Мастер рад

[1. Увод 2](#_Toc1)

[2. Пројектовање и израда система за балансирање лопте на шини 4](#_Toc2)

[2.1. Пројектовање система за балансирање лопте на шини 5](#_Toc3)

[2.2. Израда пројектованих делова 8](#_Toc4)

[2.3. Склапање и повезивање система за балансирање лопте на шини 9](#_Toc5)

[3. Математички модел система за балансирање лопте на шини 11](#_Toc6)

[3.1. Једначина кретања лопте на шини 14](#_Toc7)

[3.1.1. Једначина промене нагибног угла шине 14](#_Toc8)

[3.1.2. Једначине кретања лопте на шини и преносна функција објекта 16](#_Toc9)

[3.2. Математичко модел серво мотора једносмерне струје 18](#_Toc10)

[3.2.1. Експеримент, прикупљање и обрада података 18](#_Toc11)

[3.2.1.1. Пројектовање и израда платформе за испитивање 18](#_Toc12)

[3.2.1.2. Поставка експеримента и прикупљање података 19](#_Toc13)

[3.2.1.3. Обрада података 31](#_Toc14)

[3.2.2. Идетификација система методом сиве кутије 36](#_Toc15)

[3.2.2.1. Општи математички модел електро мотора једносмерне струје са редуктором 36](#_Toc16)

[3.2.2.2. Одређивање математичког модела 39](#_Toc17)

[3.2.3. Идетификација система методом црне кутије 46](#_Toc18)

[3.2.3.1. Вишеслојна каскадна неуронска мрежа без повратне спреге 47](#_Toc19)

[3.2.3.2. Дефиниције ауторегресивних модела 49](#_Toc20)

[3.2.3.3. Идентификација система 50](#_Toc21)

[4. Пројетовање и имплементација регулатора 53](#_Toc22)

[5. Упоређивање симулације и стварног модела 53](#_Toc23)

[6. Закључак 53](#_Toc24)

[7. Додатак 55](#_Toc25)

[7.1. Додатак А – Техничка документација стандардних компоненти 55](#_Toc26)

[7.1.1. Микроконтролер – Arduino Uno 55](#_Toc27)

[7.1.2. Ултрасонични сензор – HC – SR04 57](#_Toc28)

[7.1.3. Серво мотор - TowerPro SG-5010 58](#_Toc29)

[7.1.4. Микропроцесор - Raspberry Pi 4 59](#_Toc30)

[7.1.5. Ротациони енкодер – LDP3806–600BM–5G –24C 61](#_Toc31)

[7.2. Додатак Б – Адитивна технологија производње и 3D штампа 62](#_Toc32)

[7.3. Додатак В – Софтверски алати 63](#_Toc33)

[7.3.1. CATIA 63](#_Toc34)

[7.3.2. Matlab и Simulink 63](#_Toc35)

[7.3.3. Python 63](#_Toc36)

[7.3.4. Улти макер цура 63](#_Toc37)

[7.4. Додатак Г – Помоћне скрипте 64](#_Toc38)

[7.4.1. Data\_processing\_01.py 64](#_Toc39)

[7.4.2. Data\_processing\_02.py 64](#_Toc40)

[7.4.3. Data\_processing\_03.py 65](#_Toc41)

[7.5. Додатак Д– Технички цртежи и склопни цртеж система за балансирање лопте на шини 66](#_Toc42)

[7.6. Додатак Г – Технички цртежи тест бенча за прикупљање података 66](#_Toc43)

[8. Литература 66](#_Toc44)

# Увод

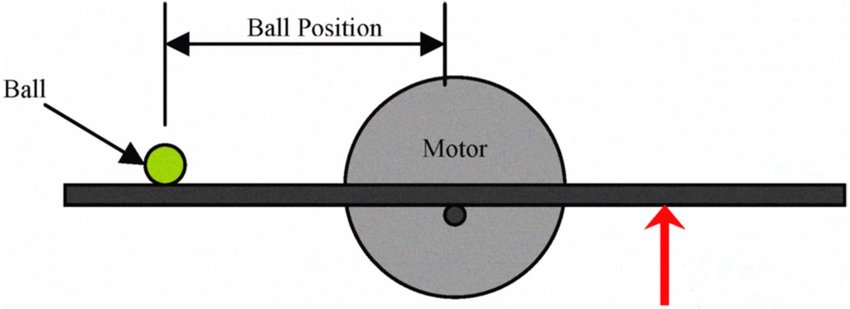
Управљање нестабилним системимам је веома важан задатак у аутоматском управаљањеу, а како се ови системи сматрају веома опасним они се испитују у лабараторијама. Систем за балансирање лопте на шини јесте нестабилан систем и може се поистоветити са реалним проблемима као што су хоризонтална стабилизација авиона током слетања или у средини турбулентног струјања.

Систем за балансирања лопте на шини је веома лак за разумевање и многе технике управљања се могу изучавати на њему, па се због тога често може наћи у лабараторијама инжињера за аутоматско управљање.

Овај систем се састоји од лопте и шине чији се угао заогретања контрилише мотором, а циљ оваквог система јесте да се лопта заустави на тачно одређеној позицији на шини. Ако је управљање лоше, лопта ће пасти са шине.

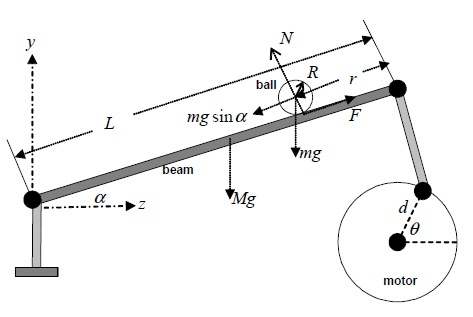
Систем балансирања лопте на шини се може видети у две конфигурације.

Прва конфигурација је приказана на слици 1.1. Са слике се види да је греда ослоњена у средини и да се ротира око своје централне осе. Оваква конфигурација поседује две предности. Прва предност је та, што је лака за израду, а друга јесте то што јој је математички модел релативно једноставан.



*Конфигурација 1 – Слика 1.1*

Друга конфигурација је нешто сложенија, а приказана је на слици 1.2. У овој конфигурацији греда има две тачке ослонца. Један крај је непокретан и греда се ротира око његове тачке, док је други крај преко система полуга спојен на мотор. Оваква конфигурација је теже за израду и има мало сложенији математички модел.



*Конфигурација 2 – Слика 1.2*

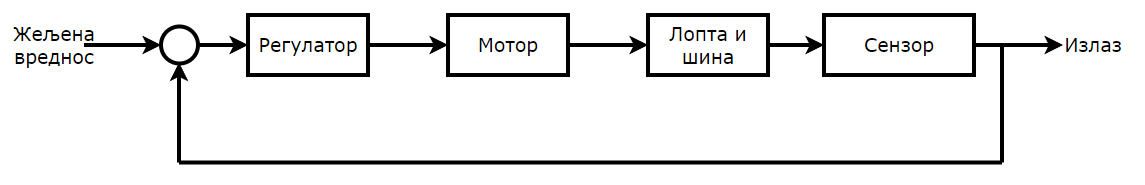
У овом раду се користи друга конфигурација, без обзира на сложеност. Такође, напоменуо бих да су следеће теме обрађене:

* Пројектовање и изрaда система за балансер лопте на шини,
* Одређивање математичког модела система за балансирање лопте на шини,
* Дизајнирање и имплементација регулатора и
* Испитивање у реалном времену.

# Пројектовање и израда система за балансирање лопте на шини

Први пробелми и задаци овог мастер рада јесу пројектовање и израда система за балансирање лопте на шини. А да би се у опште могло испројектовати систем за балансирање лопте на шини потребно је знати одговоре на питања “Шта је циљ овог система? И како у опште он ради?”.

Циљ овог система јесте да доведе лопту на одређену позицију тако што ће сам себе избацити из равнотежног стања. Да би се то боље разумело, претпоставимо да имамо шину дугачку 50cm, да се лопта налази и мирује на десном крају те шине и да је жељена позиција те исте лопте средина шине. У следећем кораку, на основу жељене вредности излаза, мотор треба да заротира излазно вратило за угао α и нагне шину на лево, односно за угао β, сензор очита нову позицију лопте, која сада креће према среди, а затим регулатор на основу разлике жељене и стварне позиције лопте формирати нови управљачки сигнал за мотор. У другом кораку мотор ће поново заротирати излазно вратило за угао α и кориговаће угао шине, а сензор очита нову позицију лопте и помоћу регулатора формирати нови сигнал за мотор. Ови кораци ће се понављати све док се лопта не доведе на жељену позицију. Све предходно објашњено може се представити и једним алгоритмом који је приказан на слици 2.1.



*Уопштнени дизајн система – Слика 2.1*

На основу уопштеног дизајна и изабране конфигурације система могу се формирати и додатна под питања.

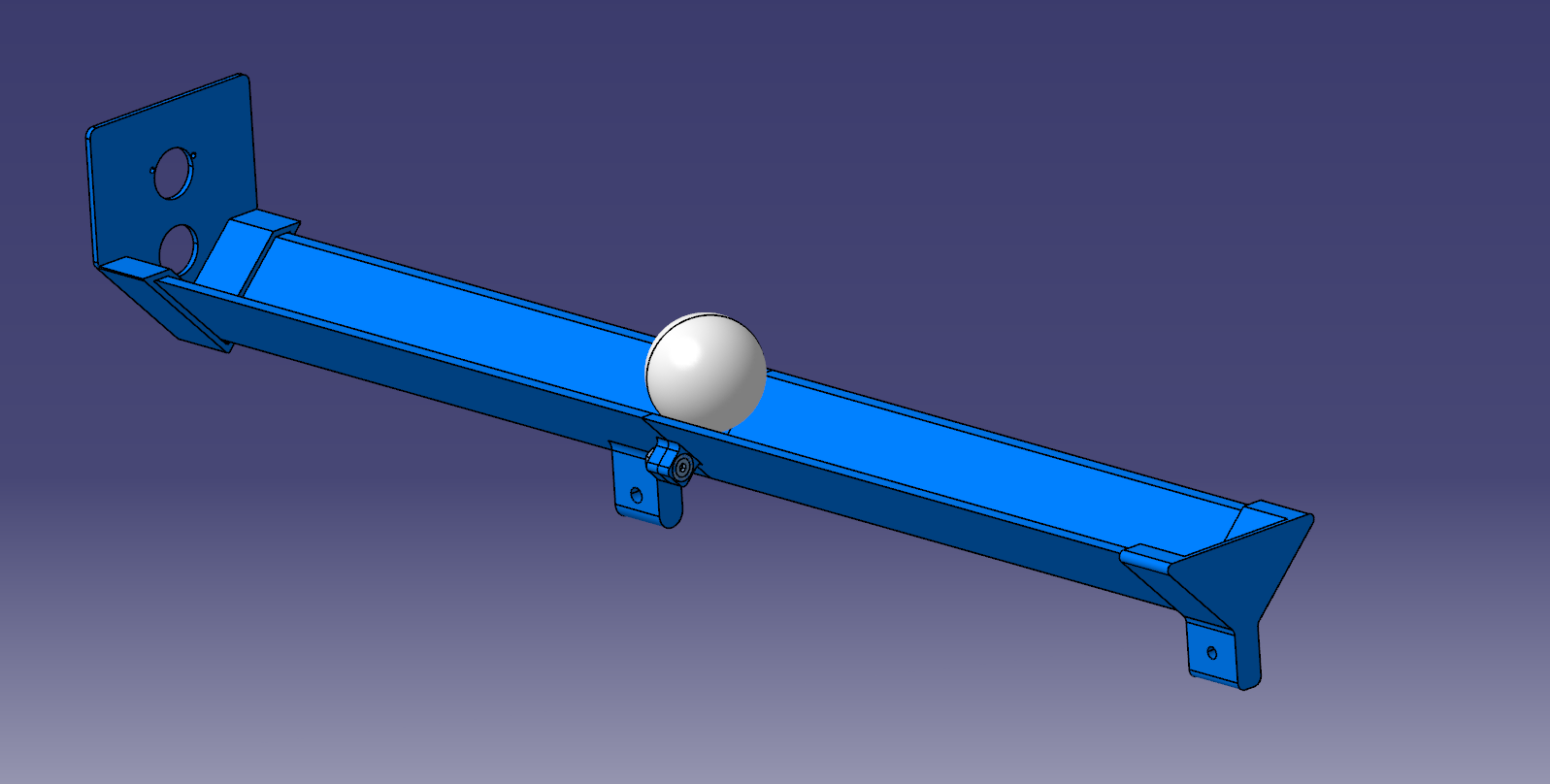
Прво питање на које треба одговорити је *“Шта се пројектује?”*, а одговор на то је: *“Систем за балансирање лопте на шини који се гони мотором преко система полуга”*. Следећа питања на која треба одговорити јесу:

* Којим сензором се мерити позиција лопте на шини? Којим мотором се гонити цео систем? Којим контролером се управља системом?
  + За мерење позиције изабран је ултрасонични сензор HC – SR04, за извршни орган серво мотор SG – 5010 и за формирање и прослеђивање управљачког сигнала микроконтролер Arduino Uno. Техничка документација и више информација о овим компонента, као и осталим компонентама које су коришћење приликом израде овог мастер рада, може се пронаћи на крају овог рада у додатку А.
* Како ће се сви испројектовани, не стандардни, делови израдити?
  + Сви делови који су испорјектовани за овај мастер рад су израђени адитивном технологијом израде, тј. на 3D штампачу.

## Пројектовање система за балансирање лопте на шини

Након што су дати одговори на сва предходна питања, слика самог система постаје јаснија и може се приступи пројектовању и моделирању система.

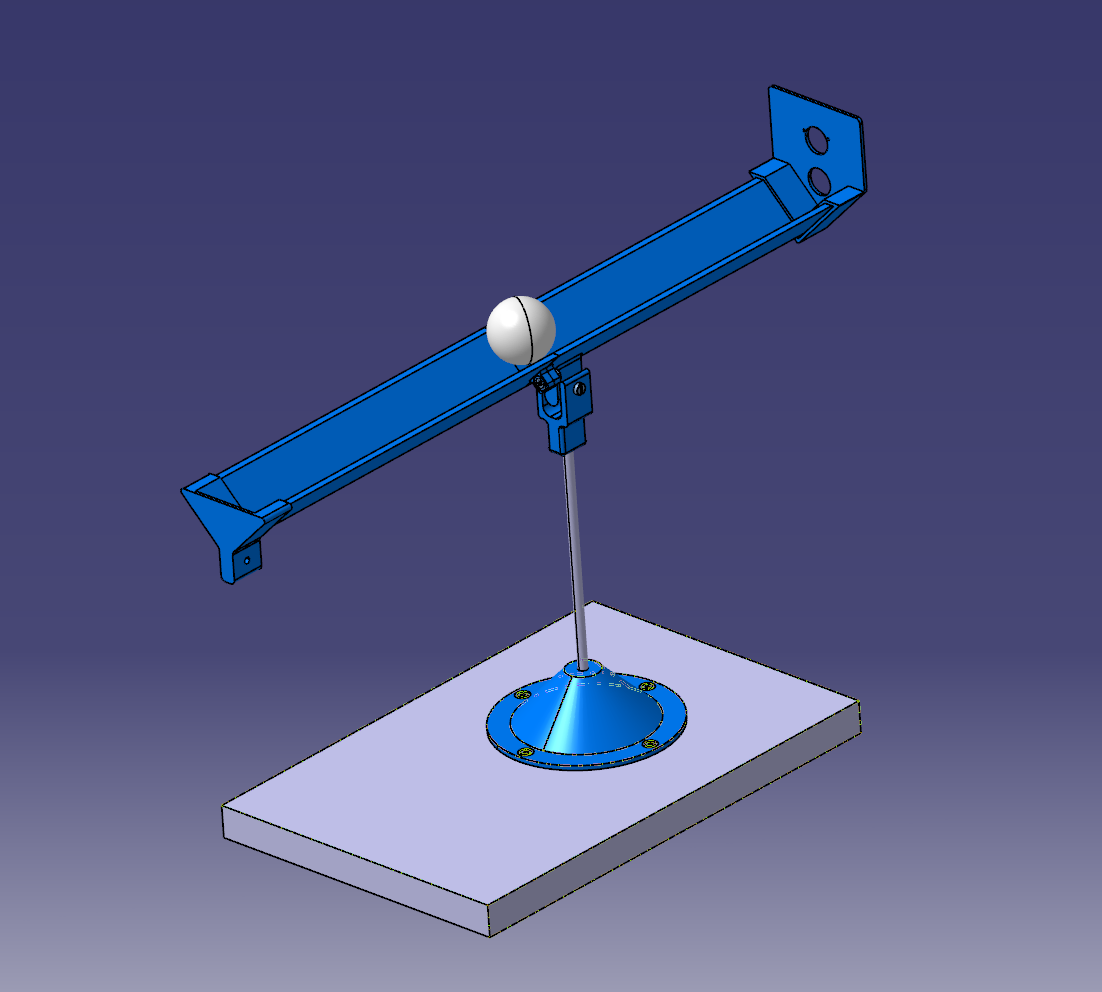
Први део који је испројектован јесте шина и она је кроз пар итерација добила следећи облик:



*Шина са лоптицом – Слика 2.2*

Са слике се види да се шина састоји из четри дела. Саме шине која се састоји из два дела, који се вијчаном везом спајају на среди, и два граничника, који су залепљена епоксидним лепком на крајевима шине. Једна половина шине поседује ушицу преко које се шина спаја са ослонцем, као и један граничник чија ушица служи за везу са мотором преко система полуга. Док други граничник је уједно и носач за ултрасонични сензор.

Следећи део који је испројектован јесте носач шине. Сам носач се састоји из три дела, ушице, навојне шипке и базе. Ова три дела су спојена или боље речено заварена. Иако на моделима се могу видети рупе за навојну шипку. Исте те рупе су напуњене уз помоћ пиштоља за лепак, а затим су крајеви навојне шипка додатно загрејани и убачена у исте те, сад већ напуњене, рупе. Оваква техника спајања је обезбедила стабилну и чврсту везу ова три дела. Такође, треба напоменути да је цео носач зашрафљен за постоље од иверице.

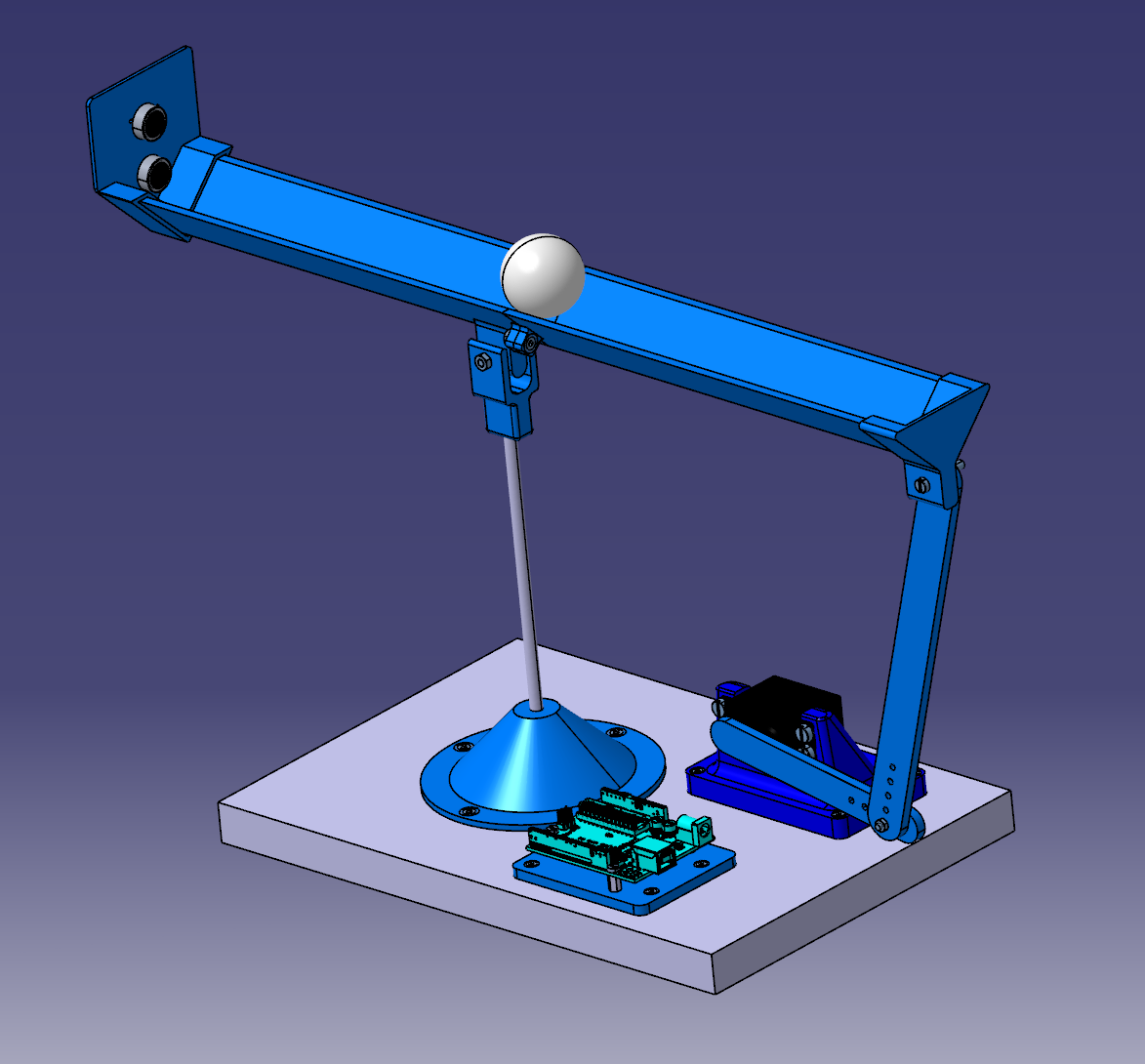


*Шина са носачем – Слика 2.3*

Након шине и носача, било је потребно испројектовати и додати везу између шине и мотора, сензор, микроконторлер и носаче за микроконтролер и серво мотор. Веза између шине и мотора се састоји из две полуге различитих дужина, где је дужа полуга повезана са шином на једном крају, а на другом крају са мањом полугом, која је поветана са серво мотором.

Финални модел система за балансирање лопте на шини приказан је на слици 2.4

Цео овај систем је измоделиран у софтверу за Компјутерски Адидитвни Дизајн (CAD) CATIA. Више информација о овом софтверском алату, као и о другим софтверским алатима који су коришћени за израду овог мастер рада, могу се пронаћи у додатку Б.



*Систем за балансирање лопте на шини – Слика 2.4*

## Израда пројектованих делова

Адитивна производња, позната и као 3D штампа или адитивна израда, је процес израде тродимензионалних објеката додавањем материјала у слојевима, уместо сечења или обликовања материјала.

У адитивној производњи, машине користе податке о CAD моделу да би додавале материјал у слојевима да би се израдио објекат, за разлику од традиционалних начина производње где машине уклањају материјал из блока или компоненте да би се добио жељени део.

Адитивна производња, у даљем тексту 3D штампа, има многе предности у односу на традиционалну производњу, укључујући већу сложеност дизајна, брзину производње, ниског трошкова производње малих серија или јединичних производа, могућност производње на захтев и низ других предности.

Због свих ових предности 3D штампач је нашао велику примену у различитим индустријама као што су авио и аутомовлиска, медицинска и многе друге индустрије, али поред индустрије 3D штампач се је постао и стандардна опрема многих лабараторија, као и радионица разних хобиста.

Постоје различите врсте 3D штампања, а неке од њих јесу фузија ласера, слој по слој (FDM) и селективна ласерска синтеризација (SLS). Која ће се врста 3D штампе користити зависи од жељеног материјала, величине објекта и других фактора.

За потребе мастер рада, сви пројектовани, не стандардни, делови су израђен на 3D штампачу Ender 3 Pro (слика 2.5) компаније Creality. Овај штампач користи технологију слој по слој (FDM), а његове основне карактеристике можете видети у табели 2.1.



*Ender 3 Pro - Слика 2.5*

|  |  |
| --- | --- |
| Назив штампача | Ender 3 Pro |
| Запремина штампања | 220x220x250mm |
| Габарити штампача | 440x410x465mm |
| Пречник млазнице | 0.4mm |
| Максимална температура млазнице | 250°C |
| Максимална температура подлоге за штампање | 110°C |
| Висина слоја | 0.1–0.4mm |
| Повезвање | USB, SD картица |
| Материјали за штампање | PLA, ABS, PETG, TPU |

*Техничке карактеристике штампача Ender 3 Pro – Табела* 2.1

Припрема делова за штампу је урађена у софтверу UltiMaker Cura. Сви делови су израђени од PLA пластике и са истом прецизношћу и за ту потребу температура подлоге за штампу је подешена на 55°C, температура млазнице на 215° и висина слоја на 0.2mm. Параметри попут дебљине зида, типа и густине испуне разликују се од дела до дела.

За израду свих делова, узимајући у обзир и шкартове, било је потребно 96 сати.

## Склапање и повезивање система за балансирање лопте на шини

Водећи се техничком документацијом из додатка В, систем је склопљен следећим редом:

1. Склапање шине;
2. Склапање носача шине;
3. Спајање шине и носача шине;
4. Позиционирање носача са шином на постоље;
5. Спајање носача са шином и постоља;
6. Повезивање шине и дуже полуге;
7. Повезивање дуже и краће полуге;
8. Спајање серво мотора и носача серво мотора;
9. Позиционирање носача са серво мотором на постоље;
10. Спајање серво мотора и краће полуге;
11. Спајање носача са серво мотором и постоља;
12. Позиционирање носача за микроконтролер на постоље;
13. Спајање носача за микроконтролер са постољем;
14. Спајање микроконтролера са носачем;
15. Повезивање свих компоненти са микроконтролером.

# Математички модел система за балансирање лопте на шини

*Дефиниција* 1 **Аутоматско управљање** је мултидисциплинарна грана инжењерства усмерена на пројектовање, анализу и имплементацију система који контролишу понашање динамичких процеса. Главни задаци ове гране јесу идентификација система, која обухвата моделирање система, процену параметара и валидацију модела, и управљање системом, чији је циљ да систем учини стабилнијим, одзивнијим и прецизнијим.

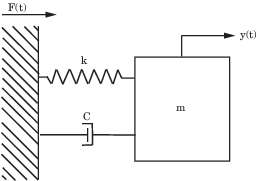
На основу дефиниције 1 лако је закљичити да су наредна два корака идентификација система и конструисање алгоритма за управљање. С обзиром да је познавање математичког модела система предуслов за конструисање алгоритма за управљање, онда је и јасно да ће у овом поглављу главни циљ бити идентификација самог система, односно одређивање његовог математичког модела.

*Дефиниција 2* **Модел физичког система** је идеализовани, замишљени, систем који задржава особине стварног система битне за његову анализу. Модел физичког система често буде

*Дефиниција 3* **Математичко модел система** је формални математички опис модела физичког система који успоставља једнозначну везу између излазних и улазних величина за произвољне промене улазних величина и за произвољне почетне услове, а исказан је помоћу математичких симбола, операција и релација.

Ако се претпостави да математички модел довољно тачно описује физички систем, тј. модел физичког система, и да представља његов веродостојан опис, онда он садржи све информације о физичким особинама система. Тада се проучавање тог физичког система могу извршити на његовом математичком моделу.

У колико су динамика система и његови параметри познати, до математичког модела система може се доћи користећи законе физике. Претпоставимо да је систем представљен на следећој слици:



*Систем пригушеног осцилаторног кретања тела – Слика 3.1*

и да су познати сви параметри овог система C, k и m. Користећи знање Механике може се закључити да овај систем претсавља пригушено осцилаторно кретање тела и да се он може описати следећом једначином:

1. .

Предходна једначина јесте математички модел система са слике која се даље може користити за анализу система. Овакав систем се у жаргону назва белом кутијом, јер су сви његови параметри познати, као и његова динамика.

Супротно од беле кутије, јесте црна кутија, односно модели чија динамика је непозната. У оваквим случајевима идентификација система, као засебна област аутоматског управљања, долази до изражаја.

Приликом одређивања математичког модела система непознате динамике, први корак јесте прикупљање информација о систему. Овај корак подразумева извођење експеримената над системом и то тако што се на систем побуђује одређеним улазним величинма, а сам излаз система мери.

Следећи корак би био одабир структуре модела. За структуру система може се одабрати преносна функција са n бројем полова и m бројем нула, диференцијална једначина, ауторегресивни модел, неуронска мрежа или било који други модел.

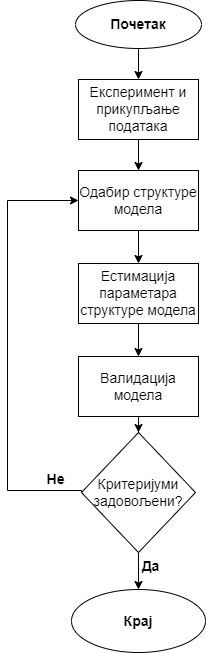
Након што су прикупљене информације о систему и одабрана структура модела, потребно је естимирати параметре и валидирати модел. Приликом валидације система треба користити нов сет података, где ће се улази тог сета искористити за пободу модела, а затим излази модела и стварног система поредити.

Када се све сумира, идентификација система се састоји од експеримента и прикупљање улазно-излазних података система, одабира структуре модела, естимације параметара и валидацију модела; али *“Како изаберати праву структуру система?”* и *“Шта радити ако након валицаије критеријуми нису задовољени?”*. Ствар јесте у томе што парвила приликом одабира стурктуре модела нема и ако након валидације модела критеријуми нису задовољени, потребно је вратити се на “почетак” и изабрати нову структуру или модификовати предходно изабрану, а за тим поново естимирати параметара и валидирати модел. Идентификација система јесте процес који захтева више итерација да би се постигли жељени резултати, али и процес у којем искуство може бити од великог значаја.

*Дефиниција 4* **Идентификација система**је методологија за конструисање математичког модела динамичког система користећи мерене улазне и излазне сигнале система.

*Дефиниција 5* **Процес идентификације система** састоји се из:

* Мерење улазних и излазних сигнала система у времеснком или фрекветном домену,
* Одабира структуре система,
* Естимације параметера за изабрану структуру модела и
* Валидације самог модела.



*Алгоритам за идентификације система – Слика 3.2*

Поред, такозваних, белих и црних кутија, постоје и сиве кутије. То су системи чија је динамика добрим делом позната, али не и параметри, или барем не сви. У оваквим случајевима се, такође, користи, предходно објашњена, метод за идентификацију система.

Претпоставимо поново да је задатак одређивање математичког модела са слике 3.1 и да је једини познати параметар маса тела. Пратећи алгортиам за идентификацију система први корак јесте прикупљање иформација о систему. Након тога потребно је одабрати конструкцију модела. Узимајући у обзир да је динамика система за конструкцију модела може се изабрати једначиа (3.1). Затим би се неком од метода минимизације естимирали непознати коефицијенти c и k и на крају, користећи нови сет улазно-излазних података, валидирао модел.

Посматрајући систем за балансирање лопте на шини се састоји од лопте на шини која је системом полуга спојена на серво мотор, где је серво мотор извршни орган, а лопта на шини објекат система. Математички модел објекта може се извести користећи законе кинематике и динамике, док је математички модел извршног органа, узимајући у обзир да ова информација није дата у техничког документацији, потребно наћи користећи један од метода за идентификацију система

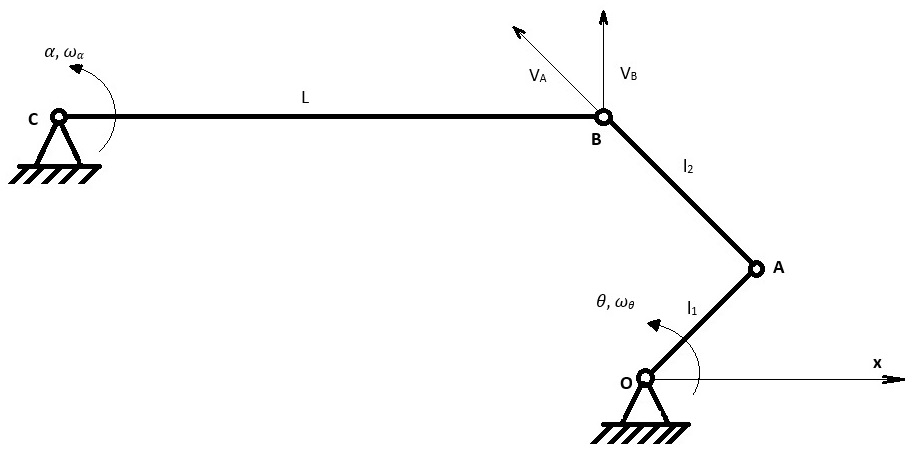
## Једначина кретања лопте на шини

Тумачећи конфигурацију система са слике 1.2 може се доћи до закључка да брзина и смер кретања лопте на шини зависи од промене нагибног угла шине, односно промене угла излазног вратила серво мотора.

### Једначина промене нагибног угла шине

Једначина промене нагибног угла шине може се наћи решамањем проблема приказонг на слици 3.3, где су:

* l1 – дужина краће полуге,
* l2 – дужина дуже полуге,
* L – растојање између ослонца шине и гоњеног краја шине,
* 𝛼 – нагибни угао шине,
* 𝜔𝛼 – угаона брзина шине,
* 𝜃 – угао излазног вратила серво мотора и
* 𝜔𝜃 – угаона брзина серво мотора.



*Систем полуга система за балансирање лопте на шини – Слика 3.3*

Брзине VA и VB су представљене у следећем облику:

1. ,
2. .

Користећи теорију сличности троуглова може се закључити следеће:

Комбиновањем једначина (3.1.1), (3.1.2) и (3.1.3) добија се:

и ако се настави са даљим разлагањем предходне једначине, може се доћи до следеће једначине:

Итеграљњем једначе (3.1.5) довија се:

Ако важе следеће претпоставке:

1. и

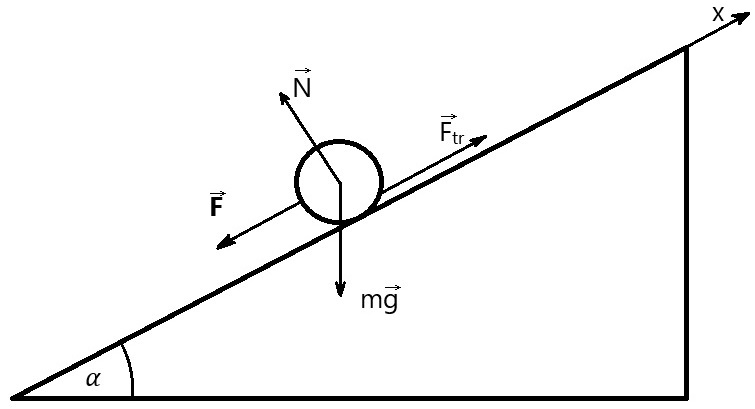
Онда из (3.1.8) и (3.1.9) следи једначина промене нагибног угла шине у зависности од промене угла вратила на серво мотору:

1. .

### Једначине кретања лопте на шини и преносна функција објекта

За потребе налажења једначине кретање лопте на шини потребно је посматрати случај када је нагибни угао шине различит од нуле (𝛼≠0). Узимајући предходну претпоставку у обзир, проблем се може поистоветити са проблемом кретање лопте по стрмој равни (слика 3.4).

*Кретање лопте по стрмој равни – Слика 3. 4*



За решавање овог проблема потребно је кренути од једначине суме свих сила на тело:

Узимајући у обзир да се за овај пројекат користити лоптицу за стони тенис, због њене мале масе може се занемарити силу потиска, а са њом и силу трења. Стога једначина (3.1.11), пројектована на x осу, иматће следећи облик:

односно, та исту једначина може се записати и у следећем облику:

1. .

Где су:

* I – момент инерције лопте,
* R – полупречник лопте,
* m – маса лопте,
* – убрзање лопте,
* g – гравитационо убрзање,
* 𝛼 – нагибни угао шине.

Ако претпоставимо да је:

тада једначину (3.1.12) можемо написати у следећем облику:

Односно, када се угао 𝛼 замени његовом релацијом из једначине (3.1.10), тада се добија:

односно:

1. .

До преносне функције објекта долази се пребацањем једначине (3.1.14) из временског домена у комплексни уз помоћ Лапласове трансформације:

1. .

Након сређивања једначине (3.1.15) добија се следећа преносна функција:

1. ,

која након замене симбола њиховим вредностима, добијамо облик:

1. .

## Математичко модел серво мотора једносмерне струје

Одређивању математичиког модела серво мотора приступљено је на два начина. Први начин јесте идетификација сиве методом сиве кутије, где је на основу општих познатих информација о серво мотрору формирана структура модела. Док за други начин искоришћена идетификација система методом црне кутије. Пратећи алгоритам за идентификацију система у овом поглављу су обрадиђене следеће тебе:

* прикупљање података помоћу експеримента,
* одабир конструкције модела,
* естимација параметара модела и
* валидацију модела.

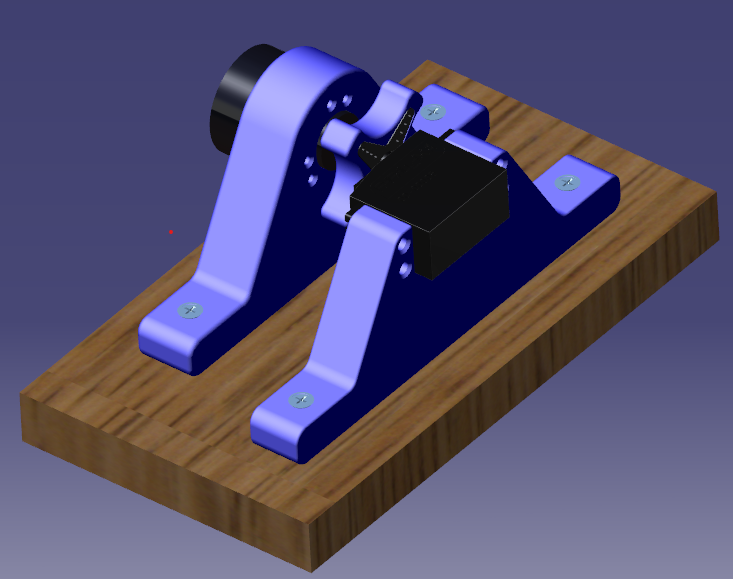
### Прикупљање података помоћу есперимента

#### Пројектовање и израда платформе за испитивање

За потребе експеримента било је потребно конструисати платформу за испитивање. До решења дођено је одговорима на следећа питања:

* Шта је циљ експеримената?
  + Циљ експеримента јесте прикупљање улазних и излазних величина система.
* Шта је улазни величина серво мотора, а шта излазна?
  + За улазну величину серво мотора је изабран жељени угао излазног вратила мотора, који је уз помоћ једноставне формуле трансформисан у ширинско модуларни пулс (стварна улазна величина), док је излаз мотора промена угла излазног вратила серво мотора.
* Како и чиме мерити промену излазне величине серво мотора?
  + Промена излазне величине серво мотора, односно промена угла излазног вратила серво мотора, мерена је ротационим енкодером LPD3806. Нешто више о овом енкодеру може се пронаћи у додатку А на крају овог рада.

Водећи се одговорима на преднохдна питања испројектована и израђена је платформа за испитивање која је поседује дрвено постоље, за које су вијцима причвршћени носачи серво мотора и ротационог енкодера. Док су вратила мотора и ротационог енкодера спрегнута наменски пројектованом спојницом. Оба носача и спојница су израђени на 3D штампачу, а финални производ је приказан на слици 3.5.



*Платформа за испитивање серво мотора – Слика 3.5*

#### Поставка експеримента и прикупљање података

У предходном делу је споменуто да је главни циљ овог експеримента прикупљање улазних и излазних величина серво мотора које ће се касније искористити за естимацију параметара модела. На основу циља може се закључити да експеримент треба да сарджи следеће кораке:

* побуду серво мотора одређеном улазном величином,
* очитавање промене излазне величине и
* чување податка о променама улазних и излазних величина.

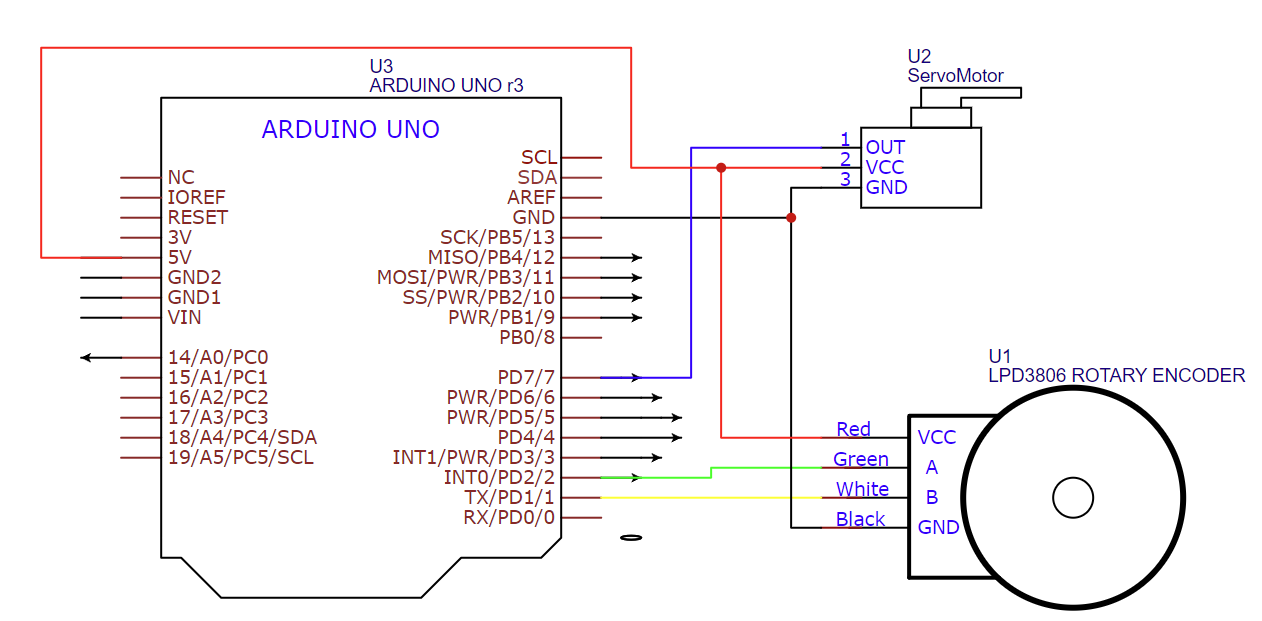
На први поглед задатак делује једноставан, међутим до финално решење дођено је тек у трећој итерацији.

**Итерација 1**

У првој итерацији, идеја је била да се и за побуду серво мотора и за очитавање промене угла излазног вратила серво мотора помоћу ротационог енкодера, користи један микроконтролер (Arduino Uno). У главној петљи се налазио код који је слао одређени ширински модуларни пулс серво мотору, док се у прекидној петљи налазио код који би очитавао вредности са енкодера.

Главна петља у је петља која ради непрекидно, извршавајући редовне задатке у секвенцијалном редоследу. Насупрот томе, прекидна петља се активира одређеним догађајима, омогућавајући контролеру да привремено заустави главну петљу и брзо одговори на хитне, временски осетљиве задатке пре него што се врати у главну петљу.

На основу предходно објашњене идеје и објашњене разлике између главне и прекидне петље, може се закључити да је систем често улазио у прекидну петљу, а да су последице тога испрекидан ширинско модуларни сигнал који је послат мотору, као и јако вибрирање мотора.



*Шема за повезивања у првој итерацији – Слика 3.6*

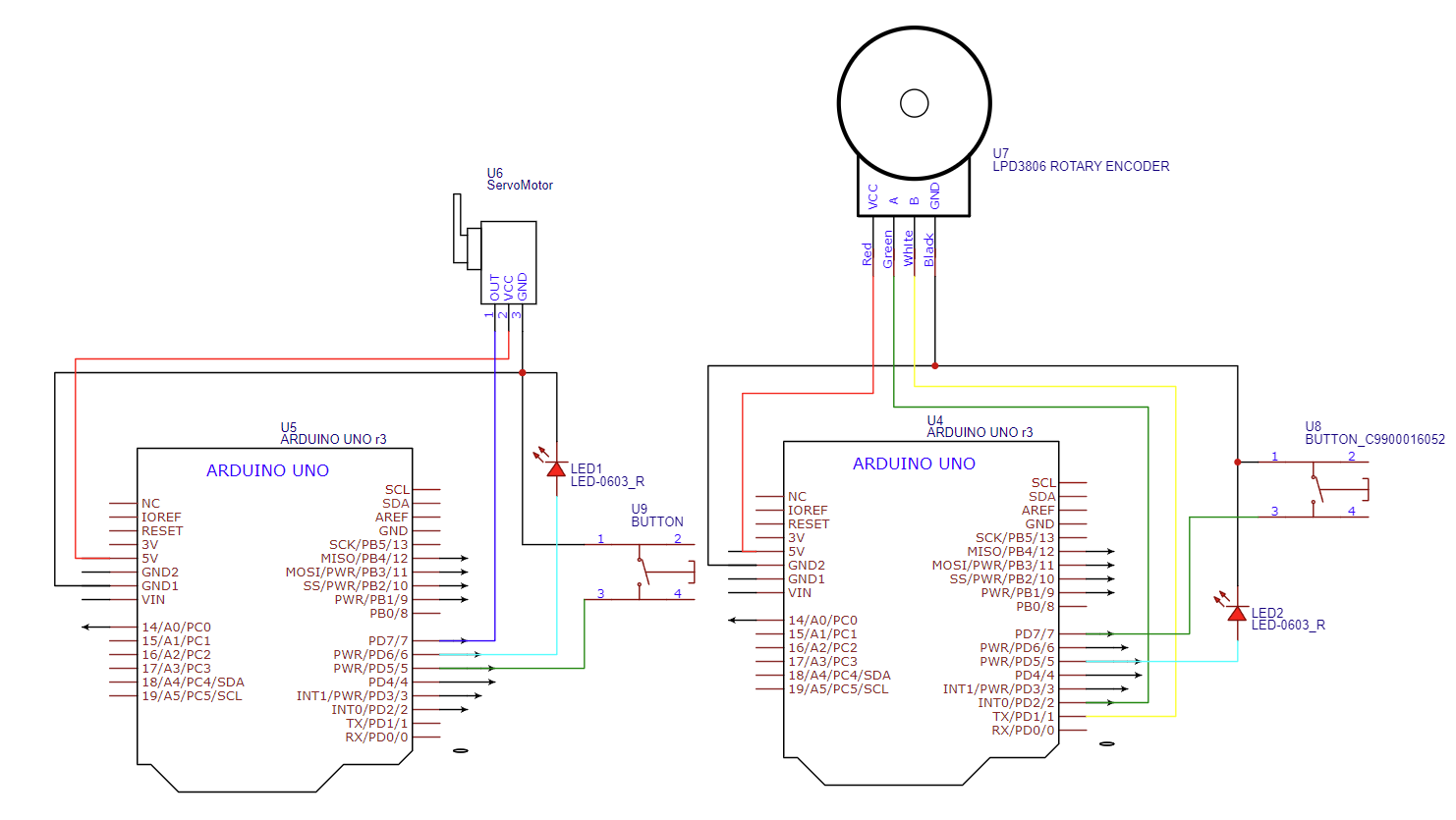
**Итерација 2**

На основу искуства из прве итерације, у другој итерацији решено је да се користе два микроконтролера (Arduino Uno). Где би један служио за побуђивање серво мотора, а други за очитавање ротационог енкодера. Поред ротационог енкодера и серво мотора, оба микроконтролера су поседовала по тастер и лед диоду (шемa повезивања приказана је на слици 3.7)

Лед диода на конролеру, који је служио за покретање серво мотора, је светлела све до притиска тастера. Након притиска тастера, лед диода би се угасила, затим би био извршен експеримент, где мотор ротира излазно вратило за одређени угао и након кратке паузе га враћа у почетни положај. Након извршеног експеримента лед диода би поново засијала.

Сличан алгортам је био и на другом микроконтролеру, али је битно напоменути да је део алгоритма за очитавање и чување вредности са енкодера и даље био у прекидној петљи. Што се тиче алгортитма у главној петљи, након притиска тастера диода би засијала, затим би се сви прикупљени подаци слали преко серијског порта на рачунар и након завршенонг слања, диода би се угасила сигнализирајући да су сви подаци послати и да је други контролер спреман за нов експеримент.

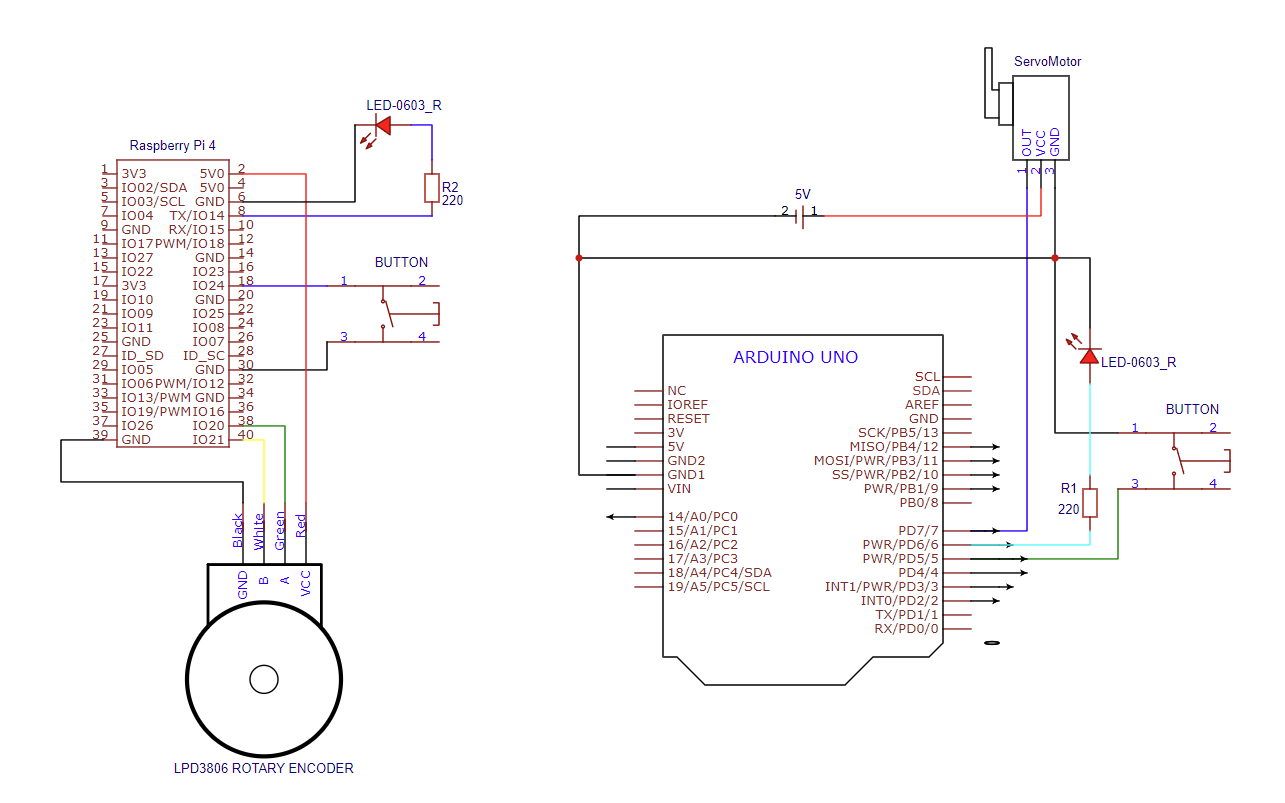
У овој итерацији пронађена су два провлема. Један од проблема јесте био нестабилан рад серво мотора услед нестабилног напајања. Док је други проблем било слање прикупљених. Због количине прикупљених података на самом почетку слања података микроконтролер би се сам од себе ресетовао.



*Шема повезивања у другој итерацији – Слика 3.7*

**Итерација 3**

У трећој итерацији, микроконторлер (ArduinoUno) који смо користили за очитавање вредности са енкодера је замењен микорпорцесором (Raspberry Pi) и проблем са напајањем серво мотора је решен. Док су алгортими за покретање серво мотора и очитавање вредности са ротационог енкодера добрим делом остали исти. У наставку су приложене шема за повезивање (слика 3.8), као и скрипте за прикупљање података и покретање серво мотора.



*Шема повезивања у тећој итерацији – Слика 3.8*

Као што је напоменуто, за мерење промене угла излазног вратила серво мотора употребљен је ротациони енкодер који је повезан на микропорцесор Raspberry Pi. Поред енкодера на овај контролер повезани су тастер и лед диода. Лед диода је индикатор која светли када је у току чување података који се чували након што се експеримент заврши и притисне тастер у датотеку json формата. Цео алгоритам је приказан на слици 3.9.

Скрипта за ове намене писана је у програмском језику Python, док су следеће блиотеке корисћене:

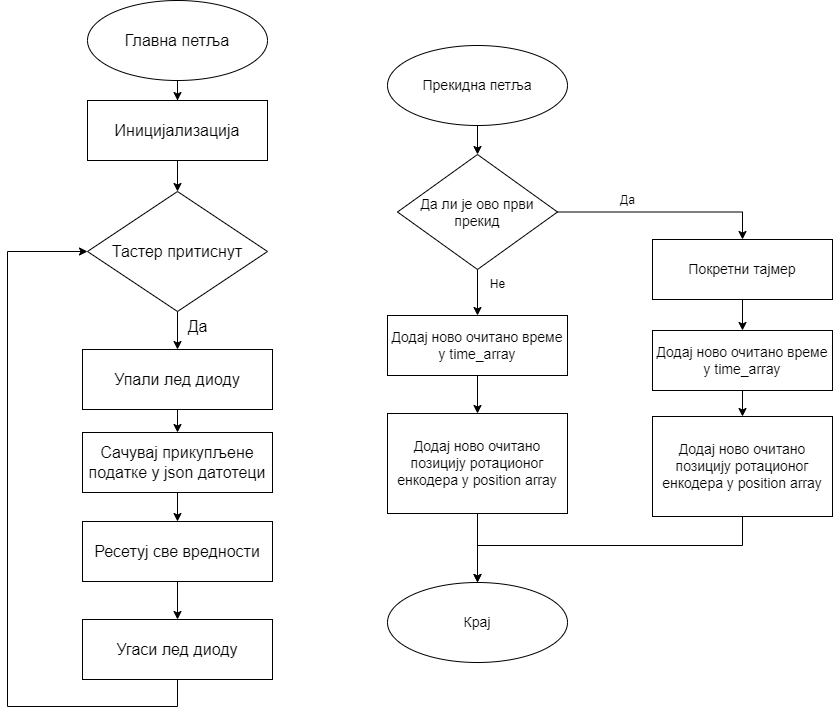
* pigpio – библиотека која омогућава коришћење и управљање улазно-излазних пинова генералне намене микропроцесоре Raspberry Pi,
* time – библиотека за очитавање времена,
* json – библиотека за рад са подацима и датотекама у json формату и
* rotary\_encoder – библиотека која омогућава очитавање вредности са ротационог енкодера.

Треба напоменути да је скрипта сачињена и од следећих функција:

* callback – функција прекидне петље која очитава вредност ротационог енкодера и повећава, односно смањује, бројач,
* collect\_data – фунција која је у директној вези са callback функцијом и служи за смештање очитаних података у привремене низове,
* save\_data – функција за чување очитаних података у датотеку json формата,
* var\_reset – функција која служи за ресетовање и брисање свих бројача и низова које користимо за привремено смештање очитаних вредности,
* debounce – функција која служи за решавање проблема са поскакивањем сигнала тастера,
* button\_press\_handler – ова функција се можда може и назвати главном функцијом главне петље, јер она у себи садржи наредбе и функције за паљење и гашење лед диоде, чување очитаних података и ресетовање привремених податка, а притом је у директној спрези са debounce функцијом.

Проблем са поскакивањам сигнала (енг. bouncing signal problem) је карактеристичан проблем за електро механичке компоненте попут тастера.

Приликом притиска тастера два метална дела долазе у контакт како би затворили електрично коло. У идеалном случају, та два дела се спајају у једном тренутку (у трену), док у стварности та два дела долазе у контакт неколико пута пре коначног спајања. Ово узрокује вишеструка окидања која контролер поистовећује са вишеструким притиском тастера. Овај ефекат се дешава и приликом отпуштања тастера.



*Алгоритам за очитавање промене угла излазног вратила серво мотора и чување очитаних података – Слика 3.9*

import pigpio

import time as t

import json

from Scripts import rotary\_encoder

BUTTON\_PIN = 23   # GPIO PIN

LED\_PIN = 14      # GPIO PIN

channel\_A = 20    # GPIO PIN

channel\_B = 21    # GPIO PIN

position = 0

position\_array = []

time\_array = []

time\_start = 0

test\_count = 0

def save\_data():

    global test\_count

    path = "/home/zelja182/Master Rad/Test\_data/Test\_"

    data = {

        "Counts": position\_array,

        "Time": time\_array,

    }

    with open(path + str(test\_count) + ".json", "w") as f:

        json.dump(data, f)

    test\_count += 1

def var\_reset():

    global position, position\_array, time\_array

    position = 0

    position\_array = []

    time\_array = []

def collect\_data():

    global time\_start

    if not position\_array:

        time\_start = t.monotonic\_ns()

        time\_array.append(0)

        position\_array.append(position)

    else:

        time\_array.append(t.monotonic\_ns() - time\_start)

        position\_array.append(position)

def callback(way):

    global position

    position += way

    collect\_data()

def debounce(func, wait\_time=0.01):

    last\_time = 0

    def debounced\_func(\*args, \*\*kwargs):

        nonlocal last\_time

        current\_time = t.time()

        if current\_time - last\_time >= wait\_time:

            last\_time = current\_time

            return func(\*args, \*\*kwargs)

    return debounced\_func

pi = pigpio.pi()

decoder = rotary\_encoder.decoder(pi, channel\_A, channel\_B, callback)

pi.set\_mode(BUTTON\_PIN, pigpio.INPUT)

pi.set\_pull\_up\_down(BUTTON\_PIN, pigpio.PUD\_UP)

pi.set\_mode(LED\_PIN, pigpio.OUTPUT)

@debounce

def button\_press\_handler():

    pi.write(LED\_PIN, 1)

    t.sleep(1)

    print(max(position\_array))

    save\_data()

    var\_reset()

    pi.write(LED\_PIN, 0)

try:

    while True:

        if not pi.read(BUTTON\_PIN):

            button\_press\_handler()

        t.sleep(0.01)  # Small sleep to avoid busy-waiting

except KeyboardInterrupt:

    print("Exiting...")

finally:

    decoder.cancel()

    pi.stop()

На микроконтролер Arduino Uno повезани су мотор, лед диода и тастер. Лед диода сија све док се не притисне тастер који ту исту диоду гаси и прелази на исвршавање експеримент из одређене групе, односно покретање test функције. Скрипта за Arduino Uno је писана у програмском језику C у наменском развојном окружењу за овај контролер (Arduino IDE). За писање ове скрипте поред стандардних функција које нуди стандардна Arduino.h библиотека, коришћене су и библиотека која нам омогућава лако управљање серво мотора (Servo.h), као и Time.h библиотека. Скрипта се састоји од следећих функција:

* setup – ова функција је задужена за иницијализацију пинова,
* loop – је функција главне петље која би гасила лед диоду приликом притиска на тастер и затим покретала један од тестова,
* test\_1 – функција за прву групу експеримената,
* test\_2 – функција за другу групу експеримената и
* test\_3 – функција за трећу групу експеримената.

Функције test\_1 и test\_2 деле исти алгоритам који је приказан на слици 3.10 (лево). Разлика између ова две групе експеримента је у почетним позицијама излазног вратила (𝜃0) и скупу углова (𝛼) који су коришћени за експерименте;

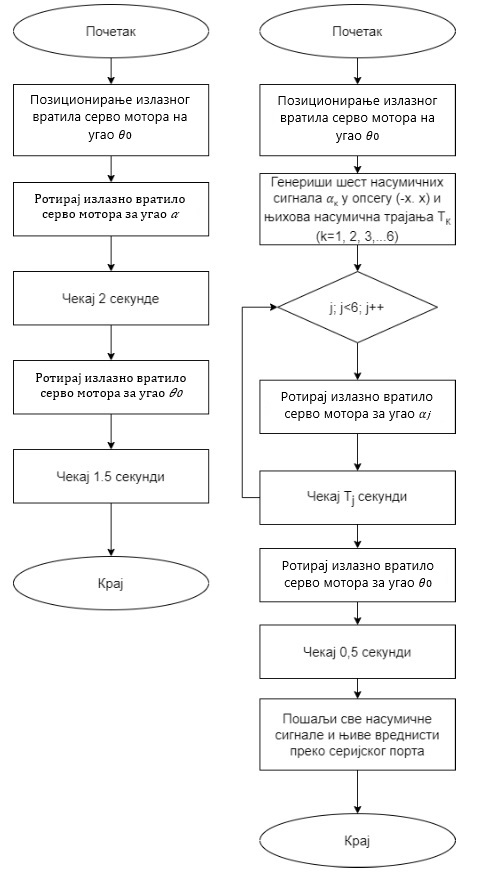
* test\_1
  + 𝜃0 = 0°
  + 𝛼 ϵ (30°, 45°, 60°, 90°)
* test\_2
  + 𝜃0 = 90°
  + 𝛼 ϵ (-60°, -45°, -30°, 30°, 45°, 60°).

Обзиром да је за потребе овог мастер рада изабран хоби серво мотор, који има позициону толерирају од ±3°, експеримент је поновљен пет пута за сваки угао 𝛼.

Функција test\_3 је мало сложенија од предходних функција. Ова функција је давала шест насумичних сигнала (углова) са насумично одређеним паузама. Након шестог сигнала, контролер би послао седми сигнал, који је за циљ имао да врати излазно вратило серво мотора у почетни положај. Користећи функцију test\_3 изведено је по десет експеримената за две различите конфигурације.

* Конфигурација 1
  + опсег насумичних углова – од -45° до 45°,
  + 𝜃0 = 90°,
* Конфигурација 2
  + опсег насумичних углова – од -30° до 30°,
  + 𝜃0 = 90°.

Алгоритам ове функције је приказан на слици 3.10 (десно).



*Алгоритми за извршавање експеримената; алгоритам лево – test\_1 и test\_2; алгоритам десно – test\_3 – Слика 3.10*

*#include <Servo.h>*

*#include "Timer.h"*

*int test\_angles\_1[4] = {30, 45, 60, 90};*

*int test\_angles\_2[6] = {-60, -45, -30, 30, 45, 60};*

*int start\_0 = 0;*

*int start\_90 = 90;*

*unsigned int time\_var = 0;*

*int i=0;*

*int j=0;*

*Servo my\_servo;*

*Timer my\_timer(MICROS);*

*void test\_1() {*

*if(i==4)   {*

*Serial.println("The end of test");*

*i++;*

*}*

*else if (i<4)   {*

*// Print Number of Test*

*Serial.print("Test\_1 no. ");*

*Serial.print(i+1);*

*Serial.print(".");*

*Serial.print(j);*

*Serial.println(" ");*

*// Print Test Angle*

*Serial.print("Test angle: ");*

*Serial.println(test\_angles\_1[i]);*

*// Execute Test*

*my\_timer.start();*

*my\_servo.write(test\_angles\_1[i]);*

*delay(2000);*

*time\_var = my\_timer.read();*

*my\_servo.write(start\_0);*

*delay(1500);*

*// Print Execution Time*

*Serial.print("Execution time: ");*

*Serial.println(time\_var);*

*my\_timer.stop();*

*// Handle Counters*

*j++;*

*if(j>=5) {*

*i++;*

*j = 0;*

*}*

*}*

*}*

*void test\_2() {*

*if(i==6) {*

*Serial.println("The end of test");*

*i++;*

*}*

*else if (i<6) {*

*int angle;*

*angle = start\_90 + test\_angles\_2[i];*

*// Print Number of Test*

*Serial.print("Test\_2 no. ");*

*Serial.print(i+1);*

*Serial.print(".");*

*Serial.print(j);*

*Serial.println(" ");*

*// Print Test Angle*

*Serial.print("Test angle: ");*

*Serial.println(test\_angles\_2[i]);*

*// Execute Test*

*my\_timer.start();*

*my\_servo.write(angle);*

*delay(2000);*

*time\_var = my\_timer.read();*

*my\_servo.write(start\_90);*

*delay(1500);*

*// Print Execution Time*

*Serial.print("Execution time: ");*

*Serial.println(time\_var);*

*my\_timer.stop();*

*// Handle Counters*

*j++;*

*if(j>=5) {*

*i++;*

*j = 0;*

*}*

*}*

*}*

*void test\_3() {*

*if(i==10)*

*{*

*Serial.println("The end of test");*

*i++;*

*}*

*else if (i<10) {*

*int random\_angles[6];*

*int random\_delay[6];*

*// Generate data for test*

*for(j=0;j<6;j++) {*

*// random\_angles[j] = start\_90 + random(-45, 45);  // Test data 3-1*

*random\_angles[j] = start\_90 + random(-30, 30);  // Test data 3-2*

*random\_delay[j] = random(10, 100) \* 10;*

*}*

*// Execute Test*

*my\_timer.start();*

*for(j=0;j<6;j++) {*

*my\_servo.write(random\_angles[j]);*

*delay(random\_delay[j]);*

*}*

*my\_servo.write(start\_90);*

*delay(500);*

*time\_var = my\_timer.read();*

*Serial.print("Test\_3 no. ");*

*Serial.print(i+1);*

*Serial.println(" ");*

*// Print Execution Time, Angles and Delays*

*for(j=0;j<6;j++) {*

*Serial.print("angle");*

*Serial.print(": ");*

*Serial.println(random\_angles[j]);*

*Serial.print("delay");*

*Serial.print(": ");*

*Serial.println(random\_delay[j]);*

*}*

*Serial.print("Execution\_time: ");*

*Serial.println(time\_var);*

*my\_timer.stop();*

*// Handle Counters*

*i++;*

*}*

*}*

*void setup() {*

*// put your setup code here, to run once:*

*Serial.begin(115200);*

*pinMode(2, INPUT\_PULLUP);  // Button Pin*

*pinMode(7, OUTPUT);        // LED Pin*

*my\_servo.attach(5);        // Servo Pin*

*// my\_servo.write(start\_0);  // Start angle for test\_1*

*my\_servo.write(start\_90);  // Start angle for test\_2 and test\_3*

*}*

*void loop() {*

*// put your main code here, to run repeatedly:*

*if(digitalRead(2)) {*

*digitalWrite(7, HIGH);*

*}*

*else {*

*delay(1500);*

*digitalWrite(7, LOW);*

*test\_3();*

*}*

*}*

#### Обрада података

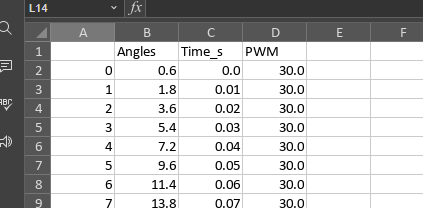
Након прикупљења података било је потребе за обрадом истих. Сви прикупљени подаци су у json формату и садрже два низа. Један низ садржи позиције у датим тренуцима, а други низ садржи временске тренутке очитаних позиција у нано секундама.

Добар део обраде података је аутоматизован скрипта написаних у Python програмском језику. Скрипта Data\_processing\_01.py је аутоматизовала обраду податка прикупљене експеримента који су покренути функцијама test\_1 и test\_2. Позиције које су дате сумом очитаних пулсева ротационог екодера су претворене у углове, време је из нано секунди претворено у секунде и додате су и вредности улазних сигнала. Узимају ћи у обзир:

* паузу од две секунде, која је карактеристична за прву и другу групу експеримената,
* да је излазно вратило серво мотора стизало до жељеног угла пре него што две секунде истекну и
* да су бележене само промене угла излазног вратила серво мотора.

Било је лако закључити да датотеке, у којима су сачуване мерене вредности, не садрже податке од тренутка када вратило достиже жељену позицију до почетног тренутка када се вратило враћа у почетни положај. Овај проблем је такође решен уз помоћ скрипте.

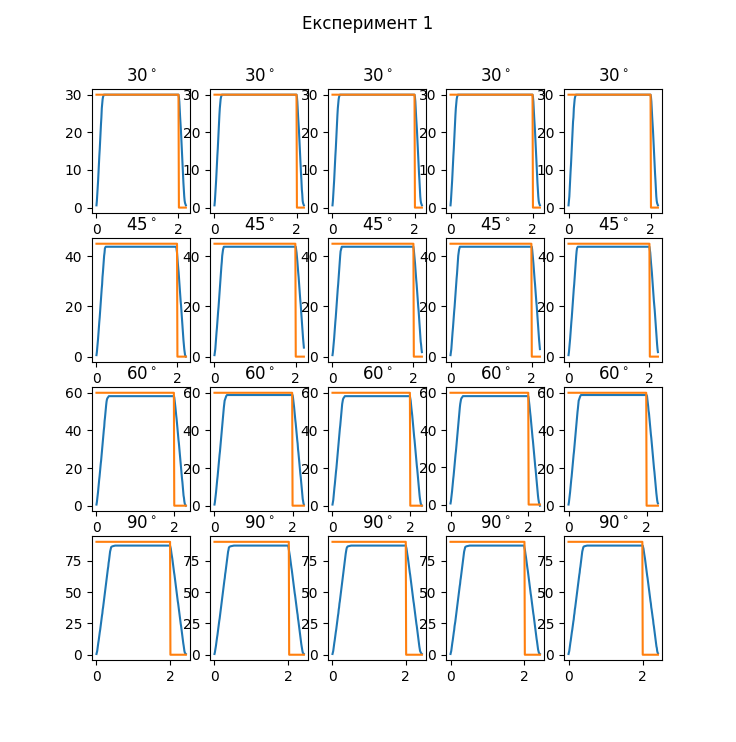
Након сређивања података помоћу скрипте, сви сетови података су прегледани и по потреби ручно исправљени.



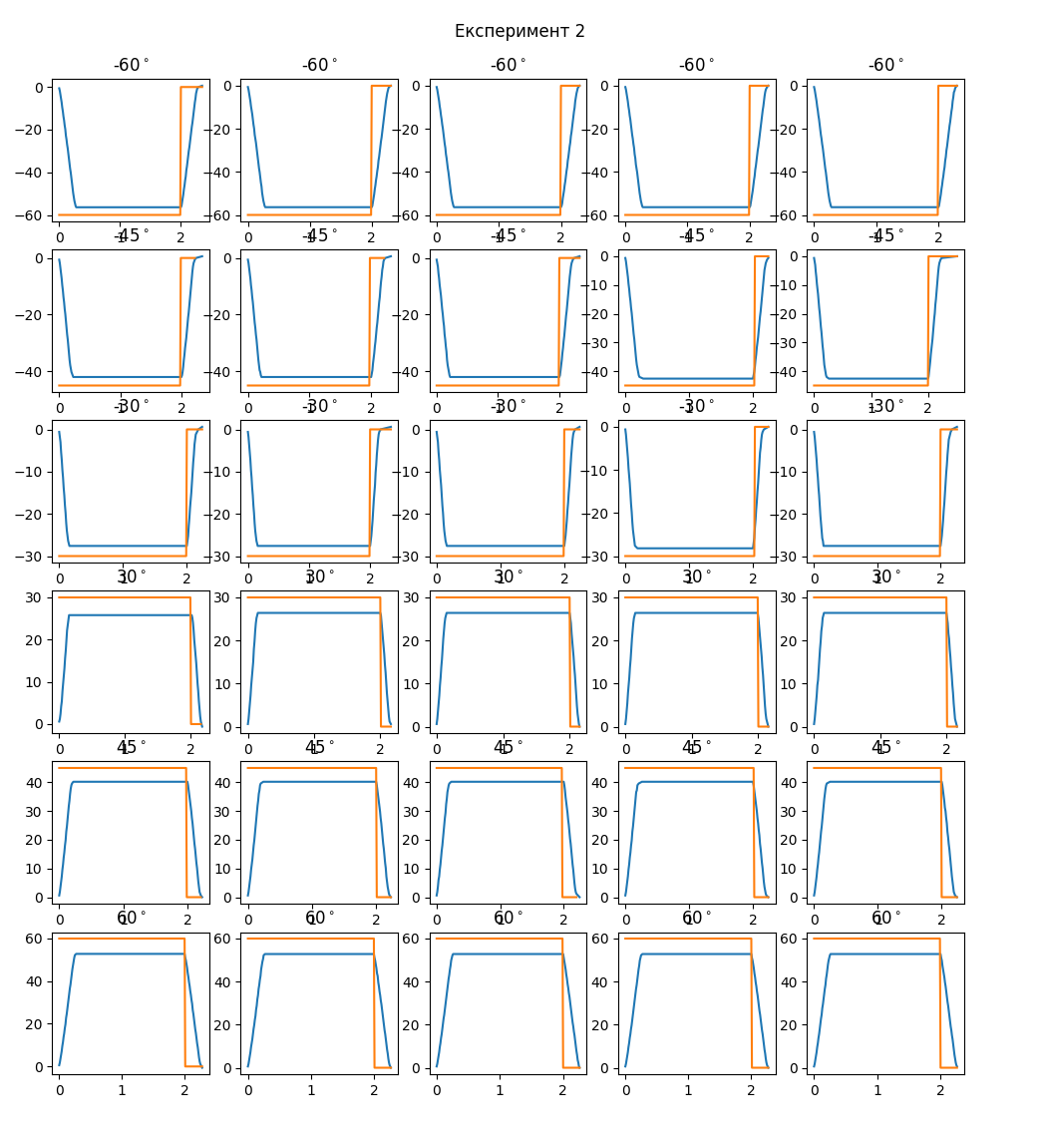
*Подаци након сређивања – Слика 3.11*

Предходно објашњен процес је примењен и на податке прикупљене извршавањем трећег експеримента (test\_3). За потребе сређивање ових података коришћене су скрипте Data\_processing\_02.py и Data\_processing\_03.py.

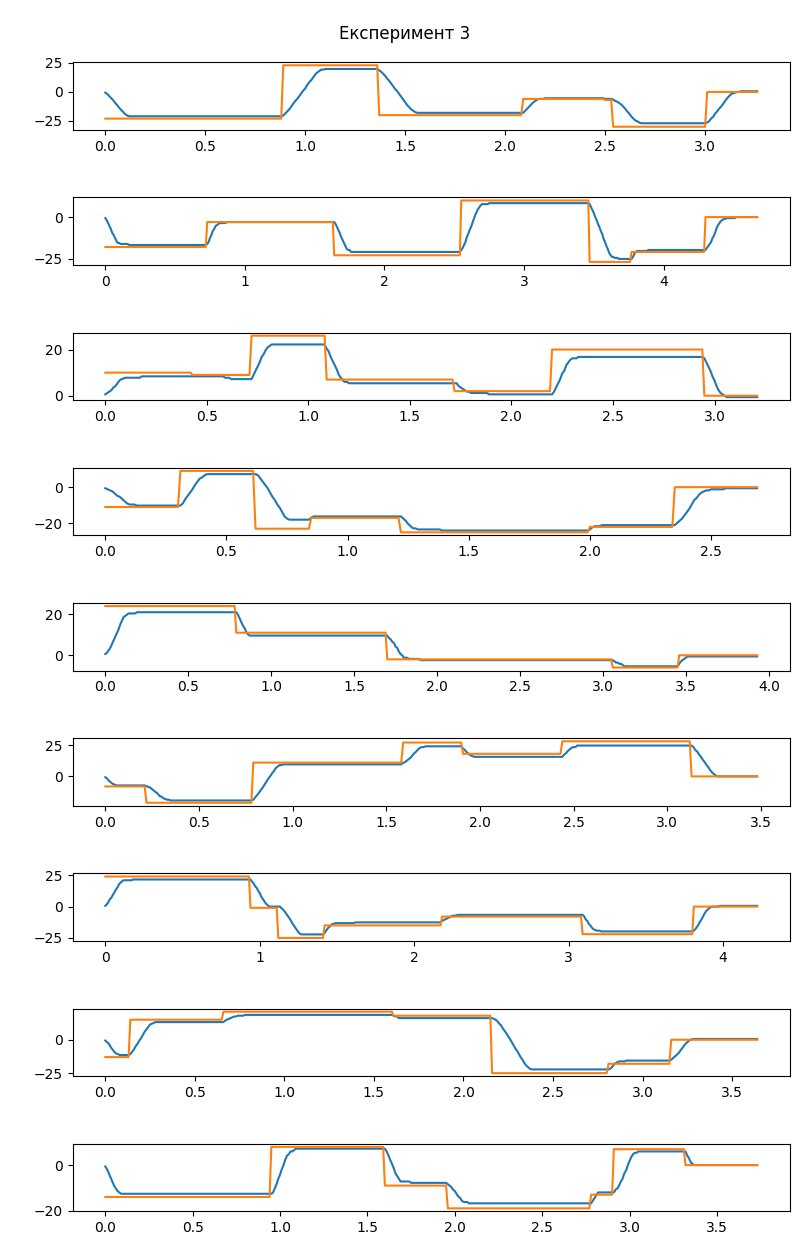
Сви обрађени подаци су графички приказани на сликама испод.



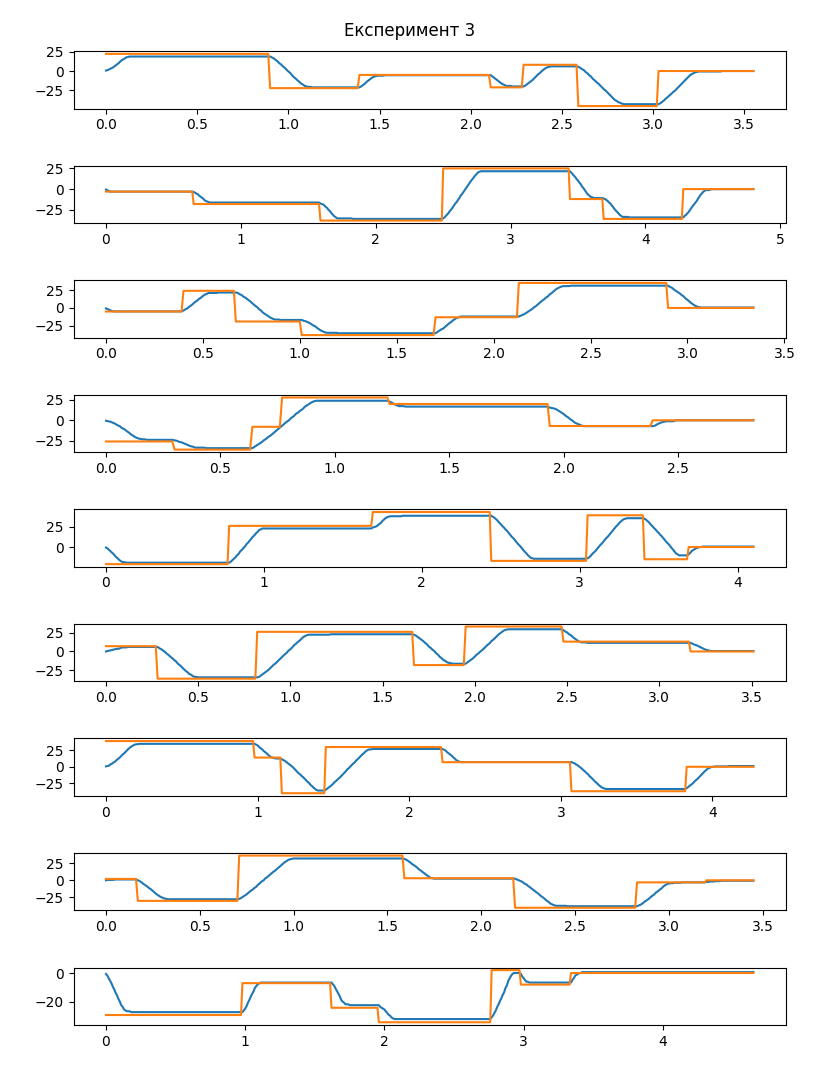
*Графички приказ података прикупљених експериметом 1 – Слика 3.12*



*Графички приказ података прикупљених експериметом 2 – Слика 3.13*



*Графички приказ података прикупљених експериметом 3 (конфигурације 1)– Слика 3.14*



*Графички приказ података прикупљених експериметом 3 (конфигурације 2)– Слика 3.15*

### Идетификација система методом сиве кутије

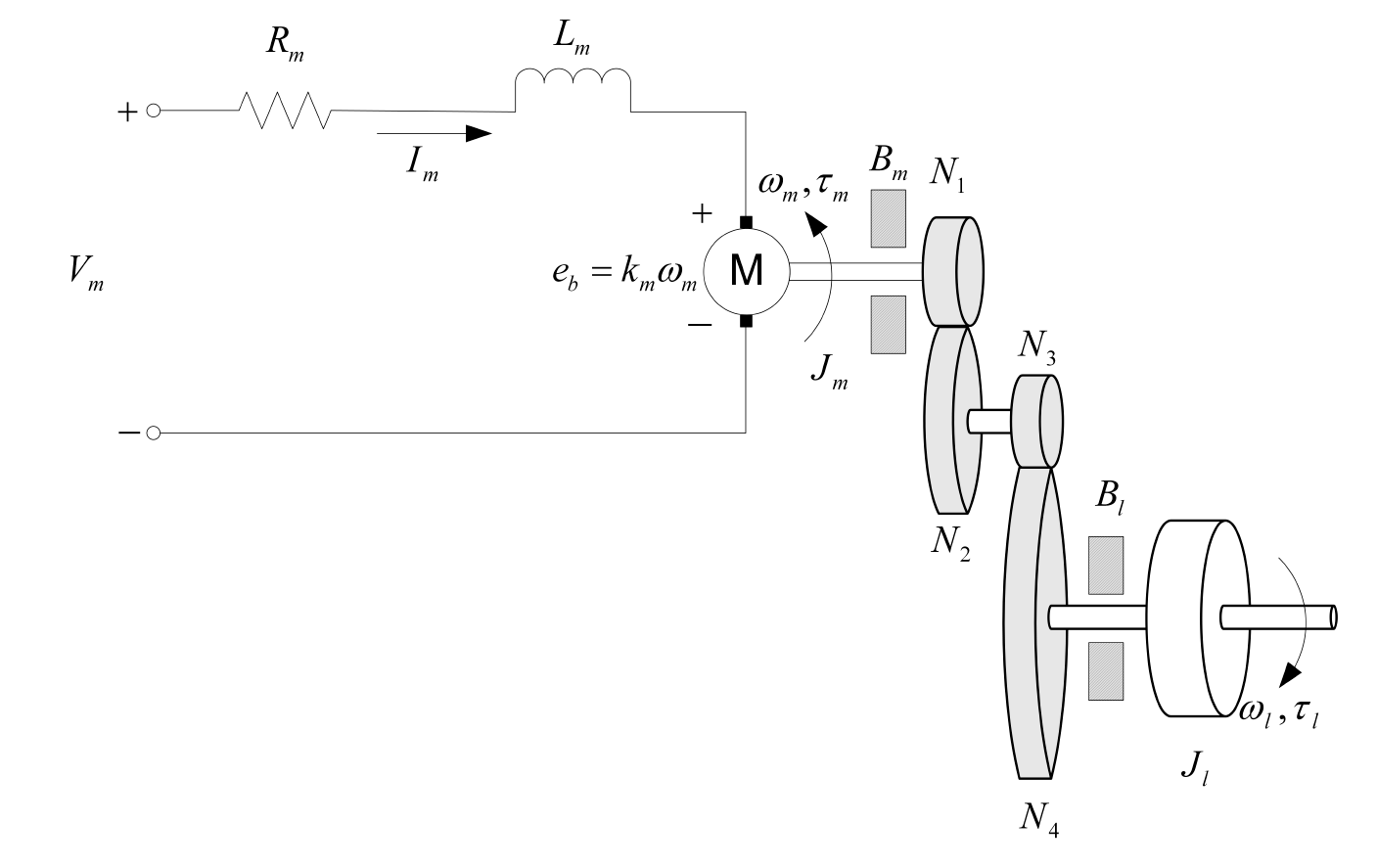
Серво мотор једносмерне струје се састоји од мотора једносмерне струје, који је директно спрегнут са редуктором, позиција излазног вратила се очитава помоћу потециометра и сам серво мотор поседује свој контролер са управљачким алгоритмом.

На основу предходно реченог може се закључити да је систем затворен и да се до структуре модела можемо доћи налажењем општег математичког модела затвореног система електро мотора једносмерне струје са редуктором.

#### Општи математички модел електро мотора једносмерне струје са редуктором

Шема електро мотора једносмерне струје са редуктором је приказана на слици 3.16, где су:

* Vm – напон,
* Im – струја,
* *Rm –* отпорност мотора,
* *Lm –* индуктивност мотора,
* *km –* константа електромоторне силе мотора,
* eb – повратни напон електромотора,
* 𝜔m – угаона брзина вратила мотора једносмерне струје,
* 𝜔l – угаона брзина излазног вратила мотора са редуктором,
* 𝜏l – обртни момент излазног вратила мотора са редуктором,
* Jx – момент инерције компоненте x, где је x = m, l,
* Bx – вискозно трење, где је x = m, l и
* Nx – број зубаца на зупчанику х, где је x = 1, 2, 3, 4.



*Шема спреге електро мотора једносмерне струје са редуктором – Слика 3.16*

**Моделовање електричног дела система**

Кренувши од другог Кирхофог закона који гласи *“Сума свих напона у контури је једнака нули”*, долази се до следеће једначине:

1. .

Узевши у обзир да је идуктивност мотора много мања од отпорнисти мотора, предходна једначина може се записати у следећем облику:



односно:

1. .

**Моделовање механичког дела система**

Кренувши од другог Њутновог закона који за ротационо кретање гласи *“Угаоно убрзање које при кретању добије једно тело сразмерно је интензитету обртног момента који делује на њега, а обрнуто сразмерно моменту инерције тела”*

1. ,

може се доћи до следећих једначина:

1. ,
2. ,

где је (3.2.4) једначина за обртни момент излазног вратила система, а (3.2.5) једначина обртног момента на излазном вратилу електро мотора. Једначина обртног момента излазног вратила система може се записати и у следећем облику:

1. ,

где су:

1. .

За потребе даљег решавања једначина (3.2.6) је усвојена у следећем облику:

1. .

Зависност промене угла излазног вратила мотора у односу на промену угла излазног вратила система мотора са редуктором може се представити следећом једначином:

1. ,

одакле следи да зависност њихових угаоних брзина може бити исказана следећом једначином:

1. .

односно на основу предходних релација може се доћи до следећег израза:

1. ,

и ако се уведу смене:

1. и
2. ,

доћи ће се до следеће једначине:

1. .

**Комбиновање електричног и механичког система**

Комбиновањем предходна два система доћиће се до општег математичког модела електро мотора једносмерне струје са редуктора.

Обртни момент мотора може се представити и следећом једначином:

1. ,

која се на основу једначине (3.2.2) можемо даље развити у облик:

односно, на основу (3.2.10):

1. .

Убацањем једначине (3.2.14) у предходну добија се:

1. .

Док сређивањем једначине (3.2.18) се добија:

односно:

1. ,

где је:

1. и
2. .

Једначина (3.2.20) представља математички модел електро мотора једносмерне струје са редуктором и применом леве Лапласове трансформације може се свести на следећи облик:

1. ,

односно:

1. .

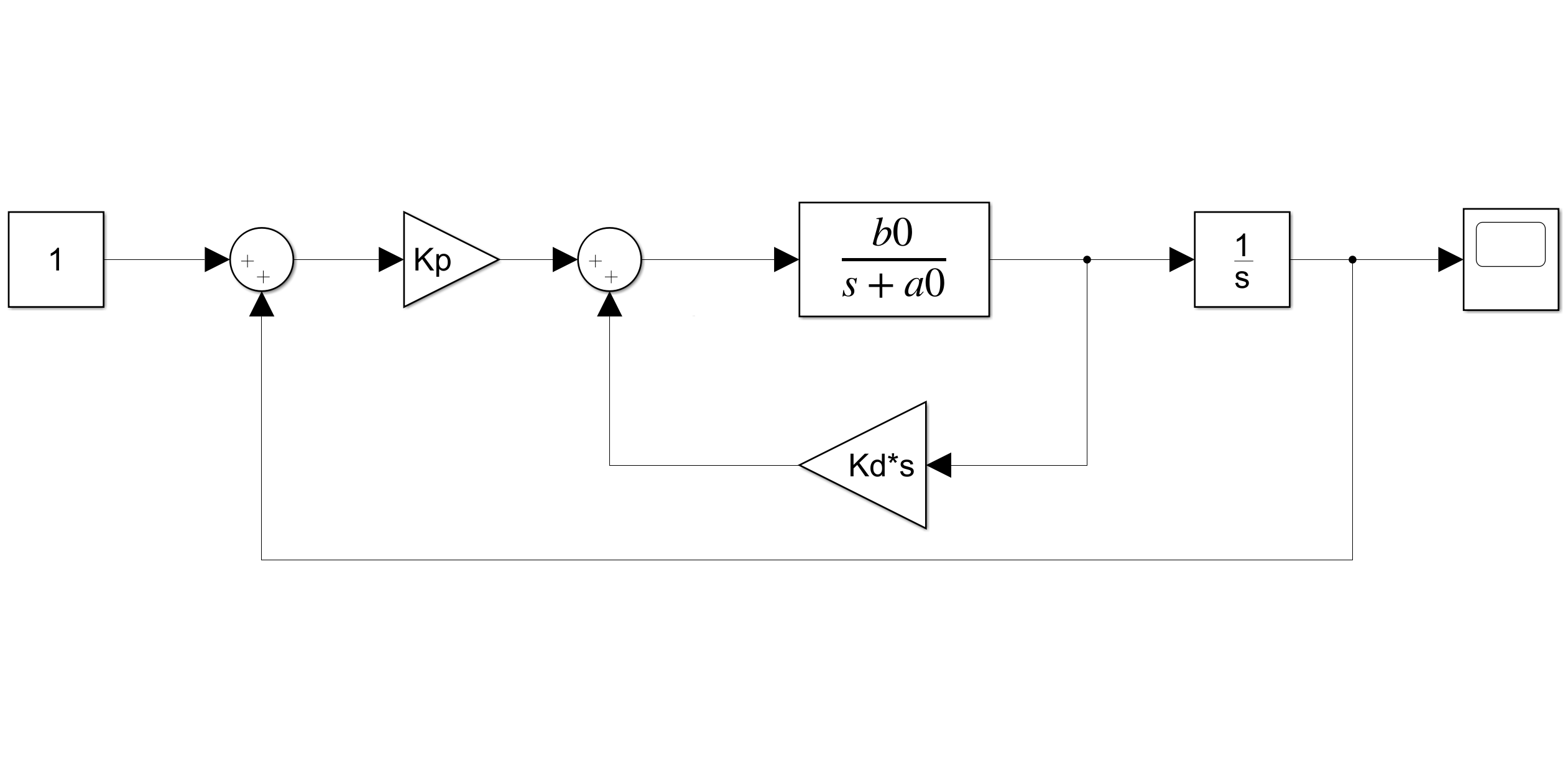
Узевши у обзир да се у свим експериментима управљало позицијом серво мотора, предходна једначина је интеграљена, чиме је добијена преносна функција преомене угла излазног вратила мотора једносмерне струје са редуктором у зависности од напона:

#### Одређивање математичког модела

Познато је да је серво мотор, сам по себи, један затворен систем. На улаз серво мотора је потребно довести одговарајући напон и он би на основу тог напона ротира излазно вратило за одређени угао. Узимајући у обзир да је за експерименте коришћена библиотека која омогућава да се за жељену вредност излаза зада жељена позиција излазног вратила серво мотора за преносну функцију је усвојен количник стварне и жељене позиције излазног вратила серво мотора:

1. .

За даљи рад, за конструкцију модела серво мотора усвојен је Simulink блок дијаграмом приказан на слици 3.17,



*Блок дијаграм серво мотора једносмерне струје (модел 1) – Слика 3.17*

односно, његова преносна функција:

1. .

Естимација параметара и први део валидације је одређен помоћу Matlab скрипте tfEstimation.m.

*Скрипта — tfEstimation.m*

close all

clear all

clc

sysData = struct();

dir\_path = "..\Ball\_And\_Beam\System\_Identification\Data\Encoder\_data\Test\_";

data\_path = "\Processed\Test\_";

% Polovi i nule

np = 2; % broj polova

nz = 0; % broj nula

file\_path = "..\Ball\_And\_Beam\System\_Identification\Data\Estimation\_data\GreyBox\Model\_1.json";

% Estimacija prenosne funkcije

for idx = 1:2

for jdx = 0:29

try

final\_path = dir\_path + int2str(idx) + data\_path + int2str(jdx) + ".csv";

T = readtable(final\_path, 'ReadVariableNames', true);

% Remove everything afeter t>= 2s and NaN values

toDelete = T.Time\_s >= 2.0;

T(toDelete, :) = [];

T = rmmissing(T);

% Estimate tf and save num and den of sys

data = iddata(T.Angles, T.PWM, 0.01);

sys = tfest(data, np, nz);

test\_no = sprintf('test\_%d\_%d', idx, jdx);

sysData.(test\_no).num = sys.Numerator;

sysData.(test\_no).den = sys.Denominator;

% Simulate and show on plot for quick validation

simTime = 0:0.01:length(T.PWM)/100-0.01;

[y, tout] = lsim(sys, T.PWM, simTime);

figure(idx\*100+jdx);

plot(tout, y, T.Time\_s, T.Angles);

grid on;

catch ME

if (strcmp(ME.identifier, 'MATLAB:readtable:OpenFailed'))

;

else

disp(ME)

end

end

end

end

% Validate data

keys = fieldnames(sysData);

for idx = 1:length(keys)

curent\_key = string(keys(idx));

MSE\_data = Validate(sysData.(curent\_key).num, sysData.(curent\_key).den);

sysData.(curent\_key).MSE = MSE\_data;

end

 jsonData = jsonencode(sysData);

fileID = fopen(file\_path, 'w');

if fileID == -1

error('Cannot open file for writing: %s', file\_path);

end

fprintf(fileID, '%s', jsonData);

fclose(fileID);

*Функција* — Validation.m

function [MSE] = Validate(num,den)

%VALIDATE Summary of this function goes here

% Detailed explanation goes here

dir\_path = "..\Ball\_And\_Beam\System\_Identification\Data\Encoder\_data\Validation\_data\Dir\_";

errorData= struct();

sys = tf(num, den);

for i = 1:2

for j = 0:8

try

final\_path = dir\_path + int2str(i) + '\Val\_Data\_' + int2str(j) + '.csv';

T = readtable(final\_path, 'ReadVariableNames', true);

[y, ] = lsim(sys, T.PWM, T.Time\_s);

test\_no = sprintf('mse\_%d\_%d', i, j);

errorData.(test\_no) = immse(T.Angles, y);

catch ME

if (strcmp(ME.identifier, 'MATLAB:readtable:OpenFailed'))

;

else

disp(ME)

end

end

end

end

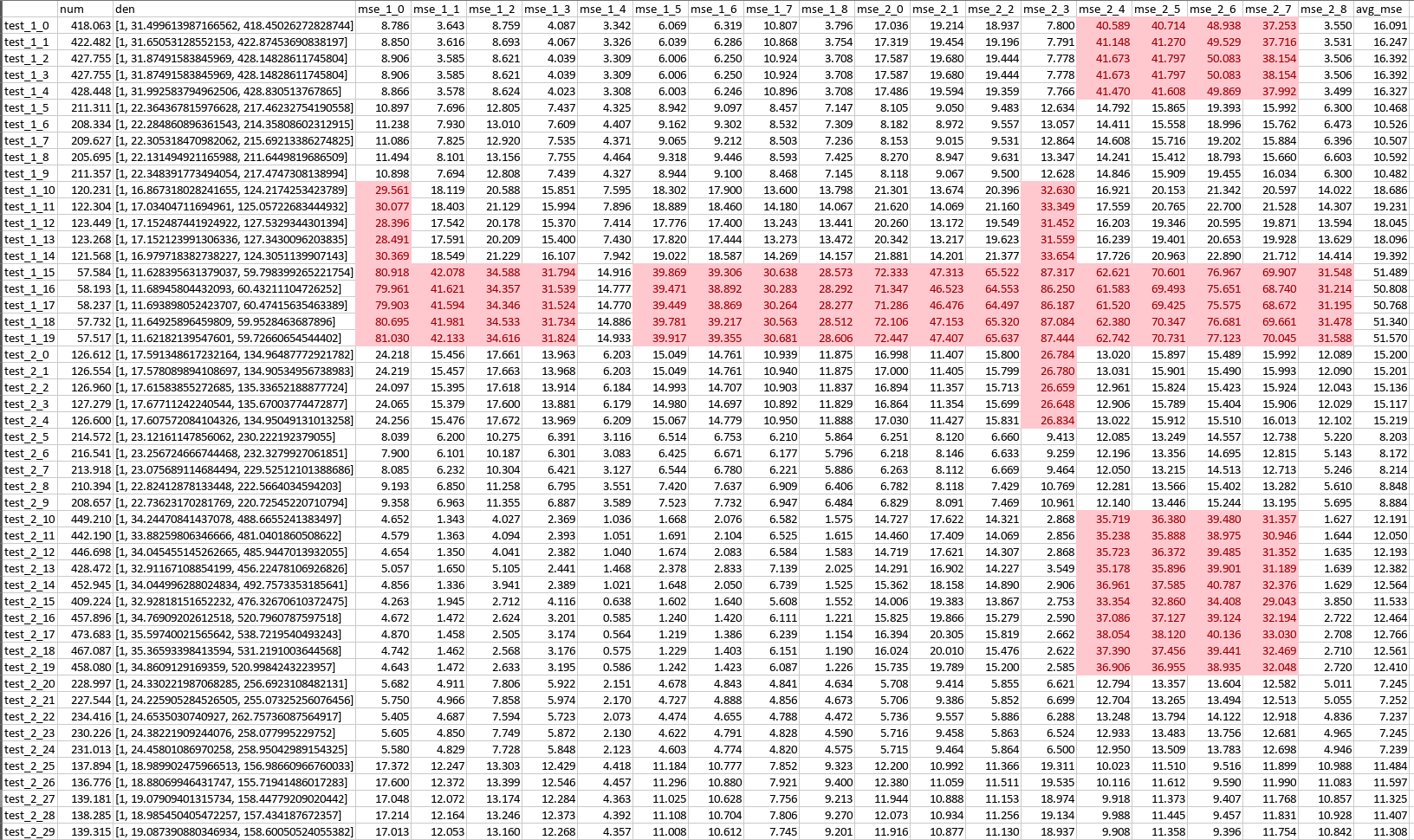
MSE = errorData;

end

Ова скрипта може да се подели у неколико делова:

* иницијализација – део на почетку скрипте где се дефинишу бројеви полова и нула преносне функције (np и nz),
* естимација параметара преносне функције – део где се на основу података сваког случаја из е групе првог и другог експеримента естимирају параметри преносне функције,
* валидација модела – циљ овог дела јесте поређење излаза модела са излазима сваког случаја треће групе експеримента. Излаз ове функције је низ средњих квадратних грешака.
* чување података – овај део скрипте је задужен за креирање датотеке json формата и чување естимираних вредности модела и низова средњих квадратних грешака у истој датотеци.

За даљу анализу предходна сачувана датотека је трансформисана у табелу која је приказана на слици 3.18.



*Табела естимираних вредности и средњих квадратних грешака – Слика 3.18*

Анализирањем податка из табеле може се закључити:

* да су случајеви, прве групе експерименте, где је Xиж = 90° лоши;
* да се случајеви, где је Xиж = ±30°, нису добро показали за половину случајева трећег групе експеримента друге конфигурације;
* да две трећине случајева има просечну средњу квадратну грешку мању од 13%.

За даљу анализу је усвојена преносна функција (3.2.28) која има најмању просечну средњу квадратну грешку:

и искоришћена је Matlab скрипта tfValidation.m. Излази предходно поменуте скрипте су:

* графички приказ излаза модела и стварног система
* и графика ауто корелације и унакрсне корелације резидуа.

*Скрипта — tfValidation.m*

close all

clear all

clc

% Generate transferfunction

num = [234.416];

den = [1, 24.654, 262.757];

sys = tf(num, den);

dir\_path = "..\Ball\_And\_Beam\System\_Identification\Data\Encoder\_data\Validation\_data\Dir\_";

for i = 1:2

for j = 0:8

try

final\_path = dir\_path + int2str(i) + '\Val\_Data\_' + int2str(j) + '.csv';

T = readtable(final\_path, 'ReadVariableNames', true);

data = iddata(T.Angles, T.PWM, 0.01);

fig\_no = i\*10+j;

figure(fig\_no);

compare(data, sys);

fig\_no\_2 = 100+i\*10+j;

figure(fig\_no\_2);

resid(sys, data);

catch ME

if (strcmp(ME.identifier, 'MATLAB:readtable:OpenFailed'))

;

else

disp(ME)

end

end

end

end

*Дефиниција 6* **Анализа резидуа** је статистичка техника која се користи у идентификацији система како би се проценила тачност и квалитет модела анализом разлике између стварног излаза система и излаза предвиђеног моделом (3.2.29).

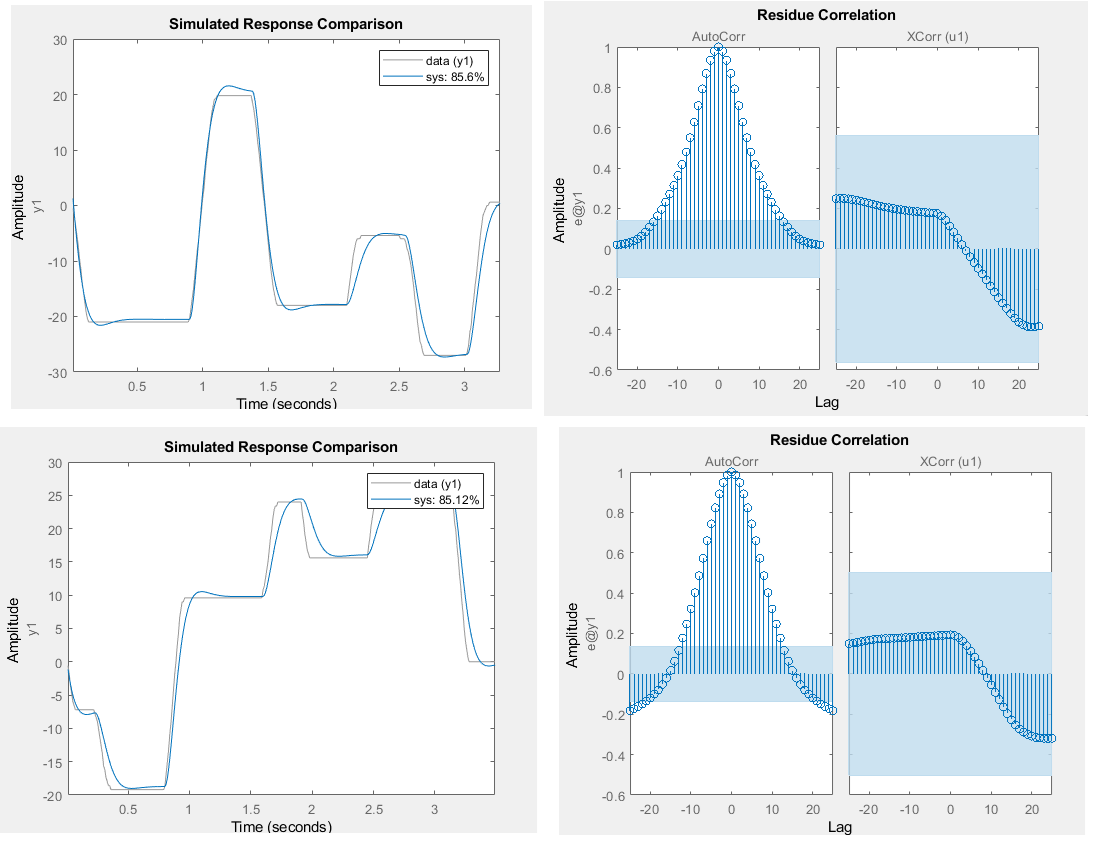
Ове разлике, познате као резидуи, представљају необјашњени део излаза који модел не хвата. Анализа резидуа помаже да се утврди да ли модел добро захвата основну динамику система или има систематске грешке.

*Дефиниција* **7 Аутокорелација** **резидуа** се односи на корелацију резидуа са самим собом на различитим временским интервалима. Ако постоји значајна аутокорелација, то сугерише да модел не хвата у потпуности понашање система.

*Дефиниција 8* **Унакрсна корелација** **резидуа** проверава да ли су резидуи у корелацији са прошлим улазима. Ако су резидуи у корелацији са улазом, то сугерише да модел не успева да ухвати неку важну динамику система под утицајем улаза.

Упоређивањем стварног излаза и излаза модела може се закључити да математички модел хоби серво мотора, описан преносном функцијом (3.2.28), ефикасно хвата динамику стварног система. Док се на основу анализе резидуа може закључити:

* на основу ауто корелације – да моделу не покрива у потпуности динамику система,
* на основу унакрсне корелације – да модел добро хвата динамику система под утицајем улаза.



*Графички приказ стварног излаза и излаза модела(лево); графици аутокорелације и унакрсне корелације резидуа (десно) — Модел 1 – Слика 3.19*

На основу предходне анализа и узимајући у обзир да се за конструкцију модела користио линеарни модел дат преносном функцијом (3.2.27) може се закључити да је достигнут максимум и да је за налажење бољег модела потребно изабрати други тип конструкције.

### Идетификација система методом црне кутије

У овом делу покушаћемо да одредимо математички модел серво мотора користећи метод црне кутије. Водећи се научним радовима:

* Deep Learning and System Identification, чији су аутори Lennart Ljung, Carl Andersson, Koen Tiels и Thomas B. Schon и
* Cascade Forward Neural Network for Time Series Prediction, чики су аутори Budi Warsito, Rukun Santoso, Suparti1, Hasbi Yasin.

одлучено је да се за идентификацију система користи вишеслојна каскадна неуроснка мрежа у комбинацији са нелинеарнинм ауторегресивним моделом са егзогеним улазима (NARX).

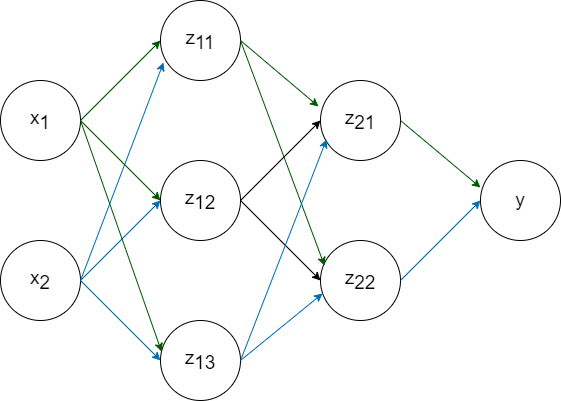
#### Вишеслојна каскадна неуронска мрежа без повратне спреге

Како би лакше објаснили вишеслојну каскадну неуронску мрежу без повратне спреге кренимо од “обичне” вишеслојне неурнске мреже без повратне спреге.

Због јендоставности објашњења претпоставимо да имамо два улата, један излаз и два скривена слоја како на слици 3.23, такође претпоставимо да сви неурони имају исту активациону функцију *f* и усвојимо следећу нотацију:

* xp — p-ти улаз неуронске мреже,
* zij — излаз неурона у скривеном слоју i,
* wijk— тежинa j-тог неурона у i-том скривеном слоју и
* y — излаз неуронске мреже,

где су i, j, k, p = 1, 2, 3, 4, ..., n.



*Вишеслојна неуронска мрежа без повратне спреге – Слика 3.20*

Излаз једног неурна првог слоја је дат следећом једначином:

1. .

Узимајући у обзир да први слој садржи три неурона можемо представити излаз целог слоја у следећем облику:

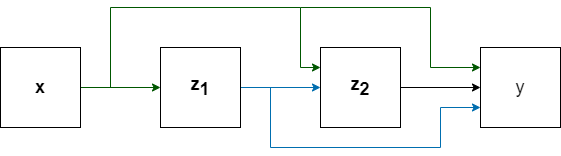
1. .

Други и излазни слој прате исти тренд и могу се представити следећим једначинама:

1. и
2. .

Вишеслојна каскадна неуронска мрежа без повратне спреге се разликује од предходно објашњене мреже у начину на који су слојеви повезани. Код каскадне мреже слој n+2 (n = 0, 1, 2, 3, ...) није повезан само са предходним слојем, већ са свим предходним слојевима.

Претпоставимо поново да наша мрежа има два улаза, један излаз и два скривена слоја као на слици 3.20, али усвојимо топологију каскадне неуронске мреже, слика 3.21. Такође, претпоставимо да сви неурони имају исту активациону функцију *f* и задржимо исту нотацију.



*Вишеслојна каскадна неуронска мрежа без повратне спреге – Слика 3.21*

Јасно је да је и у овом случају излаз једног неурона у првом слоју дат једначином (3.2.32), односно излаз целог слоја је дат изразом (3.2.33). Разлика од предходне мреже се може уочити у другом слоју, где је излаз једног неурона у овом слоју дат једначином:

1. ,

односно:

1. .

Узимајући у обзир да други слој садржи два неурона, излаз целог слоја можемо представити у следећем облику:

1. .

Пратећи исти тренд јасно је да поред излаза другог слоја **z2**, утицај на излаз ове мреже имати ће и улаз **x** и излаз првог слоја **z1** и да једначина излаза бити:

1. .

#### Дефиниције ауторегресивних модела

Ауторегресивни модели су основни алати за моделирање и анализу динамичких система, описујући како тренутни излаз система зависи од његових претходних излаза. Они играју кључну улогу у хватању унутрашње динамике система и често се проширују укључивањем спољних улаза, што доводи до свеобухватнијих модела. У области идентификације система, ауторегресивни модели се користе за креирање математичких представљања система заснованих на посматраним подацима. Ови модели су посебно ефикасни за системе где се понашање може предвидети на основу претходних стања, са фокусом на разумевање унутрашње динамике. Када су присутни спољни фактори, ауторегресивне компоненте се могу комбиновати са егзогеним улазима, формирајући напредније моделе као што су ARX или ARMAX, који укључују и унутрашњу динамику и спољне утицаје.

*Дефиниција* **9 Ауторегресивни (AR) модел**  је тип статичког модела који изражава тренутну вредност променљиве као линеарну комбинацију њених претходних вредности. Често се користи за предвиђање података који зависе од времена, претпостављајући да је тренутни излаз функција претходних излаза. Општи облик ауторегресивног модела p-тог реда дат је једначином:

1. .

Где је:

* y(t) — излаз,
* c — константа,
* φi —​ i-ти коефицијент модела,
* ε(t) — грешка,
* p — ред модела, који показује број коришћених претходних вредности.

*Дефиниција 10* **Егзогена улазна величина** представља све познате улазне величине система, односно објекта, које битно утичу на рад самог система.

Да сумирамо, егзогена улазна величина се односи и на све жељене вредности излазне величине и све мерене/познате величине поремећаја.

*Дефиниција 11* **Ауторегресивни модел са егзогеним улазом (ARX модел)** је тип линеарног динамичког модела који се користи у идентификацији система и комбинује **ауторегресивни** део (AR) са **егзогеним улазом** (X). Општи облик ARX модела је:

1. .

Где је:

* y(t) — излаз,
* u(t−j) — егзогени улаз,
* ai —​ коефицијент за ауторегресивне чланове, где је i = 1, 2, ..., na,
* bj — коефицијент за егзогене улазе, где је j = 1, 2, ..., nb,
* na —​ број претходних вредности излаза (ред ауторегресије),
* nb —​ број претходних вредности улаза које се узимају у обзир,
* ε(t) — грешка која са собом носи информације о свим не мереним улазима (углавном поремећајима).

*Дефиниција 12* **Нелинеарни ауторегресивни модел са егзогеним улазима (NARX)** је врста динамичког нелинеарног модела који се обично користи за моделирање система са информацијама о прошлим улазима и прошлим излазима који утичу на будуће излазе. Овај модел је описан следећом једначином:

1. ,

где су:

* y(t) — излаз,
* u(t) — улаз,
* f(•) — функција нелинеарности и
* 𝜀(t) — грешка која са собом носи информације о свим не мереним улазима (углавном поремећајима).



*Графички приказ нелинеарног ауторегресивног модела са егзогеним улазима —  
Слика 3.25*

#### Идентификација система

За потребе овог задатка искористили смо податке из трећег експеримента, где смо насумично одабрали и спојили тринаест случајева у један велики сет података и искористили за обучавање мреже и естимацију модела, а осталих пет смо такође спојили и искористили за валидацију.

За идентификацију и валидацију модела искористили смо скрипту Matlab скрипту bbModelEstimation.m где смо за конструкцију модела изабрали нелинеарни ауторегресивни модел са егзогеним улазом, а за излазну функцију, односно нелинеарнос, изабрали вишеслојну каскадну неуронску мрежу без повратне спреге. Кроз пар итерација дошли смо до кофигурације каскадне мреже која има пет скривених слојева са следећим бројем неурона [2, 3, 4, 3, 2] и ауторегресивног модела са егзогеним улазом који се може описати следећом једначином:

1. .

Скрипта — bbModelEstimation.m

close all

clear all

clc

training\_data\_path = "..\Ball\_And\_Beam\System\_Identification\Matlab\_Scripts\Black\_Box\TestData.csv";

validatio\_data\_path = "..\Ball\_And\_Beam\System\_Identification\Matlab\_Scripts\Black\_Box\ValidationData.csv";

T = readtable(training\_data\_path, 'ReadVariableNames', true);

training\_data = iddata(T.Angles, T.PWM, 0.01);

net = cascadeforwardnet([2,3,4,3,2]);

N2 = neuralnet(net);

sys = nlarx(training\_data, [4 4 0], N2);

% Validate against training data

figure(1);

compare(training\_data, sys);

figure(2);

resid(training\_data, sys);

% Validate against validation data

T = readtable(validatio\_data\_path, 'ReadVariableNames', true);

validatio\_data = iddata(T.Angles, T.PWM, 0.01);

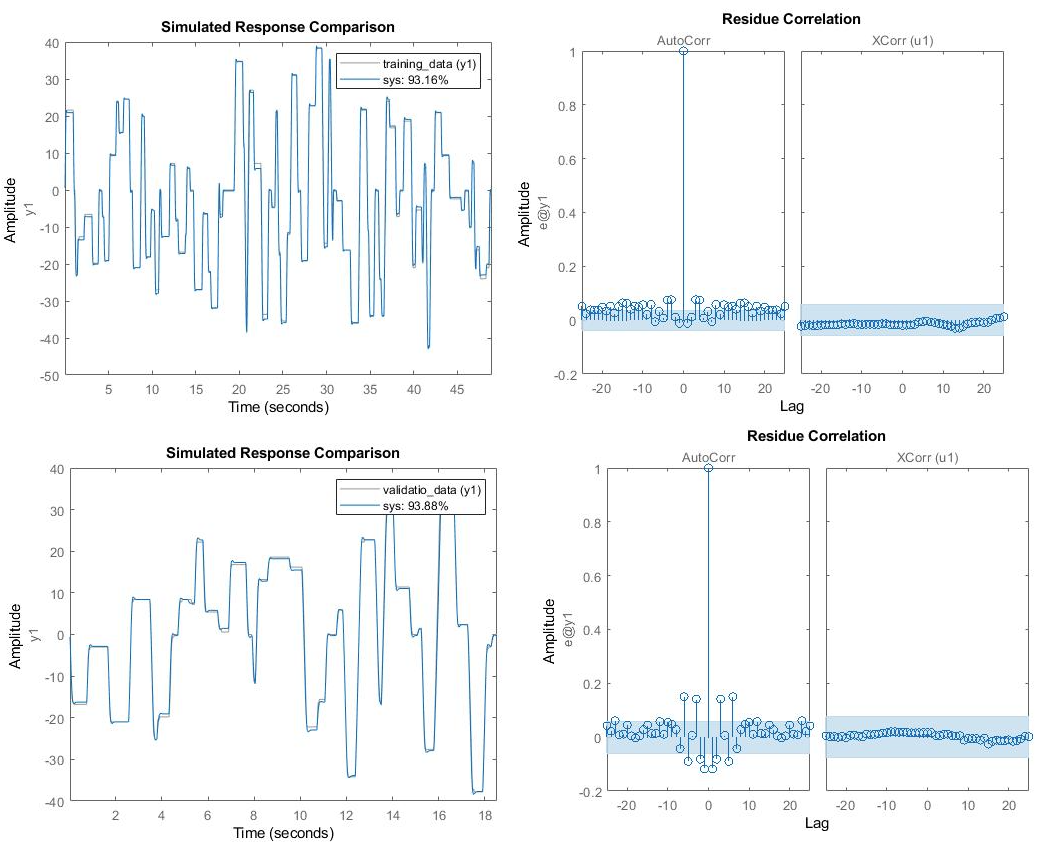
figure(3);

compare(validatio\_data, sys);

figure(4);

resid(validatio\_data, sys);

Након естимације модела, модел смо упоредили и са подацима за естимацију и подацима за валидацију. У оба случаја поклапање стварног излаза и излаза модела су нешто већи од 93%, а на основу резидуалне анализе можемо видети значајно побољшање ауто корелације резидуала, што нам говори да овај модел успешно покрива динамику система.



*Резултати валидације нелинеарног ауторегресивног модела са егзогеним улазом – Слика 3.26*

# Пројетовање и имплементација регулатора

Kratka naliza stabilnosti sistme opstim Bodeovim kriterijumom

Konsruisanje PID regulatora i odabr parametara AUTO tune metod

Implementacija PID regulatora

Upore

# Упоређивање симулације и стварног модела

# Закључак

# Додатак

## Додатак А – Техничка документација стандардних компоненти

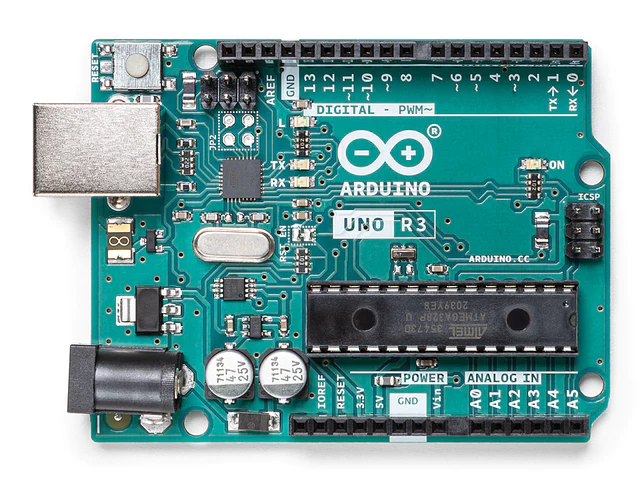
### Микроконтролер – Arduino Uno

У данашње време је немогуће замислити аутоматски систем без присуства неког типа рачунара. Рачунарске јединице попут PLC-а се данас користе у различитим индустријама где је потребно управљати робусним стварима попут производних линија и котловима за прављење полуфабриката од различитх метала, док се рачунари попут микроконтролере и микропроцесоре користе у авионима, аутомобилима, али и многим другим другим индустријама за обраду сигнала и фино управљање актуатора.

У системима аутоматског управљања рачунари на улаз добијају сигнал жељеног динамичког понашања система (мада то често може бити и записано у његовој меморији), као и мерени одзив система, односно стварно динамичко понашање система. На основу ових улазних величина и одговарајућег алгоритма управаљања он израчунава вредност управљања.

За потребе овог система изабрана је Arduino Uno (слика 2.3) развојна плоча са Atmel-овим микроконтролером. Arduino Uno је платформа отвореног кода коју је развио Arduino тим. Ова платформа се састоји од хардверског и софтверског дела, а користи се за стварање и контролисање интерактивних електронских пројеката.

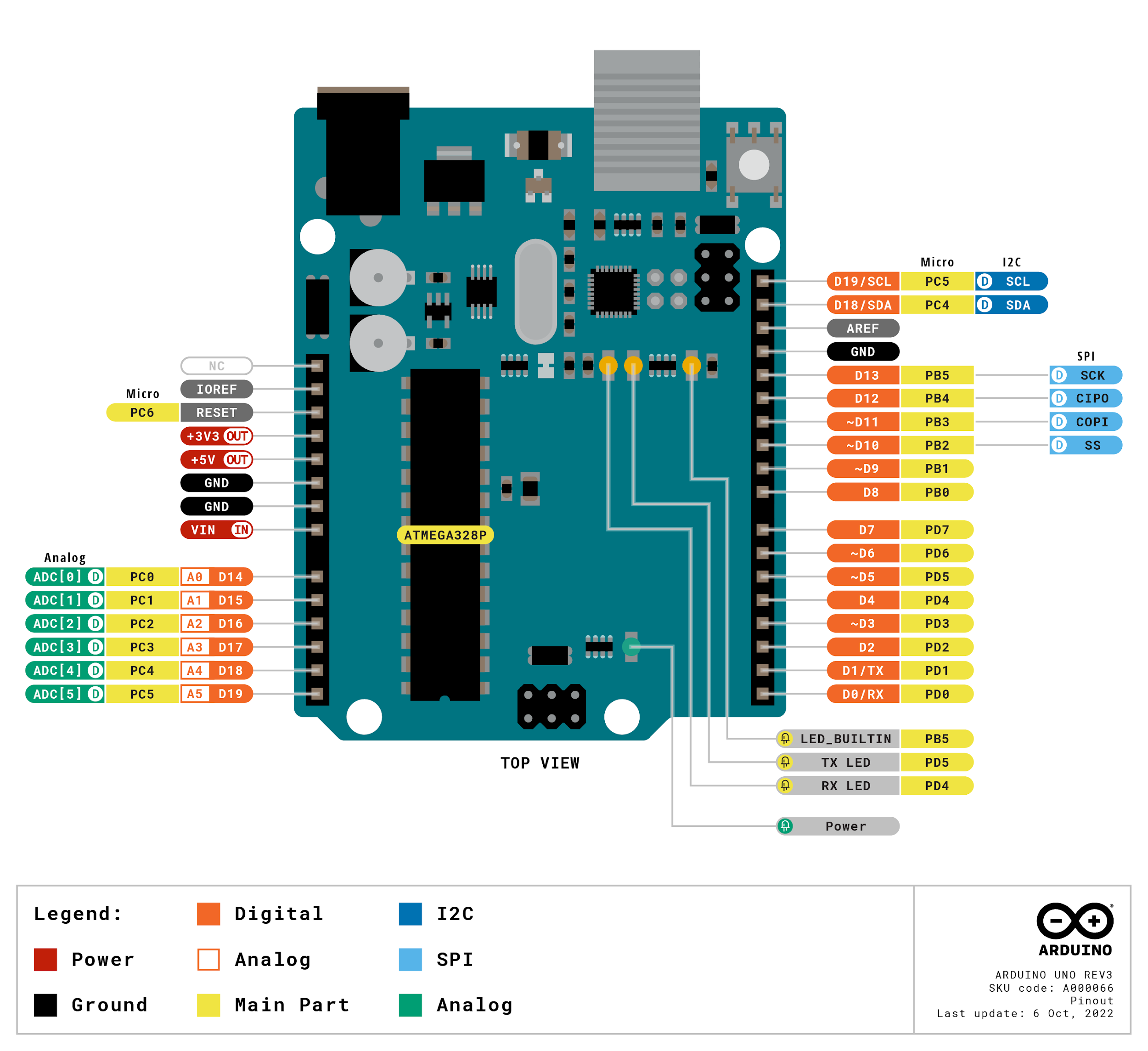
Техничке карактеристике и шема улазно/излазних пинова овог микроконтролера су приказани у табели 7.1 и слици 7.1.



*Arduino Uno – Слика 7.1*

|  |  |
| --- | --- |
| Микроконтролер | ATmega328P |
| Радни напон | 5 V |
| Улатни напон | 7-12 V |
| Дигитални У/И пинови | 14 |
| PWM дигитални У/И пинови | 6 |
| Аналогни улазни пинови | 6 |
| Једносмерна струја по У/И пину | 20 mA |
| Флеш меморија | 32 KB (ATmega328P) |
| SRAM | 2 KB (ATmega328P) |
| EEPROM | 1 KB (ATmega328P) |
| Радни такт | 16 MHz |
| Дужина | 68.6 mm |
| Ширина | 53.4 mm |
| Маса | 25 g |

Техничке карактеристике микроконтролера Arduino Uno – Табела 7.1



*Шема улазно/излазних пинова – Слика 7.2*

### Ултрасонични сензор – HC – SR04

Ултрасонични сензор HC-SR04 је широко коришћен сензор за мерење удаљености, који ради на принципу емитовања и примања звучних таласа. Овај сензор користи ултразвучне таласе, чија је фреквенција већа од 20 kHz (изван људског слуха), да би утврдио растојање до објеката у свом домету.

Принцип рада ултрасоничног сензора се може објаснити у наредна четри корака:

1. Емитовање сигнала: Сензор има два главна дела – један за емитовање ултразвучног сигнала (предајник) и други за пријем тог сигнала (пријемник). Када се покрене, предајник емитује ултразвучни сигнал у трајању од око 10 микросекунди.
2. Одбијање сигнала: Када ултразвучни сигнал наиђе на објекат, он се одбија назад према сензору.
3. Пријем сигнала: Пријемник на сензору бележи време које је прошло од тренутка емитовања сигнала до пријема одбијеног сигнала.
4. Израчунавање удаљености: Сензор користи формулу да би израчунао растојање на основу времена које је прошло од емитовања до пријема сигнала. Формула за растојање је:  
     
     
   Где је:
   * **D** - растојање до објекта
   * 𝞾 – брзина звука (приближно 343 м/с на собној температури)
   * **t** – време проласка сигнала

HC-SR04 се често користи у разним пројектима роботике, аутоматизације и система за избегавање препрека због своје једноставности, ниске цене и поузданости. Лако се интегрише са микроконтролерима попут Arduino, Raspberry Pi и других, чинећи га идеалним за примене у разним “уради сам” пројектима. Овај сензор је приказан на слици 7.3, док се његове карактеристике могу наћи у табели 7.2.

*Ултрасонични сензор HC – SR04 – Слика 7.3*



|  |  |
| --- | --- |
| Радни напон | 3.3Vdc ~ 5Vdc |
| Струја у мировању | <2mA |
| Радна струја | 15mA |
| Радна фреквенција | 40KHz |
| Опсег рада и прецизност | 2cm ~ 400cm (1in ~ 13ft) ± 3mm |
| Осетљивост | -65dB мин. |
| Ефективни угао | 15° |
| Димензије | 45mm x 20mm x 15mm |
| Тежина | 9g |

*Техничке карактеристике ултрасоничног сензора HC – SR04 – Табела 7.2*

### Серво мотор - TowerPro SG-5010

Серво мотори једносмерне струје су специјализовани типови електромотора дизајнирани за прецизну контролу угла или линеарне позиције, брзине и убрзања. Широко се користе у апликацијама где је прецизна контрола кретања критична, као што су роботи, CNC машине и аутоматизациони системи.

За овај пројекат изабран је TowerPro SG-5010 серво мотор, који је свестран и широко коришћен модел, погодан за различите апликације, укључујући хоби роботику, возила на даљинско управљање (RC) и основне задатке аутоматизације. Пружа добар баланс између перформанси и цене, чинећи га популарним избором како за почетнике тако и за искусне програмере. Техничке карактеристике овог мотора можете наћи у табели 7.3.

|  |  |
| --- | --- |
| Тип мотора | Стандардни серво мотор једносмерне струје |
| Радни напон | 4.8V до 6.6V |
| Стални обртни момент | 4.8V: 4.3 kg.cm, 6V: 5.5 kg.cm |
| Радна брзина | 4.8V: 0.20 сек/60°, 6V: 0.16 сек/60° |
| Тип зупчаника | Пластични зупчаници |
| Контролни систем | PWM (Ширина импулса модулација) |
| Опсег ширине импулса | 1000-2000 µs |
| Опсег ротације | 180° (стандардно) |
| Тежина | 38g |
| Димензије | 40.7mm x 19.7mm x 42.9mm |
| Тип конектора | тропински JR |
| Тип лежаја | Једноредни куглични лежај |
| Температурни опсег | -10°C до +50°C |
| Материјал кућишта | Пластика |

*Техничке карактеристике серво мотора TowerPro SG-5010 – Табела 7.3*



*Серво мотора TowerPro SG-5010 – Слика 7.4*

### Микропроцесор - Raspberry Pi 4

Raspberry Pi 4 је моћна и свестрана развојна плоча са микропроцесором, дизајнирана за широк спектар апликација, од образовања до напредних уграђених система. Са својим четворојезгарним ARM Cortex -А72 процесором, вишеструким опцијама меморије и обимним могућностима повезивања, Raspberry Pi обезбеђује процесорску снагу и флексибилност неопходну за једноставне прототипе, као и за сложене уграђене пројекте. Подржава два 4К дисплеја, брзе USB 3.0 везе и има робусне могућности умрежавања, што га чини идеалним за мултимедијалне апликације, IoT уређаје и много више. Било да развијате хоби пројекат малог обима или индустријску апликацију у пуном обиму, Raspberry Pi је поуздана и богата функцијама платформа.Техничке карактеристике овог сензора се могу пронаћи у табели 7.4.



*Raspberry Pi 4 и шема улазно/излазних пинова – Слика 7.5*

|  |  |
| --- | --- |
| Процесор | Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit, 1.5 GHz |
| Графичка карта | VideoCore VI, подржава OpenGL ES 3.0, Vulkan, H.265 (4Kp60 декодирање) |
| Меморија (RAM) | 8GB LPDDR4 SDRAM |
| Меморија за складиштење | MicroSD картица (подржава велике брзине UHS-I), USB boot support |
| Мрежа | Gigabit Ethernet, Dual-band 802.11ac Wi-Fi, Bluetooth 5.0 |
| USB портови | 2x USB 3.0, 2x USB 2.0 |
| Видео излаз | 2x Micro-HDMI (подржава резолуцију до 4Kp60) |
| Аудио излаз | четворо полни стерео излаз и композитни видео, HDMI аудио, I²S аудио |
| Улазно/Излазни Пинови | 40 пинова, 3.3V логика |
| Интерфејс за камеру | MIPI CSI |
| Интерфејс за дисплеј | MIPI DSI (Display Serial Interface) |
| Напајање | 5V/3A преко USB-C конектора |
| Оперативни системи | Raspberry Pi OS, Ubuntu, Manjaro, LibreELEC, RetroPie |
| Димензије | 85.6mm x 56.5mm x 17mm |
| Маса | 46g |
| Типови комуникације | 2x UART, 2x SPI, 5x I²C, 27x GPIO, 1x PCM/I²S, 1x SDIO |
| Потрошња енергије | У стању мировања 2.7W, приликом максималног оптерећења 7W |

*Техничке карактеристике микропроцесора Raspberry Pi 4 – Табела 7.4*

### Ротациони енкодер – LDP3806–600BM–5G –24C

Ротациони енкодер је електромеханички уређај који конвертује угаону позицију или кретање осовине у аналогни или дигитални сигнал. Ови сигнали су кључни за одређивање позиције, брзине и правца ротационог кретања. Ротациони енкодери се широко користе у апликацијама које захтевају прецизну контролу ротационог кретања, као што су системи за контролу мотора, роботика, CNC машине и индустријска аутоматизација.

LDP3806-600BM-G5-24C је инкрементални ротациони енкодер познат по својој високој резолуцији и робусном дизајну, што га чини погодним за апликације које захтевају прецизну контролу кретања. Идеалан је за примене које захтевају повратне информације високе резолуције у изазовним окружењима, као што су прецизна контрола мотора, роботика и индустријска аутоматизација. Његова робусна конструкција обезбеђује поуздан рад у захтевним условима.

|  |  |
| --- | --- |
| Тип енкодера | Инкрементални ротациони енкодер |
| Резолуција | 600 импулса по револуцији |
| Пречник осовине | 6mm |
| Радни напон | 5V до 24V DC |
| Тип излаза | Квадратни (А и Б фаза сигнали) |
| Излазни напон | Високо: ≥2.4V (TTL-компатибилан), Ниско: ≤0.4V |
| Максимална фреквенција одговора | 100 kHz |
| Максимална брзина ротације | 5000 ротација у минути |
| Излазна струја | 30 mA макс по каналу |
| Радна температура | -10°C до +70°C |
| Конекција | Кабл са штитом, 5-пински конектор |
| Димензије | Пречник 38mm x дужина 44mm |
| Тежина | 150g |

*Техничке карактеристике ротационог енкодера LDP3806-600BM-G5-24C – Табела 7.5*



*Ротационог енкодера LDP3806-600BM-G5-24C – Слика 7.6*

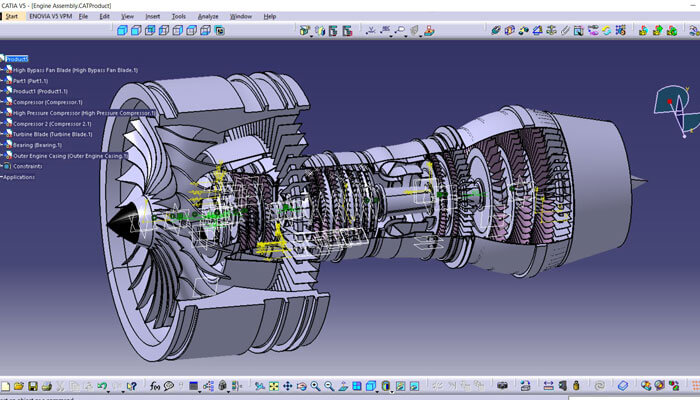
## Додатак Б – Адитивна технологија производње и 3D штампа

## Додатак В – Софтверски алати

### CATIA

CATIA је софтверски пакет за CAD/CAM/CAE дизајн и инженњеринг који се користи у различитим индустријама, као што су аутомобилска, авио, производна и многе друге индустрије. CATIA је развио француски софтверски гигант Dassault Systèmes и први пут је пуштен у употребу 1977. године.

Предности CATIA-е укључују могућност унапређења квалитета производа, повећања ефикасности производње, смањење трошкова развоја и производње и брже време доношења одлука. Међутим, мане укључују велику комплексност софтвера, велики трошак за набавку и обуку, и потребу за јаким рачунарским капацитетом.



*CATIA – Слика 7.1*

### Matlab и Simulink

### Python

### Улти макер цура

## Додатак Г – Помоћне скрипте

### Data\_processing\_01.py

import pandas as pd

import numpy as np

tmp\_path = "../Ball\_And\_Beam/System\_Identification/Data/Encoder\_data/Test\_2/Raw/tmp/"

processed\_path = "../Ball\_And\_Beam/System\_Identification/Data/Encoder\_data/Test\_2/Processed/"

for i in range(30):

    try:

        # Load data

        df = pd.read\_csv(tmp\_path + "Test\_" + str(i) + ".csv")

        # Sepertate data

        df\_1 = df.query("PWM ==" + str(df["PWM"].iloc[0]))

        df\_3 = df.query("PWM ==" + str(df["PWM"].iloc[-1]))

        # Add missing data

        missing\_time = np.arange(start=df\_1["Time\_s"].iloc[-1]+0.01, stop=df\_3["Time\_s"].iloc[0], step=0.01)

        missing\_angle = np.full(shape=np.shape(missing\_time), fill\_value=df\_1["Angles"].iloc[-1])

        missing\_pwm = np.full(shape=np.shape(missing\_time), fill\_value=df\_1["PWM"].iloc[0])

        # Create new DF with missing data

        df\_2 = pd.DataFrame()

        df\_2["Angles"] = missing\_angle

        df\_2["Time\_s"] = missing\_time

        df\_2["PWM"] = missing\_pwm

        # Create new DF from 3 separated DFs

        df\_new = pd.concat([df\_1, df\_2, df\_3], ignore\_index=True)

        # Process and save new DF

        df\_new["Time\_s"]= df\_new["Time\_s"].round(decimals=2)

        df\_new.drop\_duplicates(subset='Time\_s', keep='first', inplace=True, ignore\_index=True)

        df\_new.to\_csv(processed\_path + "Test\_" + str(i) + ".csv")

    except Exception as e:

        print("Something whent wrong!!!")

        print(e)

### Data\_processing\_02.py

import pandas as pd

import numpy as np

def add\_missing\_data(data\_1, data\_2):

    # Add missing data

    missing\_time = np.arange(start=data\_1["Time\_s"].iloc[-1]+0.01, stop=data\_2["Time\_s"].iloc[0], step=0.01)

    missing\_angle = np.full(shape=np.shape(missing\_time), fill\_value=data\_1["Angles"].iloc[-1])

    missing\_pwm = np.full(shape=np.shape(missing\_time), fill\_value=data\_1["PWM"].iloc[0])

    # Create tmp DF with missing data

    df\_tmp = pd.DataFrame()

    df\_tmp["Angles"] = missing\_angle

    df\_tmp["Time\_s"] = missing\_time

    df\_tmp["PWM"] = missing\_pwm

    return df\_tmp

tmp\_path = "../Ball\_And\_Beam/System\_Identification/Data/Encoder\_data/Test\_3/Test\_45/Raw/tmp/"

input\_data = "../Ball\_And\_Beam/System\_Identification/Data/Encoder\_data/Test\_3/Test\_45/Input\_data.json"

processed\_path = "../Ball\_And\_Beam/System\_Identification/Data/Encoder\_data/Test\_3/Test\_45/Processed/"

for i in range(10):

    try:

        df\_in = pd.read\_json(input\_data)

        if df\_in['test\_'+str(i)]['depricated']:

            print("This data can't be proecessed or requers additional manual action in order to be processed with this script")

        else:

            del df\_in

            df = pd.read\_csv(tmp\_path + "Test\_" + str(i) + ".csv", index\_col=[0])

            df["Time\_s"]= df["Time\_s"].round(decimals=3)

            # Separate data

            pwm = df["PWM"].unique()

            new\_data = {}

            for p, idx in zip(pwm, range(len(pwm))):

                new\_data[idx\*2] = df.query("PWM ==" + str(p))

            # Add missing data

            for idx in range(len(new\_data)-1):

                new\_data[idx\*2+1] = add\_missing\_data(data\_1=new\_data[idx\*2], data\_2=new\_data[idx\*2+2])

            # Merge all data frameworks into one

            df\_new = pd.DataFrame()

            for idx in range(len(new\_data)):

                df\_new = pd.concat([df\_new, new\_data[idx]], ignore\_index=True)

            # Save data

            df\_new["Time\_s"]= df\_new["Time\_s"].round(decimals=3)

            print("Test {} is unique: {}".format(i, df\_new["Time\_s"].is\_unique))  # Check if Time\_s is unique, if not correct it manualy

            df\_new.to\_csv(processed\_path + "Test\_" + str(i) + ".csv")

    except Exception as e:

        print("Something whent wrong while proccessing file number: ", i)

        print(e)

### Data\_processing\_03.py

import pandas as pd

import numpy as np

path\_1 = "../Ball\_And\_Beam/System\_Identification/Data/Encoder\_data/Test\_3/Test\_30/Processed/Test\_"

val\_path\_1 = "../Ball\_And\_Beam/System\_Identification/Data/Encoder\_data/Validation\_data/Dir\_1/Val\_Data\_"

path\_2 = "../Ball\_And\_Beam/System\_Identification/Data/Encoder\_data/Test\_3/Test\_45/Processed/Test\_"

val\_path\_2 = "../Ball\_And\_Beam/System\_Identification/Data/Encoder\_data/Validation\_data/Dir\_2/Val\_Data\_"

for i in range(10):

    try:

        df = pd.read\_csv(path\_2 + str(i) + ".csv")

        # Process and save new DF

        df["Time\_s"]= df["Time\_s"].round(decimals=2)

        df.drop\_duplicates(subset='Time\_s', keep='first', inplace=True, ignore\_index=True)

        df.to\_csv(val\_path\_2 + str(i) + ".csv")

    except Exception as e:

        print(e)

## Додатак Д– Технички цртежи и склопни цртеж система за балансирање лопте на шини

## Додатак Г – Технички цртежи тест бенча за прикупљање података

# Литература

* Mustafa Saad, Mohammad Khalallah, “Design and Implementation of an Embedded Ball-beam Controller Using PID Algorithm”, Universal Journal of Control and Automation 5(4): 63-70, 2017, DOI: 10.13189/ujca.2017.050402
* <https://www.mathworks.com/help/ident/gs/about-system-identification.html?s_eid=PSM_15028>
* <https://www.youtube.com/watch?v=Z1QS6FsxrJI&list=PLlmlmzye-q9gC7DwJ0xObASeOWUeADxJW>