Ćwiczenia VI: Akwizycja danych przez porty i sterowanie urządzeniami przez: RS232, LPT, USB, mikrofon, itd. (spektrometr, tracker)

Instrhwinfo – funkcja do informacji o dostępnych narzędziach do komunikacji z urządzeniami
https://www.mathworks.com/products/instrument/supported/gpib.html

```
% Create a serial port object.
obj1 = instrfind('Type', 'serial', 'Port', 'COM3', 'Tag', '');
% Create the serial port object if it does not exist
if isempty(obj1)
   obj1 = serial('COM3');
else
   fclose(obj1);
   obj1 = obj1(1)
end
% Connect to instrument object, obj1.
fopen(obj1);
% Communicating with instrument object, obj1.
data1 = query(obj1, '*IDN?');
% Disconnect from instrument object, obj1.
fclose(obj1);
% Clean up all objects.
delete(obj1);
```

1. Sterowanie portem LPT: Sterowanie oświetleniem, urządzeniami

```
port = digitalio('parallel', 'LPT1');
Port0=addline(port,0:7,'out');   % pins 2-9
putvalue(port.Line(1:8), vect); % put values
% ustawia Low 0 lub High 1 na odpowiednich pinach.
% 1 oznacza okolo 4.3 V
% 0 okolo 0.05 V
setvalue=getvalue(port.Line);   % get values on PINS
```

2. Sterowanie mikrofonem: detektor wyładowań atmosferycznych

```
ai = analoginput('winsound');
set(ai,'SampleRate',freq)
addchannel(ai,[1,2]);
set(ai,'TriggerDelay',0.5);
data=zeros(nrec,2);
t=[1:nrec];
set(ai,'SamplesPerTrigger',length(data))
start([ai])
[data,time,abstime,events]=getdata(ai);
```

3. Sterowanie portem RS232: AE-51, STR22 Sun Tracker, Arduinio

```
COM_PORT='COM3';
COM_PORT='/dev/ttyS0';
s=serial(COM_PORT,'BaudRate',RATE);
```

```
    set(s,'InputBufferSize',1024);
    set(s,'OutputBufferSize',1024);
    %set(s,'Terminator','LF/CR');
    set(s,'Terminator','CR');
    while (s.BytesAvailable>1)
      fscanf(s);
    end
```

4. Sterowanie portem USB: Spektrometr, USB1608FS A/D

5. TCPIP

```
% Create TCP/IP object 't'. Specify server machine and port number.
t = tcpip('www.EXAMPLE_WEBSITE.com', 80);
% Set size of receiving buffer, if needed.
set(t, 'InputBufferSize', 30000);
% Open connection to the server.
fopen(t);
% Transmit data to the server (or a request for data from the server).
fprintf(t, 'GET /');
    % Pause for the communication delay, if needed.
    pause(1)
    % Receive lines of data from server
    while (get(t, 'BytesAvailable') > 0)
     t.BytesAvailable
     DataReceived = fscanf(t)
    end
% Disconnect and clean up the server connection.
fclose(t);
delete(t);
clear t
```

6. Sterowanie portem GPIB

```
% Create a GPIB object.
obj1 = instrfind('Type', 'gpib', 'BoardIndex', 7, 'PrimaryAddress', 10, 'Tag', '');
% Create the GPIB object if it does not exist
% otherwise use the object that was found.
if isempty(obj1)
    obj1 = gpib('AGILENT', 7, 10);
    else
  fclose(obj1);
   obj1 = obj1(1)
end
    % Connect to instrument object, obj1.
    fopen(obj1);
    % Communicating with instrument object, obj1.
    data1 = query(obj1, '*IDN?');
    % Disconnect from instrument object, obj1.
    fclose(obj1);
    % Clean up all objects.
    delete(obj1);
```

7. Sterowanie spektrometrem przy użyciu zewnętrznych bibliotek Windowsowych (dll)

a) Uruchomienie spektrometru, załadowanie biblioteki

```
function StartSpectrometer()
if ~libisloaded('sdacq32mp')
  loadlibrary sdacq32mp sdacq32mp.h addheader sdacq32_error_codes.h addheader
sdacq32_types.h
end
```

libfunctionsview funkcja wyświetla strukturę biblioteki (np. sdacq32mp)

```
% otwieranie urządzenia
ret=calllib('sdacq32mp','SDACQMP_OpenOperationElectronicsDevice',1,6,1);
% inicjalizacja
ret=calllib('sdacq32mp','SDACQMP_InitializeOperationElectronics',0,1);
```

b) Zbieranie danych

```
function DATA=GetData(dt)
% liczba kanałów
nchannels=256;
Channel_ID=libpointer('int32Ptr',0);
ret=calllib('sdacq32mp','SDACQMP_AllocRawData',Channel_ID);
ret=calllib('sdacq32mp','SDACQMP_ParaSetMapping',Channel_ID,1,1);

if dt<=6500
 IntegrationTime=libpointer('doublePtr',dt);
 ret=calllib('sdacq32mp','SDACQMP_ParaSetIntegrationTime',IntegrationTime,1);
else
 IntegrationTime=libpointer('int32Ptr',dt);
 ret=calllib('sdacq32mp','SDACQMP_ParaSetIntegrationTime2',IntegrationTime,1);
end
% nrec Number of scans to averaging (1..100)

%AverageNumber=libpointer('int32Ptr',nrec);
%ret=calllib('sdacq32mp','SDACQMP_ParaSetAverageNumber',AverageNumber,1);

ret=calllib('sdacq32mp','SDACQMP_GetSpectra',1);
spectral_data=libpointer('doublePtr',zeros(nchannels,1));
ret=calllib('sdacq32mp','SDACQMP_GetStoredRawData',Channel_ID,spectral_data);
H=get(spectral_data);
DATA=H.Value;
ret=calllib('sdacq32mp','SDACQMP_FreeRawData',Channel_ID);
```

c) Sterowanie kanałami pomiarowymi za pomocą dwóch migawek przy użyciu LPT

```
if findstr(str,'sun')
  vect=[1 0 0 0 0 0 0 0];
 elseif findstr(str,'sky')
```

```
    vect=[0 1 0 0 0 0 0 0];
 elseif findstr(str,'dark')
    vect=[0 0 0 0 0 0 0 0];
 else
   disp('Wrong shutter option')
   res='ERROR';
   return
end

port=digitalio('parallel','LPT1');     % creats digital object
%get(port);
%propinfo(port);
% Add lines for the Data Interface
Port0=addline(port,0:7,'out');   % pins 2-9
putvalue(port.Line(1:8), vect); % put values
% ustawia Low 0 lub High 1 na odpowiednich pinach.
% 1 oznacza ok. 4.3 V
% 0 ok. 0.05 V

setvalue=getvalue(port.Line);   % get values on PINS
x=sum(vect-setvalue);
if x==0
   res='OK';
else
   res='ERROR';
end
```

8. Sterowanie USB1608FS A/D

```
% wymagania cbw32.dll and cbw.h
% załadowanie biblioteki
if ~libisloaded('cbw32')
  loadlibrary cbw32 cbw.h
end
%libfunctionsview cbw32    % Dostępne funkcje
DevID=0;
% Zakresy przetworników A/D
% res=1 OR (-1)  range -10 +10
% res=0          range -5 +5
% res=14 OR      range -2 +2
% res=4 OR       range -1 +1

range=0;   % voltage range
scale=10;
channel=0;
range=1;
port=0;
Slev=2;
     [a,SWITCH]=calllib('cbw32','cbDBitIn',DevID,1,port,0);
      [a,SWITCH]=calllib('cbw32','cbDBitIn',DevID,1,port,0);
     val=zeros(nrec,8);
```

```
  [a,val(i,j)]=calllib('cbw32','cbAIn',DevID,j-1,range,1);
[a,SWITCH]=calllib('cbw32','cbDBitIn',DevID,1,port,0);
[a,V(i)]=calllib('cbw32','cbAIn',DevID,8-1,range,1);
% %czyta pojedynczy port
port=0;    % od 0 do 7
[a,D]=calllib('cbw32','cbDBitIn',DevID,1,port,0)
%czyta wszystkie 8 portów na raz
[a,D]=calllib('cbw32','cbDIn',DevID,1,0)
% Konfiguracja
[a]=calllib('cbw32','cbDConfigPort',DevID,1,1)
[a]=calllib('cbw32','cbDOut',DevID,1,0)
```

9. Sterowanie aethalometrem AE-51 (sterowanie przez USB widocznym jak port szeregowy)

```
s = instrfind('Type', 'serial', 'Port', 'COM3', 'Tag', '');
% Tworzy obiekt portu szeregowego jeśli nieistnieje
if isempty(s)
   s = serial('COM3');
    else
       fclose(s);
   s = s(1)
end
% ustawienie prędkości transmisji (ewentualnie innych parametrów get(s))
set(s,'BaudRate',500000)
% Podłączenie do portu
fopen(s);
%skanowanie portu celem sprawdzenia czy są dostępne dane
if get(s,'BytesAvailable')
 % czytanie danych
str=fscanf(s);
end
% sprawdza czy długości słowa wynosi 41 – jeśli taka przetwarza dane
If length(s)==41
fprintf(s, znak)  % wysyła ciąg znak do portu
end
- Dekodowanie informacji
x=str-0;
for i=1:6
 t(i)=str(20+i-1)-0;
end

 a1=str(9)-0;
 a2=str(10)-0;
 a3=str(11)-0;
 DATA=(256*256*a3+256*a2+a1);
```

Ćwiczenia VIII – kompilator mcc i mex

- Kompilacja mcc, wymaga toolboxa, który nie jest dostępny na FUW.

- Aby skompilować program musimy zapisać go w postaci funkcji

- Przed kompilacją musimy ustawić rodzaj używanego kompilatora **`mbuild -setup`**

- opcje kompilatora:

-**x** generuje MEX file oraz kod w C

-**m** generuje plik wykonalny

-**p** generuje plik wykonalny oraz kod w C++

-**S** generuje symulinka MEX używając C

-**B** sgl generuje wykonalny plik używając C zawierający biblioteki graficzne (wymaga SGL)

-**B** generuje wykonalny plik używając C++ zawierający biblioteki graficzne (wymaga SGL)

-**B pcode** generuje p-code

Aby uruchomić skompilowany program na komputerze gdzie na ma matlaba musimy zainstalować odpowiedni do wersji MATLAB Compiler Runtime (MCR)
http://www.mathworks.com/products/compiler/mcr/

Generowanie MEX file

1. FORTRAN

Musimy napisac subrutyne, która będzie interfejsem pomiędzy kodem napisanym a Fotranie a matlabem

`nlhs`	Liczba argumentów wyjściowych lub wymiar macierzy.
`plhs`	Macierz wyjściowa.
`nrhs`	Liczba argumentów wejściowych lub wymiar macierzy
`prhs`	Macierz wejściowa.

W pierwszej linii kody wstawiamy nagłówek **fintrf.h**, zawierający deklaracje funkcji API w MATLABie

#include "fintrf.h"

Tworzymy subrutyne

subroutine mexFunction(nlhs, plhs, nrhs, prhs)
implicit none

Musimy zadeklarować zmiennie używając MATLABowego typu zmiennej mwPointer.

deklaracja argumentów mexFunction:
mwPointer plhs(*), prhs(*)
integer nlhs, nrhs

deklaracja funkcji

mwPointer mxGetPr
mwPointer mxCreateDoubleMatrix
integer mxIsNumeric
mwPointer mxGetM, mxGetN

deklaracja zmiennych lokalnych (Pointers to input/output mxArrays☺
mwPointer x_ptr, y_ptr

deklaracja macierzy

mwPointer mrows, ncols
mwSize size

Weryfikacja argumentów inputowych i outputowych

Verify the number of MEX-file input and output arguments using the nrhs and nlhs arguments. Add these statements to the mexfunction code block.

```
if(nrhs .ne. 1) then
  call mexErrMsgIdAndTxt ('MATLAB:timestwo:nInput','One input required.')
   elseif(nlhs .gt. 1) then
    call mexErrMsgIdAndTxt ('MATLAB:timestwo:nOutput','Too many output
arguments.')
   endif
```

Weryfikacja typu argumentów wejściowych.

```
if(mxIsNumeric(prhs(1)) .eq. 0) then
 call mexErrMsgIdAndTxt ('MATLAB:timestwo:NonNumeric','Input must be a number.')
endif
```

Właściwa subrutyna napisana w fortranie

```
subroutine timestwo(y_output, x_input)
real*8 x_input, y_output
y_output = 2.0 * x_input
return
end
```

Deklaracja zzmiennych dla subrutyny FOTRANowsiej

real*8 x_input, y_output

czytanie parametrów wejściowych

x_ptr = mxGetPr(prhs(1))

odczytanie rozmiaru macierzy wejściowej

mrows = mxGetM(prhs(1))
ncols = mxGetN(prhs(1))
size = mrows*ncols

Tworzenie FOTRANowskiej macierz wejsiowej
call mxCopyPtrToReal8(x_ptr,x_input,size)
Przygotowanie danych wyjściowych

Tworzenie macierzy wyjsiowej
plhs(1) = mxCreateDoubleMatrix(mrows,ncols,0)

przypisanie danych
y_ptr = mxGetPr(plhs(1))

Wykonanie obliczeń
call timestwo(y_output, x_input)

Skopiowanie wyników do argumentu wyjściowego
call mxCopyReal8ToPtr(y_output,y_ptr,size)
View Complete Source File

Kompilacja kodu
mex timestwo.F

wykonanie
timestwo(2)

2. Przykład napisany w C

```
#include "mex.h"

/*
 * timestwo.c - example found in API guide
 *
 * Computational function that takes a scalar and double it.
 *
 * This is a MEX-file for MATLAB.
 * Copyright 1984-2000 The MathWorks, Inc.
 */


void timestwo(double y[], double x[])
{
  y[0] = 2.0*x[0];
}

void mexFunction( int nlhs, mxArray *plhs[],
```

```
                int nrhs, const mxArray *prhs[] )
{
  double *x,*y;
  int     mrows,ncols;

  /* Check for proper number of arguments. */
  if(nrhs!=1) {
    mexErrMsgTxt("One input required.");
  } else if(nlhs>1) {
    mexErrMsgTxt("Too many output arguments");
  }

  /* The input must be a noncomplex scalar double.*/
  mrows = mxGetM(prhs[0]);
  ncols = mxGetN(prhs[0]);
  if( !mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]) ||
      !(mrows==1 && ncols==1) ) {
    mexErrMsgTxt("Input must be a noncomplex scalar double.");
  }

  /* Create matrix for the return argument. */
  plhs[0] = mxCreateDoubleMatrix(mrows,ncols, mxREAL);

  /* Assign pointers to each input and output. */
  x = mxGetPr(prhs[0]);
  y = mxGetPr(plhs[0]);

  /* Call the timestwo subroutine. */
  timestwo(y,x);
}
```