UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
PRAKTIKUM AUTOMATIKE
ODSJEK ZA AUTOMATIKU I ELEKTRONIKU
AKADEMSKA 2018/2019. GODINA

Mapiranje prostora uz pomoć UV senzora i koračnog motora

- Izvještaj projektnog zadatka -

Članovi tima:

Ali Al Zayat Mennatulla, 17656

Muminović Samira, 17582

Neković Ajla, 17418

Sarajevo, februar 2019.

SAŽETAK

Tema projekta je bila kreiranje i realiziranje sistema za mapiranje prostora uz pomoć ultrazvučnog senzora, mikrokontrolera PIC16F1939, modula za serijsku komunikaciju, koračnog motora i računara. Senzor smo pričvrstili za koračni motor i pomoću njega očitavali udaljenost do prepreke. Vrijednost udaljenosti, u decimetrima, smo slali na računar pomoću modula za serijsku komunikaciju. Na grafičkom interfejsu, koji smo razvili u MATLAB-u (*skr. od eng. Matrix laboratory*), se prikazuje očitana udaljenost objekta od senzora i iscrtava se položaj objekta u koordinatnom sistemu. Kako bismo pokrili što veći prostor pri mapiranju, koristimo motor čijim upravljanjem rotiramo senzor.

ABSTRACT

Our team project was about the design and realization of an area mapping system that uses an ultraviolet sensor, a microcontroller PIC16F1939, a serial communication module, a step motor and a PC. We attached the sensor to the motor and used it to determine the distance between the sensor and objects around it. The distance value was then sent using the serial communication module to the PC. We used the graphical interface designed in MATLAB to receive the distance value and display the received value in decimeters and draw the object's location in a coordinate system. In order to be able to map the largest area possible we used the step motor to rotate the sensor.

Sadržaj

1. Uvod

1.1. Opis problema

Tema ovog projekta je 2D mapiranje prostora korištenjem motora i senzora udaljenosti. Konkretno, u ovom projektnom zadataku korišteni su step motor i ultrazvučni senzor udaljenosti. Realiziran je sistem koji omogućava očitavanje udaljenosti pomoću senzora postavljenog na motor, pri tome koristeći razvojni sistem PIC16F1939. Očitane vrijednosti su obrađene i pretvorene u korisnu informaciju te putem USB *(skr. od eng. Universal Serial Bus*) komunikacijskog modula poslane na računar. Na računaru podaci se ispisuju i predstavljaju u formi grafičkog interfejsa (*GUI, skr. od eng. Graphical User Interface*), razvijenom u programskom paketu MATLAB,.

Postavljeni zadatak se mogao dizajnirati i realizirati na mnogo načina te smo se morali odlučiti za funkcionalnosti koje će naš sistem podržavati. Iz ovog razloga smo precizirali na koji će način naš sistem funkcionisati. U skladu s tim smo se odlučili da na grafičkom interfejsu, koji smo razvili u MATLAB-u, omogućimo prikaz smjera vrtnje motora, prikaz vrijednosti udaljenosti za najnoviji detektovan i iscrtan objekat te prikaz položaja detektovanih objekata u koordinatnom sistemu tokom kretanja motora u jednu stranu. Pri promjeni smjera vrtnje motora prethodni prikaz mapiranog prostora se briše i prostor se iznova mapira počevši od iste početne pozicije kao i kod prvobitnog smjera vrtnje motora. Vrijednost udaljenosti se šalje i prikazuje na grafičkom interfejsu u decimetrima. Na interfejsu imamo i prikaz trenutnog smjera vrtnje motora. Detaljna analiza rada senzora, motora, serijske komunikacije i realizacije ovog projekta slijede u narednim poglavljima.

1.2. Mikrokontroler PIC16F1939

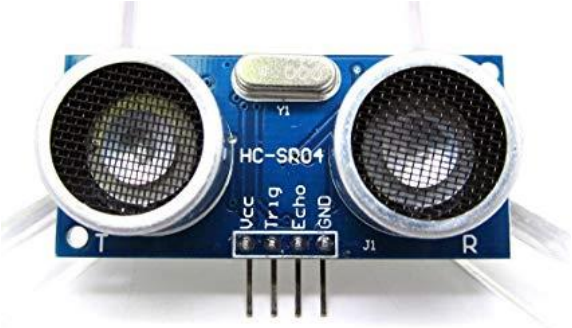
Mikrokontroleri (*skraćeno μC*) su integrisani krugovi koji imaju sve što im je potrebno da budu računari: memoriju, procesor, ulaze i izlaze. Mikrokontroleri se koriste u automatski upravljanim proizvodima i uređajima, kao što su kontrolni sistem motora automobila, medicinskim uređajima, uređajima na daljinsko upravljanje, kućanskim aparatima, igračkama i ostalim ugradbenim (*engl. embedded*) uređajima.[1] Na slici 1.1. prikazan je mikrokontroler PIC16F1939, koji je korišten za izradu projekta i u nastavku bit će detaljnije objašnjen.

Slika 1.1. PIC16F1939

Korišteni mikrokontroler je Microchip-ov mikrokontoler koji ima 40 pinova. Programsko okruženje MPLAB-X služilo je za programiranje pomenutog mikrokontrolera. Specifikacije potrebne za realizaciju zadatka su sljedeće: frekvencija internog clock-a koja iznosi do 32MHz, 4 porta (A, B, C, D) sa po 8 pinova te napon napajanja koji se kreće od 1.8V do 5.5V.[2]

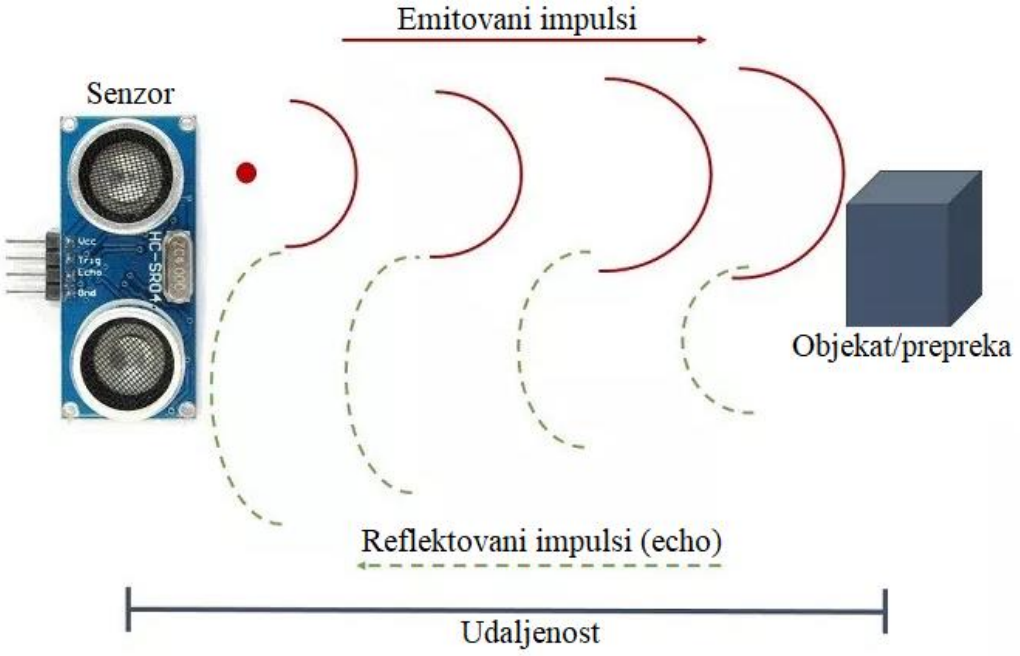
1.3. Ultrazvučni senzor udaljenosti HC-SR04

Kao što i samo ime govori, ultrazvučni senzori udaljenosti koriste se za mjerenje rastojanja između senzora i objekta koji se nalazi ispred njih. Ovi senzori nude dobru preciznost i jednostavni su za upotrebu. Za realizaciju projekta korišten je senzor HC-SR04, koji je prikazan na slici 1.2.



Slika 1.2. HC-SR04 senzor udaljenosti

Rezolucija senzora je 0.3 [cm], a opseg detektovanja objekta od 2 do 500 [cm]. Senzor pokriva horizontalni ugao do 15º. Sam koncept ove tehnologije je jednostavan. Senzor emituje zvučne impulse visoke frekvencije, koji se u slučaju da se ispred senzora nalazi prepreka, odbijaju od nje ka senzoru. Ako su impulsi detektovani nakon emitovanja, možemo pretpostaviti da se ispred senzora nalazi prepreka (*slika 1.3*).



Slika 1.3. Princip rada ultrazvučnog senzora udaljenosti

Poznata veličina je brzina kretanja zvuka (340,26 [m/s] ili 1236 [km/h]), a senzor nam daje informaciju o vremenu koje je potrebno da se emitovani zvučni talas odbije od prepreku i vrati nazad. Sa te dvije informacije možemo lako izračuati udaljenost objekta, pomoću relacije 1.1.

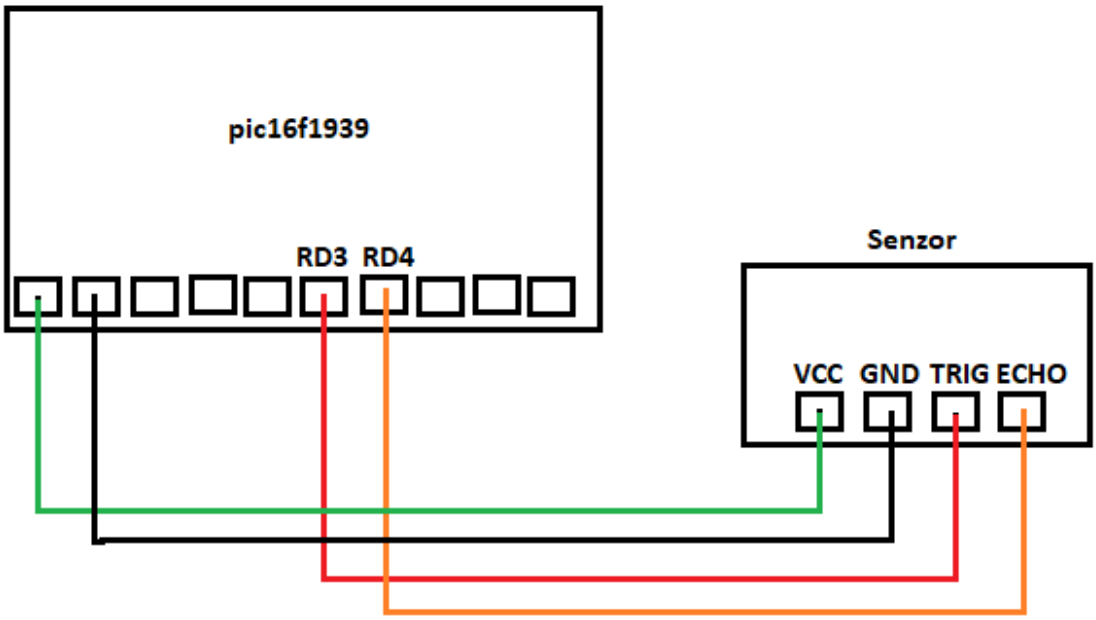
$$d = v_{zvuka} \cdot t \tag{1.1}$$

Dva osnovna dijela modula na kojima se temelji princip rada su trig (*prekidač, skr. od eng. trigger*) i echo (*refleksija*). Mikrokontrolerom se šalje 5 [V] na trig pin modula u trajanju, minimalno, 10 mikrosekundi. Na taj način aktivira se ultrazvučni transduktor koji odašilje 8 impulsa od 40 [kHz] i čeka njihovu refleksiju. Kada senzor registruje reflektovani impuls, šalje podatke nazad mikrokontroleru preko echo pina. Navedeni podatak koji se dobija iz senzora je zapravo vrijeme trajanja reflektiranog pulsa, od 150 [µs] do 25 [ms]. Ako impuls traje duže od 35 [ms], senzor takav podatak registruje kao da je predmet izvan dosega.

Modul ime sljedeće pinove:

- Pin 1: VCC (*napajanje, skr. od eng. Voltage Common Collector*) – napajanje modula, 5V
- Pin 2: Trig – okidanje/aktiviranje mjerenja
- Pin 3: Echo – povratni signal
- Pin 4: GND (*masa, skr. od eng. ground*)

Shema spajanja senzora sa razvojnim sistemom prikazana je na sljedećoj slici (*slika 1.4*).



Slika 1.4. Shema spajanja senozora sa razvojim sistemom

Za računanje udaljenosti neophodno je izmjeriti dužinu trajanja povratnog impulsa, i uvrstiti u sljedeću relaciju (*relacija 1.2*), pri tome vodeći računa da je dobijena vrijednost u decimetrima (razlog uzimanja ove mjere bit će objašnjen u jednom od narednih poglavlja). Potrebno je proizvod vremena i brzine podijeliti sa 2, jer senzor mjeri vrijeme zvučnog signala na putu do prepreke i od prepreke nazad.

$$d = \frac{t \cdot 3400}{2}\ [dm] \tag{1.2}$$

1.4. Serijska komunikacija

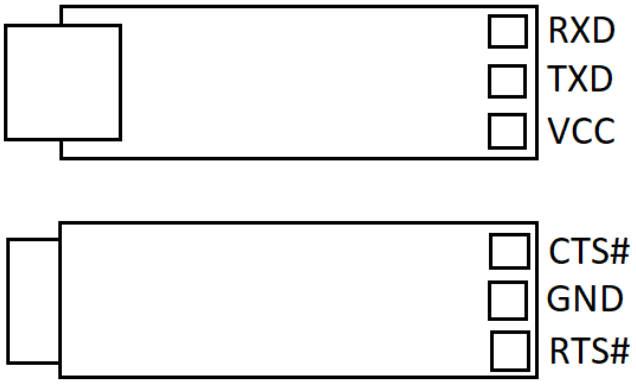
Serijska komunikacija podrazmijeva proces slanja podataka preko komunikacijskog kanala bit po bit (sekvencijalno) i koristi se uglavnom u ugradbenim sistemima. Njena prednost je korištenje malog broja linija (minimalno 2 ili 3), a ključni nedostatak je relativno mala brzina prenosa. Ova vrsta komunikacije predstavlja osnovni i najrašireniji vid komunikacije korišten u aplikacijama sa mikrokontrolerima te većina mikrokontrolera koristi hardver za serijsku komunikaciju, uključujući i mikrokontroler korišten u ovom projektu. U slučaju da u okviru mikrokontrolera ne postoji serijski port, serijsku komunikaciju je moguće realizirati programski.

Postoji više standarda za serijsku komunikaciju, za različite namjene. Među najraširenije standarde za serijsku komunikaciju između ostalih, pogotovo ako se u vidu ima komunikacija između različitih sistema (PC, periferijski uređaji, embedded sistemi, instrumentacija), spada i RS-232 (EIA/TIA-232-E, ITU-T V.24) komunikacija sa perifernim uređajima. [3]

Za serijsku komuminkaciju između računara i senzora korsitili smo TTL-232-3V3-PCB modul prikazan na slici 1.5.

Slika 1.5. Izgled modula

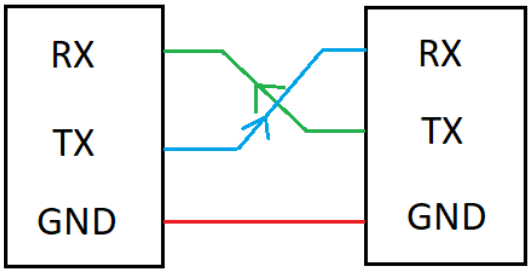
Na slici 1.6. prikazan je raspored pinova korištenog modula.



Slika 1.6. Pinovi modula

Pin RX (*prijemnik, skr. od eng. receiver*) sa modula spaja se na pin TX (*predajnik, skr. od eng. transmitter*) na mikrokontroleru, i obrnuto. Pin GND je spojen sa masom (GND pinom) mikrokontrolera. Način spajanja modula sa mikrokontrolerom prikazan je na slici 1.7. Pin na kojem je VCC nismo koristili jer se modul napaja sa računara preko USB porta.

Pinovi CTS# (*skr. od eng. Clear To Send*) i RTS# (*skr. od eng. Ready To Send*) nisu korišteni u projektu, a inače imaju svrhu da se preko njih upravlja kontrolom toka slanja i prijema podataka, tj. koriste se kao kontrolni biti. Kada se uključi modul u neki od USB portova računara, potrebno je instalirati drajvere za rad modula. Nakon toga u Device Manager-u se može vidjeti na kojem portu (COM) je spojen modul.

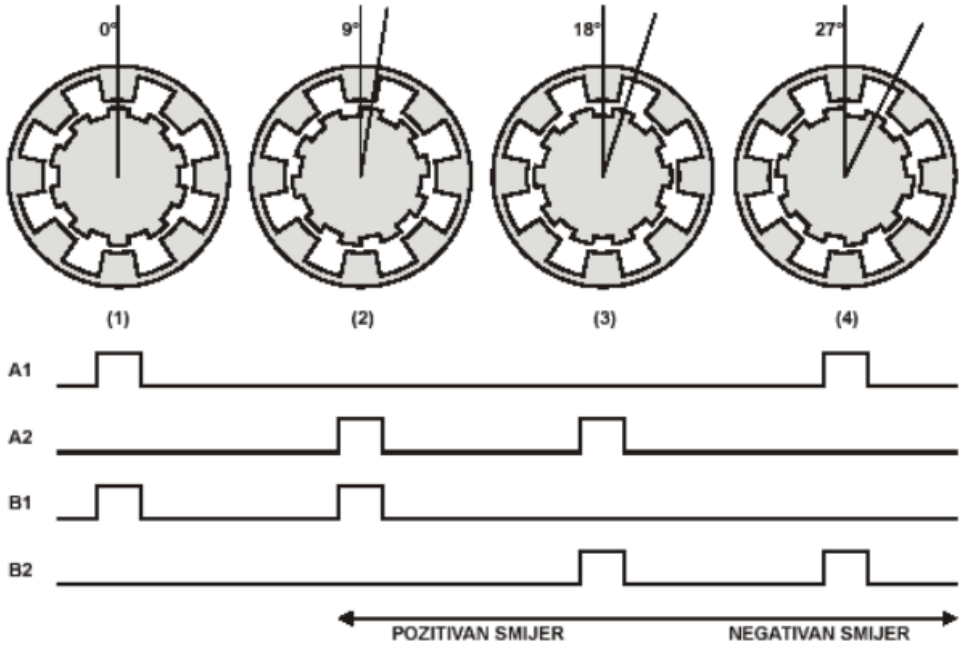


Slika 1.7. Način povezivanja modula sa mikrokontrolerom

1.5. Step motor

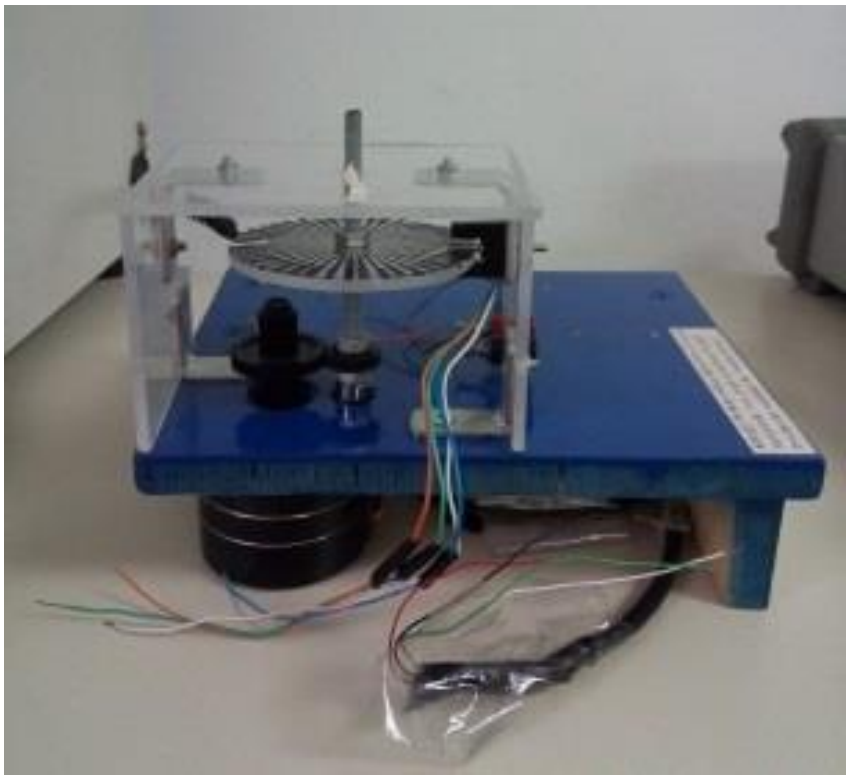
Step motor je vrsta elektromotora bez četkica koji pretvara digitalne impulse struje u fiksne inkremente ugaonog pomjeranja nazvane koraci. Ova vrsta motora obezbjeđuje precizno pozicioniranje tereta, a kontrola motora se vrši direktno računarom, mikrokontrolerom ili programibilnim logičkim kontrolerom. Zbog svoje konstrukcije bez četkica, koračni motori su pouzdani, izdržljivi, i ne zahtijevaju nikakvo održavanje.[4]

Na slici 1.8. prikazani su impulsi za upravljanje step motorom.



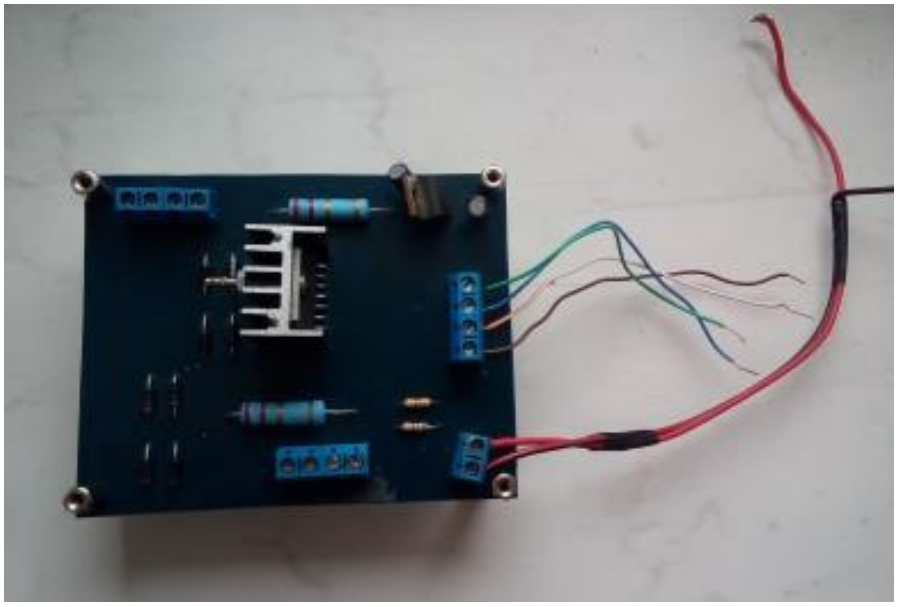
Slika 1.8. Impulsi za upravljanje step motorom[5]

Motor koji smo mi koristi u ovom projektu je dio jednog sistema koji se sastoji od motora i enkodera, napravljenog na Elektrotehničkom fakultetu u Sarajevu (*u nastavku teksta ETF*). Sistem je prikazan na slici 1.9.



Slika 1.9. Sistem sa step motorom

Osim toga, korišten je i H-most L298N koji se nalazi na pločici također napravljenoj na ETF-u. Pločica na kojoj se nalazi modul L298N i njegovi izvedeni pinovi prikazana je na slici 1.10.



Slika 1.10. H-most za upravljanje step motorom

Izgled samog modula predstavljen je na slici 1.11.

Slika 1.11. Modul L298N

1.6. Serijski asinhroni port na PIC16F1939

(E)USART (*skr. od eng. Enanced Universal Synchronous Asynchronous Receiver Transmiter*) na PIC16F1939 omogućava sinhronu komunikaciju, a namjena je komunikacija sa EEPROM (*električno izbrisiva programibilna memorija, skr. od eng. Electricaly Erasable Programmable Read-Only Memory*) i drugim vanjskim komponentama. Također omogućava i asinhronu komunikaciju sa i bez korištenja adrese, a namjena je komunikacija sa drugim perifernim uređajima i računarima. Ovdje koristimo asinhronu razmjenu 8-bitnih podataka, pri čemu će adresiranje biti realizirano softverski.

Rad ovog modula je kontrolisan sa tri registra:

- Transmit Status and Control (*skr. TXSTA*)
- Receive Rate Control (*skr. RCSTA*)
- Baud Rate Control (*skr. BAUDCON*)

Kako bi slanje podataka bilo omogućeno potrebno je dodati sljedeće naredbe:

- ✓ TXEN = 1 (*skr. od eng. Transmit Enable bit*) - TXEN bit registra TXSTA omogućava slanje putem serijske komunikacije
- ✓ SYNC = 0 (*eng. USART Mode Select bit*) - asinhrona komunikacija se izabere tako što se SYNC bit postavi na nulu
- ✓ SPEN = 1 (*skr. od eng. Serial Port Enable bit*) - SPEN bit potrebno je postaviti na jedinicu, tako da se TX pin mikrokontrolera automatski postavi kao izlazni; kako je pin TX ujedno i pin RC6, potrebno je onemogućiti analogne signale na tom portu, što se postiže resetovanjem ANSELC registra

Slanje podataka vrši se upisivanjem podatka u TXREG, odakle se on šalje u TSR registar (*skr. od eng. Transmit Shift Register*), jer se TSR registru ne može programski pristupiti. Kako je u ovom projektu bilo potrebno samo slanje podataka, za primanje podataka bit će samo ukratko navedno šta je u kodu potrebno postaviti: CREN = 1 (*skr. od eng. Continuous Receive Enable bit*), SYNC = 0 i SPEN = 1.

2. Realizacija projekta

2.1. Svođenje na formu algoritma

Kako bismo napravili željenu aplikaciju bilo je potrebno napisati kod za mikrokontroler koji upravlja kretanjem motora, aktiviranjem senzora, obradom signala koji dobivamo kao povratnu informaciju sa senzora, aktiviranjem i uspostavljanjem komunikacije između računara i mikrokontrolera te slanjem informacije o udaljenosti objekta putem uspostavljene komunikacije.

Sa druge strane bilo je neophodno razviti grafički interfejs – GUI u MATLAB-u koji prima podatke preko USB komunikacijskog modula i detektuje pravilan redoslijed potrebnih podataka. Na osnovu tih podataka primljene vrijednosti konvertujemo u udaljenost koja je poslana od strane mikrokontrolera. Osim obrade primljenih podataka u MATLAB kodu smo morali na osnovu udaljenosti i broja koraka koje je motor načinio odrediti koordinate zadnjeg detektovanog objekta i prikazati njegov položaj u koordinatnom sistemu, na kojem koordinatni početak odgovara položaju senzora.

Prvi korak u implementiranju bilo kakvog algoritma na naš sistem bio je identifikacija ulaza i izlaza pri radu sa mikrokontrolerom. Tako je triger senzora identifikovan kao digitalni izlaz, dok je echo digitalni ulaz. Treći pin senzora koji predstavlja njegovo napajanje je direktno spojen na napajanje. Četiri pina kojima upravljamo motorom smo identifikovali kao digitalne izlaze sa mikrokontrolera. USB komunikacijski modul je spojen na odgovarajući način tako da je TX pin na komunikacijskom modulu spojen sa RX pinom mikrokontrolera, dok je RX pin na komunikacijskom modulu spojen sa TX pinom mikrokontrolera. Pored porta D na mikrokontroleru, koji smo koristili za upravljanje svim navedenim digitalnim ulazima i izlazima te pored pinova preko kojih smo uspostavili komunikaciju, koristili smo još jedan od resursa mikrokontrolera s kojim smo radili a to je 16-bitni timer.

2.2. Korišteni alat

Programski alati koji su korišteni u cilju realizacije projektnog zadatka su sljedeći:

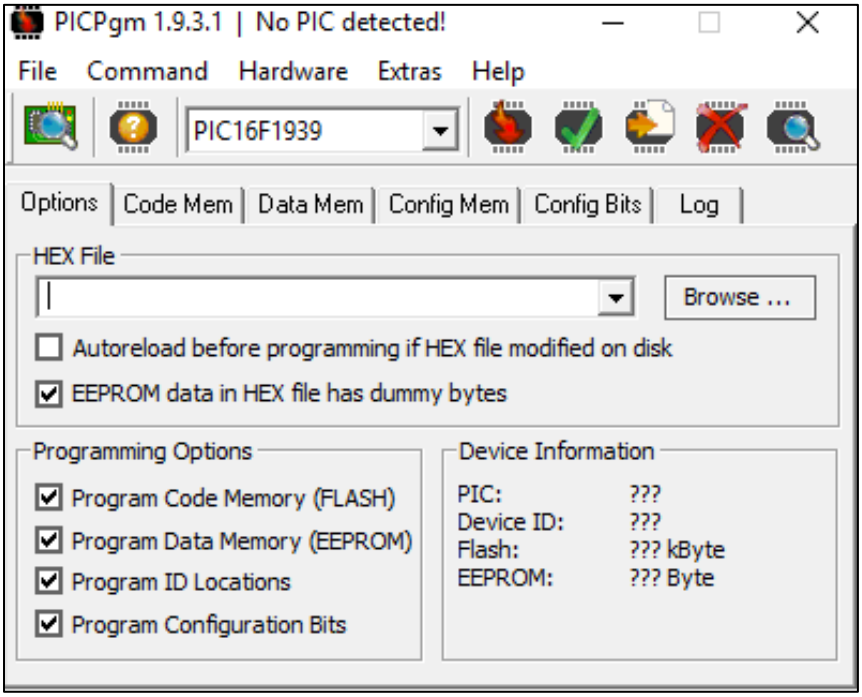
- ✓ MPLAB-X
- ✓ PICPgm
- ✓ MATLAB

2.2.1. MPLAB-X

MPLAB je besplatno razvojno okruženje, kojeg je razvila kompanija Microchip-a. MPLAB-X je najnovija verzija MPLAB-a, koja je namijenjena za realizaciju aplikacija koje se koriste u ugradbenim sistemima, a podržana je od strane Windowsa, Linuxa i macOS operativnih sistema. Kod napisan u ovom razvojnom okruženju prevodimo u mašinski kod koji se implementira na samom mikrokontoleru. U svrhu realizacije projektnog zadatka koristilli smo XC8 kompajler za C programski jezik. Unutar ovog razvojnog okruženja realizovali smo kod za upis u mikrokontroler kojim smo upravljali većinskim dijelom projekta.

2.2.2. PICPgm

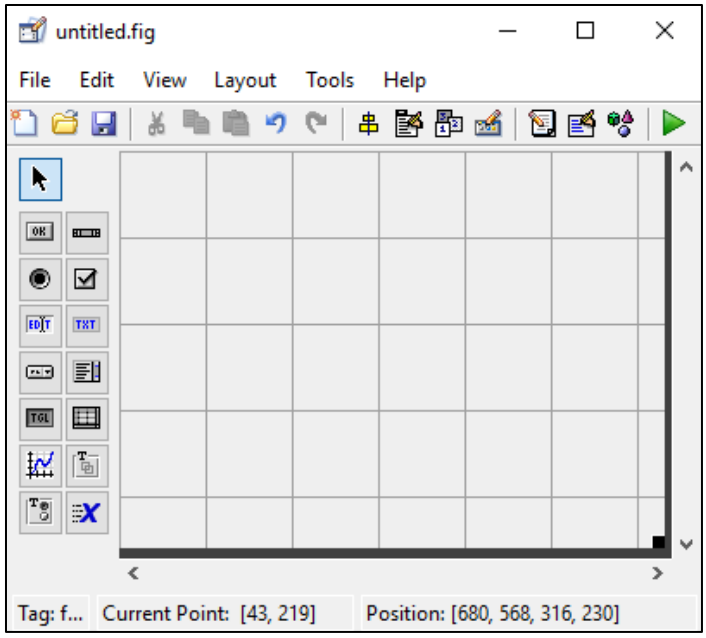
PICPgm je besplatan program koji služi za programiranje Microchip-ovih mikrokontolera. Uz pomoć ovog programa vršimo brisanje i prebacivanje koda sa PC-a na mikrokontroler posredstvom serijskog COM porta. Prije nego što pokrenemo očitavanje, koristeći GUI, potrebno je da se PICPgm zatvori. Izgled ovog programa dat je na slici 2.1.



Slika 2.1. Primjer izgleda PICPgm programa
(na PC nije spojen mikrokontroler)

2.2.3. MATLAB

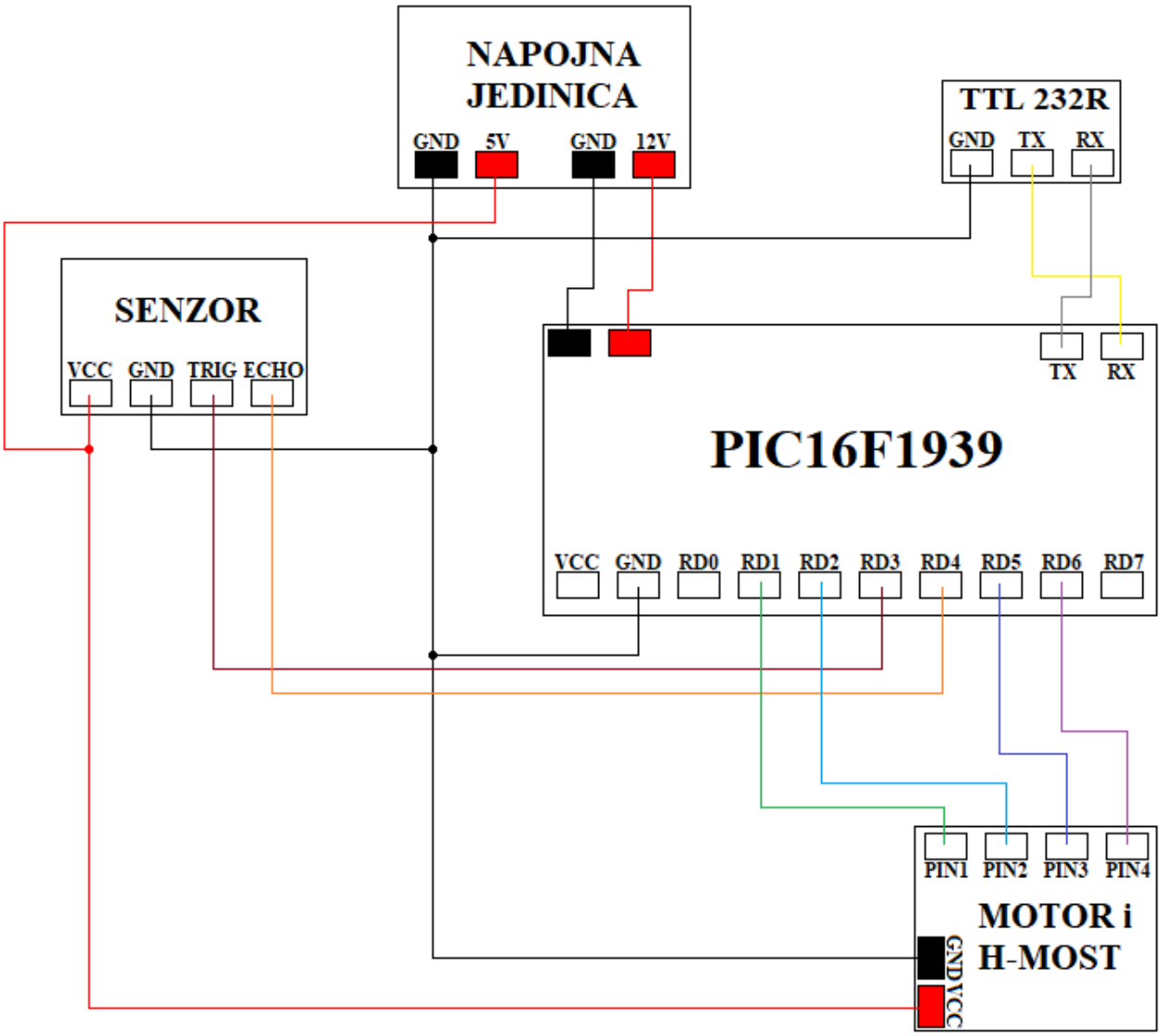
Matlab predstavlja vrlo razvijen skup alata za računanje, vizualiziranje, modeliranje, simulaciju i programiranje. MATLAB je programski jezik visoke razine pomoću kojeg se mogu analizirati podaci, izraditi algoritmi te stvoriti modeli i aplikacije. Iako se interakcija korisnika sa Matlabom odvija preko komandnog prompta, Matlab omogućava i gradnju kompleksnog grafičkog interfejsa. Ono što je zapravo korišteno pri realizaciji ovog projekta jeste kreiranje grafičkog interfejsa tj. GUI-a pisanjem odgovarajućeg koda (koji će u nekom od narednih poglavlja biti prikazan). Izgled prozora za razvoj grafičkog interfejsa prikazan je na slici 2.2.



Slika 2.2. Osnovni prozor alata guide za razvoj GUI-a

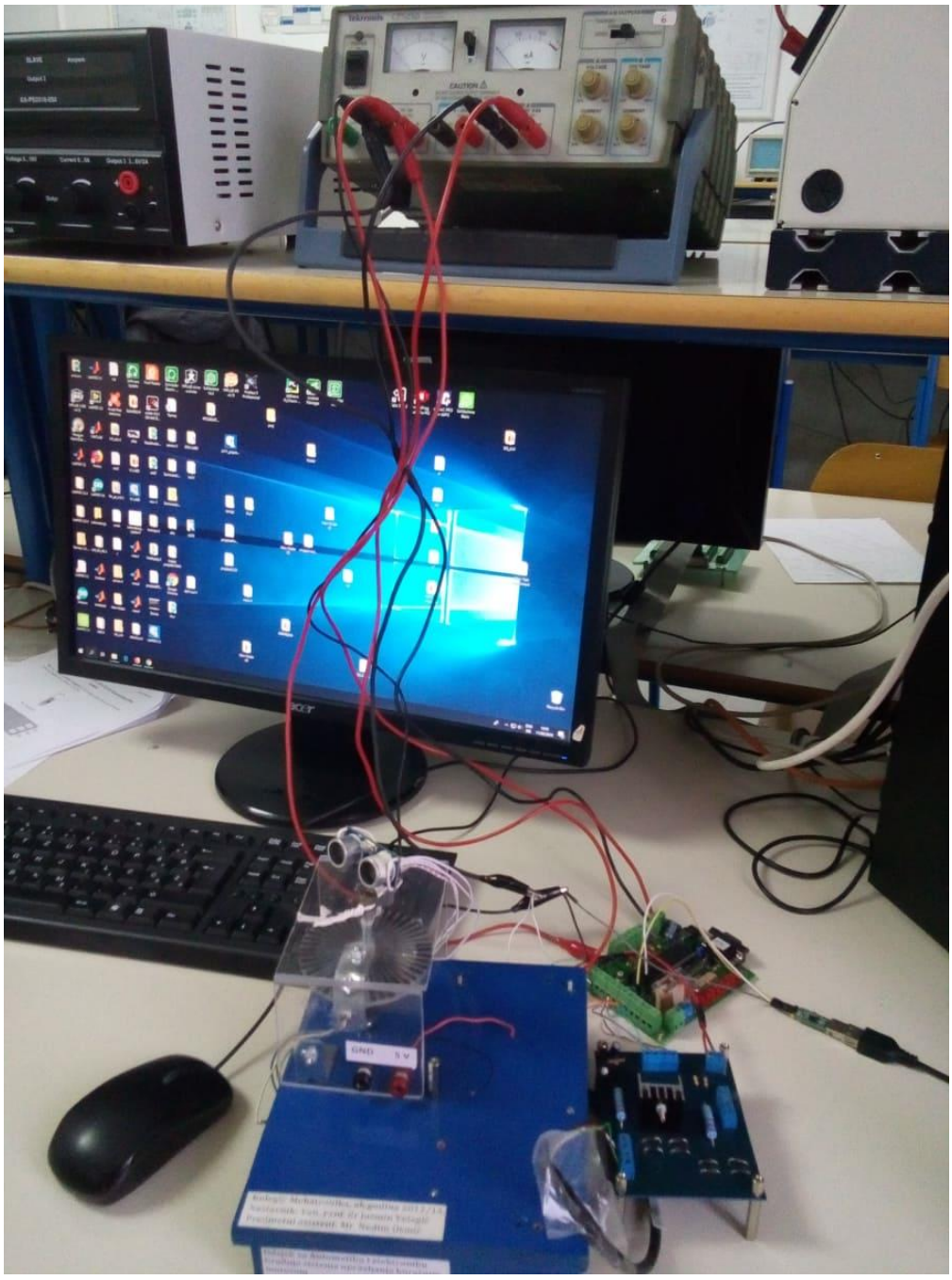
2.3. Shema spajanja

Na slici 2.3. prikazana je shema spajanja komponenti na ovom projektu. Ultrazvučni senzor spojen je na sljedeći način: triger – RD3, echo – RD4. Modul za serijsku komunikaciju je spojen na način koji je već predstavljen u izvještaju, a to je RX spojen na TX, a pin TX spojen na RX. Važno je naglasiti da za modul TTL nije bilo potrebno napajanje, jer se on napajanja preko računara već smo samo masu spajali sa masom razvojnog sistema. Motorom smo upravljali pomoću H-mosta koji je bio spojen na pinove RD1, RD2, RD5 i RD6 na mikrokontroleru PIC16F1939. Trebala su nam i dva izvora napona kako nam je bilo potrebno napajanje od 5 [V] i od 12 [V]. Izvor 5 [V] bio je zajednički za senzor i H-most, dok nam je izvor od 12 [V] bio potreban kako bi napajali mikrokontroler. Na pomenutoj slici sheme svaka žica je druge boje kako bi se bolje razlikovale, ali su crna i crvena rezervisane (redom) za masu i napajanje. Na mikrokontroleru smo označili pored porta D koji smo koristili, još pinove koji su bili u upotrebi.



Slika 2.1. Šematski prikaz spajanja komponenti

Na slici 2.2. prikazana je shema cjelokupnog sistema, spojena u laboratoriji.



Slika 2.2. Sistem spojen u laboratoriji

3. Programska implementacija

3.1. Implementacija u MPLAB-X

Kako smo u dosadašnjem tekstu naveli kod smo pisali u razvojnom okruženju MPLAB-X. Kod koji smo upisali u mikrokontroler i koristili prilažemo u tabeli 3.1.

```
#include <xc.h>
#pragma config FOSC=HS,WDTE=OFF,PWRTE=OFF,MCLRE=ON,CP=OFF,
CPD=OFF,BOREN=OFF,CLKOUTEN=OFF
#pragma config IESO=OFF,FCMEN=OFF,WRT=OFF,VCAPEN=OFF,PLLEN=OFF,
STVREN=OFF,LVP=OFF

#define _XTAL_FREQ 8000000

float udaljenost(void){
    TMR1ON = 1;
    TMR1L=TMR1H=0;
    while(!(PORTDbits.RD4));
    TMR1ON=0;
    TMR1L=TMR1H=0;
    TMR1ON = 1;
    while(PORTDbits.RD4);
    //TMR1H<<8 siftan registar za 8 mjesta
    float d = (TMR1L |(TMR1H<<8)) /588.23 ;
    if(d>50)
        d=50;
    return d;
}

void init_port(){
    ANSELD=0x00;
    TRISD=0x10;

    TRISC = 0x00;
    LATC=0x00;
    return;
}

void init_serial()
{
     //BRGH=1;  // High Baud Rate
     //SPBRG=51; // Za 9600 na Fosc=8MHz
     SYNC=0;   // Asinhrona komunikacija
    BRG16=0;
    BRGH=0;
```

```
    SPBRGL=12;
     SPEN=1;   // Aktiviranje serijskog porta
     RCIE=0; // Onemogucavanje interrupta na serijskom portu
     CREN=0; // Ukljucivanje/iskljucivanje prijemnika
    return;
}

void main(void) {
    init_port();
    init_serial();
    //inicijalizacija modula tajmera 1
    T1CON = 0x10;
    TXEN = 1;

    while(1){
        PORTDbits.RD3=1;
        __delay_us(10);
        PORTDbits.RD3=0;

        float d = udaljenost();
    //slanje izracunate udaljenosti
        int brojac=0;
        while(brojac!=64){
         PORTDbits.RD3=1;
        __delay_us(10);
        PORTDbits.RD3=0;

        float d = udaljenost();
            PORTDbits.RD1=1;
            __delay_ms(150);
            PORTDbits.RD1=0;
            __delay_ms(150);
            PORTDbits.RD5=1;
            __delay_ms(150);
            PORTDbits.RD5=0;
            __delay_ms(150);
            PORTDbits.RD2=1;
            __delay_ms(150);
            PORTDbits.RD2=0;
            __delay_ms(150);
            PORTDbits.RD6=1;
            __delay_ms(200);
            PORTDbits.RD6=0;
            __delay_ms(150);
            brojac=brojac+4;
            TXREG = (unsigned char) d;
```

```
            while(!TXIF);
            TXREG = (d - (unsigned char) d)*10;
            while(!TXIF);
            TXREG = 'A';
            while(!TXIF);
        }
        TXREG = 'B';
        while(!TXIF);

        while(brojac!=0){
            PORTDbits.RD3=1;
        __delay_us(10);
        PORTDbits.RD3=0;

        float d = udaljenost();

            PORTDbits.RD6=1;
            __delay_ms(150);
            PORTDbits.RD6=0;
            __delay_ms(150);
            PORTDbits.RD2=1;
            __delay_ms(150);
            PORTDbits.RD2=0;
            __delay_ms(150);
            PORTDbits.RD5=1;
            __delay_ms(150);
            PORTDbits.RD5=0;
            __delay_ms(150);
            PORTDbits.RD1=1;
            __delay_ms(150);
            PORTDbits.RD1=0;
            __delay_ms(150);
            brojac=brojac-4;
            TXREG = (unsigned char) d;
            while(!TXIF);
            TXREG = (d - (unsigned char) d)*10;
            while(!TXIF);
            TXREG = 'A';
            while(!TXIF);
        }
        TXREG = 'B';
        while(!TXIF);

    }
    return;
}
```

Tabela 3.1. Korišteni kod

U programu smo prvo izvršili inicijalizaciju portova. Port C je izlazni, a od korištenih 6 pinova porta D njih 5 je izlaznih, a samo jedan ulazni. U funkciji *init_serial()* vršimo konfiguraciju koja omogućava korištenje komunikacijskog modula i postavljamo brzinu prenosa na 9600 Bps (*bita po sekundi, skr. od. eng. bits per second*). U mainu prije pozivanja funkcije udaljenost vršimo trigerovanje senzora, nakon čega pozivamo funkciju udaljenost u kojoj brojimo vrijeme potrebno da se emitovana zraka vrati na senzor. Ovo određivanje vremena vršimo na osnovu signala koji smo dobili sa pina ECHO, a koji je spojen na pin RD4. Za brojanje vremena koristimo modul Timer 1. Timer1 predstavlja 16-bitni brojač čija se vrijednost čuva u registrima TMR1H i TMR1L. Obzirom da znamo da je $c = 340\ m/s$ i $d = c*t/2$, to je udaljenost senzora od prepreke data relacijom 3.1.

$$d = \frac{t}{58{,}2751}\ [cm] \tag{3.1}$$

Vrijeme t u navedenoj relaciji predstavlja vrijeme koje ćemo brojati brojačem Timer1. Pošto je Timer1 modul 16-bitni te se podaci čuvaju u dva registra, potrebno je pri računanju pravilno prevesti vrijednosti iz ovih registara u odgovarajući cijeli broj, a to postižemo uzimanjem vrijednosti iz high registra, šiftanjem ove vrijednosti za 8 mjesta, a zatim izvršavanjem logičke operacije OR nad ovom vrijednosti i vrijednosti iz low registra. Obzirom da pri slanju podataka posredstvom komunikacijskog modula u jednom trenutku ne možemo slati podatak veći od jednog bajta, a sa druge strane senzor može detektovati udaljenosti do 500 [cm], odlučili smo da vrijednost udaljenosti računamo i šaljemo u decimetrima sa zaokruživanjem na jednu decimalu. Pa tako u program za računanje udaljenosti koristimo relaciju 3.2 koja daje udaljenost u decimetrima.

$$d = \frac{t}{588{,}23}\ [dm] \tag{3.2}$$

U main funkciji nakon izvršene inicijalizacije postavljamo TXEN na 1, čime omogućavamo slanje preko serijske komunikacije. Unutar while petlje vršimo trigerovanje senzora, očitavanje udaljenosti, pokretanje motora i slanje cijelog i decimalnog dijela vrijednosti udaljenosti preko komunikacijskog modula. Kako bismo u MATLAB-u mogli detektovati koji je prvi podatak pri primanju ovih vrijednosti, šaljemo i kontrolni karakter ‘A’ koji odgovara brojnoj vrijednosti 65. Nakon što izvršimo određeni broj koraka motora u jednu stranu, šaljemo kontrolni karakter ‘B’, kako bismo u MATLAB-u mogli detektovati promjenu smjera vrtnje motora, a u C kodu vršimo isti postupak čitanja i slanja podataka uz promjenu smjera vrtnje motora.

Vrtnjom motora upravljamo pomoću modula sa dva H-mosta te pulsanjem 4 izvoda u redoslijedu od 1-2-3-4-1-2-... postižemo vrtnju motora u jednom smjeru, dok pulsanjem u suprotnom redoslijedu 4-3-2-1-4-3-... postižemo kretanje motora u suprotnom smjeru.

3.2 Implementacija GUI-a

Za drugi dio zadatka iskoristili smo alat koji nam pruža MATLAB, a to je razvoj kompleksnog grafičkog interfejsa (guide) koji smo mi koristili kako bi prikazali mapu udaljenosti prepreka u koordinatnom sistemu. Nakon kreiranja željene forme i njenih dijelova potrebno je napisati kod. Korišteni kod, koji je bilo potrebno implementirati u dugme start da bi forma radila kako želimo, predstavljen je u tabeli 3.2.

```
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

global s1
delete(instrfind)
%s1=serial('/dev/ttyS0','BaudRate',9600);
s1=serial('COM7','BaudRate',9600);
s1.InputBufferSize=1;
s1.Terminator='';
s1.Timeout=1;
fopen(s1);
ugao = 0;
x0=0;
y0=0;
smjer = 1;
s='p';
axis(handles.axes1, [-50 50 -50 50])
hold on;
while (1)
    d=0;
    a = fread(s1,1)
        if (a==65)
            a = fread(s1,1)
            if (a==66)
                smjer = smjer *(-1);
                if(smjer==-1)
                    s='n';
                else
                    s='p';
                end
                %clf(handles.axes1);

                hold off;
                a = fread(s1,1)
            end
```

```
            a = fread(s1,1)
            d=a;
            a = fread(s1,1)
            d=d+a/10.0;
            if (d>50)
                d=50;
            elseif(isempty(d))
                continue;
            end
            set(handles.edit2,'String', s);
            x=x0+d*cos(ugao);
            y=y0+d*sin(ugao);
            if (x<50 || y<50)
                viscircles(handles.axes1, [x
y],0.3,'Color','b');
                %plot(handles.axes1, x, y, 'LineWidth', 25)
                set(handles.edit1,'String', d)
            else
                set(handles.edit1,'String', 50)
                viscircles(handles.axes1, [x
y],0.3,'Color','b');
                %plot(handles.axes1, 50, 50, 'LineWidth', 25)
            end
            axis(handles.axes1, [-50 50 -50 50])
            hold on;

            pause(0.1)
            d=0;
            if (ugao>=360)
                ugao=0;
            else
                ugao=ugao+12;
            end
        end
end
```

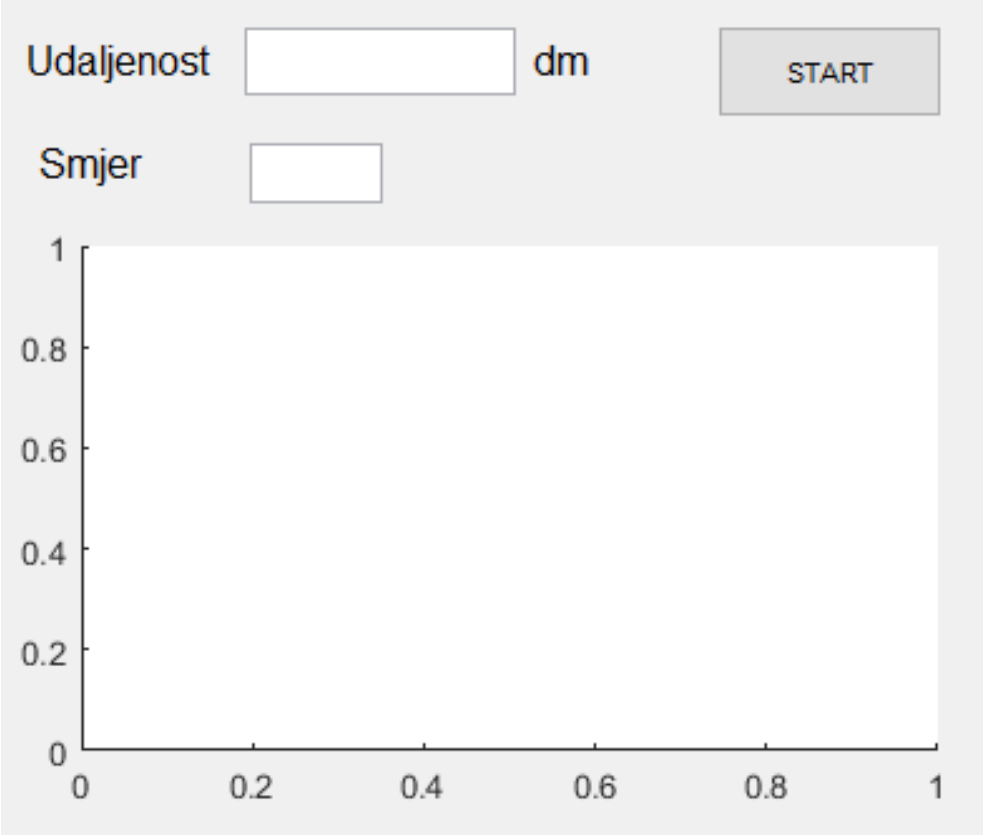
Tabela 3.2. Korišteni kod

Predstavljeni kod omogućava funkcionalnost dugmeta START. Na početku kreiramo serijski objekat s1 i izaberemo port (COM) preko kojeg je objekat povezan sa računarom. Odmah nakon kreiranja serijskog porta deklarišu se neke osnove vezane za njega kao što su: dužina buffera (broj bita koji će se očitavati), timeout (vrijeme u sekundama koje se čeka dok se izvrši operacija čitanja) i terminator (karakter kojim se terminira čitanje). Funkcijom *fopen()* omogućavamo komunikaciju između uređaja i PC-a. Varijable ugao, x0 i y0 su nam bile potrebne za iscrtavanje, tačnije za orijentaciju na koordinatnom sistemu (mapi).

Smjer je označavao smjer u kojem se kreće motor. Kako zbog žica za spajanje motor nije mogao napraviti puni krug, kretanje u jednom smjeru označili smo sa 1 i 'p' dok su za suprotni smjer korištene oznake -1 i 'n'. Kada koračni motor napravi određeni broj koraka i promijeni smjer kretanja, tzv. mapa se očisti i crtanje kreće ispočetka.

Funkcija *fread()* služi za čitanje sadržaja koji pristiže. Kako je već prethodno navedeno u izvještaju karakter 'A' (vrijednost 65) je označavao da sljedeća dva očitanja predstavljaju novi podatak, tj. udaljenost prepreke. Karakter 'B' (vrijednost 66) je bio znak za promjenu smjera. Iscrtavanje se vrši funkcijom *viscircles()* u decimetrima, vrijednosti preko 50 dm se tako označavaju jer je opseg senzora mjerenje udaljenosti do 500 cm. Za upis očitane vrijednosti u predviđeno mjesto na formi (tekstualni edit) koristi se funkcija *set()*. Čitanje se vrši veoma brzo pa je zbog toga neophodno postaviti pauzu (u našem slučaju 0.1s) koja će zadržati određeni podatak dok se ne upiše.

Nakon jednog pritiska na dugme START slijedi konstantno čitanje podataka i crtanje istih na mapi. Ponovno pokretanje očitanja moguće je tek nakon što se odabrani COM oslobodi, tj. unutar iste sesije nije moguće jer je odabrani COM već zauzet kreiranim serijskim objektom. Izgled forme prikazan je na slici 2.1.



Slika 2.1. Izgled gui-a

4. Zaključak

Projekat je predstavljao kreiranje aplikacije za mapiranje prostora koristeći step motor i ultrazvučni senzor. Motor smo upravljali kodom upisanim na razvojni sistem (opisan u poglavlju 3.1), a očitavanje udaljenosti i iscrtavanje vršili smo putem MATLAB-a (poglavlje 3.2). Za komunikaciju između računara i razvojnog sistema sa PIC16F1939 korišten je modul ttl 232r 3v3. Radi lakše izrade projektni zadatak smo podijelili u više dijelova: upravljanje motorom, uspostavljanje komunikacije, rad senzora, iscrtavanje u gui-u. Kada smo sve te dijelove uspješno obavili, posvetili smo se sastavljanju jedne cjeline. Prilikom toga nailazili smo na određene probleme, koje smo na kraju uspješno riješili.

Prvi od problema bio je vezan za step motor, za koji nismo imali datasheet, a odnosio se na pulsiranje izvoda motora kako bi se isti kretao ispravno. Uz pomoć asistenta, nakon određenog vremena, otklonili smo taj problem. Još jedan od problema bio je prvobitni modul za rs-232 komunikaciju, za koji smo nakon provjere ustanovili nepravilnosti te ga zamijenili modulom ttl 232r 3v3. Za ovaj modul bio nam je potreban drajver kojeg smo instalirali i nakon toga je komunikacija uspješno uspostavljena. Također, prilikom spajanja dijelova naišli smo na problem da zbog žica, motor ne može napraviti puni krug pa smo to riješili tako da se šalje karakter koji mijenja smjer kako bi motor pravio pola kruga.

Ovaj projekat može se iskoristiti za mapiranje prostora u opsegu udaljenosti od 2 cm do 500 cm, zbog karakteristika senzora. Pored svih problema na koje smo naišli, projekat je uspješno završen.

Literatura

[1] https://bs.wikipedia.org/wiki/Mikrokontroler

[2] Datasheet PIC16F1939 http://ww1.microchip.com/downloads/en/DeviceDoc/40001574C.pdf

[3] Predavanja iz predmeta Praktikum automatike, Predavanje 10, Samim Konjicija

[4] https://en.wikipedia.org/wiki/Stepper_motor

[5] Laboratorijska vježba 5, Aktuatori, Jasmin Velagić

[6] Praktikum automatike i informatike, Elektrotehnički fakultet u Sarajevu. Samim Konjicija, Sarajevo januar 2007.