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 и склапање 8](#_Toc4)

[2.3. Шема за повезивање серво мотора и ултрасоничног сензора са миркоконтролером 8](#_Toc5)

[3. Математички модел система за балансирање лопте на шини 9](#_Toc6)

[3.1. Једначина кретања лопте на шини 12](#_Toc7)

[3.1.1. Једначина промене нагибног угла шине 12](#_Toc8)

[3.1.2. Једначине кретања лопте на шини и преносна функција објекта 14](#_Toc9)

[3.2. Математичко модел серво мотора једносмерне струје 16](#_Toc10)

[3.2.1. Експеримент, прикупљање и обрада података 16](#_Toc11)

[3.2.1.1. Пројектовање и израда платформе за испитивање 16](#_Toc12)

[3.2.1.2. Поставка експеримента и прикупљање података 17](#_Toc13)

[3.2.1.3. Обрада података 29](#_Toc14)

[3.2.2. Идетификација система методом сиве кутије 34](#_Toc15)

[3.2.2.1. Општи математички модел спреге електро мотора једносмерне струје са редуктором 34](#_Toc16)

[3.2.2.2. Одређивање математичког модела 37](#_Toc17)

[3.2.3. Идетификација система методом црне кутије 44](#_Toc18)

[4. Пројетовање и имплементација регулатора 46](#_Toc19)

[5. Упоређивање симулације и стварног модела 46](#_Toc20)

[6. Закључак 46](#_Toc21)

[7. Додатак 49](#_Toc22)

[7.1. Додатак А – Техничка документација стандардних компоненти 49](#_Toc23)

[7.1.1. Микроконтролер – Arduino Uno 49](#_Toc24)

[7.1.2. Ултрасонични сензор – HC – SR04 51](#_Toc25)

[7.1.3. Серво мотор - TowerPro SG-5010 52](#_Toc26)

[7.1.4. Микропроцесор - Raspberry Pi 4 53](#_Toc27)

[7.1.5. Ротациони енкодер – LDP3806–600BM–5G –24C 54](#_Toc28)

[7.2. Додатак Б – Адитивна технологија производње и 3D штампа 55](#_Toc29)

[7.3. Додатак В – Софтверски алати 57](#_Toc30)

[7.3.1. CATIA 57](#_Toc31)

[7.3.2. Matlab и Simulink 58](#_Toc32)

[7.3.3. Python 58](#_Toc33)

[7.3.4. Улти макер цура 58](#_Toc34)

[7.4. Додатак Г – Помоћне скрипте 58](#_Toc35)

[7.4.1. Data\_processing\_01.py 58](#_Toc36)

[7.4.2. Data\_processing\_02.py 59](#_Toc37)

[7.4.3. Data\_processing\_03.py 60](#_Toc38)

[7.5. Додатак Д– Технички цртежи и склопни цртеж система за балансирање лопте на шини 61](#_Toc39)

[7.6. Додатак Г – Технички цртежи тест бенча за прикупљање података 61](#_Toc40)

[8. Литература 61](#_Toc41)

# Увод

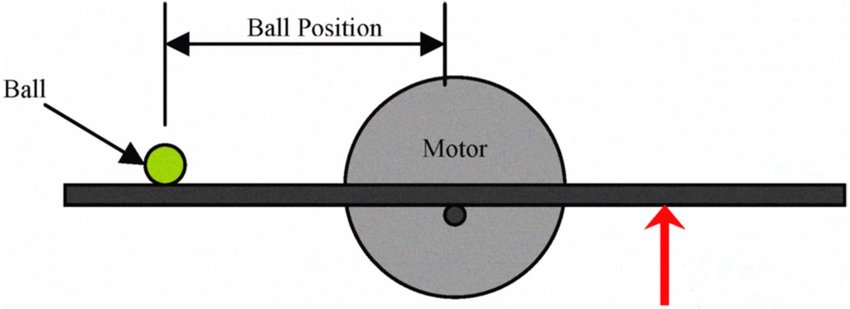
Управљање нестабилним системимам је веома важан задатак у аутоматском управаљањеу,а како се ови системи сматрају веома опасним када су нестабилни, они се испитују у лабараторијама. Систем за балансирање лопте на шини јесте нестабилан систем и може се поистоветити са реалним проблемима као што су хоризонтална 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 из 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.

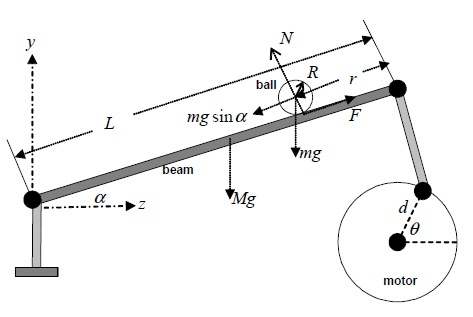
Систем балансирања лопте на шини се може видети у две конфигурације.

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*

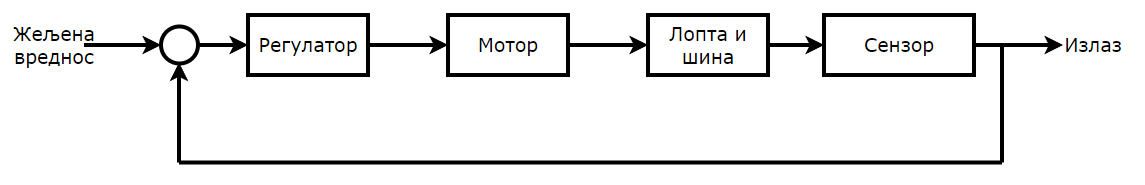
У овом раду ће се користи користи друга конфигурација, без обзира на сложеност. Такође, напоменуо бих да ће и следеће теме бити обрађене:

* Пројектовање и изрда система за балансер лопте на шини,
* Одређивање математичког модела и различите методе идетификације система,
* 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 смо у опште могли 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 лоптицу на одређену позицију тако што ће сам себе избацити из равнотежног стања. Да би смо то боље разумели, претпоставимо да имамо шину дугачку 50cm, да се лопта налази и мирује на десном крају те шине и да ми желимо да се та иста лоптица налази на средине те шине. У следећем кораку, на основу наше жељене вредности, мотор треба да се заокрене за угао α и нагне шину на лево, односно за угао β, сензор очита нову позицију лопте, која сада креће према среди, а затим ће регулатор на основу наше грешке да формира нови управљачки сигнал за мотор. У другом кораку мотор ће се поново заокренути за угао α и кориговаће угао шине, а сензор очита нову вредност и помоћи регулатор 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 (*слика 2.1*).



*Глобални дизајн система Слика 2.1*

Сада када имамо глобални дизајн и знамо генералну конфигурацију система можемо кренути и са пројектовањем и додатним питањима.

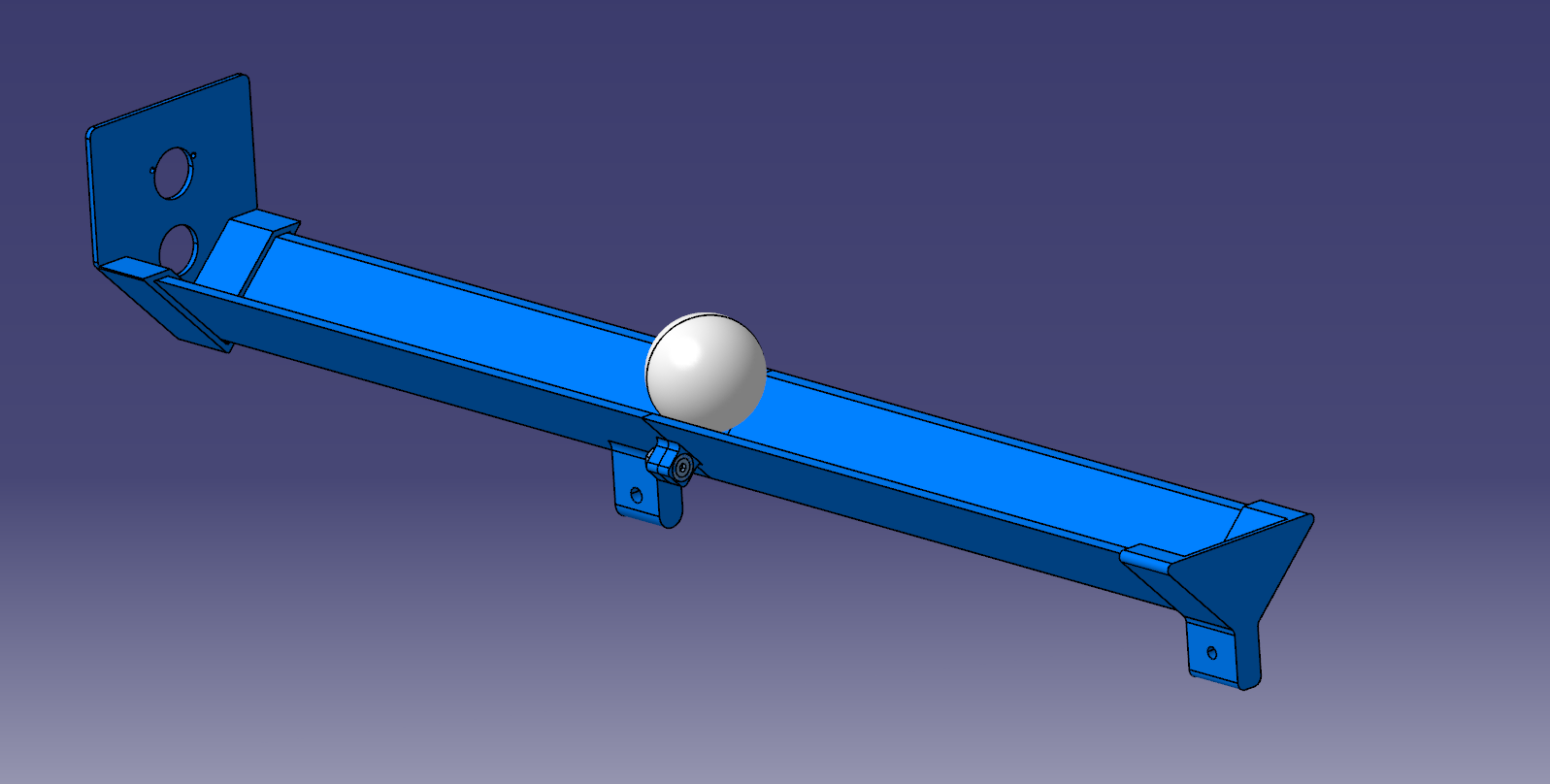
Прво питање на које треба одговорити је *“Шта се пројектује?”*, а одговор на то је: *“Систем за балансирање лопте на шини који се гони мотором преко система система полуга”*. Следећа питања на која је требало одговорити јеси:

* Како ће се и чиме мерити позиција лопте на шини? Којим мотором ће се гонити цео систем? Као и како ће се управљати целим системом?
  + Имајући у виду да је примарни циљ овог мастер рад имплементација свог стеченог знања из области аутоматског управљања све компоненте које су изабране су нискобуџетне. Па је тако за мерење позиције изабран ултрасонични сензор ХЦ-04, затим серво мотор СГ5010 и микроконтролер АРДУИНО УНО. Техничка документација и више информација о овим компонента, као и осталим компонентама које су коришћење приликом израде овог мастер рада, потражити на крају овог рада у додатку А.
* Како ће се сви испројектовани, не стандардни, делови израдити?
  + Сви делови који су испорјетовани за овај мастер рад су израђени адитивном технологијом израде, тј. уз помоћ 3Д штампача. Више речи о овој технологији, као и штампачу који је коришћен можете пронаћи на крају овог рада у додатку Б.

## Пројектовање система за балансирање лопте на шини

Након што смо дали одговоре на сва предходна питања, слика самог система постаје јаснија и може се приступи пројектовању.

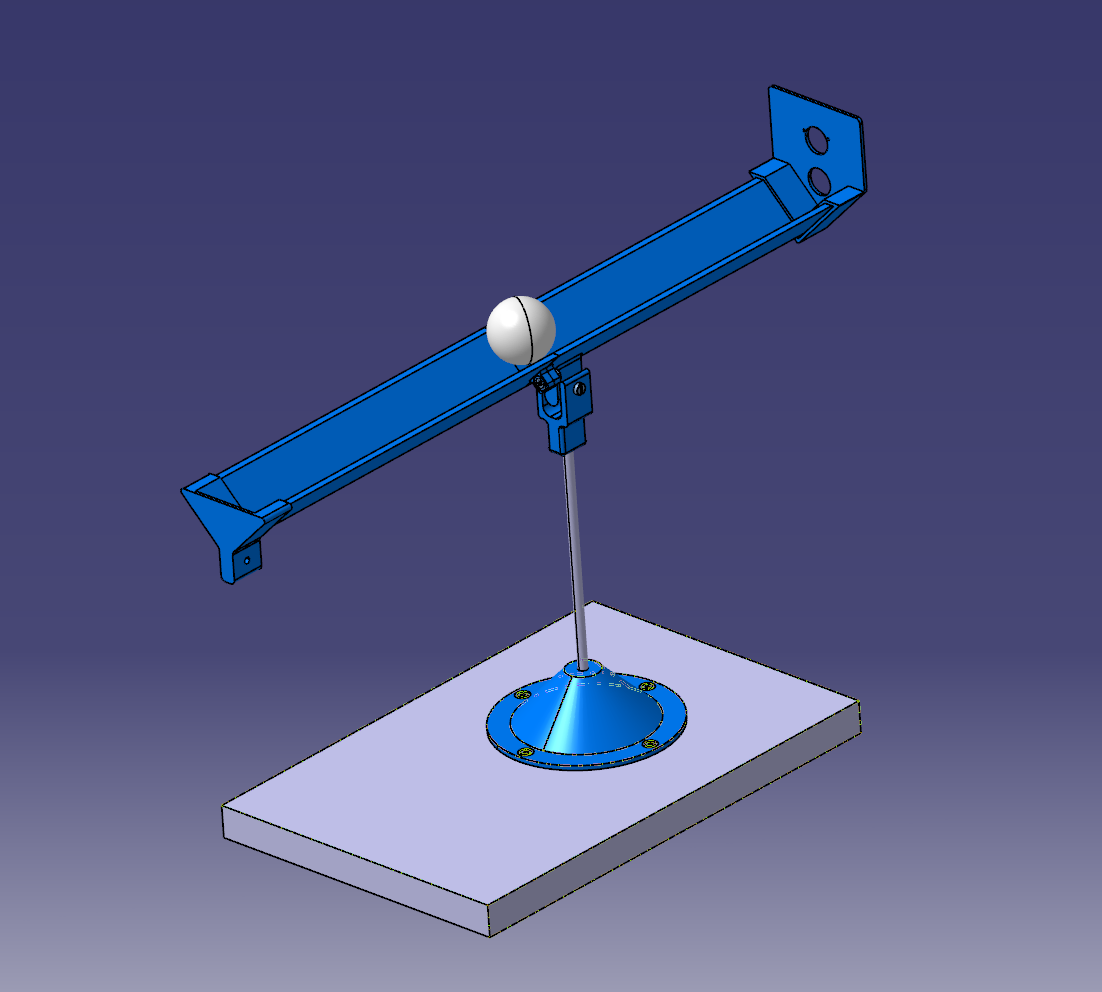
Први део који је измоделиран јесте шина и она је кроз пар итерација добила следећи изглед:



*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 крајеве шине. Једна половина шине поседује ушицу преко које ће се шина везати 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.

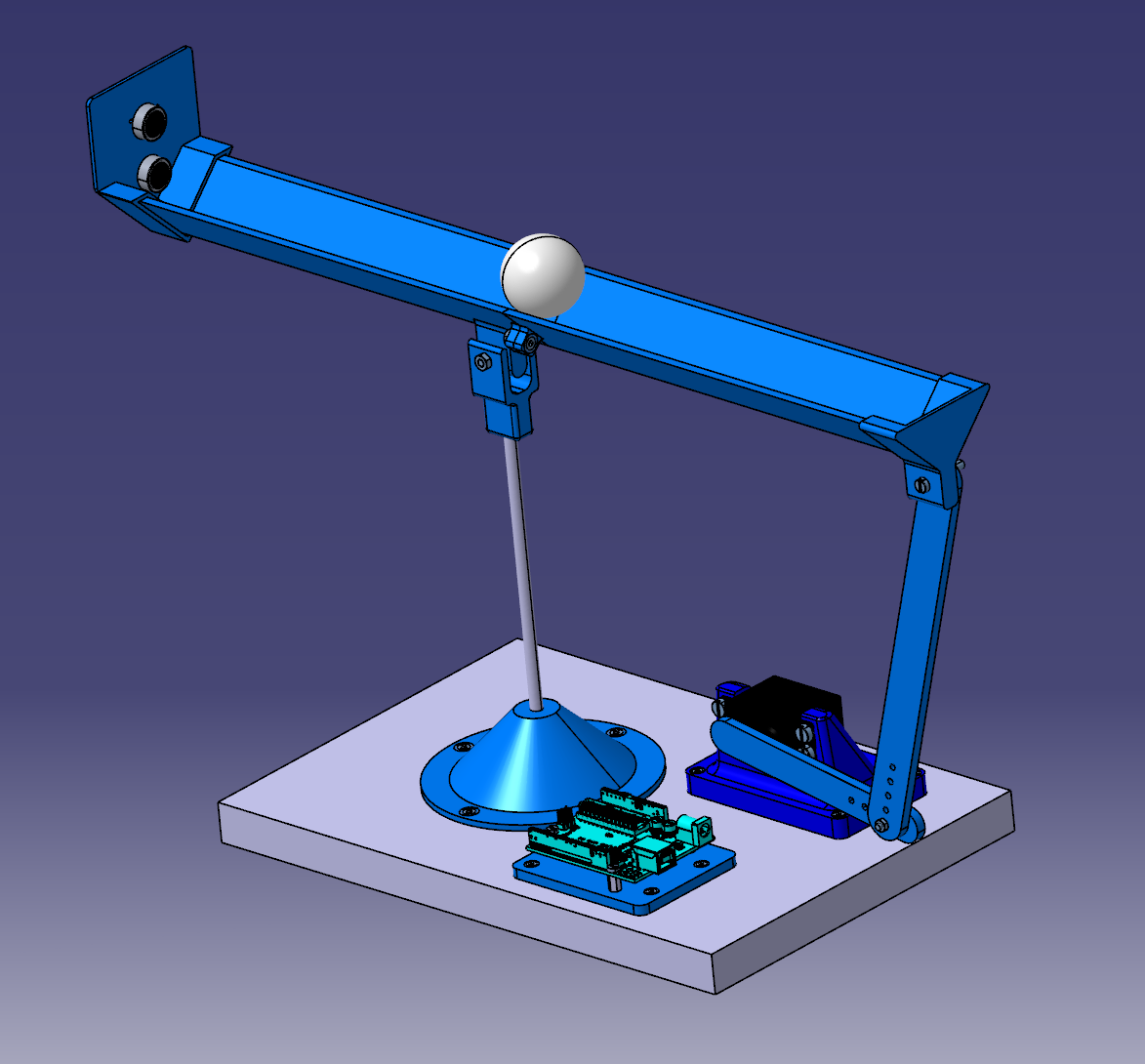
Следећи део који је испројектован јесте носач шине. Сам носач се састоји из три дела, ушице, навојне шипке и базе. Ова три дела су спојена или боље речено заварена. И ако на моделима се могу видети рупе за навојну шипку. Исте те рупе су запуњене уз помоћ пиштоља за лепак, а затим је навојна шипка загрејана и убачена у исте те, сад већ запуњене, рупе. Овака техника спајања је обезбедила стабилну и чврсту везу ова три дела. Такође, напоменуо бих да је цео носач зашрафљен за медијапанску плочу.



*Шина са носачем – Слика 2.3*

Након шине и носача, било је потребно испројетовати везу између шине и мотора и додати сензор и микроконторлер и носач за исти. Веза између шине и мотора се састоји из две полуге различитих дужина, серво мотора и носаче за серво мотор. Финални модел система за балансирање лопте на шини може се видети на слици 2.4

Цео овај систем је испројектован у софтверу за Компјутерски Адидитвни Дизајн (ЦАД) КАТИЈА. Више о овом софтверу, као и о другим софтверским алатима које сам користио за израду овог мастер рада, можете наћи и додатку В.



*Систем за балансирање лопте на шини – Слика 2.4*

## Израда и склапање

Узимају ћи обзир да је циљ овог мастер рад имплементација свог стеченог знања из области аутоматског управљања овај део ће садржати опште информације о изради, као и круцијалних информација о склапању.

Израда – Сви делови који су испројектовани за овај мастер рад су израђени адитивном методом поизводње и више о тој мети поизводње можете наћи на крају рада у додатку Б.

Склапање – сви технички и радионички цртежи који садрже се могу наћи у додатку Д. Редослед којим сам се водио приликом склапања система за балансирање лопте на шини је:

1. Склапање шине
   * Напомена – граничнике залепити епоксидним лепком за шину
2. Склапање носача за шину
3. Спајање шине и носача
4. Позиционирање шине и носача на медијапанској плочи + причвршћивање уз медијапанску плочу
   * Напомена – ножицу серво мотора и краћу полугу залепити епоксидним лепком. Затим направити отвор кроз који је потрбно провући вијак који иде на вратило серво мотора.
5. Позиционирање носача серво мотора са серво мотором и система полуга у односу на шину са носачем + причвршћивање за медијапанску плочу
6. Позиционирање носача за микроконтролер
7. Постављање микроконтролера и ултрасоничног сензора на предвиђена места за то
8. Повезивање сензора и серво мотора са микроконтролером (шема за повезивање је дата у наредном поглављу)

## Шема за повезивање серво мотора и ултрасоничног сензора са миркоконтролером

# Математички модел система за балансирање лопте на шини

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.

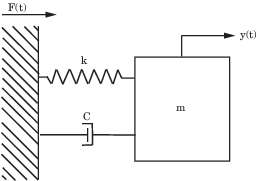
Сада када знамо главне задатке аутоматског управљања јасном нам је да су наредна два корака идентификација система и конструисање алгоритма за управљање. С обзиром да је познавање математичког модела система предуслов за конструисање алгоритма за управљање, онда је и јасно да ће у овом поглављу главни циљ бити идентификација самог система, односно одређивање његовог математичког модела.

*Дефиниција 1* **Модел физичког система** је идеализовани, замишљени, систем који задржава особине стварног система битне за његову анализу. Модел физичког система често буде

*Дефиниција 2* **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 познајемо динамику система и његове параметре, до математичког модела система можемо доћи користећи законе физике. Рецимо да је наш систем представљен на следећој слици:



*Систем пригушеног осцилаторног кретања тела – Слика 3.1*

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

Предходна једначина јесте математички модел система са слике коју даље можемо користити за анализу система.

Овакав систем се у жаргону назва Белом кутијом, јер нам је његова динамика позната, а и сви његови параметри.

Супротно од Беле кутије, јесте Црна кутија, односно модели чију динамику, а ни параметре, не познајемо. И у оваквим случајевима Идентификација система, као засебна област аутоматског управљања, долази до изражаја.

У оваквим случајевима прво је неопходно прикупити информације система и видети зависност излаза од улаза. То значи да би смо прво морали спровести пар експеримената над системом где би смо на улаз довели одређени сигнал, на пример одскочну функцију, а онда на излазу, уз помоћ одређеног сензора, очитали вредности и забележили их.

Следећи корак би био одабир структуре модела. За структуру система можемо одабрати преносну функцију са n бројем полова и m бројем нула, диференцијалну једначину, Луријеов систем, неуронску мрежу или било који други модел.

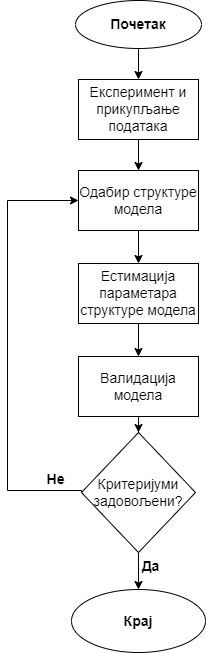
Након што смо прикупили податке и одабрали структуру модел, потребно је естимирати параметре и валидирати модел. Приликом валидације система треба користити нов сет података, где ћемо улазним сигналом побудити наш модел, а онда упоређивати излазне сигнале стварног система и нашег модела.

Када сумирамо све, идентификација система се састоји од експеримента и прикупљање вредности, одабира структуре модела, естимације параметара и валидације модела; али *“Како да изаберемо праву структуру система?”* и *“Шта да радимо ако нисмо задовољни резултатима након валидације?”*. Ствар јесте у томе што парвила приликом одабира стурктуре нема и ако након валидације модела нисмо задовољни резултатима, можемо се вратити и изабрати нову структуру или модификовати предходно изабрану и поново проћи кроз естимацију параметара и валидацију модела. 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 3* **Идентификација система**је методологија за конструисање математичког модела динамичког система користећи мерене улазне и излазне сигнале система.

*Дефиниција 4* **Процес идентификације система** састоји се из:

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



*Алгоритам за идентификације система – Слика 3.2*

Поред Белих и Црних кутија, постоје и Сиве кутије. То су системи чију динамику потпуно или добрим делом познајемо, али не параметре, или барем не све параметре. У оваквим случајевима се такође користи, предходно објашњена, метод за идентификацију система.

Вратимо се поново на систем са слике 3.1. Динамика овог система нам је позната, али рецимо да нам је једино маса тела позната од свих параметара/коефицијената. Да би смо одредили математички модел пратићемо алгоритам за идентификацију система. Прво ћемо спровести пар експеримената над моделом и прикупити податке. Следећи корак јесте одабир структуре система, а с обзиром да познајемо динамику система за структуру система можемо изабрати једначину (3.1). Затим неком од метода етимирати непознате коефицијенте (k и C) и на крају валидирати модел са новим сетом података.

Сада, ако погледамо наш систем за балансирање лопте на шини знамо да се он састоји од лопте на шини која је системом полуга спојена на серво мотор. У овом случају серво мотор је извршни орган, који се за одређени напон заокреће вратило за одређен угао одређеном брзином; док шину са лоптом можемо да посматрамо као објекат, где кретање лопте зависе он нагиба шине, односно угла заокретање серво мотора. Даљом анализом закључујемо да нам је математички модел серво мотора непознат и није дат у техничкој документацију, док математички модел објекта можемо одредити користећи законе кинематике и динамике тела.

На основу објашњења улоге и значаја идентификације система, као и анализе система за балансирање лопте на шини јасно је да циљ овог поглавља идентификација нашег система и одређивање његовог математичког модела.

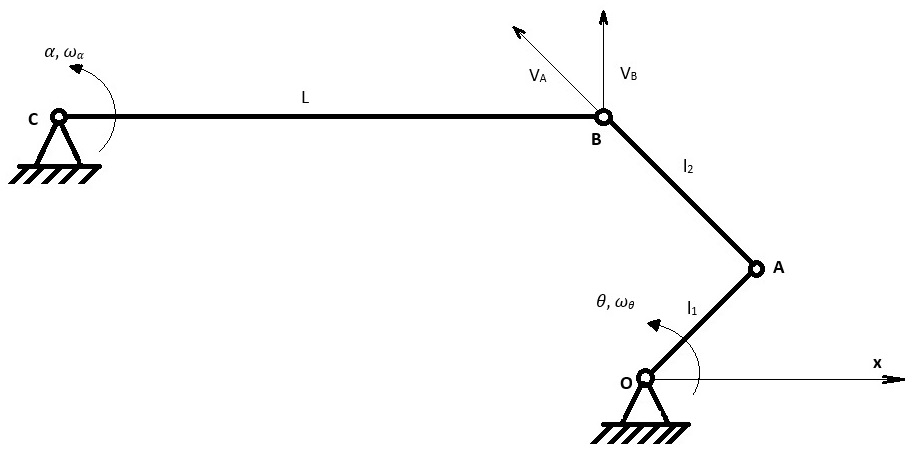
## Једначина кретања лопте на шини

Тумачећи конфигурацију система са слике 1.2 може се доћи до закључка да брзина и смер кретања лопте на шини зависи од нагибног угла шине, док исти тај угао зависи од угла излазног вратила мотора. На основу предходно реченог можемо закључити да је неопходно одредити једначину промене нагибног угла шине, да би смо могли да одредимо једначину кретања лопте.

### Једначина промене нагибног угла шине

Да би смо нашли једначину промене нагибног угла шине, поједноставимо слику 1.2 са сликом 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 нагибног угла шине,
* 𝜃 – угао излазног вратила серво мотора,
* 𝜔𝜃 – угаона брзина серво мотора.



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

Кренимо са постављањем једначина за брзине VA и VB:



Користећи теорију сличности троуглова можемо закључити следеће:

Затим комбинацијом једначина (3.1.1), (3.1.2) и (3.1.3) добија се:

и ако наставимо даље са поједностављивањем једначине, можемо да напишемо следећу диференцијалну једначину:

Затим једначину (3.1.5) интегралимо:

како би смо добили једначину промене угла 𝛼

Ако узмемо у обзир следеће:

и

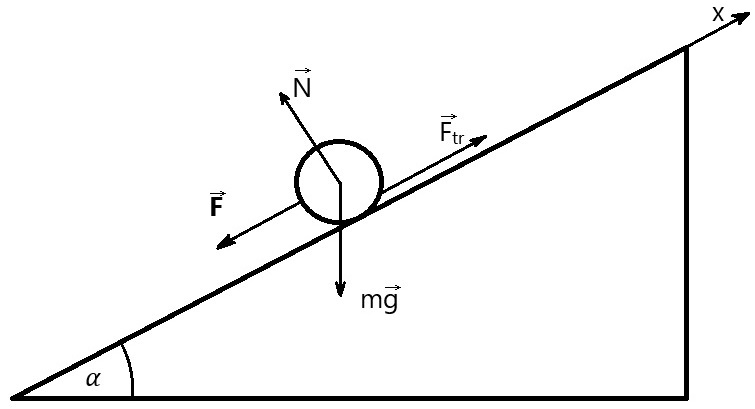
Онда из (3.1.8) и (3.1.9) следи једначина промене нагибног угла шине у зависности од промене угла вратила на серво мотору:

1. .

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

Да би смо одредили једначину кретање лопте по шини, потребно је да посматрамо случај када је нагибни угао шине 𝛼 различит од нуле. Узимајући предходну претпоставку у обзир, проблем може да се представи као кретање лопте по стрмој равни (слика 3.4).

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



Кренимо од једначине суме свих сила:

Знајући да ћемо за овај пројекат користити лоптицу за стони тенис, због њене мале масе можемо да занемаримо силу потиска, а са њом и силу трења. Стога наша једначина, пројектована на x осу ће имати следећи облик:

односно, ту исту једначину можемо да прикажемо у следећем облику:

1. .

Где су:

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

Пошто је:

тада једначину (3.1.12) можемо да напишемо у следећем облику:

Односно, када угао 𝛼 заменимо његовом релацијом из једначине (3.1.10), тада добијамо:

односно

1. .

За крај једначину (3.1.14) пребацимо у комплексни домен уз помоћ Лапласове трансформације:



то јест:

1. .

За крај једначину (3.1.14) пребацимо у комплексни домен:

1. ,

односно, ако симболе заменимо њиховим вредностима, тада добијамо:

1. .

## Математичко модел серво мотора једносмерне струје

У овом делу се бавимо одређивањем математичког модела серво мотора једносмерне струје, јер као што смо већ напоменулу, произвођач није приложио математички модел серво мотора у техничкој документацији.

Одређивању математичиког модела серво мотора приступићемо на два начина. Први јесте метод сиве кутије, где ћемо на основу општих познатих информација о серво мотрору формирати структуру модела, а потом одредити и вредности параметара тог модела. Док ћемо у другом методу серво мотор посматрати као црну кутију.

Пратећи алгоритам за идентификацију система обрадићемо следеће тебе:

* Експеримент и прикупљање података,
* Одабир конструкције модела
* Естимација параметара модела и
* Валидацију модела.

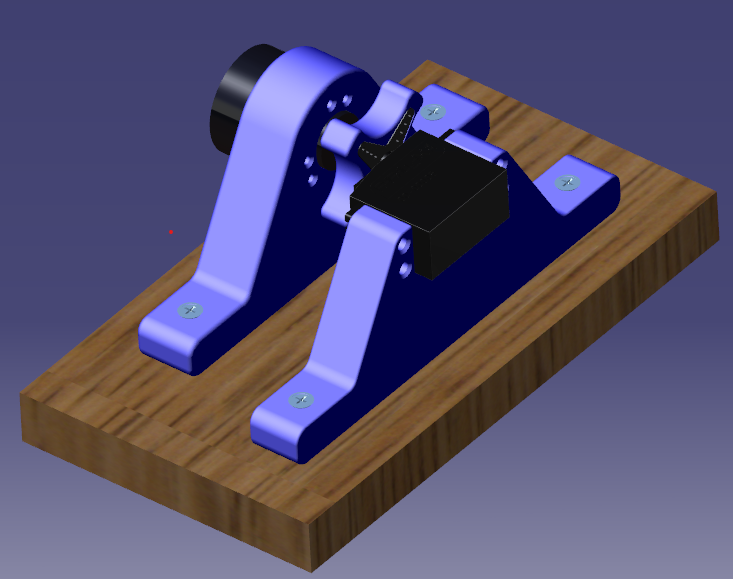
### Експеримент, прикупљање и обрада података

#### Пројектовање и израда платформе за испитивање

За потребе експеримента било је потребно конструисати платформу за испитивање. Приликом пројектовања водио сам се одговорима на следећа питања:

* Шта је циљ експеримената?
  + Циљ експеримента јесте прикупљање улазних и излазних сигнала.
* Шта је улазни сигнал серво мотора, а шта излазни?
  + За улазни сигнал серво мотора смо изабрали жељени угао вратила, који се уз помоћ једноставне формуле трансформише ширинско модуларни импулс (стварни улазни сигнал), док је излаз мотора промена угла вратила од почетне позиције до жељене позиције.
* Како и чиме мерити излазну вредност серво мотора?
  + Излазну вредност серво мотора, односно угаону промену излазног вратила серво мотора, мерили смо ротационим енкодером LPD3806. Нешто више о овом енкодеру можете пронаћи у додатку А на крају овог рада.

Водећи се одговорима на преднохдна питања испројектована и израђена је платформа за испитивање која се састојала од дрвене базе, за коју 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 се може видети на слици 3.5.



*Платформа за испитивање серво мотора – Слика 3.5*

#### Поставка експеримента и прикупљање података

У предходном делу смо рекли да је главни циљ овог експеримента прикупљање улатзних и излазних сигнала које ћемо касније искористити за естимацију параметара модела. На основу циља можемо закључити да наш експеримент треба да има следеће кораке:

* Побуди серво мотор одређеним сигналом,
* Очитај промене излазног сигнала и
* Сачувај очитане податке.

На први поглед задатак је једноставан, међутим до финалног решења је пронађено тек у трећој итерацији.

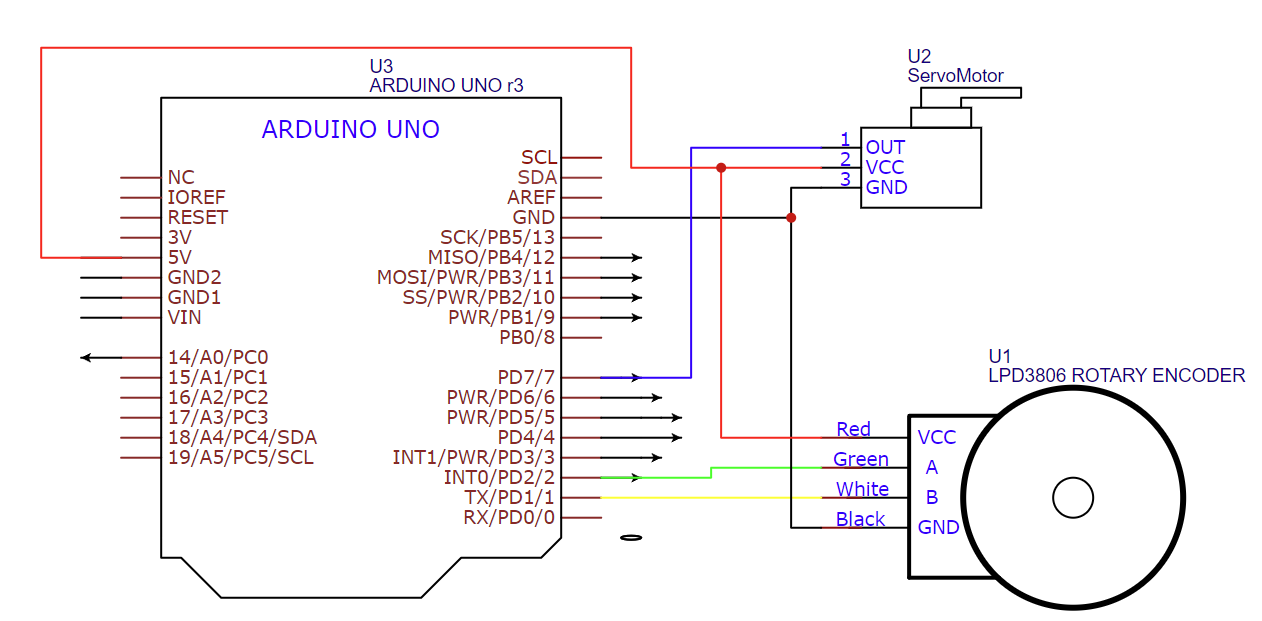
**Итерација 1**

У првој итерацији идеја је била да се и за побуду серво мотора и за очитавање промене угла излазног вратила серво мотора, уз помоћ ротационог енкодера, користи микроконтролер ArduinoUno. 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.

Пре него што дођемо до дела због чега ова идеја није успела одговоримо на следеће питање:

* Шта је главна петља, а шта прекидна петља?
  + Главна петља у систему ради непрекидно, обрађујући редовне задатке у секвенцијалном редоследу. Насупрот томе, прекидна петља се активира одређеним догађајима, омогућавајући систему 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, можемо закључити да је систем често улазио у прекидну петљу, а да су последице тога испрекидан ширинско модуларни сигнал који је слат мотору и јако вибрирање мотора.



*Шема за повезивања у првој итерацији – Слика 3.6*

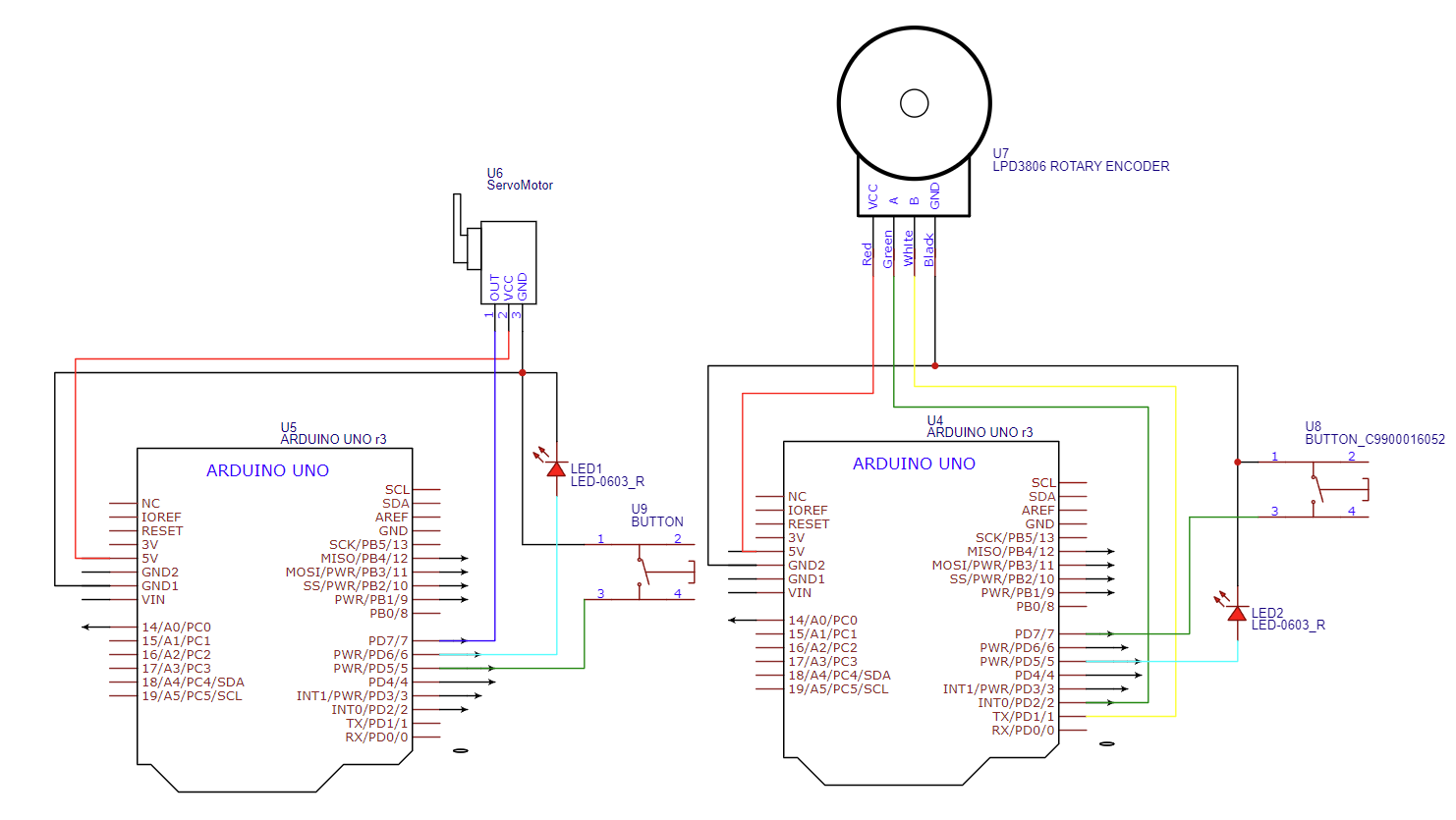
**Итерација 2**

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

Лед диода на конролеру који је слижио за покретање серво мотора је светлела све до притиска тастера. Након притиска тастера, лед диода би се угасила, затим би био извршен тест где мотор помера вратило за одређени угао и после картке паузе га враћа у почетни положај и након извршеног теста 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, диода би исто сијала све до притиска тастера. Након притиска тастера диода би се угасила, затим би се сви прикупљени подаци преко серијског порта слали на рачунар и након завршенонг слања, диода би поново засијала.

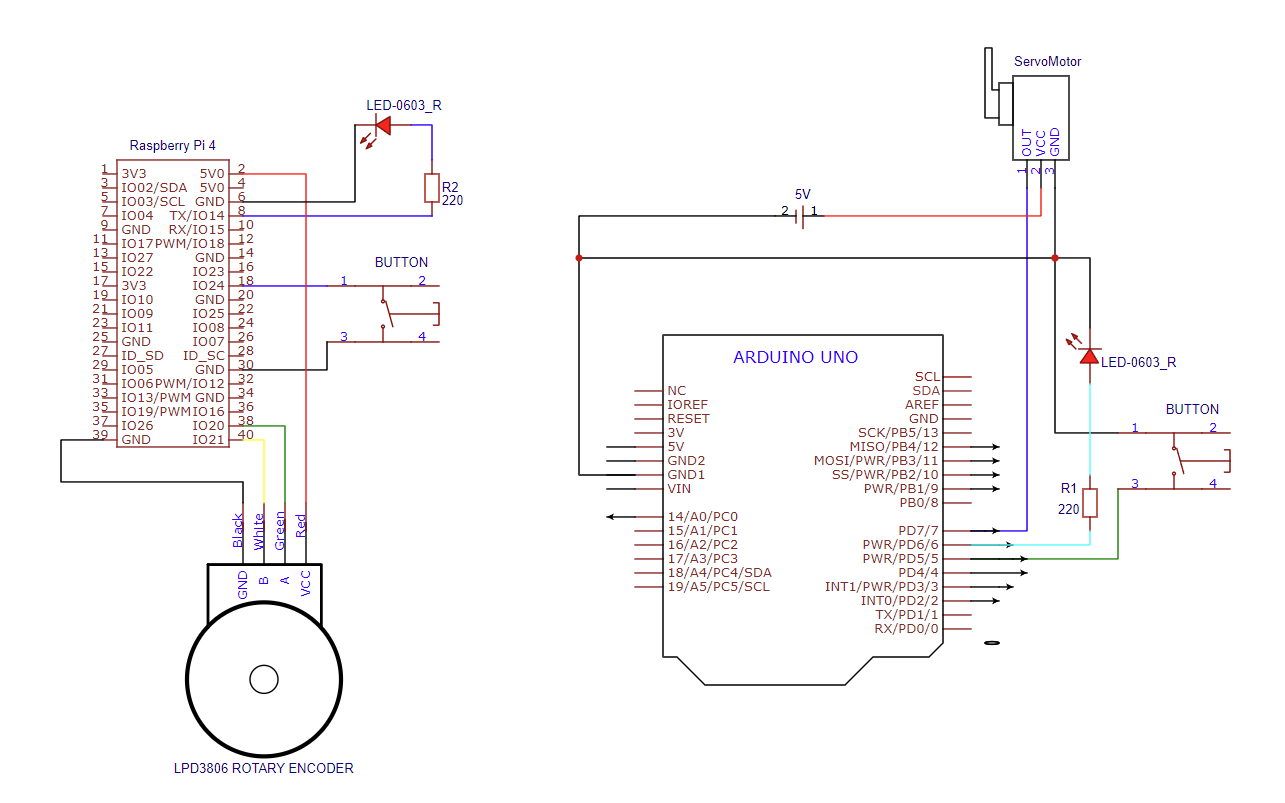
У овој итерацији сам наишао на два провлема. Један од проблема јесте била нестабилна ротација излазног вратила серво мотора услед нестабилног напајања. Док је други проблем било слање прикупљених података серијском комуникацијом. Због количине прикупљених података на самом почетку микроконтролер би се сам од себе ресетовао.



*Шема повезивања у другој итерацији – Слика 3.7*

**Итерација 3**

У трећој 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 остали исти. У наставку можете наћи шему за повезивање (слика 3.8), као и скрипте за прикупљање података и покретање серво мотора.



*Шема повезивања у тећој итерацији – Слика 3.8*

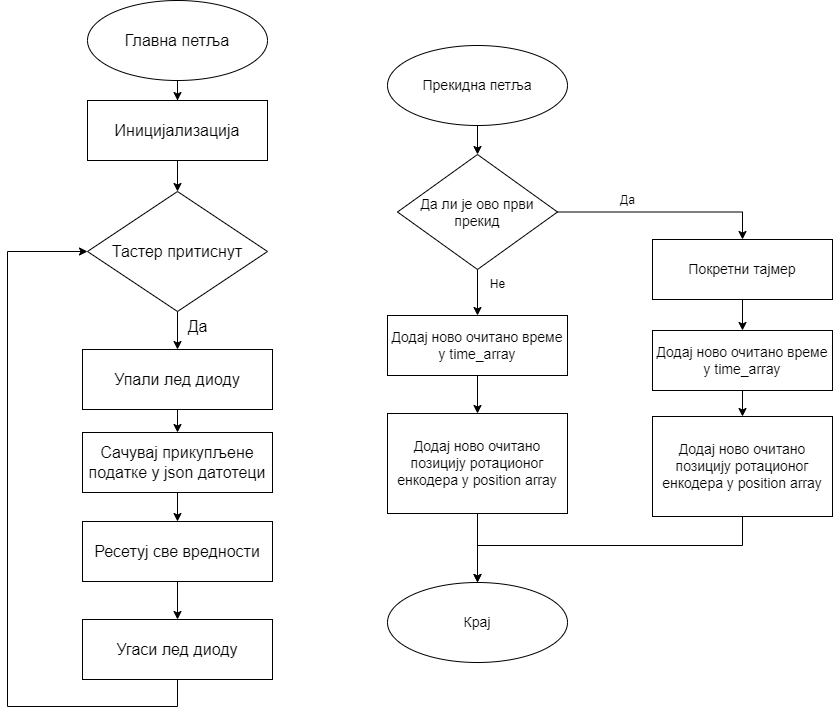
Као што смо напоменули, за мерење промене угла излазног вратила серво мотора употребили смо ротациони енкодер који је био повезан на микропорцесор Raspberry Pi. Поред енкодера на овај контролер су повезани тастер и лед диода. Лед диода нам је служила као индикатор да је у току чување података који су се чували након што се експеримент заврши и притисне тастер. Цео алгоритам можете видети на слици 3.9.

Скрипта за ове намене писана је у програмском језику Python, док смо следеће блиотеке користили за овај пројекат:

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

Напоменуо бих да је скрипта сачињена и од следећих функција:

* callback – функција очитава вредност ротационог енкодера и повећава, односно смањује, бројач,
* collect\_data – фунција која служи 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 debouncing проблема са тастером,
* button\_press\_handler – ова функција се можда може и назвати главном функцијом главне петље, јер она у себи садржи код за паљење и гашење лед диоде, чување очитаних података и ресетовање привремених податка, а притом је у директној спрези са debounce функцијом.



*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 би сијала, све док не притиснемо тастер који је исту диоду гасио и покретао експеримент, односно test функције. Скрипта за Arduino Uno је писана у програмском језику C у наменском развојном окружењу за овај контролер (Arduino IDE). За писање ове сркипте поред стандардних функција које нуди стандардна Arduino.h библиотека, искоришћене су и Servo.h библиотека која нам је омогућила лако покретање серво мотора, као и Time.h библиотека.

Скрипта се састојала од следећих функција:

* setup – ова функција је задужена за иницијализацију пино, где смо пин број 5 одабрали да буде излазни пин преко којег смо слали сигнале серво мотору, пин број 2 је подешен за читање сигнала тастера, а пин 7 за управљање лед диодом,
* loop – је главна функција која би гасила лед диоду приликом притиска на тастер и затим покретала један од тестова,
* test\_1 – функција за први сет експеримената/тестова,
* test\_2 – функција за други сет експеримената/тестова и
* test\_3 – функција за трећи сет експеримената/тестова.

Функције test\_1 и test\_2 деле исти алгоритам који се може видети на слици 3.10 (лево). Разлика између ова два експеримента је у почетним положајима излазног вратила (𝜃0) и скупу углова (𝛼) који су коришћени за експерименте;

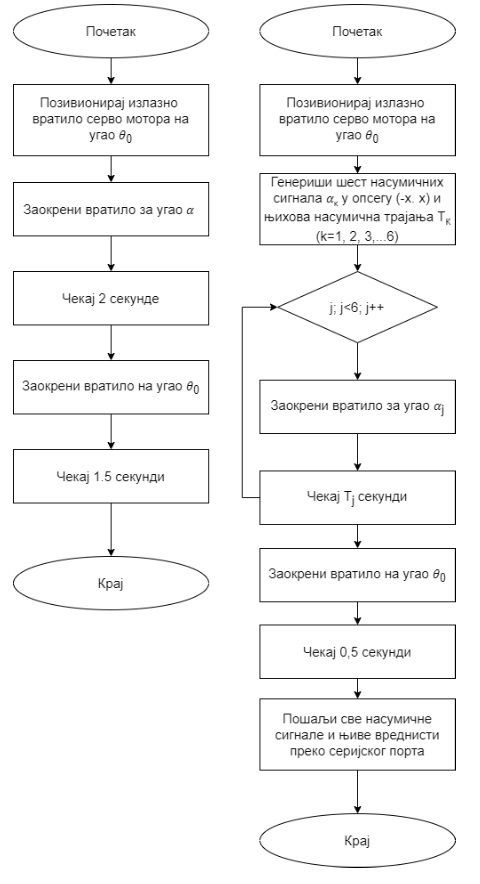
* test\_1
  + 𝜃0 = 0
  + 𝛼 ϵ (30, 45, 60, 90)
* test\_2
  + 𝜃0 = 90
  + 𝛼 ϵ (-60, -45, -30, 30, 45, 60).

Узимају у обзир да користимо хоби серво мотор који има толерирају од ±2° за сваки угао експеримент је изведен пет пута.

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 на почетни положај. Алгоритам ове функције се такође може наћи на слици 3.10 (десно).

Овај тип експеримента је изврше у две конфигурације по десет пута.

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



*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

  // attachInterrupt(digitalPinToInterrupt(2), test\_1, FALLING);

}

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

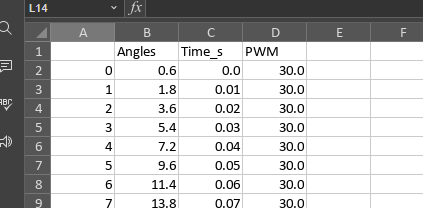
Пошто смо одрадили све експерименте и прикупили податке, било је потребно те податке обрадити и припремити за даљи рад.Сви прикупљени подаци су у json формату и садрже два низа. Један низ је садржи позиције у датим тренуцима, које треба претворити у углове, а други низ је садржи временске тренутке у нано секундама очитаних вредности које треба претворити у секунде.

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

* паузу од две секунде, која је карактеристична за функиције test\_1 и test\_2,
* да је излазно вратило серво мотора стизало до жељеног угла пре него што две секунде истекну,
* да су бележене само промене угла излазног вратила серво мотора.

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

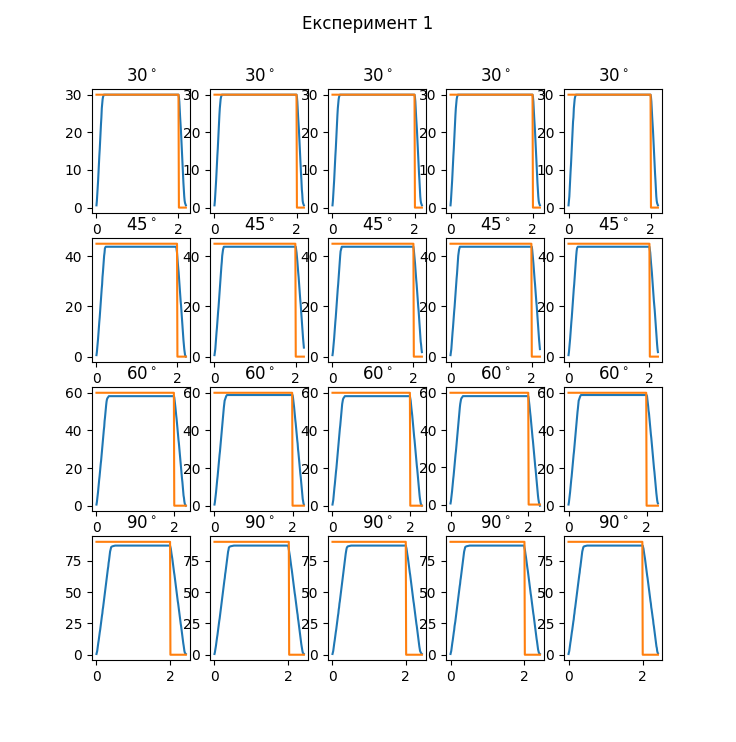
Након сређивања уз помоћ скрипте, сви подаци су прегледани и по потреби додатно, ручно, исправљени.



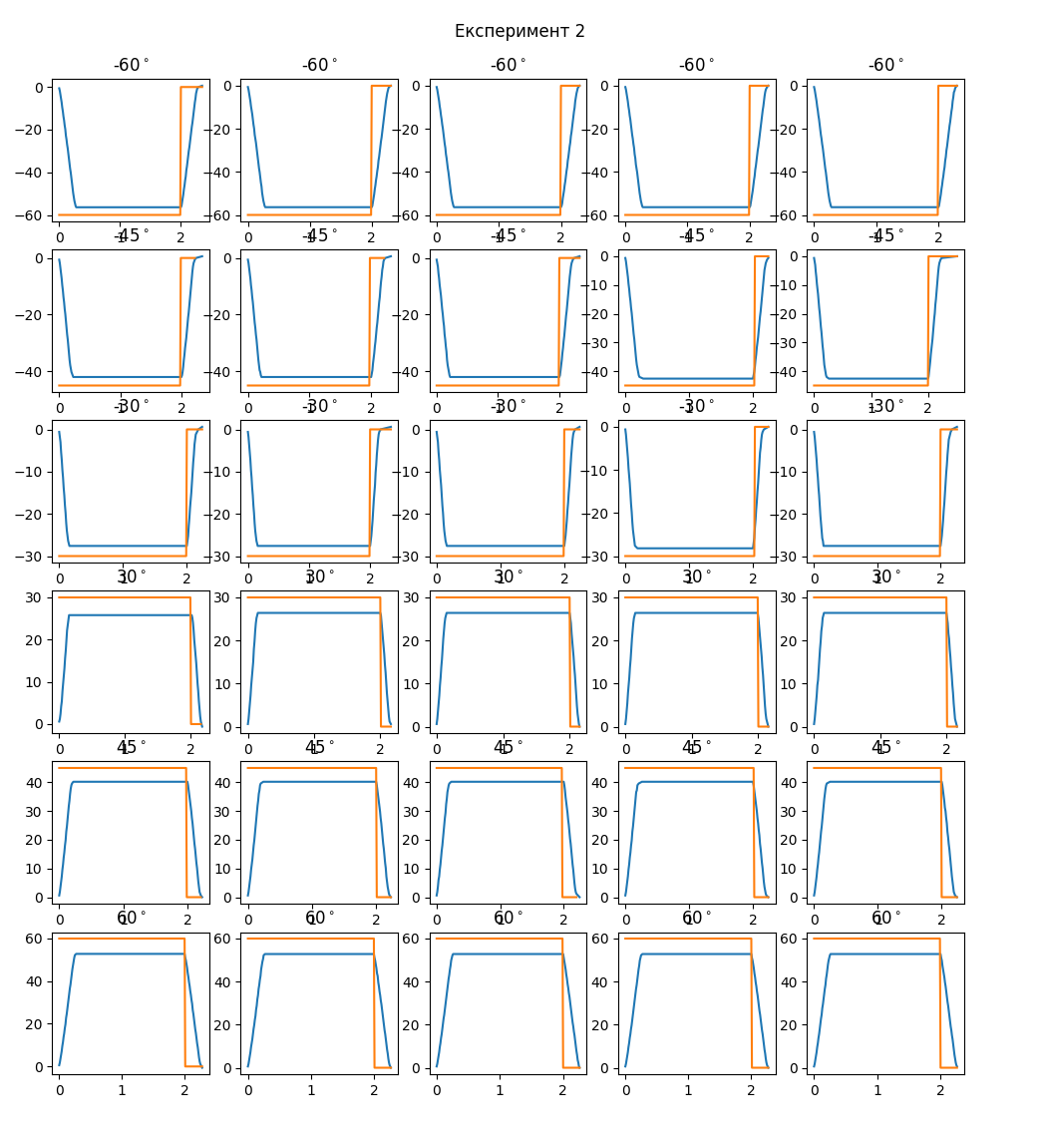
*Подаци након сређивања – Слика 3.11*

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

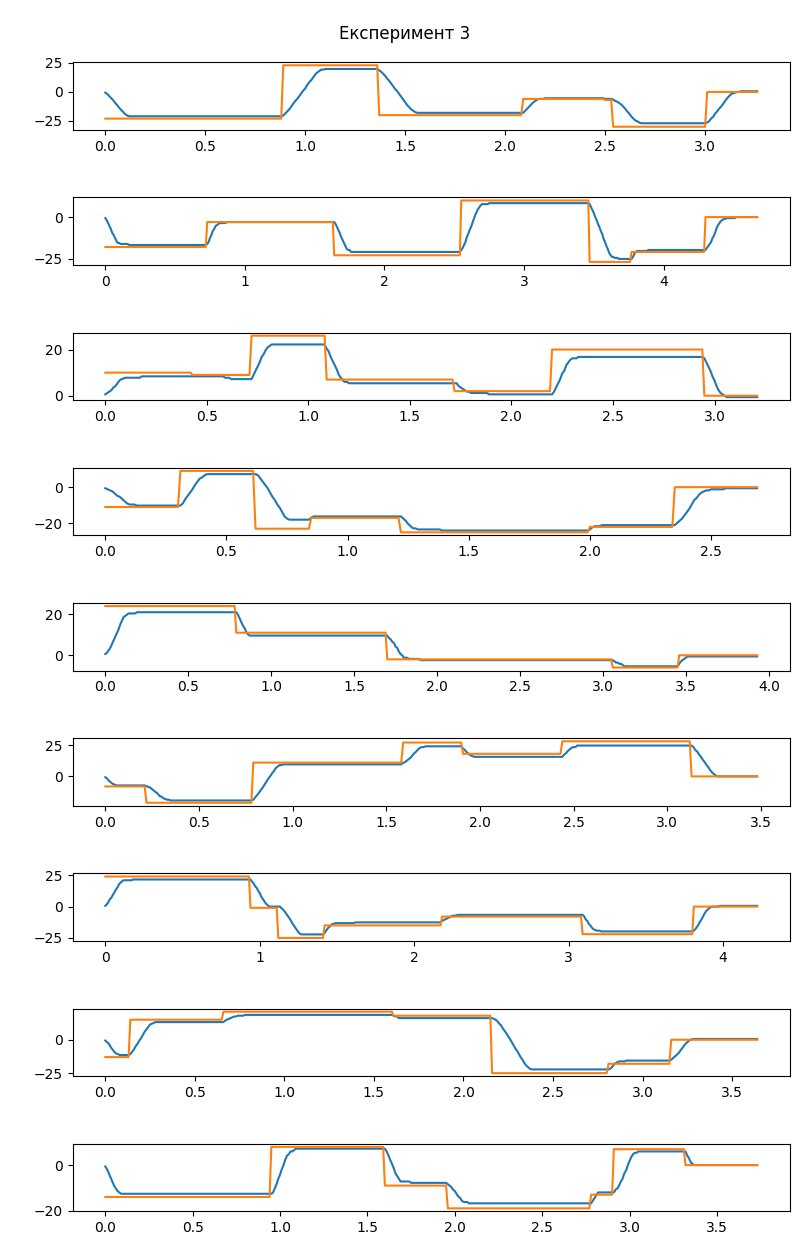
Сви обрађени подаци су графички приказани на сликама испод.



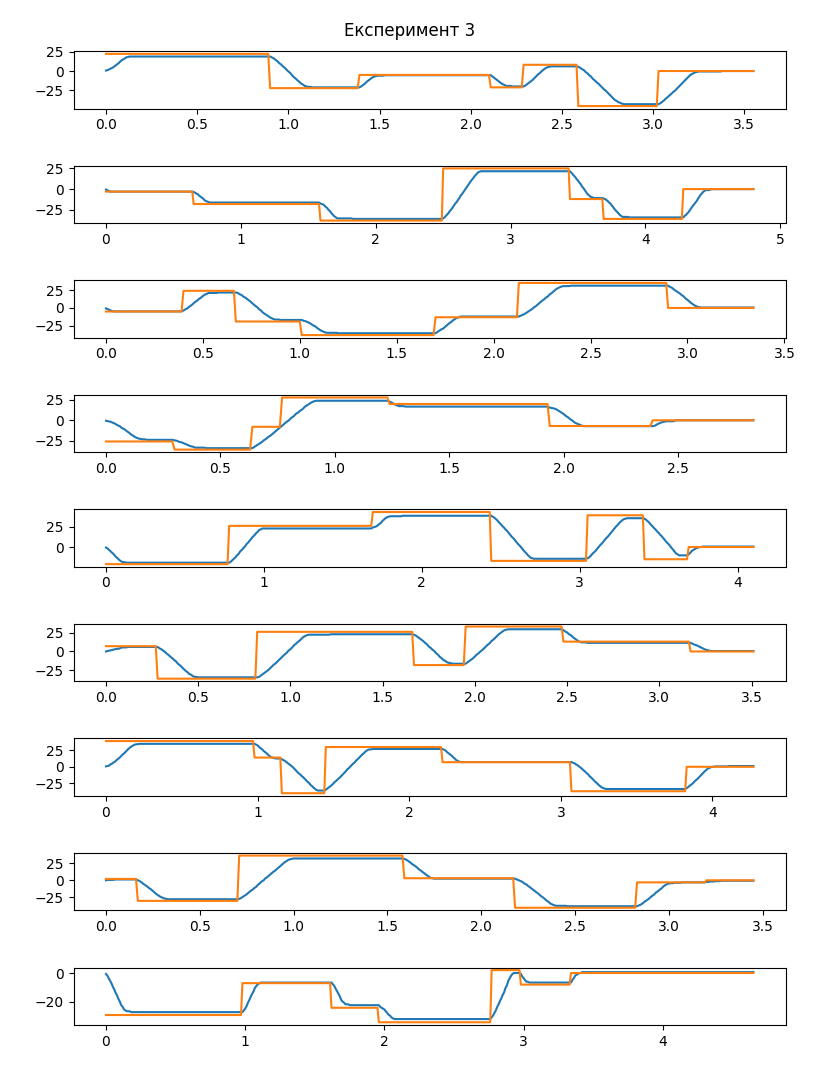
*Графички приказ података из Експеримета 1 – Слика 3.12*



*Графички приказ података из Експеримета 2 – Слика 3.13*



*Графички приказ података из Експеримета 3 Конфигурације 1 – Слика 3.14*



*Графички приказ података из Експеримета 3 Конфигурације 2 – Слика 3.15*

### Идетификација система методом сиве кутије

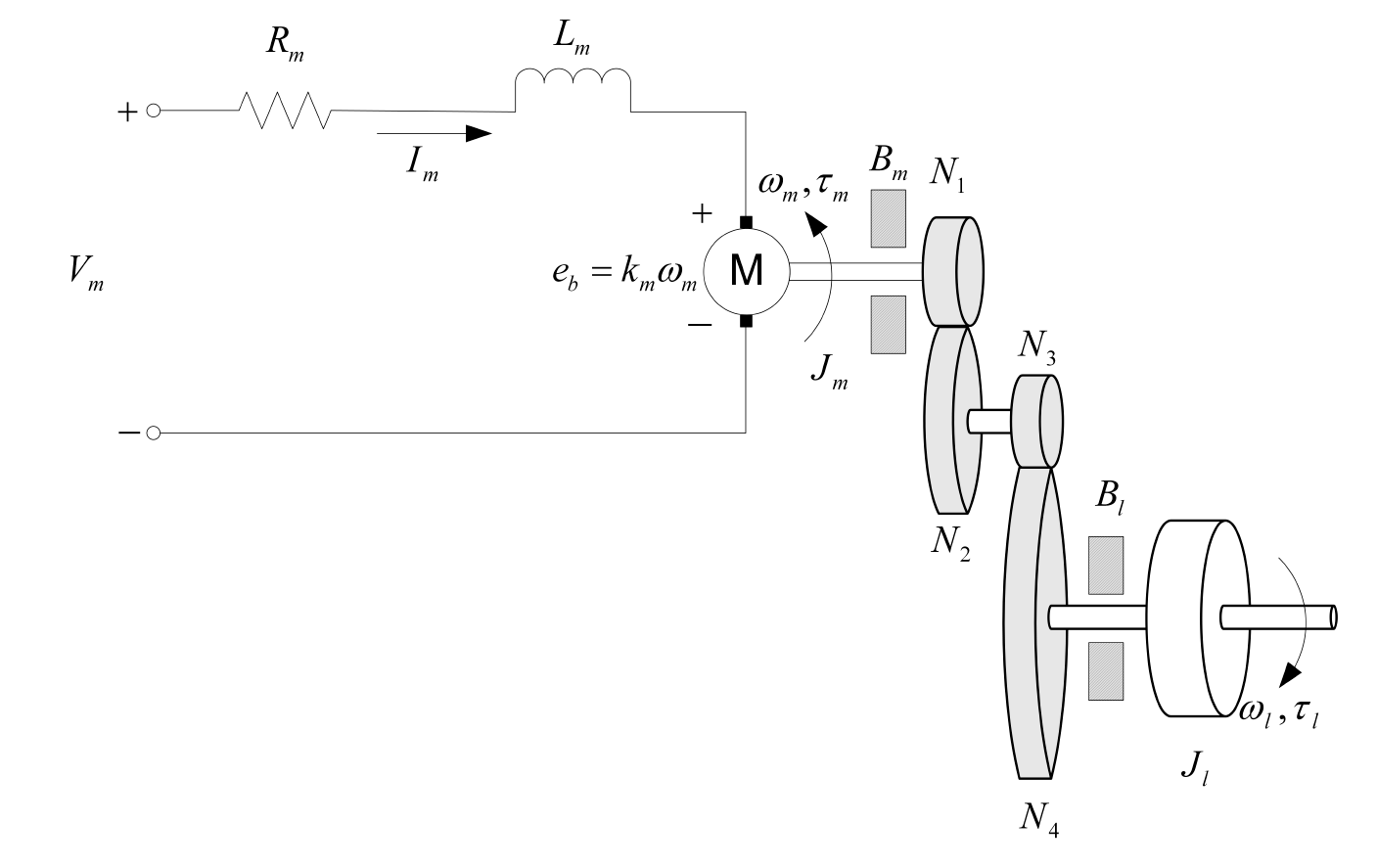
Након прикупљања података следећи корак је идентификација систем. У овом делу се фокусирамо на идетификацију система методом сиве кутије и кренућемо и први корак ће бити анализа самог система, тј. серво мотора једносмерне струје.

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 који је задужен да излазно вратило доведе у жељене позицију. На основу ових података јасно нам је да је систем затворен и да можемо доћи до структуре модела налажењем општег математичког модела спреге мотора једносмерне струје и редуктора.

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

Шема спреге 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 вратила целог система,
* 𝜏l – обртни момент целог система ,
* 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**

Кренимо од другог Кирхофог закона који каже да сума свих напона у контури је једнака нули:

1. .

Узевши у обзир да је идуктивност мотора много мања од отпорнисти мотора, предходну једначину можемо да запишемо у следећем облику:



односно

1. .

**Моделовање механичког дела система**

Кренимо од другог Њутновог закона која, који каже да производ момента инерције тела и угаоног убрзања даје обрни момент.



На основу дефиниције Њутновог закона можемо доћи до следећиј једначина, где је (3.2.4) једначина за обртни момент излазног вратила система, а (3.2.5) једначина обртног момента на излазном вратилу електро мотора:

1. ,
2. .

Једначину обртног момента излазног вратила система можемо записати и у следећем облику:

1. ,

где су:

1. .

Односно релацију (3.2.6) можемо записати и у следећем облику:

1. .

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

1. ,

односно зависност њихових угаоних брзина:

1. .

На основу предходних релација можемо доћи до следећег израза,

1. ,

и ако уведемо смене:

1. и
2. ,

можемо доћи до следеће једначина:

1. .

**Комбиновање електричног и механичког система**

Комбинациом ова два система доћи ћемо до у општеног математичког модела спреге електро мотора једносмерне струје и редуктора.

Обрни момент мотора можемо предтавити и следећом једначином:

1. ,

коју, на основу (3.2.2), можемо даље развити у следећи обли:

односно, на основу (3.2.10):

1. .

Убацањем једначине (3.2.14) у прдходну добијамо:

1. .

Док сређивањем једначине (3.2.18) добијамо:

Односно:

1. ,

где је:

1. и
2. .

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

1. ,

или следећем облику.

С обзиром да смо ми заинтересовани за промену угла излазног вратила, итеграљењем предходне једначине добићемо жељени облик преносне функције спреге електро мотора једносмерне струје са редуктором:

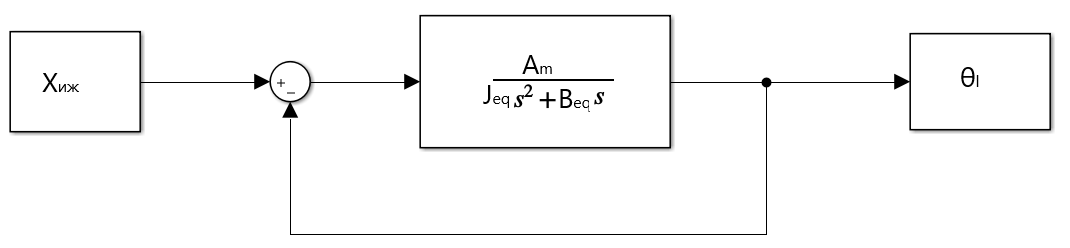
#### Одређивање математичког модела

Већ нам је познато да је серво мотор, сам по себи, један затворен систем. На улаз серво мотора је потребно довести одговарајући напон и он би на основу њега ротирао излазно вратило за одређени угао. Ми смо за експерименте користили библиотеку која нам је омогућила да за улазни сигнал задамо жељени угао, тако да ћемо у овом случају за модел серво мора усвојити следећи облик преносне функције:

1. ,

где је Хиж жељени угао.

На основу предходно реченог, претпоставимо да наш серво мотор може бити описан следећим блок дијаграмом (слика 3.17).



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

На основу блок дијаграма можемо доћи до следеће преносне функције:

1. .

Преносна функција (3.2.27) представља структуру нашег модела и следећи кораци јесу естимација параметара преносне функције, као и валидација самог модела. Естимација параметара и први део валидације је одређен уз помоћ 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

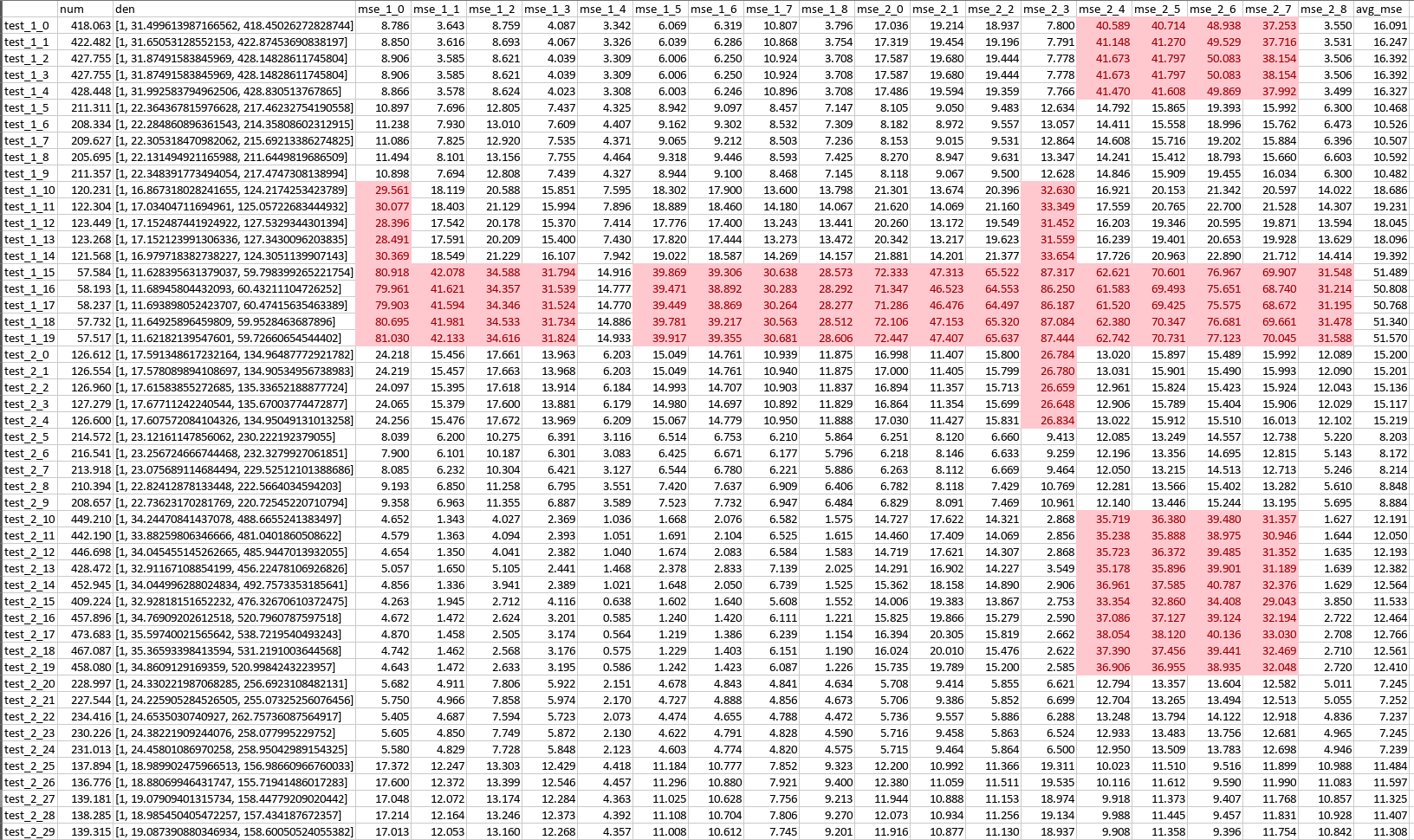
MSE = errorData;

end

Ова скрипта може да се подели у неколико делова:

* Иницијализација – део на почетку скрипте где се дефинишу бројеви полова и нула (np и nz),
* Естимација параметара преносне функције – где смо за сваки случај из прва два експеримента одредили параметре преносне функције
* Валидација модела – овде смо искористили податке из свих случајева трећег експеримента, улазима побудили преносне функције, а затим упоредили симулиране и стварне резултате. Излаз ове функције је низ средњих квадратних грешака.
* Чување података – на крају смо све естимиране вредности и низове средњих квадратних грешака сачували у датотеку json формата.

За даљу анализу податке из датотеке смо претворили у табелу која је приказана на слици 3.18.



*Табела естимираних вредности и средњих квадратних грешака за преносну функцију (3.2.27) – Слика 3.18*

Анализирајући податке из табеле могло се закључити:

* Да су случајеви првог експерименте, где је жељена вредност позиције излазног вратила 90 степени, лоши;
* Случајеви се случајеви где је жељена вредност позиције излазног вратила ±30 степени нису добро показали за половину случајева трећег експеримента где је ограничење било од -30 до 30 степени;
* Да две трећине случајева има просечну средњу квадратну грешку мању од 13%.

За даљу анализу смо усвојили преносну функцију (3.2.28) која је имала најмању просечну средњу квадратну грешку:

1. .

За даљу анализу модела искористили смо Matlab скрипту tfValidation.m. Уз помоћ скрипте смо дошли до графика излаза естимиранонг система и стварног система, као и графика ауто корелације и унакрсне корелације резидуала.

На основу излаза може се закључити да математички модел хобу серво мотора, описан преносном функцијом (3.2.28), ефикано хвата динамику стварног система. Док на основу резидуалне анализе можемо закључити:

* на основу ауто корелације – да моделу не покрива у потпуности динамику система,
* на основу унакрсне корелације – да модел добро покрива есецијална улазно-излазна понашања.

*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

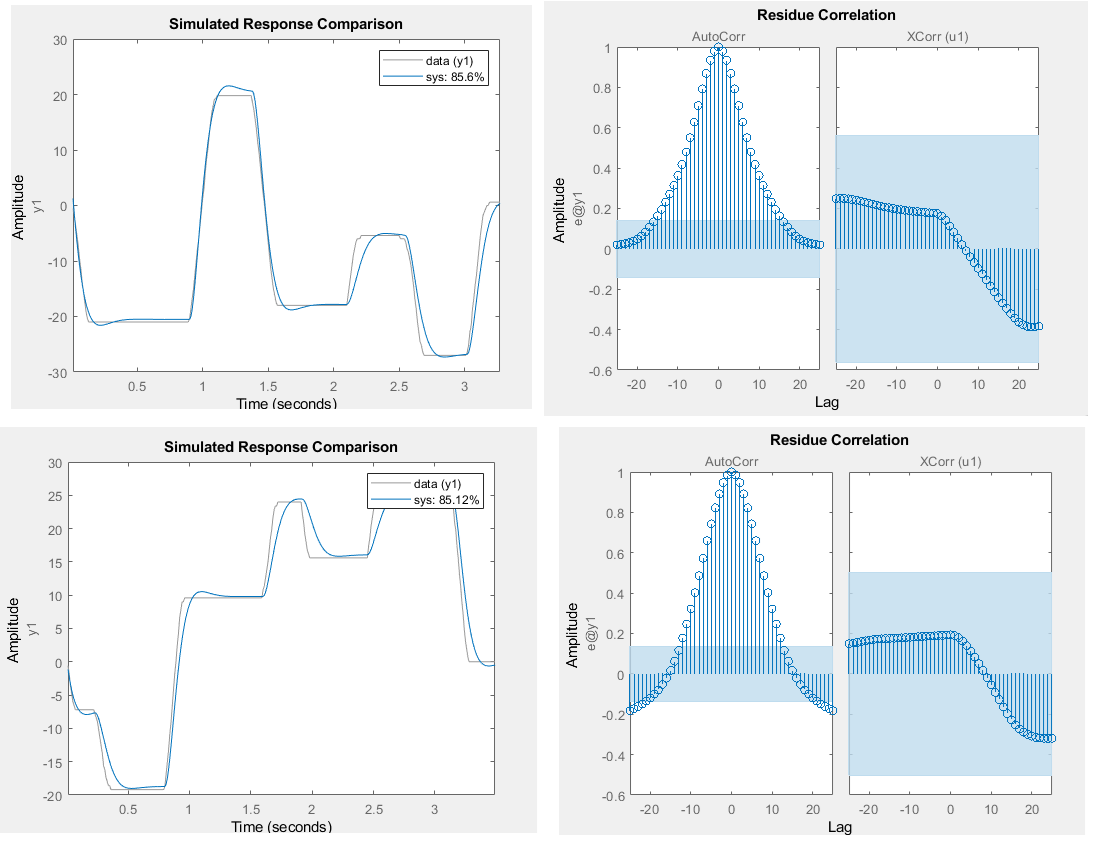
disp(ME)

end

end

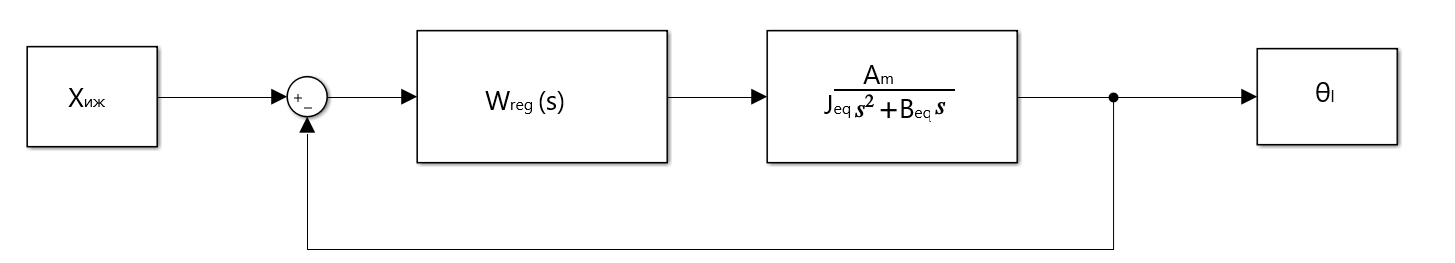
end

end



*Поређење симулираног и стварног излаза (лево); графици ауто корелације и унакрсне корелације резидуала (десно) — Модел 1 – Слика 3.19*

Резултати предходног модела делују добор, али пробајмо да дођемо до још бољих резултата додавањем PD регулатроа. Нови модел приказан је блоком дијаграмом на слици 3.20.



*Блок дијаграм серво мотора једносмерне струје (модел 2) – Слика 3.20*

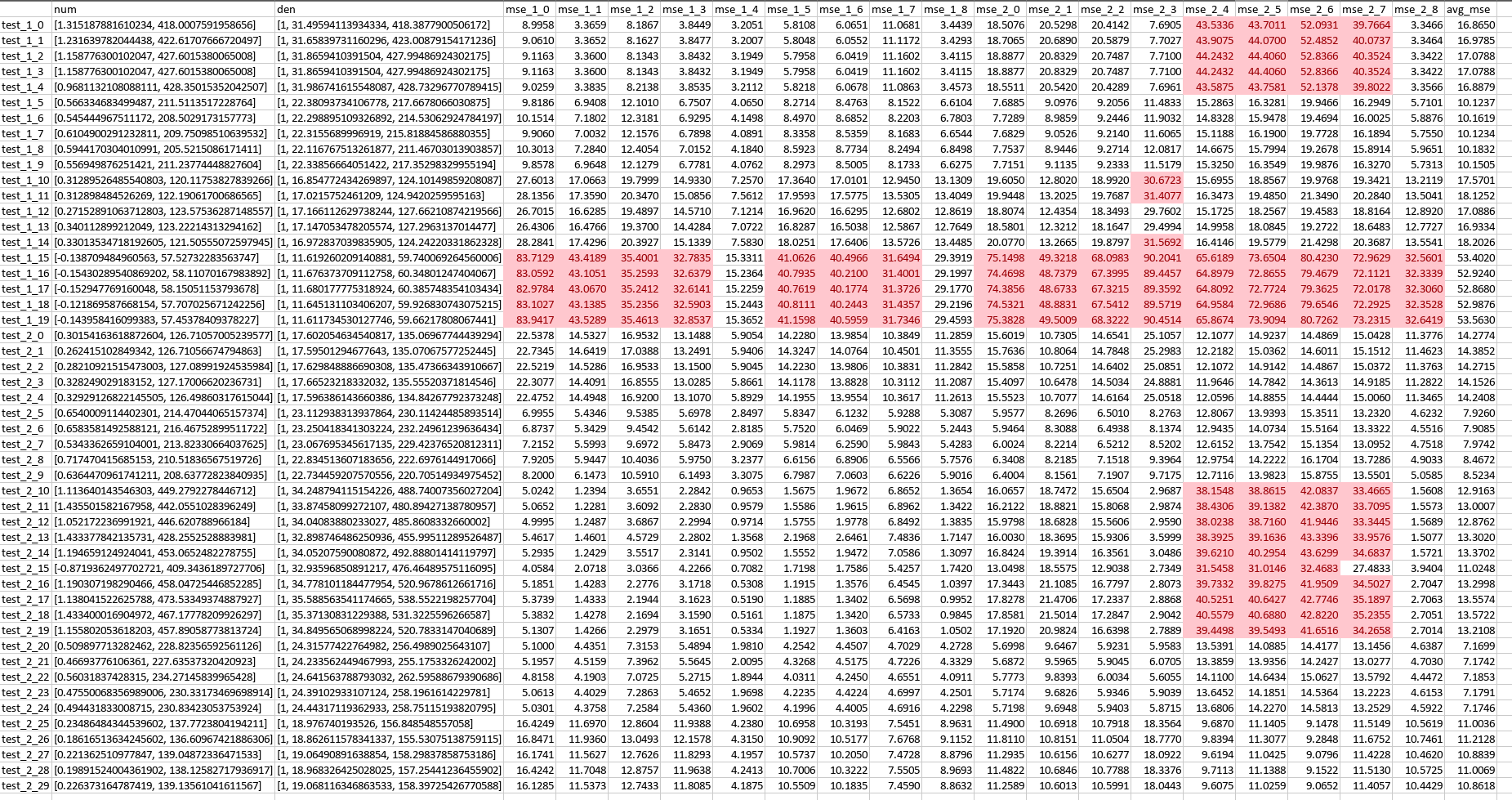
Где је:

1. ,

док сређивањем блок дијаграма можемо доћи до следеће преносне функције:

1. .

Када наш нови модел провучемо кроз Matlab скрипту tfEstimation.m, долазимо до следећих резултата, слика 3.21.



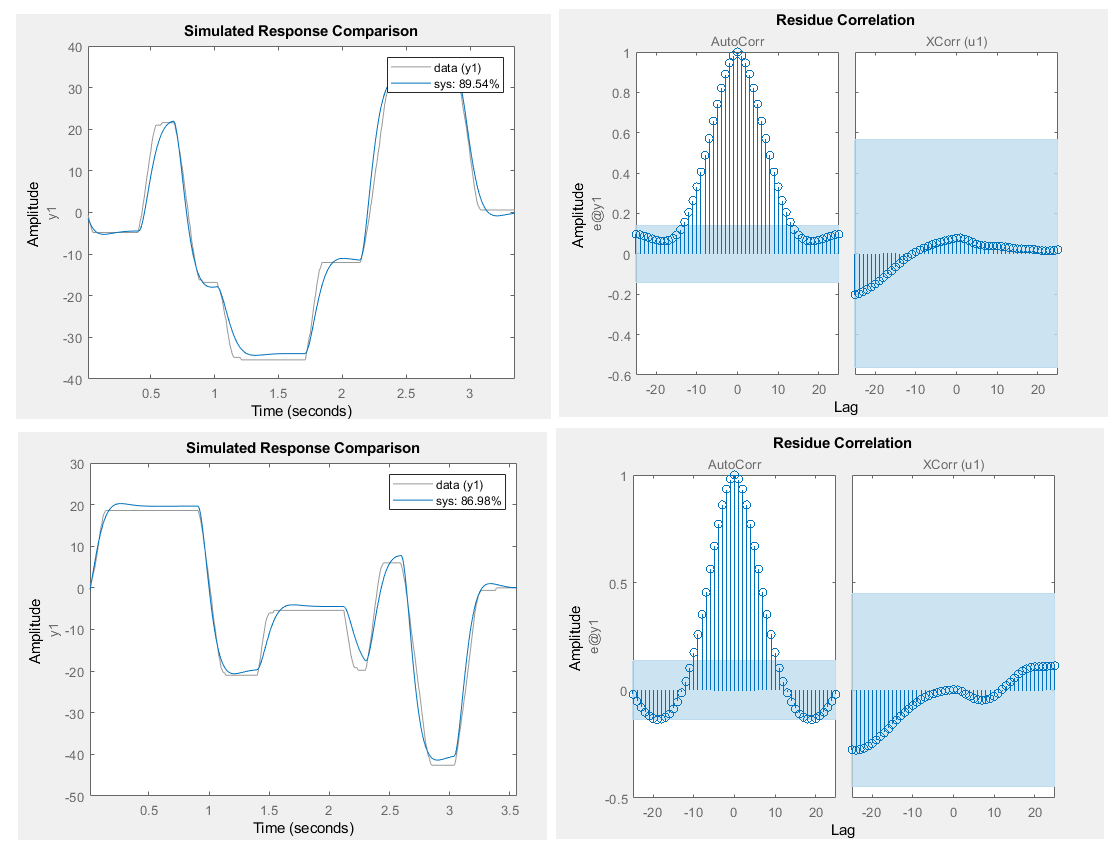
*Табела естимираних вредности и средњих квадратних грешака за преносну функцију (3.2.30) – Слика 3.21*

На основу анализе можемо закључити да је дошло до врло малог побољшања, али и не значајног. Пратећи кораке из предходне итерације нову преносну функцију (3.2.31) смо поново провукли кроз скрипту tfValidation.m и добили нове графике.

1. .

На основу графика са слике 3.22, такође, можемо видети да није дошло до значајног побољшања модела.

На основу предходних анализа и узимајући у обзир да смо за конструкцију модела узели линеарни модел претстављен преносном функцијом можемо закључити да смо дошли до максимума и ако би смо хтели да дођемо до модела који има бољу подударност, ауто корелацију и унакрсну корелацију потребно је изабрати други тип конструкције.



*Поређење симулираног и стварног излаза (лево); графици ауто корелације и унакрсне корелације резидуала (десно) — Модел 1 – Слика 3.22*

### Идетификација система методом црне кутије

У овом делу покушаћемо да одредимо математички модел серво мотора користећи метод црне кутије.

Водећи се истраживачким радом “Deep Learning and System Identification” аутора Ленорта Љунга, Карла Андерсона, Коена Тилса и Томаса Б. Шона, одредићемо математички модел користећи дубоку каскадну неуралну мрежу у комбинацији са нелинеарнинм ауторегресивним моделом са егзогеним улазима (NARX).

*Дефиниција 5* **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.

Каскадна неуронска мрежа је врста неуронска мреже која порширује основну архиктектуру неуронске мреже са ширењем унапред додавањем директних веза између улазног слоја и свих наредних слојева. Ово омогућава мрежи да обухвати сложеније односе између улаза и излаза пружајући више путева за проток информација.

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 тренирање мреже, а осталих пет смо такође спојили и искористили за валидацију. Након покретања Matlab скрипте bbModelEstimation.m добили смо следеће резултатe.

Скрипта — bbModelEstimation.m

— skripta —

— slika neuronske mreze —

— grafici —

Zakljucak

# Пројетовање и имплементација регулатора

# Упоређивање симулације и стварног модела

# Закључак

# Додатак

## Додатак А – Техничка документација стандардних компоненти

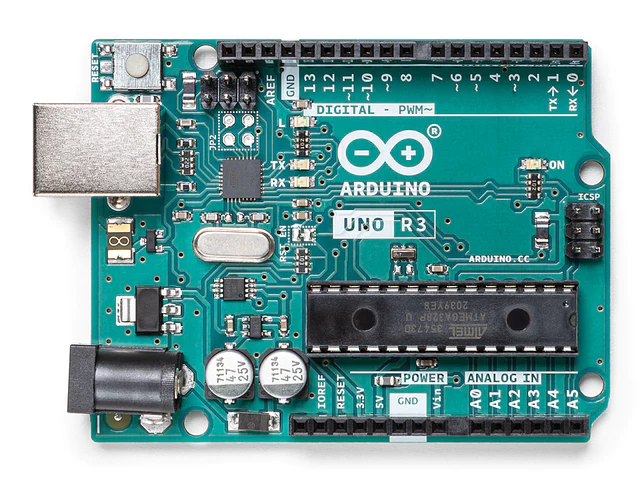
### Микроконтролер – Arduino Uno

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.

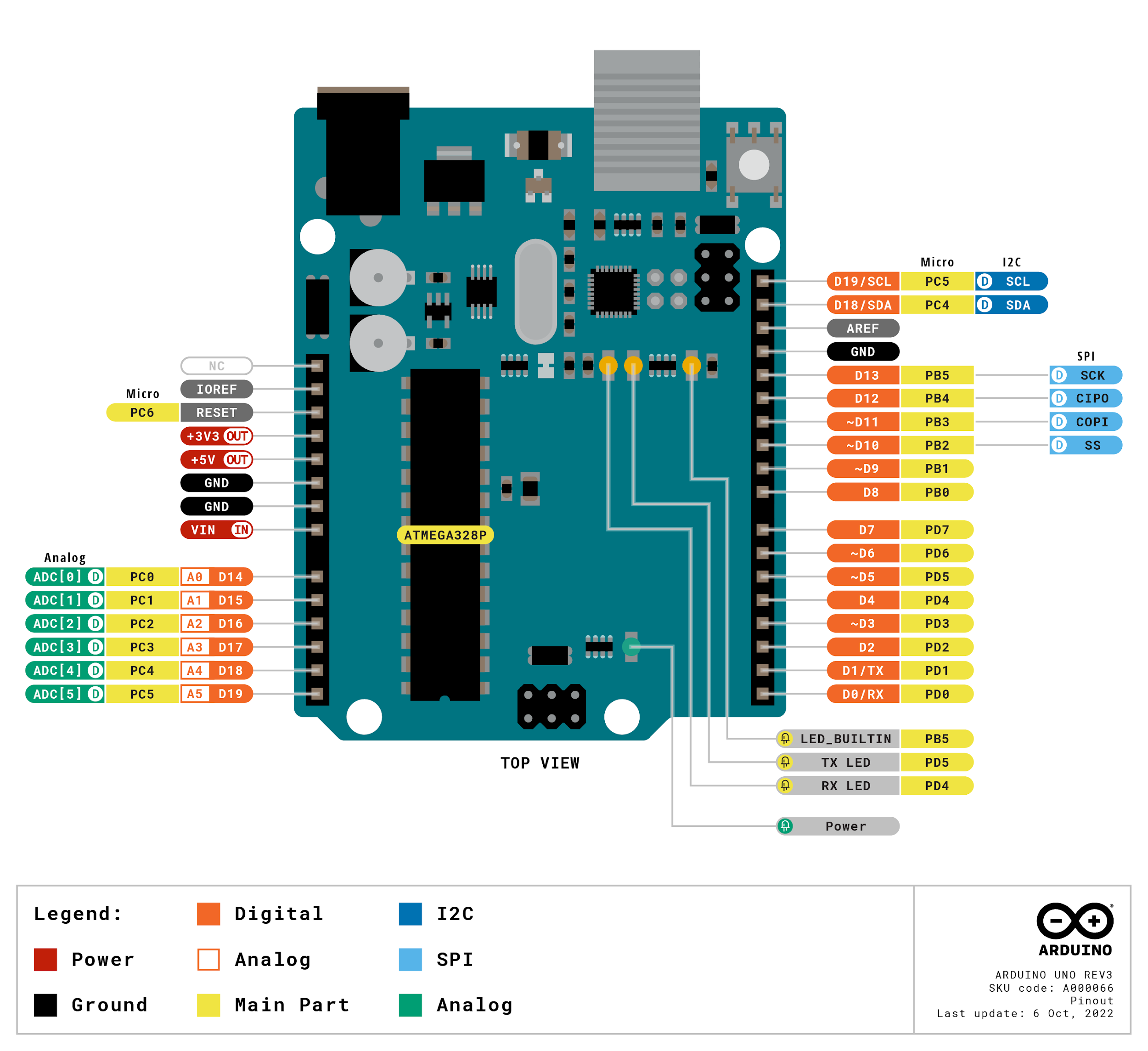
Техничке карактеристике и шема улазно/излазних пинова овог микроконтролера су приказани у табели 7.1 и слици 7.1.



*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

Адитивна производња, позната и као 3D штампање 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, добар део делова је израђен на 3D штампачу Ender 3 Pro (слика 7.7) компаније Creality. Овај штампач користи технологију слој по слој (FDM), а његове основне карактеристике можете видети у табели 7.6.



*Ender 3 Pro - Слика 7.7*

|  |  |
| --- | --- |
| 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 |
| Дебљина слоја | 0.1–0.4mm |
| Повезвање | USB, SD картица |
| Материјали за штампање | PLA, ABS, PETG, TPU |

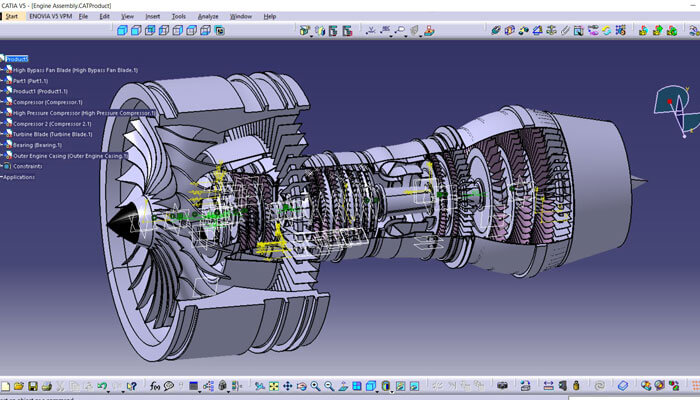
*Техничке карактеристике штампача Ender 3 Pro – Табела 7.6*

## 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>