Anexo 1: Programa del comando drivebot

```
function drivebot(a,b)
    bgcol = [135 206 250]/255;

    if isstr(a)
        % drivebot(name, j), graphical callback function
        name = a; j = b;
        rh = findobj('Tag', name);
        handles = get(gco, 'Userdata');
        scale = handles{3};
        for r=rh',
            rr = get(r, 'UserData');
            q = rr.q;
            if isempty(q),
                q = zeros(1,rr.n);
            end
            if gco == handles{1},
                % get value from slider
                q(j) = get(gco, 'Value') / scale(j);
                set(handles{2}, 'String', num2str(scale(j)*q(j)));
            else
                % get value from text box
                q(j) = str2num(get(gco, 'String')) / scale(j);
                set(handles{1}, 'Value', q(j));
            end
            rr.q = q;
            set(r, 'UserData', rr);
        end
        plot(rr, q)
        t6 = fkine(rr, q);
        h3 = get(findobj('Tag', 'T6'), 'UserData');
        for i=1:3,
            set(h3(i,1), 'String', sprintf('%.3f', t6(i,4)));
            set(h3(i,2), 'String', sprintf('%.3f', t6(i,3)));
        end
    else
        % drivebot(r, q)
        r = a;
        scale = ones(r.n,1);

        n = r.n;
        width = 300;
        height = 40;
        minVal = -pi;
        maxVal = pi;

        qlim = r.qlim;
        if isempty(qlim),
            qlim = [minVal*ones(r.n,1) maxVal*ones(r.n,1)];
        end

        if nargin < 2,
            q = zeros(1,n);
        else
            if isstr(b),
```

```
                if strncmp(b, 'deg', 3),
                    disp('** in degree mode')
                    L = r.link;
                    for i=1:r.n,
                        if L{i}.sigma == 0,
                            scale(i) = 180/pi;
                        end
                    end
                end
            else
                q = b;
            end
        end
        t6 = fkine(r, q);
        fig = figure('Units', 'pixels', ...
            'Position', [0 -height width height*(n+2)], ...
            'Color', bgcol);
        set(fig,'MenuBar','none')
        delete( get(fig, 'Children') )

        % first we check to see if there are any graphical robots of
        % this name, if so we use them, otherwise create a robot plot.

        rh = findobj('Tag', r.name);

        % attempt to get current joint config of graphical robot
        if ~isempty(rh),
            rr = get(rh(1), 'UserData');
            if ~isempty(rr.q),
                q = rr.q;
            end
        end


        % now make the sliders
        for i=1:n,
            uicontrol(fig, 'Style', 'text', ...
                'Units', 'pixels', ...
                'BackgroundColor', bgcol, ...
                'Position', [0 height*(n-i) width*0.1 height*0.4], ...
                'String', sprintf('q%d', i));

            h(i) = uicontrol(fig, 'Style', 'slider', ...
                'Units', 'pixels', ...
                'Position', [width*0.1 height*(n-i) width*0.7
height*0.4], ...
                'Min', scale(i)*qlim(i,1), ...
                'Max', scale(i)*qlim(i,2), ...
                'Value', scale(i)*q(i), ...
                'Tag', sprintf('Slider%d', i), ...
                'Callback', ['drivebot(''' r.name ''',' num2str(i) ')']);

            h2(i) = uicontrol(fig, 'Style', 'edit', ...
                'Units', 'pixels', ...
                'Position', [width*0.8 height*(n-i) width*0.2
height*0.4], ...
```

```
            'String', num2str(scale(i)*q(i)), ...
            'Tag', sprintf('Edit%d', i), ...
            'Callback', ['drivebot(''' r.name ''',' num2str(i) ')']);

        % hang handles off the slider and edit objects
        handles = {h(i) h2(i) scale};
        set(h(i), 'Userdata', handles);
        set(h2(i), 'Userdata', handles);
    end

    uicontrol(fig, 'Style', 'text', ...
        'Units', 'pixels', ...
        'FontSize', 20, ...
        'HorizontalAlignment', 'left', ...
        'Position', [0 height*(n+1) 0.8*width height], ...
        'BackgroundColor', 'white', ...
        'String', r.name);

    % X
    uicontrol(fig, 'Style', 'text', ...
        'Units', 'pixels', ...
        'BackgroundColor', bgcol, ...
        'Position', [0 height*(n+0.5) 0.06*width height/2], ...
        'BackgroundColor', 'yellow', ...
        'FontSize', 10, ...
        'HorizontalAlignment', 'left', ...
        'String', 'x:');

    h3(1,1) = uicontrol(fig, 'Style', 'edit', ...
        'Units', 'pixels', ...
        'Position', [0.06*width height*(n+0.5) width*0.2 height/2],
...

        'String', sprintf('%.3f', t6(1,4)), ...
        'Tag', 'T6');

    % Y
    uicontrol(fig, 'Style', 'text', ...
        'Units', 'pixels', ...
        'BackgroundColor', bgcol, ...
        'Position', [0.26*width height*(n+0.5) 0.06*width height/2],
...

        'BackgroundColor', 'yellow', ...
        'FontSize', 10, ...
        'HorizontalAlignment', 'left', ...
        'String', 'y:');

    h3(2,1) = uicontrol(fig, 'Style', 'edit', ...
        'Units', 'pixels', ...
        'Position', [0.32*width height*(n+0.5) width*0.2 height/2],
...

        'String', sprintf('%.3f', t6(2,4)));

    % Z
    uicontrol(fig, 'Style', 'text', ...
        'Units', 'pixels', ...
        'BackgroundColor', bgcol, ...
```

```
            'Position', [0.52*width height*(n+0.5) 0.06*width height/2],
...
            'BackgroundColor', 'yellow', ...
            'FontSize', 10, ...
            'HorizontalAlignment', 'left', ...
            'String', 'z:');

        h3(3,1) = uicontrol(fig, 'Style', 'edit', ...
            'Units', 'pixels', ...
            'Position', [0.58*width height*(n+0.5) width*0.2 height/2],
...
            'String', sprintf('%.3f', t6(3,4)));

        % AX
        uicontrol(fig, 'Style', 'text', ...
            'Units', 'pixels', ...
            'BackgroundColor', bgcol, ...
            'Position', [0 height*(n) 0.06*width height/2], ...
            'BackgroundColor', 'yellow', ...
            'FontSize', 10, ...
            'HorizontalAlignment', 'left', ...
            'String', 'ax:');

        h3(1,2) = uicontrol(fig, 'Style', 'edit', ...
            'Units', 'pixels', ...
            'Position', [0.06*width height*(n) width*0.2 height/2], ...
            'String', sprintf('%.3f', t6(1,3)));

        % AY
        uicontrol(fig, 'Style', 'text', ...
            'Units', 'pixels', ...
            'BackgroundColor', bgcol, ...
            'Position', [0.26*width height*(n) 0.06*width height/2], ...
            'BackgroundColor', 'yellow', ...
            'FontSize', 10, ...
            'HorizontalAlignment', 'left', ...
            'String', 'ay:');

        h3(2,2) = uicontrol(fig, 'Style', 'edit', ...
            'Units', 'pixels', ...
            'Position', [0.32*width height*(n) width*0.2 height/2], ...
            'String', sprintf('%.3f', t6(2,3)));

        % AZ
        uicontrol(fig, 'Style', 'text', ...
            'Units', 'pixels', ...
            'BackgroundColor', bgcol, ...
            'Position', [0.52*width height*(n) 0.06*width height/2], ...
            'BackgroundColor', 'yellow', ...
            'FontSize', 10, ...
            'HorizontalAlignment', 'left', ...
            'String', 'az:');

        h3(3,2) = uicontrol(fig, 'Style', 'edit', ...
            'Units', 'pixels', ...
            'Position', [0.58*width height*(n) width*0.2 height/2], ...
```

```
            'String', sprintf('%.3f', t6(3,3)));


        set(h3(1,1), 'Userdata', h3);
        uicontrol(fig, 'Style', 'pushbutton', ...
            'Units', 'pixels', ...
            'FontSize', 16, ...
            'Position', [0.8*width height*n 0.2*width 2*height], ...
            'CallBack', 'delete(gcf)', ...
            'BackgroundColor', 'red', ...
            'String', 'Quit');


        if isempty(rh),
            figure
            plot(r, q);
        end
    end
```

ANEXO 2: Programa del comando fkine

```
function t = fkine(robot, q)
    %
    % evaluate fkine for each point on a trajectory of
    % theta_i or q_i data
    %

    n = robot.n;

    L = robot.link;
    if length(q) == n,
        t = robot.base;
        for i=1:n,
            t = t * L{i}(q(i));
        end
        t = t * robot.tool;
    else
        if numcols(q) ~= n,
            error('bad data')
        end
        t = zeros(4,4,0);
        for qv=q',      % for each trajectory point
            tt = robot.base;
            for i=1:n,
                tt = tt * L{i}(qv(i));
            end
            t = cat(3, t, tt * robot.tool);
        end
    end
```

ANEXO 3: Programa del comando ikine

```
function qt = ikine(robot, tr, q, m)
    %
    %  solution control parameters
    %
    ilimit = 1000;
    stol = 1e-12;

    n = robot.n;

    if nargin == 2,
        q = zeros(n, 1);
    else
        q = q(:);
    end
    if nargin == 4,
        m = m(:);
        if length(m) ~= 6,
            error('Mask matrix should have 6 elements');
        end
        if length(find(m)) ~= robot.n
            error('Mask matrix must have same number of 1s as robot DOF')
        end
    else
        if n < 6,
            disp('For a manipulator with fewer than 6DOF a mask matrix
argument should be specified');
        end
        m = ones(6, 1);
    end


    tcount = 0;
    if ishomog(tr),      % single xform case
        nm = 1;
        count = 0;
        while nm > stol,
            e = tr2diff(fkine(robot, q'), tr) .* m;
            dq = pinv( jacob0(robot, q) ) * e;
            q = q + dq;
            nm = norm(dq);
            count = count+1;
            if count > ilimit,
                error('Solution wouldn''t converge')
            end
        end
        qt = q';
    else            % trajectory case
        np = size(tr,3);
        qt = [];
        for i=1:np
            nm = 1;
            T = tr(:,:,i);
            count = 0;
            while nm > stol,
```

```
                e = tr2diff(fkine(robot, q'), T) .* m;
                dq = pinv( jacob0(robot, q) ) * e;
                q = q + dq;
                nm = norm(dq);
                count = count+1;
                if count > ilimit,
                    fprintf('i=%d, nm=%f\n', i, nm);
                    error('Solution wouldn''t converge')
                end
            end
            qt = [qt; q'];
            tcount = tcount + count;
        end
    end
```

ANEXO 4: Programa del comando fdyn

```
function [t, q, qd] = fdyn(robot, t0, t1, torqfun, q0, qd0, varargin)

    % check the Matlab version, since ode45 syntax has changed
    v = ver;
    if str2num(v(1).Version)<6,
        %error('fdyn now requires Matlab version >= 6');
    end

    n = robot.n;
    if nargin == 3,
        torqfun = 0;
        x0 = zeros(2*n,1);
    elseif nargin == 4,
        x0 = zeros(2*n, 1);
    elseif nargin >= 6,
        x0 = [q0(:); qd0(:)];
    end

    [t,y] = ode45('fdyn2', [t0 t1], x0, [], robot, torqfun, varargin{:});
    q = y(:,1:n);
    qd = y(:,n+1:2*n);
```

Este programa hace uso del programa fdyn2 cuyo contenido es:

```
function xd = fdyn2(t, x, flag, robot, torqfun, varargin)

    n = robot.n;

    q = x(1:n);
    qd = x(n+1:2*n);

    % evaluate the torque function if one is given
    if isstr(torqfun)
        tau = feval(torqfun, t, q, qd, varargin{:});
    else
        tau = zeros(n,1);
    end

    qdd = accel(robot, x(1:n,1), x(n+1:2*n,1), tau);
    xd = [x(n+1:2*n,1); qdd];
```

ANEXO 5: Programa dind.m

```
DefWimajo % carga los parámetros de nuestro robot Wimajo

t = [0:0.056:5]'; % vector de tiempo
q_dmd = jtraj(qz, qr,t); % crea una trayectoria de puntos interpolados
qt = [t q_dmd];   % crea tabla de tiempos, puntos de paso
Pgain = [20 100 20 5 5]; % fija ganancia proporcional del controlador PD
Dgain = [-5 -10 -2 0 0]; % fija ganancia derivativa del controlador PD
[tsim,q,qd] = fdyn(Wimajo, 0, 5, 'taufunc1', qz, qz, Pgain, Dgain, qt);
```

ANEXO 6: Función taufunc1

```
function tau = taufunc1(t, q, qd, Pgain, Dgain, qt)
% interpolate demanded angles for this time
if t > qt(end,1), % keep time in range
t = qt(end,1);
end
q_dmd = interp1(qt(:,1), qt(:,2), t)';
% compute error and joint torque
e = q_dmd - q;
tau = diag(Pgain)*e + diag(Dgain)*qd;
```

ANEXO 7: Tabla parcial de valores [tsim,q,qd],

>> [tsim q qd]

ans =

0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0000	0.0001	-0.0000	-0.0000
0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0001	0.0001	-0.0000	-0.0000
0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0001	0.0002	-0.0000	-0.0000
0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0001	0.0002	-0.0000	-0.0000
0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0003	0.0005	-0.0001	-0.0000
0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0005	0.0007	-0.0002	-0.0000
0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0006	0.0010	-0.0002	-0.0000
0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0008	0.0012	-0.0003	-0.0000
0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0016	0.0025	-0.0006	-0.0000
0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0024	0.0037	-0.0008	-0.0000
0.0001	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0032	0.0049	-0.0011	-0.0000
0.0001	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0040	0.0062	-0.0014	-0.0000
0.0001	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0081	0.0122	-0.0028	-0.0000
0.0002	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0120	0.0181	-0.0041	-0.0000
0.0003	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0159	0.0239	-0.0053	-0.0000
0.0004	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0198	0.0296	-0.0066	-0.0000
0.0007	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0383	0.0559	-0.0120	-0.0000
0.0011	0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0555	0.0792	-0.0164	-0.0000
0.0014	0.0000	-0.0001	0.0001	-0.0000	-0.0000	0.0000	-0.0715	0.0996	-0.0198	-0.0000
0.0018	0.0000	-0.0001	0.0001	-0.0000	-0.0000	0.0000	-0.0865	0.1175	-0.0224	-0.0000
0.0023	0.0000	-0.0001	0.0002	-0.0000	-0.0000	0.0000	-0.1086	0.1414	-0.0248	-0.0000
0.0029	0.0000	-0.0002	0.0003	-0.0001	-0.0000	0.0000	-0.1284	0.1601	-0.0255	-0.0000
0.0035	0.0000	-0.0003	0.0004	-0.0001	-0.0000	0.0000	-0.1464	0.1743	-0.0245	-0.0000
0.0040	0.0000	-0.0004	0.0005	-0.0001	-0.0000	0.0000	-0.1628	0.1845	-0.0222	-0.0000
0.0046	0.0000	-0.0005	0.0006	-0.0001	-0.0000	0.0000	-0.1785	0.1917	-0.0184	-0.0000
0.0052	0.0000	-0.0006	0.0007	-0.0001	-0.0000	0.0000	-0.1928	0.1956	-0.0134	-0.0000
0.0058	0.0000	-0.0007	0.0008	-0.0001	-0.0000	0.0000	-0.2060	0.1968	-0.0074	-0.0000

0.0064	0.0000	-0.0008	0.0009	-0.0001	-0.0000	0.0000	-0.2182	0.1955	-0.0006	-0.0000
0.0071	0.0000	-0.0010	0.0011	-0.0001	-0.0000	0.0000	-0.2306	0.1917	0.0080	-0.0000
.										
.										
.										
4.9861	0.0000	-0.2600	-0.3540	0.6695	-0.0000	-0.0000	0.0774	0.0211	-0.0666	0.0000
4.9896	0.0000	-0.2598	-0.3539	0.6693	-0.0000	-0.0000	0.0791	0.0303	-0.0091	0.0000
4.9931	0.0000	-0.2595	-0.3537	0.6694	-0.0000	-0.0000	0.0806	0.0394	0.0484	0.0000
4.9965	0.0000	-0.2592	-0.3536	0.6697	-0.0000	-0.0000	0.0821	0.0482	0.1058	0.0000
5.0000	0.0000	-0.2589	-0.3534	0.6701	-0.0000	-0.0000	0.0835	0.0568	0.1631	0.0000

ANEXO 8: Programa kitrajdyn.m

```
%invoca a un robot
wymajo;
%kinematics
T=fkine(p560,qz);
qi=ikine(p560,T);

D=[.1 .2 0 -.2 .1 .1]';
diff2tr(D)

%T=transl(100, 200, 300)*roty(pi/8)*rotz(-pi/4);
DT=tr2jac(T)*D;
DTrasp=DT';

J=jacobn(p560, qz);
%det(J)

%Trajectories
t=[0:0.056:2]';
[q, qd, qdd]=jtraj(qz, qr, t);
Tq=fkine(p560, q);
subplot(5,1,1), plot(q(:,1)),grid on,title('(Desplazamiento angular de
Articulación q1)')
subplot(5,1,2), plot(q(:,2)),grid on,title('(Desplazamiento angular de
Articulación q2)')
subplot(5,1,3), plot(q(:,3)),grid on,title('(Desplazamiento angular de
Articulación q3)')
subplot(5,1,4), plot(q(:,4)),grid on,title('(Desplazamiento angular de
Articulación q4)')
subplot(5,1,5), plot(q(:,5)),grid on,title('(Desplazamiento angular de
Articulación q5)')
%break
figure
subplot(5,1,1), plot(qd(:,1)),grid on,title('(Velocidad angular de
Articulación qd1)')
subplot(5,1,2), plot(qd(:,2)),grid on,title('(Velocidad angular de
Articulación qd2)')
subplot(5,1,3), plot(qd(:,3)),grid on,title('(Velocidad angular de
Articulación qd3)')
subplot(5,1,4), plot(qd(:,4)),grid on,title('(Velocidad angular de
Articulación qd4)')
subplot(5,1,5), plot(qd(:,5)),grid on,title('(Velocidad angular de
Articulación qd5)')
%break
figure
subplot(5,1,1), plot(qdd(:,1)),grid on,title('(Aceleración angular de
Articulación qdd1)')
subplot(5,1,2), plot(qdd(:,2)),grid on,title('(Aceleración angular de
Articulación qdd2)')
subplot(5,1,3), plot(qdd(:,3)),grid on,title('(Aceleración angular de
Articulación qdd3)')
subplot(5,1,4), plot(qdd(:,4)),grid on,title('(Aceleración angular de
Articulación qdd4)')
subplot(5,1,5), plot(qdd(:,5)),grid on,title('(Aceleración angular de
Articulación qdd5)')
```

```
%break
figure
%Dynamics
tau=rne(p560, q, qd, qdd);
%plot(t, tau),grid on,title('Totques')
subplot(5,1,1), plot(t, tau(:,1)),grid on,title('(Torques de Articulación
1)')
subplot(5,1,2), plot(t, tau(:,2)),grid on,title('(Torques de Articulación
2)')
subplot(5,1,3), plot(t, tau(:,3)),grid on,title('(Torques de Articulación
3)')
subplot(5,1,4), plot(t, tau(:,4)),grid on,title('(Torques de Articulación
4)')
subplot(5,1,5), plot(t, tau(:,5)),grid on,title('(Torques de Articulación
5)')
%break
figure
tau_g=gravload(p560, q);
subplot(5,1,1),plot(t, tau_g(:,1)),grid on,title('Gravedad de
Articulación 1')
subplot(5,1,2),plot(t, tau_g(:,2)),grid on,title('Gravedad de
Articulación 2')
subplot(5,1,3),plot(t, tau_g(:,3)),grid on,title('Gravedad de
Articulación 3')
subplot(5,1,4),plot(t, tau_g(:,4)),grid on,title('Gravedad de
Articulación 4')
subplot(5,1,5),plot(t, tau_g(:,5)),grid on,title('Gravedad de
Articulación 5')
%break
M11=[];
for qi=q',
M=inertia(p560, qi');
M11=[M11;M(1,1)];
end
M12=[];
for qi=q',
M=inertia(p560, qi');
M12=[M12;M(1,2)];
end
M13=[];
for qi=q',
M=inertia(p560, qi');
M13=[M13;M(1,3)];
end
M14=[];
for qi=q',
M=inertia(p560, qi');
M14=[M14;M(1,4)];
end
M15=[];
for qi=q',
M=inertia(p560, qi');
M15=[M15;M(1,5)];
end
figure
subplot(5,1,1),plot(t,M11),grid on,title('Inercia Articulación 1')
subplot(5,1,2),plot(t,M12),grid on,title('Inercia Articulación 2')
```

```
subplot(5,1,3),plot(t,M13),grid on,title('Inercia Articulación 3')
subplot(5,1,4),plot(t,M14),grid on,title('Inercia Articulación 4')
subplot(5,1,5),plot(t,M15),grid on,title('Inercia Articulación 5')
```

ANEXO 9: Gráficas de desplazamientos, velocidades, aceleraciones, torque, gravedad e inercias

La figura 1.1 se muestra las gráficas de los desplazamientos angulares de las 5 articulaciones.

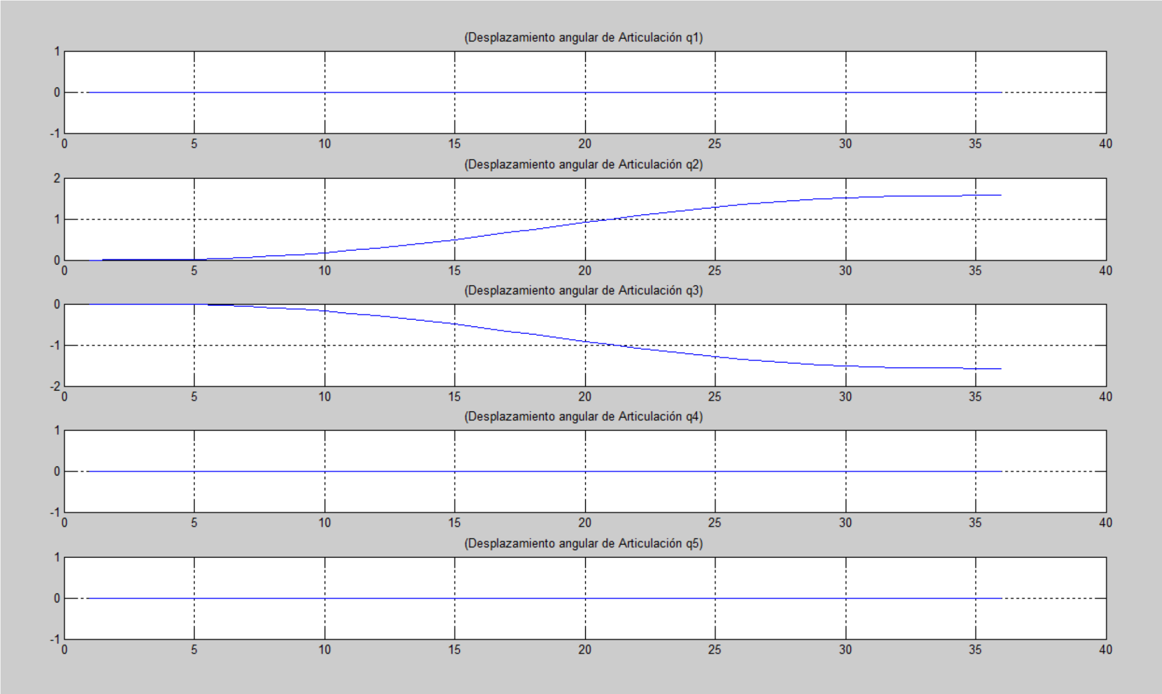


Figura 1.1 Desplazamientos angulares de las articulaciones q1, q2, q3, q4 y q5

En la figura 1.2 se muestran las gráficas de las velocidades angulares de las 5 articulaciones.

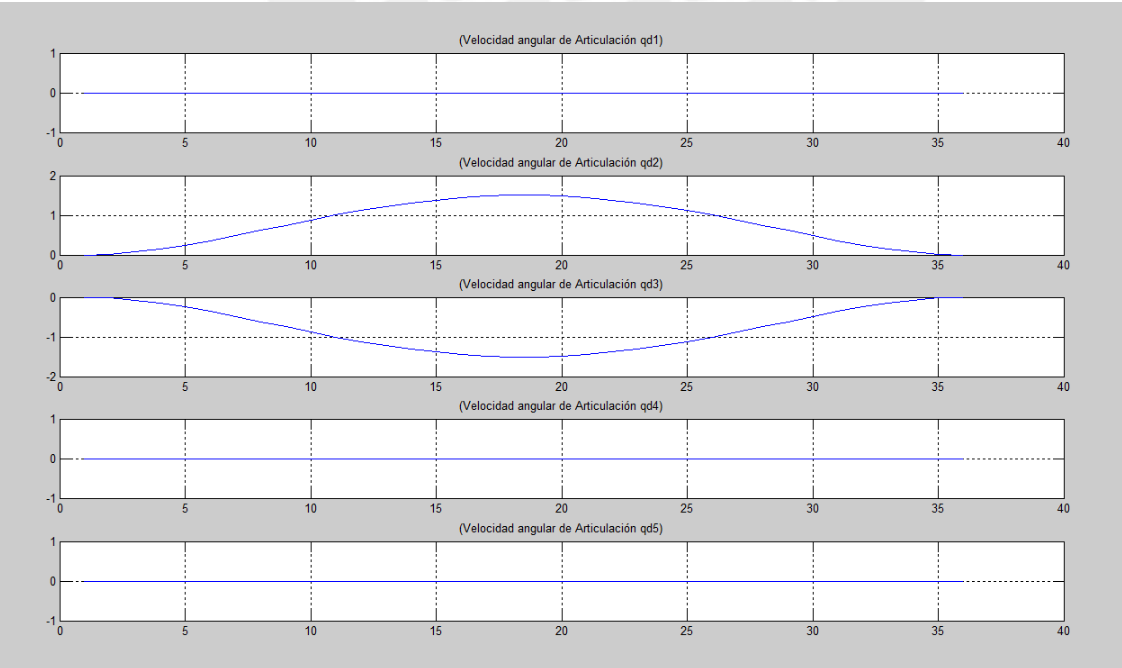


Figura 1.2 Velocidades angulares de las articulaciones q1, q2, q3, q4 y q5

En la figura 1.3 se muestran las gráficas de las aceleraciones angulares de las 5 articulaciones.

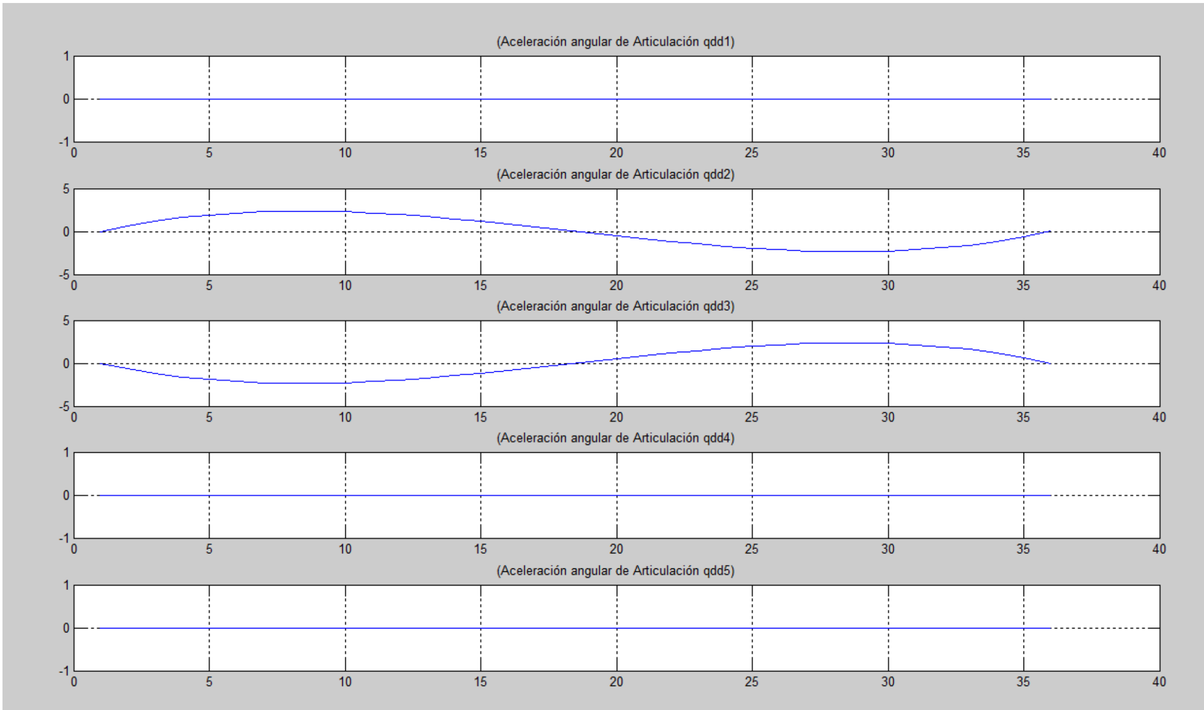


Figura 1.3 Aceleraciones angulares de las articulaciones q1, q2, q3, q4 y q5

En la figura 1.4 se muestran las gráficas de los torques de las 5 articulaciones.

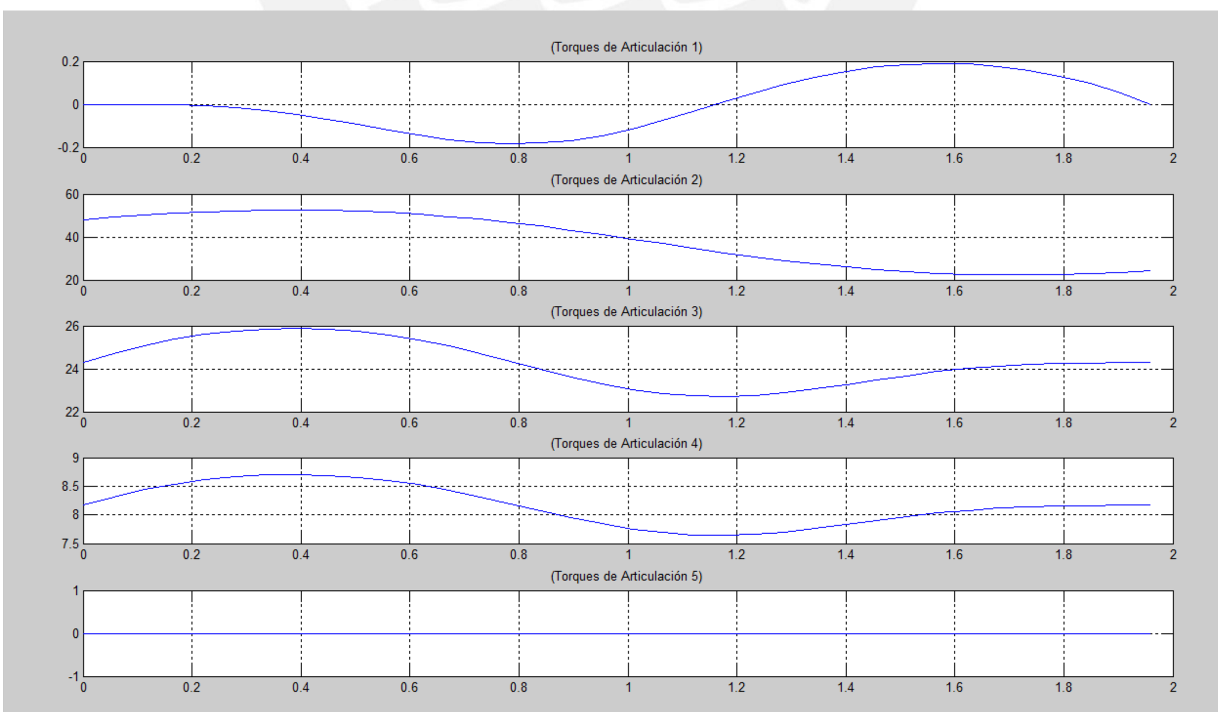


Figura 1.4 Torques de las articulaciones q1, q2, q3, q4 y q5

En la figura 1.5 se muestran las gráficas de la gravedad de las 5 articulaciones.

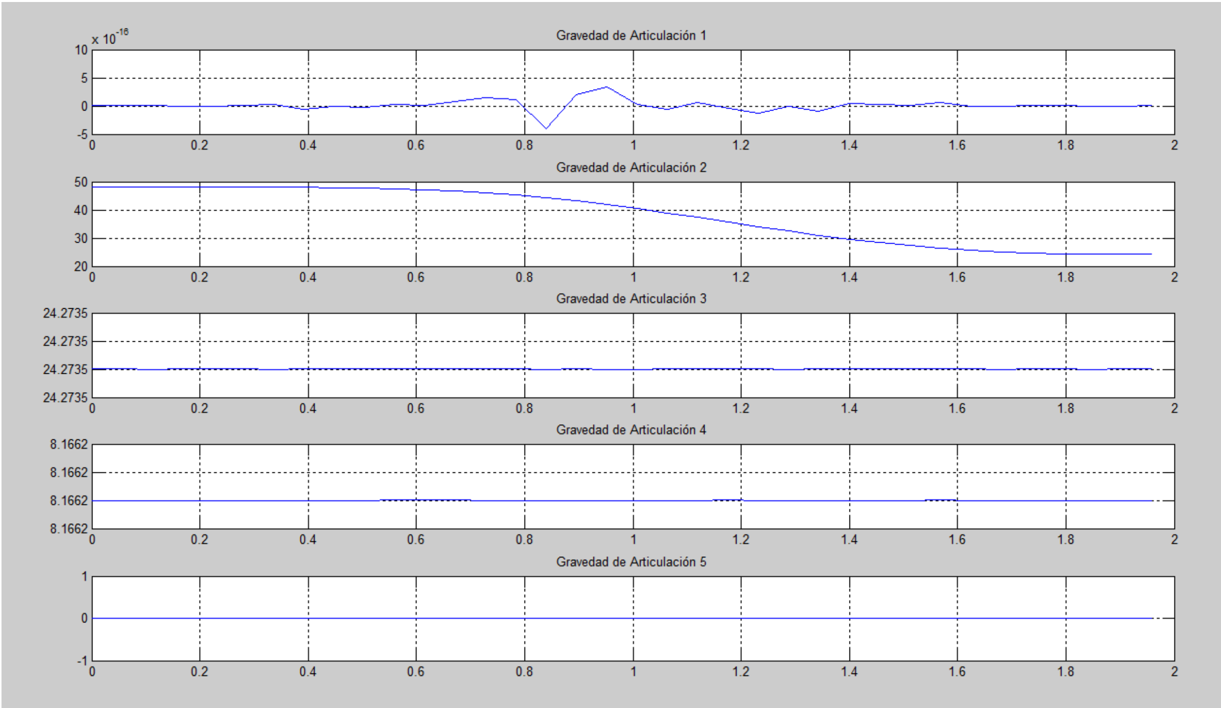


Figura 1.5 Inercia de las articulaciones q1, q2, q3, q4 y q5

En la figura 1.6 se muestran las gráficas de inercia de las 5 articulaciones.

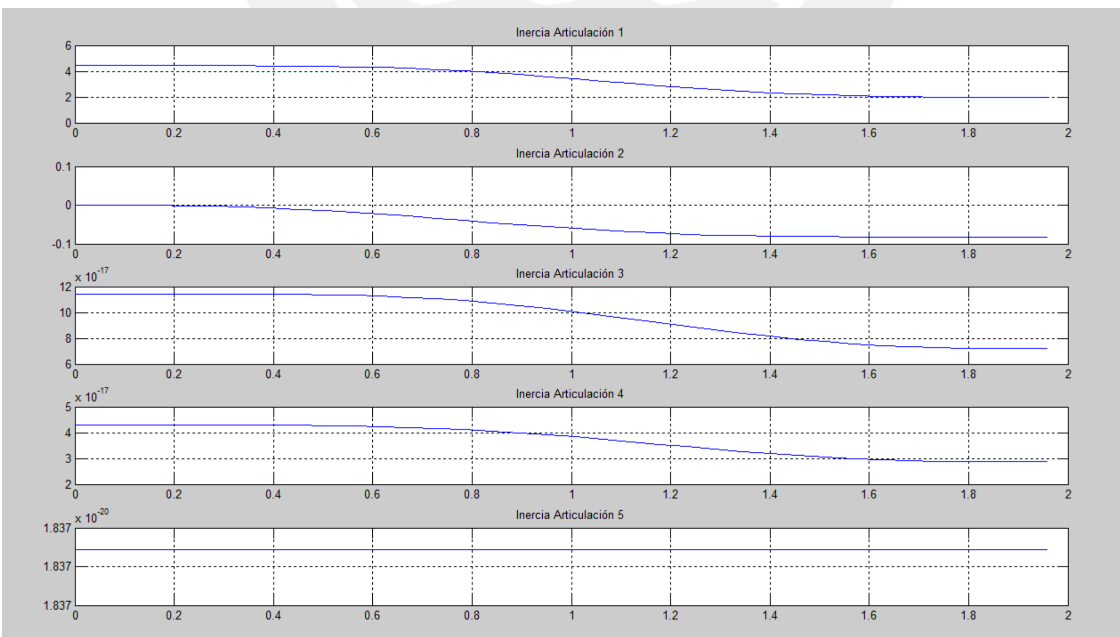


Figura 1.6 Inercias de las articulaciones q1, q2, q3, q4 y q5

ANEXO 10: Programa jtraj.m

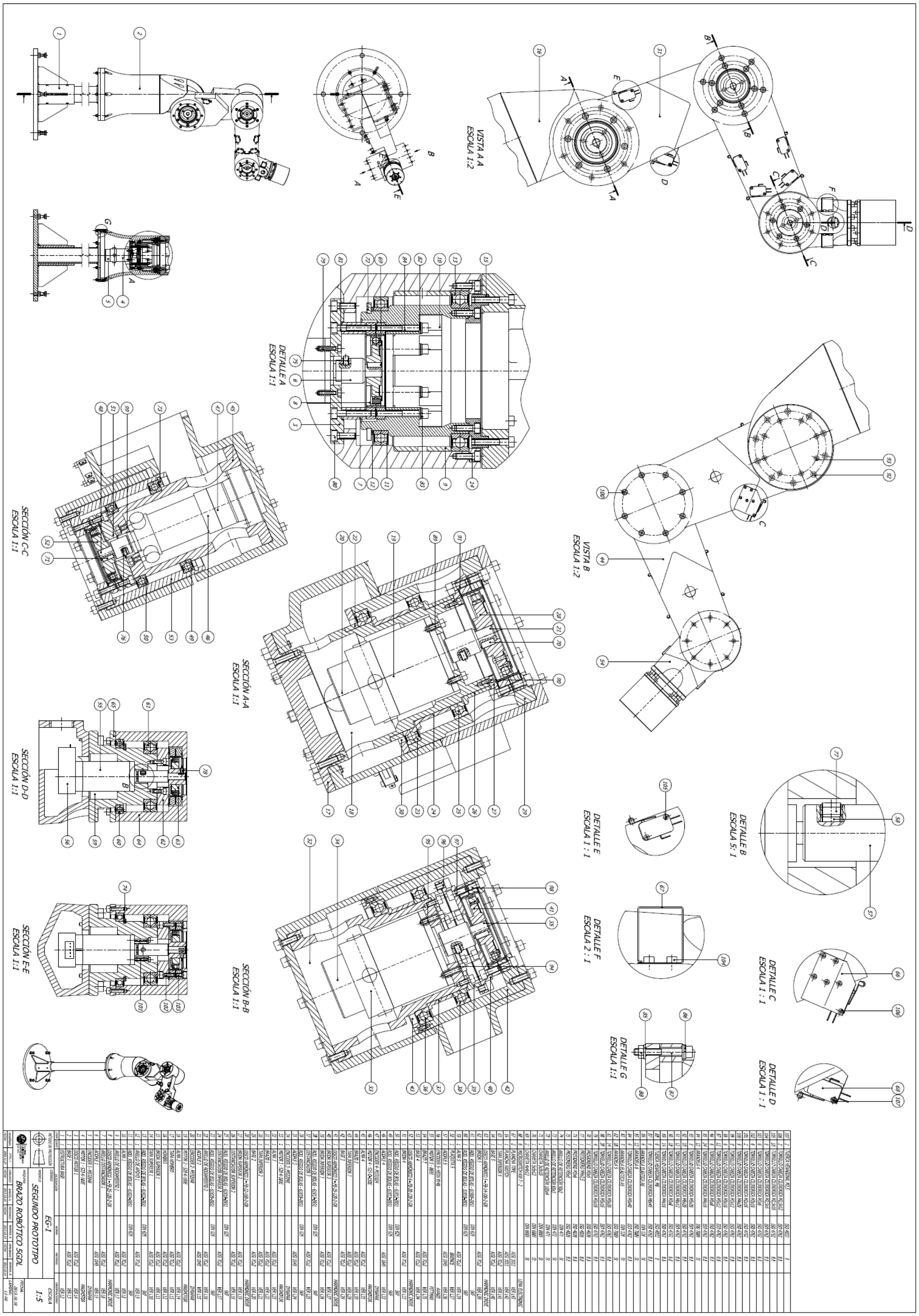
```
function [qt,qdt,qddt] = jtraj(q0, q1, tv, qd0, qd1)
    if length(tv) > 1,
        tscal = max(tv);
        t = tv(:)/tscal;
    else
        tscal = 1;
        t = [0:(tv-1)]'/(tv-1); % normalized time from 0 -> 1
    end

    q0 = q0(:);
    q1 = q1(:);

    if nargin == 3,
        qd0 = zeros(size(q0));
        qd1 = qd0;
    end

    % compute the polynomial coefficients
    A = 6*(q1 - q0) - 3*(qd1+qd0)*tscal;
    B = -15*(q1 - q0) + (8*qd0 + 7*qd1)*tscal;
    C = 10*(q1 - q0) - (6*qd0 + 4*qd1)*tscal;
    E = qd0*tscal; % as the t vector has been normalized
    F = q0;

    tt = [t.^5 t.^4 t.^3 t.^2 t ones(size(t))];
    c = [A B C zeros(size(A)) E F]';

    qt = tt*c;

    % compute optional velocity
    if nargout >= 2,
        c = [ zeros(size(A)) 5*A 4*B 3*C  zeros(size(A)) E ]';
        qdt = tt*c/tscal;
    end

    % compute optional acceleration
    if nargout == 3,
        c = [ zeros(size(A))  zeros(size(A)) 20*A 12*B 6*C
zeros(size(A))]';
        qddt = tt*c/tscal^2;
    end
```

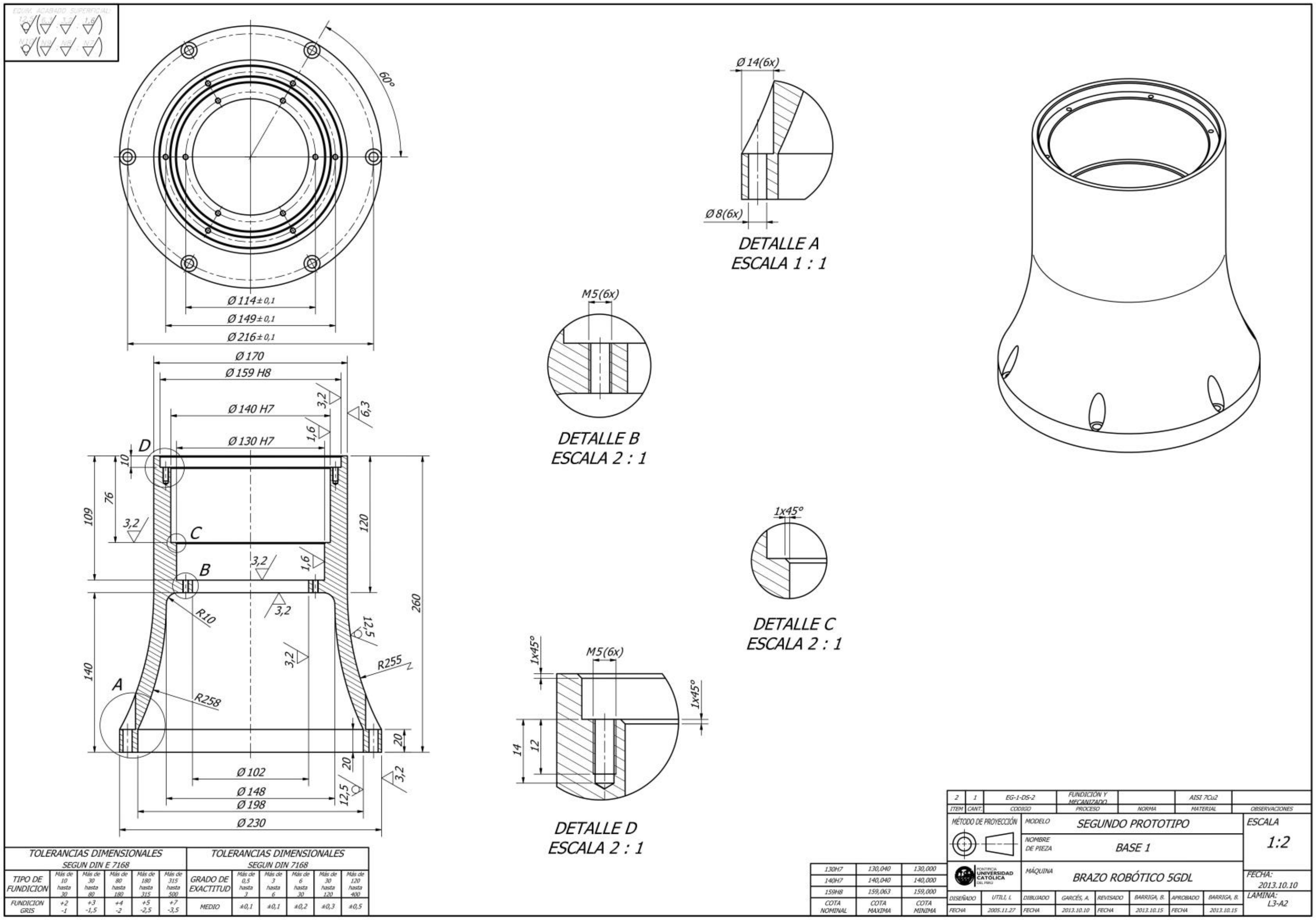
ANEXO 11: Programa rne

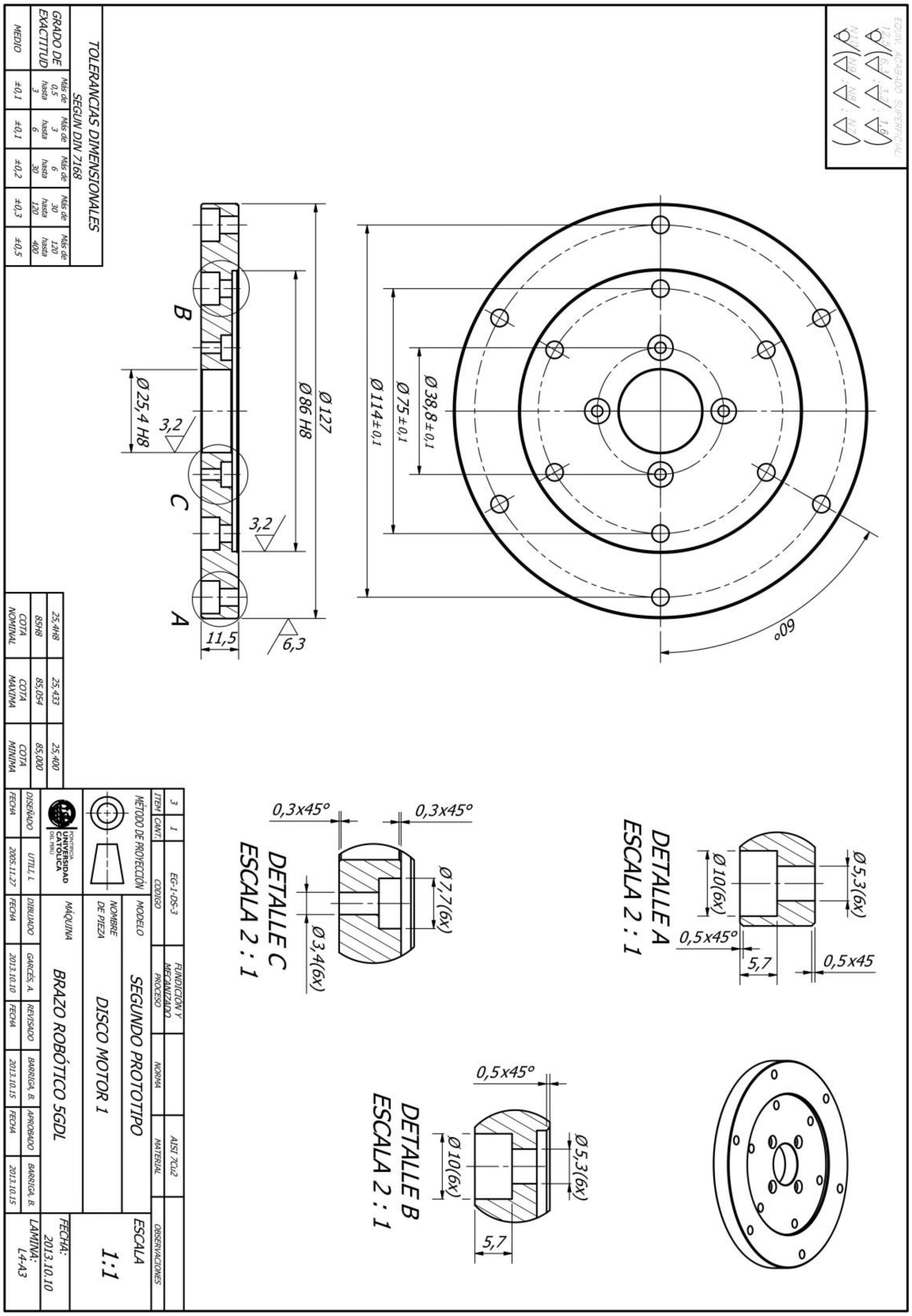
```
function tau = rne(robot, a1, a2, a3, a4, a5)
    if robot.mdh ~= 0,
        error('Jacobian only valid for standard D&H parameters')
    end
    z0 = [0;0;1];
    robot.gravity = z0;            % agregué esto
    grav = robot.gravity;   % default gravity from the object
    fext = zeros(6, 1);

    n = robot.n;
    if numcols(a1) == 3*n,
        Q = a1(:,1:n);
        Qd = a1(:,n+1:2*n);
        Qdd = a1(:,2*n+1:3*n);
        np = numrows(Q);
        if nargin >= 3,
            grav = a2;
        end
        if nargin == 4,
            fext = a3;
        end
    else
        np = numrows(a1);
        Q = a1;
        Qd = a2;
        Qdd = a3;
        if numcols(a1) ~= n | numcols(Qd) ~= n | numcols(Qdd) ~= n | ...
            numrows(Qd) ~= np | numrows(Qdd) ~= np,
            error('bad data');
        end
        if nargin >= 5,
            grav = a4;
        end
        if nargin == 6,
            fext = a5;
        end
    end

    tau = zeros(np,n);

    for p=1:np,
        q = Q(p,:)';
        qd = Qd(p,:)';
        qdd = Qdd(p,:)';

        Fm = [];
        Nm = [];
        pstarm = [];
        Rm = [];
        w = zeros(3,1);
        wd = zeros(3,1);
        v = zeros(3,1);
        vd = grav;
```

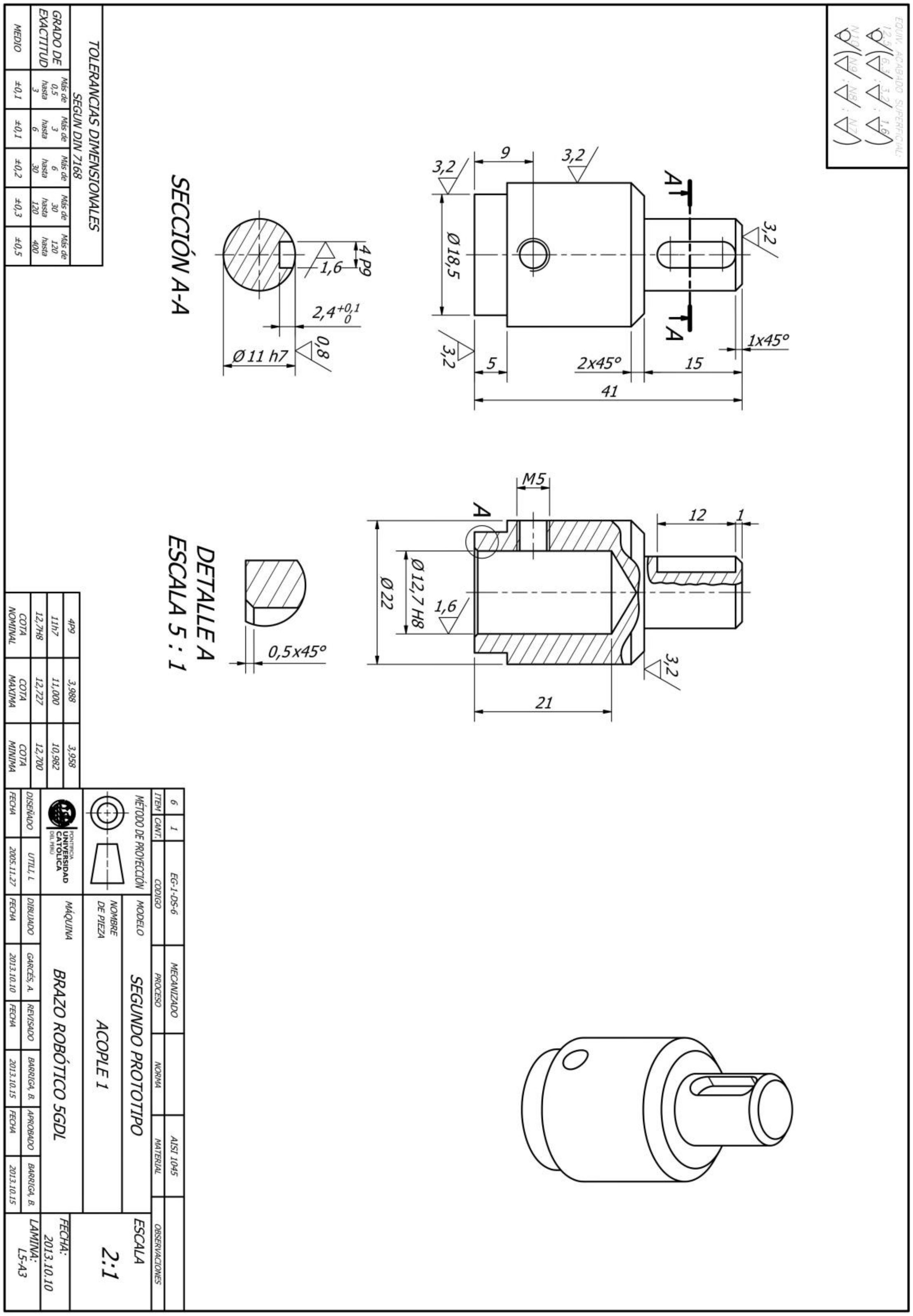
```
%
% init some variables, compute the link rotation matrices
%
    for j=1:n,
        link = robot.link{j};
        Tj = link(q(j));
        Rm{j} = tr2rot(Tj);
        if link.RP == 'R',
            D = link.D;
        else
            D = q(j);
        end
        alpha = link.alpha;
        pstarm(:,j) = [link.A; D*sin(alpha); D*cos(alpha)];
    end

%
%  the forward recursion
%
    for j=1:n,
        link = robot.link{j};

        R = Rm{j}';
        pstar = pstarm(:,j);
        r = link.r;

        %
        % statement order is important here
        %
        if link.RP == 'R',
            % revolute axis
            wd = R*(wd + z0*qdd(j) + ...
                cross(w,z0*qd(j)));
            w = R*(w + z0*qd(j));
            %v = cross(w,pstar) + R*v;
            vd = cross(wd,pstar) + ...
                cross(w, cross(w,pstar)) +R*vd;

        else
            % prismatic axis
            w = R*w;
            wd = R*wd;
            vd = R*(z0*qdd(j)+vd) + ...
                cross(wd,pstar) + ...
                2*cross(w,R*z0*qd(j)) +...
                cross(w, cross(w,pstar));
        end

        vhat = cross(wd,r) + ...
            cross(w,cross(w,r)) + vd;
        F = link.m*vhat;
        N = link.I*wd + cross(w,link.I*w);
        Fm = [Fm F];
        Nm = [Nm N];
    end
```

```
%
%   the backward recursion
%

    f = fext(1:3);      % force/moments on end of arm
    nn = fext(4:6);

    for j=n:-1:1,
        link = robot.link{j};
        pstar = pstarm(:,j);

        %
        % order of these statements is important, since both
        % nn and f are functions of previous f.
        %
        if j == n,
            R = eye(3,3);
        else
            R = Rm{j+1};
        end
        r = link.r;
        nn = R*(nn + cross(R'*pstar,f)) + ...
            cross(pstar+r,Fm(:,j)) + ...
            Nm(:,j);
        f = R*f + Fm(:,j);
        R = Rm{j};
        if link.RP == 'R',
            % revolute
            tau(p,j) = nn'*(R'*z0) + ...
                link.G^2 * ( link.Jm*qdd(j) + ...
                    friction(link, qd(j)) ...
                );
        else
            % prismatic
            tau(p,j) = f'*(R'*z0) + ...
                link.G^2 * ( link.Jm*qdd(j) + ...
                    friction(link, qd(j)) ...
                );
        end
    end
end
```


VISTA A-A
ESCALA 1:2
DETALLE A
ESCALA 1:1
SECCIÓN C-C
ESCALA 1:1
SECCIÓN A-A
ESCALA 1:1
VISTA B
ESCALA 1:2
SECCIÓN D-D
ESCALA 1:1
SECCIÓN E-E
ESCALA 1:1
SECCIÓN B-B
ESCALA 1:1
DETALLE E
ESCALA 1 : 1
DETALLE B
ESCALA 5: 1
DETALLE F
ESCALA 2 : 1
DETALLE C
ESCALA 1 : 1
DETALLE G
ESCALA 1:1
DETALLE D
ESCALA 1 : 1
SEGUNDO PROTOTIPO
BRAZO ROBÓTICO 5GDL
EG-1
1:5

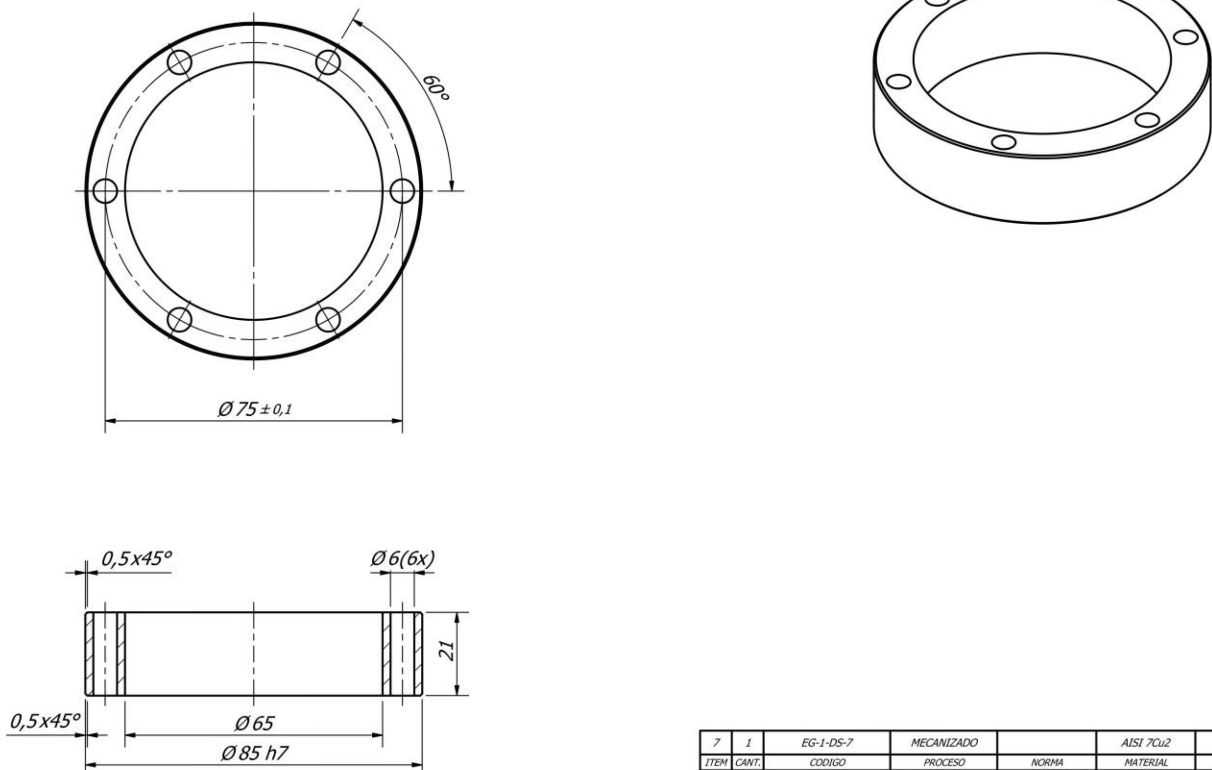

Ø114±0,1
Ø149±0,1
Ø216±0,1
60°
Ø170
Ø159 H8
Ø140 H7
Ø130 H7
Ø102
Ø148
Ø198
Ø230
109
76
10
120
140
260
20
R10
R258
R255
12,5
3,2
1,6
6,3
A
B
C
D
Ø14(6x)
Ø8(6x)
DETALLE A
ESCALA 1 : 1
M5(6x)
DETALLE B
ESCALA 2 : 1
1x45°
DETALLE C
ESCALA 2 : 1
14
12
DETALLE D
ESCALA 2 : 1
TOLERANCIAS DIMENSIONALES
SEGUN DIN E 7168
TIPO DE FUNDICION
FUNDICION GRIS
TOLERANCIAS DIMENSIONALES
SEGUN DIN 7168
GRADO DE EXACTITUD
MEDIO
130H7 130,040 130,000
140H7 140,040 140,000
159H8 159,063 159,000
COTA NOMINAL
COTA MAXIMA
COTA MINIMA
2 1 EG-1-DS-2 FUNDICIÓN Y MECANIZADO AISI 7Cu2
ITEM CANT. CODIGO PROCESO NORMA MATERIAL OBSERVACIONES
MÉTODO DE PROYECCIÓN
MODELO SEGUNDO PROTOTIPO
NOMBRE DE PIEZA BASE 1
MÁQUINA BRAZO ROBÓTICO 5GDL
ESCALA 1:2
FECHA: 2013.10.10
LAMINA: L3-A2
DISEÑADO UTILI, I. FECHA 2005.11.27
DIBUJADO GARCÉS, A. FECHA 2013.10.10
REVISADO BARRIGA, B. FECHA 2013.10.15
APROBADO BARRIGA, B. FECHA 2013.10.15


TOLERANCIAS DIMENSIONALES SEGÚN DIN 7168
GRADO DE EXACTITUD
MEDIO
Más de 0,5 hasta 3 ±0,1
Más de 3 hasta 6 ±0,1
Más de 6 hasta 30 ±0,2
Más de 30 hasta 120 ±0,3
Más de 120 hasta 400 ±0,5
Ø127
Ø86 H8
Ø114±0,1
Ø75±0,1
Ø38,8±0,1
Ø25,4 H8
3,2
6,3
11,5
60°
A
B
C
DETALLE A ESCALA 2 : 1
Ø5,3(6x)
Ø10(6x)
0,5x45°
5,7
0,5x45
DETALLE B ESCALA 2 : 1
DETALLE C ESCALA 2 : 1
Ø7,7(6x)
Ø3,4(6x)
0,3x45°
COTA NOMINAL
COTA MÁXIMA
COTA MÍNIMA
25,4H8
25,433
25,400
85H8
85,054
85,000
ITEM
CANT.
CÓDIGO
MODELO
3
1
EG-1-DS-3
FUNDICIÓN Y MECANIZADO
PROCESO
NORMA
MATERIAL
AISI 7CU2
OBSERVACIONES
MÉTODO DE PROYECCIÓN
SEGUNDO PROTOTIPO
NOMBRE DE PIEZA
DISCO MOTOR 1
MÁQUINA
BRAZO ROBÓTICO 5GDL
ESCALA
1:1
DISEÑADO
DIBUJADO
REVISADO
APROBADO
GARCÉS, A.
BARRIGA, B.
FECHA
2013.10.10
2013.10.15
FECHA:
2013.10.10
LAMINA:
L4-A3



TOLERANCIAS DIMENSIONALES SEGÚN DIN 7168
GRADO DE EXACTITUD
MEDIO
±0,1
±0,1
±0,2
±0,3
±0,5
SECCIÓN A-A
DETALLE A
ESCALA 5 : 1
0,5x45º
Ø 11 h7
2,4+0,1 0
4 P9
1,6
0,8
Ø 18,5
3,2
9
5
2x45º
15
41
1x45º
M5
12
1
Ø 22
Ø 12,7 H8
21
4P9
11h7
12,7H8
COTA NOMINAL
3,988
11,000
12,727
COTA MÁXIMA
3,958
10,982
12,700
COTA MÍNIMA
SEGUNDO PROTOTIPO
ACOPLE 1
BRAZO ROBÓTICO 5GDL
EG-1-DS-6
MECANIZADO
AISI 1045
ESCALA
2:1
FECHA: 2013.10.10
LAMINA: L5-A3
MÉTODO DE PROYECCIÓN
DISEÑADO
UTILL, L
2005.11.27
DIBUJADO
GARCÉS, A.
2013.10.10
REVISADO
BARRIGA, B.
2013.10.15
APROBADO
BARRIGA, B.
2013.10.15

EQUIV. ACABADO SUPERFICIAL:			
12,5	6,3	3,2	1,6
N10	N9	N8	N7



TOLERANCIAS DIMENSIONALES SEGUN DIN 7168

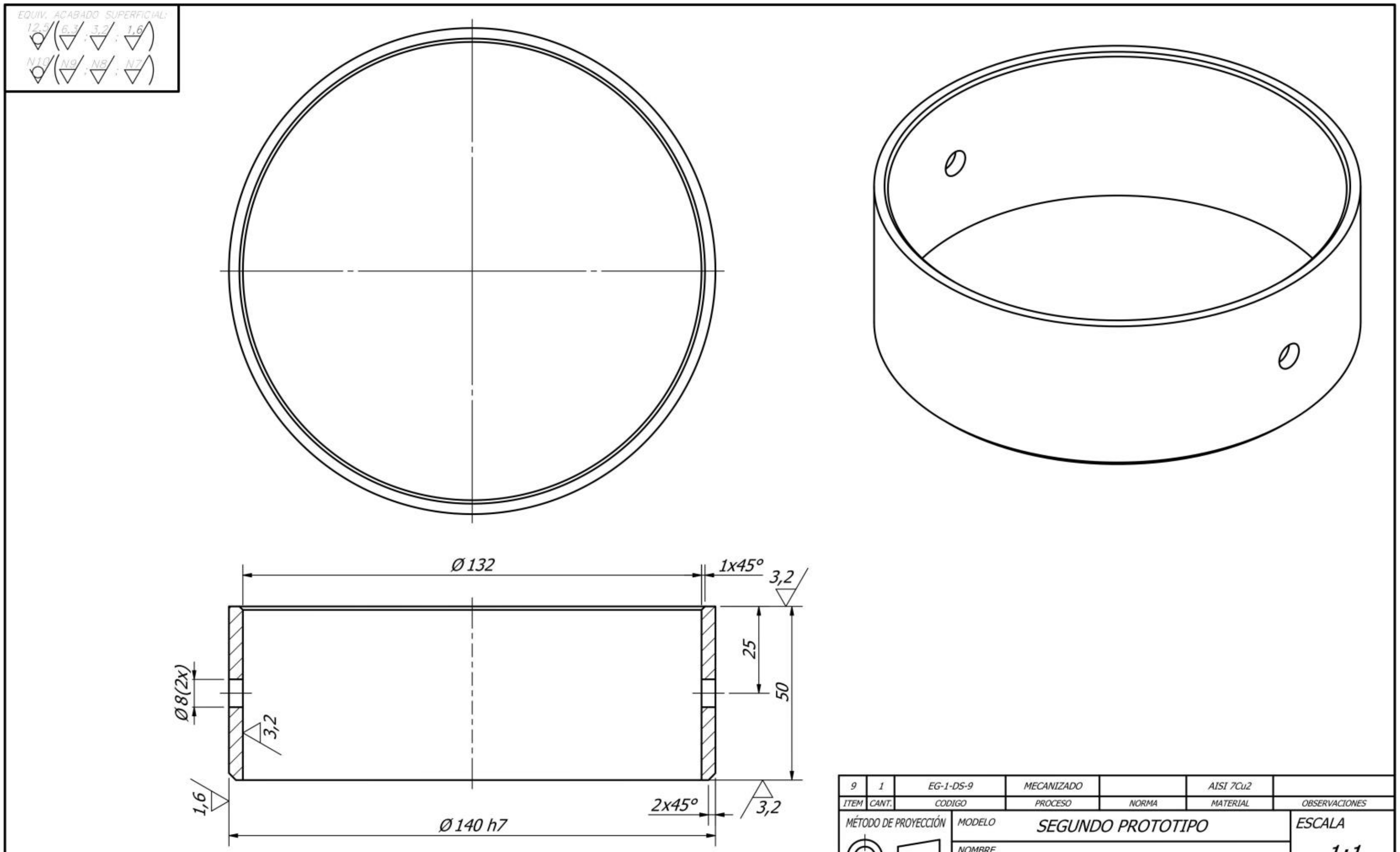
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

COTA NOMINAL	COTA MAXIMA	COTA MINIMA
85h7	85,000	84,965

ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
7	1	EG-1-DS-7	MECANIZADO		AISI 7Cu2	

MÉTODO DE PROYECCIÓN	MODELO	SEGUNDO PROTOTIPO	ESCALA 1:1
	NOMBRE DE PIEZA	ANILLO DISTANCIADOR 1	
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ	MÁQUINA	BRAZO ROBÓTICO 5GDL	FECHA: 2013.10.10

DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO	BARRIGA, B.	LAMINA: L6-A3
FECHA	2005.11.27	FECHA	2013.10.10	FECHA	2013.10.15	FECHA	2013.10.15	



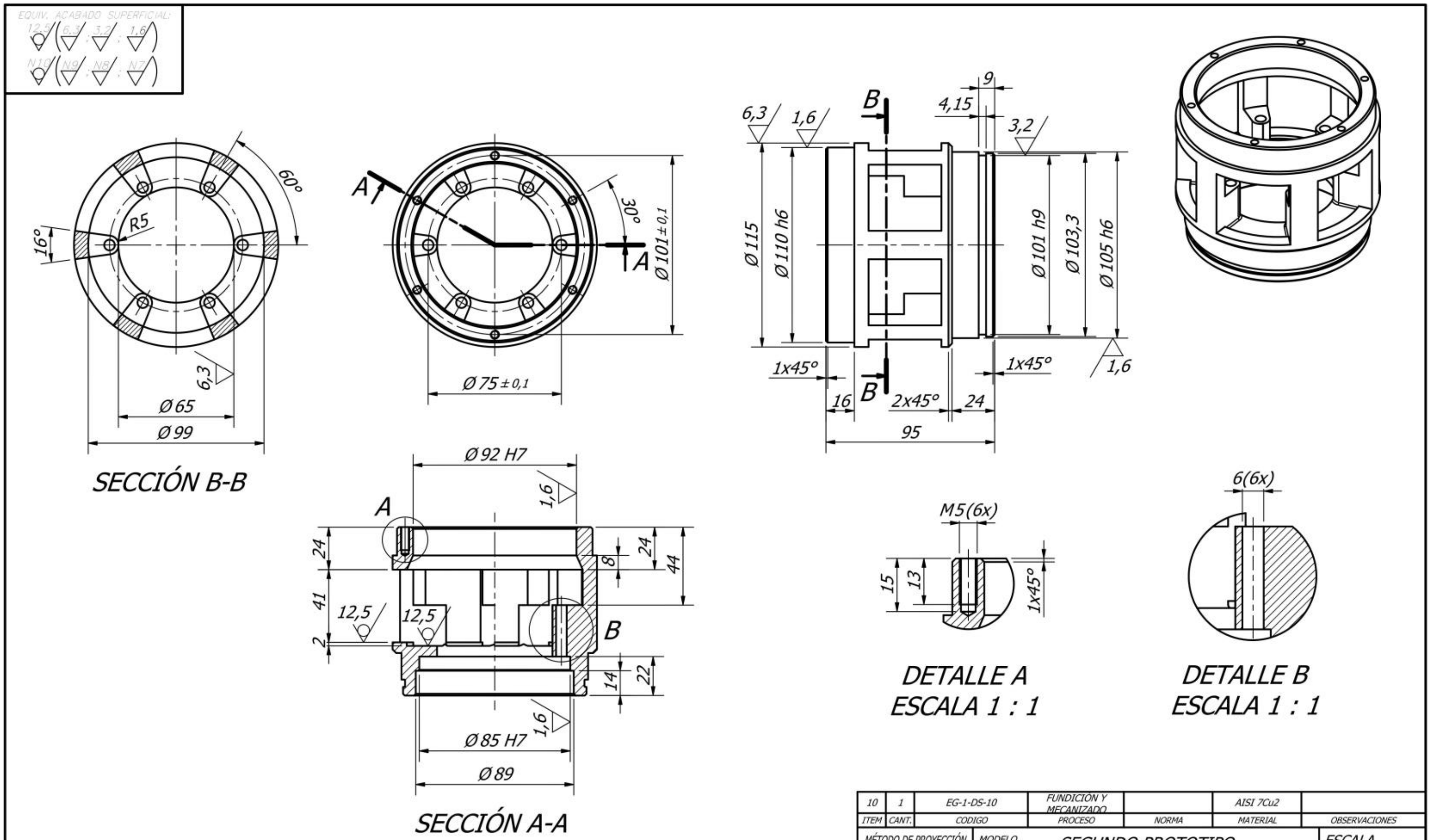
TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

COTA NOMINAL	COTA MAXIMA	COTA MINIMA
140h7	140,000	139,960

ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
9	1	EG-1-DS-9	MECANIZADO		AISI 7Cu2	

MÉTODO DE PROYECCIÓN	MODELO	SEGUNDO PROTOTIPO	ESCALA 1:1
	NOMBRE DE PIEZA	ANILLO DE RODAMIENTO 1	
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ	MÁQUINA	BRAZO ROBÓTICO 5GDL	FECHA: 2013.10.10

DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO	BARRIGA, B.	LAMINA: L7-A3
FECHA	2005.11.27	FECHA	2013.10.10	FECHA	2013.10.15	FECHA	2013.10.15	

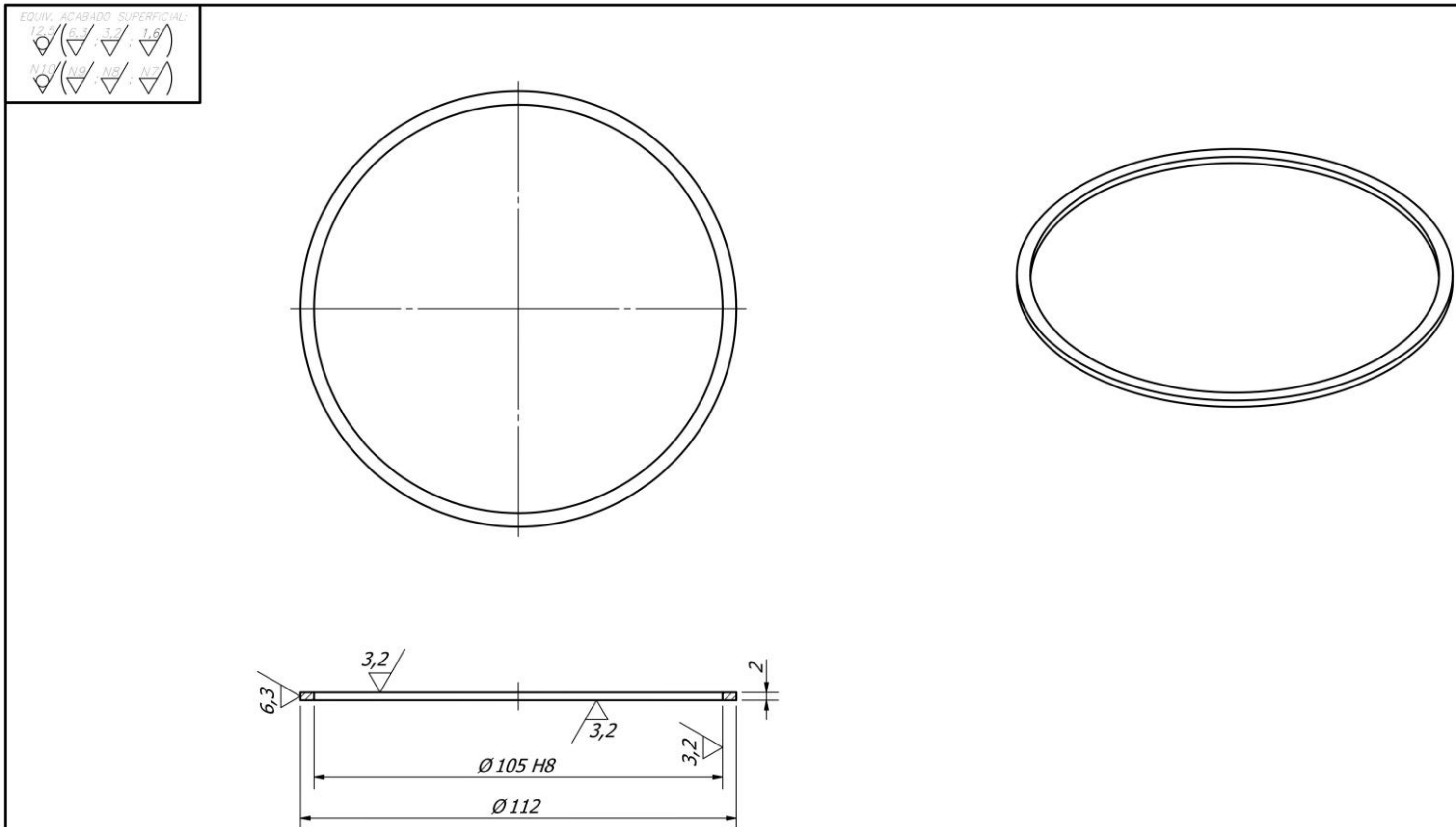


TOLERANCIAS DIMENSIONALES SEGUN DIN E 7168					
TIPO DE FUNDICION	Más de 10 hasta 30	Más de 30 hasta 80	Más de 80 hasta 180	Más de 180 hasta 315	Más de 315 hasta 500
FUNDICION GRIS	+2 -1	+3 -1,5	+4 -2	+5 -2,5	+7 -3,5

TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

85H7	85,035	85,000
92H7	92,035	92,000
101h9	101,000	100,913
105h6	105,000	104,978
110h6	110,000	109,978
COTA NOMINAL	COTA MAXIMA	COTA MINIMA

10	1	EG-1-DS-10	FUNDICIÓN Y MECANIZADO		AISI 7Cu2		
ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL		OBSERVACIONES
MÉTODO DE PROYECCIÓN		MODELO	SEGUNDO PROTOTIPO				ESCALA
		NOMBRE DE PIEZA	ALMA 1				1:2
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ		MÁQUINA	BRAZO ROBÓTICO 5GDL				FECHA: 2013.10.10
DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO BARRIGA, B.	LAMINA: L8-A3
FECHA	2005.11.27	FECHA	2013.10.10	FECHA	2013.10.15	FECHA 2013.10.15	

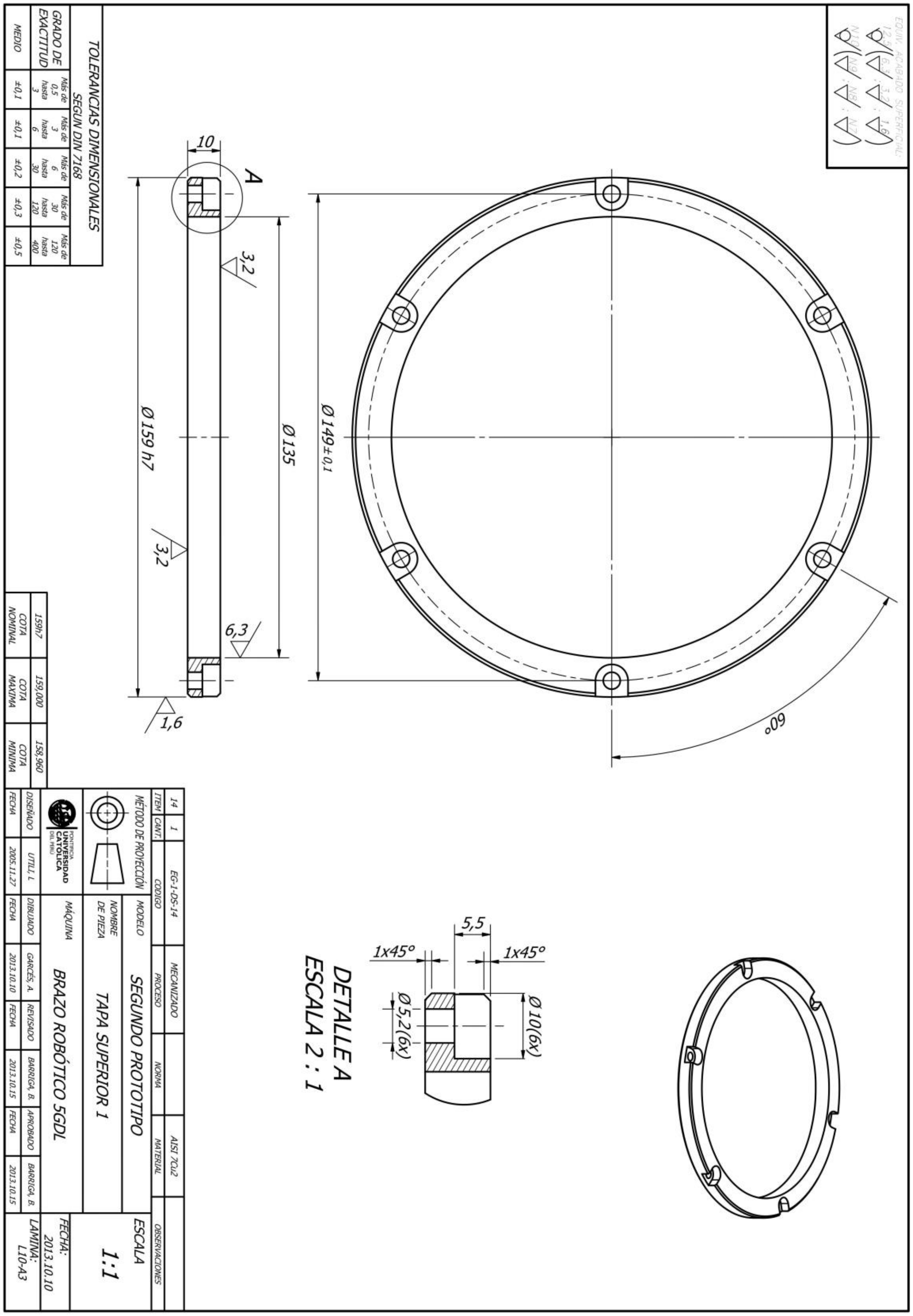


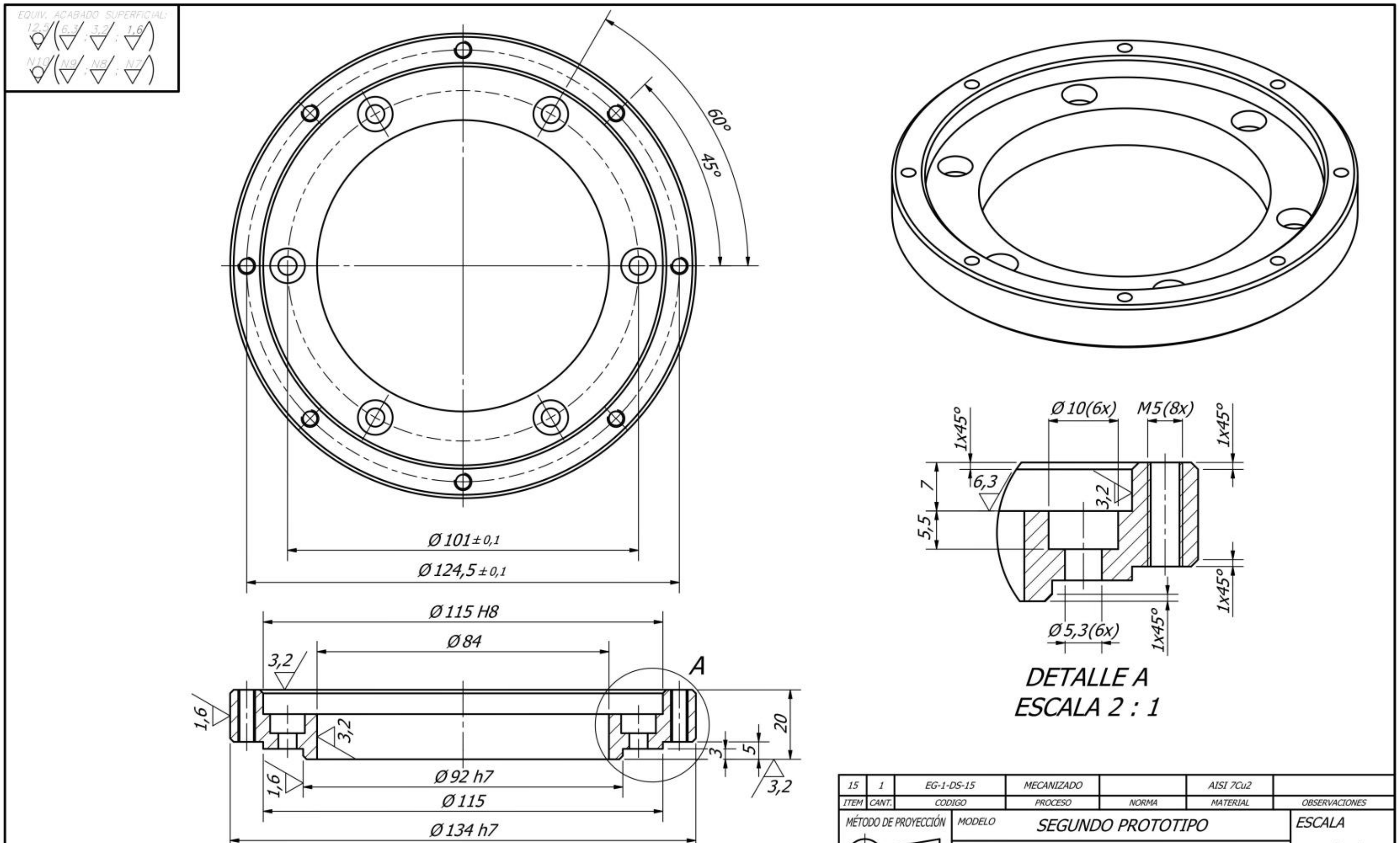
TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

COTA NOMINAL	COTA MAXIMA	COTA MINIMA
105H8	105,054	105,000

ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
12	1	EG-1-DS-12	MECANIZADO		AISI 7Cu2	

MÉTODO DE PROYECCIÓN	MODELO: SEGUNDO PROTOTIPO	ESCALA 1:1
	NOMBRE DE PIEZA: ANILLO DE APOYO 1	
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ	MÁQUINA: BRAZO ROBÓTICO 5GDL	FECHA: 2013.10.10
DISEÑADO: UTILI, L — FECHA: 2005.11.27	DIBUJADO: GARCÉS, A. — FECHA: 2013.10.10; REVISADO: BARRIGA, B. — FECHA: 2013.10.15; APROBADO: BARRIGA, B. — FECHA: 2013.10.15	LAMINA: L9-A3


TOLERANCIAS DIMENSIONALES
SEGUN DIN 7168
GRADO DE EXACTITUD
MEDIO
Más de 0,5 hasta 3
Más de 3 hasta 6
Más de 6 hasta 30
Más de 30 hasta 120
Más de 120 hasta 400
±0,1
±0,1
±0,2
±0,3
±0,5
10
A
3,2
Ø159 h7
Ø135
Ø149±0,1
3,2
6,3
1,6
60°
159h7
159,000
158,960
COTA NOMINAL
COTA MAXIMA
COTA MINIMA
DETALLE A
ESCALA 2 : 1
5,5
1x45°
1x45°
Ø5,2(6x)
Ø10(6x)
14
1
EG-1-DS-14
MECANIZADO
AISI 7Cu2
ITEM
CANT.
CODIGO
PROCESO
NORMA
MATERIAL
OBSERVACIONES
MÉTODO DE PROYECCIÓN
MODELO
SEGUNDO PROTOTIPO
ESCALA
1:1
NOMBRE DE PIEZA
TAPA SUPERIOR 1
MÁQUINA
BRAZO ROBÓTICO 5GDL
FECHA:
2013.10.10
LAMINA:
L10-A3
DISEÑADO
UTILL, L
FECHA
2005.11.27
DIBUJADO
GARCÉS, A.
FECHA
2013.10.10
REVISADO
BARRIGA, B.
FECHA
2013.10.15
APROBADO
BARRIGA, B.
FECHA
2013.10.15

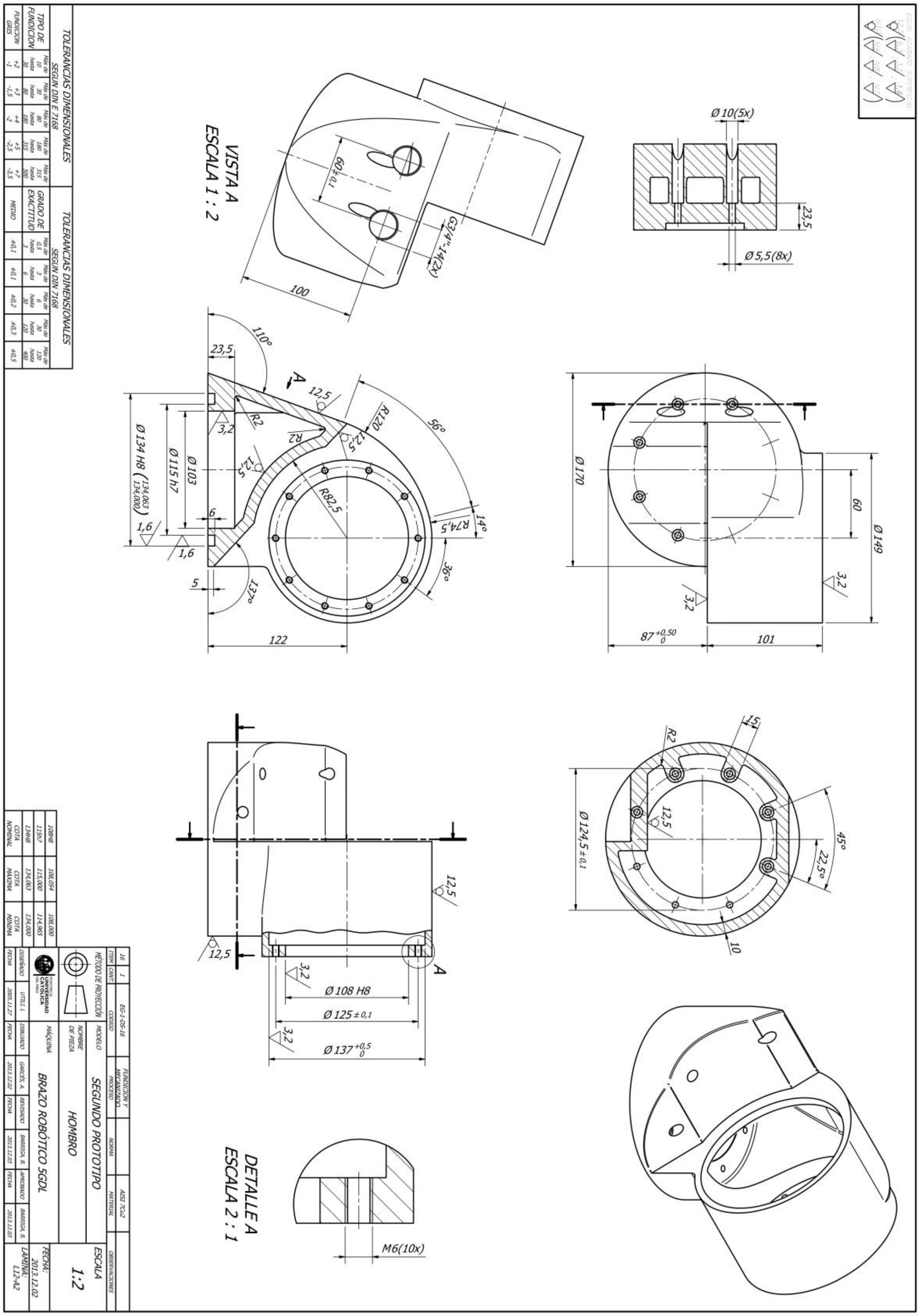


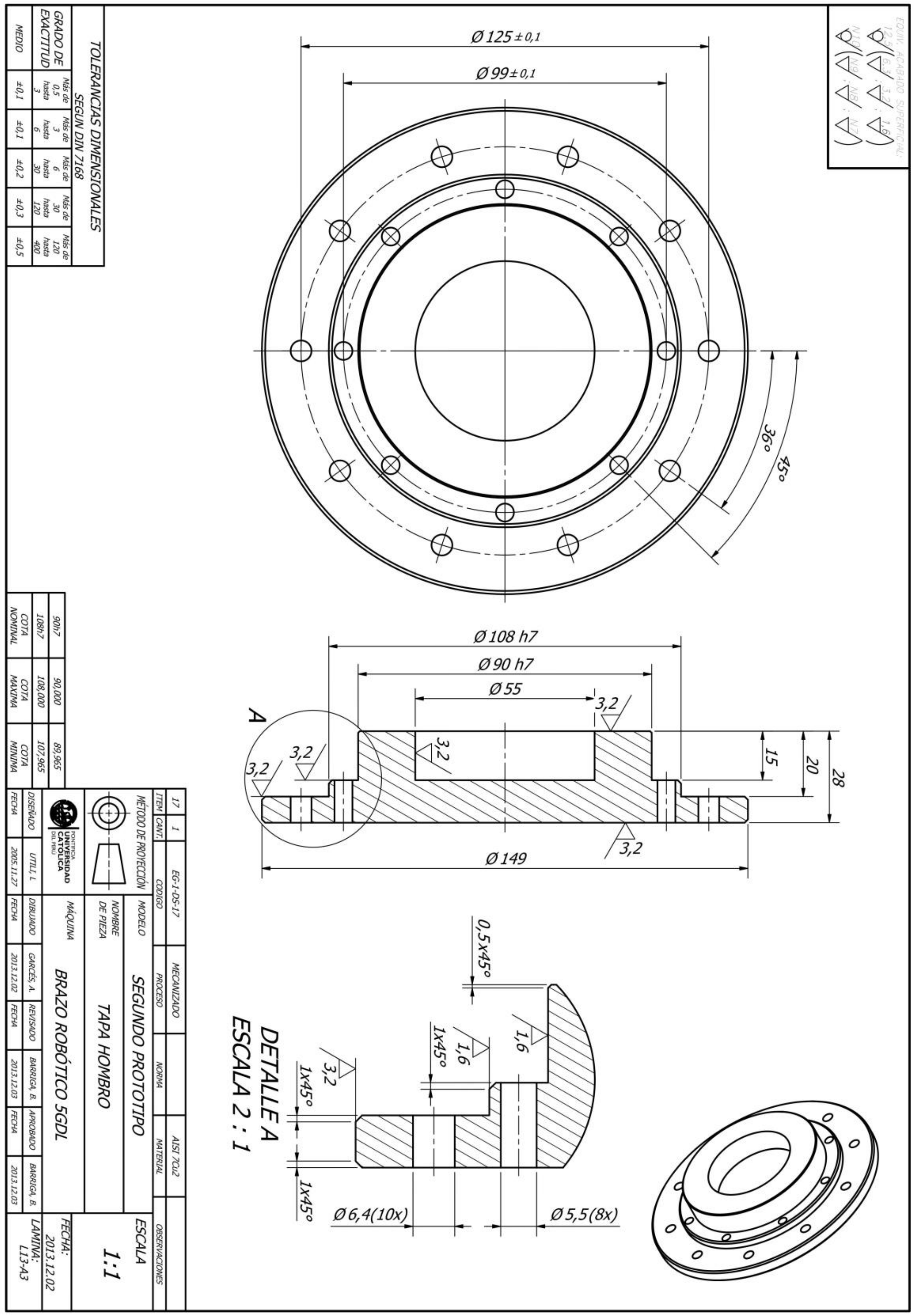
TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

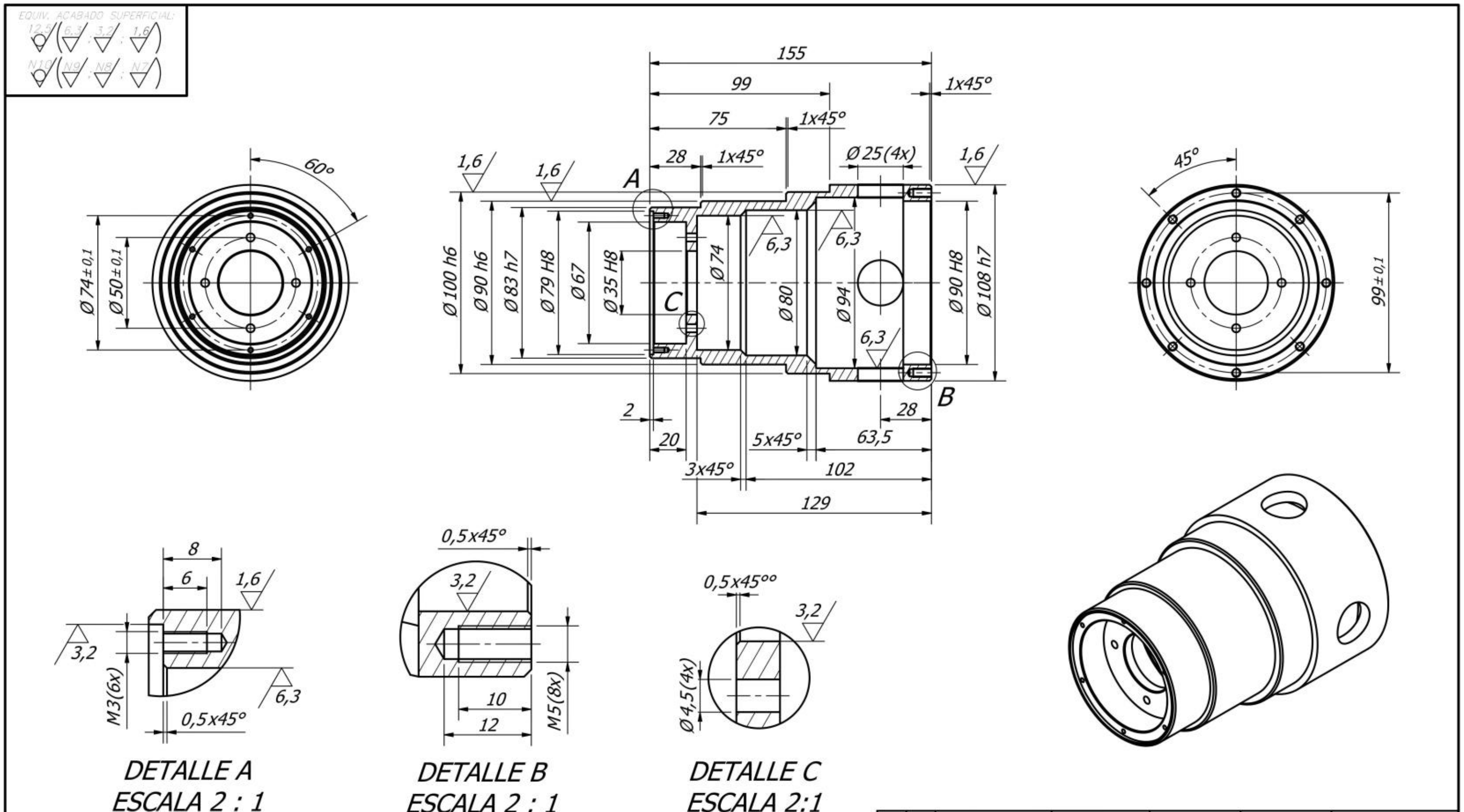
COTA NOMINAL	COTA MAXIMA	COTA MINIMA
92h7	92,000	91,965
115H8	115,054	115,000
134h7	134,000	133,960

ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
15	1	EG-1-DS-15	MECANIZADO		AISI 7Cu2	

MÉTODO DE PROYECCIÓN	MODELO: SEGUNDO PROTOTIPO	ESCALA 1:1
	NOMBRE DE PIEZA: BRIDA SUPERIOR 1	
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ	MÁQUINA: BRAZO ROBÓTICO 5GDL	FECHA: 2013.10.10
DISEÑADO: UTILI, L — FECHA: 2005.11.27	DIBUJADO: GARCÉS, A. — FECHA: 2013.10.10; REVISADO: BARRIGA, B. — FECHA: 2013.10.15; APROBADO: BARRIGA, B. — FECHA: 2013.10.15	LAMINA: L11-A3


VISTA A
ESCALA 1 : 2
DETALLE A
ESCALA 2 : 1
TOLERANCIAS DIMENSIONALES SEGÚN DIN 7168
SEGUNDO PROTOTIPO
HOMBRO
BRAZO ROBÓTICO 5GDL
ESCALA 1:2


TOLERANCIAS DIMENSIONALES SEGUN DIN 7168
Ø 125 ± 0,1
Ø 99 ± 0,1
36°
45°
Ø 108 h7
Ø 90 h7
Ø 55
Ø 149
A
15
20
28
3,2
DETALLE A
ESCALA 2 : 1
0,5x45°
1x45°
1,6
Ø 6,4(10x)
Ø 5,5(8x)
SEGUNDO PROTOTIPO
TAPA HOMBRO
BRAZO ROBÓTICO 5GDL
MECANIZADO
EG-1-DS-17
ESCALA
1:1
FECHA: 2013.12.02
LAMINA: L13-A3



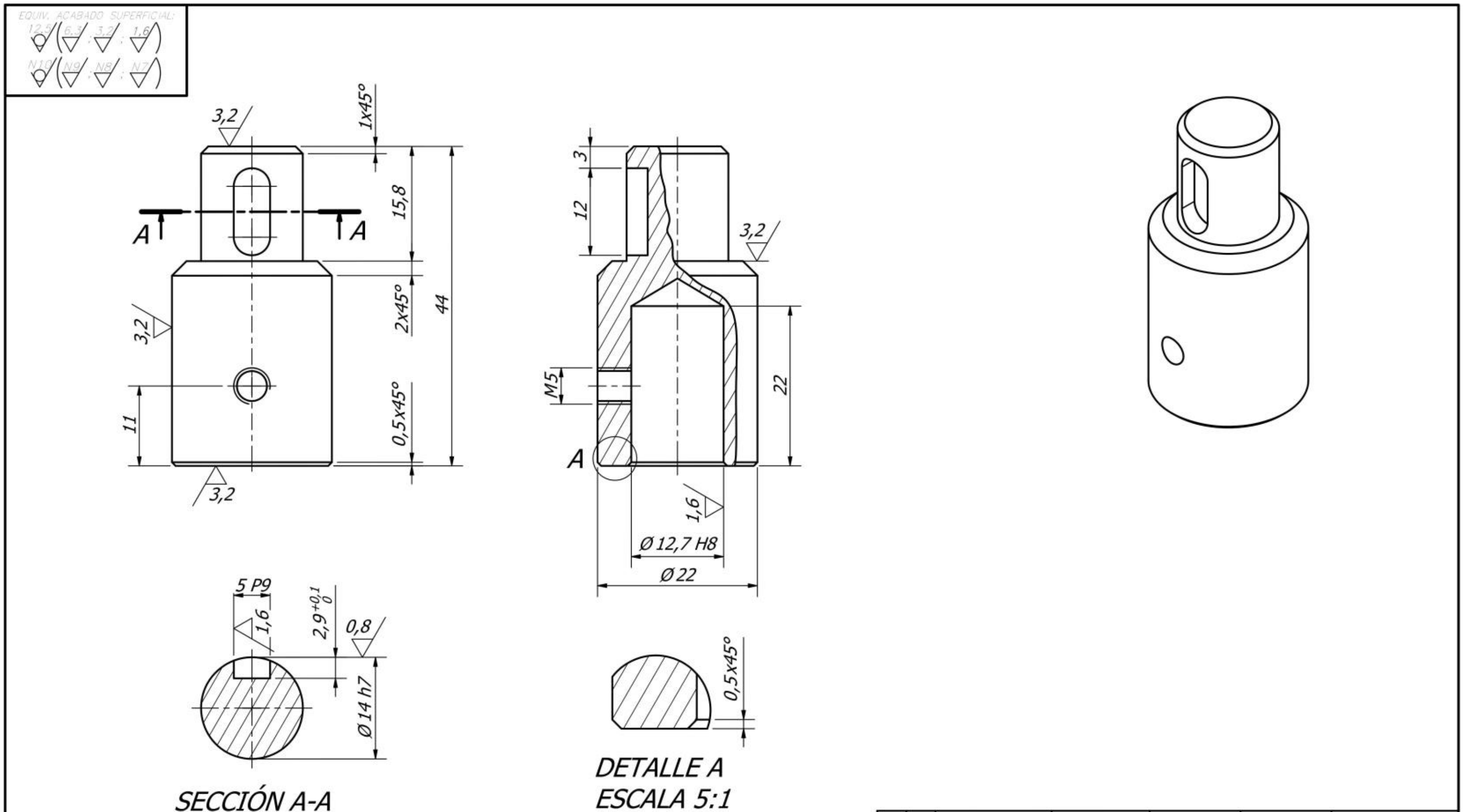
COTA NOMINAL	COTA MAXIMA	COTA MINIMA
35H8	35,039	35,000
79H8	79,046	79,000
83h7	83,000	82,965
90H8	90,054	90,000
90h6	90,000	89,978
100h6	100,000	99,978
108h7	108,000	107,965

TOLERANCIAS DIMENSIONALES SEGUN DIN 7168

GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
18	1	EG-1-DS-18	MECANIZADO		AISI 7Cu2	

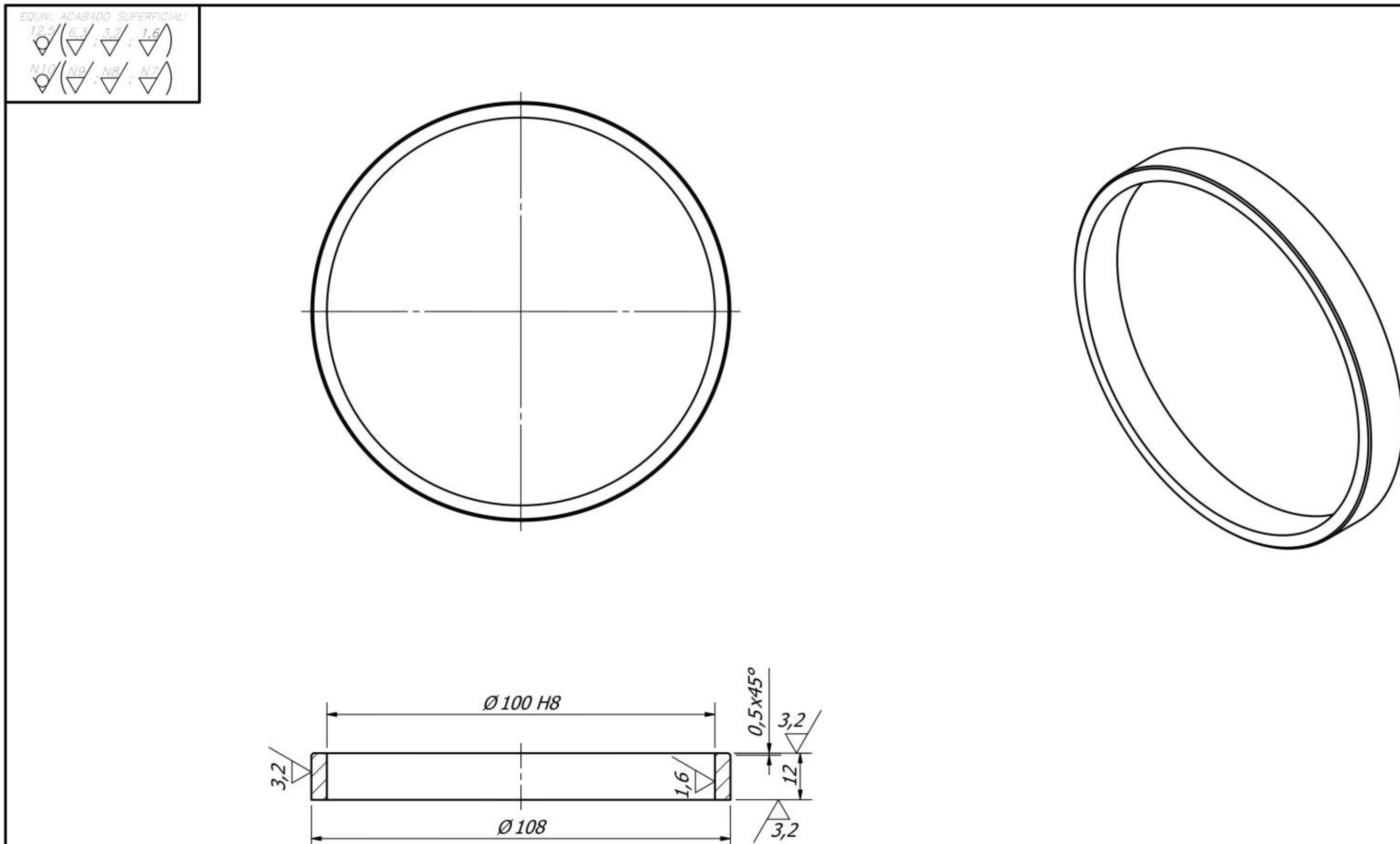
MÉTODO DE PROYECCIÓN	MODELO	SEGUNDO PROTOTIPO						ESCALA 1:2
	NOMBRE DE PIEZA	ALMA 2						
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ	MÁQUINA	BRAZO ROBÓTICO 5GDL						FECHA: 2013.12.03
DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO	BARRIGA, B.	LAMINA: L14-A3
FECHA	2005.11.27	FECHA	2013.12.02	FECHA	2013.12.03	FECHA	2013.12.03	



TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

5P9	4,988	4,958
12,7H8	12,727	12,700
14h7	14,000	13,982
COTA NOMINAL	COTA MAXIMA	COTA MINIMA

21	1	EG-1-DS-21	MECANIZADO		AISI 1045			
ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL		OBSERVACIONES	
MÉTODO DE PROYECCIÓN		MODELO	SEGUNDO PROTOTIPO				ESCALA	2:1
		NOMBRE DE PIEZA	ACOPLE 2					
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ		MÁQUINA	BRAZO ROBÓTICO 5GDL				FECHA: 2013.12.03	
DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO	BARRIGA, B.	LAMINA: L15-A3
FECHA	2005.11.27	FECHA	2013.12.02	FECHA	2013.12.03	FECHA	2013.12.03	



TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

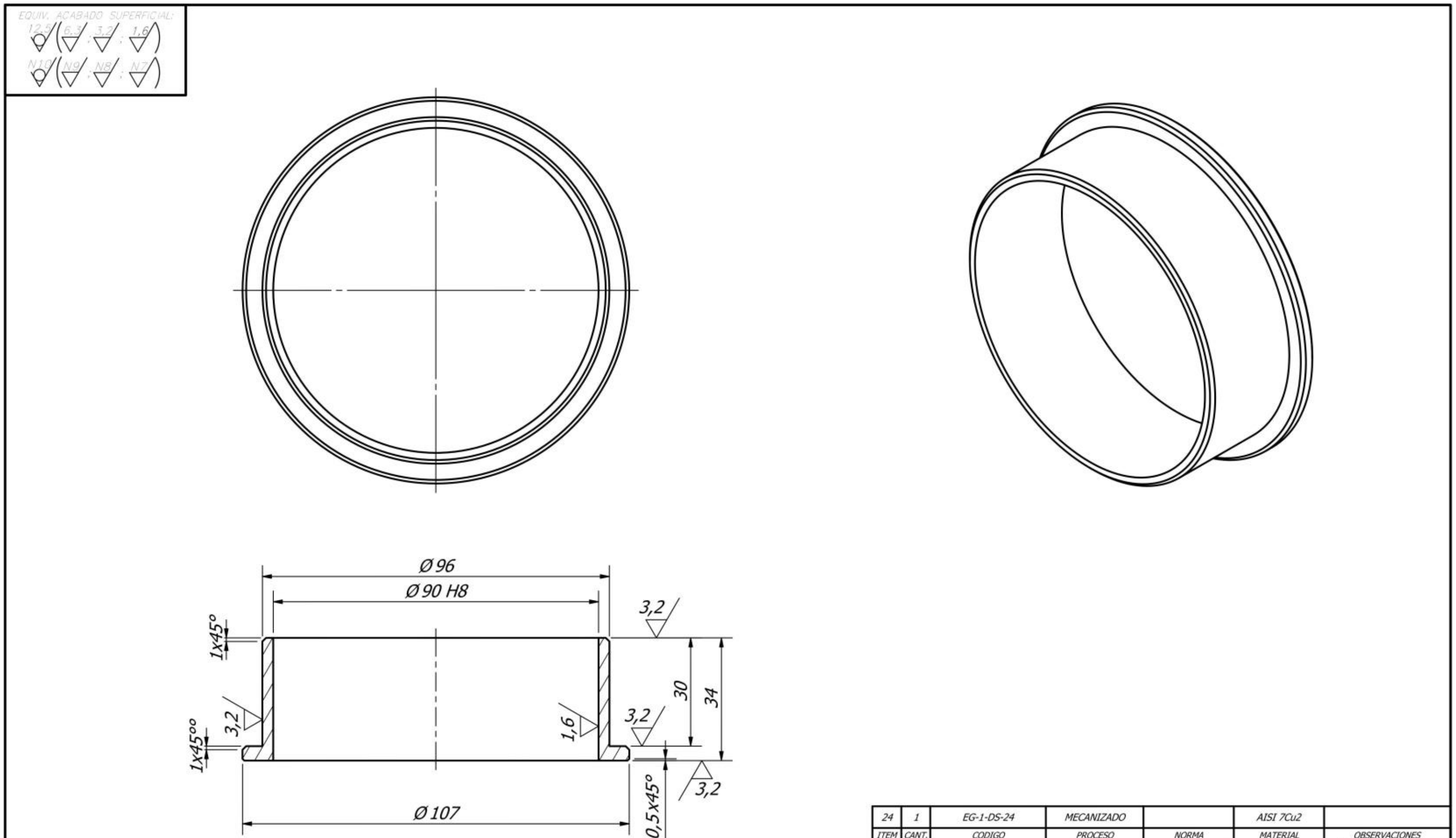
COTA NOMINAL	COTA MAXIMA	COTA MINIMA
100H8	100,054	100,000

ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
22	1	EG-1-DS-22	MECANIZADO		AISI 7Cu2	

MÉTODO DE PROYECCIÓN	
MODELO	SEGUNDO PROTOTIPO
NOMBRE DE PIEZA	ANILLO DE RODAMIENTO 2
MÁQUINA	BRAZO ROBÓTICO 5GDL
ESCALA	1:1
FECHA:	2013.10.10
LAMINA:	L16-A3

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO	BARRIGA, B.
FECHA	2005.11.27	FECHA	2013.12.02	FECHA	2013.12.03	FECHA	2013.12.03



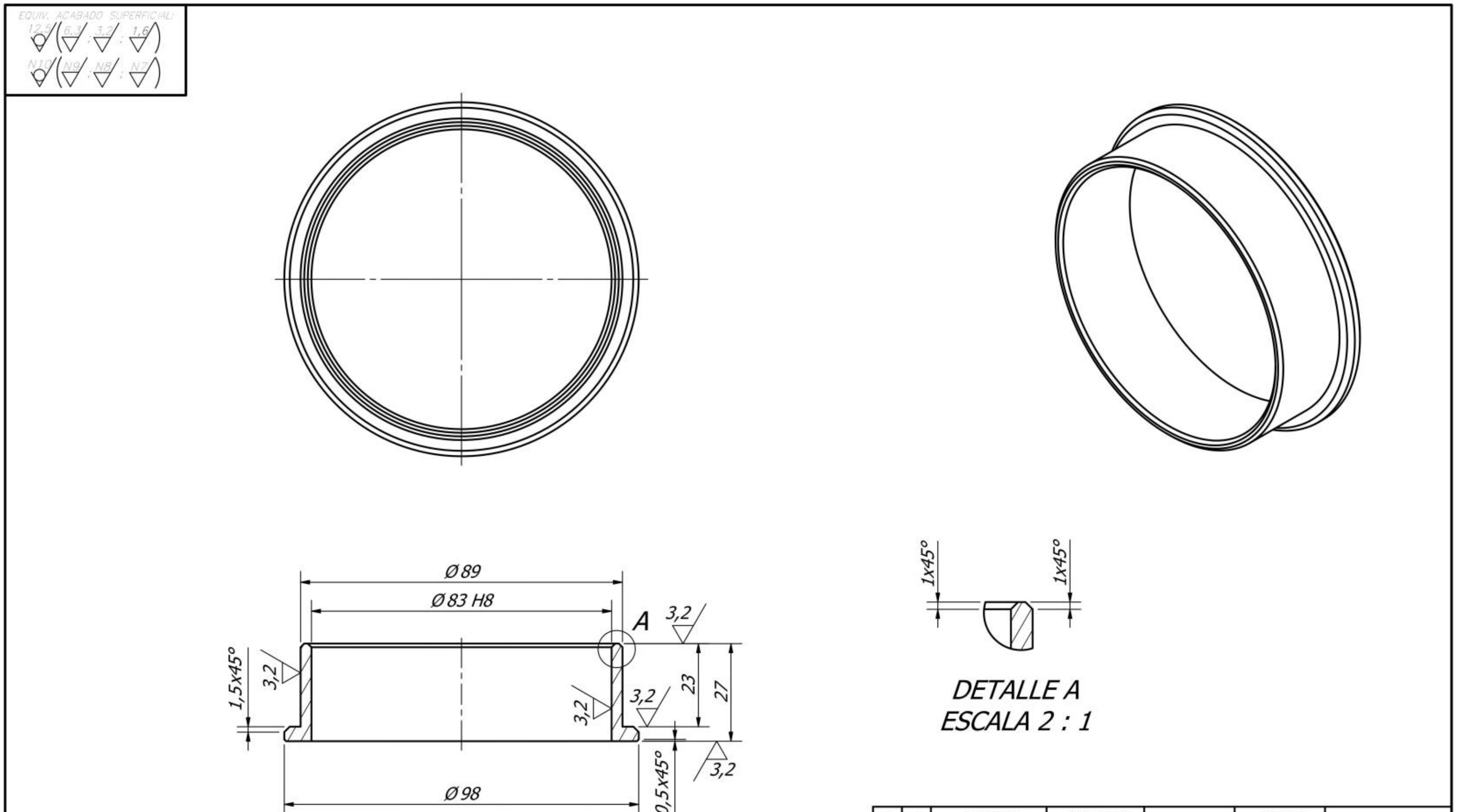
TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

COTA NOMINAL	COTA MAXIMA	COTA MINIMA
90H8	90,054	90,000

ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
24	1	EG-1-DS-24	MECANIZADO		AISI 7Cu2	

MÉTODO DE PROYECCIÓN	MODELO: SEGUNDO PROTOTIPO	ESCALA 1:1
	NOMBRE DE PIEZA: DISTANCIADOR INFERIOR 2	
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ	MÁQUINA: BRAZO ROBÓTICO 5GDL	FECHA: 2013.12.03

DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO	BARRIGA, B.	LAMINA: L17-A3
FECHA	2005.11.27	FECHA	2013.12.03	FECHA	2013.12.03	FECHA	2013.12.03	



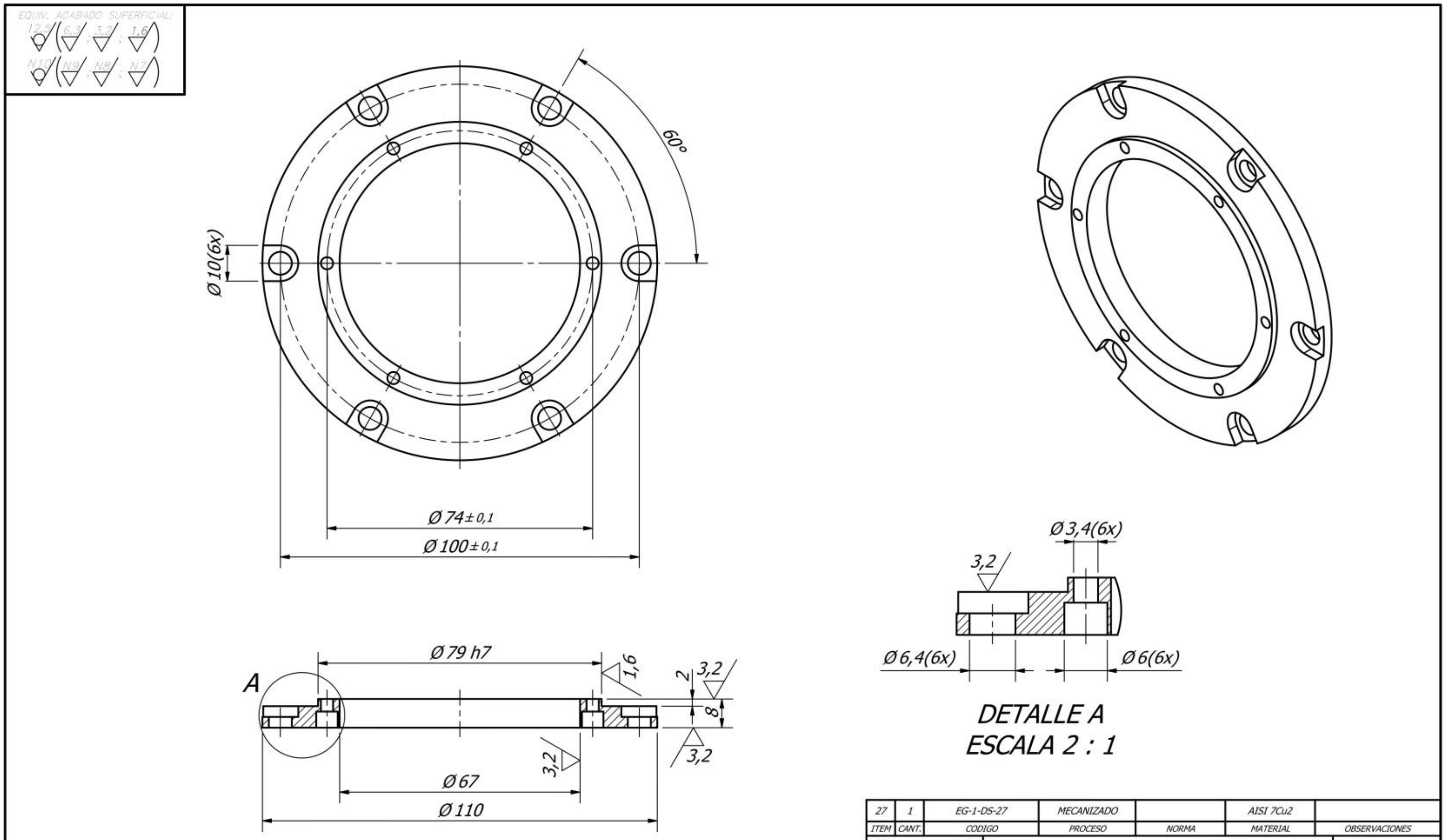
TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

COTA NOMINAL	COTA MAXIMA	COTA MINIMA
83H8	83,054	83,000

ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
26	1	EG-1-DS-26	MECANIZADO		AISI 7Cu2	

MÉTODO DE PROYECCIÓN	MODELO: SEGUNDO PROTOTIPO	ESCALA: 1:1
	NOMBRE DE PIEZA: DISTANCIADOR SUPERIOR 2	
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ	MÁQUINA: BRAZO ROBÓTICO 5GDL	FECHA: 2013.12.03

DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO	BARRIGA, B.	LAMINA: L18-A3
FECHA	2005.11.27	FECHA	2013.12.03	FECHA	2013.12.03	FECHA	2013.12.03	



TOLERANCIAS DIMENSIONALES SEGUN DIN 7168

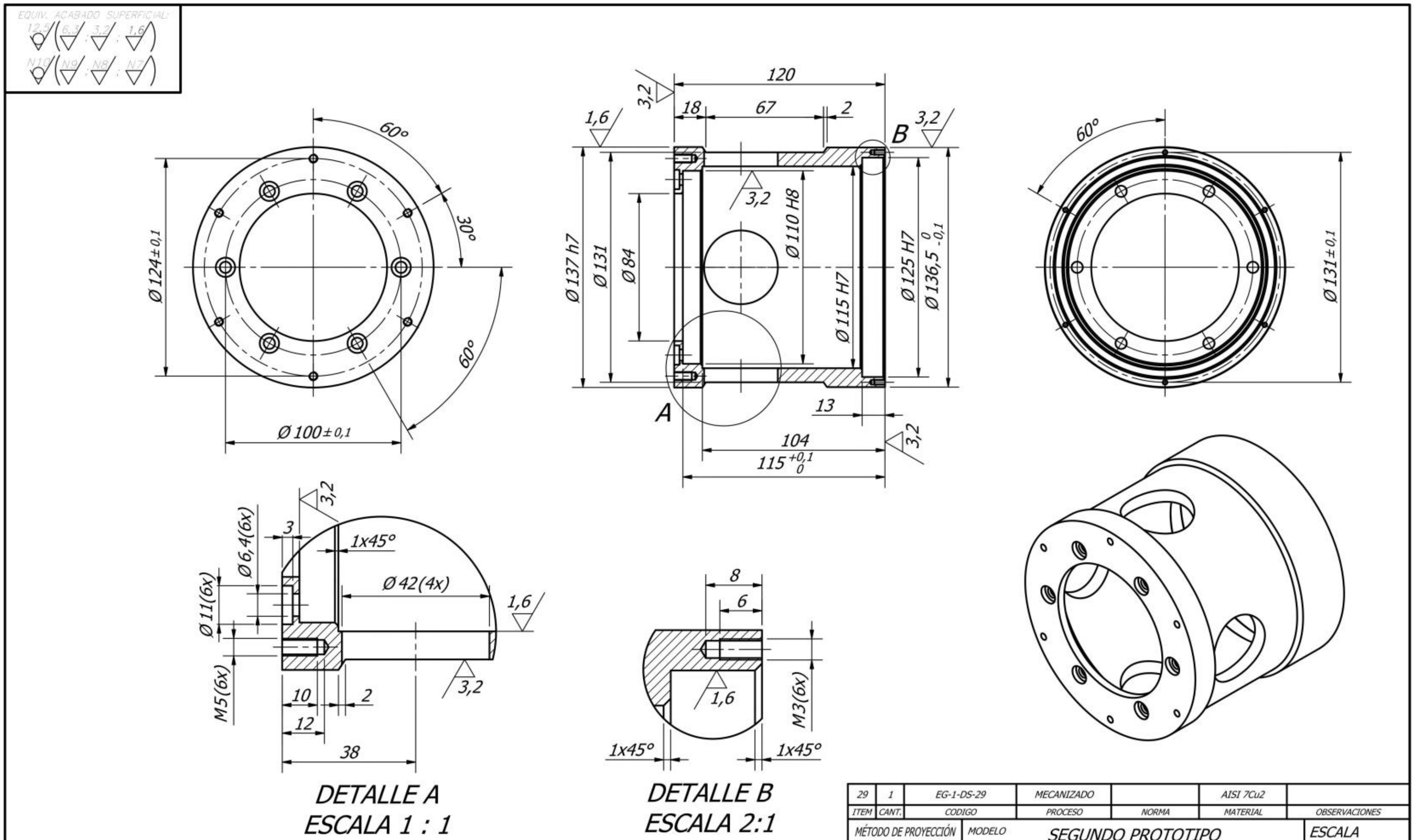
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

COTA NOMINAL	COTA MAXIMA	COTA MINIMA
79h7	79,000	78,970

ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
27	1	EG-1-DS-27	MECANIZADO		AISI 7Cu2	

MÉTODO DE PROYECCIÓN	MODELO: SEGUNDO PROTOTIPO	ESCALA 1:1
	NOMBRE DE PIEZA: BRIDA SUPERIOR 2	
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ	MÁQUINA: BRAZO ROBÓTICO 5GDL	FECHA: 2013.12.03
		LAMINA: L19-A3

DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO	BARRIGA, B.
FECHA	2005.11.27	FECHA	2013.12.02	FECHA	2013.12.02	FECHA	2013.12.03



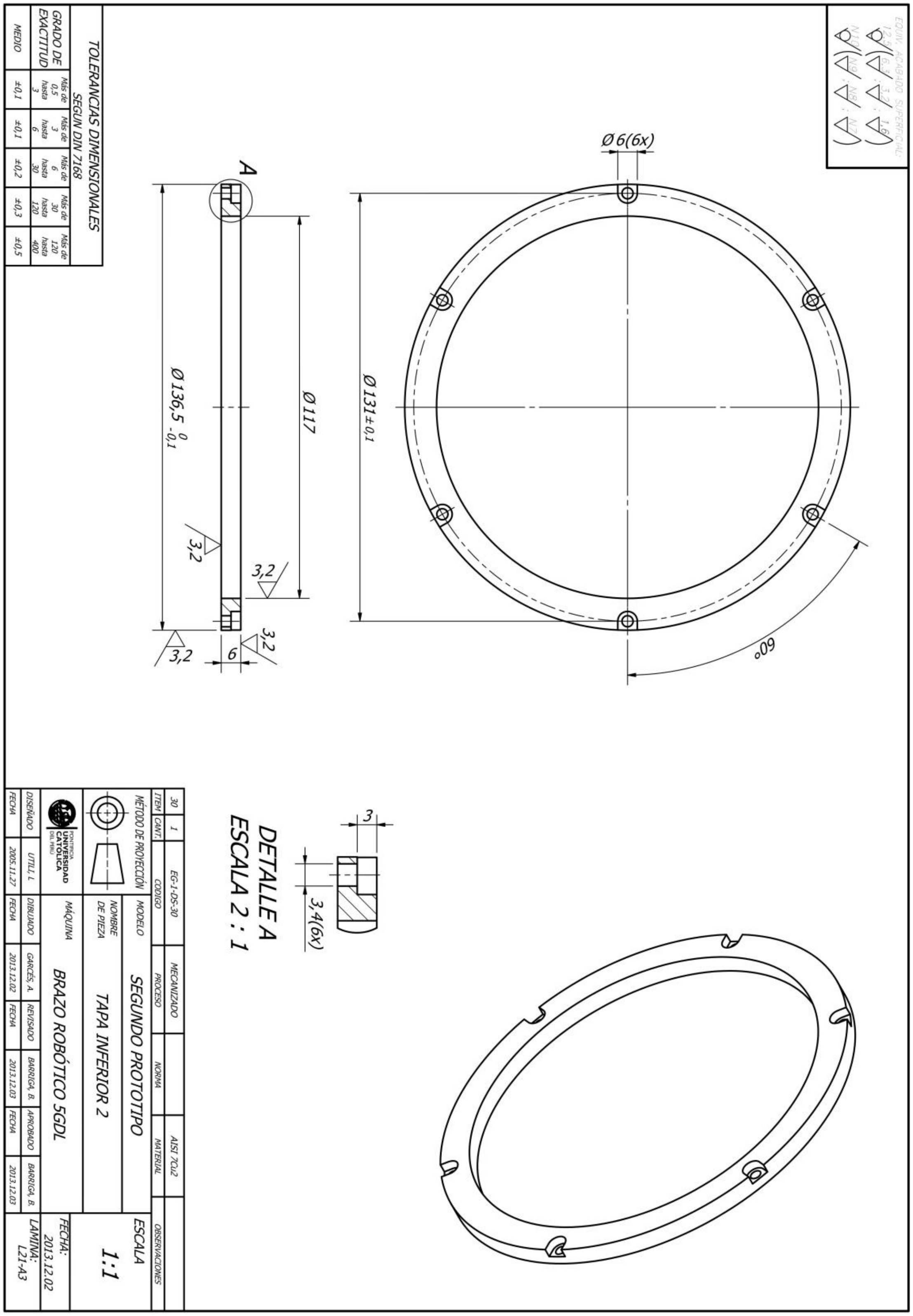
TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

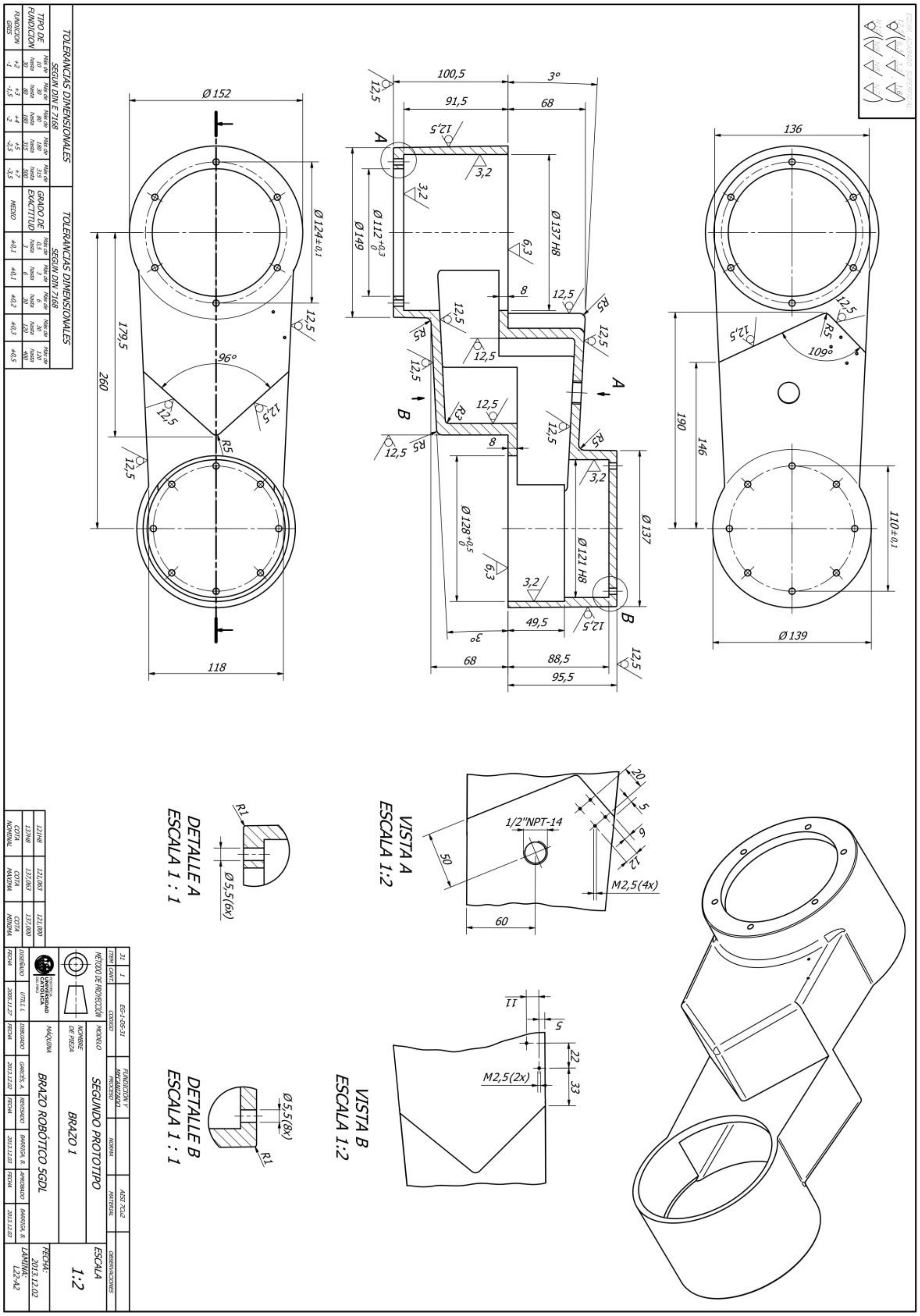
COTA NOMINAL	COTA MAXIMA	COTA MINIMA
110H8	110,054	110,000
115H7	115,035	115,000
125H7	125,040	125,000
137h7	137,000	136,960

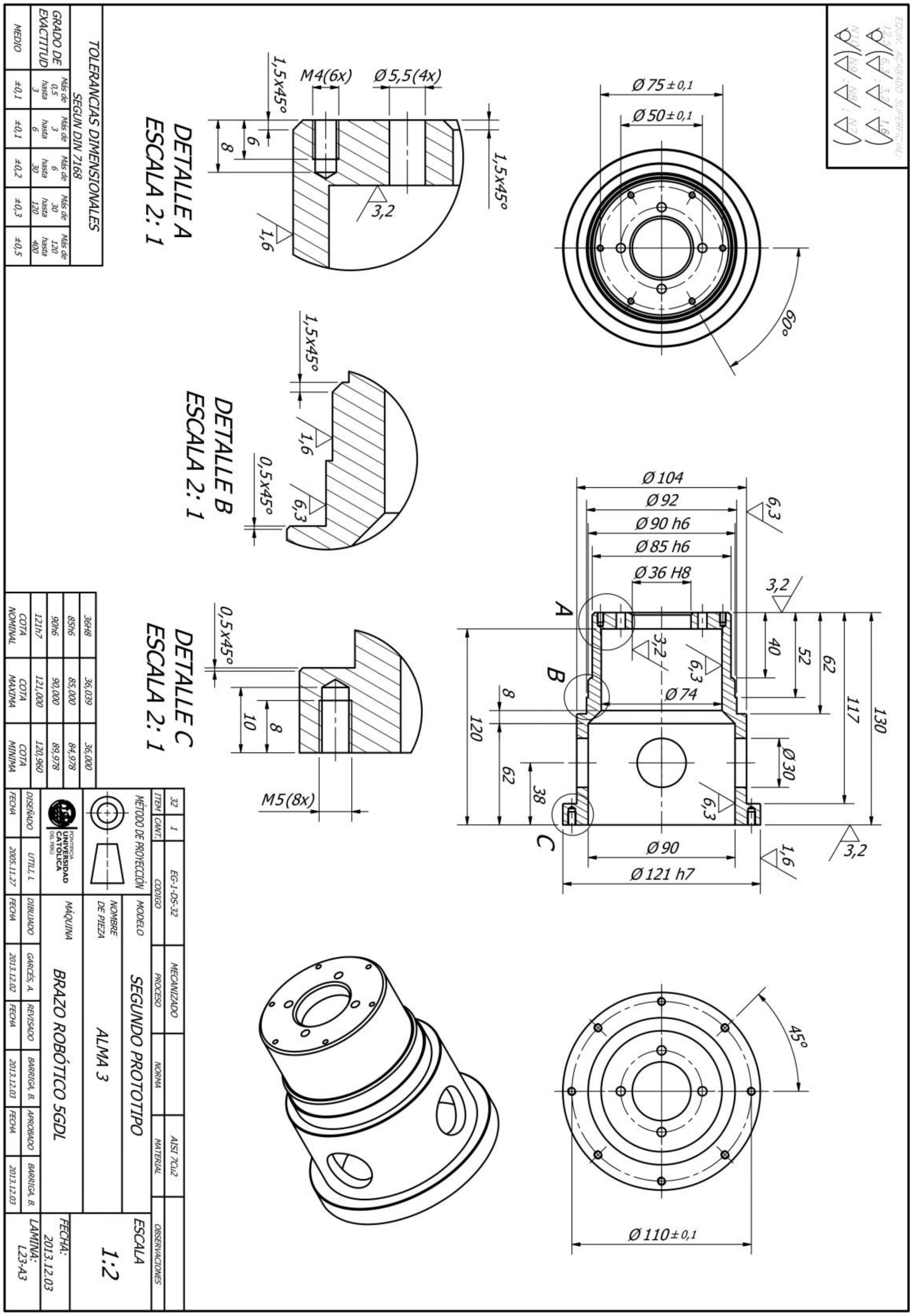
ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
29	1	EG-1-DS-29	MECANIZADO		AISI 7Cu2	

MÉTODO DE PROYECCIÓN	
MODELO	SEGUNDO PROTOTIPO
NOMBRE DE PIEZA	BASE 2
MÁQUINA	BRAZO ROBÓTICO 5GDL
ESCALA	1:2
FECHA:	2013.12.03
LAMINA:	L20-A3

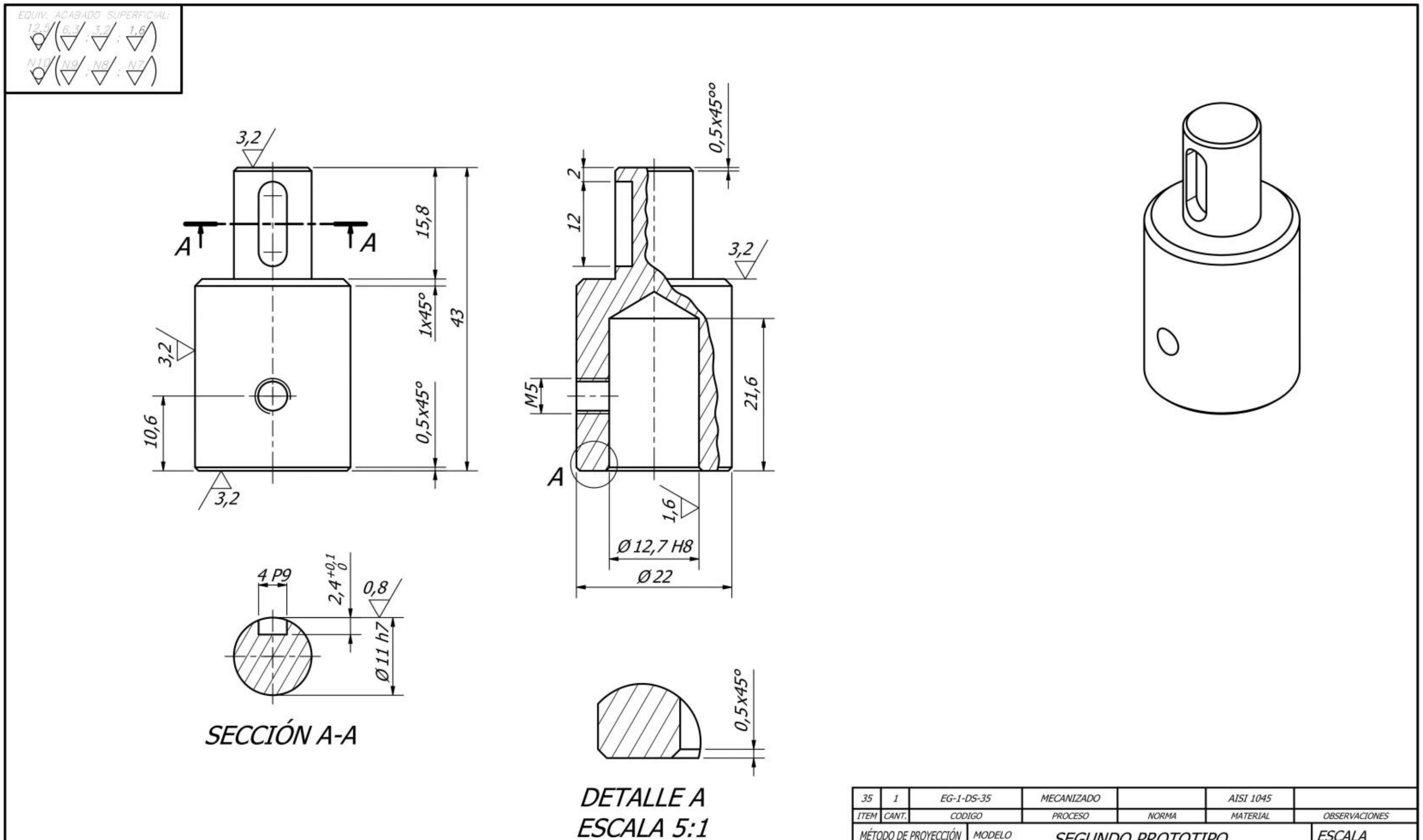
DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO	BARRIGA, B.
FECHA	2005.11.27	FECHA	2013.12.02	FECHA	2013.12.03	FECHA	2013.12.03


TOLERANCIAS DIMENSIONALES SEGÚN DIN 7168
GRADO DE EXACTITUD
MEDIO
±0,1
±0,1
±0,2
±0,3
±0,5
Ø 6(6x)
Ø 131±0,1
Ø 117
Ø 136,5
3,2
6
60°
A
DETALLE A
ESCALA 2 : 1
3
3,4(6x)
EG-1-DS-30
MECANIZADO
AISI 7Cu2
SEGUNDO PROTOTIPO
TAPA INFERIOR 2
BRAZO ROBÓTICO 5GDL
ESCALA
1:1
FECHA:
2013.12.02
LÁMINA:
L21-A3


TOLERANCIAS DIMENSIONALES SEGÚN DIN E 7168
TOLERANCIAS DIMENSIONALES SEGÚN DIN 7168
VISTA A
ESCALA 1:2
VISTA B
ESCALA 1:2
DETALLE A
ESCALA 1 : 1
DETALLE B
ESCALA 1 : 1
1/2"NPT-14
M2,5(4x)
M2,5(2x)
Ø5,5(6x)
Ø5,5(8x)
SEGUNDO PROTOTIPO
BRAZO 1
BRAZO ROBÓTICO 5GDL
ESCALA
1:2
FECHA:
2013.12.02
LÁMINA:
L2-A2



EQUIV. ACABADO SUPERFICIAL
Ø 75 ± 0,1
Ø 50 ± 0,1
60°
M4(6x)
Ø 5,5(4x)
1,5x45°
1,5x45°
8
6
3,2
1,6
DETALLE A
ESCALA 2: 1
TOLERANCIAS DIMENSIONALES SEGÚN DIN 7168
GRADO DE EXACTITUD
MEDIO
Más de 0,5 hasta 3 ±0,1
Más de 3 hasta 6 ±0,1
Más de 6 hasta 30 ±0,2
Más de 30 hasta 120 ±0,3
Más de 120 hasta 400 ±0,5
1,5x45°
1,6
0,5x45°
6,3
DETALLE B
ESCALA 2: 1
Ø 104
Ø 92
Ø 90 h6
Ø 85 h6
Ø 36 H8
6,3
3,2
A
B
C
3,2
6,3
Ø 74
40
52
62
117
130
120
8
62
38
Ø 30
6,3
Ø 90
Ø 121 h7
1,6
3,2
0,5x45°
DETALLE C
ESCALA 2: 1
10
8
M5(8x)
45°
Ø 110 ± 0,1
36H8 36,039 36,000
85h6 85,000 84,978
90h6 90,000 89,978
121h7 121,000 120,960
COTA NOMINAL
COTA MÁXIMA
COTA MÍNIMA
32 1 EG-1-DS-32 MECANIZADO AISI 7Cu2
ITEM CANT. CÓDIGO PROCESO NORMA MATERIAL OBSERVACIONES
MÉTODO DE PROYECCIÓN
MODELO SEGUNDO PROTOTIPO
NOMBRE DE PIEZA ALMA 3
MÁQUINA BRAZO ROBÓTICO 5GDL
ESCALA 1:2
FECHA: 2013.12.03
LAMINA: L23-A3
DISEÑADO UTILL. L
FECHA 2005.11.27
DIBUJADO GARCÉS, A.
FECHA 2013.12.02
REVISADO BARRIGA, B.
FECHA 2013.12.03
APROBADO BARRIGA, B.
FECHA 2013.12.03



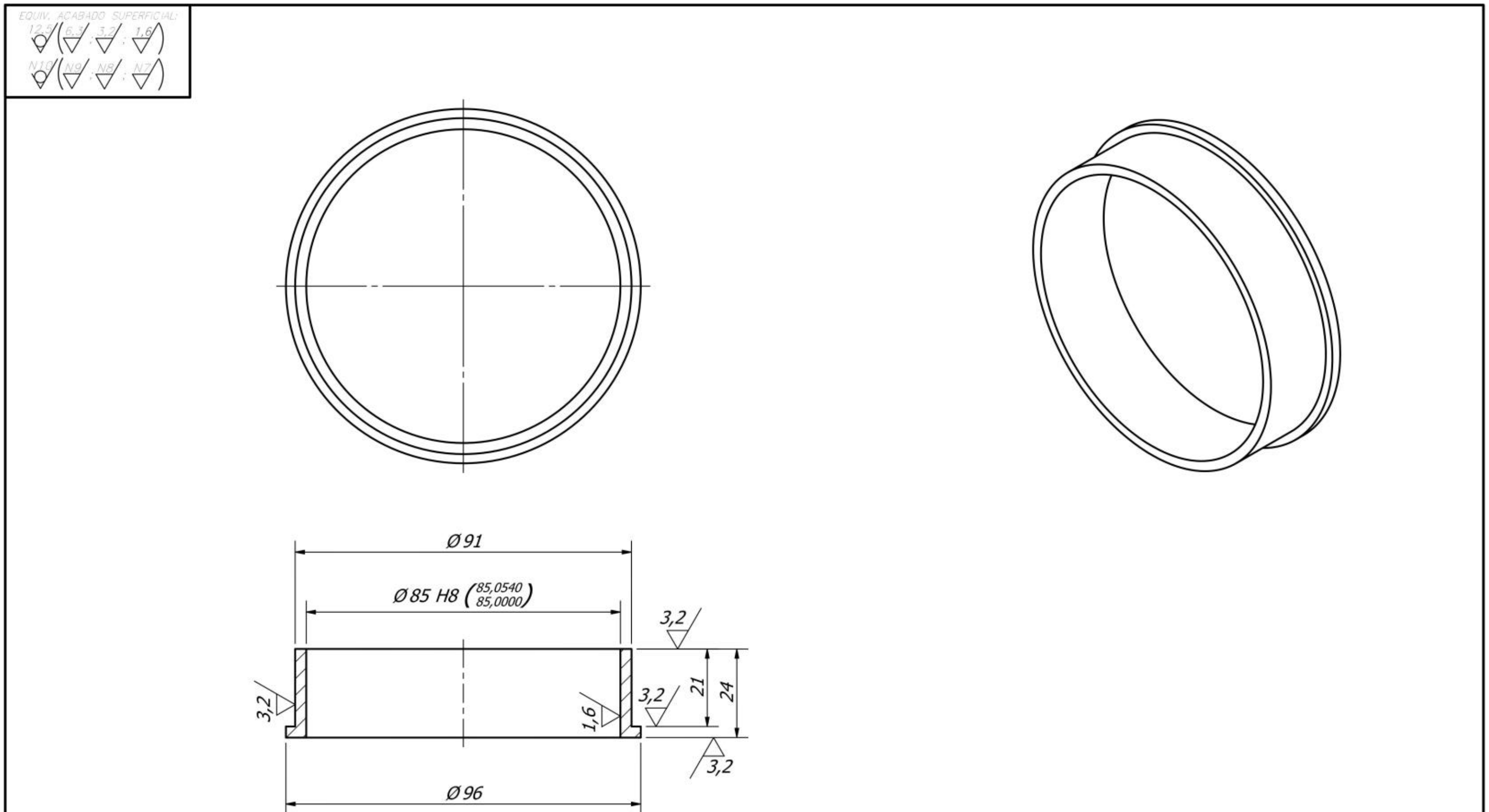
TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

COTA NOMINAL	COTA MAXIMA	COTA MINIMA
4P9	3,988	3,958
12,7H8	12,727	12,700
11h7	11,000	10,982

ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
35	1	EG-1-DS-35	MECANIZADO		AISI 1045	

MÉTODO DE PROYECCIÓN	MODELO	SEGUNDO PROTOTIPO	ESCALA 2:1
	NOMBRE DE PIEZA	ACOPLE 3	
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ	MÁQUINA	BRAZO ROBÓTICO 5GDL	FECHA: 2013.12.03
			LAMINA: L24-A3

DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO	BARRIGA, B.
FECHA	2005.11.27	FECHA	2013.12.02	FECHA	2013.12.03	FECHA	2013.12.03

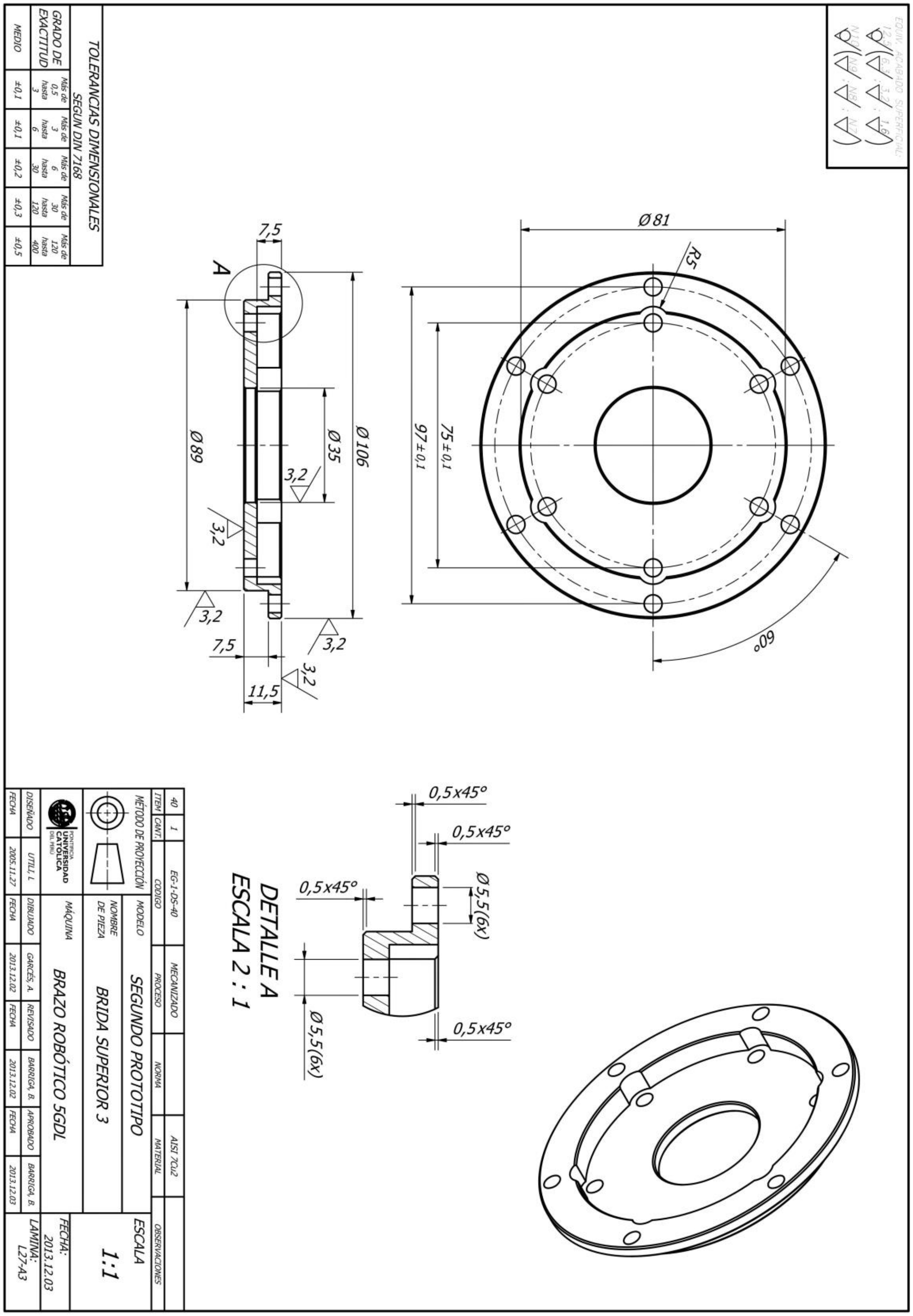


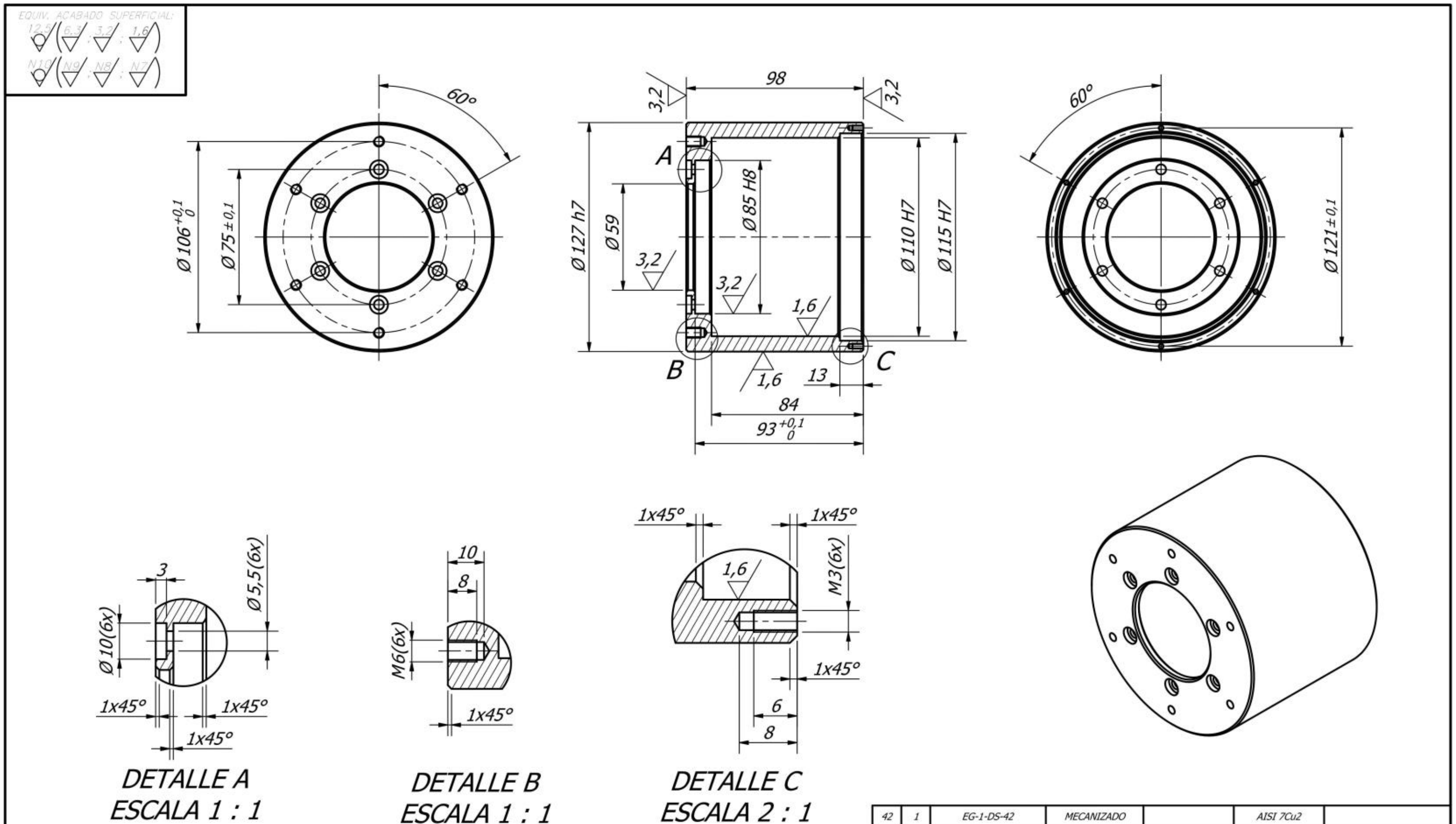
TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

COTA NOMINAL	COTA MAXIMA	COTA MINIMA
85H8	85,054	85,000

ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
37	1	EG-1-DS-37	MECANIZADO		AISI 7Cu2	

MÉTODO DE PROYECCIÓN		MODELO	SEGUNDO PROTOTIPO					ESCALA 1:1
		NOMBRE DE PIEZA	DISTANCIADOR 3					
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ		MÁQUINA	BRAZO ROBÓTICO 5GDL					FECHA: 2013.12.02
DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO	BARRIGA, B.	LAMINA: L25-A3
FECHA	2005.11.27	FECHA	2013.12.02	FECHA	2013.12.03	FECHA	2013.12.03	


TOLERANCIAS DIMENSIONALES SEGÚN DIN 7168
GRADO DE EXACTITUD
Más de 0,5 hasta 3
Más de 3 hasta 6
Más de 6 hasta 30
Más de 30 hasta 120
Más de 120 hasta 400
MEDIO ±0,1 ±0,1 ±0,2 ±0,3 ±0,5
EQUIV. ACABADO SUPERFICIAL
Ø81
R5
Ø106
Ø35
Ø89
75 ±0,1
97 ±0,1
7,5
11,5
3,2
A
60°
DETALLE A
ESCALA 2 : 1
0,5x45°
Ø5,5(6x)
40 1 EG-1-DS-40 MECANIZADO AISI 7CU2
ITEM CANT. CÓDIGO PROCESO NORMA MATERIAL OBSERVACIONES
MÉTODO DE PROYECCIÓN
MODELO SEGUNDO PROTOTIPO
NOMBRE DE PIEZA BRIDA SUPERIOR 3
MÁQUINA BRAZO ROBÓTICO 5GDL
ESCALA 1:1
DISEÑADO UTILL. L FECHA 2005.11.27
DIBUJADO GARCÉS, A. FECHA 2013.12.02
REVISADO BARRIGA, B. FECHA 2013.12.02
APROBADO BARRIGA, B. FECHA 2013.12.03
FECHA: 2013.12.03
LAMINA: L27-A3

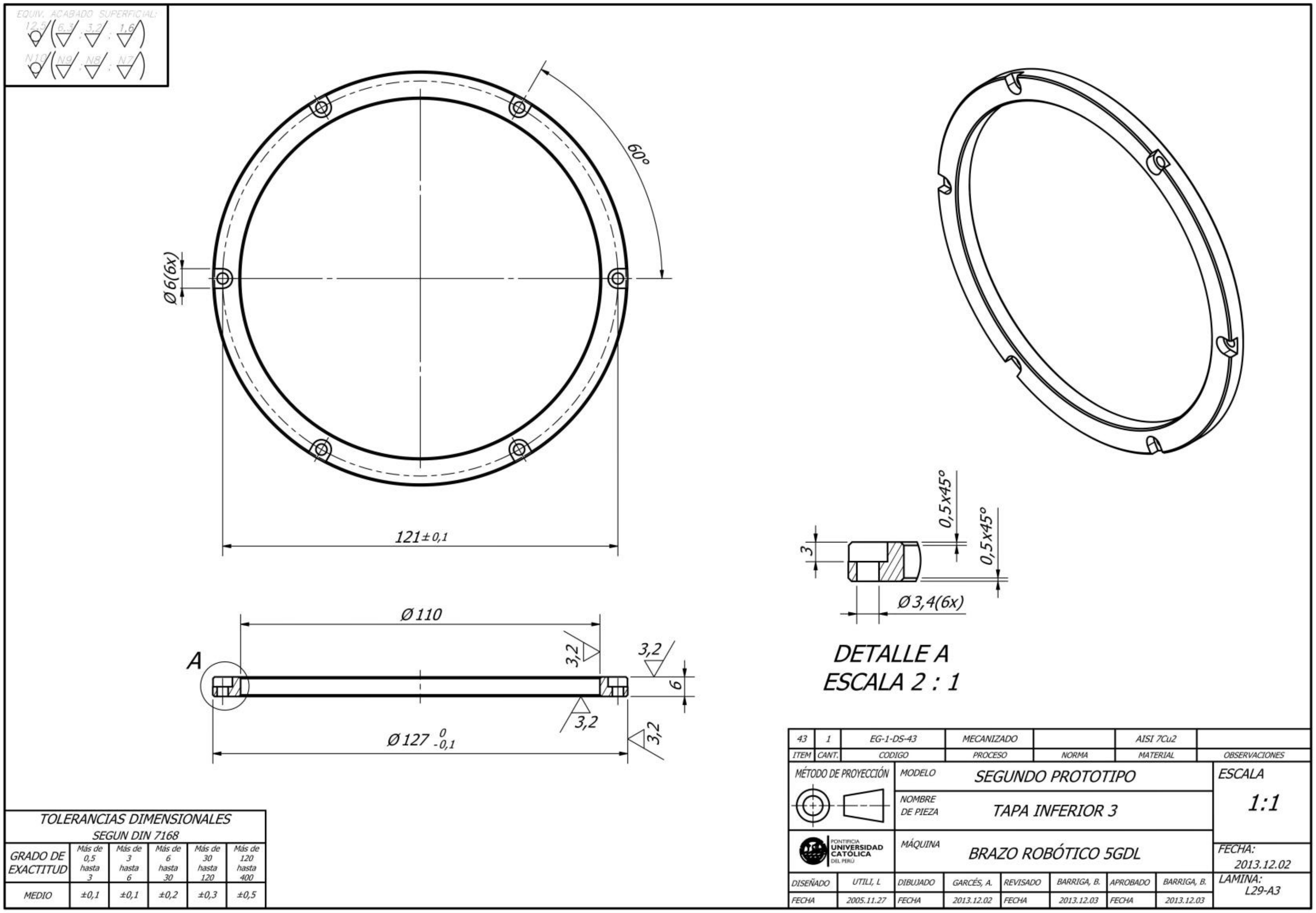


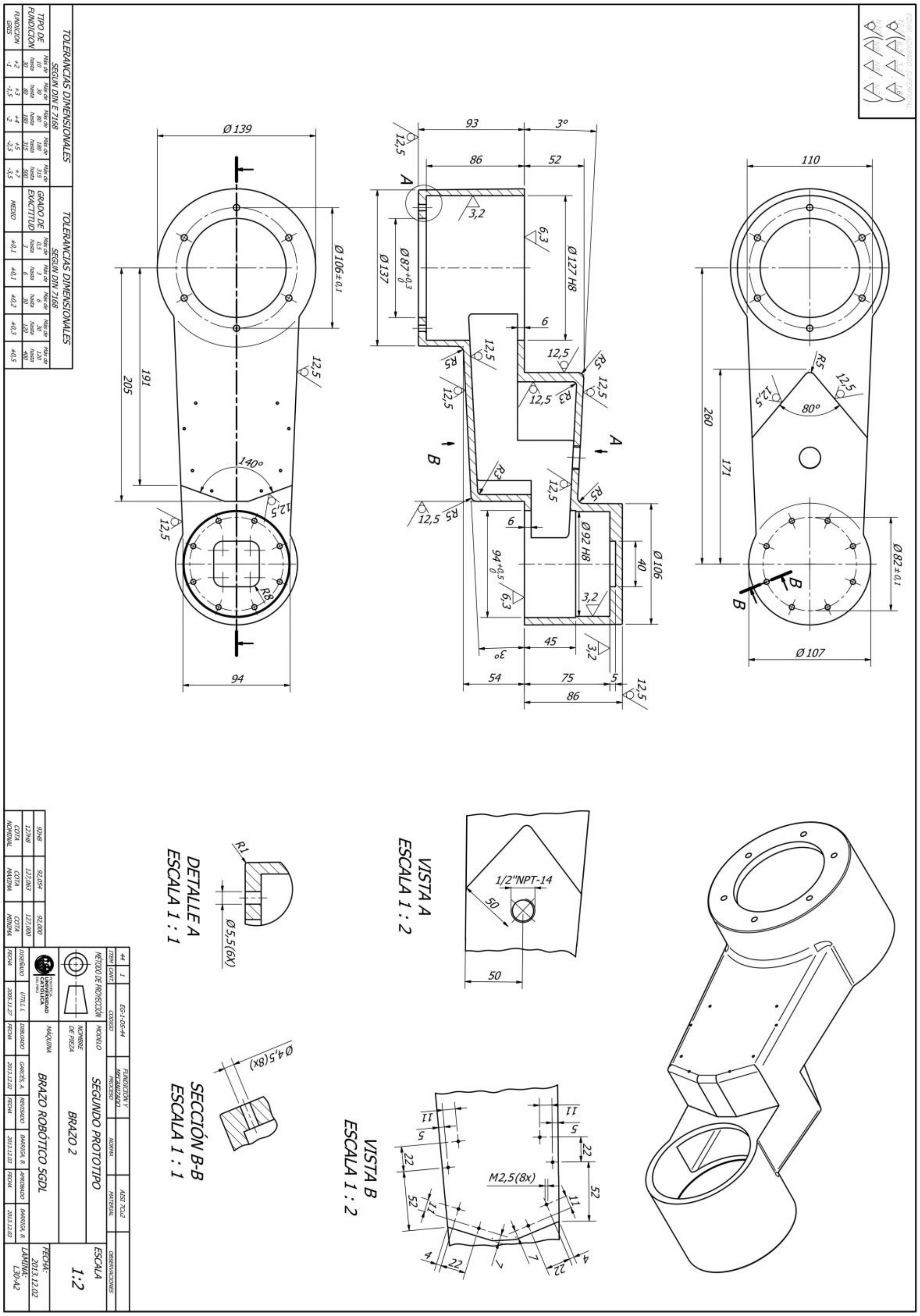
TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

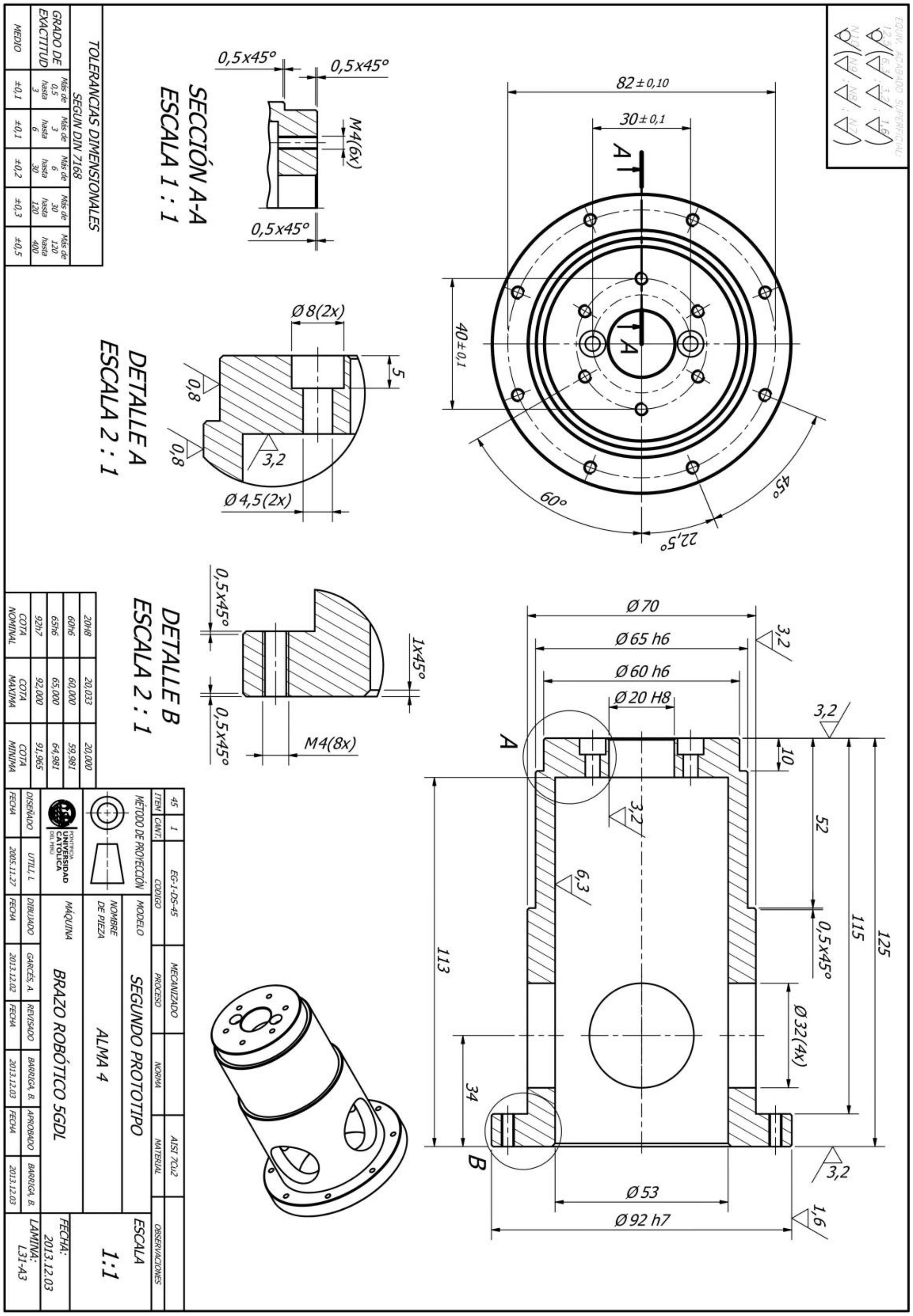
COTA NOMINAL	COTA MAXIMA	COTA MINIMA
85H8	85,054	85,000
110H7	110,035	110,000
115H7	115,035	115,000
127h7	127,000	126,960

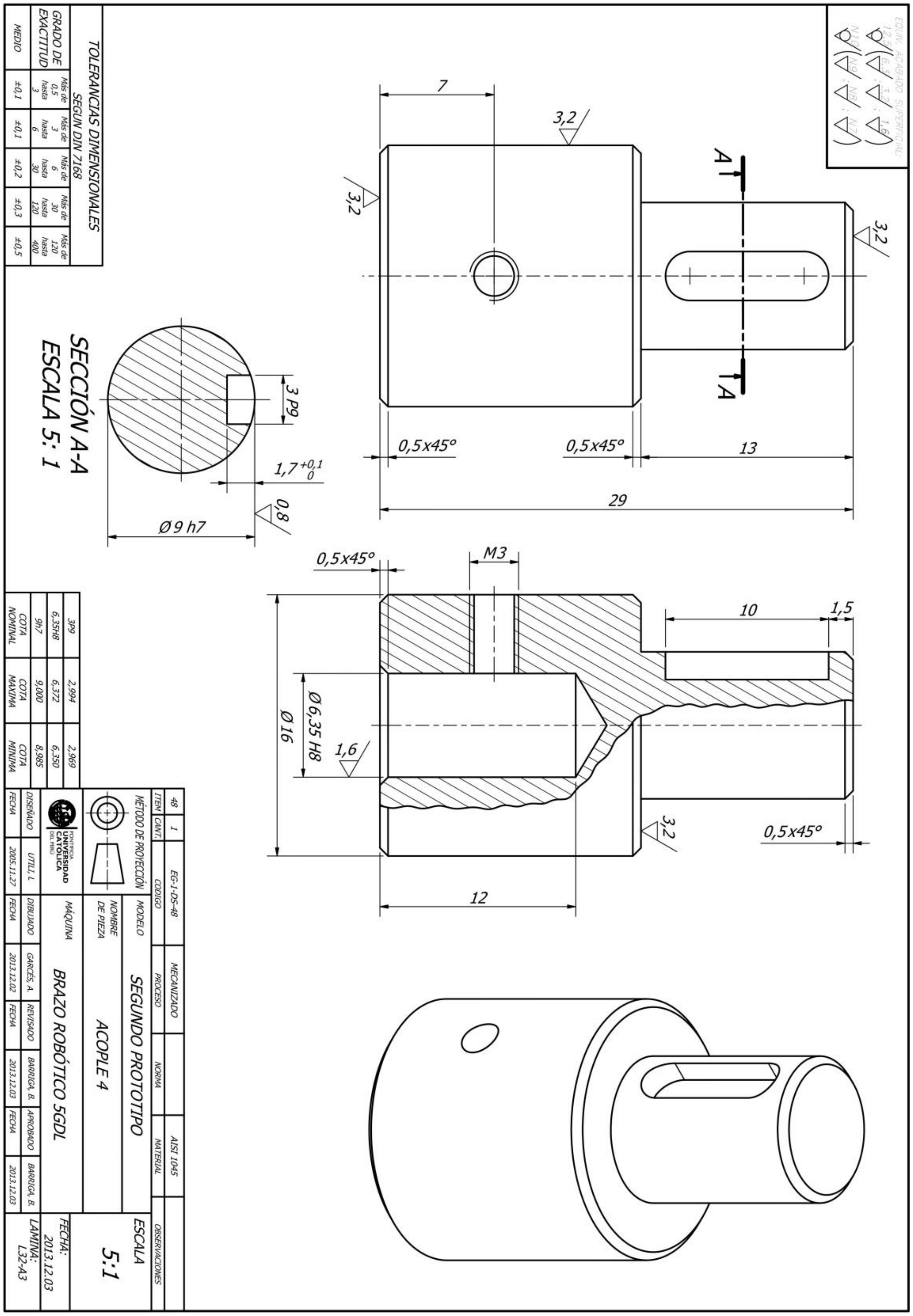
ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
42	1	EG-1-DS-42	MECANIZADO		AISI 7Cu2	

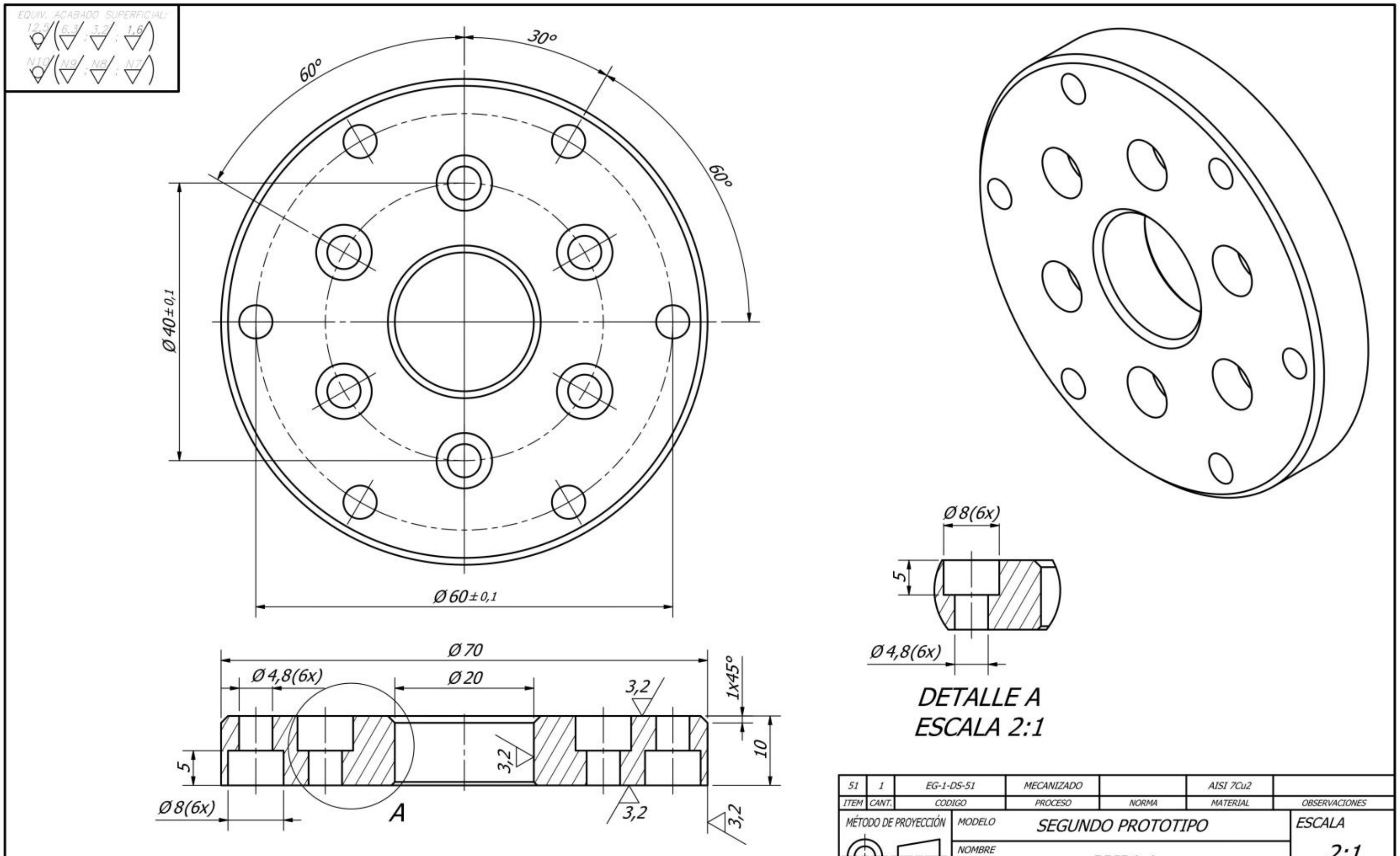
MÉTODO DE PROYECCIÓN		MODELO	SEGUNDO PROTOTIPO				ESCALA 1:2
		NOMBRE DE PIEZA	BASE 3				
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ		MÁQUINA	BRAZO ROBÓTICO 5GDL				FECHA: 2013.12.03
DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO BARRIGA, B.	LAMINA: L28-A3
FECHA	2005.11.27	FECHA	2013.12.02	FECHA	2013.12.03	FECHA 2013.12.03	


EQUIV. ACABADO SUPERFICIAL:
12,5 (6,3 3,2 1,6)
N10 (N9 N8 N7)
60°
Ø 6(6x)
121 ± 0,1
Ø 110
A
3,2
3,2
3,2
3,2
6
Ø 127 0 -0,1
0,5x45°
0,5x45°
3
Ø 3,4(6x)
DETALLE A
ESCALA 2 : 1
TOLERANCIAS DIMENSIONALES
SEGUN DIN 7168
GRADO DE EXACTITUD
Más de 0,5 hasta 3
Más de 3 hasta 6
Más de 6 hasta 30
Más de 30 hasta 120
Más de 120 hasta 400
MEDIO
±0,1
±0,1
±0,2
±0,3
±0,5
43
1
EG-1-DS-43
MECANIZADO
AISI 7Cu2
ITEM
CANT.
CODIGO
PROCESO
NORMA
MATERIAL
OBSERVACIONES
MÉTODO DE PROYECCIÓN
MODELO
SEGUNDO PROTOTIPO
ESCALA
1:1
NOMBRE DE PIEZA
TAPA INFERIOR 3
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
MÁQUINA
BRAZO ROBÓTICO 5GDL
FECHA:
2013.12.02
DISEÑADO
UTILI, L
DIBUJADO
GARCÉS, A.
REVISADO
BARRIGA, B.
APROBADO
BARRIGA, B.
LAMINA:
L29-A3
FECHA
2005.11.27
FECHA
2013.12.02
FECHA
2013.12.03
FECHA
2013.12.03


TOLERANCIAS DIMENSIONALES SEGÚN DIN E 7168
TOLERANCIAS DIMENSIONALES SEGÚN DIN 7168
VISTA A ESCALA 1 : 2
DETALLE A ESCALA 1 : 1
SECCIÓN B-B ESCALA 1 : 1
VISTA B ESCALA 1 : 2
SEGUNDO PROTOTIPO
BRAZO 2
BRAZO ROBÓTICO 5GDL
ESCALA 1:2
AISI 7Ca2
EG-1-05-44
L30-A2


SECCIÓN A-A
ESCALA 1 : 1
DETALLE A
ESCALA 2 : 1
DETALLE B
ESCALA 2 : 1
TOLERANCIAS DIMENSIONALES
SEGÚN DIN 7168
SEGUNDO PROTOTIPO
ALMA 4
BRAZO ROBÓTICO 5GDL
MECANIZADO
EG-1-DS-45
ESCALA
1:1
FECHA:
2013.12.03
LAMINA:
L31-A3

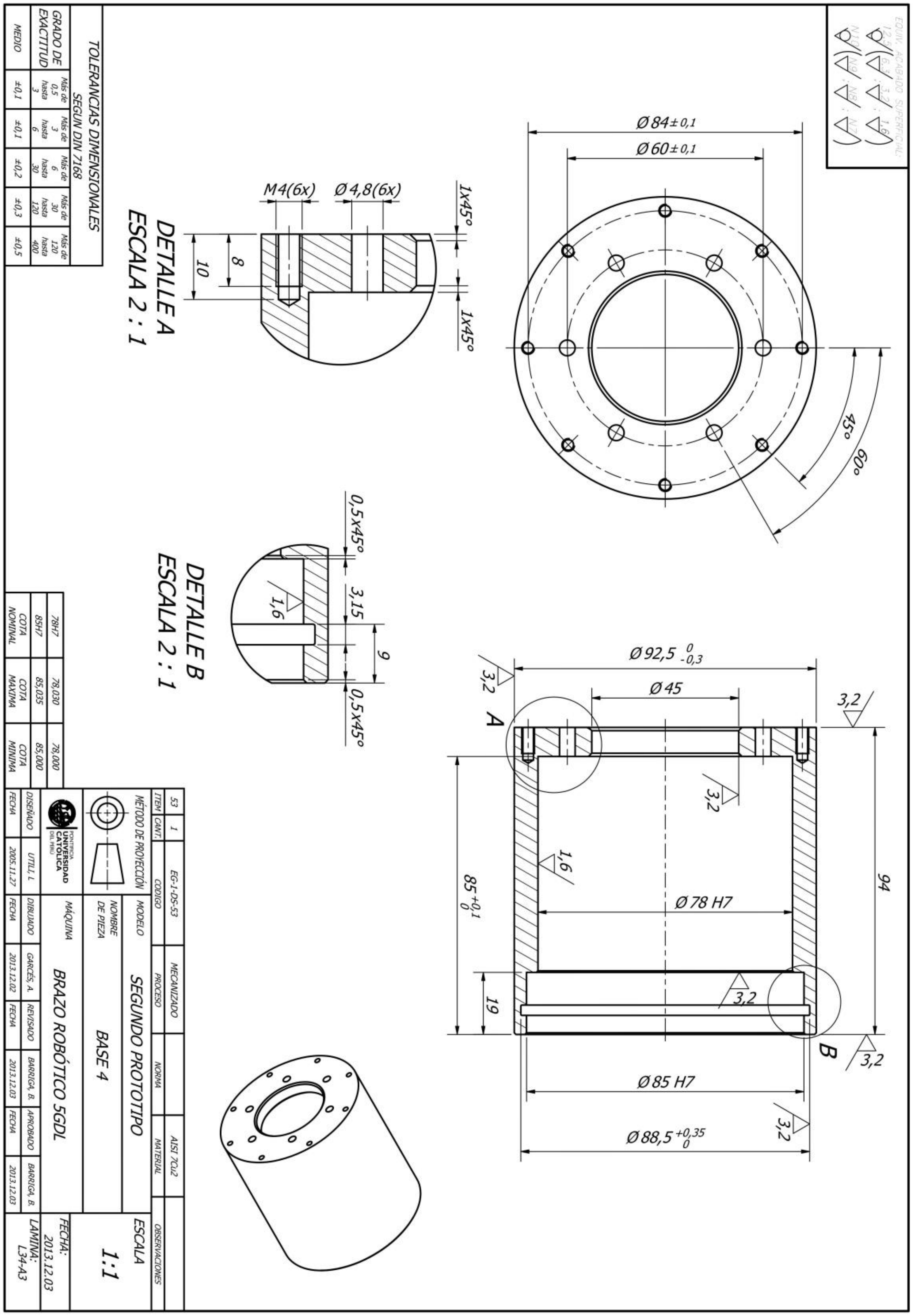

TOLERANCIAS DIMENSIONALES SEGÚN DIN 7168
GRADO DE EXACTITUD
MEDIO
Más de 0,5 hasta 3
Más de 3 hasta 6
Más de 6 hasta 30
Más de 30 hasta 120
Más de 120 hasta 400
±0,1
±0,1
±0,2
±0,3
±0,5
SECCIÓN A-A
ESCALA 5: 1
3 P9
1,7 +0,1 0
0,8
Ø 9 h7
7
3,2
A
0,5x45°
13
29
M3
10
1,5
Ø 6,35 H8
Ø 16
1,6
12
COTA NOMINAL
COTA MÁXIMA
COTA MÍNIMA
3P9
2,994
2,969
6,35H8
6,372
6,350
9h7
9,000
8,985
48
1
EG-1-D5-48
MECANIZADO
AISI 1045
ITEM
CANT.
CÓDIGO
PROCESO
NORMA
MATERIAL
OBSERVACIONES
MÉTODO DE PROYECCIÓN
MODELO
SEGUNDO PROTOTIPO
NOMBRE DE PIEZA
ACOPLE 4
MÁQUINA
BRAZO ROBÓTICO 5GDL
ESCALA
5:1
DISEÑADO
UTILL. L
FECHA
2005.11.27
DIBUJADO
GARCÉS, A.
2013.12.02
REVISADO
BARRIGA, B.
2013.12.03
APROBADO
BARRIGA, B.
2013.12.03
FECHA:
2013.12.03
LÁMINA:
L32-A3
EQUIV. ACABADO SUPERFICIAL

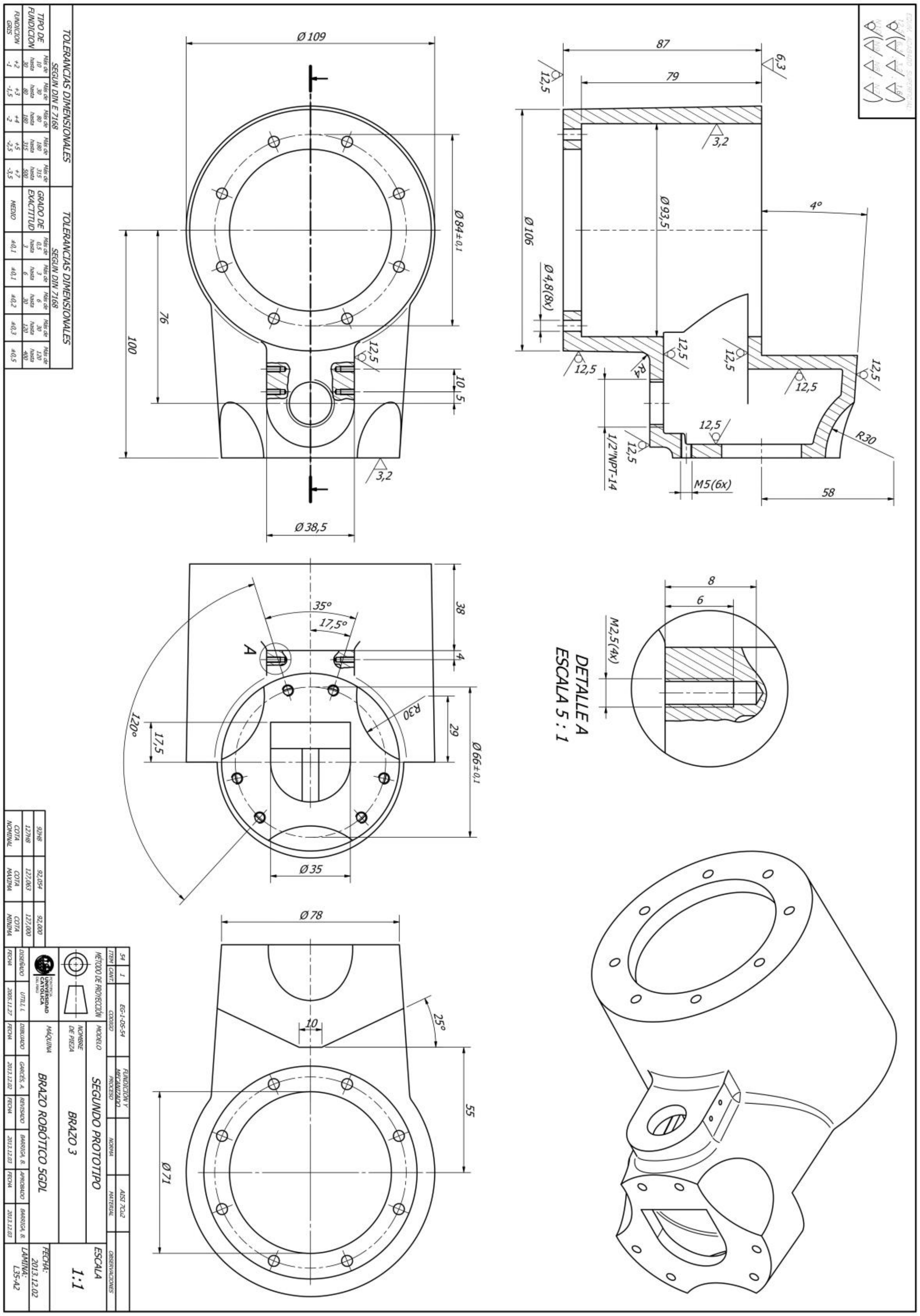


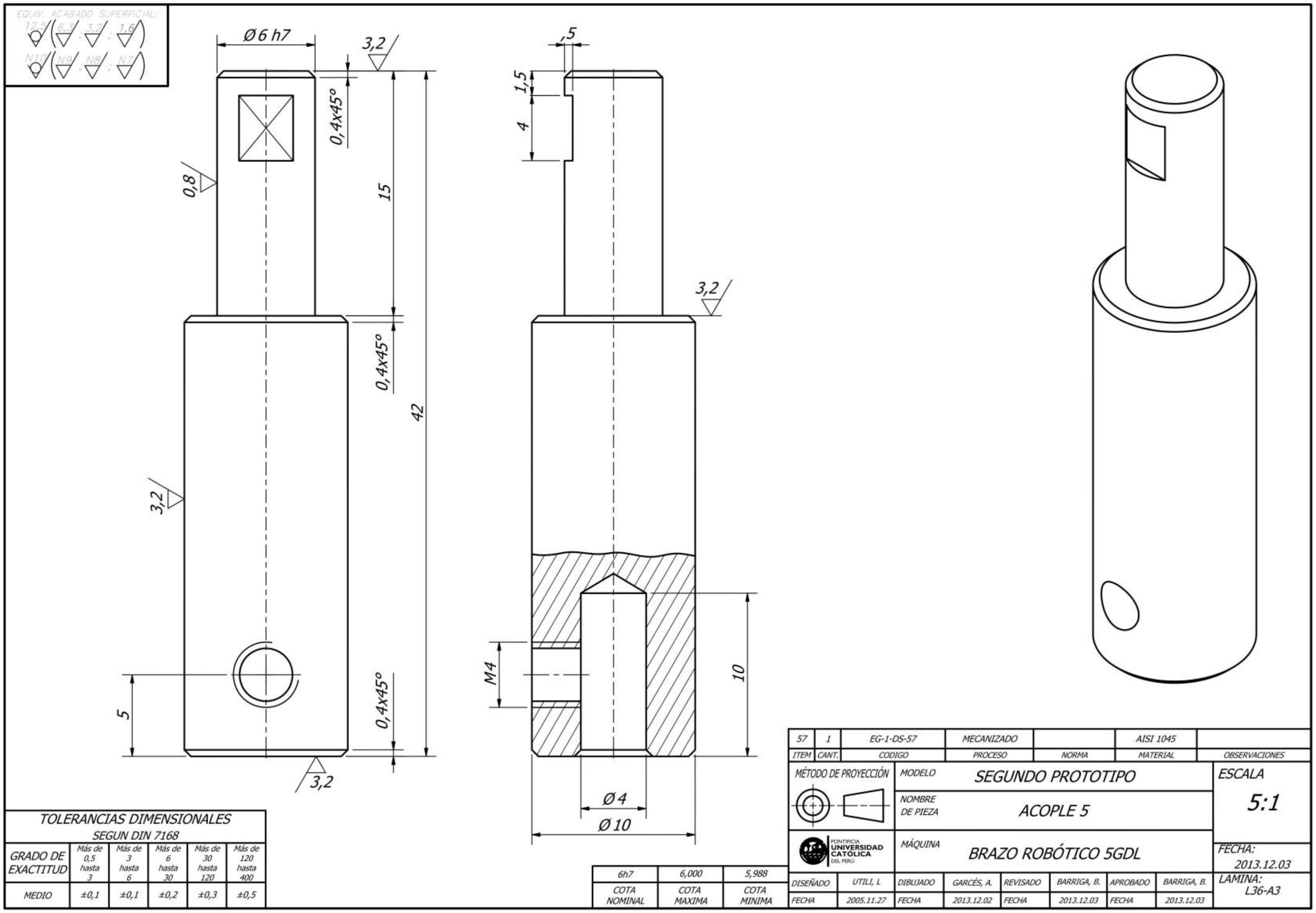
TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
51	1	EG-1-DS-51	MECANIZADO		AISI 7Cu2	

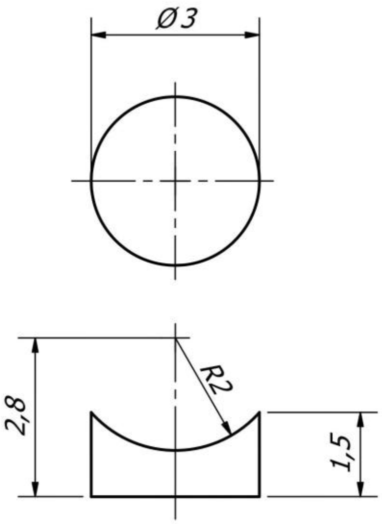
MÉTODO DE PROYECCIÓN	MODELO	SEGUNDO PROTOTIPO				ESCALA 2:1
	NOMBRE DE PIEZA	BRIDA 4				
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ	MÁQUINA	BRAZO ROBÓTICO 5GDL				FECHA: 2013.12.03
DISEÑADO: UTILI, L	DIBUJADO: GARCÉS, A.	REVISADO: BARRIGA, B.	APROBADO: BARRIGA, B.			LAMINA: L33-A3
FECHA: 2005.11.27	FECHA: 2013.12.02	FECHA: 2013.12.02	FECHA: 2013.12.03			

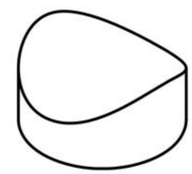

EQUIV. ACABADO SUPERFICIAL
TOLERANCIAS DIMENSIONALES SEGÚN DIN 7168
GRADO DE EXACTITUD
MEDIO
Más de 0,5 hasta 3
Más de 3 hasta 6
Más de 6 hasta 30
Más de 30 hasta 120
Más de 120 hasta 400
±0,1
±0,1
±0,2
±0,3
±0,5
Ø84±0,1
Ø60±0,1
M4(6x)
Ø4,8(6x)
1x45°
1x45°
10
8
DETALLE A
ESCALA 2 : 1
45°
60°
0,5x45°
3,15
9
1,6
0,5x45°
DETALLE B
ESCALA 2 : 1
Ø92,5 0 -0,3
Ø45
3,2
A
3,2
1,6
85 +0,1 0
Ø78 H7
94
19
3,2
B
3,2
Ø85 H7
3,2
Ø88,5 +0,35 0
78H7
85H7
COTA NOMINAL
78,030
85,035
COTA MÁXIMA
78,000
85,000
COTA MÍNIMA
53
1
EG-1-D5-53
MECANIZADO
AISI 7Cu2
ÍTEM
CANT.
CÓDIGO
PROCESO
NORMA
MATERIAL
OBSERVACIONES
MÉTODO DE PROYECCIÓN
MODELO
SEGUNDO PROTOTIPO
ESCALA
1:1
NOMBRE DE PIEZA
BASE 4
MÁQUINA
BRAZO ROBÓTICO 5GDL
DISEÑADO
UTILL, L.
DIBUJADO
GARCÉS, A.
REVISADO
BARRIGA, B.
APROBADO
BARRIGA, B.
FECHA
2005.11.27
FECHA
2013.12.02
FECHA
2013.12.03
FECHA
2013.12.03
FECHA:
2013.12.03
LÁMINA:
L34-A3


TOLERANCIAS DIMENSIONALES SEGUN DIN E 7168
TOLERANCIAS DIMENSIONALES SEGUN DIN 7168
DETALLE A
ESCALA 5 : 1
92H8
127H8
COTA NOMINAL
92,054
127,063
COTA MAXIMA
92,000
127,000
COTA MINIMA
54
1
EG-1-DS-54
FUNDICIÓN Y MECANIZADO
AISI 7Cu2
MÉTODO DE PROYECCIÓN
MODELO
SEGUNDO PROTOTIPO
NOMBRE DE PIEZA
BRAZO 3
MÁQUINA
BRAZO ROBÓTICO 5GDL
ESCALA
1:1
DISEÑADO
UTILL I.
2005.11.27
DIBUJADO
GARCÉS, A.
2013.12.02
REVISADO
BARRIGA, B.
2013.12.03
APROBADO
BARRIGA, B.
2013.12.03
FECHA:
2013.12.02
LAMINA:
L35-A2


EQUIV. ACABADO SUPERFICIAL:
12,5 6,3 3,2 1,6
N10 N9 N8 N7
Ø 6 h7
3,2
0,8
0,4x45°
15
42
5
.5
1,5
4
M4
10
Ø 4
Ø 10
TOLERANCIAS DIMENSIONALES
SEGUN DIN 7168
GRADO DE EXACTITUD
Más de 0,5 hasta 3
Más de 3 hasta 6
Más de 6 hasta 30
Más de 30 hasta 120
Más de 120 hasta 400
MEDIO
±0,1
±0,1
±0,2
±0,3
±0,5
6h7
6,000
5,988
COTA NOMINAL
COTA MAXIMA
COTA MINIMA
57
1
EG-1-DS-57
MECANIZADO
AISI 1045
ITEM
CANT.
CODIGO
PROCESO
NORMA
MATERIAL
OBSERVACIONES
MÉTODO DE PROYECCIÓN
MODELO
SEGUNDO PROTOTIPO
ESCALA
5:1
NOMBRE DE PIEZA
ACOPLE 5
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
MÁQUINA
BRAZO ROBÓTICO 5GDL
FECHA:
2013.12.03
DISEÑADO
UTILI, L
DIBUJADO
GARCÉS, A.
REVISADO
BARRIGA, B.
APROBADO
BARRIGA, B.
LAMINA:
L36-A3
FECHA
2005.11.27
FECHA
2013.12.02
FECHA
2013.12.03
FECHA
2013.12.03

EQUIV. ACABADO SUPERFICIAL:
12,5 (6,3 ; 3,2 ; 1,6)
N10 (N9 ; N8 ; N7)



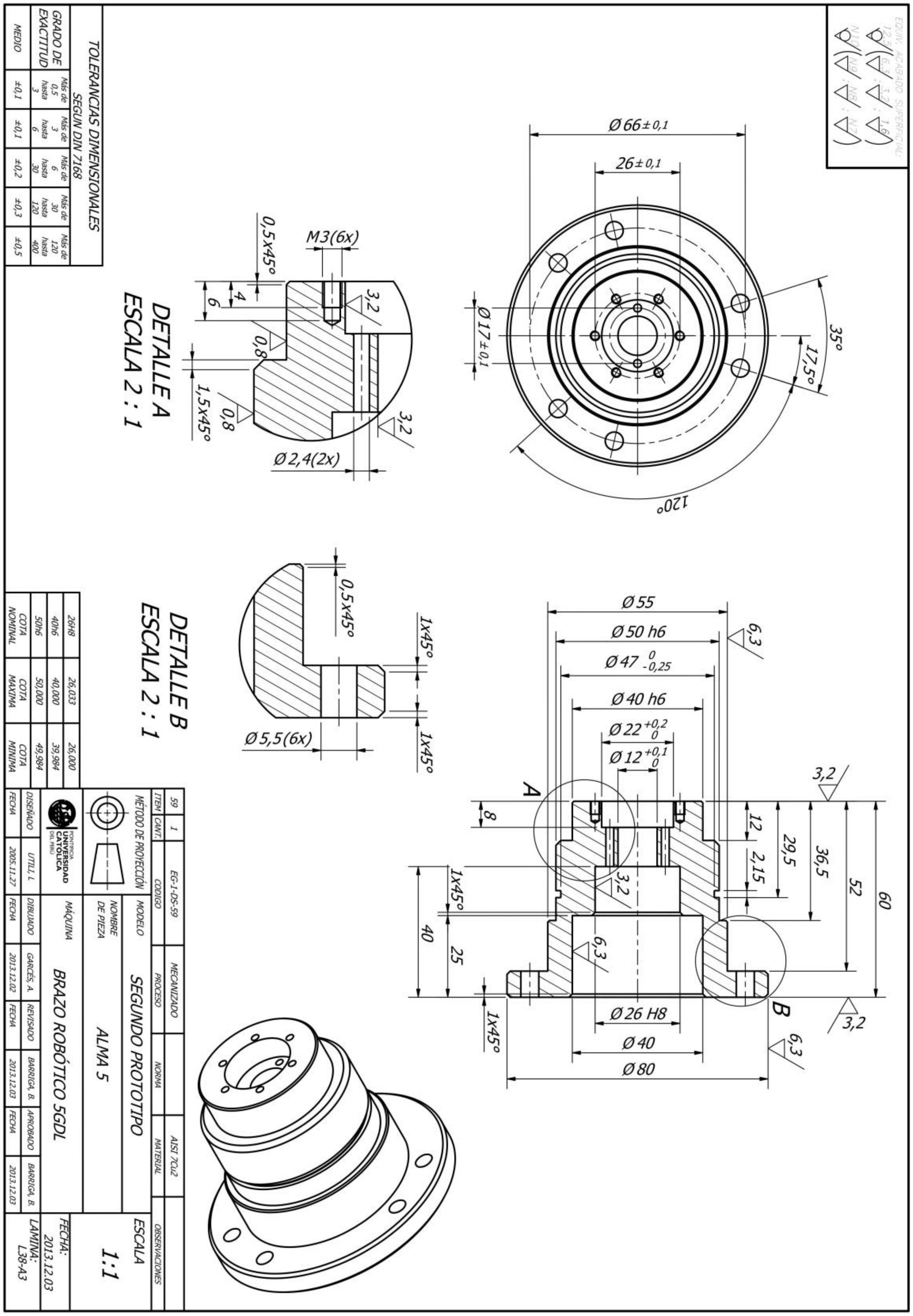


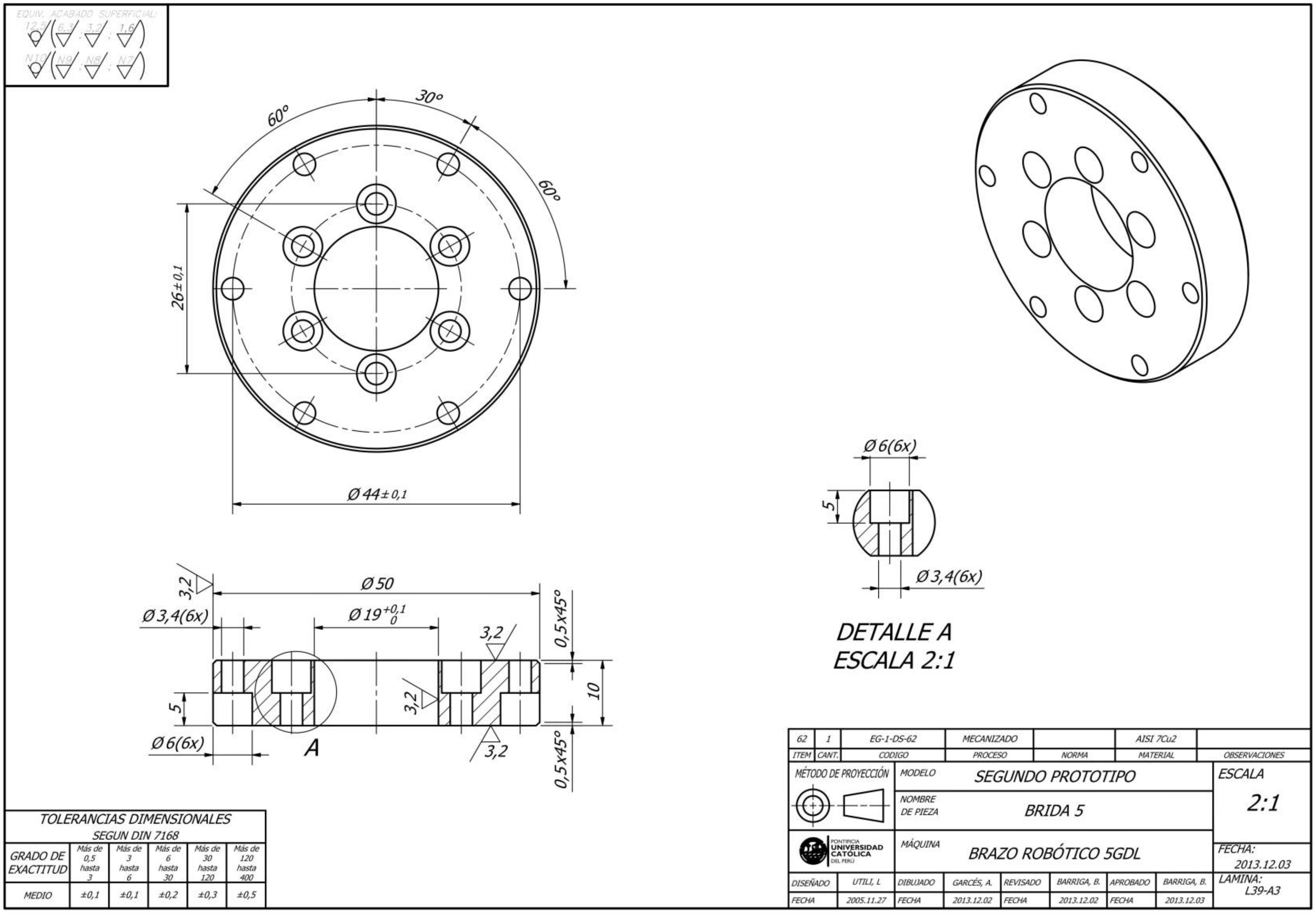
TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

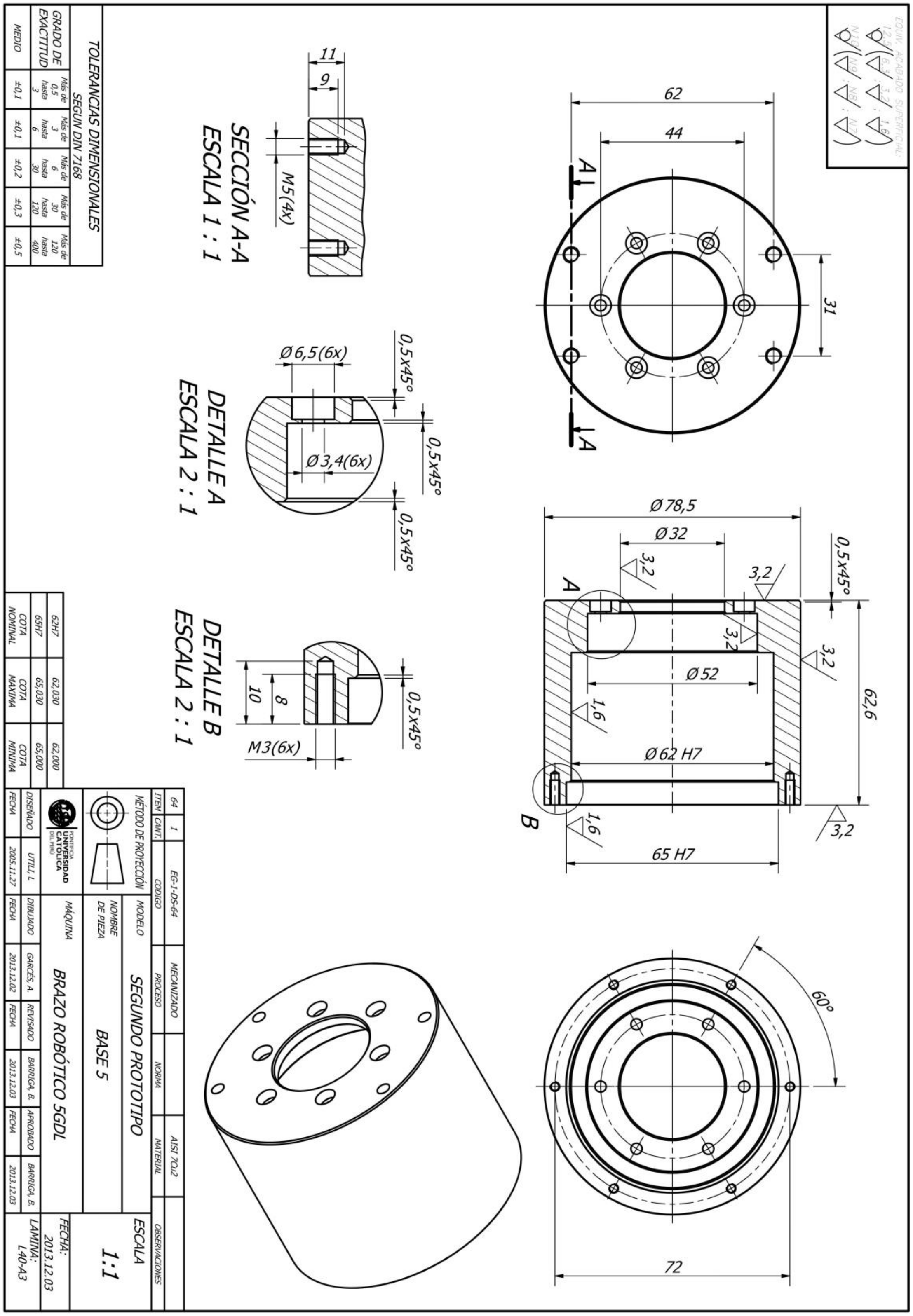
ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
58	1	EG-1-DS-58	MECANIZADO		BRONCE	

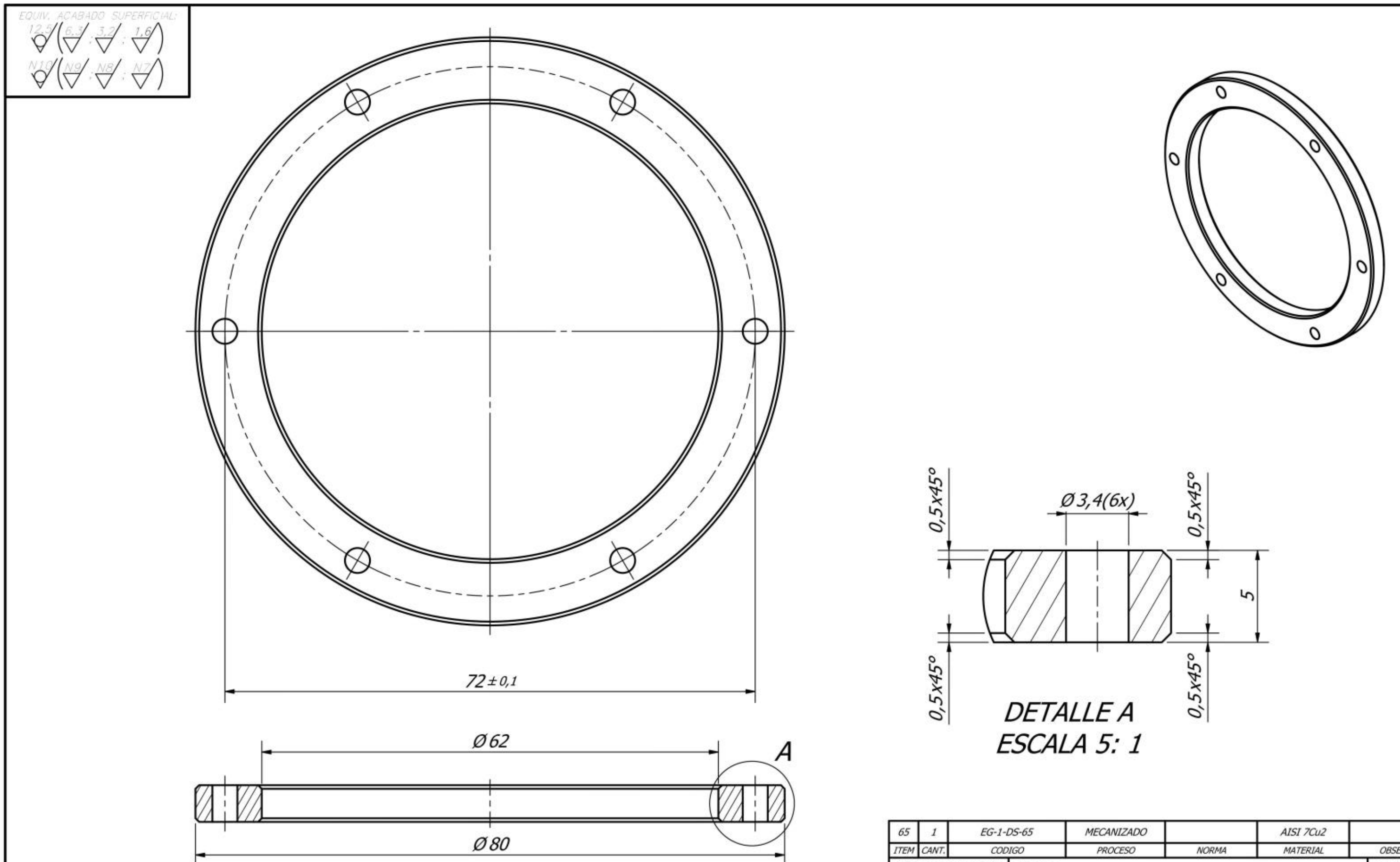
MÉTODO DE PROYECCIÓN	MODELO	SEGUNDO PROTOTIPO	ESCALA 10:1
	NOMBRE DE PIEZA	PLAQUITA 5	
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ	MÁQUINA	BRAZO ROBÓTICO 5GDL	FECHA: 2013.12.03

DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO	BARRIGA, B.	LAMINA: L37-A4
FECHA	2005.11.27	FECHA	2013.12.02	FECHA	2013.12.03	FECHA	2013.12.03	


TOLERANCIAS DIMENSIONALES SEGÚN DIN 7168
GRADO DE EXACTITUD
Más de 0,5 hasta 3
Más de 3 hasta 6
Más de 6 hasta 30
Más de 30 hasta 120
Más de 120 hasta 400
MEDIO ±0,1 ±0,1 ±0,2 ±0,3 ±0,5
COTA NOMINAL / COTA MÁXIMA / COTA MÍNIMA
26H8 26,033 26,000
40h6 40,000 39,984
50h6 50,000 49,984
DETALLE A
ESCALA 2 : 1
DETALLE B
ESCALA 2 : 1
M3(6x)
0,5x45°
Ø 2,4(2x)
1,5x45°
Ø 5,5(6x)
1x45°
Ø 66 ± 0,1
26 ± 0,1
Ø 17 ± 0,1
35°
17,5°
120°
Ø 55
Ø 50 h6
Ø 40 h6
Ø 12
Ø 26 H8
Ø 40
Ø 80
59 | 1 | EG-1-DS-59 | MECANIZADO | | AISI 7Cu2
ITEM | CANT. | CODIGO | PROCESO | NORMA | MATERIAL | OBSERVACIONES
MÉTODO DE PROYECCIÓN
MODELO SEGUNDO PROTOTIPO
NOMBRE DE PIEZA ALMA 5
MÁQUINA BRAZO ROBÓTICO 5GDL
ESCALA 1:1
DISEÑADO UTILL, L
FECHA 2005.11.27
DIBUJADO GARCÉS, A.
FECHA 2013.12.02
REVISADO BARRIGA, B.
FECHA 2013.12.03
APROBADO BARRIGA, B.
FECHA 2013.12.03
FECHA: 2013.12.03
LAMINA: L38-A3


EQUIV. ACABADO SUPERFICIAL:
12,5 6,3 3,2 1,6
N10 N9 N8 N7
60°
30°
60°
26±0,1
Ø44±0,1
Ø50
Ø3,4(6x)
Ø19 +0,1 0
3,2
0,5x45°
10
5
Ø6(6x)
A
DETALLE A
ESCALA 2:1
TOLERANCIAS DIMENSIONALES
SEGUN DIN 7168
GRADO DE EXACTITUD
Más de 0,5 hasta 3
Más de 3 hasta 6
Más de 6 hasta 30
Más de 30 hasta 120
Más de 120 hasta 400
MEDIO
±0,1
±0,1
±0,2
±0,3
±0,5
62
1
EG-1-DS-62
MECANIZADO
AISI 7Cu2
ITEM
CANT.
CODIGO
PROCESO
NORMA
MATERIAL
OBSERVACIONES
MÉTODO DE PROYECCIÓN
MODELO
SEGUNDO PROTOTIPO
ESCALA
2:1
NOMBRE DE PIEZA
BRIDA 5
MÁQUINA
BRAZO ROBÓTICO 5GDL
FECHA:
2013.12.03
DISEÑADO
UTILI, L
DIBUJADO
GARCÉS, A.
REVISADO
BARRIGA, B.
APROBADO
BARRIGA, B.
LAMINA:
L39-A3
FECHA
2005.11.27
FECHA
2013.12.02
FECHA
2013.12.02
FECHA
2013.12.03


EQUIV. ACABADO SUPERFICIAL
62
44
31
A
A
Ø 78,5
Ø 32
3,2
0,5x45°
Ø 52
62,6
1,6
Ø 62 H7
65 H7
B
60°
72
11
9
M5(4x)
SECCIÓN A-A
ESCALA 1 : 1
Ø 6,5(6x)
Ø 3,4(6x)
DETALLE A
ESCALA 2 : 1
10
8
M3(6x)
DETALLE B
ESCALA 2 : 1
TOLERANCIAS DIMENSIONALES SEGÚN DIN 7168
GRADO DE EXACTITUD
Más de 0,5 hasta 3
Más de 3 hasta 6
Más de 6 hasta 30
Más de 30 hasta 120
Más de 120 hasta 400
MEDIO ±0,1 ±0,1 ±0,2 ±0,3 ±0,5
COTA NOMINAL COTA MAXIMA COTA MINIMA
62H7 62,030 62,000
65H7 65,030 65,000
ÍTEM CANT. CÓDIGO PROCESO NORMA MATERIAL OBSERVACIONES
64 1 EG-1-DS-64 MECANIZADO AISI 7Cu2
MÉTODO DE PROYECCIÓN
MODELO SEGUNDO PROTOTIPO
NOMBRE DE PIEZA BASE 5
MÁQUINA BRAZO ROBÓTICO 5GDL
ESCALA 1:1
FECHA: 2013.12.03
LAMINA: L40-A3
DISEÑADO UTILL, L FECHA 2005.11.27
DIBUJADO GARCÉS, A. FECHA 2013.12.02
REVISADO BARRIGA, B. FECHA 2013.12.03
APROBADO BARRIGA, B. FECHA 2013.12.03

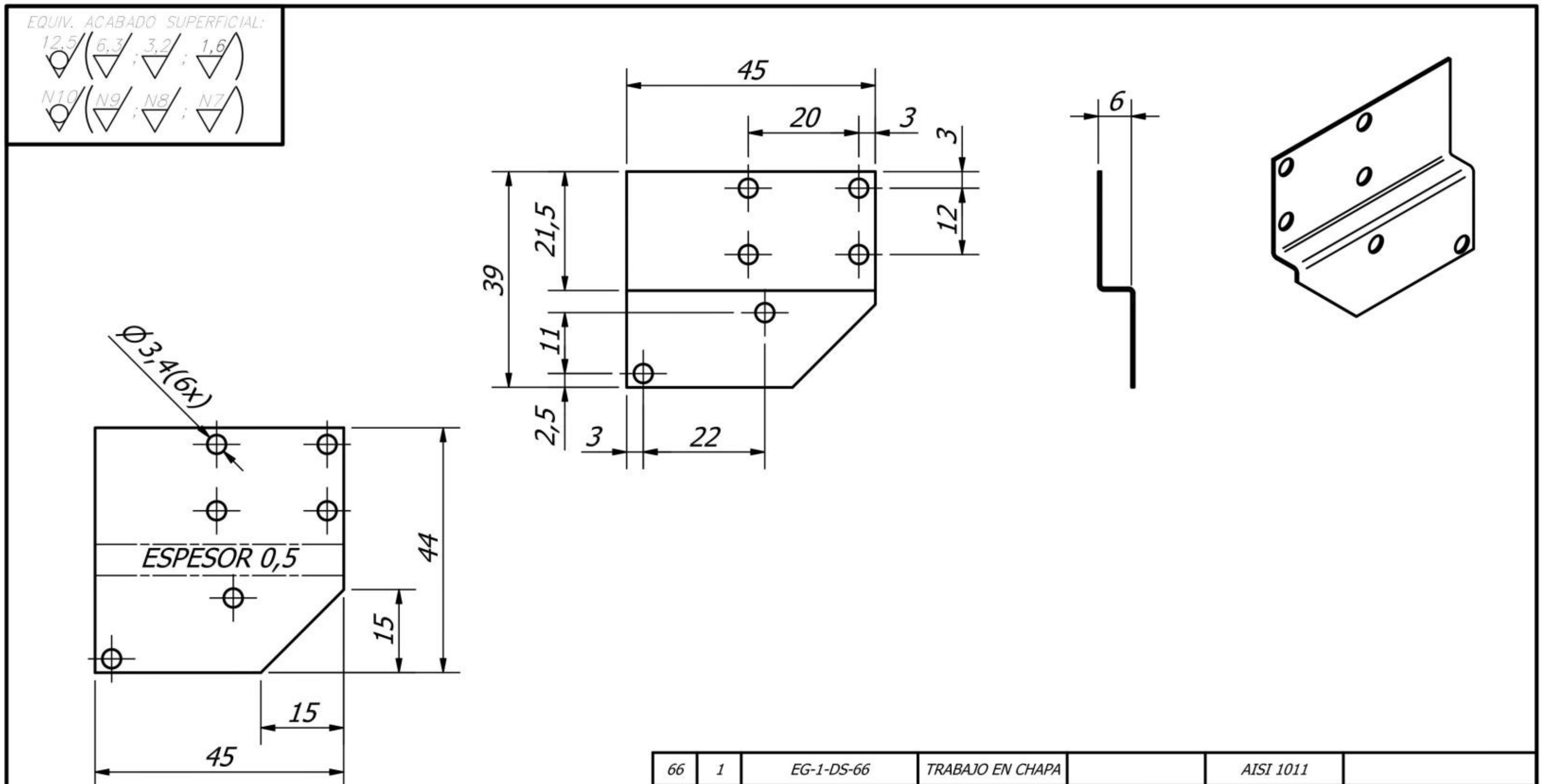


TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
65	1	EG-1-DS-65	MECANIZADO		AISI 7Cu2	

MÉTODO DE PROYECCIÓN	MODELO: SEGUNDO PROTOTIPO	ESCALA 2:1
	NOMBRE DE PIEZA: TAPA INFERIOR 5	
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ	MÁQUINA: BRAZO ROBÓTICO 5GDL	FECHA: 2013.12.02

DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO	BARRIGA, B.	LAMINA: L41-A3
FECHA	2005.11.27	FECHA	2013.12.02	FECHA	2013.12.03	FECHA	2013.12.03	



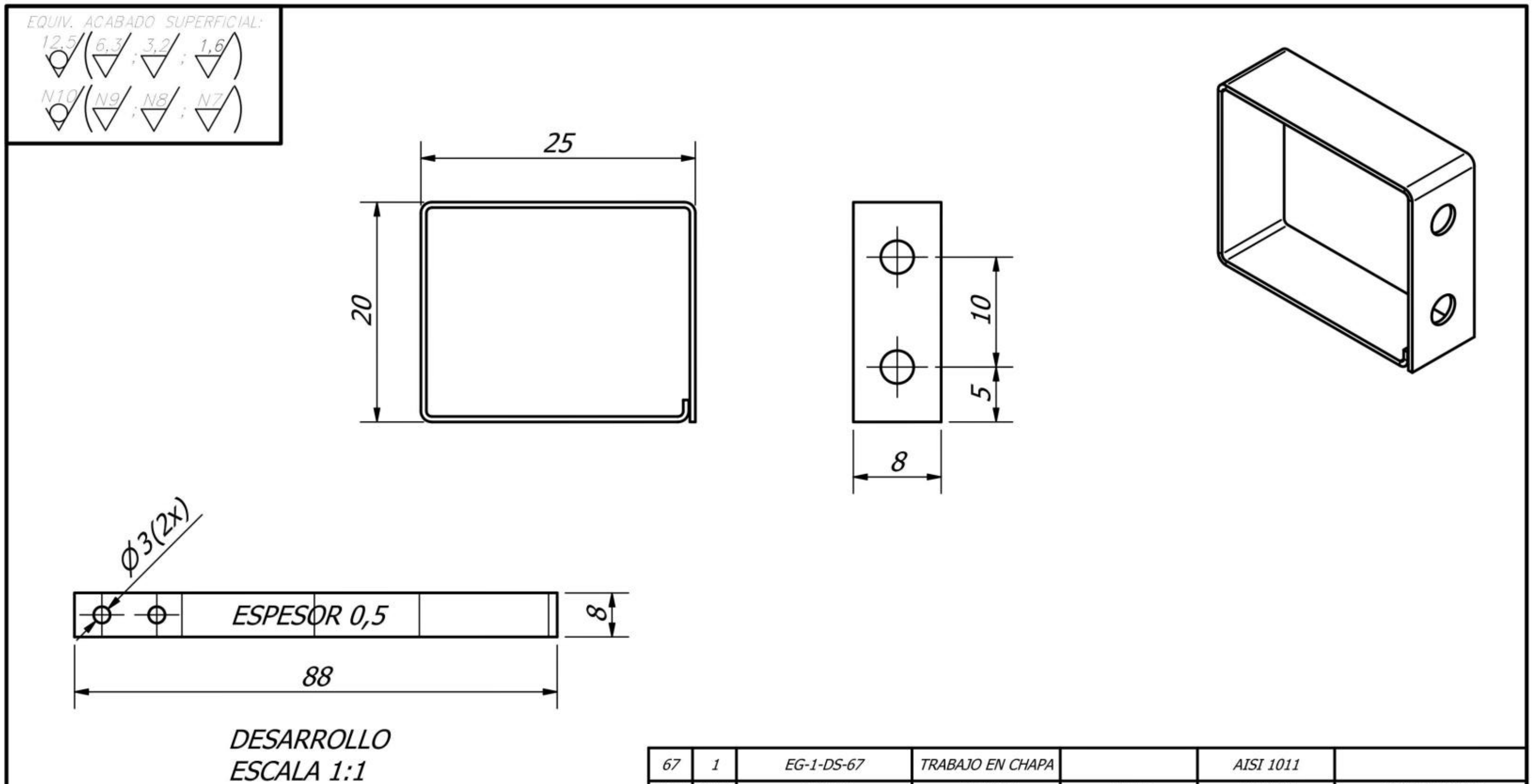
DESARROLLO
ESCALA 1:1

TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
66	1	EG-1-DS-66	TRABAJO EN CHAPA		AISI 1011	

MÉTODO DE PROYECCIÓN	MODELO: SEGUNDO PROTOTIPO	ESCALA 1:1
	NOMBRE DE PIEZA: PLANCHA SWITCH	
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ	MÁQUINA: BRAZO ROBÓTICO 5GDL	FECHA: 2013.12.03

DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO	BARRIGA, B.	LAMINA: L42-A4
FECHA	2005.11.27	FECHA	2013.12.02	FECHA	2013.12.03	FECHA	2013.12.03	



TOLERANCIAS DIMENSIONALES SEGUN DIN 7168					
GRADO DE EXACTITUD	Más de 0,5 hasta 3	Más de 3 hasta 6	Más de 6 hasta 30	Más de 30 hasta 120	Más de 120 hasta 400
MEDIO	±0,1	±0,1	±0,2	±0,3	±0,5

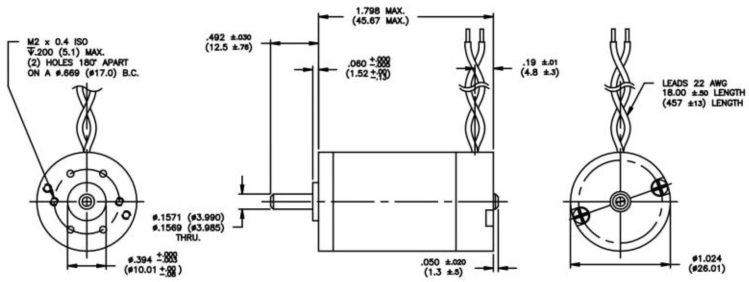
ITEM	CANT.	CODIGO	PROCESO	NORMA	MATERIAL	OBSERVACIONES
67	1	EG-1-DS-67	TRABAJO EN CHAPA		AISI 1011	

MÉTODO DE PROYECCIÓN		MODELO	SEGUNDO PROTOTIPO					ESCALA 2:1
		NOMBRE DE PIEZA	PLANCHA TOPE					
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ		MÁQUINA	BRAZO ROBÓTICO 5GDL					FECHA: 2013.12.03
DISEÑADO	UTILI, L	DIBUJADO	GARCÉS, A.	REVISADO	BARRIGA, B.	APROBADO	BARRIGA, B.	LAMINA: L43-A4
FECHA	2005.11.27	FECHA	2013.12.02	FECHA	2013.12.03	FECHA	2013.12.03	

Brush Commutated DC Servo Motors

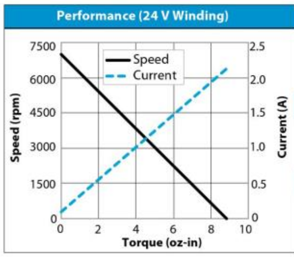
8691 Series

PITTMAN®



Specification	Units	Part/Model Number							
		8691 6.0 V	8691 7.58 V	8691 9.55 V	8691 12.0 V	8691 15.2 V	8691 19.1 V	8691 24.0 V	8691 30.3 V
Supply Voltage	VDC	6.00	7.58	9.55	12.0	15.2	19.1	24.0	30.3
Continuous Torque	oz-in	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9
	Nm	0.0134	0.0134	0.0134	0.0134	0.0134	0.0134	0.0134	0.0134
Speed @ Cont. Torque	RPM	4470	4620	4700	4780	4920	4880	4890	4890
Current @ Cont. Torque	Amps (A)	2.32	1.83	1.45	1.16	0.93	0.73	0.57	0.46
Continuous Output Power	Watts (W)	6.4	6.6	6.7	6.8	7.0	7.0	7.0	7.0
Motor Constant	oz-in/sqrt W	1.2	1.2	1.3	1.3	1.3	1.3	1.3	1.3
	Nm/sqrt W	0.008	0.008	0.009	0.009	0.009	0.009	0.009	0.009
Torque Constant	oz-in/A	1.08	1.37	1.74	2.17	2.71	3.44	4.35	5.47
	Nm/A	0.008	0.01	0.012	0.015	0.019	0.024	0.031	0.039
Voltage Constant	V/krpm	0.80	1.01	1.29	1.60	2.00	2.54	3.22	4.04
	V/rad/s	0.008	0.01	0.012	0.015	0.019	0.024	0.031	0.039
Terminal Resistance	Ohms	0.80	1.22	1.87	2.89	4.47	7.08	11.3	17.8
Inductance	mH	0.41	0.66	1.05	1.63	2.55	4.10	6.55	10.2
No-Load Current	Amps (A)	0.39	0.31	0.25	0.20	0.16	0.13	0.095	0.080
No-Load Speed	RPM	6980	6970	6920	6980	7080	7000	6990	7000
Peak Current	Amps (A)	7.50	6.21	5.11	4.15	3.40	2.70	2.13	1.71
Peak Torque	oz-in	7.68	8.09	8.45	8.58	8.78	8.83	8.86	8.89
	Nm	0.0542	0.0571	0.0597	0.0606	0.062	0.0623	0.0626	0.0628
Coulomb Friction Torque	oz-in	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
	Nm	0.0021	0.0021	0.0021	0.0021	0.0021	0.0021	0.0021	0.0021
Viscous Damping Factor	oz-in/krpm	0.0087	0.0087	0.0087	0.0087	0.0087	0.0087	0.0087	0.0087
	Nm s/rad	5.84E-7	5.84E-7	5.84E-7	5.84E-7	5.84E-7	5.84E-7	5.84E-7	5.84E-7
Electrical Time Constant	ms	0.51	0.54	0.56	0.56	0.57	0.58	0.58	0.58
Mechanical Time Constant	ms	14	13	12	12	12	12	12	12
Thermal Time Constant	min	13	13	13	13	13	13	13	13
Thermal Resistance	Celsius/W	19	19	19	19	19	19	19	19
Max. Winding Temperature	Celsius	130	130	130	130	130	130	130	130
Rotor Inertia	oz-in-sec2	0.00014	0.00014	0.00014	0.00014	0.00014	0.00014	0.00014	0.00014
	kg-m2	9.89E-7	9.89E-7	9.89E-7	9.89E-7	9.89E-7	9.89E-7	9.89E-7	9.89E-7
Weight (Mass)	oz	2.7	2.7	2.7	2.7	2.7	2.7	2.7	2.7
	g	76.5	76.5	76.5	76.5	76.5	76.5	76.5	76.5

Performance (24 V Winding)



Standard Features

- Ball Bearings
- 2-Pole Stator
- Neodymium Magnets
- 7-Slot Armature
- Heavy-Gage Steel Housing
- Silicon Steel Laminations
- Copper-Graphite Brushes
- Diamond-Turned Commutator

Complementary Products

- Encoders
- Gearboxes
- Brakes

Notes

1 All values specified at 25°C ambient temperature and without heat sink.
2 Peak values are theoretical and supplied for reference only.

This document is for informational purposes only and should not be considered as a binding description of the products or their performance in all applications. The performance data on this page depicts typical performance under controlled laboratory conditions. Actual performance will vary depending on the operating environment and application. AMETEK reserves the right to revise its products without notification. The above characteristics represent standard products. For products designed to meet specific applications, contact PITTMAN Motor Sales Department.

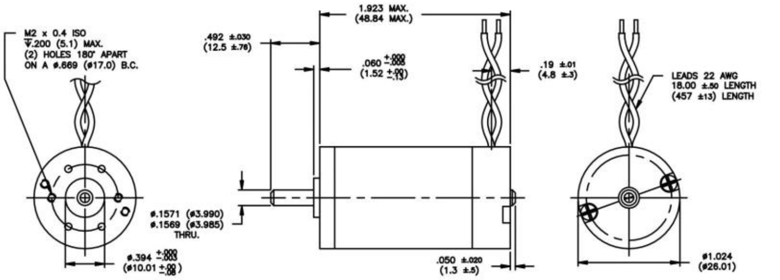
PITTMAN PRODUCTS
343 Godshall Drive, Harleysville, PA 19438
USA: +1 267 933 2105 - Europe: +33 240928751 - Asia: +86 21 5763 1258
www.pittman-motors.com

AMETEK®
PRECISION MOTION CONTROL

Brush Commutated DC Servo Motors

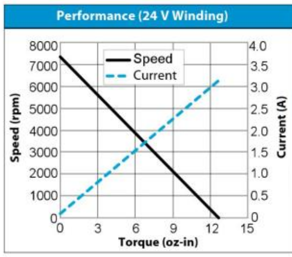
8692 Series

PITTMAN®



Specification	Units	8692 7.58 V	8692 9.55 V	8692 12.0 V	8692 15.2 V	8692 19.1 V	8692 24.0 V	8692 30.3 V	8692 38.2 V
		Part/Model Number							
Supply Voltage	VDC	7.58	9.55	12.0	15.2	19.1	24.0	30.3	38.2
Continuous Torque	oz-in	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4
	Nm	0.0169	0.0169	0.0169	0.0169	0.0169	0.0169	0.0169	0.0169
Speed @ Cont. Torque	RPM	5140	5270	5300	5470	5510	5490	5470	5540
Current @ Cont. Torque	Amps (A)	2.33	1.84	1.46	1.16	0.93	0.73	0.58	0.46
Continuous Output Power	Watts (W)	9.3	9.5	9.6	9.9	9.9	9.9	9.9	10.0
Motor Constant	oz-in/sqrt W	1.4	1.5	1.5	1.5	1.5	1.5	1.5	1.5
	Nm/sqrt W	0.01	0.011	0.011	0.011	0.011	0.011	0.011	0.011
Torque Constant	oz-in/A	1.32	1.67	2.11	2.64	3.30	4.18	5.28	6.64
	Nm/A	0.009	0.012	0.015	0.019	0.023	0.03	0.037	0.047
Voltage Constant	V/krpm	0.98	1.23	1.56	1.95	2.44	3.09	3.90	4.91
	V/rad/s	0.009	0.012	0.015	0.019	0.023	0.03	0.037	0.047
Terminal Resistance	Ohms	0.86	1.30	2.02	3.10	4.84	7.67	12.2	19.2
Inductance	mH	0.47	0.76	1.21	1.90	2.97	4.77	7.61	12.1
No-Load Current	Amps (A)	0.34	0.27	0.21	0.17	0.13	0.10	0.087	0.063
No-Load Speed	RPM	7320	7300	7270	7370	7410	7360	7340	7380
Peak Current	Amps (A)	8.81	7.35	5.94	4.90	3.95	3.13	2.49	1.99
Peak Torque	oz-in	11.2	11.8	12.1	12.5	12.6	12.6	12.7	12.8
	Nm	0.0791	0.0833	0.0854	0.0883	0.089	0.089	0.0897	0.0904
Coulomb Friction Torque	oz-in	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
	Nm	0.0021	0.0021	0.0021	0.0021	0.0021	0.0021	0.0021	0.0021
Viscous Damping Factor	oz-in/krpm	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.010
	Nm s/rad	6.71E-7	6.71E-7	6.71E-7	6.71E-7	6.71E-7	6.71E-7	6.71E-7	6.71E-7
Electrical Time Constant	ms	0.55	0.58	0.60	0.61	0.61	0.62	0.62	0.63
Mechanical Time Constant	ms	12	11	11	11	11	11	11	10
Thermal Time Constant	min	13	13	13	13	13	13	13	13
Thermal Resistance	Celsius/W	18	18	18	18	18	18	18	18
Max. Winding Temperature	Celsius	130	130	130	130	130	130	130	130
Rotor Inertia	oz-in-sec2	0.00017	0.00017	0.00017	0.00017	0.00017	0.00017	0.00017	0.00017
	kg-m2	1.2E-6	1.2E-6	1.2E-6	1.2E-6	1.2E-6	1.2E-6	1.2E-6	1.2E-6
Weight (Mass)	oz	3.1	3.1	3.1	3.1	3.1	3.1	3.1	3.1
	g	87.9	87.9	87.9	87.9	87.9	87.9	87.9	87.9

Performance (24 V Winding)



Standard Features

- Ball Bearings
- 2-Pole Stator
- Neodymium Magnets
- 7-Slot Armature
- Heavy-Gage Steel Housing
- Silicon Steel Laminations
- Copper-Graphite Brushes
- Diamond-Turned Commutator

Complementary Products

- Encoders
- Gearboxes
- Brakes

Notes

1 All values specified at 25°C ambient temperature and without heat sink.
2 Peak values are theoretical and supplied for reference only.

This document is for informational purposes only and should not be considered as a binding description of the products or their performance in all applications. The performance data on this page depicts typical performance under controlled laboratory conditions. Actual performance will vary depending on the operating environment and application. AMETEK reserves the right to revise its products without notification. The above characteristics represent standard products. For products designed to meet specific applications, contact PITTMAN Motor Sales Department.

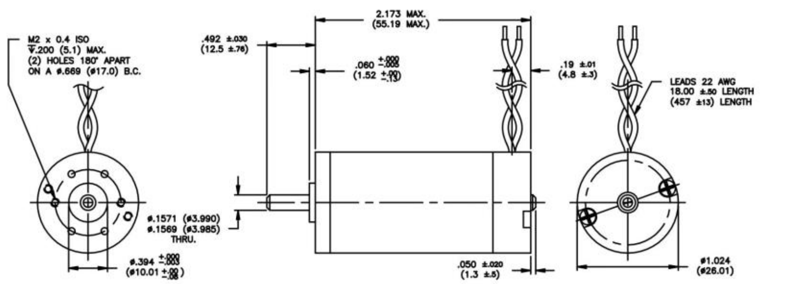
PITTMAN PRODUCTS
343 Godshall Drive, Harleysville, PA 19438
USA: +1 267 933 2105 - Europe: +33 240928751 - Asia: +86 21 5763 1258
www.pittman-motors.com

AMETEK®
PRECISION MOTION CONTROL

Brush Commutated DC Servo Motors

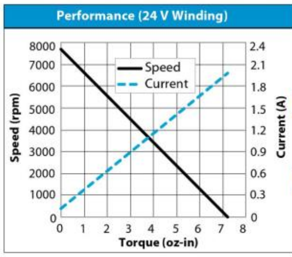
8693 Series

PITTMAN®



Specification	Units	Part/Model Number							
		8693 9.55 V	8693 12.0 V	8693 15.2 V	8693 19.1 V	8693 24.0 V	8693 30.3 V	8693 38.2 V	8693 48.0 V
Supply Voltage	VDC	9.55	12.0	15.2	19.1	24.0	30.3	38.2	48.0
Continuous Torque	oz-in	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2
	Nm	0.0226	0.0226	0.0226	0.0226	0.0226	0.0226	0.0226	0.0226
Speed @ Cont. Torque	RPM	6870	7070	7170	7180	7260	7370	7350	7310
Current @ Cont. Torque	Amps (A)	2.83	2.25	1.78	1.41	1.13	0.90	0.71	0.56
Continuous Output Power	Watts (W)	16	17	17	17	17	17	17	17
Motor Constant	oz-in/sqrt W	1.6	1.7	1.7	1.7	1.7	1.7	1.7	1.7
	Nm/sqrt W	0.011	0.012	0.012	0.012	0.012	0.012	0.012	0.012
Torque Constant	oz-in/A	1.38	1.73	2.19	2.77	3.46	4.313	5.47	6.92
	Nm/A	0.01	0.012	0.015	0.02	0.024	0.03	0.039	0.049
Voltage Constant	V/krpm	1.02	1.28	1.62	2.05	2.56	3.19	4.04	5.12
	V/rad/s	0.01	0.012	0.015	0.02	0.024	0.03	0.039	0.049
Terminal Resistance	Ohms	0.73	1.08	1.67	2.59	4.02	6.28	9.96	15.8
Inductance	mH	0.39	0.61	0.98	1.56	2.44	3.81	6.12	9.77
No-Load Current	Amps (A)	0.35	0.28	0.22	0.18	0.14	0.11	0.090	0.070
No-Load Speed	RPM	8930	8960	8980	8910	8980	9080	9040	8980
Peak Current	Amps (A)	13.08	11.11	9.10	7.37	5.97	4.82	3.84	3.03
Peak Torque	oz-in	17.6	18.7	19.5	19.9	20.2	20.4	20.5	20.5
	Nm	0.1243	0.132	0.1377	0.1405	0.1426	0.144	0.1447	0.1447
Coulomb Friction Torque	oz-in	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
	Nm	0.0021	0.0021	0.0021	0.0021	0.0021	0.0021	0.0021	0.0021
Viscous Damping Factor	oz-in/krpm	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015
	Nm s/rad	1.01E-6	1.01E-6	1.01E-6	1.01E-6	1.01E-6	1.01E-6	1.01E-6	1.01E-6
Electrical Time Constant	ms	0.53	0.56	0.59	0.60	0.61	0.61	0.61	0.62
Mechanical Time Constant	ms	12	12	11	11	11	11	11	11
Thermal Time Constant	min	12	12	12	12	12	12	12	12
Thermal Resistance	Celsius/W	16	16	16	16	16	16	16	16
Max. Winding Temperature	Celsius	130	130	130	130	130	130	130	130
Rotor Inertia	oz-in-sec2	0.00023	0.00023	0.00023	0.00023	0.00023	0.00023	0.00023	0.00023
	kg-m2	1.62E-6	1.62E-6	1.62E-6	1.62E-6	1.62E-6	1.62E-6	1.62E-6	1.62E-6
Weight (Mass)	oz	3.74	3.74	3.74	3.74	3.74	3.74	3.74	3.74
	g	106	106	106	106	106	106	106	106

Performance (24 V Winding)



Standard Features

- Sintered Bronze Bearings
- 2-Pole Stator
- Ceramic Magnets
- 7-Slot Armature
- Heavy-Gage Steel Housing
- Silicon Steel Laminations
- Copper-Graphite Brushes
- Diamond-Turned Commutator

Complementary Products

- Encoders
- Gearboxes
- Brakes

Notes

1 All values specified at 25°C ambient temperature and without heat sink.
2 Peak values are theoretical and supplied for reference only.

PITTMAN PRODUCTS
343 Godshall Drive, Harleysville, PA 19438
USA: +1 267 933 2105 - Europe: +33 240928751 - Asia: +86 21 5763 1258
www.pittman-motors.com

AMETEK® PRECISION MOTION CONTROL

HEDS-9040/9140

Three Channel Optical Incremental Encoder Modules

AVAGO TECHNOLOGIES

Data Sheet

Description

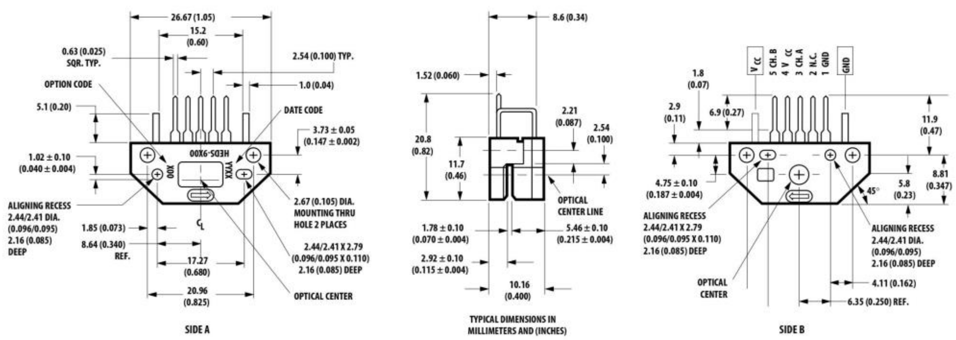
The HEDS-9040 and HEDS-9140 series are three channel optical incremental encoder modules. When used with a codewheel, these low cost modules detect rotary position. Each module consists of a lensed LED source and a detector IC enclosed in a small plastic package. Due to a highly collimated light source and a unique photodetector array, these modules provide the same high performance found in the HEDS-9000/9100 two channel encoder family.

Features

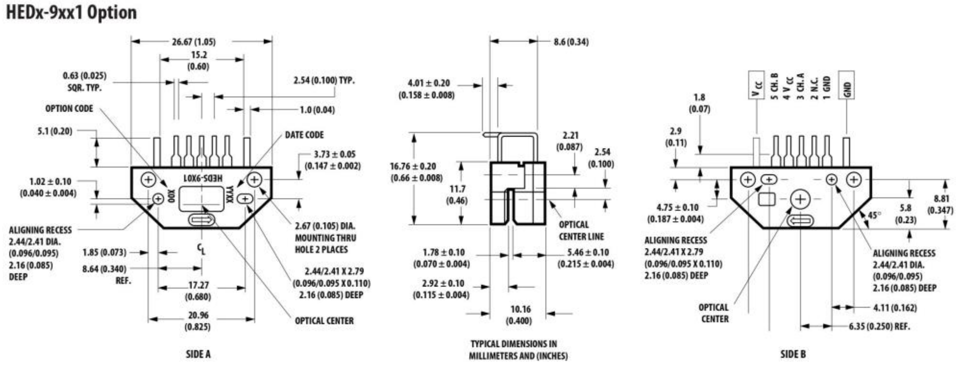
- Two channel quadrature output with index pulse
- Resolution up to 2000 CPR (Counts Per Revolution)
- Low cost
- Easy to mount
- No signal adjustment required
- Small size
- -40°C to 100°C operating temperature
- TTL compatible
- Single 5 V supply

Package Dimensions

HEDx-9xx0 Option



HEDx-9xx1 Option



ESD WARNING: NORMAL HANDLING PRECAUTIONS SHOULD BE TAKEN TO AVOID STATIC DISCHARGE.

The HEDS-9040 and 9140 have two channel quadrature outputs plus a third channel index output. This index output is a 90 electrical degree high true index pulse which is generated once for each full rotation of the codewheel.

The HEDS-9040 is designed for use with a HEDX-614X codewheel which has an optical radius of 23.36 mm (0.920 inch). The HEDS-9140 is designed for use with a HEDx-5x4x codewheel which has an optical radius of 11.00 mm (0.433 inch).

The quadrature signals and the index pulse are accessed through five 0.025 inch square pins located on 0.1 inch centers.

Standard resolutions between 256 and 2000 counts per revolution are available. Consult local Avago sales representatives for other resolutions.

Applications

The HEDS-9040 and 9140 provide sophisticated motion control detection at a low cost, making them ideal for high volume applications. Typical applications include printers, plotters, tape drives, and industrial and factory automation equipment.

Note: Avago Technologies encoders are not recommended for use in safety critical applications. Eg. ABS braking systems, power steering, life support systems and critical care medical equipment. Please contact sales representative if more clarification is needed.

Theory of Operation

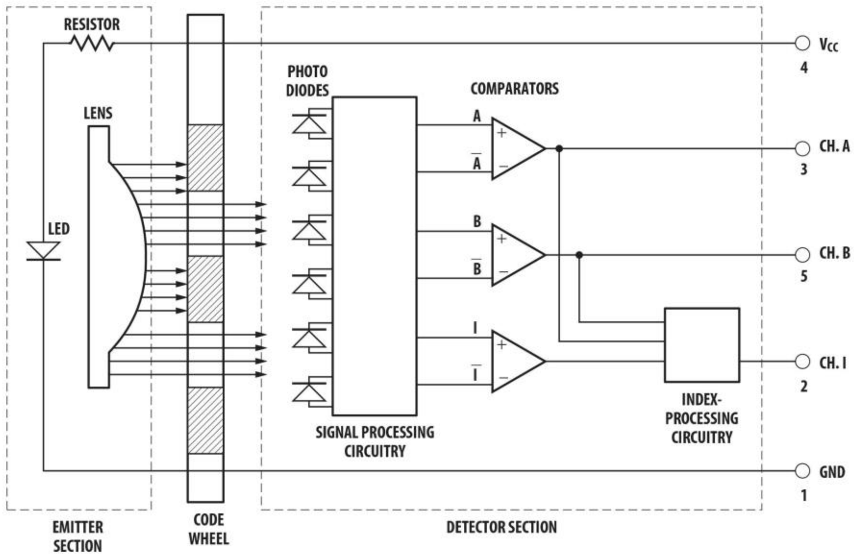
The HEDS-9040 and 9140 are emitter/detector modules. Coupled with a codewheel, these modules translate the rotary motion of a shaft into a three-channel digital output.

As seen in the block diagram, the modules contain a single Light Emitting Diode (LED) as its light source. The light is collimated into a parallel beam by means of a single polycarbonate lens located directly over the LED. Opposite the emitter is the integrated detector circuit. This IC consists of multiple sets of photodetectors and the signal processing circuitry necessary to produce the digital waveforms.

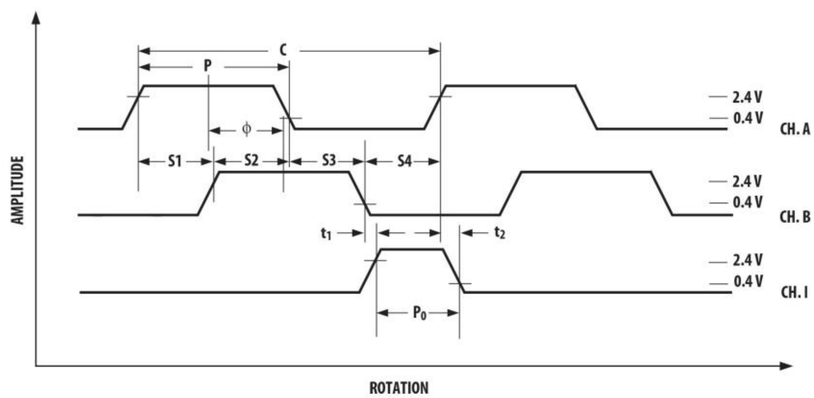
The codewheel rotates between the emitter and detector, causing the light beam to be interrupted by the pattern of spaces and bars on the codewheel. The photodiodes which detect these interruptions are arranged in a pattern that corresponds to the radius and design of the codewheel. These detectors are also spaced such that a light period on one pair of detectors corresponds to a dark period on the adjacent pair of detectors. The photodiode outputs are then fed through the signal processing circuitry resulting in A, $\overline{A}$, B, $\overline{B}$, I and $\overline{I}$. Comparators receive these signals and produce the final outputs for channels A and B. Due to this integrated phasing technique, the digital output of channel A is in quadrature with that of channel B (90 degrees out of phase).

The output of the comparator for I and I is sent to the index processing circuitry along with the outputs of channels A and B. The final output of channel I is an index pulse P_O which is generated once for each full rotation of the codewheel. This output P_O is a one state width (nominally 90 electrical degrees), high true index pulse which is coincident with the low states of channels A and B.

Block Diagram



Output Waveforms



Definitions

Count (N): The number of bar and window pairs or counts per revolution (CPR) of the codewheel.

One Cycle (C): 360 electrical degrees (°e), 1 bar and window pair.

One Shaft Rotation: 360 mechanical degrees, N cycles.

Position Error ($\Delta\Theta$): The normalized angular difference between the actual shaft position and the position indicated by the encoder cycle count.

Cycle Error (ΔC): An indication of cycle uniformity. The difference between an observed shaft angle which gives rise to one electrical cycle, and the nominal angular increment of 1/N of a revolution.

Pulse Width (P): The number of electrical degrees that an output is high during 1 cycle. This value is nominally 180°e or 1/2 cycle.

Pulse Width Error (ΔP): The deviation, in electrical degrees, of the pulse width from its ideal value of 180°e.

State Width (S): The number of electrical degrees between a transition in the output of channel A and the neighboring transition in the output of channel B. There are 4 states per cycle, each nominally 90°e.

State Width Error (ΔS): The deviation, in electrical degrees, of each state width from its ideal value of 90°e.

Phase (ϕ): The number of electrical degrees between the center of the high state of channel A and the center of the high state of channel B. This value is nominally 90°e for quadrature output.

Phase Error ($\Delta\phi$): The deviation of the phase from its ideal value of 90°e.

Direction of Rotation: When the codewheel rotates in the direction of the arrow on top of the module, channel A will lead channel B. If the codewheel rotates in the opposite direction, channel B will lead channel A.

Optical Radius (R_{OP}): The distance from the codewheel's center of rotation to the optical center (O.C.) of the encoder module.

Index Pulse Width (P_O): The number of electrical degrees that an index is high during one full shaft rotation. This value is nominally 90°e or 1/4 cycle.

Absolute Maximum Ratings

Storage Temperature, T_S .... -40°C to +100°C
Operating Temperature, T_A .... -40°C to +100°C
Supply Voltage, V_{CC} .... -0.5 V to 7 V
Output Voltage, V_O .... -0.5 V to V_{CC}
Output Current per Channel, I_{OUT} .... -1.0 mA to 5 mA
Shaft Axial Play .... ±0.25 mm (±0.010 in.)
Shaft Eccentricity Plus Radial Play .... 0.1 mm (0.004 in.) TIR
Velocity .... 30,000 RPM[1]
Acceleration .... 250,000 rad/sec^2[1]

Note:

1. Absolute maximums for HEDS-5140/6140 codewheels only.

Recommended Operating Conditions

Parameter	Symbol	Min.	Typ.	Max.	Units	Notes
Temperature	T_A	-40		100	°C	
Supply Voltage	V_{CC}	4.5	5.0	5.5	Volts	Ripple < 100 mV_{P-P}
Load Capacitance	C_L			100	pF	2.7 kΩ pull-up
Count Frequency	f			100	kHz	Velocity (rpm) x N/60
Shaft Perpendicularity				±0.25	mm	6.9 mm (0.27 in.) from
Plus Axial Play				(±0.010)	(in.)	mounting surface
Shaft Eccentricity Plus				0.04	mm (in.)	6.9 mm (0.27 in.) from
Radial Play				(0.0015)	TIR	mounting surface

Note: The module performance is guaranteed to 100 kHz but can operate at higher frequencies. For the HEDS-9040 #T00 for operation below 0°C and greater than 50 kHz the maximum Pulse Width and Logic State Width errors are 60°e.

Encoding Characteristics
HEDS-9040 (except #T00), HEDS-9140 (except #B00)

Encoding Characteristics over Recommended Operating Range and Recommended Mounting Tolerances unless otherwise specified. Values are for the worst error over the full rotation of HEDS-5140 and HEDS-6140 codewheels.

Parameter		Symbol	Min.	Typ.[1]	Max.	Units
Cycle Error		ΔC		3	5.5	°e
Pulse Width Error		ΔP		7	30	°e
Logic State Width Error		ΔS		5	30	°e
Phase Error		Δϕ		2	15	°e
Position Error		ΔΘ		10	40	min. of arc
Index Pulse Width		P_O	60	90	120	°e
CH. I rise after	-25°C to +100°C	t_1	10	100	250	ns
CH. B or CH. A fall	-40°C to +100°C	t_1	-300	100	250	ns
CH. I fall after	-25°C to +100°C	t_2	70	150	300	ns
CH. A or CH. B rise	-40°C to +100°C	t_2	70	150	1000	ns

Note:
1. Module mounted on tolerance circle of ±0.13 mm (±0.005 in.) radius referenced from module Side A aligning recess centers. 2.7 kΩ pull-up resistors used on all encoder module outputs.

Encoding Characteristics
HEDS-9040 #T00

Encoding Characteristics over Recommended Operating Range and Recommended Mounting Tolerances unless otherwise specified. Values are for the worst error over the full rotation of HEDM-614X Option TXX codewheel.

Parameter		Symbol	Min.	Typ.[1]	Max.	Units
Cycle Error		ΔC		3	7.5	°e
Pulse Width Error		ΔP		7	50	°e
Logic State Width Error		ΔS		5	50	°e
Phase Error		Δϕ		2	15	°e
Position Error		ΔΘ		2	20	min. of arc
Index Pulse Width		P_O	40	90	140	°e
CH. I rise after CH. B or CH. A fall	-40°C to +100°C	t_1	10	450	1500	ns
CH. I fall after CH. A or CH. B rise	-40°C to +100°C	t_2	10	250	1500	ns

Note:
1. Module mounted on tolerance circle of ±0.13 mm (±0.005 in.) radius referenced from module Side A aligning recess centers. 2.7 kΩ pull-up resistors used on all encoder module outputs.

Encoding Characteristic
HEDS-9140 #B00

Encoding Characteristics over Recommended Operating Range and Recommended Mounting Tolerances unless otherwise specified. Values are for the worst error over the full rotation of HEDM-504X Option BXX codewheel.

Parameter		Symbol	Min.	Typ.[1]	Max.	Units
Cycle Error		ΔC		6	12	°e
Pulse Width Error		ΔP		10	45	°e
Logic State Width Error		ΔS		10	45	°e
Phase Error		ΔΦ		2	15	°e
Position Error		ΔΘ		10	40	min. of arc
Index Pulse Width		P_o	50	90	130	°e
CH. I Rise after CH B or CH A fall	-40°C to +100°	t_1	200	1000	1500	ns
CH. I fall after CH. A or CH.B rise	-40°C to +100°	t_2	0	300	1500	ns

Note:
1. Module mounted on tolerance circle of ±0.13 mm (±0.005 in.) radius referenced from module Side A aligning recess centers. 2.7 kΩ pull-up resistors used on all encoder module outputs.

Electrical Characteristics

Electrical Characteristics over Recommended Operating Range.

Parameter	Symbol	Min.	Typ.[1]	Max.	Units	Notes
Supply Current	I_{CC}	30	57	85	mA	
High Level Output Voltage	V_{OH}	2.4			V	I_{OH} = -200 μA max.
Low Level Output Voltage	V_{OL}			0.4	V	I_{OL} = 3.86 mA
Rise Time	t_r		180[2]		ns	C_L = 25 pF R_L = 2.7 kΩ pull-up
Fall Time	t_f		49[2]		ns	

Notes:
1. Typical values specified at V_{CC} = 5.0 V and 25°C.
2. t_r and t_f 80 nsec for HEDS-9040 #T00.

Electrical Interface

To insure reliable encoding performance, the HEDS-9040 and 9140 three channel encoder modules require 2.7 kΩ (±10%) pull-up resistors on output pins 2, 3, and 5 (Channels I, A and B) as shown in Figure 1. These pull-up resistors should be located as close to the encoder module as possible (within 4 feet). Each of the three encoder module outputs can drive a single TTL load in this configuration.

Mounting Considerations

Figure 2 shows a mounting tolerance *requirement* for proper operation of the HEDS-9040 and HEDS-9140. The Aligning Recess Centers must be located within a tolerance circle of 0.005 in. radius from the nominal locations. This tolerance must be maintained whether the module is mounted with side A as the mounting plane using aligning pins (see Figure 5), or mounted with Side B as the mounting plane using an alignment tool (see Figures 3 and 4).

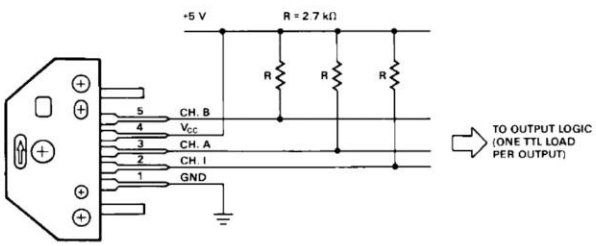


Figure 1. Pull-up Resistors on HEDS-9X40 Encoder Module Outputs.

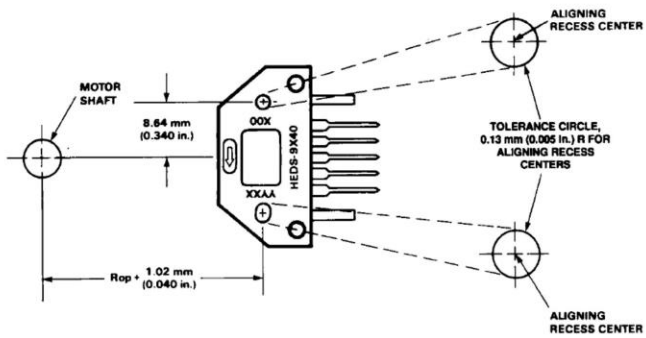


Figure 2. HEDS-9X40 Mounting Tolerance.

Mounting with an Alignment Tool

The HEDS-8905 and HEDS-8906 alignment tools are recommended for mounting the modules with Side B as the mounting plane. The HEDS-8905 is used to mount the HEDS-9140, and the HEDS-8906 is used to mount the HEDS-9040. These tools fix the module position using the codewheel hub as a reference. They will not work if Side A is used as the mounting plane.

The following assembly procedure uses the HEDS-8905/8906 alignment tool to mount a HEDS-9140/9040 module and a HEDS-5140/6140 codewheel:

Instructions:

1. Place codewheel on shaft.
2. Set codewheel height by placing alignment tool on motor base (pins facing up) flush up against the codewheel as shown in Figure 3. Tighten codewheel setscrew and remove alignment tool.
3. Insert mounting screws through module and thread into the motor base. Do not tighten screws.
4. Slide alignment tool over codewheel hub and onto module as shown in Figure 4. The pins of the alignment tool should fit snugly inside the alignment recesses of the module.
5. While holding alignment tool in place, tighten screws down to secure module.
6. Remove alignment tool.

Mounting with Aligning Pins

The HEDS-9040 and HEDS-9140 can also be mounted using aligning pins on the motor base. (Hewlett-Packard does not provide aligning pins.) For this configuration, Side A must be used as the mounting plane. The aligning recess centers must be located within the 0.005 in. R Tolerance Circle as explained above. Figure 5 shows the necessary dimensions.

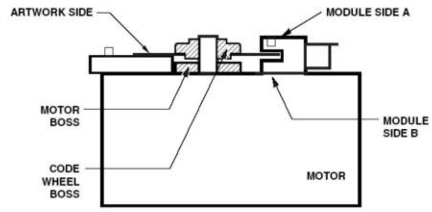


Figure 3. Alignment Tool is Used to Set Height of Codewheel.

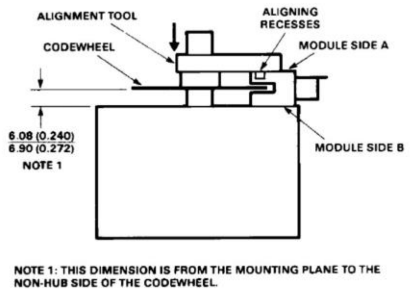


Figure 4. Alignment Tool is Placed over Shaft and onto Codewheel Hub. Alignment Tool Pins Mate with Aligning Recesses on Module.

Mounting with Aligning Pins

The HEDS-9040 and HEDS-9140 can also be mounted using aligning pins on the motor base. (Avago does not provide aligning pins.) For this configuration, Side A *must* be used as the mounting plane. The aligning recess centers must be located within the 0.005 in. Radius Tolerance Circle as explained in "Mounting Considerations." Figure 5 shows the necessary dimensions.

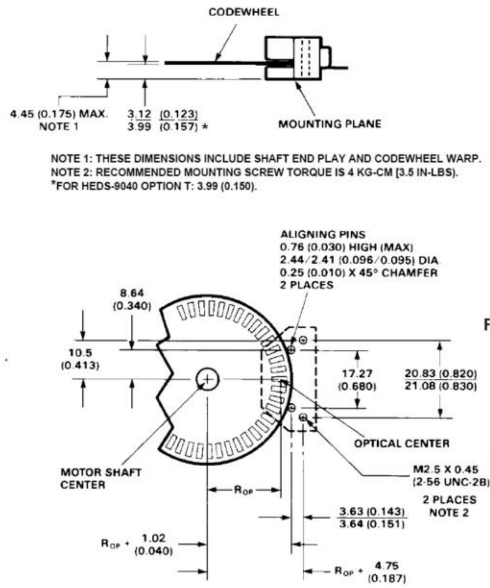


Figure 5. Mounting Plane Side A.

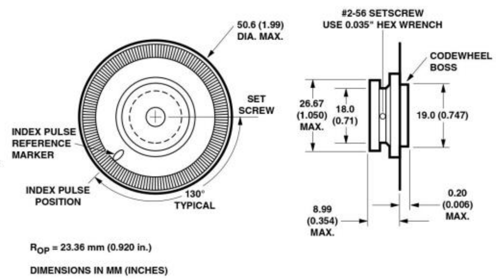


Figure 6a. HEDS-6140 Codewheel Used with HEDS-9040.

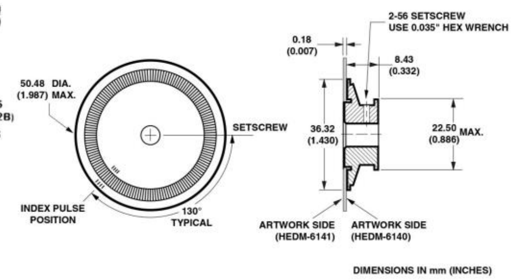


Figure 6b. HEDM-614X Series Codewheel used with HEDS-9040 #T00.

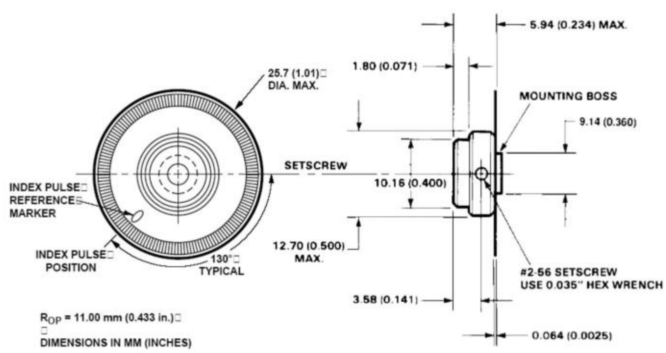
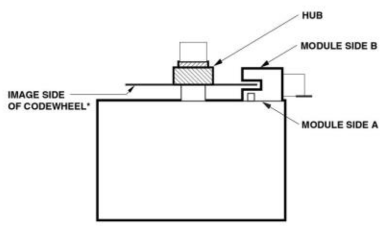


Figure 7. HEDS-5140 Codewheel Used with HEDS-9140.

Orientation of Artwork for HEDS-9040 Option T00 (2000 CPR, 23.36mm Rop) and HEDS-9140 Option B00 (1000CPR, 11.00mm Rop)

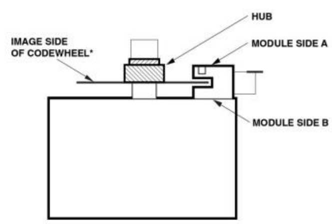
The Index area on the HEDS- 9040 Option T00, 2000 CPR and HEDS-9140 Option B00, 1000 CPR Encoder Module has a nonsymmetrical pattern as does the mating Codewheel. In order for the Index to operate, the "Rightreading" side of the Codewheel disk (the "Artwork Side") must point toward "Side A" of the Module (the side with the connecting pins).

Because the Encoder Module may be used with either "Side A" or with "Side B" toward the Mounting Surface, Avago supplies two versions of Film Codewheels for use with the Option T00 3-channel Module and Option B00 3-Channel Module: Codewheel HEDM-6140 Option TXX and HEDM-5040 Option Bxx has the Artwork Side on the "Hub Side" of the Codewheel/hub assembly and works with "Side B" of the Module on the user's mounting surface. Codewheel HEDM-6141 Option TXX and HEDM-5041-Bxx has the Artwork Side opposite the "Hub Side" and works with "Side A" of the Module on the mounting surface. For the Index to operate, these parts must be oriented as shown in Figure 7a and 7b.



* USE HEDM-6141#Txx or HEDM-5041#Bxx

Figure 7a.



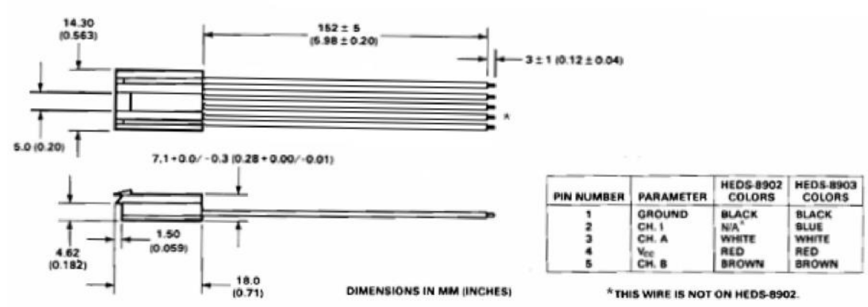
* USE HEDM-6140#Txx or HEDM-5040#Bxx

Figure 7b.

*Please note that the image side of the codewheel must always be facing the module Side A.

Connectors

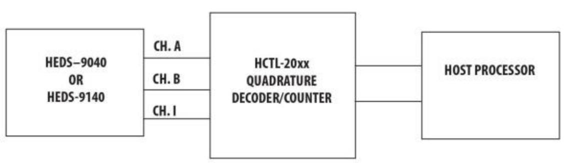
Manufacturer	Part Number
AMP	103686-4
	640442-5
Avago	HEDS-8902 (2 ch.) with 4-wire leads
	HEDS-8903 (3 ch.) with 5-wire leads
Molex	2695 series with 2759 series term.

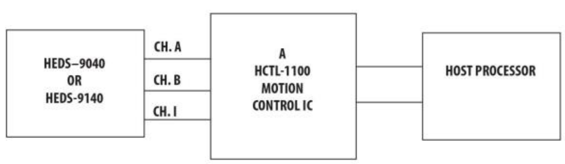


PIN NUMBER	PARAMETER	HEDS-8902 COLORS	HEDS-8903 COLORS
1	GROUND	BLACK	BLACK
2	CH. I	N/A*	BLUE
3	CH. A	WHITE	WHITE
4	V_{CC}	RED	RED
5	CH. B	BROWN	BROWN

Figure 8. HEDS-8903 Connector.

Typical Interfaces





Ordering Information

Three Channel Encoder Modules and Codewheels, 23.36 mm Optical Radius.

HEDS-904 [0] **Option** [ ] **0 0**

Lead Bend
- 0 - Straight Leads
- 1 - Bent Leads

HEDS-6140 Option [ ] [ ] [ ]

Resolution (Cycles/Rev)
- B - 1000 CPR
- J - 1024 CPR

Shaft Diameter
- 06 - 1/4 in.
- 08 - 3/8 in.
- 09 - 1/2 in.
- 10 - 5/8 in.
- 11 - 4 mm
- 12 - 6 mm
- 13 - 8 mm

Assembly Tool

HEDS-8906

Three Channel Encoder Modules and Codewheels, 23.36 mm Optical Radius

HEDS-9040 Option [ ] **0 0** **HEDM-614** [ ] **Option** [ ] [ ] [ ]

Resolution (Cycles/Rev)
- T - 2000 CPR

Artwork Orientation
- 0 - Artwork on hub side (use when module Side B is down)
- 1 - Artwork opposite hub side (use when Module Side A is down)

Shaft Diameter
- 12 - 6 mm

Assembly Tool

HEDS-8906

	A	B	C	D	E	F	G	H	I	J	K	S	T	U
HEDS-9040	*									*			*	
HEDS-9041	*													

		01	02	03	04	05	06	08	09	10	11	12	13	14
HEDS-6140	B							*	*	*	*	*	*	*
	J							*		*			*	*
HEDM-6140	T												*	

Three Channel Encoder Modules and Codewheels, 11.00 mm Optical Radius

HED [ ] -914 [ ] Option [ ] 0 0 HEDS-5140 Option [ ] [ ] [ ]

Assembly Tool

HEDS-8905

Window Option
S – Standard
W– Window Option

Lead Bend
0 - Straight Leads
1 - Bent Leads

Resolution (Cycles/Rev)
S - 50 CPR
C - 100 CPR
E - 200 CPR
F - 256 CPR
G - 360 CPR
H - 400 CPR
A - 500 CPR
I - 512 CPR

Shaft Diameter
02 - 3 mm
04 - 5/32 in.
05 - 3/16 in.
06 - 1/4 in.
11 - 4 mm
14 - 5 mm
12 - 6 mm
13 - 8 mm

Three Channel Encoder Modules and Codewheels, 11.000 Optical Radius

H E D S - 9 1 4 0 Option [ ] 0 0 H E D M - 5 0 4 [ ] Option [ ] [ ] [ ]

Assembly Tool

HEDS-8905

Resolution (Cycle/Rev)
B - 1000CPR

Artwork Orientation
0 - Artwork on Hub Side (use when module side B is down)
1 - Artwork on opposite hub side (use when module side A is down)

Shaft Diameter
01 - 2mm
02 - 3mm
05 - 3/16 in
11 - 4mm
13 - 8mm
14 - 5mm

	A	B	C	D	E	F	G	H	I	J	K	S	T	U
HEDS-9140	*	*	*		*	*	*	*	*		*			
HEDS-9141	*				*	*	*							
HEDW-9140	*								*					

		01	02	03	04	05	06	08	09	10	11	12	13	14
HEDS-5140	A		*		*	*	*				*	*	*	*
	C				*		*					*	*	
	E						*				*	*		*
	F				*							*		*
	G						*					*		*
	I		*		*		*				*	*	*	*
HEDM-5040	B	*	*			*					*		*	*
HEDM-5041	B	*	*			*					*		*	*

For product information and a complete list of distributors, please go to our website: **www.avagotech.com**

AV02-1132EN - April 5, 2011

Avago TECHNOLOGIES

SERVO DUTY

SERIES M15

Dynapar™ brand

For Stepper & Small Servo Motors

Key Features

- Modular Encoder with Easy Installation Requiring No Special Gapping Tools or Parts
- Phased Array Sensor Technology Allowing .030" Axial Shaft Play
- Wide -20 to 120C Operating Temperature Range

RoHS

SPECIFICATIONS

STANDARD OPERATING CHARACTERISTICS

Code: Incremental
Resolution: (pulses/revolution)
Incremental: 200 to 1024 PPR;
Commutation: 4, 6, or 8 pole
Accuracy:
Incremental: ±5 arc-mins. max. edge to edge;
Commutation: ±6 arc-mins. max.
Sense: (viewing encoder mounting surface)
Incremental: A leads B by 90° for CCW rotation of motor shaft;
Commutation: U leads V, V leads W by 120° for CW rotation of motor shaft
Phasing:
Incremental: 90° ±18° electrical
Commutation: 8 Pole: 30°; 6 Pole: 40°; 4 Pole: 60° mechanical
Index to U Channel: ±1° mechanical - Index center to U channel edge
Symmetry:
Incremental: 180° ±18° electrical
Commutation: 8 Pole: 45°; 6 Pole: 60°; 4 Pole: 90° mechanical
Index Pulse Width: 180° ±36° electrical (Gated with B low) standard

ELECTRICAL

Input Power Requirements:
Incremental: 5 or 12 VDC ±10% at 100 mA max. (excluding output load);
Incremental w/Commutation: 5 or 12 VDC ±10% at 120 mA max. (excluding output load)
Output Signals:
7272 Line Driver: 40 mA sink/source max.;
Open Collector w/2.0 kΩ pull-ups: 16 mA sink max.
Frequency Response: 200 kHz min.
Termination:
Connector: PCB mounted dual row head with 0.1" x 0.1" pin spacing, 10 pins (incremental only), 14 pins (w/commutation);
Cable: conductors - 28 AWG, stranded (7/36), insulation - black, PVC; Shield: aluminum/polyester foil plus tinned, copper drain wire (28 AWG, 7/36)
Noise Immunity: Conforms to EN50082-1 Light Industrial for Electro-Static Discharge, Radio Frequency Interference, Electrical Fast Transients, and Magnetic Fields (for models or applications with shielded cable)

MECHANICAL

Weight:
Connector: 0.8 oz. (23 gm) typ.
Connector w/cover: 1.0 oz. (28 gm) typ.
Cable: 1.3 oz (37 gm) typ.
Cable w/cover: 1.5 oz. (43 gm) typ.
Dimensions:
Outside Diameter: 1.60" (40.7 mm) max. w/cover, 1.50" (38.2 mm) max. without cover;
Height: 1.27" (32.3 mm) max. (w/cover, excluding connector);
Emitter to Detector Gap: 0.070" (1.8 mm) min.
Material:
Base, Housing, & Cover: high temperature, glass filled polymer;
Hub: Aluminum; Disk: 0.030" thick glass
Finish:
Base & Housing: black;
Cover: RAL 7010 (dark grey)
Moment of Inertia: 3.40×10^{-5} in-oz sec.^{2} (2.4 gm-cm^2)
Hub Diameters: 1/8", 1/4", 3/8", 3/16", 6 mm, 8 mm, 10 mm nominal
Hub Dia. Tolerance: +0.001"/-0.000" (+0.026 mm/-0.000 mm)
Mating Shaft Length: 0.45" (12 mm) min.; 0.85" (22 mm) max. inside cover
Mating Shaft Runout: 0.002" (0.05 mm) max. (Includes shaft perpendicularity to mounting surface)
Mating Shaft Endplay: +0.015"/-0.015" (+0.38 mm/-0.38 mm) nominal ("+" indicates away from mounting face)
Mounting:
Base: (2) #4-40 (M2.5) #1 Phillips fillister head cap screw on 1.812" (46 mm) B.C., or (2) #2-56 (M2.0) hex socket cap screw on 1.28" (32.5 mm) B.C.; 0.01" (0.254 mm) true position to shaft.
Shaft: split hub w/collar clamp, #2-56 hex socket cap screw (5/64" hex wrench included)
Electrical/Mechanical Alignment Range: ±15° mechanical
Acceleration: 100,000 rad/sec.^{2} max.
Velocity: 12,000 RPM max.

ENVIRONMENTAL

Operating Temperature: 0° to 120°C
Storage Temperature: -40° to 85°C
Shock: 50 G's for 11 msec duration
Vibration: 2.5 G's at 5 to 2000 Hz
Relative Humidity: 90% non-condensing
Enclosure Rating: NEMA 1 / IP40 dirt-tight (for models with cover)

DYNAPAR™
INNOVATION - CUSTOMIZATION - DELIVERY

Satellite Locations:
- **North America:** North Carolina, South Carolina, Connecticut, Massachusetts, New York, Canada, British Virgin Islands
- **West Indies:** St. Kitts • **Europe:** United Kingdom, Italy, France, Germany, Spain, Slovakia
- **South America:** Brazil • **Asia:** China, Japan, Korea, Singapore

Worldwide Brands: NorthStar™ • Acuro™ • Dynapar™ • Hengstler™ • Harowe™
Headquarters: 1675 Delany Road • Gurnee,,IL 60031-1282 • USA • Phone: 1.847.662.2666 • Fax: 1.847.662.6633

Customer Service:
Tel.: +1.800.873.8731
Fax: +1.847.662.4150
custserv@dynapar.com

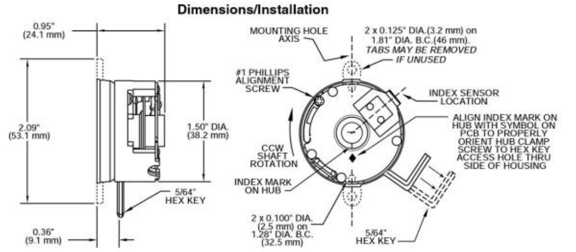
Technical Support
Tel.: +1.800.234.8731
Fax: +1.847.782.5277
dynapar.techsupport@dynapar.com

Document No. 702164-0002, Rev.– ©2010 Dynapar

SERVO DUTY

Dynapar™ brand — SERIES M15

Dimensions/Installation

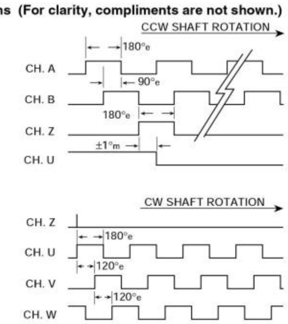


Installation Instructions:

Incremental only models: Drawing #200638-0001

Commutation models: Drawing #200638-0002

Output Waveforms (For clarity, compliments are not shown.)

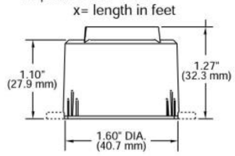


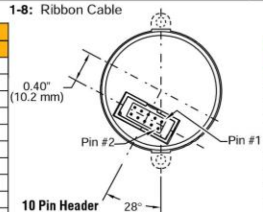
Code 6: Terminations (Not all signals present on all models)

0: Pin Header **1-8:** Ribbon Cable **A-H:** Shielded Cable

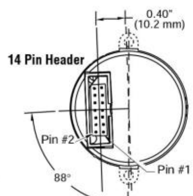
Pin	10 Pin O.C.	10 Pin L.D.	14 Pin
1	A	—	Vcc
2	Vcc	Vcc	U
3	GND	GND	U'
4	—	—	V
5	—	A'	V'
6	—	A	W
7	—	B'	W'
8	B	B	A'
9	—	Z'	A
10	Z	Z	B
11	—	—	B'
12	—	—	Z
13	—	—	GND
14	—	—	Z'

Mating Cable Assembly:
10 pin, 109524-000x
14 pin, 110527-000x
x= length in feet



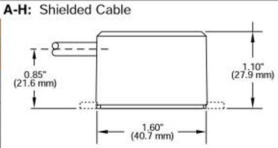


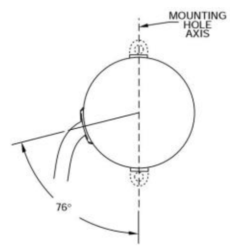
10 Pin Header



14 Pin Header

Function	Wire Color: Incr. Only	Wire Color: Incr. & Comm.
Vcc com	—	RED/WHT
Vcc Inc	RED	RED
GND Inc	BLK	BLK
GND com	—	BLK/WHT
A'	RED/BLK	BLU/BLK
A	GRN	BLU
B'	WHT/BLK	GRN/BLK
B	ORN	GRN
Z'	BLU	VIO/BLK
Z	WHT	VIO
U'	—	BRN/BLK
U	—	BRN
V'	—	GRY/BLK
V	—	GRY
W'	—	WHT/BLK
W	—	WHT





Ordering Information

To order, complete the model number with code numbers from the table below:

Code 1: Model	Code 2: PPR, Poles	Code 3: Cover	Code 4: Electrical	Code 5: Hub	Code 6: Termination
M15	□□□□/□	□	□	□	□

Ordering Information

Code 1: Model	Code 2: PPR, Poles	Code 3: Cover	Code 4: Electrical	Code 5: Hub	Code 6: Termination
M15 Size 15 Commutating Modular	Incremental channels only: **0200/0**, **1000/0**, **0400/0**, **1024/0**, **0500/0** Incremental plus Commutation channels: **0500/6**, **1024/4**, **1000/4**, **1024/6**, **1000/6**, **1024/8**, **1000/8**	**0** No cover **1** Enclosed, end-of-shaft mount **2** Through shaft	**0** 5V in, open collector out incremental only **1** 12V in, open collector out incremental only **3** 5V in, line driver out incremental only Available when Code 2 is XXXX/4, XXXX/6, or XXXX/8 **6** 5V in, line driver out incr.; 5V in, open collector out comm. **7** 5V in, line driver out incr.; 12V in, open collector out comm. **9** 5V in, line driver out incr.; 5V in, line driver out comm.	**0** 1/4 in. **1** 3/8 in. **4** 6 mm **5** 8 mm **6** 10 mm **8** 3/16 in. **9** 1/8 in.	Available when Code 4= 0,1,3,6 or 9 **0** Pin Header **1-8** Mating ribbon cable included; 1=1 ft., 2=2 ft., etc. Available when Code 4= 0 – 9 **A-H** Shielded cable; A=1 ft., B=2 ft., etc.

SERVO DUTY

SERIES M53

Dynapar™ brand

For Stepper & Small Servo Motors

NEW!

Key Features

- **2.0" Diameter Modular Encoder with Easy Installation Requiring No Special Gapping Tools or Parts**
- **Phased Array Sensor Technology Allowing .020" Axial Shaft Play**
- **Up to 2048 PPR with Commutation Tracks**

RoHS

SPECIFICATIONS

STANDARD OPERATING CHARACTERISTICS

Code: Incremental
Resolution: (pulses/revolution)
Incremental: 500 to 2048 PPR
Commutation: 4, 6 or 8 pole
Accuracy:
Incremental: ±5 arc-mins. max. edge to edge;
Sense: (viewing encoder mounting surface)
Incremental: A leads B by 90° for CCW rotation of motor shaft;
Commutation: U leads V, V leads W by 120° for CW rotation of motor shaft
Phasing:
Incremental: 90° ±18° electrical
Commutation: 8 Pole: 30°; 6 Pole: 40°; 4 Pole: 60° mechanical
Index to U Channel: ±1° mechanical - Index center to U channel edge
Symmetry:
Incremental: 180° ±18° electrical
Commutation: 8 Pole: 45°;
6 Pole: 60°; 4 Pole: 90° mechanical
Index Pulse Width: 90° ±36° electrical (Gated with A high and B low)

ELECTRICAL

Input Power Requirements:
Incremental: 5 VDC or 12 VDC ±10% at 100 mA max. (excluding output load);
Commutation: 5 VDC or 12 VDC ±10% at 75 mA max. (excluding output load)
Output Signals:
7272 Line Driver: 40 mA sink/source max.;
Open Collector w/2.0 kΩ pull-ups: 16 mA sink max.
Frequency Response: 200 kHz min.
Termination:
Connector: PCB mounted dual row head with 0.1" x 0.1" pin spacing, 10 pins (incremental only), 16 pins (w/commutation); Cable: conductors - 28 AWG, stranded (7/36), insulation - black, PVC; Shield: aluminum/polyester foil plus tinned, copper drain wire (28 AWG, 7/36)
Noise Immunity: Conforms to EN50082-1 Light Industrial for Electro-Static Discharge, Radio Frequency Interference, Electrical Fast Transients, Conducted Interference, and Magnetic Fields (for models or applications with shielded cable)

MECHANICAL

Weight:
Connector: 1 oz. (28 gm) typ.
Connector w/cover: 1.5 oz. (43 gm) typ.
Cable: 2.5 oz (71 gm) typ.
Cable w/cover: 3 oz. (85 gm) typ.
Dimensions:
Outside Diameter: 2.1" (53 mm) max. w/cover, 2.0" (51 mm) max. without cover; Height: 0.8" (20.3 mm) (w/cover, excluding connector);
Emitter to Detector Gap: 0.070" (1.8 mm) min.
Material:
Base, Housing, & Cover: high temperature, glass filled polymer;
Hub: Aluminum; Disk: 0.030" thick glass
Finish:
Base & Housing: black;
Cover: RAL 7010 (dark grey)
Moment of Inertia: 6.64 x 10^{-5} in-oz sec.^{2} (4.7 gm-cm^2)
Hub Diameters: 1/4", 3/8", 7/16", 1/2", 6 mm, 8 mm, 10 mm, 12 mm nominal
Hub Dia. Tolerance: +0.001"/-0.000" (+0.026 mm/-0.000 mm)
Mating Shaft Length: 0.45" (12 mm) min. blind hub clamp screw, 0.65" (16.5 mm) exposed hub clamp screw; 0.75" (19 mm) max. inside cover
Mating Shaft Runout: 0.002" (0.05 mm) max. (Includes shaft perpendicularity to mounting surface)
Mating Shaft Endplay: +0.011"/-0.008" (+0.30 mm/-0.21 mm) nominal ("+" indicates away from mounting face)
Mounting:
Base: (2) #4-40 (M2.5) #1 Phillips fillister head cap screw on 1.812" (46 mm) B.C., 0.01" (0.254 mm) true position to shaft; Shaft: split hub w/collar clamp, #2-56 hex socket cap screw (5/64" hex wrench included)
Electrical/Mechanical Alignment Range: ±15° mechanical
Acceleration: 100,000 rad/sec.^{2} max.
Velocity: 12,000 RPM max.

ENVIRONMENTAL

Operating Temperature: 0° to 120°C
Storage Temperature: -40° to 85°C
Shock: 50 G's for 11 msec duration
Vibration: 2.5 G's at 5 to 2000 Hz
Relative Humidity: 90% non-condensing
Enclosure Rating: NEMA 1 / IP50 dirt-tight (for models with cover)

Worldwide Brands: NorthStar™ • Acuro™ • Dynapar™ • Hengstler™ • Harowe™

DYNAPAR™
INNOVATION - CUSTOMIZATION - DELIVERY
WWW.DYNAPAR.COM

Headquarters: 1675 Delany Road • Gurnee, IL 60031-1282 • USA

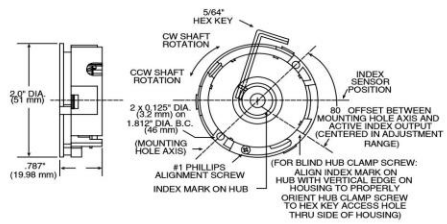
Customer Service:
Tel.: +1.800.873.8731
Fax: +1.847.662.4150
custserv@dynapar.com

Technical Support
Tel.: +1.800.234.8731
Fax: +1.847.662.4150
northstar.techsupport@dynapar.com

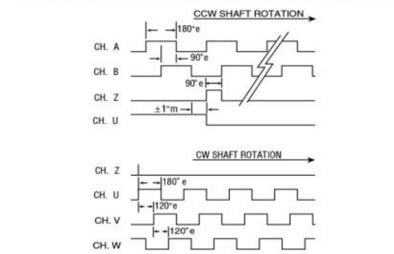
SERVO DUTY

Dynapar™ brand SERIES M53

Dimensions/Installation



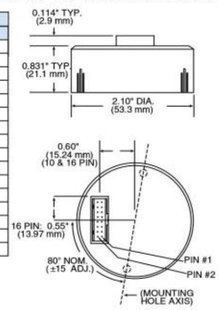
Output Waveforms (For clarity, compliments are not shown.)



Code 6: Terminations (Not all signals present on all models)

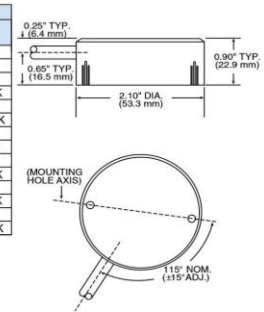
0: 16 Pin JST connector 1-8: Shielded Cable with JST connector

Pin	16 Pin
1	Vcc
2	U
3	GND
4	V
5	A
6	W
7	A'
8	NC
9	B
10	U'
11	B'
12	V'
13	Z
14	W'
15	Z'
16	NC



A-H: Shielded Cable

Function	Wire Color: Incr. Only	Wire Color: Incr. & Comm.
Vcc	RED	RED
GND	BLK	BLK
A	GRN	BLU
A'	RED/BLK	BLU/BLK
B	ORN	GRN
B'	WHT/BLK	GRN/BLK
Z	WHT	VIO
Z'	BLU	VIO/BLK
U	–	BRN
U'	–	BRN/BLK
V	–	GRY
V'	–	GRY/BLK
W	–	WHT
W'	–	WHT/BLK



Ordering Information

To order, complete the model number with code numbers from the table below:

Code 1: Model	Code 2: PPR, Poles	Code 3:Cover	Code 4: Electrical	Code 5:Hub	Code 6:Termination
M53	□□□□/□	□	□	□	□
Ordering Information					
M53 Size 20 Commutating Modular	Incremental channels only **0500/0** **1024/0** **0512/0** **2000/0** **1000/0** **2048/0** Incremental plus Commutation channels **0500/4** **1024/4** **0500/6** **1024/6** **0500/8** **1024/8** **0512/8** **2000/4** **1000/4** **2000/6** **1000/6** **2000/8** **1000/8** **2048/4** **2048/6** **2048/8** **2048/8**	**0** No cover **1** Radial exit cover(for shielded cable) **2** Axial exit(for shielded cable with JST connector)	**0** 5V in, open collector out incremental only **1** 12V in, open collector out incremental only **3** 5V in, line driver out incremental only **A** 12V in, 5V line driver out incremental only **B** 12V in,12V line driver out incremental only Available when Code 2 is XXXX/4, XXXX/6, or XXXX/8 **6** 5V in,line driver out incremental open collector out Comm **9** 5V in,line driver out incremental line driver out Comm **C** 12V in,5Vline driver out incremental,open collector **D** 12V in,12Vline driver out incremental,open collector **E** 12V in,5V line driver out incremental,5Vline driver out Comm out Comm out Comm **F** 12V in,12Vline driver out incremental,12Vline driver out Comm	Exposed hub clamp screw: **A** 1/4 in. **B** 3/8 in. **C** 7/16 in. **D** 1/2 in. **E** 6 mm **F** 8 mm **G** 10 mm **H** 12 mm	**0** JST connector Available when Code 3 is 2: **1-8** Shielded cable with connector; 1=1 ft., 2=2 ft., etc. Available when Code 3 is 1: **A-H** Shielded cable; A=1 ft., B=2 ft., etc.

FB Gearing

Harmonic Drive® gear

Precision Gearing and Motion Control

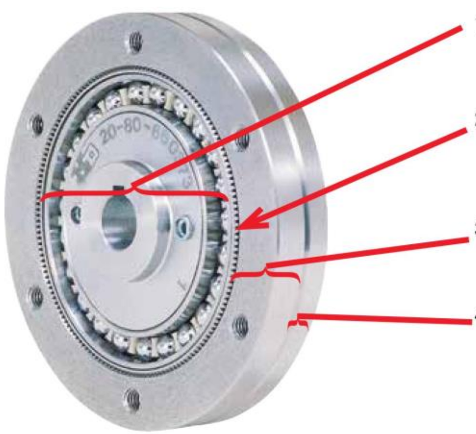
Contents

Compact, High Ratio, In-Line Gearing

Harmonic Drive® FB "Pancake" type component set offers the designer high ratio, in-line mechanical power transmissions in extremely compact configurations. The component set consists of four elements: the Wave generator, an elliptical bearing assembly; the Flexspline, a non-rigid ring with external teeth; and the Circular Spline and the Dynamic Spline, rigid internal gears.

Rotation of the Wave Generator imparts a rotating elliptical shape to the Flexspline causing progressive engagement of its external teeth with the internal teeth of the Circular Spline and the Dynamic Spline. The fixed Circular Spline has two more teeth than the Flexspline, thereby imparting relative rotation to the Flexspline at a reduction ratio corresponding to the difference in the number of teeth. With the same number of teeth, the Dynamic Spline rotates with and at the same speed as the Flexspline.

The Basic Component Set



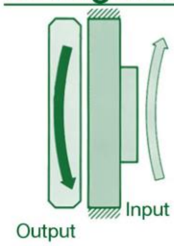
1) The Wave generator (WG) is a thin raced bearings assembly fitted onto an elliptical plug, and normally is the rotating input member.

2) The Flexspline (FS) is a non-rigid ring with external teeth on a slightly smaller pitch diameter than the Circular Spline. It is fitted over and is elastically deflected by the Wave Generator.

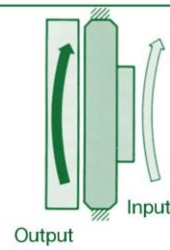
3) The Circular Spline (CS) is a rigid ring with internal teeth, engaging the teeth of the Flexspline across the major axis of the Wave Generator.

4) The Dynamic Spline (DS) is a rigid ring having internal teeth of same number as the Flexspline. It rotates together with the Flexspline and serves as the output member. It is identified by chamfered corners at its outside diameter.

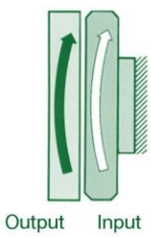
Configurations



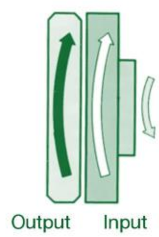
1) Reduction Gearing
WG Input
CS Fixed
DS Output
Ratio as listed
Input and output in opposite direction.



2) Reduction Gearing
WG Input
CS Output
DS Fixed
Ratio $\frac{1}{R+1}$
Input and output in same direction.



3) Reduction Gearing
WG Fixed
CS Output
DS Input
Ratio $\frac{R}{R+1}$
Input and output in same direction.



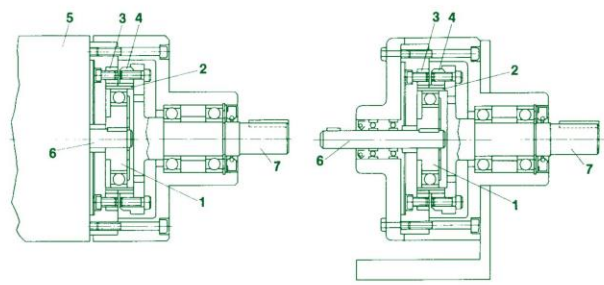
4) Differential
WG Control Input
CS Main-drive Input
DS Main-drive Output
Numerous differential functions can be obtained by combinations of speeds and rotations on the three shafts.

Typical Installation

FB "pancake" type component sets are easier to use than conventional gearing. All that is required is suitable bearing support for the input and output shaft, and a means of fixing the circular spline against rotation.

The simplicity of FB component sets is demonstrated in the typical arrangements shown below.

1. Wave Generator
2. Flexspline
3. Circular Spline
4. Dynamic Spline
5. Motor
6. Input Shaft or Motor Shaft
7. Output Shaft

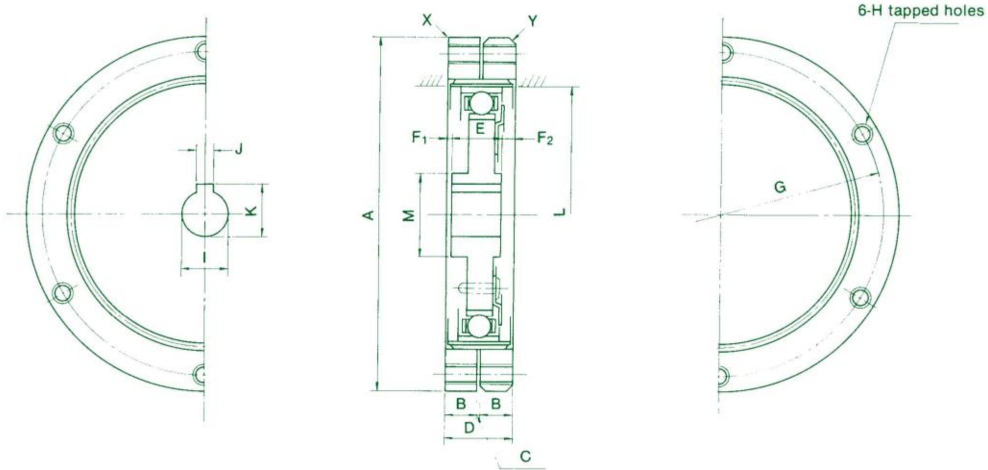


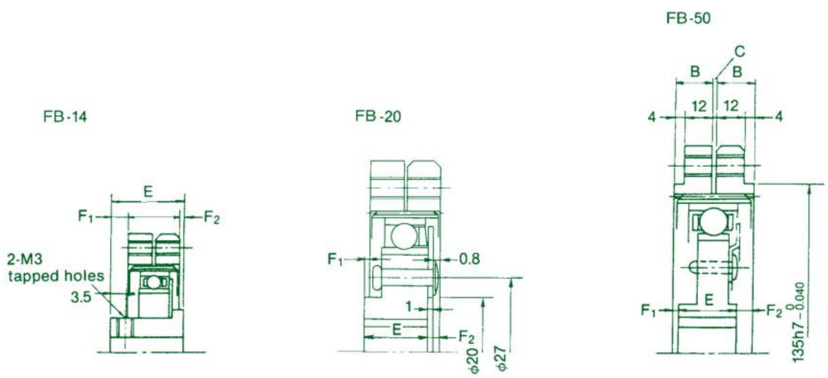
Ordering Information

FB 20-80-2G-R-SP

- FB — Pancake model with the single row bearing Wave Generator
- 20 — Size 20
- 80 — Reduction ratio 80:1
- 2G — Component set
- R — Backlash optimized to below 3 arc minutes (Optional)
- SP — Suffix indicating that the set is specially modified or designed according to customer requirements.

Dimensions





FB	A (g7)	B	C	D	E	F1	F2	G	H	I (H7)	I Max	J (JS9)	K	L	M	N	X	Y	Wt lb	Wt kgf
14	$50^{-0.009}_{-0.034}$	5	0.5	10.5	15.0	3.75	0.75	44	M3	$6^{+0.012}_{0}$	8	—	—	29	14	—	0.2	1.0	0.2	0.1
20	$70^{-0.010}_{-0.040}$	6	0.5	12.5	11.4	0.95	2.05	60	M4	$9^{+0.015}_{0}$	12	$3_{\pm 0.0125}$	10.4	42	20	—	0.2	1.0	0.2	1.0
25	$85^{-0.012}_{-0.047}$	8	0.5	16.5	12.8	0.35	3.35	75	M5	$14^{+0.018}_{0}$	15	$5_{\pm 0.0150}$	16.3	53	26	0.9	0.2	1.5	1.1	0.5
32	$110^{-0.012}_{-0.047}$	10	0.5	20.5	15.6	0.95	3.95	100	M6	$14^{+0.018}_{0}$	15	$5_{\pm 0.0150}$	16.3	69	26	0.8	0.2	1.5	2.2	1.0
40	$135^{-0.014}_{-0.054}$	13	1	27.0	19.4	1.80	5.80	120	M8	$14^{+0.018}_{0}$	20	$5_{\pm 0.0150}$	16.3	84	32	1.2	0.4	2.0	4.0	1.8
50	$170^{-0.014}_{-0.054}$	16	1	33.0	23.2	2.90	6.90	150	M10	$19^{+0.021}_{0}$	20	$6_{\pm 0.0150}$	21.8	105	32	1.1	0.4	2.0	6.4	2.9

Maximum housing I.D. for Flexspline axial containment is L. The surface hardness in the region where the Flexspline abuts the housing is recommended to be HRC 29–34.

Performance Ratings

FB	Gear Ratio	Rated Input Rotational Speed	Rated Torque at 2000rpm		Repeated Peak Torque		Max. Average Load Torque		Max. Momentary Torque		Max. Input Speed rpm		Limit for Average Input Speed, rpm		Moment of Inertia**		Backlash*** arc min.	
		rpm	N.m	In.lb	N.m	In.lb	N.m	In.lb	N.m	In.lb	Oil Lub.	Grease Lub.	Oil Lub.	Grease Lub.	kg-cm²	lb-in²	Optimized	Non-Opt.
14	50	2000	2.6	23	3.2	28	3.2	28	6.9	61	6000	3600	4000	2500	0.033	0.011	3	32
	88		4.9	43	7.8	69	7.8	69	15.7	139*								
	100		5.9	52	9.8	87	9.8	87	15.7	139*								
	110		5.9	52	9.8	87	9.8	87	15.7	139*								
20	50	2000	14	124	18	159	18	159	34	301	6000	3600	3600	2500	0.14	0.048	3	32
	80		17	150	21	186	21	186	35	310								
	100		22	195	26	230	25	221	47	416								
	128		24	212	33	292	25	221	58	513								
	160		24	212	38	336	25	221	59	522*								
25	50	2000	23	204	30	266	30	266	54	478	5000	3600	3000	2500	0.36	0.12	3	30
	80		31	274	39	345	39	345	70	620								
	100		39	345	52	460	52	460	91	805								
	120		39	345	61	540	61	540	94	832*								
	160		39	345	76	673	61	540	86	761*								
32	50	2000	44	389	60	531	60	531	108	956	4500	3600	2500	2300	1.3	0.44	3	24
	78		63	558	75	664	75	664	127	1124								
	100		82	726	98	867	98	867	176	1558								
	131		82	726	137	1212	118	1044	235	2080*								
	157		82	726	157	1389	118	1044	235	2080*								
40	50	2000	88	779	118	1044	118	1044	216	1912	4000	3300	2000	2000	3.4	1.2	3	24
	80		118	1044	147	1301	147	1301	265	2345								
	100		157	1389	186	1646	186	1646	343	3036								
	128		167	1478	235	2080	235	2080	372	3292*								
	160		167	1478	284	2513	274	2425	353	3124*								

* Torque value limited by "Ratceting".
** Moment of Inertia: 1=–1/4 GD^2.
*** Backlash measured at output with the input locked, maximum value.

Lubrication

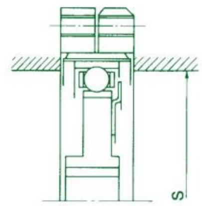
Oil lubrication ratings are based on Molub Alloy gear Oil No. 80. See table for recommended oil level and volume for horizontal shaft mounting.

For vertical mounting the recommended level is at the wave generator bearing ball centerline or midpoint of the drive.

FB		14	20	25	32	40	50
Oil Level Below Drive Centerline	mm	7.6	12.7	15.2	17.8	23.0	30.5

Grease lubricated ratings are based on Harmonic Grease SK-1A for size 12 to 100, and SK-2 for size 14. Alternate lubricants include Molub Alloy Grease No. 2, Shell Alvania EP 1 and their equivalents.

For retention of grease within the tooth mesh area and the ball bearing, it is recommended that the L dimension (see FB Dimensions, page 4) be extended further inward to at least S.

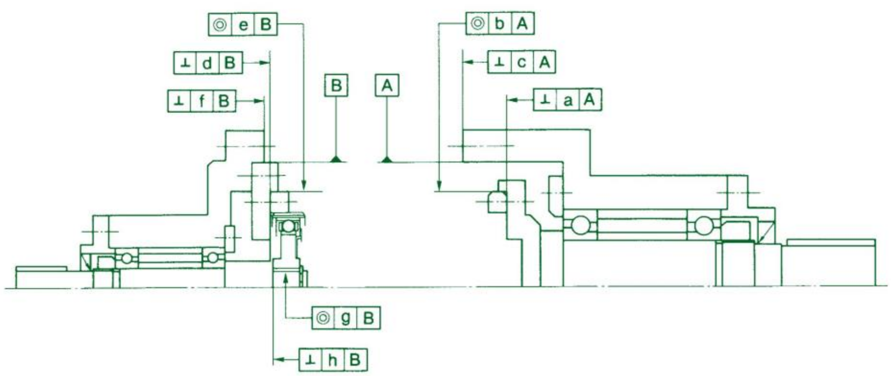


mm

FB	14	20	25	32	40	50
S	26	38	48	63	76	95

Installation

The Dynamic Spline is distinguished by its chamfered outer edge. FB Component Sets may be operated in any attitude. Recommended installed relationships are shown below:



Housing Tolerance

mm

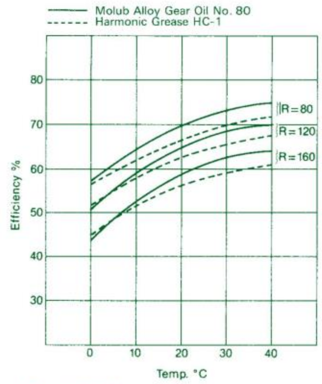
FB	a	b	c	d	e	f	g	h
14	0.013	0.015	0.016	0.013	0.015	0.016	0.011	0.007
20	0.017	0.016	0.020	0.017	0.016	0.020	0.013	0.010
25	0.024	0.016	0.029	0.024	0.016	0.029	0.016	0.012
32	0.026	0.017	0.031	0.026	0.017	0.031	0.016	0.012
40	0.026	0.019	0.031	0.026	0.019	0.031	0.017	0.012
50	0.028	0.024	0.034	0.028	0.024	0.034	0.021	0.015

Efficiency

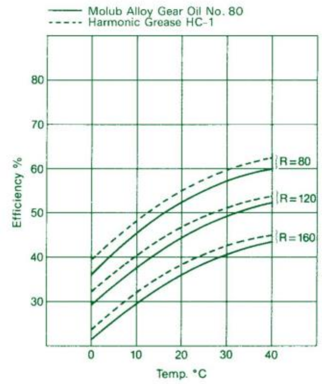
Efficiency varies depending on input speed, ratio, load level, temperature, and type of lubrication. The effects of these factors are illustrated in the curves shown below.

FB Efficiency vs. Ratio, Temperature, and Lubricant (At Rated Torque)

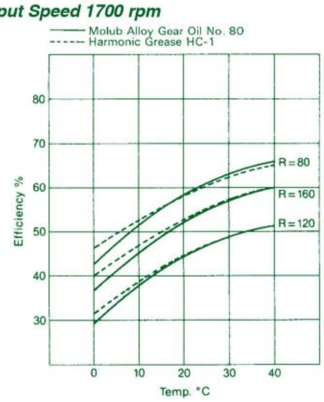
Input Speed 500 rpm



Input Speed 3400 rpm



Input Speed 1700 rpm



No-Load Running Torque, Starting Torque, and Backdriving Torque

FB		14	20	25	32	40	50
NL Running Torque @ 1500 rpm	Ncm	3~8	5~11	6~30	15~40	20~65	60~150
	oz-in	4~11	7~15	8~42	20~56	28~90	83~210
Starting Torque	Ncm	0.5~3	0.8~4	2~7	3~10	5~30	10~60
	oz-in	0.7~4	1~6	3~10	4~14	7~42	14~83
Backdriving Torque	Nm	0.8~7	2~10	3~38	4~40	8~60	20~110
	lb-in	6~60	17~87	26~330	35~350	70~520	170~950

Values quoted are based on actual tests with the component sets assembled in housings, and takes into consideration friction resistance of oils seals, and churning of oil.

Harmonic Drive LLC

Boston US Headquarters
247 Lynnfield Street
Peabody, MA 01960

New York Sales Office
100 Motor Parkway
Suite 116
Hauppauge, NY 11788

California Sales Office
333 W. San Carlos Street
Suite 1070
San Jose, CA 95110

Chicago Sales Office
137 N. Oak Park Ave., Suite 410
Oak Park, IL 60301

T: 800.921.3332
T: 978.532.1800
F: 978.532.9406

www.HarmonicDrive.net

Group Companies
Harmonic Drive Systems, Inc.
6-25-3 Minami-Ohi, Shinagawa-ku
Tokyo 141-0013, Japan

Harmonic Drive AG
Hoenbergstrasse, 14, D-6555
Limburg/Lahn Germany

Rev 7-14

10-25 oz-in Continuous Torque

Key Performance Features

- CE Compliant
- High Energy Neodymium Magnets
- Peak Torque to 150 oz-in
- Excellent Torque to Weight Ratio
- 1.5" Motor Diameter is Ideal for Restricted Space Applications
- High RPM Operation
- ISO 9001:2001‡

S15 BRUSHED SERVO MOTOR SERIES

Motor Characteristics

FRAME SIZE – STACK LENGTH	Peak Stall Torque (Tp) oz-in	Cont. Stall Torque (Tc) oz-in	Rotor Inertia (Jm) oz-in-sec²	Friction Torque (Tf) oz-in	Thermal Resistance (Rth) degC/Watt	Max Recommend Speed RPM	Max Winding Temp C°	Elect Time Constant (Te) msec	Length (L) in.	Weight (W) lb.
S15 – 100	75	10	.00027	2	6.5	6,000	155	.25	2.5	.6
S15 – 150	120	15	.0004	2.5	4.7	6,000	155	.25	3.0	.82
S15 – 250	150	25	.0006	3	3.0	6,000	155	.25	4.0	1.2

Sample Windings

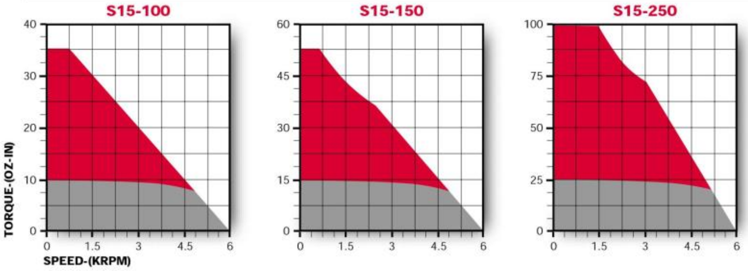
	S15-100				S15-150				S15-250			
	I	K	M	O	I	K	M	O	I	K	M	O
Torque Constant (Kt) oz-in/amp	1.2	1.9	3.0	4.5	1.8	2.9	4.6	7.3	3.0	4.8	7.8	12.0
Voltage Constant (Ke) Volts/Krpm	0.9	1.4	2.3	3.3	1.4	2.2	3.4	5.4	2.2	3.5	5.6	8.9
Arm. Resistance (Ra) Ohms (cold)	.21	0.5	1.3	2.4	0.2	0.6	1.5	3.8	.31	0.8	2.0	5.0
Peak Current (A) Amps	55	35	22	14	60	37	24	15	46	29	18	12
Cont. Current (A) Amps	7.8	5.0	3.2	2.1	7.8	5.0	3.2	2.1	7.8	5.0	3.2	2.1

MAGMOTOR FACT

For Application Specific Motor Solutions, Call or Fax Us Your Specs Today!

CONSULT MAGMOTOR APPLICATION STAFF FOR OTHER AVAILABLE WINDINGS

Performance Curves

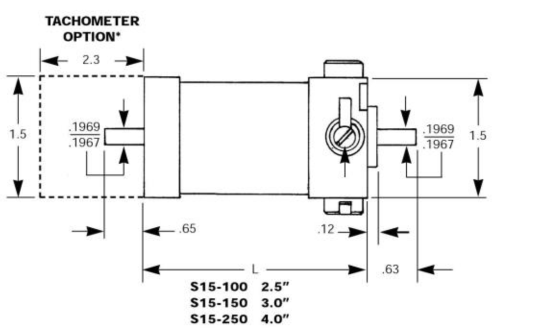


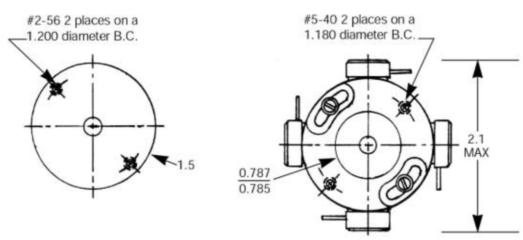
CONTINUOUS DUTY   INTERMITTENT DUTY   TYPICAL PERFORMANCE @ 25°C AMBIENT

MAGMOTOR FACT

Magmotor has been Designing and Manufacturing Motors, in the USA, Since 1976!

Magmotor
A SatCon Company





***Tachometer Specifications**

See Ordering Guide on Page 23 for Available Tach Voltages

Ripple Voltage: 3% max. P-P

†Add 1″ to Motor Length. Reference Page 24 for Additional Encoder Details.

S15 Series Options

- Optical Encoders†
- Application Specific Windings and Mechanical Designs
- Custom Cables and Connectors
- Special Shaft and Mounting Configurations

Typical Applications

- X-Y Positioning
- Medical
- Laboratory
- Automated Assembly
- Pharmaceutical
- Office Products

Motor Selection Assistance

- Please Refer to Page 23 for Application Assistance and Model Selection and Call Us:
  Tel: 508-929-1400
  Fax: 508-929-1401
  Toll Free Solutions Line: 86-Magmotor
  www. magmotor.com

Magmotor™

50-100 oz-in Continuous Torque

Key Performance Features:

- High Energy Neodymium Magnets
- Peak Torque to 1000 oz-in
- Excellent Torque to Weight Ratio
- Encoder Ready
- 12-120 VDC Typical

S23 BRUSHED SERVO MOTOR SERIES

Motor Characteristics

FRAME SIZE	STACK LENGTH	PEAK STALL TORQUE (T_P) OZ-IN	CONT. STALL TORQUE (T_C) OZ-IN	ROTOR INERTIA (J_M) OZ-IN-SEC²	FRICTION TORQUE (T_F) OZ-IN	THERMAL RESISTANCE (RM) °C/WATT	MAX RECOMMEND SPEED RPM	MAX WINDING TEMP. C°	POWER RANGE W	WEIGHT LB
S23	-- 100	500	50	0.006	5	4.5	4000	155	70	2
S23	-- 200	800	75	0.010	6	4.0	4000	155	100	2.6
S23	-- 285	1000	100	0.014	7	3.5	4000	155	130	3.2

Sample Windings

CONSULT MAGMOTOR APPLICATION STAFF FOR OTHER AVAILABLE WINDINGS

	S23 -- 100				S23 -- 200				S23 -- 285			
	E	G	I	K	F	H	J	M	E	G	I	K
Torque Constant (Kt) oz-in/amp	5.1	8.0	13.0	20.1	12.8	19.3	30.4	60.8	14.4	22.7	36.2	56.3
Voltage Constant (Ke) Volts/Krpm	3.7	6.0	9.6	14.9	9.4	14.3	22.5	45.0	0.6	16.8	26.8	41.5
Term. Resistance (Rt) Ohms (cold)	0.5	0.7	1.45	3.15	0.65	1.35	2.86	10.32	0.5	1.9	2.5	5.5
Peak Current (A) Amps	75	48	24	15	75	48	24	15	75	48	24	15
Cont. Current (A) Amps	7.3	4.7	3.0	2.0	7.3	4.7	3.0	2.0	7.3	4.7	3.0	2.0

VALUES AS LISTED ARE TEST CONDITIONS, ACTUAL RESULTS MAY VARY

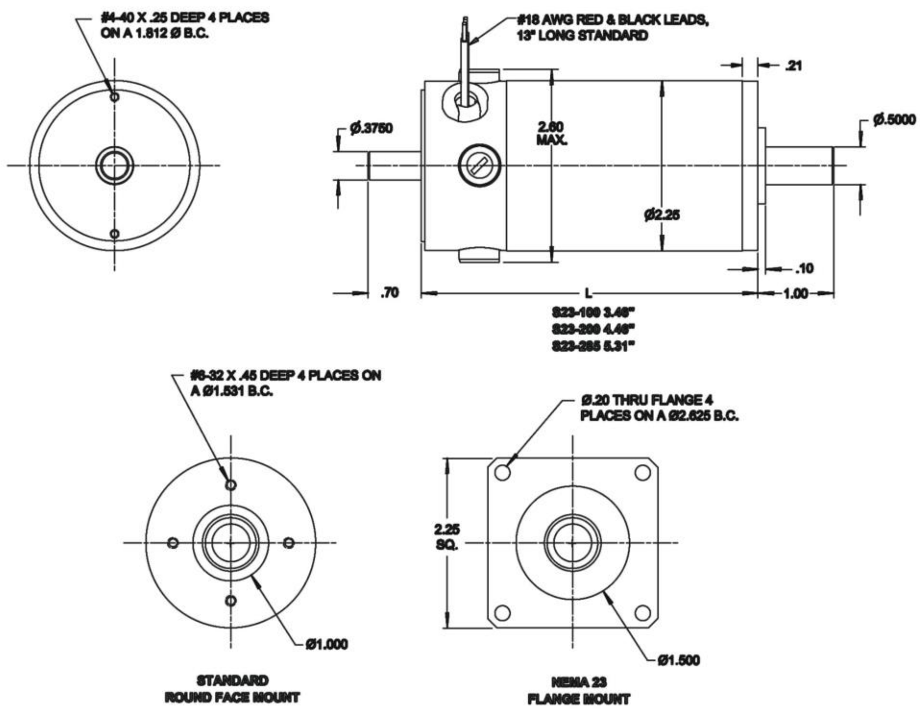
S23 Series Options

- Optical Encoders
- Tachometers and Brakes
- Application Specific Windings and Mechanical designs
- Custom Cables and Connectors
- NEMA 23 Flange Mounting
- For more options, see magmotor. com custom solutions, or call us.

Typical Applications

- Semiconductor Equipment
- Medical Equipment
- Automated Assembly Machines
- Laboratory Equipment
- Pharmaceutical Equipment
- X-Y-Z Positioning Machines

Magmotor™



Magmotor™

80-250 oz-in Continuous Torque

Key Performance Features:

- High Energy Neodymium Magnets
- Peak Torque to 1700 oz-in
- Excellent Torque to Weight Ratio
- Encoder Ready
- Low Voltage High Performance Design is Available
- 12-120 VDC Typical

S28 BRUSHED SERVO MOTOR SERIES

Motor Characteristics

FRAME SIZE -- STACK LENGTH	PEAK STALL TORQUE (T_P) OZ-IN	CONT. STALL TORQUE (T_C) OZ-IN	ROTOR INERTIA (J_M) OZ-IN-SEC²	FRICTION TORQUE (T_F) OZ-IN	THERMAL RESISTANCE (RM) °C/WATT	MAX RECOMMEND SPEED RPM	MAX WINDING TEMP. C°	POWER RANGE W	WEIGHT LB
S28 -- 100	900	80	0.01	6	3.7	4000	155	80	3
S28 -- 200	1300	130	0.02	7	2.9	4000	155	150	4
S28 -- 300	1500	200	0.025	8	2.3	4000	155	200	5.5
S28 -- 400	1700	250	0.04	9	1.8	4000	155	250	6.8

Sample Windings

CONSULT MAGMOTOR APPLICATION STAFF FOR OTHER AVAILABLE WINDINGS

	S28 -- 100				S28 -- 200				S28 -- 300				S28 -- 400			
	E	G	I	K	E	G	I	K	E	G	I	K	E	G	I	K
Torque Constant (Kt) oz-in/amp	8.8	14.1	22.9	36.4	17.5	27.9	44.8	71.1	27.5	40.6	65.6	104.1	34.4	56.1	89.0	134.6
Voltage Constant (Ke) Volts/Krpm	6.50	10.4	17.0	26.9	13.0	20.6	33.1	52.6	19.3	30.0	48.5	77.0	25.4	41.5	65.8	99.5
Term. Resistance (Rt) Ohms (cold)	0.5	0.9	1.3	3.4	0.8	2.0	3.4	7.0	0.8	1.4	4.5	9.8	0.83	2.2	5.5	12.2
Peak Current (A) Amps	60	36	23	15	60	36	23	15	60	36	23	15	60	36	23	15
Cont. Current (A) Amps	9	5	3	2	9	5	3	2	9	5	3	2	9	5	3	2

VALUES AS LISTED ARE TEST CONDITIONS, ACTUAL RESULTS MAY VARY

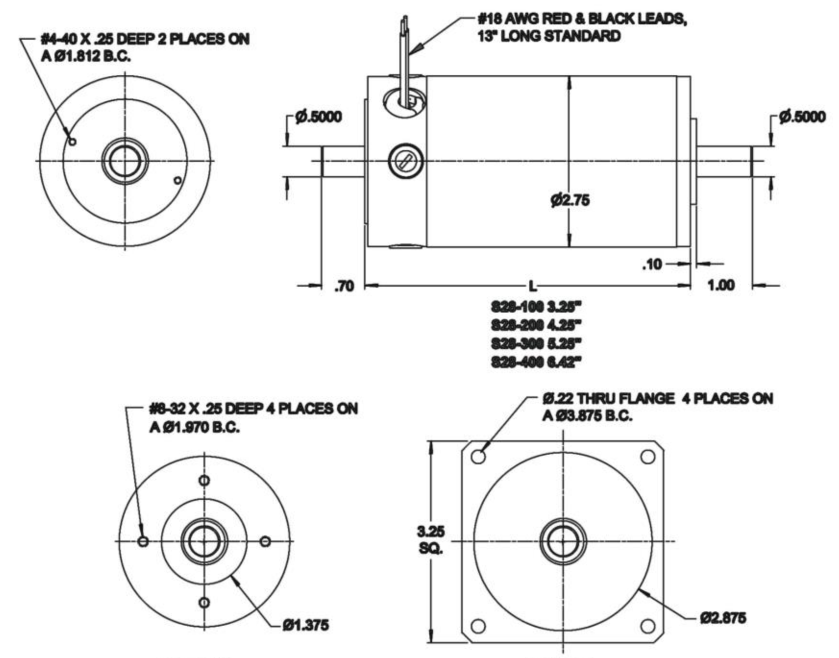
S28 Series Options

- Optical Encoders
- Tachometers, Brakes and Gear Boxes
- Custom Cables and Connectors
- IP 65 Sealing
- NEMA 34 Flange Mounting
- For more options, see magmotor.com custom solutions, or call us.

Typical Applications

- Packaging Machines
- Machine Tools
- Coil Winders
- Electric Vehicles
- Material Handling Equipment
- Textile Machines

Magmotor™



OPTION-
ROUND FACE MOUNT

NEMA 34
FLANGE MOUNT

VNH2SP30-E

Automotive fully integrated H-bridge motor driver

Features

Type	$R_{DS(on)}$	I_{out}	V_{ccmax}
VNH2SP30-E	19mΩ max (per leg)	30A	41V

- 5V logic level compatible inputs
- Undervoltage and overvoltage shut-down
- Overvoltage clamp
- Thermal shut down
- Cross-conduction protection
- Linear current limiter
- Very low stand-by power consumption
- PWM operation up to 20 kHz
- Protection against loss of ground and loss of V_{CC}
- Current sense output proportional to motor current
- Package: ECOPACK®

MultiPowerSO-30™

Description

The VNH2SP30-E is a full bridge motor driver intended for a wide range of automotive applications. The device incorporates a dual monolithic high side driver and two low side switches. The high side driver switch is designed using STMicroelectronic's well known and proven proprietary VIPower™ M0 technology which permits efficient integration on the same die of a true Power MOSFET with an intelligent signal/protection circuitry.

The low side switches are vertical MOSFETs manufactured using STMicroelectronic's proprietary EHD ('STripFET™') process. The three die are assembled in the MultiPowerSO-30 package on electrically isolated leadframes. This package, specifically designed for the harsh automotive environment offers improved thermal performance thanks to exposed die pads. Moreover, its fully symmetrical mechanical design allows superior manufacturability at board level. The input signals IN_A and IN_B can directly interface to the microcontroller to select the motor direction and the brake condition. The $DIAG_A/EN_A$ or $DIAG_B/EN_B$, when connected to an external pull-up resistor, enable one leg of the bridge. They also provide a feedback digital diagnostic signal. The normal condition operation is explained in *Table 12: Truth table in normal operating conditions on page 14*. The motor current can be monitored with the CS pin by delivering a current proportional to its value. The speed of the motor can be controlled in all possible conditions by the PWM up to 20 kHz. In all cases, a low level state on the PWM pin will turn off both the LS_A and LS_B switches. When PWM rises to a high level, LS_A or LS_B turn on again depending on the input pin state.

Table 1. Device summary

Package	Order codes	
	Tube	Tape and Reel
MultiPowerSO-30	VNH2SP30-E	VNH2SP30TR-E

Contents

List of tables

List of figures

1 Block diagram and pin description

Figure 1. Block diagram

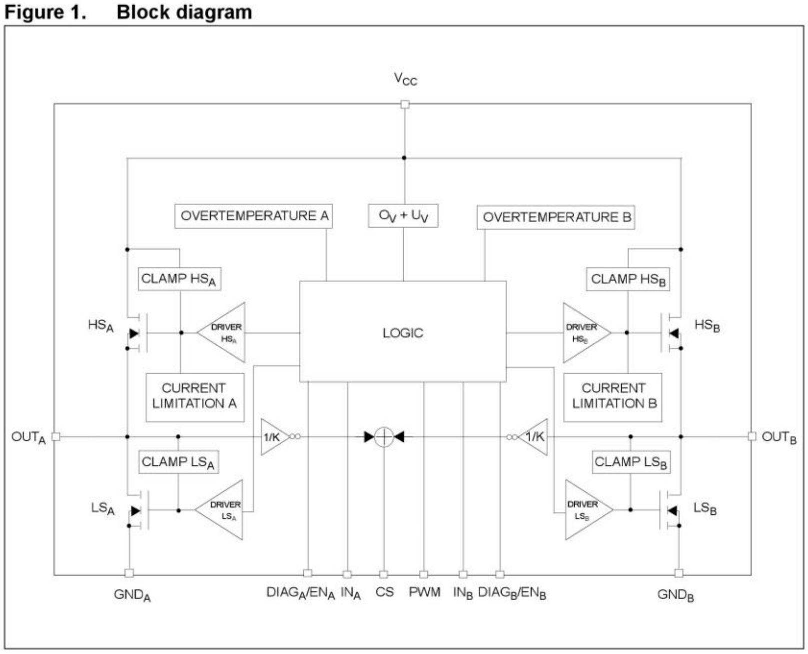


Table 2. Block description

Name	Description
Logic control	Allows the turn-on and the turn-off of the high side and the low side switches according to the truth table
Overvoltage + undervoltage	Shuts down the device outside the range [5.5V..16V] for the battery voltage
High side and low side clamp voltage	Protects the high side and the low side switches from the high voltage on the battery line in all configurations for the motor
High side and low side driver	Drives the gate of the concerned switch to allow a proper $R_{DS(on)}$ for the leg of the bridge
Linear current limiter	Limits the motor current by reducing the high side switch gate-source voltage when short-circuit to ground occurs
Overtemperature protection	In case of short-circuit with the increase of the junction's temperature, shuts down the concerned high side to prevent its degradation and to protect the die
Fault detection	Signals an abnormal behavior of the switches in the half-bridge A or B by pulling low the concerned $EN_x/DIAG_x$ pin

Figure 2. Configuration diagram (top view)

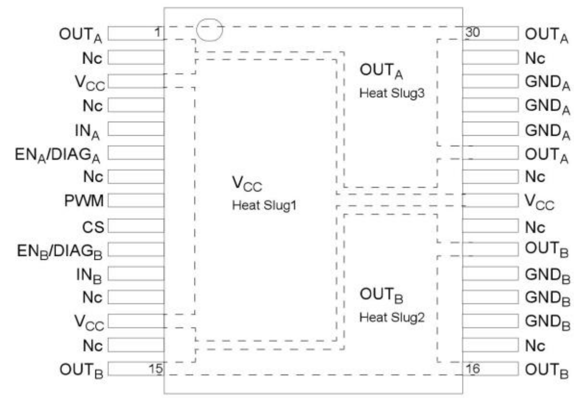


Table 3. Pin definitions and functions

Pin No	Symbol	Function
1, 25, 30	OUT_A, Heat Slug3	Source of high side switch A / Drain of low side switch A
2, 4, 7, 12, 14, 17, 22, 24, 29	NC	Not connected
3, 13, 23	V_{CC}, Heat Slug1	Drain of high side switches and power supply voltage
6	$EN_A/DIAG_A$	Status of high side and low side switches A; open drain output
5	IN_A	Clockwise input
8	PWM	PWM input
9	CS	Output of current sense
11	IN_B	Counter clockwise input
10	$EN_B/DIAG_B$	Status of high side and low side switches B; open drain output
15, 16, 21	OUT_B, Heat Slug2	Source of high side switch B / Drain of low side switch B
26, 27, 28	GND_A	Source of low side switch A(1)
18, 19, 20	GND_B	Source of low side switch B(1)

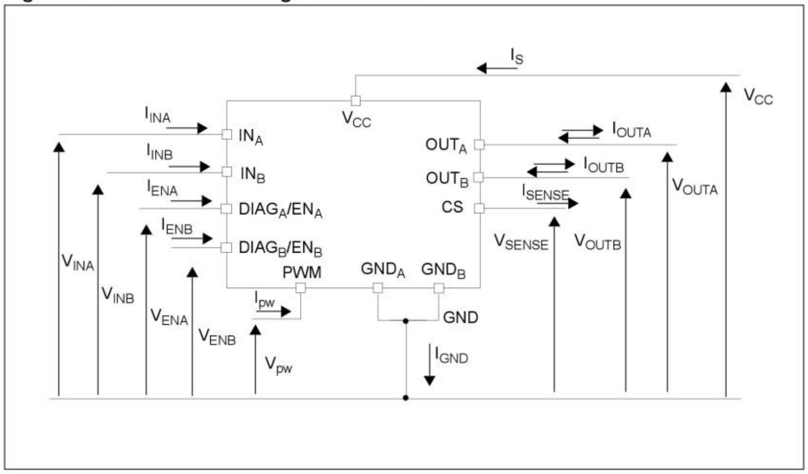
1. GND_A and GND_B must be externally connected together.

Table 4. Pin functions description

Name	Description
V_{CC}	Battery connection
GND_A, GND_B	Power grounds; must always be externally connected together
OUT_A, OUT_B	Power connections to the motor
IN_A, IN_B	Voltage controlled input pins with hysteresis, CMOS compatible. These two pins control the state of the bridge in normal operation according to the truth table (brake to V_{CC}, brake to GND, clockwise and counterclockwise).
PWM	Voltage controlled input pin with hysteresis, CMOS compatible. Gates of low side FETs are modulated by the PWM signal during their ON phase allowing speed control of the motor.
$EN_A/DIAG_A$, $EN_B/DIAG_B$	Open drain bidirectional logic pins. These pins must be connected to an external pull up resistor. When externally pulled low, they disable half-bridge A or B. In case of fault detection (thermal shutdown of a high side FET or excessive ON state voltage drop across a low side FET), these pins are pulled low by the device (see truth table in fault condition).
CS	Analog current sense output. This output sources a current proportional to the motor current. The information can be read back as an analog voltage across an external resistor.

2 Electrical specifications

Figure 3. Current and voltage conventions



2.1 Absolute maximum ratings

Table 5. Absolute maximum ratings

Symbol	Parameter	Value	Unit
V_{CC}	Supply voltage	+41	V
I_{max}	Maximum output current (continuous)	30	A
I_R	Reverse output current (continuous)	-30	
I_{IN}	Input current (IN_A and IN_B pins)	±10	mA
I_{EN}	Enable input current ($DIAG_A/EN_A$ and $DIAG_B/EN_B$ pins)	±10	
I_{pw}	PWM input current	±10	
V_{CS}	Current sense maximum voltage	-3/+15	V
V_{ESD}	Electrostatic discharge (R = 1.5kΩ, C = 100pF) – CS pin – logic pins – output pins: OUT_A, OUT_B, V_{CC}	 2 4 5	 kV kV kV
T_j	Junction operating temperature	Internally limited	°C
T_c	Case operating temperature	-40 to 150	
T_{STG}	Storage temperature	-55 to 150	

2.2 Electrical characteristics

V_{CC} = 9V up to 16 V; -40°C < T_J < 150°C, unless otherwise specified.

Table 6. Power section

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
V_{CC}	Operating supply voltage		5.5		16	V
I_S	Supply current	Off state with all Fault Cleared & ENx=0 IN_A = IN_B = PWM = 0; T_j = 25°C; V_{CC} = 13V IN_A = IN_B = PWM = 0 Off state: IN_A = IN_B = PWM = 0		12 2	30 60	µA µA mA
		On state: IN_A or IN_B = 5V, no PWM			10	mA
R_{ONHS}	Static high side resistance	I_{OUT} = 15A; T_j = 25°C			14	mΩ
		I_{OUT} = 15A; T_j = -40 to 150°C			28	
R_{ONLS}	Static low side resistance	I_{OUT} = 15A; T_j = 25°C			5	
		I_{OUT} = 15A; T_j = -40 to 150°C			10	
V_f	High side free-wheeling diode forward voltage	I_f = 15A		0.8	1.1	V
$I_{L(off)}$	High side off state output current (per channel)	T_j = 25°C; V_{OUTX} = EN_X = 0V; V_{CC} = 13V			3	µA
		T_j = 125°C; V_{OUTX} = EN_X = 0V; V_{CC} = 13V			5	
I_{RM}	Dynamic cross-conduction current	I_{OUT} = 15A (see *Figure 7*)		0.7		A

Table 7. Logic inputs (IN_A, IN_B, EN_A, EN_B)

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
V_{IL}	Input low level voltage	Normal operation ($DIAG_X/EN_X$ pin acts as an input pin)			1.25	V
V_{IH}	Input high level voltage		3.25			
V_{IHYST}	Input hysteresis voltage		0.5			
V_{ICL}	Input clamp voltage	I_{IN} = 1mA	5.5	6.3	7.5	
		I_{IN} = -1mA	-1.0	-0.7	-0.3	
I_{INL}	Input low current	V_{IN} = 1.25V	1			µA
I_{INH}	Input high current	V_{IN} = 3.25V			10	
V_{DIAG}	Enable output low level voltage	Fault operation ($DIAG_X/EN_X$ pin acts as an output pin); I_{EN} = 1mA			0.4	V

Table 8. PWM

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
V_{pwl}	PWM low level voltage				1.25	V
I_{pwl}	PWM pin current	V_{pw} = 1.25V	1			µA
V_{pwh}	PWM high level voltage		3.25			V
I_{pwh}	PWM pin current	V_{pw} = 3.25V			10	µA
$V_{pwhhyst}$	PWM hysteresis voltage		0.5			V
V_{pwcl}	PWM clamp voltage	I_{pw} = 1mA	V_{CC} + 0.3	V_{CC} + 0.7	V_{CC} + 1.0	V
		I_{pw} = -1mA	-6.0	-4.5	-3.0	
C_{INPWM}	PWM pin input capacitance	V_{IN} = 2.5V			25	pF

Table 9. Switching (V_{CC} = 13V, R_{LOAD} = 0.87Ω , unless otherwise specified)

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
f	PWM frequency		0		20	kHz
$t_{d(on)}$	Turn-on delay time	Input rise time < 1µs (see *Figure 6*)			250	µs
$t_{d(off)}$	Turn-off delay time	Input rise time < 1µs (see *Figure 6*)			250	
t_r	Rise time	(see *Figure 5*)		1	1.6	
t_f	Fall time	(see *Figure 5*)		1.2	2.4	
t_{DEL}	Delay time during change of operating mode	(see *Figure 4*)	300	600	1800	
t_{rr}	High side free wheeling diode reverse recovery time	(see *Figure 7*)		110		ns
$t_{off(min)}$ (1)	PWM minimum off time	9V < V_{CC} < 16V; T_j = 25°C; L = 250µH; I_{OUT} = 15A			6	µs

1. To avoid false Short to Battery detection during PWM operation, the PWM signal must be low for a time longer than 6µs.

Table 10. Protection and diagnostic

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
V_{USD}	Undervoltage shut-down				5.5	V
	Undervoltage reset			4.7		
V_{OV}	Overvoltage shut-down		16	19	22	
I_{LIM}	High side current limitation		30	50	70	A
V_{CLP}	Total clamp voltage (V_{CC} to GND)	I_{OUT} = 15A	43	48	54	V
T_{TSD}	Thermal shut-down temperature	V_{IN} = 3.25V	150	175	200	°C
T_{TR}	Thermal reset temperature		135			
T_{HYST}	Thermal hysteresis		7	15		

Table 11. Current sense ($9V < V_{CC} < 16V$)

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
K_1	I_{OUT}/I_{SENSE}	I_{OUT} = 30A; R_{SENSE} = 1.5kΩ; T_j = -40 to 150°C	9665	11370	13075	
K_2	I_{OUT}/I_{SENSE}	I_{OUT} = 8A; R_{SENSE} = 1.5kΩ; T_j = -40 to 150°C	9096	11370	13644	
dK_1 / K_1 (1)	Analog sense current drift	I_{OUT} = 30A; R_{SENSE} = 1.5kΩ; T_j = -40 to 150°C	-8		+8	%
dK_2 / K_2 (1)	Analog sense current drift	I_{OUT} > 8A; R_{SENSE} = 1.5kΩ; T_j = -40 to 150°C	-10		+10	%
I_{SENSEO}	Analog sense leakage current	I_{OUT} = 0A; V_{SENSE} = 0V; T_j = -40 to 150°C	0		65	μA

1. Analog sense current drift is deviation of factor K for a given device over (-40°C to 150°C and $9V < V_{CC} < 16V$) with respect to its value measured at T_j = 25°C, V_{CC} = 13V.

Figure 4. Definition of the delay times measurement

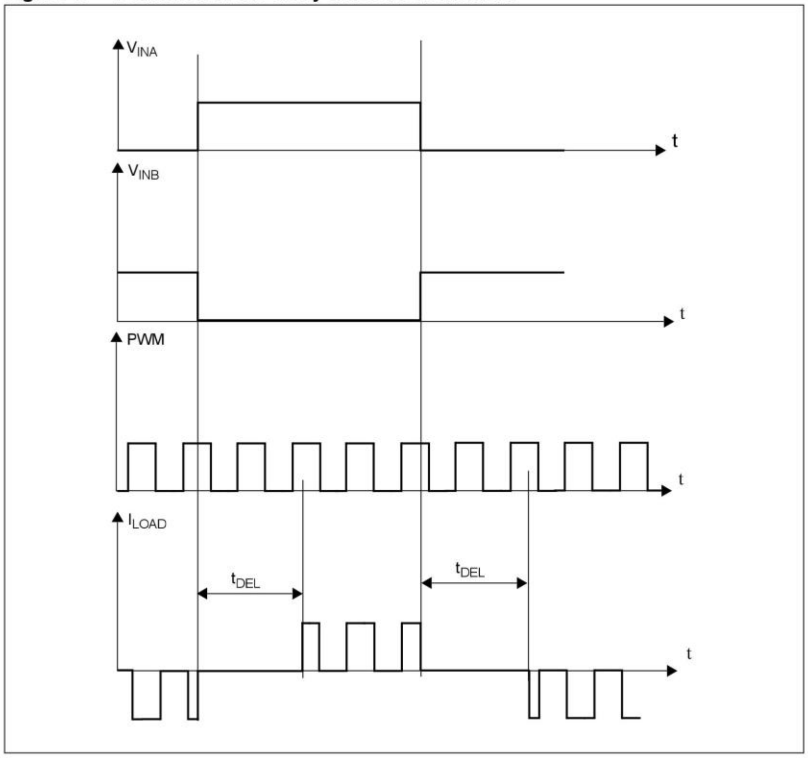


Figure 5. Definition of the low side switching times

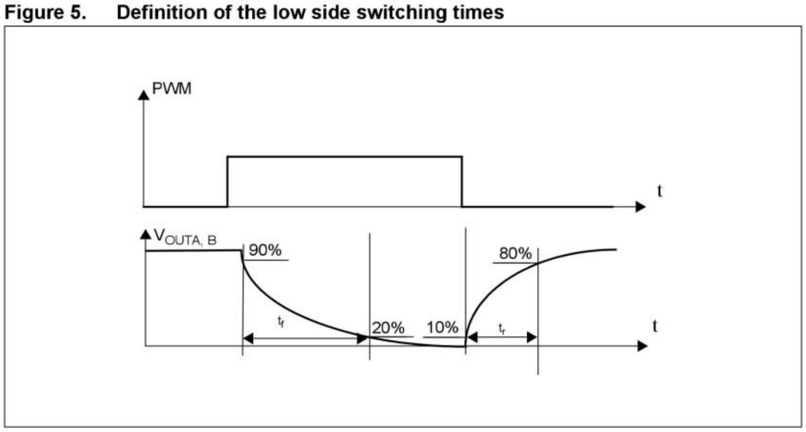


Figure 6. Definition of the high side switching times

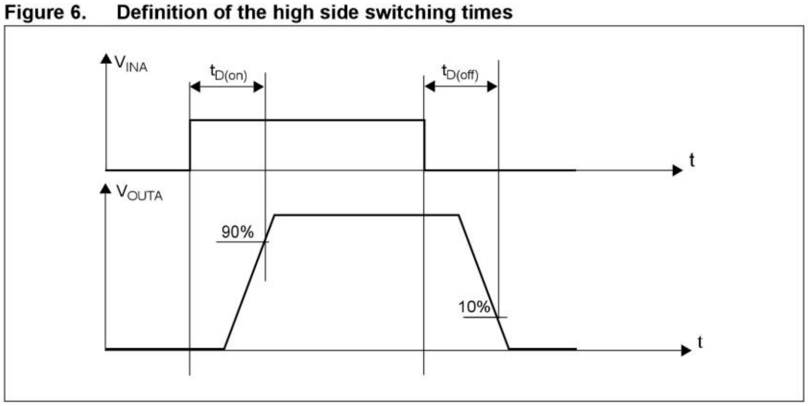


Figure 7. Definition of dynamic cross conduction current during a PWM operation

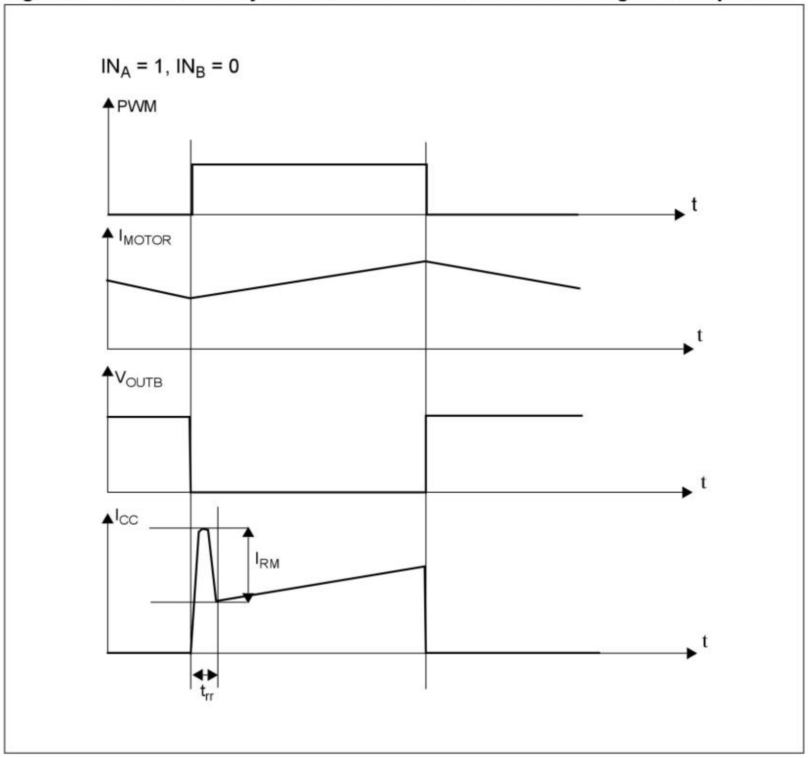


Table 12. Truth table in normal operating conditions

IN_A	IN_B	$DIAG_A/EN_A$	$DIAG_B/EN_B$	OUT_A	OUT_B	CS	Operating mode
1	1	1	1	H	H	High Imp.	Brake to V_{CC}
	0				L	$I_{SENSE} = I_{OUT}/K$	Clockwise (CW)
0	1			L	H		Counterclockwise (CCW)
	0				L	High Imp.	Brake to GND

Table 13. Truth table in fault conditions (detected on OUT_A)

IN_A	IN_B	$DIAG_A/EN_A$	$DIAG_B/EN_B$	OUT_A	OUT_B	CS
1	1	0	1	OPEN	H	High Imp.
	0				L	
0	1				H	I_{OUTB}/K
	0				L	High Imp.
X	X		0		OPEN	
	1		1		H	I_{OUTB}/K
	0				L	High Imp.
		Fault Information	Fault Information	Protection Action	Protection Action	

Note: *Notice that saturation detection on the low side power MOSFET is possible only if the impedance of the short-circuit from the output to the battery is less than 100mΩ when the device is supplied with a battery voltage of 13.5V.*

Table 14. Electrical transient requirements

ISO T/R - 7637/1 Test pulse	Test Level I	Test Level II	Test Level III	Test Level IV	Test levels delays and impedance
1	-25V	-50V	-75V	-100V	2ms, 10Ω
2	+25V	+50V	+75V	+100V	0.2ms, 10Ω
3a	-25V	-50V	-100V	-150V	0.1µs, 50Ω
3b	+25V	+50V	+75V	+100V	
4	-4V	-5V	-6V	-7V	100ms, 0.01Ω
5	+26.5V	+46.5V	+66.5V	+86.5V	400ms, 2Ω

ISO T/R - 7637/1 test pulse	Test levels result I	Test levels result II	Test levels result III	Test levels result IV
1	C	C	C	C
2				
3a				
3b				
4				
5(1)		E	E	E

1. For load dump exceeding the above value a centralized suppressor must be adopted.

Class	Contents
C	All functions of the device are performed as designed after exposure to disturbance.
E	One or more functions of the device are not performed as designed after exposure to disturbance and cannot be returned to proper operation without replacing the device.

2.3 Electrical characteristics curves

Figure 8. On state supply current

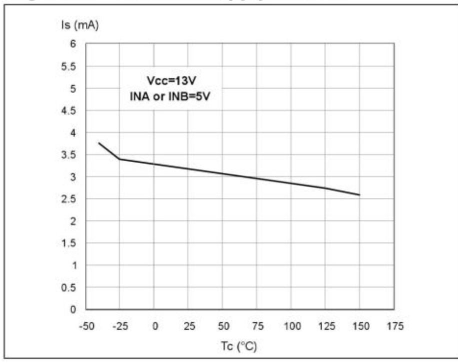


Figure 9. Off state supply current

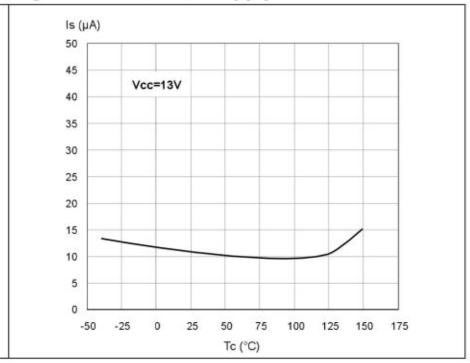


Figure 10. High level input current

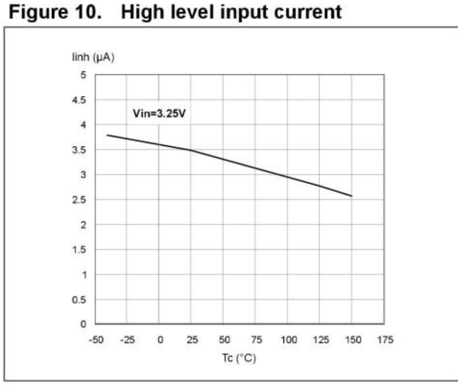


Figure 11. Input clamp voltage

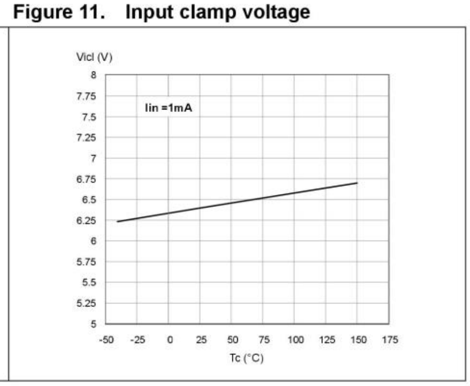


Figure 12. Input high level voltage

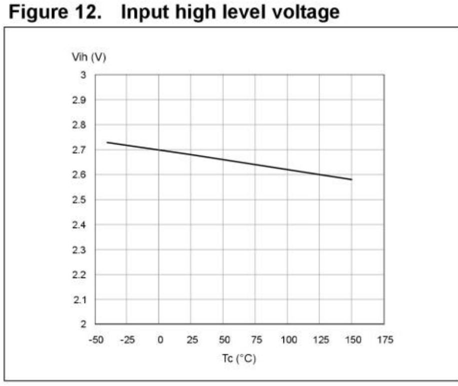


Figure 13. Input low level voltage

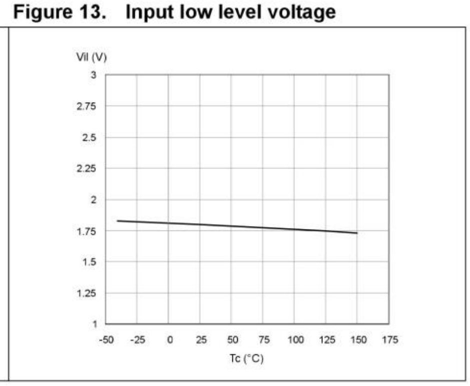


Figure 14. Input hysteresis voltage

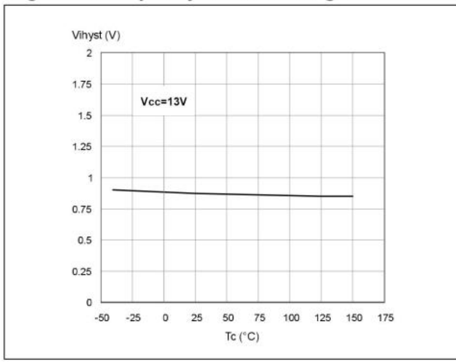


Figure 15. High level enable pin current

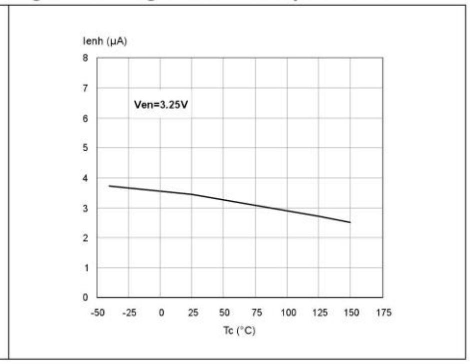


Figure 16. Delay time during change of operation mode

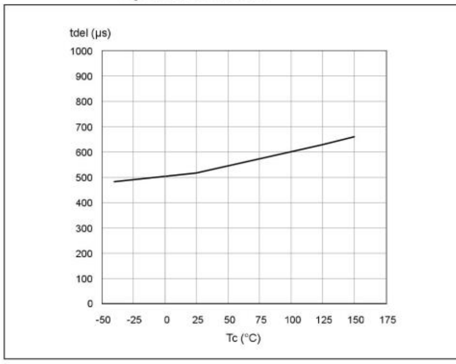


Figure 17. Enable clamp voltage

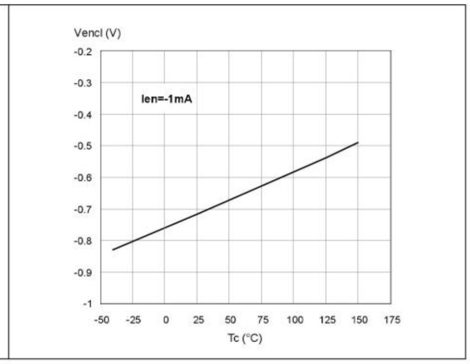


Figure 18. High level enable voltage

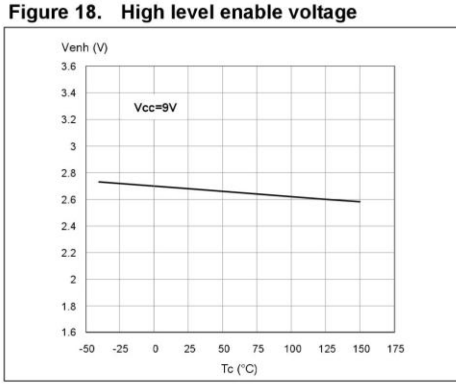


Figure 19. Low level enable voltage

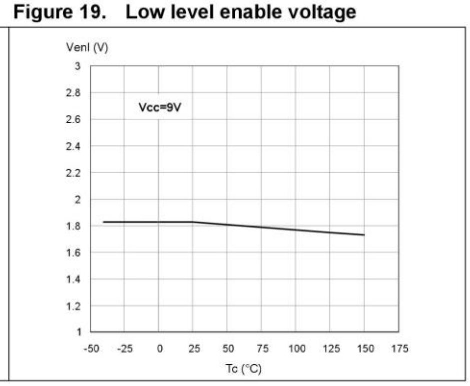


Figure 20. PWM high level voltage

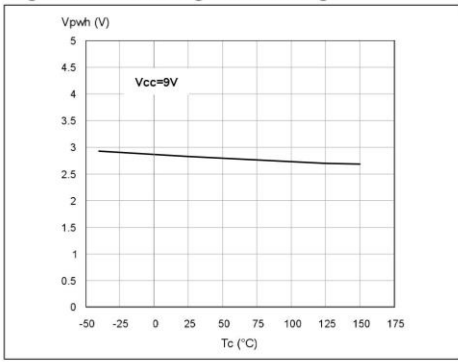


Figure 21. PWM low level voltage

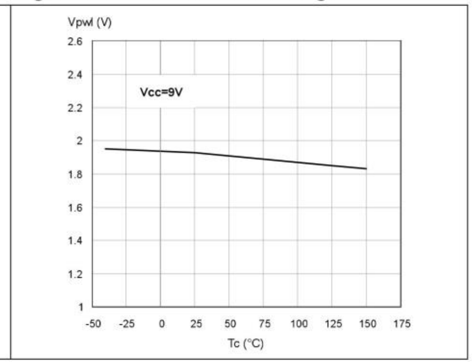


Figure 22. PWM high level current

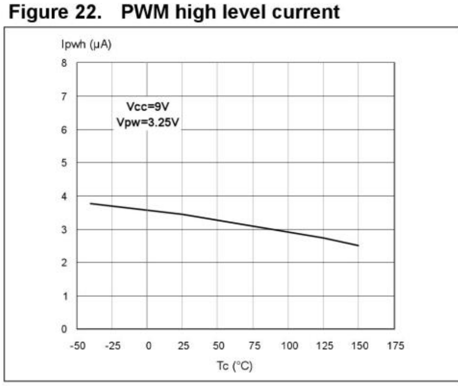


Figure 23. Overvoltage shutdown

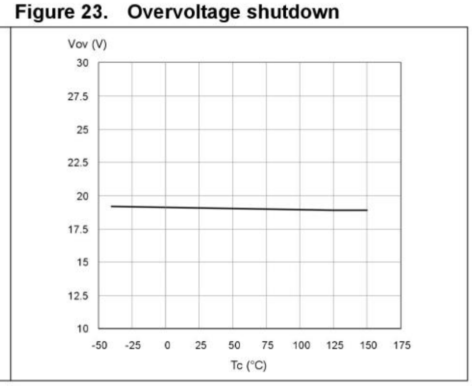


Figure 24. Undervoltage shutdown

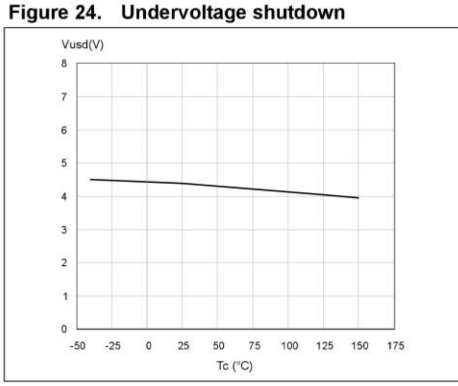


Figure 25. Current limitation

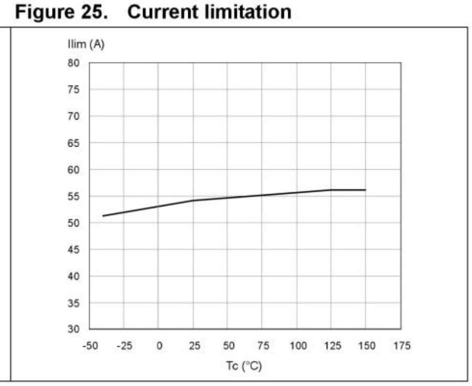


Figure 26. On state high side resistance vs T_{case}

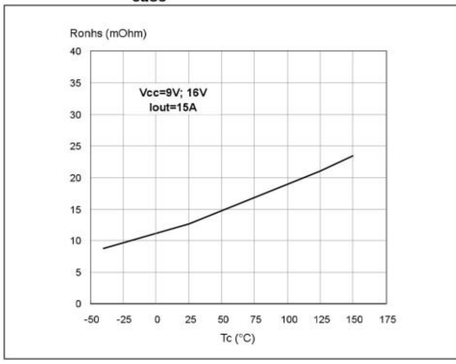


Figure 27. On state low side resistance vs T_{case}

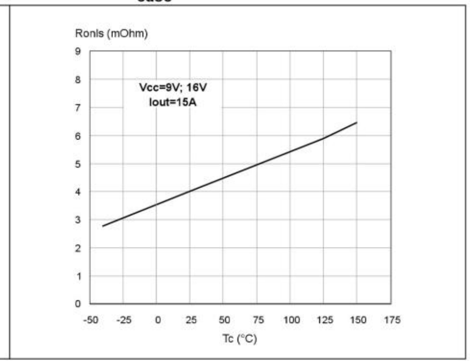


Figure 28. Turn-On delay time

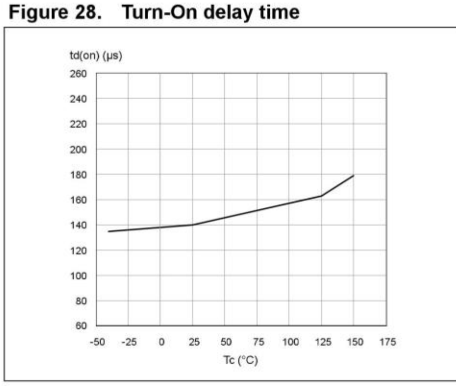


Figure 29. Turn-Off delay time

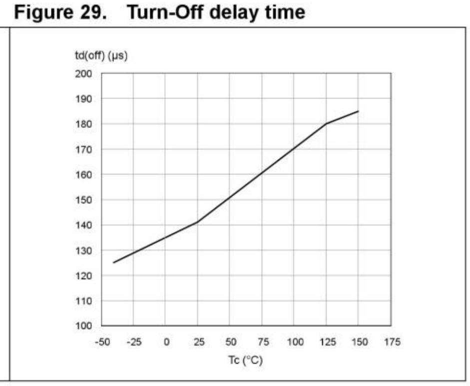


Figure 30. Output voltage rise time

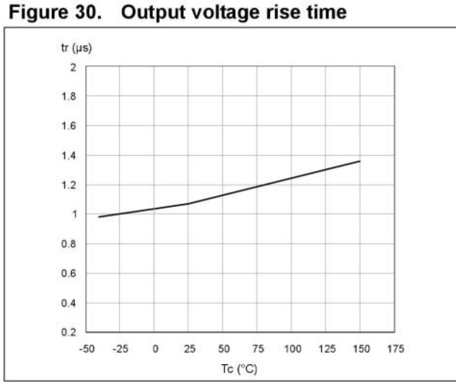
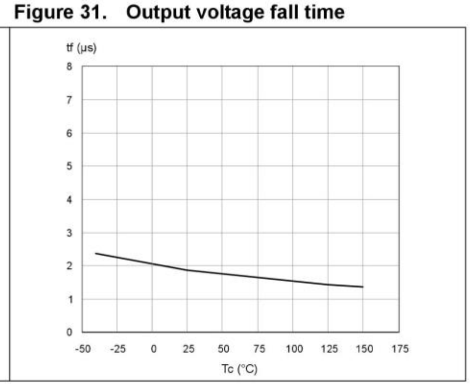


Figure 31. Output voltage fall time

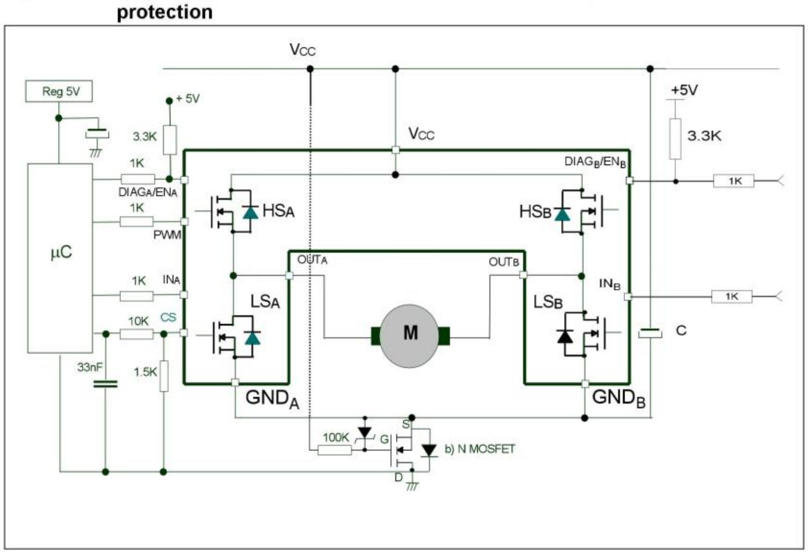


3 Application information

In normal operating conditions the $DIAG_X/EN_X$ pin is considered as an input pin by the device. This pin must be externally pulled high.

PWM pin usage: in all cases, a "0" on the PWM pin will turn off both LS_A and LS_B switches. When PWM rises back to "1", LS_A or LS_B turn on again depending on the input pin state.

Figure 32. Typical application circuit for DC to 20 kHz PWM operation short circuit protection



Note: The value of the blocking capacitor (C) depends on the application conditions and defines voltage and current ripple onto supply line at PWM operation. Stored energy of the motor inductance may fly back into the blocking capacitor, if the bridge driver goes into tri-state. This causes a hazardous overvoltage if the capacitor is not big enough. As basic orientation, 500µF per 10A load current is recommended.

In case of a fault condition the $DIAG_X/EN_X$ pin is considered as an output pin by the device.The fault conditions are:

? overtemperature on one or both high sides

? short to battery condition on the output (saturation detection on the low side power MOSFET)

Possible origins of fault conditions may be:

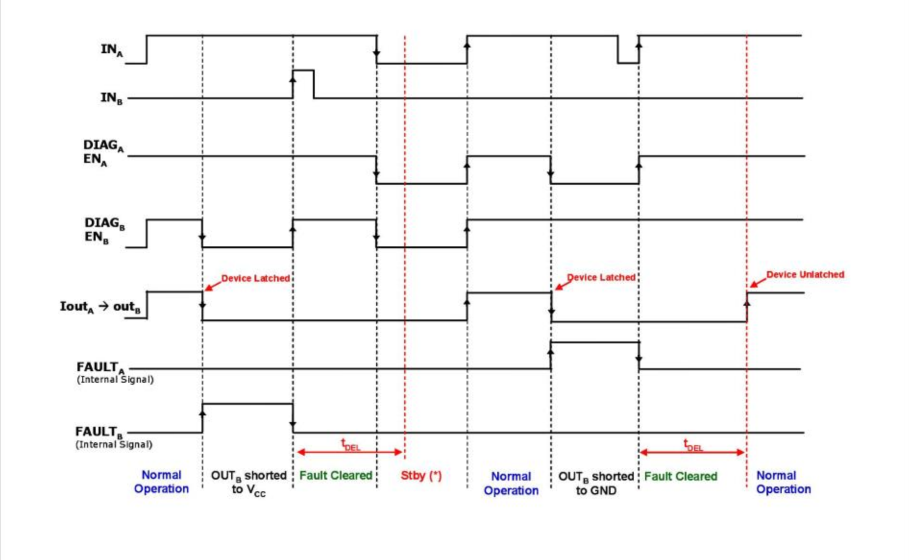
? OUT_A is shorted to ground $\rightarrow$ overtemperature detection on high side A.

? OUT_A is shorted to V_{CC} $\rightarrow$ low side power MOSFET saturation detection.

When a fault condition is detected, the user can know which power element is in fault by monitoring the IN_A, IN_B, $DIAG_A/EN_A$ and $DIAG_B/EN_B$ pins.

In any case, when a fault is detected, the faulty leg of the bridge is latched off. To turn on the respective output (OUT_X) again, the input signal must rise from low to high level.

Figure 33. Behavior in fault condition (How a fault can be cleared)



Note: *In case of the fault condition is not removed, the procedure for unlatching and sending the device in Stby mode is:*

- Clear the fault in the device (toggle : INA if ENA=0 or INB if ENB=0)

- Pull low all inputs, PWM and Diag/EN pins within tDEL.

If the Diag/En pins are already low, PWM=0, the fault can be cleared simply toggling the input. The device will enter in stby mode as soon as the fault is cleared.

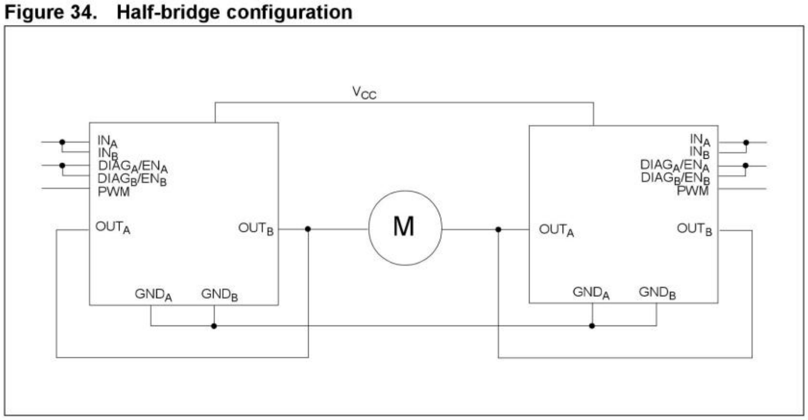
3.1 Reverse battery protection

Three possible solutions can be considered:

1. a Schottky diode D connected to V_{CC} pin
2. an N-channel MOSFET connected to the GND pin (see *Figure 32: Typical application circuit for DC to 20 kHz PWM operation short circuit protection on page 20*)
3. a P-channel MOSFET connected to the V_{CC} pin

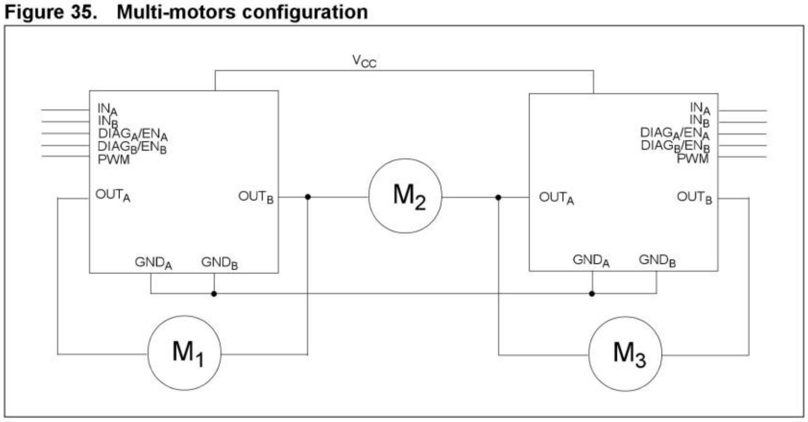
The device sustains no more than -30A in reverse battery conditions because of the two body diodes of the power MOSFETs. Additionally, in reverse battery condition the I/Os of VNH2SP30-E are pulled down to the V_{CC} line (approximately -1.5V). A series resistor must be inserted to limit the current sunk from the microcontroller I/Os. If I_{Rmax} is the maximum target reverse current through µC I/Os, the series resistor is:

Figure 34. Half-bridge configuration



Note: The VNH2SP30-E can be used as a high power half-bridge driver achieving an On resistance per leg of 9.5mΩ.

Figure 35. Multi-motors configuration



Note: The VNH2SP30-E can easily be designed in multi-motors driving applications such as seat positioning systems where only one motor must be driven at a time. $DIAG_X/EN_X$ pins allow to put unused half-bridges in high impedance.

Figure 36. Waveforms in full bridge operation

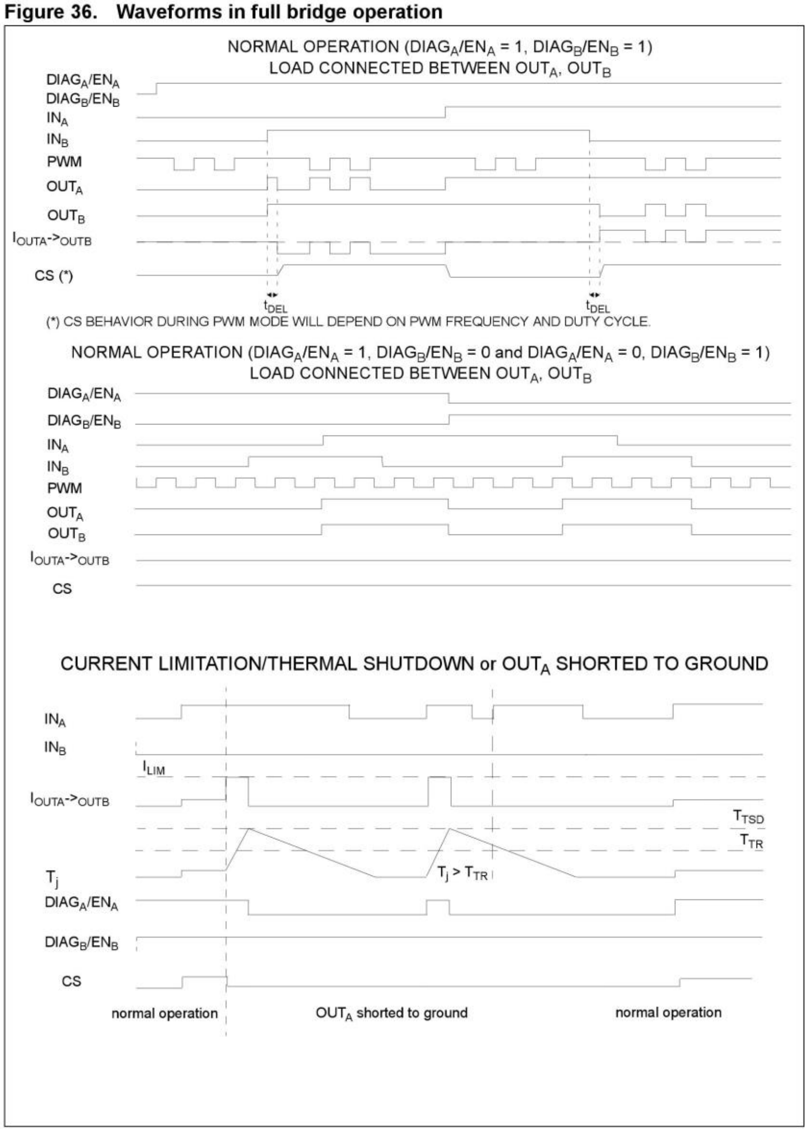
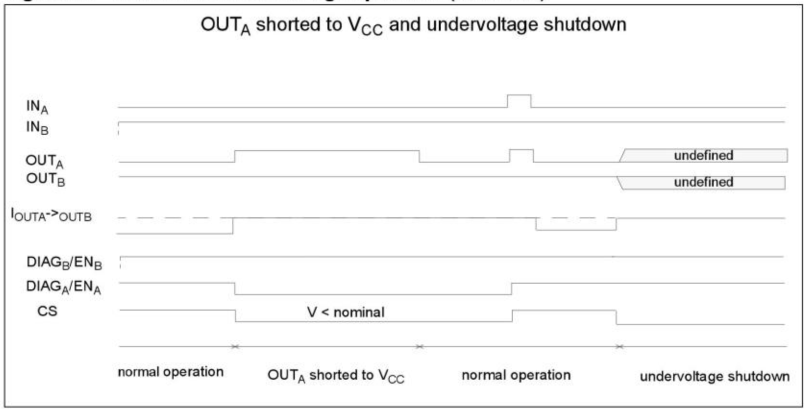


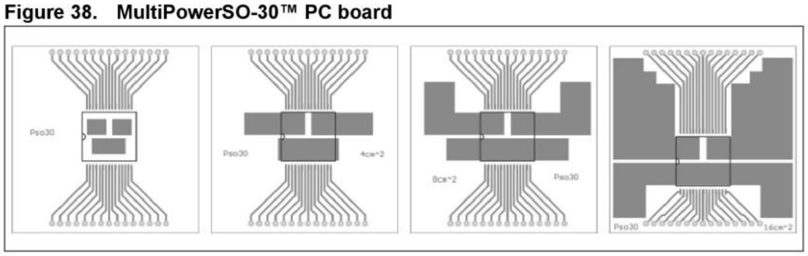
Figure 37. Waveforms in full bridge operation (continued)



4 Package and PCB thermal data

4.1 PowerSSO-30 thermal data

Figure 38. MultiPowerSO-30™ PC board



Note: *Layout condition of R_{th} and Z_{th} measurements (PCB FR4 area = 58mm x 58mm, PCB thickness = 2mm. Cu thickness = 35μm, Copper areas: from minimum pad layout to 16cm^2).*

Figure 39. Chipset configuration

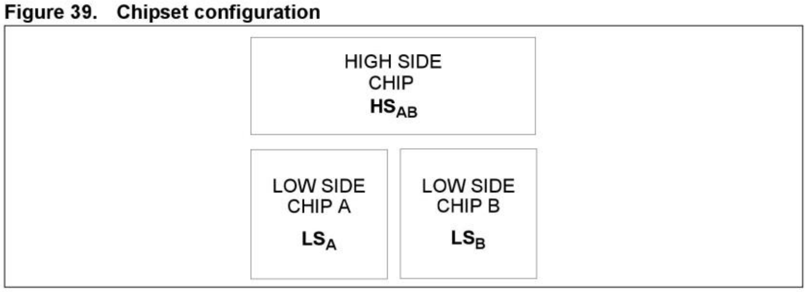
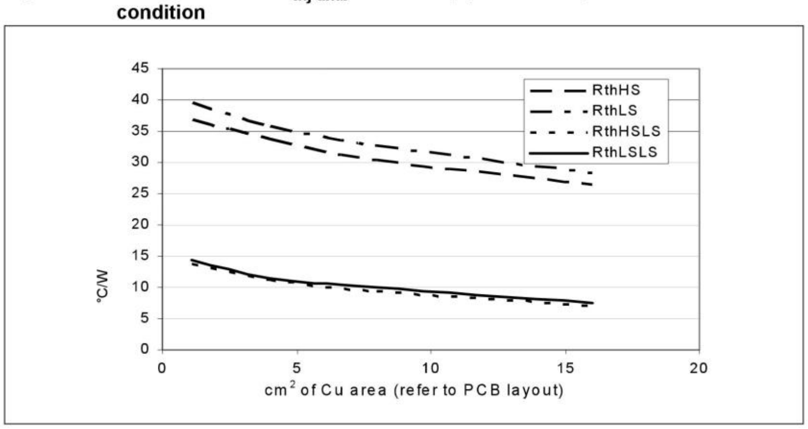


Figure 40. Auto and mutual $R_{thj\text{-}amb}$ vs PCB copper area in open box free air condition



4.1.1 Thermal calculation in clockwise and anti-clockwise operation in steady-state mode

Table 15. Thermal calculation in clockwise and anti-clockwise operation in steady-state mode

HS_A	HS_B	LS_A	LS_B	T_{jHSAB}	T_{jLSA}	T_{jLSB}
ON	OFF	OFF	ON	$P_{dHSA} \times R_{thHS} + P_{dLSB} \times R_{thHSLS} + T_{amb}$	$P_{dHSA} \times R_{thHSLS} + P_{dLSB} \times R_{thLSLS} + T_{amb}$	$P_{dHSA} \times R_{thHSLS} + P_{dLSB} \times R_{thLS} + T_{amb}$
OFF	ON	ON	OFF	$P_{dHSB} \times R_{thHS} + P_{dLSA} \times R_{thHSLS} + T_{amb}$	$P_{dHSB} \times R_{thHSLS} + P_{dLSA} \times R_{thLS} + T_{amb}$	$P_{dHSB} \times R_{thHSLS} + P_{dLSA} \times R_{thLSLS} + T_{amb}$

4.1.2 Thermal resistances definition (values according to the PCB heatsink area)

$\mathbf{R_{thHS}} = R_{thHSA} = R_{thHSB}$ = High Side Chip Thermal Resistance Junction to Ambient (HS_A or HS_B in ON state)

$\mathbf{R_{thLS}} = R_{thLSA} = R_{thLSB}$ = Low Side Chip Thermal Resistance Junction to Ambient

$\mathbf{R_{thHSLS}} = R_{thHSALSB} = R_{thHSBLSA}$ = Mutual Thermal Resistance Junction to Ambient between High Side and Low Side Chips

$\mathbf{R_{thLSLS}} = R_{thLSALSB}$ = Mutual Thermal Resistance Junction to Ambient between Low Side Chips

4.1.3 Thermal calculation in transient mode[a]

$\mathbf{T_{jHSAB}} = Z_{thHS} \times P_{dHSAB} + Z_{thHSLS} \times (P_{dLSA} + P_{dLSB}) + T_{amb}$

$\mathbf{T_{jLSA}} = Z_{thHSLS} \times P_{dHSAB} + Z_{thLS} \times P_{dLSA} + Z_{thLSLS} \times P_{dLSB} + T_{amb}$

$\mathbf{T_{jLSB}} = Z_{thHSLS} \times P_{dHSAB} + Z_{thLSLS} \times P_{dLSA} + Z_{thLS} \times P_{dLSB} + T_{amb}$

4.1.4 Single pulse thermal impedance definition (values according to the PCB heatsink area)

$\mathbf{Z_{thHS}}$ = High Side Chip Thermal Impedance Junction to Ambient

$\mathbf{Z_{thLS}} = Z_{thLSA} = Z_{thLSB}$ = Low Side Chip Thermal Impedance Junction to Ambient

$\mathbf{Z_{thHSLS}} = Z_{thHSABLSA} = Z_{thHSABLSB}$ = Mutual Thermal Impedance Junction to Ambient between High Side and Low Side Chips

$\mathbf{Z_{thLSLS}} = Z_{thLSALSB}$ = Mutual Thermal Impedance Junction to Ambient between Low Side Chips

a. Calculation is valid in any dynamic operating condition. P_d values set by user.

Equation 1: pulse calculation formula

$$Z_{TH\delta} = R_{TH} \cdot \delta + Z_{THtp}(1-\delta)$$

where $\delta = t_p / T$

Figure 41. MultiPowerSO-30 HSD thermal impedance junction ambient single pulse

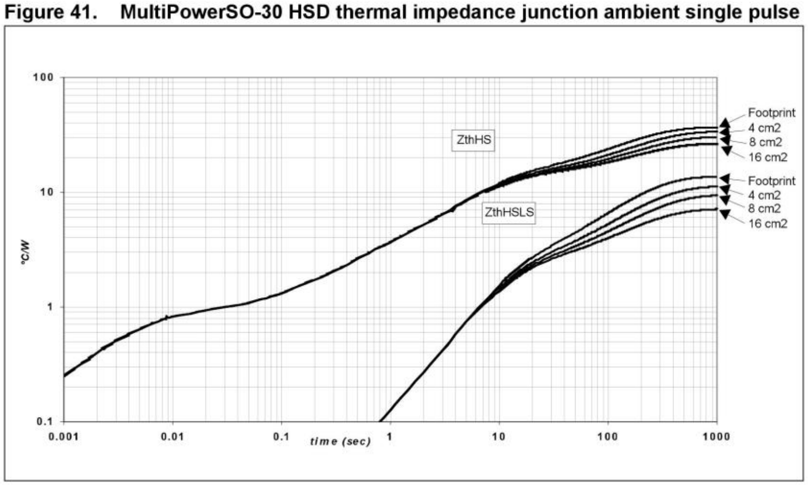


Figure 42. MultiPowerSO-30 LSD thermal impedance junction ambient single pulse

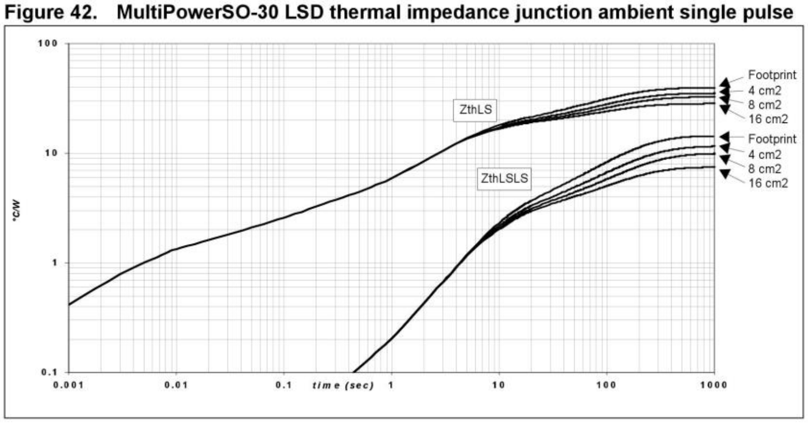


Figure 43. Thermal fitting model of an H-bridge in MultiPowerSO-30

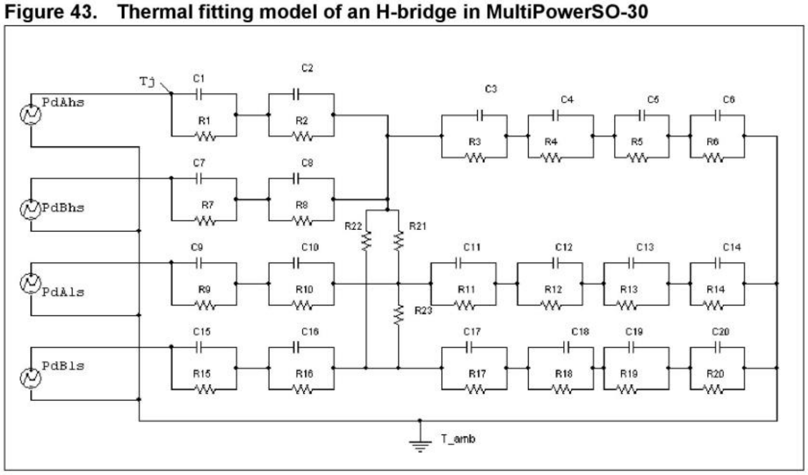


Table 16. Thermal parameters[(1)]

Area/island (cm^2)	Footprint	4	8	16
R1 = R7 (°C/W)	0.05			
R2 = R8 (°C/W)	0.3			
R3 (°C/W)	0.5			
R4 (°C/W)	1.3			
R5 (°C/W)	14			
R6 (°C/W)	44.7	39.1	31.6	23.7
R9 = R15 (°C/W)	0.2			
R10 = R16 (°C/W)	0.4			
R11 = R17 (°C/W)	0.8			
R12 = R18 (°C/W)	1.5			
R13 = R19 (°C/W)	20			
R14 = R20 (°C/W)	46.9	36.1	30.4	20.8
R21 = R22 = R23 (°C/W)	115			
C1 = C7 (W.s/°C)	0.005			
C2 = C8 (W.s/°C)	0.008			
C3 = C11 = C17 (W.s/°C)	0.01			
C4 = C13 = C19 (W.s/°C)	0.3			
C5 (W.s/°C)	0.6			
C6 (W.s/°C)	5	7	9	11
C9 = C15 (W.s/°C)	0.003			
C10 = C16 (W.s/°C)	0.006			
C12 = C18 (W.s/°C)	0.075			
C14 = C20 (W.s/°C)	2.5	3.5	4.5	5.5

1. The blank space means that the value is the same as the previous one.

5 Package and packing information

5.1 ECOPACK® packages

In order to meet environmental requirements, ST offers these devices in ECOPACK® packages. ECOPACK® packages are lead-free. The category of Second Level Interconnect is marked on the package and on the inner box label, in compliance with JEDEC Standard JESD97. The maximum ratings related to soldering conditions are also marked on the inner box label.

ECOPACK is an ST trademark. ECOPACK specifications are available at www.st.com.

5.2 MultiPowerSO-30 package mechanical data

Figure 44. MultiPowerSO-30 package outline

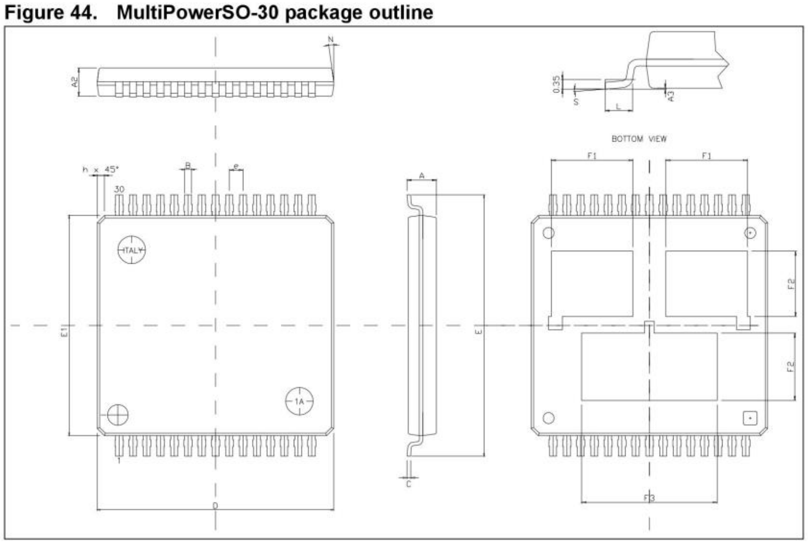
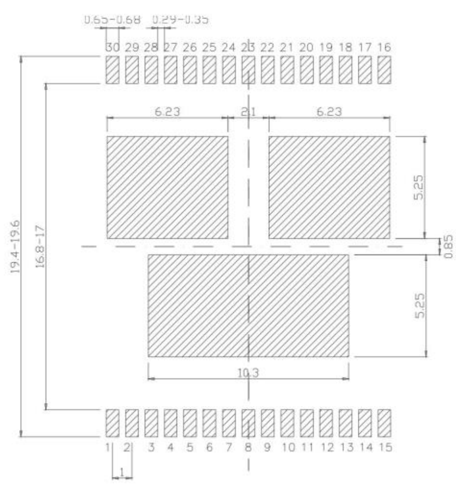


Table 17. MultiPowerSO-30 mechanical data

Symbol	Millimeters		
	Min	Typ	Max
A			2.35
A2	1.85		2.25
A3	0		0.1
B	0.42		0.58
C	0.23		0.32
D	17.1	17.2	17.3
E	18.85		19.15
E1	15.9	16	16.1
e		1	
F1	5.55		6.05
F2	4.6		5.1
F3	9.6		10.1
L	0.8		1.15
N			10deg
S	0deg		7deg

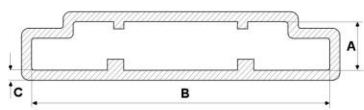
Figure 45. MultiPowerSO-30 suggested pad layout



5.3 Packing information

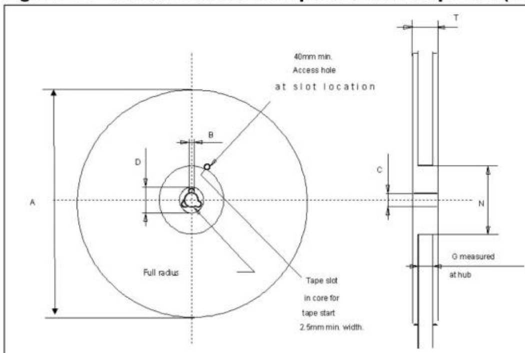
Note: The devices can be packed in tube or tape and reel shipments (see the Device summary on page 1 for packaging quantities).

Figure 46. MultiPowerSO-30 tube shipment (no suffix)



Dimension	mm
Base Q.ty	29
Bulk Q.ty	435
Tube length (± 0.5)	532
A	3.82
B	23.6
C (± 0.13)	0.8

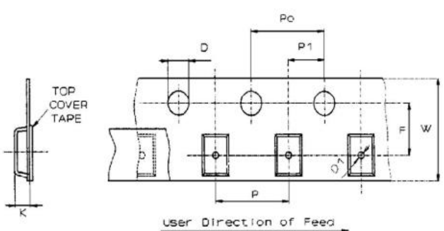
Figure 47. MultiPowerSO-30 tape and reel shipment (suffix "TR")



Reel dimensions

SO-28 tube shipment (no suffix)

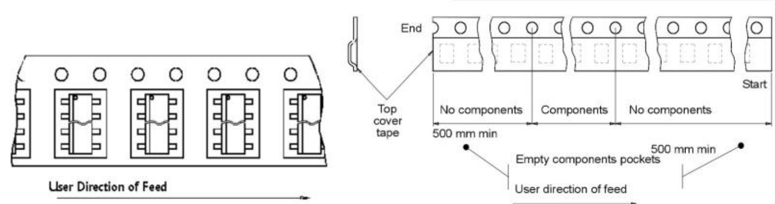
Dimension	mm
Base Q.ty	1000
Bulk Q.ty	1000
A (max)	330
B (min)	1.5
C (± 0.2)	13
D (min)	20.2
G (+ 2 / -0)	32
N (min)	100
T (max)	38.4



Tape dimensions

According to Electronic Industries Association (EIA) Standard 481 rev. A, Feb 1986

Description	Dimension	mm
Tape width	W	32
Tape Hole Spacing	P0 (± 0.1)	4
Component Spacing	P	24
Hole Diameter	D (± 0.1/-0)	1.5
Hole Diameter	D1 (min)	2
Hole Position	F (± 0.1)	14.2
Compartment Depth	K (max)	2.2



6 Revision history

Table 18. Document revision history

Date	Revision	Description of changes
Sep-2004	1	First issue
Dec- 2004	2	Inserted $t_{off(min)}$ test condition modification and note Modified I_{RM} figure number
Feb-2005	3	Minor changes
Apr-2005	4	Public release
01-Sep-2006	5	Document converted into new ST corporate template. Added table of contents, list of tables and list of figures Removed figure number from package outline on page 1 Changed Features on page 1 to add ECOPACK® package Added Section 1: Block diagram and pin description on page 5 Added Section 2.2: Electrical characteristics on page 9 Added "low" and "high" to parameters for I_{INL} and I_{INH} in Table 7 on page 9 Inserted note in Figure 32 on page 20 Added vertical limitation line to left side arrow of $t_{D(off)}$ to Figure 7 on page 13 Added Section 4.1: PowerSSO-30 thermal data on page 25 Added Section 5: Package and packing information on page 29 Added Section 5.3: Packing information on page 31 Updated disclaimer (last page) to include a mention about the use of ST products in automotive applications
15-May-2007	6	Document reformatted and converted into new ST template.
06-Feb-2008	7	Corrected Heat Slug numbers in Table 3: Pin definitions and functions.
02-Oct-2008	8	Added new infomation in Table 6: Power section Added Figure 33: Behavior in fault condition (How a fault can be cleared)
20-Sep-2013	9	Updated Disclaimer.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER'S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR "AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL" INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.
Information in this document supersedes and replaces all information previously supplied.
The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com

ANEXO 14: Demostración del uso del controlador PD

La función transferencia de la posición angular vs tensión de armadura de cada motor es:

$JS^2\theta + FS\theta = Kv - d(s)/r$

Reemplazando el voltaje v del controlador PD, tenemos:

$JS^2\theta + FS\theta = K[K_P(\theta_d - \theta) - K_D S\theta] - d(s)/r$

$= KK_P\theta_d - KK_P\theta - KK_D S\theta - d(s)/r$

$\theta\,[JS^2 + (F + KK_D)S + KK_P] = KK_P\theta_d - d(s)/r$

Siendo P el polinomio característico $P(s) = JS^2 + (F + KK_D)S + KK_P$

Entonces:

$\theta = KK_P\theta_d/P(s) - [d(s)/r]/P(s)$

El error $e = \theta_d - \theta$

Reemplazando el valor de la posición angular en esta expresión, nos queda:

$e = \theta_d - KK_P\theta_d/P(s) + [d(s)/r]/P(s)$

$e = [1 - KK_P/P(s)]\,\theta_d + [d(s)/r]/P(s)$

$$e = \frac{P(s) - KK_P}{P(s)}\theta_d + \frac{d(s)/r}{P(s)}$$

$$\boldsymbol{e} = \frac{JS^2 + (F + KK_D)S}{P(s)}\theta_d(s) + \frac{d(s)/r}{P(s)}$$

Aplicando el teorema del valor final $\lim_{s \to o} Se(s)$

$$e_\infty = \frac{d/r}{KK_P}$$

Si no hay torque de perturbación (d = 0), entonces no hay error en estado estacionario para entrada escalón de θ.

Para el comportamiento dinámico del sistema de control PD, se analiza el polinomio característico P(s) como un sistema típico de segundo orden:

Queda la siguiente relación:

$$S^2 + \frac{F + KK_D}{J}S + \frac{KK_P}{J} = S^2 + 2\delta w_n S + w_n^2$$

Entonces K_P y K_D se determinan en función de δ y w_n, donde en robótica se requiere de una respuesta críticamente amortiguada, por lo que el amortiguamiento δ = 1.

El análisis del polinomio característico P(S) por el método de Routh-Hurwitz**,** determina que la condición de estabilidad está dada por:

$$\mathrm{F} + \mathrm{KK_D} > \frac{\mathrm{J}}{\mathrm{KK_P}}$$

Reordenando esta expresión, da:

$$K_P > \frac{J}{K(F + KK_D)}$$

```
/*
* GccApplication1.c
*
* Created: 29/08/2014 05:38:06 a.m.
* Author:
*/




#ifndef F_CPU
#define F_CPU 8000000UL // or whatever may be your frequency
#endif

#include <util/delay.h>
#include <avr/interrupt.h>
#include <util/twi.h>
#include <avr/io.h>

#define USART_BAUDRATE 9600
#define MYUBRR F_CPU/8/USART_BAUDRATE-1
#define BUFFER_SERIAL 32

#define num_tramas 1
#define num_motores 1

//variables declaradas
int n=0;
unsigned char data[32];
unsigned char temp[5];
unsigned char direc_esclavo[5];
unsigned char G[10];
int h=1;
int a=0;
//---------------------------------------------------------------------
inicia programa

unsigned char USART_receive();
void USART_send(unsigned char);
void recoge_signo(int);
void recoge_numeros(int);
void recoge_motor(int);
void recoge_cantidad(int);
void configuracion_usart(unsigned int);
void configuracion_i2c(unsigned long int);
void asigna_direc_esclavo();
void recibe_datos_pc();
void trx(unsigned char,unsigned char);
int main(void)
{  //programa principal
       asigna_direc_esclavo();
       configuracion_i2c(100000);
       configuracion_usart(12);
       DDRC=0xFF;
```

```
        DDRB=0xFF;
        DDRD=0xF0;//definicion de entradas y salidas


        while(1)
        {
                a=0;
                recibe_datos_pc(); //unico programa que se ejecuta
                PORTB=0x01;
        }
}

void configuracion_i2c(unsigned long int frecuencia)
{

        TWBR = 0x18;
        TWSR=(0<<TWPS1);
        TWSR=(0<<TWPS0);

}
void trx(unsigned char direccion,unsigned char dato)
{
        TWCR=0xA4;
        while(!(TWCR&0x80));
        TWDR=direccion;
        TWCR=0x84;
        while(!(TWCR & 0x80));
        TWDR=dato;
        TWCR=0x84;
        while(!(TWCR & 0x80));
        TWCR =0x94;

}
void configuracion_usart(unsigned int UBRR)
{
        //
        UBRRH = 0;
        UBRRL = 51;

        UCSRA= (0<<U2X);//modificado
        UCSRB = (1<<RXEN)|(1<<TXEN);
        UCSRC = (1<<URSEL)|(0<<USBS)|(1<<UCSZ1)|(1<<UCSZ0); //1 bit de
parada|8 bits de datos|Sin paridad
}
unsigned char USART_receive(void)
{
        // Wait until a byte has been received
        while(!(UCSRA & (1<<RXC)));

        return UDR;
}
void USART_send(unsigned char data)
{
        // Wait until a byte has been received
```

```
        while(!(UCSRA & (1<<UDRE)));
        // Return received data
        UDR=data;
}
void asigna_direc_esclavo()
{

        direc_esclavo[0]= 0x04;
        direc_esclavo[1]= 0x06;
        direc_esclavo[2]= 0x08;
        direc_esclavo[3]= 0x0A;
        direc_esclavo[4]= 0x0C;
}

/*
unsigned char rtx(unsigned char direccion)
{
        TWCR=0xA4;
        while(!(TWCR&0x80));
        TWDR=0b10101011;DIRECCION* falta modificar
        TWCR=0x84;
        while(!(TWCR & 0x80));
        dato=TWDR;
        TWCR=0x84;
        while(!(TWCR & 0x80));
        TWCR =0b10010100

}
*/
void recibe_datos_pc()
{




        for(int i=0;i<32;i++)
        {

                data[i]=USART_receive(); //recibe dato
                a=a+1;

                 //devuelve dato
                //ultima modificación

                trx(direc_esclavo[n],data[i]);
                //falta quitarle ASCII

                //---------------------------------------------

                if(data[i]==64) //es arroba?
                {     PORTC=0x02; //prende led
                n++;}
                if(n==4){n=0;}
```

```
        }
USART_send(a);
}
```

```
/*
 * esclavo_09_10.c
 *
 * Created: 09/10/2014 04:23:36 p.m.
 *  Author:
 */



#ifndef F_CPU
#define F_CPU 8000000UL // or whatever may be your frequency
#endif
#include <avr/io.h>
#include <util/delay.h>
#include <util/twi.h>
#include <avr/interrupt.h>
#include <inttypes.h>
#include <stdlib.h>

char recibe_dato();//funcion encargada de recibir los datos de TWI
void configuracion_puertos();//funcion encargada de configurar los
puertos del
void configura_twi();//función que configura la comunicación por TWI
void configurar_interrupcion();//funcion que configura la interrupcion
externa
void configurar_pwm();//funcion que permite configurar el pwm en PB1


char arr[5];//se guardara lo siguiente: dig1,dig2,dig3,signo de giro,@
int i=0;
uint32_t angulo;//angulo que se debe mover el motor
uint32_t angulo_ant=0;//angulo anterior
uint32_t cant_pulsos=0;
char acabo_movimiento=1;//'1'-> el mov ya finalizo
//'0'-> el mov todavía no ha acabado
uint32_t cant_pulsos_cont=0;
double e=0;//error proporcional
double valor_pwm=0;//indicará el ancho de pulso del pwm
int32_t error=0;
int32_t derror=0;
int32_t error_anterior=0;
double derror_double=0;
double ierror_double=0;
uint32_t velocidad;

int main(void)
{
	configuracion_puertos();
	configura_twi();
	configurar_interrupcion();
	configurar_pwm();
	while(1)
	{
```

```
                //---------------------------------------
                //Guardamos la trama recibida en el arreglo "arr"
                while((i+1)<=5)
                {
                        if(i<=2)
                        {
                                arr[i]=recibe_dato()-'0';
                        }
                        else
                        {
                                arr[i]=recibe_dato();
                        }
                        i=i+1;
                }
                //--------------------------------------
                PORTC=0x00;
                angulo=arr[0]*100+arr[1]*10+arr[2];
/*              velocidad =100;
                if(angulo==250)
                {
                        PORTC=0x01;
                }*/
                //velocidad=arr[5]*100+arr[6]*10+arr[7];
                                //cant_pulsos=(angulo-
angulo_ant)*680;//motor 3
                                //cant_pulsos=(angulo-
angulo_ant)*180;//motor 4
                                //cant_pulsos=(angulo-
angulo_ant)*1.2;//motor 2
                                cant_pulsos=(angulo-
angulo_ant)*16;//motor 1
                                acabo_movimiento=0;
                                angulo_ant=0;
                                if(arr[3]=='+')
                                {
                                        PORTB=0b00001011;
                                }
                                else
                                {
                                        PORTB=0b00000111;
                                }
                                cant_pulsos_cont=0;
                                OCR1A=(unsigned int)velocidad;
                                while(acabo_movimiento==0)
                                {
                                        error=cant_pulsos-
cant_pulsos_cont;
                                        derror=error-error_anterior;
                                        e=(double)error;
                                        derror_double=(double)derror;

        ierror_double=(double)error+0.0001*ierror_double;
                                        valor_pwm=23*e +
2.37*derror_double + 0*ierror_double;
```

```
                                                            if (valor_pwm>=255)
                                                            {
                                                                    valor_pwm=254;
                                                            }
                                                            if(valor_pwm<=40)
                                                            {
                                                                    valor_pwm=40;
                                                            }

                                                    //valor_pwm=255;
                                                    OCR1A=(unsigned
int)valor_pwm;//modificamos el ancho de pulso del PWM
                                                    error_anterior=error;


                                            }
                                            PORTC=0x01;
                                            PORTB=0x00;
                                            OCR1A=0;
                                            i=0;
                                    }
}
ISR(INT0_vect)
{
        cant_pulsos_cont=cant_pulsos_cont+1;
        if(cant_pulsos_cont>=cant_pulsos)
        {
                acabo_movimiento=1;
        }
}
ISR(INT1_vect)
{
        PORTB=0x00;
        PORTC=0x01;
        acabo_movimiento=1;
        OCR1A=0x00;
        i=0;
}
void configuracion_puertos()
{
        DDRB=0x0F;
        DDRC=0x01;
}
void configura_twi()
{
        TWCR=0xC4;
        TWAR=0x04;
}
char recibe_dato()
{
        char p2;
        while(!(TWCR & 0x80));
```

```
        TWCR=0x84;
        while(!(TWCR & 0x80));
        p2=TWDR;
        TWCR=0x84;
        TWCR=0xC4;
        return p2;

}

void configurar_interrupcion()
{
        cli();                  //configuracion de interrupcion externa
INT0
        //MCUCR=0x03;
        MCUCR=0x0f;
        //GIFR=0x40;
        //GICR=0x40;
        GICR=0xc0;
        GIFR=0xc0;
        sei();
}
void configurar_pwm()
{
        DDRB=0x02;
        TCCR1A=0x81;
        TCCR1B=0x05;
        TCNT1=0x00;
        OCR1A=0;

}
```

```
/*
 * esclavo2_09_10.c
 *
 * Created: 09/10/2014 04:23:36 p.m.
 *  Author:
 */




#ifndef F_CPU
#define F_CPU 8000000UL // or whatever may be your frequency
#endif
#include <avr/io.h>
#include <util/delay.h>
#include <util/twi.h>
#include <avr/interrupt.h>
#include <inttypes.h>
#include <stdlib.h>

char recibe_dato();//funcion encargada de recibir los datos de TWI
void configuracion_puertos();//funcion encargada de configurar los
puertos del
void configura_twi();//función que configura la comunicación por TWI
void configurar_interrupcion();//funcion que configura la interrupcion
externa
void configurar_pwm();//funcion que permite configurar el pwm en PB1


char arr[5];//se guardara lo siguiente: dig1,dig2,dig3,signo de giro,@
int i=0;
uint32_t angulo;//angulo que se debe mover el motor
uint32_t angulo_ant=0;//angulo anterior
uint32_t cant_pulsos=0;
char acabo_movimiento=1;//'1'-> el mov ya finalizo
//'0'-> el mov todavía no ha acabado
uint32_t cant_pulsos_cont=0;
double e=0;//error proporcional
double valor_pwm=0;//indicará el ancho de pulso del pwm
int32_t error=0;
int32_t derror=0;
int32_t error_anterior=0;
double derror_double=0;
double ierror_double=0;
uint32_t velocidad;

int main(void)
{
        configuracion_puertos();
        configura_twi();
        configurar_interrupcion();
        configurar_pwm();
        while(1)
        {
```

```
                //-----------------------------------------
                //Guardamos la trama recibida en el arreglo "arr"
                while((i+1)<=5)
                {
                        if(i<=2)
                        {
                                arr[i]=recibe_dato()-'0';
                        }
                        else
                        {
                                arr[i]=recibe_dato();
                        }
                        i=i+1;
                }
                //-----------------------------------------
                PORTC=0x00;
                angulo=arr[0]*100+arr[1]*10+arr[2];
/*              velocidad =100;
                if(angulo==250)
                {
                        PORTC=0x01;
                }*/
                //velocidad=arr[5]*100+arr[6]*10+arr[7];
                                //cant_pulsos=(angulo-
angulo_ant)*680;//motor 3
                                //cant_pulsos=(angulo-
angulo_ant)*180;//motor 4
                                //cant_pulsos=(angulo-
angulo_ant)*1.2;//motor 2
                                cant_pulsos=(angulo-
angulo_ant)*12;//motor 1
                                acabo_movimiento=0;
                                angulo_ant=0;
                                if(arr[3]=='+')
                                {
                                        PORTB=0b00001011;
                                }
                                else
                                {
                                        PORTB=0b00000111;
                                }
                                cant_pulsos_cont=0;
                                OCR1A=(unsigned int)velocidad;
                                while(acabo_movimiento==0)
                                {
                                        error=cant_pulsos-
cant_pulsos_cont;
                                        derror=error-error_anterior;
                                        e=(double)error;
                                        derror_double=(double)derror;

        ierror_double=(double)error+0.0001*ierror_double;
                                        valor_pwm=22.5*e +
0.33*derror_double + ierror_double;
```

```
                                        if (valor_pwm>=255)
                                        {
                                                valor_pwm=254;
                                        }
                                        if(valor_pwm<=40)
                                        {
                                                valor_pwm=40;
                                        }

                                //valor_pwm=255;
                                OCR1A=(unsigned
int)valor_pwm;//modificamos el ancho de pulso del PWM
                                error_anterior=error;


                        }
                        PORTC=0x01;
                        PORTB=0x00;
                        OCR1A=0;
                        i=0;
                }
}
ISR(INT0_vect)
{
        cant_pulsos_cont=cant_pulsos_cont+1;
        if(cant_pulsos_cont>=cant_pulsos)
        {
                acabo_movimiento=1;
        }
}
ISR(INT1_vect)
{
        PORTB=0x00;
        PORTC=0x01;
        acabo_movimiento=1;
        OCR1A=0x00;
        i=0;
}
void configuracion_puertos()
{
        DDRB=0x0F;
        DDRC=0x01;
}
void configura_twi()
{
        TWCR=0xC4;
        TWAR=0x06;
}
char recibe_dato()
{
        char p2;
        while(!(TWCR & 0x80));
```

```
        TWCR=0x84;
        while(!(TWCR & 0x80));
        p2=TWDR;
        TWCR=0x84;
        TWCR=0xC4;
        return p2;

}

void configurar_interrupcion()
{
        cli();                  //configuracion de interrupcion externa
INT0
        //MCUCR=0x03;
        MCUCR=0x0f;
        //GIFR=0x40;
        //GICR=0x40;
        GICR=0xc0;
        GIFR=0xc0;
        sei();
}
void configurar_pwm()
{
        DDRB=0x02;
        TCCR1A=0x81;
        TCCR1B=0x05;
        TCNT1=0x00;
        OCR1A=0;

}
```

```
/*
 * esclavo3_09_10.c
 *
 * Created: 09/10/2014 04:23:36 p.m.
 *  Author:
 */




#ifndef F_CPU
#define F_CPU 8000000UL // or whatever may be your frequency
#endif
#include <avr/io.h>
#include <util/delay.h>
#include <util/twi.h>
#include <avr/interrupt.h>
#include <inttypes.h>
#include <stdlib.h>

char recibe_dato();//funcion encargada de recibir los datos de TWI
void configuracion_puertos();//funcion encargada de configurar los
puertos del
void configura_twi();//función que configura la comunicación por TWI
void configurar_interrupcion();//funcion que configura la interrupcion
externa
void configurar_pwm();//funcion que permite configurar el pwm en PB1


char arr[5];//se guardara lo siguiente: dig1,dig2,dig3,signo de giro,@
int i=0;
uint32_t angulo;//angulo que se debe mover el motor
uint32_t angulo_ant=0;//angulo anterior
uint32_t cant_pulsos=0;
char acabo_movimiento=1;//'1'-> el mov ya finalizo
//'0'-> el mov todavía no ha acabado
uint32_t cant_pulsos_cont=0;
double e=0;//error proporcional
double valor_pwm=0;//indicará el ancho de pulso del pwm
int32_t error=0;
int32_t derror=0;
int32_t error_anterior=0;
double derror_double=0;
double ierror_double=0;
uint32_t velocidad;

int main(void)
{
	configuracion_puertos();
	configura_twi();
	configurar_interrupcion();
	configurar_pwm();
	while(1)
	{
```

```
                //----------------------------------------
                //Guardamos la trama recibida en el arreglo "arr"
                while((i+1)<=5)
                {
                        if(i<=2)
                        {
                                arr[i]=recibe_dato()-'0';
                        }
                        else
                        {
                                arr[i]=recibe_dato();
                        }
                        i=i+1;
                }
                //---------------------------------------
                PORTC=0x00;
                angulo=arr[0]*100+arr[1]*10+arr[2];
/*              velocidad =100;
                if(angulo==250)
                {
                        PORTC=0x01;
                }*/
                //velocidad=arr[5]*100+arr[6]*10+arr[7];
                                //cant_pulsos=(angulo-
angulo_ant)*680;//motor 3
                                //cant_pulsos=(angulo-
angulo_ant)*180;//motor 4
                                //cant_pulsos=(angulo-
angulo_ant)*1.2;//motor 2
                                cant_pulsos=(angulo-
angulo_ant)*680;//motor 1
                                acabo_movimiento=0;
                                angulo_ant=0;
                                if(arr[3]=='+')
                                {
                                        PORTB=0b00001011;
                                }
                                else
                                {
                                        PORTB=0b00000111;
                                }
                                cant_pulsos_cont=0;
                                OCR1A=(unsigned int)velocidad;
                                while(acabo_movimiento==0)
                                {
                                        error=cant_pulsos-
cant_pulsos_cont;
                                        derror=error-error_anterior;
                                        e=(double)error;
                                        derror_double=(double)derror;

        ierror_double=(double)error+0.0001*ierror_double;
                                        valor_pwm=0.127*e +
0.233*derror_double + 0*ierror_double;
```

```
                                                if (valor_pwm>=255)
                                                {
                                                        valor_pwm=254;
                                                }
                                                if(valor_pwm<=40)
                                                {
                                                        valor_pwm=40;
                                                }

                                        //valor_pwm=255;
                                        OCR1A=(unsigned
int)valor_pwm;//modificamos el ancho de pulso del PWM
                                        error_anterior=error;


                                }
                                PORTC=0x01;
                                PORTB=0x00;
                                OCR1A=0;
                                i=0;
                        }
}
ISR(INT0_vect)
{
        cant_pulsos_cont=cant_pulsos_cont+1;
        if(cant_pulsos_cont>=cant_pulsos)
        {
                acabo_movimiento=1;
        }
}
ISR(INT1_vect)
{
        PORTB=0x00;
        PORTC=0x01;
        acabo_movimiento=1;
        OCR1A=0x00;
        i=0;
}
void configuracion_puertos()
{
        DDRB=0x0F;
        DDRC=0x01;
}
void configura_twi()
{
        TWCR=0xC4;
        TWAR=0x08;
}
char recibe_dato()
{
        char p2;
        while(!(TWCR & 0x80));
```

```
        TWCR=0x84;
        while(!(TWCR & 0x80));
        p2=TWDR;
        TWCR=0x84;
        TWCR=0xC4;
        return p2;

}

void configurar_interrupcion()
{
        cli();                  //configuracion de interrupcion externa
INT0
        //MCUCR=0x03;
        MCUCR=0x0f;
        //GIFR=0x40;
        //GICR=0x40;
        GICR=0xc0;
        GIFR=0xc0;
        sei();
}
void configurar_pwm()
{
        DDRB=0x02;
        TCCR1A=0x81;
        TCCR1B=0x05;
        TCNT1=0x00;
        OCR1A=0;

}
```

```
/*
 * esclavo4_09_10.c
 *
 * Created: 09/10/2014 04:23:36 p.m.
 *  Author:
 */




#ifndef F_CPU
#define F_CPU 8000000UL // or whatever may be your frequency
#endif
#include <avr/io.h>
#include <util/delay.h>
#include <util/twi.h>
#include <avr/interrupt.h>
#include <inttypes.h>
#include <stdlib.h>

char recibe_dato();//funcion encargada de recibir los datos de TWI
void configuracion_puertos();//funcion encargada de configurar los
puertos del
void configura_twi();//función que configura la comunicación por TWI
void configurar_interrupcion();//funcion que configura la interrupcion
externa
void configurar_pwm();//funcion que permite configurar el pwm en PB1


char arr[5];//se guardara lo siguiente: dig1,dig2,dig3,signo de giro,@
int i=0;
uint32_t angulo;//angulo que se debe mover el motor
uint32_t angulo_ant=0;//angulo anterior
uint32_t cant_pulsos=0;
char acabo_movimiento=1;//'1'-> el mov ya finalizo
//'0'-> el mov todavía no ha acabado
uint32_t cant_pulsos_cont=0;
double e=0;//error proporcional
double valor_pwm=0;//indicará el ancho de pulso del pwm
int32_t error=0;
int32_t derror=0;
int32_t error_anterior=0;
double derror_double=0;
double ierror_double=0;
uint32_t velocidad;

int main(void)
{
        configuracion_puertos();
        configura_twi();
        configurar_interrupcion();
        configurar_pwm();
        while(1)
        {
```

```
                //---------------------------------------
                //Guardamos la trama recibida en el arreglo "arr"
                while((i+1)<=5)
                {
                        if(i<=2)
                        {
                                arr[i]=recibe_dato()-'0';
                        }
                        else
                        {
                                arr[i]=recibe_dato();
                        }
                        i=i+1;
                }
                //--------------------------------------
                PORTC=0x00;
                angulo=arr[0]*100+arr[1]*10+arr[2];
/*              velocidad =100;
                if(angulo==250)
                {
                        PORTC=0x01;
                }*/
                //velocidad=arr[5]*100+arr[6]*10+arr[7];
                                //cant_pulsos=(angulo-
angulo_ant)*680;//motor 3
                                //cant_pulsos=(angulo-
angulo_ant)*180;//motor 4
                                //cant_pulsos=(angulo-
angulo_ant)*1.2;//motor 2
                                cant_pulsos=(angulo-
angulo_ant)*180;//motor 1
                                acabo_movimiento=0;
                                angulo_ant=0;
                                if(arr[3]=='+')
                                {
                                        PORTB=0b00001011;
                                }
                                else
                                {
                                        PORTB=0b00000111;
                                }
                                cant_pulsos_cont=0;
                                OCR1A=(unsigned int)velocidad;
                                while(acabo_movimiento==0)
                                {
                                        error=cant_pulsos-
cant_pulsos_cont;
                                        derror=error-error_anterior;
                                        e=(double)error;
                                        derror_double=(double)derror;

        ierror_double=(double)error+0.0001*ierror_double;
                                        valor_pwm=0.009*e +
0.0038*derror_double + 0*ierror_double;
```

```
                                                    if (valor_pwm>=255)
                                                    {
                                                            valor_pwm=254;
                                                    }
                                                    if(valor_pwm<=40)
                                                    {
                                                            valor_pwm=40;
                                                    }

                                            //valor_pwm=255;
                                            OCR1A=(unsigned
int)valor_pwm;//modificamos el ancho de pulso del PWM
                                            error_anterior=error;


                                    }
                                    PORTC=0x01;
                                    PORTB=0x00;
                                    OCR1A=0;
                                    i=0;
                            }
}
ISR(INT0_vect)
{
        cant_pulsos_cont=cant_pulsos_cont+1;
        if(cant_pulsos_cont>=cant_pulsos)
        {
                acabo_movimiento=1;
        }
}
ISR(INT1_vect)
{
        PORTB=0x00;
        PORTC=0x01;
        acabo_movimiento=1;
        OCR1A=0x00;
        i=0;
}
void configuracion_puertos()
{
        DDRB=0x0F;
        DDRC=0x01;
}
void configura_twi()
{
        TWCR=0xC4;
        TWAR=0x0A;
}
char recibe_dato()
{
        char p2;
        while(!(TWCR & 0x80));
```

```
        TWCR=0x84;
        while(!(TWCR & 0x80));
        p2=TWDR;
        TWCR=0x84;
        TWCR=0xC4;
        return p2;

}

void configurar_interrupcion()
{
        cli();                //configuracion de interrupcion externa
INT0
        //MCUCR=0x03;
        MCUCR=0x0f;
        //GIFR=0x40;
        //GICR=0x40;
        GICR=0xc0;
        GIFR=0xc0;
        sei();
}
void configurar_pwm()
{
        DDRB=0x02;
        TCCR1A=0x81;
        TCCR1B=0x05;
        TCNT1=0x00;
        OCR1A=0;

}
```

Código Form1

```
Public Class Form1
    Dim StrBufferOut As String
    Dim StrBufferIn As String
    Dim count As Double
    Dim signo1, signo2, signo3, signo4, tramafin, tramap1, tramap2, tramap3,
tramap4 As String

    '-----------------------------------------
    '-- Condiciones Iniciales del Programa --
    '-----------------------------------------
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        StrBufferOut = ""                  '-- Inicializo las varibles a usar
        StrBufferIn = ""
        btndirecta.Enabled = False
        btnTrayectoria.Enabled = False
        OvalShape1.FillColor = Color.Red
        btnConectar.Enabled = False        '-- Inicialmente no activadas
        tmrTimer.Enabled = False
        cboTrayectoria.SelectedIndex = 0   '-- Muestra valor principal por
defecto

        txtIniX.Text = Convert.ToString(My.Settings.SaveX)
        txtIniY.Text = Convert.ToString(My.Settings.SaveY)
        txtIniZ.Text = Convert.ToString(My.Settings.SaveZ)

        saveang1.Text = Convert.ToString(My.Settings.SaveAng1)
        saveang2.Text = Convert.ToString(My.Settings.SaveAng2)
        saveang3.Text = Convert.ToString(My.Settings.SaveAng3)
        saveang4.Text = Convert.ToString(My.Settings.SaveAng4)


        nPuntos = 0
        tmrTimer.Enabled = False


    End Sub
    '-----------------------------------
    '-- Verificar Puerto de Conexión --
    '-----------------------------------
    Private Sub btnDeterminarConexion_Click(sender As Object, e As EventArgs)
Handles btnDeterminarConexion.Click
        cboPuertos.Items.Clear() '-- Limpio el combobox
        For Each PuertoDisponible As String In My.Computer.Ports.SerialPortNames
            cboPuertos.Items.Add(PuertoDisponible) '-- Busco puertos en la PC
        Next
        If cboPuertos.Items.Count > 0 Then
            cboPuertos.Text = cboPuertos.Items(0)
            MessageBox.Show("SELECCIONE EL PUERTO A TRABAJAR")  '-- Agrega los
puerto serial disponible
            btnConectar.Enabled = True
        Else
            MessageBox.Show("NINGUN PUERTO DISPONIBLE")
            btnConectar.Enabled = False
            cboPuertos.Items.Clear()
            cboPuertos.Text = ("                    ")
        End If
    End Sub
    '---------------------------------------------------------
    '-- Conexión y Desconexión de la Comunicación Serial --
    '---------------------------------------------------------
```

```
    Private Sub btnConectar_Click(sender As Object, e As EventArgs) Handles btnConectar.Click
        If btnConectar.Text = "Conectar Sistema" Then
            spPuertos.PortName = cboPuertos.Text    '-- Ubica puerto
            btnConectar.Text = "Desconectar Sistema"
            tmrTimer.Enabled = True
            spPuertos.Open() '-- Activa Envio de Datos
            OvalShape2.FillColor = Color.Green '-- Muestra Indicadores
            OvalShape1.FillColor = Color.WhiteSmoke
            btndirecta.Enabled = True
            btnTrayectoria.Enabled = True
        ElseIf btnConectar.Text = "Desconectar Sistema" Then
            btnConectar.Text = "Conectar Sistema"

            '-- Desactivo Envio de Datos
            OvalShape1.FillColor = Color.Red '-- Muestra Indicadores
            OvalShape2.FillColor = Color.WhiteSmoke

            btnTrayectoria.Enabled = False


            tramap1 = conversoratrama(My.Settings.SaveAng1)
            tramap2 = conversoratrama(My.Settings.SaveAng2)
            tramap3 = conversoratrama(My.Settings.SaveAng3)
            tramap4 = conversoratrama(My.Settings.SaveAng4)
            tramafin = tramap1 & tramap2 & tramap3 & tramap4
            mostrarserial.Text = tramafin

            spPuertos.DiscardInBuffer()
            StrBufferOut = tramafin
            spPuertos.Write(StrBufferOut)


            btndirecta.Enabled = False

            '------------------------------ Actualizo valores en pantalla inicial
            txtIniX.Text = 0.84
            txtIniY.Text = 0
            txtIniZ.Text = 0.175

            '------------------------------ Actualizo valores en pantalla de directa

            Form2.txtX_i.Text = 0.84
            Form2.txtY_i.Text = 0
            Form2.txtZ_i.Text = 0.175

            My.Settings.SaveX = 0.84
            My.Settings.SaveY = 0
            My.Settings.SaveZ = 0.175

            My.Settings.SaveAng1 = 0
            My.Settings.SaveAng2 = 0
            My.Settings.SaveAng3 = 0
            My.Settings.SaveAng4 = 0
            spPuertos.Close()
            tmrTimer.Enabled = False
        End If
    End Sub
    '---------------------------------------------------
    '-- Formulario de Movimiento a través de ángulos --
    '---------------------------------------------------
```

```
    Private Sub btndirecta_Click(sender As Object, e As EventArgs) Handles
btndirecta.Click
        Form2.Show()
    End Sub
    '-----------------------------------------------------------
    '-- Varibles de btnTrayectoria --
    '-----------------------------------------------------------
    Dim qo, qov, qf, qfv, qa, qaf, tf As Double
    Dim zFinal, zInicial, xFinal, xinicial, yFinal, yinicial As Double
    Dim a0, a1, a2, a3 As Double
    Dim xant, yant, zant As Double
    Dim t1, tf1, tdef As Double
    Dim x, y, z As Double
    Dim xv, yv, zv As Double
    Dim traycubv As Double


    Private Sub btnTrayectoria_Click(sender As Object, e As EventArgs) Handles
btnTrayectoria.Click
        PictureBox1.Image = Nothing
        'Stop

        qo = Convert.ToDouble(txtIniZ.Text)      '-- Posición Inicial z
        qf = Convert.ToDouble(txtFinalZ.Text)    '-- Posición Final z
        xinicial = txtIniX.Text
        yinicial = Convert.ToDouble(txtIniY.Text)
        xFinal = Convert.ToDouble(txtFinalX.Text)
        yFinal = Convert.ToDouble(txtFinalY.Text)

        '-- Se define tiempo de trayectoria
        zFinal = qf      '-- Hasta donde tiene que llegar
        zInicial = qo    '-- Desde donde empieza
        '-- definición de X y Y

        tdef = 2
        '-- Actualización de Posición Inicial
        t1 = tf1
        xant = Convert.ToDouble(txtIniX.Text)
        yant = Convert.ToDouble(txtIniY.Text)
        zant = zInicial

        '-- Cálculo de coeficientes
        a0 = qo
        a1 = 0
        a2 = (3 * (qf - qo)) / (tdef ^ 2)
        a3 = ((-2) * (qf - qo)) / (tdef ^ 3)

        count = 0
        Timer1.Enabled = True     '-- Habilito muestreo
    End Sub
    Dim m1, m2, m3, m4 As Double

    Dim vel1, vel2, vel3 As Double
    Dim nPuntos As Double

    '-----------------------------------------------------------
    '-- Muestreo de Señal - Gráfica de la Función --
    '-----------------------------------------------------------
    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

        Dim temp_x, temp_y, temp_z As Double
        Dim temp_xant, temp_yant, temp_zant As Double
```

```
        Dim countfinal
        countfinal = 2
        'Dim m1, m2, m3 As Double

        t1 = t1 + 1
        z = (a0 + a1 * (t1 - tf1) + a2 * (t1 - tf1) ^ 2 + a3 * (t1 - tf1) ^ 3) '-
- Punto Final de cada Muestreo
        zv = a1 + 2 * a2 * (t1 - tf1) + 3 * a3 * (t1 - tf1) ^ 2

        traycubv = a1 * t1 + a2 * t1 + a3 * t1 ^ 2

        If (count < countfinal) Then
            count = count + 1
            'Stop
            '-- Actualización de Parámetros X,Y,Z actuales

            x = t1 * (xFinal - xinicial) / tdef + xinicial
            xv = (xFinal - xinicial) / tdef

            y = t1 * (yFinal - yinicial) / tdef + yinicial
            yv = (xFinal - xinicial) / tdef

            temp_x = x * (5 / 0.3)   '-- Escala
            temp_y = y * (5 / 0.3)  '-- Escala
            temp_z = z * 20 - 7  '-- Escala
            temp_xant = xant * (5 / 0.3)
            temp_yant = yant * (5 / 0.3)
            temp_zant = zant * 20 - 7

            nPuntos = nPuntos + 1

            Call Dibuja(temp_xant, temp_yant, temp_zant, temp_x, temp_y, temp_z)

            '-- Actualización de Parámetros X,Y,Z anteriores
            xant = x
            yant = y
            zant = z

            '-- Conversion mediante cinematica inversa
            Call cinematicaInversa(x, y, z, m1, m2, m3, m4)


            '-- Aplicacion de jacobiano inverso
            Call calculaJacobianoInverso(m1, m2, m3, xv, yv, zv, vel1, vel2,
vel3)


            '-- Mostrar datos en el TextBox
            My.Settings.SaveX = x
            My.Settings.SaveY = y
            My.Settings.SaveZ = z

            txtIniX.Text = x
            txtIniY.Text = y
            txtIniZ.Text = z

            Form4.txtXform4.Text = x
            Form4.txtYform4.Text = y
            Form4.txtZform4.Text = z
        End If
```

```
        'MsgBox("El Brazo 5GL se ecuentra en (x,y,z): ( " & x & ", " & y & ", " &
z & ")")
    End Sub
    '-----------------------------------------------------------
    '-- Variables Globales del Procedimiento Dibuja --

    Dim funciones As String
    Dim lapiz1 As New Pen(Color.Black)
    Dim lapiz2 As New Pen(Color.Red)
    Dim lapiz3 As New Pen(Color.Blue)
    Dim lapiz4 As New Pen(Color.Green, 5)

    Dim a As Single
    Dim k As Single
    '-----------------------------------------------------------
    Sub Dibuja(xant As Single, yant As Single, zant As Single, x As Single, y As
Single, z As Single)

        Dim objGraficos As System.Drawing.Graphics '-- Creación del objeto
Gráfico
        objGraficos = Me.PictureBox1.CreateGraphics


        Dim xcentro As Single = PictureBox1.Width / 2    '-- Calulo del punto
medio
        Dim ycentro As Single = PictureBox1.Height / 2

        Dim ant As Single       '-- Varibles para almacenar valores anteriores
        Dim kant As Single


        a = y * 0.7071 * 15 '-- Factor con      y*cos(45)*39
        k = a

        ant = yant * 0.7071 * 15
        kant = ant

        objGraficos.TranslateTransform(xcentro, ycentro) '-- Punto medio (cero
cero cero)
        '-- Línea horizontal
        objGraficos.DrawLine(lapiz1, 0, 0, xcentro * 2, 0)
        '-- Línea vertical
        objGraficos.DrawLine(lapiz2, 0, 0, 0, ycentro * -2)
        '-- Eje Z
        objGraficos.DrawLine(lapiz3, 0, 0, xcentro * -1, ycentro * 2)
        '-- Gráfica de puntos
        objGraficos.DrawLine(lapiz4, 15 * xant - kant, -15 * zant + kant, 15 * x
- k, -15 * z + k) '-- X2D = 39*x-k , -39*z+k

    End Sub
    '------------------------------
    '-- Envio de Datos al motor --
    '------------------------------

    '-----------------------------------------------------------
    '-- Preparación de Puertos para Transmisión de Datos --
    '-----------------------------------------------------------
    Private Sub tmrTimer_Tick(sender As Object, e As EventArgs) Handles
tmrTimer.Tick
        StrBufferIn = spPuertos.ReadExisting      '-- Lee el puerto
        If StrBufferIn <> "" Then
```

```
            mostrarserial.Text = StrBufferIn
            StrBufferIn = ""
            spPuertos.DiscardInBuffer()

        End If
    End Sub


    Private Sub calculaJacobianoInverso(q1 As Double, q2 As Double, q3 As Double,
v1 As Double, v2 As Double, v3 As Double, novoVel_1 As Double, novoVel_2 As
Double, novoVel_3 As Double)
        Dim matInv(,) As Double
        Dim matNovoVel(,) As Double
        Dim matJacobiano(,) As Double
        Dim matVelFin(,) As Double

        ReDim matInv(3, 3)
        ReDim matNovoVel(3, 1)
        ReDim matJacobiano(3, 3)
        ReDim matVelFin(3, 1)

        q1 = q1 * Math.PI / 180
        q2 = q2 * Math.PI / 180
        q3 = q3 * Math.PI / 180

        matJacobiano = calcularJacobiano(q1, q2, q3)
        matVelFin(0, 0) = v1
        matVelFin(1, 0) = v2
        matVelFin(2, 0) = v3


        'iNVERSA
        matInv = calcularInversa(matJacobiano)
        matNovoVel = cacularMultiplicacion(3, 3, matInv, 3, 1, matVelFin)

        'JACOBIANO INVERSO
        novoVel_1 = matNovoVel(0, 0)
        novoVel_2 = matNovoVel(1, 0)
        novoVel_3 = matNovoVel(2, 0)
        'Stop
        vel1 = novoVel_1
        vel2 = novoVel_2
        vel3 = novoVel_3

        vel1 = vel1 * 60 / Math.PI
        vel2 = vel2 * 60 / Math.PI
        vel3 = vel3 * 60 / Math.PI

        txtMostarJ1.Text = vel1
        txtMostarJ2.Text = vel2
        txtMostarJ3.Text = vel3

    End Sub
    Private Function cacularMultiplicacion(ByVal a As Integer, ByVal b As
Integer, ByVal matA As Double(,), ByVal c As Integer, ByVal d As Integer, ByVal
matB As Double(,))
        Dim matr As Double(,)
        Dim temp, sum As Double
        sum = 0
        ReDim matr(a, d)

        'Stop
```

```
        For fil = 0 To a - 1 'Fila 1'
            For col = 0 To d - 1
                For k1 = 0 To a - 1 'Fila de matriz 2
                    sum = sum + matA(fil, k1) * matB(k1, col)
                Next
                temp = Math.Round(sum, 7)
                matr(fil, col) = temp
                sum = 0
            Next
        Next
        Return matr
    End Function

    '--   q1 = ang1, q2 = ang2, q3 = ang3 ---  Valores del Jacobiano
    Private Function calcularJacobiano(q1 As Double, q2 As Double, q3 As Double)
        Dim l2, l3, a As Double
        Dim jac(,) As Double
        ReDim jac(3, 3)

        l2 = 0.275
        l3 = 0.275

        a = l2 * Math.Cos(q2) - l3 * Math.Sin(q3) * Math.Sin(q2) + l3 *
Math.Cos(q3) * Math.Cos(q2)

        jac(0, 0) = -a * Math.Sin(q1)
        jac(0, 1) = (-l2 * Math.Sin(q2) * Math.Cos(q1) - l3 * Math.Sin(q3) *
Math.Cos(q2) * Math.Cos(q1) - l3 * Math.Cos(q3) * Math.Sin(q2) * Math.Cos(q1))
        jac(0, 2) = -l3 * Math.Cos(q1) * Math.Sin(q2) * Math.Cos(q3) - l3 *
Math.Sin(q3) * Math.Cos(q2) * Math.Cos(q1)

        jac(1, 0) = a * Math.Cos(q1)
        jac(1, 1) = (-l2 * Math.Sin(q1) * Math.Sin(q2) - l3 * Math.Sin(q3) *
Math.Cos(q2) * Math.Sin(q1) - l3 * Math.Cos(q3) * Math.Sin(q2) * Math.Sin(q1))
        jac(1, 2) = -l3 * Math.Sin(q1) * Math.Sin(q2) * Math.Cos(q3) - l3 *
Math.Sin(q3) * Math.Cos(q2) * Math.Sin(q1)

        jac(2, 0) = 0
        jac(2, 1) = l2 * Math.Cos(q2) - l3 * Math.Sin(q3) * Math.Sin(q2) + l3 *
Math.Cos(q3) * Math.Cos(q2)
        jac(2, 2) = l3 * Math.Cos(q3) * Math.Cos(q2) - l3 * Math.Sin(q3) *
Math.Sin(q2)

        Return jac
    End Function

    Private Function calcularInversa(ByVal mat As Double(,))

        Dim d As Double
        Dim matTemp1 As Double(,)
        Dim matFinal As Double(,)
        ReDim matTemp1(3, 3)
        ReDim matFinal(3, 3)
        Dim i As Double
        Dim matTemp0 As Double(,)
        ReDim matTemp0(3, 3)

        d = calcularDeterminante(mat)

        i = 1 / d
        matTemp1 = calcularAdjunta(mat)
```

```
        matTemp0 = calcularTranspuesta(matTemp1)
        matFinal = constantePorMatriz(i, matTemp0)

        Return matFinal

    End Function
    Private Function conversoratrama(ByVal ang As Double)
        Dim stre As String
        If (ang >= 0) Then
            stre = Convert.ToString(ang) & "-" & "@"
            If (ang < 100) Then
                stre = "0" & Convert.ToString(ang) & "-" & "@"
                If (ang < 10) Then
                    stre = "00" & Convert.ToString(ang) & "-" & "@"
                End If
            End If
        End If
        If (ang < 0) Then
            ang = -ang
            stre = Convert.ToString(ang) & "+" & "@"
            If (ang < 100) Then
                stre = "0" & Convert.ToString(ang) & "+" & "@"
                If (ang < 10) Then
                    stre = "00" & Convert.ToString(ang) & "+" & "@"
                End If
            End If
        End If
        Return stre
    End Function
    Private Function calcularDeterminante(ByVal mat As Double(,))
        Dim matTemp2 As Double(,)
        ReDim matTemp2(5, 5)
        Dim prim1, prim2, prim3 As Double
        Dim sec1, sec2, sec3 As Double
        Dim mul1, mul2 As Double
        Dim deter As Double

        matTemp2(0, 0) = mat(0, 0) 'Primera fila
        matTemp2(0, 1) = mat(0, 1)
        matTemp2(0, 2) = mat(0, 2)
        matTemp2(1, 0) = mat(1, 0) 'Segunda fila
        matTemp2(1, 1) = mat(1, 1)
        matTemp2(1, 2) = mat(1, 2)
        matTemp2(2, 0) = mat(2, 0) 'Tercera fila
        matTemp2(2, 1) = mat(2, 1)
        matTemp2(2, 2) = mat(2, 2)
        matTemp2(3, 0) = mat(0, 0) 'Cuarta fila
        matTemp2(3, 1) = mat(0, 1)
        matTemp2(3, 2) = mat(0, 2)
        matTemp2(4, 0) = mat(1, 0) 'Quinta fila
        matTemp2(4, 1) = mat(1, 1)
        matTemp2(4, 2) = mat(1, 2)

        prim1 = matTemp2(0, 0) * matTemp2(1, 1) * matTemp2(2, 2)
        prim2 = matTemp2(1, 0) * matTemp2(2, 1) * matTemp2(3, 2)
        prim3 = matTemp2(2, 0) * matTemp2(3, 1) * matTemp2(4, 2)
        mul1 = prim1 + prim2 + prim3
        sec1 = matTemp2(0, 2) * matTemp2(1, 1) * matTemp2(2, 0)
        sec2 = matTemp2(1, 2) * matTemp2(2, 1) * matTemp2(3, 0)
        sec3 = matTemp2(2, 2) * matTemp2(3, 1) * matTemp2(4, 0)
        mul2 = sec1 + sec2 + sec3
        deter = mul1 - mul2
```

```
        Return deter
    End Function
    Private Function calcularAdjunta(ByVal mat As Double(,))
        Dim matTrans As Double(,)
        ReDim matTrans(3, 3)

        matTrans(0, 0) = mat(1, 1) * mat(2, 2) - mat(2, 1) * mat(1, 2) 'Primera
fila
        matTrans(0, 1) = -1 * (mat(1, 0) * mat(2, 2) - mat(2, 0) * mat(1, 2))
        matTrans(0, 2) = mat(1, 0) * mat(2, 1) - mat(2, 0) * mat(1, 1)

        matTrans(1, 0) = -1 * (mat(0, 1) * mat(2, 2) - mat(2, 1) * mat(0, 2))
'Segunda fila
        matTrans(1, 1) = mat(0, 0) * mat(2, 2) - mat(2, 0) * mat(0, 2)
        matTrans(1, 2) = -1 * (mat(0, 0) * mat(2, 1) - mat(2, 0) * mat(0, 1))

        matTrans(2, 0) = mat(0, 1) * mat(1, 2) - mat(1, 1) * mat(0, 2) 'Tercera
fila
        matTrans(2, 1) = -1 * (mat(0, 0) * mat(1, 2) - mat(1, 0) * mat(0, 2))
        matTrans(2, 2) = mat(0, 0) * mat(1, 1) - mat(1, 0) * mat(0, 1)

        Return matTrans
    End Function
    Private Function constantePorMatriz(ByVal cte As Double, ByVal mat As
Double(,))
        Dim matTemp3 As Double(,)
        ReDim matTemp3(3, 3)

        matTemp3(0, 0) = mat(0, 0) * cte 'Primera fila
        matTemp3(0, 1) = mat(0, 1) * cte
        matTemp3(0, 2) = mat(0, 2) * cte
        matTemp3(1, 0) = mat(1, 0) * cte 'Segunda fila
        matTemp3(1, 1) = mat(1, 1) * cte
        matTemp3(1, 2) = mat(1, 2) * cte
        matTemp3(2, 0) = mat(2, 0) * cte 'Tercera fila
        matTemp3(2, 1) = mat(2, 1) * cte
        matTemp3(2, 2) = mat(2, 2) * cte

        Return matTemp3
    End Function
    Private Function calcularTranspuesta(ByVal mat(,) As Double)
        Dim matTemp3 As Double(,)
        ReDim matTemp3(3, 3)

        matTemp3(0, 0) = mat(0, 0)  'Primera fila
        matTemp3(0, 1) = mat(1, 0)
        matTemp3(0, 2) = mat(2, 0)
        matTemp3(1, 0) = mat(0, 1)  'Segunda fila
        matTemp3(1, 1) = mat(1, 1)
        matTemp3(1, 2) = mat(2, 1)
        matTemp3(2, 0) = mat(0, 2)  'Tercera fila
        matTemp3(2, 1) = mat(1, 2)
        matTemp3(2, 2) = mat(2, 2)

        Return matTemp3

    End Function
    Private Sub btncalibrar_Click(sender As Object, e As EventArgs) Handles
btncalibrar.Click
        Form3.Show()
    End Sub
```

```
    Private Sub cinematicaInversa(px As Double, py As Double, pz As Double,
motor1 As Double, motor2 As Double, motor3 As Double, motor4 As Double)
        Dim px2, py2, px3, py3, pz3, h, cosq3, l2, l3, l4, d, q2p, q3p As Double
        Dim str1, str2, str3, str4, trama1, trama2, trama3, trama4 As String
        Dim valid As Boolean
        l2 = 28
        l3 = 28
        l4 = 28

        px = Math.Round(px * 100)
        py = Math.Round(py * 100)
        pz = Math.Round(pz * 100)

        If x = 0 Then
            motor1 = 0
        Else
            motor1 = Math.Atan2(y, x)
        End If

        motor2 = 25 * (Math.PI / 180)

        px2 = l2 * Math.Cos(motor1) * Math.Cos(motor2)
        py2 = l2 * Math.Sin(motor1) * Math.Cos(motor2)
        pz3 = pz - l2 * Math.Sin(motor2) - 17.5
        px3 = px - px2
        py3 = py - py2
        d = (px3 ^ 2 + py3 ^ 2 + pz3 ^ 2) ^ 0.5
        h = 0

        valid = True

        While (valid)
            '(d > 55.8) & (d < 19) & (Math.Abs(px3) < 5) & (Math.Abs(py3) < 5)
            If (d > 55.8) Then
                If (d < 19) Then
                    If (Math.Abs(px3) < 5) Then
                        If (Math.Abs(py3) < 5) Then
                            valid = True
                        Else
                            valid = False
                        End If
                    Else
                        valid = False
                    End If
                    valid = False
                End If
            Else
                valid = False
            End If
            h = h + 1

            motor2 = motor2 + Math.PI / 180
            px2 = l2 * Math.Cos(motor1) * Math.Cos(motor2)
            py2 = l2 * Math.Sin(motor1) * Math.Cos(motor2)
            pz3 = pz - l2 * Math.Sin(motor2) - 17.5
            px3 = px - px2
            py3 = py - py2
            d = (px3 ^ 2 + py3 ^ 2 + pz3 ^ 2) ^ 0.5
        End While
```

```
        cosq3 = ((px3 ^ 2 + py3 ^ 2 + (pz3) ^ 2 - l3 ^ 2 - l4 ^ 2) / (2 * l3 *
l4))
        q3p = Math.Atan((((1 - cosq3 ^ 2)) ^ 0.5) / cosq3)
        q2p = Math.Atan(pz3 / ((px3 ^ 2 + py3 ^ 2) ^ 0.5)) - Math.Atan(l4 *
Math.Sin(q3p) / (l3 + l4 * Math.Cos(q3p)))
        q2p = q2p - motor2

        'Convertir a grados
        motor1 = motor1 * 180 / Math.PI
        motor2 = motor2 * 180 / Math.PI
        motor3 = q2p * 180 / Math.PI
        motor4 = q3p * 180 / Math.PI

        'Redondear valores a cero decimales
        motor1 = Math.Round(motor1, 0)
        motor2 = Math.Round(motor2, 0)
        motor3 = Math.Round(motor3, 0)
        motor4 = Math.Round(motor4, 0)
        'motor2v = Math.Truncate(motor2v)

        'Stop
        txtMostrar_m1.Text = motor1
        txtMostrar_m2.Text = motor2
        txtMostrar_m3.Text = motor3
        txtMostrar_m4.Text = motor4

        If (False) Then
            My.Settings.SaveAng1 = motor1
            My.Settings.SaveAng2 = motor2
            My.Settings.SaveAng3 = motor3
            My.Settings.SaveAng4 = motor4
        End If
        str1 = analisisangulof11(motor1)
        str2 = analisisangulof12(motor2)
        str3 = analisisangulof13(motor3)
        str4 = analisisangulof14(motor4)

        motor1 = Convert.ToDouble(signo1 & str1)
        motor2 = Convert.ToDouble(signo2 & str2)
        motor3 = Convert.ToDouble(signo3 & str3)
        motor4 = Convert.ToDouble(signo4 & str4)
        m1 = motor1
        m2 = motor2
        m3 = motor3
        m4 = motor4


        '---------------------trama motor1
        'Stop
        If (signo1 = "+") Then
            If (motor1 > 100) Then
                trama1 = str1 & "+@"
            End If
            If (motor1 <= 100) Then
                trama1 = "0" & str1 & "+@"
                If (motor1 < 10) Then
                    trama1 = "0" & "0" & str1 & "+@"
                End If
            End If


        End If
```

```
        If (signo1 = "-") Then
            m1 = -m1
            str1 = Convert.ToString(m1)
            If (m1 > 100) Then
                trama1 = str1 & "-@"
            End If
            If (m1 <= 100) Then
                trama1 = "0" & str1 & "-@"
                If (m1 < 10) Then
                    trama1 = "0" & "0" & str1 & "-@"
                End If
            End If

        End If


        '---------------------trama motor2
        If (signo2 = "+") Then
            If (motor2 > 100) Then
                trama2 = trama1 & str2 & "+"
            End If
            If (motor2 <= 100) Then
                trama2 = trama1 & "0" & str2 & "+"
                If (motor2 < 10) Then
                    trama2 = trama1 & "0" & "0" & str2 & "+"
                End If
            End If
        End If

        If (signo2 = "-") Then
            m2 = -m2
            str2 = Convert.ToString(m2)
            If (m2 > 100) Then
                trama2 = trama1 & str2 & "-"
            End If
            If (m2 <= 100) Then
                trama2 = trama1 & "0" & str2 & "-"
                If (m2 < 10) Then
                    trama2 = trama1 & "0" & "0" & str2 & "-"
                End If
            End If
        End If

        '---------------------trama motor3
        If (signo3 = "+") Then
            If (motor3 > 100) Then
                trama3 = trama2 & "@" & str3 & "+"
            End If
            If (motor3 <= 100) Then
                trama3 = trama2 & "@" & "0" & str3 & "+"
                If (motor3 < 10) Then
                    trama3 = trama2 & "@" & "0" & "0" & str3 & "+"
                End If
            End If
        End If

        If (signo3 = "-") Then
            m3 = -m3
            str3 = Convert.ToString(m3)
            If (m3 > 100) Then
                trama3 = trama2 & "@" & str3 & "-"
```

```
            End If
            If (m3 <= 100) Then
                trama3 = trama2 & "@" & "0" & str3 & "-"
                If (m3 < 10) Then
                    trama3 = trama2 & "@" & "0" & "0" & str3 & "-"
                End If
            End If
        End If

        '---------------------trama motor4

        If (signo4 = "+") Then
            If (motor4 > 100) Then
                trama4 = trama3 & "@" & str4 & "+" & "@"
            End If
            If (motor4 <= 100) Then
                trama4 = trama3 & "@" & "0" & str4 & "+" & "@"
                If (motor4 < 10) Then
                    trama4 = trama3 & "@" & "0" & "0" & str4 & "+" & "@"
                End If
            End If
        End If

        If (signo4 = "-") Then
            m4 = -m4
            str4 = Convert.ToString(m4)
            If (m4 > 100) Then
                trama4 = trama3 & "@" & str4 & "-" & "@"
            End If
            If (m4 <= 100) Then
                trama4 = trama3 & "@" & "0" & str4 & "-" & "@"
                If (m4 < 10) Then
                    trama4 = trama3 & "@" & "0" & "0" & str4 & "-" & "@"
                End If
            End If
        End If

        spPuertos.DiscardInBuffer()
        StrBufferOut = trama4
        spPuertos.Write(StrBufferOut)

        mostrarserial.Text = trama4

    End Sub

    Function analisisangulof11(ByVal number1 As Double)
        Dim pos1 As Double
        Dim cuadrante1, d As Double
        Dim mover, dis1, dis2 As Double

        If (number1 >= 0) Then
            pos1 = number1
        Else
            pos1 = number1 * -1
        End If
        If ((number1 >= 0) And (number1 <= 90)) Then
            cuadrante1 = 1
            pos1 = number1
        Else
            If ((number1 <= 180) And (number1 > 90)) Then
                cuadrante1 = 2
                pos1 = number1
```

```
            Else
                If (((number1 <= 270) And (number1 > 180)) Or ((number1 >= -180)
And (number1 <= -90))) Then
                    cuadrante1 = 3
                    pos1 = 360 + number1
                Else
                    cuadrante1 = 4
                    pos1 = 360 + number1
                End If
            End If
        End If
        d = Math.Abs(cuadrante1 - My.Settings.SaveCuad1)
        If (d = 3) Then
            If (My.Settings.SaveCuad1 = 4) Then
                mover = Math.Abs(My.Settings.SaveAng1) + number1
                signo1 = "+"
            Else
                mover = My.Settings.SaveAng1 + Math.Abs(number1)
                signo1 = "-"
            End If
        Else
            If (d <= 1) Then
                If (pos1 < My.Settings.SaveAng1_pos) Then
                    mover = My.Settings.SaveAng1_pos - pos1
                    signo1 = "-"
                Else
                    mover = pos1 - My.Settings.SaveAng1_pos
                    signo1 = "+"
                End If
            Else
                If (My.Settings.SaveCuad1 < cuadrante1) Then
                    dis1 = Math.Abs(number1) + My.Settings.SaveAng1
                    dis2 = pos1 - My.Settings.SaveAng1
                    If (dis1 < dis2) Then
                        mover = dis1
                        signo1 = "-"
                    Else
                        mover = dis2
                        signo1 = "+"
                    End If
                Else
                    dis1 = Math.Abs(My.Settings.SaveAng1) + number1
                    dis2 = My.Settings.SaveAng1_pos - number1
                    If (dis1 <= dis2) Then
                        mover = dis1
                        signo1 = "+"
                    Else
                        mover = dis2
                        signo1 = "-"
                    End If
                End If

            End If

        End If
        My.Settings.SaveAng1 = number1
        My.Settings.SaveAng1_pos = pos1
        My.Settings.SaveCuad1 = cuadrante1
        saveang1.Text = Convert.ToString(My.Settings.SaveAng1)
        mover = Convert.ToString(mover)
        Return mover
```

```
    End Function

    Function analisisangulof12(ByVal number2 As Double)
        Dim mover, aux1, aux2 As Double

        aux1 = My.Settings.SaveAng2
        aux2 = number2 - aux1
        If (aux2 >= 0) Then
            signo2 = "+"
            mover = aux2
        Else
            signo2 = "-"
            mover = aux2 * -1
        End If
        My.Settings.SaveAng2 = number2

        saveang2.Text = Convert.ToString(My.Settings.SaveAng2)
        mover = Convert.ToString(mover)
        Return mover

    End Function

    Function analisisangulof13(ByVal number3 As Double)
            Dim mover, aux1, aux2 As Double

        aux1 = My.Settings.SaveAng3
        aux2 = number3 - aux1
        If (aux2 >= 0) Then
            signo3 = "+"
            mover = aux2
        Else
            signo3 = "-"
            mover = aux2 * -1
        End If
        My.Settings.SaveAng3 = number3

        saveang3.Text = Convert.ToString(My.Settings.SaveAng3)
        mover = Convert.ToString(mover)
        Return mover

    End Function

    Function analisisangulof14(ByVal number4 As Double)
        Dim mover, aux1, aux2 As Double

        aux1 = My.Settings.SaveAng4
        aux2 = number4 - aux1
        If (aux2 >= 0) Then
            signo4 = "+"
            mover = aux2
        Else
            signo4 = "-"
            mover = aux2 * -1
        End If
        My.Settings.SaveAng4 = number4

        saveang4.Text = Convert.ToString(My.Settings.SaveAng4)
        mover = Convert.ToString(mover)
        Return mover

    End Function
End Class
```

Código Form2

```
Public Class Form2
    Dim StrBufferOut As String
    Dim StrBufferIn As String
    Dim anguloenv1, anguloenv2, anguloenv3 As String
    Dim signo1, signo2, signo3, signo4 As String

    Private Sub salirform2_Click(sender As Object, e As EventArgs) Handles
salirform2.Click
        Me.Close()
    End Sub

    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        cboMotor1.SelectedIndex = 0
        cboMotor2.SelectedIndex = 0
        cboMotor3.SelectedIndex = 0
        cboMotor4.SelectedIndex = 0

        txtMotor1.Text = 0
        txtMotor2.Text = 0
        txtMotor3.Text = 0
        txtMotor4.Text = 0

        'nPuntos = 0

        Control1.Text = Convert.ToString(My.Settings.SaveAng1)
        Control2.Text = Convert.ToString(My.Settings.SaveAng2)
        Control3.Text = Convert.ToString(My.Settings.SaveAng3)
        Control4.Text = Convert.ToString(My.Settings.SaveAng4)

    End Sub

    Private Sub Realizadirecta_Click(sender As Object, e As EventArgs) Handles
Realizadirecta.Click
        Dim cadena1 As String
        Dim cadena2 As String
        Dim cadena3 As String
        Dim cadena4 As String
        Dim number1 As Double
        Dim number2 As Double
        Dim number3 As Double
        Dim number4 As Double
        Dim angulostr1, angulostr2, angulostr3, angulostr4



        cadena1 = cboMotor1.Text & txtMotor1.Text
        number1 = Convert.ToDouble(cadena1)

        cadena2 = cboMotor2.Text & txtMotor2.Text
        number2 = Convert.ToDouble(cadena2)

        cadena3 = cboMotor3.Text & txtMotor3.Text
        number3 = Convert.ToDouble(cadena3)

        cadena4 = cboMotor4.Text & txtMotor4.Text
        number4 = Convert.ToDouble(cadena4)
        If (False) Then
            Stop
        End If
        angulostr1 = analisisangulo1(number1)
```

```
        angulostr2 = analisisangulo2(number2)
        angulostr3 = analisisangulo3(number3)
        angulostr4 = analisisangulo4(number4)

        angulostr1 = conversoratrama(angulostr1)
        angulostr2 = conversoratrama(angulostr2)
        angulostr3 = conversoratrama(angulostr3)
        angulostr4 = conversoratrama(angulostr4)

        'If ((number2 < 180 And number2 > -35) Or (number2 > -180 And number2 < -
145)) And (number3 > -130 And number3 < 130) Then
        Form1.spPuertos.DiscardInBuffer()
        StrBufferOut = angulostr1 & signo1 & "@" & angulostr2 & signo2 & "@" &
angulostr3 & signo3 & "@" & angulostr4 & signo4 & "@"
        Form1.spPuertos.Write(StrBufferOut) 'Envia la cadena
        'Timer5.Enabled = True

        'Form1.tmrTimer.Enabled = True
        Form1.mostrarserial.Text = StrBufferOut

        'actualiza los valores de los angulos guardados
        My.Settings.SaveAng1 = number1
        My.Settings.SaveAng2 = number2
        My.Settings.SaveAng3 = number3
        My.Settings.SaveAng4 = number4

        Dim matA(,), matB(,), matC(,), matD(,), matR(,), matF(,), matE(,),
matTemp(,) As Double
        '--  matE(,), matF(,) --> matriz 4ta y 5ta
        Dim a, b, c, d, m, n As Integer
        Dim angleRad1, angleRad2, angleRad3, angleRad4, angleRad5, angle1,
angle2, angle3, angle4, angle5 As Double
        'Dim temp As String
        Dim al1, a1, d1 As Double
        Dim al2, a2, d2 As Double
        Dim al3, a3, d3 As Double
        Dim al4, a4, d4 As Double
        Dim al5, a5, d5 As Double
        a = 3 'fila %C
        b = 3 'columna %C
        ReDim matA(a, b)
        ReDim matB(a, b)
        ReDim matC(a, b)
        ReDim matE(a, b)
        ReDim matF(a, b)
        ' Llenar matriz A
        al1 = Math.PI / 2
        a1 = 0
        d1 = 17.5
        angle1 = number1   ' o = motor1
        angleRad1 = angle1 * Math.PI / 180
        matA = llenarMatriz(al1, a1, d1, angleRad1)
        ' Llenar matriz B
        al2 = 0
        a2 = 28
        d2 = 0
        angle2 = number2
        angleRad2 = angle2 * Math.PI / 180
        matB = llenarMatriz(al2, a2, d2, angleRad2)
        ' Llenar matriz C
        al3 = 0
        a3 = 28
```

```
        d3 = 0
        angle3 = number3
        angleRad3 = angle3 * Math.PI / 180
        matC = llenarMatriz(al3, a3, d3, angleRad3)

        ' Llenar matriz E
        al4 = -Math.PI / 2
        a4 = 0
        d4 = 0
        angle4 = number4 - 90
        angleRad4 = angle4 * Math.PI / 180
        matE = llenarMatriz(al4, a4, d4, angleRad4)

        ' Llenar matriz F
        al5 = 0
        a5 = 0
        d5 = 28
        angle5 = 0
        angleRad5 = angle5 * Math.PI / 180
        matF = llenarMatriz(al5, a5, d5, angleRad5)

        'Llenar matriz D
        c = 3
        d = 0
        ReDim matD(c, d)
        matD(0, 0) = 0 'llena matriz D
        matD(1, 0) = 0
        matD(2, 0) = 0
        matD(3, 0) = 1

        m = a 'Dimesión de matriz matTemp
        n = b
        ReDim matTemp(m, n)

        matTemp = cacularMultiplicacion(a, b, matA, a, b, matB)
        matTemp = cacularMultiplicacion(a, b, matTemp, a, b, matC)
        matTemp = cacularMultiplicacion(a, b, matTemp, a, b, matE)
        matTemp = cacularMultiplicacion(a, b, matTemp, a, b, matF)

        m = a
        n = d

        ReDim matR(m, n)
        matR = cacularMultiplicacion(a, b, matTemp, c, d, matD)

        'imprimir en nuevas celdas
        txtX.Text = matR(0, 0) / 100
        txtY.Text = matR(1, 0) / 100
        txtZ.Text = matR(2, 0) / 100

        txtX_i.Text = matR(0, 0) / 100
        txtY_i.Text = matR(1, 0) / 100
        txtZ_i.Text = matR(2, 0) / 100

        My.Settings.SaveX = Convert.ToDouble(txtX.Text)
        My.Settings.SaveY = Convert.ToDouble(txtY.Text)
        My.Settings.SaveZ = Convert.ToDouble(txtZ.Text)


        Form1.txtIniX.Text = Convert.ToDouble(txtX.Text)
        Form1.txtIniY.Text = Convert.ToDouble(txtY.Text)
```

```
        Form1.txtIniZ.Text = Convert.ToDouble(txtZ.Text)

        Control1.Text = Convert.ToString(My.Settings.SaveAng1)
        Control2.Text = Convert.ToString(My.Settings.SaveAng2)
        Control3.Text = Convert.ToString(My.Settings.SaveAng3)
        Control4.Text = Convert.ToString(My.Settings.SaveAng4)
        'Else
        'MsgBox("Ingrese otro ángulo.")
        'End If
    End Sub
    '---------------------

    Private Function llenarMatriz(ByVal al As Double, ByVal a As Double, ByVal d
As Double, ByVal motor As Double)
        Dim matTem As Double(,)
        ReDim matTem(3, 3)
        matTem(0, 0) = Math.Cos(motor)
        matTem(0, 1) = -1 * Math.Cos(al) * Math.Sin(motor)
        matTem(0, 2) = Math.Sin(al) * Math.Sin(motor)
        matTem(0, 3) = a * Math.Cos(motor)
        matTem(1, 0) = Math.Sin(motor)
        matTem(1, 1) = Math.Cos(al) * Math.Cos(motor)
        matTem(1, 2) = -1 * Math.Sin(al) * Math.Cos(motor)
        matTem(1, 3) = a * Math.Sin(motor)
        matTem(2, 0) = 0
        matTem(2, 1) = Math.Sin(al)
        matTem(2, 2) = Math.Cos(al)
        matTem(2, 3) = d
        matTem(3, 0) = 0
        matTem(3, 1) = 0
        matTem(3, 2) = 0
        matTem(3, 3) = 1
        Return matTem
    End Function
    Private Function cacularMultiplicacion(ByVal a As Integer, ByVal b As
Integer, ByVal matA As Double(,), ByVal c As Integer, ByVal d As Integer, ByVal
matB As Double(,))
        Dim matr As Double(,)
        Dim temp, sum As Double
        sum = 0
        ReDim matr(a, d)
        a = a + 1
        d = d + 1
        For fil = 0 To a - 1 'Fila 1'
            For col = 0 To d - 1
                For k = 0 To a - 1 'Fila de matriz 2
                    sum = sum + matA(fil, k) * matB(k, col)
                Next
                temp = Math.Round(sum, 7)
                matr(fil, col) = temp
                sum = 0
            Next
        Next
        Return matr
    End Function

    Private Sub btnInversa_Click(sender As Object, e As EventArgs) Handles
btnInversa.Click
        Dim x, y, z As Double
        Dim motor1, motor2, motor3, motor2v As Double
        Dim temp1, temp2, temp3, temp4, temp5, temp6, temp7, temp8 As Double
```

```
        'Asignar valores
        x = Convert.ToDouble(txtX.Text)
        y = Convert.ToDouble(txtY.Text)
        z = Convert.ToDouble(txtZ.Text)

        z = z - 0.45

        'Motor 1

        If x = 0 Then
            motor1 = 0
        Else

            motor1 = Math.Atan2(y, x)

        End If

        temp8 = (x ^ 2 + y ^ 2 + z ^ 2 - 0.275 ^ 2 - 0.09 ^ 2) / (2 * 0.275 *
0.09)
        If temp8 > 0.998 And temp8 < 1.02 Then
            temp8 = 1
        End If

        motor3 = Math.Acos(temp8)

        temp1 = Math.Sqrt(x ^ 2 + y ^ 2)
        temp6 = temp1 * -1
        temp2 = Math.Atan2(z, temp1)  '-- este es
        temp7 = Math.Atan2(z, temp6)
        temp3 = 0.09 * Math.Sin(motor3)
        temp4 = 0.275 + 0.09 * Math.Cos(motor3)
        temp5 = Math.Atan2(temp3, temp4)  '-- este es

        motor2 = temp2 - temp5
        motor2v = temp7 - temp5

        'Convertir a grados
        motor1 = motor1 * 180 / Math.PI
        motor2 = motor2 * 180 / Math.PI
        motor3 = motor3 * 180 / Math.PI
        motor2v = motor2v * 180 / Math.PI

        'Redondear valores a cero decimales
        motor1 = Math.Round(motor1, 0)
        motor2 = Math.Round(motor2, 0)
        motor3 = Math.Round(motor3, 0)
        'motor2v = Math.Truncate(motor2v)

        'Mostrar resultado
        txtMotor1M.Text = motor1
        txtMotor2M.Text = motor2
        txtMotor3M.Text = motor3
        'txtAngulo2v.Text = motor2v
    End Sub

    Function analisisangulo1(ByVal number1 As Double)
        Dim pos1 As Double
        Dim cuadrante1, d As Double
        Dim mover, dis1, dis2 As Double
        Dim aux1, aux2 As Double

        aux1 = My.Settings.SaveAng1
```

```
        aux2 = My.Settings.SaveCuad1
        If (number1 >= 0) Then
            pos1 = number1
        Else
            pos1 = number1 * -1
        End If
        If ((number1 >= 0) And (number1 <= 90)) Then
            cuadrante1 = 1
            pos1 = number1
        Else
            If ((number1 <= 180) And (number1 > 90)) Then
                cuadrante1 = 2
                pos1 = number1
            Else
                If (((number1 <= 270) And (number1 > 180)) Or ((number1 >= -180)
And (number1 <= -90))) Then
                    cuadrante1 = 3
                    pos1 = 360 + number1
                Else
                    cuadrante1 = 4
                    pos1 = 360 + number1
                End If
            End If
        End If
        d = Math.Abs(cuadrante1 - My.Settings.SaveCuad1)
        If (d = 3) Then
            If (My.Settings.SaveCuad1 = 4) Then
                mover = Math.Abs(My.Settings.SaveAng1) + number1
                signo1 = "+"
            Else
                mover = My.Settings.SaveAng1 + Math.Abs(number1)
                signo1 = "-"
            End If
        Else
            If (d <= 1) Then
                If (pos1 < My.Settings.SaveAng1_pos) Then
                    mover = My.Settings.SaveAng1_pos - pos1
                    signo1 = "-"
                Else
                    mover = pos1 - My.Settings.SaveAng1_pos
                    signo1 = "+"
                End If
            Else
                If (My.Settings.SaveCuad1 < cuadrante1) Then
                    dis1 = Math.Abs(number1) + My.Settings.SaveAng1
                    dis2 = pos1 - My.Settings.SaveAng1
                    If (dis1 < dis2) Then
                        mover = dis1
                        signo1 = "-"
                    Else
                        mover = dis2
                        signo1 = "+"
                    End If
                Else
                    dis1 = Math.Abs(My.Settings.SaveAng1) + number1
                    dis2 = My.Settings.SaveAng1_pos - number1
                    If (dis1 <= dis2) Then
                        mover = dis1
                        signo1 = "+"
                    Else
                        mover = dis2
                        signo1 = "-"
```

```
                    End If
                End If

            End If

        End If
        My.Settings.SaveAng1 = number1
        My.Settings.SaveAng1_pos = pos1
        My.Settings.SaveCuad1 = cuadrante1

        mover = Convert.ToString(mover)
        Return mover

    End Function

    Function analisisangulo2(ByVal number2 As Double)
        Dim mover, aux1, aux2 As Double

        aux1 = My.Settings.SaveAng2
        aux2 = number2 - aux1
        If (aux2 >= 0) Then
            signo2 = "+"
            mover = aux2
        Else
            signo2 = "-"
            mover = aux2 * -1
        End If
        My.Settings.SaveAng2 = number2


        mover = Convert.ToString(mover)
        Return mover

    End Function

    Function analisisangulo3(ByVal number3 As Double)
       Dim mover, aux1, aux2 As Double

        aux1 = My.Settings.SaveAng3
        aux2 = number3 - aux1
        If (aux2 >= 0) Then
            signo3 = "+"
            mover = aux2
        Else
            signo3 = "-"
            mover = aux2 * -1
        End If
        My.Settings.SaveAng3 = number3


        mover = Convert.ToString(mover)
        Return mover
    End Function

    Function analisisangulo4(ByVal number4 As Double)
        Dim mover, aux1, aux2 As Double

        aux1 = My.Settings.SaveAng4
        aux2 = number4 - aux1
        If (aux2 >= 0) Then
            signo4 = "+"
            mover = aux2
```

```
        Else
            signo4 = "-"
            mover = aux2 * -1
        End If
        My.Settings.SaveAng4 = number4


        mover = Convert.ToString(mover)
        Return mover

    End Function
    Private Function conversoratrama(ByVal angulostr As Double)
        Dim numberaux As String
        numberaux = Convert.ToString(angulostr)
        If (angulostr < 10) Then
            numberaux = "00" & numberaux
        Else
            If (angulostr < 100) Then
                numberaux = "0" & numberaux
            Else
                numberaux = numberaux
            End If
        End If
        Return numberaux
    End Function
End Class
```

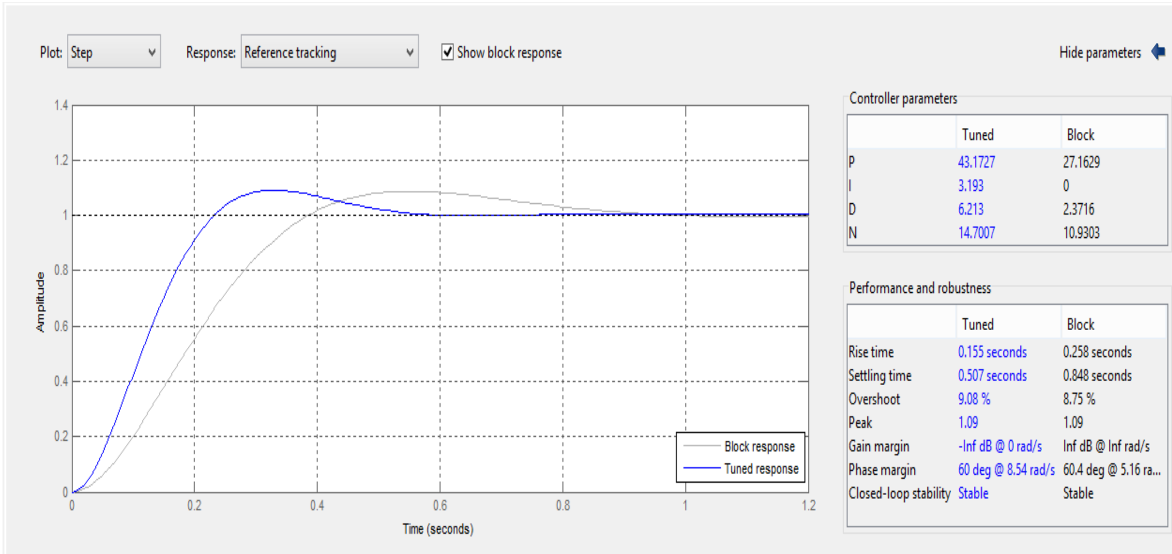
Código Form3

```
Public Class Form3
    Dim StrBufferOut As String
    Dim StrBufferIn As String
    Private Sub btncalibrar_Click(sender As Object, e As EventArgs) Handles
btncalibrar.Click
        Dim a1, a2, a3, a4 As Double

        '-------------------- Guardo X Y Z
        My.Settings.SaveX = txtX.Text
        My.Settings.SaveY = txtY.Text
        My.Settings.SaveZ = txtZ.Text

        '--------------------- Guardo ángulos
        My.Settings.SaveAng1 = txtMotor1.Text
        My.Settings.SaveAng2 = txtMotor2.Text
        My.Settings.SaveAng3 = txtMotor3.Text
        My.Settings.SaveAng4 = txtMotor4.Text
        My.Settings.SaveAng1_pos = Math.Abs(Convert.ToDouble(txtMotor1.Text))

        a1 = txtMotor1.Text
        a2 = txtMotor2.Text
        a3 = txtMotor3.Text
        a4 = txtMotor4.Text
        '------------------------------ Actualizo valores en pantalla inicial
        Form1.txtIniX.Text = txtX.Text
        Form1.txtIniY.Text = txtY.Text
        Form1.txtIniZ.Text = txtZ.Text

        '------------------------------ Actualizo valores en pantalla de directa

        Form2.txtX_i.Text = txtX.Text
        Form2.txtY_i.Text = txtY.Text
        Form2.txtZ_i.Text = txtZ.Text


    End Sub



    Private Sub detpos_Click(sender As Object, e As EventArgs) Handles
detpos.Click
        Dim matA(,), matB(,), matC(,), matD(,), matR(,), matF(,), matE(,),
matTemp(,) As Double
        '--  matE(,), matF(,) --> matriz 4ta y 5ta
        Dim a, b, c, d, m, n As Integer
        Dim angleRad1, angleRad2, angleRad3, angleRad4, angleRad5, angle1,
angle2, angle3, angle4, angle5 As Double
        'Dim temp As String
        Dim al1, a1, d1 As Double
        Dim al2, a2, d2 As Double
        Dim al3, a3, d3 As Double
        Dim al4, a4, d4 As Double
        Dim al5, a5, d5 As Double
        a = 3 'fila
        b = 3 'columna
        ReDim matA(a, b)
        ReDim matB(a, b)
        ReDim matC(a, b)
        ReDim matE(a, b)
```

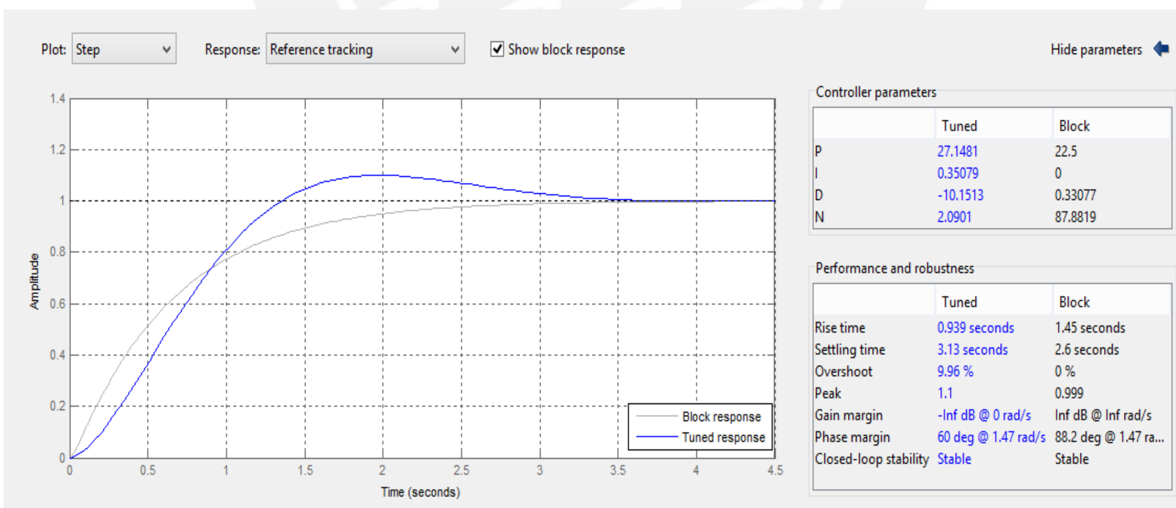
```
        ReDim matF(a, b)
        ' Llenar matriz A
        al1 = Math.PI / 2
        a1 = 0
        d1 = 0.175
        angle1 = txtMotor1.Text  ' o = motor1
        angleRad1 = angle1 * Math.PI / 180
        matA = llenarMatriz(al1, a1, d1, angleRad1)
        ' Llenar matriz B
        al2 = 0
        a2 = 0.28
        d2 = 0
        angle2 = txtMotor2.Text
        angleRad2 = angle2 * Math.PI / 180
        matB = llenarMatriz(al2, a2, d2, angleRad2)
        ' Llenar matriz C
        al3 = 0
        a3 = 0.28
        d3 = 0
        angle3 = txtMotor3.Text
        angleRad3 = angle3 * Math.PI / 180
        matC = llenarMatriz(al3, a3, d3, angleRad3)

        ' Llenar matriz E
        al4 = -Math.PI / 2
        a4 = 0
        d4 = 0
        angle4 = txtMotor4.Text - 90
        angleRad4 = angle4 * Math.PI / 180
        matE = llenarMatriz(al4, a4, d4, angleRad4)

        ' Llenar matriz F
        al5 = 0
        a5 = 0
        d5 = 0.28
        angle5 = 0
        angleRad5 = angle5 * Math.PI / 180
        matF = llenarMatriz(al5, a5, d5, angleRad5)

        'Llenar matriz D
        c = 3
        d = 0
        ReDim matD(c, d)
        matD(0, 0) = 0 'llena matriz D
        matD(1, 0) = 0
        matD(2, 0) = 0
        matD(3, 0) = 1

        m = a 'Dimesión de matriz matTemp
        n = b
        ReDim matTemp(m, n)

        matTemp = cacularMultiplicacion(a, b, matA, a, b, matB)
        matTemp = cacularMultiplicacion(a, b, matTemp, a, b, matC)
        matTemp = cacularMultiplicacion(a, b, matTemp, a, b, matE)
        matTemp = cacularMultiplicacion(a, b, matTemp, a, b, matF)

        m = a
        n = d

        ReDim matR(m, n)
        matR = cacularMultiplicacion(a, b, matTemp, c, d, matD)
```

```
        'imprimir en nuevas celdas
        txtX.Text = matR(0, 0)
        txtY.Text = matR(1, 0)
        txtZ.Text = matR(2, 0)

        txtX.Text = matR(0, 0)
        txtY.Text = matR(1, 0)
        txtZ.Text = matR(2, 0)

    End Sub
    '----------------------


    Private Function llenarMatriz(ByVal al As Double, ByVal a As Double, ByVal d
As Double, ByVal motor As Double)
        Dim matTem As Double(,)
        ReDim matTem(3, 3)
        matTem(0, 0) = Math.Cos(motor)
        matTem(0, 1) = -1 * Math.Cos(al) * Math.Sin(motor)
        matTem(0, 2) = Math.Sin(al) * Math.Sin(motor)
        matTem(0, 3) = a * Math.Cos(motor)
        matTem(1, 0) = Math.Sin(motor)
        matTem(1, 1) = Math.Cos(al) * Math.Cos(motor)
        matTem(1, 2) = -1 * Math.Sin(al) * Math.Cos(motor)
        matTem(1, 3) = a * Math.Sin(motor)
        matTem(2, 0) = 0
        matTem(2, 1) = Math.Sin(al)
        matTem(2, 2) = Math.Cos(al)
        matTem(2, 3) = d
        matTem(3, 0) = 0
        matTem(3, 1) = 0
        matTem(3, 2) = 0
        matTem(3, 3) = 1
        Return matTem
    End Function
    Private Function cacularMultiplicacion(ByVal a As Integer, ByVal b As
Integer, ByVal matA As Double(,), ByVal c As Integer, ByVal d As Integer, ByVal
matB As Double(,))
        Dim matr As Double(,)
        Dim temp, sum As Double
        sum = 0
        ReDim matr(a, d)
        a = a + 1
        d = d + 1
        For fil = 0 To a - 1 'Fila 1'
            For col = 0 To d - 1
                For k = 0 To a - 1 'Fila de matriz 2
                    sum = sum + matA(fil, k) * matB(k, col)
                Next
                temp = Math.Round(sum, 7)
                matr(fil, col) = temp
                sum = 0
            Next
        Next
        Return matr
    End Function


End Class
```

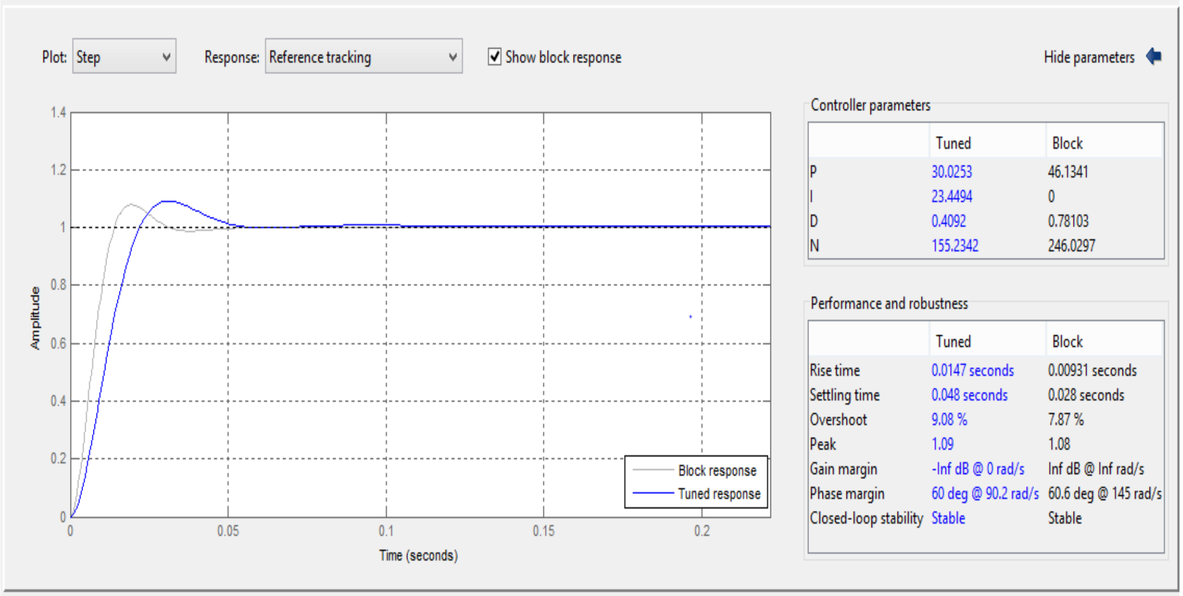
ANEXO 18: Gráficas de la sintonización en Simulink de los motores M1, M2, M3 y M4



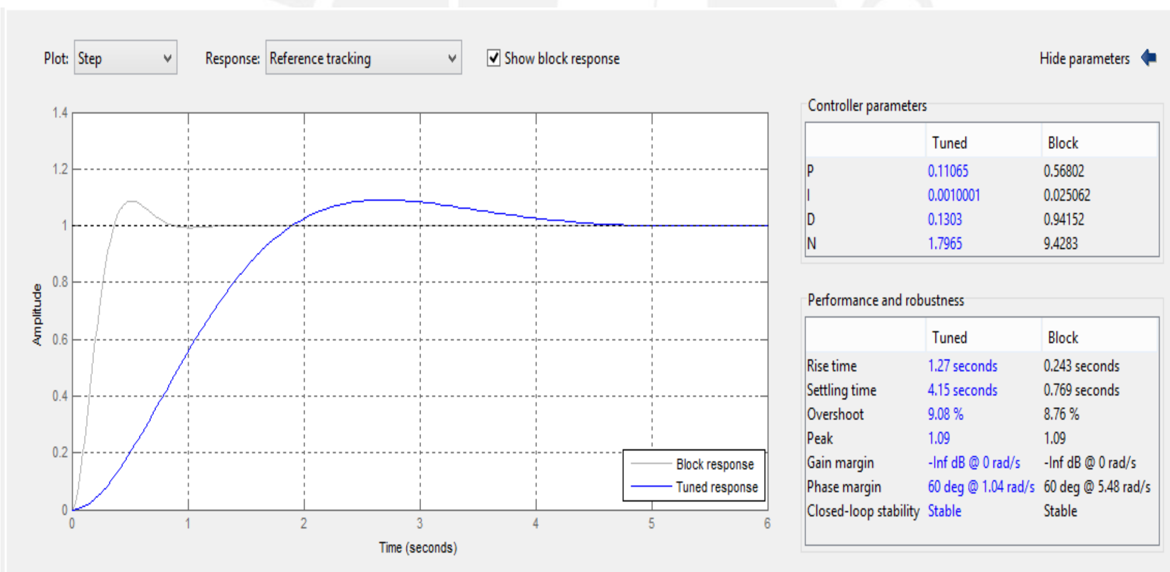
Sintonización del motor M1



Sintonización del motor M2



Sintonización del motor M3



Sintonización del motor M4