Software Requirements Specification

BE Projects 2011-12

<Autonomous Security Guard Robot>

<Group 15>

<Rohit Vijayaraghavan>

<Ayush Sanghvi>

<MCG Karthik>

<Sahil Bareja>

College: Vidyalankar Institute of Technology

Mentor - Saurav Shandilya

Guide – Dr. Anjali Deshpande

Table Of Contents

1. Introduction:

Humans have always felt very possessive of their belongings and now-a-days due to advancement of civilization witnessed better and improved means of home security system being implemented by humans.

This device is capable of sensing many different types of gases. The sensor used is capable of sensing gases like smoke, liquefied gas, butane and propane, Methane, alcohol, hydrogen, etc. Along with gas detection, a face detection algorithm also helps in identifying a person from a digital image, by comparing selected facial features from the image and a facial database.

2.Problem Statement:

The aim of the project is to design an autonomous robot which can be used to detect a gas leak as well as for face recognition.

A gas detector also sounds an alarm in the area where the leak is occurring, giving them the opportunity to leave the area or bring the gas leak under check.

This type of device can be used widely in industry as well as in a variety of locations such as on oil rigs, to monitor manufacture processes and the facial recognition system helps identifying or verifying a person from a digital image, hence making it a sophisticated security bot. It can also be used in home security systems.

3. Requirements:

A) **Hardware Requirements**:

1. Spark V : Requires 1 bot for motion
2. ATMEGA16 : Microcontroller IC
3. Motor Driver(L923D): To drive 2 servo motors
4. Servo Motors: Used to maintain high accuracy
5. Gas sensor module : Used to detect any type of gas
6. Proximity Sensor : To measure the distance from obstacles

B) Software Requirements:

1. BASCOM: For AVR
2. AVR Boot Loader: To burn HEX file into the IC

4.Implementation:

A) Functionality:

Obstacle Detection: Spark V robot’s motion will be determined by 3 proximity sensors. Once an obstacle comes within the range of the proximity sensors, accurate distance can be measured from the obstacle and direction of motion of the bot can be changed,preventing the collision between the robot and the obstacle.

Gas Detection: Many different types of gases can be detected by the gas detector module.When any gas comes in contact with the gas sensor, the buzzer will be activated.

Face detection: In the facial recognition branch of biometrics, eigenfaces provide a means of applying data compression to faces for identification purposes. In this method, processed images of faces can be seen as vectors whose components are the brightness of each pixel. The dimension of this vector space is the number of pixels. The eigenvectors of the covariance matrix associated with a large set of normalized pictures of faces are called eigenfaces.

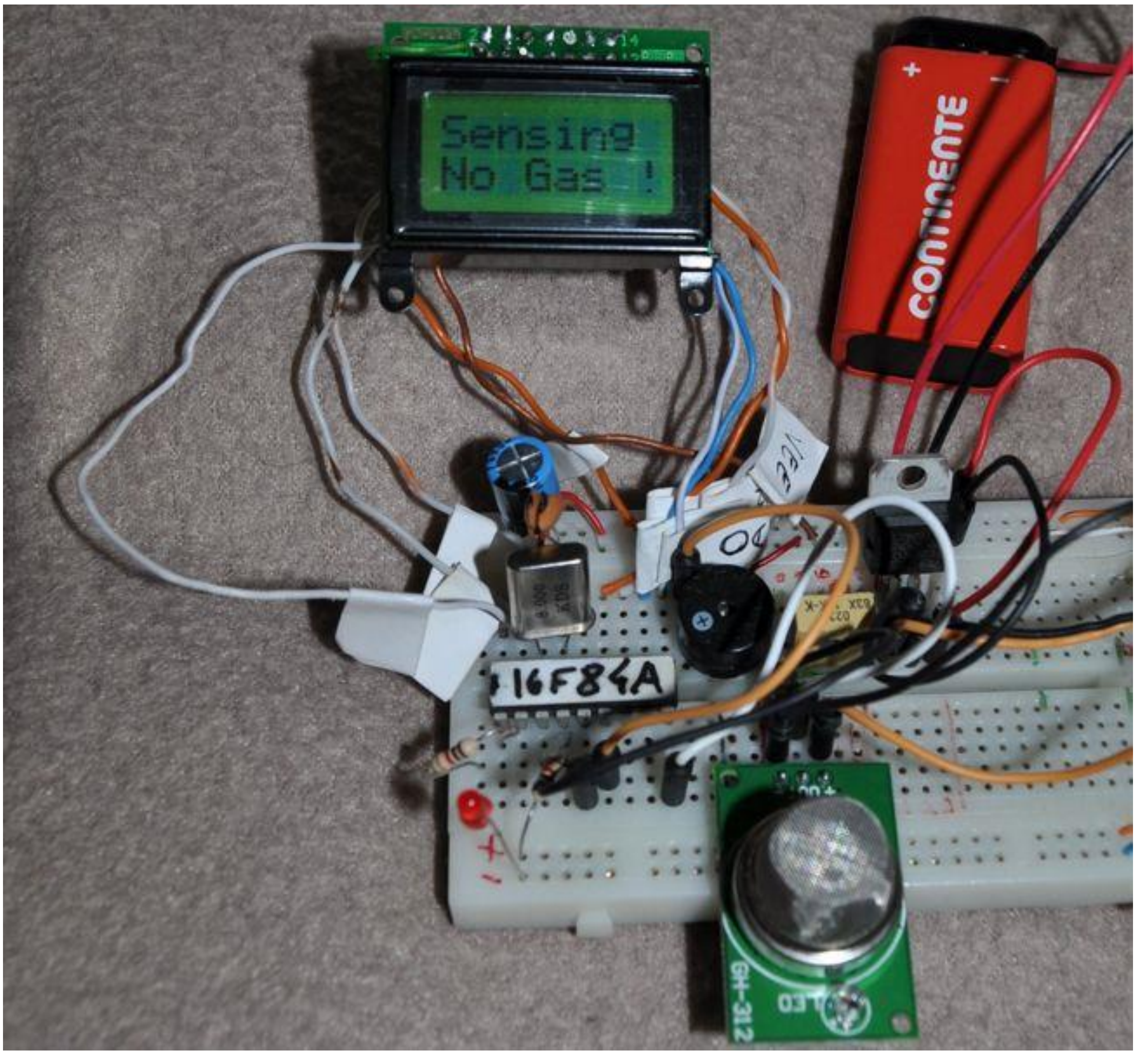
4. Testing Strategy and Data:

Testing of Gas Detector:

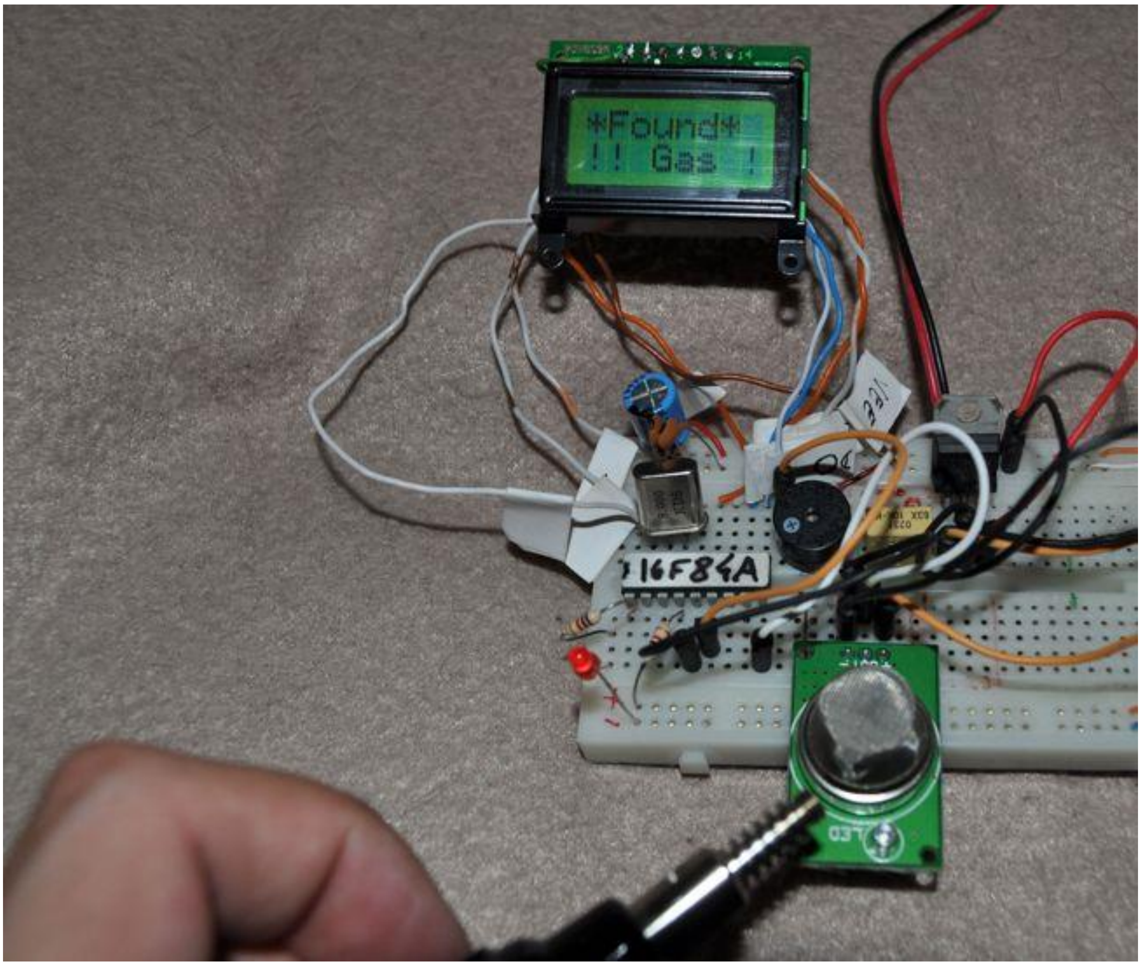
The first tests were made with the circuit mounted on a breadboard.

After initialization the circuit will enter a normal state where it detects no gas.

The display shows "Sensing...No Gas !”.



To test the sensor I used my portable gas soldering iron with the gas coming out pointed to the sensor. The sensor is able to detect the gas and the microcontroller will trigger a flashing led warning and sound. The sound is produced by a small piezo and the display show the message "Found Gas".



When the air is clean again and the sensor does not sense any gas, the circuit will return to it's normal state turning off both led and piezo sound.

Face Recognition Algorithm:

In the face recognition algorithm, first a database is created and then the required data is added to the database. Then the database graphical user interface is created which allows the user to enter the desired data into the database.

The Fisher face core is used for facial recognition. In this the FLD features of test image are extracted. The Euclidean distance between the projected test image and the images in the database is calculated.

//Face recognition codes:

A. Creating Database:

```
function T = CreateDatabase(TrainDatabasePath)

TrainFiles = dir(TrainDatabasePath);
Train_Number = 0;

for i = 1:size(TrainFiles,1)
   if
not(strcmp(TrainFiles(i).name,'.')|strcmp(TrainFiles(i).name,'..')|strcmp(T
rainFiles(i).name,'Thumbs.db'))
       Train_Number = Train_Number + 1; % Number of all images in the
training database
   end
end

%%%%%%%%%%%%%%%%%%%%%%%% Construction of 2D
matrix from 1D image vectors
T = [];
for i = 1 : Train_Number

   % I have chosen the name of each image in databases as a
corresponding
   % number. However, it is not mandatory!
   str = int2str(i);
   str = strcat('\',str,'.jpg');
   str = strcat(TrainDatabasePath,str);

   img = imread(str);
   img(:,:,1)=histeq(img(:,:,1));
   img(:,:,2)=histeq(img(:,:,2));
   img(:,:,3)=histeq(img(:,:,3));

   img = rgb2gray(img);
  % img=histeq(img);
   [irow icol] = size(img);

   temp = reshape(img',irow*icol,1);   % Reshaping 2D images into 1D
image vectors
```

```
    T = [T temp]; % 'T' grows after each turn
end

T = double(T);
```

B. Adding Data in Database:

```
function rec=Add2Database( img1,name )

    fcdb='fc_database.dat';
       if (exist(fcdb,'file')==2)
            load(fcdb,'-mat');
              fc_no=fc_no+1;
                newfile=0;
        else
              newfile=1;
              fc_no=1;
        end
rec=fc_no;


% [id1 im1]=getImgMatch(img1,'trFcdb');
% [id2 im2]=getImgMatch(img2,'trFcdb');

pname{fc_no,1}=name;
fname{fc_no,1}=strcat('trFcdb/',int2str(rec),'.jpg');
if(newfile==1)
    save(fcdb,'fname','pname','fc_no');
    newfile=0;
else
    save(fcdb,'fname','pname','fc_no','-append');
end

% imwrite(img1,strcat('trFcdb/',int2str(((rec-1)*2)+1),'.jpg'));
 imwrite(img1,strcat('trFcdb/',int2str(rec),'.jpg'));
 rec=fc_no;


end
```

C. DataBase GUI:

```
      function varargout = DBgui(varargin)
clc;
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @DBgui_OpeningFcn, ...
                   'gui_OutputFcn',  @DBgui_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
% End initialization code - DO NOT EDIT


% --- Executes just before DBgui is made visible.
function DBgui_OpeningFcn(hObject, eventdata, handles, varargin)

handles.video = videoinput('winvideo', 1,'YUY2_320x240');
set(handles.video,'TimerPeriod', 0.01, ...
'TimerFcn',['if(~isempty(gco)),'...
'handles=guidata(gcf);'... % Update handles
'imshow(ycbcr2rgb(getsnapshot(handles.video)));rectangle(''Position'',[70 20 179 199],''EdgeColor'',''r'');'... % Get picture using GETSNAPSHOT and put it into axes using IMAGE
'set(handles.VideoCam,''ytick'',[],''xtick'',[]),'... % Remove tickmarks and labels that are inserted when using IMAGE
'else '...
'delete(imaqfind);'... % Clean up - delete any image acquisition objects
'end']);
triggerconfig(handles.video,'manual');
handles.video.FramesPerTrigger = Inf;
```

```
guidata(hObject, handles);
uiwait(handles.DBgui);


% --- Outputs from this function are returned to the command line.
function varargout = DBgui_OutputFcn(hObject, eventdata, handles)
handles.output = hObject;
varargout{1} = handles.output;


% --- Executes on button press in startStopCamera.
function startStopCamera_Callback(hObject, eventdata, handles)
set(handles.Msg,'String','');
if strcmp(get(handles.startStopCamera,'String'),'Start Camera')
    set(handles.startStopCamera,'String','Stop Camera')
    axes(handles.VideoCam);
    start(handles.video);
else
    set(handles.startStopCamera,'String','Start Camera')
    axes(handles.VideoCam);
    stop(handles.video);
end




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

% Hints: get(hObject,'String') returns contents of pname as text
%        str2double(get(hObject,'String')) returns contents of pname as a
double


% --- Executes during object creation, after setting all properties.
function pname_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pname (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes when user attempts to close DBgui.
function DBgui_CloseRequestFcn(hObject, eventdata, handles)
if strcmp(get(handles.startStopCamera,'String'),'Stop Camera')
stop(handles.video);
end
delete(hObject);



% --- Executes on button press in DB.
function DB_Callback(hObject, eventdata, handles)
name=get(handles.pname,'String');
if ~strcmp(name,'')
    no=Add2Database(handles.face1,name);

    set(handles.Msg,'String',strcat('New record added at:-',int2str(no)));
else
    set(handles.Msg,'String','Please enter Person name');
end




% --- Executes on button press in captureFace.
function captureFace_Callback(hObject, eventdata, handles)
if strcmp(get(handles.startStopCamera,'String'),'Stop Camera')

    face=ycbcr2rgb(getsnapshot(handles.video));
    face=imcrop(face,[70,20,179,199]);
    stop(handles.video);
    if strcmp(get(handles.Fcnt,'String'),'0')
```

```
        axes(handles.FaceAxes1);
        handles.face1=face;
        guidata(hObject, handles);
        imshow(handles.face1);
        set(handles.Fcnt,'String','0');
    %else if strcmp(get(handles.Fcnt,'String'),'1')
    %    axes(handles.FaceAxes2);
    %    handles.face2=face;
    %    guidata(hObject, handles);
    %    imshow(handles.face2);
    %    set(handles.Fcnt,'String','0');
    %    end
    end


    axes(handles.VideoCam);
    start(handles.video);

else
    set(handles.Msg,'String','Please Turn ON the camera');
end
%guidata(hObject, handles);



% --- Executes on button press in browseFingerprint.
function browseFingerprint_Callback(hObject, eventdata, handles)
!sgdx
if strcmp(get(handles.startStopCamera,'String'),'Start Camera')

   %
[namefile,pathname]=uigetfile({'*.bmp;*.tif;*.tiff;*.jpg;*.jpeg;*.gif','IMAGE
Files (*.bmp,*.tif,*.tiff,*.jpg,*.jpeg,*.gif)'});
img=imread('new.bmp');
if (exist('new.bmp','file')==2)
    delete('new.bmp');
end
if size(img,3)==3
    img=rgb2gray(img);
end
handles.finger1=img;
```

```
guidata(hObject, handles);
axes(handles.FingerAxes1);
imshow(handles.finger1);
axes(handles.VideoCam);

else
stop(handles.video);
%[namefile,pathname]=uigetfile({'*.bmp;*.tif;*.tiff;*.jpg;*.jpeg;*.gif','IMAGE Files (*.bmp,*.tif,*.tiff,*.jpg,*.jpeg,*.gif)'});
%img=imread(strcat(pathname,namefile));
img=imread('new.bmp');
if (exist('new.bmp','file')==2)
   delete('new.bmp');
end
if size(img,3)==3
   img=rgb2gray(img);
end
handles.finger1=img;
guidata(hObject, handles);
axes(handles.FingerAxes1);
imshow(handles.finger1);
axes(handles.VideoCam);
start(handles.video);

end
%guidata(hObject, handles);



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

% Hints: get(hObject,'String') returns contents of Fcnt as text
%        str2double(get(hObject,'String')) returns contents of Fcnt as a double


% --- Executes during object creation, after setting all properties.
```

```
function Fcnt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Fcnt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



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

% Hints: get(hObject,'String') returns contents of Pcnt as text
%        str2double(get(hObject,'String')) returns contents of Pcnt as a
double


% --- Executes during object creation, after setting all properties.
function Pcnt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Pcnt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in browse.
function browse_Callback(hObject, eventdata, handles)
if strcmp(get(handles.startStopCamera,'String'),'Start Camera')

[namefile,pathname]=uigetfile({'*.jpg;*.jpeg','IMAGE Files (*.jpg,*.jpeg)'});
img=imread(strcat(pathname,namefile));
[r c z]=size(img);
if r>200 && c>180
   img=imcrop(img,[70,20,179,199]);
end
 if strcmp(get(handles.Fcnt,'String'),'0')
       axes(handles.FaceAxes1);
       handles.face1=img;
       guidata(hObject, handles);
       imshow(handles.face1);
       set(handles.Fcnt,'String','0');
   %else if strcmp(get(handles.Fcnt,'String'),'1')
   %    axes(handles.FaceAxes2);
   %    handles.face2=img;
   %    guidata(hObject, handles);
   %    imshow(handles.face2);
   %    set(handles.Fcnt,'String','0');
   %    end
 end
axes(handles.VideoCam);

else
stop(handles.video);
[namefile,pathname]=uigetfile({'*.jpg;*.jpeg','IMAGE Files (*.jpg,*.jpeg)'});
img=imread(strcat(pathname,namefile));
[r c z]=size(img);
if r>200 && c>180
   img=imcrop(img,[70,20,179,199]);
end

   if strcmp(get(handles.Fcnt,'String'),'0')
       axes(handles.FaceAxes1);
       handles.face1=img;
       guidata(hObject, handles);
       imshow(handles.face1);
```

```
        set(handles.Fcnt,'String','0');
    %else if strcmp(get(handles.Fcnt,'String'),'1')
    %    axes(handles.FaceAxes2);
    %    handles.face2=img;
    %    guidata(hObject, handles);
    %    imshow(handles.face2);
    %    set(handles.Fcnt,'String','0');
    %    end
    end

axes(handles.VideoCam);
start(handles.video);

end
%guidata(hObject, handles);
```

D. Fisher face Core:

```
function [m_database V_PCA V_Fisher ProjectedImages_Fisher] =
FisherfaceCore(T)

Class_number = ( size(T,2) ); % Number of classes (or persons)
Class_population = 1; % Number of images in each class
P = Class_population * Class_number; % Total number of training
images

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% calculating the mean
image
m_database = mean(T,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Calculating the
deviation of each image from mean image
A = T - repmat(m_database,1,P);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Snapshot method of
Eigenface algorithm
L = A'*A; % L is the surrogate of covariance matrix C=A*A'.
[V D] = eig(L); % Diagonal elements of D are the eigenvalues for both
L=A'*A and C=A*A'.
```

```
%%%%%%%%%%%%%%%%%%%%%%%% Sorting and eliminating
small eigenvalues
L_eig_vec = [];
for i = 1 : P%-Class_number
    L_eig_vec = [L_eig_vec V(:,i)];
end

%%%%%%%%%%%%%%%%%%%%%%%% Calculating the
eigenvectors of covariance matrix 'C'
V_PCA = A * L_eig_vec; % A: centered image vectors

%%%%%%%%%%%%%%%%%%%%%%%% Projecting centered
image vectors onto eigenspace
% Zi = V_PCA' * (Ti-m_database)
ProjectedImages_PCA = [];
for i = 1 : P
    temp = V_PCA'*A(:,i);
    ProjectedImages_PCA = [ProjectedImages_PCA temp];
end

%%%%%%%%%%%%%%%%%%%%%%%% Calculating the mean of
each class in eigenspace
m_PCA = mean(ProjectedImages_PCA,2); % Total mean in eigenspace
m = zeros(P,Class_number);
Sw = zeros(P,P); % Initialization os Within Scatter Matrix
Sb = zeros(P,P); % Initialization of Between Scatter Matrix

for i = 1 : Class_number
    m(:,i) = mean( ( ProjectedImages_PCA(:,i) ), 2 )';

    S  = zeros(P,P);
   % for j = i : i
        S = S + (ProjectedImages_PCA(:,i)-
m(:,i))*(ProjectedImages_PCA(:,i)-m(:,i))';
    %end

    Sw = Sw + S; % Within Scatter Matrix
    Sb = Sb + (m(:,i)-m_PCA) * (m(:,i)-m_PCA)'; % Between Scatter
Matrix
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%% Calculating Fisher discriminant basis's
% We want to maximise the Between Scatter Matrix, while minimising the
% Within Scatter Matrix. Thus, a cost function J is defined, so that this condition is satisfied.
[J_eig_vec, J_eig_val] = eig(Sb,Sw); % Cost function J = inv(Sw) * Sb
J_eig_vec = fliplr(J_eig_vec);

%%%%%%%%%%%%%%%%%%%%%%%% Eliminating zero eigens and sorting in descend order
for i = 1 : Class_number-1
   V_Fisher(:,i) = J_eig_vec(:,i); % Largest (C-1) eigen vectors of matrix J
end

%%%%%%%%%%%%%%%%%%%%%%%% Projecting images onto Fisher linear space
% Yi = V_Fisher' * V_PCA' * (Ti - m_database)
for i = 1 : Class_number*Class_population
   ProjectedImages_Fisher(:,i) = V_Fisher' * ProjectedImages_PCA(:,i);
End
```

E. MainGUI:

```
    function varargout = MainGUI(varargin)
clc;

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MainGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @MainGUI_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
% End initialization code - DO NOT EDIT


% --- Executes just before MainGUI is made visible.
function MainGUI_OpeningFcn(hObject, eventdata, handles, varargin)

handles.video = videoinput('winvideo', 1,'YUY2_320x240');
set(handles.video,'TimerPeriod', 0.01, ...
'TimerFcn',['if(~isempty(gco)),'...
'handles=guidata(gcf);'... % Update handles
'imshow(ycbcr2rgb(getsnapshot(handles.video)));rectangle(''Position'',[70 20 179 199],''EdgeColor'',''r'');'... % Get picture using GETSNAPSHOT and put it into axes using IMAGE
'set(handles.VideoCam,''ytick'',[],''xtick'',[]),'... % Remove tickmarks and labels that are inserted when using IMAGE
'else '...
'delete(imaqfind);'... % Clean up - delete any image acquisition objects
'end']);
triggerconfig(handles.video,'manual');
handles.video.FramesPerTrigger = Inf;

guidata(hObject, handles);
%startStopCamera_Callback(hObject, eventdata, handles);
% UIWAIT makes MainGUI wait for user response (see UIRESUME)
 uiwait(handles.MainGUI);
%uiwait(handles.MainGUI);


% --- Outputs from this function are returned to the command line.
function varargout = MainGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
handles.output = hObject;
varargout{1} = handles.output;


% --- Executes on button press in startStopCamera.
function startStopCamera_Callback(hObject, eventdata, handles)
set(handles.Msg,'String','');
if strcmp(get(handles.startStopCamera,'String'),'Start Camera')
    set(handles.startStopCamera,'String','Stop Camera')
    axes(handles.VideoCam);
    start(handles.video);
else
    set(handles.startStopCamera,'String','Start Camera')
    axes(handles.VideoCam);
    stop(handles.video);
end


% --- Executes when user attempts to close MainGUI.
function MainGUI_CloseRequestFcn(hObject, eventdata, handles)
if strcmp(get(handles.startStopCamera,'String'),'Stop Camera')
stop(handles.video);
end
delete(hObject);


% --- Executes on button press in Face.
function Face_Callback(hObject, eventdata, handles)

if strcmp(get(handles.startStopCamera,'String'),'Stop Camera')

    imgface=ycbcr2rgb(getsnapshot(handles.video));
    imgface=imcrop(imgface,[70,20,179,199]);
    stop(handles.video);
    axes(handles.faceAxes);
    imshow(imgface);
    if(size(imgface)==3)
        imgface=rgb2gray(imgface);
    end
    %[ EV IW m M] = CreateEigenVector( 'trFcdb','jpg' );
    %[ id im ] = EigenMatch( imgface,EV,IW,m,M);
   % [id im]=getImgMatch(imgface,'trFcdb');
    T = CreateDatabase('trFcdb');
[m V_PCA V_Fisher ProjectedImages_Fisher] = FisherfaceCore(T);
OutputName = Recognition(imgface, m, V_PCA, V_Fisher,
ProjectedImages_Fisher);

SelectedImage = strcat('trFcdb\',OutputName);
display(SelectedImage);
    set(handles.fc,'String',SelectedImage);
    axes(handles.VideoCam);
    start(handles.video)

else
    set(handles.Msg,'String','Please Turn ON the camera');
end




% --- Executes on button press in Match.
function Match_Callback(hObject, eventdata, handles)
face=(get(handles.fc,'String'));
fcdb='fc_database.dat';
if (exist(fcdb,'file')==2)
    load(fcdb,'-mat');
end
for i=1:fc_no
    if strcmp(face,fname)
        set(handles.Msg,'String',pname);
    end
end
if strcmp(get(handles.startStopCamera,'String'),'Stop Camera')

    stop(handles.video);
    axes(handles.RegFaceAxes);
    imgface=imread(face);
    imshow(imgface);
    axes(handles.VideoCam);
    start(handles.video)

else
    axes(handles.RegFaceAxes);
    imgface=imread(face);
    imshow(imgface);

    axes(handles.VideoCam)
end




% --- Executes during object creation, after setting all properties.
function fc_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in browse.
function browse_Callback(hObject, eventdata, handles)
if strcmp(get(handles.startStopCamera,'String'),'Start Camera')

[namefile,pathname]=uigetfile({'*.jpg;*.jpeg','IMAGE Files (*.jpg,*.jpeg)'});
img=imread(strcat(pathname,namefile));
[r c z]=size(img);
if r>200 && c>180
    img=imcrop(img,[70,20,179,199]);
end
axes(handles.faceAxes);
        handles.face1=img;
        guidata(hObject, handles);
        imshow(handles.face1);
       T = CreateDatabase('trFcdb');
[m V_PCA V_Fisher ProjectedImages_Fisher] = FisherfaceCore(T);
OutputName = Recognition(img, m, V_PCA, V_Fisher,
ProjectedImages_Fisher);
```

```
SelectedImage = strcat('trFcdb\',OutputName);
display(SelectedImage);
    set(handles.fc,'String',SelectedImage);
    %[ EV IW m M] = CreateEigenVector( 'trFcdb','jpg' );
    %[ id im ] = EigenMatch( img,EV,IW,m,M );

axes(handles.VideoCam);

else
stop(handles.video);
[namefile,pathname]=uigetfile({'*.jpg;*.jpeg','IMAGE Files (*.jpg,*.jpeg)'});
img=imread(strcat(pathname,namefile));
[r c z]=size(img);
if r>200 && c>180
    img=imcrop(img,[70,20,179,199]);
end


        axes(handles.faceAxes);
        handles.face1=img;
        guidata(hObject, handles);
        imshow(handles.face1);
        T = CreateDatabase('trFcdb');
[m V_PCA V_Fisher ProjectedImages_Fisher] = FisherfaceCore(T);
OutputName = Recognition(img, m, V_PCA, V_Fisher,
ProjectedImages_Fisher);

SelectedImage = strcat('trFcdb\',OutputName);
display(SelectedImage);
    set(handles.fc,'String',SelectedImage);
   % [ EV IW m M ] = CreateEigenVector( 'trFcdb','jpg' );
   % [ id im ] = EigenMatch( img,EV,IW,m,M );
    %display(id);


axes(handles.VideoCam);
start(handles.video);

end
```

```
%guidata(hObject, handles);
```

F. Recognition :

```
 InputImage = TestImage;
    InputImage(:,:,1)=histeq(InputImage(:,:,1));
    InputImage(:,:,2)=histeq(InputImage(:,:,2));
    InputImage(:,:,3)=histeq(InputImage(:,:,3));
temp = InputImage(:,:,1);

[irow icol] = size(temp);
InImage = reshape(temp',irow*icol,1);
Difference = double(InImage)-m_database; % Centered test image
ProjectedTestImage = V_Fisher' * V_PCA' * Difference; % Test image
feature vector

%%%%%%%%%%%%%%%%%%%%%%%% Calculating Euclidean
distances
% Euclidean distances between the projected test image and the
projection
% of all centered training images are calculated. Test image is
% supposed to have minimum distance with its corresponding image in
the
% training database.

Euc_dist = [];
for i = 1 : Train_Number
    q = ProjectedImages_Fisher(:,i);
    temp = ( norm( ProjectedTestImage - q ) )^2;
    Euc_dist = [Euc_dist temp];
end
display(Euc_dist);

[Euc_dist_min , Recognized_index] = min(Euc_dist);
display(Euc_dist_min);
if Euc_dist_min>(0.65)*1.0e+016
    display('No match found')
else
    display('Match found');
end
OutputName = strcat(int2str(Recognized_index),'.jpg');
```

Future Work:

A) In our current implementation, we have used only a single gas detector, by using different types of detectors the device can be used in different applications as well as different industries.

B) This work can also be extended to provide security in safety vaults in banks by using motion sensors and thus preventing robberies.

Conclusion:

The device can be generalised as a multi-purpose security system. Some of the real world applications in which this device can be applied are as Home security system, Fire alarm system, Smoke detector,Surveillance systems etc.

References:-

1. Laurence R. Rabiner, Bernard Gold. Theory and Applications of Digital Signal Processing. London: Prentice-Hall International, Inc, 1975.
2. Security Guard Robots using Map Information. IEEEXplore. Mar 10, 2009. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00637950
3. Florida International University's students project, 'Parallel Parking Vehicle'. http://web.eng.fiu.edu/~fanj/course_materials/senior_spring08/proposal_parallel.pdf
4. Patent Storm. 'Home cleaning robot'. http://www.patentstorm.us/patents/6459955.html
5. Rickey's World. http://www.8051projects.net