MonitorCEC

Software para el Monitoreo de Clima Espacial desde Colombia

Versión 1.0

Códigos Fuente

2017

Grupo de Investigación en Astroingeniería Alfa Orión
Observatorio Astronómico Universidad Tecnológica de Pereira. Edificio 13 Bloque A. Piso 7. 13A-511
Carrera 27 #10-02 Barrio Alamos - Risaralda - Colombia - AA: 97 - Código postal: 660003
Contacto: http://www.utp.edu.co/observatorioastronomico/ - obsastronomico@utp.edu.co – 314 503 8 504

```matlab
% Inicio de la aplicación
function varargout = MonitorCEC(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MonitorCEC_OpeningFcn, ...
                   'gui_OutputFcn',  @MonitorCEC_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function MonitorCEC_OpeningFcn(hObject, ~, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

%--------------------------------------------------------------------------
% Muestra el listado de los dispositivos de entrada conectados y devuelve
% la opción elegida por el usuario.
function varargout = MonitorCEC_OutputFcn(hObject, ~, handles)
    salida = audiodevinfo;
    tarjetas = {salida.output.Name};
    in = listdlg('PromptString','Selecione la entrada de sonido',...
                'SelectionMode','single',...
                'ListString',tarjetas,'ListSize',[500 200]);
    if ( isempty(in))
        set(handles.togglebutton1,'Visible','off');
        waitfor( errordlg('Debe selecionar un canal para poder iniciar la
aplicación',...
               'Alerta!','OK'));
    else
        set(handles.togglebutton1,'Visible','on');
    end
    handles.in = in;
    varargout{1} = handles.output;
    guidata(hObject,handles);

% ------------------------------------------------------------------------
function archivo_menu_Callback(hObject, eventdata, handles)

% ------------------------------------------------------------------------
function graficar_menu_Callback(hObject, eventdata, handles)

% ------------------------------------------------------------------------
function ayuda_menu_Callback(hObject, eventdata, handles)
```

```matlab
%Muestra el manual de usuario del software.
function menu_ayuda_verayuda_Callback(hObject, eventdata, handles)
    winopen('ManualUsuario.pdf')
    guidata(hObject,handles);

% --------------------------------------------------------------------
% Muestra información general del software como su nombre,
% versión, creadores e institución educativa.
function menu_ayuda_acercaSID_Callback(hObject, eventdata, handles)
    AcercaDeAp

% --------------------------------------------------------------------
% Carga los datos de un archivo en MS Excel y genera la gráfica
% correspondiente según los datos cargados
% Se pintan las regiones de la gráfica según el horario que el grupo de
% datos represente (tomados del archivo Excel).
% 'AZUL' representa la noche y 'AMARILLO' representa el día.
function menu_graficar_cargar_Callback(hObject, eventdata, handles)
    [FileName, Path] = uigetfile({'*.xlsx'},'Abrir Documento');
    NameSave = ([Path FileName]);
    if NameSave == 0
        return
    else
        SG = xlsread(NameSave,1,'B1:B17281');
    end
    h = 17280;
    SG(1)= (SG(2)+SG(3))/2;
    SG(17280)= (SG(17279)+SG(17278))/2;
    for i=2:h-1
        if (i~=1 || i~=h)
            if (SG(i)==0 )
                SG(i)= (SG(i-1)+SG(i+1))/2;
            end
        end
    end
    windowSize = 10;
    SG=filter(ones(1,windowSize)/windowSize,1,SG);
    ejex =linspace(0,24,17280);
    SG=SG';
    j = size(ejex);
    v = size(SG);
    figure(1)
    hold off
    plot(ejex(1:4321),SG(3600:7920),'b')
    hold on
    plot(ejex(4321:12960),SG(7921:16560),'y')
    hold on
    plot(ejex(12961:13680),SG(16561:17280),'b')
    hold on
    plot(ejex(13681:17280),SG(1:3600),'b')
    min(SG);
    ylim([min(SG)-2 max(SG)+1])
    xlabel ('Time GMT-5')
    ylabel ('Power Spectrum(dB)')
    grid on
    hold on
```

```matlab
    guidata(hObject,handles);
% --------------------------------------------------------------------
function menu_graficar_ver_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
% Toma un pantallazo de la imagen actual que el software este mostrando y
% lo guarda en formato png.
function menu_archivo_nuevo_Callback(hObject, eventdata, handles)
    X = getframe(gca);
    X = frame2im(X);
    [FileName, PathName] = uiputfile('*.png', 'Save As');
    if FileName == 0
        return
    else
        imwrite(X , FileName ,'png');
    end
    guidata(hObject,handles);

% --------------------------------------------------------------------
function menu_archivo_abrir_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
% Toma un pantallazo de la imagen actual que el software este mostrando
% y se genera una imagen preliminar para su impresión.
function menu_archivo_imprimir_Callback(hObject, eventdata, handles)
    X = getframe(gca);
    printpreview()
    guidata(hObject,handles);

% --------------------------------------------------------------------
% Función para salir de la plataforma
function menu_archivo_salir_Callback(hObject, eventdata, handles)
    close all;

% --------------------------------------------------------------------
% Muestra la pestaña con la opción de ver la señal original
function menu_graficar_versenal_Callback(hObject, eventdata, handles)
    global l1;
    l1 = 1;
    hold off;
    guidata(hObject,handles);

% --------------------------------------------------------------------
% Muestra la pestaña con la opción de ver la señal DPS
function menu_graficar_verDPS_Callback(hObject, eventdata, handles)
    global l1;
    l1=0;
    hold off;
    guidata(hObject,handles);
% --------------------------------------------------------------------
function popupmenu1_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function popupmenu1_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% ----------------------------------------------------------------------
function pushbutton1_Callback(hObject, eventdata, handles)



      ------------------- FUNCIÓN PRINCIPAL --------------------------


function togglebutton1_Callback(hObject, eventdata, handles)
    global NAA fecha3 dia2 k i l1;
    NAA(1:17280) = zeros;
    %Verificamos el estado del toogle button
    k = get(handles.togglebutton1, 'Value');
        if (k == 1)
            set(handles.togglebutton1, 'String', 'Detener Proceso');
            set(handles.pushbutton11,'Visible','on');
        else
            set(handles.togglebutton1, 'String', 'Iniciar Proceso');
            set(handles.pushbutton11,'Visible','off');
        end
    set(handles.axes1,'Visible','on');
    grid on;
    d = daq.getDevices;
    disp = d(handles.in).ID;
    AI = daq.createSession('directsound');

    % Adicionamos el canal
    addAudioInputChannel(AI , disp , 1:2);
    AI.UseStandardSampleRates = false;
    % Configuramos el tiempo de muestreo y la duración
    %AI.DurationInSeconds = 1;    % Periodo en segundos
    AI.Rate = 96000; % Máximo tmuestreo soportado
    l1 = 0;

    %Inicializamos variables para el cálculo de la DPS
    %para ahorrar memoria en Matlab
    pxx = zeros(2501,1);
    f = zeros(2501,1);
    Pxx = zeros(2501,1);
    while(k == 1)
    % Cálculamos el valor de tiempo donde deben almacenarse los datos
        c = round(clock);
            while( mod(c(6),5) || c(6) == 0);
                c = round(clock);
            end
        cc = c  + [0 0 0 5 0 0];  % cc toma la hora dada por la señal de
reloj y le suma y 5 horas.
        if(c(6) == 60)
            c(6) = 0;
            c(5) = c(5) + 1;
            if(c(5) == 60)
                c(5) = 0;
                c(4) = c(4) + 1;
```

```
            end
        end;

%Convierte la hora solapada a una fecha valida haciendo de la hora 24 un
nuevo día con horas en 0s. Ej. 19:00:00 al sumar 5 horas serán las 24:00
del día N: esta hora es la 00:00:00 del día N+1

        cc = datevec(datetime(cc));

% Calculamos la variable i para saber en que punto de la tabla Excel
% metemos los nuevos valores calculados donde:
% c(4)= horas,     c(5)= minutos,     c(6)= segundos
        i = (((cc(4)*60*60)+(cc(5)*60)+cc(6))/5);

        %% CÁLCULO DE LA HORA Y FECHA EN TIEMPO REAL
        %Ajustamos la fecha y hora en tiempo real
        ano = num2str(c(1));
        mes = num2str(c(2));
        dia = num2str(c(3));

        %Si el mes o el día es de solo un dato entonces le pones un cero
        %para completarlo
        if (length(mes)==1)
            mes = strcat('0',mes);
        end
        if (length(dia)==1)
            dia = strcat('0',dia);
        end

        % Guardamos la fecha real
        fecha3 = [ano,'-',mes,'-',dia];
        dia2 = [int2str(c(4)),':',int2str(c(5)),':',int2str(c(6))];

        %Guardamos datos almacenados en el día
        if i == 0
            i = 17280;
        end
        my_cell = strcat('A',num2str(i));
        [~,struc] = fileattrib;
        PathCurrent = struc.Name;
        PathFolder = ([PathCurrent '/Datos/']);
        buffer_name = strcat('Buffer_Actual','.xlsx');
        NameBF = ([PathFolder (buffer_name)]);
        senales = {'NAA'};
        xlswrite(NameBF, {datestr([fecha3,' ',dia2,' '])} ,1 , my_cell);
        xlswrite(NameBF, NAA', 1, 'B1');

        % Configuramos la tarjeta para iniciar la adquisición
        % Guarda los datos de la memoria en una variable en el espacio de
      trabajo
        data =  startForeground(AI);
        wait(AI , AI.DurationInSeconds);
        handles.data = data;

        % Vector eje X para graficar la señal original
```

```
        if (l1 == 1)
            N = length(data);
            t(N) = zeros;
            parfor j = 1:N
                t(j) = j;
            end
        set(handles.axes1,'Visible','off');
        set(handles.axes2,'Visible','on');
        %axes(handles.axes2);
        plot(t,data)
        hold off
        grid on
        end



%% GRAFICA DE LA SEÑAL ACTUAL
        %% CÁLCULO Y GRÁFICA DE LA DENSIDAD ESPECTRAL  DE POTENCIA
        % mediante la aproximación pwelch con oversampling = 1024
        [pxx , f] = pwelch(data , 5000 , 1024 , 5000, AI.Rate);
        handles.f = f;
        Pxx = 10.4*log10(pxx)+143;
        handles.Pxx = Pxx;
        guidata(hObject,handles);

        % Graficamos la DSP
        if (l1==0)
            set(handles.axes2,'Visible','off');
            set(handles.axes1,'Visible','on');
            axes(handles.axes1);
            plot(f,Pxx,'m')
            hold off

set(gca,'XTickLabel',{'0','5000','10000','15000','20000','25000','30000',
'35000','40000','45000','50000'})
            grid on
            xlabel('Frequency (Hz)'); ylabel('Power Spectrum (dB)');
        end

%% FIN DEL CÁLCULO DE LA DENSIDAD ESPECTRAL DE POTENCIA




%% ALMACENAMIENTO DE LA SEÑAL DE INTERÉS (NAA)

       NAA(i) = Pxx(1251);
       if (NAA(i) <= 0)
            NAA(i) = 0;
       end

       %Copia los datos del buffer y los guarda en otro archivo Excel que
       %será el definitivo del día, estos datos se guardaran a las 7pm y
%el archivo buffer se reiniciará.
```

```
        if(c(4) == 19 && c(5) == 0 && c(6) == 0)
             utp_name = strcat('UTP_', senales, '_', ano, '-', mes, '-',
dia, '.xlsx');
             P = xlsread(NameBF,1,'B17279:B17279');
             xlswrite(NameBF,P,1,'B17280');
             buffer2utp(utp_name);
             NAA(1:17280) = zeros;
             xlswrite('Datos/Buffer_Actual.xlsx',zeros,1,'A1:A17280');
        end
        texto =['    ',fecha3,'
',int2str(c(4)),':',int2str(c(5)),':',int2str(c(6)),' [',int2str(i),']
NAA=', int2str(floor(NAA(i)))];
        set(handles.text1, 'String',texto); % Imprimimos los datos en la
barra inferior del programa
        k = get(handles.togglebutton1, 'Value'); % Verifica el estado del
toogle button
    end
% Eliminamos el canal de recepción de datos.
    delete(AI);
    set(handles.text1, 'String','   Esperando Datos'); % Imprimimos los
datos en la barra inferior del programa
    guidata(hObject,handles);

      %% -------------- FIN FUNCIÓN PRINCIPAL -------------------

% ----------------------------------------------------------------------
%Copia los datos del buffer actual y los guarda en otro archivo Excel que
%será el definitivo del día.
function buffer2utp(utp_name)
    [~,struc] = fileattrib;
    PathCurrent = struc.Name;
    PathFolder = strcat(PathCurrent, '\Datos\');
    buffer_name = strcat('Buffer_Actual','.xlsx');
    nameBF = strcat(PathFolder, buffer_name);
    disp(utp_name);
    nameUTP = strcat(PathFolder, utp_name);
    disp(nameUTP);
    [~, ~, raw] = xlsread(nameBF);
    xlswrite(char(nameUTP), raw);

% ----------------------------------------------------------------------
function pb_with_bg_Callback(hObject, eventdata, handles)

% ----------------------------------------------------------------------
function pushbutton3_Callback(hObject, eventdata, handles)

% ----------------------------------------------------------------------
function togglebutton1_CreateFcn(hObject, eventdata, handles)

% ----------------------------------------------------------------------
function togglebutton2_Callback(hObject, eventdata, handles)

% ----------------------------------------------------------------------
function togglebutton3_Callback(hObject, eventdata, handles)

% ----------------------------------------------------------------------
```

```matlab
function radiobutton1_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function radiobutton2_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function pushbutton4_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function radiobutton3_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function edit1_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function edit1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --------------------------------------------------------------------
function pushbutton5_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function uitoggletool1_ClickedCallback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function edit2_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function edit2_CreateFcn(hObject, eventdata, handles)

% --------------------------------------------------------------------
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --------------------------------------------------------------------
function pushbutton7_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function pushbutton8_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function checkbox1_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function edit3_Callback(hObject, eventdata, handles).



function edit3_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --------------------------------------------------------------------
function config_menu_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
% Muestra el listado de los dispositivos de entrada conectados y devuelve
% la opción elegida por el usuario.
function  menu_configurar_audioinput_Callback(hObject,~, handles)
    salida = audiodevinfo;
    tarjetas = {salida.output.Name};
    in = listdlg('PromptString','Selecione la entrada de sonido',...
                'SelectionMode','single',...
                'ListString',tarjetas,'ListSize',[500 200]);
    if ( isempty(in))
        set(handles.togglebutton1,'Visible','off');
        waitfor( errordlg('Debe selecionar un canal para poder iniciar la
aplicación',...
               'Alerta!','OK'));
    else
        set(handles.togglebutton1,'Visible','on');
    end
    handles.in = in;
    varargout{1} = handles.output;
    guidata(hObject,handles);

% --------------------------------------------------------------------
function figure1_ResizeFcn(hObject, eventdata, handles)

% --------------------------------------------------------------------
function tools_menu_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
%Herramienta "Mover" del software
function menu_tools_mover_Callback(hObject, eventdata, handles)
    axes(handles.axes1)
    zoom off
    pan on
    guidata(hObject,handles);



% --------------------------------------------------------------------
%Herramienta "Zoom" del software
function menu_tools_zoom_Callback(hObject, eventdata, handles)
    axes(handles.axes1)
    zoom on
    pan off
    guidata(hObject,handles);




%Herramienta "Cursor" del software
```

```
function menu_tools_cursor_Callback(hObject, eventdata, handles)
    axes(handles.axes1)
    datacursormode on
    guidata(hObject,handles);

% --------------------------------------------------------------------
function pushbutton10_Callback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function uipushtool4_ClickedCallback(hObject, eventdata, handles)

% --------------------------------------------------------------------
function uitoggletool9_OnCallback(hObject, eventdata, handles)
    global l1;
    l1=1;
    hold off;
    guidata(hObject,handles);

% --------------------------------------------------------------------
function uitoggletool9_OffCallback(hObject, eventdata, handles)
    global l1;
    l1=0;
    hold off;
    guidata(hObject,handles);

%Cierra el programa
function figure1_CloseRequestFcn(hObject, eventdata, handles)
    opc = questdlg('¿Desea salir del programa?',...
              'SALIR','Si','No','No');
    if strcmp(opc,'No')
              return;
    end
    delete(hObject);

% --------------------------------------------------------------------
function uipushtool3_ClickedCallback(hObject, eventdata, handles)
    [FileName, Path] = uigetfile({'*.xlsx'},'Abrir Documento');
    NameSave = ([Path FileName]);
    SG = xlsread(NameSave,1, 'C2:C17281');
    ejex =linspace(0,24,17280);
    SG=SG';
    figure(1)
    hold off
    plot(ejex(1:4320),SG(1:4320),'b')
    hold on
    plot(ejex(4321:12961),SG(4321:12961),'y')
    hold on
    plot(ejex(12962:17280),SG(12962:17280),'b')
    grid on
    hold on
    guidata(hObject,handles);



function uipushtool5_ClickedCallback(hObject, eventdata, handles)
```

```
    X=getframe(gca);
    X=frame2im(X);
    [FileName, PathName] = uiputfile('*.png', 'Save As');
    if FileName == 0
        return
    else
        imwrite(X,FileName,'png')
    end
    guidata(hObject,handles);

% --------------------------------------------------------------------
function Open_ClickedCallback(hObject, eventdata, handles)
  [FileName, Path] = uigetfile({'*.xlsx'},'Abrir Documento');
    NameSave = ([Path FileName]);
    if NameSave == 0
        return
    else
        SG = xlsread(NameSave,1,'B1:B17281');
    end
    h = 17280;
    SG(1)= (SG(2)+SG(3))/2;
    SG(17280)= (SG(17279)+SG(17278))/2;
    for i=2:h-1
        if (i~=1 || i~=h)
            if (SG(i)==0 )
                SG(i)= (SG(i-1)+SG(i+1))/2;
            end
        end
    end
    windowSize = 10;
    SG=filter(ones(1,windowSize)/windowSize,1,SG);
    ejex =linspace(0,24,17280);
    SG=SG';
    j = size(ejex);
    v = size(SG);
    figure(1)
    hold off
    plot(ejex(1:4321),SG(3600:7920),'b')
    hold on
    plot(ejex(4321:12960),SG(7921:16560),'y')
    hold on
    plot(ejex(12961:13680),SG(16561:17280),'b')
    hold on
    plot(ejex(13681:17280),SG(1:3600),'b')
    min(SG);
    ylim([min(SG)-2 max(SG)+1])
    xlabel ('Time GMT-5')
    ylabel ('Power Spectrum(dB)')
    grid on
    hold on
    guidata(hObject,handles);




function uipushtool2_ClickedCallback(hObject, eventdata, handles)
```

```
%% Se debe imprimir la imagen
    X=getframe(gca);
    printpreview()
    guidata(hObject,handles);

% --------------------------------------------------------------------
function pushbutton11_Callback(hObject, eventdata, handles)
    figure(2)
    plot(handles.f,handles.Pxx,'m')
    hold off

set(gca,'XTickLabel',{'0','5000','10000','15000','20000','25000','30000',
'35000','40000','45000','50000'})
    grid on
    xlabel('Frequency (Hz)'); ylabel('Power Spectrum (dB)');
    guidata(hObject,handles);

% --------------------------------------------------------------------
function axes2_CreateFcn(hObject, eventdata, handles)

% --------------------------------------------------------------------
function axes2_DeleteFcn(hObject, eventdata, handles)

% --------------------------------------------------------------------
function axes2_ButtonDownFcn(hObject, eventdata, handles)
```