Univerzitet u Sarajevu
Elektrotehnički fakultet u Sarajevu
Odsjek za automatiku i elektroniku

# Razvoj softverskih modula za mjerenje, nadzor i upravljanje ultrazvučnom kamerom

**Završni rad**
I ciklusa studija

**Mentor:** **Kandidat:**

Red.prof.dr Melita Ahić-Đokić Kenan Softić

Sarajevo, septembar 2014

**Red. prof. dr Melita Ahić-Đokić, dipl.el.inž.**
**Viši asistent mr Emir Sokić, dipl.el.inž.**
**Odsjek za automatiku i elektroniku**
**Sarajevo, 12.01.2014.**

# Tema za završni rad

studenta I ciklusa studija koji studira na ETF-u u skladu sa principima Bolonjskog procesa na Odsjeku za automatiku i elektroniku (šk.2013/14)

## Tema: Razvoj softverskih modula za mjerenje, nadzor i upravljanje ultrazvučnom kamerom

**Student: Softić Kenan**

**Sažetak:**

U okviru rada potrebno je dizajnirati, razviti i testirati MATLAB grafičko okruženje koje omogućava sljedeće:

- upravljanje akvizicijom/generisanjem signala preko NI6024E akvizicione kartice;
- upravljanje RS485 sabirnicom;
- upravljanje USB sabirnicom;
- interpretaciju odnosno slanje podataka sa ultrazvučnih prijemnih odnosno predajnih modula, kao i pozicionih sistema;
- mogućnost implementacije real-time analize okruženja korištenjem ultrazvučne kamere i standardne video kamere;
- mogućnost korištenja u XPC Target Real time modu.

**Polazna literatura:**

1. Melita Ahić-Đokić, *"Signali i sistemi"*, Elektrotehnički fakultet u Sarajevu, 2010.
2. T. Brodić, *"Analogna integrirana elektronika*", Sarajevo: Svjetlost, 1986.
3. B. Carter i L. Huelsman, "*Handbook Of Operation Amplifier Active RC Networks"*, Texas Instruments, 2001.
4. D. Lancaster, *"Active-Filter Cookbook",* Indianapolis: Howard W. Sams & Co. Inc, 1975.
5. M. McRoberts, *Arduino starter kit manual*, London: Earthshine Electronics, 2009.
6. M. Rafiquzzaman, *"Fundamentals of Digital Logic and Microcomputer Design",* Hoboken: JohnWiley & Sons, Inc., 2005.
7. Alan V. Oppenheim, Alan S. Willsky: *"Signals and Systems",* Prentice-Hall, 1997
8. Alan V. Oppenheim, Ronald W. Schafer,John R.Buck: *"Discrete-time Signal Processing",* Prentice Hall, 1999.
9. D. A. Neamen, *"Microelectronics: Circuit Analysis and Design",* New York: McGraw-Hill, 2010.

**Mentor:**

Red. prof. dr Melita Ahić-Đokić

## Sažetak

Rad se bavi razvojem softverskih modula koji bi omogućili mjerenje, nadzor i upravljanje ultrazvučnom kamerom. Razvijene su klase i korisnički interfejsi koji su omogućili akviziciju i generisanje signala, sa posebnim osvrtom na akviziciju i generisanje signala putem SIMULINK-a. Također su razvijene klase i interfejs koji su omogućili serijsku komunikaciju između računara i mikrokontrolera. Oba interfejsa su proširena sa softverskim alatima za prikaz slike okruženja, te analizom ove slike. Pored ovih interfejsa, razvijen je i interfejs koji je omogućio akviziciju signala korištenjem posebnog okruženja za real-time simulaciju xPC Target.

## Abstract

The aim of this work is the development of software modules, which would allow measurement, supervisory and control of ultrasound camera. Classes and intefaces for signal acquisition and generation were developed, and signal acquisition and generation in SIMULINK were reviewed in particular. Class and interface for serial communication between PC and microcontrollers were also developed. Both interfaces were expanded with software tools for display and analysis of enviroment. Along these interfaces, interface for acquisition using special real-time enviroment xPC Target was developed.

# SADRŽAJ

**Uvod** **1**

**1. Uvod u akviziciju i generisanje signala** **3**

1.1. Uvod u signale 3

1.2. Akvizicioni sistem- uvod 5

1.3. Generisanje signala 6

1.4. NI6024E kartica 8

**2. Akvizicija i generisanje signala u MATLAB-u** **10**

2.1. Akvizicija podataka koristeći MATLAB 10

2.2. Generisanje podataka koristeći MATLAB 15

2.3. Korisnički interfejs (GUI) za akviziciju i generisanje podataka 19

2.4. Akvizicija i generisanje podataka korištenjem SIMULINK-a 22

2.4.1. Akvizicija podataka u SIMULINK-u 22

2.4.2 Generisanje podataka u SIMULINK-u 24

2.5.Kreiranje GUI-a za upravljanje SIMULINK modelom 26

2.5.1. Metode za razvoj GUI-a za upravljanje SIMULINK modelom 26

2.5.1.1. Metoda za razvoj GUI-a korištenjem funkcije set_param i callback funkcija 26

2.5.1.2. Metoda za razvoj GUI-a korištenjem s-funcija 26

2.5.1.3. Metoda za razvoj GUI-a korištnjem event_listener-a 27

2.5.2. GUI za upravljanje SIMULINK-om korištenjem s-funkcija 28

2.5.3. GUI za upravljanje SIMULINK-om korištenjem event_listener-a 32

**3. Realizacija serijske komunikacije koristeći MATLAB 39**

3.1. RS-485 komunikacija 39

3.2. USB komunikacija 40

3.3. Klasa za serijsku komunikaciju u MATLAB-u 42

3.4. ARDUINO razvojno okruženje 45

3.4.1. ARDUINO Uno razvojno okruženje 46

3.4.2. Akvizicija podataka putem ARDUINO Uno razvojnog okruženja 47

3.4.3. Serijska komunikacija na ARDUINO Uno razvojnom okruženju 48

3.5. GUI za serijsku komunikaciju između PC-a i Arduino razvojnog okruženja 52

**4. Obrada i prikupljanje slike u MATLAB-u 54**

4.1. Uvod u obradu slike 54

4.2. Akvizicija slike u MATLAB-u 55

4.3. Obrada slike u MATLAB-u 59

**5. GUI za mjerenje, nadzor i upravljanje ultrazvučnom kamerom 64**

5.1. GUI za mjerenje, nadzor i upravljanje ultrazvučnom kamerom preko akvizicione kartice 64

5.2. GUI za mjerenje, nadzor i upravljanje ultrazvučnom kamerom serijskom komunikacijom 66

**6. Real-time simulacija koristeći xPC Target mod 68**

6.1. Uvod u xPC Target mod 68

6.2. Akvizicija i generisanje signala putem xPC Target-a 69

6.3. GUI za mjerenje, nadzor i upravljanje ultrazvučnom karticom u xPC Target modu 71

**7. Zaključak 73**

**8. Lista skraćenica** **74**

**9. Literatura** **75**

**10. Prilog** **77**

# UVOD

U današnje vrijeme posebna pažnja se posvećuje sistemima vizije kod robotskih sistema. Ovo podrazumijeva prikaz okoline u kojoj se nalaze, ali i analizom tog okruženja, kako bi se kontrolisali procesi, otkrivali događaji, ili realizirao određeni tip interakcije. Primjer uređaja koji bi omogućio analizu okruženja bila bi ultrazvučna kamera, koja bi na osnovu ultrazvučnih signala davala informacije o okruženju u kojem se robotski sistem nalazi.

Za ovakav hardver je potrebno razviti softverske module koji omogućuju upravljanje ultrazvučnom kamerom, preko generisanja različitih vrsta signala, mjerenja ultrazvučnom kamerom, što uključuje akviziciju ultrazvučnih signala preko akvizicione kartice ili razvojnog sistema, te nadzor ultrazvučne kamere, pri čemu se vrši prikaz analizirane slike okruženja na osnovu prikupljenih podataka.

U prvom poglavlju su definisani pojmovi akvizicije i generisanja signala, izvršena je podjela signala na odgovarajuće grupe, te su date osnovne informacije o akvizicionoj kartici koja se koristi.

Drugo poglavlje se isključivo bavi razvojem interfejsa za akviziciju i generisanje podataka. Ovo poglavlje uključuje funkcije za akviziciju i generisanje podataka u MATLAB-u, razvijene klase sa akviziciju i generisanje podataka, te korisnički interfejs razvijen na bazi pojašnjenih funkcija i klasa. Također, napravljen je osvrt na akviziciju i generisanje podataka u SIMULINK-u, a posebno su obrađene metode za razvoj interfejsa koji upravljaju SIMULINK modelom. Na osnovu dvije takve metode (s-funkcije i event_listeneri) su razvijeni interfejsi koji su upravljali akvizicijom i generisanjem podataka u okviru SIMULINK-a.

Tipovi serijske komunikacije su obrađeni u trećem dijelu. Ovo poglavlje također uključuje pregled osnovnih funkcija za serijsku komunikaciju u MATLAB-u uz razvijenu klasu i interfejs. Obzirom da je za komunikaciju korišten Arduino Uno, dat je pregled karakteristika ovog okruženja uz programski kôd za ovo okruženje koji je omogućio akviziciju i serijsku komunikaciju sa računarom.

Četvrto poglavlje daje teoretski i praktični uvod u procesiranje slika u okviru MATLAB-a, te su pojašnjeni modeli za akviziciju slike sa kamere u SIMULINK-u. Također, objašnjene su i funkcije za analizu slike okruženja.

Konačni izgledi softverskih modula su prikazani u petom poglavlju, a oni su uključivali interfejs razvijen na osnovu SIMULINK modela za akviziciju i generisanje podataka i interfejs na bazi serijske komunikacije računara i Arduino Uno okruženja.

Šesto poglavlje se bavi mogućnostima akvizicije u okviru real-time okruženja xPC Targeta, te korištenjem napravljenih interfejsa u ovom okruženju, uz kratak osvrt na osnovne podatke o ovom softverskom okruženju.

# 1. Uvod u akviziciju i generisanje signala

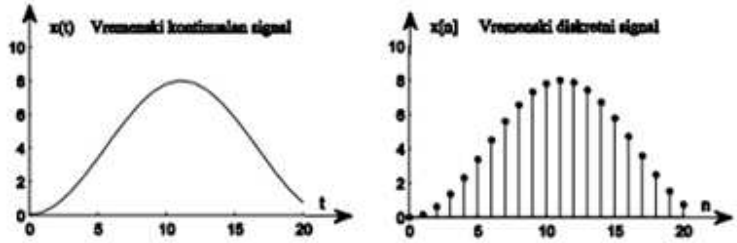
## 1.1. Uvod u signale

Signali su fizičke veličine koje zavise od vremena, prostornih koordinata, ili nekih drugih nezavisno promjenljivih, te nose informaciju koja je sadržana u promjenama signala. Primjeri signala iz svakodnevnog života su brojni: ljudski govor, svjetlost, muzika, video signali, itd. Najčešći signali u elektrotehnici su naponi i struje, a pored njih postoje i drugi, kao što su električno polje, električno opterećenje, itd. Podjela signala se može izvršiti na razne načine. Ovdje će biti pomenuta samo podjela na vremenski kontinualne i vremenski diskretne signale koja se odnosi na ponašanje signala isključivo u vremenu, te podjela signala na analogne i digitalne.

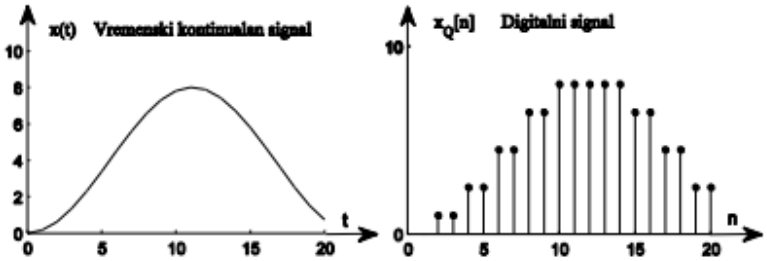
Vremenski kontinualni signal je definiran u svakom trenutku vremena u određenom vremenskom intervalu, dok vremenski diskretni signal ima definirane vrijednosti samo u diskretnim trenucima nezavisno promjenjive, ali ne i između njih. Kontinualni signali su signali koji se susreću u svakodnevnom životu, poput video signala, napona za napajanje tehničkih uređaja, dok za diskretne signale se mogu uzeti redovna mjesečna primanja.

Ukoliko amplituda kontinualnog signala može imati bilo koju vrijednost iz nekog kontinualnog opsega, tada se takav signal naziva analogni signal. U slučaju da amplituda kontinualnog signala može poprimiti samo određene vrijednosti, tada se takav kontinualni signal naziva kvantizirani signal. Ukoliko je amplituda vremenski diskretnog signala također diskretna, tada se takav signal naziva digitalni signal. Većina signala u prirodi je kontinualna, dok su signali koji se obrađuju u računarima (*PC*) digitalni, kao što su muzika na CD-u, snimljene slike na PC-u, itd. Prikaz kontinualnih i diskretnih signala je dat na slici 1.1., dok je prikaz analognih i digitalnih signala dat na slici 1.2.

**Vremenski kontinualni i vremenski diskretni signali**

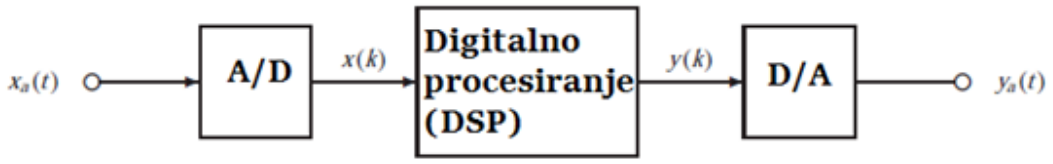



**Slika 1.1.** Prikaz vremenski kontinualnog i vremenski diskretnog signala [1]




**Slika 1.2.** Prikaz analognog i digitalnog signala [1]

Od velikog značaja je i mogućnost pretvaranja analognih u digitalne signale i obrnuto. Proces pretvaranja analognih signala u digitalne za naziva analogno-digitalna konverzija (*A/D)*, dok se proces pretvaranja digitalnih signala u analogne naziva digitalno-analogna konverzija (*D/A*). Prilikom A/D konverzije analogni signal se diskretizuje po vremenu (eng. *sampling*), i kvantizira po amplitudi. Ovaj postupak se obavlja pomoću analogno-digitalnog konvertora. U većini slučajeva digitalni signal se obrađuje propuštanjem kroz digitalni procesor (DSP), te se nakon toga vrši D/A konverzija. Proces D/A konverzije zapravo predstavlja rekonstrukciju analognog signala iz uzoraka, a ovo se obavlja u digitalno-analognom konvertoru. Proces A/D konverzije i D/A konverzije je prikazan na slici 1.3.

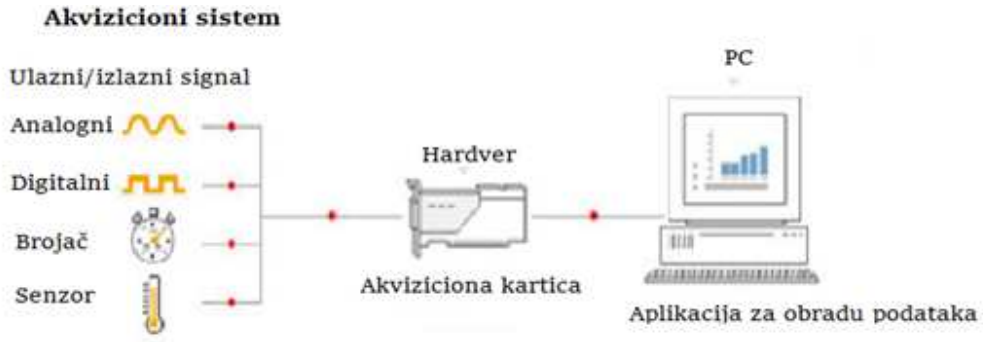



**Slika 1.3.** Proces A/D konverzije i D/A konverzije [3]

Specijalne vrste signala su periodički signali. Kontinualni signal je periodički, ako važi: $f(t)=f(t+T_0)$ za svako $t$, pri čemu $T_0$ predstavlja osnovni period signala, dok je diskretni signal periodički, ako važi: $f[n]=f[n+N]$ za svako $n$, pri čemu $N$ predstavlja osnovni period diskretnog signala. Primjer periodičkog signala je napon mreže sa periodom 0.02 s. Periodički signali su pomenuti, jer će se rad baviti i generisanjem signala, a generisat će se upravo periodički signali.

## 1.2. Akvizicioni sistemi - uvod

Akvizicija (*lat. aquistion* prikupljanje, prisvajanje) predstavlja proces u kojem se realni kontinualni signali (bilo koja fizička veličina poput napona, struje, temperature) prevode u digitalne signale, kako bi se ovi signali mogli mjeriti, obrađivati, ili pohranjivati na PC-u. Prednosti akvizicije u odnosu na tradicionalne metode mjerenja su: ekonomičnost, efikasnost mjerenja, produktivnost zbog korištenja PC-a, obrada mjerenih veličina, mogućnost povezivanja više računara, itd. Akvizicioni sistem, odnosno sistem putem kojeg vršimo akviziciju, se sastoji od senzora koji fizičku veličinu pretvara u neku električnu veličinu, akvizicionog uređaja (često korišten pojam je i akviziciona kartica, odnosno *DAQ device* u anglo-saksonskoj literaturi), te PC-a koji mora imati softver koji će služiti za interpretaciju, obradu, prikaz, ili pohranjivanje podataka sa akvizicione kartice. Pojednostavljena shema akvizicionog sistema je data na slici 1.4.
Sam proces akvizicije nije previše složen. Sastoji se od dovođenja signala sa senzora na ulaz akvizicione kartice, potom uzorkovanja dovedenog signala, te slanja dobijenog digitalnog signala na PC. Ukratko će se pojasniti i proces akvizicije.

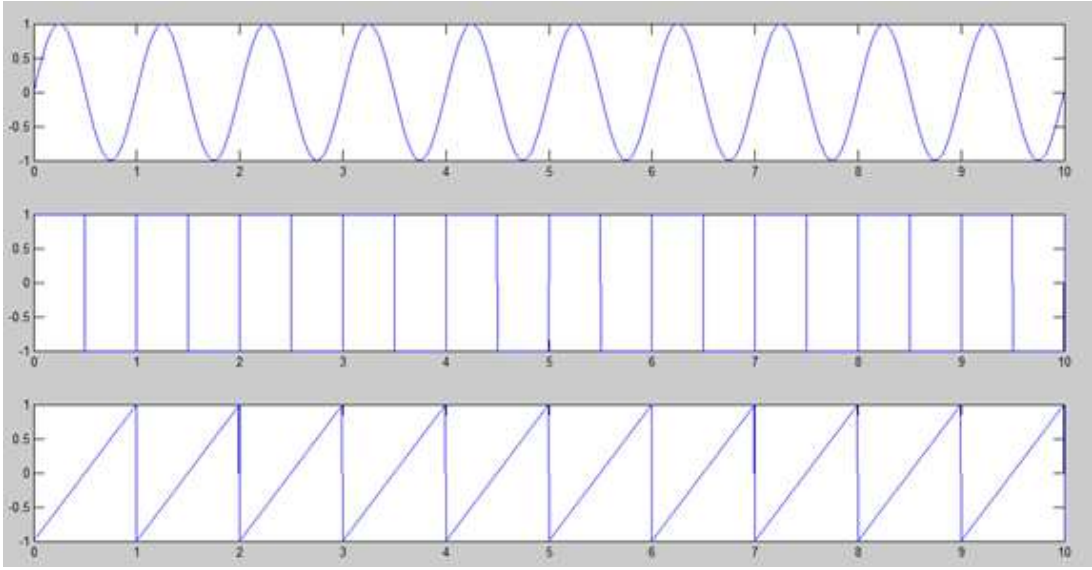



**Slika 1.4.** Shema sistema za akviziciju

Na samom početku signal sa senzora se dovodi na ulaz pojačala, a zatim se vrši uklanjanje, ili u potrebnoj mjeri prigušenje smetnji, šumova, koji se pojavljuju u signalu. Potom se vrši A/D konverzija signala koristeći akvizicionu karticu, pri čemu se mora poštovati Nyuqistova teorema o uzorkovanju, kako ne bi došlo do aliasinga, odnosno kako se ne bi izgubila mogućnost rekonstrukcije originalnog signala. Nakon uzorkovanja signal se šalje PC-u na dalju obradu. Kvaliteta akvizicije ovisi od niza parametara, kao što su: rezolucija, tačnost, maksimalna frekvencija uzorkovanja, brzina slanja podataka PC-u, itd. Kao primjer akvizicije se može uzeti proces snimanja govora (općenito nekog zvuka) na računaru. U tom slučaju zvučna kartica (eng. *winsound*) predstavlja akvizicionu karticu, a računar, zajedno sa zvučnom karticom, aplikacijama i softverom za pohranjivanje, odnosno obradu zvučnog signala, te mikrofonom koji pretvara zvučni signal u električne signale, predstavlja akvizicioni sistem.

## 1.3. Generisanje signala

Pored akvizicije koja predstavlja tipičan primjer A/D konverzije, akviziciona kartica se može koristiti i za proces D/A konverzije. Upravo koristeći akvizicionu karticu, te računar sa odgovarajućim softverom možemo generisati i proizvoljne signale. Generisanje signala predstavlja generisanje periodičnog naponskog signala određenog oblika, amplitude i frekvencije. Najčešći oblici signala koji se generišu su sinusni signal, signal četvrtke, te pilasti signal, prikazani na slici 1.5.



**Slika 1.5.** Sinusni signal, signal četvrtke, te pilasti signal

Matematički opis ovih signala je sljedeći:

Sinusni signal je opisan relacijom:

$$y(t) = A\sin(\omega t + \varphi),$$

gdje je $A$ amplituda signala, ω kružna učestanost signala, a φ faza signala.

Signal četvrtke se opisuje kao:

$$y(t) = A\sum_{n=-\infty}^{+\infty} u(t - nT + \frac{1}{2}) - u(t - nT + \frac{1}{2}),$$

gdje je *u(t)* Heavisidova funkcija, *T* period signala, a *A* amplituda signala.

Pilasti signal je opisan kao:

$$y(t) = 2A(\frac{t}{T} + \left\lfloor \frac{1}{2} + \frac{t}{T} \right\rfloor) \quad ,$$

gdje je *T* period signala, a *A* amplituda signala.

Već je pomenuto da generatori signala moraju generisati signale različite amplitude i frekvencije, pa će ovi pojmovi biti definisani. Amplituda signala predstavlja maksimalnu vrijednost naponskog signala. Pored ove amplitude u praksi se definišu i *peak-to-peak* amplituda, kao razlika između najveće i najmanje vrijednosti signala, te *Root Square Amplitude* (RMS), kao srednja vrijednost signala na periodu. Frekvencija se definiše kao

recipročna vrijednost perioda signala, odnosno $f = \frac{1}{T}$. Pored frekvencije često se definiše i kružna učestanost signala, kao $\omega=2\pi f$. Već je navedeno da će se u radu pažnja posvetiti samo softverskoj realizaciji generisanja signala, tako da se elektronička realizacija generatora signala neće obrađivati.

### 1.4. NI6024E kartica

Kao akviziciona kartica će se koristiti NI6024E kartica, proizvođača National Instruments, pa će u skladu tim biti i opisane njene karakteristike. Prethodno će biti definisani pojmovi, kojima se opisuje akviziciona kartica.

**Sample Rate (frekvencija uzorkovanja)** predstavlja broj uzoraka koji se prikupi u toku jedne sekunde.

**Bit Rate (bps)** predstavlja broj bita koji se prenese, ili obradi u jedinici vremena.

**Dinamički opseg** predstavlja odnos između najveće i najmanje vrijednosti napona koju akviziciona kartica može registrovati.

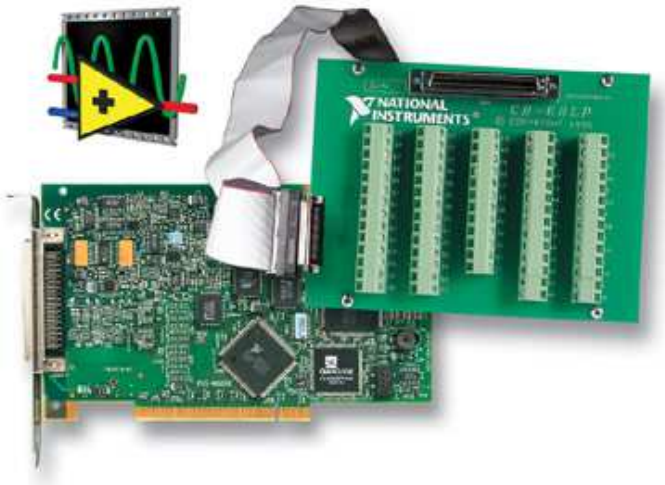
**Rezolucija** se može definisati relacijom $Q = \frac{R}{2^b}$, gdje R predstavlja opseg ulaznog napona u akvizicionu kartica, a b broj bita koji akviziciona kartica ima na raspolaganju.

Pregled karakteristika NI6024E kartice je prikazan u tabeli 1.1.

| | |
|---|---|
| Analogni ulazi | 16 ulaznih kanala za diferencijalni mod rada<br>8 ulaznih kanala za single-ended mod<br>Maksimalni opseg napona: 20 V (bipolarno)<br>Frekvencija uzorkovanja: 200 000 semplova po sekundi<br>Rezolucija: 12 bita |
| Analogni izlazi | 2 izlazna kanala<br>Maksimalna vrijednost izlaznog napona: 10 V<br>Maksimalna frekvencija izlaznog signala: 100 kHZ<br>Rezolucija: 12 bita |
| Digitalni ulazi/izlazi | 8 digitalnih ulaza/izlaza |

| | |
|---|---|
| | Kompatibilni sa TTL tehnologijom, nivo logičke jedinice 5V |
| Brojački ulazi/izlazi | 2 brojačka ulaza/izlaza<br>Rezolucija: 24 bita<br>Maksimalna frekvencija: 20 MHz |

**Tabela 1.1.** Prikaz karakteristike NI6024E kartice [5]




**Slika 1.6**. Fizički izgled NI6024E kartice, preuzeto iz [6]

| | | | |
|---|---|---|---|
| ACH8 | 34 | 68 | ACH0 |
| ACH1 | 33 | 67 | AIGND |
| AIGND | 32 | 66 | ACH9 |
| ACH10 | 31 | 65 | ACH2 |
| ACH3 | 30 | 64 | AIGND |
| AIGND | 29 | 63 | ACH11 |
| ACH4 | 28 | 62 | AISENSE |
| AIGND | 27 | 61 | ACH12 |
| ACH13 | 26 | 60 | ACH5 |
| ACH6 | 25 | 59 | AIGND |
| AIGND | 24 | 58 | ACH14 |
| ACH15 | 23 | 57 | ACH7 |
| DAC0OUT[1] | 22 | 56 | AIGND |
| DAC1OUT[1] | 21 | 55 | AOGND |
| RESERVED | 20 | 54 | AOGND |
| DIO4 | 19 | 53 | DGND |
| DGND | 18 | 52 | DIO0 |
| DIO1 | 17 | 51 | DIO5 |
| DIO6 | 16 | 50 | DGND |
| DGND | 15 | 49 | DIO2 |
| +5 V | 14 | 48 | DIO7 |
| DGND | 13 | 47 | DIO3 |
| DGND | 12 | 46 | SCANCLK |
| PFI0/TRIG1 | 11 | 45 | EXTSTROBE* |
| PFI1/TRIG2 | 10 | 44 | DGND |
| DGND | 9 | 43 | PFI2/CONVERT* |
| +5 V | 8 | 42 | PFI3/GPCTR1_SOURCE |
| DGND | 7 | 41 | PFI4/GPCTR1_GATE |
| PFI5/UPDATE* | 6 | 40 | GPCTR1_OUT |
| PFI6/WFTRIG | 5 | 39 | DGND |
| DGND | 4 | 38 | PFI7/STARTSCAN |
| PFI9/GPCTR0_GATE | 3 | 37 | PFI8/GPCTR0_SOURCE |
| GPCTR0_OUT | 2 | 36 | DGND |
| FREQ_OUT | 1 | 35 | DGND |

**Slika 1.7.** Raspored pinova NI6024E kartice [5]

# 2. Akvizicija i generisanje podataka koristeći MATLAB

Od posebnog interesa je softverski dio akvizicije, tj. na koji način pohraniti, prikazati i obrađivati prikupljene podatke, te vršiti generisanje podataka koristeći određene softver na PC-u (konkretno koristeći MATLAB).

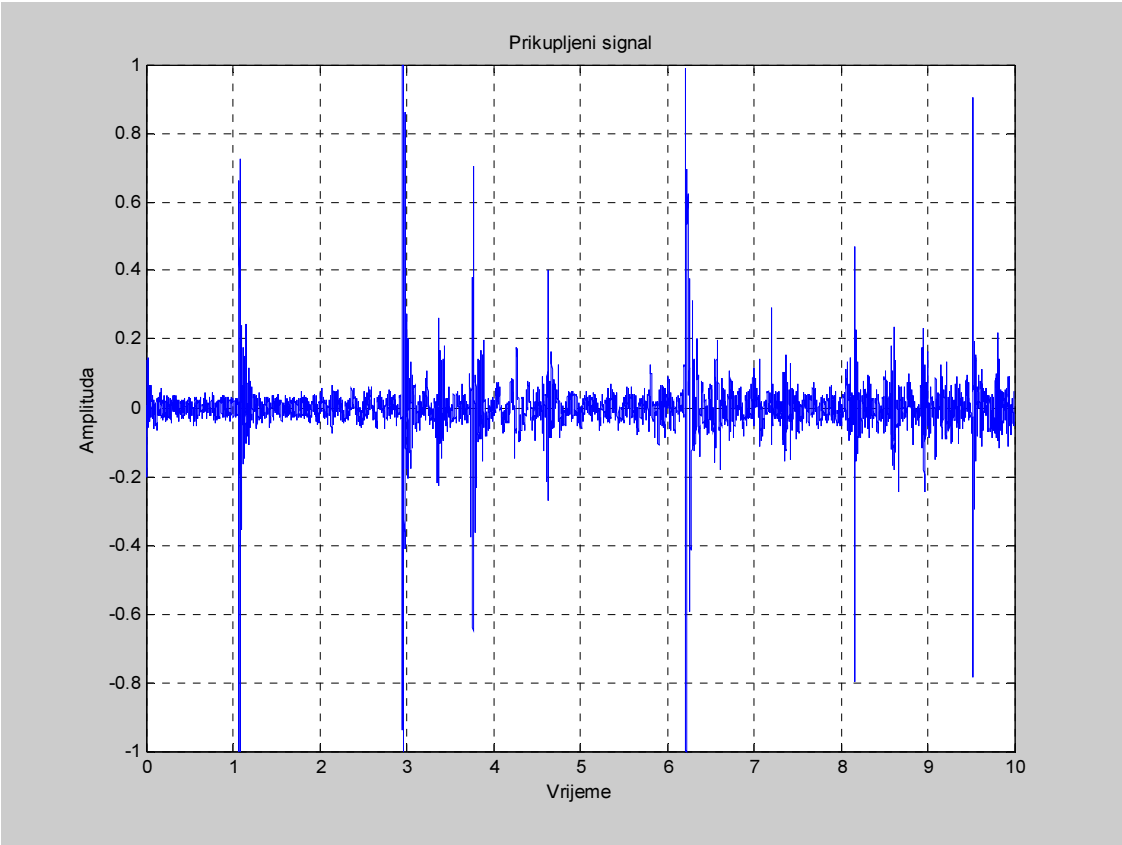
## 2.1. Akvizicija podataka korištenjem MATLAB-a

Već je navedeno da će se za akviziciju koristiti programski paket MATLAB, konkretno MATLAB-ov *Data Acquisition Toolbox.* Zahvaljujući ovom alatu MATLAB može vršiti akviziciju, preko različitih hardvera, kao što su: USB, PCI, PCI Express, PXI, itd. Također, ovaj alat omogućava i slanje podataka, korištenje brojača, digitalnih ulaza i izlaza, zatim direktan pristup naponskim i strujnim signalima, korištenje virtuelnog osciloskopa, itd. Jednostavna akvizicija u okviru MATLAB-a se može izvršiti korištenjem sljedećeg m.fajla [4]:

```
1   ai=analoginput('winsound'); addchannel(ai,1);
2   ai.SampleRate=8000; ai.SamplesPerTrigger=8000*5;
3   start(ai)
4   [d,t]=getdata(ai)
5   plot(t,d);grid on;xlabel('Vrijeme');ylabel('Amplituda')
6   title('Prikupljeni podaci')
```

U prvoj liniji koda se vrši kreiranje objekta tipa *analoginput*, koji je nazvan *ai*. Kao parametar je proslijeđen naziv akvizicione kartice, koja će se koristiti. Pored ovog parametra konstruktor za kreiranje objekta tipa *analoginput,* može primiti i broj, koji predstavlja adresu akvizicione kartice. Pored ove naredbe u prvoj linije je korištena i naredba *addchannel,* preko koje se definišu kanali koji će koristiti za akviziciju, pri čemu se odabrani kanali proslijeđuju kao vektor. U drugoj liniji koda su definisani osnovni parametri neophodni za akviziciju, a to su *SampleRate* i *SamplesPerTrigger*. Preko atributa *SampleRate* se definiše frekvencija uzorkovanja, dok atribut *SamplesPerTrigger*

određuje broj uzoraka, koji će se prikupiti u toku jedne akvizicije. Ovaj parametar određuje dužinu trajanja akvizicije, kao količnik drugog i prvog parametra. U trećoj liniji koda se pokreće akvizicija, dok se u četvrtoj liniji prikupljeni podaci smještaju u varijable. Varijabla *d* predstavlja amplitude prikupljenih podataka, dok su u varijabli *t* smješteni podaci o vremenu. Na samom kraju izvršen je i prikaz prikupljenih podataka, koristeći funkciju *plot*. Pokretanjem ove funkcije se dobija grafik, kao na slici 2.1.




**Slika 2.1.** Prikaz izvršene akvizicije u MATLAB-u

Na ovaj način se izvršava akvizicija u MATLAB-u. Međutim, uočljivo je to da će se podaci prikazati tek nakon što se akvizicija u potpunosti završi, što je veliki nedostatak ukoliko je potreban trenutni prikaz prikupljenih podataka. Iako bi se gornja funkcija mogla dopuniti, kako bi zadovoljavala traženu funkcionalnost, težište će biti na kreiranju vlastite klase za akviziciju. Klasa će imati za cilj trenutno prikazivanje prikupljenih podataka, a pored toga će omogućiti i spremanje prikupljenih podataka u određene varijable. U skladu sa ovim atributi klase za akviziciju (koja će biti nazvana akvizicija) će

biti hardver koji se koristi za akviziciju, frekvencija uzorkovanja, trajanje akvizicije, vrijeme osvježavanja, odnosno vrijeme nakon kojeg će se prikazivati novi podaci na grafiku, kanali za vršenje akvizicije, prikupljeni podaci, vremenski intervali u kojima se vršila akvizicija, te odabir načina akvizicije, odnosno da li će akvizicija biti pokrenuta softverski, ili hardverski. Atributi ove klase[1] su u tom slučaju:

```
classdef akvizicija
  Properties
     fs      % frekvencija uzorkovanja
     t_akv   % trajanje akvizicije
     ref_akv % vrijeme osvježavanja
     Daq     % akviziciona kartica koja se koristi
     Kanali  % kanali koji se koriste za akviziciju
     mod     % način akvizicije
     t       % vrijeme akvizicije
     Data    % prikupljeni podaci
  end
```

Pored atributa, bit će potrebno definisati i metodu ove klase za pokretanje akvizicije (nazvana *aquisit*), te *konstruktor* koji će kreirati objekat tipa *akvizicija*. Kako bi se obezbjedio trenutni prikaz akvizicije potrebno je definisati i odgovarajuću *TimerFcn*, odnosno funkciju koja će se pozivati nakon određenog vremenskog intervala, a koja će omogućiti trenutni prikaz signala. Ova funkcija će biti definisana unutar klase, kao statička metoda. Također, klasa će imati i metodu *help* koja daje osnovne informacije o ovoj klasi. Metode ove klase će biti:

*Konstrukor* bez parametara se može realizirati kao:

```
function akv=akvizicija
      akvizicija.fs=8000;
      akvizicija.t_akv=2;
      akvizicija.ref_akv=0.1;
      akvizicija.kanali=1;
      akvizicija.daq='winsound';
      akvizicija.mod='start'
      akvizicija.data=[];
      akvizicija.t=[];
      akv.fs=8000;
      akv.t_akv=2;
      akv.ref_akv=0.1;
```

[1] Akvizicija\akvizicija.m

```
        akv.kanali=1;
        akv.daq='winsound';
        akv.mod='start'
        akv.data=[];
        akv.t=[];
   end
```

Na ovaj način se može kreirati objekat tipa *akvizicija.* Obzirom da su svi podaci javni, neće biti razmatran *konstruktor* sa više parametara, jer se mijenjanje može izvršiti naknadno. Pri kreiranju *konstruktora* koriste se podrazumijevane vrijednosti za parametre akvizicije.

Funkcija za pokretanje akvizicije:

```
function aquisit(obj,a,b,c,d,e,f)
      if nargin==7
           obj.fs=a;
           obj.t_akv=b;
           obj.ref_akv=c;
           obj.kanali=d;
           obj.daq=e;
           obj.mod=f;
      elseif nargin==6
           obj.fs=a;
           obj.t_akv=b;
           obj.ref_akv=c;
           obj.kanali=d;
           obj.daq=e;
       elseif nargin ==5
          obj.fs=a;
           obj.t_akv=b;
           obj.ref_akv=c;
           obj.kanali=d;
       elseif nargin==4
           obj.fs=a;
           obj.t_akv=b;
           obj.ref_akv=c;
       elseif nargin==3
           obj.fs=a;
           obj.t_akv=b;
       elseif nargin==2
           obj.fs=a;
       elseif nargin>7
           error('Nisu proslijedeni ispravni parametri');
        end
           global ai;
           global data1;
```

```
            data1=[];
            global t1;
            t1=[];
            y=obj.daq;
            ai=analoginput(y);
            x=obj.kanali;
            addchannel(ai,x);
            ai.SampleRate=obj.fs;
            ai.SamplesPerTrigger=obj.fs*obj.t_akv;
            set(ai,'TimerPeriod',obj.ref_akv);
            set(ai,'TimerFcn',{@akvizicija.plott, obj,obj.data,obj.t});
        if strcmpi(obj.mod,'start')
            start(ai)
        elseif strcmpi(obj.mod,'trigger')
            trigger(ai);
        else
            error('neispravan parametar');
        end
        while(strcmpi(get(ai,'Running'),'On'))
            pause(obj.ref_akv);
        end
            akv.data=data1;
            akv.t=time;
            delete(ai);
            clear
```

Funkcija kojom se vrši pokretanje akvizicije može biti pozvana, koristeći jedan do sedam parametara, a pri tome se postavljaju parametri akvizicije: frekvencija uzorkovanja, trajanje akvizicije, način pokretanja akvizicije, kanali koje se koriste, vrijeme osvježavanja i akviziciona kartica. Unutar ove funkcije vrši se definiranje odgovarajućeg objekta za akviziciju, te se akvizicija pokreće hardverski, ili softverski u ovisnosti od načina rada. Pored ovoga definiše se i pomoćna funkcija *plott,* koja se koristi, kao *TimerFcn*, a njena uloga jeste da iscrtava prikupljene podatke tokom vremena sa zadanim vremenom osvježavanja.

Pomoćna funkcija *plott* koja se koristi kao *TimerFcn*:

```
function plott(hObject,event)
    [d,time]=getdata(hObject,hObject.SamplesAvailable);
    global data1;
    global t1;
    data1 = [data1; d];
    t1= [t1; time];
    cla;
```

```
plot(t1,data1);
grid on;
xlabel('Vrijeme')
ylabel('Amplituda')
title('Prikupljeni podaci')
end
```

Funkcija prima dva parametra; prvi je *hObject* koji predstavlja objekat akvizicije, a drugi *event* nema ulogu, ali se po pravilu definiše za slučaj pomoćnih funkcija unutar funkcije. Funkcija varijable *data1* i *t1* dopunjava postojeće vektore novim vrijednostima, koje odgovaraju prikupljenim podacima, odnosno podacima o vremenu. Pri tome se podaci stalno prikazuju na grafiku, a ovo je omogućeno naredbom *cla,* koja iscrtava grafik na već otvorenom prozoru.

Pokretanje akvizicije korištenjem klase se može obaviti na sljedeći način.

```
1   aqui=akvizicija
2   aquisit(aqui);
```

Prvo se definiše objekat tipa *akvizicija*, imena *aqui*, a potom se pokeće akvizicija preko naredbe metode *aquisit* koju sadrži klasa. Obzirom da je proslijeđen samo objekat, pokrenut će se akvizicija sa podrazumijevanim vrijednostima.

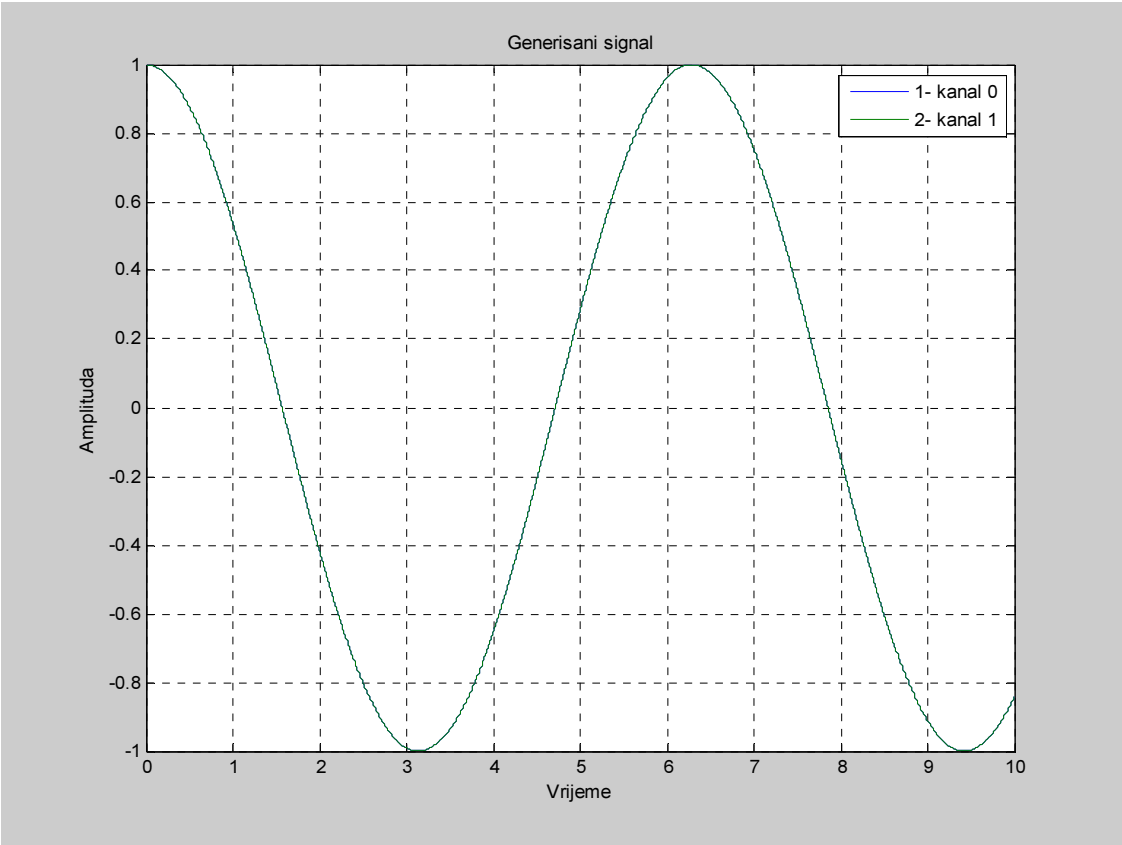
### 2.2. Generisanje signala korištenjem MATLAB-a

Jednostavno generisanje signala se može izvršiti koristeći sljedeći m.fajl:

```
1   ao=analogoutput('winsound');addchannel(ao,[1 2])
2   ao.SampleRate=8000;
3   t=0:1/8000:10-1/8000;
4   data=sin(2*pi*10*t);data=[data; data];putdata(ao,data')
5   start(ao);
6   delete(ao);
7   plot(t,data)
8   grid on;xlabel('Vrijeme');ylabel('Amplituda');
9   title('Generisani signal')
```

Kôd za generisanje signala je dosta sličan kodu za akviziciju signala. Naime, u prvoj liniji koda se definiše objekat tipa *analogouptut* (na isti način kao i *analoginput*), te se potom definišu kanali preko kojih će se slati signal, dok se u drugoj liniji koda definiše

odgovarajuća frekvencija uzorkovanja. U trećoj liniji koda se definišu vremenski trenuci u kojima se šalje izlazni signal (može se primjetiti da je razlika između dva vremenska trenutka u kojima se šalje signal obrnuto proporcionalna frekvenciji uzorkovanja). Nakon toga u četvrtoj liniji koda se definiše oblik izlaznog signala u funkciji varijable vremena, te se preko naredbe *putdata* ovaj signal sprema za slanje. Pokretanje slanja signala se vrši preko naredbe *start (ao)*, a na samom kraju se ovaj signal iscrtava na grafiku.




**Slika 2.2.** Generisani signal

Moguće je i definisanje klase[2] za generisanje podataka na sličan način, kao i klase za akviziciju podataka. Za generisanje signala potrebno je definisati trajanje signala, frekvenciju uzorkovanja, amplitudu i frekvenciju signala, akvizicionu karticu koja će generisati signal, kanale preko kojih će se generisati signal, te oblik signala. Iz ovog razloga atributi klase *generisanje* će biti:

[2] Akvizicija\generisanje.m

```
1   classdef generisanje
2   properties
3       fs;    % frekvencija uzorkovanja
4       amp;   % amplituda signala
5       frekv; % frekvencija signala
6       t_gen; % vrijeme trajanja signala
7       kanali % kanali preko kojih se vrši generisanje podatka
8       daq;   % akviziciona kartica preko koje se generiše signal
9       signal % oblik signala
10    end
```

Pored ovih atributa *klasa* će morati imati *konstruktor* za kreiranje objekata tipa generisanje, te metodu za pokretanje generisanja signala.

*Konstruktor* bez parametara se može realizirati sljedećom funkcijom:

```
    function gen=generisanje
        generisanje.fs=8000;
        generisanje.t_gen=10;
        generisanje.amp=1;
        generisanje.kanali=1;
        generisanje.frekv=10;
        generisanje.signal='sinus';
        generisanje.daq='winsound'
        gen.fs=8000;
        gen.t_gen=10;
        gen.amp=1;
        gen.kanali=1;
        gen.frekv=10;
        gen.daq='winsound';
        gen.signal='sinus';
    end
```

Treba istaći da su parametri *konstruktora* postavljeni na podrazumijevane vrijednosti. Pošto su svi atributi klase javni, mogu se mijenjati naknadno, pa iz toga razloga nije u interesu definiranje dodatnih metoda, ili pravljenje konstruktora sa više parametara. Za metodu kojom se generišu signali može se koristiti sljedeća funkcija:

```
function generisi(obj,a,b,c,d,e,f,g)
        if nargin==8
            obj.fs=a;
            obj.amp=b;
            obj.frekv=c;
```

```
        obj.t_gen=d;
        obj.kanali=e;
        obj.daq=f;
        obj.signal=g;
    elseif nargin==7
        obj.fs=a;
        obj.amp=b;
        obj.frekv=c;
        obj.t_gen=d;
        obj.kanali=e;
        obj.daq=f;
    elseif nargin==6
        obj.fs=a;
        obj.amp=b;
        obj.frekv=c;
        obj.t_gen=d;
        obj.kanali=e;
    elseif nargin ==5
        obj.fs=a;
        obj.amp=b;
        obj.frekv=c;
        obj.t_gen=d;
    elseif nargin==4
        obj.fs=a;
        obj.amp=b;
        obj.frekv=c;
    elseif nargin==3
        obj.fs=a;
        obj.amp=b;
    elseif nargin==2
        obj.fs=a;
    elseif nargin>8
        error('Nisu proslijedeni ispravni parametri');
    end
        y=obj.daq;
        td=0:1/obj.fs:obj.t-gen-1/obj.fs;
    if(strcmpi('sinus',obj.signal))
        data=obj.amp*sin(2*pi*obj.frekv*td)
    elseif strcmpi('cetvrtka',obj.signal)
        data=obj.amp*square(2*pi*obj.frekv*td)
    elseif strcmpi('trokut',obj.signal)
        data=obj.amp*sawtooth(2*pi*obj.frekv*td)
    elseif strcmpi('pila',obj.signal)
        data=obj.amp*sawtooth(2*pi*obj.frekv*td,0.5)
    else
       error('Parametar nije ispravan')
    end
        ao=analogoutput(y)
        x=obj.kanali;
        addchannel(ao,x)
```

```
            tdata=[];
        for i=1:size(x,2)
            tdata=[tdata data']
        end
            putdata(ao,tdata)
            start(ao)
            wait(ao,obj.drtion+1)
            plot(td,tdata)
            xlabel('Vrijeme')
            ylabel('Amplituda')
            title('Generisani signal')
            grid on
            delete(ao)
            clear
end
```

Ova funkcija može primiti jedan do osam parametara, pri čemu svaki parametar definiše atribute klase. Ukoliko se proslijedi samo jedan parametar, a to je objekat tipa generisanje, atributi će biti postavljeni na podrazumijevane vrijednosti. Pokretanje generisanja signala se može izvršiti sljedećim kodom:

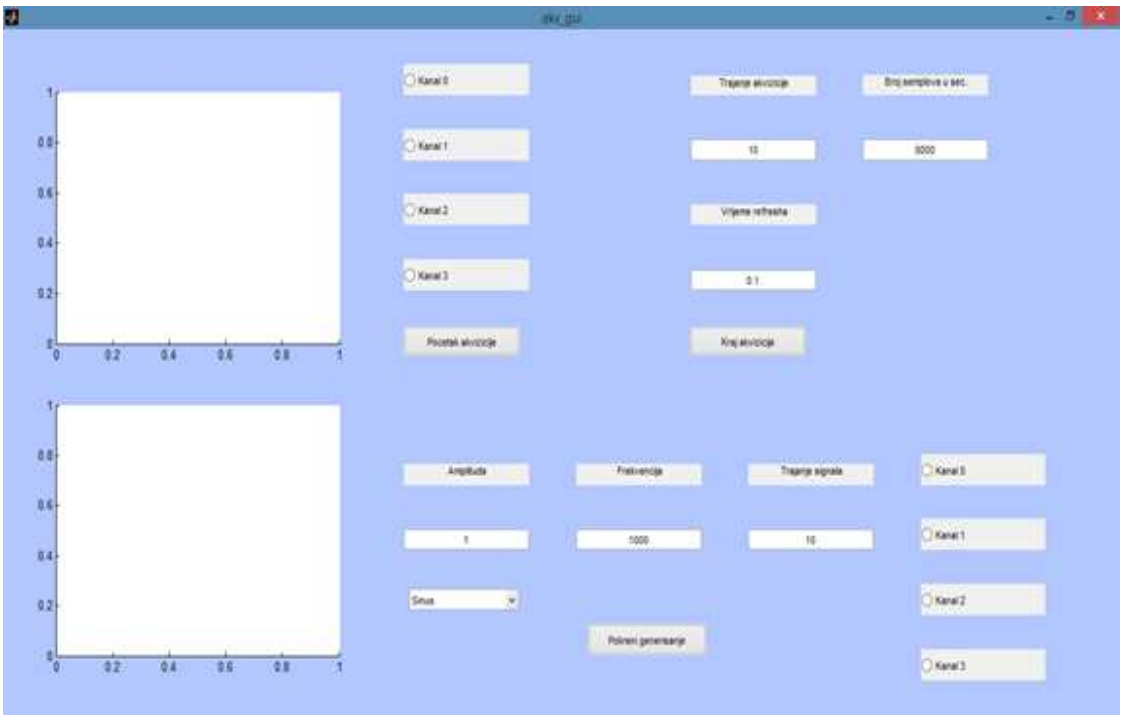
```
1   gen=generisanje
2   generisi(gen)
```

Prvo se kreira objekat tipa *generisanje*, a nakon toga se poziva metoda *generisi* za generisanje podataka. Ovo će pokrenuti generisanje podataka sa podrazumijevanim vrijednostima atributa, te će prikazati signal koji se generisao.

Na ovaj način se mogu izvršiti i akvizicija i generisanje podataka u MATLAB-u. Međutim, primjetno je da ovakav pristup ima nemogućnost zaustavljanja akvizicije, ili generisanja podataka, koji se ne može otkloniti ni proširivanjem klasa sa dodatnim metodama za zaustavljanje akvizicije, odnosno generisanja. Dakle, ukoliko bi bilo potrebno istovremeno i prikupljati podatke i generisati signal, ovakav pristup ne bi bio moguć. Ovakav problem se može otkloniti koristeći korisnički intefejs (GUI).

### 2.3. Korisnički interfejs (GUI) za akviziciju i generisanje podataka

Na slici 2.3. je prikazan GUI kojim se može vršiti akvizicija, odnosno generisanje podataka.

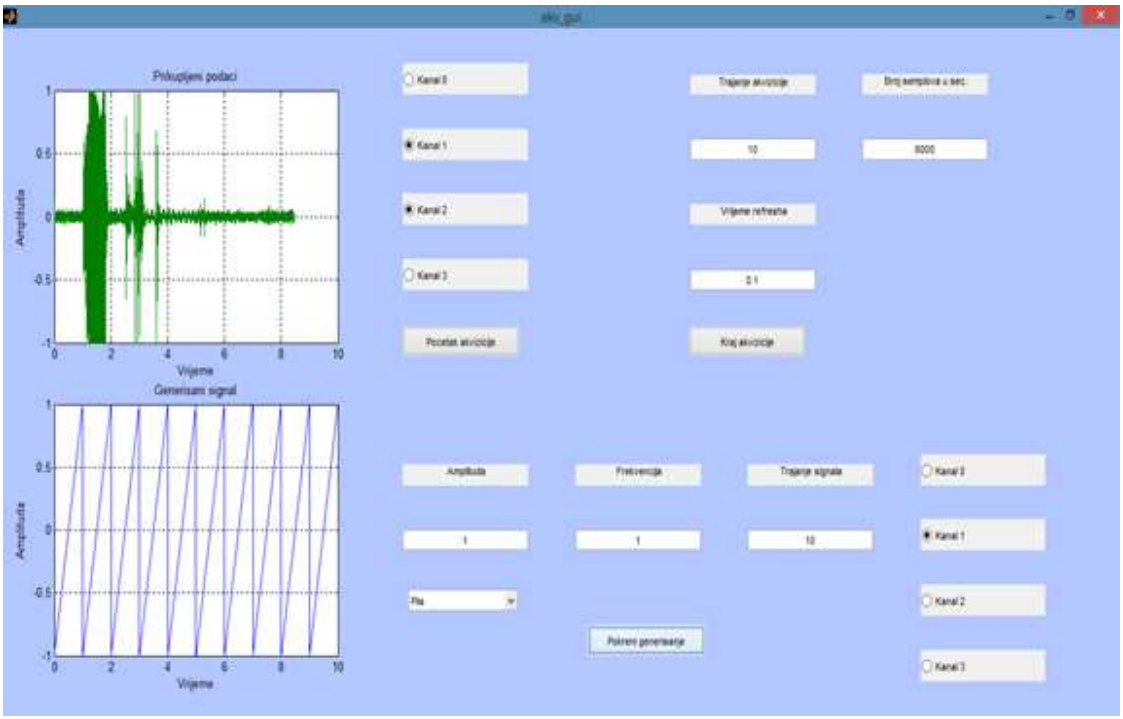



**Slika 2.3.** GUI za akviziciju i generisanje signala

Prednost korištenja GUI-a jeste to što je sada moguće zaustavljanje akvizicije. Također, odabir parametara akvizicije i signala koji se želi generisati je pojednostvaljen. Pored toga moguće je vršiti paralelno akviziciju i generisanje podataka. Pokretanjem akvizicije dobijamo grafik na slici 2.4.

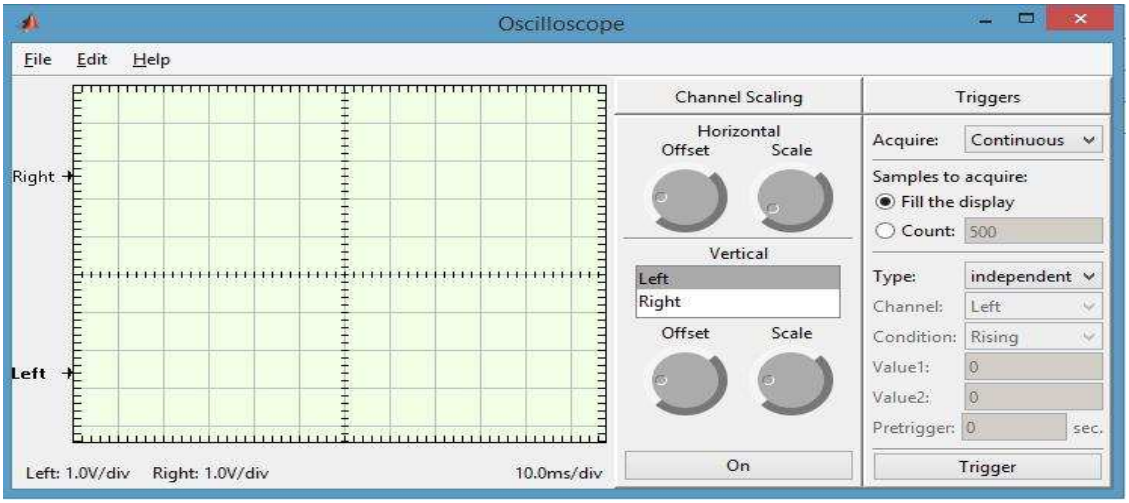
Algoritam kojim je realiziran GUI je zasnovan na već opisanim postupcima akvizicije, odnosno generisanja podataka. Mada je sada omogućeno zaustavljanje akvizicije, te istovremeno prikupljanje i generisanje podataka i dalje postoje određeni nedostaci. Prije svega generisanje signala sa jako velikim trajanjem zaustavlja prikupljanje podataka, pa se gube podaci. Pored toga nije moguće promijeniti prikaz signale, koji se pokazuju na GUI-u .

Akvizicija i generisanje podataka u MATLAB-u se može izvršiti i pomoću SIMULINK-a, a ovakav pristup otklanja osnovne nedostatke prethodnog GUI-a, prije svega nemogućnost istovremene akvizicije i generisanja podatka.




**Slika 2.4.** Izgled GUI-a prilikom pokretanja i akvizicije i generisanja podataka

Prije nego što u nastavku bude više riječi o akviziciji i generisanju podataka putem SIMULINK-a, treba istaći da MATLAB nudi i virtuelni osciloskop. Pristup ovom virtuelnom instrumentu je preko naredbe *softscope*. Pokretanjem dobijamo instrument, kao na slici 2.5.




**Slika 2.5.** Virtuelni osciloskop u MATLAB-u

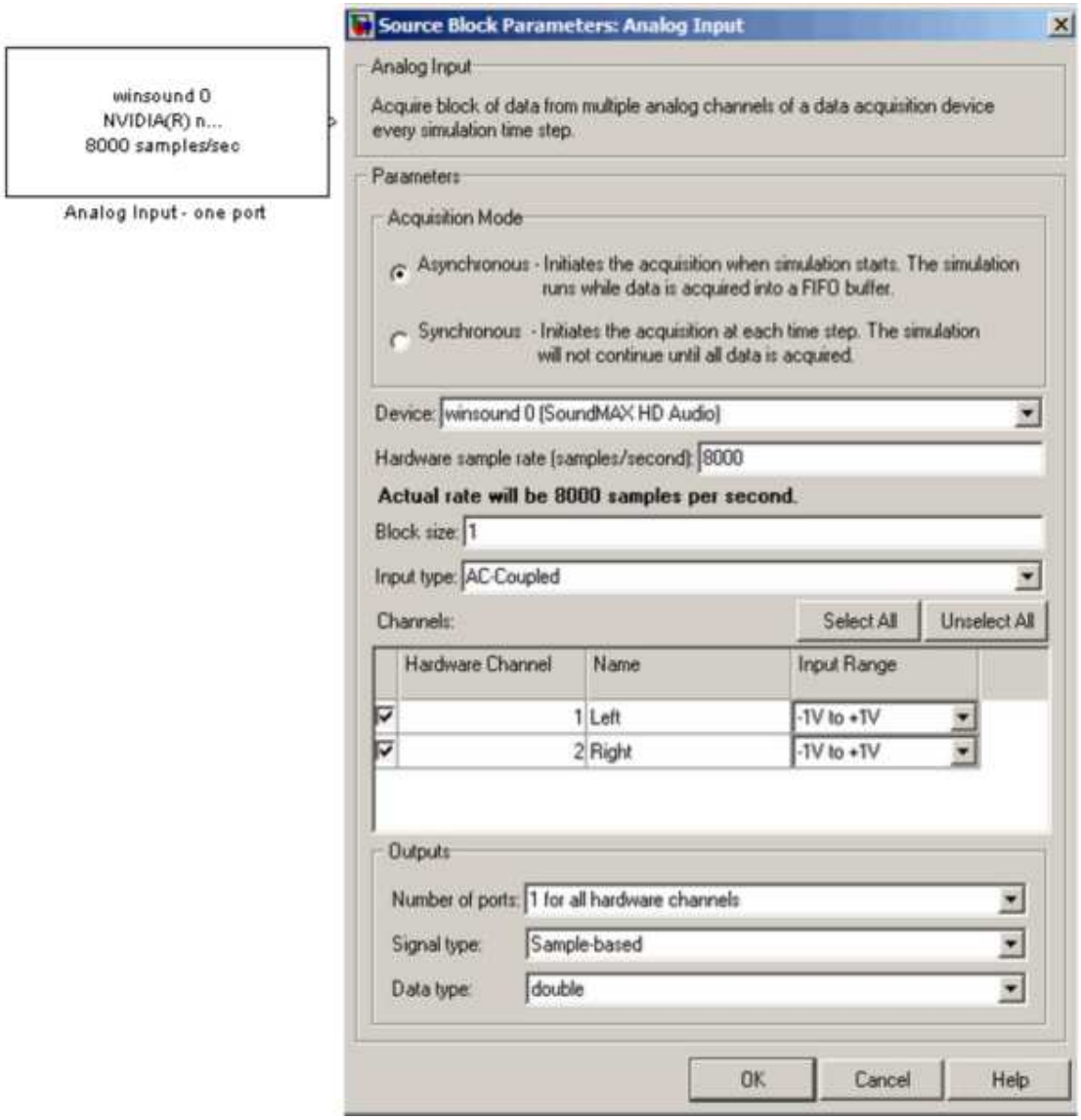
Na ovaj način se jednostavno mogu prikazati prikupljeni podaci, te mijenjati parametri akvizicije.

## 2.4. Akvizicija i generisanje podataka korištenjem SIMULINK-a

Osnov za akviziciju i generisanje podataka u SIMULINK-u predstavljaju blokovi *AnalogInput* i *AnalogOutput*. U nastavku će biti pojašnjeni ovi blokovi.

### 2.4.1. Akvizicija podataka u SIMULINK-u

Akvizicija podataka u SIMULINK-u se vrši dodavanjem bloka *AnalogInput* u SIMULINK model. Pri tome klikom na ovaj blok se mogu podesiti parametri akvizicije, poput odabira kanala preko kojih će se vršiti akvizicija, frekvencija uzorkovanja, ulazni opseg, itd. Parametri bloka *AnalogInput* su prikazani na slici 2.6.
Ukratko će biti objašnjeni i parametri ovog bloka:
**Acquisition Mode-** odabir načina akvizicije, pri čemu su mogući asinhroni (*asynchronous*) i sinhroni (*synchronous*) način rada. Asinhroni način rada podrazumijeva da će se signal na izlazu iz bloka pojaviti tek nakon što se prikupi dovoljno podataka u bufferu (spremniku podataka). Koristeći ovaj način rada, akvizicija je neprekidna, dok se podaci prikupljaju (ali ne i šalju) prilikom svakog koraka simulacije. Sinhroni način rada omogućava da se podaci prikupljeni tokom jednog koraka simulacije direktno šalju na izlaz, bez obzira da li se buffer napunio. Od posebnog interesa jeste prikaz akvizicije u realnom vremenu, pa će biti korišten sinhroni način rada.
**Device-** akviziciona kartica koja se koristi za prikupljanje podataka.Već je rečeno da će se kao akviziciona kartica koristiti NI6024E, pa se i ovaj parametar neće mijenjati.
**Hardware sample rate-** frekvencija uzorkovanja. Ovaj parametar će se moći mijenjati preko GUI-a.
**Block size-** broj uzoraka na izlazu iz bloka prilikom svakog koraka simulacije.
**Input Type-** definiše mod kanala, odnosno ostavlja odabir između diferencijalnog i single-ended moda.




**Slika 2.6.** Prikaz bloka *AnalogInput* sa njegovim parametrima

**Hardware Channel-** odabir kanala koji se koriste za akviziciju, pri čemu je pored odabira moguć i izbor imena kanala, njegov broj, te odabir opsega za kanal.

**Number of ports-** odabir načina slanja signala iz bloka. Pri tome je moguć odabir između *1 per hardware channel* (za svaki kanal će biti definisan poseban izlaz), ili *1 for all hardware channels* (svi prikupljeni podaci će biti slati preko samo jednog izlaza).

**Signal Type-** odabir tipa izlaznog signala. Pri ovome je moguć odabir između *SampleBased* i *FrameBased* tipa. Prvi tip se češće koristi i podrazumijeva da će izlazni signal biti zapravo uzorci prikupljenog signala, dok drugi tip podrazumijeva da će izlazni

signal biti formiran na način da se prvo prikupi određeni broj uzoraka, pa tek onda vrši slanje signala. Za tip izlaznog signala će se koristiti *SampleBased*, iz razloga što je dosta jednostavniji za korištenje.

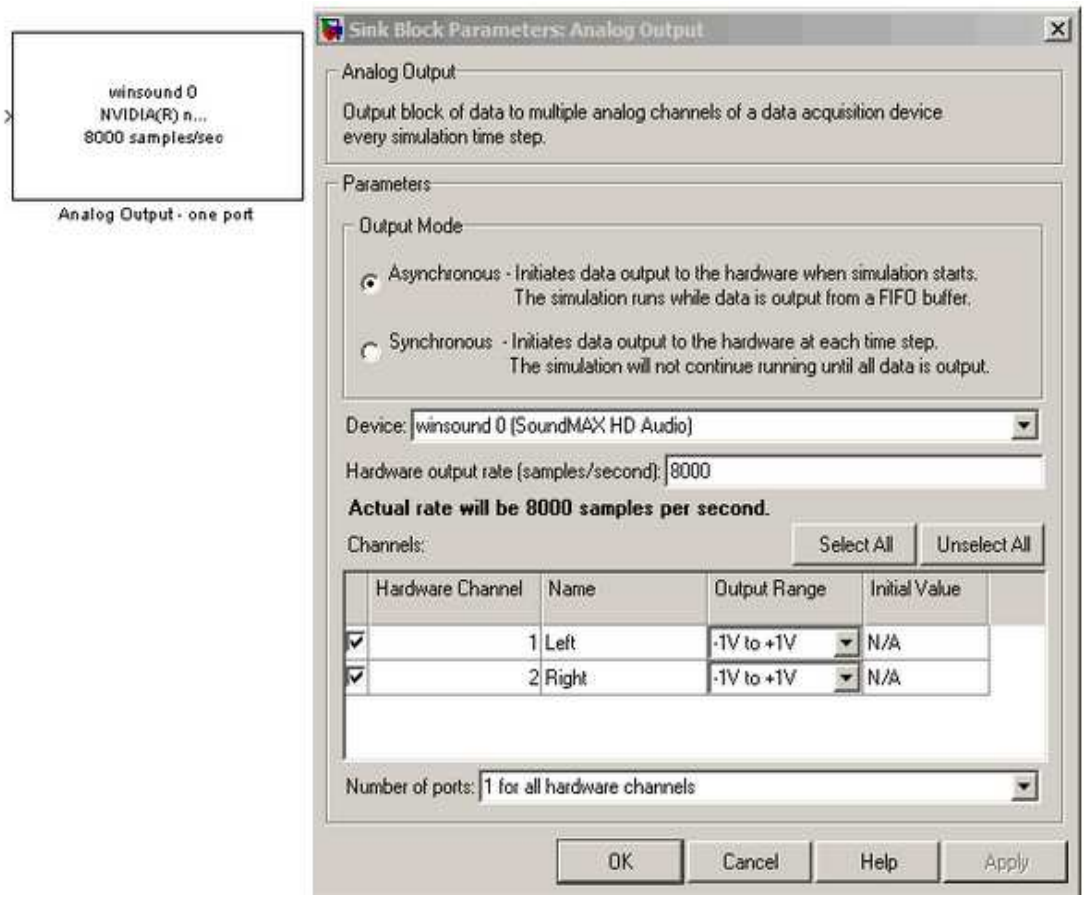
**Data Type-** preko kojeg tipa podataka će se predstavljati izlazni signal. Moguć je odabir *double,*, *int16* i *int8.* Ovaj parametar nije od pretjeranog značaja, pa će koristiti tip *double*.

### 2.4.2. Generisanje podataka u SIMULINK-u

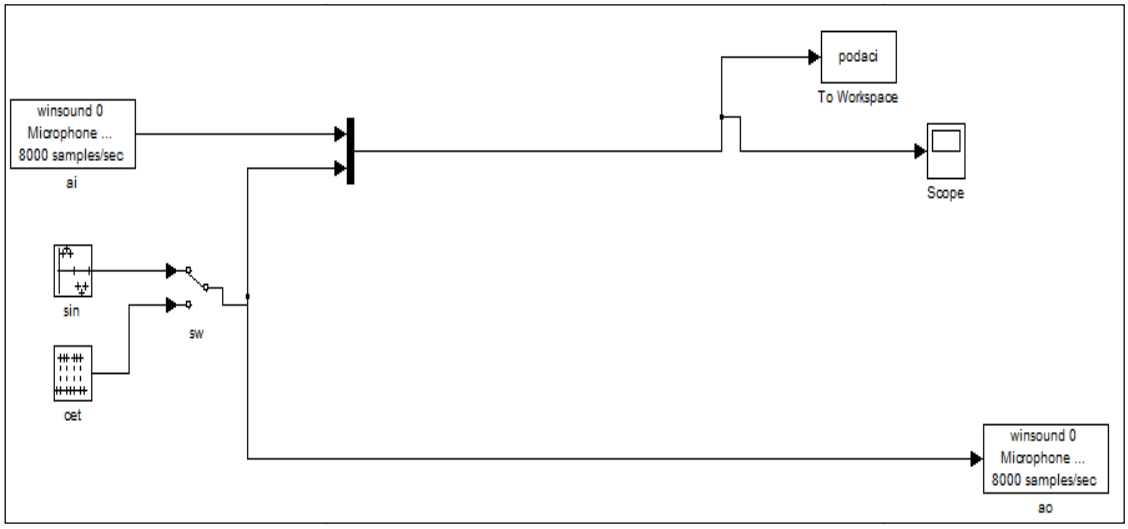
Generisanje podataka se vrši dodavanjem bloka *AnalogOutput*. Analogno prethodnom bloku, klikom se mogu podesiti parametri, kao što su akviziciona kartica, frekvencija uzorkovanja, izlazni opseg, kanali preko kojih se šalju podaci, itd. Parametri ovog bloka su prikazani na slici 2.7. Obzirom da su parametri ovog bloka slični parametrima bloka *AnalogInput* detaljnije neće biti pojašnjeni.

Pored korištenja analognih ulaza, odnosno izlaza, SIMULINK omogućava i korištenje digitalnih ulaza i izlaza. Iako korištenje SIMULINK-a, kao što je već navedeno, otklanja osnovne nedostatke prethodno navedenog GUI-a, i dalje postoje određeni nedostaci. Promjena parametara blokova se mora realizirati otvaranjem svakog bloka, te se ne može izvršiti, ako je simulacija pokrenuta. Također, prikaz prikupljenih podataka je moguć korištenjem *Scope*-a, a pristup podacima je moguć tek nakon što akvizicija završi, ili se zaustavi. Također, otvaranje SIMULINK-a je znatno sporije i zahtijeva više računarskih resursa od korištenje m.fajlova, ili GUI-a za akviziciju. Neki od ovih nedostataka, prije svega jednostavniju promjenu parametara simulacije u SIMULINK-u se mogu otkloniti pravljenjem GUI-a za upravljanje modelom u SIMULINK-u.

Pored ovih blokova u SIMULINK-u postoje i blokovi *AnalogInput(Single Sample)* i *AnalogOutput(Single Sample)*, koji prikupljaju, odnosno generišu po jedan uzorak za svaku periodu uzorkovanja, ali oni nisu ovdje detaljnije pojašnjeni, obzirom da je njihov princip rada jako sličan prethodno navedenim blokovima *AnalogInput*, odnosno *AnalogOutput*.




**Slika 2.7.** Prikaz bloka *AnalogOutput* i njegovih parametara




**Slika 2.8.** SIMULINK model za akviziciju i generisanje podataka

## 2.5. Kreiranje GUI-a za upravljanje SIMULINK modelom

Za razvoj GUI-a koji omogućava upravljanje SIMULINK modelom koriste se tri metode [8]. U nastavku će biti nakratko pojašnjena svaka od tri metode, te njihove prednosti i nedostaci.

### 2.5.1. Metode za razvoj GUI za upravljanje SIMULINK modelom

#### 2.5.1.1. Metoda za razvoj GUI-a korištenjem funkcije set_param i callback funkcija

Prva metoda se zasniva na korištenju funkcije *set_param*, pri mijenjanju određenih podataka na GUI-u, koje se registruje preko *callback* funkcija. Korištenje funkcije *set_param* omogućava mijenjanje parametara blokova u SIMULINK-u. Pomoću ove funkcije se mogu mijenjati parametri akvizicije, ili generisanja signala, poput frekvencije uzorkovanja, broja uzoraka koji će prikupiti, trajanja akvizicije, itd. Pozivanje ove funkcije se vrši na sljedeći način:

```
1   load_system('modelakv.mdl')
2   set_param('modelakv/AnalogInput','SampleRate','44100')
```

Na ovaj način se poziva model, koji nosi ime "modelakv", te parametar "SampleRate" njegovog bloka AnalogInput se postavlja na vrijednost 44100 uzoraka u sekundi. Osnovna prednost ove metode jeste jednostavnost, jer se lako mogu mijenjati parametri blokova. Osnovna mana ove metode je nemogućnost pristupu signalima u SIMULINK-u, sve dok simulacija traje.

#### 2.5.1.2. Metoda za razvoj GUI-a korištenjem s-funkcija

Druga metoda se zasniva na korištenju *s-funkcije*. S-funkcije u suštini predstavljaju funkcije, preko kojih se realiziraju tražene funkcionalnosti blokova u SIMULINK-u. Pored toga što ove funkcije mogu biti pisane u okviru MATLAB-a, omogućeno je i pisanje ovih funkcija u okviru programskog jezika C, odnosno C++.

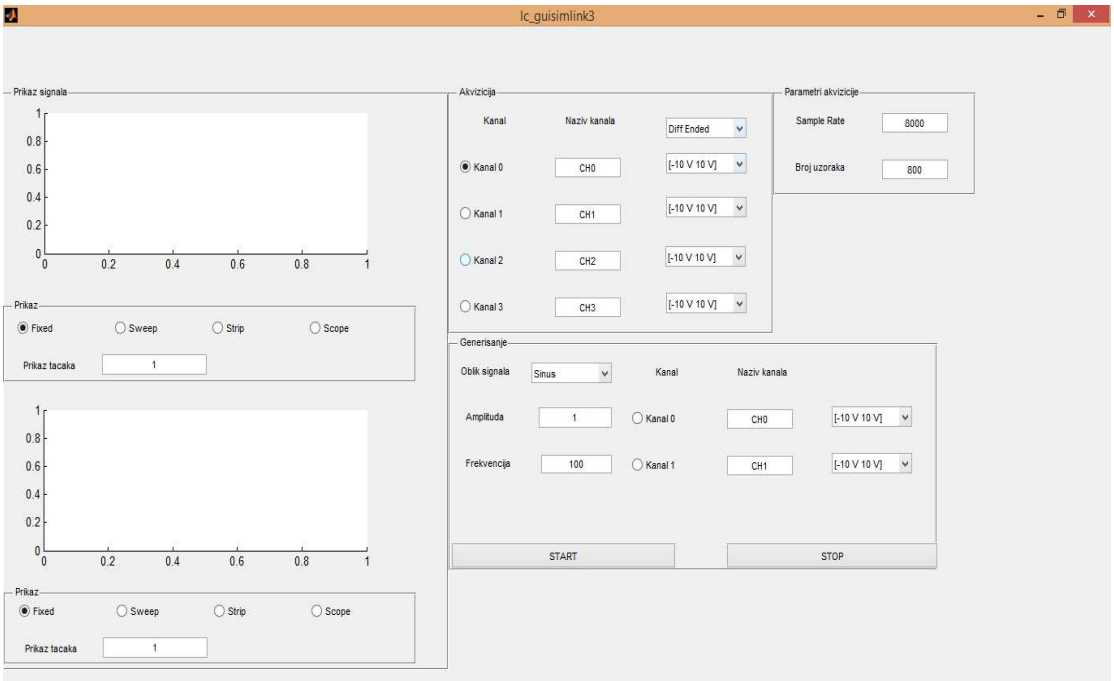
Ideja ove metode jeste pisanje s-funkcija, koje omogućuju prikaz podataka iz SIMULINK-a na GUI-u, te postavljanje parametara simulacije na osnovu podataka iz GUI-a. Prednost ove metode, jeste ta što se parametri simulacije mogu mijenjati, bez zaustavljanja, ili pauziranja simulacije, a signali iz simulacije se mogu prikazati bez

čekanja kraja simulacije. Mana ove metode jeste što je nešto složenija u odnosu na prethodnu i što zahtijeva korištenje dodatnih blokova u SIMULINK modelu.

#### 2.5.1.3. Metoda za razvoj GUI-a korištenjem event_listener-a

Treća metoda se zasniva na korištenju *event_listener*-a, odnosno pravljenju funkcija koje će mijenjati parametre u SIMULINK-u ukoliko se desi određena promjena u okviru GUI-a. Prednost ove metode je što GUI postaje neovisan od SIMULINK modela i što omogućava prikaz signala u SIMULINK-u. Osnovna mana ove metode je to što je nešto složenija, te što zahtijeva nešto više računarskih resursa u odnosu na prethodne metode.

Već je pomenuto da je od posebnog interesa prikaz akvizicije signala u realnom vremenu, što omogućavaju druga i treća metoda, pa će više pažnje biti posvećeno upravo njima. Za oba slučaja posmatrat će se GUI, kao na slici 2.9.




**Slika 2.9.** Izgled GUI-a

Korisnik ima mogućnost odabira kanala koji se koriste za akviziciju i generisanje signala, njihov naziv, opseg te način rada. Također, korisnik može odabrati i parametre akvizicije, amplitudu, frekvenciju i oblik izlaznog signala. Pri tome će se prikupljeni i generisani signali prikazivati na graficima GUI-a, a korisnik može odabrati između četiri načina

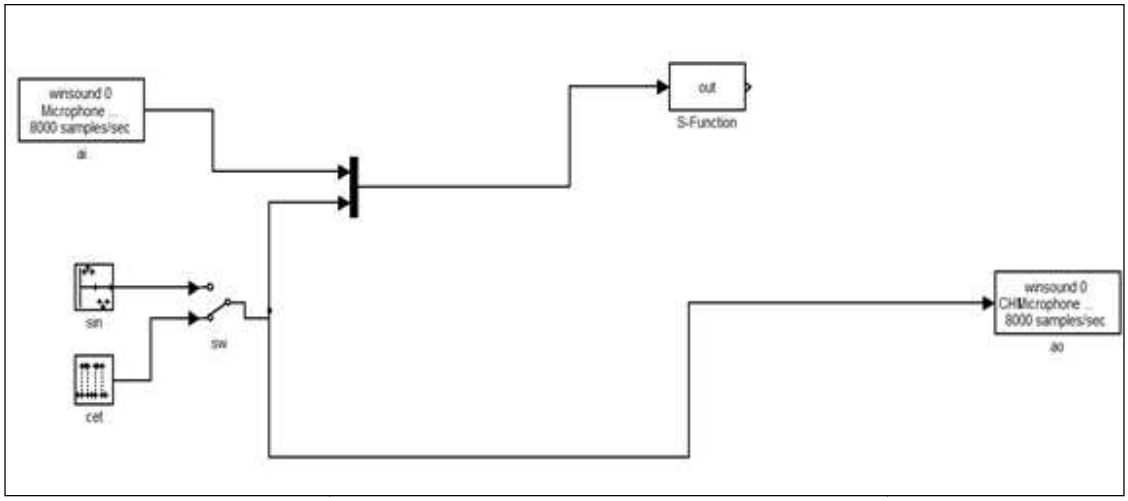
prikazivanja signala (*fixed*, *sweep*, *strip* i *scope*), te prikaz određenog broja tačaka. U nastavku će biti i prikazana funkcija *aqplot* koja je omogućila različite prikaze signala.

### 2.5.2. GUI za upravljanjem SIMULINK-om korištenjem s-funkcija

Već je navedeno da s-funkcije predstavljaju programske kodove, preko kojih se definiše funkcionalnost blokova u SIMULINK-u, pri čemu se pored kodova napisanih u MATLAB-u mogu koristiti i drugi programski jezici, poput C, C++ ili Fortrana. Programski kodovi će biti pisani isključivo u okviru MATLAB-a (m. funkcije). MATLAB nudi nekoliko gotovih blokova čije se funkcionalnosti djelomično mogu promijeniti (*Level 1* s-funkcije), ali zbog velikih nedostataka (nemogućnost promjene parametara simulacije tokom simulacije, nemogućnost korištenja višedimenzonalnih ulaza, ili izlaza) koristit će se *Level 2* s-funkcije.

Prilikom definisanja proizvoljnog bloka moguć je odabir funkcije, koja će definisati njegovu funkcionalnost,odabir parametara bloka koji će se proslijediti napisanoj funkciji, te odabir modula ukoliko se ne koristi MATLAB, a koji odgovarajući C, ili Fortran kôd prevodi u MATLAB. U toku simulacija s-funkciji se automatski prosljeđuju 4 parametra: *t*, *x*, *u* i *flag*. Parametar *t* predstavlja vrijeme simulacije, parameter *u* ulaz u blok, parametar *x* predstavlja vektor stanja bloka, dok parametar *flag* određuje funkciju koja će se pozvati u okviru s-funkcije (ukoliko su te funkcije definisane), te javlja grešku ukoliko se simulacija ne može izvršiti. U zavisnosti od vrijednosti flaga predstavljene cijelobrojnim brojem, pozvat će se jedna od ovih funkcija *mdlInitializeSizes, mdlUpdate*, te *mdlOutputs*. Uloga ovih funkcija će biti opisana na konkretnom primjeru s-funkcije u nastavku teksta.

Neka je dat model koji vrši jednokanalnu akviziciju i jednokanalno generisanje podataka (slika 2.10.). Već je pomenuto da će težište biti na kontroli modela u okviru GUI-a. Iz tog razloga model će se proširiti dodatnim blokom, čija je funkcionalnost opisana preko s-funkcije. Uloga ovog bloka je da vrši prikaz prikupljenih i generisanih signala na GUI-u. Korisniku se ostavlja mogućnost zaustavljanja akvizicije, te mijenjanja oblika, amplitude i frekvencije izlaznog signala. Tako će se koristiti model na slici 2.10.




**Slika 2.10.** Model za akviziciju i generisanje signala uz prikaz GUI-a

S-funkcija[3] koja opisuje funkcionalnost bloka je prikazana ispod:

```
function [sys,x0,str,ts] = out(t,x,u,flag,Ts)
   switch flag,
   case 0,
      [sys,x0,str,ts]=mdlInitializeSizes(Ts);
   case 2,
      sys=mdlUpdate(t,x,u);
   case 3,
      sys = mdlOutputs(t,x,u);
   case { 1, 4, 9 },
      sys = [0 0 0 0];
   otherwise
      error(['Unhandled flag = ',num2str(flag)]);
   end
function [sys,x0,str,ts]=mdlInitializeSizes(Ts)
      sizes = simsizes;
      sizes.NumContStates  = 0;
      sizes.NumDiscStates  = 1;
      sizes.NumOutputs     = 0;
      sizes.NumInputs      = 2;
      sizes.DirFeedthrough = 0;
      sizes.NumSampleTimes = 1;
      sys = simsizes(sizes);
      x0  = [0];
      str = [];
      ts  = [Ts 0];

function [sys]=mdlUpdate(t,x,u,Ts)
      sys=[];
      global tt;global aqdata;global gendata;
      tt=[tt;t];aqdata=[aqdata;u(1)];gendata=[gendata;u(2)];
      plott(tt,aqdata);
      plott1(tt,gendata);
```

[3] Akvizicija\out.m

```
function sys = mdlOutputs(t,x,u)
       sys=[];
```

U gornjem dijelu teksta navedeno je da će se u ovisnosti od parametra *flag*, pozivati jedna od odgovarajućih funkcija. Pored automatski proslijeđenih parametara, funkciji se dodaje i još jedan parametar, a to je korak simulacije, koji se postavlja u okviru GUI-a. Pored ovog funkcija se može proširiti i dodatnim parametrima, kao što su GUI na kojem se želi prikazati prikupljeni signal, ili varijable u koju je potrebno spremiti prikupljene podatke. Pošto je ostvarena veza između modela i GUI-a, pa svaki model se odnosi na pojedinačni GUI, dodatni parametri nisu bili potrebni. Izlazi iz ove funkcije su *sys*, *x0*, *str* i *ts*. Ovi parametri su potrebni za rad sa s-funkcijama, pa se moraju pozivati, iako često nisu od pretjeranog značaja. Pri tome parametar *sys* predstavlja opšti izlazni parametar, a njegova vrijednost ovisi od parametra *flag*. *x0* predstavlja vektor početnih stanja bloka, vektor *ts* predstavlja vektor koraka simulacije, dok se parametar *str* po pravilu predstavlja prazan vektor, a namijenjen je budućim verzijama MATLAB-a [8]. Prva funkcija koja će se pozvati je *mdlInitializeSizes*. Ova funkcija se poziva prilikom svakog koraka simulacije, a služi sa inicijalizaciju parametara bloka.

```
function [sys,x0,str,ts]=mdlInitializeSizes(Ts)
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 1;
sizes.NumOutputs     = 0;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [0];
str = [];
ts  = [Ts 0];
```

Po pravilu parametar *sizes* se postavlja na vrijednost *simsizes*. Ostali parametri su:

**NumContStates** i **NumDiscStates-** predstavlja broj kontinualnih, odnosno diskretnih stanja modela. Ovi parametri se postavljaju u ovisnosti od načina odabira koraka simulacije. Za sve modele akvizicije korišten je *disrete solver,* pa model neće imati nikakvih kontinualnih stanja, odnosno imat će samo jedno diskretno stanje

**NumOutputs** i **NumInputs**- predstavljaju broj ulaza, odnosno izlaza bloka. Kako će svi prikupljeni, odnosno generisani signali preko mux-a dovoditi na ulaz, potreban je samo jedan ulaz, a izlaznih signala neće biti.

**DirFeedthrough**- predstavlja ovisnost izlaza od ulaza. Kako nema nikakvih izlaza, parametar je postavljen na nulu.

**NumSampleTimes**- predstavlja vrijeme uzorkovanja bloka. Blok će imati samo jedno vrijeme uzorkovanja koje odgovara prethodno odabranoj frekvenciji uzorkovanja. Pored inicijalizacije parametara, definisani su i izlazni parametri funkcije.

Pored funkcije *mdlInitializeSizes*, prilikom svakog koraka simulacije poziva se i funkcija *mdlUpdate*.

```
function [sys]=mdlUpdate(t,x,u,Ts)
    sys=[];
    global tt;global aqdata;global gendata;
    tt=[tt;t];aqdata=[aqdata;u(1)];gendata=[gendata;u(2)];
    plott(tt,aqdata);
    plott1(tt,gendata);
```

Unutar ove funkcije se prikupljeni i generisani podaci smještaju u varijable, zajedno sa vremenom simulacije, te se vrši pozivanje funkcija *plott* i *plott1*, koje će iscrtavati prikupljene, odnosno generisane signale na GUI-u. Također, unutar ove dvije funkcije će se i mijenjati parametri modela u zavisnosti od zadanih vrijednosti. Kako blok nema izlaza, funkciju *mdlOutputs* nije potrebno definisati.

Pored funkcija koje je neophodno definisati za ispravan rad bloka, potrebno je i definisati funkcije *plott* i *plott1* koje će prikazivati signale na GUI-u. Funkcijama se proslijeđuju dva parametra. Za obje funkcije prvi parametar predstavlja vrijeme simulacije, dok za funkciju *plott* drugi parametar predstavlja prikupljene, a za funkciju *plott1* generisane podatke. Obje funkcije će biti pojašnjene u nastavku, obzirom da su se koristile i za pristup za event_listenerima.

Po prethodno opisanoj proceduri se može realizirati veza između GUI-a i SIMULINK-a. Pristup korištenja s-funkcija nije previše zahtjevan, te zanemarivo utiče na strukturu modela (dodavanjem jednog bloka). Pri tome je pisanje s-funkcija jako jednostavno, ukoliko se slijedi opisana procedura. Također, pristup se može proširiti i na druge

programske jezike, što je svakako prednost. U nastavku će biti opisana procedura korištenjem event_listenera.

### 2.5.3. GUI za upravljenje SIMULINK-om korištenjem event_listenera

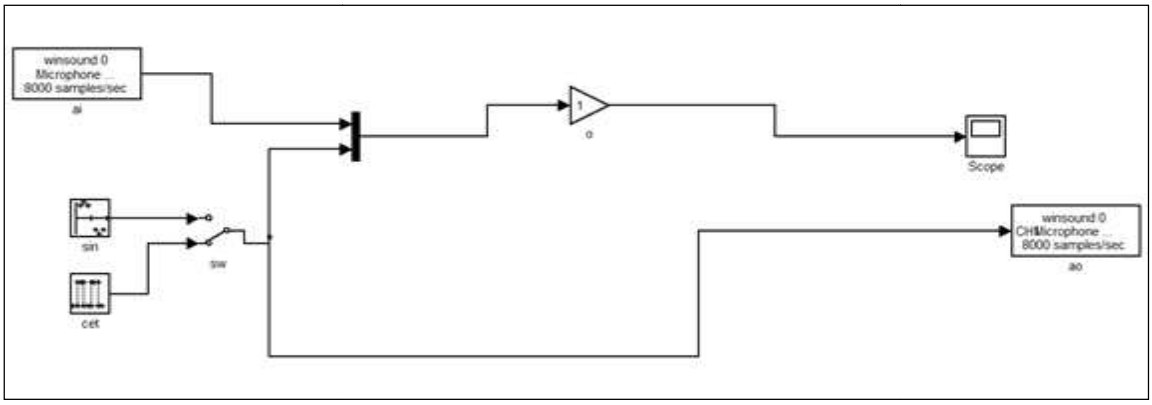
Osnovna ideja ove metode je pisanje funkcije koja će se vezati za određeni blok, te pozivati prilikom svakog koraka simulacije. Pri tome će se vezivanje funkcije uz blok definisati prilikom samog pokretanja modela, a potom će se definisati uz koji blok se veže, te trenutak pozivanja ove funkcije. Na ovaj način svaku promjena unutar GUI-a će se direktno odraziti i na sam model, a sve promjene unutar modela će se prikazati na GUI-u. Prednost ovakvog pristupa je to što se ovakva funkcija može definisati u okviru samog GUI-a, pa nije potrebno pisanje dodatnih funkcija. Neka je dat GUI kao na slici 2.11. Prije pokretanja akvizicije potrebno je pozvati naredbu *set_param* sa parametrima prikazanim ispod.

```
set_param(name,'StartFcn','localAddEventListener');
```

Ovim se uz model, kao na slici 2.11. definiše funkcija *localAddEventListener*, koja se poziva prilikom pokretanja modela. Izgled ove funkcije je:

```
function eventdat=localAddEventListener
global name;
eventdat=add_exec_event_listener(strcat(name,'/o'),'PostOutput',@localEv
entListener);
```

Ovim se datom bloku u SIMULINK-u dodjeljuje odgovarajuća funkcija, nazvana *localEventListener*, te se definiše trenutak pozivanja postavljanjem parametra *PostOutput.* Pored ovog parametra mogu se koristiti i parametri: *PreOutput*, *PreDerivative, PostDerivative, PreUpdate* i *PostUpdate*, a koji definišu trenutak pozivanja *event_listenera.* Pri tome se u toku simulacije prvo računaju izlazi iz bloka (*Output*), potom se stanja bloka mijenjaju (*Update*), a potom vrši njihova derivacija (*Derivative*). Na osnovu ovoga se može izvršiti adekvatan odabir gore navedenih parametara.




**Slika 2.11.** Korišteni model za akviziciju i generisanje podataka

Da bi ovaj program radio ispravno potrebno je još definisati funkciju *localEventListener*.

```
function localEventListener(block,eventdata)
xs1=block.OutputPort(1).Data;
global tt;tt=[tt;block.CurrentTime];
global aqdata;aqdata=[aqdata;xs1(1,1)];
global gendata;gendata=[gendata;xs1(2,1)];
plott(t,aqdata);
plott1(t,gendata);
```

U okviru funkcije se pozivaju funkcije *plott* i *plott1*, koje iscrtavaju prikupljene, odnosno generisane signale, te se podaci o ovim signalim smještaju u odgovarajuće varijable. Iako u većini programskih jezika nije preporučeno korištenje globalnih varijabli, pa tako i u MATLAB-u, ipak u ovom slučaju one olakšavaju spremanje podataka, obzirom da je alternativa snimanje i osvježvanja podataka u MATLAB-ovom radnom prostoru (eng. *workspace*), ali je znatno komplikovanije od ovog pristupa.

Izgled funkcije *plott* je:

```
function plott(t,u)
       global lc_hand;
       i=str2num(get(lc_hand.rsm1,'String'));
xg=str2num(get(lc_hand.smpl,'String'))/str2num(get(lc_hand.srate,'String
'));
       T=xg/10;
    if(get(lc_hand.fix1,'Value'))
       xn='Fix';
       yl=1;
```

```
        xl=xg;
    elseif(get(lc_hand.sweep1,'Value'))
        xn='Sweep';
        xl=[];yl=[];
    elseif(get(lc_hand.scope1,'Value'))
        xn='Scope';
        xl=[];yl=[];
    elseif(get(lc_hand.strip1,'Value'))
        xn='Strip';
        xl=[];yl=[];
    end
        aqplot(t,u,T,i,lc_hand.axes1,xn,xl,yl);
```

Na početku je definisana globalna varijabla *lc_hand*, koja sadrži elemente GUI-a i preko nje se može pristupiti svakom elementu GUI-a pojedinačno. Varijabla *i* predstavlja broj tačaka koje korisnik želi da prikaže, varijabla *xg* predstavlja dužinu trajanja simulacije, dok *T* predstavlja vremenski inteval prikazivanja na grafiku. *xl* i *yl* predstavljaju varijable za granice x-ose i y-ose (ovo je potrebno definisati samo za slučaj *fix* prikaza). Pored toga korisnik je birao način prikaza, a mogao je birati između prikaza *fix*, *sweep*, *strip* i *scope*. Prikaz signala i njihov način prikaza je realiziran funkcijom *aqplot*[4]. Izgled i objašnjenje ove funkcije su dati u nastavku.

```
function aqplot(t,d,T,n,fig,x,xl,yl)
    if nargin ==8
        if strcmpi(x,'Fix')
            plot(fig,t(1:n:end,1),d(1:n:end,1));
           xlim(fig,[0,xl]);ylim(fig,[-yl,yl]);
           grid (fig,'on');
           xlabel(fig,'Vrijeme [sec]');ylabel(fig,'Napon [V]');
        elseif(strcmpi(x,'Scope'))
            plot(fig,t(1:n:end,1),d(1:n:end,1));
            grid (fig,'on');
            xlabel(fig,'Vrijeme [sec]');ylabel(fig,'Napon [V]');
         if(t(end)<T)
            xlim(fig,[0 T]);
         else
            xlim(fig,[t(end)-T t(end)]);
         end
        elseif(strcmpi(x,'Sweep'))
            k=floor(t(end)/T);
            plot(fig,t(1:n:end,1),d(1:n:end,1));
            xlim(fig,[k*T (k+1)*T]);
            grid (fig,'on');
            xlabel(fig,'Vrijeme [sec]');ylabel(fig,'Napon [V]');
```

[4] Akvizicija\aqplot.m

```
        elseif(strcmpi(x,'Strip'))
            k=floor(t(end)/T);
            rx=(k+1)*T;
         if(floor(t(end)/rx)>=1)
            plot(fig,t(1:n:end,1),d(1:n:end,1));
            grid (fig,'on');
            xlabel(fig,'Vrijeme [sec]');ylabel(fig,'Napon [V]');
         end
            xlim(fig,[k*T (k+1)*T]);
        elseif (strcmpi(x,'Tscope'))
            plot(fig,t(1:n:end,1),d(1:n:end,1));
            grid (fig,'on');
            xlabel(fig,'Vrijeme [sec]');ylabel(fig,'Napon [V]');
        end
    elseif nargin==6
        if(strcmpi(x,'Scope'))
            plot(fig,t(1:n:end,1),d(1:n:end,1));
            grid (fig,'on');
            xlabel(fig,'Vrijeme [sec]');ylabel(fig,'Napon [V]');
         if(t(end)<T)
            xlim(fig,[0 T]);
         else
            xlim(fig,[t(end)-T t(end)]);
         end
        elseif(strcmpi(x,'Sweep'))
            k=floor(t(end)/T);
            plot(fig,t(1:n:end,1),d(1:n:end,1));
            xlim(fig,[k*T (k+1)*T]);
            grid (fig,'on');
            xlabel(fig,'Vrijeme [sec]');ylabel(fig,'Napon [V]');
        elseif(strcmpi(x,'Strip'))
            k=floor(t(end)/T);
            rx=k*T;
            gc=get(gca,'xlim');
         if(k~=gc(2) && k~=0)
            plot(fig,t(1:n:end,1),d(1:n:end,1));
            grid (fig,'on');
            xlabel(fig,'Vrijeme [sec]');ylabel(fig,'Napon [V]');
         end
         if k==0
            xlim(fig,[0 T])
         else
            xlim(fig,[(k-1)*T k*T]);
         end
        elseif (strcmpi(x,'Tscope'))
            plot(fig,t(1:n:end,1),d(1:n:end,1));
            grid (fig,'on');
            xlabel(fig,'Vrijeme [sec]');ylabel(fig,'Napon [V]');
        end
    end
```

```
end
```

Funkcija, kao parametre prima vrijeme i podatke koji se žele prikazati, vremenski interval koji se prikazuje, broj tačaka koji se prikazuje, grafik na koji se podaci prikazuju, način prikaza i granice za x-osu i y-osu.

Zbog potrebe da se signali koji su generišu mogu mijenjati u okviru same simulacije, funkcija *plott1* je morala biti drugačija nego funkcija *plott*. Izgled ove funkcije je :

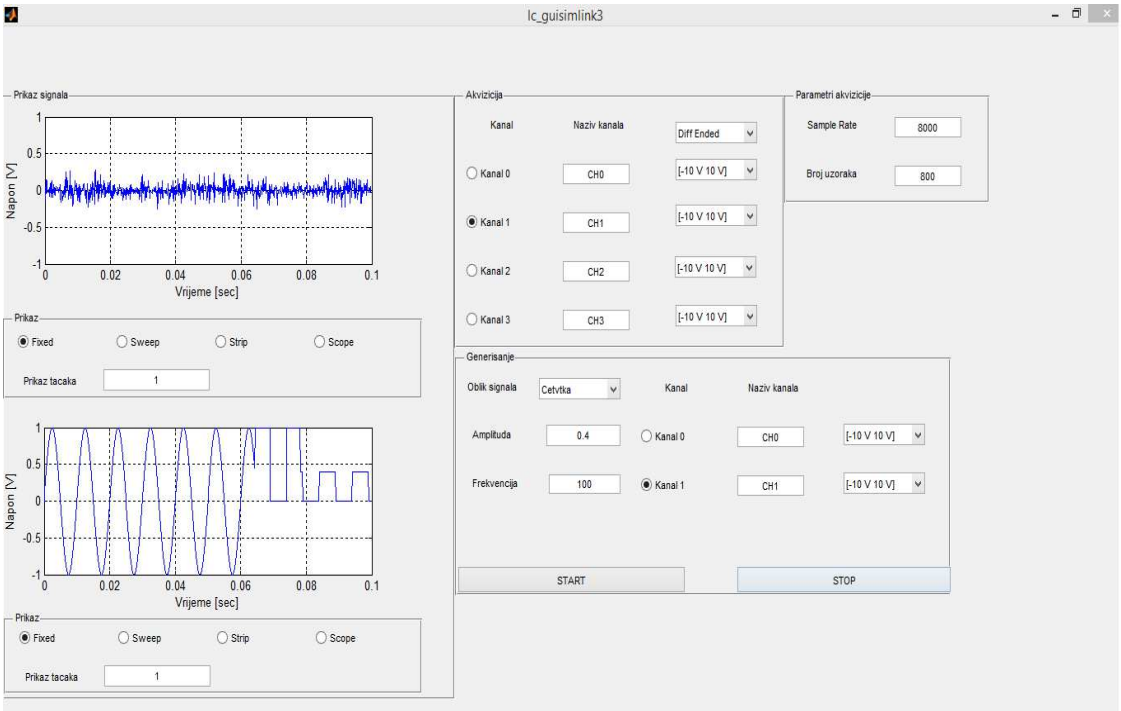
```
function plott1(t,u)
       global lc_hand;global lc_mdl;
   if get(lc_hand.sig,'Value')==1
       set_param(strcat(lc_mdl,'/sw'),'sw','1');

set_param(strcat(lc_mdl,'/sin'),'Amplitude',get(lc_hand.amp,'String'));

set_param(strcat(lc_mdl,'/sin'),'Samples',num2str(str2num(get_param(strc
at(lc_mdl,'/ai'),'SampleRate'))/str2num(get(lc_hand.frek,'String'))));
   elseif get(lc_hand.sig,'Value')==2
       set_param(strcat(lc_mdl,'/sw'),'sw','0');

set_param(strcat(lc_mdl,'/cet'),'Amplitude',get(lc_hand.amp,'String'));
       set_param(strcat(lc_mdl,'/cet'),'PulseWidth','1');

set_param(strcat(lc_mdl,'/cet'),'Period',num2str(str2num(get_param(strca
t(lc_mdl,'/ai'),'SampleRate'))/str2num(get(lc_hand.frek,'String'))));

set_param(strcat(lc_mdl,'/cet'),'PulseWidth',num2str(0.5*str2num(get_par
am(strcat(lc_mdl,'/ai'),'SampleRate'))/str2num(get(lc_hand.frek,'String'
))));
   end
       i=str2num(get(lc_hand.rsm2,'String'));

xg=str2num(get(lc_hand.smpl,'String'))/str2num(get(lc_hand.srate,'String
'));
       T=xg/10;
   if(get(lc_hand.fix2,'Value'))
       xn='Fix';
       yl=1;xl=xg;
   elseif(get(lc_hand.sweep2,'Value'))
       xn='Sweep';
       xl=[];yl=[];
   elseif(get(lc_hand.scope2,'Value'))
       xn='Scope';
       xl=[];yl=[];
   elseif(get(lc_hand.strip2,'Value'))
       xn='Strip';
       xl=[];yl=[];
   end
       aqplot(t,u,T,i,lc_hand.axes2,xn,xl,yl);
```

Na sličan način, kao i u prethodnoj funkciji su definisane varijable za pristup elementima GUI-a, prikaz broja tačaka, granice za x-osu i y-osu. Pored ovih varijabli definisana je i globalna varijabla *lc_mdl* koja je sadržavala ime SIMULINK modela. Ova varijabla je potrebna kako bi se parametri modela mogli mijenjati u toku simulacije, što je realizirano kodom prikazanim ispod.

```
    if get(lc_hand.sig,'Value')==1
       set_param(strcat(lc_mdl,'/sw'),'sw','1');

set_param(strcat(lc_mdl,'/sin'),'Amplitude',get(lc_hand.amp,'String'));

set_param(strcat(lc_mdl,'/sin'),'Samples',num2str(str2num(get_param(strc
at(lc_mdl,'/ai'),'SampleRate'))/str2num(get(lc_hand.frek,'String'))));
    elseif get(lc_hand.sig,'Value')==2
       set_param(strcat(lc_mdl,'/sw'),'sw','0');

set_param(strcat(lc_mdl,'/cet'),'Amplitude',get(lc_hand.amp,'String'));
       set_param(strcat(lc_mdl,'/cet'),'PulseWidth','1');

set_param(strcat(lc_mdl,'/cet'),'Period',num2str(str2num(get_param(strca
t(lc_mdl,'/ai'),'SampleRate'))/str2num(get(lc_hand.frek,'String'))));

set_param(strcat(lc_mdl,'/cet'),'PulseWidth',num2str(0.5*str2num(get_par
am(strcat(lc_mdl,'/ai'),'SampleRate'))/str2num(get(lc_hand.frek,'String'
))));
    end
```

Naredba *set_param* je već pojašnjena, pa će samo ukratko biti pojašenja ideja. Prvo je biran oblik signala koji se generiše (sinus, ili četvrtka), a potom se generisanom signalu definisala amplituda (predstavljena parametrom *Amplitude*), te frekvencija generisanog signala. U slučaju sinusnog signala, frekvencija se postavlja preko parametra *Samples*, a u slučaju četvrtke frekvencija se postavlja preko parametra *Period.* Pokretanjem ovog GUI-a, dobija se slika 2.12.

Ovaj GUI je bio osnova za složeniji GUI, koji će biti prikazan u nastavku,na slici 5.1., a ovdje će biti navedene mogućnosti tog GUI-a.




**Slika 2.12.** GUI za akviziciju i generisanje podataka primjenom SIMULINK-a

Korisnik je mogao birati:

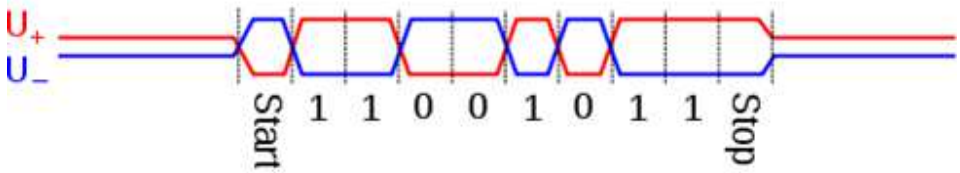
- frekvenciju uzorkovanja, trajanje simulacije definisanjem vremena simulacije, broja uzoraka koji će se prikupiti, ili trajanjem simulacije u realnom vremenu.
- kanale preko kojih se vrši akvizicija, njihova imena, njihov opseg i način rada, te sa kojih kanala će se signali prikazivati.
- kanale preko kojih se vrši generisanje, njihova imena, njihov opseg, te sa kojih kanala će se signali prikazivati.
- amplitudu, frekvenciju i oblik signala, pri čemu je mogao birati između sinusnog signala, signala četvrtke, te proizvoljno definisanog signala.
- način pokretanja simulacije (*continous*, *trigger* i *periodic*).
- način prikaza signala (*fix*, *sweep*, *strip* i *scope*), broj tačaka koji se prikazuju, te vremenski interval prikazivanja signala.
- ime varijable u koju će se svi prikupljeni i generisani podaci snimiti (pri tome je snimljena svaka varijabla za ponovo pokretanje modela).

# 3. Realizacija serijske komunikacije koristeći MATLAB

Serijska komunikacija predstavlja proces slanja podataka bit po bit, sekvencijalno, koristeći komunikacijski kanal. Danas serijska komunikacija predstavlja najrašireniji vid komunikacije posebno korišten u aplikacijama sa mikrokontrolerima. Iako postoji više tipova serijske komunikacije, kao što su: RS-232, RS-422, $I^2C$ i razni drugi u nastavku će se posmatrati standard RS-485.

## 3.1. RS-485 komunikacija

RS-485 standard komunikacije (često se koristi i naziv TIA-485) omogućava komunkaciju između više uređaja, za razliku od RS-232 standarda koji omogućava komunikaciju samo između dva uređaja. Pored ove prednosti RS-485 omogućava i prenos podataka na većim dužinama u poređenju sa ostalim standardima serijske komunikacije. Prilikom komunikacije se koristi diferencijalni signal, sa definisanim nivoima logičke 0 i logičke 1. Nivo logičke nule je veći od + 200 mV, dok je nivo logičke jedinice manji od -200 mV. Minimalni napon koji se koristi je 1.5 V bipolarno, dok je maksimalni 6 V bipolarno.

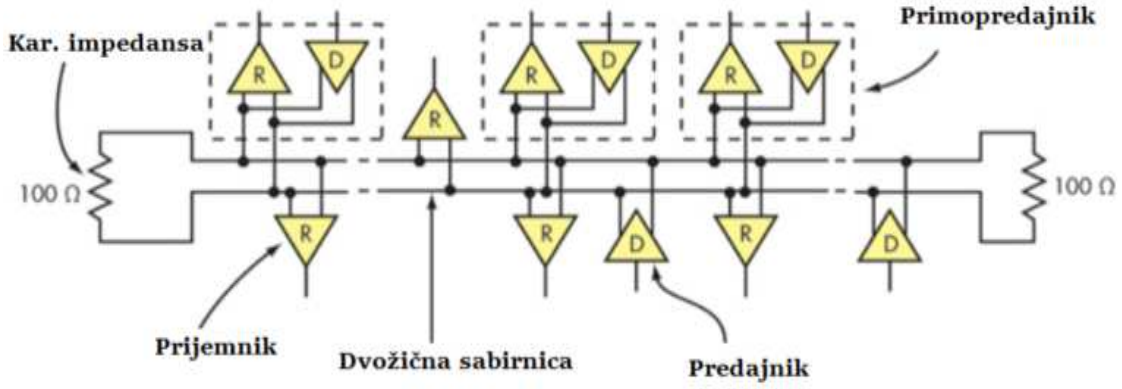



**Slika 3.1.** Signali RS-485 komunikacije

Za komunikaciju su potrebne dvije žice, mada se ponekad koriste tri, ili četiri. Pri tome se žice posmatraju kao vodovi, koji se zatvaraju karakterističnom impedansom od 100 Ω, ili 120 Ω. Za komunikaciju nisu definisani specijalni konektori, pa se u većini slučajeva

koriste oni za RS-232 komunikaciju. Brzina prenosa podataka ovisi od dužine kabla. Naprimjer, brzina prenosa, za kablove dužine oko 20 m, iznosi oko 5 Mbits/s, dok za kablove dužine oko 1200 m, iznosi oko 100 kbits/s.

RS-485 omogućava komunikaciju do 32 prijemnika i 32 predajnika. Pri tome su svi povezani na istu sabirnicu zatvorenu karakterističnom impedansom.




**Slika 3.2.** Prikaz RS-485 komunikacije

## 3.2. USB komunikacija

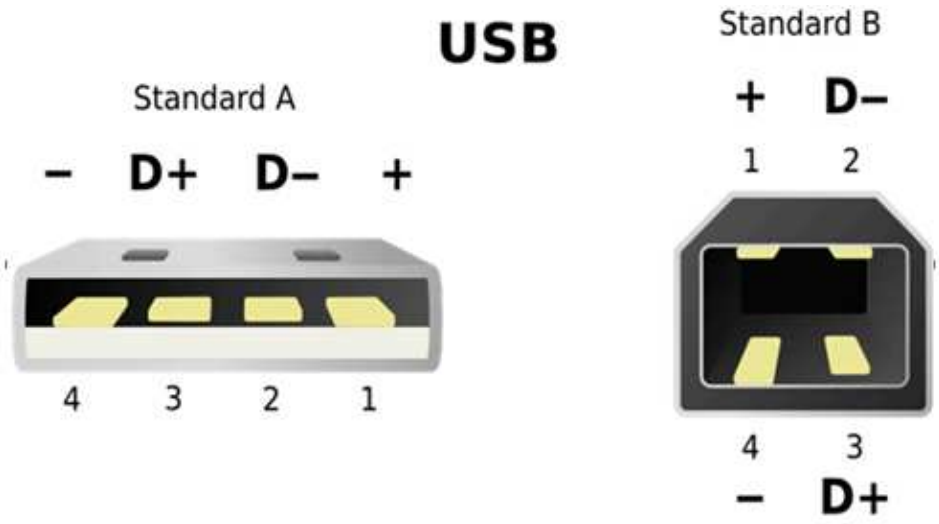
USB (eng. *Universal Serial Bus*) predstavlja industrijski standard koji definiše komunikacijske protokole, kablove i konektore, a koji se koriste za komunikaciju, konekciju, ili napajanje uređaja. Standard se razvio tokom sredine 90-ih godina prošlog stoljeća od strane kompanija, kao što su Microsoft, Intel, Compaq, itd. Danas je USB najčešće korišten način za povezivanje hardverskih dijelova, kao što su tastature, miševi, ili web-kamere na računar i sve više zamjenjuje već pomenute RS standarde, prvenstveno zbog veće brzine prenosa podataka i većeg broja učesnika u komunikaciji. Danas postoje različiti USB standardi, kao što su USB 1.0, USB 2.0, USB 3.0 i USB 3.1., dok u novije vrijeme razvija se i bežična USB komunikacija. Danas se pod pojmom USB uglavnom označava flash memorija, koja ima USB port za komunikaciju sa PC-om.

**Slika 3.3.** Prikaz USB sabirnice i USB porta na PC-u

USB komunikacija se sastoji od jednog host sistema i više perifernih uređaja za komunikaciju povezanih sa host sistemom, pri čemu su ovi uređaji koriste zvjezdastu struktutru. Tačke u kojima su spojeni host sistem i periferni uređaj (eng. *endpoint*) su od posebne važnosti, jer se prenos podataka odigrava od jedne tačke do druge. Pri tome je moguće spajanje do 127 uređaja na jedan host sistem. Komunikaciju pokreće host sistem, slanjem odgovaraćujeg znaka. Nakon što uređaj primi znak, dešava se slanje podataka od host sistema ka perifernim uređajima, ili slanje ka host sistemu u ovisnosti od smjera komunikacije. Pri tome se podaci šalju brzinama od 12 Mbps za USB 1.0 verzije, odnosno 480 Mbps za USB 2.0 verzije. Za razliku od RS-232 standarda koji isključivo predstavlja serijski način komunikacije, USB standard omogućava više različitih načina komunikacije, ali u nastavku oni neće biti pojašnjeni.

USB se na računar povezuje preko USB konektora, a danas su najrasprostranjenije dvije varijante ovih konektora.

Prva varijanta, naziva se varijanta A i sastoji se od četiri pina, dok druga varijanta, varijanta B se sastoji od 5 pinova. Za novije verzije USB komunikacije (USB 3.0) konektori se sastoje od više pinova.




**Slika 3.4.** Prikaz USB konektora varijanta A i varijanta B

Varijante su dosta slične. Prvi pin predstavlja napon napajanja USB-a, koji iznosi +5 V, sa tolerancijom od 5 %, pin broj 4 predstavlja uzemljenje, dok pinovi 2 i 3 predstavljaju diferencijalne pinove za slanje podataka. Pored ovih pinova tip B ima i peti pin, koji se često ne označava.

Obzirom na skoro svaki hardver ima USB port sa vezu sa PC-om, od posebnog interesa je razvoj softverskih modula koji će ovo omogućiti.

### 3.3. Klasa za serijsku komunikaciju u MATLAB-u

Serijska komunikacija u MATLAB-u započinje kreiranjem odgovarajuće varijable tipa *serial*. Ova metoda se poziva na na sljedeći način

```
port=serial('COM1');fopen(port);
```

Ovim se kreira objekat koji će imati pristup odgovarajućem portu (COM) na PC-u. Naredbom *fopen* ovaj port se otvara i sada se može pristupiti podacima ovog COM-a, slati, ili primati podatke na ovaj COM. Pored definisanja odgovarajućeg COM-a, često je potrebno definisati i ostale parametre neophodne za serijsku komunikaciju. Ovdje će se pomenuti oni najbitniji:

**Baud Rate-** određuje brzinu komunikacije, odnosno koliko maksimalno bitova u sekundi se može primiti, ili poslati.

**InputBufferSize-** predstavlja broj bitova koji se sprema u buffer prilikom slanja podataka.

**OutputBufferSize-** predstavlja broj bitova koji se sprema u buffer prilikom prijema podataka.

**DataBits-** broj bitova koji se prenosi.

**Timeout-** ovo je potrebno definisati u slučaju greške prilikom komunikacije, a predstavlja vrijeme u kojem se očekuje prijem podataka.

U nastavku će biti prikazan m.fajl kojim se može vršiti slanje podataka u MATLAB-u.

```
    s1=serial('COM1','BaudRate',9600);
    c=63;
    s1.InputBufferSize=1;
    fopen(s1)
    s1.Terminator='';
    s1.Timeout=1;
    str=strcat('#A','N');
for i=1:length(str)
    s1.RequestToSend='off';
    fprintf(s1,str(i))
end
    fwrite(s1,68)
    s1.RequestToSend='on';
    fclose(s1);delete(s1);clear(s1);
```

Na početku se kreira objekat tipa *serial*, definiše brzina komunikacija i broj bita koji se sprema u buffer prilikom slanja podataka. Potom se definiše *Timeout*, te *Terminator* se definiše, kao prazan znak, kako ne bi došlo do neželjenih podataka. Potom se definišu podaci koji se šalju u obliku stringa, te se naredbom *fprintf* upisuju podaci, znak po znak, preko COM-a. Potom se naredbom *fwrite* u COM upisuje određena brojna vrijednost, te na samom kraju podaci šalju na COM, a objekat se zatvara i briše iz radnog prostota. U ovom slučaju poslani podaci ne bi imali nikakvog značenja za uređaj povezan na COM, ukoliko uređaj ne bi bio definisan za prepoznavanje ovakvih karaktera.

Za prijem podataka se može koristiti sljedeći m.fajl:

```
    s1=serial('COM1','BaudRate',9600);
    s1.InputBufferSize=1;
```

```
    fopen(s1)
    s1.Terminator='';
    s1.Timeout=1;
    x=fscanf(s1);
    fclose(s1);delete(s1);clear s1;
```

Po već opisanoj proceduri definiše se odgovarajući objekat. Naredbom *fscanf* se čitaju podaci iz COM-a, te se dodijeljuju varijabli *x*, te se na kraju COM zatvara, a objekat briše. Treba reći da MATLAB ne može pristupati portovima, koje koriste drugi programi. Sada je moguće razviti i klasu[5] za serijsku komunikaciju u okviru MATLAB-a koja će omogućiti slanje, odnosno prijem podataka preko serijskih portova PC-a.

Atributi klase će biti serijski port koji se koristi za komunikaciju, *BaudRate*, *Timeout* i *InputBufferSize*. Klasa će imati *konstruktor*, metodu za slanje i metodu za prijem podataka. Pored ovih metoda klasa će imati i metodu *help* koja daje osnovne informacije o klasi. Metode ove klase su zasnovane na već opisanoj proceduri slanja, odnosno prijema podataka.

```
classdef commun
    %   Klasa za serijsku komunikaciju
    %   Atributi: COM,BaudRate,DataBits,BufferSize
    %   Metode:primi i posalji;
    properties
            com;
            baudrate;
            buffersize;
            timeout;
    end
    methods

        function coms=commun
            coms.com='COM9';
            coms.baudrate=9600;
            coms.buffersize=15;
            coms.timeout=1;
            commun.com='COM9';
            commun.baudrate=9600;
            commun.buffersize=52;
            commun.timeout=1;
        end

        function posalji(obj,a)
            s=serial(obj.com,'BaudRate',obj.baudrate);
            s.InputBufferSize=obj.buffersize;
            s.Timeout=obj.timeout;
```

[5] Serial_com\commun.m

```
          fopen(s);
          flushinput(s);
          fwrite(s,a);
          fclose(s);
          delete s;
          clear s;
        end
        function x=primi(obj,n)
          s=serial(obj.com,'BaudRate',obj.baudrate);
          s.InputBufferSize=obj.buffersize;
          s.Terminator='';
          s.Timeout=1;
          fopen(s);
          x=fread(s,n);
          fclose(s);
          delete s;
          clear s;
        end
        function help(obj)
          disp('Klasa- za ser. komunikaciju');
          disp('Atributi COM- naziv porta, BaudRate- brzina komunikacije,
BufferSize-broj bita u bufferu');
          disp(' Timeout- vrijeme za citanje podataka');
          disp('Metode- primi (broj bitova koji se cita) prima podatke,
posalji (podaci koji se salju) salje podatke');
        end
    end

end
```

## 3.4. Arduino razvojno okruženje


**Slika 3.5.** Arduino Uno razvojno okruženje

Arduino je jednostavno razvojno okruženje talijanskog proizvođača Smart Projects. Okruženje se sastoji od jednog mikroprocesora; 8-bitnog ATMEL AVR procesora, ili 32-bitnog ARM procesora. Okruženje sadrži 14 digitalnih pinova, koji se mogu koristiti, kao izlazni, ili ulazni pinovi, te 6 pinova koje služe, kao analogni ulazi. Arduino okruženje sadrži i USB port, preko kojeg se mikroprocesor može napajati sa 5 V, pa nije potreban eksterni izvor napajanja. Arduino okruženje je relativno novo (prva Arduino okruženja pojavila su se 2005. godine ), ali za kratko vrijeme ova okruženja su postala popularna, te se sve više koriste. Ovo je posljedica prednosti Arduino okruženja, koje uključuju nisku cijenu, jednostavno programiranje mikroprocesora, napajanje preko USB-a, te je okruženje veoma jednostavno za upotrebu. Za programiranje se koristi Arduinov softver, a mikroprocesor se može programirati i u C, ili C++. Kako je korišteno Arduino Uno razvojno okruženje u nastavku će biti par riječi o ovom okruženju.

### 3.4.1. Arduino Uno razvojno okruženje

Arduino Uno okruženje je posebna verzija razvojnog okruženja, koja koristi 8-bitni ATmega 328 procesor. Osnovna razlika ovog razvojnog okruženja u odnosu na ostale Arduino sisteme (Arduino Nano, ili Arduino Micro) je u tome što ovo okruženje koristi Atmega16U2 čip, umjesto FTDI čipa. U tabeli 3.1. su date osnovne karakteristike Arduino Uno razvojnog okruženja:

| | |
|---|---|
| Mikroprocesor | ATmega328 |
| Napajanje | 5V |
| Ulazni napon (preporučeni) | 7-12V |
| Ulazni napon (u granicama) | 6-20V |
| Digitalni ulazi/izlazi | 14 (6 izlaza služi za generisanje PWM signala) |
| Analogni ulazi | 6 |
| Vrijednost DC struje za ulaze/izlaze | 40 mA |
| Vrijednost DC struje za 3.3V | 50 mA |
| Flash memorija | 32 KB (ATmega328); 0.5 KB je zauzeto od strane *bootloader* |
| SRAM | 2 KB (ATmega328) |
| EEPROM | 1 KB (ATmega328) |

| | |
|---|---|
| Frekvencija oscilatora | 16 MHz |

**Tabela 3.1.** Karakteristike Arduino Uno razvojnog okruženja [13]

Već je pomenuto ovo okruženje ima 14 digitalnih pinova. Pored toga što imaju opću namjenu, svaki pin ima i specijalnu funkciju:

- pinovi 0 i 1 služe za serijsku komunikaciju, odnosno pin 0 za slanje podataka (TX), a pin 1 za primanje podataka (RX).
- pinovi 2 i 3 se mogu koristiti za eksterne prekide, odnosno mogu se aktivirati pri dovođenju logičke nule, ili jedinice, ili dovođenja rastuće, odnosno negativne ivice.
- pinovi 3, 5, 6, 9, 10 i 11 služe za generisanje analognog izlaza, odnosno preko njih se može generisati PWM signal.
- na pin 13 je spojena LED dioda, koja se može kontrolisati.

Prikaz Arduino Uno razvojnog okruženja je na slici 3.5., a u nastavku će biti par riječi o akviziciji i serijskoj komunikaciji preko ovog okruženja.

### 3.4.2. Akvizicija podataka preko Arduino Uno razvojnog okruženja

Akvizicija podataka je omogućena putem 6 analognih ulaza, koje Arduino Uno posjeduje. Pri tome je moguće mjeriti napone u opsegu od 0 do 5 V. Odgovarajuća vrijednost napona se konvertuje u binarnu vrijednost u opsegu od 0 do 1023, preko A/D konvertora u okviru Arduino Una.

Sa softverskog aspekta osnova akvizicije jeste naredba *analogRead()*. Ova naredba sa definisanog analognog ulaza izmjerenu vrijednost pretvara u odgovarajuću vrijednost u opsegu 0-1023, te kao rezultat upravo vraća ovu vrijednost. Jednostavan kôd kojim se može vršti akvizicija podataka u Arduinu, koristeći pin A0 je prikazan ispod:

```
int anpin = A0;
int vrijednost = 0;

void setup() {
}

void loop() {
```

```
  vrijednost = analogRead(anpin);
}
```

### 3.4.3. Serijska komunikacija na Arduino Uno razvojnom okruženju

Već je pomenuto da se pinovi 0 i 1 koriste za prijem, odnosno slanje podataka. Pri tome je omogućena standardna TTL komunikacija, mada okruženje ima ugrađen 16U2 čip, pa je moguć prenos podataka preko USB porta, bez pomoćnih kola za prilagođenje.

Za serijsku komunikaciju Arduino nudi i odgovarajuće biblioteke, te odgovarajuće funkcije, preko kojih se mogu slati odnosno primati podaci.

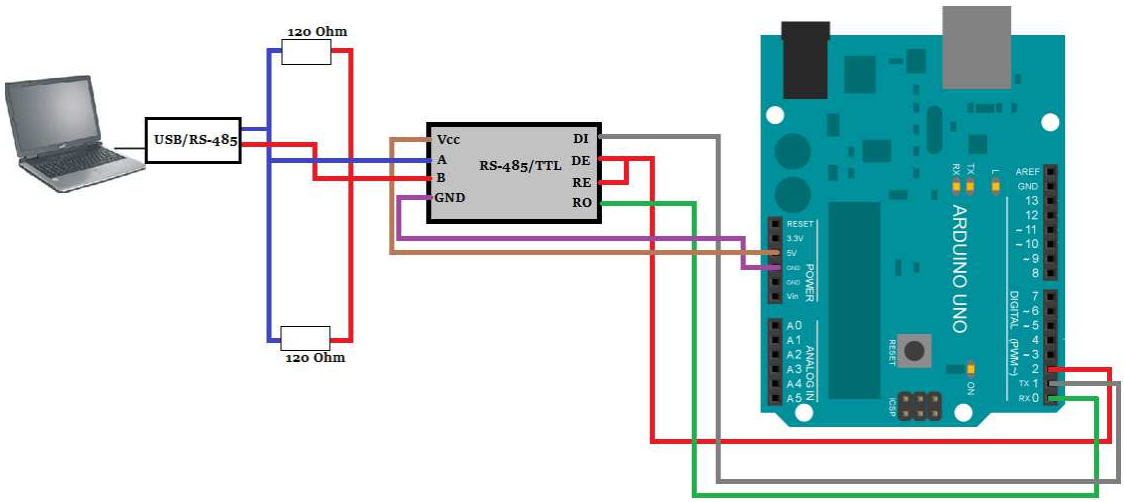
Na samom početku je potrebno pokrenuti serijsku komunikaciju naredbom *Serial.Begin*. Pri tome je ovoj funkciji, kao parametar potrebno proslijediti brzinu komunikacije (*BaudRate*). Postojanje komunikacije između Arduina i PC-a se ispituje naredbom *Serial.Available.* Za slanje podataka se mogu koristiti naredbe *Serial.Write* koja šalje binarnu vrijednost podatka, ili *Serial.PrintIn* kojom se mogu slati stringovi. Za prijem podataka koristi se funkcije *Serial.Read* koja vraća primljenu vrijednost. Jednostavan Arduino kôd, koji bi vršio prijem i slanje podataka je prikazan ispod.

```
int comdig=2;
String as;
void setup ()
{
 pinMode(digo,OUTPUT);
 pinMode(comdig,OUTPUT);
 digitalWrite(digo,LOW);
 digitalWrite(comdig,LOW);
 Serial.begin(9600);
 as="Hello world";
}

char Poredi(char* sign){
```

```
  while(Serial.available() > 0)
  {
    if(k < 9)
    {
      znak = Serial.read();
     Serial.flush();
      token[k] =znak;
      k++;
      token[k] = '\0';
    }
  }

  if(strcmp(token,sign)  == 0){
    for(int i=0;i<9;i++){
      token[i]=0;
    }
   k=0;
    return(0);

  }
  else{
    return(1);



  }
}
void loop()
{
  if(Poredi("AS")==0)
  {
    digitalWrite(comdig,HIGH);
    Serial.println(data);
    Serial.flush();
    digitalWrite(comdig,LOW);
  }
}
```

Kôd je omogućio da Arduino prima podatke, te u slučajnu odgovarajućeg pristiglog podatka (definisan stringom „AS“), Arduino je vršio slanje poruke. Pri tome je digitalnim pinom 2, definisano stanje Arduina. U slučaju da je pin postavljen na vrijednost logičke nule, Arduino je primao podatke. Postavljanjem vrijednosti na pina na nivo logičke jedinice Arduino je mogao slati podatke. Naredba *Serial.Flush* je omogućila da se osigura slanje podataka. Također je realizirana i funkcija *Poredi* koja je poredila pristiglu poruku sa odgovarajućim stringom.

Cilj je bio napraviti program u Arduinu, koji će omogućiti mjerenje dovedenog napona, te slanje izmjerenog napona PC-u, a istovremeno omogućiti prijem podataka sa PC-a. Pri tome je trebala biti omogućena serijska RS-485 komunikacija jednog PC-a, te 4 Arduino Uno razvojnog okruženja. Obzirom da se Arduino koristi TTL standard prvo se moralo izvršiti prilagođenje TTL standarda RS-485 standardu, a potom se moralo izvršiti prilagođenje RS-485 standarda USB-u, kako bi se omogućila komunikacija sa PC-om. U skladu sa tim data je shema spajanja na slici 3.6, a u nastavku će biti dat i kôd koji je omogućio zahtjeve na Arduinu.




**Slika 3.6.** Shema za RS-485 komunikaciju između Arduina i PC-a

Kôd kojim je Arduino[6] isprogramiran je prikazan ispod :

```
int anai=A0; // analogni ulaz
int digo=12; // digitalni izlaza;
```

[6] Serial_com\serialcom.ino

```
int in=0; // vrijednost ulaza
char token[10];
char znak=-1;
int ai;
String cc;
byte k=0;
int bs;
String data;
String as;
boolean x;  // tokeni slanja
void setup ()
{
 pinMode(digo,OUTPUT);
 digitalWrite(digo,LOW);// definisanje izlaza i stanja
 Serial.begin(9600);// Baudrate
 as="A";
 cc="-";
}
char Poredi(char* sign){

  while(Serial.available() > 0)
  {
    if(k < 9)
    {
      znak = Serial.read();
      token[k] =znak;
      k++;
      token[k] = '\0';
    }
  }
  if(strcmp(token,sign)  == 0){
    for(int i=0;i<9;i++){
      token[i]=0;
    }
   k=0;
    return(0);
  }
  else{
    return(1);
  }
}
void loop()
{
  ai=analogRead(anai);

  if(Poredi("AS")==0)
  {
    x=true;
  }
  if (Poredi("AZ")==0)
```

```
  {
    x=false;
  }
  if(Poredi("AN")==0)
  {
  digitalWrite(digo,HIGH);
}
 if(Poredi("AM")==0)
  {
  digitalWrite(digo,LOW);
}
  if(x==true)
  {
    String data=as+ai;
    while (data.length()!=5)
    {
      data=data+cc;
    }
    Serial.println(data);
  }
}
```

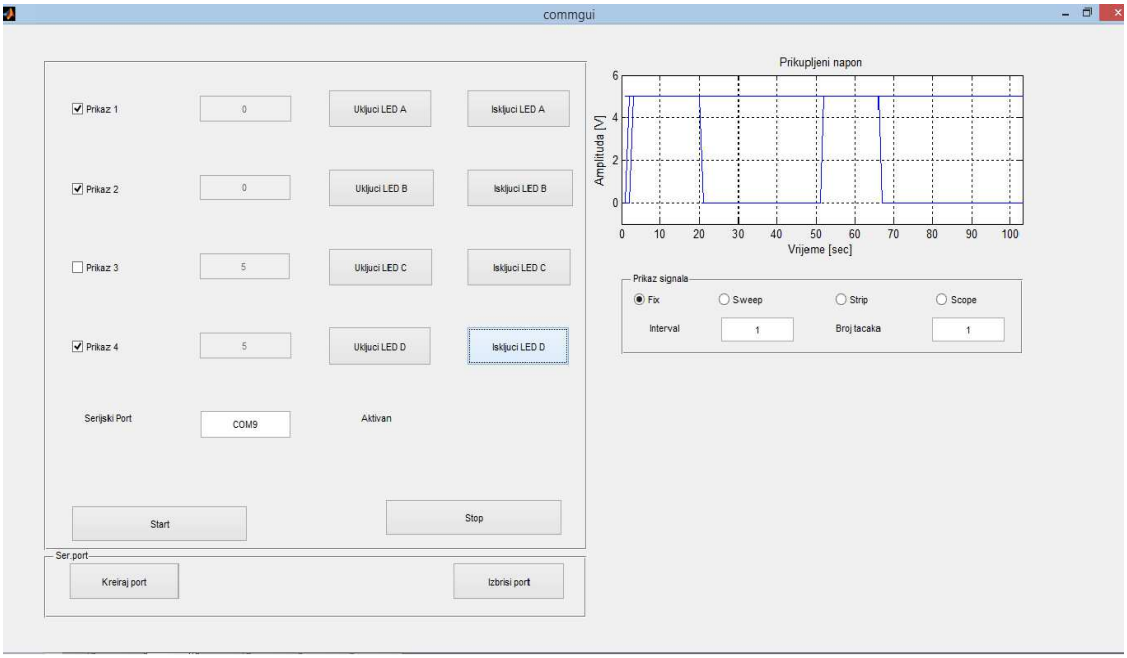
Kodom je omogućeno da se na Arduinu mjeri određeni napon, te se ta vrijednost proslijeđuje računaru, preko RS-485 komunikacije, dok preko računara se mogu zadavati određene naredbe Arduinu (u ovom slučaju uključivanje LED diode).

### 3.5. GUI za serijsku komunikaciju između PC-a i Arduino razvojnog okruženja

Na osnovu opisanih naredbi u MATLAB-u sa serijsku komunikaciju i Arduino koda napravljen je i GUI za serijsku komunikaciju. Cilj GUI-a jeste da omogući prijem podataka o prikupljenom napona sa Arduina, te date podatke prikaže na GUI-u, dok se istovremeno Arduino mogao kontrolisati slanjem odgovarajućih naredbi. Pri ovome se podrazumijevala komunikacija između PC-a i 4 Arduino Uno razvojna okruženja. Izgled GUI-a je na slici 3.7.

Korisnik je mogao da kreira odgovarajući serijski port, te da Arduinu odgovarajuće naredbe. Također, je mogao pokrenuti akviziciju putem jednog od Arduina, što je realizirano preko *timer*-a na način da se vršio prijem podataka od strane Arduina, te su se na osnovu ovih podataka dobijale vrijednosti napona. Vrijednosti su spremljene u

odgovarajuće varijable, a njihove vrijednosti su prikazane na GUI-u, uz mogućnost odabira načina prikaza.




**Slika 3.7.** GUI za serijsku komunikaciju između Arduina i MATLAB-a

Prednost ovog pristupa je prvenstveno cijena obzirom da je cijena četiri Arduina dosta manje od cijene jedne akvizicione kartice, koja je neophodna za GUI u prethodnom poglavlju. Generisanje signala je nešto složenije preko Arduina i podrazumijeva korištenje PWM-signala. Također, zadavanje parametara, poput amplitude, frekvencije, ili oblika generisanog signala, bi zahtijevalo dosta složeniji kôd u Arduinu. Pored toga Arduino može izmjeriti napon samo u opsegu [0,5] V, dok NI6024E može mjeriti napone u većem opsegu.

# 4. Obrada i prikupljanje slike u MATLAB-u

## 4.1. Uvod u procesiranje slike

Prije nego što bude napravljen osvrt na procesiranje slike, nakratko će biti govora i o samom procesu obrade slike. Proces obrade slike (eng. *Image Processing*- IP) predstavlja specijalan slučaj obrade signala, gdje ulaz u sistem predstavlja slika, koja može biti u formi videa, ili fotografije. Pri tome izlaz iz sistema može biti također slika, ili pak određeni podaci o samoj slici. Iako postoji i analogna obrada slike (eng. *Analog Image Processing*), ipak pod pojmom obrade slike se u većini slučajeva misli na digitalnu obradu slike (eng. *Digital Image Processing*- DIP). DIP podrazumijeva obradu digitalnih slika, odnosno obradu slika u okviru računara. Da bi se slike mogle obrađivati, prethodno moraju biti dovedene na ulaz računara i prevedene u digitalnu formu. Osnovna ideja prevođenja jeste predstavljanje slike, kao funkcije dvije varijable, u opštem obliku $f(x,y)$, gdje su $x$ i $y$, pozicije u ravni, odnosno pozicije određene tačke na slici, dok je $f$ predstavlja određenu amplitudu, kao naprimjer nivo sive boje na slici (eng. *intensity of gray level*). Pri tome parametri $x$, $y$ i $f$ imaju konačne diskretne vrijednosti. Može se primjetiti da se digitalna forma slike sastoji od konačnog broja tačaka, koje imaju određene vrijednosti. Mada postoje različiti nazivi za ove tačke, u praksi najčešće korišten termin je piksel (eng. *pixel*). Pri tome računar sliku prepoznaje kao dvodimenzionalni niz, odnosno u formi matrice. Ovo je naročito važno zbog primjene MATLAB-a. Ukoliko se želi prikazati slika u boji tada se najčešće prikazuje u formi trodimenzionalnog niza, a koristi se standardni RGB prikaz slike.

Proces obrade slike se može podijeliti u tri različite grupe (obrada slike, analiza slike i razumijevanje slike).

Obrada slike, gdje se je cilj od jedne dobiti drugu sliku obrađenu na osnovu određenih zahtjeva i primjenom određenih algoritama, npr. pretvaranje slike u crno-bijelu sliku.

Analiza slike, gdje je cilj na osnovu slike dobiti željene informacije, npr. analiza oblika koji se pojavljuju na slici.

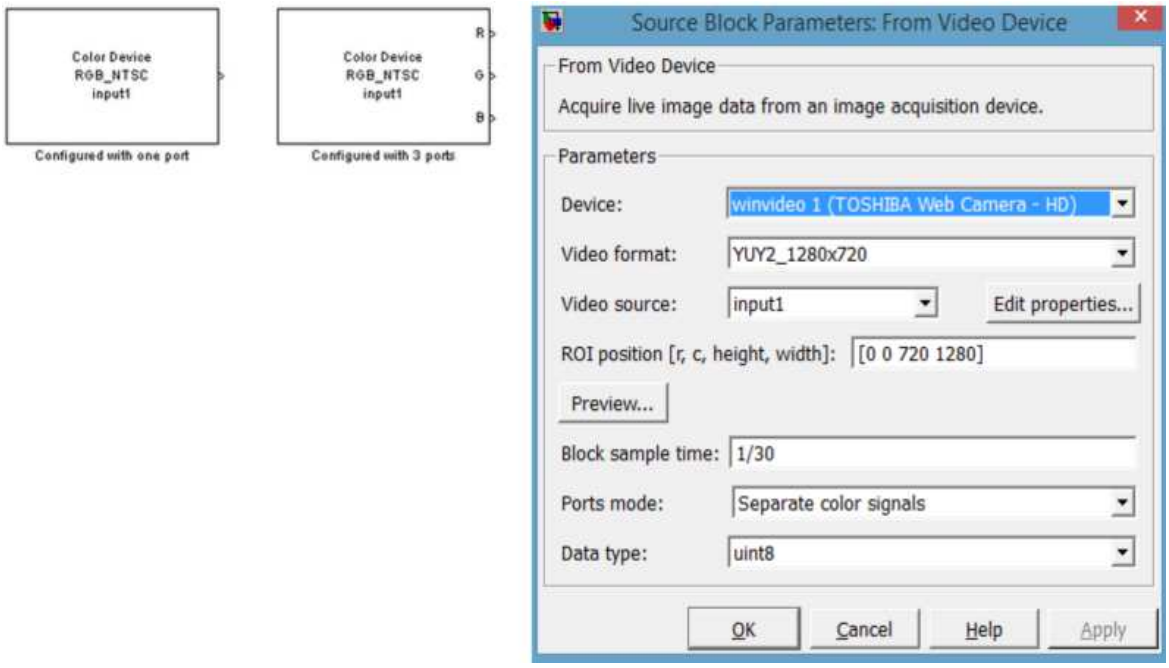
Razumijevanje slike, gdje se na osnovu slike donose određeni zaključci. Ova problematika je značajna kod kognitivne robotike.

Proces obrade slike se u većini slučajeva sastoji od niza etapa. U prvom koraku vrši se filtracija slike, kako bi se uklonile razne smetnje, zbog vanjskih uticaja, ili kvara na samoj kameri. Naredni korak je segmentacija slike. Segmentacija slike predstavlja podjelu slike na određene dijelove, pri čemu se svakom pikselu dodjeljuje dio kojem taj piksel pripada. Segmentacija ima naročit značaj kada je potrebno izvršiti prepoznavanje određenih objekata na slici. Nakon segmentacije vrši se analiza slike. Sada se dobijaju podaci, koji su od interesa, kao što su tekstura, boja, ili oblici na slici. Pri tome se vrši posebna analiza svakog segmenta. Nakon toga svakom segmentu se daje određeno značenje, odnosno svaki segment se klasificira na osnovu postojećih grupa segmenata. Na samom kraju vrši se kompresija i kodiranje slike. Pri tome sve veći značaj predstavlja i količina memorije potrebna za snimanje slike.

Primjene obrade slike su danas mnogobrojne, a ovdje će biti pomenute samo neke. Obrada slike koristi se u industriji, kada je potrebno prepoznati određene objekte, a u cilju poboljšanja i ubrzanja proizvodnje. Također, obrada slike se koristi za prepoznavanje kvarova, sve češću primjenu ima i u medicini (ultrazvuk, tomograf, X-zrake,itd.). Pored mnogobrojnih primjena obrada slike se koristi i u analizi struktura određenih površina.

## 4.2. Akvizicija slike u MATLAB-u

U okviru MATLAB-a moguće je jednostavno izvršiti akviziciju slike ili videa koristeći *Image Acquisition Toolbox*. Akvizicija se može vršiti koristeći širok opseg hardvera, od jednostavnih web-kamera, pa do sofisticiranijih industrijskih kamera [13]. Za prikaz slike će se, u nastavku, koristiti jednostavna web-kamera. Prednost ovog toolboxa jeste ta što se akvizicija slike može vršiti i preko SIMULINK-a, korištenjem bloka *From Video Device.*




**Slika 4.1.** Blok *From Video Device* i njegovi parametri

**Device**- omogućuje odabir hardvera, odnosno kamere koja se koristi za prikaz slike.

**Video Format**- omogućuje odabir formata u skladu sa odabranim hardverom.

**Video source**- omogućuje odabir kontrasta, osvjetljenja i ostalih parametara slike.

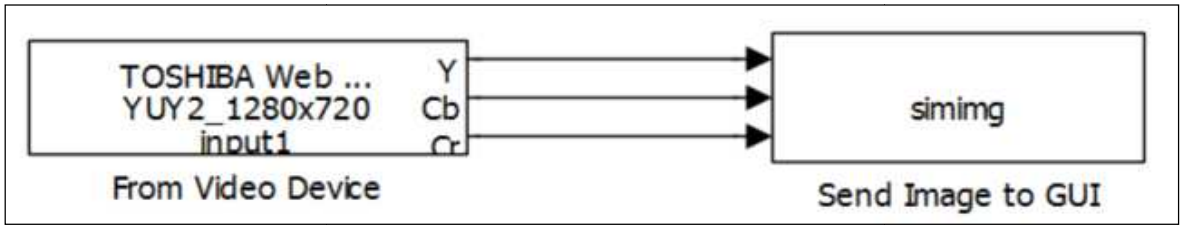
**ROI position**- određuje veličinu i poziciju slike.

**Sample Time**- predstavlja frekvenciju uzorkovanja.

**Ports mode**- određuje da li će blok imati jedan izlaz, ili tri. Ukoliko blok ima jedan izlaz, tada će izlazni signal biti trodimenzionalni niz, a u slučaju tri izlaza izlazi su matrice.

**Data Type**- predstavlja tip varijable izlaznog signala.

Model koji će vrštiti akviziciju i slike i istu prikazivati na odgovarajućem GUI je prikazan na slici 4.2.




**Slika 4.2.** Model za akviziciju slike

Model se sastoji od već navedenog bloka *From Video Device* te jednog bloka opisanog S-funkcijom[7] *simimg*. U nastavku je dat opis i prikaz ove funkcije:

```
function simimg(block)
   setup(block);
function setup(block)
   block.NumInputPorts  = 3;
   block.NumOutputPorts = 0;
   block.SetPreCompInpPortInfoToDynamic;
   block.InputPort(1).DatatypeID   = 3;
   block.InputPort(1).Complexity   = 'Real';
   block.InputPort(1).SamplingMode = 'Sample';
   block.InputPort(2).DatatypeID   = 3;
   block.InputPort(2).Complexity   = 'Real';
   block.InputPort(2).SamplingMode = 'Sample';
   block.InputPort(3).DatatypeID   = 3;
   block.InputPort(3).Complexity   = 'Real';
   block.InputPort(3).SamplingMode = 'Sample';
   block.NumDialogPrms = 0;
   block.SimStateCompliance = 'HasNoSimState';
   block.RegBlockMethod('SetInputPortDimensions',  @Dimen);
   block.RegBlockMethod('Outputs',              @Output);
function Dimen(block,~,di)
   block.InputPort(1).Dimensions = di;
   block.InputPort(2).Dimensions = di;
   block.InputPort(3).Dimensions = di;
function Output(block)
   global iaqhand;
   slika =cat(3,...
   block.InputPort(1).Data,...
   block.InputPort(2).Data, ...
   block.InputPort(3).Data);
   slika=ycbcr2rgb(slika);
   imshow(slika,'Parent',iaqhand.axes1);
   title(iaqhand.axes1,'Slika');
```

Može se primjetiti razlika između ranije korištenih s-funkcija i ove funkcije. Ova s-funkcija je za niži nivo klase od ranije navedenih, a razlike će biti pojašnjene u nastavku. Razlog je taj što s-funkcije ne mogu vršiti operacija sa trodimenzionalnim signalima, pa će se umjesto toga koristiti tri signala, koja će kasnije biti spojena u jedan.

```
function setup(block)
   block.NumInputPorts  = 3;
   block.NumOutputPorts = 0;
   block.SetPreCompInpPortInfoToDynamic;
   block.InputPort(1).DatatypeID   = 3;
   block.InputPort(1).Complexity   = 'Real';
   block.InputPort(1).SamplingMode = 'Sample';
   block.InputPort(2).DatatypeID   = 3;
```

[7] Image_proc\sf_iaq.m

```
    block.InputPort(2).Complexity   = 'Real';
    block.InputPort(2).SamplingMode = 'Sample';
    block.InputPort(3).DatatypeID   = 3;
    block.InputPort(3).Complexity   = 'Real';
    block.InputPort(3).SamplingMode = 'Sample';
    block.NumDialogPrms = 0;
    block.SimStateCompliance = 'HasNoSimState';
    block.RegBlockMethod('SetInputPortDimensions',  @Dimen);
    block.RegBlockMethod('Outputs',              @Output);
```

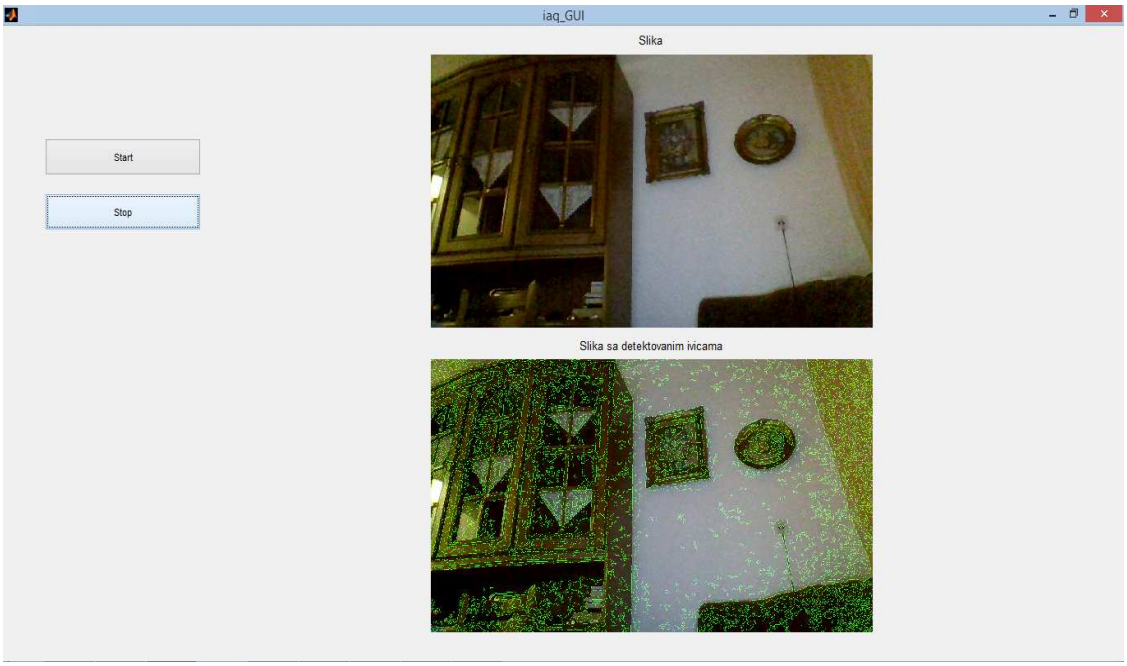
Na početku se definiše broj izlaza i broj ulaza u blok. Kako će se signal koji dovodi sa bloka *From Video Device* biti u formi zasebnih RGB signala, broj ulaza će biti uvijek postavljen na tri, dok sa izlazima iz sistema nema potrebe. Nakon toga je korištena naredba *SetPreCompInpPortInfoToDynamic*. Ova naredba omogućuje da se karakteristike ulaznog signala naslijede iz modela. Ove karakteristike su u nastavku funkcije definisane. Ovdje treba naglasiti da odabirom dimenzija, tipa signala ovakav pristup je manje općenit od ranije korištenog, jer omogućava korištenje samo jasno definisanih signala. Obzirom da je ovdje signal uvijek isti (slika sa kamere u boji, odnosno tri vektora) nema potrebe za korištenjem složenijeg pristupa, koji je ranije opisan.

Nakon toga se za svaki ulazni signal definiše tip (broj 3 konkretno označava korištenje tipa *uint8*), da li je signal realan ili imaginaran, a naredbom *SamplingMode* koristi se sa *SampleBased* signal.

Na samom kraju se definišu dvije metode: *Dimen* i *Output*. Metoda *Dimen* koristi se za dobijanje dimenzija ulaznog signala, dok metoda *Ouput* prikazuje sliku na GUI-u. Može se primjetiti da se slika u boji dobija sklapanjem tri ulazna signala u jednu trodimenzionalnu matricu naredbom *cat*, a potom se vrši konverzija formata slike (iz YCbCr u RGB), te prikazuje na GUI-u.

Mada je bi u ovom slučaju se mogao koristiti i pristup sa event_listenerima, zbog trodimenzionalnosti niza ne bi bilo moguće procesirati ove signale zajedno sa onim koji se dobijaju akvizicijom, odnosno generisanim, pa će se zbog toga za prikaz slike na GUI isključivo koristiti s-funkcije.

Za prikaz slike je kreiran jednostavni GUI sa naredbama za pokretanje i zaustavljanje SIMULINK modela, a na njemu su prikazani slika okruženja i slika okruženja sa detektovanim ivicama.




**Slika 4.3.** GUI za prikupljanje i obradu slike (detektovanje ivica na slici)

## 4.3. Obrada slike u MATLAB-u

Obrada slike u MATLAB-u se zasniva na *Image Processing Toolbox-u*. Ovdje će biti pomenute samo osnovne naredbe ovog alata, a koje su se koristile za razvoj GUI-a. Već je pomenuto da većina softvera, pa tako i MATLAB slike sprema kao 2D, ili 3D niz, u ovisnosti od toga da li je slika crno-bijela, ili je u boji. Učitavanje slike u MATLAB se vrši preko naredbe *imread()*. Ova naredba, kao rezultat vraća 2D, ili 3D niz, a kao parametar joj se prosljeđuje naziv slike. Za prikaz slike se mogu koristiti naredbe *imshow()* i *image()*. Razlika između ovih naredbi je što naredba *image* direktno vizualizira matricu, dok naredba *imshow* je nešto kompleksnija i prvo vrši konverziju matrice u sliku i tako je prikazuje. Pored ovoga naredba *image* prikazuje i x-osu i y-osu na slici. Također, naredbi *image* se mogu proslijediti samo matrice kao parametri, dok naredbi *imshow* se mogu proslijediti i nazivi slika.

MATLAB omogućava i pretvorbu slike u boji u crno-bijelu sliku, korištenjem naredbe *rgb2gray()*. Jedinstvena naredba koja crno-bijelu sliku pretvara u sliku u boji ne postoji, ali postoji niz načina da se crno-bijela slika prikaže kao slika u boji, prvenstveno koristeći *colormaps*. MATLAB ima definisan niz *colormaps*-a prikazanih na slici 4.4.

Prikaz proizvoljnog *colormapa* se može dobiti jednostavnim kodom.

```
cmp='jet'
v1=[0:0.1:1];
imshow(v1);colormap(cmp);
```

Za obradu slike su napravljenje dvije funkcije *imgfill* i *sqfill* koje će biti objašnjenje u nastavku. Funkcija *imgfill* će na slici prikazivati pravougaonike odgovarajuće veličine i boje, pri čemu će se slika ispod pravougaonika i dalje moći vidjeti. Sa druge strane naredba *sqfill* će odgovarajuće dijelove slike obojiti sa odgovarajućim bojama, odnosno ova funkcija će mijenjati same podatke o slici.


**Slika 4.4.** Definisani *colormaps* u okviru MATLAB-a

Izgled funkcije *imgfill*[8] je prikazan ispod :

```
function imgfill(s,dots,x,hand,lx,sd,tp)
  if nargin<7
    tp=0.333;
```

[8] Image_proc\imgfill.m

```
    sd=50;
  elseif nargin<6
    tp=0.333;
  end
    m=dots+sd;
    [cmp,cx]=cmaps(x);
    imshow(s,'Parent',hand);
for i=1:size(dots,1)
  if ismember(i,lx)
    p=patch([dots(i,1) m(i,1) m(i,1) dots(i,1)],[dots(i,2) dots(i,2) m(i,2)
m(i,2)],[cmp(dots(i,3),1) cmp(dots(i,3),2)
cmp(dots(i,3),3)],'Parent',hand);
    set(p,'FaceAlpha',tp);
  end
end
end
```

Funkcija može primiti 7 parametara, pri čemu zadnja dva nisu obavezna. Prvi parametar predstavlja sliku koja se obrađuje, drugi parametar predstavlja vektor sa koordinatama tačaka i amplitudom, dok treći parametar predstavlja *colormap*. Ostali parametri predstavljaju grafik na kojem se slika prikazuje, tačke koje se žele prikazati, veličinu pravougaonika i transparentnost obojenog pravougaonika. Izostavljanjem zadnja dva parametra se postavljaju na podrazumijevane vrijednosti. Nakon toga se odabire *colormap* preko definisane funkcije *cmaps*, te se nakon toga prikazuje slika, te se na osnovu odabranih tačaka prikazuju pravougaonici na definisanim tačkama, odgovarajuće veličine, te obojeni bojom u ovisnosti od odabranog *colormapa* i amplitude. Pravougaonici se prikazuju naredbom *patch*, koja kao parametar prima transparentnost boje kojom se pravougaonik boji.

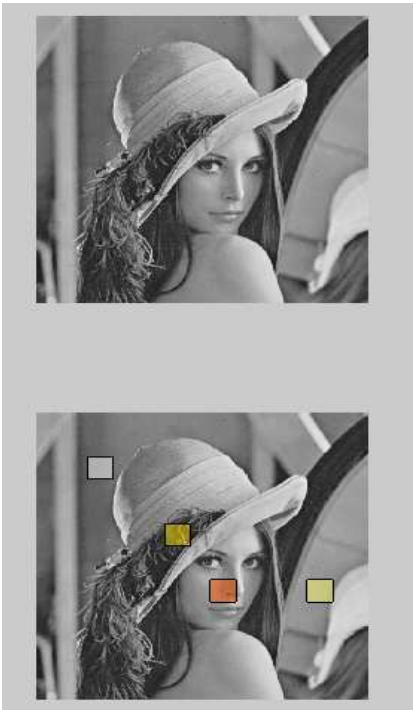
Funkcija *sqfill*[9] je prikazana ispod :

```
function sqfill(s,dots,x,hand,lx,sd)
  if nargin<6
    sd=50;
  end
    [cmp,cx]=cmaps(x);
    disp(size(cmp))
    subplot(2,1,1)
    plot([],[]);
    imshow(s);
    c=s;
for i=1:size(dots,1)
  if ismember(i,lx)
```

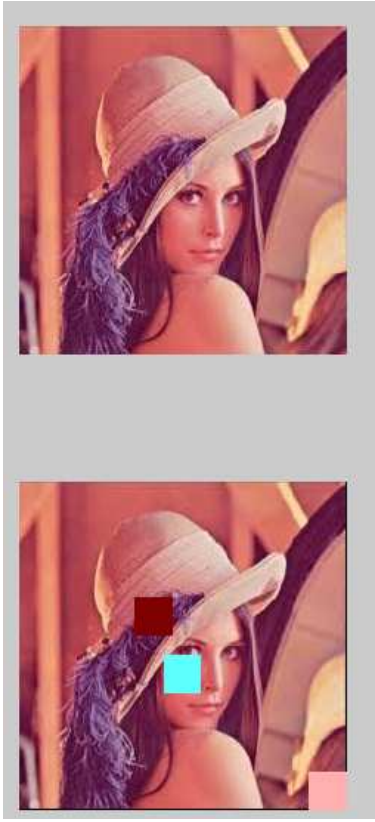
[9] Image_proc\sqfill.m

```
s(dots(i,1):dots(i,1)+sd,dots(i,2):dots(i,2)+sd,1)=cmp(dots(i,3),1)*255;

s(dots(i,1):dots(i,1)+sd,dots(i,2):dots(i,2)+sd,2)=cmp(dots(i,3),2)*255;

s(dots(i,1):dots(i,1)+sd,dots(i,2):dots(i,2)+sd,3)=cmp(dots(i,3),2)*255;
  end
end
    subplot(2,1,2)
    plot([],[]);
    imshow(s);
end
```

Funkcija je dosta slična prethodnoj, a razlika je u tome što se ne prikazaju pravougaonici, već se dijelovima slike se dodijeljuju vrijednosti odgovarajuće boje, odnosno dio slike se oboji odgovarajućom bojom.



**Slika 4.5.** Prikaz slike i obrađene slike uz korištenje funkcije *imgfill*



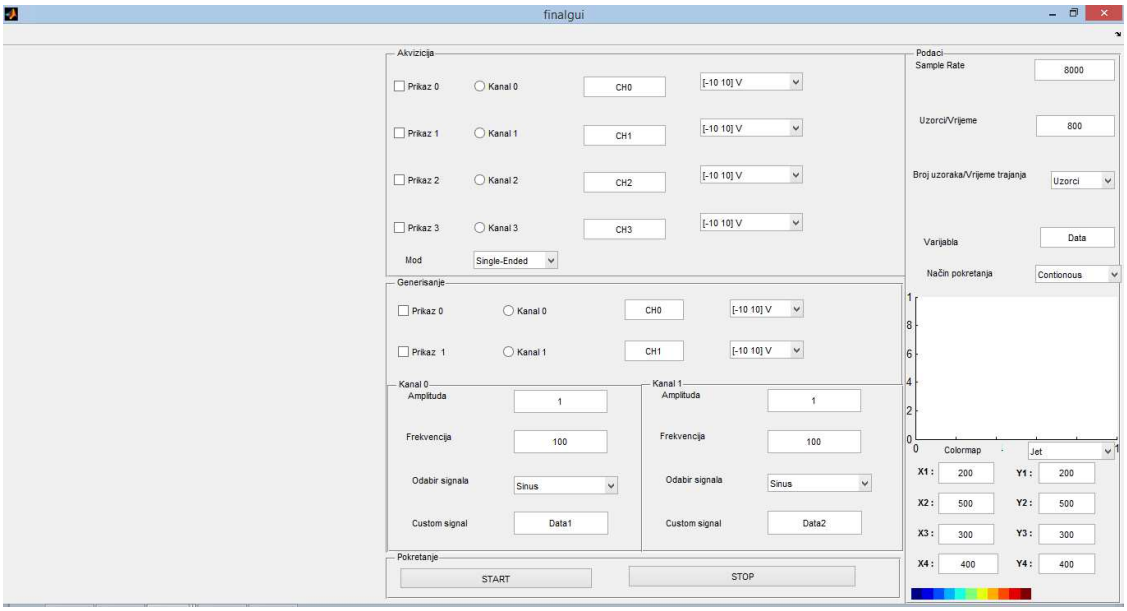
**Slika 4.6.** Prikaz slike i obrađene slike uz korištenje funkcije *sqfill*

# 5. GUI za mjerenje, nadzor i upravljanje ultrazvučnom kamerom

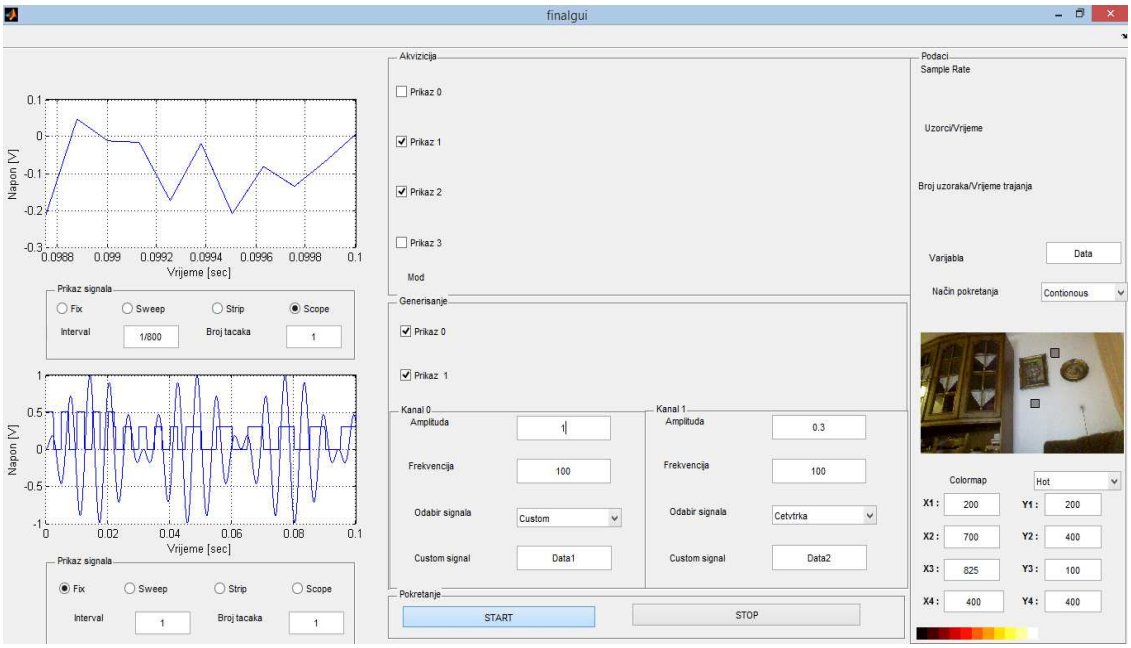
Razvoj softverskih modula za mjerenje, nadzor i upravljanje ultrazvučnom kamerom se može podijeliti u dva različita GUI-a. Prvi GUI se zasnivao na akviziciji i generisanju podataka putem akvizicione kartice, dok se drugi GUI zasnivao na akviziciji podataka preko Arduina i prenosu podataka putem RS-485 komunikacije.

## 5.1. GUI za mjerenje, nadzor i upravljanje ultrazvučnom kamerom koristeći akvizicionu karticu

Preko akvizicione kartice vršilo se mjerenje i generisanje signala, a ovi signali su se prikazivali na GUI-u. Izgled GUI-a, prije pokretanja je na slici 5.1., dok je izgled GUI-a nakon pokretanja na slici 5.2.

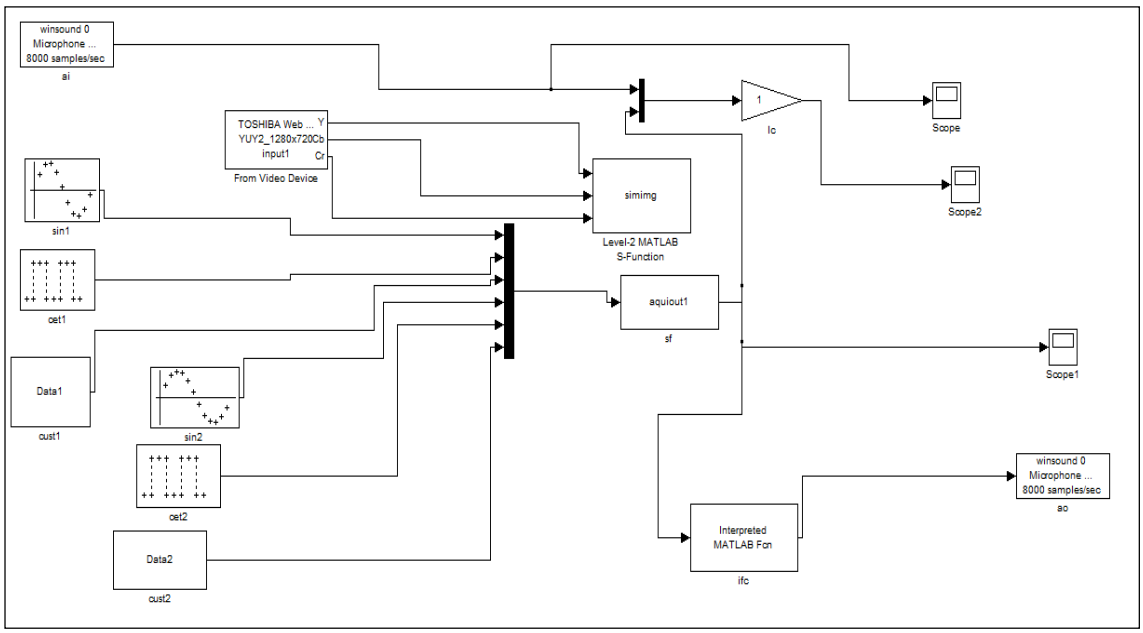



**Slika 5.1.** GUI za mjerenje, nadzor i upravljanje koristeći akvizicionu karticu, prije pokretanja




**Slika 5.2.** GUI za mjerenje, nadzor i upravljanje koristeći akvizicionu karticu, nakon pokretanja

Osnov za GUI je bio SIMULINK model na slici 5.3.

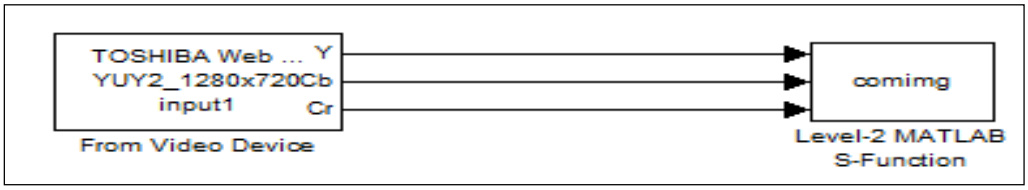



**Slika 5.3.** SIMULINK model za akviziciju, generisanje podataka i prikaz okruženja

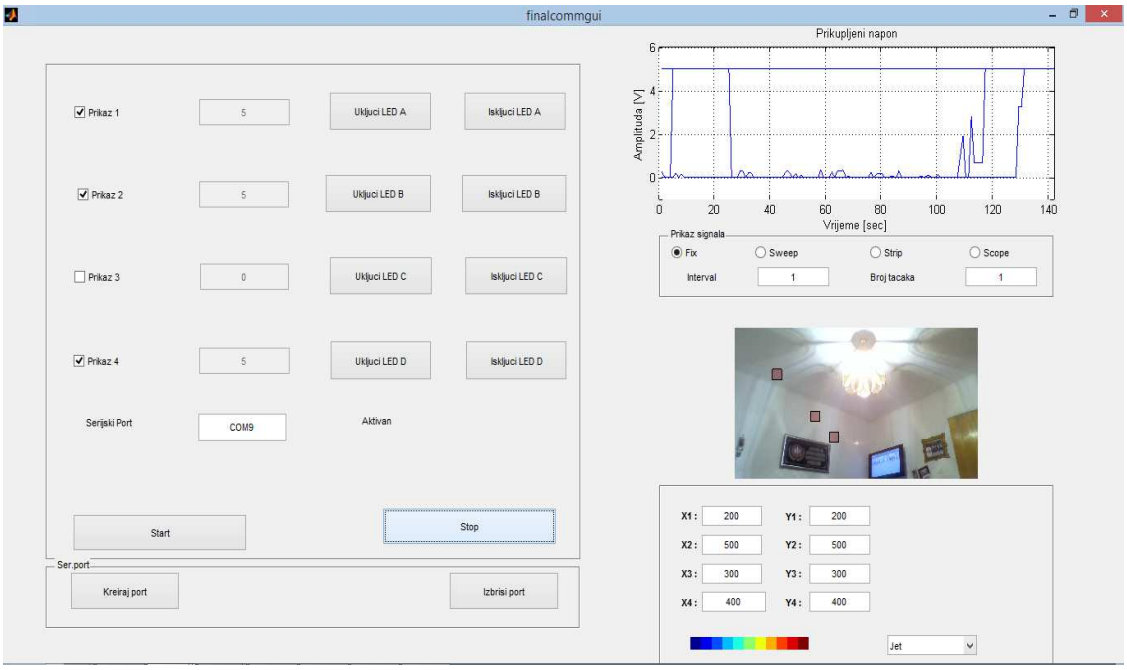
Mogućnosti GUI-a su ranije navedene, a ovdje će se dodati da se za analizu slike okruženja koristila funkcija *imgfill*, a da su boje prikazanih pravougaonika odgovarale amplitudama prikupljenih signala. Također, korisnik je, preko GUI-a, mogao birati pozicije pravougaonika. Za upravljanje GUI-em su korištene metode *event_listener*-a, i s-funkcija. Preko *event_listener*-a je realizirana akvizicija i slanje podataka, dok je preko s-funkcije realiziran prikaz i analiza slike na GUI-u, te prikaz odabranog *colormapa.*

## 5.2. GUI za mjerenje, nadzor i upravljanje ultrazvučnom kamerom koristeći serijsku komunikacij sa Arduino okruženjem

Za ovaj GUI korišten je već ranije navedeni GUI za komunikaciju PC-a sa Arduinom, pri čemu je GUI proširen na način da se sada vršio prikaz i analize slike okruženja. Kao i u prethodno navedenom GUI-u korisnik je mogao birati odgovarajući *colormap*, te poziciju pravougaonika, pri čemu je boja pravougaonika ovisila od amplitude napona koji se dovodio na ulaz Arduina, a primljen je od strane PC-a. Za prikaz slike na GUI-u korišten je model na slici 5.3., te odgovarajuća s-funkcija. Izgled GUI-a je prikazan na slici 5.4.




**Slika 5.4.** SIMULINK model za prikaz slike okruženja na GUI-u




**Slika 5.5.** GUI za mjerenje, nadzor i upravljanje preko serijske komunikacije

# 6. Real-time simulacija, koristeći xPC Target mod

Testiranje aplikacije, koja je koristila SIMULINK model se jednostavno moglo izvršiti na istom PC-u na kojem se nalazi aplikacija. Međutim, MATLAB omogućava testiranje iste aplikacije na istom hardveru (NI6024E), ali preko drugog PC-a, koristeći xPC target. Ovo omogućava bolje performanse aplikacije, jer sada aplikacija može koristiti sve računarske resurse, što svakako poboljšava njen rad.

## 6.1. Uvod u xPC target mod

xPC Target mod predstavlja softver kojim se vrši real-time simulacija i testiranje SIMULINK modela i Stateflow dijagrama na stvarnom sistemu. Cilj xPC Targeta jeste da simulaciju u okviru SIMULINK-a što više pribiliži realnoj simulaciji. Za xPC Target potrebni su *host PC*, *target PC*, te hardver koji se testira. Na *host PC*-u se nalaze MATLAB, SIMULINK, te *Real-Time* i *xPC Target Toolbox*. *Target PC* pokreće SIMULINK model, a pri tome se koriste njegovi računarski resursi (RAM, CPU, itd.). *Target PC* može biti obični PC, ali se prethodno mora pokrenuti preko odgovarajućeg *boot* diska. Nakon završene simulacije, računar može nastaviti sa normalnim radom. Pored običnih PC-a za *target PC* se mogu koristiti i specijalni računari, preko *xPC Embeded* opcije, a pokreću se preko specijalnih *boot* diskova. Pri tome *host* i *target PC* moraju biti povezani preko serijske (RS-232, RS-485), ili mrežne komunikacije (LAN, Internet, ili Ethernet kabal). Za hardver se može koristiti širok opseg akvizicionih kartica, a bitno je istaći da ovaj toolbox omogućuje korištenje i NI6024E kartice.

Već je pomenuto da je za xPC Target potreban odgovarajući *boot* disk. MATLAB omogućuje automatsko generisanje ovog diska. Za uspješan rad xPC targeta potrebno je prvo kreirati objekati tipa *xpc*. Pri tome se kao parametar proslijeđuje adresa *target PC*-a. Primjer ove naredbe je prikazan ispod:

```
adr= 144.212.206.251;port=2222;

setxpcenv('TcpIpTargetAddress',address,'TcpIpTargetPort',port);
tg=xpc('TcpIP',adr,port);
```

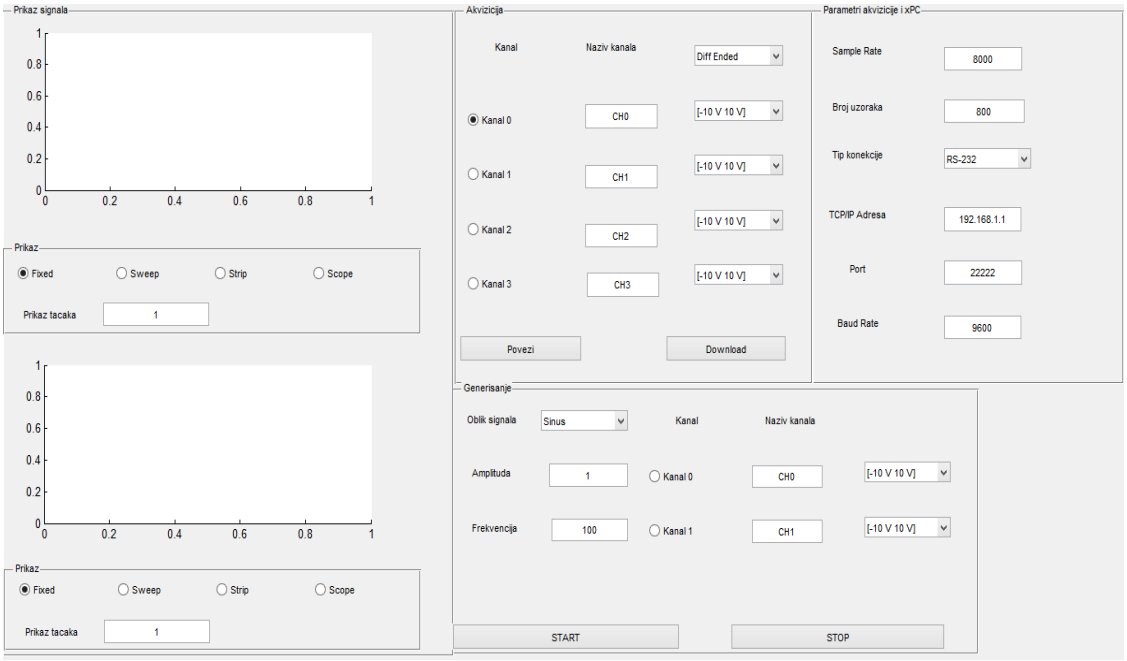
Ova naredba će povezati *host PC* sa odgovarajućim *target PC*-om, preko TCP/IP veze. Pored ovog načina povezivanja moguće je koristiti i druge tipove komunikacije, poput RS-232, Ethernet, a povezivanje je moguće i preko web-pretraživača. Ukoliko se uspješno povežu dva PC-a, nakon toga se omogućuje povezivanje i prenos podataka na *host PC*. Naredba koja ovo omogućava je *load*. Ova naredba prenosi odgovarajući SIMULINK model na *target* PC, a kao parametre prima objekat tipa *xpc*, te naziv SIMULINK modela. Testiranje ispravnosti povezanosti dva PC-a se izvršava preko naredbe *targetping*. Za kompajliranje SIMULINK modela u odgovarajući kôd koristi se naredba *rtwbuild*, koja kao parametar prima naziv SIMULINK modela. Pri tome je potrebno imati odgovarajući kompajler, poput *Visual Studia.*

Za prikaz signala moguće je definisati dva *Scope*-a, *host Scope* i *target Scope*. *Host Scope* služi za prikaz simuliranih podataka na *host PC*-u, dok *target Scope* omogućava prikaz podataka na *target PC*-u. Za *targetScope* postoji odgovarajući blok u SIMULINK-u, preko kojeg se vrši prikaz signala na *target PC*-u. Definisanje odgovarajućeg *Scope*-a, vrši se preko naredbe *addscope*, koja kao parametre prima *xpc* objekat, vrstu *Scope*-a, te broj *Scope*-a koji se želi definisati. Pored ove naredbe postoji i naredba *remscope*, koja briše dati *Scope*.

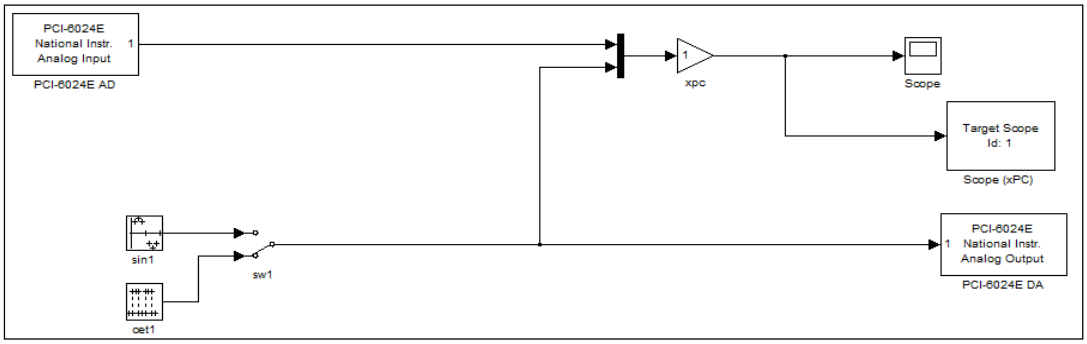
## 6.2. Akvizicija i generisanje signala putem xPC Target-a

Na osnovu osnovnih naredbi *xPC Target Toolbox*-a, se može kreirati GUI preko kojeg bi se vršila akvizicija i generisanje signala. Korisnik će kao i ranije imati mogućnost kanala za akviziciju, odnosno generisanje signala, opseg i mod kanala, frekvenciju uzorkovanja, broj prikupljenih uzoraka, te odabir načina prikaza signala i broj tačaka koji se prikazuje. Pored ovih mogućnosti korisnik može birati i način povezivanja dva PC-a, pri čemu ima mogućnost izbora između RS-232, ili Ethernet-a. Izgled GUI-a je prikazan na slici 6.1. Za ispravan rad GUI bilo je potrebno kreirati i odgovarajući SIMULINK model prikazan na slici 6.2. Ovdje je bitno istaći da će za promjenu parametara koristiti drugačiji pristup od ranije navedenih. Definisat će se odgovarajući *timer*, te u okviru *TimerFcn* vršit će se prikaz parametara, te promjena parametara simulacije. Pristup sa s-funkcijama se neće koristiti iz razloga što kompajliranje ovih funkcija zahtijeva odgovarajuće *dll* fajlove, koji

se ne moraju nalaziti na *host PC*-u, a pristup koristeći *event_listener*-e zahtijeva definisanje funkcije za SIMULINK model koja se ne nalazi na *host PC*-u, već na *target* PC-u. Ovdje treba istaći da se promjena parametara simulacija može izvršiti na nekoliko različitih načina, kao što su korištenje *xPC Target Explorer*-a, međutim neki od ovih načina neće promijeniti parametre koji se šalju na *target PC*-a, sve dok traje simulacija. Korištenje već navedene funkcije *set_param* omogućava da se promijenjeni parametri šalju na *target PC* u toku simulacije, ukoliko se kao tip simulacija postavi *External*.



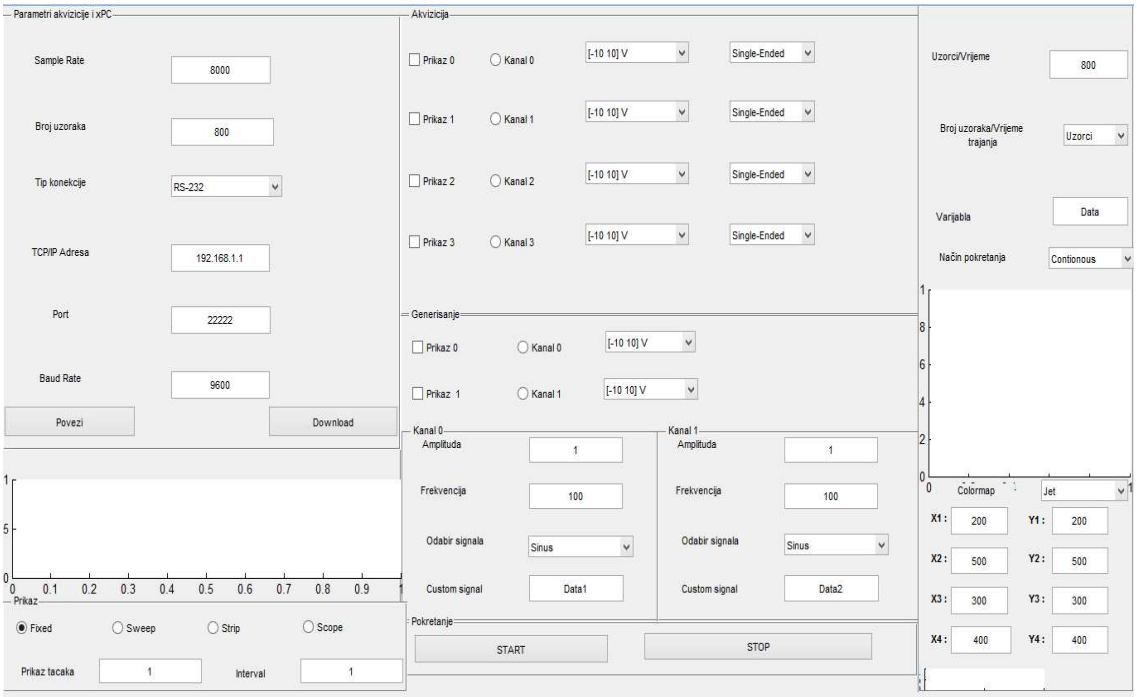

**Slika 6.1.** GUI za akviziciju i generisanje podataka koristeći xPC Target




**Slika 6.2.** SIMULINK model za xPC Target mod koristeći NI6024E karticu

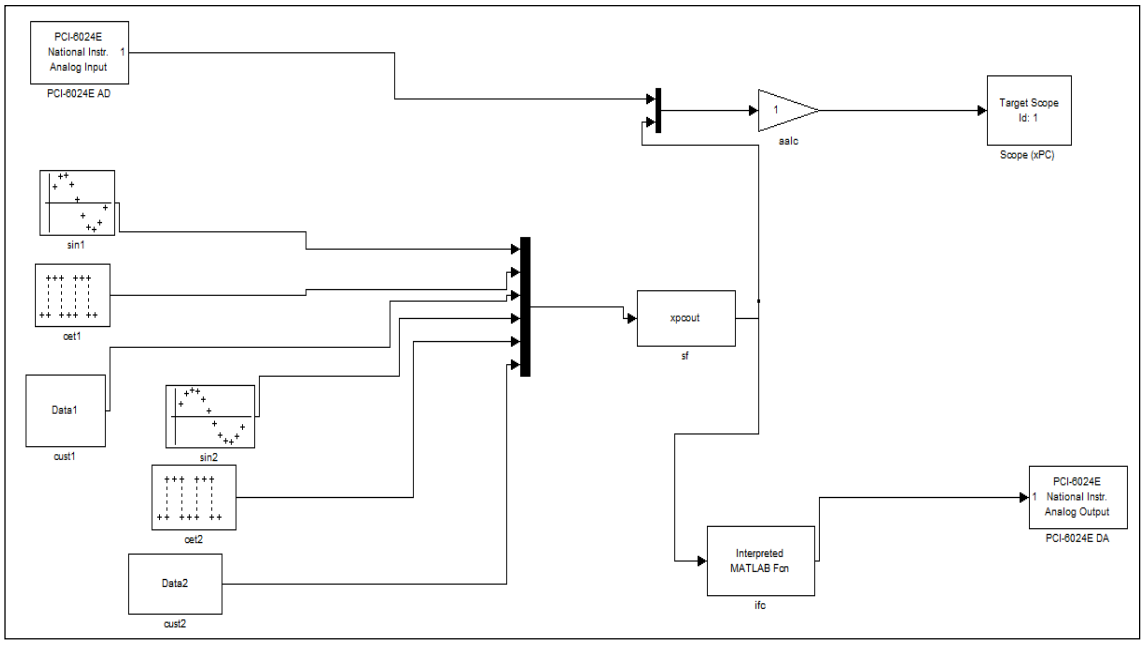
## 6.3. GUI za mjerenje, nadzor i upravljanje ultrazvučnom karticom u xPC Target modu

Raniji GUI za mjerenje, nadzor i upravljanje ultrazvučnom kamerom putem akvizicione kartice se može modifikovati, tako da radi u xPC Target modu. Pri tome je potrebno modifikovati SIMULINK modele, obzirom da se u ovom modu mogu samo koristiti određene vrste kamere. Zbog ovoga vršit će se simulacija dva SIMULINK modela paralelno. U okviru prvog SIMULINK modela vršit će se akvizicija slike, a u okviru drugog vršit će se akvizicija signala u xPC Target modu. Na GUI-u bi se vršio samo prikaz analizirane slike okruženja, uz prikaz prikupljenih signala, dok bi se pokrenuo i *target Scope* kako bi se jasno mogli vidjeti svi prikupljeni i generisani signali. Izgled GUI-a je na slici 6.3.

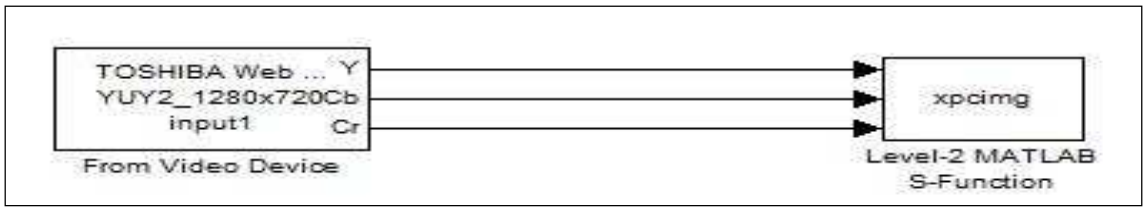



**Slika 6.3.** GUI za nadzor, upravljanje i mjerenje ultrazvučnom kameromu xPC Target modu

Za funkcionalan rad GUI-a potrebno je odabrati koristiti dva SIMULINK modela, koji su prikazani na slikama 6.4. i 6.5.




**Slika 6.4.** SIMULINK model za akviziciju i generisanje podataka u xPC Target modu




**Slika 6.5.** SIMULINK model za prikaz slike na GUI-u

# 7. Zaključak

U okviru rada su obrađeni razni aspekti akvizicije i generisanja signala u MATLAB-u, koji uključuje klase za akviziciju i generisanje signala, modele u SIMULINK-u za akviziciju i generisanje signala, te interfejse koji su omogućili akviziciju i generisanje ultrazvučnih signala. Pored akvizicije podataka u MATLAB-u, obrađena je i akvizicija u okviru Arduino razvojnog okruženja, zajedno sa modulima za serijsku komunikaciju za Arudino i MATLAB. Prikupljeni podaci su iskorišteni kako bi se vršila analiza slike okruženja korištenjem obične kamere, za što bilo potrebno napraviti odgovarajuće softverske module u okviru MATLAB-a. Za poboljšanje akvizicije i simulacije napravljen je i xPC Target mod ovakvog GUI-a. Ovi moduli su uspjeli omogućiti upravljanje, nadzor i mjerenje ultrazvučnom kamerom.

Pored ovoga softverski moduli se mogu koristiti i za druge svrhe. Implementiranjem regulatora u okviru MATLAB koda, ili dodavanjem odgovarajućeg bloka u SIMULINK, postojeći interfejsi bi omogućili regulaciju tehnoloških procesa korištenjem akvizicione kartice. Također, implementacijom regulatora u okviru Arduina, ili u okviru MATLAB-a mogli bi se iskoristiti za regulisanje i nadzor procesa preko serijske komunikacije korištenjem Arduina, ili nekog drugog razvojnog okruženja.

Osim toga, interfejsi se mogu koristiti i za vizuelno i audio-snimanje korisnika, pri čemu se postojeća analiza slika mogla izmjeniti, pa bi korisnik imao mogućnost snimanja različitih videa, kao što je snimanje u crno-bijeloj tehnici.

U radu je obrađeno i nekoliko *toolbox*-a u MATLAB-u, pa se rad može koristiti i za dobijanje osnovnih informacije za rad sa ovim *toolbox*-ima.

Na samom kraju treba istaći da pored osnovne namjene softverskih modula koje su uključivale mjerenje, nadzor i upravljanje ultrazvučnom kamerom, oni se mogu koristiti u raznim oblastima, koje uključuju automatske procese, obradu slika, multimedijalne softvere, itd.

# 8. Lista skraćenica

A/D konverzija- analogno/digitalna konverzija

bps- eng. *bit per second*, bit po sekundi

COM- eng. *Communication Port*, port za serijsku komunikaciju na PC-u

D/A konverzija- digitalno/analogna konverzija

DAQ- eng. *Data Aquisition Hardware*, akviziciona kartica

DC- eng. *Direct Current*, istosmjerna struja

DCE- eng. *Data Circuit Terminating Equipment*, host uređaj za komunikaciju

DIP- eng. *Digital Image Processing*, digitalna obrada slike

DTE- eng. *Data Terminal Equipment*, periferni uređaj za komunikaciju

DSP- eng. *Digital Signal Processor,* digitalni procesor za obradu signala

EEPROM- eng. *Electrically Erasable Programmable Read-Only Memory*, električki izbrisive programabilne memorije

GUI- eng. *Graphic User Interface*, korisnički interfejs

IP- eng. *Image Processing*, obrada slike

PC- eng. *Personal Computer*, osobni kompjuter

PWM- eng. *Pulse Width Modulation*, širinsko-impulsna modulacija

RX- pin za primanje podataka

SRAM- eng. *Static random-access memory,* memorija nasumičnog pristupa

TX- pin za slanje podataka

USB - eng. *Universal Serial Bus*

# 9. Literatura


[1] Melita Ahić-Đokić, “Signali i sistemi”, Elektrotehnički fakultet Sarajevo, 2010.

[2] Alan V. Oppenheim, Alan S. Willsky: “Signals and Systems”, Prentice-Hall, 1997

[3] Robert J. Schilling, Sandra L. Harris, “Fundamentals of Digital Signal Processing using MATLAB 2nd Edition”, Clarkson University Potsdam NY, 2012.

[4] MathWorks, “Data Aquisition Toolbox, User’s Guide”,The MathWorks, Inc., 2014.

[5] National Instruments,”6023E/6024E/602E UserManual”, National Instruments, 2000.

[6] Slika preuzeta iz [*www.sine.com*], 31.03.2014.

[7] MathWorks, ”Object-Oriented Programming”, The MathWorks, Inc., 2014.

[8] [*http://blogs.mathworks.com/pick/2012/06/01/use-matlab-guis-with-simulink-models/*]

[9] MathWorks, “SIMULINK Developing S-Functions”, The MathWorks, Inc., 2014.

[10] Electronic Industries Association, “Interface between data terminal equipment and data communication equipment employing serial binary data interchange”, Electronic Industries Association, Engineering Department, 1969.

[11] Samim Konjicija, Predavanja iz predmeta Praktikum Automatike, 2014. [*http://c2.etf.unsa.ba/course/view.php?id=11*]

[12] Jan Axelson,“ USB Complete Develepor's Guide Fourth Edition“, Lakeview Research LLC, 2009.

[13] [*http://arduino.cc/en/Main/arduinoBoardUno*]

[14] Mike McRoberts, “Arduino starter kit manual”, London: Earthshine Electronics, 2009.

[15] MathWorks, “Instrument Control Toolobox“, The MathWorks, Inc., 2014.

[16] Mark S. Manalo, Ashkan Ashrafi, “ USB Interfacing and Real Time Data Plotting with MATLAB“, San Diego University Department of Electrical and Computer Engineering Real-Time DSP and FPGA Development Lab, 2012.

[17] Rafael C. Gonzalez, Richard E.Woods, “Digital Image Processing”, Prentice-Hall, 2001.

[18] Tinku Acharya i Ajoy K. Ray ,“Image Processing Principles and Applications“, John Wiley and sons, 2005.

[19] MathWorks, “Image Aquisiton Toolbox User Guide“, TheMathWorks, Inc., 2014.

[20] MathWorks, “Image Proccessing Toolbox User Guide“, The MathWorks, Inc., 2014

[21] MathWorks, “xPC Target Getting Started Guide“, The MathWorks, Inc., 2010

[22] MathWorks, „xPC Target for use with Real-Time Workshop I/O Reference Guide“, The MathWorks, Inc., 2000

[23] Melita Ahić-Đokić, “Logički dizajn”, Elektrotehnički fakultet Sarajevo, 2006.

# 10. Prilog

U nastavku će biti priloženi kodovi za korištene funkcije, a koje nisu pojašnjenje u okviru rada.

S-funkcija[10] za slanje podataka u SIMULINK-u *aquiout*:

```
function [sys,x0,str,ts] = aquiout(t,x,u,flag,Ts)
switch flag,
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(Ts);
  case 2,
    sys=mdlUpdate(t,x,u,Ts);
  case 3,
    sys = mdlOutputs(t,x,u);
  case { 1, 4, 9 },
    sys = [0 0 0 0];
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes(Ts)
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 1;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 6;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [0];
str = [];
ts  = [Ts 0];
function [sys]=mdlUpdate(t,x,u,Ts)
sys=[];
function sys = mdlOutputs(t,x,u)
 global hand;
sys=[];
if get(hand.chg0,'Value')
    if(get(hand.sig1,'Value')==1)
        sys=[sys;u(1)];
    elseif (get(hand.sig1,'Value')==2)
        sys=[sys;u(2)];
    elseif(get(hand.sig1,'Value')==3)
       sys=[sys;u(3)];
   end
end
```

[10] GUI za ultrazvucnu kamer AQ_kartica\aquiout.m

```
if get(hand.chg1,'Value')
    if(get(hand.sig2,'Value')==1)
        sys=[sys;u(4)];
    elseif (get(hand.sig2,'Value')==2)
       sys=[sys;u(5)];
    elseif(get(hand.sig2,'Value')==3)
        sys=[sys;u(6)];
    end
end
```

Funkcija[11] za inicijalizaciju SIMULINK modela *init*:

```
function init(name,x,sr,xd,dx)
load_system(strcat(name,'.mdl'));
set_param(strcat(name,'/ai'),'ActualRate',num2str(sr));
set_param(strcat(name,'/ao'),'ActualRate',num2str(sr));
set_param(strcat(name,'/ai'),'SampleRate',num2str(sr));
set_param(strcat(name,'/ao'),'OutputRate',num2str(sr));
set_param(name,'SolverType','Fixed-Step');
set_param(name,'SolverName','FixedStepDiscrete');
set_param(name,'FixedStep',num2str(1/sr));
set_param(name,'StopTime',strcat(num2str(x)));
set_param(strcat(name,'/sin1'),'SineType','Sample-Based');
set_param(strcat(name,'/sin2'),'SineType','Sample-Based');
set_param(strcat(name,'/ai'),'AcqMode','1');
set_param(strcat(name,'/ai'),'NChannelsSelected',num2str(dx));
set_param(strcat(name,'/ai'),'NPorts','1 for all hardware channels');
set_param(strcat(name,'/ao'),'OutMode','1');
set_param(strcat(name,'/ao'),'NChannelsSelected',num2str(xd));
set_param(strcat(name,'/ao'),'NPorts','1 for all hardware channels');
set_param(strcat(name,'/cet1'),'PulseType','Sample Based');
set_param(strcat(name,'/cet2'),'PulseType','Sample Based');
set_param(strcat(name,'/sf'),'Parameters',num2str(1/sr));
set_param(strcat(name,'/sin1'),'SampleTime',num2str(1/sr));
set_param(strcat(name,'/cet1'),'SampleTime',num2str(1/sr));
set_param(strcat(name,'/sin1'),'Samples',num2str(sr/1000));
set_param(strcat(name,'/cet1'),'PulseWidth',num2str(1));
set_param(strcat(name,'/cet1'),'Period',num2str(sr/1000));
set_param(strcat(name,'/cet1'),'PulseWidth',num2str(sr/2000));
set_param(strcat(name,'/sin2'),'SampleTime',num2str(1/sr));
set_param(strcat(name,'/cet2'),'SampleTime',num2str(1/sr));
set_param(strcat(name,'/sin2'),'Samples',num2str(sr/1000));
set_param(strcat(name,'/cet2'),'PulseWidth',num2str(1));
set_param(strcat(name,'/cet2'),'Period',num2str(sr/1000));
set_param(strcat(name,'/cet2'),'PulseWidth',num2str(sr/2000));
set_param(strcat(name,'/cust1'),'SampleTime',num2str(1/sr));
set_param(strcat(name,'/cust2'),'SampleTime',num2str(1/sr));
end
```

[11] GUI za ultrazvucnu kameru AQ_kartica\init.m

Funkcija[12] za pretvaranje proizvoljnog signala u strukturu *strukt*:

```
function  siminx=strukt(x)
    simin.time=[];
    str.values(1,:,:)=x;
    str.dimensions=[1 1];
    simin.signals=str;
    siminx=simin;
end
```

Funkcija[13] za konverziju primljenih podataka u napon *ard*:

```
function r=ard(n,z)
       z=double(z);
       r=[];
       jj=false;
    for i=1:max(size(n))
     if jj==true
       r=[r n(1,i)];
     end
     if n(i)==z
       jj=true;
     end
     if n(i)==double('-')
       jj=false;
       break;
     end
    end
       r=strrep(r,'-','');
       r=str2num(r)*5/1023;
end
```

Funkcija[14] za odabir *colormaps cmaps*:

```
function [s,j]=cmaps(x)
if x==1
s=colormap('Jet');
j='Jet';
elseif x==2
    s=colormap('HSV');
    j='HSV';
elseif x==3
    s=colormap('Hot');
    j='Hot';
elseif x==4
    s=colormap('Cool');
    j='Cool';
elseif x==5
    s=colormap('Spring');
    j='Spring';
elseif x==6
```

[12] GUI za ultrazvucnu kameru AQ_kartica\strukt.m
[13] Serial_com\ard.m
[14] Image_proc\cmaps.m

```
    s=colormap('Summer');
    j='Summer';
elseif x==7
    s=colormap('Autumn');
    j='Autumn';
elseif x==8
    s=colormap('Winter');
    j='Winter';
elseif x==9
    s=colormap('Copper');
    j='Copper';
elseif x==10
    s=colormap('Bone');
    j='Bone';
elseif x==11
    s=colormap('Gray');
    j='Gray';
end
end
```

S-funkcija za prikaz slike na GUI-u i detektovnje ivica na slici *sf_iaq*:

```
function sf_iaq(block)
  setup(block);
function setup(block)
  block.NumInputPorts  = 3;
  block.NumOutputPorts = 0;
  block.InputPort(1).DatatypeID   = 3; %uint8
  block.InputPort(1).Complexity   = 'Real';
  block.InputPort(1).SamplingMode = 'Sample';
  block.InputPort(2).DatatypeID   = 3; %uint8
  block.InputPort(2).Complexity   = 'Real';
  block.InputPort(2).SamplingMode = 'Sample';
  block.InputPort(3).DatatypeID   = 3;
  block.InputPort(3).Complexity   = 'Real';
  block.InputPort(3).SamplingMode = 'Sample';
  block.NumDialogPrms = 0;
  block.SimStateCompliance = 'HasNoSimState';
  block.RegBlockMethod('SetInputPortDimensions',  @SetInpPortDims);
  block.RegBlockMethod('Outputs',               @Output);
function SetInpPortDims(block,~,di)
  block.InputPort(1).Dimensions = di;
  block.InputPort(2).Dimensions = di;
  block.InputPort(3).Dimensions = di;
function Output(block)
global iaqhand;
    slika =cat(3,...
    block.InputPort(1).Data,...
    block.InputPort(2).Data, ...
    block.InputPort(3).Data);
    slika=ycbcr2rgb(slika);
    imshow(slika,'Parent',iaqhand.axes1);
    title(iaqhand.axes1,'Slika');
    gr = rgb2gray(slika);
    bw = uint8(edge(gr,'canny')*255);
    img = slika+cat(3,bw/255,bw,bw/255);
```

```
    imshow(img,'Parent',iaqhand.axes2);
    title(iaqhand.axes2,'Slika sa detektovanim ivicama');
```