Institut za matematiku i informatiku

Prirodno – matematički fakultet

Univerzitet u Kragujevcu

Predmet:
Mikroprocesorski sistemi

Tema projektnog zadatka:
Troosni sistem za održavanje ravnoteže

Profesor:
Dr Aleksandar Peulić

Student:
Tamara Jerinić

Sadržaj

Definisanje neophodne opreme i komponenti

Naziv	Količina
STM32F103C6T6	1
Servo motor	3
Potenciometar	3
Napajanje 5V	1

Kako bi se projekat realizovao, neophodno je razvojno okruženje *STMCubeIDE* zajedno sa *STMCubeMX.*

Za simulaciju rada hardverskih komponenti, neophodna je instalacija softvera za dizajniranje i simulaciju elektronskih kola *Proteus 8 Professional*.

Namena projekta

Tokom razvoja tehnologije, razvila se i potreba za postojanjem uređaja koji su sposobni da održavaju balans prilikom promene položaja i kretanja. Bilo da je u pitanju stativ koji omogućava kameri da uprkos kretanju i pomeranju položaja, uvek ostane stabilna i bez nepoželjnih pokreta, ili sistem za održavanje ravnoteže letelica i dronova, kao i robota koji se kreću, mogućnosti primene su brojne.

Upravo takvi uređaji dele jednu zajedničku funkcionalnost, a to je sposobnost održavanja ravnoteže upotrebom senzora koji prate kretanje objekta i na osnovu čijih se očitavanja vrši balansiranje pomoću motora koji koriguju položaj objekta, u skladu sa očitanim vrednostima.

Samim tim, namena projekta biće razvoj troosnog sistema za održavanje ravnoteže koji je upotrebljiv samostalno ili kao ugradni sistem u drugim tehničkim projektima različitih namena.

Sastav projekta

Hardversko rešenje

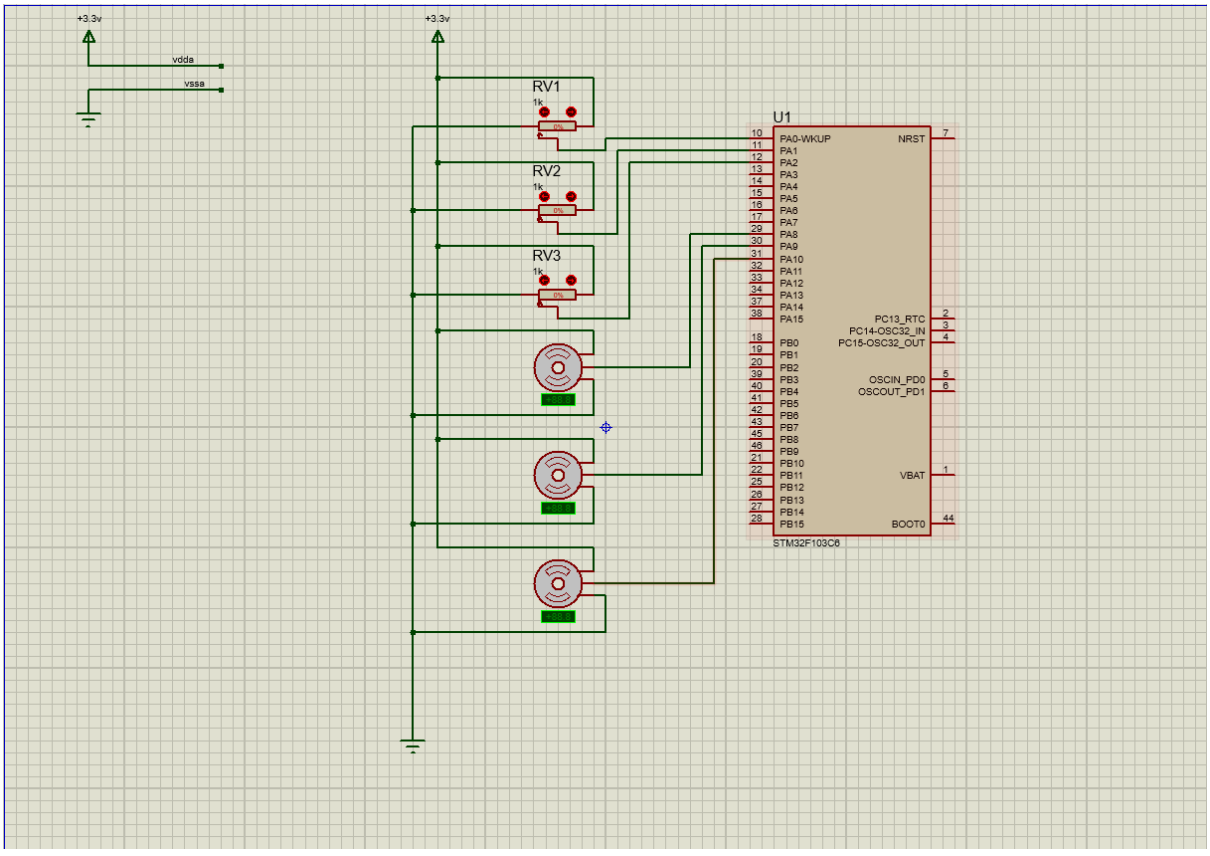
Projekat predstavlja izradu troosnog sistema za održavanje ravnoteže. Kako bi se ravnoteža mogla uspešno održavati u trodimenzionalnom prostoru, neophodno je praćenje položaja objekta kroz tri koordinatne ose: X, Y i Z.

Usled poreba simulacije i ograničenja u pogledu komponenti, zamišljena promena položaja sistema se simulira korišćenjem tri potenciometra. Svaki potenciometar predstavlja po jedan pravac u kom se vrši promena položaja sistema.

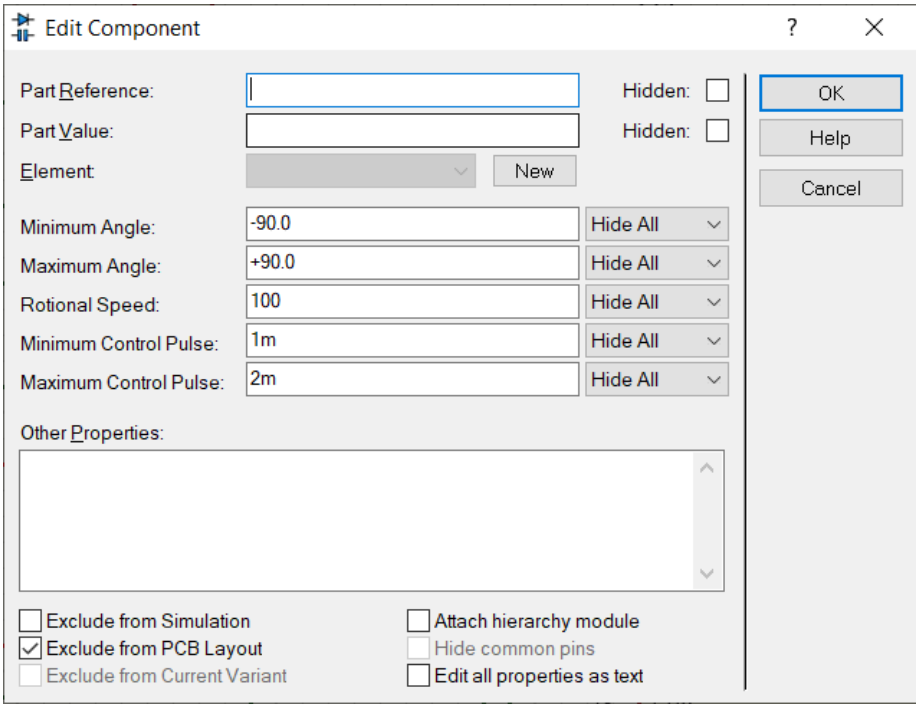
Uspostavljanje ravnoteže se vrši upotrebom servo motora. U projektu su upotrebljena tri servo motora, u skladu sa trodimenzionalnim prostorom. Uloga servo motora jeste da nakon prijema signala o promeni položaja sistema, utiču na održavanje sistema u ravnoteži i njegovu stabilizaciju - paralelno sa X, Y i Z osom.

Implementacija je izvršena na način da se prilikom vršenja promene položaja potenciometara, signal obrađuje i skalira i potom se prosleđuje adekvatnom servo motoru. Kako bi održavanje ravnoteže bilo moguće, odgovarajući servo motor vrši promenu položaja u suprotnom smeru od smera koji je zadat potenciometrom.

Prikaz implementacije simulacije hardverskog rešenja:



U izradi simulacije korišćeni su servo motori, koji su konfigurisani na sledeći način:



Svaki servo motor se može napraviti ugao od 180 stepeni, i to 90 stepeni u jednom smeru i 90 stepeni u drugom smeru.

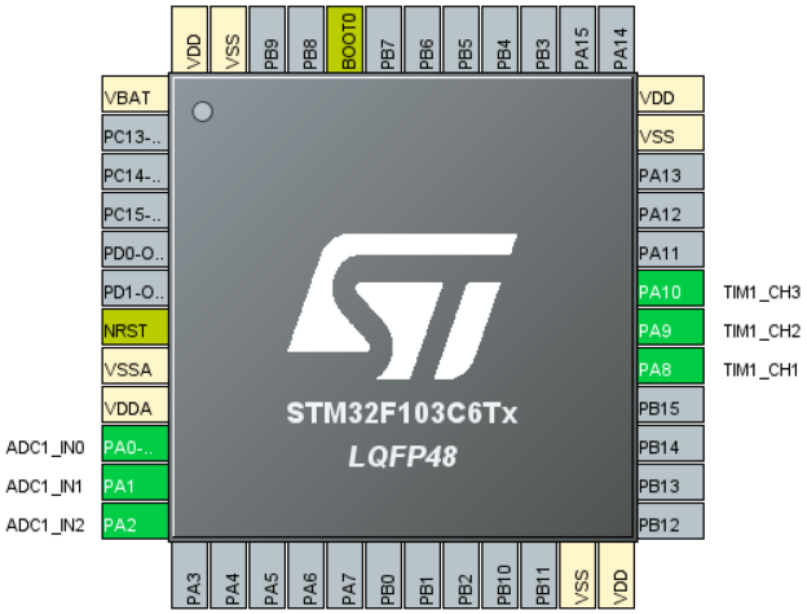
Softversko rešenje

Za uspešno funkcionisanje sistema potrebno je izraditi i odgovarajući softver koji je usklađen sa radom hardvera. Izrada softverskog rešenja vršena je u razvojnom okruženju *STM32CubeIDE i STM32CubeMX.* Rešenje je sastavljeno upotrebom mikrokontorlera *STM32F103C6T6* čiji pinovi PA0,PA1 i PA2 se koriste za prijem analognih signala potenciometara koji su na njih povezani.

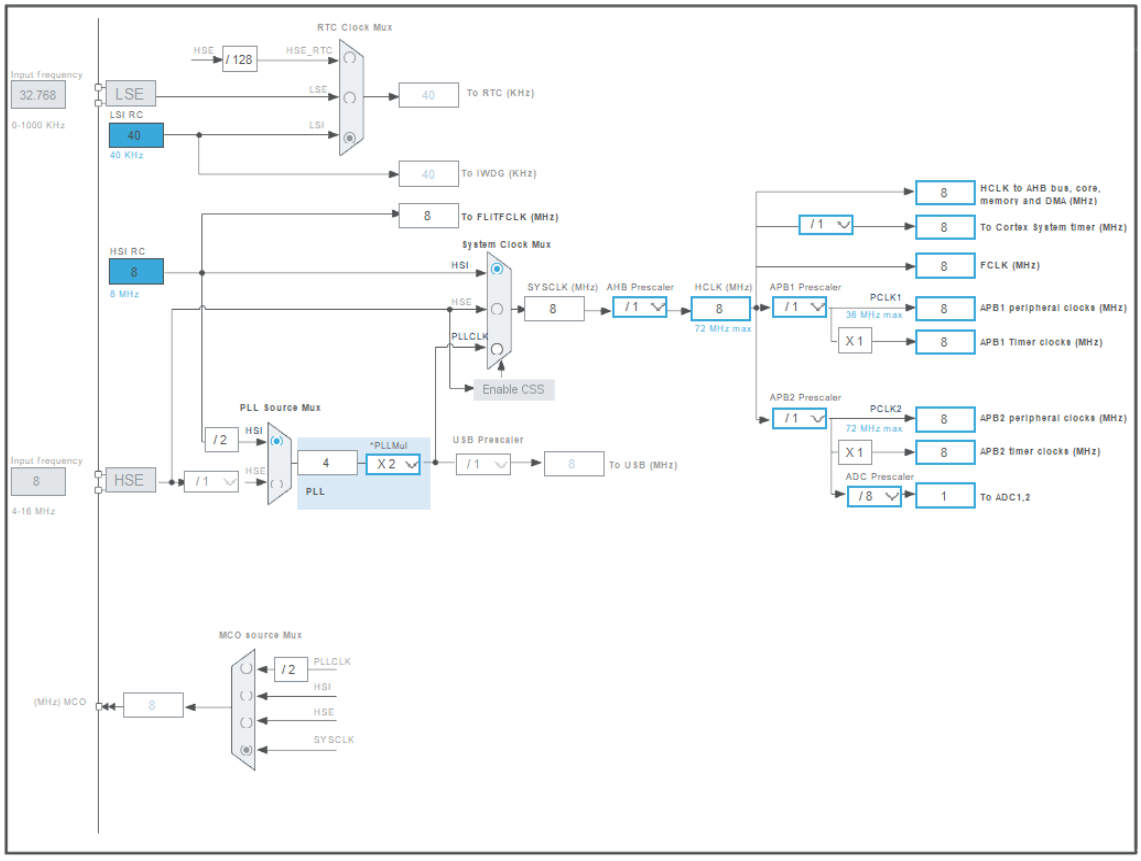
Pinovi PA8, PA9 i PA10 se koriste za slanje podataka servo motorima, u zavisnosti od primljenih i obrađenih signala sa potenciometara.

Naziv pina	Namena
Pin PA0	Konfigurisan je tako da očitava signal ADC1 sa kanala ADC_CHANNEL_0
Pin PA1	konfigurisan je tako da očitava signal ADC1 sa kanala ADC_CHANNEL_1
Pin PA2	Konfigurisan je tako da očitava signal ADC1 sa kanala ADC_CHANNEL_2
Pin PA8	Konfigurisan je tako da prenosi signal, koji je dobijen upotrebom AD konverzije nad očitanim vrednostima sa pina PA0, ka servo motoru povezanom na pin PA8.
Pin PA9	Konfigurisan je tako da prenosi signal, koji je dobijen upotrebom AD konverzije nad očitanim vrednostima sa pina PA1, ka servo motoru povezanom na pin PA9.
Pin PA10	Konfigurisan je tako da prenosi signal, koji je dobijen upotrebom AD konverzije nad očitanim vrednostima sa pina PA2, ka servo motoru povezanom na pin PA10.

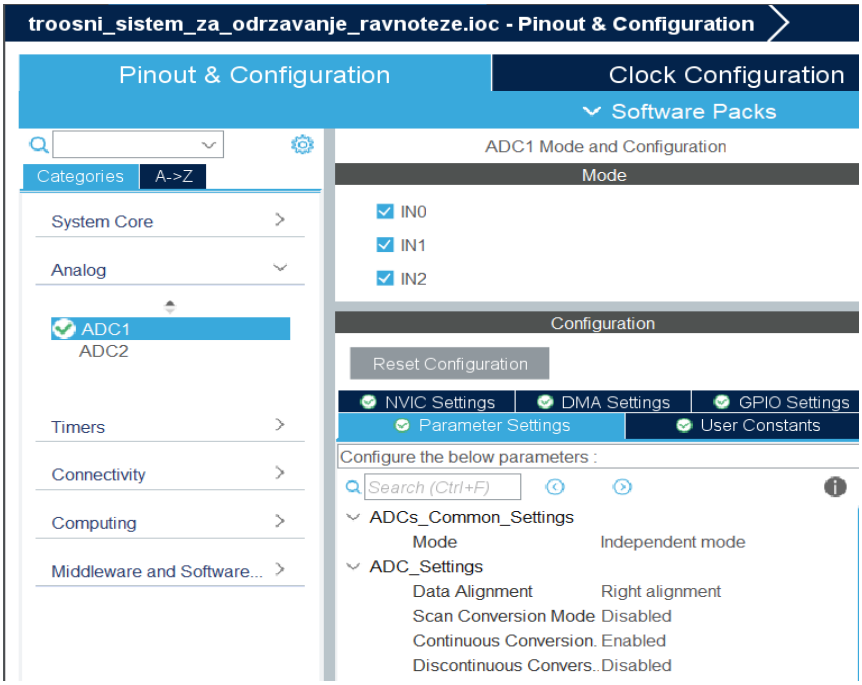
Konfiguracija pinova u okruženju *STM32CubeMX*



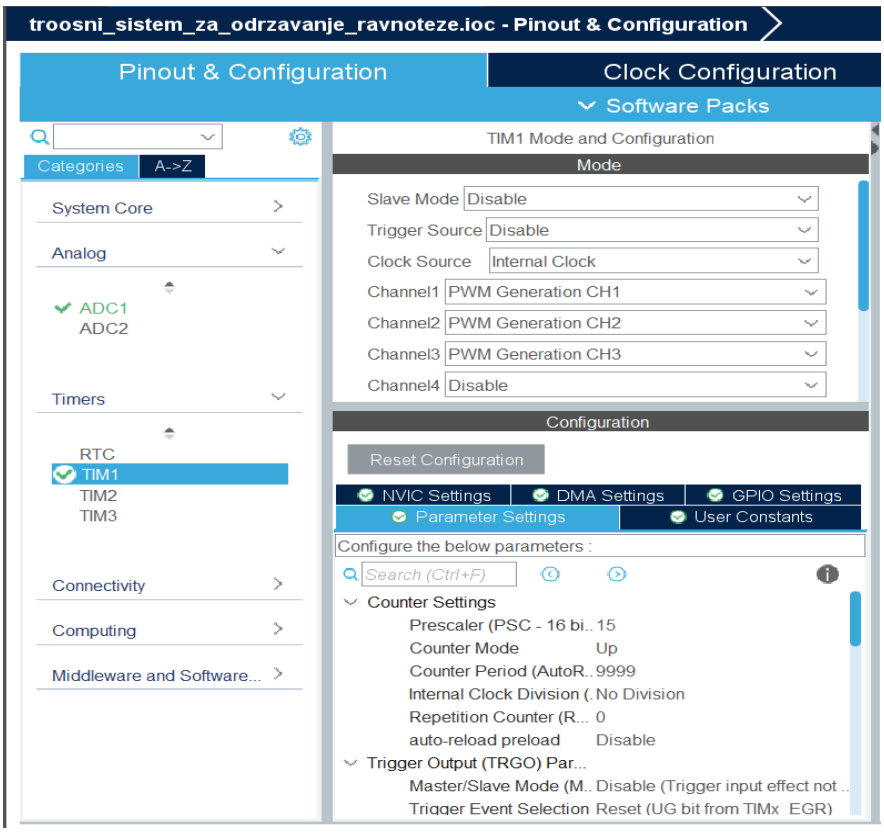
Prikaz konfiguracije sistemskog clock-a



Prikaz konfiguracije Analogno-Digitalne konverzije



Prikaz konfiguracije Tajmera



Opis koda

Kod je izrađen u skladu sa konfiguracijom pinova. Uzevši u obzir da su vrednosti koje generiše potenciometar analogne, pinovi koji se koriste za prenos podataka između potenciometara i ploče su konfigurisani za analogno – digitalnu konverziju.

S obzirom da je potrebno pratiti promenu u troosnom sistemu, svaki pin je zaslužan za prijem podataka sa jednog potenciometra koji predstavlja promene na jednoj osi.

Parametri hardverskih komponenti I samog strujnog kola utiču na vrednosti koje pružaju potenciometri I analogno-digitalna konverzija.

Za strujno kolo koje se napaja strujom napona 5V I potenciometar koji ima otpornost 10 kΩ, digitalna vrednost nakon analogno-digitalne konverzije će biti u opsegu od 0 do 4095.

Ova vrednost se ne može poslati servo motoru, već je potrebno da se pre prenosa obradi kako bi bila u opsegu od 8999 do 9450, što su vrednosti potrebne za rad konfigurisanog servo motora.

Kako bi se vrednosti dobijene AD konverzijom preslikale u odgovarajuće vrednosti neophodne za funkcionisanje servo motora, primeniće se sledeća formula.

$$y=8999.0+(501.0/4095.0*x)$$

Pri čemu je y vrednost koja se prosleđuje servo motoru, dok je x vrednost koja se dobija analogno-digitalnom konverzijom.

Opis *while* petlje

While petlja u main.c fajlu rešenja služi za očitavanje podataka AD konverzije i njihovo slanje ka servo motorima.

U svakoj iteraciji se ocitavaju vrednosti AD konverzije. Prvo se vrši očitavanje potenciometra koji predstavlja promenu na x osi. Nakon toga se vrednost dobijena konverzijom obrađuje primenom formule i prosleđuje adekvatnom servo motoru.

Isti proces se nakon obrade podataka za x osu ponavlja za y osu, a zatim i za z osu. Nakon obrade sva tri signala, iteracija se ponavlja u beskonačnoj petlji.

Prikaz koda *while* petlje

```
while (1)
{

    sConfigPrivate.Channel = ADC_CHANNEL_0;
    HAL_ADC_ConfigChannel(&hadc1, &sConfigPrivate);
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1,1000);
    potenciometarX = HAL_ADC_GetValue(&hadc1);

    rezultat1=8999.0+(501.0/4095.0*potenciometarX);
    x=(int)rezultat1;
    __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,x);
    HAL_ADC_Stop(&hadc1);

    //y
    sConfigPrivate.Channel = ADC_CHANNEL_1;
    HAL_ADC_ConfigChannel(&hadc1, &sConfigPrivate);
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1,1000);
    potenciometarY = HAL_ADC_GetValue(&hadc1);

    rezultat2=8999.0+(501.0/4095.0*potenciometarY);
    y=(int)rezultat2;
    __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_2,y);
    HAL_ADC_Stop(&hadc1);

    //z
    sConfigPrivate.Channel = ADC_CHANNEL_2;
    HAL_ADC_ConfigChannel(&hadc1, &sConfigPrivate);
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1,1000);
    potenciometarZ = HAL_ADC_GetValue(&hadc1);

    rezultat3=8999.0+(501.0/4095.0*potenciometarZ);
    z=(int)rezultat3;
    __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_3,z);
    HAL_ADC_Stop(&hadc1);
}
```