ИНСТИТУТ ЗА МАТЕМАТИКУ И ИНФОРМАТИКУ
ПРИРОДНО-МАТЕМАТИЧКИ ФАКУЛТЕТ
УНИВЕРЗИТЕТА У КРАГУЈЕВЦУ

Пројекат

Предмет: Микропроцесорски системи
Тема: Систем за позиционирање, угао

Студент:
Оливера Обретковић, 70/2020

Професор:
др Александар Пеулић

Крагујевац, 2024. год.

Садржај

УВОД

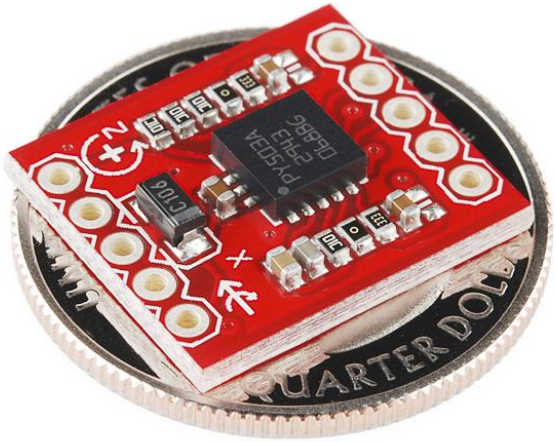
У савременом свету, где је прецизност кључна у многим техничким областима, развој система за позиционирање и детекцију угла има значајан утицај на различите индустрије. Тачно одређивање угла је од суштинског значаја у апликацијама као што су роботика, аутоматизација и контролни системи. Ови системи омогућавају бољу контролу и управљање, побољшавајући ефикасност и сигурност у процесима који захтевају високу прецизност. Примена система за детекцију угла такође доприноси развоју иновативних решења у области електронике, где је тачност критична за постизање оптималних перформанси уређаја и система. Унапређењем технологија за позиционирање и мерење угла, отварају се нове могућности за развој иновативних решења која доприносе побољшању ефикасности и функционалности у разним индустријама и научним истраживањима.

СЕНЗОР ЗА МЕРЕЊЕ УГЛА - ЖИРОСКОП

За реализацију система за позиционирање и мерење угла неопходно је користити прецизан сензор. Најпогоднији сензор за ову намену је жироскоп. Жироскоп је уређај који мери промену угла тела у односу на његову осу, омогућавајући прецизну контролу оријентације у простору.

Жироскоп функционише на принципу очувања угаоног момента, што значи да одржава свој правац ротације чак и када је под утицајем спољашњих сила. Уређај се састоји од ротирајућег диска или точка чија оса може слободно да се окреће у различитим правцима. Због великог угаоног момента који се генерише током ротације, свака промена положаја осе жироскопа је минимална у поређењу са оним што би се десило без овог принципа.

Овај механизам чини жироскоп изузетно корисним за примене где је неопходно прецизно мерење и одржавање угла, као што су стабилизација камера, дронова, или навигационих система. У нашем систему, жироскоп би омогућио тачне и поуздане податке о углу, који би се затим користили за даље процесирање и приказивање кориснику.



Слика 2- Жироскоп

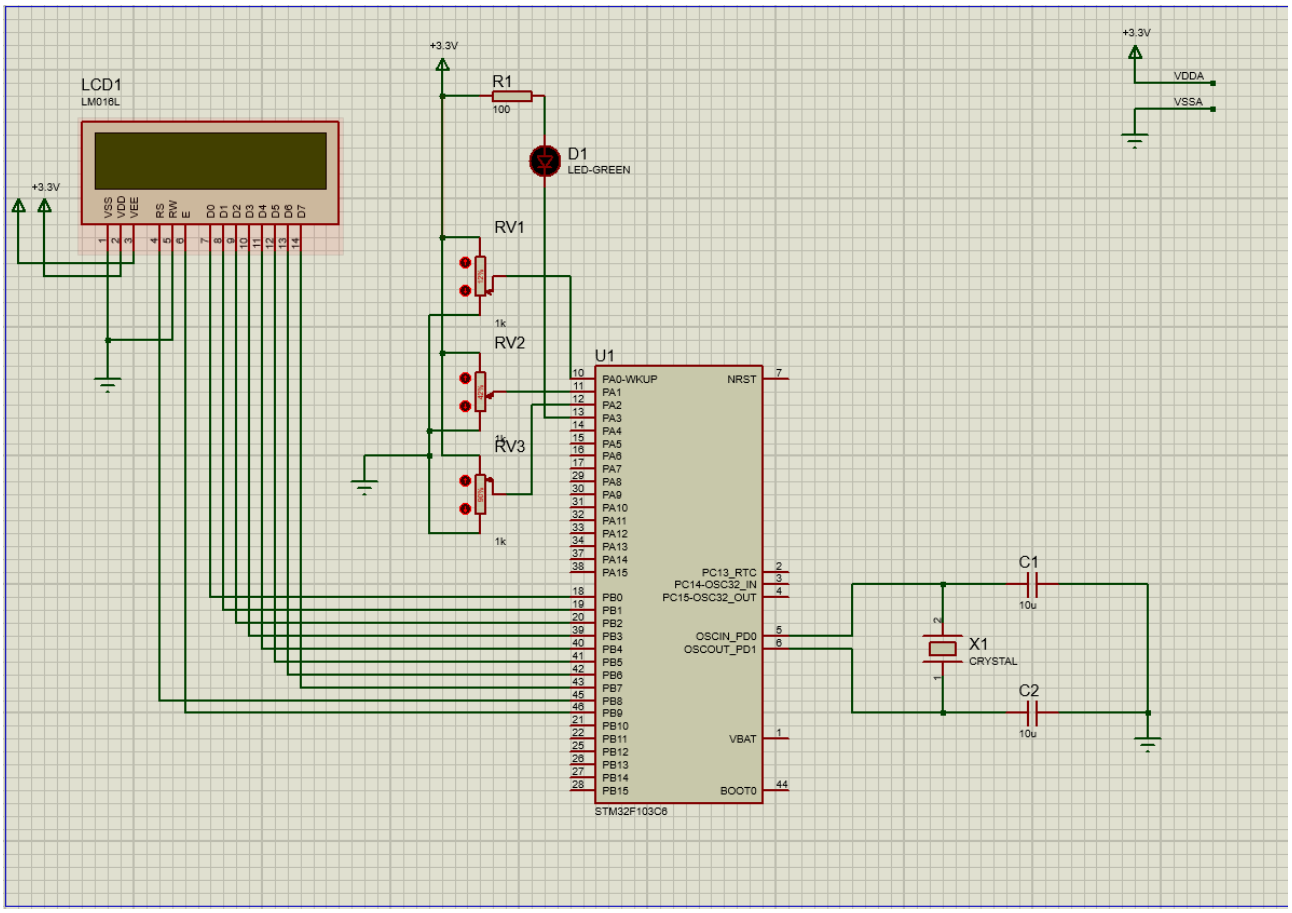
Слика 1- Изглед жироскопа

ШЕМА ПРОЈЕКТА – PROTEUS

У Proteus програму, који смо користили за симулацију система, жироскоп као сензор није доступан. Из тог разлога, коришћен је потенциометар као замена за жироскоп. Иако потенциометар не може директно мерити угаону брзину или промену оријентације као жироскоп, његова способност да обезбеди варијације у напону у зависности од положаја чини га прикладном заменом у симулационом процесу.

Вредности очитане са потенциометра биће приказане на LCD дисплеју. Такође је постављена и LED диода која визуелно приказује када је мерење у току (угашена), и када је могуће мењати вредности на потенциометрима (упаљена).

Жироскоп мери брзину ротације за сваку осу, X, Y i Z, тако да је неопходно користити 3 потенциометра ради симулације. Вредност која треба бити приказана на дисплеју је изражена у °/s, а опсег вредности које се мере може бити различит, у односу на модел. Типични опсези које сензор може мерити су ±250 °/s, ±500 °/s, ±1000 °/s, ±2000 °/s. У овом примеру је за опсег коришћен ±250 °/s. То значи да за вредност 0% на потенциометру, приказаће се -250°/s на дисплеју, за вредност 50% на потенциометру, приказаће се 0°/s на дисплеју, и за вредност 100% на потенциометру, приказаће се +250°/s на дисплеју.

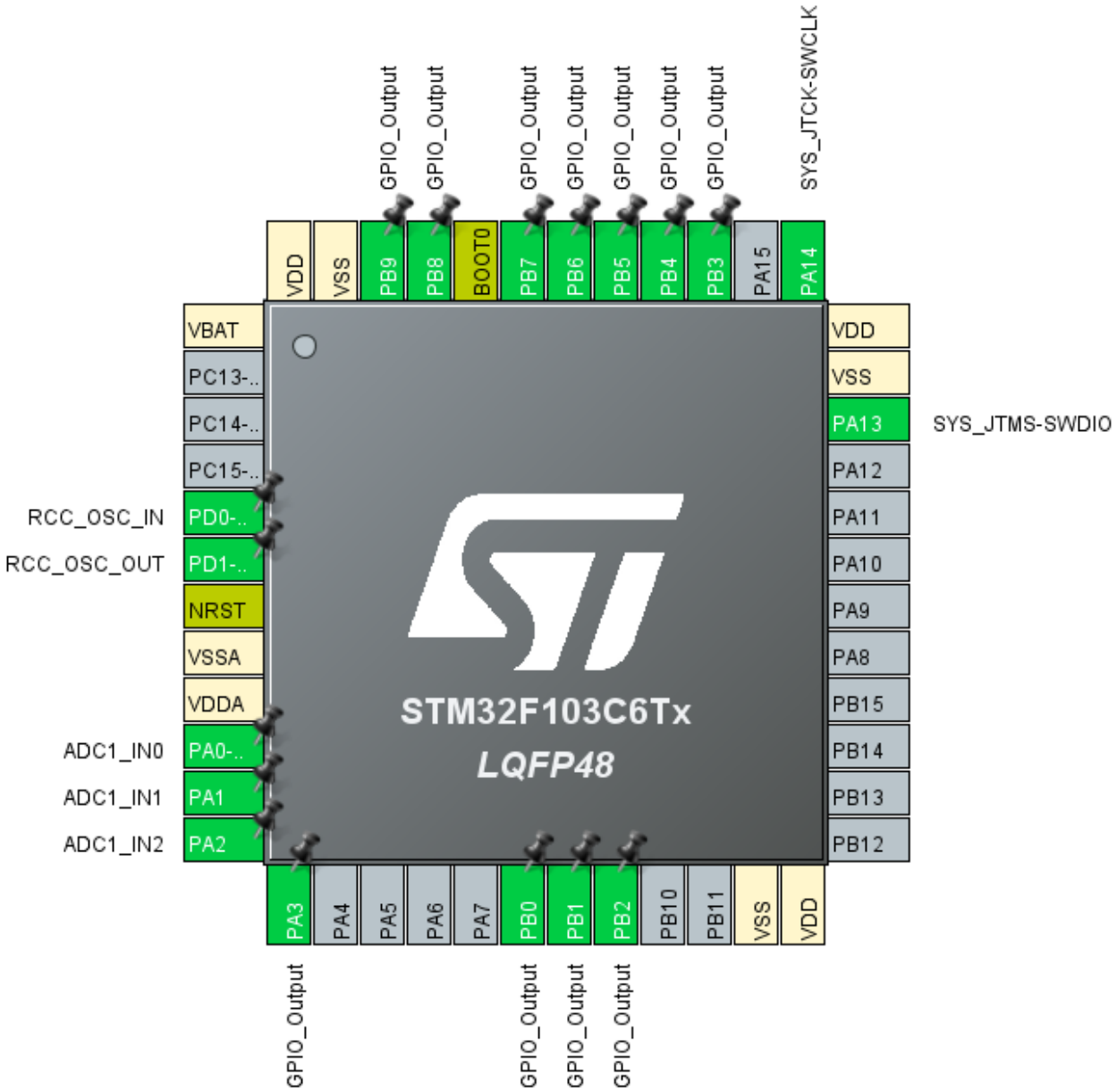


Слика 3 - Изглед шеме пројекта

РЕАЛИЗАЦИЈА РЕШЕЊА У STM32

Након креирања шеме за пројекат, било је неопходно изабрати одговарајућу картицу, и на њој одредити одговарајуће пинове. То је урађено у програму STM32CubeMX. Картица која је изабрана је STM32F103C6Tx.

За потенциометре су коришћени пинови A0-A2, затим A3 за LED диоду, B0-B9 за LCD дисплеј, и на крају PD0 і PD1 за екстерни осцилатор.



Слика 4 - Изглед картице са селектованим пиновима

Након геренисања кода било је потребно написати потребне функције за реализацију система. То је урађено у програму STM32CubeIDE.

Функције које су биле потребне ради манипулације LCD дисплејем су следеће:

```
void LCD(uint8_t val_1, uint8_t cmd)
{
    uint8_t data1;

    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, cmd); // set RS = cmd; (cmd=0)=>Command; (cmd=1) => data

    data1 = val_1 & 0x01;
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, data1);

    data1 = (val_1 >> 1) & 0x01;
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, data1);

    data1 = (val_1 >> 2) & 0x01;
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, data1);

    data1 = (val_1 >> 3) & 0x01;
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, data1);

    data1 = (val_1 >> 4) & 0x01;
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, data1);

    data1 = (val_1 >> 5) & 0x01;
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, data1);

    data1 = (val_1 >> 6) & 0x01;
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, data1);

    data1 = (val_1 >> 7) & 0x01;
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, data1);

    //Enable
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9, GPIO_PIN_SET);
    HAL_Delay(5);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9, GPIO_PIN_RESET);
}
```

Слика 5 - LCD функција

LCD функција се користи за слање података или команде дисплеју. Уколико се као вредност cmd пошаље 0, то значи да је val_1 команда, а 0 значи да су у променљивој val_1 подаци за испис.

```
// Helper function to send command to the LCD
void LCD_command(uint8_t command)
{
    LCD(command, 0); // Send command
}

// Helper function to send data to the LCD
void LCD_data(const char* str)
{
    // Iterate through each character in the string
    while (*str)
    {
        // Cast the character to uint8_t and send it to the LCD as data
        LCD((uint8_t)(*str), 1);
        str++;  // Move to the next character
    }
}

void LCD_init()
{
    LCD_command(0x38); //2 lines, 5*7 matrix
    LCD_command(0x0C); //Display on, cursor off
    LCD_command(0x06); //Increment cursor (shift to right)
    LCD_command(0x01); //Clear display screen
    LCD_command(0x80); //Force cursors to beginning (1st line)
}
```

Слика 6 - Остале LCD фунцкије

Постоје још две помоћне функције LCD_command и LCD_data које служе да проследе информације LCD функцији. LCD_data функција прима стринг и претвара сваки карактер у одговарајући тип који прима LCD функција.

LCD_init функција је неопходна за иницијализацију дисплеја, чишћење претходних вредности, постављањем курсора на почетак.

```
void Print(float number)
{
    int ceo, ostCeo=0;
    float ost;
    int i=0;
    char buffer[10];

    ceo = number;
    ost = number-ceo;
    ostCeo = ost*100;
    ostCeo=abs(ostCeo);

    sprintf(buffer,"%d",ceo);
    while(buffer[i])
    {
        LCD(buffer[i],1);
        i++;
    }

    LCD('.',1);
    sprintf(buffer,"%d",ostCeo);
    i=0;

    while(buffer[i])
    {
        LCD(buffer[i],1);
        i++;
    }
}
```

Слика 7 - Функција Print

Функција Print се користи за испис реалних бројева на дисплеј. Она ради тако што прво нађе цео број и остатак, а затим их испише на дисплеј, цифру по цифру.

```c
void ADC_Select_CH0 (void)
{
    ADC_ChannelConfTypeDef sConfig = {0};
      sConfig.Channel = ADC_CHANNEL_0;
      sConfig.Rank = 1;
      if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
      {
        Error_Handler();
      }
}

void ADC_Select_CH1 (void)
{
    ADC_ChannelConfTypeDef sConfig = {0};
      sConfig.Channel = ADC_CHANNEL_1;
      sConfig.Rank = 1;
      if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
      {
        Error_Handler();
      }
}

void ADC_Select_CH2 (void)
{
    ADC_ChannelConfTypeDef sConfig = {0};
      sConfig.Channel = ADC_CHANNEL_2;
      sConfig.Rank = 1;
      if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
      {
        Error_Handler();
      }
}
```

Слика 8 - Функције за промену канала

Обзиром на то да имамо 3 потенциометра, било је потребно направити 3 канала на ADC1 како бисмо могли да очитамо вредности. Зато је било потребно написати функције које ће мењати канал са ког се тренутно чита. То је урађено у функцијама ADC_Select_CH0, ADC_Select_CH1 и ADC_Select_CH2.

```
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_SET);
LCD_init();
LCD_data("MERIM UGAO...");
HAL_Delay(500);

HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_RESET);
LCD_init();
HAL_Delay(200);

ADC_Select_CH0();
HAL_ADC_Start(&hadc1);
HAL_ADC_PollForConversion(&hadc1, 1000);
X = HAL_ADC_GetValue(&hadc1);
HAL_ADC_Stop(&hadc1);

ADC_Select_CH1();
HAL_ADC_Start(&hadc1);
HAL_ADC_PollForConversion(&hadc1, 1000);
Y = HAL_ADC_GetValue(&hadc1);
HAL_ADC_Stop(&hadc1);

ADC_Select_CH2();
HAL_ADC_Start(&hadc1);
HAL_ADC_PollForConversion(&hadc1, 1000);
Z = HAL_ADC_GetValue(&hadc1);
HAL_ADC_Stop(&hadc1);
```

Слика 9 - While петља, први део

У while петљи главног дела програма позивамо претходно написане функције, исписујемо поруку на дисплеј, и очитавамо вредности са сва 3 потенциометра.

```
float degrees = (500.0*X)/2702.0 - 250;
LCD_data("X:");
Print(degrees);
LCD_data(" ");
LCD(223, 1);
LCD_data("/s");
HAL_Delay(500);
LCD_init();

degrees = (500.0*Y)/2702.0 - 250;
LCD_data("Y:");
Print(degrees);
LCD_data(" ");
LCD(223, 1);
LCD_data("/s");
HAL_Delay(500);
LCD_init();

degrees = (500.0*Z)/2702.0 - 250;
LCD_data("Z:");
LCD_data(" ");
Print(degrees);
LCD(223, 1);
LCD_data("/s");
HAL_Delay(500);
```

Слика 10 - While петља, други део

У другом делу while петље претварамо добијене вредности са потенциометара у вредности које одговарају опсегу +250°/s. Максимална вредност коју потенциометар може да мери је 2702 и зато је формула коју користимо: $\omega = \frac{500 \cdot x}{2702} - 250$.