

```

1 classdef SiPMulator_Initialization < matlab.apps.AppBase
2
3 % Properties that correspond to app components
4 properties (Access = public)
5     SERIALCOMMUNICATION          matlab.ui.Figure
6     Image3                       matlab.ui.control.Image
7     Panel_3                      matlab.ui.container.Panel
8     AboutButton_2                matlab.ui.control.Button
9     SiPMulatorInitializationPanel matlab.ui.container.Panel
10    DetectButton                 matlab.ui.control.Button
11    RButton                      matlab.ui.control.Button
12    BaudrateEditField            matlab.ui.control.NumericEditField
13    BaudrateEditFieldLabel       matlab.ui.control.Label
14    SiPMulatornotdetectedLamp   matlab.ui.control.Lamp
15    SiPMulatordetectedLabel     matlab.ui.control.Label
16    StateLabel                  matlab.ui.control.Label
17    ConnectButton               matlab.ui.control.Button
18    SerialPortDropDown          matlab.ui.control.DropDown
19    SerialPortDropDownLabel     matlab.ui.control.Label
20    Image2                      matlab.ui.control.Image
21    Image                        matlab.ui.control.Image
22 end
23
24
25 properties (Access = private)
26     Property
27     sp = []
28 end
29
30 methods (Access = private)
31     function shareSerialWithNextApp(app)
32         if isempty(app.sp) || ~isValid(app.sp)
33             error('Port not open');
34         end
35         setappdata(0,'SiPSession_sp', app.sp);
36         setappdata(0,'SiPSession_info',
37                     struct('port',char(app.SerialPortDropDown.Value)));
38     end
39 end
40
41 % Callbacks that handle component events
42 methods (Access = private)
43
44     % Value changed function: SerialPortDropDown
45     function SerialPortDropDownValueChanged(app, event)
46
47         val = app.SerialPortDropDown.Value;
48         disp("Selected port:" + val);
49     end
50
51     % Drop down opening function: SerialPortDropDown
52     function SerialPortDropDownOpening(app, event)
53 p = serialportlist("available");
54 prev = app.SerialPortDropDown.Value;
55
56 if isempty(p)
57     app.SerialPortDropDown.Items = {'<no ports>};
58     app.SerialPortDropDown.Value = '<no ports>';
59     return
60 end
61
62 items = cellstr(p(:)');
63 app.SerialPortDropDown.Items = items;
64
65
66     % Button pushed function: RButton
67     function RButtonPushed(app, event)
68         p = serialportlist("available");
69
70 if isempty(p)
71     app.SerialPortDropDown.Items = {'<no ports>'};

```

```

73         app.SerialPortDropDown.Value = '<no ports>';
74     else
75         app.SerialPortDropDown.Items = cellstr(p(:)');
76         app.SerialPortDropDown.Value = app.SerialPortDropDown.Items{1};
77     end
78
79     disp("Port list updated");
80
81     end
82
83     % Button pushed function: ConnectButton
84     function ConnectButtonPushed(app, event)
85
86     if strcmp(app.ConnectButton.Text, 'Connect')
87         port = char(app.SerialPortDropDown.Value);
88         if isempty(serialportlist)
89             errordlg('No ports available', 'Info');
90             return;
91         end
92         br = 112500;
93
94         try
95             app.sp = serialport(port, br, "Timeout", 1);
96             configureTerminator(app.sp, "LF");
97             disp(['Connected to the port ', port, ' @ ', num2str(br), ' baudios']);
98             app.ConnectButton.Text = 'Disconnect';
99
100        catch ME
101            uialert(app.SERIALCOMMUNICATION, ME.message, 'Error connecting',
102                    'Icon', 'none');
103        end
104    else
105        try
106            if ~isempty(app.sp) && isvalid(app.sp)
107                configureCallback(app.sp, "off");
108                flush(app.sp);
109                clear app.sp;
110                disp('Port closed successfully');
111            end
112        catch
113        end
114        app.ConnectButton.Text = 'Connect';
115    end
116    try
117        if isappdata(0,'SiPSession_sp'), rmappdata(0,'SiPSession_sp'); end
118        if isappdata(0,'SiPSession_info'), rmappdata(0,'SiPSession_info'); end
119    end
120
121
122     % Button pushed function: DetectButton
123     function DetectButtonPushed(app, event)
124     if ~isprop(app,'sp') || isempty(app.sp) || ~isValid(app.sp)
125         d = warndlg('Open the port first.', 'Warning', 'modal');
126         d.Position(3:4) = [260 130];
127         movegui(d, 'center');
128         return;
129     end
130     try
131         flush(app.sp);
132         write(app.sp, uint8(170), "uint8");
133         pause(0.05);
134
135         if app.sp.NumBytesAvailable >= 1
136             r = read(app.sp, 1, "uint8");
137             disp("Resp: " + num2str(r));
138
139             app.SiPMulatordetectedLabel.Text = 'SiPMulator detected';
140             app.SiPMulatornotdetectedLamp.Color = [0 1 0];
141             drawnow; pause(1);
142
143             % Ventana "INITIALIZING SIPMULATOR..."
144             w = 420; h = 140;

```

```

145 scr = get(groot,'ScreenSize');
146 x = (scr(3)-w)/2; y = (scr(4)-h)/2;
147 splash = uifigure('Name','Initializing', ...
148             'Position',[x y w h], ...
149             'Resize','off', 'WindowState','normal');
150 uilabel(splash,'Text','INITIALIZING SIPMULATOR...', ...
151         'FontWeight','bold','FontSize',18, ...
152         'HorizontalAlignment','center', ...
153         'Position',[20 (h/2-20) w-40 40]);
154 drawnow; pause(3);
155 if isvalid(splash), close(splash); end
156
157 app.shareSerialWithNextApp();
158 disp("INIT setappdata: " + string(isappdata(0,'SiPSession_sp')));
159
160 if exist('SiPMulator','class')
161     SiPMulator;
162 else
163     uialert(app.SERIALCOMMUNICATION,'"I can't find SiPMulator.mlapp','File
164         not found');
165     return;
166 end
167 delete(app);
168 else
169     app.SiPMulatordetectedLabel.Text = 'SiPMulator not detected';
170     app.SiPMulatornotdetectedLamp.Color = [1 0 0];
171 end
172 catch ME
173     app.SiPMulatordetectedLabel.Text = 'Error';
174     app.SiPMulatornotdetectedLamp.Color = [1 0 0];
175     uialert(app.SERIALCOMMUNICATION, ME.message, 'Serial error', 'Icon','warning');
176 end
177 end
178
179 % Button pushed function: AboutButton_2
180 function AboutButton_2Pushed(app, event)
181
182 f = uifigure('Name','About SiPMulator','Position',[600 280 520 480], ...
183             'Color','white','Resize','off');
184 movegui(f,'center');
185 axS = uiaxes(f,'Position',[20 415 190 55]);
186 axD = uiaxes(f,'Position',[520-190-20 415 150 55]);
187 for ax = [axS axD]
188     ax.XColor = 'none';
189     ax.YColor = 'none';
190     ax.Toolbar.Visible = 'off';
191     ax.Interactions = [];
192     ax.Color = 'white';
193 end
194 imshow('5cm_SiPM.png','Parent',axS);
195 imshow('4cm-DUNE.png','Parent',axD);
196
197
198 desc = ['SiPMulator is a prototype that reproduces real SiPM sensor pulses '
199 ...
200             'from ProtoDUNE II, enabling the analysis and validation of the ' ...
201             'DAPHNE electronic system through controlled analog signals.'];
202 uilabel(f,'Text',desc,'Position',[25 375 470 35], ...
203         'FontSize',10,'HorizontalAlignment','left',...
204         'WordWrap','on','BackgroundColor','white');
205
206 axP = uiaxes(f,'Position',[80 180 360 190]);
207 axP.XColor = 'none';
208 axP.YColor = 'none';
209 axP.Toolbar.Visible = 'off';
210 axP.Interactions = [];
211 axP.Color = 'white';
212 imshow('cajaaaa.png','Parent',axP);
213
214 info = sprintf(['App developed by Slyan S. Pinzón and Javier F. Castaño\n' ...
215                 'Semillero de investigación SITAD\n' ...
216                 'Universidad Antonio Nariño\n' ...]

```

```

216             'Villavicencio Campus\n' ...
217             '2025']);
218     uilabel(f,'Text',info,'Position',[25 145 470 60], ...
219             'FontSize',10,'HorizontalAlignment','center', ...
220             'WordWrap','on','BackgroundColor','white');
221
222
223     axU = uiaxes(f,'Position',[20 20 300 95]);
224     axU.XColor = 'none';
225     axU.YColor = 'none';
226     axU.Toolbar.Visible = 'off';
227     axU.Interactions = [];
228     axU.Color = 'white';
229     imshow('5cm-UAN.jpeg','Parent',axU);
230
231 end
232
233 % Size changed function: SiPMulatorInitializationPanel
234 function SiPMulatorInitializationPanelSizeChanged(app, event)
235     position = app.SiPMulatorInitializationPanel.Position;
236
237 end
238
239
240 % Component initialization
241 methods (Access = private)
242
243     % Create UIFigure and components
244     function createComponents(app)
245
246         % Get the file path for locating images
247         pathToMLAPP = fileparts(mfilename('fullpath'));
248
249         % Create SERIALCOMMUNICATION and hide until all components are created
250         app.SERIALCOMMUNICATION = uifigure('Visible', 'off');
251         app.SERIALCOMMUNICATION.AutoResizeChildren = 'off';
252         app.SERIALCOMMUNICATION.Color = [1 1 1];
253         app.SERIALCOMMUNICATION.Position = [100 100 265 315];
254         app.SERIALCOMMUNICATION.Name = 'SiPMulator initialization';
255         app.SERIALCOMMUNICATION.Resize = 'off';
256
257         % Create Panel_3
258         app.Panel_3 = uipanel(app.SERIALCOMMUNICATION);
259         app.Panel_3.AutoResizeChildren = 'off';
260         app.Panel_3.ForegroundColor = [1 1 1];
261         app.Panel_3.TitlePosition = 'centertop';
262         app.Panel_3.BackgroundColor = [1 1 1];
263         app.Panel_3.FontWeight = 'bold';
264         app.Panel_3.Position = [9 9 248 299];
265
266         % Create Image
267         app.Image = uiimage(app.Panel_3);
268         app.Image.Position = [9 221 100 100];
269         app.Image.ImageSource = fullfile(pathToMLAPP, '5cm_SiPM.png');
270
271         % Create Image2
272         app.Image2 = uiimage(app.Panel_3);
273         app.Image2.Position = [168 237 70 70];
274         app.Image2.ImageSource = fullfile(pathToMLAPP, '4cm-DUNE.png');
275
276         % Create SiPMulatorInitializationPanel
277         app.SiPMulatorInitializationPanel = uipanel(app.Panel_3);
278         app.SiPMulatorInitializationPanel.AutoResizeChildren = 'off';
279         app.SiPMulatorInitializationPanel.TitlePosition = 'centertop';
280         app.SiPMulatorInitializationPanel.Title = 'SiPMulator initialization';
281         app.SiPMulatorInitializationPanel.BackgroundColor = [1 1 1];
282         app.SiPMulatorInitializationPanel.SizeChangedFcn = createCallbackFcn(app,
283             @SiPMulatorInitializationPanelSizeChanged, true);
284         app.SiPMulatorInitializationPanel.FontName = 'Times New Roman';
285         app.SiPMulatorInitializationPanel.FontWeight = 'bold';
286         app.SiPMulatorInitializationPanel.FontSize = 13;
287         app.SiPMulatorInitializationPanel.Position = [9 55 229 202];

```

```

288 % Create SerialPortDropDownLabel
289 app.SerialPortDropDownLabel = uilabel(app.SiPMulatorInitializationPanel);
290 app.SerialPortDropDownLabel.Position = [32 130 61 22];
291 app.SerialPortDropDownLabel.Text = 'Serial Port';
292
293 % Create SerialPortDropDown
294 app.SerialPortDropDown = uidropdown(app.SiPMulatorInitializationPanel);
295 app.SerialPortDropDown.Items = {};
296 app.SerialPortDropDown.DropDownOpeningFcn = createCallbackFcn(app,
297 @SerialPortDropDownOpening, true);
298 app.SerialPortDropDown.ValueChangedFcn = createCallbackFcn(app,
299 @SerialPortDropDownValueChanged, true);
300 app.SerialPortDropDown.Position = [99 130 69 22];
301 app.SerialPortDropDown.Value = {};
302
303 % Create ConnectButton
304 app.ConnectButton = uibutton(app.SiPMulatorInitializationPanel, 'push');
305 app.ConnectButton.ButtonPushedFcn = createCallbackFcn(app,
306 @ConnectButtonPushed, true);
307 app.ConnectButton.Position = [26 47 89 23];
308 app.ConnectButton.Text = 'Connect';
309
310 % Create StateLabel
311 app.StateLabel = uilabel(app.SiPMulatorInitializationPanel);
312 app.StateLabel.Position = [26 12 36 22];
313 app.StateLabel.Text = 'State:';
314
315 % Create SiPMulatordetectedLabel
316 app.SiPMulatordetectedLabel = uilabel(app.SiPMulatorInitializationPanel);
317 app.SiPMulatordetectedLabel.HorizontalAlignment = 'right';
318 app.SiPMulatordetectedLabel.Position = [74 12 133 22];
319 app.SiPMulatordetectedLabel.Text = 'SiPMulator not detected';
320
321 % Create SiPMulatornotdetectedLamp
322 app.SiPMulatornotdetectedLamp = uilamp(app.SiPMulatorInitializationPanel);
323 app.SiPMulatornotdetectedLamp.Position = [62 16 13 13];
324 app.SiPMulatornotdetectedLamp.Color = [1 0 0];
325
326 % Create BaudrateEditFieldLabel
327 app.BaudrateEditFieldLabel = uilabel(app.SiPMulatorInitializationPanel);
328 app.BaudrateEditFieldLabel.Position = [32 95 54 22];
329 app.BaudrateEditFieldLabel.Text = 'Baudrate';
330
331 % Create BaudrateEditField
332 app.BaudrateEditField = uieditfield(app.SiPMulatorInitializationPanel,
333 'numeric');
334 app.BaudrateEditField.ValueDisplayFormat = '%.0f';
335 app.BaudrateEditField.Editable = 'off';
336 app.BaudrateEditField.Position = [99 95 69 22];
337 app.BaudrateEditField.Value = 112500;
338
339 % Create RButton
340 app.RButton = uibutton(app.SiPMulatorInitializationPanel, 'push');
341 app.RButton.ButtonPushedFcn = createCallbackFcn(app, @RButtonPushed,
342 true);
343 app.RButton.FontSize = 10;
344 app.RButton.Position = [170 132 24 19];
345 app.RButton.Text = 'R';
346
347 % Create DetectButton
348 app.DetectButton = uibutton(app.SiPMulatorInitializationPanel, 'push');
349 app.DetectButton.ButtonPushedFcn = createCallbackFcn(app,
350 @DetectButtonPushed, true);
351 app.DetectButton.Position = [122 47 88 23];
352 app.DetectButton.Text = 'Detect';
353
354 % Create AboutButton_2
355 app.AboutButton_2 = uibutton(app.Panel_3, 'push');
356 app.AboutButton_2.ButtonPushedFcn = createCallbackFcn(app,
357 @AboutButton_2Pushed, true);
358 app.AboutButton_2.FontSize = 10;
359 app.AboutButton_2.Position = [158 8 36 18];
360 app.AboutButton_2.Text = 'About';

```

```
354
355     % Create Image3
356     app.Image3 = uiimage(app.SERIALCOMMUNICATION);
357     app.Image3.Position = [19 -32 114 136];
358     app.Image3.ImageSource = fullfile(pathToMLAPP, '5cm-UAN.jpeg');
359
360     % Show the figure after all components are created
361     app.SERIALCOMMUNICATION.Visible = 'on';
362 end
363
364
365 % App creation and deletion
366 methods (Access = public)
367
368     % Construct app
369     function app = SiPMulator_Initialization
370
371         % Create UIFigure and components
372         createComponents(app)
373
374         % Register the app with App Designer
375         registerApp(app, app.SERIALCOMMUNICATION)
376
377         if nargout == 0
378             clear app
379         end
380     end
381
382     % Code that executes before app deletion
383     function delete(app)
384
385         % Delete UIFigure when app is deleted
386         delete(app.SERIALCOMMUNICATION)
387     end
388 end
389 end
```

```

1 classdef SiPMulator < matlab.apps.AppBase
2
3 % Properties that correspond to app components
4 properties (Access = public)
5     SiPMulatorAppv10UIFigure      matlab.ui.Figure
6     Panel_3                      matlab.ui.container.Panel
7     Image3                       matlab.ui.control.Image
8     FINISHButton                 matlab.ui.control.Button
9     Image2                       matlab.ui.control.Image
10    Image                         matlab.ui.control.Image
11    Panel_5                      matlab.ui.container.Panel
12    Panel_6                      matlab.ui.container.Panel
13    UIAxes                        matlab.ui.control.UIAxes
14    ControlpreloadpulsesPanel    matlab.ui.container.Panel
15    ChangepulseButton            matlab.ui.control.Button
16    FolderNameEditField          matlab.ui.control.EditField
17    FolderNameEditFieldLabel     matlab.ui.control.Label
18    SetfolderButton              matlab.ui.control.Button
19    SIGNALButtonGroup            matlab.ui.container.ButtonGroup
20    Panel_7                      matlab.ui.container.Panel
21    SiPMulatorconnectedLamp_2    matlab.ui.control.Lamp
22    SiPMulatorconnectedLamp_2Label matlab.ui.control.Label
23    ButtonGroup                  matlab.ui.container.ButtonGroup
24    UserPulseButton_2            matlab.ui.control.RadioButton
25    PreloadPulseButton_2         matlab.ui.control.RadioButton
26    ApplyButton                  matlab.ui.control.Button
27    ModeDropDown                matlab.ui.control.DropDown
28    ModeDropDownLabel           matlab.ui.control.Label
29    ControluserpulsesPanel      matlab.ui.container.Panel
30    SendButton_2                 matlab.ui.control.Button
31    FileNameEditField           matlab.ui.control.EditField
32    FileNameEditFieldLabel      matlab.ui.control.Label
33    ImportfileButton             matlab.ui.control.Button
34 end
35
36
37 properties (Access = public)
38     sp = []
39     ImportedData
40     LastFolder char = pwd
41     salida
42 end
43
44 properties (Access = private)
45     modeByte uint8 = uint8(hex2dec('BB'));
46     pendingSend logical = false;
47     HasCSV      logical = false;
48     HasPlot     logical = false;
49     lastAppliedMode uint8 = [];
50     lastSentMode  uint8 = [];
51     PreloadFolder
52     CheckTimer
53     LostConnection logical = false;
54 end
55
56 methods (Access = private)
57     function ok = hasSerial(app)
58         ok = ~isempty(app.sp) && isvalid(app.sp);
59     end
60
61     function sendModeFrame(app)
62         flush(app.sp);
63         write(app.sp, uint8(0xFF), "uint8");
64         pause(0.02);
65         write(app.sp, app.modeByte, "uint8");
66         disp("Send envío: 0xFF y 0x" + upper(dec2hex(double(app.modeByte), 2)));
67         app.lastSentMode = app.modeByte;
68     end
69
70 function onSerialError(app, evt)
71     setDisconnectedUI(app);
72 end
73
```

```

74 function checkSiPM(app)
75     if ~app.hasSerial()
76         setDisconnectedUI(app);
77         return;
78     end
79
80     try
81         flush(app.sp);
82         write(app.sp, uint8(170), "uint8");
83         pause(0.05);
84
85         if app.sp.NumBytesAvailable >= 1
86             read(app.sp, 1, "uint8");
87             app.SiPMulatorconnectedLamp_2Label.Text = 'SiPMulator connected';
88             app.SiPMulatorconnectedLamp_2.Color = [0 1 0];
89             drawnow;
90         else
91             setDisconnectedUI(app);
92         end
93
94     catch
95         setDisconnectedUI(app);
96     end
97 end
98
99
100 function setDisconnectedUI(app)
101     app.SiPMulatorconnectedLamp_2Label.Text = 'SiPMulator not connected';
102     app.SiPMulatorconnectedLamp_2.Color = [1 0 0];
103
104     app.ModeDropDown.Enable      = 'off';
105     app.ApplyButton.Enable      = 'off';
106     app.ButtonGroup.Enable      = 'off';
107     app.SetfolderButton.Enable = 'off';
108     app.FolderNameEditField.Enable = 'off';
109     app.ChangepulseButton.Enable = 'off';
110     app.ImportfileButton.Enable = 'off';
111     app.FileNameEditField.Enable = 'off';
112     app.SendButton_2.Enable    = 'off';
113
114     try
115         if ~isempty(app.CheckTimer) && isvalid(app.CheckTimer)
116             stop(app.CheckTimer);
117         end
118     end
119     if ~app.LostConnection
120         app.LostConnection = true;
121         uialert(app.SiPMulatorAppv10UIFigure, ...
122             'Serial connection lost. Reconnect SiPMulator.', ...
123             'Connection lost', 'Icon','warning');
124     end
125     drawnow;
126     end
127     end
128
129     % Callbacks that handle component events
130     methods (Access = private)
131
132         % Code that executes after component creation
133         function startupFcn(app)
134
135             disp("SIP start: isappdata=" + string(isappdata(0,'SiPSession_sp')));
136             app.sp = getappdata(0,'SiPSession_sp');
137             disp("SIP start: class(sp)=" + string(class(app.sp)));
138
139             if isappdata(0,'SiPSession_sp')
140                 app.sp = getappdata(0,'SiPSession_sp');
141                 if ~isvalid(app.sp)
142                     uialert(app.SiPMulatorAppv10UIFigure,'Sesión inválida','Warning');
143                     return;
144                 end
145
146             try

```

```

147         app.SiPMulatorconnectedLamp_2.Color = [0 1 0];
148         app.SiPMulatorconnectedLamp_2Label.Text = 'SiPMulator connected';
149     end
150
151     app.sp.ErrorOccurredFcn = @(src, evt) onSerialError(app, evt);
152
153     app.CheckTimer = timer( ...
154         'ExecutionMode','fixedSpacing', ...
155         'Period', 1, ...
156         'TimerFcn', @(~,~) checkSiPM(app));
157     start(app.CheckTimer);
158
159 else
160     uialert(app.SiPMulatorAppv10UIFigure,'Open from
161     Initialization.', 'Warning', 'Icon', 'warning');
162 end
163
164 ModeDropDownValueChanged(app, []);
165
166 app.HasCSV = false;
167 app.HasPlot = false;
168 app.pendingSend = false;
169 app.ApplyButton.Enable = 'off';
170 app.lastAppliedMode = [];
171 app.lastSentMode = [];
172 app.SetfolderButton.Enable = 'on';
173 app.FolderNameEditField.Enable = 'on';
174 app.ChangepulseButton.Enable = 'on';
175 app.ImportfileButton.Enable = 'off';
176 app.FileNameEditField.Enable = 'off';
177 app.SendButton_2.Enable = 'off';
178     end
179
180     % Button pushed function: ImportfileButton
181     function ImportfileButtonPushed(app, event)
182 [f,p] = uigetfile({'*.*','All files'}, ...
183                     'Select a file', app.LastFolder);
184 if isequal(f,0)
185     return;
186 end
187 app.LastFolder = p;
188
189 fullpath = fullfile(p,f);
190 app.FileNameEditField.Value = f;
191
192 app.salida = fullfile(p,"salida.csv");
193 sipm_pulse_to_32(fullpath, app.salida);
194
195 app.ImportedData = readmatrix(fullpath);
196
197 data = app.ImportedData;
198 data = data(:);
199 fs = 1.875e6;
200 Ts = 1/fs;
201 N = numel(data);
202 t_us = (0:N-1) * Ts * 1e6;
203
204 plot(app.UIAxes, t_us, data, '-o');
205 app.UIAxes.XLabel.String = 'Time (us)';
206 app.UIAxes.YLabel.String = 'Dac value 8 bits';
207 title(app.UIAxes, 'Discretized signal 1.875 Msps');
208
209 app.HasCSV = true;
210 app.HasPlot = true;
211 app.pendingSend = false;
212 app.lastAppliedMode = [];
213 app.ChangepulseButton.Text = 'Send';
214 app.ChangepulseButton.Enable = 'off';
215 app.ApplyButton.Enable = 'off';
216
217 try
218     figure(app.SiPMulatorAppv10UIFigure);

```

```

219     end
220     end
221
222     % Value changed function: ModeDropDown
223     function ModeDropDownValueChanged(app, event)
224         val = app.ModeDropDown.Value;
225         switch val
226             case 'Periodic pulse signal', app.modeByte = uint8(hex2dec('DD'));
227             case 'Pulse signal with noise', app.modeByte = uint8(hex2dec('CC'));
228             case 'Noise Signal', app.modeByte = uint8(hex2dec('BB'));
229             otherwise, app.modeByte = uint8(hex2dec('BB'));
230     end
231
232
233     if app.pendingSend && app.HasCSV && app.HasPlot && app.hasSerial()
234         if isempty(app.lastAppliedMode) || app.modeByte ~= app.lastAppliedMode
235             app.ApplyButton.Enable = 'on';
236         else
237             app.ApplyButton.Enable = 'off';
238         end
239     end
240
241
242     % Button pushed function: ApplyButton
243     function ApplyButtonPushed(app, event)
244         if ~app.pendingSend
245             uialert(app.SiPMulatorAppv10UIFigure, 'Press Send
first.', 'Error', 'Icon', 'warning'); return;
246         end
247         if ~app.hasSerial()
248             uialert(app.SiPMulatorAppv10UIFigure, 'Serial not
available.', 'Warning', 'Icon', 'warning'); return;
249         end
250         try
251             write(app.sp, app.modeByte, "uint8");
252             disp("Apply send mode: 0x" + upper(dec2hex(double(app.modeByte), 2)));
253
254             app.lastAppliedMode = app.modeByte;
255             app.ApplyButton.Enable = 'off';
256         catch ME
257             uialert(app.SiPMulatorAppv10UIFigure, "Failed to apply: " + ME.message,
'Error', 'Icon', 'error');
258         end
259     end
260
261
262     % Button pushed function: FINISHButton
263     function FINISHButtonPushed(app, event)
264         try
265             if app.hasSerial()
266                 flush(app.sp);
267                 write(app.sp, uint8(hex2dec('BB')), "uint8");
268             else
269                 disp("Default mode not restored: serial not available.");
270             end
271             catch ME
272                 disp("Could not send default mode:" + ME.message);
273         end
274         try
275             if ~isempty(app.sp) && isvalid(app.sp)
276                 disp("Closing port: " + app.sp.Port);
277                 delete(app.sp);
278                 disp("Port released successfully.");
279             else
280                 disp("Port already invalid or empty, nothing to close.");
281             end
282             catch ME
283                 disp("Error closing serial port:" + ME.message);
284         end
285         app.sp = [];
286
287         try
288             if ~isempty(app.CheckTimer) && isvalid(app.CheckTimer)

```

```

289         stop(app.CheckTimer);
290         delete(app.CheckTimer);
291     end
292 end
293
294 end
295
296 try
297     if isappdata(0,'SiPSession_sp'),    rmappdata(0,'SiPSession_sp');    end
298     if isappdata(0,'SiPSession_info'), rmappdata(0,'SiPSession_info'); end
299     if isappdata(0,'SiPM_open'),       rmappdata(0,'SiPM_open');       end
300 catch
301 end
302 try
303     delete(app);
304 catch ME
305     disp("Error closing the app: " + ME.message);
306 end
307
308 end
309
310 % Button pushed function: SetfolderButton
311 function SetfolderButtonPushed(app, event)
312 folderPath = uigetdir(pwd, 'Select folder with preload pulses');
313 if isequal(folderPath, 0)
314     return;
315 end
316
317 app.PreloadFolder = folderPath;
318
319 [~, folderName] = fileparts(folderPath);
320 app.FolderNameEditField.Value = folderName;
321
322
323 try, figure(app.SiPMulatorAppv10UIFigure); end
324 end
325
326 % Button pushed function: ChangepulseButton
327 function ChangepulseButtonPushed(app, event)
328
329 folderPath = app.PreloadFolder;
330
331 if isempty(folderPath)
332     uialert(app.SiPMulatorAppv10UIFigure, ...
333         'Primero seleccione la carpeta con Set folder.', 'Error');
334     return;
335 end
336
337 persistent idx;
338 if isempty(idx)
339     idx = 1;
340 end
341
342 archivos = { ...
343     'pulse_rom_1.csv', ...
344     'pulse_rom_2.csv', ...
345     'pulse_rom_3.csv', ...
346     'pulse_rom_4.csv' };
347
348 csv_fullpath = fullfile(folderPath, archivos{idx});
349
350 if ~isfile(csv_fullpath)
351     uialert(app.SiPMulatorAppv10UIFigure, msg, 'Error');
352     return;
353 end
354
355 data = readmatrix(csv_fullpath);
356
357 fs    = 1.875e6;
358 Ts    = 1/fs;
359 N     = numel(data);
360 t_us = (0:N-1) * Ts * 1e6;
361

```

```

362 plot(app.UIAxes, t_us, data, '-o');
363 app.UIAxes.XLabel.String = 'Time (μs)';
364 app.UIAxes.YLabel.String = 'Dac value 8 bits';
365 title(app.UIAxes, 'Discretized signal 1.875 Msps');
366
367 idx = idx + 1;
368 if idx > numel(archivos)
369     idx = 1;
370 end
371
372
373 if ~app.hasSerial()
374     uialert(app.SiPMulatorAppv10UIFigure, 'No file uploaded.', 'Error');
375     return;
376 end
377
378 flush(app.sp);
379 write(app.sp, uint8(0xFF), "uint8");
380 pause(0.05);
381 write(app.sp, uint8(0xDD), "uint8");
382 end
383
384 % Selection changed function: ButtonGroup
385 function ButtonGroupSelectionChanged(app, event)
386 selectedObj = app.ButtonGroup.SelectedObject;
387
388 if selectedObj == app.PreloadPulseButton_2
389     app.SetfolderButton.Enable      = 'on';
390     app.FolderNameEditField.Enable = 'on';
391     app.ChangepulseButton.Enable  = 'on';
392
393     app.ImportfileButton.Enable    = 'off';
394     app.FileNameEditField.Enable  = 'off';
395     app.SendButton_2.Enable       = 'off';
396
397 elseif selectedObj == app.UserPulseButton_2
398
399     app.SetfolderButton.Enable      = 'off';
400     app.FolderNameEditField.Enable = 'off';
401     app.ChangepulseButton.Enable  = 'off';
402
403     app.ImportfileButton.Enable    = 'on';
404     app.FileNameEditField.Enable  = 'on';
405     app.SendButton_2.Enable       = 'on';
406 end
407 end
408
409 % Button pushed function: SendButton_2
410 function SendButton_2Pushed(app, event)
411 if isempty(app.ImportedData)
412     uialert(app.SiPMulatorAppv10UIFigure, ...
413         'First import a CSV file with 'Import file'.', ...
414         'File not loaded', 'Icon','warning');
415     return;
416 end
417
418 if ~app.hasSerial()
419     uialert(app.SiPMulatorAppv10UIFigure, ...
420         'Serial communication not available. Check the SiPMulator connection.', ...
421         ...
422         'SSerial not available', 'Icon','warning');
423     return;
424 end
425
try
    data = app.ImportedData;
    data = data(:);
    data = uint8(max(0, min(255, round(data))));

    flush(app.sp);
    write(app.sp, uint8(0), "uint8");
    pause(0.2);

```

```

434     for k = 1:numel(data)
435         write(app.sp, data(k), "uint8");
436         pause(0.2);
437     end
438
439     disp("SendButton_2: They were sent " + num2str(numel(data)) + "bytes after
440 0x00.");
441     app.pendingSend = true;
442
443     catch ME
444         uialert(app.SiPMulatorAppv10UIFigure, ...
445             "Error enviando datos por serial: " + ME.message, ...
446             'Error de envio', 'Icon','error');
447     end
448 end
449
450 % Component initialization
451 methods (Access = private)
452
453 % Create UIFigure and components
454 function createComponents(app)
455
456     % Get the file path for locating images
457     pathToMLAPP = fileparts(mfilename('fullpath'));
458
459     % Create SiPMulatorAppv10UIFigure and hide until all components are
460     % created
461     app.SiPMulatorAppv10UIFigure = uifigure('Visible', 'off');
462     app.SiPMulatorAppv10UIFigure.AutoScaleChildren = 'off';
463     app.SiPMulatorAppv10UIFigure.Color = [1 1 1];
464     app.SiPMulatorAppv10UIFigure.Position = [100 100 469 647];
465     app.SiPMulatorAppv10UIFigure.Name = 'SiPMulator App v1.0';
466     app.SiPMulatorAppv10UIFigure.Resize = 'off';
467
468     % Create Panel_3
469     app.Panel_3 = uipanel(app.SiPMulatorAppv10UIFigure);
470     app.Panel_3.AutoScaleChildren = 'off';
471     app.Panel_3.BackgroundColor = [1 1 1];
472     app.Panel_3.Position = [12 16 444 619];
473
474     % Create Panel_5
475     app.Panel_5 = uipanel(app.Panel_3);
476     app.Panel_5.AutoScaleChildren = 'off';
477     app.Panel_5.BorderType = 'none';
478     app.Panel_5.TitlePosition = 'centertop';
479     app.Panel_5.BackgroundColor = [1 1 1];
480     app.Panel_5.FontName = 'Times New Roman';
481     app.Panel_5.FontWeight = 'bold';
482     app.Panel_5.FontSize = 18;
483     app.Panel_5.Position = [9 63 421 509];
484
485     % Create ControluserpulsesPanel
486     app.ControluserpulsesPanel = uipanel(app.Panel_5);
487     app.ControluserpulsesPanel.AutoScaleChildren = 'off';
488     app.ControluserpulsesPanel.Title = 'Control user pulses';
489     app.ControluserpulsesPanel.BackgroundColor = [0.9412 0.9412 0.9412];
490     app.ControluserpulsesPanel.FontWeight = 'bold';
491     app.ControluserpulsesPanel.Position = [13 268 409 69];
492
493     % Create ImportfileButton
494     app.ImportfileButton = uibutton(app.ControluserpulsesPanel, 'push');
495     app.ImportfileButton.ButtonPushedFcn = createCallbackFcn(app,
496     @ImportfileButtonPushed, true);
497     app.ImportfileButton.Position = [200 10 88 23];
498     app.ImportfileButton.Text = 'Import file';
499
500     % Create FileNameEditFieldLabel
501     app.FileNameEditFieldLabel = uilabel(app.ControluserpulsesPanel);
502     app.FileNameEditFieldLabel.HorizontalAlignment = 'right';
503     app.FileNameEditFieldLabel.Position = [10 10 60 22];

```

```

504 % Create FileNameEditField
505 app.FileNameEditField = uieditfield(app.ControluserpulsesPanel, 'text');
506 app.FileNameEditField.Editable = 'off';
507 app.FileNameEditField.Position = [71 10 113 22];
508
509 % Create SendButton_2
510 app.SendButton_2 = uibutton(app.ControluserpulsesPanel, 'push');
511 app.SendButton_2.ButtonPushedFcn = createCallbackFcn(app,
512 @SendButton_2Pushed, true);
513 app.SendButton_2.Position = [297 10 100 23];
514 app.SendButton_2.Text = 'Send';
515
516 % Create SIGNALButtonGroup
517 app.SIGNALButtonGroup = uibuttongroup(app.Panel_5);
518 app.SIGNALButtonGroup.AutoResizeChildren = 'off';
519 app.SIGNALButtonGroup.TitlePosition = 'centertop';
520 app.SIGNALButtonGroup.Title = 'SIGNAL';
521 app.SIGNALButtonGroup.FontWeight = 'bold';
522 app.SIGNALButtonGroup.Position = [12 420 410 90];
523
524 % Create ModeDropDownLabel
525 app.ModeDropDownLabel = uilabel(app.SIGNALButtonGroup);
526 app.ModeDropDownLabel.Position = [151 6 35 22];
527 app.ModeDropDownLabel.Text = 'Mode';
528
529 % Create ModeDropDown
530 app.ModeDropDown = uidropdown(app.SIGNALButtonGroup);
531 app.ModeDropDown.Items = {'Periodic pulse ', 'Pulse+seudorandom noise',
532 'Seudorandom noise'};
533 app.ModeDropDown.ValueChangedFcn = createCallbackFcn(app,
534 @ModeDropDownValueChanged, true);
535 app.ModeDropDown.Position = [184 6 129 22];
536 app.ModeDropDown.Value = 'Periodic pulse ';
537
538 % Create ApplyButton
539 app.ApplyButton = uibutton(app.SIGNALButtonGroup, 'push');
540 app.ApplyButton.ButtonPushedFcn = createCallbackFcn(app,
541 @ApplyButtonPushed, true);
542 app.ApplyButton.Position = [323 6 72 23];
543 app.ApplyButton.Text = 'Apply';
544
545 % Create ButtonGroup
546 app.ButtonGroup = uibuttongroup(app.SIGNALButtonGroup);
547 app.ButtonGroup.AutoResizeChildren = 'off';
548 app.ButtonGroup.SelectionChangedFcn = createCallbackFcn(app,
549 @ButtonGroupSelectionChanged, true);
550 app.ButtonGroup.ForegroundColor = [1 1 1];
551 app.ButtonGroup.BorderType = 'none';
552 app.ButtonGroup.Position = [5 18 123 49];
553
554 % Create PreloadPulseButton_2
555 app.PreloadPulseButton_2 = uiradiobutton(app.ButtonGroup);
556 app.PreloadPulseButton_2.Text = 'Preload Pulse';
557 app.PreloadPulseButton_2.Position = [11 24 97 22];
558 app.PreloadPulseButton_2.Value = true;
559
560 % Create UserPulseButton_2
561 app.UserPulseButton_2 = uiradiobutton(app.ButtonGroup);
562 app.UserPulseButton_2.Text = 'User Pulse';
563 app.UserPulseButton_2.Position = [11 2 81 22];
564
565 % Create Panel_7
566 app.Panel_7 = uipanel(app.SIGNALButtonGroup);
567 app.Panel_7.AutoResizeChildren = 'off';
568 app.Panel_7.BorderType = 'none';
569 app.Panel_7.Position = [171 33 227 34];
570
571 % Create SiPMulatorconnectedLamp_2Label
572 app.SiPMulatorconnectedLamp_2Label = uilabel(app.Panel_7);
573 app.SiPMulatorconnectedLamp_2Label.Position = [63 2 145 22];
574 app.SiPMulatorconnectedLamp_2Label.Text = 'SiPMulator connected';
575
576 % Create SiPMulatorconnectedLamp_2

```

```

572 app.SiPMulatorconnectedLamp_2 = uilamp(app.Panel_7);
573 app.SiPMulatorconnectedLamp_2.Position = [203 4 18 18];
574
575 % Create ControlpreloadpulsesPanel
576 app.ControlpreloadpulsesPanel = uipanel(app.Panel_5);
577 app.ControlpreloadpulsesPanel.AutoResizeChildren = 'off';
578 app.ControlpreloadpulsesPanel.Title = 'Control preload pulses';
579 app.ControlpreloadpulsesPanel.FontWeight = 'bold';
580 app.ControlpreloadpulsesPanel.Position = [12 344 410 69];
581
582 % Create SetfolderButton
583 app.SetfolderButton = uibutton(app.ControlpreloadpulsesPanel, 'push');
584 app.SetfolderButton.ButtonPushedFcn = createCallbackFcn(app,
585 @SetfolderButtonPushed, true);
586 app.SetfolderButton.Position = [200 13 88 21];
587 app.SetfolderButton.Text = 'Set folder';
588
589 % Create FolderNameEditFieldLabel
590 app.FolderNameEditFieldLabel = uilabel(app.ControlpreloadpulsesPanel);
591 app.FolderNameEditFieldLabel.HorizontalAlignment = 'right';
592 app.FolderNameEditFieldLabel.Position = [10 12 74 22];
593 app.FolderNameEditFieldLabel.Text = 'Folder Name';
594
595 % Create FolderNameEditField
596 app.FolderNameEditField = uieditfield(app.ControlpreloadpulsesPanel,
597 'text');
598 app.FolderNameEditField.Editable = 'off';
599 app.FolderNameEditField.Position = [85 12 100 22];
600
601 % Create ChangepulseButton
602 app.ChangepulseButton = uibutton(app.ControlpreloadpulsesPanel, 'push');
603 app.ChangepulseButton.ButtonPushedFcn = createCallbackFcn(app,
604 @ChangepulseButtonPushed, true);
605 app.ChangepulseButton.Position = [298 12 100 23];
606 app.ChangepulseButton.Text = 'Change pulse';
607
608 % Create Panel_6
609 app.Panel_6 = uipanel(app.Panel_5);
610 app.Panel_6.AutoResizeChildren = 'off';
611 app.Panel_6.Position = [14 12 408 247];
612
613 % Create UIAxes
614 app.UIAxes = uiaxes(app.Panel_6);
615 title(app.UIAxes, {'Discretized signal 1.875 Msps'; ''})
616 xlabel(app.UIAxes, 'Time (μs)')
617 ylabel(app.UIAxes, 'Dac value 8 bits')
618 zlabel(app.UIAxes, 'Z')
619 app.UIAxes.FontName = 'Times New Roman';
620 app.UIAxes.Box = 'on';
621 app.UIAxes.FontSize = 11;
622 app.UIAxes.Position = [22 12 367 223];
623
624 % Create Image
625 app.Image = uiimage(app.Panel_3);
626 app.Image.Position = [23 578 149 29];
627 app.Image.ImageSource = fullfile(pathToMLAPP, '5cm_SiPM.png');
628
629 % Create Image2
630 app.Image2 = uiimage(app.Panel_3);
631 app.Image2.Position = [343 559 70 67];
632 app.Image2.ImageSource = fullfile(pathToMLAPP, '4cm-DUNE.png');
633
634 % Create FINISHButton
635 app.FINISHButton = uibutton(app.Panel_3, 'push');
636 app.FINISHButton.ButtonPushedFcn = createCallbackFcn(app,
637 @FINISHButtonPushed, true);
638 app.FINISHButton.Position = [321 20 97 21];
639 app.FINISHButton.Text = 'FINISH';
640
641 % Create Image3
642 app.Image3 = uiimage(app.Panel_3);
643 app.Image3.Position = [24 3 134 61];
644 app.Image3.ImageSource = fullfile(pathToMLAPP, '5cm-UAN.jpeg');

```

```
641      % Show the figure after all components are created
642      app.SiPMulatorAppv10UIFigure.Visible = 'on';
643    end
644  end
645
646  % App creation and deletion
647  methods (Access = public)
648
649    % Construct app
650    function app = SiPMulator
651
652      % Create UIFigure and components
653      createComponents(app)
654
655      % Register the app with App Designer
656      registerApp(app, app.SiPMulatorAppv10UIFigure)
657
658      % Execute the startup function
659      runStartupFcn(app, @startupFcn)
660
661      if nargout == 0
662        clear app
663      end
664    end
665
666    % Code that executes before app deletion
667    function delete(app)
668
669      % Delete UIFigure when app is deleted
670      delete(app.SiPMulatorAppv10UIFigure)
671    end
672  end
673 end
674
```

```

1 function plot_pulse_32_csv(csv_path)
2 % plot_pulse_32_csv Grafica 32 muestras asumiendo Fs = 1.875 Msps.
3 % USO:
4 %   plot_pulse_32_csv('pulso_32.csv')
5
6 %--- Parámetros ---
7 Fs = 1.875e6;           % Hz
8 Ts = 1 / Fs;            % s
9 N_expected = 32;
10
11 %--- Leer CSV (sin encabezado; fila o columna) ---
12 y = readmatrix(csv_path);
13 y = y(:); % vector columna
14
15 if numel(y) ~= N_expected
16     error('Se esperaban %d muestras; el archivo tiene %d.', N_expected, numel(y));
17 end
18
19 %--- Eje de tiempo en microsegundos ---
20 t_us = (0:N_expected-1)' * Ts * 1e6;    % μs
21 ventana_us = N_expected * Ts * 1e6;
22
23 %--- Gráfica ---
24 figure('Color','w');
25 plot(t_us, y, '-o', 'LineWidth', 1); grid on;
26 title(sprintf('Pulso (32 puntos a 1.875 Msps) | Ventana: %.3f \\\mu s', ventana_us));
27 xlabel('Tiempo [\mu s]');
28 ylabel('Valor [LSB]');
29 xlim([t_us(1) t_us(end)]);
30
31 end

```

```

1 classdef SerialSession < handle
2     properties
3         sp serialport = []
4         port string = ""
5         baud double = 112500
6         isDetected logical = false
7     end
8     methods
9         function obj = SerialSession(port, baud)
10            if nargin>0, obj.port = string(port); end
11            if nargin>1, obj.baud = baud; end
12        end
13        function connect(obj)
14            if ~isempty(obj.sp) && isvalid(obj.sp), return; end
15            obj.sp = serialport(obj.port, obj.baud, "Timeout", 1);
16            configureTerminator(obj.sp,"none");
17            flush(obj.sp);
18        end
19        function ok = detect(obj)
20            ok = false; obj.connect();
21            flush(obj.sp); write(obj.sp,uint8(170),"uint8"); pause(0.05);
22            if obj.sp.NumBytesAvailable>=1
23                ok = read(obj.sp,1,"uint8")==170;
24            end
25            obj.isDetected = ok;
26        end
27        function writeByte(obj,b), write(obj.sp,uint8(b),"uint8"); end
28        function close(obj)
29            if ~isempty(obj.sp) && isvalid(obj.sp), delete(obj.sp); end
30            obj.sp = []; obj.isDetected=false;
31        end
32        function delete(obj), obj.close(); end
33    end
34
35 end

```

```

1 function sipm_pulse_to_32(in_csv, out_csv, opts)
2 % sipm_pulse_to_32 Lee un CSV con ~12-15 muestras de un pulso y genera 32 muestras
3 % 8-bit tipo SiPM.
4 %
5 % USO:
6 %   sipm_pulse_to_32('entrada.csv','salida_32.csv');
7 %   sipm_pulse_to_32('entrada.csv','salida_32.csv',
8 % struct('peakTarget',190,'baselineTarget',128));
9 %
10 % ENTRADAS:
11 %   in_csv    : ruta del CSV de entrada (1 fila o 1 columna; solo números).
12 %   out_csv   : ruta del CSV de salida (32 enteros 0..255 en una columna).
13 %   opts      : (opcional) estructura con:
14 %     .baselineMethod 'auto'|'first'|'median' (default 'auto')
15 %     .baselineTarget valor 8-bit para baseline (default 128)
16 %     .peakTarget    pico deseado (default 190)
17 %     .minHeadroom   margen por arriba/abajo en LSB (default 2)
18 %     .shapeMode     'fitexp'|'interp' (default 'fitexp')
19 %     .smoothWin     ventana media móvil en muestras (default 1 = sin
20 % suavizado)
21 %
22 % NOTAS:
23 %   - 'fitexp' ajusta una cola exponencial para "SiPM-izar" la forma.
24 %   - Si la cola estimada es inestable, cae a 'interp' automáticamente.
25 %   - La salida mantiene baseline=baselineTarget y recorta [0,255].
26
27 if nargin < 3, opts = struct; end
28 opts = filldefaults(opts, struct( ...
29     'baselineMethod','auto', ...
30     'baselineTarget',128, ...
31     'peakTarget',190, ...
32     'minHeadroom',2, ...
33     'shapeMode','fitexp', ...
34     'smoothWin',1));
35
36 % ----- 1) Cargar y vectorizar -----
37 x = readmatrix(in_csv);
38 x = x(:); % vector columna
39 x = x(~isnan(x)); % quitar NaN
40 assert(numel(x) >= 8, 'Se esperaban al menos ~8 muestras en el CSV.');
41
42 % ----- 2) Suavizado opcional -----
43 if opts.smoothWin > 1
44     x = movmean(x, opts.smoothWin);
45 end
46
47 % ----- 3) Baseline -----
48 switch lower(opts.baselineMethod)
49     case 'first'
50         b = x(1);
51     case 'median'
52         % mediana global, robusta a outliers
53         b = median(x);
54     otherwise % 'auto': usa zonas "planas" al inicio/fin si existen
55         n = numel(x);
56         k = max(2, round(0.2*n)); % 20% al inicio/fin
57         cand = [x(1:k); x(end-k+1:end)];
58         b = median(cand);
59 end
60
61 % ----- 4) Detectar pico y segmento de decaimiento -----
62 [~, ipk] = max(x);
63 % Busca la vuelta a baseline ( $\pm 10\%$  de la amplitud) tras el pico
64 amp = max(x) - b;
65 th = b + 0.1*amp;
66 ireturn = find(x(ipk:end) <= th, 1, 'first');
67 if isempty(ireturn), ireturn = numel(x)-ipk+1; end
68 iend = ipk + ireturn - 1;
69
70 % ----- 5) Normalizar alrededor de baseline -----
71 y = x - b; % baseline → 0
72 % separa subida (0..ipk) y decaimiento (ipk..end)

```

```

71 % ----- 6) Elegir modo de forma: ajustar cola exponencial o interpolar -----
72 Nout = 32;
73 t_in = (0:numel(y)-1).'; % indice como tiempo relativo (paso = 1)
74
75 if strcmpi(opts.shapeMode,'fitexp')
76     % Ajuste exponencial simple en la cola para forma SiPM:  $y \approx A \cdot \exp(-(t-t_0)/\tau)$ 
77     t_tail = (ipk:numel(y)).';
78     tail = y(t_tail);
79     tail_pos = tail - min(0, min(tail)); % desplazar por si hay negativos pequeños
80     tail_pos = max(tail_pos, eps);
81     try
82         % Ajuste lineal en log:  $\ln(\text{tail}) \approx \ln(A) - (t/\tau)$ 
83         L = log(tail_pos);
84         T = t_tail - t_tail(1);
85         p = polyfit(T, L, 1);
86         tau = -1/p(1);
87         A = exp(p(2));
88         if ~isfinite(tau) || tau <= 0
89             error('tau_no_valida');
90         end
91
92     % Reconstrucción: subida por interpolación suave + cola exponencial ajustada
93     % Subida: desde max(1, ipk-3) hasta ipk con PCHIP
94     t_up = (1:ipk).';
95     y_up = y(t_up);
96     % Cola: desde ipk hasta completar Nout
97     t_all = (0:Nout-1).';
98     y_rec = zeros(Nout,1);
99     % Subida por pchip sobre datos de subida
100    y_rec(1:ipk) = pchip(t_up-1, y_up, (0:ipk-1).');
101    % Cola exponencial anclada en el pico
102    t0 = ipk-1;
103    A0 = max(y(ipk), 0);
104    for k = ipk:Nout
105        tk = (k-1) - t0;
106        y_rec(k) = A0 * exp(-tk / tau);
107    end
108    y_proc = y_rec;
109
110 catch
111     % Falla → degradar a interp PCHIP en todo el pulso
112     y_proc = pchip(t_in, y, linspace(0, numel(y)-1, Nout).');
113 end
114 else
115     % Solo remuestrear suavemente a 32 puntos (shape-preserving)
116     y_proc = pchip(t_in, y, linspace(0, numel(y)-1, Nout).');
117 end
118
119 % ----- 7) Re-anclar baseline y dar "idea" de los 32 del constant -----
120 % - baseline plano en la zona final si la cola ya es ~cero
121 % - aseguramos que el pico esté en la vecindad de opts.peakTarget
122 % Recolocar baseline en destino:
123 y_proc = y_proc + opts.baselineTarget;
124
125 % Forzar tramo final a baseline si ya cayó lo suficiente:
126 post_idx = round(0.6*Nout); % a partir de ~60% de la ventana
127 epsLSB = 1.5; % tolerancia en LSB
128 tail_near = abs(y_proc(post_idx:end) - opts.baselineTarget) <= epsLSB;
129 if any(tail_near)
130     first_flat = post_idx + find(tail_near,1,'first') - 1;
131     y_proc(first_flat:end) = opts.baselineTarget;
132 end
133
134 % ----- 8) Escalado a pico objetivo y límites 8-bit -----
135 % Ajustar amplitud (solo por encima del baseline):
136 above = y_proc - opts.baselineTarget;
137 peak_current = max(above);
138 if peak_current < opts.minHeadroom
139     peak_current = opts.minHeadroom;
140 end
141 scale = (opts.peakTarget - opts.baselineTarget) / peak_current;
142 above = above * scale;
143 y8 = opts.baselineTarget + above;

```

```

144
145 % Limitar y redondear
146 y8 = round(y8);
147 y8 = max(0, min(255, y8));
148
149 % Garantizar que el primer y último valor sean exactamente baseline (estilo LUT dada)
150 y8(1) = opts.baselineTarget;
151 y8(end) = opts.baselineTarget;
152
153 % ----- 9) Escribir CSV de salida (una columna, sin encabezado) -----
154 writematrix(y8(:), out_csv);
155
156 % ----- 10) Mensaje resumen -----
157 fprintf('OK: %s -> %s\n', in_csv, out_csv);
158 fprintf('baseline estimado: %.3f | baselineTarget: %d | picoTarget: %d\n', ...
159     b, opts.baselineTarget, opts.peakTarget);
160 fprintf('pico salida: %d | min salida: %d | max salida: %d\n', ...
161     max(y8), min(y8), max(y8));
162
163
164 % ===== utilidades =====
165 function o = filldefaults(o, d)
166 fn = fieldnames(d);
167 for k=1:numel(fn)
168     if ~isfield(o, fn{k}) || isempty(o.(fn{k}))
169         o.(fn{k}) = d.(fn{k});
170     end
171 end
172
173

```