru kocka s fizikalnimi lastnostmi, drugi pa prožilec, ki ne vpliva na pot kocke, pač pa ob stiku s kocko le pokliče metodo `OnTriggerEnter()` skripte `InductCollider.cs`. V metodi `OnTriggerEnter()` se preveri oznaka objekta, ki je v stiku s prožilcem. Če je ta oznaka kocka ali deska, se v polju, ki definira vhode, nastavi vrednost ujemajočega vhoda na resnično. Podobno se zgodi ob klicu metode `OnTriggerExit()`, ko objekt ni več v stiku s prožilcem, le s to razliko, da se v takem primeru vrednost vhoda v polju nastavi na neresnično, saj je objekt zapustil zaznavno območje senzorja. Na enak princip delujejo tudi vsi ostali senzorji na simulatorju stroja, le simulacija optičnega dajalnika impulzov je narejena tako kot je opisano v poglavju 6. Po definiranih senzorjih in vhodih stroja je bil 3D simulator pripravljen za upravljanje. Manjkal je še del, ki določa logiko delovanja stroja in ki upravlja oziroma krmili z njim. V tem primeru je bil za avtomatsko upravljanje s strojem določen industrijski krmilnik Mitsubishi FX5U. Da s 3D simulatorjem stroja v programu Unity lahko upravlja dejanski krmilnik, ki se nahaja v nadzorni omari, ali pa le simulator tega krmilnika, ki se nahaja na istem računalniku kot 3D simulacija, pa je poskrbljeno z uporabo protokola Modbus.

## 4.5 Modbus

Modbus je protokol, ki se uporablja za porazdeljen nadzor aplikacij. Objavljen je bil leta 1979 kot serijski protokol za komunikacijo med inteligentnimi napravami. Dandanes je Modbus de facto standardni komunikacijski protokol, implementiran s strani več različnih proizvajalcev iz različnih industrij. Njegove prednosti so, da je odprt, brezplačen, enostaven za implementacijo in predvsem razvit z namenom za uporabo v industrijskih aplikacijah, zato danes večina modernih SCADA sistemov uporablja verzijo Modbusa, ki je implementirana preko TCP/IP [34, 23].

### 4.5.1 Serijski protokol Modbus

Pri izvedbi Modbusa preko serijskega protokola je več naprav povezanih na eno vodilo. Implementacija uporablja način gospodar-suženj za komunikacijo brez kolizij med napravami. Komunikacijo lahko začne samo gospodar, sužnji pa so naprave, ki le odgovarjajo. Dopuščena so tudi sporočila z več prejemniki, ki potujejo od gospodarja do sužnjev, vendar pa nimajo odgovora [8].

### 4.5.2 Protokol Modbus TCP

Protokol Modbus TCP uporablja TCP kot komunikacijski sloj, vendar pa poizkuša ostati združljiv z definicijo serijskega protokola. V specifikaciji je definirano okvirjanje Modbus paketkov v okvirje TCP z običajnimi glavami IP in TCP, ki so jim določena vrata 502. Modbus preko protokola TCP je tako bolj fleksibilen zaradi velike razširjenosti omrežij TCP/IP, vendar pa se ob implementaciji preko protokola TCP/IP pojavijo številni varnostni pomisleki, saj protokol Modbus ni bil zasnovan z mislijo na varnost. Zato je potrebno poskrbeti za varnost že na nivoju lokalnega omrežja ali pa uporabiti kakšno izmed novejših implementacij protokola Modbus, v kateri je implementirana tudi varnost [8].

### 4.5.3 Entitete protokola Modbus

Informacije o napravah so shranjene v štirih različnih tabelah. Dve tabeli hranita digitalne vrednosti, dve pa analogne. Dostop do teh informacij je mogoč preko protokola Modbus z naslavljanjem registrov, opisanih v spodnji tabeli (tabela 4.1) [26].

Če v praksi naslovimo naslov 40001, pomeni, da želimo izvedeti vrednost prvega analognega izhodnega registra. Številka 4 pove, da gre za analoge izhodne registre, 0001 pa, da gre za prvi analogni izhodni register. Včasih se je lahko naslavljajo največ do naslova x9999, kar pomeni, da je bilo največje število naslovov za določen tip registra 9999. Danes pa je to število naslovov

| Tip registra | Dostop | Naslov | Velikost |
|---|---|---|---|
| Diskretni izhodi | Branje in pisanje | 1-9999 | 1 bit |
| Diskretni vhodi | Branje | 10001-19999 | 1 bit |
| Analogni vhodi | Branje | 30001-39999 | 16 bitov |
| Analogni izhodi | Branje in pisanje | 40001-49999 | 16 bitov |

Tabela 4.1: Velikosti in naslovi registrov naprav, naslovljivi preko protokola Modbus.

večje, saj je možno dodati v naslov za vodilno številko še eno ali več ničel, kar povzroči, da se naslov 40001 spremeni v 400001 in se tako poveča največje možno število naslovov [1]. Enako velja tudi za vse ostale registre.

### 4.5.4 EasyModbus

Je zbirka programske opreme in knjižnic za razvoj rešitev z uporabo protokolov Modbus, Ethernet/ip,OPC DA, OPC UA in MQTT. Zbirka je izdana pod odprtokodno licenco [9] in podpira uporabo v jezikih C#,java in python. V primeru te diplomske naloge je bila uporabljena izvedba razvita v okolju .NET.

V program Unity je bila knjižnica EasyModbus dodana kot dinamična knjižnica. Po dodanem ukazu, ki omogoča uvoz tipov, definiranih v drugih imenskih prostorih, je bilo možno dostopati do vseh objektov in metod te knjižnice. Za začetek smo ustvarili nov objekt, ki je odjemalec protokola Modbus (slika 4.6). Ob inicializaciji tega objekta je potrebno dodati tudi naslov IP in vrata krmilnika oziroma naprave, na katero se bo program povezal.

Obstaja pa tudi možnost inicializacije objekta, ki je odjemalec protokola Modbus, z uporabo serijske povezave. V takem primeru bi bilo potrebno kot parameter dodati vmesnik COM namesto naslova IP in vrat. Metoda `ReadCoils()` je bila uporabljena za branje vrednosti izhodov krmilnika oziroma naprave, s katero je bila vzpostavljena povezava preko protokola Modbus. Kot parametre ta metoda sprejme naslov in pa odmik. Z odmikom

```
try
{
    //Debug.Log ("Trying to connect to plc over modbus");
    mc = new ModbusClient(ip, port);
    mc.Connect();
    Debug.Log("Connected to plc");
    plc_status.text = "OK";

    mc.connectedChanged += new EasyModbus.ModbusClient.ConnectedChanged(connection_changed);
    //mc.sendDataChanged += new EasyModbus.ModbusClient.SendDataChanged(send_data_changed);
    //mc.receiveDataChanged+= new EasyModbus.ModbusClient.ReceiveDataChanged(received_data_changed);


}
catch
{
    Debug.Log("Connection to plc failed");
    plc_status.text = "FAILED";
}
chainSlider.change_chain_pos(0.1722f);
```

Slika 4.6: Ustvarjanje novega objekta, ki je odjemalec pri protokolu Modbus.

je določeno število izhodov, ki bodo prebrani od začetnega naslova naprej. Odmik tudi določa dolžino seznama, v katerega so kot rezultat operacije branja vnesene diskretne vrednosti izhodov. Ta seznam vrednosti izhodov pa določa, kateri del stroja oziroma izhod na 3D simulatorju bo spremenil svoje stanje.

Take vrste komunikacija je zadoščala za povezavo z dejanskim krmilnikom, saj je bilo na 3D simulatorju v programu Unity odraženo pravo stanje stroja v realnem času. Za potrebe razvoja programa krmilnika pa 3D simulator pošilja vrednosti virtualnih senzorjev na vhode krmilnika. Ker protokol Modbus ne dovoljuje pisanja po vhodnih registrih, je bila za prenos vrednosti uporabljena metoda `WriteMultipleCoils()`, ki piše po izhodnih registrih in kot parametra sprejme začetni naslov izhodnega registra in pa seznam diskretnih vrednosti, ki so vpisane od prej določenega naslova naprej. Te vrednosti smo kasneje v programu GX Works3 s pomočjo funkcije move 4.10 premaknili na vhode simuliranega krmilnika. Tako smo dosegli, da se v simulatorju krmilnika programa GX Works3 pojavijo vrednosti vhodov 3D simulacije stroja, narejenega v programu Unity. Vendar pa je bilo pred tem potrebno še narediti vmesnik med simulatorjem krmilnika in strežnikom Modbus, ker simulator krmilnika v programu GX Works3 ne podpira omrežnih povezav

in s tem onemogoča komunikacijo preko protokola Modbus.

### 4.5.5 Vmesnik med simulatorjem krmilnika in strežnikom Modbus

Da je 3D simulator, narejen v programu Unity, lahko komuniciral s simulatorjem krmilnika, imenovanem GX simulator, v programu GX Works3 preko protokola Modbus, je bilo potrebno narediti vmesnik (slika 4.7), ki se na eni strani izdaja za napravo protokola Modbus, oziroma neke vrste krmilnik, do katerega vhodov in izhodov dostopa 3D simulator, narejen v programu Unity. Na drugi strani pa vmesnik piše na vhode simulatorja krmilnika in pa pošilja statuse izhodov simulatorja krmilnika nazaj na 3D simulator stroja v programu Unity prek protokola Modbus. To smo izvedli z uporabo programskih orodji EasyModbus in MX Component.

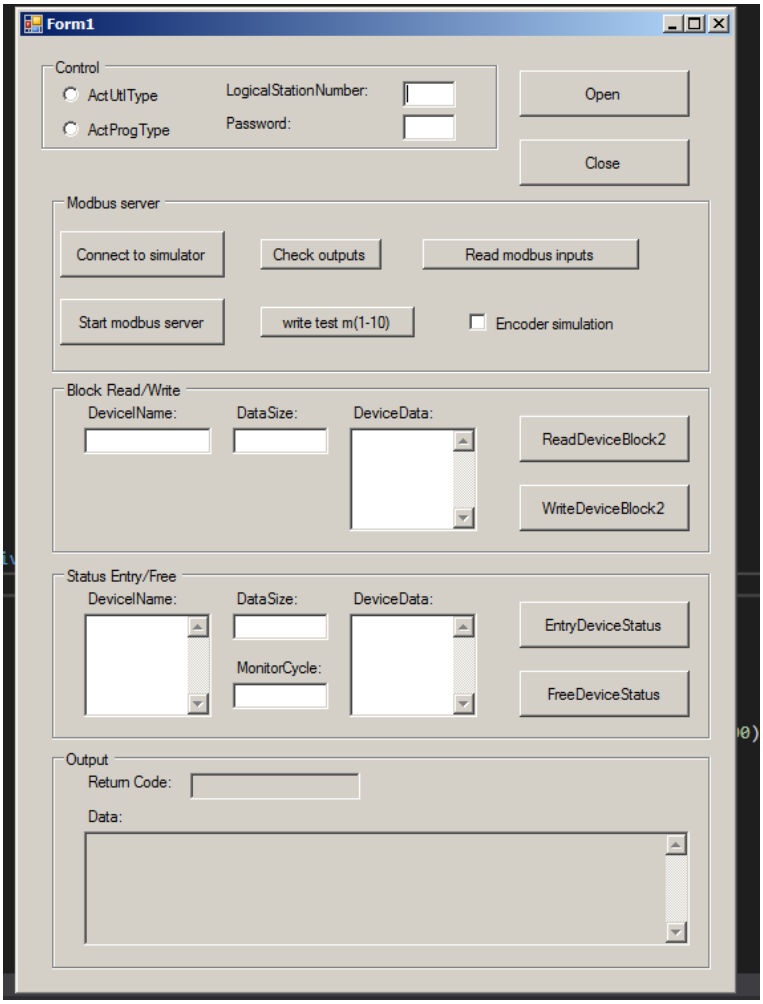
#### 4.5.5.1 MX Component

Je zbirka programskih orodij in knjižnic, ki omogočajo povezovanje krmilnikov proizvajalca Mitsubishi s programskimi orodij in jeziki proizvajalca Microsoft [7]. Ta zbirka omogoča bralni in pisalni dostop do krmilnikov ne glede na vrsto povezave z uporabo Microsoftovih programov in razvojnih okolij kot so Access, Excel, Visual studio, .NET in programskih jezikov kot Visual Basic, C++, C#.

#### 4.5.5.2 Program Communication Setup Utility

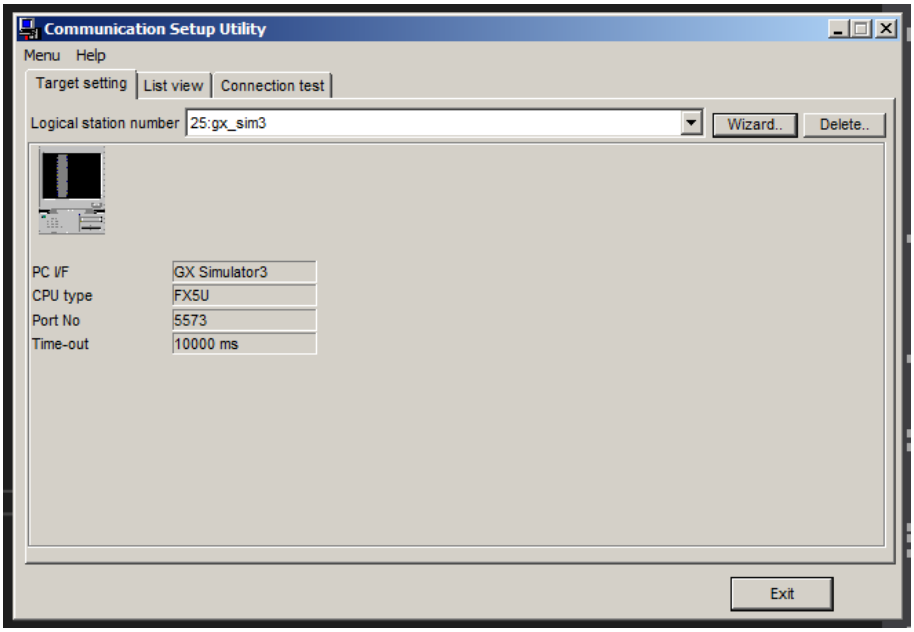
Preko tega programa je omogočen programski dostop do simulatorja krmilnika programa GX Works3. Pred tem je potrebno nastaviti povezavo do želene naprave. To smo storili s programom Communication Setup Utility, ki je del zbirke orodij MX component.

V čarovniku nastavitev tega programa smo najprej izbrali vrsto povezave do krmilnika [3]. Na voljo je veliko možnosti povezav, med katere spadajo USB, serijska, Ethernet, modem in GX simulator. Za potrebe dostopa do




Slika 4.7: Uporabniški vmesnik vmesnika med simulatorjem krmilnika in strežnikom protokola Modbus.

simulatorja programa GX Works3 smo izbrali vrsto povezave GX simulator3 (slika 4.8). Nato smo določili še vrsto krmilnika, ki je FX5U. Številka vrat 5573 in čas prekinitve povezave 10000 ms sta ostala nespremenjena. Novo dodano povezavo smo lahko videli v zavihku z napisom Target setting in List view. Zavihek Connection test pa omogoča, da se posamezno povezavo testira, če deluje.




Slika 4.8: Nastavitev komunikacije s simulatorjem krmilnika.

#### 4.5.5.3 Povezovanje

V naslednjem koraku smo uporabili vzorčni program zbirke MX component, ki je napisan v jeziku C#, kot projekt programa Visual studio. V tem projektu je narejen uporabniški vmesnik, ki uporablja glavne funkcije, namenjene branju in pisanju ter vzpostavljanju povezave s krmilnikom oziroma simulatorjem. Za odpiranje povezave je uporabljen kontrolnik ActiveX imenovan `ActUtlType`, ki je projektu dodan kot referenca in se nahaja v datoteki `ActUtlTypeLib.dll`. Povezavo se odpre s klicem metode `Open()` tega kontrolnika. Ta metoda tudi vrne statusno kodo glede na uspešnost povezave. Statusna koda bo imela vrednost 0 pod pogojem, da je bila povezava in njena naprava uspešno dodana že prej v programu Communication Setup Utility in da je naprava v času povezovanja nanjo v aktivnem stanju. V primeru neuspešnega povezovanja pa bo vrednost te kode različna od 0.

#### 4.5.5.4 Branje in pisanje

Za branje vrednosti naprave, s katero je vzpostavljena povezava, smo uporabili metodo `ReadDeviceBlock()`, ki sprejme tri parametre:

- naslov lokacije v napravi, od katerega naprej so vrednosti prebrane,
- odmik, ki pove, koliko blokov 16-bitnih vrednosti bo prebranih,
- tabelo celih števil, v katero bodo zapisane prebrane vrednosti kot decimalna števila.

Ravno tako kot metoda `open()` tudi metoda `ReadDeviceBlock()` vrne statusno kodo, ki sporoča neuspeh, če je različna od 0. V primeru branja izhodov krmilnika, začenši s prvim, se kot naslov vstavi naslov Y0 (slika 4.9). Kot odmik se vstavi število 1 in na tak način v tabeli vrnjenih vrednosti dobimo statuse prvih šestnajstih izhodov, od Y0 do Y7 in od Y10 do Y17. Vendar pa se v tabeli ne nahajajo vrednosti, ki so digitalne, se pravi so resnične ali pa neresnične, pač pa se nahaja samo ena decimalna vrednost, katero je potrebno pretvoriti v binarno število, zapisano s 16-biti. Pretvorjena vrednost za vključen prvi izhod izgledala takole 0000000000000001. Decimalna vrednost za prve tri vključene izhode bi bila 7, binarna pa 0000000000000111.

Pisanje poteka podobno kot branje. Metoda `WriteDeviceBlock()` sprejme enake tri parametre kot metoda `ReadDeviceBlock()`, le s to razliko, da bo vrednost, ki se nahaja v tabeli celih števil, zapisana na napravo. Ker pa so želene vrednosti, ki bodo zapisane, hranjene v programu kot tabela binarnih vrednosti, je v tem primeru potrebno pretvoriti tabelo binarnih vrednosti v 16-bitno decimalno število [4].

### 4.5.6 Strežnik Modbus

Posrednik, ki omogoča dostop do vrednosti vhodov in izhodov simulatorja krmilnika, je strežnik Modbus. V uporabniškem vmesniku vmesnika je gumb z napisom start modbus server (slika 4.7). Ob pritisku na ta gumb se ustvari nov objekt tipa strežnik Modbus knjižnice EasyModbus, ki je v

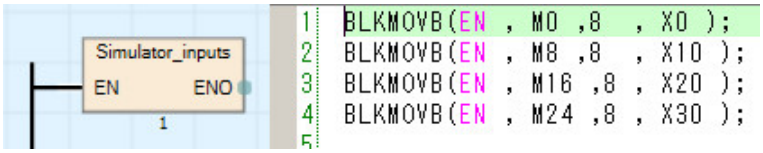
```
private void check_outputs_clean()
{
    int iReturnCode = 1;
    int[] outputs = new int[16];
    iReturnCode = axActUtlType1.ReadDeviceBlock("Y0", 1, out outputs[0]);

    String output_s = Convert.ToString(outputs[0], 2).PadLeft(16, '0');
    int c1 = 0;

    foreach (char c in output_s)
    {
        bool b_var = !(c == '0');
        modbusServer.coils[c1] = b_var;
        Debug.Write("Y" + (c1.ToString()) + ": " + b_var + "  ");
        c1++;
    }
    Debug.Write("\n");
}
```

Slika 4.9: Branje izhodov simulatorja krmilnika.

projekt dodana kot referenca, tako kot tudi knjižnica ActiveX kontrolnika `ActUtlTypeLib`. Objekt tipa strežnik Modbus ima metodo `Listen()`, s klicem katere začne strežnik poslušati na vratih 502. Vrednosti izhodov in vhodov objekta strežnik Modbus se nahajajo v tabelah s poimenovanjem coils in discreteInputs. Ker se po tabeli z imenom discreteInputs ne da pisati, se vrednosti vhodov in izhodov simulatorja v programu Unity vpisujejo v tabelo coils, s tem, da se do naslova 64 nahajajo izhodi, od naslova 64 naprej pa vhodi.

Na tak način je zagotovljeno, da se sprememba statusa posameznega senzorja na 3D simulatorju zapiše v tabelo coils objekta strežnik Modbus. Za tem je uporabljena knjižnica `ActUtlTypeLib`, ki uporablja komponento ActiveX, ki omogoča programski dostop do simulatorja krmilnika v programu GX Works3. Z uporabo te knjižnice smo poskrbeli, da se vrednost vhoda, ki je bila poslana preko protokola Modbus, posodobi tudi v simulatorju krmilnika. Poslana vrednost vhoda se sprva zapiše na pomnilniško mesto simulatorja krmilnika od naslova M0 naprej, saj knjižnica `ActUtlTypeLib` ne dovoljuje pisanja po vhodih. Da se ta vrednost pojavi na dejanskem vhodu simulatorja krmilnika, pa je v programu, napisanemu za krmilnik, na začetku uporabljen

funkcijski blok move, ki premakne 32 vrednosti od naslova M0 na mesta od X0 naprej (slika 4.10).




Slika 4.10: Preslikava naslovov v programu GX Works3.

# Poglavje 5

# Razvoj programa za krmilnik

## 5.1 Koraki v razvoju posameznega programa za krmilnik

Razvoj posameznega programa za krmilnik poteka v razvojnem okolju programa GX Works3. Najprej ustvarimo nov projekt in mu določimo parametre modulov. Potem ustvarimo posamezne organizacijske enote programa. Tem določimo zapovrstje in tipe izvajanja. Ko imamo določeno zapovrstje in tipe izvajanja, se lahko lotimo poimenovanja globalnih in lokalnih oznak. Sledi urejanje programa posamezne organizacijske enote. Pred nalaganjem in razhroščevanjem napisanega programa na simulatorju moramo izvesti še pretvorbo programa. Ko smo zadovoljni s programom, se lahko povežemo na fizični krmilnik. Na krmilnik se lahko povežemo na tri različne načine:

- preko USB vrat,
- preko mrežnih kartic,
- preko serijskih vrat.

Na krmilnik potem naložimo napisan program in ostale parametre. Krmilnik preklopimo v način izvajanja programa in preverimo izvajanje naloženega

programa. Uspešnemu izvajanju naloženega programa sledi samostojno delovanje krmilnika [6].

## 5.2 Razvoj programa stroja za zbijanje palet

### 5.2.1 Ustvarjanje projekta

Najprej smo ustvarili nov projekt v programu GX Works3. Zatem smo nastavili parametre modulov. Določili smo oznako uporabljenega krmilnika, ki je FX5U. Nato smo ustvarili posamezne organizacijske enote programa, ki so funkcijski bloki. Programu smo določili, da se izvaja vsak procesorski cikel, zato smo izbrali tip izvajanja s poimenovanjem Scan. Temu je sledilo določanje globalnih in lokalnih oznak.

### 5.2.2 Določanje globalnih oznak vhodov in izhodov

Za razvidnejše sledenje posameznim naslovom vhodov in izhodov smo najprej nastavili globalne oznake, s katerimi so bili naslovi vhodov in izhodov opisno poimenovani. Na tak način smo zagotovili lažjo berljivost programa in dostopnost do poimenovanih vhodov in izhodov (tabela 5.1 in tabela 5.2).

| Naslov | Ime | Opis |
|---|---|---|
| X0 | Optični dajalnik impulzov A | Vhoda, ki dajeta signal ob premiku optičnega dajalnika impulzov. |
| X1 | Optični dajalnik impulzov B | |
| X2 | Takt | Signal induktivnega senzorja, ki zaznava pozicijo posamezne letve sojemala na začetku obdelovalne mize. |
| X3 | Zalogovnik kock 1 | Signali treh kapacitivnih senzorjev, ki zaznavajo, če so lesene kocke prisotne v zalogovnikih. |
| X4 | Zalogovnik kock 2 | |
| X5 | Zalogovnik kock 3 | |
| X6 | Zalogovnik desk | Signal kapacitivnega senzorja, ki zaznava prisotnost lesenih desk v zalogovniku za deske. |
| X7 | Zalogovnik žebljev 1 | Signali treh induktivnih senzorjev, ki zaznavajo, če so žeblji prisotni v zalogovnikih žebljev. |
| X10 | Zalogovnik žebljev 2 | |
| X11 | Zalogovnik žebljev 3 | |
| X12 | Kocke pridržane | Induktivni senzor Reed na cilindru bata, ki pridržuje kocke s spodnje strani. Signal sporoča, da je bat iztegnjen in da so kocke pridržane. |
| X13 | Kocke spuščene | Induktivni senzor Reed na cilindru bata, ki pridržuje kocke s spodnje strani. Signal sporoča, da je bat skrčen in da so kocke spuščene. |
| X14 | Izmet deske levo nazaj | Induktivna senzorja Reed na cilindrih dveh batov, ki na levi in na desni strani izmetavata deske. Na posameznem cilindru sta dva senzorja, ki sporočata, da je bat skrčen ali pa iztegnjen. |
| X15 | Izmet deske levo naprej | |
| X16 | Izmet deske desno nazaj | |
| X17 | Izmet deske desno nazaj | |
| X20 | Pištola 1 dol | Induktivni senzorji na nosilcih pištol, ki zaznavajo, če je pištola pravilno naslonjena na obdelovanec ali pa ni naslonjena nikamor. Če je posamezna pištola pravilno naslonjena, se prižge eden od vhodov z imenom pištola dol, drugače pa so prižgani vhodi z imenom pištola gor. |
| X21 | Pištola 2 dol | |
| X22 | Pištola 3 dol | |
| X23 | Pištola 1 gor | |
| X24 | Pištola 2 gor | |
| X25 | Pištola 3 gor | |
| X26 | Pištole nazaj | Induktivna senzorja Reed na cilindru bata, ki prečno premika gred, na katero so pričvrščene pištole. |
| X27 | Pištole naprej | |
| X30 | Start cikla | Signal tipke, ki je povezana kot normalno odprt kontakt. Signal sporoča, da uporabnik želi pognati avtomatski cikel delovanja. |
| X31 | Izhodi | Signal, ki sporoča, da je varnostni tokokrog vklopljen in da imajo vsi izhodi napetost. |

Tabela 5.1: Vhodi.

| Naslov | Ime | Opis |
|---|---|---|
| Y0 | Auto | Luč, ki signalizira avtomatsko delovanje. |
| Y1 | Kontaktor sojemala | Kontaktor, ki da napetost elektromotorju za začetek kroženja sojemal. |
| Y2 | Spuščanje kock | Izhod, ki premika bate za zapiranje zalogovnika s kockami. Če je izhod Y2 izključen so bati iztegnjeni in posamezni zalogovniki zaprti. Ob vklopu signala izhoda Y2, pa se bati, ki zapirajo posamezne zalogovnike, skrčijo in jih tako odprejo. Bati ostanejo skrčeni vse do ponovnega izklopa Y2. Bati so vezani na skupni ventil, s katerim upravlja izhod Y2. |
| Y3 | Pridrževanje kock | Izhod, ki premika bate za pridrževanje kock v posameznem zalogovniku. Deluje ravno obratno kot izhod za spuščanje kock, saj so v neaktivnem stanju izhoda Y3 bati skrčeni, v aktivnem pa iztegnjeni. Bati, s katerimi upravlja izhod Y3, so vezani na skupen ventil. |
| Y4 | Izmet deske levo | Izhoda, ki upravljata z iztegovanjem batov za izmet deske. V neaktivnem stanju izhoda je posamezen bat skrčen, v aktivnem pa iztegnjen. |
| Y5 | Izmet deske desno | |
| Y6 | Pištole dol | Izhod, ki upravlja z bati za spuščanje pištol. Vsi trije bati so vezani na en ventil, s katerim upravlja ta izhodni signal, zato se ob vklopu tega izhoda spustijo vse tri pištole. Ob izklopu signala tega izhoda, pa se pištole dvignejo. |
| Y7 | Pištole naprej | Izhoda, ki upravljata z batom za premikanje gredi na katero so pričvrščene vse pištole. Deluje na enak način kot bat za pridrževanje kock. |
| Y10 | Pištole nazaj | |
| Y11 | Pištola zabijanje 1 | Izhodi katerih signali sprožijo posamezno pištolo, da pribije žebelj. |
| Y12 | Pištola zabijanje 2 | |
| Y13 | Pištola zabijanje 3 | |
| Y14 | STF | Signal na frekvenčni pretvornik, ki ob aktivnem stanju začne z vrtenjem motorja za premikanje verige s sojemali v smeri naprej. |
| Y15 | STR | Ravno tako signal za premikanje verige, le s to razliko, da povzroči, da se motor začne vrteti v smeri nazaj. |

Tabela 5.2: Izhodi.

### 5.2.3 Ročni način

V ročem načinu operater sam upravlja s posameznim izvršnim členom stroja z omejitvami, ki preprečujejo mehanske poškodbe stroja. V našem primeru operater upravlja stroj v ročnem načinu preko na dotik občutljivega zaslona. Na njem se nahajajo gumbi, ki opisujejo, kaj se s pritiskom na njih vklopi. Operater lahko izbira med:

- pomikom sojemala naprej,
- pomikom sojemala nazaj,
- doziranjem kock,

- doziranjem desk,
- spuščanjem pištol,
- dviganjem pištol,
- zabijanjem posamezne pištole,
- prečnim pomikom pištol naprej,
- prečnim pomikom pištol nazaj.

Pritiske na gumbe smo preslikali na pomnilniške naslove pomnilnika od naslova M150 naprej. Pritisk gumba „pomik sojemala“ v ročnem načinu naprej vklopi bit M150, ki hkrati vklopi tudi funkcijski blok pomik. Ta pa vključi vrtenje elektromotorja, ki premika sojemala naprej. Dokler je gumb pritisnjen, je vključen tudi elektromotor, ki zagotavlja premikanje sojemala.

### 5.2.4 Avtomatski način

Razvoju ročnega načina je sledil razvoj avtomatskega načina, pri katerem smo začeli s programom za doziranje obdelovancev.

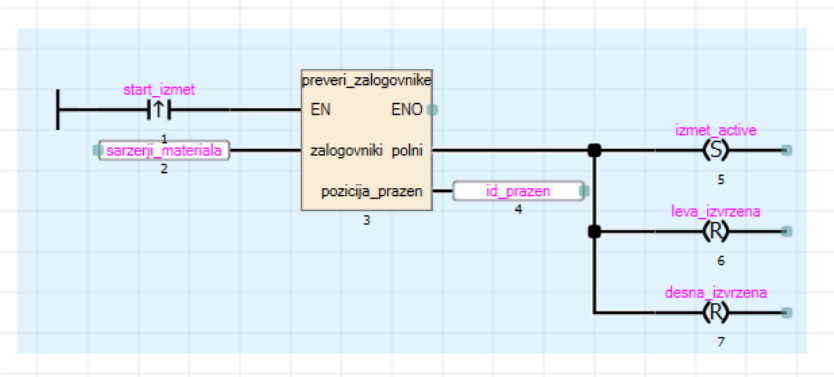
#### 5.2.4.1 Doziranje obdelovancev

Doziranje obdelovancev poteka v več korakih. Začeli smo s preverjanjem stanja zalogovnikov.

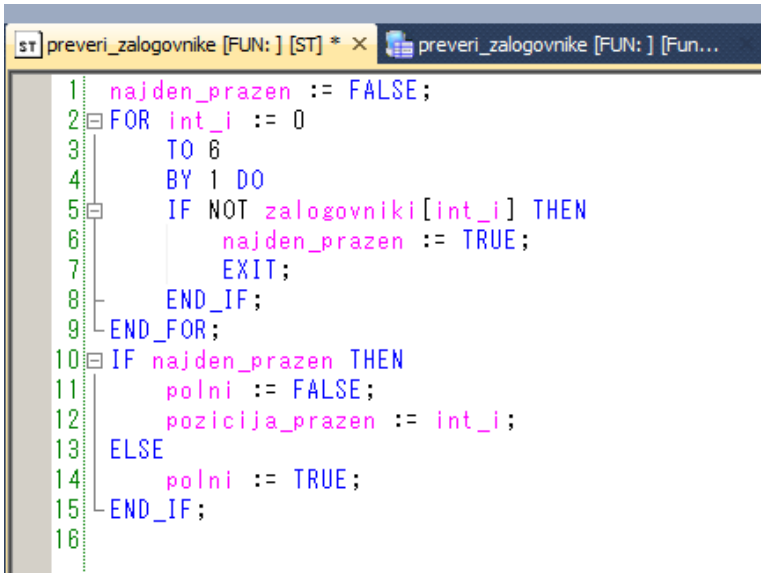
##### 5.2.4.1.1 Preverjanje zalogovnikov

Ob pritisku na tipko start cikla se začne avtomatsko delovanje stroja. Najprej preverimo, če imajo vsi zalogovniki dovolj materiala. V programu je za tako preverjanje zadolžen funkcijski blok, kateri na vhodu sprejme seznam zalogovnikov (slika 5.1). Nato se v zanki preveri, če je slučajno vrednost kakšnega vhoda, ki jo predstavljajo induktivna stikala na zalogovnikih, neresnična. Če je, pomeni, da obstaja zalogovnik, ki je prazen. Najdena neresnična vrednost potem zagotovi, da je tudi izhodna vrednost funkcijskega

bloka neresnična (slika 5.2). Drugi izhod funkcijskega bloka, poimenovan „pozicija prazen“, pa pove, kateri zalogovnik je prazen.



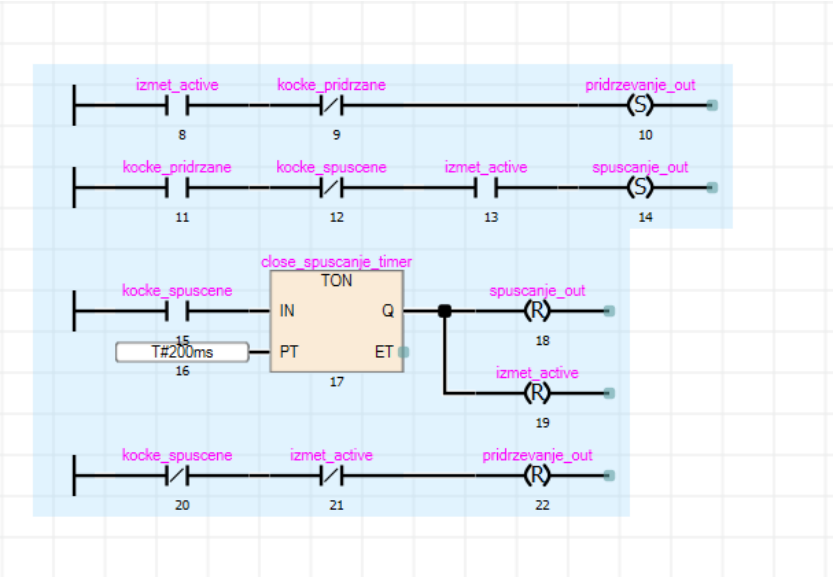

Slika 5.1: Funkcijski blok, ki preveri zalogovnike.




Slika 5.2: Strukturiran tekst funkcijskega bloka, ki preveri zalogovnike.

#### 5.2.4.1.2 Izmet kock

Doziranje obdelovancev se lahko začne pod pogojem, da funkcijski blok, ki preverja stanja zalogovnikov, ne javi napake. Med obdelovance spadajo tri kocke in deska. Najprej se sproži doziranje kock. V funkcijskem bloku izmet se nastavi bit, ki povzroči, da se vklopi izhod Y3, poimenovan pridrževanje kock (slika 5.3). S pridržanjem kock, ki so druge v vrsti, je preprečeno, da bi izpadle vse kocke iz zalogovnikov, potem ko se skrči cilinder, ki s spodnje strani zapira zalogovnike kock. Ko so kocke pridržane, se vklopi tudi izhod Y2, poimenovan spuščanje kock, ki skrči cilindre, ki s spodnje strani zapirajo zalogovnike in tako povzroči, da kocke padejo na obdelovalno mizo. Izklop izhoda Y2, ki povzroči izteg cilindra in ponovno zaprtje zalogovnikov s kockami, pa je pogojen s funkcijskim blokom časovnika, ki vklopi svoj signal po določenem pretečenem času. Za dolžino tega časa smo sprva določili 500 ms, kar je dovolj, da gravitacija pred ponovnim iztegom cilindra premakne kocke na obdelovalno mizo. Ta čas smo kasneje še zmanjšali.



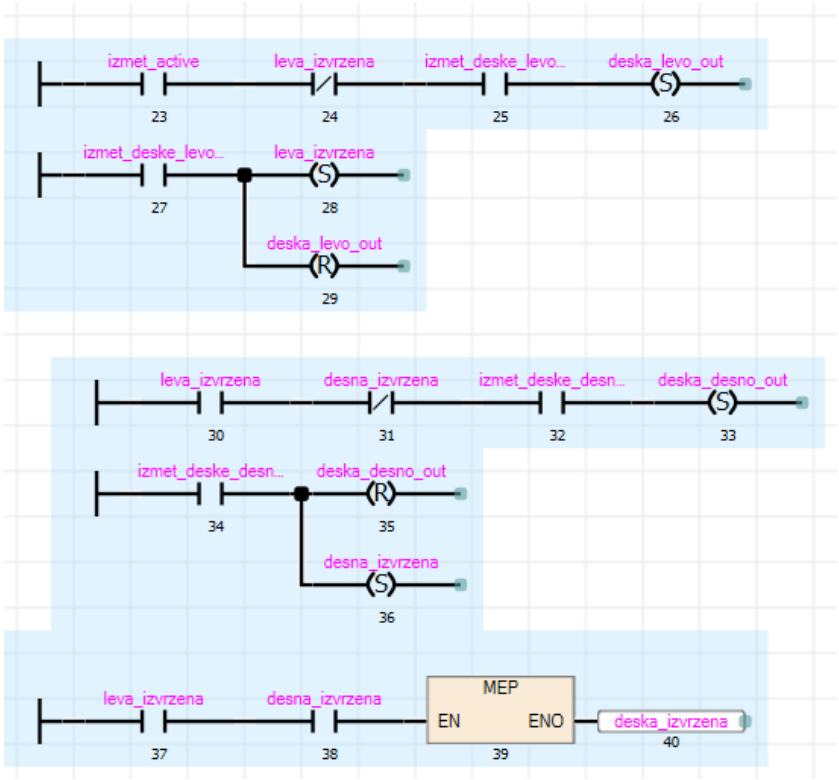

Slika 5.3: Lestvični diagram poteka izmeta kock.

#### 5.2.4.1.3 Izmet desk

Istočasno z izmetom kock poteka tudi izmet desk. Pri izmetu desk je pomembno, da je naprej izvržena leva stran deske, zato da lahko potem izteg desnega bata hkrati izvrže in poravna desno stran deske ob omejevalec na levi strani. Izhod Y4, ki je poimenovan izmet leve deske, se vklopi in ostane vklopljen vse dokler izteg bata ne vklopi vhoda X15, s poimenovanjem izmet deske levo naprej (slika 5.4). Ob tem se izhod Y4 izklopi in povzroči, da se bat vrne nazaj. Ko se vhod X14, ki je označen z imenom izmet deske levo nazaj, ponovno vklopi, se začne tudi izmet deske na desni strani, za kar je poskrbljeno z vklopom izhoda Y5. Izteg desnega bata poteka na enak način kot na levi strani, le s to razliko, da se ob končanem izmetu in vrnitvi desnega bata v začetno pozicijo vklopi izhod funkcijskega bloka izmet, ki signalizira, da so bili obdelovanci izvrženi. Naslednji korak je pomik obdelovancev do pištol za zabijanje.

### 5.2.4.2 Pomik obdelovancev

Za sledenje poziciji posamezne letve sojemala in obdelovancev je uporabljen optični dajalnik impulzov, ki je nameščen na isti gredi kot zobnik za premikanje verige sojemala. Gred, na katero je nameščen zobnik, je gnana z elektromotorjem. Da se elektromotor začne vrteti v smeri naprej, poskrbi vklop izhoda Y14, ki je povezan na frekvenčni pretvornik, ki upravlja z elektromotorjem. Obenem pa se mora vključiti tudi izhod Y1, ki vključi kontaktor, ki zagotovi, da ima elektromotor električno napetost. Ko se začne rotor elektromotorja vrteti, se vrti tudi gred, na katero sta nameščena zobnik verige in optični dajalnik impulzov. Optični dajalnik impulzov daje dva signala, ki sta povezana na vhoda X0 in X1. Za branje signalov optičnega dajalnika impulzov je bilo najprej potrebno zmanjšati čas branja vhodov na katere je optični dajalnik impulzov priključen. Pri tovarniških nastavitvah je ta čas branja 10 ms, kar je preveč za sledenje signalom optičnega dajalnika impulzov, zato smo ta čas zmanjšali na 100 µs (slika 5.5).

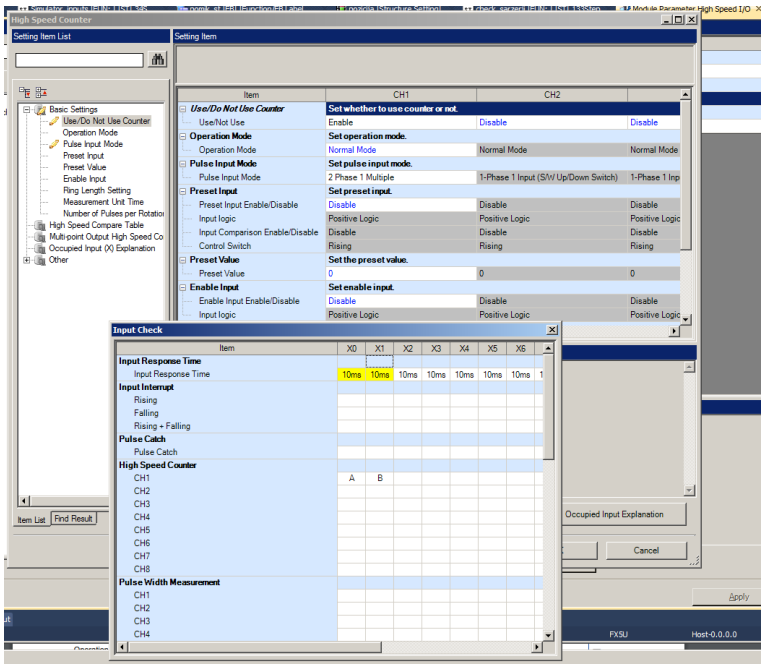



Slika 5.4: Lestvični diagram poteka izmeta deske.

To smo nastavili v nastavitvah parametrov krmilnika. Na istem mestu smo nastavili tudi lastnosti števnika, ki lahko šteje z veliko hitrostjo. Ta števnik je uporabljen za štetje premikov optičnega dajalnika impulzov. Razpon tega števnika je od -2147483648, pa do +2147483647. En impulz pri vrtenju optičnega dajalnika impulzov povzroči, da se vrednost števnika poveča za 1. Na nalepki optičnega dajalnika impulzov je napisana ločljivost, ki pove število impulzov na rotacijo gredi optičnega dajalnika impulzov. Iz tega podatka in iz obsega zobnika, ki premika verigo s sojemali, izračunamo razdaljo, prepotovano v enem impulzu 5.1.

$$razdalja = c/(R * res), \tag{5.1}$$

kjer je $c$ obseg zobnika, $R$ prestavno razmerje reduktorja, $res$ pa ločljivost




Slika 5.5: Spreminjanje hitrosti branja vhodov.

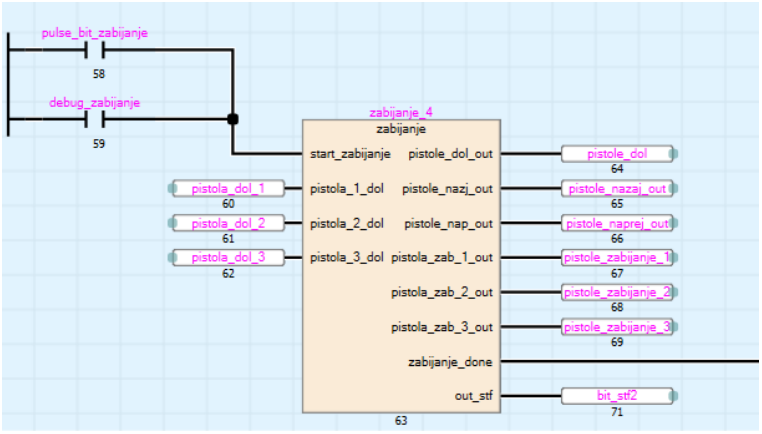
optičnega dajalnika impulzov.

Izračunano prepotovano razdaljo na en impulz smo v programu zabeležili kot konstanto. Vrednost števnika, ki lahko šteje z veliko hitrostjo, pomnožena s prej izračunano konstanto, pa predstavlja prepotovano pot posamezne letve sojemala, ki premika kocke in desko od zalogovnikov za izmet kock naprej. Induktivni senzor z imenom takt na vhodu X2 ponastavi vrednost števnika, ki lahko šteje z veliko hitrostjo, ko zazna letev sojemala. Tako smo določili začetno referenčno točko, ki se uporablja pri merjenju prepotovane razdalje obdelovancev. Razdaljo od induktivnega senzorja s poimenovanjem takt pa do pozicije pištol smo v programu ravno tako definirali kot konstanto, saj se ne spreminja. Ko je ta razdalja dosežena, se pomik sojemal ustavi. V taki poziciji se lesene kocke in deska nahajajo ravno pod pištolami in so pripravljeni na pribitje prvega žeblja. Medtem, ko se sojemalo pomika na želeno mesto, se pri prepotovani razdalji 20 cm ponovno vključi izmet obdelovancev. Izmet se vključi pod pogojem, da zalogovniki niso prazni. Prepotovana razdalja 20

cm zagotavlja, da so vsi izvrženi obdelovanci umaknjeni. V primeru, da so zalogovniki prazni, se nastavi bit, ki je uporabljen po končanem pribijanju žebljev. Ta bit poskrbi, da obdelovanci ne ostanejo pod pištolami, ampak da pridejo ven iz stroja.

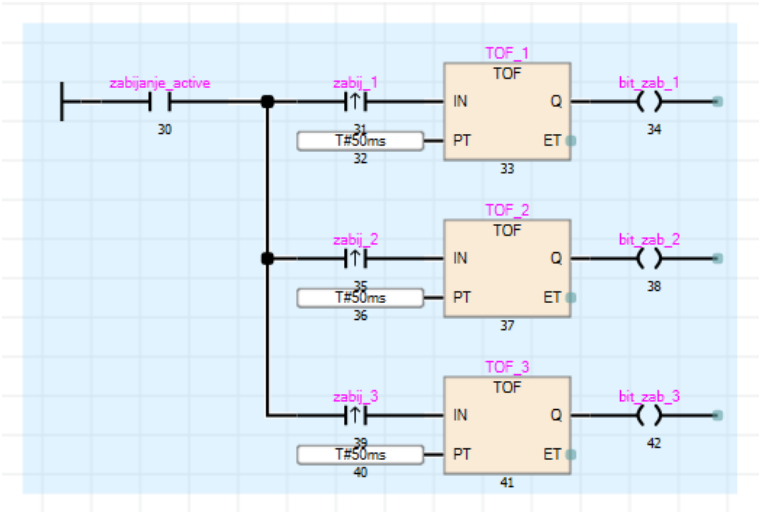
#### 5.2.4.3 Povezovanje kock z desko

Za zabijanje žebljev s pištolami poskrbi funkcijski blok z imenom zabijanje (slika 5.6).




Slika 5.6: Funkcijski blok, ki skrbi za zabijanje.

Ob signalu, ki omogoči izvajanje tega funkcijskega bloka, se nastavi bit, ki upravlja z izhodom Y6 in spušča pištole. Stroj je mehansko tako zasnovan, da lahko zabija tudi v krajšo desko s samo dvema kockama. Da tretja pištola v takem primeru ne zabije žeblja v prazno je poskrbljeno v programu. Vse tri pištole imajo na vrhu induktivne senzorje, ki zaznajo, če se je pištola naslonila na desko ali pa ni naslonjena nikamor. Ob signalu posameznega vhoda, ki je lahko X20, X21 ali pa X22 in predstavlja spuščeno prvo, drugo ali pa trejo pištolo, se vklopi tudi ujemajoč izhod, ki je lahko Y11, Y12 ali pa Y13 in sproži zabijanje. Trajanje vklopa zabijanja pištole je pogojeno s časovnikom (slika 5.7), ki drži izhod vklopljen malo dlje časa, da se pištola lahko normalno sproži, saj je en cikel procesorja krmilnika premalo za aktiviranje pnevmatskega mehanizma. Pištole se dvignejo po pretečenem časovniku, ki se sproži ob zadnji spremembi enega izmed izhodov za zabijanje iz aktivnega stanja v neaktivno. Da so pištole ponovno dvignjene in ne več naslonjene na desko, je razvidno iz vhodov X23, X24, X25. Ko so ti vhodi prižgani, se vklopi izhod Y7, ki premakne gred, na katero so nameščene pištole naprej. Ta izhod ostane




Slika 5.7: Lestvični diagram trajanja vklopa zabijanja posamezne pištole.

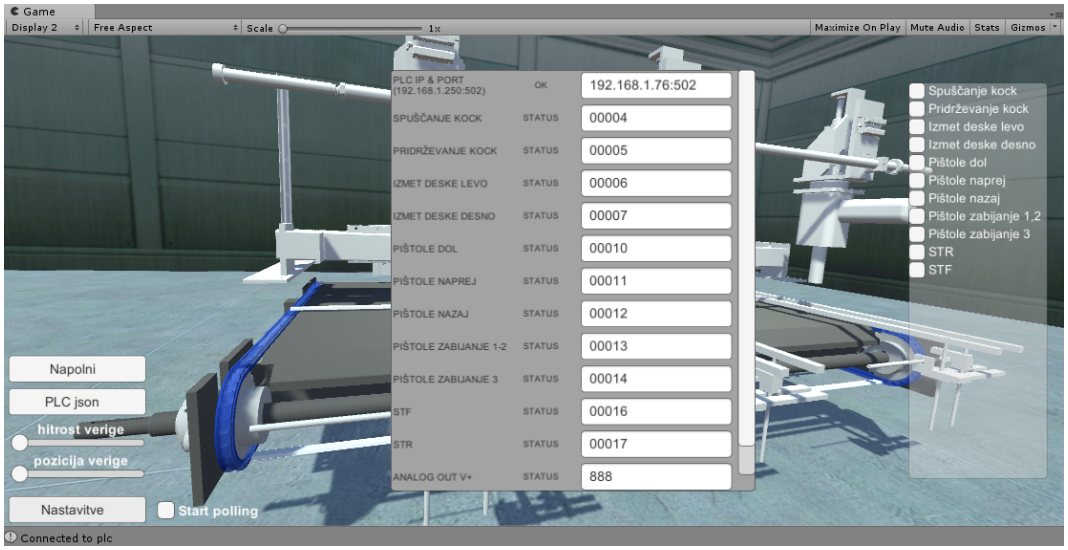
vklopljen vse do vklopa vhoda X27, ki sporoča, da je bat, ki prečno premika gred pištol, iztegnjen. Prečni pomik pištol naprej zagotovi, da žeblja nista pribita na isti liniji, ampak da sta diagonalno zamaknjena. Pred pribitjem drugega žeblja je potrebno premakniti sojemalo s kockami in desko naprej za centimeter manj kot je širina deske. Za to poskrbi prej definiran funkcijski blok pomik in določena konstanta širine deske. Ko so deska in kocke na pravem mestu, se ponovno sproži pribijanje žebljev, ki poteka na enak način kot za prvi žebelj. Po končanem pribijanju se pištole dvignejo in gred, na katero so pričvrščene, se pomakne nazaj v začetno pozicijo, za kar poskrbi vklop izhoda Y10, ki skrči bat, ki premika to gred. Ta ostane vključen, dokler se ne vklopi vhod X26, ki naznanja, da je bat skrčen in da so pištole v začetni poziciji. Po končanem pribijanju drugega žeblja se cikel pomika in pribijanja ponovi pod pogojem, da so bili izvrženi novi obdelovanci. Drugače se zadnji obdelovanci pomaknejo naprej do izhoda stroja in cikel se ustavi.

# Poglavje 6

# Zagon programa

## 6.1 Simulator

Razvoj programa za krmilnik je potekal vzporedno s testiranjem na simulatorju krmilnika in na 3D simulatorju stroja z uporabo vmesnika (slika 6.1).



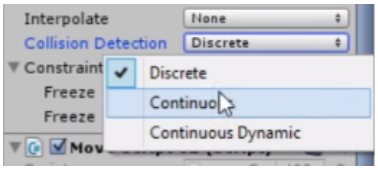

Slika 6.1: Menu z nastavitvami 3D simulatorja v programu Unity.

Vsak na novo napisani funkcijski blok smo testirali z nalaganjem na si-

mulator krmilnika, imenovan GX simulator, ki spada pod orodja programa GX Works3. Na vmesnik med GX simulatorjem in strežnikom Modbus smo povezali tudi 3D simulator stroja v programu Unity. Funkcijski blok, ki preverja stanja zalogovnikov, je prestal testiranje brez problema. V načinu monitor v programu GX Works3 se je ob ročnem vklopu bita, ki je namenjen razhroščevanju, prižgal izhod funkcijskega bloka, ki signalizira, da so zalogovniki polni. Za zagotovitev, da funkcijski blok deluje pravilno tudi ko ni obdelovancev v zalogovnikih, smo v programu Unity odstranili kocke iz enega izmed zalogovnikov. Po odstranitvi kock je funkcijski blok kazal, da zalogovniki niso polni in kateri od zalogovnikov je prazen.

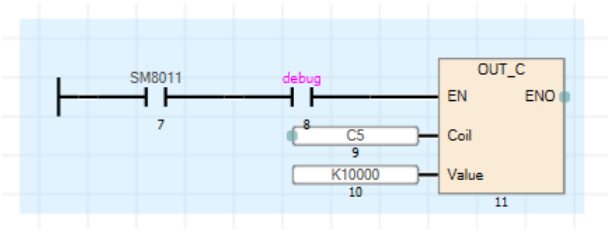
Za tem je sledil razvoj in testiranje funkcijskega bloka, ki upravlja z izmetom obdelovancev. Kako je potekal razvoj, je opisano v poglavju 5.2.4.1. Testiranje je potekalo tako, da smo na 3D simulatorju stroja najprej nastavili polne zalogovnike. Nato smo vklopili bite, ki sprožijo izmet kock. Za zanesljivejše delovanje je bil nastavljen daljši čas odprtja zalogovnika kock s spodnje strani. S tem smo zagotovili, da se ne bi bat, ki zapira ta zalogovnik, prehitro iztegnil in ujel kocko. Testiranje na simulatorju je pokazalo, da je ta čas lahko mnogo krajši, ker kocka pade dol dovolj hitro in ni potrebe po daljši odprtosti zalogovnika. Skupaj z izmetom kock je potekal tudi izmet desk. Tu se je pojavila težava v tem, da se je bat, ki potiska desko s strani, prehitro iztegnil in je potiskalo desko izpodrinilo, tako da je ta končala na vrhu potiskala, namesto da bi jo odrinilo dol z roba, na katerem sloni. Razlog za ta problem ni bil v programu, napisanem za krmilnik, pač pa v programu Unity, ker je bila komponenta togo telo, ki je določena deski in potiskalu, nastavljena tako, da je zaznavanje trkov potekalo na diskretni način, v katerem se zna včasih zgoditi, da gre kakšen objekt skozi drugega. To smo rešili s spremembo načina zaznavanja trkov, ki je bil nastavljen na neprekinjen dinamičen (slika 6.2). Po tej spremembi je tudi doziranje desk potekalo na pravilen način.

Sledilo je testiranje pomika. Ob vklopu bita, ki sproži vrtenje elektromotorja, so se sojemala v simulatorju začela premikati. Vendar pa se niso




Slika 6.2: Sprememba načina zaznavanja trkov v programu Unity.

ustavila takrat, ko bi se morala. Temu je botrovalo nedelovanje števnika, ki lahko šteje z veliko hitrostjo, saj ni podprt v programu GX simulator, ki simulira delovanje krmilnika. To smo odpravili s posebnim relejem, ki se nahaja na krmilniku na naslovu SM8011. Rele na tem naslovu se vklopi in izklopi vsakih 10 ms. Zaradi tega je bil povezan na funkcijski blok, ki povečuje vrednost števnika ob vsaki spremembi stanja.

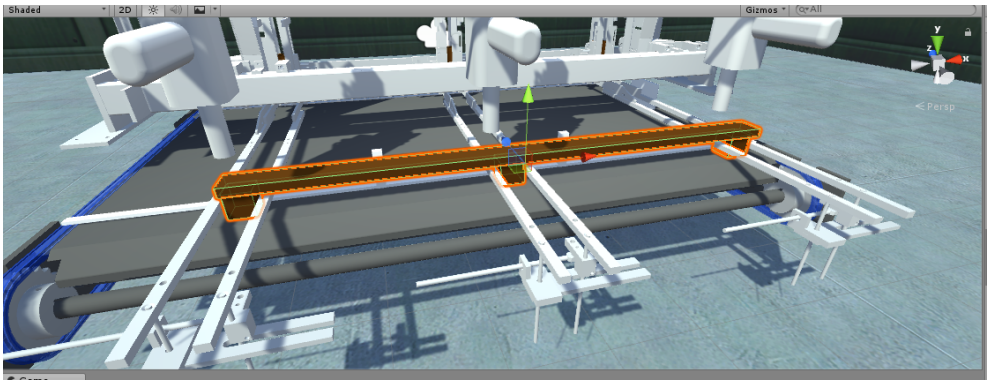



Slika 6.3: Simuliranje delovanja optičnega dajalnika impulzov.

Tako smo zagotovili način delovanja, ki je približno podoben delovanju optičnega dajalnika impulzov in prištevanju števnika z veliko hitrostjo (slika 6.3). Za pravilno ustavljanje sojemal simulatorja stroja smo določili še eno konstanto, ki predstavlja opravljeno pot na en impulz, oziroma spremembo števnika in je uporabljena pri izračunavanju razdalj. Nova konstanta je bila določena zato, ker improviziran optični dajalnik impulzov deluje počasneje kot pravi. Zatem je tudi pomik na simulatorju deloval tako, kot je bilo zamišljeno.

Obdelovanec v simulatorju stroja je na tak način prišel do pištol, kjer se je tudi ustavil. Sledilo je pribijanje žebljev, ki je delovalo brez večjih težav. V simulatorju stroja v programu Unity so po zabijanju deske in kocke postale

en objekt (slika 6.4), kar je pokazalo, da je bilo proženje pištol za žeblje uspešno. Delovanje programa krmilnika je bilo na tem mestu zadovoljivo in sledil je zagon na pravem stroju.

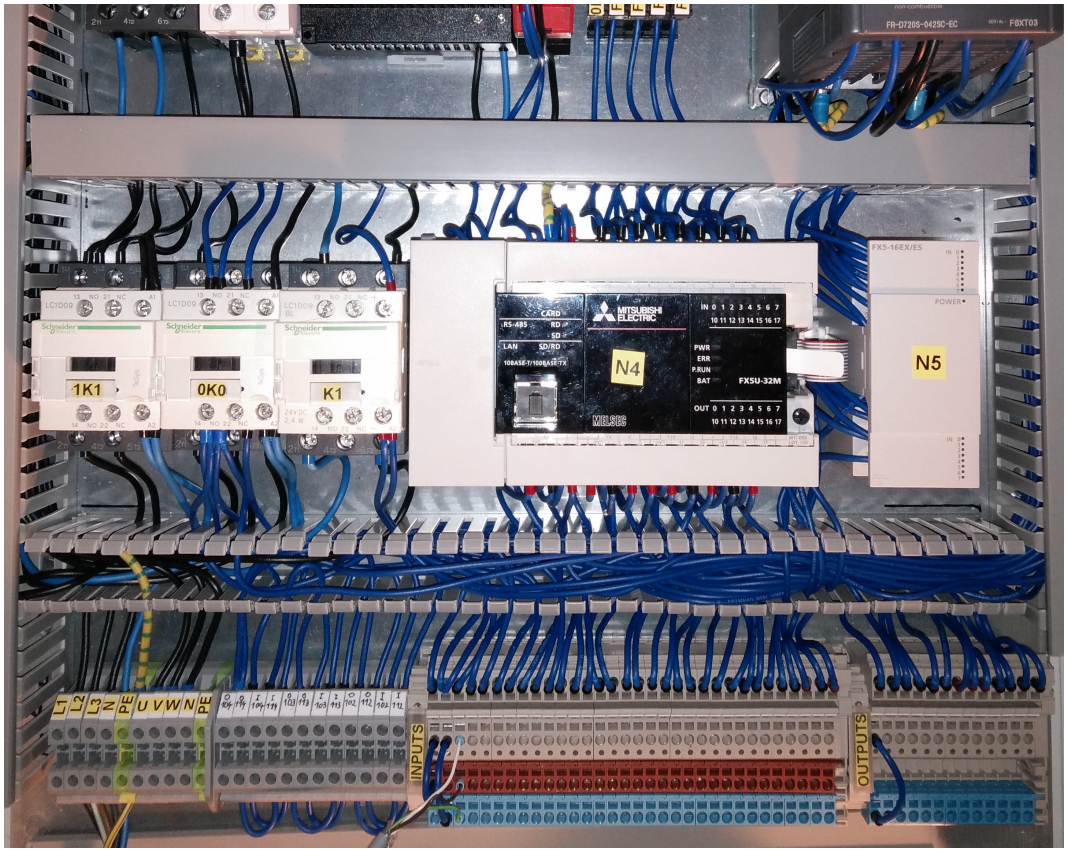



Slika 6.4: Združeni objekti kock in deske v 3D simulatorju.

## 6.2 Fizični sistem

Napisan program smo najprej naložili na krmilnik Mitsubishi FX5U, ki se nahaja v elektro omari na samem stroju (slika 6.5). Vhodi in izhodi krmilnika so vezani na sponke, ki se nahajajo na spodnji strani omare. Na te sponke pa so potem vezani ostali senzorji, aktuatorji in motorji. Za nalaganje je bilo najprej potrebno povezati računalnik na krmilnik, za kar smo uporabili omrežni kabel. Povezovanju je sledilo nastavljanje povezave. Pri nastavljanju povezave je bilo potrebno v nastavitvah operacijskega sistema zagotoviti, da ima mrežna kartica računalnika naslov IP, ki je v istem podomrežju kot je naslov IP krmilnika. Temu nastavljanju je sledilo nastavljanje povezave v programu GX Works3. Po uspešno nastavljeni povezavi smo na krmilnik prenesli napisan program. Naloženega programa nismo takoj pognali. Najprej smo preverili, če so vsi vhodi pravilno povezani na krmilnik. To smo storili tako, da smo v načinu monitor programa GX Works3 opazovali statuse posameznih vhodov, medtem ko je nekdo drug ročno prožil posamezne

senzorje.

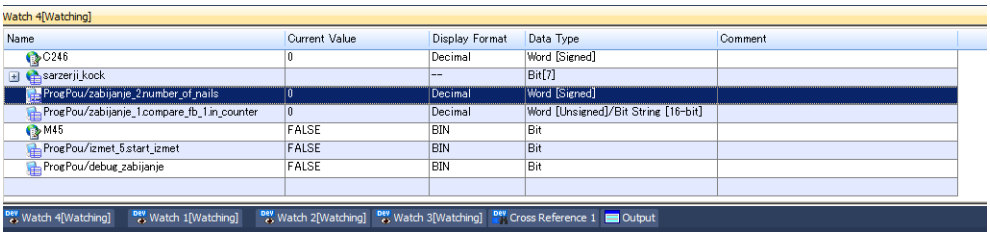



Slika 6.5: Krmilnik Mitsubishi FX5U v elektro omari s še ne priklopljenimi vhodi in izhodi.

Ker se je vse ujemalo, smo nadaljevali s testiranjem izhodov. V ročnem načinu smo na zaslonu vklapljali in izklapljali posamezne dele stroja. Začeli smo s premikanjem sojemal naprej in nazaj, kjer se je pojavila težava, ker so se sojemala premikala ravno obratno, kakor je zapisano na zaslonu. Težava je bila rešena z menjavo dveh faz kabla, ki pripelje posamezno fazo na trifazni električni motor. Po prevezavi so se sojemala premikala tako, kot se morajo. Sledilo je testiranje doziranja obdelovancev brez in z obdelovanci v zalogovnikih. Najprej smo testirali doziranje brez obdelovancev, kjer smo samo preverili, če se iztegnejo pravi bati, nato pa smo v zalogovnike vstavili

obdelovance in ponovno preizkusili posamezno doziranje. Testiranju doziranja kock je sledilo doziranje desk in oba sta delovala brez težav. Pri testiranju spuščanja in dviganja pištol ni moglo iti skoraj nič narobe, saj se je s pritiskom na gumb na zaslonu le vklapljal in izklapljal izhod, ki sproži bat za dviganje in spuščanje pištol. Ravno tako kot dviganje in spuščanje pištol je brez težav deloval tudi prečni premik pištol. Proženje posamezne pištole deluje v ročnem načinu tudi s praznim zalogovnikom žebljev in neaktivnim senzorjem, ki sporoča, da je posamezna pištola spuščena. S proženjem praznih pištol za zabijanje smo preverili, če so pravilno povezane in delujejo. Sledilo je testiranje avtomatskega cikla.

Pri testiranju avtomatskega cikla smo napolnili zalogovnike z lesenimi kockami in deskami. Temu je sledil pritisk tipke za zagon avtomatskega cikla. Ker so bili zalogovniki napolnjeni, je stroj pričel z izmetom kock in desk. Istočasno smo v programu GX Works3 opazovali posamezne vrednosti programa in funkcijskih blokov, ki so bili nastavljeni v oknu za razhroščevanje (slika 6.6).




Slika 6.6: Okno za spremljanje vrednosti programa in razhroščevanje v programu GX Works3.

Ko so bile kocke in deska izvržene, se je vklopil pomik, ki je premaknil kocke in desko do pištol za zabijanje. Zatem se je sprožilo spuščanje pištol in pribijanje žebljev. Po uspešno pribitih prvih žebljih je sledil pomik in pribijanje drugih žebljev. Medtem so bile izvržene kocke in deska, ki so bile namenjene naslednjemu ciklu. Nadaljnji pomik je premaknil desko s pribitimi kockami na izhod stroja. Še ne pribito desko in kocke pa je premaknil pod

pištole. Tako se je cikel ponavljal vse do izpraznjenja zalogovnikov. Ko so bili zalogovniki prazni, je pomik le še premaknil zadnji produkt na izhod stroja. Stroj je deloval tako, kot je pred tem deloval ob testiranju na 3D simulatorju.

# Poglavje 7

# Fizični sistem in simulator

3D Simulator smo skušali oblikovati tako, da se čim manj razlikuje od fizičnega stroja. To smo dosegli z uporabo 3D modela stroja, ki je bil ustvarjen v programu Autodesk Inventor. Iz istega 3D modela so bili narejeni načrti za sestavne dele fizičnega stroja, ki so bili uporabljeni pri razrezu kovine z računalniško vodenim plazemskim rezalnikom. Izvršne člene smo v simulatorju animirali tako, da posnemajo gibanje izvršnih členov fizičnega stroja. Razlike med 3D simulatorjem in fizičnim strojem smo zmanjševali tudi v programu Unity z zrcaljenjem fizikalnih lastnosti fizičnih sestavnih delov stroja in obdelovancev na simulirane.

Kljub temu pa so se pri zagonu pravega sistema pojavile nekatere razlike. Ena od teh je bila hitrost premikanja sojemal. Konstanta, ki določa opravljeno pot na en impulz optičnega dajalnika impulzov, se je v simulatorju razlikovala od tiste, ki smo jo izračunali za fizični sistem. Obe konstanti smo določili že prej, tako da smo morali v programu samo spremeniti, katera naj se uporabi. V 3D simulatorju smo po zagonu stroja tudi spremenili težo kock, ker smo na resničnem stroju videli, da padejo iz zalogovnikov še hitreje kot smo testirali na simulatorju. Zaradi tega smo v programu za krmilnik še dodatno zmanjšali čas odprtosti zalogovnika za doziranje kock in tako za malenkost skrajšali čas enega cikla stroja. Poziciji sojemal fizičnega in simuliranega stroja se do prvega proženja induktivnega senzorja, s poimenovanjem

takt, ne ujemata, zato smo dodali drsnik, s katerim na simuliranem stroju lahko nastavimo začetno pozicijo sojemal. To pride v upoštev pri uporabi 3D simulatorja kot dela sistema SCADA, kjer 3D simulator v živo posnema fizični stroj. Pri razvoju programa za krmilnik pa nas to ni motilo, ker nas pozicija sojemal na začetku ne zanima. Delovanje izhodov 3D simulatorja smo naknadno pogojili še z vrednostjo vhoda, ki sporoča, če je varnostni tokokrog vključen. To smo storili zato, ker na fizičnem stroju spremembe statusov izhodov krmilnika nimajo učinka na izvršne člene stroja, če varnostni tokokrog ni vključen. V času razvoja programa za krmilnik smo za 3D simulator predpostavljali, da so njegovi izvršni členi vedno pod napetostjo. To je na 3D simulatorju delovalo zato, ker še ni bil pogojen s tem vhodom. V programu GX Works3 pa smo ročno nastavili vrednost vhoda, ki sporoča, da so izhodni izvršni členi pod napetostjo, na resnično. V simulator smo kasneje dodali še polje, ki predstavlja lučko v tipki za zagon avtomatskega načina. Ta lučka mora v avtomatskem načinu utripati. Na samo funkcionalnost stroja nima vpliva, zato smo jo v simulator dodali naknadno.

Potem ko smo odpravili glavne razlike, večjih sprememb v funkcionlnosti fizičnega stroja ni bilo. Bile pa so spremembe na 3D simulatorju, ki je pridobil pri omejivah in prilagoditvi fizikalnih lastnosti, ki veljajo za fizični stroj.

3D simulator je že pred zagonom fizičnega stroja deloval dovolj dobro, da smo z njegovo pomočjo lahko razvili program za krmilnik. Po spremembah 3D simulatorja, opravljenih po zagonu fizičnega stroja, smo izboljšali tudi možnosti uporabe 3D simulatorja kot dela sistema SCADA.

3D simulator bi lahko še izboljšali s tem, da bi vanj dodali tipke in stikala, ki se nahajajo na krmilni omarici ter niso povezane na krmilnik, vendar upravljajo s krmilnimi tokokrogi. Sem spada glavno stikalo, tipke za vklop in izklop napetosti, tipka za vklop varnosti ter varnostna stop tipka. Ta del bi prišel v uporabo predvsem pri urjenju novih operaterjev na 3D simulatorju.

# Poglavje 8

# Zaključek

Z igralnim pogonom Unity smo izdelali 3D simulator stroja. Simulator smo razvili zato, da smo zmanjšali stroške zagona stroja. Z njim smo preprečili mehansko škodo, ki bi lahko nastala na stroju zaradi programskih napak. Zaradi simulatorja smo tudi porabili manj časa za delo na terenu pri samem stroju.

Na 3D simulatorju smo poganjali program, napisan za krmilnik, in tako skušali predvideti nepredvidena nezaželena stanja stroja ter poiskati morebitne napake v programu za krmilnik. Za povezovanje 3D simulatorja stroja in simulatorja krmilnika smo s pomočjo knjižnice EasyModbus razvili vmesnik, ki komunicira med tema dvema simulatorjema z uporabo protokola Modbus. Preko vmesnika se lahko povežeta tudi resnični krmilnik in 3D simulator. To nam omogoča, da lahko 3D simulator pretvorimo v samostojno namizno ali pa mobilno aplikacijo, ki preko prej razvitega vmesnika služi kot del sistema SCADA. 3D simulator lahko uporabimo tudi kot samostojno aplikacijo, namenjeno izobraževanju novih operaterjev.

Pri razvoju programa za krmilnik smo si pogledali, kako na splošno poteka razvoj posameznega programa za krmilnik v razvojnem okolju GX Works3. Lotili smo se razvoja programa za krmilnik, ki upravlja s strojem. Ustvarili smo nov projekt v razvojnem okolju GX Works3 in določili parametre projekta. Razvili smo program za upravljanje s strojem v ročnem in avto-

matskem načinu. Funkcijske bloke programa smo si organizirali tako, da se ujemajo z glavnimi koraki delovanja stroja.

Vzporedno z razvojem programa za krmilnik smo vsak na novo razvit funkcijski blok testirali na simulatorju krmilnika v razvojnem okolju GX Works3 in na 3D simulatorju stroja. To nam je zagotovilo, da pri zagonu pravega stroja nismo imeli večjih težav.

Za razvoj simulatorja smo porabili veliko časa. Vendar pa smo ta čas nadoknadili pri razvoju programa za krmilnik, saj je uporaba 3D simulatorja stroja pohitrila razvoj v programu GX Works3. Razvoj z uporabo 3D simulatorja stroja je tudi zmanjšal število napak v programu za krmilnik, saj smo napisan program sproti poganjali na 3D simulatorju pravega stroja. Tudi če je do napake prišlo, na simulatorju ni mogla povzročiti nikakršne materialne škode. Na tak način smo v celoti razvili program za krmilnik, s katerim smo nato uspešno zagnali stroj. 3D simulator s funkcijami digitalnega dvojčka nam je kot razvijalcem koristil že pri razvoju programa za krmilnik. Končni naročnik pa je dobil stroj z dodano vrednostjo, s katerim lahko izkoristi nekatere možnosti, ki jih prinašajo digitalni dvojčki in Tovarna 4.0. Preko vmesnika, razvitega za povezavo simulatorja krmilnika in 3D simulatorja, lahko na 3D simulator priključi zaledne poslovne aplikacije, ki potrebujejo podatke o proizvodnji za predvidevanje skladiščenja, transporta, logistike in popravil. Pridobljene podatke lahko uporabi pri optimizaciji delovanja in uporabe stroja ter izboljšanju produkta brez ustavljanja proizvodnega cikla.

# Literatura


[1] Peter Chipkin. Modbus for field technicians. Dosegljivo: http://www.modbusbacnet.com/includes/pdf/MODBUS_2010Nov12.pdf, 2010. [Dostopano 14. 6. 2018].

[2] Microsoft Corporation. Visual studio. Dosegljivo: https://www.visualstudio.com/vs/features/ide/, 2018. [Dostopano 16. 6. 2018].

[3] Mitsubishi Electric Corporation. Mx component version 4 operating manual. Dosegljivo: http://dl.mitsubishielectric.com/dl/fa/document/manual/plc/sh081084eng/sh081084engo.pdf, 2012. [Dostopano 16. 6. 2018].

[4] Mitsubishi Electric Corporation. Mx component version 4 programming manual. Dosegljivo: http://dl.mitsubishielectric.com/dl/fa/document/manual/plc/sh081085eng/sh081085engn.pdf, 2012. [Dostopano 16. 6. 2018].

[5] Mitsubishi Electric Corporation. Gx works3 operating manual. Dosegljivo: http://dl.mitsubishielectric.com/dl/fa/document/manual/plc/sh081215eng/sh081215engp.pdf, 2014. [Dostopano 16. 6. 2018].

[6] Mitsubishi Electric Corporation. Gx works3 programming manual. Dosegljivo: http://www.atronika.com/Mitsubishi/PLC/MITSUBISHI_manual_plc_fx5_program_design.pdf, 2014. [Dostopano 16. 6. 2018].

[7] Mitsubishi Electric Corporation. Mx component. Dosegljivo: https://eu3a.mitsubishielectric.com/fa/en/products/cnt/plceng/items/mx_components/, 2018. [Dostopano 14. 6. 2018].

[8] Bruno Dutertre. Formal modeling and analysis of the modbus protocol. In *International Conference on Critical Infrastructure Protection*, pages 189–204. Springer, 2007.

[9] Rossmann engineering 2017. Easymodbus licence. Dosegljivo: http://easymodbustcp.net/en/licenseinfo, 2015. [Dostopano 14. 6. 2018].

[10] Blender Foundation. Blender features. Dosegljivo: https://www.blender.org, 2018. [Dostopano 16. 6. 2018].

[11] Epic Games. Unreal engine frequently asked questions. Dosegljivo: https://www.unrealengine.com/en-US/faq, 2018. [Dostopano 12. 7. 2018].

[12] Real Games. Factoryi/o editions. Dosegljivo: https://factoryio.com/editions/, 2018. [Dostopano 20. 7. 2018].

[13] Real Games. Factoryi/o features. Dosegljivo: https://factoryio.com/features/, 2018. [Dostopano 20. 7. 2018].

[14] Crytek GmbH. Cryengine frequently asked questions. Dosegljivo: https://www.cryengine.com/faq, 2018. [Dostopano 12. 7. 2018].

[15] Crytek GmbH. Cryengine programming. Dosegljivo: http://docs.cryengine.com/display/CEPROG/CRYENGINE+Programming, 2018. [Dostopano 12. 7. 2018].

[16] Michael Grieves. Origins of the digital twin concept, 08 2016.

[17] Autodesk Inc. Autodesk 3ds max. Dosegljivo: https://www.autodesk.com/products/3ds-max/overview, 2018. [Dostopano 16. 6. 2018].

[18] Autodesk Inc. Autodesk inventor features. Dosegljivo: https://www.autodesk.com/products/inventor/features, 2018. [Dostopano 14. 6. 2018].

[19] Logic Design Inc. 3d worlds. Dosegljivo: https://www.plclogix.com/3d-worlds.php, 2018. [Dostopano 20. 7. 2018].

[20] Nirtec. Machines simulator. Dosegljivo: http://www.nirtec.com/index.php/machines-simulator/, 2018. [Dostopano 20. 7. 2018].

[21] Nirtec. Machines simulator sdk. Dosegljivo: http://www.nirtec.com/index.php/easyplc-sofware-development-kit/, 2018. [Dostopano 20. 7. 2018].

[22] Oracle. Digital Twins for IoT Applications. Dosegljivo: http://www.oracle.com/us/solutions/internetofthings/digital-twins-for-iot-apps-wp-3491953.pdf, 2017. [Dostopano 14. 6. 2018].

[23] Spehro Pefhany. Modbus protocol. Dosegljivo: http://www.interlog.com/~speff/usefulinfo/modbus_protocol.pdf, 2000. [Dostopano 14. 6. 2018].

[24] PLCopen. Status iec 61131-3. Dosegljivo: http://www.plcopen.org/pages/whats_new/tc1/status.htm, 2012. [Dostopano 16. 6. 2018].

[25] Michael Rüßmann, Markus Lorenz, Philipp Gerbert, Manuela Waldner, Jan Justus, Pascal Engel, and Michael Harnisch. Industry 4.0: The future of productivity and growth in manufacturing industries. *Boston Consulting Group*, 9, 2015.

[26] SimplyModbus. Frequently asked questions. Dosegljivo: http://www.simplymodbus.ca/FAQ.htm, 2017. [Dostopano 16. 6. 2018].

[27] Slant.co. Best 3d game engines. Dosegljivo: https://www.slant.co/topics/1495/~3d-game-engines, 2018. [Dostopano 10. 7. 2018].

[28] Unity Technologies. Unity Frequently asked questions. Dosegljivo: https://unity3d.com/unity/faq, 2018. [Dostopano 14. 6. 2018].

[29] Unity Technologies. Unity game engine. Dosegljivo: https://en.wikipedia.org/wiki/Unity_(game_engine), 2018. [Dostopano 14. 6. 2018].

[30] Unity Technologies. Unity User Manual Physics. Dosegljivo: https://docs.unity3d.com/Manual/PhysicsSection.html, 2018. [Dostopano 14. 6. 2018].

[31] Wikipedia. Autodesk 3ds max. Dosegljivo: https://en.wikipedia.org/wiki/Autodesk_3ds_Max, 2018. [Dostopano 16. 6. 2018].

[32] Wikipedia. Blender. Dosegljivo: https://en.wikipedia.org/wiki/Blender_(software), 2018. [Dostopano 16. 6. 2018].

[33] Wikipedia. Cryengine. Dosegljivo: https://en.wikipedia.org/wiki/CryEngine, 2018. [Dostopano 12. 7. 2018].

[34] Wikipedia. Modbus. Dosegljivo: https://en.wikipedia.org/wiki/Modbus, 2018. [Dostopano 14. 6. 2018].

[35] Wikipedia. Unreal engine. Dosegljivo: https://en.wikipedia.org/wiki/Unreal_Engine, 2018. [Dostopano 14. 7. 2018].