



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Design & Sizing of a Testing Bench for a Turbocompressor

Document:

Appendix

Author:

David Hernandez Gomez

Director/Co-director:

Jordi Ventosa Molina
Àngel Comas Amengual

Degree:

Bachelor in Aerospace Vehicle Engineering

Examination session:

Spring

BACHELOR FINAL THESIS

Contents

A	Flowmeter	1
A.1	Matlab codes	1
A.2	Arduino data processing codes	10
B	Pressure	12
B.1	Matlab codes	12
B.2	Arduino data processing codes	18
C	Tachometer	20
C.1	Matlab codes	20
C.2	Arduino data processing codes	22
D	Thermometer	24
D.1	Matlab codes	24
D.2	Arduino data processing codes	26
D.3	PT100 table	27
E	Efficiency error	28
E.1	Matlab codes	28
F	System unification	31
F.1	Matlab code for obtaining a compressor map point	31
F.2	Arduino data processing codes	34
G	Orifice plate construction and assembly	39

Listings

A.1	Function call for the calculation of differential pressure values for given values of \dot{q}_m as a function of β	1
A.2	Computation of differential pressure values for given values of q_m as a function of β	1
A.3	Computation of differential pressure values for values of β defined as a function of \dot{q}_m	4
A.4	Differential pressure measurement error at the orifice plate.	8
A.5	Differential pressure data processing at the orifice plate.	10
B.1	Calculation of the pressure at the compressor's inlet.	12
B.2	Pressure measurement and pressure ratio error.	14
B.3	Compressor's output pressure with transducer data processing.	18
C.1	Revolutions measurement error.	20
C.2	Data processing of the revolutions measurement.	22
D.1	Revolutions measurement error.	24
D.2	Data processing of the temperature at the compressor outlet.	26
E.1	Efficiency calculation error.	28
F.1	Calculations to obtain a compressor map's point for a given measures done.	31
F.2	\dot{q}_m and $\delta\dot{q}_m$ calculations function for a given β and ΔP values.	32
F.3	Π and $\delta\Pi$ calculations function for a given P_1 and P_2 values.	32
F.4	η and $\delta\eta$ calculations function for a given Π and T_2 values.	33
F.5	rpm and δrpm calculations function for a given rpm value.	34
F.6	All data processing required united in a single code.	34

List of Figures

G.1	3D printed orifice plates	39
G.2	Orifice plate at the pipe with pressures tappings	40
G.3	Orifice plate testing assembly	41
G.4	ΔP as a function of the blower power [%]	42
G.5	\dot{q}_m as a function of the blower power [%]	42
G.6	ΔP as a function of \dot{q}_m	43



List of Tables

D.1 Resistance value of a PT100 as a function of temperature [1].	27
G.1 Results obtained from testing	42

Appendix A

Flowmeter

A.1 Matlab codes

```

1 clear all
2 clc
3 close all
4
5 %% DATA INPUT %%
6
7 % Fisical data
8 air_density = 1.225; % [kg/m^3]
9 D = 59.25 * 10^-3; % [m]
10 visosity = 1.802 * 10^-5; % [kg/ms]
11
12 % diferent constant flow values
13 % the mass airflow is between 0.03779 & 0.22679 kg/s
14 m = 5;
15 q_m = linspace(0.03779, 0.22679, m); % [kg/s]
16
17 % different values of beta
18 n = 100;
19 beta = linspace(0.15, 0.75, n);
20
21 % Pressure accuracy
22 min_press_increm = 250; % [Pa]
23
24
25 %% FUNCTION CALL
26
27 [increm_P_corner, reynolds_D] = flowmeter_corner_tapp(air_density, D, visosity, q_m, beta, m, n);

```

Listing A.1: Function call for the calculation of differential pressure values for given values of \dot{q}_m as a function of β .

```

1 function [increm_P_corner, reynolds_D] = flowmeter_corner_tapp(air_density, D, visosity, q_m, beta, m, n)
2
3 % Tapping data
4 % As it has corner tapping
5 L_1 = 0;

```

```

6 L_2 = 0;
7
8
9 %% Vectros definition
10 reynolds_D = zeros(1, m);
11 aux_A = zeros(m, n);
12 aux_c_1 = zeros(m, n);
13 aux_c_2 = zeros(m, n);
14 aux_c_3 = zeros(m, n);
15 aux_c_4 = zeros(m, n);
16 aux_c_5 = zeros(m, n);
17 M_2 = zeros(m, n);
18 aux_m = zeros(m, n);
19 coef_c = zeros(m, n);
20
21 increm_P_corner = zeros(m, n);
22 coef_e = zeros(m, n);
23 coef_e_aux = zeros(m, n);
24 error_e = zeros(m, n);
25 P_2 = zeros(m, n);
26
27
28 %% CALCULATIONS
29
30 for j = 1:m
31
32     reynolds_D(1, j) = (4 * q_m(1, j)) / (pi * visosity * D);
33
34     for i = 1:n
35         aux_A(j, i) = ((19000 * beta(1, i)) / reynolds_D(1, j))^0.8;
36
37         aux_c_1(j, i) = 0.5961 + 0.0261 * beta(1, i)^2 - 0.216 * beta(1, i)^8;
38         aux_c_2(j, i) = 0.000521 * ((10^6 * beta(1, i)) / reynolds_D(1, j))^0.7;
39         aux_c_3(j, i) = (0.0188 + 0.0063 * aux_A(j, i)) * beta(1, i)^3.5 * (10^6 / reynolds_D(1, j))^0.3;
40         aux_c_4(j, i) = (0.043 + 0.08 * exp(-10 * L_1) - 0.123 * exp(-7 * L_1)) * (1 - 0.11 * aux_A(j, i)) * (beta
(1, i)^4 / (1 - beta(1, i)^4));
41         aux_c_5(j, i) = 0.011 * (0.75 - beta(1, i)) * (2.8 - (D * 10^3) / 25.4);
42         M_2(j, i) = (2 * L_2) / (1 - beta(1, i));
43         aux_m(j, i) = -0.031 * (M_2(j, i) - 0.8 * M_2(j, i)^1.1) * beta(1, i)^1.3;
44
45         coef_c(j, i) = aux_c_1(j, i) + aux_c_2(j, i) + aux_c_3(j, i) + aux_c_4(j, i) + aux_c_5(j, i) +
aux_m(j, i);
46
47         coef_e_aux(j, i) = 1;
48         error_e(j, i) = 100;
49         while(error_e(j, i) > 10^-4)
50             increm_P_corner(j, i) = ((q_m(1, j) * sqrt(1 - beta(1, i)^4) * 4) / (coef_c(j, i) * coef_e_aux(
j, i) * pi * beta(1, i)^2 * D^2))^2 * 1 / (2 * air_density);
51             if increm_P_corner(j, i) >= 101325
52                 error_e(j, i) = 10^-5;
53             else
54                 P_2(j, i) = 101325 - increm_P_corner(j, i);
55
56                 coef_e(j, i) = 1 - (0.351 + 0.256 * beta(1, i)^4 + 0.93 * beta(1, i)^8) * (1 - (P_2(j, i) / 101325)
^(1/1.4));
57                 error_e(j, i) = abs(coef_e(j, i) - coef_e_aux(j, i));
58                 coef_e_aux(j, i) = coef_e(j, i);
59             end

```



```

60     end
61     end
62
63 end
64
65
66
67 %% PLOTS
68 set(groot,'defaultAxesTickLabelInterpreter','latex');
69 set(groot,'defaultTextInterpreter','latex');
70 set(groot,'defaultLegendInterpreter','latex');
71
72 %%%
73 fig1 = figure(1);
74 hold on
75 set(gca, 'FontSize', 13)
76 plot(beta, increm_P_corner, LineWidth=2)
77 set(gca, 'YScale', 'log')
78 xlabel('\beta', 'FontSize', 15)
79 ylabel('\Delta P [Pa]', 'FontSize', 15)
80 legend('\dot{q}_m$ = $16.7\%', '\dot{q}_m$ = $37.5\%', '\dot{q}_m$ = $58.3\%', '\dot{q}_m$ = $79
    .1\%', '\dot{q}_m$ = $100\%', 'Location','best');
81 ylim([0 101325])
82 grid on
83 grid minor
84 box on;
85 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
86 hold off;
87
88 set(fig1, 'units', 'points');
89 pos = get(fig1, 'position');
90 set(fig1, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
91     'PaperSize',[pos(3), pos(4)]);
92 print(fig1, 'plots/DeltaP for as a function of beta for corner tapp, epsilon', '-dpdf', '-r0', '-bestfit'
93     );
94 %%%
95
96 fig4 = figure(4);
97 hold on
98 set(gca, 'FontSize', 13)
99 plot(beta, coef_c, LineWidth=2)
100 xlabel('\beta', 'FontSize', 15)
101 ylabel('Discharge coefficient, $C_d$', 'FontSize', 15)
102 xlim([0.15 0.75])
103 legend('\dot{q}_m$ = $16.7\%', '\dot{q}_m$ = $37.5\%', '\dot{q}_m$ = $58.3\%', '\dot{q}_m$ = $79
    .1\%', '\dot{q}_m$ = $100\%', 'Location','best');
104 grid on
105 grid minor
106 box on;
107 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
108 hold off;
109
110 set(fig4, 'units', 'points');
111 pos = get(fig4, 'position');
112 set(fig4, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
113     'PaperSize',[pos(3), pos(4)]);
114 print(fig4, 'plots/coef_c as a function of beta', '-dpdf', '-r0', '-bestfit');

```

```

115 %%%
116 coef_e_1 = coef_e(1, :);
117 coef_e_2 = coef_e(2, :);
118 coef_e_3 = coef_e(3, :);
119 coef_e_4 = coef_e(4, :);
120 coef_e_5 = coef_e(5, :);
121 fig5 = figure(5);
122 hold on
123 set(gca, 'FontSize', 13)
124 plot(beta(19:end), coef_e_1(19:end), LineWidth=2)
125 plot(beta(41:end), coef_e_2(41:end), LineWidth=2)
126 plot(beta(56:end), coef_e_3(56:end), LineWidth=2)
127 plot(beta(69:end), coef_e_4(69:end), LineWidth=2)
128 plot(beta(80:end), coef_e_5(80:end), LineWidth=2)
129 xlabel('\beta', 'FontSize', 15)
130 ylabel('Expansibility facotr, \varepsilon', 'FontSize', 15)
131 xlim([0.15 0.75])
132 legend('\dot{q}_m$ = $16.7\%$', '\dot{q}_m$ = $37.5\%$', '\dot{q}_m$ = $58.3\%$', '\dot{q}_m$ = $79
    .1\%$', '\dot{q}_m$ = $100\%$', 'Location','best');
133 grid on
134 grid minor
135 box on;
136 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
137 hold off;
138
139 set(fig5, 'units', 'points');
140 pos = get(fig5, 'position');
141 set(fig5, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
142     'PaperSize',[pos(3), pos(4)]);
143 print(fig5, 'plots/coef_e as a function of beta', '-dpdf', '-r0', '-bestfit');

```

Listing A.2: Computation of differential pressure values for given values of q_m as a function of β .

```

1 clear all
2 clc
3 close all
4
5 %% DATA INPUT
6
7 air_density = 1.225; % [kg/m^3]
8 D = 59.25 * 10^-3; % [m]
9 visosity = 1.802 * 10^-5; % [kg/ms]
10
11 % Tappings
12 L_1 = 0;
13 L_2 = 0;
14
15 % diferent constant flow values
16 % the mass airflow is between 0.03779 & 0.22679 kg/s
17 n = 100;
18 aux_1234 = (0.03779 / 0.22679) * 100;
19 aux_12354 = (0.22679 / 0.22679) * 100;
20 q_m_por = linspace(aux_1234, aux_12354, n); % [kg/s]
21 q_m = linspace(0.03779, 0.22679, n);
22
23 % Beta defined, only one value
24 m = 2;
25 beta = [0.48 0.68];

```

```

26
27 % Pressure error
28 min_press_increm = 250 + 3.815; % [Pa]
29
30 %% VECTOR DEFINITION
31
32 reynolds_D = zeros(1, n);
33 aux_A = zeros(m, n);
34 aux_c_1 = zeros(m, n);
35 aux_c_2 = zeros(m, n);
36 aux_c_3 = zeros(m, n);
37 aux_c_4 = zeros(m, n);
38 aux_c_5 = zeros(m, n);
39 M_2 = zeros(m, n);
40 aux_m = zeros(m, n);
41 coef_c = zeros(m, n);
42 increm_P_corner = zeros(m, n);
43 coef_e_aux = zeros(m, n);
44 error_e = zeros(m, n);
45 P_2 = zeros(m, n);
46 coef_e = zeros(m, n);
47 uncertainty_c = zeros(m, n);
48 uncertainty_e = zeros(m, n);
49 uncertainty_d = zeros(m, n);
50 uncertainty_p = zeros(m, n);
51 uncertainty_aux = zeros(m, n);
52 uncertainty = zeros(m, n);
53 delta_P = zeros(1, n);
54
55 %% CALCULATIONS
56
57 for j = 1:n
58     reynolds_D(1, j) = (4 * q_m(1, j)) / (pi * visosity * D);
59     for i = 1:m
60         aux_A(i, j) = ((19000 * beta(1, i)) / reynolds_D(1, j))^0.8;
61         aux_c_1(i, j) = 0.5961 + 0.0261 * beta(1, i)^2 - 0.216 * beta(1, i)^8;
62         aux_c_2(i, j) = 0.000521 * ((10^6*beta(1, i)) / reynolds_D(1, j))^0.7;
63         aux_c_3(i, j) = (0.0188 + 0.0063 * aux_A(i, j)) * beta(1, i)^3.5 * (10^6 / reynolds_D(1, j))^0.3;
64         aux_c_4(i, j) = (0.043 + 0.08*exp(-10*L_1) - 0.123*exp(-7*L_1)) * (1 - 0.11*aux_A(i, j)) * (beta
(1, i)^4 / (1-beta(1, i)^4));
65         aux_c_5(i, j) = 0.011 * (0.75 - beta(1, i)) * (2.8 - (D*10^3)/25.4);
66         M_2(i, j) = (2*L_2) / (1-beta(1, i));
67         aux_m(i, j) = -0.031*(M_2(i, j) - 0.8*M_2(i, j)^1.1) * beta(1, i)^1.3;
68         coef_c(i, j) = aux_c_1(i, j) + aux_c_2(i, j) + aux_c_3(i, j) + aux_c_4(i, j) + aux_c_5(i, j) +
aux_m(i, j);
69         coef_e_aux(i, j) = 1;
70         error_e(i, j) = 100;
71         while(error_e(i, j) > 10^-4)
72             increm_P_corner(i, j) = ((q_m(1, j) * sqrt(1-beta(1, i)^4) * 4) / (coef_c(i, j) * coef_e_aux(
i, j) * pi * beta(1, i)^2 * D^2))^2 * 1/(2*air_density);
73             if increm_P_corner(i, j) >= 101325
74                 error_e(i, j) = 10^-5;
75             else
76                 P_2(i, j) = 101325 - increm_P_corner(i, j);
77                 coef_e(i, j) = 1 - (0.351+0.256*beta(1, i)^4+0.93*beta(1, i)^8) * (1 - (P_2(i, j)/101325)
^(1/1.4));
78                 error_e(i, j) = abs(coef_e(i, j) - coef_e_aux(i, j));
79                 coef_e_aux(i, j) = coef_e(i, j);

```

```

80     end
81     end
82
83     if increm_P_corner(i, j) < 101325
84         % uncertainty coef_c
85         if beta(1, i) <= 0.6
86             uncertainty_c(i, j) = 0.5 + 0.9*(0.75-beta(1, i))*(2.8-(D*10^3)/25.4);
87         else
88             uncertainty_c(i, j) = (1.667*beta(1, i)-0.5) + 0.9*(0.75-beta(1, i))*(2.8-(D*10^3)/25.4);
89         end
90         % uncertainty epsilon
91         uncertainty_e(i, j) = 3.5 * (increm_P_corner(i, j)/(1.4*101325));
92         % uncertainty diameter
93         uncertainty_d(i, j) = ((2*beta(1, i)^4) / (1-beta(1, i)^4))^2 * (0.05/59.25)^2;
94         % uncertainty pressure
95         uncertainty_p(i, j) = 1/4*(min_press_increm/increm_P_corner(i, j))^2;
96
97         uncertainty_aux(i, j) = sqrt(uncertainty_p(i, j) + uncertainty_d(i, j) + (uncertainty_c(i, j)
98         /100)^2 + (uncertainty_e(i, j)/100)^2);
99
100         uncertainty(i, j) = uncertainty_aux(i, j) * 100;
101
102     else
103     end
104 end
105 for t = 1:n
106     if t <=37
107         delta_P(1, t) = increm_P_corner(1, t);
108     else
109         delta_P(1, t) = increm_P_corner(m, t);
110     end
111 end
112
113 %% PLOTS
114 set(groot,'defaultAxesTickLabelInterpreter','latex');
115 set(groot,'defaultTextInterpreter','latex');
116 set(groot,'defaultLegendInterpreter','latex');
117
118 fig5 = figure(5);
119 hold on
120 set(gca, 'FontSize', 13)
121 plot(q_m_por, increm_P_corner, LineWidth=2)
122 xlabel('$\dot{q}_m$ [kg/s]', 'FontSize', 15)
123 ylabel('$\Delta P$ [Pa]', 'FontSize', 15)
124 legend('$\beta = 0,48$', '$\beta = 0,68$', 'Location','best');
125 grid on
126 grid minor
127 ylim([0 101325])
128 box on
129 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
130 hold off;
131
132 set(fig5, 'units', 'points');
133 pos = get(fig5, 'position');
134 set(fig5, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
135     'PaperSize',[pos(3), pos(4)]);
136 print(fig5, 'plots/DeltaP as a function of q_m 048 068', '-dpdf', '-r0', '-bestfit');

```

```

137 %%%%
138 uncertainty_1 = uncertainty(1, :);
139 fig7 = figure(7);
140 hold on
141 set(gca, 'FontSize', 13)
142 plot(q_m_por(1:50), uncertainty_1(1:50), LineWidth=2)
143 plot(q_m_por, uncertainty(2, :), LineWidth=2)
144 xlabel('$\dot{q}_m$ [%]', 'FontSize', 15)
145 ylabel('$\dot{q}_m$ uncertainty [%]', 'FontSize', 15)
146 legend('$\beta$ = 0,48', '$\beta$ = 0,68', 'Location','best');
147 grid on
148 grid minor
149 box on
150 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
151 hold off;
152
153 set(fig7, 'units', 'points');
154 pos = get(fig7, 'position');
155 set(fig7, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
156     'PaperSize',[pos(3), pos(4)]);
157 print(fig7, 'plots/uncertainty_total', '-dpdf', '-r0', '-bestfit');
158 %%%%
159 fig8 = figure(8);
160 hold on
161 set(gca, 'FontSize', 13)
162 plot(q_m_por, delta_P, LineWidth=2)
163 xlabel('$\dot{q}_m$ [%]', 'FontSize', 15)
164 ylabel('$\Delta P$ [Pa]', 'FontSize', 15)
165 grid on
166 grid minor
167 box on
168 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
169 hold off;
170
171 set(fig8, 'units', 'points');
172 pos = get(fig8, 'position');
173 set(fig8, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
174     'PaperSize',[pos(3), pos(4)]);
175 print(fig8, 'plots/DeltaP as a function of q with both plates', '-dpdf', '-r0', '-bestfit');
176 %%%%
177 uncertainty_e_1 = uncertainty_e(1, :);
178 fig10 = figure(10);
179 hold on
180 set(gca, 'FontSize', 13)
181 plot(q_m_por(1:50), uncertainty_e_1(1:50), LineWidth=2)
182 plot(q_m_por, uncertainty_e(2, :), LineWidth=2)
183 xlabel('$\dot{q}_m$ [%]', 'FontSize', 15)
184 ylabel('$\varepsilon$ uncertainty [%]', 'FontSize', 15)
185 legend('$\beta$ = 0,48', '$\beta$ = 0,68', 'Location','best');
186 grid on
187 grid minor
188 box on
189 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
190 hold off;
191
192 set(fig10, 'units', 'points');
193 pos = get(fig10, 'position');
194 set(fig10, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...

```

```

195     'PaperSize',[pos(3), pos(4)];
196 print(fig10, 'plots/uncertainty_e', '-dpdf', '-r0', '-bestfit');
197
198
199 uncertainty_p_1 = uncertainty_p(1, :);
200 fig11 = figure(11);
201 hold on
202 set(gca, 'FontSize', 13)
203 plot(q_m_por(1:50), 100*sqrt(uncertainty_p_1(1:50)), LineWidth=2)
204 plot(q_m_por, 100*sqrt(uncertainty_p(2, :)), LineWidth=2)
205 xlabel('$\dot{q}_m$ [%]', 'FontSize', 15)
206 ylabel('$\Delta P$ uncertainty [%]', 'FontSize', 15)
207 legend('$\beta = 0,48', '$\beta = 0,68', 'Location','best');
208 grid on
209 grid minor
210 box on
211 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
212 hold off;
213
214 set(fig11, 'units', 'points');
215 pos = get(fig11, 'position');
216 set(fig11, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
217     'PaperSize',[pos(3), pos(4)]);
218 print(fig11, 'plots/uncertainty_p', '-dpdf', '-r0', '-bestfit');
219
220 rel_error_tot = relative_error_press_flowmeter(q_m_por, n, increm_P_corner, m);
221 [P_1, P_2] = pressure_calc(increm_P_corner, beta, q_m, n, m, air_density, D, coef_c);

```

Listing A.3: Computation of differential pressure values for values of β defined as a function of \dot{q}_m .

```

1 function rel_error_tot = relative_error_press_flowmeter(q_m_por, n, increm_P_corner, m)
2
3 %% DATA INPUT
4 accuracy = 0.5;
5 max_press = 500;
6 abs_error_ard = (1000 / 13107) * 0.5 * 100;
7
8 %% VECTOR DEFINITION
9 rel_error_sensor = zeros(m, n);
10 rel_error_ard = zeros(m, n);
11 rel_error_tot = zeros(m, n);
12
13 %% CALCULATIONS
14
15 abs_error_sensor = (accuracy / 100) * max_press * 100;
16 for j =1:m
17     for i = 1:n
18         rel_error_sensor(j, i) = (abs_error_sensor/increm_P_corner(j, i)) * 100;
19         rel_error_ard(j, i) = (abs_error_ard/increm_P_corner(j, i)) * 100;
20         rel_error_tot(j, i) = rel_error_ard(j, i) + rel_error_sensor(j, i);
21     end
22 end
23
24 %% PLOTS
25 set(groot, 'defaultAxesTickLabelInterpreter', 'latex');
26 set(groot, 'defaultTextInterpreter', 'latex');
27 set(groot, 'defaultLegendInterpreter', 'latex');
28

```

```

29 rel_error_sensor_1 = rel_error_sensor(1,:);
30 fig31 = figure(31);
31 hold on
32 set(gca, 'FontSize', 13)
33 plot(q_m_por(1:50), rel_error_sensor_1(1:50), LineWidth=2)
34 plot(q_m_por, rel_error_sensor(2, :), LineWidth=2)
35 xlabel('\dot{q}_m$ $[\%]$', 'FontSize', 15)
36 ylabel('\Delta \Delta P$ $[\%]$', 'FontSize', 15)
37 legend('\beta$ = 0,48', '\beta$ = 0,68', 'Location','best');
38 grid on
39 grid minor
40 box on
41 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
42 hold off;
43
44 set(fig31, 'units', 'points');
45 pos = get(fig31, 'position');
46 set(fig31, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
47     'PaperSize',[pos(3), pos(4)]);
48 print(fig31, 'plots/rel_error_DeltaP_sens', '-dpdf', '-r0', '-bestfit');
49 %%%
50 rel_error_ard_1 = rel_error_ard(1,:);
51 fig32 = figure(32);
52 hold on
53 set(gca, 'FontSize', 13)
54 plot(q_m_por(1:50), rel_error_ard_1(1:50), LineWidth=2)
55 plot(q_m_por, rel_error_ard(2, :), LineWidth=2)
56 xlabel('\dot{q}_m$ $[\%]$', 'FontSize', 15)
57 ylabel('\Delta \Delta P$ $[\%]$', 'FontSize', 15)
58 legend('\beta$ = 0,48', '\beta$ = 0,68', 'Location','best');
59 grid on
60 grid minor
61 box on
62 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
63 hold off;
64
65 set(fig32, 'units', 'points');
66 pos = get(fig32, 'position');
67 set(fig32, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
68     'PaperSize',[pos(3), pos(4)]);
69 print(fig32, 'plots/rel_error_DeltaP_ard', '-dpdf', '-r0', '-bestfit');
70 %%%
71 rel_error_tot_1 = rel_error_tot(1,:);
72 fig33 = figure(33);
73 hold on
74 set(gca, 'FontSize', 13)
75 plot(q_m_por(1:50), rel_error_tot_1(1:50), LineWidth=2)
76 plot(q_m_por, rel_error_tot(2, :), LineWidth=2)
77 xlabel('\dot{q}_m$ $[\%]$', 'FontSize', 15)
78 ylabel('\Delta \Delta P$ $[\%]$', 'FontSize', 15)
79 legend('\beta$ = 0,48', '\beta$ = 0,68', 'Location','best');
80 grid on
81 grid minor
82 box on
83 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
84 hold off;
85
86 set(fig33, 'units', 'points');

```

```

87 pos = get(fig33, 'position');
88 set(fig33, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
89     'PaperSize', [pos(3), pos(4)]);
90 print(fig33, 'plots/rel_error_DeltaP_tot', '-dpdf', '-r0', '-bestfit');

```

Listing A.4: Differential pressure measurement error at the orifice plate.

A.2 Arduino data processing codes

```

1 #include <AMS.h>           //include the AMS library
2 #include <LiquidCrystal_I2C.h> // allows to control I2C displays
3 #include <Wire.h>         //allows communication over i2c devices
4
5 LiquidCrystal_I2C lcd(0x20, 16, 2); // Location of the display 0x20 - I2C
   address, 16 x 2 display size
6
7 float Pressure;
8 String DataString;
9 char PrintData[48];
10 /*define the sensor's instance with the sensor's family, sensor's I2C address as
   well as
11 its specified minimum and maximum pressure*/
12 int sensor_model = 5915;
13 int I2C_address = 0x78; // must be defined with a code to search for I2C address
14 int press_min = -500;
15 int press_max = 500;
16 AMS pressure_sensor(int sensor_model, int I2C_address, int press_min, int
   press_max);
17
18 void setup() {
19     Serial.begin(9600);
20     lcd.init();
21     lcd.backlight(); // Initializes display
22 }
23
24 void loop() {
25     if (pressure_sensor.Available() == true) { //check if the pressure sensor
   responds to the given I2C address
26         Pressure = pressure_sensor.readPressure();
27         if (isnan(Pressure)) { //check if an error occurred leading to the function
   returning a NaN
28             Serial.write("Please check the sensor family name.");
29         }
30         else {

```



```
31     DataString = String(Pressure) + " mbar \n"; //put the data into a string
32     DataString.toCharArray(PrintData, 48); //convert string into CharArray
33     // Printing to serial
34     Serial.print("diff. pressure orifice plate: ");
35     Serial.write(PrintData); //write the sensor's data on the serial
port
36     Serial.print(" mbar");
37     // Printing to display
38     lcd.setCursor(0, 0);
39     lcd.print("diff.pres: ");
40     lcd.print(PrintData);
41     lcd.print("mbar");
42 }
43 }
44 else {
45     Serial.write("The sensor did not answer.");
46 }
```

Listing A.5: Differential pressure data processing at the orifice plate.

Appendix B

Pressure

B.1 Matlab codes

```

1 function [P_1, P_2] = pressure_calc(increm_P, beta, q_m, n, m, air_density, D, coef_c)
2
3 %% DATA INPUT
4 P_atm = 101325; % [Pa]
5 Area_int = pi * (D/2)^2;
6
7 aux_1234 = (0.03779 / 0.22679) * 100;
8 aux_12354 = (0.22679 / 0.22679) * 100;
9 q_m_por = linspace(aux_1234, aux_12354, n);
10
11 %% VECTOR DEFINITION
12 velocitat_fluid_1 = zeros(m, n);
13 P_1 = zeros(m, n);
14 P_1_por = zeros(m, n);
15 aux_1 = zeros(m, n);
16 aux_2 = zeros(m, n);
17 press_loss = zeros(m, n);
18 P_2 = zeros(m, n);
19 P_2_por = zeros(m, n);
20 P_2_sist = zeros(1, n);
21 P_2_sist_2 = zeros(1, n);
22 salto_min = zeros(1, n);
23
24 %% CALCULATIONS
25 for i = 1:m
26     for j = 1:n
27         velocitat_fluid_1(i, j) = q_m(1, j) / (air_density * Area_int);
28
29         P_1(i, j) = P_atm - (velocitat_fluid_1(i, j)^2 * air_density) / 2;
30         P_1_por(i, j) = P_1(i, j) / P_atm;
31     end
32 end
33
34 for i = 1:m
35     for j = 1:n
36         aux_1(i, j) = sqrt(1 - beta(1, i)^4*(1-coef_c(i, j)^2));

```

```

37     aux_2(i, j) = coef_c(i, j) * beta(1, i)^2;
38     press_loss(i, j) = ((aux_1(i, j) - aux_2(i, j)) / (aux_1(i, j) + aux_2(i, j))) * increm_P(i, j);
39     P_2(i, j) = P_1(i, j) - press_loss(i, j);
40     P_2_por(i, j) = P_2(i, j) / P_atm;
41
42     end
43 end
44
45 for t = 1:n
46     if t <=37
47         P_2_sist(1, t) = P_2_por(1, t);
48     else
49         P_2_sist(1, t) = P_2_por(m, t);
50     end
51 end
52
53
54 % minimum pressure jump to avoid recirculation
55
56 for k = 1:n
57     if k <= 37
58         P_2_sist_2(1, k) = P_2(1, k);
59     else
60         P_2_sist_2(1, k) = P_2(m, k);
61     end
62 end
63
64 for i = 1:n
65     salto_min(1, i) = P_atm/P_2_sist_2(1, i);
66 end
67
68 %% PLOTS
69
70 set(groot, 'defaultAxesTickLabelInterpreter', 'latex');
71 set(groot, 'defaultTextInterpreter', 'latex');
72 set(groot, 'defaultLegendInterpreter', 'latex');
73
74 P_2_por_1 = P_2_por(1, :);
75 fig10 = figure(10);
76 hold on
77 set(gca, 'FontSize', 13)
78 plot(q_m_por(1:50), P_2_por_1(1:50), LineWidth=2)
79 plot(q_m_por, P_2_por(2, :), LineWidth=2)
80 yline(1)
81 xlabel('\dot{q}_m$ $[\%]', 'FontSize', 15)
82 ylabel(['Pressure far downstream $[\% P_{atm}]$'], 'FontSize', 15)
83 legend('\beta$ = 0,48', '\beta$ = 0,68', 'Location','best');
84 grid on
85 grid minor
86 box on
87 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
88 hold off;
89
90 set(fig10, 'units', 'points');
91 pos = get(fig10, 'position');
92 set(fig10, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
93     'PaperSize',[pos(3), pos(4)]);
94 print(fig10, 'plots/P_2 as a function of airlofw', '-dpdf', '-r0', '-bestfit');

```

```

95 %%%%
96 fig11 = figure(11);
97 hold on
98 set(gca, 'FontSize', 13)
99 plot(q_m_por, P_2_sist, LineWidth=2)
100 xlabel('$\dot{q}_m$ $[\%]$', 'FontSize', 15)
101 ylabel('Pressure far downstream $[\% P_{atm}]$', 'FontSize', 15)
102 grid on
103 grid minor
104 box on
105 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
106 hold off;
107
108 set(fig11, 'units', 'points');
109 pos = get(fig11, 'position');
110 set(fig11, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
111     'PaperSize',[pos(3), pos(4)]);
112 print(fig11, 'plots/P_2 as a function of airlofw both plates', '-dpdf', '-r0', '-bestfit');
113 %%%%
114 fig12 = figure(12);
115 hold on
116 set(gca, 'FontSize', 13)
117 plot(q_m_por, salto_min, LineWidth=2)
118 xlabel('$\dot{q}_m$ $[\%]$', 'FontSize', 15)
119 ylabel('$\left(\frac{P_2}{P_1}\right)_{min}$', 'FontSize', 15)
120 grid on
121 grid minor
122 box on
123 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
124 hold off;
125
126 set(fig12, 'units', 'points');
127 pos = get(fig12, 'position');
128 set(fig12, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
129     'PaperSize',[pos(3), pos(4)]);
130 print(fig12, 'plots/salt de presio minim', '-dpdf', '-r0', '-bestfit');

```

Listing B.1: Calculation of the pressure at the compressor's inlet.

```

1 clear all
2 clc
3 close all
4
5 %% DATA INPUT
6 P_atm = 101325; % [Pa]
7 % P_2
8 precision_transducer = 0.25; % [%]
9 max_press = 6; % [bar]
10 precision_arduino = 0.00366748; % [bar]
11 % P_1
12 accuracy = 0.5; % [%]
13 max_press_p1 = 500; % [mbar]
14 abs_error_ard = (1000 / 13107) * 0.5 * 100; % [Pa]
15 % press. ration
16 m = 3;
17 n = 100;
18 P_1 = [0.90 0.85 0.75]; % [bar]
19 P_2 = linspace(1, 2.75, n); % [bar]

```

```

20
21 %% VECTOR DEFINITION
22 % P_2
23 abs_press_outlet = linspace(1, max_press + 1, n);
24 abs_press_outlet_Pa = abs_press_outlet * 100000;
25 rel_error_transducer = zeros(1, n);
26 rel_error_arduino = zeros(1, n);
27 rel_error_total = zeros(1, n);
28 % P_1
29 expected_p1 = linspace(70927, 99298, n);
30 rel_error_p1_sensor = zeros(1, n);
31 rel_error_p1_ard = zeros(1, n);
32 rel_error_total_p1 = zeros(1, n);
33 % press. ratio
34 press_ratio = zeros(m, n);
35 rel_error_pr = zeros(m, n);
36
37 %% CALCULATIONS
38 abs_error_transducer = (precision_transducer/100) * max_press;           % [bar]
39 abs_error_p1 = (accuracy/100) * max_press_p1 * 100;                     % [Pa]
40
41 for i = 1:n
42     rel_error_transducer(1, i) = (abs_error_transducer / abs_press_outlet(1, i)) * 100; % [bar/bar]
43     rel_error_arduino(1, i) = (precision_arduino/abs_press_outlet(1, i)) * 100; % [bar/bar]
44     rel_error_total(1, i) = rel_error_arduino(1, i) + rel_error_transducer(1, i);
45
46     rel_error_p1_sensor(1, i) = (abs_error_p1/(P_atm-expected_p1(1, i))) * 100; % [Pa/Pa]
47     rel_error_p1_ard(1, i) = (abs_error_ard/(P_atm-expected_p1(1, i))) * 100; % [Pa/Pa]
48     rel_error_total_p1(1, i) = rel_error_p1_ard(1, i) + rel_error_p1_sensor(1, i);
49
50     for j = 1:m
51         press_ratio(j, i) = P_2(1, i) / P_1(1, j);
52         rel_error_pr(j, i) = ((abs_error_transducer / (P_2(1, i))) * 100) + ((precision_arduino/P_2(1, i))
53         ) * 100) + ((abs_error_p1/(P_atm-(P_atm*P_1(1, j)))) * 100) + (abs_error_ard/(P_atm-(P_atm*P_1(1, j))
54         )) * 100;
55     end
56 end
57
58 %% PLOTS
59
60 set(groot,'defaultAxesTickLabelInterpreter','latex');
61 set(groot,'defaultTextInterpreter','latex');
62 set(groot,'defaultLegendInterpreter','latex');
63
64 fig13 = figure(13);
65 hold on
66 set(gca, 'FontSize', 14)
67 plot(abs_press_outlet_Pa, rel_error_transducer, LineWidth=2)
68 xlabel('$P_2$ $[Pa]$', 'FontSize', 16)
69 ylabel('$\Delta P_{sensor}$ $[\%]$', 'FontSize', 16)
70 xlim([100000 300000])
71 grid on
72 grid minor
73 box on
74 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
75 hold off;
76
77 set(fig13, 'units', 'points');

```

```

76 pos = get(fig13, 'position');
77 set(fig13, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
78     'PaperSize',[pos(3), pos(4)]);
79 print(fig13, 'plots/relative error transductor', '-dpdf', '-r0', '-bestfit');
80 %%%
81 fig14 = figure(14);
82 hold on
83 set(gca, 'FontSize', 15)
84 plot(abs_press_outlet_Pa, rel_error_arduino, LineWidth=2)
85 xlabel('$P_2$ $[Pa]$', 'FontSize', 16)
86 ylabel('$\Delta P_{\text{Arduino}}$ $[\%]$', 'FontSize', 16)
87 xlim([100000 300000])
88 grid on
89 grid minor
90 box on
91 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
92 hold off;
93
94 set(fig14, 'units', 'points');
95 pos = get(fig14, 'position');
96 set(fig14, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
97     'PaperSize',[pos(3), pos(4)]);
98 print(fig14, 'plots/relative error arduino', '-dpdf', '-r0', '-bestfit');
99 %%%
100 fig15 = figure(15);
101 hold on
102 set(gca, 'FontSize', 15)
103 plot(abs_press_outlet_Pa, rel_error_total, LineWidth=2)
104 xlabel('$P_2$ $[Pa]$', 'FontSize', 16)
105 ylabel('$\Delta P$ $[\%]$', 'FontSize', 16)
106 xlim([100000 300000])
107 grid on
108 grid minor
109 box on
110 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
111 hold off;
112
113 set(fig15, 'units', 'points');
114 pos = get(fig15, 'position');
115 set(fig15, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
116     'PaperSize',[pos(3), pos(4)]);
117 print(fig15, 'plots/relative error p_2 total', '-dpdf', '-r0', '-bestfit');
118 %%%
119 fig16 = figure(16);
120 hold on
121 set(gca, 'FontSize', 15)
122 plot(expected_p1/P_atm, rel_error_p1_sensor, LineWidth=2)
123 xlabel('$P_1$ $[\% P_{\text{atm}}]$', 'FontSize', 16)
124 ylabel('$\Delta P_{\text{sensor}}$ $[\%]$', 'FontSize', 16)
125 grid on
126 grid minor
127 box on
128 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
129 hold off;
130
131 set(fig16, 'units', 'points');
132 pos = get(fig16, 'position');
133 set(fig16, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...

```

```

134     'PaperSize',[pos(3), pos(4)];
135 print(fig16, 'plots/relative error p_1 sensor', '-dpdf', '-r0', '-bestfit');
136 %%%
137 fig17 = figure(17);
138 hold on
139 set(gca, 'FontSize', 15)
140 plot(expected_p1/P_atm, rel_error_p1_ard, LineWidth=2)
141 xlabel('$P_1$ $[\% P_{atm}]$', 'FontSize', 16)
142 ylabel('$\delta P_{Arduino}$ $[\%]$', 'FontSize', 16)
143 grid on
144 grid minor
145 box on
146 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
147 hold off;
148
149 set(fig17, 'units', 'points');
150 pos = get(fig17, 'position');
151 set(fig17, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
152     'PaperSize',[pos(3), pos(4)]);
153 print(fig17, 'plots/relative error p_1 ard', '-dpdf', '-r0', '-bestfit');
154 %%%
155 fig18 = figure(18);
156 hold on
157 set(gca, 'FontSize', 15)
158 plot(expected_p1/P_atm, rel_error_total_p1, LineWidth=2)
159 xlabel('$P_1$ $[\% P_{atm}]$', 'FontSize', 16)
160 ylabel('$\delta P_{sensor}$ $[\%]$', 'FontSize', 16)
161 grid on
162 grid minor
163 box on
164 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
165 hold off;
166
167 set(fig18, 'units', 'points');
168 pos = get(fig18, 'position');
169 set(fig18, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
170     'PaperSize',[pos(3), pos(4)]);
171 print(fig18, 'plots/relative error p_1 total', '-dpdf', '-r0', '-bestfit');
172 %%%
173 fig19 = figure(1);
174 hold on
175 set(gca, 'FontSize', 15)
176 plot(press_ratio(1,:), rel_error_pr(1,:), LineWidth=2)
177 plot(press_ratio(2,:), rel_error_pr(2,:), LineWidth=2)
178 plot(press_ratio(3,:), rel_error_pr(3,:), LineWidth=2)
179 xlabel('$\Pi$', 'FontSize', 16)
180 ylabel('$\delta \Pi$ $[\%]$', 'FontSize', 16)
181 legend('$P_1= 0,9 \ \% P_{atm}$', '$P_1 =0,85 \ \% P_{atm}$', '$P_1 =0,75 \ \% P_{atm}$', 'Location','best');
182 xlim([1 2.8])
183 grid on
184 grid minor
185 box on
186 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
187 hold off;
188
189 set(fig19, 'units', 'points');
190 pos = get(fig19, 'position');
191 set(fig19, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...

```

```
192 'PaperSize',[pos(3), pos(4)]);
193 print(fig19, 'plots/relative error press_ratio', '-dpdf', '-r0', '-bestfit');
```

Listing B.2: Pressure measurement and pressure ratio error.

B.2 Arduino data processing codes

The data processing of the differential pressure between the compressor's inlet pressure and the ambient pressure can be developed the same way as code A.5.

```
1 #include <LiquidCrystal_I2C.h> // allows communications with an I2C display
2
3 #include "Wire.h" //allows communication over i2c devices
4
5 const int pressureInput = A0; //select the analog input pin for the pressure
   transducer
6 const int pressureZero = 205; //analog reading of pressure transducer at 0bar
7 const int pressureMax = 1023; //analog reading of pressure transducer at 6bar
8 const int pressuretransducermaxBAR = 6; //psi value of transducer being used
9 const int baudRate = 9600; //constant integer to set the baud rate for serial
   monitor
10 const int sensorreadDelay = 250; //constant integer to set the sensor read delay
   in milliseconds
11 const int P_atm = 1;
12
13 float pressureValue = 0; //variable to store the value coming from the pressure
   transducer
14
15 LiquidCrystal_I2C lcd(0x20, 16, 2); // Location of the display 0x20 - I2C
   address, 16 x 2 display size
16
17 void setup() //setup routine, runs once when system turned on or reset
18 {
19   Serial.begin(baudRate); //initializes serial communication at set baud rate bits
   per second
20   lcd.init();
21   lcd.backlight();
22 }
23
24 void loop() //loop routine runs over and over again forever
25 {
26   pressureValue = analogRead(pressureInput); //reads value from input pin and
   assigns to variable
27   Serial.print(pressureValue);
```



```
28  pressureValue = ((pressureValue-pressureZero)*pressuretransducermaxBAR)/(
    pressureMax-pressureZero) + P_atm; //conversion equation to convert analog
    reading to bar
29
30  // Printing to serial
31  Serial.print("press: ");
32  Serial.print(pressureValue, 2);
33  Serial.println("bar");
34  // Printing to display
35  lcd.setCursor(0, 0);
36  lcd.print("press: ");
37  lcd.print(pressureValue, 2);
38  lcd.print("bar");
39  delay(sensorreadDelay); //delay in milliseconds between read values
40 }
```

Listing B.3: Compressor's output pressure with transducer data processing.

Appendix C

Tachometer

C.1 Matlab codes

```

1 clear all
2 clc
3 close all
4
5 %% DATA INPUT
6 measure_error = 0.5; % [us]
7 min_rpm = 34000; % [rpm]
8 max_rpm = 85000; % [rpm]
9 max_freq = 3000; % [Hz]
10 sensor_acc = 0.01; % [1%]
11 n = 100;
12
13 %% VECTOR DEFINITION
14 rpms = linspace(min_rpm, max_rpm, n);
15 rel_error_rpm = zeros(1, n);
16 freq = zeros(1, n);
17 period = zeros(1, n);
18 rel_error_rpm_sensor = zeros(1, n);
19 rel_error_rpm_tot = zeros(1, n);
20
21 %% CALCULATIONS
22 T_error = measure_error * 2;
23 sensor_error = sensor_acc * max_freq;
24
25 for i = 1:n
26     freq(1, i) = rpms(1, i) / 60;
27     period(1, i) = 1 / freq(1, i);
28     rel_error_rpm(1, i) = ((T_error*10^-6) / period(1, i)) * 100;
29     rel_error_rpm_sensor(1, i) = (sensor_error / freq(1, i)) * 100;
30     rel_error_rpm_tot(1, i) = rel_error_rpm_sensor(1, i) + rel_error_rpm(1, i);
31 end
32
33 %% PLOTS
34 set(groot, 'defaultAxesTickLabelInterpreter', 'latex');
35 set(groot, 'defaultTextInterpreter', 'latex');
36 set(groot, 'defaultLegendInterpreter', 'latex');

```

```

37
38 fig16 = figure(16);
39 hold on
40 set(gca, 'FontSize', 13)
41 plot(rpms, rel_error_rpm, LineWidth=2)
42 xlabel('$rpm$', 'FontSize', 15)
43 ylabel('$\delta rpm$ $[\%]$', 'FontSize', 15)
44 grid on
45 grid minor
46 box on
47 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
48 hold off;
49
50 set(fig16, 'units', 'points');
51 pos = get(fig16, 'position');
52 set(fig16, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
53     'PaperSize',[pos(3), pos(4)]);
54 print(fig16, 'plots/relative error arduino_rpm', '-dpdf', '-r0', '-bestfit');
55
56 fig17 = figure(17);
57 hold on
58 set(gca, 'FontSize', 13)
59 plot(rpms, rel_error_rpm_sensor, LineWidth=2)
60 xlabel('$rpm$', 'FontSize', 15)
61 ylabel('$\delta rpm$ $[\%]$', 'FontSize', 15)
62 grid on
63 grid minor
64 box on
65 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
66 hold off;
67
68 set(fig17, 'units', 'points');
69 pos = get(fig17, 'position');
70 set(fig17, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
71     'PaperSize',[pos(3), pos(4)]);
72 print(fig17, 'plots/relative error sensor_rpm', '-dpdf', '-r0', '-bestfit');
73
74 fig18 = figure(18);
75 hold on
76 set(gca, 'FontSize', 13)
77 plot(rpms, rel_error_rpm_tot, LineWidth=2)
78 xlabel('$rpm$', 'FontSize', 15)
79 ylabel('$\delta rpm$ $[\%]$', 'FontSize', 15)
80 grid on
81 grid minor
82 box on
83 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
84 hold off;
85
86 set(fig18, 'units', 'points');
87 pos = get(fig18, 'position');
88 set(fig18, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
89     'PaperSize',[pos(3), pos(4)]);
90 print(fig18, 'plots/relative error tot_rpm', '-dpdf', '-r0', '-bestfit');

```

Listing C.1: Revolutions measurement error.

C.2 Arduino data processing codes

```
1 #include <LiquidCrystal_I2C.h>
2
3 #include "Wire.h" //allows communication over i2c devices
4
5 LiquidCrystal_I2C lcd(0x20, 16, 2); // Location of the display 0x20 - I2C
   address, 16 x 2 display size
6
7 unsigned long Htime = 0; //integer for storing high time
8
9 unsigned long Ltime = 0; //integer for storing low time
10
11 double Ttime; // integer for storing total time of a cycle
12
13 double frequency; //storing frequency
14 double rpm;
15
16 const int baudRate = 9600;
17 const int sensorreadDelay = 5;
18
19 void setup()
20
21 {
22   pinMode(8,INPUT); // reads digital value from input pin 8
23
24   Serial.begin(baudRate);
25   lcd.init();
26   lcd.backlight();
27 }
28
29 void loop() {
30   Htime = pulseIn(8, HIGH); // measures high time at input pin 8
31   Ltime = pulseIn(8, LOW); // measures low time at input pin 8
32   Ttime = (Htime + Ltime); // total time, period
33   frequency = 1000000 / Ttime;
34   rpm = frequency * 60;
35
36   // Printing to serial
37   Serial.print("rpm: ");
38   Serial.print(rpm);
39   Serial.print(" rpm");
40   // Printing to display
```

```
41  lcd.setCursor(0, 0);  
42  lcd.print(rpm);  
43  lcd.print(" rpm");  
44  }
```

Listing C.2: Data processing of the revolutions measurement.

Appendix D

Thermometer

D.1 Matlab codes

```

1 clear all
2 clc
3 close all
4
5 %% DATA INPUT
6 T_min = 30;
7 T_max = 190;
8 n = 100;
9 a = -83.151;
10 b = 685.7;
11 c = -2078.8;
12 d = 2210.4;
13 DeltaV = 0.00244;
14
15 %% VECTOR DEFINITION
16 temp = linspace(T_min, T_max, n);
17 temp_abs_error = zeros(1, n);
18 temp_rel_error = zeros(1, n);
19 V_r = zeros(1, n);
20 temp_abs_error_ard = zeros(1, n);
21 temp_rel_error_ard = zeros(1, n);
22 temp_rel_error_total = zeros(1, n);
23
24 %% CALCULATIONS
25 for i = 1:n
26     % sensor
27     temp_abs_error(1, i) = 0.3 + 0.005 * temp(1, i);
28     temp_rel_error(1, i) = (temp_abs_error(1, i) / temp(1, i)) * 100;
29     % arduino
30     fun = @(V_r) a*V_r^3 + b*V_r^2 + c*V_r + d - temp(1, i);
31     result = fzero(fun, 2);
32     V_r(1, i) = result;
33
34     temp_abs_error_ard(1, i) = (3*a*V_r(1, i)^2 + 2*b*V_r(1, i) + c) * DeltaV;
35     temp_rel_error_ard(1, i) = (abs(temp_abs_error_ard(1, i)) / temp(1, i)) * 100;
36

```

```

37     temp_rel_error_total(1, i) = temp_rel_error(1, i) + temp_rel_error_ard(1, i);
38 end
39
40 %% PLOTS
41 set(groot, 'defaultAxesTickLabelInterpreter', 'latex');
42 set(groot, 'defaultTextInterpreter', 'latex');
43 set(groot, 'defaultLegendInterpreter', 'latex');
44
45 fig20 = figure(20);
46 hold on
47 set(gca, 'FontSize', 13)
48 plot(temp, temp_rel_error, LineWidth=2)
49 xlabel('T [°C]', 'FontSize', 15)
50 ylabel('Δ T [%]', 'FontSize', 15)
51 xlim([30 190])
52 grid on
53 grid minor
54 box on;
55 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
56 hold off;
57
58 set(fig20, 'units', 'points');
59 pos = get(fig20, 'position');
60 set(fig20, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
61     'PaperSize', [pos(3), pos(4)]);
62 print(fig20, 'plots/relative error temp_sensor', '-dpdf', '-r0', '-bestfit');
63
64 fig21 = figure(21);
65 hold on
66 set(gca, 'FontSize', 13)
67 plot(temp, temp_rel_error_ard, LineWidth=2)
68 xlabel('T [°C]', 'FontSize', 15)
69 ylabel('Δ T [%]', 'FontSize', 15)
70 xlim([30 190])
71 grid on
72 grid minor
73 box on;
74 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
75 hold off;
76
77 set(fig21, 'units', 'points');
78 pos = get(fig21, 'position');
79 set(fig21, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
80     'PaperSize', [pos(3), pos(4)]);
81 print(fig21, 'plots/relative error temp_ard', '-dpdf', '-r0', '-bestfit');
82
83 fig22 = figure(22);
84 hold on
85 set(gca, 'FontSize', 13)
86 plot(temp, temp_rel_error_total, LineWidth=2)
87 xlabel('T [°C]', 'FontSize', 15)
88 ylabel('Δ T [%]', 'FontSize', 15)
89 xlim([30 190])
90 grid on
91 grid minor
92 box on;
93 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
94 hold off;

```

```

95
96 set(fig22, 'units', 'points');
97 pos = get(fig22, 'position');
98 set(fig22, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
99     'PaperSize', [pos(3), pos(4)]);
100 print(fig22, 'plots/relative error temp_tot', '-dpdf', '-r0', '-bestfit');

```

Listing D.1: Revolutions measurement error.

D.2 Arduino data processing codes

```

1 #include <LiquidCrystal_I2C.h>           // allows to control I2C displays
2 #include <Wire.h>                       //allows communication over i2c devices
3
4 LiquidCrystal_I2C lcd(0x20, 16, 2);     // Location of the display 0x20 - I2C
   address, 16 x 2 display size
5
6 // Temperature //
7 const int tempInput = A1;               // Analog input pin for the temperature
8 float tempValue;                       // Variable to store the value coming from the
   temperature
9 float temperature;                     // Storing temperature value
10 float x;                               // Value for developing the equations
11 const float a = -83.151;                // Cubic term of the equation
12 const float b = 685.7;                  // Squared term of the equation
13 const float c = -2078.8;                // Lineal term of the equation
14 const float d = 2210.4;                 // Constant term of the equation
15 const int baudRate = 9600;              //constant integer to set the baud rate
   for serial monitor
16 const int sensorreadDelay = 250;
17
18 void setup()
19 {
20   Serial.begin(baudRate);                // Initializes serial communication
21   lcd.init();
22   lcd.backlight();                       // Initializes display
23 }
24
25 void loop() {
26   tempValue = analogRead(tempInput);     // Reads value from input pin
27   x = (tempValue/1023) * 5;              // Conversion equation to convert the
   analog value to V
28   temperature = a*x^3 + b*x^2 + c*x + d; // Cubic equation of the relation V
   - C
29

```



```

30 // Printing to serial
31 Serial.print("Temperature: ");
32 Serial.print(temperature, 1);
33 // Printing to display
34 lcd.setCursor(0, 0);
35 lcd.print("Temp: ");
36 lcd.print(temperature, 1);
37 lcd.print(" C");
38 }

```

Listing D.2: Data processing of the temperature at the compressor outlet.

D.3 PT100 table

° C	0	1	2	3	4	5	6	7	8	9
0	100.00	100.39	100.78	101.17	101.56	101.95	102.34	102.73	103.12	103.51
10	103.90	104.29	104.68	105.07	105.46	105.85	106.24	106.63	107.02	107.40
20	107.79	108.18	108.57	108.96	109.35	109.73	110.12	110.51	110.90	111.28
30	111.67	112.06	112.45	112.83	113.22	113.61	113.99	114.38	114.77	115.15
40	115.54	115.93	116.31	116.70	117.08	117.47	117.85	118.24	118.62	119.01
50	119.40	119.78	120.16	120.55	120.93	121.32	121.70	122.09	122.47	122.86
60	123.24	123.62	124.01	124.39	124.77	125.16	125.54	125.92	126.31	126.69
70	127.07	127.45	127.84	128.22	128.60	128.98	129.37	129.75	130.13	130.51
80	130.89	131.27	131.66	132.04	132.42	132.80	133.18	133.56	133.94	134.32
90	134.70	135.08	135.46	135.84	136.22	136.60	136.98	137.36	137.74	138.12
100	138.50	138.88	139.26	139.64	140.02	140.39	140.77	141.15	141.53	141.91
110	142.29	142.66	143.04	143.42	143.80	144.17	144.55	144.93	145.31	145.68
120	146.06	146.44	146.81	147.19	147.57	147.94	148.32	148.70	149.07	149.45
130	149.82	150.20	150.57	150.95	151.33	151.70	152.08	152.45	152.83	153.20
140	153.58	153.95	154.32	154.70	155.07	155.45	155.82	156.19	156.57	156.94
150	157.31	157.69	158.06	158.43	158.81	159.18	159.55	159.93	160.30	160.67
160	161.04	161.42	161.79	162.16	162.53	162.90	163.27	163.65	164.02	164.39
170	164.76	165.13	165.50	165.87	166.24	166.61	166.98	167.35	167.72	168.09
180	168.46	168.83	169.20	169.57	169.94	170.31	170.68	171.05	171.42	171.79
190	172.16	172.53	172.90	173.26	173.63	174.00	174.37	174.74	175.10	175.47
200	175.84	176.21	176.57	176.94	177.31	177.68	178.04	178.41	178.78	179.14
210	179.51	179.88	180.24	180.61	180.97	181.34	181.71	182.07	182.44	182.80
220	183.17	183.53	183.90	184.26	184.63	184.99	185.36	185.72	186.09	186.45
230	186.82	187.18	187.54	187.91	188.27	188.63	189.00	189.36	189.72	190.09
240	190.45	190.81	191.18	191.54	191.90	192.26	192.63	192.99	193.35	193.71
250	194.07	194.44	194.80	195.16	195.52	195.88	196.24	196.60	196.96	197.33

Table D.1: Resistance value of a PT100 as a function of temperature [1].

Appendix E

Efficiency error

E.1 Matlab codes

```

1 clear all
2 clc
3 close all
4
5 %% DATA INPUT
6 T_1 = 25+273; % [ K ]
7 P_atm = 101325; % [Pa]
8 m = 5;
9 press_ratio = [1.5 1.7 2 2.3 2.7];
10 P_1 = 0.85; % [bar]
11 abs_error_transducer = 0.0150;
12 precision_arduino = 0.00366748;
13 abs_error_p1 = 250;
14 abs_error_ard = 3.81475547;
15 T_min = 30;
16 T_max = 190;
17 n = 100;
18 a = -83.151;
19 b = 685.7;
20 c = -2078.8;
21 d = 2210.4;
22 DeltaV = 0.00244;
23
24 %% VECTOR DEFINITION
25 P_2 = zeros(1, m);
26 rel_error_pr = zeros(1, m);
27 abs_error_pr = zeros(1, m);
28 aux_1 = zeros(m, n);
29 aux_1_2 = zeros(m, n);
30 aux_2 = zeros(m, n);
31 aux_2_2 = zeros(m, n);
32 eff_abs_error = zeros(m, n);
33 eff_rel_error = zeros(m, n);
34 rendimiento = zeros(m, n);
35
36 %% CALCULATIONS

```

```

37 for i = 1:m
38     P_2(1, i) = press_ratio(1, i) * P_1;    % bar
39     rel_error_pr(1, i) = ((abs_error_transducer / (P_2(1, i))) * 100) + ((precision_arduino/P_2(1, i)) *
40     100) + (abs_error_p1/(P_atm-(P_atm*P_1)) * 100) + (abs_error_ard/(P_atm-(P_atm*P_1))) * 100;
41     abs_error_pr(1, i) = rel_error_pr(1, i)/100 * press_ratio(1, i);
42 end
43 %% VECTOR DEFINITION
44 temp = linspace(T_min, T_max, n);
45 temp_abs_error = zeros(1, n);
46 temp_rel_error = zeros(1, n);
47 V_r = zeros(1, n);
48 temp_abs_error_ard = zeros(1, n);
49 temp_rel_error_ard = zeros(1, n);
50 temp_rel_error_total = zeros(1, n);
51 temp_abs_error_tot = zeros(1, n);
52
53 % Tenoerature error
54 for i = 1:n
55     % sensor
56     temp_abs_error(1, i) = 0.3 + 0.005 * temp(1, i);
57     temp_rel_error(1, i) = (temp_abs_error(1, i) / temp(1, i)) * 100;
58
59     % arduino
60     fun = @(V_r) a*V_r^3 + b*V_r^2 + c*V_r + d - temp(1, i);
61     result = fzero(fun, 2);
62     V_r(1, i) = result;
63
64     temp_abs_error_ard(1, i) = (3*a*V_r(1, i)^2 + 2*b*V_r(1, i) + c) * DeltaV;
65     temp_rel_error_ard(1, i) = (abs(temp_abs_error_ard(1, i)) / temp(1, i)) * 100;
66
67
68     temp_rel_error_total(1, i) = temp_rel_error(1, i) + temp_rel_error_ard(1, i);
69     temp_abs_error_tot(1, i) = temp_rel_error_total(1, i)/100 * temp(1, i);
70 end
71
72 %% EFFICIENCY ERROR
73 temp = temp + 273;
74 for j = 1:m
75     for i = 1:n
76         % parte de press ratio
77         aux_1(j, i) = (T_1*0.4*press_ratio(1, j)^(-5/7)) / ((temp(1, i) - T_1)*1.4);
78         aux_1_2(j, i) = aux_1(j, i) * abs_error_pr(1, j);
79         % parte temp
80         aux_2(j, i) = (T_1*(press_ratio(1, j)^(2/7)-1)) / ((temp(1, i) - T_1)^2);
81         aux_2_2(j, i) = aux_2(j, i) * temp_abs_error_tot(1, i);
82         % total
83         eff_abs_error(j, i) = sqrt(aux_1_2(j, i)^2 + aux_2_2(j, i)^2);
84         % rendimiento
85         rendimiento(j, i) = ((T_1)/((1/press_ratio(1, j))^(2/7)) - T_1) / (temp(1, i) - T_1);
86         % relative error
87         eff_rel_error(j, i) = (eff_abs_error(j, i) / rendimiento(j, i)) * 100;
88     end
89 end
90
91 %% PLOTS
92 set(groot, 'defaultAxesTickLabelInterpreter', 'latex');
93 set(groot, 'defaulttextinterpreter', 'latex');

```

```
94 set(groot,'defaultLegendInterpreter','latex');
95
96 fig22 = figure(22);
97 hold on
98 set(gca, 'FontSize', 15)
99 plot(rendimiento(1,:), eff_rel_error(1,:), LineWidth=2)
100 plot(rendimiento(2,:), eff_rel_error(2,:), LineWidth=2)
101 plot(rendimiento(3,:), eff_rel_error(3,:), LineWidth=2)
102 plot(rendimiento(4,:), eff_rel_error(4,:), LineWidth=2)
103 plot(rendimiento(5,:), eff_rel_error(5,:), LineWidth=2)
104 xlim([0.55 0.87])
105 xlabel('Efficiency,  $\eta$ ', 'FontSize', 16)
106 ylabel('$\delta \eta$ [%]', 'FontSize', 16)
107 lgd = legend('$\Pi=1,5$', '$\Pi=1,7$', '$\Pi = 2$', '$\Pi = 2,3$', '$\Pi=2,7$', 'Location','best');
108 grid on
109 grid minor
110 box on
111 set(gcf, 'units', 'points', 'position', [50,50,676,420]);
112 hold off;
113
114 set(fig22, 'units', 'points');
115 pos = get(fig22, 'position');
116 set(fig22, 'PaperPositionMode', 'Auto', 'PaperUnits', 'points', ...
117     'PaperSize', [pos(3), pos(4)]);
118 print(fig22, 'plots/relative error efficiency', '-dpdf', '-r0', '-bestfit');
```

Listing E.1: Efficiency calculation error.

Appendix F

System unification

F.1 Matlab code for obtaining a compressor map point

```

1 clear all
2 clc
3 close all
4
5 %% DATA INPUT
6
7 air_density = 1.225;
8 visosity = 1.802 * 10^-5;
9 D = 59.25 * 10^-3;
10 P_atm = 101325;
11 k = 1.4;
12
13 %% VARIABLES
14 % Orifice plate used
15 beta = ;
16 T_o = 298;
17
18 % Measurements
19 DeltaP = ;           % differential pressure at the orifice palte in Pa
20 P_1 = ;             % differential pressure at the compressor's inlet in Pa
21 P_2 = ;             % differential pressure at the compressor's outlet in bar
22 temp = ;           % temperature at the compressor's outlet in C
23 rpm = ;            % revolutions measurement in rpm
24
25 %% MASS AIRFLOW CALCULATION
26 [q_m, uncertainty] = mass_airflow_calc(air_density, beta, DeltaP, visosity, D, P_atm, k)
27 %% PRESSURE RATIO CALCULATION
28 [press_ratio, relative_error_press_ratio] = press_ratio_calculation(P_2, P_atm, P_1)
29 %% EFFICIENCY CALCULATION
30 [efficiency, relative_error_eff] = efficiency_calculation(temp, T_o, k, press_ratio,
    relative_error_press_ratio)
31 %% RPM
32 [rpm, rel_error_rpm_tot] = rpm_calculation(rpm)

```

Listing F.1: Calculations to obtain a compressor map's point for a given measures done.

```

1 function [q_m, uncertainty] = mass_airflow_calc(air_density, beta, DeltaP, visosity, D, P_atm, k)
2 %% DATA
3 q_m_aux = 0.1;
4 error = 100;
5
6 %% CALCULATION
7 while error > 10^-5
8     reynolds_D = (4*q_m_aux) / (pi*visosity*D);
9
10    % coef c
11    aux_A = ((19000*beta) / reynolds_D)^0.8;
12    aux_c_1 = 0.5961 + 0.0261*beta^2 - 0.216*beta^8;
13    aux_c_2 = 0.000521*((10^6*beta)/reynolds_D)^0.7;
14    aux_c_3 = (0.0188 + 0.0063*aux_A) * beta^3.5*(10^6/reynolds_D)^0.3;
15    % aux_c_4 = 0;
16    aux_c_5 = 0.011 * (0.75-beta) * (2.8-(D*10^3)/25.4);
17
18    coef_c = aux_c_1 + aux_c_2 + aux_c_3 + aux_c_5;
19
20    % coef e
21    P_2 = P_atm - DeltaP;
22    coef_e = 1 - (0.351 + 0.256*beta^4 + 0.93*beta^8) * (1 - (P_2/P_atm)^(1/k));
23
24
25    % mass airflow calculation
26    q_m = (coef_c / sqrt(1-beta^4)) * coef_e * pi/4 * beta^2 * D^2 * sqrt(2*DeltaP*air_density);
27    error = abs(q_m_aux - q_m);
28    q_m_aux = q_m;
29 end
30
31 %% uncertainty
32 % uncert_d
33 uncert_d = ((2*beta^4) / (1-beta^4))^2 * (0.05/59.25)^2;
34 % uncert C_d
35 if beta < 0.6
36     uncert_Cd = 0.5 + 0.9*(0.75-beta)*(2.8-(D*10^3)/25.4);
37 else
38     uncert_Cd = (1.667*beta - 0.5) + 0.9*(0.75-beta)*(2.8-(D*10^3)/25.4);
39 end
40 % uncert e
41 uncert_e = 3.5 * (DeltaP / (k * P_atm));
42 % uncert_p
43 min_press_increm = 250 + 3.815;
44 uncert_p = 1/4*(min_press_increm/DeltaP)^2;
45
46 uncertainty_aux = sqrt(uncert_p + uncert_d + (uncert_Cd/100)^2 + (uncert_e/100)^2);
47 uncertainty = uncertainty_aux * 100;

```

Listing F.2: \dot{q}_m and $\delta\dot{q}_m$ calculations function for a given β and ΔP values.

```

1 function [press_ratio, relative_error_press_ratio] = press_ratio_calculation(P_2, P_atm, P_1)
2 %% DATA
3 abs_error_transducer = (0.25/100) * 6;
4 precision_arduino = 0.00366748;
5 abs_error_p1 = (0.5/100) * 500 * 100;
6 abs_error_ard = (1000 / 13107) * 0.5 * 100;
7
8 %% PRESSRE RATIO CALCULATION

```

```

9 P_2_pa = P_2 * P_atm;
10 press_ratio = P_2_pa / P_1;
11
12 %% RELATIVE ERROR CALCULATION
13 aux_1 = (abs_error_transducer / P_2) * 100;
14 aux_2 = (precision_arduino/P_2) * 100;
15 aux_3 = (abs_error_p1/(P_atm-P_1)) * 100;
16 aux_4 = (abs_error_ard/(P_atm-P_1)) * 100;
17 relative_error_press_ratio = aux_4 + aux_3 + aux_2 + aux_1;

```

Listing F.3: Π and $\delta\Pi$ calculations function for a given P_1 and P_2 values.

```

1 function [efficiency, relative_error_eff] = efficiency_calculation(temp, T_o, k, press_ratio,
   relative_error_press_ratio)
2 %% DATA
3 a = -83.151;
4 b = 685.7;
5 c = -2078.8;
6 d = 2210.4;
7 DeltaV = 0.00244;
8
9 %% EFFICIENCY CALC
10 T_2_k = temp + 273;
11 T_s = T_o / ((1/press_ratio)^((k-1)/k));
12 efficiency = (T_s - T_o) / (T_2_k - T_o);
13
14 %% EFFICIENCY RELATIVE ERROR
15 % temperature error
16 % sensor
17 temp_abs_error = 0.3 + 0.005 * temp;
18 temp_rel_error = (temp_abs_error / temp) * 100;
19 % arduino
20 fun = @(V_r) a*V_r^3 + b*V_r^2 + c*V_r + d - temp;
21 result = fzero(fun, 2);
22 V_r = result;
23 temp_abs_error_ard = (3*a*V_r^2 + 2*b*V_r + c) * DeltaV;
24 temp_rel_error_ard = (abs(temp_abs_error_ard) / temp) * 100;
25
26 temp_rel_error_total = temp_rel_error + temp_rel_error_ard;
27 temp_abs_error_tot = temp_rel_error_total/100 * temp;
28
29 % pressure ratio absolute error
30 abs_error_pr = relative_error_press_ratio/100 * press_ratio;
31
32 % efficiency error
33 % press_ratio_part
34 aux_1 = (T_o*0.4*press_ratio^(-5/7)) / ((T_2_k - T_o)*1.4);
35 aux_1_2 = aux_1 * abs_error_pr;
36 % parte temp
37 aux_2 = (T_o*(press_ratio^(2/7)-1)) / ((T_2_k - T_o)^2);
38 aux_2_2 = aux_2 * temp_abs_error_tot;
39 % total
40 eff_abs_error = sqrt(aux_1_2^2 + aux_2_2^2);
41 relative_error_eff = (eff_abs_error / efficiency)*100;

```

Listing F.4: η and $\delta\eta$ calculations function for a given Π and T_2 values.

```

1 function [rpm, rel_error_rpm_tot] = rpm_calculation(rpm)
2 %% ERROR CALCULATION
3 measure_error = 0.5; % [us]
4 max_freq = 3000; % [Hz]
5 sensor_acc = 0.01;
6 T_error = measure_error * 2;
7 sensor_error = sensor_acc * max_freq;
8
9 frequency = rpm / 60;
10 period = 1 / frequency;
11
12 rel_error_rpm = ((T_error*10^-6) / period) * 100;
13 rel_error_rpm_sensor = (sensor_error / frequency) * 100;
14 rel_error_rpm_tot = rel_error_rpm_sensor + rel_error_rpm;

```

Listing F.5: *rpm* and δrpm calculations function for a given *rpm* value.

F.2 Arduino data processing codes

```

1 #include <Wire.h> //allows communication over i2c devices
2 #include <LiquidCrystal_I2C.h> // allows to control I2C displays
3 #include <AMS.h> //include the AMS library
4
5 LiquidCrystal_I2C lcd1(0x20, 20, 4); // Location of the display
6
7 // Differential pressure orifice plate //
8 float Pressure_op;
9 String DataString_op;
10 char PrintData_op[48];
11 /*define the sensor's instance with the sensor's family, sensor's I2C address as
12 well as
13 its specified minimum and maximum pressure*/
14 int sensor_model = 5915;
15 int I2C_address_op = 0x78; // must be defined with a code to search for I2C
16 address
17 int press_min = -500;
18 int press_max = 500;
19 AMS pressure_sensor_op(int sensor_model, int I2C_address_op, int press_min, int
20 press_max);
21
22 // Differential pressure inlet compressor //
23 float Pressure_ic;
24 String DataString_ic;
25 char PrintData_ic[48];
26 /*define the sensor's instance with the sensor's family, sensor's I2C address as
27 well as

```



```

24 its specified minimum and maximum pressure*/
25 int sensor_model = 5915;
26 int I2C_address_ic = 0x78; // must be defined with a code to search for I2C
    address
27 int press_min = -500;
28 int press_max = 500;
29 AMS pressure_sensor_ic(int sensor_model, int I2C_address_ic, int press_min, int
    press_max);
30
31 // Pressure transducer //
32 const int pressureInput = A0;           // Analog input pin for the pressure
    transducer
33 const int pressureMin = 205;           // Analog reading at 0 bar
34 const int pressureMax = 1023;         // Analog reading at 6 bar
35 const int pressuretransducermaxBAR = 6; // Maximum pressure measurable
36 const int P_atm = 1;
37 float pressureValue;                  // Variable to store the value coming from
    the pressure transducer
38
39 // Frequency //
40 const int freqInput = 8;              // Digital input pin for the frequency
    measure
41 unsigned long Htime = 0;              // Integer for storing high time
42 unsigned long Ltime = 0;              // Integer for storing low time
43 float Ttime;                          // Integer for storing total time of a cycle
44 float frequency;                      // Storing frequency
45 float rpm;                            // Storing rpm value
46
47 // Temperature //
48 const int tempInput = A1;             // Analog input pin for the temperature
49 float tempValue;                      // Variable to store the value coming from the
    temperature
50 float temperature;                   // Storing temperature value
51 float x;                              // Value for developing the equations
52 const float a = -83.151;              // Cubic term of the equation
53 const float b = 685.7;                // Squared term of the equation
54 const float c = -2078.8;              // Lineal term of the equation
55 const float d = 2210.4;               // Constant term of the equation
56
57 const int baudRate = 9600;           // Constant integer to set the baud rate
    for serial monitor

```

```
58 const int sensorreadDelay = 100;           // Constant integer to set the sensor
      read delay in milliseconds
59
60 void setup() {
61     pinMode(freqInput, INPUT);             // Pin 8, where the frequency is read, is
      set up as an input pin
62
63     Serial.begin(baudRate);                // Initializes serial communication
64     lcd1.init();                           // Initializes display
65     lcd1.backlight();
66     lcd2.init();                           // Initializes display
67     lcd2.backlight();
68 }
69
70 void loop() {
71     // Differential pressure orifice plate //
72     if (pressure_sensor_op.Available() == true) { //check if the pressure sensor
      responds to the given I2C address
73         Pressure_op = pressure_sensor_op.readPressure();
74         if (isnan(Pressure_op)) { //check if an error occurred leading to the
      function returning a NaN
75             Serial.write("Please check the sensor family name.");
76         }
77         else {
78             DataString_op = String(Pressure_op) + " mbar \n"; //put the data into a
      string
79             DataString_op.toCharArray(PrintData_op, 48); //convert string into
      CharArray
80             // Printing to serial
81             Serial.print("diff. pressure orifice plate: ");
82             Serial.write(PrintData_op); //write the sensor's data on the serial
      port
83             Serial.print(" mbar");
84             // Printing to display
85             lcd.setCursor(0, 0);
86             lcd.print("diff.pres: ");
87             lcd.print(PrintData_op);
88             lcd.print("mbar");
89         }
90     }
91     else {
92         Serial.write("The sensor did not answer.");
```

```

93 }
94
95 // Differential pressure inlet compressor //
96 if (pressure_sensor_ic.Available() == true) { //check if the pressure sensor
    responds to the given I2C address
97     Pressure_ic = pressure_sensor_ic.readPressure();
98     if (isnan(Pressure_op)) { //check if an error occurred leading to the
        function returning a NaN
99         Serial.write("Please check the sensor family name.");
100     }
101     else {
102         DataString_ic = String(Pressure_ic) + " mbar \n"; //put the data into a
        string
103         DataString_ic.toCharArray(PrintData_ic, 48); //convert string into
        CharArray
104         // Printing to serial
105         Serial.print("diff. pressure inlet compressor: ");
106         Serial.write(PrintData_ic); //write the sensor's data on the serial
        port
107         Serial.print(" mbar");
108         // Printing to display
109         lcd.setCursor(0, 1);
110         lcd.print("diff.pres: ");
111         lcd.print(PrintData_ic);
112         lcd.print("mbar");
113     }
114 }
115 else {
116     Serial.write("The sensor did not answer.");
117 }
118
119 // Pressure //
120 pressureValue = analogRead(pressureInput); // Reads value from input pin
121 pressureValue = ((pressureValue-pressureMin)*pressuretransducermaxBAR)/(
    pressureMax-pressureMin) + P_atm; // Conversion equation to convert the analog
    value to bar
122
123 // Frequency //
124 Htime = pulseIn(freqInput, HIGH); // Measure high time of the signal
125 Ltime = pulseIn(freqInput, LOW); // Measure low time of the signal
126 Ttime = (Htime + Ltime); // The period of the signal can be
    computed as the sum of high and low time

```

```

127 frequency = 1000000 / Ttime;           // The frequency is computed from the
    period
128 rpm = frequency * 60;                 // Frequency is converted to rpm
129
130 // Temperature //
131 tempValue = analogRead(tempInput);     // Reads value from input pin
132 x = (tempValue/1023) * 5; // Conversion equation to convert the analog value to
    V
133 temperature = a*x^3 + b*x^2 + c*x + d; // Cubic equation of the relation V
    - C
134
135 // Printing //
136 Serial.print("press: ");
137 Serial.print(pressureValue, 2);
138 Serial.print("bar  ");
139 Serial.print("freq: ");
140 Serial.print(frequency, 2);
141 Serial.print("Hz  ");
142 Serial.print("Temperature: ");
143 Serial.print(temperature, 2);
144 Serial.print(" C  ");
145     Serial.print("rpm: ");
146 Serial.print(rpm, 2);
147 Serial.print(" rpm");
148 Serial.print('\n');
149
150 lcd1.setCursor(0, 2);
151 lcd1.print("P:");
152 lcd1.print(pressureValue, 2);
153 lcd1.print("bar ");
154 lcd1.setCursor(9,2);
155 lcd1.print("T:");
156 lcd1.print(temperature, 1);
157 lcd1.print("C");
158 lcd1.setCursor(0, 3);
159 lcd1.print(rpm);
160 lcd1.print(" rpm");
161 delay(sensorreadDelay); //delay in milliseconds between read values
162 }

```

Listing F.6: All data processing required united in a single code.

Appendix G

Orifice plate construction and assembly

The first step in the construction of the system is the construction of the orifice plate and the testing of its functionality. In order to test and assemble the orifice plate configuration, the first step is to get build the sized orifice plates according to the system requirements, those that have been defined along the report. For the construction of the plates itself, to obtain them quickly and economically while complying with the necessary requirements and strength, they have been manufactured using 3D printing following the relevant drawings for each of the plates. This printing has been done by *UPC FabLab Terrassa*, where some 3D printers are available. The results obtained are the shown in figure G.1



Figure G.1: 3D printed orifice plates

To test the correct functioning of the plates, it has been decided, instead of feeding the turbine of the turbocompressor so that the compressor aspirates air, to feed the orifice plate directly. For this purpose, the outlet of the blower is connected directly to the inlet of the orifice plate pipe. Thus, a certain amount of air can be blown over the orifice plate and the differential pressure generated can be measured, and consequently the mass airflow rate passing through can be calculated. In addition, this measurement of the outgoing mass

airflow rate from the blower will serve another purpose. The blower does not indicate the flow rate or velocity of the outgoing air, only its power can be adjusted. Therefore, by directly concentrating the blower output to the orifice plate, not only the correct functioning of the plate itself will be tested, but also the relationship between power and airflow supplied by the blower will be characterised. The assembly made is shown in figures G.2 and G.3.

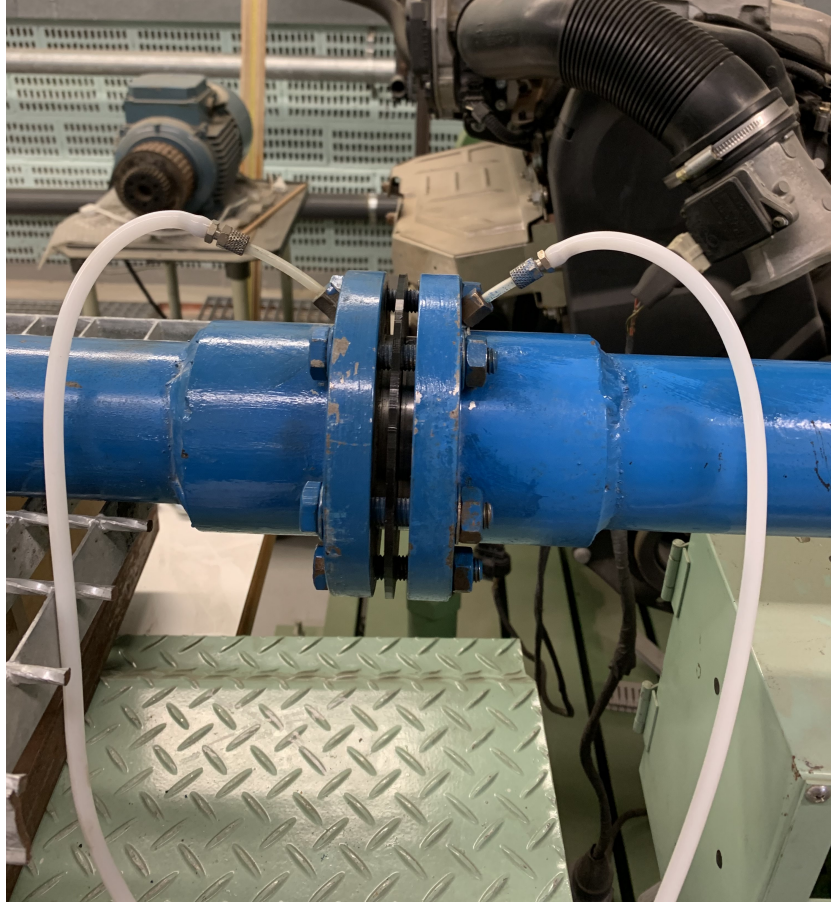


Figure G.2: Orifice plate at the pipe with pressures tapings



Figure G.3: Orifice plate testing assembly

There are two important points to be made regarding the assembly made.

1. To connect the orifice plate tube inlet to the blower outlet directly, it was necessary to increase the support height, as the table was low. Hence the present structure under the tube of the orifice plate. However, safety aspects were taken into account and the tube is properly fixed by means of various fastening elements. In addition to the fact that as this is not the final configuration in which the tube will be in the system, but it is only to do the testing, it has been decided not to make a definitive fastening, since at the final configuration neither the blower is connected to the orifice plate, nor the tube of the plate is in that position, but in the wall.
2. The differential pressure measuring device is different from the one presented in the report. This is due to the fact that the differential pressure measuring device was not available at the time of the test. The measuring instrument is a *Testo 510i*, which is used to measure differential pressures. This element, although available, was not chosen as an option as it does not have sufficient measuring range to measure the full range of differential pressures expected to be generated.

After running tests at different blower power settings, the results obtained are as shown in table G.1 and represented in figures G.4, G.5 and G.6.

Blower power [%]	ΔP [Pa]	\dot{q}_m [kg/s]
2	5	0,0015
11,5	260	0,0102
20	790	0,0176
30	1840	0,0267
40	3210	0,035
50	4910	0,0431

Table G.1: Results obtained from testing

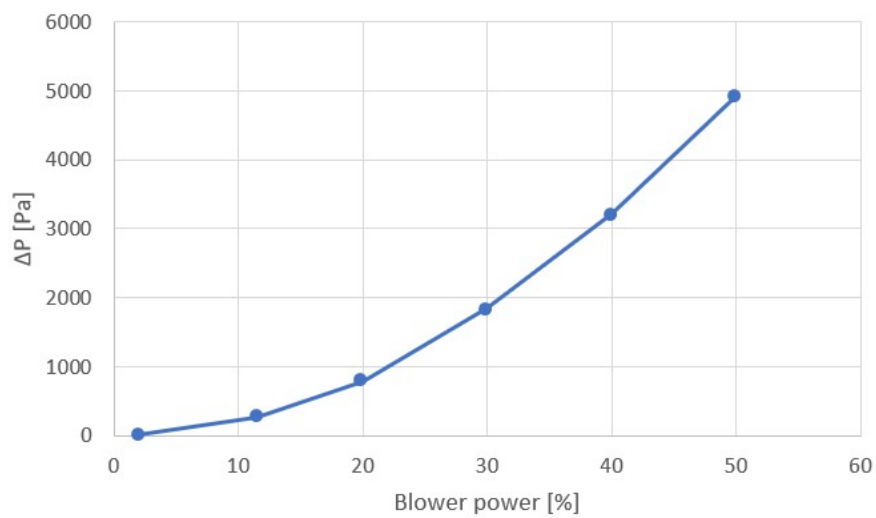


Figure G.4: ΔP as a function of the blower power [%]

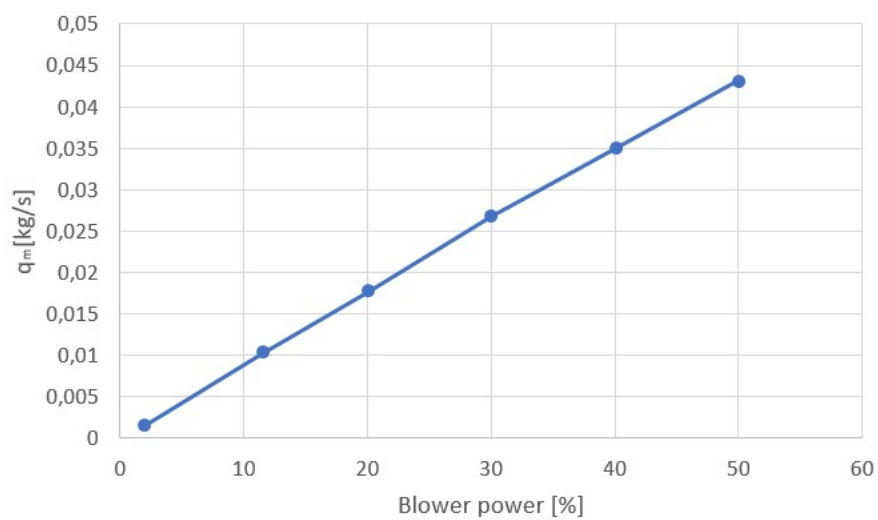
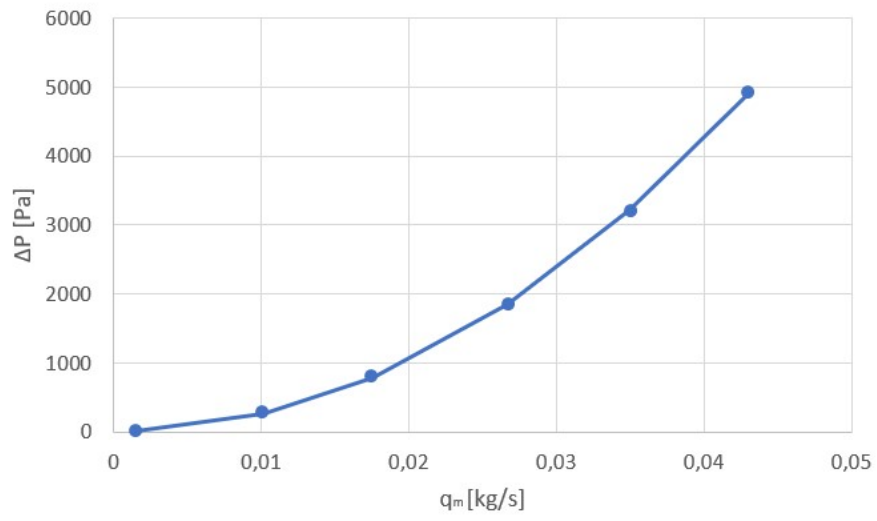


Figure G.5: \dot{q}_m as a function of the blower power [%]

Figure G.6: ΔP as a function of \dot{q}_m

Bibliography

1. SENSORS, Guilcor. *VALORES DE RESISTENCIA EN OHMIOS DE 0 ° C A + 400 ° C*. Available also from: <https://www.guilcor.es/content/31-table-de-conversion-des-sondes-pt100>.