최종 보고서

Ultrasonic Ruler

강　　의 : 전자HW설계

강 의 자 : 나종화 교수님

참여인원 : 2012122043 김신재

2012122059 김종윤

제출기한 : 2016.12.8

● 프로젝트 목표

초음파 센서와 UART DE1-SOC보드를 종합적으로 이용하여 정확한 거리를 측정하는 초음파 Ruler 제작한다.

● System 설명

하드웨어 구성은 Sonar sensor, DE1-SOC , UART 이렇게 3가지,
소프트웨어 구성은 DE1-SOC를 컨트롤 할 quartus의 코드와 받은 값을 분석 및 설정하는 Matlab 코드로 구성되어 있다.

첫 번째, Quartus에서 DE1_SoC보드로 명령을 보내 Sonar Sensor를 작동시킨다.

두 번째, 소나 센서가 일정한 시간주기마다 물체와의 거리를 측정한다.

세 번째, Matlab에서 UART 통신(TX)을 통하여 Command 명령을 DE1-SOC로 보내면 보드에서 센서로부터 받은 값을 다시 UART통신(RX)을 통해 Matlab으로 값을 전송한다.

세 번째, 정해진 Sampling 갯수만큼 전송받은 길이 Data를 array에 저장하여. 그 후 잡음 제거 및 정확한 값 출력을 위해 Matlab 안에서 구현한 Filter를 거쳐 정확한 값을 출력한다.

네 번째, 약간의 오차를 수렴시키기 위해 칼만필터를 사용한다.

마지막으로, 칼만필터를 통해 정해진 값을 MATLAB GUI로 보내어 Display한다.

● 소자 설명

1)Sonar Sensor

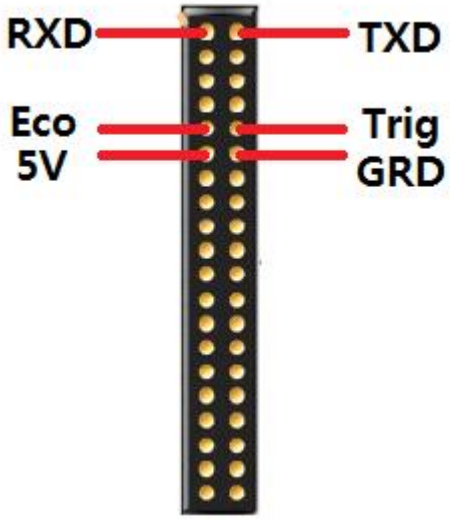
센서에는 4개의 핀이 있는데 Vcc, Trig, Echo, GND핀이 있다.

Vcc는 +5V로 연결해 주어야 하며 DE1-SOC의 GPIO핀에서 5V를 제공해주는 핀에 연결하면 된다.

GND핀 또한 GPIO핀과 맞추어 연결을 한다.

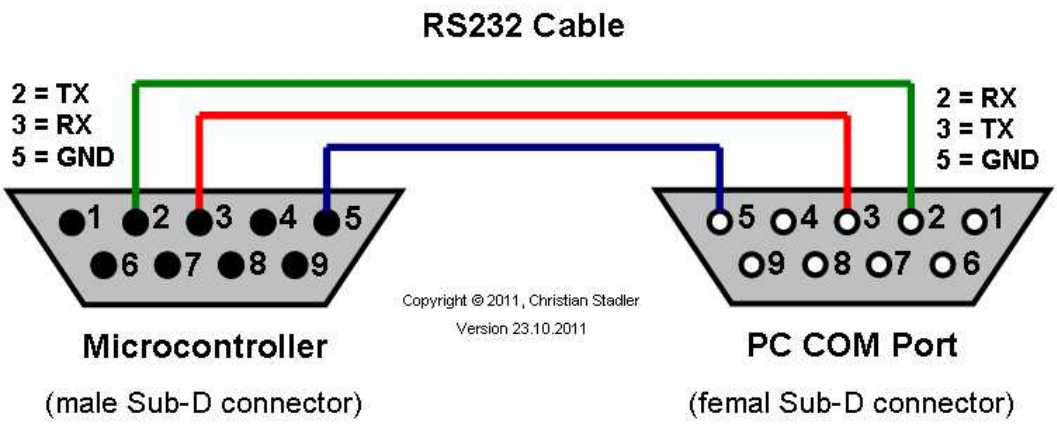
Trig핀은 센서에 초음파 출력 명령을 주는 역할을 한다.

Echo핀은 ping에서 발사된 초음파가 반사되고 trig로 받아 그 반사된 초음파를 보내는 부분이다.



2)DE1-SOC UART

초음파 센서에서 측정한 거리를 UART 통신을 통하여 Computer로 전송을 하게된다.



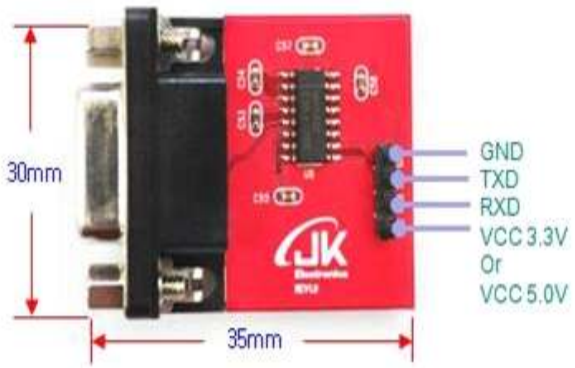
여러 핀 중,

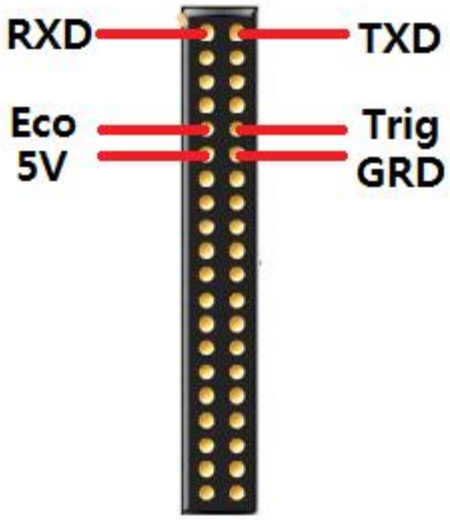
RX는 Receive, Device가 UART통신을 통하여 Decive로부터 값을 받을 때 사용된다.

TX는 Transmit, Device가 UART통신을 통하여 Decive로 값을 전송할 때 사용된다.

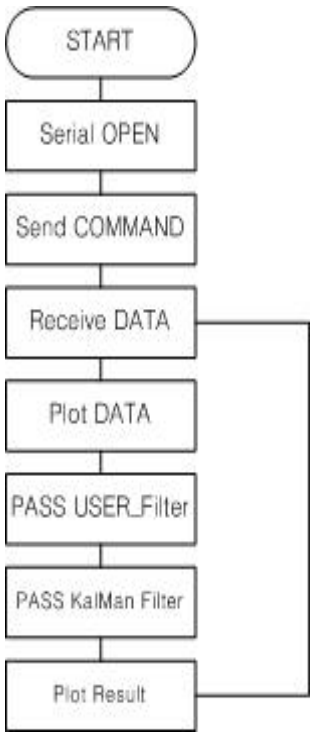
9핀의 케이블중 2,3,5번 포트만 사용한다. 이외의 핀은 사용하지 않는다.

RXD, TXD, GND는 보드의 GPIO에 맞추어 꽂아 사용한다.

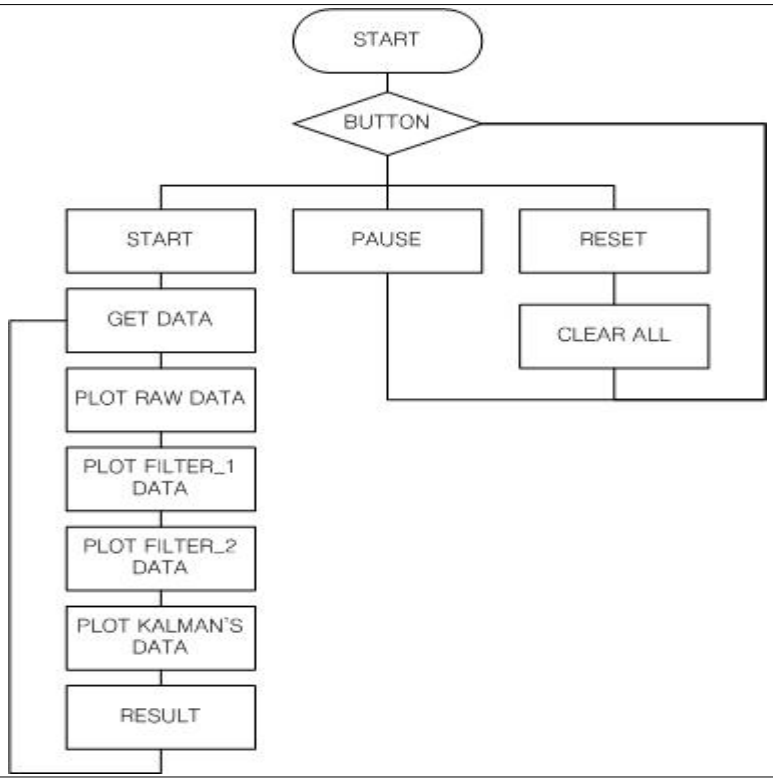




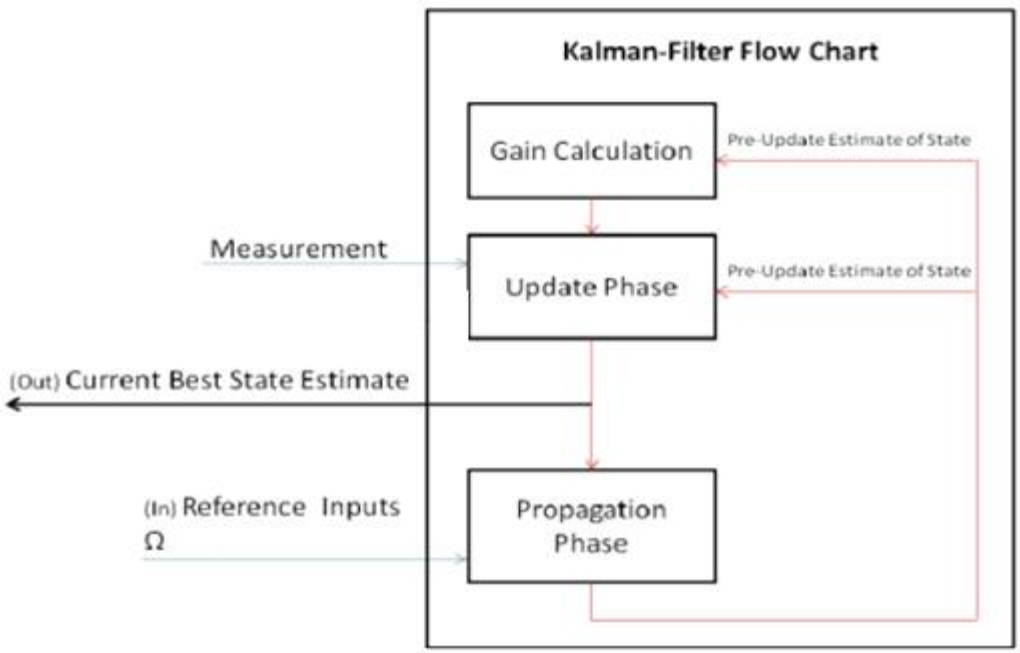
● FLOWCHART

Quartus	Matlab
	

MATLAB GUI

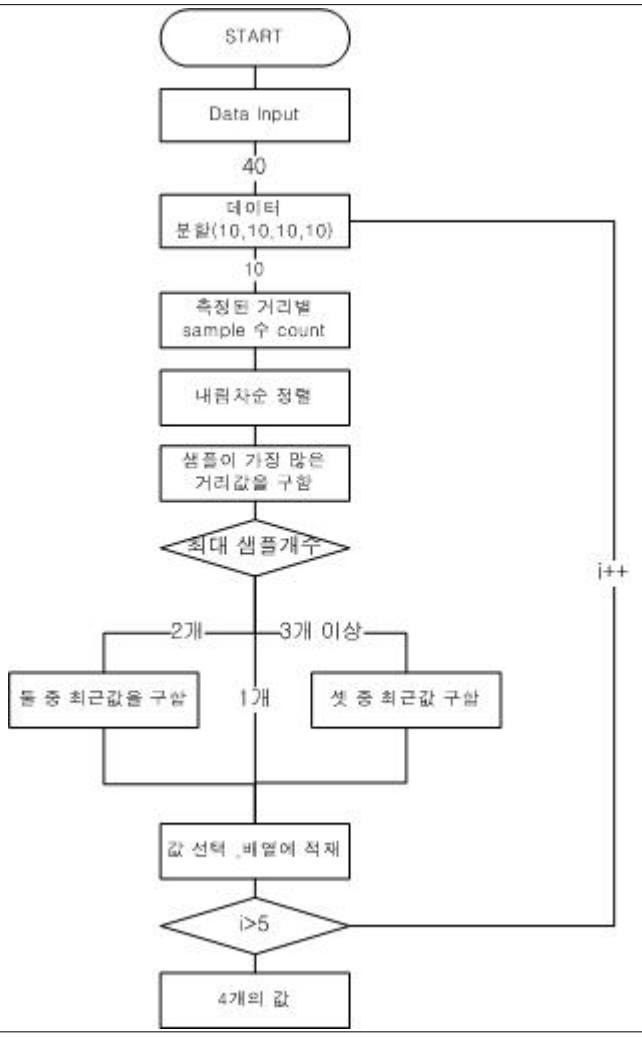


KalMan's Filter

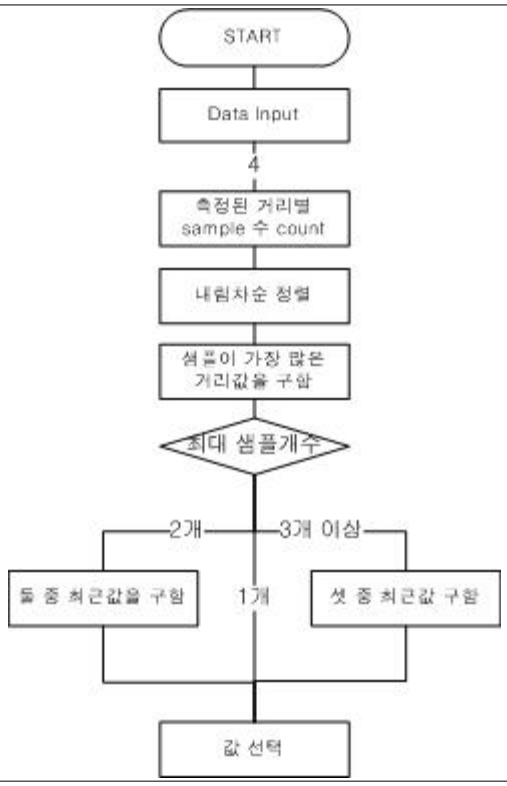


참고 : https://www.researchgate.net/figure/228339108_fig1_Figure-6-Kalman-Filter-flowchart-for-both-camera-and-odometry

User_filter_1



User_filter_1



● quartus 코드 설계방향

UART의 RX TX 송수신은 char의 자료형을 사용하여 통신하기 때문에 받은 값의 데이터를 수정하기 위해서는 int자료형으로 변형해야 한다.
이러한 변환은 각각의 자릿수 값 마다 계산을 할 필요가 있다.
자료형의 변경을 위해 문자로 받은 숫자에 48을 더하여 int자료형 계산을 수행한다. int자료형으로 변경 후 다시 matlab으로 TX통신을 해야 하기 때문에 char자료형으로 바꾸어 준다. 아스키코드 13인 Carriage Return을 넣음으로 값을 보낸다.

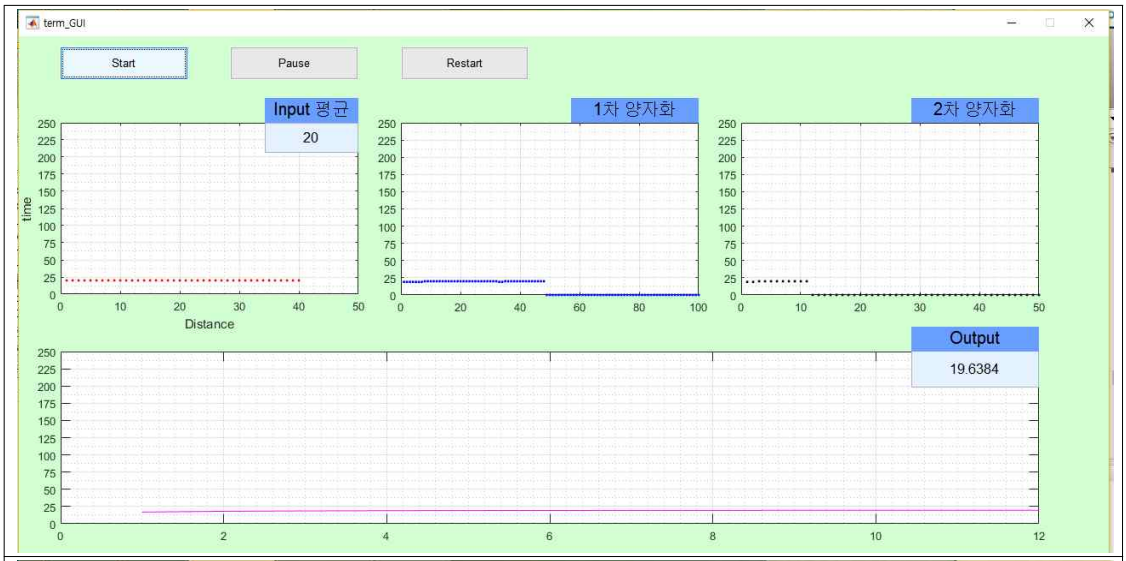
● 필터 설계 방향

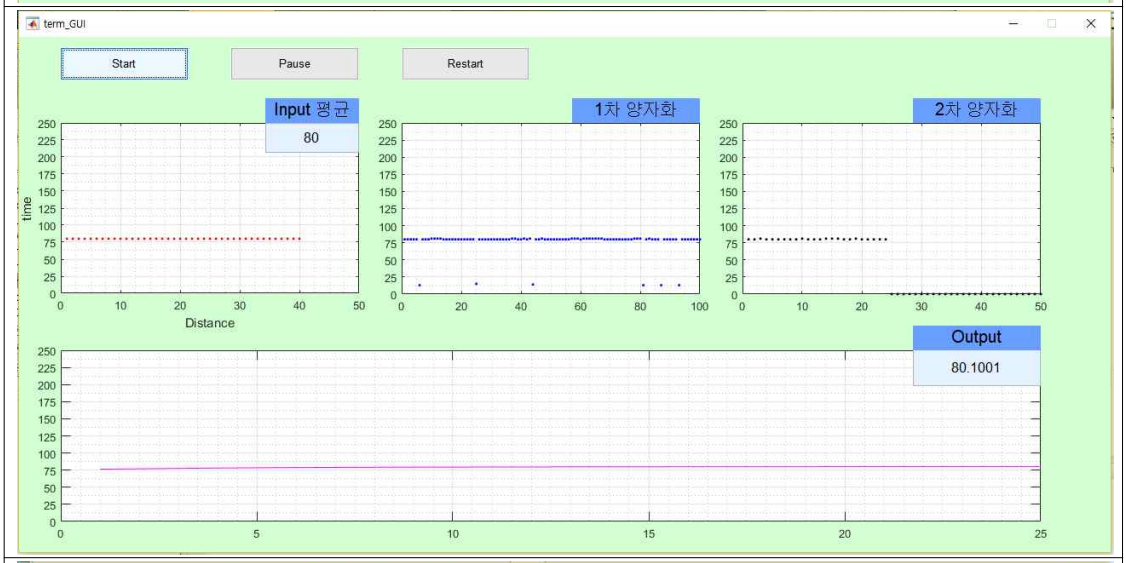
센서로부터 받은 최초 input값들의 오류 즉, 튀는 값들을 제거하기 위해 비선형 양자화 개념을 적용하였다. 비선형 양자화란 sample의 분포가 많이 되어있는 곳을 한 개의 점으로 mapping하는 방법인데, 최초 input값을 40개의 sample로 받아서 10개씩 나눈 다음 분포가 많이 이루어진 점으로 10개를 1개로 mapping 하였다. 그렇게 해서 나온 4개의 sample들을 다시 1개로 mapping하여 1차 양자화를 거쳐도 걸러지지 않는 오류 값들을 다시 걸러냈다. 양자화를 두 번 걸쳐서 걸러낸 거리 값들을 칼만 필터에 통과 시켜 정확한 거리에 근사하도록 설계하였다.

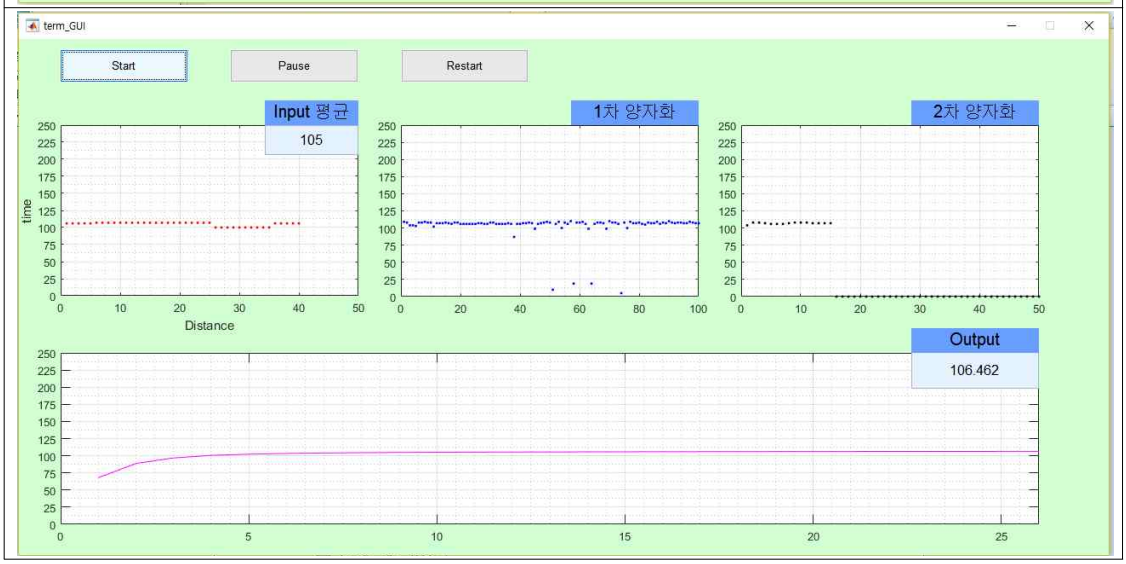
● 결론 및 고찰

초음파센서의 특성상 주변의 잡음과 온도에 매우 민감하다.
이러한 상황을 고려하여 각종 변수의 값을 정하여 거리를 측정하였으나 어느 정도의 오차를 피할 수는 없었다. 특히 측정 도중 잡음의 정도가 심하여 튀는 값이 자주 발생하였다.
이 값을 제외하고 측정하기 위해 각종 필터를 이용하였고 측정값의 정확도를 높이고자 칼만필터를 이용하게 되었다.
초음파 센서로 이용한 hr04소자는 측정거리가 최대 4m이지만 실험 중 2m까지는 허용오차의 수준으로 근사한 값이 출력되나, 그 이상의 거리를 측정 할 때에는 측정이 잘 되지 않거나 허용오차를 크게 넘어서서 길이 측정이 힘들었다. 물체의 표면이 울퉁불퉁하여 반사각이 커지게 되면 정확한 값을 측정하기 어려웠으며 일정거리 이상에서 작은 물체를 측정하게 될 때에도 값이 부정확했다.
한번에 40개의 값을 받아서 필터를 출력할 때 값의 범위에 따라 40개를 정렬하여 일정 구간으로 나눈 후 측정이 많이 된 값을 기준삼아 1차 필터 2차필터를 통과했다.

● GUI 결과 화면


term_GUI
Start
Pause
Restart
Input 평균
20
1차 양자화
2차 양자화
time
Distance
Output
19.6384


term_GUI
Start
Pause
Restart
Input 평균
80
1차 양자화
2차 양자화
time
Distance
Output
80.1001


term_GUI
Start
Pause
Restart
Input 평균
105
1차 양자화
2차 양자화
time
Distance
Output
106.462

●부록 code

1) quartus

```
#include <stdio.h>
#include <stdlib.h>
#include "system.h"
#include "altera_avalon_uart_regs.h"
#include "time.h"
#include <math.h>

volatile int * GPIO_UART_ptr = (int *) UART_0_BASE; // UART address
void delay_t(int count);
int getTxLength(char *string);
double Calculate(int dis_cnt);
void make_put_line(char* c, int index);
char Start_line[161];

int main()
{
    volatile int *Ultra = ((volatile int *)ULTRA_SONIC_0_BASE);
    int dis_cnt =0;
    int i;
    double distance =0;
    char DATA;
    while(1)
      {
        DATA = IORD_ALTERA_AVALON_UART_RXDATA(GPIO_UART_ptr);
        if(DATA == '<')
        {
          for(i = 0; i<160; i++)
          {
            dis_cnt =*Ultra;
            distance = Calculate (dis_cnt);

            Start_line[i++] = (char) ((((int)distance)/100)%10)+48;
            Start_line[i++] = (char) ((((int)distance)/10)%10)+48;
            Start_line[i++] = (char) (((int)distance)%10)+48;
            Start_line[i] = ' ';
            delay_t(10);
          }
          Start_line[i] = 13;
          make_put_line(Start_line, 161);
```

```
          DATA = 0;
        }
      }
}
double Calculate(int dis_cnt)
{
 double distance = dis_cnt*0.000001*331.6/2;
 return distance;

}
void delay_t(int count)
{
volatile int * Timer_ptr= (int *) INTERVAL_TIMER_BASE;
 unsigned int counter= 100000*count;
 *(Timer_ptr + 0x2) = (counter&0xFFFF);
 *(Timer_ptr + 0x3) = ((counter>>16)&0xFFFF);
 *(Timer_ptr + 0x1)= 0x5;
 int Status = *(Timer_ptr);

 while(Status==2)
 {
  Status = *Timer_ptr;
 }
 *(Timer_ptr + 0x1)=0x0;
}

int getTxLength(char * string)
{
 int index = 0;

 while((string[index])!=0) index++;
 return index;
}

void make_put_line(char* c, int index)
{
 int Count=0;
 char* text_string = c;
 while(Count < index)
 {
  IOWR_ALTERA_AVALON_UART_TXDATA(GPIO_UART_ptr, text_string[Count]);
  delay_t(10);
  Count++;
 }
}
```

2) matlab

```
function varargout = term_GUI(varargin)
% TERM_GUI MATLAB code for term_GUI.fig
%      TERM_GUI, by itself, creates a new TERM_GUI or raises the existing
%      singleton*.
%
%      H = TERM_GUI returns the handle to a new TERM_GUI or the handle to
%      the existing singleton*.
%
%      TERM_GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TERM_GUI.M with the given Output
arguments.
%
%      TERM_GUI('Property','Value',...) creates a new TERM_GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before term_GUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      pause.  All inputs are passed to term_GUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help term_GUI

% Last Modified by GUIDE v2.5 07-Dec-2016 00:07:14

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @term_GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @term_GUI_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 term_GUI is made visible.
function term_GUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to term_GUI (see VARARGIN)

% Choose default command line output for term_GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes term_GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = term_GUI_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
varargout{1} = handles.output;


% --- Executes on button press in Start.
function Start_Callback(hObject, eventdata, handles)
% hObject    handle to Start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc
delete(instrfindall);
COMPORT = 'COM4'
s = serial(COMPORT, 'BaudRate', 115200,'Timeout',10, 'Terminator', 'CR');

h = 0;
h2 = 0;
h3 = 0;
Z2=zeros(1,10000);
Z3=zeros(1,10000);
Z4=zeros(1,10000);
distance=0:1:250; %측정거리 범위
sample=40; %측정샘플 개수

while(1)
    fopen(s);
    fprintf(s,'<CAO>');

    a = fgets(s);
    A = str2num(a);

    Am=mean(A);
    set(handles.input,'String',Am);

    Ah=A;
    for i=1:sample
        if Ah(i)>250 || Ah(i)==0;
            Ah(i)=0;
        end
    end
    axes(handles.axes1);
    cla;
    plot(A,'.r','LineWidth',2);
    axis([0,40,0,250]);
    set(gca,'YTick',[0:25:250]);
    xlabel('Distance')
    ylabel('time')
    grid on
    grid minor
    drawnow

    qsample=sample/4;
    Ah1=zeros(1,qsample);
    Ah2=zeros(1,qsample);
    Ah3=zeros(1,qsample);
    Ah4=zeros(1,qsample);
    for i=1:10
        Ah1(i)=Ah(i);
    end
    for i=11:20
        Ah2(i-10)=Ah(i);
    end
    for i=21:30
        Ah3(i-20)=Ah(i);
    end
    for i=31:40
        Ah4(i-30)=Ah(i);
    end
```

```
    fQ1=newQuantization(Ah1,length(distance),qsample); lfQ1=length(fQ1);
    fQ2=newQuantization(Ah2,length(distance),qsample); lfQ2=length(fQ2);
    fQ3=newQuantization(Ah3,length(distance),qsample); lfQ3=length(fQ3);
    fQ4=newQuantization(Ah4,length(distance),qsample); lfQ4=length(fQ4);
    fQ=zeros(1,lfQ1+lfQ2+lfQ3+lfQ4);
    for i=1:lfQ1
        fQ(i)=fQ1(i);
    end
    for i=lfQ1+1:(lfQ1+lfQ2)
        fQ(i)=fQ2(i-lfQ1);
    end
    for i=(lfQ1+lfQ2+1):(lfQ1+lfQ2+lfQ3)
        fQ(i)=fQ3(i-(lfQ1+lfQ2));
    end
    for i=(lfQ1+lfQ2+lfQ3+1):(lfQ1+lfQ2+lfQ3+lfQ4)
        fQ(i)=fQ4(i-(lfQ1+lfQ2+lfQ3));
    end
    for i=1:length(fQ)
    Z2(i+h2)=fQ(i); %1차 양자화 결과
    end
    h2=h2+length(fQ);

    axes(handles.axes2);
    plot(Z2,'.b','LineWidth',2);
    axis([0,200,0,250]);
    set(gca,'YTick',[0:25:250]);
    grid on
    grid minor
    drawnow;

    q=zeros(1,h2);
    lfQ=sum(sign(Z2));
    if lfQ == h2
        for i=1:(h2)
          q(i)=Z2(i+(h2-length(fQ)));
        end
    end

    sQ=newQuantization(q,length(distance),h2);
    for i=1:length(sQ)
    Z3(i+h3)=sQ(i);
    end
    h3=h3+length(sQ);

    axes(handles.axes3);
    plot(Z3,'.k','LineWidth',2);
    axis([0,200,0,250]);
    set(gca,'YTick',[0:25:250]);
    grid on
    grid minor
    drawnow

    count=1;
    for i=2:length(Z3)
        if Z3(i)>0
            count=count+1;
            if abs(Z3(i)-Z3(i-1))>50
                Z3(i)=Z3(i-1);
            end
        end
    end
for i=1:count
        kal = SimpleKalman(Z3(i));
end
    for i=1:length(kal)
        Z4(i+h)=kal(i);
    end
     h=h+length(kal);

    set(handles.Output,'String',kal);
    axes(handles.axes5);
```

```
    plot(Z4,'.m','LineWidth',2);
    axis([0,h+1,0,250]);
    set(gca,'YTick',[0:25:250]);
    grid on
    grid minor
    drawnow
    fclose(s);
end



% --- Executes on button press in Pause.
function Pause_Callback(hObject, eventdata, handles)
% hObject    handle to Pause (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
pause;



function Output_Callback(hObject, eventdata, handles)
% hObject    handle to Output (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 Output as text
%        str2double(get(hObject,'String')) returns contents of Output as a double



% --- Executes during object creation, after setting all properties.
function Output_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Output (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 restart.
function restart_Callback(hObject, eventdata, handles)
% hObject    handle to restart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

close(gcbf)
clear all
 term_GUI


function input_Callback(hObject, eventdata, handles)
% hObject    handle to input (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 input as text
%        str2double(get(hObject,'String')) returns contents of input as a double


% --- Executes during object creation, after setting all properties.
function input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to input (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 output = newQuantization(input,distance,sample)

samp_dist=zeros(1,distance); %거리값당 샘플수 가산
for i=1:distance
    for ii=1:sample
        if i==input(ii)
            samp_dist(i)=samp_dist(i)+1;
        end
    end
end

sort_samp_dist=sort(samp_dist,'descend'); %가장많은 샘플수를 지닌 거리값 순으로
정렬
if sort_samp_dist(1)~=samp_dist(1) %가장많은 샘플의 거리값이 0이 아닌경우
    max_samp_num=sort_samp_dist(1);
else  %가장많은 샘플의 거리값이 0인경우
    max_samp_num=sort_samp_dist(2);
end

md=zeros(1,sample);
x=1;
for i=1:distance
    if samp_dist(i)==max_samp_num %가장많은 샘플을 갖고 있는 거리값을 찾자
       md(x)=i; %찾으면 md에 저장
       x=x+1;
    end
end

Q=zeros(1,sample);
neomd=zeros(1,sample);
    if sum(sign(md))==1 %% level이 1인 경우
        Q(sample)=md(1); %걍 양자화
    elseif sum(sign(md))==2 %% level이 2인 경우
        if md(1)==input(sample) %% 둘중에 최신값부터 찾자
                neomd(1)=md(1);
                neomd(2)=md(2);
        elseif md(2)==input(sample)
                neomd(1)=md(2);
                neomd(2)=md(1);
        end
        Q(sample)=neomd(1); %둘중에 최신값으로 양자화
    elseif sum(sign(md))>=3  %% level이 3,이상인 경우
        if md(1)==input(sample)%% 셋중에 최신값부터 찾자
            neomd(1)=md(1);
        elseif md(2)==input(sample)
            neomd(1)=md(2);
        elseif md(3)==input(sample)
            neomd(1)=md(3);
        end
        Q(sample)=neomd(1);
    end
    o=0;
    for i=1:length(Q)
        if Q(i)~=0
            o=o+1;
        end
    end
    out=zeros(1,o);
    oo=1;
```

```
      for i=1:length(Q)
          if Q(i)~=0
               out(oo)=Q(i);
               oo=oo+1;
          end
       end
     for i=1:length(out)
     if abs(out(i)-out(1))>3
          out(i)=out(1);
     end
     end
     output=out;
end
```

```
function volt = SimpleKalman(z)
%
%
persistent A H Q R
persistent x P
persistent firstRun

if isempty(firstRun)
  A = 1;
  H = 1;

  Q = 0;
  R = 4;

  x = 14;
  P =  6;

  firstRun = 1;
end

xp = A*x;
Pp = A*P*A' + Q;

K = Pp*H'*inv(H*Pp*H' + R);

x = xp + K*(z - H*xp);
P = Pp - K*H*Pp;

volt = x;
```