Function Library

Creates a structure of function handles. To use, call the function at the beginning of your script and assign to a variable. From there call a function within the structure as you would any structure field

Ex:

```
f = Func_lib;
a = f.V_Comp(V_1,V_2,V_s);
```

Main Function

```
function f = Func_Lib
    f.connectAlicat = @connectAlicat;
    f.setFlow = @setFlow;
    f.calcFlow = @calcFlow;
    f.flushAlicatBuffer= @flushAlicatBuffer;
    f.ConnectArduino = @ConnectArduino;
    f.InitializeSensors = @InitializeSensors;
    f.thermocouple_read = @thermocouple_read;
    f.radiometer_read = @radiometer_read;
    f.pressure_read = @pressure_read;
    f.scale_read = @scale_read;
    f.thermistor_read = @thermistor_read;
end
```

Alicat Setup and Functions

a

```
function aliComm=connectAlicat
    % * Purpose
    % Form a connection to the Alicat controller. By default
    % connects to COM port hard-coded into file. User may overide
    % this.

    COM='COM1';

    fprintf('Connecting to Alicats on port %s\n',COM)

    aliComm=serial(COM,...
        'TimeOut', 2,...
        'BaudRate', 19200,...
        'Terminator','CR');
    %delete(instrfind)
    fopen(aliComm);
    %fclose(aliComm)
    %delete(aliComm)
    %clear aliComm
end
```

a

```
function setFlow(flowRate,odourConc)
```

```
% function setFlow(flowRate,odourConc)
%
% Set the flow rate of the combined odour stream and the odour
% concentration. Or change flow of just one controller:
%
% flowRate - desired flow rate produced by both controllers (unless
%            odourConc is a controller ID). UNITS: SLPM.
% odourConc- i. maintains the same overall flow rate but changes the odour
%            concentration. UNITS: 0--100% of 1 SLPM
%            ii. If odourConc is a controller ID (e.g. 'A' or 'B') then the
%            function sets the named controller to the flow  pecified by
%            flowRate. Can be a char array of several unit IDs in which
%            case all are set to flowRate.
% examples -
% setFlow(0.5,'A') %Set controler A to half full scale
% setFlow(2100/5000,'C') %Set 5 SLPM controler, C, to 2.1 SLPM
%
% Rob Campbell - 21st March 2008 - CSHL

% flowRate = flowRate/20;

global aliComm;
if isempty(aliComm), aliComm=connectAlicat; end

if ischar(odourConc)

    flowRate=round(flowRate*1000)/1000;
    flowRate=str2double(sprintf('%0.3g',flowRate));
    for i=1:length(odourConc)
        unitID=odourConc(i);
        %flow=calcFlow(flowRate);
        flow=calcFlow(flowRate,odourConc);
        flow = round(flow);
        fprintf(aliComm, sprintf('%s%g',unitID,flow) );
        A = readMFC(aliComm);
        sprintf('%s%g',unitID,flow);
    end
flushAlicatBuffer;

elseif isnumeric(odourConc)

    % We will pass 1 SLPM total and allow the odour to from 0 to 100
    % percent of this range.
    flowA = flowRate*(1-odourConc); % MFC A: clean carrier air stream.
    flowB = flowRate*odourConc;     % MFC B: odourised air stream.

    if flowA>1 || flowA<0, error('FlowA out of bounds'), end
    if flowB>1 || flowB<0, error('FlowB out of bounds'), end

   %Convert to correct units for MFC
    flowA=calcFlow(flowA);  flowB=calcFlow(flowB);

   fprintf(aliComm, sprintf('%s%0.0f','A',flowA) );
   fprintf(aliComm, sprintf('%s%0.0f','B',flowB) );
```

```
    end
end
```

a

```
function flow=calcFlow(setPoint, UnitID)
    % function flow=calcFlow(setPoint)
    %
    % Convert a desired flow rate into the correct units for the MFC. See p. 22
    % of operating manual.
    % (desired set point * 64000)/Full scale range

    %format shortG %Formats the notation such that there's no exp notation

    %flow = (setPoint * 64000)/5; %64000/5 for MFC E

    if UnitID == 'A'|| UnitID == 'B'
        flow = (setPoint * 64000)/50; %ISA50 and ISB50 %64000/50 for MFC A and B
    elseif UnitID == 'C'||UnitID == 'D'
        flow = (setPoint * 64000)/1; %ISA1 and ISAB1 %64000/1 for MFC C and D
    else %if UnitID is E
        flow = (setPoint * 64000)/5; %64000/5 for MFC E
    end
end

function flushAlicatBuffer

    global aliComm;
    if isempty(aliComm), aliComm=connectAlicat; end

    %fprintf('Flushing Alicat serial buffer')
    while aliComm.BytesAvailable>0
        fread(aliComm,aliComm.BytesAvailable);
        %fprintf('.')
        %pause(0.05)
    end
    fprintf('\n')
end
```

Arduino Setup & Functions

Connect to Arduino

Purpose: Form a connection to the Arduino controller. If no input argument, connects to an Uno board on COM port 6.

```
function ar = ConnectArduino(Com,Board)
    COM='COM6';
    board = 'Uno';

    switch nargin
        case 1
            COM = Com;
        case 2
```

```
            board = Board;
    end

    ar = arduino(COM, board, 'Libraries', 'I2C');
end
```

Connect Sensors & Transducers

May need to accept arduino as input from app class

```
function [Sensors,Transducers] = Initialize(ar)
    %if isempty(ar), ar = ConnectArduino; end

    addrs = scanI2CBus(ar);

    for i = 1:length(addrs)
        switch addrs{i}
            case {'0x48','0x49'}
                Sensors.Radiometer = Radiometer(ar,addrs{i});
            case {'0x4A','0x4A'}
                Sensors.Thermistor_1 = Thermistor(ar);
                Sensors.Thermistor_2 = Thermistor(ar,1);
            case '0x67'
                Sensors.Thermocouple = mcp9600(ar, addrs{i});
            case {'0x18','0x19'}
                Transducers.Relay = Relay(ar,addrs{i});
            %case -pressure sensor address-
                %Pressure sensor class
            %case -Scale address-
                %Scale Class
        end
    end
    %Sensors.Camera =
    Transducers.ArmMotor = DCMotor(ar);

end
```

```
%{
function Report = SensorCheck()
    global Sensors
    if isempty(Sensors), Sensors = InitializeSensors; end

    needed = {'',''};

    if ~isfield(Sensors,needed)
        Report = [fieldnames(Sensors,)];
    end

end
%}
```

Sensor/Transducer Functions

```
function T = thermocouple_read(Sensors)
    T = Sensors.Thermocouple.readHotJunc()
end
```

```
function F = radiometer_read(Sensors)
    v = readVoltage(Sensors.ADC1.Device,Sensors.ADC1.RadPin)

    T_d = thermistor_read(Sensors.ADC2.Device,Sensors.ADC2.Therm1Pin);
    T_c = thermistor_read(Sensors.ADC2.Device,Sensors.ADC2.Therm2Pin);

    %F =
end
```

```
function T = thermistor_read(Dev,pin)
    v = readVoltage(Dev,pin)

    %T =
end
```

```
function [P,T] = pressure_read(Sensors)

end
```

```
function W = scale_read(Sensors)

end
```