УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
Трг Доситеја Обрадовића 6, 21000 Нови Сад, Република Србија
Деканат: 021 6350-413; 021 450-810; Централа: 021 485 2000
Рачуноводство: 021 458-220; Студентска служба: 021 6350-763
Телефакс: 021 458-133; e-mail: ftndean@uns.ac.rs
ИНТЕГРИСАНИ СИСТЕМ МЕНАЏМЕНТА СЕРТИФИКОВАН ОД:

# PROJEKAT IZ PRIMENJENE ELEKTRONIKE

**NAZIV PROJEKTA:**

Autonomni robot za detekciju svetlosti i izbegavanje prepreka

**MENTOR PROJEKTA:**

Prof. dr Vladimir Rajs
Msc Milan Bodić
Msc Marko Vasiljević Toskić

**PROJEKAT IZRADILI:**

Anđela Smiljanić ee133/2019
Luka Urošević ee154/2019
David Janković ee157/2019

**DATUM ODBRANE PROJEKTA:**

# Sadržaj

# 1. Uvod

Kada smo bili deca igrali smo se sa raznim igračkama koje su se kretale po unapred zadatim putanjama, bez obzira na prepreke koje predstoje. Ovaj projekat podseća na te trenutke, ali sada umesto praćene fiksne rute, reaguje na promene u okruženju, prilagođava svoje kretanje prema svetlosti i izbegava prepreke.

Cilj ovog projekta je razviti algoritam koji omogućava robotu da se kreće u pravcu suprotnom od izvora svetlosti, koristeći fotootpornike za detekciju njenog intenziteta. Postavljanje četiri fotootpornika omogućava robotu da „oseća“ svetlost iz različitih pravaca i automatski prilagođava svoj pravac kretanja – povlači se iz osvetljenih zona i traži tamnija područja.

Robot takođe koristi Bluetooth komunikaciju za primanje jednostavnih komandi "START" i "STOP", pružajući korisniku mogućnost da započne ili prekine njegovo kretanje u bilo kom trenutku. Bluetooth modul nije samo sredstvo kontrole, već i veza koja robotu daje mogućnost interakcije sa spoljnim svetom, otvarajući vrata za dalji razvoj sistema koji mogu reagovati na složenije signale.

Da bi osigurao sigurno kretanje, robot je opremljen sa dva HC-SR04 senzora za izbegavanje prepreka, čime postaje sposoban da se autonomno kreće bez rizika od sudara. Ovi senzori skeniraju okolinu ispred i iza robota, omogućavajući mu da izbegne prepreke i sačuva stabilnost u svakoj fazi kretanja.

Na taj način, ovaj projekat predstavlja savršenu kombinaciju jednostavnih tehnologija, poput fotootpornika i senzora daljine, sa osnovnim elementima robotike i komunikacije. Rezultat je robot koji uči da reaguje na svetlo i prepreke, pružajući korisniku jedinstven uvid u svet interaktivnih i autonomnih sistema.

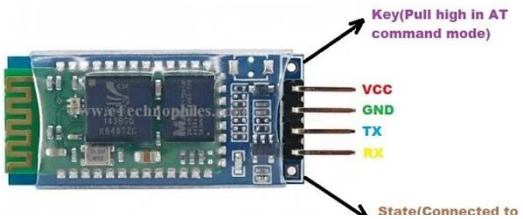
# 2. Opis elemenata projekta

## 2.1.Bluetooth HC- 06

HC-06 Bluetooth modul je jednostavan i pouzdan uređaj koji omogućava bežičnu komunikaciju sa pametnim telefonima, računarima ili drugim Bluetooth uređajima. Konfigurisan je kao **slave** uređaj, što znači da uvek čeka da se poveže sa uređajem koji je konfigurisani kao **master**, kao što su telefoni ili računari. Modul podržava Bluetooth verziju 2.0 i omogućava prenos podataka pri brzinama do 2.1 Mbps, što ga čini pogodnim za mnoge projekte koji zahtevaju pouzdanu bežičnu vezu.

HC-06 koristi radni napon od 3.3V i ima radnu struju manju od 40mA, što ga čini efikasnim u potrošnji energije. Komunikacija sa ovim modulom ostvaruje se putem **UART** interfejsa, što omogućava jednostavno povezivanje sa mikrokontrolerima i drugim uređajima koji podržavaju serijski port. Modul dolazi sa integrisanom antenom koja omogućava stabilan prenos podataka na udaljenosti do 10 metara, što je dovoljno za mnoge aplikacije gde se traži bežična kontrola ili prenos podataka.

Povezivanje HC-06 sa mikrokontrolerom je prilično jednostavno. Modul ima četiri pina – VCC, GND, TX i RX. VCC pin se povezuje na napajanje od 3.3V, GND na uzemljenje, dok se TX i RX koriste za slanje i primanje podataka.



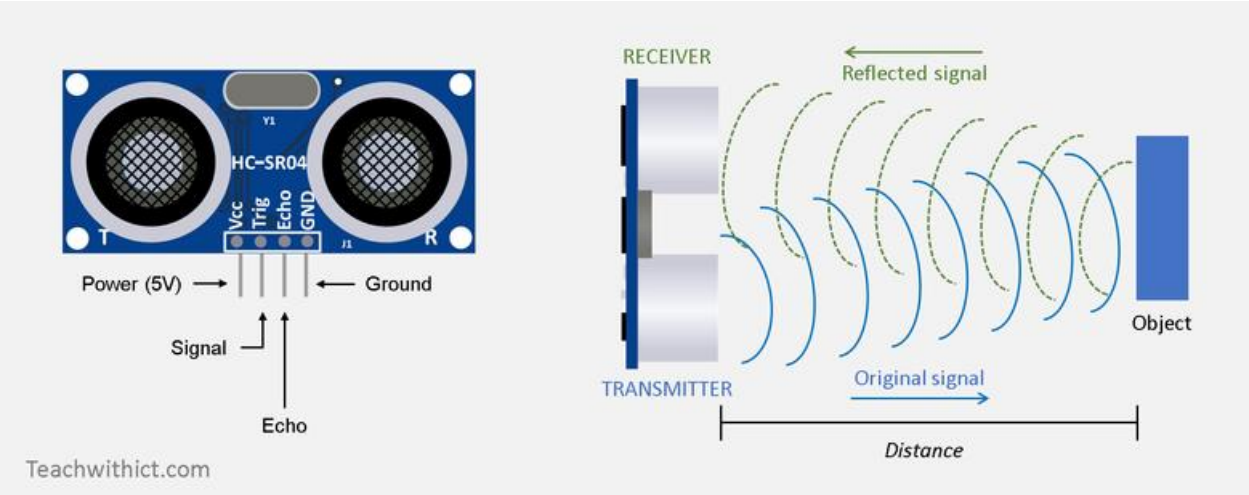

**Slika 1 Prikaz Bluethooth modul-a**

Kada se uključi, HC-06 postaje dostupan za uparivanje, a pametni telefon ili računar mogu ga pronaći putem Bluetooth pretrage. Nakon što je uparen, uređaj može slati i primati podatke preko serijskog interfejsa. To omogućava jednostavnu implementaciju bežične kontrole robota, prikupljanje podataka sa senzora ili daljinsko upravljanje različitim uređajima.

HC-06 je široko korišćen u projektima koji zahtevaju bežičnu komunikaciju, posebno u robotici, Internet of Things (IoT) uređajima i aplikacijama gde je potrebna jednostavna bežična interakcija sa mikrokontrolerom. [1]

## 2.2.Ultrazvučni senzor HC-SR04

Ovaj projekat koristi HC-SR04 ultrazvučni senzor kako bi robot mogao da detektuje prepreke i meri udaljenost od objekata u svom okruženju. Cilj je da robot autonomno izbegava prepreke, koristeći podatke o udaljenosti u realnom vremenu, što omogućava glatko i bezbedno kretanje. Senzor funkcioniše tako što emituje ultrazvučni signal koji se odbija od objekata, a zatim meri vreme potrebno da se taj signal vrati. Na osnovu izmerenog vremena, izračunava se udaljenost do objekta. HC-SR04 može meriti udaljenost u rasponu od 2 cm do 4 metra, pružajući precizne i pouzdane informacije.




***Slika 2.2a Ultrazvučni senzor***

Komunikacija sa senzorom je jednostavna i ostvaruje se putem četiri pina – VCC, GND, Trigger i Echo. VCC pin se povezuje na napajanje od 3.3V ili 5V, dok se GND povezuje na uzemljenje. Trigger pin je zadužen za slanje impulsa mikrokontrolera, a Echo pin vraća signal koji predstavlja dužinu trajanja eha. Na osnovu ovog povratnog signala mikrokontroler izračunava vreme koje je bilo potrebno da se ultrazvučni talas odbije od prepreke i vrati, a zatim na osnovu toga računa udaljenost do objekta koristeći formulu:

$$D = \frac{V * T}{2}$$

Gde je D rastojanje, V brzina zvuka (340 m/s), a T proteklo vreme od emitovanja do povratka talasa. Proizvod brzine zvuka i proteklog vremena deli se sa dva jer talasi putuju od modula do prepreke i nazad.

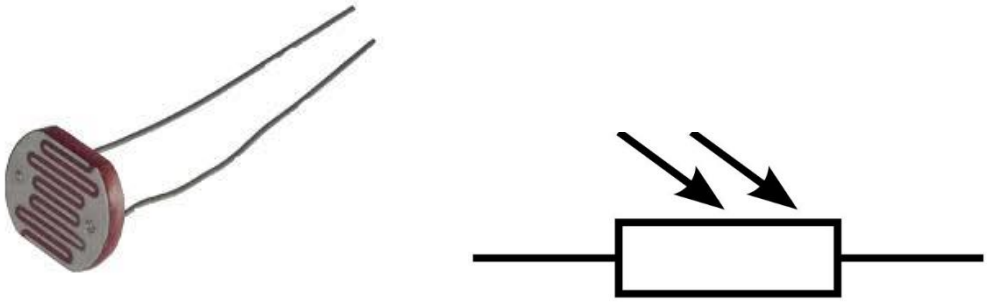
Brzina zvuka je ključna za precizno merenje udaljenosti. Na primer, na temperaturi od 20°C u suvom vazduhu, brzina zvuka je oko 340 m/s. Ovo daje dovoljno vremena da senzor pošalje impuls i primi povratni signal. Međutim, brzina prostiranja zvuka može značajno varirati u različitim sredinama – u tečnostima može biti preko 1000 m/s, a u čvrstim materijalima i nekoliko hiljada metara u sekundi. S obzirom na ove faktore, HC-SR04 je pouzdano rešenje za detekciju prepreka u vazduhu, omogućavajući robotima da izbegnu sudare u realnom vremenu.

Zahvaljujući jednostavnosti korišćenja, ovaj senzor je pogodan za različite aplikacije, od robotike i detekcije prepreka, do naprednijih sistema za merenje i navigaciju. [2]

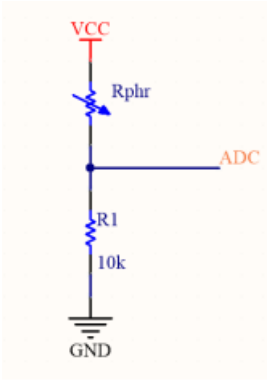
## 2.3.Fotootpornik

Ovaj otpornik je izrađen od poluprovodnog materijala. Kada fotoni svetlosti padaju na fotootpornik sa dovoljno visokom frekvencijom (odnosno dovoljnom energijom), oni pružaju vezanim elektronima dovoljnu energiju da postanu slobodni elektroni u poluprovodniku. Ovaj proces rezultuje smanjenjem otpora fotootpornika pod uticajem svetlosti.

Fotootpornici imaju prednosti u vidu niske cene i pouzdanosti, ali imaju i svoje nedostatke, među kojima je spor odziv. Naime, obično je potrebno nekoliko sekundi za potpunu promenu otpora fotootpornika prilikom naglog promenjenog osvetljenja. Ovo često može biti ograničavajuće u situacijama gde je brza reakcija na promene u osvetljenju od suštinskog značaja. [3]



***Slika 2a izgled fotootpornika(levo) i šematski prikaz(desno)***




***Slika 2b Izgled šematika za povezivanje fotootporinika sa mikrokontrolerom***

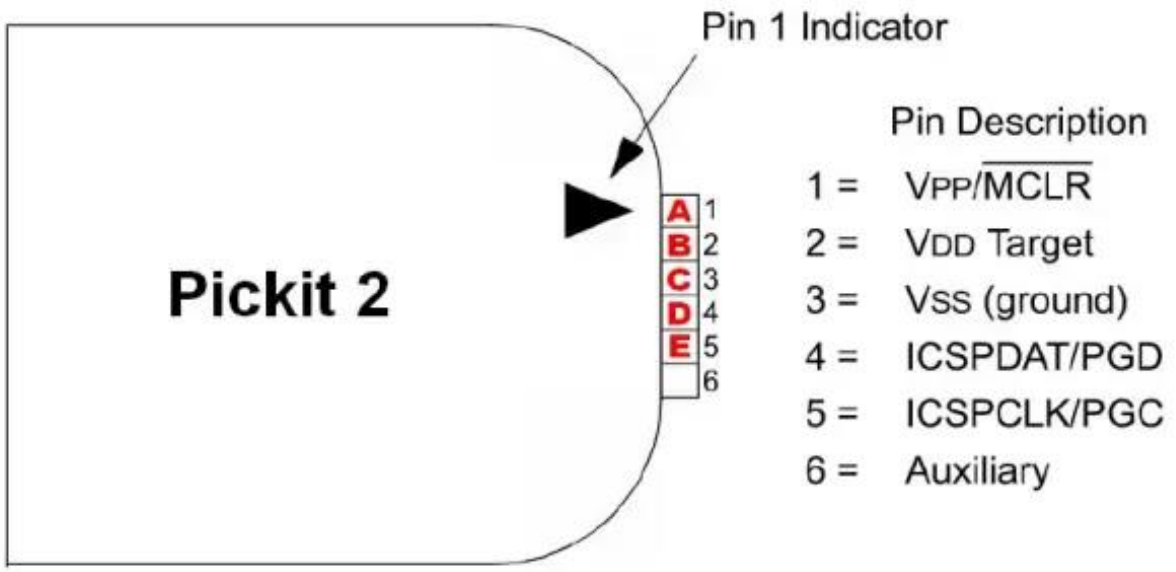
## 2.4.Programator

PICkit je serija programatora za PIC mikrokontrolere koju proizvodi Microchip Technology. Ovi uređaji se koriste za programiranje mikrokontrolera, otklanjanje grešaka i programiranje EEPROM memorija. U našem projektu koristili smo PICkit 2, koji se izdvaja po tome što ima odvojenu jedinicu za programiranje i debagovanje koja se povezuje sa pločom na kojoj se nalazi čip koji treba programirati.


PICkit 2 dolazi sa 128 KB memorije kao standardnom opremom, dok je moguće proširiti kapacitet na 256 KB modifikovanjem hardvera ili korišćenjem klonova trećih strana. Ovaj programator takođe uključuje ugrađeni trokanalni logički analizator od 500 kHz i UART alat, što dodatno olakšava analizu i debagovanje sistema.

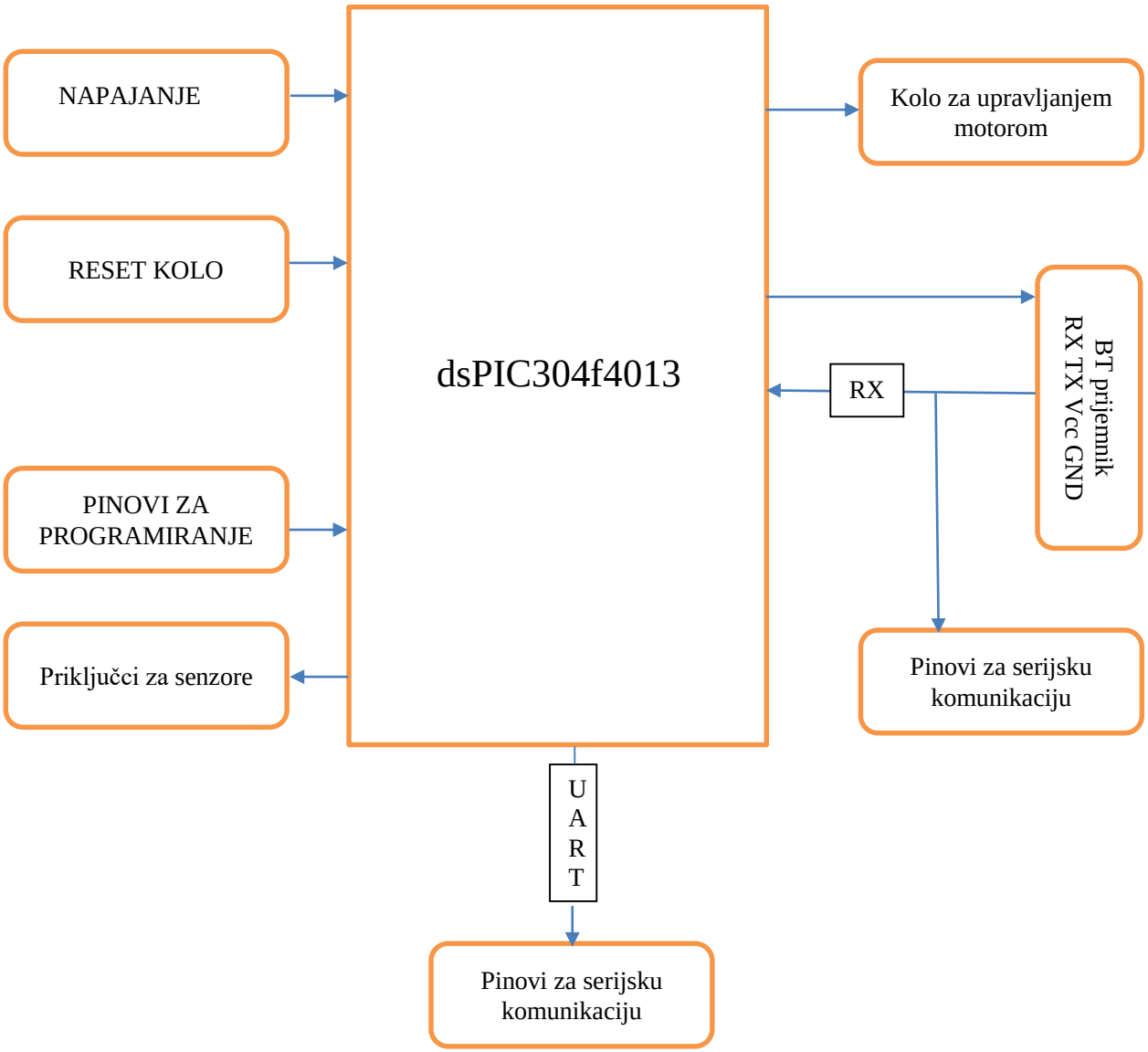
***Slika 2.4a*** **Pickit 2**

Sistem PICkit 2 sadrži pet ključnih pinova za programiranje mikrokontrolera: VDD (napajanje), VSS (masa), PGD (Program Data), PGC (Program Clock) i MCLR (RESET), uz dodatni šesti slobodni pin. Pinovi PGC i PGD su povezani sa RB6 i RB7 pinovima mikrokontrolera, omogućavajući efikasnu komunikaciju i programiranje. [4]

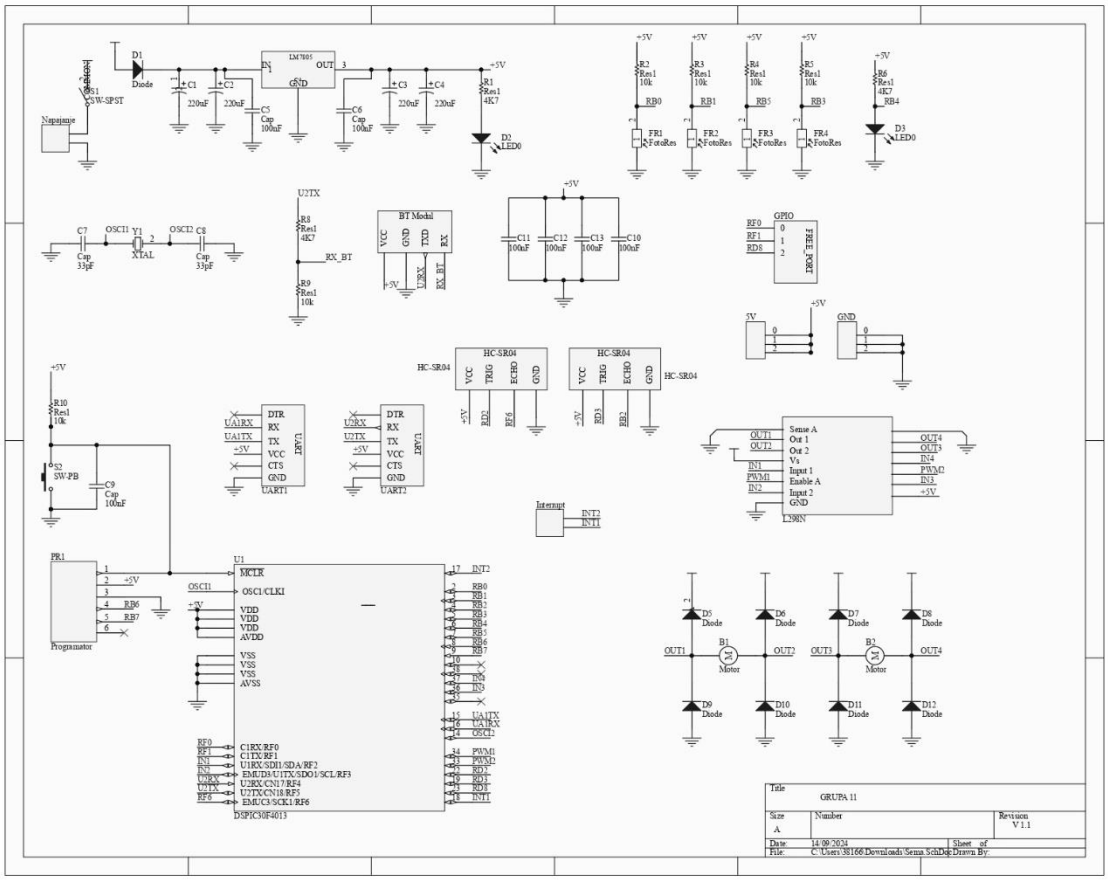



***Slika 2.4b*** **Pickit 2 pinovi**

# 3. HARDVERSKA STRUKTURA



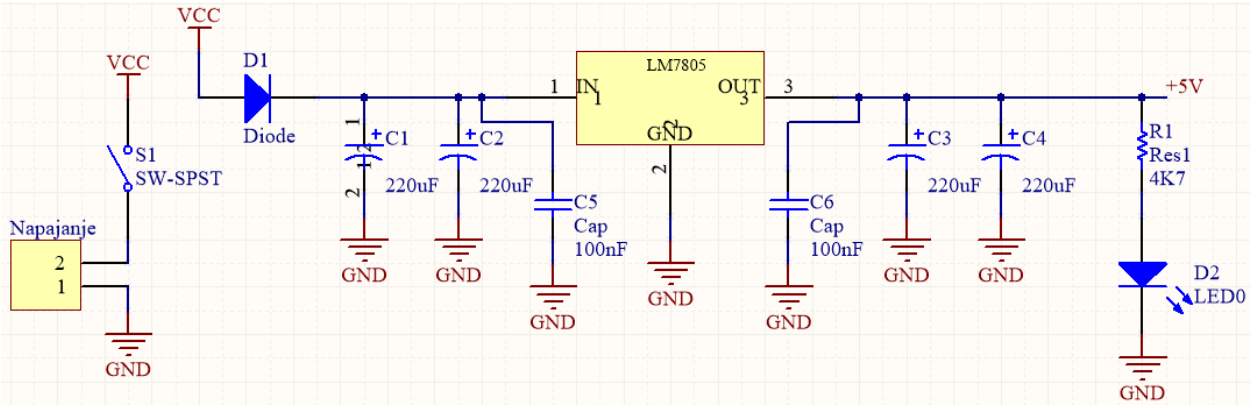
# 4. ŠEMATIK




***Slika 4.1a*** **Izgled šematika u Altium Designer-u**

# 5. OPIS PODSISTEMA ŠEMATIKA

## *5.1.* Blok za napajanje

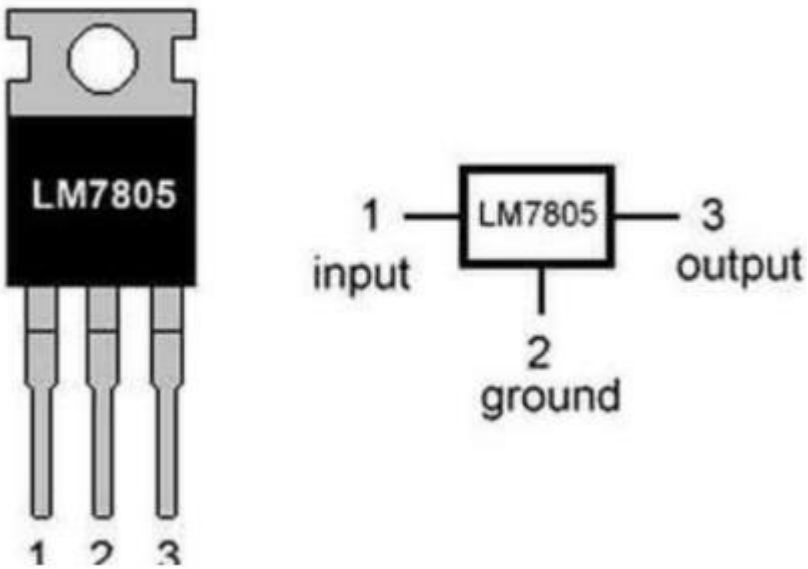



***Slika 5.1a*** **Šematski prikaz bloka za napajanje**

Glavni deo ovog segmenta čini stabilizator napona LM7805.

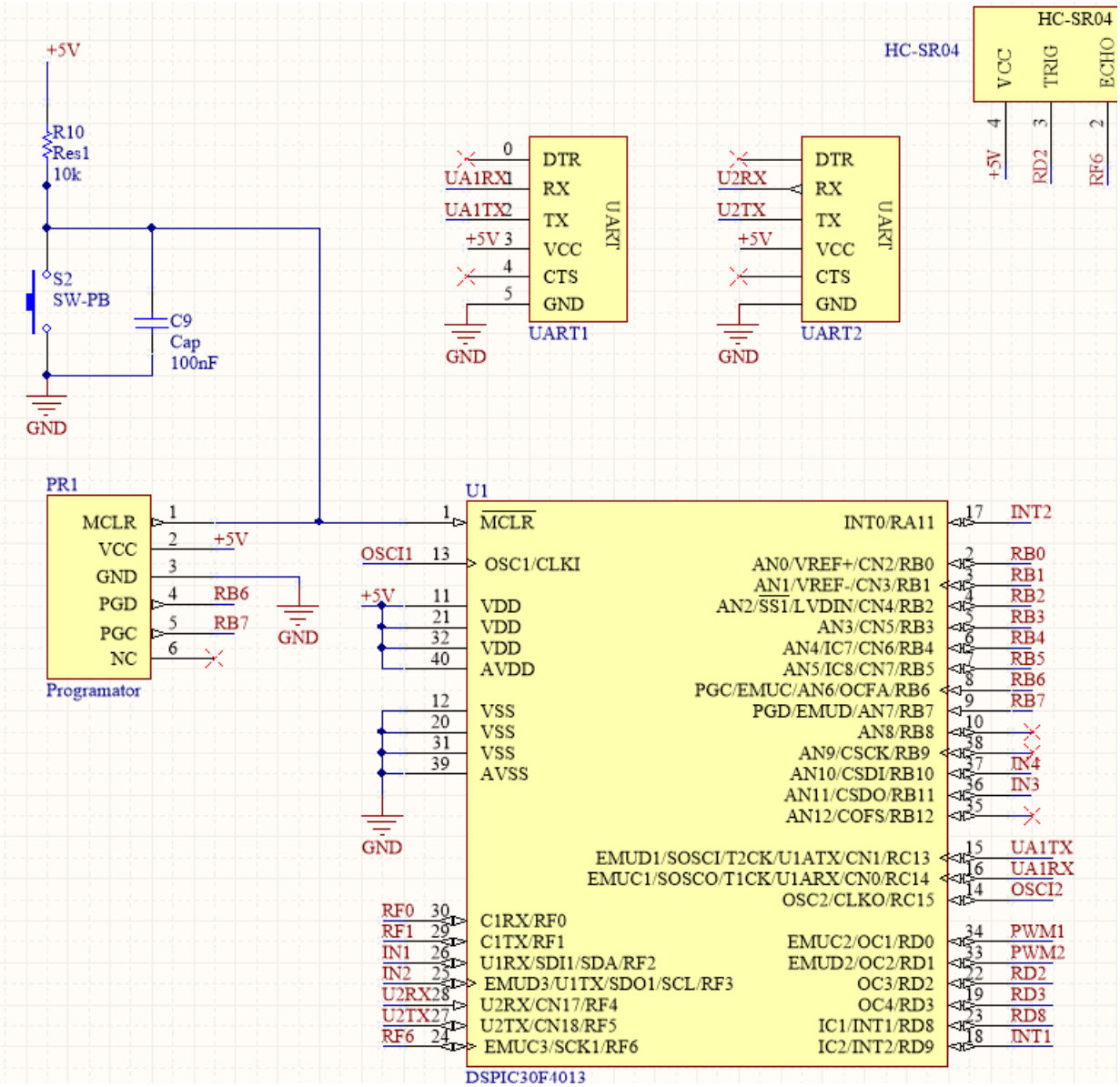
On predstavlja jedan od najčešćih naponskih regulatora korišćen za ograničavanje napona. Na izlazu naponskog regulatora dobija se vrednost od 5V sa 2% maksimalne devijacije.

Kako bi se ostvario stabilan izlaz, neophodno je napajati kolo sa minimum 7V. Za potrebe projekta je upotrebljeno napajanje od 9V. Na izlazu kola nalazi se LED koja služi za indikaciju da li je kolo u funkciji. [5]



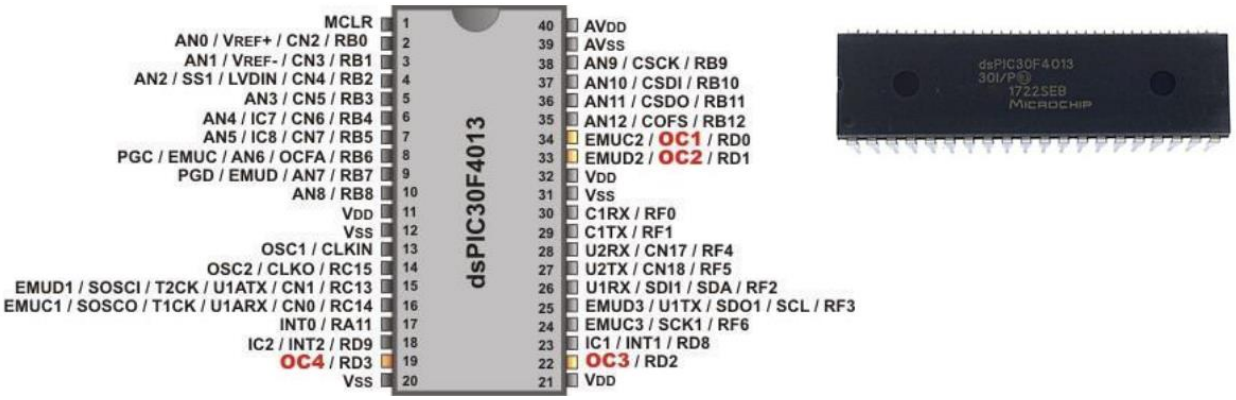

***Slika 5.1b*** **Izgled kućišta LM7805 sa rasporedom pinova**

## *5.2.* Mikrokontroler dsPIC30f4013



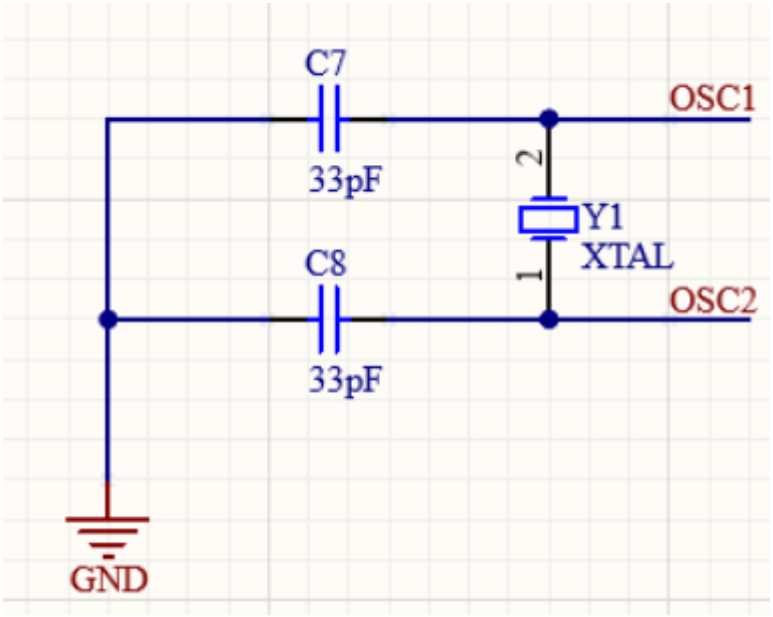

***Slika 5.2a*** **Šematski prikaz mikrokontrolera dsPIC30f4013**

Mikrokontroleri su postali neophodni element svakog elektronskog uređaja, igrajući ključnu ulogu u prikupljanju podataka, ograničene obrade tih podataka i upravljanju okruženjem na osnovu rezultata izračunavanja. U okviru ovog projekta, izabran je mikrokontroler dsPIC30f4013 proizvođača Microchip. Ovaj mikrokontroler ima radni napon radni napon u rasponu od 2.5V do 5.5V. Poseduje pet vrsta portova (A,B,C,D,E,F), pet tajmera, dva UART modula sa FIFO baferima i 12-bitni Analogno-Digitalni (A/D) konvertor sa 13 ulaznih kanala. Njegove karakteristike uključuju nisku potrošnju energije i visoku brzinu rada. [6]



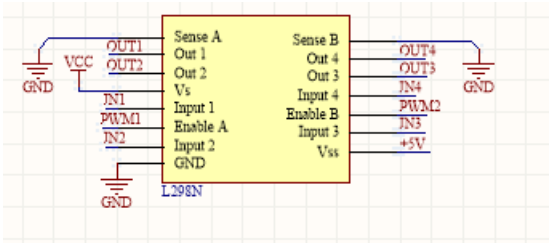

***Slika 5.2b Raspored priključaka Mikrokontrolera dsPIC30f4013(levo) i izgled mikrokontrolera(desno)***

U zadatku smo koristili eksterni oscilator čiji je instrukcioni takt iznosi 10MHz.



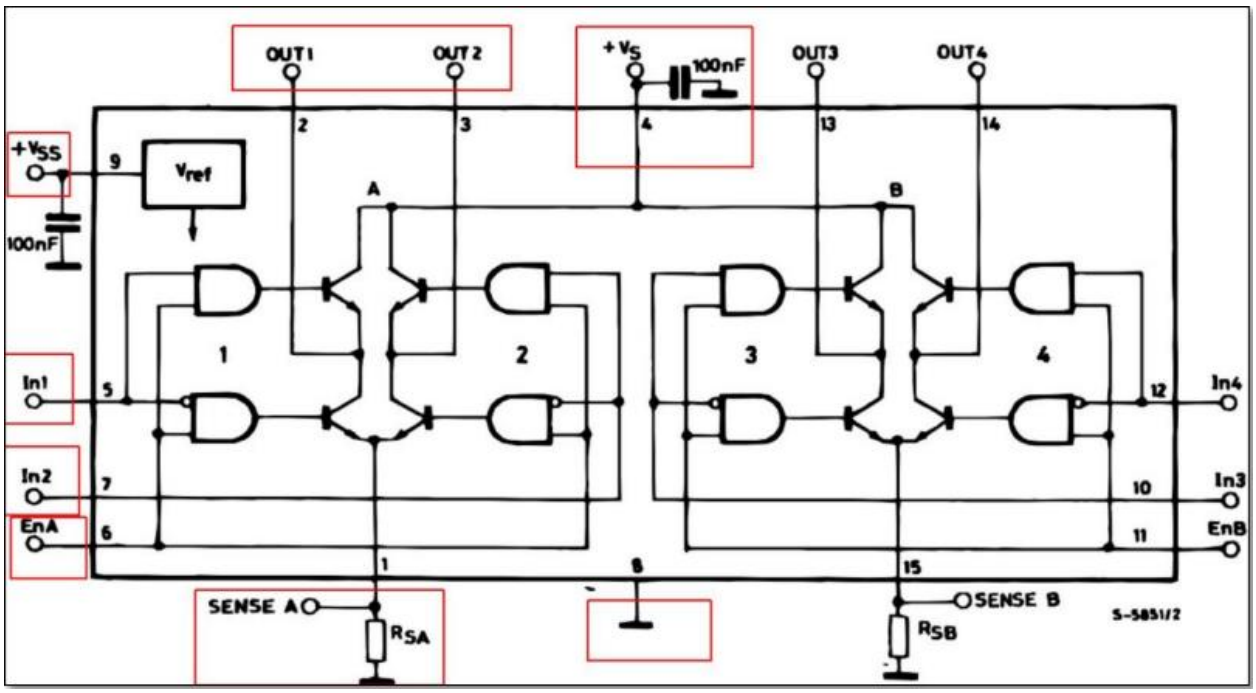

***Slika 5.2c Prikaz oscilatora***

## 5.3. Drajver za motor




*Slika 5.3a Prikaz šematika drajvera L298N*

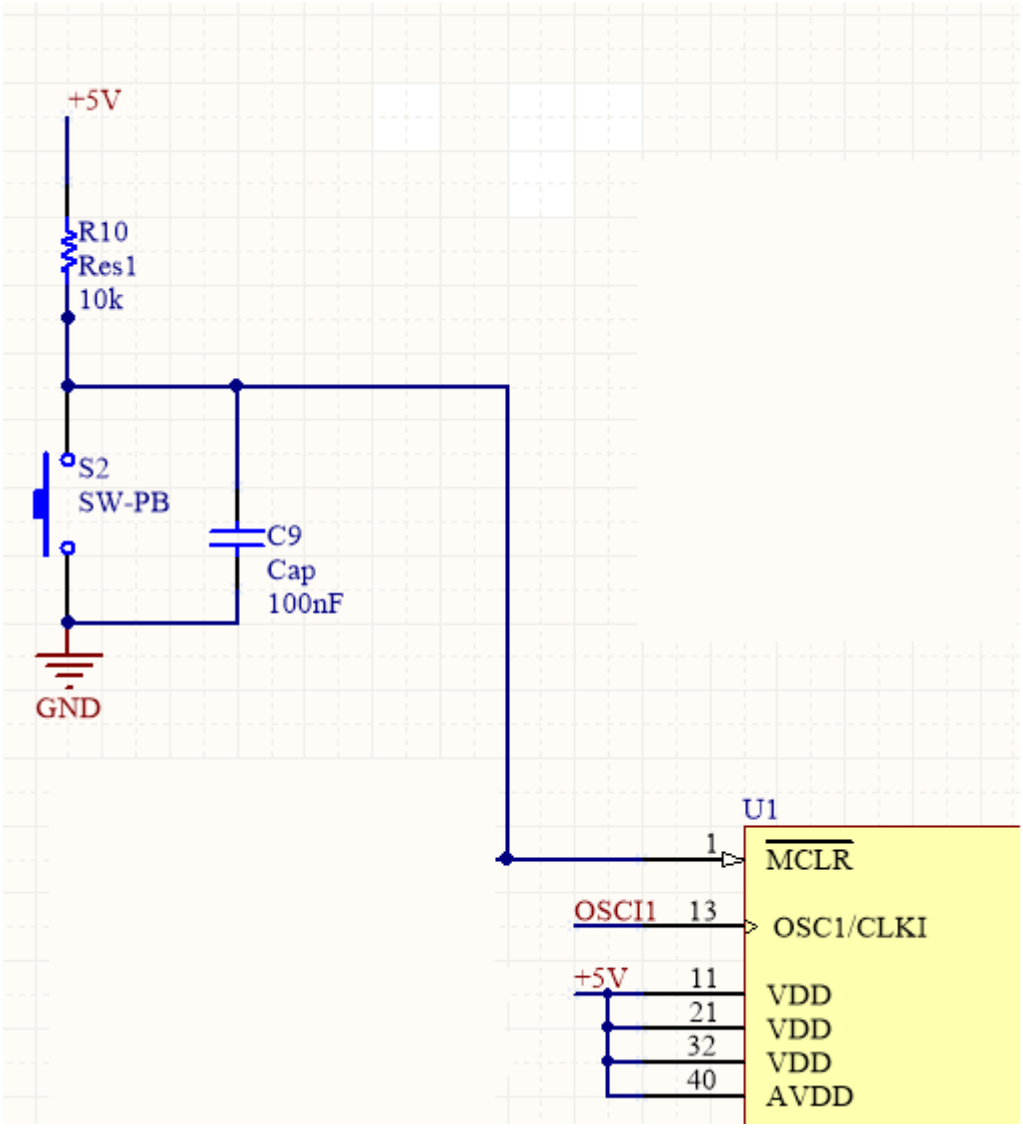
L298N je visokokvalitetni drajver koji podržava dvostruku H-most konfiguraciju za upravljanje motorima. Sa svojim integrisanim H-mostovima i zaštitnim funkcijama, L298N obezbeđuje pouzdanu i efikasnu kontrolu motora. Drajver ima mogućnost upravljanja motorima sa radnim naponom do 35V i strujom do 2A po kanalu. [7]




*Slika 5.3b Elktrična šema drajvera L298N*


*Slika 5.3c Raspored pinova (levo)i izgled kućišta drajvera L298N(desno)*

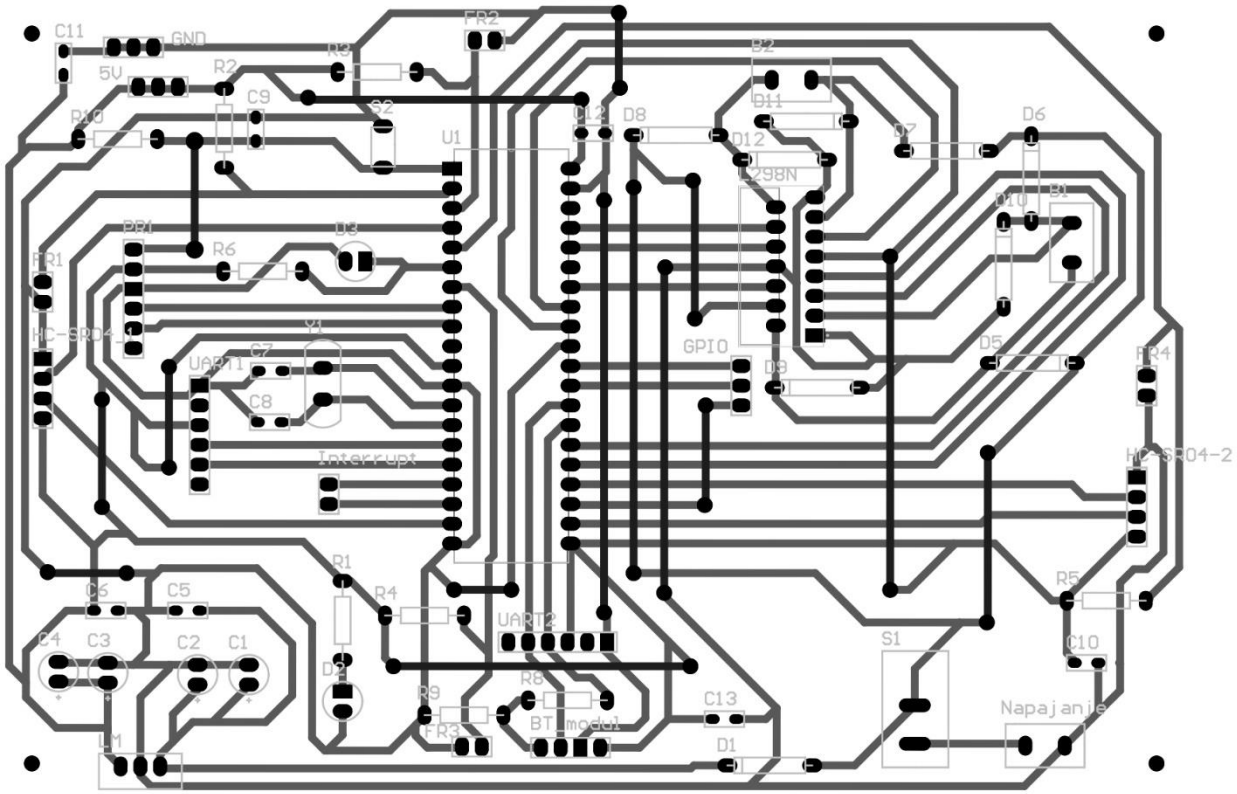
## 5.4. Reset kolo



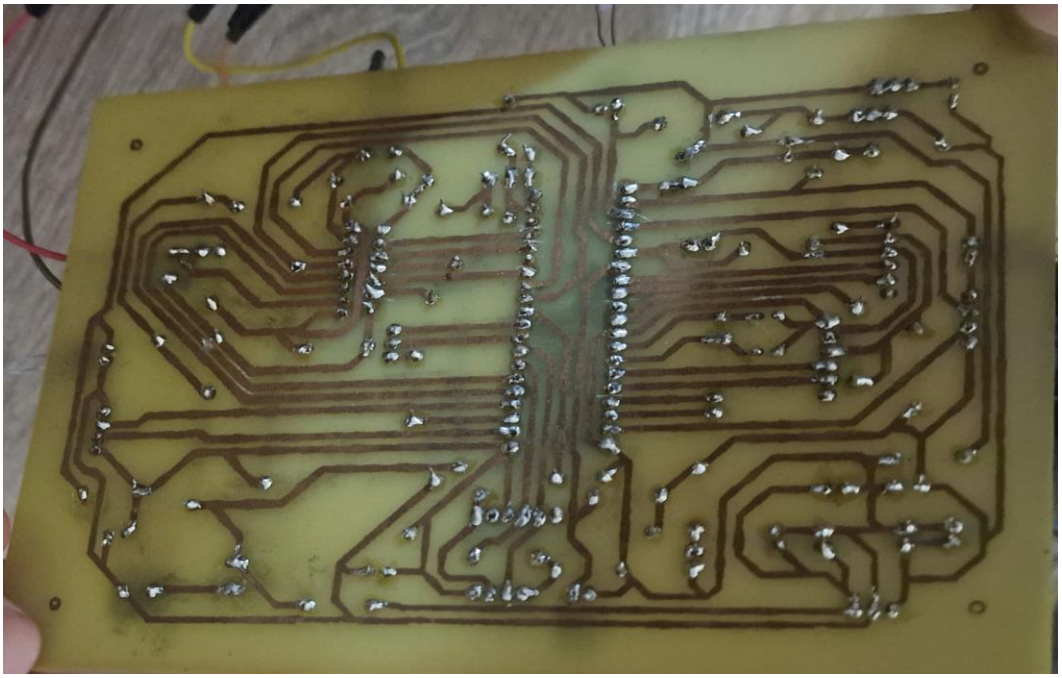

***Slika 5.4a Šematik reset kola***

Reset kolo služi da se resetuje mikrokontroler. Realizovano je u vidu pull-up tastera, kod koga se taster nalazi paralelno kondenzatoru (100pF), jedan kraj ide na masu, drugi na otpornik koji se povezuje na napajanje (5V). Kondenzator služi kao hardversko rešenje poskakivanju tastera (eng. debouncing). Izlaz reset kola je izmedju otpornika i paralelne veze tastera i kondenzatora. Pritiskom na taster dobijamo nizak logički nivo, što signalizira sistemu da izvrši reset i odgovara samom $\overline{MCLR}$ na koji se vezuje izlaz našeg reset kola. Kada se taster otpusti prekida se putanja ka masi i imamo visok logički nivo i vraćamo se u stanje mirovanja, neakativno stanje reset kola.

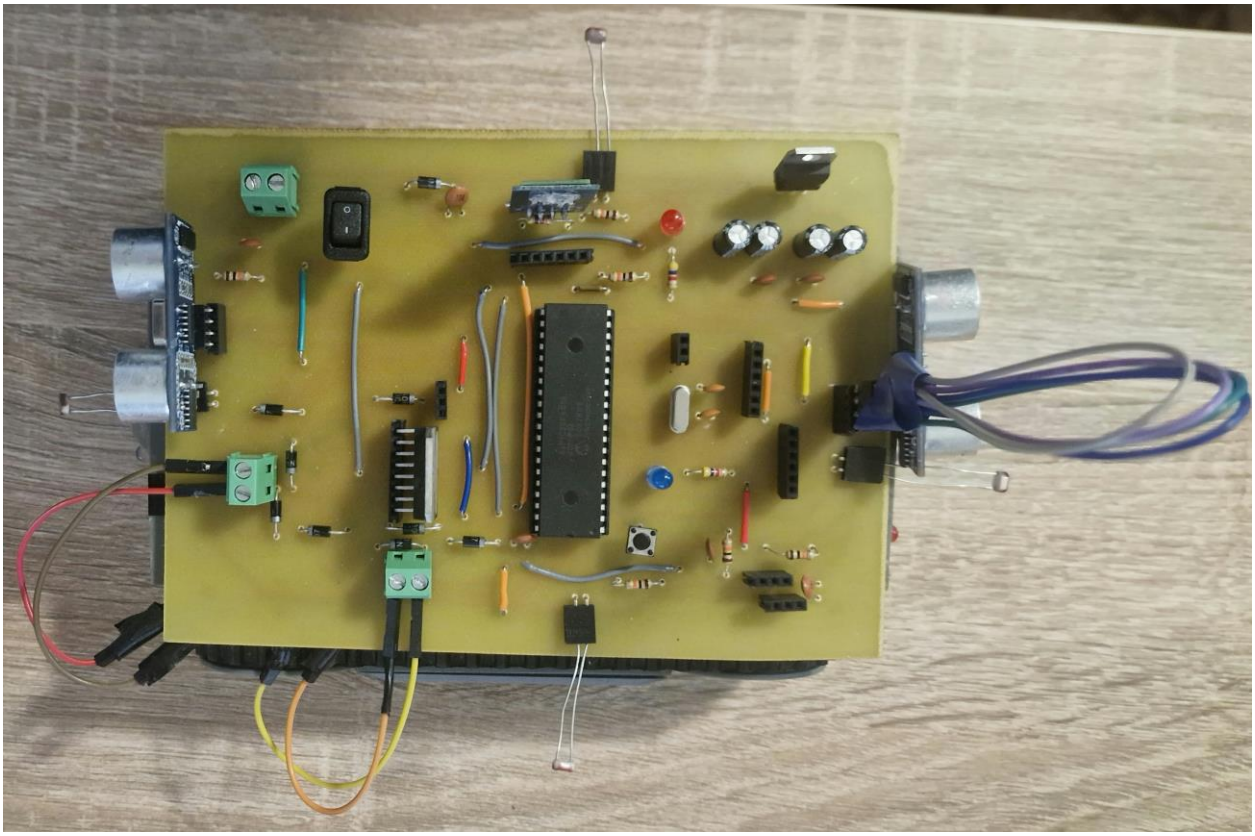
# 6. Prikaz PCB




***Slika 6a Izgled PCB-a u Altium Designer-u***



***Slika 6b Izgled PCB-a nakon izrade***

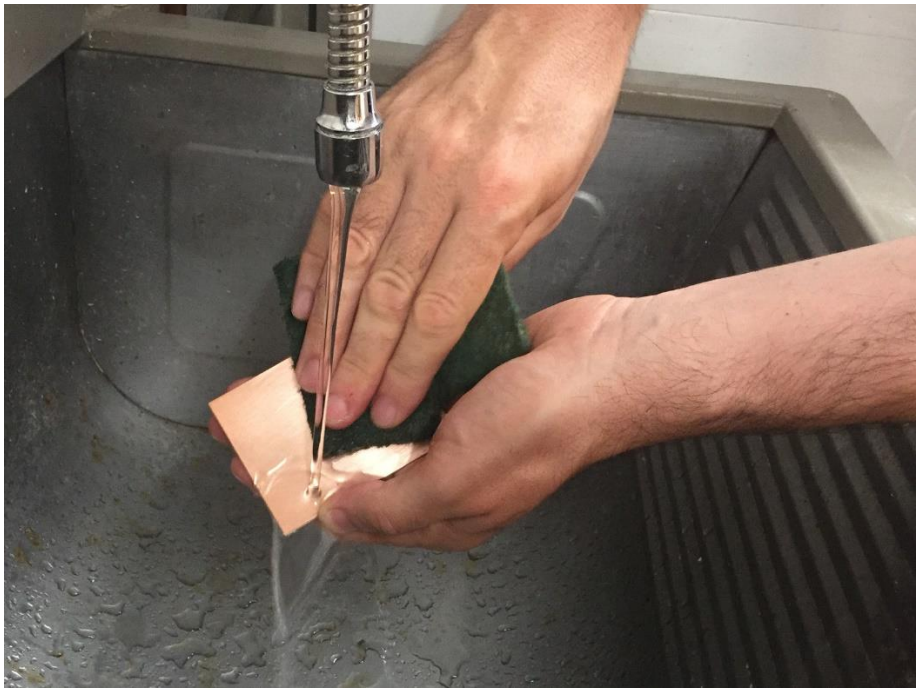


***Slika 6c Izgled povezanog autića spreman za test***

# 7. Izrada pločice

Koraci prilikom izrade štampane pločice:

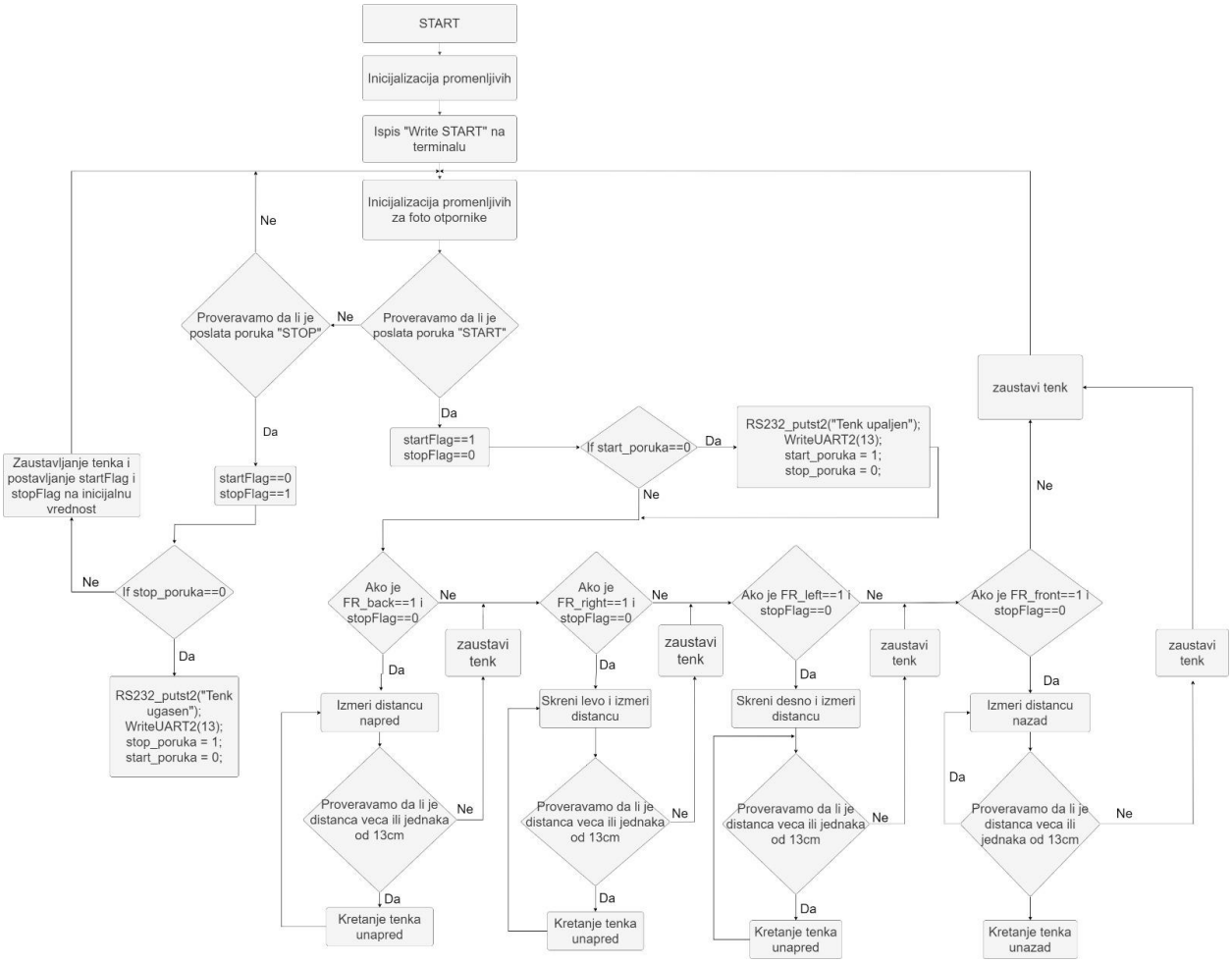
- Priprema vitroplasta: Šmirglanje i čišćenje pločice.
- Prebacivanje dizajna: Prenos dizajna na pločicu peglanjem.
- Utapanje u kiselinu: Nagrizanje bakra na nepokrivenim delovima dizajna.
- Bušenje rupa: Izrada rupa za komponente.
- Postavljanje komponenata: Instalacija svih elektronskih delova.
- Testiranje: Provera funkcionalnosti pločice.



***Slika 7a Način izrade PCB-a***

# 8. Algoritam rada

## *8.*1Prikaz algoritma




***Slika 8a Izgled algoritma rada***

## 8.2Realizacija main-a

```
#include <stdio.h>
#include <stdlib.h>
#include <p30fxxxx.h>
#include <outcompare.h>
#include <string.h>
#include "adc.h"
#include "tajmer.h"
#include "tajmer_pwm.h"
#include "UART.h"
#include "fotootpornik.h"
#include <stdint.h>
#define BUFFER_SIZE 6

//#define TRIG_NAPRED LATDbits.LATD2
//#define ECHO_NAPRED PORTFbits.RF6

//#define TRIG_NAZAD LATDbits.LATD3
//#define ECHO_NAZAD PORTBbits.RB2


#define TRIG_NAPRED LATDbits.LATD3
#define ECHO NAPRED PORTBbits.RB2

#define TRIG_NAZAD LATDbits.LATD2
#define ECHO_NAZAD PORTFbits.RF6


#define SPEED_OF_SOUND (0.0343)         // centimetri po mikcrosec.
#define INSTRUCTION_CLOCK_PERIOD (0.1) // microsekundi

_FOSC(CSW_FSCM_OFF &XT_PLL4); // instruction takt je isti kao i kristal 10MHz
_FWDT(WDT_OFF);

unsigned char tempRX1, tempRX2_bluetooth;
unsigned int us_counter;
unsigned int sirovi0, sirovi1, sirovi2, sirovi3;
unsigned int FR_front, FR_back, FR_left, FR_right;
int startFlag = 0;
int stopFlag = 0;

char buffer[BUFFER_SIZE];
unsigned int bufferIndex = 0;

static unsigned char time_overflow_back = 0;
static unsigned char time_overflow_forward = 0;
static float measured_distance_back = 0;
static float measured_distance_forward = 0;

void __attribute__((__interrupt__, no_auto_psv)) _T1Interrupt(void) // svakih
1us
{
TMR1 = 0;

us_counter++; // brojac za funkciju delay_1us()

IFS0bits.T1IF = 0;
}

void __attribute__((__interrupt__, no_auto_psv)) _T2Interrupt(void) // za PMW
```

```
{
TMR2 = 0;
IFS0bits.T2IF = 0;
}

void __attribute__((__interrupt__, no_auto_psv)) _T4Interrupt(void)//za dig
senzor nazad
{
    TMR4 = 0;
    ECHO_NAZAD = 0;
    time_overflow_back = 1;
    IFS1bits.T4IF = 0;
}

void __attribute__((__interrupt__, no_auto_psv)) _T5Interrupt(void) // za dig
senzor napred
{
    TMR5 = 0;
    ECHO_NAPRED = 0;
    time_overflow_forward = 1;
    IFS1bits.T5IF = 0;
}

void __attribute__((__interrupt__, no_auto_psv)) _U2RXInterrupt(void) {
    IFS1bits.U2RXIF = 0;

    tempRX2_bluetooth = U2RXREG;

    //proveri da li je primljeni karakter prelaz u novi red
    //Provera da li je primljen karakter novi red
    if (tempRX2_bluetooth == '\n' || tempRX2_bluetooth == '\r') {
        // Null- prekida ga
        buffer[bufferIndex] = '\0';

        // Slanje poruke "START"
        if (strcmp(buffer, "START") == 0) {
            //startFlag postavljamo na 1
            startFlag = 1;

        }// Slanje poruke "STOP"
        else if (strcmp(buffer, "STOP") == 0) {
            //stopFlag postavljamo na 1
            stopFlag = 1;
        }
        //Resetujemo bufferIndex za sledeci prijem
        bufferIndex = 0;
    }//Proveravamo da li je buffer pun
    else if (bufferIndex < BUFFER_SIZE - 1) {
        //Dodajemo primljen karakter u bafer
        buffer[bufferIndex] = tempRX2_bluetooth;
        bufferIndex++;
    }
}

void __attribute__((__interrupt__, no_auto_psv)) _U1RXInterrupt(void) {
    IFS0bits.U1RXIF = 0;
    tempRX1 = U1RXREG;
}

void __attribute__((__interrupt__, no_auto_psv)) _ADCInterrupt(void) {
```

```
    sirovi0 = ADCBUF0;
    sirovi1 = ADCBUF1;
    sirovi2 = ADCBUF2;
    sirovi3 = ADCBUF3;

    IFS0bits.ADIF = 0;
}

void initPins() {

    // FOTOOTPORNICI analogni
    ADPCFGbits.PCFG0 = 0; // levi-fotoR na RB0
    ADPCFGbits.PCFG1 = 0; // gornji-fotoR na RB1
    ADPCFGbits.PCFG5 = 0; // donji-fotoR na RB5
    ADPCFGbits.PCFG3 = 0; // desni-fotoR na RB3

    TRISBbits.TRISB0 = 1; // ulazni
    TRISBbits.TRISB1 = 1; // ulazni
    TRISBbits.TRISB5 = 1; // ulazni
    TRISBbits.TRISB3 = 1; // ulazni

    // ULTRAZVUCNI NAPRED
    TRISDbits.TRISD2 = 0; // trigger
    TRISFbits.TRISF6 = 1; // echo

    // ULTRAZVUCNI NAZAD
    TRISDbits.TRISD3 = 0; // trigger

    ADPCFGbits.PCFG2 = 1; // digitalni pin
    TRISBbits.TRISB2 = 1; // echo

    // pinovi za kretanje
    TRISFbits.TRISF2 = 0; // in1
    TRISFbits.TRISF3 = 0; // in2

    ADPCFGbits.PCFG10 = 1;
    TRISBbits.TRISB10 = 0; // in4

    ADPCFGbits.PCFG11 = 1;
    TRISBbits.TRISB11 = 0; // in3

    // pwm
    TRISDbits.TRISD1 = 0; // enB
    TRISDbits.TRISD0 = 0; // enA
}

void delay_1us(int vreme) {
    us_counter = 0;
    T1CONbits.TON = 1;
    while (us_counter < vreme)
        ;
    T1CONbits.TON = 0;
}

void delay_for(uint32_t num) // unsigned int ide do 65535
{
    uint32_t broj;
    for (broj = 0; broj < num; broj++)
        ;
}
```

```
static void MeasureBackDistance() {
    //logicka jedinica traje 10us
    TRIG_NAZAD = 1;
    delay_1us(3); //3 od 10 da bi nam logicka jedinica trajala 10us
    TRIG_NAZAD = 0;
    delay_1us(3);
    while (!ECHO_NAZAD)
        ; //echo pin postaje 1(rastuca ivica detektovana)
    TMR4 = 0; // resetuje se tajmer T4
    IFS1bits.T4IF = 0;
    T4CONbits.TON = 1; // ukljucujemo tajmer T4
    while (ECHO_NAZAD)
        ; //echo pin postaje 0(opadajuca ivica detektovana)
    T4CONbits.TON = 0; //isljucujemo tajmer T4
    unsigned int measured_time_back;
    if (time_overflow_back == 1) //
    {
        measured_time_back = TMR4_period;
        time_overflow_back = 0;
    } else //poslati signal se vratio
    {
        measured_time_back = TMR4;
    }
    TMR4 = 0;
    // delimo sa 2 jer svetlosni signal putuje do prepreke i nazad
    // mnozimo sa INSTRUCTION_CLOCK_PERIOD da bi dobili vreme u
mikrosekundama
    measured_distance_back = (measured_time_back * INSTRUCTION_CLOCK_PERIOD)
/ 2 * SPEED_OF_SOUND;
}

static void MeasureForwardDistance() {

    TRIG_NAPRED = 1;
    delay_1us(3);
    TRIG_NAPRED = 0;
    delay_1us(3);
    while (!ECHO_NAPRED)
        ;
    TMR5 = 0;
    IFS1bits.T5IF = 0;
    T5CONbits.TON = 1;
    while (ECHO_NAPRED)
        ;
    T5CONbits.TON = 0;
    unsigned int measured_time_forward;
    if (time_overflow_forward == 1)
    {
        measured_time_forward = TMR5_period;
        time_overflow_forward = 0;
    } else
    {
        measured_time_forward = TMR5;
    }
    TMR5 = 0;
    measured_distance_forward = (measured_time_forward *
INSTRUCTION_CLOCK_PERIOD) / 2 * SPEED_OF_SOUND;
}

void idiNapred() {
```

```
    //   OC1RS=250;
    //  OC2RS=250;

    LATFbits.LATF2 = 0; // in1
    LATFbits.LATF3 = 1; // in2
    LATBbits.LATB11 = 1; // in3
    LATBbits.LATB10 = 0; // in4
}

void idiNazad() {

    // OC1RS=385;
    // OC2RS=400;

    LATFbits.LATF2 = 1; // in1
    LATFbits.LATF3 = 0; // in2
    LATBbits.LATB11 = 0; // in3
    LATBbits.LATB10 = 1; // in4
}

void idiLevo() {

    //    OC1RS=250;
    //   OC2RS=250;

    LATFbits.LATF2 = 1; // in1
    LATFbits.LATF3 = 0; // in2
    LATBbits.LATB11 = 1; // in3
    LATBbits.LATB10 = 0; // in4
}

void idiDesno() {

    //  OC1RS=250;
    // OC2RS=250;

    LATFbits.LATF2 = 0; // in1
    LATFbits.LATF3 = 1; // in2
    LATBbits.LATB11 = 0; // in3
    LATBbits.LATB10 = 1; // in4
}

void zaustaviSve() {

    //  OC1RS = 0;
    // OC2RS = 0;

    LATFbits.LATF2 = 0; // in1
    LATFbits.LATF3 = 0; // in2
    LATBbits.LATB11 = 0; // in3
    LATBbits.LATB10 = 0; // in4
}
int start_poruka = 0;
int stop_poruka = 0;

int main(int argc, char **argv) {
    initPins();
    ADCinit();
    initUART2(); // bluetooth
    initUART1(); // za serijsku i debugging
    Init_T1();
```

```
    Init_T4();
    Init_T5();
    initPWM();

    RS232_putst2("Write START");
    WriteUART2(13);

    while (1) {

        FR_back = fotootpornik(sirovi0); // promenjive za svaki fotootpornik
pokazuju da li je on osvjetljen
        FR_right = fotootpornik(sirovi1);
        FR_front = fotootpornik(sirovi2);
        FR_left = fotootpornik(sirovi3);

        if (startFlag == 1 && stopFlag == 0) {

            if (start_poruka == 0) {
                RS232_putst2("Tenk upaljen");
                WriteUART2(13);
                start_poruka = 1;
                stop_poruka = 0;
            }

           if (FR_back == 1 && stopFlag == 0) {

            MeasureBackDistance(); // Izmeri udaljenost unazad
             if (measured_distance_back > 13) {
              idiNazad(); // Ako je udaljenost unazad veca od 13, kreni nazad
                while (measured_distance_back >= 13 && stopFlag == 0) {
                    MeasureBackDistance(); // Ponovo izmeri udaljenost unazad
                      if (measured_distance_back < 13) {
         zaustaviSve(); // Zaustavi tenk ako je udaljenost unazad manja od 13
                      }
                }
              } else {
   zaustaviSve(); // Zaustavite tenk ako je udaljenost unazad vec manja od 13
              }
} else if (FR_right == 1 && stopFlag == 0) {

 idiDesno();
 delay_for(705000); // delay realizovan ovako da se tenk okrene za 90 stepeni
              MeasureForwardDistance(); // Izmeri udaljenost unapred
              if (measured_distance_forward >= 13) {
           idiNapred(); // Ako je udaljenost unapred veca od 13, kreni napred
                while (measured_distance_forward >= 13 && stopFlag == 0)
{
               MeasureForwardDistance(); // Ponovo izmeri udaljenost unapred
                      if (measured_distance_forward < 13) {
        zaustaviSve(); // Zaustavi tenk ako je udaljenost unapred manja od 13
                      }
                }
              } else {
   zaustaviSve(); // Zaustavi tenk ako je udaljenost unapred vec manja od 13
              }
           } else if (FR_front == 1 && stopFlag == 0) {

              MeasureForwardDistance(); // Izmeri udaljenost unapred
              if (measured_distance_forward > 13) {
           idiNapred(); // Ako je udaljenost unapred veca od 13,kreni napred
```

```
                    while (measured_distance_forward >= 13 && stopFlag == 0)
{
                MeasureForwardDistance(); // Ponovo izmeri udaljenost unapred
                            if (measured_distance_forward < 13) {
        zaustaviSve(); // Zaustavi tenk ako je udaljenost unapred manja od 13
                            }
                        }
                } else {
     zaustaviSve(); //Zaustavi tenk ako je udaljenost unapred vec manja od 13
                }
            } else if (FR_left == 1 && stopFlag == 0) {

                idiLevo();
 delay_for(705000); //delay realizovan ovako da se tenk okrene za 90 stepeni
                MeasureForwardDistance();
                if (measured_distance_forward >= 13) {
          idiNapred(); // Ako je udaljenost unapred veca od 13, kreni napred
                    while (measured_distance_forward >= 13 && stopFlag == 0)
{
                MeasureForwardDistance(); // Ponovo izmeri udaljenost unapred
                            if (measured_distance_forward < 13) {
        zaustaviSve(); // Zaustavi tenk ako je udaljenost unapred manja od 13
                            }
                        }
                } else {
    zaustaviSve(); // Zaustavi tenk ako je udaljenost unapred vec manja od 13
                }
            } else {
      zaustaviSve(); // Zaustavi tenk ako nijedan fotootpornik nije osvetljen
            }
        } else if (stopFlag == 1) {

            if (stop_poruka == 0) {
                RS232_putst2("Tenk ugasen");
                WriteUART2(13);
                stop_poruka = 1;
                start_poruka = 0;
            }
            zaustaviSve();
            stopFlag = 0;
            startFlag = 0;
        }

    } // od while

    return 0;
}
```

# 9.ZAKLJUČAK

Nakon završetka projekta, utvrdili smo da uređaj uspešno izvršava sve zadate funkcije koristeći potrebne komponente. Ipak, uočeni su i neki izazovi. Fotootpornici imaju ograničen domet detekcije, dok HC-SR04 senzori mogu biti osetljivi na ambijentalnu buku ili prepreke koje deformišu ultrazvučne talase, što može uzrokovati netačna merenja udaljenosti. Takođe, mogu biti podložni interferencijama kada se koriste u blizini drugih uređaja koji emituju ultrazvuk.

Poboljšanja bi mogla uključivati dodavanje dodatnih senzora, poput infracrvenih za precizniju detekciju prepreka, kao i implementaciju navigacionih algoritama koji bi omogućili složenije odluke pri kretanju robota, poput pronalaženja optimalnih puteva ili mapiranja okoline. Integracija kamere bi omogućila vizuelno prepoznavanje okoline, dok bi dodavanje više funkcija putem Bluetooth-a proširilo mogućnosti uređaja.

Tokom ovog projekta, naš tim je stekao značajno iskustvo u projektovanju, programiranju i testiranju robotskih sistema, kao i u radu sa senzorima i Bluetooth komunikacijom. Pored tehničkih veština, projekat je omogućio razvijanje sposobnosti rešavanja problema i timskog rada.

# 10.LITERATURA


[1] U. electronics, „HC-06 bluetooth modu,“ [Na mreži]. Available: https://www.utmel.com/components/hc-06-bluetooth-module-pinout-datasheet-pdf-and-arduino-connection?id=884. [Poslednji pristup 14. 09. 2024].

[2] „OSEPP Electronics LTD HC-SR04,“ [Na mreži]. Available: https://www.utmel.com/productdetail/oseppelectronicsltd-hcsr04-8415088. [Poslednji pristup 14. 09. 2024].

[3] „Data sheet za fotootpornik“ [Na mreži]. Available: https://eepower.com/resistor-guide/resistor-types/photo-resistor/. [Poslednji pristup 14. 09. 2024].

[4] „PICkit 2 Programmer/Debugger User's Guide,“ 14 09 2024. [Na mreži]. Available: https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/UserGuides/51553E.pdf.

[5] „Data sheet za Lm7805,“ [Na mreži]. Available: https://www.alldatasheet.com/datasheet-pdf/pdf/82833/FAIRCHILD/LM7805.html. [Poslednji pristup 19. 09. 2024].

[6] „Data Sheet mikrokontrolera dsPIC30f4013,“ [Na mreži]. Available: https://ww1.microchip.com/downloads/en/devicedoc/70138c.pdf. [Poslednji pristup 19. 09. 2024].

[7] „Data Sheet za L298N,“ [Na mreži]. Available: https://www.alldatasheet.com/datasheetpdf/pdf/22440/STMICROELECTRONICS/L298N.html;. [Poslednji pristup 19. 9. 2024].

[8] Rajs V. Praktikum za vežbe iz Primenjene elektronike. Novi Sad: Fakultet tehničkih nauka, 2020.

[9] M. Živanov, Uvod u elektroniku, Novi Sad: FTN, 2013.

[10] Materijal u vidu prezentacija iz predmeta Primenjena elektronika za školsku 2023/2024