Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Рачунарска техника и софтверско инжењерство

Ниво студија: Основне академске студије

Модул: Општи

Предмет: Основи електротехнике

Број индекса: 557/2015

Невена П. Сташић

Серво мотори у примени троосне стабилизације

завршни рад

Комисија за преглед и одбрану:

1. **др Јасна Радуловић, ред. проф.**
2. _____
3. _____

Датум одбране:_____

Оцена:_____

У оквиру овог завршног рада кандидат треба да истражи и проучи литературу на задату тему и на бази тога детаљно и конкретно опише цео процес реализације једног оваквог система, од почетног прикупљања потребних компоненти, преко софтверске реализације до утврђивања недостатака развијеног система.

Препоручена литература:

[1] Владан Вучковић: Електрични погони, Електротехнички факултет, Београд 1997.

[2] Слободан Вукосавић: Дигитално управљање електричним погонима, Академска мисао, Београд 2003.

[3] Милић Р. Стојић: Дигитални системи управљања, N наука, Београд 1994.

[4] Reference manual STM32F advanced Arm-based 32bit MCUs, Приступљено: 21.08.2019. године, линк: https://www.st.com/resource/en/datasheet/cd00220364.pdf

Крагујевац, 26. 08. 2019. године

Ментор:

др Јасна Радуловић, ред. проф.

_____

Садржај:

Резиме:

У овом раду се говори о серво моторима и њиховој примени у троосној стабилизацији. Одабран је AC серво мотор MS серије и њему одговарајући драјвер (driver) компаније Xinje и ознаке DC2 – AS. Сваки код који је коришћен за развој нашег система, приказан је и сликовито, поред текстуалног описа.
Након развијеног система за симулацију кретања борбеног возила и стабилизацију кретања, потребно је реализовати аквизицију и мерење динамике самог система. Универзални систем за мерења састоји се из хардверског модула и софтвера развијеног на бази LabView пакета. Као једини недостатак троосне стабилизације јавља се то што је он развијен само у три осе, а у будућности, настојаћемо да развијемо систем са шест оса, како би стабилизација била прецизнија и тачнија. Тежићемо развоју Стјуартове (Stewart) платформе.

Кључне речи: AC серво мотор MS, компанија Xinje и ознаке DC2 – AS, LabView, Stewart платформа.

Summary:

This paper discusses servo motors and their application in triaxial stabilization. The choice fell on the MS servo AC servo motor and the corresponding Xinje driver and DC2 - AS marks. Each code that was used to develop our system is presented in a colorful way, in addition to the textual description.

After developing a system for simulating the movement of a combat vehicle and stabilizing movement, it is necessary to realize the acquisition and measurement of the dynamics of the system itself. The universal measurement system consists of a hardware module and software developed on the basis of the LabView package. The only disadvantage of three-axis stabilization is that it is developed in only three axes, and in the future, we aim to develop a six-axis system, in order to make the stabilization more accurate and accurate, we strive to develop the Stewart platform.

Keywords: AS servo motor MS, Xinje company and DC2 designations - AS, LabView, Stewart platform.

1. Предговор

У овом раду биће речи о серво моторима, њиховим карактеристикама и врстама серво мотора. Кроз даљу израду рада приказаћемо који се то серво мотори користе у примени троосне стабилизације.

За развој нашег система троосне стабилизације, неопходно је прво направити систем који ће симулирати одређено кретање борбеног возила и чије померање треба бити стабилисано. Одабран је AC серво мотор MS серије и њему одговарајући драјвер компаније Xinje и ознаке DC2 – AS.

Сваки код који је коришћен у пракси, биће приказан сликовито и објашњен текстуално. Након развијеног система за симулацију кретања борбеног возила и стабилизацију кретања, потребно је реализовати аквизицију и мерење динамике самог система. Универзални систем за мерења састоји се из хардверског модула и софтвера развијеног на бази LabView пакета са више мерних инструмената и метода. LabView програми се називају виртуелни инструменти или VI, зато што њихова појава и операције имитирају физичке инструменте попут осцилоскопа и мултиметра. LabView садржи сет алата за прикупљање, анализу, приказивање и смештање података, као и алате за отклањање проблема у коду.

За управљање аквизиционом картицом неопходан је DAQ софтвер, који контролише систем аквизицијом изворних података, анализом и презентовањем резултата.

2. Увод

Сервомеханизам или серво је систем управљања са повратном спрегом код кога је управљана променљива механичка позиција или кретање. Основна функција сервомеханизма је да његов излаз брзо и прецизно прати промене улаза. Серво систем обезбеђује три повратне везе: позициону, брзинску и струјну. [1]

Серво мотори су самостални електрични уређаји који ротирају или потискују делове машине с великом прецизношћу. Састоје се од одговарајућег мотора у комбинацији са сензором који даје информацију о позицији. Серво мотори такође садрже и релативно софистицирани контролер, често одвојен модул, намењен за употребу са серво моторима. RC (Radio Control) серво мотори се најчешће користе у апликацијама са даљинским управљањем (мали роботи, даљински управљани аутомобили, авиони, бродови итд). [2]

Основни облик дејствовања борбених возила је гађање из покрета где се услед кретања возила по терену јављају линијски и угаони помераји нишанске осе и осе оруђа. Да би се умањиле грешке које настају услед кретања возила врши се жироскопска стабилизација оруђа. Стабилизација, бар по елевацији, неопходна је и да би било могуће рафално гађање. Потпуна стабилизација захтева реализацију два независна серво-система, један за покретање куполе по правцу (азимуту) и други за покретање цеви топа по висини (елевацији). Блок-шема оба ова система је иста, само се разликују параметри система. [6]

3. Теоријска разматрања

3.1. Сервосистем (Сервомеханизам)

Сервомеханизам или сервосистем је систем управљања са повратном спрегом код кога је управљана променљива механичка позиција или кретање. Основна функција сервомеханизма је да његов излаз брзо и прецизно прати промене улаза. [1]

Сервомеханизам функционише на тај начин што се излазна величина стално мери и посредством повратне спреге упоређује са улазном величином, дејствујући при том тако да грешку која настаје као резултат поређења излазне и улазне величине своди на нулу, уз појачавање снаге. Код сервомеханизама су излазне величине механички положај и његови изводи. Ако би код сервомеханизма улазна величина имала константну вредност онда би и излазна величина била константна, па би у том случају сервомеханизам вршио функцију аутоматског регулисања (регулатора). У зависности од примењене енергије за појачање снаге, сервомеханизми могу бити:

- електрични,
- хидраулични,
- пнеуматски и
- комбиновани.

Серво систем обезбеђује три повратне везе:

- позициону,
- брзинску и
- струјну.

Повратна спрега која враћа информацију по позицији – позициона повратна спрега, служи за угаоно позиционирање мотора, при томе излазна величина из позиционе петље је брзинска команда. Позициона петља добија информације о позицији са енкодера. [1]

Повратна спрега која враћа информацију о брзини окретања – брзинска повратна спрега, служи за контролу брзине окретања мотора, при томе излазна величина из брзинске петље је струјна команда. Брзинска петља добија информације о брзини са енкодера.

Повратна спрега која враћа информацију о струји – струјна повратна спрега, служи за контролу командне струје серво мотора.

У оваквом систему извршни орган је серво мотор, који је елемент директне гране система аутоматског управљања којим се непосредно мења извршна величина. У табели 1. је приказана класификација серво мотора. [1]

Табела 1. Класификација сервомотора

СЕРВО МОТОРИ – Класификација:
1. DC (direct current) серво мотор
2. AC (alternating current) серво мотор: ➢ Синхрони серво мотор и ➢ Индукциони тип серво мотора
3. Корачни мотор

3.2. Функционисање серво мотора

Једноставност серво мотора је једна од карактеристика која их чини толико поузданим. Срце серво је мали мотор једносмерне струје DC, сличан ономе у јефтиној играчки. Ови мотори покрећу електричну енергију из батерије и окрећу се са високим бројем обртаја у минути (ротације у минути), али стављају врло мали обртни момент (сила која се користи за рад - примењујете обртни момент када отворите посуду). [2]

Уређење зупчаника узима велику брзину мотора и успорава га док истовремено повећава обртни момент. Мали електромотор нема много обртног момента, али се може брзо окретати (мала сила, велико растојање). Дизајн зупчаника у серво кућишту претвара излаз на много спорију брзину ротације, али са више обртног момента (велика сила, мало растојање). Зупчаници у типичном серво уређају су направљени од пластике и конвертују брзо покретање мотора (са десне стране) на излазну осовину (са леве стране), приказано на слици 1. [2]

Слика 1. Типичан серво серво уређај [3]

У серво-уређају високе снаге, пластични зупчаници се замењују металним за снагу. Мотор је обично моћнији него у јефтинијем серво-у, а укупни излазни обртни момент може бити чак 20 пута већи од јефтиније пластике. Бољи квалитет је скупљи, а сервиси високог излаза могу коштати два или три пута више од стандардних, приказано на слици 2. [2]

Слика 2. Серво уређај високе снаге [3]

Са малим DC мотором, примењује се напајање из батерије, а мотор се врти. Међутим, за разлику од једноставног DC мотора, серво вратило мотора се успорава са зупчаницима. Сензор позиције на завршном реду је прикључен на малу плочу, што је приказано на слици 3. [2]

Слика 3. Спојница и ДЦ мотор у серво уређају високог напона [4]

Сензор говори о овој плочи колико је серво излазна вратила ротирао. Електронски улазни сигнал са рачунара или радија у возилу са даљинским управљањем се такође уводи у ту плочу. Електроника на плочи декодира сигнале како би утврдила колико далеко корисник жели да се серво ротира. Затим упоређује жељену позицију са стварним положајем и одлучује који смер ротира осовину тако да стигне до жељеног положаја. [2]

3.3. Врсте серво мотора

Постоје три основне врсте серво мотора: ротација положаја, континуална ротација и линеарна. [6]

3.3.1. Ротација положаја

Серво серво ротације : То је најчешћи тип серво мотора. Излазна вратила ротира у око пола круга, или 180 степени. Има физичке зауставе постављене у механизам преноса да би се спречило окретање ових граница како би се заштитио сензор ротације. Ови уобичајени серво мотори се налазе у ауто-контролисаним аутомобилима и воденим и ваздухопловним уређајима, играчкама, роботима и многим другим апликацијама. [6]

3.3.2. Континуална ротација

Серво константне ротације: Ово је сасвим слично са заједничким серво моторима ротације, осим што се у било ком смеру може окретати на неодређено време. Контролни сигнал, уместо постављање статичког положаја серво-а, тумачи се као смер и брзина ротације. Опсег могућих команди узрокује да се серво окреће у смеру казаљке на сату или у супротном смеру казаљке на сату по жељи, при различитим брзинама, у зависности од командног сигнала. [6]

3.3.3. Линеарна ротација

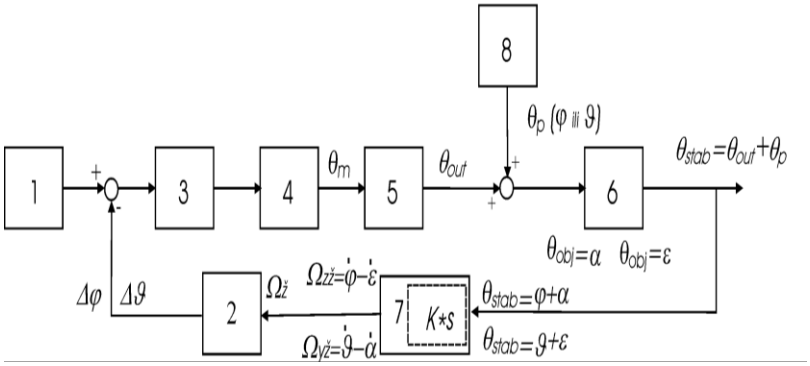
Линеарни серво: Ово је исто као серво мотори позиционирања који су горе описани, али са додатним зупчаницима (обично зупчасти и зупчасти механизам) за промену излаза са кружног на назад и назад. [6]

3.4. Блок – шема стабилизације

За мерење угаоне брзине поремећаја користе се два брзинска жироскопа као сензори. Један мери угаону брзину поремећаја (односно грешке стабилизације) по правцу, а други по висини. Блок-шема оба ова система је иста и приказана је на слици број 4. , само се разликују параметри система: оптерећење, погонски уређај, редуктор, гранична учестаност, итд. [6]

Блокови на слици обављају следеће функције:

- управљачки сигнал из система за праћења циља (нишанџија, систем за управљање ватром - СУВ),
- интегратор (израчунава грешку стабилизације),
- компензатор, генерише управљачки сигнал за погонски уређај,
- погонски уређај за покретање објекта (цев топа по висини) са повратним спрегама за контролу угаоне брзине,
- редуктор - као нелинеарни елемент,
- објекат управљања (купола са топом по правцу, цев топа по висини),
- блок сензора апсолутне угаоне брзине закретања, брзински жироскопи, и
- поремећај услед кретања возила. [6]



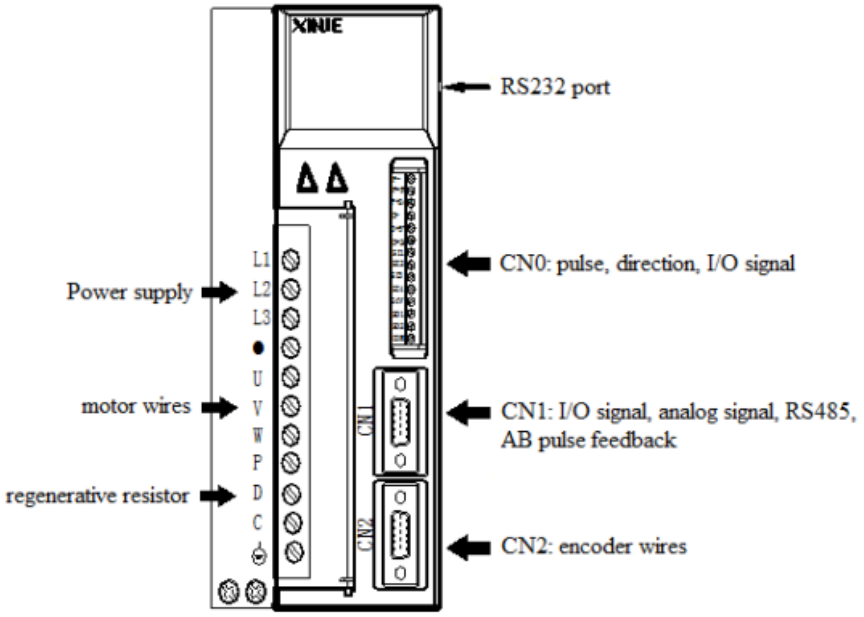
Слика 4. Блок – шема сервосистема за стабилизацију [5]

4. Практични део рада

4.1. Развој система за троосну стабилизацију

4.5.1. Избор потребних компоненти и блок шема самог система

За развој нашег система троосне стабилизације, неопходно је прво направити систем који ће симулирати одређено кретање борбеног возила и чије померање треба бити стабилисано. Обзиром да ћемо се у овом раду бавити искључиво развојем система са гледишта електронике, све остале потребне делове искористићемо као готове производе машинског и механичког сектора. Сада када имамо све потребне машинске и механичке делове, можемо приступити одабиру мотора који ће се користи. Одабир је пао на AC серво мотор MS серије и на њему одговарајући драјвер компаније Xinje и ознаке DC2 – AS. Блок шема самог драјвера приказана је на слици 5. [7]



Слика 5. Блок шема драјвера [7]

Изглед АС серво мотра који је коришћен је приказан на слици 6.

Слика 6. АС серво мотор [8]

Оно што је још потребно пре креирања овог система симулације кретања возила јесу сензори метала, уз помоћ којих ћемо направити више различитих секвенци – праваца кретања.

Слика 7. Сензор за метал [9]

За трооосну стабилизацију су потребна три мотора, три драјвера и шест сензора за метал, који ће се налазити са леве и десне стране сваког од ова три мотора.

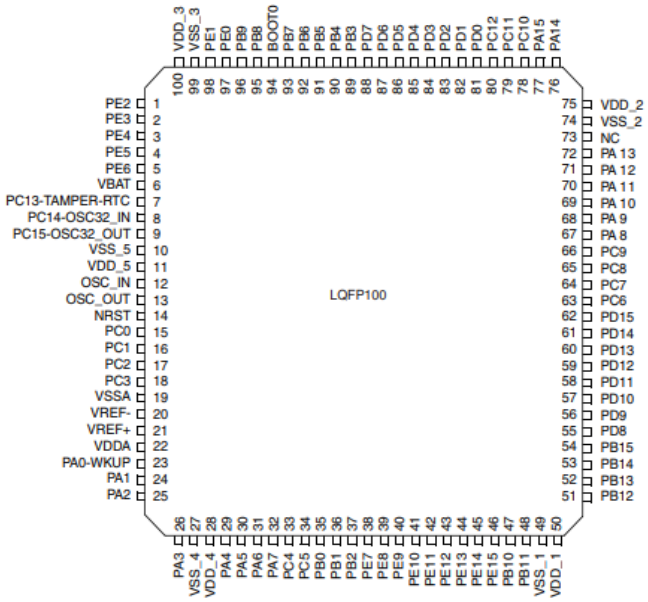
Слика 8. Систем за симулирање кретања возила [10]

4.5.2. Програмирање симулације кретања возила

Следећи корак у развоју система симулације кретања возила јесте програмирање серво мотора да раде на задати начин. За програмирање серво мотора који су повезани на драјвер марке Хиње, користићемо развојну плочу чију срж чини микроконтролер произвођача Микроелектроника, ознаке: STM32F107VC. [11]

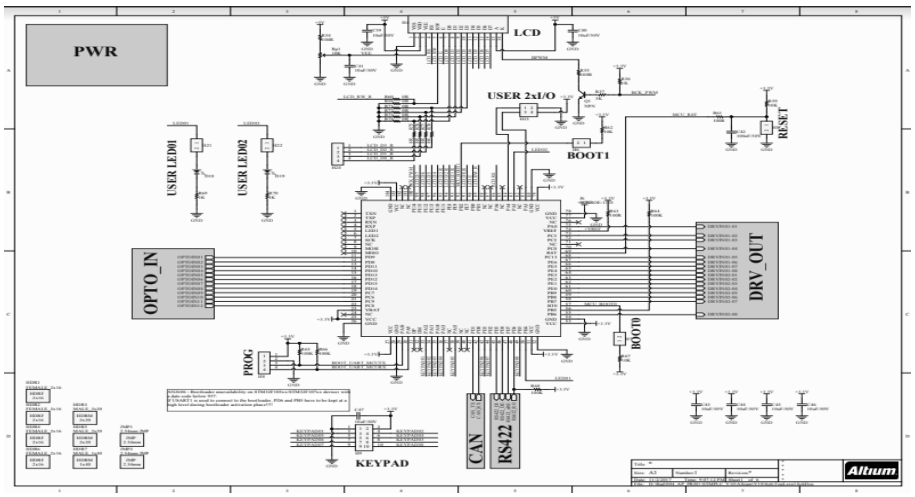
Микроконтролери из ове породице погодни су за широк спектар примена као што су: моторни погони и контрола апликације, медицинска и ручна опрема, индустријска примена, PLC – ови, претварачи, штампачи и скренери, алармни системи, видео интерфон, кућна аудио опрема итд. Породица овог микроконтролера садржи висококвалитетно ARM Cortex – M3 32 – битно RISC језгро које ради на фреквенцији 72 MHz, има уграђену меморију високих брзина и
широк спектар побољшаних input / output и периферних уређаја. [11]

На слици 9. је приказан коришћен микроконтролер.



Слика 9. Mikrokontroler STM32F107VC [11]

На слици 10. је приказана Блок шема целе развојне плоче на којој се налази и поменути микроконтролер и коју смо користили за повезивање мотора, драјвера и сензора са њим.



Слика 10. Блок шема развојне плоче [10]

Преко развојне плоче која је приказана на слици 10. , пре почетка програмирања потребно је повезати све компоненте у целину. Веома је битно проверити у упутству за коришћење микроконтролера, који пинови су слободни, које пинове смемо, а које не смемо користити. Пин спецификација самог микроконтролера дата је на сликама испод од слике 11. до слике 17.

Након успешно извршеног повезивања свих компоненти на основу блок шеме, следеће чему можемо приступити јесте писање програма и програмирање микроконтролера чиме ћемо извршити крајњу симулацију кретања возила.

За програмирање овог микроконтролера користићемо програмско окружање Mikro C for ARM.

На самом почетку писања програма, неопходно је извршити адекватно подешавање пинова, односно дефинисање пинова који се користе као излазни и оних који се користе као улазни. Тачније, извршено је подешавање улаза и то: пин PD9, пин PD10 и пин PD11, као улази са сензора и пин PD8 као button start (тастер за почетак). Функција која се користи за постављање улаза је функција:

- GPIO_DIGITAL_INPUT(&GPIO^_BASE, _GPIO_PINMASK_*)

Први аргумент ове функције дефинише назив порта (уместо ^ пише се ознака порта(А, Б, Ц..)), док други аргумент представља број тог порта (уместо * се пише број).

```
// Set GPIO_PORTD poins 8 as digital input
GPIO_Digital_Input(&GPIOD_BASE, _GPIO_PINMASK_8);

// set GPIO points 9, 10 and 11 digital input  sensors input
GPIO_Digital_Input(&GPIOD_BASE, _GPIO_PINMASK_9);
GPIO_Digital_Input(&GPIOD_BASE, _GPIO_PINMASK_10);
GPIO_Digital_Input(&GPIOD_BASE, _GPIO_PINMASK_11);
```

Слика 11. Код за подешавање улаза [10]

Подешавање излаза је урађено на следећи начин:

1. Пин PA0, пин PC3 и пин PC2, као излази за мотор број 1;
2. Пин PC0, пин PC13 и пин PE6, као излази за мотор број 2;
3. Пин ПЕ5, пин ПЕ4 и пин ПЕ3, као излази за мотор број 3.

Функција која се користи за постављање излаза је:

- GPIO_Digital_Output(&GPIO^_BASE, _GPIO_PINMASK_*);

Први аргумент ове функције дефинише назив порта (уместо ^ пише се ознака порта (A, B, C..)), док други аргумент представља број тог порта (уместо * пише се број). Након успешно извршеног подешавања пинова микроконтролера, на реду је да се изврши подешавање параметара и функција за тастатуру.

Тастатура је директно повезана са микроконтролером на следећи начин:

1. KEYPAD01 на пин PA12;
2. KEYPAD02 на пин PA11;
3. KEYPAD03 на пин PA14;
4. KEYPAD04 на пин PA15;
5. KEYPAD05 на пин PD3;
6. KEYPAD06 на пин PD2;
7. KEYPAD07 на пин PD3

Први део кода за подешавање тастатуре је приказан на слици 12. У овом делу кода дефинисане су потребне променљиве за колоне и редове матричне тастатуре, као и две функције за иницијализацију редова и колона.

```
bit col1;
bit col2;
bit col3;
bit row1;
bit row2;
bit row3;
bit row4;
unsigned char row;
unsigned char col;
int key;
bit fleg;
void GPIO_init_row();
void GPIO_init_col();
unsigned int brojac = 0;
char matrixl[4][3]={        {1,2,3},
                            {4,5,6},
                            {7,8,9},
                            {10,0,11}} ;
```

Слика 12. Део кода за подешавање тастатуре [10]

Функција иницијализације редова, позива се на следећи начин:

- GPIO_init_row();

У оквиру ове функције портови редова и колона постављају се за улазе, док се портови колона постављају за излазе. Синтакса кода је приказана на слици 13. испод.

```
void GPIO_init_row()
{
 GPIOA_ODRbits.ODR15 = 0;
 GPIOA_ODRbits.ODR14 = 0;
 GPIOA_ODRbits.ODR11 = 0;
 GPIOA_ODRbits.ODR12 = 0;
 GPIO_Digital_Input(&GPIOA_ODR, _GPIO_PINMASK_15 | _GPIO_PINMASK_14);
 GPIO_Digital_Input(&GPIOA_ODR, _GPIO_PINMASK_11 | _GPIO_PINMASK_12);
 GPIO_Digital_Output(&GPIOD_ODR, _GPIO_PINMASK_2 | _GPIO_PINMASK_3);
 GPIO_Digital_Output(&GPIOB_ODR, _GPIO_PINMASK_3);
 GPIOD_ODRbits.ODR2 = 1;
 GPIOD_ODRbits.ODR3 = 1;
 GPIOB_ODRbits.ODR3 = 1;
}
```

Слика 13. Код за иницијализацију редова [10]

На исти начин се позива и функција иницијализације колона:

- GPIO_init_col();

У оквиру ове функције портови колона су постављени за улазе ове функције, док су портови редова њени излази.

```
void GPIO_init_col()
{
 GPIOD_ODRbits.ODR2 = 0;
 GPIOD_ODRbits.ODR3 = 0;
 GPIOB_ODRbits.ODR3 = 0;
 GPIO_Digital_Output(&GPIOA_ODR, _GPIO_PINMASK_15 | _GPIO_PINMASK_14);
 GPIO_Digital_Output(&GPIOA_ODR, _GPIO_PINMASK_11 | _GPIO_PINMASK_12);
 GPIO_Digital_Input(&GPIOD_ODR,  _GPIO_PINMASK_2 | _GPIO_PINMASK_3);
 GPIO_Digital_Input(&GPIOB_ODR,  _GPIO_PINMASK_3);
 GPIOA_ODRbits.ODR15 = 1;
 GPIOA_ODRbits.ODR14 = 1;
 GPIOA_ODRbits.ODR11 = 1;
 GPIOA_ODRbits.ODR12 = 1;
}
```

Слика 14. Код за иницијализацију редова [10]

Функција која препознаје који тастер је притиснут је фунцкија key_board(), коју смо дефинисали на следећи начин:

```
int key_board() {
GPIO_Alternate_Function_Enable(&_GPIO_MODULE_SWJ_JTAGDISABLE);
fleg = 1;
key = -1;
col1 = col2 = col3 = 0;
      col = 0;
      row = 0;
      GPIO_init_row();
      row1 = Button(&GPIOA_IDR, 15, 0, 1)==255;
      row2 = Button(&GPIOA_IDR, 14, 0, 1)==255;
      row3 = Button(&GPIOA_IDR, 12, 0, 1)==255;
      row4 = Button(&GPIOA_IDR, 11, 0, 1)==255;
      GPIO_init_col();
      col1 = Button(&GPIOD_IDR, 2, 0, 1)==255;
      col2 = Button(&GPIOB_IDR, 3, 0, 1)==255;
      col3 = Button(&GPIOD_IDR, 3, 0, 1)==255;
      if (col1)
          { col = 1; }
      else if (col2)
         { col = 2; }
      else if (col3)
         { col = 3; }
      if (row1)
         { row = 1; }
      else if (row2)
         { row = 2; }
      else if (row3)
         { row = 3; }
     if (row!=0 && col!=0)
      {key = matrix1[row-1][col-1]; }
      else {key = -1;}
    if (key != -1 && fleg == 1)
     { fleg = 0;}
     else if (key == -1) {fleg = 1;}
     delay_us(50);
     return key;
}
```

Слика 15. *Подешавање параметра [10]*

Функција за стопирање мотора позива се на следећи начин:

- Stop_Motor_x();

```
void Stop_Motor_x()
{
  // omoguci rad aktiviraj ENA
  ENA_1 = 0;   //zabrani motor
  // postavi smer
  DIR_1 = 0;   // 0 ili 1 za smer
  PLS_1 = 0;   // turn on


}
```

Слика 16. Функција за стопирање мотора X [10]

Функција за тестирање се позива на следећи начин:

- Test_motora();

```
void Test_motora(){          // moraju motori da budu slobodni, da je ukljucen servo
   while(br_impulsa < br_polu_periode)
      {
       // omoguci rad motora 1 aktiviraj ENA za sve motore da ne budu zakoceni
       ENA_1 = 1;    // mora da bude 1 za servo
       ENA_2 = 1;    // mora da bude 1 za servo
       ENA_3 = 1;    // mora da bude 1 za servo
       // postavi smer
       DIR_1 = 1;    // promeni smer

       PLS_1 = 1;              // turn on
       Delay_Cyc(delay);     // Delay_Cyc(5) je oko 2-2.5 mikro ceo ciklus je 5uS   minimum je 5us
       PLS_1 = 0;              // turn on
       Delay_Cyc(delay);
       br_impulsa = br_impulsa+1;
      }
```

Слика 17. Функција за тестирање мотора x на првој полупериоди [10]

У следећем услову се понавља исти поступак као и у претходном, само за другу полупериоду. Једина разлика је у смеру, који ће уместо јединице бити нула. Важно је нагласити да ово није цео садржај функције Test_motora(), већ да се ова два услова понављају за сваки мотор посебно.

У наредном делу биће објашњене функције секвенци, које активирају све моторе и покрећу рад мотора на одређени начин.

Прва функција јесте функција Sekvenca_1(). Ова функција активира ENA за све моторе, за прву полупериоду поставља мотор x у једном смеру (DIR_1 = 0), и моторе y и z у истом, другом смеру (DIR_2 = 1 и DIR_3 = 1), затим укључи и искључи све моторе. Исто ово понови и за другу полупериоду, уз промену смера рада мотора.

Ова функција се позива на следећи начин:

- Sekvenca_1();

```
void Sekvenca_1(){
  ENA_1 = 1;
  ENA_2 = 1;
  ENA_3 = 1;
  while(br_impulsa < br_polu_periode-1000)   //skrati da ne udari u senzor
  {
  DIR_1 = 0;     //motor x dole
  DIR_2 = 1;      //motor y gore
  DIR_3 = 1;   // motor z gore
  PLS_1 = 1;
  PLS_2 = 1;
  PLS_3 = 1;              // turn on
  Delay_Cyc(delay);//Delay_Cyc(5) je oko 2-2.5 mikro ceo ciklus je 5uS
  PLS_1 = 0;
  PLS_2 = 0;
  PLS_3 = 0;              // turn on
  Delay_Cyc(delay);
  br_impulsa++;
  }
  br_impulsa = 0;
   while(br_impulsa < 2*br_polu_periode-1000)
   {
    DIR_1 = 1;     //motor x gore
    DIR_2 = 0;      //motor y dole
    DIR_3 = 0;   // motor z dole
    PLS_1 = 1;
    PLS_2 = 1;
    PLS_3 = 1;              // turn on
    Delay_Cyc(delay);     // Delay_Cyc(5) je oko 2-2.5 mikro ceo ciklus je 5uS
    PLS_1 = 0;
    PLS_2 = 0;
    PLS_3 = 0;              // turn on
    Delay_Cyc(delay);
    br_impulsa++;
   }
   br_impulsa = 0;
  }
```

Слика 18. Функција Секвенца_1 [10]

Друга функција је Sekvenca_2(). Ова функција такође активира ENA за све моторе. За прву полупериоду поставља смерове мотора тако да мотор x стоји, а мотори y и z наизменично раде у истом смеру. Затим све моторе укључи и искључи и све понови за другу полупериоду уз промену смера свих мотора.

Описана функција се позива на следећи начин:

- Sekvenca_2();

```
void Sekvenca_2(){
  ENA_1 = 1;
  ENA_2 = 1;
  ENA_3 = 1;   // mora da bude 1 za servo, za obican step je 0
  while(br_impulsa < br_polu_periode-1000)
  {
  DIR_2 = 1;      //motor y gore
  DIR_3 = 0;   // motor z dole
  PLS_2 = 1;
  PLS_3 = 1;              // turn on
  Delay_Cyc(delay);//Delay_Cyc(5) je oko 2-2.5 mikro ceo ciklus je 5uS
  PLS_2 = 0;
  PLS_3 = 0;              // turn on
  Delay_Cyc(delay);
  br_impulsa++;
  }
  br_impulsa = 0;
   while(br_impulsa < 2*br_polu_periode-1000)
   {
      DIR_2 = 0;      //motor y dole
      DIR_3 = 1;   // motor z gore
      PLS_2 = 1;
      PLS_3 = 1;              // turn on
      Delay_Cyc(delay);//Delay_Cyc(5) je oko 2-2.5 mikro ceo ciklus je 5uS
      PLS_2 = 0;
      PLS_3 = 0;              // turn on
      Delay_Cyc(delay);
      br_impulsa++;
    } br_impulsa = 0;       }
```

Слика 19. Функција Секвенца_2 [10]

Следећа функција јесте Sekvenca_3(). У овој функцији се као и у претходним активира ENA за све моторе. За прву полупериоду постављају се исти смерови мотора. Затим се врши укључивање и искључивање мотора. Све ово се понови и за другу полупериоду, уз исте, промењене смерове за све моторе. Ова фунцкија се позива на следећи начин:

- Sekvenca_3();

```
void Sekvenca_3(){
  ENA_1 = 1;
  ENA_2 = 1;
  ENA_3 = 1;   // mora da bude 1 za servo, za obican step je 0
  while(br_impulsa < br_polu_periode-1000)
  {
   DIR_1 = 1;     //motor x gore
   DIR_2 = 1;      //motor y gore
   DIR_3 = 1;   // motor z gore
   PLS_1 = 1;
   PLS_2 = 1;
   PLS_3 = 1;              // turn on
   Delay_Cyc(delay);     // Delay_Cyc(5) je oko 2-2.5 mikro ceo ciklus je 5uS
   PLS_1 = 0;
   PLS_2 = 0;
   PLS_3 = 0;              // turn on
   Delay_Cyc(delay);
   br_impulsa++;
   }
   br_impulsa = 0;
   while(br_impulsa < 2*br_polu_periode-1000)
   {
     DIR_1 = 0;     //motor x dole
     DIR_2 = 0;      //motor y dole
     DIR_3 = 0;   // motor z dole
     PLS_1 = 1;
     PLS_2 = 1;
     PLS_3 = 1;              // turn on
     Delay_Cyc(delay);     // Delay_Cyc(5) je oko 2-2.5 mikro ceo ciklus je 5uS
     PLS_1 = 0;
     PLS_2 = 0;
     PLS_3 = 0;              // turn on
     Delay_Cyc(delay);
     br_impulsa++;
   }
     br_impulsa = 0;
```

Слика 20. Функција Секвенца_3 [10]

Последња функција везана за комбинацију рада мотора је функција Sekvenca_4(). Ова функција такође прво активира ENA за све моторе. Мотори се постављају у слободан режим рада и за једну и за другу полупериоду. Ова функција се позива на следећи начин:

- Sekvenca_4();

```
void Sekvenca_4(){
  ENA_1 = 1;
  ENA_2 = 1;
  ENA_3 = 1;   // mora da bude 1 za servo, za obican step je 0
  while(br_impulsa < br_cetvrt_periode)
  {
  DIR_1 = 0;     //motor x dole
  DIR_2 = 1;      //motor y gore
  DIR_3 = 0;   // motor z dole
  PLS_1 = 1;
  PLS_2 = 1;
  PLS_3 = 1;              // turn on
  Delay_Cyc(delay);     // Delay_Cyc(5) je oko 2-2.5 mikro ceo ciklus je 5uS
  PLS_1 = 0;
  PLS_2 = 0;
  PLS_3 = 0;              // turn on
  Delay_Cyc(delay);
  br_impulsa++;
  }
  br_impulsa = 0;
   while(br_impulsa < br_cetvrt_periode)
   {
    DIR_1 = 1;     //motor x gore
    DIR_2 = 0;      //motor y dole
    DIR_3 = 1;   // motor z gore
    PLS_1 = 1;
    PLS_2 = 1;
    PLS_3 = 1;              // turn on
    Delay_Cyc(delay);     // Delay_Cyc(5) je oko 2-2.5 mikro ceo ciklus je 5uS
```

Слика 21. Први део функције Секвенца_4 [10]

```
  PLS_1 = 0;
  PLS_2 = 0;
  PLS_3 = 0;              // turn on
  Delay_Cyc(delay);
  br_impulsa++;
}
br_impulsa = 0;
while(br_impulsa < br_cetvrt_periode)
{
  DIR_1 = 1;     //motor x dole
  DIR_2 = 0;      //motor y gore
  DIR_3 = 0;   // motor z gore

  PLS_1 = 1;
  PLS_2 = 1;
  PLS_3 = 1;              // turn on
  Delay_Cyc(delay);     // Delay_Cyc(5) je oko 2-2.5 mikro ceo ciklus je 5uS
  PLS_1 = 0;
  PLS_2 = 0;
  PLS_3 = 0;              // turn on
  Delay_Cyc(delay);
  br_impulsa++;
}
br_impulsa = 0;
while(br_impulsa < br_cetvrt_periode)
{
// postavi smer
  DIR_1 = 0;     //motor x gore
  DIR_2 = 1;      //motor y dole
```

Слика 22. Други део функције Секвенца_4 [10]

```
DIR_3 = 1;   // motor z dole

PLS_1 = 1;
PLS_2 = 1;
PLS_3 = 1;              // turn on
Delay_Cyc(delay);     // Delay_Cyc(5) je oko 2-2.5 mikro ceo ciklus je 5uS
PLS_1 = 0;
PLS_2 = 0;
PLS_3 = 0;              // turn on
Delay_Cyc(delay);
br_impulsa++;
}
br_impulsa = 0;
```

Слика 23. Трећи део функције Секвенца_4 [10]

На слици 24. је приказана најбитнија линија кода у којој се од двоструке вредности броја полупериоде одузима вредност 1000, како би се избегло ударање система у сензор.

```
while(br_impulsa < 2*br_polu_periode-1000)
```

Слика 24. Битна линија кода [10]

Функција уз помоћ које се на основу сензора, мотори доводе у почетни положај је функција Sekvenca_pocetni(). У овој функцији се постављају услови за сваки сензор посебно и на основу резултата са сензора поставља мотор у почетни положај у случају да већ није. Важно је нагласити да је на почетку ове функције, као и на почетку свих функција потребно активирати ENA за све моторе.

Ова функција се позива на следећи начин:

- Sekvenca_pocetni();

На слици 25, испод, приказана је синтакса кода за само један сензор. За све остале сензоре је код потпуно идентичан, уз промену мотора за који се проверава услов и који се поставља у почетни положај по потреби.

```
void Sekvenca_pocetni(){
 //motori se spustaju dole, svaki motor zaustavlja posebni senzor

   while(SENZ_X==1)
      {
       // omoguci rad motora 1 aktiviraj ENA za sve motore da ne budu zakoceni
       ENA_1 = 1;   // mora da bude 1 za servo
       ENA_2 = 1;   // mora da bude 1 za servo
       ENA_3 = 1;   // mora da bude 1 za servo
       // postavi smer
       DIR_1 = 0;   // motor x dole

       PLS_1 = 1;              // turn on
       Delay_Cyc(delay);  // Delay_Cyc(5) je oko 2-2.5 mikro ceo ciklus je 5uS
       PLS_1 = 0;              // turn on
       Delay_Cyc(delay);
      }
}
```

Слика 25. Функција Секвенца_почетни [10]

Последња функција везана за овај главни део програма јесте функција Sekvenca_polovina(). Ова функција поставља моторе на половину механичког опсега. Прво поставља свим моторима исти смер, затим их укључује док не дођу до половине и на крају их искључује. Ова функција се позива на следећи начин:

- Sekvenca_polovina();

На слици 26. , приказана је цела синтакса ове функције.

```
void Sekvenca_polovina(){
  br_impulsa = 0;
   while(br_impulsa < br_polu_periode)
   {
     DIR_1 = 1;     //motor x gore
     DIR_2 = 1;      //motor y gore
     DIR_3 = 1;   // motor z gore

     PLS_1 = 1;
     PLS_2 = 1;
     PLS_3 = 1;              // turn on
     Delay_Cyc(delay);     // Delay_Cyc(5) je oko 2-2.5 mikro ceo ciklus je 5us
     PLS_1 = 0;
     PLS_2 = 0;
     PLS_3 = 0;              // turn on
     Delay_Cyc(delay);
     br_impulsa++;
   }
   br_impulsa = 0;
}
```

Слика 26. Функција Секвенца_половина [10]

У оквиру главне main функције остало је да позовемо све до сада дефинисане функције односно секвенце за рад мотора. Поред тога, потребно је дефинисати и функцију за timer. На слици 27. , испод, приказан је део главне main функције у којем се позивају функције мотора у зависности од тастера који је притиснут као и почетно дефинисање промењивих.

```
#include <Settings.h>
#include <motors.h>
#include <keyboard.h>
sbit START_MOTOR at GPIOD_IDR.B8;
unsigned long int delay = 50;
#define TURN_ON 1
#define TURN_OFF 0
// pokretanje masine na dugme RD_8
bit motor_flag;
int flag = 0;   //pocetno stanje motora ne radi, tek na taster E_0 se pokrece
bit key_flag;
int key_value = -1;
int key_temp = -1;
void Settings_MCU();
void Sekvenca_pocetni();
void Sekvenca_polovina();
void Test_motora();
void Sekvenca_1();
void Sekvenca_2();
void Sekvenca_3();
void Sekvenca_4();
void Stop_motor_x();
void Stop_motor_y();
void Stop_motor_z();
int key_board();
void InitTimer4();
```

Слика 27. Почетни део главне main функције [10]

На слици 28. , приказана је главна main функција.

```
void main()
{
 GPIO_Digital_Output(&GPIOE_BASE, _GPIO_PINMASK_8);
 GPIO_Digital_Output(&GPIOE_BASE, _GPIO_PINMASK_9);
 GPIO_Digital_Output(&GPIOE_BASE, _GPIO_PINMASK_10);
  Settings_MCU();
  motor_flag == 1;

  key_flag = 0;
  InitTimer4();

  while (1)
  {
    if (START_MOTOR == 1 && flag == 1)
    {
      flag = 0;
      motor_flag = ~motor_flag;
    }
    else if (START_MOTOR == 0 && flag == 0)
    {
      flag = 1;
    }
    if (motor_flag == 1)
    {
      NVIC_IntEnable(IVT_INT_TIM4);
      if (key_value!=-1 && key_value!=5)
      {
        while (motor_flag)
        {
          if (key_value == 7)
          { delay=50; } else
          if (key_value == 8)
          { delay=60; } else
          if (key_value == 9)
          { delay=80; }
          if (key_flag)
          {
            if (key_value<6) {
            Sekvenca_pocetni();
            Sekvenca_polovina();  }
            key_flag = 0;
          }
         if (key_value == 1)  {
            Sekvenca_1();
          } else if (key_value == 2) {
            Sekvenca_2();
          } else if (key_value == 3) {
            Sekvenca_3();
          } else if (key_value == 4) {
            Sekvenca_4();
          } else if (key_value == 5) {
            delay=50;
            break;
          } }} }}
```

Слика 28. Главна main функција [10]

Две нове функције, које су дефинисане у оквиру маин функције су функција *InitTimer4()* и функција *Timer4_interrupt()ivIVT_INT_TIM4.*. Њихов код је приказан на сликама 29. и 30. [11]

```
void InitTimer4(){
  RCC_APB1ENR.TIM4EN = 1;
  TIM4_CR1.CEN = 0;
  TIM4_PSC = 119;
  TIM4_ARR = 59999;
  //period na koji proveravamo tastaturu je 100 ms
  NVIC_IntDisable(IVT_INT_TIM4);
  //NVIC_IntEnable(IVT_INT_TIM4);
  TIM4_DIER.UIE = 1;
  TIM4_CR1.CEN = 1;
}
```

Слика 29. Код функције InitTimer4 [10]

Линија кода NVIC_intDisable(IVT_INT_TIM4) се односи на забрану прекида – interrupt – а.

```
void Timer4_interrupt() iv IVT_INT_TIM4 {
  TIM4_SR.UIF = 0;
  //Enter your code here
  key_temp = key_board();
  if  (key_flag==0)
  //fleg postaje 1 ako je prethodnobio 0 i ako
  //se pritisne neki drugi taster u odnosu na onaj koji je bio u prethodnom
  //trenutku
  (key_flag = ((key_temp!=-1) && key_value != key_temp);}
  if (key_temp!=-1) {key_value = key_temp;}
  //NVIC_IntDisable(IVT_INT_TIM4);


}
```

Слика 30. Код функције Timer4_interrupt [10]

4.2. Мерење динамике система

Након развијеног система за симулацију кретања борбеног возила и стабилизацију кретања, потребно је реализовати аквизицију и мерење динамике самог система. Прилагодљива решења за аквизицију података и управљање су моћно оружје када је реч о мерењима у научним и истраживачким лабораторијама, имајући у виду да се у њима врло често користи делимично модификована или опрема сопствене конструкције. Прикупљени подаци са сензора се, након кондиционирања у мерном интерфејсу сопствене израде, уводе преко аквизиционе картице у рачунар. Мерни софтвер је развијен коришћењем LabView моћног, графичког, програмерског окружења. Због својих карактеристика, рачунари се све више употребљавају у лабораторијама за мерења, прикупљање, обраду података и управљање. Уместо класичне мерне опреме, све је чешће у употреби рачунар, који се опремљен неком аквизиционом картицом и одговарајућим софтвером претвара у систем погодан за мерења, прикупљање и обраду података и управљање различитим експериментима. Снажну подршку развоју софтвера за управљање мерењима пружа LabView (Labaratory Virtual Instrument Engineering Workbench), моћно, графичко, програмерско окружење, које представља стандард на пољу виртуалне инструментације. [13]

Слика 31. LabView [10]

Универзални систем за мерења састоји се из хардверског модула и софтвера развијеног на бази LabView пакета са више мерних инструмената и метода. Хардверски модул који може да генерише струјне и напонске сигнале и бележи одзив испитиваног сигнала чине: рачунар, вишенаменска мерно-управљачка картица и спољашњи мерно – управљачки интерфејс сопственог дизајна и израде, као и одговарајући сензори. [14]

LabView програми се називају виртуелни инструменти или VI, зато што њихова појава и операције имитирају физичке инструменте попут осцилоскопа и мултиметра. LabView садржи сет алата за прикупљање, анализу, приказивање и смештање података, као и алате за отклањање проблема у коду.

За управљање аквизиционом картицом неопходан је DAQ софтвер, који контролише систем аквизицијом изворних података, анализом и презентовањем резултата.

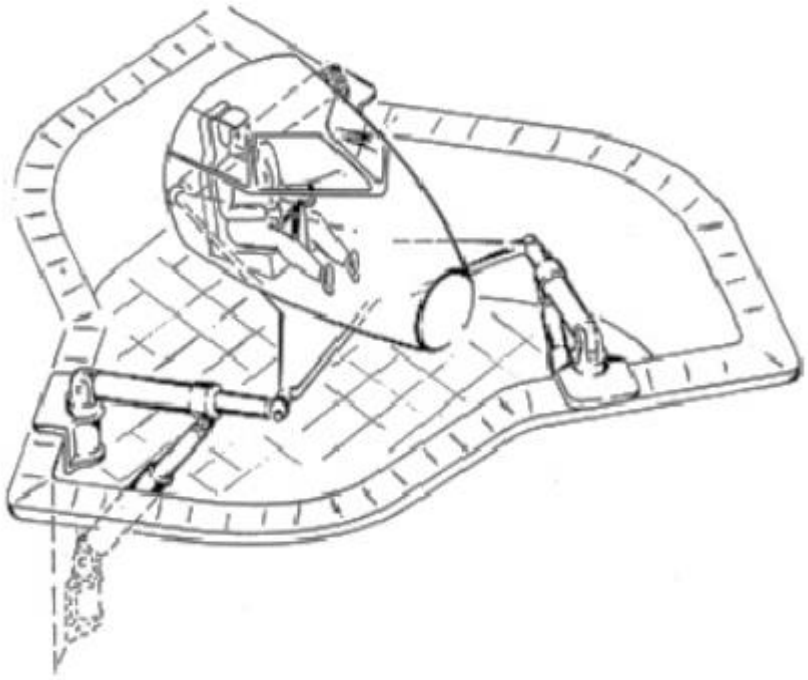
DAQ софтвер најчешће укључује и драјвере и апликативни софтвер. Драјвери су од значаја за уређај или тип уређаја и обухватају скуп команди које уређај прихвата. Апликативни софтвер приказује и анализира прикупљене податке.

Фирма National Instruments поред софтверског пакета LabView нуди и велики број уређаја за мерење, којима се управља коришћењем NI-DAQ колекције софтвер драјвера. NI-DAQ драјвери се користе за конфигурацију, прикупљање и слање података уређајима за мерење. [14]

Након чувања прикупљених података уз помоћ аквизиционе функције, оно што нам остаје јесте да те податке прочитамо и да добијемо информације које су нам од великог значаја за даљи развој система. Ово је оно докле смо ми, у развоју и реализацији система троосне стабилизације стигли. Уз помоћ програмског окружења Matlab, извршићемо читање измерених података о динамици система.

Оно што можемо препознати као један недостатак развијеног система, јесте то што је он развијен само у три осе. У будућности, настојимо да развијемо систем са шест оса, како би стабилизација била прецизнија и тачнија, тачније настојимо развоју Стјуартове (Stewart) платформе. [12]

Стјуартова платформа или хексапод представља манипулатор са паралелном кинематиком. Ову платформу је 1965. године дизајнирао инжењер D. Stewart користећи паралелну структуру која је грађена од шест идентичних хидрауличних управљачких кракова који су универзалним зглобовима везани за постоље за симулатор лета, а касније даљим развојем нашла је вишеструку примену у различитим областима. Кракови чине паралелне кинематске ланце који повезују две платформе, па због тога ова конструкција припрада породици паралелних манипулатора. [12]



Слика 32. Прва шема Стјуартове платформе [12]

Основне карактеристике Стјуартове платформе су:

1. Шест степени слободе кретања – могућност координираног управљања сваким краком омогућује хексаподу велики број положаја и комплетну контролу сваког степена слободе кретања.
2. Висока тачност – због великог броја положаја постиже се висока тачност позиционирања алата и сила се равномерно распоређује на све кракове.
3. Висока крутост – због компактне грађе, платформу одликује висока крутост у било којем положају.
4. Различите величине – хексапод може бити мале или велике величине, зависно од намене.
5. Софтверска контрола – за управљање се користе посебни софтверски алгоритми и методе.
6. Радни простор хексапода је релативно мали и ограничен дужинама које кракови могу поприми те угловима који се могу постићи код зглобова на спојевима кракова и платформи.

Због добрих карактеристика, научници и стручњаци се у последње две деценије интересују много за Стјуартову платформу, а све бољи развој технологије помаже у прецизнијем управљању и одржавању. [12]

5. Закључак

Можемо закључити следеће:

- Сервомеханизам функционише на тај начин што се излазна величина стално мери и посредством повратне спреге упоређује са улазном величином, дејствујући при том тако да грешку која настаје као резултат упоређења излазне и улазне величине своди на нулу, уз појачање снаге.
- Постоје три основне врсте серво мотора: ротација положаја, континуална ротација и линеарна.
- За мерење угаоне брзине поремећаја користе се два брзинска жироскопа као сензори. Блок-шема оба ова система је иста, само се разликују параметри система: оптерећење, погонски уређај, редуктор, гранична учестаност, итд.
- За развој система троосне стабилизације, неопходно је прво направити систем који ће симулирати одређено кретање борбеног возила и чије померање треба бити стабилисано. Обзиром да се у овом раду бавимо искључиво развојем система са гледишта електронике, све остале потребне делове искористили смо као готове производе машинског и механичког сектора.
- Након развијеног система за симулацију кретања борбеног возила и стабилизацију кретања, потребно је реализовати аквизицију и мерење динамике самог система. Универзални систем за мерења састоји се из хардверског модула и софтвера развијеног на бази LabView пакета са више мерних инструмената и метода.
- Оно што можемо закључити као један недостатак развијеног система, јесте то што је он развијен само у три осе. У будућности, настојаћемо да развијемо систем са шест оса, како би стабилизација била прецизнија и тачнија, тачније тежићемо развоју Стјуартове (Stewart) платформе.

5. Литература

[1] Миливоје Р. Секулић, Основи теорије аутоматског управљања – сервомеханизм, Београд, 1976. године.

[2] Увод у серво моторе, Приступљено: 03.08.2019. године, линк: http://ba.china-steppermotor.com/news/baigela-7732819.html

[3] Приказ типчног серво серво уређаја, Приступљено: 03.08.2019. године, линк: http://ba.china-steppermotor.com/Content/upload/2017138833/20170822141044568537 7.png

[4] Приказ серво уређаја високе снаге, Приступљено: 03.08.2019. године, линк: http://ba.china-steppermotor.com/Content/upload/2017138833/20170822141058430314 9.png -

[5] Приказ спојница и ДЦ мотора у серво уређају високог напона, Приступљено: 03.08.2019. године, линк: http://ba.china-steppermotor.com/Content/upload/2017138833/20170822141112693501 3.png

[6] Слободан О., Павле К., Стабилизација кретања возила и система стабилизације на борбеном возилу, Виша електротехничка школа, Београд, 2005. године.

[7] Xinje DS2-AS series servo drive User manual, Приступљено: 21.08.2019. године, линк: http://www.sah.rs/Frekventni_I_Servo/Manual/DS2AS-manual.pdf

[8] SAH electronics, серво мотори, Приступљено: 21.08.2019. године, линк: http://www.sah.co.rs/upravljanje-motorima/servo-sistemi/servo-motori.html

[9] SAH electronics, сензор за метал, Приступљено: 21.08.2019. године, линк: http://www.sah.co.rs/senzori/senzori-pritiska-nivoa.html

[10] Невена Сташић, Техничка документација пројекта троосна стабилизација, DLS Специјални системи, 2018.

[11] Reference manual STM32F advanced Arm - based 32bit MCUs, Приступљено: 21.08.2019. године, линк: https://www.st.com/resource/en/datasheet/cd00220364.pdf

[12] Mathworks, Creating a Stewart Platform Model Using SimMechanics, Приступљено: 21.08.2019. године, линк:

https://www.mathworks.com/company/newsletters/articles/creating-a-stewart-platform-model-using-simmechanics.html

[13] Аленка М., Мирослав Б. и Бранко К., Виртуелна инструментација, Универзитет у Крагујевцу, Технички факултет у Чачку, 2010

[14] National Instruments, LabVIEW Basic I, Introduction Course Manual, Приступљено: 21.08.2019. године, линк: https://www.ni.com